**AWS Regions & Avilability Zones:**

* Each region has many availability zones (min = 3, max = 6)
* AZ is one/more datacentres with power, networking & connectivity.
* AZs are separate from each other, so that they’re isolated from disasters.
* Connected with high bandwidth, ultra low-latency networking.

**Choose AWS Region:**

* Compliance - the government wants to store data locally.
* Proximity - deploy close to users for reduced latency.
* Available Services - new Services & regions are not available in every region.
* Pricing - varies from region to region.

**AWS Points of Presence:**

* Amazon has 400+ POP (400+ edge locations & 10+regional caches) in 90+ cities across 40+ countries.
* Content delivered to users with low latency.

**===================================================================================================**

**Scalability & High Availability:**

* Scalability means an application can handle greater loads.
* Kinds of scalability,
  + **Vertical scalability (scale up/down)**
    - Increasing size of instance
    - If app is running on t2.micro, then running it on t2.large is vertical scaling
    - Is common for non-distributed systems such as DBs (RDS, ElastiCache)
    - From : t2.nano - 0.5G of ram, 1vCPU
    - To : u-I2tbl.metal – 12.3TB of ram, 448 vCPUs
  + **horizontal scalability(scale out/in)**
    - increasing no.of instances/systems
    - it implies distributed systems
    - common for web apps/modern apps
      * Auto Scaling Group
      * Load Balancer
* **High availability**
  + Goes hand in hand with horizontal scaling
  + Running same app in atlease 2 datacentres (==availability zones)
  + Goal of high availability is to survive a data centre loss.
    - Auto scaling group multi AZ
    - Load balancer multi AZ

**===================================================================================================**

**IAM: Users & Groups:**

* IAM - Identity & Access Management, Global service.
* Users - are people within your organization & can be grouped.
* Groups - only contain users not other groups.
* Roles - some AWS service will need to perform actions on your behalf.
  + To do so we will assign permissions to AWS services with IAM roles.
  + Similar to users but managed by AWS instead of physical people.
  + Common roles,
    - EC2 instance roles
    - Lambda function roles
    - Roles for cloud formation
* Policies - define permissions fo the users.
  + Inline policies - policy attached to a user directly.
  + If IAMReadOnlyAccess allows s3:GetObject and IAMWriteOnly denies s3:GetObject, the user will not have permission to perform s3:GetObject because the denial takes precedence.
* IAM is a global service.
* Users don’t have to belong to a group & can belong to multiple groups.
* Root account created by default, shouldn’t be used/shared.
* Users & Groups can be assigned using JSON docs called policies.
* Don’t give more permissions than needed.
* IAM security,
  + **IAM Password Policy**
  + **IAM MFA**

**How Users access AWS:**

* Users can access AWS & manage AWS services,
  + **AWS Management Console - protected by MFA+password**
  + **AWS CLI - protected by access key**
  + **AWS SDK (for code) - protected by access key**
* Users can manage their own access keys.
* Access keys are secret, just like password. Don’t share it.
* **Access Key ID - username**
* **Secret Access Key - password**

**IAM Security Tools:**

* **IAM Credentials Report (account level)**
  + Lists all your account users & status of their various credentials.
* **IAM Access Advisor (user level)**
  + Shows the service permissions granted to a user & when those services were last accessed.
  + We can use this to revise policies (by checking which policy is not used)
  + Is accessable by clicking on the user

**IAM Best Practices:**

* Don’t use root account except for AWS account setup.
* One physical user = one AWS user.
* Assign users to groups & assign permissions to groups.
* Create a strong password policy.
* Use MFA.
* Use Roles for giving permissions to AWS services.
* Use access keys for programmatic access (CLI/SDK)
* Audit permissions of account using IAM Credentials Report & IAM access advisor.
* Never ever share access keys.

