**EC2 – Elastic Cloud Computing ( class-7 )**

**Terminologies that are used in EC2 context**

**Latency and Bandwidth:**

**Network Factor that has Latency and Bandwidth are** like optical fiber cables, network equipment, ISP infrastructure quality, and distance.

**Latency :**  refers to the amount of time(millisec) it takes for a data packet to travel from the source to the destination and back (i.e., **time(millisec) taken for a request to reach the server and for the server's response to return to the client**.).

* Lower latency means faster responses
* Higher latency causes delays, buffering, or lag.
* **Longer distances result in higher delays.**

**How to Reduce Latency:**

1. Fiber optic cables provide lower latency than satellite connections.
2. Use Content Delivery Networks (CDNs) to serve content from servers closer to the user.
3. Use faster storage, processors, and network equipment.

**Bandwidth** refers to the maximum amount of data that can be transmitted over a network in a given amount of time. It is measured in **bits per second (bps)** or its multiples like **Mbps (megabits per second)** and **Gbps (gigabits per second).**

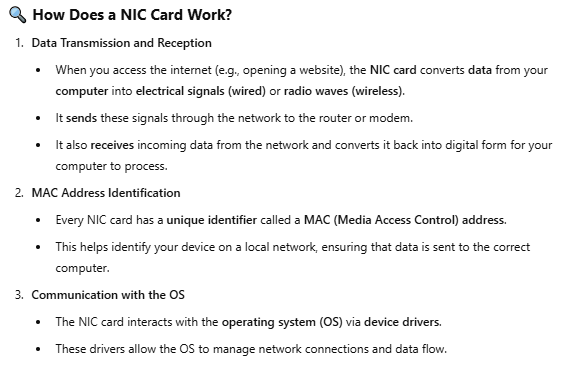
* A higher bandwidth means more data can flow through the connection simultaneously.
* Bandwidth can be shared among devices on a network. If multiple devices are using the same connection, they compete for bandwidth.

**Latency and Bandwidth are closely related in terms of their effect on network performance**

* Bandwidth determines how much data can be sent at once.
* Latency determines how quickly the data is delivered.
  + High Bandwidth + Low Latency = Optimal Performance
  + Low Bandwidth + High Latency = slow performance, buffering, and lag.
  + High Bandwidth Doesn't Fix High Latency
    - **For example**, a satellite internet connection may have high bandwidth but high latency due to the long-distance data must travel to the satellite and back.

**NIC (Network Interface Card)** is a hardware component that connects a computer (or any device) to a network, allowing it to **send and receive data**. It is also called a **network adapter** or **network controller**.

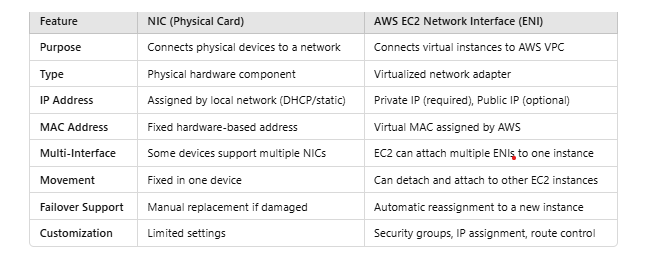
NIC cards can be **wired (Ethernet)** or **wireless (Wi-Fi)**



**Similar to NIC card EC2 has Network interface**

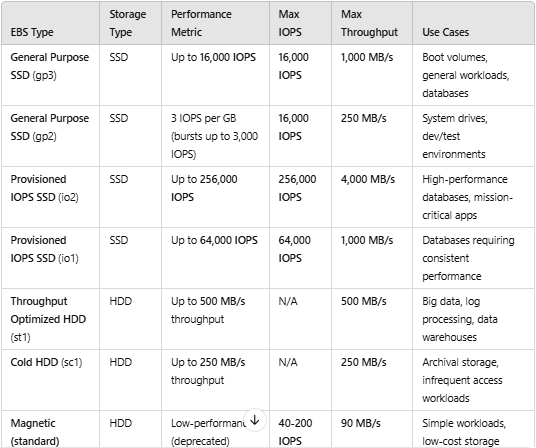
- In AWS, a **network interface** is called an **Elastic Network Interface (ENI)**. It allows an EC2 instance to connect to a Virtual Private Cloud (**VPC**) and facilitates communication over the network.

- It is a **virtualized version** of a **NIC card** in AWS.

****

**Amazon Elastic Block Store (EBS)** is a **persistent** block **storage service** designed to be used with **EC2 instances**. It functions like a **hard drive** for your virtual server, storing **data, applications, and system files**.

When you launch an **EC2 instance**, AWS attaches an **EBS volume** to it, which holds the **operating system** (root volume) and any additional data.



**In General , building a server(can be File server/web/application/game) requires:**

* **Hardware Components -** CPU, RAM, Motherboard, Storage device, Power Supply

Unit, NIC card, Cooling System

* **Networking -** ISP Router, Switch, Static IP address, Cabling
* **Software Requirements -** OS, Server Software(ex., Apache for webserver, MySQL for

DB server) ,Security tools(like Firewall), Virtualization

tools (like VMware)

* **Infrastructure -** like Backup power, backup server
* **Configuration and setup -** user permissions, Installing OS & Software
* **Monitoring Tools**

**Virtualization:**

* VMware is a company that provides **virtualization software** and services. Its tools allow you to create and run **virtual machines (VMs -** act like real computers, each with its own OS, applications, and settings**)**  on a single physical computer or server.
* Instead of running many physical servers, you can run multiple virtual servers on a single physical server. This saves hardware costs and energy.
* VMware software manages resources efficiently, allocating CPU, RAM, and storage to each VM based on its needs
* VMware software creates a **virtual layer** on top of physical hardware. This layer allows multiple **operating systems (OS)** or **applications** to run on the same hardware independently. It achieves this by **virtualizing** the hardware resources like CPU, memory, storage, and networking.
* Each VM is isolated, so issues in one VM don’t affect others.
* VMware Used in – Data Centres, Cloud Computing, Development and Testing.

**VMware uses technology like vSphere HA:**

* In general, If a hardware host (physical server) fails, VMware HA automatically restarts the Virtual Machines (VMs) running on that host onto other available hosts in the cluster.
* **Downtime:** There may be a small delay because the VMs need to be restarted on the new host.
* Requires sufficient shared resources and human oversight.

Here to overcome this AWS is providing EC2 instance

* VMware works well for organizations managing **on-premises** infrastructure
* AWS EC2 provides a **more robust, automated, and scalable solution** for managing hardware failures compared to VMware.
* EC2 eliminates the complexities of hardware management, reduces downtime, and scales globally with minimal user intervention.

**EC2 Instance**

Amazon EC2 (**Elastic Compute Cloud**) is a web service provided by Amazon Web Services (AWS) that allows you to launch and manage **virtual servers (instances)** in the AWS cloud. It provides scalable computing capacity (You can start small and scale up or down based on your application needs. This elasticity ensures you only pay for what you use.), enabling you to run applications without investing in physical hardware.

