OpenStack

MARK LAMOURINE

Mark Lamourine is a senior software developer at Red Hat. He's worked for the last few years on the OpenShift project. He's a coder by training, a sysadmin and toolsmith by trade, and an advocate for the use of Raspberry Pi style computers to teach computing and system administration in schools. Mark has been a frequent contributor to the *;login:* Book Reviews column. markllama@gmail.com

The goal of OpenStack is nothing less than to virtualize and automate every aspect of a corporate computational infrastructure. The purpose is to provide self-service access to all of the resources that traditionally have required the intervention of a collection of teams to manage. In this article, I describe where OpenStack came from and what the various parts do.

OpenStack is an implementation of Infrastructure as a Service (annoyingly abbreviated as IaaS and amusingly pronounced "eye-ass"). Several commercial IaaS services are available, with the best known being Amazon Web Services, Google Compute Engine, and Rackspace. IaaS is also known as a "cloud" service, and there are also commercial products to create a "private cloud," most notably VMware.

OpenStack was developed as an alternative to the commercial cloud providers. With it, you can create private or public cloud services and move resources between them. OpenStack uses concepts and terminology that are compatible with Amazon Web Services and that are becoming de facto standards.

Both Red Hat, through the Red Hat Distribution of OpenStack (RDO) community project, and Canonical offer installers designed to make the installation of a demo or proof-of-concept service relatively easy for those who want to experiment with running their own cloud.

Origin, History, and Release Naming

In 2010 Rackspace and NASA created the OpenStack Foundation with the goal of producing an open source IaaS project. Since then, more than 150 other companies and organizations have joined the project. NASA dropped out in 2012 citing lack of internal progress implementing OpenStack services and redirected funding to using commercial cloud providers. In the past two years, the pace of development and improvement has increased dramatically, to the point that all three major commercial Linux distribution providers (Canonical, SuSE, and Red Hat) have customized OpenStack offerings.

OpenStack development versioning employs a code-word scheme using English alphabetic ordering. The first development cycle was code named Austin (2010.1) and was released in October 2010. New versions have been released approximately every six months since the initial release. These are the most recent development code names:

- Essex (2012.1)
- Folsom (2012.2)
- Grizzly (2013.1)
- Havana (2013.2)
- Icehouse (2014.1) (release pending as of this writing)

Once released, the stable version is numbered with the four-digit year and a single-digit serial number, (YYYY.N). No one expects more than nine releases in a calendar year. Most people continue to refer to them by their code names.

The development pace is so rapid that the OpenStack foundation only lists the current release and the previous one as supported at any given time. I see the adoption by commercial distributions as an indicator that they think the current code base is stable enough to allow economical long-term support.

Any references to capabilities here will be to the Havana (2013.2) release unless otherwise noted.

Function and Stability

OpenStack is designed as a set of agent services, each of which manages some aspect of what would otherwise be a physical infrastructure. Of course, the initial focus was on virtualized computation, but there are now subservices that manage storage (three ways), network topology, authentication, as well as a unified Web user interface. As with the development cycles, OpenStack uses code names for the active components that make up the service. The service components are detailed below.

It's been four years since the initial release, and it's really only with the Folsom (2012.2) release that enough components are available and stable to attempt to create a reliable user service. As development progresses, usage patterns are discovered that indicate the need for creating new first-class components to manage different aspects of the whole. The Folsom release was the first to include a networking component (Neutron), which had previously been part of the computation service (Nova/Quantum).

Bare metal provisioning, which has been handled by a plugin to Nova, is getting a service agent of its own (code named "Ironic," in Icehouse 2014.1), which will be more capable and flexible. Communication between the components is currently carried over an AMQP bus implemented using RabbitMQ or QPID. A new OpenStack messaging and RPC service is in the works (code named Oslo, also in Icehouse). Each of these new agent services will replace and enhance some aspect of the current systems, but one can hope that the existing components have become stable enough that most changes will be additive rather than transformative.

Components

OpenStack is enamored of code names, and for anyone interested in deploying an OpenStack service, the first task is to learn the taxonomy. There are actually some good reasons to use code names. OpenStack is meant to be a modular system; and, at least once so far, an implementation of a subsystem (Quantum) has been replaced with a completely new implementation (Neutron).

Most components are active agents. They subscribe to a messaging service (AMQP) to accept commands and return responses. Each service also has a CLI tool that can communicate directly with the agent. The active components also have a backing database for persistence across restarts.

Compute (Nova)

The core of an IaaS system is its virtualized computers. The Nova service provides the compute resources for OpenStack. It controls the placement and management of the virtual machines within the running service. Originally, Nova also contained the networking, which has since moved to Neutron.

Storage

OpenStack offers several flavors of persistent storage, each of which is designed for a specific set of tasks.

IMAGE STORAGE (GLANCE)

Glance is the image store used for creating new running instances or for storing the state of an instance that has been paused. Glance takes complete file systems and bootable disk images as input and makes them available to boot or mount on running instances.

BLOCK STORAGE (CINDER)

Cinder is the OpenStack block storage service. This is where you allocate additional disk space to your running instances. Cinder storage is persistent across reboots. Cinder can be backed by a number of traditional block storage services such as NFS or Gluster.

OBJECT STORAGE (SWIFT)

Swift offers a way to store and retrieve whole blobs of data. The data are accessed via a REST protocol, which can be coded into applications using an appropriate API library. Object storage provides a means for an application to store and share data across different instances. The object store is arranged as a hierarchical set of "container" objects, each of which can hold other containers or discrete data objects. In this way, it corresponds roughly to the structure of a file system.