**Notes:**

* **AWS cloudshell - Terminal provided by AWS.**

**===================================================================================================**

**Amazon EC2:**

* EC2 = Elastic Compute Cloud = Infrastructure as a service
* Mainly consists of,
  + Renting VMs - (EC2)
  + Storing data on virtual drives - (EBS)
  + Distribute load across machines - (ELB)
  + Scaling services using ASG - (ASG)
* EC2 Configuration,
  + OS
  + CPU
  + RAM
  + Storage
    - Network attached (EBS & EFS)
    - Hardware (EC2 Instance store)
  + Network card: speed of the card, public IP address
  + Security Group - Firewall rules
  + EC2 user data - bootstrap script (configure at first launch)
    - Runs only once at instance first start.
    - Script run with root user.

**#!/bin/bash**

**# Use this for your user data (script from top tom bottom)**

**# install httpd (Linux 2 version)**

**yum update -y**

**yum install -y httpd**

**systemctl start httpd**

**systemctl enable httpd**

**echo "<h1>Hello World from $(hostname -f)</h1>" > /var/www/html/index.html**

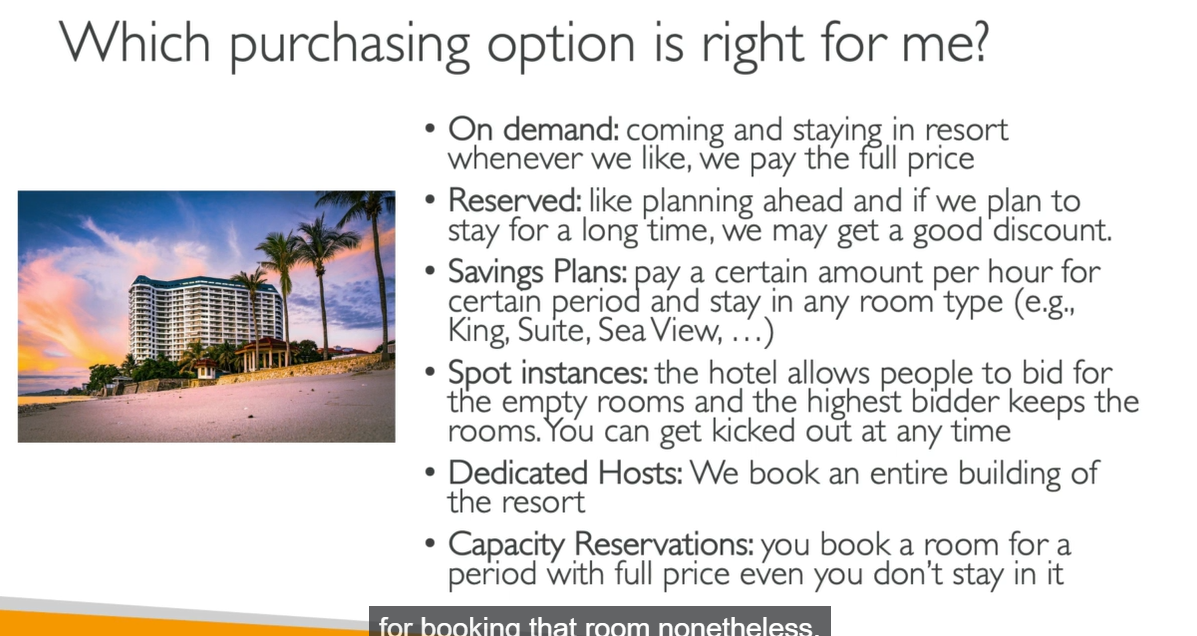
**EC2 Instance types:**

* **General purpose:**
  + suitable for general purpose.
  + t2.micro
  + web servers, code repos
* **Compute optimized:**
  + Suitable for high computing.
  + C5, C6g, C5, C5a, C5n ,C4
  + batch processing, high perf webservers, machine learning, gaming servers, high perf computing
* **Memory optimized:**
  + Suitable for large data sets processing
  + R6g, R5, R%a, R%b, R4, X1e, X1, z1d, ...
  + relational & non-relational DBs, in memory DBs for BI, distributed web scale cache stores
* **Storage optimized:**
  + Suitable for storage-intensive tasks that require high, sequential read & write access to large data sets on local storage.
  + I3, I3en, D2, D3, D3en, H1
  + OLTP systems, cache for inmemory dbs(redis)
  + Relational & NoSQL dbs
  + Datawarehousing apps
  + Distributed file systems

