**Chapter 3 : Elastic Compute Cloud (EC2)**

**Amazon EC2 (Elastic Compute Cloud)** is a web service interface that provides resizablecompute capacity in the AWS cloud. It is designed for developers to have complete control over web-scaling and computing resources.

EC2 instances can be resized and the number of instances scaled up or down as per our requirement. These instances can be launched in one or more geographical locations or regions, and **Availability Zones (AZs)**. Each region comprises of several AZs at distinct locations, connected by low latency networks in the same region.



**EC2 Components**



In AWS EC2, the users must be aware about the EC2 components, their operating systems support, security measures, pricing structures, etc.

**Operating System Support**

Amazon EC2 supports multiple OS in which we need to pay additional licensing fees like: Red Hat Enterprise, SUSE Enterprise and Oracle Enterprise Linux, UNIX, Windows Server, etc. These OS needs to be implemented in conjunction with Amazon Virtual Private Cloud (VPC).

**Security**

Users have complete control over the visibility of their AWS account. In AWS EC2, the security systems allow create groups and place running instances into it as per the

requirement. You can specify the groups with which other groups may communicate, as well as the groups with which IP subnets on the Internet may talk.

**Pricing**

AWS offers a variety of pricing options, depending on the type of resources, types of applications and database. It allows the users to configure their resources and compute the charges accordingly.

**Fault tolerance**

Amazon EC2 allows the users to access its resources to design fault-tolerant applications. EC2 also comprises geographic regions and isolated locations known as availability

zones for fault tolerance and stability. It doesn’t share the exact locations of regional data centers for security reasons.

When the users launch an instance, they must select an AMI that's in the same region where the instance will run. Instances are distributed across multiple availability zones to provide continuous services in failures, and Elastic IP (EIPs) addresses are used to quickly map failed instance addresses to concurrent running instances in other zones to avoid delay in services.

**Migration**

This service allows the users to move existing applications into EC2. It costs $80.00 per storage device and $2.49 per hour for data loading. This service suits those users having large amount of data to move.

**Features of EC2**



Here is a list of some of the prominent features of EC2:

* **Reliable**: Amazon EC2 offers a highly reliable environment where replacement ofinstances is rapidly possible. Service Level Agreement commitment is 99.9% availability for each Amazon EC2 region.
* **Designed for Amazon Web Services**: Amazon EC2 works fine with Amazonservices like Amazon S3, Amazon RDS, Amazon DynamoDB, and Amazon SQS. It provides a complete solution for computing, query processing, and storage across a wide range of applications.
* **Secure**: Amazon EC2 works in Amazon Virtual Private Cloud to provide a secureand robust network to resources.
* **Flexible Tools**: Amazon EC2 provides the tools for developers and systemadministrators to build failure applications and isolate themselves from common failure situations.
* **Inexpensive**: Amazon EC2 wants us to pay only for the resources that we use. Itincludes multiple purchase plans such as On-Demand Instances, Reserved Instances, Spot Instances, etc. which we can choose as per our requirement.

It is very difficult to predict how much computing power one might require for an application which you might have just launched. There can be two scenarios, you may over-estimate the requirement, and buy stacks of servers which will not be of any use, or you may under-estimate the usage, which will lead to the crashing of your application. In this AWS EC2 Tutorial we will understand all the key concepts using examples and also perform hands-on to launch an Ubuntu instance.

## **What is AWS EC2 ?**



Amazon Elastic Compute Cloud, EC2 is a web service from Amazon that provides **re-sizable** compute services in the cloud.

**How are they re-sizable?**

They are re-sizable because you can quickly scale up or scale down the number of server instances you are using if your computing requirements change.

**What is an Instance?**

An instance is a virtual server for running applications on Amazon’s EC2. It can also be understood like a tiny part of a larger computer, a tiny part which has its own Hard drive, network connection, OS etc. But it is actually all virtual. You can have multiple “tiny” computers on a single physical machine, and all these tiny machines are called Instances.