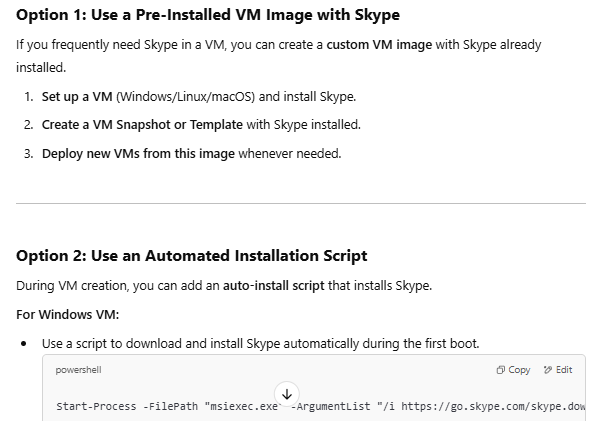
**OS Image / Amazon Machine Image**

In context of Virtualization, we install and launch VMware software in physical server then we follow configuration steps ( like Choosing and installing the type of OS, Assigning **CPU cores** 2 vCPUs and **RAM** 4GB RAM for the VM, Configuring Virtual Hard Disk, selecting network type Bridged / NAT/ Host-only etc) to create single/multiple virtual machines in single physical server.

Example:

If we require to create 50 VMs with software’s like Skype, Mozilla, VPN etc

1. instead of **manually** installing each software in each 50 VMs
2. we create a single VM in physical server and manually install & configure Skype, Mozilla, VPN and Firewall = ON in that VM and then we took a backup of this VM make it as an **ISO** **Image** which act as a **download source** just like floppy’s & CD’s



1. Then we create and run 50 VMs using custom VM image with software’s already installed

**Creating EC2 Instance:** EC2 Instance is a virtual server in AWS cloud.

1. Open EC2 dashboard in AWS. Remember - EC2 instance is region based.
   1. **RULE in AWS :** In **AWS**, every **EC2 instance** must be created within a **VPC (Virtual Private Cloud)** because AWS no longer allows instances to be launched outside of a VPC.
   2. **If you want to create EC2 outside VPC** then AWS automatically provides a **default VPC** in every region with public subnets, making it easy to launch instances without creating a custom VPC network.
2. Click on Instances on the EC2 dashboard menu
3. Click on button Launch instances
   1. Name and Tag – helpful in finding specific instances from multiple instances
      * We give a name to the instance. **e.g.** My Web Sever
      * A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value
      * Click on **Add new tag** button to add more tags, like
        + Key = Name , Value = My Web Server
        + Key = Dev- type , Value = Production
        + Key = OS-Type, Value = Windows
   2. At Quick start you can select an AMI. AWS provides selected AMIs details like **description**, **Architecture**, **AMI** **ID**

An AMI is a template that contains the software configuration (operating system (OS), application server, and applications) required to launch your instance.

* + - At Browse more AMIs - you find more AMIs from AWS, from Marketplace, from community.
    - From Amazon Machine Image (AMI), select an AMI that is marked Free Tier eligible.

**AWS uses** **two main virtualization types** **for Amazon EC2 instances:**

HVM (Hardware Virtual Machine) virtualization uses **full hardware emulation**. It means the AMI operating system runs directly on **virtual hardware**, similar to how it would on a physical machine.

PV(Paravirtualization) uses **software-assisted** virtualization. It modifies the AMI OS to work more efficiently in a virtual environment without hardware-level virtualization support.

* 1. Instance Type – It’s a combination of CPU and RAM
     + Select one type **e.g.** t2.micro – it’s a free tier eligible
     + At Compare Instance types - we find a wide range of instance types optimized to fit different use cases.

**NOTE:** As AWS has removed **instance store(temp storage)** option from below **Configure storage**. However, if your instance requires **Instance Volume**(temporary storage i.e., Data will be lost when instance get stopped/terminated) then you **should** specify the instance type which supports **Instance Volume** at this level **Instance Type.**

**To select Instance Type which support Instance Storage/volume:**

* **Go to Compare Instance Types , all true options under Local Instance storage option do support instance store volumes.**
* If **Local Instance storage** option is not available then click on settings wheel button, there you find it and click on enable.
* **Instance store volumes** will be shown at  **Configure storage** option in EC2 template9**.** Click on **Show details** to display selected **Instance storage** associated with selected **Instance type** 
  1. **Key** **Pair**(**login**) – we can use existing or create new **key pair**(which is **username** and **password**) to securely connect to your instance. Ensure that you have access to the selected existing key pair before you launch the instance.
     + **Click on Create new key pair** – a window will open
       - **Key pair name –** provide any name as you prefer
       - **Key pair type –** RSA
       - Click on **Create key pair** button – To download it in secure drive location
     + NOTE : When you launch an **EC2 instance**, AWS provides a **default username** depending on the **AMI (Amazon Machine Image)** you choose.

**EX :** Amazon Linux – ec2-user, Ubuntu – ubuntu, Centos – centos, Windows – Administrator, Custom AMI – as you prefer

* 1. **Network** **Settings** – Adds instance to VPC

Implicitly an AWS default VPC has been selected

* Click on Edit button – to select our VPC
* **VPC** **– required** - Select your VPC that you want to launch your instance into
* **Subnet**

select public subnet if instance needs to expose to world

select private subnet if instance doesn’t need to expose.

* **Enable** **Auto-assign public IP** **only** if you launch instance in public subnet – then automatically AWS assigns a public IP to instance.
* **Firewall (security groups)** - A security group is a set of firewall rules that control the traffic for your instance.

You can assign one or more security groups to your instance, if do then all the rules are evaluated to control inbound and outbound traffic.

* Select **Create security group** button – To allow N/W traffic

**Security group name –** provide any name to your instance SG (e.g., Indraja)

**Description –** e.g., Indraja created at feb 9 2025

**Inbound Security Group Rules -**

**Security group rule 1** - According to selected **AMI OS** the network traffic **type** and its **protocol** and **port range** are **automatically filled in by AWS** and **Source type** says who should connect to EC2 instance(e.g. MY IP).

* Click on **Add Security Group Rule** button

**Security group rule 2** - Let say anyone could connect to Web App that is hosted in EC2 instance on port 80.

**Type –** HTTP

**Protocol and port range –** auto filled by AWS based on **Type** mentioned.

**Source Type –** select **Anywhere** to allow everyone to connect(i.e., **Source** will be **0.0.0.0/0**)

**NOTE:**

* The security group is attached to the **Elastic** **Network Interface** (which connects your instance to the VPC.) of the instance.
* AWS applies the security group **at the network level**, filtering traffic **before it reaches the instance**.

**Rule-1:** By default, request from any sources 0.0.0.0/0

Will be denied by security group

To allow request from specific IP source then

You must manually define an **Inbound/Ingress rule**

**Rule-2**: By default, a**ll outbound/Egress traffic is allowed**

(0.0.0.0/0 for all ports and protocols).

To **restrict outbound** traffic, you must manually

define rules

Example : 1. **Even** i**f you block all inbound traffic** then **by**

d**efault**, instances can initiate outbound

connections unless specifically restricted.

2. **Even allowed inbound requests cannot respond**

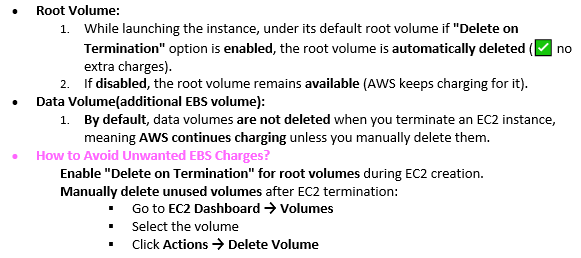
**back** if you restrict outbound traffic

* **3 ways to attach security groups to instance.**
  + **SG to each instance**
  + **Multiple SG(Max 5) to each Instance**
  + **SG to multiple instance when all has same s/w & N/w configs**
* Similarto **NIC card, AWS EC2** has **Network interface**

AWS **automatically** adds a **primary network interface** to every **EC2 instance** when you launch it.

