Setting up VMs on Cybera's RAC for CMPUT 481/681

V.1.1 November 5, 2016

Key Ideas:

1. **Backups:** Use *snapshots* to help you backup and save the state of your VMs. Use git for version control and backup of your source code. Cybera does **not** backup your data or VMs. Reboot, do not Terminate your VM instances.
2. **Security:** Understand Secure Shell, public-private key pairs, and port filtering.
3. **Not a "normal" cluster:** There is no DNS service for RAC nodes, nor a shared file system. Learn about `/etc/hosts` (see below).

To "bump up" your default resource allocation, and to ask questions of Cybera, email: rac-admin@cybera.ca, and mention you are taking CMPUT 481/681 with Paul Lu.

Cybera Documentation can be found at:

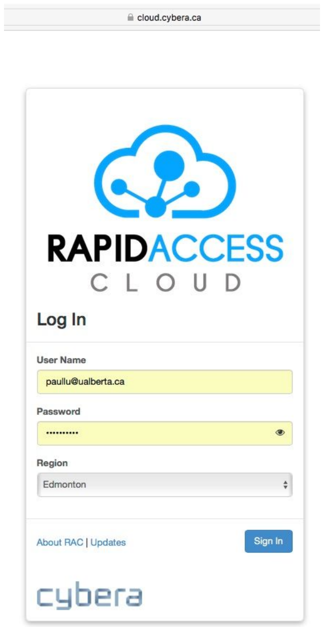
http://www.cybera.ca/services/rapid-access-cloud/
http://www.cybera.ca/uncategorized/faq/
http://www.cybera.ca/projects/cloud-resources/documentation/

Security and Access:

Before setting up the virtual machines (VMs) themselves, you will want to set up the important security aspects of the VMs, namely your:

1. **Open ports**, to allow the use of SSH and for MPI to use for socket connections
2. Secure Shell (SSH) **public-private key pair**, for logging into the VMs, both from the outside world into RAC, and within RAC between your VMs.

From your RAC Dashboard at http://cloud.cybera.ca


cloud.cybera.ca

Log In

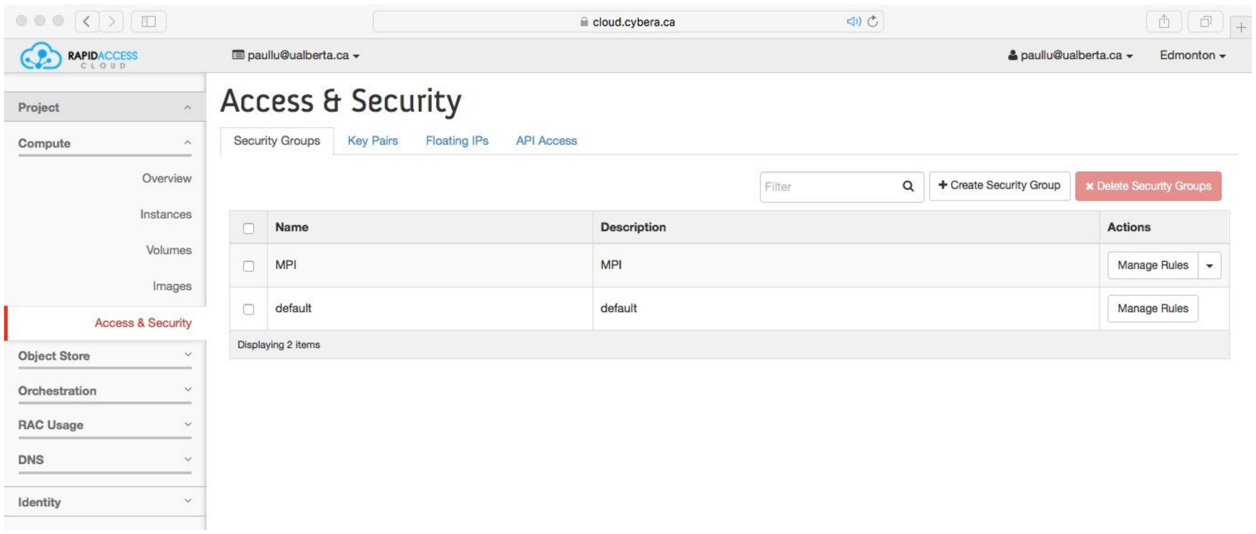
User Name: paullu@ualberta.ca

Password: ••••••••••

Region: Edmonton

Be sure you are consistent in your use of Region (e.g., Edmonton, as shown here). And, I find that my first attempt to login usually fails due to a time-out error. Logging in a second time seems to work. This behaviour might be a property of my browser (Safari) and how it automatically fills in my User Name and Password. But, be aware.

From the Menu Column on the LHS, select "Access & Security".