**Difference between a service and an Instance?**

Let’s understand it this way:

* EC2 is a service along with other Amazon Web Services like S3 etc.
* When we use EC2 or any other service, we use it through an instance, e.g. t2.micro instance, in EC2 etc.

## **Why AWS EC2 ?**

Why not buy your own stack of servers and work independently? Because, suppose you are a developer, and since you want to work independently you buy some servers, you estimated the correct capacity, and the computing power is enough. Now, you have to look after the updation of security patches every day, you have to troubleshoot any problem which might occur at a back end level in the servers and so on.These are all extra chores that you will be doing or maybe you will hire someone else to do these things for you.

But if you buy an EC2 instance, you don’t have to worry about any of these things as it will all be managed by Amazon; you just have to focus on your application. That too, at a fraction of a cost that you were incurring earlier! Isn’t that interesting?

**Let’s understand Cost Savings using an example.**

Suppose instead of taking AWS EC2, we consider taking a dedicated set of servers, so, what all we might have to face:

* Now for using these servers we have to hire an IT team which can handle them.
* Also, having a fault in the system is unavoidable, therefore we have to bear the cost of getting it fixed, and if you don’t want to compromise on your up-time you have to keep your systems redundant to other servers, which might become more expensive.
* Your own purchased assets will depreciate over the period of time, however, as a matter of fact the cost of an instance have dropped more than 50% over a 3-year period, while improving processor type and speed. So eventually, moving to Cloud is all more suggested.
* For scaling up we have to add more servers, and if your application is new and you experience a sudden traffic, scaling up that quickly might become a problem.

These are just a few problems and there are many others scenarios which make the case for EC2 stronger!

## **Let’s understand the types of EC2 Computing Instances:**

Computing is a very broad term, the nature of your task decides what kind of computing you need.

Therefore, AWS EC2 offers 5 types of instances which are as follows:

* *General Instances*
  + For applications that require a balance of performance and cost.
    - E.g email responding systems, where you need a prompt response as well as the it should be cost effective, since it doesn’t require much processing.
* *Compute Instances*
  + For applications that require a lot of processing from the CPU.
    - E.g analysis of data from a stream of data, like Twitter stream
* *Memory Instances*
  + For applications that are heavy in nature, therefore, require a lot of RAM.
    - E.g when your system needs a lot of applications running in the background i.e multitasking.
* *Storage Instances*
  + For applications that are huge in size or have a data set that occupies a lot of space.
    - E.g When your application is of huge size.
* *GPU Instances*
  + For applications that require some heavy graphics rendering.
    - E.g 3D modelling etc.

**Now, every instance type has a set of instances which are optimized for different workloads:**

* General Instances
  + t2
  + m4
  + m3
* Compute Instances
  + c4
  + c3
* Memory Instances
  + r3
  + x1
* Storage Instances
  + i2
  + d2
* GPU Instances
  + g2

Now let’s understand the kind of work that each instance is optimized for, in this AWS EC2 Tutorial:

**Burstable Performance Instances**

* *T2 instances* are burstable instances, meaning the CPU performs at a baseline, say 20% of its capability. When your application needs more than 20% of the performance of the CPU, the CPU enters into a burst mode giving higher performance for a limited amount of time, therefore work happens faster.
  + You get these credits when your CPU is idle.
  + Each CPU credit gives a burst of 1 minute to the CPU.
  + If your CPU credits are not used they are credited to your account and they stay there for 24 hours.
  + Based on your credit balance, you can decide whether the t2 instance, should be scaled up or down.
  + These bursts happen at a cost, every time a burst happens in a CPU, CPU credits are used.

### **EBS-optimized Instances**

* *C4, M4, and D2 instances*, are EBS optimized by default, EBS means Elastic Block Storage, which is a storage option provided by AWS in which the IOPS\* rate is quite high. Therefore, when an EBS volume is attached to an optimized instance, single digit millisecond latencies can be achieved.

