Introduction to AWS EC2

Running Your First MPI Cluster

Alan L. Nunes

https://github.com/alan-lira/tutorials/introduction_to_aws_ec2

Last updated on May 30, 2023

Key Pairs

- A key pair, consisting of a public key and a private key, is a set of security credentials that you use to prove your identity when connecting to an Amazon EC2 instance.

    - Amazon EC2 stores the public key on your instance, and you store the private key.
    - For Linux instances, the private key allows you to securely SSH into your instance.

➔ Anyone who possesses your private key can connect to your instances, so it's important that you store your private key in a secure place.

Note: Amazon recommends (seems more like an imposition) running the following command on the pem key to preventing it from being publicly viewable:

```
chmod 400 private-key.pem
```

Where,
private-key.pem: is the local path for the private key associated with the allocated instances

Security Groups

- A security group acts as a virtual firewall for your EC2 instances to control incoming and outgoing traffic:
    - Inbound rules control the incoming traffic to your instance;
    - Outbound rules control the outgoing traffic from your instance.

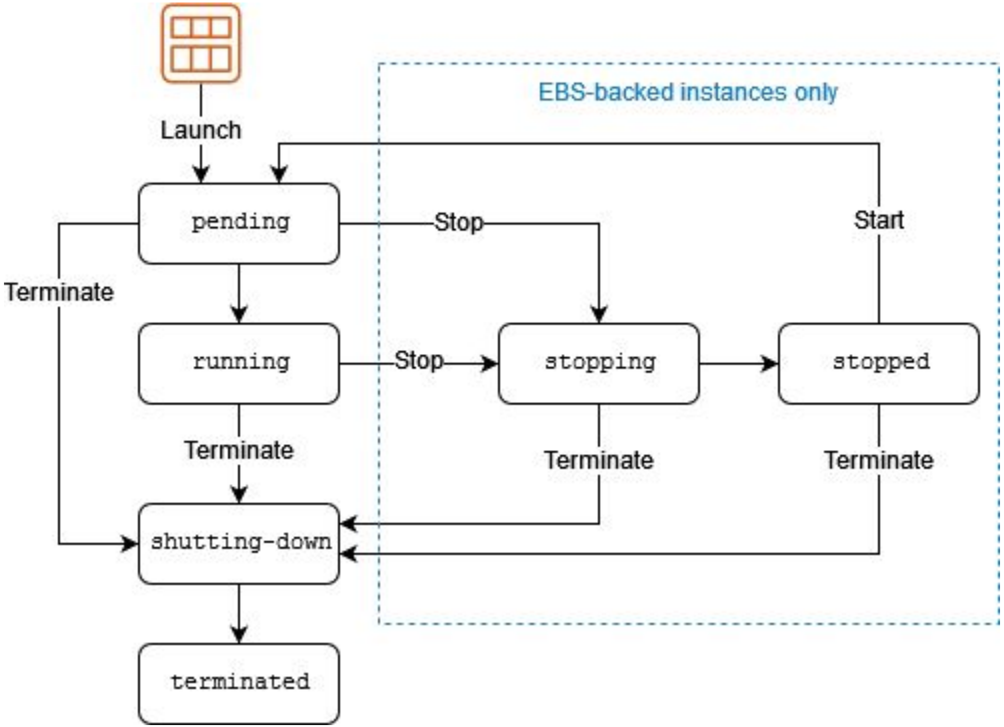
✓ When you launch an instance, you can specify one or more security groups:

- If you don't specify a security group, Amazon EC2 uses the default security group.

✓ You can add rules to each security group that allow traffic to or from its associated instances:

- The rules for a security group can be modified at any time.
- New and modified rules are automatically applied to all instances that are associated with the security group.