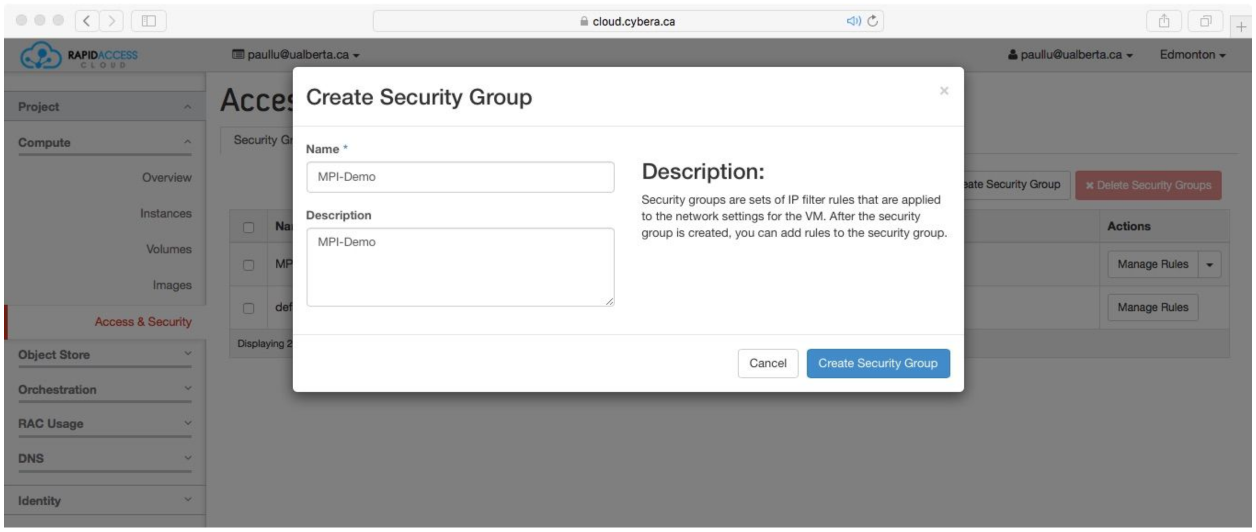

cloud.cybera.ca

Access & Security

	Name	Description	Actions
☐	MPI	MPI	Manage Rules
☐	default	default	Manage Rules

Displaying 2 items

As shown, I have already created the Security Groups (i.e., sets of rules) "MPI" and "default". To create a new Security Group, click on "+ Create Security Group" (white button). I will call the new Security Group "MPI-Demo". But, some of the screenshots might reflect Security Groups called "Demo" or something similar.



Create Security Group

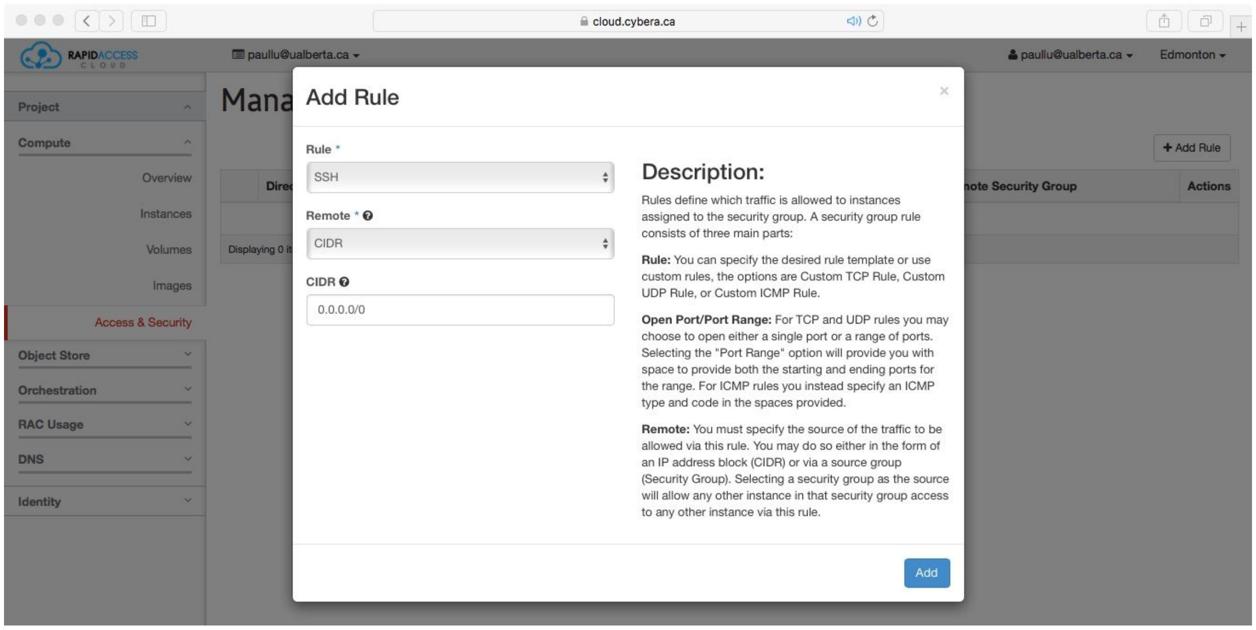
Name *: MPI-Demo

Description: MPI-Demo

Description:

Security groups are sets of IP filter rules that are applied to the network settings for the VM. After the security group is created, you can add rules to the security group.

Click on “Create Security Group” (blue button). Then select “Manage Rules” for the new rule and...



Add Rule

Rule *: SSH

Remote *: CIDR

CIDR: 0.0.0.0/0

Description:

