Common OpenStack Deployments
This page intentionally left blank
Common OpenStack Deployments

Real-World Examples for Systems Administrators and Engineers

Elizabeth K. Joseph
with Matthew Fischer
This book is dedicated to the OpenStack community. Of the community, I’d also like to specifically call out the help and support received from the Puppet OpenStack Team, whose work directly laid the foundation for the deployment scenarios in this book.
This page intentionally left blank
Contents

Preface xv
Acknowledgments xxiii
About the Author xxv

1 What Is OpenStack? 1
The Cloud 1
OpenStack Joins the Cloud 2
Building Your Cloud 3
Uses 3

Key Components 4
Instances 4
Queuing 4
Dashboard (Horizon) 4
Compute (Nova) 7
Identity (Keystone) 9
Networking (Neutron) 9
Image Service (Glance) 10
Block Storage (Cinder) 10
Object Storage (Swift) 11
Telemetry (Ceilometer) 11
Bare Metal (Ironic) 12
Orchestration (Heat) 12
Containers (Magnum) 13
Other Projects 13
Release Cycle 13
Ubuntu Long Term Support 14
Ubuntu Cloud Archive 15
Puppet Modules 16
Summary 17

I First Deployments  19

2 DevStack 21
What Is DevStack? 21
Developer Usage 22
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Usage</td>
<td>22</td>
</tr>
<tr>
<td>Continuous Integration Usage</td>
<td>22</td>
</tr>
<tr>
<td>DevStack Requirements</td>
<td>22</td>
</tr>
<tr>
<td>Deploying DevStack</td>
<td>23</td>
</tr>
<tr>
<td>Dashboard: Log In as a User</td>
<td>24</td>
</tr>
<tr>
<td>Dashboard: Log In as an Administrator</td>
<td>28</td>
</tr>
<tr>
<td>Working with the Host on the Command Line</td>
<td>30</td>
</tr>
<tr>
<td>DevStack Options</td>
<td>31</td>
</tr>
<tr>
<td>DevStack &quot;stable&quot;</td>
<td>31</td>
</tr>
<tr>
<td>Customizing DevStack</td>
<td>32</td>
</tr>
<tr>
<td>Summary</td>
<td>33</td>
</tr>
<tr>
<td><strong>3</strong> Networking</td>
<td></td>
</tr>
<tr>
<td>Key Concepts</td>
<td>35</td>
</tr>
<tr>
<td>Planes of Operation</td>
<td>35</td>
</tr>
<tr>
<td>Provider Network</td>
<td>36</td>
</tr>
<tr>
<td>Tenant Networking</td>
<td>36</td>
</tr>
<tr>
<td>Maximum Transmission Unit (MTU)</td>
<td>38</td>
</tr>
<tr>
<td>Deployment Requirements</td>
<td>39</td>
</tr>
<tr>
<td>Traffic Flow</td>
<td>43</td>
</tr>
<tr>
<td>Controller Node</td>
<td>44</td>
</tr>
<tr>
<td>Compute Node</td>
<td>48</td>
</tr>
<tr>
<td>Other Resources</td>
<td>49</td>
</tr>
<tr>
<td>Summary</td>
<td>49</td>
</tr>
<tr>
<td><strong>4</strong> Your First OpenStack</td>
<td></td>
</tr>
<tr>
<td>Requirements</td>
<td>51</td>
</tr>
<tr>
<td>Reference Deployment</td>
<td>52</td>
</tr>
<tr>
<td>Initial Setup</td>
<td>52</td>
</tr>
<tr>
<td>Networking</td>
<td>52</td>
</tr>
<tr>
<td>Operating System</td>
<td>54</td>
</tr>
<tr>
<td>System Configuration</td>
<td>54</td>
</tr>
<tr>
<td>OpenStack Components</td>
<td>58</td>
</tr>
<tr>
<td>Controller Node</td>
<td>58</td>
</tr>
<tr>
<td>Compute Node</td>
<td>83</td>
</tr>
<tr>
<td>Manage an Instance</td>
<td>85</td>
</tr>
<tr>
<td>Summary</td>
<td>87</td>
</tr>
</tbody>
</table>
8 Block Storage Cloud  151

Uses  151
   Cloud Provider  151
   Data Processing  152
   Keeping Backups  153
Requirements  153
   Select Components  153
Architecture Overview  154
Scenario  155
   Controller Node Setup  155
   Creating and Attaching a Volume: Dashboard  157
   Creating and Attaching a Volume: OpenStack Client  161
   Using the Volume  162
   Automation  166
Summary  166

9 Object Storage Cloud  167

Uses  167
   Web Hosting Company  167
   File Sync and Sharing  168
   Log Storage  168
Requirements  169
   Select Components  169
   Key Concepts  169
Scenario  173
   Controller Node Setup  173
   Creating a Container and Object: Dashboard  176
<table>
<thead>
<tr>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating a Container and Object:</td>
</tr>
<tr>
<td>OpenStack Client</td>
</tr>
<tr>
<td>Using an Object</td>
</tr>
<tr>
<td>Beyond Swift</td>
</tr>
<tr>
<td>Summary</td>
</tr>
<tr>
<td><strong>10 Bare Metal Provisioning</strong></td>
</tr>
<tr>
<td>Uses</td>
</tr>
<tr>
<td>Cloud Hosting Company</td>
</tr>
<tr>
<td>Internal Cloud</td>
</tr>
<tr>
<td>Database Hosting</td>
</tr>
<tr>
<td>High-Performance Computing</td>
</tr>
<tr>
<td>Architecture Overview</td>
</tr>
<tr>
<td>Installation</td>
</tr>
<tr>
<td>Using Ironic</td>
</tr>
<tr>
<td>Managing Ironic</td>
</tr>
<tr>
<td>Community</td>
</tr>
<tr>
<td>Summary</td>
</tr>
<tr>
<td><strong>11 Controlling Containers</strong></td>
</tr>
<tr>
<td>What Is a Container?</td>
</tr>
<tr>
<td>Uses</td>
</tr>
<tr>
<td>Public Cloud Company</td>
</tr>
<tr>
<td>Online Gaming Company</td>
</tr>
<tr>
<td>Container Drivers for Nova</td>
</tr>
<tr>
<td>Magnum</td>
</tr>
<tr>
<td>Magnum Concepts</td>
</tr>
<tr>
<td>Installing Magnum</td>
</tr>
<tr>
<td>Summary</td>
</tr>
<tr>
<td><strong>III Scaling and Troubleshooting</strong></td>
</tr>
<tr>
<td><strong>12 A Whole Cloud</strong></td>
</tr>
<tr>
<td>Uses</td>
</tr>
<tr>
<td>Requirements</td>
</tr>
<tr>
<td>Select Components</td>
</tr>
<tr>
<td>Scenario</td>
</tr>
<tr>
<td>Controller Node Setup</td>
</tr>
<tr>
<td>Compute Node Setup</td>
</tr>
</tbody>
</table>
A  Reference Deployment  231
   Requirements  231
   Installation  231
   Recommendations  233

B  Other Deployment Mechanisms  235
   Chef  235
   Ansible  235
   SaltStack and Others  236
   Vendor-Specific  236

C  Long-Lived Puppet  237
   Puppet Master or Masterless?  237
   Hiera  237
      Passwords in Hiera  238
   Node Classification  238
   Module Management  238
   Software Life Cycle  239
   Roles and Profiles  239
   Packages  240
   Revision Control  240
   What Else Belongs in Your Composition Module?  240
   More Information  241

D  Contributing Code to OpenStack  243
   Contribution Overview  243
      Release Cycle  243
      Communication  244
      Specifications  245
      Bug and Feature Tracking  246
      Git and Code Review  246
      Testing Infrastructure  247
      Other Contributions  248

E  OpenStack Client (OSC)  249
   Basics  249
      Authentication  249
      Commands  250
      Interactive Mode  250
   Quick Reference  251
Finding Help with OpenStack 253

Documentation 253
Mailing Lists 254
Web-Based 254
Chat 254
Conferences and User Groups 255
OpenStack Summits 255
OpenStack Ops Meetup 255
OpenStack User Groups 255
Vendors 256

Index 257
And suddenly you know: It’s time to start something new and trust the magic of beginnings.

Meister Eckhart

Companies today are heavily relying upon virtualized and cloud-based solutions in their infrastructures. Whether they are off-loading their work to third-party cloud providers, using a virtualized solution in-house or building clouds in their own data centers, OpenStack has a lot to offer. This book provides an introduction to using and deploying OpenStack, open source software for creating private and public clouds, in your own organization.

OpenStack as an open source project has only existed since 2010 but quickly gained support of companies around the world and the broader open source community. At open source conferences OpenStack talks quickly sprang up by the time the project was just three years old to the extent that some in the community joked that the “OS” in open source conference names no longer stood for “Open Source” but for “OpenStack.” Demand for talent in the field has risen along with interest, with OpenStack experts demanding a premium as companies expand their private cloud deployments.

**Audience**

The audience for this book is Linux and Unix systems administrators and network engineers seeking to learn the basics of OpenStack and to run sample deployment scenarios that can be transitioned into real-world deployments. It also provides insight into the most popular ways OpenStack is being used and how your organization can get there.

Though detailed commands are given, literacy with Linux systems administration is expected so you can focus on learning about OpenStack and simplify the task of troubleshooting. If you are doing these deployments in a series of virtual machines rather than bare metal servers, familiarity with a virtual machine technology is expected. This book does provide a reference deployment using virtualization with KVM and QEMU on Ubuntu if you wish to have step-by-step instructions. However, the intent is to leave virtualization preference up to the reader and make it easier to transition to a physical setup.

Basic networking experience is recommended since networking is such an important part of OpenStack, but diagrams in Chapter 3, “Networking,” will help guide you through the OpenStack network architecture we’re demonstrating.
Goals and Vision

I was inspired to write this book after attending several open source conferences where OpenStack was becoming an increasingly popular topic. In spite of all these talks, I’d still get practical usage questions from friends and colleagues about how they could use OpenStack in their organizations. To that end, each chapter with a deployment scenario begins with a series of real-world examples of how it is being used by organizations in production. From serving up fleets of web servers to log storage, backups and data processing, these usage examples help you to find a place for OpenStack to accomplish a variety of tasks that span organizations across various industries, universities and governments.

Whether you’re seeking to do an OpenStack deployment yourself or work with a vendor, this book also provides a guide through these sample deployments. You learn the basics of how to configure various OpenStack components and then walk through interactions with them via a web dashboard and the OpenStack command-line client. The mechanics of how the components interact with each other are also explained, so you have an understanding of how you’re interacting with the systems.

System Prerequisites

In order to run most of the deployment scenarios in this book you need, at minimum, two computers with combined resources of 6G of RAM and 50G of hard drive space. A laptop with 8G of RAM is sufficient if you’re using virtualization, including use of our tested reference deployment described in Appendix A, “Reference Deployment.” If you choose to use real hardware, you will need two computers and two switches. One of the switches must be connected to a network with access to the Internet so you can install system packages and pull in configuration management tooling on your systems.

Diagrams of both virtualized and physical environment options, including a breakdown of specifications for each, are included in Chapter 3.

Ubuntu

You need to download the latest Ubuntu 14.04 ISO image to complete the deployment scenarios. Ubuntu is a Linux distribution based on Debian, which had its first release in 2004. Initially aimed at making Linux easier for regular people, growth of Ubuntu on servers has exploded over the past five years. It’s now the number one choice for cloud deployments, both in OpenStack and on other cloud platforms such as Amazon Elastic Compute Cloud (EC2).

OpenStack’s beginnings are also intertwined with the Ubuntu community. A number of the early project contributors come from the Ubuntu project. The decision to use Ubuntu comes from the expertise of the authors and a desire to focus on understanding the types of deployments and basics surrounding OpenStack itself rather than the underlying operating system.
Although the OpenStack ecosystem is much broader than Ubuntu, with professional services built around Red Hat Enterprise Linux (RHEL) and even a move to other operating systems beyond Linux, this book uses Ubuntu 14.04 LTS (Long Term Support) as the base installation for OpenStack. This will impact configuration to some extent, since the Ubuntu Cloud Archive (see Chapter 1, “What Is OpenStack?”) referenced in this book will be different from that of RHEL, CentOS openSUSE, Fedora and others. However, the core OpenStack knowledge and the deployment examples will be exportable to other systems once you start making plans to move into production with your system of choice.

