OpenStack Administration
with Ansible

Most organizations are seeking methods to improve business agility because they have realized that just having a cloud is not enough. Being able to improve application deployments, reduce infrastructure downtime, and eliminate daily manual tasks can only be accomplished through some sort of automation.

Packed with real-world OpenStack administrative tasks, this book will walk you through working examples and explain how these tasks can be automated using one of the most popular open source automation tools—Ansible.

We will start with a brief overview of OpenStack and Ansible and highlight some best practices. Each chapter will provide an introduction to handling various cloud operator administration tasks such as creating multiple users/tenants, setting up multi-tenant isolation, customizing your cloud quotas, taking instance snapshots, adding OpenStack features, and running cloud health checks; there will be a step-by-step tutorial on how to automate these tasks with Ansible.

Who this book is written for

If you are an OpenStack-based cloud operator and/or infrastructure administrator and are interested in automating administrative functions, then this book is exactly what you are looking for. Having a functioning OpenStack environment is helpful, but most certainly not necessary.

What you will learn from this book

- Efficiently execute OpenStack administrative tasks
- Familiarize yourself with how Ansible works and assess the defined best practices
- Create Ansible playbooks and roles
- Automate tasks to customize your OpenStack cloud
- Review OpenStack automation considerations when automating administrative tasks
- Examine and automate advanced OpenStack tasks and designated use cases
- Get a high-level overview of OpenStack and the current production-ready projects
- Dive deep into OpenStack CLI tools and find out how to use them

In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'An Introduction to OpenStack'
- A synopsis of the book’s content
- More information on OpenStack Administration with Ansible
Walter Bentley is a private cloud solutions architect with Rackspace. He has a diverse background in production systems administration and solutions architecture. He has over 15 years of experience across numerous industries, such as online marketing, finance, insurance, aviation, food, education, and now in technology product space. In the past, he has typically been the requestor, consumer, and advisor to companies in the use of technologies such as OpenStack. He is now a promoter of OpenStack technology and a cloud educator.

In Walter's current role, he is focused on helping customers build, design, and deploy private clouds built on OpenStack. His role also involves professional services, engagements around operating the built OpenStack clouds, and DevOps engagements creating playbooks/roles with Ansible.

Recently, Walter was given the honor to be present at the OpenStack Vancouver and Tokyo Summits. As well as this, he was given the great opportunity to present at AnsibleFest events in London and New York. Some of his other experiences involve performing webinars, blogging about OpenStack, and speaking at various technology conferences. Another wonderful experience is Walter being allowed to take part in the technical review of the newly launched OpenStack Cloud Computing Cookbook, Third Edition, Packt Publishing.
As OpenStack is now being considered to be more of a mainstream cloud platform, the challenge of operating it after it is built has become prevalent. While all cloud tasks can be executed via the API or CLI tool on a one-by-one basis, it will not be the best method to handle larger cloud deployments. The need for a more automated approach to administer OpenStack is now clear. Most organizations are seeking methods to improve business agility and have realized that just having a cloud is not enough. Being able to improve application deployments, reduce infrastructure downtime, and eliminate daily manual tasks, can only be accomplished through some sort of automation. OpenStack and Ansible will help any organization close that gap. With the IaaS capabilities that OpenStack has to offer coupled with Ansible, an easy to use configuration management tool assures a more complete cloud implementation.

Whether you are new to OpenStack or a seasoned cloud administrator, this book will aid you in managing your OpenStack cloud once it is all set up. Packed with real world OpenStack administrative tasks, we will first have a look at the working examples natively and then transition to walking through instructions on how to automate these tasks using one of the most popular open source automation tools, Ansible.