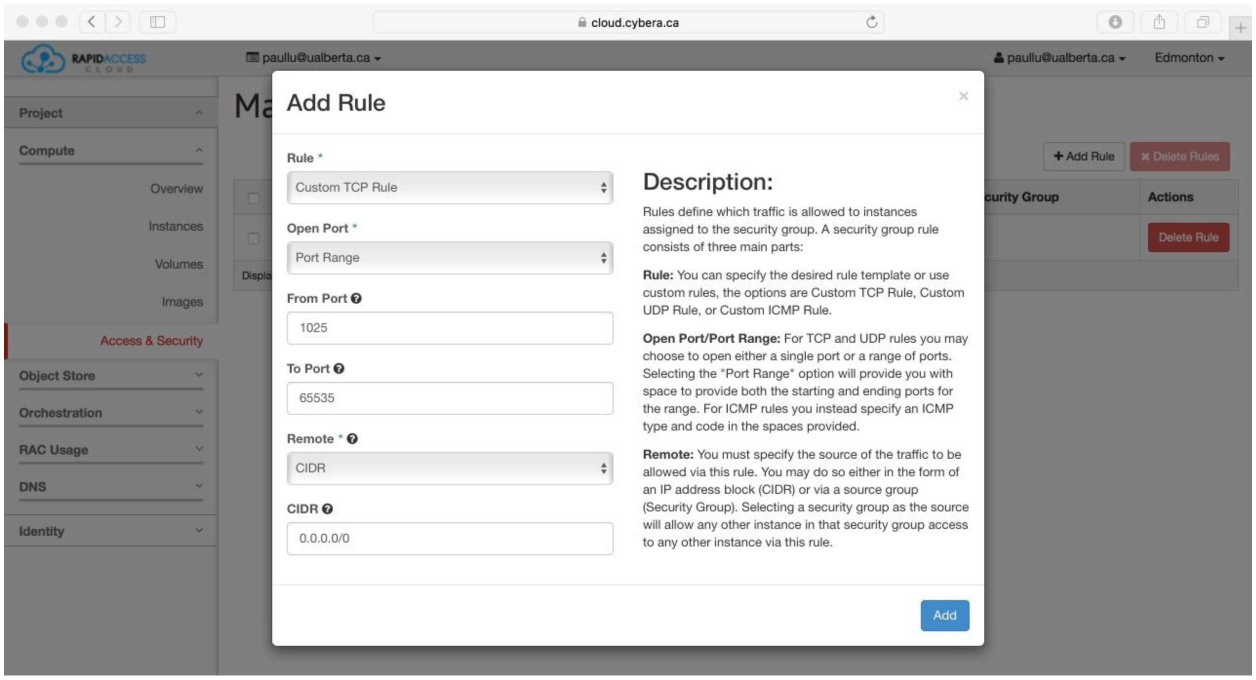
Rules define which traffic is allowed to instances assigned to the security group. A security group rule consists of three main parts:

Rule: You can specify the desired rule template or use custom rules, the options are Custom TCP Rule, Custom UDP Rule, or Custom ICMP Rule.

Open Port/Port Range: For TCP and UDP rules you may choose to open either a single port or a range of ports. Selecting the "Port Range" option will provide you with space to provide both the starting and ending ports for the range. For ICMP rules you instead specify an ICMP type and code in the spaces provided.

Remote: You must specify the source of the traffic to be allowed via this rule. You may do so either in the form of an IP address block (CIDR) or via a source group (Security Group). Selecting a security group as the source will allow any other instance in that security group access to any other instance via this rule.

Add Rule(s): First, add a rule to allow the use of Secure Shell (SSH) to login into your VMs. You will want this for all of your VMs, not just the “head node” (called `mpi0` in my example). Using SSH is common enough that you just select it from the pop-up menu for “Rule”. Click on “Add” (blue button)



Add Rule

Rule *: Custom TCP Rule

Open Port *: Port Range

From Port: 1025

To Port: 65535

Remote *: CIDR

CIDR: 0.0.0.0/0

Description:

Rules define which traffic is allowed to instances assigned to the security group. A security group rule consists of three main parts:

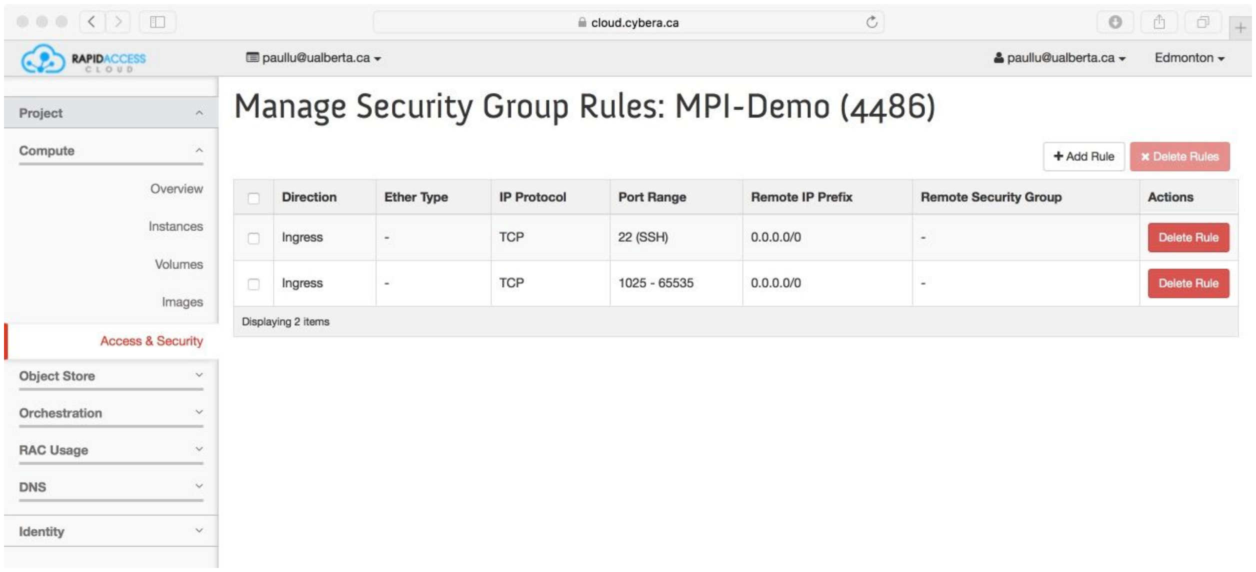
Rule: You can specify the desired rule template or use custom rules, the options are Custom TCP Rule, Custom UDP Rule, or Custom ICMP Rule.

