### 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:

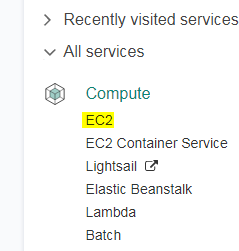
1. Login to AWS account and navigate to Amazon EC2 console.
2. In the navigation pane, choose **Elastic IPs**.
3. Choose Allocate new address.
4. Select Allocate. Close the confirmation screen.

**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.

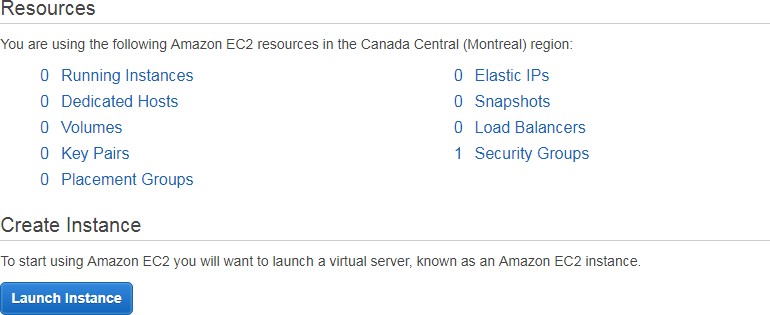
###### Instance launch process:

Login to Your AWS Account, Select and switch to the required Region and find

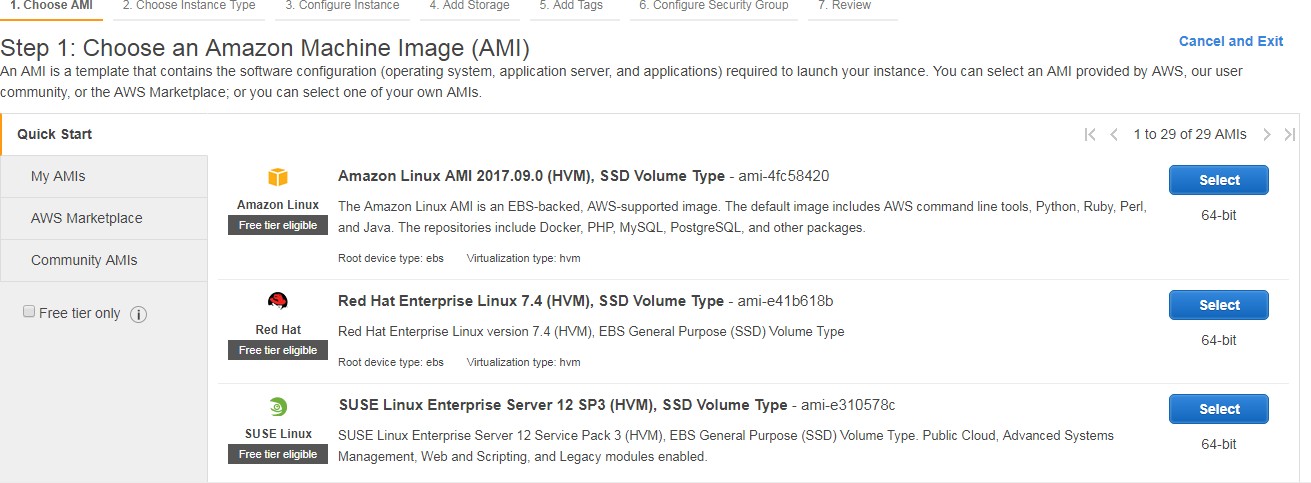
###### EC2 under Compute Section.



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.

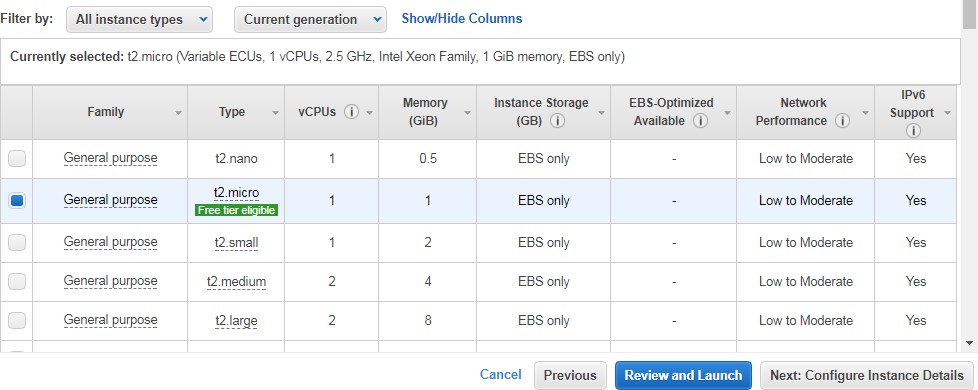


* We have Windows and Linux operating systems available here in Quick start option
* Along with the Quick Start option, you can also spin up your instances using the AWS Marketplace and the Community AMIs section. Both these options contains list of customized AMIs that have been created by either third-party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