Ansible has become a market leader in the open source orchestration and automation space. With it being built using Python, similar to OpenStack, it makes for an easy marriage. The ability to leverage existing and/or new OpenStack modules will allow you to quickly move along your playbook creation.
Preface

We will start with a brief overview of OpenStack and Ansible, highlighting some of the best practices. Next, the beginning of each of the following chapters will allow you to become more familiar with handling cloud operator administration tasks, such as creating multiple users/tenants, setting up multi-tenant isolation, customizing your cloud's quotas, taking instance snapshots, deploying additional OpenStack features, running cloud health checks, and so on. Finally, each chapter will conclude with a step-by-step tutorial on how to automate these tasks with Ansible. As an added bonus, the fully functional Ansible code will be published on GitHub for your reference while reviewing the chapter and/or for later review.

Consider this book to be a 2-for-1 learning experience, a deep OpenStack-based cloud administrative knowledge and familiarity with how Ansible works. As a reader, you will be encouraged to put hands to the keyboard and give the tasks a try.

What this book covers

Chapter 1, An Introduction to OpenStack, is a high-level overview of OpenStack and the projects that make up this cloud platform. This introduction will set the level of the reader on the OpenStack components, concepts, and verbiage.

Chapter 2, An Introduction to Ansible, is a detailed review of Ansible, its features, and the best practices to use, in order to set a solid starting foundation. It reviews why leveraging Ansible to automate OpenStack tasks is the easiest option.

Chapter 3, Creating Multiple Users/Tenants, guides the reader through the process of creating users and tenants within OpenStack manually and the creation considerations in order to automating such a process using Ansible.

Chapter 4, Customizing Your Cloud's Quotas, makes you understand what quotas are and how they are used to restrict your cloud's resources. It shows the reader how to create quotas manually in OpenStack. After that, it walks the reader through how to automate this process with Ansible to take into account when handle tasks for multiple tenants at the same time.

Chapter 5, Snapshot Your Cloud, shows you how to create snapshots of your cloud instances manually within OpenStack and how to automate this process using Ansible. It also lets you explore the power of being able to snapshot all instances within a tenant in one shot.

Chapter 6, Migrating Instances, introduces the concept of migrating select instances across compute nodes in the traditional OpenStack method. It then demonstrates the required steps to automate this task while grouping instances together. It also shows the additional options that Ansible can offer in handling a task of this matter.
Chapter 7, Setting up Isolated Tenants, walks the reader through the concept of multi-tenant isolation functionality within OpenStack. It tells us about the building blocks of accomplishing this successfully and next demonstrate how to automate all the steps required to set it up. It emphasizes how automation of this level prevents incorrect configurations, missed steps, and provides an easily repeatable process.

Chapter 8, Deploying OpenStack Features, presents the process to add additional OpenStack features to your cloud with Ansible. We will use the example of multi-hypervisor support built into OpenStack, as an example to demonstrate how Ansible code can be created to deploy new features.

Chapter 9, Inventory Your Cloud, explores how the reader can dynamically inventory all the OpenStack cloud user resources with one Ansible playbook. It walks them through the necessary metrics to gather information and how that can be stored for later reference. This is a very powerful tool to have as a cloud administrator/operator.

Chapter 10, Health Check Your Cloud, demonstrates some useful tips and tricks on how to check the health of your cloud manually and leveraging Ansible to trigger hourly and/or daily reports.
An Introduction to OpenStack

This chapter will serve as an overview of OpenStack and all the projects that make up this cloud platform. It is very important to lay a clear foundation of OpenStack, in order to describe the OpenStack components, concepts, and verbiage. Once the overview is covered, we will transition into discussing the core features and benefits of OpenStack. Lastly, the chapter will finish up with two working examples of how you can consume the OpenStack services via the application program interface (API) and command-line interface (CLI). In this chapter, we will cover the following points:

- OpenStack overview
- Reviewing the OpenStack services
- OpenStack supporting components
- The features and benefits
- Working examples: listing the services and endpoints