\*IOPS (Input/Output Operations Per Second, pronounced eye-ops) is a performance measurement used to characterize computer storage devices.

**Cluster Networking Instances**

* *X1, M4, C4, C3, I2, G2 and D2 instances* support cluster networking. Instances launched into a common placement group are put in a logical group that provides high-bandwidth, low latency between all the instances in the group.
  + A placement group is basically a logical cluster where some select EC2 instances which are a part of that group can utilize up to 10Gbps for single flow and 20Gbps for multi flow traffic in each direction.
  + Instances which are not a part of that group are limited to 5 Gbps speed in multi flow traffic. Cluster Networking is ideal for high performance analytics system.

**Dedicated Instances**

* They are the instances that run on single-tenant hardware dedicated to a single customer.
* They are perfect for workloads where a corporate policy or industry regulation requires that your instance should be isolated from any other customer’s instance, therefore they go for their own separate machines, and their instances are isolated at the hardware level.

Let’s understand this through an example. Suppose in our company Edureka, we have the following tasks:

* Analysis of customer’s data
  + Customer’s website activity, etc. should all be monitored in real-time. There will be times when the traffic on the website will be minimum, therefore using a very powerful processor should not be considered, since it will become expensive for the company because it will not be used for every hour of the day. Hence, for this task, we might take **t2 instances** because they give **Burstable CPU performance** i.e when the traffic will be more the CPU performance will be increased accordingly to meet the requirements.
* Our auto-response emailing system
  + It should be quick, therefore we would require systems, where the response time is as short as possible. This could be achieved by using **EBS optimized instances**, as they offer high IOPS and hence, low latencies.
* The search engine on our website
  + It should be able to sort the keywords and return relevant results, therefore we might have 2 servers for this. One is the database and the other server for processing the keywords. Therefore, the communication between these servers should be at the maximum possible rate. To achieve this, we can put them in a placement group and for that we have to use **Cluster Networking Instances.**
* Some processes in every organisation are highly confidential
  + Because these processes give us an edge over other companies, no matter how secure the servers, maybe, some policies are still made to be sure. Therefore, we might use **Dedicated Instances** for these kind of processes.

We now know about instances, let’s learn how to launch these instances?

## **How to run systems in EC2 ?**

* Login to your AWS account and click on AWS EC2.
* Under create instance, click on launch instance.

Now you have to select an **Amazon Machine Image (AMI),** AMIs are templates of OS and they provide the information needed to launch an instance.

When we want to launch an instance we have to specify which AMI we want to use. It could be Ubuntu, windows server etc.

* The AMIs could be preconfigured or you can configure it on your own according to your requirements.
  + For preconfigured AMIs you have to select it from AWS marketplace.
  + For setting up your own, go to quick-start and select one.
* While configuring you will reach a point where you have to select an **EBS** storage option.

**Elastic Block Storage (EBS)** is a persistent block level storage volumes which are used with EC2. Here each block acts as a hard drive.

But why do we need EBS with EC2?

Just like your computer needs a hard drive, you need AWS EC2 Tutorial, AWS EC2 needs a storage volume to store the OS that your instance will be specifying. Options for EBS are:

**Provisioned IOPS**: This category is for workloads which are mission critical, it provides high IOPS rates.

**General Purpose:** It is for workloads which need a performance and cost balance.

**Magnetic:** It is for data which is accessed less frequently, and also retrieval time is more.

* After selecting a suitable option in EBS, we give the instance a name and then we create a **security group**.
* A security group acts as a firewall to control inbound and outbound traffic. Each security group has rules according to which the traffic is governed.
* Each instance, can be assigned up to 5 security groups.
* Finally, in the last step the console shows all the sethertings that you have done, you can verify and launch it.

Now in this AWS EC2 Tutorial, when you launch the instance, it will need some authentication from you; otherwise anyone can access your instance! To address that in this AWS EC2 Tutorial, let’s take a plunge into the security aspects of EC2,

**Security in AWS EC2**

To authenticate users to their instances, AWS employs a **key pair** method.

