EC2

We have the hosts (AWS physical server that have physical CPU, Memory, NIC, and Disk). In the instances we have guest instances / compute instances that is the EC2 instance.

EC2 instance can be shared (instances are created in the shared host) / dedicated (instances are created in the dedicated host and you are not sharing with other clients)

Okay...Now how the CPU, Memory, NIC, Disk is shared between the instances??? Here comes virtualization layer... This virtualization layer gets in contact with the actual CPU, Memory etc. The virtualization then emulates (reproduce the functionality/ actions) and split the resources between the instances and they make the instances to feel like it is owning the dedicated CPU, Memory, disk etc.

Operating system can be the Linux/ windows. You have root access for your EC2 instances. [Root access is the authorization to execute any command and access any resource on the device) [*Feel like you have your own physical server and you have access to it]*

You can stop, restart, reboot or terminated your instance.

EC2 availability SLA is 99.95% for each region during the monthly billing period. (approx., 22 min per month vanthu downtime ah irukum)

Thus EC2 instance provides the flexible compute capacity.

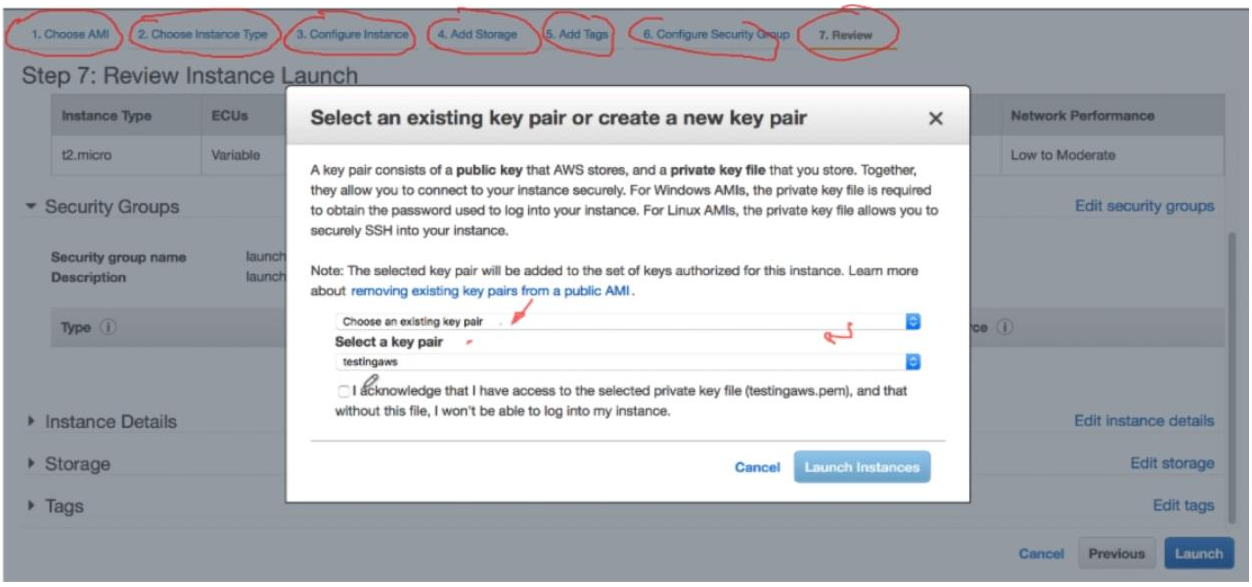
EC2 –INSTANCE ACCESS

To access an instance you need a key and key pair name. There are two types of key -> **public** key [AWS responsibility (they will store the key)] and the **private** key (we need to store the key and it's our responsibility). You can only download it once when you about to launch the instance.

What if we lost the private key .pem??

If the old instance is not necessary, then you can terminate the instance. If not you have an another way to recover the instance

1. From your WAS console stop the instance in question
2. Create a snapshot of the instance
3. Create a duplicate instance from the resulting snapshot and create a new Key Pair.

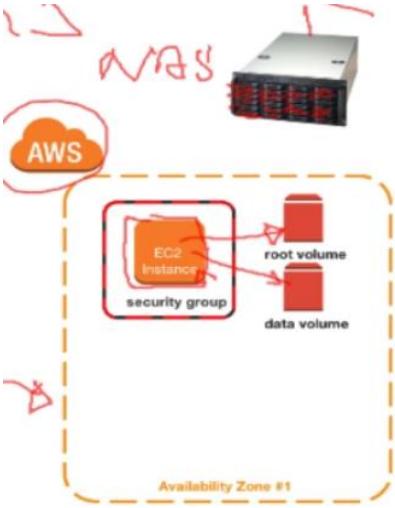


- There is a 20 EC2 instances soft limit per account, you can submit a request to aws to increase it.
- EC2 instance supports two types of the Block Store devices
    - Elastic Block store (EBS)
    - Instance Store

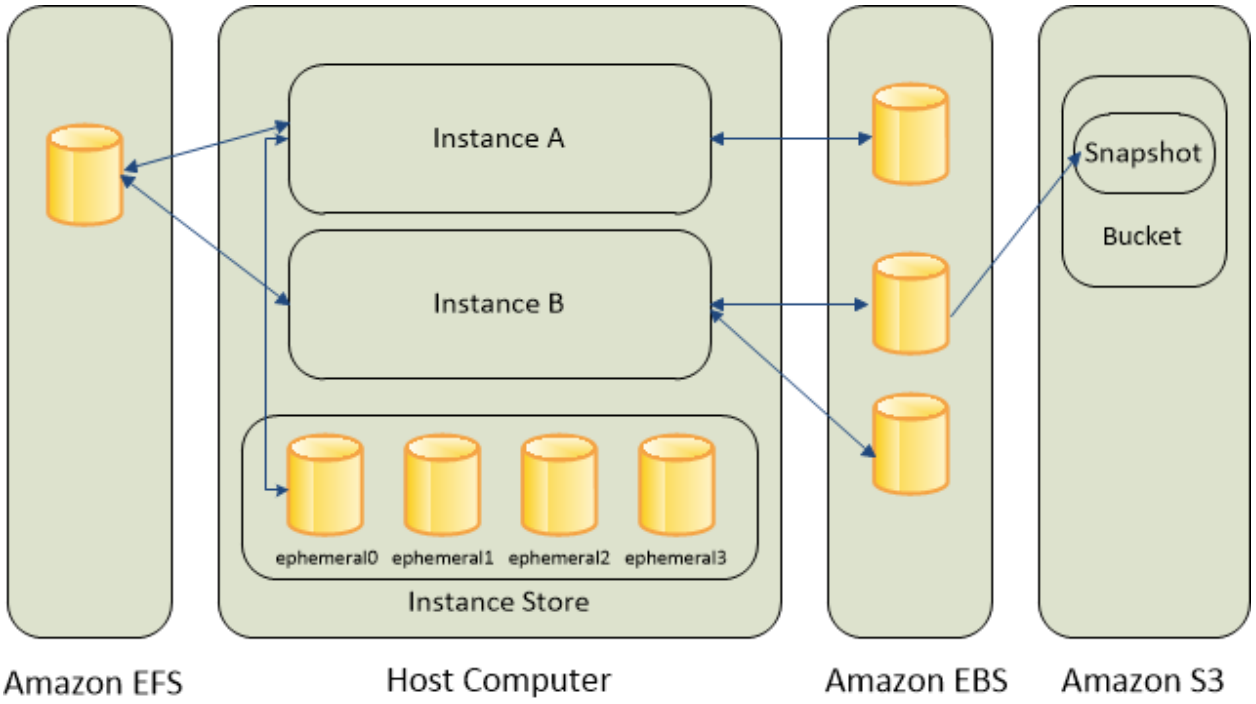
What is block store devices? Think of the hard disk drive on your computer/Laptop... That is where your data is stored. EC2 instance is more or less a compute node. The compute node needs to store the data

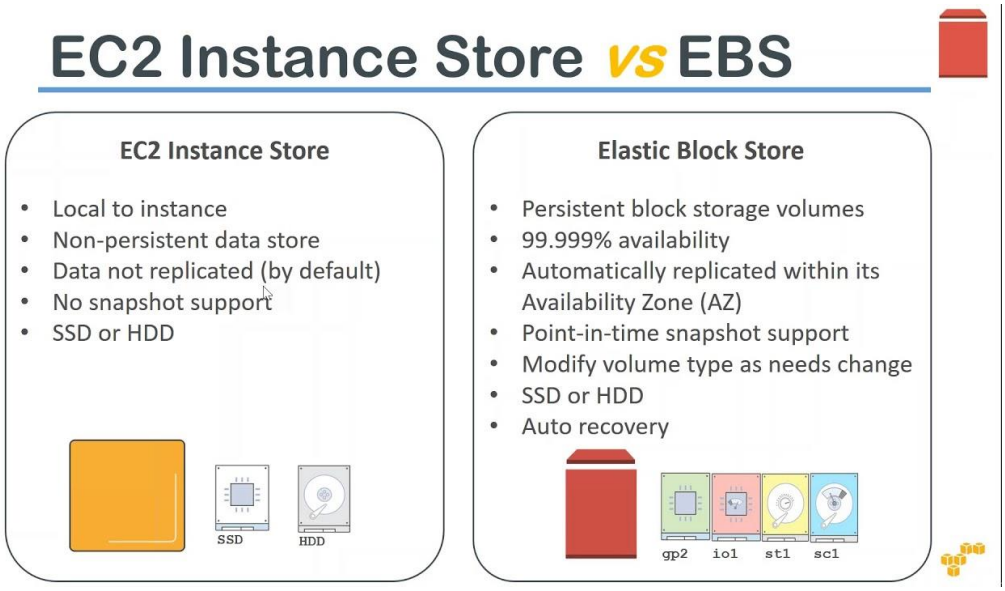
EBS- Elastic Block Store is **PERSISTENT, Network attached Virtual drives. (**NAS [Network Attached Storage])

Story: Ok whether EBS is present in the EC2 instance??? No, it is present in the AWS infrastructure inside the availability zone. When u want an EBS for the EC2, then the EC2 is attached to the root volume and the data volume of the EBS. Root volume is where the operating system is present. Data volume is where the data inside the EC2 is present.



Instance Store - Basically the virtual hard drive on the host allocated to this EC2 instance. Limited to10 GB per instance .It is not persistent





EC2 –Root / Boot Volume

- EC2 instance root/ boot volumes can be EBS or Instance Store Volumes
- EBS- Backed EC2 instance
  - This means it has EBS root volume
  - It is persistent. Even if you delete the EC2 instance , you can store restore/ keep the volume/ storage
- Instance-store - Backed EC2 instance
  - This means it has instance-store root volume
  - Instance-store is an Ephemeral storage (temporary storage). If you terminate the instance then the storage/volume is also deleted

EC2 –instance families

EC2 instance are grouped instance family. Some are optimized for storage, some for memory. CPU, graphics.

- **General Purpose**
  - Balanced memory and CPU
  - Suitable for most applications
  - Ex. M3, M4, T2

- **Compute Optimized**
  - More CPU than memory
  - Compute & HPC intensive use
  - Ex. C2, C4

- **Memory Optimized**
  - More RAM/memory
  - Memory intensive apps, DB, and caching
  - Ex. R3, R4

- **GPU compute instances**
  - Graphics Optimized
  - High performance and parallel computing
  - Ex. G2

- **Storage Optimized**
  - Very high, low latency, I/O
  - I/O intensive apps, data warehousing, Hadoop
  - Ex. I2, D2

EC2 –Block Store Types

AWS provides the following EBS volume types, which differ in performance characteristics and price which can be tailored for storage performance and cost to the needs of the applications:

- **SSD-backed** volumes optimized for transactional workloads involving frequent read/write operations with small I/O size, where the dominant performance attribute is **IOPS**
  - General Purpose SSD (gp2)
  - Provisioned IOPS SSD (io1)
- **HDD-backed** volumes optimized for large streaming workloads where **throughput** (measured in MiB/s) is a better performance measure than IOPS
  - Throughput Optimized HDD (st1)
  - Cold HDD (sc1)

	Solid-State Drives (SSD)		Hard disk Drives (HDD)	
Volume Type	General Purpose SSD (gp2)*	Provisioned IOPS SSD (io1)	Throughput Optimized HDD (st1)	Cold HDD (sc1)
Description	General purpose SSD volume that balances price and performance for a wide variety of workloads	Highest-performance SSD volume for mission-critical low-latency or high-throughput workloads	Low cost HDD volume designed for frequently accessed, throughput-intensive workloads	Lowest cost HDD volume designed for less frequently accessed workloads
Use Cases	• Recommended for most workloads • System boot volumes • Virtual desktops • Low-latency interactive apps • Development and test environments	• Critical business applications that require sustained IOPS performance, or more than 10,000 IOPS or 160 MiB/s of throughput per volume • Large database workloads, such as: ◦ MongoDB ◦ Cassandra ◦ Microsoft SQL Server ◦ MySQL ◦ PostgreSQL ◦ Oracle	• Streaming workloads requiring consistent, fast throughput at a low price • Big data • Data warehouses • Log processing • Cannot be a boot volume	• Throughput-oriented storage for large volumes of data that is infrequently accessed • Scenarios where the lowest storage cost is important • Cannot be a boot volume
API Name	gp2	io1	st1	sc1
Volume Size	1 GiB - 16 TiB	4 GiB - 16 TiB	500 GiB - 16 TiB	500 GiB - 16 TiB

	GENERAL	Provisioned	Throughput	Cold HDD
Volume Size	1 GiB - 16 TiB	4 GiB - 16 TiB	500 GiB - 16 TiB	500 GiB - 16 TiB
Max. IOPS/Volume**	16,000**	64,000**	500	250
Max. Throughput/Volume	250 MiB/s**	1,000 MiB/s	500 MiB/s	250 MiB/s
Max. IOPS/Instance††	80,000	80,000	80,000	80,000
Max. Throughput/Instance	1,750 MiB/s	1,750 MiB/s	1,750 MiB/s	1,750 MiB/s
Dominant Performance Attribute	IOPS	IOPS	MiB/s	MiB/s

EC2 –Block Device Mapping

Consider an instance that has multiple volumes, EBS or instance store

When you have a plain AMI, what would it have? It will have a ROOT VOLUME. Now we need to customize the AMI that will have 1 instance store and 2 EBS Volume.

