**AWS Global Infrastructure:**

**AWS Regions, AZ, Edge Locations**

-Region is a separate geographical area, completely independent, isolated from other regions.

- Communication b/w region is over public internet.

- Every region has multiple AZ

- AZ’s is physically isolated, geographically separated from each other and designed as an independent failure zone and are connected with low-latency links.

- Edge Locations are maintained by AWS through a worldwide network of data centers for distribution of content to reduce latency.

**AWS Wavelength**

* AWS infrastructure deployments embed AWS compute and storage services within the telecommunications providers’ datacenters and help seamlessly access the breadth of AWS services in the region.
* AWS Wavelength brings services to the edge of the 5G network, without leaving the mobile provider’s network reducing the extra network hops, minimizing the latency to connect to an application from a mobile device

**AWS Outposts**

* AWS Outposts bring native AWS services, infrastructure, and operating models to virtually any data center, co-location space, or on-premises facility.
* AWS Outposts is designed for connected environments and can be used to support workloads that need to remain on-premises due to low latency, compliance or local data processing needs.

**AWS Organizations**

* AWS Organizations offers policy-based management for multiple AWS accounts
* Organizations allows creation of groups of accounts and then apply policies to those groups
* Organizations enables you to centrally manage policies across multiple accounts, without requiring custom scripts and manual processes.
* Organizations helps simplify the billing for multiple accounts by enabling the setup of a single payment method for all the accounts in the organization through consolidated billing

**Consolidate Billing**

* Paying account with multiple linked accounts
* Paying account is independent and should be only used for billing purpose
* Paying account cannot access resources of other accounts unless given exclusively access through Cross Account roles
* All linked accounts are independent and soft limit of 20
* One bill per AWS account
* provides Volume pricing discount for usage across the accounts
* allows unused Reserved Instances to be applied across the group
* Free tier is not applicable across the accounts

**Tags & Resource Groups**

* are metadata, specified as key/value pairs with the AWS resources
* are for labelling purposes and helps managing, organizing resources
* can be inherited when created resources created from Auto Scaling, Cloud Formation, Elastic Beanstalk etc
* can be used for
  + Cost allocation to categorize and track the AWS costs
  + Conditional Access Control policy to define permission to allow or deny access on resources based on tags
* Resource Group is a collection of resources that share one or more tags

**AWS Services Region, AZ, Subnet VPC limitations**

* Services like IAM (user, role, group, SSL certificate), Route 53, STS are Global and available across regions
* All other AWS services are limited to Region or within Region and do not exclusively copy data across regions unless configured
* AMI are limited to region and need to be copied over to other region
* EBS volumes are limited to the Availability Zone, and can be migrated by creating snapshots and copying them to another region
* Reserved instances  (can be migrated to other Availability Zone now) cannot be migrated to another region
* RDS instances are limited to the region and can be recreated in a different region by either using snapshots or promoting a Read Replica
  + Cluster Placement groups are limited to single Availability Zones
  + Spread Placement groups can span across multiple Availability Zones
* S3 data is replicated within the region and can be move to another region using cross region replication
* DynamoDB maintains data within the region can be replicated to another region using DynamoDB cross region replication (using DynamoDB streams) or Data Pipeline using EMR (old method)
* Redshift Cluster span within an Availability Zone only, and can be created in other AZ using snapshots

**Disaster Recovery Whitepaper**

* **RTO** is the **time** it takes **after a disruption** to restore a business process to its service level and **RPO** acceptable **amount of data loss** measured in time **before the disaster occurs**
* Techniques (**RTO & RPO reduces and the Cost goes up** as we go down)
  + **Backup & Restore** – Data is backed up and restored, within nothing running
  + **Pilot light** – Only minimal critical service like RDS is running and rest of the services can be recreated and scaled during recovery
  + **Warm Standby** – Fully functional site with minimal configuration is available and can be scaled during recovery
  + **Multi-Site** – Fully functional site with identical configuration is available and processes the load

**Services**

* + Region and AZ to launch services across multiple facilities
  + EC2 instances with the ability to scale and launch across AZs
  + EBS with Snapshot to recreate volumes in different AZ or region
  + AMI to quickly launch preconfigured EC2 instances
  + ELB and Auto Scaling to scale and launch instances across AZs
  + VPC to create private, isolated section
  + Elastic IP address as static IP address
  + ENI with pre allocated Mac Address
  + Route 53 is highly available and scalable DNS service to distribute traffic across EC2 instances and ELB in different AZs and regions
  + Direct Connect for speed data transfer (takes time to setup and expensive then VPN)
  + S3 and Glacier (with RTO of 3-5 hours) provides durable storage
  + RDS snapshots and Multi AZ support and Read Replicas across regions
  + DynamoDB with cross region replication
  + Redshift snapshots to recreate the cluster
  + Storage Gateway to backup the data in AWS
  + Import/Export to move large amount of data to AWS (if internet speed is the bottleneck)
  + CloudFormation, Elastic Beanstalk and Opsworks as orchestration tools for automation and recreate the infrastructure

**Kinesis Data Streams**

* enables real-time processing of streaming data at massive scale
* provides ordering of records, as well as the ability to read and/or replay records in the same order to multiple Kinesis applications
* data is replicated across three data centers within a region and preserved for 24 hours, by default and can be extended to 7 days
* streams can be scaled using multiple shards, based on the partition key, with each shard providing the capacity of 1MB/sec data input and 2MB/sec data output with 1000 PUT requests per second
* Data encryption can be supported either using client side encryption before pushing the data to data streams or server side encryption.

**Kinesis vs SQS**

* + real-time processing of streaming big data vs reliable, highly scalable hosted queue for storing messages
  + ordered records, as well as the ability to read and/or replay records in the same order vs no guarantee on data ordering (with the standard queues before the FIFO queue feature was released)
  + data storage up to 24 hours, extended to 7 days vs up to 14 days, can be configured from 1 minute to 14 days but cleared if deleted by the consumer
  + supports multiple consumers vs single consumer at a time and requires multiple queues to deliver message to multiple consumers

**Producer & Consumers**

* + API PutRecord and PutRecords are **synchronous**, while KPL producer supports synchronous or **asynchronous** use cases
  + KCL uses a unique DynamoDB table to keep track of the application’s state, so if Kinesis Data Streams application receives provisioned-throughput exceptions, increase the provisioned throughput for the DynamoDB table

**Kinesis Firehose**

* is a fully managed service
* data transfer solution for delivering real time streaming data to destinations such as **S3,  Redshift,  Elasticsearch service, and Splunk**.
* is **NOT real time** (**min. 60 secs**) as it buffers incoming streaming data to a certain size or for a certain period of time before delivering it
* supports multiple producers as datasource, which include Kinesis data stream, Kinesis Agent, or the Kinesis Data Firehose API using the AWS SDK, CloudWatch Logs, CloudWatch Events, or AWS IoT
* supports out of box data transformation as well as custom transformation using Lambda function to transform incoming source data and deliver the transformed data to destinations
* supports interface VPC endpoint to keep traffic between the Amazon VPC and Kinesis Data Firehose from leaving the Amazon network.