**To edit the added network interface, click** on **Advanced network configuration** option which opens **Network interface 1** template.

* + - * + In AWS, a **network interface** is called an **Elastic Network Interface (ENI)**. It allows an EC2 instance to connect to a Virtual Private Cloud (**VPC**) and facilitates communication over the network.
        + It is a **virtualized version** of a **NIC card** in AWS.
        + **Primary ENI**: Every EC2 instance has one main ENI (default adapter).
        + **Secondary ENIs**: Based on hardware you can add extra ENIs for advanced use cases.
  1. Configure storage
     + In **AWS EC2**, the default **AMI root volume** refers to the primary storage volume that is created from the **Amazon Machine Image (AMI)** when you launch an **EC2 instance**.
     + **Root vs. Additional Volumes(EBS)** in EC2:
       - Root Volume(i.e., C-drive):
         * Contains the OS (e.g., Linux, Windows) and any **pre-installed software** defined in the AMI.
         * Default EBS volume created when you launch an instance. It can be gp3 or io2 type.
       - Additional Volumes(i.e., D- drive):
         * Attach extra volumes for data storage.
         * Can be detached and re-attached to other instances.



* + - **Click on Advance** and **Add new volume** button then **it allows you to create, customize and allocate** **the storage** (EBS volumes in AWS) to the instance.
    - A **Custom Volume** template will open, there you see the options like **volume size**, **volume type**, **IOPS** , **delete on termination**(option **Yes** – deletes storage when you terminate the instance). **Custom volume** used to store **data**, **application**, **System files etc. Select** volume **size** and **type.**

**NOTE:** In AWS EC2, Volumes are divided into two types –

1. **Instance Volumes** –

* **Ephemeral (Temporary)** storage attached to EC2 instances
* Data is **lost when the instance stops or terminates. (Data won’t be lost when restart.)**
* **Faster than EBS** as it tied to the instance.
* **NOTE**: AWS has removed **instance store/instance volume** option available at **Configure** **storage**. However, if your instance requires **Instance Volume** then you should specify the **instance type** which supports **Instance Volume** at above mentioned level **Instance** **Type**.

1. **Elastic Block Store (EBS)** – chargeable in AWS

* Permanent storage that remains intact even if the instance stops or restarts.
* Can attach/detach from instances
* More IOPS = High speed

**Input/output Operations Per Second**— The requested number of I/O operations per second that the volume can support.

* Used for Backup, Data Movement, Encryption.

**NOTE:**

**According to our requirement and use case capacity of EBS volumes we select Volume to our instance.**

<https://docs.aws.amazon.com/ebs/latest/userguide/ebs-volume-types.html> - AWS EBS Documentation

<https://uptime.is/> - Uptime and downtime SLA calculator site

**Performs 3 IOPS per GB for GP SSD (gp2)** : **IOPS = EBS size \* 3** but its min & max IOPS are 100 / 3000

* 1. **Advanced** **details**
     + **Shutdown behaviour** - The instance behaviour when an OS-level shutdown is performed.
       - **Stop** - Shuts down the instance temporarily.
       - **Terminate** - Permanently deletes the instance.
     + **Stop – Hibernate behaviour** - Hibernation stops your instance and saves the contents of the instance’s RAM to the root volume.
     + **Termination protection** – If enabled, the instance can't be terminated using the console, API, or CLI until **termination protection** is disabled.
     + **Stop protection** – Enabling this option can protect instances from being accidentally stopped. you won't be able to stop this until **stop protection has been disabled.**
     + **Placement group -** The cluster group in which to launch the instance.

Supported on certain instance types only; specifying an incompatible instance type **will fail the instance launch**.

Assume, Houses -> Ec2 instances, Water **🡪 data** being sent and received by your EC2 instances, **Water Pipes → Network Bandwidth, Water Tanks → Physical Racks, Farm → Availability Zone (AZ).**

* + - * Cluster placement group –

**Placement: All instances are placed in a single rack within** **one AZ**.

Think of this as building **all houses close together(inside an Availability zone)** near a **central water tank(must be on** same rack)**.**

If one house sends water to another, it travels **short distances** and **moves faster** because they’re all **close together**. (low network latency, high bandwidth).

* + - * Spread Placement Group –

**Placement**: Instances are **spread across multiple racks** in **different AZs** (maximum **7 instances per AZ**).

For example, in a Region with three Availability Zones, you can run a total of 21 instances in the group, with seven instances in each Availability Zone.

 If you try to start an eighth instance in the same Availability Zone and in the same spread placement group, the instance will not launch.

If you need more than seven instances in an Availability Zone, we recommend that you use multiple spread placement groups.

Here, you build **each house far apart** on the farm, with **separate water tanks** for each.

If one water tank (rack) has an issue, others won’t be affected

**High availability**, but **slightly slower** communication due to the distance.

* + - * Partition Placement Group –

Placement - **Split across 7 partitions** in **each AZ**. Each **partition** gets its **own rack**. Amazon EC2 tries but doesn’t guarantee to evenly distribute the instances across all partitions

Group of instances in one partition do not share the underlying hardware with group of instances in different partitions.

**Data Transfer**: **Fast** within a partition, **slower** across partitions.

* + - **EBS-optimized instance** - an EC2 instance that provides **dedicated bandwidth** between the EC2 instance and its attached **Elastic Block Store (EBS)** volumes. This keeps **network traffic** and **storage traffic** **separate**, improving performance.

Supported on certain instance types only; specifying an incompatible instance type **will fail the instance launch**.

**Imagine the scenario**:

You have a **house** (EC2 instance) connected to a **water tank** (EBS storage) that supplies water. Normally, both your **household water needs** (EC2 processes) and **garden irrigation** (EBS data transfer) share the **same pipeline** (network connection).

If you turn on too many taps (heavy I/O workloads), **water pressure drops**—causing slowdowns.

AWS installs a **dedicated pipeline** then shower water and garden water **flow independently**.

**Without EBS Optimization:** Shared pipeline = **Slower Performance**

* + - **Metadata accessible – Enabled default which allows you to retrieve your instance mata-data with in your instance using a Link-local address**
    - **User-data** – You can specify up to 16KB user data in form of commands or a command script(bootstrapping script) to run when you launch your instance.

<https://github.com/kumarawsbit/awsbit-1/blob/master/Sample%20Code/powershell_userdata_windows.ps1> - PowerShell user data link

**NOTE :**

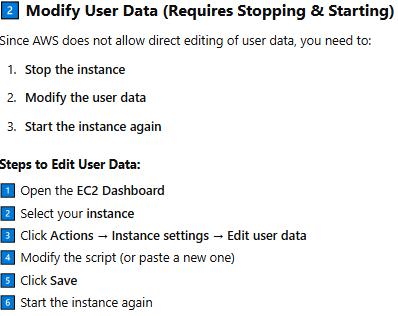
If you use any programming language like java, python then that software must be pre-exist in instance.