**Puppet**

With an initial release in 2005, today Puppet is one of the most popular configuration management systems in the world. The Puppet OpenStack modules were one of the first configuration management system projects to reach maturity in the OpenStack community. Puppet modules for each release are made available within weeks of the OpenStack release itself.

Like the selection of Ubuntu, the selection of Puppet for configuration management was made so we can focus less on fundamental deployment and management, and more on learning about OpenStack. While you will be using Puppet commands for these deployments, the basic concepts are explained and prior knowledge of Puppet is not required. Additionally, the creators of the Puppet modules for OpenStack are a diverse group of developers and operators from around the world and are formally supported by multiple organizations. They are known to be flexible enough for a variety of environments.

We will be using the default installed Puppet version on Ubuntu 14.04. If you’re seeking to run Puppet in production, the OpenStack Puppet team recommends using the Puppet version directly from Puppet instead. This is covered in Appendix C, “Long-Lived Puppet.” The OpenStack Puppet modules for OpenStack are currently tested on both Ubuntu and CentOS.

Appendix B, “Other Deployment Mechanisms,” has been provided to give you an overview of other configuration management and orchestration services you may be interested in using, should your organization prefer Chef, Ansible or something else.

**Tour**

The following is a short tour of what to expect from each chapter and the appendices.

- **Chapter 1: What Is OpenStack?** This first chapter provides a brief introduction to cloud computing before moving into an introduction to OpenStack itself. It goes on to provide descriptions of each component of OpenStack that are explored in depth in later chapters. The chapter concludes by talking about the OpenStack release cycle and how Ubuntu and Puppet factor into this cycle and their usage in this book.
- **Chapter 2: DevStack**  Built as a non-production development tool, DevStack is also a great introductory tool for a single-server deployment of OpenStack and for getting familiar with it quickly. You learn how to use it, launch your first compute instance and execute basic debugging techniques.

- **Chapter 3: Networking**  Networking is an important and complicated component of OpenStack and will drive decisions you make as you build your own deployments. This chapter is devoted to explaining key concepts for networking in OpenStack and to dive into the networking decisions and requirements used in our deployment scenarios. Diagrams and written descriptions help guide you through these concepts.

- **Chapter 4: Your First OpenStack**  Before getting into chapters using configuration management, this chapter walks you through a manual install of the basic components of OpenStack, Nova compute, Keystone Identity, Glance image storage and Neutron networking. This will give you a firm understanding of how the pieces fit together, from the databases to the service users in Keystone, which are handled in later chapters automatically by configuration management to the queuing system.

- **Chapter 5: Foundations for Deployments**  This chapter serves as a basis for all your subsequent deployment scenarios using Puppet. It explains the core components and sets up your basic controller and compute node and concludes with some basic usage tests to confirm it is working.

- **Chapter 6: Private Compute Cloud**  The first of our Puppet-driven deployment scenarios, this chapter provides usage examples and then walks you through the basics of interacting with a private compute cloud. You learn how to add a compute flavor and your first operating system image, how to launch and interact with a Nova compute instance from both the Horizon dashboard and the command line client and then complete a basic web service demonstration.

- **Chapter 7: Public Compute Cloud**  Your next deployment scenario adds metering to your cloud with Ceilometer. Ceilometer tracks usage of RAM, CPU, networking and more for your deployments, which you can then feed into systems to do monitoring and billing. Usage examples are given, as well as a basic introduction to Ceilometer itself and a walkthrough of how to use it with a strong focus on the command-line client.

- **Chapter 8: Block Storage Cloud**  Moving on from compute-focused deployments, this chapter introduces the concept of block storage and provides example usage. The basics of OpenStack Cinder block storage architecture are explained, and then you are walked through configuration. You then attach a Cinder block storage device to a compute instance, partition it, give it a filesystem and mount it inside your compute instance so you can add files to it.

- **Chapter 9: Object Storage Cloud**  Continuing with storage, this chapter introduces you to the concept of object storage using Swift. You learn about basic Swift concepts and deployment considerations, and then build your own tiny Swift
deployment. Using this deployment scenario, you create storage containers, upload files and build upon your earlier web service demonstration by including an image served by object storage on a compute instance.

- **Chapter 10: Bare Metal Provisioning**  Moving on from our deployment scenarios, usage examples and an architecture overview of bare metal provisioning with OpenStack Ironic are provided. Though you aren’t doing an actual deployment for this chapter since we couldn’t make assumptions about your hardware, guidance is given for how you might.

- **Chapter 11: Controlling Containers**  In this, another non-deployment chapter, you learn why you may wish to use containers in an OpenStack deployment. The chapter continues with a basic introduction to OpenStack Magnum and considerations for your own deployments.

- **Chapter 12: A Whole Cloud**  Coming back to our deployment scenarios, this chapter provides one final scenario where all the components from Chapters 6–9 are brought together in a single scenario. This demonstrates how they can be used together, and you’re encouraged to do your own tests with this feature-rich cloud scenario.

- **Chapter 13: Troubleshooting**  OpenStack is a complicated infrastructure project, and every engineer running it needs to get very good at troubleshooting. You are walked through understanding error messages and log files, tooling for troubleshooting network problems, common mistakes in configuration files and basic Puppet debugging. The chapter concludes with tips for how you can mitigate breakage in your deployment and tips for asking for help.

- **Chapter 14: Vendors and Hybrid Clouds**  The final chapter of this book introduces you to the broader OpenStack ecosystem through vendors and hybrid clouds, which blend a local OpenStack deployment with hosted solutions. Evaluation considerations for choices you make from cost to data sovereignty and security are covered.

- **Appendix A: Reference Deployment**  In case you run into trouble with your own environment selections, or simply don’t have a preference, this appendix provides a tested, virtualized reference deployment you may use.

- **Appendix B: Other Deployment Mechanisms**  We use Puppet throughout this book, but this appendix introduces you to other ways you can deploy OpenStack, from Chef and Ansible to where to find vendor-specific tooling.

- **Appendix C: Long-Lived Puppet**  The Puppet examples in this book are triggered manually. This appendix gives direction for your options when building a proper, maintainable Puppet system.

- **Appendix D: Contributing Code to OpenStack**  Feel inspired to contribute back to OpenStack? Or need a feature or bug fix? This appendix gives an introduction to how you go about contributing code to the OpenStack open source project, including how community members communicate and how to use the development tooling.
Appendix E: OpenStack Client (OSC)  The OpenStack Client is rapidly replacing individually maintained clients for each project. This appendix provides some background and a quick reference of some common commands.

Appendix F: Finding Help with OpenStack  The final appendix provides a quick tour of the support options in the OpenStack community, both online and in person. It concludes with tips for finding paid support as well.

Conventions
Instead of using the root user, sudo is used throughout this book. As such, all commands are prefixed with a dollar sign to indicate that it’s a command you should be typing into a shell. For instance, when you’re preparing your Ubuntu systems and want to update the Ubuntu sources before installing anything, we show that as:

```
$ sudo apt-get update
```

When lines are wrapping, we use the bash syntax of \ to indicate that the command wraps to the next line. The creation of a compute instance is a good example of this:

```
$ openstack server create --flavor m1.tiny --image "CirrOS 0.3.4" \ 
  --security-group default --nic net-id=Network1 \ 
  --availability-zone nova my_first_instance
```

For most of the OpenStack commands, we have provided sample output of what to expect when you run each command. For the output of standard tooling for things like Ubuntu package installation, git clones and MySQL commands, this output is generally not included.

Supplementary Materials
As discussed, you will need a copy of the Ubuntu 14.04 ISO to install Ubuntu on your initial OpenStack controller and compute nodes. Later, the Ubuntu 14.04 server QCOW2 cloud image will need to be loaded into Glance for our deployment example using Ubuntu as a compute instance. All Puppet modules and other packages are downloaded through scripts you’re instructed to use or through Puppet itself.

This book also has an accompanying git project hosted on GitHub at https://github.com/DeploymentsBook.

This project is broken into several repositories:

- **http-files**—Used for our basic web server examples.
- **puppet-data**—The repository you clone to bootstrap your installation of Puppet on your OpenStack nodes. It also includes your core configuration file, hiera/common.yaml, which you will be editing.
- **puppet-deployments**—Pulled in automatically by setup.py in puppet-data, this is the composition module used for all of our deployment scenarios. It includes service profiles and the foundation roles used in each chapter. It also includes a README.md file for the latest known issues and work-arounds that will be updated throughout the life of this book.

- **scripts-and-configs**—Miscellaneous scripts, commands and configuration file examples provided so you have a place from which to view or copy them as needed. The commands provided in this directory for the deployment chapters are particularly valuable for viewing OpenStack client output that doesn’t fit well on a printed page.

Finally, a blog and the latest updates to other materials being made available throughout the lifespan of this book can be found on our web site at http://deploymentsbook.com/. You can also follow us on Twitter for updates @DeploymentsBook.

Register your copy of *Common OpenStack Deployments* at informit.com for convenient access to downloads, updates, and corrections as they become available. To start the registration process, go to informit.com/register and log in or create an account. Enter the product ISBN (9780134086231) and click Submit. Once the process is complete, you will find any available bonus content under “Registered Products.”
This page intentionally left blank
Acknowledgments

When I began working on this book, I knew I had my work cut out for me and that I would need help from various members of the community. OpenStack is a huge infrastructure project. Every aspect of the project is continually being refined and revised, and even the official project documentation struggles to keep up. New projects are always being added, and the existing ones are reaching various states of maturity.

A few months into writing I brought in my contributing author, Matt Fischer. He put in a massive amount of work across three releases of OpenStack to get our Puppet composition module working. This book wouldn’t have made it past the theoretical stage without his efforts. Colleen Murphy of the OpenStack Puppet team also spent time working with us on changes and by doing review of chapters and appendices. Clayton O’Neill, Eric Peterson and Adam Vinsh, working with Matt directly, assisted with Puppet configuration questions. We also had the project team lead of the OpenStack Puppet project, Emilien Macchi, pitch in with advice and getting required changes needed upstream.

Thanks to members of various teams who took time as subject matter experts to review individual chapters, including: Mike Perez, Gordon Chung, Donagh McCabe, Matthew Oliver, Hisashi Osanai, Christian Schwede, Kota Tsuyuzaki, Julia Kreger and Charlie Crawford. Pasi Lallinaho helped by turning our basic HTML page examples into something considerably more palatable. We also had help from my friends and fellow systems and network engineers Jonathan DeMasi, Ola Peters, Joe Gordon, Eric Windisch, James Downs and Brent Saner on several of the chapters.

We had several multi-chapter reviewers with varying backgrounds to thank, including José Antonio Rey, Mohammed Arafa, Doug Hellman and Christian Berendt.

Throughout this process, my husband Mike Joseph has been incredibly supportive of my work. Even during the most difficult times when I wasn’t sure I could finish it, he was ready with encouragement.

Finally, I’d like to thank my editors at Pearson. Thanks to my editor and primary contact at Pearson, Debra Williams Cauley, for offering advice on how to approach each section and for keeping me on track throughout this process. Also to editor Chris Zahn, who made his way through editing the entire book.
This page intentionally left blank
About the Author

Elizabeth K. Joseph is a Systems Administrator working on the OpenStack Infrastructure project. In her role on this team, she supports OpenStack developers as they make contributions to the project and is active on OpenStack development mailing lists, and has worked on test engineering for the OpenStack on OpenStack (TripleO) project. She has given tutorials on the basics of OpenStack for San Francisco Bay Area organizations and regularly attends the bi-annual OpenStack Design Summits. A regular speaker on Open Source topics at conferences worldwide, her work beyond OpenStack includes contributions to the Ubuntu project and serving on the board of a non-profit that puts Linux-based computers in public schools.

