15-719 / 18-847B, Fall 2013, Project 2: Auto Scaling with OpenStack (Part 1)

Assigned: September 23rd, 2013
Due: November 8th, 2013

OpenStack is an open-source computing platform that is rapidly gaining traction within many companies and organizations, such as NASA, RackSpace, and HP. In this project, you will implement an auto-scaling service as a PaaS addition to OpenStack. This document covers part 1 of the project, which involves bootstrapping necessary functionality. For this part of the project, you will:

1. Set up an account on a shared cluster called Marmot, so that you can allocate a bare-metal machine for your work.
2. Instantiate a bare-metal node on Marmot.
3. Deploy a single-node configuration of OpenStack (called DevStack) on the Marmot node.
4. Deploy and launch VMs within your DevStack instance.
5. Write a simple performance monitor that runs within DevStack VMs and collects data.
6. Write a simple performance collector that receives performace data from your monitors.
7. Demonstrate that your performance monitors and performance collector can communicate with each other.

Part 2 of the project will involve converting your basic performance aggregator and performance collector into a full-fledged auto-scaling service. It will be released after part 1 is due.

1 Collaboration & cheating policy

This project **must** be completed in groups of two. Students who wish to work individually must obtain special permission from the instructors. Do not share code or pseudo-code with members who are not in your group.

Cheating will not be tolerated. You are prohibited from using any solutions or code you find online. Please check the official 15-719 cheating policy for additional details.

2 Step 1: Allocate a Marmot node

Marmot is a computing cluster that provisions bare-metal hardware. It can be accessed at `https://marmot.pdl.cmu.edu`. Marmot requires its users to be part of a *project*. Users provision bare-metal nodes by creating an *experiment*. When creating an experiment, users can specify the number of machine desired, the network topology between them, and OS images to be used. An overview of Marmot can be found at: https://www.nmc-probe.org/wiki/Getting_Started.

You should have already received e-mail with information about how to create an account on Marmot. When you create an account via the link in the e-mail, you are automatically enrolled in OpenStackSys, the Marmot project for this class.

To create an instance, ssh to marmot-ops.pdl.cmu.edu with your Marmot username and password and type the following command:

```
/share/probe/bin/probe-makebed -e <experiment name> -p OpenStackSys
-i UBUNTU12-64-PROBE -g <your group number> -n 1
-s /proj/OpenStackSys/scripts/startup
```

Note that `<experiment name>` is a custom name for your experiment. <experiment name> **MUST** be a single word with no underscores. The `-n 1` option indicates that you only want one machine to be included in your experiment. On success, the output of the command will list the hostname of your newly created instance. The script run by the `-s` option will provision a 64GB scratch partition, mounted on `/l0`.

You can log into to your node via `ssh`. Note that running experiments will be swapped out after a period of inactivity. Running experiments also have a maximum lifetime, after which they will be terminated. You can restore swapped-out experiments via the dashboard at marmot-ops.pdl.cmu.edu. Your scratch partition will not be restored. To avoid data loss, periodically create snapshots of your VM image (see the instructions for doing so at https://www.nmc-probe.org/wiki/UserGuide).

To terminate your instance, type:

```
/share/probe/bin/probe-makebed -e <experiment name> -p OpenStackSys -r
```

Further details can be found at: https://www.nmc-probe.org/wiki/UserGuide.

In addition to using the command line, you can also use the Marmot dashboard at https://marmot.pdl.cmu.edu to create, delete, and manage your Marmot instances.

3 Step 2: Create a Devstack cloud within your Marmot node

DevStack is the name for a single-node OpenStack installation. To configure the Marmot node for a DevStack cloud and to install DevStack, type the following commands:

1. `setenv http_proxy http://ops.marmot.pdl.cmu.local:8888`
2. `setenv https_proxy http://ops.marmot.pdl.cmu.local:8888`
3. `cd /users/<your user id>`.
4. `git clone http://github.com/openstack-dev/devstack.git -b stable/havana`.
5. `/proj/OpenStackSys/setup_devstack_env.sh`
6. `cp /proj/OpenStackSys/scripts/localrc </users/<your user id>/devstack`

7. `cd /users/<your user id>/stack.sh`
8. `./stack.sh`

Please be sure to read `setup_devstack_env.sh` and `localrc`. They have been commented to help you understand the details of setting up Devstack. The logs stored in `/l0/logs` can help you debug any problems you might encounter.