OpenStack overview

In the simplest definition possible, OpenStack can be described as an open source cloud operating platform that can be used to control large pools of compute, storage, and networking resources throughout a data center. It is all managed through a single interface controlled by either an API, CLI, and/or web graphical user interface (GUI) dashboard. The power that OpenStack offers the administrators is the ability to control all these resources, while still empowering the cloud consumers to provision the same resources through other self-service models. OpenStack was built in a modular fashion; the platform is made up of numerous components. Some of the components are considered as core services and are required in order to have a functional cloud, while the other services are optional and are only required unless they fit into your personal use case.
The OpenStack foundation

Back in early 2010 Rackspace, at that time was just a technology hosting company focused on providing service and support thru an offering called "Fanatical Support", decided to create an open source cloud platform. After two years of managing the OpenStack project with its 25 initial partners, it was decided to transfer the intellectual property and governance of OpenStack to a non-profit member run foundation that is known as the OpenStack Foundation.

The OpenStack Foundation is made up of voluntary members governed by an appointed board of directors and project based tech committees. The collaboration occurs around a six-month, time-based major code release cycle. The release names are run in alphabetical order and refer to the region encompassing the location where the OpenStack design summit will be held. Each release incorporates something called OpenStack Design Summit, which is meant to build collaboration among OpenStack operators/consumers; thus, allowing the project developers to have live working sessions and also agree on release items.

As an OpenStack Foundation member, you can take an active role in helping develop any of the OpenStack projects. There is no other cloud platform that allows such participation.

To learn more about the OpenStack Foundation, you can go to the [www.openstack.org](http://www.openstack.org) website.

Reviewing the OpenStack services

Getting to the heart of what makes up OpenStack as a project would be to review the services that make up this cloud ecosystem. One thing to be kept in mind in reference to the OpenStack services is that each service will have an official name and a code name associated with it. The use of the code name has become very popular among the community and most documentation will refer to the services in that manner. Thus, becoming familiar with the code names is important and will ease the adoption process.

The other thing to be kept in mind is that each service is developed as an API driven REST web service. All the actions are executed via the API, enabling ultimate consumption flexibility. Behind the scenes, API calls are executed and interpreted even while using the CLI or web-based GUI.

As of the Kilo release, the OpenStack project consists of twelve services/programs. The services will be reviewed in the order of their release to show an overall service timeline. That timeline will show the natural progression of the OpenStack project overall, also showing how it is now surely Enterprise ready.
OpenStack Compute (codename Nova)

It was integrated in the release Austin and was one of the first and is still the most important service part of the OpenStack platform. Nova is the component that provides the bridge to the underlying hypervisor, which is used to manage the computing resources.

One common misunderstanding is that Nova is a hypervisor in itself, which is simply not true. Nova is a hypervisor manager of sorts and is capable of supporting many different types of hypervisors.

Nova will be responsible for scheduling instance creation, sizing options for the instance, managing the instance location, and, as mentioned earlier, keeping track of the hypervisors available to the cloud environment. It also handles the functionality of segregating your cloud into isolation groups called cells, regions, and availability zones.

OpenStack Object Storage (codename Swift)

It was also integrated in the Austin release and this service is one of the first services that were a part of the OpenStack platform. Swift is the component that provides object storage as a service to your OpenStack cloud. It is capable of storing petabytes of data; in turn, adding highly available, distributed, and eventually consistent object/blob store. Object storage is intended to be a cheap and cost effective storage solution for static data such as images, backups, archives, and static content. The objects can then be streamed over standard web protocols (HTTP/HTTPS), to or from the object server to the end user initiating the web request. The other key feature of Swift is that all data is automatically made available as it is replicated across the cluster. The storage cluster is meant to scale horizontally, by simply adding new servers.