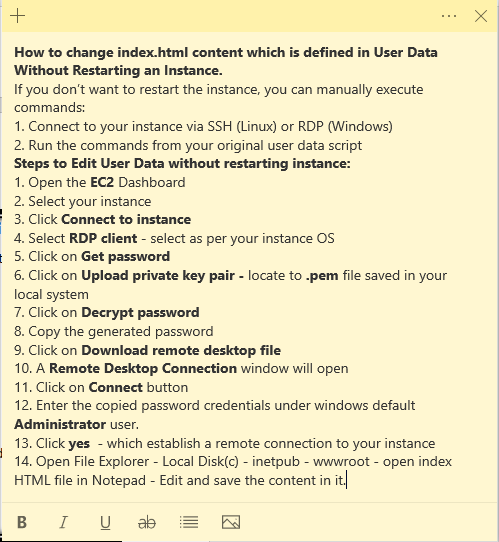
Hence the best way is : For **windows instance** – **power shell** scripting

For **Linux instance** – **shell** scripting

**To host a web-based app on server you typically need a web server to deliver static files (HTML, CSS, JavaScript) to client browser.**

Web servers are – Node.js(Express)/**Apache/NGINX/IIS(for windows)/Tomcat.**

****

****

* 1. **Summary** – Here you can specify the **Number of instances** (i.e., similar instances) to launch.

1. Click on **Launch Instance** button – AWS will add your public instance to your VPC by allocated private and public IP address. After launching an Instance is created with an ID. Click on generated instance ID which takes you to EC2 instances template.

**Here Status Check (2/2) -** verifies whether the instance is working correctly from

both **AWS infrastructure** and **your operating system (OS)**.

**NOTE:**

EC2, EBS, and NIC **and other resource are not necessarily on the same physical rack in data centres**, but they are within the same **Availability Zone** for low latency.

**To view the Instance summary(metadata) after the launch**

* + At EC2 **Instances** console, click on launched **instance ID to view its summary**
  + At Details – you see instance details
  + At Security – you see the allowed incoming traffic to your instance.

**To check whether you can connect to your windows instance**

- use **telnet** command instead of **ping** command

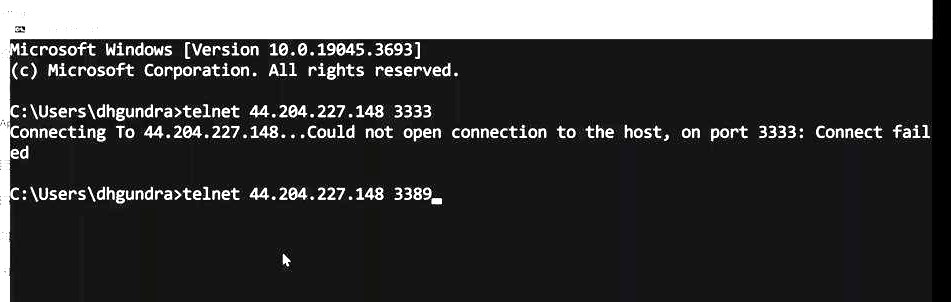
Telnet command - telnet <server address> <port>

e.g., telnet google.com 443 or telnet 142.250.183.174 443

Ping Command - ping [options] <target>

e.g., ping google.com

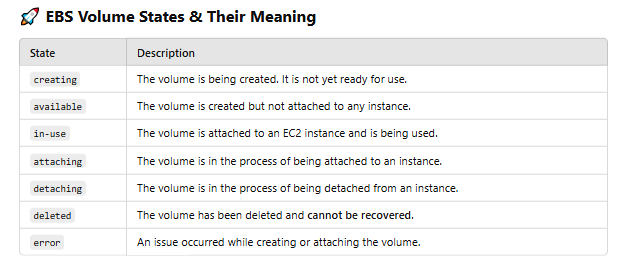
* If telnet is not installed on your windows :
  + Open **Control Panel** → **Programs** → **Turn Windows features on or off**.
  + select **Telnet Client** and then click on **OK**
* **To check instance connectivity through Command prompt** - At launched instance summary, copy your instance public IPv4 address and open command prompt



* **To connect with your window’s instance through AWS EC2 console**
  + Open instances at EC2 dashboard
  + Click on Instance ID
  + Click on Connect button
  + Click on RDP client – If your instance is windows, RDP is the default windows protocol
  + AWS provide the **Public DNS** name to your Instance – You can use this DNS name to access the machine (e.g., for rdp, web applications).
  + Click on Get Password – To get the Key pair login credential that you have download
  + Click on Upload private key file – select the .pem file
  + Click on Decrypt password button
  + Copy the generated Password
  + Click on Download remote desktop file button and save it
  + A Remote Desktop connection window will open
  + Click on connect button
  + Enter the password that you have copied earlier
  + Click on OK and then Yes – Establishes a connection with your Windows Instance
* How to manually install web server or software in your instance after connection established.

NOTE : **Server Manager** is a built-in **Windows Server** tool that allows system administrators to manage both **local and remote servers** from a **single dashboard**.

* + Search for **Server Manager on your instance** to install **IIS** windows web server
  + NOTE : Open chrome in your instance to install, download and run other software.
  + A window will open, click on **Add Roles and Features**. Click on **Next** button
  + Click on **Server Roles** option
  + Select **Web server(IIS)** and Click on **Next** and **Add Features** button
  + Keep clicking on **Next** and finally click on **Install** button
  + Complete the installation process and restart if required.
  + All **Core IIS Components** will be installed along with a small “Windows Server” HTML program open on local host, port 80 (check using <http://localhost> or using the AWS **Public DNS name** of the instance http://<your-ec2-public-DNS> )
    - **Web Server (IIS)** – The main component that hosts and serves websites.
    - HTTP and HTTPS Protocols – handles the request
    - Default Document - e.g., index.html or default.aspx
    - Static Content - Serves static files (HTML, CSS, JavaScript, images, etc.).
  + To view IIS HTML file - open Local Disk© -> inetpub -> wwwroot
    - Right click on **iisstart** HTML file – which open HTML text in a notepad
* **How to retrieve the instance metadata with in instance (not on AWS console)**
  + Open browser in your instance
  + <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instancedata-data-retrieval.html> - AWS provided a Link-local address (http://169.254.169.254/latest/meta-data/) to access instance metadata properties from within a running instance
* **How to modify or add new Security Group rules to your Instance**
  + Open **Instances** at EC2 dashboard
  + Click on **Instance ID**
  + Click on Security
  + Click on **Security ID**
  + Click on Inbound and click on Edit Inbound Rules button
  + An Inbound rules template will open and at **Source** input box change the IP address to Custom/Anywhere/My IP
  + By clicking on Add rule button – you can add new security rule to your instance
  + Click on Save rule button
* How to **attach and delete a volume** to an instance:



Remember, **EBS volumes in the "available" state** are still chargeable in AWS. Even if they are not attached to an EC2 instance, you are charged based on the **allocated storage size per GB per month**.

AWS **charges for "in-use" EBS volumes** based on the **storage size per GB per month**, regardless of whether the attached EC2 instance is running or stopped.

* + Open Volumes template under EC2 EBS dashboard
  + Attach volume:
    - Select the volume – Click on Actions – select Attach volume
    - A template will open, at Instance – specify the instance ID
    - Click on Attach volume
  + Delete volume:
    - Select the volume – Click on Actions – select Delete Volume
    - Confirm deletion and click on delete button – delete the volume permanently

and no longer volume can be attached to instance

**NOTE : Until you detach an "in-use" volume,** you **cannot attach** to another EC2 instance **while it is attached to one**. EBS volumes work like **physical hard drives** and can only be attached to **one EC2 at a time**.

But some high-performance EBS volumes (**io1/io2**) support multi-attach to multiple EC2 instances in the same **AZ**.