4 Step 3: Learn how to use DevStack

The next step is to create and destroy a few DevStack VM instances and to gain intuition about how DevStack works. Please look over the user guide at http://docs.openstack.org/user-guide/content/ for detailed instructions on how to manage DevStack VMs. You might find the commands listed in Table 1 useful.

We recommend using the command line to manage DevStack, but you can also use a Web-based UI by navigating to `http://127.0.0.1:8800` from your Marmot node. If you chose to use the web-based UI, you will have to use a text-based browser.

Choosing images: To verify that your DevStack cloud is working, use the built in `Cerros` image. For development, use the Ubuntu image in `/proj/OpenStackSys/proj2_images`.

Choosing flavors when launching a VM: Flavors indicate the amount of resources that will be allocated to a VM instance. Type `nova flavor-list` to see details. We recommend flavor 1 for the built-in Cerros image and flavour 2 for the Ubuntu image.

Choosing security groups when launching a VM: Security groups specify networking rules for VMs (e.g., what ports open). You should always specify a security group when launching a VM. The security group you use for your instances should allow access to SSH, TCP, ICMP, and UDP connections. You may to add more rules depending on the implementation of your performance monitor and collector.

Specifying a key when launch a VM: You should specify a public key when launching a VM. To do so, use the command `nova keypair-add <path to your public key>`. You can then log in to a running instance by via `ssh <instance name> -i <path to your private key>`.

5 Develop the performance monitor & performance collector

Your performance monitor should run inside any VM that you launch. Your performance collector should run as a separate process in the Marmot node (not within a VM) and should collect performance information sent to it by the monitors. You should run both as a daemon processes that launch on startup. We recommend keeping a copy of your code either on your personal machine or on Marmot in `/users/<your username>`. If you store your code in any other directory on Marmot, it might be erased when your experiment is terminated.

During development, You should periodically save snapshots of both your Marmot node and DevStack instances, so that you can restart them again without losing state.

Command	Description
`glance list`	List available VM images.
`glance image-create`	Create a new image for use by DevStack.
`nova list`	Liss running VMs.
`nova image-show`	List information about a particular image.
`nova image-create`	Create a new image.
`nova flavor-list`	List VM resource configurations.
`nova boot`	Boot a new VM.
`nova delete`	Delete a VM.
`nova suspend`	Suspend a VM.
`nova secgroup-list`	List current security groups.
`nova secgroup-list-rules`	Lists rules for a specific security group.
`nova secgroup-create`	Create a new security group.
`nova secgroup-add-rule`	Add a rule to the given security group.
`nova secgrop-delete-rule`	Delete a rule from the given security group.

Table 1: **Commonly used DevStack commands for managing VMs**. Details about these commands can be found at `http://docs.openstack.org/user-guide/content/`. You can also type in the command without any parameters to see details about it.

5.1 Performance monitor requirements

This section lists requirements for the performance monitor.

- It should run as a daemon that starts automatically at boot.
- It must communicate to the performance monitor via socket-based I/O.
- It must send current CPU usage (defined as the sum of the values in the "CPU" row of `/proc/stat`) to the performance collector every minute.

5.2 Performance collector requirements

This section lists requirements for the performance collector.

- It should run as a daemon that runs automatically at boot.
- It must print CPU usage values it receives from every VM to a log file.

6 Demo & Grading

We will grade this part of the project by asking groups to demo the functionality of their performance monitor & collector. During the demo, we will ask you to:

1. Launch two DevStack VMs running Ubuntu.
2. Show us the output of the performance collector when there is no load applied to the VMs.

3. Show us the output of the performance collector when load is applied to one of the VMs. We will apply load by running the `sysbench` benchmark as follows `sysbench -test=cpu -cpu-max-prime=20000 run`.

Your grade will be based on the functionality of your code and on your general understanding of DevStack, as determined by questions we will ask during the demo. Demos will take place on Friday, November 8th. If your code does not work properly, we will provide feedback and ask you to re-demo your code on Saturday, November 9th. You will lose 25 points if a re-demo is necessary.

References

1. Marmot user guide: `https://www.nmc-probe.org/wiki/Getting_Started`
2. OpenStack user guide: `http://docs.openstack.org/user-guide/content/`
3. OpenStack architecture (old): `http://cssoss.files.wordpress.com/2012/05/openstackbookv3-0_csscorp2.pdf`
4. Sysbench: `http://sysbench.sourceforge.net`