EC2 (ELASCTIC COMPUTE CLOUD)

Amazon Elastic Compute Cloud (Amazon EC2)

Amazon EC2 is AWS primary web service that provides resizable compute capacity in the cloud.

Amazon EC2 allows you to acquire compute through the launching of virtual servers called **instances.** Instance is nothing but a Virtual Server.

Instance Types:

The instance type defines the virtual hardware supporting an Amazon EC2 instance. There are many instance types available, based on the following dimensions:

- General purpose
- Compute Optimized (vCPUs)
- GPU Compute
- Memory Optimized
- Storage Optimized

General Purpose: General purpose instance family provides a balance of compute, memory, and network resources, and it is a good choice for many applications.

Compute Optimized (vCPUs): Compute Optimized instances are optimized for compute-intensive workloads and delivers high performance computing, batch processing.

GPU Compute: GPU Compute instances are next generation of general purpose GPU computing instances. We can use GPU instances for 3D visualizations, graphics-intensive remote workstation, 3D rendering, application streaming, video encoding, Machine/Deep learning, high performance computing and other server-side graphics workloads.

Memory Optimized: Memory Optimized category instances are most suitable for high performance databases, distributed memory caches, in-memory analytics, large-scale, enterprise-class, and In-memory applications.

Storage Optimized:

Optimized category instances are most suitable for low latency, very high random I/O performance, high sequential read throughput and provide high IOPS and NoSQL databases like Cassandra, MongoDB, Redis and In-memory databases.

Compute optimized	For workloads requiring significant processing
Memory optimized	For memory-intensive workloads
Storage optimized	For workloads requiring high amounts of fast SSD storage

GPU-based instances	Intended for graphics and general-purpose GPU compute workloads

Instance launch pricing Options:

- On-Demand Instances
- Reserved Instances
- Spot Instances

On-Demand Instances:

The price **per hour** for each instance type published on the AWS website represents the price for On-Demand Instances.

- On-Demand is most flexible pricing option, as it doesn't requires up-front commitment.
- We will have control over when the instance is launched and when it is terminated.
- Suitable for unpredictable workloads.

Reserved Instances:

When purchasing a reserved instance we have to specify the instance type and Availability Zone for that Reserved Instance and achieves a lower effective hourly price for that instance for the duration of the reservation. You can select duration from 1 Yr to 3 yrs.

- We have three payment options for Reserved Instances.
  - **All Upfront**—Pay for the entire reservation up front. There is no monthly charge for the customer during the term.
  - **Partial Upfront**—Pay a portion of the reservation charge up front and the rest in monthly installments for the duration of the term.
  - **No Upfront—**Pay the entire reservation charge in monthly installments for the duration of the term.
- We can save up to 75 percent over on-demand hourly rate if we reserve instance through Reserved Option.

Spot Instances:

For workloads that are not time critical and are tolerant of interruption, Spot Instances offer the greatest discount.

- We can specify the price they are willing to pay for a certain instance type.
- When the bid price is above the current Spot price, we'll get the requested instance.

- These instances will operate like all other Amazon EC2 instances, and the customer will only pay the Spot price for the hours that instance(s) run.

The instances will run until:

- Till we terminate them manually.
- The Spot price goes above our bid price.
- There is not enough unused capacity to meet the demand for Spot Instances.
- If Amazon EC2 needs to terminate a Spot Instance, the instance will receive a termination notice providing a **two-minute warning prior to termination**.
- If we terminate Instance manually we have to pay for Partial hours, if amazon terminates we will not get charged for partial hours.

Tenancy Options:

Shared Tenancy: Shared tenancy is the default tenancy model for all Amazon EC2 instances. A single host machine may house instances from different customers. (One host may share with multiple customers).

Dedicated Instances: Dedicated Instances run on hardware that's dedicated to a single customer. As a customer runs more Dedicated Instances, more underlying hardware may be dedicated to their account.

Dedicated Host: An Amazon EC2 Dedicated Host is a physical server with Amazon EC2 instance capacity fully dedicated to a single customer's use. We will get complete control over which specific host runs an instance at launch.

Placement Groups: A placement group is a logical grouping of instances within a single Availability Zone.

- Placement groups enable applications to participate in a low-latency, 10 Gbps network.
- Recommended for applications that benefit from low network latency, high network throughput, or both.
- Only certain types of instances can be launched in a placement group.
- A placement group can't span multiple Availability Zones.
- The name you specify for a placement group must be unique within your AWS account.
- AWS recommend homogenous instances within placement groups.
- You can't merge placement groups.
- You can't move an existing instance into a placement group.

Amazon Machine Images (AMIs)

The Amazon Machine Image (AMI) defines the initial software that will be on an instance when it is launched.

- The Operating System (OS) and its configuration
- The initial state of any patches
- Application or system software

All AMIs are based on x86 OSs, either Linux or Windows.

We can launch instances from four options

1. Published by AWS
2. AWS Marketplace
3. Generated from existing Instance
4. Uploaded Virtual Servers

Accessing an Instance: We can access our Instances by Using Public DNS, Public IP address and Elastic IP addresses.

Public DNS: When we launch instance, we will get one Public DNS associated for that instance.

- Public DNS will generate automatically. We can't specify
- We can found this information in Instance description
- We cannot transfer this Public DNS to another instance.
- We will get public DNS when the instance is in running state.

Public IP:

- When we launch instance, we will get one Public IP address also.
- AWS will allocate this address, no option to select specific IP.
- This is unique on the Internet.

Elastic IP

- An Elastic IP address is a static IPv4 address designed for dynamic cloud computing. An Elastic IP address is associated with your AWS account.
- To use an EIP address, we have to generate one to our AWS account, and then associate it with your instance or a network interface.
- We can disassociate an EIP address from a resource, and reassociate it with a different resource.
- A disassociated EIP address remains allocated to your account until you manually release it.

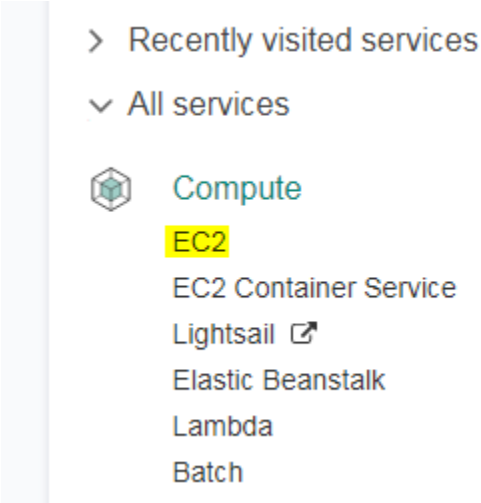
Steps to get EIP Address:

1. Login to AWS account and navigate to Amazon EC2 console.
2. In the navigation pane, choose **Elastic IPs**.
3. Choose Allocate new address.
4. Select Allocate. Close the confirmation screen.

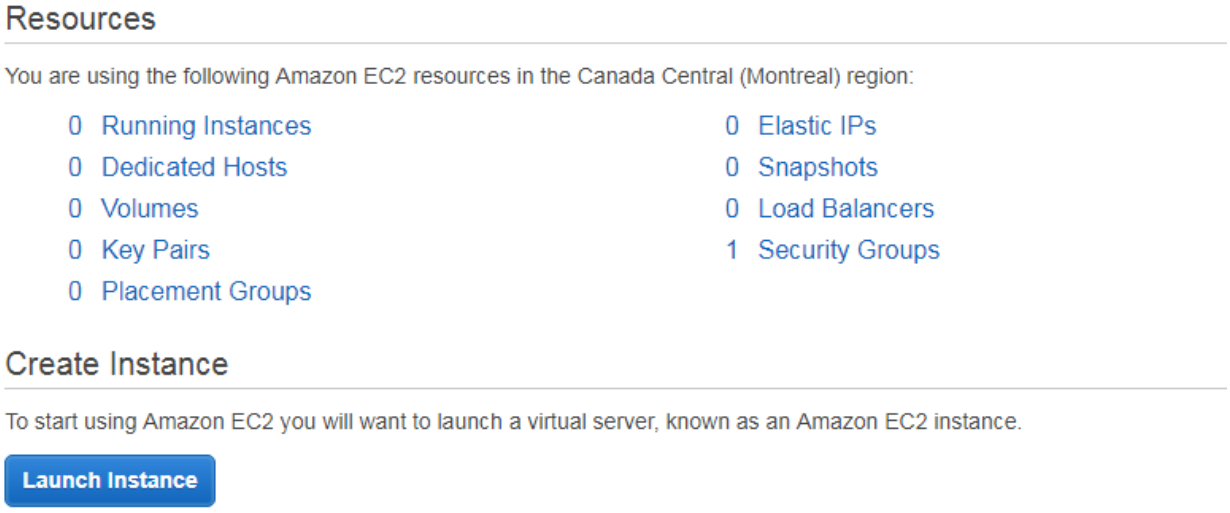
Enhanced networking: reduces the impact of virtualization on network performance by enabling a capability called Single Root I/O Virtualization (SR-IOV). This results in more Packets per Second, lower latency, and less jitter.

Instance launch process:

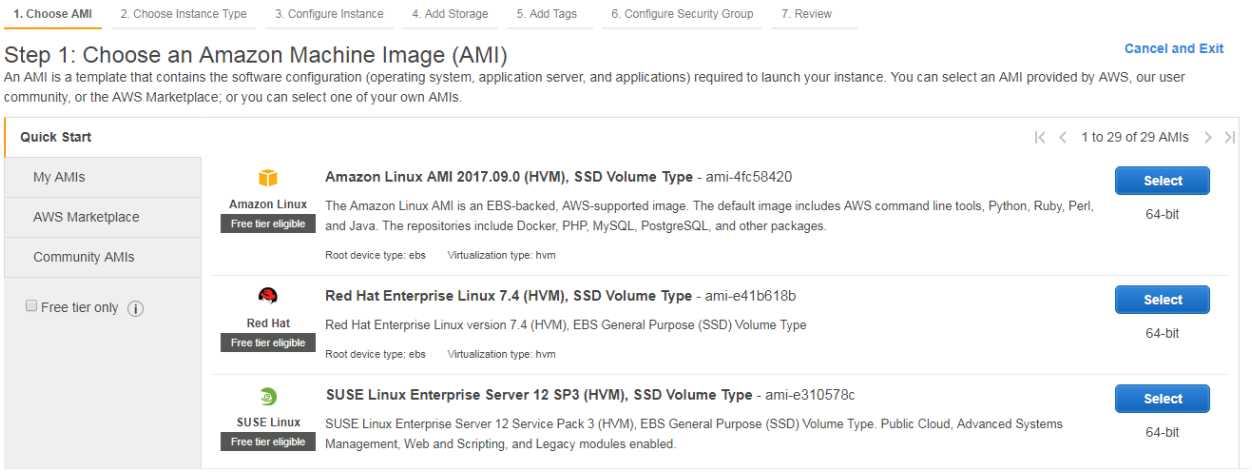
Login to Your AWS Account, Select and switch to the required Region and find **EC2 under Compute Section**.



Select the Launch instance option and it will launch an instance launch wizard.



I want to launch an Amazon Linux AMI, so selecting Amazon Linux AMI from the Quick Start menu.

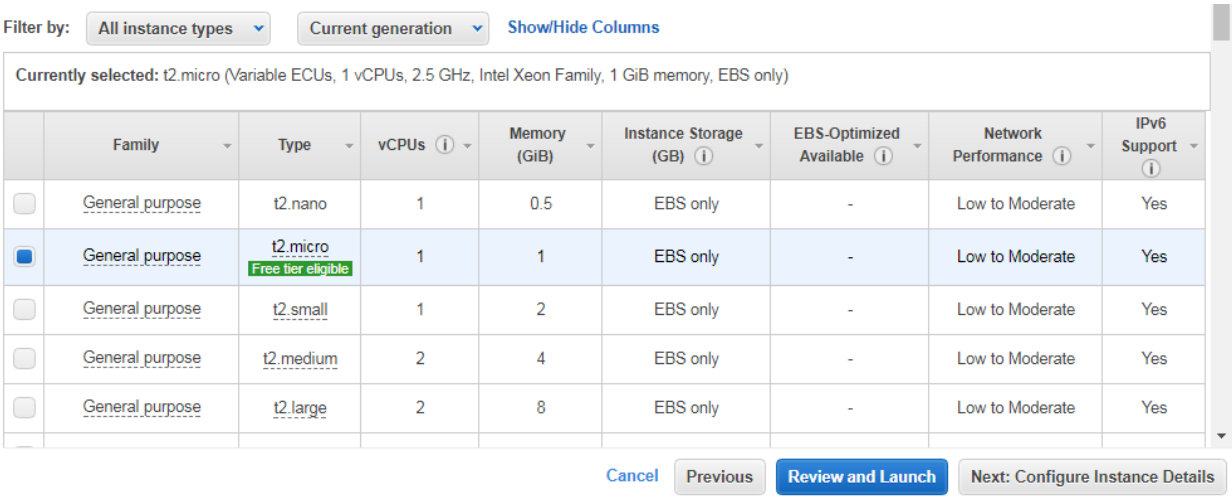


- We have Windows and Linux operating systems available here in Quick start option
- Along with the Quick Start option, you can also spin up your instances using the AWS Marketplace and the Community AMIs section. Both these options contains list of customized AMIs that have been created by either third-party companies or by developers and can be used for a variety of purposes.

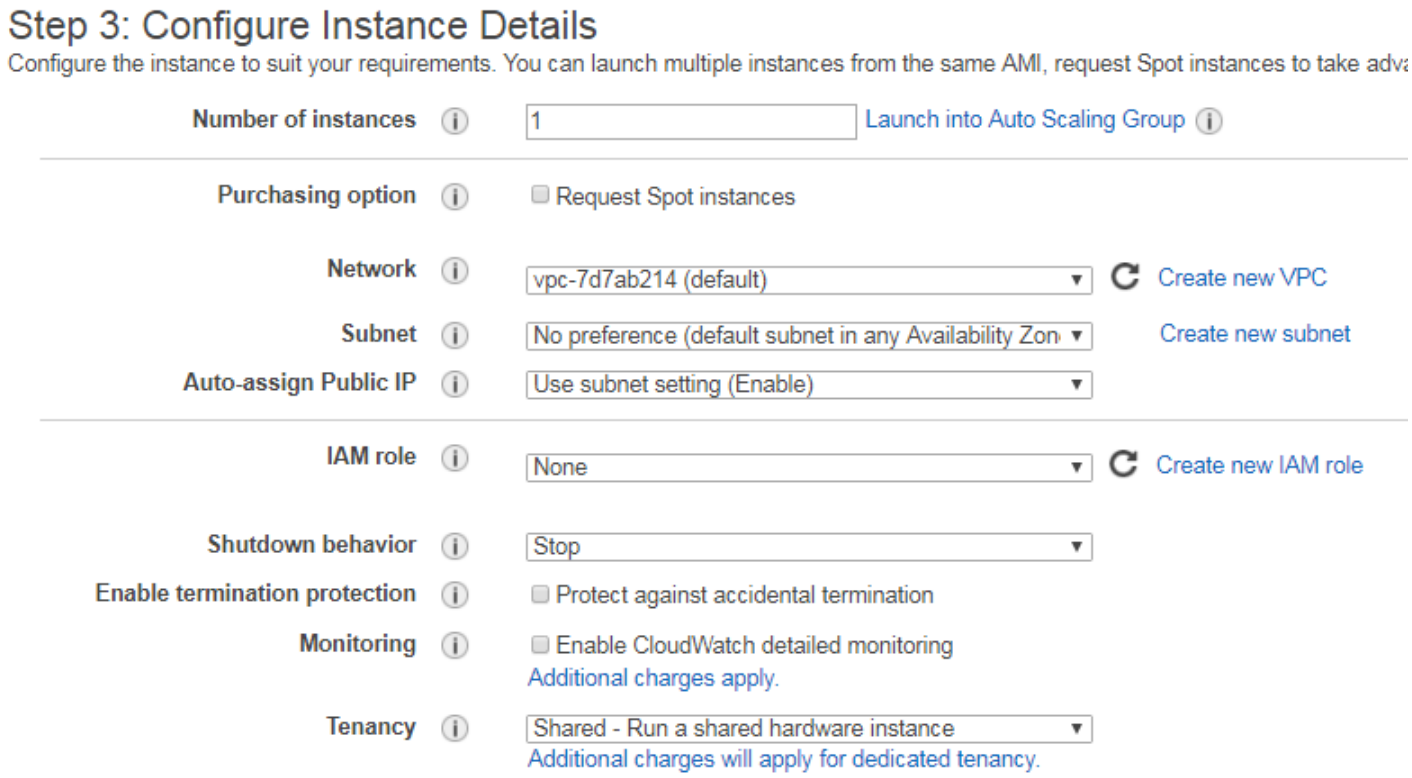
Choose an instance type

In the next step, we have to select the instance type as per our requirements. You can filter instances according to their families.

We can use the general purpose t2.micro instance type, which is comes under the free tier eligibility and configuration is 1 vCPU and 1 GB of RAM.



Configure instance details



Here is Step 3, we have multiple options,

Number of instances: You can specify how many instances the wizard should launch using this field. By default, the value is always set to one single instance.

Purchasing option: We can this instance under spot instances request. For now let's leave this option.

Network: Select the default **Virtual Private Cloud** (**VPC**) network that is displayed in the dropdown list. We can even go ahead and create a new VPC network for this instance, but we will leave and will see VPC in later chapters.

Subnet: select the **Subnet** in which you wish to deploy your new instance.

You can either choose to have AWS select and deploy your instance in a particular subnet from an available list or you can select a particular choice of subnet on your own.

Auto-assign Public IP: Each instance that you launch will be assigned a Public IP. We are going to use this public IP to connect to our Instance over Internet.

IAM role: You can additionally select a particular IAM role to be associated with your instance.

Shutdown behavior: This option allows us to select whether the instance should stop or be terminated when issued a shutdown request. In this case, we have opted for the instance to stop when it is issued a shutdown command.

Enable termination protection: Select this option in case you wish to protect your instance against accidental deletions. It adds additional step for instance termination. If, we enable this option, we need to manually Disable to terminate the instance.

Monitoring: By default, AWS will monitor few basic parameters about your instance for free, but if you wish to have an in-depth insight into your instance's performance, then select the **Enable CloudWatch detailed monitoring** option. But you'll get charged for detailed monitoring.

Tenancy: We can choose to run our instances on physical servers fully dedicated for your use. The use of host tenancy will request to launch instances onto dedicated hosts.

Bootstrapping We can configure instances and install applications programmatically when an instance is launched. The process of providing code to be run on an instance at launch is called bootstrapping.

On Linux instances this can be shell script, and on Windows instances this can be a batch style script or a PowerShell script.

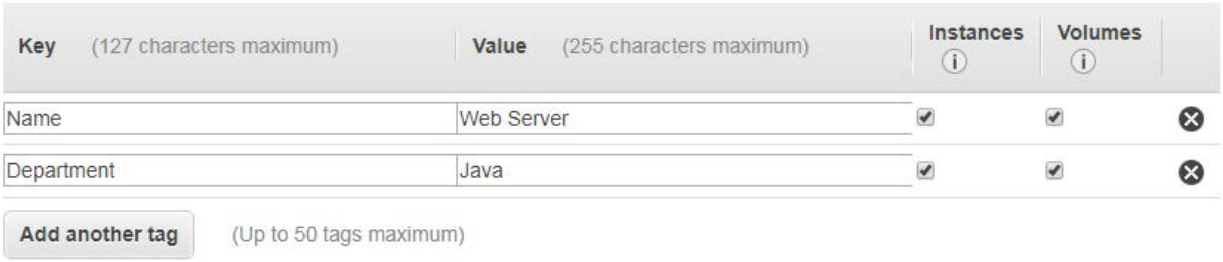
Step 4: Add Storage

We can add EBS volumes to your instances. To add new volumes, simply click on the Add New Volume button. This will provide you with options to provide the size of the new volume along with its mount points. There is an 8 GB volume already attached to our instance. This is the t2.micro instance's root volume.

- Try to keep the volume size under 30 GB, It'll comes under free tier eligibility.
- We can create volumes and attach to instance even after instance launch also.

Step 5: Add Tags

Tags are normal key-value pairs. We can manage our AWS resources with Tags options. We can create maximum of 50 tags per Instance.

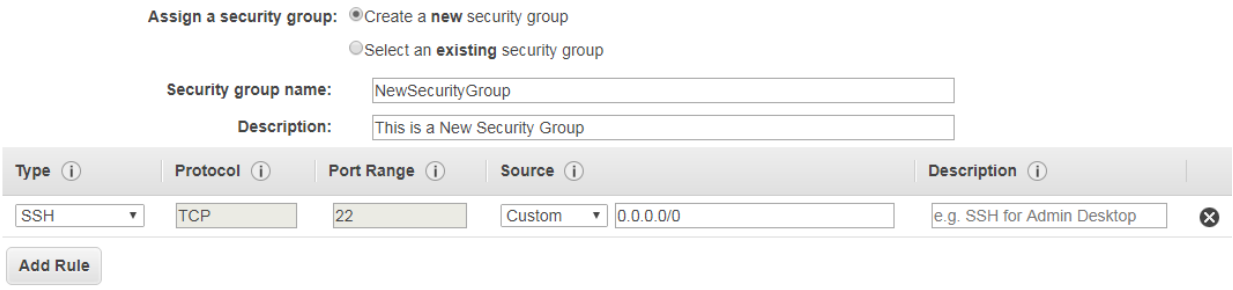


Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for our instance. We can add rules to allow specific traffic to reach our instance.

For example, if you want to set up a web server and allow Internet traffic to reach our instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. We can create a new security group or select from an existing one.

Select the **Create a new security group** option and enter the suitable Security group name and Description.



- You need to open SSH to Connect Linux machines, RDP for Windows machines. HTTP and HTTPS if webservers.
- We can give 0.0.0.0/0 to connect this instance from any network and subnet.
- We can select custom option and give the particular Network's public IP, then the service will be available for that particular network only.

Some Important points about Security Groups:

- You can create up to 500 security groups for each Amazon VPC.
- You can add up to 50 inbound and 50 outbound rules to each security group. If you need to apply more than 100 rules to an instance, you can associate up to five security groups with each network interface.

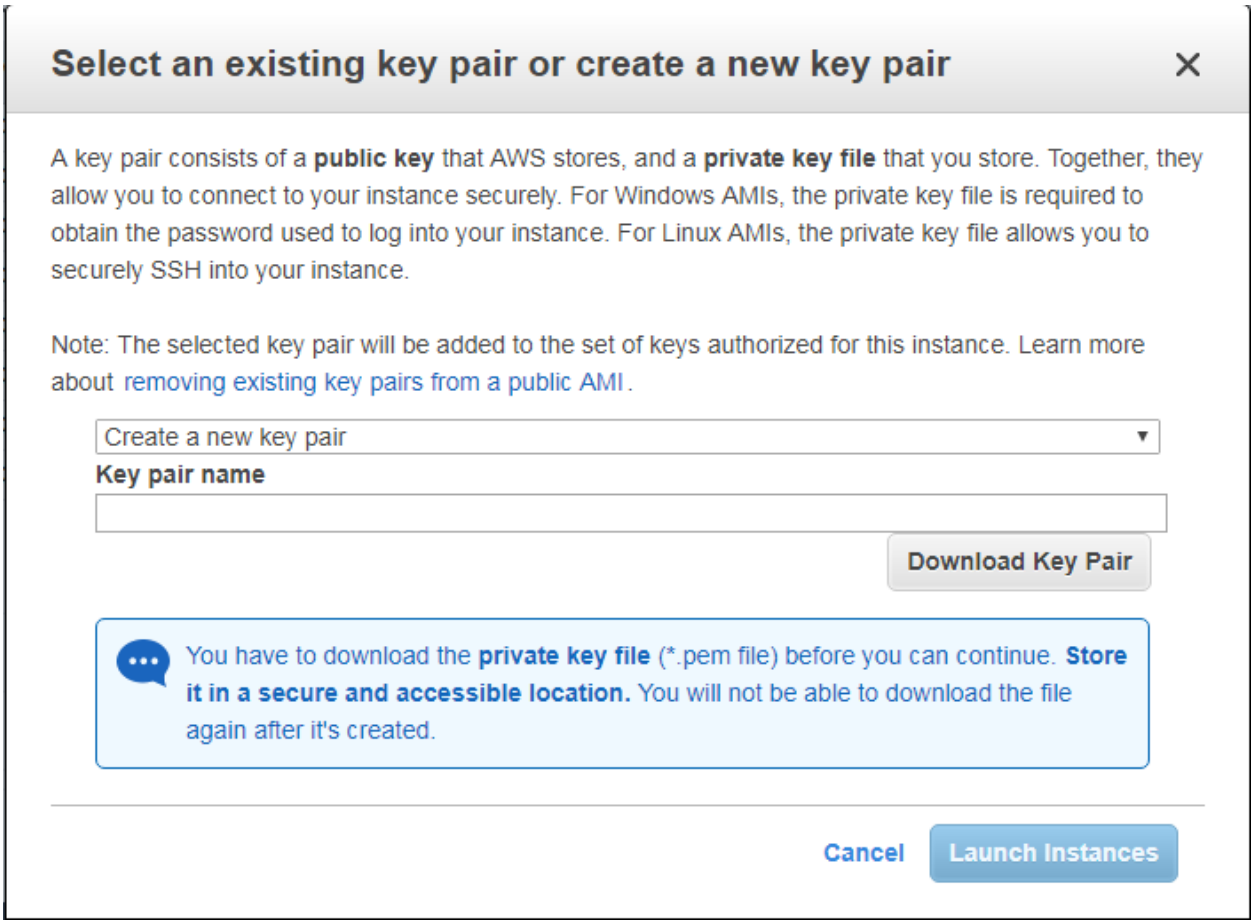
- You can specify allow rules, but not deny rules. This is an important difference between security groups and ACLs.
- By default, no inbound traffic is allowed until you add inbound rules to the security group.
- By default, new security groups have an outbound rule that allows all outbound traffic.
- Security groups are **stateful**. This means that responses to allowed inbound traffic are allowed to flow outbound regardless of outbound rules and vice versa.
- You can change the security groups with which an instance is associated after launch, and the changes will take effect immediately

Step 7: Review Instance Launch

Here in step 3, we will get review screen. We will get complete summary of our instance's configuration details, including the AMI details, instance type selected, instance details, and so on. If all the details are correct, then simply go and click on the Launch option.

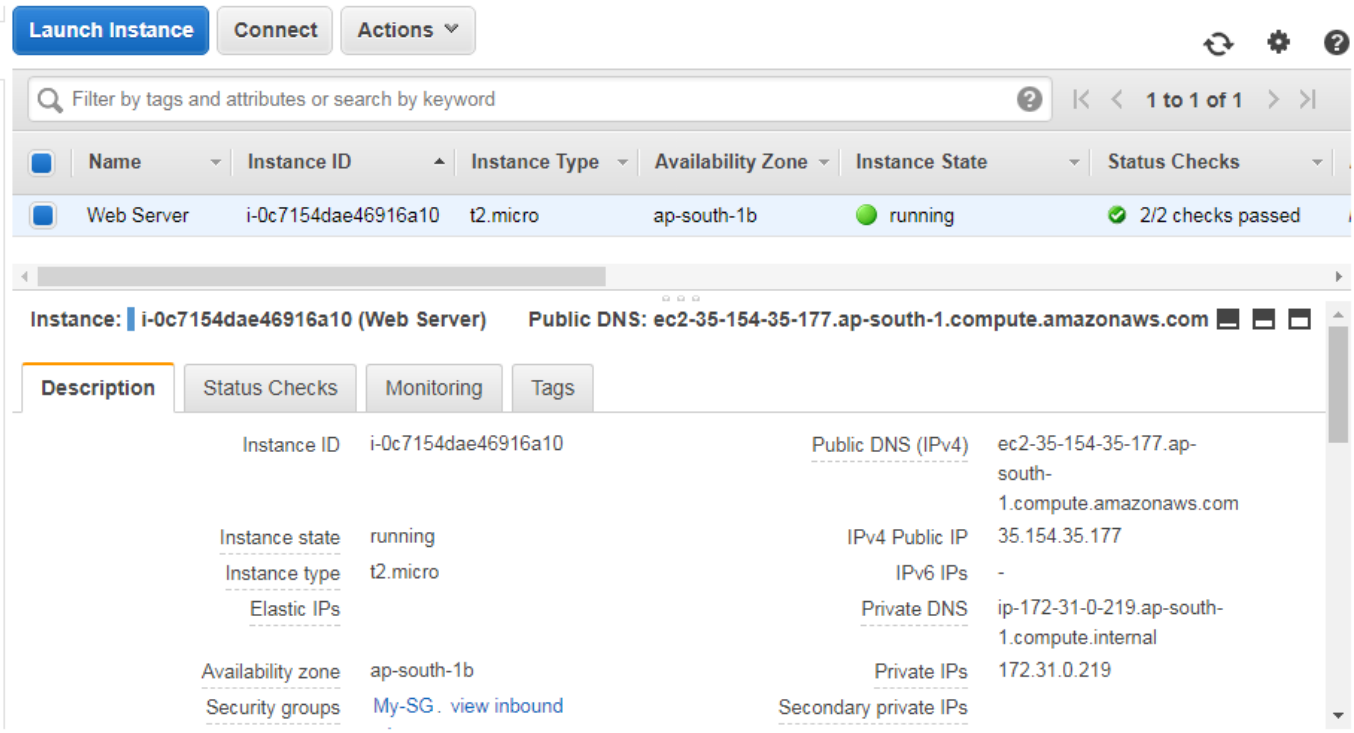
Then we have to associate a key pair to our instance.

A key pair is basically a combination of a public and a private key, which is used to encrypt and decrypt your instance's login info. AWS generates the key pair for you which you need to download and save locally to your computer.



Once a key pair is created and associated with an instance, we need to use that key pair itself to access the instance. We will not be able to download this key pair again so, save it in a secure location.

Select the **Create a new key pair** option from the dropdown list and provide a suitable name for your key pair as well. Click on the **Download Key Pair** option to download the **.PEM file**. Once completed, select the **Launch Instance** option.



- The dashboard provides all of the information about our instance. We can view instance's ID, instance type, IP information, AZ, Security Group, and a whole lot more info.
- We can also obtain instance's health information using the Status Checks tab and the Monitoring tab.
- We can perform power operations on your instance such as start, stop, reboot, and terminate using the Actions tab located in the preceding instance table.

Connecting to Instance:

Once the instance is launched we have multiple options to connect to the instance. Mostly we can use **PuTTY** to connect Linux machines and **Remote Desktop** Feature for Windows Machine.

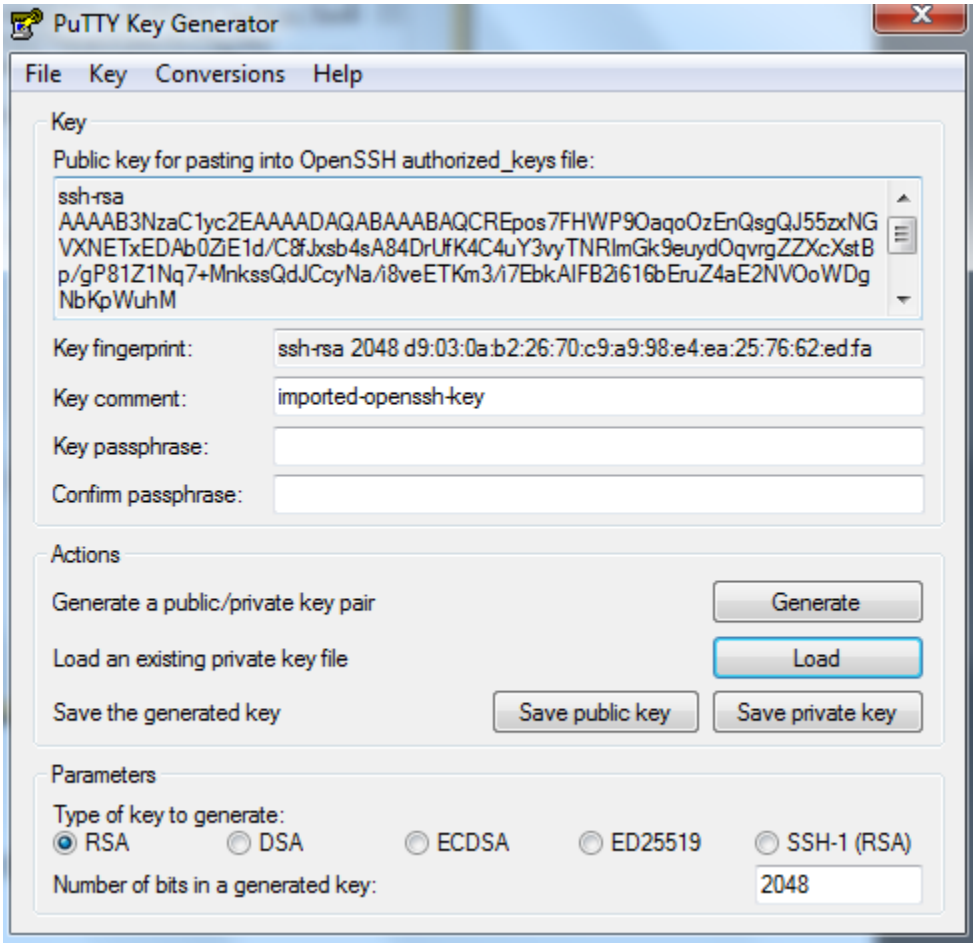
As we launched Linux machine, here we are going to see PuTTY option now.

PuTTY is basically an SSH and telnet client that can be used to connect to remote Linux instances. But before you get working on Putty, we need a tool called **PuttyGen** to convert the PEM file to PPK (Putty Private Key).

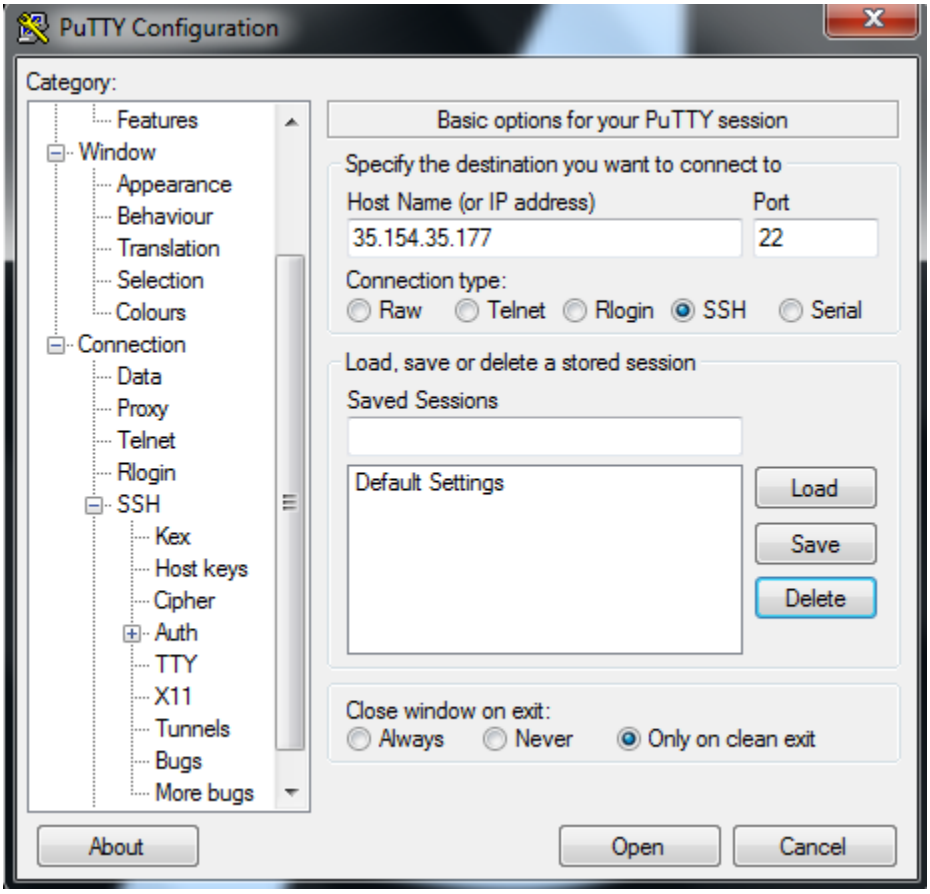
We can download the Putty.exe and PuttyGen.exe from the below URL:

https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

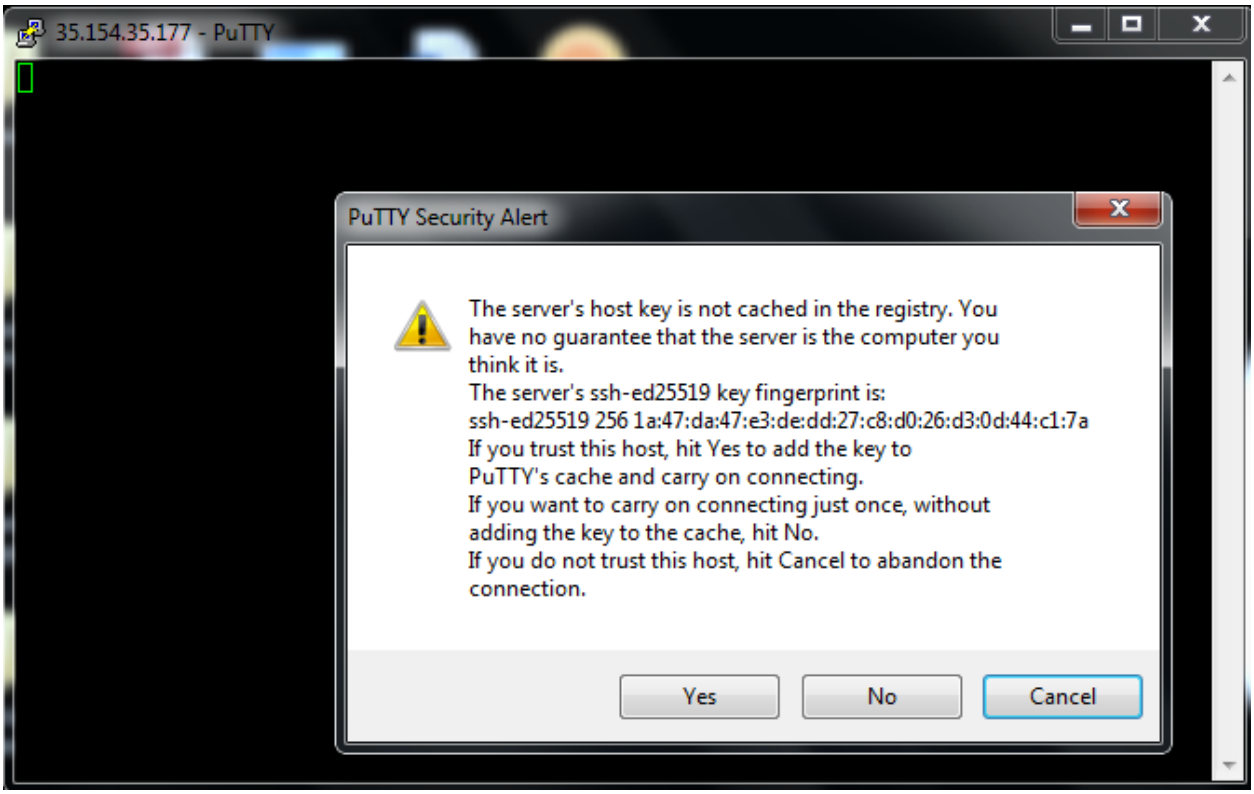
1. Download and install the latest copy of Putty and PuttyGen on local computer.
2. Launch PuttyGen and select the Load button and browse the downloaded Pem file (Which is created at the time of Instance launch).



