**AWS Certified Solutions Architect - Professional**

**ADVANCED PERMISSIONS & ACCOUNTS**

AWS Organizations

* Separate Payment and set of IAM users for separate accounts
* Organization is created from an AWS account, which will become Management Account
* Other existing accounts can be invited
  + Becoming Member accounts
  + Billing will pass through Payer/Management Account
  + Cost benefits are also pooled(reservations, volume discounts)
* New Accounts can be created inside the orga
* Hierarchical Structure: Organization Root -> Organizational Units (OUs)
* Best practice: Have single account for login or use identity federation(SSO)
  + Use role switch to access other member accounts (assume role in other accounts)

AWS RAM

* Share resources between principals(accounts), product needs to support RAM
* No extra cost
* Owner retains full ownership

Service Quotas

* <https://docs.aws.amazon.com/general/latest/gr/aws-service-information.html>
* <https://us-east-1.console.aws.amazon.com/servicequotas/home/>

**ADVANCED IDENTITIES & FEDERATION**

SAML2.0 Identity Federation

* Security Assertion Markup Language
* Indirectly use onprem ID’s in AWS, works with Enterprise Identity Provider(IDP) compatible with SAML 2.0
* Giving a Single source of truth and can be maintained by existing identity management team
* Uses IAM Roles & Temporary credentials (12h)
* Two way trust between IAM and IDP
* Identities are mapped to roles within IDP
* Communication with SAML Assertion over STS:AssumeRoleWithSAML
* Works for Application access as well as AWS console access

IAM Identity Center( AWS SSO)

* Manage SSO Access – AWS Accounts and External Applications with a flexible Identity Source
* Built-in Identity Store, AWS Managed Microsoft AD, On-premises Microsoft AD, External IDP
* **Preferred** by AWS vs traditional identity federation for **Workplace** Identities
* Allows SSO via Identity Source and Provides Centralized Management
* Works for Stuff like Dropbox, MS365, Slack, Salesforce etc.

Amazon Cognito

* Authentication, Authorization and user management for apps
* USER POOLS: Sign-up/in and get Jason Web Token (JWT)
  + User directory management and profiles, MFA, etc.
* IDENTITY POOLS: Allow to access Temporary AWS Credentials
  + Unauthenticated Identities – guest users
  + Federated Identities, SWAP (Google, FB, SAML, etc.) for short term AWS creds
* USER POOLS and IDENTITY POOLS can be combined

Amazon Workspaces

* Desktop as a Service like Citrix
* Windows and Linux
* Directory Service for authentication and user management
* Workspaces run in a AWS managed VPC and use ENI for injection into customer VPC
* Encrypted at rest
* Not HA by design

Directory Service – Microsoft AD

* Built using Microsoft AD 2012 using Standard Active Directory tools
* Group policies, SSO, Schema extension, SharePoint, SQL, DFS
* HA by default
* Supports one-way and two-way external and forest trusts
* RADIUS-based MFA (SAMBA implementation)
* Best choice for more than 5000 users and if trust relationships between AWS and on-prem needed
* Identity Federation with other clouds, Azure AD or MS365

Directory Service AD Connector

* Pair of directory endpoints running in AWS (ENI’s in VPC)
* Redirects requests to existing directory servers (No directory data in AWS)
* Use existing on-prem AD with AWS directory compatible services
* Small and large size available, multiple connectors can be used though
* Requires working network connection (Direct Connect or VPN)
* Useful for PoC or for legal or compliance reasons or small projects

Control Tower

* Quick and easy multi-account environments setup
* Orchestrates other AWS services (Organizations, IAM identity Center, Cloudformation, Config, etc.)
* Landing Zone (SSO Federation, Centralized Logging & Audit)
* Guard Rails – Automates and Standardizes new Account creation
* Management Account with Control Tower, Organizations and SSO
* Log Archive Account
* Audit Account
* Account Factory provisions member accounts with SCPs and Guardrails

**Networking & Hybrid**

Private and Public AWS Services

* AWS Public Zone is connected to Public Internet and Public Services run from
* AWS Private Zone is where VPC’s and services in them run from

DHCP in a VPC

* Dynamic Host Configuration Protocol 🡪 auto configuration for network resources
* Starts with Layer 2 broadcast to get (IP address, Subnet Mask & Default GW)
* Configures DNS Servers, Domain, NTP, NetBios, NodeType
* AmazonProvidedDNS and Custom DNS Domain
* DHCP Option Sets cannot be edited and be associated with 0 or more VPCs, each VPC can have only 1 Option Set attached
* VPC Router = Subnet +1, R53 Resolver = Subnet + 2
* AWS Supply public and private DNS names

VPC Router

* Virtual Router within VPC, HA within all AZs in the region, Scalable
* Routes traffic between subnets and from external networks into VPC and other way around
* Interface in every subnet with subnet+1 address
* Controlled with VPC route tables
* Every VPC is created with Main route table(default RT)
* Subnets are associated with exactly one RT
* Most specific routes have highest priority

