**Amazon EC2:**

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.

Amazon EC2’s simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon’s proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use. Amazon EC2 provides developers the tools to build failure resilient applications and isolate themselves from common failure scenarios.

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

**Features of Amazon EC2:**

Amazon EC2 provides the following features:

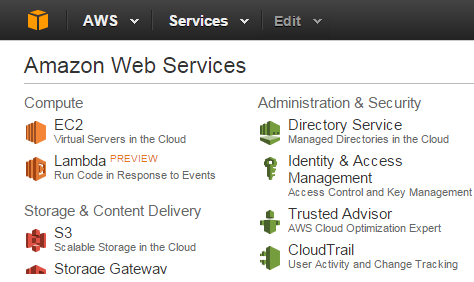
* Virtual computing environments, known as instances.
* Preconfigured templates for your instances, known as Amazon Machine Images (AMIs), that package the bits you need for your server (including the operating system and additional software).
* Various configurations of CPU, memory, storage, and networking capacity for your instances, known as instance types.
* Secure login information for your instances using key pairs (AWS stores the public key, and you store the private key in a secure place).
* Storage volumes for temporary data that's deleted when you stop or terminate your instance, known as instance store volumes.
* Persistent storage volumes for your data using Amazon Elastic Block Store (Amazon EBS), known as Amazon EBS volumes.
* Multiple physical locations for your resources, such as instances and Amazon EBS volumes, known as regions and Availability Zones.
* A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using security groups.
* Static IP addresses for dynamic cloud computing, known as Elastic IP addresses.
* Metadata, known as tags that you can create and assign to your Amazon EC2 resources.
* Virtual networks you can create that are logically isolated from the rest of the AWS cloud, and that you can optionally connect to your own network, known as virtual private clouds (VPCs)

<http://aws.amazon.com/ec2/>

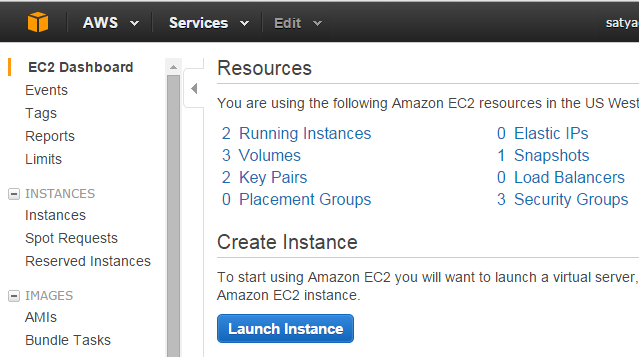
<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>

**How to launch an instance using Amazon Console:**

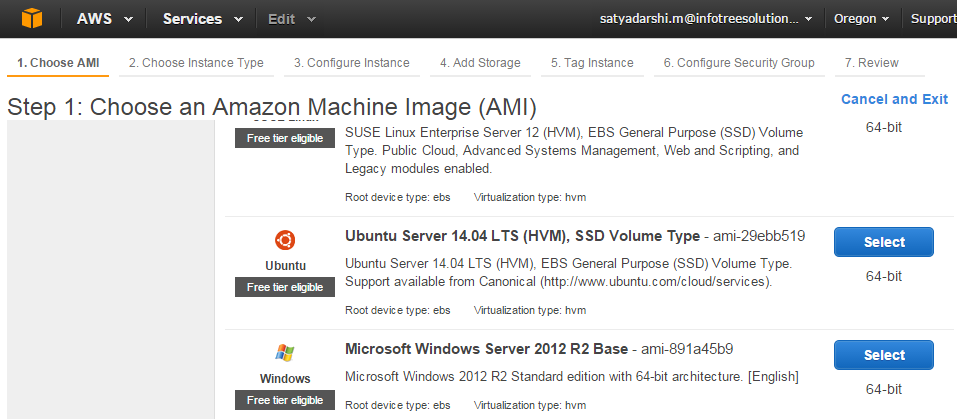
Login to amazon console and choose EC2.



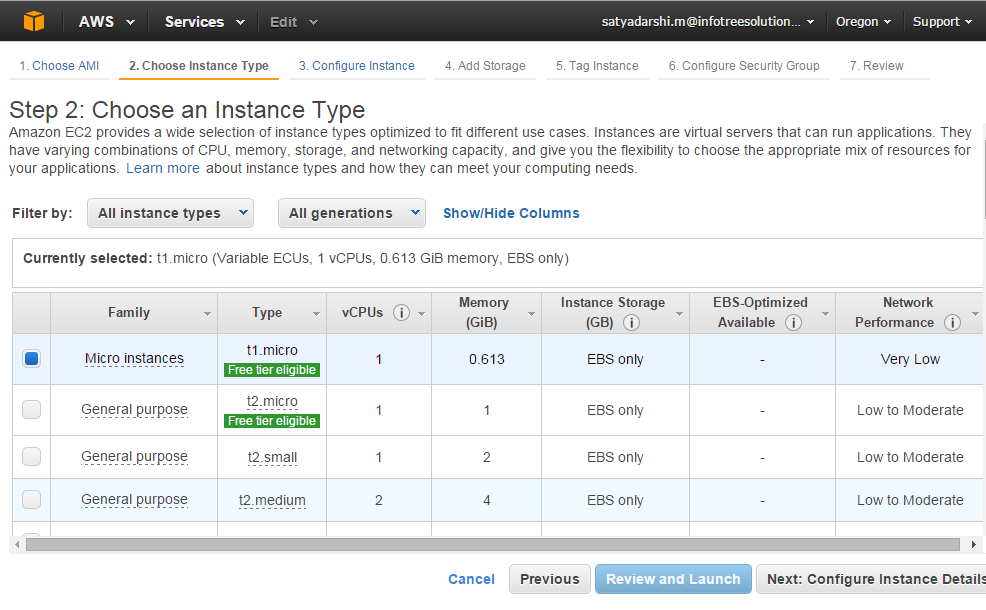
Click on Launch Instance submit button.