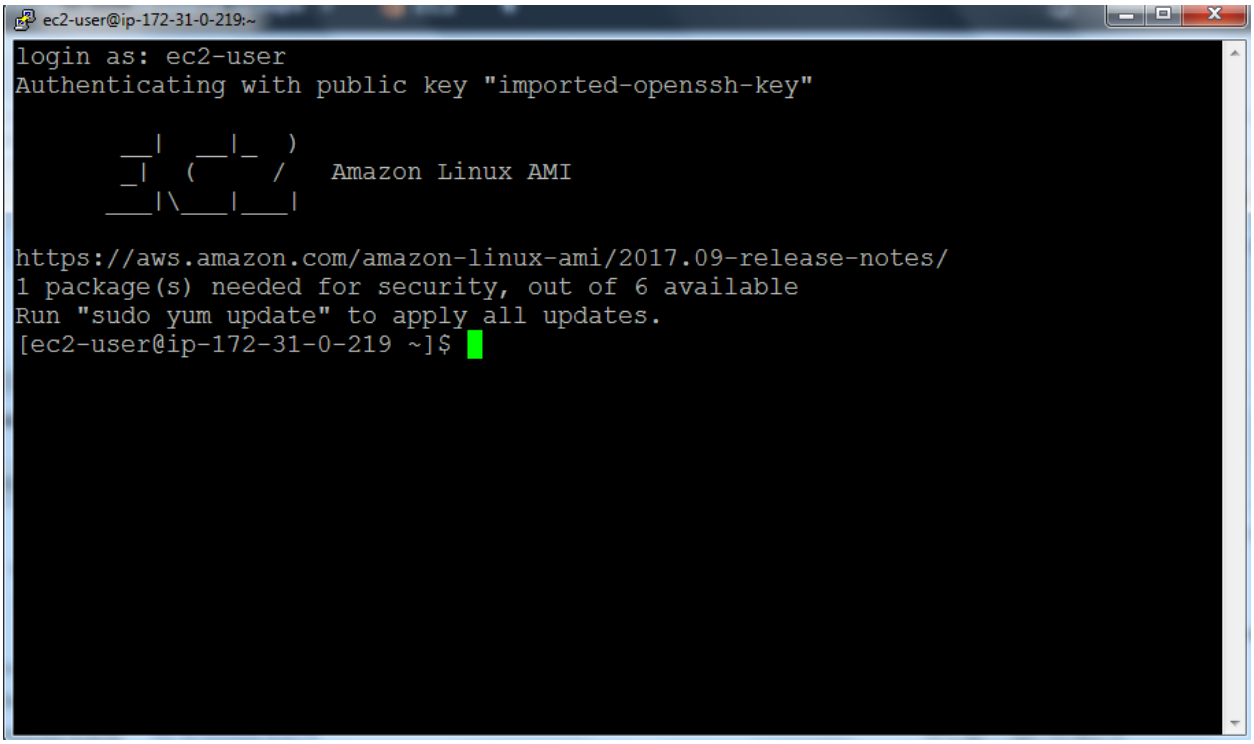
3. Once pem file is loaded, Select **"Save private key"** option.
    a. PuttyGen will prompt you with a warning message that you are saving this key without a passphrase and would you like to continue, Select **YES.**
4. Provide a name and save the new file (*.PPK) at a secure location. You can use this PPK file to connect to your instance using Putty
5. Please note down the **public IP address/ public DNS** of your instance.
6. Now open the **Putty** and enter the public IP in Host Name field and make sure to enter Port **22**



7. In Putty, under **Category pane**, expand the **SSH** option and then select **Auth**, then browse and upload the recently saved PPK file in the **Private key file for authentication** field. Once uploaded, click on Open to establish a connection to instance.
8. Give yes for on the Putty Security Alert.



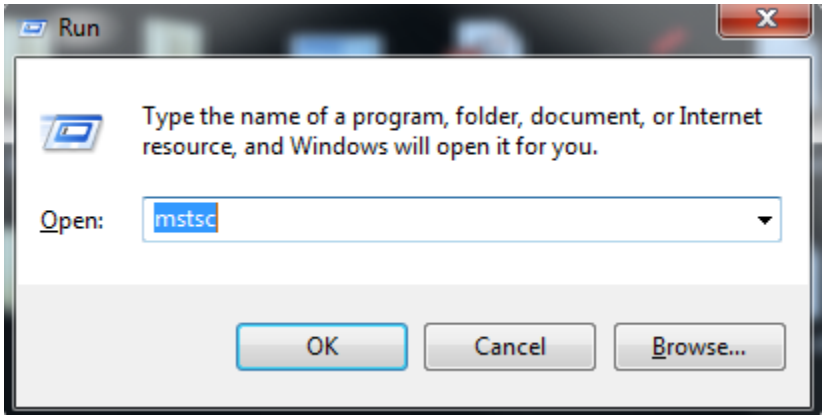
9. In the Putty terminal window, provide the user name for your Amazon Linux instance (ec2-user) and hit the *Enter* key. Now we have connected to our first instance and it is ready for use
10. Each Linux instance type launches with a default Linux system user account. For Amazon Linux, the user name is ec2-user. For RHEL, the user name is **ec2-user** or **root**. For Ubuntu, the user name is **ubuntu** or **root**. For Centos, the user name is **centos**. For Fedora, the user name is **ec2-user**. For SUSE, the user name is **ec2-user** or **root**. Otherwise, if **ec2-user** and **root** don't work, check with your AMI provider.



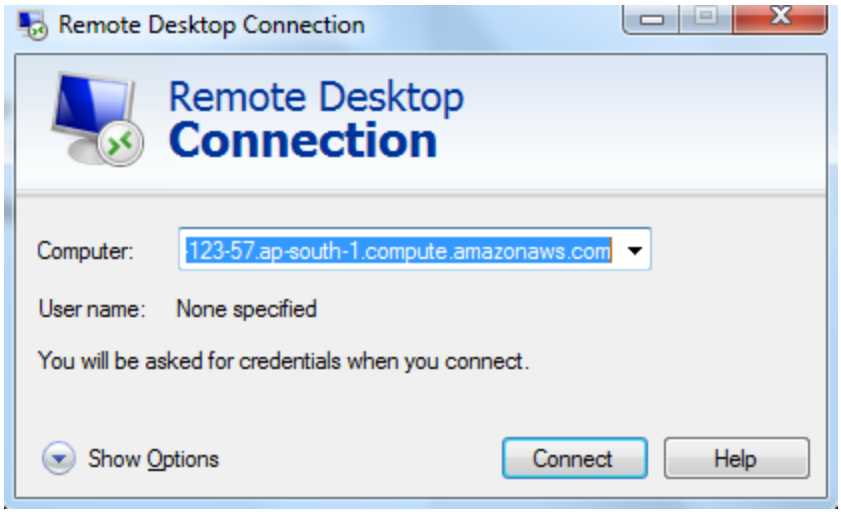
For RHEL-based AMIs (Redhat), the user name is either **root or the ec2-user**, and for Ubuntu-based AMIs, the user name is generally **Ubuntu** itself.

11 **To connect to Windows Instanc**e we have to use Remote Desktop Connection application.

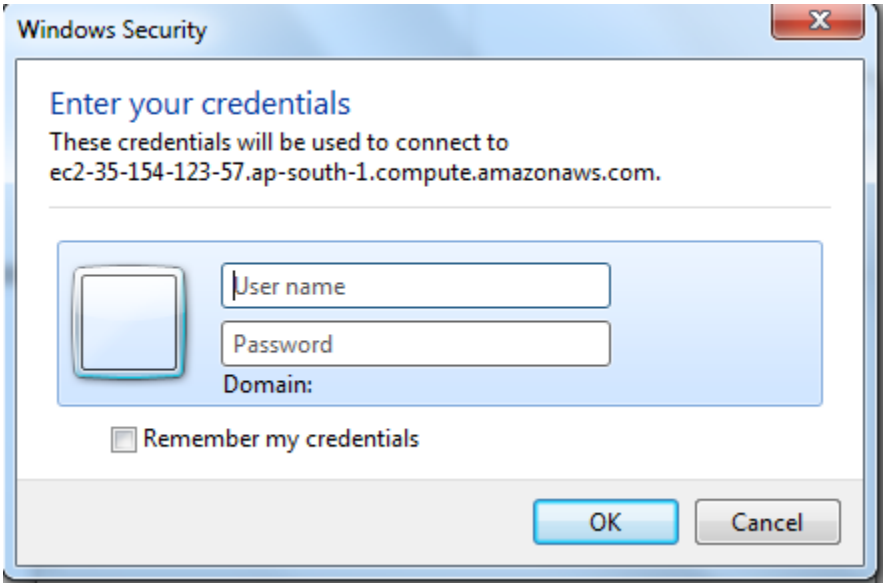
12 Open Run and enter **mstsc** and press enter



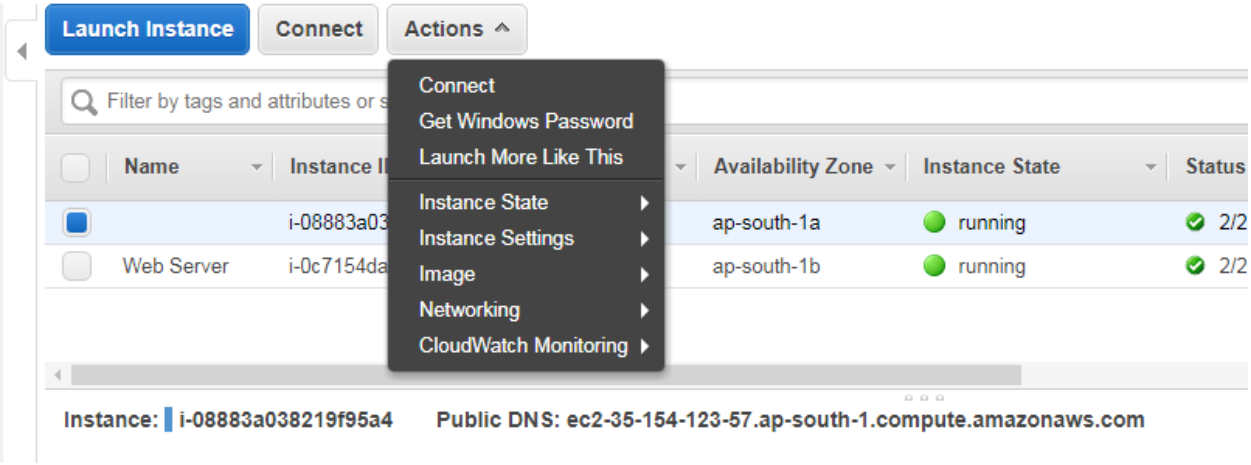
13 Note the public DNS/IP of the windows instance and enter it computer field and click on Connect.



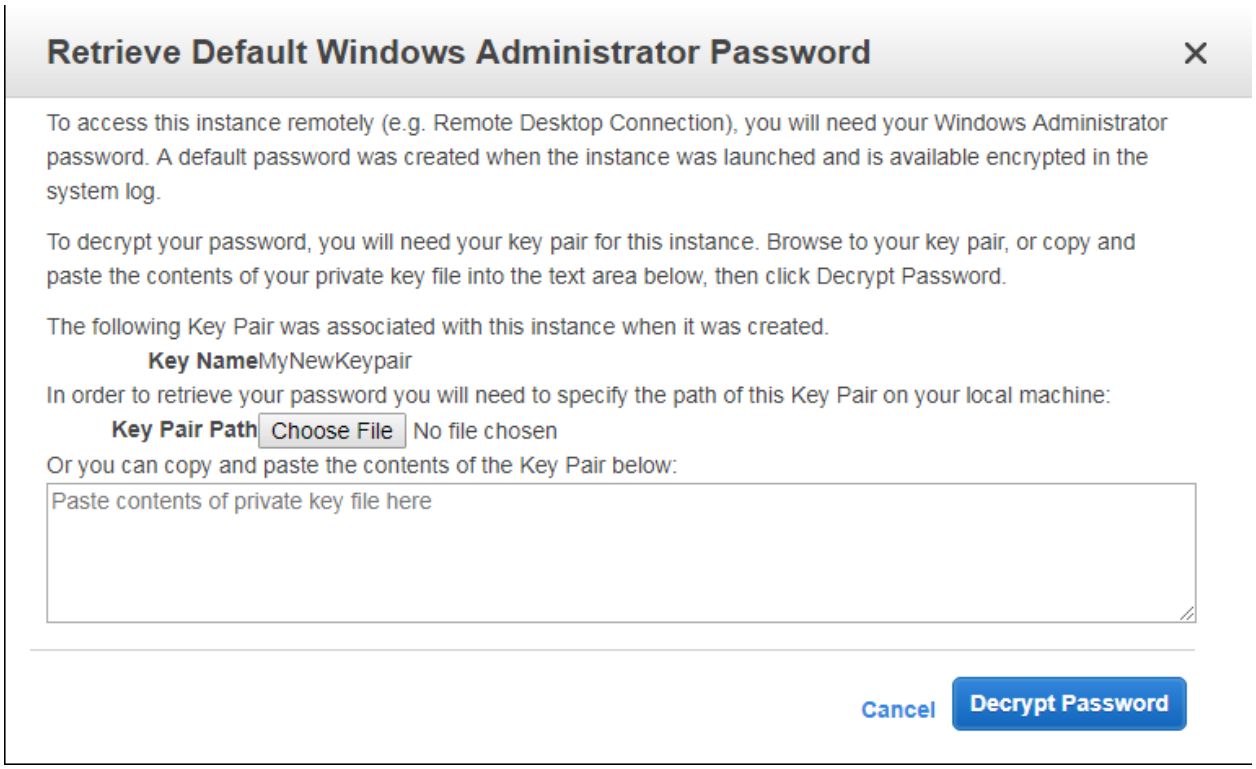
14 Now, It will ask you to enter the username and password to login to the instance.

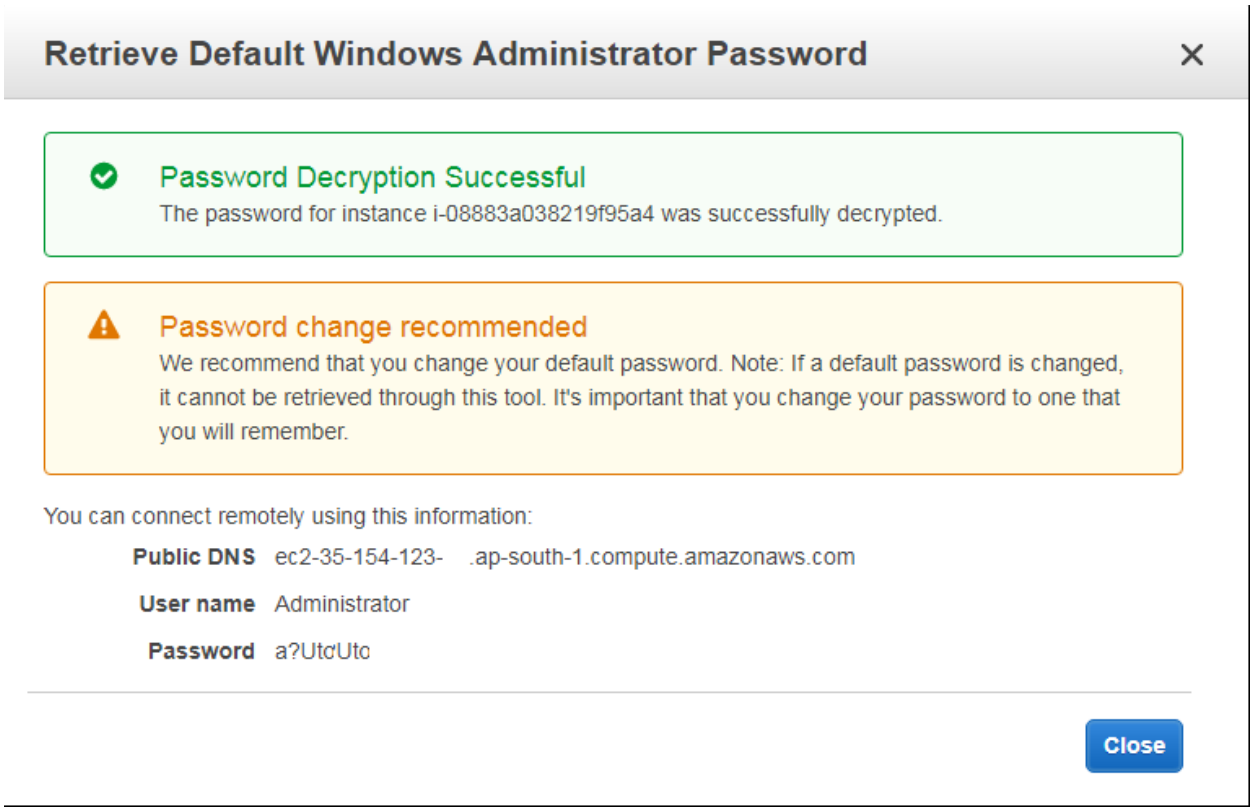


15 To get the Username and password to login to the instance we have get it from EC2 console.

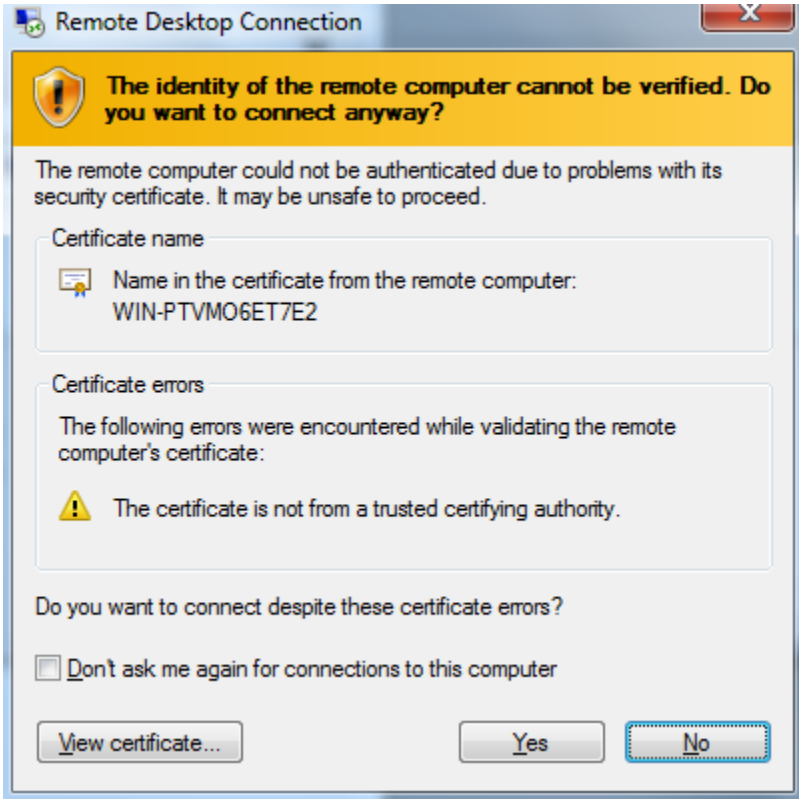


16 Select the instance which you want to get the UN & PWD. Go to Actions and select the "Get Windows Password", then browse the PEM file and select "Decrypt Password" button.

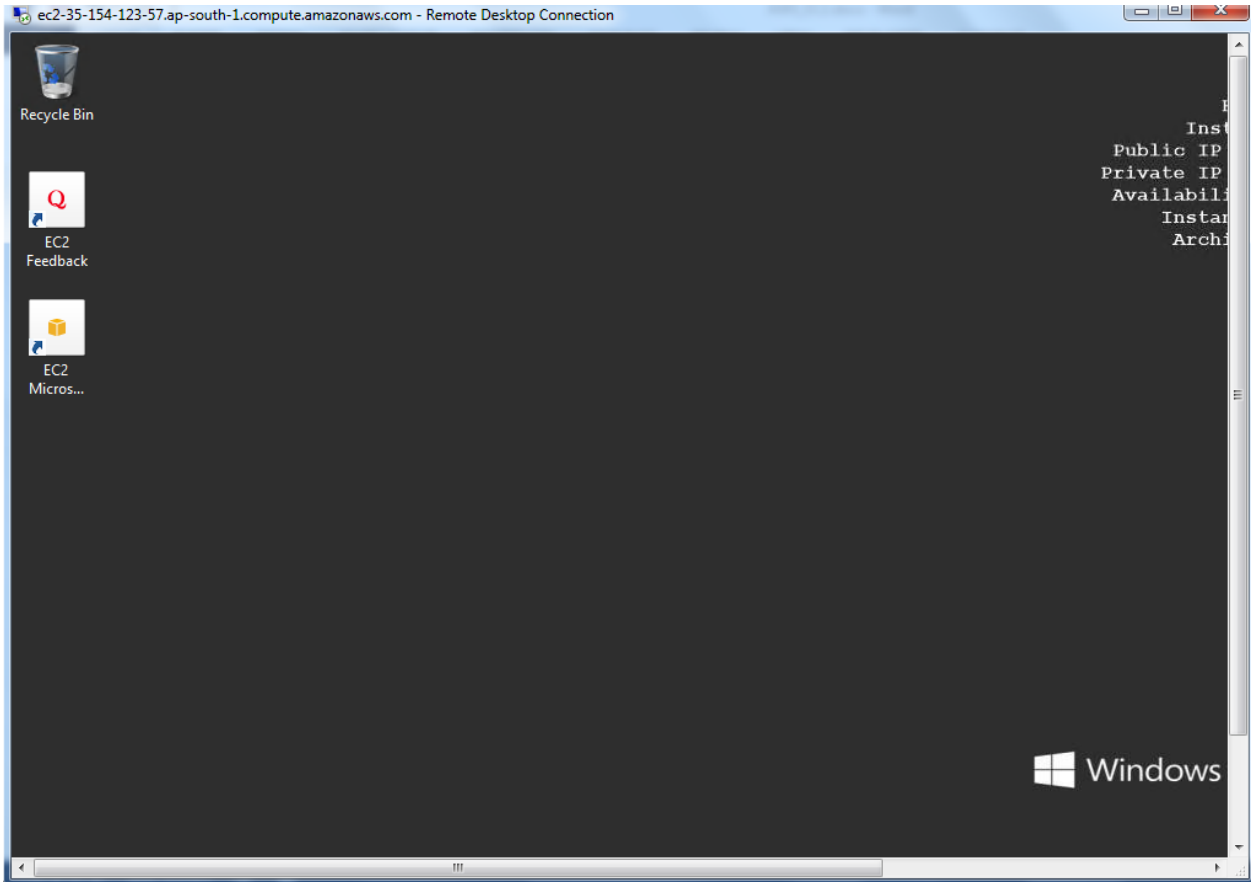




17 Then you'll get the UN and Password, you can enter this UN & Pwd and click on connect, You'll asked for Certificate error prompt, simply click on **Yes** to connect to this machine.



18 Now we have successfully connected to Windows Instance



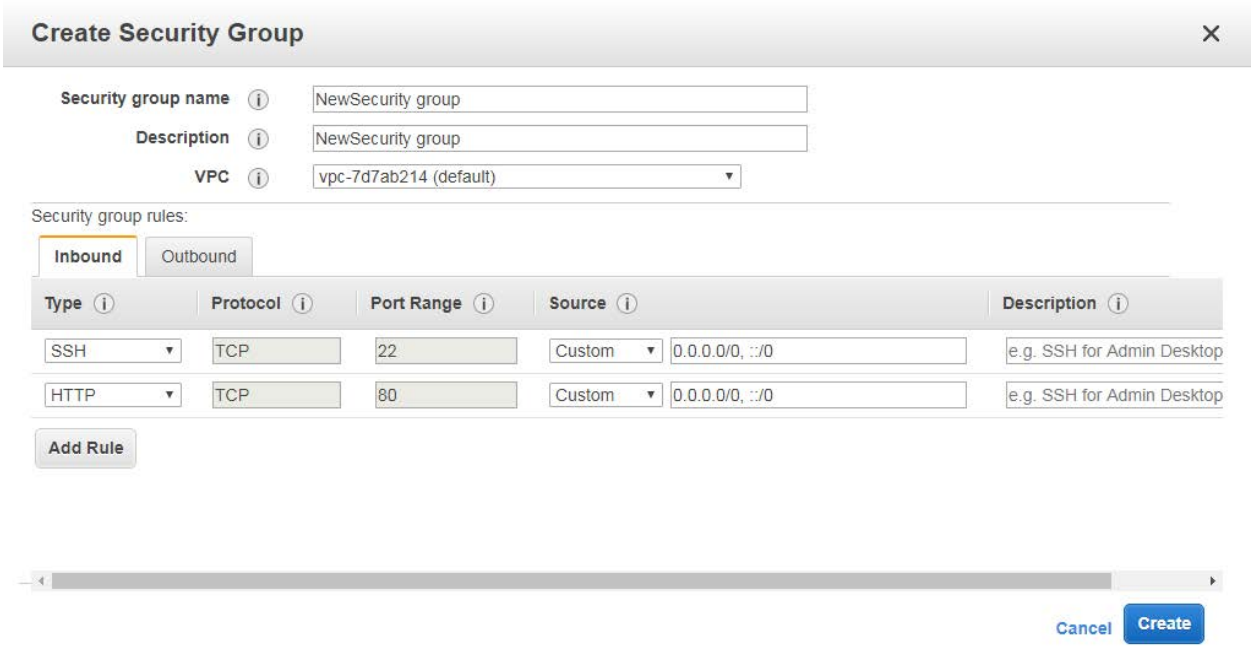
Security groups

Security groups allow you to control traffic based on port, protocol, and source/destination.

You can use Security Groups to restrict and filter out both the inbound and outbound traffic of an instance using a set of firewall rules. Each rule can allow traffic based on a particular protocol—TCP or UDP, based on a particular port—such as 22 for SSH, or even based on individual source and destination IP addresses. This provides lot of control and flexibility in terms of designing a secure environment for instances to run from.

- Security groups are associated with instances when they are launched. Every instance must have at least one security group but can have more.
- A security group is **default deny**; that is, it does not allow any traffic that is not explicitly allowed by a security group rule.
- A security group is a **stateful firewall,** If you open some port in inbound, it'll automatically allowed for outbound also.

- Security groups are applied at the instance level.
- Changes to Security Groups take effect immediately
- We cannot block specific IP address using security groups.
- We can specify allow rules, but not deny rules.
- We can modify the firewall rules of Security Groups any time, even when your instance is running.



You can select the Protocol Type inn Type field, automatically it'll show the protocol type and Port Range, and then we have to select the source.

Source field where you can basically specify any of these three options:

Anywhere: Using this option as the source, particular application port will be accessible from any and all networks out there (0.0.0.0/0). This is not a recommended configuration by AWS.

My IP: AWS will autofill the IP address of your local computer/Network here. If you select My IP option then the service works only in that particular network only.

Custom IP: This is the most preferable option, the Custom IP option allows you to specify your own custom source IP address or IP range as per our requirements. Ex: allow the particular application to access only via traffic coming from the network 202.153.31.0/24 CIDR.

VOLUMES AND SNAPSHOTS

An Amazon EBS volume is a durable, block-level storage device that you can attach to a single EC2 instance.

Amazon EBS provides persistent block-level storage volumes for use with Amazon EC2 instances. Each Amazon EBS volume is automatically replicated within its Availability Zone to protect you from component failure, offering high availability and durability.

Multiple Amazon EBS volumes can be attached to a single Amazon EC2 instance, although a volume can only be attached to a single instance at a time.

Types of Amazon EBS Volumes

Amazon EBS provides the following volume types:

- General Purpose SSD (gp2),
- Provisioned IOPS SSD (io1),
- Throughput Optimized HDD (st1),
- Cold HDD (sc1), and
- Magnetic (standard, a previous-generation type).

SSD-backed volumes optimized for transactional workloads involving frequent read/write operations with small I/O size, where the dominant performance attribute is IOPS

HDD-backed volumes optimized for large streaming workloads where throughput (measured in MiB/s) is a better performance measure than IOPS.

General Purpose SSD (gp2):

General Purpose SSD (gp2) volumes offer cost-effective storage that is ideal for a broad range of workloads. These volumes deliver single-digit millisecond latencies and the ability to burst to 3,000 IOPS for extended periods of time. Between a minimum of 100 IOPS (at 33.33 GiB and below) and a maximum of 10,000 IOPS (at 3,334 GiB and above), baseline performance scales linearly at 3 IOPS per GiB of volume size. AWS designs gp2 volumes to deliver the provisioned performance 99% of the time. A gp2 volume can range in size from 1 GiB to 16 TiB.

Provisioned IOPS SSD (io1):

Provisioned IOPS SSD (io1) volumes are designed to meet the needs of I/O-intensive workloads, particularly database workloads, that are sensitive to storage performance and consistency.

- An io1 volume can range in size from 4 GiB to 16 TiB and you can provision 100 up to 20,000 IOPS per volume.

Throughput Optimized HDD (st1):

Throughput Optimized HDD (st1) volumes provide low-cost magnetic storage that defines performance in terms of throughput rather than IOPS. This volume type is a good fit for large, sequential workloads such as Amazon EMR, ETL, data warehouses, and log processing.

- Not supported to use with root volume (Not Bootable)
- volume sizes ranging from 0.5 to 16 TiB

Cold HDD (sc1) Volumes

Cold HDD (sc1) volumes provide low-cost magnetic storage that defines performance in terms of throughput rather than IOPS. With a lower throughput limit than st1, sc1 is a good fit ideal for large, sequential cold-data workloads. If you require infrequent access to your data and are looking to save costs, sc1 provides inexpensive block storage.

- Not supported to use with root volume (Not Bootable)
- volume sizes ranging from 0.5 to 16 TiB

Magnetic volumes:

Magnetic volumes are backed by magnetic drives and are suited for workloads where data is accessed infrequently, and scenarios where low-cost storage for small volume sizes is important. These volumes deliver approximately 100 IOPS on average, with burst capability of up to hundreds of IOPS, and they can range in size from 1 GiB to 1 TiB.

- volume sizes ranging from 1 GiB to 1 TiB.

Solid-State Drives (SSD)		**Hard disk Drives (HDD)**		**sfsffd**
Volume Type	General Purpose SSD (gp2)*	Provisioned IOPS SSD (io1)	Throughput Optimized HDD (st1)	Cold HDD (sc1)

Description	General purpose SSD volume that balances price and performance for a wide variety of transactional workloads	Highest-performance SSD volume designed for mission-critical applications	Low cost HDD volume designed for frequently accessed, throughput-intensive workloads	Lowest cost HDD volume designed for less frequently accessed workloads
API Name	gp2	io1	st1	sc1
Volume Size	1 GiB - 16 TiB	4 GiB - 16 TiB	500 GiB - 16 TiB	500 GiB - 16 TiB
Max. IOPS/Volume**	10,000	20,000	500	250
Max. Throughput/Volume†	160 MiB/s	320 MiB/s	500 MiB/s	250 MiB/s
Max. IOPS/Instance	75,000	75,000	75,000	75,000
Max. Throughput/Instance	1,750 MB/s	1,750 MB/s	1,750 MB/s	1,750 MB/s

Previous Generation Volumes	
Volume Type	EBS Magnetic
Description	Previous generation HDD
Use Cases	Workloads where data is infrequently accessed
API Name	standard
Volume Size	1 GiB-1 TiB
Max. IOPS/Volume	40–200

Max. Throughput/Volume	40–90 MiB/s
Max. IOPS/Instance	48,000
Max. Throughput/Instance	1,250 MiB/s

Throughput is the maximum rate of production or the maximum rate at which something can be processed.

Network throughput is the rate of successful message delivery over a communication channel.

INSTANCE STORE VOLUME

An instance store provides temporary block-level storage for your instance. This storage is located on disks that are physically attached to the host computer. Instance store is ideal for temporary storage of information that changes frequently, such as buffers, caches, scratch data, and other temporary content

Instance Store Lifetime

- The underlying disk drive fails
- The instance stops
- The instance terminates

Instance Store Volumes are called as Ephemeral Storage.

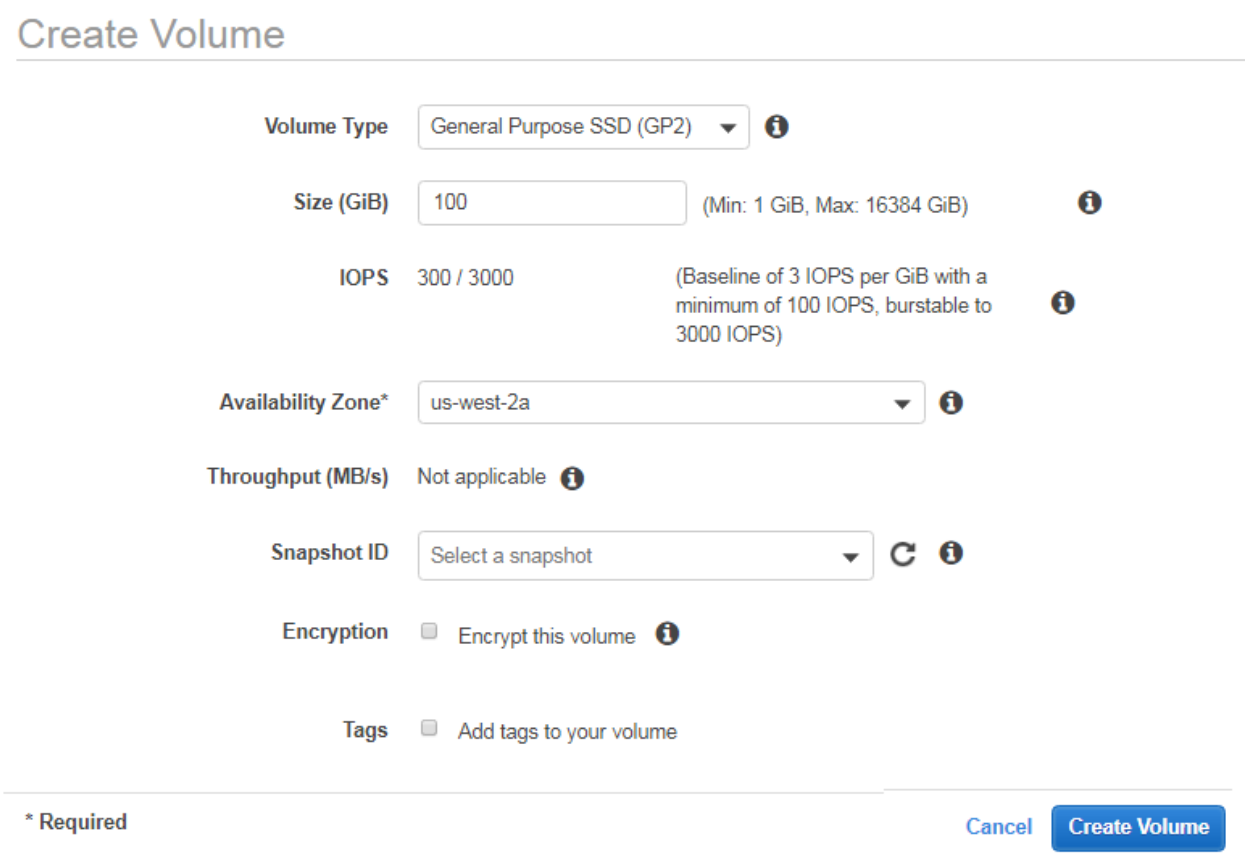
Instance store volumes cannot be stopped. If the underlying host fails, you will lose your data.