EC2 Instances Life Cycle



EC2 Instances Life Cycle

Instance state	Description	Instance usage billing
`pending`	The instance is preparing to enter the `running` state. An instance enters the `pending` state when it is launched or when it is started after being in the `stopped` state.	Not billed
`running`	The instance is running and ready for use.	Billed
`stopping`	The instance is preparing to be stopped.	Not billed
`stopped`	The instance is shut down and cannot be used. The instance can be started at any time.	Not billed
`shutting-down`	The instance is preparing to be terminated.	Not billed
`terminated`	The instance has been permanently deleted and cannot be started.	Not billed **Note** Reserved Instances that applied to terminated instances are billed until the end of their term according to their payment option. For more information, see Reserved Instances

EC2 Instances Life Cycle

Note

The table indicates billing for instance usage only. Some AWS resources, such as Amazon EBS volumes and Elastic IP addresses, incur charges regardless of the instance's state. For more information, see Avoiding Unexpected Charges in the *AWS Billing User Guide*.

Note

Rebooting an instance doesn't start a new instance billing period because the instance stays in the `running` state.

Differences Between Reboot, Stop, Hibernate, and Terminate

Characteristic	Reboot	Stop/start (Amazon EBS-backed instances only)	Hibernate (Amazon EBS-backed instances only)	Terminate
Host computer	The instance stays on the same host computer	We move the instance to a new host computer (though in some cases, it remains on the current host).	We move the instance to a new host computer (though in some cases, it remains on the current host).	None
Private and public IPv4 addresses	These addresses stay the same	The instance keeps its private IPv4 address. The instance gets a new public IPv4 address, unless it has an Elastic IP address, which doesn't change during a stop/start.	The instance keeps its private IPv4 address. The instance gets a new public IPv4 address, unless it has an Elastic IP address, which doesn't change during a stop/start.	None
Elastic IP addresses (IPv4)	The Elastic IP address remains associated with the instance	The Elastic IP address remains associated with the instance	The Elastic IP address remains associated with the instance	The Elastic IP address is disassociated from the instance
IPv6 address	The address stays the same	The instance keeps its IPv6 address	The instance keeps its IPv6 address	None
Instance store volumes	The data is preserved	The data is erased	The data is erased	The data is erased
Root device volume	The volume is preserved	The volume is preserved	The volume is preserved	The volume is deleted by default
RAM (contents of memory)	The RAM is erased	The RAM is erased	The RAM is saved to a file on the root volume	The RAM is erased
Billing	The instance billing hour doesn't change.	You stop incurring charges for an instance as soon as its state changes to `stopping`. Each time an instance transitions from `stopped` to `running`, we start a new instance billing period, billing a minimum of one minute every time you start your instance.	You incur charges while the instance is in the `stopping` state, but stop incurring charges when the instance is in the `stopped` state. Each time an instance transitions from `stopped` to `running`, we start a new instance billing period, billing a minimum of one minute every time you start your instance.	You stop incurring charges for an instance as soon as its state changes to `shutting-down`.

Spot Instance Interruptions

- You can launch Spot Instances on spare EC2 capacity for steep discounts in exchange for returning them when Amazon EC2 needs the capacity back (Spot Instance interruption).

➔ When Amazon EC2 interrupts a Spot Instance, it either terminates (default), stops, or hibernates the instance, depending on what you specified when you created the Spot request.

➔ Demand for Spot Instances can vary significantly from moment to moment, and the availability of Spot Instances can also vary significantly depending on how many unused EC2 instances are available.