OpenStack Image Service (codename Glance)

It was integrated in the Bextar release and this service was introduced during the second OpenStack release and is responsible for managing/registering/maintaining server images for your OpenStack cloud. It includes the capability to upload or export OpenStack compatible images and store instance snapshots, and is used as a template/back up for later use. Glance can store the images on a variety of locations either locally and/or on distributed storage, such as object storage. Most Linux kernel distributions already have OpenStack compatible images available for download. You can also create your own server images from existing servers. There exists support for multiple image formats including: RAW, VHD, QCOW2, VMDK, OVF, and VDI.
An Introduction to OpenStack

OpenStack Identity (codename Keystone)
It was integrated in the Essex release and this service was introduced during the fifth OpenStack release. Keystone is the authentication and authorization component built into your OpenStack cloud. Its key role is to handle creation, registry, and management of users, tenants, and all the other OpenStack services. Keystone is the first component to be installed when starting an OpenStack cloud. It has the capability to connect to external directory services, such as LDAP. Another key feature of Keystone is that it is built based on role-based access controls (RBAC), thus allowing cloud operators to provide distinct role-based access to individual service features to the cloud consumers.

OpenStack Dashboard (codename Horizon)
It was also integrated in the Essex release and this service is the second service to be introduced in the fifth OpenStack release. Horizon provides cloud operators and consumers with a web based GUI to control their compute, storage, and network resources. The OpenStack dashboard runs on top of Apache and the Django REST framework; thus, making it very easy to integrate into and extend to meet your personal use case. On the backend, Horizon also uses the native OpenStack APIs. The basic principle behind Horizon was to be able to provide cloud operators with a quick and overall view of the state of their cloud and cloud consumers a self-service provisioning portal to the cloud resources designated to them.

Note that horizon can handle approximately 70% of the overall available OpenStack functionality. To leverage 100% of the OpenStack functionality, you will need to utilize the API's directly and/or utilize the CLI for each service.

OpenStack Networking (codename Neutron)
It was integrated in the Folsom release and this service is probably the second most powerful component within your OpenStack cloud next to Nova.

OpenStack Networking is intended to provide a pluggable, scalable, and API-driven system to manage networks and IP addresses.
This quote was taken directly from the OpenStack Networking documentation, as it best reflects exactly the purpose behind Neutron. Neutron is responsible to create your virtual networks with your OpenStack cloud. This will entail the creation of virtual networks, routers, subnets, firewalls, load balancers, and similar network functions. Neutron was developed with an extension framework, which allows the integration of additional network components (physical network device control) and models (flat, layer-2 and/or layer-3 networks). The various vendor specific plugins and adapters have been created to work in line with Neutron. This service adds to the self-service aspect of OpenStack; thus, removing the network aspect from being a roadblock to consume your cloud.

With Neutron being one of the most advanced and powerful components within OpenStack, a whole book was dedicated to it.

**OpenStack Block Storage (codename Cinder)**

It was also integrated in the *Folsom* release, Cinder is the component that provides block storage as a service to your OpenStack cloud by leveraging local disks or attached storage devices. This translates into persistent block level storage volumes available to your instances. Cinder is responsible for managing and maintaining the block volumes created, attaching/detaching those volumes, and also backup creation of that volume. One of the highly notable features of Cinder is its ability to connect to multiple types of backend shared storage platforms at the same time. This capability spectrum spans all the way down to being able to leverage simple Linux server storage as well. As an added bonus, Quality of Service (QoS) roles can be applied to different types of backends; thus, extending the ability to use the block storage devices to meet various application requirements.

**OpenStack Orchestration (codename Heat)**

It was integrated in the *Havana* release and this was one of the two services to be introduced in the eighth OpenStack release. Heat provides the orchestration capability over your OpenStack cloud resource. It is described as a main-line project part of the OpenStack orchestration program. This infers the additional automation functionality that is in the pipeline for OpenStack.

The built-in orchestration engine is used to automate provisioning of applications and its components, known as stack. A stack might include instances, networks, subnets, routers, ports, router interfaces, security groups, security group rules, auto scaling rules, and so on. Heat utilizes templates to define a stack and is written in a standard markup format, YAML. You will notice these templates being referred to as Heat Orchestration Template (HOT) templates.
OpenStack Telemetry (codename Ceilometer)
It was also integrated in the Havana release and this is the second of the two services introduced in the eighth OpenStack release. Ceilometer collects the cloud usage and performance statistics together into one centralized data store. This capability becomes a key component to a cloud operator as it gives clear metrics into the overall cloud, which can be used to make scaling decisions.

