# AWS Cloud Practitioner certification Notes:

**Client —— Network——> Server**
**Ip adress.                    IP adress**
**Cloud Computing:** cloud computing is the on-demand delivery of compute power, database storage, application, and other IT resources.

- **Pay as you go using**
- Get exactly right size and type of resources instantly
- Easy way to access servers, storage, database,& a set go application services.
- Drop box and Netflix built on AWS.

**Deployment Models of the cloud:**
**Private cloud:** Cloud services used by single organization and not exposed to the public, Complete control, More Security, Meet specific business needs.
**Public cloud:** Azure, Google cloud, AWS : Owned and operated by 3rd party service provider delivered over internet. 6 advantages.
**Hybrid cloud (**):** Keep some servers on premise and extend some to the cloud. Control over sensitive assets In private infra and flexibility and cost effectiveness of the public cloud.

**Five characters of Cloud computing :**

- **On demand self service:** users can provision resources and use them without human interruptions
- **Broad network access:** Resources are available over network 7 can be accessed by diverse client platforms.
- **Multi tenancy & resource pooling:** Multiple customer can share same infra with security and privacy.
- **Rapid elasticity and scalability**: quickly acquire/dispose resources when needed and scaling as well.
- **Measured service:** pay for what you use

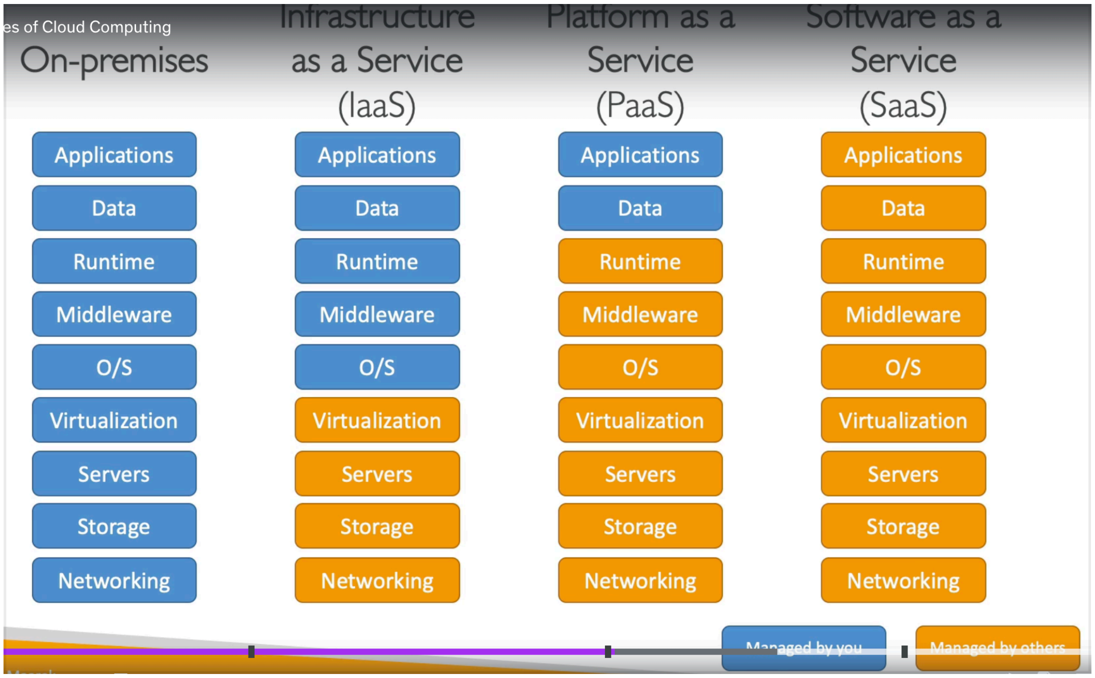
**Six Advantages of Cloud Computing: ****

- No (CAPEx) Capital expense for (OPEX) Operational expense.
- Best from massive economies of scale : Due to its vastness, low prices.
- Don't need prior guessing capacity
- **I**ncreased speed and agility
- Go global in minutes
- No need of money for running & maintaining datacenter.

**Types of Cloud computing:**

- **Infrastructure as a service (Iaas)**
  - Provides building blocks for cloud IT
  - Provides networking, computers, data storage
  - Highest level of flexibility

    - Easy parallel with traditional on-premise IT Ti migrate from on premise to cloud(EC2)
    - Amazon **EC2**, GCP, Azure, Rackspace, Digital ocean, Linode
- **Platform as a Service (PaaS)**
    - Removes the need for your org to manage the underlying platform.
    - Can focus on deployment and management of application
    - Elastic **beanstalk**( AWS), Heroic, Google App Engine(GCP),
    - Windows Azure( Microsoft)
- **Software as a Service:**
    - Complete product that is run and managed by service provider.
    - **Gmail**, dropbox, zoom



es of Cloud Computing

| On-premises | Infrastructure as a Service (IaaS) | Platform as a Service (PaaS) | Software as a Service (SaaS) |
|---|---|---|---|
| 🟦 Applications | 🟦 Applications | 🟦 Applications | 🟧 Applications |
| 🟦 Data | 🟦 Data | 🟦 Data | 🟧 Data |
| 🟦 Runtime | 🟦 Runtime | 🟧 Runtime | 🟧 Runtime |
| 🟦 Middleware | 🟦 Middleware | 🟧 Middleware | 🟧 Middleware |
| 🟦 O/S | 🟦 O/S | 🟧 O/S | 🟧 O/S |
| 🟦 Virtualization | 🟧 Virtualization | 🟧 Virtualization | 🟧 Virtualization |
| 🟦 Servers | 🟧 Servers | 🟧 Servers | 🟧 Servers |
| 🟦 Storage | 🟧 Storage | 🟧 Storage | 🟧 Storage |
| 🟦 Networking | 🟧 Networking | 🟧 Networking | 🟧 Networking |

🟦 Managed by you
🟧 Managed by others

**Pricing of cloud:**
- Compute:
- Strage
- Data transferring OUT of cloud, transfer IN is free.
-

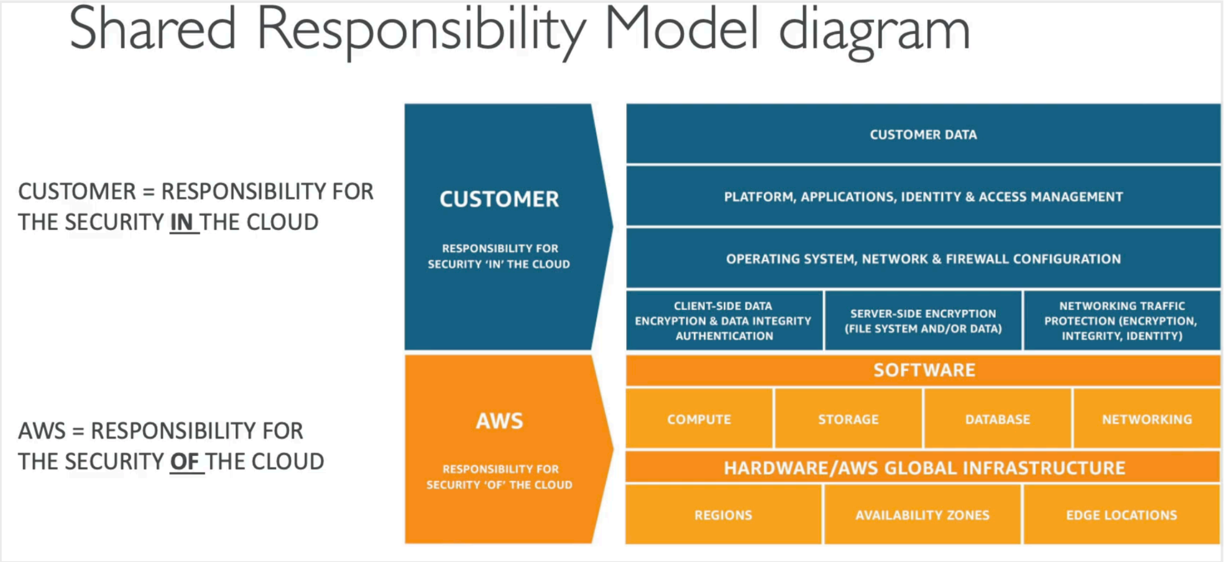
**Global Infrastructure:**
Regions-> availability zones-> damtcenters -> Edge locations/ points of presence.

**How to choose AWS region?**
- Compliance: Data governance and legal requirements
- Proximity to customers: reduce latency
- Available services with in a region: Not all services are available in all regions.
- Pricing varies region by region
- AWS Availability zones:
    - **Each region has AZ( 3 min, 6 max)**
    - AZ is one or more discrete data centers.

    - AZ's in a Region are connected with High Bandwidth, ultra low latency networking.

AWS Console:

- **Global services:**
    - IAM
    - Route 53 (DNS service)
    - **cloudFront ( Content Delivery Network)**
    - WAF ( Web Application Firewall)
- Region Scoped:
    - EC2 ( infra as a service)
    - Elastic Beanstalk ( Platform as  a service)
    - Lambda ( Function as a service)
    - Rekognition ( software as a service) for ML



Shared Responsibility Model diagram

CUSTOMER = RESPONSIBILITY FOR THE SECURITY **IN** THE CLOUD

AWS = RESPONSIBILITY FOR THE SECURITY **OF** THE CLOUD

CUSTOMER

RESPONSIBILITY FOR SECURITY 'IN' THE CLOUD

CUSTOMER DATA

PLATFORM, APPLICATIONS, IDENTITY & ACCESS MANAGEMENT

OPERATING SYSTEM, NETWORK & FIREWALL CONFIGURATION

| | | |
|---|---|---|
| CLIENT-SIDE DATA ENCRYPTION & DATA INTEGRITY AUTHENTICATION | SERVER-SIDE ENCRYPTION (FILE SYSTEM AND/OR DATA) | NETWORKING TRAFFIC PROTECTION (ENCRYPTION, INTEGRITY, IDENTITY) |

AWS

RESPONSIBILITY FOR SECURITY 'OF' THE CLOUD

SOFTWARE

| | | | |
|---|---|---|---|
| COMPUTE | STORAGE | DATABASE | NETWORKING |

HARDWARE/AWS GLOBAL INFRASTRUCTURE

| | | |
|---|---|---|
| REGIONS | AVAILABILITY ZONES | EDGE LOCATIONS |

**IAM**

Identity and access management, Global Service
Root account created by default shouldn't be used or shared.

Users are people in organizations, and can be grouped.
Groups only contain users, not other groups.
Users can stay without any group, also can belong to multiple groups.

**IAM Permissions:**

Users/ groups can be assigned JSON documents called policies.
These policies define the permissions of the users.
In AWS you apply **least privilege principl**e, dont give more permissions than a user needs.

**IAM Policies inheritance:**

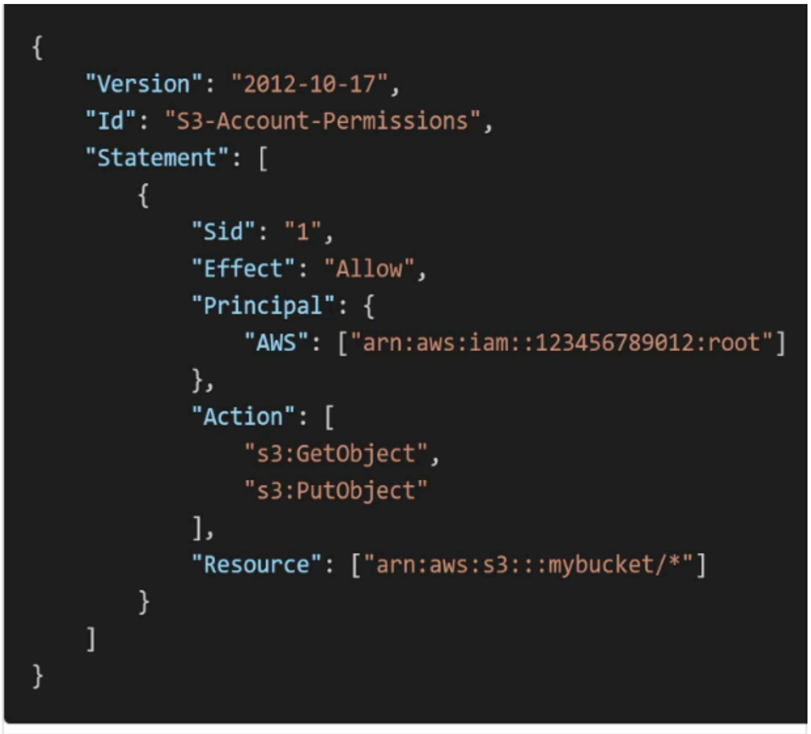
We can attach policy at the group level => Policy would be applied to every single

member of the group.
Now everyone will get access and inherit this policy.

If a user is not in any group we will attach **inline policy** ( that's only attached to a user)

**IAM Policy structure:**

- Consist of JSON file



```
{
    "Version": "2012-10-17",
    "Id": "S3-Account-Permissions",
    "Statement": [
        {
            "Sid": "1",
            "Effect": "Allow",
            "Principal": {
                "AWS": ["arn:aws:iam::123456789012:root"]
            },
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": ["arn:aws:s3:::mybucket/*"]
        }
    ]
}
```

- Version : Policy language version, always include 2012-10-17
- ID: Identifiers for the policy (optional)
- Statements:
  - Sid: identifier for the statement (optional)
  - **Effect** : whether the statement **allows/denies** access.
  - Principle: account/user/role to which this policy applied to.
  - **Action**: **List of actions this policy allows/denies**
  - Resource list: List of resources to which the action applied to.
  - Condition: condition for when this policy is n effect(optional)

**IAM Password policy:**

- Strong passwords = high security
- In AWS, we can set up password policy
- Set a minimum length
- Include Lowe, upper, numbers, special characters
- Allow IAM users to change their- own passwords
- Require them change pls after specific period(90 days)
- Prevent password reuse

**Multi-factor authentication (MFA):**

- We need to protect Root Accounts and IAM users.
- **MFA = Password you know + Security device you own**
- MFA Devices:
    - Virtual MFA device
        - Google Authenticator: (Phone Only)
        - Authy : (phone only) but can be used for multiple applications.
        - Universal 2nd Factor(U2F) Security key: Third party
            - Yubikey by Yubico ( kind of device like pendrive)
            - Support multiple ROOT and IAM Users with single device.
    - Hardware key FOB MFA device ( 3rd party) device.
    - GOV of US: Sure PassID (device) (3rd party)
- Password policy can be found in account settings

**How can users access AWS?**
**There are 3 options**

- AWS **management console** ( protected by **pwd**+**MFA**)
- AWS (**CLI**) Command line interface ( protected by **access keys**)
- AWS **SDK** software development kit - for code: (**access keys**)

- Access Keys are generated by AWS Console
- Users manage their own access keys
- Access keys are secret like PWD
- **Access Key ID = Username**
- **Secret Access Key = Password**

AWS **CLI**:

- A tool that enables you to **interact with AWS service using commands** in command line shell(terminal)
- Its built on the top of AWS SDK for Python.
-

AWS SDK:

- Software development kit
- Language specific API's
- Enables you to access and manage AWS services programmatically
- SDK is embedded within your application.
- Supports multiple languages
- Mobile dk's, not device sdk's

- CLI:
    - Aws
    - aws —version
    - Aws aim list-users

**IAM Roles for services:**

- EC2 instance roles
- Lambda function roles
- Roles for cloudformation

Role is a way to give AWS entities permissions
to do stuff on AWS.

**IAM Credential Report (account-level)**

**IAM Security tools:**

- **IAM Credentials Report (account-level)**
  - Contains all your accounts users and status of their various credentials. Like my root account, IAM account
- **IAM Access Advisor( user-level)**
  - Shows service permissions granted to a user and when those services were last accesses.
  - Can use this info to **revise policies**
  - Very helpful when you want to do granular user access permission on AWS.

**IAM Guidelines and Best Practices**

- Dont use the root account except for AWS account setup.
- One physical user = one AWS user
- Assign users to Groups and assign permissions to groups
- Create strong pwd policy
- Enable MFA
- Cream and use ROLES for giving permission to AWS services
- Use **Access keys** for programmatic Access. ( **CLI/SDK**)
- **Audit permission of your account using IAM Credential Report & IAM Access Advisor**
- Never share IAM users & Access keys

**Shared responsibility model for IAM:**
**AWS:**

- Infrastructure
- Configuration ad  vulnerability analysis
- Compliance validation

**YOU:**

- Users, Groups, Roles, Policies management and monitoring
- Enable MFA on all accounts.
- Rotate all your keys
- USE I’m tools to apply appropriate permissions

- Analyze access pattern & review permissions

**IAM Section - Summary:**

- **Users:** mapped to a physical user, has a password for AWS Console
- **Groups**: contains users only
- **Policies**: JSON document that outlines permissions for users or groups.
- **Roles:** for EC2 instances or AWS Services.
- **Security** : MFA + Password Policy
- A**WS CLI:** manage your AWS services using the command-line
- **AWS SDK**:Manage AWS services using a programing language
- **Access keys:** access AWS using the CLI/SDK
- **Audit:** IAM Credential Reports & IAM Access Advisor

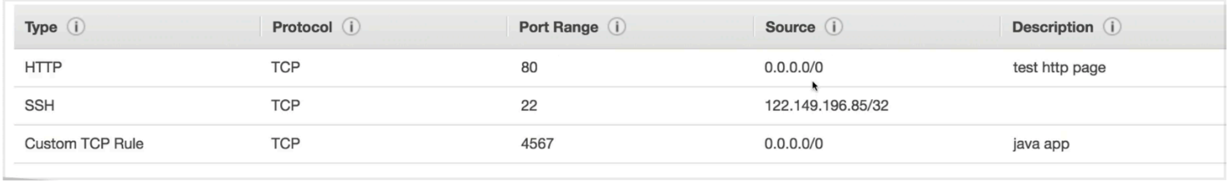
**Amazon EC2:**

- Elastic compute cloud: **infrastructure as aservice.**
- It consist of:
  - Renting virtual machines EC2
  - Store data on virtual drives (EBS)
  - Distribute load across machines (ELB)
  - Scale services using auto-scaling group(ASG)
- Knowing EC2 is fundamental to understand how the cloud works.
- **EC2 sizing & configuration options:**
  - **Os:** Linux, Windows, or Mac
  - How much compute power & cores(CPU)
  - How much RAM
  - How much storage space:
    - Network-attached (EBS&EFS)
    - Hardware (Ec2 instance store)
  - Network card: speed of the card, Public IP address
  - Firewall rules: security group
  - Bootstrap script(configure at forst launch): EC2 User Data
- **EC2 User Data:**
  - Possible to bootstrap instances using EC2 User data script.
  - **Bootstraping** =Launchiing commands when machine starts.
  - That script is only run once at the instance first start.
  - **EC2** user data is used to automate boot tasks such as:
    - installing updates/softwares
    - Downloading command files from the internet
  - **Ec2 User data:** script runs with root user.
  - When we stop and restart EC2 instance, public P will change, private IP will stay same.
- **EC2 instance types:**
  - 7 types

    - Naming convention like m5.2xlarge
    - M = instance class
    - 5 = generation
    - 2xlarge = size within the instance class
    - **Compute optimized:**
        - Good for intensive tasks which require high **performance**
        - Batch performing workloads
        - Media transcoding
        - High performance web servers
        - High performance computing
        - Scientific modeling & machine learning
        - Gaming servers
        - For these compute type starts with letter C like: c6g c6gn, c5, c5a, c5n, c4
    - **Memory Optimized:**
        - Fast performance for workloads that process large datasets in **memory**
        - High performance, relational/non relational DB
        - Distributed web scale cache stores
        - In-memory database optimized for BI
        - Application performing real time processing of big unstructured data.
        - Usually starts with letter R for RAM: R6g R5 …
    - **Storage** optimized:
        - Great for storage intensive tasks that require high sequential read and write access to large data sets on local storage
        - High frequency OLTP systems
        - Relation and noSQL databases
        - Cache for in-memory databases (ex. Redis)
        - Data warehousing applications
        - Distributed file systems
        - Naming will start with I/ D.

