Join the discussion @ p2p.wrox.com

Wrox Programmer to Programmer™

Openstack

Cloud Application Development

John Belamaric, Scott Adkins, Jason E. Robinson, Vincent Giersch

CONTENTS

List of Illustrations

INTRODUCTION

OpenStack is a set of software packages that manage virtualized resources, including computing, networking, and storage. It enables you to create and destroy virtual machines, connect them together with private networks, provide network-based storage, and make them available to the rest of your network and the world. OpenStack provides consistent, uniform API services for all of this, hiding hypervisor and vendor specific details from the applications that are using the APIs. It also provides a user interface, built on top of the same APIs, that allows users to see and manage their virtual resources.

WHO THIS BOOK IS FOR

This book is for application developers that are interested in learning more about OpenStack and how it will transform the application design and development process. It is for someone who is new to the cloud environment, who wants a broad understanding of that environment, as well as a deep enough knowledge to make practical use of OpenStack.

WHAT THIS BOOK COVERS

This book will provide a broad understanding of cloud concepts and how they fit into the life of an application developer. It will drill in deeply to the OpenStack services that are most important to an application developer, and show you how these services will change not only how you deploy applications, but also how you design them. It will provide detailed information on each service, and provide examples of how each service may be used by an application developer.

HOW THIS BOOK IS STRUCTURED

This book was written in two parts. Part 1 provides an overview of OpenStack. The purpose of this part is to lay the groundwork, covering all of the OpenStack technologies and what is most important.

Part 2 takes the reader through developing and deploying applications with OpenStack. In this part you will build an example on top of OpenStack that drills down much deeper on the technologies, provides tips, and helps you learn about OpenStack through the lens of these same technologies.

Here is a list of the chapters:

- Part I: OpenStack Overview
    - Chapter 1: Introduction to OpenStack
    - Chapter 2: Understanding the OpenStack Ecosystem: Core Projects
    - Chapter 3: Understanding the OpenStack Ecosystem: Additional Projects
- Part II: Developing and Deploying Applications with OpenStack
    - Chapter 4: Application Development
    - Chapter 5: Improving on the Application
    - Chapter 6: Deploying the Application

WHAT YOU NEED TO USE THIS BOOK

You should understand the basics of application development - how applications are composed of multiple servers like web servers, application servers, and database servers. You do not need any cloud-specific knowledge,

though you should be aware of what virtualization and virtual machines are, and have a basic understanding of networks.

CONVENTIONS

To help you get the most from the text and keep track of what's happening, we've used a number of conventions throughout the book.

Examples that you can download and try out for yourself generally appear in a box like this:

EXAMPLE TITLE

This section gives a brief overview of the example.

Source

This section includes the source code.

```
Source code
Source code
Source code
```

Output

This section lists the output:

```
Example output
Example output
Example output
```

NOTE *Notes indicates notes, tips, hints, tricks, or and asides to the current discussion.*

As for styles in the text:

- We *highlight* new terms and important words when we introduce them.
- We show code within the text like so: `persistence.properties`.

SOURCE CODE

As you work through the examples in this book, you may choose either to type in all the code manually, or to use the source code files that accompany the book. All the source code used in this book is available for download at www.wrox.com. Specifically for this book, the code download is on the Download Code tab at:

www.wrox.com/go/openstackcloudappdev

and at:

https://github.com/johnbelamaric/openstack-appdev-book

You can also search for the book at www.wrox.com by ISBN (the ISBN for this book is 978-1-119-19431-6) to find the code. And a complete list of code downloads for all current Wrox books is available at www.wrox.com/dynamic/books/download.aspx.

> **Note** *Because many books have similar titles, you may find it easiest to search by ISBN; this book's ISBN is 978-1-119-19431-6.*

Once you download the code, just decompress it with your favorite compression tool. Alternately, you can go to the main Wrox code download page at www.wrox.com/dynamic/books/download.aspx to see the code available for this book and all other Wrox books.

ERRATA

We make every effort to ensure that there are no errors in the text or in the code. However, no one is perfect, and mistakes do occur. If you find an error in one of our books, like a spelling mistake or faulty piece of code, we would be very grateful for your feedback. By sending in errata, you may save another reader hours of frustration, and at the same time, you will be helping us provide even higher quality information.