You have the option of choosing the data store backend to Ceilometer. Such options include: MongoDB, MySQL, PostgreSQL, HBase, and DB2.

OpenStack Database (codename Trove)
It was integrated in the Icehouse release, Trove is the component that provides database as a service to your OpenStack cloud. This capability includes providing scalable and reliable relational and non-relational database engines. The goal behind this service was to remove the burden of needing to understand database installation and administration. With Trove, the cloud consumers can provision database instances just by leveraging the services API. Trove supports multiple single-tenant databases within a Nova instance.

The following data store types are currently supported by OpenStack: MySQL, MongoDB, Cassandra, Redis and Couch.

OpenStack Data Processing (codename Sahara)
It was integrated in the Juno release, Sahara is the component that provides data processing as a service to your OpenStack cloud. This capability includes the ability to provision an application cluster tuned to handle large amounts of analytical data. The data store options available are Hadoop and/or Spark. This service will also aid the cloud consumer in being able to abstract the complication of installing and maintaining this type of cluster.
OpenStack Bare Metal Provisioning
(codename Ironic)

It was integrated in the Kilo release and this service has been one of the most anxiously awaited components of the OpenStack project. Ironic provides the capability to provision physical bare metal servers from within your OpenStack cloud. It is commonly known as a bare metal hypervisor API and leverages a set of plugins to enable interaction with the bare metal servers. It is the newest service to be introduced to the OpenStack family and is still under development.

OpenStack supporting components

It is similar to any traditional application, there are dependent core components that are pivotal to the functionality but not necessarily the application itself. In the case of the base OpenStack architecture, there are two core components that will be considered as the core or backbone of the cloud. The OpenStack functionality requires access to an SQL-based backend database service and an Advanced Message Queuing Protocol (AMQP) software platform. Just as with most things OpenStack related there are the most commonly used/recommended choices adopted by the OpenStack community. From a database perspective, the common choice will be MySQL and the default AMQP package is RabbitMQ. These two dependencies must be installed, configured, and functional before you can start an OpenStack deployment.

There are additional optional software packages that can also be used to provide further stability as a part of your cloud design. The information about this management software and OpenStack architecture details can be found at the following link:

http://docs.openstack.org/arch-design/generalpurpose-architecture.html

Features and benefits

The power of OpenStack has been tested true by numerous Enterprise grade organizations; thus, gaining the focus of many of the leading IT companies. As this adoption increases, we will surely see an increase in consumption and additional improved features/functionality. For now, let's review some of OpenStack's features and benefits.
An Introduction to OpenStack

**Fully distributed architecture**

Every service within the OpenStack platform can be grouped together and/or separated to meet your personal use case. Also, as mentioned earlier, only the core services (Keystone, Nova, and Glance) are required to have a functioning cloud; all other components can be optional. This level of flexibility is something every administrator seeks for an *Infrastructure as a Service (IaaS)* platform.

**Uses commodity hardware**

OpenStack was uniquely designed to accommodate almost any type of hardware. The underlying OS is the only dependency of OpenStack. As long as OpenStack supports the underlying OS and that OS is supported on the particular hardware, you are all set to go! There is no requirement to purchase OEM hardware or even hardware with specific specs. This gives yet another level of deployment flexibility for administrators. Good example of this can be giving your old hardware sitting around in your data center new life within an OpenStack cloud.

**Scale horizontally or vertically**

The ability to easily scale your cloud is another key feature to OpenStack. Adding additional compute nodes is as simple as installing the necessary OpenStack services on the new server. The same process is used to expand the OpenStack services control plane as well. Just as with other platforms, you can add more computing resources to any node as an alternate approach to scaling up.

