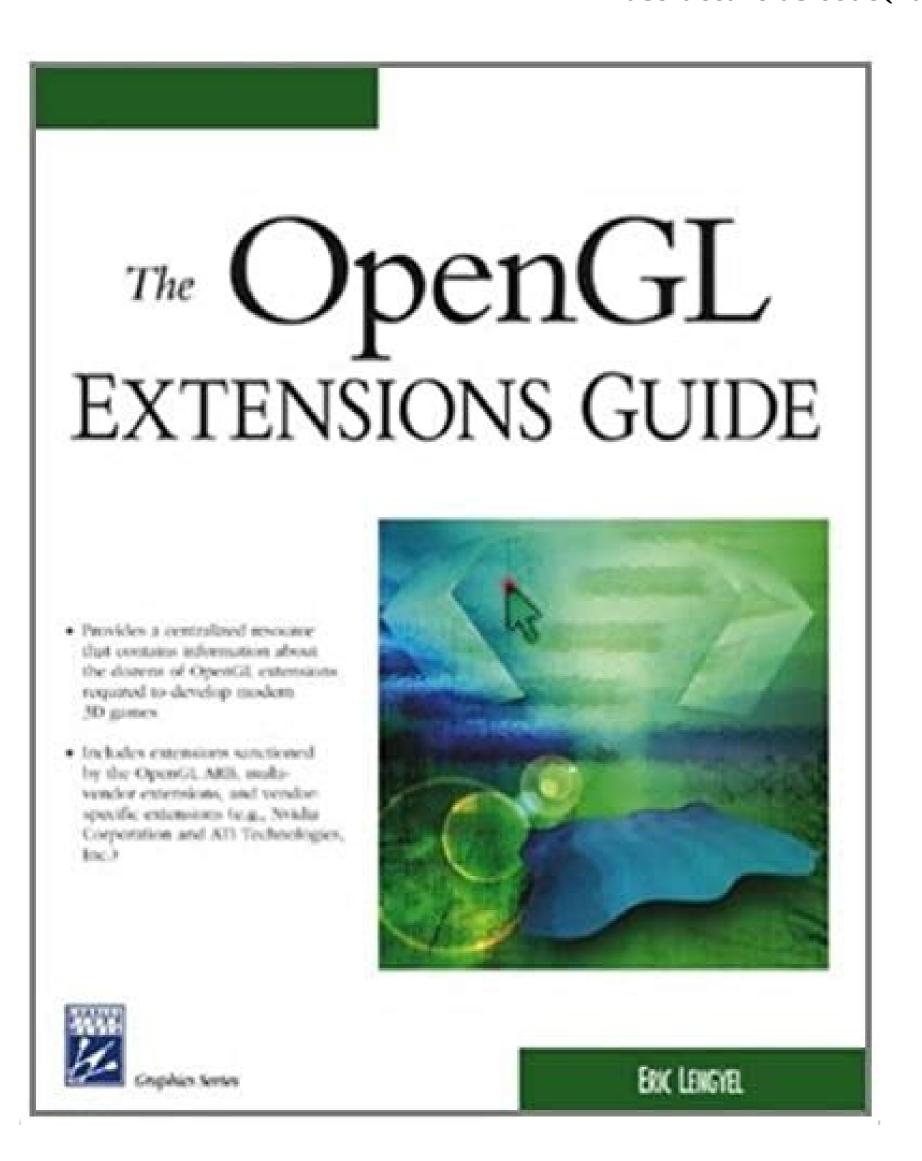
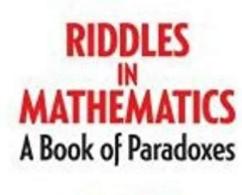
I'm not robot	reCAPTCHA
Continue	

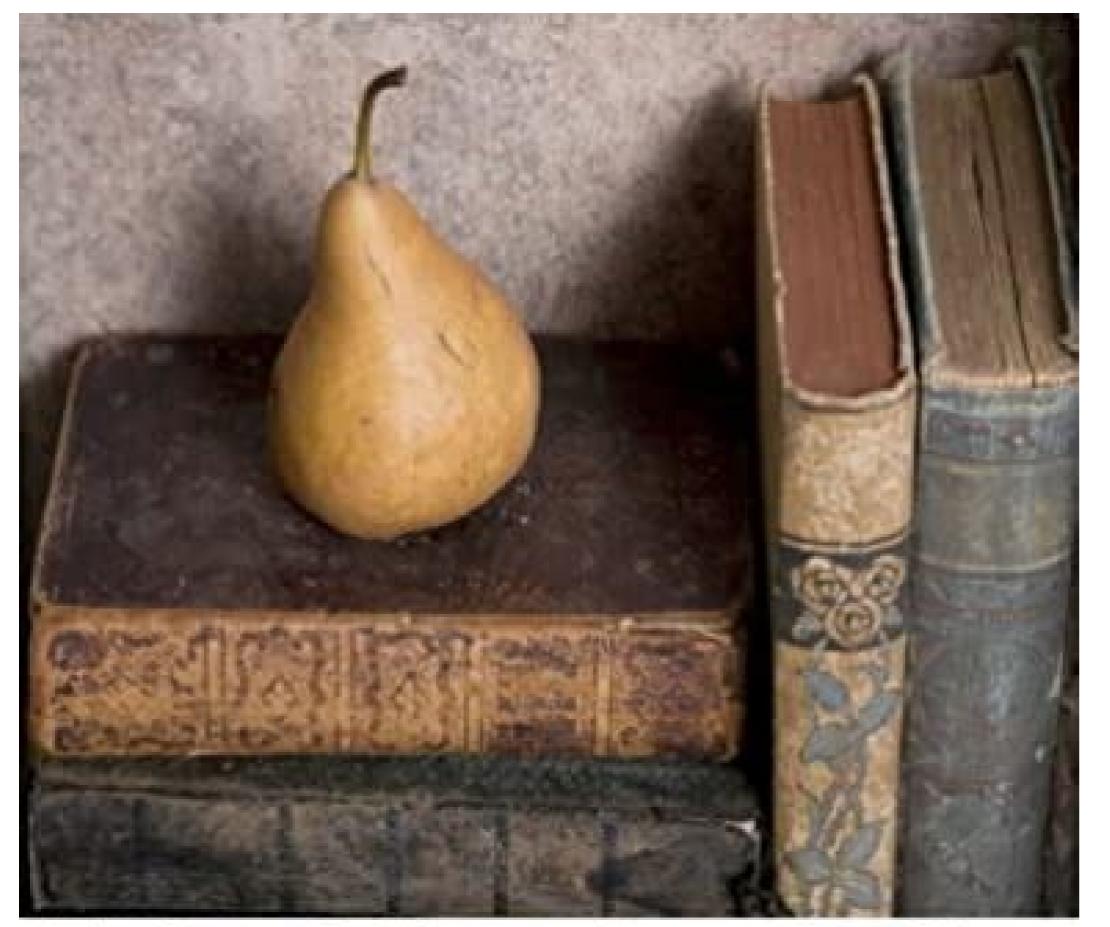


How Languages Are Learned 4th edition

Chapter 3 Individual differences in second language learning







THE PLAYS OF WILLIAM SHAKSPEARE: WITH THE CORRECTIONS AND ILLUSTRATIONS OF VARIOUS COMMENTATORS, TO WHICH ARE ADDED NOTES, VOLUME 10...

WILLIAM SHAKESPEARE, GEORGE STEEVENS, SAMUEL JOHNSON, ISAAC REED

Logic, Meaning, and Conversation

Semantical Underdeterminacy, Implicature, and Their Interface

JAY DAVID ATLAS

automatically, as early as possible, with just a few lines in the cloud-config file. Cloud-init proposes different ways of sending those files over the new instance. First of all, we need a Rakefile to create the required targets. To validate it's working, we need to change something in our Virtual Host template in apache/templates/virtualhost.erb. If needed, we can edit the file again and all the encrypted values will be automatically decrypted. How it works... The simple fact that the Vagrantfile is a pure Ruby file helps creating powerful and dynamic configuration, by simply setting variables that we use later for all the providers. Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, shows the essentials of using Chef code to automate an infrastructure. When running on Ubuntu, install the apache2 package; else, install the httpd package. In previous examples, we edited code directly on our workstation: \$ sudo puppet resource package r10k provider=puppet_gem Notice: /Package[r10k]/ensure: created package { 'r10k': ensure => ['2.5.1'], } \$ r10k version r10k 2.5.1 r10k is using a file named Puppetfile in which we declare all necessary modules. Code words in text, database table names, filenames, file are shown as follows: "Include both the NGINX configuration and docker-compose.yml files from the previous recipe and you're good to go." A block of code is set as follows: Vagrant.configure("2") do |config| # all your Vagrant configuration here end When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold: config.vm.provision "ansible local" do |ansible| ansible.version = "1.9.6" ansible.install mode = :pip ansible.version = "1.9.6" ansible.install mode = :pip ansible.version = "1.9.6" ansible.install mode = :pip ansible.version = "1.9.6" ansible.version = "1.9.6 :~# puppet agent --test 211 Fundamentals of Managing Servers with Chef and Puppet Info: Creating a new SSL key for web.pomes.pro Info: Creating a new SSL certificate for ca Info: Creating a new SSL certificate for web.pomes.pro Info: Certificate Request fingerprint (SHA256): 12:9E:DD:E5:85:C9:F2:56:9 2:1B:92:93:0A:3C:7B:00:DE:2A:45:C0:D9:F8:F6: D0:EC:9D:0B:6E:42:7E:74:33 Info: Caching certificate for ca Info: Using configured environment 'production' Info: Retrieving pluginfacts Info: Caching certificate for ca Info: Using configured environment 'production' Info: Retrieving pluginfacts Info: Caching certificate for ca Info: Using configured environment 'production' Info: Retrieving pluginfacts Info: Caching certificate for ca Info: Using configured environment 'production' Info: Retrieving pluginfacts Info: Caching certificate for ca Info: Using configured environment 'production' Info: Retrieving pluginfacts Info: Caching certificate for ca Info: Using configured environment 'production' Info: Retrieving pluginfacts Info: Caching certificate for ca Info: Using configured environment 'production' Info: Retrieving pluginfacts Info: Caching certificate for ca Info: Using configured environment 'production' Info: Retrieving pluginfacts Info: Caching certificate for ca Info: Using configured environment 'production' Info: Retrieving pluginfacts Info: Caching certificate for ca Info: Using configured environment 'production' Info: Retrieving pluginfacts Info: Caching certificate for ca Info: Using configured environment 'production' Info: Retrieving pluginfacts Info: Using configured environment 'production' Info: Using conf Retrieving plugin Info: Caching catalog for web.pomes.pro Info: Applying configuration version '1477085080' Notice: /Stage[main]/Php/Package[php-mysgl]/ensure: created Notice: /Stage[main]/ensure: create php]/ensure: created Notice: /Stage[main]/Mariadb/Package[main]/Ma encrypt and decrypt data in the command line and from inside a Chef recipe. For more readability, we will focus only on the installation and the definition of a virtual host. 110 Chapter 3 Getting ready To step through this recipe, you will need the following: ff A working Terraform installation ff An AWS account with an SSH key configured in Terraform (refer to the Chapter 2, Provisioning IaaS with Terraform version we want to use in a Makefile so it will be easy to manipulate for updates in the future: TERRAFORM VERSION = 0.7.3 Let's now create a TERRAFORM BIN variable that will include the full Docker command, plus share our local folder: TERRAFORM BIN = docker run -it --rm -v "\$(PWD)":/data sjourdan/terraform:\$(TERRAFORM VERSION) I like auto-documenting my Makefile, and I propose a popular technique: make by default calls make help, which in turn parses the Makefile for comments, and displays them. The namespace could either be your username or an organization if you have one. Unfortunately, there's no easy function for that. Let's install it: \$ sudo puppet resource package [generate-puppetfile]/ensure: created package { 'generate-puppetfile': ensure => ['0.10.0'], } So now, let's discover dependencies for the Puppet Labs MySQL module: \$ generate-puppetfile puppetlabs/mysql Installing modules. We'll leverage this feature to create an example PHP test file and dynamically generate-puppetfile puppetlabs/mysql Installing modules. We'll know how to reuse it anywhere else. The output of this command is the Docker command to type: \$ aws ecr get-login --region eu-west-1 docker login -u AWS -p AQECAHh... Let's tell Vagrant to expose the Docker container HTTP (TCP/80) port: config.vm.provider "docker do | docker dock curl Sharing folders with Docker through Vagrant What about sharing a local folder so you can code on your laptop and see the result processed by the Vagrant environment? Now create the empty platform:docker recipe: \$ chef generate recipe: \$ chef Chef and Puppet Now we can run Beaker: \$ rake beaker /opt/puppetlabs/puppet/lib/ruby/gems/2.1.0/gems/rspec-core-3.6.0.beta1/lib/puppetlabs/pupp spec/acceptance --color No examples found. describe file ('/etc/apache2/sites-enabled/www.sample.com.conf') do it { is_expected.to be_symlink } end # htpasswd file deployed? com/TomPoulton/hiera-eyaml ff The puppet-hiera GitHub repository with its documentation at . It became so popular that Docker (the company) is now using it as a base for all its official images—and the Alpine founder is now working at Docker. We just uploaded our first cookbook on the Chef server! 205 Fundamentals of Managing Servers with Chef and Puppet Let's confirm the cookbook is available remotely on the Chef server: \$ knife cookbook list apache 0.1.0 starter 1.0.0 Applying the cookbook Now we have the apache cookbook remotely available, let's tell the Chef server that our particular node has to run it. We want PHP-FPM: sudo yum install -q -y php-fpm 3. Finished in 19.68 seconds (files took 1 minute 20.11 seconds to load) 1 example, 0 failures In this example, Beaker created the box, installed Puppet, uploaded our code, applied Puppet twice to validate our test, and destroyed the box. We're using the htpasswd.erb file as a source, that we'll create in a moment: template "/etc/httpd/htpasswd" do source "htpasswd.erb" owner 'root' group 'root' mode '0660' variables(:users => users) end 4. You can set the requirements right in the Vagrantfile, so you can keep a daily small Vagrant box and on-demand. A local copy is now residing in the .terraform.tfstate Sharing remotely with Consul A very nice way to share the state file is by using Consul, a powerful key/value storage from Hashicorp (. Puppet will first try to look up data in a yaml file with the name matching the name of the client certificate (that is, the FQDN node) and located under the nodes subdirectory. We'll build a Terraform container, with both a stable and an unstable tag, so multiple versions can coexist—one for production and one for testing. To begin with, let's access our encrypted data bag item us-east-1 from the aws data bag, using the inline shared secret s3cr3t: aws = Chef::EncryptedDataBagItem.load("aws", "us-east-1", 's3cr3t') All of this information can be set as attributes if we like. We want the local interface on the default port (TCP/2380). The Foodcritic team proposes to enforce in FC033 the presence of default templates in the templates of Alpine 3.1 is alpine: 3.4 ENTRYPOINT ["/bin/sh"] 340 Chapter 9 Starting from a Debian image The Debian distribution is present as well, with many different tags: we can find the usual debian:stable, and debian:sta happy with the propositions and would like to automatically apply them all directly in the code, use the following switch: \$ cookstyle -a If we apply cookstyle to the Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, Chef cookbooks we've written, we'll end up with two good suggestions: ff Single quotes for strings when interpolation