To find the errata page for this book, go to

www.wrox.com/go/openstackcloudappdev

And click the Errata link. On this page you can view all errata that has been submitted for this book and posted by Wrox editors.

If you don't spot "your" error on the Book Errata page, go to www.wrox.com/contact/techsupport.shtml and complete the form there to send us the error you have found. We'll check the information and, if appropriate, post a message to the book's errata page and fix the problem in subsequent editions of the book.

P2P.WROX.COM

For author and peer discussion, join the P2P forums at http://p2p.wrox.com. The forums are a Web-based system for you to post messages relating to Wrox books and related technologies and interact with other readers and technology users. The forums offer a subscription feature to e-mail you topics of interest of your choosing when new posts are made to the forums. Wrox authors, editors, other industry experts, and your fellow readers are present on these forums.

At http://p2p.wrox.com, you will find a number of different forums that will help you, not only as you read this book, but also as you develop your own applications. To join the forums, just follow these steps:

1. Go to http://p2p.wrox.com and click the Register link.
2. Read the terms of use and click Agree.
3. Complete the required information to join, as well as any optional information you wish to provide, and click Submit.
4. You will receive an e-mail with information describing how to verify your account and complete the joining process.

> **NOTE** *You can read messages in the forums without joining P2P, but in order to post your own messages, you must join.*

Once you join, you can post new messages and respond to messages other users post. You can read messages at any time on the Web. If you would like to have new messages from a particular forum e-mailed to you, click the Subscribe to this Forum icon by the forum name in the forum listing.

For more information about how to use the Wrox P2P, be sure to read the P2P FAQs for answers to questions about how the forum software works, as well as many common questions specific to P2P and Wrox books. To read the FAQs, click the FAQ link on any P2P page.

PART I
OpenStack Overview

1
Introducing OpenStack

WHAT'S IN THIS CHAPTER?

- Models of cloud computing
- Relevance of cloud computing to application developers
- Why OpenStack is a good cloud platform choice
- How OpenStack is put together

WHAT IS CLOUD COMPUTING?

There is so much hype around cloud computing that it is often difficult to get a clear sense of what anyone means by those words. Is it just virtualization? Is it Software-as-a-Service (SaaS), such as Microsoft's Office 365 and Salesforce.com? Or is it the ability to get a virtual machine instantly from Amazon Web Services (AWS) or Azure? And what about online storage such as Dropbox?

Types of Cloud Computing

The reality is that cloud computing refers to all of these things just described and more. The National Institute of Standards and Technology (NIST) has come up with an "official" definition based upon five key components: on-demand self-service, broad network access, pooled resources, elasticity, and metered service. In general, these characteristics may be provided in several different models. These models help sort out the confusion and hype. In fact, these can be thought of as layers in a stack, with each layer being built on top of the previous one (see Figure 1.1).

