Connecting to Instance

Introduction

We are now ready to connect to the instance with SSH

First, we will see how to connect to these systems from Mac and Linux machines.

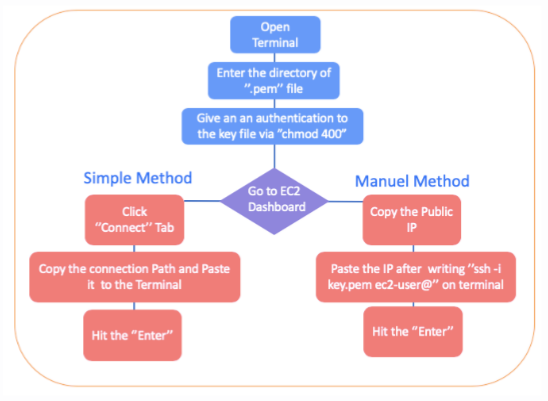
In fact, we can use the terminal of MAC computers and Linux Machines. These options don't require an additional application.

As for Windows, we need to download **Windows Subsystem for Linux** and some additional applications and files.

We'll use **Bash Terminal** for now. But you'll also learn how to connect from the Windows machine.

First of all, we should save the key file that we download before, in a folder named KEY on the desktop or elsewhere you determined.

SSH Connection Method and Workflows


Open Terminal
Enter the directory of ".pem" file
Give an an authentication to the key file via "chmod 400"
Simple Method
Go to EC2 Dashboard
Manuel Method
Click "Connect" Tab
Copy the Public IP
Copy the connection Path and Paste it to the Terminal
Paste the IP after writing "ssh -i key.pem ec2-user@" on terminal
Hit the "Enter"
Hit the "Enter"

EC2 Connection Workflows

We can connect to an instance in different ways. But for now, we'll see 2 of them, Simple Method and Manuel Method, as you see in the picture above.

Authentication of Key File

First of all, we access the key (.pem) file that we saved on the desktop or where you prefer to save.

To do this, we will access the file directory by entering the following command from the Bash Terminal and then complete the preparations by entering the following command. This command means as you remember from Linux Basic Lessons, go to the KEY folder under the desktop.

```
cd desktop/KEY
```

💡Tips:

- cd desktop/KEY is the file path of key.pem file. It is located on Desktop of my computer for this scenario . But, your's may be located on C:/ or somewhere else. Be sure to write here your file path correctly

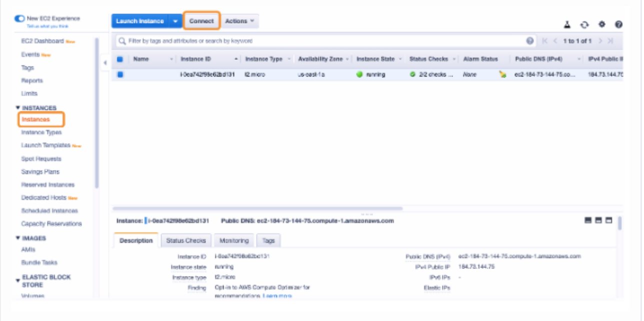
And then, we will give an authentication to this key pair with this code seen below.

```
chmod 400 key.pem
```

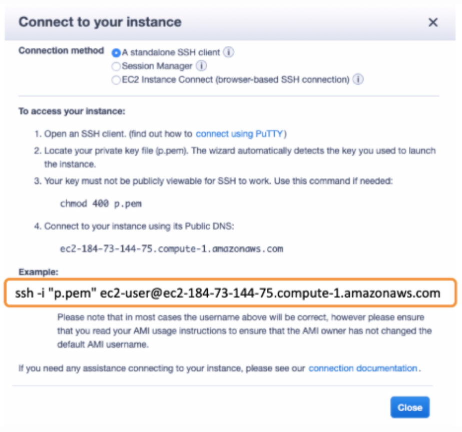
Thus, we give an authentication via **chmod 400** command. But don't forget, we don't need to enter this command every time, we only enter it once for Mac and Linux Terminals.