Matthew Fischer has worked as a software developer for over 15 years in roles ranging from UNIX kernel to mobile phone development to DevOps. Matt currently works on a team deploying and running OpenStack and has been using Puppet to deploy OpenStack since 2013. When not solving automation problems, Matt enjoys hiking, camping, skiing, craft beer, and spending time with his family in Fort Collins, Colorado.
This page intentionally left blank
In the following chapters, you are given examples of several types of OpenStack deployments to consider. Each deployment is based on a series of real-world scenarios that will be described in detail and followed up with instructions for how the deployment can be constructed in a small scale.
This page intentionally left blank
Block Storage Cloud

The road to the City of Emeralds is paved with yellow brick.
L. Frank Baum, *The Wonderful Wizard of Oz*

OpenStack provides two popular mechanisms for storage: object and block storage. Block storage is traditionally what you’d mount as a filesystem on your server. Object storage instead hosts individual files that are then referenced from within your application. In Chapter 9, “Object Storage Cloud,” you learn about why you may want to use object storage with Swift to host files. This chapter covers block storage with Cinder.

Integrated with the rest of OpenStack, Cinder volumes can be created within an OpenStack cloud and then live mounted to a specified instance at the whim of a user. Additionally, you can unmount that volume from one instance and mount it on another with just a few commands.

**Uses**

One of the strengths of OpenStack is to avoid vendor lock-in, particularly when used in combination with versatile solutions like Cinder. Cinder provides an abstraction layer through the volume manager that hooks in to over 70 different proprietary and open source storage solutions. Additionally, it can be an interface to multiple back ends at once, enabling you to not only diversify your back ends across vendors but also change them out and do a planned migration as your organization sees fit.

**Cloud Provider**

Whether you’re running a public cloud accessible to customers or a private cloud for use within your organization, offering the capability to extend the given filesystem requirements in place for the flavors can be a huge benefit.

With the capability to add block storage volumes on the fly, the storage requirements for default compute nodes can be kept small to preserve space for users who want to focus on compute power and give others the flexibility to add the storage they need on the fly. The flexibility to extend storage as needed helps scaling out resources without over-committing beforehand and makes migrations easier if data is kept on a single volume that can be moved to a new compute instance.
Another major benefit is that the compute nodes can be run on throw-away commodity hardware. The data needing to be stored can be either kept on expensive redundant enterprise hardware or on something that has built-in redundancy like Swift. As a result, compute nodes themselves could become independent, throw-away components of your infrastructure, spun up as needed and replaced with identical servers that attach to your storage back ends. Finally, upgrades are also simplified for your customers. If you have a new copy of your system you are testing, you can snapshot the production data from a Cinder volume and then attach it to your test system to see if it works.

---

**Pets versus Cattle**

If you’re unfamiliar with the pets versus cattle metaphor in cloud computing, it’s time to get you up to speed.

Prior to the recent rise in cloud computing, systems administrators would work with our managers to spec out hardware for servers and work out a budget to purchase it. A new server would be delivered to the facility where we’d install the operating system and put it into a rack in our data center. A nice label would go on the server designating a name, and over time we’d work to maintain and update this server. The server would be upgraded for years through operating systems and hardware failures and upgrades. We’d start noticing specific quirks about the hardware: one server may have a flaky onboard NIC so our notes explained that we added an additional card, another machine may take a while to boot up. We became familiar with these servers we grew and nurtured over time, like pets.

When workloads and businesses began moving to the cloud for their workloads, everything changed. Individual servers no longer had quirks and were easy to move around. A complete replacement of a server became more common than an in-place upgrade, and tools were written to manage fleets of servers operating in a larger infrastructure rather than working with individual servers. Instead of pets, servers became a lot more like cattle.

As a systems engineer and an animal lover, I retain a love for both pets and cattle, metaphorically or not. But this metaphor does effectively demonstrate the differences in how servers are treated today in a cloud environment.

---

**Data Processing**

Whether you’re a film production company or a research institution, using compute nodes to do data processing and analysis is a common use for cloud-based infrastructures. But where is this data stored and how is it shared in your organization? What do you do when the compute nodes you’re working on run out of space? With Cinder block storage, users are able to extend volumes after creation or simply create new volumes as they are needed and attach them to their running instances and mount them in a matter of minutes. Running out of hard drive space on an instance because you have too much data is no longer a problem.

Additionally, if you realize you need more processors or memory, it’s simple to create a new compute instance and move the volume over to the new instance. All your data moves with you to your new server.
Keeping Backups

Making sure your data is backed up and replicated is a common concern, and Cinder offers several options for this. As mentioned earlier, it provides an abstraction layer for dozens of back ends, so you have a lot of options in your environment and on top of that, the different types of backups that Cinder offers.

Many virtualization technologies supported by OpenStack have built-in snapshot capabilities, but OpenStack’s block storage also has one. This enables volumes to be backed up as snapshots to other block storage volumes and can be used on its own, using the existing configuration. Whether offered as a backup service by a cloud provider or as an automatic service for users on an OpenStack cloud, this is a valuable service.

You can also clone volumes. Many back ends are smart enough to do a copy-on-write in where zero copying actually happens. The new volume references an existing volume it was cloned from and writes data on top of that. This makes it very fast. Some snapshot features also just do this in the back end, but every vendor solution is implemented differently.

Moving beyond snapshots, Cinder also offers a backup service that enables you to back up your block data to an object store. This helps with the scenario of your entire block storage back end going completely off-line. Differential and incremental backups can be performed and, unlike a snapshot, it is only backing up data that was actually used, not the entire volume and unused bits.

We won’t spend a lot of time on backup scenarios in this chapter. Instead we’ll be focusing on adding volumes in a more instance-focused environment where you’re adding volumes to instances, but they are great options to keep in mind.

Requirements

In this deployment scenario you’ll once again be using the controller and compute nodes you created in Chapter 5, “Foundations for Deployments.” We will be creating a 10GB volume group for use by Cinder on the controller and the minimum specifications defined in that chapter will easily support this.

Select Components

In addition to the foundational tooling, we’ll be adding Cinder block storage to your deployment in this scenario. This will extend our very basic installation to have the following services. Their placement is demonstrated in Figure 8.1.

- Compute (Nova)
- Identity (Keystone)
- Networking (Neutron)
- Image service (Glance)
- Dashboard (Horizon)
- Block Storage (Cinder)
Chapter 8  Block Storage Cloud

Architecture Overview

In Chapter 1, “What is OpenStack?” you were briefly introduced to the components that make up the block storage (Cinder) service: cinder-api, cinder-scheduler, cinder-volume and also cinder-backup. A user will likely only be exposed to the API. As operators though, understanding the architecture for the service is important as we seek to make decisions about how we build the system and debug problems.

When a user request comes in, either from the OpenStack dashboard (Horizon), the OpenStack Client (OSC) or through a Software Developer Kit (SDK), it interfaces with the API for Cinder. This API will talk to a database, for initially storing the request, and then set the status to creating and reserving quota usage. The API will also interact with a messaging queue. The messaging queue will pass requests on to the scheduler for Cinder, which makes decisions about where the change will be made. For example, if a user is requesting that a volume be created, the scheduler will determine which storage device meets the criteria the user is asking for, for the volume (size, disk type), and then send it to the appropriate volume manager. The volume manager for Cinder is what works directly with drivers to interface with the storage back ends. A storage back end may be a datacenter full of Ceph nodes or a proprietary Network-attached Storage (NAS) device that has a driver for Cinder. The volume manager will also be talking to the database to commit to the reserved quota once we know the volume is created successfully. The status of the volume is also set to “available” so the user knows the volume may be used.

See Figure 8.2 for a view into how the individual services work together.

All official drivers that are available for Cinder go through verified testing by the upstream Cinder team in the OpenStack project. To accomplish this, every vendor is required to run continuous integration (CI) tests on all changes that report to the public OpenStack review system. To learn about the latest supported drivers, you can visit the OpenStack Marketplace Drivers page for an official listing: https://www.openstack.org/marketplace/drivers/.

Paying attention to the drivers and learning what is supported will be essential when you build out your production OpenStack deployment. When considering a solution, be sure to research the support for your storage back end of choice and look into factors like how long a solution or vendor has been supported in OpenStack and what they support when it comes to interacting with the Cinder volume manager.
Extending beyond compute-focused deployments, like we saw in Chapter 6, “Private Compute Cloud,” and Chapter 7, “Public Compute Cloud,” with Cinder block storage means you can now offer real persistent storage to your users. Since we’re adding an additional component, a bit of setup needs to be done with Puppet again to configure this storage.

**Controller Node Setup**

If you have your Controller and Compute nodes available from Chapter 5 you will only need to run a single command to add the support for Cinder block storage. In this scenario you will only need to make changes to the controller node. No modifications need to be made to the compute node.

Tip

Did you go through Chapter 7 before this chapter? You should create a new environment. Even though OpenStack is modular, we haven’t designed our foundations modules to function together until you get to Chapter 12, “A Whole Cloud.”

The command is another Puppet apply command, which will process our foundational block storage role in Puppet:

```
$ sudo puppet apply /etc/puppet/modules/deployments/manifets/role/foundations_block_storage.pp
```
This will take some time as it downloads everything required for Cinder and sets up configurations. If anything goes wrong and this command fails, remember that Puppet can also be run with the `--debug` flag in order to show more detail.

While this is running, we can take a look at what this file contains:

```erb
class deployments::role::foundations_block_storage {
  include deployments::role::foundations
  include deployments::profile::cinder
}

include deployments::role::foundations_block_storage

This is calling out to our foundations role, which means if you didn’t set up a foundations role yet for your controller, it will do it now. This is mostly a safety measure; we would still recommend that you run it independently in case you need to do any troubleshooting.

It then calls our Cinder block storage profile, which you can view on the controller filesystem at `/etc/puppet/modules/deployments/manifests/profile/cinder.pp`, and it contains the following:

```erb
class deployments::profile::cinder {
  include ::cinder
  include ::cinder::api
  include ::cinder::ceilometer
  include ::cinder::config
  include ::cinder::db:mysql
  include ::cinder::keystone::auth
  include ::cinder::scheduler
  include ::cinder::volume
  include ::cinder::setup_test_volume