**What is key pair?**

Amazon EC2 uses public–key cryptography to encrypt and decrypt login information. Public–key cryptography uses a public key to encrypt a piece of data, such as a password, then the recipient uses the private key to decrypt the data. The public and private keys are known as a *key pair*.

**Additional Benefits:** Every service from Amazon is designed keeping in mind the customer. They claim to be the earth’s most customer-obsessed company. Having said that let’s understand some other benefits of EC2.

## **Auto Scaling**

Auto Scaling is a service designed by AWS EC2, which automatically launch or terminate EC2’s instances based on user defined policies, schedules and health checks.

**Elastic Load Balancing**

Elastic Load Balancing (ELB) automatically distributes incoming application traffic across multiple EC2 instances, in multiple Availability Zones.

Availability zones are basically places where amazon has set up their servers. Since they have customers from the whole globe, they have set up multiple Availability zones to reduce the latency.

Elastic IP Addresses are static IP addresses which are associated with your AWS account, they can be used to mask the failure of an instance by automatically remapping your address to another working instance in your account.

## **AWS EC2 Pricing**

In this AWS EC2 Tutorial, let’s start with the free things first!

AWS EC2 free tier allows 750 hrs of t2.micro instance usage per month!

The free tier for EC2 is valid for 1 year from SignUp of your AWS account.

There are basically 3 pricing options in EC2:

* Spot Instances
* On Demand Instances
* Reserved Instances

Spot Instances is a pricing option which enables you to bid on unused EC2 instances. The hourly price for a Spot Instance is set by AWS EC2, and it fluctuates according to the availability of the instances in a specific Availability zone.

* Basically, you will set a price for an instance above which you do not wish to get charged for.
* The price that you set is for per hour basis, therefore the moment the price for that instance becomes greater than what you have set, the instance gets shut down automatically.

On Demand Instances are used when you want to pay for the hour, with no long term commitments and upfront payments. They are useful for applications that may have unpredictable workloads or for test applications that are being deployed for the first time.

Reserved Instances provide you with significant discounts as compared to On Demand Instances. With Reserved Instances you reserve instances for a specific period of time with three payment options:

* No Upfront
* Partial Upfront
* Full Upfront

And two term lengths:

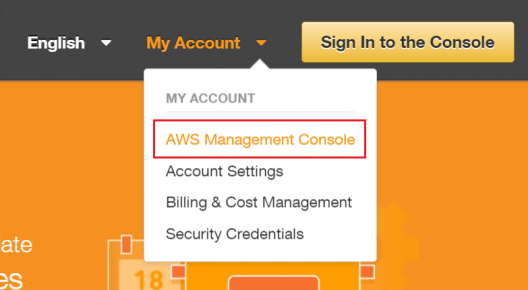
* One Year Term
* Three Year Term

The higher the upfront payment is, the more you save money.

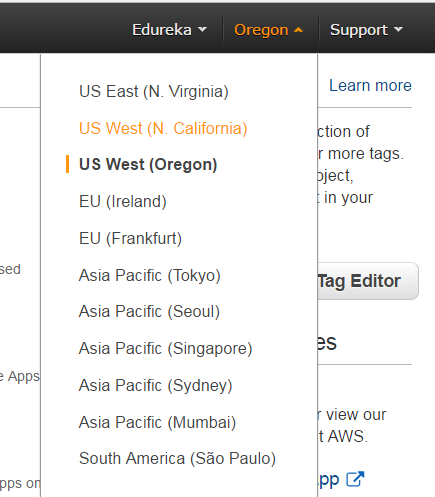
## **AWS EC2 Use Case**

Next in this AWS EC2 Tutorial, let’s understand the whole EC2 instance creation process through a use case in which we’ll be creating an Ubuntu instance for a test environment.