EBS backed instances can be stopped. You will not lose the data on this instance if it is stopped.

By default, both ROOT volumes will be deleted on termination, however with EBS volumes, you can keep the root device volume by Unchecking the “Delete on Termination” option.

Create a Volume:

From the Volume Management dashboard, select the Create Volume option.



Type: From the Type drop-down list, select either General Purpose (SSD), Provisioned IOPS (SSD), or Magnetic as per the requirements.

Size (GiB): Provide the size of your volume in GB.

IOPS: This field will only be editable if you have selected Provisioned IOPS (SSD) as the volume's type. Enter the max IOPS value as per your requirements.

Availability Zone: Select the appropriate availability zone in which you wish to create the volume.

Snapshot ID: This is an optional field. We can choose to populate your EBS volume based on a third party's snapshot ID.

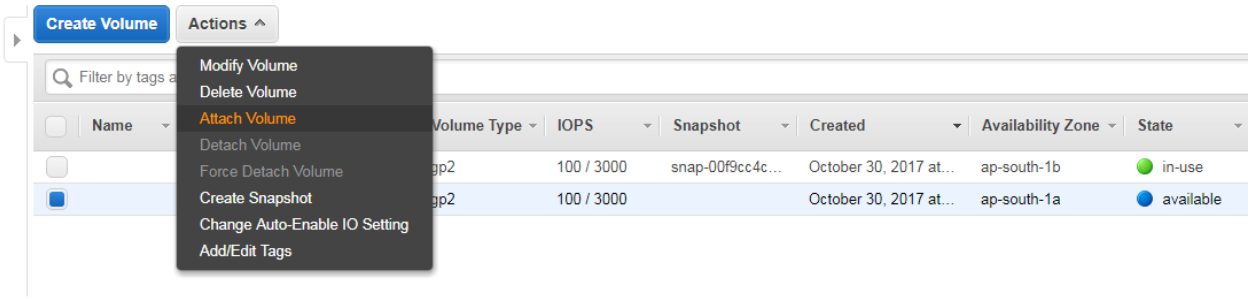
Encryption: We can choose whether or not to encrypt EBS Volume. Select Encrypt this volume checkbox if you wish to do so.

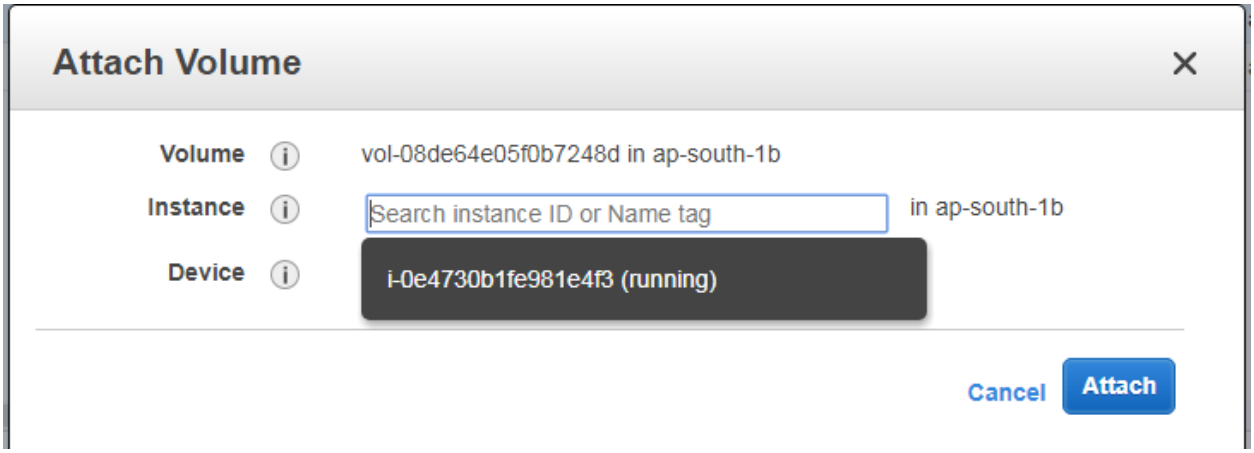
Master Key: On selecting the Encryption option, AWS will automatically create a default key pair for the AWS's KMS.

Once configuration settings are filled in, select Create to complete the volume's creation process. The new volume will take a few minutes to be available for use. Once the volume is created, we can now attach this volume to running instance.

Attaching EBS Volumes: Once the EBS volume is created, make sure it is in the available state before you go ahead and attach it to an instance. You can attach multiple volumes to a single instance at a time.

To attach a volume, select the **volume** from the Volume Management dashboard. Then select the **Actions** tab and click on the **Attach Volume** option.





When you select instance field, automatically you'll get thee running instances list from that particular availability zone. Select the Instance you want to attach this volume. Then click on **Attach**. Now the Volume state will change to **in-use** from Available.

We have to mount this volume from operating system level. For windows, you have to perform it though Disk Management option.

In Linux:

1. Elevate your privileges to root.
2. Type **df –h** command to check the current disk partitioning of instance.
3. Give **fdisk –l** command to verify the newly added disk.

```
[root@ip-172-31-7-51 ~]# fdisk -l
WARNING: fdisk GPT support is currently new, and therefore in an experimental phase.
 Use at your own discretion.

Disk /dev/xvda: 8589 MB, 8589934592 bytes, 16777216 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: gpt


#         Start          End    Size  Type            Name
 1         4096     16777182      8G  Linux filesyste Linux
128         2048         4095      1M  BIOS boot parti BIOS Boot Partition

Disk /dev/xvdf: 1073 MB, 1073741824 bytes, 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

4. We have to choose the file system type. Here am using ext4 file system. Then run the following command.

   Mkfs –t ext4 /dev/xvdf

```
[root@ip-172-31-7-51 ~]# mkfs -t ext4 /dev/xvdf
mke2fs 1.42.12 (29-Aug-2014)
Creating filesystem with 262144 4k blocks and 65536 inodes
Filesystem UUID: 380ed17c-022a-440e-a696-ccd0caa3bd78
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

5. Now volume is formatted, we can create a new directory on Linux instance and mount the volume to it using standard Linux commands:

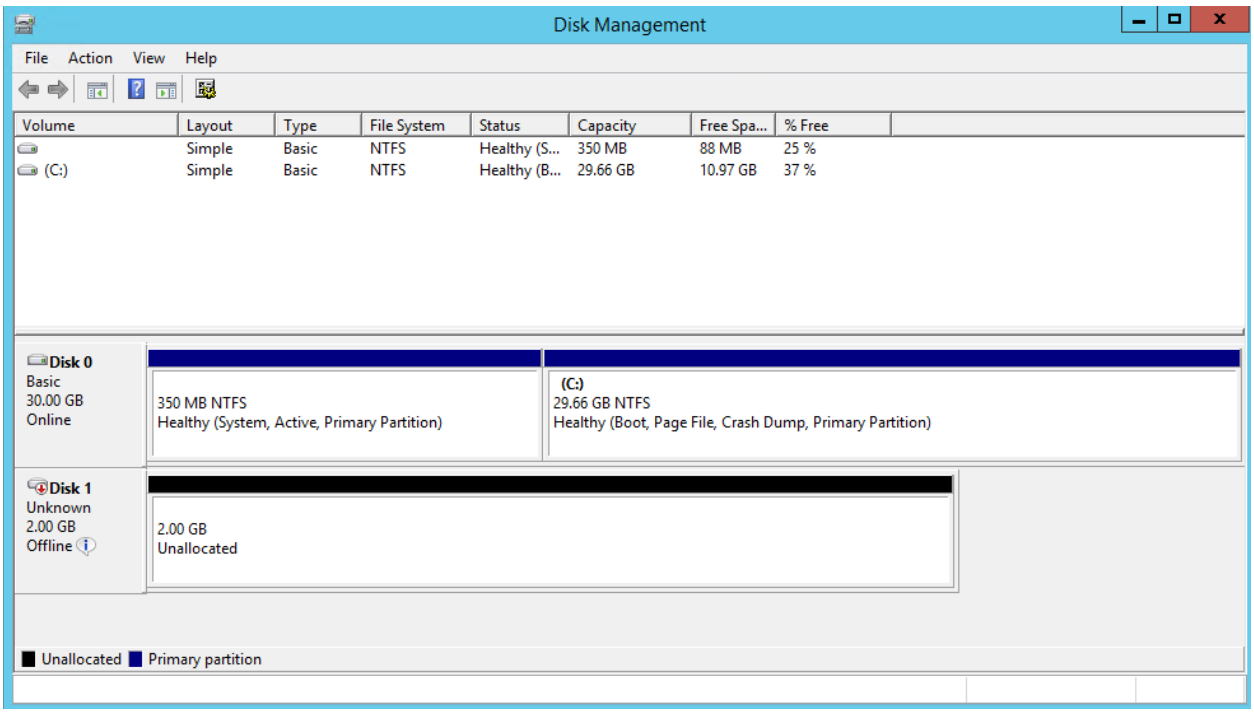
   mkdir /newvolume
   mount /dev/xvdf /newvolume

```
[root@ip-172-31-7-51 ~]# mkdir /newvolume
[root@ip-172-31-7-51 ~]# mount /dev/xvdf /newvolume
[root@ip-172-31-7-51 ~]#
```

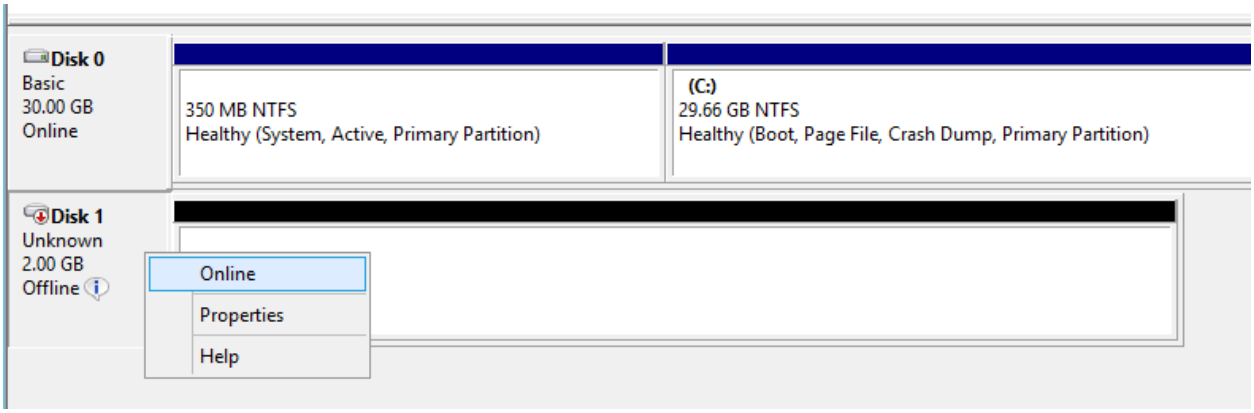
6. Now the volume is available for the use.

For Windows Instances:

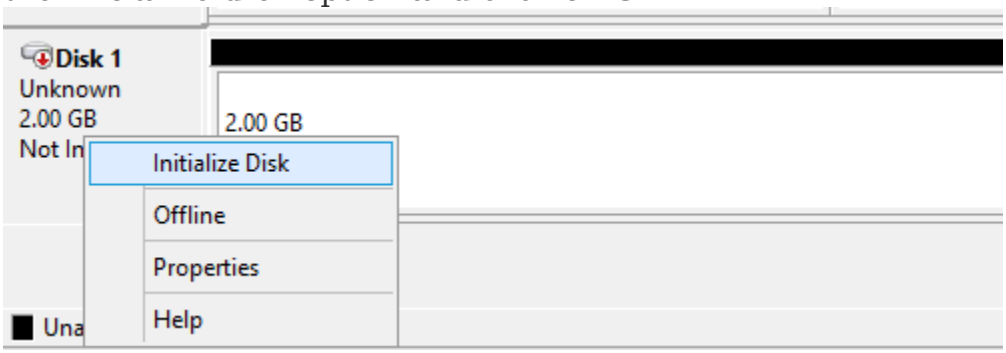
1. Attach the volume to the windows instance same as previous step.
2. Login to the windows instance and open Disk management console.
3. Open Run and give **diskmgmt.msc** command to open the Disk Management.

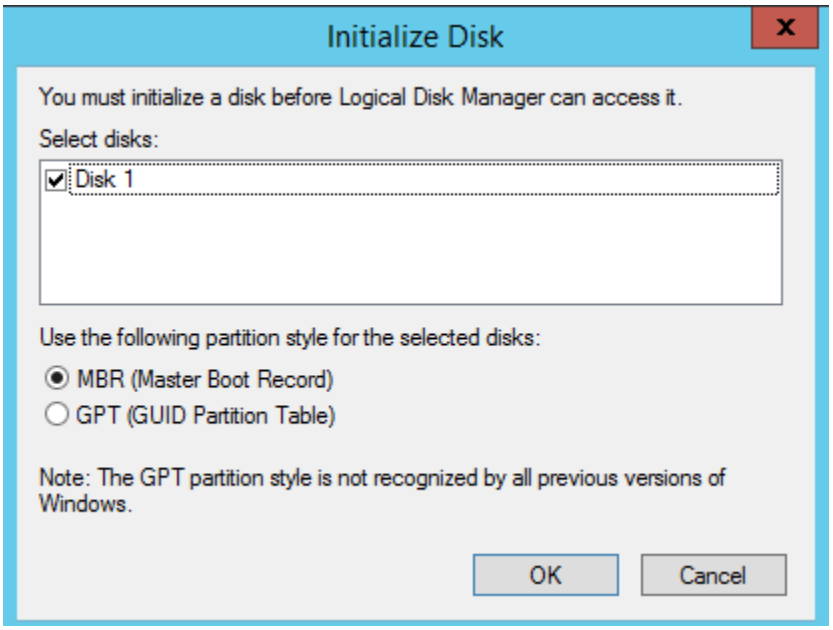


4. The newly created 2GB volume is attached to the Windows instance and by default the status of this drive will set to offline, Select the Disk 1, then choose **Online** option to make the volume online.

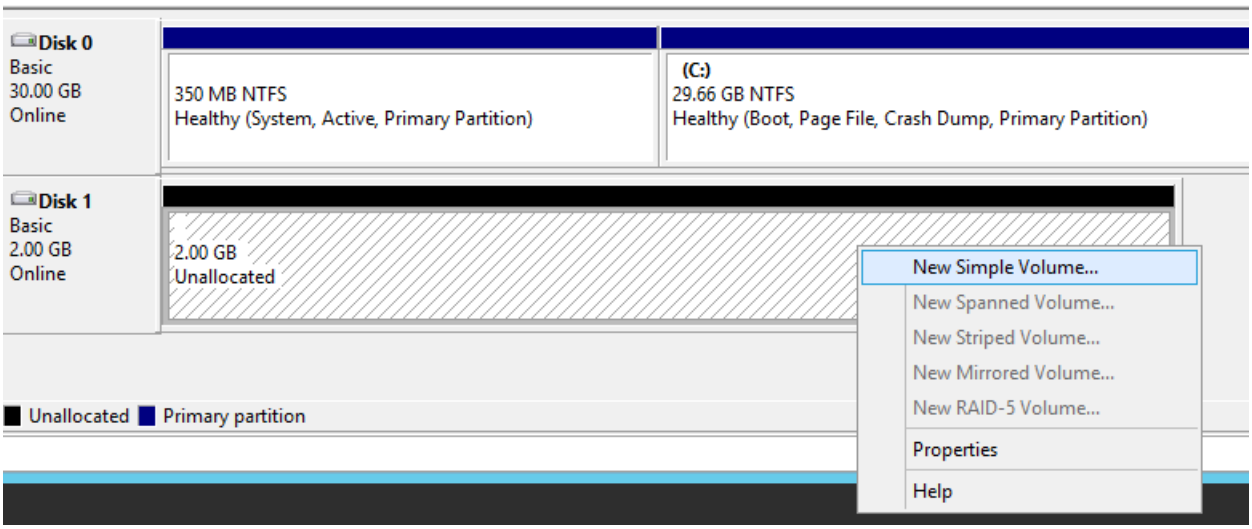


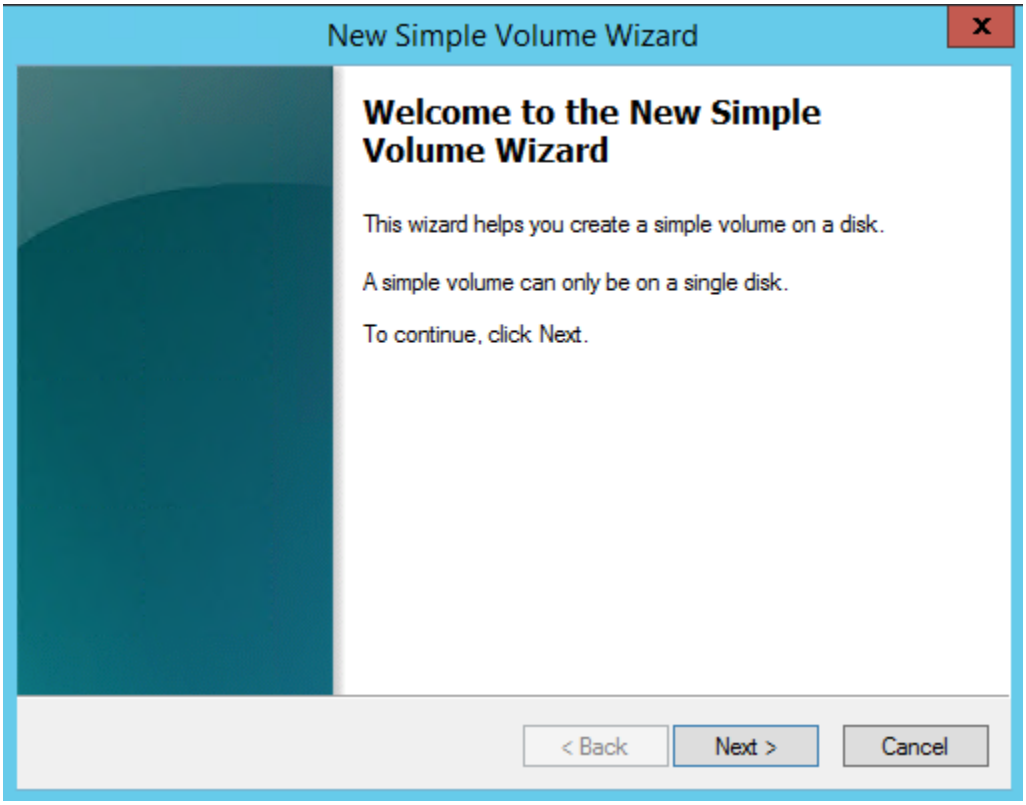
5. Here we have to initialize the Disk, Give right click on Disk then select the **initialize disk** option and click on **OK**

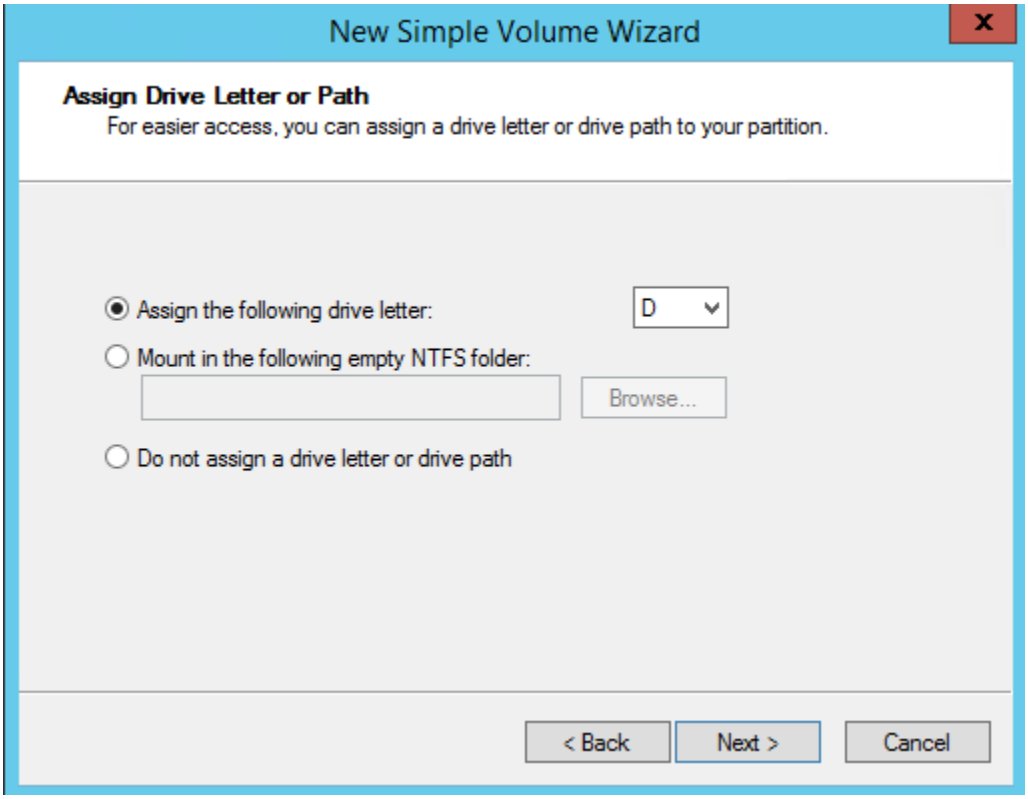


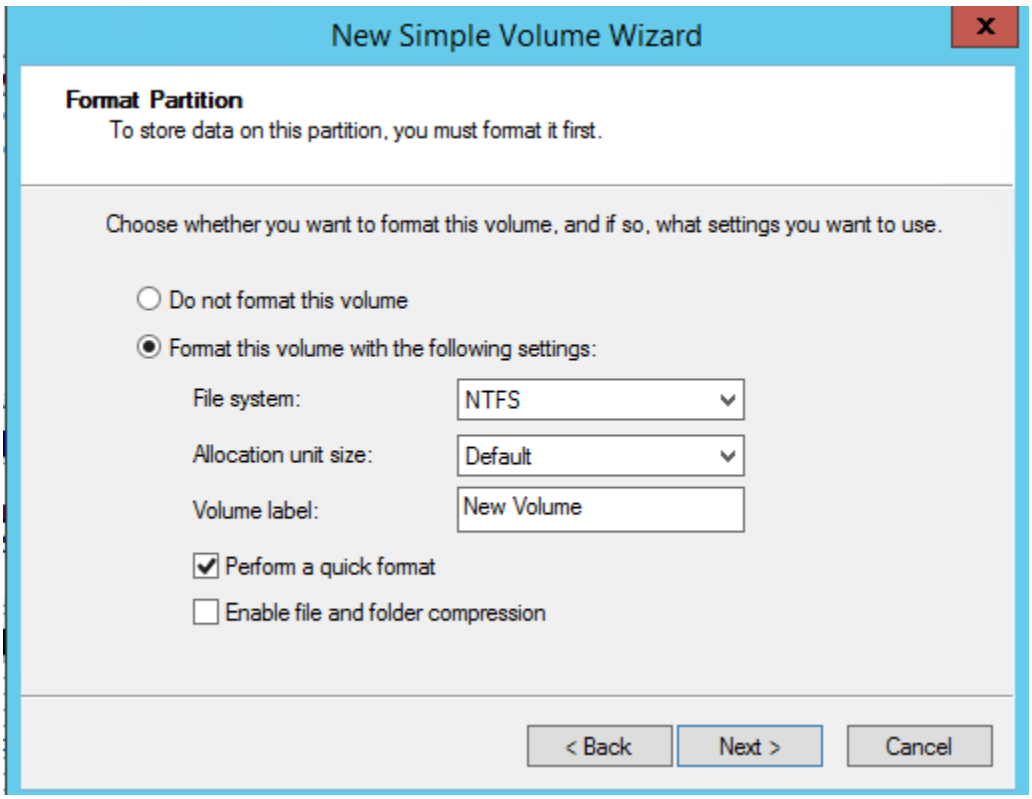


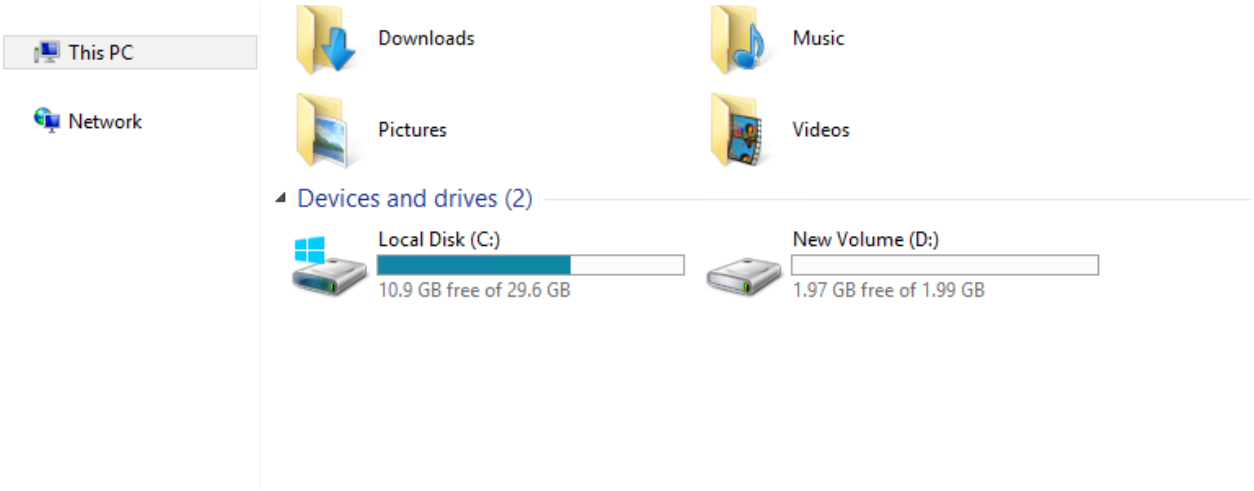
6. Now we have to create a volume, Give right click on dive select the “New Simple Volume” option, It will open up a Volume creation wizard, follow the wizard as below images


Disk 0
Basic
30.00 GB
Online
350 MB NTFS
Healthy (System, Active, Primary Partition)
(C:)
29.66 GB NTFS
Healthy (Boot, Page File, Crash Dump, Primary Partition)
Disk 1
Basic
2.00 GB
Online
2.00 GB
Unallocated
New Simple Volume...
New Spanned Volume...
New Striped Volume...
New Mirrored Volume...
New RAID-5 Volume...
Properties
Help
Unallocated
Primary partition


New Simple Volume Wizard
Welcome to the New Simple Volume Wizard
This wizard helps you create a simple volume on a disk.
A simple volume can only be on a single disk.
To continue, click Next.
< Back
Next >
Cancel


New Simple Volume Wizard
Assign Drive Letter or Path
For easier access, you can assign a drive letter or drive path to your partition.
Assign the following drive letter:
D
Mount in the following empty NTFS folder:
Browse...
Do not assign a drive letter or drive path
< Back
Next >
Cancel


New Simple Volume Wizard
Format Partition
To store data on this partition, you must format it first.
Choose whether you want to format this volume, and if so, what settings you want to use.
Do not format this volume
Format this volume with the following settings:
File system:
NTFS
Allocation unit size:
Default
Volume label:
New Volume
Perform a quick format
Enable file and folder compression
< Back
Next >
Cancel



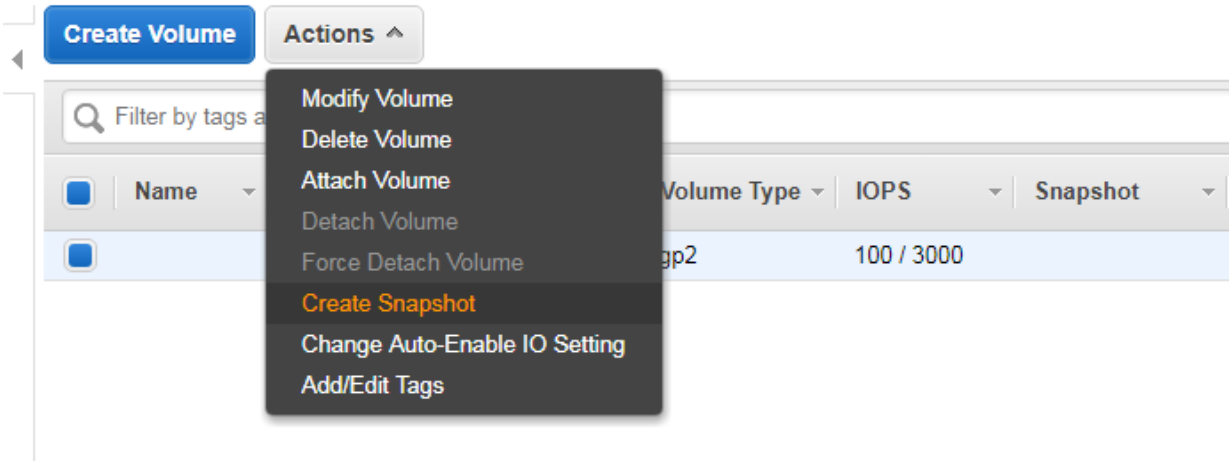
7. Now we can see the newly created volume along with other volumes. You can use the Disk Management console to Shrink, extend or to delete the volumes.

Backup of EBS volumes

We can back up the data on our Amazon EBS volumes, regardless of volume type, by taking point-in-time snapshots.

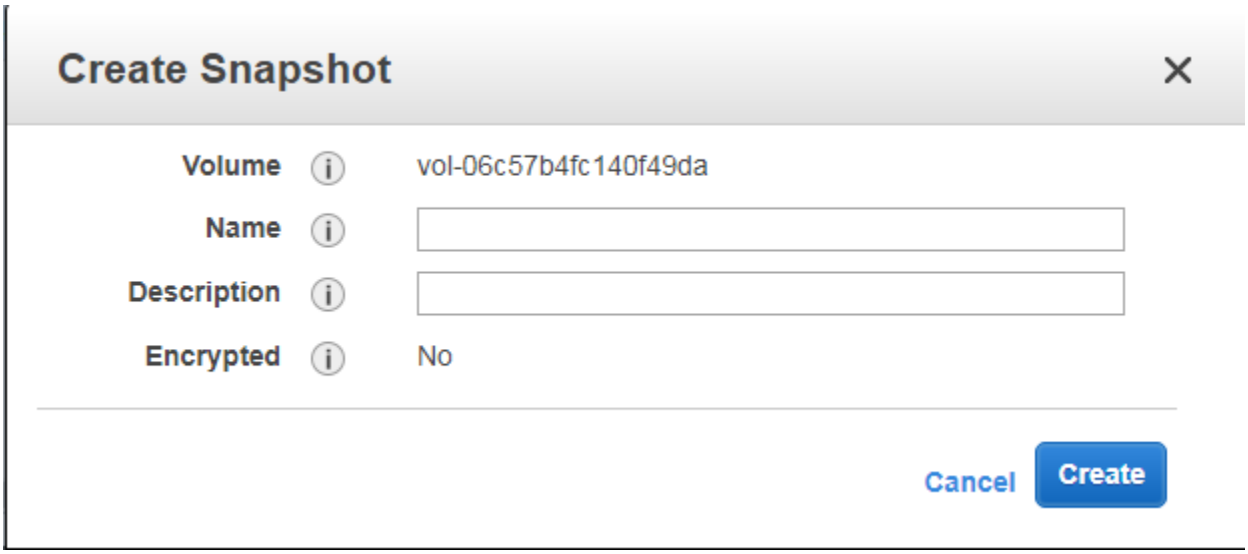
- Snapshots are incremental backups, which means that only the blocks on the device that have changed since your most recent snapshot are saved.
- Data for the snapshot is stored using Amazon S3 technology.
- While snapshots are stored using Amazon S3 technology, they are stored in AWS-controlled storage and not in your account's Amazon S3 buckets.
- Snapshots are constrained to the region in which they are created, meaning you can use them to create new volumes only in the same region.
- If you need to restore a snapshot in a different region, you can copy a snapshot to another region.
- Snapshots can also be used to increase the size of an Amazon EBS volume.
    - To increase the size of an Amazon EBS volume, take a snapshot of the volume, then create a new volume of the desired size from the snapshot. Replace the original volume with the new volume.

To create a snapshot of volumes, select the particular volume from the Volume Management dashboard. Click on the **Actions** tab and select the **Create Snapshot** option.

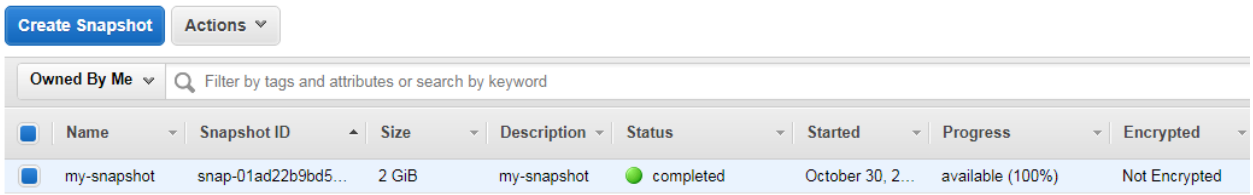


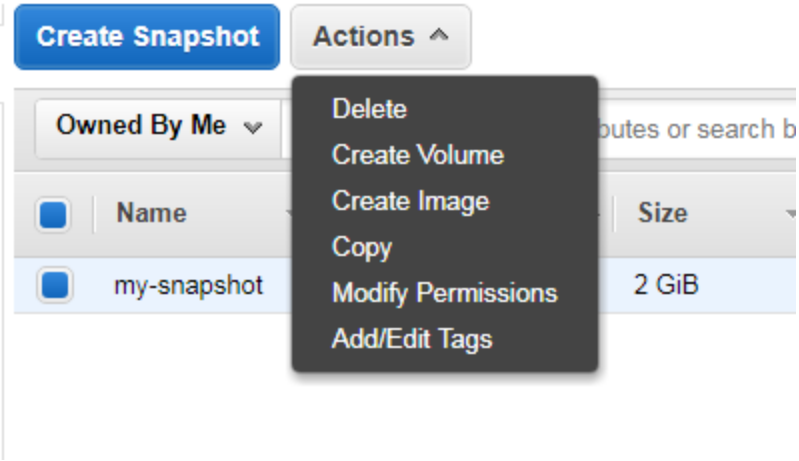
Give a Name and Description for the Snapshot.

- Snapshot of an Encrypted root volume is going to be an encrypted one.
- Volume creating from the encrypted snapshot also going to be an encrypted one.
- We can share the snapshots, but the snapshot must be an **unencrypted.**



We can go to Snapshot dashboard to verify the snapshot creation.





The above are the options available for snapshot.

Delete: we can delete the selected snapshot with this option.

Create Volume: We can create a new volume from this snapshot, while creating the new snapshot, we can change the volume type or increase the size if we want.

Create Image: We can create an AMI from this snapshot.

Copy: We can copy the snapshot from one region to another region.