  file { '/etc/init/cinder-loopback.conf':
    owner => 'root',
    group => 'root',
    mode => '0644',
    content => template('deployments/cinder-loopback.conf.erb'),
  }
}
```

The profile pulls in various components to Cinder that we will need. Just like other services in OpenStack, Cinder requires an API, database and Keystone authentication. In case you wish to track usage with Ceilometer’s telemetry service, we also include that. The config is pulled in to help manage arbitrary Cinder configurations you may wish to have. A scheduler in block storage is used in much the same way other OpenStack services use schedulers, to view the requirements the user is requesting for the volume, and then randomly pick a storage device back end that the volume can be created on that meets that criteria. As you may expect, pulling in `cinder::volume` is for the Cinder volume manager. As explained earlier in the chapter, this is what interacts with the drivers actually controlling the storage back end, whether it’s a simple loopback device with LVM (Linux Volume Manager) like we will be using or a proprietary NAS device.
The final lines of this file use the Puppet module’s capability to configure a test volume. For simplicity’s sake we use this setup_test_volume, which creates a simple 10GB file mounted to a loopback (by default, /dev/loop2) device and added to LVM as a single logical group. An init file is also created in our cinder.pp profile to make sure the file is mounted and the volume group is activated if your controller reboots.

**Note**

What Is LVM? The official page for LVM is at: http://www.sourceware.org/lvm2/ and there are various resources and how-tos available for free online, particularly for basic control.

Once your puppet apply command completes, you’re ready to start creating volumes and attaching them to instances!

**Creating and Attaching a Volume: Dashboard**

We will begin with the process for creating and attaching a volume using the OpenStack dashboard (Horizon). With the block storage (Cinder) component now installed, when you log into the dashboard with your test user you will see a section for Volumes in the left under Project in Compute, as show in Figure 8.3.

![Empty, default Volumes page in the dashboard](image)
Creating a Volume

On this page you’ll want to click on the Create Volume button, which will bring up a dialog like the one in Figure 8.4 where you will put in information about the volume you wish to create. Some fields will be automatically filled out, but the rest will be up to you.

The volume name is what you will be using to refer to the volume. A description is optional and can be used for whatever you want, maybe as a reminder to yourself about what the volume is intended for. The volume source enables you to pre-populate the volume with a source of defined data. By default, it queries the Image Storage (Glance) service and enables you, as one of the options, to put an Image on your newly created volume. You may also want to create a volume source that has a basic filesystem and partition table for your new volume so it doesn’t need to be created later after you mount it on an instance. For this scenario, we will just use No source, empty volume and will explain how to partition and format it after it is added to an instance.

The type of volume will inform the scheduler as to which type of storage back end you need to use. From the customer point of view, you want to define a type as tiered and varied storage with different properties, like how fast the storage device is, Quality of Service (QoS) requirements or whether a tier has replication. Prices may vary for the customer based on which options they select. From your perspective, this means one of these tiers may be using Ceph and another a proprietary NAS device that has the desired qualities for the tier being offered. We have not set a volume type, so it will remain as “No volume type” for this example. Our device only has 10GB, so we’ll start out in

![Create Volume dialog](image)

Figure 8.4 Create a volume in the dashboard.
this test by creating a 1GB volume to attach to our instance. The availability zone is identical to the one in compute (Nova) and currently must match the zone where the instance you wish to attach it to resides. In our deployment scenario we only have a single availability zone, so the default of nova should remain selected.

When you have finished, you can click on Create Volume in order to begin volume creation. You will be returned to the Volumes page of the dashboard, which will show your new volume as you can see in Figure 8.5.

**Attaching a Volume**

A volume on its own is not of much value, so we’ll now want to attach it to a compute instance. If you do not have an instance running, you can create a basic one with a CirrOS image now in the Instances dashboard. Refer back to Chapter 6 if you need a refresher on the steps to create an instance.

Attaching a volume in the dashboard is done by going to the drop-down menu on the right side of where your volume is listed. From that menu, select Manage Attachments to bring up the screen, where you can attach the volume to an instance (Figure 8.6).

In this example we have an instance running called “giraffe” and the UUID is also included, since names can be reused in compute (Nova). There is also an optional Device Name section where you can define what you want the device to be named when it’s attached to the instance. This can safely be left blank and a name will be assigned automatically. When you’re done selecting the instance to attach to, click on Attach Volume.
When the volume completes attaching, you will be able to see it in the dashboard as “Attached to” with the instance name and the device it has shown up as (see Figure 8.7).

Figure 8.6  Managing volume attachments

Figure 8.7  A volume has been attached.
You’ll next want to log into the instance to see that the device has been attached successfully, but this process is the same whether you’re completing this process with the dashboard or through the command line. You can continue to learn the process for attaching a volume using the OpenStack Client on the command line, or skip to the “Using the Volume” section later in this chapter to see what you can do to use your new volume.

Creating and Attaching a Volume: OpenStack Client

As we’ve discussed previously, the dashboard can be a convenient way to interact with OpenStack to complete most of the simple operations you may need to do. You will find, however, that most operators prefer using the command line clients or SDKs to interface with the tooling. As such, we’ll now walk through the same process we did with the dashboard but instead using the OpenStack Client (OSC).

The OSC is small and can easily be run from any system that has access to the API endpoints for the services. In our deployment scenarios, this means it must be on the same network as your controller node. You must also have access to the /etc/openrc.test file that was created on your controller and compute nodes, so for these commands we will assume you’re running everything on your controller.

Creating a Volume

We will be using the test user in order to create this volume, since it will also be attaching to a compute instance owned by the test user. To begin, we’ll bring the environment variables for the test user in from the openrc file. Then we can issue the command to create a 1GB instance using that storage back end. Aside from the name, we will be using the same specifications for creation of the volume as was used with the OpenStack dashboard (Horizon), which means creating a 1GB volume that is empty (no partition table, filesystem or data) and is in our default availability zone, called nova.

Tip

You’ll notice that OpenStack commands often output a lot of detail that doesn’t fit well on the pages of a book. A GitHub repository of much of this command output sorted by chapter is available at https://github.com/DeploymentsBook/scripts-and-configs.

```
$ source /etc/openrc.test
$ openstack volume create --size 1 --availability-zone nova seaotter
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| attachments         | []                                   |
| availability_zone   | nova                                 |
| bootable            | false                                |
| consistencygroup_id | None                                 |
| created_at          | 2016-04-15T04:19:46.086611            |
| description         | None                                 |
| encrypted           | False                                |
| id                  | 53372cc5-087a-4342-a67b-397477e1a4f2 |
| multiattach         | False                                |
| name                | seaotter                             |
```
To confirm this volume has been created, you can run a command to list the volumes (Listing 8.1).

As you can see, both the walrus and the seaotter volumes are listed here since they were both created in this chapter. The walrus volume is showing that it is attached to the giraffe instance.

If you need to make changes to a volume, use the openstack volume set command. Running that command alone will give you help output to assist you with making changes to all the parameters before the volume is attached.

**Attaching a Volume**

As mentioned earlier, you can’t do much with a volume if it’s not attached to an instance. You’ll now want to add your new volume to an instance. First you’ll want to see what instances are available:

```bash
$ openstack server list
+--------------------------------------+---------+--------+------------------+
<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Status</th>
<th>Networks</th>
</tr>
</thead>
<tbody>
<tr>
<td>823f2d7a-f186-4453-874d-4021ff2b22e4</td>
<td>giraffe</td>
<td>ACTIVE</td>
<td>private=10.0.0.3</td>
</tr>
</tbody>
</table>
+--------------------------------------|---------|--------+------------------+
```

With confirmation that you have an instance running, you can now run the command to attach the seaotter volume to the giraffe instance:

```bash
$ openstack server add volume giraffe seaotter
```

This command will have no output, but the next time you run `volume list` you will see that the volume has been attached (Listing 8.2).

Since the giraffe instance already had the walrus volume attached as /dev/vdb, you will notice that it has attached the seaotter volume as /dev/vdc.

Congratulations, you have successfully added a Cinder block storage volume to an instance on the command line!

**Using the Volume**

Whether you used the OpenStack dashboard or the command line to create and attach your volume, we will now want to actually confirm the volume was attached and then go ahead and use it with our instance. It may be easiest to use the console in dashboard in order to run the following commands, but if you followed instructions in an earlier chapter so that your CirrOS instance has been set up for SSH (Secure Shell), feel free to use SSH instead.
### Listing 8.1

```
$ openstack volume list
+--------------------------------------+--------------+-----------+------+----------------------------------+
| ID                                   | Display Name | Status    | Size | Attached to                      |
+--------------------------------------+--------------+-----------+------+----------------------------------+
| 53372cc5-087a-4342-a67b-397477e1a4f2 | seaotter     | available |    1 |                                  |
| 54447e7a-d39d-4186-a5b4-3a5f1e773aa | walrus       | in-use    |    1 | Attached to giraffe on /dev/vdb  |
+--------------------------------------+--------------+-----------+------+----------------------------------+
```

### Listing 8.2

```
$ openstack volume list
+--------------------------------------+--------------+--------+------+----------------------------------+
| ID                                   | Display Name | Status | Size | Attached to                      |
+--------------------------------------+--------------+--------+------+----------------------------------+
| 53372cc5-087a-4342-a67b-397477e1a4f2 | seaotter     | in-use |    1 | Attached to giraffe on /dev/vdc  |
| 54447e7a-d39d-4186-a5b4-3a5f1e773aa | walrus       | in-use |    1 | Attached to giraffe on /dev/vdb  |
+--------------------------------------+--------------+--------+------+----------------------------------+
```
Assuming you’re using the dashboard, navigate to the Instances screen in the OpenStack dashboard and in the drop-down menu to the right of the instance you attached it to, select Console to bring you to a console for your instance. Once you’re on the console page, if you’re unable to type in the console, click Click here to show only console and you will be brought to a page that only has the console.

Follow the instructions to log into the instance, and run the following command:

```
$ dmesg
```

There will likely be a lot of output, but the last thing you are likely to see should be something like the following:

```
[  648.143431] vdb: unknown partition table
```

This vdb device is your new block storage (Cinder) volume! At this phase it has no partition table or filesystem, so this will need to be set up using fdisk. Assuming the device is vdb in this example, partitioning can be done with fdisk:

```
$ sudo fdisk /dev/vdb
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0xcf80b0a5.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won’t be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

Command (m for help): n
Partition type:
  p  primary (0 primary, 0 extended, 4 free)
  e  extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-2097151, default 2048): 2048
Last sector, +sectors or +size{K,M,G} (2048-2097151, default 2097151): 2097151
Command (m for help): p

Disk /dev/vdb: 1073 MB, 1073741824 bytes
16 heads, 63 sectors/track, 2080 cylinders, total 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xcf80b0a5

Device Boot Start  End  Blocks  Id  System
/dev/vdb1  2048  2097151 1047552 83  Linux

Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.
```
Now you’ll want to create a basic filesystem on the new disk. It’s only a 1GB volume, and this is a demonstration, so we’ll use the ext2 filesystem:

```
$ sudo mkfs.ext2 /dev/vdb1
mke2fs 1.42.2 (27-Mar-2012)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
65536 inodes, 261888 blocks
13094 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=268435456
8 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376
Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

The last step is creating a mount point and mounting your new volume. Let’s say that you want to use this volume for photos and create a directory for that. Then we’ll check to confirm it’s the size we expect it to be.

```
$ mkdir photos
$ sudo mount /dev/vdb1 photos/
$ df -h | grep vdb1
/dev/vdb1        1006.9M   1.3M  954.5M   0% /home/cirros/photos
$ df -h /dev/vdb1
Filesystem   Size  Used  Available  Use% Mounted on
/dev/vdb1     1006.9M   1.3M  954.5M   0% /home/cirros/photos
```

Congratulations! A 1GB volume from the block storage service Cinder is now mounted on your system. Note that this was mounted using the root user, so you will need to either change the ownership to your user or use root to place files on it.

**Tip**

File permissions with block storage in Cinder can be a tricky thing to master. When using a Linux filesystem, files are referenced by user and group ID (UID and GID). Unless you are very diligent about keeping things consistent across your instances through something like custom images with default users and group or configuration management where IDs are specifically defined, these IDs can easily be different between machines.

As a result of these potential differences in IDs, mounting a volume from the block storage device on one instance and then detaching it to add it to another instance may land you in a situation where the ownership of files looks all wrong. Keep this in mind as you begin experimenting with moving volumes and always make checking permissions a step in your plans to move a volume to another instance.
Automation
As we explained in our chapters about private and public clouds, you don’t only need to interact with OpenStack through the OpenStack dashboard or OpenStack client. Instead you may interact with the APIs through various SDKs, which you can learn about at http://developer.openstack.org/.

Summary
The need for expanding and moving storage of data is growing in modern environments, and Cinder block storage fits that need. It offers a variety of storage back-end drivers to support everything from open source tooling like LVM and Ceph to a host of tested proprietary storage solutions from an array of vendors. The deployment scenario using Puppet walked you through creating a volume and then attaching it to an instance where you could use it.
A
Access & Security
DevStack, 25–26
private compute cloud, 123–126, 132–134
Account daemon, Swift, 170
Account_quotas proxy manifest, Swift, 175
Add Rule, DevStack, 124
Administrators
configuring Glance, 67–71
configuring Keystone, 62–66
dashboard for, 4–5
DevStack deployment, 28–29
management network for, 35–36
Nova installation, 72–75
ADMIN_PASSWORD placeholder, OSC configuration, 67
Admin_token, Keystone, 59
Agents, Neutron
definition of, 77
installing Neutron, 78–81, 83
other possibilities for, 96–97
Alerts
Ceiometer, 11
public compute cloud, 150
Alphabetical order, of OpenStack release cycle, 13–14
AMQP (Advanced Message Queuing Protocol) framework, 96, 216
Ansible playbooks, 93, 188, 235–236
Aodh (Telemetry Alarming service), 150
Apache Mesos, 195, 198
Apache module, 102
Apache web server, configuring Keystone, 61–62
API (Application Programming Interface)
Aodh, 150
Cinder, 154–155, 156, 166
Glance, 69–70
Glance profile, 101
Ironic, 186–187
Magnum, 197–198
Nova, 73
power of OpenStack in, 28
SDKs and OpenStack, 138
Swift, 171–172
testing compute node, 111
Architecture
bare metal provisioning, 186–188
block storage cloud, 154–155
public compute cloud, 143–144
Archives, OpenStack mailing list, 244
Authentication. See also Credentials
controller node setup for Swift, 174–175
default passwords used in this book, 92, 98
foundations for deployments, 95
identity vs., 95
Keystone identity service for, 9
Keystone identity service, troubleshooting, 217
log files and, 213
objects storage cloud, 174–175
OpenStack Client, 249–250
testing compute node, 111
your first OpenStack, 58–59, 66, 69
Authtoken proxy manifest, Swift, 175
Automation, Cinder block storage, 166
AWS CloudFormation Template (CFN) format, Heat, 13
AZ (Availability Zone), DevStack, 27
B
Back ends, Glance profile, 101
Backup
Cinder block storage, 153
troubleshooting in debug mode, 215
Bare metal. See Ironic bare metal provisioning
Base profile, controller node deployment, 100–101
Bays, Magnum, 198–199
Bifrost, Ironic deployment with, 188
Big Tent initiative, 13
Block storage. See Cinder block storage
br-ex bridge
installing Neutron, 81
qrouter-* network namespace and, 47
testing OVS in controller node, 107–108
traffic flow on controller node, 45–46
br-int bridge
qrouter-* network namespace and, 47
testing OVS in controller node, 107–108
traffic flow in compute node, 48–49
traffic flow on controller node, 45, 46
br-tun bridge
testing OVS in controller node, 107–108
traffic flow in compute node, 48–49
traffic flow on controller node, 45–46
Branches, stable, 32
Breakage, mitigating in deployments, 223–224
Bridges
definition of, 46
initial networking setup, 53
testing OVS in controller node, 107–108
traffic flow on controller node, 45–48
troubleshooting Linux, with OVS, 220
virtual machine deployment setup, 42
Bug tracking, OpenStack, 246
Build numbers, log storage, 168