**Kinesis Data Analytics**

* helps analyze streaming data, gain actionable insights, and respond to the business and customer needs in real time.
* reduces the complexity of building, managing, and integrating streaming applications with other AWS service

**Redshift**

* Redshift is a fast, fully managed data warehouse
* provides simple and cost-effective solution to analyze all the data using standard SQL and the existing Business Intelligence (BI) tools.
* manages the work needed to set up, operate, and scale a data warehouse, from provisioning the infrastructure capacity to automating ongoing administrative tasks such as backups, and patching.
* automatically monitors your nodes and drives to help you recover from failures.
* **only supports Single-AZ** deployments.
* replicates all the data within the data warehouse cluster when it is loaded and also continuously backs up your data to S3.
* attempts to maintain at least three copies of your data (the original and replica on the compute nodes and a backup in S3).
* supports cross-region snapshot replication to another region for disaster recovery
* Redshift supports four distribution styles; AUTO, EVEN, KEY, or ALL.
  + KEY distribution uses a single column as distribution key (DISTKEY) and helps place matching values on the same node slice
  + Even distribution distributes the rows across the slices in a **round-robin fashion**, regardless of the values in any particular column
  + ALL distribution replicates whole table in every compute node.
  + AUTO distribution lets Redshift assigns an optimal distribution style based on the size of the table data
* Redshift supports Compound and Interleaved sort keys
  + A compound key is made up of all of the columns listed in the sort key definition, in the order they are listed and is more efficient when **query predicates use a *prefix*, or query’s filter applies conditions, such as filters and joins, which is a subset of the sort key columns in order.**
  + An interleaved sort key gives **equal weight to each column in the sort key, so query predicates can use any subset of the columns that make up the sort key, in any order.**
* Column encodings **CANNOT** be changed once created.
* Redshift provides **query queues for Workload Management**, in order to manage concurrency and resource planning. It is a best practice to have separate queues for long running resource-intensive queries and fast queries that don’t require big amounts of memory and CPU

**EMR**

* is a web service that utilizes a hosted **Hadoop** framework running on the web-scale infrastructure of EC2 and S3
* launches all nodes for a given cluster in the **same Availability Zone**, which improves performance as it provides higher data access rate
* seamlessly supports Reserved, On-Demand and Spot Instances
* consists of Master Node for management and Slave nodes, which consists of Core nodes holding data and Task nodes for performing tasks only
* is fault tolerant for slave node failures and continues job execution if a slave node goes down
* does not automatically provision another node to take over failed slaves
* supports Persistent and Transient cluster types
  + Persistent which continue to run
  + Transient which terminates once the job steps are completed
* supports **EMRFS** which allows S3 to be used as a durable HA data storage

**Glue**

* fully-managed ETL service that automates the time-consuming steps of data preparation for analytics
* recommends and generates ETL code to transform the source data into target schemas, and runs the ETL jobs on a fully managed, scale-out Apache Spark environment to load your data into its destination.
* helps setup, orchestrate, and monitor complex data flows.
* **AWS Glue Data Catalog**
  + is a central repository to store structural and operational metadata for all the data assets.
  + automatically discovers and profiles the data
  + automatically discover both structured and semi-structured data stored in the data lake on S3, data warehouse in Redshift, and other databases
  + provides a unified view of the data that is available for ETL, querying and reporting using services like Athena, EMR, and Redshift Spectrum.
* **AWS Glue crawler**
  + connects to a data store, progresses through a prioritized list of classifiers to extract the schema of the data and other statistics, and then populates the Glue Data Catalog with this metadata

**Data Pipeline**

* orchestration service that helps define **data-driven workflows** to automate and schedule regular data movement and data processing activities
* integrates with **on-premises and cloud-based** storage systems
* allows scheduling, **retry, and failure logic** for the workflows

**SQS**

* extremely scalable queue service and potentially handles millions of messages
* helps build fault tolerant, distributed loosely coupled applications
* **stores copies of the messages on multiple servers** for redundancy and high availability
* guarantees **At-Least-Once Delivery**, but does not guarantee Exact One Time Delivery which might result in **duplicate** messages (Not true anymore with the introduction of FIFO queues)
* **does not maintain or guarantee message order**, and if needed sequencing information needs to be added to the message itself (Not true anymore with the introduction of FIFO queues)
* **supports multiple readers and writers** interacting with the same queue as the same time
* holds message for 4 days, by default, and can be changed from 1 min – 14 days after which the message is deleted
* message needs to be **explicitly deleted** by the consumer once processed
* allows send, receive and delete **batching** which helps club up to 10 messages in a single batch while charging price for a single message
* handles visibility of the message to multiple consumers using **Visibility Timeout**, where the message once read by a consumer is not visible to the other consumers till the timeout occurs
* can handle load and performance requirements by scaling the worker instances as the demand changes (**Job Observer pattern**)
* message sample allowing **short and long polling**
  + returns immediately **vs** waits for fixed time for e.g. 20 secs
  + might not return all messages as it samples a subset of servers **vs** returns all available messages
  + repetitive **vs** helps save cost with long connection
* supports **delay queues** to make messages available after a certain delay, can you used to differentiate from priority queues
* supports **dead letter queues**, to redirect messages which failed to process after certain attempts instead of being processed repeatedly
* **Design Patterns**
  + **Job Observer Pattern**can help coordinate number of EC2 instances with number of job requests (Queue Size) automatically thus Improving cost effectiveness and performance
  + **Priority Queue Pattern** can be used to setup different queues with different handling either by delayed queues or low scaling capacity for handling messages in lower priority queues