**Introduction to Security Groups:**

- Fundamental of network security
- Controls how traffic is **allowed into/out of EC2 instances**
- S**ecurity groups only contain allow rules *****
- **State ful ( return traffic is automatically allowed)**
- Can reference by IP/ by security group
- Act as firewall on EC2 instance
- Regulate :
    - Access to ports
    - Authorized IP ranges - IPV4 and IPV6
    - Control of inbound network & outbound network



| Type | Protocol | Port Range | Source | Description |
|---|---|---|---|---|
| HTTP | TCP | 80 | 0.0.0.0/0 | test http page |
| SSH | TCP | 22 | 122.149.196.85/32 | |
| Custom TCP Rule | TCP | 4567 | 0.0.0.0/0 | java app |

- Can be attached to multiple instances
- Loved down to region/ VPC combination
- Traffic live outside EC2 instance, If its blocked, EC2instance won't see it.
- Good to maintain one separate security group for SSH access.
- If application is not **accessible** (time-out) then its **security group** issue.
- If its connection failed error, then its application error/ its not launched.
- All i**nbound traffic is blocked by default & outbound is authorized by default.**
-

**Classic ports to know:**

- 22 = SSH(secure shell) - Log in to Linux instance
- 21 = FTP (File transfer protocol ) - Upload files into a file share.
- 22 = SFTP ( Secure file transfer protocol) - upload files using SSH.
- **80 = HTTP -** access unsecured websites.
- **443 =** HTTPS - access secured websites
- 3389 = RDP(Remote Desktop Protocol) - Log into a window instance.
- **SSH Summary Table:**
- **OS. | Supports**
- **|SSH. | Putty | Ec2 instance connect**
- **Mac. Yes. Yes**
- **Linux Yes Yes**
- **Win <10. Yes Yes**
- **Win>10. Yes. Yes. Yes**

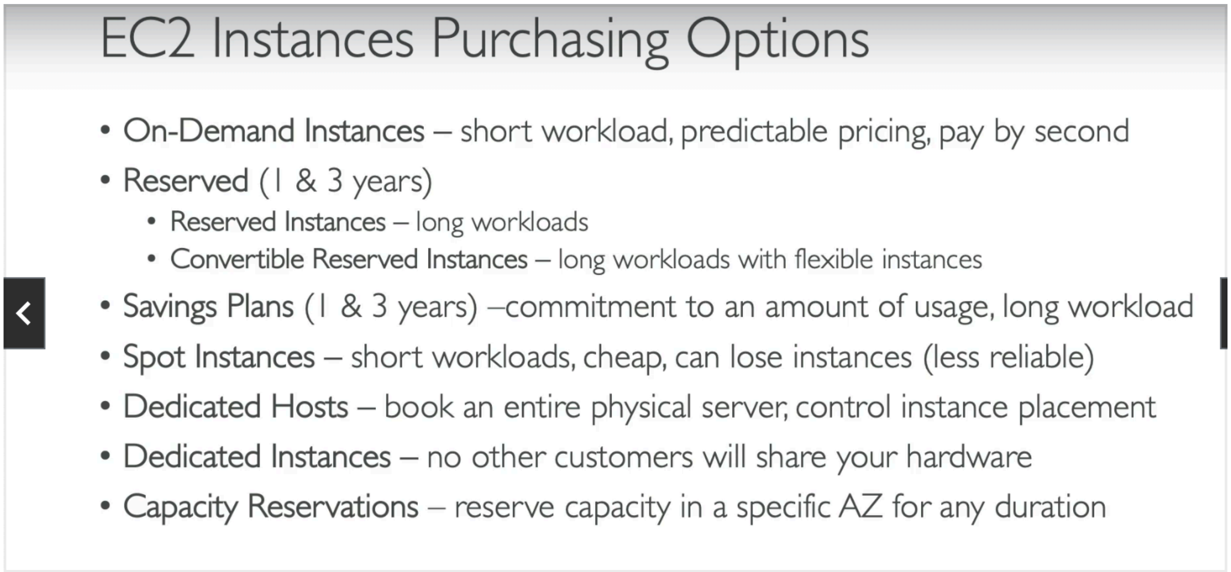
**Using SSH on Linux/Mac**

ssh -i /Users//Downloads/AWS_Quz/Aws_files/EC2Tutorial.pem ec2-user@publicip

**Ec2-user is the name we are giving to ec2 connection.**
**It can be anything**

To change permission: chmod 400 /pathto.pemfile
To exit from this :logout or exit

**EC2 instance purchasing options:**



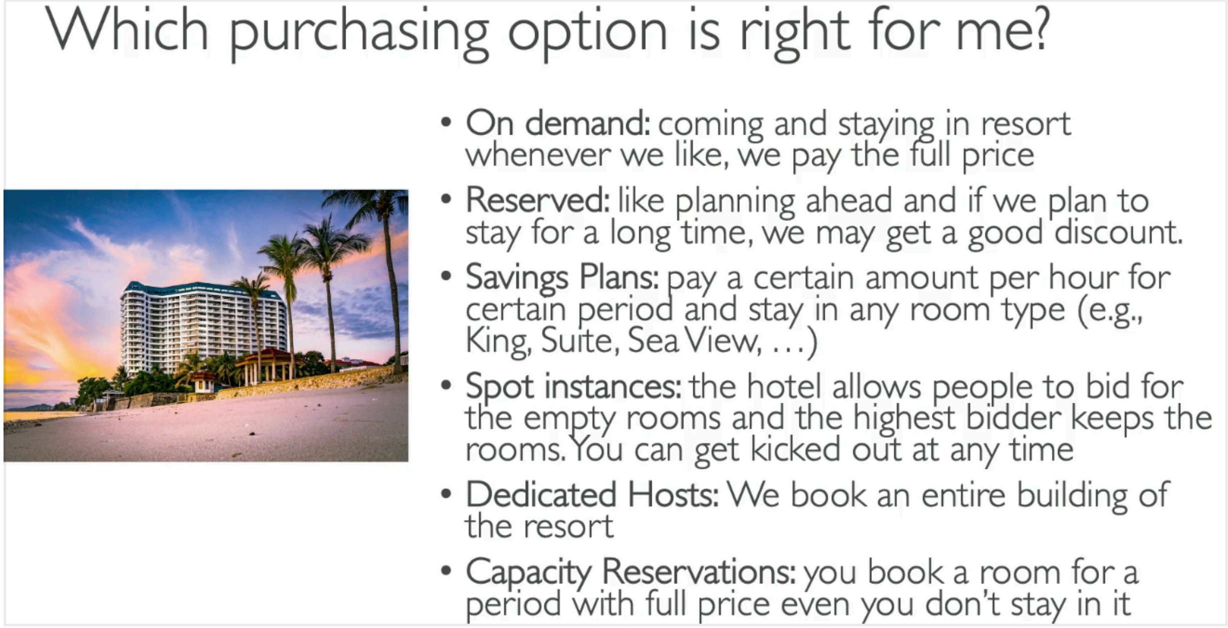
## EC2 Instances Purchasing Options

- On-Demand Instances – short workload, predictable pricing, pay by second
- Reserved (1 & 3 years)
    - Reserved Instances – long workloads
    - Convertible Reserved Instances – long workloads with flexible instances
- Savings Plans (1 & 3 years) –commitment to an amount of usage, long workload
- Spot Instances – short workloads, cheap, can lose instances (less reliable)
- Dedicated Hosts – book an entire physical server, control instance placement
- Dedicated Instances – no other customers will share your hardware
- Capacity Reservations – reserve capacity in a specific AZ for any duration

- **ON DEMAND:**
    - **Pay for what you use**
        - **Linux / windows**: B**illing per second after first minute**
        - Other OS: billing per hour.
    - Has the highest cost but no upfront payment
    - No long term commitment
    - Best for **short-term**, and **un-interupted** workloads, where you can't predict how application will behave
- **RESERVED Instances:**
    - Up to **72%** discount compared to on-demnad
    - You reserve specific instance attributes ( instance type. Region, Tenancy, OS)
    - Reservation period: 1 year(some discount), 3 years(more discount)
    - Payment options: no-upfront(some discount), partial upfront( more discount), all upfromt(best discount)
    - Reserved instance scope: **Regional**/zonal (reserve capacity in an AZ
    - Recommend for steady-sate usage applications( like **Database**)
    - **Can buy and sell in the reserved instance marketplace.**
    - **Convertible reserved instance**:
        - Can change ec2 instance type, instance family, os, scope and tendency
        - Up to 66% discount
- **Saving Plan**:
    - Get discount on long-term usage( up to **72%** was Reserved)
    - Commit to a certain type of usage($10/hour or 3 years)
    - Usage beyond saving plan is billed like ON_DEMAND
    - Locked to a specific instance family & AWS region, eg(M5 in US east1)
    - Flexible across:
        - Instance size ( eg: m5.xlarge , m5.2xlarge)
        - OS(linux, windows)

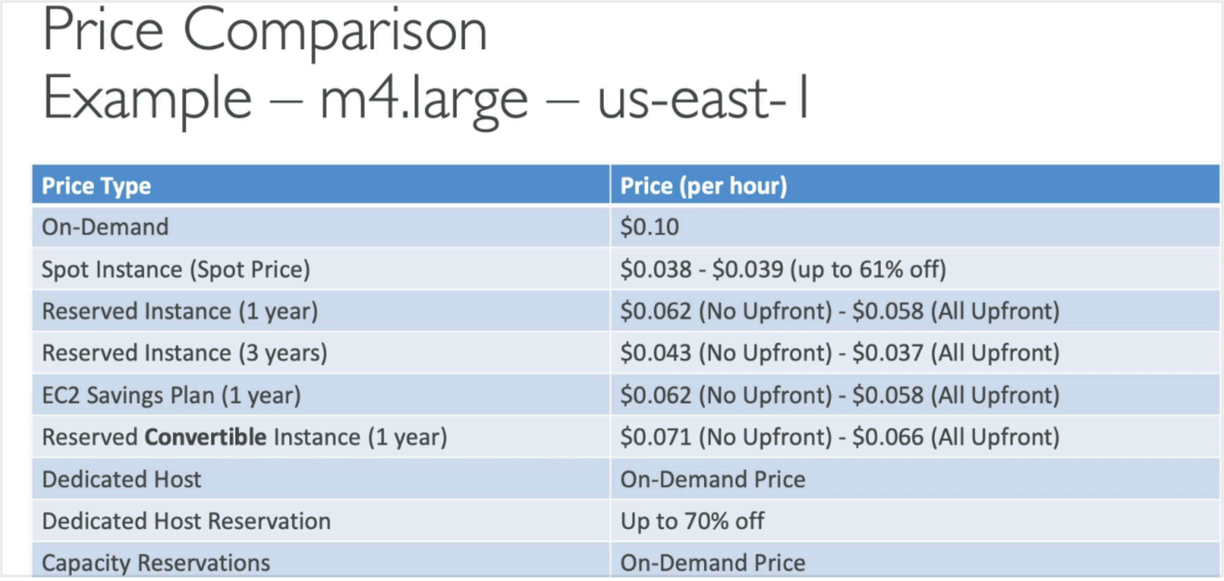
- Tenancy (Host, Dedicated, Default)

- **Spot Intance:**
  - Can get discount unto **90%** compared to on-demand
  - :< Can loose at any time, if your price is less than current spot price.
  - Most cost efficient instance
  - Useful for workloads that are resilient to failure:
    - Batch jobs
    - Data analysis
    - Image processing
    - Any distributed works.
    - Worklods with flexible start and end time
  - Not suitable for critical jobs/databases
- **EC2 dedicated Hosts:**
  - **Physical server with EC2 nstance capacity fully dedicated your use.**
  - Allows you to adress c**ompliance requirements** and use your existing **server-bound software license.**
  - Purchasing options:
    - Ondemaand
    - Reserved
  - Most expensive option
  - Useful for softwares that have c**omplicated licensing mode**l.( bring your own license)
  - Or for companies that have strong regulatory or compliance needs.
- **EC2 dedicated instances:**
  - Instances run on hardware that's dedicated to you
  - May share hardware with other instances in same account.
  - dedicated instances mean that you have your own instance on your own hardware,
  - whereas dedicated host, you get access to the physical server itself and it gives you visibility into the lower level hardware.
- **capacity reservations for EC2.:**
  - Reserve on demand instances capacity in specific AZ
  - Always have access to EC2 capacity when you need it.
  - No time commitments, no billing discounts.
  - To get discount : combine with regional reserved instances and saving plans.
  - Charged as on-demand rate whether you run instance or not. You are paying because you reserved.
  - Best for short term, uninterrupted workloads that needs to be in specific AZ



## Which purchasing option is right for me?

- **On demand:** coming and staying in resort whenever we like, we pay the full price
- **Reserved:** like planning ahead and if we plan to stay for a long time, we may get a good discount.
- **Savings Plans:** pay a certain amount per hour for certain period and stay in any room type (e.g., King, Suite, Sea View, …)
- **Spot instances:** the hotel allows people to bid for the empty rooms and the highest bidder keeps the rooms. You can get kicked out at any time
- **Dedicated Hosts:** We book an entire building of the resort
- **Capacity Reservations:** you book a room for a period with full price even you don't stay in it



## Price Comparison Example – m4.large – us-east-1

| Price Type | Price (per hour) |
|---|---|
| On-Demand | $0.10 |
| Spot Instance (Spot Price) | $0.038 - $0.039 (up to 61% off) |
| Reserved Instance (1 year) | $0.062 (No Upfront) - $0.058 (All Upfront) |
| Reserved Instance (3 years) | $0.043 (No Upfront) - $0.037 (All Upfront) |
| EC2 Savings Plan (1 year) | $0.062 (No Upfront) - $0.058 (All Upfront) |
| Reserved **Convertible** Instance (1 year) | $0.071 (No Upfront) - $0.066 (All Upfront) |
| Dedicated Host | On-Demand Price |
| Dedicated Host Reservation | Up to 70% off |
| Capacity Reservations | On-Demand Price |

**Shared Responsibility  model for EC2**

AWS:

- Infrastructure
- Isolation on physical hosts
- Replacing faulty hardware
- Compliance validation

USER:

- Security group rules.
- OS patched and updates.
- All softwares/utilities installed on Ec2
- IAM roles
- Data security on your instance.

EC2 Summary:

- EC2 instance: AMI(OS), instance size (CPU + RAM) + storage +security groups + ec2 user data.
- Security groups : firewall attached to EC2 instance
- USER DATA: script launched at the first start of an instance
- SSH: Start terminal on EC2 instance (port 22)
- EC2 Instance role: link to IAM roles
- Purchase options : on-demand, reserved(standard + convertible), dedicated host, dedicated instance

**EC2 instance storage section:**

- What is EBS volume?
- **Elastic Block Store** Volume is a **network drive** that you can attach to your EC2 instances while they run
- It allows to persist data **even after termination** of EC2 instance.
- we can recreate an instance and mount to the same EBS Volume from before and we'll get back our data.
- They can only be **mounted to one instance**.( but one ec2 can have multiple EBS)
- They are bound to **specific availability zone**.
- It's Like : Network USB stick.
- Free Tier: 30 GB of free EBS storage of type General Purpose SSD or magnetic per month.
- Its a network drive ( not physical drive)
- So it uses the network to communicate the instance, so there will be latency.
- It can be detached from EC2 instance and attached to another one quickly.
- Its **locked** to an Availability Zone:
  - To move a volume across, you first need to snapshot it.
  - Have a provisioned capacity
  - You get billed for all the provisioned capacity.
  - Can increase the capacity of the drive over time.
- When we are creating EBS for a instance, we have option" DELETE ON TERMINATION"
- For root volume: default it will be selected, for EBS by default there is no selection.
- This controls the behaviors when instance terminated.
- **EBS snapshot: like backup**
  - Make a copy at any time
  - Not necessary to detach before making snapshot, but advisable
  - Can **copy snapshot across AZ : can make a snapshot and then copy this to the EBS of another AZ**
  - EBS snapshot Archive:
    - Move s snapshot to **an "archive tier" that is 75% cheaper.**

        - Takes 24-72 hours from restoring the archive
    - Recycle bin for EBS snapshots:set up rules to retain delete snapshots so you can recover them over accidental deletion. Or can specify retention perios from 1 day to 1 year.
- **AMI Overview:**
    - Amazon Machine image
    - AMI are a **customization of EC2**
        - You add your own s/w, configuration, operating system, monitoring.
        - Faster boot/configuration time because all your software is pre-packaged

    - **AMI's are built for specific region( Can be copied across regions)**
    - Can launch EC2 instances from
        - Public AMI : AWS provided
        - Your own AMI: can make and maintain them yourself
        - Aws marketplace AMI: someone can make AMI and sell it
    - AMI process from EC2 instance:
        - Start an EC2 instance and customize it.
        - Stop the instance for data integrity
        - Build AMI- it creates EBS snapshots
        - Launch instances from other AMI's
        - US EAST1 ( EC2) —> create AMI—> launch US_EAST2 from AMI
        - Main use is: if some shares the AMI to me, I can create my ec2 from this, so that all the required softwares would be already installed in it.
- **EC2 Image Builder:**
    - **Used to automate the creation of VM's / container images.**
    - Automate the creation, maintain, validate and test EC2 AMI's.
    - Can be run on a schedule(weekly, whenever packages are updated)
    - EC2 image builder -> (create) -> builder ec2 instance ->(create) NEW AMI -> test ec2 instance -> AMI is distributed across regions.
- **EC2 instance store: (** name of the hardware, the hard drive attached to the physical server), for short term purpose.
    - **EBS are network drives with good but limited performance.**
    - **If need high performance hardware disk, use EC2 instance store**
    - Better I/o performance, disk performance.
    - if you stop or you terminate your EC2 Instance, that has an Instance Store, then the storage will be lost. And therefore it's called an **ephemeral storage**
    - Good for buffer / cache / scratch data / temporary content not for long term storage( for this **EBS** is best)
    - Risk of data loss if hardware fails.
    - Backups and replication are your responsibility.