A list of AMI (Amazon Machine Image) will come.

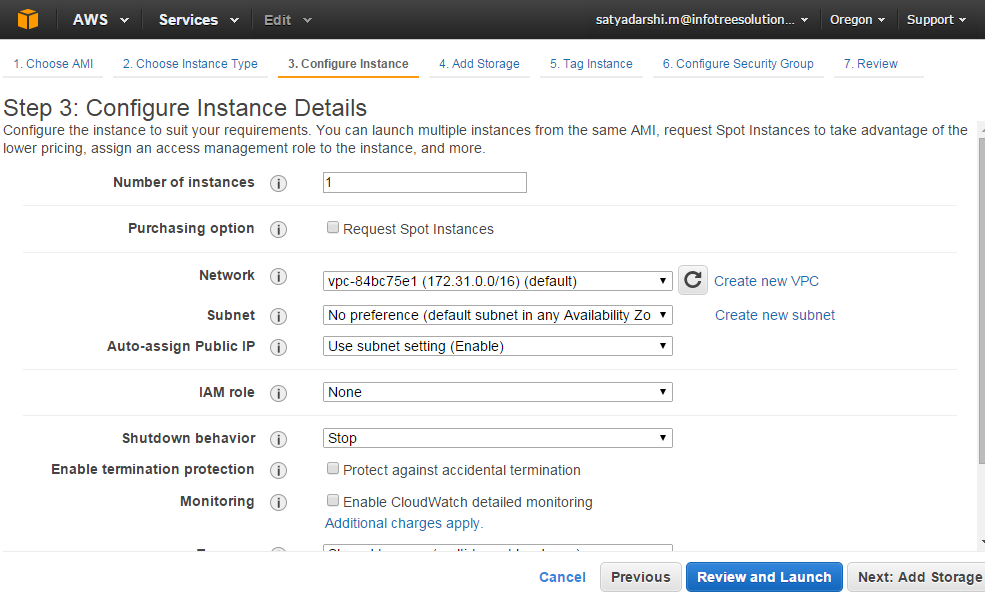


We have to choose one AMI depending upon our requirement.

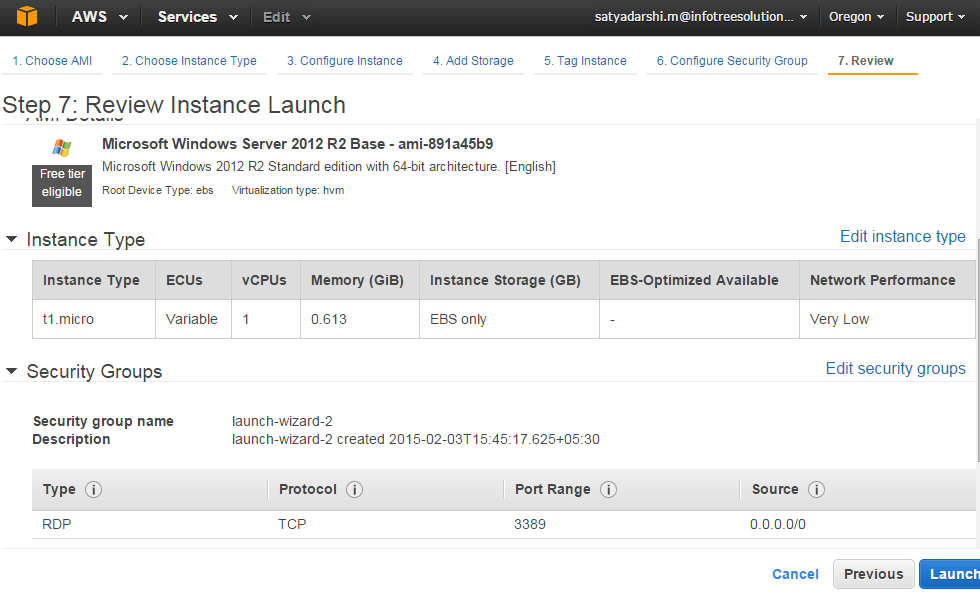


Then we have to choose our instance type. I selected t1.micro as my instance type and click on “Configure instance Details”.

