



# Guide To Getting Started with MPI/EC2

AN AWS BASED ELASTIC CLOUD COMPUTING PLATFORM

# Contents:



## **Launching**

Launching your own  
EC2 instance



## **Configuring**

Configuring your  
instance



## **Connecting**

Connecting to your  
instance

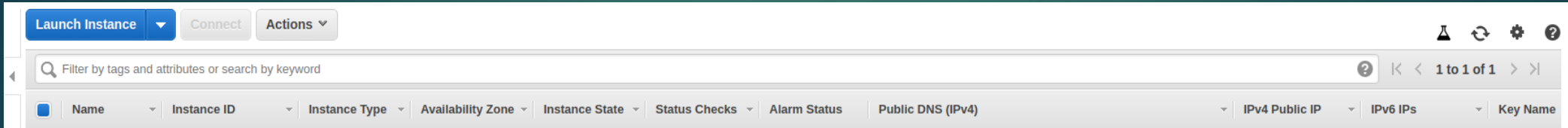


## **Terminating**

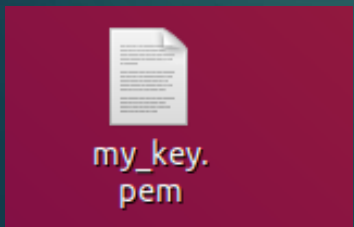
Terminating Instances

# Launching/Configuring:

1. Log in to your Amazon Web Services Account.
2. Head over to your EC2 dashboard and select "Launch Instance"

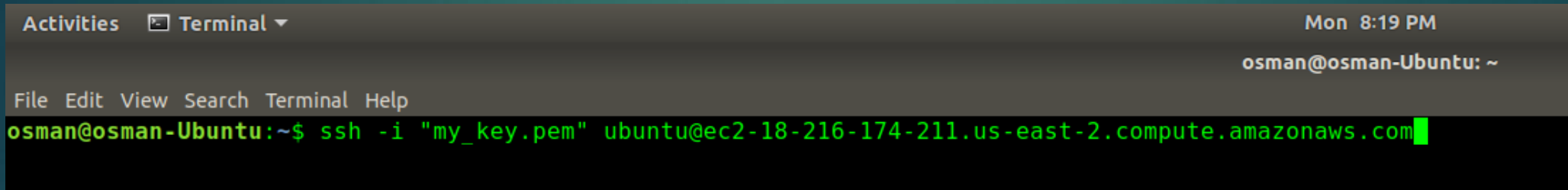


3. Once you have done so will be directed to the instance configuration, where you may choose your instance type (Linux is recommended)
4. You may additionally edit any security group settings, once that is done you can may click on Launch instance.
5. Now you need to create a Key-value pair for your instance which you may use to log into this instance. (Download this file in a safe and accessible location as you won't get a copy of it again if you lose it!). We saved it in our desktop.



# Connecting

1. Connect to your EC2 instance by first launching your terminal and then using the following command:

A screenshot of a Linux terminal window. The title bar shows 'Activities' and 'Terminal'. The top right corner displays 'Mon 8:19 PM' and the user's location 'osman@osman-Ubuntu: ~'. Below the title bar is a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The main terminal area shows the command 'ssh -i "my\_key.pem" ubuntu@ec2-18-216-174-211.us-east-2.compute.amazonaws.com' being entered at the prompt 'osman@osman-Ubuntu:~\$'.

```
Activities  Terminal  Mon 8:19 PM
osman@osman-Ubuntu: ~
File Edit View Search Terminal Help
osman@osman-Ubuntu:~$ ssh -i "my_key.pem" ubuntu@ec2-18-216-174-211.us-east-2.compute.amazonaws.com
```

2. **Please be sure that you are in the same directory as you have stored your Key-value pair in**
3. Once you press enter you will be logged into your EC2 instance.
4. Now we will move onto Installing MPICH2 on our instance to allow us to use the MPI library for our programs.