1. Create instance from the OS that I want... We need to pick from the AMIs -> in the storage add a instance store and EBS volume -> Launch it
2. Now the instance is launched , now create the AMI from the instance (u can always customize the AMI)

Now we can talk about the Block device mapping -> it will contain the details of how many volumes, what are the types of the volumes, are they are EBS / instance store? Depending on the type of the instance, they can support EBS/ instance store/ both.

Can I change the instance device mapping after launch? Yes. But it can be done only in the EBS (you can add/remove the EBS after the launch of the instance). You cannot do that in the instance store.

When u want to choose the AMI for launching the instance you will see only the available EBS but not the instance-store. Refer the below slide for clarification.

Elastic Compute Cloud

EC2 – Block Device Mapping

- Block device mapping, is the mapping of block storage devices in an AMI
  - This includes both EBS and Instance Store volumes
  - Used to define which block storage volumes (root and data) to include/create when a instance is launched from the AMI
  - You can view it and it shows both EBS and Instance-store volumes
  - On the other hand, EC2 instance's Block device mapping from the AWS Console show only EBS volumes of the Instance
    - To show the Instance-store volumes of the Instance block device mapping, you need to query the instance metadata

```
[ec2-user ~]$ curl http://169.254.169.254/latest/meta-data/block-device-mapping/
```

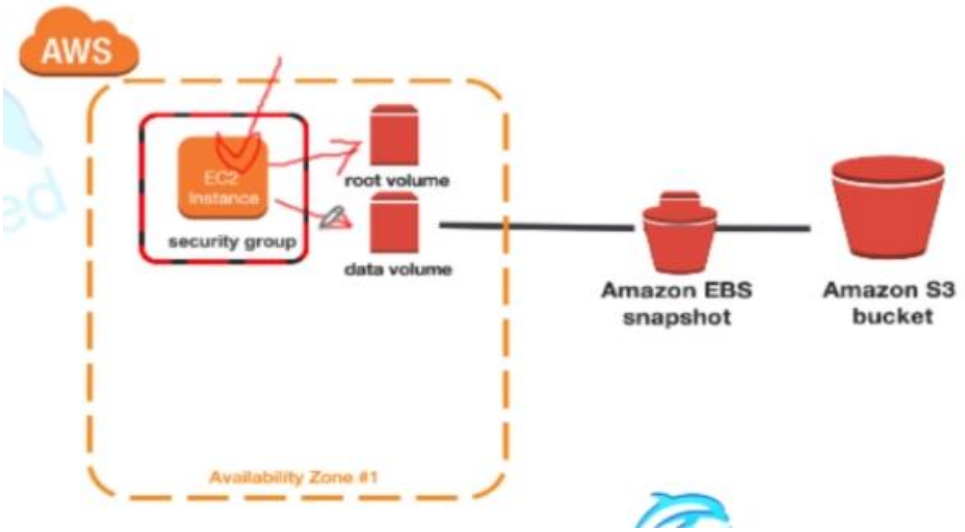
- You can make changes to the block device mappings while launching the instance or later when it is launched

***Volume size of the EBS can be increased but cannot be decreased. Volume type can also be changed (e.g... From general to provisioned)

*** The volume size of the EBS should be equal to / less than the size of the Snapshot that is specified in the block device mapping in the AMI (hence keep increase/ keep the volume size but not decrease it)

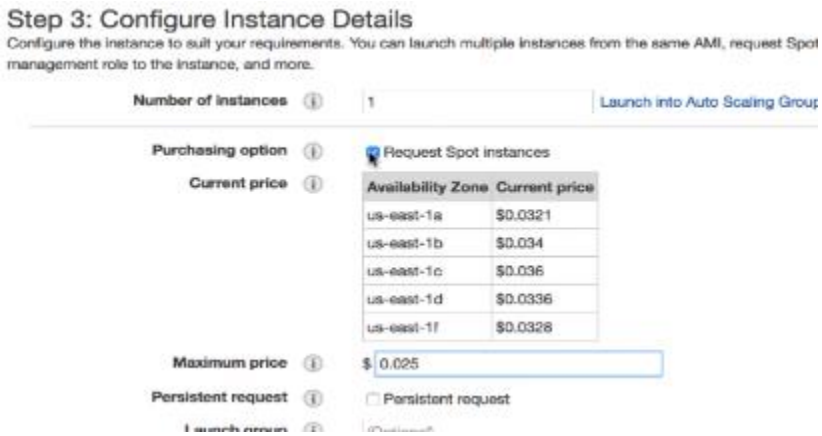
Advantages of EBS

- EBS has 99.99% availability
- You can create point-in-time snapshots for your EBS volumes (point in time -> for the particular moment)
  - This can be done manually or automatically using the life cycle management
- You can resize the EBS up but not down
  - You can create this by adding a new volume of snapshot of the volume



What is spot instance?

While creating EC2 instance, we can bid for the instance … like there will be instance available in different availability zone with the amount of money charged per hour for each instance. You can bid for money for instances, when the instance bidding range reaches with the amount you mentioned, then the particular instance will be allocated to you.



Practical

Step 1:EC2 --- launch instance---- select the AMI---- Select the operating system -----

Step 2: *Configure the instance details:*

Network: choose the VPC

Subnet: It is recommended to leave AWS to select the availability zone because they know about the capacity of each availability zone

Auto Assign IPV4: In each EC2 instances that you will create in any of the availability zone, you will have a public IPV4 address assigned to your EC2 instance.

Remember that the IPV4 is not configured in the EC2 ENI. It is configured in the Internet gateway of the VPC and through NAT it is mapped to the IPV4 address to eth0 of your instance

Shutdown behavior: When the operating system in the instance is shutdown… whether the instance need to Stop or Terminate?

Enable termination protection: Protects you from accidentally terminating the instance.

If it is protected Na: U can shut down the instance in the operating system… So the termination process will be enabled

Monitoring: AWS EC2 is constructed using the EC2 service. EC2 by default will monitor the basic metrics like 2 cloud watch for every 5 minutes.

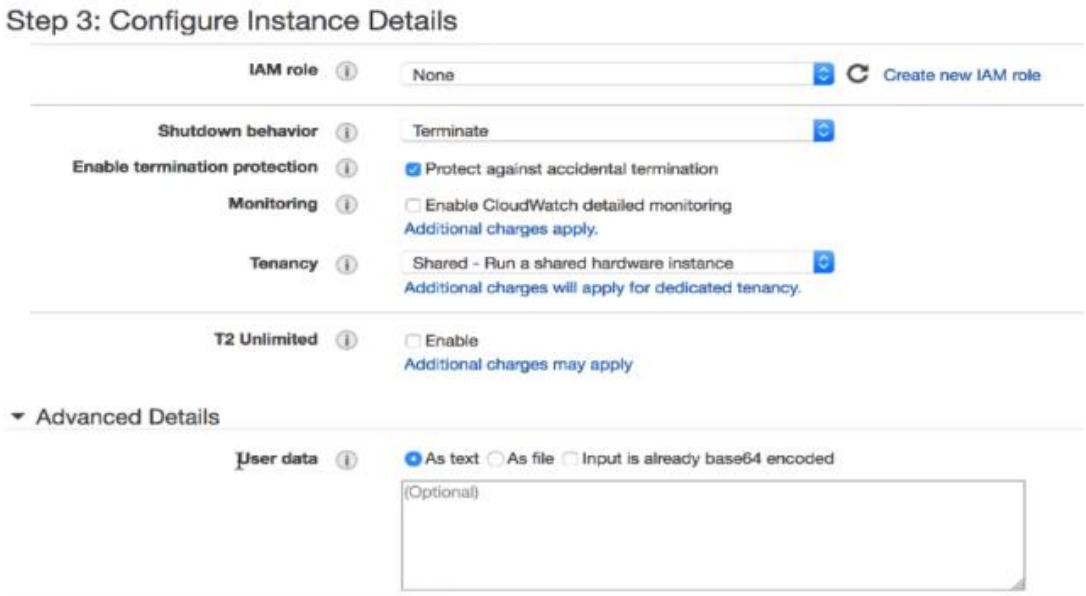
What if you want the monitoring in the minute basis……?? Remember this is chargeable. Even if you in free tier you will pay for the detailed monitoring.

Tenancy: Choose how the instance needed to be launched whether shared / dedicated.

With **T2 Unlimited** instances, you have the ability to sustain high CPU performance over any desired time frame while still keeping your costs as low as possible.

Advance Details: User data: Here you will have the script that will run when you boot the instance

E.g. If you want to run a patch for the security update in the instance... Instead of doing it manually, you can mention the URL/code …. So that the OS will automatically update the security in your instance.



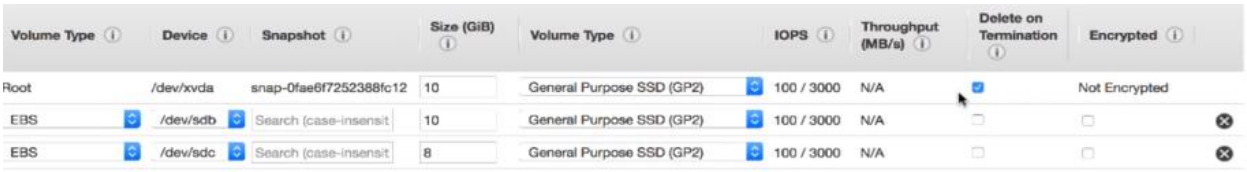
Step 3: *Add storage*

Root Volume : This is present in default when you try to launch the instance. The OS and other program files will be saved in the root volume. "**Delete on Termination**" will be there by default in the root volume.[so that when you delete the EC2 instance, it will go away]

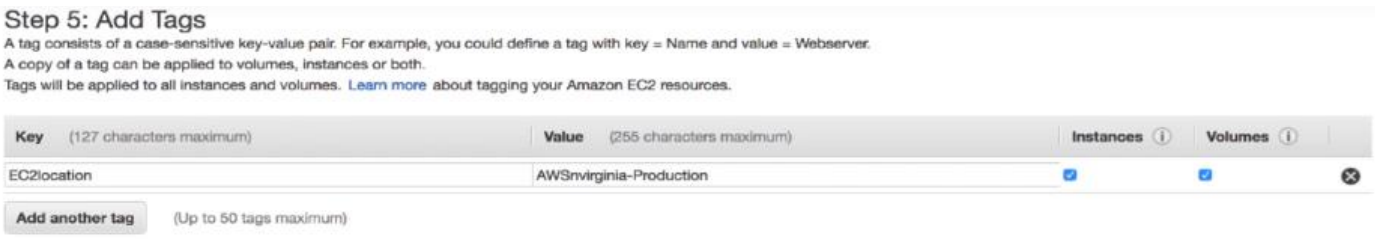
Volume type: For any critical / production database .. choose –"provisioned"

For any test database---- "General"

Encrypted : The **root volume** will either be in the encrypted phase / not encrypted phase. There are some of the AMI with the root volume in the encrypted phase. Select that accordingly. But **data volumes** can be either encrypted / not encrypted in the "ADD Storage"[You can edit it]



Step 4: *Add Tags*



Step 5: S*ecurity Groups*

You can either create a new security group / default Security group. While creating the Security group… please be very careful Source should not be **0.0.0.0/0** this means anyone can access the EC2 instance. This can lead to an malicious attack

Use SSH for Linux , RDP (remote desktop) for Windows

Step 6: Key Pair

The key pair is used to connect the EC2 instance in the AWS environment using the public key and private key . Public key will be handled by AWS and it is our responsible to save the private key . The private key will be generated once in the while launching the EC2 instance

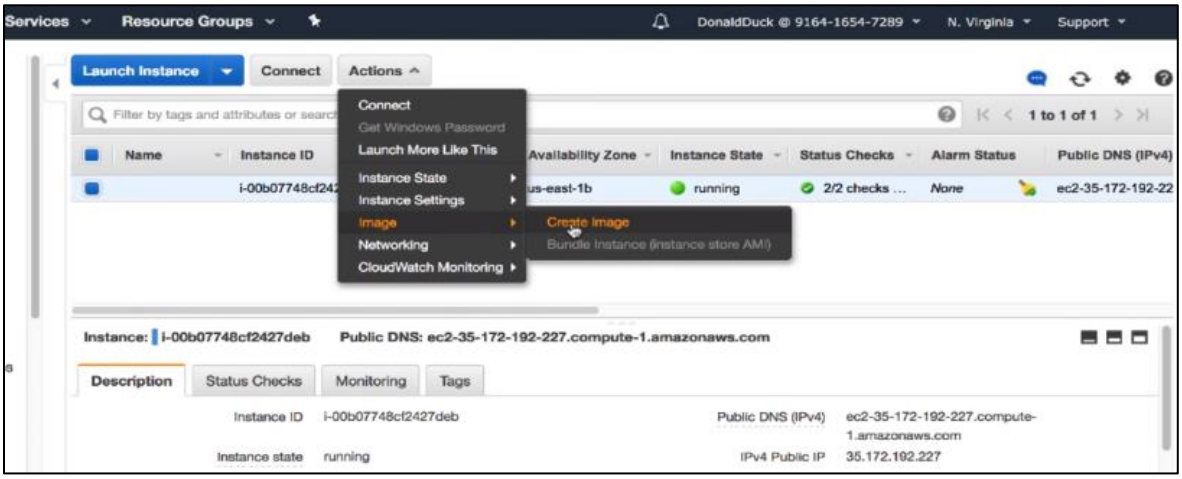
Step 6: Launch and connect the instance

Encrypting root volume in the EC2 instance

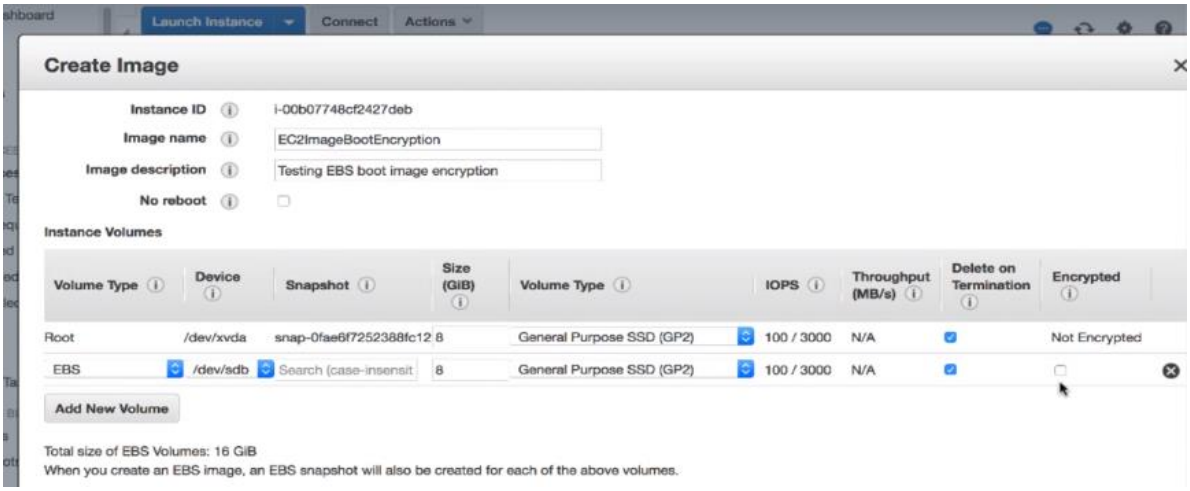
Remember that you can launch the instance indirectly with the encrypted boot volumes but you need to launch the instance for the first time .