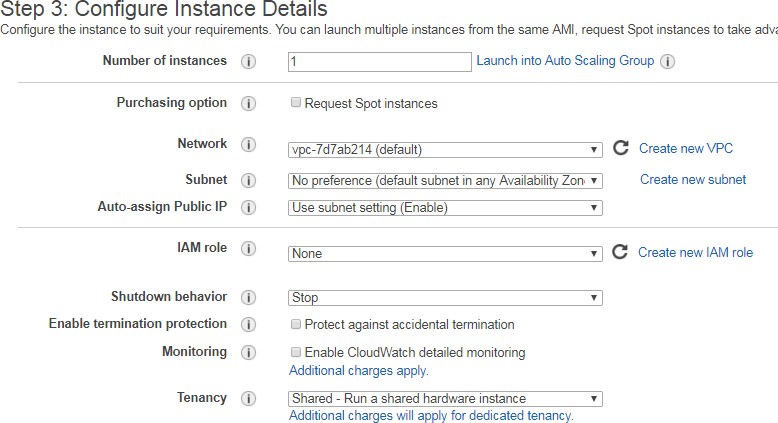
###### Choose an instance type

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

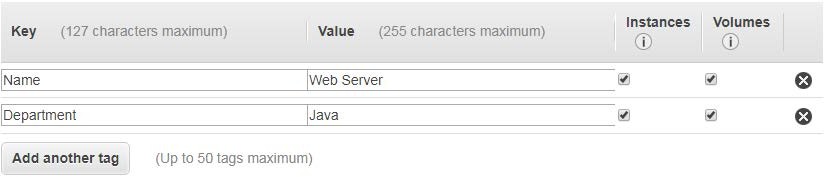
We can add EBS volumes to your instances. To add new volumes, simply click on the Add New Volume button. This will provide you with options to provide the size of the new volume along with its mount points. There is an 8 GB volume already attached to our instance. This is the t2.micro instance’s root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

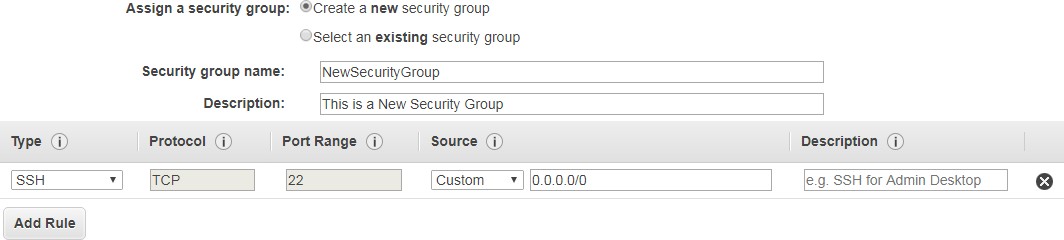


###### Step 6: Configure Security Group

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:

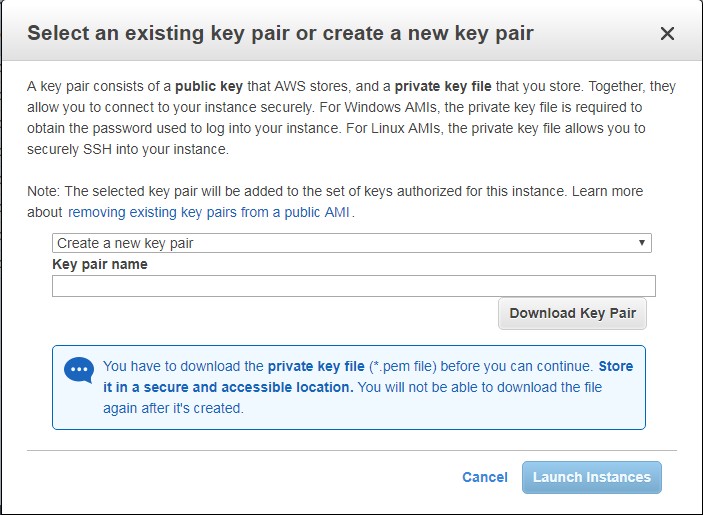
* You can create up to 500 security groups for each Amazon VPC.
* You can add up to 50 inbound and 50 outbound rules to each security group. If you need to apply more than 100 rules to an instance, you can associate up to five security groups with each network interface.
* You can specify allow rules,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