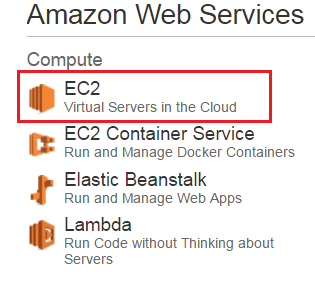
* **Login to AWS Management Console.**



* **Select your preferred Region.** Select a region from the drop down, the selection of the region can be done on the basis of the criteria discussed earlier in the blog.



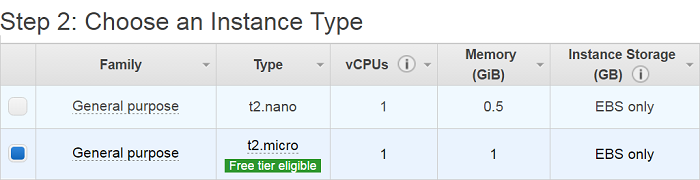
* **Select EC2 Service** Click EC2 under Compute section. This will take you to EC2 dashboard.



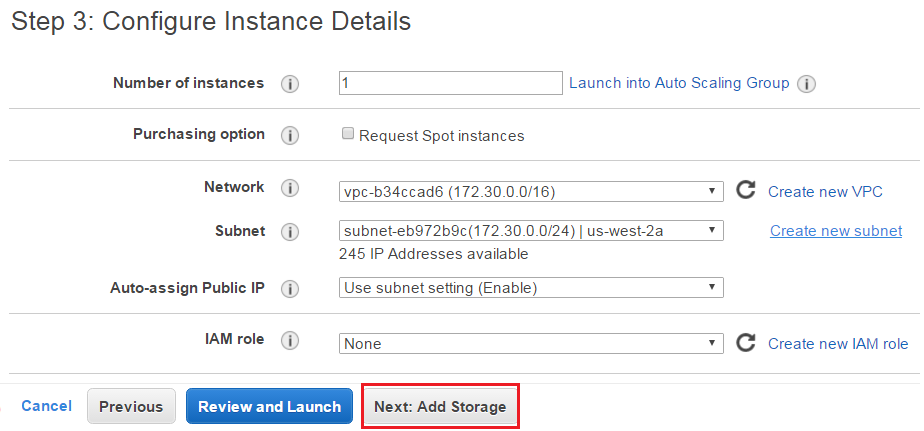
* Click **Launch Instance**.
* **Select an AMI :** because you require a Linux instance, in the row for the basic 64-bit Ubuntu AMI, click Select.



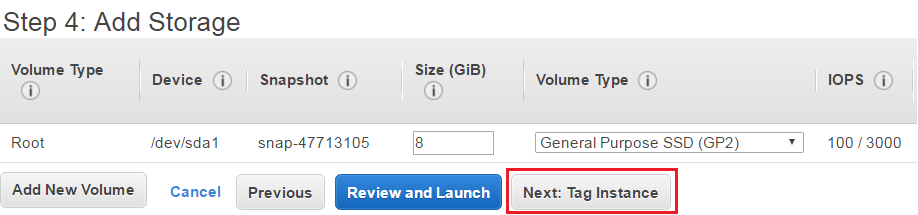
* **Choose an Instance**

Select t2.micro instance, which is free tier eligible.  


* **Configure Instance Details.**Configure all the details and then click on add storage