**Security Groups:**

* Fundamental of network security in AWS.
* Controls in/out traffic of EC2 instances.
* SG Only contains allow rules.
* SG rules can reference by IP/SG (security Group)
* Can be attached to multiple instances.
* Locked down to a region/VPC combination.
* Its good to maintain one separate SG for SSH access, since SSH is complicate.
* if app is showing,
  + not accessible (time out) - SG issue
  + connection refused - app error/it is not launched
* Regulates access to ports.
* Ip ranges - IPv4 & IPv6
* Ports to know,
  + 22 - SSH - log into a linux instance
  + 21 - FTP - uploads file into file share
  + 22 - SFTP - uploads file using SSH
  + 80 - HTTP - access unsecured websites
  + 443 - HTTPS - access secured websites
  + 3389 - RDP - log into windows instance

**EC2 purchasing Options:**

* **On-demand instance**
  + Short workload,pay by second (linux/windows), other os (pay by hour), predictable pricing
* **Reserved (1& 3 yrs)**
  + Reserved instances – long workloads
  + Upto 72% discount compared to on-demand
* **Savings plans (1&3 yrs)**
  + Commitment to an amount of usage, long workload
  + Upto 72% discount compared to on-demand
* **Spot instances**
  + Short workloads, cheap, can lose instances (less reliable) if max price is less than current spot price
  + Upto 90% discount compared to on-demand
  + Used for,
    - Batch jobs,data analysis,image processing,distributed workloads
    - Not suitable for critical jobs/databases
* **Dedicated hosts**
  + Book an entire physical server, control instance placement
  + No control over instance placement
  + Can have visibility into them
* **Dedicated instances**
  + No other customer will share your hardware
* **Capacity reservations**
  + Reserve capacity in a specific AZ for any duration
  + No discounts
  + Charged at on demand rate whether you run instances/not
  + Suitable for short term workloads, uninterrupted workloads
* **Which is best,**
* 

**Notes:**

* public ip of an instance changes every time it is restarted.
* Ways to connect to ec2 instance,
  + Using command prompt
  + AWS connect
  + Using putty
* SSH into linux instance using command,
  + **ssh -i key.pem ec2-user@<instance-public-ip>**
  + **ec2-user = instance user name**
* never ever enter access key in ec2 instances, so instead of this we can use IAM Role & attach it to instance.

**EBS Volume:**

* EBS = Elastic Block Store
* Is a network drive, we can attach to your instances while they run.
* Allows your instances to persist data, even after their termination.
* One EBS Mounted to only one instance at a time (CCP = certified cloud practitioner level)
  + But one instance can have more than 1 EBS
* They are bound to a specific availability zone
  + EBS volume in us-east-1a can’t be attached to us-east-1b instance
  + To move volume across, we first need to snapshot it
* Works like a network USB stick (we can remove & attach it to another computer)
  + Since it uses network, there might be bit of latency
  + Can be detached from an EC2 & can be attached to another one quickly
* Delete on instance termination can be done
* Types,
  + gp2/gp3 (SSD) - general purpose, balances price & performance for workloads
    - system boot volumes, Virtual desktops, dev & test envs
    - gp3:
      * min: 3000 IOPS, throughput 125 MiB/s
      * max: 16,000 IOPS, throughput 1000 MiB/s independently
    - gp2:
      * max: 16,000 IOPS, volume & IOPS are linked
      * 3 IOPS per GB, at 5334 GB we are at max IOPS
  + io1/io2 (SSD) - high performance, critical low-latency/high-throughput workloads
    - critical business applications/apps that need more than 16,000 IOPS
    - great for DB workloads
    - max: 64,000 IOPS for Nitro instances & 32,000 for other
    - can increase PIOS independently from storage size
    - io2 have more durability & more IOPS per GiB at same price
    - io2 Block Express (4GiB – 64TiB)
      * sub millisecond latency
      * max: 256,000 with IOPS:GiB ration of 1,000:1
    - supports EBS Multi-attach
  + st1 (HDD) - low cost, for frequently accessed, throughput-intensive workloads
    - 25Gib-16TiB
    - Big data, data warehouses, log processing
    - Max: throughtput 500MiB/s, 500 IOPS
  + sc1 (HDD) - lowest cost, for less frequently accessed workloads
    - also called Cold HDD
    - max: throughtput 250 MiB/s, 250 IOPS
  + only gp2/gp3 & io1/io2 can be used as boot volumes.
* **Multi-Attach,**
  + Attach same EBS volume to multiple EC2 instances in same AZ.
  + Each instance has full read & write permissions to the high performance volume
  + Use case,
    - Achieve higher application availability in clustered Linux apps (ex: Teradata)
    - Applications must manage concurrent write operations
  + Upto 16 EC2 instances at a time
  + Must use a file system that’s cluster aware (not XFS, EXT4 etc…)