Let's connect to the instance.

Connecting to EC2 Instance on Linux/Mac-Simple Method



- After authentication of the key file, first go to the EC2 Dashboard.
- Then Click the **Connect** tab as you see in the picture above.



Connect to your instance

Connection method
- A standalone SSH client
- Session Manager
- EC2 Instance Connect (browser-based SSH connection)

To access your instance:

1. Open an SSH client. (find out how to connect using PuTTY)
2. Locate your private key file (p.pem). The wizard automatically detects the key you used to launch the instance.
3. Your key must not be publicly viewable for SSH to work. Use this command if needed:
   `chmod 400 p.pem`
4. Connect to your instance using its Public DNS:
   `ec2-184-73-144-75.compute-1.amazonaws.com`

Example:

`ssh -i "p.pem" ec2-user@ec2-184-73-144-75.compute-1.amazonaws.com`

Please note that in most cases the username above will be correct, however please ensure that you read your AMI usage instructions to ensure that the AMI owner has not changed the default AMI username.

If you need any assistance connecting to your instance, please see our connection documentation.

Close

Connect Page

- Connect page will pop up on the screen, then copy the path script that is marked like in the picture above.
- After that, paste this path script to the Bash Terminal screen and hit **Enter**

```
ssh -i "key.pem" ec2-user@ec2-184-73-144-75.compute-1.amazonaws.com
```

Then, Bash Terminal asks you **Yes /No**, write **Yes** then hit **Enter** again.

Now, we achieved to connect instance. You'll see these scripts seen below on your Bash Terminal monitor:

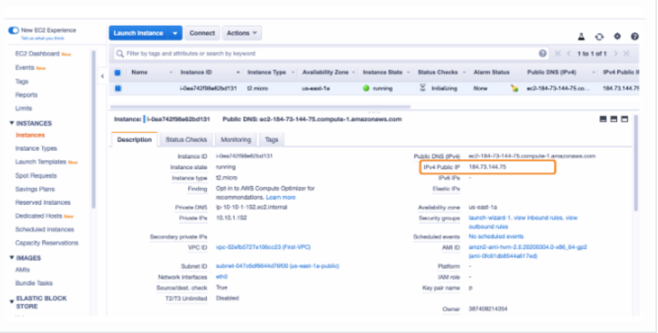
```
       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-34.254.185.28 ~]$
```

Congratulations !!

Connecting to EC2 Instance on Linux/Mac-Manuel Method

Before connecting to the instance, go to the EC2 dashboard and copy the Public IP of your instance.



Copy the Public IP

Write the following command (ssh -i key.pem ec2-user) and after "@" paste the Public IP that you copied just before as you see below. Then, hit the **Enter** to connect to EC2.

```
ssh -i key.pem ec2-user@34.254.185.28
```

💡Tips:

- "key.pem" is the name of my key file. While creating our instance we determine this name. So it can be different for everybody. Be sure to write here your file name of the key correctly

After that, Bash Terminal asks you **Yes /No**, write **Yes** then press **Enter**.

After that, Bash Terminal asks you **Yes /No**, write **Yes** then press **Enter**.

Now, we achieved to connect instance. You'll see these scripts seen below on your Bash Terminal monitor.

```
       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-34.254.185.28 ~]$
```

But, there is one step to do. After connecting to the instance, we need to enter the following command in order to use the root privileges for this instance.

```
sudo su
```

And you will see this on Bash Terminal monitor :

```
[root@ip-172-31-41-196 ec2-user]#
```

Congratulations !!

Connecting to Instance on Windows

In order to connect to the machine with SSH on Windows, we need to use third-party applications.