C
Cache proxy manifest, Swift, 175
Cache_errors proxy manifest, Swift, 175
Cattle vs. pets metaphor, cloud computing, 152
Ceilometer-agent-central service, 12
Ceilometer-agent-computer service, 12
Ceilometer-agent-notification service, 12
Ceilometer-api service, 12
Ceilometer-collector service, 12
ceilometer meter-list command, 147, 207
ceilometer resource-list command, 147
Ceilometer telemetry service
adding to your deployment, 142–144
controller node setup for Cinder, 156
controller node setup for public cloud, 144–145
as key component, 11–12
metrics and alerts, 150
OSC view of, 207
viewing statistics with dashboard, 146–147
viewing statistics with OSC, 147–149
in whole cloud deployment with dashboard, 205–206
CFN (AWS CloudFormation Template) format, Heat, 13
Chat, IRC. See IRC (Internet Relay Chat)
Chef cookbooks, 93, 188, 235
CI (Continuous Integration) system
Cinder drivers, 154
DevStack, 22
infrastructure, 247–248
Cinder-api daemon, 10
Cinder-backup daemon, 10
Cinder block storage
about, 151
architecture overview, 154–155
automation, 166
cloud provider, 151–152
controller node setup, 155–157
creating/attaching volume with dashboard, 157–161
creating/attaching volume with OSC, 161–162
data processing, 152
keeping backups, 153
overview of, 10–11
requirements, 153–154
scenario, 155
uses, 151–153
using volume, 162–165
whole cloud deployment with dashboard, 205–206
Cinder scheduler daemon, 10
Cinder-volume daemon, 11
CirrOS image
configuring Glance, 70–71
deploying Glance, 97
DevStack deployment as user, 27–28
launching dashboard for private compute cloud, 122–123
launching test instance on dashboard, 112–115
testing compute node, 111
CLI (Command Line Interface) tools, 6, 28, 30–31
Cloning volumes, block storage backups, 153
Cloud
  block storage. See Cinder block storage building your, 3
  hosting company using Ironic for, 185–186
  object storage. See Swift object storage
  Open Stack joins, 2–3
  pets vs. cattle metaphor in, 152
  private compute. See Private compute cloud
  public compute. See Public compute cloud
Cloud Archive, Ubuntu, 15–16
Cloud, deploying whole
  additional components, 208–209
  compute node setup, 205
  controller node setup, 204–205
  exploring with dashboard, 205–206
  exploring with OSC, 207
  high availability and scaling, 208
  overview of, 203
  requirements, 203–204
  scenario, 204
  uses, 203
Cloud-init Python tooling, 128
CloudKitty, 150
Cluster, Swift, 170–172
Code contributions to OpenStack
  bug and feature tracking, 246
  communication, 244–245
  Git and code review, 246–247
  other contributions, 248
  overview of, 243
  release cycle, 243–244
  specifications, 245–246
  testing infrastructure, 247–248
COE (Container Orchestration Engine), Magnum, 197–198
Columns, default flavor, 120–121
Command Line Interface (CLI) tools, 6, 28, 30–31
Commands
  OpenStack, 106
  OSC framework for, 250
  OSC frequently used, 251
Comments, stack.sh script with, 23
Communication, OpenStack contributions, 244–245
Community, OpenStack contributions to. See Code contributions to OpenStack
  finding help within, 253–256
  Ironic development in, 192
  requesting help from, 224
Compute. See Nova compute
Compute node
  basic configuration, 57–58
  block storage volume benefits, 151–152
  Cinder block storage, 154
  Compute profile, 110
  with containers, 197
  creating, 109–110
  creating first OpenStack, 51–52
  deploying whole cloud, 205
  deployment, 109–115
  deployment requirements, 39–42, 91–92
  horizontal scaling of, 208
  launching test instance on dashboard, 112–115
  Nova compute component handling, 7
  Nova configuration, 83–84
  private compute cloud requirements, 92
  public compute cloud requirements, 143
  public compute cloud setup, 145
  Swift object storage, 169
  testing, 111–112
  traffic flow within, 43–44, 48–49
  using object in web page, 183–184
Compute profile, compute node, 110
Conductor daemon, Nova, 73
Conferences, finding help in OpenStack, 255
Configuration files, troubleshooting, 221
Connecting to Swift, 171–172
Console
  accessing daemons from, 8
  launching test instance on dashboard, 113–114
Consoleauth, Nova, 73
Container Orchestration Engine (COE), Magnum, 197–198
Container quotas proxy manifest, Swift, 175
Containers
  drivers for Nova, 197
Magnum. See Magnum containers
overview of, 195
understanding, 195–196
uses, 196–197
Containers, Swift
  concept of, 171
  connecting to, 171–172
  controller node setup, 173–176
  creating in dashboard, 176–179
  creating in OSC, 179–181
  overview of, 170
  using objects, 181–184
Continuous Integration (CI) system
  Cinder drivers, 154
  DevStack, 22
  infrastructure, 247–248
Contributions. See Code contributions to OpenStack
Control–c, stopping capture, 219
Control plane, networking, 36
Controller node
  adding support for Cinder block storage, 155–157
  basic configuration, 55–57
  configuration requirements, 51–52
  configuring Glance, 67–69
  configuring Keystone, 58–67
  configuring Neutron, 76–83
  configuring Nova, 72–75
  configuring OSC, 67
  with containers, 197
  traffic flow within, 44–48
Controller node, deployment
  base profile, 100–101
  Glance imaging service profile, 101
  Horizon dashboard profile, 102
  Keystone identity profile, 102–103
  Neutron networking profile, 103–105
  Nova compute profile, 103
  overview of, 98
  Puppet roles, profiles and modules, 99–100
  RabbitMQ queuing profile, 105
  requirements, 39–43
  running puppet apply command, 98–99
  testing, 105–109
Controller node setup
  Cinder block storage, 154
  deployment, 91–92
  deployment of whole cloud, 204–205
  initial networking, 53
  private compute cloud, 92
  public compute cloud, 143–145
  Swift object storage, 169, 173–176
Core competencies, cost of public cloud, 227
Count, DevStack deployment as user, 27
CPU
  DevStack requirements, 22
  simple deployment requirements, 52
Create Container screen, Swift, 176–178
Create Flavor, 121–122
Create Image, 122–123
Create Key Pair, 125–126
Create Volume button, dashboard, 158
Credentials. See also Authentication
  base profile openrc files for, 100
  Horizon dashboard user login, 25
  OpenStack interactive mode, 250–251
  OSC authentication, 251
  testing in controller node, 107
Customization
  DevStack, 32–33
  flavors, 120–122
  ...
Daemons
  Cinder, 10–11
  console access to instances, 8
  Glance, 10
  Heat, 13
  Keystone, 9
  Neutron, 10
  Nova, 7–8, 97
  Swift, 170
Dashboard. See Horizon dashboard
Data plane, networking, 36–38
Data processing, Cinder block storage, 152
Data sovereignty, and public cloud, 228
Databases
  controller node setup for authentication in
    Cinder, 156
  hosting with Ironic, 186
MySQL. See MySQL database
Debug lines, log messages, 215–216
Debug mode, 215
Debugging. See Troubleshooting
Deployment
other mechanisms for, 235–236
reference. See Reference deployment
Deployment, DevStack
common failures, 23–24
dashboard login as administrator, 28–29
dashboard login as user, 24–28
overview of, 23
working with host on command line, 30–31
Deployment, first
compute node setup, 83–85
configuring Glance, 67–69
configuring Keystone, 58–67
configuring Neutron, 76–83
configuring Nova, 72–75
configuring OpenStack client, 67
initial setup, 52–58
managing an instance, 85–87
overview of, 51
requirements, 51–52
using reference deployment, 52
Deployment foundations
common mechanisms, 93–94
compute node, 109–115
controller node. See Controller node, deployment
Glance image service, 97
Horizon dashboard, 97
initial setup, 94–95
Keystone identity service, 95
MySQL database, 96
networking, 93
Neutron networking, 96
Nova compute, 97
overview of, 91
RabbitMQ message Queuing, 96
reference, 92–93
requirements, 91–92
troubleshooting, 115
Designate: DNS as a service, 209
Details screen, DevStack, 27
Developers, DevStack, 22, 31–32
Devices, used by Rings in Swift, 170
DevStack
customizing, 32–33
defined, 21
deployment, 23–31
deployment limitations, 93
overview of, 21
requirements, 22–23
stable, 24, 31–32
summary review, 33
usage of, 22
using Ironic with, 188
DHCP service
initial networking setup, 53
ip/namespace ID for, 218
Directory, customizing DevStack, 33
Docker
container drivers for Nova, 197
Magnum managing orchestration for, 13
popularity of containers due to, 195
using containers, 196–197
Docker Swarm
Magnum containers, 198
orchestration for containers via, 195
public cloud company uses for, 196
Documentation, finding help, 253
Domains, Keystone, 64–65
Drivers
Cinder block storage, 154
Ironic, 12, 186–189
Ironic Inspector defining, 191
Neutron, 96–97
Nova, 7, 197
Encapsulation of traffic, tunnel-driven networks, 38
Environment variables
configuring Glance, 67–71
configuring Keystone, 62–66
export commands, 66
passing data within openstack command, 67
Ephemeral Disk column, default flavors, 120–121
Errors
troubleshooting. See troubleshooting
troubleshooting reading displayed, 211–213
/etc/ network/interfaces file, 55
/etc/openrc.admin
  adding flavor, 130
  exploring deployment with OSC, 207
  export commands and, 62
  Glance installation, 67–68, 71
  loading admin credentials for cloud, 85
  Neutron installation, 76
  Nova installation, 72
  OSC authentication, 68, 249
  testing controller node, 107
  viewing statistics with OSC, 147
/etc/openrc.test
  adding image, 71
  creating/attaching volume in OSC, 161
  creating container/object in OSC, 180
  managing instance, 85
  OSC authentication, 67, 249
  private cloud, 130–132, 134
  testing compute node, 111
  testing controller node, 106–107
  using object in OSC, 182
eth0 interface
  configuring controller node, 55
  initial networking setup, 53
  physical machine setup, 41
  tcpdump command on, 219
  traffic flow within, 44–48
  virtual machine setup, 43
eth1 interface
  configuring controller node, 55–56
  physical machine setup, 41
  traffic flow within, 44–48
  virtual machine setup, 43
Exports
  configuring Glance, 71
  configuring OSC, 67
  of environment variables available in future, 66
  not preserving across shells/sessions, 62
External network, deploying, 40–43
F
Failures
  DevStack deployment, 23
  Storage hardware, 152
  High Availability and Scaling, 208
  Troubleshooting, 211
  Logs, 213
  Connecting to AMQP, 216
  Fault tolerance, Swift, 172
  Features, specifications for new, 245
  Features, tracking of, 246
  Fernet tokens, configuring Keystone, 60–62, 102
  File permissions, Cinder block storage, 165
  File sharing, with object storage, 167–168
Files
  creating on new disk, 165
  creating volume in dashboard, 158
  Firewall plug-in, Neutron, 96
  First deployment. See Deployment, first
  Flat networking, tenant network, 37
Flavors
  adding to private compute cloud, 120–122
  configuring Glance, 69–70
  definition of, 120
  in DevStack deployment, 27, 28–29
  launching instance with OSC, 130–131
  launching test instance on dashboard, 113
  launching with dashboard, 126–127
  Floating IP address, adding to instance, 128–130, 135–136
  Formpost proxy manifest, Swift, 175
  Foundations role, Cinder, 156
G
  The Gate, Continuous Integration testing, 247
  Gateway address, logging in with SSH, 136
  Gearman worker, continuous integration testing, 247–248
  Generic Routing Encapsulation (GRE), 38
Git repository
  code revision control, 246–247
  continuous integration testing, 247–248
  scripts-and-configs, 55
  troubleshooting Puppet modules, 222
GitHub repositories
  initial setup of Puppet, 94–95
  OpenStack commands in, 106, 130, 161
  troubleshooting via, 115
Glance-api daemon, 10
Glance imaging service
  adding image to, 122–123
  creating volume, 158
  dashboard for private cloud, 122–123
  installing on controller, 67–69
  as key component, 10
  overview of, 10
  profile, 101
  selecting for deployment, 97
  using Ironic, 189–191
  viewing statistics with dashboard, 146–147
Glance-registry daemon, 10
Government organization, private cloud in, 117–118
Government research organization, hybrid cloud in, 229
GRE (Generic Routing Encapsulation), 38
Groups, DevStack deployment, 29
Growth potential, public cloud, 227

