Rackspace Private Cloud – Updated to OpenStack Grizzly Release!

Just in case you didn't spot this in the Twitter stream yesterday, there was some really cool news from the team at Rackspace! The Rackspace Private Cloud software is now updated with the Grizzly release of OpenStack.
(http://www.rackspace.com/knowledge_center/article/about-rackspace-private-cloud).

Dubbed the Rackspace Private Cloud v4.0, this is a great way to get your OpenStack cloud up and running with significantly less configuration required. The installation scripts will deploy onto Ubuntu 12.04 and CentOS 6.3 and includes the Chef server with cookbooks for simple management and deployment once you build your environment.

Scripts versus ISO

One of the really neat things that Rackspace did in the past was to build a standalone ISO to be able to deploy the Rackspace Private Cloud which was running the Folsom release of OpenStack. This ISO is still available, but for now, the Grizzly release is only available with the scripted deployment.

If you still want to use the standalone ISO, you can continue to use my series on deploying OpenStack with VMware Workstation (Getting Started, Setting up the All-In-One, Adding Cinder) and you can still download the archived versions of the ISO, and also get to the OpenCenter (version 3.0) installation through links in the Rackspace Knowledge Center
– http://www.rackspace.com/knowledge_center/article/rackspace-private-cloud-software-archived-versions

What's new with this Release?

The use of Grizzly will add all of the new features of OpenStack of course, but along with that are new Chef cookbooks packaged into the build which make your continued management of OpenStack Grizzly much easier; Especially for those who are just dipping their toes into the OpenStack waters.

All of the core services are included (Nova, Horizon, Swift, Cinder, Keystone, Neutron, Glance) as well as Ceilometer which is the metering tool that is most likely slated for core project inclusion with the upcoming Havana release of the OpenStack software in Q4 of 2013.

Image courtesy of Rackspace:
http://www.rackspace.com/cloud/private/openstack_software/

HA all the way!

Another really cool feature in the Rackspace 4.0 release is the use of HA features of RabbitMQ and database services. The use of HA design in the build gives us another edge up over building from scratch because their are some challenges for many to build out the underlying services in a multi-master configuration.

You have to ensure that your node deployments are built with HA in mind of course, but the steps saved in designing the HA database and message queuing are appreciated because of the time savings you receive from these features being pre-configured.

Thank You Rackspace for another great way to deploy OpenStack and I encourage everyone to take a look at this and reach out to the Rackspace crew. All that you have to do is to click the Download button on the Rackspace Private Cloud page here: http://www.rackspace.com/cloud/private/openstack_software/ and you will receive contact information in email from the Rackspace support team.

OpenStack Lab on VMware Workstation – Adding Cinder volume features

Now that we have our OpenStack lab up and running, we want to click around and kick the tires on all of the features. One of the first things that we will find is that the All-In-One is really an "almost" All-In-One. What I mean by that, is that there are services available which aren't configured, and require a little more care and feeding.

Where is my Cinder?

The Block Storage feature of OpenStack is appropriately (or perhaps tongue-in-cheek) named Cinder. This service allows you to create block storage in your OpenStack cloud and attach to virtual machine instances. While the administration dashboard has the Volumes tab and appears to be all set to go, it requires a few extra steps to get Cinder volumes up and running.

Testing our Cinder volumes out of the box

Before we do anything, I wanted to show you how it doesn't work. That sounds strange, but I'm stubborn and I like to prove things out, and if you are like me, you probably clicked through the dashboard provisioning some goodies and then realized that something wasn't right with the Volumes page.

First, log in to your All-In-One dashboard as Demo (user account created during the deployment):

In your dashboard (aka Glance) click on the **Volumes** tab and click the **Create Volume** button:

In the Create Volume options box, fill in the name of your volume and a description first:

Next, scroll down and fill in the Size (GB) field. I'm using a 1 GB volume for my test:

In a matter of a few seconds we see the results of our volume creation:

Yup…it failed. Booooo! But that's ok, because as I mentioned at the start of the post, we have to configure Cinder and the volume space before this would work.

Adding the Cinder foundation

We need to do the following to enable and configure Cinder for our All-In-One build. These steps are:

1. Create a disk drive to store our Cinder volumes
2. Add Cinder components to the OpenStack build
3. Create the virtual storage volume and assign it to our Cinder volume pool
4. Restart the newly installed services
5. Test Cinder volume creation

Although we have a large disk assigned to the All-In-One (200 GB in my case), we still need another disk to use for the Cinder volumes. Remember that our 200 GB disk is for the virtual machines only. Additional volumes will be stored in our dedicated Cinder volumes.