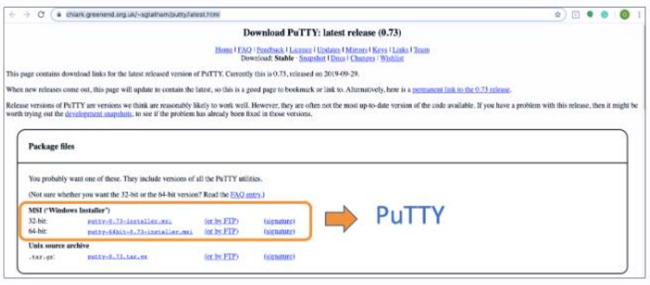
To do this, we will need PuTTy and PuTTy Gen.

PuTTY is an SSH client, which allows you to remotely log into other computers.

PuTTYgen is a program that can generate SSH key pairs, which are special files you can use for encryption, authentication, and so on.

> You can download both of them from the link following.
> /www.chiark.greenend.org.uk

PuTTY and PuTTYgen Download



chiark.greenend.org.uk/~sgtatham/putty/latest.html

Download PuTTY: latest release (0.73)

Home | FAQ | Feedback | Licence | Updates | Mirrors | Keys | Links | Team
Download: **Stable** · Snapshot | Docs | Changes | Wishlist

This page contains download links for the latest released version of PuTTY. Currently this is 0.73, released on 2019-09-29.

When new releases come out, this page will update to contain the latest, so this is a good page to bookmark or link to. Alternatively, here is a permanent link to the 0.73 release.

Release versions of PuTTY are versions we think are reasonably likely to work well. However, they are often not the most up-to-date version of the code available. If you have a problem with this release, then it might be worth trying out the development snapshots, to see if the problem has already been fixed in those versions.

Package files

You probably want one of these. They include versions of all the PuTTY utilities.

(Not sure whether you want the 32-bit or the 64-bit version? Read the FAQ entry.)

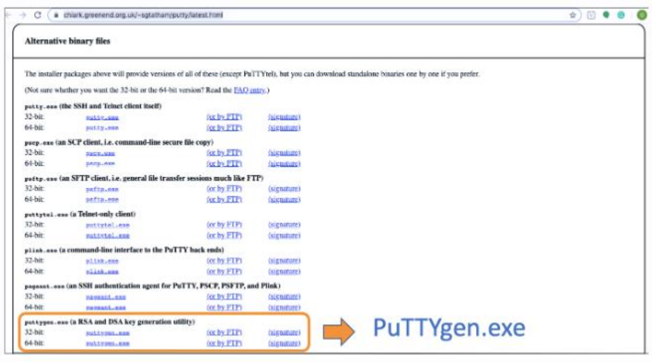
MSI ('Windows Installer')

32-bit:	putty-0.73-installer.msi	(or by FTP)	(signature)
64-bit:	putty-64bit-0.73-installer.msi	(or by FTP)	(signature)

Unix source archive

.tar.gz:	putty-0.73.tar.gz	(or by FTP)	(signature)

Download PuTTY



chiark.greenend.org.uk/~sgtatham/putty/latest.html

Alternative binary files

The installer packages above will provide versions of all of these (except PuTTYtel), but you can download standalone binaries one by one if you prefer.

(Not sure whether you want the 32-bit or the 64-bit version? Read the FAQ entry.)

putty.exe (the SSH and Telnet client itself)			
32-bit:	putty.exe	(or by FTP)	(signature)
64-bit:	putty.exe	(or by FTP)	(signature)
pscp.exe (an SCP client, i.e. command-line secure file copy)			
32-bit:	pscp.exe	(or by FTP)	(signature)
64-bit:	pscp.exe	(or by FTP)	(signature)
psftp.exe (an SFTP client, i.e. general file transfer sessions much like FTP)			
32-bit:	psftp.exe	(or by FTP)	(signature)
64-bit:	psftp.exe	(or by FTP)	(signature)
puttytel.exe (a Telnet-only client)			
32-bit:	puttytel.exe	(or by FTP)	(signature)
64-bit:	puttytel.exe	(or by FTP)	(signature)
plink.exe (a command-line interface to the PuTTY back ends)			
32-bit:	plink.exe	(or by FTP)	(signature)
64-bit:	plink.exe	(or by FTP)	(signature)
pageant.exe (an SSH authentication agent for PuTTY, PSCP, PSFTP, and Plink)			
32-bit:	pageant.exe	(or by FTP)	(signature)
64-bit:	pageant.exe	(or by FTP)	(signature)
puttygen.exe (a RSA and DSA key generation utility)			
32-bit:	puttygen.exe	(or by FTP)	(signature)
64-bit:	puttygen.exe	(or by FTP)	(signature)

Download PuTTYgen

After download PuTTY, install it on the computer.