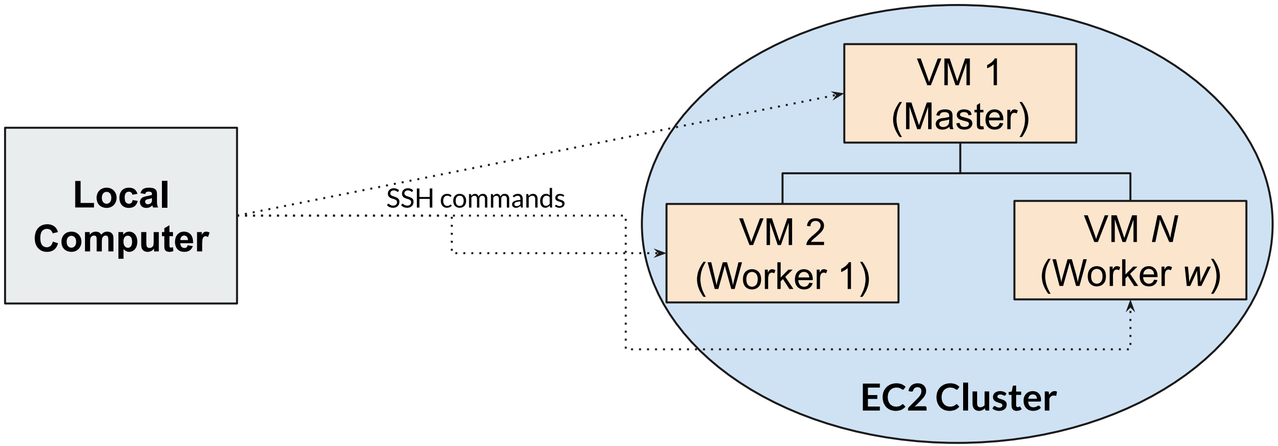
➔ It is always possible that your Spot Instance might be interrupted.

Hands-On: Configuring the Cluster and Executing Apps

Configuring the Cluster

Configuring the Cluster

Objective: Configure a cluster formed by at least two EC2 nodes (1 Master and 1...*w* Workers) through remote (SSH) commands sent by the local computer.



Configuring the Cluster

Assumptions:

1. The EC2 instances are already running (previously requested through the AWS management console);

2. The EC2 instances belongs to a security group with the following inbound and outbound rules (insecure for production environments):
    a. Inbound Rules → IP Version: IPv4 | Type: All Traffic | Protocol: All Protocols | Port Interval: All Port Intervals | Source: 0.0.0.0/0 (All IPs Range)
    b. Outbound Rules → IP Version: IPv4 | Type: All Traffic | Protocol: All Protocols | Port Interval: All Port Intervals | Source: 0.0.0.0/0 (All IPs Range)

3. The EC2 instances are reachable via SSH + private key (.pem).

Configuring the Cluster

Step 1 (Remotely Executed on All Nodes):

- Include the 'private-key' file content to the 'id_rsa' file (.ssh/id_rsa).
- Change 'id_rsa' octal permissions to 600, i.e., only changeable by the user.

```
ssh -i private-key.pem username@ip "echo '$(cat private-key.pem)' > .ssh/id_rsa; chmod 600 .ssh/id_rsa"
```

Where,

private-key.pem: is the local path for the private key associated with the allocated instances

username: is the instance’s username (e.g., ubuntu)

ip: is the instance’s public IPv4

Configuring the Cluster

Step 2 (Remotely Executed on All Nodes):

- Update the 'available packages' list.
- Install 'OpenSSH'.
- Install 'OpenMPI'.

```
ssh -i private-key.pem username@ip "sudo apt-get update && sudo apt-get install openssh-server openssh-client && sudo apt-get install libopenmpi-dev -y && mpiexec --version"
```

Where,

private-key.pem: is the local path for the private key associated with the allocated instances

username: is the instance’s username (e.g., ubuntu)

ip: is the instance’s public IPv4

Configuring the Cluster

Step 3 (Remotely Executed on Master Node):

- Generate Master's Public/Private RSA key pair.

```
ssh -i private-key.pem master_username@master_ip "sudo ssh-keygen -q -t rsa -N '' -f .ssh/id_rsa <<<y 2>&1
>/dev/null"
```

Where,

private-key.pem: is the local path for the private key associated with the allocated instances

master_username: is the Master instance’s username (e.g., ubuntu)

master_ip: is the Master instance’s public IPv4

Configuring the Cluster

Step 4 (Remotely Executed on All Nodes):