**SNS**

* delivery or sending of messages to subscribing endpoints or clients
* **publisher-subscriber** model
* Producers and Consumers communicate **asynchronously** with subscribers by producing and sending a message to a topic
* supports **Email (plain or JSON), HTTP/HTTPS, SMS, SQS**
* supports **Mobile Push Notifications** to push notifications directly to mobile devices with services like Amazon Device Messaging (ADM), Apple Push Notification Service (APNS), Google Cloud Messaging (GCM) etc. supported
* **order is not guaranteed** and **No recall** available
* **integrated with Lambda** to invoke functions on notifications
* **for Email notifications, use SNS or SES directly, SQS does not work**

**SWF**

* **orchestration service** to coordinate work across distributed components
* helps define tasks, stores, assigns tasks to workers, define logic, tracks and monitors the task and maintains workflow state in a durable fashion
* helps define tasks which can be executed on AWS cloud or **on-premises**
* helps coordinating tasks across the application which involves managing intertask dependencies, scheduling, and concurrency in accordance with the logical flow of the application
* supports **built-in retries, timeouts and logging**
* supports **manual tasks**
* Characteristics
  + deliver exactly once
  + uses long polling, which reduces number of polls without results
  + Visibility of task state via API
  + Timers, signals, markers, child workflows
  + supports versioning
  + keeps workflow history for a user-specified time

**AWS SWF vs AWS SQS**

* + task-oriented **vs** message-oriented
  + track of all tasks and events **vs** needs custom handling

**SES**

* highly scalable and cost-effective email service
* uses **content filtering technologies** to scan outgoing emails to check standards and email content for spam and malware
* **supports full fledged emails to be sent as compared to SNS where only the message is sent in Email**
* ideal for **sending bulk emails** at scale
* **guarantees first hop**
* eliminates the need to support custom software or applications to do heavy lifting of email transport

**RDS**

* provides Relational Database service
* supports MySQL, MariaDB, PostgreSQL, Oracle, Microsoft SQL Server, and the new, MySQL-compatible Amazon Aurora DB engine
* as it is a managed service, shell (root ssh) access is not provided
* manages backups, software patching, automatic failure detection, and recovery
* supports use initiated manual backups and snapshots
* **daily automated backups with database transaction logs** enables **Point in Time recovery** up to the last five minutes of database usage
* **snapshots** are user-initiated storage volume snapshot of DB instance, backing up the **entire DB instance and not just individual databases** that can be restored as a independent RDS instance
* support encryption at rest using KMS as well as encryption in transit using SSL endpoints
* for encrypted database
  + logs, snapshots, backups, read replicas are all encrypted as well
  + cross region replicas and snapshots does not work across region (Note – this is possible now with latest AWS [enhancement](https://aws.amazon.com/about-aws/whats-new/2017/01/amazon-rds-now-supports-read-replicas-of-encrypted-database-instances-across-regions/))
* Multi-AZ deployment
  + provides **high availability and automatic failover support and is NOT a scaling solution**
  + maintains a **synchronous standby replica in a different AZ**
  + **transaction success is returned only if the commit is successful both on the primary and the standby DB**
  + Oracle, PostgreSQL, MySQL, and MariaDB DB instances use **Amazon technology**, while SQL Server DB instances use SQL **Server Mirroring**
  + **snapshots and backups are taken from standby & eliminate I/O freezes**
  + during automatic failover, its seamless and **RDS switches to the standby instance** and **updates the DNS record to point to standby**
  + failover can be **forced** with the Reboot with failover option
* Read Replicas
  + uses the PostgreSQL, MySQL, and MariaDB DB engines’ built-in replication functionality to create a separate Read Only instance
  + updates are **asynchronously** copied to the Read Replica, and data might be stale
  + can help **scale applications** and **reduce read only load**
  + **requires automatic backups enabled**
  + **replicates all databases** in the source DB instance
  + for disaster recovery, can be **promoted to a full fledged database**
  + can be **created in a different region** for MySQL, Postgres and MariaDB, for disaster recovery, migration and low latency across regions
* RDS does not support all the features of underlying databases, and if required the database instance can be launched on an EC2 instance
* RMAN (Recovery Manager) can be used for Oracles backup and recovery when running on an EC2 instance

**DynamoDB**

* fully managed NoSQL database service
* synchronously **replicates data across three facilities** in an AWS Region, giving high availability and data durability
* runs exclusively on **SSDs** to provide high I/O performance
* provides **provisioned table reads and writes**
* **automatically partitions, reallocates and re-partitions the data** and provisions additional server capacity as data or throughput changes
* provides **Eventually consistent (by default) or Strongly Consistent** option to be specified during an read operation
* creates and maintains **indexes for the primary key attributes** for efficient access of data in the table
* supports secondary indexes
  + allows querying attributes other then the primary key attributes without impacting performance.
  + are automatically maintained as **sparse objects**
* Local vs Global secondary index
  + shares partition key + different sort key vs different partition + sort key
  + search limited to partition vs across all partition
  + unique attributes vs non unique attributes
  + linked to the base table vs independent separate index
  + only created during the base table creation vs can be created later
  + cannot be deleted after creation vs can be deleted
  + consumes provisioned throughput capacity of the base table vs independent throughput
  + returns all attributes for item vs only projected attributes
  + Eventually or Strongly vs Only Eventually consistent reads
  + size limited to 10Gb per partition vs unlimited
* supports **cross region replication** using DynamoDB streams which leverages Kinesis and provides **time-ordered sequence of item-level changes** and can help for lower RPO, lower RTO disaster recovery
* Data Pipeline jobs with EMR can be used for disaster recovery with higher RPO, lower RTO requirements
* supports**triggers**to allow execution of custom actions or notifications based on item-level updates