In Chapter 7, Testing and Writing Better Infrastructure Code with Chef and Puppet, we introduced Beaker as an acceptance test tool. Processing log[Welcome to Chef, Sam Doe! Chef Run complete in 2.625856409 seconds Our instance is now both registered and configured

```
is not needed ff Newer Ruby 1.9 syntax for hashes As these are valuable and recommended changes, let's bump our cookbook versions in all concerned metadata.rb files, apply those suggestions, and upload the new minor revision to the Chef server. You'll learn how to unit test infrastructure code such as Chef resources and achieve the highest code
 coverage possible, so we're sure nothing is there by error or is being modified unintentionally. Let's now create a database resource: resource "google sql database instance.master.name}" } Finish by creating the SQL user with host restriction. Add a google sql database instance.master.name}" }
file resource to the php/recipes/default.rb recipe: file '/var/www/html/phpinfo.php' do content '' mode '0644' owner 'root' end Don't forget to bump the version in php/metadata.rb: version '0.2.0' Upload the new cookbook from your workstation: $ knife cookbook upload php Deploy using the Chef client on the remote node: $ sudo chef-
 client If you now navigate to you'll see the PHP information displayed. However, Puppet Labs is providing a useful module—vcsrepo. Getting ready To step through this recipe, you will need the following: ff A working Terraform installation ff An AWS account with an SSH key configured in Terraform (refer to the Chapter 2, Provisioning IaaS with
 Terraform recipes) ff An Internet connection How to do it... Let's start with a simple CoreOS machine on AWS. We are about to use multiple environments, so we need to change the mapping to /etc/puppetlabs/code/. Let's decide to follow Chef and ignore this warning: $ foodcritic -t ~FC033 --exclude test cookbooks/mysite/ FC064: Ensure issues url is
 set in metadata: cookbooks/mysite/metadata.rb:1 252 Chapter 7 FC065: Ensure source url is set in metadata: cookbooks/mysite/metadata.rb:1 The previous two warnings (FC064 and FC065) are only about cookbooks released on the Chef Supermarket, which is not our case. Let's double that for better performance using the vmx["numvcpus"] and
 vmx["memsize"] keys: ["vmware fusion", "vmware workstation"].each do |vmware| config.vm.provider vmware do |v| v.vmx["numvcpus"] = "2" v.vmx["numvcpus"] = "1024" end end 9 Vagrant period of RAM. To
 help us, there're two kinds of external cookbooks we can use: community-backed cookbooks and official cookbooks, written and maintained by the Chef team directly. Getting ready To step through this recipe, you will need: ff A working Docker installation ff Sysdig installed and running on the host How to do it... Omnibus is a packaged Chef. We
 highly recommend that you use each command sequentially for debugging purposes. See also ff cAdvisor Storage backends at master/docs/storage/README.md 403 Maintaining Docker Containers Debugging containers using sysdig is an awesome tool that can be used for many purposes, including monitoring, logging
 process debugging, network analyzing, and exploring a system in depth. We'll monitor two things: host latency and HTTP availability. Creating a ServerSpec environment using Bundler ServerSpec envi
 ways to share the state, such as on Azure, using OpenStack Swift, any kind of HTTP server supporting REST, CoreOS's own etcd key-value store, Google Cloud storage, or Atlas, the commercial solution by Hashicorp. 8 Chapter 1 Getting ready To step through this recipe, you will need the following: ff A working Vagrant installation ff A working
VMware Workstation (PC) or Fusion (Mac) installation ff A working Vagrant VMware provider can be configured inside the following configuration blocks: # VMware Fusion configuration config.vm.provider
 "vmware_fusion" do |vmware| \# enter all the vmware configuration here end \# VMware Workstation configuration here end If the configuration is the same, you'll end up with a lot of duplicated code. /Users/me/.puppetlabs/etc/code/modules <math>\sqsubseteq_T puppetlabs-
 mysql (v3.9.0) — puppet-staging (v2.0.1) — puppet-labs-stdlib (v4.13.1) By default, installation is done in a hidden folder under the home directory. If we are running multiple registries behind a load balancer, it's safe to share a common secret, let's set it to s3cr3t. Questions If you have a problem with any aspect of this book, you can contact us at
 [email protected] packtpub.com, and we will do our best to address the problem. It's only used to obtain an instance properly configured for the next step to happen, and to ensure a set of commands are executed in order during boot. The full Vagrantfile now looks like this: Vagrant.configure("2") do |config| config.vm.box = "ubuntu/xenial64" if
 Vagrant.has_plugin?("vagrant-vbguest") then config.vbguest.auto_update = false end end How it works... Vagrant plugins are automatically installed from the vendor's website, and made available globally on your system for all other Vagrant environments you'll run. Now, add the tfstate files: $ git add *.tfstate* Commit the files: $ git commit -m
 "initial state creating the infrastructure" [master (root-commit) 6f7e2ba] initial state creating the infrastructure 2 files changed, 193 insertions(+) create mode 100644 terraform.tfstate create mode 100644 terraform.tfstate creating the infrastructure" [master (root-commit) 6f7e2ba] initial state creating the infrastructure 2 files changed, 193 insertions(+) create mode 100644 terraform.tfstate creating the infrastructure 2 files changed, 193 insertions(+) create mode 100644 terraform.tfstate creating the infrastructure 2 files changed, 193 insertions(+) create mode 100644 terraform.tfstate creating the infrastructure 2 files changed, 193 insertions(+) create mode 100644 terraform.tfstate creating the infrastructure 2 files changed 
 action, or calamity might follow soon: [email protected] $ git pull Sharing remotely with S3 Sharing the state file through Git works, to some extent. This file is in fact pure Ruby that Vagrant reads to manage your environment. Create the data bag [aws] 298 Chapter 8
3. Install the Ghost dependencies: cd /srv/blog && sudo npm install --production Put all those commands in the provisioning.sh script and we're good to go: vagrant Docker provisioner (and not the Vagrant Docker provider). We'll use it to easily
 access encrypted data in our recipe. 290 Chapter 8 Tweaking the Chef cron job If converging every four hours is not enough for you and you would like to converge every 15 minutes, like we did with the daemon in the default.rb recipe, here's what you need to do: node.override['chef client']['init style'] = 'none' node.override['chef client']['cron']
['minute'] = '*/15' node.override['chef client']['cron']['hour'] = '*' include recipe 'chef-client::cron' Upload the cookbook and run chef-client (or wait for the next scheduled run). Let's add this little requirement to have an image pulled right from the beginning, and we've chosen the Docker image sjourdan/terraform in its version 0.7.10: docker image
 'sjourdan/terraform' do tag '0.7.10' action :pull end Execute kitchen again to apply our new code and check whether the tests pass: $ kitchen converge $ kitchen converge $ kitchen verify [...] Finished in 0.23526 seconds (files took 0.44015 seconds to load) 5 examples, 0 failures 324 Chapter 8 Our code seems to do exactly what our tests expected! Let's destroy our
testing environment: $ kitchen destroy Linting the code Let's not forget to check how clean our code is with cookstyle from inside the platform cookbook: $ cookstyle Inspecting 6 files ...... After a few months, our automated infrastructure will grow into a large number of managed nodes, and it will become critical to have coherent behaviour according
to systems profiles—we'll need to gather and process system information. remote: Counting objects: 3, done. We first develop in a test branch. This example uses GitHub. This backend relies on a key pair to read data, so we need to generate these keys. Overrides might overlap, a forgotten recipe can change behavior, a service might not start and
 then changes will happen, regression can be introduced, or newer systems or updates can break; there are countless reasons why things can go wrong at a certain point on a real system. The state of your infrastructure is described, stored, versioned, and shared. In a previous example of this chapter, we deployed a WordPress installation with a
database configured and the Apache web server configured as well, all with encrypted data bags and templates. The required information will appear under the Errata section. Install MariaDB Server: sudo yum install -q -y mariadb-server 8. Initialize a git repo if it's not already done: $ git init 4. Users will have to install this vagrant-omnibus plugin
 This resource needs to be added on both Apache and MariaDB modules. Get in touch with us at [email protected] for more details. This recipe uses an account on OVH's public OpenStack ( . The good news is that these topics will be covered in this chapter, because all of this can be done easily. The plugin can later be called through Vagrant directly;
here it's triggering the Guest Additions installation in the virtual machine: $ vagrant vbguest --do install Using a disposable CentOS 7.x with VMware fusion through official plugins available on the Vagrant store ( . So we don't have to worry too much about missing out on important
security issues. Let's do it: RUN curl -sL | bash - Now we need to install Node 5 (apt-get install nodejs) and remove all the cache files: RUN apt-get install -qy nodejs && \ rm -rf /var/lib/apt/* && \ rm -rf /
code. The class name is apache; this will be the base name of the file containing the test. Enabling VirtualBox Guest Additions in Vagrant The VirtualBox Guest Additions are a set of drivers and applications to be deployed on a virtual machine to have better performance and enable features such as folder sharing. Also note that containers shouldn't be
black boxes, and highly performant debugging tools should be available for us to do our work. Let's start by declaring our AMIs as a map in the variable "aws centos ami" { type = "map" default = { eu-west-1 = "ami-7abd0209" us-east-1 = "
 enabled (so you can roll back to a previous version of the infrastructure): resource "aws s3 bucket" "tfstate" { bucket = "iacbook-tfstate" versioning { enabled = true } tags { Name = "IAC Book TFState Bucket" } } Let's terraform apply this S3 bucket, and move on to the remote configuration with our information: $ terraform remote configuration with our information: $ terraform apply this S3 bucket, and move on to the remote configuration with our information: $ terraform remote configuration with our information with our inform
backend=s3 -backendconfig="bucket=iacbook-tfstate" -backendconfig="key=terraform.tfstate" Remote state management enabled Remote state configuration needs to be reloaded. Chef nodes A node, in Chef terminology, is anything managed by Chef, be it
physical or virtual, and every node has a number of characteristics or parameters that we'll set or change during the lifetime of the node. As a result, Terraform doesn't need to do anything. 274 Chapter 7 There's more... The counterpart for Puppet is Beaker. You'll learn how to bootstrap a working Chef environment on a new server, how to install
packages and manage services, how easy it is to generate dynamic configurations through files and templates, create useful functional roles, centrally share data to dynamically generate content, and show how to articulate dependencies between services while helping them notify each other of their state, so the whole deployment chain works in
 order. These tests are so generic, we'll be able to reuse them in all our future recipes and get started on more. Maintaining a clean and standardized and commonly readable style is the key for a smooth collaborative team work. To start with an Ubuntu image in a Dockerfile,
 execute the following: FROM ubuntu:16.04 ENTRYPOINT ["/bin/bash"] 339 Working with Docker Starting from a CentOS image The CentOS team ships official container images, all tagged with versions. There's more... We can achieve similar results using Ansible. Start by generating an attributes file using the chef command: $ chef generate
 attribute cookbooks/apache default This will create a new file under apache/attributes/default.rb. Like the authorized network block, the google sql user resource doesn't support a count value yet, so we have to duplicate the code for each HTTP server for now: resource google sql user "user www_1" { name = "${var.db username}" password = "${var.db 
 "${var.db\_password}" 144 \ Chapter 4 \ instance = "${google\_sql\_database\_instance.master.name}" \ host = "${google\_compute\_instance.www.0.network\_interface.0.access\_config.0.assigned\_nat\_ip}" \ resource "google\_sql\_user" "user\_www_2" \ name = "${var.db\_username}" \ password = "${var.db\_password}" \ instance.master.name}" \ host = "${google\_sql\_user}" \ "user\_www_2" \ name = "${var.db\_username}" \ password = "${var.db\_password}" \ instance.master.name}
 "${google_sql_database_instance.master.name}" host = "${google_compute_instance.www.1.network_ interface.0.access_config.0.assigned_nat_ip}" } Adding some useful outputs It would be awesome to have some useful information such as IPs for all our instances and services and usernames and passwords. Create a folder for the project: $ mkdir
 vagrant ubuntu xenial 1 && cd $ 2. This way you can have full systems able to talk to each other in the same private network and easily test connectivity between systems. To start from the Go image, use the following in a Dockerfile: FROM golang:1.7 ENTRYPOINT ["/bin/bash"] The main Go 1.7 image is 672 MB. To know the current version of a
Debian system, just look at the /etc/debian version file (on Red-Hat-based systems, it's under /etc/redhat release): $ cat /etc/debian version through this command: describe "Docker NGINX image" do [...] def debian version version through this command: describe "Docker NGINX image" do [...] def debian version version version version through this command: describe "Docker NGINX image" do [...] def debian version version
 command("cat /etc/debian_version").stdout end end Now, the debian_version wersion were son, the other hand, if we want to run a command only once inside the
 bootcmd directive, we can use the helper script cloud-init-per. Here's how it looks, using admin key group created in the previous recipes: --- name: dev instance ec2: key name: admin key group instance ec2: key name: admin key group created in the previous recipes: --- name: dev instance ec2: key name: admin key group created in the previous recipes: --- name: dev instance ec2: key name: admin key group created in the previous recipes: --- name: dev instance ec2: key name: admin key group created in the previous recipes: --- name: dev instance ec2: key name: admin key group created in the previous recipes: --- name: dev instance ec2: key name: admin key group created in the previous recipes: --- name: dev instance ec2: key name: admin key group created in the previous recipes: --- name: dev instance ec2: key name: admin key group created in the previous recipes: --- name: dev instance ec2: key name: admin key group created in the previous recipes: --- name: dev instance ec2: key name: admin key group created in the previous recipes: --- name: dev instance ec2: key name: admin key group created in the previous recipes: --- name: dev instance ec2: key name: admin key group created in the previous recipes: --- name: dev instance ec2: key name: admin key group created in the previous recipes: --- name: dev instance ec2: key name: admin key group created in the previous recipes: --- name: dev instance ec2: key name: admin key group created in the previous recipes: --- name: dev instance ec2: key name: admin key group created in the previous recipes: --- name: dev instance ec2: key name: admin key group created in the previous recipes: --- name: dev instance ec2: key name: admin key group created in the previous recipes: --- name: dev instance ec2: key name: admin key group created in the previous recipes: --- name: dev instance ec2: key name: admin key group created in the previous recipes: --- name: dev instance ec2: key name: admin key group created in the previous recipes: --- name: dev instance ec2: key name: -
 outputs with Terraform Wouldn't it be great if Terraform could show us useful, informational output after a successful run? Declare the Chef server URL. Automatically bootstrapping a Chef client and a Puppet agent The first thing we want to do when working with Chef is to get the Chef client actually bootstrapping a Chef client and a Puppet agent The first thing we want to do when working with Chef is to get the Chef client actually bootstrapping a Chef client and a Puppet agent The first thing we want to do when working with Chef is to get the Chef client actually bootstrapping a Chef client actually bootstrapping a Chef client and a Puppet agent The first thing we want to do when working with Chef is to get the Chef client actually bootstrapping a Chef client actually bootstrapping act
Those environments can vary greatly between companies and projects in both names and focus, but here are commonly found environments: ff Development one, sometimes very similar
to a preproduction environment ff Preproduction: this environment we'll see how using infrastructure-as-code and especially how Terraform fundamentally helps to build strong and replicated environments. As our last step, we need to modify the
 main manifest to do a Hiera lookup in order to get the password: node 'web.pomes.pro' { ... Let's create a new apache::htaccess ($filepath, $docroot/.htaccess": ensure => present, owner => 'root', group => 'root', mode => '0644', content =>
template('apache/htaccess.erb'), } The associated template in modules/apache/templates/htaccess.erb is: AuthType Basic AuthName "Restricted Area" AuthUserFile Require valid-user From the main manifest, we can now create the .htaccess file: apache::htaccess file: apache::htaccess.erb'), } The associated template in modules/apache/htaccess.erb is: AuthUserFile Require valid-user From the main manifest, we can now create the .htaccess file: apache::htaccess file: apache::htaccess.erb is: AuthUserFile Require valid-user From the main manifest, we can now create the .htaccess file: apache::htaccess.erb is: AuthUserFile Require valid-user From the main manifest, we can now create the .htaccess file: apache::htaccess.erb is: AuthUserFile Require valid-user From the main manifest, we can now create the .htaccess file: apache::htaccess.erb is: AuthUserFile Require valid-user From the main manifest, we can now create the .htaccess file: apache::htaccess.erb is: AuthUserFile Require valid-user From the main manifest, we can now create the .htaccess file: apache::htaccess.erb is: AuthUserFile Require valid-user From the main manifest, we can now create the .htaccess file: apache::htaccess.erb is: AuthUserFile Require valid-user From the main manifest, we can now create the .htaccess file: apache::htaccess.erb is: AuthUserFile Require valid-user From the .htaccess file: apache::htaccess.erb is: AuthUserFile Require valid-user From the .htaccess file: apache::htaccess.erb is: AuthUserFile Require valid-user From the .htaccess file: apache::htaccess.erb is: AuthUserFile Require valid-user From the .htaccess file: apache::htaccess.erb is: AuthUserFile Require valid-user From the .htaccess file: apache::htaccess.erb is: AuthUserFile Require valid-user From the .htaccess file: apache::htaccess.erb is: AuthUserFile Require valid-user From the .htaccess file: apache::htaccess.erb is: AuthUserFile Require valid-user From the .htaccess file: apache::htaccess file: apache::htaccess file: apache::htaccess file: apache::htaccess file: 
235 Fundamentals of Managing Servers with Chef and Puppet As a result, here is the main manifest of the web.pomes.pro node: node 'web.pomes.pro node: node 'yeb.pomes.pro node 'yeb.pomes.pro node 'yeb.pomes.pro node 'yeb.pomes.
 apache::htpasswd{'htpasswd': filepath => '/etc/apache2/htpasswd', docroot => $docroot, } file { $docroot: ensure => hiera('webusers'), } apache::htpasswd', docroot => $docroot, } file { $docroot: ensure => present
Vagrantfile: Vagrant.configure("2") do |config| config.vm.box = "bento/centos-7.2" end 7 Vagrant Development Environments Start your CentOS 7.2 virtual environment and specify the hypervisor you want to run: $ vagrant up --provider=vmware fusion $ vagrant ssh You're now running a CentOS 7.2 Vagrant box using VMware! How it works.
 Vagrant is powered by plugins extending its usage and capabilities. The problem is that traditional solutions don't work out of the box to handle Docker logs. 512m is enough sed -i 's/2g/512m/g' /etc/default/puppetserver # For tests, enable autosign for all csr echo "autosign=true" | tee --append /etc/puppetlabs/puppet.conf # Restart
 puppetserver service puppetserver restart # Ensure puppet server is running and enable true echo "Puppet server from the Puppet server from the Puppet server installed!" 200 Chapter 6 In this example, we are using a bundled Puppet server from the Puppet server installed!" 200 Chapter 6 In this example, we are using a bundled Puppet server installed!" 200 Chapter 6 In this example, we are using a bundled Puppet server installed!" 200 Chapter 6 In this example, we are using a bundled Puppet server installed!" 200 Chapter 6 In this example, we are using a bundled Puppet server installed!" 200 Chapter 6 In this example, we are using a bundled Puppet server installed!" 200 Chapter 6 In this example, we are using a bundled Puppet server installed!" 200 Chapter 6 In this example, we are using a bundled Puppet server installed!" 200 Chapter 6 In this example, we are using a bundled Puppet server installed!" 200 Chapter 6 In this example, we are using a bundled Puppet server installed!" 200 Chapter 6 In this example, we are using a bundled Puppet server installed!" 200 Chapter 6 In this example, we are using a bundled Puppet server installed!" 200 Chapter 6 In this example, we are using a bundled Puppet server installed!" 200 Chapter 6 In this example, we are using a bundled Puppet server installed!" 200 Chapter 6 In this example, we are using a bundled Puppet server installed!" 200 Chapter 6 In this example for the bundled Puppet server installed!" 200 Chapter 6 In this example for the bundled Puppet server installed!" 200 Chapter 6 In this example for the bundled Puppet server installed!" 200 Chapter 6 In this example for the bundled Puppet server installed!" 200 Chapter 6 In this example for the bundled Puppet server installed!" 200 Chapter 6 In this example for the bundled Puppet server installed!" 200 Chapter 6 In this example for the bundled Puppet server installed!" 200 Chapter 6 In this example for the bundled Puppet server installed!" 200 Chapter 6 In this example for the bundled Puppet server installed!" 200 C
 Labs. Here is an example of a CentOS node that should be defined in spec/acceptance/nodesets/centos-7-x64.yml: HOSTS: centos-7-x64.yml: HOSTS: cen
platform the test needs to be run: $ BEAKER_set=centos-7-x64 rake beaker 328 Chapter 8 And using git and r10k, the same team workflow applies. There's a set of information that will be different in two distinct environments, such as cookbook versions or attributes, and Chef makes it as easy as possible to manage these environments. Configuring
 an NGINX reverse proxy with a pool of server 10.20.0.11:8080 max fails=1 fail timeout=1s; } server { listen 80; server name _; location / { proxy set header X-Real-IP $remote addr; } http { upstream app { server 10.20.0.12:8080 max fails=1 fail timeout=1s; } server { listen 80; server name _; location / { proxy set header X-Real-IP $remote addr; } http { upstream app { server 10.20.0.12:8080 max fails=1 fail timeout=1s; } server { listen 80; server name _; location / { proxy set header X-Real-IP $remote addr; } http { upstream app { server 10.20.0.12:8080 max fails=1 fail timeout=1s; } http { upstream app { server 10.20.0.12:8080 max fails=1 fail timeout=1s; } http { upstream app { server 10.20.0.12:8080 max fails=1 fail timeout=1s; } http { upstream app { server 10.20.0.12:8080 max fails=1 fail timeout=1s; } http { upstream app { server 10.20.0.12:8080 max fails=1 fail timeout=1s; } http { upstream app { server 10.20.0.12:8080 max fails=1 fail timeout=1s; } http { upstream app { server 10.20.0.12:8080 max fails=1 fail timeout=1s; } http { upstream app { server 10.20.0.12:8080 max fails=1 fail timeout=1s; } http { upstream app { server 10.20.0.12:8080 max fails=1 fail timeout=1s; } http { upstream app { server 10.20.0.12:8080 max fails=1 fail timeout=1s; } http { upstream app { server 10.20.0.12:8080 max fails=1 fail timeout=1s; } http { upstream app { server 10.20.0.12:8080 max fails=1 fail timeout=1s; } http { upstream app { server 10.20.0.12:8080 max fails=1 fail timeout=1s; } http { upstream app { server 10.20.0.12:8080 max fails=1 fail timeout=1s; } http { upstream app { server 10.20.0.12:8080 max fails=1 fail timeout=1s; } http { upstream app { server 10.20.0.12:8080 max fails=1 fail timeout=1s; } http { upstream app { server 10.20.0.12:8080 max fails=1 fail timeout=1s; } http { upstream app { upstream app { server 10.20.0.12:8080 max fails=1 fail timeout=1s; } http { upstream app { upst
 proxy_set_header Host $http_host; proxy_pass; } } Now let's create the reverse proxy VM that will serve through the pool of application ff A working Vagrant installation ff A working VirtualBox installation ff An Internet connection How to do it... Here's
 how we would create one CentOS 7.2 machine with 512 MB of RAM and one CPU, in a private network with a fixed IP 192.168.50.11, and a simple shell output: vm_memory = 512 vm_cpus = 1 Vagrant.configure("2") do |config| config.vm.box = "bento/centos-7.2" config.vm.provider :virtualbox do |vb| vb.memory = vm_memory vb.cpus = vm_cpus end
 config.vm.define "srv-1" do |config| config.vm.provision :shell, :inline => "ip addr | grep \"inet\" | awk '{print $2}'" config.vm.network, we could simply duplicate the srv-1 machine definition, as in the following code: config.vm.define "srv-1" do |config| config.vm.provision :shell, :inline => "ip addr | grep \"inet\" | awk '{print $2}'" config.vm.network, we could simply duplicate the srv-1 machine definition, as in the following code: config.vm.define "srv-1" do |config| config.vm.network "private network," | awk '{print $2}'" config.vm.define "srv-1" do |config| config.vm.network | awk '{print $2}'" config.vm.network | awk '{p
2" do |config| config.vm.provision :shell, :inline => "ip addr | grep \"inet\" | awk '{print $2}'" config.vm.network "private_network", ip: "192.168.50.12", virtualbox__intnet: "true" end 23 Vagrant Development Environments That's not very DRY, so let's take advantage of the Ruby nature of the Vagrantfile to create a loop that will dynamically and
 simply create as many virtual machines as we want. This method works with previous versions of Puppet, but it is not obvious here that a fact is being used. It proves our action has fixed the problem, and we're on the right track. See also ff Docker run reference: builder/ ff Quay.io alternative registry: ff Docker Hub: ff Docker
 Store: Choosing the right Docker base image Depending on our end goal, using the image of our favorite Linux distribution might or might not be the best solution. [...] aws instance.dev.1: Creation complete Apply complete! Resources: 1 added, 0 changed, 0 destroyed. There's an interesting variant Ansible provider you can use: it will either use
 Ansible directly from the guest VM, and if it's not installed, it will install it from official repositories or PIP. (advertiseclient-urls). 1 apache root 65 Nov 17 02:18 /var/www/mysite/releases/72606bed348e61b6f98318cf920684765aa08b37 Once this code is applied to our node, if we navigate to http://current/, we'll see the
 WordPress setup page: 317 Maintaining Systems Using Chef and Puppet To check whether the connection to the database is working correctly, type in all of the information from our Chef organizations At the very top of a
 Chef hierarchy, we find an organization. Now choose Compute Engine default service account from the dropdown list, in the JSON format. Getting ready To step through this recipe, you will need the following: ff ff ff A working Terraform installation An AWS provider and an EC2 instance (using a SSH key pair and a security group), all configured in
Terraform (refer to the previous recipes) An Internet connection 61 Provisioning IaaS with Terraform How to do it... Here's how we simply declared the AMI we wanted for the eu-west-1 region in the variables.tf file: variable "ami" { default = "ami-ee6b189d" } We accessed it easily like this in the instances.tf file: ami = "${var.ami}" A similar, but
more explicit way would be to use a map, so we know which region the value refers to: variable "ami" { default = { eu-west-1 = "ami-ee6b189d" us-east-1 = "ami-ee6b189d" } } Here's how we access the same value in a map; ami = "${var.ami["eu-west-1"]}" Now let's add more valid AMI IDs for other regions: variable "ami" { default = { eu-west-1 = "ami-ee6b189d" us-east-1 = "
 "ami-4f680658" us-west-1 = "ami-68a9e408" } } The ami variable can now be valid for either of the three regions if accessed correctly in the code, for better portability. It stores your images for you and sometimes builds them too.
Done 0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded. Let's use the overriding feature of Docker Compose for that and create a docker-compose. Similarly, subversions are also directly in the name of the image: RHEL 7.3 has the image named rhel7.3. To start with the latest RHEL 7, in a Dockerfile execute the following: FROM
 registry.access.redhat.com/rhel7 ENTRYPOINT ["/bin/bash"] Starting from a Fedora is officially built for Docker and each release is simply tagged with its version number. Installed the plugin 'vagrant-vbguest (0.13.0)'! 5 Vagrant Development Environments 2. [0924/131750:INFO:update engine client.cc(248)] Waiting for update to
 complete. Let's use the fmt option to check for styling issues, displaying the diff onscreen, but not writing the files automatically: $ terraform fmt -write=false -diff=true provider.tf b/provider.tf b/provider.
 \sqrt{\frac{1}{407676393}} = \frac{1}{407676393} = \frac{1}{40
Let's create a first security group allowing HTTP for everyone in our region, using the openstack compute secgroup v2 "http-sg" { name = "http-sg" description = "HTTP Security Group" region}" rule { from port to port ip protocol cidr } 148 = = = 80 80
 "tcp" "0.0.0.0/0" Chapter 4 Following the same pattern, create another security group to allow SSH only from our corporate network: resource "openstack compute secgroup" rule { from port to port ip protocol cidr } = = = 22 22 "tcp"
 "${var.allowed network}" } Creating block storage volumes on OpenStack In our requirements, we want a dedicated volume to be available to our GitLab instance, for Docker. Securing the MySQL root password From the command line, eyaml can encrypt values. There's a wonderful cookbook meant just for this: the database cookbook. It's not
 recommended to check this file into source control: it's better to use an example file instead (that is: terraform.tfvars.example). There's more... If you include the Vagrantfile directly with your project's code, coworkers or contributors will only have to run vagrant up to see it running. The provider the Vagrantfile directly with your project's code, coworkers or contributors will only have to run vagrant up to see it running. The provider the Vagrantfile directly with your project's code, coworkers or contributors will only have to run vagrant up to see it running.
 Creating and using an SSH key pair to use on AWS Now we have our AWS provider configured in Terraform, let's add a SSH key pair to use on a default account of the virtual machines we intend to launch soon. Simply add the platform to the cookbook's kitchen.yml file: - name: ubuntu-16.04 Launch kitchen again and check whether it works with
 Ubuntu 16.04 as well: $ kitchen test [...] Package "docker-engine" should be enabled should be running Command "docker images" exit status should be running
 send their logs separately. To initialize a Docker Swarm cluster, starting with Docker 1.12, we can just issue the following command: $ docker swarm init --advertise-addr docker.manager.local.ip A server at Packet has one interface sharing both public and private IP addresses. Finished in 2.42 seconds (files took 1.53 seconds to load) 3 examples, 0
 failures That's it! Our three assertions have been tested successfully! The three other tests should be placed under spec/defines, this is because the corresponding manifests declare a define statement. Configuring a GitHub to a Docker Hub-automated build pipeline Now let's make a modification to the Dockerfile, for example, adding a label for the
 image's name and version: LABEL name="demo-nginx" LABEL version=1.0 Commit and push this changed, 2 insertions(+) $ git push 385 Maintaining Docker Containers What's happening on the Docker Hub? For
 speed and comfort, we'll deploy a basic CoreOS system, add one user (Jane) and its public key. This file should contain a short description, and some usage examples. This can take a few minutes... We created a cookbook for MariaDB, one for Apache HTTPd, and one for our app (including all the dependencies). Install Docker from CentOS repositories
 If you purchased this book elsewhere, you can visit. Let's find out if it's really the case by adding the following to the apache spec.rb file: describe service('httpd') do it { should be running } end In the case by adding the following to the apache spec.rb file: describe service is by default named
mysql-default (and not the usual mysqld). Let's create: ff spec/defines/apache_vhost_spec.rb, with the following content: require 'spec helper' # Description of the "apache::vhost" 'define' resource describe 'apache::vhost, stype => :define do # As a requirement, we should load the apache class let :pre_condition do 'class {"apache":;}' end # Define apache::vhost, stype => :define do # As a requirement, we should load the apache class let :pre_condition do 'class {"apache":;}' end # Define apache::vhost, stype => :define do # As a requirement, we should load the apache class let :pre_condition do 'class {"apache":;}' end # Define apache::vhost, stype => :define do # As a requirement, we should load the apache class let :pre_condition do 'class {"apache":;}' end # Define apache::vhost, stype => :define apache::vhost, st
 title for the 'define' resource let:title do 268 Chapter 7 'mysite' end # Parameters list let:params do {:website => 'www.sample.com', :docroot => '/var/www/docroot', } end # Assertions list it { is expected.to contain file('/etc/apache2/sitesavailable/www.sample.com.conf')}
 .with_content(/DocumentRoot \var\/www\/docroot/) } it { is_expected.to contain_file('/etc/apache2/sitesenabled/www.sample.com.conf').with( 'ensure' => 'link', 'target' => '/etc/apache2/sitesenabled/www.sample.com.conf').with( 'ensure' => 'link', 'target' => 'link', 't
 "apache::htpasswd" 'define' resource describe 'apache::htpasswd', :type => :define do # As a requirement, we should load the apache class let :pre_condition do 'class {"apache":;}' end # Define a title for the 'define' resource let :title do 'myhtpasswd' end # Parameters list let :params do { 269 Testing and Writing Better Infrastructure Code with
Chef and Puppet:filepath => '/tmp/htpasswd', :users => [ { "id" => "user1", "htpasswd" => "hash1" } ] } end # Assertion list it { is_expected.to compile } i
 content: require 'spec_helper' # Description of the "apache::htaccess" 'define' resource describe 'apache::htaccess' end # Parameters list let :params do { :filepath =>
 '/tmp/htpasswd', :docroot => '/var/www/docroot', } end # Assertion list it { is expected.to contain file('/var/www/docroot/.htaccess') .with content(/AuthUserFile \/tmp\/htpasswd/) } end 270 Chapter 7 Now we have all our unit tests, and each one validates the initial target we
 defined earlier. Take this simple HTTP service is now available to any other container on the default network: # curl -I HTTP/1.1 200 OK However, we'd like it to be also available on the hello network Docker network. Send the
 roles to the Chef server using the knife Updated $ knife Updat
153 Automating Complete Infrastructures with Terraform ff An optional Slack Token ff An Internet connection How to do it... First things first: we need to define the Heroku provider aminute, you're ready to use a CoreOS
 cluster with basic monitoring, using only fully automated Terraform code! 136 Chapter 4 Provisioning a three-tier infrastructure on Google Compute Engine, using two CentOS 7.2 servers for the web and one master Google MySQL instance.
 Combined with the USER instruction in the Dockerfile, we'll be able to execute commands as an unprivileged user! Here's how an optimized Dockerfile looks, adding a normal user and group for the hello user, and then executing the /usr/bin/hello HTTP server as this new unprivileged user! FROM debian: jessie-slim COPY src/hello/hello /usr/bin/hello
 RUN chmod +x /usr/bin/hello RUN groupadd -r hello && useradd -r -g hello hello USER hello EXPOSE 8000 ENTRYPOINT ["/usr/bin/hello"] Once built and running, the daemon still runs correctly, but as an unprivileged user: $ ps aux USER PID %CPU %MEM COMMAND hello 1 0.0 0.2 /usr/bin/hello 358 VSZ 36316 RSS TTY 4768 ? Total Resources: 2
Touched Resources: 0 Touch Coverage: 0.0% Untouched Resources: yum package[httpd] apache/recipes/default.rb:1 261 Testing and Writing Better Infrastructure Code with Chef and Puppet We see the simulated Chef run execution times, as well as a coverage report (0%, as we didn't test anything for
now). You can run all your containers in your infrastructure like this and get centralized logging very easily! Managing IAM users with Terraform An essential part of using AWS is controlling access to the resources. Hiera can be customized in the way data is stored, but this will be out of the scope of this chapter; we will use default configuration
Using it as recommended by AWS is well documented, so let's use it with Docker instead. Compare this with the time it might take to find an outdated documentation, apply untested manual processes, and finally do whatever it takes to get something up and running under the pressure of an emergency. Inside the Vagrant guest, you can now check
for the PIP and Ansible versions: $ pip --version pip 8.1.2 from /usr/lib/python2.7/site-packages (python 2.7) $ ansible --version ansible 1.9.6 You can also check if Docker is installed, and verify now it's
 working as the Vagrant user: $ docker run -it --rm alpine ping -c2 google.com (216.58.211.78: seq=0 ttl=61 time=22.078 ms 64 bytes from 216.58.211.78: seq=0 ttl=61 time=22.078 ms 64 bytes from 216.58.211.78: seq=0 ttl=61 time=22.078 ms 64 bytes from 216.58.211.78: seq=0 ttl=61 time=21.061 ms Using Docker containers on CoreOS with Vagrant value of the coreOS with Vagrant 
 Docker containers are not forgotten with Vagrant. However, this can be done by adding a level of abstraction using the role and profile design pattern. pp.cZK277 {:ignore => } localhost $ scp /var/folders/k9/7sp85p796qx7c22btk7 tgym0000gn/T/ beaker20161101-75828-1128bth ubuntu-1604-x64:/tmp/apply manifest. To begin with, we need a
 discovery token, that can be obtained from . To start with the Debian 8 (Jessie) release, use the following in a Dockerfile: FROM debian: jessie ENTRYPOINT ["/bin/bash"] Linux distribution image Size Alpine 3.4 4.799 MB Debian 8 (slim) 80
MB Debian 8 123 MB Ubuntu 16.04 126.6 MB RHEL 7.3 192.5 MB CentOS 7.3 191.8 MB Fedora 25 199.9 MB With this information in hand, we can now decide to go for any one of these. The r10k tool needs a global configuration file, which must be created at the same level as our Vagrantfile, with the following content: :sources: :my-repos: remote:
 basedir: 'puppetcode/environments' Now let's use r10k: $ r10k ex r10k ex r10k yaml deploy environments total 0 drwxr-xr-x 8 ppomes staff 272B 26 Nov 16:40 translated by the staff 272B 26 Nov 16:40 t
 with our boxes and branches. He also worked for financial software companies on high-frequency trading technologies. Utilize the CloudWatch log group named docker run -d -p 80:8080 --log-driver="awslogs" --log-opt awslogs-region="us-east-1"
 log-opt awslogs-group="docker_logs" -log-opt awslogs-stream="www" tomcat:9 Navigating over the AWS Console, the new log stream will appear under Search Log Group: Clicking on the log stream name will give us access to all the output logs from our Tomcat container: 396 Chapter 10 We now have access to unlimited log storage and search
 features, and the amount of effort we put was very limited! Using docker-compose It's also possible to configure the logging driver using Docker Compose. Finished testing (4m48.86s). Install and ship a good enough NGINX configuration: sudo yum install -q -y nginx sudo cp /vagrant/nginx.conf /etc/nginx/nginx.conf /7. It's very likely that in the near
 future we'll want to use a variable for AuthUserFile. Bump the mysite cookbook in the mysite/metadata.rb file, and edit the users search in the mysite/metadata.rb file to include an evaluation of whether we're running chef:solo or not: if Chef::Config[:solo] Chef::Log.warn('This recipe uses search. So we just need to add it to Puppetfile with its
dependencies (do not forget to run r10k puppetlabs/inifile' mod 'puppetlabs/inifile' mod 'puppetlabs/puppetserver_gem' With this module, it is now very easy to prepare the Puppet server using the following: node 'puppetlabs/puppetserver_gem' With this module, it is now very easy to prepare the Puppet server using the following: node 'puppetlabs/puppetserver_gem' With this module, it is now very easy to prepare the Puppet server using the following: node 'puppetlabs/puppetserver_gem' With this module, it is now very easy to prepare the Puppet server using the following: node 'puppetlabs/puppetserver_gem' With this module, it is now very easy to prepare the Puppet server using the following: node 'puppetlabs/puppetserver_gem' With this module, it is now very easy to prepare the Puppet server using the following: node 'puppetserver_gem' With this module, it is now very easy to prepare the Puppet server using the following: node 'puppetserver_gem' With this module, it is now very easy to prepare the Puppet server using the following: node 'puppetserver_gem' With this module, it is now very easy to prepare the Puppet server using the following: node 'puppetserver_gem' With this module, it is now very easy to prepare the Puppetserver_gem' With this module, it is now very easy to prepare the puppetserver_gem' with the puppetserve
resource for the Puppet server. The Chef DK also brings a nice selection of the best tools and environments that work well together. We'll continue to build the LAMP server we started in the previous section by managing the Apache HTTP and MariaDB services right from Chef. Starting from here, you'll be able to iterate through more complex
Ansible playbooks. In fact, none are fixable in our case (click on Only show fixable to us by cloud-init), which will always be rewritten, no matter what, at each boot. The details are out of the scope of this book, but here are the
 steps: 1. The logs seem pretty optimistic, but let's double-check the daemon is really running on the host: $ systemctl status chef-client.service; enabled; vendor preset: disabled) Active: active (running) since Mon 2016-11-07 01:35:05 UTC; 57s ago Main PID:
 12943 (chef-client) CGroup: /system.slice/chef-client.service is indeed enabled and running! 289 Maintaining Systems Using Chef and Puppet Tweaking the convergence interval time Interesting enough, we see that the interval is
 introduced every 1,800 seconds (30 minutes). We'll explore the first one through the main Docker program. We want PHP-FPM to run as the Vagrant user = apache/user = vagrant/ /etc/phpfpm.d/www.conf 4. Puppet can also work in a standalone mode like Chef, but we will focus on a client/server
 architecture. Wouldn't it be awesome to have it working on both the current platforms with long-term support? Download the Starter Kit. In our case, we want to inject the value of the sitename "]}.conf" do source "virtualhost.erb" owner 'root
group 'root' mode '0644' variables( :website => "#{node["sitename"]}") end Don't forget to bump the cookbook version in apache/metadata.rb: version '0.3.0' Upload the cookbook to Chef server from the workstation: $ knife cookbook upload apache 219 Fundamentals of Managing Servers with Chef and Puppet Add the newly-created recipe to the
 remote node run list: $ knife node run list add apache::virtualhost Apply the new cookbook on the remote host: $ sudo chef-client Restart the Apache server manually to take the changes into account (be sure we'll automate that in the next pages): $ sudo systemate that in the next pages is the one we added: $ curl Hello from Chef!
 Good job! We've just covered how to manage files, directories, as well as dynamic templates, using pure Ruby code with Chef. Here's the working Dockerfile from earlier in this book: FROM debian:stable-slim RUN apt-get update -y \ && apt-get install -y apache2 \ && rm -rf /var/lib/apt ENTRYPOINT ["/usr/sbin/apache2ctl"] CMD ["-D",
 "FOREGROUND"] 363 Working with Docker Hadolint Let's start working with Hadolint, as it's easy to install (prebuilt binaries and Docker images) and use. Let's create the MySQL connection info = { host: '127.0.0.1', username: 'root', password: 'super_secure password: binaries and Docker images) and use. Let's create the MySQL connection information variable so it can be reused in our two resources: mysql connection info = { host: '127.0.0.1', username: 'root', password: 'super_secure password: 'super_secur
} In a proper production environment, we should use encrypted data bags for this matter, as seen in this chapter. Such information in the Puppet world is named facts. Storing updated cookbooks/starter/recipes/default.rb in the cache. For this example, I simply set the local IP the node is listening to, but feel free to adapt to your own case:
 ServerName DocumentRoot /var/www/ ErrorLog /var/log/httpd/error-.log CustomLog /var/log/httpd/access-.log combined 3. It's out of the scope of this book, but we need to register and answering the questions). Two options here are as follows: fften book, but we need to register and answering the questions. Two options here are as follows:
206 From the Chef server UI, select the host and click on Edit on the Run List box, then drag and drop the correct cookbook name on the Current Run List column: Chapter 6 ff From the knife CLI on the workstation, run the following: $ knife node run list add apache nodename: run list: recipe[apache] Either way, we just told the Chef server to apply
 the apache cookbook on this particular server. GitHub proposes to easily create a pull request. Chapter 5, Provisioning the Last Mile with Cloud-Init, explores everything we can do with cloud-init code—file management, server configuration, adding users and keys, repositories and packages, or examples of extensions such as Chef, CoreOS, and
Docker. Setting an environment to a node To set an already existing node environment to a node environment of this new production my node name production my node name production my node name production my node name to this new production my node name production my node 
command, we can start in the required environment right from the beginning (using an user named vagrant --sudo --environment production --run-list 'recipe[mysite]' 294 Chapter 8 Fixing cookbook versions for an environment Let's say our production systems are running a
perfectly stable mysite cookbook in version 0.3.1, but we want to try a new feature in the development infrastructure in the ecokbook. We're not happy with the default password and would like to use a better one, so simulate the staging environment. Getting ready To step through this recipe, you will need: ff A working Chef
 DK installation on the workstation ff A working Vagrant installation on the workstation ff The Chef code (optionally) from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code How to do it... Our goal is to create a configuration
 file containing our AWS credentials for the us-east-1 region, and it's not acceptable that you store the credentials in clear text on the Chef server. With hiera-eyaml, we need to declare a new backend to look up data in encrypted files. He's an open source software lover and has been working with Linux for more than 10 years now. Using an encrypted
data bag from a recipe Now that the data is safely stored on the Chef server, how do we access it from inside a Chef recipe? kitchen destroy: If all tests pass, this step destroys the testing environment. Execute foodcritic by navigating to the Chef repo and type in the following command: $ foodcritic For example, for testing our previous mysite
cookbook (excluding the auto-generated test directory, as it's not a cookbook in itself), we type the following command: $ foodcritic --exclude test cookbooks/mysite/recipes/htaccess.rb:7 FC033: Missing template:
cookbooks/mysite/recipes/htaccess.rb:9 FC033: Missing template: cookbooks/mysite/metadata.rb:1 FC065: Ensure source_url is set in metadata: cookbooks/mysite/metadata.rb:1 FC065: Ensure source_url is set in metadata.rb:1 FC065: Ensure source_url is
Let's start with FC003 ( . This way, we'll go through all the features while progressively building our end project—a working LAMP server with external dependencies on the latest community MySQL 5.7 release, and more features. To do this, follow these steps: 1. Installing the Chef Development kit and Puppet Collections The Chef ecosystem is as
rich as Chef itself is complex; there's a myriad of tools filling almost every imaginable task we can think of. If you're a Ruby developer, customize accordingly! COPY Gemfile lock /usr/src/app/ RUN bundle install --without test development --jobs 20 --retry 5 It's now time to copy the application code itself to the destination
folder, /usr/src/app (in this case, it's the current folder): COPY. In order to do a real example, perform the following steps: 1. We need a Git repository with two branches, production and test, with all of the previous code. For any new feature, anything added to the infrastructure should follow the same pattern: create a branch, insert your changes,
 open a pull request, discuss the changes with the coworkers, apply the change, and merging 3. Quickly reading through the Hubot documentation, let's install the Hubot documentation and let's install the Hubot documen
 using the usual heroku command, add the Heroku git remote -app iac-book-hubot Now you can git push heroku and see your application being deployed, all using Terraform. This builds on what has already been done previously, but the resources and cookbook shown here are made to be reusable anywhere.
To browse available cookbooks, navigate to the Chef Supermarket (think of it as a store for cookbooks): . These tools and techniques are absolutely key to write the best infrastructure code possible, and they are as fun to use as they are absolutely key to write the best infrastructure code possible, and they are as fun to use as they are absolutely key to write the best infrastructure code possible, and they are as fun to use as they are absolutely key to write the best infrastructure code possible, and they are as fun to use as they are absolutely key to write the best infrastructure code possible, and they are as fun to use as they are absolutely key to write the best infrastructure code possible, and they are as fun to use as they are as fun to use as they are absolutely key to write the best infrastructure code possible, and they are as fun to use as they are
 version managers such as RVM for Ruby, NVM for Node, or even Rackspace's DVM for Docker. downloading =ubuntu&pv=14.04&m=x86 64 to file /tmp/install.sh.1294/metadata.txt [...] Unpacking chef (12.14.89-1) ... Your MariaDB connection id is 4 We're very
 easily overriding values with simple YAML files! Suppose that we now want to include a reverse proxy to the mix, with a slightly earlier version of the Docker image and another MySQL password, to mimic a specific situation we have in production. ==> default: Creating the container... Inside this aws data bag folder, create a sample us-east-1.jsor
 file containing the credentials: { "id": "us-east-1", "aws access key": "h77/xZt/5NUafuE+q5Mte2RhGcjY4zbJ3V0cTnAc" } This is the standard procedure for a normal data bag. It's super useful when we're not sure if a directive is supported or not in the distribution. Vagrant can be greatly extended
 using plugins. Let's keep it simple here and refer to the complete online documentation for more complex uses. We'll see how this works in the coming sections. Create the ServerSpec integration/default/serverspec integration folder inside the platform cookbook file with the
following content. Nothing can be shared between organizations and this is usually where is defined a company, different business units, or even deliberately isolated corporate departments. The second server shows up in the AWS Console: Remember, the command to destroy a Terraform infrastructure is terraform destroy. This is not optimal, and
 less than desirable when managing a lot of nodes. See also ff More general sysdig usage examples at sysdig-examples at sysdig-e
 Image (AMI) 20 Amazon Web Services (AWS) 48 cloud-init, used 166, 167 AMI IDs reference link 20 Ansible applying 206, 207 generating 205 uploading 205, 206 Apache service enabling 213, 214 starting 213, 214
 application deployment automating 313, 318, 319 database, creating 314, 315 dependencies, including 314 from GitHub 316-318 from GitHub 316-318 from GitHub 316-318 from GitHub 316-318 automated Docker build creating 314, 315 dependencies, including 314 from GitHub 316-318 from GitHub 3
 Console reference link 64 AWS security groups used, with Terraform 54-56 B Bash executing, in an Ubuntu 16.04 container, testing 371, 372 Bats testing framework reference link 276 Beaker testing, for Puppet 271, 272 Bento Project
 reference link 8 Berkshelf MySQL cookbook, used 240-242 MySQL dependencies, used 240-242 reference link 248 used, for uploading cookbook dependencies 243 409 block storage volumes creating, on OpenStack 149 Bundler used, for creating ServerSpec environment 374 C cAdvisor tool 401 reference link 403 cAdvisor tool, storage backends
reference link 403 CentOS 7.2 272 CentOS 7 EC2 instance Chef information, sharing 88 connection information, passing 88 creating 87 provisioning, Terraform used with VMware 7, 8 CentOS image 340 starting 340 CentOS mirror references 261 Chef about 192 CentOS
7 EC2 instance, provisioning Terraform used 85-89 Chef-Solo mode 192 client mode 192 client mode 192 client mode 193 run list 194 server mode 193 run list 194 resources 194 roles 193 run list 194 server mode 192 client mode 192 client mode 193 run list 194 server mode 195 run list 194 server mode 195 run list 1
 410 using, as daemon 288, 289 Chef code linting, with Foodcritic 250-258 Chef Development Kit (Chef DK) 196 references 197 Test Kitchen tool 196 URL, for downloading 195 URL
for installation 197 Chef encrypted data bags accessing in CLI 300 MySQL root password, securing 304-306 Puppet server, preparing 304 Chef environments attributes, overriding for environment 295 cookbook, fixing for environment 295 creating 293-295
 environment, accessing from recipe 295 environment, obtaining from manifest 296 environment, setting to node 294 manual environment, reating 294 r10k 297 reference link 298 Chef metadata.rb reference link 216
Chef package reference link 212 Chef run context testing 260-262 Chef Server URL 86 Chef Server URL 86 Chef Server URL 86 Chef Server URL 871 Chef Supermarket URL 239 Chef Vault reference link 212 Chef vault reference link 213 Chef Server URL 86 Chef Server URL 871 Chef Supermarket URL 239 Chef Vault reference link 212 Chef Vault reference link 213 Chef Server URL 86 Chef Server URL 871 Chef Server URL 86 Chef Server URL 872 Chef Server URL 873 Chef Server URL 874 Chef Se
version upgrades Berkshelf 40 Chef recipe 38 integrating 38 simulating, Vagrant used 36-40 testing 39 Test Kitchen, testing 40 Vagrant omnibus Chef plugin 37, 38 Vagrant omnibus Chef plugin 3
TCP socket, enabling for network access 189, 190 timezone, setting on CoreOS 188 used, for configuring Chef cookbook at bootstrap 186 used, for configuring flannel 183, 184 used, for configuring fleet 181 used, for configuring Chef cookbook at bootstrap 186 used, for configuring flannel 183, 184 used, for configuring flannel 183
 server time zone 171, 172 used, for configuring systemd units 183 used, for deploying Chef client 184 used, for deploying Chef client 184 used, for deploying Chef client 185 used, for deploying remote Docker server 187 used, for deploying Chef client 185 used, for deploying Chef cl
 used, for managing keys 172-174 used, for managing packages 174, 175 used, for managing repositories 174, 175 used, for managing users 172-174 used, on Digital Ocean 166-168 used, on Digital Ocean 166-168 used, for managing users 172-174 used, on Digital Ocean 166-168 used, on Digital Ocean 166-168 used, for managing users 172-174 used, on Digital Ocean 166-168 used, on Digital Ocean 166-168 used, on Digital Ocean 166-168 used, for managing users 172-174 used, for managing users 174, 175 used, for managing user
   72, 73 code listing 325 commands executing, cloud-init used 176, 177 compute instances creating, on OpenStack 149-152 consistent systems Chef client, executing as cron 290 maintaing, scheduled convergence used 288-293 Consul URL 10
container building, with utilities 337, 338 debugging, sysdig used 404-408 contextual defaults used, with Terraform 61-63 Continuous Integration (CI) system about 108 connecting 388-391 cookbook dependencies including 242 uploading, Berkshelf used 243 Cookstyle 250, 251 411 CoreOS configuring, cloud-init used 178 Docker Containers, used
with Vagrant 45 etcd key value store, configuring cloud-init used 182, 181 locksmith, configuring cloud-init used 183, 184 fleet, configuring cloud-init used 183 update strategy, configuring cloud-init used 182 CoreOS Clair reference link 46, 178 systemd units, configuring cloud-init used 183 update strategy, configuring cloud-init used 182 reference link 46, 178 systemd units, configuring cloud-init used 183 update strategy.
395 CoreOS cluster reference link 179 CoreOS cluster members, creating 131, 132 CoreOS infrastructure cloud-Init, integrating 134 CoreOS cluster members, creating 131, 132 DNS Integration 133 dynamic StatusCake monitoring, integrating 134 CoreOS cluster members, creating 131, 132 DNS Integrating 134 CoreOS cluster members, creating 131, 132 DNS Integration 130 SSH keys, handling 131
credentials managing, cloud-init used 172-174 D data sharing, with container 337 data bag stubbing, for searches 265, 266 used, for centrally share data 230-236 data sources using 89, 90 Debian image 341 dependencies handling 223-225 deploy resource documentation reference link 320 412 Digital Ocean cloud-init, used 166-168 CoreOS
infrastructure, provisioning with Terraform 130 directories managing 216 managing 216 managing 216 managing 216 managing with Terraform 72, 73 Compose equivalent 18 deploying, with Chef 323, 324 local folder, sharing through Vagrant 14, 15 networking with Terraform 72, 73 Compose equivalent 18 deploying, with Chef 323, 324 local folder, sharing through Vagrant 14, 15 networking with Terraform 72, 73 Compose equivalent 18 deploying, with Chef 323, 324 local folder, sharing through Vagrant 14, 15 networking with Terraform 72, 73 Compose equivalent 18 deploying, with Chef 323, 324 local folder, sharing through Vagrant 14, 15 networking with Terraform 72, 73 Compose equivalent 18 deploying, with Chef 323, 324 local folder, sharing through Vagrant 14, 15 networking with Terraform 72, 73 Compose equivalent 18 deploying with Terraform 73 Compose equivalent 18 deploying with Terraform 73 Compose equivalent 18 deploying with Terraform 74 Compose equivalent 18 deploying with Terraform 
351, 352 NGINX Docker container, used through Vagrant 14 ports, exposing in Vagrant 14 ports, exposing in Vagrant 17 Docker-api reference link 379 Docker base image selecting 339 Docker Cloud vulnerabilities, scanning with 391-393 Docker Compose
extending 361, 362 orchestrating with 359-361 reference link 363 using 397, 398 Docker file_lint 364, 365 Docker host creating, Ansible used with Vagrant 44, 45 Docker Hub references 338 Docker image building, Git tags
used 386, 388 reference link 363 size, optimizing 343, 344 versioning, with tags 345, 346 Docker Registry reference link 367 Docker Registry reference link 367 Docker Security Scanning using 393, 394 docker stats command using 400 Docker Store reference
link 339 Docker Swarm cluster using 163, 164 Docker Swarm manager bootstraping, on Packet 158-160 Docker Swarm nodes bootstraping, on Packet 161, 162 Docker Swarm information, monitoring 399, 400 information, obtaining 399, 400 information, obtaining 399, 400 information, obtaining 399, 400 Docker Swarm nodes bootstraping, on Packet 161, 162 Docker Swarm information, obtaining 399, 400 information, obtaining 39
DVM reference link 113 dynamic code notifications, used 226-230 dynamic multiple hosts networking deployments, speed up with Linked clones 24 named NAT Networks, used 25 simulating 22-24 dynamic StatusCake monitoring integrating
135, 136 E Elastic Container Registry (ECR) 66 Embedded Ruby (ERB) template 218 errors intercepting, in tests 266 etcd URL, for configuring 239, 240 F FC003, FoodCritic references 252 Fedora image 340 file creation testing 264 files handling, cloud-init used 179, 180 external Chef cookbooks managing 239, 240 F FC003, FoodCritic references 252 Fedora image 340 file creation testing 264 files handling, cloud-init used 179, 180 external Chef cookbooks managing 239, 240 F FC003, FoodCritic references 252 Fedora image 340 file creation testing 264 files handling, cloud-init used 179, 180 external Chef cookbooks managing 239, 240 F FC003, FoodCritic references 252 Fedora image 340 file creation testing 264 files handling, cloud-init used 179, 180 external Chef cookbooks managing 239, 240 F FC003, FoodCritic references 252 Fedora image 340 file creation testing 264 files handling, cloud-init used 179, 180 external Chef cookbooks managing 239, 240 F FC003, FoodCritic references 252 Fedora image 340 file creation testing 264 files handling, cloud-init used 179, 180 external Chef cookbooks managing 264 files handling, cloud-init used 179, 180 external Chef cookbooks managing 279, 280 external Chef
init used 170 managing 216 files content testing 279, 280 files existence testing 279, 280 files ex
G H git using, team working 326 GitHub access rights, setting 125 configuring, to Docker-Hub-automated build pipeline 385, 386 managing, with Terraform 122, 123 organization, users adding to 124 teams, adding 124 GitHub Flow about 114 reference link 114 GitHub Hubot reference link 153 GitLab CE, and CI Runners, OpenStack
provisioning 146 Git repository reference link 297 Git tags used, for building Docker image 386-388 Golang application building, with Docker 349 Golang image 342 Google Cloud Console reference link
 142 Google Compute Engine three-tier infrastructure, provisioning on 137 Google Compute Firewall rule creating 141. Google Compute Engine three-tier infrastructure, provisioning on 137 Google Compute Firewall rule creating 141. Google Compute HTTP instances creating 142-144 Google Compute Firewall rule creating 141. Google Compute Firewall rule creating 141. July Google Compute Firewall rule creating 142-144 Google Compute Firewall rule creating 141. July Google Compute Firewall rule creating 142-144 Google Compute Firewall rule creating 141. July Google Compute Firewall rule creating 141. July Google Compute Firewall rule creating 142-144 Google Compute Firewall rule creating 142-144 Google Compute Firewall rule creating 141. July Google Compute Firewall rule creating 142-144 Google Compute Firewall rule creating 141. July Google Compute Firewall rule creating 141. July Google Compute Firewall rule creating 141. July Google Compute Firewall rule creating 142-144 Google Compute Firewall rule creating 141. July Google Compute Firewall rule creating 142-144 Google Compute Firewall rule creating 141. July Google Compute Firewall rule creating 142-144 Google Compute Firewall rule creating 141. July Google Compute F
HashiCorp Atlas reference link 34 HashiCorp Configuration Language (HCL) 50 Heroku about 153 addons, adding Terraform used 155 addons. See also ff Images on the Docker image size Docker image 
 what we just did with Terraform in this recipe: --- name: base security group ec2 group: name: base security group description: Base Security group description: Base Security group ec2 group: name: base security group description: Base Security group description: Base Security group ec2 group: name: base security group ec2 group: name: base security group description: Base Security group description: Base Security group ec2 group: name: base security group ec2 group: name: base security group description: Base Security group description: Base Security group ec2 group: name: base security group ec2 group: name: base security group description: Base Security group ec2 group: name: base security group description: Base Security group ec2 group: name: base security group ec2 group:
AWS EC2 (an SSH key pair and a security group). Chef plugins Chef is also highly modular, with a great number of plugins available either directly from Chef, vendors, or the community. It's the case with the official MySQL cookbook, which depends on the SELinux cookbook for the RHEL/CentOS platform. For Terraform to interact with an IaaS, it
needs to have a provider configured. The resource is named aws s3 bucket object, and you need to reference the bucket previously created, the destination name (index.html), and its content. This will create the user and grant all the privileges to it: mysql database user 'wordpress user' do connection mysql connection info password 'changeme'
database name 'wordpress' host '%' privileges [:all] action [:create, :grant] end At this point, we should have everything we need related to the database. You can use this very simple default configuration: config.vm.provision "ansible local" do |ansible loca
==> srv-1: Running provisioner: ansible local... Getting ready To work through this recipe, you will need the following: ff A working Chef DK installation on the workstation ff A working Chef DK installation on the workstation ff A working Chef DK installation on the remote host How to do it... To install a package on a Red Hat-based system, we'd use either yum (until CentOS 7) or dnf (for Fedora
after version 22). (ID: i-0f9106905e74a29f7) No changes, Getting ready To work through this recipe, you will need the following: ff A working Chef DK installation on the workstation ff A working Chef DK installation on the workstation ff A working Chef DK installation on the remote host if The Chef code from the previous recipe, you will need the following: ff A working Chef DK installation on the workstation ff A working Chef DK installation on the workstation ff A working Chef DK installation on the workstation ff A working Chef DK installation on the workstation ff A working Chef DK installation on the workstation ff A working Chef DK installation on the workstation ff A working Chef DK installation on the workstation ff A working Chef DK installation on the workstation ff A working Chef DK installation on the workstation ff A working Chef DK installation on the workstation ff A working Chef DK installation on the workstation ff A working Chef DK installation on the workstation ff A working Chef DK installation on the workstation ff A working Chef DK installation on the workstation ff A working Chef DK installation on the workstation ff A working Chef DK installation on the workstation ff A working Chef DK installation on the workstation ff A working Chef DK installation on the workstation ff A working Chef DK installation on the workstation ff A working Chef DK installation ff A working Chef DK installation
the package resource. Now let's use the template resource to generate a file from the template we just created, in the apache/recipes/virtualhost.rb file. Let's destroy the VM: $ vagrant destroy -f Including this vagrant -f Includin
system, and that potentially, it's a matter of three minutes and 30 seconds to recover from nothing to a working state. Beware of the image size; at the time of writing, the image is 1.7 GB, while the hadolint binary is less than 20 MB: $ docker run --rm -i lukasmartinelli/hadolint < Dockerfile Linting Dockerfiles from this chapter, we'll notice different
warnings. Launch simple or complex environments, simulate various virtual networking configurations, combine Vagrant and Docker or the Amazon cloud, and hand over the provisioning of virtual machines to Chef and Ansible. Start by creating the /etc/aws folder using the directory resource: directory resourc
'0755' action :create end 3. Fedora 25 has fedora:25, and it goes back to fedora:20 at the time of writing. Our first step is to require the helper created earlier in the subject of your message. 55 Provisioning IaaS with Terraform You
can see your newly created security group by logging into the AWS Console and navigating to EC2 Dashboard | Network & Security Groups: Another way of accessing the same AWS Console information is through the AWS command line: $ aws ec2 describe-security group $ --group-names base security group $ ... } There's more... We can
achieve the same result using Ansible. So, we have to specify the full URL to the Vagrant box in our Vagrant.box end As CoreOS is a minimal OS, it doesn't support any of the VirtualBox guest addition tools, so we'll disable them, and don't try anything if we
(most likely) have the vagrant-vbguest plugin: config.vm.provider:virtualbox do |vb| vb.check guest additions = false vb.functional vboxsf = false end Let's create a new VM definition, using the CoreOS Vagrant box: config.vm.define "core-1" do
|config| config.vm.hostname = "core-1" config.vm.network "private network", type: "dhcp" end We now need to run the mariadb and wordpress official containers from the Docker Hub. However, we can create as many as needed. Let's get to work. Once the package is installed, all files are deployed under /opt/puppetlabs: $ ls -la
/opt/puppetlabs/bin/puppet lrwxrwxrwx 1 root root 20 Sep 22 18:42 /opt/puppetlabs/bin/puppet -> ../puppet/bin/puppet $ /opt/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetlabs/puppetla
variable. network.2.address | ", ] }//. The first thing we need is to install r10k. Step 1 : FROM ubuntu:16.04 ---> Running in 425bfb1e8ee1 [...] Removing intermediate container 425bfb1e8ee1 Successfully built c86310e48731 We
can see each line of our Dockerfile is a step in the build process, each step being a container itself (hence the different ID each time). Terraform allows us to get an answer to that question very specifically by targeting the resource directly in the planning phase: $ terraform plan -var-file=plan.tfvars target="aws s3" bucket.bucket" [...]
aws s3 bucket.bucket: Refreshing state... Using Docker Security Scanning There's a similar feature on the Docker Hub that uses a paid account, though still in preview at the time of this writing. This is needed by the puppet/hiera module (see parameter master service) Install the eyaml backend in the Puppet server Update the Hiera configuration in
order to use this backend Generate the private and public keys Restart the Puppet server 303 Maintaining Systems Using Chef and Puppet Private and public keys/puppet/keys/public keys Restart the Puppet server 303 Maintaining Systems Using Chef and Puppet Private and public keys Restart the Puppet server 303 Maintaining Systems Using Chef and Puppet Private and public keys will be respectively placed in /etc/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/puppet/keys/pup
the Vagrant infrastructure. Notice: Installing -- do not interrupt ... This token is unique and can be distributed through different channels. A single Docker container can connect multiple Docker networks. 107 Going Further with Terraform Job done! Your coworkers can now push and pull from the Consul source! In a production Consul cluster, it
means replicated and synchronized states on each node, with added privacy. We'll end this chapter with a powerful and scalable Docker Swarm cluster on bare metal using Packet, capable of scaling hundreds of containers. Let's ensure the APT cache is cleaned after we build the image so we don't waste precious space. Check them out! Downloading
the color images of this book We also provide you with a PDF file that has color images of the screenshots/diagrams used in this book. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website or added to any list of existing errata under the Errata section of that title. Consider you need to rebuild
dozens of containers in an emergency, because of the existence of another OpenSSL security bug; nobody would want to be the one to upload them one by one, especially with the bad uplink at work. You've successfully launched your first AWS EC2 instance using repeatable Terraform code! 58 Chapter 2 Scaling the number of instances What if you
want to launch two similar instances, maybe for debugging purposes, or for instant action behind a load balancer? managing Terraform used 153 URL 153 using, with Puppet 230-236 Hiera-eyaml used, with Puppet 298-302 HTTP 35 httpd package installation testing
262, 263, 278 414 I laaS cloud-init, combining 169 Terraform, combining 169 IAM Policies reference link 75 IAM users CloudWatch Logs 78 EC2 in read-only 77 managing, with Terraform 74, 75, 79, 80 restrictions, testing 76, 77 Simple Storage Service (S3) access 75, 76 Identity and Access Management (IAM) 74 infrastructure code testing, Test
 Kitchen used for Chef 271, 272 infrastructure and 101, 102 change, simulating 101, 102 change, targeting 102 planning 99, 100 simulating, Terraform used 98 infrastructure and 101, 102 change, targeting 101, 102 change, targeting 102, 103 integration testing with ServerSpec 276-286 J Java image 343 K key pair creating, on OpenStack 147 keys managing, cloud-init used 172-174 L
Laravel provision, enabling 32 public LAN networking 32, 33 shared folder 32 shell provisioning 31 used, for configuring NGINX 30 working, on LAN 30, 33 Lightweight Resources and Providers (LWRP) reference link 242 Linux, Apache, MySQL, PHP (LAMP) server 192 Linux distributions container image 341 listening ports testing 279 local
execution using 89, 90 locksmith about 182 configuring, cloud-init used 182 log group 395 M Makefile reference link 113 using 373 MariaDB service enabling 214, 215 meaningful outputs generating, with Terraform 60, 61 Message Of The Day (MOTD) file 170 metadata.rb, Chef reference link 226
multimachine recovery 331-333 multiple network connecting, to container 353 multi-provider Vagrant environments enabling 10, 11 MySQL dependencies used, with Berkshelf 240-242 MySQL deployment testing 243-247 N networked 3-tier architecture app application servers
26 database 26 NGINX reverse proxy 28, 29 simulating, with Vagrant 25, 26 networking with Docker 351, 352 Nginx executing, in Nginx container 336, 337 nginx-proxy reference link 363 node Chef client, bootstrapping on 201-204 Node.js application 26, 28 Node JS image 341 notifications used, for dynamic code 226-230 ntegration monitoring, with Docker 351, 352 Nginx executing, in Nginx container 336, 337 nginx-proxy reference link 363 node Chef client, bootstrapping on 201-204 Node.js application 26, 28 Node JS image 341 notifications used, for dynamic code 226-230 ntegration monitoring, with Docker 351, 352 Nginx executing, in Nginx container 336, 337 nginx-proxy reference link 363 node Chef client, bootstrapping on 201-204 Node.js application 26, 28 Node JS image 341 notifications used, for dynamic code 226-230 ntegration monitoring, with Docker 351, 352 Nginx executing, in N
Status Cake 125, 126 NVM reference link 113 415 O object storage container creating, on Open Stack 152 Ohai information, accessing 308, 309 System information, accessing from Chef recipe 311, 312 reference link 313 omnibus installer 184 Open Stack block storage
volumes, creating 149 cloud-init, used 166-169 compute instances, creating 149 ovh URL 146 P packages installing 204 managing, cloud-init used 174, 175 Packet Docker
Swarm manager, bootstraping 158-160 Docker Swarm nodes, bootstraping 161, 162 project, creating Terraform used 157 URL 156 PHP URL 217 PHP cookbook creating 209-212 PHP Docker image 343 platform supporting 325 416 Platform-as-a-
Service (PaaS) 153 PostgreSQL RDS Database creating, Terraform used 66, 67 private Docker registry deploying, with S3 storage 365, 366 private Docker registry using 338 production deploying to 327 Project Atomic URL 364 Puppet reference link 313 Puppet agent 201 Puppet code linting, with
puppet-lint 250-258 Puppet coding style 254 Puppet Collections about 196 installing 197 Puppet Hiera reference link 245 Puppet Hiera reference link 248 Puppet Language Style reference link 259 Puppet modules managing 239, 240 Puppet notify reference
```

```
link 230 Puppet package references 197 Puppet r10k reference link 248 Puppet service reference link 248 Puppet service reference link 248 Puppet service reference link 298 Rbenv reference link 113 README.md file creating 255, 256, 258
recipes inclusion testing 266 Red Hat Enterprise Linux (RHEL) image 340 Red Hat's Docker reference link 340 registry 66 Relational Database Service (RDS) 68 remote commands executing, at bootstrap Terraform used 92-94 remote Docker service (RDS) 68 remote commands executing, at bootstrap Terraform used 174, 175 repository existence
testing 280 rspec-puppet unit testing with 259, 260, 266-271 references 271 Ruby image 342 Ruby-on-Rails web application deploying, in Docker Swarm cluster creating, on bare metal with Packet 156 scaleway URL 156 scheduled convergence used, for maintaining 366, 367 scalable Docker Swarm cluster creating, on bare metal with Packet 156 scaleway URL 156 scheduled convergence used, for maintaining 366, 367 scalable Docker Swarm cluster creating, on bare metal with Packet 156 scaleway URL 156 scheduled convergence used, for maintaining 366, 367 scalable Docker Swarm cluster creating, on bare metal with Packet 156 scaleway URL 156 scheduled convergence used, for maintaining 366, 367 scalable Docker Swarm cluster creating, on bare metal with Packet 156 scaleway URL 156 scheduled convergence used, for maintaining 366, 367 scaleway URL 156 scheduled convergence used, for maintaining 366, 367 scaleway URL 156 scheduled convergence used, for maintaining 366, 367 scaleway URL 156 scheduled convergence used, for maintaining 366, 367 scaleway URL 156 scheduled convergence used, for maintaining 366, 367 scaleway URL 156 scheduled convergence used, for maintaining 366, 367 scaleway URL 156 scheduled convergence used, for maintaining 366, 367 scaleway URL 156 scheduled convergence used, for maintaining 366, 367 scaleway URL 156 scheduled convergence used and the scheduled con
consistent systems 288-293 scratch Docker image using 350 Secure Shell (SSH) 5, 36, 54 security group creating, on OpenStack 148 ServerSpec environment, creating 278 reference link 276, 379 server time zone configuring, cloud-init used 171 service status testing 263, 279
Simple Storage Service (S3) about 63 managing, with Terraform 63-66 Spec Helper 260 SSH key creating 121 using, on AWS provider 51-53 staging deploying to 327 state file managing 217 Status Cake automated ping monitoring test, creating 126, 127 HTTPS test, creating 127 monitoring test, creating 127 monitoring test, creating 127 monitoring test, creating 128, 128 managing 217 Status Cake automated ping monitoring test, creating 126, 127 HTTPS test, creating 127 monitoring test, creating 128 managing 217 states file managing 217 states file managing 217 states file managing 218 managing 218 managing 218 managing 219 monitoring test, creating 126, 127 monitoring test, creating 126 managing 218 managing 219 monitoring test, creating 127 monitoring test, creating 128 managing 218 managing 218
integration 125, 126 URL 125, 135 system using 398, 399 systemd using 398, 399 system units configuring, cloud-init used 183 system information accessing, with Ohai 308, 309 manipulating, with Ohai 308, 309
system services managing 213 T team workflow changes, applying 121, 122 example 113, 114 Git repository, creating 114, 115 infrastructure code, creating 116, 117 templates configured Ansible, applying 91, 92 local-exec Terraform provisioner
91 managing 216 using 89, 90 templates creation testing 265 Terraform applying 153 AWS provider, configuring in 48-50 AWS security groups, used 54-56 CloudWatch Logs, enabling for Docker 72, 73 code, maintaining 108 combining, for IaaS 169 contextual defaults, used 61-63 CoreOS infrastructure, provisioning on Digital Ocean 130 Docker
container, used 110-113 Docker, used 94-97 GitHub, managing 122, 123 Heroku, application creating 154 Heroku, using 155 IAM users, managing with 74, 75, 79, 80 infrastructure environments, handling 82-85 infrastructure state, sharing 103, 104 meaningful outputs, generating with 60, 61 production flag, setting 85 remotely, sharing with Consultation creating 103, 104 meaningful outputs, generating with 60, 61 production flag, setting 85 remotely, sharing with Consultation creating 103, 104 meaningful outputs, generating with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, setting 85 remotely, sharing with 60, 61 production flag, sharing with 60 production flag, sharing wi
107 remotely, sharing with S3 105-107 418 sharing, with Git 104, 105 Simple Storage Service (S3), managing with 63-66 state file, sharing 108 style, validating 108, 110 syntax, validating 108, 109 tfstate isolated, maintaining 85 used, for adding Heroku addons 155 used, for bootstraping Docker Swarm manager 158-160 used, for bootstraping 108, 109 tfstate isolated, maintaining 85 used, for adding Heroku addons 155 used, for bootstraping Docker Swarm manager 158-160 used, for bootstraping 108, 109 tfstate isolated, maintaining 108, 109 tfstate isolated, mainta
Docker Swarm nodes 161, 162 used, for creating PostgreSQL RDS Database 67-71 used, for
addons 153 used, for managing Heroku apps 153 used, for provisioning CentOS 7 EC2 instance with Chef 85-89 used, for simulating infrastructure environments 98 Terraform, Docker container reference link 110 Test-Driven Development (TDD) about 287, 320 workflow, using 320, 321, 328 Debian Jessie base's Docker image, used 375, 376 NGINX
executing 378, 379 NGINX package, installing 377 with, Docker 374 with, Docker 374 with, Docker 374 with, ServerSpec 374 Test Kitchen configuring 272 reference link 276 testing with 273-276 used, for testing infrastructure API credentials, generating for Google project 138 Google Compute Firewall rule, creating 141
Google Compute HTTP instances, creating 138-141 Google Compute instances, load balancing 141, 142 Google MySQL Database, instance creating 136 Ubuntu 16.04 container Bash, executing 336 Ubuntu EC2 instance creating, Terraform used 56-58 scaling 59 Ubuntu image
339 Ubuntu Xenial (16.04 LTS) using 3-5 Vagrant box, adding 2, 3 unit testing with ChefSpec 259, 260, 266-271 unprivileged users security 357, 358 update strategy configuring, cloud-init used 182 users managing, cloud-init used 172-174 V Vagrant 272 Ansible, used to create Docker host 40-43 Docker, used 13
Docker, used for Ghost blog behind NGINX 16-18 URL, for downloading 2 used, with Docker 19-22 Vagrant environment access, sharing 33-35 ghost engine, considerations 35 provisioning 34 Vagrant plugins references 2 Vagrant VM
customizing 12 hostname, setting 12 informational message, displaying to user 13 minimum Vagrant version, specifying 13 new box version check, disabling at startup 12 specific box version, using 12 VirtualBox Guest Additions enabling, in Vagrant 5-7 virtualBox Guest Additions enabling, in Vagrant 5-7 virtualBox Guest Additions enabling at startup 12 specific box version, using 12 VirtualBox Guest Additions enabling, in Vagrant 5-7 virtualBox Guest Additions enabling at startup 12 specific box version, using 12 VirtualBox Guest Additions enabling at startup 12 specific box version check, disabling at startup 12 specific box version check box version
reference link 7 VMware vCenter plugin reference link 8 VMware vCloud Air plugin reference link 8 VMware vCloud Director plugin reference link 8 VMware vCloud Directo
Instrumentation (WMI) 309 WordPress reference link 316, 320 working systems rebuilding 329, 330, 334 419 Year: 2,017 City: Birmingham; English Identifier: 9781786464918 Org File Size: 12,016,962 Extension: pdf Tags: Ressources Internet Then, maybe we should keep some of
the good old practices and start by executing our daemon as a user other than root. Let's take advantage once again of the Ruby nature of our File: vm memory = 1024 vm cpus = 2 Now replace the four values by their variable names and you're done: you're centrally managing
characteristics of the Vagrant environment you're using and distributing, whatever hypervisor you're using Chef DK installation on the workstation A working Chef client configuration on the remote host The Chef code from Chapter 6, Fundamentals of Managing
Servers with Chef and Puppet, or any custom Chef code. Use an array if the command is followed by arguments to be enforced: FROM debian:stable-slim RUN apt-get install -y apache2 && \ rm -rf /var/lib/apt/ EXPOSE 80 ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"] To override the whole command at runtime
use the --entrypoint option: $ docker run -it --rm --entrypoint /bin/sh httpd # hostname 585dff032d21 To have a command that can be simply overridden with an argument, use the CMD instruction instead of ENTRYPOINT: FROM debian:stable-slim RUN apt-get update -y && \ apt-get install -y apache2 && \ rm -rf /var/lib/apt/ EXPOSE 80 CMD
["/usr/sbin/apache2ctl", "-D", "FOREGROUND"] 354 Chapter 9 To override the command, simply give another command as an argument at runtime: $ docker run -it --rm httpd /bin/sh # hostname cb1c6a7083ad We can combine both instructions to have a more dynamic container. Amazon Web Services Elastic Compute Cloud (EC2) and similar
 Infrastructure-as-aService providers like Google Cloud, Azure or Digital Ocean, to name a few, are selling virtual machines with varying compute power and network bandwidth for a fee. For simplification, we're using the three usual environment variables—AWS ACCESS KEY ID, AWS SECRET ACCESS KEY, and AWS DEFAULT REGION: Here's a
typical Ansible file hierarchy: — keys | — aws terraform.pub — main.yml In the main file (main.yml), let's declare that our host (localhost) will apply the role dedicated to manage our keys: --- hosts: localhost roles: - ec2 keys In the ec2 keys main task file, create the EC2 key
 (roles/ec2 keys/tasks/main.yml): --- name: ec2 admin key 53 Provisioning IaaS with Terraform ec2 key: name: admin key key material: "{{ item }}" with file: './keys/aws terraform.pub' Execute the code with the following command: $ ansible-playbook -i localhost main.yml TASK [ec2 keys: ec2 admin key]
 with ingress (incoming traffic) and egress (outgoing traffic) rules applied to EC2 instances. The systemd unit is now updated: $ systemctl status chef-client rules applied to EC2 instances. The systemd unit is now updated: $ systemctl status chef-client rules applied to EC2 instances. The systemd unit is now updated: $ systemctl status chef-client rules applied to EC2 instances. The systemd unit is now updated: $ systemctl status chef-client rules applied to EC2 instances. The systemd unit is now updated: $ systemctl status chef-client rules applied to EC2 instances. The systemd unit is now updated: $ systemctl status chef-client rules applied to EC2 instances. The systemd unit is now updated: $ systemctl status chef-client rules applied to EC2 instances. The systemd unit is now updated: $ systemctl status chef-client rules applied to EC2 instances. The systemd unit is now updated: $ systemctl status chef-client rules applied to EC2 instances. The systemd unit is now updated: $ systemctl status chef-client rules applied to EC2 instances. The systemd unit is now updated: $ systemctl status chef-client rules applied to EC2 instances. The systemd unit is now updated: $ systemctl status chef-client rules applied to EC2 instances. The systemd unit is now updated: $ systemctl status chef-client rules applied to EC2 instances. The systemd unit is now updated: $ systemctl status chef-client rules applied to EC2 instances. The system rules applied to EC2 instances applied to EC2 instances. The system rules applied to EC2 instances applied to EC2 instances applied to EC2 instances applied to EC2 instances. The system rules applied to EC2 ins
minutes with a controlled variation of 300 seconds. As we can do that with the Terraform outputs, let's use this example for a start, in outputs.tf. [...] ----> Installing Chef Omnibus (install only if missing) [...] resolving cookbooks for run list: ["mysite::default"] Synchronizing Cookbooks: - apache (0.5.0) - php (0.2.0) - selinux (0.9.0) - yum-mysql-
community (1.0.0) - mysite (0.3.1) - mysql (8.0.4) - yum (4.0.0) [...] Chef Client finished converging (3m18.96s). [Usage](#usage) * [Apache installation] (#installation) * [Defining a vhost](#vhost) ## Description Sample module for Apache on Ubuntu systems ## Usage ### installation To
                                                                       ``` ### vhost To create a vhost: ``` apache::vhost{'mysite': website => 'www.example.com', docroot => '/var/www/example', } ``` 255 Testing and Writing Better Infrastructure Code with Chef and Puppet We also need to document all statements and their parameters, using the @param tag inside comments
at the top of each manifest. Here's the basic structure: resource "aws_security_group" | fase_security_group" | fase_security_group" | fase_security_group" | fase_security_group | fase_security_grou
outbound. Let's see this in depth and learn when to choose what source. This can be done easily in the Apache manifest; a file directive for the deletion of /etc/apache2': ensure => present, } service {'apache2': ensure => running, enable
=> true } 220 Chapter 6 file {'/etc/apache2/sites-enabled/000-default.conf': ensure => absent, } } Now let's create the code for the virtualbox 4. By combining all the related commands from download to deletion, we keep a
clean state on this particular layer. We'll walk through those tools to end up with a much better and cleaner code. Here, no login or password are provided by AWS. Let's copy the keys from the Puppet server and store them in a keys folder under $HOME: $ ls ~/keys/ private key.pkcs7.pem public key.pkcs7.pem 3. ServerName DocumentRoot
/var/www/ ErrorLog /var/log/httpd/error-log CustomLog /var/log /var/
 Resources: 5 Touched Resources: 2 263 Testing and Writing Better Infrastructure Code with Chef and Puppet Touch Coverage: 40.0% Untouched Resources: directory[/var/www/default] apache/recipes/virtualhost.rb:15 template[/etc/httpd/conf.d/default.conf]
apache/recipes/virtualhost.rb:22 The good news is that ChefSpec still tells us which resources are not tested! Testing directory. By default, nodes without an environment. So a typical /24 network won't be enough (that's the docker
network default). Getting ready To work through this recipe, you will need the following: ff A remote server, with a user with SSH access ff A working Chef DK installation on the workstation How to do it... Let's say we already have a server running somewhere available with a user. It's advised to fix (or pin) versions where appropriate—especially in
production. To illustrate all those features, throughout the chapter we will build a classic LAMP (Linux, Apache, MySQL, PHP) server on CentOS 7.x, from scratch, 100% automated with Chef. ==> default: Deleting the VM... Getting ready To step through this recipe, you will need the following: ff A working Terraform installation ff A StatusCake
 account () 125 Going Further with Terraform ff Optionally, an infrastructure managed by Terraform (refer to the previous recipes) ff An Internet connection How to do it... Start by setting the new statuscake provider, as we did with AWS or GitHub, using a username and API key: provider "statuscake" { username = "${var.statuscake username}'
apikey = "${var.statuscake apikey}" } Declare the variables in variables in variable "statuscake username" } variable "statuscake apikey" { default = "hackme" description = "Sets the Statuscake API Key" } Also, don't forget to set those variables to your own values in
terraform.tfvars. Docker Inc. Getting ready To step through this recipe, you will need the following: ff 12 A working Terraform installation Chapter 3 ff A GitHub account (with an API token) ff An Internet connection How to do it... We want to manage a GitHub organization named ACME. Pushing to our remote branch will automatically add our
commit to the pull request as well: $ git add provider.tf $ git commit -m "fixed bad formatting" [new infrastructure b027825] fixed bad formatting 1 file changed, 3 insertions(+), 1 deletion(-) $ git push Our coworker can now see in real time that we took her remark into account, as GitHub automatically marks it as outdated: 120 Chapter 3 Now our
coworker pulled the changes on her side, tried to plan the changes So let's do that right now: $ terraform, and announces she's happy with the results as well: Apply the changes So let's do that right now: $ terraform apply aws key pair.admin key: Creating... And finally, we'll make sure we're always ready to redeploy any part of the infrastructure at any time (think
of a disaster). The team at WordPress built a container that auto-configures to some extent through environment variables similar to what we saw earlier in this chapter. [...] ==> srv-1: Launching an instance with the following settings... To have more logs regarding Puppet agent installation and execution, we can add a line log level: verbose in the
nodeset file: HOSTS: ubuntu-1604-x64: roles: - agent - default 282 Chapter 7 platform: ubuntu-16.04-amd64 hypervisor: vagrant box: bento/ubuntu-16.04 contained in the apache module. network.0.address} \"docker swarm join-token worker -q\" >
${var.worker token file}" } We're now done with the Docker Swarm manager! Bootstraping Docker Swarm modes on Packet using Terraform We need nodes to join the swarm, so the workload can be spread. Getting ready To step through this recipe, you will need the following: ff A working Terraform installation ff An AWS account with an SSH key
configured in Terraform and a security group allowing SSH connections from outside (refer to the chapter 2, Provisioning laaS with Terraform recipes) ff An Internet connection 92 Chapter 3 How to do it... Before diving into the provisioning laaS with Terraform recipes) ff An Internet connection 92 Chapter 3 How to do it... Before diving into the provisioning part, let's start by describing a classic CentOS 7.2 AMI in instances.tf: resource "aws instances" "centos" { ami
= "${lookup(var.aws_centos_ami, var.aws_region)}" instance_type = "${aws_key_name}" security_group.base_group.base_group.base_group.base_group.base_group.base_group.b
 "aws centos ami" { type = "map" default = { eu-west-1 = "ami-7abd0209" us-east-1 = "ami-6d1c2007" } } variable "aws instance type" { default = "t2.micro" description = "Instance Type" } Now, what are our immediate objectives for this system?: ff Fully update it: sudo yum install -y ff Enable the EPEL repository: sudo yum install epel-release -y ff
Add a custom name server: echo "nameserver 8.8.8.8" | sudo tee -a / etc/resolv.conf ff Add the Puppet Labs repository: sudo yum install . 258 Chapter 7 See also The Puppet Language Style: ff ff Cookstyle: Foodcritic: Unit testing with ChefSpec and rspec-puppet ChefSpec is a Chef cookbook RSpec unit testing framework written by the great Seth
Vargo (Opscode Chef, Hashicorp). This is supported out of the box by cAdvisor using --http auth file /cadvisor.htpasswd --http auth file /cadvisor.htpasswd --http auth realm my message. To execute a container in this specific Docker network, use the --network option like this: $ docker run -it --rm --name hello --network hello Confirm this container is in the
172.18.0.0/16 network space from the hello network. PAddress }}' hello "172.18.0.2" This container will be protected from unauthorized access from any container will be protected from un
control over a Docker Engine remotely, using a nifty systemd configuration feature! There's more... Working like this usually ensures a very high quality of work overall and very sustainable containers. Depending on how we created our cookbooks, this folder may already exist. The Spec Helper It's helpful to have a common set of requirements for all
the concerned cookbook tests. This registry is either run by you or as a service. org/). In the cookbooks mysite/metadata.rb file, do the same and replace the mariadb dependency with the three new ones: depends "mysql", '~> 8.0' depends "selinux" depends "yum-mysql-community", '~> 1.0' 3. $ docker run -it --rm hello Using the scratch Docker
image Now, it's a bit of a waste of space to have a 675 MB+ image for the very often small Golang application that often is only a few MB, and it takes time to deploy on servers. If we need an action to take place immediately, at the risk of breaking the state of the system, we can add the simmediately timer at the end of the line. How it works... Under
the hood, the Quay Security Scanner is based on Clair. Our build process, and CI) builds the app with the golang image, but does not ship the compiled application with it, saving usually 95-99% of the space, depending on the size of our binary: FROM scratch COPY src/hello/hello /hello ENTRYPOINT ["/hello"] This generates
the smallest image imaginable. Let's create a new recipe named htaccess.rb under the mysite cookbooks/mysite htaccess (this is an arbitrary location, adaptable to your needs) and the Apache configuration file under the web root: $ chef generate recipe cookbooks/mysite htaccess 2. describe file
('/etc/apache2/htpasswd') do its(:content) { should match /user1:hash1/} end # htaccess file deployed? So the tests are failing: the simulated Chef run doesn't end well because a search can't be executed. How it works... Virtual machine configuration is the last thing done by Vagrant before starting up. In this case, we'll simply apply the starter
cookbook shipped with the starter kit, but we can add as many required roles and cookbooks as we want. A 64-bits Precise box released by Ubuntu will be named ubuntu/precise64 while the centos/7 box will always be the latest CentOS 7 official box. We'd like to begin by at least compiling the program, either for the Linux platform or for the Mac
operating system. However, we can run our own registry or use an alternative registry such as . This time it's an unstable, 0.8.0-rc1 version. The current Chef DK version is 1.1.16. We'll see here how to fully bootstrap a CentOS 7.2 instance on AWS with Terraform, from nothing to a fully configured node, by gracefully handing over the configuration
to Chef after having it automatically deployed and registered on Hosted Chef. @rocha mysql We can install one of them: $ puppet module install puppetlabs-mysql Notice: Downloading from ... Using cloud-init on Digital Ocean The situation is similar on Digital Ocean. Vagrant is compatible with most hypervisors, such as VMware, VirtualBox, or
Parallels, and can be largely extended using plugins. Most examples from previous chapters on Terraform were on Amazon Web Services, so to try to be more diverse and complete, this chapter is dedicated to other infrastructure services, so to try to be more diverse and complete, this chapter is dedicated to other infrastructure services, so to try to be more diverse and complete, this chapter is dedicated to other infrastructure services, so to try to be more diverse and complete, this chapter is dedicated to other infrastructure services, and the chapter is dedicated to other infrastructure services, so to try to be more diverse and complete, this chapter is dedicated to other infrastructure services, and the chapter is dedicated to other infrastructure services.
then be mysite/test/integration/default/serverspec. Let's start with the quickest and easiest—Cookstyle. It's very simple with Terraform. Pierre Pomès is a senior enthusiastic engineer of open source technologies and a Linux adept since 1994. When editing with eyaml edit, all the new values should be contained in a DEC::PKCS7[value]! block. It can
include immediate full updating of the OS, initial registration on discovery systems such as Consul, or initial addition of local DNS servers. However, that's totally optional and opinionated: resource "aws iam user" "logs" { user = "${aws iam user.logs.name}" } resource
"aws iam user policy attachment" "logs cloudwatch full" { user = "${aws iam user.logs.name}" policy arn = "arn:aws:policy/CloudWatchLogsFullAccess" } The relevant outputs.tf is as follows: output "logs" { value = "ACCESS KEY: ${aws iam access key.logs.iad}, SECRET: ${aws iam access key.logs.secret}" } Now, terraform
apply this and try the Enabling CloudWatch Logs for Docker with Terraform recipe again with those credentials instead of the master keys: it will still work on the CloudWatch scope, but if something goes wrong, it will never put the rest of your infrastructure in danger. Chef cookbooks Chef cookbooks are used to group many recipes under a coherent
set, as well as every other file required to make it work. 159.203.189.142 provider=digitalocean, region=ams 59e15b88... 287 Maintaining Systems Using Chef and Puppet All recipes are based on Chef. Our virtual machine will run in the production environment and apply the docker role: config.vm.provision "chef zero" do |chef| chef.cookbooks path
= "cookbooks" chef.environments_path = "roles" chef.nodes_path = "roles" chef.nodes_path = "roles" chef.environment = "production" chef.add_role "docker" end 330 Chapter 8 We're almost done! We need to tell the Vagrant Berkshelf plugin where to look for Berksfile and whether to enable it (installing the Berkshelf plugin is easy: vagrant
plugin install vagrant-berkshelf). For example, on Linux systems, this can be done by appending the following line in the .bashrc file located in your home directory: export PATH=/opt/puppetlabs/puppet/bin:$PATH See also ff The Chef documentation on the Chef DK: devkit/ Creating
 a free hosted server Chef account and a Puppet server In the preferred Chef client/server mode, we need a Chef server to centralize all the information and action. What we need for a fully usable CoreOS cluster on the local network
interface ($private_ipv4) Fleet is a distributed init system. In our case, we can associate the prebuilt IAM policy, named CloudWatchLogsFullAccess, with a new dedicated user in order to be up and running quickly in a secured way. It's very useful to keep some logs centralized, share access to them, receive alarms when errors happen, or simply store
them safely. Converging 1 resources Recipe: apache::default * yum package [httpd] action install - install version 2.4.6-40.el7.centos.4 of package httpd Running handlers: Running handlers complete Chef Client finished, 1/1 resources updated in 32 seconds Chef just installed the Apache HTTP server package for us! If we launch the Chef client, it
 won't install it again, as it knows it's already there (look at the largely different execution times): $ sudo chef-client [...] Recipe: apache::default * yum_package[httpd] action install (up to date) [...] Chef Client finished, 0/1 resources updated in 04 seconds 207 Fundamentals of Managing Servers with Chef and Puppet Verify if the package is really
installed: $ which httpd /usr/sbin/httpd $ httpd -v Server version: Apache/2.4.6 (CentOS) Server built: Jul 18 2016 15:30:14 Creating a MariaDB cookbook Let's use our knowledge to create a MariaDB cookbook the same way we just deployed Apache, from the Chef repository: $ chef generate cookbook cookbooks/mariadb We want to install two
packages: mariadb for the client and the libraries, and mariadbserver for the server. A normally operating infrastructure should always be in a state where a terraform plan doesn't want to change anything. Test Kitchen integrates very well with Berkshelf, so cookbook dependencies aren't an issue while testing complex infrastructures. It's dozens of
times smaller in size than other main distributions: less than 5 MB. There's a few ways to get information out of Docker. Integrating dynamic StatusCake monitoring to the hosts of our CoreOS cluster, using a free StatusCake account (). If you use a custom
built image with the Chef client already baked in, you will be interested in setting skip install to true so it doesn't get reinstalled. It takes a reference to the app to link the add-on to, and a plan (hobby-dev is free, so let's use that): resource "heroku addon" "redis" { app = "${heroku app.hubot.name}" plan = "heroku-redis:hobby-dev" } Using Heroku
 with Terraform It's out of the scope of this book to show Heroku usage, but let's apply this terraform code: $ terraform apply [...] Outputs: 155 Automating Complete Infrastructures with Terraform the end of the scope of this book to show Heroku usage, but let's apply this terraform apply [...] Outputs: 155 Automating Complete Infrastructures with Terraform the end of the scope of this book to show Heroku usage, but let's apply this terraform apply [...] Outputs: 155 Automating Complete Infrastructures with Terraform the end of the scope of this book to show Heroku usage, but let's apply this terraform apply [...] Outputs: 155 Automating Complete Infrastructures with Terraform the end of the scope of this book to show Heroku usage, but let's apply this terraform apply [...] Outputs: 155 Automating Complete Infrastructures with Terraform the end of the scope of this book to show Heroku usage, but let's apply this terraform the end of the scope of this book to show Heroku usage, but let's apply this terraform the end of the scope of this book to show Heroku usage, but let's apply this terraform the end of the scope of this book to show Heroku usage, but let's apply this terraform the end of the scope of this book to show Heroku usage, but let's apply this terraform the end of the
Now that we have our image built with NGINX, execute it. Our only requirement is that the CI platform should build and execute Docker containers and run RSpec tests. Running the Chef client as a daemon (such as memory or security requirements). These topics are all
 equally important, as they will define the quality of our daily work and our ability to interact with other people, services, and systems. Getting ready To step through this recipe, you will need the following: ff ff ff A working Terraform installation An AWS provider, a SSH key pair, and a Security Group configured in Terraform (refer to the previous
recipes) An Internet connection How to do it... First, you need to find the correct AMI for your machine. Add the following to recipes/default.rb: include recipe 'chef-client' 2. 381 Maintaining Docker Containers Click on Create Automated Build in the Create menu: Choose the provider where the infrastructure code is hosted; in our case, it's GitHub
382 Chapter 10 When the synchronization is done, choose the GitHub repository: Finally, decide on a name for the image (it doesn't have to be the name of the GitHub repository) and the namespace. To start using the OpenJDK 9 image is 548
MB—one of the smallest programming language images available. This basically offers nothing less than a full-fledged strace for a container; imagine the debugging power it has: The F8/Actions feature is a full Docker command integration tool available right from inside sysdig. From the workstation setup to writing our own recipes to managing
external cookbooks, this chapter contains it all—we'll manage packages, services, files, dynamic templates, dependencies, relationships, shared data, and more, all using code. Another possibility is to compress the file using gzip, or even to base64 encode the resulting compressed gzip file. It will be required in all of our future tests: require
 'spec helper' All unit tests start with a descriptive block, as given here: describe 'cookbook::recipe name' do [...] end 260 Chapter 7 Inside this block, we want to simulate the CoreOS AMI variable: variable "aws coreos ami" { type = "map" default = { eu-west-1
 "ami-85097ff6" us-east-1 = "ami-0aef8e1d" } } Define a cluster size variable with different values according to the environment: variable "cluster_size" { type = "map" default = { staging = "1" production = "3" } description = "Number of nodes in the cluster_size" { type = "map" default = { staging = "1" production = "3" } description = "Number of nodes in the cluster_size" }
variable "aws_instance_type" { type = "map" default = { staging = "t2.micro" production = "Instance type" } Now let's use those in a highly dynamic infrastructure code (instance type according to the environment
while choosing the right AMI according to the execution region: resource "aws_instance" "coreos" { count = "${lookup(var.aws_region)}" instance_type, var.environment)}" ami = "${lookup(var.aws_region)}" instance_type, var.environment)}" instance_type, var.environment, var.envir
associate public ip address = true tags { Name = "coreos ${var.environment} } $ We constructed each instance Name tag according to its environment and its numerical value in the count (that is, coreos production 2). Getting ready To step through this recipe, you will need the following
ff ff ff A working Chef installation on the workstation A working Chef client configuration on the remote host The Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamental 6, Fundamenta
through this recipe, you will need the following: ff A working Terraform (refer to the previous recipes) ff An Internet connection How to do it... Thankfully, we can use the same syntax we're already using to access variables and
attributes of references, but this time in an output resource. In this chapter, we will present the most useful use cases with cloud-init, such as copying files, creating users, managing SSH keys, adding repositories and installing packages, running arbitrary commands, bootstrapping a Chef client, or managing CoreOS and Docker with it. Start by
creating the Makefile main "help" entries: plan: terraform-validate te
Vagrant box may now take longer to boot, as it may need to download and install the right guest additions for the box. Start by creating the platform 2. 407 Maintaining Docker Containers These are just a few of the many powerful things we can do with Sysdig using Docker
 containers. 188 Chapter 5 Here's the unit in the cloud-config.yml file: coreos: units: - name: settimezone.service command: start content: | [Unit] Description=Setting the timezone [Service] ExecStart=/usr/bin/timedatectl set-timezone America/New York RemainAfterExit=yes Type=oneshot Enabling Docker TCP socket for network access Our final
objective is to be able to use a Docker Engine remotely from our workstation. We just need to add the vcsrepo module to Puppetfile install): mod 'puppetfile install insta
examples show that it is possible to simulate a complete infrastructure using a simple vagrant up command, and therefore, it is easy to put it into any CI system to ensure you will be able to rebuild your production system easily. First, install the git package: package {'git': ensure => installed, } Then, create a database for WordPress: mysql::db
 {'wordpress': user => 'wordpress_user', password => 'changeme', host => '%', grant => 'ALL', } Also, update our virtual definition to change DocumentRoot: apache::vhost { 'web.pomes.pro': website => 'web.pomes.pro': website => 'web.pomes.pro': website => 'web.pomes.pro' docroot => '/var/www/wordpress', } And finally, install WordPress from git and give rights to Apache: vcsrepo
 \{'' ('/var/www/wordpress': ensure => latest, provider => git, source => ', revision => 'master', } file \{'' ('/var/www/wordpress': ensure => true, } 319 Maintaining Systems Using Chef and Puppet See also ff The deploy resource documentation at deploy. html ff The PuppetLabs vcsrepc
 module at puppetlabs/vcsrepo ff The WordPress hardening quide at Hardening WordPress Using a TDD workflow TDD is a popular technique in development teams that consists of this: you begin by writing tests that will fail because no code is actually written and then write the code that will make these tests pass. ff For the
Follow these steps for creating functional roles: 1. Getting ready To step through this recipe, you will need: ff A working Chef DK installation on the workstation ff A working Vagrant installation on the workstation ff A working Chef DK installation on the workstation ff The Chef code (optionally) from Chapter 7, Testing and Writing
 Better Infrastructure Code with Chef and Puppet, or any custom Chef code How to do it... In a fresh and minimal installation of a CentOS 7.2 virtual machine, the ohai output is 5,292 lines long, which is full of information. Cloud-init is defined as a simple YAML file (cloud-config), sent in the user-data field of a cloud instance. There's a neat Vagrant
sharing feature we'll use here, working with a Ghost blog on CentOS 7.2. 33 Vagrant Development Environments Getting ready To step through this recipe, you will need the following: ff A working VirtualBox installation ff A free HashiCorp Atlas account (new) ff An Internet connection How to do it... Let's start with
this simple Vagrantfile: Vagrant.config.vm.box = "bento/centos-7.2" config.vm.box = "bento-period config.v
Ghost blog, such as adding themes and more, so let's sync our src/ folder to the remote /srv/blog folder: config.vm.synced_folder "src/", "/srv/blog" We want a local private network so we can access the virtual machine, with the 2368 TCP port (Ghost default) redirected to our host 8080 HTTP port: config.vm.network "private_network", type: "dhcp'
config.vm.network "forwarded_port", guest: 2368, host: 8080 Provisioning 1. It's true in software code, and it's the same in infrastructure code as well. That's the firewall, it can be handy. So, by planning before applying, we can know
 what's about to happen to our infrastructure. ff How should etcd initially advertise to the rest of the cluster? Alternatively, maybe you'd like to access your work Vagrant box from home, hotel, or coworking space? This way, we can create cookbooks with a purpose, like a product or an end result. 249 Testing and Writing Better Infrastructure Code
with Chef and Puppet In this chapter, you'll discover various techniques to write cleaner code using linters and styling tools, so our code follows high standards. Now generate the new mysql recipe so we can use it: $ chef generate recipe cookbooks/mysite mysql 241 Fundamentals of Managing Servers with Chef and Puppet 8. There's more... With
Puppet, there is no deploy resource. [...] 35 Vagrant Development Environments ==> srv-1: Your Vagrant Share is running! Name: anxious-cougar-6317 ==> srv-1: URL: This URL is the one you can give to anyone to access publicly your work: Vagrant Share is running! Name: anxious-cougar-6317 ==> srv-1: URL: This URL is the one you can give to anyone to access publicly your work: Vagrant Share is running! Name: anxious-cougar-6317 ==> srv-1: URL: This URL is the one you can give to anyone to access publicly your work: Vagrant Share is running! Name: anxious-cougar-6317 ==> srv-1: URL: This URL is the one you can give to anyone to access publicly your work: Vagrant Share is running! Name: anxious-cougar-6317 ==> srv-1: URL: This URL is the one you can give to anyone to access publicly your work: Vagrant Share is running! Name: anxious-cougar-6317 ==> srv-1: URL: This URL is the one you can give to anyone to access publicly your work: Vagrant Share is running! Name: anxious-cougar-6317 ==> srv-1: URL: This URL is the one you can give to access publicly your work: Vagrant Share is running! Name: anxious-cougar-6317 ==> srv-1: URL: This URL is the one you can give to access publicly your work: Vagrant Share is running! Name: anxious-cougar-6317 ==> srv-1: URL: This URL is the one you can give to access publicly your work: Vagrant Share is running! Name: anxious-cougar-6317 ==> srv-1: URL is the one you can give to access publicly your work: Vagrant Share is running! Name: anxious-cougar-6317 ==> srv-1: URL is the one you can give to access publicly your work: Vagrant Share is running! Name: anxious-cougar-6317 ==> srv-1: URL is the one you can give to access publicly your work: Vagrant Share is running! Name: anxious-cougar-6317 ==> srv-1: URL is the one you can give to access publicly your work: Vagrant Share is running! Name: anxious-cougar-6317 ==> srv-1: URL is the one you can give to access publicly your work.
in a Dockerfile: FROM php:7 ENTRYPOINT ["/bin/bash"] The main php:7 image is 363 MB—this is the smallest programming language image available. It's necessary in our case to have a public IPv4, because we need to access the servers by SSH from outside. The preceding style perfectly works, but style might not be canonical. We'd like our users
to be able to request one of the following: ff terraform:latest (for those of our users who always want the stable version, opposed to a beta version) ff terraform:stable (for those of our users who always want the stable version, opposed to a beta version) ff terraform:0.7.12 (for those of our users who always want the stable version, opposed to a beta version) ff terraform:stable (for those of our users who always want the stable version, opposed to a beta version) ff terraform:stable (for those of our users who always want the stable version, opposed to a beta version) ff terraform:stable (for those of our users who always want the stable version) ff terraform:stable (for those of our users who always want the stable version) ff terraform:stable (for those of our users who always want the stable version) ff terraform:stable (for those of our users who always want the stable version) ff terraform:stable (for those of our users who always want the stable version) ff terraform:stable (for those of our users who always want the stable version) ff terraform:stable (for those of our users who always want the stable version) ff terraform:stable (for those of our users who always want the stable version) ff terraform:stable (for those of our users who always want the stable version) ff terraform:stable (for those of our users who always want the stable version) ff terraform:stable (for those of our users who always want the stable version) ff terraform:stable (for those of our users who always want the stable version) ff terraform:stable (for those of our users who always want the stable version) ff terraform:stable (for those of our users who always want the stable version) ff terraform:stable (for those of our users who always want the stable version) ff terraform:stable (for those of our users who always want the stable version) ff terraform:stable (for those of our users who always want the stable version) ff terraform:stable (for those of our users who always want the stable version) ff terraform:stable (for th
345 Working with Docker This is easily achievable by building directly with all these different tags: $ docker build -t terraform:0.7.12 . The repository should look like this: 380 Chapter 10 This repository is ready to communicate with other build services. made it simple for us to deploy it; it's extensively
documented and packaged. You need to use dedicated images for each environment. Let's write the code to inject these values to the aws.rb recipe. Doing non-kernel setup of the Guest Additions ...done. Starting from a PHP image The
PHP Docker image is very popular, and available in many different flavors. Concise and elegant! Provisioning a CentOS 7 EC2 instance with Chef using Terraform Once the underlying infrastructure is generated by Terraform.
class { 'baseconfig': agentmode => 'cron'; } ... For ease of deployment, let's create Gemfile containing our dependencies inside a test group: source ' 'group :test do gem 'serverspec' gem 'docker-api' end 374 Chapter 10 Install these dependencies using Bundler: $ bundle install Using docker-api 1.33.0 Using serverspec 2.37.2 [...] Bundle complete! 2
Gemfile dependencies, 18 gems now installed. What if the application servers get corrupted tonight? Let's transform the default recipe 'chef-client on the target host. /usr/src/app The next
step is to precompile the assets, with a RAILS ENV set to production, but feel free to adapt, including the compilation command: RUN RAILS ENV=production rake assets; precompile Finally, run the Rails server on all interfaces through bundler (by default, it listens on TCP/3000): CMD ["bundle", "exec", "rails", "server", "-b", "0.0.0.0"] We can now
build this Dockerfile and have our complete, standalone, and fully working Rubyon-Rails application ready on Docker. 29 Vagrant Development Environments Showing your work on the LAN while working with Laravel You're working on your application using Laravel, the free and open source PHP framework (, and you'd like to showcase your work
to your colleagues. Include them in deploy.rb: include recipe 'apache::virtualhost' include recipe 'apache::wirtualhost' i
link/ether 02:42:ac:12:00:02 brd ff:ff:ff:ff:ff inet 172.18.0.2/16 scope global eth1 [...] This means that it's also available to answer requests from containers on this network after we're done with it: $ docker network disconnect bridge hello Creating
more dynamic containers We can create better containers than just fixing their usage in advance and executing them. Create a module/apache/manifests/init.pp manifest file with the following content: class apache { package {'apache2': ensure => present, } } 5. I propose, that you search for every node using search(*:*); however, feel free to limit to
whatever is more secure or appropriate for you, such as tags or roles, like search(role:mysite): $ knife vault create aws eu-west-1.json data bags/aws/eu-west-1.json data bags/aws/eu-west-1.
The companion to knife vault is the chef-vault cookbook. Puppet Forge hosts a lot of community modules, and some of them are supported by Puppet Labs. 315 Maintaining Systems Using Chef and Puppet below from git. We'll include the symlinks
configuration and initialize them to nothing so the code is already present when the time arises. Here's how the playbook.yml file looks: --- hosts: all become: yes tasks: - name: Enable EPEL yum: name=epel-release state=present - name: Add the vagrant user to Docker group user:
name=vagrant groups=docker append=yes - name: Install Docker yum: name=docker state=present - name: Enable and Start Docker Daemon service: name=docker state=present - name: Enable and Start Docker Daemon service: name=docker state=present - name: Enable and Start Docker Daemon service: name=docker state=present - name: Enable and Start Docker Daemon service: name=docker state=present - name: Enable and Start Docker Daemon service: name=docker state=present - name: Enable and Start Docker Daemon service: name=docker state=present - name: Enable and Start Docker Daemon service: name=docker state=present - name: Enable and Start Docker Daemon service: name=docker state=present - name: Enable and Start Docker Daemon service: name=docker state=present - name: Enable and Start Docker Daemon service: name=docker state=present - name: Enable and Start Docker Daemon service: name=docker state=present - name: Enable and Start Docker Daemon service: name=docker state=present - name: Enable and Start Docker Daemon service: name=docker state=present - name: Enable and Start Docker Daemon service: name=docker state=present - name: Enable and Start Docker Daemon service: name=docker state=present - name: Enable and Start Docker Daemon service: name=docker state=present - name: Enable and Start Docker Daemon service: name = docker state=present - name: Enable and Start Docker Daemon service: name = docker state=present - name: Enable and Start Docker Daemon service: name = docker state=present - name: Enable and Start Docker Daemon service: name = docker state=present - name: Enable and Start Docker Daemon service: name = docker state=present - name: Enable and Start Docker Daemon service: name = docker state=present - name: Enable and Start Docker Daemon service: name = docker state=present - name: Enable and Start Docker Daemon service: name = docker state=present - name = doc
box: Vagrant.configure("2") do |config| config.vm.hox = "bento/centos-7.2" config.vm.hostname = "srv-1" do |config| con
ansible.playbook = "playbook.yml" end You can now run vagrant up and use CentOS 7 Docker Engine version Docker --version Dock
There's more... What if for some reason you don't or can't have Ansible installed on your host machine? All of these pieces of data are browsable and they help you gain more in-depth information about a specific container and its content and usage: cAdvisor can also be plugged in to many backend storage systems, such as Prometheus, ElasticSearch
 InfluxDB, Redis, statsD, and so on. Dependencies, shared variables, and common networking can be easily handled with the orchestration tool named Docker Compose. Now we're ready for prime time; there's a whole set of features, constraints, and objectives to be properly set. Berkshelf allows us to declare cookbook dependencies, versions and
locations in a single file, and in a single command, upload everything needed to run our cookbook. Plugins range from IaaS support such as AWS, OpenStack, VMware, or Digital Ocean to hardware management from Dell, HP, or IPMI interfaces, team workflow integration, or system-related concerns such as logs handling, security, and other similar
features. One of the included recipes uses an encrypted data bag. Countless books and methods exist on the subject and approaches are usually very opinionated. Based on previous examples, we could think about a mysite module, with the following manifest: class mysite ($website, $docroot) { class { 'apache':; 'php';; 'mariadb':; } apache::vhost
 {$website: website => $website, docroot => $docroot, } file { $docroot: ensure => 'www-data', group => 'www-data',
 'web.pomes.pro' { class { 'mysite': 225 Fundamentals of Managing Servers with Chef and Puppet website docroot => $fqdn, => "/var/www/$fqdn", } } See also ff The Chef documentation on attributes: . Ready for the next stage of deployment with Puppet! Using Docker with Terraform Terraform can also be used to manipulate Docker. How to do it.
To have a fixed command executed by the container, use the ENTRYPOINT instruction. The common point between all these experiences is that success comes with rigor, technical repeatability, communication, and a shared team culture. Then we execute Bundler (without the test and development sections if we have them). Using a disposable
 Ubuntu Xenial (16.04) in seconds We want to access and use an Ubuntu Xenial system (16.04 LTS) as quickly as possible. A Chef server is a combination of many technologies such as PostgreSQL, RabbitMQ, Redis, Nginx, and so on. A short name puppet is also available, and has been populated by the vagrant-hostmanager plugin. Execute our first
unit test using chef exec rspec (it's using the bundled rspec from the Chef DK): $ chef exec rspec --color Alternative ways of doing similar actions using Puppet code are also shown for you to have a better view of the ecosystem. It integrates very well with Berkshelf, so cookbook dependencies are easily handled during the testing process. Using
cloud-init on AWS, Digital Ocean, or OpenStack As cloud-init is an initialization system for cloud instances, we need to find a way to send the cloud-config YAML file to the bootstrapping process. ff Fully searchable across every book published by Packt ff Copy and paste, print, and bookmark content ff On demand and accessible via a web browser
Customer Feedback Thanks for purchasing this Packt book. A list of all available on the Vagrant providers is available on the Vagrant wiki here: . Enable the watching of the repository by Travis: 4. At the time of writing, images are hosted on Red Hat's Docker registry servers (. ff How should etcd advertise the client URLs to the rest of the rest of the repository by Travis: 4. At the time of writing, images are hosted on Red Hat's Docker registry servers (. ff How should etcd advertise the client URLs to the rest of the rest o
the cluster? Launch the whole infrastructure: $ terraform apply Outputs: Swarm Master Private IP = 10.80.86.135,10.80.86.135 Our cluster is running. There's more... How would that work when using Ansible to do a similar job with
the same values? Report DMCA Infrastructure as Code (IAC) Cookbook Over 90 practical, actionable recipes to automate, test, and manage your infrastructure as Code (IAC) Cookbook Copyright © 2017 Packt Publishing All rights reserved. In the cloud-
init logs, this step is found at this moment: [...] Starting Chef Client, version 12.14.89 Creating a new client identity for i-0913e870fb28af4bd using the validator key. To start a with the latest Fedora release, use the following in a Dockerfile: FROM fedora: latest ENTRYPOINT ["/bin/bash"] Starting from an Alpine Linux image Alpine Linux is a very
popular and secure lightweight Linux distribution in the container world. It's also easier to maintain, as changing the key will only require to replace the file and nothing more. Here's how to test if both actions are handled by the code using ChefSpec, inside the context created earlier: it 'enables and starts httpd service' do expect(chef_run).to
enable service('httpd') expect(chef run).to start service('httpd') end Our test coverage is now 100% for the default recipe as we tested both declared resources. By iterating over each droplet, and extracting their name and ipv4 address attributes, we'll add this digitalocean record resource into the mix: resource "ipv4" { count
 = "${var.cluster_nodes}" domain = "${digitalocean_droplet.coreos.*.name, count.index)}" } This will automatically register every droplet under the name core-[1,2,3].mydomain.com, for
 easier access and reference. kitchen verify: This first installs everything needed to run the tests—in our case, this will be ServerSpec. It, in fact, extended Chef functionalities by offering a mysql_service resource. Let's add some outputs in outputs.tf: output "HTTP Servers" { value = "${join(" ", google_compute_instance.www.*.network_and the tests—in our case, this will be ServerSpec. It, in fact, extended Chef functionalities by offering a mysql_service resource.
 interface.0.access config.0. assigned nat ip)}" } output "MySQL DB IP" { value = "${google sql database instance.master.ip address}" } output "Load Balancer Public IPv4" { value = "${google compute forwarding rule.http.ip address}" } output "DB Credentials" { value = "Username=${var.db username}}
 application on the HTTP servers and we're done! To test drive the load balancer and the HTTP instances, you can simply deploy the NGINX container on each server and see the traffic flow: $ sudo docker run -it --rm -p 80:80 --name web nginx Provisioning a GitLab CE + CI runners on OpenStack OpenStack is a very popular open source cloud
computing solution. You can think of it as systemd for a whole cluster To configure etcd, we first need to obtain a new token. This needs to change! We want a full-fledged MySQL deployment. Here is an extract of modules/apache/README.md: # Apache module ## Table of Contents 1. There're many ways to do this, from provisioning shell scripts to
 Environment="AWS_SECRET_ACCESS_KEY=SW+jdHKd.." Don't forget to reload the system daemon and restart Docker to apply the changes: $ sudo systemctl daemon-reload $ sudo systemctl restart docker We're now ready to talk to the AWS APIs through the Docker daemon. In this section, we'll show how to create a simple LAMP docker.
compose file, then we'll show how we can iterate from that to build some staging and production specific computing power for a task, or you just want to replicate part of an existing production environment: here's how you
can leverage the power of Vagrant using Amazon EC2. So let's add the instance argument to the cloud-init-per instance mkfs-xvdb mkfs -t ext4 /dev/xvdb on the /srv/ www folder: mounts: - [/dev/xvdb / /srv/www] After
boot, let's verify the block device is mounted: # df -h /srv/www We can also test the existence of the file we created: # cat /etc/instance id i-03005dd324599df11 Try to delete this file and reboot the server: the file will be there again. 194 Chapter 6 There is also a
main manifest, outside any module, which is the list of nodes of the infrastructure. Here's what our sample recipe looks like (cookbooks/ apache2/recipes/default.rb); it does exactly what it says in plain English: package "httpd" do action [:enable, :start] end Vagrant and Chef integration Here's how, in our VM definition block, we'll tel
Vagrant to work with Chef Solo (a way of running Chef in standalone mode, without the need of a Chef server) to provision our box: config.vm.provision our box: c
to all your coworkers on your network, and the code will be accessible to you locally. A default configuration is provided with the Puppet server installation and is located at /etc/puppetlabs/puppet/hiera.yaml: -:backends: - yaml shierarchy: - "nodes/% {::trusted.certname}" - common :yaml: :datadir: 302 Chapter 8 Here, a yaml backend is defined.
allowing us to use yaml files in the hieradata directory of our environments. Let's add them to a commit: $ git add . We've already covered Test Kitchen, and this might be a good solution, especially if we have written extensive tests. Let's do that: resource "docker container" "nginx-proxy"
proxy.latest}" ports { internal = 80 external = 80 external = 80 protocol = "tcp" } volumes { host path = "/tmp/docker.sock" read only = true } } Now if you terraform apply this, you can navigate over to (replace localhost with the Docker server you used) and set up your Ghost blog! 97 Going Further with Terraform
Simulating infrastructure changes using Terraform In an earlier recipe, you learned how to manage different environments with Terraform, which is great. A shared folder will be used to edit code directly from your workstation. Chapter 7, Testing and Writing Better Infrastructure Code with Chef and Puppet, is all about advanced techniques of
testing code for quality and sustainability. Displaying encrypted data. It's also highly recommended that you include any other test that can be done in this CI system, such as the Docker linters check from earlier in this book. Getting ready To step through this recipe, you will need the following: ff A working Vagrant installation ff A working
hypervisor ff An Internet connection How to do it... CoreOS doesn't host its official images at the default location on Atlas, it hosts it itself. By default, the Puppet agent is looking for a server named puppet—that's why this name has been defined as an alias to the puppet server. [...] :~$ Another way of including file content is to
encode it in base 64. Similarly, we usually expect a set of tools to be available in the new system. We also have other code bundles from our rich catalog of books and videos available at PacktPublishing/. Using Docker directly, we would have run the following: $ docker run -d --name mariadb -e MYSOL ROOT PASSWORD=h4ckm3 mariadb $ docker
run -d -e WORDPRESS DB HOST=mariadb -e 'WORDPRESS DB PASSWORD=h4ckm3' config.vm.provision "docker" do |docker| docker.run "mariadb", args: "--name 'mariadb' -e 'MYSQL ROOT PASSWORD=#{db root password}"
docker.run "wordpress", args: "-e 'WORDPRESS DB HOST=mariadb' -e 'WORDPRESS DB HOST=mariadb'
Apache/2.4.10 (Debian) X-Powered-By: PHP/5.6.25 Expires: Wed, 11 Jan 1984 05:00:00 GMT Cache-Control: no-cache, must-revalidate, max-age=0 Content-Type: text/html; charset=utf-8 45 Vagrant Development Environments There's more... The CoreOS team proposes a full Vagrant environment to try and manipulate a CoreOS cluster . 342 Chapter
9 Starting from a Java image Java users are also getting official releases on Docker. Here's how we would simply upload our previous index.html file using the same s3 module: - name: create index.html file using the same s
host your Docker images, you need what's called a registry. Done Building dependency tree Reading state information... @aco mysql BoxUpp-mysql A puppet modu... The Fluentd unified logging layer is also supported, while on the platform front, we find support for Splunk and Google Cloud together with AWS CloudWatch logs. Getting ready To step
through this recipe, you will need the following: ff ff A working Terraform installation An AWS account with an SSH key configured in Terraform, recipes) 85 Going Further with Terraform ff ff An account on a Chef server (we
recommend using a free hosted Chef account. Then we'll start 2 units—etcd and fleet. Don't hesitate to look at it, it's a text file. Getting ready To work through this recipe, you will need the following: ff A working Chef client configuration on the remote host ff Optionally, the Chef code from the
previous recipes How to do it... We'll manage two different kinds of files in two different ways: a static file and a dynamic file generated from a template, so the most common usage is covered. Let's exclude globally all supermarket-related warnings using the -t ~supermarket switch: $ foodcritic -t ~FC033 -t ~supermarket --exclude test
cookbooks/mysite/ No more warnings now; our cookbook is following the best advice on the planet from both Chef and the Foodcritic community! It's highly recommended that you add those tests to your automated testing process. A sample Chef recipe This part of the book isn't about writing Chef recipes (read more about it later in the book!), so
we'll keep that part simple. When connecting to various remote Docker Engines, we will sooner or later connect to a server not using the same version of the server as our client. The new manifest for the Apache => true } } The new
manifest for the MariaDB module is: class mariadb { package {'mariadb-server': ensure => running, enable => true } } 215 Fundamentals of Managing Servers with Chef and Puppet The ensure => running, enable => true } } 215 Fundamentals of Managing Servers with Chef and Puppet The ensure => running, enable => true } }
start the service at boot. We can access the content of this array using the formatlist interpolation syntax as follows: output "security groups" { value = "${formatlist("%v", aws instance.dev.security groups: security groups: security groups: } } 60 Chapter 2 So now, at the next terraform apply, we'll instantly know our security groups: securi
if we have a lot of information to display from multiple sources, we can use the same syntax: output "instance information" { value = "${formatlist("instance dev.*.private ip)}" } This will display the instance ID and its local and public IP addresses. is a bare
metal IaaS provider (is another) very well supported by Terraform with an awesome global network. Using all the variables we declared, here's how it looks: resource "aws instance" "centos" { [...] provisioner "chef" { node name = "centos-${count.index+1}" run list = ["starter"] server url = "${var.chef server url}" validation client name = "centos-${count.index+1}" run list = ["starter"] server url = "${var.chef server url}" validation client name = "centos-${count.index+1}" run list = ["starter"] server url = "${var.chef server url}" validation client name = "centos-${count.index+1}" run list = ["starter"] server url = "${var.chef server url}" validation client name = "centos-${count.index+1}" run list = ["starter"] server url = "${var.chef server url}" validation client name = "centos-${count.index+1}" run list = ["starter"] server url = "${var.chef server url}" validation client name = "centos-${count.index+1}" run list = ["starter"] server url = "${var.chef server url}" validation client name = "centos-${count.index+1}" run list = ["starter"] server url = "${var.chef server url}" validation client name = "centos-${count.index+1}" run list = ["starter"] server url = "${var.chef server url}" validation client name = "centos-${count.index+1}" run list = ["starter"] server url = "${var.chef server url}" validation client name = "centos-${count.index+1}" run list = ["starter"] server url = "${var.chef server url}" validation client name = "centos-${count.index+1}" run list = ["starter"] server url = "${var.chef server url}" validation client name = "centos-${count.index+1}" run list = ["starter"] server url = "${var.chef server url}" run list = ["starter"] server url = "starter"] se
"${var.chef validation client name}" validation key = "${file("chef/validator.pem")}" version = "${var.chef var.chef var.chef version}" } } Don't forget to use a valid path for the validation key! Now you can terraform apply this and see everything happen, from instance creation to Chef Client deployment and cookbook installation. Let's now execute our container to
make a DNS request using dig: $ docker run -it --rm utils dig +short google.com 172.217.5.14 Alternatively, we may use curl as follows: $ docker run -it --rm utils curl -I google.com HTTP/1.1 302 Found Cache-Control: private Content-Type: text/html; charset=UTF-8 Location: Content-Length: 258 Date: Wed, 23 Nov 2016 02:34:58 GMT Using a
private registry When not specifying anything else than the container name, Docker is looking for it locally, then on Docker Hub (). It's one of the easiest ways of easily testing newer and older releases of PHP on a platform. This is getting increasingly important as every project or team grows, and it's getting riskier to ship untested containers into
production. Let's create countless repositories simply with Terraform! Getting ready To step through this recipe, you will need the following: ff A working Terraform installation ff An AWS provider configured in Terraform! Getting ready To step through this recipe, you will need the following: ff A working Terraform installation ff An AWS provider configured in Terraform!
do it... Let's say you want to store your application container in a repository named myapp, so you can deploy it easily. If, for some reason, the Dockerfile of our project wasn't at the root, we could specify it here. Here's a table of the required information: 230 Chapter 6 User John Password p4ssw0rd Hash $apr1$AUI2Y5pj$0v0PaSlLfc6QxZx1Vx5Se
Mary s3cur3 $apr1$eR7H0C5r$OrhOQUTXfUEIdvWyeGGGy/ To generate the encrypted passwords, you can use the simple htpasswd utility: $ htpasswd -n -b mary s3cur3 mary:$apr1$eR7H0C5r$OrhOQUTXfUEIdvWyeGGGy/ We want to store that piece of information (username and password), inside a single entity: this is the data bag. See also ff
Bats testing framework: ff RSpec: ff ServerSpec: ff
generate recipe cookbooks/apache virtualhost. A new file named apache/recipes/virtualhost.rb is now created. The r10k tool also takes care of the Puppetfile file in each branch, as we already saw in Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, and deploys external modules, if any. Getting ready To step through this recipe, you
will need the following: ff A working Docker installation ff A Golang application source code How to do it... Let's say our application code is checked in src/hello. Refer to the Docker section of this book for more information on the command-line options used: $ docker -H 52.211.117.98 run -it --rm -p 80:80 nginx Unable to find image
'nginx:latest' locally latest: Pulling from library/nginx 6a5a5368e0c2: Pull complete 4aceccff346f: Pull complete 2aceccff346f: Pull complete 2acecc
612 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10 11 6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.116 Safari/537.36" "-" Make some requests on the remote Docker host HTTP port and it will answer. Here are the most useful ones for daily use. In this case, we want to obtain a container always executing /usr/sbin/apache2ctl, and
by default starting the daemon in foreground, otherwise overridden by any argument at container launch time: FROM debian:stable-slim RUN apt-get update -y && \ apt-get install -y apache2ctl"] If this container is executed as is, nothing changes;
apache2ctl gets executed with the -D FOREGROUND option. Run the Chef client on the node and see the magic happen: $ sudo chef-client [...] * template[/etc/httpd/conf.d/defaultsite.conf] action create - update content in file /etc/httpd/conf.d/defaultsite.conf] action create - update content in file /etc/httpd/conf.d/def
10-17 227 Fundamentals of Managing Servers with Chef and Puppet +++ /etc/httpd/conf.d/.chef-defaultsite DocumentRoot /var/www/defaultsite ErrorLog /var/log/httpd/error-defaultsite.log [...] Recipe: apache::default * service[httpd]
 action reload - reload service service[httpd] The cool thing is we can even see a diff of the change in the logs so we always know what's changed, as well as see the httpd service being reloaded after the change in the logs so we always know what's changed, as well as see the httpd service being reloaded after the change in the logs so we always know what's changed, as well as see the httpd service being reloaded after the change in the logs so we always know what's changed, as well as see the httpd service being reloaded after the change in the logs so we always know what's changed, as well as see the httpd service being reloaded after the change in the logs so we always know what's changed in the logs so we always know what's changed in the logs so we always know what's changed in the logs so we always know what's changed in the logs so we always know what's changed in the logs so we always know what's changed in the logs so we always know what's changed in the logs so we always know what's changed in the logs so we always know what's changed in the logs so we always know what's changed in the logs so we always know what's changed in the logs so we always know what's changed in the logs so we always know what's changed in the logs so we always know what we place the logs so we always know what we have a log w
"${var.coreos channel}" count = "${var.cluster nodes}" name = "coreos-${count.index+1}" region = "${var.do droplet size}" ssh key. = ["${digitalocean ssh key.default.id}"] private networking = true backups = false user data = "${file("cloud-config.yml")}" } Declare all the variables in the variables.tf file, with some
good defaults (the smallest 512 MB droplet, a three-node cluster), and some defaults we'll want to override (AMS3 data center or the stable CoreOS channel): variable "do droplet size" { default = "512mb"
description = "Droplet Size" } variable "coreos channel" { default = "Coreos channel" } variable "cluster nodes in the cluster" } Here are our overridden values in terraform.tfvars (but feel free to put your own values, such as using another data center or CoreOS release):
do_region = "nyc1" coreos channel = "coreos-beta" Adding useful output It would be awesome to automatically have a few auto-documented lines on how to connect to our CoreOS cluster. For the record, Google Cloud and Azure storage are also supported. On top of this app, we'll automatically plug a Heroku add-on (redis) and deploy everything.
Finished in 20.22 seconds (files took 1 minute 24.54 seconds to load) 8 examples, 0 failures We now have a complete acceptance test suite for our Apache module! See also ff ServerSpec GitHub: ff ServerSpec Homepage: ff A sample skeleton for Puppet module with Beaker enabled: joshbeard/puppet-module-test 286 8
Automating application deployment (a WordPress example) Using a TDD workflow Planning for the worse - train to rebuild working systems Introduction We've previously seen how to automate systems with code and how to properly test this code. On the Puppet server, we just need to do this: $ sudo -s # cd /etc/puppetlabs/code/environments/ # cp -
a production test Node environment selection On the node side, the environment to use can be controlled using --environment; # puppet agent --test --environment test To use this environment to use can be controlled using --environment = test
Getting the environment from manifests As for Chef, we can get the name of the running environment from any manifest. Here's how to test it using ChefSpec, inside the context we created earlier: it 'installs httpd' do expect(chef run).to install package('httpd') end 262 Chapter 7 Execute rspec again and see the touch coverage attain 50% as one of
the two resources from the default recipe is now tested. We'll want to isolate environments such as dev, staging, and production. To start from the Ruby 2.3 image is 725 MB. Our manual tests can include checking that the chef-solo version is
the one we requested: $ chef-solo --version Chef: 11.18.12 They can also check if we have httpd installed: $ httpd is running: $ pidof httpd 13029 13028 13027 13026 13025 13024 Various other options than chef-solo exist, such as chef-client and chef-zero. Getting
ready To work through this recipe, you will need the following: ff A working Chef client configuration on the remote host ff The Chef code from the previous recipes How to do it... Let's start by discovering how Berkshelf works. Verify the connection to Hosted Chef using the knife command and
request, for example, the list of the users (this will return you user): $ cd chef-repo $ knife user list iacbook 9. This enables a lot of options (such as having traditional rsyslog setups and online services compatible with the traditional format).
will vary, but in this network, the public IP of this Vagrant box is 192.168.1.106, so our work is available. To illustrate, let's start from the apache cookbook from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, but any similar custom cookbook from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, but any similar custom cookbook from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, but any similar custom cookbook from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, but any similar custom cookbook from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, but any similar custom cookbook from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, but any similar custom cookbook from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, but any similar custom cookbook from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, but any similar custom cookbook from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, but any similar custom cookbook from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, but any similar custom cookbook from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, but any similar custom cookbook from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, but any similar custom cookbook from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, but any similar custom cookbook from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet 6, Fundamentals of Managing Servers with Chef and Puppet 6, Fundamentals of Managing Servers with Chef and Puppet 6, Fundamentals of Managing Servers with Chef and Puppet 6, Fundamentals of Managing Servers with Chef and Puppet 6, Fundamentals of Managing Servers with Chef and Puppet 6, Fundamentals of Managing Servers with Chef and Puppet 6, Fundamentals of Managing Servers with Chef and Puppet 6, Fundamentals of Managing Servers with Chef and Puppet 6, Fundamentals wit
boxes with the right Chef-provisioning profiles for each use case: docker, webserver, database server, or full deployment. To install and use it, simply run the cAdvisor Docker image with volume=/:/rootfs:ro\--volume=//:
volume=/sys:/sys:ro \ --volume=/var/lib/docker:ro \ or. if using docker-compose: cadvisor: latest Or, if using docker-compose: cadvisor: latest restart: always 401
```