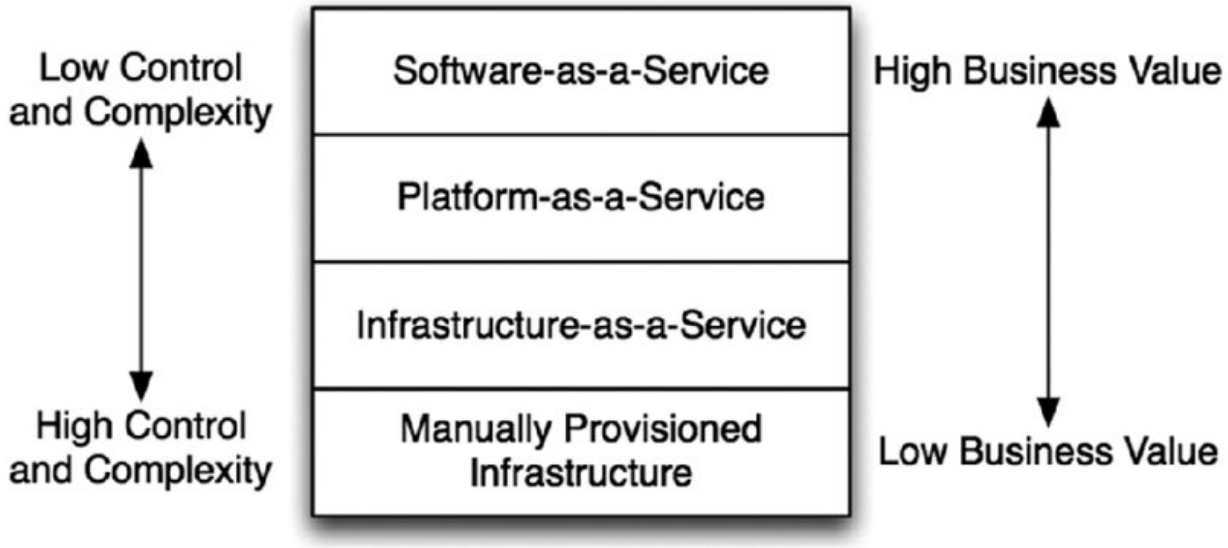

Figure 1.1

In Figure 1.1, “Manually Provisioned Infrastructure” represents the traditional method of building your information technology infrastructure—this is not cloud computing. In this environment, physical machines are racked, connected, and configured on a one-by-one basis. This provides complete control, but requires substantial time and effort to build out, or to change when necessary. Of course, all clouds need to run on physical gear at some point, so this provides the basic foundation for everything else. One of the keys to making cloud computing successful, however, is to move the complexity out of this layer and up higher in the stack.

Infrastructure-as-a-Service (*IaaS*) is the most basic layer in the cloud computing stack. This is OpenStack’s primary focus, as well as the primary focus for AWS. It enables automated or self-service provisioning of compute, networking, and storage. Typically, these resources are provided as Virtual Machines (VMs), but you could also use it to spin up bare metal servers (i.e. physical hosts). This is known as “Metal-as-a-Service,” and OpenStack provides a project for managing this service as well. Alternatively, you

can also spin up containers rather than VMs or bare metal servers. The essential point is that it enables the provisioning of compute instances, with (optionally) attached networking and storage.

Platform-as-a-Service (*PaaS*) builds on top of IaaS to enable the provisioning of applications, rather than simply the infrastructure that might be used to run the application. So, a PaaS provides core common services needed by applications, along with the machinery to configure and deploy applications to use those services. A PaaS typically will provide a complete application stack (web server, application server, database server, etc.) into which you can easily deploy your application. Heroku (`https://www.heroku.com`) is an example of a popular PaaS for applications built with a variety of standard frameworks, such as Ruby-on-Rails. With Heroku you can deploy your application to the Internet with a simple `git push`. As the application author and deployer, you don't need to worry about configuring and deploying the different tiers, or even worry about how to scale them. If you follow the Heroku conventions, everything is handled by the PaaS.

Software-as-a-Service (*SaaS*) is the layer farthest from the underlying physical infrastructure. It may be built on IaaS or a PaaS, but need not be—the point is the user never really knows. This is the simplest form of cloud computing from the point of view of the user because they have no insight into the actual mechanics or systems behind the service. It's just a service they use. Often this is provided in the form of a website, such as Salesforce.com. But you can also get lower-level services such as Database-as-a-Service, where you simply request via an API (or website) for a database with certain parameters, and are given an IP and port to connect to. As a user of the service, you don't need to worry about how to scale that service—though you will need to pay more as your use of the service increases.

Put succinctly, IaaS provides the tools to “build” your systems from the ground up. PaaS allows you to “deploy” your applications, without needing to worry about the underlying infrastructure. SaaS allows you to “buy” your applications—you do not even need to deploy or manage them at all. This is a steady progression of decreasing control and complexity, while increasing direct business value.

While these are general models for cloud computing, in reality the distinctions between them are not always crystal clear. The relationship of SaaS to PaaS in particular can be complicated. A specific, complex Software-as-a-Service may use PaaS or even other more granular Software-as-a-Service. Even a PaaS may assemble lower-level pieces as a collection of software services. For example, most services will require an identity management (authentication, authorization, and accounting) service. This identity service is one of the key features a PaaS provides to applications. However, there is no reason that service cannot be, in turn, provided by some external SaaS! In this case, a key function of the PaaS is provided via a low-level SaaS.

Cloud Infrastructure Deployment Models

In addition to the functionality provided by a cloud, there are several different deployment models for clouds. *Public clouds* are the ones familiar to most developers. These cloud services are made available to the general public for a fee. The fee is generally on a usage basis, enabling organizations to utilize their operating budgets rather than their capital budgets. The customers have no need to maintain or operate the hardware or cloud infrastructure, leaving that responsibility completely to the cloud operator.