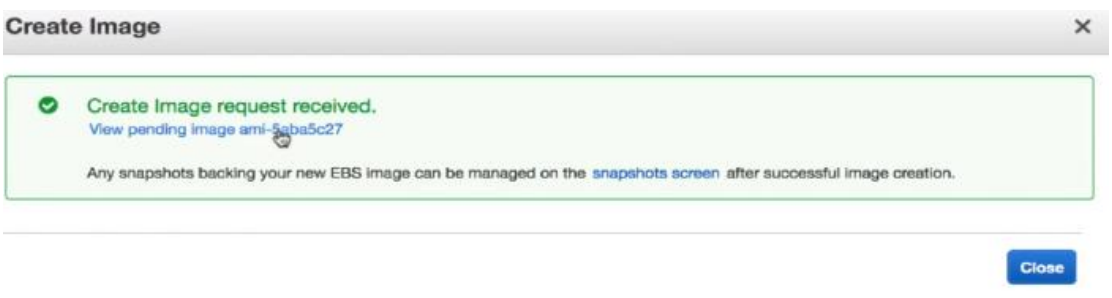
Let us see the process….

- CREATE THE IMAGE



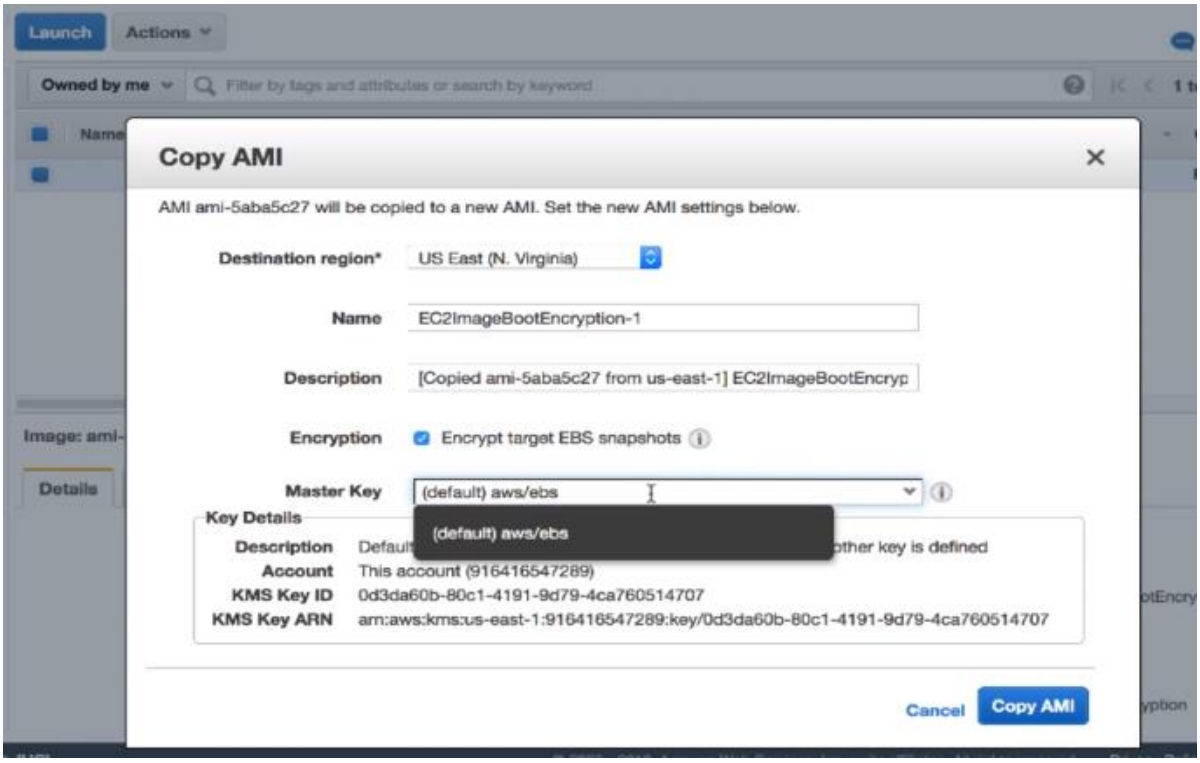
You are creating AMI from the existing instance





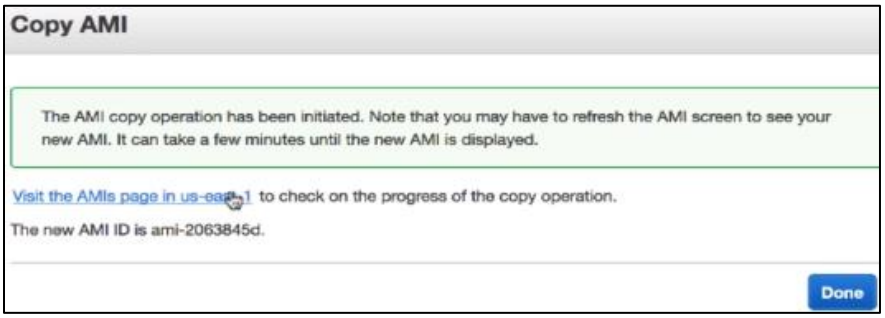
Images [side] --→ AMIs -→ Now wait until the image is available → “Now we have created a image for the EC2 instance that does not have the root volume ENCRYPTED”

Now the image is active→ select that image → Go to Actions(top) → “ Copy AMI” →



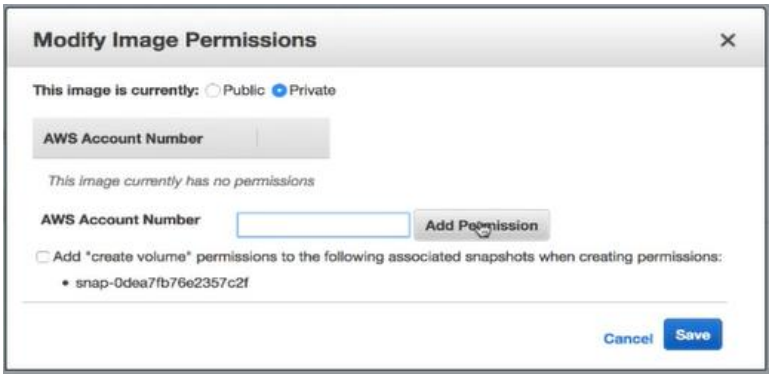
When I am copying the AMI , I am requesting the snapshots for the root volume and any other EBS volume we have which will be like the original AMI.

Now we are going to encrypt that volumes in the original AMI using a default Master key .



Now the copy image is created. This image **will appear only for this account.** You can modify the account into public/ private in **PERMISSIONS (**below). You can share the image with your friends/ colleagues using the Account number . As the image is encrypted with the KMS (Key management service) . You cannot make it public at this point of situation.

PERMISSION FOR SHARING THE IMAGE PRIVATELY



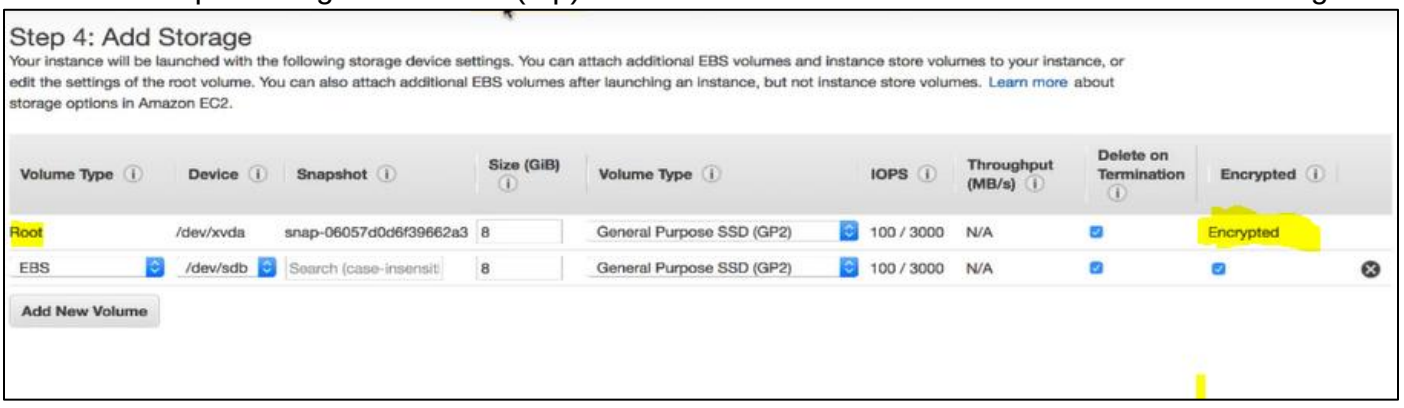
PERMISSION FOR SHARING THE IMAGE PUBLICLY

Note : The Not encrypted root volume can be made both public and private.

The encrypted root volume can be made only private and not public

Now we are going to create an instance using this AMI. (Now the root volume is supposed to be encrypted)

Select that copied image → Actions (top) → Launch → Select free tier instance → Add Storage →



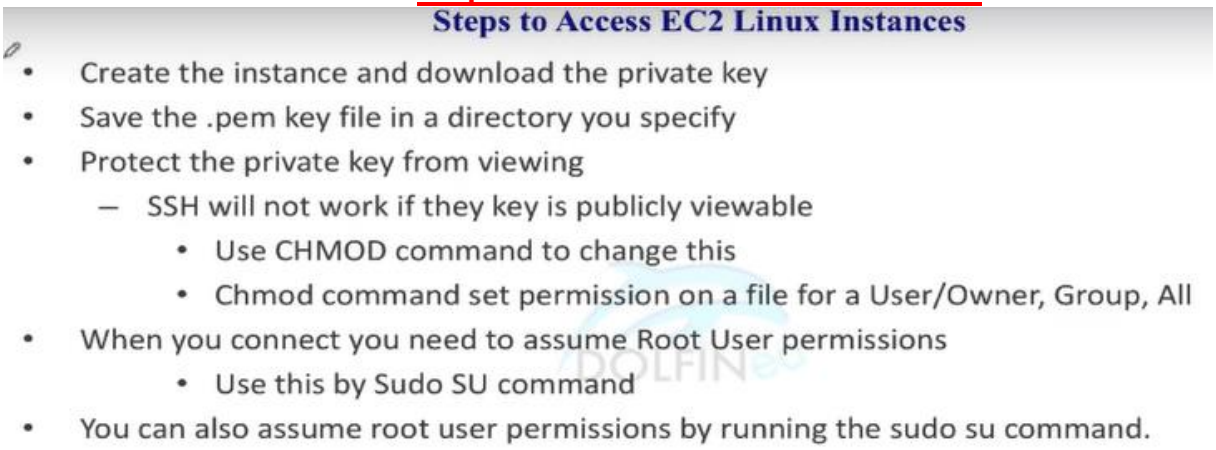
Now launch the instance -> Acknowledge -> Now the instance is launched

“”””If you go to the free tier AMI and launch the instance na -> you cannot encrypt the root volume”””