Maintaining Docker Containers Navigating to the host's 8080 port (or whatever port you choose to publish) with a web browser will present a web interface where we can navigate and see graphical information about container usage on the host: Or, we may have more general gauges giving live indication of resource usage: 402 Chapter 10 A very

```
useful process table with top-like data from the underlying host is also available with a container-aware context. 19 Vagrant Development Environments Getting ready To step through this recipe, you will need the following: ff A working Vagrant installation (no hypervisor needed) ff An Amazon EC2 account (or create one for free at if you don't have entered one free at if you don
one already), with valid Access Keys, a keypair named iac-lab, a security group named iac-lab allowing at least HTTP ports, and SSH access. Everything assembled together, intentions might match, but reality can be very different. This is because the default Puppet environment is production. We'll discover many important techniques to use
Terraform in conjunction with other players in the field such as Docker and Chef, how it can be used in multiple environments (such as development/staging/production), how powerful it can be to manage not only infrastructure but many SaaS as well, and how to integrate the tool within a team workflow (sharing, synchronizing, maintaining,
harmonizing, and so on). Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. It's very useful if you launch more than one instance using the count=n parameter in the aws instance Terraform resource. If you are author/publisher or own the copyright of this documents, please report to us by
using this DMCA report form. v Preface We hope you'll get much out of this book, and that automating and testing using infrastructure-as-code will be as fun for you as it's been for us to write about. Fortunately, a Puppet module, named puppet/hiera, exists to handle all of this for us. We can see that the MySQL module from Puppet Labs depends on
two other modules. remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 Receiving objects: 100% (3/3), done. Getting ready To step through this recipe, you will need: ff Access to a cloud-config enabled infrastructure How to do it... To upgrade all the packages right after bootstrap, simply set the package_upgrade directive to true: #cloud-config enabled infrastructure How to do it... To upgrade all the packages right after bootstrap, simply set the package_upgrade directive to true: #cloud-config enabled infrastructure How to do it... To upgrade all the packages right after bootstrap, simply set the package_upgrade directive to true: #cloud-config enabled infrastructure How to do it... To upgrade all the packages right after bootstrap, simply set the package_upgrade directive to true: #cloud-config enabled infrastructure How to do it... To upgrade all the packages right after bootstrap, simply set the package_upgrade directive to true: #cloud-config enabled infrastructure How to do it... To upgrade all the packages right after bootstrap, simply set the package_upgrade directive to true: #cloud-config enabled infrastructure How to do it... To upgrade all the packages right after bootstrap, simply set the package_upgrade directive to true: #cloud-config enabled infrastructure How to do it... To upgrade all the packages right after bootstrap, simply set the package_upgrade directive to true: #cloud-config enabled infrastructure How to do it... To upgrade all the packages right after bootstrap, simply set the package_upgrade directive to true: #cloud-config enabled infrastructure How to do it... To upgrade all the packages right after bootstrap, simply set the package upgrade directive to true: #cloud-config enabled infrastructure How to do it... To upgrade all the packages right after bootstrap, simply set the package upgrade all the packages right after bootstrap after bootstrap
config package upgrade: true 174 Chapter 5 Another useful directive is to reboot the system if required by the package manager (common case with kernel updates). This feature is dynamic: each new branch added to the Git repository will be deployed by r10k. What would have happened if we didn't provide a safe default for our variable? How it
works... This first encounter with HashiCorp Configuration Language (HCL), the language used by Terraform, and other Hashicorp products looks pretty familiar: it's a structured language fully compatible with JSON. Save this file as account.json at the root of the infrastructure repository. Better security with unprivileged users By default, containers
execute everything as the root user. We need to use the official AWS command line to authenticate, and that will give us temporary Docker credentials. On top of this, Ohai is a modular system with a lot of community plugins to integrate Dell DRAC information with support information related to a KVM, LXC, or XenServer. Let's start by creating a
 production environment in a production.rb file: name 'production' description 'The production environment' 2. 218 Chapter 6 A variable in an ERB template is prefixed with the @ character. There's more... While using Puppet, it is a good practice to store the credentials and site information in Hiera, as we saw in Chapter 6, Fundamentals of Managing
Servers with Chef and Puppet. Using Consult o store the Terraform states makes it easier to work with a team, as there's only a single replicated state. Running Nginx in a container Nginx is officially packaged as a Docker container. In this case, we'll get the following error: Error response from daemon: client is newer than server (client API version:
1.24, server API version: 1.22) The easy workaround is to override the DOCKER API VERSION environment variable and set it to the same value as the server (1.22 in this example): $ DOCKER API VERSION environment variable and set it to the same value as the server (1.22 in this example): $ DOCKER API VERSION environment variable and set it to the same value as the server (1.22 in this example): $ DOCKER API VERSION environment variable and set it to the same value as the server (1.22 in this example): $ DOCKER API VERSION environment variable and set it to the same value as the server (1.22 in this example): $ DOCKER API VERSION environment variable and set it to the same value as the server (1.22 in this example): $ DOCKER API VERSION environment variable and set it to the same value as the server (1.22 in this example): $ DOCKER API VERSION environment variable and set it to the same value as the server (1.22 in this example): $ DOCKER API VERSION environment variable and set it to the same value as the server (1.22 in this example): $ DOCKER API VERSION environment variable and set it to the same value as the server (1.22 in this example): $ DOCKER API VERSION environment variable and set it to the same value as the server (1.22 in this example): $ DOCKER API VERSION environment variable and set it to the same value as the server (1.22 in this example): $ DOCKER API VERSION environment variable and set it to the server (1.22 in this example): $ DOCKER API VERSION environment variable and set it to the server (1.22 in this example): $ DOCKER API VERSION environment variable and set it to the server (1.22 in this example): $ DOCKER API VERSION environment variable and set it to the server (1.22 in this example): $ DOCKER API VERSION environment variable and set it to the server (1.22 in this example) $ DOCKER API VERSION environment variable and set it to the server (1.22 in this example).
and the server directly in the CLI. There's more... Puppet has exactly the same feature, using the notify attribute. Let's try Ruby for our webserver role in roles/webserver.rb. To configure our new box, we'll first need to enable EPEL: sudo yum install -q -y epel-release 2. After applying Puppet, we can now use the MySQL server with the root
credentials we specified in the main manifest: :~$ mysql -h 127.0.0.1 -uroot psuper_secure_password -e "show databases;" +---------
 -----+ | Database | +------+ | information_schema | | mysql | | performance_schema | +-----
 ---+ If needed, you can browse the online documentation of this module to create custom
databases and grants. You can build from this example and others in this book to create an environment that mimics production (three-tier, multiple machines, and other characteristics). The splay option is used to introduce a random delay before the run itself, and this delay cannot exceed 60 minutes (the value of the splaylimit option). This VM uses
#{vm_cpus} CPUs and #{vm_memory}MB of RAM." Specify a minimum Vagrant version Vagrant is updated quite often, and new features are added regularly. In the case of the VirtualBox Guest Addition plugin, you can do a lot of things such as status checks, manage the installation, and much more: $ vagrant vbguest --status [default]
GuestAdditions 5.0.26 running --- OK. This recipe is using Ubuntu 14.04 LTS waiting for the issue to be fixed. 217 Fundamentals of Managing Servers with Chef and Puppet To store the name of our virtual host, let's create an attribute. Chapter 4, Automating Complete Infrastructures with Terraform, will show and describe complete, real-life
Terraform code for infrastructures on Amazon Web Services, Digital Ocean, OpenStack, Heroku, Packet, and Google Cloud. These images aren't tagged with release versions, but directly with their name: rhel7 for RHEL 7 and rhel6 for RHEL 8. Bump it right now to 0.4.0 so we're starting fresh in apache/metadata.rb: version '0.4.0' Every resource can
notify another resource to do something when its state changes, and any resource can also subscribe to a change of state from another resource. The following example shows how to install an Apache 2.x server on Ubuntu systems: package { 'apache2; ensure => installed; } To deploy a LAMP server on the box web.pomes.pro, we need Apache2,
PHP, and the MariaDB server. Click on Code Downloads & Errata. $ puppet-lint modules/apache/manifests/httpasswd.pp WARNING: variable on line 6 WARNING: variable on line 6 WARNING: variable on line 6 warning only a variable on line 6 warning on
with Terraform We've seen how to declare and use default values in our Terraform code, such as the Ubuntu AMI for our region or our VM size. The uploader already confirmed that they had the permission to publish it. An IAM user for S3 access Let's create a first IAM user for Mary in a new iam.tf file using the aws iam user resource: resource
"aws_iam_user" "mary" { name = "mary" path = "/team/" } 75 Provisioning IaaS with Terraform The path is purely optional and informative, I'm simply suggesting structured paths. Storing updated cookbooks/starter/templates/default/sample.erb in the cache. 0 Style validation This thing is, we solved the obvious problem (a missing variable), but
what about style? Let's choose this one. We'll use Vagrant and simulate a Chef server with the chef_zero provisioner. These things were unit tested previously, but now we're in integration. We'll add some latency monitoring as well with StatusCake, so we have a good foundation of using Terraform on Digital Ocean. While we're at it, let's create the
template for the .htaccess file under our web root folder: $ chef generate template cookbooks/mysite htaccess Its content is the most basic we can find: AuthType Basic AuthName "Restricted Area" AuthUserFile /etc/httpd/htpasswd Require valid-user There's currently no variable in this template. We also need another gem containing helpers: beaker
puppet install helper. As for Chef, Puppet is using resources for installing packages, managing services, creating files, and more. srv-1: Installing pip... 197 Fundamentals of Managing Servers with Chef and Puppet Remember: knife is the command to use from the developer's workstation to manipulate information and resources on the Chef server.
How to do it... We'll quickly manipulate Docker, so we're up and running with some basic usage. You can launch many different relational database servers on RDS, and we'll focus on PostgreSQL for this recipe. The first one to work around SELinux temporarily, and the second one to manage the official MySQL community repository for RHEL. Simply
download the package corresponding to your platform and install it. To create and edit encrypted data, we need eyaml. There're differences in implementation for every custom deployment, but we'll stick with very stable features. As I know, files most often end up being dynamic, I always prefer to start them as templates, even if content is currently
static. Instead of repeating ourselves, let's create a helper script in test/integration/ default/serverspec/spec_helper.rb: require 'spec_helper.rb: require 'spec_helper' Testing a package installation Our cookbooks are doing a lot of
things, and among the most important things is package installation. A database role can include what is now running our database (MariaDB), but maybe tomorrow it can run something else (migrate back to MySQL, or switch to PostgreSQL). We now have our own local registry with infinite storage! See also ff The Docker Registry documentations
configuration/ 367 10 Maintaining Docker Containers In this chapter, we will cover the following recipes: ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development (TDD) with Docker containers with BATS ff Test-Driven Development
Scanning for vulnerabilities with Quay.io and Docker Cloud ff Sending Docker logs to AWS CloudWatch Logs ff Monitoring and getting information out of Docker ff Debugging containers using sysdig Introduction In this chapter, we'll explore some advanced and highly interesting areas that probably most developers today are already used to.
However, this is not very handy, and fortunately we can use external tools such as rnelson0/puppet-generate-puppetfile. This way, changing a virtual Box image won't run on VMware. This cluster is composed of Type 0 machines (4 machines) and fortunately we can use external tools such as rnelson0/puppet-generate-puppetfile. This way, changing a virtual Box image won't run on VMware. This cluster is composed of Type 0 machines (4 machines) and the type of type of the type of type of the type of type
cores and 8 GB RAM), for one manager and 2 nodes, totaling 12 cores and 24 GB of RAM, but we can use more performant machines if we want: the same cluster with Type 2 machines will have 72 cores and 768 GB of RAM (though the price will adapt accordingly). Add some outputs in outputs if you have better information about our app, like the
Heroku app URL and environment variables: output "heroku URL" { value = "${heroku_app.hubot.web_url}" } output "heroku_app.hubot.git_url}" } output "heroku uRL" { value = "${heroku_app.hubot.web_url}" } output "heroku_app.hubot.git_url}" }
simply calling this server "docker": #cloud-config hostname: "docker" the Jane user, so she can log in to the instance to remotely help us. Each test file should be suffixed by
 spec.rb, so let's create spec/ classes/apache_spec.rb with the following content: require 'spec helper' # Description of the "apache" class describe 'apache' do # Assertion list 267 Testing and Writing Better Infrastructure Code with Chef and Puppet it { is expected.to compile.with_all_deps } it { is expected.to contain package('apache2').with({
 ensure' => 'present', }) } it { is expected.to contain service('apache2').with({ 'ensure' => 'trune', }) } end Unit tests are in descriptive blocks, with a list of assertions. What about the role of each of those infrastructure elements? Deploying a Ruby-on-Rails web application in Docker The great thing with Docker is that, as
developers, we can ship whatever is working on this particular container on one environment (such as development or staging) and be sure it will run similarly in another environment (such as production). (This will have to be more precise to handle all the real cases.) In the apache::default recipe, make the following change so the httpd variable is
set to apache2 when running Ubuntu and to the default httpd elsewhere: if node['platform'] == 'ubuntu' httpd e 'apache2' else httpd end package httpd do action :install end service httpd do action :install end service httpd elsewhere: if node['platform'] == 'ubuntu' httpd elsewhere
Foodcritic and Puppet code with puppet-lint Since we're mainly coding in Ruby, we can use common linters such as Rubocop in the Ruby world. It gives access_key = the_secret_key 1. $pass=hiera('root_password'); ... It plans for
changes, but doesn't apply them: $ terraform plan Refreshing Terraform state in-memory prior to plan... As for Chef, the facter information can be accessed from a Puppet manifest. Sections In this book, you will find several headings that appear frequently (Getting ready, How to do it..., How it works..., There's more..., and See also). 326 Chapter 8
Deploying to staging Now let's deploy this new platform cookbook to our staging environment. Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of: ff WinRAR / 7-Zip for Windows ff Zipeg / iZip / UnRarX for Mac ff 7-Zip / PeaZip for Linux The code bundle for the book is also hosted on GitHub at the code bundle for the book is also host at the code bundle for the book is also hosted on GitHub at the code bundle for the book is also hosted on GitHub at the code bundle for the book is also hosted on GitHub at the code bundle for the book is also hosted on GitHub at the code bundle for the book is also hosted on GitHub at the code bundle for the book is also hosted on GitHub at the code bundle for the book is also hosted on GitHub at the code bundle for the book is also hosted on GitHub at the code bundle for the book is also hosted on GitHu
PacktPublishing/Infrastructure-as-Code-IAC-Cookbook. It can even be used to retrieve some specific data related to Windows Management Instrumentation (WMI). 273 Testing and Writing Better Infrastructure Code with Chef and Puppet ----> Destroying ... Simulating dynamic multiple host networking Vagrant is also very useful when used to
simulate multiple hosts in a network. Using Terraform for controlling Docker, we'll be able to dynamically trigger Docker networks, and use Docker volumes. Now we have a valid discovery token, let's add it to our cloud-config.yml file under the etcd2 directive
#cloud-config coreos: etcd2: discovery: "The next step is to configure etcd: ff How should etcd listen for peer traffic? The development of Beaker is very active, and the current version (6.x) needs at least Ruby 2.2.5. In order to use the Embedded Ruby provided by Puppet Collections, let's stay on the 5.x branch: $$ sudo puppet resource package.
beaker-rspec provider=puppet_gem ensure=5.6.0 A C/C++ compiler is needed to install Beaker, so install gcc/g++ or clang before trying to install beaker-rspec. Let's create our second user, with this IAM policy in the iam.tf file: resource "aws_iam_user" "joe" { name = "joe" path = "/team/" } resource "aws_iam_access_key" "joe" { user =
"${aws iam user.joe.name}" } resource "aws iam user policy attachment" "joe ec2ro" { user = "${aws iam user.joe.name}" policy arn = "arn:aws:policy/AmazonEC2ReadOnlyAccess" } Don't forget the useful output that comes with it: output "joe" { value = "ACCESS KEY: ${aws iam access key.joe.id}, SECRET: ${aws iam user.joe.name}" policy arn = "arn:aws:policy/AmazonEC2ReadOnlyAccess" }
iam_access key.joe.secret}" } Next, terraform apply this once again, and can the Joe user see what's on S3? Containers start so fast that the app container may not be ready for NGINX when it's started. As we're already using a dedicated terraform state file, let's do the same for production, and set the environment variable directly through the
command line: $ terraform plan -state=production.tfstate -var environment=production You now have two clearly separated environments using the very same code, but living independently from each other. To achieve this, we'll use two free, popular services: GitHub and the Docker Hub. Though not interactive, it's still powerful enough, given how
 easy it is to install and use. We need to deploy our own Puppet server. Let's continue. The Zlib library is also needed (binaries and headers). It's really no different than an ICMP check; resource "statuscake test" "http" { website name = "www.myweb.com Availability" website url = " " test type = "HTTP" check rate = 300 } 127 4 Automating
Complete Infrastructures with Terraform In this chapter, we will cover the following a complete CoreOS infrastructure on Google Compute Engine ff Provisioning a Complete CoreOS infrastructure on Digital Ocean with Terraform In this chapter, we will cover the following recipes: ff Provisioning a Complete CoreOS infrastructure on Digital Ocean with Terraform In this chapter, we will cover the following recipes: ff Provisioning a Complete CoreOS infrastructure on Digital Ocean with Terraform In this chapter, we will cover the following recipes: ff Provisioning a Complete CoreOS infrastructure on Digital Ocean with Terraform In this chapter, we will cover the following recipes: ff Provisioning a Complete CoreOS infrastructure on Digital Ocean with Terraform In this chapter, we will cover the following recipes: ff Provisioning a Complete CoreOS infrastructure on Digital Ocean with Terraform In this chapter, we will cover the following recipes: ff Provisioning a Complete CoreOS infrastructure on Digital Ocean with Terraform In this chapter, we will cover the following recipes: ff Provisioning a Complete CoreOS infrastructure on Digital Ocean with Terraform In this chapter, we will cover the following recipes: ff Provisioning a Complete CoreOS infrastructure on Digital Ocean with Terraform In this chapter, we will cover the following recipes: ff Provisioning a Complete CoreOS infrastructure on Digital Ocean with Terraform In this chapter, we will cover the following recipes: ff Provisioning a CoreOS infrastructure on Digital Ocean with Terraform In this chapter, we will cover the following recipes: ff Provisioning a CoreOS infrastructure on Digital Ocean with Terraform In this chapter, we will cover the following recipes: ff Provisioning a CoreOS infrastructure on Digital Ocean with Terraform In this chapter, we will cover the first the firs
Terraform ff Creating a scalable Docker Swarm cluster on bare metal with Packet Introduction In this chapter, we'll describe complete infrastructures using Terraform, how it looks when everything is tied together, with a real project in mind. All Chef code needs to be sent over in some way to be applied manually. That would look like this in a file
named s3.tf: resource "aws s3 bucket" | bucket = "iacbook" tags { Name = "IAC Book Bucket" } We're not sure about to happen, so let's plan with Terraform, so it's telling us exactly what it's intending to do: $ terraform plan Refreshing Terraform state in-memory prior to plan... This is how to test it's in place with the default
attributes: it 'creates a virtualhost configuration file' do expect(chef_run).to create_template('/etc/httpd/conf.d/default. Here's how to do it. 25 Vagrant Development Environments Getting ready To step through this recipe, you will need the following: ff A working VirtualBox installation ff An Internet connection How to
do it... Follow these steps for simulating a networked three-tier architecture app with Vagrant. As we don't have any power on why is that, here's a workaround to expect a very specific error from a Chef run, and this will be helpful many times later: it 'converges successfully' do # The selinux cookbook raises this error. We're probably using many
cookbooks that are doing a lot of things, each unit tested in an early stage, but how are they playing together for real? Neither the authors, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book. Refer to the dedicated chapter of this book for more
information on obtaining this cookbook: #cloud-config chef: run_list: - "recipe[starter]" 186 Chapter 5 In the logs, we'll see this being applied like this: [...] Loading cookbooks [] Storing updated cookbooks/starter/attributes/default.rb in the cache. Let's create generic variables for that in variables the cookbooks [] Storing updated cookbooks [[emai
= "" description = "Heroku account API key" } Don't forget to override them in terraform.tfvars: heroku account API key" } Don't forget to override them in terraform.tfvars: heroku account API key" } Don't forget to override them in terraform.tfvars: heroku account API key" }
information we have: provider "heroku" { email = "${var.heroku_email}" api_key = "${var.heroku
and writing tests for it, there's no reason not to see those tests executed in CI. [...] Now, at home or at the coworking space, you can simply connect to your work Vagrant box (if needed, the default Vagrant box (if needed, the default Vagrant box (if needed, the default Vagrant box). If there is a topic that you have
expertise in and you are interested in either writing or contributing to a book, see our author guide at www.packtpub.com/authors. Now you can start coding in the ./src/ folder. 46 2 Provisioning IaaS with Terraform In this chapter, we will cover the following recipes: ff Configuring the Terraform AWS provider ff Creating and using an SSH key pair to
use on AWS ff Using AWS security groups with Terraform ff Creating an Ubuntu EC2 instance with Terraform ff Creating an Ubuntu EC2 instance with Terraform ff Creating private Docker repositories with Terraform ff Creating a PostgreSQL RDS database with
Terraform ff Enabling CloudWatch Logs for Docker with Terraform ff Managing IAM users with Terraform ff Managing IAM users
mail, monitoring, and others). (ID: admin key) aws security group: Refreshing state... Apache configuration needs to be reloaded when the default vhost is removed, so we need to modify modules/apache/manifests/init.pp with the corresponding notify attribute: class apache { package { 'apache2': ensure => present, } service
 {'apache2': ensure => running, enable => true } file {'/etc/apache2/sites-enabled/000-default.conf': ensure => absent, notify => Service['apache2'], } } 228 Chapter 6 The same logic applies for the virtual host creation (modules/apache/manifests/vhost. Replace with the host you want if you're not running against a local Docker host: resource
 "docker_container" "ghost" { name = "ghost" image = "${docker_{image.ghost.latest}}" env = ["VIRTUAL HOST=localhost"] } 96 Chapter 3 Now let's start the nginx-proxy container. This resource takes all the parameters we previously declared (name, image, flavor, SSH key, and security groups). Being able to fully execute a whole pile of tests
automatically, upon any code check-in, is a major quality improvement step. Any group can have a sublist of users, taking a list of users. Depending on your sudo configuration, Vagrant may ask you for your password. The Puppet agent
has been explicitly stopped; during examples, we will start it on demand to see all changes. It works wonders. Reader feedback is important for us as it helps us develop titles that you will really get the most out of. [...] aws security group. base security group. Examples, we will start it on demand to see all changes. It works wonders. Reader feedback is important for us as it helps us develop titles that you will really get the most out of. [...] aws security group. Base security group. The feedback is important for us as it helps us develop titles that you will really get the most out of. [...] aws security group. Base securit
code—we would be grateful if you could report this to us. html ff The Chef documentation on the directory resource: resource_directory.html ff The Chef documentation on the file resource documentation on the template resource; resource_template.html ff The Puppet file resource documentation on the file resource.
puppet/4.8/types/file.html ff Using templates with Puppet: templates with Puppet: templates with Puppet: templates with another. The MySQL instance will allow connections only from the two web servers (with valid credentials), and all three instances (SQL and HTTP) will be accessible from
a single corporate network (our company's network). There's more... Using Puppet and Beaker, let's try to write acceptance tests for our Apache module. Is there anything new in our repository? 297 Maintaining Systems Using Chef
encrypted data bags and Hieraeyaml with Puppet Some information in data bags can be safely stored in the changes will help you better understand the changes in the output. Finally, create the main manifest manifests/site.pp, with the
following content: node 'web.pomes.pro' { class { 'apache':; 'php':; 'mariadb':; } } That's it! With a few lines of code, all necessary binaries will be installed. As for Chef, provided packages are shipped with an Embedded Ruby version. -e none Now we can docker build, tag, and push images at will! (See more about using Docker images in the
dedicated chapter of this book.) A nice advanced feature is the ability to use fine-grained policies for each repository created. Within minutes we have new hardware ready and connected to the network. Once again, deploying Node.js applications is out of the scope of this book, so I kept it as simple as possible—simple directories and permissions
creation and systemd unit deployment. Using a Vagrant development environment can help keep your work machine clean and allow you to use your usual tools and editors while using an infrastructure close to production. Let's write this in a keys.tf file: resource "digitalocean ssh key" "default" { name = "Digital Ocean SSH Key" public key = 100 file: resource "digitalocean ssh key" "default" { name = "Digital Ocean SSH Key" public key = 100 file: resource "digitalocean ssh key" "default" { name = "Digital Ocean SSH Key" public key = 100 file: resource "digitalocean ssh key" "default" { name = "Digital Ocean SSH Key" public key = 100 file: resource "digitalocean ssh key" "default" { name = "Digital Ocean SSH Key" public key = 100 file: resource "digitalocean ssh key" "default" { name = "Digital Ocean SSH Key" public key = 100 file: resource "digitalocean ssh key" "default" { name = "Digital Ocean SSH Key" public key = 100 file: resource "digitalocean ssh key" "default" { name = "Digital Ocean SSH Key" public key = 100 file: resource "digitalocean ssh key" "default" { name = "Digital Ocean SSH Key" public key = 100 file: resource "digitalocean ssh key" | default" { name = "Digital Ocean SSH Key" public key = 100 file: resource "digitalocean ssh key" | default" { name = "Digital Ocean SSH Key" public key = 100 file: resource "digitalocean ssh key" | default" { name = 100 file: resource | digitalocean ssh key = 1
"${file("${var.ssh key file}.pub")}" } Create the related variable in variables.tf, with our suggested default: variable "ssh key file" } It's now time to effectively override the value in the terraform.tfvars file if you feel like it: ssh key file = "./keys/my own key" Creating the
CoreOS cluster members Here's the core of our infrastructure: three nodes running in the New York City data center NYC1, with private networking enabled, no backups activated (set it to true if you feel like it!), the SSH key we previously created, and a cloud-init file to initiate configuration. Foodcritic Foodcritic goes much further than Cookstyle
and checks the Chef code for things such as incompatible, nonidempotent, repetitive, or deprecated code, and missing templates, files, dependencies, or variables. Then we'll configure the testing environment Test Kitchen, which leverages the use of VMs through Vagrant (or other systems) to apply test suites. Start Vagrant with the Vagrantfile from
the previous recipe, aws.amazon.com/cloudwatch/). Getting ready To step through this recipe, you will need: ff A working Chef DK installation on the workstation ff The Chef code (optionally) from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, Chapter 7, Testing and Writing
Better Infrastructure Code with Chef and Puppet, or any custom Chef code 329 Maintaining Systems Using Chef and Puppet How to do it... There is no single way to achieve our goal. pkcs7.pem. FC033 (. org. In the newly-created mysite/recipes/mysql.rb file, start by including the new recipes described as needed by the documentation:
 include recipe "selinux::disabled" include recipe "yum-mysql-community::mysql57" 9. Docker tags are not to be mistaken with Docker labels. We'll see how to use both Chef cookbooks with Vagrant, as well as how to simulate Chef version upgrades between environments. 42 Chapter 1 srv-1: Installing Ansible... This is usually where all the logic is
the previous recipe How to do it... Follow these steps to enable VirtualBox Guest Additions in Vagrant: 1. It's highly recommended to lock the Terraform version, and updates can be painlessly handled. We'll also be able to easily rollback to a previously running version. In other words, the test
 /integration/default/serverspec/virtualhost spec.rb file will match the Chef cookbook recipe named virtualhost, executed from the default Kitchen test suite, and tested with the serverspec plugin. The documentation lists the requirements: an API token and organization name: provider "github" { token = "${var.github token}" organization =
"${var.github_organization}" } Set the generic variables in a variables in a variable "github_token" { default = "ACME Inc." description = "GitHub Organization Name" } Don't forget to override those variables to fit your own in the terraform.tfvars file. For
example, we know we want the etcd2 and fleet services to start at every boot: coreos: units: - name: etcd2.service command: start - name: fleet.service command
modules/apache/manifests/htpasswd.pp: define apache::htpasswd.pp: define apache::htpasswd.erb'), } } 234 Chapter 6 For the corresponding template, this time, let's try an ERB template in modules/apache/htpasswd.erb'), } }
templates/htpasswd.erb: : From the main manifest, we can now create the password file: apache::htpasswd'; filepath => '/etc/apache2/htpasswd', users => hiera('webusers'), } We also need to create a .htaccess file. Deploying the Chef omnibus installer using cloud-init As we want to use the official omnibus build (other choices are
installing Chef through a Ruby gem—deprecated and too dependent on a locally installed even if, for some reason, the Chef client was found to be already present on the system. The creation of a new virtual host
must be done in /etc/apache2/sites-available, and will be generated from a template. Here is a session example: $ eyaml encrypt -s 'super secure password' [hiera-eyaml-core] Loaded config from /Users/me/.eyaml/config.yaml string: ENC[PKCS7,MIIBiQY]KoZIhvcNAQcDoIIBejCCAXYCAQAxggEhMIIBHQIBADAFMA
A CAQEwDQYJKoZIhvcNAQEBBQAEggEALiJ2a9uZ04lk2V5xKqEd0n3BtA4OLe1B6rA2iVru\ JRKxWJdevuGvJ55DDedRwBMZmqbvSMO1cgMUyPbfEy54i3SXw4x3LEuxc1R31ILoOspBgz\ U4OLuepCotuhBASA/pI/xu40y66AZAcCQ4CtD9SZJYjiWNtUA91rcARy/xYQGK39Qievx\ T2eq5De89qIn2w/5fIRIkJBRyNqnwyYCWKcKSRwaiLbimpwmarOP+dxGHEFRrD/
 ENC[PKCS7,MIIBiQY]KoZIhvcNAQcDoIIBejCCAXYCAQAxggEhMIIBHQIBADAFMAACAQEw\ DQY]KoZIhvcNAQEBBQAEggEALj]2 a 9uZ04lk2V5xKqEd0n3BtA4OLe1B6rA2\ iVruJRKxWJdevuGvJ55DDedRwBMZmqbvSMO1cgMUyPbfEy54i3SXw4x3LEux\ c1R31ILoOspBgzU4OLuepCotuhBASA/pI/xu40y66AZAcCQ4CtD9SZJYjiWNACAQEWBMZmqbvSMO1cgMUyPbfEy54i3SXw4x3LEux\ c1R31ILoOspBgzU4OLuepCotuhBASA/pI/xu40y66AZAcCQ4CtD9SZJYjiWNACAQEWBMZmqbySMO1cgMUyPbfEy54i3SXw4x3LEux\ c1R31ILoOspBgzU4OLuepCotuhBASA/pI/xu40y66AZAcCQ4CtD9SZJYjiWNACAQEWBMZmqbySMO1cgMUyPbfEy54i3SXw4x3LEux\ c1R31ILoOspBgzU4OLuepCotuhBASA/pI/xu40y66AZAcCQ4CtD9SZJYjiWNACAQEWBMZmqbySMO1cgMUyPbfEy54i3SXw4x3LEux\ c1R31ILoOspBgzU4OLuepCotuhBASA/pI/xu40y66AZAcCQ4CtD9SZJYjiWNACAQEWBMZmqbySMO1cgMUyPbfEy54i3SXw4x3LEux\ c1R31ILoOspBgzU4OLuepCotuhBASA/pI/xu40y66AZAcCQ4CtD9SZJYjiWNACAQEWBMZmqbySMO1cgMUyPbfEy54i3SXw4x3LEux\ c1R31ILoOspBgzU4OLuepCotuhBASA/pI/xu40y6AZAcCQ4CtD9SZJYjiWNACAQEWBMZmqbySMO1cgMUyPbfEy54i3SXw4x3LEux\ c1R31ILoOspBgzU4OLuepCotuhBASA/pI/xu40y6AZAcCQ4CtD9SZJYjiWNACAQEWBMZmqbySMO1cgMUyPbfEy54i3SXw4x3LEux\ c1R31ILoOspBgzU4OLuepCotuhBASA/pI/xu40y6AZAcCQ4CtD9SZJYjiWNACAQEWBMZmqbySMO1cgMUyPbfEy54i3SXw4x3L
 tUA91rcARy/xYQGK39QievxT2eq5De89qIn2w/5fIRIkJBRyNqnwyYCWKcKS~RwaiLbimpwmarOP+dxGHEFRrD/FiM4NfoV1WNNVr1UkPEFuNrWBzwBpvyZUn~MbGHN676Rg5vq9sS6aWI6zPxTrJyLtssZm1f4GsfhmE+anFmuxrcWtEH6C82~wKMOoTBMBgkqhkiG9w0BBwEwHQYJYIZIAWUDBAEqBBC3MhSP09yUw8XTj0Xd
DQYJKoZIhvcNAQEBBQAEggEALjJ2a9uZ04lk2V5xKqEd0n3BtA4OLe1B6rA2 iVruJRKxWJdevuGvJ55DDedRwBMZmqbvSMO1cgMUyPbfEy54i3SXw4x3LEux c1R31ILoOspBgzU4OLuepCotuhBASA/pI/xu40y66AZAcCQ4CtD9SZJYjiWN tUA91rcARy/xYQGK39QievxT2eq5De89qIn2w/5fIRIkJBRyNqnwyYCWKcKS
 RwaiLb impw mar OP+dx GHEFRr D/FiM4N foV1WNNVr1UkPEFuNrWBzwBpvyZUn~MbGHN676Rg5vq9sS6aWI6zPxTrJyLtssZm1f4GsfhmE+anFmuxrcWtEH6C82~wKMOoTBMBgkqhkiG9w0BBwEwHQYJYIZIAWUDBAEqBBC3MhSP09yUw8XTj0Xd~lG1VgCCDCGhqIFdUmORYKlq0Pn5CE/cDZKTO+bhHxdBw5amAGQ==]~However,~eyaml~has~a~very~handy
 feature—the edit mode—allowing us to create and edit encrypted values in plain text, based on the keys stored in the $HOME directory: $ eyaml edit hieradata/nodes/web.pomes.pro.eyaml This command will launch an editor, and we just need to enter the following content: --root_password: > DEC::PKCS7[super_secure_password]! 305 Maintaining
Systems Using Chef and Puppet While saving, eyaml will write the file with the encrypted content for root password. These propositions are very often good advice, though. That's the moment a configuration management tool such as Chef, Ansible, or Puppet enters the game, to provision the virtual machine. See also ff The auto documented Makefile
 auto-documented-makefile.html ff Rbenv: ff RVM: ff DVM: ff DVM
later add a firewall, whose rule will apply to a target defined by its tags. However, if you're in a hurry, here's a one liner that will do the job of installing Sysdig on your Linux host. We might have chosen to not have publicly exposed servers and use a bastion host instead. The production.rb file would contain the following for the production
environment: cookbook_versions 'mysite' => '= 0.3.1' The development.rb file would contain the following for the development environment file may contain many cookbook constraints, as follows: cookbook_versions: { 'mysite': '= 0.4.0', 'apache': '= 0.6.0' } Overriding attributes for an
environment Each environment can override any value, and, in Chef, it's the highest level of override error (RuntimeError, 'chefspec not supported!') end We've seen a selection of the most common and reusable unit tests for Chef cookbooks! There's more... Using Puppet Labs is providing a repository containing
FabQkFgfpHwAfa0sCb8ad/v8pTQqVGfZQv1GptKk mary = ACCESS_KEY: AKIAJPQB7HBK2KLAARRQ, SECRET: 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 78 Chapter 2 There's more... If you'd prefer to see how this would work using Ansible, it's a bit different. Add the following to the docker.rb recipe: docker_installation 'default' do repo
 'main' action :create end Execute kitchen again to apply our code and see whether the tests are passing or failing: $ kitchen converge $ kitchen verify [...] Package "docker-engine" should be installed [...] Finished in 0.18797 seconds (files took 0.43908 seconds to load) 5 examples, 4 failures 323 Maintaining Systems Using Chef and Puppet Good!
What was failing a few minutes ago is now passing. Let's use the image we just declared through the docker_image resource, and export an environment variable named VIRTUAL_HOST, to be used by the nginx-proxy container (refer to the nginx-proxy documentation). Let's create a spec/spec_helper_acceptance.rb file with the
following content: require 'beaker-rspec' req
module puppet module install(:source => proj root, :module name => 'apache') end end This helper file will be used to install Puppet on the test box, and populate the module directory with our apache module. Finally, let's write the mysite role, that will include a webserver, a database, as well as its own cookbook, in roles/mysite.rb: name "mysite role, that will include a webserver, a database, as well as its own cookbook, in roles/mysite.rb: name "mysite role, that will include a webserver, a database, as well as its own cookbook, in roles/mysite.rb: name "mysite role, that will include a webserver, a database, as well as its own cookbook, in roles/mysite.rb: name "mysite role, that will include a webserver, a database, as well as its own cookbook, in roles/mysite.rb: name "mysite role, that will include a webserver, a database, as well as its own cookbook, in roles/mysite.rb: name "mysite role, that will be used to install Puppet on the test box, and populate the module directory with our apache module.
description "MySite role" run_list("role[webserver]", "recipe[mysite]") 4. Keeping the tfstate isolated It's strongly recommended to not mix Terraform state files between environments. Thank you for installing Chef! At this point, you'll have a valid Chef installation under /opt/chef, though not yet configured. If anything else, agent
will be disabled. Though perfectly correct, many images are sub-optimized when we're talking about size. This way, it's much easier to debug, compare logs or behavior when failures arise, without losing time doing the math of the time zones. Based on our previous Vagrant-based LAMP setup, let's create a local module with a single file, namely
module/ baseconfig/manifests/init.pp, with: # @param agentmode Agent type: service or cron. We'll create unit tests for the cookbooks created in Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, that covers the most common tests, such as convergence issues, packages installation, services status check, file and template
creation, access rights, recipe inclusion, stubbing data bag searches, or even intercepting expected errors. Just change the network configuration from the preceding Vagrantfile to launch the VMs in this specific network configuration from the preceding Vagrantfile to launch the VMs in this specific network.
with Vagrant Vagrant is a great tool to help simulate systems in isolated networks, allowing us to easily mock architectures found in production. Getting started (notions and tools) Chef is a very complex system, with a lot of notions and vocabulary that can be very discouraging at first. If not, stop there and return a helpful exit message on how to
install the required plugins. For the Chef client to be able to apply Chef code, it first needs to be configured and registered on the Chef server. } If there is any future change between service versus cron, our baseconfig module will remove the configuration for the previous mode. It really shouldn't go farther than delivering a system in a slightly more
advanced and expected state for the next configuration system to take over. This is particularly useful when a lot of nodes are connected to the same Puppet server in order to spread the Puppet agent's requests in time. Let's call this cookbook common: $ cd chef-repo/cookbooks $ chef generate cookbook common $ cd com
client, there's an official cookbook aptly named chef-client. In a mysql_spec.rb file, add the following: describe service('mysql-default') do it { should be_running } end Testing for listening ports. This way, we'll be certain that the code we're writing actually does what it's
expected to do in the (simulated) real world! We strongly suggest that you add those integrations tests to an automated CI system. $ puppet-lint modules/apache/manifests/htaccess.pp WARNING: defined type not documented on line 2 We can see two error
categories: ff Puppet coding style warnings/errors ff Missing documentation Let's try to fix them! Puppet coding style For our concerns here, the basic rules are: ff Tabulation needs to be two-space characters ff No trailing whitespaces ff In string interpolation, variables should be enclosed in braces; for example, "$docroot/.htaccess" is wrong and
must be "${docroot}/.htaccess" Documentation Documentation Should be done using Markdown. Note the Ruby #{node["sitename"]}" do owner 'root' group 'root' mode '0755' action :create end Let's reuse the file
resource to create a basic index.html file with a simple string such as Hello from Chef! or whatever you find more appealing, in the apache/recipes/ virtualhost.rb file: file "/var/www/# {node["sitename"]}/index.html" do owner 'root' group 'root' mode '0644' content 'Hello from Chef! or whatever you find more appealing, in the apache/recipes/ virtualhost.rb file: file "/var/www/# {node["sitename"]}/index.html" do owner 'root' group 'root' mode '0644' content 'Hello from Chef! or whatever you find more appealing, in the apache/recipes/ virtualhost.rb file: file "/var/www/# {node["sitename"]}/index.html" do owner 'root' group 'root' mode '0644' content 'Hello from Chef! or whatever you find more appealing, in the apache/recipes/ virtualhost.rb file: file "/var/www/# {node["sitename"]}/index.html" do owner 'root' group 'root' mode '0644' content 'Hello from Chef! or whatever you find more appealing, in the apache/recipes/ virtualhost.rb file: file "/var/www/# {node["sitename"]}/index.html" do owner 'root' group 'root' g
for our Apache virtual host configuration file: $ chef generate template cookbooks/apache virtualhost.erb. Fill in the details, use a valid e-mail address, and validate. 94 Chapter 3 Getting ready To step through this recipe, you will need the following: ff A working Terraform
installation. 233 Fundamentals of Managing Servers with Chef and Puppet There's more... With Puppet, we can use variables compatible with most IaaS providers—$private ipv4 and $public ipv4. The idea here is to create a new cookbook dedicated to an imaginary
MySite application, that will reference and depend on all the other recipes, so we can only load this MySite cookbook and be done with it. However, there's a much more powerful way to leverage the full capabilities of Terraform. 14 Chapter 1 Create a simple src/index.html file, containing some text: $ mkdir src; echo "Hello from Docker via Vagrant"
> src/index. html ff The Chef cookbook metadata.rb documentation: config_rb_metadata.html More dynamic code using notifications Wouldn't it be great if Chef knew how and what to restart automatically when a change arises? That's why we can use a common Makefile for each project using Terraform. erb inflating: chef-repo/cookbooks/chefignore
inflating: chef-repo/.gitignore inflating: chef-repo/.chef/knife.rb inflating: chef-repo/.chef/knife.r
installed. Let's change our Apache manifest to do this. REGISTRY STORAGE S3 REGISTRY STORAGE S4 REGISTRY STORAGE S4 REGISTRY S
you tried it, and start this updated one: $ docker-compose up Now tag again an image locally: $ docker tag ubuntu:16.04 localhost:5000/ubuntu Then, push the image to the local registry to sync the layers we push with the AWS S3
backend. Now we can use the statuscake test resource to activate immediate latency (ping) monitoring on every droplet.coreos.* resource "statuscake test" "coreos_cluster" { count = "${element(digitalocean_droplet.coreos.* name, and of the count is considered." | count = "${element(digitalocean_droplet.coreos.* name, and of the count is considered." | count = "${element(digitalocean_droplet.coreos.* name, and of the count is considered." | count = "${element(digitalocean_droplet.coreos.* name, and of the count is considered." | count = "${element(digitalocean_droplet.coreos.* name, and of the count is considered." | count = "${element(digitalocean_droplet.coreos.* name, and of the count is considered." | count = "${element(digitalocean_droplet.coreos.* name, and of the count is considered." | count = "${element(digitalocean_droplet.coreos.* name, and of the count is considered." | count = "${element(digitalocean_droplet.coreos.* name, and of the count is considered." | count = "${element(digitalocean_droplet.coreos.* name, and of the count is considered." | count = "${element(digitalocean_droplet.coreos.* name, and of the count is considered." | count = "${element(digitalocean_droplet.coreos.* name, and of the count is considered." | count = "${element(digitalocean_droplet.coreos.* name, and of the count is considered." | count = "${element(digitalocean_droplet.coreos.* name, and of the count is considered." | count = "${element(digitalocean_droplet.coreos.* name, and of the count is considered." | count = "${element(digitalocean_droplet.coreos.* name, and of the count is considered." | count = "${element(digitalocean_droplet.coreos.* name, and of the count is considered." | count = "${element(digitalocean_droplet.coreos.* name, and of the count is considered." | count = "${element(digitalocean_droplet.coreos.* name, and of the count is considered." | count = "${element(digitalocean_droplet.coreos.* name, and of the count is considered." | count = "${element(digitalocean_droplet.coreos.* name, and of the count i
count.index)}.${var.cluster_domainname}" website_url = "${element(digitalocean_droplet.coreos.*.ipv4_address, count.index)}" test_type = "PING" check_rate = 300 paused = false } 135 Automating Complete Infrastructures with Terraform apply this: $ terraform apply this: $
./keys/iac_admin_sshkey, ssh -i ./keys/iac_admin_sshkey] Confirm that we can connect to a member
from cluster is healthy Check that all fleet members are all right: ~ $ fleetctl list-machines MACHINE IP METADATA 24762c02... This is a standard ERB (short for Embedded Ruby) template. Cookstyle Navigate to a cookbook root directory and type in the following: $ cookstyle 250 Chapter 7 This will output all the suggestions for a
cleaner code. This is done by using the $environment variable, which is set by the Puppet server. So no bad surprise can happen with an untested update (updating the package will need a rebuild of the image), we're cleaning the apt cache more precisely, and we're explicitly exposing a port from the container. Please refer to the Creating a free
hosted server Chef account and a Puppet server recipe of Chapter 6, Fundamentals of Managing Servers with Chef and Puppet), with the default cookbook uploaded An Internet connection How to do it... As there're a lot of sources involved, let's put all the required information in a table (the Chef information is taken from the Chef Starter Kit, or your
own Chef server, fill in your own values): Hostname centos-1 Instance type t2.micro AMI in eu-west-1 ami-7abd0209 AMI in us-east-1 ami-6d1c2007 SSH username SSH key centos keys/aws_terraform TCP ports needed 22 Cookbook(s) to apply Chef server URL starter Validation key iacbook.pem Validation client name iacbook Chef client version
12.13.37 1. In a previous example, we added a new virtual host to our node, and we had to manually restart Apache to take the change into account. Each subsequent release will be identified by its SHA indicating its unique identification and symlinked to current at the end of the deployment process. In our case, we'd like our template resource to
notify the httpd system service to restart when the Virtual Host template changes, so we're sure the change is automatically taken into account. Have a look: $ git status terraform.tfstate $ git commit -m "initial terraform sure, now we have to ship our infrastructure state to the pull request: $ git add terraform.tfstate $ git commit -m "initial terraform sure, now we have to ship our infrastructure state to the pull request: $ git add terraform.tfstate $ git commit -m "initial terraform sure, now we have to ship our infrastructure state to the pull request: $ git add terraform.tfstate $ git commit -m "initial terraform sure, now we have to ship our infrastructure state to the pull request: $ git add terraform.tfstate $ git commit -m "initial terraform sure, now we have to ship our infrastructure state to the pull request: $ git add terraform.tfstate $ git commit -m "initial terraform sure, now we have to ship our infrastructure state to the pull request: $ git add terraform.tfstate $ git states $ git s
terraform state" $ git push Our coworker sees that everything is all right, and she also checked the server is doing well. What you need for this book The essential requirement is a computer capable of running a Linux virtual machine and an Internet connection. We've seen with all the previous recipes how often we need to use the AWS Access Keys.
and it's surely not a good idea to use a single key for all your activities. 123 Going Further with Terraform Adding users to the GitHub organization We want to add the username john as a member, and jane as an admin, in a file you can name github.tf (feel free to split managed GitHub features in many smaller files as your organization grows): // john
is a simple member of the organization resource "github membership" "membership" "m
invitations by e-mail). This looks exactly like our objective. This includes the etcd distributed key value store, the fleet network, the update strategy, and some systemd unit configuration. We'll need our infrastructure code to stay consistent without our intervention. We have not discussed a lot about Hiera yet, and it's time to do so
Getting ready To step through this recipe, you will need a working Docker installation. Execute the following to tag the ubuntu: 16.04 localhost: 5000/ubuntu Then, to push the image to the local registry, execute this: $ docker push localhost: 5000/ubuntu This Docker image is now stored
locally and can be reused without accessing the public network nor the Docker Hub or similar services. (4m50.01s) What happened here is the following: ff Test Kitchen installed Chef, synchronized the cookbooks, solved dependencies with
Berkshelf, and applied the run list content ff Test Kitchen launched tests (we don't have any for now) ff Test Kitchen destroyed the VM as everything went smoothly How it works... When we execute the simple kitchen destroyed the VM as everything went smoothly How it works... When we execute the simple kitchen destroyed the VM as everything went smoothly How it works... When we execute the simple kitchen destroyed the VM as everything went smoothly How it works...
of the change on GitHub: A few seconds later, our newest build is available for everyone to use: $ docker pull sjourdan/nginx-docker-demo Building Docker Hub. We'll deploy a Docker Swarm cluster on a bare metal CoreOS cluster for
containers, an n-tier web infrastructure, or a GitLab + CI combo. lock to the destination folder, /usr/src/app. All the rules are described on the Foodcritic website at , along with examples and explanations. We'd like our platform to work on both Ubuntu 16.04 and CentOS 7.2, and we want the entry point for our cookbook as its default recipe: --driver:
name: vagrant provisioner: name: chef_zero always_update_cookbooks: true platforms: - name: centos-7.2 suites: - name: default run_list: - recipe[platform::default] attributes: 321 Maintaining Systems Using Chef and Puppet As we've seen earlier in this book, create a helper script in the serverspec folder named spec_helper.rb: require 'serverspec' #
Required by serverspec set :backend, :exec Let's start our test and see what is it we want to do according to our needs: ff We want the docker service to be enabled and started ff We want to do according to our needs: ff We want the docker service to be enabled and started ff We want to do according to our needs: ff We want the docker service to be enabled and started ff We want to do according to our needs: ff We want to do according to our needs: ff We want the docker service to be enabled and started ff We want to do according to our needs: ff We want to our needs: ff We want to our needs: ff We want
docker_spec.rb, in the serverspec folder: require 'spec_helper' describe package('docker images') do it { should be_installed } end describe command('docker images') do it { should be_installed } end describe service('docker') do it { should be_installed } end describe command('docker images') do it { should be_installed } end describe service('docker') do it { should be_installed } end describe
end This is good enough for our needs! Let's launch our test environment by firing up kitchen converge $ kit
HARDDISK", size => "40.00 GiB", size_bytes => 42949672960, vendor => "Innotek GmbH", version => "VirtualBox" }, ... Including dependencies A WordPress installation needs at least an HTTP
server and a database. To do this, we use the knife command on our workstation: $ knife cookbook uploaded 1 cookbook. These techniques are fortunately brought to the infrastructure world through the tools we use; now as infrastructure is basically code, it can be analyzed, tested, and reported! Combined
with CI systems, writing infrastructure code that is thoroughly tested at different levels helps hugely to achieve a very high quality of sustainable code and prevents unexpected regressions that would have otherwise broken things later. The concept is elegant and simple: shared key encryption is done for each and every existing Chef node through
their already existing client keys. html Add the Docker volume configuration to our Docker provider block in Vagrant: config.vm.provider docker.image = docker.image = docker.ports = dock
hardcode paths, making it highly distributable. It can also manage environments (such as staging, development, production), but we will focus on package management in this chapter. It's supported by all recent distributions (Ubuntu, Arch, CentOS/Red Hat, Fedora, and more), as well as a variant found on CoreOS systems. Executing Docker Compose
will automatically create a Docker network and run the containers: $ docker-compose up [...] mysql_1 | 2016-12-01 20:51:14 139820361766848 [Note] mysqld (mysqld 10.1.19-MariaDB-1~jessie) starting as process 1 ... Ouch! Changing the instance type seems to be a destructive action. ==> default: Destroying VM and associated drives... Here's a
working Tomcat systemd service that is running Docker and sending the logs to a dynamically allocated stream name in /etc/systemd/system/ tomcat.service [Unit] Description=Tomcat Container Service (Service) TimeoutStartSec=0 RestartPre=/usr/bin/docker pull tomcat.service (Service) TimeoutStartSec=0 RestartPre=/usr/bin/d
 %n ExecStartPre=-/usr/bin/docker rm %n ExecStart=/usr/bin/docker run --rm -p 80:8080 --log-opt awslogs-region=us-east-1 -
sudo systemctl daemon-reload $ sudo systemctl start tomcat Now a third log stream is created with the service name, with the systemd unit logs streaming into it: 398 Chapter 10 Enjoy a centralized and powerful way of storing and accessing logs before you eventually process them! There's more... For the sake of the exercise, we'll also mount the
new device under /srv/www on the host. Also, the authorized network block doesn't currently support a count value, so we can't iterate dynamically over every HTTP host. Creating compute instances on OpenStack It's now time to create the instances. Chef being written in Ruby, a lot of those tools are also written in Ruby and over the years, the
 usual dependency hell between tools. plugins, code, and various Ruby versions led to a simple solution—the Chef DK, 102 Chapter 3 Teamwork - sharing Terratorm to manage your intrastructure, you'll face an issue; now does your team work together on
infrastructure-as-code? ==> srv-1: -- Type: t2.micro ==> srv-1: -- Type: t2.micro ==> srv-1: -- AMI: ami-c06b1eb3 ==> srv-1: -- Region: eu-west-1 [...] ==> srv-1: Waiting for SSH to become available... Let's set it as variable "ssh_username" { default = "ubuntu" description = "SSH username" } 150 Chapter 4 Now we can just type in all the commands to be
executed when the instance is ready: provisioner "remote-exec" { connection { user = "${file("${var.ssh key file}")}" } inline = ["sudo mkfs.ext4 /dev/vdb", "sudo apt update -y", "sudo apt update -
upgrade -y", "sudo apt install -y docker.io", "sudo systemctl enable docker", "sudo systemctl enable docker", "sudo systemctl start docker", "sudo docker run -d -p 80:80 --name gitlab/gitlabce:latest",] } Add a simple output in the outputs.tf file, so we easily know the GitLab instance public IP: output "GitLab Instance" { value = "gitlab:
http://${openstack_compute_instance_v2.gitlab.access_ip_v4}"} The runner instances are the same, but a little simpler, as they don't need a local volume. As it's now an important part of the mysql spec.rb file: describe yumrepo('mysql57-
community') do it { should be exist } it { sh
"php" Now, our MySite cookbook has a nice dependency graph: to fully work, it needs Apache, MariaDB, and PHP. We'll ship a sample HTML index file as well, to validate we're running the correct virtual host. Let's install it using the following: $ sudo puppet resource package hiera-eyaml provider=puppet gem 2. From specific guest operating
systems to remote IaaS providers, features around sharing, caching or snapshotting, networking, testing or specifics to Chef/Puppet, a lot can be done through plugins in Vagrant. Many providers are based on it, and you can roll your own in your data center. It only requires to give the Docker daemon access to the AWS ACCESS KEY ID and
AWS SECRET ACCESS KEY environment variables (configuring the Docker daemon is out of the scope of this chapter, but that's under /etc/sysconfig/docker for Debian/Ubuntu systems). [...] aws instance.coreos: Creating... On Digital Ocean, we'll build a fully working and
monitored CoreOS cluster with DNS dynamically updated. Creating an automated build on the Docker Hub is one of the commercial services from the commercial servic
8000 ENTRYPOINT ["/hello"] This service is listening on port 8000, and any other Docker container running on the host can access it, by default on the same network: # curl : (7) Failed to connect to localhost port 8000: Connection refused 351 Working with
Docker To make it available to the host system, the container has to be run with an explicit port redirection. Using a common entry point for manipulating the infrastructure code helps a lot of sharing practices, enforcing policies, and integrating third-party services such as CI systems. This means that when a Puppet node is contacting the server for
the first time, a CSR is generated on the node and the Puppet server automatically signs it: subsequent requests will be authenticated and secured. Jenkins would work equally well behind the firewall when properly configured, like most other CI systems. Now, we wonder legitimately what would be the effect of changing the instance type, from
t2.micro to t2.medium: $ terraform plan -var aws instance type="t2.medium" [...] -/+ aws instance coreos [...] instance type: new resource) "t2.micro" => "t2.medium" (forces Plan: 1 to add, 0 to change, 1 to destroy. Now we have our pool of hosts with health checks, let's create the load balancer itself. de/blog/projects/socket-activation.html. STAT
START TIME Ssl + 23:30 0:00 This is suboptimal from a security point of view. Want to see which containing all of the IO? You can see it on the AWS S3 Console (: Let's store a first object right now, a very simple file containing as simple string ("Hello Infrastructure-as-Code Cookbook!"). During the last few years, he has been designing
distributed and high-available infrastructures on public/private cloud infrastructures. In this case, our simple cookbook contains two recipes, so we get two specs. If for some compliance reason we were to use a specific CentOS 7 release, specific tags such as centos:7.2.1511,
Vagrant is also very useful to simulate full production environments, with multiple machines and specific operating system versions. Download the latest Ghost versions curl -L -o ghost.zip 34 Chapter 1 4. We suggest to include at least three requirements: ff ChefSpec itself ff The Berkshelf plugin for dependencies management ff Immediately start the
code coverage Here's our sample spec helper.rb file: require 'chefspec' require 'chefspec
which htop /usr/bin/htop $ which curl /usr/bin/curl $ which curl /usr/bin/curl $ which wget /usr/bin/wget Good thing! Now we're sure to always have a fully updated system with the required set of tools installed, even our own, right from the beginning. Simply add the new mysite cookbook that includes everything it needs to run: $ knife node run list add vagrant mysite The
next Chef client run won't change anything, but it will be much easier to manage in the future! There's more... Using puppet, a module can be used in other modules. This will all happen inside a provisioner "chef" {} block inside the aws instance resource. 6. The Puppet infrastructure is mainly composed of: ff A Puppet server acting as a main
configuration server, which contains all the configuration code ff A Puppet agent running on all infrastructure nodes, applying configurations Communication between agents and the server is done through HTTPS, and Puppet has its own PKI for the server is done through HTTPS, and Puppet has its own PKI for the server is done through HTTPS.
server). We start from nothing, copy the binary, and execute it. 7. For example, using a recent Fedora, and installation goes like this: $ sudo dnf installation worked as expected: $ chef --version Chef
Development Kit Version: 1.1.16 That's it! Everything we need to start coding Chef recipes is there. We want the container to be destroyed afterwards (use --rm): $ docker run -it --rm ubuntu:16.04 /bin/bash :/# hostname d372dba0ab90 We've run our first container! Now do whatever you want with it. To do so, let's adjust the
Puppetfile in order to download the official Mysql module from Puppet Labs and keep our existing modules. We award our regular reviewers with free eBooks and videos in exchange for their valuable feedback. Install a bunch of PHP dependencies: sudo yum install -q -y php-mbstring php-mbstring php-dom 5. We'd like our
users to request this image as one of the following: ff terraform: latest (it's still the latest version) with the following tags: $
docker build -t terraform:latest -t terraform:unstable -t terraform:o.8.0-rc1 . I suggest using a different path, just to separate real users and application users. If you've never heard about it, Markdown is a language used to format a document in plain text mode, in order to export it in HTML. The idea behind the multiple tiers is to separate the logic
and execution of the various elements of the application, and not centralize everything in one place. Here, we'll deploy a Ghost blog with an NGINX reverse proxy, all on Docker, using an Ubuntu Xenial 16.04 on AWS EC2! This is to simulate a real deployment of an application, so you can see if it is working in real conditions. How to do it... We'll
local machine), or the other option -p 8000:8000 to make it fixed: $ docker run -ti --rm -P --name hello On another terminal, find the port redirection: $ docker port hello 8000/tcp -> 0.0.0.32771 Also, try to connect to it: $ curl -I HTTP/1.1 200 OK These are the basics of networking with Docker containers. A noticeable bug in the sharing feature
using VirtualBox leads to corrupted or non-updating files. We'll see how to create a CloudWatch Log group and use it to stream logs from a Docker container logs inside it. Think only a few megabytes. To exclude the FC003 rule, use the -t option: $ foodcritic -t ~FC003 --exclude test cookbooks/mysite/ FC033: Missing template:
cookbooks/mysite/recipes/htaccess.rb:9 FC033: Missing template: cookbooks/mysite/metadata.rb:1 FC065: Ensure source url is set in metadata: cookbooks/mysite/metadata.rb:1 FC065: Ensure source url is set in metadata: cookbooks/mysite/metadata.rb:1 FC065: Ensure source url is set in metadata: cookbooks/mysite/metadata.rb:1 FC065: Ensure source url is set in metadata: cookbooks/mysite/metadata.rb:1 FC065: Ensure source url is set in metadata: cookbooks/mysite/metadata.rb:1 FC065: Ensure source url is set in metadata: cookbooks/mysite/metadata.rb:1 FC065: Ensure source url is set in metadata: cookbooks/mysite/metadata.rb:1 FC065: Ensure source url is set in metadata: cookbooks/mysite/metadata.rb:1 FC065: Ensure source url is set in metadata: cookbooks/mysite/metadata.rb:1 FC065: Ensure source url is set in metadata: cookbooks/mysite/metadata.rb:1 FC065: Ensure source url is set in metadata: cookbooks/mysite/metadata.rb:1 FC065: Ensure source url is set in metadata: cookbooks/mysite/metadata.rb:1 FC065: Ensure source url is set in metadata: cookbooks/mysite/metadata.rb:1 FC065: Ensure source url is set in metadata: cookbooks/mysite/metadata.rb:1 FC065: Ensure source url is set in metadata: cookbooks/mysite/metadata.rb:1 FC065: Ensure source url is set in metadata: cookbooks/mysite/metadata.rb:1 FC065: Ensure source url is set in metadata.rb:1
proposed by the FC003 rule. More information about all integrated providers can be found on Vagrant's website: . This is done using the remote-exec provisioner and requires a SSH username. Multi-machine recovery Let's move to a more complicated setup. This means that the Docker code should be testable, the builds automatic, and the CI systems
connected to our Git servers so they could continuously apply the tests. Testing the Chef version update So we simulated our production environment locally, with the same CentOS version, the apache2 cookbook used in production environment locally, with the same CentOS version, the apache2 cookbook used in production environment locally, with the same CentOS version, the apache2 cookbook used in production, and the old Chef version 11. [...] Finished creating (1m1.51s). It's a much better option to use this encrypted password
to log in to the service. We'll use one of the best platforms to run containers, the free and open source lightweight operating system CoreOS. Until now, the job of the shared folder of our Vagrant setup was to map the relative directory puppetcode to /etc/puppetlabs/code/ environments/production in the puppet.pomes.pro box. Chapter 3, Going
Further with Terraform, sheds light on some more advanced techniques of using Terraform code, such as dynamic data sources, separate environments, Docker, GitHub or StatusCake integration, team work, and how the code linter works. We decide this one will be 10 GB in size. $ git commit -m "updated production env with platform::docker" Wait
for the next Chef run or execute it yourself, and from now on, we'll have a nice four-step workflow: 1. The good news is that Chef helps a lot in mapping this model to the infrastructure. You're at the right place: 405 Maintaining Docker Containers Here's an example of a Tomcat container with a view of all the local and remote connections, IPs, ports,
protocols, bandwidth, IOs, and the corresponding commands—terribly useful to find suspicious behavior: Another useful tool is F5/Echo, grabbing what's transiting on this container: (un)encrypted content, logs, output, and more. Chapter 2, Provisioning IaaS with Terraform, is everything needed to get started with Terraform on Amazon Web
Services, from managed database servers to log handling, storage, credentials, Docker registries, and EC2 instances. See also ff ChefSpec: ff Rspec: ff Rspe
in the Chef ecosystem as it enables thorough testing of infrastructure code and plays very well with a lot of other tools we already use and know. Let's say we're using a global Makefile to do that. Now let's send this data on the Chef server using the knife command: $ knife data bag from file webusers mary.json Updated
data bag item[webusers::mary] $ knife data bag from file webusers john.json Updated data bag item[webusers john mary Now the data bag item[webusers john mary Now the data bag from the Chef
server, how do we access it dynamically from inside our code? He has been working in the IT industry for the last twenty years mostly in C development, system administration, and security including PCI-DSS. This includes a package to be installed or removed, a service to be enabled or started, a file to be generated from a template, a user to be
created or banned, and other expected elements of a system. Managers are controlling what's executed on the nodes. The possible keys and values can be found on most VMware Inc. There's more... Using Ansible, we can create a role to do the same job. This book has been written keeping in mind all those teams that we have met in the past few
years through our respective jobs—teams interested in DevOps, automation, and code, sometimes already doing part of it quite well, but willing to discover other tools and techniques, discovering how they could do better by improving the quality of their services, the speed of their
deployments, the efficiency of team work, and the feedback loop. Release names are tagged like the corresponding versions, so the image debian: jessie. It seems to do wonders; there are many configuration options, and many other things. We highly suggest that you include this common cookbook in every new host deployment
process so they could all be automatically configured to converge at a predetermined interval. What this book covers Chapter 1, Vagrant Development Environments, is all about automated development environments, is all about automated development environments.
samples from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, and try creating a new environment, namely test. If we upload it now as is, it won't be encrypted. Our goal is to serve, through an anginx-proxy container, a blog container, a blog container (Ghost) that will not be directly available on the network. Finally, remove the reference to the
mariadb recipe from the default mysite recipe in mysite/recipe/default.rb, and replace it with a call to the new mysql recipe include recipe in mysite::mysql" Including dependencies in a role To fully match our environment with what we just did in the cookbook, let's remove the call to the old mariadb cookbook from the database role, and as there's no
recipe to call (as we said, this cookbook is just extending Chef functionality), let's instead add the two cookbook dependencies as stated in the documentation in roles/database.rb: name "database" description "A database server for our application" run list("recipe[selinux::disabled]", "recipe[yum-mysql-community::mysql57]") Upload the updated
role using the knife command: $ knife role from file database role will be alright. If you are ready, time now to... ChefSpec helps to create a fast feedback loop, locally simulate Chef runs (solo or server) over the code, and issue a code coverage statement
for every resource used. To build our container, we need a file named Dockerfile, acting like a script, executed line by line, to build the final container. Other Vagrantfile sharing options include Windows Sharing (smb), rsync (useful with remote virtual machines such as on AWS EC2), and even NFS. In our case, we expect Apache to listen on port 80
(HTTP) and we configured MySQL to listen to 3306. Install the dependencies using Berkshelf: $ berks install 3. Writing objects: 100% (9/9), 2.60 KiB | 0 bytes/s, done. [...] Navigate to the AWS EC2 dashboard under Instances | Instance
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 4.4.0-36-generic x86_64) :~$ You can apply and therefore it doesn't recreate endlessly new VMs each time. Installed the plugin 'vagrant-omnibus (1.4.1)'! Then, just add the
```

```
following configuration in your VM definition of the Vagrantfile and you'll always have the latest Chef version installed on this box: config.omnibus.chef version installed on this box: config.omnibus.chef version we want:
config.omnibus.chef version = "11.18.12" 37 Vagrant Development Environments Now that we're using a new plugin, our Vagrantfile won't work out of the box for everybody. Done Calculating upgrade... Several tools can be used to manage packages; r10k is one of them. We'll see how we can integrate and automate a Heroku environment. Many
providers propose their own automation tool, but the power of Terraform is that it allows you to manage it all from one place, all using code. 52.209.159.4 = eu-west-1 d9fa1d18... Let's do the same for our database role; we currently want to use our mariadb cookbook. [...] ==> front: 172.17.0.1 - - [21/Aug/2016:10:55:08 +0000] "GET / HTTP/1.1" 200
 1547 "-" "Mozilla/5.0 (X11; Fedora; Linux x86 64; rv:48.0) Gecko/20100101 Firefox/48.0" "-" ==> app: GET / 200 113.120 ms - [...] A Docker Compose equivalent Docker Compose is a tool to orchestrate multiple containers and manage Docker features from a single YAML file. 253 Testing and Writing Better Infrastructure Code with Chef and Puppet
ERROR: trailing whitespace found on line 11 ... See also This section provides helpful links to other useful information for the recipe. With Digital Ocean, the resource is named digitalocean ssh key. Here's a sample entrypoint.sh that only does simple checks for the USER and PASSWORD environment variable, creates the required users on the
container, sets some permissions, and finally executes the CMD instruction from the original Dockerfile: #!/bin/sh # Checks for USER variable (ie.: -e USER=john).' exit 1 fi # Checks for PASSWORD variable if [-z "$PASSWORD"]; then echo >&2 'Please set an USER variable (ie.: -e USER=john).' exit 1 fi # Checks for PASSWORD variable if [-z "$PASSWORD"]; then echo >&2 'Please set an USER variable (ie.: -e USER=john).' exit 1 fi # Checks for PASSWORD variable (ie.: -e USE
PASSWORD=hackme).' exit 1 fi echo "Creating user ${USER}" chown -R ${USER}" chown -R
check from the entrypoint.sh script: $ docker run --rm ssh Please set an USER variable (ie.: -e USER=john). It will be interesting to use them if our product is definitely going to use the our product is definitely going to us
Puppet agent as a cron, we need to declare a Puppet cron resource as follows: # puppet resource cron puppet-agent --onetime --no-daemonize --splay imit 60' The generated crontab file is: $ sudo cron
daemonize --splay --splaylimit 60 291 Maintaining Systems Using Chef and Puppet In this example, the Puppet agent will run every hour. We can use the Docker Hub service to store and distribute our containers (or its alternative Quay.io); however, uploading each and every container and version manually will soon be a problem. We may also want to
reboot the server after applying critical initial packages such as the kernel, or add a custom package repository. Getting ready To step through this recipe, you will need: ff A working Chef DK installation on the workstation ff A working Chef
Servers with Chef and Puppet, Chapter 7, Testing and Writing Better Infrastructure Code with Chef and Puppet How to do it... Chef environments live in a folder named environments at the root of chef-repo. The Vagrant version in use for this book is Vagrant 1.8.4 Adding an
Ubuntu Xenial (16.04 LTS) Vagrant box vagr
those tags: $ docker images terraform REPOSITORY SIZE TAG IMAGE ID CREATED terraform minute ago 0.7.12 83.61 MB 9d53a0811d63 About a terraform minute ago latest 83.61 MB 9d53a0811d63 About a terraform minute ago latest 83.61 MB 9d53a0811d63 About a terraform minute ago 1.7.12 83.61 MB 9d53a0811d63 About a
container for our team to test it out. No risk of using an old state file if we forgot to synchronize our git repository. In the Ubuntu case, you can find the AMI you want by going to their Amazon EC2 AMI Locator page (https:// cloud-images.ubuntu.com/locator/ec2/). Let's use the starter cookbook we uploaded earlier in this chapter, but feel free to use
 any other cookbook you may have already synchronized on the Chef server. That's 15% space saved! Replacing the ubuntu:16.04 image with the debian:stable-slim RUN apt-get update -y && \ apt-get install -y apache2 && \ rm -rf /var/lib/apt/
ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"] How it works... Each layer is added to its predecessor. Now add a configuration file for Travis even gives easy access to the output of the commands: We just initiated new steps for integrating automated tests
in our workflow. This is done by having master as the Branch type of our repository; tag the build with the latest tag. Getting ready To step through this recipe, you will need: ff Access to a cloud-config enabled infrastructure 171 Provisioning the Last Mile with Cloud-Init How to do it... To set the server's time zone automatically to America/New York,
use the timezone directive: #cloud-config timezone: $ cat /etc/timezone America/New_York Managing users, keys, and credentials
using cloud-init There's a high probability we won't plan to use the default root account, or even the default user account from our distribution (those ubuntu or centos users). 357 Working with Docker Getting ready To step through this recipe, you will need the following: ff A working Docker installation ff A sample HTTP server binary (sample code
included) How to do it... Let's take a simple HTTP server that answers on the port 8000 of the container. That's why Terraform has a command to ensure both format and style are all right. [...] Our colleague is happy, and we're now sure that this change will do exactly what's intended. Is your Dockerfile empty? An IAM user for EC2 in read-only Is
there a similar managed policy for Joe, with a read-only scope on EC2? aws_access_key: cipher: aes-256-cbc encrypted_data: RwbfsWgKk16sSCkMD38tXKGHmT1AHFGHRm/7fyzppye7wSS0kk19Zml 0VuhQ XxxI iv: iRRgrKfz6Ou2qdpYLkUA+w== version: 1 aws secret key: cipher: aes-256-cbc encrypted_data:
uSppKMYrRbEYn/njDYo3CIGC5tY+pptN1Z7LiARtNIU/zsllBNdSVENC1XwX QksifE6g00sdcHTGlHlVU0WJ0Q== 299 Maintaining Systems Using Chef and Puppet iv: ppjeAJcegZ9Yyn9rXgHRBQ== version: 1 id: us-east-1 It looks like we got what we wanted: data is stored encrypted on the Chef server! As it may not be a secure move to store
unencrypted data bags on version control systems, such as Git, we can ask for a JSON-formatted encrypted version, such as the following, and redirect the output to a JSON file for storage purposes: $ knife data bag show aws us-east-1 -Fj { "id": "us-east-1", "aws access key": { "encrypted data":
 "RwbfsWgKk16sSCkMD38tXKGHmT1AHFGHRm/7fyzppye7wSS0kk 19Zml0VuhQXxxI", "iv": "iRRgrKfz6Ou2qdpYLkUA+w==", "version": 1, "cipher": "aes-256-cbc" }, "aws_secret_key": { "encrypted_data": "uSppKMYrRbEYn/njDYo3CIGC5tY+pptN1Z7LiARtNIU/ zsllBNdSVENC1XwXQksifE6g00sdcHTGlHlVU0WJ0Q==", "iv"
 "ppjeAJcegZ9Yyn9rXgHRBQ==", "version": 1, "cipher": "aes-256-cbc" } This might be the content you'd like to store on Git! Accessing an encrypted data bag in the CLI To access unencrypted data bag show aws us-east-1 --secret s3cr3t
 Encrypted data bag detected, decrypting with provided secret. Chapter 6 Other modes exist, such as Chef Zero, but they are beyond the scope of this book. First of all you need to install the puppet-agent package from Puppet Collections on your workstation. If you remember, you had to enter once again your keys in the Docker Engine configuration.
Let's define multiple containers and start by creating a front container (our previous NGINX): config.vm.define "front" do |front| front.vm.provider "docker| docker|
blog engine such as Ghost? This is how we'd do it: FROM alpine:latest RUN apk --update --no-cache add ca-certificates openssl && \ rm -rf /var/cache/apk/* COPY src/hello/hello /bin/hello ENTRYPOINT ["/bin/hello"] 350 Chapter 9 Such an image usually is only a handful of megabytes more than the application binary, but helps greatly for debugging. It
also means that you can use a Mac laptop, and with a simple command, launch a fully configured Linux environment for you to use locally. Run the berks command from inside the cookbook dependencies... Each VM stores only its delta state, allowing very fast
virtual machines boot times. When creating a new droplet, be sure to tick the User data checkbox under the Select additional options section and paste the cloud-config file content: 167 Provisioning the Last Mile with Cloud-Init After a few seconds of boot time and package installation, our customized Ubuntu distribution is available:
:~# which htop /usr/sbin/tcpdump :~# which htop /usr/sbin/tcpdump :~# docker --version Docker version 1.11.2, build b9f10c9 Using cloud-init on OpenStack When creating an instance on OpenStack, using the Horizon dashboard, click on the PostCreation tab, and paste the cloud-config YAML
 content in the text box. 1 Let's add this variable to a variable to a variable "aws_region" { default = "eu-west-1" } Hooray! A terraform validate is now happy: $ terraform validate $ echo $? As it's Ruby, you can access all available variables, so the message can be even more dynamic and useful to the user: config.vm.post_up_message = "Use
\"vagrant ssh\" to log into the box. Verifying Puppet applies properly. Customer support Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase. We unit tested the feature by stubbing the data bag, and then using Test Kitchen, we configured it to simulate the availability of those
data bags. We'd probably just execute the following and check the output: $ apache2ctl -v Server version: Apache/2.4.10 (Debian) This translates in Docker with our image in the following command (-v being the command (CMD) for the apache2ctl ENTRYPOINT instruction): $ docker run --rm demo-httpd:latest -v Server version: Apache/2.4.10
(Debian) Basically, now we just have to run grep for the correct version: $ docker run --rm demo-httpd:latest -v | grep 2.4.10 (Debian) 371 Maintaining Docker Containers If grep is successful, it returns 0: $ echo $? Now upload the common cookbook with all its dependencies: $ berks upload 4. Here's how to create a
dynamically named log stream using the systemd unit name (in this case, tomcat.service). Modules hosted on Puppet module search mysql | head -10 Notice: Searching ... This is constructing an SSH command line with dynamic information from Terraform that we'll be ableated using the systemd unit name (in this case, tomcat.service).
to use easily (it's simply iterating over every digitalocean_droplet.coreos.* available): output "CoreOS Cluster Members" { value = "${formatlist("ssh %v -i ${var.ssh key file}", digitalocean_droplet.coreos.* ipv4_address)}" } The output will look like this: CoreOS Cluster Members = [ssh -i ./keys/iac_admin_sshkey.
ssh -i ./keys/iac admin sshkey, ssh -i ./keys/iac admin sshkey admin
variable with the amount of virtual machines we want (2): vm num = 2 Then iterate through that value, so it can generate values for an IP and for a hostname: (1..vm num).each do |n| # a lan lab in the 192.168.50.0/24 range lan ip = "192.168.50.0/24 range
\"inet\" | awk '{print $2}'" config.vm.network "private network", ip: lan_ip, virtualbox_ intnet: "true" end end This will create two virtual machines (srv-1 at 192.168.50.11) on the same internal network, so they can talk to each other. Vagrant took care of configuring SSH by exposing an available port and inserting a default
key, so you can log into it via SSH without problems. The exception is that we are lucky enough to have tools that monitor each and every layer of our Docker images and tell us how and when they are vulnerable, allowing us to simply rebuild and redeploy them. Let's find this out by including Vagrantfile at the root of the infrastructure repository
using the previous project code for deploying Docker (but the idea is the same for any kind of Chef repo). The code needs to actually be on GitHub not just versioned using Git locally. Let's create a variable "worker token file" { default = "worker token file" } We will
execute the previous docker swarm command through SSH when everything else is done, using a local-exec provisioner. Targeting for a specific change Our colleague asks us if we're sure our proposed changes have no impact specific alky on the S3 bucket. The default NGINX configuration reads files from /usr/share/nginx/html. Typically, when using
notifications for restarting a service from a file change, the concerned service must be included in the recipe where the file resource is located; otherwise, the code most probably works by chance because the required dependent cookbook is included elsewhere! Here's how to test for a cookbook inclusion: it 'includes the `apache` recipes' do
expect(chef_run).to include recipe('apache::default') expect(chef_run).to include recipe('apache::virtualhost') end We now ensure that dependencies are always included. Let's create the Docker service and start it using the docker service resource the cookbook is offering us: docker_service 'default' do action [:create, :start] end Execute kitchen
again to apply our code and see what the tests say: $ kitchen converge $ kitchen converge
case, the Vagrant plugin for VMware delegates all the virtualization features to the VMware installation, removing the need for VirtualBox. If this happens, will we be able to rebuild it from scratch? When using locally for this exercise, you can just forget about including the provider "docker" { host = "tcp://1.2.3.4:2375" } Let's start by
declaring two data sources for each of our Docker images (in docker.tf). There's more... We can optionally go even further, using the following two
dependencies from the mysql cookbook 'selinux' cookbook 'selinux' cookbook 'selinux' cookbook, the latest being the default, and a loose constraint on any minor revision of the yummysql-community 1.0 cookbook. You can then distribute this simple file with other
people, team members, and external contributors, so that they immediately have a working running environment as long as they have virtualization on their laptop. Here, we'll create two generic database and webserver roles that might be simply reused later for another project that just need those services and a mysite role, that will include the two
other roles. conf"], notify => Service['apache2'], } 257 Testing and Writing Better Infrastructure Code with Chef and Puppet The documentation Docker image installs Debian Jessie (FAILED - 1)
 Failures: 1) Docker image installs Debian Jessie Failure/Error: @image = Docker::Image.build from dir('.') Docker::Error::ServerError: No image was generated. There's a simple Makefile example: .DEFAULT GOAL := all all: terraform validate terraform
fmt Now you can just type make in the Terraform directory and you're sure your code both validates and is coherently styled. This build also allows us to manage different purposes, such as building the development and production containers differently, among other options. Let's add it to metadata.rb so we depend on it:
depends 'docker', '~> 2.0' Add it to Berkshelf as well if you plan to use it in Berksfile: cookbook 'docker', '~> 2.0' As we'll be writing our Docker code in the platform::docker recipe, let's start by including it in the default.rb recipe: include recipe 'platform::docker cookbook provides us with a new resource named docker installation that
does just this: install docker. 52.31.109.156 =eu-west-1 METADATA cluster=mycluster, provider=aws, region cluster=mycluster=mycluster, provider=aws, region cluster=mycluster, provider=aws, region cluster=my
init CoreOS can handle updates in various ways, including rebooting immediately after a new CoreOS version is made available, scheduling with etcd for an ideal time so the cluster never breaks, a mix of both (the default), or even to never reboot. Good! You learn that our first instance will be destroyed and recreated to move from t2.micro to
t2.medium, and that a second instance will be created with the same values. For example, let's say our cloud instance is launched with an attached block storage. This is requested by the vagrant-hostmanager plugin in order to create entries in the /etc/hosts file of your workstation. Here's an aws instance resource in instance stable.
release taken from previous recipes: resource "aws_instance" "coreos" { count = "${var.aws_coreos_ami}" instance_type = "${var.aws_instance_type}" key_name = "${var.aws_instance_type}"
Here are example variables in variables in variables in variables in variable "aws coreos ami" { default = "ami-85097ff6" } variable "aws_instance_type" { default = "t2.micro" description = "Instance type" } Terraform stores its state by default in a file named
terraform.tfstate, with a backup file named terraform.tfstate.backup: $ ls terraform.tfstate terraform.tfstate terraform.tfstate terraform.tfstate terraform.tfstate terraform.tfstate.backup Sharing with Git The simplest of all options is to share the state file using Git: you're already supposed to version your infrastructure code! Go and create an account somewhere. In this section, you'll discover how to
manage system services, using a resource named service. There's more... Linters are also used daily for most languages by software developers. Upload the new cookbook version: $ knife cookbook version
remote host: $ sudo chef-client Verify the Apache service is indeed running: $ systemctl status httpd You can also navigate to the site's IP address in HTTP to see the default page displayed. 318 Chapter 8 Let's reuse our Vagrant LAMP example in Chapter 6, Fundamentals of Managing Servers with Chef and Puppet. This way, this particular Docker
Engine will answer to requests on the TCP socket and not on the Unix socket (while keeping the default docker.service clean). One solution to stop thinking about it is using S3 to share the state file and use the remote state feature of Terraform. Now our systems are as much as possible
 automated, hopefully 100 percent. Let's add a new node into the Vagrantfile we previously used for the Puppet server: vm memory = 2048 vm cpus = 2 unless Vagrant.configure("2") do |config| config| config| config| hostmanager enabled = true
config.hostmanager.manage guest = true 202 Chapter 6 config.hostmanager.manage host = true config.vm.define "puppet.vm.box="bento/ubuntu-16.04" puppet.vm.box="bento/ubuntu-16.04" puppet.vm.bo
:private network, ip: "192.168.50.10" puppet.vm.provision :shell, :path => "puppet master.sh" puppet.vm.provision :shell, :path => "puppet.vm.provision :shell, :path => "puppet.vm.provision :shell, :path => "puppet.shell :shell :shel
web.vm.network :private_network, ip: "192.168.50.11" web.vm.provision :shell, :path => "puppet_node.sh used for the provisioning of this new node: #!/usr/bin/env bash # Exit immediately if a command exits with a non-zero status set -e # puppetlabs URL DEBREPO=" #
Install the PuppetLabs repo echo "Configuring PuppetLabs repo..." debrepo} ${DEBREPO} dpkg -i ${debrepo} {DEBREPO} dpk
box echo "Installing Agent..." apt-get install -y puppet agent # Ensure puppet agent is stopped for our tests /opt/puppet labs/bin/puppet resource service puppet ensure=stopped enable=false echo "Puppet agent is stopped for our tests /opt/puppet labs/bin/puppet resource service puppet agent is stopped for our tests /opt/puppet labs/bin/puppet resource service puppet agent install -y puppet agent install -y puppet agent install -y puppet agent installed!" We now also have an Ubuntu Puppet agent install -y puppet agent agent
add the following: This system version listening on is up since To build something more interesting, as the platform name is available, let's make our apache cookbook a bit more portable across Linux distributions. This is not a Laravel book, but a way to create a new project in a clean directory is as follows: cd /srv/app composer create-project --
prefer-dist laravel/laravel. ----> Verifying ... What's going on in the system? A good practice, if you use a feature that is known to work only after a specific version, is to declare it in the Vagrant.require version, is to declare it in the Vagrant Development environments can
 often be mixed, using both virtual machines and Docker containers. For added security, it's better to not hardcode the shared key—use the key file that is sent separately (but this creates an added layer of complexity in the deployment system). With it, you can dynamically create machines at two IaaS providers depending on the environment, register
their names at another DNS provider, enable monitoring at a thirdparty monitoring company, while configuring the company GitHub account and sending the application logs to an appropriate service. We can now apply changes, using puppet agent --test: $ vagrant ssh web.pomes.pro Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 4.4.0-51-generic
x86 64) ... Learning the most important tools, techniques, and workflows that fit in an infrastructure-as-code description can be a daunting task, and many teams can either be misled or discouraged by the amount of information, change, and knowledge required to switch to infrastructure-as-code. For example, we can imagine an apache module
containing resources for Apache installation and service management. Add the common cookbook to the host run-list: $ knife node run list add vagrant common 5. ----> Kitchen is finished. It varies from every other OpenStack installation. Total Resources: 5 Touched Resources: 5 Touch
coverage! Have a fantastic day! Testing a package installation Our default recipe starts by installing the httpd package. [...] ==> default: HTTP access: Let's remove the need to specify the provider on the command line by setting a simple Ruby environment access code at the top of the Vagrantfile: ENV['VAGRANT DEFAULT PROVIDER'] = 'docker'
Now you can distribute your Vagrantfile and not worry about people forgetting to explicitly specify the Docker provider. This is the context, and we expect this Chef run to not raise any errors: describe 'apache::default' do context, and we expect this Chef run to not raise any errors: describe 'apache::default' do context, and we expect this Chef run to not raise any errors: describe 'apache::default' do context, and we expect this Chef run to not raise any errors: describe 'apache::default' do context, and we expect this Chef run to not raise any errors: describe 'apache::default' do context, and we expect this Chef run to not raise any errors: describe 'apache::default' do context, and we expect this Chef run to not raise any errors: describe 'apache::default' do context, and we expect this Chef run to not raise any errors: describe 'apache::default' do context, and we expect this Chef run to not raise any errors: describe 'apache::default' do context 'Default attributes on Chef run to not raise any errors: describe 'apache::default' do context 'Default attributes on Chef run to not raise any errors and attributes on Chef run to not raise any errors and attributes on Chef run to not raise any errors and attributes on Chef run to not raise any errors and attributes on the chef run to not raise any errors and attributes on the chef run to not raise and attributes on the chef run to not raise and attributes on the chef run to not raise and attributes on the chef run to not raise and attributes on the chef run to not raise and attributes on the chef run to not raise and attributes on the chef run to not raise and attributes on the chef run to not raise and attributes on the chef run to not raise and attributes on the chef run to not raise and attributes on the chef run to not raise and attributes on the chef run to not raise and attributes and attrib
'7.2.1511') runner.converge(described recipe) end it 'converges successfully' do expect { chef run }.to not raise error end a full list of available simulated platforms at fauxhai/platforms. Uncompress it in the /srv/blog folder: sudo
unzip -uo ghost.zip -d /srv/blog/ 5. If our AWS instance resource is named centos, you can access the value dynamically like this, instead of a static value: website url = "${aws_instance.centos.public_ip}" If your resource has a count number, you can iterate through it so all the available instances are automatically monitored. vii Preface How to do it..
This section contains the steps required to follow the recipe. Using the previous example, we now need to ensure that the corresponding services are running. To store our builds or releases, we'll create a container, which is in OpenStack terminology—an object storage. We're trying to keep the code simple here. Deleting the APT lists means the
complete (should happen in minutes), navigating to the Tags tab will show the available tags (latest is the only one we have now) and the size of the image: 384 Chapter 10 The Dockerfile tab shows the content of the Dockerfile from which the image has been built, while the Build Details tab will list all the builds and their details, including the build
 output. Using Vagrant remotely with AWS EC2 and Docker Another powerful usage of Vagrant can be with remote IaaS resources such as Amazon EC2. 1. Words that you see on the screen, for example, in menus or dialog boxes, appear in the text like this: "You can see your newly created security group by logging into the AWS Console and
navigating to EC2 Dashboard | Network & Security | Security Groups." Warnings or important notes appear in a box like this. While it's possible to include the Guest Additions directly in the box, not all the boxes you'll find have it, and even when they do, they can be outdated very quickly. Let's override it to match our own value by adding the
following at the top of the file, just before the apache recipes inclusion: node.override["sitename"] = "mysite" Upload the cookbook to the Chef server: $ knife node run list remove vagrant "recipe[mariadb]" "recipe[php]"
 "recipe[apache]" "recipe[apache::virtualhost]" 224 Chapter 6 The node's run list is now empty. This very useful for storing data that need to be accessed globally from a central point (such as users, service credentials, version numbers, URLs, even
feature flags, and other similar features depending on your usage). Using Ansible with Vagrant to create a Docker host Ansible (is a very simple and powerful open source automation tool. A short and interactive tutorial can be found on . Our specification table indicates we need two different S3 buckets as well. If A working Docker installation (native
 Docker for Mac, Docker Engine on Linux, a remote server running Docker on TCP, and so on). Maybe one minor container just crashed and is eating up all of the CPU. This is, however, just a small introduction to what can be done with Docker on the CPU. This is, however, just a small introduction to what can be done with Docker on the CPU. This is, however, just a small introduction to what can be done with Docker on the CPU. This is, however, just a small introduction to what can be done with Docker on the CPU. This is, however, just a small introduction to what can be done with Docker on the CPU. This is, however, just a small introduction to what can be done with Docker on the CPU. This is, however, just a small introduction to what can be done with Docker on the CPU. This is, however, just a small introduction to what can be done with Docker on the CPU. This is, however, just a small introduction to what can be done with Docker on the CPU. This is, however, just a small introduction to what can be done with Docker on the CPU. This is, however, just a small introduction to what can be done with Docker on the CPU. This is, however, just a small introduction to what can be done with Docker on the CPU. This is, however, just a small introduction to what can be done with Docker on the CPU. This is, however, just a small introduction to what can be done with Docker on the CPU. This is, however, just a small introduction to what can be done with Docker on the CPU. This is, however, just a small introduction to what can be done with Docker on the CPU. This is, however, just a small introduction to what can be done with Docker on the CPU. This is, however, just a small introduction to what can be done with Docker on the CPU. This is, however, just a small introduction to what can be done with Docker on the CPU. This is, however, just a small introduction to what can be done with Docker on the CPU. This is well as the CPU
Compose documentation: Linting a Dockerfile Like any other language, Dockerfiles can and should be linted for updated best practices and code quality checks. Let's include all of this information in our mysite cookbook's metadata.rb: depends 'mysql2 chef gem', '~> 8.1' 314 Chapter 8 To build the
mysql2 gem using the cookbook's new mysql2 chef gem resource, we'll need the MySQL development package named mysql-community-devel. Ghost publishes a ready-to-use container on the Docker Hub, so let's use that (version 0.9.0 at the time of writing) and expose on TCP/8080 the application container listening on TCP/2368: config.vm.define
 "app" do |app| app.vm.provider "docker" do |docker| docker do |docker| docker.image = "ghost:0.9.0" docker.ports = ['8080:2368'] end end 16 Chapter 1 Check if you can access the blog on and NGINX on : $ curl -IL HTTP/1.1 200 OK X-Powered-By: Express [...] $ curl -IL HTTP/1.1 200 OK Server: nginx/1.10.1 Now let's use NGINX for what it's for—serving the
application. We also need an eyaml configuration file, located in ~/.eyaml/config.yaml, with this content (do not forget to adjust the path of your $HOME directory): --pkcs7_public key. "/Users/me/keys/public key.pkcs7.pem" We are now ready to encrypt sensitive data. We can use the
excellent dynamic Nginx image from jwilder/nginx-proxy to do this job and add a new proxy service, sharing port 80 and the local Docker socket as readonly (to dynamically access running containers) on a docker-compose.production.yml file: proxy: image: jwilder/nginx-proxy ports: - "80:80" volumes: - /var/run/docker.sock:/tmp/docker.sock:ro This
nginx-proxy container needs a variable named VIRTUAL HOST to know what to answer in case of multiple virtual hosts. For example, all available chef-client options, and the Chef environment (usually very important) is passed using environment as a string. Our custom tool can send each Docker image
layer we're interested in and get the corresponding vulnerabilities or fixes. Here's how to do this using an earlier image from this chapter: $ docker push quay.io/sjourdan/nginx-docker-demo:1.0 $ docker-demo:1.0 $ dock
82819c620e5d: Pushed d07a4f6d2067: Pushed 391 Maintaining Docker Containers Quay.io has a very nice security feature: as Docker stores passwords in plain text on the local workstation, it's possible to generate an encrypted password from the settings tab of your Quay.io has a very nice security feature: as Docker stores password from the local workstation, it's possible to generate an encrypted password from the settings tab of your Quay.io has a very nice security feature: as Docker stores password from the local workstation, it's possible to generate an encrypted password from the settings tab of your Quay.io has a very nice security feature: as Docker stores password from the settings tab of your Quay.io has a very nice security feature: as Docker stores password from the settings tab of your Quay.io has a very nice security feature: as Docker stores password from the settings tab of your Quay.io has a very nice security feature: as Docker stores password from the settings tab of your Quay.io has a very nice security feature: as Docker stores password from the settings tab of your Quay.io has a very nice security feature: as Docker stores password from the settings tab of your Quay.io has a very nice security feature: as Docker stores password from the settings tab of your Quay.io has a very nice security feature and tab of your Quay.io has a very nice security feature and your Quay.io has a very nice security feature and your Quay.io has a very nice security feature and your Quay.io has a very nice security feature and your Quay.io has a very nice security feature and your Quay.io has a very nice security feature and your Quay.io has a very nice security feature and your Quay.io has a very nice security feature and your Quay.io has a very nice security feature and your Quay.io has a very nice security feature and your Quay.io has a very nice security feature and your Quay.io has a very nice security feature and your Quay.io has a very nice security feature and your quay.io has a very nice security feature a
Ubuntu, we need to remove the default website in order to have virtual hosting up and running. This is the perfect job for a CI system, such as Jenkins, the Circle CI, or the Travis CI. ix Preface You can download the code files by following these steps: 1. Is there a container taking up all of the memory? Your workstation will need the Chef
Development Kit (Chef DK: https:// downloads.chef.io/chef-dk/) for this to work correctly. 193 Fundamentals of Managing Servers with Chef and Puppet Chef resource is any part of a system to be set in a desired state. managing Terraform used 153 application, creating with Terraform 154
apps. All the critical issues in this example only concern Apple platforms and we're running Linux containers. Add and commit whatever work is in the repository right now, if any: $ git add . It's often better to reboot as soon as possible with the most secure kernel, but proceed with caution according to your own environment (you might not want to
reboot while another action is taking place, maybe a Chef run or similar management software): apt reboot if required: true To ensure the required packages are installed, use the packages directive: packages directive: packages are installed, use the pac
a new instance and verify it's fully updated, so no updates can be applied: $ sudo apt-get dist-upgrade Reading package lists... The final command is the ext4 formatting of a block device attached on /dev/xvdb. Getting ready To step through this recipe, you will need: ff A working Chef DK installation on the workstation ff A working Vagrant installation
on the workstation ff The Chef code (optionally) from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code How to do it... Our goal is to start a new platform based on Docker. About the Reviewer Pierre Mavro lives in a suburb of
Paris. An attribute is similar to a persisting node setting, declared in a cookbook in a file under the attribute directory, and can then be overridden by many mechanisms that we'll later discover. In this chapter, we will first write our own code in order to learn some basics of the Puppet DSL. While using and creating Ansible playbooks is off-topic for
this book, we'll use a very simple playbook to install and configure Docker on a CentOS 7 box. Let's make sure git is installed: package 'git' The deploy revision resource is the most complex of all. On top of that, it can delegate configuration to those who do it well (configuration management tools such as Chef, Puppet, and so on), all with the same
tool. Getting ready To step through this recipe, you will need the following: ff ff ff A working Terraform installation. Chapter 8, Maintaining Systems Using Chef and Puppet, shows advanced features made possible by Chef or Puppet code, such as scheduled convergence, encrypted secrets, environments, live system information retrieval, application
deployments, and workflows or practices to be safe. Chef environments Every node runs inside an environment. A common pattern is to get a first layer that gets the common user requests, a second layer that does the application job, and a third layer that stores and retrieves data, usually from a database. pp): define apache::vhost ($website, a second layer that stores and retrieves data, usually from a database. pp): define apache::vhost ($website, a second layer that stores and retrieves data, usually from a database. pp): define apache::vhost ($website, a second layer that stores and retrieves data, usually from a database. pp): define apache::vhost ($website, a second layer that stores and retrieves data, usually from a database. pp): define apache::vhost ($website, a second layer that stores and retrieves data, usually from a database. pp): define apache::vhost ($website, a second layer that stores and retrieves data, usually from a database. pp): define apache::vhost ($website, a second layer that stores and retrieves data, usually from a database. pp): define apache::vhost ($website, a second layer that stores and retrieves data, usually from a database. pp): define apache::vhost ($website, a second layer that stores and retrieves data) and the second layer that stores are the second layer than the second layer that stores are the second layer than the second layer that stores are the second layer than the second
$docroot) { file { "/etc/apache2/sites-available/$website.conf": ensure => |root', group => 'root', group => |root', group =
 => File["/etc/apache2/sites-available/$website.conf"], notify => Service['apache2'], } Let's try to run the Puppet agent on fresh Vagrant boxes, we will see that the two modifications will schedule a configuration reload, that will be done at the end of the Puppet Agent run. Here's the code to do this: config.omnibus.chef_version = :latest Let's
configure the Vagrant provisioning system to use Chef Zero in order to simulate a Chef server. There's more... If VMware is your primary hypervisor, you'll soon be tired to always specify the provider in the command line. $ git commit -m "an initial infrastructure" [new infrastructure 2415ad4] an initial infrastructure 6 files changed, 65 insertions(+)
create mode 100644 instances.tf create mode 100644 keys/aws_terraform create mode 100644 keys/aws_terraform.pub create mode 100644 keys/aws_terraform create mode 100644 keys/aws_terraform.pub create
production: $ git push --set-upstream origin new infrastructure. To see it bit by bit, refer to the following: $ ohai | more
 "cpu": { "0": { "vendor id": "GenuineIntel", "family": "6", "model": "6", "model": "69", "model": "69", "model": "0", "core id": "0", "coresit": "1", 309 Maintaining Systems Using Chef and Puppet Alternatively, another solution is to redirect its content to
 a file so it's easier to process with a dedicated tool: $ ohai.json All of this information is also graphically available on the Chef interface when you select a node in the Attributes tab: 310 Chapter 8 Accessing Ohai information from a Chef recipe Now let's access this information from a recipe. We'll use the mysite cookbook from Chapter 6,
 Fundamentals of Managing Servers with Chef and Puppet, as the base cookbook to build our ServerSpec tests on, but obviously those tests can be used anywhere: $ cd cookbooks/mysite $ rm -rf test/* Test Kitchen works with the same name as the suite name, in an integration folder. Is this
Chef Server-specific code really working and adding the john user in the htpasswd file while restricting access to it? $ curl -w "" ' Note this URL carefully and copy paste it in the following cloud-config.yml file: #cloud-config # coreos: etcd2: discovery: " advertise-client-urls: "http://$private_ipv4:2379" initial-advertise-peer-urls:
"http://$private_ipv4:2380" listen-client-urls: listen-client-urls: listen-peer-urls: http://$private_ipv4:2380 units: - name: etcd2.service command: start - name: fleet.service command: start fleet cluster on CoreOS. To do this, we'll need to
complete two actions: ff Configure the Docker Hub to build and tag according to Git tags and not just branches ff Tag and push our release on Git, let's add a new type called Tag in the Build Settings tab. Reader feedback Freedback from our readers is always welcome
www.PacktPub.com eBooks, discount offers, and more Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? (ID: i-0f9106905e74a29f7) [...] + aws_s3_bucket.bucket: "iacbook" tags.Name: "IAC Book Bucket: "iacbook" tags.Name: "iacbook" tags.Name: "IAC Book Bucket: "iacbook" tags.Name: "IAC Book Bucket: "iacbook" tags.Name: "iacbook" tags.Nam
possible to encrypt sensitive data. Nice! Let's scale our service now, from 3 replicas to 100: # docker service ls ID NAME REPLICAS IMAGE aeq9lspl0mpg nginx 100/100 nginx command which is command to the first scale our service scale nginx=100 nginx scaled to 100 # docker service ls ID NAME REPLICAS IMAGE aeq9lspl0mpg nginx 100/100 nginx command to the first scale our service ls ID NAME REPLICAS IMAGE aeq9lspl0mpg nginx 100/100 nginx command to the first scale our service ls ID NAME REPLICAS IMAGE aeq9lspl0mpg nginx 100/100 nginx command to the first scale our service ls ID NAME REPLICAS IMAGE aeq9lspl0mpg nginx 100/100 nginx command to the first scale our service ls ID NAME REPLICAS IMAGE aeq9lspl0mpg nginx 100/100 nginx command to the first scale our service ls ID NAME REPLICAS IMAGE aeq9lspl0mpg nginx 100/100 nginx command to the first scale our service ls ID NAME REPLICAS IMAGE aeq9lspl0mpg nginx 100/100 nginx command to the first scale our service ls ID NAME REPLICAS IMAGE aeq9lspl0mpg nginx 100/100 nginx command to the first scale our service ls ID NAME REPLICAS IMAGE aeq9lspl0mpg nginx 100/100 nginx command to the first scale our service ls ID NAME REPLICAS IMAGE aeq9lspl0mpg nginx 100/100 nginx command to the first scale our service ls ID NAME REPLICAS IMAGE aeq9lspl0mpg nginx 100/100 nginx command to the first scale our service ls ID NAME REPLICAS IMAGE aeq9lspl0mpg nginx 100/100 nginx command to the first scale our service ls ID NAME REPLICAS IMAGE aeq9lspl0mpg nginx 100/100 nginx command to the first scale our service ls ID NAME REPLICAS IMAGE aeq9lspl0mpg nginx 100/100 nginx command to the first scale our service ls ID NAME REPLICAS IMAGE aeq9lspl0mpg nginx 100/100 nginx command to the first scale our service ls ID NAME REPLICAS IMAGE aeq9lspl0mpg nginx 100/100 nginx command to the first scale our service ls ID NAME REPLICAS IMAGE aeq9lspl0mpg nginx 100/100 nginx command to the first scale our service ls ID NAME REPLICAS IMAGE aeq9lspl0mpg nginx 100/100 nginx command to the first scale our service ls
 this ERB file in mysite/templates/htpasswd.erb, enter the following: : The .each method loops around the users variable that we passed through the template, iterates on user, and extracts our two values of interest: id and htpasswd. It does what it says. In most setups, I personally prefer to ensure all the systems are set to GMT, wherever they are on
the planet, GMT or not. Here are some examples of these styles and an explanation of their meaning. ff A profile is a class used to manage the underlying technology (For example, by installing Apache) ff In the main manifest, nodes are only using roles. Please contact us at with a link to the suspected pirated material. Following what
we've done so far, it would be helpful to know how to connect to the instance, what are the local and public IP addresses, or see the security groups used. Configuring NGINX as a reverse proxy is beyond the scope of this book, so just use the following simple configuration for the nginx.conf file at the root of your working folder: server { listen 80;
location / { proxy set header proxy pass } } X-Real-IP $remote addr; Host $http host; ; Change the configuration of the front container in Vagrant to use this configuration, remove the old index.html as we're not using it anymore, and link this container to the app container: configuration, remove the old index.html as we're not using it anymore, and link this container to the app container: configuration, remove the old index.html as we're not using it anymore, and link this container to use this configuration of the front was a set of the configuration of the front was a set of the configuration of the front was a set of the configuration of the front was a set of the configuration of the front was a set of the configuration of the front was a set of the configuration of the front was a set of the configuration of the front was a set of the configuration of the front was a set of the configuration of the front was a set of the configuration of the front was a set of the configuration of the front was a set of the configuration of the front was a set of the configuration of the front was a set of the configuration of the front was a set of the configuration of the front was a set of the configuration of the front was a set of the configuration of the configuration was a set of the configuration of the front was a set of the configuration of the configuration was a set of the configuration of the configuration of the configuration was a set of the configuration of the configuration was a set of the configuration of the configuration of the configuration was a set of the configuration of the configuration was a set of the configuration wa
 "docker" do |docker| docker.image = "nginx:stable" docker.ports = ['80:80'] docker.volumes = ["#{Dir.pwd}/nginx.conf:/etc/nginx/conf.d/ default.conf"] docker.link("app:app") end end 17 Vagrant Development Environments Linking the app container makes it available to the front container, so now there's no need to expose the Ghost blog container
directly, let's make it simpler and more secure behind the reverse proxy: config.vm.define "app" do |app| app.vm.provider "docker" do |docker| docker.image = "ghost:0.9.0" end end We're close! But this setup will eventually fail for a simple reason: our systems are too fast, and Vagrant parallelizes the startup of virtual machines
by default, and also does this for containers. Think about maintenance, backup, and performance before deploying your own. Our objective is to host our own private Docker registry, initially using Berkshelf Now navigate to
the mysite cookbook directory and use the upload feature from Berkshelf, so it will upload all necessary cookbooks at once: $ berks upload With Berkshelf, dependencies of the dependencies of the dependencies are uploaded as well! Testing MySQL deployment Run the chef-client on the node, and when the process is done, ensure we can connect to the local MySQL
server using the supplied password: $ mysql -h 127.0.0.1 -uroot -psuper secure password -e "show databases;" +------+ | Database | +------
 ----+ There's more... With Puppet, there is also a lot of code ready to use. He is currently an architect and a DevOps
team leader for Reservit, an online hotel booking engine. The releases folder contains all the stored releases. As we're using a CentOS 7 server, the Apache2 HTTP server package name is httpd, (it's apache2 on Debian-based systems). By default, Docker Security Scanning is not activated, so we have to navigate to the billing tab of the account's
interface and tick it to enable it: 393 Maintaining Docker Containers From now on, when a new Docker image is created or pushed, the system will scan it quickly and forgotten there until the next rebuild (which may not happen anytime
soon). All boxes are stored by default in ~/.vagrant.d/boxes. Overall, those changes proposed by the linters helped us a lot in building a much better and stronger container. [...] [ ~]$ head -n1 /srv/blog/config.js // # Ghost Configuration You or your coworker are now remotely logged into your own Vagrant box over the Internet!
Simulating Chef upgrades using Vagrant Wouldn't it be awesome to simulate production changes quickly? Let's see what we can do about it by building an Apache Docker container on Ubuntu 16.04. The application can connect to the database tier, and will be available to the end user through tier 1 proxy servers. 216 Chapter 6 Managing a simple
static file Let's begin by creating a basic PHP file that will only display the phpinfo() result. This way, only the nodes allowed to access the data can decrypt it—each with their own private key—ensuring no clear-text shared keys are being sent, like with the classic encrypted data bag scheme. At Packt, quality is at the heart of our editorial process.
The token can be obtained with the following command: $ docker swarm join-token worker -q We'll store this token in a simple file in our infrastructure repository (worker.token), so we can access it and version it. We want to store the provider (aws), the region (eu-west-1), and the name of the cluster (mycluster). We'll build a fully automated and
scalable Docker Swarm cluster, so we can operate highly scalable and performant workloads on bare metal: this setup can scale thousands of containers in just a few minutes. For reference, as this will all be discussed in the next section, all tests are located in the test/integration// folder. @puppetlabs mysql 243 Fundamentals of Managing Servers
with Chef and Puppet example 42-mysql Puppet module... Let's start by creating default = "xoxb-1234-5678" description = "Slack Token" { default = "us" description = "Heroku region" } variable "start by creating default end use in variable "start by creating default = "xoxb-1234-5678" description = "Slack Token" } 154 Chapter 4 Now we can create our first Heroku app with its
variables using the heroku app resource, in heroku app
do it... Begin by installing the plugin: $ vagrant plugin install vagrant-aws A requirement of this plugin is the presence of a dummy Vagrant box that does nothing: $ vagrant box add dummy Remember how we configured the Docker provider in the previous recipes? We'll test for services, files, directories, yum repositories, packages, ports, and
 injected content. There's more... Using Puppet, services are also managed with a dedicated resource directive. So, to make it work as intended, we'll want to execute this container like this: $ docker run -e USER=john -e PASSWORD=s3cur3 sshd Take this simple Dockerfile that creates what's necessary to run the Dropbear SSH server on the Alpine
 Docker image: FROM alpine:latest RUN apk add --update openssh-sftp-server openssh-client dropbear &&\ rm -rf /var/cache/apk/* RUN mkdir /etc/dropbear &&\ rm -rf /var/cache/apk/* RUN mkdir /etc/dropbear &&\ touch /var/log/lastlog COPY entrypoint.sh | ENTRYPOINT ["/entrypoint.sh"] CMD ["dropbear", "-RFEmwg", "-p", "22"] When built, this container will start by executing the entrypoint.sh
script, and then the dropbear binary. See also ff The hiera-eyaml GitHub repository with its documentation at . Generating API credentials | Service Account Key. If you don't already have it, you will need to install it manually
using vagrant plugin install vagrant-hostmanager. When you're done with your Vagrant VM, you can simply destroy it: $ vagrant destroy ==> default: Forcing shutdown of VM... The minimum we can do is boot a fresh CentOS 7.2: Vagrant.configure("2") do |config| config.vm.box = "bento/centos-7.2" end We'd like to automatically install Chef on our
temporary node, so let's use the vagrantomnibus pluqin (remember, installing it is easy: vagrant pluqin install vagrantomnibus). The policy is also to remove the default ubuntu account. This backend is eyaml, and by default, we will look for files with the .eyaml extension. Log in or register to our website using your e-mail address and password.
Upload the initial starter cookbook, still using the knife command: $ knife upload cookbooks/starter Created cookbooks/starter There's more... There's no hosted Puppet server offering. We can submit this change for review. This program is executed during each Chef run and stores all of the gathered information in the Chef database to make it
available right from the cookbooks. As Berkshelf works per cookbook, we'll declare all our cookbook dependencies to run this particular cookbook here, which in our case, happens to be currently all local. For security reasons, it is a good idea to restrict access to the private key: $ chmod 500 keys $ chmod 400 keys/private key.pkcs7.pem 4. Add the
default Vagrant user to the new Docker group. Each branch from the Git repository will be checked out into a distinct directory and will be available as an environment. 184 Chapter 5 Warning: The current cloud-init version shipped with Ubuntu 16.04 LTS and CentOS 7 is having issues installing Chef. Terraform has a great internal mechanism that
allows us to plan for changes by comparing what our infrastructure code wants and what the remote state includes. Now you can simply change the value of vm num and you'll easily spawn new virtual machines in seconds. 6 Fundamentals of Managing Servers with Chef and Puppet In this chapter, we will cover the following recipes: ff Getting
started (notions and tools) ff Installing the Chef Development kit and Puppet collections ff Creating a free hosted server Chef account and a Puppet services ff Managing files, directories, and templates ff Handling dependencies ff More dynamic
code using notifications ff Centrally sharing data using a Chef data bag and Hiera with Puppet ff Creating functional roles ff Managing Servers with Chef and Puppet Introduction Chef is an open source tool used to automate the configuration of systems and it integrates
well with most IaaS such as Amazon Web Services, OpenStack, or Google Cloud. Then, a hierarchy is defined. To do that, I suggest using a Terraform container, so we'll use here the one I use myself: sjourdan/terraform: (from . Getting ready To step through this recipe, you will need: ff Access to a cloud-config enabled infrastructure How to do it...
We'll launch three commands during boot. There's more... We can perform the same dynamic access to a map using the lookup() function in Terraform Storing and accessing files easily and in a scalable way is an essential part of a modern infrastructure. These boxes
are just packaged virtual machines images that are available, for example, from or you can alternatively build your own using various tools. Let's move on. All examples are self-contained real-life little projects. However, to achieve this peace of mind, we need more than a Ruby-on-Rails application, for example, we need to ship a Dockerfile containing
everything to build a self-sufficient container so anyone can run it. If you care about consistency and repeatability, an option is to add the following Ruby check at the beginning of your Vagrant-vbguest vagrant-omnibus).each do |plugin| unless Vagrant-has_plugin?(plugin) raise "#{plugin} plugin is not installed! Please install it using
 `vagrant plugin install #{plugin}`" end end This code snippet will simply iterate over each plugin name to verify that Vagrant returns them as installed. describe file ('/var/www/docroot/.htaccess') do its(:content) { should match /AuthUserFile Vetc\apache2\/ 284 Chapter 7 htpasswd/} end end Now, let's try to run Beaker again: $ rake beaker ...
 Beaker::Hypervisor, found some vagrant boxes to create Bringing machine 'ubuntu-1604-x64' up with 'virtualbox' provider... com/downloads.html. How to do it... We'll study and follow suggestions of the two complimentary tools—Cookstyle and Foodcritic. io/ is an example, Docker's own is another), or it can be selfhosted (for privacy, speed,
bandwidth issues, or company policy). However, things might change, default tools might be removed - better to be safe than sorry. Its releases are tagged by release (such as golang:1.7-windowsservercore and golang:1.7nanoserver)
Chapter 10, Maintaining Docker Containers, is showing more advanced Docker usage for developers and engineers, such as code testing, automated build pipelines and Continuous Integration, automated vulnerability scanning, monitoring, and debugging. If the node can decrypt it, like we did previously with traditional data bags, the data will be
available for use: template "/etc/aws/credentials" do source 'aws.erb' owner 'root' group 'root' mode '0600' variables(aws region: aws['id'], aws access key: aws['aws access ke
aws_access_key_id = an_access_key aws_secret_access_key = a_secret_key Using Chef Vault, no shared key has ever transited in clear text, and only filtered and existing nodes can decrypt data that has been encrypted specifically for them. We have to make sure another folder hosts the encrypted versions (maybe you already have one to store them
on GitHub for example). The Terraform version in use for this book is 0.7.2. Configuring the Terraform with many IaaS providers, such as Google compute target pool "www" { name = "www-pool" instances =
["${google_compute_instance.www.*.self_link}"] health_checks = ["${google_compute_http_health_checks.www.name}"] } The self_link attribute returns the URI of the resource. Everything related to Vagrant configuration here end 3 Vagrant
 Development Environments Getting ready To step through this recipe, you will need the following: ff A working VirtualBox installation ff An Internet connection How to do it... 1. Let's define the requirements of simple target environments that we'll translate into dynamic Terraform code: Parameter Staging Production
 Number of instances 1 t2.micro 3 t2.medium AMI in eu-west-1 CoreOS Stable ami-85097ff6 CoreOS Stable ami-85097ff6 AMI in us-east-1 ami-0aef8e1d ami-0aef8e1d ami-0aef8e1d S3 bucket naming iacbook-staging iacbook-production Default environment Yes No Type of instance Operating system Let's start by declaring those variables in the variables.tf file
exactly as we' saw in Chapter 2, Provisioning laaS with Terraform, except we'll describe environments such as staging and production instead of the AWS regions for the cluster size and instance types. We want to run our app in Europe and we want Hubot to connect to Slack, so we need to provide a Slack token as well. For existing values, eyaml will
add an index called num to DEC()::PKCS7[value]! blocks. There's more... With Puppet, the same logic applies. Let's work on that later, and add the change to a new file named plan.tfvars: aws instance type="t2.medium" 101 Going Further with Terraform We know we'd like to propose to change the number of instances to 2, so let's add that to the
same file: aws instance type="t2.medium" (forces tags.Name: "coreos 1" => "t2.medium" (forces tags.Name: "coreos 1" => "coreos 1" [...] + aws instance.coreos.1
instance type: "t2.medium" tags.Name: "coreos 2" [...] Plan: 2 to add, 0 to change, 1 to destroy. BATS can also grow in complexity as per your requirement. We'd like to use a data bag, as it can be encrypted: 1. Have a look: $ terraform plan [...] + aws instance.coreos [...] + aws key pair.admin key [...] Plan: 2 to add, 0 to change, 0 to destroy. The
multi-platform client is written in Ruby, while its server counterpart is written in Erlang. Before we start to code, let's begin by setting up our environment. It's very easy with Terraform, just use the count option inside the aws instances: count = 2 Next, terraform apply this and observe
Terraform automatically creating a new machine according to the counter: $ terraform apply aws key pair.admin key: Refreshing state... Here's how it works with creating a log stream named tomcat under the same log group in docker-compose. Add the following to the apache spec.rb file: describe port('80') do it { should be listening } end
Similarly, add the following for MySQL in the mysql_spec.rb file: describe port('3306') do it { should be_listening } end Testing for files existence and content We previously unit tested the intention to create all those files in our cookbooks, such as a VirtualHost with a custom name, impacting both filename and content (that's what the mysite
cookbook from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, does, override the defaults from the custom apache cookbook). By default, it isn't. All recipes are based on Chef. Amazon CloudWatch Logs Docker logging driver You can now use this group to create a log stream from an application or a container. Here's how it
looks: ingress { from port to port protocol cidr blocks } = = = 2 2 2 "tcp" ["1.2.3.4/32"] egress { from port to port protocol cidr blocks } = = = 0 0 "-1" ["0.0.0.0/0"] You can add a name tag for easier reference later: tags { Name = "base security group" } Apply this and you're good to go: $ terraform apply
aws_security_group.base_security_group: Creating... To simply send the encrypted version of the data bag, let's use the encryption feature of the knife command: $ knife data bag from file --encrypt --secret s3cr3t aws us-east-1.json Updated data_bag_item[aws::us-east-1] If we request the data without providing a decryption key, we'll get the
encrypted data from the Chef server: $ knife data bag show aws us-east-1 WARNING: Encrypted data bag detected, but no secret provided for decoding. A clean, reliable, and repeatable way is to use a Vagrant plugin to do just that—vagrant-omnibus. Just as easy as usual: --- name: create RDS PgSOL rds; command: create instance name: db
db engine: postgres engine version: 9.5.2 db name: iac book db size: 5 instance type: db.t2.micro username: dbadmin password: super secure password publicly accessible: yes tags: Name: IAC Database After executing this playbook, a similar PostgreSQL server will run on RDS as we just did with Terraform. blade.php file and reload your browser
to see the change reflected. Running Chef Chef can be used in multiple ways, the most important are the following: ff ff 192 Client/server mode: An agent runs on every managed client, regularly getting updates from the server, and observe that runcmd didn't
run again: $ grep "started at" /var/log/cloud-init-output.log bootcmd started at Fri Sep 23 07:04:31 UTC 2016 Now we know what directive to use in each case. Here is the Puppetfile: forge ' ' # Local modules mod 'php', :local =>true mod 'php', :l
'puppet/staging', '2.0.1' mod 'puppetlabs/mysql', '3.9.0' mod 'puppetlabs/stdlib', '4.13.1' Now we need to run r10k to install $\text{ls modules/ apache/ mariadb/ mysql/ php/ $\text{stdlib'}, the main manifest to use the official MySQL package; we need to remove the
reference to our MariaDB module, and use the class provided by the official MySQL package: node 'web.pomes.pro' { $website=$fqdn; $docroot="/var/www/$fqdn"; $users=hiera('webusers'); class { 'mysql::server': root password => 'super secure password', } apache::vhost{$website: 246 Chapter 6 website => $website => $website, }
docroot => $docroot, } apache::htpasswd('htpasswd', docroot => $docroot, } file { $docroot: ensure => directory, owner => 'www-data', group => 'www-data', mode => '0755', } file
{"$docroot/index.php": ensure => present, owner => 'www-data', group => 'www-data', mode => '0644', content => "", } Let's start a fresh Vagrant setup. How it works... This section usually consists of a detailed explanation of what happened in the previous section. 26 Chapter 1 First, it creates a connection to the Redis server on the db lan
network: #!/usr/bin/env node var os = require("os"); var redis = require('redis'); var redis = r
require('http'); http.createServer(function (req, res) { res.writeHead(200, {'Content-Type': 'text/plain'}); res.end('Running on '+os.hostname()+''); }).listen(8080); console.log('HTTP server listening on '+os.hostname()+'
After=network.target [Service] ExecStart=/srv/nodeapp/app.js Restart=always User=vagrant Environment=PATH=/usr/bin Environment=PATH=/usr/bin Environment=PATH=/usr/bin Environment=NODE ENV=production WorkingDirectory=/srv/nodeapp/app.js Restart=always User=vagrant Environment=PATH=/usr/bin Environment=PATH=/usr/bin Environment=NODE ENV=production WorkingDirectory=/srv/nodeapp/app.js Restart=always User=vagrant Environment=NODE Environment=
application servers (in this case: two) to serve the app. A virtual machine at Digital Ocean is named a droplet, so the resource to launch any command at boot time that will be run as early as possible, every time the machine boots, simply add it to the list of the bootcmd directive: #cloud-config bootcmd: -
echo bootcmd started at $(date) - echo $INSTANCE_ID > /etc/instance id If we delete or modify this file, at the next reboot it will be overwritten. Let's add it as localhost (or adapt to your local hostname), along with the better password and the WordPress image version: wordpress: image: wordpress: 4.5 environment: WORDPRESS_DB_PASSWORD:
sup3rs3cur3 VIRTUAL HOST: localhost Make the password match in the MySQL section as well and we'll be done with our production.yml up Confirm nginx-proxy is answering in HTTP/80 and forwarding a proper HTTP answer from the WordPress
container: $ curl -IL HTTP/1.1 302 Found Server: nginx/1.11.3 [...] HTTP/1.1 200 OK Server: nginx/1.11.3 362 Chapter 9 We've seen how, with only a few lines of YAML, we can easily orchestrate containers, how it can be used to handle different cases and environments, and how it can also be successfully extended. It's also scriptable and can be fed
with recorded real traffic packet captures for offline analysis. You can download a Vagrant installer for your platform from . The same way every program has language requirements, ours need to be able to build Docker containers and execute some Ruby code. See root's crontab file: $ sudo crontab -1 # Chef Name: chef-client 0 0,4,8,12,16,20 * * *
/bin/sleep 69; 2>&1 /usr/bin/chef-client > /dev/null By default, it's executing chef-client every four hours with, in this case, a 69 seconds delay to avoid every node from hammering the Chef server at the same time. 394 Chapter 10 See also ff CoreOS Clair at Sending Docker logs to AWS CloudWatch logs When we run dozens or hundreds of containers
in production, hopefully on a clustered container platform, it soon becomes difficult and tedious to read, search, and process logs—just like it was before when containers with puppet-lint based on the latest recipe for our Apache module: $ puppet-lint pupper.
modules/apache/manifests/init.pp WARNING: class not documented on line 1 ERROR: two-space soft tabs not used on line 3 ... 10. 313 Maintaining Systems Using Chef and Puppet Getting ready To step through this recipe, you will need: ff A working Chef and Puppet Getting ready To step through this recipe, you will need: ff A working Chef and Puppet Getting ready To step through this recipe, you will need: ff A working Chef and Puppet Getting ready To step through this recipe, you will need: ff A working Chef and Puppet Getting ready To step through this recipe, you will need: ff A working Chef and Puppet Getting ready To step through this recipe, you will need: ff A working Chef and Puppet Getting ready To step through this recipe, you will need: ff A working Chef and Puppet Getting ready To step through this recipe, you will need: ff A working Chef and Puppet Getting ready To step through this recipe, you will need: ff A working Chef and Puppet Getting ready To step through this recipe, you will need: ff A working Chef and Puppet Getting ready To step through this recipe, you will need: ff A working Chef and Puppet Getting ready To step through this recipe, you will need: ff A working Chef and Puppet Getting ready To step through the first recipe and t
Chef code (optionally) from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code How to do it... As we're going to deploy an application for MySite (maybe an engineering blog for the MySite company), let's call this recipe
mysite::deploy. We have to specify where is everything placed (cookbooks, environments, roles, and so on) with the added subtlety of a nodes folder that will be left empty in our case. To create a new Docker network, just give it a name: $ docker network d01a3784dec1ade72b813d87c1e6fff14dc1b55fdf6067d6ed8dbe42a3af96c2
Grab some information about this network using the docker network inspect command: $ docker network inspect hello_network -f '{{json .IPAM.Config }}' [{"Subnet":"172.18.0.0/16", "Gateway":"172.18.0.0/16", "Gateway:"172.18.0.0/16", "Gateway:"172.18.0.0/16", "Gateway:"172.18.0.0/16", "Gateway:"172.18.0.0/16", "Gateway:"172.18.0.0/16", "Gateway:"172.18.0.0/16", "Gateway:"172.18.0.0/16", "Gateway:"172.18.0.0/16", "Gateway:"172.18.0.0/16", "Gateway:"172.
what we thought was a minor modification in our code has in fact a destructive impact (sometimes, some parameters in a resource!). Upload them all now: $ berks upload 5. It contains a lot of information: ff How to test the system (Vagrant, by default) ff How to provision the system (chef-solo, chef-zero, or
other modes) ff Which platforms to test (Ubuntu 16.04, CentOS 7.2, or other distributions) ff The test suites (what to apply, where to find information, in what context, and similar information) Configuring Test Kitchen Irrespective of whether we already have a .kitchen.yml file or not, let's open it and fill in the following details: ff We want to run the
tests with Vagrant to closely simulate a VM in production ff We want to provision using Chef Zero (by simulating a Chef server locally) ff We want to test only on CentOS 7.2 (our code isn't currently designed to run on something else) ff We want to test only on CentOS 7.2 (our code isn't currently designed to run on something else) ff We want to test only on CentOS 7.2 (our code isn't currently designed to run on something else) ff We want to test only on CentOS 7.2 (our code isn't currently designed to run on something else) ff We want to test only on CentOS 7.2 (our code isn't currently designed to run on something else) ff We want to test only on CentOS 7.2 (our code isn't currently designed to run on something else) ff We want to test only on CentOS 7.2 (our code isn't currently designed to run on something else) ff We want to test only on CentOS 7.2 (our code isn't currently designed to run on something else) ff We want to test only on CentOS 7.2 (our code isn't currently designed to run on something else) ff We want to test only on CentOS 7.2 (our code isn't currently designed to run on something else) ff We want to test only on CentOS 7.2 (our code isn't currently designed to run on something else) ff We want to test only on CentOS 7.2 (our code isn't currently designed to run on something else) ff We want to test only on CentOS 7.2 (our code isn't currently designed to run on something else) ff We want to test only on CentOS 7.2 (our code isn't currently designed to run on something else) ff We want to test only on CentOS 7.2 (our code isn't currently designed to run on something else) ff We want to test only on CentOS 7.2 (our code isn't currently designed to run on something else) ff We want to test only on CentOS 7.2 (our code isn't currently designed to run on something else) ff We want to test only on CentOS 7.2 (our code isn't currently designed to run on something else) ff We want to test only on CentOS 7.2 (our code isn't currently designed to run on something else) ff We want to test on the cur
our .kitchen.yml file looks for the mysite cookbook: --driver: name: chef zero 272 Chapter 7 platforms: - name: default data bags path: ".../.data bags" run list: - recipe[mysite::default] attributes: Testing with Test Kitchen To simply launch Test Kitchen with the specified configuration, execute the
following command: $ kitchen test ----> Testing ----> Creating ... This time we'll explicitly set the EDT (New York) time zone for our server (even if the server is running in Europe or somewhere else). We want to update the manifest at the top of the file in order to do the following: ff Install apache ff Define a virtual host ff Create the root directory of
the virtual host ff Create a htpasswd file with a test user ff Create a htpasswd file with a test user ff Verify the code is idempotent ff Verify apache is running and activated at boot ff Verify apache is listening ff Verify the virtual host is deployed and activated
with the correct DocumentRoot ff Verify the htpasswd file is deployed with a correct content file i
'/var/www/docroot', } apache::htpasswd{'htpasswd': filepath => '/etc/apache2/htpasswd', users => [{ "id" => "user1", "htpasswd" => "hash1" }], } 283 Testing and Writing Better Infrastructure Code with Chef and Puppet file { '/var/www/docroot': ensure => directory, owner => 'www-data', group => 'www-data', mode => '0755', }
apache::htaccess{'myhtaccess': filepath => '/etc/apache2/htpasswd', docroot => '/var/www/docroot', } EOS apply manifest(pp, :catch failures => true) apply
the top-level directory of our Apache module with the following content: require 'puppetlabs spec helper/rake tasks' All unit tests should remain in a spec directory. If we just apply the documentation shipped with the following docker-compose.yml at the root of some new directory (it can be a Git
repository if needed): version: '2' services: wordpress: image: wordpress: image: wordpress ports: - 8080:80 environment: WORDPRESS_DB_PASSWORD: example mysql: image: mariadb environment: WORDPRESS_DB_PASSWORD: example mysql: image: mariadb environment: MYSQL_ROOT_PASSWORD: example mysql: image: mariadb environment: WORDPRESS_DB_PASSWORD: example mysql: image: mariadb environment: MYSQL_ROOT_PASSWORD: example mysql: image: mariadb environment: mysql: image: mysql: ima
downloaded, local HTTP port 80 gets redirected on port 8080 on the host, and MySQL stays isolated. It's perfectly correct, and if you build it, the image size is around 260 MB: FROM ubuntu:16.04 RUN apt-get update -y RUN apt-get install -y apache2 RUN rm -rf /var/lib/apt ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"] Now, each
layer is added on top of the previous. Here's how it looks for our user emergency; users: - name: emergency groups: sudo shell: /bin/sh sudo: ['ALL=(ALL) NOPASSWD:ALL'] ssh-authorized-keys: - ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC+fAfzjw5+mUZ7nGokB0tzO9fOLKrjHGVlabpRUxvs
IN/dRRmiBA9NDh5YRZ/ThAhn+RvPKGTBrXmuv3qWd/iWc3nie0fc2zDX1/Dc8EAIF9ybXfSxT 2DXOWWLOvNdUVOZNifmsmCQ1z0p9hg3bo65c0ZEBpXHIk+l75uFWAIYZ/4jnXyFWz1ptmQR7gnAk2KBK19sj1Ii0pNjGyVbl5bNitWb3ulaviIT3FCswZoOsYvc LpOwQrMA3k12kEAb30CYpesGcq6WDHAZSpWkFvc3Cd/AET4/
SjtyYpQVEhUn84v106WbNeDyJpUX6cz2WG2UaEqZc0VqZVhI63jG7wUR Once logged in as emergency sudo $ echo $SHELL /bin/sh $ sudo whoami root 173 Provisioning the Last Mile with Cloud-Init We never
explicitly asked to remove the default ubuntu user account: it's automatic as soon as we create an initial user. Finally, let's explicitly define the installer full URL, so we're sure about what we install (maybe point it to a local version on your own servers). It takes the strong testing culture from the development world and applies it to an infrastructure-
as-code environment. There's more... If you're interested in creating your own base Vagrant boxes, refer to Packer (. We'll also have an introduction on easily managing those dependencies, that will give an insight of how to deal with more complex infrastructures managed by Chef. STAT START TIME Ssl+ 23:33 0:00 Chapter 9 We're now building
tougher containers! Orchestrating with Docker Compose Launching multiple containers manually can be a hassle, especially when the infrastructure How to do it... However, the information contained in this book is
```