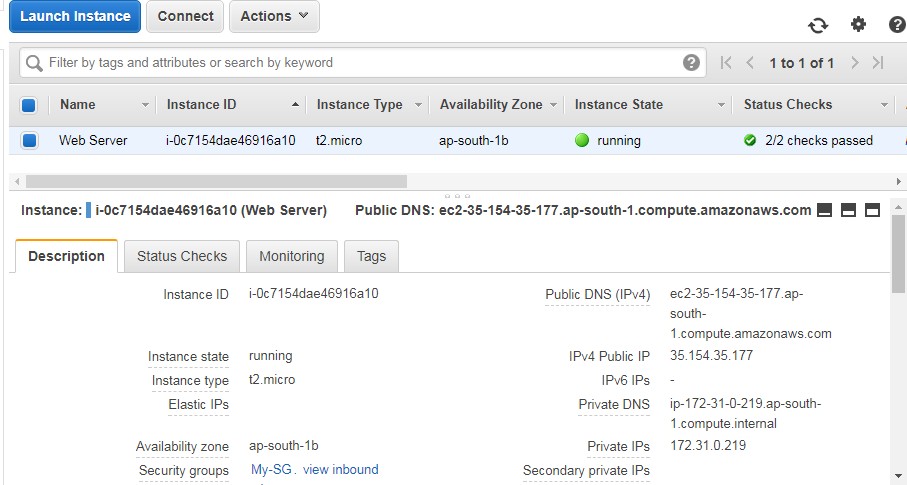
Then we have to associate a key pair to our instance.

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.



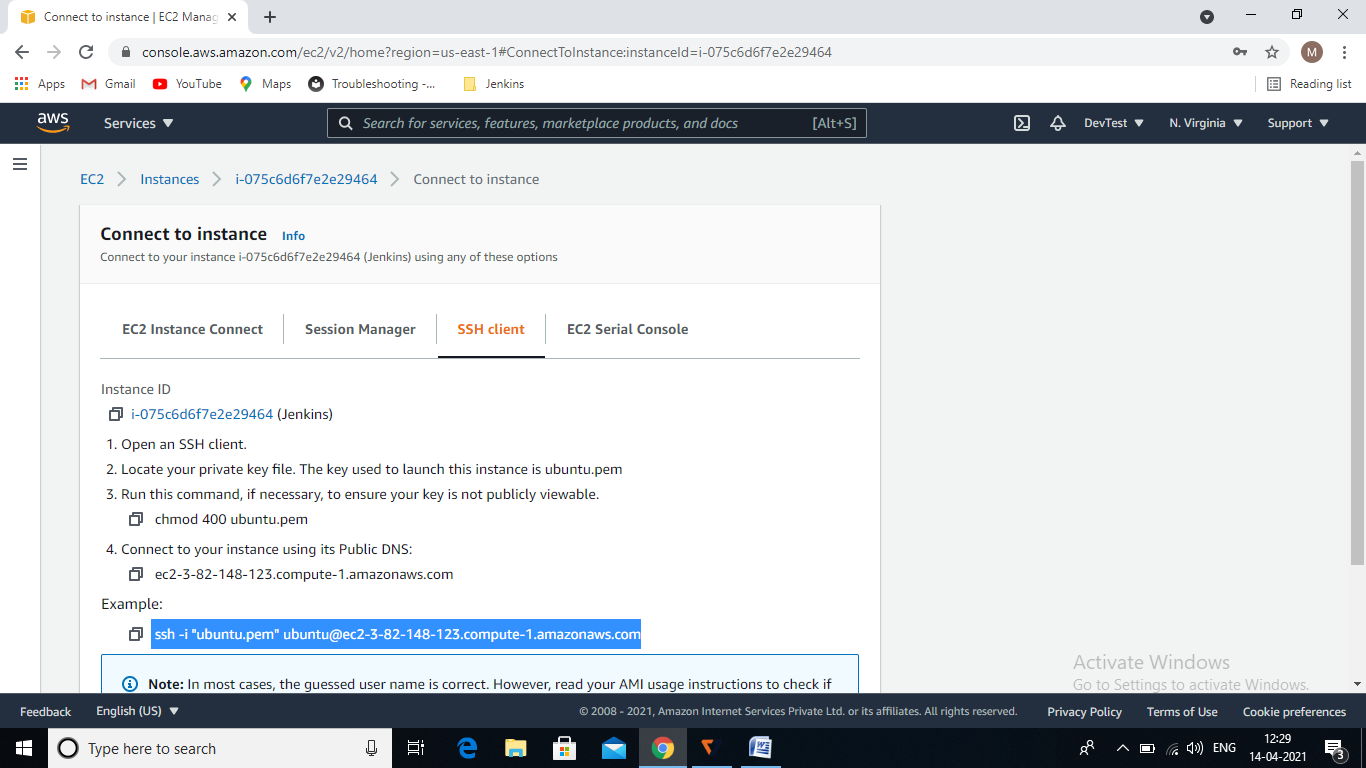
* The dashboard provides all of the information about our instance. We can view instance’s ID, instance type, IP information, AZ, Security Group, and a whole lot more info.
* We can also obtain instance’s health information using the Status Checks tab and the Monitoring tab.
* We can perform power operations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