Steps to Attach an ” In-Use” Volume to Another Instance:

1. **Detach the volume** from the current instance:

Go to **EC2 Dashboard → Volumes**

Select the **in-use** volume **→** Click **Actions → Detach Volume**

1. **Attach the volume to another EC2 instance**:

Go to **EC2 Dashboard → Volumes**

Select detached volume **→** Click **Actions → Attach Volume**

Select the new **EC2 instance →** Choose a device name (e.g., /dev/xvdf)

1. **Mount the volume inside the new instance**: For Windows, go to **Disk**

**Management** and assign a drive letter.

* What does windows instance contain according to the User data provided while launching?

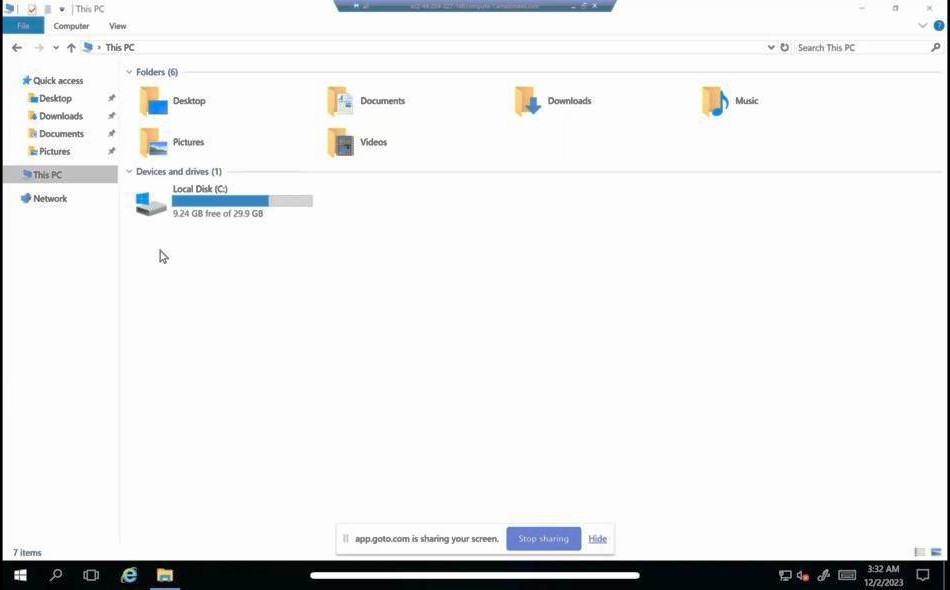


Fig-1 : The file System of your Windows EC2 instance

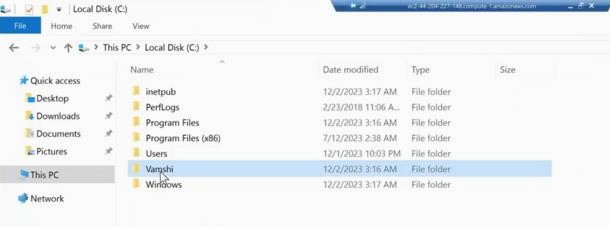


Fig-2:Inside Local Disk© - Vamshi is the folder that we created using **create** **command** in User Data

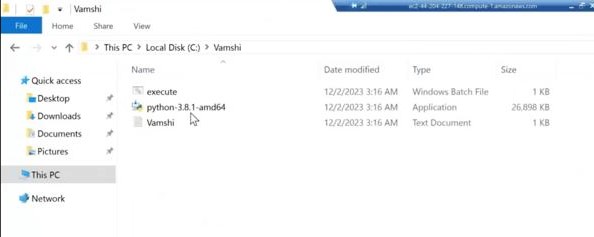


Fig-3: Files inside Vamshi folder

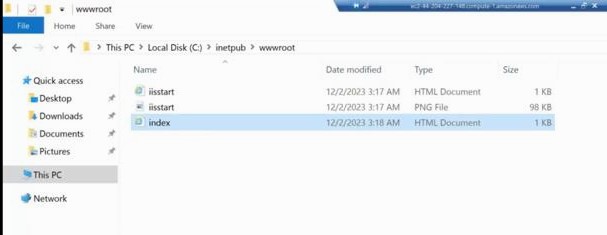


Fig-4 : index.html file that we insisted in User data – file located in C-drive->inetpub->wwwroot

* Even after specifying 2 EBS volumes(30GB & 8GB) to attach to instance we only see

Local Disk© , as you can see in fig-1. This is because of Windows OS management.

Hence, we explicitly need to activate another EBS volume

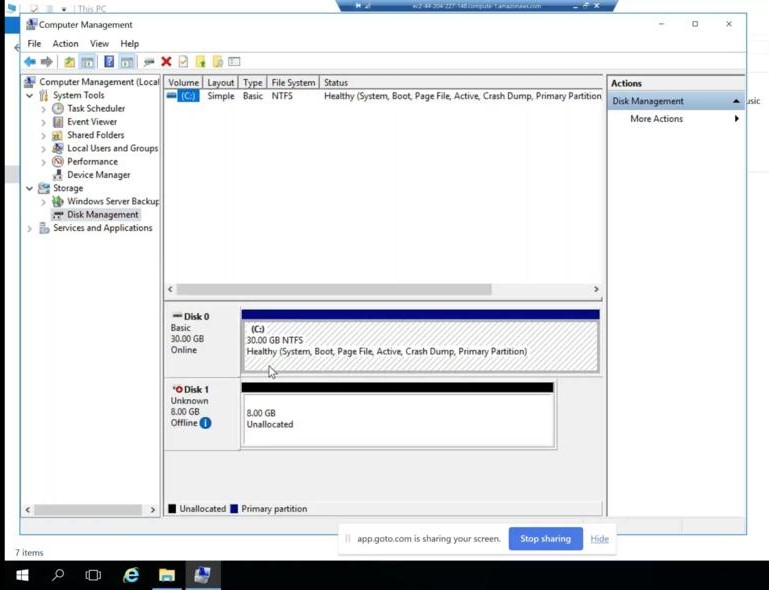


Fig-1: Open Computer management -> Disk management -> Right click on offline volume

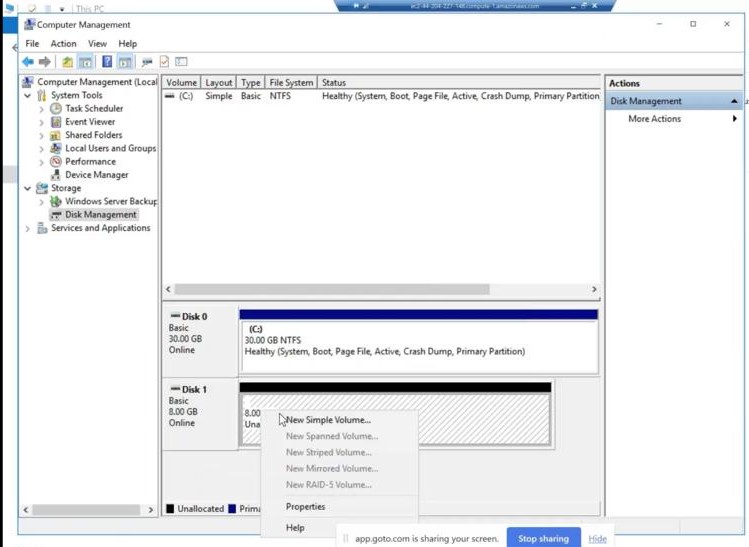


Fig-2,3,4 : After right clicking on volume – click on Online again, right click – click on Initialize Disk

