**ELASTIC COMPUTE CLOUD(EC2)**

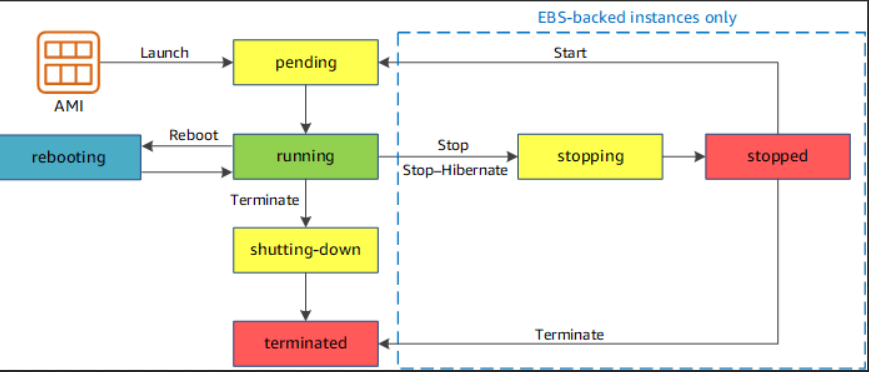
* Amazon Elastic Compute Cloud (EC2) is webservice from **AWS that provides resizable compute services in the cloud**
* They are resizable because you can quickly scale up or scale down the number of server instances you are using if your computing requirements changes
* Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity both up and down as your computing requirements change.
* EC2 is reginal

**Instances**

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

**EC2 instance status**

|  |  |  |  |
| --- | --- | --- | --- |
| **AMI** | **Launch** | **pending** | **running** |
| **Running** | **Reboot** | **Rebooting** | **running** |
| **Running** | **stop** | **stopping** | **stopped** |
| **Running** | **Terminate** | **shutting-down** | **terminate** |
| **stopped** | **start** | **pending** | **running** |
|  |  |  |  |



**PRICING MODEL IN AWS**

**1.On Demand Instance**

* Fixed price hourly based
* *“****Pay what you have used****”*
* No Commitments and No upfront payments
* No Predictable usage

On Demand pricing is used for

Users that want low cost and flexibility of Amazon E2 without any up-front payment or long-term commitment

Application with short term, spiky or unpredictable wok loads that cannot be interrupted

Applications being developed or tests on Amazon EC2 for first time

**2.Reserved Instance (RI)**

* Provides you with the capacity reservation, and offer a significant discount on the hourly charge for an instance.
* Long Term Commitment (Contract term Min 1 or 3 years)
* Upfront payment (Full, Partial)
* 75% discount on hourly price
* Predictable Usage

**Reserved pricing types**

* **Standard RI :** These offers up to 75% off on demand instances, The more you pay upfront and the longer contract, the the greater the discount
* **Convertible RI :** These offers up to 54% off on demand capability to change the attributes of RI as long as the exchange results in the creation of RI of equal or greater value
* **Schedule RI :** These are available to launch with in the time window you reserve. This option allows you to match your capacity reservation to predictable recurring schedule that on requires fraction of a **day, week or a month.**

**3.SPOT Instance**

* Enable you to bid where price you want for instance capacity, providing for even greater saving if your applications have flexible start and end times
* Huge Capacity with cheaper price (90% discount)
* Application that has flexible start and end time

**Dedicated Host**

* If any customer need dedicated physical machine or a Host
* License based software and more cost
* Dedicated host means, we need to purchase the whole hardware machine and can we convert it to multiple small EC2 since we own an entire Host

**Dedicated instance pricing is used for**

* Useful for regulatory requirements that may not support multi-tenant virtualization
* Great for licensing which does not support multi-tenancy or cloud deployments.
* Can be purchase on demand hourly. Can be purchase as a reservation for up to 70% of the on-demand price

**TYPES OF EC2 COMPUTING INSTANCES**

* Computing is a very broad term, the nature of your task decides what kind of computing you need
* Therefore, AWS offers many types of instances which are few follows

**General purpose**

* For applications that require a balance of performance and cost
* All kind of application we can use general purpose ec2 instance

**Eg** email responding system, where you need a prompt response as well as the it should be cost effective, since does not require much processing

***t2, m4, m3***

**Computed Optimized**

Idle for compute bound applications that benefits high performance

* Well suitable for batch processing workloads
* Media transcoding
* Hight performance web servers
* High performance computing

***c4,c3***

**Memory Optimized**

* For applications that are heavy in nature, therefore, require a lot of RAM
* **Example**: Multitasking applications ***r3,x1***

**Accelerate Computing**

* Instances use hardware or co-processors to perform functions, such as floating-point number calculation graphic processing or data pattern matching more efficiency than is possible in software running on CPU’s

**Storage Optimized**

* Instances are designed for workload that require high sequential read and write access to very large data sets on local storage.