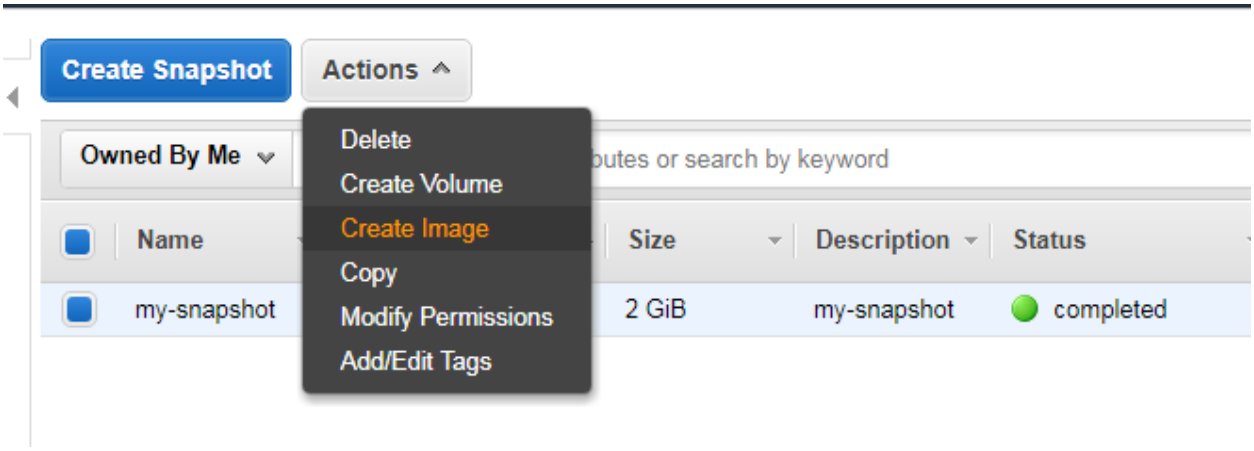
Modify Permissions: We can share the snapshots with specific AWS account user or made available to public, but this option will not enable if our snapshot is an encrypted.

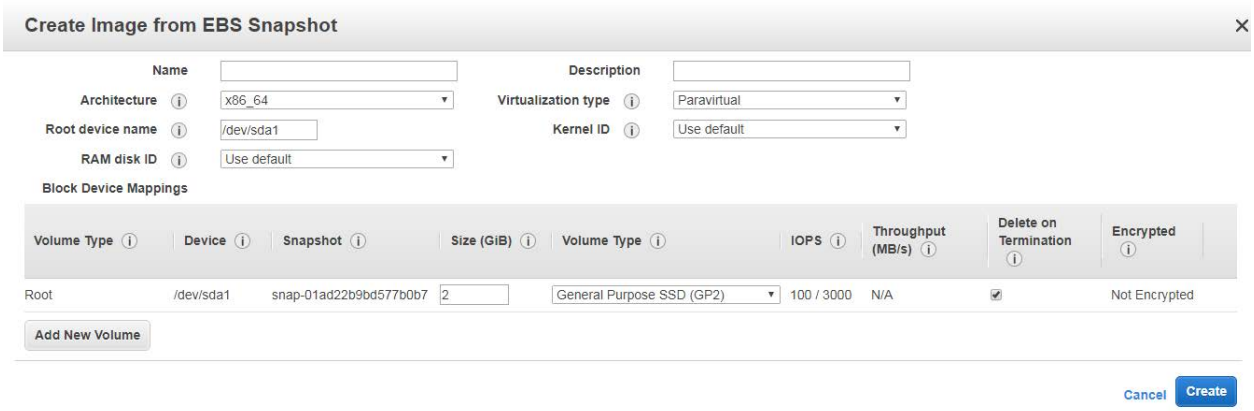
Creating an AMI

An Amazon Machine Image (AMI) provides the information required to launch a virtual server in the cloud. You specify an AMI when you launch an instance, and you can launch as many instances from the AMI as you need. You can also launch instances from as many different AMIs as you need.

- A template for the root volume for the instance
- Launch permissions that control which AWS accounts can use the AMI to launch instances.

To create an AMI, Select the root volume's Snapshot, then select **Create Image** option.





Name: Provide a suitable and meaningful name for your AMI.

Description: Provide a suitable description for your new AMI.

Architecture: We can either choose between i386 (32 bit) or x86_64 (64 bit).

Root device name: Enter a suitable name for your root device volume.

Virtualization type: We can choose whether the instances launched from this particular AMI will support Para virtualization (PV) or Hardware Virtual Machine (HVM) virtualization.

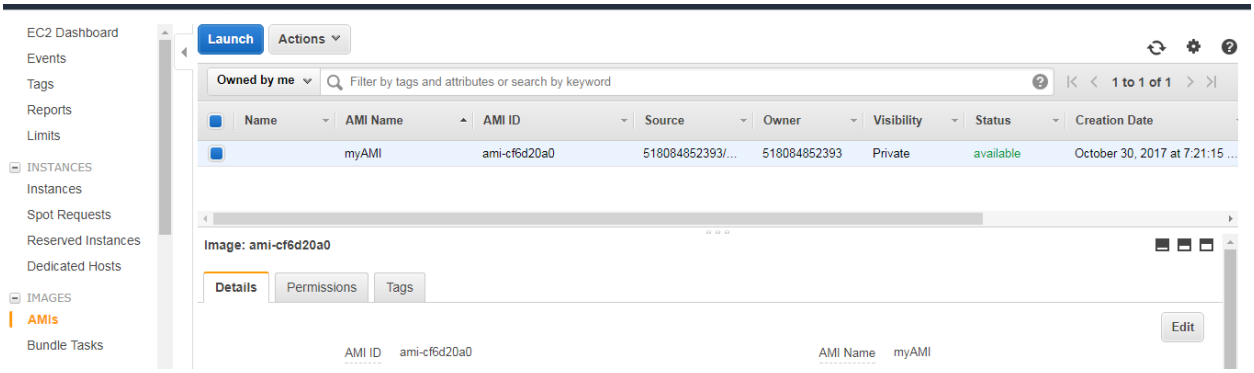
- **Xen** is an hypervisor that runs on metal (the pc / server) and then hosts virtual machines called domains.
- **A Xen PV** domain is a paravirtualized domain, that means the operating system has been modified to run under Xen, and there's no need to actually emulate hardware. This should be the most efficient way to go, performance wise.

- **A Xen HVM** domain is hardware emulated domain, that means the operating system (could be Linux, Windows, whatever) has not been modified in any way and hardware gets emulated.

RAM disk ID, Kernel ID: We can select and provide your AMI with its own RAM disk ID (ARI) and Kernel ID (AKI); however, in this case I have opted to keep the default ones.

Block Device Mappings: We can use this dialog to either expand root volume's size or add additional volumes to it. We can change the Volume Type from General Purpose (SSD) to Provisioned IOPS (SSD) or Magnetic as per our AMI's requirements.

Click on **Create** to complete the AMI creation process. The new AMI will take a few minutes to spin up.



We can select the AMI and choose **Launch** option to launch a new instance. We will get the instance launch wizard.

- AMI are regional, if required we can copy AMI to another region with Copy option.
- We can share the AMI to any other AWS account users or we can make it public.
- Every AMI will associate with a Snapshot.
- AMI are registered with the AWS accounts, if you no longer required any AMI, you can select Deregister option under **Actions**.
- We cannot delete the Snapshot if it is associated with an AMI.

Elastic Load Balancing

The Elastic Load Balancing service allows you to distribute traffic across a group of Amazon EC2 instances enabling you to achieve high availability in your applications.

Elastic Load Balancing supports routing and load balancing of Hypertext Transfer Protocol (HTTP), Hypertext Transfer Protocol Secure (HTTPS), Transmission Control Protocol (TCP), and Secure Sockets Layer (SSL) traffic to Amazon EC2 instances.

Elastic Load Balancing supports health checks for Amazon EC2 instances to ensure traffic is not routed to unhealthy or failing instances.

We will not get any public IP address for ELBs, We will get a DNS record for every LB.

Advantages of ELB

- Elastic Load Balancing is a managed service, it scales in and out automatically to meet the demands of increased application traffic and is highly available within a region itself as a service.
- ELB helps you achieve high availability for your applications by distributing traffic across healthy instances in multiple Availability Zones.
- ELB seamlessly integrates with the Auto Scaling service to automatically scale the Amazon EC2 instances behind the load balancer.
- ELB is secure, working with Amazon Virtual Private Cloud (Amazon VPC) to route traffic internally between application tiers, allowing you to expose only Internet-facing public IP addresses.
- ELB also supports integrated certificate management and SSL termination.

Internet-Facing Load Balancers: An Internet-facing load balancer is a load balancer that takes requests from clients over the Internet and distributes them to Amazon EC2 instances that are registered with the load balancer.

Internal load balancers: Internal Load Balancers that connect and route traffic to private subnets. We can use internal load balancers to route traffic to your Amazon EC2 instances in VPCs with private subnets.

Listeners: Every load balancer must have one or more listeners configured. A listener is a process that checks for connection requests.

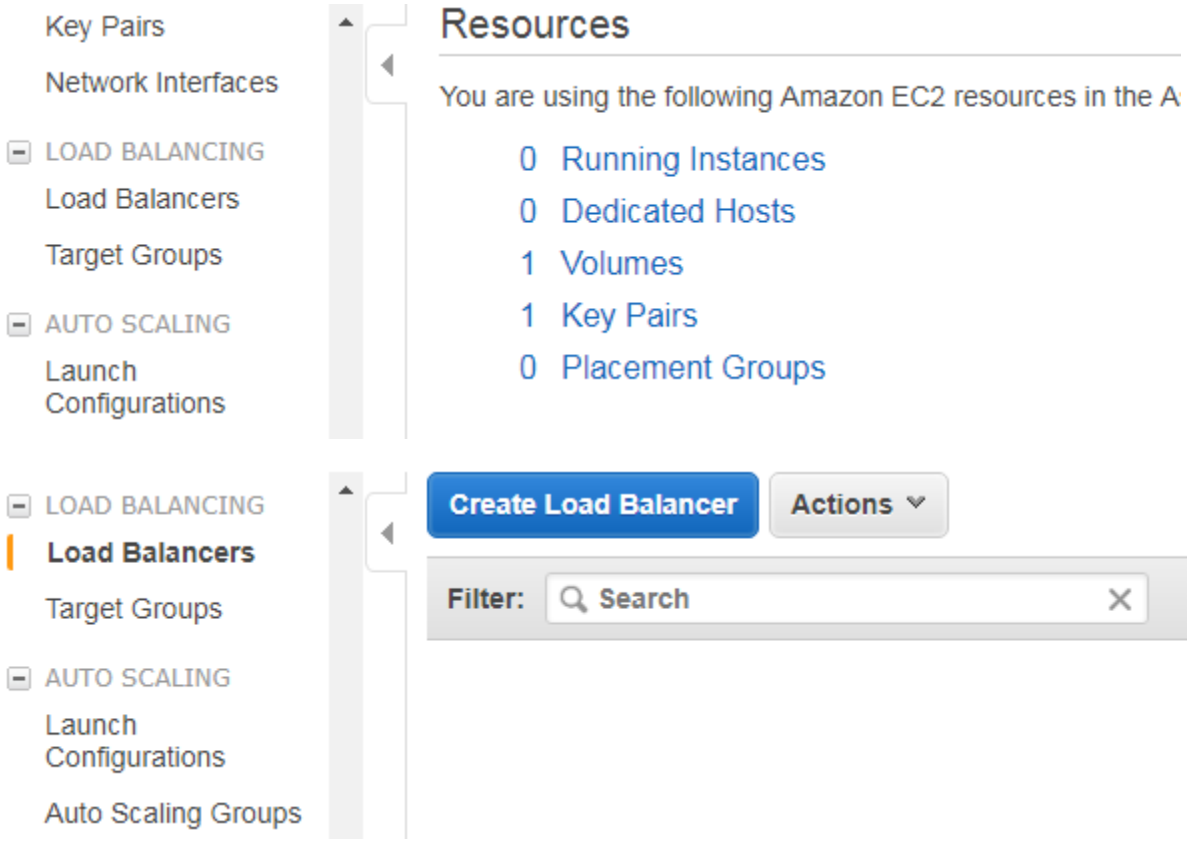
Health Checks

Elastic Load Balancing supports health checks to test the status of the Amazon EC2 instances behind an Elastic Load Balancing load balancer.

- The status of the instances that are healthy at the time of the health check is InService. The status of any instances that are unhealthy at the time of the health check is OutOfService.

- The load balancer performs health checks on all registered instances to determine whether the instance is in a healthy state or an unhealthy state.
- A health check is a ping, a connection attempt, or a page that is checked periodically. You can set the time interval between health checks and also the amount of time to wait to respond in case the health check page includes a computational aspect.
- We can set a Threshold for the number of consecutive health check failures before an instance is marked as unhealthy.

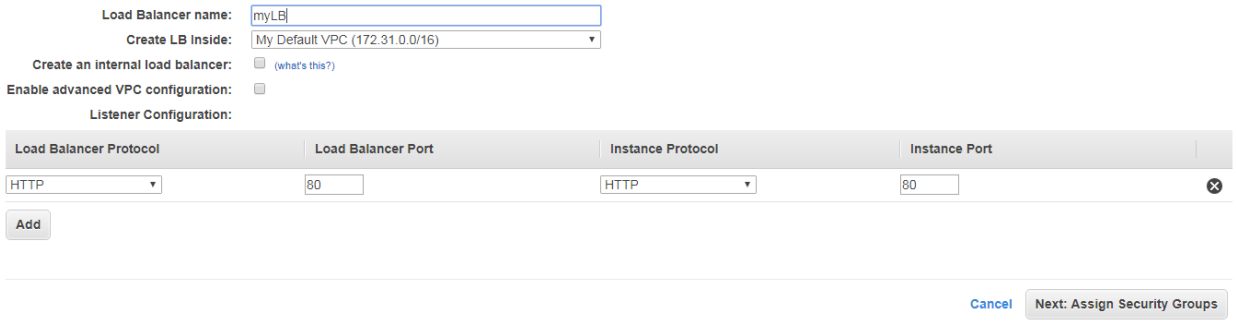
To create ELB navigate to EC2ManagementConsole. Next, from the navigation pane, select the Load Balancers option, this will bring up the ELB Dashboard as well, using which you can create and associate ELBs.



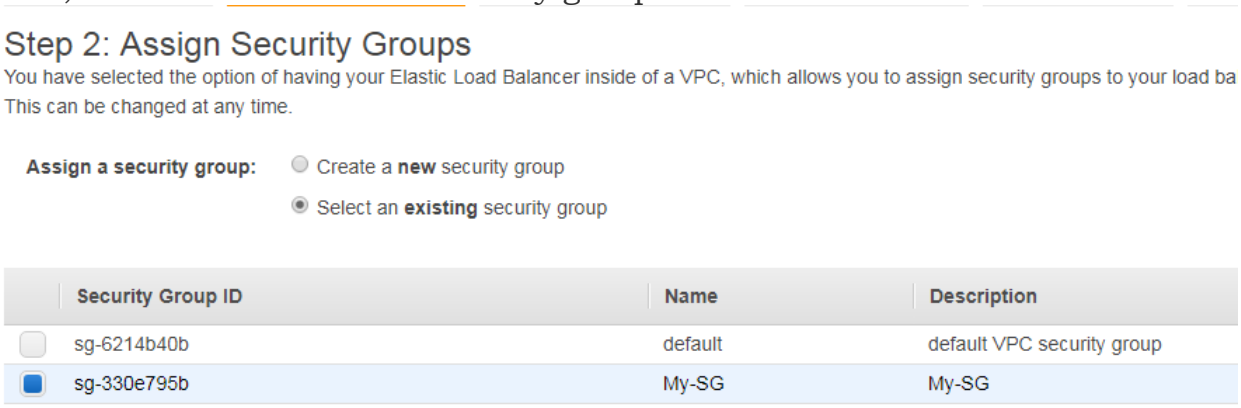
Step 1 – Defining the Load Balancer

1. Select **Create Load Balancer** option and provide a suitable name for ELB in the Load Balancer name field. Next select the VPC option in which you wish to deploy ELB.

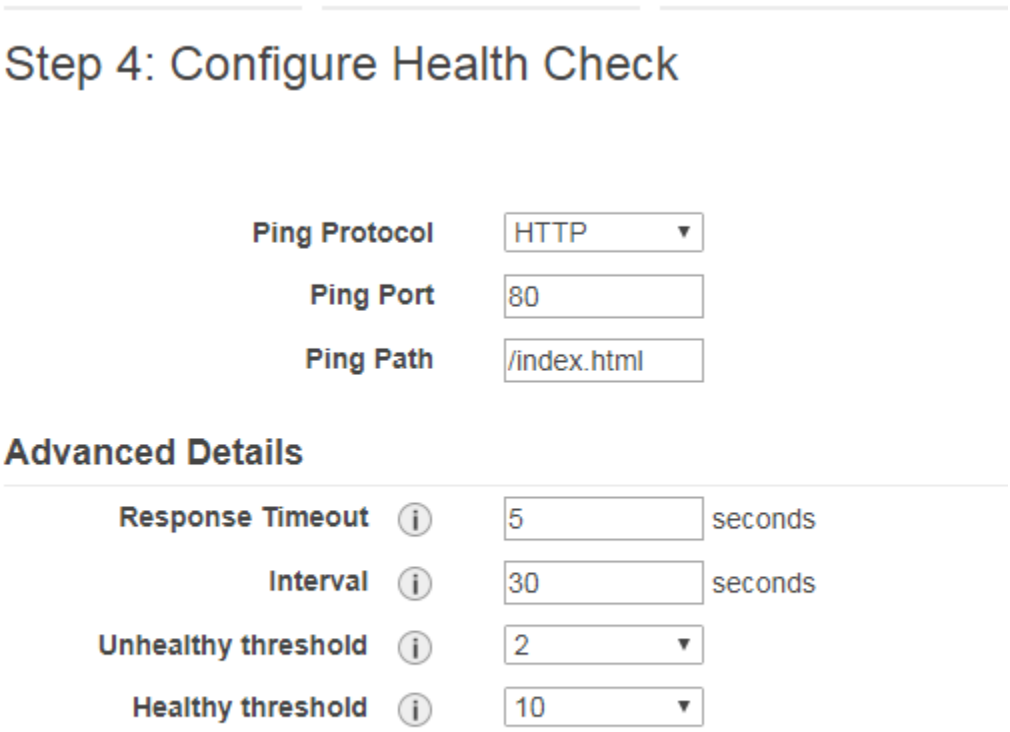
2. Do not check the Create an internal load balancer option as in this scenario, we are creating an Internet-facing ELB for Web Server.
3. In the Listener Configuration section, select HTTP from the Load Balancer Protocol drop-down list and provide the port number 80 in the Load Balancer Port field, as shown in the following screenshot. Provide the same protocol and port number for the Instance Protocol and Instance Port fields.



4. Here, We have to select the Security group for ELB



5. In Step 3 we have to configure security settings, This is an optional page that basically allows you to secure your ELB by using either the HTTPS or the SSL protocol for your frontend connection. But since we have opted for a simple HTTP-based ELB, we can ignore this page. Click on Next: **Configure Health Check** to proceed to the next step.
6. In step 4 we have to configure the health checks.



Ping protocol: This field indicates which protocol the ELB should use to connect to EC2 instances. We can use the TCP, HTTP, HTTPS, or the SSL options.

Ping port: This field is used to indicate the port which the ELB should use to connect to the instance.

Ping path: This value is used for the HTTP and HTTPS protocols. Can also use a /index.html here.

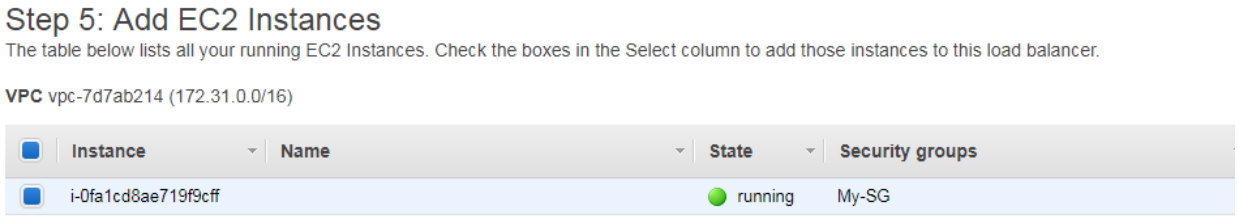
Response time: The Response Time is the time the ELB has to wait in order to receive a response. The default value is 5 seconds with a maximum value up to 60 seconds.

Health Check Interval: This field indicates the amount of time (in seconds) the ELB waits between health checks of an individual EC2 instance. The default value is 30. Maximum value is 300 seconds.

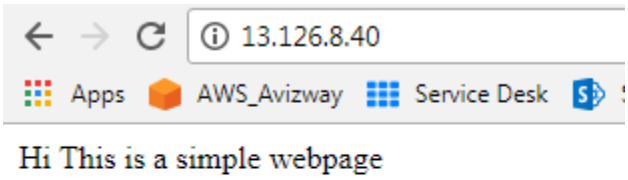
Unhealthy Threshold: This field indicates the number of consecutive failed health checks an ELB must wait before declaring an instance unhealthy. The default value is 2 with a maximum threshold value of 10.

Healthy Threshold: This field indicates the number of consecutive successful health checks an ELB must wait before declaring an instance healthy. The default value is 2 with a maximum threshold value of 10.

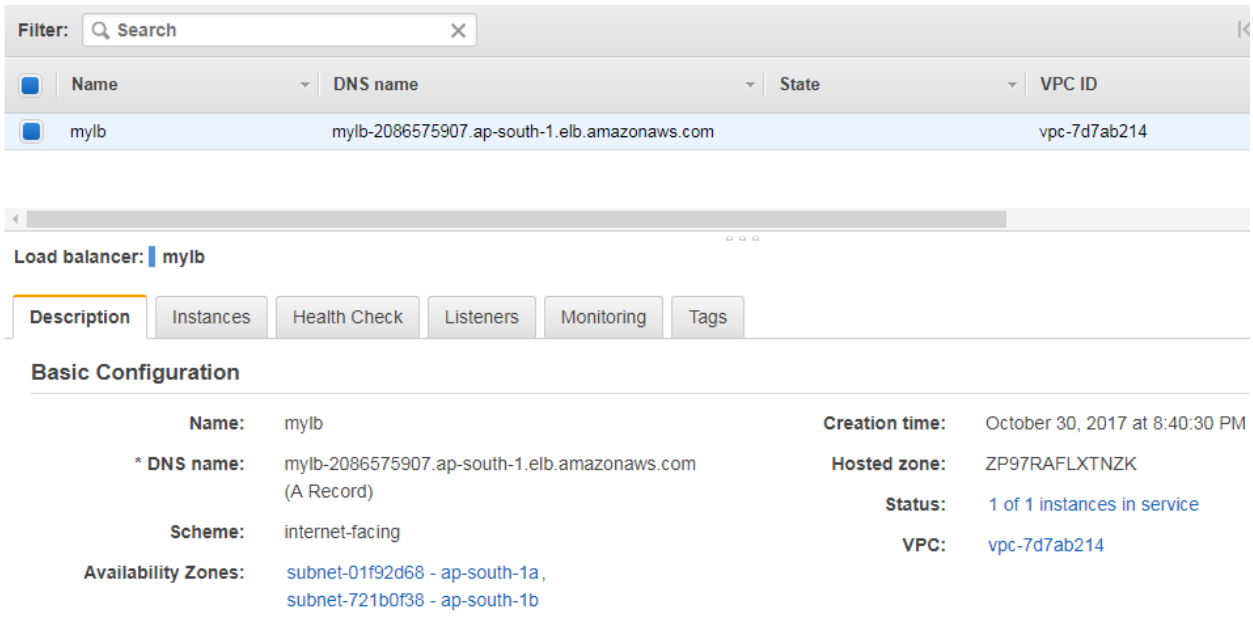
7. Step 5 – Add EC2 instances: We can select any running instance from Subnets to be added and registered with the ELB. Select the EC2 instances you want to launch under this ELS then Click on Next: Add Tags to proceed with the wizard.



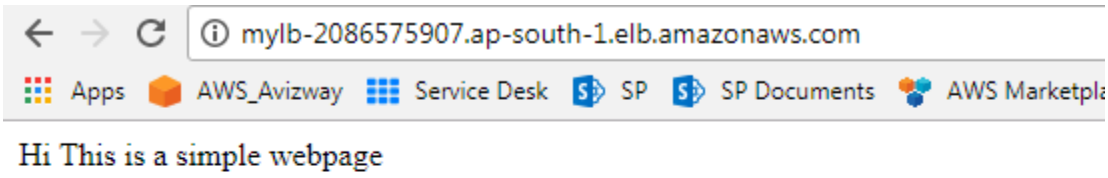
8. In next step, Add any of the tags required and Review the option and click on **Create** option.
9. I have a installed httpd package and created an Index.html file under /var/www/html path in ec2 instance then started the httpd service and am able to get the webpage using the Instance's public IP.



10. And Here is the details for created ELB, As we know we'll get a DNS name for our created ELB, We can access the same webpage by using the ELB's DNS name also.



11. We are able to get the same page by using the DNS name of ELB. This means our ELB configured successfully.



Auto Scaling

Auto Scaling is a service that allows us to scale our Amazon EC2 capacity automatically by scaling out and scaling in according to criteria that we define. With Auto Scaling, we can ensure that the number of running Amazon EC2 instances increases during demand spikes or peak demand periods to maintain application performance and decreases automatically during demand lulls or troughs to minimize costs.

Launch Configuration

A launch configuration is the template that Auto Scaling uses to create new instances, and it is composed of the configuration name, Amazon Machine Image (AMI), Amazon EC2 instance type, security group, and instance key pair. Each Auto Scaling group can have only one launch configuration at a time.

Auto Scaling Group

An Auto Scaling group is a collection of Amazon EC2 instances managed by the Auto Scaling service. Each Auto Scaling group contains configuration options that control when Auto Scaling should launch new instances and terminate existing instances. An Auto Scaling group must contain a name and a minimum and maximum number of instances that can be in the group. You can optionally specify desired capacity, which is the number of instances that the group must have at all times. If you don't specify a desired capacity, the default desired capacity is the minimum number of instances that you specify.

Scaling plans

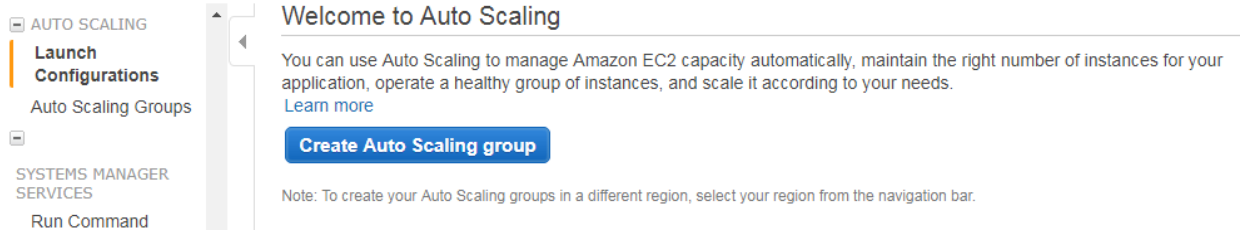
With your Launch Configuration created, the final step left is to create one or more scaling plans. Scaling Plans describe how the Auto Scaling Group should actually scale.

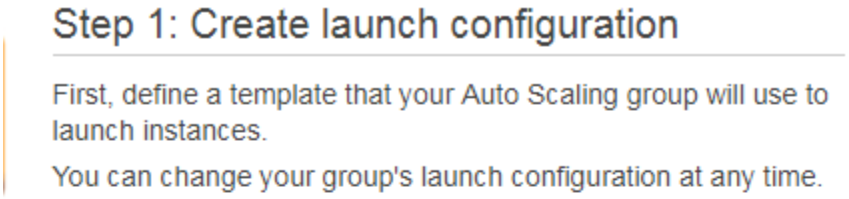
- Manual scaling: here is specify a new desired number of instances value or change the minimum or maximum number of instances in an Auto Scaling Group and the rest is taken care of by the Auto Scaling service itself
- Scheduled scaling: We can scale resources based on a particular time and date
- Dynamic scaling: Dynamic scaling, or scaling on demand is used when the predictability of your application's performance is unknown.

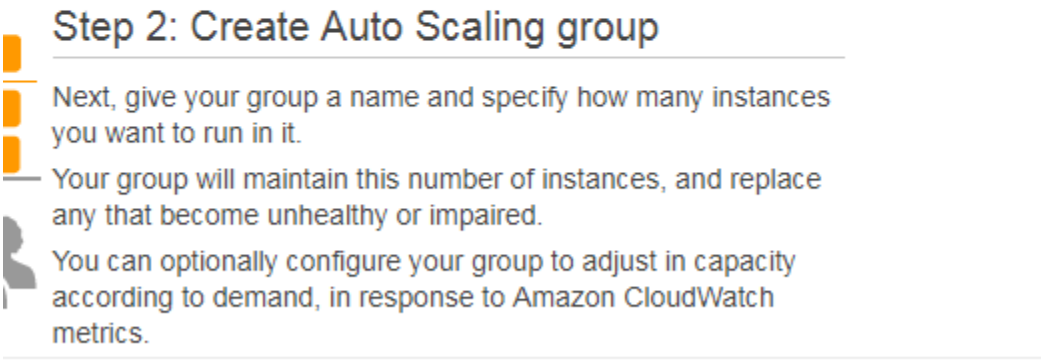
Auto scaling group creation involves with two steps. First one is Creating a Launch Configuration and second is Creating Auto Scaling group.

Creating the Launch Configuration steps

1. Go to **EC2 Management Dashboard** option, select the **AutoScaling Groups** option from the navigation pane. This will bring up the Auto Scaling Groups dashboard. Next, select the **Create Auto Scaling group** option to bring up the Auto Scaling setup wizard.





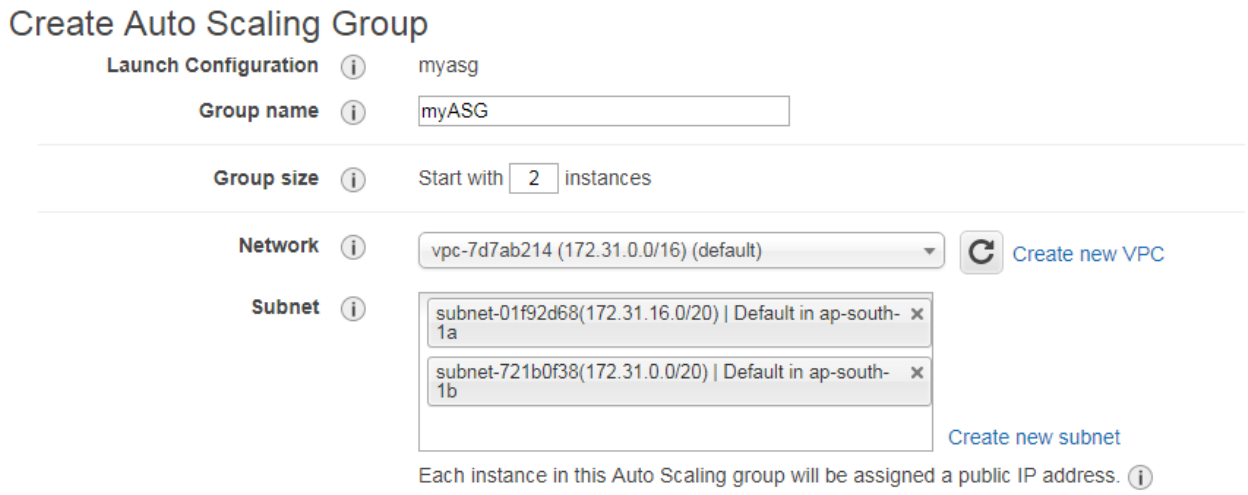


2. Select Create launch configuration is similar to the instance launch wizard. If you have any custom AMIs you can select here.
3. Give a valid name for the Launch configuration. Choose Instance configuration, Storage options, security groups, tags and key pairs and select Create Launch Configuration to complete the process

Step 2: Creating the Auto Scaling Group

An Auto Scaling Group is nothing more than a logical grouping of instances that share some common scaling characteristics between them. Each group has its own set of criteria specified which includes the minimum and maximum number of instances that the group should have along with the desired number of instances which the group must have at all times.

4. When we completes with creating launch configuration, it will take us to Step 2, Here we have to give a name for the Group, We can select the Group size and VPC.



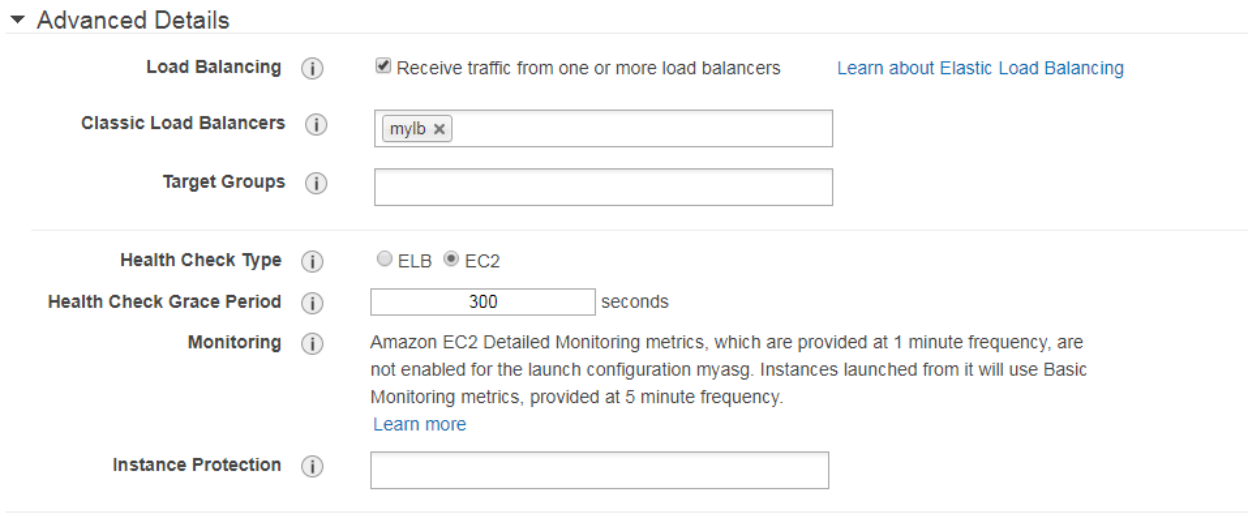
Each instance in this Auto Scaling Group will be provided with a public IP address.

5. We can expand Advanced details option to configure.

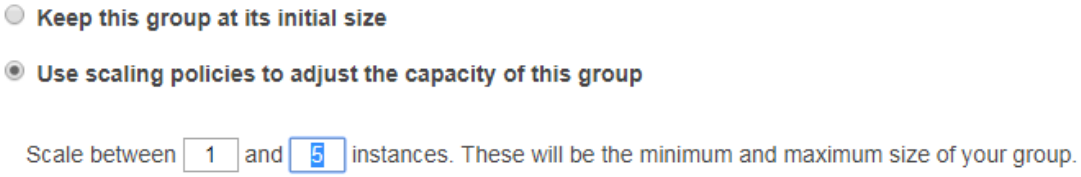
Load Balancing: These are optional settings that you can configure to work with your Auto Scaling Group. Since we have already created and configured our ELB, we will be using that itself to balance out incoming traffic for our instances. Select the Receive traffic from Elastic Load Balancer option.

Health Check Type: You can use either your EC2 instances or even your ELB as a health check mechanism to make sure that your instances are in a healthy state and performing optimally. By default, Auto Scaling will check your EC2 instances periodically for their health status. If an unhealthy instance is found, Auto Scaling will immediately replace that with a healthy one.

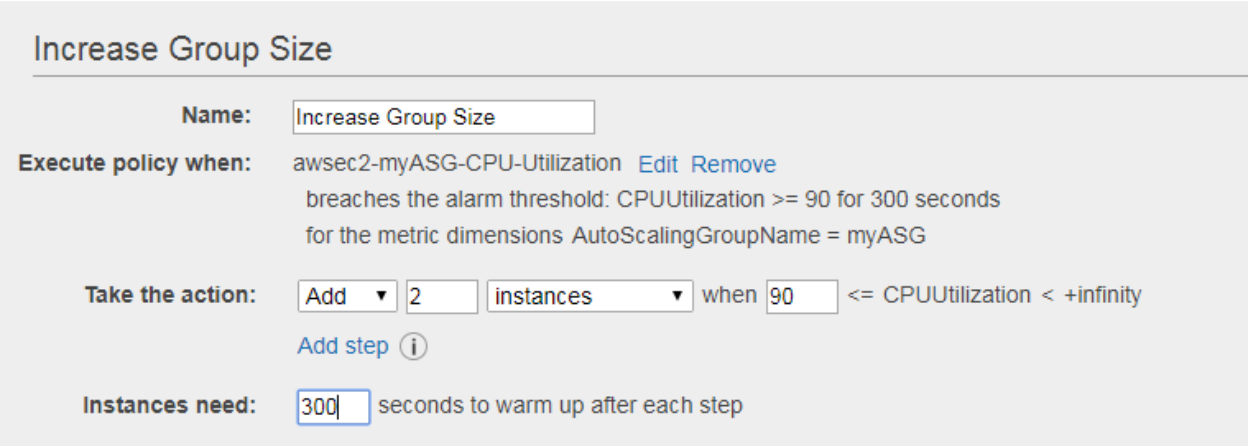
Health Check Grace Period: Enter the health check's grace period in seconds. By default, this value is set to 300 seconds.