Stateful vs Stateless Firewalls

* Every connection has two parts – REQUEST and RESPONSE
* Directionality INBOUND and OUTBOUND depend on perspective
* Stateless
  + Requires two rules per connection (one for INBOUND and one for OUTBOUND)
  + response uses ephemeral ports
* Stateful
  + Is intelligent and identifies REQUEST and RESPONSE of a connection
  + Only REQUEST needs to be allowed but not the RESPONSE

Network Access Control Lists (NACL)

* Associated with subnets and filter data crossing the subnet INBOUND or OUTBOUND
* Can be associated with MANY subnets
* Stateless (Rule needed for INBOUND and OUTBOUND)
* Offer explicit allows and explicit denies
* Processed in order, lowest rule number first, processing stops with a match
* Default NACL are not used, all traffic allowed

Security Groups

* Stateful – response is automatically allowed
* No explicit deny only allow 🡪 can’t block bad actors
* Supports CIDR and logical resources (eg. Security groups)
  + Security group can also reference itself
* Attached to ENI’s not instances

AWS Local Zones

* Offer really low latencies by giving you for example a city as a Zone
* Local zones are related to parent regions but have independent internet connection
* Support direct connect
* No built-in resilience, like a single AZ but near your location

Border Gateway Protocol

* Autonomous System (AS) – Routers controlled by one entity
* ASN unique and allocated by IANA (0-65535), 64512-65534 are private
* Operates over tcp 179
* Not automatic – peering is manually configured
* Path-vector protocol, exchanges the best path (ASPATH)
* iBGP (internal BGP), eBGP (external BGP)
* AS Path Prepending can be used to artificially make paths longer
* AS propagates all shortest paths to all it’s peers

AWS Global Accelerator

* 2 anycast IP Addresses(allow single IP to be in multiple locations)
* Traffic initially uses public internet & enters Global accelerator edge location
  + From there traffic is going over AWS network to destination 🡪 less hops and improved performance
* Moves AWS network closer to customers
* Can be used for NON HTTP/S (TCP/UDP) – Difference to Cloudfront

IP Sec VPN

* Group of protocols for secure tunnels across insecure networks between two peers
* Provides Auth and encryption
* Two main phases with:
  + IKE Phase 1 (slow and heavy): Authenticate – create symmetric key
  + IKE Phase 2 (fast and agile): uses phase 1 key, agrees on encryption and creates tunnel

Site2SiteVPN

* Logical connection between VPC and on-prem network encrypted using IPSec
* Full HA, Quick to provision
* VPN connection between Virtual Private Gateway and Customer Gateway
* Considerations: Speed Limitations(1.25 Gbps), latency considerations, cost, speed of setup, can be used as backup for Direct Connect or used with Direct Connect

Transit Gateway

* Network Transit Hub to connect VPCs and VPN
* Reduces network complexity
* Single network object, Attachments to other network types
* Considerations:
  + Supports transitive routing, can be used to create gobal networks
  + Can be shared with other accounts using RAM
  + Can be peered with different regions and accounts
  + Less complex than VPC peering etc

VPC Routing

* Subnets only 1 RT, RT can be associated with IGW or VGW
* Static and propagated routes
* Prio: Longest Prefix 🡪 Static Routes 🡪 Propagated Routes 🡪 DX, VPN Static, VPN BGP, AS\_PATH
* CIDR Overlap:
  + Peering between overlapping CIDRs isn’t supported, but possible if paired from VPC C
  + Possible workarounds:
    - Split subnets in VPC C for each paired VPC
    - Add more specific routes (e.g. /32)
* Ingress: Using Gateway Route Tables to direct inbound traffic, e.g. forwarding through Firewall

Accelerated Site-To-Site VPN

* Performance enhancement to VPN
* Traditional VPN: 2 IPSEC Tunnels between VGW and CGW , physically indirect transit path (public internet)
* Using TGW: allows pair of VPN tunnels to provide access to many VPCs, still over public internet
* Accelerated VPN: Global accelerator network is used 🡪 IP’s are global and connections routed to closest edge location
  + Combination of Site to Site VPN, Global accelerator and TGW
  + less public internet and better connection
  + Fixed fee + transfer fee

AWS Client VPN

* Managed OpenVPN implementation, Connecting to a Client VPN Endpoint
* Associated to ONE VPC
* 1 + Target Networks (HA)
* Billed based on network associations
* Clients are handled like any other instance within a VPC, so IGW etc. is needed as well
* Split Tunnel (not default): offers internet access without using the Client VPN Endpoint