Open Port/Port Range: For TCP and UDP rules you may choose to open either a single port or a range of ports. Selecting the "Port Range" option will provide you with space to provide both the starting and ending ports for the range. For ICMP rules you instead specify an ICMP type and code in the spaces provided.

Remote: You must specify the source of the traffic to be allowed via this rule. You may do so either in the form of an IP address block (CIDR) or via a source group (Security Group). Selecting a security group as the source will allow any other instance in that security group access to any other instance via this rule.

Add

Second, open up the ports 1025 to 65535 (i.e., the non-privileged TCP ports). Select “+ Add Rule”, select “Custom TCP Rule” on the pop-up menu, then select “Port Range” on the pop-up menu, and type “1025” and “65535” in the text fields. Then click “Add” (blue button).



Manage Security Group Rules: MPI-Demo (4486)

Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Actions
Ingress	-	TCP	22 (SSH)	0.0.0.0/0	-	Delete Rule
Ingress	-	TCP	1025 - 65535	0.0.0.0/0	-	Delete Rule

Displaying 2 items

SSH Key Pair: Another important step before creating your VMs is to create an SSH public-private key pair. This is used for *public-key authentication* to get into your “head node” VM (i.e., `mpi0` in my demonstrations) from the outside world. Other than using console (which I will not discuss any further), SSHing into a VM is the only way to log into a VM on RAC.

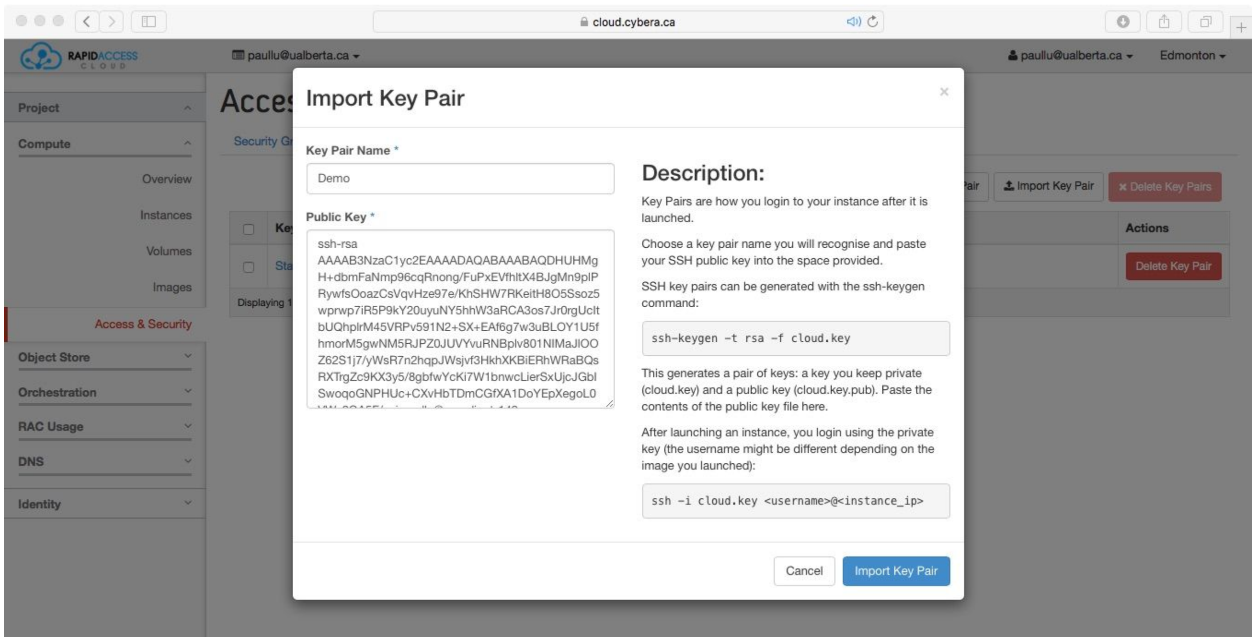
You will also use SSH behind the scenes as part of mpirun/mpiexec and how your MPI processes are started. In essence, mpirun/mpiexec will SSH into your other VMs (e.g., `mpi1`, `mpi2`, `mpi3`) to start the MPI process(es) on that machine.

For more details on public-key authentication, Google the terms “public key authentication secure shell ssh”

Next, I recommend you create or re-use a public-key pair outside of RAC/OpenStack. You have the choice of letting RAC/OpenStack create a key pair for you. Or, you can provide the public key from a pair you have already created.

In my demonstration, I reuse a key pair that I already have and give it the name “Demo”, since my `ssh-agent` on my desktop is already setup to use it. I cut-and-paste (from outside my browser) from my `~/.ssh/id_rsa.pub` file into the “Public Key” text window, then click “Import Key Pair” (blue button).

NOTE: The name of and path to your .pub file might be different, depending on how exactly you create the key pair.



Import Key Pair

Key Pair Name *

Demo

Public Key *

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDHUHMg
H+dbmFaNmp96cqRnong/FuPxEVfhltX4BJgMn9plP
RywfsOoazCsVqvHze97e/KhSHW7RKeitH8O5Ssoz5
wprwp7iR5P9kY20uyuNY5hhW3aRCA3os7Jr0rgUclt
bUQhplrM45VRPv591N2+SX+EAf6g7w3uBLOY1U5f
hmorM5gwNM5RJPZ0JUVYvuRNBplv801NIMaJlOO
Z62S1j7/yWsR7n2hqpJWsjvf3HkhXKBiERhWRaBQs
RXTrgZc9KX3y5/8gbfwYcKi7W1bnwcLierSxUjcJGbl
SwoqoGNPHUc+CXvHbTDmCGfXA1DoYEpXegoL0
```

Description:

Key Pairs are how you login to your instance after it is launched.

Choose a key pair name you will recognise and paste your SSH public key into the space provided.

SSH key pairs can be generated with the ssh-keygen command:

```
ssh-keygen -t rsa -f cloud.key
```

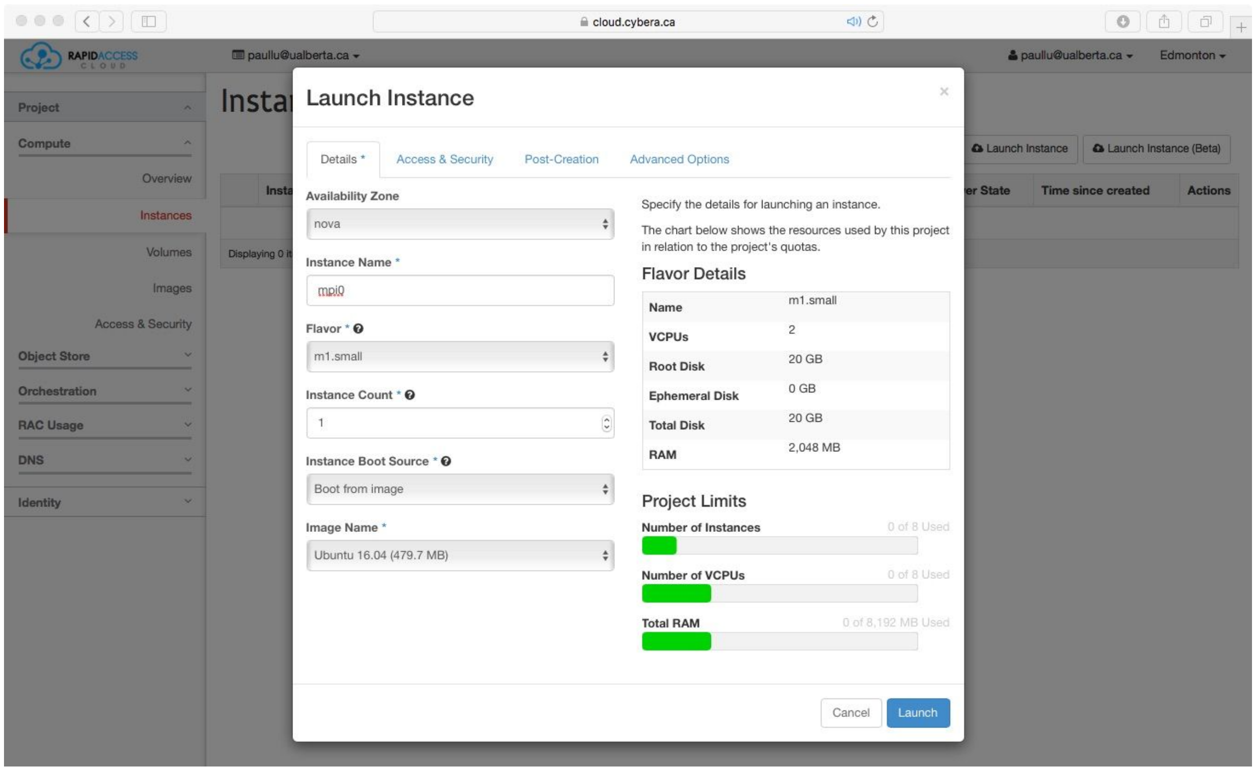
This generates a pair of keys: a key you keep private (cloud.key) and a public key (cloud.key.pub). Paste the contents of the public key file here.

After launching an instance, you login using the private key (the username might be different depending on the image you launched):