6. Step 2 of ASG creation is Configure scaling policies: This is the important part of creating any Auto Scaling Group is defining its scaling policies.



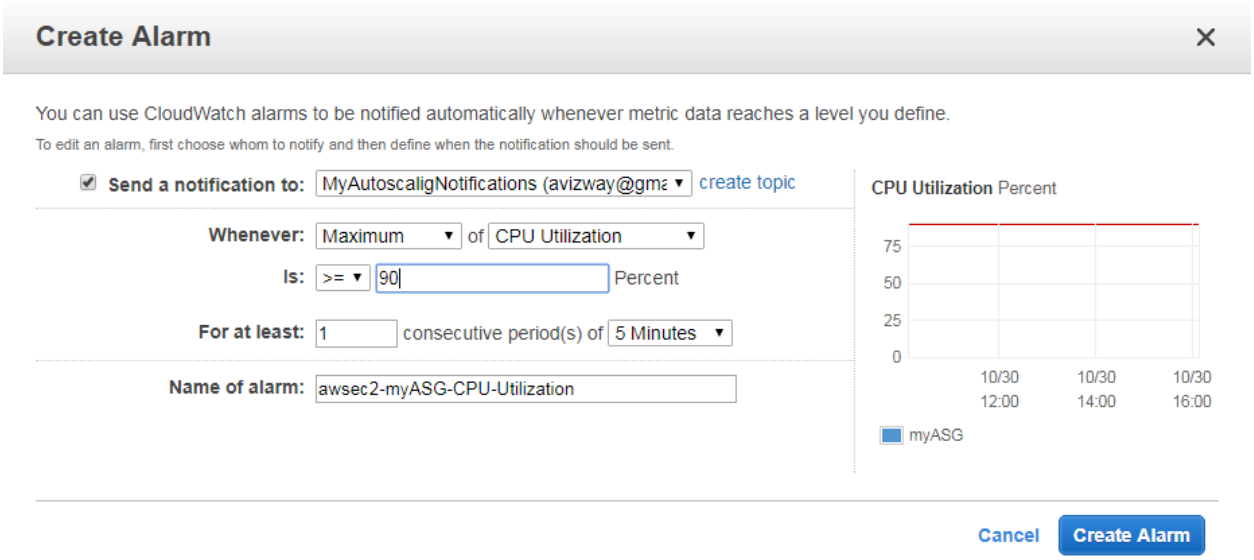
7. Selecting the scaling policies option.



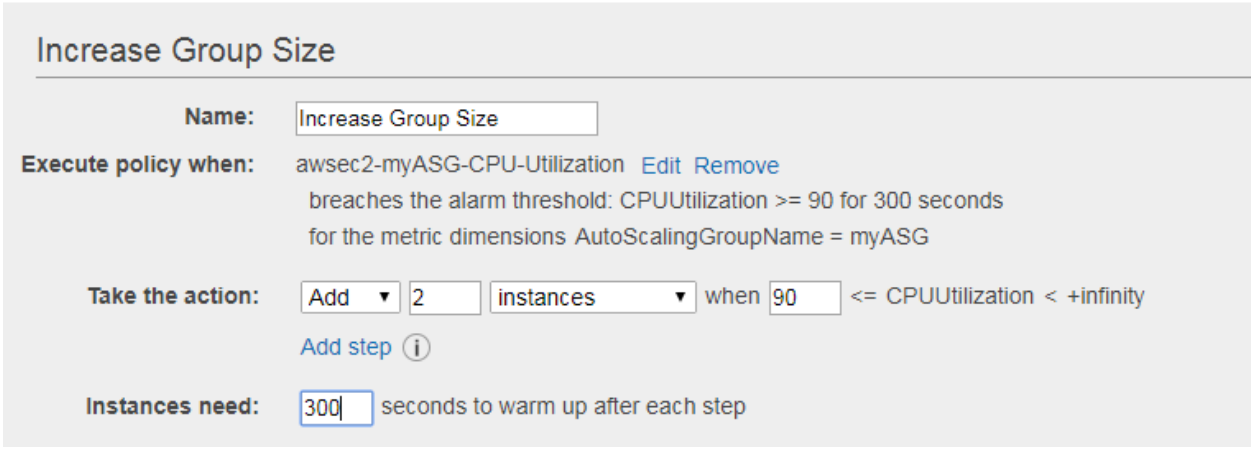
Name: Provide a suitable name for your scale-out policy.

Execute policy when: Here we have to select a pre-configured alarm using which the policy will get triggered. Since this is our first time configuring, select the **Add new alarm** option. This will pop up the Create Alarm dialog,

Creating the alarm is a very simple process; for example, we want our Auto Scaling Group to be monitored based on the CPU Utilization metric for an interval of 5 minutes. If the average CPU Utilization is greater than or equal to 90 percent for at least one consecutive period, then send a notification mail to the specified SNS Topic. click on Create Alarm.



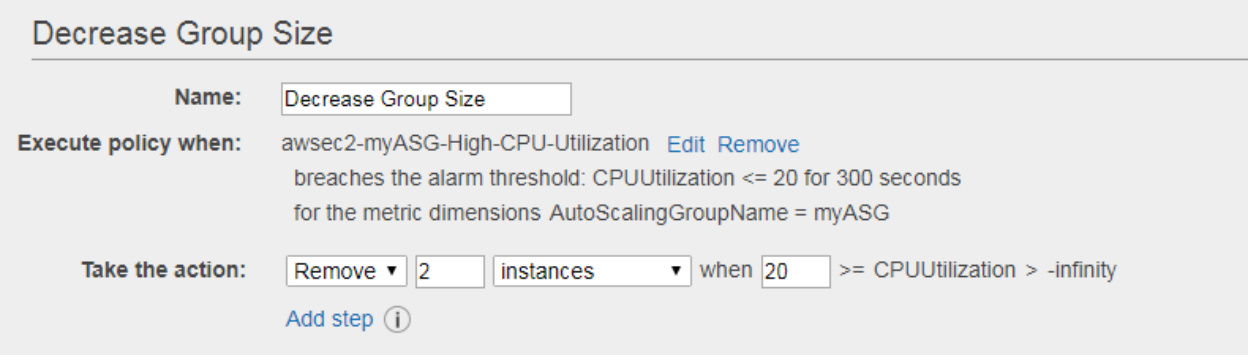
Take the action: Now we can define the policy what action it has to take if the particular threshold is breached. Select Add from the dropdown list and provide a suitable number of instances that you wish to add when a certain condition matches.



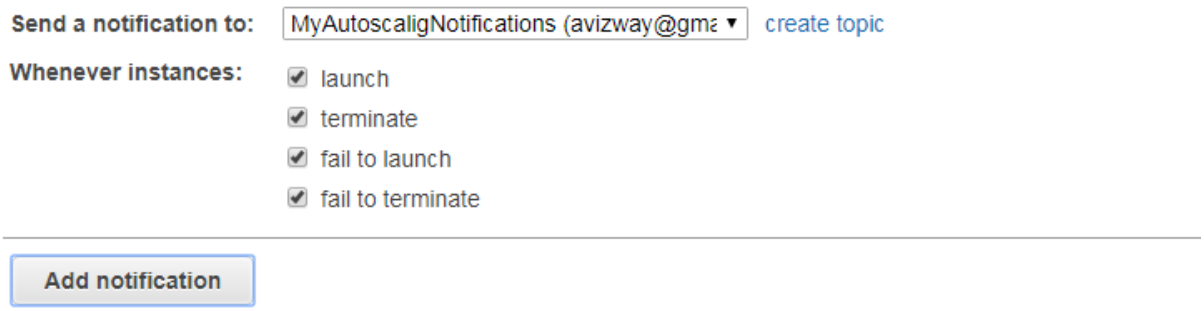
Instances need: The final field is the Cooldown period. By default, this value is set to 300 seconds and can be changed as per your requirements. A Cooldown period is like a grace period that we assign to the Auto Scaling Group to ensure

that we don't launch or terminate any more resources before the effects of previous scaling activities are completed.

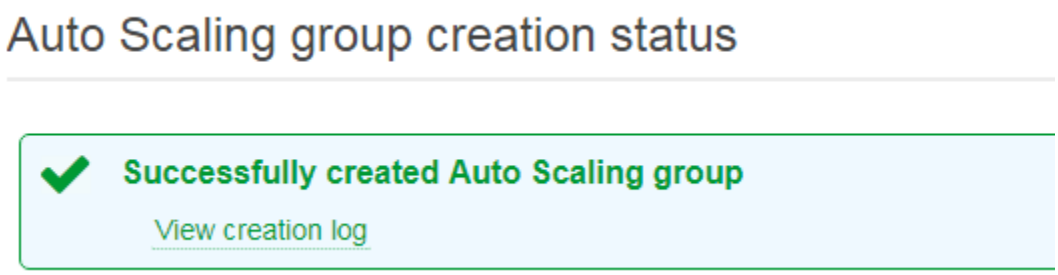
8. By the same way we can configure policies for Decrease Group Size also

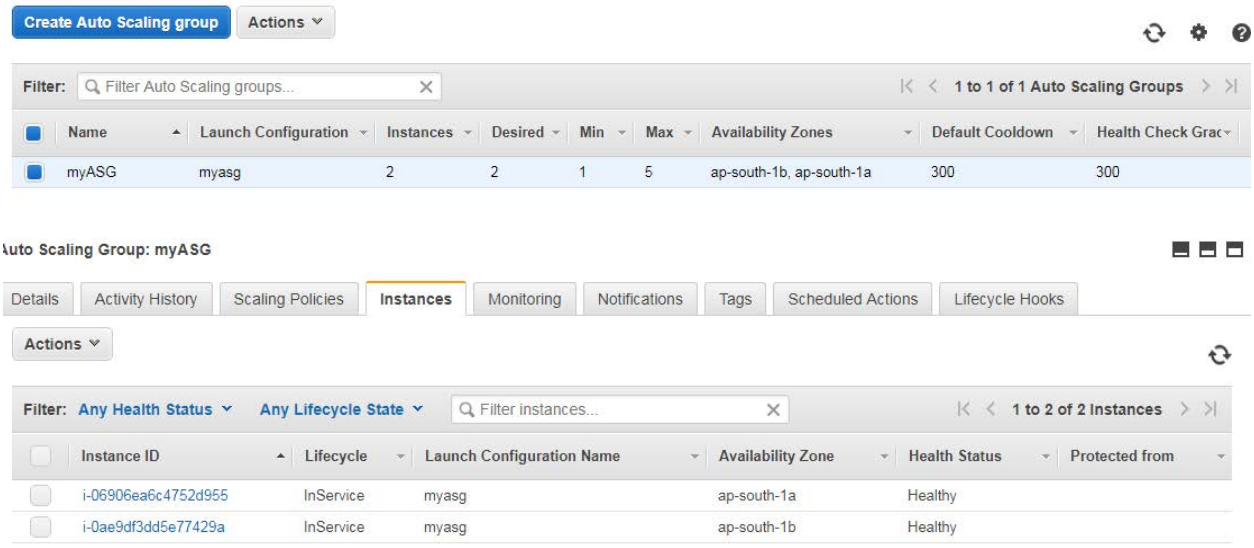


9. Select the **Next: Configure Notifications** to proceed with the next steps
10. You can select Add Notification button and select an existing SNS topic or create a new.



11. Select the review option and Click on Create Auto Scaling option to finish the process.





USER DATA:

When you launch an instance in Amazon EC2, you have the option of passing user data to the instance that can be used to perform common automated configuration tasks and even run scripts after the instance starts.

You can pass two types of user data to Amazon EC2: shell scripts and cloud-init directives. You can also pass this data into the launch wizard as plain text, as a file (this is useful for launching instances using the command line tools), or as base64-encoded text.

Here is a simple User Data script to use with **Linux EC2 instances** to make as a simple webserver with a simple index.html page.

```
#!/bin/bash
yum update -y
yum install httpd -y
echo "Hi This is a Bootstrap script generated webpage" > /var/www/html/index.html
service httpd start
chkconfig httpd on
```

"yum update" for updating the Operating system with latest security patches.

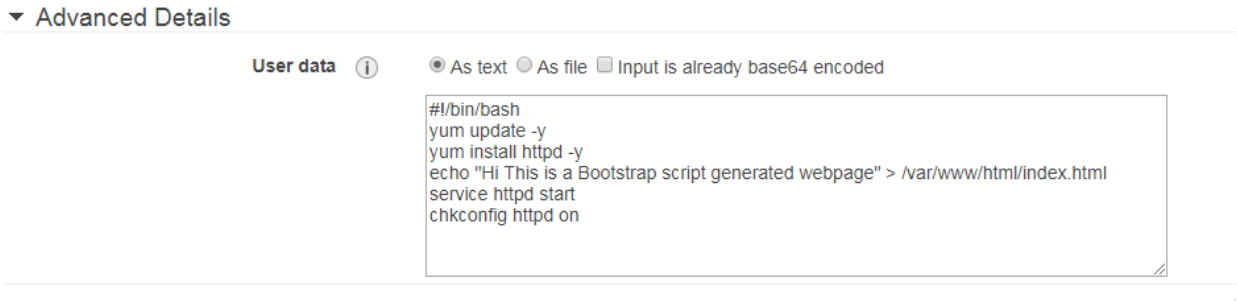
"Yum install httpd" for installing Apache to make this instance as a webserver

By Using echo command generating a string and copying the generated string to a file named "index.html" and saving the file under "/var/www/html" directory.

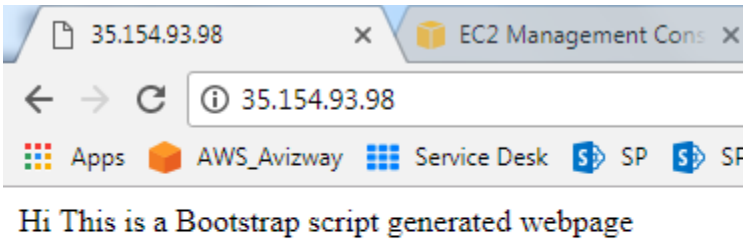
"Service httpd start" to start the apache service

"Chkconfig httpd on" starting and turning the service on / startup service.

1. While launching instance I've entered the bootstrap scripting



2. Then launching the instance and entering the public IP in the web browser without connecting to my instance. (Make sure port 80 open in the Security groups)



3. We got the output without login to the instance.

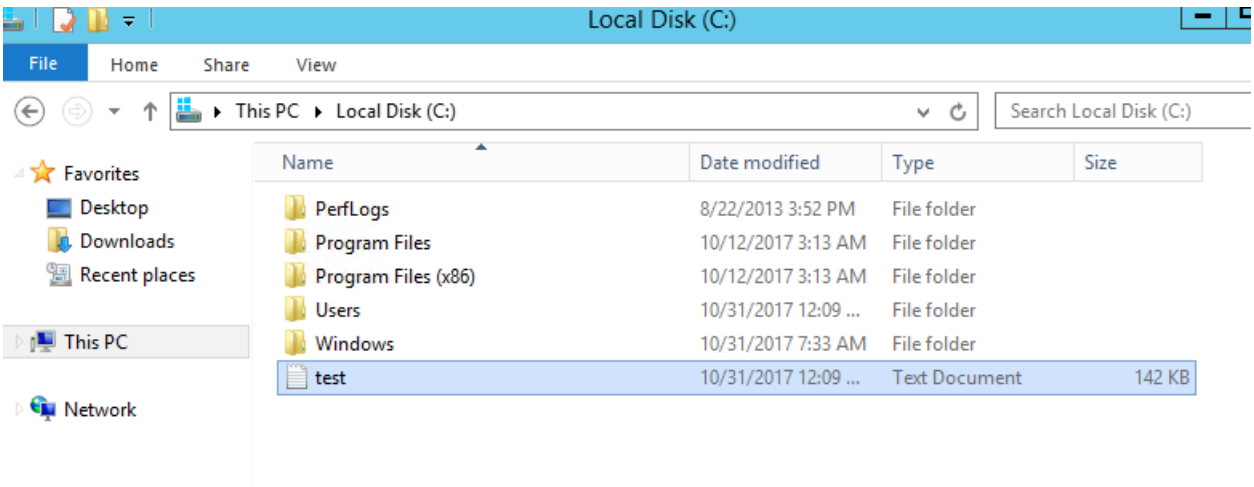
For Windows:

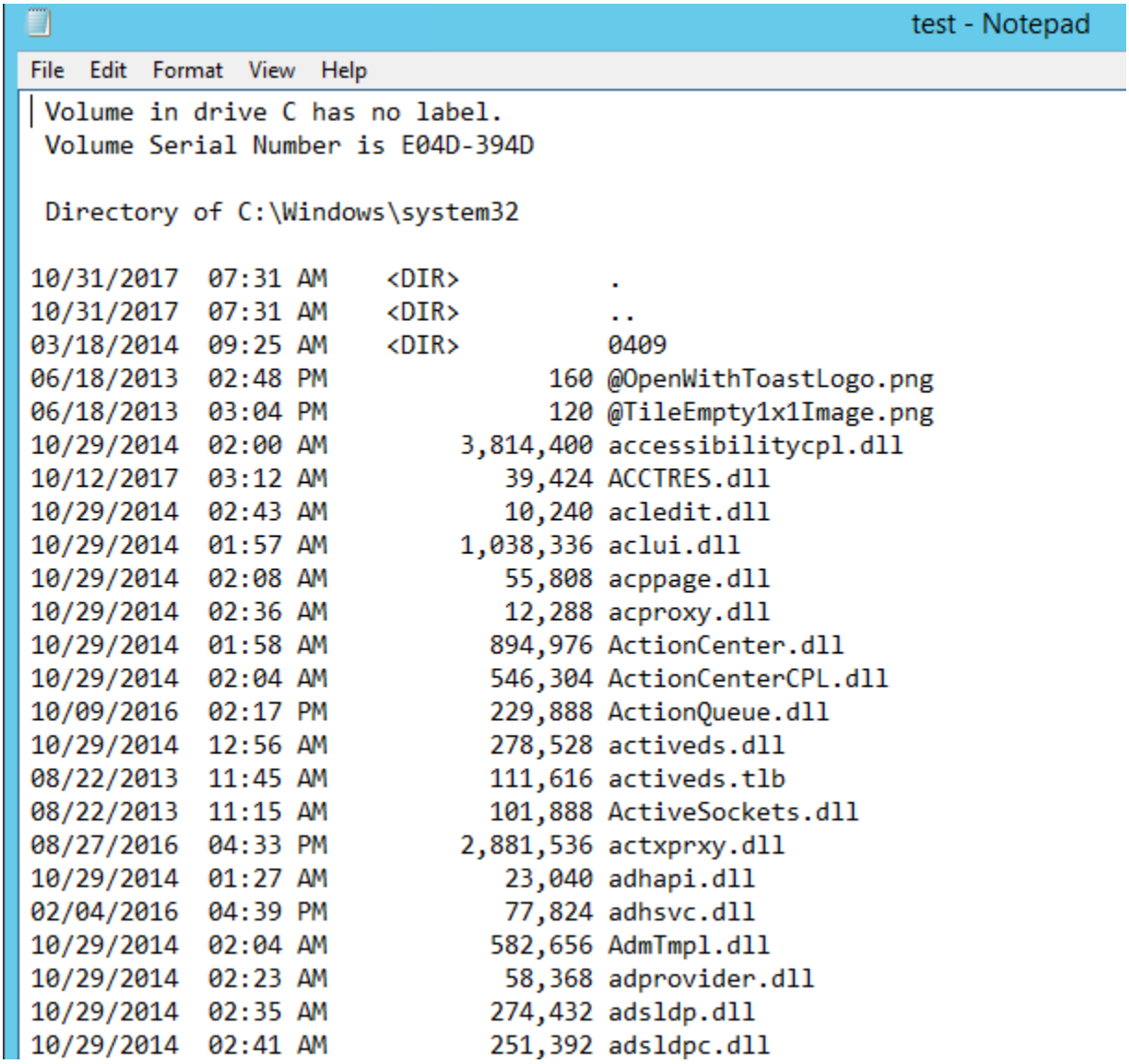
For EC2Config or EC2Launch to execute user data scripts, you must enclose the lines of the specified script within one of the following special tags:

<script> </script>

Example: <script>dir > c:\test.log</script>

1. Here we have run very simple script get directory information to a log file. New doc is created with all the information of the given directory.
2.





AWS CLI (Command Line Interface):

The AWS Command Line Interface (CLI) is a unified tool to manage AWS services. With just one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.

- We can download the AWS tolls by using this URL: https://aws.amazon.com/cli/
- You can select the setup file based on your system architecture, if you are a windows user.
- Amazon Linux will get the CLI tools pre-installed.

Windows

Download and run the 64-bit or 32-bit Windows installer.

Mac and Linux

Requires Python 2.6.5 or higher.

Install using pip.

```
pip install awscli
```

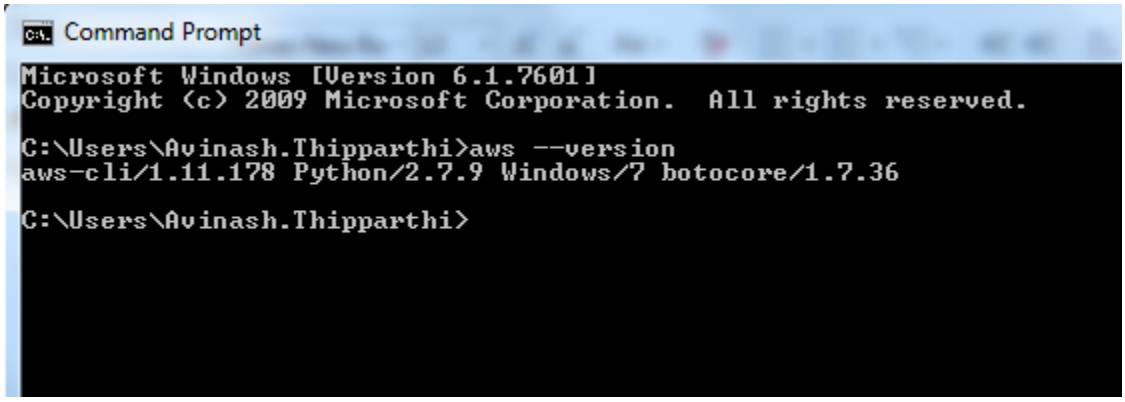
Amazon Linux

The AWS CLI comes pre-installed on Amazon Linux AMI.

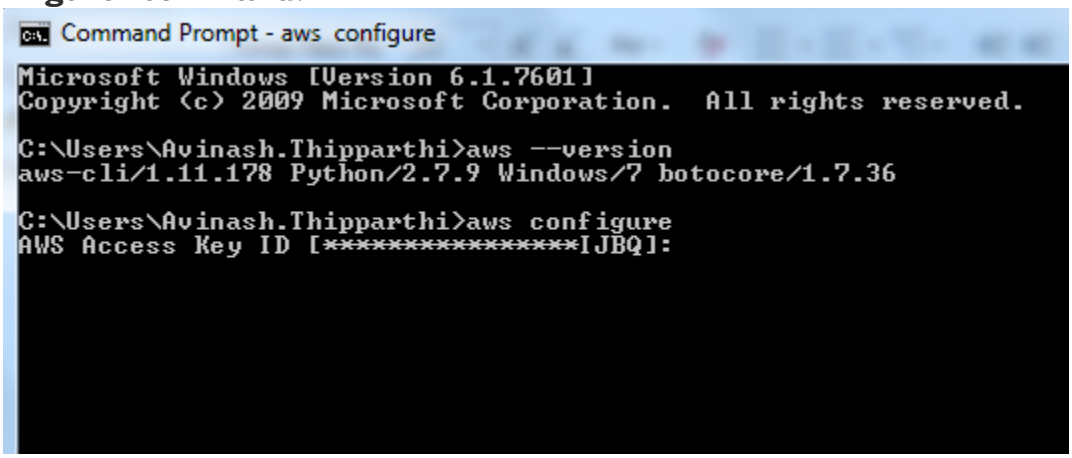
- Here is the url to get all the commands for each and every AWS service: http://docs.aws.amazon.com/cli/latest/reference/

Steps to configure CLI tools on windows Operating systems:

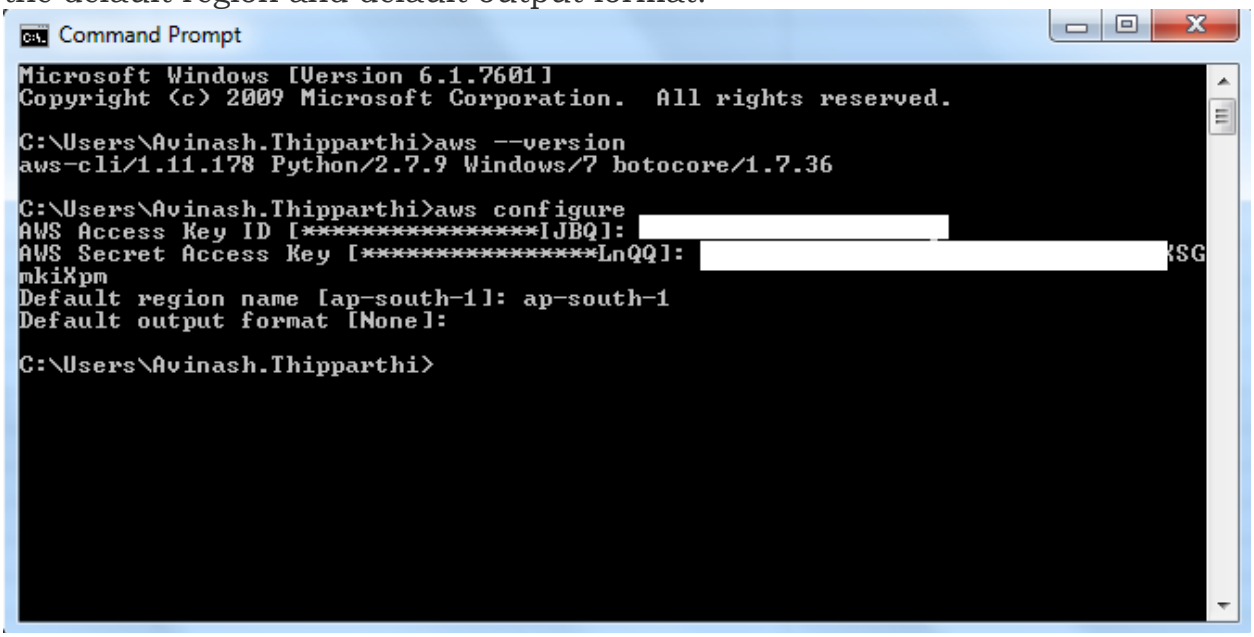
1. First we have to download the setup file from the above mentioned webpage, then follow the simple installation wizard.
2. After installing these tools, we can use the windows command prompt to connect to AWS resources/services.
3. To verify CLI tools installation, open command prompt and enter **"AWS –version",** it should return with installed version information as the below image.



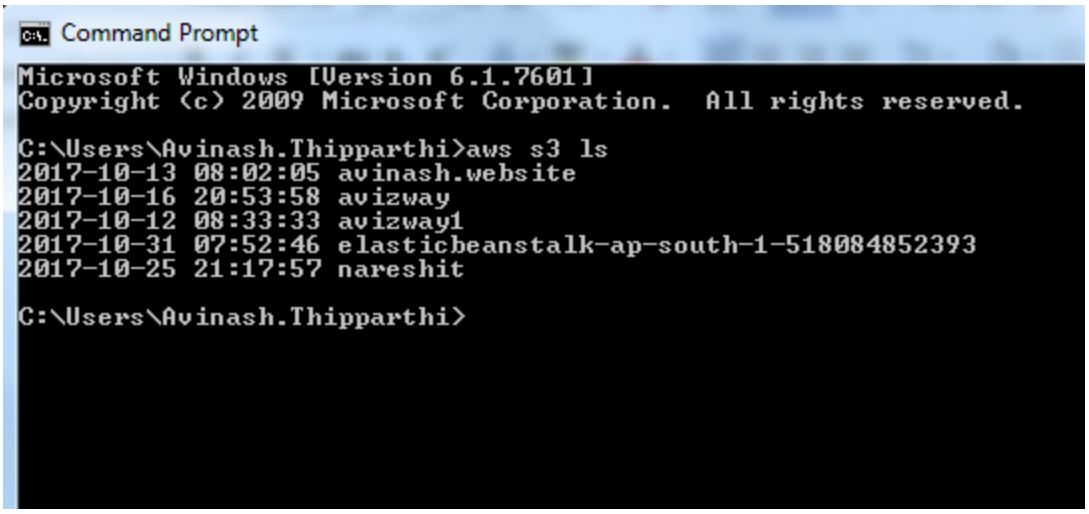
4. But we cannot configure CLI tools using IAM Management console access users, we need to have Programmatic Access IAM user.
5. When we create a Programmatic Access IAM user we will get **Access key ID** and **Secret Access Key**. Please create a user and allocate appropriate permissions.
6. To configure IAM user in local windows machine, we have to "**AWS configure**" command.



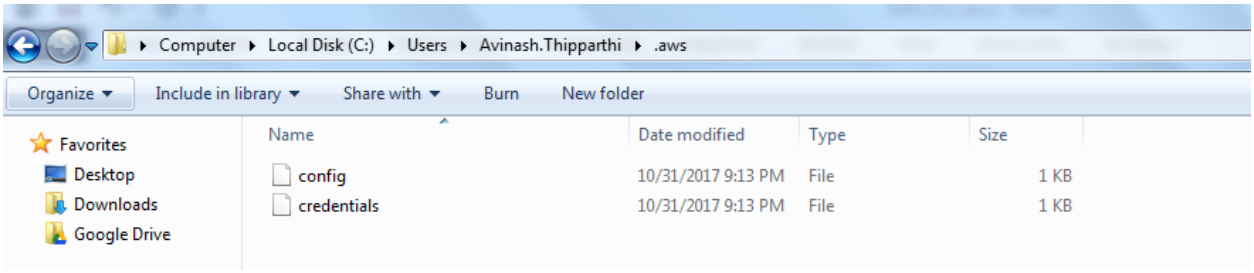
7. Enter the AWS Access Key ID and then enter the Secret Access key, choose the default region and default output format.



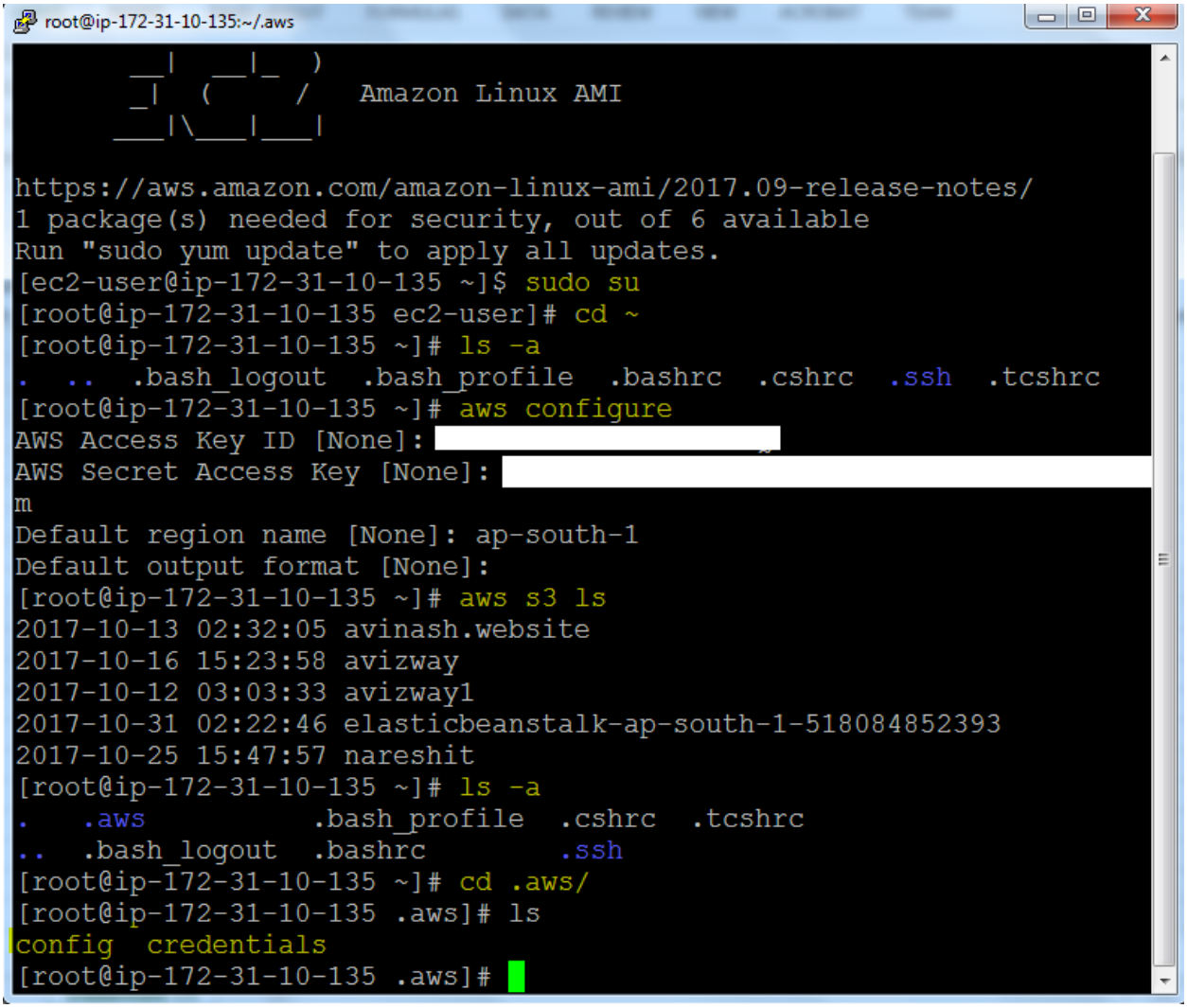
8. We have successfully configured the CLI tools and now try to access any of the AWS resource from the CLI configured device. Here am trying to list my S3 buckets for that am using **aws s3 ls** command.



9. We are able to get the details that means we are connecting to AWS account resources by using the Programmatic access IAM user credentials.
10. But, the IAM user credentials will store in a directory called .aws , In windows the path is **C:\Users\WindowsUserName\.aws** , if you open credentials file, we will get the Configured IAM user's Aceess Key ID and Secret Access Key.



11. In Linux, The .aws directory will store under / (root) and It is a hidden directory, we can give **ls –a** command to get it, and inside the .aws directory we will have config and credentials files.



12. In the above image, I've logged into the linux instance and switched to root, looked for .aws directory, but it is not existed. Then Configured the IAM user with Access Key IA and Secret Access Key and accessed the AWS resources and we get the required resource information.
13. After installing CLI IAM user, we got .aws directory under **/** (give **ls –a** to verify), inside that .aws directory we have config and credentials files, Credential file will contains the Access Key id and secret access key.
14. So this is not a secure method, anybody can view these credentials and configure CLI tools on their own machines and they may access, So amazon will **recommend to use the ROLES** instead of storing the credentials in local machines.

Policy: A policy is a JSON document that fully defines a set of permissions to access and Manipulate AWS resources. Policy documents contain one or more permissions.

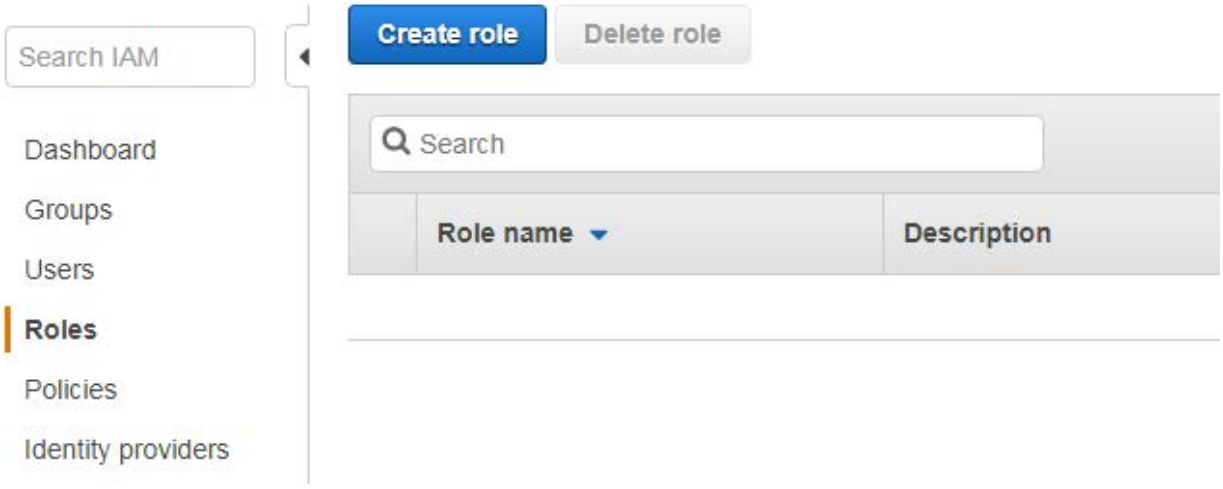
IAM ROLES:

Roles are used to allow AWS services to perform actions on your behalf. Roles are used to grant specific privileges to specific actors.