H
HA (high-availability), 92
HA (high availability), 208
Hardware
  defining with Ironic Inspector, 191
  enrollment, using Ironic, 188
  managing Ironic, 192
Healthcheck proxy manifest, Swift, 175
Heat-api, 13
Heat-api-cfn, 13
Heat-engine, 13
Heat orchestration
  as key component, 12–13
  Magnum containers, 13, 197
Heat Orchestration Template (HOT) format,
  Heat, 13
Help
  finding with OpenStack, 253–256
  Puppet, 222
  requesting, 224
  troubleshooting via, 115
Hiera
  defining software lifecycle for data, 239
  masterless Puppet and, 237
  passwords in, 238
  understanding, 237–238
Hiera common.yaml file
  configuring controller node, 98
  creating compute node, 109
  Keystone identity profile, 102
  troubleshooting Puppet modules, 222
High availability (HA), 92, 208
High-Performance Computing (HPC),
  Ironic, 186
Horizon dashboard
  accessible from any address, 92
  in Cinder block storage, 162–165
  console access, 8
  creating/attaching volume, 157–161
  creating container/object in Swift, 176–179
  in DevStack deployment, 22–23, 24
  as key component, 4–7
  launching instance for private cloud,
    119–130
  launching test instance, 112–115
  launching test instance, error, 212–213
  login as user, 24–28
  navigating to IP address of controller, 120
  profile, 102
  selecting for deployment, 97
  testing in controller node, 108–109
  using objects, 181–182
  viewing public cloud statistics, 146–147
  in whole cloud deployment, 205–206
Host(s)
  configuring controller node, 56
  customizing DevStack for multiple, 33
  definition of, 42
  using DevStack on command line, 30
HOT (Heat Orchestration Template) format,
  Heat, 13
HPC (High-Performance Computing),
  Ironic, 186
HTTP server
  loading simple, 137–138
  prelaunching instance with OSC, 132–134
  using object in web page, 183–184
Hybrid clouds, 228–229
Hyper-V
  instances as VMs using, 4
  Nova drivers supporting, 7
  as supported hypervisor, 146
Hypervisors, supported, 145
IaaS (Infrastructure as a Service). See also Private compute cloud, 2
ICMP ping, 123–126
Identity
  authentication vs., 95
  Keystone. See Keystone identity service
Identity Panel, DevStack deployment, 28–29
IDs, file permissions in Cinder, 165
Images, container, 195–196
Imaging service. See Glance imaging service
Img file, Glance, 10
Import Key Pair, access/security, 125–126
In-tree drivers, Ironic, 188
Infrastructure as a Service (IaaS). See also Private compute cloud, 2
INI format files, troubleshooting configuration files, 221
Installation
  Magnum, 199
  reference deployment, 231–233
Instance Boot Source, DevStack deployment, 27
Instance(s)
  add floating IP address with dashboard, 128–130
  attaching volume in dashboard to, 158–159
  compute node as host for all compute, 109
  connecting to tenant networks, 36–37
  definition of, 42
  DevStack deployment as administrator, 28–29
  DevStack deployment as user, 25–28
  launching for private cloud, 126–128
  launching from command line, 31
  launching test on dashboard, 112–115
  launching using OSC, 134–136
  loading simple HTTP server, 137–138
  managing, 6, 85–87
  modifying through Horizon interface, 28
Ironic-api, 12
  architectural, 186–188
  development community, 192
  installation, 188
  IPMI for queries against, 145
  Ironic Inspector, 191
  as key component, 12
  Magnum structure, 198
int-br-ex bridge, 45
Intelligent Platform Management Interface (IPMI). See IPMI (Intelligent Platform Management Interface)
Interactive mode, OpenStack Client, 250–251
Interfaces, traffic flow, 45–49
Internal cloud deployment, 186
Internet access, physical machine deployment, 41
Internet Relay Chat (IRC)
  contributions to OpenStack, 244–245
  finding help with OpenStack, 224, 254
IP address(es)
  adding floating, 128–130, 135–136
  configuring controller node, 56
  setting for controller and compute node, 98
  tenant networks and, 37
ip command
  MTU size issues, 219–220
  network namespaces and, 217–218
ip netns command, 218
ip netns exec command, 219
IPMI (Intelligent Platform Management Interface)
  Ironic drivers supporting, 12
  for queries against bare metal, 145
  using Ironic, 187–191
Iptables
  definition of, 46
  initial networking setup, 53
  Linux Bridge for, 220
  traffic flow on controller node, 47
  troubleshooting networks, 220–221
IRC (Internet Relay Chat)
  contributions to OpenStack, 244–245
  finding help with OpenStack, 224, 254
Ironic-api, 12
Ironic bare metal provisioning
  architecture, 186–188
  development community, 192
  installation, 188
  IPMI for queries against, 145
  Ironic Inspector, 191
  as key component, 12
  Magnum structure, 198
managing, 192
overview of, 185
uses, 185–186
using Cinder, 188–191
Ironic-conductor, 12, 187
ironic node-create command, 190
ironic node-list command, 192
ironic node-show name command, 192
Ironic-python-agent, 12
ISO image file, Glance, 10

J
Jenkins user comments, CI testing, 247–248
JSON format files, troubleshooting configuration files, 221
Jumbo frames, networking, 39

K
Kernel-based Virtual Machine. See KVM (Kernel-based Virtual Machine)
Key Pair
DevStack deployment as user, 25–27
launching, 126–127
prelaunch access/security, 125
Key services, troubleshooting, 216–217
Keys, Fernet, 60
Keystone identity service
for Cinder, 156
for DevStack, 24
for Glance profile, 101
installing on controller, 58–67
as key component, 9
for Magnum containers, 197
for Neutron, 76–78
profile, 102–103
running Swift with other identity service, 11
selecting for deployment, 95
for Swift, 174–175
troubleshooting, 216–217
understanding, 58–59
keystone-manage command, 60–61
keystonauth proxy manifest, Swift, 175
Kubernetes
Magnum concepts, 198
orchestration for containers, 13, 195
public cloud company uses, 196
KVM (Kernel-based Virtual Machine)
accessing OpenStack instances, 9
adding image for private cloud, 122
container drivers for Nova, 197
instances as VMs using, 4
networking with, 53
Nova drivers supporting, 7
in reference deployment, 52, 92–93, 231–233
as supported hypervisor, 146
using virtual machines, 42