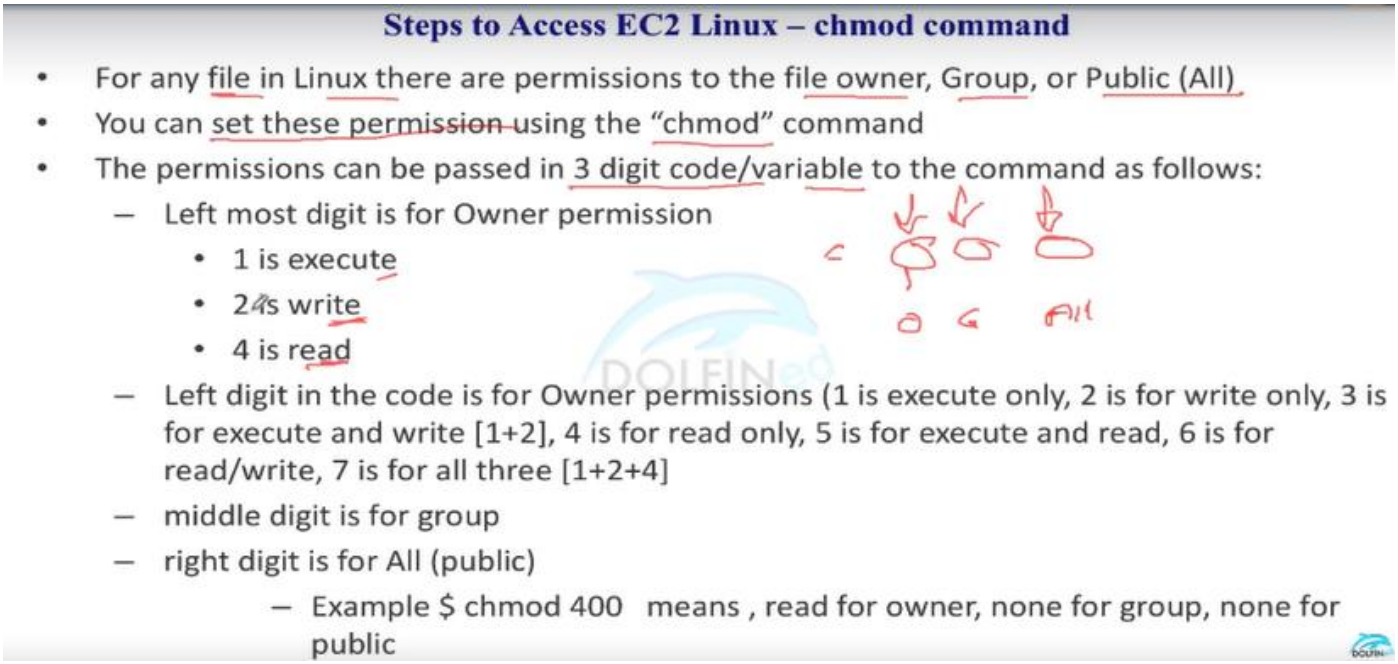
You need to encrypt the root volume when you create the EC2 instance. Because it is an compliance and all the customers who launch the instances in the cloud will expect the root volume in the encrypted way .

Now the root volume of the instance is encrypted. Remember that the EBS is not present inside the instance . They are operated through the AWS infrastructure. The Encrypted data will be in the EBS root volume and if the EC2 instance is having the correct key then the data is decrypted accordingly

Steps to access EC2 Linux Instances



Steps to access EC2 instance in the CHMOD[CHange MODe] command



Steps to access EC2 instance in the SUDO command

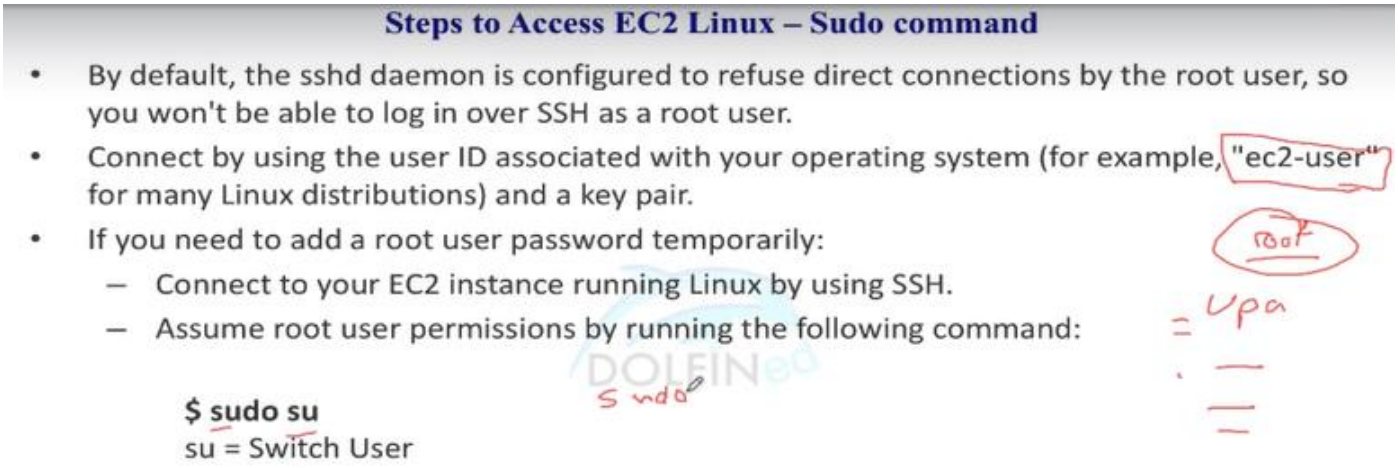
Sudo is Super User Do . **su:** Switch User

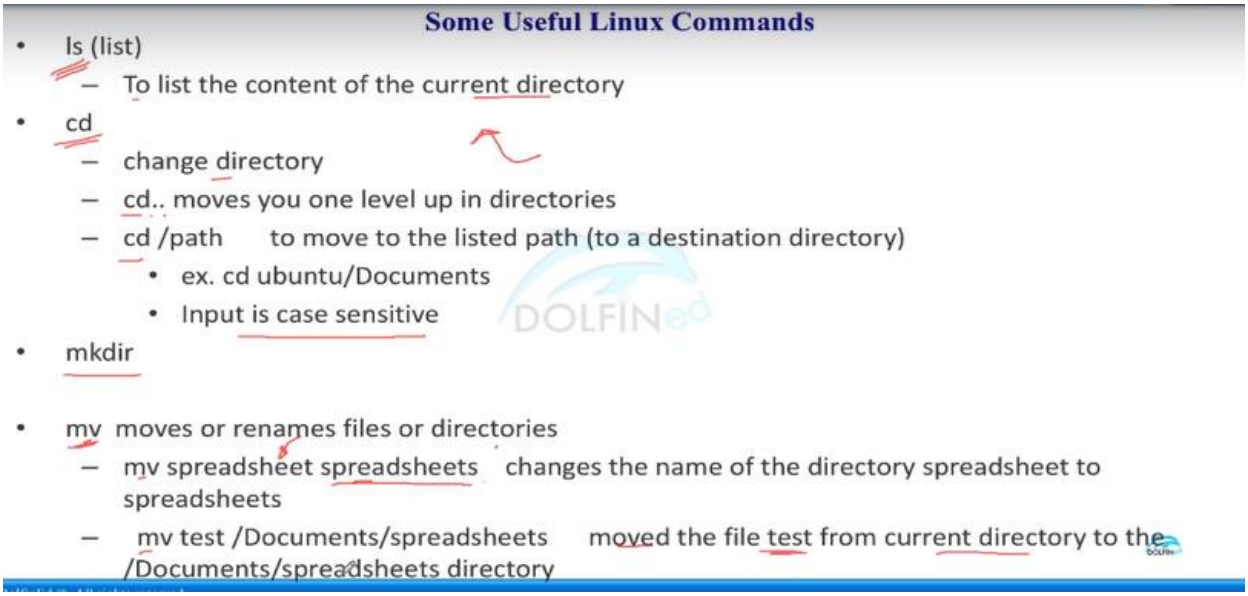
All the instances will be having the Root user with some root privileges for execute, read, write, update etc . For example : Karthika is the root user in instance A then Kalpana wants to update the software in the instance A. Then in SUDO database I will add kalpana . So that she will get all the access given by the root user.

Sudo su- This will make you to login as the different **Super User.**

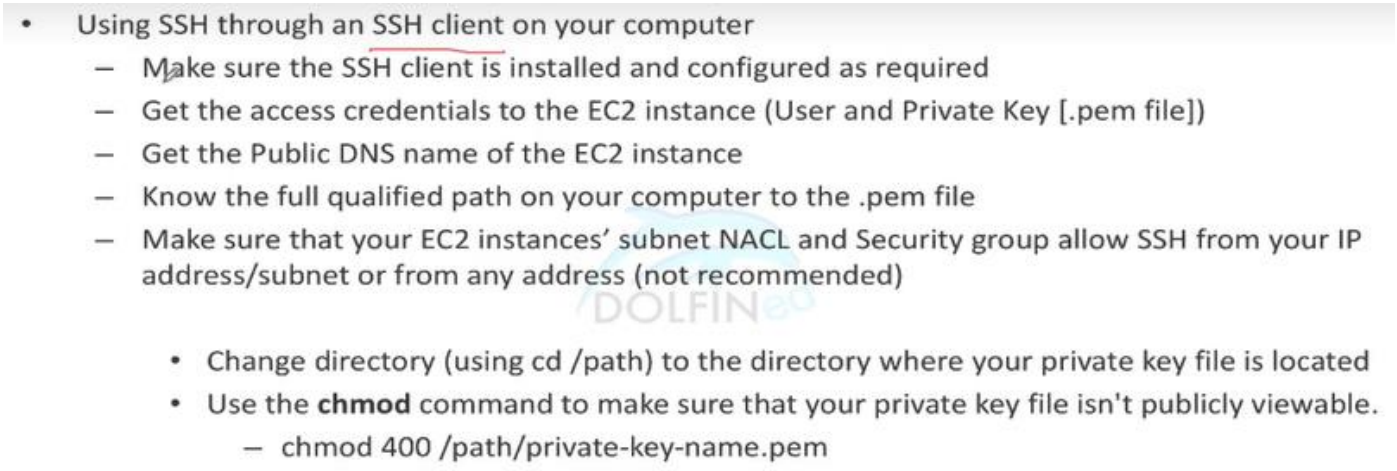
Why we are not able to access the instance using the root user? This is because the root user has all the access to change the system files. This is filled with many security risks. The root user can do many things an ordinary user cannot, such as changing the ownership of files, mounting disk, formatting & restating new file system, starting/stopping services, and binding to

ports numbered below 1024 and more. So we are logging in as different user which will have the privileges of the root user except editing of files that will have security risks.





To connect to the LINUX EC2 instance



- Use the **ssh** command to connect to the instance.
  - You specify the private key (.pem) file and*user_name@public_dns_name*.
    - For Amazon Linux, the user name is ec2-user.
    - For RHEL, the user name is ec2-user or root.
    - For Ubuntu, the user name is ubuntu or root.
    - For Centos, the user name is centos.
    - For Debian, the user name is admin or root.
    - For Fedora, the user name is ec2-user.
    - For SUSE, the user name is ec2-user or root.
    - Otherwise, if ec2-user and root don't work, check with your AMI provider.
- ssh -i */path*/my-key-pair.pem ec2-user@ec2-198-51-100-1.compute-1.amazonaws.com

YUM

YUM (Yellowdog Updater Modified) is an open source command-line as well as graphical based package management tool for RPM (RedHat Package Manager) based Linux systems. It allows users and system administrator to easily install, update, remove or search software packages on a systems. It was developed and released by Seth Vidal under GPL (General Public License) as an open source, means anyone can allowed to download and access the code to fix bugs and develop customized packages. YUM uses numerous third party repositories to install packages automatically by resolving their dependencies issues.

YUM installer package/command

- YUM = Yellodog Updater Modified
  - Is a software packages tool that allows users and system administrator to easily install, update, remove or search software packages on a system.
    - It automatically computes dependencies and figures out what things should occur to install packages.
    - It can automatically perform system updates
- Examples:
  - Yum update –y (-y tells linux to assume Yes to any question while doing the update)
    - If run without any packages, update will update every currently installed package.
    - If one or more packages or package globs are specified, **yum** will only update the listed packages.
  - Yum install <package-name>
    - $ yum install httpd [-y]
- $ service httpd start

TEXT File Editors

- Nano and vi
- Nano is simpler to use
  - $ nano default.html

CONNECT TO THE INSTANCE

When you connect the instance

$ cd downloads //change the directory towards the downloads

$chmod 400 EC2Testing.pem //Change mode- read for file owner, no access to group, no access to public

$ ssh –l “EC2Testing.pem” ec2-user@ec2-35-172-192-227.compute-1.amazonaws.com

$yum update

//You will get the error here . You need to be root to execute the above command

// Now we are changing the super user

$sudo su

$yum update –y

Now connect to the instance2 where we created EC2 instance using the Copied AMI where the root volume is Encrypted.

$ ssh –l “EC2Testing.pem” root@ec2-35-172-192-227.compute-1.amazonaws.com

//Now 4 packages needed for security out of 5 available

//Run Sudo yum update to apply all the updates

$yum update

// You need to be root to execute the above command

You can do this in 2 ways $Sudo su **or** $sudo su ec2-user

$sudo su ec2-user **or** $sudo su

$sudo yum update –y **or** $yum update -y

$yum install httpd –y (instead of httpd u can install python/ mysql)

Can I do ICMP ping to the instance ?

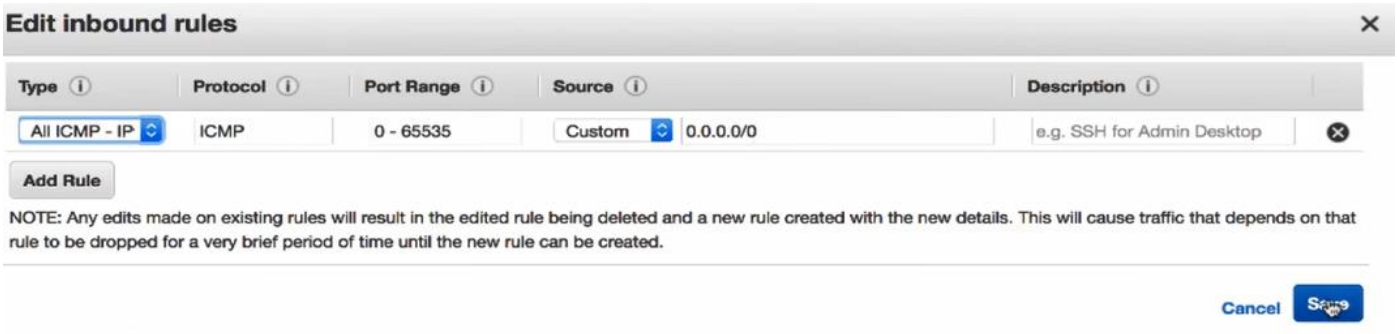
Try to ping the public IP address of the instance…..