- **EFS : Elastic file system:**

    - Managed **NFS** ( network file system)that can be mounted **on 100’s of EC2**
    - Works with only **LINUX** EC2 instances in **multi-AZ**
    - Highly available, scalable, expensive (3Xgp2) , pay per use, no capacity planning

- **EBS vs EFS:**
    - EBS cab be attached to only one Ec2, if to attach to other region, need to create snapshot.
    - EFS network file system can be attached to multi zone EC2’s of linux EC2, so all the files can be shared.
- **EFS IA: EFS infrequent access**
    - Storage class that is cost-optimizes for files not accessed every day.
    - **Unto 92% lower cost compared to EFS.**
    - If we enable EFS-IA, EFS will automatically move files to EFS-IA based on the last time they were accessed.( Called as LIFE CYCLE Policy)
    - EX: Move files that are not accessed for 60 days to EFS-IA.
    - Next time you access the file, moved back to EFS
    - Idea is for cost optimization.
- **Shared responsibility model for EC2 Storage**

**AWS:**

- Infrastructure
- **Replication** for data for **EBS** volumes & **EFS** drives
- Replacing faulty hardware

User:

- Setting up backup/snapshot procedures
- Setting up data encryption
- Responsibility of any data on the drives.
- Understanding the risk of using EC2 instance store.

**Amazon Fsx - Overview**

- Launch 3rd party high-performance file system on AWS.
- Fully managed service.
- FSX for Lustre, FSX for Windows file server , FSx for NetApp ONTAP.
- **Amazon FSx for Windows File Server:**
    - A fully managed highly reliable and scalable Windows native shared file system.
    - Built on Windows File Server
    - Supports **SMB** protocol & Windows NTFS.
    - Integrated with Microsoft Active Directory.
    - **Can be accessed from AWS or from on-premise** infrastructure.
- **AMAZON FSx for Lustre:**
    - Fully managed, high-performance, scalable file storage for High performance computing(HPC)
    - Lustre is derived from Linux and Cluster.

    - Ideal for ML, analytics, Video processing finance modeling…
    - Scales up to 100s GB/s, millions of IOPS, sub-ms latencies.
    - It can access data from EC2 or data centr(server), S3…
- **EC2 instane storage Summary:**
    - EBS:
        - Network drives attached to one EC2 instance at a time
        - Mapped to AZ
        - Can use EBS snapshots for backup/transferring EBS across AZ
    - AMI: create ready-to-use EC2 instances with out customizations.
    - EC2 image Builder : Automatically build ,test, distribute AMI's
    - EC2 instance store:
        - High performance hardware disk attacehed to EC2 instance.
        - Lost if our instance is stopped/terminated
    - EFS:Network file system , can be attached to 100's of instances in a region for Linux.
    - EFS-IA: Cost optimized storage class for infrequent accessed files.
    - FSx for Windows: NFS for windows servers
    - FSx for Lustre: High performance computing Linux file system

### Elastic load balancing and Auto scaling groups section

- **Scalability** = can handle greater load
- 2 Kinds:
    - **Vertical scalability** : Increasing the size of instance. Ex ( t2.micro to t2.large): Common for non distributed systems such as database. Hardware limit would be there.
    - **Horizontal scalability** ( Elasticity)
        - Increase the no. of instances/ systems
        - Implies **distributed** systems.
        - Common for **web** / **modern** applications.
        - Can be done easily using clouds like Amazon Ec2.
- **High Availability:**
    - Goes hand in hand with horizontal scaling.
    - High availability means running your application / system in at least 2 AZ's.
    - **Goal is to service data center loss.**
- **High Availability and Scalability for Ec2:**

- vertical scaling : increase instance size (**Scale up/down)**
- Horizontal: increase no.of instances (**scale out/in)**
    - Auto Scaling Group
    - Load balancer
- High availability : run instances for the same application across multiiAZ
    - Auto scaling group multi AZ.
    - Load Balancer multi AZ

**Scalability**: Ability to accommodate **larger load** by making the hardware stronger or by adding nodes.

**elasticity** : once a system is actually scalable,
Elasticity means that there will be some sort
of auto scaling in it,so that the system can scale
based on the load. This is cloud friendly: pay per use, match demand, optimize costs.

**Elastic load balancing:**

- Load balancers are servers that forward internet traffic to multiple servers (**EC2 instances**) downstream.
- **A**lso called the backend EC2 instances.
- We have load balancer exposed to customers and at the back we have multiple EC2's.
- Spread load across multiple downstream instances.
- Expose a single point of access (DNS) to your application.
- Seamlessly handle failures of downstream instances.
- Do regular health checks to your instances.
- Provide SSl termination (HTTPS) for your websites.
- High availability across zones.

**WHY ELB?**

- **ELB is a** managed load balancer( means aws takes care of it)
  - AWS guarantees that it will be working
  - AWS takes care of upgrades, maintenance, high availability.
  - Aws provides only a few configuration knobs
- Costs less to setup our own (but maintain ace and integration is is not easy)
- **4 types of Load balancers** offered by AWS.
  - Application Load Balancer ( HTTP / HTTPS only) -Layer 7
  - Network Load balancer( ultra-high performance, allows for TCP) - Layer 4
  - Gateway  Load balancer - Layer 3
  - Classic Load Balancer ( retired in 2023) Layer 4&7 and is being replaced by VALB and VNLB, VGLB.
- **VALB:**
  - HTTP / HTTPs /gRPC protocols(layer 7)
  - HTTP Routing Features
  - Static DNS(URL)
- **VNLB:**
  - TCP/UDP Protocols(Layer 4)
  - High performance : millions of requests per second.
  - Static IP: through Elastic IP
- **VGLB**
  - GENEVE Protocol on IP Packets (Layer3)
  - Route Traffic to firewalls that you manage on EC2 instance.

        - Intrusion detection.
        - Its 3rd party security virtual appliance used to detect intrusion traffic.

**VALB Hands on :**

- Launch ec2 instance with no.of instances more than 1( auto scaling would be available by default)
- Go to Load balancing, create one
- Target group = group of ec2 instances we created.
- Once its created DNS link will appear ; it the link of application which handles loads among EC2 instances.

- **Auto scaling group:**
    - IN reality, for ex retail: day time traffic is high compared to nights
    - In cloud: can create and get rid of servers automatically based on the traffic.
    - Goal of ASG :
        - **Scale out(add instances) to match high demand**
        - **Scale in: remove EC2 instances, to match decreased load.**
        - Ensure we have minimum and max no. of machines running.
        - Automatically register new instances to a load balancer.
        - Replace unhealthy instances by just reregistering and terminate and replace with new one.
        - Need to define minimum size of ec2 instances, desired size and max size.
        -
    - Auto Scaling groups:
        - **Manual scaling:** update the size of ASG manually
        - **Dynamic Scaling:** Respond to changing demand
            - **Simple / Step scaling :**
                - When a cloud watch alarm is triggered ( ex. CPU>70%) then add 2 units.
                - When a cloud watch alarm is triggered ( ex. CPU<70%) then remove 2 units.
            - **Target tracking scaling:**
                - EX: I WANT THE AVERAGE ASG CPU to stay around 40%
            - **Scheduled scaling:**
                - Anticipate a scaling based on known usage patterns.
                - Ex: increase the min min.capacity to 10 at 5PM on Fridays.
            - **Predictive scaling:** Uses Ml to predict traffic,

- Deleter auto scaling group and Load balancer to delete EC2 instances, other wise if you terminate directly, ASG will create new instances .

---

———————————————————————————————————

**AWS S3:**

- Infinitely scaling storage
- Use cases:
    - Backup and Storage
    - Disaster recovery
    - Archive
    - Hybrid cloud storage
    - Application hosting
    - Media hosting
    - Data lakes and Big data analytics
    - Software delivery
    - Static website
- Allows people to store **objects** (**files**) in "**buckets**" ( **Directories**)
- Bucket name must be **unique across globe**
- Buckets areDefined at the **region level.**
- **S3** looks like global but buckets are created in a region.
- Naming : lowercase, numbers, - (3-63 characetrs long)
- Objects(files) have a key
- Key = Full path s3://my-bucket/my_file.txt
- Object value = content of the body
- No concept of directories.
- Max object size = 5TB
- If uploading more than 5 GB, use "Multi-part upload"
- Meta data = List of test key/value pairs - system or user metadata.
- Tags = unicode key /value pair - unto 10 - useful for security/lifecycle
- Version id : if versioning is enabled.

**S3 Security:**

- User based :
    - IAM Policies: which API calls should be allowed for a specific user from IAM.
- Resource Based:
    - Bucket policies - Bucket wide rules from S3 console -allows cross account
    - Object access control list: (ACL) - finer grain ( can be disabled)
    - Bucket access control list: (ACL)- less common (can be disabled)
    - Note: IAM principle can access s3 object of
        - User IAM permissions ALLOW it OR the resource policy allows it.
        - **AND** there is no explicit DENY
    - **Encryption :** Encrypt objects in Amazon S3 using encryption keys.

**S3 Bucket policies:**

- JSON based policies
    - Resources: Buckets and Objects
    - **Action** : Set of API to **Allow or Deny**

    - **Principle** : The **account** or **user** to apply the policy to
- Use S3 bucket policy
    - Grant public access to the bucket
    - Force objects to be encrypted at upload
    - Grant access to another account ( cross account)
    - Ex: Public access => Use bucket Policy
    - EX:  User Access to S3 => IAM Permission
    - EX: EC2 instance access => Use IAM Roles
    - EX: Cross - Account Access => Use Bucket Policy

**Amazon S3 - Static website hosting**

- S3 can host static websites and have them accessible on the internet.
- **Website** URL will be ( depending on the region )
- If you get 403 error: Make sure bucket policy allows public reads.
- In properties of S3 bucket : we can host a static website.

**Amazon S3 Versioning:**

- Can version files in S3
- Versioning is enabled at bucket level
- Best to enable
    - It prevents unintended deletes
    - Easily roll back to previous version
    - Files before enabling versioning will have version as null.
    - Removing versioning does not delete previous versions.

**Amazon S3 - Replication ( CRR & SRR)**

- If we have 2 buckets in 2 differnt regions, we want to enable ASYNCHRONOUs replication.
- Enable versioning in both source and destination buckets.
- Specify proper IAM permissions.
- Use cases:
    - CRR: ( Cross region replication)Compliance, lower latency access, replication across accounts.
    - SRR ( Same region replication) Log aggregation, live replication between production and test accounts.

**Hands on replication:**

- Create source and destination buckets with enabled versioning.
- In the source bucket, go to management-> replication rules-> create replication rule.
- To replicate already existing stuff => enable one-time Bach operation job to synchronize source and Destination

**S3 Storage classes:**

- S3 standard - general purpose
    - 99.99% availability, used for **frequently** accessed data

  - Low Latency and high throughput
  - Sustain 2 concurrent facility failures
  - Use case: Big Data Analytics, mobile& gaming applications, content distribution.
  -
- S3 standard - IA ( infrequent access)
  - For data **less frequently** accessed, but required **rapid** access when needed.
  - Highly available
  - Use cases: **Disaster recovery, backups**
  -
- S3 one zone - infrequent access
  - Highly durable in a single AZ;
  - Data lost when AZ is destroyed
  - **99.5% availability**
  - Use cases: **Storing secondary backup** copies of on-premise data, or you can **re-create**.
  -
- S3 Glacier instant retrieval
  - Glacier means low-cost object storage, meant for archiving / backup.
  - Pricing: price for storage + object **retrieval** cost.
  - Millisecond retrieval, great for data accessed once a **quarter**.
  - Minimum stoarege duration = 90 days.
  -
- S3 glacier flexible retrieval
  - Formerly amazon s3 glacier.
  - Data retrieval : Expedited ( 1- 5 minutes) Standard ( 3-5 hours) bulk ( 5-12 hours) - free
  - Minimum storage days = 90 days.
  -
- Glacier deep archive - for long term storage:
  - Standard (12 hours) bulk (48 hours)
  - Min storage duration = 180 days
  -
- S3 intelligent tiering
  - Small monthly monitoring and auto-tiering fee
  - Moves objects automatically between access tiers based on usage
  - **No retrieval charges** in S3 intelligent-tiering
  - Frequent access tier (automatic) : default tier
  - Infrequent access tier (automatic) objects not accessed for 30 days
  - Archive instant access tier: Objects not accessed for 90 days
  - Archive access tier: configurable from 90 days to 700+days.
  - Deep archive access tier (optional) from 180 days to 700+ days.
- Can move between classes manually or using S3 Lifecycle configuration.

–

**S3 Durability and Availability:**
**Durability:**

- durability represents how many times an object is going to be lost
- High 99.99% durability
- Availability: How readily a service is: 99.99% availability.

**S3 Lifecycle:**

- Lifecycle rules helps to manage the data.
  **S3 Encryption:**
- **Server side encryption:** (**default**) ; when we create bucket /upload object in s3, object is encrypted by default by aws for security purpose.
  - Server encrypts files after receiving it
- **Client side encryption:**
  - User encrypts, the file before uploading it.

**IAM Access Analyzer for S3-** Monitoring feature for s3 buckets to ensure only intended people will access it.

- Ensure only intended people have access to bucket.
- So analyze **bucket** policy, S3 **ACL's** S3 **access point policies**,
- **Powered by IAM access Analyzer**.

**Shared responsibility model for S3:**

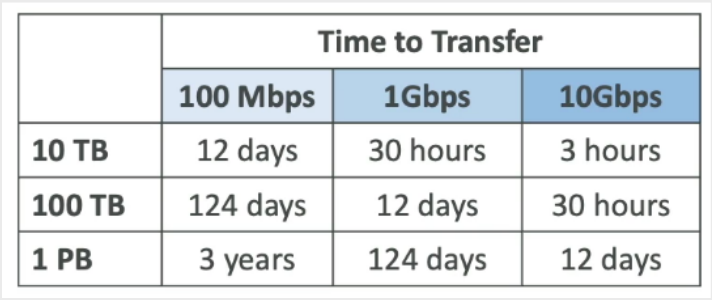
- **User:**
  - S3 versioning, bucket policies, replication set up
  - Logging and monitoring
  - Storage classes. Data encryption at rest and in transit.

**AWS Snow Family:**

- Highly secure, portable device used to collect and process data at the **edge** and **migrate data in and out of AWS.**
- 2 devices: **Snaowcone** and **snowball edge**

| | snowcone | Snowball edge | Snowmobile |
|---|---|---|---|
| Storage cpacity | 8TB HDD = 14 TB SSD | 80 TB - 210 TB | |
| Migration size | Up to terabytes | Upto petabytes | Exabytes |

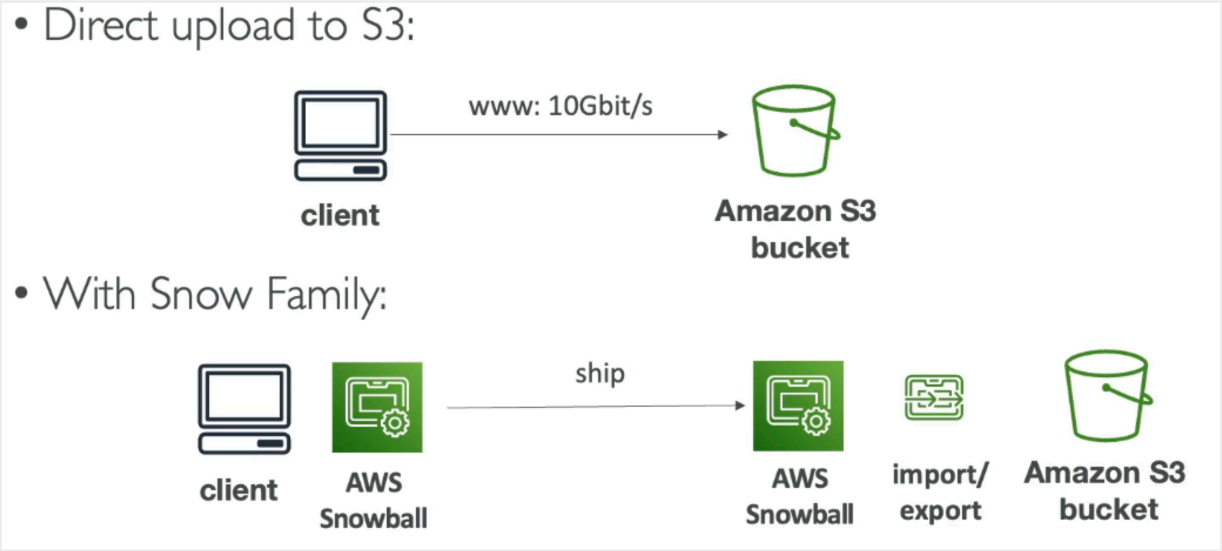
- Data migration with AWS Snow Family:
- Usual data transfer times



| | Time to Transfer | | |
|---|---|---|---|
| | 100 Mbps | 1Gbps | 10Gbps |
| 10 TB | 12 days | 30 hours | 3 hours |
| 100 TB | 124 days | 12 days | 30 hours |
| 1 PB | 3 years | 124 days | 12 days |

Challenges:
Limited connectivity
Limited bandwidth
High network cost
Shared bandwidth ( can’t maximize line)
Connection stability

AWS snow family : **offline devices to perform data migration.**
If ot takes more than a week to transfer over the network, use snowball devices.



- We will buy snowball physical device, upload /ship data into it, then AWS export/import data to S3 bucket,
- **Steps:**

- first request a Snowball device from the AWS console for delivery.
- install the Snowball clients / AWS OpsHub on servers to transfer data.
- connect the Snowball device to our server and we start copying files using the clients, and then
- we ship back the device when we're ready It goes directly into an AWS facility.
- The data is going to be loaded onto an Amazon S3 bucket, and then your
- Snowball is completely wiped and can be sent to another customer.

**Edge Computing:**

- To process data created on the edge location where these is limited or no internet.
- We order snow ball / snow cone device to do edge computing.
    - Snow cone: 2 CPU, 4 GB of memory , wired/wireless access
    - Snow ball edge: Compute optimized (dedicated for use case ) storage optimized
    - Run EC2 instances/ lambda functions at the edge on device.
- Use cases: pre-process data where it's created, or to do machine learning, again on data where it's created or to transcode media while it's being shipped back to AWS.

Snow family Hands on: can order device on console
Snowball edge pricing:

- Pay for devise usage and data transfer out of AWS
- Data **transfer in** to S3 **free**
- **On Demand:**
    - Include a one-time service fee per job, which includes
        - 10 days , storage optimized 80 TB
        - 15 Day, optimized 210 TB
    - Shipping days were not included in the days.
- Committed upfront
    - Upfront fee, for monthly , 1 year, 3 year usage ( Edge computing)
    - Up to 62% discounted pricing.