Amazon Web Services (AWS) is currently the largest public cloud and dominates the industry. Microsoft and VMware

also operate public clouds, and a number of service providers do as well. Rackspace, in particular, provides an OpenStack-based public cloud, and is one of the primary contributors to the OpenStack project.

Private clouds, on the other hand, are internal to an organization. They represent the evolution of the traditional corporate data center. Only internal customers within the enterprise, and perhaps close partners, use private clouds. The corporate IT department or a contractor will purchase, setup, and maintain the hardware and software for the cloud. The cloud infrastructure may use chargeback to distribute costs among the business units, but the cloud itself is still dedicated to the single enterprise.

Organizations may operate private clouds for a number of reasons. The cost of a private cloud, if well run, may be less than utilizing the public clouds. Additionally, many industries have security or regulatory reasons that disallow the use of a public cloud for many workloads. These organizations are required to run those workloads in a private cloud. See Figure 1.2 for a look at the structure of public, private, and hybrid clouds.

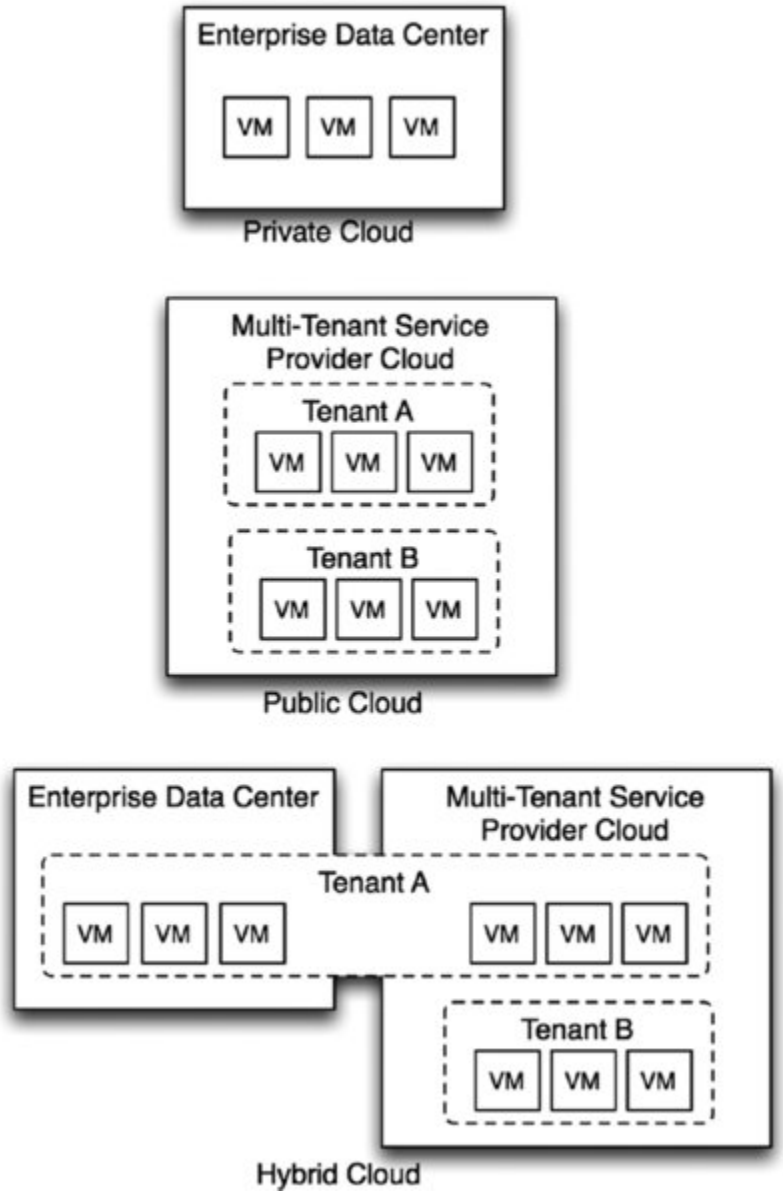

Figure 1.2

Hybrid clouds combine both private and public clouds. The goal with hybrid clouds is to keep general operating costs low by using the private cloud for most of the workloads, but to enable spillover into the public cloud when necessary. The spillover could happen due to capacity reasons—perhaps during the holiday season your private cloud doesn't have enough capacity—or for disaster

recovery. This model avoids the capacity constraints of a private cloud while still keeping costs under control.