As for PuTTYgen, no need to install PuTTYgen. It will operate when you just double click on it.

You can also watch this video, as you see below, that explains how to download PuTTY and PuTTYgen:

Setting up SSH Key

The PuTTY program does not support key file with the extension of **.pem**. So we need to set up a private key for PuTTY.

> Tips:
> - You can also watch the video including "Setting up SSH Key" at the end of the next page.

The name of the tool that we will use in this process is the PuTTYgen that we downloaded in the previous lesson.

First, let's open the **PuTTYgen** file.

- Click the **Load** tab.
- Go to the directory where the **.pem** file is located and select the **.pem** file. If you cannot see the file in the directory, you can see it after selecting **All Files**.
- Click **Open** and then the message window will informe us that we have successfully imported. Click **OK.**
- Then, click the **Save Private Key** tab. You'll be asked if you're sure that you don't want a password. Since we don't want to enter a password, let's continue by saying **Yes**.
- Save the file to the same folder with the same name, but this time with the extension **.ppk**.

Now our key file is ready to use over PuTTY.

Connecting to EC2 Instance on Windows via PuTTY

- First, let's open PuTTY,
- Then, paste our IP address that we copied from the **Instance Description** section to **Hostname or IP Adress** tab

```
ec2-user@ec2-172-31-41-196.eu-west-1.compute.amazonaws.com
```

- Navigate to **Connection > SSH > Auth.** on PuTTY in the left-hand file tree,
- Click **Browse** which is the right-hand side at the bottom,
- Locate the **.ppk private key** file and click **Open.**
- Then, click the **Session** tab at the top of the file tree on the left-hand side and save this session as **AWS** and click **Save**. So, we can reach the machine easily for the next time.
- Finally, click **Open** again to log into the remote server with key pair authentication,
- Once PuTTY is opened it will give us a warning again. Let's continue by saying **Yes**.
- And then, if we see this image on the screen, it means we have succeeded.

```
Using username "ec2-user".
Authenticating with public key "imported-openssh-key"
Last login: Thu Dec 26 10:59:13 2019 from 173.44.55.155

       __ |  __ |  _)
       _  |  (AMI  /  Amazon Linux 2)
      ___ |  \ ___ |  ___ |

https://aws.amazon.com/amazon-linux-2/
[ec2-user @ ip-172-31-41-196 ~] $
```

Great! We are now able to connect to our virtual machine using PuTTY over Windows.

Installing Updates to EC2 Instance

We'll go to connect the EC2 instance and try to update it.

We first access the key (.pem) file, connect instance and take root privileges with the following commands

```
cd desktop/KEY

ssh -i key.pem ec2-user@34.254.185.28

[ec2-user@ip-172-31-41-196 ~]$ sudo su
```

> Tips:
> - cd desktop/KEY is the file path of key.pem file. It is located on Desktop of my computer for this scenario . But, your's may be located on C:/ or somewhere else. Be sure to write here your file path correctly

💡Tips:

- key.pem is the name of key file. While creating our instance we determine this name. So it can be different for everybody. Be sure to write here your file name correctly

And you will see the script on Bash monitor :

```
[root@ip-172-31-41-196 ec2-user]#
```

And then try to install updates via the following command:

```
[root@ip-172-31-41-196 ec2-user]# yum update
```

But you'll see on Bash monitor :

```
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
No packages marked for update
```

It means no need to install any update. Because when creating this instance, we entered a script in **Advanced Details** section like below.

```
#!/bin/bash
yum update -y
```

Thanks to these commands, this virtual machine updates automatically when you operate this machine first. So we could not find anything to update.