**ElastiCache**

* managed web service that provides **in-memory caching** to deploy and run Memcached or Redis protocol-compliant cache clusters
* ElastiCache with Redis,
  + like RDS, supports **Multi-AZ, Read Replicas and Snapshots**
  + Read Replicas are created across AZ within same region using **Redis’s asynchronous replication technology**
  + Multi-AZ differs from RDS as there is no standby, but **if the primary goes down a Read Replica is promoted as primary**
  + **Read Replicas cannot span across regions**, as RDS supports
  + **cannot be scaled out** and **if scaled up cannot be scaled down**
  + **allows snapshots for backup and restore**
  + **AOF** can be enabled for **recovery scenarios**, to recover the data in case the node fails or service crashes. But it does not help in case the underlying hardware fails
  + **Enabling Redis Multi-AZ as a Better Approach to Fault Tolerance**
* ElastiCache with Memcached
  + **can be scaled up by increasing size and scaled out by adding nodes**
  + nodes can **span across multiple AZs** within the same region
  + **cached data is spread across the nodes**, and a node failure will always result in some data loss from the cluster
  + **supports auto discovery**
  + **every node should be homogenous** and of same instance type
* ElastiCache Redis vs Memcached
  + complex data objects vs simple key value storage
  + persistent vs non persistent, pure caching
  + automatic failover with Multi-AZ vs Multi-AZ not supported
  + scaling using Read Replicas vs using multiple nodes
  + backup & restore supported vs not supported
* can be used state management to keep the web application stateless

**Redshift**

* fully managed, fast and powerful, petabyte scale data warehouse service
* uses replication and continuous backups to enhance availability and improve data durability and can automatically recover from node and component failures
* provides Massive Parallel Processing (MPP) by distributing & parallelizing queries across multiple physical resources
* columnar data storage improving query performance and allowing advance compression techniques
* **only supports Single-AZ deployments** and the nodes are available within the same AZ, if the AZ supports Redshift clusters
* spot instances are **NOT** an option

**VPC**

* helps define a logically isolated dedicated virtual network within the AWS
* provides control of IP addressing using CIDR block from a minimum of /28 to maximum of /16 block size
* supports IPv4 and IPv6 addressing
* ~~cannot be extended once created~~
* can be extended by associating secondary IPv4 CIDR blocks to VPC
* **Components**
  + Internet gateway (**IGW**) provides access to the Internet
  + Virtual gateway (**VGW**) provides access to on-premises data center through **VPN** and **Direct Connect** connections
  + VPC can have only one IGW and VGW
  + **Route tables** determine where network traffic from subnet is directed
  + Ability to create **subnet** with VPC CIDR block
  + A Network Address Translation (**NAT**) server provides outbound Internet access for EC2 instances in private subnets
  + **Elastic IP addresses** are static, persistent public IP addresses
  + Instances launched in the VPC will have a **Private IP address** and can have a **Public or a Elastic IP address** associated with it
  + **Security Groups and NACLs** help define security
  + **Flow logs** – Capture information about the IP traffic going to and from network interfaces in your VPC
* **Tenancy option** for instances
  + **shared**, by default, allows instances to be launched on shared tenancy
  + **dedicated** allows instances to be launched on a dedicated hardware
* **Route Tables**
  + defines rules, termed as routes, which determine where network traffic from the subnet would be routed
  + Each VPC has a Main Route table, and can have multiple custom route tables created
  + Every route table contains a local route that enables communication within a VPC which cannot be modified or deleted
  + Route priority is decided by matching the most specific route in the route table that matches the traffic
* **Subnets**
  + **map to AZs** and do not span across AZs
  + have a CIDR range that is a portion of the whole VPC.
  + **CIDR ranges cannot overlap** between subnets within the VPC.
  + AWS **reserves 5 IP addresses in each subnet – first 4 and last one**
  + Each subnet is associated with a route table which define its behavior
    - **Public subnets** – inbound/outbound Internet connectivity via IGW
    - **Private subnets** – outbound Internet connectivity via an NAT or VGW
    - **Protected subnets** – no outbound connectivity and used for regulated workloads
* **Elastic Network Interface (ENI)**
  + a default ENI, eth0, is attached to an instance which cannot be detached with one or more secondary detachable ENIs (eth1-ethn)
  + has primary private, one or more secondary private, public, Elastic IP address, security groups, MAC address and source/destination check flag attributes associated
  + AN ENI in one subnet can be attached to an instance in the same or another subnet, in the same AZ and the same VPC
  + Security group membership of an ENI can be changed
  + with pre allocated Mac Address can be used for applications with special licensing requirements
* **Security Groups vs Network Access Control Lists**
  + Stateful **vs** Stateless
  + At instance level **vs** At subnet level
  + Only allows Allow rule **vs** Allows both Allow and Deny rules
  + Evaluated as a Whole **vs** Evaluated in defined Order
* **Elastic IP**
  + is a **static IP address** designed for dynamic cloud computing.
  + is **associated with AWS account**, and not a particular instance
  + can be **remapped** from one instance to an other instance
  + is **charged for non usage**, if not linked for any instance or instance associated is in stopped state
* **NAT**
  + allows internet access to instances in private subnet
  + performs the function of both address translation and port address translation (PAT)
  + needs source/destination check flag to be disabled as it is not actual destination of  the traffic
  + NAT gateway is a AWS managed NAT service that provides better availability, higher bandwidth, and requires less administrative effort
  + are not supported for IPv6 traffic
* **Egress-Only Internet Gateways**
  + outbound communication over IPv6 from instances in the VPC to the Internet, and prevents the Internet from initiating an IPv6 connection with your instances
  + supports IPv6 traffic only
* **Shared VPCs**
  + allows multiple AWS accounts to create their application resources, such as EC2 instances, RDS databases, Redshift clusters, and AWS Lambda functions, into shared, centrally-managed VPCs
* **VPC Peering**
  + allows routing of traffic between the peer VPCs **using private IP addresses** and no IGW or VGW required
  + No single point of failure and bandwidth bottlenecks
  + supports inter-region VPC peering
  + IP space or **CIDR blocks cannot overlap**
  + **cannot be transitive**, one-to-one relationship between two VPC
  + Only one between any two VPCs and have to be explicitly peered
  + **Private DNS values cannot be resolved**
  + Security groups from peered VPC can now be referred, however the VPC should be in the same region.
* **VPC Endpoints**
  + enables you to privately connect VPC to supported AWS services and VPC endpoint services powered by PrivateLink
  + **does not require a public IP address, access over the Internet, NAT device, a VPN connection or Direct Connect**
  + traffic between VPC & AWS service does not leave the Amazon network
  + are virtual devices.
  + are horizontally scaled, redundant, and highly available VPC components that allow communication between instances in your VPC and services without imposing availability risks or bandwidth constraints on your network traffic.
  + Gateway Endpoints
    - is a gateway that is a target for a specified route in the route table, used for traffic destined to a supported AWS service.
    - only S3 and DynamoDB are currently supported
  + Interface Endpoints
    - is an elastic network interface with a private IP address that serves as an entry point for traffic destined to a supported service
    - services supported API Gateway AWS CloudFormation, CloudWatch, CloudWatch Events, CloudWatch Logs AWS CodeBuild AWS CodeCommit AWS Config, EC2 API Elastic Load Balancing API, Elastic Container Registry, Elastic Container Service AWS Key Management Service, Kinesis Data Streams, SageMaker and, SageMaker Runtime, SageMaker Notebook Instance AWS Secrets Manager AWS Security Token Service AWS Service Catalog, SNS, SQS AWS Systems Manager