***i2,d2***

**GPU Instances**

* For applications that require some heavy graphics rendering
* Eg 3D modelling etc.

***G2***

* t2.nano = 0.5GB RAM, 1CPU
* t2.micro = 1.0GB RAM, 1CPU
* t2.small = 2.0GB RAM, 1CPU
* t2.medium = 4.0GB RAM, 2CPU
* t2.large = 8.0GB RAM
* t2.xlarge = 16GB RAM
* t2.2xlarge = 32GB RAM
* Scalability can be achieved by changing the instance type
* Any Time scale up and down -> No Data Lost
* We should STOP the EC2 Instance to change the instance type
* If you change the instance type, data in the instance will not be lost.

|  |  |  |  |
| --- | --- | --- | --- |
| **Instance type** | | | |
| **nano** | **xlarge** | **9xlarge** | **32xlarge** |
| **micro** | **2xlarge** | **10xlarge** | **52xlarge** |
| **small** | **4xlarge** | **12xlarge** | **metal** |
| **medium** | **6xlarge** | **16xlarge** |  |
| **large** | **8xlarge** | **24xlarge** |  |

**\*\*\*\*\*BURSTABLE PERFORMANCE INSTANCES(CPU CREDITS)\*\*\*\*\***

* T2 and T3 instances are burstable instances, meaning the CPU performs at a baseline, say 20% of its capacity
* When our application needs more than 20% of the performance of the CPU, the CPU enters in a burst mode giving higher performance for a limited amount of time, therefore work happens faster
* EC2 Instance will enter into burstable mode and give high performance for limited period of time only
* Applicable for Only CPU credits

**VOLUMES**

Amazon Elastic Block Storage provides persistent block storage volumes for use with Amazon EC2 instances in the AWS Cloud.

Each Amazon EBS volume is automatically replicated within its availability zone to protect you from component failure, offering high availability and durability.

* Volumes are 2 types
* EBS Volumes
* Instance Store Volumes
* **EBS VOLUMES**
* Persistence Storage/Permanent
* If you STOP and START the EC2 Instance, ***DATA will not be LOST***
* EBS Volumes max size 16TB
* EBS is chargeable
* If you Reboot data will not be LOST

**TYPES OF EBS VOLUMES**

* **General Purpose(gp2,gp3) - SSD** = General Purpose(Min 1GB to Max 16TB)
* **Provisioned IOPS(io1,io2)** **- SSD** = High Performance (Min 4GB to Max 16TB)
* **Throughput Optimized(st1) – HDD** = Frequently accessed data with cheaper price (Min 500GB to 16TB)
* **Cold (sc1) – HDD** = Not Frequently accessed data cheaper price (Min 500GB to 16TB)
* **Magnetic (Standard)** – HDD = Previous generation ( 1GB to 1TB)
* **IOPS** = Input Output Per Seconds
* (io1,io2 and gp3) are IOPS configurable = you can input IOPS values
* **#gp2** has default IOPS=1:3 =1GB:3IOPS 🡪 IOPS not configurable
* **#gp2** is the default volume type
* **#Root** Volumes support these volumes types (gp2, gp3,io1,io2 and standard)
* **#Root** Volume doesn’t support st1 and sc1
* Additional volumes support ALL TYPES

**EBS-OPTIMIZED INSTANCES**

* C4, M4, and D2 instances, are EBS optimized by default, EBS means Elastic Block Storage, which is a storage option provided by AWS in which the IOPS\* rate is quite high.
* Therefore, when an EBS volume is attached to an optimized instance, single digit millisecond latencies can be achieved.
* \*IOPS (Input/Output Operations Per Second, pronounced eye-ops) is a performance measurement used to characterize computer storage devices.