Installing Nginx to EC2 Instance

Now let's install a web server to our instance. The name of the software we choose to install is Nginx, which has become very popular web server in recent years.

⚠Avoid ! :

- As we have shown in the previous lessons, we consider you to conncet to the instance and get the "root " privilege via "sudo su" command

- First, write the command seen below and install software on our machine.

```
amazon-linux-extras install nginx1.12
```

The software will then be installed on my virtual machine.

Let's begin to upload the software. So, write the initial-start command.

```
service nginx start
```

💡Tips:

- This command (service nginx start) sometimes gives the following error.
- Redirecting to / bin / systemctl start nginx.service Job for nginx.service failed because the control process exited with error code. See "systemctl status nginx.service" and "journalctl -xe" for details.
- After writing the following commands for the solution, let's rewrite the start command.
- [root @ 172-31-41-196 ec2-user] # sudo fuser -k 80 / tcp 80 / tcp: 9130 9131.
- [root @ ip-172-31-41-196 ec2-user] # service nginx start
- Redirecting to / bin / systemctl start nginx.service
- [root @ ip-172-31-41-196 ec2-user] #

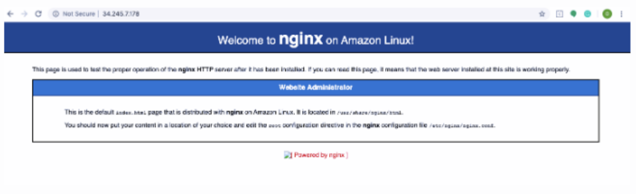
Nginx was successfully installed. Let's write the following command so that the Nginx service can run automatically.

```
chkconfig nginx on
```

But, is our Nginx installed properly? There's an easy way to check this:

- Copy the Public IP of the server,
- Then open a blank page in the browser and paste it,
- After press enter.

  If your page is seen as below, you've done it.



Not Secure | 34.245.7.178

Welcome to **nginx** on Amazon Linux!

This page is used to test the proper operation of the **nginx** HTTP server after it has been installed. If you can read this page, it means that the web server installed at this site is working properly.

Website Administrator

This is the default `index.html` page that is distributed with **nginx** on Amazon Linux. It is located in `/usr/share/nginx/html`.

You should now put your content in a location of your choice and edit the `root` configuration directive in the **nginx** configuration file `/etc/nginx/nginx.conf`.

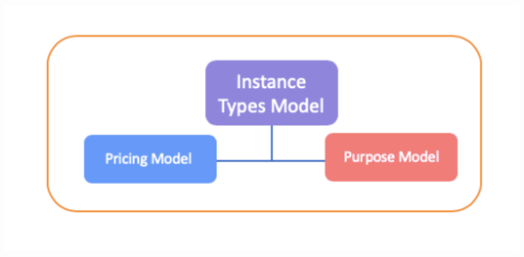
[Powered by nginx]

Nginx Web Page on Working

Congratulations you have successfully installed your first software to the instance

AWS Instances

Types of Instances



Types of Instances

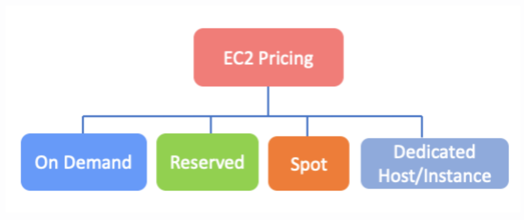
Amazon has come up with a wide range of Instances that are designed to fulfill the wide variety of needs of an organization. Thus, Amazon serves different configurations of CPU, memory, storage, and networking resources to suit user needs.

In AWS, there are 275 kinds of instances for virtually every business need.

Instance types are grouped into a variety of families based on target application profiles and pricing options. It is possible to categorize EC2 types under two main perspective:

These are **Pricing Model** and **Purpose Model**.