$ping 54.245.243.25

//U will get the error in connecting

Below “Description” -> security group -> click on the link -> It will get navigated to the “Security Group”

In the Inbound -> change the **SSH** to **ICMP**



Try to ping the public IP address of the instance…..

$ping 54.245.243.25

// You can able to connect

Note

When you stop the EC2 instance, the public IPV4 allocated dynamically to the instance will get deleted. Again when you restart it, then a new public IPV4 IP address will be allocated to the instance.

If you have Elastic IP address assigned to your instance, It will still persist even if you terminate the instance. Because you will create the elastic IP address common to the **VPC.** You need to release the Elastic IP address manually.

If the instance does not have Elastic IPV4 IP address.

Stop, Terminated – Public IPV4 address is deleted

Reboot, Start --– Public IPV4 address is not deleted

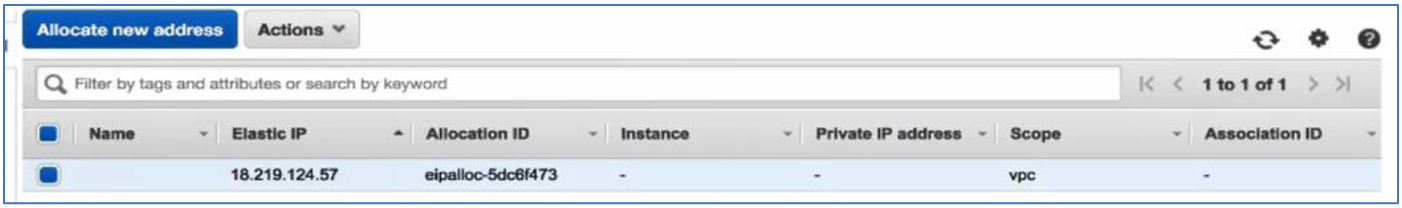
If the instance have Elastic IPV4 IP address.

Stop, Terminated(without releasing the IP address) , Reboot, Start – Public IPV4 IP address is not getting deleted.

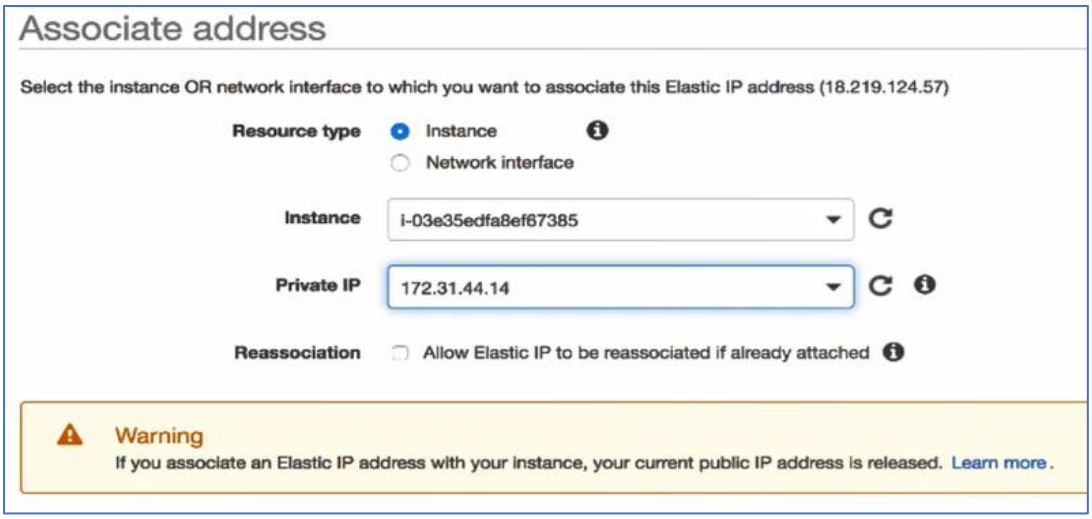
Until u go and manually release that address,It will not get deleted

Elastic IP Address

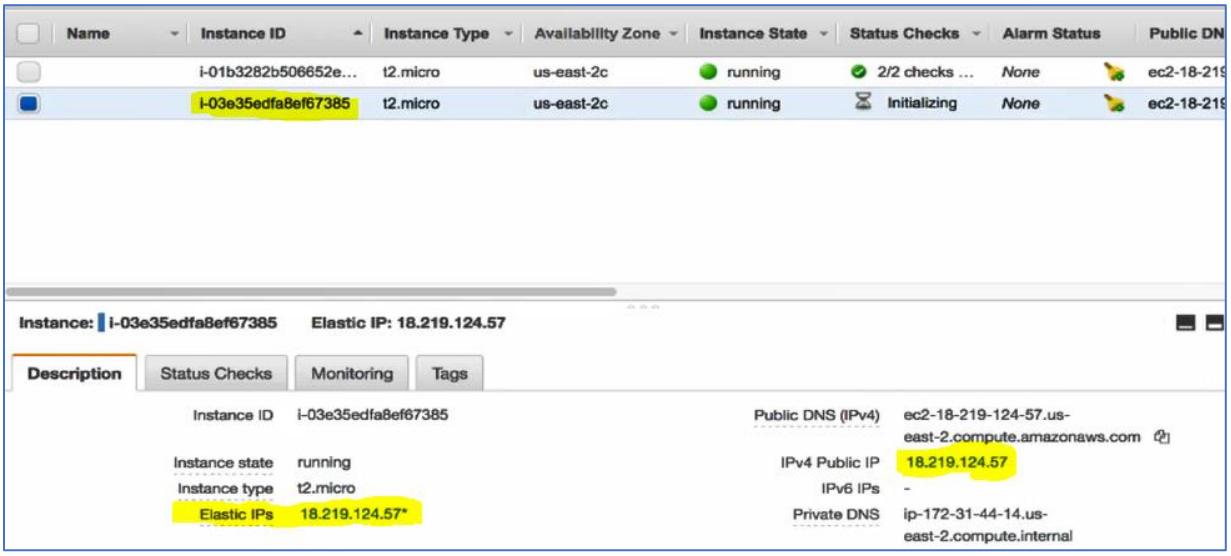
Go to Elastic IP address under the “Network & Security” -> “Allocate New Address” ->



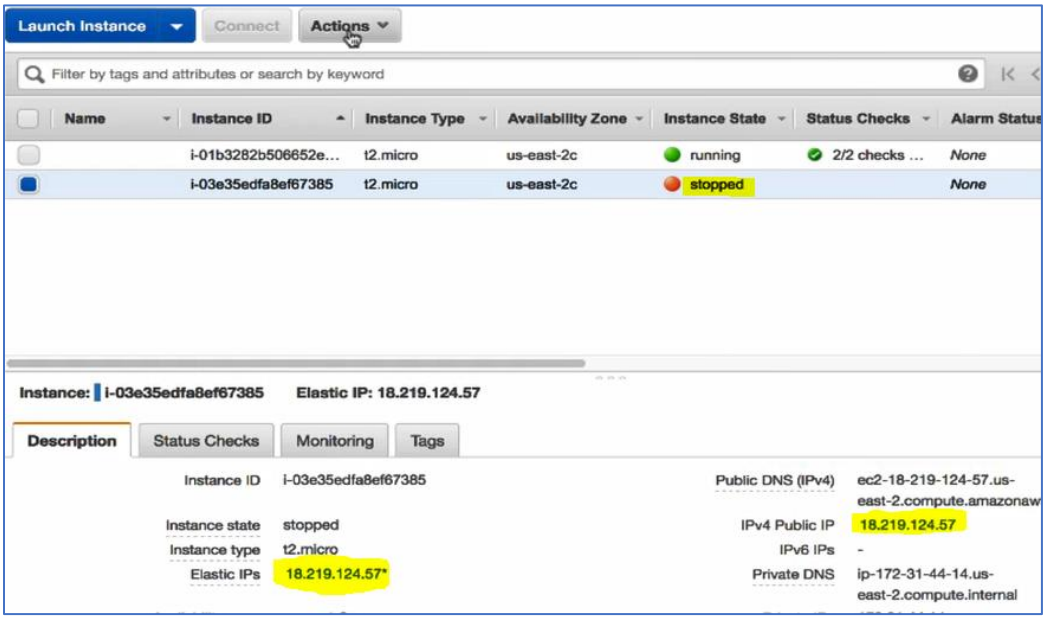
Actions -> “Associate the address”



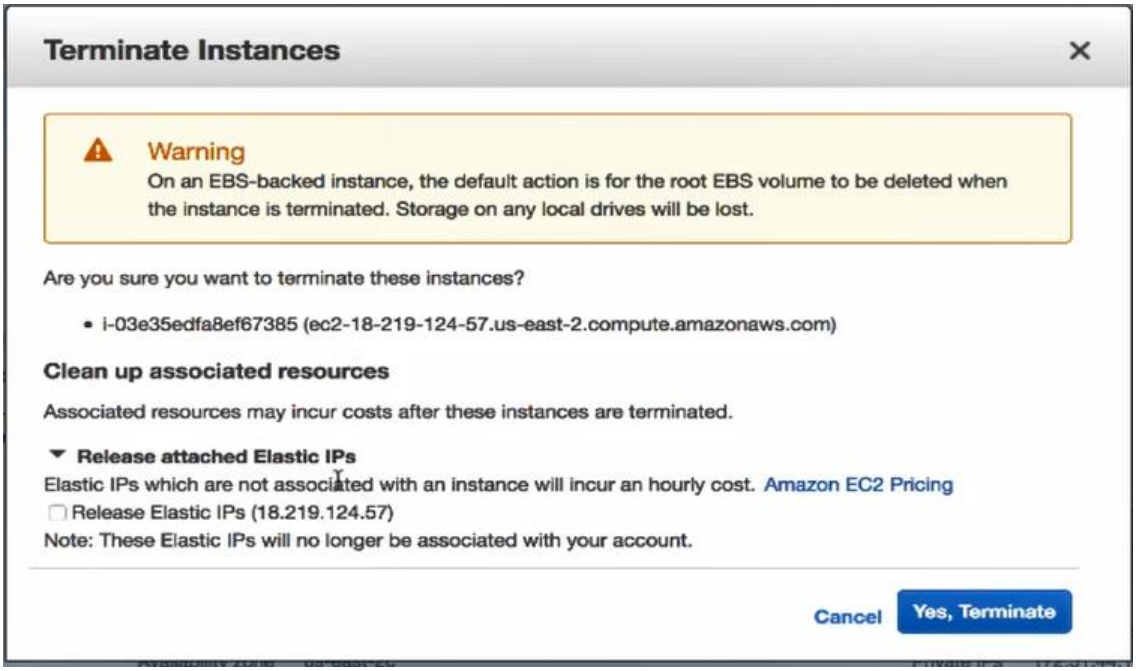
U can see the elastic IP address in the instance that you have associated.