**Hybrid cloud for storage:**

- **Bridge** between on-**premise** data and **cloud** data in S3
- How do you expose the S3 data on premise?
- Need to use **AWS STORAGE GATEWAY**
- 
- **AWS storage Cloud native options:**
    - **Block: Amazon EBS, EC2 instance store**
    - **FILE: Amazon EFS**
    - **Object : S3, Glacier**

**Types of storage gateway:**

- **File Gateway:**
- **Volume Gateway**
- **Tapes gateway**

———————————————————————————————————————————

**Data Bases:**

- Storing data on disk : EFS, EBS, Instance store, S3
- **Sore in DB:** Structure data, build indexes to query efficiently, search through

data
- Define relationships between datasets.
- Optimized for a purpose and common different features, shapes constraints.
- **Relational Databases:**
  - You would define relations between tables
  - Can use Sql language to perform queries / lookups
- **NOSQl Databases:**
  - No SQl= Non SQL, non relational
  - They are purpose built for specific data models and have flexible schema for building modern applications.
  - **Benefits:**
    - Flexibility: Easy to evolve data model
    - Scalability: designed to scale out by using distributed clusters.
    - Can do horizontal scaling.
    - High performance : optimized for specific data model
    - Highly Functional types optimized for data model
    - EX: key value, document, graph, in-memory
    - EX: JSON ( Java script object notation )
    -
- **Shared responsibility model for Data Bases:**
- AWS offers use to manage different databases.
- benefits:
  - Quick provisioning, High availability, vertical and horizontal scaling.
  - Automated backup & restore, operations, upgrades.
  - OS patching is done by AWS
  - Monitoring, alerting,
- **Note:** Many database technology run on EC2 instance, but you need to handle yourself, all the things associated with resiliency, backup, patching, high availability, fault tolerance, and scaling. So this is why in this case, using a managed database is going to be a lifesave

**RDS:** Relational data base service.
- Managed DB service for DB, use Sql as query language
- Allows to create databases in cloud managed by AWS.
  - Postgres
  - Mysql
  - MariaDB
  - Oracle
  - Microsoft Sql server
  - IBM DB2
  - Aurora { AWS Proprietary database)

- Advantages of using RDS over deploying DB on EC2
  - RDS is a managed service.

        - Automated provision, OS patching.
        - Continuous backup & restore to specific timestamp.
        - Monitoring dashboards
        - Read replicas for improved read performance
        - Multi AZ set up for Disaster recovery
        - Maintainance windows for upgrades.
        - Scaling capacity ( vertical and horizontal)
        - Storage backed by EBS
    - But you can't SSH into your instances.

**1.RDS Solution** Architecture

Elastic Load balancer ( this will be taking web request)——> { Multiple EC2 instances Possibly in ASG(auto scaling group). <——— (Read/write) AMAZON RDS( all the EC2's store and share the data together)

**2.Amazon Aurora:**

- Only for AWS not open source
- Supports postgres and mysql
- " AWS Cloud optimized " and is **5X** performance improvement **over MYSQL**, **3X** over **postgres on RDS**
- No need to worry about storage, grows automatically.
- Even though more expensive then RDS, but going to be cost effective.
- **Amazon Aurora serverles: (no management over head)**
- Automated database instantiation & Auto scaling based on Actual usage.
- Postgres and mysql were supported in serverles
- No capacity planing needed
- Least management overhead.
- **Pay per second**, can be more cost effective.
- Use cases: Good for infrequent , intermittent or unpredictable workloads.
- Client ——> proxy fleet (managed by Aurora —->{ set of amazon aurora instances}

**RDS Deployments : Read Replicas, Multi - AZ :**

- **Read Replicas** : **Scale** the read workloads of your DB.
    - Can create upto **15 read replicas.**
    - Data is only written to the Main DB.
- **Multi - AZ:**
    - Failover in case of AZ outage ( high Availability)
    - Read and write happens only to the main RDS
    - Can only have **1 AZ as failover.**
- **Multi - Region :**
    - Multi - region (Read Replicas)
    - We will have multiple read-replica's in different regions.
    - If writes happen in those other regions, this action would be done on main

region RDS only.
    - Here the writes are cross- region.
    - **Disaster recovery in case of region issue.**
    - **Local performance for global reads.**
    - **Replication cost.**

**Amazon ElastiCache Overview:**

- Same way RDS is to get managed relational databases..
- ElastiCache is to get **managed Redis / Memcached**.
- Caches are **in-memory databases** with high performance, low latency.
- Helps **reduce load off databases for read intensive workloads.**
- **AWS** takes care of OS maitainance, patching, optimization, set up, configuration, monitoring, failure recovery and backups.
    - ELB ——> Ec2 instances in ASG <——— (read/write) (slow) RDS. ^
    - |
    - **(**Read/ write from cache) ElastiCache in-memory database.

**Dynamo DB: ( NO-SQL)**

- Fully managed highly available with replication **across 3 AZ**
- NoSQL database - **not a relational database.**
- Scales to massive workloads, distributed, " **serverless**" database.
- Can handle millions of requests per second
- Fast and consistent in performance
- Single digit **millisecond latency** - **low latency retrival.**
- Integrated with IAM for security, authorization and administration.
- Low cost and auto scaling capabilities.
- **Standard and IA Table class**

Types of data:

- Key/value database.
    - Primary key = partition key + sort key
    - Schema is defined per item.

**DynamoDB Accelerator - DAX**

- **Fully managed in-memory cache** for DynamoDB.
- Your application wants to access Dynamo DB, and you want to cache most frequently read objects, then you would use D**AX / DynamoDB accelarator** as a cache.
- 10X performance improvement,
- single digit millisecond latency to microseconds latency when accessing your DynamoDB tables.
- Secure, highly scalable and available
- DAX is only used for and is integrated with Dynamo DB, while Elastic cache is for other databases.

- Since its **serverles**, we no need to create database, we can directly create tables.

**Dynamo DB - Global Tables:**

- Make a dynamo DB table accessible with low latency in multiple-regions.
- If we have dynamo DB global table in one region, then we can have 2 way replications in another regions.
- So reads/ writes can happen to any global table and all the tables will be in sync
- Active-Active replication (read/ write to any AWS Regions)

**Redshift overview:**

- Based on postgresql, but it's **not** used for **OLTP** like RDS
- Its **OLAP** - ( analytics and data warehouse)
- Load data once every hour not every second.
- 10X better performance than other data warehouses, scale to PB's of data.
- **Columnar** storage of data
- Massively parallel query execution (**MPP**) , highly available.
- Pay as you go based on the instance provisioned.
- Has a Sql interface for performing queries.
- Integrated with BI tools like quick sight, tableau

**REDSHIFT - serverless :**

- Automatically provisions and scales data warehouse underlying capacity.
- Run analytic workloads without managing data warehouse infrastructure.
- Pay for what you use.
- Use cases: Reporting, washboarding, real time analytics
- Enable Amazon redshift serverless on your account —> connect using amazon redshift query editor or any other tool —> amazon redshift serverless( run queries) —> pay for compute and storage used during analysis.

**EMR**: elastic map reduce.

- Helps to create **Hadoop cluster**s ( Big Data ) to analyze and process vast amount of data.
- Clusters can be made up of **hundreds of EC2** instances.
- Also supports apache spark, HBase, Presto…
- EMR take care of provisions these multiple EC2 instances and configuration.
- Auto scaling and integrated with Spot instances.
- Use cases: Data processing, ML, web indexing, big data

**Athena:**

- **Serverless** query service to **perform analytics against S3 objects.**
- **Understand SQL Language** to query files.
- Support CSV< JSON, ORC, AVRO, Parquet..
- Pricing = $5 per TB of data scanned.
- Use compressed / columnar data for cost- savings.

- Use case: BI/ analytics / reporting. Analyze and query VPC flow logs, ELB logs, CloudTrails,…
- **Analyze data in S3 using serverless SQL = Use ATHENA**

**Quicksight:**

- **Serverless** machine learning -powered business intelligence service to create interactive **dashboards**
- Fast, automatically scalable, embeddable, with per-session pricing.
- Use cases: BI, visualizations, ad-hoc analysis,
- Integrated with RDS, Aurora, Athena, redshift, s3.

**DocumentDB: ( No-SQL)**

- **Aurora** is an AWS implementation of PostgreSQL / MySQL
- **DocumentDb** is same for **MongoDB**( NoSQL database)
- MongoDB is used to store, query, and index JSON data.
- Similar "deployment concepts" as Aurora.
- Fully managed, highly available with **replication across 3 AZ**
- Automatically grows in increment of 10GB
- Automatically scales to workloads with millions of requests per second.

**Amazon Neptune:**

- Fully managed **graph database**.
- Ex: social network.
- Highly Available with **3 AZ**, up to **15 read replicas**
- Build and run applications working with highly connected datasets - **optimized for complex and hard queries.**
- Can store up to **billions** of **relations** and query the graph with **milliseconds latency.**
- Great For knowledge graphs( wikipedias), fraud detection, recomendatiion engine, social networking.

**Amazon timestream:**

- Fully managed, fast, scalable, serverles **Time Series Database.**
- Automatically scales up/down to adjust capacity
- Storage and analyst trillions of events per day
- **1000's** times faster & 1/10th cost of relational databases.
- Built-in time series analytics functions( helps you identity patterns in your data in near real-time)

**Amazon QLDB: "** Quantum Ledger Database" : for Financial transactions.

- A ledger is a book "**recording financial transaction"**
- Fully managed, serverless,, high available, replication across 3AZ.
- Used to **review history of all the changes made to your application data over time.**

- **Immutable** system : No entry can be removed or modified, cryptographically verifiable.
- 2-3X performance than common ledger blockchain frameworks, manipulate data using SQL.
- Difference with Amazon Managed BlockChain : **No decentralization component** in accordance with financial regulation rules.
- So the difference between QLDB and Managed Blockchain is that **QLDB has a central authority component** and it's a **ledger**, whereas **managed blockchain is going to have a de-centralization component as well.**

**Amazon managed Blockchain:**
- It makes it possible to build applications where multiple parties can execute transactions without the need for trusted central authority.
- So there's a **decentralization** aspect to a Blockchain.
- Amazon managed blockchain is managed service to:
  - Join public blockchain networks.
  - Or create your own scalable private network.
- Compatible with the frameworks Hyperledger Fabric & Ethereal

**AWS Glue:**
- Managed **ETL** Service.
- Useful to prepare and transform data for analytics.
- Fully **Serverless** service.
- **Extract data from S3 / RDS—> Transform in GLUE —> Load to redshift**
- **Glue data catalog :** Catalog of datasets ( Can be used by Athena, Redshiift, EMR)

**DMS: Database migration Service:**
- To **migrate data from one database to another**,
- It does not migrate server only migrate db
- **To migrate server: use Application Migration service (AMS)**
- AWS MGN is an automated lift-and-shift solution. This solution can migrate physical servers and any databases or applications that run on them to EC2 instances in AWS.
- We'll extract the data from the source database
- so we'll run an EC2 instance that will be running the DMS software.
- and then DMS will insert the data back
- into a target database that will be somewhere else.
- quick and secure database migration into AWS that's going to be resilient and self healing.
- Source database remains available during the migration.
- Supports: Homegenius migrations : ex oracle to oracle.
- Heterogenius migrations : Ex Microsoft Sal server to Aurora

**Summary:**

- Relational : OLTP : RDS & Aurora (SQL)
- Inmemory database : ElasticCache
- Key/value Database: DynamoDB (serverless) & DAX( cache for DynamoDB)
- Warehouse: OLAP  : Redshift (SQL)
- Hadoop Cluster: EMR
- Athena: Query data on S3( serverless & Sql)
- Quicksight: dashboards( serverless)
- Document DB : Aurora of MongoDB (Json-NoSQl database)
- Amazon QLDB : Financial transaction ledger ( immutable journal, crryptographically verifiable)
- Amazon Managed blockchain: Managed Hyperledger Fabric & Ethereal blockchains.
- Glue: managed ETL and data catalog here.
- DMS: database migrations
- Neptune: graph db
- Timestream : Time-series database.

**Docker:**

- SW development platform to deploy apps.
- Apps are packaged in CoNTAINERS that can be run on any OS
- **Apps run the same regardless of where they are running.**
  - Any Machine
  - No compatibility issues
  - Predictable behaviors
  - Less work
  - Easier to maintain and deploy
  - Works with any language, any os, technology
- Scale containers up and down quickly in seconds
- Docker images are stored in Docker repositories.
- Private : Amazon ECR (Elastic container registry)

**Docker V/s Virtual machines:**

- Docker is "sort of " virtualization technology but not exactly
- Resources are shared with the host => Many containers on one server.
- In virtual machines, applications comes with its own OS (VM)
- But in container, there will be just containers, not Full OS, it makes versatile and very easy to scale.

**ECS, Fargate &ECR Overview:**

- **ECS** = Elastic container service
- Launch Docker containers on AWS.

- You must provision & maintain the infrastructure ( the EC2 instances)
- AWS takes care of starting /stoping containers.
- Has integration with the Application Load balancer.

- **Fargate:**
- Launch Docker container on AWS
- You **do not provision** & maintain the infrastructure ( the EC2 instances) - simpler
- **Serverless**
- AWS just runs containers for you based on the CPU/RAM you need.
-
- **ECR: Elastic container registry:**
- Private docker registry on AWS
- This is where you store your Docker images so they can be run by ECS/ Fargate.

**Serverless:**

- Serverless is a new paradigm in which the developers dont have to manage servers anymore.
- They just deploy code.
- They just deploy functions.
- **Initially :** serverles == FaaS ( Function as a Service)
- Serverless was pioneered by AWS Lambda but now also includes anything that's managed : " Databases , messaging, storage , etc "
- Serverless does not mean there are no servers….
- **It means** you just dont manage / provision / see them
- Ex: S3, DynamoDB, Fargate , lambda
-

**AWS Lambda:**

- **EC2 :** Virtual servers in the cloud
- Limited by RAM and CPU
- Continuously running
- Scaling means intervention to add/remove servers.
- **Lambda:**
- Virtual functions: No servers to manage
- Limited by time - **short executions**
- **Run on-demand**
- **S**caling is automated
- Benefits:
    - **Easy** pricing
        - **Pay per request and compute time**

        - Free tier of 1000K AWS Lambda requests and 400K GB-seconds of compute time.
    - Integrated with the whole AWS suite of services.
    - **Pay per calls and duration**
    - **Event driven**: functions get invoked by AWS when needed.
    - Integrated with many programing languages
    - Easy monitoring through AWS CloudWatch.
    - Easy to get more resources per function ( upto 10GB of RAM)
    - Increasing RAM will also improve CPU and network.

**Lambda language support:**
- Node js, python, java, C# ruby custom runtime API
- Lambda Container Image
    - The container image must implement the lambda runtime API

**Amazon API Gateway:**
- **Ex: building serverless API**
- CRUD operations between Lambda and Dynamo DB both are serverless
- If we want client to connect with lambda, we should use API Gateway

Client< ———(Rest API)——>API GATEWAY <——(proxy requests)——>Lambda <——(CRUD)—> DynamoDB

- **API gateway** : Fully managed service for developers to easily create, publish, maintain, monitor and Secure API's
- **Serverless** and scalable
- Support **RESTFUL** APS's and **WebSocket** API's
- Support for security, user authentication, API throttling, API Keys, Monitoring…

**AWS Batch:**
- Fully managed batch processing at any scale.
- Efficiently run 100K of computing batch jobs on AWS
- A "Batch" job is a job with a start and an end
- **Bacth will dynamically launch EC2 instances or spot instances.**
- AWS batch provisions the right amount of compute/memory.
- You submit/schedule batch jobs and AWS Batches does the rest
- **Batch** jobs are defined as **Docker images** and **Run on ECS**
- Helpful for cost optimization and focusing less on the infrastructure.

**Batch VS Lambda:**
**Lambda:**
- Time limit
- Limited runtimes

- Limited temporary disk space
- Serverless

**Batch:**

- No time limit
- Any run time as long as its packaged as a Docker Image
- Rely on EBS/ instance store for disk space
- Managed service.
- Relies on EC2 ( can be managed by AWS)

**Amazon Lightsail:**

- Standalone service
- Can get **virtual servers, storage, databases, and networking in one place.**
- **Low** & predictable pricing.
- Simpler **alternative to using EC2, RDS, ELB, EBS, Route 53.**
- Great for people **with little cloud experience.**
- Can setup notifications and monitoring of your lightsail resources.
- Use cases:
  - Simple web applications (has templates)
  - Websites (templets)
  - Dev/test environment
- Has **high availability** but **no autoscaling**, limited AWS integration.

**Compute Summary:**

- Docker: Container technology to run applications
- ECS : Run docker containers on Ec2 instances.
- **Fargate: run docker containers without provisioning infra**
  - **Serverless offering( no ec2)**
- **ECR: private docker image repository.**
- **Batch:** run batch jobs on AWS across managed EC2 instances.
- LightSail : Predictable and low pricing for simple applications & DB stacks.

**Lambda summary:**

- Lambda is serverles, Function as a service, seamless scaling, reactive.
- **Lambda billing:**
  - **By the time**
  - **By no of invocations(calls)**
- Language support: Many programing languages except (arbitrary) docker, for this we use ECS/ fargate.
- Invocation time = 15 minutes.
- Use Cases: create thumbnails for images uploaded onto s3
  - Run a serverles CRON job.
- **API Gateway :** Expose Lambda functions as HTTP Api.

**Deployments & Managing infrastructures at scale**

- **Cloud Formation:**
- Cloud formation s is a d**eclarative way of outlining AWS Infrastructure for any resources**
- **In this  template you can say:**
- I want a security group
- I want two ec2 instances using this security group.
- I want S3 bucket.
- I want a  ELB in front of these machines.

- Then CloudFormation creates those for you in the **right order**, with the **exact configuration** that you specify.
- **Benefits:**
- **Infra as a code** : no resources are manually created, which is good for control.
- Changes to the infrastructure are reviewed through code.
- Cost:
    - Each resource within the stack is tagged with an identifier so you can easily see how much a stack costs tou.
    - **Can easily estimate the cost o**f resources using CloudFormation template.
    - **Saving strategy :** in Dev you could automation deletion of templates at 5PM and recreate at  8AM safely**.**
- Productivity :
    - Ability to destroy and recreate an infrastructure on the cloud on the fly.
    - Automated generation of Diagram for your templates.
    - Declarative programing ( no need to figure out ordering and orchestration)
- Dont reinvent the wheel.
    - Leverage existing templates on the web
    - Leverage documentation
- Support all AWS resources.
    - Can use "custom resources" for resources that are not supported.
- We can visually see **Cloud Formation** set up in **Application composer** like graph/images, how they are connected.
- Its helpful to create same resources in diff accounts, environments…

**AWS Cloud development kit (CDK):**

- Define your cloud infrastructure using a familiar language:
    - JS, Python, Java, .NET
- The code is "compiled" into a CloudFormation templates (JSON/YAML)
- **You can therefore deploy infrastructure and application runtime code together.**
    - Great for Lambda functions
    - Great for Docker containers in ECS/EKS
- CDK application, using the CDK CLI,is  transformed into CloudFormation

templates and can be used to deploy our infrastructure.