* again, right click – click on New Simple Volume

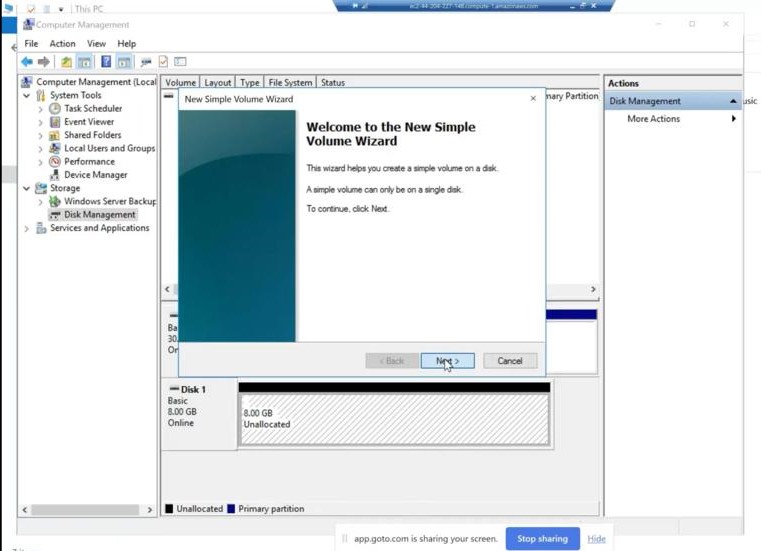


Fig-5&6 – Keep clicking Next button

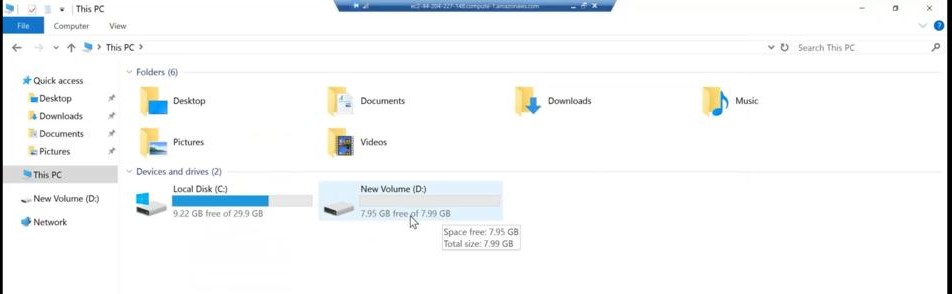


Fig-8 : Finally, an 8GB EBS volume as Disk(D) will be initialized.

AWS provides facility increase or decrease the instance EBS volume:

* Go to EC2 dashboard -> Instances
* Click on Instance ID to which you want to increase or decrease size
* Click on Storage option – you find the Volumes that are attached to instance
* Click on Volume ID to which you want to increase or decrease size
* It takes you to Volumes console
* Check box the instance you want to increase or decrease size
* Click on Action button then click on Modify Volume – a template will open
* Here at Size input box increase or decrease the size
* Click on Modify button
* By this EBS size has increased but as our Instance is windows we must manually Extend the size at instance system

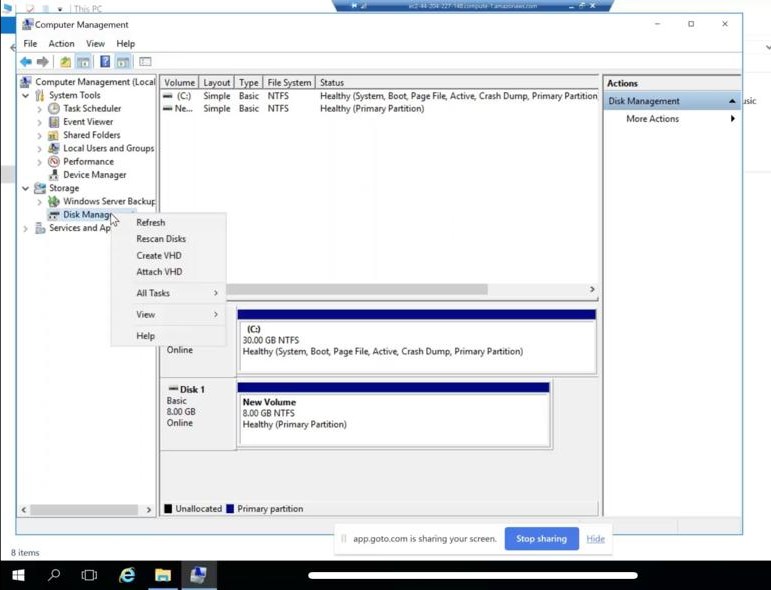
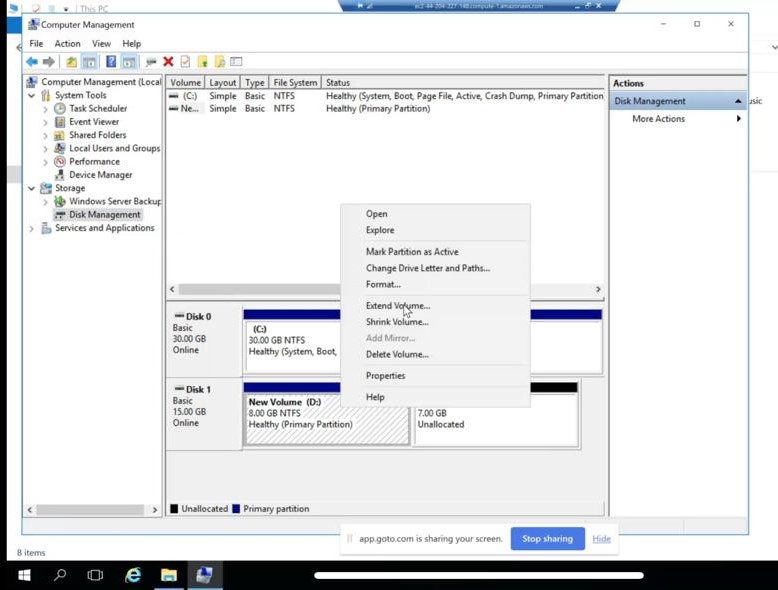


Fig-1: Right Click on New volume(D) -> Click on Refresh (you have extended the 8GB volume of EC2)



* Fig-2: Again, **right click** on New Volume(D) -> Extend Volume -> a window will open where you should **click** Next and Finish button and then refresh the File system to view the extended Disk(D) size from 8GB to 15GB

**To Stop or Terminate the Instance:**

* Go to EC2 dashboard -> Instances
* Check box the instance you want to stop or terminate
* Click on instance State button
* Best learning practice, Select Terminate instance – which completely deletes the instance
* Click on terminate button

**Create Linux EC2 instance –** As Linux is fast as has no GUI it requires very less RAM.

1. Open Instances at EC2 dashboard
2. Click on launch instance button
3. Name and tags
   1. Name – LinuxInstance
   2. Application and OS Images(Amazon Machine Image)
      1. Quick Start – Amazon Linux
      2. Amazon Machine Image(AMI) – select Amazon Linux 2 AMI(HVM)
   3. Instance type – select t2.micro (1 vCPU and 1 GB memory)
   4. Key pair(login) – Create new key pair or select existing key pair
   5. Network settings – Click on Edit button,
      1. Network - Select VPC
      2. Subnet – select subnet (select public subnet if you want internet access to your instance)
      3. Firewall (Security Groups)
         1. select Create security group button or Select existing security group
         2. Security group name - LinuxSecurity
         3. Description – launch-LinuxSecurity-on-13mar25
         4. Inbound Security Group Rules
            1. Security group rule 1 – Aws provide default rule based on your selected AMI image (we selected Linux)

Type – ssh

Protocol – TCP (default Linux protocol)

Port range – 22 (default Linux port)