```
ssh -i cloud.key <username>@<instance_ip>
```

Creating VM Instances

Creating VM instances starts with “Instances” on the Menu Column on the LHS of the dashboard. Then select “Launch Instance” along the top of the dashboard.



Launch Instance

Availability Zone: nova

Instance Name *: mpi0

Flavor *: m1.small

Instance Count *: 1

Instance Boot Source *: Boot from image

Image Name *: Ubuntu 16.04 (479.7 MB)

Specify the details for launching an instance.

The chart below shows the resources used by this project in relation to the project’s quotas.

Flavor Details

Name	m1.small
VCPUs	2
Root Disk	20 GB
Ephemeral Disk	0 GB
Total Disk	20 GB
RAM	2,048 MB

Project Limits

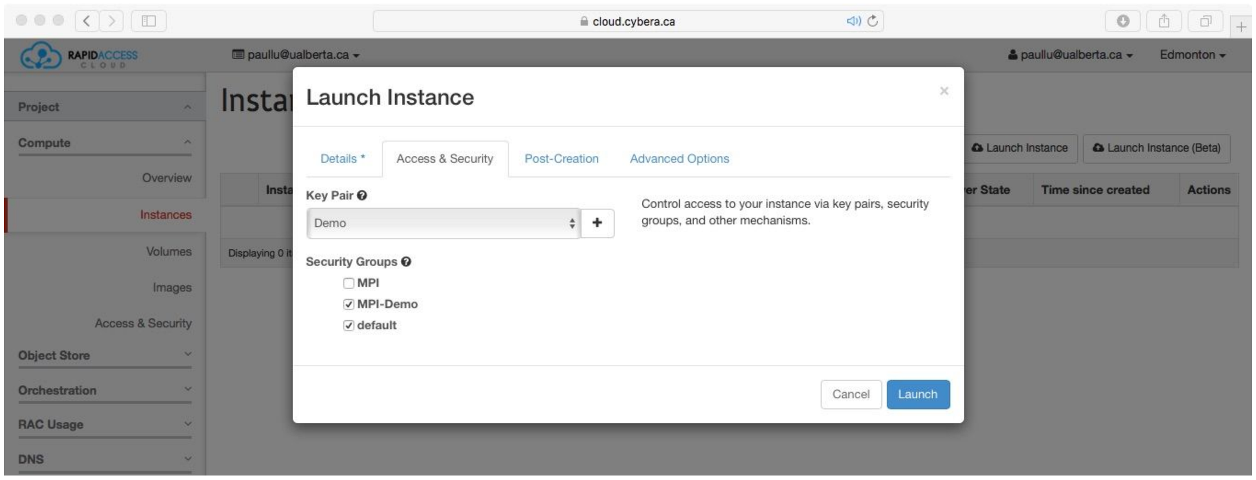
Number of Instances: 0 of 8 Used

Number of VCPUs: 0 of 8 Used

Total RAM: 0 of 8,192 MB Used

Under the tab “Details” select Availability Zone=nova, give the instance a name (`mpi0` is going to be my head node among the VMs), Flavor=m1.small, Instance Count=1.

When you select “Boot from image”, you will get a pop-up menu of different image names. I choose to use Image Name=Ubuntu 16.04. **But, do not launch yet.** We have to set up “Access & Security”.



Under the “Access & Security” tab, select the Key Pair setup previously (here, called “Demo”). I also checkbox the “MPI-Demo” and “default” options for “Security Group”. These were created earlier and allow for SSH access and MPI launching via open ports.

Now, you can launch this instance by clicking “Launch” (blue button).

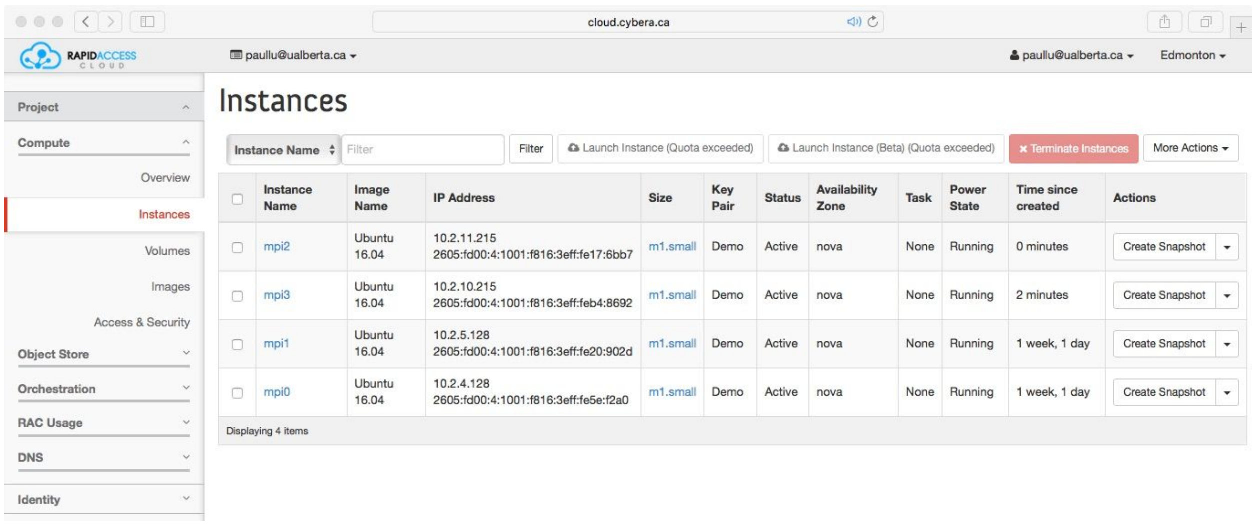
Repeat the above process multiple times to create VMs called `mpi1`, `mpi2`, `mpi3`. You may choose to get `mpi0` and `mpi1` “just right” before creating more VMs to save yourself some work. But, be aware that VMs can be terminated (i.e., deleted, cannot recover) and re-launched easily to start over. If you want to save a VM instance as a backup, use the snapshot feature (details in the Cybera documentation; links at beginning of the document).

Logging Into mpi0 Using a Floating IP Address

Getting a routable IP address is our goal now. The RAC/OpenStack call a routable IP address a “floating IP” address because I can associate it with different VMs by choice (i.e., floating) via the dashboard. Under normal allocation, I get exactly one IP address to use for this purpose.

By default, my VMs only have private IP addresses. This is analogous to the IP addresses given to my computers at home, which live behind a Network Address Translation (NAT) device, namely my home WiFi router. These NATed addresses are commonly in the 10.x.x.x and 192.168.x.x ranges. Devices on the same private IP network can communicate with each other (e.g., using MPI) but not with the outside world, which requires a routable IP address.