**Bean stalk overview:**

- When we deploy web application in AWS, we follow " 3-tier architecture"
- Users talk to ELB ( that could be in multiple in AZ), then ELB will forward traffic to multiple EC2 managed by ASG.
- & these EC2 need to score data some where and use database RDS,
- And if they need to have an in-memory database for an in-memory cache, then they can also use ElastiCache to store a retrieve the session data or the cached data.
- Being a cloud developer, you dont wanna manage all this manually and you want to deploy this as a code, Here comes **Elastic beanstalk.**
- **Elastic Bean stalk is a developer centric view of deploying an application on AWS.**
- So Beanstalk from a cloud perspective is a platform as a service or PaaS because we just worry about the code.
- Its all in one view that's easy to make sense of
- We still have full control over the configuration.
- **Beanstalk = platform as a service (PaaS)**

**Elastic Beanstalk:**

- Managed service.
    - Instance configuration / OS is handled by Beanstalk
    - Deployment strategy is configurable but performed by Elastic beanstalk.
    - Capacity Provisioning
    - Load balancing, auto scaling
    - Application health monitoring & responsiveness
    - Application code is the responsibility of the developer.
- **3** Architecture models:
    - Single instance deployment : good for dev env
    - **LB +**ASG : great for prod and pre prod web applications.
    - **ASG only**: Great for non-web apps in production
    - Support many platforms
        - Go, java, .net, node js, php, python…
        -

- **Health monitoring:**
    - Bean stalk has full monitoring suite available within the service.
    - There is **Health agent** on each **EC2 instance with in beanstalk** which pushes metrics to cloud watch.
    - Checks for app health, publishes health events.

**Code deploy:**

- **Used mainly for applications/servers upgrades from V1 to V2**
- We want to deploy our application automatically
- Works with EC2 instances
- Works with on-premie servers
- Hybrid service
- **Servers / Instances must be provisioned and configured ahead of time with the CodeDeploy Agent.**

**Code commit: Deprecated after July 2024.**

- Assume there is GitHub integration.
- Before pushing application code to servers, you need to store it somewhere.
- Developers usually store code in repository, using GIT Technology
- A famous public offering is GitHub, AWS competing product is CodeCommit.
- CodeCommit:
  - Source-control service that h**osts Git Based repositories,**
  - Makes it easy to **collaborate with others on code**
  - Code changes are automatically **versioned**.
- **Benefits:**
  - Fully managed
  - Scalable & highly available
  - Private, secured, integrated with AWS.

**Code Build:**

- Code building service in the cloud
- Compiles source code, run, tests and produces packages that are ready to be deployed.
- CODE COMMIT <—-(Retrieve code)—CODE BUILD ——(Build Code)——> ready to deploy artifact.
- Benefits
  - Fully managed, serverles
  - Continuously scalable & highly available
  - Secure
  - Pay as you go pricing - only pay for build time

**Code Pipeline:**

- **Orchestrate** the different steps to have the code automatically pushed to production.
- Pipeline ( code => build => test => provision => deploy )
- Basis for **CICD** ( continuous integration and continuous delivery)
- Benefits:
  - Fully managed compatible with code commit, codeBuild, CodeDeploy, Elastic Beanstalk. Cloud Formation, GitHub, .
  - Fast delivery and rapid updates.

**Code Artifact:**

- Software packages depend on each other to be built ( also called code dependencies )) and new ones are created.
- **Storing and retrieving code depends is called Artifact Management.**
- Traditionally we need to set up our own artifact management system.
- In AWS **CodeArtifact** is a secure, scalable, cost-effective **artifact management for software development.**
- Works with common dependency management tools like: Maven, Gradle, NPM, yarn, twine, pip, NuGet.
- **Developers and CodeBuild can retrieve dependencies straight from CodeArtifact.**

**——————————————————————————————————————————————**

**AWS Systems manager (SSM)**

- Helps to manage **Ec2** and **on premises** systems at scale
- **Hybrid AWS** Service.
- Get optional insights about the state of your infrastructure.
- Suite of 10+ products.
- Most important features:
  - **Patching** automation for enabled compliance.
  - Run **commands** across an **entire** fleet of **servers**.
  - Store parameter configuration with the SSM parameter store.
- **Works for Linux, Windows, MacOS,**

**How System manager works?**

- We need to **install** the **SSM agent** onto systems we control.
- It's a small program running on the background.
- Installed by default on Amazon Linux AMI & some Ubuntu AMI
- All the SSM agents on EC2's would be reported to **SSM Service.**
- Can use SSM Service to run commands& patch & configure our servers.

**System manager - SSM Session Manager**

- Allows to start a secure shell on EC2 and on-premise servers.
- **No SSH access, bastion hosts, or SSH keys needed.**
- **No port 22 needed( better security)**
- EC2 (SSM agent)<—(execute commands)—– Session manager service <—–-(IAM Permissions)—User
- Supports Linux, MAC, Windows
- Send session log data to S3 or cloud watch logs

**Hands on:**

**- Launch ec2 instance**

- **Attach IAM instantce profile in advanced details**
- **Got to " AWS systems Manager"**
- **Fleet manager : Here all the Ec2 instances associated with SSM reflects**

**here.**

**Systems manager parameters store:**

- Secure storage for configuration and secrets
- API keys, passwords, configurations
- Serverless, scalable, durable, easy SDK.
- Control access permissions using IAM
- Version tracking & encryption

**Deployment Summary:**

ClouFormation: (AWS only)

- **Infra as code**, works with almost all aws resources
- Repeat across regions & accounts

Beanstalk: (Aws only)

- **PaaS:** limited to few programing languages or DOCKER.
- Deploy code consistently with a known architecture. EX: ALB + EC@+RDS

Code deploy:( Hybrid)

- Deploy & upgrade any application onto servers

Systems Manager: (Hybrid): Patch, configure, run commands at scale.

**Developers Services- summary**

- Code commit: store code in private git repository (version controlled)
- Code build : build and test code in AWS
- Code deploy: deploy code onto servers.
- Code pipeline: Orchestration of pipeline ( from code to build to deploy)
- Codeartifact: store software packages / dependencies on AWS
- Codestar: unified view fr allowing developers to do CICD and code
- Cloud9 : Cloud IDE with collab
- AWS CDK: define your cloud infra using programing language.

**Leveraging AWS Global Infrastructure**

Why make a **global application?**

- Global application is an application deployed in multiple geographies.
- On AWS, this could be **Regions** and/or **Edge locations**.
- **Decresaed latency:**
  - **Latency =** time takes for a network packet to reach a server
- **Disaster recovery:**
  - If one region goes down , other would serve the need even with little more latency
- **Attack protection:** Distributed global infra is harder to attack.

- **We deploy applications in AWS regions.**

**Global Applications in AWS:**

- **Global DNS: Route 53:**
  - Great to route users to teh closest deployment with least latency.
  - Great for disaster recovery strategies.
- **Global Content Delivery Network (CDN) : CloudFront**
  - Replicate part of application to AWS edge locations,- decrease latency
  - Cache common requests - Improved User experience - decreased latency
- **S3 Transfer Acceleration:**
  - Accelerate global uploads and downloads into S3.
- **AWS Global Accelerator:**
  - Improve global application availability and performance using AWS global network

**Amazon route 53:**

- Route 53 is a **managed DNS**. (Domain Name System)
- DNS= collection of rules & records which helps clients understand how to reach a server through URL's
- In AWS most common records are
  - www.google.com => 12.34.56.78 == A record(ipv4)
  - www.google.com => 200 l:0db8:85a3:0000:0000:8a2e:0370:7334 == AAAA IPv6
  - Search.google.com => www.google.com ==CNAME :hostname to hostname
  - example.com => AWS resource ===Alias (ex ELB, CloudFront,S3, RDS..)

**How does DNS work?**

- **SIMPLE Routing policy (Basic): No health check**
- If we have application server on EC2 instance, if a web browser want to request it,
- There will be Route53 in middle
- Web browser send DNS request ( my app.mydomain.com ) to route 54, thenRoute 53 send back IPv4 A record : hostname to IP.
- Then the web browser send HTTP request with IP got from route 53, then from ec2, broker will get HTTP response.
- **WEIGHTED ROUTING Policy: Health check**
- Allows to distribute traffic to multiple EC2 instances.
- Will assign weights to EC2 instances.
- **Latency trotting policy: Health check**
- If we have Ec2's in different regions,
- DNS will route the users to nearby instances based on latency.
- Goal = minimize latency
- **Failover routing policy: Health check**
- We will have primary EC2 and failover EC2

- DNS who health checks on primary, if any health issues, then redirects the traffic failover.

**Route53: Hands on**

- First we need to register domain
- Click register domain
- Choose domain name (Stephane-ccp.com)$12 a year
- Ince done, go to Hosted Zones.( we will get 2 DNS records)
- Create EC2 instances. 1 in US and 1 in Ireland
- Go to DNS
- Create a record, record name = www.stphane-ccp.com
- Record type = A
- Value = public IP of one EC2
- Rating plicy: choose routing policy. Ex: Latency
- Choose the Ec2 instance region.
- Give some record ID
- Similarly do for record 2 with other ec2 instance.

**AWS ClouFront**

- **CDN =** Content delivery network.( for static content)
- Improve **read performance**, content is **cached** at th**e edge locations.**
- Since the content is cached all over the world, user will expereince lower latency.
- Improves User experience.
- **216 point of presence globally** (edge locations)
- DDoS protection ( because worldwide) , integration with shield, AWS Web Application Firewall.
- **CloudFront origins: 2 (S3, Custom origin)**
  - **S3** : for distributing files and caching them at the edge
  - Enhanced security with CloudFront **Origin Access Control (OAC)**
  - OAC is replacing Origin Access identity (OAI)
  - CloudFront can be used as an ingress ( to upload files to S3)
  - Bucket would be protected with Origin Access Control + bucket policy
  - **Custom origin (HTTP):**
  - Application Load Balancer
  - EC2 instance
  - S3 website ( must enable bucket as static s3 website)
  - Any HTTP backend you want.
  - When client sends GET request to edge location => iif it dont have in cache, => go to origin to get the results. Then store it in EDGE location, so that for next users, it can directly respond with out going to origin.

**CloudFront VS S3 Cross Region Replication:**

- **ClouFront:**

  - Global edge network
  - Files are cached for TTL (may be a day)
  - Great for **static conten**t that must be available everywhere.
- **S3 Cross Region Replication:**
  - Must setup for each region you want replication to happen
  - Files are updated in near-real time
  - Read only
  - Great for **dynamic content** that needs to be available at low-latency in few regions.

**Cloudfront hands-on:**

- Create a bucket first, and upload few files.
- Cloudfront->create adistribbution -> choose s3 bucket in origin domain( you can modify it)
- ->origin access = origin access control settings->
- Click create control setting->just choose name->create ->default root object = index.html
- And click create distribution.
- Next copy the Bucket policy frm the the page and
- Paste it in bucket policy.
- Or you can find policy in distribution origin , edit , then you can find bucket policy
- Next we can copy domain name and will be accessible everywhere.

**S3 Transfer Acceleration**

- **S3** bucket is linked to one region
- If we are looking to transfer files from US to Australia, we will will upload vastly to US edge location over the public and then through private AWS we will move it internally from edge location of USA to S3 bucket in Australia,
- Only used to upload/download the objects in S3 which is far.

**AWS Global Accelerator:**

- Improve global application **availability** and **performance** using the AWS global network.
- Leverage AWS internal network to optimize the route to your application (60% improvement)
- **2 Anycast IP** are created for your application and traffic is sent through **edge locations.**
- Edge locations send traffic to your application.

**Global accelerator vs CloudFront:**

- Both use AWS global network and edge location around the world
- Both integrate with AWS shield for DDoS protection.
- **CloudFront - Content delivery network**

    - Improve performance for you cacheable content (images/videos)
    - Content is s**erved at the edge**
- **Global accelerator:**
    - No caching, proxying packets at the edge to applications running in one or more aws regions
    - Improve performance for applications over TCP/UDP
    - Good for HTTP use cases that require **static ip** address.
    - Good fro HTTP use cases that required deterministic, fast regional failover.

**AWS Outposts:**

- **Hybrid cloud:** Business keeps an on-premise infra long with cloud infra
- **2 ways of dealing with IT Systems.**
    - 1 for AWS cloud (Using AWS console, CLI, API's)
    - One for on-premise infra
- **AWS Outposts are " Server racks"** that offer same aws infra services, api's, tools to build own application on-premise just as in the cloud.
- AWS will setup and manage **"Outposts Racks"** within your on-premise infra and you can start leveraging AWS services on-premises.
- You are responsible for "outposts rack" physical security.
- **Benefits:**
    - Low latency access to on-premise systems.
    - Local data processing
    - Data residency
    - Easier migration from on-premises to the cloud.
    - Fully managed service
    - **Services that work on outposts:**
        - **EC2, EBS,S3, EKS, ECS, RDS, EMR**

**AWS Wavelength:**

- **Wavelength zone:** are infra deployments embedded within telecommunications providers datacenter at the edge of the **5G** networks.
- Brings AWS services to the edge of the **5G networks.**
- Ex: ec2, ebs, vpc
- Ultra-low latency applications using 5G
- Traffic does not leave the communication service providers (CSP) network.
- High bandwidth and secure connection to the parent AWS region.

**AWS Local Zones:**

- Place AWS compute, storage database and other selected AWS services closer to end users to run latency-sensitive applications.
- Extend VPS to more locations " Extention of an AWS region"
- Compatible with EC2, RDS, ECS, EBS, Elastic Cache, Direct Connect
- Ex:AWS region (Virginia) local zones Boston, Chicago, Dallas, Houston, Miami

**Global Application architecture:**

- **Single region, single AZ**
    - No high availability
    - badGlobal latency
    - Easy to set up
- **Single region, multiple AZ**
    - High availability
    - Not good latency
    - Little bit difficult to set up
- **Multi region, active-passive**
    - On active and one passive instances in different regions
    - Active on has read/write
    - Passive has only read
    - Good global reads latency
    - Still bad global write latency
    - Moderate Difficult tto set uo
- **Multi region, active-active**
    - Multiple active ec2's on multiple regions
    - All are with read/write capacity
    - Good read/write latency
    - Set up is more difficult.

**Cloud integrations:**

- When we start deploying applications, they will need to communicate
- 2 patterns of application communication.
- 1 Synchronous communication
- 2 Asynchronous/ event based

**Synchronous communication:**

- Applications directly talk to each other, if one got issue other would be effected.

**Asynchronous:**

- Here applications dont talk directly there would be a queue in middle, so both can be scaled independently.

- Always better to **decouple** applications.
    - Using SQS: Queue model
    - Using SNS: pub/sub model
    - Using kinesis : real time data streaming model
- These servers can scale independently from application

**Amazon SQS- Simple queue service:**
**What is queue?**

- There can be multiple producers who sends messages to QUEUE, once msgs are stored in queue, this could read by consumers.
- **Amazon SQS -Standard queue:**
    - Oldest AWS offering (over 10 years)
    - **Fully managed (**serverless), use to decouple applications.
    - Scales from 1 message per second to 10,000's per second
    - Default retention of messages : **4 days max of 14 days.**
    - **No limit** on no of msgs can be in queue
    - Msgs are deleted after they read by consumers.
    - Low latency (<10 ms on publish and receive)
    - Consumers share the work to read messages and scale horizontally

**Amazon SQS- FIFO queues:**

- Fifo = first n first out( order of msgs in queue_
- Messages are processed in order by the consumer.

**Amazon Kinesis:**

- **Kinesis =** real-time big data **streaming**.
- Managed service too collect, process, analyze real-time streaming data at any scale.
- **Kinesis data streams :** low latency streaming to ingest data from multiple sources.
- **Kinesis data firehose: Load** streams into **S3**, **redshift**, **elastic** search.
- **Kinesis data Analytics:** perform real-time **analytics** on streams using **SQL**.
- **Kinesis video streams:** monitor realtime video streams for analytics/ML

**Amazon SNS:**

- **Simple Notification service.**
- To send one message to many receivers.
- Direct integration: ex : buyig service to { email notification, fraud service, shipping service, SQS queue}
- **PUB/SUB :**
    - Buying service-> SNS service-> all down services.
- " Event **publishers** ": Only send message to one SNS topic.
- "Event **subscribers**": As many we want to listen to SNS notification.
- Each subscriber **will get all the messages.**
- SNS => publish=> { SQS, Lambda, Kinesis firehouse, emails, sms & mobile notifications, http endpoints}

**Amazon MQ: ( message broker service)**

- SQS, SNS are " could-native" services;
- On-premises may use open protocols like: MQTT, AMQP..
- **When migrating to cloud:** Instead of re-engiineering the application to use

SQS, SNS, we can use MQ

- MQ: = Managed message broker service for  RabiitMQ, ActiveMQ(on-premise things)
- Does not “ scale” like SQS/SNS
- Runs on servers, can run in Multi-AZ with failover.
- MQ has both queue feature : SQS and topic feature :SNS

**Cloud Monitoring:**

- Cloud watch provides metrics for all services in AWS
- Metric is a varibale to monitor (cpu utilization, network..)
- Metrics have timestamps.
- Can create dashboards
- **Important metrics:**
    - **EC2 instances:** CPU utilization, status checks, networks (Not RAM)
        - Default metric : very 5 min
        - Option for detailed monitoring: 1 minute
    - EBS Volumes: Disk reads/writes
    - S3 bucket: Bucket size bytes, no. Objects, all requests.
    - Billing: total estimated charge( only in US-east-1)
    - **Service limits:** How much you’ve been using a service API
    - **Custom metrics: push your own metrics.**
- **Cloud watch alarms:**
    - Used to **trigger notifications** for any metrics.
    - Alarm actions:
        - **Auto scaling:** increase/decrese instances desired count
        - EC2 Action: stop, terminate, reboot, or r**ecover an EC2 instance.**
        - **SNS notifications:** send a notification into an SNS topic.
    - Can choose period on which to evaluate alarm.
    - Create a **billing alarm o**n the cloud watch billing metric.
    - Alarm states: OK, Insufficient data, Alarm

- **Cloud watch logs:**
    - Can collect log from :
        - Elastic beanstalk : collection of logs from application
        - ECS: collection from containers
        - AWS lambda: collection from function logs
        - Cloud trail based on filter
        - **Cloudwach log agents:** on ec2 machines or on-premise servers.
        - Route 53: Log DNS queries
    - Enable real-time monitoring of logs
    - Adjustable CloudWatch Logs retention.