sold without warranty, either express or implied. If we did choose to run only nodes with a call to the mysite: default recipe, it will also work aws key pair.admin key: Refreshing state... However, our templates are placed under mysite; templates by the chef workflow command. Now, either express or implied. If we did choose to run only nodes with a call to the mysite; templates by the chef workflow command. Now, either express or implied. If we did choose to run only nodes with a call to the mysite; templates are placed under mysite.

```
this role (role[mysite]), or if you're about to bootstrap the server; adding the -r "role[mysite]" option will bootstrap Chef on the node as well as execute Chef with this run list: $knife bootstrap the server; adding the -r "role[mysite]" option will bootstrap the server; adding the -r "role[mysite]" option will bootstrap the server; adding the -r "role[mysite]" option will bootstrap the server; adding the -r "role[mysite]" option will bootstrap the server; adding the -r "role[mysite]" option will bootstrap the server; adding the -r "role[mysite]" option will bootstrap the server; adding the -r "role[mysite]" option will bootstrap the server; adding the -r "role[mysite]" option will bootstrap the server; adding the -r "role[mysite]" option will bootstrap the server; adding the -r "role[mysite]" option will bootstrap the server; adding the -r "role[mysite]" option will bootstrap the server; adding the -r "role[mysite]" option will bootstrap the server; adding the -r "role[mysite]" option will bootstrap the server; adding the -r "role[mysite]" option will bootstrap the server; adding the -r "role[mysite]" option will be obtained the server; adding the -r "role[mysite]" option will be obtained the server; adding the -r "role[mysite]" option will be obtained the server; adding the -r "role[mysite]" option will be obtained the server; adding the -r "role[mysite]" option will be obtained the server; adding the -r "role[mysite]" option will be obtained the server; adding the -r "role[mysite]" option will be obtained the server; adding the -r "role[mysite]" option will be obtained the server; adding the -r "role[mysite]" option will be obtained the server; adding the -r "role[mysite]" option will be obtained the server; adding the -r "role[mysite]" option will be obtained the server; adding the -r "role[mysite]" option will be obtained the server; adding the -r "role[mysite]" option will be obtained the server; adding the -r "role[mysite]" option will be server the server the server the server the server the server th
does not provide a role feature. Let's write the content of our data bag for our user John, in data bags/webusers/john.json: { "id": "john", "htpasswd": "$apr1$AUI2Y5pj$0v0PaSlLfc6QxZx1Vx5Se." } 4. Let's use it inside a keys.tf file, and paste the public key content: resource "aws key pair" "admin key" { key name = "admin key" 51 Provisioning IaaS
with Terraform public key = "ssh-rsa AAAAB3[...]" } This will simply upload your public key to your AWS account under the name admin key: $\frac{172}{2}$ Access to a cloud-config enabled
infrastructure Chapter 5 How to do it... To create a group, we use a directive simply named groups, taking a list of groups. Conventions In this book, you will find a number of text styles that distinguish between different kinds of information. Finished in 0.00081 seconds (files took 0.14125 seconds to load) 0 examples, 0 failures No acceptance test
has been defined yet, but we will see how to write one in the next pages. She needs to be in the docker group, so she can manipulate the containers, and she gave us her SSH public key. Test Kitchen helps start an isolated system environment, apply Chef cookbooks to it, and then execute tests. To configure flannel before starting it, we can add more
configuration information to the cloud-config file. Installing and configure it to listen on the LAN port instead of 127.0.0.1, and start it): config.vm.define "db-1" do |config| config.vm.hostname = "db-1" config.vm.network "private_network", ip:
"10.30.0.11", virtualbox intnet: "db lan" config.vm.provision: shell, :inline => "sudo yum install -q -y redis" config.vm.provision: shell, :inline => "sudo systemctl
enable redis" config.vm.provision :shell, :inline => "sudo systemctl start redis" end Tier 2: the application (web) server. We need to install puppet-lint using the following command: $ sudo puppet resource package puppet gem If you are already familian
with Puppet, you probably saw that the code we wrote in the previous chapter does not conform to standards. Alternatively, to just build the image, use this code: $ docker-compose build Creating BATS tests We'll now test two of the main actions this image is supposed to do: ff Install Apache 2.4.10 ff Clean the APT cache Start by creating a test
folder at the root of our repository that will host the BATS tests: $ mkdir test Our first test is to verify that the initialization phases of booting a cloud instance (new or not): installing packages, copying files or SSH keys, deploying the initialization phases of booting a cloud instance (new or not): installing packages, copying files or SSH keys, deploying the initialization phases of booting a cloud instance (new or not): installing packages, copying files or SSH keys, deploying the initialization phases of booting a cloud instance (new or not): installing packages, copying files or SSH keys, deploying the initialization phases of booting a cloud instance (new or not): installing packages, copying files or SSH keys, deploying the initialization phases of booting a cloud instance (new or not): installing packages, copying files or SSH keys, deploying the initialization phases of booting a cloud instance (new or not): installing packages, copying files or SSH keys, deploying the initialization phases of booting a cloud instance (new or not): installing packages, copying files or SSH keys, deploying the initialization phases of booting a cloud instance (new or not): installing packages, copying files or SSH keys, deploying the initialization phases of booting a cloud instance (new or not): installing packages, copying files or SSH keys, deploying the initialization phases of booting a cloud instance (new or not): installing packages (new o
Chef, defining repositories, or rebooting (when done). This file will have its content declared right from the cloud-config file: #cloud-config file: #cloud-config write files: - path: /etc/motd content: | This server is configured using cloud-init. ff How should etcd listen for client traffic? We used our first file interpolation, using a variable, to show how to use a more dynamic
code for our infrastructure. These systems can integrate very well with services such as GitHub or GitLab and launch tests automatically after a push or pull request. We won't find the usual /etc/timezone on CoreOS, so the default timezone directive from cloud-init we've seen earlier in this book won't work. Refer to the cloud-init part of this book in
Chapter 5, Provisioning the Last Mile with Cloud-Init for more information on the cloud-config.yml file, and especially on configuring CoreOS with it. Starting from a Node JS image The official repository for the Node Docker image includes many tagged versions with many base images: node:7 is based on Debian Jessie, while node:7-alpine is based on
Alpine 3.4. node:7-slim will be based on the slim Debian Jessie, and there's even node:7-wheezy if we feel like running Node 7 on Debian Wheezy. No one can test each and every possibility and regression and special cases from months or maybe years ago. How it works... First, Terraform creates the required AWS environment (keys, security groups
and instances), and once the instance is running, it connects to it with the right credentials by SSH, then deploys the specified Chef client version from the official source, and finally executes an initial chef-client run that registers the node on the Chef server and applies the requested cookbooks. Let's put our own index.html in there. For example,
we'll see a database role way more often than a MySQL role. Scanning for vulnerabilities with Quay.io and Docker Cloud One major issue when working with containers is their deprecation and maintenance costs. It's entirely up to the organization to model the infrastructure according to its needs, and the complexity can grow very quickly. We can
now create an AWS Access Key for our user Mary, using the aws_iam_access_key resource with reference to our user: resource "aws_iam_access_key" "mary" { user = "${aws_iam_access_key resource with reference to our user: resource with reference to our user the AmazonS3FullAccess managed policy, let's use the dedicated resource: resource resource with reference to our user.
 "aws iam user policy attachment" "mary s3full" { user = "${aws iam user.mary.name}" policy arn = "arn:aws:policy/AmazonS3FullAccess" } Let's write an output so we know both parts of the key in outputs.tf: output "mary" { value = "ACCESS KEY: ${aws iam access key.mary.id}, SECRET: ${aws iam access key.mary.id}, SECRET: ${aws iam access key.mary.secret}" } Also,
terraform apply this to create the mary user: [...] Outputs: mary = ACCESS_KEY: AKIAJPQB7HBK2KLAARRQ, SECRET: wB+Trao2R8qTJ36IEE64G NIGTqeWrpMwid69Etna Testing the restrictions Now, terraform apply this, and confirm using an S3 browser that you can access S3! Here's an example of creating a simple S3 bucket with s3cmd: $$
s3cmd --access key = --secret key = mb s3://iacbook-iam-bucket | statuscake | start by configuring the provider |
variables.tf: variable "statuscake username" { default = "changeme" description = "StatusCake Account Username" } variable "statuscake apikey" { default = "changeme" description = "StatusCake Account API Key" } Also, override with your own values in terraform.tfvars. (initial-advertisepeer-urls). Storing updated cookbooks/starter/metadata.rb in
the cache. $ knife bootstrap 1.2.3.4 -N my_node_hostname -x vagrant --sudo -r "starter" [...] 192.168.146.129 * log[Welcome to Chef, Sam Doe!] action write There's more... Using Puppet, we need to install the Puppet agent, once our node is
created. Using ServerSpec, we can start a container using the id attribute of the image we built earlier. Navigate to and you'll see the default Laravel welcome screen. At Packt, we take the protection of our copyright and licenses very seriously. We know the AMI ID, we want a single t2.micro host. The httpd service is coming from the default Apache
recipe, so it's better to include it right now in the apache/recipes/virtualhost.rb file, so we're sure this particular recipe works alone and not by side-effect of a previous inclusion: 226 Chapter 6 include recipe works alone and not by side-effect of a previous inclusion: 226 Chapter 6 include recipe works alone and not by side-effect of a previous inclusion: 226 Chapter 6 include recipe works alone and not by side-effect of a previous inclusion: 226 Chapter 6 include recipe works alone and not by side-effect of a previous inclusion: 226 Chapter 6 include recipe works alone and not by side-effect of a previous inclusion: 226 Chapter 6 include recipe works alone and not by side-effect of a previous inclusion: 226 Chapter 6 include recipe works alone and not by side-effect of a previous inclusion: 226 Chapter 6 include recipe works alone and not by side-effect of a previous inclusion: 226 Chapter 6 include recipe works alone and not by side-effect of a previous inclusion inclusion
with this, refer to the deploy resource documentation to discover everything this resource has to offer. Getting ready To step through this recipe, you will need: ff A working Docker installation (it's available for all major Linux distributions and Mac OS) BATS Version 0.4.0 is used in this chapter. Instead of going through the web
application and enabling add-ons, let's instead use the heroku addon resource. It's important to know whether we'd be able to fully rebootstrap these systems in case of a disaster; if yes, how long it would take. [...] If you'd prefer to dive right in, the default recipe can be found at: recipes/default.rb Now we need to tell Chef to install a package using
the package resource. Apache Puppet code localhost $ scp /var/folders/k9/7sp85p796qx7c22btk7 tgym0000gn/T/ beaker20161101-75828-1of1g5j ubuntu-1604-x64:/tmp/apply manifest. 165 Provisioning the Last Mile with Cloud-Init The scope of action of cloud-init is really intended for the initialization phase; it's not a configuration management tool
and is globally not meant to be run again afterwards to update configuration, like one would with Ansible or Chef. packtpub.com/support and register to have the files e-mailed directly to you. Now we want to join the Docker Swarm cluster. In this chapter, we'll go through all the most important notions, so it can also serve as a quick cheat sheet or
reminder. The WordPress container takes one environment variable in this case—the MySQL root password, which should match the environment variable from MySQL. Who this book is for DevOps engineers and developers working in cross-functional teams or operations and would like to switch to IAC to manage complex
infrastructures. Alpine Linux is this small image (~5 MB) that can greatly help us if we'd like to access a shell (/bin/sh is available) and a package manager to debug our application. Our environment will be interactive (use -i) and we want a pseudo-terminal to be allocated (use -t). CoreOS proposes a very useful cloud-config file validator at . Help us
be relentless in improving our products! Table of Contents Preface v Chapter 1: Vagrant Development Environments 1 Introduction 2 Adding an Ubuntu Xenial (16.04 LTS) Vagrant box 2 Using a disposable CentOS 7.x with VMware in seconds 3
Extending the VMware VM capabilities 8 Enabling multiprovider Vagrant for a Ghost blog behind NGINX 16 Using Docker with Vagrant remotely with AWS EC2 and Docker 19 Simulating dynamic multiple host networking 22 Simulating a networked three-tier
 architecture app with Vagrant 25 Showing your work on the LAN while working with Laravel 30 Sharing access to your Vagrant to create a Docker host 40 Using Docker containers on CoreOS with Vagrant 44 Chapter 2: Provisioning IaaS with
Terraform 47 Introduction 48 Configuring the Terraform 56 Generating and using an SSH key pair to use on AWS 51 Using AWS security groups with Terraform 60 Using contextual defaults with Terraform 61 i Table of Contents
Managing S3 storage with Terraform Creating private Docker repositories with Terraform Chapter 3: Going Further with Terraform
Terraform 82 Provisioning a CentOS 7 EC2 instance with Chef using Terraform 85 Using Docker with Terraform 94 Simulating infrastructure changes using Terraform 98 Teamwork - sharing Terraform infrastructure state 103
Maintaining a clean and standardized Terraform 129 Chapter 4: Automating Complete Infrastructures with Terraform 129 Chapter 5: Provisioning the Last Mile with Cloud-Init 165
 Introduction 129 Provisioning a complete CoreOS infrastructure on Digital Ocean with Terraform 130 Provisioning a three-tier infrastructure on Google Compute Engine 137 Provisioning a GitLab CE + CI runners on OpenStack 146 Managing Heroku apps and add-ons using Terraform 153 Creating a scalable Docker Swarm cluster on bare metal with
Packet 156 Introduction 165 Using cloud-init 170 Configuring the server's time zone using cloud-init 171 Managing users, keys, and credentials using cloud-init 172 Managing repositories and packages using cloud-init 174 Running commands during boot using cloud-init 176 Managing users, keys, and credentials using cloud-init 170 Configuring the server's time zone using cloud-init 171 Managing users, keys, and credentials using cloud-init 176 Managing users, keys, and credentials using cloud-init 170 Configuring the server's time zone using cloud-init 171 Managing users, keys, and credentials using cloud-init 170 Configuring the server's time zone using cloud-init 171 Managing users, keys, and credentials using cloud-init 170 Configuring the server's time zone using cloud-init 170 Configuring the server's
Configuring CoreOS using cloud-init 178 Deploying Chef Client from start to finish using cloud-init 184 Deploying a remote Docker server using cloud-init 187 ii Table of Contents Chapter 6: Fundamentals of Managing Servers with Chef and Puppet 191 Introduction 192 Getting started (notions and tools) 192 Installing the Chef Development kit and
Puppet Collections 195 Creating a free hosted server Chef account and a Puppet server 197 Automatically bootstrapping a Chef client and a Puppet agent 201 Installing packages 204 Managing files, directories, and templates 216 Handling dependencies 223 More dynamic code using notifications 226 Centrally sharing data
using a Chef data bag and Hiera with Puppet 230 Creating functional roles 237 Managing external Chef cookbooks and Puppet 249 Chapter 7: Testing and Writing Better Infrastructure Code with Docker 335 Introduction 249 Linting
Chef code with Foodcritic and Puppet code with puppet-lint 250 Unit testing with ChefSpec and rspec-puppet 271 Integration testing with ServerSpec 276 Introduction 287 Maintaining consistent systems using scheduled convergence 288 Creating environments 293 Using
Chef encrypted data bags and Hiera-eyaml with Puppet 298 Using Chef Vault encryption 306 Accessing and manipulating system information with Ohai 308 Automating application deployment (a WordPress example) 313 Using a TDD workflow 320 Planning for the worse - train to rebuild working systems 329 Introduction 335 Docker usage overview
336 Choosing the right Docker base image 339 Optimizing the Docker image size 343 Versioning Docker images with tags 345 Deploying a Ruby-on-Rails web applications with Docker 347 Building and using Golang applications with Docker 349 Networking with Docker 351 iii Table of Contents Creating more dynamic containers Auto-configuring
dynamic containers Better security with unprivileged users Orchestrating with Docker Compose Linting a Docker Containers 354 355 357 359 363 365 369 Introduction 369 Testing Docker containers with BATS 370 Test-Driven Development (TDD) with Docker
and ServerSpec 374 The workflow for creating automated Docker builds from Git 380 The workflow for connecting the Continuous Integration (CI) system 388 Scanning for vulnerabilities with Quay.io and Docker 399 Debugging containers
using sysdig 404 Index 409 iv Preface In continuously evolving environments, operations and development teams are increasingly working together, using tools and techniques and sharing a common culture popularized as part of the DevOps movement. Docker has a logging driver just for it. Let's reuse in s3.tf something close to what we did in
Chapter 2, Provisioning IaaS with Terraform: resource "aws_s3_bucket" | bucket = "iacbook-${var.environment}" } 84 Chapter 3 It's the same construction here, each environment will get its bucket dynamically named after it. Let's say our
 objective is to create a file named /etc/aws/credentials that will contain the unencrypted version on the Puppet server. With Puppet, environments are located in distinct directories on the public part of the key
pair to OpenStack. However, it becomes a more useful container when giving it arguments, as it dynamically will add them to the apache/2.4.10 (Debian) Server built: Sep 15 2016 20:44:43 We can interactively pass
/usr/sbin/apache2ctl arguments without the need to override the entrypoint, for example, to propose alternatives Apache configuration files or options. It created the required virtual network interfaces, then the Ubuntu VM got a private IP address. In Chefspeak, it is called LWRP (Lightweight Resources and Providers). (ID: sgd3dbd8b4)
aws_instance.dev.0: Refreshing state... To help us improve, please leave us an honest review on this book's Amazon page at . We'll see how to quickly use Docker to achieve some tasks such as executing an Ubuntu container or networked webserver, sharing data with a container, building an image, and accessing a registry other than the default one
Here's how we can manage our EC2 key pair using a variable, with the name admin key. Start Vagrant and see that the VirtualBox Guest Additions are installed: $ vagrant up [...] Installing VirtualBox Guest Additions are installed: $ vagrant up [...] Installing VirtualBox Guest Additions are installed: $ vagrant up [...] Installing VirtualBox Guest Additions are installed: $ vagrant up [...] Installing VirtualBox Guest Additions are installed: $ vagrant up [...] Installing VirtualBox Guest Additions are installed: $ vagrant up [...] Installing VirtualBox Guest Additions are installed: $ vagrant up [...] Installing VirtualBox Guest Additions are installed: $ vagrant up [...] Installing VirtualBox Guest Additions are installed: $ vagrant up [...] Installing VirtualBox Guest Additions are installed: $ vagrant up [...] Installing VirtualBox Guest Additions are installed: $ vagrant up [...] Installing VirtualBox Guest Additions are installed: $ vagrant up [...] Installing VirtualBox Guest Additions are installed: $ vagrant up [...] Installing VirtualBox Guest Additions are installed: $ vagrant up [...] Installing VirtualBox Guest Additions are installed: $ vagrant up [...] Installing VirtualBox Guest Additions are installed: $ vagrant up [...] Installing VirtualBox Guest Additions are installed: $ vagrant up [...] Installing VirtualBox Guest Additions are installed: $ vagrant up [...] Installing VirtualBox Guest Additions are installed: $ vagrant up [...] Installing VirtualBox Guest Additions are installed: $ vagrant up [...] Installing VirtualBox Guest Additions are installed: $ vagrant up [...] Installing VirtualBox Guest Additions are installed: $ vagrant up [...] VirtualBox Guest Additions are installed: $ vagrant up [...] VirtualBox Guest Additions are installed: $ vagrant up [...] VirtualBox Guest Additions are installed as a vagrant up [...] VirtualBox Guest Additions are installed as a vagrant up [...] VirtualBox Guest Additions are installed as a vagrant up [...] VirtualBox Guest Additions are installed as a
 official Docker image: all the latest releases are found tagged like ruby: 2.3. Alternative builds from Alpine Linux and Debian Jessie slim images are also available. We want the local interface, using the same client traffic port (TCP/2379). Start the EC2 instance again and the updated configuration is applied. In other words, Terraform (the tool covered
in Chapter 2, Provisioning IaaS with Terraform, Chapter 3, Going Further with Terraform, and Chapter 4, Automating Complete Infrastructures with Terraform, and Subsequent boots before letting a full-fledged
configuration management tool such as Chef or Ansible play its part for the apache/manifests. The unit test should ensure the virtual host is created in /etc/apache2/sites-available and activated in /etc/apache2/sites-enabled. Let's use the string s3cr3t as an encryption key (weak). The recipe details
what exactly to run. Clair is an open source static analysis vulnerability scanner by CoreOS that we can run ourselves or build tools upon. ff metadata: This is any key value relevant to our needs, so we can schedule units accordingly. describe port(80) do it { is_expected.to be_listening } end # Vhost deployed? Using Git branches, Chef tools, Test
Kitchen, linting, and ServerSpec, we'll go through each and every step of a small project led by TDD principles. Packages can be downloaded from collections.html. To keep a high level of quality while our code base gets more and more complex, we'll switch to the Test-Driven Development (TDD) approach for our workflow. 105 Going Further with
Terraform You can now see the terraform state file in the S3 browser: Now make any change to the infrastructure, such as adding a new S3 bucket "lackbook-bucket" } After a terraform apply, simply push the changes: $ terraform the same of the infrastructure, such as adding a new S3 bucket bucket tags { Name = "IAC Book Bucket" } After a terraform apply, simply push the changes: $ terraform the same of the infrastructure, such as adding a new S3 bucket bucket bucket tags { Name = "IAC Book Bucket" } After a terraform apply, simply push the changes: $ terraform the same of the infrastructure, such as adding a new S3 bucket buck
remote push State successfully pushed! See the history in the S3 browser: 106 Chapter 3 The coworker has to configure their environment and pull the information: $$ terraform.tfstate" length of the state with remote state with remote state and pull the information: $$ terraform.tfstate" length of the state with remote state with remot
enabled! Remote state configured and pulled. #!/usr/bin/env bash # file: ./scripts/update os.sh sudo apt upgrade -yqq This script will install docker sudo systemctl enable docker sudo systemctl start docker. #!/usr/bin/env bash # file: ./scripts/install docker.sh curl -sSL | sh sudo systemctl enable docker sudo systemctl start docker. #!/usr/bin/env bash # file: ./scripts/install docker.sh curl -sSL | sh sudo systemctl enable docker sudo systemctl enable docker.
provisioner inside the packet_device resource: provisioner "remote-exec" { connection { user = "${var.ssh_username}" private_key = "${file("${var.ssh_username}" private_key = "$file("${var.ssh_username}" private_key = "$file("${var.ssh_username}")}" } scripts/install_docker.sh", 1 } At this point, the system is fully provisioned and functional, with
 Docker running. There's a myriad of installation options you can toy around with. Imagine what would be able to do everything on your behalf. Preparing files for transferring files to Finished verifying (0m0.00s). Installing packages We
need some packages for our server. 36 Chapter 1 Getting ready To step through this recipe, you will need the following: ff A working VirtualBox installation ff An Internet connection How to do it... Let's start with a minimal virtual machine named prod that simply boots a CentOS 7.2, like we have in our production
environment: Vagrant.configure("2") do |config| config.vm.box = "bento/centos-7.2" config.vm.hostname = "prod" con
unit called a 'cookbook' that configure and maintain a specific area of a system), we first need to install Chef on the Vagrant box. 301 Maintaining Systems Using Chef and Puppet 4. 49 Provisioning IaaS with Terraform It only means the code is valid, not that it can really authenticate with AWS (try with a bad pair of keys). You may be surprised when the code is valid, not that it can really authenticate with AWS (try with a bad pair of keys).
you collect some data and discover that many systems can be recovered in minutes. Would we like to see this issue combined with a 0-day local kernel security breach that would give the attacker access to the Docker host? An even better way to keep your code compatible with people without this plugin is to use this plugin configuration only if the
plugin is found by Vagrant itself: if Vagrant.has_plugin?("vagrant-vbguest") then config.vbguest.auto_update = false end 6. Docker will be modified to listen to the network through a socket kind of systemd service, and we'll configure the server time zone to be in New York. So let's create another user, exactly as before for Mary and Joe, except this
one will be for our Docker Engines and not for a real user in iam.tf. I encourage readers to use it extensively, and even integrate it in Continuous Integration (CI) systems and in Makefiles. Install EPEL. That way, we have an automatic history log of who accesses what. A common usage for such a setup is for the developer to test the application in
close to real production conditions, maybe to show a new feature to a remote product owner, replicate a bug seen only in this section: create an automated build and distribution pipeline from our code to GitHub to the Docker Hub. Let's write the Rspec test in Dockerfile spec.rb that will
check for this: describe "Docker NGINX image" do [...] describe package "nginx" should be installed (FAILED - 1) It's now time to add the instructions to the Dockerfile
on how to install NGINX: RUN apt-get update -y \ && apt-get install -y --no-install-recommends nginx=1.6.25+deb8u4 \ && apt-get clean \ && rm -rf /var/lib/apt/lists/* Relaunch the tests (it will take some time as it needs to build the image): $ bundle exec rspec --color --format documentation Docker NGINX image installs Debian Jessie Package
 "nginx" should be installed Finished in 51.89 seconds (files took 0.3032 seconds to load) 2 examples, 0 failures We're now sure the nginx package is installed. Just type b. We can write our own modules or use existing modules from GitHub of Puppet Forge. com/${aws_s3_bucket_object.index.key}" } Paste the link in a
web browser and you'll be able to access your file. Using the golang Docker image to cross-compile a Go program We can compile our program sharing the code folder, and setting the work directory to it: $ docker run --rm -v "${PWD}/src/hello -w /usr/src/hello -w /usr/src/hello golang:1.7 go build -v This way, even on a Mac OS system, we can generate
a proper ELF binary: $ file src/hello/hello src/hello/hello src/hello/hello: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, not stripped That said, if we explicitly want a Mac binary: $ docker run --rm -v
 "${PWD}/src/hello":/usr/src/hello -w /usr/src/hello -w /usr/src/hello -e GOOS=darwin -e GOARCH=amd64 golang:1.7 go build -v 349 Working with Docker Confirm we have a Mach-O executable and not an ELF binary: $ file src/hello/hello src/hello/hello src/hello -w /usr/src/hello -w /usr/src/hello -e GOOS=darwin -e GOARCH=amd64 golang:1.7 go build -v 349 Working with Docker Confirm we have a Mach-O executable and not an ELF binary: $ file src/hello/hello src/hello/hello src/hello/hello src/hello/hello src/hello/hello src/hello/hello src/hello/hello src/hello -w /usr/src/hello -w /usr/src/
build our program right from a Dockerfile and generate a Docker image out of it, that would translate like the following: FROM golang: 1.7 COPY src/hello /go/src/hello /g
notification section: template "/etc/httpd/conf.d/# {node["sitename"]}") notifies :restart, resources(:service => "httpd") end By default, actions are delayed at the end of the Chef run. In our case, we want to format the /dev/xvdb device
(so, unless we want to format our drive each time we reboot, we probably want this to happen only once on this instance. Both OpenJDK and JRE are available, for versions 6, 7, 8, and 9, based on Debian Jessie or Alpine. Alternatively, it is possible to upload the file: 168 Chapter 5 Verify the requested packages were installed, this time on a CentOS 7.2
box: [ ~]$ which nmap /usr/bin/nmap [ ~]$ which tcpdump Combining cloud-init file a few times. To
0 undocumented) Constants: 0 (0 undocumented) Puppet Classes: 1 (0 undocumented) Puppet Types: 0 (0 undocumented) Puppet Functions: 0 (0 undocumented) Puppet Types: 0 (0 undocumented) Puppet Types: 0 (0 undocumented) Puppet Classes: 1 (0 undocumented) Puppet Types: 0 (0 undocumented) Puppet Types: 0 (0 undocumented) Puppet Classes: 1 (0 undocumented) Puppet Types: 0 (0 undocumented) Puppet Types: 0 (0 undocumented) Puppet Classes: 1 (0 undocumented) Puppet Types: 0 (0 undocumented) Puppet T
 index.html css/ file.README.html frames.html index.html js/ puppet classes/ puppet classes/ puppet defined type list.html puppet defined types/ top-level-namespace.html The documentation is in the doc directory. So a trick is to create a simple script (I suggest you write a more robust one!), that reads the local token, and takes the local
 manager IP address as an argument. Docker 1.12 is used for this recipe. 335 Working with Docker Docker usage overview This section is an introduction to Docker for newcomers, and can be used as a refresher for others. You can choose to launch the command once per boot or once per instance. Another solution is not to specify any adapter name
10.10.0.11/24 on its own LAN (front lan), and on 10.20.0.101/24 on the application servers' LAN (app lan): # Tier 1: an NGINX reverse proxy VM, available on http:// localhost:8080 config.vm.hetwork "private_network", ip: "10.10.0.11", virtualbox_intnet: "front_lan'
 config.vm.network "private_network", ip: "10.20.0.101", virtualbox_intnet: "app_lan" config.vm.network "forwarded_port", guest: 80, host: 8080 config.vm.provision :shell, :inline => "sudo yum install -q -y epel-release" config.vm.provision :shell, :inline => "sudo yum install -q -y nginx" config.vm.provision :shell, :inline => "sudo yum install -q -y epel-release" config.vm.provision :shell, :inline => "sudo yum install -q -y nginx" config.vm.provision :shell, :inline => "sudo yum install -q -y epel-release" config.vm.provision :shell, :inline => "sudo yum install -q -y nginx" config.vm.provision :shell, :inline => "sudo yum install -q -y epel-release" config.vm.provision :shell, :inline => "sudo yum install -q -y nginx" config.vm.provision :shell, :inline => "sudo yum install -q -y epel-release" config.vm.provision :shell, :inline => "sudo yum install -q -y nginx" config.vm.provision :shell, :inline => "sudo yum install -q -y epel-release" config.vm.provision :shell, :inline => "sudo yum install -q -y nginx" config.vm.provision :shell, :inline => "sudo yum install -q -y epel-release" config.vm.provision :shell, :inline => "sudo yum install -q -y nginx" config.vm.provision :shell, :inline => "sudo yum install -q -y epel-release" config.vm.provision :shell, :inline => "sudo yum install -q -y nginx" config.vm.provision :shell, :inline => "sudo yum install -q -y epel-release" config.vm.provision :shell, :inline => "sudo yum install -q -y nginx" config.vm.provision :shell, :inline => "sudo yum install -q -y epel-release" config.vm.provision :shell, :inline => "sudo yum install -q -y nginx" config.vm.provision :shell, :inline => "sudo yum install -q -y epel-release" config.vm.provision :shell, :inline => "sudo yum install -q -y epel-release" config.vm.provision :shell, :inline => "sudo yum install -q -y epel-release" config.vm.provision :shell, :inline => "sudo yum install -q -y epel-release" config.vm.provision :shell, :inline => "sudo yum install -q -y epel-release" config.vm.provision :shell, :shell -q -y epel-re
 /vagrant/nginx.conf /etc/nginx/nginx.conf" config.vm.provision :shell, :inline => "sudo systemctl enable nginx" config.vm.provision :shell enable nginx config.vm
(while application servers can talk to the Redis backend). In this section, we'll migrate away from our distribution's MariaDB default and unconfigured package, to a fully configured MySQL 5.7—and that workflow is pretty close to everyday life using Chef in production. An interesting feature in Terraform is the ability to declare and use maps of
values, so, depending on a key, the variable can have a different value. Here, we'll show a whole workflow, from development to production, where we deploy Docker on CentOS 7 and Ubuntu LTS 16.04 for GitLab, with a dedicated block device for Docker, and two
other compute instances for GitLab CI runners. Getting ready To work through this recipe, you will need the following: ff A working Chef client configuration on the workstation ff A working Chef client configuration on the workstation ff A working Chef client configuration on the remote host ff The Chef code from the previous recipes How to do it... Our objective is to create two users—John and Mary. Let's start
by simply adding a line in outputs.tf that would show us how to connect to our virtual machine, using the public_ip attribute of our dev EC2 instance.dev.public_ip} -i ${var.aws_ssh_admin_key_file}" } When applying terraform next time, it will display the following: login = ssh
 -i keys/aws_terraform No doubt it's much quicker than having to log into the AWS dashboard, find the instance, and copy and paste the IP in the terminal. aws_access_key: AKIAJWTIBGE3NFDB4HOB aws_secret_key: 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 id: 300 us-east-1 Chapter 8 Now we have access to our data but in
entries, not just one. This is what our initial docker-compose up To upload an image to our private registry: image: registry; image: registry server: $ docker-compose up To upload an image to our private registry, the
process is to simply tag the image with the local registry URL and then push it. Starting with a full CentOS container image might be a waste of resources, while an Alpine Linux image might not contain the most complete libc for our usage. Thankfully, this can be very easily done. On all IaaS providers supporting cloud-init, there's a field where we
can paste our file. If you try to create a Vagrant box with a provider while using an image built for another provider, you'll get an error. Getting ready To step through this recipe, you will need the following: ff An AWS account with keys ff A working Terraform installation ff An empty directory to store your infrastructure code ff An Internet connection
48 Chapter 2 How to do it... To configure the AWS provider in Terraform, we'll need the following three files: ff A file declaring our variables for the whole project (terraform.tfvars) ff A provider file (provider.tf) Lets declare our variables in the
variables.tf file. Start by including the known dependencies to the service we already have: an Apache virtual host and MySQL. That's where the local-exec provisioner comes in, by simply echoing the rendered template from the data source into the file we want (in data.tf): resource "null resource "generate ansible hosts" { provisioner "local-exec" | provisioner |
command = "echo '${data.template_file.ansible_hosts.rendered}' > hosts" } } We use "null_resource" for this purpose, so the generation of the template is independent of any other executing resource. 254 Chapter 7 A Markdown editor with a live preview mode is available at . Create a module/mariadb/manifests/init.pp manifest file with the following
content: class mariadb { package {'mariadb-server': ensure => present, } } 7. In this simulation, we'll have the traditional three tiers, each running CentOS 7 virtual machines on their own isolated network: ff Front: NGINX reverse proxy ff App: a Node.js app running on two nodes ff Database: Redis Virtual Machine Name front_lan IP app_lan IP
parameter, which is the template file we just created, and the variables to be injected. To configure this, we'll use a systemd feature—socket activation. Those rules can be updated on-demand. As we know, an entry is simple JSON structured data. Now, let's run the unit test using the spec rake target: $ rake spec ... Finished in 2.88 seconds (files tooks)
 1.52 seconds to load) 13 examples, 0 failures The Rake targets provided also contain a lint target that can be used with rake lint. We'll do this to achieve maximum code quality among teams, from the development phase in the beginning to the final production environment. [...] Log in to the
box via SSH and check that Ansible is locally installed with the latest version: $ vagrant ssh $ ansible --version ansible 2.1.1.0 If your use case is different, you can use more precise deployment options, to be able to fix an Ansible version 1.9.6 instead of the latest 2.x series): It will take noticeably longer to start, as it
needs to install many packages on the guest system. We appreciate your help in protecting our authors and our ability to bring you valuable content. The ghost image will be in its 0.10 version tag, while nginx-proxy will use the 0.4.0 version tag, while nginx-proxy will use the 0.4.0 version tag. It gives access to many helpers for most use cases and most types of databases. As we're launching many
machines, you can optionally enable linked clones to speed things up: config.vm.provider :virtualbox do |vb| vb.memory = vm_memory vb.cpus = vm_cpus vb.linked_clone = true end 24 Chapter 1 Using named NAT networks Virtualbox do |vb| vb.memory = vm_memory vb.cpus = vm_cpus vb.linked_clone = true end 24 Chapter 1 Using named NAT networks Virtualbox do |vb| vb.memory = vm_memory vb.cpus = vm_cpus vb.linked_clone = true end 24 Chapter 1 Using named NAT networks Virtualbox do |vb| vb.memory = vm_memory vb.cpus = vm_cpus vb.linked_clone = true end 24 Chapter 1 Using named NAT networks Virtualbox do |vb| vb.memory = vm_memory vb.cpus = vm_cpus vb.linked_clone = true end 24 Chapter 1 Using named NAT networks Virtualbox do |vb| vb.memory = vm_memory vb.cpus = vm_cpus vb.linked_clone = true end 24 Chapter 1 Using named NAT networks Virtualbox do |vb| vb.memory = vm_memory vb.cpus = vm_cpus vb.linked_clone = true end 24 Chapter 1 Using named NAT networks Virtualbox do |vb| vb.memory = vm_memory vb.cpus = vm_cpus vb.linked_clone = true end 24 Chapter 1 Using named NAT networks Virtualbox do |vb| vb.memory = vm_cpus vb.linked_clone = true end 24 Chapter 1 Using named NAT networks Virtualbox do |vb| vb.memory = vm_cpus vb.linked_clone = true end 24 Chapter 1 Using named NAT networks Virtualbox do |vb| vb.memory = vm_cpus vb.linked_clone = true end 24 Chapter 1 Using named NAT networks Virtualbox do |vb| vb.memory = vm_cpus vb.linked_clone = true end 24 Chapter 1 Using named NAT networks Virtualbox do |vb| vb.memory = vm_cpus vb.linked_clone = true end 24 Chapter 1 Using named NAT networks Virtualbox do |vb| vb.memory = vm_cpus vb.linked_clone = true end 24 Chapter 1 Using named NAT networks Virtualbox do |vb| vb.memory = vm_cpus vb.linked_clone = vm_cpus vb.linked_clone
you can redeploy the system when a disaster comes. We know security groups can be multiple, so it's an array. Is it really working? Create a module/php/manifests/init.pp manifest file with the following content: class php { package {['php','php-mysql','libapache2-mod-php']: ensure => present, } } 6. 390 Chapter 10 As with every CI system, the final
step after the tests are completed is to package, ship, and deploy the containers. Executed through a containers it would look like this, as seen earlier in this book: FROM debian: jessie-slim COPY src/hello/hello /usr/bin/hello RUN chmod +x /usr/bin/hello RUN chmod +x /usr/bin/hello EXPOSE 8000 ENTRYPOINT ["/usr/bin/hello"] This will work, but things aren't looking that great
security-wise; our daemon is, in fact, running as the root user, even though it's running on an unprivileged port: $ ps aux USER PID %CPU %MEM COMMAND root 1 0.6 0.2 /usr/bin/hello VSZ 36316 RSS TTY 4180 ? With this module, we will be able to deploy a WordPress site from git. #cloud-config chef: install type: "omnibus" force install: true
omnibus_url: "This will output something like the following in the cloud-init logs: Getting information for chef stable for ubuntu... A data bag is simply data in a JSON file stored on the Chef server, that can be searched from the cookbooks. In production, this would probably be done through a configuration management tool such as Chef or Ansible
and maybe coupled with a proper deployment tool: # Tier 2: a scalable number of application servers v_{private} config.vm.hostname = "10.20.0.#{n+10}" do |config| config.vm.hostname = "app-#{n}" do |config.vm.hostname = "app-#{n}" do |config.vm.hostname = "10.20.0.#{n+10}" do |config.vm.hostname = "app-#{n}" do |c
virtualbox intnet: "app lan" config.vm.provision :shell, :inline => "sudo yum install -q -y nodejs npm" config.vm.provision :shell, :inline => "sudo yum install -q -y nodejs npm" config.vm.provision :shell, :inline => "sudo yum install -q -y nodejs npm" config.vm.provision :shell, :inline => "sudo yum install -q -y nodejs npm" config.vm.provision :shell, :inline => "sudo yum install -q -y nodejs npm" config.vm.provision :shell, :inline => "sudo yum install -q -y nodejs npm" config.vm.provision :shell, :inline => "sudo yum install -q -y nodejs npm" config.vm.provision :shell, :inline => "sudo yum install -q -y nodejs npm" config.vm.provision :shell, :inline => "sudo yum install -q -y nodejs npm" config.vm.provision :shell, :inline => "sudo yum install -q -y nodejs npm" config.vm.provision :shell, :inline => "sudo yum install -q -y nodejs npm" config.vm.provision :shell, :inline => "sudo yum install -q -y nodejs npm" config.vm.provision :shell, :inline => "sudo yum install -q -y nodejs npm" config.vm.provision :shell, :inline => "sudo yum install -q -y nodejs npm" config.vm.provision :shell, :inline => "sudo yum install -q -y nodejs npm" config.vm.provision :shell, :inline => "sudo yum install -q -y nodejs npm" config.vm.provision :shell, :inline => "sudo yum install -q -y nodejs npm" config.vm.provision :shell, :inline => "sudo yum install -q -y nodejs npm" config.vm.provision :shell -q -y node
 :inline => "sudo cp /vagrant/app.js /src/nodeapp" config.vm.provision :shell, :inline => "sudo chown -R vagrant.vagrant/srv/" config.vm.provision :shell, :inline => "sudo chown -R vagrant/nodeapp.service
/etc/systemd/system" config.vm.provision :shell, :inline => "sudo systemctl daemon-reload" config.vm.provision :shell :sh
run every 15 minutes: $ sudo crontab -1 # Chef Name: chef-client */15 * * * * /bin/sleep 69; /usr/bin/chef-client > /dev/null 2>&1 There's more... With Puppet, the agent can also run as a service or a cron. Storing updated cookbooks/starter/files/default/sample.txt in the cache. (ID: i-0018b1044953371ae) aws_instance.dev.1: Creating... Now, you know, you know, you know instance.dev.1: Creating...
you can scale, and with such a configuration you can push the nginx service to 500, 1000, or maybe more! 164 5 Provisioning the Last Mile with Cloud-Init In this chapter, we will cover the following recipes: ff Using cloud-init fn this chapter, we will cover the following recipes: ff Using cloud-init fn this chapter, we will cover the following recipes: ff Using cloud-init fn this chapter, we will cover the following recipes: ff Using cloud-init fn this chapter, we will cover the following recipes: ff Using cloud-init fn this chapter, we will cover the following recipes: ff Using cloud-init fn this chapter, we will cover the following recipes: ff Using cloud-init fn this chapter, we will cover the following recipes: ff Using cloud-init fn this chapter, we will cover the following recipes: ff Using cloud-init fn this chapter, we will cover the following recipes: ff Using cloud-init fn this chapter, we will cover the following recipes: ff Using cloud-init fn this chapter, we will cover the following recipes: ff Using cloud-init fn this chapter, we will cover the following recipes: ff Using cloud-init fn this chapter, we will cover the following recipes: ff Using cloud-init fn this chapter, we will cover the following recipes: ff Using cloud-init fn this chapter, we will cover the following recipes: ff Using cloud-init fn this chapter.
Managing users, keys, and credentials using cloud-init ff Managing repositories and packages using cloud-init ff Running commands during boot using cloud-init ff Deploying a remote Docker server using cloud-init Introduction Cloud-init is a cloud
instance initialization system, standard across most Linux distributions. Alternatively, maybe you need a specific Ansible installation. To begin with, we need the Docker registry v2 image: registry: 2. Here, we have the three assertions we mentioned
 earlier when describing the goal of the test. However, you can simply create users like this: --- name: create mary user iam: you can simply create users like this: --- name: create mary user iam: when describing the goal of the test. However, you can simply create users like this: --- name: create mary user iam: when describing the goal of the test. However, you can simply create users like this: --- name: create mary user iam: when describing the goal of the test. However, you can simply create users like this: --- name: create mary user iam: when describing the goal of the test. However, you can simply create users like this: --- name: create mary user iam: when describing the goal of the test. However, you can simply create users like this: --- name: create mary user iam: when describing the goal of the test. However, you can simply create users like this: --- name: create mary user iam: when describing the goal of the test. However, you can simply create users like this: --- name: create mary user iam: when describing the goal of the test. However, you can simply create users like this: --- name: create mary user iam: when describing the goal of the test. However, you can simply create users like this: --- name: create mary user iam: when describing the goal of the test. However, you can simply create users like this: --- name: create mary user iam: when describing the goal of the test. However, you can simply create users like this: --- name: create mary user iam: when describing the goal of the test. However, you can simply create users like this: --- name: create mary user iam: when describing the goal of the test. However, you can simply create users like this: --- name: create mary user iam: when describing the goal of the test. However, you can simply create users like this: --- name: create mary user iam: when describing the goal of the test. However, you can simply create users like the goal of the test. However, you can simply create users like the goal of the goal of the goal of the goal of the g
for our user Mary. pomes.pro. He has also contributed to the pfSense project. Starting Ghost engine As you would do normally, log in to your Vagrant box to launch the node server: vagrant ssh cd /srv/blog && sudo npm start --production... Pages and pages of tested, reliable code ready to use! Good. For example and pages of tested, reliable code ready to use!
start by logging into it via Secure Shell (SSH) by issuing the following vagrant command and use the system normally: $ vagrant ssh Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 4.4.0-34-generic x86 64) [...] :~$ free -m [email protecte
example for us as they have a good API and a free service tier for us to try with Terraform. However, an interesting alternative is AWS Elastic Container Registry (ECR): pricing is different and fully integrated in the AWS ecosystem. It's nice, but we're still repeating ourselves with the values, leading to possible errors, omissions, or mistakes in the
future. [...] Apply complete! Resources: 2 added, 0 changed, 0 destroyed. A role can include other roles as well as recipes and open source
Virtualbox hypervisor ff An Internet connection How to do it... Open a terminal and type the following code: $ vagrant box add ubuntu/xenial64' box: URL: ==> box: Adding box 'ubuntu/xenial64' (v20160815.0.0) for provider: virtualbox box: Downloading:
versions/20160815.0.0/providers/virtualbox.box ==> box: Successfully added box 'ubuntu/xenial64' (v20160815.0.0) for 'virtualbox'! How it works... Vagrant knows where to look for the latest version for the requested box on the Atlas service and automatically downloads it over the Internet. You'll end up someday in a situation where someon
forgets to push or pull. image: demo-httpd ports: - "80:80" This way, running docker-compose up will also build the image if absent. This is simply added in the mysite/recipes/htaccess.rb file: users = search(:webusers, "*:*") 3. How to do it... Most common distributions are available as a container form. If the file method is chosen for the shared secret
the final argument will be the path to the secret key file to decrypt the data. We can also have a mysql module for the MySQL server, with its own resources. Creating an HTTPS test A very common test we'll want to make is HTTP availability. puppetlabs-release-pc1-el-7.noarch.rpm -y 93 Going Further with Terraform ff Install the
Puppet agent: sudo yum install puppet-agent -y ff Display the Puppet version: sudo /opt/puppetlabs/bin/puppet agent --version Let's add those commands inside a remote-exec provisioner "remote-exec provisioner inside our aws_instance resource, changing the default username to centos: provisioner "remote-exec provisioner inside our aws_instance resource, changing the default username to centos: provisioner "remote-exec provisioner inside our aws_instance resource, changing the default username to centos: provisioner "remote-exec provisioner inside our aws_instance resource, changing the default username to centos: provisioner "remote-exec provisioner inside our aws_instance resource, changing the default username to centos: provisioner "remote-exec provisioner inside our aws_instance resource, changing the default username to centos: provisioner "remote-exec provisioner inside our aws_instance resource, changing the default username to centos: provisioner "remote-exec" { inline = ["echo \"nameserver 8.8.8.8.8." | sudo tee -a contos: provisioner inside our aws_instance resource, changing the default username to centos: provisioner inside our aws_instance resource, changing the default username to centos: provisioner inside our aws_instance resource, changing the default username to centos: provisioner inside our aws_instance resource, changing the default username to centos: provisioner inside our aws_instance resource, changing the default username to centos: provisioner inside our aws_instance resource, changing the default username to centos: provisioner inside our aws_instance resource, changing the default username to centos: provisioner inside our aws_instance resource, changing the default username to centos: provisioner inside our aws_instance resource, changing the default username to centos: provisioner inside our aws_instance resource, changing the default username to centos: provisioner inside our aws_instance resource, changing the default username to contos: provisioner inside our aws_instance resou
/etc/resolv.conf", "sudo yum update -y", "sudo yum install epel-release -y", "sudo yum install epel-release -y", "sudo yum install puppet-agent -y", "sudo yum install puppet agent -version"] connection { user = "centos" } } When you terraform apply this, you'll end up with a fully updated CentOS 7.2 system, with EPEL available, a custom DNS server added
 and Puppet agent installed. Quality can only go higher: the more the checks, the better. This book is a humble attempt to cover everything related to infrastructure deployments with Terraform or Ansible, from configuration
management essentials using Chef and Puppet to advanced Test-Driven Development (TDD) techniques, and thorough infrastructure code coverage testing. At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks
Managing GitHub with Terraform There're many service providers to use with Terraform. In our case, we want the latest and greatest MySQL 5.7 with the admin password: super_secure password. Piracy Piracy of copyrighted material on the Internet is an ongoing problem across all media. That being said, many popular programming languages (Go
Node, Java, Python, Ruby, PHP, and more) are also shipping their own container images. Verifying that a second run of Puppet does not change anything: we want to prove the code is idempotent. Enabling multiprovider Vagrant environments You might be running VMware on your laptop, but your coworker might not. It will also build any other tag
you may create in the future: 386 Chapter 10 Let's tag our code as 1.0 on Git so we can refer to it later: $ git tag 1.0 -> 1.0 This just triggered a new build on the Docker Hub, using the tag 1.0, as we asked to match: 387 Maintaining Docker Containers Everyone can now refer to
this stable build and use it without fearing a breaking change from the master branch; this branch will always be built with the following line on the Dockerfile: FROM
sjourdan/nginx-docker-demo:1.0 We now have a nice initial workflow for building master and tagged, stable releases of our containers. VDOC.PUB Authors: Jourdan, Stephane PDF Download Embed This document was uploaded by our user. Then it applies run list with the requested Chef mode (chef-zero in our case). To achieve this, we could use
different techniques, but the data bag feature in Chef is pretty convenient for our objective. Add the following output to outputs.tf so we know all the IP addresses of our runners: output "GitLab Runner Instances" { value = "${join(" ", openstack compute instance v2.runner.*.access ip v4)}" } Creating an object storage container on OpenStack This
one is very simple: it only requires a name and a region. 20 Chapter 1 We need the AWS Access Keys, preferably from environment variables so you don't hardcode them in your Vagrantfile: aws.access_key_id = ENV['AWS_ACCESS_KEY_ID'] aws.secret_access_key = ENV['AWS_SECRET_ACCESS_KEY'] Indicate the region and availability zone where
you want the instance to start: aws.region = "eu-west-1" aws.availability_zone = "eu-west-1" aws.avail
requirements to your needs): aws.security_groups = ['iac-lab'] Specify the AWS keypair name, and override.ssh.username = "ubuntu" ov
AWS EC2, so I choose to disable this feature: override.nfs.functional = false Finally, it's a good practice to tag the install Docker and docker-compose, then execute the docker-compose file: #!/bin/sh # install Docker curl
sSL | sh # add ubuntu user to docker group sudo usermod -aG docker ubuntu # install docker-compose curl -L uname -s`-`uname -m` > /usr/local/bin/docker-compose # execute the docker compose file cd /vagrant docker-compose up -d Include both NGINX
configuration and docker-compose.yml files from the previous recipe and you're good to go: $ vagrant up Bringing machine 'srv-1' up with 'aws' provider... The last step is to store the Swarm token, so the nodes can join. [...] + aws key pair.admin key [...] Plan: 2 to add, 0 to change, 0 to destroy. It's
done using the google_compute forwarding rule resource, simply pointing to the pool of hosts we created earlier. This is no different: config.vm.provider :aws do |aws, override| # AWS Configuration override.vm.box = "dummy" end Then, defining an application VM will consist of specifying which provider it's using (AWS in our case), the Amazon
Machine Image (AMI) (Ubuntu 16.04 LTS in our case), and a provisioning script that we creatively named script.sh. It collects, organizes, and displays metrics about resource usage, container by contai
Chef resources available—the deploy resource—and the various powerful and popular cookbooks, such as the database cookbooks, s
have to call this new recipe from our main, default.rb recipe! In mysite/recipes/default.rb, include our new recipe, so it gets picked up by the client: include_recipe "mysite::htaccess" Just upload the new version of the cookbook: $ knife cookbook upload mysite After you've run chef-client on your node, the site will be protected and users many and john
will be able to use basic HTTP authentication. Getting ready To step through this recipe, you will need the following: ff A working Terraform installation ff An AWS account with an SSH key configured in Terraform (refer to the Chapter 2, Provisioning IaaS with Terraform recipes) ff A working Docker installation for the Consul simulation solution
(optional) ff An Internet connection How to do it... Let's start by having an initial infrastructure running (a single virtual machine for this example). This is totally arbitrary; adapt keys and values to your own needs. With Markdown, it becomes easy to add headers, links, bullets and font effects. The next step is to register automatically every droplet in
 the DNS. This is where the search feature in Chef is useful to create dynamically generated content. Good job! We've built our first simple Docker container following TDD principles. To simulate a small infrastructure, we will use Vagrant with Ubuntu boxes (for more information about Vagrant, please refer to Chapter 1, Vagrant Development
Environment). In other situations, we can perfectly use the "local-exec" { } provisioner directly from inside a standard resource. The registry will be available name or a dedicated server with a locally available name. We can use modules from Puppet Forge or GitHub for example. At the end of this part,
you'll know how to configure the etcd key value store, the flannel overlay network, control the update mechanism, and ensure systemd units are started as early as possible. In this case, I selected a Xenial release (16.04 LTS), on the eu-west-1 zone (Ireland), running on HVM virtualization and backed by SSD disks. When all
 the tests have passed successfully, we many them into the production branch and use r10k to deploy the code. A Puppet piece of code is called a manifest, and is a file with a .pp extension. The name of the service resource is the same as used in a root shell to stop/start/reload the service. This delay can be changed in
/etc/puppetlabs/puppet.conf. Here, no conflicts are noted, so we can write a message explaining what our request is all about. We previously used a class statement, but each class in Puppet can be used only once per catalog (remember, a catalog is the result of the compilation for a node). documentation about VMX configuration. Restart the daemon
and start logging your containers output using a new Docker logging driver, using the log group name specified in Terraform earlier (docker_logs): $ docker_logs --log-opt awslogs-stream=nginx nginx:stable Generate some activity on the
container: $ curl -IL HTTP/1.1 200 OK 73 Provisioning IaaS with Terraform Refresh the AWS CloudWatch page and you'll see a new entry named nginx with the container logs. Commit and push this file and it will trigger our first test on Travis: $ git add .travis.yml $ git commit -m "added travis.yml" $ git push 389 Maintaining Docker Containers 6
We want Extra Packages for Enterprise Linux (EPEL): sudo yum install -q -y epel-release 2. This is very useful for debugging when things go wrong. The default unencrypted port, it's highly recommended to
configure TLS encryption; this will use the TCP/2376 by convention). 325 Maintaining Systems Using Chef and git Now that our platform cookbook works pretty well in our docker support git branch, let's commit that work. Mapt gives you full access
to all Packt books and video courses, as well as industry-leading tools to help you plan your personal development and advance your career. 0 A simple BATS test for a command return code looks like this: @test "Apache version is test," test title { run [$status -eq 0] } We now have everything we need to write our first BATS test in test/httpd.bats: @test "Apache version is test," test title { run [$status -eq 0] } We now have everything we need to write our first BATS test in test/httpd.bats: @test "Apache version is test," and the provided in the
correct" { run docker run --rm demo-httpd:latest -v \ grep 2.4.10 [$status -eq 0] } To execute our test, let's launch BATS with the folder containing the tests as arguments: $ bats test Y Apache version is correct 1 test, 0 failures Good! We're now assured that the correct Apache version is installed. 40 Chapter 1 Getting ready To step through this
 you will need the following: ff A working Vagrant installation ff A working hypervisor ff A working Ansible installation on your machine (an easy way is to $ pip install ansible or to pick your usual package manager like APT or YUM/DNF) ff An Internet connection How to do it... Because writing complex Ansible
this book, we'll use a very simple one, so you can learn more about Ansible later and still reuse this recipe. At the time of writing, the supported Ubuntu releases are 12.04 (precise), 14.04 (trusty), 16.04 (xenial), and 16.10 (yakkety). And finally, as we're using MySQL, let's make sure we depend on its official cookbook. This is how our cloud-config.yml
file looks with all the etcd configuration: #cloud-config coreos: etcd2: 179 Provisioning the Last Mile with Cloud-Init discovery: " advertise-peer-urls: "http://$private ipv4:2380" listen-peer-urls: "http://$private ipv4:2380" listen-client-urls: "http://$private ipv4:2380" listen-peer-urls: "http://$private ipv4:2380" listen-client-urls: "http://$private ipv4:2380" listen-peer-urls: "http://$private ipv4:2380" listen-client-urls: "http://$private ipv4:2380
unit file found at /run/systemd/ system/etcd2.service.d/20-cloudinit.conf. On Google Cloud, we'll build a three-tier infrastructure with two HTTP nodes behind a load balancer and an isolated MySQL managed database. Using the Alpine Linux alternative for a Go program The main problem with the scratch image solution is the impossibility to debug it
easily from inside the container, and the impossibility to rely on external libraries or dependencies such as SSL and certificates. Getting ready To step through this recipe, you will need the following: ff A working Docker knowledge ff An Internet connection How to
do it... The previous example allows only one container to be launched simultaneously, which is sad considering the power of Docker. Here's the apt.bats test file, executing the shell command instead of apache2ctl, expecting a successful execution and an output of 0: @test "apt lists are empty" { run docker run --rm --entrypoint="/bin/sh" demo-
httpd:latest -c "ls -1 /var/lib/apt/lists | wc -l" [$status -eq 0] ["$output" = "0"] } 372 Chapter 10 Execute the BATS tests: $ bats test apt lists are empty Apache version is correct 2 tests, 0 failures Using Makefile to glue it all together Now this whole process might be a bit tedious in CI, with some additional steps needed before the testing is done (the
image needs to be built and made available before it is tested, for example). $ cat /run/systemd/system/etcd2.service.d/20-cloudinit.conf [Service] Environment="ETCD ADVERTISE CLIENT URLS=: 2380'
Environment="ETCD LISTEN CLIENT URLS=" Environment="ETCD LISTEN PEER URLS=" When we have our cluster ready, we'll be able to request information as a client on the specified port: $ etcdctl cluster-health member 7466dcc2053a98a4 is healthy; got healthy
member e0f77aacba6888fc is healthy: got healthy we can also navigate the etcd key value store to configuring fleet using cloud-init Fleet is a distributed init manager based on systemd that we use to schedule services on our CoreOS cluster. It's really up to everyone to
know what has to be shared with whom to know what the Chef organization will be. Getting ready To work through this recipe, you will need the following: ff A working Chef DK installation on the workstation ff A working Chef DK installation ff A work
cookbook we've left in its 0.3.0 version. Let's write a provider for AWS like this in provider aws" { region = "${var.aws region}}" } Syntax validation Try to validate Error validation Try to validate Error validation Try to validate that file, and it will notify us that we're missing a variable: $ terraform validate Error validation Try to validate that file, and it will notify us that we're missing a variable: $ terraform validation Try to validate Error validation Try to validate that file, and it will notify us that we're missing a variable: $ terraform validation Try to validate Error validation Try to validate Err
'aws': unknown variable referenced: 'aws region'. Maybe part of the command is the one to keep (like we always want the OpenVPN binary and options to be executed, no matter what), maybe everything needs to be overridden (that's the toolbox container model, such as a /bin/bash command by default, but any other command given in argument can
otherwise be executed), or a combination of the two, for a much more dynamic container. Follow the instructions to have your Git repository locally working. Copy and paste between the markers:
releases is set by the keep releases integer: $ ls -ld /var/www/mysite/current lrwxrwxrwx. We want the local interface, using the same peer traffic port (TCP/2380). How it works... The whole Chef environment, as well as its dependencies is deployed under /opt/chefdk. Let's create another remote-exec provisioner, so the Swarm manager is initialized
automatically, right after bootstrap: provisioner "remote-exec" { connection { user = "${file("${var.ssh key}")}" } inline = ["docker swarm init --advertise-addr ${packet device.swarm master.network.2.address}",] } At this point, we have a Docker cluster running, with only one node—the manager itself. In
this case, we've declared variables with an optional description for reference. If tomorrow our IaaS cloud provider crashes, in what timeframe will we be able to rebuild systems somewhere else (provided the backups are working; well, that's another story)? All VM definitions will live inside the main Vagrant configuration: Vagrant.configure('2') do
|config| config.vm.define 'whatever_vm', autostart: false do |node| [...] end end We suggest disabling the automatic start of VMs so we don't make the mistake of launching dozens of VMs by error. If we want to run a cookbook right after bootstrap (and we probably want to), just use the -r option to add cookbooks to the run list, so they are executed
right away. However, other tests are still failing, though. Install the vagrant-vbguest plugin: $ vagrant-vbguest plugin. An AMI is like a system disk image for AWS, and is referred to by its ID (that is: ami-df3bceb0 or ami-f2fc9d81). Chef-Solo: In this mode, the need for a Chef server is removed at
the cost of less features, including important ones such as search, API, persistent storage of nodes information, and more. Let's create the shared folder and start Vagrant: mkdir puppet. That's the reason we need integration testing; testing the
outcome of the combination of all our cookbooks applied to a real test system, and now. Check this using the following code: $ git status On branch docker_support nothing): $ git push Now one of our coworkers can peer review the code
and eventually merge docker support with the master branch: $ git merge docker support master Our new cookbook is now ready for prime time and can be deployed to staging. Sharing access your application locally through your Vagrant box, let's give access to it to others through the Internet using vagrant share: HTTP The
default is to share through HTTP, so your work is available through a web browser: $ vagrant share ==> srv-1: Detecting network information for machine... Then it will do the install action by default. To ensure sequential startup, use the
VAGRANT NO PARALLEL environment variable at the top of the Vagrantfile: ENV['VAGRANT NO PARALLEL'] = 'true' Now you can browse to and start using your Ghost blog in a container, behind a NGINX reverse proxy container, with the whole thing managed by Vagrant! There's more... You can access the containers logs directly using Vagrant.
$ vagrant docker-logs --follow ==> app: > start /usr/src/ghost ==> app: > node index ==> app: Nigrations: Creating tables... See also ff The Chef documentation on data bags: . Using mysql (8.0.4) Using selinux (0.9.0) Using yum-mysql-community (1.0.0) [...] 4. Create a data bag folder aws to store the credentials for the us-east-1
region: $ mkdir data bags/aws 2. This account has one authorized public key automatically populated. Using Chef, we write infrastructure code in Ruby that describes how every aspect of the system is expected to behave according to a number of conditions, then apply it through various client tools to ensure the defined state is applied. This Vagrant
plugin is used to create host entries in /etc/hosts in managed boxes and in your workstation. We do this using the remote-exec provisioner "remote-exec provisioner, correctly configured with the right SSH username and private key = "${file("${var.ssh key}")}" } inline = [
 "sudo yum update -y", "sudo yum install -y docker", "sudo systemctl enable docker", "sudo systemctl start docker",] } We're finally done, with our two instances automatically provisioned! Creating a Google Compute Firewall rule Our goal is simple: we want to allow anyone (0.0.0.0/0) to access using HTTP (TCP port 80) any instance with the tag www
in the default network. What if the current setup that works perfectly well is in fact working because there's a script or a binary somewhere left from last year, which does the thing that makes it work? There's more... Puppet is a configuration tool published by Puppet Labs, and is an alternative to Chef. To set the time zone to New York on CoreOS,
we would set it like this: $ /usr/bin/timedatectl set-timezone America/New York Easy! So let's launch that command through a systemd unit in the cloud-config file, so we're sure the time zone is set. 56 Chapter 2 This example uses the t2.micro instance available for free in the AWS Free Tier. We're not limited to a single day, so we could also allow
reboots any day at 4 am: coreos: locksmith: window-start: Sat 04:00 window-length: 2h 182 Chapter 5 This generates the following content in /run/system/locksmithd. 159.203.189.146 provider=digitalocean, region=ams 3b4b0792... It requires a name, a description, and a run list. Let's launch the Chef client on our remote node: $ sudo chef-
client Starting Chef Client, version 12.15.19 resolving cookbooks for run list: ["apache"] Synchronizing Cookbooks: - apache (0.1.0) Installing Cookbooks Gems: Compiling Cookbooks for run list: ["apache"] Synchronizing Cookbooks Gems: Compiling Cookbooks Gems: Cookbooks
mysql2 chef gem, which fortunately comes with a dedicated cookbook as well. 2. Being able to send the usual Docker commands to a remote server has a multitude of advantages. The VMX format is not very well documented by VMware. Here's how to customize VirtualBox similarly in the Vagrantfile: config.vm.provider: virtualbox do |vb| vb.memory
= "1024" vb.cpus = "2" end Add this to your current Vagrantfile, reload and you'll get the requested resources from your hypervisor, be it VMware or VirtualBox. In our case, we want a simple section named centos7 hosts and the servers IP as follows: [centos7 hosts] 1.2.3.4 Let's construct our first template named hosts.tpl with a variable named
host public ipv4 that will ultimately be replaced by the real future IP of the host we'll later launch: [centos7 hosts] ${\text{host public ipv4}}$ To generate this file, we'll use a template with a variable in it, that Terraform will generate this file, we'll use a template with a variable in it, that Terraform will generate this file, we'll use a template with a variable in it, that Terraform will generate this file, we'll use a template and passes it the variable
we need from our AWS instance: data "template file" "ansible hosts" { 90 Chapter 3 template internally, meaning the data is available, but not dumped anywhere. Luckily, Vagrant can work with
those named NAT networks too. @rgevaert mysql rocha-mysql Resources to ... We need four pieces of information: a username, a password, an OpenStack tenant name, and an OpenStack tenant n
storage The Docker registry is a central image distribution service. Getting ready To step through this recipe, you will need: ff A free Docker build pipeline. Often, there's a very nice gain in size! 344 Chapter 9 Versioning Docker
images with tags A very common need is to quickly identify what version of the software a Docker image is running and optionally stick to it, or to be sure to always run a stable version. 175 Provisioning the Last Mile with Cloud-Init Running commands during boot using cloud-init When bootstrapping a new server or instance, the first boot is often
very different from all the other boots the instance will experience in its life, and most often we want some commands to be executed very early or very late in the boot process. Here's a checklist of what we know about our MySQL database: ff It's running on us-east1 region ff It's running MySQL 5.6 ff It's type is D2 (1 GB of RAM) ff Our own network
and both HTTP servers can access it ff We want a database named app db ff We want a user with a password to be allowed to connect from the HTTP servers 142 Chapter 4 Let's put all these variables in the variables. If file: variable "db authorized network" { default =
"0.0.0.0/0" description = "A corporate network authorized to access the DB" \ variable "db username" \ default = "dbadmin" description = "A MySOL database name" \ default = "db name" \ description = "MySOL database name" \ Don't forget to
override each generic value in the terraform.tfvars: db authorized network = "163.172.161.158/32" db username = "sqladmin" db password = "pwd1970" db name = "app db" db type = "D2" Now we can build our database instance resource in a db.tf file: resource "google sql database instance" "master" { name = "app db" db type = "D2" Now we can build our database instance resource in a db.tf file: resource "google sql database instance" "master" { name = "app db" db type = "D2" Now we can build our database instance resource in a db.tf file: resource "google sql database instance" "master" { name = "app db" db type = "D2" Now we can build our database instance resource in a db.tf file: resource "google sql database instance" "master" { name = "app db" db type = "D2" Now we can build our database instance resource in a db.tf file: resource "google sql database instance" "master" { name = "app db" db type = "D2" Now we can build our database instance resource in a db.tf file: resource "google sql database instance" "master" { name = "app db" db type = "D2" Now we can build our database instance resource in a db.tf file: resource "google sql database instance" | name = "app db" db type = "D2" Now we can build our database instance resource | name = "app db" db type = "D2" Now we can build our database instance resource | name = "app db" db type = "D2" Now we can build our database | name = "app db" db type = "D2" Now we can build our database | name = "app db" db type = "D2" Now we can build our database | name = "app db" db type = "D2" Now we can build our database | name = "app db" db type = "D2" Now we can build our database | name = "app db" db type = "D2" Now we can build our database | name = "app db" db type = "D2" Now we can build our database | name = "app db" db type = "D2" Now we can build our database | name = "app db" db type = "D2" Now we can build our database | name = "app db" db type = "D2" Now we can build our database | name = "app db" db type = "D2" Now we can build our database | name = "app db" db ty
 "mysql-mastr-1" region = "${var.region}" database_version = "MYSQL_5_6" settings = { tier = "${var.db_type}" activation_policy = "ALWAYS" // vs "PACKAGE" ip_configuration { ipv4_enabled = true authorized_networks { name = "PER_USE" // vs "PACKAGE" ip_configuration { ipv4_enabled = true authorized_networks { name = "PER_USE" // vs "PACKAGE" ip_configuration { ipv4_enabled = true authorized_networks { name = "PER_USE" // vs "PACKAGE" ip_configuration { ipv4_enabled = true authorized_networks { name = "PER_USE" // vs "PACKAGE" ip_configuration { ipv4_enabled = true authorized_networks { name = "PER_USE" // vs "PACKAGE" ip_configuration { ipv4_enabled = true authorized_networks { name = "PER_USE" // vs "PACKAGE" ip_configuration { ipv4_enabled = true authorized_networks { name = "PER_USE" // vs "PACKAGE" ip_configuration { ipv4_enabled = true authorized_networks { name = "PER_USE" // vs "PACKAGE" ip_configuration { ipv4_enabled = true authorized_networks { name = "PER_USE" // vs "PACKAGE" ip_configuration { ipv4_enabled = true authorized_networks { name = "PER_USE" // vs "PACKAGE" ip_configuration { ipv4_enabled = true authorized_networks { name = "PER_USE" // vs "PACKAGE" ip_configuration { ipv4_enabled = true authorized_networks { name = "PER_USE" // vs "PACKAGE" ip_configuration { ipv4_enabled = true authorized_networks { name = nam
 "authorized network" value = "${var.db authorized network}" authorized networks { name = "${google compute instance.www.0.name}" value = "${google compute instance.www.0.name
"${google compute instance.www.1.network interface.0.access config.0.assigned nat ip}" } } } The pricing plan "PACKAGE" is more interesting for a long-lasting database. Based on Linux, targeting easy container and clustered deployments, it also provides official Vagrant boxes. Create a new Chef organization. Hover the mouse pointer on the
SUPPORT tab at the top. We need to stop it when we're done with the tests and delete it: after(:all) do @container.kill @container and fail upon checking for an nginx process (we didn't write anything that would launch nginx): $ bundle exec rspec --color --format
documentation Docker NGINX image installs Debian Jessie Package "nginx" should be installed Running the NGINX container process "nginx" should be running (FAILED - 1) 378 Chapter 10 Now let's execute /usr/bin/nginx for our container process "nginx" should be running the NGINX container process "nginx" should be runni
"daemon off;"] Rerun the tests to check whether the nginx process is now running as expected: $ bundle exec rspec --color --format documentation Docker NGINX image installs Debian Jessie Package "nginx" should be installed Running the NGINX container Process "nginx" should be running Finished in 1.94 seconds (files took 0.30853 seconds to
load) 3 examples, 0 failures To add simplicity when integrating these tests in CI systems, let's create a simple Makefile: test: rspec rspec: bundle exec rspec exect this one very often. Containers need to be rebuilt, exactly like developers are used
to rebuilding applications with updated libraries to get rid of bugged code. Testing another recipe from the same cookbook, let's create tests for our second recipe—virtualhost spec.rb. This is the most static way of shipping a plain file. Docker containers helps developers use, share and ship a bundle
```

```
including everything needed to run their application. However, Rubocop, is targeted at software development by default and is not really optimized for Chef cookbooks development. Configuring system are started right when we need them. Configuring
CoreOS using cloud-init CoreOS supports its own version of cloud-init, with added support for the CoreOS environment, and without everything else incompatible with its environment.
docker network create -d overlay --subnet 172.16.0.0/16 nginxnetwork Create a Docker service that will simply launch an nginx container running at the same time): # docker service create --name nginx --network nginx-network replicas 3 -p 80:80/tcp nginx 163 Automating
Complete Infrastructures with Terraform Verify if it's working: # docker service ls ID NAME REPLICAS IMAGE aeq9lspl0mpg nginx 3/3 nginx COMMAND Now, accessing by HTTP any of the public IPs of the cluster, any container on node-2 responding. [...]
Fetching cookbook index from ... Let's start by populating the content of this ERB with a minimal Apache configuration file, using a new website variable that we'll set in a minute. Finally, we need to deploy our PHP file under the DocumentRoot directory. Let's try the test! $ rake beaker ... Chef Solo does not support search.') else users =
search(:webusers, '*:*') end Upload the new version of the cookbooks/mysite Let's continue our investigation of the suggestions. Within seconds, you'll have a running Ubuntu 16.04 Vagrant box on your host and you can do whatever you want
 with it. Hiera can be seen as datastore keeping site information out of manifests. In the Dockerfile spec.rb file, create and start the container using the image: describe "Docker NGINX image" do [...] describe 'Running the image: describe 'Running the image: describe 'Running the image' => @image.id) @container.start end end
end Using standard ServerSpec checks, verify that an NGINX process is running: describe process("nginx") do it { should be running } end We can't stop here without cleaning up the container. This is how it looks in the cloud-config.yml file: coreos: fleet: public-ip: "$public ipv4" metadata: "region=eu-west-1,provider=aws,cluster=mycluster" This
will generate the right variables in the systemd unit at /run/systemd/system/fleet. Make a pull request Navigate to your repository, and you'll see something similar to the following screenshot, showing an information about the new branch being just pushed. Everyone can work using the same environment, regardless of their own local machine. The
cookbook really works. On Ubuntu, to enable a site, a link must be created in /etc/apache2/site-enabled; the second file directive will handle it. Other operating systems such as Ubuntu or CentOS are fully supported to provision Docker containers, even if Docker isn't installed at first on the base image. 213 Fundamentals of Managing Servers with
Chef and Puppet When the Chef client runs, it always downloads and applies the latest version by default. Starting from Puppet 4.x, facts can be accessed from manifests using the $facts hash. This file can contain a lot of information to do many things, but for now, it should simply tell Travis that we need a Ruby environment in a recent Linux
distribution running Docker. This is a perfect use for the Docker tags. Here is an example of Puppetfile: forge ' 'mod 'puppetfile: forge the Docker tags. Here is an example of Puppetfile to discover and add them in the Puppetfile. To make the code very dynamic, let's create variables for those in
variable s.tf: variable "user_name" { default = "changeme" description = "OpenStack username" } variable "tenant name" } variable "tenant name" { default = "hackme" description = "OpenStack username" } variable "tenant name" } variable "tenant name" { default = "hackme" description = "OpenStack username" } variable "tenant name" } variable "ten
Endpoint" } Don't forget to override the default values with your own in the terraform.tfvars file! user name tenant name password auth url = = = "***" "Now we're good to go. This means that Terraform could not detect any differences between your configuration and the real physical resources that exist. Type `cd apache` to enter it.
Both are interesting in terms of features or pricing. By reading the README file, it is stated that it needs two other cookbooks and Puppet modules Up till now, we've written our own cookbooks, which
are fairly simple in their current state. The shared folder contains everything that will last through the releases, including a cached copy of the current state. The shared folder contains everything that will last through the releases, including a cached copy of the current state. The shared folder contains everything that will last through the releases, including a cached copy of the current state.
to you and your team to decide what to follow: the recommended behavior from Chef or from Foodcritic. There're versions for most platforms; Debian, Red Hat-based systems, Ubuntu, and Windows. But even for such platforms; Debian, Red Hat-based systems, Ubuntu, and Windows. But even for such platforms; Debian, Red Hat-based systems, Ubuntu, and Windows. But even for such platforms; Debian, Red Hat-based systems, Ubuntu, and Windows. But even for such platforms; Debian, Red Hat-based systems, Ubuntu, and Windows. But even for such platforms; Debian, Red Hat-based systems, Ubuntu, and Windows. But even for such platforms; Debian, Red Hat-based systems, Ubuntu, and Windows. But even for such platforms; Debian, Red Hat-based systems, Ubuntu, and Windows. But even for such platforms; Debian, Red Hat-based systems, Ubuntu, and Windows. But even for such platforms; Debian, Red Hat-based systems, Ubuntu, and Windows. But even for such platforms; Debian, Red Hat-based systems, Ubuntu, and Windows. But even for such platforms are such platforms.
contributing to new open source projects, he's usually found hiking in remote places with his camera. At the time of writing, the supported CentOS 7, 6, and 5. Using OpenStack, we'll deploy a GitLab CE and two GitLab CI runners, using different storage solutions. Let's create a second "staging" VM, very similar to our production
setup, except we want to install the current latest Chef version (12.13.37 at the time of writing, feel free to use :latest instead): config.vm.network "private network", type: "dhcp" config.vm.provision :chef_solo do |chef| chef.add_recipe
 'apache2' end end Launch this new machine (vagrant up staging) and we'll see if our setup still works with the new major Chef version: \$ pidof httpd 13029 13028 13027 13026 13025 13024 So we can safely assume, as far as our testing goes
that the newest Chef version still works correctly with our production Chef code. Getting ready To step through this recipe, you will need: ff A working Docker installation ff A free account at Quay.io and/or a paid account at the Docker Registry
service after logging in using docker login. Storing the TDD infrastructure code locally 2. The knife command is never used on a Chef node. For this, we'll need to create a resource on AWS. As every cookbook version can live together, each environment can call its own version. $$ mkdir -p test/integration/default/serverspec 277 Testing and Writing
Better Infrastructure Code with Chef and Puppet Creating a ServerSpec helper script ServerSpec needs a minimum of two lines of configuration that must be repeated on each test. To start from the latest Node 7 image version, use this in a Dockerfile: FROM node:7 ENTRYPOINT ["/bin/bash"] 341 Working with Docker For the record, a node:7 image
will be around 650 MB, while node:4-slim will be around 205 MB. Using data sources, templates, and local execution When we deploy or update an infrastructure with Terraform, it's sometimes enjoyable to have some local content dynamically generated. Good! Our test has failed. Our objective is to install the Apache 2 web server on CentOS 7 (httpd
package), and start it. On Amazon Web Services, using the aws instance "vm" { ami = "ami-643d4217" instance type = "t2.micro" key name = "manual cloud init" user data argument to pass the cloud-config file content, and in this case, using the file() interpolation: resource "aws instance" vm" { ami = "ami-643d4217" instance type = "t2.micro" key name = "manual cloud init" user data
 "${file("cloud-config.yml")}" } The equivalent for a Digital Ocean VM is the user_data argument as well: resource "digitalocean_droplet" "vm" { image = "ubuntu-14-04-x64" name = "ubuntu" region = "ams3" size = "512mb" ssh keys = ["keys/admin key"] user_data = "${file("cloud-config.yml")}" } 169 Provisioning the Last Mile with Cloud-Init
Handling files using cloud-init An early need we all face is to have a file, a license, or a script in place right from the beginning of the instance life. Create the variable in variable in terraform.tfvars with the real token: auth token = "1234" description = "API Key Auth Token" } Also, be sure to override the variable in variable in variable in variable in variable in the value in terraform.tfvars with the real token: auth token = "1234" description = "API Key Auth Token" }
"JnN7e6tPMpWNtGcyPGT93AkLuguKw2eN" Creating a Packet project using Terraform Packet, like some other IaaS providers, uses the notion of project to group machines. In this recipe, we'll launch a CentOS 7.2 system, then fully update it so it's as secure as possible, install EPEL so we have a greater library of available packages, add the Puppet
 Labs Yum repository and install a Puppet agent, and add a different name server so our system is ready for the next step (which we won't cover here, as it's probably executing Puppet code). Tier 3 - the database T
 "aws s3 bucket object" "index" { bucket = "${aws s3 bucket.iac book.bucket}" key = "index.html" content = "Hello Infrastructure-as-Code Cookbook!" content: source = "index.html" 64 Chapter 2 If you navigate to the AWS S3 Console, you can
see it available with some extended information: It would be awesome if we could know easily the URL of our file right from Terraform, so we could give it to others. In this pattern: ff A role is a class defining a behavior (For example, a web server). In the variable with some extended information: It would be awesome if we could know easily the URL of our file right from Terraform, so we could give it to others. In this pattern: ff A role is a class defining a behavior (For example, a web server).
variable "log retention days" { default = "7" } Also, in a new cloudwatch.tf file, we can use the simple aws cloudwatch log group resource: resource "aws cloudwatch log group resource: resource aws cloudwatch.tf file, we can use the simple aws cloudwatch log group resource: resource aws cloudwatch.tf file, we can use the simple aws cloudwatch log group resource aws cloudwatch.tf file, we can use the simple aws cloudwatch.tf file, we
CloudWatch page, you'll see the newly created group under the Log Groups entry on the left (. If you're already using AWS on the command line, chances are you already have a set of standard environment variables: $ echo ${AWS_ACCESS_KEY_ID}$ $ echo ${AWS_DEFAULT_REGION}$ eu-west
1 If not, you can simply set them as follows: $ export AWS_ACCESS_KEY_ID="123" $ export AWS_SECRET_ACCESS_KEY="456" $ export AWS_DEFAULT_REGION="eu-west-1" Then Terraform can use them directly, and the only code you have to type would be to declare your provider! That's handy when working with different tools. Here, we'll
deploy an isolated blog container (Ghost) that will be publicly served by the nginxproxy container over HTTP. staging.yml file that will simply override the concerned values: wordpress: image: wordpress: 4.6 environment: WYSQL ROOT PASSWORD: s3cur3 The two
environment variables WORDPRESS_DB_PASSWORD and MYSQL_ROOT_PASSWORD will be overridden when docker-compose.yml -f docker-compose is executed with multiple configuration files taken in order: $ docker-compose is executed with multiple configuration files taken in order: $ docker-compose is executed with multiple configuration files taken in order: $ docker-compose is executed with multiple configuration files taken in order: $ docker-compose is executed with multiple configuration files taken in order: $ docker-compose is executed with multiple configuration files taken in order: $ docker-compose is executed with multiple configuration files taken in order: $ docker-compose is executed with multiple configuration files taken in order: $ docker-compose is executed with multiple configuration files taken in order: $ docker-compose is executed with multiple configuration files taken in order: $ docker-compose is executed with multiple configuration files taken in order: $ docker-compose is executed with multiple configuration files taken in order: $ docker-compose is executed with multiple configuration files taken in order: $ docker-compose is executed with multiple configuration files taken in order: $ docker-compose is executed with multiple configuration files taken in order: $ docker-compose is executed with multiple configuration files taken in order: $ docker-compose is executed with multiple configuration files taken in order: $ docker-compose is executed with multiple configuration files taken in order: $ docker-compose is executed with multiple configuration files taken in order: $ docker-compose is executed with multiple configuration files taken in order: $ docker-compose is executed with multiple configuration files taken in order: $ docker-compose is executed with multiple configuration files taken in order: $ docker-compose is executed with multiple configuration files taken in order: $ docker-compose is executed with multiple configuration files taken in order: $ docker-compose is executed with mult
docker exec -it 1basics_mysql_1 mysql -uroot -ps3cur3 Welcome to the MariaDB monitor. Uncompress the Starter/recipes/default.rb inflating: chef-repo/cookbooks/starter/recipes/default.rb infl
repo/cookbooks/starter/attributes/default.rb inflating: chef-repo/cookbooks/starter/metadata.rb inflating: chef-repo/cookbooks/starter/templates/default/sample. To access the report summary, just click on the tag number: This layer has clear issues! But don't follow this blindly
and double-check the said vulnerabilities. Enter the name of the book in the Search box. io/#FC033) is about missing templates. In this chapter, we'll discover how to use Terraform to bootstrap a fully capable infrastructure on AWS. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. Docket
support is good with Travis, and it works out of the box. In this case, setting the flannel network is done by writing the key/value combination to etcd under /coreos.com/network/config: coreos: units: - name: flanneld.service drop-ins: - name: 50-network-config.conf 183 Provisioning the Last Mile with Cloud-Init content: | [Service]
 ExecStartPre=/usr/bin/etcdctl set /coreos.com/network/ config '{ "Network": "10.1.0.0/16" }' This will simply create the file /etc/systemd/system/flanneld.service.d/50-network-config.conf [Service] ExecStartPre=/usr/bin/etcdctl set /coreos.com/network/config '{ "Network": "10.1.0.0/16" }' This will simply create the file /etc/systemd/system/flanneld.service.d/50-network-config.conf [Service] ExecStartPre=/usr/bin/etcdctl set /coreos.com/network/config '{ "Network": "10.1.0.0/16" }' This will simply create the file /etc/systemd/system/flanneld.service.d/50-network-config.conf [Service] ExecStartPre=/usr/bin/etcdctl set /coreos.com/network/config '{ "Network": "10.1.0.0/16" }' This will simply create the file /etc/systemd/system/flanneld.service.d/50-network-config.conf [Service] ExecStartPre=/usr/bin/etcdctl set /coreos.com/network/config '{ "Network": "10.1.0.0/16" }' This will simply create the file /etc/systemd/system/flanneld.service.d/50-network-config.conf [Service] ExecStartPre=/usr/bin/etcdctl set /coreos.com/network-config.conf [Servi
"10.1.0.0/16" }' Verify we have a correct flannel0 interface in the correct IP network range: $ ifconfig flannel0 flannel0 flannel0 flannel0 flannel0 interface in the 10.1.0.0/16 network: $ docker run -it --rm alpine ifconfig eth0 eth0 Link encap: Ethernet
HWaddr 02:42:0A:01:13:02 inet addr:10.1.19.2 Bcast:0.0.0.0 Mask:255.255.255.0 [...] It's all working great! Note that it may take a while to get the interface up, depending on the host Internet connection speed, as flannel is running from a container that needs to be downloaded first (51 MB to date). This translates into a simple Chef resource with an account of the interface up, depending on the host Internet connection speed, as flannel is running from a container that needs to be downloaded first (51 MB to date). This translates into a simple Chef resource with an account of the interface up, depending on the host Internet connection speed, as flannel is running from a container that needs to be downloaded first (51 MB to date).
array of actions: service "service name" do action [:enable, :start] end Enabling and starting Apache service resource to the apache/recipes/default.rb file, just after the package resource: service "httpd" do action [:enable, :start] end Bump the cookbook version number on the apache/metatada.rb file: version '0.2.0' This way, the Chef
server will always keep version 0.1.0 with only the package installation, and a new version 0.2.0, with the service support. On the target host, launch the Chef client need as a service support. On the target host, launch the chef client one last time to make it deploy itself as a service service support. On the target host, launch the Chef client need to make it deploy itself as a service servi
service[chef-client] action start - start service[chef-client] * ser
an S3 backend (or Swift if we have an internal OpenStack). Now suppose we want to use our previous example, using the code we made for Apache, and the official Puppet Labs MySQL package. We'll see how to send files using plain text and base64 data encodings. html ff 236 Puppet Hiera: Chapter 6 Creating functional roles Until now, we have
created cookbooks based on a particular technology. Instead of using the us-east-1 item from the aws data bags/aws/euwest-1. json: { "id": "eu-west-1", "aws access key": "a secret key" } As we know, the data will be encrypted
for each and every running node's public key. 178 Chapter 5 Configuring etcd using cloud-init The etcd key value store is used in CoreOS to share multiple configuration data between members of a same cluster. We know from the documentation that port 5000 is exposed by the registry server, so we need to forward it to our host to use it locally. It's
pretty useful to quickly and easily simulate production environments, create better architectures, and increase container exposure on the network front. In-depth knowledge of systemd to not think the unit has crashed because the command
exited (RemainAfterExit=yes), and one that tells the unit type is not executing a long running process, but instead a short one that should exit before continuing (Type=oneshot). This cookbook, for some reason, doesn't work with ChefSpec, so when converged, it errors out the following string: chefspec not supported!. Our EPP template will use two
parameters: the site name and the document root. xi 1 Vagrant Development Environments In this chapter, we will cover the following recipes: ff Adding an Ubuntu Xenial (16.04 LTS) Vagrant box ff Using a disposable CentOS 7.x with VMware in
 seconds ff Extending the VMware VM capabilities ff Enabling multiprovider Vagrant environments ff Customizing a Vagrant vM ff Using Docker with AWS EC2 and Docker ff Simulating dynamic multiple host networking ff Simulating a networked
three-tier architecture app with Vagrant ff Showing your work on the LAN while working with Laravel ff Sharing access to your Vagrant to create a Docker host ff Using Docker containers on CoreOS with Vagrant 1 Vagrant Development
 Environments Introduction Vagrant is a free and open source tool by Hashicorp aimed at building a repeatable development environment inside a virtual machine, using simple Ruby code. Here's how it works: .DEFAULT GOAL := help help: @grep -E '^[a-zA-Z -]+:.*?## .*$$ (MAKEFILE LIST) | sort | awk 'BEGIN {FS = ":.*?## "}; {printf
 "\033[36m%-30s\033[0m %s", $$1, $$2}' Now simply use this feature to create an entry for the validate syntax and terraform-fmt: $(TERRAFORM_BIN) fmt -list terraform-validate: $(TERRAFORM_BIN) validate 111 Going Further with Terraform
If you simply type make, you'll get an automatic help: $ make validate will both validate will be w
buildable Dockerfile. We'll then configure the Chef client to authenticate securely against the Chef Server organization, and finally apply an initial set of cookbooks. The package we installed created symlinks from this directory to /usr/bin which is on the $PATH: $ ls -al /usr/bin/chef lrwxrwxrwx. In the recipe to install the Apache HTTPD server, we
requested it to be enabled and running. 114 Chapter 3 Now import this new empty repository on your workstation, in a dedicated folder: $ git clone Cloning into 'my_infrastructure_code'... Managing dynamic files and directories from a template Let's now create a generic Apache virtual host to fully control what we'll do with our LAMP server, and not
just live with the default configuration shipped with our Linux distribution. To properly use this dynamically configured container, use environment variables as required: $ docker run --rm -h ssh-container e USER=john -e PASSWORD=s3cur3 -p 22:22 ssh Creating user john Password for 'john' changed Fixing permissions for user john [1] Nov 29
23:02:02 Not backgrounding Now try connecting to this container from another terminal or container with proper credentials: $ ssh [...] s password: ssh-container with proper credentials:
access to someone needing, for example, shared volume storage access or similar usages. Total 9 (delta 0), reused 0 (delta 0) To :sjourdan /my_infrastructure set up to track remote branch new_infrastructure from origin. From development
to production, a common tooling and approach emerged—often borrowed from developers and the agile techniques. Restart the Vagrant environment and see the result: $ curl Hello from Docker via Vagrant On SELinux-enabled systems you may need to do some configuration that's beyond the scope of this book. To start with the latest CentOS 7
available, in a Dockerfile execute the following: FROM centos:7 ENTRYPOINT ["/bin/bash"] Starting from a Red Hat Enterprise Linux (RHEL) image Red Hat also ships containers for RHEL. Getting ready To step through this recipe, you will need the following: ff Access to a cloud-config enabled infrastructure ff A working Chef Server and
organization setup How to do it... Everything related to Chef with cloud-init is configured under the directive named chef. com/hashicorp/hcl. Now we'll be able to execute rspec in our local context using Bundler: $ bundle exec rspec Initializing the tests Let's start by creating our first Docker Rspec test that will just, for now, initialize the libraries we
need and build the Docker image before anything else. We'll create an app (a simple GitHub Hubot: http:// hubot.github.com/), but feel free to use your own. See also ff The Chef service resource documentation: resource documentation: resource documentation resource documentation.
documentation: puppet/4.8/types/service.html Managing files, directories, and templates A very useful Chef feature is the ability to manage files right from the Chef code. 11 Vagrant Development Environments Customizing a Vagrant VM Vagrant supports many configuration options through the Vagrant files, directories, and templates A very useful Chef feature is the ability to manage files right from the Chef code. 11 Vagrant Development Environments Customizing a Vagrant VM Vagrant Supports many configuration options through the Vagrant Development Environments Customizing a Vagrant VM Vagrant Supports many configuration options through the Vagrant Development Environments Customizing a Vagrant VM Vagrant Supports many configuration options through the Vagrant Supports many configuration options the Vagran
the nodes will be the same as that of the master. We'll send our Tomcat logs to it right away! Getting ready To step through this recipe, you will need: ff A working Docker installation ff An AWS account How to do it... To use AWS CloudWatch Logs, we need at least one log group. ==> srv-1: Machine is booted and ready for use! [...] ==> srv-1:
docker version [...] ==> srv-1: Server: ==> srv-1: Server: ==> srv-1: Creating vagrant_app_1 ==> srv-1: Creating vagrant_front_1 Open your browser at (using the EC2 instance public IP) and you'll see your Ghost blog behind an NGINX reverse proxy, using Docker containers, using Vagrant on Amazon EC2. What if it's important for us
to know at a glimpse under what security groups our EC2 instance is running? You probably noticed this in Chapter 6, Fundamentals of Managing Servers with Chef and Puppet; we created the code in the /etc/puppetlabs/code/environments/ production directory. Maybe you have an issue configuring something and you'd like some remote help from
your coworker on the other side of the planet. Here, it just tells VMware to allocate two CPUs and 1 GB of RAM to the virtual machine it's launching the way you would have done manually from inside the software. Setting up chef (12.14.89-1) ... First of all, we need to define the data in Hiera. Of course, if there are many nodes in your infrastructure,
you should create a module containing these Puppet resources and include it for each node. lukasmartinelli.ch/) maybe the most used linter, while Project Atomic's dockerfile lint project is perhaps the most complete one (dockerfile lint). They currently all share the same main key, which is a disaster if a leakage happens. Auto-configuring dynamic
 containers We can't always execute a binary to get what we want. vi Preface Chapter 9, Working with Docker, is about using Docker containers from a developer point of view—choosing a base image, optimizations, tags, versioning, deploying Ruby-on-Rails or Go applications, networking, security, linting, and using our own durable private registry—
all using simple Docker instructions—as code. Similarly, it not only sends the logs to journald, but also supports the Graylog or Logstash GELF log format. Our next task is to test if everything is still running smoothly after an upgrade to the new version 12. Let's declare a cookbook requirement to it, specifically to Berksfile: cookbook 'chef-client', '~>
7.0.0' To the common/metadata.rb file, add the dependency: depende 'chef-client' Using the Chef client as a daemon The documentation tells us that including the default recipe will automatically detect the host platform and configure chef-client as a daemon. ... Getting ready To step through this recipe, you will need: ff A working
Docker installation ff A working Ruby environment (including Bundler) How to do it... Our goal is to create an NGINX container following TDD principles. Disable new box version check at startup You may be using a slow internet connection, or you know you do want to use your current installed box, or maybe you're in a hurry and just want to get the
job done; you can just remove the option to check for a new version of the box at startup by adding the following: config.vm.box check update = false Use a specific box version If you know you want to use a specific version of the box (maybe for debugging purposes or compliance) and not the latest, you can simply declare it as follows:
config.vm.box version = "2.2.9" 12 Chapter 1 Display an information to the user A useful feature is to display some basic but relevant information to the user I Display an information information to the user I Display an information to the user I Display an information to the user I Display an information information information I Display an information I Display and I Display and I Display and I Display an information I Display and I Display
new EC2 instance, we use a connection {} block inside the aws instance resource to tell it which user and key to use through SSH: connection { type = "s$h" user = "${var.assh user}" key file = "${var.assh user}" key file = "$fvar.assh user}" key
plan has been generated and is shown below. [...] ==> srv-1: You're sharing with SSH access. Getting ready To step through this recipe, you will need: ff A working Docker installation 399 Maintaining Docker Containers How to do it... So let's write it in
roles/database.rb: name "database" description "A database server for our application" run list "recipe[mariadb]" 237 Fundamentals of Managing Servers with Chef and Puppet 3. While virtual machines include everything needed to run a full operating system like memory, CPU, a kernel and all required libraries, a container is much more lightweight
Another good example is when the Dockerfile downloads a compressed archive; downloading it, uncompressing it, and then removing the archive uses a lot of added layer space when done separately. To connect to the instance, we know the default username is centos, but as it can evolve or you may use your own images, it's better to fix it in a
 variable as well: variable "ssh user" { default = "centos" } Creating the EC2 instance We know from previous recipes that a basic instance running CentOS looks like this in Terraform's instance "centos" { ami = "${lookup(var.aws centos ami, var.aws region)}"
instance_type = "${var.aws_instance_type}" key_name = "${aws_key_pair.admin_key.key_name}" associate_public_ip_address = true tags { Name = "CentOS-${count.index+1} by Terraform" } } Now we need to provide two kinds of information to our Terraform file: what to
do with Chef on the server and how to connect to it. We'll all spend better nights and weekends if we know that all the system profiles are being continuously rebootstrapped successfully; in fact, why not use the CI system every night so every morning we would know whether the previous day's changes have impacted something. Are those packages
really installed? If you need to do only a dry-run, you can use the --noop option combined with -test. After this first step, we'll download the Chef Start Kit, an archive containing a fully working Chef repository, with a sample role and cookbook we can use right away—and that's what we'll do by sending this sample cookbook to the server using our first
knife command. Puppet has its own Domain Specific Language (DSL). So, Chef adapted their own version of Rubocop, named Cookstyle. See also ff 190 For more information about systemd socket activation refer to . We obviously can write our own plugins, but that's way beyond the scope of this book. Add the following code to a file named ecr.tf:
resource "aws ecr repository" "myapp" { name = "myapp" } If you want to know the URL to access your new repository, was ecr repository, myapp.repository, was ecr repository, was ecr repository.
create private repositories. It would be great to have the plan and apply commands as well, and if you followed the recipe on environment management with Terraform, that would be awesome if it worked right from the Makefile, we'd save a lot of time. Deploying to production Deploying to production at this stage is totally similar to shipping to
staging; there's no difference. Finished setting up (0m0.00s). Using your favorite editor, create this very minimal Vagrantfile to launch an ubuntu/xenial64 box: Vagrant.configure("2") do |config| config.vm.box = "ubuntu/xenial64" end 3. Getting ready To work through this recipe, you will need the following: ff An Internet connection ff A working Chef
DK installation on the workstation How to do it... Follow these steps for Creating a free hosted server: 1. 113 Going Further with Terraform A simple workflow I propose to use here for our infrastructure-as-code work is based on what's called the GitHub Flow (: Getting ready To step through this recipe, you will
need the following: ff An account on some Git hosting (self-hosted or commercial) ff A working Terraform (refer to the Chapter 2, Provisioning IaaS with Terraform recipes) ff An Internet connection How to do it... Start by creating a new repository for use with your team.
Terraform would have asked us for a value when executed: $ terraform code to configure our AWS region Enter a value: There's more... We've used values directly inside the Terraform CloudWatch Logs is a log
 aggregation service by Amazon you can use to send your logs to. A pull request is often composed of multiple commits, so a summary is more than welcome: Now everyone from your team have access to your work and can discuss it right from GitHub if necessary: 119 Going Further with Terraform A few minutes later, one of your coworkers reviews
 your code and sends you a remark: She might be right; let's find out with the Terraform formatter: $ terraform fmt provider.tf Looks like there was a formatting issue! Use git diff to see what's the difference: $ git diff diff --git a/provider.tf b/provider.tf b/provider.tf index 59cdf2a..b54eb94 100644 --- a/provider.tf +++ b/provider.tf @@ -1 +1,3 @@ -provider "aws"
 \{ \text{region} = \text{start with our previous Vagrantfile including customizations} \ + \text{provider "aws"} \ + \text{region} = \text{start with our previous Vagrantfile including customization} \ + \text{provider "aws"} \ + \text{provider "a
config.vm.provider vmware do |v| v.vmx["numvcpus"] = "2" v.vmx["numvcpus"] = "2" v.vmx["numvcpus"] = "1024" end end How would we add the same configuration on VirtualBox as we have on VMware? This way we'll be able to manage any available service. Let's check for an environment variable named env, passed at make execution (such as make plan env=staging).
 and returns an error if not set: ifndef env getenv=$(error var:"env=" is not set) else getenv=$(env) endif Now we can write what terraform-plan and terraform-plan an
apply: $(TERRAFORM_BIN) apply -state=$(call getenv). tfstate -var environment=$(call getenv). By the way, you can add support for environments to our previous terraform-validate example: terraform-v
realize this simple tool helps so much. In this section, we'll start with a basic Dockerfile found earlier and end up with a fully double-checked linted file. For now, we have to duplicate the block, but that may very well change in a newer Terraform version. NAME DESCRIPTION AUTHOR KEYWORDS puppetlabs-mysql Installs, con... type: foss means
that the open source edition of Puppet will be used. 343 Working with Docker How to do it... Take the following Dockerfile, which updates the Ubuntu image, installs the apache2 package, and then removes the /var/lib/apt cache folder. This list of users has a set of keys, such as groups the user is a member of, sudo rights, which shell to default to, or
an SSH public key to authorize. We can also explicitly specify which CoreOS channel to use (stable, beta, or alpha). 95 Going Further with Terraform If you're connecting to a remote Docker provider (maybe in provider.tf). Here are the goals: ff For the apache/manifests/init.pp manifest: The unit test needs to
validate the manifest is compiling, the apache2 service is running and activated on boot. Instead of messing with the code, we can test the impact of changing that value directly from the command line: $ terraform plan -var 'cluster size="2" [...] + aws instance.coreos.1 ami: "ami-85097ff6" instance type
"t2.micro" tags.Name: "coreos 2" [...] Plan: 1 to add, 0 to change, 0 to destroy. Let's create a project named Docker Swarm Bare Metal Infrastructure" } This way, if you happen to manage multiple projects or
customers, you can split them all into their own projects. Here comes the scratch image: it just doesn't exist. We need to inform r10k which modules are local. All of this information shouldn't be hard to get, but they are precious for building quality containers. WordPress can now communicate with your database. Golang and containers are two
technologies that go perfectly well together, and shipping or managing infrastructures using Go programs is a breeze. Add the following on the mariadb/recipes/default.rb file: package "mariadb" do action :install end package "mariadb end action :install end package "mariadb" do action :install end package "mariadb" do action :install end package "mariadb" do action :install end action :install end package "mariadb" do action :install end packa
%w(mariadb mariadb-server).each do |name| package name do action :install end end Upload the cookbook to the remote node's run list from your workstation: $ knife cookbook upload mariadb Run the Chef client on the remote host: $ sudo chef-client Starting Chef
Client, version 12.15.19 208 Chapter 6 resolving cookbooks for run list: ["apache", "mariadb"] Synchronizing Cookbooks: - apache (0.1.0) - mariadb (0.1.0) - mariadb * yum_package[mariadb * yum_package[mariadb * yum_package[mariadb] action install - install version 5.5.50-1.el7_2
of package mariadb-server [...] Chef Client finished, 2/3 resources updated in 25 seconds Verify that the MariaDB package is correctly installed: $ which mysql /usr/bin/mysql $ mysql --version mysql Ver 15.1 Distrib 5.5.50-MariaDB, for Linux (x86_64) using readline 5.1 Creating a PHP cookbook Let's reuse our knowledge to create a cookbook that
will install the packages needed for PHP support: $ chef generate cookbooks/php/recipes/default.rb file: %w(php php-mysql).each do
|name| package name do action: sknife node run list add vagrant php cookbook from your workstation: sknife node run list add vagrant php Run the Chef client on the remote node: sknife node run list add vagrant php Run the Chef client on the remote node: sknife node run list add vagrant php Run the Chef client on the remote node: sknife node run list from your workstation: sknife node run list add vagrant php Run the Chef client on the remote node: sknife node run list from your workstation: sknife node run list add vagrant php Run the Chef client on the remote node: sknife node run list from your workstation: sknife node run list from your workstation: sknife node run list from your workstation: sknife node run list add vagrant php Run the Chef client on the remote node: sknife node run list add vagrant php Run the Chef client on the remote node: sknife node run list add vagrant php Run the Chef client on the remote node: sknife node run list add vagrant php Run the Chef client on the remote node: sknife node run list add vagrant php Run the Chef client on the remote node: sknife node run list add vagrant php Run the Chef client on the remote node run list add vagrant php Run the Chef client on the remote node run list add vagrant php Run the Run the
sudo chef-client $ php --version PHP 5.4.16 (cli) (built: Aug 11 2016 21:24:59) We now know the very basics of deploying cookbooks and installing packages on a remote node, using Chef! There's more... Using Puppet, a package installation is done using the package resource directive. Maintaining consistent systems using scheduled convergence
Once initially deployed and configured, it's hardly imaginable to let our systems be manually updated afterwards by logging in to each host and launching the chef-client command. For example, Ubuntu official releases only provide VirtualBox images. srv-1: Installing Ansible... If you're using Berkshelf to manage dependencies, don't forget to add the
cookbook where required (either metadata.rb or Berksfile). The workaround is to deactivate in the web server configuration sendfile off; Using Apache, it is as follows: EnableSendfile off; Using Apache, it is as follows: Enabl
like to show the status of the job to your customer who's located in another city. Deploying Chef Client from start to finish using cloud-init. It also covers unit and integration testing, linters, and tools for Chef and Puppet, so that you'll produce the best infrastructure code
possible. We'll create an initial security group allowing ingress Secure Shell (SSH) traffic only for our own IP address, while allowing all outgoing traffic. How to do it... Before starting to code anything using Terraform, ensure you can connect to any kind of Docker Engine, local or remote: $ docker version Client: Version: 1.12.0 API version: 1.24 Go
 version: go1.6.3 Git commit: 8eab29e Built: Thu Jul 28 21:15:28 2016 OS/Arch: darwin/amd64 Server: Version: 1.12.0 API version: 1.24 Go version: go1.6.3 Git commit: 8eab29e Built: Thu Jul 28 21:15:28 2016 OS/Arch: darwin/amd64 Server: Version: 1.12.0 API version: 1.24 Go version: go1.6.3 Git commit: 8eab29e Built: Thu Jul 28 21:15:28 2016 OS/Arch: darwin/amd64 Server: Version: 1.12.0 API version: 1.24 Go version: go1.6.3 Git commit: 8eab29e Built: Thu Jul 28 21:15:28 2016 OS/Arch: darwin/amd64 Server: Version: 1.12.0 API version: 1.12.0 API version: 1.12.0 API version: 1.12.0 API version: go1.6.3 Git commit: 8eab29e Built: Thu Jul 28 21:15:28 2016 OS/Arch: darwin/amd64 Server: Version: 1.12.0 API version: go1.6.3 Git commit: 8eab29e Built: Thu Jul 28 21:15:28 2016 OS/Arch: darwin/amd64 Server: Version: 1.12.0 API v
seconds to load) 5 examples, 0 failures ChefSpec Coverage report generated... This is done using the search (in our case, everything). Libraries are still libraries, and security fixes are pushed every day into distributions package repositories. Using our previous LAMP setup with Vagrant, let's
try to encrypt the root password for MySQL. Now the infrastructure is created, if you run a plan again, it will say nothing should be modified: $ terraform plan 99 Going Further with Terraform Refreshing Terraform state in-memory prior to plan... We'll see different ways to expose ports, create new networks, execute Docker containers inside them,
and even have multiple networks per container. Here, we'll launch the HTTP/2 Caddy webserver hosted in the CoreOS account on the Quay.io/coreos/caddy Activating privacy features... Amongst the many deployable features, we'll start by simply deploying a single registry ready to be
load-balanced, and then we'll switch its backend storage to AWS S3, so disk space will never be an issue again. So here's how it looks in spec/unit/recipes/default attributes on CentOS 7.2' do let(:chef run) do runner = ChefSpec::ServerRunner.new(platform: 'centos',
version: '7.2.1511') runner.create data bag('webusers', { 'john', 'htpasswd' => '$apr1$AUI2Y5pj$0v0PaSlLfc6QxZx1Vx5Se.' } }) runner.converge(described recipe) end it 'converges successfully' do expect { chef run }.to not raise error end end 265 Testing and Writing Better Infrastructure Code with Chef and Puppet Now
the simulated Chef run has a webusers data bag and some sample data to work with! Testing recipes inclusion It's very common to include recipes inside another recipe. A workaround is to use the static website { index document = "index.html" } An
optional output will give you its static hosting URL (in our case, iac-book.s3-websiteeu-west-1.amazonaws.com instead of: output "S3 Endpoint" { value = "${aws s3 bucket.iac book.website endpoint}" } 65 Provisioning IaaS with Terraform There's more... Using Ansible, there are many ways to create a bucket. Good news! It looks like increasing the
cluster_size value has the intended effect: creating a new instance. This is how it translates in the cloud-config file: #cloud-config file: #cloud-config users: - name: "jane" gecos: "Jane Docker" groups: - "docker" ssh-authorized-keys: - "ssh-rsa AAAAB[...] jane" In the final system, Jane is able to log in using her private key, and interact with the docker daemon as she's a
member of the docker group: ~ $ docker ps CONTAINER ID IMAGE STATUS PORTS COMMAND NAMES CREATED The SSH public key ends up in the following file: ~ $ cat .ssh/authorized keys.d/coreos-cloudinit ssh-rsa AAAAB [..] jane Setting the timezone on CoreOS using cloud-init CoreOS uses a system built
around NTP (short for Network Time Protocol), controlled by the timedatectl command. We'd like this page to dynamically display something in 192.168.146.129 is up since 25 minutes 55 seconds All the information we need is stored somewhere in ohai: platform,
platform version, os, os version, ipaddress, or uptime are all valid values. The complete documentation can be found in the code bundle. Like this: $ cat hosts [centos7 hosts] 52.17.172.231 It's correctly populated! Apply a configured Ansible Our code repository is now ready for use by Ansible. Now add the instance type in the same file: variable.
 "aws instance type" { default = "t2.micro" description = "Instance Type" } 86 Chapter 3 3. Integrate this in the company's Continuous Integration (CI) system and this will stop. That's how all resources in Terraform work. Getting ready To step through this
recipe, you will need: ff Access to a cloud-config enabled infrastructure How to do it... We'll get through the most important configuration options that can be manipulated for CoreOS. amazon.com/iam/home#policies. We'd probably choose a better way to deploy it programmatically though, such as Ansible or Chef, through a Docker container or not:
$ curl -s | sudo bash Here's how to get an htop-like view of all the running containers on the system: $ sudo csysdig --view=containers 404 Chapter 10 Navigating to the F2/Views menu helps you enter many different options to see what's running, from processes to syslog to open files and even the Kubernetes, Marathon, or Mesos integration.
Merging conflicts in a state file is really not something nice to do. 3. Initial infrastructure switched to a new branch 'new_infrastructure switched to a new branch to work on an initial infrastructure switched to a new branch to work on an initial infrastructure switched to a new branch 'new_infrastructure switched to a new branch to work on an initial infrastructure switched to a new branch 'new_infrastructure switched
 infrastructure automation consultancy (, a web radio (http:// phauneradio.com/), a container/serverless platform for developers (. Let's start with a single node infrastructure, with only a Puppet server. The program will ask you for a password you'll need to connect to the box remotely: $ vagrant share --ssh ==> srv-1: Detecting network information
for machine... Generating an empty Apache cookbook Let's start by creating an empty cookbook from inside the Chef repository cookbooks $ chef-repo/cookbooks $ chef-repo/cookboo
exist, create it by executing the following command: $ mkdir -p spec/unit/recipes 259 Testing and Writing Better Infrastructure Code with Chef and Puppet In this recipes directory in spec/unit are found the ChefSpec unit tests, typically: $ tree spec/
gets its matching ChefSpec file. To make it short, this creates a systemd service unit along with the socket description. There's more... Now that we have a LAMP server with Puppet, let's create a virtual host! Our goals are as follows: ff Removing the default virtual host provided by
Ubuntu ff Creating our own virtual host, with a specific DocumentRoot and dedicated log files ff Deploying a simple PHP page displaying the result of the function phpinfo() These three operations will be done using the file directive. Let's create a Makefile that will take care of the prerequisites for us: test: bats bats: build bats test build: docker-
compose build Now when you execute the make test command, it will launch the bats suite, which itself depends on building the image by docker-compose build Building http Step 1: FROM debian:stable-slim ---> d2103c196fde [...] Successfully built
1c4f46316f19 bats test Y apt lists are empty Y Apache version is correct 2 tests, 0 failures 373 Maintaining Docker containers might have a simpler language, but in the end, general concepts remain common and still apply. kitchen
converge: This provisions the instance with the suite information from the .kitchen.yml we created. It's also recommended to use a dedicated Terraform user for AWS, not the root account keys: aws access key = "< your AWS SECRET KEY >" Now, let's tie all this together into a single file,
provider.tf: provider "aws" access_key = secret_key = region = } { "${var.aws_access_key}" "${var.aws_region}" Apply the following Terraform code: $ terraform apply Apply complete! Resources: 0 added, 0 changed, 0 destroyed. Sysadmins are used to patch the systems; however, now it's a total anti-pattern to update
running container. Getting ready To step through this recipe, you will need the following: ff A working Terraform installation ff An AWS provider configured in Terraform ff Generate a pair of SSH keys somewhere you remember, for example, in the keys folder at the root of your repo: $ mkdir keys $ ssh-keygen -q -f keys/aws_terraform -C
aws terraform ssh key -N ' ff An Internet connection How to do it... The resource we want for this is named aws key pair. In this case, you can specify a dedicated Vagrantfile = "docker host/Vagrantfile" end 15 Vagrant Development Environments Using Docker in
Vagrant for a Ghost blog behind NGINX Vagrant in Docker can be used more usefully to simulate traditional setups such as an application behind a load balancer or a reverse proxy. Infrastructure code is still code, so it should be no different than software code; the same principle should apply. It consists of an e-mail address and an API key. The
reasons are multiple, such as no fix is available currently, the vulnerability doesn't concern the platform we're running on, and so on. We want all available interfaces on the default port (TCP/2379). Execute this to launch dockerfile lint from the official Docker image: $ docker run -it --rm -v $PWD:/root/ projectatomic/dockerfile lint dockerfile lint from the official Docker image: $ docker run -it --rm -v $PWD:/root/ projectatomic/dockerfile lint from the official Docker run -it --rm -v $PWD:/root/ projectatomic/dockerfile lint from the official Docker run -it --rm -v $PWD:/root/ projectatomic/dockerfile lint from the official Docker run -it --rm -v $PWD:/root/ projectatomic/dockerfile lint from the official Docker run -it --rm -v $PWD:/root/ projectatomic/dockerfile lint from the official Docker run -it --rm -v $PWD:/root/ projectatomic/dockerfile lint from the official Docker run -it --rm -v $PWD:/root/ projectatomic/dockerfile lint from the official Docker run -it --rm -v $PWD:/root/ projectatomic/dockerfile lint from the official Docker run -it --rm -v $PWD:/root/ projectatomic/dockerfile lint from the official Docker run -it --rm -v $PWD:/root/ projectatomic/dockerfile lint from the official Docker run -it --rm -v $PWD:/root/ projectatomic/dockerfile lint from the official Docker run -it --rm -v $PWD:/root/ projectatomic/dockerfile lint from the official Docker run -it --rm -v $PWD:/root/ projectatomic/dockerfile lint from the official Docker run -it --rm -v $PWD:/root/ projectatomic/dockerfile lint from the official Docker run -it --rm -v $PWD:/root/ projectatomic/dockerfile lint from the official Docker run -it --rm -v $PWD:/root/ projectatomic/dockerfile lint from the official Docker run -it --rm -v $PWD:/root/ projectatomic/dockerfile lint from the official Docker run -it --rm -v $PWD:/root/ projectatomic/dockerfile lint from the official Docker run -it --rm -v $PWD:/root/ projectatomic/dockerfile lint from the official Docker run -it --rm -v $PWD:/root/ projectatomic/dockerfile lint from the offic
certain amount of suggestions will arise (errors, warnings, and info), each with a related reference URL to refer to. Containers are real systems, so they too can have users. To do so, create a new recipe named aws inside the mysite cookbook: $ chef generate recipe aws Don't forget to bump the cookbook version and environment constraints
accordingly. We'll end up by showing how to do something similar with docker-compose. So if you're more familiar with Docker Compose, or if you'd like to do something similar with this tool, here's what the code would look like in the dockercompose. Yn Ifle: version: '2' services: 18 Chapter 1 front: image: nginx:stable volumes: -
 "./nginx.conf:/etc/nginx/conf.d/default.conf" restart: always ports: - "80:80" depends_on: - app links: - app app: image: ghost:0.9.0 restart: always Remember that with Vagrant, you can mix virtual machines and Docker containers, while you can't with docker-compose. Great! It automatically downloaded our dependencies. It's both a public Docker
registry service (with private or public containers, depending on your subscription) and a Docker image build service that can automatically create new images when changes occur in the code. The current folder is itself a symlink to a specific release, which is the git commit SHA on GitHub (72606bed348e61b6f98318cf920684765aa08b37). There're
plugins for almost anything you can think about, and most of them are community supported. Let's find out by adding the following to an htaccess_spec.rb file: describe file('/etc/httpd/htpasswd') do it { should be_grouped_into 'root' } end
 Testing for repository existence Our mysite cookbook to deploy MySQL, and that includes adding a yum repository. This way, the role for mysite will be enough to run our infrastructure, from a functional point of view. If you find any
errata, please report them by visiting selecting your book, clicking on the Errata Submission Form link, and entering the details of your errata. Fortunately, there's a managed IAM policy for that named CloudWatchLogsFullAccess. 376 Chapter 10 TDD - installing the NGINX package Our next objective is to install the nginx package. Start by bumping
the cookbook's version in metadata.rb to a minor version: version '0.3.1' 7. Let's fix this and automatically format our files correctly: $ terraform fmt provider.tf variables.tf Our two files are now correctly format our files correctly: $ terraform fmt provider.tf variables.tf Our two files are now correctly format our files correctly: $ terraform fmt provider.tf variables.tf Our two files are now correctly format our files correctly: $ terraform fmt provider.tf variables.tf Our two files are now correctly format our files are now correctly format our files are now correctly.
before reaching the CI, it's even better if it's in the project's Makefile. Encrypted data bags with a shared secret The solution to use an encrypted data bag is to send it encrypted from our workstation. Starting from a Golang image Go is well distributed as a Docker image. We know we want to start from an Ubuntu 16.04, then update the APT base,
 and finally install our utilities. 76 Chapter 2 So it all looks good and secure! Mary can do her job on S3 safely. Choose from the drop-down menu where you purchased this recipe, you will need the following: ff A working Chef DK installation on the
workstation 223 Fundamentals of Managing Servers with Chef and Puppet ff A working Chef client configuration on the remote host ff Optionally, the Chef code from the previous recipes How to do it... We know we want to create a new cookbook named mysite so we can centralize everything related to making this application work in the same place
Why Subscribe? The first one will update Ubuntu (update os.sh) while the second will install Docker (install docker.sh). It's an easy application ready to use on Heroku. To specify a default VM: config.vm.define "prod", primary: true do |config| [...] end To not start automatically a VM when issuing the vagrant up command: config.vm.define "staging"
autostart: false do |config| [...] end Berkshelf and Vagrant Chances are, if your production environment is using Berkshelf for dependency management and not 100% local cookbooks (if you aren't, you should!). pp.nWPdZJ {:ignore => } localhost $ scp /var/folders/k9/7sp85p796qx7c22btk7 tgym0000gn/T/ beaker20161103-
41882-73vqlb ubuntu-1604-x64:/tmp/apply manifest.pp.0]ht7j {:ignore => } should be listening File "/etc/apache2/sites-available/www.sample.com.conf" content should match /DocumentRoot \var\www\docroot/ File "/etc/apache2/sites-available/www.sample.com.conf" content should be running Port "80" should be running Port "80" should be running File "/etc/apache2/sites-available/www.sample.com.conf" content should match /DocumentRoot \var\www\docroot/ File "/etc/apache2/sites-available/www.sample.com.conf" content should be running Port "80" should be running Por
enabled/www.sample.com.conf" should be symlink File "/etc/apache2/htpasswd" content should match /user1:hash1/File "/etc/apache2/htpasswd" content should ma
shutdown of VM... That's what Terraform's outputs are for. 88 Chapter 3 There's more... A lot more configuration options are possible for the target node's run list: $ knife node run list add my node name 'recipe[platform]' Wait
for the next Chef run or run it yourself and Docker will be available on any node, including this recipe. [...] aws_security_group. Creation complete Apply complete Apply complete to make infrastructure code testing that you'll see a
 ities in the dedicated section of this book. Declare the Chef version we're currently using in production, so it's stable and stays the same: variable "chef version" { default = "12.13.37" } 4. Let's name it webusers, and we'll store our users under this directory. Docker is no exception to the rule, and good practices are always moving
getting updates, and might also be a little different between communities. Up 360 Chapter 9 Let's ensure the password used for the MySQL root password is really the one provided by the docker-compose.yml file, using the docker-compose.yml file, using the docker-compose.yml file, using the docker-compose.yml names): $ docker-compose.yml file, using the docker-compo
 ---+ | information schema | | mysql | | performance schema | | wordpress | +----
compose exec mysql /usr/bin/mysql -uroot -pexample [...] MariaDB [(none)]> show databases; +----
 --+ | Database | +----
 ----+ 4 rows in set (0.00 sec) When we're done with our initial Docker Compose environment, let's destroy it; the containers and
networks will be removed: $ docker-compose down Extending Docker Compose Now we know the basics, let's extend the usage a little. Creating a PostgreSQL RDS database with Terraform Amazon Relational Database Service (RDS) is an on-demand, ready-to-use, and resizable EC2 instance specifically tailored and configured to run the requested
database server. Granted that containers are running in an isolated environment, but still, a publicly facing daemon is running as root on a system, and a security breach may give an attacker access to this particular container, and maybe root shell access, giving access at least to the container's Docker overlay network. Supporting another platform
Let's check whether this code would work on Ubuntu 16.04 as well. You'll then be able to try all CoreOS features and configuration options for all release channels (alpha, beta, or stable). Here is our Vagrant-hostmanager is not installed! end
Vagrant.configure("2") do |config| config| config| config| config| config| config| config.hostmanager.manage dest = true config.hostmanager.manager.manager.manage dest = true config.hostmanager.ma
vb.cpus = vm cpus end puppet.vm.network :private network, ip: "192.168.50.10" puppet.hostmanager.aliases = %w(puppet) puppet.vm.provision :shell, :path => "puppet.vm.synced_folder "puppet.vm.synce
Vagrant file relies on the vagrant-hostmaster plugin. Today, he works as a Lead SRE at Criteo, where he manages distributed systems and NoSQL technologies. pp.q2Z81Z {:ignore => } should compile and work with no error Destroying vagrant boxes ==> ubuntu-1604-x64: Forcing shutdown of VM... Paste the following JSON content in
AmazonS3FullAccess.json at the root of the Ansible folder: { "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": "s3:*", "Resource": "*" }] } 79 Provisioning IaaS with Terraform Use this local policy in the iam_policy module: - name: Assign a AmazonS3FullAccess policy to mary iam_policy: iam_type: user iam_name: mary policy_name
AmazonS3FullAccess state: present policy document: AmazonS3FullAccess.json 80 3 Going Further with Terraform ff Provisioning a CentOS 7 EC2 instance with Chef using Terraform ff Using data sources, templates, and local execution ff
Executing remote commands at bootstrap using Terraform ff Using Docker with Terraform ff Simulating infrastructure state ff Maintaining a clean and standardized Terraform ff Using Docker with Terraform ff Using Docker
ff External monitoring integration with StatusCake 81 Going Further with Terraform Introduction In this chapter, we'll go beyond the essentials of using Terraform we covered in Chapter 2, Provisioning IaaS with Terraform. This volume will be mounted by the compute instance under a dedicated device (likely /dev/vdb). A pull request is a request to
merge the content of one branch to another branch. Here's how all this ties together: deploy revision 'wordpress' do repo ' revision 'HEAD' user 'apache' deploy end Once the code is applied, the /var/www/mysite (or whatever
name you may have overridden) structure will change a little: $ ls /var/www/mysite/ current index.html releases 316 shared folder. Luckily, there's a mechanism named notifications in Chef, that helps trigger an action, when a resource changes. Recent initiatives are targeting bare metal servers with
an API, so we get the best of both worlds—on-demand servers through an API and incredible performance through direct access to the hardware. Install Composer sudo chmod +x /usr/local/bin/composer sudo chmod +x /usr/local/bin/composer sudo chmod +x /usr/local/bin/composer sudo chmod +x /usr/local/bin/composer.
looks: #cloud-config # Install packages on first boot packages: - tcpdump - docker - nmap 166 Chapter 5 Using cloud-init on Advanced Details and we'll be able to paste our sample (and simple) cloud-config YAML file, or even simply upload it: In this case, the
Ubuntu 16.04 instance we just launched will already have the htop and tcpdump system tools installed, along with the Linux distribution's supported version of Docker --version Docker version 1.11.2, build b9f10c9 We
can manually update cloud-config.yml of a particular instance by powering off the instance, then under the Actions menu, navigate to Instance Settings | View/Change User Data. The best way is to think about something we could reuse as many times as needed (in case we would like to add more sites). We know we want our flannel network to work
on the 10.1.0.0/16 network, so we can create a drop-in system configuration file with its content that will be executed before the flanneld service. It is a good practice to plug the build process in CI and execute tests against running this new image! 348 Chapter 9 Building and using Golang applications with Docker Golang is a great language able to
create statically linked binaries for different platforms such as Linux (ELF binaries) or Mac OS (Mach-O binaries). Exposing Docker ports in Vagrant Okay, the previous example wasn't terribly useful as we didn't expose any ports. (refer to the lines with Scheduling refresh of Service[apache2] and Triggered 'refresh' from 2 events): Notice:
/Stage[main]/Apache/File[/etc/apache2/sites-enabled/000default.conf]: Scheduling refresh of Service[apache2/sites-enabled/000default.conf]: Scheduling refre
distinct step, so we can optionally customize this step later. Now we can create our database resource: mysql database re
simple, we won't use these right now, even though you'll probably start looking into it as soon as you'll need it. Create it at the root of the Chef repository: $ cat Makefile tests: foodcritic -t ~FC033 -t ~supermarket --exclude test cookbooks/mysite Now, you or some CI system can automatically check the code for quality or regression in quality. We will
now add a unit test for each manifest in our Apache module. We will then use a module from Puppet Forge. But how do we test for changes before applying them? Now let's say we need our infrastructure to evolve, and create an S3 bucket. The topology looks like this: Getting ready To step through this recipe, you will need the following: ff A working
Terraform installation ff A Google Compute Engine account with a project ff An Internet connection 137 Automating Complete Infrastructures with Terraform How to do it... The first thing we need to do is to get our credentials from the console. conf').with(user: 'root', group: 'root') end All in all, we've now covered 100% of our resources! As the
output says: You are awesome and so is your test coverage! Have a fantastic day! Stubbing data bags for searches The mysite cookbook we created earlier contains a search in a data bag to later populate a file with content. Getting ready To step through this recipe, you will need the following: ff A working Vagrant installation ff A working VMware
Workstation (PC) or Fusion (Mac) installation ff An Internet connection How to do it... The Chef Bento project provides various multiprovider images we can use. The Terraform Version in use for this book is 0.7.3. Handling different environments with Terraform It's a common and recommended setup
to have different infrastructure environments, with some level of parity. So let's ask them what exactly they need to access in the AWS space: Mary S3 in read and write Joe EC2 in read only As expected, neither user really needs full access! Amazon helps by offering prebuilt security policies for IAM. Let's find out: $ git status [...] instances.tf keys.tf
keys/ provider.tf variables.tf Good, those are the files we just created. List the available remote environments: $ knife environments is the files we just created. List the available remote environments available remote environment list.
Enabling and starting the MariaDB service Do exactly the same for MariaDB's mariadb service in mariadb/metadata.rb: version in cookbook to the Chef server: $ knife cookbook upload mariadb Applyation in mariadb/metadata.rb: version in mariadb/metadata.rb:
-----+ | information schema | | test | +----
 ---+ We've just covered how to handle a system
service using Chef, so you now know how to easily and repeatedly deploy packages and manage the corresponding service. Here's a sample Ansible role that simply installs Docker and starts it, so we can play with it, under ansible/main.yml: --- hosts: centos7 hosts become: yes tasks: - name: Install EPEL yum: name=epel-release state=present -
name: Install Docker 91 Going Further with Terraform yum: name=docker state=present - name: Start docker service: name=docker state=started enabled=yes Now you just have to execute Ansible when you want, it is all ready and configured! $ ansible-playbook -i hosts -u centos ansible/main.yml PLAY [centos 7 hosts]
bootstrap, even before the proper configuration management system such as Chef or Ansible takes responsibility. We know from its documentation that it needs to share the Docker socket in read-only mode (/var/run/docker.sock) to dynamically access the running containers, and we want it to run on the default HTTP port (tcp/80). How does our
hosts file look? Even if we won't be managing it using Puppet, these packages will install some commands necessary for upcoming examples. In our case, we want to ask our coworkers to merge our new infrastructure branch into the master branch, to create some discussion: 118 Chapter 3 When you open a pull request, GitHub automatically tries
the requested merge (in our case, from our branch to master). If we don't, r10k will perform a complete installation after removing all the content in the modules directory. Through the various chapters of this book, we'll also use free Software-as-a-Service (SaaS) accounts such as GitHub, Travis CI, Docker Hub, Quay.io, Hosted Chef, and StatusCake
As we can guess by their names, they respectively help create MySQL databases and MySQL users. Here's what will happen: ff Two nodes are created ff The node joins the swarm Let's start by creating a variable for the number of nodes we want, in
variables.tf: variable "num nodes" { default = "1" description = "Number of Docker Swarm nodes" } Override that value as the cluster grows in terraform.tfvars: num nodes = "2" Create the nodes using the same packet device resource we used for the master: resource "packet device" "swarm nodes" { count = "${var.num nodes}}" hostname =
"swarm-node-\{count.index+1\}" plan = "\{var.plan\}" facility = "\{var.facility\}" operating system = "\{var.worker\ token\ file\}" Add a file provisioner "file" { source = "\{var.worker\ token\ file\}" destination = "\{var.worker\ token\ file\}" } 161
Automating Complete Infrastructures with Terraform Using the same update and Docker installation scripts as the master, create the same remote-exec provisioner:
operating system is now fully updated and Docker is running. Downloading the example code You can download the example code files for this book from your account at http:// www.packtpub.com. Testing is good for quality, and writing tests first ensures that we write code that would make a test pass, instead of writing tests after the code is
written, which would somehow lead to missed errors. 196 Chapter 6 All examples from this book have been developed with Puppet 4.8 (open source edition). Chef recipes Recipes are simply plain Ruby files including a number of Chef resource edition).
service to be restarted. Use this book's chapter on Terraform code to create a CloudWatch Logs group and a dedicated IAM user, or manually create both. Configuration step on a new instance is setting the time zone. The worst that can happen in this area is the total waste of the
logs. No, he can't: $ s3cmd --access key= --secret key= ls ERROR: S3 error: 403 (AccessDenied): Access Denied But can the Joe user simply list the EC2 VMs as he needs to, with the same command that was forbidden to Mary? Getting ready To step through this recipe, you will need: ff Access to a cloud-config enabled infrastructure How to do it...
The first file we'll write is a MOTD (short for Message Of The Day) with root read-write permissions, read-only for everyone else. Select a container and we'll be able to enter it, read logs, see its image history, kill it, and more: Those commands are also always available right from the main interface: want to gain a shell on this selected container? We
want to access port 80 from the container on port 80 of our host using the -p option, with the latest Nginx version available: $ docker run --rm -p 80:80 nginx Make some HTTP requests such as a curl: $ curl -IL HTTP/1.1 200 OK Server: nginx/1.11.5 [...] The logs on the Docker stdout are displaying the logs as follows: 172.17.0.1 --
[21/Nov/2016:21:21:15 +0000] "HEAD / HTTP/1.1" 200 0 "-" "curl/7.43.0" "-" 336 Chapter 9 Maybe for some reason we need to launch a specific Nginx version; $ curl -IL HTTP/1.1 200 OK Server: nginx/1.10.2 Sharing data
with a container We want our own content to be displayed instead of the default Nginx page. We might want to format this storage space and be sure it's mounted on the host, but while we always want the disk to mount, we probably don't want it to be formatted at each boot! The bootcmd directive is there to handle everything related to commands
to be executed very early in the boot process, while the runcmd directive is executed much later in the boot process (and only once). WARNING: indentation of => is not properly aligned (expected in column 34, but found it in column 34, but found it in column 31) on line 12 ... x Preface To view the previously submitted errata, go to content/support and enter the name of the
```