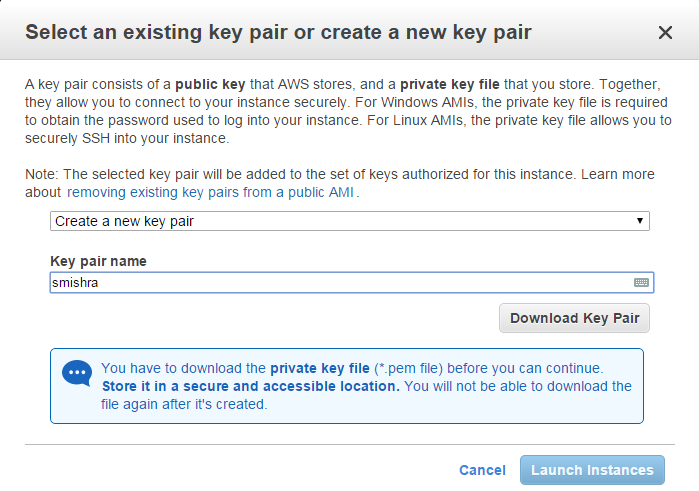
http://aws.amazon.com/ec2/instance-types/



Here we have to provide our instance details and then click on “Review and Launch”.



Here we can edit the instance type and also can edit the security group then click on “Launch”.



In the Select an existing key pair or create a new key pair dialog box, we can select Choose an existing key pair, to select a key pair we have already created.

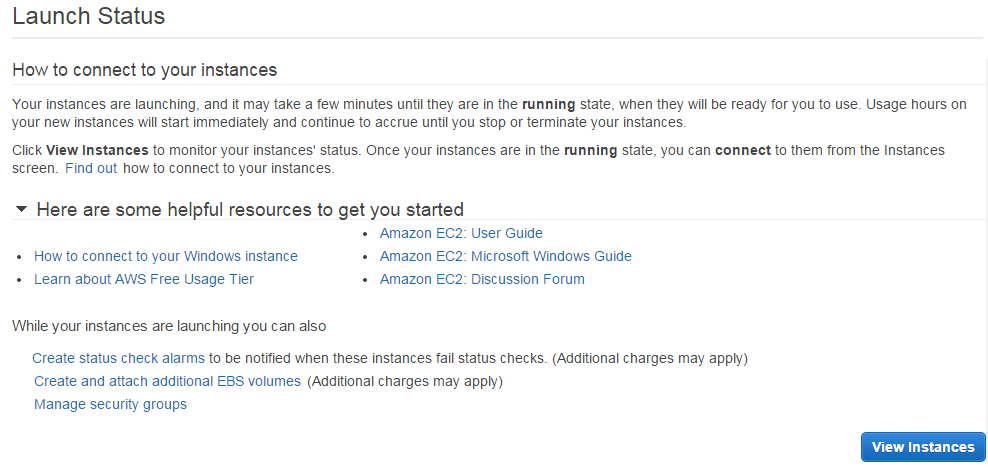
Alternatively, we can create a new key pair. For that we have to Select **Create a new key pair** option then, we have to enter a name for the key pair, and then click **Download Key Pair**. Save the private key file in a safe place. We need to provide the name of our key pair when we want to launch this instance and the corresponding private key we have to provide each time when we want to connect to that instance.

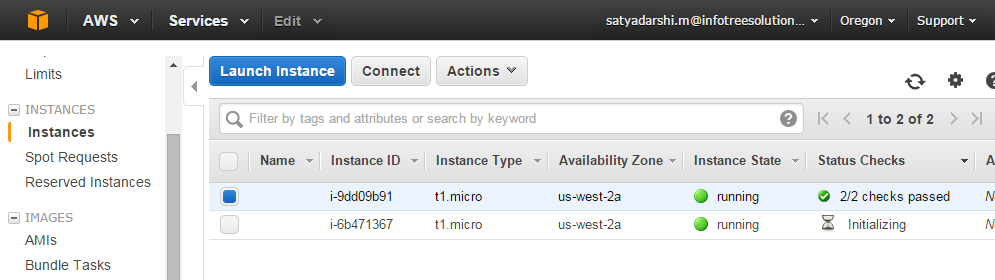
**Note:**

We never select the **Proceed without a key pair** option. If we launch our instance without a key pair, then we can't connect to it.

Then we need to click **Launch Instances.**

A confirmation page **Launch Status** page will come then we confirm that our instance is launching. Click **View Instances** to close the confirmation page and return to the console.





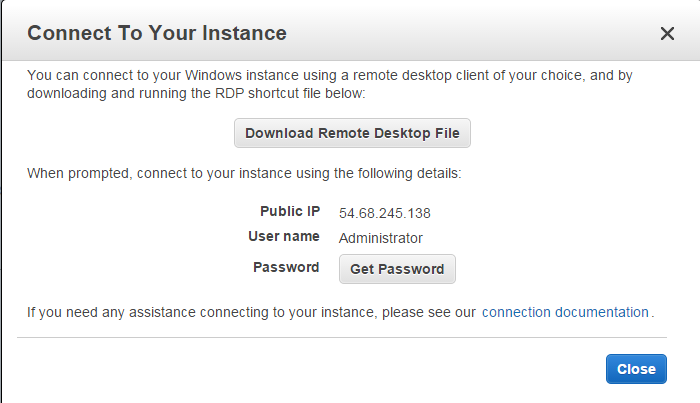
Now in the list our newly created instance will be present with

1. On the **Instances** page, we can view the status of the launch. It takes a short time for an instance to launch. When we launch an instance, its initial instance state is **pending**. After the instance starts, its initial state changes to **running**.
2. In the Status Checks initially it is in **initializing** state after 3-4 minutes its status changes to **2/2 Checks Pass**. Then we can able to connect to this instance.