**INSTANCE STORE VOLUMES**

* An instance store is a temporary storage type located on disks that are physically attached to a host machine
* Not persistence
* Temporary Volume
* If we STOP and START the EC2 instance, ***DATA will be LOST***
* Instance store volumes are Free
* Also called as “***Ephemeral Storage*** “

**NOTE:**

* If we Terminate EC2 instance, by default ROOT volume will also be deleted because “***delete on termination”*** option is enabled/checked
* If we Terminate EC2 instance, by default Additional volume will NOT BE DELETED, because “***delete on termination***” option is disable /un-checked
* These check box can be customized depends on the requirement check or un-checked.

**SNAPSHOTS**

* Snapshot is a *“****point in time copy of the volume”***
* Which is lazily copied to Amazon Simple Storage Service (Amazon S3)
* Backup of the volume is also called snapshot
* EBS snapshots are created from EBS Volume
* You can create snapshots from the volumes
* EBS Volume -> EBS Snapshot -> EBS Volumes
* You cannot attach the snapshots directly to EC2 instance, you have to create volumes out of snapshot first and then attach the volume to the EC2 instance
* *Snapshots are stored in S3(Providers S3)*
* Is it possible to login or to use snapshot directly -> **NO**
* Snapshots does not have any availability zones (AZ)
* Snapshots are reginal
* By default, snapshots are private, if required we can make it public
* We can copy snapshot from one region to another region in the same account
* Snapshot can be shared from one account to another account (Private)
* EBS volumes cannot be moved directly to another region
* EBS snapshots are incremental copies of data. This means that only unique blocks of EBS volume data that have changed since the last EBS Snapshot are stored in the next EBS snapshot
* EBS snapshots are incremental copies of data. This means that only unique blocks of EBS volume data that have changed since the last EBS snapshot are stored in the next EBS snapshot

**NOTE:**

* ***EBS Volumes are created from EBS Snapshots***
* ***Instance store volumes are created from a template stored in S3***
* To create ***SNAPSHOT***, we need to STOP the instance
* By defaults Volumes are not encrypted
* All encryption keys are stored in KMS (Key Management Service)
* KMS is completely managed by AWS

**AMAZONS MACHINE IMAGE (AMI)**

* AMI’s are templates of OS and they provide the information needed to launch an instance
* Copy of Operating System is called ***images***
* Image = AMI = Amazon Machine Image
* Images includes all configuration that we did on original instance
* EC2 instance 🡪 image (AMI) 🡪 EC2 Instance
* 1 AMI, can be used multiple times to launch multiple EC2 instances
* AMI’s are re-usable
* AMI does’t have any AZ’s
* We can’t directly use AMI to login, instead launch EC2 instance from the image and then login to instance
* By default, AMI are private, and locked for your account /region
* AMI’s is reginal
* We can make it AMIs public and share them with other AWS account or sell them on the AMI Market place
* AMI’s can be copied from one region to another region
* AMI’s can be shared to one account to another account using AWS account id’s
* *All public images are located at market place*
* Image contain OS Root Volume/instance store volume
* AMI can be found and published on the Amazon Marketplace
* Images are backed by either EBS Volume to instance store Volumes

**Note:**

Snapshot 🡪 copy of Volume

Image 🡪 Copy of the entire EC2 (Includes Volumes)

* If we customize the app on OS, and take the image is called
* ***CUSTOMIZE AMI OR GOLDEN AMI (Automated)***

**CLUSTER NETWORKING INSTANCE**

X1, M4, C4, C3, I2, G2 and D2 instances support cluster networking. Instances launched into a common placement group are put in a logical group that provides high-bandwidth, low latency between all the instances in the group.

Three Types of Placement Groups

**CLUSTER PLACEMENT GROUP**

* A Cluster placement group is a grouping of instances with in a single availability zone. Placement groups are recommended for applications that need low network latency, high network throughput or both.
* Only certain instances types can be launched in to a Clustered Placement Group
* Whenever launch a new EC2 instance, the EC2 service will place the instance such a way that all your instances are spread out across different hardware’s
* Grouping the servers in same /single AZ / Same rack and we will get more performance and low **HA (High Availability)**
* Placement groups recommended to the homogenous instances
* We can add the instances to the place group any time
* **SPREAD PLACEMENT GROUP**
* EC2 Instance are spread across multiple AZ’s,
* A spread placement group is a group of instances that are each placed on distinct underlying hardware.(Max 7 instances per group per AZ)
* Spread placement group are recommended for applications that have a small number of critical instances that should be kept separate from each other.
* We will use for critical applications
* We will get more ***High Availability***
* Per Availability Zone (AZ) = Max 7 Instances