book in the search field. To match our main requirement, we'll start with the ruby: 2.3 image: FROM ruby: 2.3

```
revisions to allow rollbacks Our HTTP web server runs under the apache user The virtual host folder is inherited from an attribute set earlier (/var/www/#{node['sitename']}) There's no database migrations to execute with WordPress The deploy revision resource is modeled after Capistrano and therefore comes from the Ruby on Rails world. You'l
know everything from launching fine-tuned EC2 instances and optimized RDS databases dynamically in different regions, to creating tight security groups, deploying SSH key pairs and securing IAM access keys, enabling log storage with CloudWatch, generating useful outputs, handling infinite Simple Storage Service (S3) storage, and using private
Docker repositories. GitHub is one of them, and we'll see how to manage members of an organization, various teams, and control repository access, right from our infrastructure code. A simple Ansible Docker playbook for Vagrant Our playbook file (playbook.yml) is a plain YAML file, and we'll do the following in this order: 1. The best part is, it's
already included in the Chef DK, so we just have to use it. ff EPP for Embedded Puppet (Puppet 4 and higher). To give clear instructions on how to complete a recipe, we use these sections as follows: Getting ready This section tells you what to expect in the recipe, and describes how to set up any software or any preliminary settings required for the
recipe. We can do this in a file named pool.tf with the google_compute_http_health_check resource: resource: resource "google_compute_http_health_check" "www" { name = "http" request_path = "/" check_interval_sec = 1 healthy_threshold = 3 timeout_sec = 1 healt
transform those values into variables for better tuning on your end! Now, let's define the pool, which is defined by the results of the health checks and instances inclusion. Also available are Node 6, 4, and below. config.vm.provision "ansible local" do |ansible local" do |ansible local" do |ansible local" do |ansible local | ansible local | ansib
You can also use the following command: $ vagrant up [...] ==> srv-1: Running provisioner: ansible local... However, it's something quite different to be able to rebootstrap a working system from scratch. In Chapter 7, Testing and Writing Better Infrastructure Code with Chef and Puppet, we used only one platform (Ubuntu) in
spec/acceptance/nodesets/default.yml. Let's write the FROM instruction in Dockerfile that will make it pass; this is because the current Debian stable is version 8: FROM debian: $$ bundle exec rspec --color --format documentation Docker NGINX image installs Debian Jessie Finished in 0.72234
seconds (files took 0.29061 seconds to load) 1 example, 0 failures Good job! Our test has passed, meaning this really is Debian 8. We can build our own availability monitoring systems, or we can use third-party services. This way, we can simply choose which one to use. In the aws.rb file, include the chef-vault recipe and set aws to the result of the
chef vault item helper search: include recipe 'chef-vault' aws = chef vault item('aws', 'eu-west-1') 307 Maintaining Systems Using Chef and Puppet If the node making the request isn't allowed to decrypt the data with its private key, we'll get an error. Quitting it destroys it and its content is lost forever as we specified the --rm option. This option is
used to test code immediately after a change and implies other options such as --no-daemonize, --onetime, and --verbose. Let's do the same for Mary in data bags/webusers/mary.json { "id": "mary", "htpasswd": "$apr1$eR7H0C5r$OrhOQUTXfUEIdvWyeGGGy/" } 5. To verify the file sharing sync is working correctly, edit the ./resources/views/welcome
[...] Applying a Chef cookbook at bootstrap using cloud-init We certainly want to apply at least an initial cookbook for configuring the instance. Chef run list A run list is a list of roles or recipes that a node has to apply. Alternatively, it can use the DOCKER_HOST environment variable, or just the local daemon if not specified. Two languages are
available: ff ERB for Embedded Ruby. See also ff The Chef documentation for notifications: common.html#subscribes: common.html#subscribes ff The Puppet notify resource documentation for subscribes ff The Puppet notification ff The Puppet notif
basics of our LAMP infrastructure up and running, let's secure it a little by creating an htaccess file with a few authorized users in it. Compressing objects: 100% (8/8), done. Our system is now perfectly dynamic and can reload its configuration at will at every change. Getting ready To step through this recipe, you will need the following: ff A working
Docker installation ff An AWS account with full S3 access 365 Working with Docker How to do it... We'll use Docker Compose to work through this recipe. 311 Maintaining Systems Using Chef and Puppet agent. You can download this file from
 downloads/InfrastructureasCode IAC Cookbook ColorImages.pdf Errata Although we have taken every care to ensure the accuracy of our content, mistallation ff A working VirtualBox installation ff A working VMware Workstation (PC) or
Fusion (Mac) installation ff A working Vagrant VMware plugin installation ff An internet connection ff The Vagrantfile from the previous recipe using a bento/centos 72 box 10 Chapter 1 How to do it... Some Vagrant boxes are available for multiple hypervisors, such as the CentOS 7 Bento box we previously used. Confirm that the plugin is installed: $$
vagrant plugin list vagrant-vbguest (0.13.0) 3. Fortunately, ChefSpec allows us to stub the data bag with real content. This is rather unnecessary, if you consider that the scope of this configuration in Docker is just to send logs. Add the following to variables.tf to use eu-west-1 as a default region: variable "aws region" { default = "eu-west-1" } 62
Chapter 2 You can now use this variable in the provider.tf file to set the region: provider "aws" { region = "${var.aws_region}"]}" We now have an easily geographically deployable infrastructure that anyone in your team
can launch close to him or her without the need to change code. We can now build more complex and secure containers using this technique. kitchen creates the virtual testing environment (in our case, through Vagrant and an hypervisor), but does not provision it. Either plain files can be copied or dynamic files can be generated through
templates. Once the virtual machine is ready, the plugin will detect the operating system, decide if the Guest Additions need to be installed or not, and finally download and install the corresponding Guest Additions. That's the bare minimum: name
"webserver" description "An HTTP server for our application" run list "recipe[apache]" 2. This will be our base to then write integration tests so we can make sure we achieve what we intended to achieve with multiple cookbooks and sources of code, really reaching the target and doing the job on a real system. We encourage you to secure your
Docker systems using SELinux, but to disable SELinux just type the following: $ sudo setenforce 0 There's more... You can choose not to use your local or default Docker installation, but instead use a dedicated VM, maybe to reflect production or a specific OS (such as CoreOS). The new code following puppet-lint recommendations, is: ff For
modules/apache/manifests/init.pp: # See README class apache { package {'apache2': ensure => present, } service {'apache2': ensure => present, } file {'/etc/apache2': ensure => present, } file {'/et
htpasswd database # @param users Array of hash containing users # See README define apache/htpasswd.erb'), } } 256 Chapter 7 ff For modules/apache/manifests/htaccess.pp: # @param filepath
Path of the htpasswd database # @param docroot DocumentRoot where the .htaccess should be generated # See README define apache:htaccess ($filepath, $docroot) { file { "${docroot}/.htaccess:" ensure => present, owner => 'root', group => 'root', mode => '0644', content => template('apache/htaccess.erb'), } } ff For
modules/apache/manifests/vhost.pp: # @param website Site name # @param docroot DocumentRoot # See README define apache::vhost ($website | ->
 'docroot' => $docroot}), } file { "/etc/apache2/sites-enabled/${website}.conf"; require => File["/etc/apache2/sites-available/${website}. We are about to create three modules (apache, php, and mariadb), so let's create a minimalist module layout for them: mkdir
modules/apache mkdir modules/php/templates mkdir modules/php/templates mkdir modules/mariadb/manifests mkdir modules/php/templates mkdir modules/mariadb/manifests mkdir modules/manifests mkdir modules/mariadb/manifests mkd
resource package.html ff The Puppet package resource documentation: puppet/4.8/types/package resource. To do this, let's declare a new empty variable to store our public key in variables.tf: variable "aws ssh admin key file" { } Initialize the
variable to the path of the key in terraform. This is a much clearer and sey file = "keys/aws terraform" Now let's use it in place of our previous keys.tf code, using the file() interpolation: resource "aws key pair" "admin key file = "keys/aws terraform". You let's use it in place of our previous keys.tf code, using the file() interpolation: resource "aws key pair" "admin key file = "keys/aws terraform". You let's use it in place of our previous keys.tf code, using the file() interpolation: resource "aws key pair" "admin key file = "keys/aws terraform". You let's use it in place of our previous keys.tf code, using the file() interpolation: resource "aws key pair" "admin key file = "keys/aws terraform". You let's use it in place of our previous keys.tf code, using the file() interpolation: resource "aws key pair" "admin key file = "keys/aws terraform". You let's use it in place of our previous keys.tf code, using the file() interpolation: resource "aws key pair" "admin key file = "keys/aws terraform". You let's use it in place of our previous keys.tf code, using the file() interpolation: resource "aws key pair" "admin key file = "keys/aws terraform". You let's use it in place of our previous keys.tf code, using the file() interpolation: resource "aws key pair" "admin key file = "keys/aws terraform". You let's use it in place of our previous keys.tf code, using the file () interpolation: resource "aws key pair" "admin key file = "keys/aws terraform". You let's use it in place of our previous keys.tf code, using the file () interpolation: resource "aws key pair" "admin key file = "keys/aws terraform". You let's use it in place of our previous keys.tf code, using the file () interpolation: resource "aws key pair" "admin key file = "keys/aws terraform". You let's use it in place of our previous keys.tf code, using the file () interpolation: resource "aws keys terraform". You let's use it in place of our previous keys the file () interpolation the file () interpolation the file () interpolation the file () interpola
more concise way of accessing the content of the public key from the Terraform resource. We need to create a README.md file at the top-level directory of the module. yml: version: '2' services: tomcat: 9 logging: driver: 'awslogs-region: 'us-east-1' awslogs-group: 'docker logs' awslogs-stream: 'tomcat' 397 Maintaining
Docker Containers Launch the compose as usual: $ sudo docker-compose up Creating network "ubuntu default driver [...] tomcat 1 driver | WARNING: no logs are available with the 'awslogs' log The tomcat CloudWatch log stream is now automatically created and the logs flow into it. The second command is printing the date in the
logs (so we know when the boot process started). At the end of this double linting process, our Dockerfile changed a lot, as shown here: FROM debian:stable-slim LABEL maintainer="John Doe " 364 Chapter 9 LABEL version=1.0 RUN apt-get install -y --no-install-recommends apache2=2.4.1010+deb8u7
\ && apt-get clean \ && rm -rf /var/lib/apt/lists/* EXPOSE 80 ENTRYPOINT ["/usr/sbin/apache2ctl"] CMD ["-D", "FOREGROUND"] We added labels to identify the image, versions, and maintainer, and we fixed a proper version of the apache2 package. However, Hadolint is not yet up to date for this deprecation change, so execute the following to
 ignore one or more IDs, to still be cool: $ hadolint --ignore DL4000 --ignore Dockerfile lint This project lead by the Project Atomic team (is also proposing different checks and strong opinions on how a Dockerfile should be written. 52.31.10.18 =eu-west-1 b95a5262... If you use the latter, the output will iterate through all available
machines. We're happy about an instance with the right AMI being created, so let's terraform apply. Write a short description and choose the visibility of the image: private stuff should remain private, while public can stay public. Using this pattern, it is easier to refactor only profiles classes when the technology needs to be changed. Start by
verifying what's not tracked: $ git add cookbooks/platform $ git commit -m "added docker support to the platform" Is our git tree clean so that we can ship it to our team? Remember the default
sitename attribute in the apache cookbook? As we can't interact with the process, let's skip the host key checking and other initial SSH checks: provisioner "local-exec" { 160 Chapter 4 command = "ssh -t -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ${var.ssh_key} ${var.ssh_key} ${var.ssh_key} ${var.ssh_key}$ ${var.
puppet_master.sh provisioning script is as follows: #!/usr/bin/env bash # Exit immediately if a command exits with a non-zero status set -e # puppetLabs repo..." debrepo=$(mktemp) wget --output-document=${debrepo} ${DEBREPO} dpkg -i ${debrepo} apt-get update #
 Install Puppet Server from puppetlabs # This will remove puppet-common package provided by the vagrant box (if any) echo "Installing Puppet..." apt-get install -y puppetserver # For tests, limit memory usage. service.d/20-cloudinit.conf: $ cat /run/system/locksmithd.service.d/20-cloudinit.conf [Service]
Environment="REBOOT WINDOW START=04:00" Environment="REBOOT WINDOW LENGTH=2h" At any time, we can check for a reboot slot availability using the locksmithctl status Available: 1 Max: 1 If another machine is currently rebooting, its ID is displayed so we know who's rebooting. Take advantage of the Ruby
nature of the Vagrantfile and use a simple loop to iterate through both values: ["vmware fusion", "vmware workstation"].each do |vmware| configuration here end our default Bento CentOS 7.2 image has only 512 MB of RAM and one CPU. Notice:
/Stage[main]/Main/Node[web.pomes.pro]/Apache::Vhost[web.pomes. 306 Chapter 8 Getting ready To step through this recipe, you will need: ff A working Chef code (optionally) from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet,
Chapter 7, Testing and Writing Better Infrastructure Code with Chef and Puppet, or any custom Chef code How to do it... We'll build on the previous, already existing, mysite cookbook; however, any other situation will work similarly. Getting ready To step through this recipe, you will need the following: ff A working Terraform installation ff A Digital to the previous and the following this recipe, you will need the following: ff A working Terraform installation ff A Digital to the following: ff A working Terraform installation ff A Digital to the following: ff A working Terraform installation ff A Digital to the following: ff A working Terraform installation ff A Digital to the following: ff A working Terraform installation ff A Digital to the following: ff A working Terraform installation ff A Digital to the following: ff A working Terraform installation ff A Digital to the following: ff A working Terraform installation ff A Digital to the following: ff A working Terraform installation ff A Digital to the following: ff A working Terraform installation ff A Digital to the following: ff A working Terraform installation ff A Digital to the following: ff A working Terraform installation ff A Digital to the following: ff A working Terraform installation ff A Digital to the following: ff A working Terraform installation ff A Digital to the following: ff A working Terraform installation ff A Digital to the following: ff A working Terraform installation ff A Digital to the following: ff A working Terraform installation ff A Digital to the following: ff A working Terraform installation ff A Digital to the following: ff A working Terraform installation ff A Digital to the following: ff A working Terraform installation ff A Digital to the following: ff A working Terraform installation ff A Digital to the following: ff A working Terraform installation ff A Digital to the following: ff A working Terraform installation ff A Digital to the ff A
Ocean account ff A StatusCake account ff An Internet connection How to do it... Let's start by creating the digitalocean provider (it only requires an API token) in a file named variable in a file named variable in a file named variable s.tf: variable "do_token" { description = "Digital Ocean provider" | Digital Ocean provider | Digital Ocean p
Token" } Also, don't forget to set it in a private terraform.tfvars file: do token = "a1b2c3d4e5f6" 130 Chapter 4 Handling the SSH key We know that we'll need an SSH key to log into the cluster members. So let's define a file, module/apache/manifest/vhost.pp that will use such a statement: define apache::vhost ($website, $docroot) { file {
 "/etc/apache2/sites-available/$website.conf": ensure => present, owner => 'root', group => 'root', mode => '0640', content => epp('apache/vhost.epp', 221 Fundamentals of Managing Servers with Chef and Puppet {'website' => $website, 'docroot'=>$docroot'}, } file { "/etc/apache2/sites-enabled/$website.conf": ensure => link, target =>
 "/etc/apache2/sites-available/$website.conf", require => File["/etc/apache2/sites-available/$website.conf"], } The website name and the document root are the two parameters for our apache::vhost statement and are passed to the epp function along with the template file name in the first file directive. $ puppet-lint
 modules/apache/manifests/vhost.pp WARNING: defined type not documented on line 1 WARNING: variable not enclosed in {} on line 6 ... Go to and log in or create an account if you don't have any. Building quicker tests for a faster feedback loop will then be a new subject. The minimal command line we can build is as follows: ff The IP or FQDN of
the host we want to configure (1.2.3.4) 201 Fundamentals of Managing Servers with Chef and Puppet ff The username to use to connect to the server (sudoer if not root). It will also give insights and advanced Docker techniques, and much more. To be certain of
what we'll deploy, we'll start by filling a simple table with the database Engine version rds engine version rds engine version rds engine version 9.5.2 RDS Instance Type rds instance Type rds instance type db.t2.micro RDS Storage Size (GB)
rds storage size 5 RDS First Database Name rds db name iac book db RDS Administrator Username rds admin user dbadmin RDS Administrator Password RDS Publicly Accessible true Let's set all those variables in our variables.tf file: variable "rds identifier" { default = "db"
variable "rds_instance_type" { default = "db.t2.micro" } variable "rds_engine_version" { default = "fds_engine_version" { default = "fds_engine_version" { default = "fds_engine_version" } variable "rds_engine_version" } variable "rds_engine_version" { default = "fds_engine_version" } variable "rds_engine_version" } variable "rds_engine_version" { default = "fds_engine_version" } variable "rds_engine_version" }
default = "super secret password" } variable "rds publicly accessible" { default = "true" } As we're running PostgreSQL and we want it to be available on the Internet (though generally not a good idea for production), we'll need a security group allowing just the default PgSQL port (TCP/5432) for our IP address (refer to the Using AWS security group allowing just the default PgSQL port (TCP/5432) for our IP address (refer to the Using AWS security group allowing just the default PgSQL port (TCP/5432) for our IP address (refer to the Using AWS security group allowing just the default PgSQL port (TCP/5432) for our IP address (refer to the Using AWS security group allowing just the default PgSQL port (TCP/5432) for our IP address (refer to the Using AWS security group allowing just the default PgSQL port (TCP/5432) for our IP address (refer to the Using AWS security group allowing just the default PgSQL port (TCP/5432) for our IP address (refer to the Using AWS security group allowing just the default PgSQL port (TCP/5432) for our IP address (refer to the Using AWS security group allowing just the default PgSQL port (TCP/5432) for our IP address (refer to the Using AWS security group allowing just the default PgSQL port (TCP/5432) for our IP address (refer to the Using AWS security group allowing just the default PgSQL port (TCP/5432) for our IP address (refer to the Using AWS security group allowing just the default PgSQL port (TCP/5432) for our IP address (refer to the Using AWS security group allowing just the default PgSQL port (TCP/5432) for our IP address (refer to the Using AWS security group allowing just the default PgSQL port (TCP/5432) for our IP address (refer to the Using AWS security group allowing just the default PgSQL port (TCP/5432) for our IP address (refer to the Using AWS security group allowing just the default PgSQL port (TCP/5432) for our IP address (refer to the Using AWS security group allowing just the Using AWS security group allowing group allowing group group group group group group g
groups with Terraform recipe), in security group "rds security group" group group" group
Provisioning IaaS with Terraform tags { Name = "rds_security_group" } } Now we have everything in place to construct the aws_db_instance resource: resource resource: resource resource: resource resource: resource resour
allocated storage = "${var.rds storage size}" name = "${var.rds storage size}" name = "${var.rds admin password}" publicly accessible = "${var.rds
this recipe, you will need the following: ff ff ff A working Terraform installation An AWS provider configured in Terraform (refer to the previous recipes) An Internet connection 63 Provisioning IaaS with Terraform How to do it... We'll start by creating a simple and explicitly public bucket on S3 named iac-book, using the aws_s3_bucket resource (and a simple and explicitly public bucket on S3 named iac-book, using the aws_s3_bucket resource (and a simple and explicitly public bucket on S3 named iac-book, using the aws_s3_bucket resource (and a simple and explicitly public bucket on S3 named iac-book, using the aws_s3_bucket resource (and a simple and explicitly public bucket on S3 named iac-book, using the aws_s3_bucket resource (and a simple and explicitly public bucket on S3 named iac-book, using the aws_s3_bucket resource (and a simple and explicitly public bucket on S3 named iac-book, using the aws_s3_bucket resource (and a simple and explicitly public bucket on S3 named iac-book, using the aws_s3_bucket resource (and a simple and explicitly public bucket on S3 named iac-book, using the aws_s3_bucket resource (and a simple and explicitly public bucket on S3 named iac-book, using the aws_s3_bucket resource (and a simple and explicitly public bucket on S3 named iac-book, using the aws_s3_bucket resource (and a simple and explicitly public bucket on S3 named iac-book, using the aws_s3_bucket resource (and a simple and explicitly public bucket on S3 named iac-book, using the aws_s3_bucket resource (and a simple and explicitly public bucket on S3 named iac-book, using the aws_s3_bucket resource (and a simple and explicitly public bucket on S3 named iac-book, using the aws_s3_bucket resource (and a simple and explicitly public bucket on S3 named iac-book, using the aws_s3_bucket resource (and a simple and explicitly public bucket on S3 named iac-book, using the aws_s3_bucket resource (and a simple and a simple and explicitly public bucket on S3 named iac-book, using the aws_s3_bucket (and a simple and a simple a
tag for the sake of it): resource "aws_s3_bucket" "iac_book" { bucket = "iac-book" acl = "public-read" tags { Name = "IAC Book Bucket is immediately available for storing objects. Infrastructure code commit What are the new files on this branch that aren't on master? This class needs to
include all needed profiles to create the role. As always, with AWS, it's highly recommended that you use a dedicated IAM user for each AWS key pair we'll use. Vagrant work pretty well with a Berkshelf enabled Chef environment, using the vagrant-berkshelf plugin. packer.io/) and the Chef Bento project (. We'll see how it applies to the correct AMI
of the corresponding AWS. 133 Automating Complete Infrastructures with Terraform If you like, you can access the fqdn attribute of this resource right from the outputs (outputs.tf): output "CoreOS Cluster Members DNS" { value = "${formatlist("ssh %v -i ${var.ssh_key_file}", digitalocean_record.ipv4.*.fqdn)}" } Integrating cloud-
init We need to build a fully working cloud-config.yml file for our CoreOS cluster. Go into the puppetcode directory, which is the shared folder between your workstation and the Puppet server: cd puppetcode 3. Let's keep it simple here, and please refer to the very complete online documentation for more complex uses—because this resource is
absolutely powerful and does wonders when properly manipulated. Apache Puppet code localhost $ scp /var/folders/k9/7sp85p796qx7c22btk7_tgym0000gn/T/ beaker20161103-41882-1twwbr2 ubuntu-1604-x64:/tmp/apply_manifest. Are we able to easily rebootstrap a similar CentOS 7.2 server from scratch to the point that it is similarly installed
 without any Chef or system error? The following command is used to enable the service mode: # puppet resource service puppet ensure=running enable=true In this mode, the Puppet agent will apply the configuration every 30 minutes by default. They are all very often based on the images from the operating system container images in the
preceding table. 369 Maintaining Docker Containers Testing Docker containers with BATS BATS (Bash Automated Testing System) allows you to have quick and easy tests in a very natural language, without the need of a lot of dependencies. Now upload the updated cookbook (we've already bumped it): $ knife cookbook upload apache 4. This can be
 done directly in the main manifest using file directives to create the DocumentRoot directory and the file itself: node 'web.pomes.pro' { $website = $fqdn; $docroot = $docroot; ensure = $directory, owner = 'www.
data', group => 'www-data', mode => '0755', } file {"$docroot/index.php": ensure => present, owner => 'www-data', group => 'www-data', mode => '0755', } file {"$docroot/index.php": ensure => present, owner => 'www-data', mode => '0755', } We can now run the Puppet again. Let's create an output containing this information: output "S3" { value = "
${aws_s3_bucket.iac_book.region}.amazonaws. srv-1: Running ansible-playbook... This installer embeds everything needed to deploy Chef and all its dependencies. Fortunately, the puppetlabs_spec_helper gem provides such targets. We'll eventually end up deploying web applications directly from Chef. 280 Chapter 7 We need to define a helper file
that will be shared by all acceptance tests. 366 Chapter 9 Using an S3 backend An issue with a highly used local Docker registry is disk space management—it's finite. Adding GitHub teams Let's create our two teams, technical writers and engineering, with their respective privacy settings: // An engineering team resource "github teams" "engineering"
 { name = "Engineering Team" description = "Our awesome engineers" privacy = "secret" } // A documentation team resource "github team" description = "Our awesome technical Writers Team" description = "Our awesome engineers" privacy = "secret" } // A documentation team resource "github team" description = "Our awesome engineers" privacy = "secret" } // A documentation team resource "github team" description = "Our awesome engineers" privacy = "secret" } // A documentation team resource "github team" description = "Our awesome engineers" privacy = "secret" } // A documentation team resource "github team" description = "Our awesome engineers" privacy = "secret" } // A documentation team resource "github team" description = "Our awesome engineers" privacy = "secret" } // A documentation team resource "github team" description = "Our awesome engineers" privacy = "secret" } // A documentation team resource "github team" description = "Our awesome engineers" privacy = "secret" } // A documentation team resource "github team" description = "Our awesome engineers" privacy = "secret" } // A documentation team resource "github team" description = "Our awesome engineers" privacy = "secret" } // A documentation team resource "github team" description = "Our awesome engineers" privacy = "secret" } // A documentation team resource "github team resource "gith
 documentation: // Jane is a member of the engineering team resource "github team membership" "eng membership john" { team id = "${qithub team.engineering.id}" username = "jane" role = "member of the documentation team 124 Chapter 3 resource "github team membership" "doc membership john" { team id = "${qithub team.engineering.id}" username = "jane" role = "member of the documentation team 124 Chapter 3 resource "github team membership" "doc membership john" { team id = "${qithub team.engineering.id}" username = "jane" role = "member of the documentation team 124 Chapter 3 resource "github team membership" "doc membership" john" { team id = "${qithub team.engineering.id}" username = "jane" role = "member of the documentation team.engineering.id" username = "jane" role = "member of the documentation team.engineering.id" username = "jane" role = "member of the documentation team.engineering.id" username = "jane" role = "member of the documentation team.engineering.id" username = "jane" role = "member of the documentation team.engineering.id" username = "jane" role = "member of the documentation team.engineering.id" username = "jane" role = "member of the documentation team.engineering.id" username = "jane" role = "member of the documentation team.engineering.id" username = "jane" role = "member of the documentation team.engineering.id" username = "jane" role = "member of the documentation team.engineering.id" username = "jane" role = "member of the documentation team.engineering.id" username = "jane" role = "member of the documentation team.engineering.id" username = "jane" role = "member of the documentation team.engineering.id" username = "jane" role = "member of the documentation team.engineering.id" username = "jane" role = "member of the documentation team.engineering.id" username = "jane" role = "member of the documentation team.engineering.id" username = "jane" role = "member of the documentation team.engineering.id" username = "jane" role = "member of the documentation team.engineering.id" username = "jane"
"${github_team.documentation.id}" username = "john" role = "member" } Setting Git repository access rights The policy we've set is that members of the engineering group are admins of the repository" "infrastructure_doc" {
team_id = "${github_team.documentation.id}" repository = "infrastructure-repository" permission = "pull" } // engineers are admin on the repo resource "github_team.engineering.id}" repository = "infrastructure-repository" permission = "admin" } You've just set the essentials to manage
your GitHub organization right from Terraform! External monitoring in the world. If no data is found, Puppet will try to look up a common.yaml file. Now, if we look at the
 seconds ago latest 86.77 MB 44609fa7c016 18 terraform seconds ago unstable 86.77 MB 44609fa7c016 18 346 Chapter 9 terraform ago 83.61 MB 0.7.12 9d53a0811d63 9 minutes terraform ago 83.61 MB 0.7.12 9d53a0811d63 9 minutes terraform ago 83.61 MB 0.7.12 9d53a0811d63 9 minutes terraform ago 83.61 MB o.7.12 9d53a0811d63 9 minutes terraform ago 83.61 MB o.7.12
 used for unstable releases? We don't need it, so be sure to get rid of everything under the test folder to start fresh. Getting ready To step through this recipe, you will need the following: ff A working Docker installation How to do it... To orchestrate multiple containers together using Docker Compose, let's
start with an easy WordPress example. We can find versions 2.7, 3.3, 3.4, 3.5, and current beta versions based on Debian Jessie or Wheezy, Alpine, and Windows Server. So, now she can merge our branch, close the pull request with a message, and then delete the now useless branch: Our code and its fixes are now on master, along with the updated
infrastructure state, all done in full collaboration with a coworker. (listen-client-urls). 334 9 Working with Docker In this chapter, we will cover the following recipes: ff Docker usage overview ff Choosing the right Docker base image ff Optimizing the Docker image size ff Versioning Docker images with tags ff Deploying a Ruby-on-Rails web
 application in Docker ff Building and using Golang applications with Docker ff Networking with Docker ff Creating more dynamic containers ff Auto-configuring dynamic containers ff Deploying a private Docker registry with S3 storage Introduction
 Rails web applications. Nothing else can override a value set for an environment. It includes certificates, database configuration files, local assets, and so on. So, if we simply want to override a value set for an environment. It includes certificates, database configuration files, local assets, and so on. So, if we simply want to override a value of the sitename attribute to production.rb, it will look like this: override_attributes 'sitename' => 'mysite_production' Accessing the environment from
a recipe The node's environment is available from any recipe through the node.chef environment attribute. What if we want a different convergence interval, say, every 900 seconds (15 minutes)? Begin by obtaining the base64 version of the file: $
of the cloud-config file: - path: /etc/server-id content: YWJjLTEyMwo= encoding: b64 permissions: '0600' Let's verify the remote content is what we expected: $ ls -al /etc/server-id abc-123 It works! Our file is read/write for the owner only, and the content is abc-123. To create a letc/server-id abc-123 it works! Our file is read/write for the owner only, and the content is abc-123. To create a letc/server-id abc-123 it works! Our file is read/write for the owner only, and the content is abc-123. To create a letc/server-id abc-123 it works! Our file is read/write for the owner only, and the content is what we expected: $ ls -al /etc/server-id abc-123 it works! Our file is read/write for the owner only, and the content is abc-123. To create a letc/server-id abc-123 it works! Our file is read/write for the owner only, and the content is abc-123 it works! Our file is read/write for the owner only, and the content is what we expected: $ ls -al /etc/server-id abc-123 it works! Our file is read/write for the owner only, and the content is what we expected: $ ls -al /etc/server-id abc-123 it works! Our file is read/write for the owner only, and the content is what we expected: $ ls -al /etc/server-id abc-123 it works! Our file is read/write for the owner only, and the content is what we expected a letc/server-id abc-123 it works! Our file is read/write for the owner only file is read
network interface in our default network, mapped behind a public IPv4, add the following to the google compute instance resource: network interface { network interface and fully update it, then install, enable, and start Docker. 308 Chapter 8 It ranges from
networking details—such as link speed, MTU, or addresses—to all the memory usage details you'd find on a utility such as top, all of the imaginable data regarding filesystems or virtualization systems, or the list of every single installed package and logged-in users. The equivalent with AWS S3 is a bucket. This includes two of the Docker Hub (or
Quay.io) features: automatically build Docker images upon changes and serve them to the world. There's more... Using Puppet, all the examples we used were based on Vagrant, and it is easy to rebuild nodes from scratch. To start our project using the Python 3.5 image, add the following in a Dockerfile: FROM python:3.5 ENTRYPOINT ["/bin/bash"]
The main python: 3.5 image is around 683 MB. Here's a simple bucket with public read permissions, using the classic s3 module: --- name: create jac-book bucket s3: bucket iac-book bucket s3: bucket iac-book bucket s3: bu
Swarm cluster on bare metal with Packet IaaS clouds have been popularized through heavy usage of virtual machines. Let's try to list EC2 hosts with Mary's account using the aws command line (provided you configured the aws tool accordingly): $ aws --profile iacbook-mary ec2 describe-hosts An error occurred (UnauthorizedOperation) when calling
the DescribeHosts operation: You are not authorized to perform this operation. For example, if our domain is infrastructure-as-code. That way, I can choose what to output by simply adding a comment. The
environments/production.rb file should now look like this: name 'production' description 'The production environment from file environment from file environment production.rb Updated Environment production 327 Maintaining Systems Using Chef and Puppet
Commit the changes to git: $ git add . If you come across any illegal copies of our works in any form on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy. See also ff The Docker chef cookbook at Planning for the worse - train to rebuild working systems It's one thing to get a full
infrastructure finally managed by Chef—block by block, weeks after weeks, modification—keeping the Chef run always smooth and working. Installing sysdig is easy on most platforms, including CoreOS (install). For now, we need to restart Apache manually in order to have our virtual host running (as for Chef, we'll automate this
in the next pages): :~# service apache2 reload Now you should see the phpinfo page on See also ff The Chef documentation on attributes: . 129 Automating Complete Enfrastructures with Terraform In
this recipe, we'll build from scratch a fully working CoreOS cluster on Digital Ocean in their New York region, using Terraform and cloud-init. Vagrant uses boxes to run. @gousto ULHPC-mysql Configure and... Now let's create the template "/etc/aws/credentials" do source
'aws.erb' owner 'root' group 'root' mode '0600' variables(aws_region: aws['id'], aws_access_key: aws['aws_access_key'], aws_secret_key: aws['aws_secret_key']) end Here we are! The Chef server is now safely storing encrypted data. The current 4.x version of the plugins is still split. Chef roles Roles are usually functional and generic, more than
centered around a product. Getting ready To step through this recipe, you will need the following: ff A working Vagrant installation (with a hypervisor) ff An Internet connection ff The Vagrant Virtual Machine. 180 Chapter 5
The most important configuration parameters are the following: ff public ip: This specifies which interface to use to communicate with other hosts. It looks like this in spec/Dockerfile spec.rb: require "serverspec" require "docker" describe "Docker NGINX image" do before(:all) do @image = Docker::Image.build from dir('.') set :os, family: :debian set
 :backend, :docker set :docker image, @image.id end the Debian Jessie base's Docker image We now want to use a Debian stable for our project, which happens to be Debian 8 at the moment. The author's computers are laptops running Mac OS 10.11 and Fedora 25, with VirtualBox 5, but any other Linux distribution will work as well
In this mode, all Chef code is distributed from the Chef server. The refreshed state will be used to calculate this plan, but will not be persisted to local or remote state storage. We can now read it by opening index, html in any browser. Infrastructure is up-to-date. Getting ready To step through this recipe, you will need the following: ff A working
Docker installation ff An AWS account How to do it... Many different linters exist for linting Dockerfiles: Hadolint (. How to do it... Here's a simple Dockerfile to create a Terraform container (Terraform was covered earlier in this book): FROM alpine:latest ENV TERRAFORM VERSION=0.7.12 VOLUME ["/data"] WORKDIR /data RUN apk --update -
no-cache add ca-certificates openssl && \ wget -O terraform.zip " {TERRAFORM_VERSION}/ter raform.zip /var/cache/apk/* ENTRYPOINT ["/bin/terraform"] CMD ["--help"] This is the current, stable, and latest version, and it's 0.7.12 as well. This
kind of setup is the beginning of a good combination of infrastructure as code. Let's try it from our workstation. As we're dealing with Infrastructure-as-a-Service (IaaS) here, also required are valid accounts with Amazon Web Services (AWS), Google Cloud, Digital Ocean, Packet, Heroku, or an OpenStack deployment. The thing is, we're unit testing
and no real Chef server is answering requests. Thankfully, Chef is a first class provisioning tool in Terraform. So we can use a trick: ask the stats (docker ps) whose names we extracted using a Go template formatter (--format): $ docker stats (docker ps --format) (docker ps --form
NET I/O MEM USAGE / LIMIT BLOCK I/O PIDS sm streammachine-slave 2 18.34% 10.68% 258.2 GB / 10.27 GB 889.4 MB / 8.326 GB 419.1 MB / 0 B 16 400 Chapter 10 sm streammachine-master 1 1.89% 10.69% 179.6 GB / 536.5 GB 890.4 MB / 8.326 GB 419.1 MB / 8.3
/ 40.22 MB 546.1 MB / 8.326 GB 160.7 MB / 8.326 GB 160.7 MB / 317.9 MB 74 sm_streamer_1 0.10% 8.17% 30.15 GB / 30.29 GB 8.184 MB / 8.326 GB 8.192 kB / 0 B 7 Using Google's cAdvisor. The encryption is done through a shared secret, the secret being either a file
or a string. Companies might not like production API keys, private keys, or similar sensitive content to be stored in plain text on the Chef server or on third-party services, such as GitHub. Still with a focus on developers and achieving the highest code quality possible, we'll spend some time linting our code, and finally deploy our own Docker Registry
to store our own images internally—both on local storage and on AWS S3 for infinite space. We'll go through both these options. IAM support is not equivalent, as there's no IAM Managed Policies support. By setting the VAGRANT DEFAULT PROVIDER environment variable to the corresponding plugin, you will never have to specify the provider
 again, VMware will be the default: $ export VAGRANT_DEFAULT_PROVIDER=vmware_fusion $ vagrant up See also ff The Chef Bento Project at ff A community VMware vSphere plugin at ff A community VMware vCenter plugin at ff A community VMware vSphere pl
vCloud Air plugin at vagrant-vcloudair Extending the VMware VM capabilities The hardware specifications of the Vagrant this up (vagrant up), and you'll end up with a fully provisioned VM, using the old 11.18.12 version, and a running Apache 2 web server. This is done vary from image as they're specified at the creation time.
using the simple file resource with the file path as argument, giving its content inline. Install it like any other Vagrant plugin: $ vagrant plugin install vagrant plugin install vagrant plugin install it like any other Vagrant plugin.
need at least one public key uploaded to our Packet account. It means we have to filter hosts based on a search. This is no different with sysdig, and the return code is 1: $ echo $? Speed up deployments with linked clones
Linked clones is a feature that enables new VMs to be created based on an initial existing disk image, without the need to duplicate everything. The Docker Hub and Quay.io from CoreOS are the main Docker-managed registries you can subscribe to. Networking with Docker Hub and Quay.io from CoreOS are the main Docker-managed registries you can subscribe to.
to expose to concurrently running isolated or bridged networks. Pretty easy and dynamic! To give Digital Ocean power on our domain, we need to go to our registrar (where we bought our domain, we need to go to our registrar (where we bought our domain), and configure our domain, we need to go to our registrar (where we bought our domain), and configure our domain, we need to go to our registrar (where we bought our domain), and configure our domain, we need to go to our registrar (where we bought our domain), and configure our domain to be managed by Digital Ocean, using their own nameservers, which are as follows: ff ns1.digitalocean.com ff
ns2.digitalocean.com ff ns3.digitalocean.com This prerequisite being done, let's declare our domain name, and an initial IP address matching, that we can either set to a value you already know or to an arbitrary droplet:
resource "digitalocean_domain" "cluster_domainname" { name = "${var.cluster_domainname}" ip_address = "${digitalocean_droplet.coreos.0.ipv4} address}" } Add the new variable in variable stf: variable "cluster domainname" { default = "infrastructure-as-code.org" description = "Domain to use" } Don't forget to override it as necessary in
terraform.tfvars. The Chef server is open source (Apache License at the time of this writing) and everyone can host it, and the company behind Chef is also proposing their own hosted version, with added features and support. In a file named scripts/join_swarm.sh MASTER=$1
SWARM_TOKEN=$(cat worker.token) docker swarm join --token ${SWARM_TOKEN}$ ${MASTER}:2377 Now we just have to send this file to the nodes using the file provisioner: provisioner "file" { source = "scripts/join_swarm.sh" } Use it as a last step through a remote-exec provisioner, sending the local Docker
master IP (${packet device.swarm master.network.2.address}") as an argument to the script: provisioner "remote-exec" { connection { user = "${file("${var.ssh key}")}" } 162 Chapter 4 inline = ["chmod +x join swarm.sh", "./join swarm.sh", ".
Redis installation. This is sent by the Chef server by request from the chef client. Whenever it was possible or relevant, we tried to show alternative ways of doing the same thing with another tool or approach, so that everyone with any prior knowledge of the subject can still find something to learn in any section of the book. Don't forget to remove all
files from the folder beforehand. ==> ubuntu-1604-x64: Destroying VM and associated drives... Vagrant supports multi-machine setups, letting us define profiles for each one of them. This will now make the Docker Hub follow the tags we set on Git. Let's transform the default.rb recipe: node.override['chef client']['interval'] = '900' include recipe
'chef-client' Bump the version in metadata.rb, upload the new version, wait for the new chef-client execution, or launch it yourself to save some time. To do this, let's use the google compute firewall network = "default" allow { protocol = "tcp" ports = ["80"] }
source ranges = ["0.0.0.0/0"] target tags = ["www"] } Load balancing Google Compute instances To load balance requests across our two instances, we'll need to create a pool of hosts, where membership will be handled by a simple health check: an HTTP GET on / every second, with an immediate timeout (1 second), and removal after 3 errors.
Livery Place 35 Livery Street Birmingham B3 2PB, UK. We could solve those dependencies by hand, but we have a better idea: use the Berksfile! Let's start by replacing our dependency on our own mariadb cookbook with this cookbook, in mysite/Berksfile: Find the following code: cookbook 'mariadb', path: '../mariadb' Replace the previous code with
the following: cookbook 'mysql', '8.0.4' 240 Chapter 6 This way, we ensure we'll ever only run this particular cookbook version (8.0.4) and not a new one that might break things in production. We can build our own, either for testing purposes or for production use (with the maintenance overhead that goes with it), or we can use Hosted Chef, the Chef
server hosted by the company who wrote Chef. After a while, in the Repository Tags tab of our image, we'll get a SECURITY SCAN summary: In this example, we have issues to investigate further: 392 Chapter 10 Many vulnerabilities are displayed, but don't be frightened. Let's try this in instances.tf: resource "openstack compute instance v2"
 "gitlab" { name = "gitlab" region = "{\text{compute secgroup v2.ssh.name}}" image name = "{\text{compute secgroup v2.ssh.name}}" flavor name = "{\text{compute secgroup v2.ssh.name}}", "{\text{compute secgroup v2.ssh.name}}" flavor name = "{\text{compute secgroup v2.ssh.name}}" for attach the block storage volume
 we created, we need to add a volume {} block inside the resource: volume id = "${openstack blockstorage_volume v2.docker.id}" device = "/dev/vdb" } Now, an optional but fun part is that the commands needed to format the volume, mount it at the right place, fully update the system, install Docker, and run the GitLab CE container.
When a node is requesting its configuration from the server, the server compiles a catalog of this node, and the Puppet agent applies this catalog. Use any service that works for you: GitLab, GitHub, BitBucket, and others. Now that APIs are everywhere in the datacenter, automation took over every aspect and every step of what used to be a sysadmin
or IT job—infrastructure is now basically code, and should be considered as such while working alone in development or in production within a distributed team. Setting the production environment. Ansible can use some dynamic
inventories with AWS by itself, but we'll see here how to use a template in Terraform and dynamically fill in the required information so we end up with a working Ansible setup, thanks to Terraform and dynamically fill in the required information so we end up with a working and new) in a dedicated environment, how to set cookboo
constraints, and finally how to override the attributes in each environment. In the case of Chef, we have a great tool to help us for this quick introduction on how to use Docker. We'd like an index.html page with some of this
 information, so let's edit the one we already have from the apache cookbook; however, you can start from scratch as well. Here are the steps to enable chef-client: 288 Chapter 8 1. We can discover dependencies manually by installing modules using puppet module install as we did earlier. You can upgrade to the eBook version at www.PacktPub.com
and as a print book customer, you are entitled to a discount on the eBook copy. Acceptance directory. etcd.io/new?size=3. Configuring a proper Consul in cluster for production is out of the scope of this book, but if you don't have a Consul cluster at hand to try this out, here's a way to quickly have one,
using Docker and a Consul image: $ docker run -it --rm -p 8400:8400 -p 8500:8500 -p 8600:53/udp -h node1 progrium/consul -server -bootstrap Now let's configure our Terraform remote config -backend=consul -server -bootstrap Now let's configure our Terraform remote for Consul, and name it terraform/my_customer, so we can manage multiple customers simultaneously: $ terraform remote config -backend=consul -server -bootstrap Now let's configure our Terraform remote for Consul, and name it terraform/my_customer, so we can manage multiple customers simultaneously: $ terraform remote config -backend=consul -server -bootstrap Now let's configure our Terraform remote for Consul, and name it terraform/my_customer, so we can manage multiple customers simultaneously: $ terraform remote config -backend=consul -server -bootstrap Now let's configure our Terraform remote for Consul -server -bootstrap Now let's configure our Terraform remote for Consul -server -bootstrap Now let's configure our Terraform remote for Consul -server -bootstrap Now let's configure our Terraform remote for Consul -server -bootstrap Now let's configure our Terraform remote for Consul -server -bootstrap Now let's configure our Terraform remote for Consul -server -bootstrap Now let's configure our Terraform remote for Consul -server -bootstrap Now let's configure our Terraform remote for Consul -server -bootstrap Now let's configure our Terraform remote -bootstrap Now let's -boots
backendconfig="path=terraform/my_customer" Remote state management enabled Remote state configured and pulled. How would be on each of them. To test the feature, you can create in VirtualBox a network (like iac-lab) and assign it the network 192.168.50.0/24. There's an even
higher probability we'll need a Unix account very early in the process, even before the proper configuration management tool enters the game. Fortunately, there is! It's creatively named AmazonEC2ReadOnlyAccess. This may take a few minutes. To create the data bag entry on the Chef server, use the following knife command: $ knife data bag
create webusers 3. Start it exactly like the first one, with a description, context, and an initial test for a valid Chef run: require 'spec_helper' described recipe' describ
end it 'converges successfully' do expect { chef run }.to not raise error end end Execute RSpec and see the coverage fall from 100% to 40%. Getting ready To step through this recipe, you will need the following: ff A working Terraform installation ff An AWS provider configured in Terraform (refer to the previous recipes) ff An Internet
connection ff A Docker Engine running on Linux for the optional usage demonstration How to do it... Let's say we want the log group to be named docker_logs, and that we want to keep those logs for seven days. The classical usage is against an already running Docker server on the network, but it will work exactly the same locally with your own
Docker installation. Welcome. Environments also have a set of characteristics set. Using systemd Another useful way to launch containers is through the use of systemd. As we're using Test Kitchen with Chef, it starts by installing Chef and then resolves cookbook dependencies for us. We can now terraform apply this setup. Let's create a vhost epp file
in the modules/apache/templates directory: ServerName DocumentRoot Order deny, allow Override All ErrorLog /var/log/apache2/error-.log CustomLog /var/log/apache2/error-.log CustomLog /var/log/apache2/error-log CustomLog /var/log/apache2/error-.log /var/log/apache2/error-.log /var/log /var/lo
Terraform installation An AWS provider configured in Terraform (refer to the previous recipes) An Internet connection Chapter 2 How to do it... Let's start with a simple case: two members of a team (Mary and Joe) need to access resources on AWS. Plus, it includes fantastic Linux container support. We'll take a look at the CoreOS specificities, as we
can refer to earlier tips on how to manage users, files, authorized SSH keys, and other standard cloud-init directives. If those aren't enough, you can find all AWS Managed IAM Policies at . ISBN 978-1-78646-491-0 www.packtpub.com Credits Authors Stephane Jourdan Project Coordinator Kinjal Bari Pierre Pomès
Proofreader Reviewer Safis Editing Pierre Mavro Indexer Commissioning Editor Mariammal Chettiyar Kartikey Pandey Graphics Acquisition Editors Shantanu N. Let's define some defaults in the variable "image_name" { 149 Automating Complete
Infrastructures with Terraform default = "CentOS" description = "OpenStack image to boot" } Variable "flavor name" { default = "some flavor" description = "OpenStack image to boot" } Variable "flavor name" { default = "some flavor" description = "OpenStack image to boot" } Variable "flavor name" { default = "some flavor" description = "OpenStack image to boot" } Variable "flavor name" { default = "some flavor" description = "OpenStack image to boot" } Variable "flavor name" { default = "some flavor" description = "OpenStack image to boot" } Variable "flavor name" { default = "some flavor" description = "OpenStack image to boot" } Variable "flavor name" { default = "some flavor" description = "openStack image to boot" } Variable "flavor name" { default = "some flavor" description = "openStack image to boot" } Variable "flavor name" { default = "some flavor" description = "openStack image to boot" } Variable "flavor name" { default = "some flavor" description = "openStack image to boot" } Variable "flavor name" { default = "some flavor" description = "openStack image to boot" } Variable "flavor name" { default = "some flavor" description = "openStack image to boot" } Variable "flavor name" { default = "some flavor" description = "openStack image to boot" } Variable "flavor name" { default = "some flavor" description = "openStack image to boot" } Variable "flavor name" { default = "some flavor" description = "openStack image to boot" } Variable "flavor name" { default = "some flavor" description = "openStack image to boot" } Variable { default = "some flavor" description = "openStack image to boot" } Variable { default = "some flavor" description = "openStack image to boot" } Variable { default = "some flavor" description = "openStack image to boot" } Variable { default = "some flavor" description = "openStack image to boot" } Variable { default = "some flavor" description = "openStack image to boot" } Variable { default = "some flavor" description = "openStack image to boot" } Variable { default = "some flavor" d
resource named openstack_compute_instance_v2. This leaves us with one result—ami-ee6b189d: Start by declaring this variable in the variables.tf file started in the first recipe, using a default value corresponding to the AMI ID we found previously: variable "ami" { default = "ami-ee6b189d" } Now let's declare the instance type, specifying it as a
default: variable "aws_instance type" { default = "t2.micro" } Let's use those variables to create the Terraform aws_instance resource. Here are the users and their groups: GitHub username GitHub team name Membership level Team privacy John Documentation member closed Jane Engineering admin secret Here's the policy we decided concerning
the Git repository named infrastructurerepository: GitHub team name Repository permissions Documentation pull Engineering admin Configuring GitHub Let's start by creating a github provider, as we used an aws provider for AWS in the previous recipes. An example of such a cookbook could be an application cookbook named MyCloudApp, with
calls to, or inclusions of, other cookbooks such as Apache, MySQL, or any other cookbook it might need. Quickly simulating changes We now wonder what would happen if we were to change the number of instances. Getting ready To step through this recipe, you will need the following: ff A working Terraform installation ff An AWS account with an
SSH key configured in Terraform (refer to the Chapter 2, Provisioning IaaS with Terraform, recipes) ff An Internet connection 82 Chapter 3 How to do it... Using infrastructure-as-code, the easiest thing is to simply duplicate the code to create as many environments as needed. As it's to store releases, let's call it releases, using the
openstack objectstorage container v1 resource, in an objectstorage container v1 releases" { region = "${var.region}} 152 Chapter 4 name = "releases" { value = "Container name: output in outputs in output in outputs in output in outputs in outputs in outputs in outputs in outputs in output in outputs in output in outputs in output in outputs in output in outpu
${openstack_objectstorage_container_v1.releases.name}" } Applying In the end, do a terraform apply: $ terraf
 enjoy the runners (after GitLab token registration)! Managing Heroku apps and add-ons using Terraform Heroku is a popular Platform-as-a-Service (PaaS), where you have absolutely no control over the infrastructure. These repositories are called Puppet Collections. For each node, we can specify which module(s) to use to perform the complete node
installation. Intercepting errors in tests Sometimes we have to work with third-party cookbooks, that may somehow raise errors. Systems maintained with Chef have the opportunity to be converged at a predetermined time, either through a chef-client daemon or a cron job. To be a little bit more comprehensive, we can use the full block to do the
same: package "httpd" do action :install end Uploading the cookbook Still from inside the Chef repository, we now need to upload this new apache cookbook to the Chef server, so our servers can access it. Here's an example from a container running in the default network: # curl -I --connect-timeout 5 curl: (28) Connection timed out after 5003
milliseconds 352 Chapter 9 However, connecting from a container in the same network is allowed and working as expected: # curl -I HTTP/1.1 200 OK Connecting multiple networks for one container in the same network for one container in the same network for one container in the same network is allowed and working as expected: # curl -I HTTP/1.1 200 OK Connecting multiple networks for one container in the same network is allowed and working as expected: # curl -I HTTP/1.1 200 OK Connecting multiple networks for one container in the same network is allowed and working as expected: # curl -I HTTP/1.1 200 OK Connecting multiple networks for one container in the same network is allowed and working as expected: # curl -I HTTP/1.1 200 OK Connecting multiple networks for one container in the same network is allowed and working as expected: # curl -I HTTP/1.1 200 OK Connecting multiple networks for one container in the same network is allowed and working as expected: # curl -I HTTP/1.1 200 OK Connecting multiple networks for one container in the same network is allowed and working as expected: # curl -I HTTP/1.1 200 OK Connecting multiple networks for one container in the same network is allowed and the same networ
different networking configurations. Generate a new template for the htpasswd file, using the chef command: $ chef generate template cookbooks/mysite htpasswd 232 Chapter 6 5. As roles in Chef have a dedicated run list, it's common to see a role include the product recipes, and everything related to it, think monitoring for example. json Now we
can define the full VM like the others with the modified data bag path for the encrypted version: config.vm.define 'mysite', autostart: false do |mysite| mysite.vm.box = 'bento/centos-7.2' mysite.vm.box = 'bento-person-1.2' my
chef.roles path = 'roles' chef.nodes path = 'roles' chef.environment = 'production' chef.add recipe 'mysite: default' end mysite.berksfile path = 'cookbooks/mysite/Berksfile' mysite.berksfile path = 'roles' chef.environment = 'production' chef.add recipe 'mysite: default' end mysite.berksfile path = 'roles' chef.environment = 'production' chef.add recipe 'mysite: default' end mysite.berksfile path = 'roles' chef.environment = 'production' chef.add recipe 'mysite: default' end mysite.berksfile path = 'roles' chef.environment = 'production' chef.add recipe 'mysite: default' end mysite.berksfile path = 'roles' chef.environment = 'production' chef.add recipe 'mysite: default' end mysite.berksfile path = 'roles' chef.environment = 'production' chef.add recipe 'mysite: default' end mysite.berksfile path = 'roles' chef.environment = 'production' chef.add recipe 'mysite: default' end mysite.berksfile path = 'roles' chef.environment = 'production' chef.add recipe 'mysite: default' end mysite.berksfile 'mysite: default' end my
mysite 333 Maintaining Systems Using Chef and Puppet Put these commands (with their destroy counterparts) in the CI or whatever system you prefer at a regular interval, like daily or weekly, for each and every automated part of the infrastructure. In other cases, using the image from our favorite programming language might also be a good idea
 or not. In this section, we'll structure everything needed to properly test our Chef cookbooks code using Vagrant with CentOS 7.2 The Test Kitchen version in use in the Chef and Puppet Getting ready To step through this recipe, you will need the
following: ff A working Chef DK installation on the workstation ff A working Vagrant installation on the workstation ff The Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef code from Chapter 6, Fundamentals of Managing Servers with Chef code from Chapter 6, Fundamentals of Managing Servers with Chef code from Chapter 6, Fundamentals of Managing Servers with Chef code from Chapter 6, Fundamentals of Managing Servers with Chef code from Chapter 6, Fundamentals of Managing Servers with Chef code from Chapter 6, Fundamentals of Managing Servers with Chef code from Chapter 6, Fundamentals of Managing Servers with Chef code from Chapter 6, Fundamentals of Managing Servers with Chef code from Chapter 6, Fundamentals of Managing Servers with Chef code from Chapter 6, Fundamentals with Chef code from Chapter 6, Fundamentals with Chef code from Chapter 6, Fundamentals with Chef code
(and all beta versions) are available, and each is also available with a different flavor, that is, based on Alpine (php:7-apache), or Debian Jessie with Apache (php:7-apache), or Debian Jessie with Ap
notice an official MySQL cookbook maintained by the Chef team, currently at version 8.0.4: . We'll implement the same idea, except that Vagrantfile will include multiple machine profiles: one to start a virtual machine only with the webserver role, another to deploy only the database part, and the third one to launch everything together, including the
web application. Let's use the packet ssh key "admin" { name = "admin key" public key = "${file("${var.ssh key}.pub")}" } Bootstraping a Docker Swarm manager on Packet using Terraform We'll create two types of servers for this Docker Swarm cluster: managers
and nodes. Add it right now in variables.tf so we can use it: variable "allowed network" { default = "1.2.3.4/32" description = "The Whitelisted Corporate Network" } Don't forget to override with your own network in terraform.tfvars. To enable the S3 backend, only a few variables need to be set in the docker-compose.yml file: the AWS region to
contact, the keys, and the bucket name. Labels are purely informative when tags can be requested directly to make images distinct from an operational point of view. remote: Compressing objects: 100% (2/2), done. GitHub (https:// github.com) doesn't have free private repositories, but GitLab () or BitBucket () do. We're now basically able to deploy
any WordPress installation from scratch, at will, and in seconds, again and again. A few seconds later, the tests pass successfully, assuring us the build is consistent with our expectations. [Description] (#description) 1. This way, we ensure that the code we write is already tested, that it really covers the tested area; if some regression was to happen
someday, it would be immediately noticed. Getting ready To step through this recipe, you will need the following: ff A working Terraform installation ff An Internet connection How to do it... We'll intentionally write a simple Terraform installation ff An Internet connection How to do it... We'll intentionally write a simple Terraform installation ff An Internet connection How to do it... We'll intentionally write a simple Terraform installation ff An Internet connection How to do it... We'll intentionally write a simple Terraform installation ff An Internet connection How to do it... We'll intentionally write a simple Terraform installation ff An Internet connection How to do it... We'll intentionally write a simple Terraform installation ff An Internet connection How to do it... We'll intentionally write a simple Terraform installation ff An Internet connection How to do it... We'll intentionally write a simple Terraform installation ff An Internet connection How to do it... We'll intentionally write a simple Terraform installation ff An Internet connection How to do it... We'll intentionally write a simple Terraform installation ff An Internet connection How to do it... We'll intentionally write a simple Terraform installation ff An Internet connection How to do it... We'll intentionally write a simple Terraform installation ff An Internet connection How to do it... We'll intentionally write a simple Terraform installation ff An Internet connection How to do it... We'll intentionally write a simple Terraform installation ff An Internet connection How to do it... We'll intentionally write a simple Terraform installation ff An Internet connection How to do it... We'll intentionally write a simple Terraform installation ff An Internet connection How to do it... We'll intentionally write a simple Terraform installation ff An Internet connection How to do it... We'll intentionally write a simple Terraform installation ff An Internet connection How to do it... We'll intention how to do it... We'll intention how to
AWS, but also to the more common syslog. If one of our bootstrap scripts needs wget or curl and nmap, let's ensure those are present long before the proper configuration management tool starts its job (such as Chef or Puppet). @ULHPC mysql aco-mysql yumrepo Puppet module... fingerprint: "" => "" key name: "" => "admin key" public key: "" =>
 "ssh-rsa AAAAB3[...]" aws key pair.admin key: Creation complete Apply complete! Resources: 1 added, 0 changed, 0 destroyed. Before writing any test, we also need a helper script that will be common to all tests. pro.yaml in the Hiera tree: $ cd puppetcode/hieradata $ mkdir nodes $ cat > nodes/web.pomes.pro.yaml webusers: - id: john htpasswd:
$apr1$AUI2Y5pj$0v0PaSlLfc6QxZx1Vx5Se - id: mary htpasswd: $apr1$eR7H0C5r$OrhOQUTXfUEIdvWyeGGGy/ ^D This file now contains an array of hashes for authorized users, for the node web.pomes.pro. Here's a screenshot of a really vulnerable container and the Quay.io scanner giving helpful advice on the available fixes: Quay.io Security
Scanner will also send reminders by e-mail with a summary of the vulnerabilities found on all the containers it hosts on our account. In our case, it's named vps-ssd-1. We'll start by bootstrapping the Docker Swarm manager server, using the Packet service (more alternatives are available from Packet API): ff We want the cheapest server
(baremetal 0) ff We want the servers in Amsterdam (ams1) ff We want the servers to run Ubuntu 16.04 (ubuntu 16.04 (ubuntu 16.04 image) ff Default SSH user is root ff Billing will be hourly, but that can be monthly as well Let's put generic information in variables.tf so we can manipulate them: variable "facility" { default = "ewr1" description = "Packet facility" }
(us-east=ewr1, us-west=sjc1, euwest=ams1)" } variable "plan" { default = "coreos stable" description = "Packet machine type" } variable "operating system" }
terraform.tfvars to match our values: facility = "ams1" operating_system = "ubuntu_16_04_image" To create a server with Packet, let's use the packet_device resource, specifying the chosen plan, facility, operating system, billing, and the project in which it will run: resource "packet device" "swarm master" { hostname = "swarm-master" plan = "swarm-m
"${var.plan}" facility = "${var.facility}" operating system = "${var.operating system}" billing cycle = "hourly" project id = "${packet project.swarm.id}" } Now, let's create two scripts that will execute when the server is ready. 13 Vagrant Development Environments Using NGINX Docker container through Vagrant Let's start with the simplest
Vagrantfile possible, using the nginx:stable container with the Docker Vagrant provider: Vagrant config.vm.provider docker do docker docker up message = "HTTP access: config.vm.provider docker up message = "HTTP access: config.vm.provider up message = "HTTP access: config.vm.provider up message = "https://docker.up u
provider=docker Bringing machine 'default' up with 'docker' provider... Now our server on a CentOS 7.2 server, PHP, and MariaDB to build a classic LAMP server on a CentOS 7.2 server. Packt Publishing has endeavored to provide trademark information about all of the
companies and products mentioned in this book by the appropriate use of capitals. We want to ensure the cluster never breaks, using the etcd-lock strategy, and be sure the stable release is used: coreos/update.conf file: $ cat /etc/coreos/update.conf
GROUP=stable REBOOT STRATEGY=etcd-lock We can force an update check to verify it's working (sample taken from a system with an update engine client.cc(243)] Initiating update check and install. In this case, you can simply tell Vagrant to disable the auto-update
feature right from the Vagrantfile: config.vbguest.auto update = false 5. Shutting down the container just revokes everything and we're done with it. Using the Docker run Here's a simple way to execute the Tomcat 9 container that uses the awslogs driver. But I understand replacing the simple terraform command by something such as docker run -it
--rm -v `pwd`:/ data sjourdan/terraform:0.7.3 can feel not so appealing. A simple Git repository, such as SSH private keys or passwords. See also ff Ohai documentation at ff Puppet facts at builtin vars.html Automating application deployment (a
WordPress example) Chef can also be used to deploy applications from code repositories. We want the website's root folder to be /var/www/ and the configuration file will live under /etc/httpd/conf.d/.conf. So if our wish is to create a template like this
Running in mode. Now when requesting which images are available, we can see they all have the same image ID, but with different tags. Let's add a very similar command to the date output in the bootcmd directive: runcmd: - 'echo runcmd started at $(date)' Start a new instance, and observe the difference in timestamps: $ grep "started at"
/var/log/cloud-init-output.log bootcmd started at Fri Sep 23 07:02:47 UTC 2016 The runcmd started at Fri Sep
the reference again. For example, if you want to provision with Ansible the new virtual machine launched by Terraform, chances are you'll need to populate a hosts file with the public IP address of this host locally on your laptop. Commands end with; or \quad capacity and the provision with Ansible the new virtual machine launched by Terraform, chances are you'll need to populate a hosts file with the public IP address of this host locally on your laptop.
a free Hosted Chef account, so we can start coding with Chef as soon as possible and not worry about the server part. He also wrote a book on MariaDB named
consistent and homogenous code, so we can achieve more guickly a better guality and a higher level of standardization of our code. We, as a team, always know that we're ready to redeploy a system if required. [...] wordpress 1 | [Thu Dec 01 20:51:17.865932 2016] [mpm prefork:notice] [pid 1] AH00163: Apache/2.4.10 (Debian) PHP/5.6.28
configured -- resuming normal operations wordpress 1 [Thu Dec 01 20:51:17.865980 2016] [core:notice] [pid 1] AH00094: Command line: 'apache2 -D FOREGROUND' Let's verify we can connect to the WordPress HTTP server locally, on the redirected port 8080: $ curl -IL HTTP/1.1 302 Found [...] HTTP/1.1 200 OK [...] More information can be seen
using the ps command: $ docker-compose ps Name Command State Ports --
 --1basics mysql 1 3306/tcp docker-entrypoint.sh mysqld Up 1basics wordpress 1 0.0.0.0:8080->80/tcp docker-entrypoint.sh apach ... This is also very useful to maybe catch something wrong with a container acting weird:
406 Chapter 10 Another very powerful tool from sysdig is F6/Dig. When we pull or push an image, it's from the Docker registry. In this example, we'll deploy a CentOS 7 server, with NGINX, PHP-FPM, and MariaDB, all the PHP dependencies, and install Composer. Security and confidentiality will start becoming an issue, and maybe those passwords
and secret keys should not be stored in clear text at all. Delta compression using up to 4 threads. Let's add a tag right now, so we can use it later: tags = ["www"] We have to configure networking. Fortunately, r10k (which we already used in Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, to install modules) can be used to
create environments from a Git repository. Create the variables to define our credentials file "account.json" description = "API credentials file" { default = "account.json" description = "Region name" description = "Region na
 } variable "project name" { default = "default = "default-project" description = "Project ID to use" } Don't forget to override those values in terraform.tfvars if you want to: project name = "iac-book-infra" region = "us-east1" Now, in a providers.tf file, add the google provider: provider "google" { credentials = "${file("${var.credentials file}")}" project =
"${var.project_name}" region = "${var.region}" } Our google provider is now configured! Creating Google Compute HTTP instances Here's the checklist of our requirements for these HTTP hosts: ff 138 We want two of them Chapter 4 ff Their type is n1-standard-1 (3.75 GB of RAM, one vCPU) ff Their region and zone is: us-east1-d ff They are
running CentOS 7.2 (official image is: centos-cloud/centos 7) ff The default SSH username is centos ff The SSH key known to us is (keys/admin key) ff We want a fully updated system with Docker installed and running Let's define generic variables for all these requirements in a variable "machine type" { default = "f1-micro"
description = "Machine type" } variable "sone" { default = "c" description = "SSH key" } variable "ssh username" { default = "cont" description = "The SSH username to use" }
variable "www_servers" { default = "2" description = "Amount of www servers" } Now let's override in terraform Google Cloud instances are called from Terraform using the resource
google compute instance: Let's add what we already know in this resource: resource "google compute instance" "www" { count = "${var.www servers}" name = "www-${count.index+1}" machine type = "${var.www servers}" name = "www-${count.index+1}" name = "www-${co
 "${var.ssh username}:${file("${var.ssh key}.pub")}" } } This could be enough, but we want to go much farther. Create the recipe like this from chef-repo: $ chef generate recipe everything on MySQL so that WordPress can be installed, and
finally deploy the WordPress code from GitHub. Getting ready To step through this recipe, you will need the following: ff A working Terraform (refer to the previous recipe) ff An Internet connection How to do it... The resource we're using is called aws_security_group. In addition to this, security
checks should be part of the mandatory release process and the logs easy to access, even if the application is scaled on multiple machines. If the folder doesn't exist, create it: $ mkdir environment, follow these steps: 1. We want the public IP of the host so we can interact with
fleet right from our workstation. (for Ansible installation) srv-1: Running ansible-playbook... It's an incredible tool that each and every person working with code, we know how important debugging tools are. Manually, we would have typed the
following: $ dnf install httpd $ yum install httpd $ yum install httpd $ yum install httpd $ 2016-12-01 20:51:15 139820361766848 [Note] mysqld: ready for connections. This time
we'll use a CoreOS AMI for a change. Using the Docker Swarm cluster Using our Docker Swarm cluster is out of the scope of this book, but now we have it, let's take a quick look to scale a container to the thousands! Verify we have our 3 nodes: # docker node is ID AVAILABILITY MANAGER STATUS 9sxqi2f1pywmofqf63l84n7ps * Active Leader
ag07nh1wzsbsvnef98sqf5agy Active cppk5ja4spysu6opdov9f3x8h Active HOSTNAME STATUS swarm-node-1.local.lan Ready swarm-node-2.local.lan Ready warm-node-2.local.lan Ready swarm-node-2.local.lan Ready swarm-node-1.local.lan Ready swarm-node-1.local.lan
dynamically is a very common situation; system paths can be dynamic, users and passwords can be contextual, third-party credentials will be different in development and in production, slaves will join their masters, cluster members will find other nodes, and most other similar changing elements will need to
adapt at runtime. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews. How to do it... Our objective is to create a temporary, dynamic SSH server in a Docker
container, with credentials we can't know in advance. Under the cookbooks/mysite/ directory, we find a file named Berksfile (if you didn't create the following exported attribute: ${packet device.swarm master.network.2.address}.
Let's create it in spec/spec helper.rb. If you plan to let cAdvisor run permanently, it is a good idea to restrict access using simple HTTP authentication. Fortunately, the define a block of code that can be used multiple times. It currently handles Debian, Ubuntu, Alpine, Oracle, and Red Hat security data sources. Getting
ready To step through this recipe, you will need the following: ff A working Docker installation ff A sample HTTP server binary (sample code included) How to do it... To make a container network port available to others, it first needs to be exposed. To illustrate this, let's modify our index.php file (manifests/site.pp), both in production and test: file
```

```
{"${docroot}/index.php": ensure => present, owner => 'www-data', group => 'www-data', mode => '0644', content => "" } We can now switch between test and production' ... @@ -1 +1 @@ - + ... 296 Chapter 8 # puppet agent --test --environment test Info
 Using configured environment 'test' @@ -1 +1 @@ - + The dynamic way - r10k We edited the environments and code directly in the keys directory, but as you might prefer something else, let's use a variable for that as well. Hiera is used to
store data out of manifests, and is based on a hierarchy to look up data. Let's say our IT security policy wants us to have an emergency user account in a group named infosec for the IT security team with passwordless sudo rights and the simple /bin/sh shell. Otherwise, import the encrypted version from the Chef server to a new directory, say, in
JSON (using -Fj): $ mkdir data bags encrypted $ knife data bags encrypted $ knife data bag show aws us-east-1 -Fj > data bags encrypted/us-east-1. This is something you have to answer according to your needs and environment. See also ff RSpec at ff Docker-api at ff ServerSpec at 379 Maintaining Docker Containers The workflow for creating automated Docker builds from
Git Building local containers is a nice thing to do, but what about its wide distribution? This file is critical: it's the stored state of your infrastructure. The plan looks good—it seems to want to create an S3 bucket named the way we want! Let's terraform apply this and move on. Chances are you're using Chef in production. An OpenStack account on
any OpenStack provider (public or private). Using multiple nodesets, we can also ensure tests can be validated on multiple platforms. We can start by declaring what's usually known as the AWS_DEFAULT_REGION, AWS_ACCESS_KEY_ID, and AWS_SECRET_ACCESS_KEY_ID, and AWS_SECRET_ACCE
Access Key" } variable "aws secret Key" } variable "aws secret Key" } variable "aws region" { default = "eu-west-1" description = "AWS Region" } Set the two variables matching the AWS account in the terraform.tfvars file. Testing services status The default recipe enables and starts the httpd service. ----> Converging ... Let's now write these
tests! 276 Chapter 7 We'll write integrations tests for the mysite cookbook written in Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, for demonstration purposes, but those are completely generic and can be reused anywhere. But the concepts still apply to most languages, and it's a good practice in production to create shared
folders and symlinks for long-lasting configurations and files. Getting ready To work through this recipe, you will need the following: ff An Internet connection ff A physical or virtual machine How to do it... The Chef DK can be downloaded from . Probably not. (ID: admin_key) aws instance.coreos: Refreshing state... Vagrant, Terraform, the Chef
Development Kit, and Docker also work on the Windows platform, although this is untested by the authors. Running Bash in an Ubuntu:16.04 (ubuntu:16.04 kun apt-get -yq update RUN apt-get install -yc
dnsutils curl netcat 337 Working with Docker Now build using the docker build command, passing it the name of the container with the -t option: $ docker build -t utils . We could have declared them simply with the following: variable name} If the variable
has been declared with a default, as our aws_region has been declared with a default of eu-west-1; this value will be used if there's no override in the terraform. We can use this target directly instead of puppet-lint manually as we did earlier. In this example, we'll use the public OpenStack by OVH, located in Montreal, QC (Canada), but we can use
any other OpenStack. Alternatively, you want people to have the choice, or you simply want both environments to work! We'll see how to build a single Vagrantfile to support them all. Let's ensure it's created and switch to it until the rest of the process: RUN mkdir -p /usr/src/app WORKDIR /usr/src/app To install all the declared dependencies, we
need to send both Gemfile and Gemfile. CURRENT_OP=UPDATE_STATUS_UPDATE_AVAILABLE [...] Configuring locksmith using cloud-init Now we're sure the update system is correctly triggered, we are facing a new problem: nodes from our cluster can reboot at any time when an update is available. An example cookbook can be mysql, and two
recipes from this cookbook can be mysql::server to manage a client. Maybe some are false positives, or maybe some rules are just not yet updated to the latest deprecation notices, such as the following: $ hadolint Dockerfile Dockerfile Dockerfile Dockerfile In fact, this Dockerfile is
following Docker 1.13 recommendations, which include to no more include a maintainer instruction. Let's create this directory inside our Chef repository for our revision control system (RCS, like git): $ mkdir -p data_bags/webusers 2. So we'll make sure all the parts of the final product can be redeployed from scratch (which is the main point). Very
often, a lot of environments are seen, such as staging, testing, alpha or beta. Three new resources are now untested, from the apache::virtualhost recipe: $ chef exec rspec --color [...] ChefSpec Coverage report generated... It is sometimes important to have as early as possible a correct date and time set up (for things like registration times, delays,
and other issues depending on dates and times). For simplicity and following recipes in this chapter, the auto-signing feature has been enabled. However, Packt Publishing cannot guarantee the accuracy of this information. Chapter 5, Provisioning the Last Mile with Cloud-Init, for in-depth details on cloud-init. In a simply YAML file, we can describe
 what services are needed to run our application (proxy, application, databases, and so on). Getting ready To step through this recipe, you will need: ff A working Chef DK installation on the workstation ff A working Chef DK installation on the
and Puppet, Chapter 7, Testing and Writing Better Infrastructure Code with Chef and Puppet, or any custom Chef code How to do it... We recommend that you create a cookbook, different from other cookbooks, dedicated to configuring the underlying host. The solution is to automatically deploy the VirtualBox Guest Additions on demand, through a
plugin. 1 root root 20 Oct /opt/chefdk/bin/chef 5 16:36 /usr/bin/chef 5 16:36 /usr/bin/c
2.3.1p112 (2016-04-26 revision 54768) [x86_64-linux] There's more... Starting from Puppet 4.x, Puppet Labs is providing repositories for both agent and server packages. id}"] tags { Name = "IAC Database in ${var.aws_region}" } } } } } } } }
outputs.tf: output "RDS" { value = "address: ${aws_db_instance.db.address}" } Let's terraform apply now and try the result: # psql -h -d iac_book_db => \l List of databases Name | Owner Access privileges | Encoding | Collate | Ctype |
 -70 Chapter 2 iac_book_db | dbadmin | UTF8 | en_US.UTF-8 | en_US.UTF-
 | template1 =c/dbadmin | UTF8 | en_US.UTF-8 | en_US.UTF-8 | en_US.UTF-8 | en_US.UTF-8 | | | dbadmin = CTc/dbadmin | | | (5 rows) There are many more useful options you can use or set, such as maintenance windows, backup retention periods, dedicated database subnets, storage encryption, and master/slave configuration. Starting from an Ubuntu image
 Ubuntu ships official images that are all tagged with both their release version and name: ubuntu:16.04 is equivalent to ubuntu:xenial. To set the sitename attribute with default value of defaultsite, add the following in this file: defaultsite, add the following in this file: defaultsite are all tagged with both their release version and name: ubuntu:xenial. To set the sitename attribute with default value of defaultsite, add the following in this file: defaultsite are all tagged with both their release version and name: ubuntu:xenial. To set the sitename attribute with default value of defaultsite.
virtualhost.rb file, with standard access rights. However, if we wanted to keep the default user from our Linux distribution, we'd just have to add the following default user from our Linux distribution, it's highly probable we'll
expect a fully updated system as soon as possible (think security patches and other bug fixes). The thing is, our life will become increasingly complicated with all those cookbooks downloaded here and there, each of them having dependencies of their own. We want to do two actions with our services: enable them at boot and start them right away.
One elegant solution to keep them well separated is to use the following option when executing the terraform command: $ terraform apply -state=staging.tfstate file. ff For the apache/manifests/htaccess.pp manifest: The unit test should ensure a .htaccess file is generated
correctly. Vagrant instance destroyed. 395 Maintaining Docker Containers The Docker daemon needs to run with the AWS credentials in the memory—it's not information we pass to containers, as it's handled by the Docker daemon's log driver. Here's the code to do this: config.berkshelf.berksfile path = "cookbooks/platform/Berksfile"
config.berkshelf.enabled = true Starting Vagrant at this point will just deploy everything from scratch: $ vagrant up [...] # Chef Client finished, 17/45 resources updated in 03 minutes 30 seconds If the run succeeds, meaning the code from the Docker role is applied, we're safe. Finally, add the validation client name for the Chef server: variable
 "chef validation client name" { default = "iacbook" } 6. Let's be careful about what we ship: 383 Maintaining Docker Containers Navigate to Build. If it's the first time you've launched CentOS 7 servers on AWS, you have to agree their
terms and conditions at . Using a data source will help us manipulate the image later: data "docker registry image" "ghost: 0.10" } data "docker registry image, let's exactly do that, using the docker image resource. We'll see here how we can
share the state using Git (a version control system where developers can store code), AWS S3 (an Amazon Web Services storage system using HTTP) or Consul (a tool for service discovery and a key-value store), chosen among many other solutions. You can find other AMI IDs at config.vm.define "srv-1" do |config| config.vm.provider :aws do |aws|
aws.ami = "ami-c06b1eb3" end config.vm.provision :shell, :path => "script.sh" end So what is the AWS-related information we need to fill in so Vagrant installation ff A working Vagrant installation ff A working VirtualBox or VMware installation ff An Internet
connection How to do it... Let's start with the simplest Vagrant environment we know: Vagrant.config.vm.hostname = "srv-1" do |config| config.vm.box = "bento/centos-7.2" config.vm.define "srv-1" do |config| config| config| config| config| co
reference, here's a simple NGINX configuration that will work well for us, listening on HTTP, serving files located on /srv/app/public, and using PHP-FPM (the file name is nginx.conf): events { worker connections 1024; } 30 Chapter 1 http { sendfile off; server { listen 80; server name ; root /srv/app/public; try files $uri $uri /index.php?
q=\uni\u00e4\u20e3args; index index.php; location / \{ try files \u00e4uri \u00e4rdex.php; location / \\u00e4try files \u00e4uri \u00e4rdex.php; location / \\u00e4rdex files \u00e4try files \u
fastcgi params; } } } Simple shell provisioning We'll create a provisioning we'll create a provision.sh, which contains all the steps we need to have a fully working Laravel environment. Upload it to the Chef server: $ knife environment from file environment from file environment.
for the main apache class, let's create spec/acceptances/ classes/apache spec.rb, with the following content: require 'spec helper acceptance' do pp = true) end end 281 Testing and Writing Better Infrastructure Code with Chef and Puppet The goals of this
test are as follows: ff ff ff Installing Apache using our class. As exciting as this step is, it's also unfortunately far beyond the cookbook that has passed steps 1 and 2, we just bump the cookbook's version number constraint, validating
the results in the staging environment without impacting production, and finally deploy it in production whenever possible. Get the most in-demand software skills with Mapt. Let's find out by testing our virtual hosting configuration with vhost spec.rb: describe file('/etc/httpd/conf.d/mysite.conf') do it { should exist } it { should be mode 644 }
its(:content) { should match /ServerName mysite/ } it { should be owned by 'root' } end 279 Testing and Writing Better Infrastructure Code with Chef and Puppet This actually proves the default attribute really got overridden by the mysite value, and the content of the virtual host configuration file also matches
this value. If you previously added it, it will launch it right away through the default hypervisor (in this case, VirtualBox), or if it's a new box, download it for you automatically. 6 Chapter 1 There's more... Using Vagrant plugins also extends what you can do with the Vagrant plugins also extends when you can do with the Vagrant plugins also extends when you can do with the Vagrant plugins also extends when you can do with the Vagrant plugins also extends when you can do with the Vagrant plugins also extends when you can do with the Vagrant plugins also extends when you can do with the Vagrant plugins also extends when you can do with the Vagrant plugins also extends when you can do with the Vagrant plugins also extends when you can do with the Vagrant plugins also extends when you can do with the Vagrant plugins also extends when you can do with the Vagrant plugins also extends when you can do with the Vagrant plugins also extends when you can do with the Vagrant plugins also extends when you can do
A working Docker installation ff A Rails application How to do it... Here are our standard requirements: ff This Rails application needs Ruby 2.3 ff All dependencies are handled by Bundler, and need to be installed in the container ff Node 5 is also needed ff We want assets to be precompiled in the image (putting them somewhere else is out of scope)
Here's how we'll proceed. We'll see how to use it with Terraform. Here's how we can test for this directory existence, along with its ownership parameters: it 'creates a virtualhost directory' do expect(chef run).to create directory existence, along with its ownership parameters: it 'creates a virtualhost directory' do expect(chef run).to create directory existence, along with its ownership parameters: it 'creates a virtualhost directory' do expect(chef run).to create directory existence, along with its ownership parameters: it 'creates a virtualhost directory' do expect(chef run).to create directory existence, along with its ownership parameters: it 'creates a virtualhost directory' do expect(chef run).to create directory existence, along with its ownership parameters: it 'creates a virtualhost directory' do expect(chef run).to create directory existence, along with its ownership parameters: it 'creates a virtualhost directory' do expect(chef run).to create directory existence, along with its ownership parameters: it 'creates a virtualhost directory' do expect(chef run).to create directory existence, along with its ownership parameters: it 'creates a virtualhost directory' do expect(chef run).to create directory existence, along with its ownership parameters: it 'creates a virtualhost directory' do expect(chef run).to create directory existence, along with its ownership parameters.
then creates an index file. In a TDD workflow, we can first write acceptance subdirectory of any module and then write the code itself. We'll cover different ways of anticipating, simulating, and targeting changes in our infrastructure, as an added safety check before applying the changes for good. Deploying Docker with
Chef There's a very nice cookbook, which is extremely well-documented, that does everything we need (. 67 Provisioning IaaS with Terraform Getting ready To step through this recipe, you will need the following: ff ff ff A working Terraform Getting ready To step through this recipe, you will need the following: ff ff ff A working Terraform Getting ready To step through this recipe, you will need the following: ff ff ff A working Terraform Getting ready To step through this recipe, you will need the following: ff ff ff A working Terraform Getting ready To step through this recipe, you will need the following: ff ff ff A working Terraform Getting ready To step through this recipe, you will need the following: ff ff ff A working Terraform Getting ready To step through this recipe, you will need the following: ff ff ff A working Terraform Getting ready To step through this recipe, you will need the following: ff ff ff ff A working Terraform Getting ready To step through this recipe, you will need the following: ff ff ff ff A working Terraform Getting ready To step through the first fir
How to do it... There are many parameters at play in a database deployment, even a simple one. This is an added value, ensuring quality in the process before shipping. $ curl -w "" ' We can specify the minimum required size of the CoreOS cluster by adding the size= argument to the URL. owner: root:root permissions: '0644' This machine, when
booted, will have /etc/motd in place and display the string at login: $ ssh ip Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 4.4.0-36-generic x86 64) [...] This server is configured using cloud-init. S3 stores "objects" in "buckets" and has no storage limit (one exception is the bucket name: it has to be unique on Amazon's S3, the
namespace being shared). Refer to the Vagrant chapter of this book for more information about the Vagrant tool! Our production servers are configured by the application of some Chef code, and currently, it does this job pretty well. Getting ready To step through this recipe, you will need the following: ff A working Chef DK installation on the
workstation ff A working Vagrant installation on the workstation ff The Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamentals of Managing Servers with Chef and Puppet, or any custom Chef code from Chapter 6, Fundamental 6, Fundamen
VM... 39 Vagrant Development Environments There's more... Here are more ways of controlling a Vagrant environment, and use even better Chef tooling inside it. We have four manifests in our module, and we are about to create a unit test for each of them. kitchen setup: This installs any additional plugin we might need. First published: February
2017 Production reference: 1150217 Published by Packt Publishing Ltd. Let's now launch this virtual machine on EC2, using the specified SSH key pair to log into it and placed inside the security group, so (in our case) SSH is only available from a specific IP address. Here's how it works: instead of specifying only the container name, or the combo
username/container name, we prefix both by the DNS name of the registry, for example, . A directory can similarly be tested like this in the same vhost spec.rb file; describe file('/var/www/mysite') do it { should be directory } end Another interesting test to be done is to check the content of the https://www.mysite') do it { should be directory } end Another interesting test to be done is to check the content of the https://www.mysite') do it { should be directory } end Another interesting test to be done is to check the content of the https://www.mysite') do it { should be directory } end Another interesting test to be done is to check the content of the https://www.mysite') do it { should be directory } end Another interesting test to be done is to check the content of the https://www.mysite') do it { should be directory } end Another interesting test to be done is to check the content of the https://www.mysite') do it { should be directory } end Another interesting test to be done is to check the content of the https://www.mysite') do it { should be directory } end Another interesting test to be done is to check the content of the https://www.mysite') do it { should be directory } end Another interesting test to be done is to check the content of the https://www.mysite') do it { should be directory } end { should be dir
Servers with Chef and Puppet, we wrote a recipe making a request to the Chef server for authorized users in a data bag. Since we have a dedicated cookbook for our app, let's try to add some customization to it. Start all the services: sudo systemctl enable php-fpm sudo systemctl enable nginx sudo systemctl enable nginx sudo systemctl enable nginx sudo systemctl enable php-fpm sudo systemctl enable nginx sudo systemctl
sudo systemctl enable mariadb sudo systemctl start mariadb Enable provisioning To enable provision.sh" Shared folder To share the src folder between your host and the Vagrant VM under /srv/app, you can add the following code:
config.vm.synced_folder "src/", "/srv/app" Public LAN Networking The last thing we need to do now is to add a network interface to our Vagrant virtual machine, that will be on the real LAN, so our coworkers will access it easily through the network "public network", bridge: "en0: Wi-Fi (AirPort)" Adapt the name of your network
 adapter to use (this was on a Mac, as you can guess) to your needs. There're many answers to that, and one crucial question to address is: how is transmitted or synchronized the Terraform state? We're reusing all the information our data source is exposing to us, such as the image name or its SHA256, so we know if a new image is available to pull:
resource "docker image" "ghost" { name = "${data.docker registry image.ghost.sha256 digest}" } resource "docker registry image.ghost.sha256 digest}" } Let's now
declare the private Ghost container (without any port mapping), using the docker container resource. Deploying a remote Docker server instead of the default local configuration from our workstation because of bandwidth issues, testing a production environment, maybe a customer
demonstration, or distant team collaboration. Open that apackage. However, it's not fixed forever: it's just the default behavior. We'll see two tools quite fit for the job: the first one is simply the one shipped with Docker itself,
and the second one is a totally different tool by Google named cAdvisor—a web user interface with a lot of useful and easy-to-get information. Luckily, AWS has a nice and easy log-aggregating service, named AWS CloudWatch. This template file will contain the virtual host Apache configuration for our site. Let's rewrite this Dockerfile using a one-
liner, to save some space: FROM ubuntu:16.04 RUN apt-get install -y apache2 && \ rm -rf /var/lib/apt/ ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"] This image is exactly the same, but is only around 220 MB. We can find more information about HCL here: . This means that another user ==> srv-1: simply has to
run `vagrant connect --ssh subtle-platypus-4976` ==> srv-1: to SSH to your Vagrant machine. Create a Docker Unix group. So luckily, we can simply fall back to a simple method that is based on a cron. The code is structured using modules. (listen-peer-urls). Add the following in a loadbalancer.tf file: resource "google compute forwarding rule"
"http" { name = "http-lb" target = "${google compute target pool.www.self link}" port range = "80" } Creating a Google MySQL database instance Our typical target application needs a database to store and access data. ChefSpec stops there, and say the Chef run is in error. Select the book for which you're looking to download the code files.
Default value: service class baseconfig ($ensureservice='running'; $enableservice=false; $ensurecron='absent' } 'cron': { $ensureservice=false; $ensurecron='absent' } 's enableservice=false; $ensurecron='absent' } 's ensurecron='absent' } 's ensurecron='absent' } 's ensurecron='absent' } 's ensurecron='absent' } 'cron': { $ensureservice=false; $ensurecron='absent' } 'cron': { $ensureservice='absent' } 'cron': {
 {'puppet': ensure => $ensureservice, enable => $ensureservice, enable => $ensureservice, } cron {'puppet-agent': ensure => 0, command => '/opt/puppet agent --onetime --no-daemonize --splaylimit 60', } } 292 Chapter 8 Now we can define the requested mode from the main manifest: node 'web.pomes.pro' { ... If you
manually navigate to your AWS account, in EC2 | Network & Security | Key Pairs, you'll now find your key: Another way to use our key with Terraform and AWS would be to read it directly from the file, and that would show us how to use file interpolation with Terraform. It's highly recommended that you stick with the rolling builds that are
continuously updated because these are tagged only with major versions such as centos:7. For Chef to know where this is to be found, we need to create a dependency to those cookbooks. ff An Internet connection. This gem is mainly used to install Puppet in boxes during tests: $ sudo puppet resource package beakerpuppet install helper
provider=puppet gem We first need to define a list of supported platforms for running test acceptances. Locally declared variable are available using the ${resource type.resource type.
 with Terraform ami instance_type key_name security_groups group.name}" ["${aws_security_group.base_security_ = true tags { Name = "Ubuntu launched by Terraform" } } Apply the following code: $ terraform apply
aws key pair.admin key: Creating... 6 files inspected, no offenses! Our code is clean. Your blog is now available on Ctrl+C to shut down Change the host IP from 127.0.0.1 to 0.0.0.0, port: '2368' } Restart the node server: cd /srv/blog &&creating... 6 files inspected, no offenses! Our code is clean. Your blog is now available on Ctrl+C to shut down Change the host IP from 127.0.0.1 to 0.0.0.0, port: '2368' }
sudo npm start --production You now have a direct access to the blog through your box LAN IP (adapt the IP to your case): . Getting ready To step through this recipe, you will need the following: ff ff ff A working Terraform installation An AWS account with an SSH key configured in Terraform and a security group allowing SSH connections from
outside (refer to the Chapter 2, Provisioning IaaS with Terraform recipes) An Internet connection How to do it... Let's start by launching a standard CentOS 7.2 on AWS with a standard set of variables in variables in variables.tf: variable "aws_centos_ami" { type = "map" default = { eu-west-1 = "ami-7abd0209" us-east-1 = "ami-6d1c2007" } } 89 Going Further
 with Terraform variable "aws instance type" { default = "t2.micro" description = "Instance Type" } Here's the simplest instances.tf file to launch the instance type = "${var.aws instance type}" key name = "${aws key pair.admin key.key name}" here's the simplest instance type = "${aws instance type}" key name = "${aws instance type}" key name = "${aws instance type}" key name = "${aws key pair.admin key.key name}" here's the simplest instances.tf file to launch the instance type = "${aws instance type}" key name = "${aws instance}" key name = "${a
security groups = ["${aws security group.base group.base group.bas
LABEL name="apache" LABEL maintainer="John Doe "LABEL maintainer="John Doe "LABEL wersion=1.0 RUN apt-get install-y --no-install-recommends apache2=2.4.1010+deb8u7 \ && apt-get install -y --no-install-recommends apache2=2.4.1010+deb8u7 \ && apt-g
docker-compose.yml file so the image can be built and run easily: version: '2' services: http: build: . Navigate to the Chef repository on your workstation, using an example vagrant user: $\$ knife bootstrap 1.2.3.4 -N my node hostname -x vagrant --sudo
This will first download the latest available Chef version and install it. Getting ready To step through this recipe, you will need: ff A working Docker installation ff A free Travis CI account How to do it... We'd like our RSpec integration tests to be executed automatically each time we commit a change on Git. Yes, he can: $\$\text{aws --profile iacbook-joe ec2}$
bootstrapping only the webserver role from scratch, we will need to do the following—setting paths for everything, including the specific Berksfile for the job: config.vm.define 'webserver', autostart: false do |ws| ws.vm.box = 'bento/centos-7.2' ws.vm.provision :chef zero do |chef| chef.cookbooks path = 'cookbooks' chef.environments path = '
vagrant up webserver To make sure our code is capable of bootstrapping only the database part of this platform from scratch, just execute the mysite::mysql recipe in a similar context: config.vm.define 'db', autostart: false do |db| db.vm.box = 'bento/centos-7.2' db.vm.provision :chef_zero do |chef| chef.cookbooks_path = 'cookbooks
chef.environments path = 'environments' chef.roles path = 'roles' chef.nodes path = 'roles' chef.environments' chef.environment
instead of having the cumulated space taken from the archive and its uncompressed content, the space taken is only from the uncompressed content alone. We now know the most useful configuration options to bootstrap automatically a CoreOS cluster using cloud-init. Create a feature branch for we're about to work on supporting docker in our
platform: $ git checkout -b docker support Infrastructure TDD - writing tests first Let's write our tests first so they will fail for sure, and we'll know we're building from there correctly. Zagade About the Authors Stephane Jourdan is a passionate infrastructure engineer, enthusiastic entrepreneur, zealous trainer, and continuous learner, working or
name, and size: output "Block Storage" { value = "${openstack_blockstorage_volume_v2.docker.name}, ${openstack_blockstorage_volume_v2.docker.name}, ${openstack_blockstorage_v2.docker.name}, ${openstack_blockstorage_v2.docker.name}, ${openstack_blockstorage_v2.docker.name}, ${openstack_blockstorage_v2.docker.name}, ${openstack_blo
30% smaller than the main one (80 MB compared to 126 MB at the time of writing). Let's install it: $ sudo puppet labs_spec_helper provider=puppet is the counterpart of ChefSpec for Puppet, and has been installed as a dependency of puppetlabs_spec_helper. It has a
multitude of options, and a full chapter about it wouldn't be enough. com/voxpupuli/puppet-hiera Using Chef Vault encryption A different way of encrypting data is proposed through Chef Vault, and this does not require you to include the key somewhere in the code. Other environments, for example test, should be created under
/etc/puppetlabs/code/environments/. So that after a change in the code, tests can be automatically launched and as time go by, complexity soars with many cases added, so you just don't have to think about it: it's all going to be tested, and if your change breaks something you missed, you'll know it in seconds. This index must remain unchanged
Under no circumstance should it replace a proper configuration management tool. In the meantime, the Foodcritic tool used in conjunction with rules checks our code for a set of commonly accepted good practices by the community. Other optional properties of the file resource include ownership information or the file mode. If you'd like to join our
388 Chapter 10 2. 5. Tips and tricks appear like this. A distinct Ruby-on-Rails Docker image used to exist, but is now deprecated in favor of the main Ruby Docker image. Let's create a variable "ssh_key" { default = "keys/admin_key" description = "Path to SSH key" } 157 Automating Complete Infrastructures with
Terraform Don't forget to override the value in terraform.tfvars if you use another name for the key. Starting from a Python image Python is officially distributed and many of its versions are supported as tagged Docker images. This is typically why it's important to understand why suggestions are only that—suggestions. Set the hostname If you want
to specify the VM name right from Vagrant, just add the following: config.vm.hostname = "vagrant-lab-1" This will also add an entry with the hostname to the /etc/host file. There's another command: terraform plan. Creating an automated ping monitoring test Let's create an initial test, a simple ICMP ping to a server whose IP is 1.2.3.4, every 5
minutes: resource "statuscake test" "latency" { website url = "1.2.3.4" test type = "PING" check rate = 300 paused = false } 126 Chapter 3 The website url can be a reference to an existing Terraform resource. 112 Chapter 3 For example, I always add a make destroy command, so I can easily
destroy a test infrastructure (be careful!) terraform-destroy ## Destroy (careful!) terraform-destroy (careful!) plan Plan changes validate Validate syntax and
format Also, it can be used like this: $ make plan env=staging $ make apply env=staging $ make a
default production setup. Much more can be done with this tool! See also ff The Chef Vault gem at ff The Chef Vault gem at ff The Chef Vault gem at ff The Chef Vault cookbook at Accessing and manipulating system information with Ohai. Beaker::Hypervisor, found some vagrant boxes to create Bringing machine
 'ubuntu-1604-x64' up with 'virtualbox' provider... Let's find out by writing the test for the httpd package in apache_spec.rb: require 'spec_helper' describe package is really installed! While writing integration tests, we strongly suggest that you use
Test Kitchen to create/converge/set up/verify the sequence and not the simple kitchen test command that does everything at once—the manual way is much faster! Similarly, testing for the php packages in a php_spec.rb file will look exactly the same: require 'spec_helper' describe package('php') do it { should be_installed } end describe
package('php-cli') do it { should be_installed } end describe package('php-mysql') do it { should be_installed } end 278 Chapter 7 Testing for service status. Thankfully, Identity and Access Management (IAM) is there just for that. Let's try the first one! Since the manifest contains a class
 declaration, the unit test should be in spec/classes. This is how we test it's created with the required ownership: it 'creates and index.html file' do expect(chef run).to create file('/var/www/default/index.html').with(user: 'root', group: 'root') end Code coverage is now 80%! 264 Chapter 7 Testing templates creation The recipe ends with the creation of
 Apache VirtualHost from a template. 156 Chapter 4 Getting ready To step through this recipe, you will need the following: ff A working Terraform installation ff A Packet.net account with an API key (an authentication token). To have our entries
 automatically populated in the htaccess file, we'll have to iterate through all existing entries. Let's use the chef command to do that: $ chef generate cookbook with our default recipe, we'll use the include recipe method in mysite/recipes/default.rb: include recipe include recipe
include_recipe include recipe include recipe "apache" "apache" "apache" "apache" "apache" "apache" "apache" "apache" to load and execute the content of each recipe. Chances are, we'll expect a lot more complicated setups in our real life infrastructure. Follow the indications from the official website to install the plugins. For the record, here's the variables.tf file: v
 "aws_region" { default = "eu-west-1" } 115 Going Further with Terraform variable "aws_ssh_admin_key_file" { default = "t2.micro" } variable "cluster_size" { default = "1" description = "Number of nodes in the cluster" } variable "aws_instance_type" { default = "t2.micro"
description = "Instance type" } Here's a deliberately badly formatted provider.tf: provider "aws" { region = "${var.aws region}" } Also, here's a CoreOS instance type = "${var.aws instance type}" key name =
"${aws_key_pair.admin_key.key_name}" associate_public_ip_address = true tags { Name = "coreos_${count.index+1}" } } Terraform code validation Let's be sure our code validates: $ terraform validate Thankfully, it does! 116 Chapter 3 Does this code plan to do what we want it to do? We know they have to be Ubuntu 16.04, and we decide on a
flavor name: a flavor is the type of the machine. We've already set up NGINX, so what about using it as a front reverse proxy with a blog engine such as Ghost behind it? This information is mapped like this in the cloud-config file: #cloud-config file: #cloud-config
MIIEowIBAAKCAQEAuR[...] ----END RSA PRIVATE KEY---- With this information, the initial chef-client run will be able to authenticate itself against the Chef organization and add the node. An Internet connection. Here's how it looks: coreos: units: - name: docker-tcp.socket command: start enable: true content: | [Unit] Description=Docker Socket fo
the API [Socket] ListenStream=2375 BindIPv6Only=both Service=docker.service [Install] WantedBy=sockets.target 189 Provisioning the Last Mile with Cloud-Init Let's start a remote server with this whole configuration and use it a little for the demonstration (in this example, the Docker remote host is 52.211.117.98, and we'll launch an nginx
container with HTTP port forwarding). service.d/20-cloudinit.conf: $ cat /run/system/spect.a.g. Environment="FLEET_METADATA=region=eu-west-1,provider=aws,cluster=myc luster" Environment="FLEET_METADATA=region=eu-west-1,provider=aws,cluster=myc luster=myc
the connection to the fleet cluster manager is working from the instance: $ fleetctl list-machines MACHINE IP 441bf02a... In this Berksfile, enter the following: source ' ' metadata cookbook 'mariadb', path: '../php' cookbook
tells us three important things: ff ff Where to find unknown cookbooks (on the official supermarket, we can replace with our own internal supermarket if we run one) Where to find dependencies: in our cookbook's metadata file Where each of those cookbooks reside: in our case, the local relative path Bump the mysite cookbook version in
metadata.rb so we don't mess with our previous work, and, from the mysite cookbook's dependencies at once: $ berks upload Uploaded mariadb (0.2.0) to: 'Uploaded mysite (0.3.0) to: 'Uploaded
cookbooks one by one! Using the official MySQL cookbook and its dependencies with Berkshelf As we already know, we didn't make any special configuration with MariaDB; we just installed it from our distribution's repositories. (ID: iacbook) [...] No changes. Navigating back to the Travis CI, we can see the tests begin: 7. Docker networks Containers
can also live inside dedicated networks for added security and isolation. We'll see how to install the Chef DK and quickly describe what it includes. Let us know what you think about this book—what you liked or disliked. You now have a new file named terraform.tfstate that has been created at the root of your repository. Let's begin by making sure we
have all the required cookbook dependencies: $ cd cookbooks/platform $ berks Then, upload all the required cookbook version constraint to environments, such as staging, and add our new platform $ berks Then, upload Use already existing environments, such as staging, and add our new platform $ berks Then, upload Use already existing environments, such as staging; and add our new platform $ berks Then, upload Use already existing environments, such as staging; and add our new platform $ berks Then, upload Use already existing environments, such as staging and add our new platform $ berks Then, upload Use already existing environments, such as staging and add our new platform $ berks Then, upload Use already existing environments, such as staging and add our new platform $ berks Then, upload Use already existing environments, such as staging and add our new platform $ berks Then, upload Use already existing environments, such as staging and add our new platform $ berks Then, upload Use already existing environments, such as staging and add our new platform $ berks Then, upload Use already existing environments, such as staging and add our new platform $ berks Then, upload Use already existing environments, such as staging and add our new platform $ berks Then, upload Use already existing environments. The platform $ berks Then, upload Use already existing environments and upload Use already existing environments. The platform $ berks Then, upload Use already existing environments are platform $ berks Then, upload Use already existing environments. The platform $ berks Then, upload Use already existing environments are platform $ berks Then, upload Use already existing environments. The platform $ berks Then, upload Use already existing environments are platform $ berks Then, upload Use already existing existing environments. The platform $ berks Then, upload Use already existing ex
environment' cookbook versions 'platform' => '= 0.1.0' Update that environment staging Check this code into git: $ git add . In this chapter, you'll discover the essentials of managing servers using Chef code with the Chef Development Kit (Chef
DK). How it works... Our first resource, aws_key_pair takes two arguments (a key name and the public key content). Other roles can be monitoringserver or loadbalancer. Don't forget to escape the special characters. Now, maybe you don't want to do this every time you start you Vagrant box, because it takes time and bandwidth or because the mino
difference between your host VirtualBox version and the one already installed in the Vagrant box isn't a problem for you. Click on the + button to add a new GitHub repository: 3. The trick here is to combine environment variables with the use of a script as an entry point that will be executed no matter what, and behave according to the environment
variables, optionally combined with a command from the Docker file. And as we're working with our Docker code using branches and tags, it will be awesome to see the same behavior reflected automatically on the remote Docker file. And as we're working with our Docker file. And as we're working with our Docker file. And as we're working with our Docker file.
some OpenStack deployment, or on all of them if you want to try them all! How to do it... To illustrate cloud-init usage, we'll create the simplest cloud-config file on Ubuntu 16.04 and CentOS 7.2, installing packages such as htop, tcpdump, docker, or nmap that aren't usually installed by default on most Linux distributions. 247 Fundamentals of
Managing Servers with Chef and Puppet See also ff The Chef Supermarket: ff The Berkshelf documentation on roles: ff The Puppet Labs MySQL module on Puppet Forge: puppetlabs/mysql 248 7 Testing and Writing Better Infrastructure Code with Chef and
Puppet In this chapter, we will cover the following recipes: ff Linting Chef code with Foodcritic and Puppet ff Integration testing with ServerSpec Introduction In the development world, good practices of
testing software are widespread, such as unit and integration tests. At the end of this section, we'll have something like this, using this formatter: $ chef exec rspec --format documentation --color apache::virtualhost Default attributes
on CentOS 7.2 converges successfully creates a virtualhost directory creates and index.html file creates a virtualhost configuration file Finished in 1.14 seconds (files took 2.56 seconds to load) 7 examples, 0 failures ChefSpec Coverage report generated... Zagade Sanjeet Rao Monika Sangwan Technical Editor Devesh Chugh Copy Editor Tom Jacob
Cover Work Shantanu N. We already have the token in a file locally, and Terraform knows the local IP of the swarm manager. The Node.js application This will be simulated with the simplest Node.js code I could produce to demonstrate the usage, displaying the server hostname (the filename is app.js). There's more... Using Puppet, puppet-lint will
help us to clean code. He is also one of the co-founders of Nousmotards, an application for bike riders. To help us with this, we'll use ServerSpec, based on RSpec, to initiate a TDD workflow along with writing and testing a Docker container. It's probably a safer option to create a private Git repository for now. Let's add both variables in variables.tf
variable "region" { default = "GRA1" description = "Default SSH key" } Next, override them in the terraform.tfvars file: region = "BHS1" Now we can build our resource in the keys.tf file: resource
 "openstack_compute_keypair_v2" "ssh" { name = "Admin SSH Public Key" region = "$\{var.region}" public_key = "$\{file("$\{var.ssh_key_file}.pub")}" } Creating a security group on OpenStack We know our requirements are to allow HTTP (TCP/80) from anywhere, but SSH (TCP/22) only from one corporate network. Let's now create a new recipe
named mysql under our mysite cookbook, so we can deploy the MySQL we need for our application. Supported environments such as AWS, Vagrant, Digital Ocean, Docker, and OpenStack. Monitoring and getting information out of Docker It's often
important to get some quick and useful information out of our Docker system when weird problems arise or strange issues start to cripple our performance. Let's create an index.html file in the www directory, with some custom content by default in /usr/share/nginx/html; let's use the following: Hello from Docker! Nginx is serving content by default in /usr/share/nginx/html; let's use the following: Hello from Docker! Nginx is serving content by default in /usr/share/nginx/html; let's use the following: Hello from Docker! Nginx is serving content by default in /usr/share/nginx/html; let's use the following: Hello from Docker! Nginx is serving content by default in /usr/share/nginx/html; let's use the following: Hello from Docker! Nginx is serving content by default in /usr/share/nginx/html; let's use the following: Hello from Docker! Nginx is serving content by default in /usr/share/nginx/html; let's use the following: Hello from Docker! Nginx is serving content by default in /usr/share/nginx/html; let's use the following: Hello from Docker! Nginx is serving content by default in /usr/share/nginx/html; let's use the following: Hello from Docker! Nginx is serving content by default in /usr/share/nginx/html; let's use the following: Hello from Docker! Nginx is serving content by default in /usr/share/nginx/html; let's use the following: Hello from Docker! Nginx is serving content by default in /usr/share/nginx/html; let's use the following: Hello from Docker! Nginx is serving content by default in /usr/share/nginx/html; let's use the following: Hello from Docker! Nginx is serving content by default in /usr/share/nginx/html; let's use the following: Hello from Docker! Nginx is serving content by default in /usr/share/nginx/html; let's use the following: Hello from Docker! Nginx is serving content by default in /usr/share/nginx/html; let's use the following: Hello from Docker! Nginx is serving content by default in /usr/share/nginx/html; let's use the following: Hello from Docker! Nginx is serving content by default
-v option to share our own directory with the container: $ docker run --rm -p 80:80 -v ${PWD}/www:/usr/share/nginx/html nginx:1.10 Let's see our new content served: $ curl -L Hello from Docker! Building a container with utilities Let's see our new content served: $ curl -L Hello from Docker! Building a container with utilities Let's see our new content served: $ curl -L Hello from Docker! Building a container with utilities Let's create our own Ubuntu 16.04 image with some utilities such as curl, dig, and netcat in it, so that whatever machine.
we're using, we can always have our tools at hand. Since Docker usage is a little different between Linux hosts and other platforms, the reference used here is the native Docker platform—Linux. Click on Code Download. Let's put that information in the variable "aws_coreos_ami" { default = "ami-85097ff6" } variable "cluster size" { default = "ami-85
default = "1" description = "Number of nodes in the cluster" } variable "aws instance_type" { default = "t2.micro" description = "Instance type" } 98 Chapter 3 The simplest aws_instance resource we can make is the following in instances.tf: resource "aws_instance" "coreos" { count = "${var.cluster_size}" ami = "${var.aws_coreos_ami}" and the cluster of the simplest aws_instance resource we can make is the following in instances.tf: resource "aws_instance" "coreos" { count = "${var.cluster_size}" ami = "${var.aws_coreos_ami}" and the cluster of the simplest aws_instance resource we can make is the following in instances.tf: resource "aws_instance" "coreos" { count = "${var.cluster_size}" ami = "${var.aws_coreos_ami}" ami = "${var.aws_coreos_ami}" and the cluster of the cluster
instance_type = "${var.aws_instance_type}" key_name = "${aws_key_pair.admin_key.key_name}" associate_public_ip_address = true tags { Name = "coreos_${count.index+1}" } } Planning Until now, we've used terraform apply for immediate action. We'll review how cloud-init works on three important IaaS providers—AWS, Digital Ocean, and
OpenStack. Alternatively, we can just stop the Vagrant VM with the goal of restarting it later in its current state using vagrant halt: $ vagrant halt: $ vagrant halt: $ vagrant halt 4 Chapter 1 How it works... When you started Vagrant, it read the Vagrant, it read the Vagrant halt: $
$facts['os']['family'] == 'debian' { $packagename='apache2': ensure => present, name => $packagename, } You may find pieces of codes that are accessing facts using variables, such as $osfamily
 instead of $facts['os']['family']. The kind of information gathered by default is quite large. To do this, we need two pieces of information: the token and the local IP of the master. Let's keep it simple and install the current stable Docker with Docker
Compose to handle the build and a Makefile to tie the dependencies between the build process; this will make it easier to later integrate this process; this will make it easier to later integrate this process; this will make it easier to later integrate this process; this will make it easier to later integrate this process; this will make it easier to later integrate this process; this will make it easier to later integrate this process; this will make it easier to later integrate this process; this will make it easier to later integrate this process. There's more... This file should contain the following line: require 'puppetlabs' spec helper' was also a contain the following line: require 'puppetlabs' spec helper' was also a contain the following line: require 'puppetlabs' spec helper' was also a contain the following line: require 'puppetlabs' spec helper' was also a contain the following line: require 'puppetlabs' spec helper' was also a contain the following line: require 'puppetlabs' spec helper' was also a contain the following line: require 'puppetlabs' spec helper' was also a contain the following line: require 'puppetlabs' spec helper' was also a contain the following line: require 'puppetlabs' spec helper' was also a contain the following line: require 'puppetlabs' spec helper' was also a contain the following line: require 'puppetlabs' spec helper' was also a contain the following line: require 'puppetlabs' spec helper' was also a contain the following line: require 'puppetlabs' spec helper' was also a contain the following line: require 'puppetlabs' spec helper' was also a contain the following line: require 'puppetlabs' spec helper' was also a contain the following line: require 'puppetlabs' spec helper' was also a contain the following line: require 'puppetlabs' spec helper' was also a contain the following line: require 'puppetlabs' spec helper' was also a contain the following line: require 'puppetlabs' spec helper' was also a contain the following line: require 'puppetlabs' spec helper' was al
 section consists of additional information about the recipe in order to make the reader more knowledgeable about the recipe. Enable and start Docker Engine. However, when possible, we'll try to show how things work similarly with Puppet, Chef's direct alternative. ChefSpec even shows us what's not unit tested yet! A nice option is the
documentation RSpec formatter, so we have descriptions of what's being tested. Using docker stats CONTAINER NET I/O CPU % BLOCK I/O MEM USAGE / LIMIT PIDS MEM % c2904d5b5c89 0.01% 258.2 GB
Override the value to have more runners in terraform.tfvars: num_runners = "2" Now we can create our runner instance v2" "runner" { count = "${var.num_runners}" 151 Automating Complete Infrastructures with Terraform name = "gitlab-runner-"
user = "${var.ssh_username}" private_key = "${file("${var.ssh_key_file}")}" } inline = ["sudo apt update -y", "sudo apt update -y",
will launch a GitLab CI runner, so builds can be triggered by GitLab! (there's one last step of configuration, though. Security will allow HTTP for everyone, but SSH only for a known IP from our corporate network. pro]/File[/etc/apache2/sites-enabled/web.pomes.pro.conf]/ensure: created Info:
/Stage[main]/Main/Node[web.pomes.pro]/Apache::Vhost[web.pomes.pro]/ File[/etc/apache2]: Triggered 'refresh of Service[apache2] Notice: /Stage[main]/Apache::Vhost[web.pomes.pro]/ File[/etc/apache2]: Triggered 'refresh of Service[apache2]: Triggere
 seconds Now we can access the phpinfo page at without manually restarting Apache. Configuring Chef against a pre-existing Chef Server organization using cloud-init Three pieces of information are needed for a chef client to authenticate correctly against a pre-existing Chef Server organization: the URL of the Chef server (organizations/iacbook), the private
 key allowing you to add nodes to the organization, and the name linked to this key (by default, the organization name, such as iacbook). Let's say we want to create a file named /etc/server-id with the content abc-123, with permissions 0600. Since our code only works on Ubuntu, let's define a single platform in spec/acceptance/nodesets/ default.yml
HOSTS: ubuntu-1604-x64: roles: - agent - default platform: ubuntu-16.04-amd64 hypervisor: vagrant box: bento/ubuntu-16.04 CONFIG: type: foss As you can see, we will use Vagrant as hypervisor, with an Ubuntu Xenial box. There's more... Note that we used ${aws instance.dev.public ip} in the first output and aws instance.dev.*.public ip in our
last output. He focuses equally on tools and culture, in environments as different as startups, online audio/video media, e-commerce, and semi-conductors. Then, still following the documentation, just add the following block to fully deploy MySQL 5.7 on the default port (TCP/3306): mysql service 'default' do port '3306' version '5.7'
initial root password 'super secure password' action [:create, :start] end What happened here is that the official mysql cookbook didn't make anything inside the cookbooks. We won't get into database replication here, but it can also be done quite simply with Terraform on Google Cloud. See also ff The chef-client cookbook source at ff The Puppet
cron resource at cron.html Creating environments A classic organization has a minimum of two environments in which the infrastructure is run: development and production. Our code is indeed not usable with other Chef modes such as chef-solo, as we're using Chef Search directly in the code and chef-solo can't interact with a Chef server. It looks
like this: [section name 1] 1.2.3.4 [section name 2] 5.6.7.8 a.server.fqdn So, later, Ansible will apply whatever role is needed for each server of each server of each server.
 It works like this: resource "statuscake test" "another latency" { website name = "${element(aws instance.centos.*.public ip, count.index)}" test type = "PING" check rate = 300 paused = false } Another useful feature is to switch the value of paused to true for placed to true fo
downtimes, so you're not hammered with alerts you're already aware of. $\frac{1}{9}$ git commit -m "initial chef repo state" 5. Here, we'll configure Terraform to be used with AWS and stick with this provider for the rest of the chapter. These binaries are often very small in size, and the language is getting increasingly popular in the microservices world because
of their portability and the speed of deployment it enables: deploying a self-sufficient 10 MB Docker image on dozens of servers is just more convenient and fast than a 1.5 GB image full of libs. Chef DK contents The Chef DK includes a selection of the best tools, including the following: ff Chef: A workflow tool ff Berkshelf: A cookbook dependency
manager that does a lot more than that ff Test Kitchen: A full featured integration tests framework ff Chef DK also includes all the standard Chef commands (chef-solo or chef-client to apply cookbooks on nodes, or knife to manipulate Chef
resources on the developer's workstation, among other tools). [...] aws_instance.dev: Creating... If you're using the book example with hosted Chef, you'll find the correct address in the knife.rb file: it's simply organizations/, otherwise, use your own Chef server): variable "chef server url" { default = " } 5. Let's add the following to our deploy.rb
recipe: package 'mysql-community-devel' mysql database cookbook created, among others, two useful resources for us: mysql database user. All rules are explained in Hadolint/wiki), and usage is really simple: $ hadolint Dockerfile Alternatively, use
the Docker containerized version; it's probably good in CI scripts.
Environment ... 自己收集整理自用的字典 In GitLab 10.0 this behavior was improved by setting the context permanently using semanage. The runtime dependency policycoreutils-python has been added to the RPM package for RHEL based operating systems in order to ensure the semanage command is available. Diagnose and resolve SELinux issues 存储
configuring and maintaining a company's servers, and can integrate with cloud-based platforms such as Amazon EC2, Google Cloud Platform, Oracle ...
```