**Connecting to Instance via ssh:**

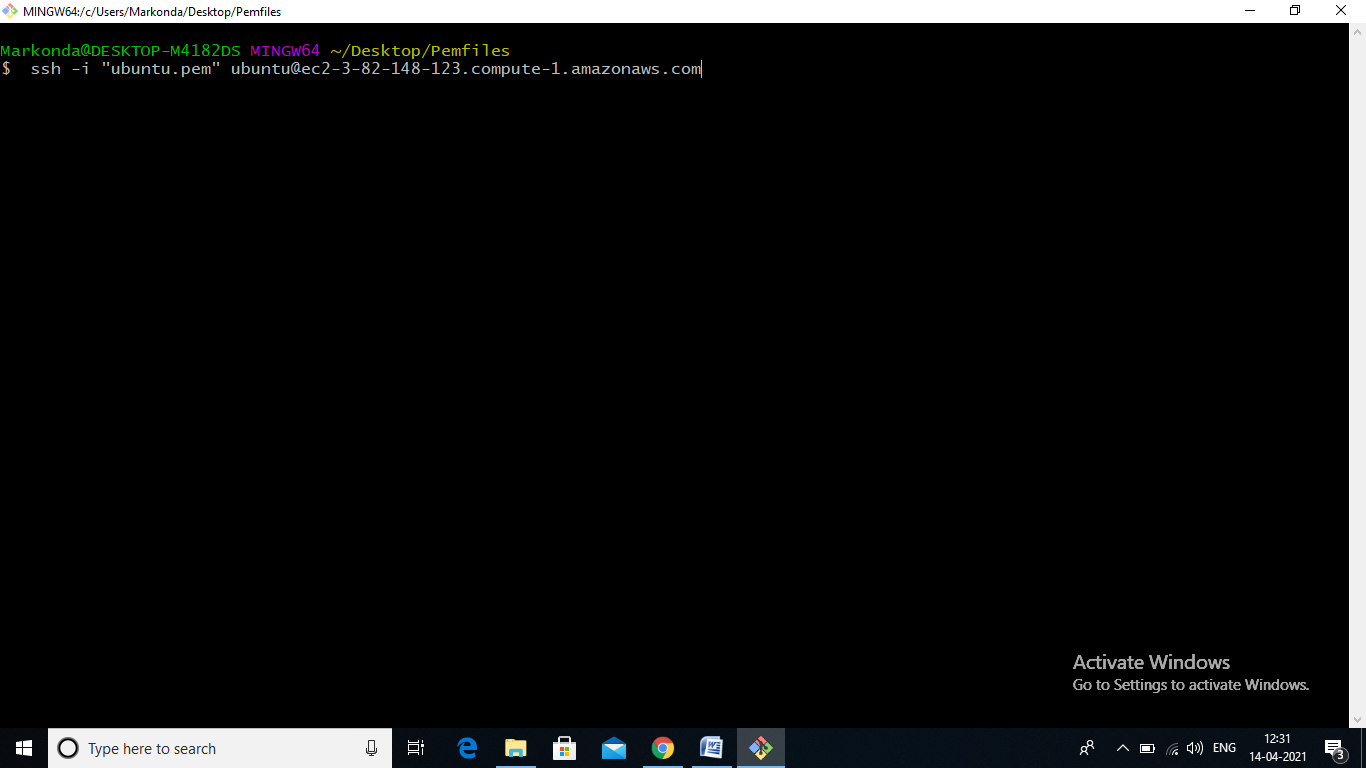
Goto AWS Instance -> Click on Instance -> Connect