Network (Neutron)

The Neutron service provides network connectivity for the Nova instances. It creates a software-defined network (SDN) that allows tight control over communication between instances, as well as access from outside of the OpenStack network.

Authentication/Authorization (Keystone)

All of the OpenStack services require user access control, and the Keystone service provides the user management and resource control policy.

User Interface (Horizon)

The Horizon user interface is one of the more recent additions. It provides a single-pane Web UI to OpenStack as a whole. It layers a task-related view of OpenStack over the functional services, which allows end users and administrators to focus on their jobs without being concerned with...well, this list of agents.

Monitoring (Ceiliometer)

Another recent addition is the monitoring function provided by the Ceiliometer service. Ceiliometer is focused on monitoring and providing metrics for the OpenStack services themselves, but the documentation claims that it is designed to be extensible. This functionality would allow implementers to add probes and metrics to reach inside instances and applications as well.

Orchestration (Heat)

The Heat service gives the OpenStack user a standardized means to define complex configurations of compute, storage, and networking resources and to apply them repeatably. Although Heat deals mainly with managing the OpenStack resources, it has interfaces with several popular OS and application-level configuration management (CM) tools, including Puppet and Chef.

Heat uses templates to define reusable configurations. The user defines a configuration including compute, storage, and networking as well as providing input to any OS configuration that will be applied by a CM system.

Installation Tools

As mentioned, OpenStack is a complex service. However, it does follow several standard patterns. Additionally, several efforts are in progress to ease the installation process and to make installs consistent.

Both RPM and Debian-based Linux systems have installer efforts.

Puppet Modules

Each of the OpenStack services has a corresponding Puppet module to aid in installation and configuration. If you're familiar with both Puppet and OpenStack, you could probably use these to create a working service, but this method isn't recommended.

RDO—Packstack (RPM)

Packstack is intended for single host or small development or demonstration setups. It's produced by the RDO foundation, which is the community version of Red Hat's implementation of OpenStack.

Packstack can run either as an interactive session, or it can accept an answer file that defines all of the responses. It can install the component services on a single host or a small set of hosts.

RDO—The Foreman

The Foreman is primarily a hardware-provisioning tool, but it also has features to make use of Puppet to define the OS and application configuration of managed systems. The RDO project has defined and packaged an installer, which makes use of the Foreman host group definitions and the OpenStack Puppet modules to create a working installation. With the Foreman, it is possible to create more complex configurations by directly using the Puppet module inputs.

Ubuntu OpenStack Installer (Debian Package)

Canonical and Ubuntu also have an OpenStack installer effort. Canonical has taken a different approach from Red Hat. They provide a bootable disk image that can be written to a USB storage device or to a DVD. On first boot, the installer walks the user through the process.

Conclusion

The idea of IaaS is too powerful to ignore in the long term. Although it is still experiencing growing pains, so many players, large and small, are now backing and contributing resources to OpenStack that it's a safe bet it will be around for a while and will improve with time.

If you're eyeing a service like Amazon Web Services and thinking, "I wish I could do that here," then you should look at OpenStack. Plan to do several iterations of installation so that you can get to know the component services and their interactions as well as your real use cases. With commercial cloud service developers and project managers starting to get used to the idea of on-demand resources, they're going to be clamoring for it soon, and OpenStack offers you the means to provide it.

References

A lot of people are working on OpenStack, so there are a lot of references. These are just a few select ones to get started.

- OpenStack: http://www.openstack.org
- OpenStack documentation repository: http://docs.openstack.org
- OpenStack Administrator's Guide: http://docs.openstack.org/admin-guide-cloud/content/
- Packstack: https://wiki.openstack.org/wiki/Packstack

SOFTWARE

- GitHub OpenStack: https://github.com/openstack
- GitHub Packstack: https://github.com/stackforge/packstack
- GitHub OpenStack Puppet modules: https://github.com/stackforge/?query=puppet-

VENDOR AND DISTRIBUTION COMMUNITIES

- Red Hat RDO: http://openstack.redhat.com/Main_Page
- Ubuntu: http://www.ubuntu.com/download/cloud/install-ubuntu-cloud

SAVE THE DATE!

LISA'14

Sponsored by USENIX in cooperation with LOPSA

NOVEMBER 9–14, 2014
SEATTLE, WA

www.usenix.org/lisa14

USENIX's LISA conference is the premier meeting place for professionals who make computing work efficiently across a variety of industries. If you're an IT operations professional, site-reliability engineer, system administrator, architect, software engineer, researcher, or otherwise involved in ensuring that IT services are effectively delivered to others—this is your conference, and we'd love to have you here.

The sessions at LISA '14 will address the topics most relevant to those working in information technology today, including:

- Systems Engineering
- Monitoring and Metrics
- DevOps
- Security
- Culture
- And much more!

The 6-day program includes invited talks, workshops, panels, AMA conversations, training courses, and refereed paper and poster presentations. The on-site Hack Lab will give attendees and speakers the opportunity to demo, collaborate, and test out new ideas. Evening receptions and Birds-of-a-Feather sessions provide opportunities to meet and network with those that share your interests.

There's a better way to get your job done. Learn it at LISA.

Stay Connected...

twitter.com/lisaconference
www.usenix.org/youtube
www.usenix.org/gplus
www.usenix.org/facebook
www.usenix.org/linkedin
www.usenix.org/blog