WHY SHOULD I CARE?

As an application developer or architect, you may wonder—why does all of this matter to me? All of this discussion covered so far focuses on the reason a business may want to move to the cloud. But why should that affect the application developer? The answer lies in a couple of different areas: the effect on the development process, and the effect on your application architecture.

Cloud services enable much more efficient processes for managing development, test, and production environments. These updated processes and methods represent the "DevOps" mentality—applying standard software development practices, such as source code version control, to the operational aspects of the application. This means capturing all of the configuration and deployment information in scripts and templates, and controlling their changes just as you would application code.

Scripts and templates can be built that produce a complete application environment. These can be used to automatically deploy not only the application, but also infrastructure required for the application, including virtual machines, networking, firewalls, load balancers, domain name services—you name it, and someone is working on making it available "as-a-Service." By automating the creation and destruction of these environments, you can ensure consistency between development, test, and production environments. For complex applications with many different services running on different machines, this can be a dramatic time saver.

OpenStack, and “as-a-Service” thinking in particular, will also end up changing the software and deployment architectures of your application. By relegating the common and routine functions to the cloud infrastructure, you free your time and thought to focus on the most important thing—your application’s functionality. For example, a traditional application that allows large file uploads will need to designate temporary and permanent storage locations for those files, and manage the storage resources to ensure that the disk doesn’t fill. The system administrator or deployer will need to devise a strategy to backup that data or replicate it to other data centers. But with the right cloud platform, you can simply delegate that function to the infrastructure, and get all of the benefits without devoting special effort.

Designing your application to work with the cloud services also dramatically simplifies scaling the application. The scalability of the individual services becomes the responsibility of the cloud operator, not the application developer or administrator. As long as the application makes effective use of those services, it will scale as needed with little to no work from the developers themselves.

Being able to utilize “as-a-Service” functions is one way your design will shift. Another is to plan for horizontal scaling rather than vertical scaling. That is, scaling by adding more machines (horizontally) rather than creating bigger machines (vertically). With most applications today, it is easiest to scale by getting a bigger, faster machine. This locks you into planning for peak capacity of each application individually. For each application you need to provision the largest machine you may need at peak load. But with applications built for the cloud, you instead scale by adding more machines. These machines can be smaller, and with cloud automation, can be added, removed, or

resized as needed. This ability to scale up and down as needed is called *elastic scaling*, and is one of the key features of cloud computing.

A frequently used analogy is that traditional servers are like “pets,” while cloud-based servers are “cattle.” This describes a necessary shift in mentality for a traditional application architect. The idea is that a pet is unique and special, with its own unique name. A lot of resources are spent to raise and nurture one, and if it is sick, it will be nursed back to health. Cattle, on the other hand, are not treated specially or carefully raised. They are treated en masse—they are given numbers, not names—and a sick one is culled to prevent any spread of disease through the herd.

The implication here is that cloud-based servers should be disposable and easily re-deployed, and not require careful hand configuration. That way, if there is a problem with one, you do not spend time trying to figure it out and fix it —you simply replace it with a new one. This is the logical extension of the ability to scale elastically. Why take the time to figure out what’s wrong with a machine when it’s behaving badly? Just pull it out of the application and replace it with a new one while you debug the problem (not to fix that machine, but to prevent the issue in the future).

What Is OpenStack?

OpenStack bills itself as a “cloud operating system.” Fundamentally, it solves the IaaS problem. It provides the ability to abstract the physical compute, storage, and networking resources into pools. Those resources can then be divvied up among users in a secure way. Users only need to pay for what they are using, rather than having to provision their applications for peak load.

OpenStack is a collection of open source software projects, backed by a non-profit organization, the OpenStack

Foundation. These projects work together to provide a consistent API layer, while enabling the actual services to be provided by a variety of different vendor or open source implementations. At the core, these services include the functionality you need to run a cloud, that is, the ability to spin up virtual machines, the ability to allocate, manage, and share storage among those machines, and the ability enable these machines to communicate with one another securely over the network.