For cleanliness, I'm going to shut down my OpenStack instance and add the disk to my VMware Workstation. We edit the settings and click on the **Add** button:

Choose **Disk** and click **Next**:

We choose to **Create a new virtual disk** and click **Next**:

Take the default which is **SCSI** and click **Next**:

For fun, I've set it at 40 GB and as always I choose the **Store virtual disk as a single file**:

I'll name the disk appropriately as **OPENSTACK01-cinder.vmdk** so I remember which one it is:

Now that we have our new disk added, we start up our All-In-One VM. Once we are up and running we will SSH into the machine to configure the new Cinder volume. You will need a client such as PuTTY, or whatever your SSH client of choice is.

Log in as the user you have created during the initial build. Once we are logged in, we will elevate

our privileges to configure everything:

Type in **sudo su –** which puts you into the console with su privileges and the full environment. Next you type **source openrc** which loads the environment variables for our OpenStack configuration that we need for the other script processes:

Before we add new things, run the **knife node show <your node name>** which shows the results like the screen below. You see under the Run List that there is no Cinder mentioned:

Now we type **knife node run_list add <your node name> 'role[cinder-all]'** which will add the Cinder roles to the Run List:

Now we type **chef-client** which executes the cookbook recipe to create and initialize the Cinder services which are cinder-volume, cinder-api and cinder-scheduler.

This step takes a while once you start it, so you can leave it running for 5 minutes or so:

Once the installation process completed, you will see a screen like the one below with the last message being **Report Handlers Complete**:

Now we create the volume by using the **pvcreate /dev/sdb** command:

Next we create the volume group using the **vgcreate cinder-volumes /dev/sdb** command:

Finally we have to restart the Cinder services with the following commands:

service cinder-volume restart

service cinder-api restart

service cinder-scheduler restart

Phew! We did it! Now we head back to our dashboard to retry the process of creating a volume. Just as before, log in as **demo**, go to the **Volumes** tab and click on the **Create Volume** button:

In a few seconds you will see the console working away on your volume creation and finally it shows

our new volume with a status of **Available**:

We did it!

Now you can deploy Cinder volumes as a part of your All-In-One lab deployment. This will come in handy later

Veeam Cloud Backup – Another Great Way to Protect with Veeam!

For current and potential customers of Veeam Backup & Replication, there is a new sheriff in town! Thanks to the awesome folks at Veeam Software, they are introducing the new Veeam Backup Cloud Edition.

Why Cloud Backup?

The advantage to using Veeam Backup Cloud Edition is that you can leverage numerous cloud storage services (15 key vendors already on board!) and thanks to pre-transfer compression, AES 256-bit encryption and the fantastic Veeam de-duplication capability, you can get the best of every side of the equation.

How does it work?

The Cloud Backup is an add-on for existing customers with service agreements in place, and for new customers of the Veeam Backup & Replication suite, the cloud storage gateway is packed inside there to so you get full functionality in-site and now off-site with your favorite cloud storage provider.

Is This Right for You?

The challenge with using any cloud storage service for storage is that it can be costly depending on how you use it. That being said, the Cloud Backup Edition is pretty nicely targeted towards small to mid-sized businesses who can add the flexibility and protection of cloud storage and keep their costs down by reducing or even eliminating tape and archival off-site.

Many companies are not running backups at all because of the potential costs. Now the barrier to true system and data protection has been lifted.

Hybrid Cloud? Yes please!!

For those who have some concerns over putting all of their backed up and replicated eggs in one

public cloud basket, there is also the option of using private cloud or hybrid cloud configurations like OpenStack. Another great reason to take a strong look at Veeam Backup Cloud Edition and OpenStack

The Cloud Edition is available by subscription on a per-CPU socket licensing model so you buy what you need and big capital cost to get you up and rolling.

Time for the teaser view!

Here are some screen grabs from the user interface and you can see what you will be looking at when you get your Veeam environment up and running:

This is a quick walk through of the backup process. First we setup our Cloud providers (lots to choose as you can see) and these credentials will be securely used for the backup plans:

Simply setup your access and secret key information. You can even use the Cost Estimates tab to assign the storage usage rates for an estimate of the overall storage utilization for your object storage. NOTE: These are estimates only unfortunately. At this point, the Cloud providers do not fully expose their billing information via their storage API.