**PARTITION PLACMENT GROUP**

* EC2 Instance are spread across multiple AZ’s
* spreads instances across many different partitions (which rely on different sets of racks) within an AZ. Scales to 100s of EC2 instances per group (HDFS,HBase, Cassandra, Kafka)
* Per Availability Zone (AZ) = Max 7 Instances
* 1 partition can contain ***100’s of EC2 instance***
* The instances in a partition do not share racks with the instances in the other partitions
* A partition failure can affect many EC2 but won’t affect other partitions

**KEY-PAIR**

* ***Key-Pair*** is used to retrieve password of EC2 Instance
* We don’t have any key-pair by default, we have to create it by ourself
* Key-pair extension is **.pem**
* Key-pair is also called as ***.PEM*** file
* When ever we launch the EC2 instance, the console will ask you to attach key-pair to the EC2 instance
* We can create multiple key-pairs
* 1 key-pair can attach to multiple EC2 instance
* EC2 instance can have 1 key-pair attached at any point of time
* For every EC2 instance, password is different
* Once .pem file is attached, you can’t change the .pem file to the EC2 instance
* Keep the .pem file in SAFE and SECURE it
* Every time you retrieve the password using .pem file, you will get the same password from the EC2 instance
* AWS is public key and Customer has private key(.pem file)
* *Key-Pair = combination of both public + private key*
* Using putty generator tool we will convert .pem file to .ppk file

**Window Instance**

IP Address will be provided by AWS

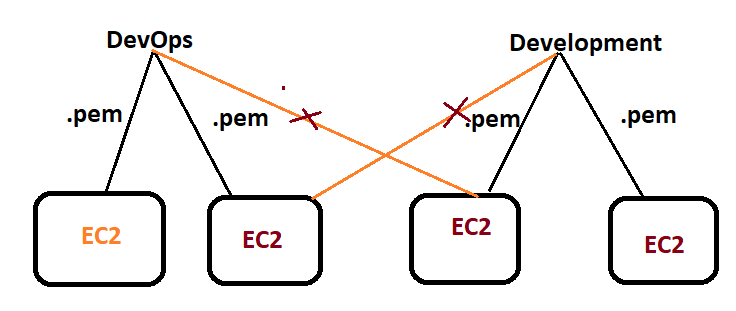
By default, username is Administrator

Protocol is RDP (remote desktop protocol)

**Linux Instance**

IP Address will be provided by AWS

By default, username is ec2-user

Protocol is SSH

**SECURITY GROUP**

* Fire wall which stops un-authorized access to the network
* Security Group acts as a firewall to control inbound and outbound traffic
* Each security group has rules according to which the traffic is governed
* Security Groups are the fundamental of network security in AWS
* Can be attached to multiple instances
* It’s good to maintain one separate security group for SSH access
* If application is not accessible (time out), then it’s a security group issue
* If application gives a ***“connection refused”,*** *then its an application error or its not launched*
* Fire wall = Security Group
* We can’t deny on SG
* Security Group act as a fire wall to secure EC2 instance
* Security Groups stops un-authorized access to the EC2 instance
* Every EC2 instance must have at least one security group (SG) should be attached
* We can create multiple security groups (SG’s) and we can attach multiple SG’s to the single EC2 instance
* AWS has a default Security Group (SG)
* A brand-new Security Group inbound rules are denied and outbound rules are allowed
* If you create any new SG, ***all inbound rules are deny and outbound rules are allowed***
* Security Groups are ***STATEFULL***
* If you allow any inbound rules, you no need to allow that on outbound rule
* You cannot block specific IP addresses using security groups, instead use NACL

Security groups has two types of rules

**Inbound rule**

* Which allowed traffic towards EC2 instance
* By default, inbound rules are denied
* It is not possible to denied any/particular protocol in SG because by default inbound rules are denied
* Every EC2 instance must have at least one security group (SG) should be attached

**State Full**

* If you allow any inbound rule, you no need to allow that on outbound rule

**State Less**

* If you allow any inbound rule, you have to allow that also in outbound rule

**Outbound rule**

* Which sends the traffic outside EC2 instance
* By default, outbound rules are allowed

**Source**

* **Custom** : customize network
* **Anywhere** : for every one can access
* **MyIP** : access only you