KEEPING TRACK OF RELEASES

OpenStack has official releases every six months. In order to make it easier to keep track of all these releases, they are given names in alphabetical order. Below is the name of each release, and its release date, through the Liberty release.

- Austin: October 2010
- Bexar: February 2011
- Cactus: April 2011
- Diablo: September 2011
- Essex: April 2012
- Folsom: September 2012
- Grizzly: April 2013
- Havana: October 2013
- Icehouse: April 2014
- Juno: October 2014
- Kilo: April 2015
- Liberty: October 2015

In addition to the release name, each release is identified by the year and release during that year—<year>.<release>.<patch>. For example, Kilo is also known as 2015.1, as the first release in 2015. Patch releases for Kilo are 2015.1.1, 2015.1.2, etc. The second major release of 2015 is Liberty, which is also known as 2015.2.

All of these services are accessible via RESTful APIs, as well as command-line interfaces and a web-based user interface called Horizon. Horizon is convenient for setting up things on an ad-hoc basis, but doesn't offer the full capabilities of the APIs—and of course the APIs and CLI tools can be easily scripted (see Figure 1.3).

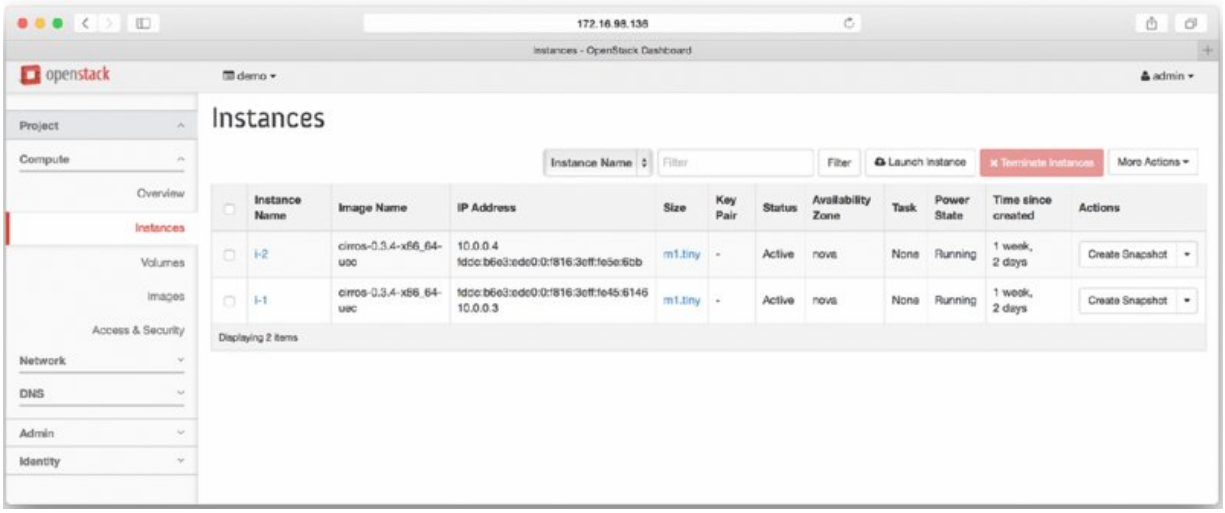

Figure 1.3

The next table shows the major services provided by OpenStack, along with their names. OpenStack community members will usually refer to each service by its name, so it's helpful to see them all in one place and get a handle on what each one does. In fact, there are many more services, but these are the most common ones you will find.

Name	Service	Description
Horizon	Dashboard	A graphical user interface for managing your cloud
Keystone	Identity	Authentication, authorization, and OpenStack service information
Nova	Compute	Spin up, manage, and terminate virtual machines
Cinder	Block Storage	Disk volumes (that outlive an instance) and snapshots of instances
Swift	Object Storage	Shared, replicated, redundant storage for images, files, and other media accessible via Hypertext Transfer Protocol (HTTP)
Neutron	Network	Provide secure tenant networking
Glance	Image	Provide storage and access to VM images and snapshots
Heat	Orchestration	Spin up groups of machines, networks, and other resources via templates
Designate	DNS	Create domains and records in the DNS infrastructure
Ceilometer	Telemetry	Monitor resources usage across the cloud
Trove	Database	Provide access to private tenant databases
Ironic	Bare Metal	Spin up instances on physical hardware