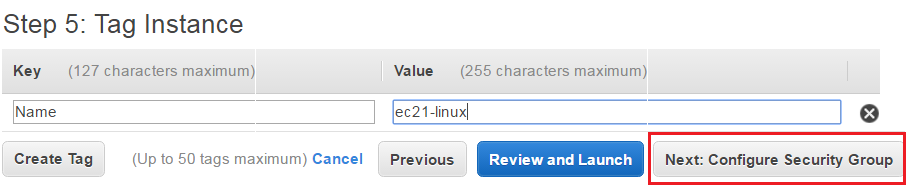


* **Add Storage**

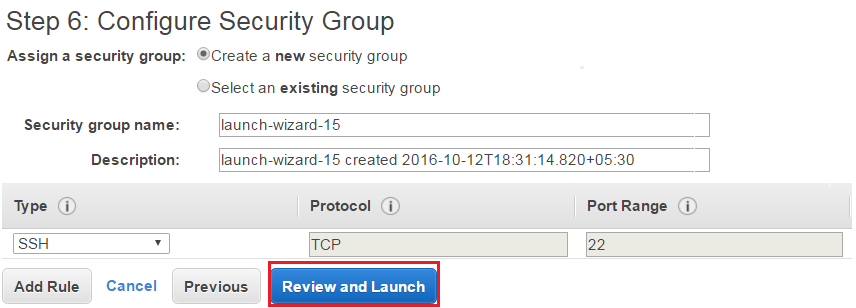


* **Tag an Instance**

Type a name for your AWS EC2 instance in the value box. This name, more correctly known as tag, will appear in the console when the instance launches. It makes it easy to keep track of running machines in a complex environment. Use a name that you can easily recognize and remember.

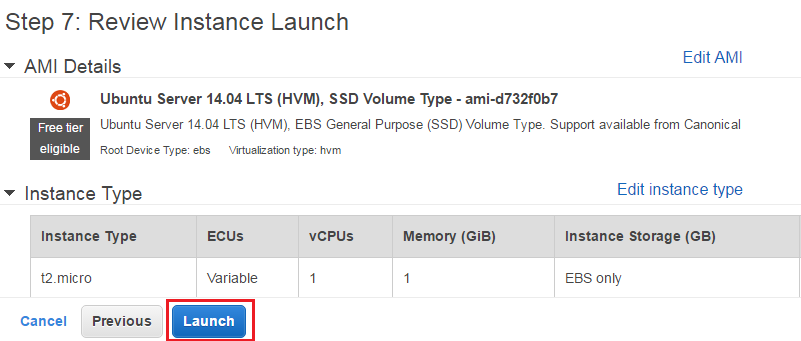


* **Create a Security Group**



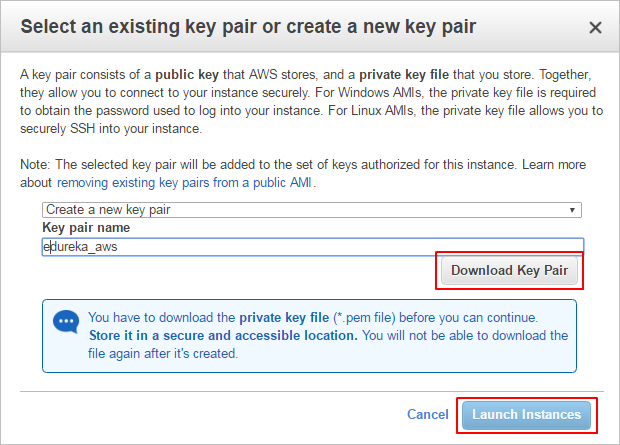
* **Review and Launch an Instance**

Verify the details that you have configured to launch an instance.

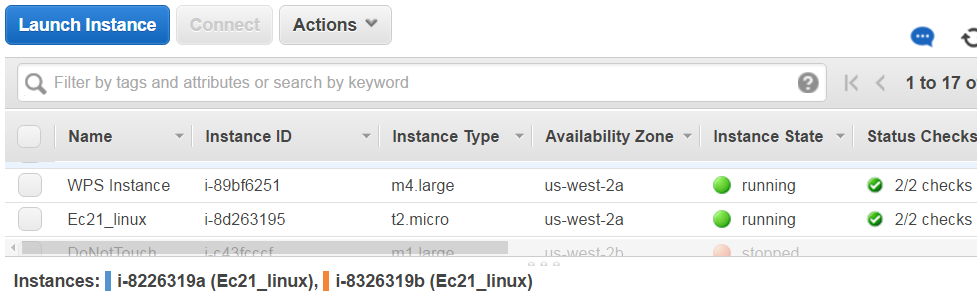


* **Create a Key Pair & launch an Instance**

Next in this AWS EC2 Tutorial, select the option ‘Create a new key pair’ and give a name of a key pair. After that, download it in your system and save it for future use.