Pricing Model



Pricing Model

Pricing Model :

There are four types of instance in the pricing model as you see in the picture above.

On-Demand

On-Demand (Pay as you go, no commitment, price is fixed and pre-determined):

In this choice, EC2 virtual machines are available on request. Although some instances have Per Second Billing features, each virtual machine model has an hourly price. At the end of each month, the more hours AWS has run this virtual machine, the more your credit card charges.

For example, a T2 micro model virtual machine costs $ 0.02 per hour. If we run this instance for 30 hours a month, AWS will issue us a $ 0.6 bill at the end of the month. Or if you think you've been working all month, there's a bill of $ 15.0.

This method is used for usually short-term needs, especially testing and temporary needs.

Reserved Instance (RI)

Reserved Instance

Reserved Instance (RI) (1 or 3 years commitment, 30%-75% price advantage, Partial upfront or All upfront, capacity is reserved):

Reserved Instance is a tariff that takes advantage of the discounted price by giving AWS a 1 or 3-year commitment.

For example; You have a website and these servers will serve for at least 1 year. You can buy Reserve Instance from AWS and pay in full or in part. So you get a 30-% 75% discount. In addition, AWS assures you that you can always access this type of server during this time.

There is also the **Scheduled Reserved Instance** model derived from RI. This model is very similar to the Reserved Instance and provides you to make the purchase over 24 hours. For example, we have an application that works only between 10:00 and 19:00 in the morning. In advance, we could buy a 24-hour RI but, thanks to the Scheduled Reserved Instance, you can buy an instance only between these hours now.

Spot Instance

Spot Instance

Spot Instance (Cost advantage up to 90%)

Almost all types of servers in EC2 have an On-Demand price, which is a fixed price, and there is a spot price that changes according to this idle capacity and changes in seconds. Just like the stock market.

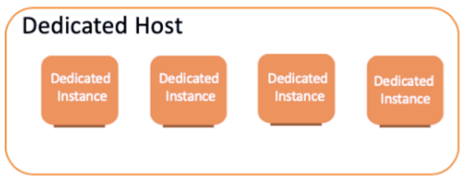
While everyone is creating and running virtual machines at daytime pick hours, the spot price and on-demand price are the same, but at midnight the spot price seems to fall to one-fifth of the on-demand price. You can save up to 90% cost advantage in this way.

Here, you can enter a purchase order by setting a target price. The machine **runs when the price falls below the target price**.

The only **downside** is that the machine automatically **shuts down if the price exceeds that target price**.

So you may prefer to use it for **non-continuity jobs** such as testing.

Dedicated Host



Dedicated Host&Dedicated Instance

Dedicated Host

A Dedicated Host is a **physical server** the whole capacity of with EC2 instance is dedicated to your use. This can help you to reduce costs by using your existing server-bound software licenses.

A Dedicated Host consists of Dedicated Instance capacities according to your needs. You may choose to buy a Dedicated Host or only one Dedicated Instance also.

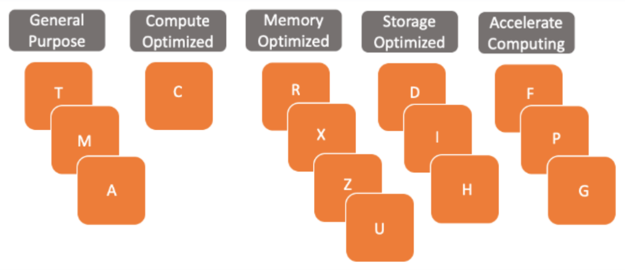
In this option, not only your instances are reserved but also they physically separated from the other servers.

In some cases, using a dedicated host may be a legal necessity or a requirement for application licenses. And also regulations may require that customer information relating to certain special issues to be stored in this way.

For example, Oracle requires the use of a dedicated host in order to use some programs on AWS so that they can track and calculate the license usage properly.

Dedicated Host instances are useful when application license is calculated according to the number of processers. It can be purchased as a Reservation for up to 80% off On-Demand price.