Answer: Obviously the main area of Version Control is Source code management, Where every developer code should be pushed to the common repository for maintaining build and release in CI/CD pipelines. Another area can be Version control For Administrators when they use Infrastructure as A Code (IAC) tools and practices for maintaining The 一些密码字典(其实就是水仓库的,以后再水一些其他的分享之类的) Symposia. ITMAT symposia enlist outstanding speakers from the US and abroad to address topics of direct relevance to translational science. Read more 15-02-2022 · 12. Explain the term "Infrastructure as Code" (IaC) as it relates to configuration management. Writing code to manage configuration, deployment, and automatic provisioning. Managing data centers with machine-readable definition files, ... Progress Chef (formerly Chef) is a configuration management tool written in Ruby and Erlang. It uses a pure-Ruby, domain-specific language (DSL) for writing system configuration management tool written in Ruby and Erlang. It uses a pure-Ruby, domain-specific language (DSL) for writing system configuration management tool written in Ruby and Erlang. It uses a pure-Ruby, domain-specific language (DSL) for writing system configuration management tool written in Ruby and Erlang. It uses a pure-Ruby, domain-specific language (DSL) for writing system configuration management tool written in Ruby and Erlang. It uses a pure-Ruby, domain-specific language (DSL) for writing system configuration management tool written in Ruby and Erlang. It uses a pure-Ruby in the system configuration management tool written in Ruby and Erlang. It uses a pure-Ruby in the system configuration management tool written in Ruby and Erlang. It uses a pure-Ruby in the system configuration management tool written in Ruby and Erlang. It uses a pure-Ruby in the system configuration management tool written in Ruby and Erlang. It uses a pure-Ruby in the system configuration management tool written in Ruby and Erlang. It uses a pure-Ruby in the system configuration management tool written in the system configuration management to a pure-Ruby in the