**EBS Snapshot:**

* Make a backup(snapshot) of your EBS volume at a point in time.
* While taking a snapshot it is recommended to detach the volume
* Can copy snapshots across AZ/region
  + Create snapshot from volume
  + From snapshot create volume in desired AZ & attach it to instance
* **Features,**
  + **EBS Snapshot Archive**
    - Move snapshot to “archive tier”, 75% cheaper
    - Takes 24-72 hrs for restoring the archive
  + **Recycle Bin for EBS Snapshots:**
    - Deleted snapshots moved to recycle bin
    - Recover period (from 1 day to 1year), we can setup it
  + **Fast Snapshot Restore:**
    - Force full initialization of snapshot & no latency on first use, costs more

**AMI:**

* AMI = Amazon Machine Image
* Are customization of an EC2 instance
  + We can add our own s/w, configuration, os, monitoring, …, & offers faster boot time
* AMIs are built for specific region (can be copied across regions)
* We can launch instances from,
  + Public AMI - AWS provided
  + Own AMI - we make & maintain them
  + AWS Marketplace AMI - AMI someone else made

**EC2 Instance Store:**

* EBS volumes are network drives with good but limited performance.
* For high-performance hardware dish, use EC2 instance store
  + Better I/O performace
  + EC2 instance Store lose storage it they’re stopped
  + Good for buffer/cache/scratch data/temporary content
  + Risk of data loss if hardware fails
  + Backup & replication are our responsibility

**EFS:**

* EFS = Elastic File System
* Managed NFS (network file system) that can be mounted on many EC2
* EFS works with EC2 instances in multi-AZ
* Highly available, scalable, expensive (3x of gp2), pay per use
* Use cases,
  + Content management, web serving, data sharing, wordpress
* Uses NFSv4.1 protocol
* Uses security group to control access to EFS
  + Create separate security group for EFS
* Compatible with Linux based AMI (not windows)
* Encryption at rest using KMS (Key Management Service)
* Uses POSIX file system (linux) that has a standard file API
* File system scales automatically, pay-per use, no capacity planning

**Performance & storage classs:**

* **EFS Scale:**
  + 1000s of concurrent NFS clients, 10GB+ /s throughput
  + Grow to petabyte scale NFS automatically
* **Performance mode (set at EFS creation time):**
  + **General purpose - web server, CMS, etc.. (latency-sensitive)**
  + **MAX I/O - big data, media processing (high latency & parallel)**
* **Throughput Mode:**
  + **Bursting - 1TB = 50 MiB/s + burst of up to 100 MiB/s**
  + **Provisioned - set throughput regardless of storage size**
  + **Elastic - automatically scales throughput up/down based on your workloads**
    - **Upto 3GiB/s for reads & 1GiB/s for writes**
    - **Used for unpredicted workloads**
* **Storage Classes: (standard = regional)**
  + **Storage Tiers (lifecycle management feature – move file after N days)**
    - **Standard - for frequently accessed files**
    - **Infrequent Access (EFS-IA) - cost to retrieve files, lower price to store. Enables EFS-IA with a lifecycle policy**
  + **Availability & durability,**
    - **Standard - Multi-AZ, great for Prod**
    - **One Zone - great for dev, backup enabled by default, compatible with IA (EFS One Zone-IA)**
  + **Over 90% In cost savings**

**Hands-on:**

**Create EFS & add it to two instances in diff AZ’s**

**===================================================================================================**

**Notes:**

* Billing info access should be managed from root account.