**Meets high availability requirements**

OpenStack is able to certify meeting high availability (99.9%) requirements for its own infrastructure services, if implemented via the documented best practices.

**Compute isolation and multi-DC support**

Another key feature of OpenStack is the support to handle compute hypervisor isolation and the ability to support multiple OpenStack regions across data centers. Compute isolation includes the ability to separate multiple pools of hypervisors distinguished by hypervisor type, hardware similarity, and/or vCPU ratio.

The ability to support multiple OpenStack regions, which is a complete installation of functioning OpenStack clouds with shared services, such as Keystone and Horizon, across data centers is a key function to maintain highly available infrastructure. This model eases overall cloud administration; thus, allowing a single pane of glass to manage multiple clouds.
Robust role-based access control

All the OpenStack services allow RBAC while assigning authorization to cloud consumers. This gives cloud operators the ability to decide the specific functions allowed by the cloud consumers. An appropriate example will be to grant a cloud user the ability to create instances but denying the ability to upload new server images or adjust instance-sizing options.

Working examples – list the services and endpoints

So we have covered what OpenStack is, the services that make up OpenStack, and some of the key features of OpenStack. It is only appropriate to show a working example of the OpenStack functionality and the methods available to manage/administer your OpenStack cloud.

To re-emphasize, OpenStack management, administration, and consumption of services can be accomplished either by an API, CLI, and/or web dashboard. When considering some level of automation, the last option of the web dashboard is normally not involved. So, for the remainder of this book, we will solely focus on using the OpenStack APIs and CLIs.

Listing the OpenStack services

Now, let’s take a look at how you can use either the OpenStack API or CLI to check for the available services and endpoints active within your cloud. We will first start with listing the available services.

Via API

The first step in using the OpenStack services is authentication against Keystone. You must always first authenticate (tell the API who you are) and then receive authorization (API ingests your username and determines what predefined task(s) you can execute) based on what your user is allowed to do. That complete process ends with providing you with an authentication token.

The Keystone can provide two different types of token formats: UUID or PKI. A typical UUID token looks similar to 53f7f6ef0cc344b5be706bccc8b1479e1. While a PKI token is a much longer string and harder to work with. It is suggested to set Keystone to provide UUID tokens within your cloud.
In the following section, there is an example of making an authentication request for a secure token. Making API requests using **cURL**, a useful tool to interact with RESTful APIs, is the easiest approach. Using cURL with various options, you can simulate actions similar to ones using the OpenStack CLI or the Horizon dashboard:

```bash
$ curl -d @credentials.json -X POST -H "Content-Type: application/json" http://127.0.0.1:5000/v2.0/tokens | python -mjson.tool
```

Since the credential string is fairly long and easy to manipulate incorrectly, it is suggested to utilize the `-d @<filename>` functionality part of cURL. This allows you to insert the credential string into a file and then pass it into your API request by just referencing the file. This exercise is very similar to creating a client environment script (also known as OpenRC files).

Adding `| python -mjson.tool` to the end of your API request makes the JSON output easier to read.

The following is an example of the credential string:

```json
{"auth": {"tenantName": "admin", "passwordCredentials": {"username": "raxuser", "password": "raxpasswd"}}}
```

When the example is executed against the Keystone API, it will respond with an authentication token. That token should be used for all subsequent API requests. Keep in mind that the token does expire, but traditionally, a token is configured to last 24 hours from the creation timestamp.