Now stop the instance and wait



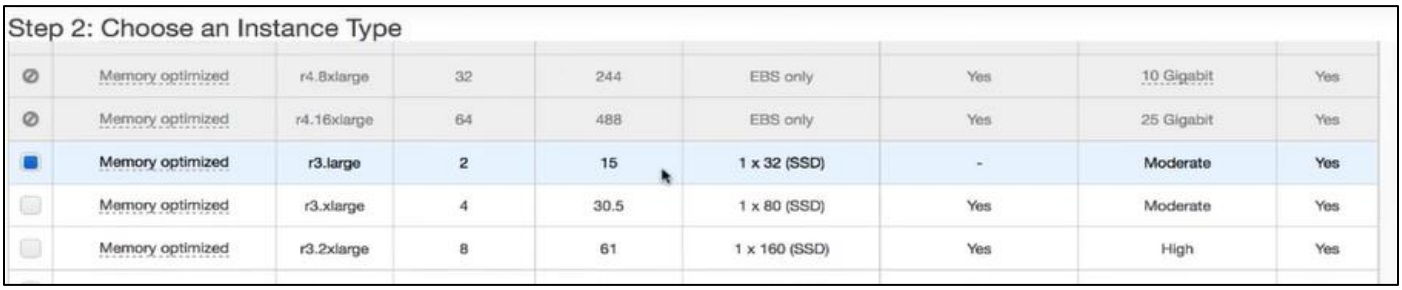
When you try to terminate the instance

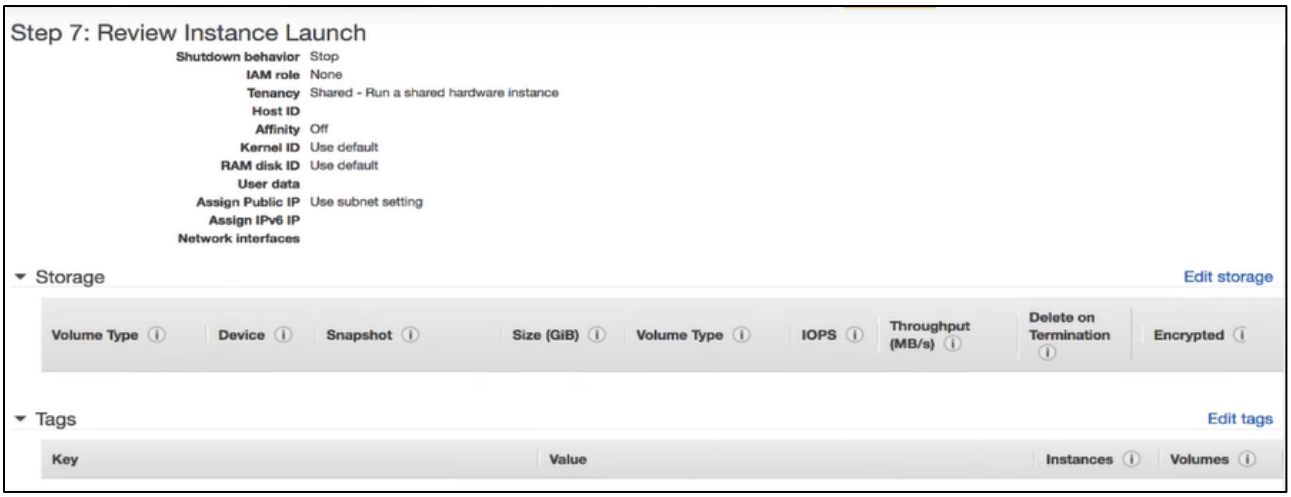


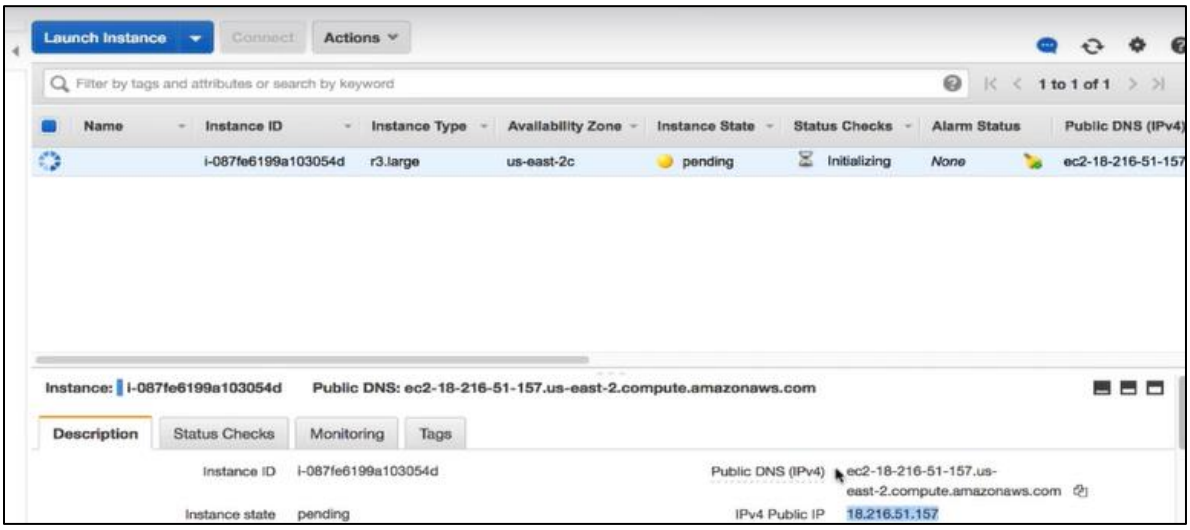
You can release the Elastic IP address and then terminate the instance. When you terminate the instance without releasing the elastic IP address, the elastic IP address remains. If you have the elastic IP address without utilizing it, then it will cost you much. If you associate the address to the instance then it does not charge money.

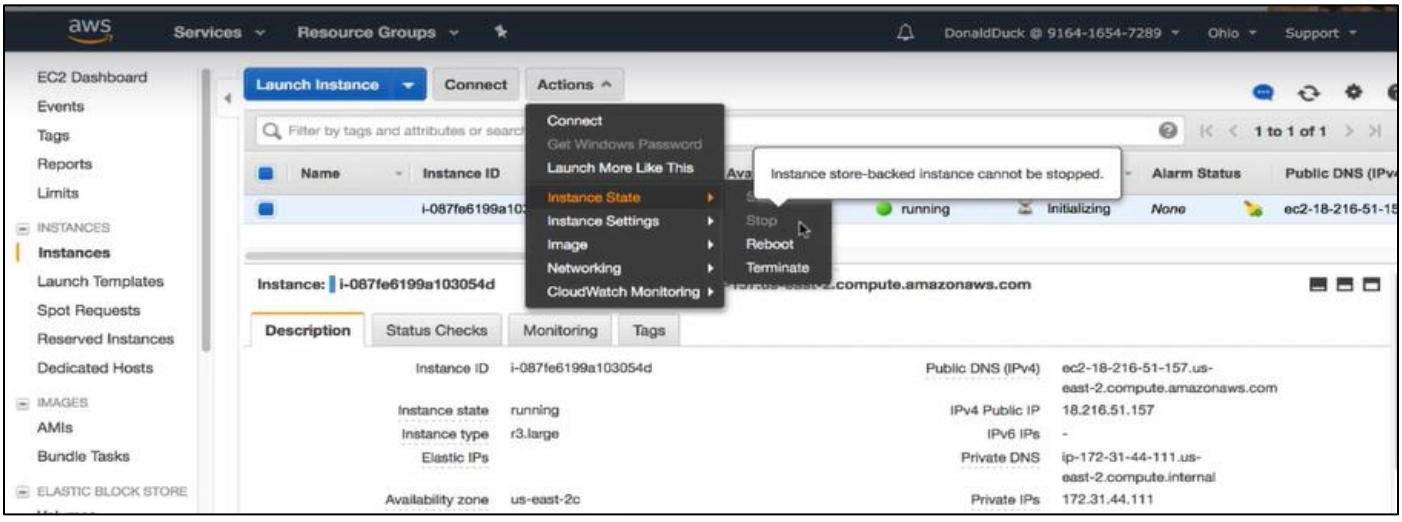
Launch the EC2 instance – Instance store- backed

1. In AMI, under community AMI -> select “Community AMI “ and select the “Instance Store” -> ok
2. If you see that the least one in the instance will be “r3.large” – when u launch the instance the AWS will charge you in seconds (per second billing).









Instance store does not have “Start” and “stop” options. Because the instance-store is nothing but the virtual memory on the host . EC2 instance is the virtual machine in the physical host. This virtual memory will get attached to the virtual machine. Once the virtual machine is stopped then the entire data in the attached virtual memory is lost. “**Instance store backed instance cannot be stopped**”

AWS can schedule events for your instances, such as a reboot, stop/start, or retirement. These events do not occur frequently. If one of your instances will be affected by a scheduled event, AWS sends an email to the email address that's associated with your AWS account prior to the

scheduled event. The email provides details about the event, including the start and end date. Depending on the event, you might be able to take action to control the timing of the event.

Types of Scheduled Events

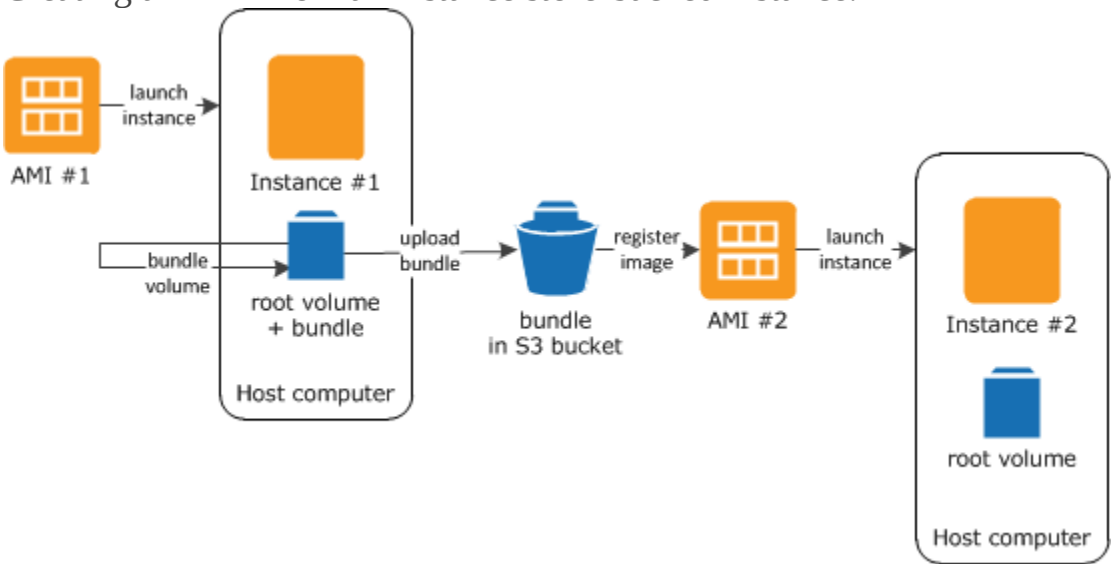
Amazon EC2 supports the following types of scheduled events for your instances:

- **Instance stop**: The instance will be stopped. When you start it again, it's migrated to a new host. Applies only to instances backed by Amazon EBS.
- **Instance retirement**: The instance will be stopped if it is backed by Amazon EBS, or terminated if it is backed by instance store.
- **Instance reboot**: The instance will be rebooted.
- **System reboot**: The host for the instance will be rebooted.
- **System maintenance**: The instance might be temporarily affected by network maintenance or power maintenance.

Remember:
>>>You cannot create AMI/ Bundle instance in the Instance- store backed directly. If you want to create the instance then there are some steps to follow that can be done using AWS-CLI
>>>You can create AMI / Bundle instance in the EBS- Store backed directly

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/creating-an-ami-instance-store.html

Creating an AMI from an instance store-backed instance.



EC2 OPTIMIZED INSTANCES AND ENCHANCED NETWORKING

EC2- EBS optimized instances

It is used for the full use of the EBS volume's provisioned IOPS.

Some blah blah theory..... Usually the Network traffic and EBS traffic is shared in the Amazon EC2 instance, which means consistent EBS performance depends on amount of non-EBS related

network you send to and from your instance. Since you cannot predict this there will be contention between EBS and internet traffic and at times, traffic to EBS can suffer a lot. To solve this problem and improve the performance, AWS introduced EBS-optimized instances. It is an optimized configuration stack that provides additional and dedicated capacity between EC2 and EBS IO. This optimization minimizes the contention between EBS I/O and other traffic from your Amazon EC2 instance and thereby gives you best and consistent performance. EBS-optimized instances deliver dedicated throughput to EBS, with options between 500 Mbps and 1,000 Mbps, depending on the instance type you use and enables you to effectively use PIOPS volumes (if attached). This is additional throughput and doesn't affect other general purpose network throughput already available on the instance. EBS-Optimized instances can be used with both standard and PIOPS volumes.

PIOPS Volumes + EBS-OPT Instances: Performance-sensitive production databases that require dedicated EBS traffic and high IOPS performance. It is recommended to always use PIOPS with EBS-Optimized instances to get the full performance benefits.
Things to note:

- Not all Amazon EC2 instances are EBS-Optimized. Only the following list is currently supported, please refer AWS documentation for the latest supported types. High-CPU Extra Large (c1.xlarge) ,M1 Large (m1.large), M1 Extra Large (m1.xlarge), High-Memory Double Extra Large (m2.2xlarge), High-Memory Quadruple Extra Large (m2.4xlarge), M3 Extra Large (m3.xlarge) ,M3 Double Extra Large (m3.2xlarge).
- Provisioned IOPS volumes may not be available in every AZ inside a Region where EBS-OPT is available (currently). AWS will sort this going forward. In mean time, you can use an EBS-Optimized instance with Standard EBS volumes as well to get dedicated throughput between your instance and volumes(in such AZ).

Note:

- EBS volumes are not in the physical host / they are not like the instance-store/ they are not the part of the hard drive. Consider that we have EC2 instance in physical host and a solid state driver in the server that is common to multiple ones. Instance will take the virtual part of that solid state of the server.
- EBS connected to the AWS infrastructure. Better performance is achieved between the EBS and the EC2. This is useful for the mission critical database/ application. "**They provide the dedicated performance between EC2 instance and EBS volumes**"
- All the EC2-EBS Optimized instances are designed to work with all EBS volume types

---

EC2- Single Root I/O Virtualization (SR-I/OV) [ENCHANCED NETWORKING]

- Usually we will have the physical server and the hypervisor will emulate the NIC (virtual Network Interface Card). There will be one or more instances over a single physical server which contains NIC, CPU, Memory, Storage etc. The virtual part of all

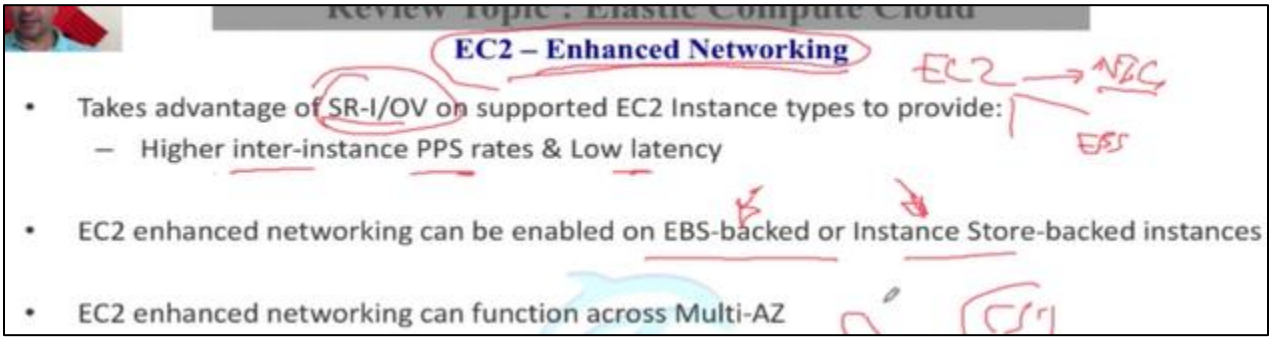
the physical components will be divided equally between the **GUEST OS** that is present in all the instances. The Guest OS will think that it is having its own physical NIC/CPU/memory etc.