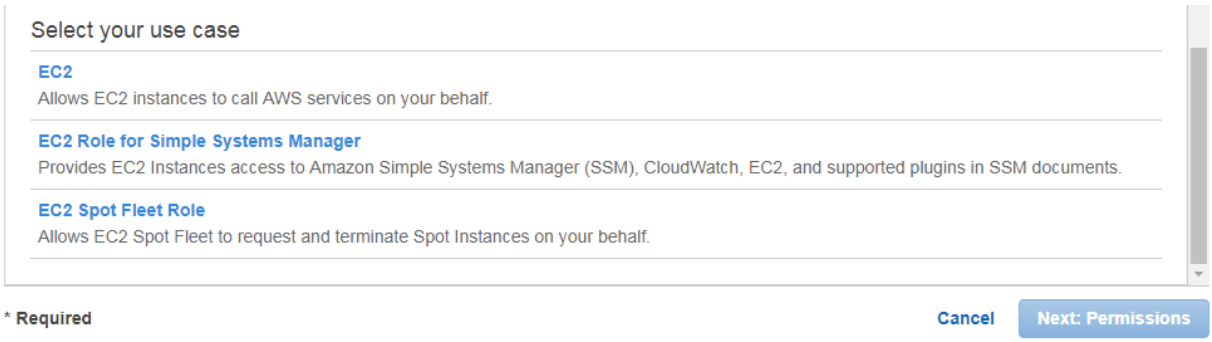
- Roles are more secure than storing your access key and secret access key on individual EC2 instances.
- Roles are easier to manage
- We can attach or Remove role to a running instance now. Previously this option is not available.
- Roles are universal, you can use them in any region.

Steps to create a role and attaching to EC2 instance.

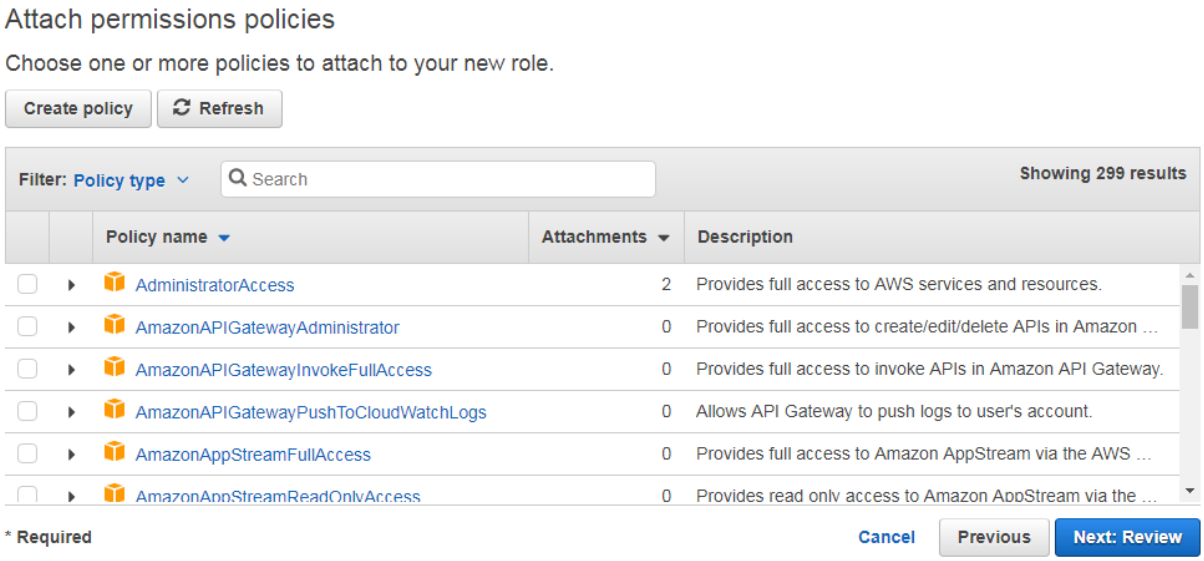
1. Navigate to IAM dashboard to create an IAM role.
2. Select Roles option from dashboard and select **"Create Role"** option.



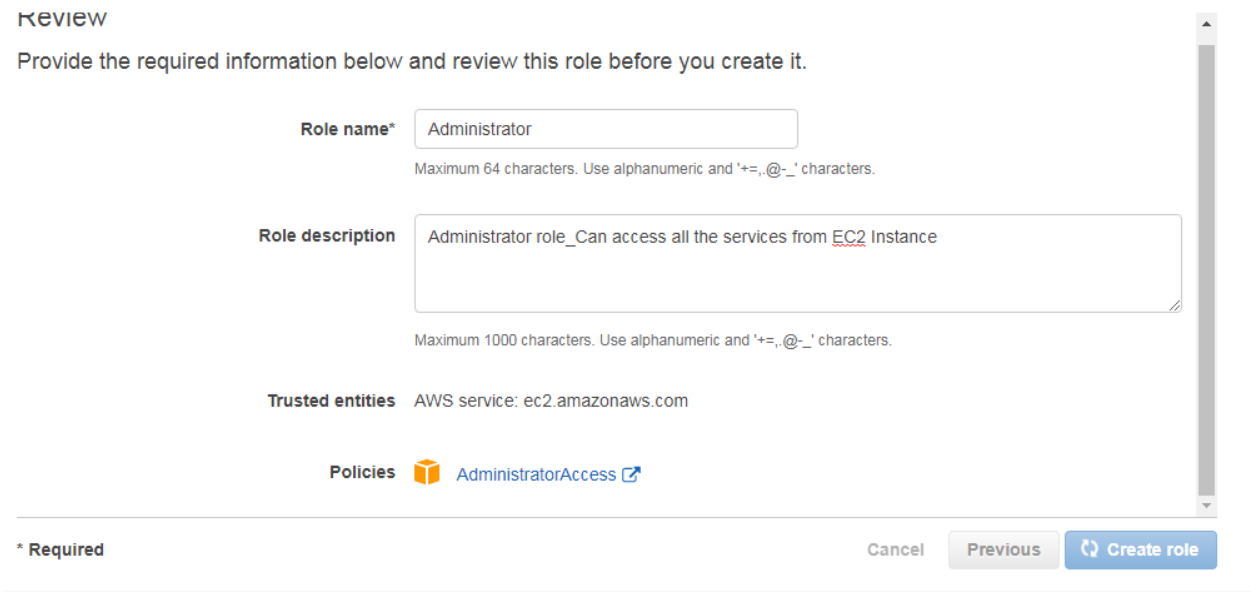
3. We have four option in the roles, We are going to create this role under "AWS Services", and select the **EC2.**
4. After selecting EC2, we have to select the appropriate Use Case. We would like to call some AWS services on our behalf to the EC2 instance. Select EC2 and click on **Next: Permissions** button.



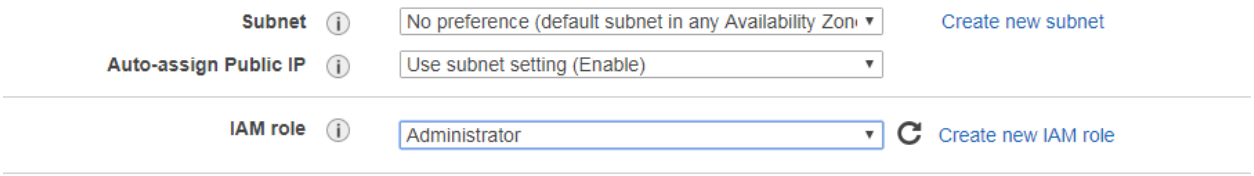
5. In this step, we have to select the policy, you can generate a new policy based on your requirement or choose existing policy.



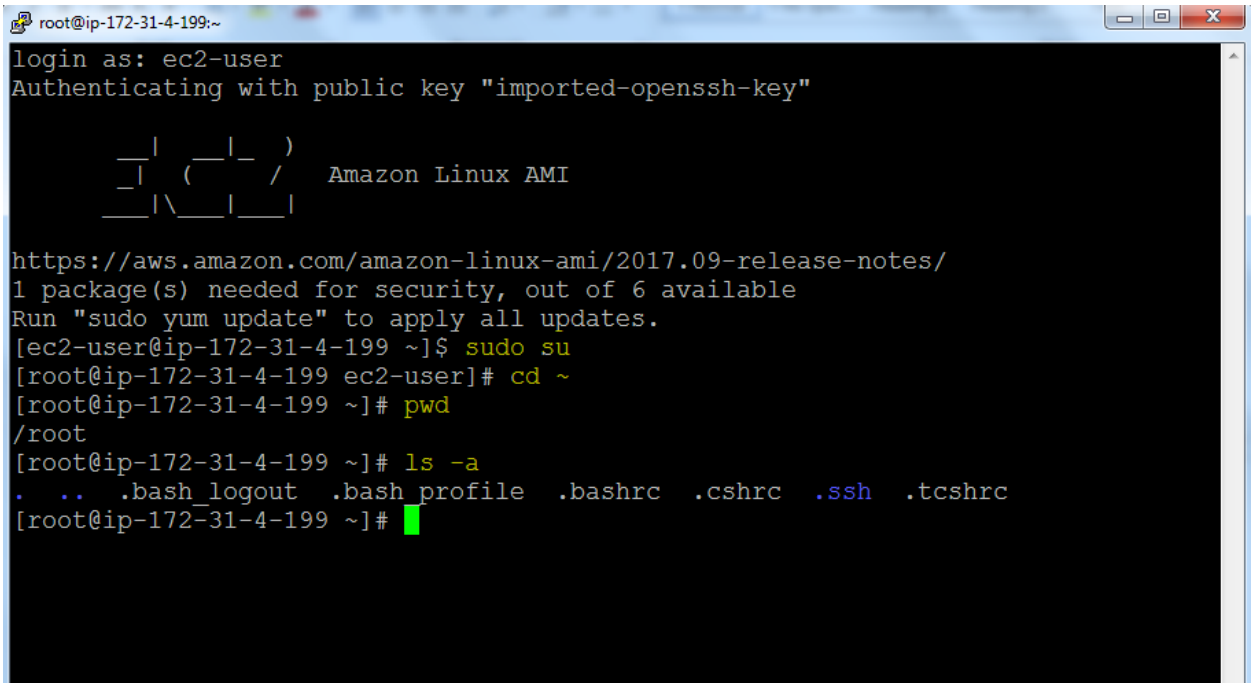
6. Select appropriate role, based on your requirement, am selecting AdministratorAccess role here. Then Select **Review.**
7. In review page, Give a name for the role and a valid description and select **Create Role** option.



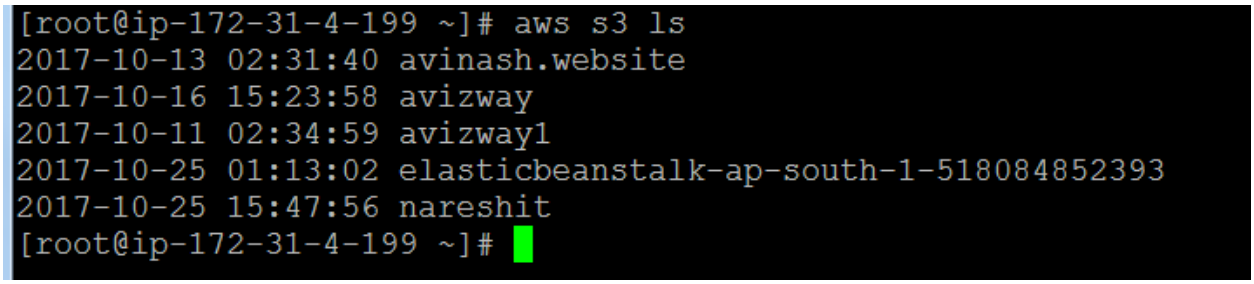
8. Now launch an EC2 instance and try to access/call any AWS service to verify the role.



9. Logged into EC2 instance and elevated privileges to root and trying to find the .aws directory under / , but we cannot find, That means we don't have any credentials on instance.



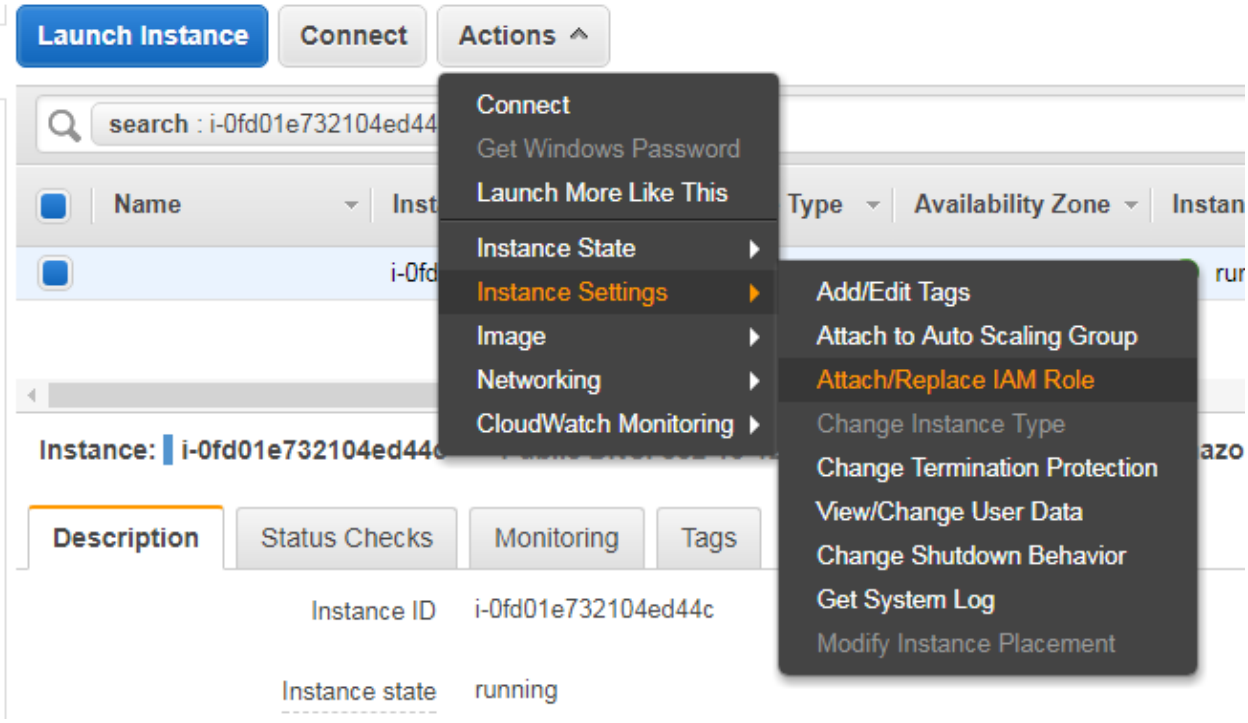
10. Try to access any AWS service, here am trying to list the S3 buckets by **AWS s3 ls** command.



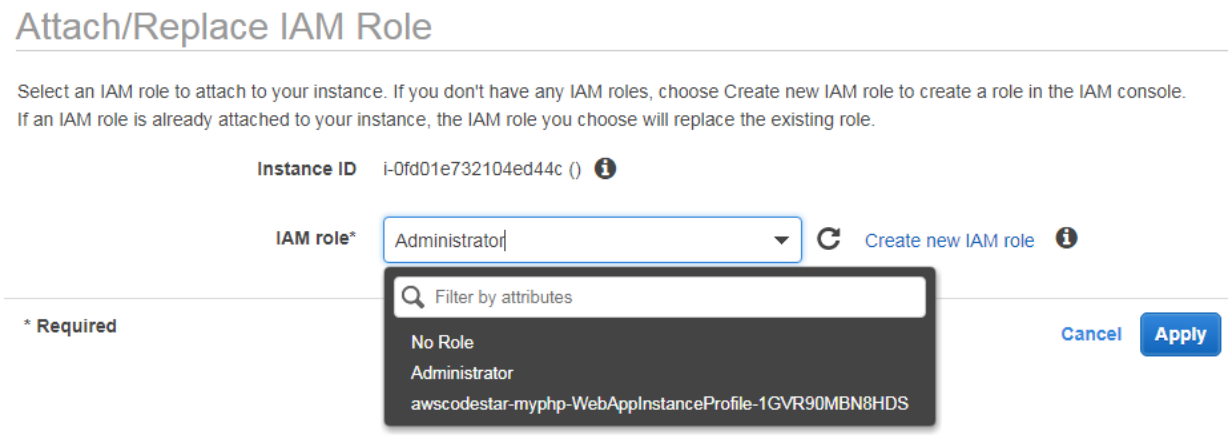
11. We are able to access the resources and nowhere storing the Access key ID and Secret Access key.

Steps to Attach/Replace role from a Running Instance:

1. Select the Instance and go to Actions button and we can find Attach/Replace IAM Role under Instance Settings.



2. Select IAM role filed, automatically it will dropdown the available roles along with No Role option, Select the required option and click on Apply. It will take effect immediately.



Instance Metadata:

Instance metadata is data about your instance that you can use to configure or manage the running instance. This is unique in that it is a mechanism to obtain AWS properties of the instance from within the OS. By using below URL we can query the local instance metadata.

- Curl http://169.254.169.254/latest/meta-data/
- When you enter this URL, it'll return with all the available information to get. We can give the required option after meta-data/ you'll get the information.

Steps to get the instance Metadata:

1. I've logged into my EC2 instance
2. Enter the metadata url

```
[root@ip-172-31-23-113 ec2-user]# curl http://169.254.169.254/latest/meta-data/
ami-id
ami-launch-index
ami-manifest-path
block-device-mapping/
hostname
instance-action
instance-id
instance-type
local-hostname
local-ipv4
mac
metrics/
network/
placement/
profile
public-hostname
public-ipv4
public-keys/
reservation-id
security-groups
services/[root@ip-172-31-23-113 ec2-user]#
```

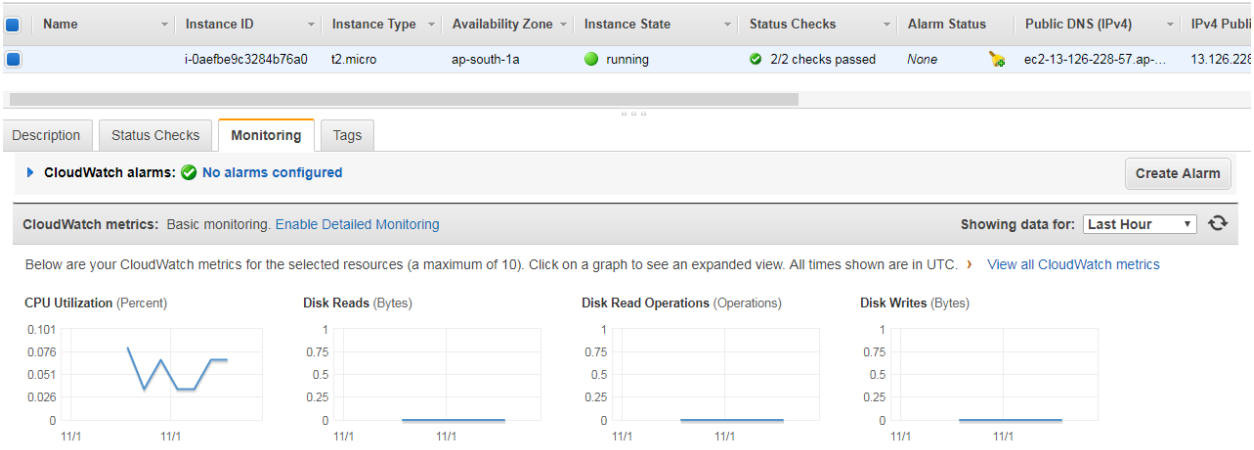
3. It is returned with all the available option, now whatever the information you want to get, give it along with the URL.
   Ex: if you want to know hostname, give as Curl http://169.254.169.254/latest/meta-data/hostname

```
[root@ip-172-31-23-113 ec2-user]# curl http://169.254.169.254/latest/meta-data/hostname
ip-172-31-23-113.ap-south-1.compute.internal[root@ip-172-31-23-113 ec2-user]#
```

AWS CLOUDWATCH

Amazon CloudWatch is a service that you can use to monitor your AWS resources and your applications in real time. With Amazon CloudWatch, you can collect and track metrics, create alarms that send notifications, and make changes to the resources being monitored based on rules you define.

- You can specify parameters for a metric over a time period and configure alarms and automated actions when a threshold is reached.
- Amazon CloudWatch offers either basic or detailed monitoring for supported AWS products.
- Basic monitoring sends data points to Amazon CloudWatch every five minutes for a limited number of preselected metrics at no charge.
- Detailed monitoring sends data points to Amazon CloudWatch every minute and allows data aggregation for an additional charge. If you want to use detailed monitoring, you must enable it—basic is the default.
- AWS provides a rich set of metrics included with each service, but you can also define custom metrics to monitor resources and events.
- Amazon CloudWatch Logs can be used to monitor, store, and access log files from Amazon EC2 instances.
- Amazon CloudWatch Logs can also be used to store your logs in Amazon S3 or Amazon Glacier.
- Each AWS account is limited to 5,000 alarms per AWS account, and metrics data is retained for two weeks by default.



Sample image for EC2 instance cloudwatch monitorings.

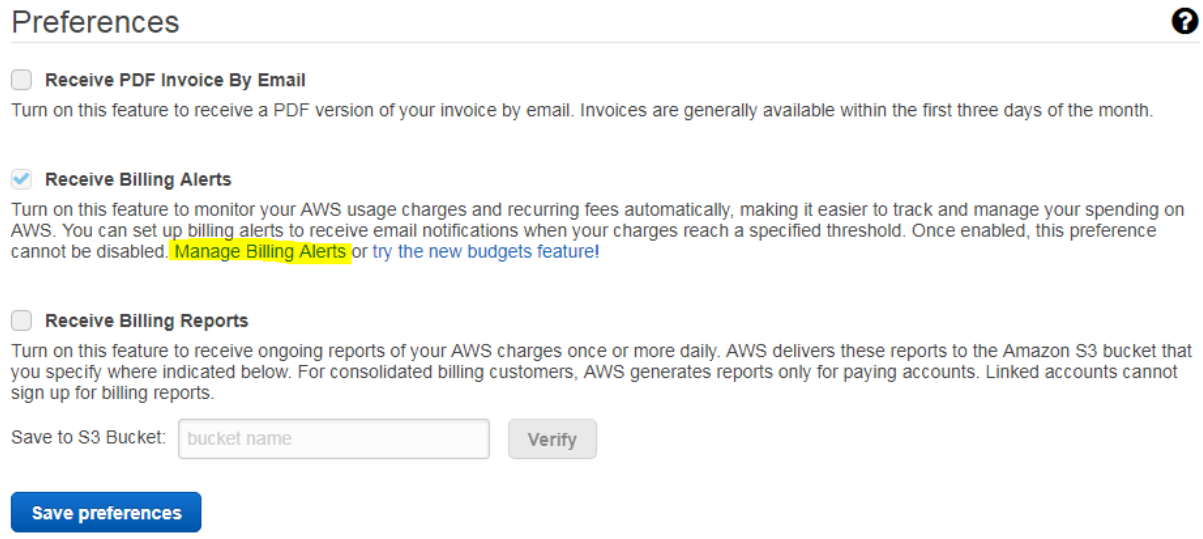
Metrics: Metrics form the core of Amazon CloudWatch's functionality. Essentially, these are nothing more than certain values to be monitored. Each metric has some data points associated with it which tend to change as time progresses.

Alarms: An alarm basically watches over a particular metric for a stipulated period of time and performs some actions based on its trigger. These actions can be anything from sending a notification to the concerned user using the Simple Notification Service (SNS).

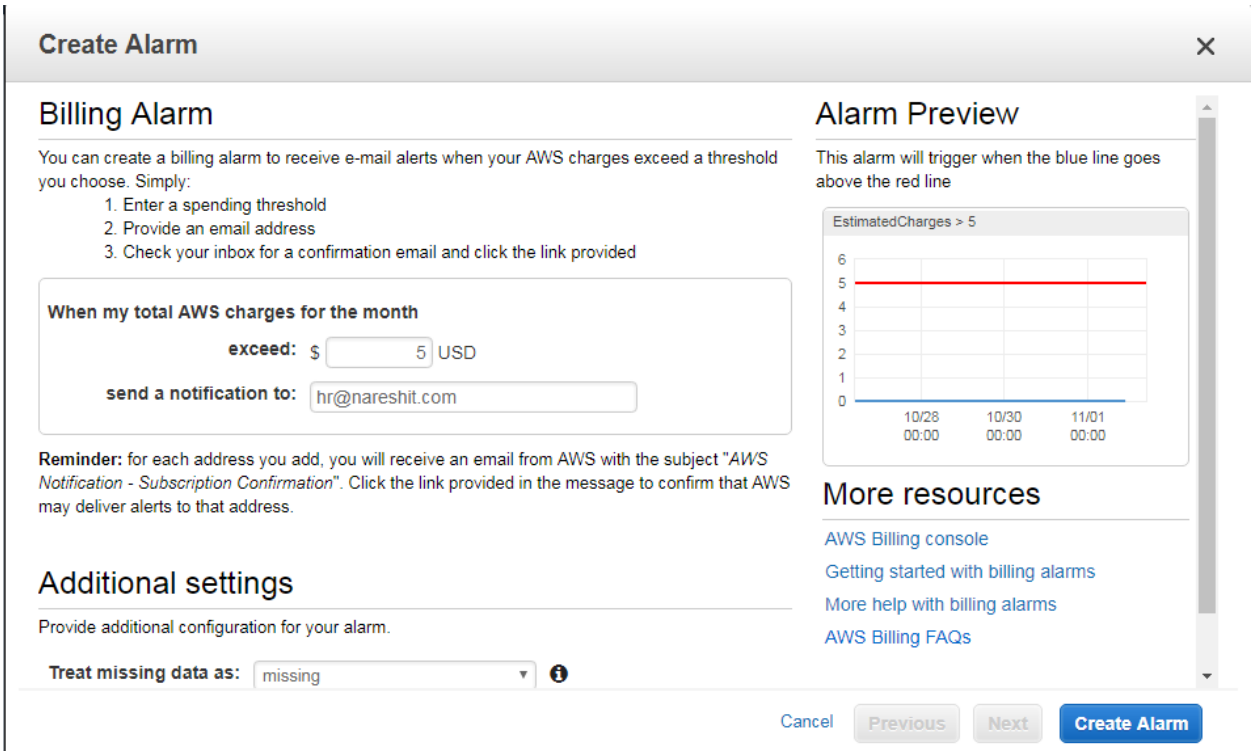
Monitoring your account's estimate charges using CloudWatch

You can configure the alerts on your AWS usage by using the Cloudwatchh alarms. Here is the steps to create an alarm on estimated charges.

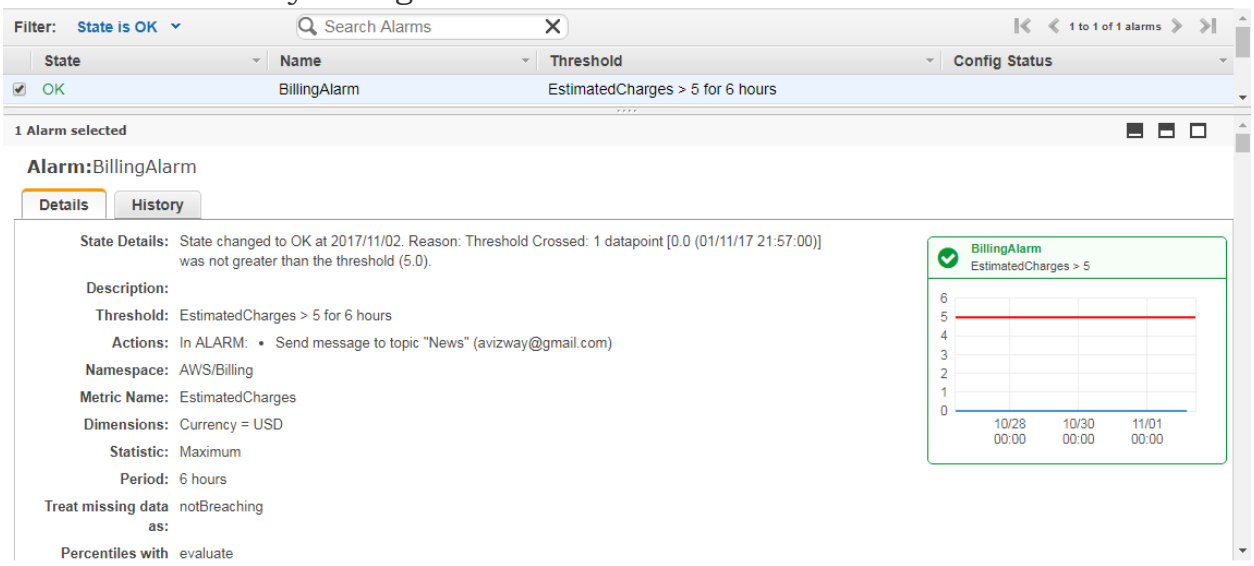
1. Login with root account credentials.
2. Select My Account option and navigate to "**Preferences**"
3. Go to Select Receive Billing Alerts checkbox and select **"Manage Billing Alerts**" option. (Cloudwatch alarms will create in North Virginia region).



4. When you click on **"Manage Billing Alerts**" option, you'll redirect to Cloudwatch dashboard, there select Create a Billing alert option. Automatically Create Alarm windows will open.



5. In this windows, enter the USD value, when you want to receive the notifications and enter your email id which you want to get the notifications, Click on **"Create Alarm"** When your monthly usage reaches to 5$ you'll get notified by the cloudwatch service through the mentioned email.
6. AWS does not allow the billing alarm's period to be set less than 6 hours. Here is how exactly billing alarm looks like.



ALARM Threshold details:

With the Alarm's threshold set, the final thing that you need to do is define what action the alarm must take when it is triggered. From the Notification section, fill out the required details, as mentioned in the following:

Whenever this alarm: This option will allow you to determine when the alarm will actually perform an action. There are three states of an alarm out of which you can select any one at a single time:

State is ALARM: Triggered when the metric data breaches the threshold value set by you

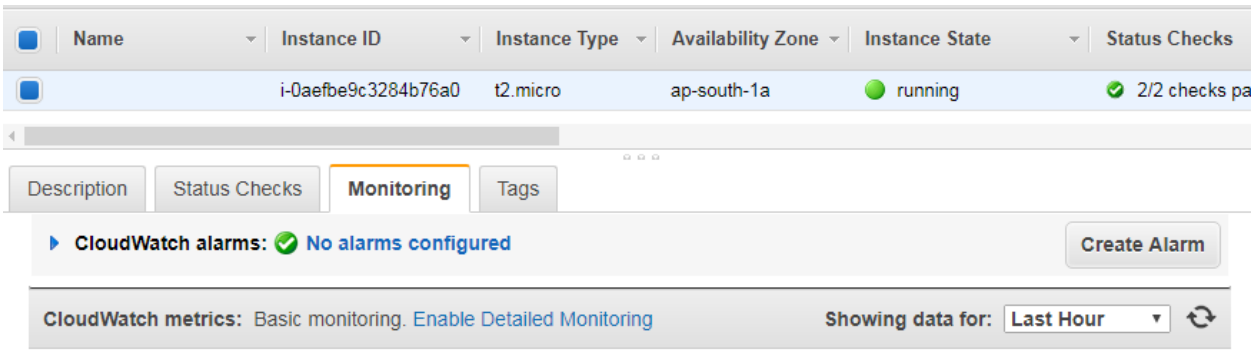
State is OK: Triggered when the metric data is well within the supplied threshold value

State is INSUFFICIENT: Triggered when the alarm generally doesn't have enough data with itself to accurately determine the alarm's state.

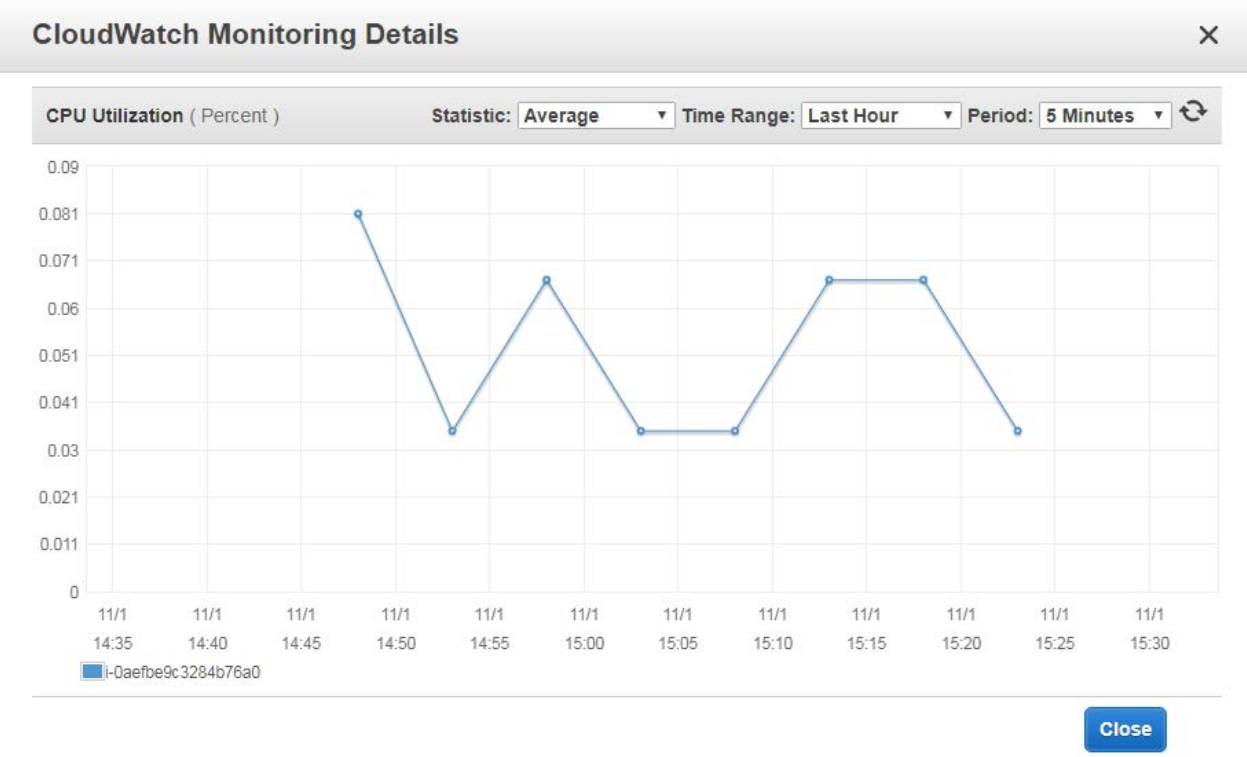
Monitoring your instance's CPU Utilization using CloudWatch

We are going to creating a simple alarm to monitor an instance's CPU utilization. If the CPU utilization breaches a certain threshold, say 75 percent, then the alarm will trigger an email notification as well as perform an additional task such as stop/restart the instance.

AWS makes creating alarms a really simple and straightforward process. The easiest way to do this is by selecting **your individual instances** from the **EC2 Management Dashboard** and selecting the **Monitoring tab**. Each instance is monitored on a five-minute interval by default. We can modify this behavior and set the time interval as low as one minute by selecting the Enable Detailed Monitoring option.