Below are my 4 VMs, shown with 10.2.x.x IP addresses. **The IP addresses of your VMs will be different. NOTE:** The specific IP address given to a VM might change each time it is terminated or created. And, due to differences in time when some of these screenshots were taken, you might see some inconsistencies in the IP addresses shown. So, follow the principles below and do not get hung up on the specific IP addresses in the screenshots.

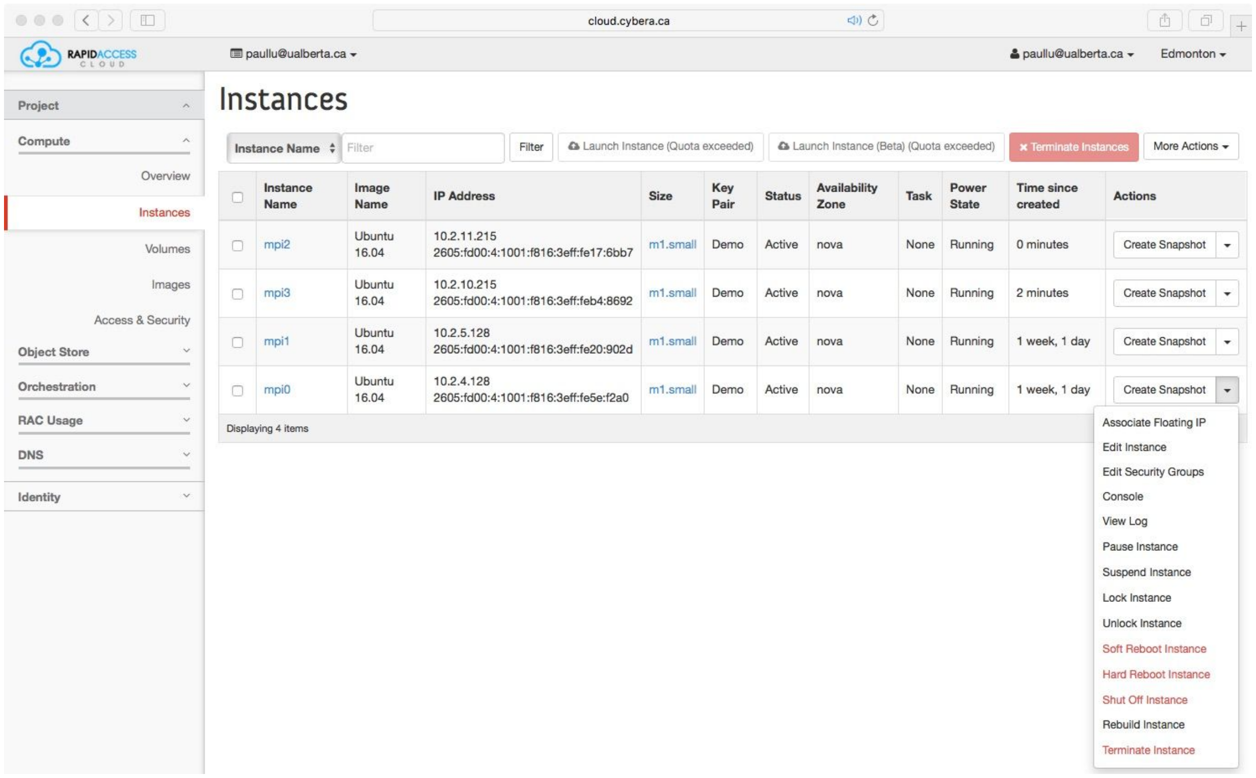


Instances

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
mpi2	Ubuntu 16.04	10.2.11.215 2605:fd00:4:1001:f816:3eff:fe17:6bb7	m1.small	Demo	Active	nova	None	Running	0 minutes	Create Snapshot
mpi3	Ubuntu 16.04	10.2.10.215 2605:fd00:4:1001:f816:3eff:feb4:8692	m1.small	Demo	Active	nova	None	Running	2 minutes	Create Snapshot
mpi1	Ubuntu 16.04	10.2.5.128 2605:fd00:4:1001:f816:3eff:fe20:902d	m1.small	Demo	Active	nova	None	Running	1 week, 1 day	Create Snapshot
mpi0	Ubuntu 16.04	10.2.4.128 2605:fd00:4:1001:f816:3eff:fe5e:f2a0	m1.small	Demo	Active	nova	None	Running	1 week, 1 day	Create Snapshot

Displaying 4 items

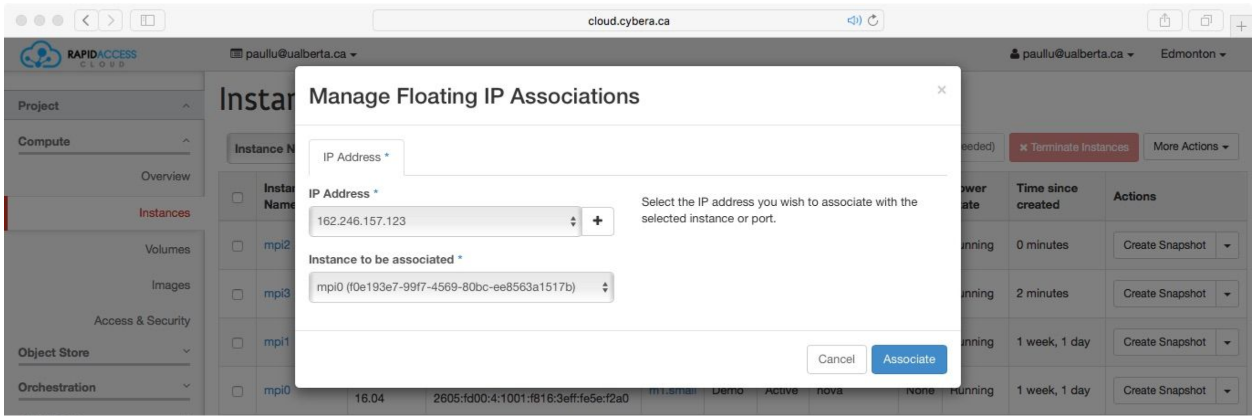
Since there is only one floating, routable IP address available, we will associate it with `mpi0`, the so-called head node. Select the “Associate Floating IP” using the pop-up menu shown.



Instances

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
mpi2	Ubuntu 16.04	10.2.11.215 2605:fd00:4:1001:f816:3eff:fe17:6bb7	m1.small	Demo	Active	nova	None	Running	0 minutes	Create Snapshot
mpi3	Ubuntu 16.04	10.2.10.215 2605:fd00:4:1001:f816:3eff:feb4:8692	m1.small	Demo	Active	nova	None	Running	2 minutes	Create Snapshot
mpi1	Ubuntu 16.04	10.2.5.128 2605:fd00:4:1001:f816:3eff:fe20:902d	m1.small	Demo	Active	nova	None	Running	1 week, 1 day	Create Snapshot
mpi0	Ubuntu 16.04	10.2.4.128 2605:fd00:4:1001:f816:3eff:fe5e:f2a0	m1.small	Demo	Active	nova	None	Running	1 week, 1 day	Create Snapshot

Displaying 4 items



Manage Floating IP Associations

IP Address *: 162.246.157.123

Instance to be associated *: mpi0 (f0e193e7-99f7-4569-80bc-ee8563a1517b)

Select the IP address you wish to associate with the selected instance or port.

If this is the first time you have used your floating IP address, you may have to press on the "+" sign to add the address to the pop-up menu. In the above example, the address 162.246.157.123 was previously used, so it was selected from the pop-up menu. And, the `mpi0` VM instance is also selected before clicking on the "Associate" (blue) button.

The result is that `mpi0` has both an internal IP address (10.2.4.128) and a floating, routable IP address (162.246.157.123).

Now, from your desktop or laptop, you can setup your SSH keys (e.g., using `ssh-agent`) and log into `mpi0` with:

```
% ssh -A ubuntu@162.246.157.123
```

NOTE: The "-A" was added on Nov. 8, 2016. That's what I did in class but forgot to add it in the documentation here.

If it is the first time you have logged into this IP address, you will see output like:

```
The authenticity of host '162.246.157.123 (162.246.157.123)' can't be established.
ECDSA key fingerprint is SHA256:oFCPqM2odWZazctDcYuDgKK4lVAaGJscIOxBFZR98QI.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '162.246.157.123' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 4.4.0-45-generic x86_64)
```

However, if you have used this IP address before (e.g., with a different VM instance), you might see (depending on how you have configured SSH):

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ECDSA key sent by the remote host is
SHA256:oFCPqM2odWZazctDcYuDgKK4lVAaGJscIOxBFZR98QI.
Please contact your system administrator.
Add correct host key in /Users/paullu/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /Users/paullu/.ssh/known_hosts:294
ECDSA host key for 162.246.157.123 has changed and you have requested strict checking.
```

Unless you think there actually is a security problem, the above warning about a possible "man-in-the-middle attack" is mostly likely because you logged into that IP previously and you have now changed the VM instance. SSH remember the cryptographic identity of the first VM you logged into in file `~/.ssh/known_hosts` and checks it each time to detect such an attack.

But, if it is just a change in the VM, you can delete the relevant line (i.e., line 294 in my case) from `~/.ssh/known_ho sts` and try again. (Thanks to Hamidreza): Or it can be removed using the following command `ssh-keygen -R162.246.157.123`

Configuration Within a VM Instance

The Ubuntu image used is fairly minimal. To get the gcc compiler and make, I had to do:

```
% sudo apt-get install gcc make
```

But, once I downloaded mpich, I did the usual untar and configured it with:

```
% ./configure --prefix=/home/ubuntu/mpich-install/ --disable-fortran
--disable-cxx
```

Then, make it, make install it, and add the `mpich-install/bin` directory to your shell's path (default shell is Bash) by changing the last line of `~/.profile` (on all nodes, `mpi0`, `mpi1`, `mpi2`, `mpi3`) to look like:

```
PATH="$HOME/bin:$HOME/.local/bin:$HOME/mpich-install/bin:$PATH"
```

Note that the mpich-install directory needs to be on all nodes, as well as the PATH. Whether you want to build it separately on all VMs (which is OK, but time consuming) or simply copy the directory to each of the other VMs (e.g., using tar, scp; or rsync, if you know how to us it) is up to you. I copied the directory.

Lastly, given the fact that there is no DNS service for your VMs, you will want to edit `/etc/hosts` on all the VMs via (for example) "`sudo vi /etc/hosts`", to look like:

```
127.0.0.1 localhost
10.2.4.128 mpi0
10.2.5.128 mpi1
10.2.11.215 mpi2
10.2.10.215 mpi3

# The following lines are desirable for IPv6
capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
```

```
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

Running MPI programs

Since none of the VMs share a file system, you have to make sure that the MPI binary is on all the nodes in which you want to run. The easiest way to do it is to compile it on `mpi0` and then copy the binary to the other nodes.

Common mistake: A common mistake is to change the MPI program's source code, recompile it on `mpi0`, and forget to copy it to the other nodes. If the binary was never on the other nodes, mpirun/mpiexec will complain about not finding the executable on the other nodes. A more nefarious scenario is that `mpi0` has the latest binary, but the other nodes have the old binary. Then, your MPI program will likely start running, but you will experience weird, hard-to-find bugs. **Recommendation:** Modify your makefile to copy the new binary to all the nodes every time you rebuild.

Recommendation: Use absolute pathnames to refer to your binaries and other files. Unless you set up your per-VM directories "just right", full pathnames are the best way to avoid path-related bugs and problems.

For mpirun/mpiexec to know which VMs to run on, you will want to use a host machine file, which we will just "hosts". There are a variety of ways to do it, but I recommend having a simple textfile called "hosts", which looks like:

```
110.2.5.128
10.2.11.215
10.2.10.215
10.2.4.128
```

Note that "hosts" only contain IP addresses and no machine names (e.g., no `mpi0` label). Also, I put the IP address for `mpi0` as the 4th line, because mpirun/mpiexec uses a round-robin placement algorithm.

Now, with your “cpi” binary on all the VMs, with mpich-install on all the VMs, with `/etc/hosts` set properly, with the “hosts” file ready, you can run your MPI program as follows:

```
% mpirun -np 4 -f hosts /home/ubuntu/cpi
```