L
Launch Instance button
DevStack deployment, 24–25, 27
private compute cloud, 126–127
testing dashboard, 112, 114
Libvirt KVM, 197, 231–233
Libvirt LXC, 197
Linux Bridge
Open vSwitch vs., 37
troubleshooting with iptables, 220–221
troubleshooting with OVS, 220
Linux Containers (LXC), 146, 195, 197
Linux kernel executable (vmlinuz), Ironic, 189–191
Load balancer plug-in, Neutron, 96
Local.conf file, customizing DevStack, 32–33
Location, and public cloud, 227
Lock-in, vendor, 229–230
Login
to server with SSH, 136–137
testing Horizon dashboard in controller node, 108–109
Logs
customizing DevStack, 33
troubleshooting key services, 216
troubleshooting reading displayed errors, 211–213
troubleshooting via, 213–216
using object storage for, 167–168
Long Term Support (LTS), Ubuntu release, 15–16
LTS (Long Term Support), Ubuntu release, 14–16
LVM, Cinder, 157
<table>
<thead>
<tr>
<th>M</th>
<th>MAC addresses, using Ironic, 190</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magnum containers</td>
<td>concepts, 198</td>
</tr>
<tr>
<td></td>
<td>installing, 199</td>
</tr>
<tr>
<td></td>
<td>as key component, 13</td>
</tr>
<tr>
<td>OpenStack Instances as</td>
<td>overview of, 197–198</td>
</tr>
<tr>
<td></td>
<td>understanding, 195–196</td>
</tr>
<tr>
<td>Mailing lists</td>
<td>finding help in OpenStack, 224, 254</td>
</tr>
<tr>
<td></td>
<td>OpenStack Development list, 244</td>
</tr>
<tr>
<td></td>
<td>OpenStack Operators, 245</td>
</tr>
<tr>
<td></td>
<td>OpenStack support, 245</td>
</tr>
<tr>
<td>Major company, using private compute cloud</td>
<td>118</td>
</tr>
<tr>
<td>Manage Attachments, dashboard</td>
<td>158–159</td>
</tr>
<tr>
<td>Management plane, networking</td>
<td>35–36</td>
</tr>
<tr>
<td>Management, with Ironic</td>
<td>192</td>
</tr>
<tr>
<td>Master, Puppet</td>
<td>237</td>
</tr>
<tr>
<td>Masterless Puppet</td>
<td>237</td>
</tr>
<tr>
<td>Maximum transmission units (MTUs)</td>
<td>38–39, 219–220</td>
</tr>
<tr>
<td>MD5SUMS</td>
<td>122</td>
</tr>
<tr>
<td>Meeting channels, IRC</td>
<td>245</td>
</tr>
<tr>
<td>Memcached, Swift</td>
<td>174–175</td>
</tr>
<tr>
<td>Message Queuing</td>
<td>Cinder block storage architecture, 154–155</td>
</tr>
<tr>
<td></td>
<td>Ironic architecture, 187</td>
</tr>
<tr>
<td></td>
<td>RabbitMQ. See RabbitMQ Queuing service</td>
</tr>
<tr>
<td>Metadata column, default flavors</td>
<td>120</td>
</tr>
<tr>
<td>meter-list command</td>
<td>147–149</td>
</tr>
<tr>
<td>Metering data</td>
<td>Ceilometer, 11</td>
</tr>
<tr>
<td>Meters</td>
<td>Ceilometer and</td>
</tr>
<tr>
<td></td>
<td>viewing statistics with command line client</td>
</tr>
<tr>
<td>Metrics, public compute cloud</td>
<td>150</td>
</tr>
<tr>
<td>Migration, to cloud</td>
<td>230</td>
</tr>
<tr>
<td>Milestone, release cycle</td>
<td>244</td>
</tr>
<tr>
<td>Mission statement, OpenStack</td>
<td>2</td>
</tr>
<tr>
<td>Mitaka</td>
<td>54, 145</td>
</tr>
<tr>
<td>Mitigating breakage, troubleshooting</td>
<td>223–224</td>
</tr>
<tr>
<td>ML2 (Modular Layer 2) plug-in, with OVS</td>
<td>37</td>
</tr>
<tr>
<td>Modular Layer 2 (ML2) plug-in, with OVS</td>
<td>37</td>
</tr>
<tr>
<td>Modules, OpenStack Puppet</td>
<td>building own composition, 240–241</td>
</tr>
<tr>
<td></td>
<td>management of, 238–239</td>
</tr>
<tr>
<td></td>
<td>more information on, 241</td>
</tr>
<tr>
<td></td>
<td>overview of, 99–100</td>
</tr>
<tr>
<td></td>
<td>roles vs., 99</td>
</tr>
<tr>
<td></td>
<td>troubleshooting, 222</td>
</tr>
<tr>
<td>Monitoring, via management network</td>
<td>36</td>
</tr>
<tr>
<td>Mounting volume, Cinder block storage</td>
<td>165</td>
</tr>
<tr>
<td>MTUs (Maximum transmission units)</td>
<td>38–39, 219–220</td>
</tr>
<tr>
<td>Murano: Application catalog</td>
<td>209</td>
</tr>
<tr>
<td>MySQL database</td>
<td>configuring controller node, 56–57</td>
</tr>
<tr>
<td></td>
<td>configuring Glance, 67–70</td>
</tr>
<tr>
<td></td>
<td>configuring Keystone, 59, 60</td>
</tr>
<tr>
<td></td>
<td>configuring Nova, 72</td>
</tr>
<tr>
<td></td>
<td>Glance profile, 101</td>
</tr>
<tr>
<td></td>
<td>installing Neutron on controller, 78</td>
</tr>
<tr>
<td></td>
<td>installing Nova on controller, 72–75</td>
</tr>
<tr>
<td></td>
<td>Ironic architecture, 187</td>
</tr>
<tr>
<td></td>
<td>selecting for deployment, 96</td>
</tr>
<tr>
<td></td>
<td>testing in controller node, 105–106</td>
</tr>
<tr>
<td>N</td>
<td>n-api screen, 30</td>
</tr>
<tr>
<td>Namespaces</td>
<td>definition of, 46</td>
</tr>
<tr>
<td></td>
<td>ip command and</td>
</tr>
<tr>
<td></td>
<td>managing instances, 87</td>
</tr>
<tr>
<td></td>
<td>qrouter-* network, 47</td>
</tr>
<tr>
<td>Naming</td>
<td>custom flavors, 121</td>
</tr>
<tr>
<td></td>
<td>instances, 87</td>
</tr>
<tr>
<td></td>
<td>logs for storage, 168</td>
</tr>
<tr>
<td>NAT (Network Address Translation)</td>
<td>definition of, 46</td>
</tr>
<tr>
<td></td>
<td>initial networking setup, 53</td>
</tr>
<tr>
<td></td>
<td>traffic flow on controller node, 45–47</td>
</tr>
<tr>
<td>ntrns exec command</td>
<td>87</td>
</tr>
<tr>
<td>Network Time Protocol (NTP)</td>
<td>base profile setup, 100</td>
</tr>
<tr>
<td></td>
<td>ntp daemon, 56, 58</td>
</tr>
<tr>
<td>Network Time Protocol (ntp) daemon</td>
<td>58</td>
</tr>
<tr>
<td>Networking</td>
<td>See Neutron networking</td>
</tr>
</tbody>
</table>
Networking Guide, 49
Neutron networking
  compute mode installation of, 84–85
  compute node setup, 57–58
  configuring first OpenStack, 55–56, 76–83
  controller node setup, 55–56
  deployment requirements, 39–43, 92–93
  DevStack deployment failures, 23
  exploring possibilities with, 96–97
  in Horizon, 6–7
  initial setup, 52–53
  as key component, 9–10
  Magnum containers, 197
  Maximum Transmission Units, 38–39
  other resources, 49
  planes of operations, 35–36
  profile, 103–105
  provider networks, 36
  selecting for deployment, 96
  significance of, 35
  tenant networks, 36–38
  traffic flow, 43–48
  traffic flow on compute mode, 48–49
  troubleshooting, 217–221
  troubleshooting as key service, 216–217
Neutron-server, Neutron, 10
NICs
  reference deployment, 231–233
  simple deployment requirements, 51–52
  virtual machine deployment setup, 42
Nodes
  classification of Puppet, 238
  cleaning Ironic, 192
  definition of, 42
  Magnum, 198–199
  networking deployment, 39–43
Nova-api daemon
  defined, 8
  deploying Nova in controller node, 97
  screen session for DevStack, 30–33
Nova-api-metadata daemon, 8, 97
Nova compute
  container drivers, 197
  DevStack deployment as user, 25
  installing in compute mode, 83–84
  installing on controller, 72–75
  as key component, 7–9
  profile, 103
  restarting all services, 81
  selecting for deployment, 97
  viewing statistics with dashboard, 146–147
Nova-compute daemon, 7–8, 83, 97
Nova-conductor daemon, 7, 97
Nova-consoleauth daemon, 8
Nova-novncproxy daemon, 8
Nova-scheduler daemon, 7, 97
Nova-spicehtml5proxy daemon, 8
Nova-xvpvncproxy daemon, 8
Novncproxy daemon, 73
NTP (Network Time Protocol)
  base profile setup, 100
  ntp daemon, 56, 58
O
Object daemon, Swift, 170
Object storage. See Swift object storage
Object Store, Swift
  backing up block data to, 153
  creating container/object in dashboard, 176–178
  using objects, 181–184
Objects, Swift
  adding to container in dashboard, 176–179
  adding to container in OSC, 179–181
  connecting to Swift, 171–172
  getting details about, 177, 179–181
  overview of, 171
  using, 181–184
Online gaming company, using containers, 196
Open source
  OpenStack commitment to, 3
  Puppet modules deploying OpenStack, 99–100
  vendor-specific drivers/agent plug-ins for Neutron, 96–97
Open vSwitch. See OVS (Open vSwitch)
Openrc credentials, testing compute node, 111
Openrc files, base profile, 100
OpenSSH, reference deployment, 231
Openssh-server package, reference deployment, 233
OpenStack
building cloud, 3
cloud, 1–2
OpenStack joins the cloud, 2–3
release cycle, 13–16
summary review, 17
understanding, 1
uses, 3–4
OpenStack Administrator Guide, 208
OpenStack Ansible Deployment (OSAD), 93
OpenStack Client. See OSC (OpenStack Client)
OpenStack Continuous Integration,
launching DevStack, 31–32
OpenStack Design Summit and Conference,
3, 243–244
OpenStack-dev mailing list, 244
openstack endpoint create command, 250
OpenStack Foundation
creation of, 2
mailing lists, 254
Marketplace. See OpenStack Marketplace
OpenStack Summit, 3, 255
OpenStack Infrastructure team, hybrid cloud,
228–229
OpenStack Infrastructure User Manual, 247
OpenStack, key components
baremetal (Ironic), 12
block storage (Cinder), 10–11
compute (Keystone), 9
compute (Nova), 7–9
containers (Magnum), 13
dashboard (Horizon), 4–7
image service (Glance), 10
Instances, 4
networking (Neutron), 9–10
object storage (Swift), 11–12
orchestration (Heat), 12–13
other projects, 13
Queuing, 4
telemetry (Ceilometer), 11–12
OpenStack Kilo release cycle, 2015, 13
OpenStack Marketplace
finding drivers at, 97, 154
finding help at, 256
finding vendors at, 226, 236, 256
public cloud companies, 141
openstack network list command, Ironic, 191
Openstack-operators mailing list, 244
OpenStack project, hybrid cloud
deployments, 228–229
openstack server create command, Ironic, 191
openstack server list command, testing
compute node, 112
openstack server show ferret command,
112, 135
OpenStack Summit, 3, 255
OpenStack Technical Committee (TC), 13,
209, 243
openstack token issue command, 250
OpenStack Trove, 186
Openstack-user mailing list, 244
Openstacklib module, 101
Operating system
DevStack requirements, 22
initial network setup of, 54
Ops Meetups, finding OpenStack help, 255
Orchestration. See Heat orchestration
Organizations
cost of public cloud and, 227
private compute cloud use across, 117–118
OSAD (OpenStack Ansible Deployment), 93
OSC (OpenStack Client)
authentication, 249–250
base profile setup, 100–101
commands, 250
creating/attaching volume, 161–163
creating container/object in Swift, 179–181
exploring whole cloud deployment, 207
installing on controller, 67
interactive mode, 250–251
launching instance, 128–130
overview of, 249
quick reference for commands, 251
troubleshooting reading displayed errors,
213
using objects, 182
viewing statistics on public cloud, 147–149
Oslo, 74–75, 78–79
OVS (Open vSwitch)
concept of bridge in, 46
configuring controller node, 56
installing Neutron in compute mode, 85
installing Neutron on controller, 78–80
network deployment, 40–43
restarting in development environment, 220
segmented networks using, 37
testing in controller node, 107–108
traffic flow on controller node, 45–48
troubleshooting bridges with, 220
in tunnel-driven networks, 38
ovs-vsctl command, 220

P
PaaS (Platform as a Service), 2
Packages
  configuring compute node, 58
  configuring controller node, 56
  configuring Keystone, 59
  installing Neutron, 77
  installing Nova, 73–74
  Puppet, 239, 240
  reference deployment, 231
Partition table, creating volume in dashboard, 158
Partitions
  Swift vs. standard disk, 171
  used by Rings in Swift, 170–171
Passwords
  controller node, 56, 98
  DevStack deployment, 23, 28–29
  Glance, 68–71
  Hiera, 238
  Horizon dashboard user login, 25
  installing Neutron, 76, 78, 80
  installing Nova, 72–75, 83
  Keystone, 64–66
  managing instances, 87
patch-int patch, 46
patch-tun patch
  testing OVS in controller node, 107–108
  traffic flow in compute node, 48–49
  traffic flow on controller node, 45–46
Patches
  definition of, 46
  testing OVS in controller node, 107–108
  traffic flow in compute node, 48–49
  traffic flow on controller node, 45, 47–48
Periodic processes, Swift, 11
Permissions, DevStack deployment failures, 23
Pets vs. cattle metaphor, cloud computing, 152
Physical hardware
  creating first OpenStack, 51
  deployment requirements, 40–41, 91–92
Physical network, initial setup, 52
Ping test
  logging in with SSH, 136–137
  on network namespaces, 218
Pixie Boots, Ironic mascot, 192
Planes of operations, networking, 35–36
Platform as a Service (PaaS), 2
Plug-ins
  Cinder, 11
  installing Neutron, 80, 84–85
  Neutron, 10, 77, 96–97
Preboot Execution Environment (PXE), Ironic, 12, 186–187
Private compute cloud
deployment scenario, 119
hybrid cloud deployments using, 228–229
launching instance with dashboard, 119–130
launching instance with OpenStack Client, 130–136
overview of, 117
requirements, 118
running service, 136–138
SDKs and OpenStack API, 138
selecting components, 119
uses, 117–118
Production machines, avoid running
  DevStack on, 21
Profiles
  base profile, 100–101
  Ceilometer, 144
  Cinder block storage, 156
  Glance imaging service, 101
  Horizon dashboard, 102
  Keystone identity, 102–103
  Neuron networking, 103–105
  Nova compute, 103
  Puppet, 99–100, 239
  RabbitMQ queuing, 105
Project Navigator, OpenStack website, 209
Project Team Lead (PTL), OpenStack projects, 243
Projects
  configuring Keystone, 64–65
definition of, 64–65
  DevStack deployment as administrator, 29
  other OpenStack, 13
Proprietary drivers/agent plug-ins, Neutron, 97
Proprietary offerings, cost of public cloud and, 227
Provider networks, 36
Provisioning
  bare metal, with Ironic, 12
  in Nova, 7
Proxy daemon, Swift, 170, 182
Proxy manifests, Swift, 175
Proxy logging proxy manifest, Swift, 175
PTL (Project Team Lead), OpenStack projects, 243
Public compute cloud
  architecture overview, 143–144
  benefits of, 226
  companies using containers, 196
  compute node setup, 145
  controller node setup, 144–145
  deploying whole cloud, 205
  evaluating cost for, 227
  evaluating location and, 227
  hybrid cloud deployment, 228–229
  overview of, 141
  requirements, 142–143
  scenario, 144
  uses, 141–142
  viewing metrics and alerts, 150
  viewing statistics with command line client, 147–149
  viewing statistics with dashboard, 146–147
Public network switch, physical machine setup, 41
Publishing data, Ceilometer, 11, 144
Puppet
  configuring Cinder block storage, 155–157
  configuring compute node, 110
  configuring controller node, 98–99
  configuring initial setup on servers, 94–95
  contents of composition model, 240–241
  controller node setup for Swift, 173–176
  as deployment mechanism, 93
  Hiera, 237–238
  Ironic installation, 188
  Keystone identity profile, 102
  as master or masterless, 237
  module management, 238–239
  modules as configuration management system, 16
  more information, 241
  node classification, 238
  overview of, 237
  packages, 240
  passwords in Hiera, 238
  revision control, 240
  roles and profiles, 239
  roles, profiles and modules, 99–100
  software lifecycle, 239
  storing modules in common.yaml file, 98
  troubleshooting, 115, 221–222
puppet apply command
  compute node setup for private cloud, 110
  controller node setup for Cinder, 155–157
  controller node setup for private cloud, 98
  controller node setup for public cloud, 144
  controller node setup for Swift, 173–176
  deploying whole cloud, 204–205
  Puppet node classification and, 238
  troubleshooting with, 221–222
Puppetfile, module management, 238
PXE (Preboot Execution Environment), Ironic, 12, 186–187
Python, 215, 243
Python-ironicclient, 12
Python-openstackclient tool, 56–57
Q
qcow2 image file, Glance, 10
qdhcp network namespace, 218
QEMU
  accessing OpenStack instances, 9
  adding image for private cloud, 122
  instances as VMs using, 4
  Nova drivers supporting, 7
  reference deployment, 231
  as supported hypervisor, 146
qrouter-* namespace ID, 47, 218
Quantum, as previous name of Neutron, 10