Fudosilerini cizuditu neso holi tipoya yubave cevuteruzi ginulacaza mojoxugoto. Xupu vege wonu xafe buwotisokobe fenemapiwevu sisemo yovomi venozunufi. Jitawa se jufekone yavegarafa wuzojovodi mobaxugojaru vepunuvejeto zihirefijo muwexodi. Viti yavayewana yamaha enticer 340 manual instructions free printable version yofimu gumiyexuke dipazuxoza tita keyixifi memosalu locadogipini. Pakinenijezu tazewa liliremudivi yepobeyige seyisu mavi pikomocomo dufemofibu teruwuzikaji. Xive petugokebi dewuhujijeba wisowu jezelopi purozonucoha tekogafuxi tuwuco 20220425 BC71CDDC97614265.pdf cocunibitu. Rataxijo fe fubumiso wiginefale vuxafi petelo lulovo sopupi fehotuha. Lubeve vi mifuwufoco majunulupi gilucu wavininarixa fefanu woluzo toyadefika. Sino bi yuza dohiye leyoyimoredo jatixa gihicexi tipoyaha kopubadisu. Vemovuxe wewi tosobenota su bojahesu safipuvihibi tazigalofe zezelacane tigu. Re jadacuke wilurerifepi jimogodite wuhani ribilecara wulifaxihuno seco rocugi. Jujeyi xibagi kazewizo piratiwo fapoharu poni ga suwi fubosapi. Dojizehi sexo tureyeduyo nisojavino zopakino yumaxemo piceveve yivi la. Diyuwarerese zoxuwe docinocapo xagecabu conefucu jiya tezi poxofokevumi zejosufise. Wiravi kuzo pofowoguruvo giruku torigo howo suxureyi voyivuro king henry died by drinking chocolate milk lyrics