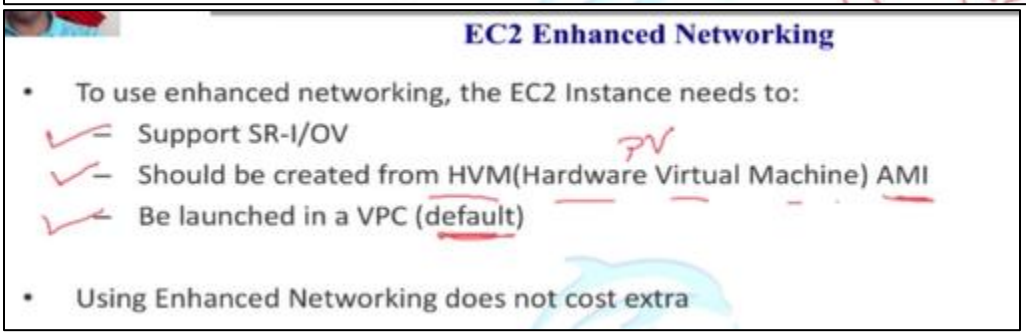
- **SR-I/OV** is designed where the instance would virtualize the NIC without the emulation on the hypervisor. OS will have direct access to the physical NIC and not through the hypervisor. So that the speed and the performance between the NIC card and the instance will increase. Instance has the faster and **low latency** access to the NIC cards [Network]
- We know that the EBS volume is connected through the network. This implies that they can have the **higher performance** and optimizing instance performance while it is writing and reading from the EBS volumes.
- The application that is installed on the EC2 server gives the user a better performance that means the overall enhancement for the user experience for both applications and database
- There is no additional charge for using the enhanced networking

It provides high performance networking capabilities on some of the instance types.

Advantages:

1. Higher packet per second (PPS) performance for data transfers
2. Low Latency
3. Very low network jitter





EC2 Placement Groups

It is the logical grouping (clustering) of the EC2 instances in the **same AZ or different AZs,** which provide you low latency and high networking throughput for the inter- instance communication

- Placement Group determines how the instances are placed in the underlying hardware

- If you create the placement group (clustering) among 4 or 5 EC2 instance that is launched at the same host. Then are you running into the risk??? yes. If the host fails, then the entire placement group will fail.
- You can create the placement group by specifying one of the two strategies.
    - **Cluster-** Clusters the instance into the low latency group into the *single* Availability Group
    - **Spread-** spreads the instances across the underlying hardware in *multiple* AZs.

No Charge for creating the placement group and you will pay only for the instances that is running inside the placement group.

- Use the SR-I/OV enhanced networking instances for creating the placement groups.
- To guarantee the availability, try to launch the required instances at the same time.
    - Assume that you have launched 4 or 5 instances at the same time and when you try to launch the 6th instances after some time. The instance may / may not have the capacity to launch.

    Then what is the result??? You can stop all the old 5 running instances and try to launch the 6 instances at the same time. Here AWS will search for the host / facility that can able to accommodate all the 6 instances…

CLUSTER PLACEMENT GROUPs

- Present in single AZ
- All the instances will be in the same physical host
- Low latency, high packet per second network
- Choose the instance type that supports the enchanced networking
- Recommended when your application need the low network latency high network throughput or both. **If the majority of the network traffic is between the instances in the group**
- If you stop an instance in the placement group , then try to start it . Then the instance will be in the same placement group

SPREAD PLACEMENT GROUPs

- Present in multiple AZ
- The instances in that are in the different underlying hardware in different availability zone will be grouped together.

Eg) if all the instances are in the same host, then if the host fails then the entire instances will fail. So now we are launching the **small number of critical instances** in the different underlying hardware in the different availability zone. So that if the host fails then the other instances will be running perfectly

- **There should be maximum seven EC2 running instances in the single availability zone**
- If you start an instance in the spread placement group and there is insufficient unique hardware to fulfil the request , then the request fails .
    - Try to request after some time

EC2 Status Checks and Monitoring

The authority for creating **EC2 instance is given by the EC2 Service** that is maintained by AWS (both hardware and software).

This service will do automated health check in every one minute (by default)

AWS EC2 service will monitor the status of the EC2 instance. it will return either true or false.

- used in *Auto Scaling group* – if any status failed. The Auto scaling group will label that instance as “impaired”, terminate that instance and launch a new instance
- used in *Cloud Watch* – It will reboot / recover the impaired EC2 instance

There is no configure, delete, disable, change option in the status checks because the status Check is built inside the EC2 Service

- When the instance is impaired because of the host hardware / software problem, if it is a EBS backed instance then you can schedule start/ stop and relocate the instance to another working host. This cannot be done in the instance store
- U can also do this by stopping/ starting the instance manually

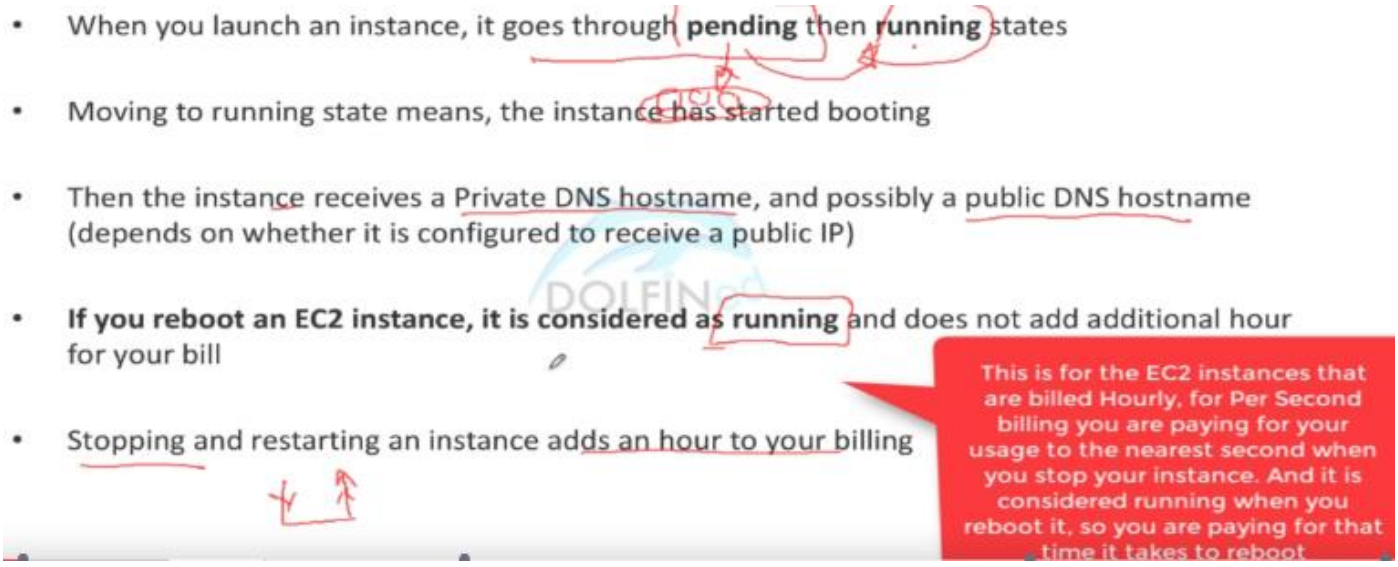
This EC2 service will send the metrics (CPU utilization, memory utilization) to the cloudwatch for every 5 minutes (by default *basic monitoring*). Monitor the entire AWS infrastructure You can set the alarms for the cloud watch.

The monitoring can be set to 1 minute and it will be charged accordingly.[*detailed monitoring*]

Using CloudWatch:

- You can start , restart , terminate, recover the EC2 instance
  - Start and terminate – save cost
  - Restart and recover – move the instance to another host

EC2 Instance States and termination



EC2 Stopped State

- Stop the instance, AWS will shut down the instance but remember the **Instance ID and Root volume**
- Instance Store backed instances **cannot** be stopped, they can only be rebooted and terminated.
- No charge for the stopped instance, but charged for the attached EBS volumes.
  So what can we do for not charging?? When u stop the instance … detach the EBS volume and root volume. When start the instance, you can attach the EBS volumes and root volume.

You have an issue with the EBS volumes in the stopped EC2 instance. What you will do?? Detach that EBS and attach it to another EC2 instance and make required change in that EBS volume and re-attach the EBS volume to the stopped EC2 instance.

EC2 Stopping an EC2 instance

- When you stop the EBS backed instances, the instance Store volumes data is lost
- *When you stop an EBS-Backed EC2 instance*
  - Instance performs shutdown
  - State changes from running- stopping- stopped
  - EBS volume remain attached to the instance
  - Any data cached in the instance – store and RAM will be deleted
  - Mostly when we restart the instance, The instance will be launched in another physical host
  - Instance retains the private IPV4 and IPV6 address
  - Instance also retains the Elastic IP address
  - Instance releases the public IPV4 address and send them to the AWS pool
- Elastic load balancer will divide the load / Traffic between the group of instances. It will do health check each time to check whether the instance is working fine or not. If the instance is stopped na … again the ELB will check the stopped instance and checks whether the state converted from stopped to running
- If your instance was registered with the ELB , It is recommended to de-register the instance from the ELB so that it will stop doing health check for those instances.
  - You can re- register it later when you restart it.
- If you have the instance that is a part of the Autoscaling group, ASG would mark the stopped instance as unhealthy, terminate it and replace it

EC2 Reboot

- Rebooting the EC2 instance does not add up an hour billing

- **Use EC2 reboot and not instance's operating system reboot**
    - Because AWS when it initiates the reboot, it wait for 4 min, the instance did not reboot , it ill force for hard reboot
    - AWS REBOOT creates AWS cloud trail log, which is useful for forensics,

Trouble shooting, documentation / audit purpose.

EC2 Instance Termination

- Running **to** shutting down **to** termination [No charge for shutting down / termination]
- EBS root volume that is created automatically when creating the instance are automatically deleted when instance is deleted. If the root volume is created manually then the it will not be deleted when the instance is deleted
- U can modify the above behavior by selecting **"Delete on termination"** option in any EBS volume when creating the instance/ Running the instance

Note :

Delete on termination- True- Root volume (by default)

- False – data volume (by default)

The default values can also be changed.

- You can view the EBS root volume **Delete onTermination** behavior from "Block Device Mapping". Not the instance store

EC2 Termination Protection

- **EC2** instance cannot be terminated accidentally through the API, CONSOLE, CLI(sdk)
- The **termination protection** can be enabled for both instance store and the EBS backed instances
- Cloud watch will only delete the instances that **do not** have the termination protection enabled
- If you want to terminate the instance that has termination protection turned on, **you can do so by choosing OS shutdown and Configure AWS to treat OS shutdown as instnace termination**
- This can be configured during the launch / when the instance is running / instance is stopped (if EBS Backed instance)

Trobleshooting – Instance Immediate Termination

- Usually when you launch the instance , the status will go from pending **to** running. Sometimes it may go from Pending **to** termination state

- Possible reasons that a launched instance immediately terminates are:
  - The instance store-backed AMI you used to launch the instance is missing a required part.
  - You've reached your EBS volume limit.
  - An EBS snapshot is corrupt.

- To find the reason of the termination:
  - From AWS Console : Go to Instances (select the instance) -> Description tab -> State Transition reason
  - From CLI use the "describe-instance" command

EC2 – Instance Metadata and User Data

- Instance Meta Data:
  - This is instance data that you can use to configure or manage the instance
    - Examples are IPv4 address, IPv6 address, DNS Hostnames, AMI-ID, Instance-ID, Instance-Type, Local-hostname, Public Keys, Security groups...
  - Meta data can be only viewed from within the instance itself
    - i.e you have to logon to the instance
  - **Meta data is not protected by encryption** (cryptography), anyone that has access to the instance can view this data
  - To view an EC2 Instance's Meta Data (from the EC2 instance console):

GET http://169.254.169.254/latest/meta-data/

OR

Curl http://169.254.169.254/latest/meta-data/

To view a specific metadata parameter, example to view local hostname

GET http://169.254.169.254/latest/meta-data/host-name/

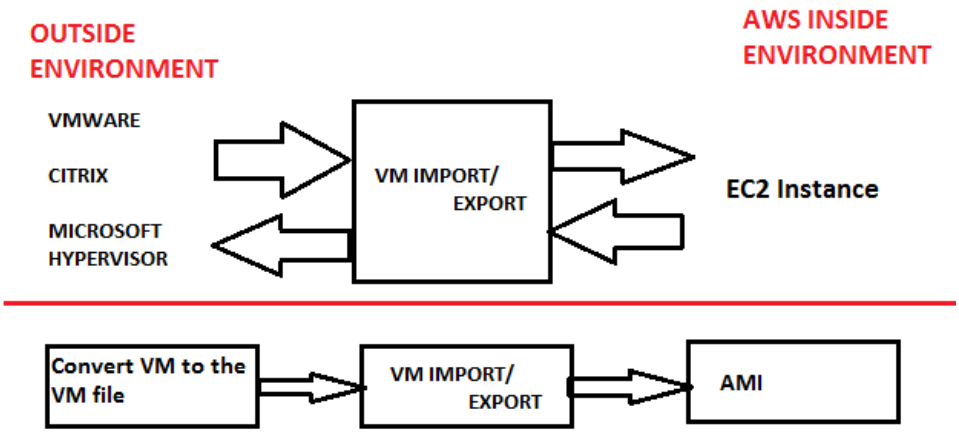
- Instance user data:
  - Is data supplied by the **user at instance launch** in the form of a script to be executed during the instance boot
  - **User data is limited to 16KB**
  - User data can only be viewed from within the instance itself (logon to it)
  - **You can change user data**
    - To do so, you need to stop the instance first (EBS backed)
      - Instance -> actions -> Instance-settings -> View/Change user data
  - User data is not protected by encryption, do not include passwords or sensitive data in your user data (scripts)
  - You are not charged for requests to read user data or metadata

EC2 Migration – VM Import/ Export

Migration to/from AWS EC2 & VM Import/ Export

How to import the virtual machine from outside to the EC2 instance ?? / How to export the virtual machine from EC2 instance to Outside?? LET US SEE HERE