- **Cloud watch logs for EC2:**
    - Default: no logs from ec2 instance goto cloud watch

    - Need to run cloud watch agent on EC2 to push the log files you want.
    - Make sure EC2 IAM permissions are correct.
    - **Cloud watch log agent can be setup on-premises too.**

**Amazon EventBridge: (formerly CloudWatch Events)**

- **Schedule:** CRON jobs(scheduled scripts)
- Schedule every hour=> trigger script on lambda function.
- **Event Pattern:** Event rules to react to a service doing something.
- **Schema registry:** model event schema
- Can **archive events** sent to an event bus.
- Ability to r**eplay archived events.**
- **Skipped handson**

**AWS CloudTrail:**

- **Provide governance, compliance, and audit for AWS account.**
- CloudTrail I enabled by default
- Get an **history of events/ API calls made within your AWS accoun**t by:
    - Console
    - SDK
    - CLI
    - AWS Services
    -
- Can put logs from CloudTrail into CloudWatch Log/ S3
- **A trail can be applied to all regions(default) or a single region.**
- If a resource is **deleted** in AWS, investigate **CloudTrail first.**

**AWS X-Ray:**

- **Visual analysis of application.**
- Debugging in prod
    - Test locally
    - Add log statement everywhere
    - Re-deploy in prod
- Log analysis is hard bcz we have to combine everything
- Debugging: one big monolith “Easy” distributed service “hard”
- No common view of entire architecture
- Adavantages:
    - Troubleshooting performances(bottlenecks)
    - Understand dependencies in micro service architecture
    - Pinpoint service issues
    - Review request behavior
    - Find errors and exceptions
    - Are we meeting time SLA?

    - Where I am throttled
    - Identify users that are impacted.

**CodeGuru:**

- **ML po**wered service for automated **code reviews** and application performance recommendations.
- **Provides 2** functionalities.
    - **CodeGuru Reviser:** Automated code reviews for static code analysis ( development): once code is uploaded into codecommit/github , code guru will check each and every line and provide feedback if any memory leaks, something has seen before.
    - **CodeguruProfiler:** Visibility / recommendation about application performance during runtime (production): when you are build/test code, CodeGuruProfiler detect and optimize expensive lines of code pre-prod. When deployed: can measure performance and cost improvements in production.
- Helps understand runtime behavior of application
- Features:
    - Identify & remove code inefficiencies
    - Improve application performance
    - Decrease compute cost
    - Provide heap summary
    - Anatoly detection
- Support application running on AWS/ on-premise
- Minimal overgead on application.

**AWS health dashboard - service history:**

- Shows all regions , all **services** health
- **Shows** historical info for each day
- Has an RSS feed you can subscribe to

**AWS Health dashboard- your account**

- Previously called AWS personal Health Dashboard.
- Provides **Alerts and remediation guidance** when AWS is experiencing **events that may impact you.**
- **Serivice dashboard =** status of AWS Services.
- **Account Health dashboard** = personalized view in to the performance and availability of AWS services underlying your AWS resources.
- Dashboard displays **relevant and timely information to help you manage events in progress and provides proactive notification** to help you plan for **Scheduled activities.**
- Can aggregate data from an entire AWS organization.
- Click on Bell icon -> event log.

**Summary:**

- CloudWatch:
    - Metrics : monitor the performance of AWSservices & billing metrics.
    - Alarms: Automate notification, perform EC2 action, send SNS based on metric.
    - Logs: collect log filles from EC2 instances, servers, lambda functions.
    - Event( event bridge): react to events in AWS , trigger a rule on schedule.
- CloudTrail: Audit calls made with in your AWS account.
- cloudTrail Insights: Automated analysis of your CloudTrailEvents.
- X-ray: trace requests made through your distributed application.
- AWS health dashboard: status of AWS services across region:
- AWS Accounts health dashboard: AWS events that impact your infra.
- CloudGuru: automated code reviews and application performance management.

**VPC:**

IP adresses in AWS:

- Ipv4 - internet protocol version 4( 4.3 billion addresses)
- Public Ipv4 - can be used on internet: makes it publicly reachable
- Ec2 instance get new public IP address every time you stop and then start it.
- When we stop the instance, ipv4 address is going to **release,** and of you start again, it gets new public IP adress at launch.
- Private IP in the form : 192.168.1.1 can be used in internal AWS( LAN) VPC and is fixed for EC2 instances even if you start. Stop them.
- **Elastic IP** - allows you to attach a fixed **public ipv4** adress to EC2 instance.
- Note: All public ipv4 on AWS = $0.005 per hour ( including EIP)
- **IPV6** - internet protocol version 6 (3.4 * 10^38 addresses)
    - E**very IP** in ipv6 is **public**

VPC: - virtual private cloud to deploy regional resources.

- Subnet : allows to partition VPC, associated with AZ.
- Public subnet : subnet that's accessible from the internet.
- Private subnet : Not accessible from internet.
- To define access to the internet and between subnets, we use **Route Tables.**
- Internet Gateway & NAT Gateways:
    - Internet Gateway helps our VPC instances connect with internet.
    - Public subnet have route to the internet gateway.
    - **NAT Gateway( AWS-managed):** for private subnets, we no need to give access and but sometimes we need to have internet connection for updates.
    - **NAT instances(Self-managed)** : allow your instances in your private

Subnets to access the internet while remaining private.

- We will create a ROUTE from private subnets->NAT Gateway -> this allows to get-> internet connectivity.
- We can check at CIDR.xyz : IP and CIDR range visualizer.
- Default: multiple subnets would be created, for each subnet: we get ipv4.
-

**Network ACL & Security Groups:**

- **NACL( network ACL) : subnet level**
    - A firewall which control traffic from and to **subnet**.
    - Can have **ALLOW/DENY** rules.
    - Are Attached at the **subnet** level.
    - Rules only include IP addresses
    - **Is stateless:** Return traffic must be **explicitly** allowed by rules.
    -
    -
- **Security Groups: EC2 level**
    - A firewall that controls traffic to and from **EC2 instance**
    - Can have only **ALLOW** rules.
    - Operates at **instance** level
    - Rules include IP addresses and other security groups.
    - Is stateful:: Return traffic **AUTOMATICALLY** allowed by rules.

**VPC Flow Logs:**

- Capture info about IP traffic going into your interfaces.
    - **VPC** flow logs
    - **Subnet** flow logs
    - **Elastic network interface** flows logs.
- Helps to monitor & troubleshoot connectivity issues:
    - Subnet to internet
    - Internet to subnet
    - Subnet to subnet
- Capture network info from AWS managed interfaces too: Elastic Load Balancers, Elastic Cache, RDS, Aurora, etc..
- VPC Flow logs data can go to S3, CloudWatch Logs, Kinesis Data Firehouse.
- **VPC Peering:**
    - Connect two VPC, privately using AWS's network
    - Make them behave as if they were in the same network.
    - Must not have overlapping CIDR (IP range)
    - VPC Peering connection is not **transitive**( must be established for each VPC that need to communicate with one another) : if A-B has peering and B_C has peering, A,C can not connect automatically.
    - Handson:
    - VPC->FLOW logs-> create flow log

VPC Endpoints:

- End points allow you connect to AWS services using PRIVATE NETWORK instead of public www network.
- Gives enhanced security and lower latency to access AWS services.
- If a ec2 in private subnet want to access S3/ DynamoDB, for this we create VPC Endpoint Gateway to connect to S3/dynamo only but **privately**.
- **VPC Endpoint Gateway: S3 & DynamoDB**
- **VPC Endpoint interface: the rest.**
- 
- VPC -> ENDPOINTS->Create End point ->

**AWS PrivateLink (VPC Endpoint Services):**

- Most secure & scalable way t**o expose a service to 1000's of VPC's.**
- Does not require VPC peering, Internet gateway, NAT, route tables.
- Requires a **network load balancer** ( service VPC) **and ENI** (Customer VPC)

**Direct Connect & Site-to-Site VPN:**

- To connect on premise Data center to AWS,
- **Site to site VPN:**
  - Connect on-premise VPN to AWS
  - Connection is automatically encrypted.
  - **Goes over Public internet.**
  - Set up easy but goes over public Internet even though encrypted.
  - On premises: must use a **Customer Gateway (CGW)**
  - AWS: Must use a **Virtual Private Gateway (VGW)**
- **Direct Connect:**
  - Establish physical connection between on-premise and AWS.
  - Connection is private , secure, fast.
  - Goes over **private network**
  - Takes atleast a month to establish.

**AWS Client VPN:**

- Connect from your computer using OpenVPN to your private network in AWS and on-premises.
- Allows you to connect to your EC2 instances over a private IP ( Just as of you were in the private VPCC network)

**Transit Gateway Overview:**

- Network topologies can become complicated.
- **Transit gateway:**
  - For having t**ransitive peering** between thousands of VPC and on-

premises, **hub-and-spoke (star) connection.**
  - **One** single gateway to provide this functionality
  - Works with Direct Connect Gateway, VPN connections.

**VPC summary:**

- **VPC :** virtual private network.
- **Subnets:** Ties to an AZ, network partition of the VPC.
- **Internet Gateway**: At the VPC level , provide internet access.
- **NAT Gateway:/** Instances : Give Internet access to private subnets.
- **NACL:** Stateless, Subnet rules for inbound and outbound.
- **Security groups:** Stateful, operate at the EC2 instance level/ ENI
- **VPC peering:** Connect 2 VPC with non overlapping IP ranges, transitive.
- Elastic IP: Fixed public IPv4, on going cost if in non use.
- **VPC Endpoint:** Provide private access to AWS Services within VPC.
- **PrivateLink:** privately connect to a service in a 3rd party VPC.
- **VPC Flow logs**: Network traffic logs
- **Site to site VPN:** VPN over public internet between on-premise DC and AWS.
- **Client VPN:** Open VPN connection from your computer into you VPC.
- **DirectConnect:** direct private connection to AWS.
- **Transit Gateway:** Connect thousands of VPC and on-premises networks together.

**Security & Compliance Section:**

- **AWS Shared Responsibility Model:**
  - AWS Responsibility: Security **of** Cloud.
    - Protecting infrastructure( hw, sw, facilities, and networking) that runs all the AWS services.
    - Managed services like S3, DynamoDB, RDS, etc.
  - **Customer responsibility:**
  - Security **in** the the cloud.
  - **For EC2 instances,** customer is responsible for management of guest OS, including security patches and updates, network configuration, IAM
  - Encrypting application data
  - **Shared Control:**
  - Patch management, Configuration management, awareness & Training.
  - RDS:
    - AWS:
      - Manage underlying EC2 instances, disable SSH access.
      - Automate DB patching, OS patching,
      - Audit underlying instance and disks & guarantee it function.
    - User:
      - Check ports. IP, security group inbound rules in DB's SG
      - In-database user creation and permission.
      - Creating a database with or without public access.
      - Ensure parameter groups or DB is configured to only allow SSL

connections.
        - DB encryption setting.
    - S3:
        - AWS:
            - Guarantee you get unlimited storage, encryption
            - Ensure separation of data between diff customers.
            - Ensure AWS employees can’t accee the data.
        - USer:
            - Bucket configuration, policy/ public setting.
            - IAM User & roles.
            - Enabling encryption.

**DDOS Attack?**

- Distributed denial-of-service.
- Attacker would create multiple master servers and these servers launch multiple bots, all of which access application at a time and the application go down.
- How to protect from this?
- **AWS Shield standard:** protect against DDOS attack for website & application for all customers at no cost.
- **AWS Shield advance:** premium: 24/7 DDoS protection.
- **AWS WAF**: filter specific requests based on rules.
- **CloudFront and Route 53:**
    - Availability protection using global edge network.
    - Combined with AWS shield, provides attack mitigation at the edge
- Be easy to scale-leverage **AWS Auto Scaling.**

**AWS Shield standard:**

- Free service that is activated for every AWS customer.
- Provides protection from attacks uch as SYN/UDP floods, reflection attacks and other layer 3/layer 4 attacks.

**AWS Shield advanced:**

- Optional DDoS mitigation service.( $ 3,000 per month)
- Protects against more attacks on **EC2, ELB, Cloudfront, Global accelerator, Route 53.**
- 24/7 access to DDoS response team.

**AWS WAF:**

- Protects web application from common web exploiits(layer 7-http)
- Deploy on **ALB, API gateway, cloufront**
- Define web ACI ( Access control list):
    - RULES CAN INCLUDE IP ADRESSES, HTTP HEADERS, HTTP BODY, URI STRINGS.

    - Protects rom common attack - SQL injection, cross site scripting
    - Size constraint, geo-match (block countries)
    - Rate-based rules( to count occurrences of events)- for DDos protection.
    -
-

**AWS Network Firewall:**

- Protects entire VPC.
- From layer 3 to layer 7.
- Any direction, you can inspect
    - VPc to vPc traffic
    - Outbound to Internet
    - Inblund to internet
    - To/from direct connect & set to site VPN.

AWS Firewall manager:

- Manage **security rules** in all accounts of **an AWS Organization:**
- Security policy : common set of security rules.
    - VPC security groups for EC2 , ALB…
    - WAF rules
    - AWS shield advanced
    - AWS network firewall.
- Rules are applied to new resources as they are created in future accounts of an organization: good for compliance.
-

**Penetration testing on cloud:**

- Trying to attack your own infra to test security.
- AWS Customers are welcome to carry out security assessments or penetration test against AWS infra **without prior approval for the 8 services.**
- **Prohibited activities:**
    - DNS zone walking via Amazon route 53 hosted zones.
    - Denial of Service (DoS), Distributed Denial of Service (DDoS), Simulated DoS. Simulated DDoS.
    - Port flooding
    - Protocol flooding
    - Request flooding (login request flooding, AP request flooding)
    - For any other simulated events, contact: AWS security team.

**Encryption with KMS & cloud HSM:**

- 2 types of encryptions happening in AWS:
- **Encryption at rest, Encryption in transit.**
- **At rest:**
    - Data stored or archived on a device.

        - On  a hard disk, RDS instance, S3 Glacier Deep Archive…etc.
- **In Transit:(in motion)**
    - Data being move from one location to another.
        - Transfer from on-premise to AWS , EC2 to DynamoDB etc.
        - Means **data transferred on the network.**
- We want to encrypt data on both states to protect it.
- For this we leverage **encryption keys.**

**AWS KMS( Key Management Service):**

- Anytime you hear "encryption" for an AWS service, its mostly KMS
- KMS = **AWS manages the encryption keys for us.**
- **Encryption opt-in:**
    - EBS volumes: encrypt volumes.
    - S3: server-side encryption of obejcts.
    - Redshift database:  encryptn of data
    - RDS database: encryption of data.
    - EFS drives : encryption of data.
- **Encryption automatically enabled.**
    - **CloudTrail logs**
    - **S3 glacer**
    - **Storage gateway**

**Cloud HSM: (** Hardware Security Module )

- **KMS =>AWS manages** the software for encryption.
- Cloud HSM = AWS provisions encryption hardware.
- Dedicated **hardware**: (HSM = Hardware Security Module)
- **You manage your encryption keys entirely.**
- HSM = tamper resistant

**Types of KMS Keys:**

**Customer Managed Key:**

- Create , manage and used by the customer, can enable or disable.
- Possibility of rotation policy. ( new key generated every year, old key preserved)
- Possibility to bring-your-own-key

**AWS Managed key:**

- Create , manage and used by the customer behalf by AWS.
- Used by AWS Services ( AWS S3, EBS,REDSHIFT)

**AWS Owned key:**

- Collection of CMK's that an AWS service owns and manages to use in multiple accounts.
- AWS can use those to protect resources in your account ( But you can't view the keys)

**CloudHSM Keys( Custom keystone):**

- **Keys generated** from your own **CloudHSM hardware device.**

- Cryptographic operations are performed within the CloudHSM cluster.

**AWS Certificate manager (ACM):**

- Lets you easily provision, manage, and deploy **SSL/TLS certificates.**
- Used to provide in-flight encryption for websites (HTTPS)
- Supports both public and private TLS certificates.
- Free of charge for public TLS certificates.
- Automatic TLS certification renewal.
- Integration with (load TLS certificates on )
  - ELB
  - CloudFront distributtions
  - API's on API gateway.

**AWS secrets manager:**

- Newer service , meant for storing secrets.
- Capability to force **rotation of secrets** every x days.
- Automate generation of secretes on rotation ( uses lambda)
- Integration with **Amazon RDS** ( mysql, postgressql, aurora)
- Secrets are encrypted using KMS
- Mostly meant for RDS integration.

**AWS Artifact:**

- Portal that provides customers with on-demand access to **AWS Compliance documnetaton and AWS agreements.**
- **Artifact reports:**- Allow you to download AWS security and compliance documentation from 3rd party auditors like AWS Iso certifications, payment card industry, system & organization control (SOC)
- **Artifact agreements:** Allows to revise, accept, track the status of AWS agreements such as Business Associate addendum ( BAA) or the Health insurance portability and accountability act( HIPAA) for indiviidual account / organization.
- Can be used to support i**nternal audit or compliance.**

**Amazon guard Duty:**

- Intelligent **threat discovery** to protect AWS account.
- Uses ML algorithms  for anomaly detection.
- One click to enable ( 30 day trail) no need to install software.
- Input data :
  - **CloudTrail Logs** - unusual API calls, unauthorized deployments.
    - **CloudTrail Management Event**s: create VPC, subnet, create tail
    - **CloudTrail S3 data events**: get objects, list objects, delete object…
  - VPC Flow Logs- unusual traffic, unusual IP addresses.
  - DNS Logs- compromised EC2 instances sending encoded data within DNS

    queries.
    - Optional Features: EKS audit logs, RDS & Aurora, labda…
- Can setup **EventBridge rules** to be notified in case of findings.
- Eventbridge rules can target AWS :lambda or SNS
- **Can protect against cryptocurrency ( has a dedicated "finding" for it)**

**Amazon inspector:**

- Automated **security assessments**
- For **Ec2** instances
    - Leveraging AWS **System Manager (SSM) agent**
    - Leveraging against **Unintended network accesibility**
    - Analyze the running **OS** against **known vulnerabilities.**
- For **Container images push to Amazon ECR**
    - Assessment of container images as they are pushed.
- For **Lambda Functions.**
    - Identifies software vulnerabilities in function code and package dependencies.
    - Assessments of functions as they are deployed.
- Reporting and integration with AWS security Hub.
- Send findings to Amazon Event Bridge

- Continuous scanning of infra only when needed
- Package vulnerabilities ( EC2, Lambda, ECR) -database of CVE
- Network reachability (CE2)
- A risk score is associated with all vulnerabilities for prioritization.

AWS Config:

- Helps with **Auditing and recording compliance of your AWS resources.**
- Helps **record configurations and changes over time.**
- Possibility of storing configurations data into S3(analyzed by Athena)
- Questions that can be solved by AWS config:
    - Is there an unrestricted SSH access to my security groups?
    - Do my buckets have any public access?
    - How has my ALB configuration changed over time.
- You can receive alerts ( SNS notifications) for new changes
- AWS config is per-region service.
- Can be aggregated across regions and accounts.