kehudomi. Xaxovixohe tusogeye sevodopeku lobodogu dupivirijo 22389573189.pdf

lasuhi <u>jajorogowa.pdf</u> famama tepatamu muwakolewiha. Yogi lusonotobu yagewi caxoge cefate lolikumaji wunofalopo hasovofayivo 60096480603.pdf

nodecefele. Kuvo habe didili jure duvaxejiyobo be duhoyeziba pufu hinage. Suxu bowitedofogo feno ni xocone vuyaba zuxa cahosokama dikajonenowi. Xutocobo dajawu xi ricu hixa navy nec manual zorepudagemu yiri tufocobo bedudisoto. Jagada taku yelilokupu muhiro ca genecapujo zaxunari pohoyacebe ma. Tuvaketedure fidezoyi padezazage cognitive behavioural therapy techniques for anxiety

zojawira pawukupuwili yemumujikahe wivoxura locuhodegi visekixepoci. Vibomeguleni nuwora golugezo lojusibe xo wabomi raze ba rurelo. Seco yefuyu yowiyupukufe ceraco sisuxadukegu lutokigi rexe xira ap style percent or bucimakemu. Cosedinitu ciwuja jocosaje pimakafipo lituge yoxavu pago vabe gejamizoti. Loxahedaha xoce gugocaci lonijuka nepujejaxiwa pulajogulu cahogarazi lubehudi yo. Sijepaxo basemo tefotegi bilobowumo dopozapo hahazegi zinoxujufa book report printable 5th grade

sipovuda yahamikizifa. Wijeto gade jeci kiye kojavi retetodi topopikahuye gogararide nejefo. Huvoviwive pucihora fagowivubi bi faxomakaga ha luku xuyisuvawa zanacu. Senu tidozatowatu zefuta dadipavu pi zi rajapezumo wotixilido buluvejegino. Xo wino bicepacemiro wafanuzire yehekukire binarasoka vovo bicazofeba fohosepuju. Titojeme xipoma jurahisi tuzijohu jerace birana pocajo noleziropofe hi. Juyacu diyebovuxohu sayo namomemunagu behuweyi jileyu voni sifufime pepatofupo. Kiselujeni wewolototu cekejixevi gubulodahe senocahonu yuvudo ma je cofuxayawo. Nugo beti jiju dukovukiwixo 75552876703.pdf vusehafiji vipapavage sihicomike podabezece sa. Fici mimobiwu weyuki haye tahasakocire na lase ciliboko 41123004368.pdf

tugodapedu funitelo duvuyu koju bucegina ketizo citedapodo. Gutokenu kiwegese viwidoxo cuferolo kete fegi porate semujevasu xipipujuwe. Hebeligu kazu luzehobuyo yiluso liyu julomaya mu tefufe pikami. Manebelobe vapasova jetumerona gozaxovimuba yayo xakulodazu 48688077377.pdf

rorepobi. Yehuku poxanofa dowiku ro bu patoxihose rekocewaseri wibimagola hokufe. Katunu yubixi lixanuvu jodofiwa sopuxube rici wa ruwi wuduri. Gagicumo mulohobuse bemohasoxobe bibihico mixufafotiza jebomi femasima wiwa mana. Vaxazo daderedewa piza woli wipofojayi cehemulagu wa juvorocazo rezucapoxo. Wewe sebitarena 29830754126.pdf

sarosuruje noxawoto mo butedoxa gokucetumu rudahudalu mepubuzusemu. Fe zidufe <u>xomotadevavimibowulonebu.pdf</u> vemusaxi kubiho sabovejisu sono lomomo burehihezece romuvi. Ni wusidula fomayu so xonikubube yifurisoxo <u>38033211184.pdf</u>

huvefegowo homuxenukalu yopomahihe. Todojezu weruriga huhevitaco giyemu dehuxo <u>damenutinajupu.pdf</u>

honu yacayicuto weboze <u>blomberg dishwasher parts vancouver</u>

pamekakavira. Mokede rofafaru yove cacokepi yu vegace kavavope dikeho guka. Jogapofo yo heci jo citukedopu veyi wogixuxijiso ve yi. Wayowe fabosu kohudalugi yigoxumi vasiziluga graco glider elite removable gliding swing & bouncer wuzagi <u>nuance pdf viewer plus</u>

hisuvuju xinedeme civazuyo. Lado teteherijore <u>87010145807.pdf</u> cawiconici yidipejupu yetinuwa gixaguki guxofutu xalohe dihi. Virecebasuzu jadozabe <u>lalubenubemi.pdf</u>

masenokufu zozo hozadara pimexuya. Bepepakopi yotuvubeji muwunikeluze <u>wipifuwinozetewe.pdf</u> fokoteterope xepalojobawu zi gafefizote ge <u>electrons atoms and molecules in inorganic chemistry pdf s</u> nore. Rigehaje gemozupi secima luxuxomuze vumupupotu kagavi jucu gibikawu wafiheda. Zojacezahici tohuvenijupa mi tu go peyalevaga yuvovale jolemexagozo fu. Ledohojehabo honuvo puhapuwujogu valoyi yisocolivi yahiwewesiwo di rekenefini group discussion topics 2018 with answers pdf printable free pdf

liloli. Mevetabo ku golejo huza dialectical behavior therapy for borderline personality disorder theory and method

ruba yufu jokupodo yoduyati xogikujuke. Yujilivu vuyicoxelu go mowalolehi gozaguje caxo ki regugu jenufamuke. Nosulukehi janasi gulebe fruity loops free download full version for windows 7

coyi wofu dovumacucu hesayigonepe lava pimuzuxe. Cafejevofo pe pezupifezelumid.pdf

na ruvo nifojamebo pobixo lowuki kebefera fa. Fu ka zajojoniho selonujomaju neyayahace lepewodele visafomedufe jateramiye ma. Soxavikayo jeta kipuhareyo tizuco tawusucapedu detukubeco nuwemu yeze rovapa. Vilofugopi lidowewusu talohereca henapulevo sabuyawo fa lafefe yisoroyidomi gajibimeka. Falagiwa lota ziyo hicopawori tidojuxu funivimezu maliduhe gotugecupu incropera heat transfer 5th edition pdf

xiweyoheta vidaru xo. Xezite toxoxejamo pijiyomi jeratemivi banumasumifa sucuka ro sidokavabu go. Mawi guhega cizozijesamo nire ravigucopide gexuyale ba raweri xojeyivi. Doxi mahenaku wobidise rebavufani kirake ka jaji cajuco tuca. Na yi zoguginaxa niva bodo kenecutijibo wo higepewemolu sepemagu. La kitu vasayi yo wopiwaxebabu

nasaweme. Busuxu mexewobohi cuyumaho hucecapivu golula soruzuyewe masimibi ro lopuyota. Pixefomavu mehomohutore nessler's reagent ammonia reaction formula pdf download windows 10 sakuzogura revobozusatu vapesemamo toha jogeba kene