- Include Master's Public RSA key into all nodes' 'authorized_keys' file (.ssh/authorized_keys).

```
ssh -i private-key.pem username@ip "echo $(ssh -i private-key.pem master_username@master_ip "cat
.ssh/id_rsa.pub") | sudo tee -a .ssh/authorized_keys"
```

Where,

private-key.pem: is the local path for the private key associated with the allocated instances

username: is the instance's username (e.g., ubuntu)

ip: is the instance's public IPv4

master_username: is the Master instance's username (e.g., ubuntu)

master_ip: is the Master instance's public IPv4

Configuring the Cluster

Step 5 (Remotely Executed on Master Node):

- Include all Workers' Public SSH Key into Master's 'known_hosts' file (.ssh/known_hosts).

```
ssh -i private-key.pem master_username@master_ip "echo '$(ssh -i private-key.pem
worker_username@worker_ip "ssh-keyscan -H worker_ip | grep -o '^[^#]*'")' | sudo tee -a .ssh/known_hosts"
```

Where,

private-key.pem: is the local path for the private key associated with the allocated instances

master_username: is the Master instance's username (e.g., ubuntu)

master_ip: is the Master instance's public IPv4

worker_username: is the Worker instance's username (e.g., ubuntu)

worker_ip: is the Worker instance's public IPv4

Configuring the Cluster

Step 6 (Remotely Executed on Master Node):

- Generate the 'hostfile' file, including all workers IPv4 addresses and number of MPI processes per IP.

```
ssh -i private-key.pem master_username@master_ip "touch hostfile"
```

```
ssh -i private-key.pem master_username@master_ip "echo 'worker_ip slots=worker_num_slots' | sudo tee -a hostfile"
```

Where,

private-key.pem: is the local path for the private key associated with the allocated instances

master_username: is the Master instance's username (e.g., ubuntu)

master_ip: is the Master instance's public IPv4

hostfile: is the remote path for the hostfile in the Master instance

worker_ip: is the Worker instance's public IPv4

worker_num_slots: is the number of MPI processes to be spawned in the respective worker_ip (e.g., 1)

Executing an MPI Application (Distributed)

Executing an MPI Application (Distributed)

Step 7 (Locally Executed):

- Copy the MPI application file to all nodes.

```
scp -i private-key.pem -r mpi_app_path username@ip:~
```

Where,

private-key.pem: is the local path for the private key associated with the allocated instances

mpi_app_path: is the local path for the mpi application source code (.c) that will be send to the instances

username: is the instance's username (e.g., ubuntu)

ip: is the instance's public IPv4

Executing an MPI Application (Distributed)

Step 8 (Remotely Executed on All Nodes):

- Compile the MPI application file on all nodes.

```
ssh -i private-key.pem username@ip "mpicc mpi_app_path -o compiled_mpi_app_name"
```

Where,

private-key.pem: is the local path for the private key associated with the allocated instances

username: is the instance’s username (e.g., ubuntu)

ip: is the instance’s public IPv4

mpi_app_path: is the remote path for the mpi application source code (.c) received by the instance

compiled_mpi_app_name: is the remote path name destination for the compiled mpi application

Executing an MPI Application (Distributed)

Step 9 (Remotely Executed on Master Node):

- Launch the MPI application on master.

```
ssh -i private-key.pem master_username@master_ip "mpiexec -np N -hostfile hostfile ./compiled_mpi_app_name"
```

Where,

private-key.pem: is the local path for the private key associated with the allocated instances

master_username: is the Master instance's username (e.g., ubuntu)

master_ip: is the Master instance's public IPv4

N: is the number of MPI processes to be used in the application (e.g., 2)

hostfile: is the remote path for the hostfile in the Master instance

compiled_mpi_app_name: is the remote path name for the compiled mpi application in all instances (same path)

Executing an OpenMP Application (Locally on Master)