Source type – select who can have access to Linux system (select MY IP)

Click on Add security group rule – To add more traffic rules

* + - * 1. Security group rule 2

Type – HTTP

Protocol – TCP (default HTTP protocol)

Port range – 80 (default HTTP port)

Source type - select who can have access to Linux system(select Anywhere i.e., 0.0.0.0/0)

* + - 1. Advance network configuration – BY default AWS add Elastic network interface(a virtual NIC card) to every EC2 instance. It allows EC2 to connect with VPC.
  1. Configure storage – Click on Advance – to show more details about storage
     1. Root volume – EBS volume which is automatically selected by AWS
     2. Click on add new volume - to add Additional EBS volume
  2. Advance details
     1. Metadata accessible – Enabled by default
     2. Metadata version – select V1 and V2 (token optional)
     3. Allow tags in metadata – Select Enable
  3. Summary – mention number of same instances you want to create
  4. Click on launch Instance button

**How to connect with Linux EC2 instance from a windows system**

**NOTE:**

We use RDP protocol to connect with windows EC2 instance from a windows system

1. RDP allows you to access and control Windows GUI remotely.
2. RDP provides secure encrypted communication between your local system and the EC2 instance.
3. It allows you to perform administrative tasks (e.g., install software, manage IIS, configure settings) directly from the Windows interface.

**How to Connect to a Windows EC2 Instance Using RDP:**

* + - Go to the AWS EC2 Console → Select your Windows instance → Click Connect.
    - Choose RDP client → Download the RDP file.
    - Click **Get Windows Password** (if it's the first time) and decrypt it using the **.pem** key.
    - Open the **RDP**  file → Enter the **username** (usually Administrator) and

**Password (copy and paste the AWS provided password)** → Click **Connect**.

Similarly, we use **SSH (Secure Shell Protocol)** to establish a remote connection.

1. Install **SSH Client** (if needed) - Windows 10 and later already have **OpenSSH** installed or you can use **XShell** or **PuTTY** (a popular SSH client).
2. In the **AWS EC2 Console**, check that your Linux EC2 **Security Group** allows **inbound traffic on port 22** (SSH default port).

**Step-1**

1. After launching an instance AWS creates one ID to the launched instance
2. Click on that generated ID – which takes you to Instances template in EC2 dashboard
3. Select the instance (our instance name is LinuxInstance)
4. You have 4 ways to connect to Linux instance **– Ec2 Instance connect**, **Session Manager**, **SSH Client**, **EC2 serial console**.
   1. EC2 Instance Connect – a **browser-based client** (Directly access your instance from the AWS Management Console.) to connect to a **Linux** EC2 instance without needing to manage SSH keys(No .pem file required.) or use external SSH clients like **PuTTY** or **OpenSSH**.

**Temporary credentials** (public IP address e.g., 100.24.60.155 and

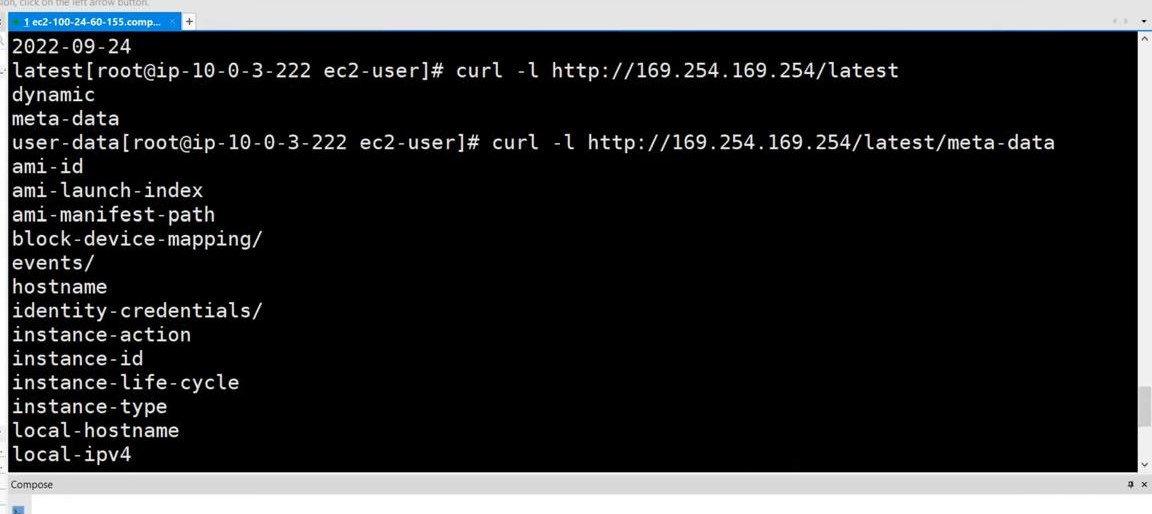
user name e.g., ec2-user for Amazon Linux) are generated by AWS, ensuring

secure and controlled logins.

* 1. Select SSH Client
* Open SSH client (must be installed in your windows) in your local system
* Run the command cd path/to/key\_folder to locate your private key pair file created to your instance while launching (i.e., .pem file)
* Run the command chmod 400 my-key.pem to ensure your key is not publicly viewable.
* Connect to your instance using its public DNS (e.g., ssh -i “my-key.pem” [ec2-user@ec2-100-24-60-155.compute-1.amazonaws.com](mailto:ec2-user@ec2-100-24-60-155.compute-1.amazonaws.com))

**Example :** Set Up **Xshell** to Connect to Linux EC2

* Download **Xshell** in your local windows system
* Open **Xshell**
* A **Sessions** window will open
* Click on **New** session
* At **Connection Settings** :
  + **Name –** Any general name you prefer(you can also use Instance DNS name provided by AWS)
  + **Protocol** : SSH
  + **Host**: copy paste the public DNS of your EC2 instance that AWS provided (e.g., ec2-xx-xx-xx-xx.compute.amazonaws.com)
  + **Port** : 22 (default for SSH)
* **At** Authentication :
  + **User Name : e.g., ec2-user**
  + **Password :** Not required
  + **Method:** Select **Public key** i.e., Key-pair
  + **Click on setup button –** a window will open, locate your instance key-pair file i.e., .**pem** file. Click on **OK** button
  + **Finally click on Connect button.** **Accept and save**
  + **A connection will be established**
  + Clear the Xshell terminal using **cls** command
  + Run **sudo -s** command **–** to havethe **highest level of administrative control** over the operating system.
  + Run **yum install httpd -y** command– installs Apache HTTP server (**httpd**)which is a popular web server that serves web pages over the internet.
    - **Yum** isa package manager used in **Linux** based systems like **Amazon Linux**, **CentOS**, etc. It helps install, update, and remove software.
    - The **-y** flag **automatically answers "yes"** to any prompts during installation.
    - Command to read the instance metadata from the remote instance itself

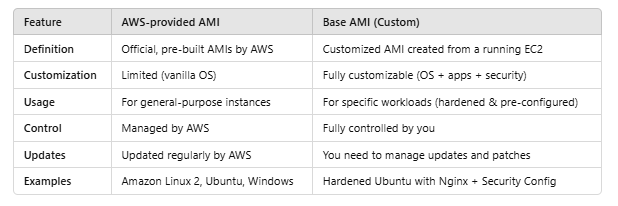


* + - Command # curl -l <http://169.254.169.254/latest/meta-data/placement> - gives the details in which availability zone does your instance is available.

**Creating Base AMI / Golden AMI:**

***We create our own image from an running EC2 instance -*** *to* perform additional security rules on OS or on application then that image is called **Base AMI**.