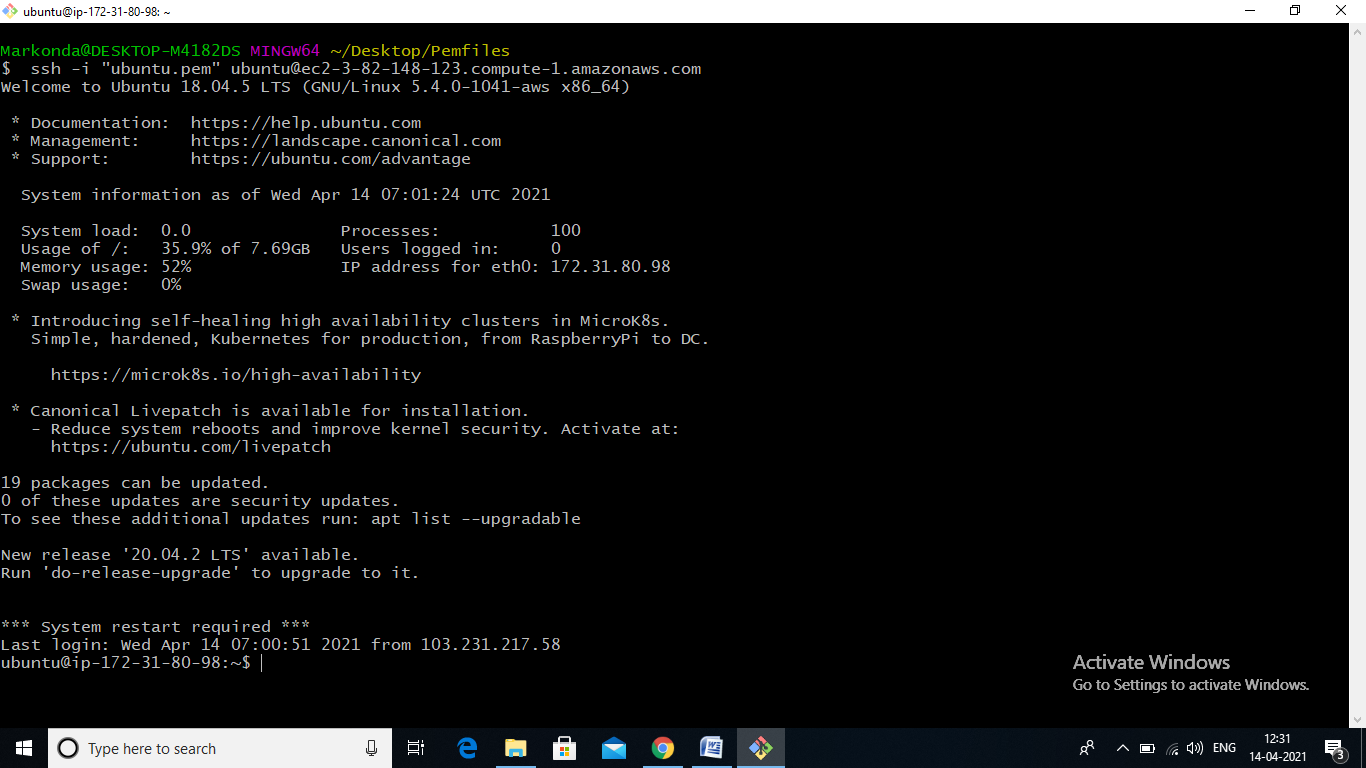
Goto -> SSH

Copy :  **ssh -i "ubuntu.pem"** [**ubuntu@ec2-3-82-148-123.compute-1.amazonaws.com**](mailto:ubuntu@ec2-3-82-148-123.compute-1.amazonaws.com)