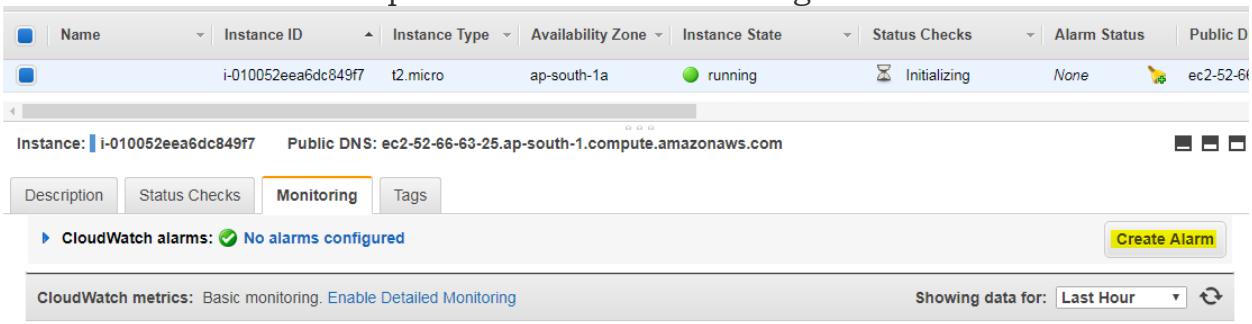


Each instance Monitoring graphs display important metric information such as CPU utilization, disk Read/Writes, bytes transferred in terms of network IO. We can expand on each of the graphs by simply selecting them.

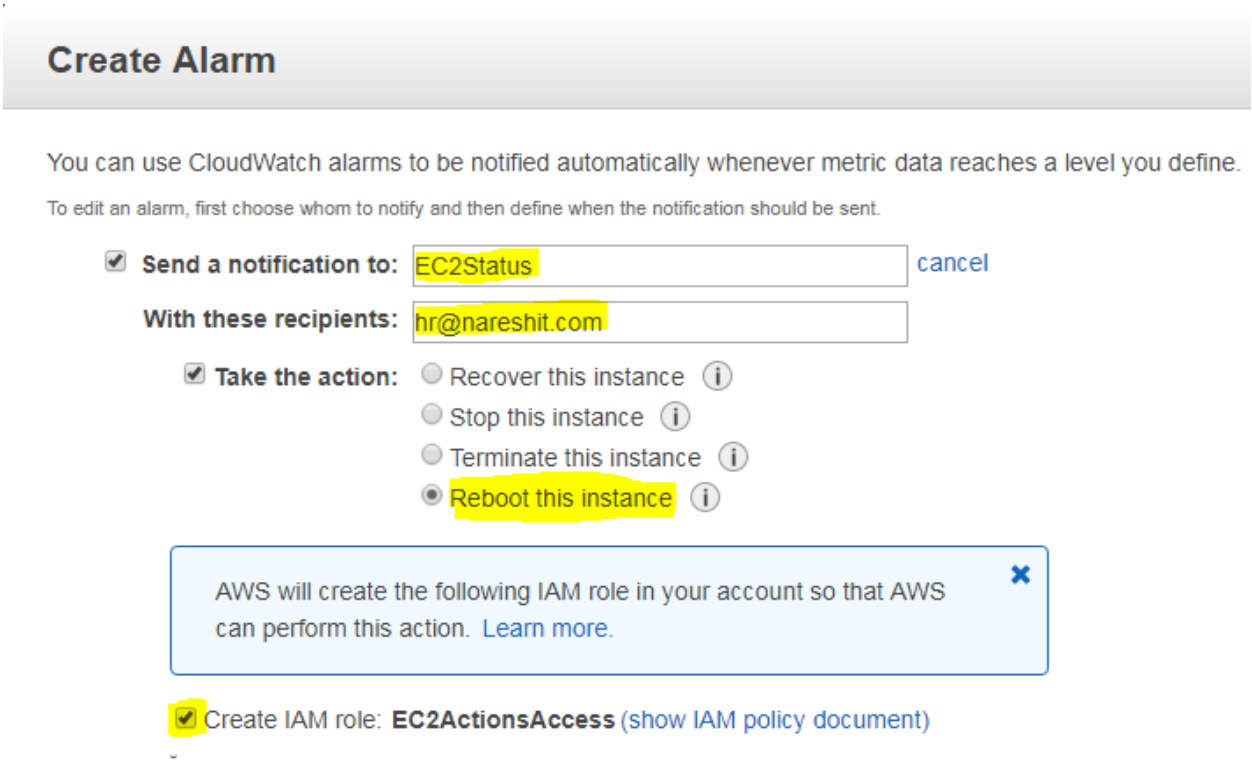


The x axis displays the CPU utilization in percent whereas the y axis display the time as per the current period's settings. We can view the individual data points and their associated values by simply hovering over them on the graph. Alternatively, you can also switch between the Statistics, Time Range, and Period as per our requirements.

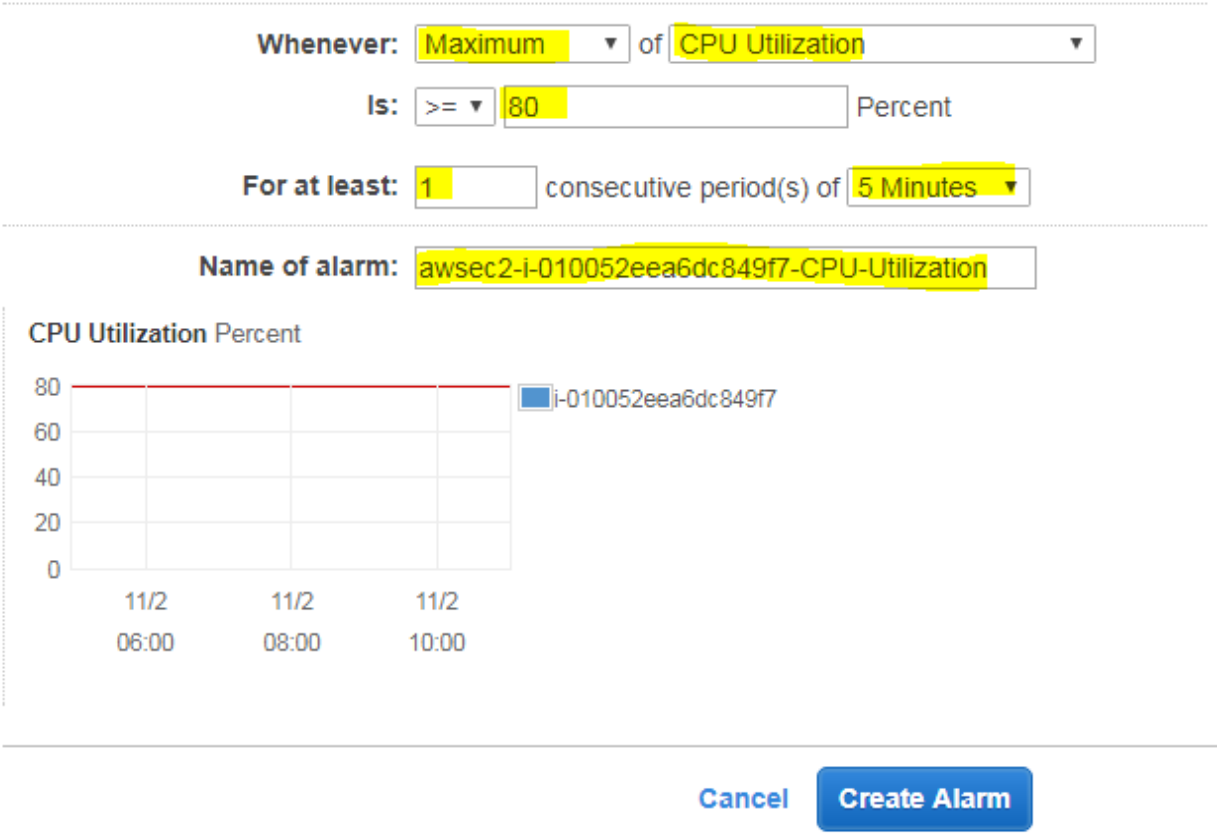
1. Once you have viewed your instance's performances, you can create a simple alarm by selecting the Create Alarm option provided in the Monitoring tab.
2. Click on **Create Alarm** option as shown below image.



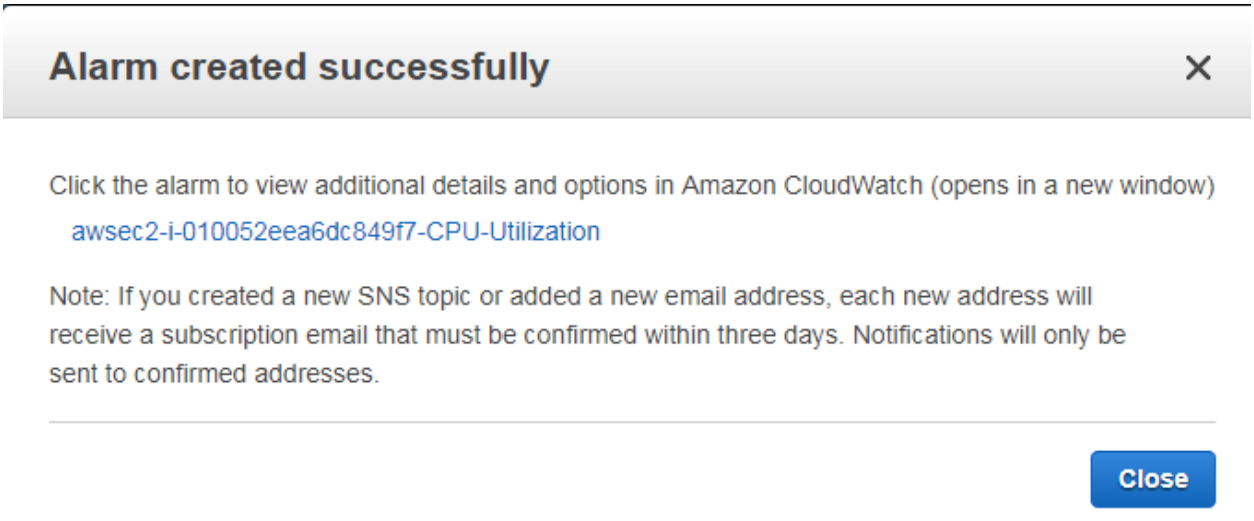
3. Now you'll get a windows with all the available options to create an alarm.



- If you want to get the notifications to an email ID, we need to depend on another service called SNS, click on **"Create topic on Send notifications to"** Then give a name for the topic. Enter a valid email to get the notifications in "**With these recipients field"**.
- Select the Take the action, what action you want to perform on instance, when the alarm matches with the defined threshold. In this case am selecting **Reboot this instance** option. (Criteria am mentioning is when CPU utilization >80 % for consecutive of 5 minutes).
- To perform this action, we have to create a **role**, If we have any existing role, we can attach it, otherwise select the option **"Create IAM role".**



- Here am defining the thresholds about the alarm, Whenever **Maximum of CPU Utilization** is **>= 80** Percent for at least **1** consecutive period of **5 Minutes.**
- Then allocating a name for this Alarm.



- Alarm created successfully, we can verify the same from.
- We have 1,377 Metrics till date. We can use any of the one.

Dashboard: Dashboard is a centralized place to monitor all your resources.
Free Tier

- New and existing customers also receive 3 dashboards of up to 50 metrics each per month at no additional charge. ($3.00 per dashboard per month after that)
- Basic Monitoring metrics (at five-minute frequency) for Amazon EC2 instances are free of charge, as are all metrics for Amazon EBS volumes, Elastic Load Balancers, and Amazon RDS DB instances.
- New and existing customers also receive 10 metrics, 10 alarms and 1 million API requests each month at no additional charge.

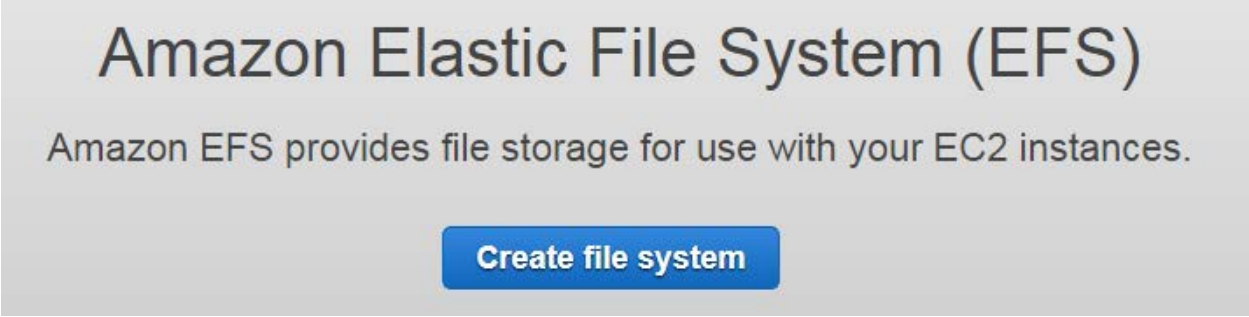
ELASTIC FILE SYSTEM (EFS)

- Amazon EFS is easy to use and offers a simple interface that allows you to create and configure file systems quickly and easily. With Amazon EFS, storage capacity is elastic, growing and shrinking automatically as you add and remove files.
- Supports the Network File System version 4 (NFSv4.1) protocol.
- Multiple Amazon EC2 instances can access an Amazon EFS file system, so applications that scale beyond a single instance can access a file system.
- Amazon EC2 instances running in multiple Availability Zones (AZs) within the same region can access the file system, so that many users can access and share a common data source.
- It is also based on the pay-per-use model, which means that you only have to pay for the storage used by your filesystem
- Using Amazon EFS with Microsoft Windows Amazon EC2 instances is not supported.
- Multiple Amazon EC2 instances can access an Amazon EFS file system at the same time, allowing Amazon EFS to provide a common data source for workloads and applications running on more than one Amazon EC2 instance.
- You can mount your Amazon EFS file systems on your on-premises datacenter servers when connected to your Amazon VPC with AWS Direct Connect.

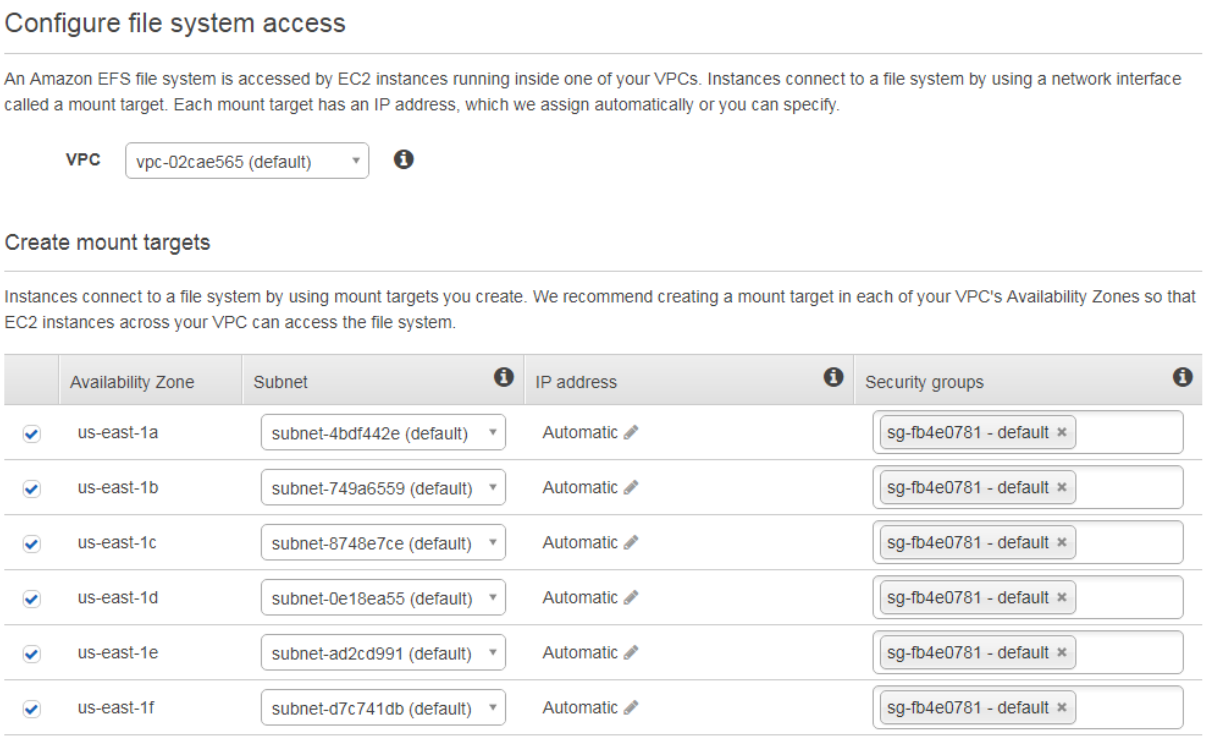
Steps to Create EFS:

1. We can find the EFS under storage category.
2. EFS is not available in all the regions as of now. Here is the supported regions. Switch to the region where you wish to create.

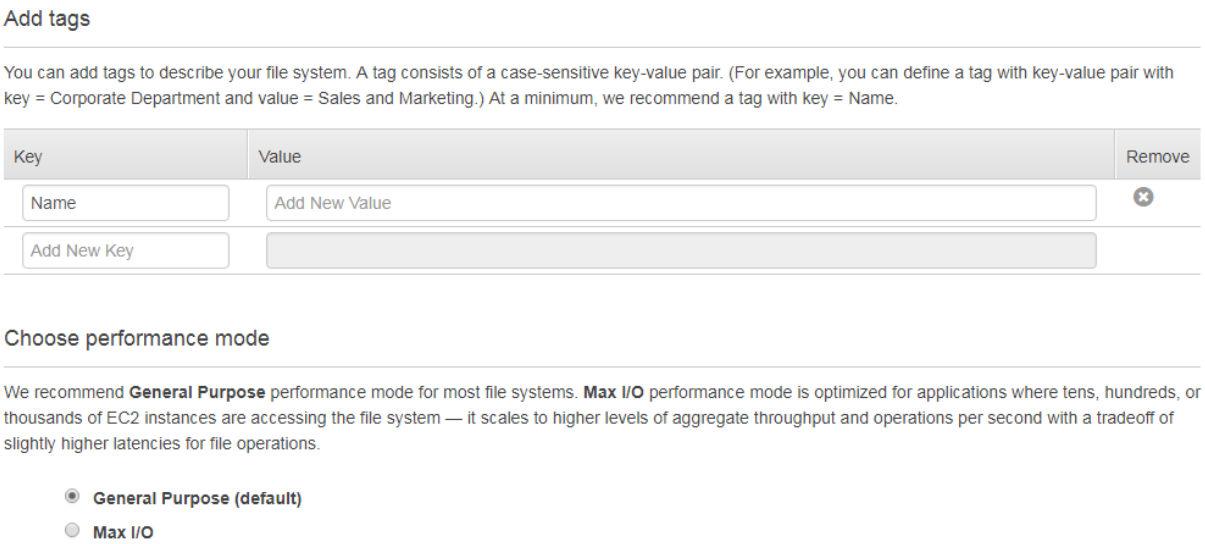
3. So, I switched to N. Virginia to perform the lab and selected EFS and select **Create file system** option.



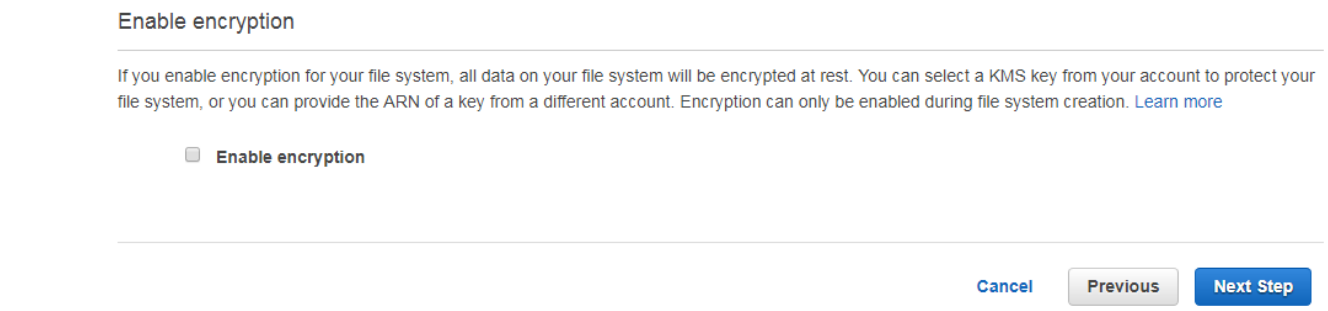
4. Select your VPC and Subnets, if you don't want to make this file system available to any specific subnet, Just untick that here. Then select **Next.**



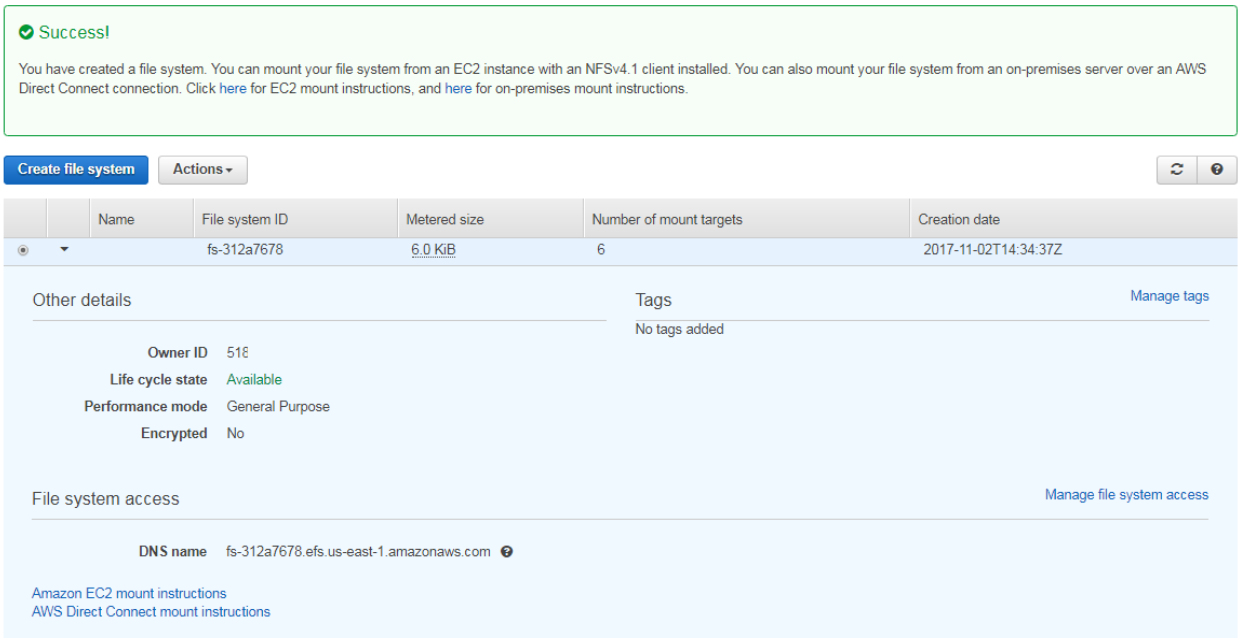
5. If we want to add tags, we can add here and we need to select the Performance Mode. We have to select this based on EC2 instance count.



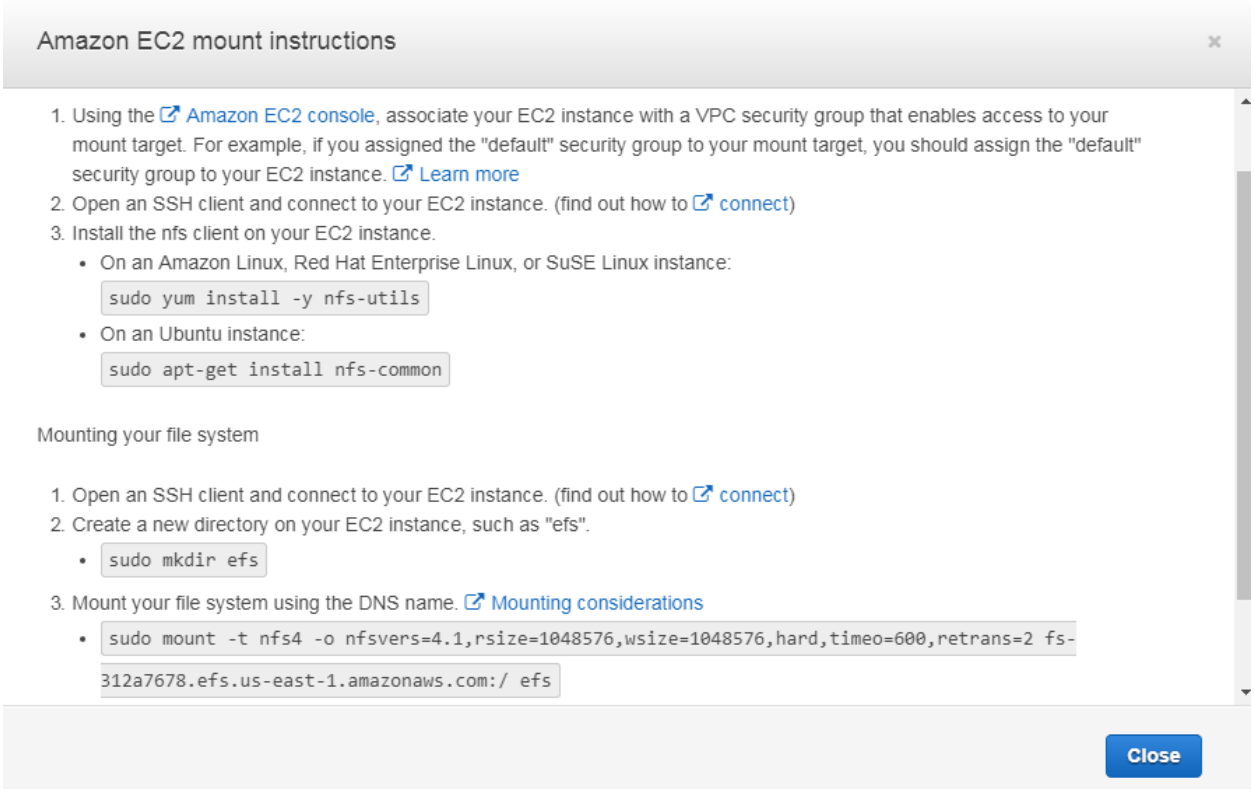
6. If we want to encrypt the data storing under EFS, we can enable the option on same page, then click on **NEXT.**



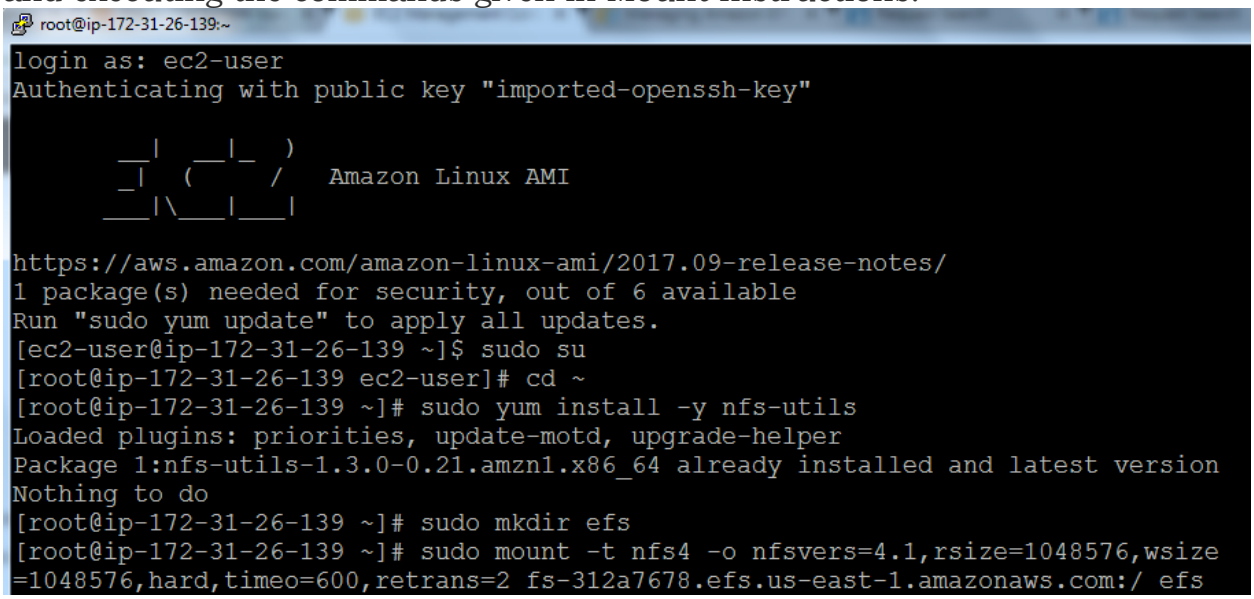
7. Review all the options and select Create File System option, file system will be created now and available for usage.



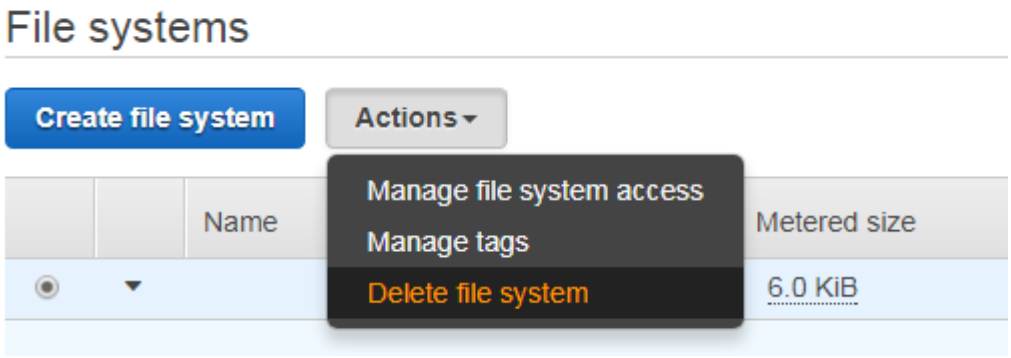
8. Now we have to mount it to EC2 instances, for mounting we need to login to Instance and need to follow mounting instructions. To get the Instructions select the **Amazon EC2 mount instructions** option.



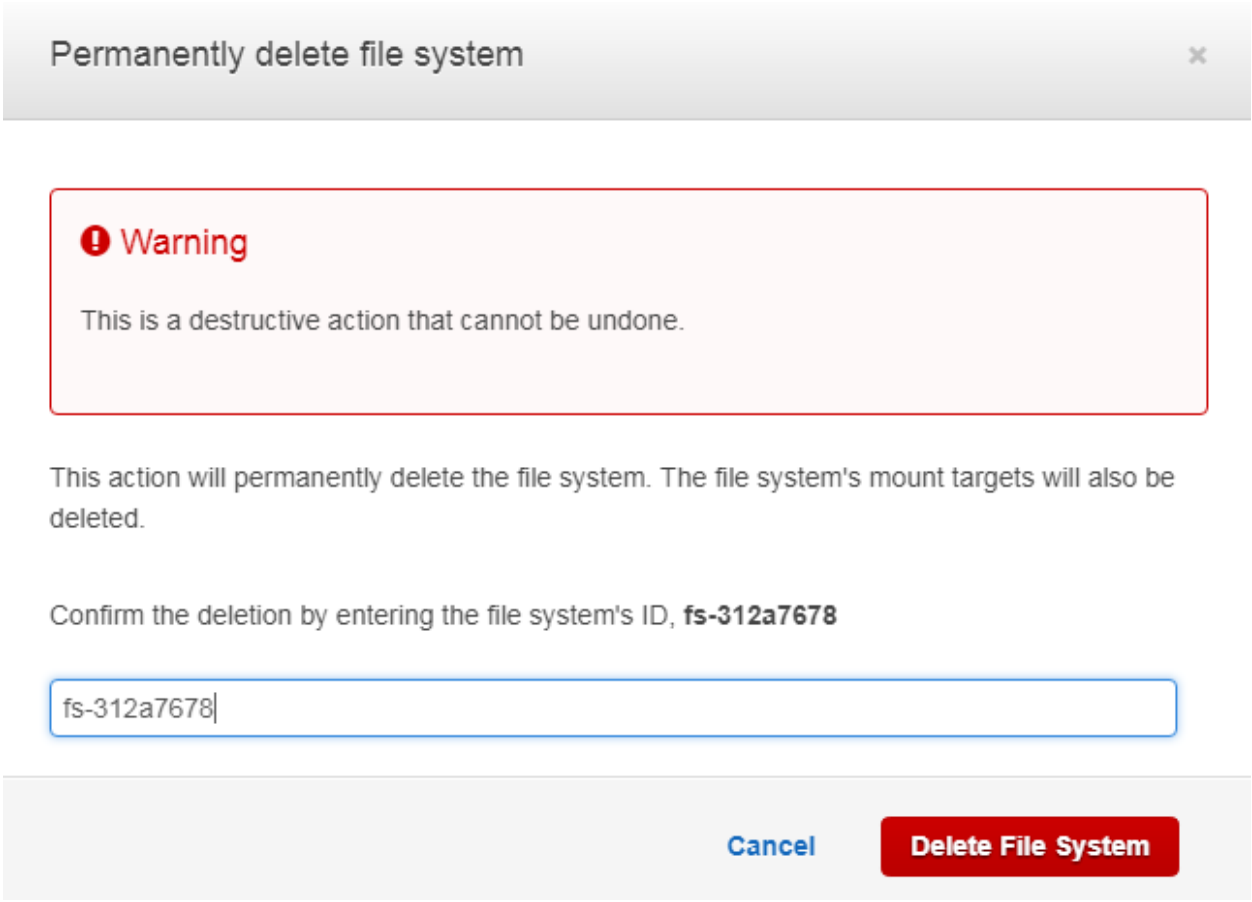
9. You can run the following commands on your EC2 instance.
10. **Your instance must be member of the Default Security group for successful EFS mounting.**
11. Here am launching Linux EC2 instance, as windows not supportable and executing the commands given in Mount Instructions.



12. In above image, I've elevated my privileges to root and tried to install the required **nfs-utils,** but It'll installed by default in Amazon Linux Instances.
    - Created a directory named efs with "sudo mkdir efs" command.
    - And executed the mounting command to the created directory, now whatever the files I created under "efs" is going to available for all EC2 instances.
    - If you want to test this, perform the same steps in another EC2 instance and test it.
13. If you want to delete the EFS, Select the EFS and go to "**Actions"** and **"Delete File System".**



14. Enter the file system's ID in the box and select the "Delete File System" button, File system will delete now.



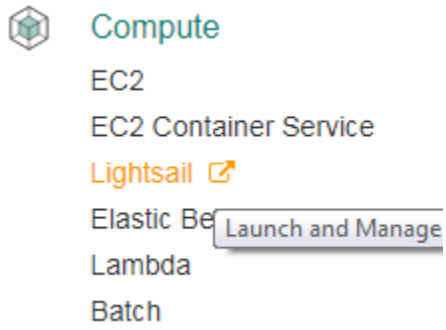
LIGHTSAIL

With Amazon Lightsail with a couple of clicks we can choose a configuration from a menu and launch a virtual machine preconfigured with SSD-based storage, DNS management, and a static IP address.

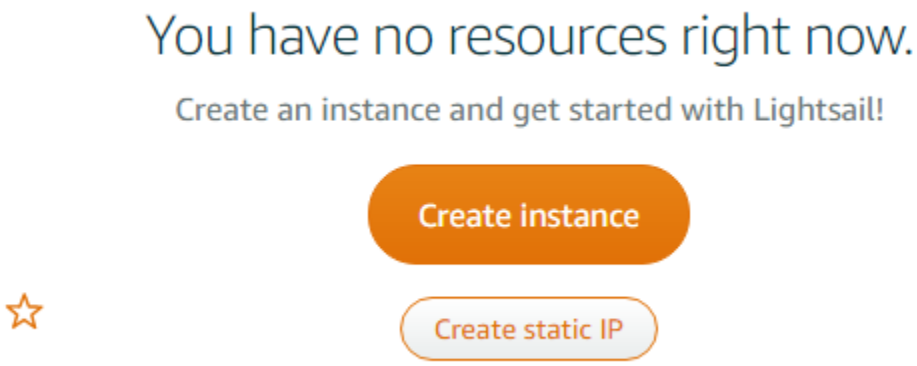
We can launch it on Amazon Linux AMI or Ubuntu operating system, developer stack (LAMP, LEMP, MEAN, or Node.js), or application (Drupal, Joomla, Redmine, GitLab, and many others), with flat-rate pricing plans that start at $5 per month including a generous allowance for data transfer.