The **token** can be found in the second to last section of the JSON output, in the section labeled **token** as shown in the following code snippet:

```json
"token": {
    "audit_ids": [
        "tWnOdGc-Qpu71Ag6QUo9JQ"
    ],
    "expires": "2015-06-30T04:53:27Z",
    "id": "907ca229af164a0918a661fffa224747",
    "issued_at": "2015-06-29T16:53:27.191192",
    "tenant": {
        "description": "Admin Tenant",
        "enabled": true,
        "id": "4cc43830491046ada1f0f26317da41c0",
        "name": "admin"
    }
}
```
Once you have the authentication token, you can begin crafting subsequent API requests to request information about your cloud and/or execute tasks. Now, we will request the list of services available in your cloud, using the following command:


The output from this API request will be the complete list of services registered within your cloud by name, description, type, id, and whether it is active. An abstract of the output will look similar to the following code block:

```json
{
    "OS-KSADM:services": [
    {
        "description": "Nova Compute Service",
        "enabled": true,
        "id": "020cc772b9c942eb979fc587877a9239",
        "name": "nova",
        "type": "compute"
    },
    {
        "description": "Nova Compute Service V3",
        "enabled": true,
        "id": "1565c929d84b423fb3c9561b22e4468c",
        "name": "novav3",
        "type": "computev3"
    },
    ...
}
```

**Via CLI**

All the base principles applied to using the API in the preceding section also applies to using the CLI. The major difference with the CLI is that all you need to do is create an OpenRC file with your credentials and execute defined commands. The CLI handles the formatting of the API calls behind the scenes and also takes care of grabbing the token for subsequent requests and even handles formatting the output.

As discussed earlier, first you need to authenticate against Keystone to be granted a secure token. This action is accomplished by first sourcing your OpenRC file and then by executing the `service-list` command. The following example will demonstrate this in more detail.
Here is an example of an OpenRC file named openrc:

```bash
# To use an OpenStack cloud you need to authenticate against keystone.
export OS_ENDPOINT_TYPE=internalURL
export OS_USERNAME=admin
export OS_TENANT_NAME=admin
export OS_AUTH_URL=http://127.0.0.1:5000/v2.0

# With Keystone you pass the keystone password.
echo "Please enter your OpenStack Password: "
read -sr OS_PASSWORD_INPUT
export OS_PASSWORD=$OS_PASSWORD_INPUT
```

Once you create and source the OpenRC file, you can begin using the CLI to execute commands, such as requesting the list of services; you can follow the following working example:

```bash
$ source openrc
$ keystone service-list
```

The output will look similar to this:

```
Listing the OpenStack endpoints
We will now move onto listing the available endpoints registered within your cloud. You will note that the process is very similar to the previous steps just explained.
```
Via API
Since we are already authenticated against Keystone in the previous example, we can just execute the following command to get back the full list of API endpoints available for your OpenStack cloud.

```
$ curl -X GET http://127.0.0.1:35357/v2.0/endpoints -H "Accept: application/json" -H "X-Auth-Token: 907ca229af164a09918a661ffa224747" | python -m json.tool
```

The output of this API request will be the complete list of endpoints registered within your cloud by `adminurl`, `internalurl`, `publicurl`, `region`, `service_id`, `id`, and whether it is active. An abstract of the output will look similar to the following code block:

```
{
    "endpoints": [
        {
            "adminurl": "http://172.29.236.7:8774/v2/%(tenant_id)s",
            "enabled": true,
            "id": "90603842a5a54958a7768dd909d43237",
            "internalurl": "http://172.29.236.7:8774/v2/%(tenant_id)s",
            "publicurl": "http://172.29.236.7:8774/v2/%(tenant_id)s",
            "region": "RegionOne",
            "service_id": "020cc772b9c942eb979fc587877a9239"
        }, ...
```

Via CLI
As with the earlier CLI request, after sourcing the OpenRC file you will simply execute the following command:

```
$ keystone endpoint-list
```

The output will look similar to the following screenshot:
An Introduction to OpenStack

Summary
At this point in the book, you should have a clear understanding of OpenStack and how to use the various services that make up your OpenStack cloud. Also, you should be able to communicate some of the key features and benefits to using OpenStack.

We will now transition into learning about Ansible and why using it in conjunction with OpenStack is a great combination.
Where to buy this book

You can buy OpenStack Administration with Ansible from the Packt Publishing website. Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.