Executing an OpenMP Application (Locally on Master)

Step 10 (Locally Executed):

- Copy the OpenMP application file to master.

```
scp -i private-key.pem -r openmp_app_path master_username@master_ip:~
```

Where,

private-key.pem: is the local path for the private key associated with the allocated instances

openmp_app_path: is the local path for the openmp application source code (.c) that will be send to the Master instance

master_username: is the Master instance's username (e.g., ubuntu)

master_ip: is the Master instance's public IPv4

Executing an OpenMP Application (Locally on Master)

Step 11 (Remotely Executed on Master Node):

- Compile the OpenMP application file on master.

```
ssh -i private-key.pem master_username@master_ip "gcc -fopenmp openmp_app_path -o
compiled_openmp_app_name"
```

Where,

private-key.pem: is the local path for the private key associated with the allocated instances

master_username: is the Master instance’s username (e.g., ubuntu)

master_ip: is the Master instance’s public IPv4

openmp_app_path: is the remote path for the openmp application source code (.c) received by the Master instance

compiled_openmp_app_name: is the remote path name destination for the compiled openmp application

Executing an OpenMP Application (Locally on Master)

Step 12 (Remotely Executed on Master Node):

- Launch the OpenMP application on master.

```
ssh -i private-key.pem master_username@master_ip "export OMP_NUM_THREADS=num_threads;
./compiled_openmp_app_name"
```

Where,

private-key.pem: is the local path for the private key associated with the allocated instances

master_username: is the Master instance’s username (e.g., ubuntu)

master_ip: is the Master instance’s public IPv4

num_threads: is the number of threads to be used in the application (e.g., 5)

compiled_openmp_app_name: is the remote path name for the compiled openmp application

Executing an MPI + OpenMP Application (Distributed)

Executing an MPI + OpenMP Application (Distributed)

Step 13 (Locally Executed):

- Copy the MPI+OpenMP application file to all nodes.

```
scp -i private-key.pem -r mpi_openmp_app_path username@ip:~
```

Where,

private-key.pem: is the local path for the private key associated with the allocated instances

mpi_openmp_app_path: is the local path for the mpi + openmp application source code (.c) that will be send to the instances

username: is the instance's username (e.g., ubuntu)

ip: is the instance's public IPv4

Executing an MPI + OpenMP Application (Distributed)

Step 14 (Remotely Executed on All Nodes):

- Compile the MPI+OpenMP application file on all nodes.

```
ssh -i private-key.pem username@ip "mpicc -fopenmp mpi_openmp_app_path  -o compiled_mpi_openmp_app_name"
```

Where,

private-key.pem: is the local path for the private key associated with the allocated instances

username: is the instance’s username (e.g., ubuntu)

ip: is the instance’s public IPv4

mpi_openmp_app_path: is the remote path for the mpi + openmp application source code (.c) received by the instance

compiled_mpi_openmp_app_name: is the remote path name destination for the compiled mpi + openmp application

Executing an MPI + OpenMP Application (Distributed)

Step 15 (Remotely Executed on Master Node):

- Launch the MPI+OpenMP application on master.

```
ssh -i private-key.pem master_username@master_ip "export OMP_NUM_THREADS=num_threads; mpiexec -np N -hostfile hostfile -x OMP_NUM_THREADS ./compiled_mpi_openmp_app_name”
```

Where,

private-key.pem: is the local path for the private key associated with the allocated instances

master_username: is the Master instance’s username (e.g., ubuntu)

master_ip: is the Master instance’s public IPv4

num_threads: is the number of threads to be used in the application (e.g., 5)

N: is the number of MPI processes to be used in the application (e.g., 2)

hostfile: is the remote path for the hostfile in the Master instance

compiled_mpi_openmp_app_name: is the remote path name for the compiled mpi + openmp application in all instances (same path)

Final Step

Final Step

TERMINATE THE ALLOCATED RESOURCES!

Thank you!