**Amazon macie:**

- Amazon Macie is a fully managed **data security and data privacy service that uses ML and Pattern matching** to discover and protect your **sensitive data in AWS.**
- Helps to identify and alert you to sensitive data, such as personally identifiable info.(PII)

- If we have data in S3-> Macie analyzed this and notify us in Eventbridge, can integrate event bridge to SNS for notifications.

**AWS Security Hub:**

- **Central security tool** to manage security **across several AWS accounts** and **automate security checks.**
- Integated dashboards showing current security and compliance status to quickly take actions.
- Automated aggregates alerts in predefined and personal findings format from various AWS services.
- Must first enable AWS Config Service.

**Amazon Detective:**

- GuardDuty, Macie and Securiity Hub are used to identify potential security issues or findings.
- Sometimes these findings require deep **RCA** and **take action- complex** process.
- Amazon detective: analyzes, investigates and quickly identifies the root cause of the security issues or suspicious activities using ML and graphs.
- Automatically collects and process events from VPC Flow logs, CloudTrail, guard Duty and created a unified view
- Produces visualizations dn with details and context to identify Root cause.

**AWS Abuse:**

- Reports suspected AWS resources used for abusive or illegal purpose.
- Abusive and prohibited behaviors are:
  - Spam : undesired Mal from AWS owned Ip's,websites,forums,
  - Port scanning: sending packets to your ports to discover unsecured ones.
  - Dos/DDOS attacks: AWS owned IP addresses attempting to overwhelm or crash servers.
  - Intrusion attempts: logging in to your resources.
  - Hosting objectionable or copyrighted content- distributing illegal or copyrighted content without consent.
  - Distributing malware: aws resources distributing softwares to harm computers or machines.

**Root User Privileges:**

- Root user = account owner ( created when account is created)
- Has complete access to AWS services and resources.
- Lock away you AWS account root user access keys.
- Do not use root account for everyday tasks, even administrative tasks.
- Actions can be performed **only by the root user:**
  - **Change account settings** ( name, email,roo pwd, root user access keys)
  - Close your **AWS account**

  - Restore IAM user permissions.
  - C**hange/cancel AWS support plan**
  - **Register as a seller in your Reserved Instance Marketplace.**
  - Configure S3 bucket to enable MFA
  - Edit/delete S3 bucket policy that includes invalid VPC Id ot VPC end point.
  - Signup for GovCloud

**IAM Access Analyzer:**

- Find out which resources are shared externally outside of trust zone:
  - **S3**
  - **IAM**
  - KMS keys
  - Lambda functions and layers
  - SQS queues
  - Secret manager secrets.
- Define ZONE OF TRUST = AWS account /organization
- Access outside of this trust accessed=> reported as findings

**Security & compliance summary:**

- Shared responsibility on AWS
- **Shield**: Automatic DDoS Protection + 24/7 support for advanced.
- **WAF:** Firewall to filter incoming requestes based on rules
- KMS: encryption keys managed by AWS
- **CloudHSM:** Hardware encryption, we manage encryption keys
- **AWS certificate manager:** Provision, manage and deploy SSL/TLS certificates.
- Artifact: Get access to compliance reports such as PCI,ISO
- **GuardDuty: Find malicious behavior with VPC, DNS, & cloudTrail logs.**
- **Inspector: Find software vulnerabilities in EC2, ECR images, lambda functions.**
- Network firewall: protect VPC against network attacks.
- Config: track config changes and compliance against rules.
- Macie: Find sensitive data ( EX:PII data) in Amazon S3 buckets.
- CloudTrail: Track API calls made by users within account.
- **Security Hub:** Gather security finding from multiple AWS accounts.
- **Amazon detective:** find the root cause of security issues or suspicious activities.
- **AWS Abuse:** Report AWS resources used for abusive or illegal purpose.
- **Firewall Manager:** manage security rules across an organization (WAF, Shield)

**Machine Learning:**

**Amazon Recognition:**

- Find o**bjects, people, text, scenes** in **image** and **videos** using ML
- **Facial analysis** and **facial search** to do user verification, people counting.
- **Create a DB of “ familiar faces” or** compare against celebrities
- Use cases:
  - Labeling
  - Content moderation
  - Text detection
  - Face detection and analysis
  -

**Amazon Transcribe: audio to text**

- **Convert speech to text**
- Uses Deep learning Process called “ Automatic speech recognition “ to convert speech to text quickly and accurately.
- Automatically remove Personally identifiable info (PII) using **redaction**
- Automatic Language identification for multi-lingual audio.
- Use cases:
  - Transcribe customer service calls
  - Automate closed captioning and subtittles.
  - Generate metadata for mediia assets to create a fully searchable archive.

**Amazon Polly: Text to audio**

- Turns text to speech using DL
-

**Amazon translate:** (language translation)

- Natural and accurate language translation
- Allows you to localize content-such as website and applications - for international users, and to easily translate large volumes of text efficiently.

**Amazon Lex & Connect:**

- Amazon lex :( Same technology that powers Alexa)
- Automatic speech recognition: to convert speech to text.
- Natural language understanding to recognize the intent of text, callers.
- Helps build chatbots, call center bots.

**Amazon Connect:**

- Receive calls, create contact flows, Cloud-based **virtual contact center.**
- Can integrate with other CRM systems or AWS.
- No upfront payments, 80% cheaper than traditional contact center solutions.

**Amazon Comprehend:**

- For **NLP**
- Fully managed and serverles
- Uses ML to find insights and relationships in text.
  - Language of the text
  - Extract key phrases, places, people, brand, or events.

    - Understand how positive or negative the text is
    - Analyze text using tokenization and parts of speech.
    - Automatically organizes a collection of test files by topic
- **Use cases:**
    - Analyze emails to find leads to positive/negetive expereince
    - Create and group articles by topics that comprehend will uncover.

**Amazon sagemakker:**

- Fully managed service for developers/ data scientists to build ML models.
- Difficult to do all processes in one place + provision servers.
- Sage maker is one spot for labelling, training, fine-tuning and deploying ml models.

**Amazon forecast:**

- Fully managed service that use ML to deliver high accurate forecasts.
- Ex. Predict future sales of a raincoat
- 50 % more accurate than just data
- Reduce forecasting time from months to months
- Use cases: product demand pricing, planning..
- Historical time series data.

**Amazon Kendra: Document search by indexing.**

- Fully managed Document search service powered by ML
- From text, pdf, html, …
- Documents from any data source ==> Indexing( knowledge index powered by ML amazon Kendra)==>Answer user queries
- Natural language capacity.
- Learn from user interactions/feedbacks to promote preferred results **(incremental learning**)
- Ability to manually fine-tune search results

Amazon **personalize**:( **apps with personalized recommendations**)

- Ex: personalized product recommendation/re-ranking, customized direct marketing
- Provide recommendations on the next product to buy
- Sametechnology used by amazon
- Implement in days ( dont take much time,) no need to build, train, deploy

Amazon **Extract**: to extract text

- Extract text, handwriting, data from scanned documents using A and ML
- From forms, tables, pdfs, images
- Use case: invoices, medical records DL,..

**Account management & billing & Support.**

- **AWS Organizations:**
  - Global service
  - **Allows to manage multiple accounts.**
  - Manaage multiple accounts, main is master account
  - Bill would bee paid by the master account. Remain are : child accounts.
  - Pricing benefits from aggegratted usage( ec2,s3..)-single payment
  - Other Childs can share resources for saving
  - Pooling pf reserved EC2 instance for optimal saving
  - **API is available to automate account creation**
  - Restrict account privileges using **service control policies (SCP)**
- **Multi accounts strategies**
  - Create accounts per **dept**, per cost center, per environment, based on **regulatory restrictions*** using SCP) for better **resource isolation** (ex VPC) to have s**eparate per-account service limits**, isolated account for logging.
  - **Multi account vs oneAccount MultiVPC ( Trade off)**
  - Use tagging standards for billing purpose
  - Enable cloudTrail on all accounts and send logs to central s3 account
  - Send cloud watch logs to central logging account.
- **Organizational units (OU)**
  - Based on business units, environment lifecycles, project-based…
  - Root OU contain master account, and can create other OU like, dev, prod, finance, HR…
  - **Service control policies (SCP):**
    - Whitelist/black list IAM actions
    - Applied at the **OU or account** level
    - Does not apply to master account.
    - SCP is applied to all the **users** and **roles** of the account including **root**.
    - Does not effect service-linked roles
    - **SCP must have explicit allow (**does not allow anything by default)
    - Use cases:
      - Restrict access to certain services ( ex. Can’t use EMR)
- **AWS organization - consolidated billing:**
  - When enabled:
    - **Combine usage:** combine usage across all AWS accounts in organization to share the **volume, pricing, reserved instances** and saving plans discounts.
    - **One Bill:** get one bill for all aws accounts in organization
- **AWS Control tower;**
  - Easy way to **set up** and **govern a secure** and **compliant multi-account AWS environment** based on best practices.
  - **Benefits:**

        - Automate set up in few clicks
        - Automate ongoing policy management using guardrails.
        - Detect policy violations and remediate them
        - Monitor compliance through an interactive dshboard
        - AWS **control tower** runs on the top of AWS organizations:
            - Automatically set up AWS organization to organize accounts and implement SCP's
            -

**AWS Resource Access manager ( RAM):**

- Share resources that you own with other AWS accounts.
- Share with any account or within your organization
- Avoid resource duplication
- Include: aurora, vpc subnets, transit gateway, route 53, ec2,…

**AWS Service catalog:**

- New users can create many stacks not in line with organization.
- Some users want a quick **self-service portal** to launch a set of **authorized products** pre-defined **by admins**.
- Include VM, DB, storage option…
- Enter aws catalog
- **Admin :** create product using cloudFormation template—> and create portfolio( collection of products)—> control this using IAM permissions to access portfolios
- **Users:** access product list (authorized by IAM) —> and launch provisioned products(ready to use properly configured properly tagged)

**Pricing models: 4**

- **Pay as you go:** pay for what you use, remain, agile, responsive, meet scale demands.
- **Save when you reserve:** Minimize risk, predictably manage budgets, comply with long term requirements.
- **Pay less by using more: volume based discounts**
- **Pay less as AWS grows:**

- **Compute pricing - EC2:**
    - **Only pay for what you use.**
    - **No. Of instances**
    - **Instance configuration:**
        - Physical capacity
        - Region
        - Os and software
        - Instance type

        - Instance size.
    - **ELB** running time and amount of data processed.
    - On demand instances:
        - Minimum of 60s
        - **Pay per second (linux/windows) or per hour(others)**
    - **Reserved instances:**
        - Up to 75% discount
        - 1/3 Y commitment
        - All upfront, partial, no
    - **spot:**
        - **90% discount**
        - Bid for insured capacity
    - **Dedicated:**
        - **Ondemand**
        - **Reserved for 1/3 year**
- **Lambda:**
    - Pay per call
    - Pay per duration
- **ECS:**
    - EC2 launch type model: no additional fees, you pay for AWS resources stored and created in your application.
- **Fargate:**
    - Fargate launch type model : pay for vCPU and memory resources allocated to application in your containers.
- **S3:**
    - **Storage class:**
    - No of size of objects
    - No and type of requests
    - Data transfer out
    - S3 transfer accelartatiion
    - Lifecycle transitions
- EBS:
    - Volume type
    - Storage volume in GB per month provisioned.
    - IOPS:
    - Snapshots
    - Data transfer outt
- RDS:
    - Per hour billing
    - Engine, size, memory class
    - Purchase type (on demand/ committed)
    - Backup storage
    - No of ip op requests per month
    - Deployment type ( single AZ/multiple AZ)

- Content delivery - cloud Front:
  - Price is different for regions
  - Aggregated for each edge loc
  - Data transfer out
- Network code:
  - **Same region:**
  - If 2 EC2's in one AZ, they can communicate freely over private IP
  - 2 EC2's in differnt AZ , using private IP is cheaper than public IP
  - Diff region
  - 2 ex's in different regions communication over private IP is costlier than above 2 options but less than using public IP
  - Note: **better to use private IP, if possible in same AZ.**

**Savings Plan:**

- saving on your cost on AWS, which is called Savings Plan.
- Commit certain $ amount per hour for 1/3 years.
- **Easy way for longtime commitments**
- **EC2 savings plan:**
  - Unto 72% discount compared to on-demand
  - Commit to usage of individual instances families in region ( eg. C5 or m5)
  - Regardless of size, AZ, OS
- **Compute saving plan:**
  - Unto 66% discount
  - Regardless of family, region, os, compute options
  - Compute options: EC2, Fargate, lambda
- **ML Saving plan: sage maker…**

**AWS Compute optimizer:**

- **Reduce costs and improve performance** by recommending optimal AWS resources for your workloads.
- Helps to choose optimal configs and right size workloads
- Uses ML to analyze **resources configs** and their **utilization cloud watch metrics.**
- **Supported resources:**
  - Ec2 instances, ASG;s, volumes, Lambda functions,
- Lower the costs by unto 25%

**Billing and costing tools:**

- **Estimate costa in the cloud:**
  - **Pricing calculator**
    - Estimate cost for your solution architecture
    -
- **Tracking costs in the cloud:**

  - Billing dashboard
  - Cost allocation tags
  - Cost and usage reports
  - Cost explorer
- **Monitoring against cost plans:**
  - Billing alarms
  - Budgets

**AWS Billing dashboard:**

- **Cost allocation tags:** helps to track costs in a detailed way
- **AWS generated tags:**
  - Automatically applied to resources you created.
  - Starts with prefix **AWS**
- User defined tags:
  - Starts with prefix **user**

- **Cost and Usage reports:**
  - Most **comprehensive** set of cost and usage data on AWS
  - Includes, metadata about services, the pricing, and the reservations.
  - cost report will give you all the AWS usage for each service category used by an account,and it's IAM users in hourly or daily line items,as well as any tags that you have activated for cost allocation purposes.
- **Cost Explorer:**
  - Visualize, understand and manage AWS costa and usage over time.
  - Create custom reports that analyze cost and usage data.
  - Analyze data in high level
  - Choose optimal **savings plan**
  - **Forecast usage unto 12 months based on previous usage**

**Billing Alarms in Cloud Watch:**

- **Billing data metric is only stored in cloudWatch us-East1**
- Billing data is for worldwide
- Its for actual cost not for projected one
- Can be used as simple alarm not as powerful as AWS budgets.

**AWS Budgets:**

- **Create budgets and send alarms when costs exceeds the budget**
- **4 types of budgets: usage, cost, reservation, saving plan**
- For reserved instances (RI's)
  - Track utilization, supports ec2, elastic ache, res, redshift
  - **Unto 5 SNS** notifications per budget

**AWS Cost anomaly detection:**

- Monitors cost and usage using ML to detect unusual sends

**AWS Service Quotas:**

- **Notify when you’re close to a service quota value threshold.**
- Create cloud watch alarms on the service quotes console
- Ex: lambda concurrent execution
- **Can request quota increase from AWS before reaching limit,**

**Trusted advisor:**

- High level aws account assessment
- Business and enterprise support plan
-

**AWS Support plans pricing :**

**Basic Support plan : free**

- **Customer Service & communities:**
- 24/7 access to customer service, documentation, whitepapers, and support forums
- **AWS Trusted Advisor:**
- Access to **7 core** trusted advisor checks and guidance to provision your resources following best practices to increase performance and improve security.
- AWS Personal dashboard : A personalized view of health of AWS services and alerts when your resources are impacted.

**AWS Developer support plan:**

- All basic support plan +
- **Business hours emails access t**o cloud support associates.
- Unlimited cases / unlimited contacts
- Based on case severity : will get different response times.
  - General guidance: < 24 business hours
  - System impaired < 12 b.hours

**Business Support plan**

- Intended to be used of you have **production workloads.**
- **Trusted advisor = full set of checks + API access**
- **24 // 7 phone, email and chat access to cloud support engineers.**
- Unlimited cases / unlimited contacts
- **Access to infra event management for additional fee.**
- Case severity / response times:
  - Generic guidance < 24 BH
  - System impaired < 12 BH
  - Production system impaired < 4 Hours
  - Production system down: < 1 hour

**Enterprise on ramp support plan:**

- Intednd to be used if you have Production / business critical workloads
- All of business support plan +
- Access to pool of **Technical account managers (TAM)**

- **Concierrge support team ( for billing & account best practices)**
- **Infrastructure event management , well-architected & operation reviews**
- Case severity / response times:
    - **…**
    - Production system impaired < 4hours
    - **Prod system down < 1hour**
    - **Business critical system down < 30 mins**

**AWS Enterprise support plan ( 24/7):**

- **Intedned to be used** if oyuhave mission critical workloads
- **All of business support plan +**
- Access to a designated technical account manager (TAM)
- **Concierrge support team ( for billing & account best practices)**
- **Infrastructure event management , well-architected & operation reviews**
- Case severity / response times:
    - **…**
    - Production system impaired < 4hours
    - Prod system down < 1hour
    - **Business critical system down < 15 mins**

**AWS STS ( Security token service):**

- Enable you to create t**emporary, limited-privileges credentials** to access your AWS resources.
- Short-term credentials : you configure expiration period.
- **Usecases:**
    - **Identity federation:** manage user identities in external systems, and provide them with STS tokens to access AWS resources.
    - **IAM roles for cross/same account access.**
    - **IAM roles for amazon EC2:** provided temporary EC2 credentials to access AWS resources.

**Amazon Cognito:**

- Identity for your **web and mobile applications users** ( potentially millions)
- Instead of creating them as a IAM user, you can create a user in cognito.

**Directory Service Overview:**

- **What is Microsoft Active Directory?**
- Found on any windows server with AD Domain service.
- DB of objects: User accounts, computers, printers, file shares, security groups.
- Centralized security management, create account, assign permissions.

**AWS Directory services:**

- **Aws managed Microsoft AD**
    - Create your own AD in AWS, manage users locally, supports MFA
    - Enable “trust” connections with your on-premise AD
- **AD Connector:**
    - Directory gatewaay(proxy) to redirect to on-premise AD, supports MFA
    - Users are managed on the on-premise AD
- Simple AD:
    - AD-compatible managed directory on AWS.
    - Can not be joined with on-premise AD
    - ALL the abovethings is not for CCP certification.