# Installing MPICH2

- ▶ Once logged in you will see the following screen:

```
- Reduce system reboots and improve kernel security. Activate at:
  https://ubuntu.com/livepatch

76 packages can be updated.
0 updates are security updates.

Last login: Mon May  6 02:06:56 2019 from 108.207.215.64
ubuntu@ip-172-31-20-139:~$
```

- ▶ Now you may install the MPICH2 package by using the following command, once you press enter it will prompt you for installation(Y/N):

```
$ sudo apt install mpich
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  cpp-5 g++-5 gcc-5 gcc-5-base gfortran gfortran-5 hwloc-nox libasan2 libatomic1
  libcc1-0 libcilkrts5 libcr-dev libcr0 libgcc-5-dev libgfortran-5-dev libgfortran3
  libgomp1 libhwloc-plugins libhwloc5 libitm1 liblsan0 libmpich-dev libmpich12 libmpx
  libquadmath0 libstdc++-5-dev libstdc++6 libtsan0 libubsan0
Suggested packages:
  gcc-5-locales g++-5-multilib gcc-5-doc libstdc++6-5-dbg gcc-5-multilib libgcc1-dbg
  libgomp1-dbg libitm1-dbg libatomic1-dbg libasan2-dbg liblsan0-dbg libtsan0-dbg
  libubsan0-dbg libcilkrts5-dbg libmpx0-dbg libquadmath0-dbg gfortran-multilib
  gfortran-doc gfortran-5-multilib gfortran-5-doc libgfortran3-dbg libcr-dkms
  libhwloc-contrib-plugins libstdc++-5-doc bcr-util mpich-doc
The following NEW packages will be installed:
  gfortran gfortran-5 hwloc-nox libcr-dev libcr0 libgfortran-5-dev libhwloc-plugins
  libhwloc5 libmpich-dev libmpich12 mpich
The following packages will be upgraded:
  cpp-5 g++-5 gcc-5 gcc-5-base libasan2 libatomic1 libcc1-0 libcilkrts5 libgcc-5-dev
  libgfortran3 libgomp1 libitm1 liblsan0 libmpx0 libquadmath0 libstdc++-5-dev libstdc
  libtsan0 libubsan0
19 upgraded, 11 newly installed, 0 to remove and 213 not upgraded.
Need to get 41.0 MB of archives.
After this operation, 39.0 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

# Installing StarCluster



- ▶ The next tool we will use to create our virtual cluster is [MIT's StarCluster toolkit](#). StarCluster is a set of tools that automates the process of building and accessing a cluster on EC2. Along with launching the cluster, StarCluster also installs OpenMPI (an implementation of MPI) and other software for writing parallel applications.

- ▶ We may install StarCluster using the following command :

```
$ sudo easy_install StarCluster
```

- ▶ Once installed we move onto configuring our cluster, the configuration file is to be first accessed from our instance using the following command:

```
$vim .starcluster/Config
```

- ▶ The next slide will provide users with a ready-made configuration file that is optimal for our experimental needs.

# StarCluster Configuration

## StarCluster and Mpich2 Configurations:

Following are the configurations that the StarCluster on our EC2 instance is running on.

These values can be changed in the `$vim .starcluster/Config` file under [aws info]:

[aws info]

AWS\_ACCESS\_KEY\_ID = #your\_aws\_access\_key\_id  
AWS\_SECRET\_ACCESS\_KEY = #your\_secret\_access\_key  
AWS\_USER\_ID = #your\_userid

[key mykey]

KEY\_LOCATION = ~/.ssh/mykey.rsa

[cluster smallcluster]

KEYNAME = mykey  
CLUSTER\_SIZE = 8  
CLUSTER\_USER = sgeadmin  
CLUSTER\_SHELL = bash  
NODE\_IMAGE\_ID = ami-3393a45a  
NODE\_INSTANCE\_TYPE = t1.micro  
MASTER\_INSTANCE\_TYPE = t1.micro  
PLUGINS = mpich2

We only need to edit/ uncomment fields under the following headings only

Your AWS access key id can be located on your EC2 dashboard

THE LAST LINE IS THE MOST IMPORTANT  
PLUGIN

Once you have edited this file successfully save and close it.

# Starting/Stopping/Terminating the Cluster



- ▶ Creating a new cluster with the given configuration and name "mycluster":

```
$ starcluster start mycluster
```

OR

- ▶ Accessing a already created cluster with name "cluster-name":

```
$starcluster start -x cluster-name
```

- ▶ To Terminate the cluster use:

```
starcluster terminate mpiclustert
```

- ▶ The next slide goes over how to run a simple MPI program over starcluster using 8 nodes.