We can also take backup from our own image as per our scheduled time(i.e., every secs/mins/hrs/days) it helps when accidental EC2 deletion or data corruption happens then we can easily use a backup image and **launch another EC2 from backed up image**.



* A **Base AMI (Amazon Machine Image)** is a foundational image used to launch **EC2 instances**.
* It typically includes a **clean operating system** (e.g., Amazon Linux, Ubuntu, Windows) and minimal essential software.
* You can customize a Base AMI by adding security configurations, software packages, and application settings to create a **golden image** for consistent deployments.

**Steps to create an image(backup AMI) from running instance:**

**Use case :** when you require to take backup of all EC2 EBS volumes then you create an

Image which takes the snapshot of all EBS volumes attached to running

instance. **AWS charges for stored snapshots until you delete them.**

You can see the created snapshot under **Snapshots** templateat EC2 dashboard.

1. Open Instances at EC2 dashboard
2. Select the instance
3. Click on Action button
4. Select Image and templates and then Create Image
5. A **Create image** template will open
   * **Image name** – Enter image name as you prefer (e.g., IndrajaBaseAMI)
   * **Image description** – Indraja base AMI contains software with OS hardened.
   * **No reboot** check box – do not enable

**Generally,** when you create an AMI (backup of your instance), AWS usually **reboots the instance** to ensure a **clean, consistent** snapshot of the instance’s storage.

**With No Reboot Enabled: The instance will NOT reboot** during the AMI creation process. AWS captures the snapshot **while the instance is still running**.

**Use case –** Minimal Downtime, for quick testing without disturbing running services.

Risk of enabling No Reboot – Data Inconsistency

1. **Instance volumes** - When you **create an AMI** from an EC2 instance. **A snapshot** of the **root volume** is automatically created. You can increase its **size.** The snapshots are stored in **Amazon S3** (behind the scenes).
2. Add Volume button – If your EC2 instance has **extra data volumes**, AWS will also create **snapshots** of those volumes **only if** you select the **"Add volume"** option while creating the AMI.
3. Click on create Image button – An AMI with **ID** will be created from the running instance.

If you Click on the newly created **AMI** **ID** which takes you

to AMIs template in EC2 dashboard

Click on AMI ID – Which shows Image summary where

you can see Block devices (a snapshot of instance EBS

volume).

* + - However, by creating an image form a running instance, we don’t actually take backup of an entire EC2 but **we only take the backup of the instance storage** i.e., Local Disk© and Local Disk(D).
    - The created **backup AMI image can be available** only in the region we selected because **AMIs are region based.**

Example, **we can’t use Mumbai regional AMI in Hyderabad region**

**But we can copy the Mumbai backup AMI and use to launch an EC2 instance in Hyderabad region,**

* Open AMIs template under **EC2 Images** dashboard
* Select the backup AMI – click Actions – select copy AMI
* **A template will open and at Destination region** – Select any region (e.g., Hyderabad)
* This way AMI image present in Mumbai & Hyderabad.

**We can also use others AWS account backup AMIs in our AWS Account**

* Open AMIs template under **EC2 Images** dashboard
* Select the backup AMI – click Actions – select Edit AMI permissions
* A template will open and at Ami availability – select Private
* At Shared accounts – click on ADD account ID
* A **share AMI with AWS account** window will open – Specify the Aws account ID with which you want to share the AMI

You can check the shared AMIs at **AMIs** template under EC2 dashboard – Change the dropdown box option **Owned by me** to **Private images**

**How to deregister the AMI -** it means the AMI **can no longer be used** to launch new EC2 instances. However, the **underlying snapshots (EBS volume snapshots) are not deleted automatically you must delete them manually**.

If you need to use the AMI again, you must **create a new AMI** from the snapshots manually.

* Open AMIs template under **EC2 Images** dashboard
* **Select** the backup AMI – click Actions – select Deregister AMI
* Click on Deregister AMI button

**Steps to create EBS snapshot of a running instance**

**Use case :** When you require to take backup of only one EBS volume then best practice is to create snapshot of that EBS volume. In this case we can easily delete the snapshot as it was created by you. **AWS charges for stored snapshots until you delete them.**

1. Open Snapshots template at EC2 EBS dashboard - where you can see one snapshot which is created while creating an image form running instance. This snapshot is the copy of root volume of running instance.

When you create an EBS volume from a snapshot, the new volume begins as an exact replica of the volume that was used to create the snapshot.

EBS Snapshot : You can back up the data on your Amazon EBS volumes by making point-in-time copies, known as **Amazon EBS snapshots**.

A snapshot is an **incremental backup**, which means that AWS save only the blocks on the volume that have changed since the most recent snapshot. This minimizes the time required to create the snapshot and saves on storage costs by not duplicating data

AWS does not automatically back up the data stored on your EBS volumes. For data resiliency and disaster recovery, it is your responsibility to create EBS snapshots on a regular basis

1. The Snapshot which is created by **create image** process, **we cannot delete that snapshot as it is created from running instance**.

Select Snapshot which is created by **CreateImage**

Click on Actions – Select Delete Snapshot - Click on delete button

You can see Failed to delete the snapshot

**Hence to delete the snapshot created by CreateImage first you must deregister the backup AMI**

It means the AMI **can no longer be used** to launch new EC2 instances.

However, the **underlying snapshots (EBS volume snapshots) are not deleted automatically you must delete them manually**.

* Open AMIs template under **EC2 Images** dashboard
* **Select** the backup AMI – click Actions – select Deregister AMI
* Click on Deregister AMI button
* **Now** open Snapshots template under EC2 EBS dashboard
* **Select** Snapshot whose backup AMI is deregistered
* **Click** Actions – Select Delete Snapshot
* **Click**  Delete button – Successfully delete the snapshot

**How to create the snapshot of specific EBS volume of an instance:**

* 1. Open **Instances** at EC2 dashboard
  2. Select Instance ID
  3. Click on Storage
  4. Click on Volume ID which you want to take a snapshot
  5. This will take you to Volumes template under EC2 EBS dashboard
  6. Select Volume ID – click on Action button – click on Create snapshot
  7. Then a **Create snapshot** template will open
     1. Description – e.g., snapshot of additional EBS volume of IndrajaInstance
     2. Click on create snapshot button

**In AWS, EC2 is an Infrastructure as a service.**

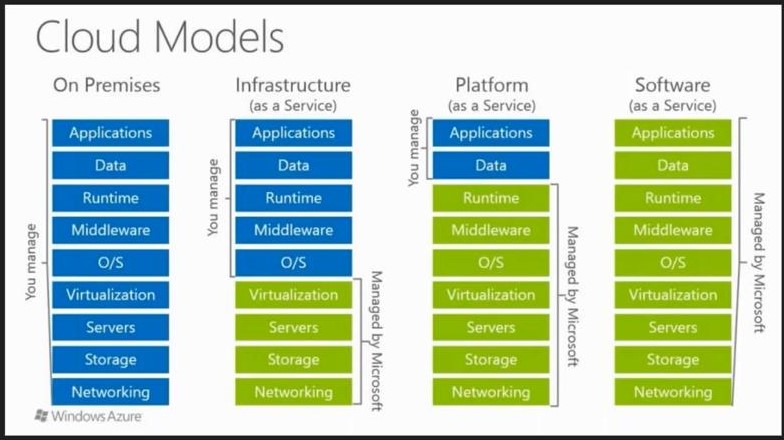
****

Fig: what are maintained by AWS and what are maintained by us.