Backup plans are assigned compression and encryption configurations:

Encryption options available for multiple bit lengths and encryption algorithms:

Purge options for the backup plan give you flexibility to retain deleted files per-plan, or using the global policy assigned to the entire product:

Using the Backup Plan Wizard lets you choose files, folders and volumes as needed:

Selecting your destination storage provider shows your cloud options as well as File System targets:

Getting your Veeam Backup Cloud Edition

It’s as easy as clicking the graphic below to get your Free Trial and then you are on your way to new ways to protect your environment! And don’t forget to tell them that DiscoPosse sent you

OpenStack Lab on VMware Workstation – Setting up the All-In-One VM

In our first post, we got to the start of the OpenStack install using the Rackspace Private Cloud ISO (Alamo). Because we had to set up the VM, we stopped at our first menu option. Now we are going to set up the base system using the All-In-One option which co-locates all of the OpenStack node roles (controller and compute) in a single system.

Understanding the OpenStack Framework

First we need to do some quick review of what exactly will be found in our OpenStack All-In-One VM. There are numerous components in the OpenStack framework including:

- Compute (Nova)
- Image Service (Glance)
- Dashboard (Horizon)
- Identity (Keystone)
- Block Storage (Cinder)

Object Storage (Swift) is one of the components that's not packaged with the All-In-One deployment, but you can install it after the fact if you desire.

Here is a logical layout of the design (image care of OpenStack.org):

So what exactly are these components is your next question right?

Compute (Nova)

Nova is the compute node system of OpenStack which provisions virtual servers. It was built by NASA. No, seriously, it was built by NASA. This is some true space age computing!

Image Service (Glance)

The role of Glance is to store a catalog of virtual images. The images will be deployed into compute nodes and will be key for any real dynamic environments. For lab purposes, we won't be using this as much but we need to fully understand its operational requirements and behavior.

Dashboard (Horizon)

This is your web application which will be how you perform most activities in the OpenStack cloud. This is how you launch a new instance, manage ACLs and network addresses.

Identity (Keystone)

The identity management of OpenStack is inside Keystone. This is your authentication and

authorization system for your cloud. This is also a service catalog for your OpenStack cloud.

Block Storage (Cinder)

Newly minted for the Folsom release of OpenStack, the Cinder block storage system houses your VMs. Just like your traditional VMware storage environment, this is block, not NFS or CIFS.

Let's Get Started

We begin where we left off, which is at the first menu during the OpenStack installation process. We are choosing the **All-In-One** deployment option:

We give our VM a static IP address from the VMware Workstation NAT range so it can communicate with the outside world:

Provide the gateway address next. Note that it will default to x.x.x.1 so you have to make sure it is the NAT gateway for your virtual network which should be x.x.x.2

We need DNS to be able to resolve addresses. I've chosen to use 8.8.8.8 which is the Google public DNS:

I always love a little humor during an install. The "you can make something up here" always makes me smile. I've used **openstack01** as the hostname for my instance:

This can be whatever is appropriate for your system, but I've opted for my own domain which is corp.discoposse.com:

The Nova range (private internal range) is already selected, so I've accepted the default:

Enter a password for your **admin** user (I've used **openstack** to make it easy to remember:

Re-enter your password:

Next we create the OpenStack user account (consumer of our services). I've opted to keep the default which is **demo**:

Again, we need to select a password. I've also used **openstack** to make it easy to remember:

Re-enter the password:

We provide a full name to a non-admin user account. I've used my name:

Next we set the username for the non-admin account:

Set your password:

Re-enter your password:

Now the installation will begin. This takes a little while, but you will be able to monitor the progress on the console:

Once the base installation is done, the system needs to contact the public mirror for Ubuntu to be able to download additional libraries and applications:

The subfolder will be defaulted to /Ubuntu and you can leave this as-is and continue:

If you have an HTTP proxy between your VMware Workstation and the internet, enter the information here. I'm directly connected so I have just left this blank:

Next the additional components are downloaded:

You will notice the installer screen changes and the actual Rackspace OpenStack environment begins to install and configure itself:

A little bit of waiting (depending on speed of your system) and you will see the final screen with the current configuration on the console of your VM:

Browse to the IP address using https://youripaddress as shown in the configuration screen. You are presented with the login screen where you will use the **admin** user and your password to log in:

Now you see the Rackspace OpenStack console for your new VM deployment:

Yay!!!

Congratulations! You are now at the next step in our journey into discovering OpenStack. More posts to come!