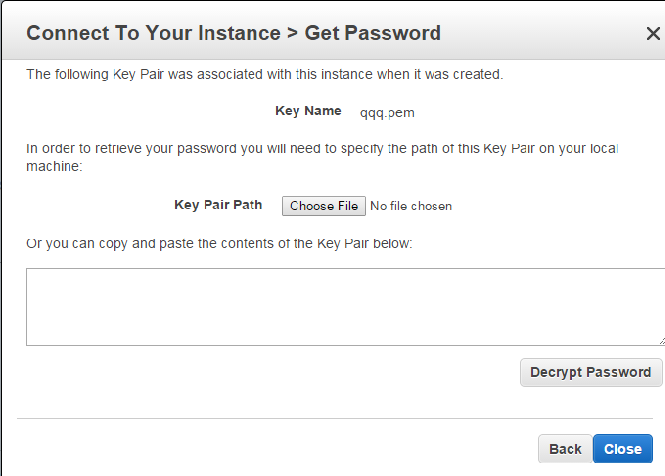
**How to connect to that instance using Remote Desktop Protocol:**

Right click on running instance and select on connect.

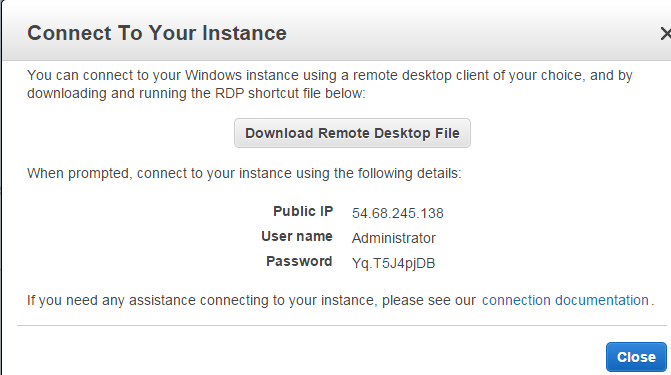
Then a dialogue box will come like below:



Click on **Get Password** button**.**

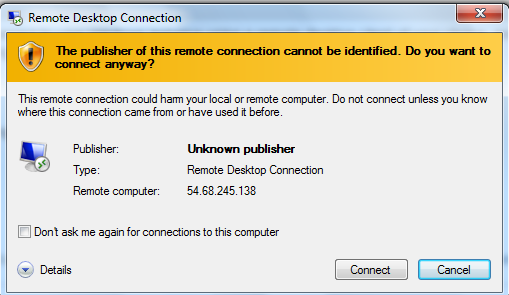


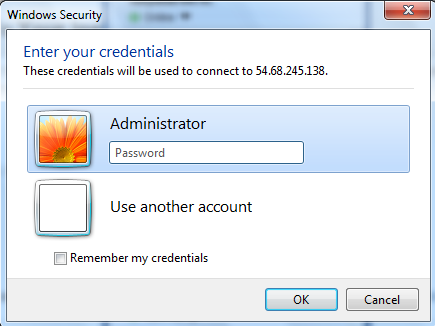
Here we have to choose our key pair file (.pem file) and then click on **Decrypt Password** .



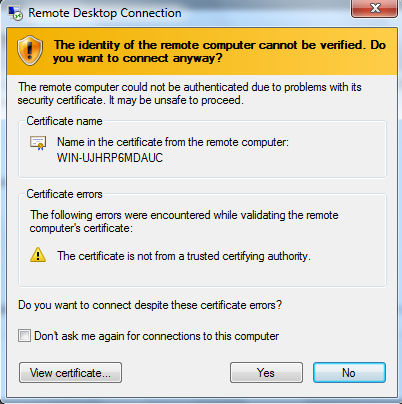
Click on **Download Remote Desktop File** then a .rdp file will automatically downloaded to our machine.

Then click on that file and click on **connect**.

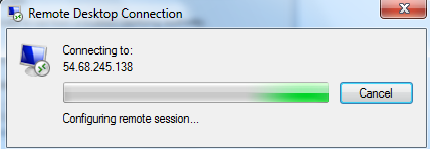




Here we have to provide our decrypted password (in this case Yq.T5J4pjDB) and click on OK.



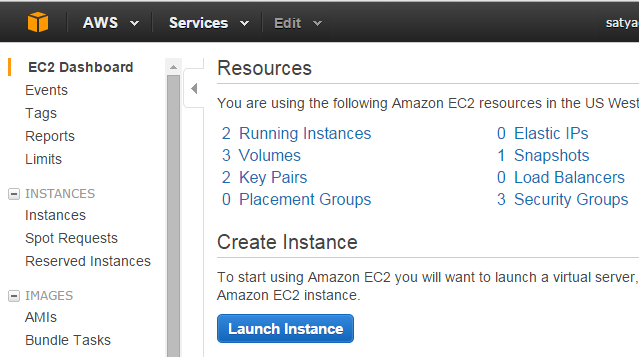
Click on **yes**.