**Direct Connect & VPN**

* **VPN**
  + provide secure IPSec connections from on-premise computers or services to AWS over the Internet
  + is quick to setup, is cheap however it depends on the Internet speed
* **Direct Connect**
  + is a network service that provides an alternative to using Internet to utilize AWS services by using private dedicated network connection
  + provides Virtual Interfaces
    - **Private VIF** to access instances within an VPC via VGW
    - **Public VIF** to access non VPC services
  + **requires time to setup** probably months, and should not be considered as an option if turnaround time is less
  + **does not provide redundancy**, use either second direct connection or IPSec VPN connection
  + Virtual Private Gateway is on the AWS side and Customer Gateway is on the Customer side
  + **route propagation is enabled on VGW** and not on CGW
* **Direct Connect vs VPN IPSec**
  + Expensive to Setup and Takes time vs Cheap & Immediate
  + Dedicated private connections vs Internet
  + Reduced data transfer rate vs Internet data transfer cost
  + Consistent performance vs Internet inherent variability
  + Do not provide Redundancy vs Provides Redundancy

**Route 53**

* Highly available and scalable DNS & Domain Registration Service
* Reliable and cost-effective way to route end users to Internet applications
* Supports **multi-region and backup architectures for High availability. ELB , limited to region, does not support multi region HA architecture**
* supports private Intranet facing DNS service
* **internal resource record sets only work for requests originating from within the VPC** and currently cannot extend to on-premise
* Global propagation of any changes made to the DN records within ~ 1min
* Route 53 to create an [**alias resource record set**](http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resource-record-sets-choosing-alias-non-alias.html) that points to ELB, S3, CloudFront. An alias resource record set is a Route 53 extension to DNS. It’s similar to a CNAME resource record set, but supports both for root domain – zone apex  *e.g. example.com*, and for subdomains for e.g. *www.example.com*.
* CNAME resource record sets can be created only for subdomains and cannot be mapped to the zone apex record
* Route 53 **Split-view (Split-horizon)** DNS enables you to access an internal version of your website using the same domain name that is used publicly
* **Routing policy**
  + Simple routing – simple round robin policy
  + Weighted round robin – assign weights to resource records sets to specify the proportion *for e.g. 80%:20%*
  + Latency based routing – helps improve global applications as request are sent to server from the location with minimal latency, is based on the latency and cannot guarantee users from the same geographic will be served from the same location for any compliance reasons
  + Geolocation routing – Specify geographic locations by continent, country, state limited to US, is based on IP accuracy
  + Failover routing – failover to a backup site if the primary site fails and becomes unreachable
* Weighted, Latency and Geolocation can be used for Active-Active while Failover routing can be used for Active-Passive multi region architecture

**CloudFormation**

* gives developers and systems administrators an easy way to create and manage a collection of related AWS resources
* Resources can be updated, deleted and modified in a **orderly, controlled and predictable fashion**, in effect applying **version control** to the AWS **infrastructure as code** done for software code
* CloudFormation **Template is an architectural diagram**, in JSON format, and **Stack is the end result of that diagram**, which is actually provisioned
* template can be used to set up the resources consistently and repeatedly over and over across multiple regions and consists of
  + List of AWS **resources** and their configuration values
  + An optional **template file format version number**
  + An optional list of **template parameters** (input values supplied at stack creation time)
  + An optional list of **output values** like public IP address using the Fn::GetAtt function
  + An optional list of **data tables** used to lookup static configuration values *for e.g., AMI names per AZ*
* **supports Chef & Puppet Integration** to deploy and configure right down the the application layer
* supports **Bootstrap scripts** to install packages, files and services on the EC2 instances by simple describing them in the CF template
* **automatic rollback on error** feature is enabled, by default, which will cause all the AWS resources that CF created successfully for a stack up to the point where an error occurred to be deleted
* provides a ***WaitCondition*** resource to block the creation of other resources until a completion signal is received from an external source
* allows **DeletionPolicy** attribute to be defined for resources in the template
  + **retain** to preserve resources like S3 even after stack deletion
  + **snapshot** to backup resources like RDS after stack deletion
* **DependsOn** attribute to specify that the creation of a specific resource follows another
* **Service role** is an IAM role that allows AWS CloudFormation to make calls to resources in a stack on the user’s behalf
* support **Nested stacks** that can separate out reusable, common components and create dedicated templates to mix and match different templates but use nested stacks to create a single, unified stack

**Elastic BeanStalk**

* makes it easier for developers to quickly deploy and manage applications in the AWS cloud.
* automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling and application health monitoring
* **CloudFormation supports** ElasticBeanstalk
* provisions resources to support
  + a web application that handles HTTP(S) requests or
  + a web application that handles background-processing (worker) tasks
* supports Out Of the Box
  + Apache Tomcat for Java applications
  + Apache HTTP Server for PHP applications
  + Apache HTTP server for Python applications
  + Nginx or Apache HTTP Server for Node.js applications
  + Passenger for Ruby applications
  + MicroSoft IIS 7.5 for .Net applications
  + Single and Multi Container Docker
* supports custom AMI to be used
* is designed to **support multiple running environments** such as one for Dev, QA, Pre-Prod and Production.
* **supports versioning** and stores and tracks application versions over time allowing easy rollback to prior version
* can provision RDS DB instance and connectivity information is exposed to the application by environment variables, but is NOT recommended for production setup as the RDS **is tied up with the Elastic Beanstalk lifecycle** and if deleted, the RDS instance would be deleted as well