Goto .pem file folder -> open GitBash Here



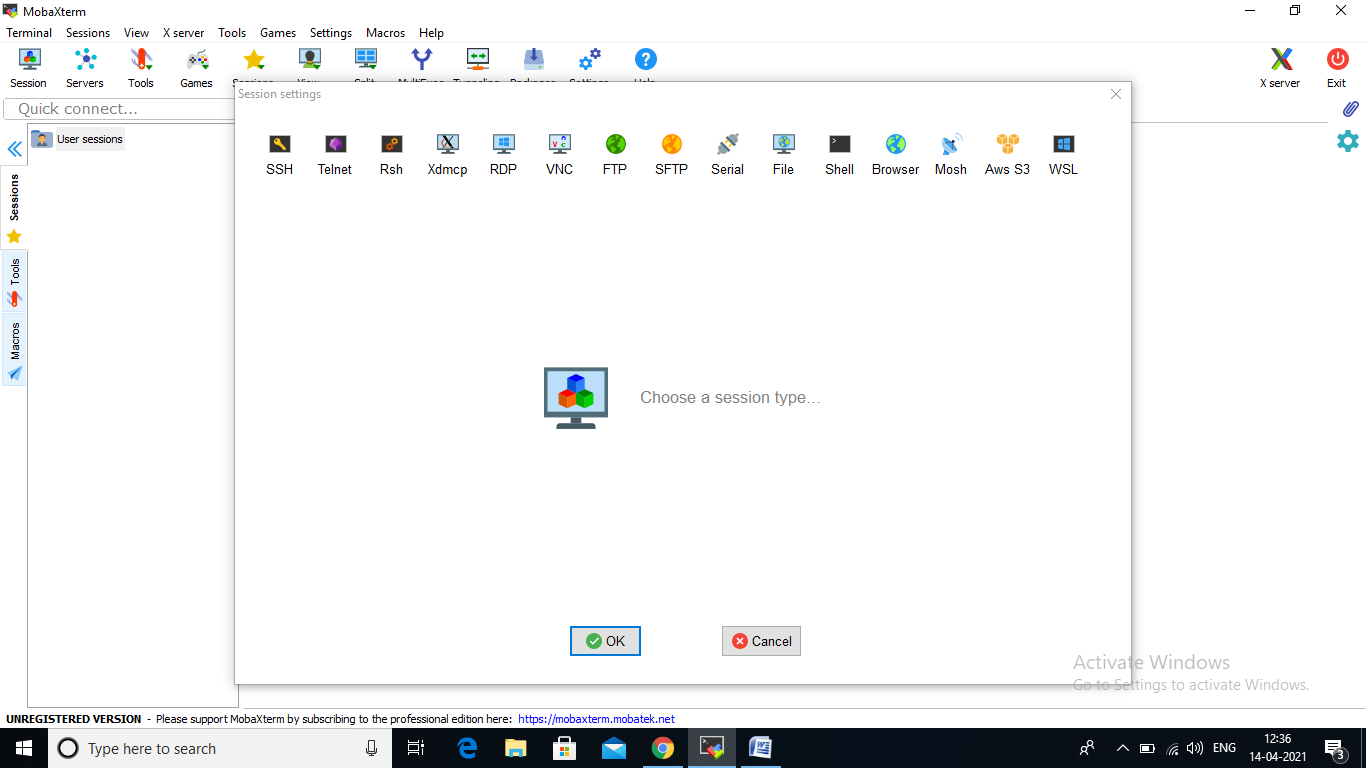


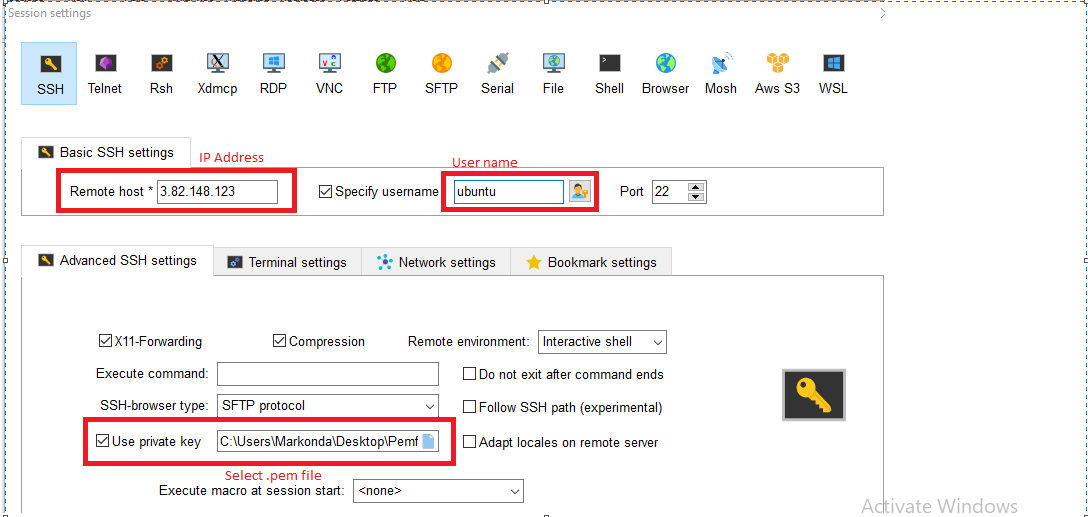
**Connecting to Instance via MobaXterm:**