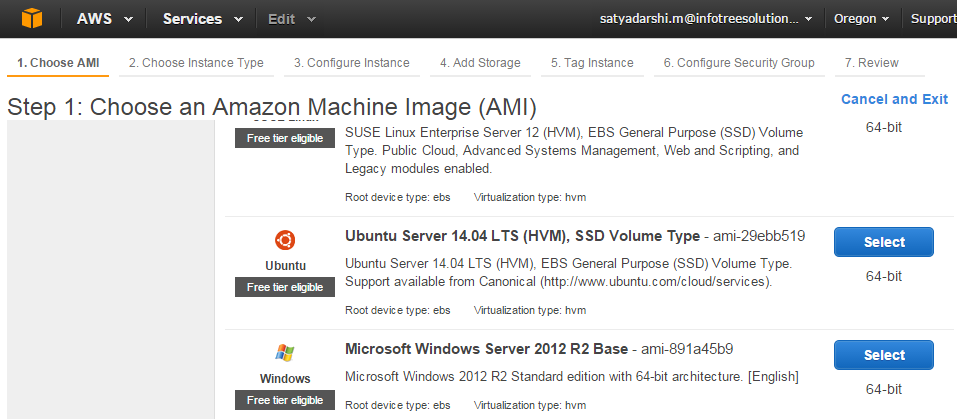
It will take some second then the virtual machine will come.

Use Case-1

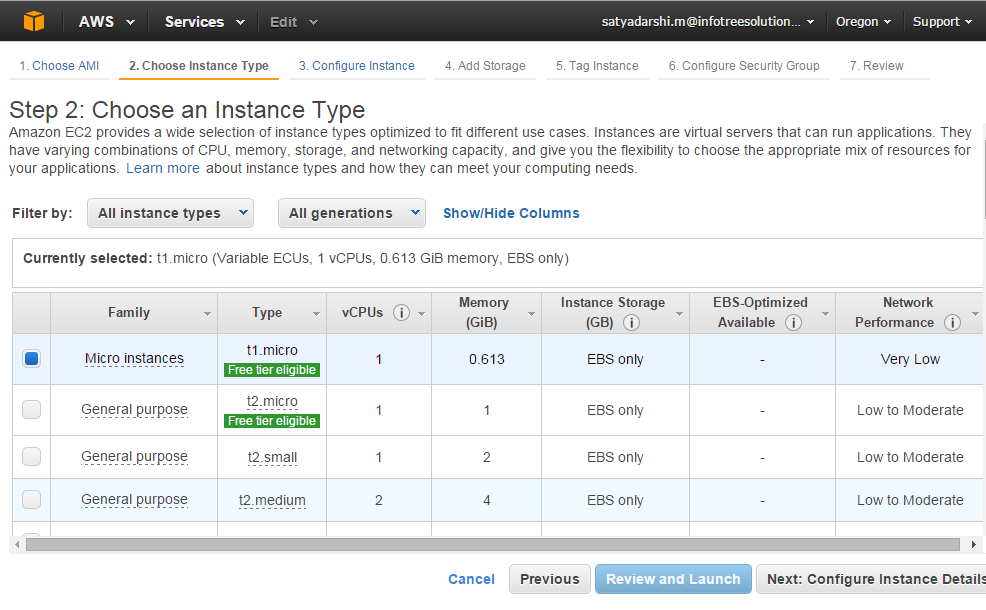
Here we want to create an instance having our own volume. We want to create an instance having two partitions. ‘C’ partition having 50 GB memory and ‘D’ partition having 40 GB memory. We have to follow the below steps for doing this:



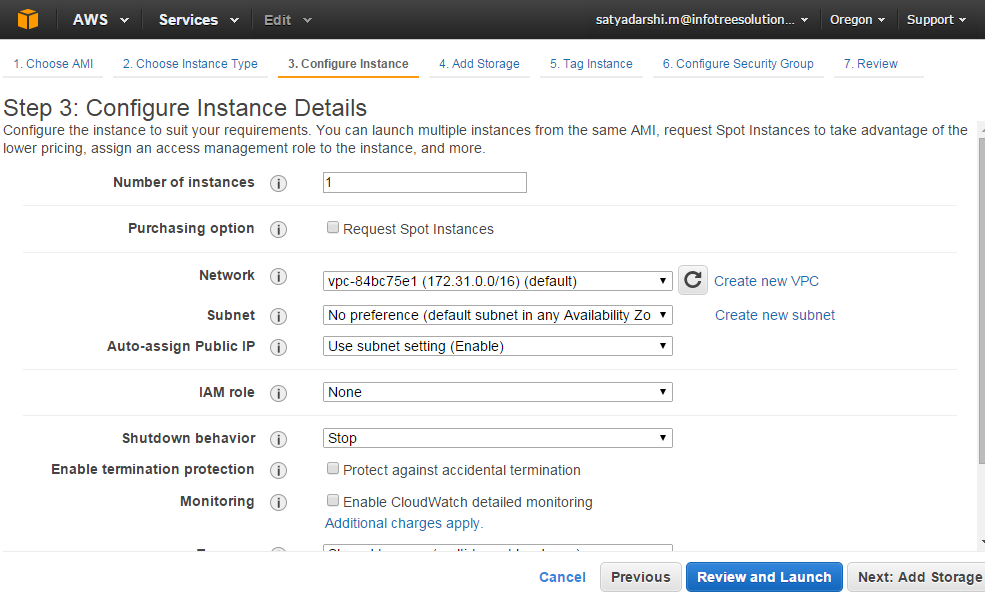
Click on Launch Instance.