```
$ starcluster sshmaster mycluster
```

► Once logged in the following screen should appear:

```
host 'ec2-54-227-107-126.compute-1.amazonaws.com' (54.2  
ECDSA key fingerprint is SHA256:IyZCZ97g/l2+xtMZrg3lDftZEroEQOJ2r49D7X/sZC  
Are you sure you want to continue connecting (yes/no)? yea  
Please type 'yes' or 'no': yes  
Warning: Permanently added 'ec2-54-227-107-126.compute-1.amazonaws.com,54.  
  
███████  
███████  
███████
```

StarCluster Ubuntu 13.04 AMI  
Software Tools for Academics and Researchers (STAR)  
Homepage: <http://star.mit.edu/cluster>  
Documentation: <http://star.mit.edu/cluster/docs/latest>  
Code: <https://github.com/jtriley/StarCluster>  
Mailing list: <http://star.mit.edu/cluster/maillinglist.html>

This AMI Contains:

- \* Open Grid Scheduler (OGS - formerly SGE) queuing system
- \* Condor workload management system
- \* OpenMPI compiled with Open Grid Scheduler support
- \* OpenBLAS - Highly optimized Basic Linear Algebra Routines
- \* NumPy/SciPy linked against OpenBlas
- \* Pandas - Data Analysis Library
- \* IPython 1.1.0 with parallel and notebook support
- \* Julia 0.3pre
- \* and more! (use 'dpkg -l' to show all installed packages)

Open Grid Scheduler/Condor cheat sheet:

- \* qstat/condor\_q - show status of batch jobs
- \* qhost/condor\_status - show status of hosts, queues, and jobs
- \* qsub/condor\_submit - submit batch jobs (e.g. qsub -cwd ./job.sh)
- \* qdel/condor\_rm - delete batch jobs (e.g. qdel 7)
- \* qconf - configure Open Grid Scheduler system

Current System Stats:

System load:	0.09	Processes:	90
Usage of /:	34.6% of 7.84GB	Users logged in:	0
Memory usage:	19%	IP address for eth0:	172.31.17.128
Swap usage:	0%		

<https://landscape.canonical.com/>  
root@master:~# █

- ▶ Now we must log into sgeadmin in order to create any .py files and execute them this can be done by the command:

```
$su - sgeadmin
```

- ▶ Verify the login by the change in the prompt name on the terminal it should look like this:

```
root@master:~# su - sgeadmin
sgeadmin@master:~$
```

- ▶ Now let's run a test MPI program on our cluster in order to verify the working of our program:

1. Create a new .py file hello\_world.py:

```
sgeadmin@master:~$ vim hello_world.py
```

2. Once an empty file opens copy-paste this code:

```
1  #!/usr/bin/env python
2  """
3  Parallel Hello World
4  """
5
6  from mpi4py import MPI
7  import sys
8
9  size = MPI.COMM_WORLD.Get_size()
10 rank = MPI.COMM_WORLD.Get_rank()
11 name = MPI.Get_processor_name()
12
13 sys.stdout.write(
14     "Hello, World! I am process %d of %d on %s.\n"
15     % (rank, size, name))
```

- ▶ Once copy-pasted save and close the file.
- ▶ Now run the program using the following command:

```
sgadmin@master:~$ mpirun -n 8 python hello_world.py
```

- ▶ Once you enter the command your program will run, giving the following output:

```
Hello, World! I am process 0 of 8 on master.  
Hello, World! I am process 2 of 8 on node002.  
Hello, World! I am process 3 of 8 on node003.  
Hello, World! I am process 1 of 8 on node001.  
Hello, World! I am process 4 of 8 on node004.  
Hello, World! I am process 5 of 8 on node005.  
Hello, World! I am process 7 of 8 on node007.  
Hello, World! I am process 6 of 8 on node006.
```

- ▶ This verifies that all 8 nodes on your cluster are responsive and ready for experimentation.

# Detailed Example

A detailed example with explained code regarding all MPI syntax can be found in the following github repository, take a look at it as it also contains a screenrecording that will run you through the process of executing your MPI program :



<https://github.com/oza5/MPI-Matrix-Vector-Multiplication>