Purpose Model



Purpose Model

AWS offers 14 different types of virtual machines in 5 categories, which are customized for almost every type of need.

Purpose Types-1

General Purpose:

It is used in situations requiring workloads such as WEB servers, microservices, cache fleets, distributed data stores, and development environments.

There are T, M and A options that we can use for standard and application needs. This is the most commonly used server type.

Compute Optimized:

Preferred when you need High-performance web servers scientific modeling, batch processing, high computing, machine learning, and multiplayer coding.

There are C-type servers that require CPU power and are optimized for this.

Purpose Types-2

Memory Optimized:

It is used in situations requiring a high-performance database, real-time large data analytics, and high memory usage.

There are R, X, Z and U type instances in this category

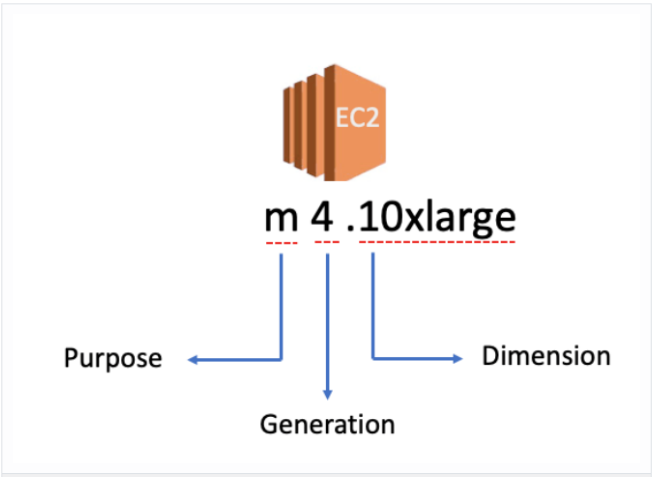
Storage Optimized:

There are D, H and I type of instances in this category. It is the best used for the fast disk structures we need in NoSQL databases or data warehouse solutions.

Accelerated Computing:

It is preferred when you need machine learning, deep learning calculation, fluid dynamics, and analysis.

Code Definition of Instance Type



Code Definition of Instance Type

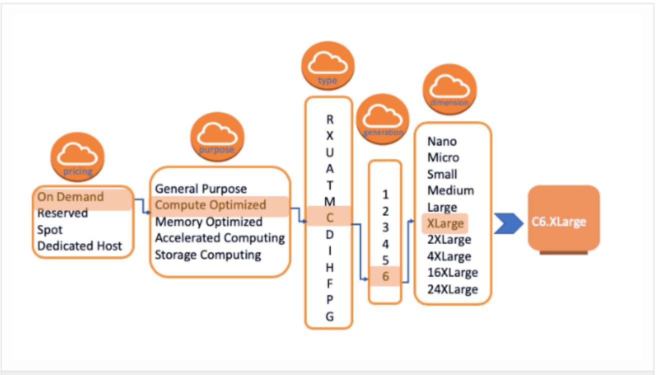
In the figure above you can see EC2 instance. If you pay attention to this model, you 'll see the first letter **m** which stands for its purpose. It means this EC2 is General Purpose instance.

And then, the number of **4** comes which indicates instance generations. For example, the last generation of the M family is m5.

And the last, **10xlarge** shows the dimension of instance. AWS has built servers of various sizes to suit every need in server families. For example, the m5 family has 6 different sizes starting from large to 24xlarge.

The differences provide different RAM, CPU, storage, and bandwidth. We choose one of these models according to our own needs.

Choosing an Instance



Process of Choosing an Instance

When we want to create a virtual server for ourselves, we can follow these steps seen in the picture above;

- First, we need to choose a pricing method and then find your correct purpose family that suits our job.
- We'll find the right model for our job.
- Then we decide which generation to use.
- We determine what size virtual machine we need.

So we have chosen a virtual machine that suits our needs.

⚠Avoid ! :
- It is just a modeling notation shown in the picture above. Not all models may have instances in every generation and size.