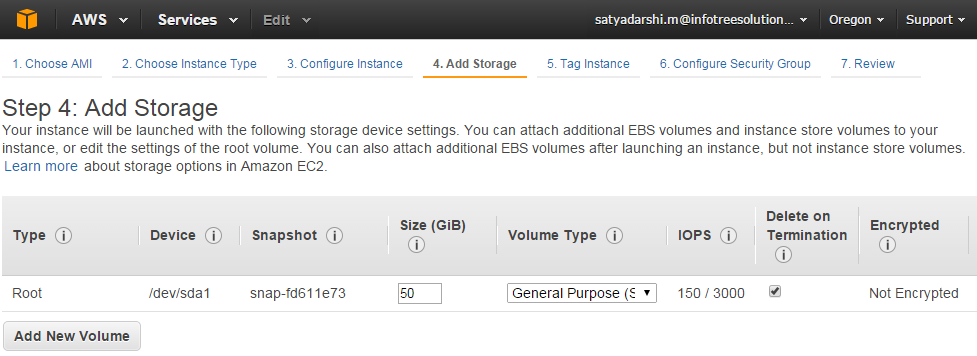
Here we have to choose our required AMI.



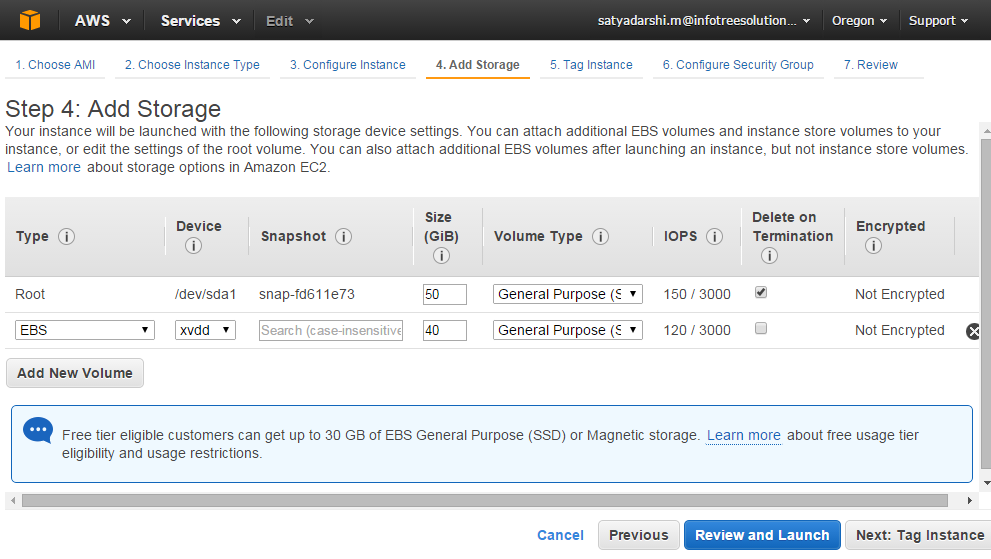
Here we have to choose our required Instance Type.



Here click on **Add Storage** button.



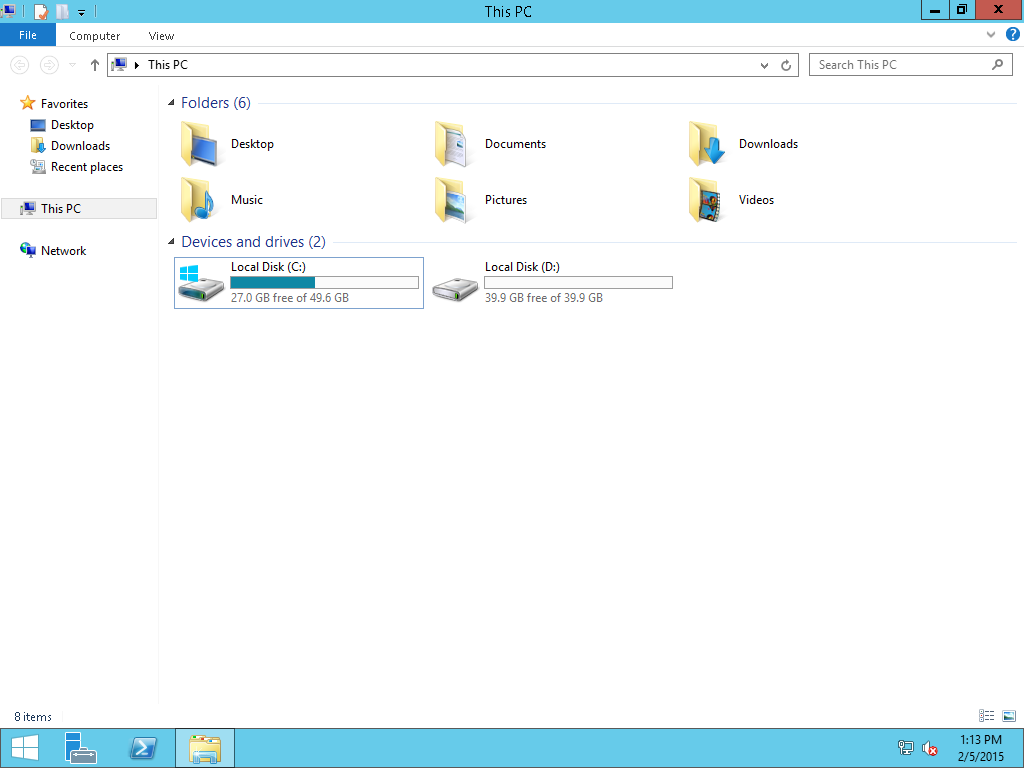
Here click on **Add New Volume**.