Steps to launch Lightsail Instance

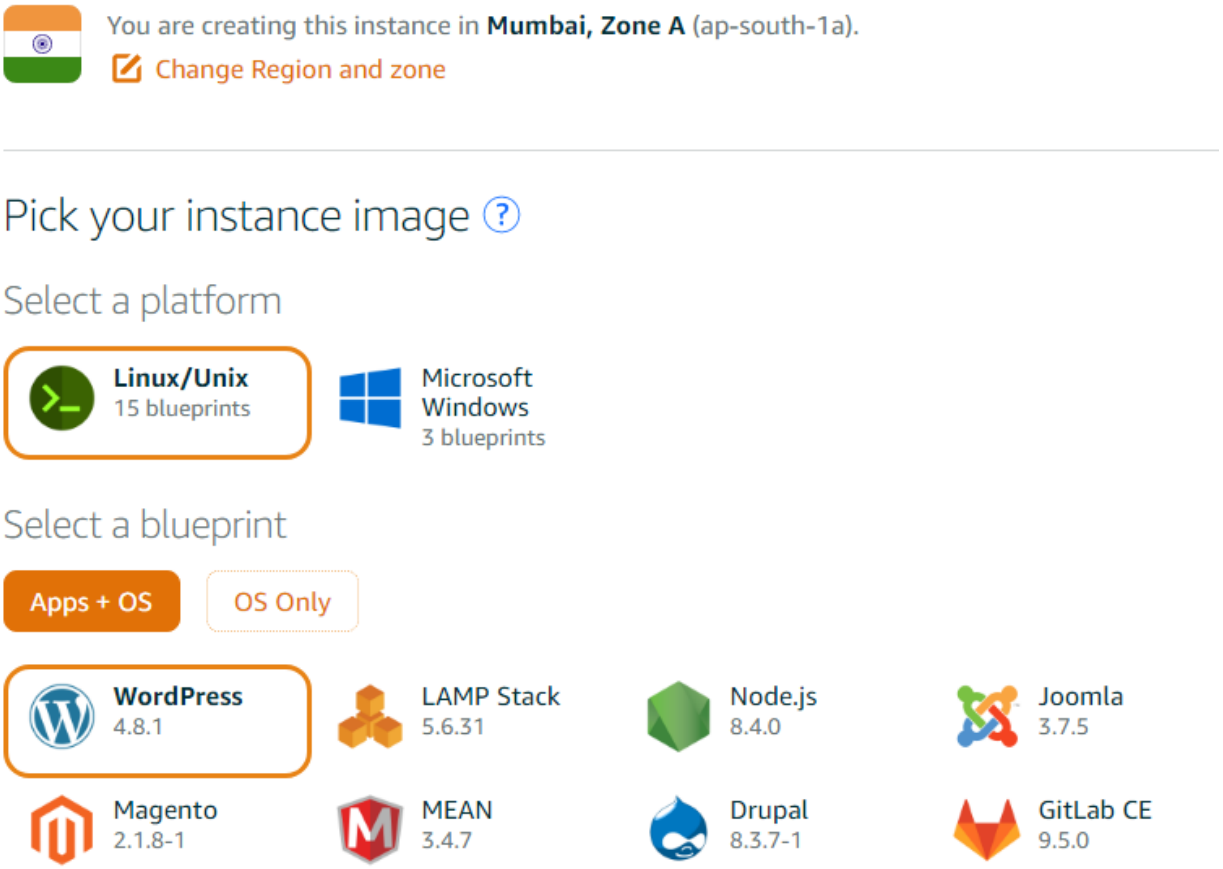
1. Select the **Lightsail** from Compute Service.



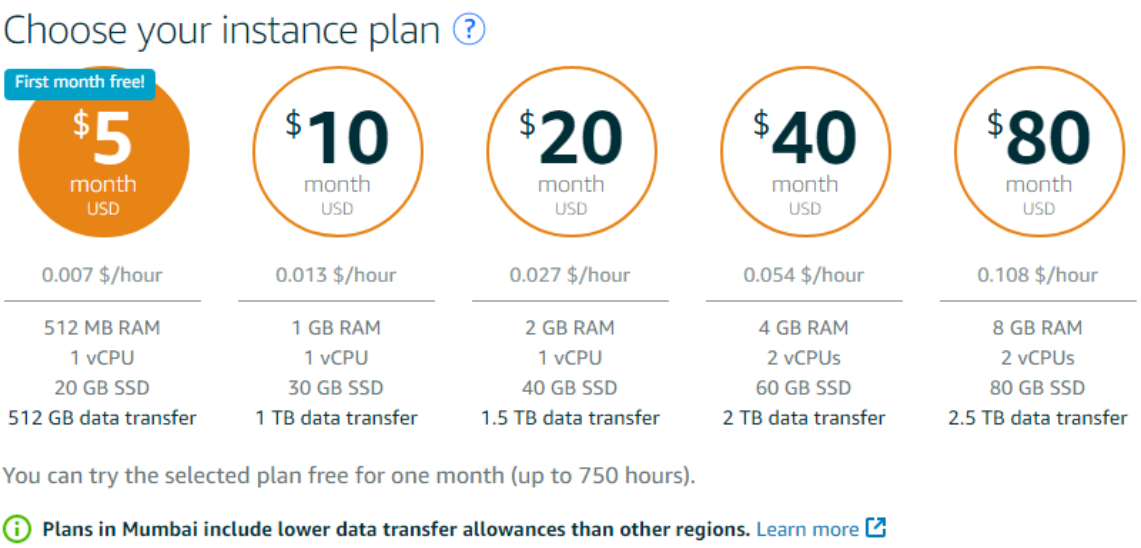
2. Select the **Create instance** option.



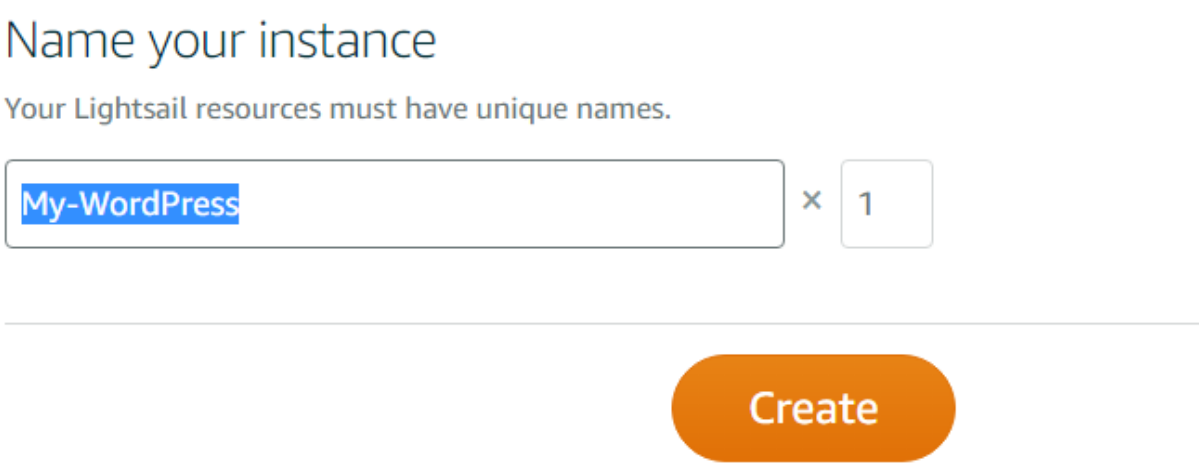
3. Select the Region and Zone, then select the Platform, and a blueprint what instance what application we required. Now am going to launch **Wordpress** website.



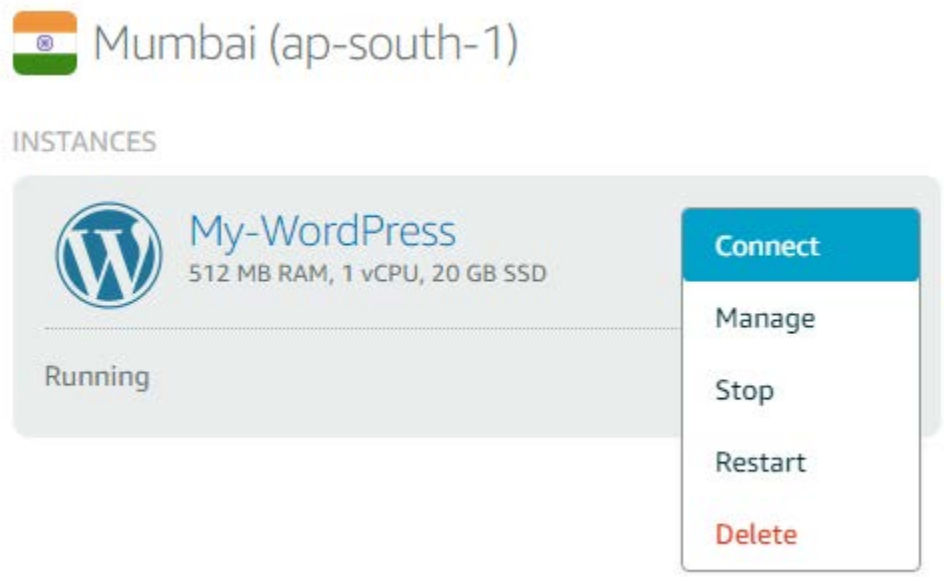
3. Then choose instance plan, am selecting $5/Month.



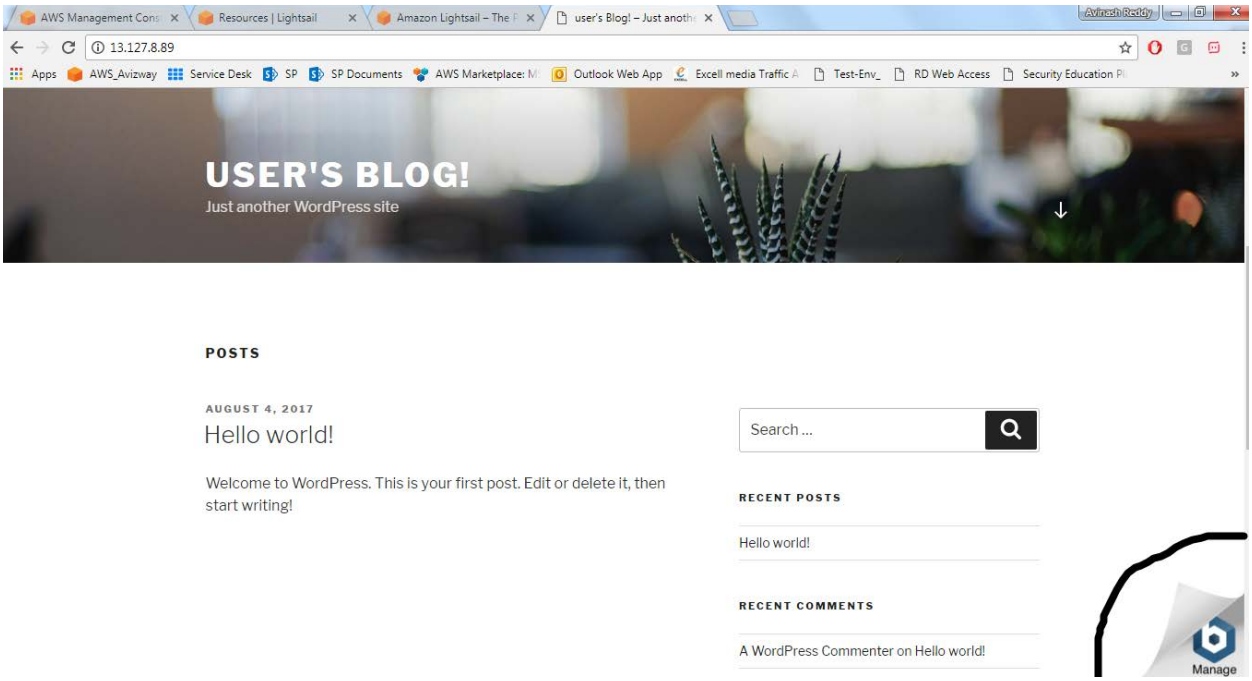
4. And give a name for your instance and select **Create** option.



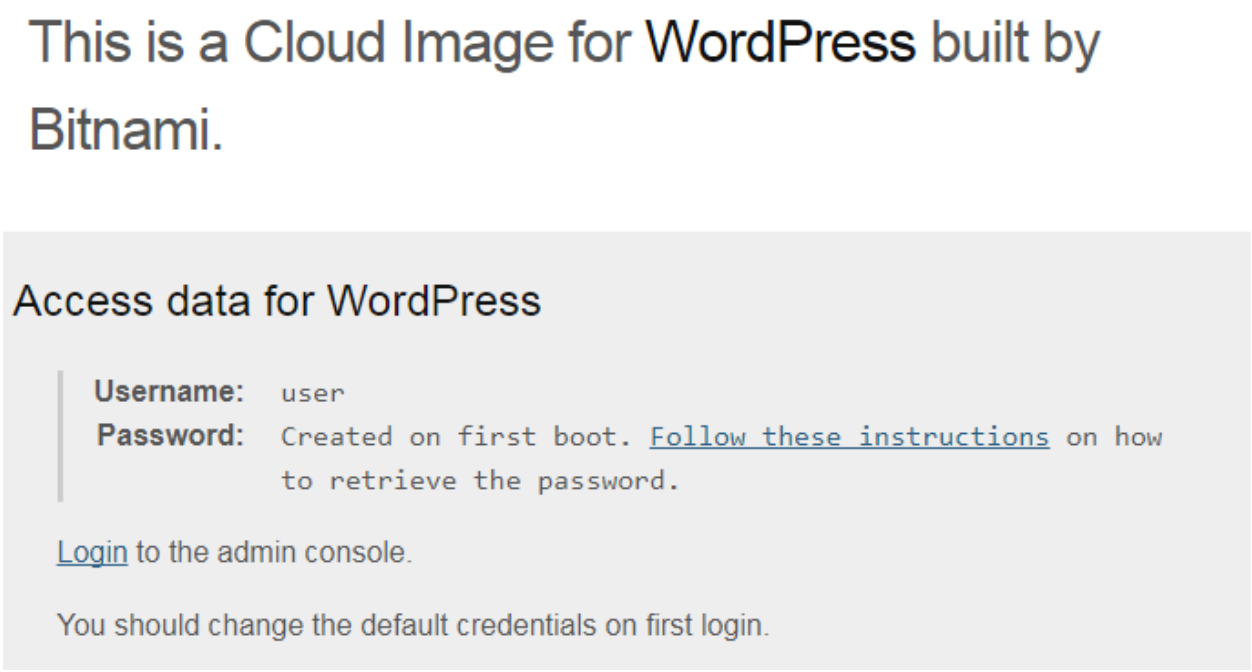
5. When the instance is ready select the connect option and you'll get a console.



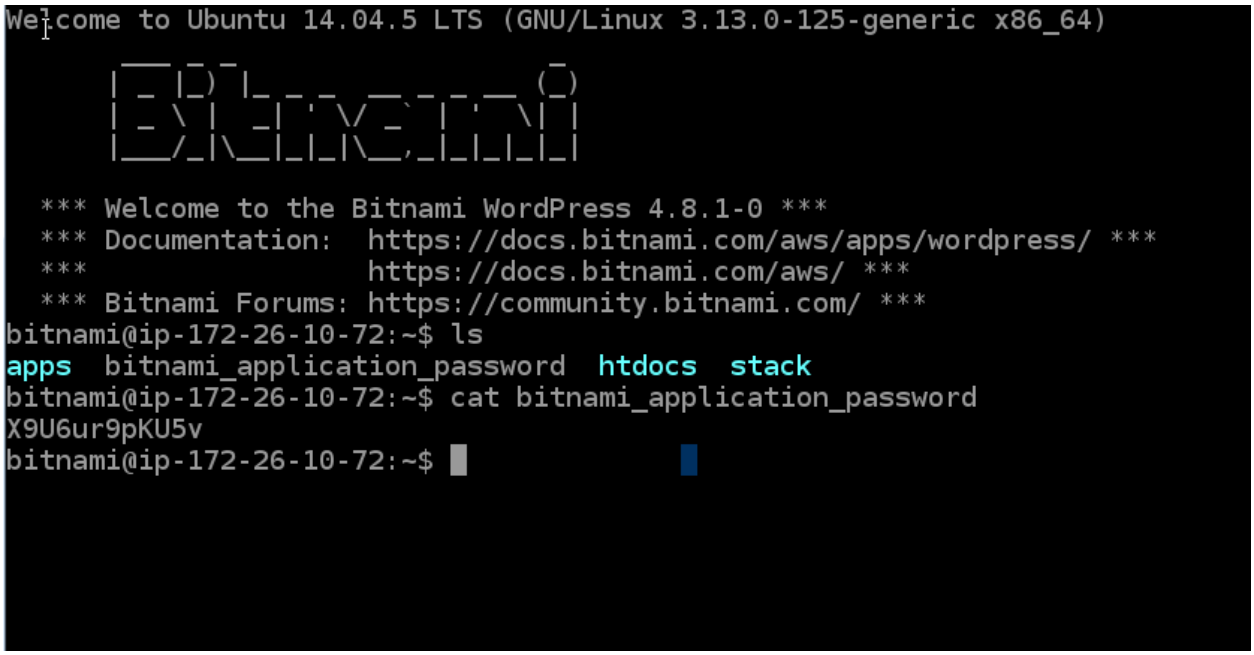
6. We'll get a public IP address by using that Public IP, we can access the WP website.
7. We will get a default template, if you want to customize that we have to login to the Admin panel. Here I've entered public IP the browser. In bottom corner, We will get Manage button, select that to login.



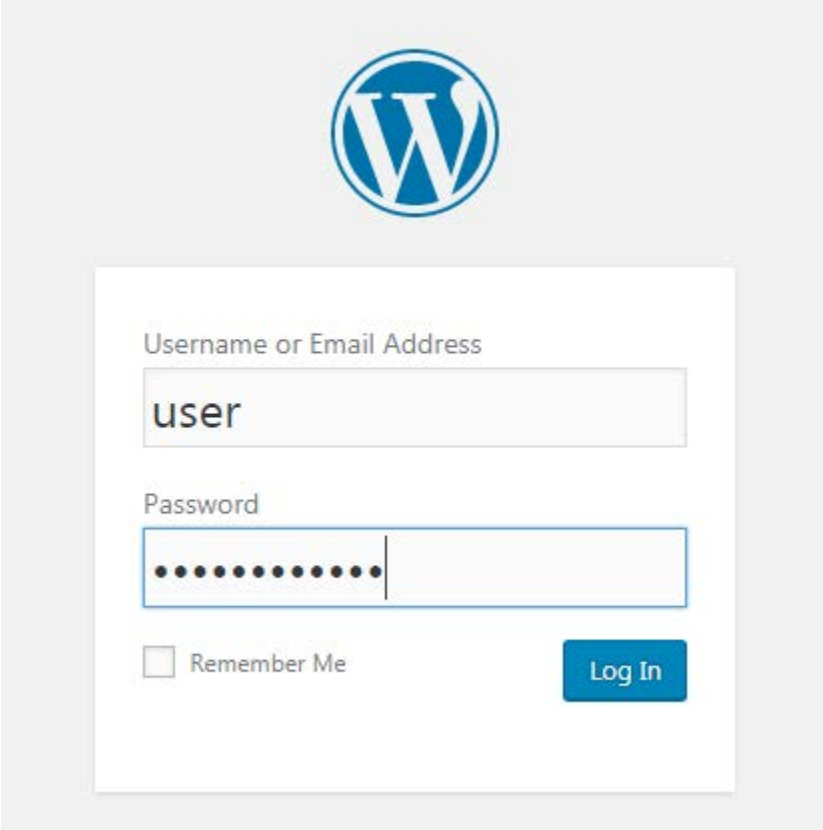
8. Default username is **user** and to get the password am connecting to the instance and entering command as below image. Select on **Login** option.



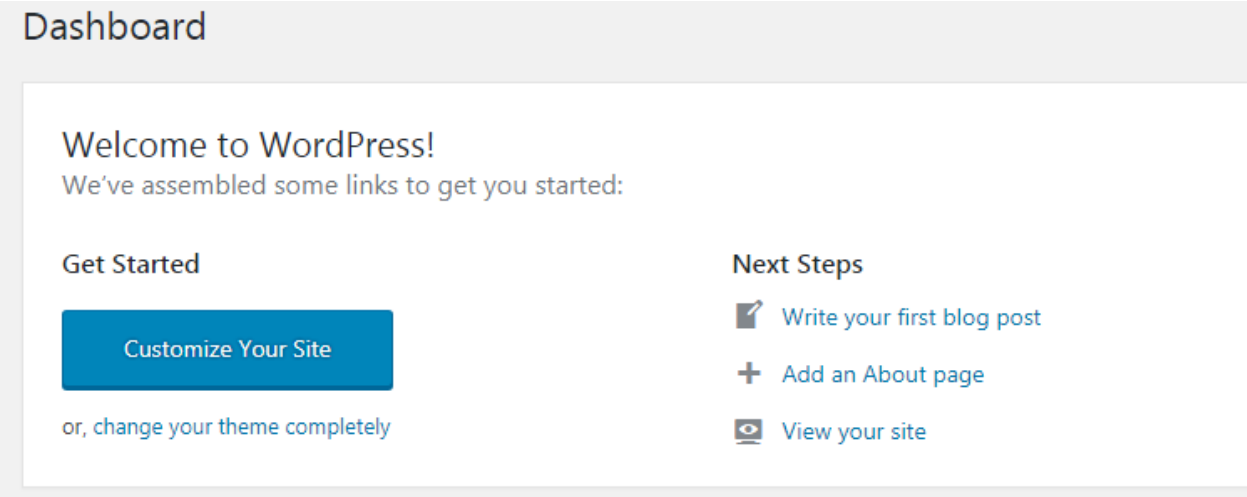
9. After connecting the instance give ls command you'll find bitname_application_password file, open it with cat command you'll get password to login, note it and enter in the login page.



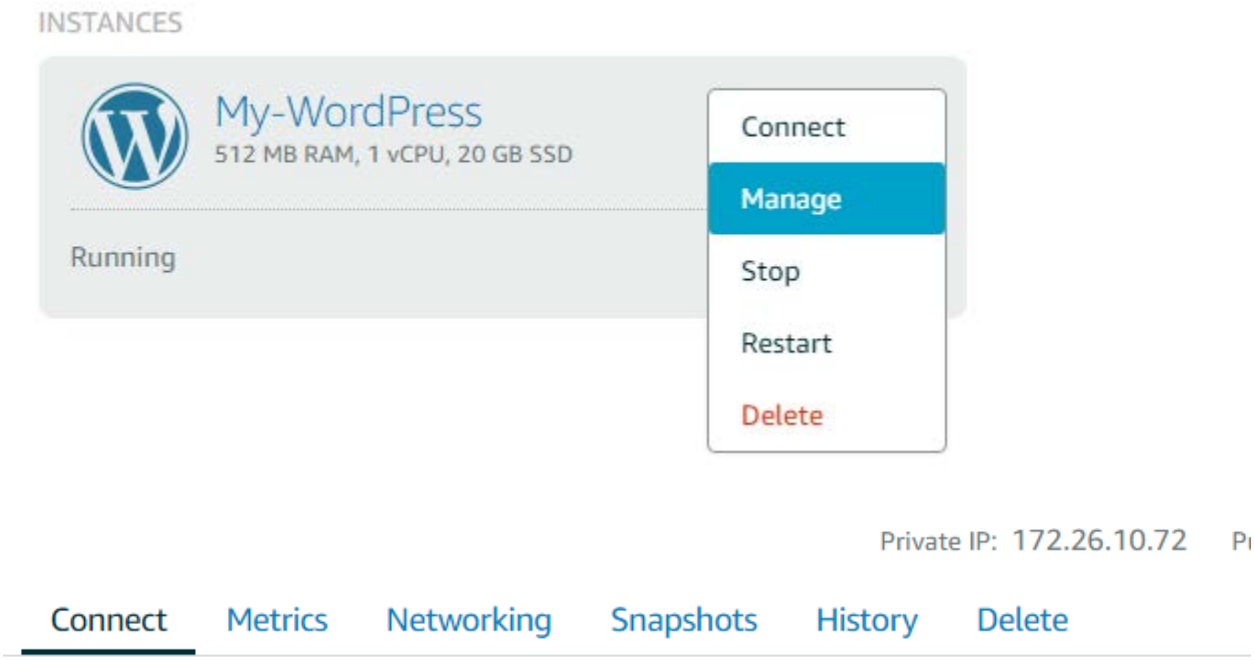
10. Give the username and password in the listed fields.



11. After authenticating, we'll login to the WP website and we can start customizing the website and select the Publish then the changes will update immediately.



12. If you want to manage your instance you can select the Manage option and you'll get the options to view the Metrics, Networking, Snapshots for backup, History and Delete options.



13. You can delete it anytime, by Delete option.

Elastic Beanstalk

With Elastic Beanstalk, we can deploy, monitor, and scale an application quickly and easily.

AWS Elastic Beanstalk is an orchestration service offered from Amazon Web Services for deploying infrastructure which orchestrates various AWS services, including EC2, S3, Simple Notification Service (SNS), CloudWatch, autoscaling, and Elastic Load Balancers.

AWS Elastic Beanstalk supports the following languages and development stacks:

- Apache Tomcat for Java applications

- Apache HTTP Server for PHP applications
- Apache HTTP Server for Python applications
- Nginx or Apache HTTP Server for Node.js applications
- Passenger or Puma for Ruby applications
- Microsoft IIS 7.5, 8.0, and 8.5 for .NET applications
- Java SE
- Docker
- Go

Application Deployment requires a number of components to be defined as follows

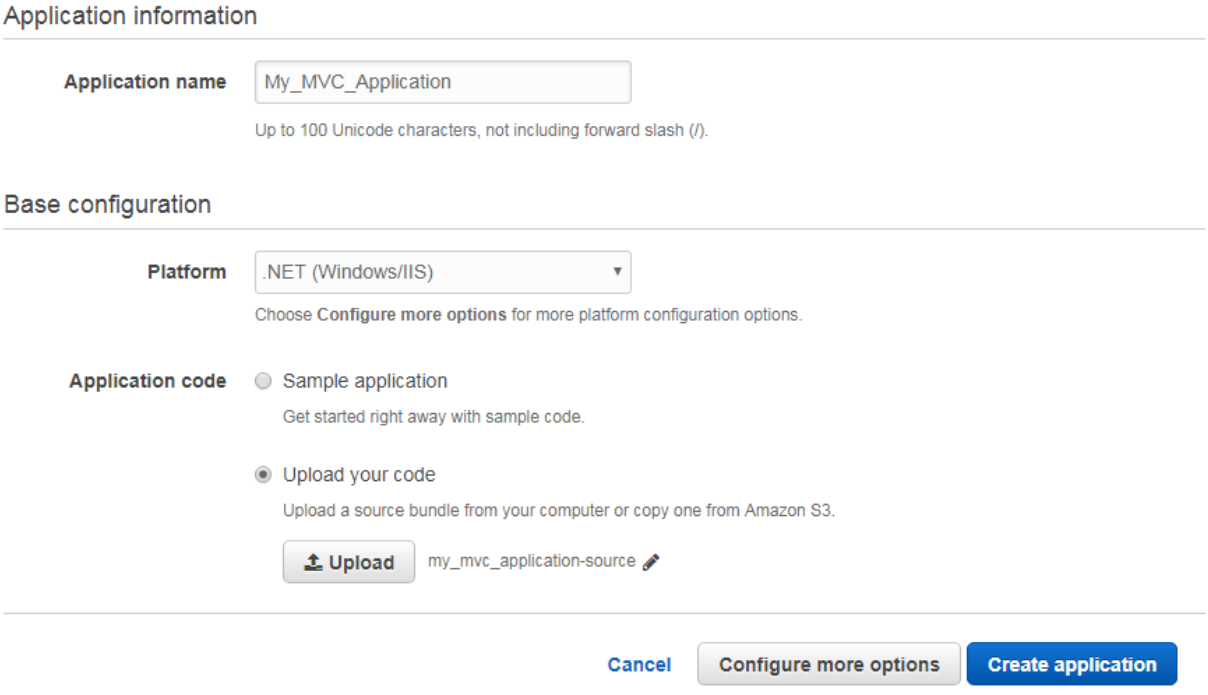
Application: as a logical container for the project.

Version: which is a deployable build of the application executable.

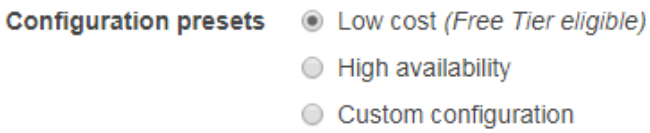
Configuration template: This contains configuration information for both the Beanstalk environment and for the product.

Environment: combines a 'version' with a 'configuration' and deploys them.

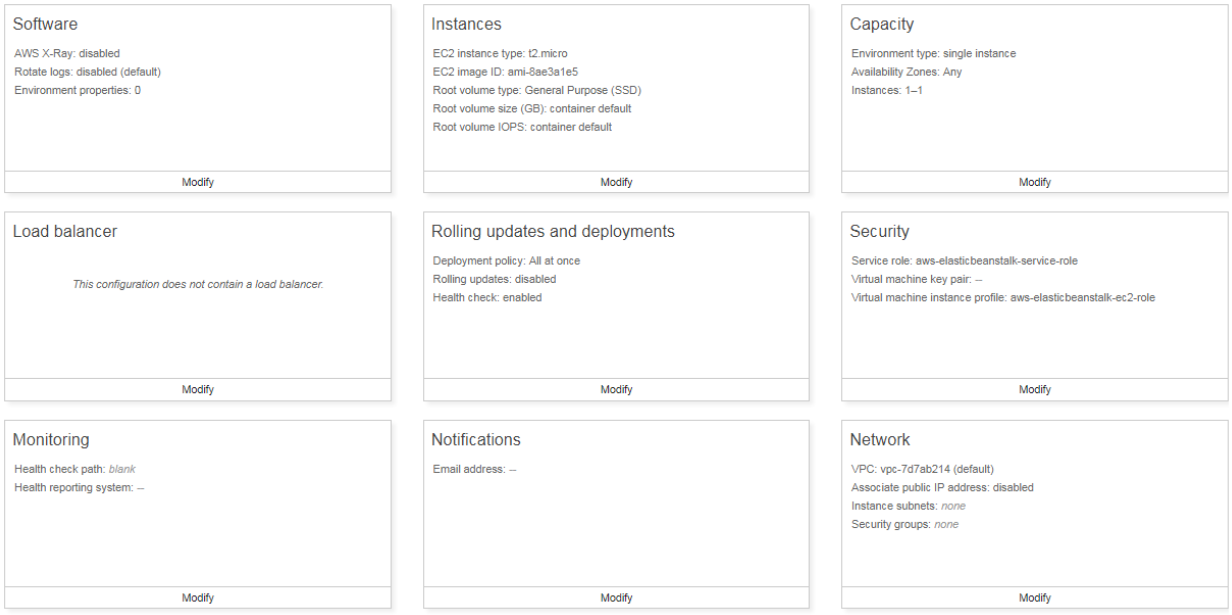
1. Create a Web Application. It involves with multiple options. By creating an environment, we allow AWS Elastic Beanstalk to manage AWS resources and permissions on behalf of us.



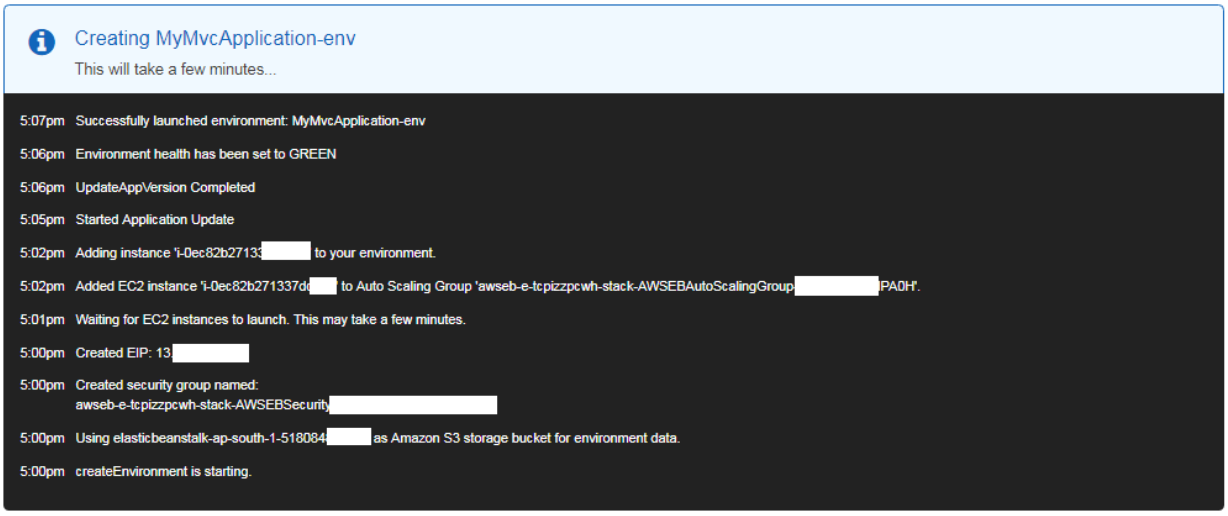
2. You can simply select the Create application option to perform the deployment and selecting the appropriate configuration for our instances.
3. If you want to customize each and every step, as you required, Select **Configure more options** option.
    - Then we'll get three options for **Configuration presets**
        i. **Low Cost (Free Tier eligible)**
        ii. **High Availability**
        iii. **Custom Configuration**



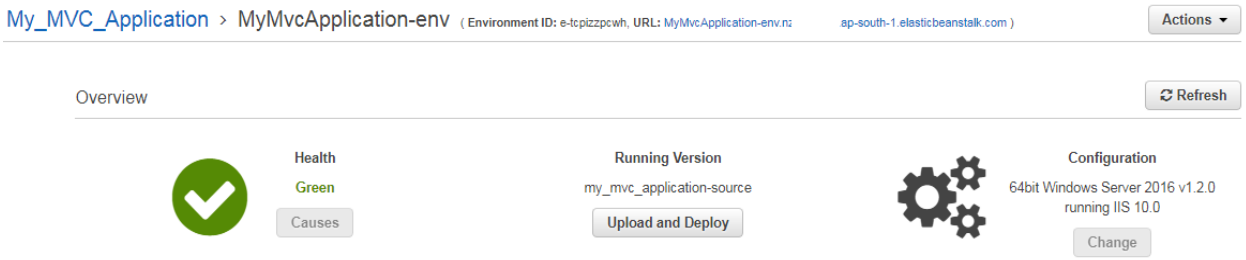
4. If we want to change the Platform of Windows server or IIS, we can select change platform configuration option otherwise go with the default option.
5. Select the appropriate option, here am selecting the Low Cost, Free Tier eligible.
6. Here is the available options to customize



7. Status of Instance creation, and all the required resources are provisioning by Elastic BS i.e; Security group, EIP, EC2, S3, Simple Notification Service (SNS), CloudWatch, autoscaling, and Elastic Load Balancers.

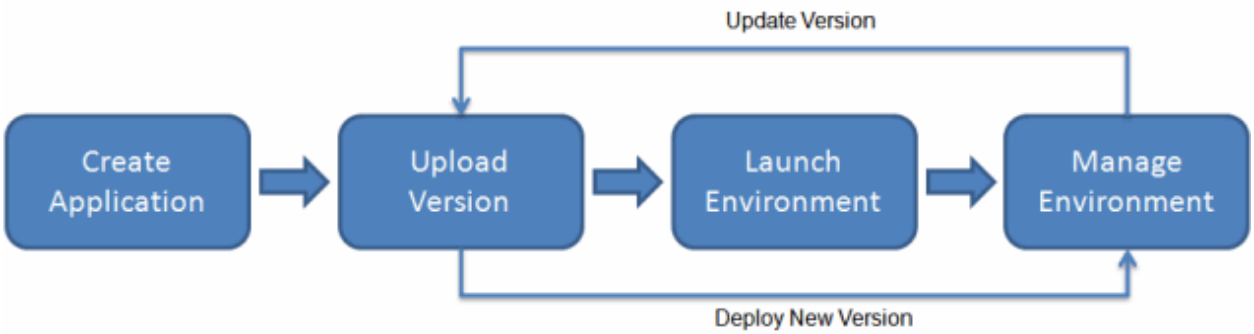


8. Here is the status we'll get when the application is deployed.

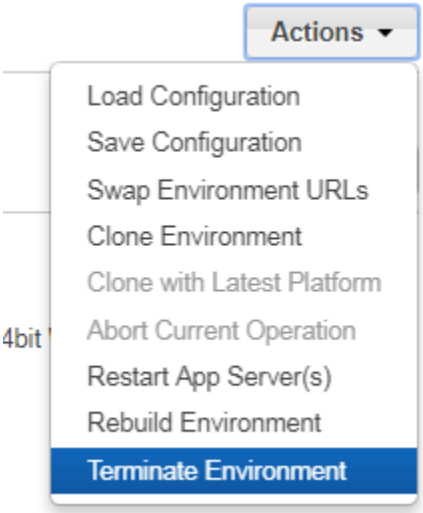


9. We'll get Environment ID to access the application.
10. Here is the output for my uploaded code.

11. If you made any changes to your existing code, you can zip it and upload it.
12. Here is the illustration diagram of workflow



13. If you want to terminate the environment, select the **Actions** option in Top right corner, then choose Terminate Environment.



14. Or go back to the applications page and delete the application.