**RDP (RemoteDesktopProtocal)**

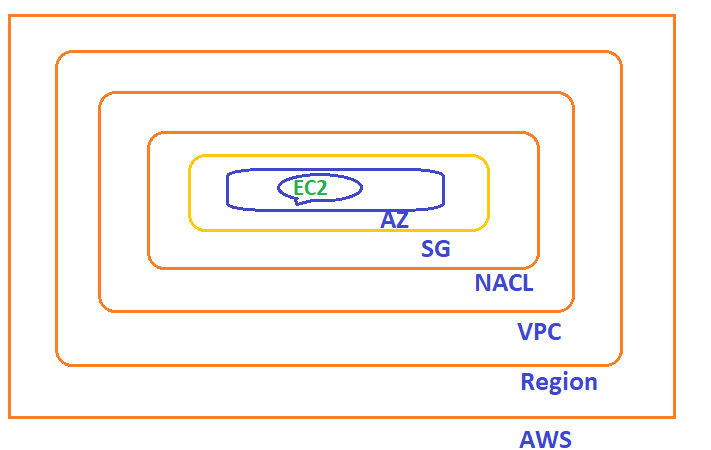
* Login connect remotely to the windows EC2 instance

**SSH:**

* Login and connect remotely to the Linux EC2 instance

**NACL (Network Access Control List)**

* Another layer of security for EC2 instance(if required we can useit)
* If you want to tight the security, go for NACL
* Like SG, NACL also has inbound and outbound rules
* NACL will hit first and then SG
* Isolated network is called as ***SUBNET***
* SG is EC2 instance level and NACL is Subnet level
* We have default NACL
* By default, ***inbound and outbound rules are allowed***
* NACL we can deny and allowed as well
* ***If you create any new NACL, all inbound rules are DENY and outbound rules are DENY***
* NACL are STATELESS
* If you allow any inbound rules, you must allow that on outbound rule as well.



**AWS:**

1. Amazon Web Service (AWS) has Global infrastructure
2. AWS is providing infrastructure as a Service Cloud
3. AWS is a cloud provider who provides infrastructure as a service.

**On-Premises:** Infrastructure handled by own

**CLOUD:**

1. Cloud is present in the remote location
2. We need internet to connect to the cloud
3. “Without buying the hardware as a customer if I can use hardware resource of my Vendor”

**Cloud Computing:**

Instead of doing computing on local machine /on -premises, you will do now computing on remote location (Cloud) that is called **Cloud Computing**

**DEPLOYMENT MODELS (TYPE OF CLOUDS):**

1.**Public Cloud:**

1. The Service which are accessed by everyone is called PUBLIC CLOUD.
2. Infrastructure is managed by cloud vendor

Example: AWS, AZURE, GCP

2.**Private Cloud:**

1. The Service which are accessed with in organization
2. Infrastructure is dedicated to a particular organization
3. More expensive

Example: (1) Apache Cloud Stack (2) OpenStack

3.**Hybrid Cloud**: it is combination of both PUBLIC and PRIVATE CLOUD.

SERVICE MODELS

**1.IAAS (Infrastructure as a service) :**

1.AWS is providing infrastructure as a service

2.here, Application + Data and OS (Operating System) is customer responsible

3.Network+Datacenters (DC) and Virtualization PROVIDER (i.e AWS) is responsible

4.Customer has full access/control on the servers

5.In future any application down customer has to take care.

|  |
| --- |
| Application |
| Data |
| Operating System |
| Virtualization |
| Server |
| Storage |
| Network |

CUSTOMER

PROVIDER

**2.PAAS (Platform as a Service) :**

1.Azure is providing platform as a Service

2, Here, Application + Data is customer responsible

3.Network+Datacenters and Virtualization is PROVIDER is responsible

4.No need to worry about the servers (i.e Customer has no access/control on the servers).

5. In future any application down PROVIDER has to take care.

|  |
| --- |
| Application |
| Data |
| Operating System |
| Virtualization |
| Server |
| Storage |
| Network |

CUSTOMER

PROVIDER

**3.SAAS (Software as Service)**

1.Salesforce is providing software as a service

` 2.Here, everything take care by PROVIDER

3.Network+DC+Virtualization+OS+Data+Application all this thing PROVIDER will take care

Example: Gmail, ZOOM etc

|  |
| --- |
| Application |
| Data |
| Operating System |
| Virtualization |
| Server |
| Storage |
| Network |

PROVIDER