Here provide 50GB for first and 40GB for second partition and click on **Review and Launch.**

Then follow the same steps like earlier.

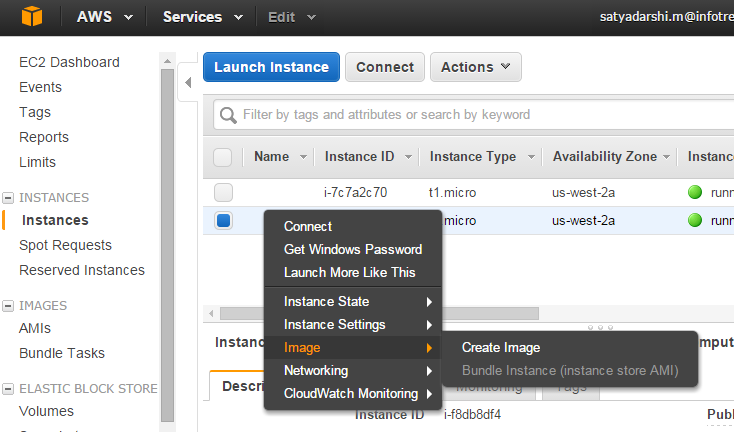
If we connect to that instance we will get like below:

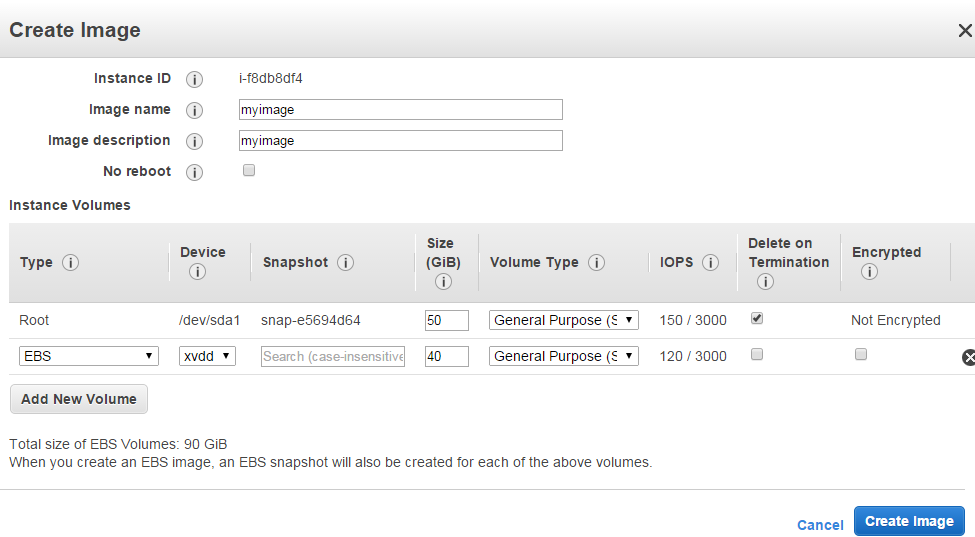


**Creation of Image:**

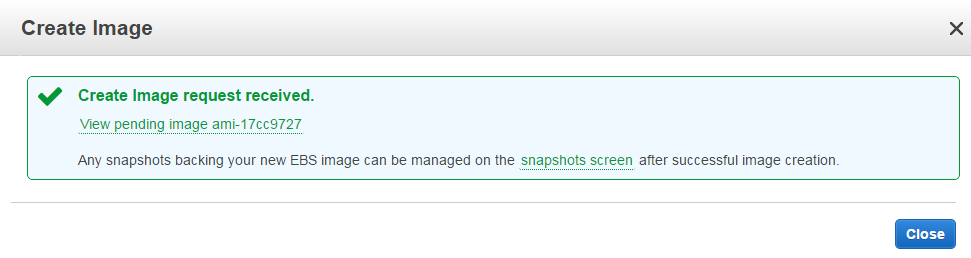
If from an existing instance we want to create multiple instances then we can create the image of the existing instance and from that image we can create our required instances.