**OpsWorks**

* is a **configuration management service** that helps to configure and operate applications in a cloud enterprise by using **Chef**
* helps **deploy and monitor applications in stacks with multiple layers**
* supports preconfigured layers for Applications, Databases, Load Balancers, Caching
* OpsWorks Stacks features is a set of lifecycle events – Setup, Configure, Deploy, Undeploy, and Shutdown – which automatically runs specified set of recipes at the appropriate time on each instance
* Layers depend on **Chef recipes** to handle tasks such as installing packages on instances, deploying apps, running scripts, and so on
* OpsWorks Stacks **runs the recipes for each layer**, even if the instance belongs to multiple layers
* supports **Auto Healing** and Auto Scaling to monitor instance health, and provision new instances

**CloudWatch**

* allows monitoring of AWS resources and applications in real time, collect and track pre configured or custom metrics and configure alarms to send notification or make resource changes based on defined rules
* **does not aggregate data across regions**
* **stores the log data indefinitely**, and the retention can be changed for each log group at any time
* **alarm history is stored for only 14 days**
* can be used an **alternative to S3 to store logs** with the ability to configure Alarms and generate metrics, however logs **cannot be made public**
* Alarms exist only in the created region and the Alarm actions must reside in the same region as well

**CloudTrail**

* records access to API calls for the AWS account made from AWS management console, SDKs, CLI and higher-level AWS service
* support many AWS services and tracks who did, from where, what & when
* can be **enabled per-region basis**, a region can include global services (like IAM, STS etc), is applicable to all the **supported services within that region**
* log files from different regions can be sent to the **same S3 bucket**
* can be integrated with SNS to notify logs availability, CloudWatch logs log group for notifications when specific API events occur
* call history enables **security analysis, resource change tracking, trouble shooting and compliance auditing**

**IAM**

* securely control access to AWS services and resources
* helps create and manage user identities and grant permissions for those users to access AWS resources
* helps create groups for multiple users with similar permissions
* not appropriate for application authentication
* is Global and does not need to be migrated to a different region
* helps define Policies,
  + in JSON format
  + all permissions are implicitly denied by default
  + most restrictive policy wins
* **IAM Role**
  + helps grants and delegate access to users and services without the need of creating permanent credentials
  + IAM users or AWS services can assume a role to obtain temporary security credentials that can be used to make AWS API calls
  + needs Trust policy to define who and Permission policy to define what the user or service can access
  + used with Security Token Service (STS), a lightweight web service that provides temporary, limited privilege credentials for IAM users or for authenticated federated users
  + IAM role scenarios
    - Service access *for e.g. EC2 to access S3 or DynamoDB*
    - Cross Account access for users
      * with user within the same account
      * with user within an AWS account owned the same owner
      * with user from a Third Party AWS account with External ID for enhanced security
    - Identity Providers & Federation
      * Web Identity Federation, where the user can be authenticated using external authentication Identity providers like Amazon, Google or any OpenId IdP using AssumeRoleWithWebIdentity
      * Identity Provider using SAML 2.0, where the user can be authenticated using on premises Active Directory, Open Ldap or any SAML 2.0 compliant IdP using AssumeRoleWithSAML
      * For other Identity Providers, use Identity Broker to authenticate and provide temporary Credentials using AssumeRole (recommended) or GetFederationToken

**CloudHSM**

* provides **secure cryptographic key storage** to customers by making hardware security modules (HSMs) available in the AWS cloud
* **single tenant, dedicated physical device** to securely generate, store, and manage cryptographic keys used for data encryption
* are**inside the VPC** (not EC2-classic) & isolated from the rest of the network
* can use VPC peering to connect to CloudHSM from multiple VPCs
* integrated with Amazon Redshift and Amazon RDS for Oracle
* EBS volume encryption, S3 object encryption and key management can be done with CloudHSM but requires custom application scripting
* is **NOT fault tolerant** and would need to build a cluster as if one fails all the keys are lost
* **expensive,**prefer AWS Key Management Service (KMS) if cost is a criteria

**AWS Directory Services**

* gives applications in AWS access to Active Directory services
* different from SAML + AD, where the access is granted to AWS services through Temporary Credentials
* Simple AD
  + least expensive but does not support Microsoft AD advance features
  + provides a Samba 4 Microsoft Active Directory compatible standalone directory service on AWS
  + No single point of Authentication or Authorization, as a separate copy is maintained
  + trust relationships cannot be setup between Simple AD and other Active Directory domains
  + Don’t use it, if the requirement is to leverage access and control through centralized authentication service
* AD Connector
  + acts just as an hosted proxy service for instances in AWS to connect to on-premises Active Directory
  + enables consistent enforcement of existing security policies, such as password expiration, password history, and account lockouts, whether users are accessing resources on-premises or in the AWS cloud
  + needs VPN connectivity (or Direct Connect)
  + integrates with existing RADIUS-based MFA solutions to enabled multi-factor authentication
  + does not cache data which might lead to latency
* Read-only Domain Controllers (RODCs)
  + works out as a Read-only Active Directory
  + holds a copy of the Active Directory Domain Service (AD DS) database and respond to authentication requests
  + they cannot be written to and are typically deployed in locations where physical security cannot be guaranteed
  + helps maintain a single point to authentication & authorization controls, however needs to be synced
* Writable Domain Controllers
  + are expensive to setup
  + operate in a multi-master model; changes can be made on any writable server in the forest, and those changes are replicated to servers throughout the entire forest

**AWS WAF**

* is a web application firewall that helps monitor the HTTP/HTTPS traffic and allows controlling access to the content.
* helps protect web applications from attacks by allowing rules configuration that allow, block, or monitor (count) web requests based on defined conditions. These conditions include IP addresses, HTTP headers, HTTP body, URI strings, SQL injection and cross-site scripting.
* helps define Web ACLs, which is a combination of Rules that is a combinations of Conditions and Action to block or allow
* integrated with CloudFront, Application Load Balancer (ALB), API Gateway services commonly used to deliver content and applications
* supports custom origins outside of AWS, when integrated with CloudFront

**AWS Shield**

* is a managed service that provides protection against Distributed Denial of Service (DDoS) attacks for applications running on AWS
* provides protection for all AWS customers against common and most frequently occurring **infrastructure (layer 3 and 4) attacks** like SYN/UDP floods, reflection attacks, and others to support high availability of applications on AWS.
* provides AWS Shield Advanced with additional protections against more sophisticated and larger attacks for applications running on EC2, ELB, CloudFront, AWS Global Accelerator, and Route 53