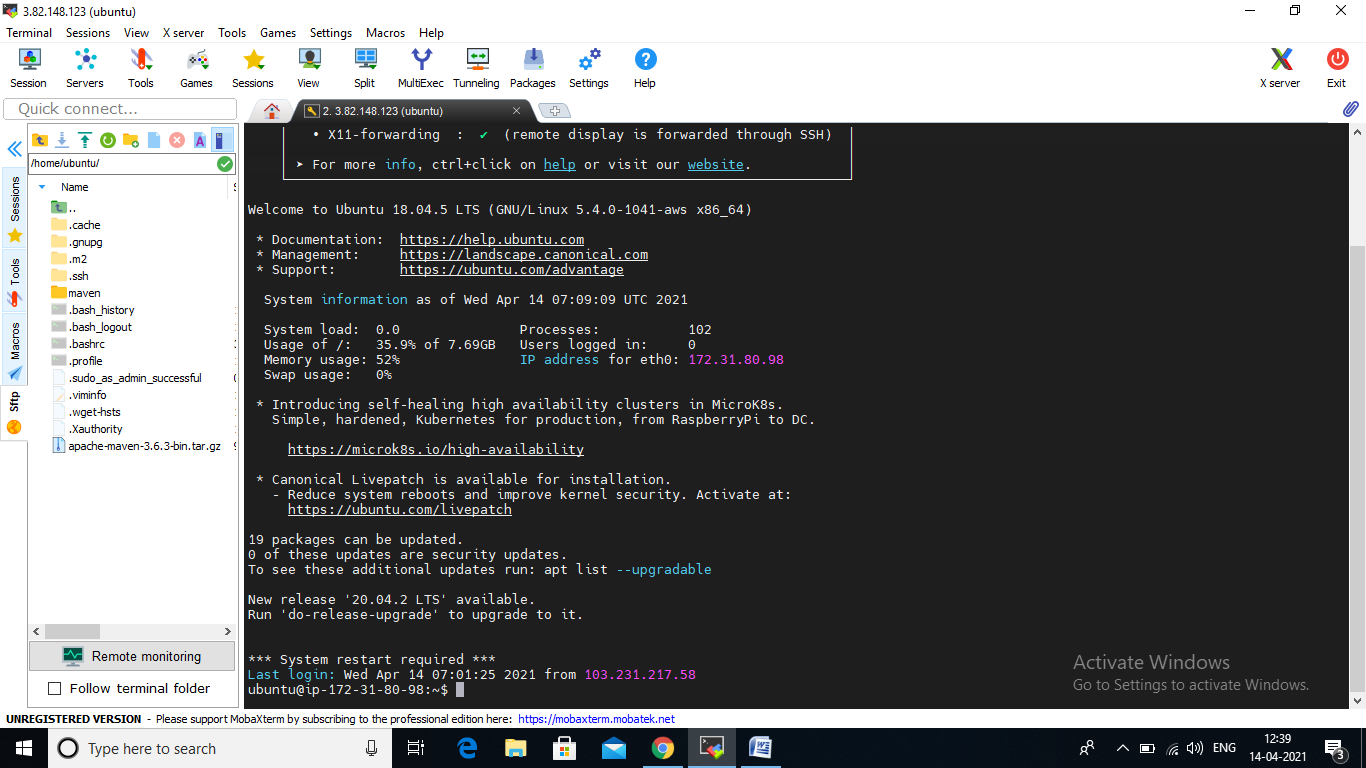
Once Goto AWS-> Instances -> Click on Instance ID

Copy public IP address: **3.82.148.123**

Goto to MobaExterm -> Top left Session -> SSH







**Connecting to Instance via PuTTY:**

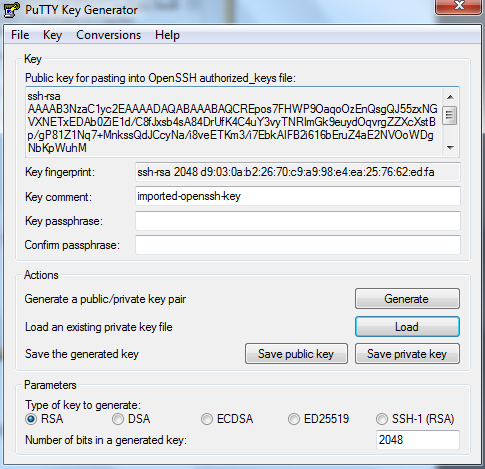
Once the instance is launched we have multiple options to connect to the instance. Mostly we can use **PuTTY** to connect Linux machines and **Remote Desktop** Feature for Windows Machine.