Right click on the existing instance and click on **image** then choose **create Image.**



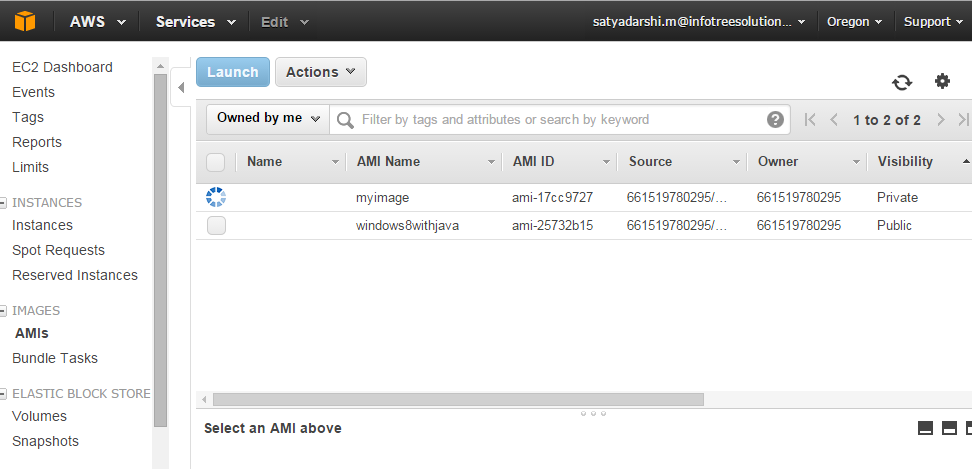


Here provide the image name and image description and click on **create Image** button.



Click on **Close**.

Creating Image means it will automatically create one AMI and snapshots and number of snapshots always same with number of volumes we provided in our instances.



When we launch a new instance it is internally create a volume and that volume is always associated with this instance. By default the volume size always depends on instance type. Initially the instance is in running state. Each Instance have four states start, stop, terminate, reboot.

If we want to remove the associate volume manually then it is not possible to remove because it is used by the instance. If we stop the instance and again want to remove the volume then now it is possible to remove the volume but after removal if we want to start the instance then the instance will be not start because it is not associated with the volume. If we want we can create the volume and again attach that volume with the instance then we can start that instance.

Creating instance programmatically:

AWSCredentials credentials=null;

try {

credentials=new PropertiesCredentials(InstanceCreation.class.getClassLoader().getResourceAsStream("aws.prop erties"));

} catch (IOException e) {

e.printStackTrace();

}

AmazonEC2Client amazonEC2Client = new AmazonEC2Client(credentials);

//our instance location we have to provide as endpoint

amazonEC2Client.setEndpoint("ec2.us-west-2.amazonaws.com");

//Instance creation from AMI

RunInstancesRequest runInstancesRequest = new RunInstancesRequest();

runInstancesRequest.withImageId("ami-25732b15")//ami id

.withInstanceType("t1.micro")

.withMinCount(1)

.withMaxCount(1)

.withKeyName("smishra")//key name should be generated by us

.withSecurityGroups("launch-wizard-20");//existing security groups name

RunInstancesResult runInstancesResult =

amazonEC2Client.runInstances(runInstancesRequest);

**Start, Stop and termination of an instance programmatically:**

//Starting of the instance

StartInstancesRequest startRequest = new StartInstancesRequest().withInstanceIds("i-e47b3be8");

StartInstancesResult startResult = amazonEC2Client.startInstances(startRequest);

System.out.println("Instance Started. . . ");

//Stopping of the instance

StopInstancesRequest stopRequest = new StopInstancesRequest().withInstanceIds("i-e47b3be8");

StartInstancesResult startResult = amazonEC2Client.startInstances(startRequest); System.out.println("Instance Stop. . . ");

//Terminate the instance

TerminateInstancesRequest terminateRequest=new TerminateInstancesRequest().withInstanceIds("i-e47b3be8");

TerminateInstancesResult terminationResult= amazonEC2Client.terminateInstances(terminateRequest); System.out.println("Instance Terminated . . . ");

**How to get the state of the Instance :**

DescribeInstanceStatusRequest describeInstanceRequest = new DescribeInstanceStatusRequest().withInstanceIds(instanceId);

DescribeInstanceStatusResult describeInstanceResult = amazonEC2Client.describeInstanceStatus(describeInstanceRequest);

List<InstanceStatus> state = describeInstanceResult.getInstanceStatuses();

//When state size is one then we got the created state

while (state.size() < 1) {

// Do nothing, just wait, have thread sleep if needed

describeInstanceResult = amazonEC2Client.describeInstanceStatus(describeInstanceRequest);

state = describeInstanceResult.getInstanceStatuses();

System.out.print("..");

}

System.out.println();

System.out.println("InstanceId: "+state.get(0).getInstanceId()+" , State "+state.get(0).getInstanceState().getName());

System.out.println("\nState size : "+state.size());

**How to get the status of the Instance:**

//Initially the instance status is ‘initializing’ after that the status will change to ‘2/2 checks passed’ //programmatically if we access then we get ‘OK’ as the status.

while(state.get(0).getInstanceStatus().getStatus().equalsIgnoreCase("initializing")){ describeInstanceResult = amazonEC2Client.describeInstanceStatus(describeInstanceRequest);

state = describeInstanceResult.getInstanceStatuses();

**try** {

Thread.*sleep*(30000);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}//while end

Amazon Elastic Map Reduce (Amazon EMR) is a web service that makes it easy to quickly and cost-effectively process vast amounts of data.

Amazon EMR uses Hadoop, an open source framework, to distribute your data and processing across a resizable cluster of Amazon EC2 instances. It can also run other distributed frameworks such as Spark and Presto. Amazon EMR is used in a variety of applications, including log analysis, web indexing, data warehousing, machine learning, financial analysis, scientific simulation, and bioinformatics. Customers launch millions of Amazon EMR clusters every year.

<http://aws.amazon.com/elasticmapreduce/>