**AWS GuardDuty**

* offers threat detection that enables continuous monitoring and protect the AWS accounts and workloads.
* analyzes continuous streams of meta-data generated from AWS account and network activity found in AWS CloudTrail Events, VPC Flow Logs, and DNS Logs.
* integrated threat intelligence such as known malicious IP addresses, anomaly detection, and machine learning to identify threats more accurately.
* operates completely independently from the resources so there is no risk of performance or availability impacts to the workloads

**AWS Inspector**

* is an automated security assessment service that helps test the network accessibility of EC2 instances and the security state of the applications running on the instances.
* helps automate security vulnerability assessments throughout the development and deployment pipeline or against static production systems

**AWS Artifact**

* is a self-service audit artifact retrieval portal that provides customers with on-demand access to AWS’ compliance documentation and agreements
* can use AWS Artifact Reports to download AWS security and compliance documents, *such as AWS ISO certifications, Payment Card Industry (PCI), and System and Organization Control (SOC) reports.*

## EC2

* provides **scalable computing capacity**
* **Features**
  + Virtual computing environments, known as ***EC2 instances***
  + Preconfigured templates for EC2 instances, known as ***Amazon Machine Images (AMIs)***, that package the bits needed for the server (including the operating system and additional software)
  + Various configurations of CPU, memory, storage, and networking capacity for your instances, known as ***Instance types***
  + Secure login information for your instances using ***key pairs*** (public-private keys where private is kept by user)
  + Storage volumes for temporary data that’s deleted when you stop or terminate your instance, known as ***Instance store volumes***
  + Persistent storage volumes for data using **Elastic Block Store (EBS)**
  + Multiple physical locations for your resources, such as instances and EBS volumes, known as ***Regions and Availability Zones***
  + A firewall to specify the protocols, ports, and source IP ranges that can reach your instances using ***Security Groups***
  + Static IP addresses, known as ***Elastic IP addresses***
  + Metadata, known as ***tags***, can be created and assigned to EC2 resources
  + Virtual networks that are logically isolated from the rest of the AWS cloud, and can optionally connect to on premises network, known as **Virtual private clouds (VPCs)**

### [**Amazon Machine Image – AMI**](https://jayendrapatil.com/aws-ec2-ami/)

* + **template** from which EC2 instances can be launched quickly
  + **does NOT span across across regions**, and needs to be copied
  + **can be shared with other specific AWS accounts or made public**

### [**Instance Types**](https://jayendrapatil.com/aws-ec2-instance-types/)

* + **T** for applications needing **general** usage
  + **R** for applications needing more **RAM or Memory**
  + **C** for applications needing more **Compute**
  + **M** for applications needing more **Medium or Moderate** performance on both Memory and CPU
  + **I** for applications needing more **IOPS**
  + **G**for applications needing more **GPU**

### [**Purchasing Option**](https://jayendrapatil.com/aws-ec2-instance-purchasing-option/)

* + **On-Demand Instances**
    - pay for instances and compute capacity that you use by the hour
    - **no long-term commitments** or **up-front payments**
  + **Reserved Instances**
    - provides **lower hourly running costs** by providing a billing discount
    - **capacity reservation** is applied to instances
    - suited if **consistent, heavy, predictable usage**
    - **provides benefits with Consolidate Billing**
    - can be modified to **switch Availability Zones** or the **instance size within the same instance type**, given the instance size footprint (**Normalization factor**) remains the same
    - **pay for the entire term** regardless of the usage, so if the question targets cost effective solution and answer mentions reserved instances are purchased & unused, it can be ignored
    - is **not a physical instance** that is launched, but rather a **billing discount** applied to the use of On-Demand Instances
  + **Scheduled Reserved Instances**
    - enable capacity reservations purchase that **recur on a daily, weekly, or monthly basis**, with a specified start time and duration, for a one-year term.
    - Charges are incurred for the time that the instances are scheduled, even if they are not used
    - good choice for workloads that **do not run continuously, but do run on a regular schedule**
  + **Spot Instances**
    - **cost-effective choice** but **does NOT guarantee availability**
    - **applications flexible in the timing**when they can run and also**able to handle interruption**by storing the state externally
    - provides a **two minute warning** if the instance is to be terminated to save any unsaved work
    - **Spot blocks**can also be launched with a required duration, which are **not interrupted** due to changes in the Spot price
    - **Spot Fleet** is a collection, or fleet, of Spot Instances, and optionally On-Demand Instances, which attempts to launch the number of Spot and On-Demand Instances to meet the specified target capacity
  + **Dedicated Instances**
    - is a tenancy option which enables instances to run in VPC on hardware that’s isolated, dedicated to a single customer
  + **Dedicated Host**
    - is a **physical server with EC2 instance capacity fully dedicated** to your use
  + **Light, Medium, and Heavy Utilization Reserved Instances are no longer available** for purchase and were part of the Previous Generation AWS EC2 purchasing model

### **Enhanced Networking**

* + results in **higher bandwidth, higher packet per second (PPS) performance, lower latency, consistency, scalability and lower jitter**
  + supported using **Single Root I/O Virtualization (SR-IOV)** only on supported instance types
  + is **supported only with an VPC (not EC2 Classic), HVM virtualization type**and available by default on Amazon AMI but can be installed on other AMIs as well

### [**Placement Group**](https://jayendrapatil.com/aws-ec2-placement-groups/)

* + **Cluster Placement Group**
    - provide **low latency, High Performance Computing** via 10Gbps network
    - is a logical grouping on instances within a Single AZ
    - **don’t span availability zones**, can span multiple subnets but subnets must be in the same AZ
    - **can span across peered VPCs** for the same Availability Zones
    - **Existing instance can be moved to a placement group, or moved from one placement group to another, or removed from a placement group, given it is in the stopped state.**
    - for **capacity errors, stop and start the instances** in the placement group
    - use **homogenous instance types** which support enhanced networking and **launch all the instances at once**
  + **Spread Placement Groups**
    - is a group of instances that are each placed on distinct underlying hardware i.e. **each instance on distinct rack across AZ**
    - recommended for applications that have a small number of critical instances that should be kept separate from each other.
    - reduces the risk of simultaneous failures that might occur when instances share the same underlying hardware.
  + **Partition Placement Groups**
    - is a **group of instances spread across partitions**i.e. group of instances spread across racks across AZs
    - reduces the likelihood of correlated hardware failures for the application.
    - can be used to spread deployment of large distributed and replicated workloads, such as HDFS, HBase, and Cassandra, across distinct hardware