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](https://www.chiark.greenend.org.uk/%7Esgtatham/putty/latest.html)

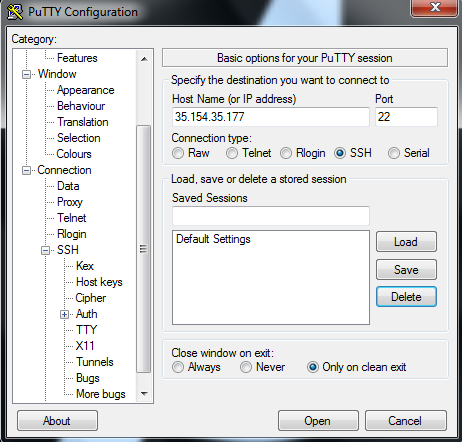
1. Download and install the latest copy of Putty and PuttyGen on local computer.
2. Launch PuttyGen and select the Load button and browse the downloaded Pem file (Which is created at the time of Instance launch).



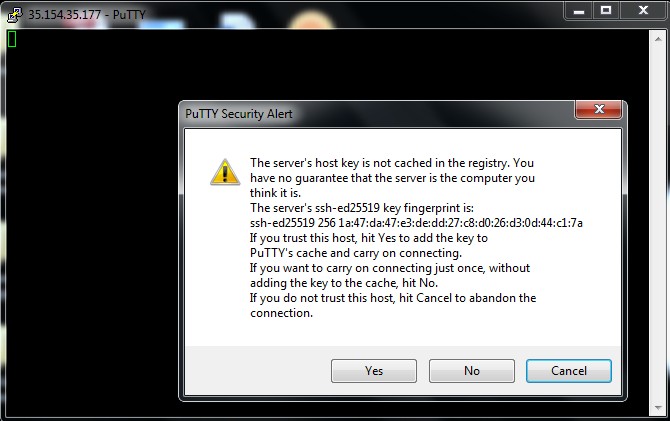
1. Once pem file is loaded, Select **“Save private key”** option.

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.**

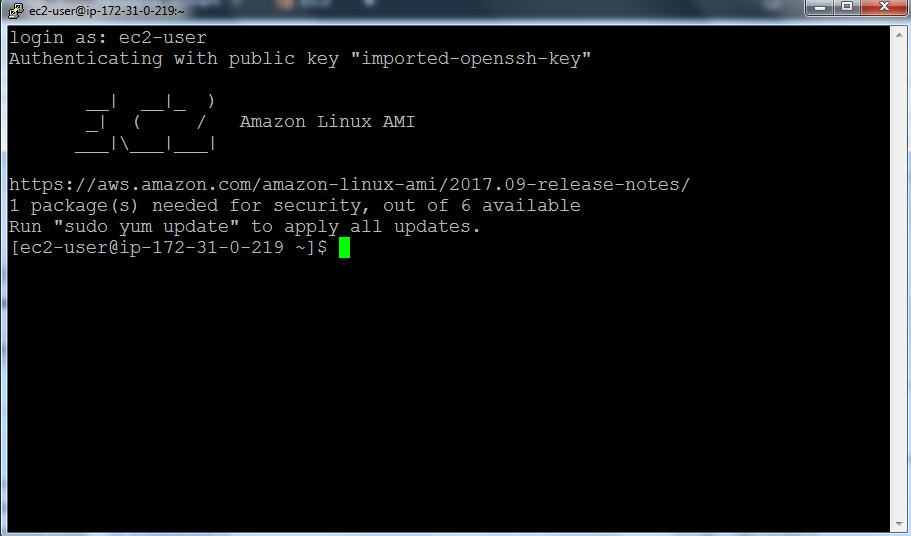
1. Provide a name and save the new file (\*.PPK) at a secure location. You can use this PPK file to connect to your instance using Putty
2. Please note down the **public IP address/ public DNS** of your instance.
3. Now open the **Putty** and enter the public IP in Host Name field and make sure to enter Port **22**



1. 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.
2. Give yes for on the Putty Security Alert.



1. 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
2. . 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.