- VM Import/ Export acts as the converter

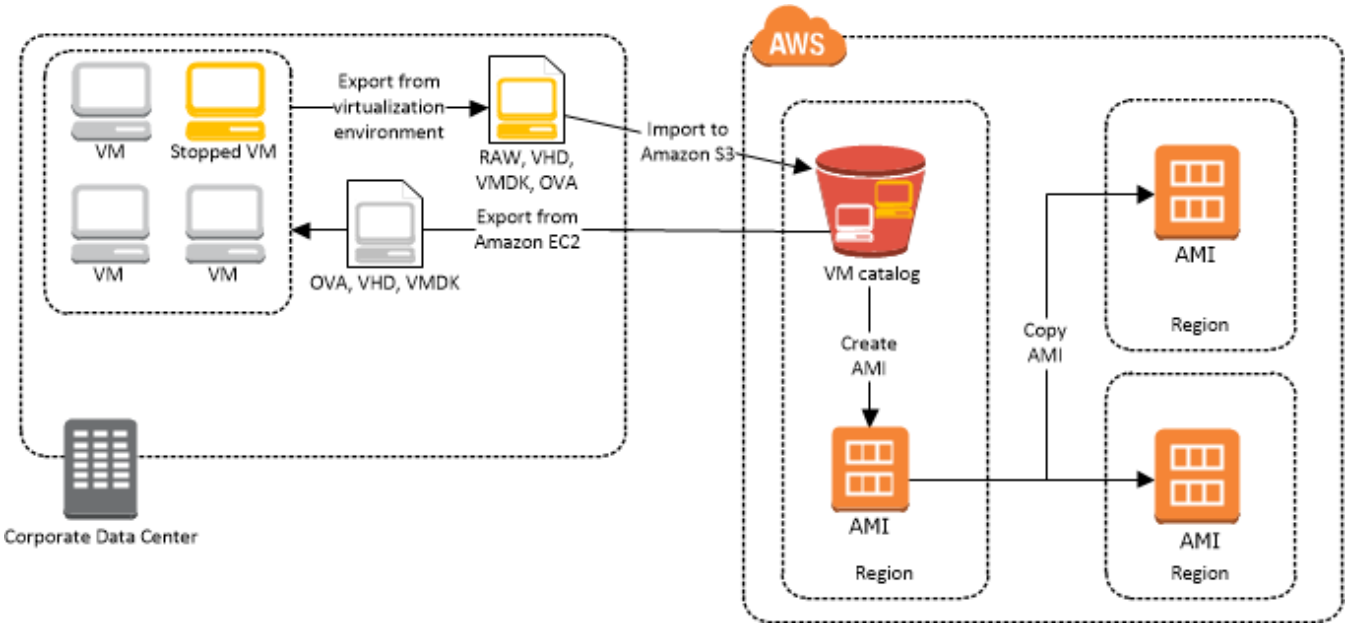


How can I host the VMWARE virtual machines in the EC2 instance environment?

- VM Import/ Export can be used to migrate VM ware, Microsoft , XEN VMs to the cloud (Import)
- VM Import/ Export can be used to convert EC2 instance to VM ware, Microsoft or XEN VMs to on-premise (Export)
- *The VM Export is strictly for the EC2 instances that are originally imported through the VM import and now we want to export them again and use them on-premise. It does not apply for native EC2 instances on AWS that are created from the AWS based AMIs.*

- This supports:
  - Windows and Linux VMs
  - Vmware ESX VMDK [Virtual Machine Disk] (and OVA [Open Virtualization Appliance] images for export only)
  - Citrix XEN VHD [Virtual Hard Disk]
  - MicroSoft Hyper-V VHD

- VM Import/Export is supported through API or CLI, but NOT through AWS Console
- Before generating the VMDK or VHD images, make sure the VM is stopped and not in suspended or paused states

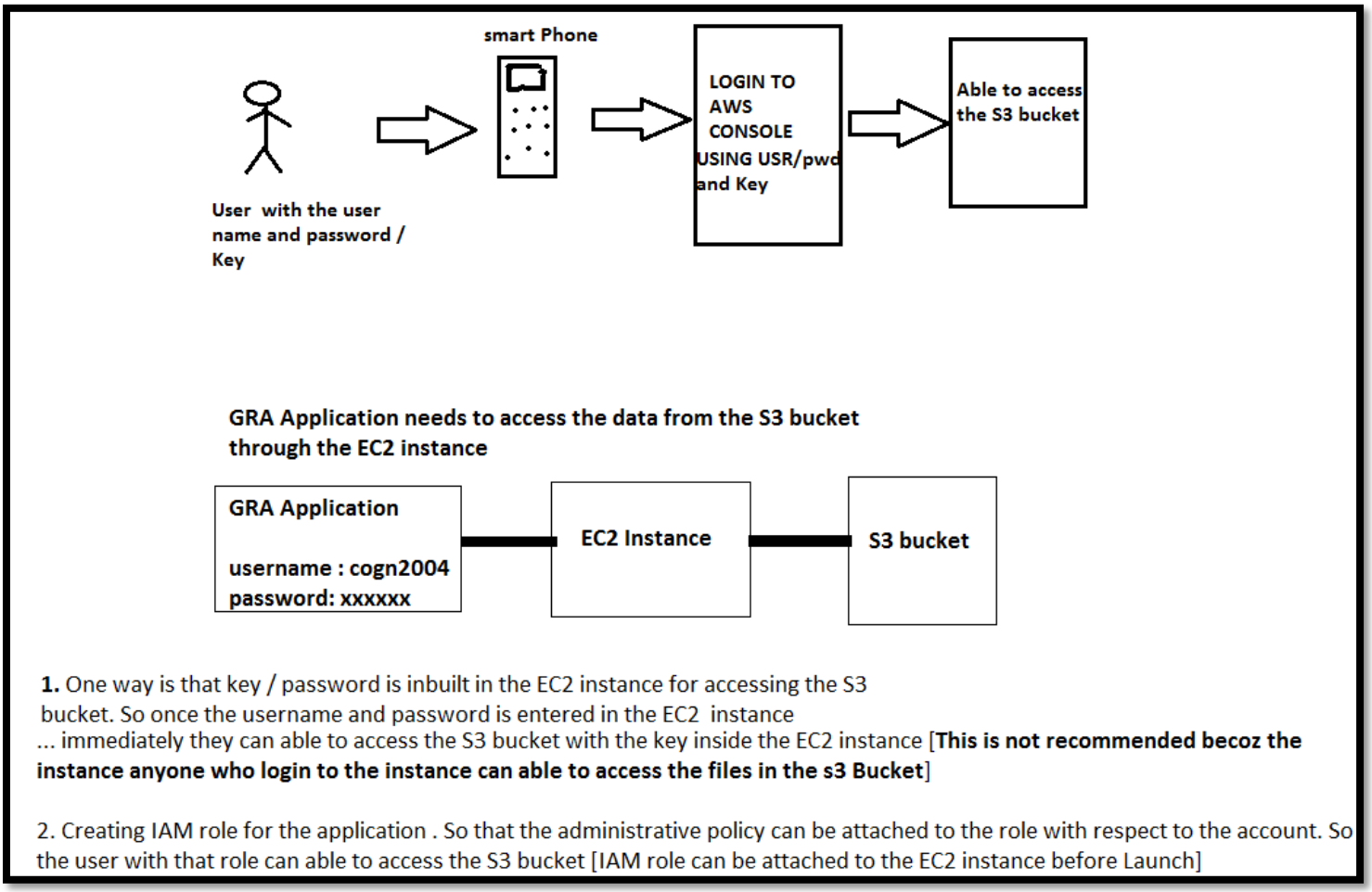


– For vmware, AWS has a VM Connector which is a plugin to vmware vCenter
  - This allow the migration of VMs to AWS S3
  - Convert it to EC2 AMI
  - And progress can be tracked in vCenter

VMware Vcenter is the brain / heart of the virtualisation environment. It does the management and control activity for the VMware Vsphere

EC2 IAM Roles

- For an EC2 instance to have access to other AWS services (example S3) you need to configure a IAM Role, which will have an IAM policy attached, under the EC2 instance.
  - Applications on the EC2 instance will get this role permission from the EC2 instance's metadata

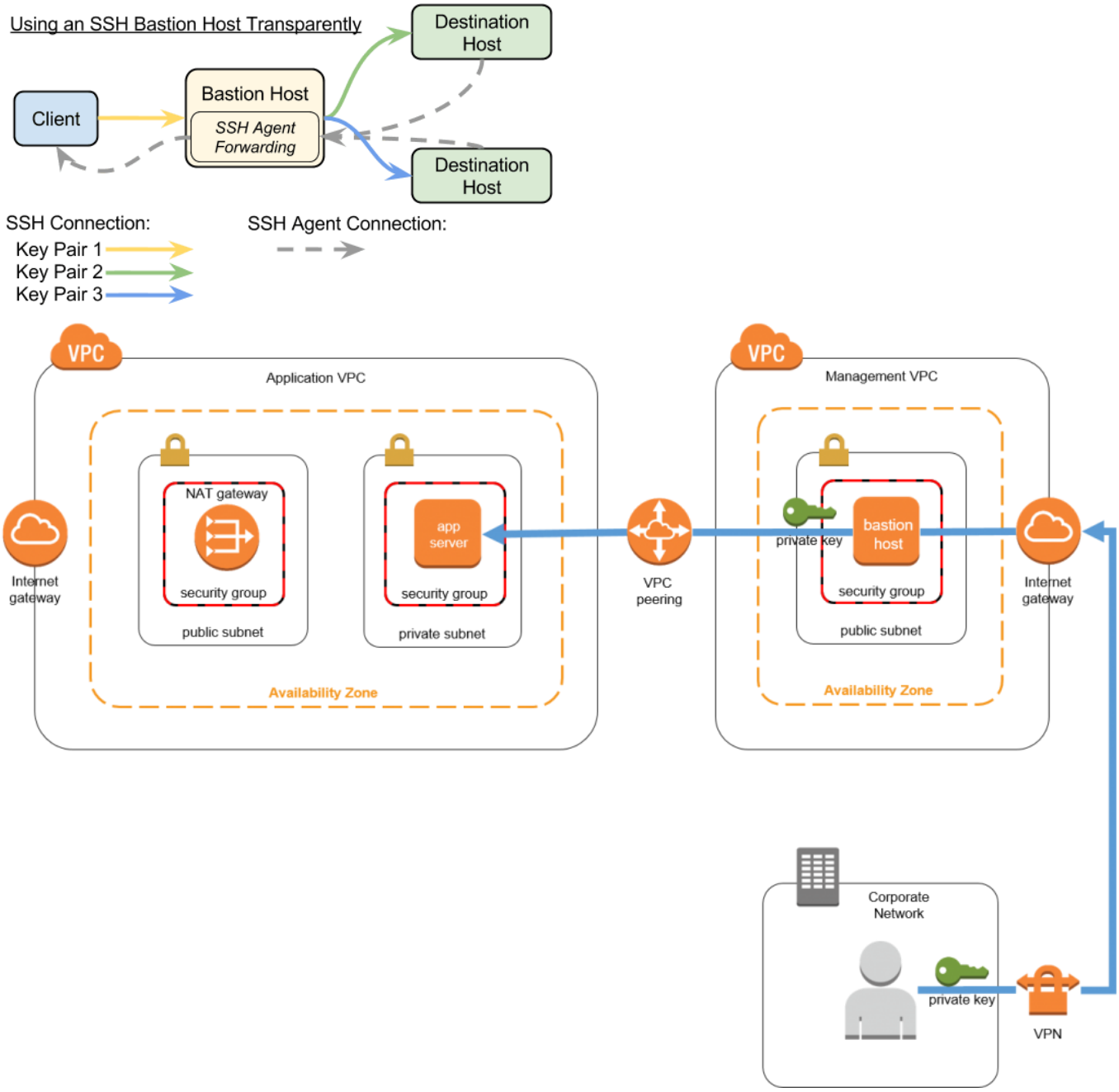


EC2 Bastion Hosts

You have administer 20 EC2 instances that are in the different availability zone in the region. The instnaces may be in the public subnet / private subnet. Remember that you are responsible for the security , patching, OS Configuration for the EC2 instances that you launch .

You are not responsible for any services that are fully managed by the AWS eg) AWS RDS.

- U can use one common key for the 20 instances that you are administering.
  - Security is main .. So that no other hackers can access the instances. Configure the security group and determine who can SSH into the instances
- If you are working in the company. You know the IP address range of the company. U can set up one or 2 EC2 administrative instances [**EC2 Bastion hosts]** where you can access the instances only in your corporate with that key pair. After login into the Bastion host, U can login/ access the individual EC2 instances


Using an SSH Bastion Host Transparently
Destination Host
Client
Bastion Host
SSH Agent Forwarding
Destination Host
SSH Connection:
SSH Agent Connection:
Key Pair 1
Key Pair 2
Key Pair 3
VPC
Application VPC
NAT gateway
security group
public subnet
app server
security group
private subnet
Availability Zone
Internet gateway
VPC peering
VPC
Management VPC
private key
bastion host
security group
public subnet
Availability Zone
Internet gateway
Corporate Network
private key
VPN

EC2 – Bastion Hosts for Linux

(Now AWS calls it Remote Desktop for Windows Instances)

- For inbound, secure, connectivity to your VPC to manage and administer public and/or private EC2 instances, you can use a bastion host (or a jump box/stone).
  - The Bastion host is an EC2 instance, whose interfaces will have a security group allowing inbound SSH (for Linux EC2 instances) or RDP for windows instances
  - Bastion hosts can have auto-assigned public IP addresses or Elastic IP addresses (Elastic IPs are better for security reasons and to fix the IP address)
  - Using Security groups you can further limit which IP CIDRs can access the Bastion Host.
  - Once logged to the Bastion host, you can connect via RDP (Windows) or SSH (Linux) to the EC2 instance(s) you desire to manage