## Elastic Load Balancer & Auto Scaling

### [**Elastic Load Balancer**](https://jayendrapatil.com/aws-elastic-load-balancing/)

* + Managed load balancing service and scales automatically
  + distributes incoming application traffic across multiple EC2 instances
  + **is distributed system that is fault tolerant and actively monitored by AWS scales it as per the demand**
  + are engineered to **not be a single point of failure**
  + need to **Pre Warm** ELB if the demand is expected to shoot especially during load testing. AWS documentation does not mentioned it now
  + supports routing traffic to instances in **multiple AZs in the same region**
  + performs **Health Checks** to route traffic only to the healthy instances
  + support Listeners with HTTP, HTTPS, SSL, TCP protocols
  + has an associated IPv4 and dual stack DNS name
  + can offload the work of encryption and decryption (**SSL termination**) so that the EC2 instances can focus on their main work
  + supports **Cross Zone load balancing** to help route traffic **evenly across all EC2 instances regardless of the AZs** they reside in
  + to help identify the IP address of a client
    - supports **Proxy Protocol header** for TCP/SSL connections
    - supports **X-Forward headers** for HTTP/HTTPS connections
  + supports **Stick Sessions** (session affinity) to bind a user’s session to a specific application instance,
    - it is not fault tolerant, if an instance is lost the information is lost
    - requires HTTP/HTTPS listener and does not work with TCP
    - requires SSL termination on ELB as it users the headers
  + supports **Connection draining** to help complete the in-flight requests in case an instance is deregistered
  + For High Availability, it is recommended to attach one subnet per AZ for at least two AZs, even if the instances are in a single subnet.
  + **cannot assign an Elastic IP** address to an ELB
  + IPv4 & IPv6 support  VPC now supports IPV6.
  + **HTTPS listener does not support Client Side Certificate**
  + For **SSL termination at backend instances or support for Client Side Certificate** use TCP for connections from the client to the ELB, use the SSL protocol for connections from the ELB to the back-end application, and deploy certificates on the back-end instances handling requests
  + ELB supports multiple SSL certificates

### [**Application Load Balancer**](https://jayendrapatil.com/aws-elb-application-load-balancer/)

* supports **HTTP and HTTPS (Secure HTTP) protocols**
* supports **HTTP/2**, which is enabled natively. Clients that support HTTP/2 can connect over TLS
* supports **WebSockets** and Secure WebSockets natively
* supports **Request tracing**, by default.
  + request tracing can be used to track HTTP requests from clients to targets or other services.
  + Load balancer upon receiving a request from a client, adds or updates the **X-Amzn-Trace-Id** header before sending the request to the target
* supports **Sticky Sessions (Session Affinity)** using load balancer generated cookies, to route requests from the same client to the same target
* supports **SSL termination**, to decrypt the request on ALB before sending it to the underlying targets.
* supports layer 7 specific features like **X-Forwarded-For** headers to help determine the actual client IP, port and protocol
* automatically **scales** its request handling capacity in response to incoming application traffic.
* supports **hybrid load balancing, to route traffic to instances in VPC and an on-premises location**
* provides **High Availability**, by allowing more than one AZ to be specified
* integrates with **ACM** to provision and bind a SSL/TLS certificate to the load balancer thereby making the entire SSL offload process very easy
* supports **multiple certificates** for the same domain to a secure listener
* supports **IPv6** addressing, for an Internet facing load balancer
* supports **Cross-zone load balancing**, and cannot be disabled.
* supports **Security Groups** to control the traffic allowed to and from the load balancer.
* provides **Access Logs**, to record all requests sent the load balancer, and store the logs in S3 for later analysis in compressed format
* provides **Delete Protection**, to prevent the ALB from accidental deletion
* supports **Connection Idle Timeout** – ALB maintains two connections for each request one with the Client (front end) and one with the target instance (back end). If no data has been sent or received by the time that the idle timeout period elapses, ALB closes the front-end connection
* integrates with **CloudWatch** to provide metrics such as request counts, error counts, error types, and request latency
* integrates with **AWS WAF**, a web application firewall that helps protect web applications from attacks by allowing rules configuration based on IP addresses, HTTP headers, and custom URI strings
* integrates with **CloudTrail** to receive a history of ALB API calls made on the AWS account
* back-end server authentication is NOT supported

### [**Network Load Balancer**](https://jayendrapatil.com/aws-elb-network-load-balancer/)

* handles **volatile workloads** and **scale to millions of requests per second**, without the need of pre-warming
* offers **extremely low latencies** for latency-sensitive applications.
* provides **static IP/Elastic IP addresses** for the load balancer
* allows **registering targets by IP address, including targets outside the VPC (on-premises) for the load balancer**.
* supports **containerized applications**. Using Dynamic port mapping, ECS can select an unused port when scheduling a task and register the task with a target group using this port.
* monitors the health of its registered targets and routes the traffic only to healthy targets
* enable **cross-zone loading** balancing only after creating the NLB
* preserves **client side source IP** allowing the back-end to see client IP address. Target groups can be created with target type as instance ID or IP address. If targets registered by instance ID, the source IP addresses of the clients are preserved and provided to the applications. If register targets registered by IP address, the source IP addresses are the private IP addresses of the load balancer nodes.
* supports both network and application target health checks.
* supports long-lived TCP connections ideal for WebSocket type of applications
* supports **Zonal Isolation**, which is designed for application architectures in a single zone and can be enabled in a single AZ to support architectures that require zonal isolation

### 

### **Auto Scaling**

* + ensures correct number of EC2 instances are always running to handle the load by scaling up or down automatically as demand changes
  + **cannot** span multiple regions.
  + attempts to distribute instances evenly between the AZs that are enabled for the Auto Scaling group
  + performs checks either using EC2 status checks or can use ELB health checks to determine the health of an instance and terminates the instance if unhealthy, to launch a new instance
  + can be scaled using manual scaling, scheduled scaling or demand based scaling
  + **cooldown period** helps ensure instances are not launched or terminated before the previous scaling activity takes effect to allow the newly launched instances to start handling traffic and reduce load
* Auto Scaling & ELB can be used for **High Availability and Redundancy** by spanning Auto Scaling groups across multiple AZs within a region and then setting up ELB to distribute incoming traffic across those AZs
* **With Auto Scaling use ELB health check with the instances to ensure that traffic is routed only to the healthy instances**