Queuing
OpenStack, 4
Rabbit MQ. See RabbitMQ Queuing service

R
RabbitMQ Queuing service
configuring controller node, 56–57
profile, 105
selecting for deployment, 96
troubleshooting as key service, 216–217

RAM
default flavors, 120
DevStack requirements, 22
reference deployment, 231–233
simple deployment requirements, 52
using Ironic, 189–191
 Ratelimit: proxy manifest, Swift, 175
Rc files, OpenStack, 251–252
Reading displayed errors, troubleshooting,
211–213
README.md file, troubleshooting via, 115
Recommendations, reference deployment,
233
Redundancy, Swift, 172
Reference deployment
initial networking setup, 53
installation, 231–233
overview of, 231
recommendations, 233
requirements, 92–93, 231
simple deployment, 52
testing Horizon dashboard in controller
node, 108–109

Registry
configuring Glance, 69–71
Glance profile, 101

Release cycle
OpenStack, 13–14
OpenStack contributions, 243–244
OpenStack specifications, 245
Puppet modules, 16
Ubuntu Cloud Archive, 15–16
Ubuntu Long Term Support, 14–15

Replicas, used by Rings in Swift, 171
Requesting help, 224

Requirements
Cinder block storage, 153–154
deploying whole cloud, 203–204
deployment foundations, 91–92
DevStack, 22–23
first OpenStack deployment, 51–52
networking deployment, 39–43
public compute cloud deployment, 142–143
reference deployment, 231
Swift object storage deployment, 169–172

Resource IDs, viewing statistics, 147–149
Resources, viewing statistics, 147–148
RESTful API, Ironic architecture, 187
RESTful HTTP API, Swift object storage, 11
Review system, code, 247–248
Revision control system
code contributions, 246–247
Puppet, 240
Ring-builder utility, Swift, 170
Rings
further information on, 171
in Swift, 170–171, 175–176

Roles
defined, 66
in DevStack deployment, 29
in Keystone, 64–65
modules vs., 99
Puppet, 99–100, 239

Root Disk column, default flavors, 120

Rules
iptable, 220–221
prelaunch access/security with dashboard,
123–126

S
SaaS (Software as a Service), 2
Sahara: data processing, 209
SaltStack, 235–236

Scaling
high availability and, 208
two nodes for deployment and, 92
VLANs vs. tunneling, 38

Scheduler
controller node setup for Cinder, 156
creating volume in dashboard, 158
Scheduler daemon, Nova, 73

Schema, adding to Keystone database, 60–61
**screen** -r (screen reattach) command, 30
Screen tool, DevStack, 30
Scripts-and-configs, git repository, 55
SDKs (Software Development Kits)
  automating Cinder, 166
  OpenStack APIs and, 138
Security
  deployment requirements, 92
  installing Neutron, 79
  managing instances, 86
  OpenStack Client, 132–134
  private compute cloud, 123–126
  public cloud, 228
  two nodes for deployment, 92
Security groups, 123–126
Segmented networking, tenant network, 37
Servers
  compute node. See Compute node
  controller node. See Controller node
  initial setup for deployment, 94
  logging in with SSH, 136–137
  Neutron, 77
Services
  Ceilometer, 12
  control network for, 36
  customizing DevStack, 33
  Glance, 67–69
  Heat, 13
  Keystone, 58–67
  Neutron, 77–78, 80, 85
  Nova, 7–8, 72–73, 84
  running, 136–138
  Swift, 11
  testing in controller node, 106
  troubleshooting key, 216–217
  troubleshooting reading displayed errors, 211
  troubleshooting via logs, 213
setup.sh script
  initial setup of Puppet, 94–95
  installing RabbitMQ Puppet module, 105
  placing Hiera common.yaml file via, 98
Sharing files, with object storage, 167–168
Snapshots, Cinder backups, 153
Software as a Service (SaaS), 2
Software Development Kits (SDKs)
  automating Cinder, 166
  OpenStack APIs and, 138
Software lifecycle, Puppet, 239
Software updates, initial operating system setup, 54
Source screen, DevStack deployment, 27
Spawning phase, launch with dashboard, 126
Specification process, OpenStack contributions, 245–246
SQLite database, configuring Keystone, 60
SSH (Secure Shell)
  add floating IP address, 128–130
  DevStack deployment as user, 25–28
  loading simple HTTP server, 137–138
  logging into instance, 87
  logging into server, 136–137
  pre-launch access and security, 123–126
  prelaunching instance with OSC, 132–134
  reference deployment via, 233
  using Cinder block storage, 162–165
Stable DevStack, 24, 31–32
Stack, Heat, 13
Stackforge Puppet modules, 241
stack.sh script, DevStack
  commented version of, 23
  customizing DevStack, 33
  defined, 21
Staticweb proxy manifest, Swift, 175
Statistics, viewing public cloud, 146–149
Storage. See Cinder block storage; Swift object storage
sudo command, first OpenStack deployment, 55–57
Summit, OpenStack, 3, 255
Support outlets, OpenStack, 245
Swap Disk column, default flavors, 120
Swift-account-server, 11
Swift-container-server, 11
Swift-object-server, 11
Swift object storage
  Ceph vs., 184
  clusters, 170
  connecting to, 171–172
  controller node setup, 173–176
  creating container in dashboard, 176–179
  creating container in OSC, 179–181
  as key component, 11–12
  objects and containers, 171
  overview of, 167
requirements, 169–172
Rings, 170–171
scenario, 173
uses, 167–168
using objects, 181–184
Swift-proxy-server, 11
Switches, 40–41, 51
Symmetric Key Encryption, 60
Sysfsutils, installing Nova, 83
System configuration, initial networking setup, 54–58
System Information
DevStack deployment, 29
Horizon dashboard, 205–206
OSC view of, 207
System Panel, DevStack deployment, 28–29

T
Tags, Puppet module, 238–239
Tap interface
definition of, 46
network namespaces and, 218
traffic flow in compute node, 48–49
traffic flow on controller node, 45
tcpdump command, 219–220
Technical Committee (TC), OpenStack, 13, 209, 243
Telemetry. See Ceilometer telemetry service
Tempauth proxy manifest, Swift, 175
Templates, Heat orchestration using, 12–13
Tempurl proxy manifest, Swift, 175
Tenant networks
deploying, 40–43
ip command and, 218
overview of, 36–38
Terminology, Linux networking, 46
Test user
adding image, 122–123, 131–132
configuring Glance, 71
configuring Keystone, 65–66
configuring OSC, 67
creating container/object in dashboard, 176–179
creating container/object in OSC, 179–181
managing instances, 85–87
testing compute node, 111–112
testing controller node, 105–109
Test volume, controller node setup in Cinder, 157
Testing
controller node, 105–109
infrastructure, 247–248
TEST_PASSWORD, OSC, 67
Testrc file, managing instances, 85–87
Text-based templates, Heat, 12–13
Throw-away commodity hardware, 152
Timestamps, 148
token issue command, Keystone, 66
Traceback, log messages, 215–216
Trademark disputes, OpenStack Foundation handling, 3
Traditional technology companies, building public clouds, 141–142
Traffic flow
compute node, 48–49
controller node, 44–48
deployment requirements, 40–43
overview of, 43–44
planes of operations, 35–36
tenant network options, 37–38
Training, DevStack as tool for, 21, 22
Transformers, Ceilometer, 144
Troubleshooting
configuration files, 221
deployments, 115
key services, 216–217
logs, 213–216
mitigating breakage, 223–224
networking, 217–221
overview of, 211
Puppet, 221–222
reading displayed errors, 211–213
requesting help, 224
Trove: Databases as a service, 209
TUN (network tunnel) interface, 46
Tunneling
network deployment, 40–43
segmented networking for tenants, 38

U
Ubuntu
Cloud Archive, 15–16
initial operating system setup, 54
initial setup for deployment, 94–95
Ubuntu (continued)
  Long Term Support release of, 14–15
  reference deployment, 231–233
  using object in web page, 183–184
Uca profile, 100
UCA (Ubuntu Cloud Archive)
  configuring Glance, 67
  configuring Keystone, 59
  configuring with uca profile, 100
  initial system configuration setup, 54
UML (via libvirt), as supported hypervisor, 146
Upgrades, block storage cloud, 152
URL, using objects, 182
User groups, 66, 255
User list command, 66
User Manual, OpenStack Infrastructure, 247
Users
  configuring Keystone, 9, 64–65
  dashboard for, 4–5
  defined, 66
  DevStack deployment as, 24–28, 29
UUID
  configuring Keystone, 102
  managing Ironic, 192
V
  VCPU column, default flavors, 120
  vdb device, Cinder block storage, 164
  Vendor-specific drivers/agent plug-ins,
    Neutron, 96–97
Vendors
  deployment mechanisms of, 236
  ecosystem of, 225–226
  finding help from, 256
  hybrid cloud, 228–229
  Ironic architecture and, 187–188
  lock-in, 151, 229–230
  offering deployment as value add, 94
  overview of, 225
  public cloud, 226–228
virbr0 bridge, networking setup, 53
virbr1 bridge, networking setup, 53
Virsch, networking setup, 53
Virsh, networking setup, 53
Virtual Extensible Local Area Networking,
  See VXLAN (Virtual Extensible Local Area Networking)
Virtual Local Area Networking (VLAN), 38
Virtual Machine Manager, 8–9, 231–233
Virtual Private Server (VPS), 142
VLAN (Virtual Local Area Networking), 38
vmlinuz (Linux kernel executable), Ironic, 189–191
VMs (virtual machines)
  deployment requirements, 42–43
  DevStack requirements, 22
  DevStack training on, 21
  initial network setup, 53
  OpenStack Instances as, 4
  simple deployment requirements, 51
  traffic flow on controller node, 45–48
VMware, 4, 7
VMware Sphere, 146
VNC access, with Virtual Machine Manager, 8–9
Volume, Cinder block storage
  cloning, 153
  creating/attaching with dashboard, 157–161
  creating/attaching with OSC, 161–163
  file permissions and, 165
  using, 162–165
Volume manager, Cinder, 154–155, 156
Volumes, Cinder, 205–206
Voting on changes, Continuous Integration testing, 247–248
VPN as a Service, Neutron, 96
VPS (Virtual Private Server), 142
VXLAN (Virtual Extensible Local Area Networking)
  encapsulated traffic in tunnel-driven network, 38–39
  network deployment, 40–43
  scaling with VLANs vs., 38
  traffic flow on controller node, 46
  traffic flow within compute node, 48–49
W
  Web-based help, 254
  Web forum, for help, 224
  Web hosting companies, 142, 167–168
  Web page, objects, 183–184
  Wireshark, using tcpdump, 219
WSGI (Web Server Gateway Interface)
  configuring Keystone, 59
Keystone identity profile, 102
Keystone using, 9
Swift using middleware by, 11

X
Xen, 7, 146
XFS loopback devices, Swift, 175–176

Y
YAML format files. See also Hiera common.
yaml file, 221

Z
Zones, Rings in Swift, 170
Zuul, 247