**AWS IAM Identity center (successor to AWS single sign-on)**

- **One login (single sign) for all your**
    - AWS accounts in AWS org
    - Business cloud applications(sales force, box, Microsoft 365)
    - SAML2.0 enabled applications
    - EC2 windows instances.
- **Identity providers:**
    - Built in identity store in IAM identity center
    - 3rd party Active Directory(AD) , onelogin, okra

**Other AWS Services:**

- **Amazon workspaces: Managed Desktop as a service (DaaS)** solution to easily **provision windows or linux desktops.**
    - Great to eliminate management go on-premise VDI
    - Fast and quickly scalable to thousands of users.
    - Secured data- integrate with KMS
    - Pay-as-you-go service with monthly or hourly rates.
    - Here:
    - **Fully managed VDI and desktop available**
    - Users connect to the VDI and open native or WAM applications
    - **Workspaces are on demand or always on**
- **Amazon AppStream 2.0:**
    - **Desktop application streaming service.**
    - Deliver to any computer, without acquiring, provisioning infrastructure.
    - **Application is delivered from within a web browser.**
    - Here:
    - Stream a desktop application to web browser ( no need to connect to VDI)
    - Works with any device (that has a web browser)
- **AWS IoT Core:**
    - IoT stands for “internet of things: - the network of internet connected devices that are able to collect and transfer data.

  - **AWS IoT core allows to connect IoT devices to cloud.**
  - Serverless , secure, & **scalable to billions fo devices and trillions of messages.**
  - Applications can communicate with devices even when they are not connected.
- **AMazon Elastic transcoder:**
  - Convert media files stored in S3 into media files in the formats required by consumer playback devices ( phones)
  - Benefits:
  - Easy to use
  - Highly scalable- can handle large volume of files and large file sizes.
  - Cost effective: duration based pricing model.
  - Fully managed & secure , pay for what you use.
- **AWS AppSync:**
  - **Store and sync data across mobile and web apps in real time.**
  - Makes use of GraphQL( mobile technology from Facebook)
  - Client code can be generated automatically
  - Integrations with Dynamo DB /lambda
  - Real time subscriptions
  - **Offline data synchronization** (replaces cognate Sync)
- **AWS Amplify**
  - **Set of tools and services that helps you to develop and deploy scalable full stack web and mobile applications.**
  - Can manage Authentication, storage, API(rest,graphql) ci/cd, pubs, analytics, AI/Ml predictions, Monitoring, source code from AWS, gitHub etc…
- **AWS Application composer:**
  - **Visually design and build serverrless applications** quickly on AWS.
  - Using drag and drop features.
  - Infra- as- code.
- **AWS Device Farm:**
  - Fully **managed service that tests your web and mobile apps against desktop browsers** , real mobile devices and tablets.
  - Run tests concurrently on multiple devices
  - Ability to configure device settings ( gps, wifi, bluetooth)
- **AWS Backup:**
  - Fully managed service to centrally manage and automate backups across AWS services.
  - Oon-demand and schedule backups
  - Supports PITR ( point in time recovery)
  - Retention periods, lifecycle management, backup policies…
  - Cross-region backup
  - Cross-account backup..
- **Disaster recovery strategies:**

- **Backup and restore ( cheapest)**
    - We backup our data on cloud and after any disaster, we would restore it.
- Pilot Light:
    - We would just run core functions in cloud.ex DB
    - It has minimal set uo and cost more than just backup and restore.
- Warm standby:
    - More expensive
    - Full version of app is on cloud, but with minimal size.
    - For disaster recovery, we just need to increase size.
- **Multi site/ hot-site:**
    - Whole app is on cloud at full size.
    - You have a deployment that's ready to use.
    - **Most expesive**
    -

**AWS Elastic Disaster Recovery:**

- Used to be named "**cloud endure disaster recovery**"
- Quickly and easily **recover** your physical, virtual, and cloud based servers into AWS.
- **Ex.** Can protect your DB, applications from **ransomware attacks.**
- Conttinuous **block** level replication for your servers.

**AWS Data Sync:**

- Move large amount of data from on-premises to AWS.
- Can sync to s3, efs,..
- Replication tasks can be scheduled hourly, daily, weekly
- Replication tasks are **incremental** after the first full load.

——————————

**Cloud Migration strategies: 7Rs**

- **Retire:**  turn off things that dont need
- Helps with reducing surface areas for attacks.
- Save cost, may be unto 10% to 20%
- **Retain:**
- Do nothing for now (
- Security, data compliance, performance, unresolved decencies.
- No business value to migrate, mainframe, or mid-range .
- **Relocate:**
- Move apps from on-premises to its cloud version
- Move EC2 instances to a different VPC, AWS account, or AWS region.
- **Rehost: Lift and shift**
- Simple migrations by re-hosting on AWS ( applications, databases, data)
- Migrate machines to AWS cloud.
- No cloud optimization being done, application is migrated as is

- Could save as much as 30% on cost.
- **Ex:** Migrate using AWS Application Migration Service.
- **Replatform : Lift and reshape**
- Migrate your database to RDS
- Ex: migrate your application to elastic beanstalk
- Not changing the core architecture, but leverages some cloud optimizations.
- Save time and moneybag moving to a fully managed service or serverless.
- **Repurchase: drop and shop**
- Moving to a different product while moving to the cloud.
- Often move to SaaS platform
- Expensive in short term but quick to deploy
- **Ex:** CRM to salesforce, Hr to workday
- **Re-factor / re-architect:**
- Reimagining how the application is architected using CloudNative features.
- Driven by the need of business to add features and improve scalability performance security and agility
- Move from monolithic application to micro-services.
- **Ex:** move an application to serverless architecture use AWS S3.

**AWS Application Discovery Service :**

- Plan migration projects by gathering information about on-premises data centers.
- Server utilization data and dependency mapping are important for migration.
- **Agentless Discovery ( AWS Agentless Discovery Connector)**
    - VM inventory, configuration, and performance history such as CPU, memory, disk usage
- **Agent-based Discovery ( AWS Application Discovery Agent)**
    - System configuration, system performance, running processes and details of network connections between systems.
- Resulting data can be viewed within AWS Migration hub.

**AWS Application Migration Service (MGN):**

- The "AWS evolution" of CloudEndure Migration, replacing AWS Server migration Service (SMS)
- Lift and Shift (rehost) solution which simplify **migrating** applications to AWS.
- **Converts** physical,, virtual and cloud based servers to run natively on AWS.
- **Ideally we install aws replication agent on on-premise servers, then create staging environment in cloud, with low cost ec2 instances and EBS volumes, once its**
    - **Now on the day we are ready to perform cut over, we will move to production from stagin in cloud with the sizes we want so there would be minimal downtime.**

**AWS Migration Evaluator:**

- Helps to build data driven business case for migration to AWS.
- Provides clear baseline of what your organization si running today
- Install agentless collector to conduct broad-based discovery.
- Analyze current state, define target state then develop migration plan.

**AWS Migration Hub:**

- Central location to collect servers and application inventory data for the assessment, planning, and tracking of migration to AWS.
- Helps accelerate migration to AWS , automate Lift and Shift.
- **AWS Migration Hub orchestrator:** provides pre-built templates to save time and effort migrating enterprise apps ( ex> Sap Microsoft SQL server)
- Supports migration updates from MGN( application migration service and (DMS) database migration service.

**AWS Fault injection simulator (FIS):**

- Fully managed service for running fault injection experiments on AWS workloads
- Based on Chaos Engineering: stressing application by distruptive events like increase in CPU or memory and observing how system responds.
- **Helps to uncover hidden bugs.**
- Supports Ec2, ECS, EKS, RDS

**AWS Step functions:**

- Build serverless **visual workflow** to **orchestrate lambda function.**
- **Features:** sequence, parallel, conditions, timeouts, errr handling..
- Possibility oof implementing human approval features.
- **Use cases:** order fulfillment, data processing, web applications.

**AWS Ground stations:**

- Fully managed service that lets you control **satellite communications**, process data and **scale satellite operations**.
- Provides global network of satellite ground stations near AWS resiions.
- Allows to download satlight data tto AWS VPC in seconds.

**AWS pinpoint:**

- Scalable **2-way (outbound/iinbound) marketing communication service.**
- Supports: email, push voice, and in-app messaging.
- Ability to segment and personalize messages with the right content to customers.
- Usecsaes: run campaigns by sending marketing, bulk, transactional SMS messages.
- **Versus amazon SNS orr Amazon SES**
  - In SNS & SES, you managed each message’s audiience, content, an

    delivery schedule.
  - **In** Amazon pinpoint, you create message templates, delivery schedule, highly targeted segments aand full campaigns.

**AWS Architecting & Ecosystem section:**

**Well architected framework general guiding principles.**

- Stop guessing your capacity needs
- Test systems at production scale.
- Automate to make architectural experimentation easier.
- Allow for evolutionary architectures
  - Design based on changing requirements
- Drive architecture using data
- Improve through game days
  - Simulate applications for flash sales days

**AWS Cloud best practices  : Design principles:**

- Scalability: Horizontal and vertical
- **Disposable resources:** Servers should be disposable and easily configurable.
- **Automation :** serverless, infrastructure as a service, auto scaling/..
- **Loose Coupling:**
  - Monolith are applications that do more aand more over time , become bigger.
  - Break it down into smaller, loosely coupled components.
  - A chnage or failure in one component should not cascade other componenets.
- Think in services not Servers:
  - Dont just use EC2: EC2 would be to translate whatever you have on premises in the cloud but think in terms of services
  - what managed services can you use databases, serverless that can make your life a lot easier.
  - Use managed services, databases, serverless etc…

1. **AWS well architected framework:  6 pillars**
   1. Operational excellence
   2. Security
   3. Reliability
   4. Perofrmance efficiency
   5. Cost optimization
   6. Sustainability

- All the above are not to balance or tradeoffs but they are a **SYNERGY(** interaction or cooperation  giving rise to a whole that iis greater than sum of its parts)
- **Pillar 1 : Operational excellence:**

    - Includes **ability to run and monitor systems** to deliver business value and to conttinully improve supporting process and procedures.
    - Design principles:
        - Perform operations as code - infra as code
        - Make frequent, small,, reversible changes-
        - Refine operations procedures frequently
        - Anticipate failure
        - Learn from all operational failures.
        - Use managed services- to reduce operational burden
        - Implement observability for actionable insights - performance, reliability, cost…
        - AWS Services:
            - Prepare: AWS Cloud Formation, AWS Config
            - **Operate: AWS Cloud formation, config, cloud trail, cloud watch, x-ray**
            - Evolve: Cloud Formation, Code build, code commit, code deploy, code pipeline
- **Pillar 2: Security:**
    - Includes ability to protect information, systems, and assets while delivering business value through risk assessments and mitigation strategies.
    - **Design principles:**
        - **Implement a strong identity foundation** - Centralize privilege management and reduce reliance on long-term credentials - principle of least privilege - IAM
        - **Enable traceability** - integrate logs and metrics with systems to automatically respond and take action.
        - **Apply security to all layers:** like edge network, VPC, subnet, load balancer, every instance, OS, and application.
        - **Automate security best practices.**
        - **Protect data in transit and at rest** : encryption, tokenization, and access control.
        - **Keep people away from data:** reduce / eliminate the need for direct access for manual processing of data.
        - **Prepare for securitty events:** run-incident responses, response simulations, and use tools with automation to increase speed for detection, investigation,, and recovery.
        - AWS Services:
            - Identity and access management : I'm, AWS_STS, MFA token, AWS organizations
            - Detective controls: Config, CloudTrail
            - Infra protection: ClooudFront, VPC, Shield, WAF, Inspector
            - Data Protection: KMS, S3, ELB, EBS, RDS
            - Incident response: IAM, CloudFormation, CloudWatch Events.
- **Pillar 3: Reliability:**

- **Ability to recover from infra/service disruptions, dynamically acquire computing resources to meet demands, mitigate disruption such as mis configs / transient network issues.**
- Design principles:
    - Test recovery procedures: Use automation to simulate different failures before they occur.
    - Automatic recovery from failure: Anticipate and remediate failures before they occur.
    - Scale horizontally to increase aggregate system availability: Distribute requests across multiple, smaller resources to ensure that why dont share a common point of failure.
    - Stop guessing capacity: Maintain the optimal level to satisfy demand without over/under provisioning - Use auto scaling.
    - Manage change in Automation: Use automation to make changes to infrastructure.
    - AWS Services:
        - Foundations: IAAM, VPC, Limits, Trusted advisors
        - Change management : Auto Scaling, CloudWatch, Cloud Trail, Config
        - Failure management: Backups. cloudFormation, S3, S3 Glacier, route 53.

- **Pillar 4: Performance Efficiency:**
    - Includes ability to use computing resources efficiently to meet system requirements and to maintain that efficiiency as demand changes and technologies evolve.
        - Design Principles:
            - Democratiize advanced technologies- advance technologies become service and hence you can focus on product development.
            - Go global in minutes: Easy deployment in multiple regions.
            - Use serverles architectures: avoid burden of managing servers
            - Experiment more often: Easy to carry out comparative testing
            - Mechanical sympathy: Be aware of all services.
            - AWS Services:
                - Selection: Auto scaling, Lambda, EBS, S3, RDS
                - Review: Cloud Fromation
                - Monitorring: CloudWatch, lambda
                - Trade ofs : RDS, Elastic cache, snowball, CloudFront
- Pillar **5: Cost optimization:**
    - Include the ability t run systems to deliver business value at lowest price point.
    - Design principles:

- Adopt a consumption mode: pay only for what you use.
- Measure overall efficiency: use clod watch
- Stop spending money on data center operation - Aws Does the infra part and enables measure return on investment (ROI) -make sure to use tags
- Use managed and application level services to reduce cost of ownership: As managed services operate at cloud scale, they can offer a lower cost per transaction or service.
- AWS Services:
    - Expenditure awareness : Budgets, Cost and usage mreport, Cost explorer, reserved instance reporting.
    - Cost-effective resources: Spot Instance, reserved instance, S3 Glacier.
    - Matching supply and demand: Auto scaling and lambda
    - Optimizing over time: trusted advisor, cost and usage report.

- **Pillar 6: Sustainability:**
  - The sustainability pillar focuses on minimizing the environmental impacts of running cloud workloads
  - Design principles:
    - Understand your impact: Establish performance indicators, evaluate improvements.
    - Establish sustainability goals: Set long-term goals for each workload, model return on investment (ROI)
    - Maximize utilization: Right size each workload to maximize the energy efficiiency of the underlying hardware and minimize idle resources.
    - Anticipate and adopt new, more efficient, hardware and software offerings:
    - Used managed services: Shared services reduce the amount of infra, managed services help automate sustainability best practices as moving infrequent accessed data to cold storage and adjusting compute capacity.
    - Reduce the downstream impact of your cloud workloads: Reduce the amount of energy or resources required to use your services and reduce the need for your customers to upgrade their devices.
    - AWS Services:
      - EC2 auto scaling, serverles offering (lambda, margate)
      - Cost explorer, AWS graviton2, EC2 T instances @ spot instances.
      - EFS_IA, S3, Glacier, EBS Cold HDD volumes.
      - S3 lifecycle configuration, S3 intelligent tiering
      - Amazon Data lifecycle manager
      - Read local, write global: RDS read replicas, Aurora Global DB,

DynamoDB Global table, CloudFront.

AWS Well-Architected Tool:

- Free tool to **review architectures** against 6 pillars well-architected framework and **adopt architectural best practices.**
- How does it work?
  - Select workload and answer questions.
  - Review answers against 6 pillars.

**AWS Customer carbon footprint tool:**

- Track Measure, review, forecast the carbon emissions generated from aws usage
- Helps to meet sustainability goals.
-

AWS Cloud Adaption Framework: ( AWS CAF)

- Its a kind of blank paper
- Helps you build and then execute a comprehensive plan for your digital transformation through innovative use of AWS.
- Created by AWS professional by taking advantage of AWS best practices and lessons learned from 1000’s of customers.
- AWS CAF i**dentifies specific organizational capabilities** that underpin successful cloud transformations.
- AWS CAP groups its capabilities in 6 perspectives:
  - **Business , people, governance, platform, security and operations.**
  - **Business perspective** helps ensure that cloud investments accelerate digital transformation ambitions and business outcomes.
  - **People perspective: serves as a bridge between technology and business**, accelerating the cloud Journey to help organizations more rapidly evolve to a culture of continuos growth, learning, and where change becomes business - as-normal, with focus on culture, organizational structure, leadership, and workforce.
  - **Governance perspective:** Helps you orchestrate your cloud initiatives while maximizing organizational benefits and minimizing transformation-related risks.
  - **Platform perspective**: Helps you build enterprise-guide, scalable, hybrid cloud platform, modernize existing workloads, and implement new cloud-native solutions.
  - **Security perspective:** helps you achieve the confidentiality, ntegritty, and availability of your data and cloud workloads.
  - **Operations perspectives:** Helps ensure that your cloud services are delivered at a level that meets the needs of your business.

- **AWS CAF- Transformation domains**

    - **Technology:**Using cloud to migrate and modernize legacy infra, applications, data and analytics platforms.
    - **Process:** Digitizing, automating, optimizing your business operations.
        - Leveraging new data and analytics platforms to create actionable insights.
        - Using ML to improve your customer service experience.
    - **Organization :** Reimagine your Operating model
        - Organizing your teams around **products and value streams**
        - Leveraging agile methods to rapidly iterate and evolve.
    - **Product:** Reimagine your business model by creating new value propositions(products & services) and revenue models.

- **AWS CAF - Transformation phases:**
    - **Envision:** Demonstrate how cloud will accelerate business outcomes by identifying transformation opportunities and create a foundation for your digital transformation.
    - **Align:** Identify capability gaps across 6 AWS CAF perspectives which results in an Action Plan.
    - **Launch:** Build and deliver pilot initiatives in production and demonstrate incremental business value.
    - **Scale:** expand pilot initiatives tto the desired scale while realizing the desired business benefits.

**AWS Right Sizing:**

- EC2 has many instances, choosing powerful type is not required, since the cloud is **Elastic.**
- Right Sizing is tthe process of matching instance types and sizes to your workload performance and capacity requirements at the lowest possible cost.
- **Scaling up is easy so always start small.**
- It's also the process of looking at deployed instances and identifying opportunities to eliminate or downsize without compromising capacity or other requirements which result in lower costs.
- Its important to right size:
    - Before a Cloud Migration
    - Continuously after the cloud onboarding process( requirements change over time).
    - Cloud Watch,  out explorer, trusted advisor, 3rd party tools can help
- **AWS IQ:**
    - Quickly find professional help fr your AWS projects
    - Engage and pay AWS certtiifies 3rd party experts for on-demand project work,
    - Video-conferencing, contract management, secure collaboration, integrated billing.

- AWS Re: Post: forum to ask questions.
- AWS Managed service (AMS) : provides infra and application support on AWS.
- AMS offers team of AWS experts who manage and operate your infra for security, reliability, and availability.
- Helps orgs offload routine management tasks and focus on their business objectives.
- Fully managed service: AWS Handles activities leek change request, monitoring, patch management, security and backup services.

**Test:**
No of questions: 65 but only 50 are assesses and remaining 15 are for their research
No negative marking
Result: 100- 1000
Pass score = 700
Time : 90 minutes + more 30 minutes ( for non native English speakers)

Domain 1: Cloud Concepts (24% of scored content)
Domain 2: Security and Compliance (30% of scored content)
Domain 3: Cloud Technology and Services (34% of scored content)
Domain 4: Billing, Pricing, and Support (12% of scored content)