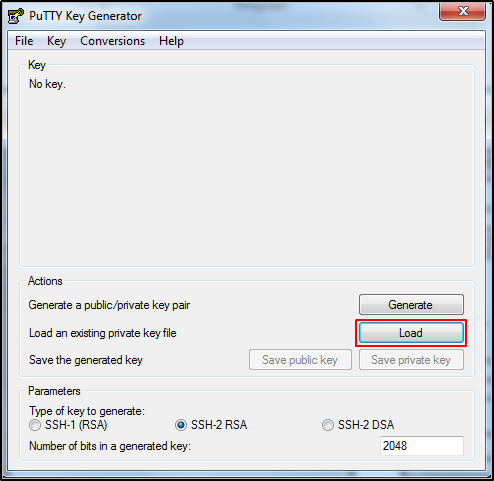
* **Check the details of a launched instance.**



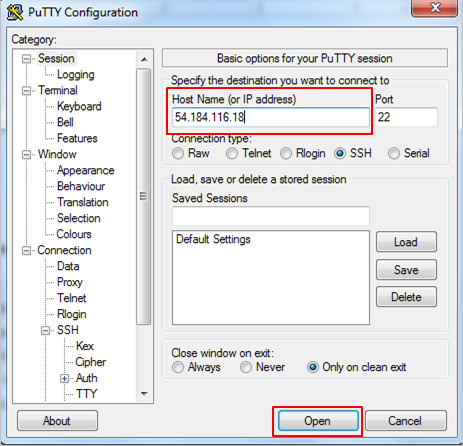
* **Converting Your Private Key Using PuTTYgen**

PuTTY does not natively support the private key format (.pem) generated by Amazon EC2. PuTTY has a tool called PuTTYgen, which can convert keys to the required PuTTY format (.ppk). You must convert your private key into this format (.ppk) before attempting to connect to your instance using PuTTY.

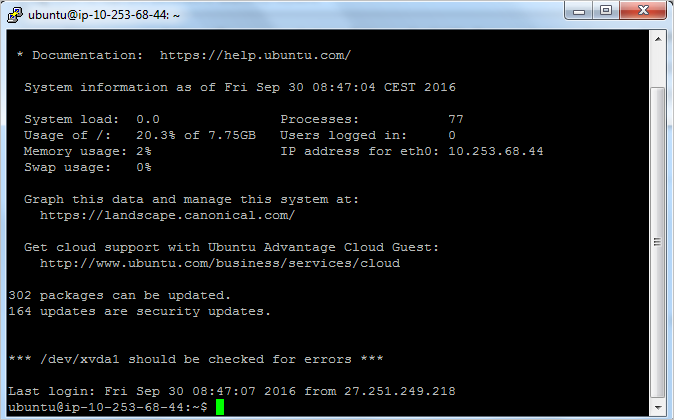
* Click Load. By default, PuTTYgen displays only files with the extension .ppk. To locate your .pem file, select the option to display files of all types.



* Select your.pem file for the key pair that you specified when you launch your instance, and then click Open. Click OK to dismiss the confirmation dialog box.
* Click Save private key to save the key in the format that PuTTY can use. PuTTYgen displays a warning about saving the key without a passphrase. Click Yes.
* Specify the same name for the key that you used for the key pair (for example, my-key-pair). PuTTY automatically adds the. ppk file extension.
* **Connect to EC2 instance using SSH and PuTTY**
* Open PuTTY.exe
* In the Host Name box, enter Public IP of your instance.
* In the Category list, expand SSH.
* Click Auth (don’t expand it).
* In the Private Key file for authentication box, browse to the PPK file that you downloaded and double-click it.
* Click Open.



* Type in Ubuntu when prompted for login ID.



**Congratulations! You have launched an Ubuntu Instance successfully.**