AWS Direct Connect

* Physical connection(cable) into AWS region (1, 10 or 100 Gbps)
* Business Premises 🡪 DX Location 🡪 AWS Region
* Port Hourly Cost + Outbound Data Transfer Cost
* Low Latency + High Speed
* Offers MACSec
* Authorization is needed from all parties and needs (LOA-CFA)
* Offers Virtual Interfaces(VIFs) to connect to TGW and stuff (using VLANs and BGP)
* Private VIFs to access 1 VPC using private IPs
  + No encryption on private VIFs
  + Max 100 specific corp prefixes
  + VPC needs to be in same region as DX location
  + IPv4 or IPv6
  + You configure YOUR ASN on the VIF (Private ASN 64512 to 65535) or public owned one
* Public VIFs access Public Zone services (elastic Ips, SNS,SQS, S3 etc.)
  + No direct access to VPC services
  + Not transitive (prefixes don’t leave AWS)
* VPN can be run over direct connect (encrypted and authenticated tunnel with low and consistent latency)
* Direct Connect Gateway: Global network device, accessible in all regions
  + Associate with VGW’s attached to VPCs globally, no inter VPC routing allowed
  + 500 VPCs can be connected, also over multiple Accounts
* Transit VIFs can be used with TGW(Hub-Spoke Architecture) to route between VPCs
* Resilience can be problematic if only one cable is used
  + Can be improved with multiple devices at direct connect and customer location, or even using different customer locations and multiple locations with multiple devices
* Link Aggregation Groups (LAG): Multiple Physical connections act as one – Speed \* n
  + Active / Active Architecture, max 4 connections per LAG

Route53

* Register Domains, Global service with single Database, globally resilient
* R53 can be Domain Registrar or Domain Hosting
* Hosted Zones on AWS managed nameservers (DNS as a service)
  + Stores records
    - NS: Name Server
    - A & AAAA: Map hostname to IP (IPv4 and IPv6)
    - CNAME: Canonical Name, Host to Host records, pointing to A records
    - Alias:
      * CNAME can’t point to naked domain, Alias can
      * Alias map Name to AWS resource, without charge
    - TXT: Add text to domain for additional functionality (origin, etc.)
    - MX: To find Mail Server, Consists of Priotity and location
  + TTL (Time To Live): Seconds how long DNS result is cached (important for changes)
  + Public Hosted Zone:
    - DNS Database (zone file) hosted by R53
    - Accessible from public internet and VPCs
    - Hosted on 4 NS specific for the zone
    - External registered domains can point at R53 domain
  + Private Hosted Zone:
    - Associated with VPCs (cross account) and only accessible in those
    - Can be combined with Public Hosted Zones
* Health Checks:
  + Separate but used by records, located globally
  + Every 30s or 10(increased cost)
  + TCP, HTTP/HTTPS or HTTP/HTTPS with String Matching
* Simple Routing: Just does routing, No Health Checks
* Failover Routing: Primary and Secondary Name as failover with Health Check
* Multi Value Routing: Multiple endpoints for value, selecting endpoint at random
* Weighted Routing: Multi Value but with weights for endpoints
* Latency Routing: Multi Value but with checking where lowest latency is
* Geolocation Routing: Records are tagged with location(country or continent) or default and matched with user location
* Geoproximity Routing: Records(tagged by AWS region or coordinates) are selected by closest location to user, bias can be defined to change distances for Zone
* DNSSEC can be used with R53

AWS Private Link

* Securely connect services between different AWS Accounts over a VPC endpoint and LBs
* HA via multiple Endpoints
* Only IPv4 & TCP, Private DNS supported

VPC Endpoints

* Gateway Endpoints:
  + Private access to S3 and DynamoDB
  + Prefix List added to route table 🡪 Gateway Endpoint
  + HA across all AZs by default, regional(no cross region)
  + Endpoint policy to control access
  + Not accessible outside the VPC
  + Network Interfaces within the VPC
* Interface Endpoints:
  + Private access to AWS Public Services
  + Added to specific subnets 🡪 not HA by default
  + Access controlled via Security Groups and Endpoint Policies
  + Uses Private Link
  + Provides a NEW service endpoint DNS
    - PrivateDNS overrides default DNS
* Endpoint Policies:
  + Limits access via that endpoint only
  + Contains a principal
  + Commonly used to limit what private VPC can access
* Advanced VPC DNS & DNS Endpoints
  + .2 reserved in every subnet (Route53 Resolver), provides R53 Public & Associated Private Zones
  + Route 53 Endpoints, Accessible over VPN or DX
    - Inbound: Onprem can forward to R53 resolver
    - Outbound: Conditional Forwarders, R53 to onprem
    - Rules control what requests are forwarded

IPv6 Capabilitiy in VPCs

* All IPv6 addresses in AWS are publicly routable, NAT is not used
* Not every service supports IPv6
* Needs to be enabled on VPC, AWS gives unique /56 range, 256 /64 subnets
  + Existing VPC, subnet can be migrated to v6
* IPv6 and IPv4 Routing is handled separately
  + IPv6 should have a separate Egress only IGW
  + IPv4 has the normal IGW

How many AZ’s

* AZs in region minus Buffer AZs = Nominal AZs
* Minimum App Requirements = Nominal Instances
* Nominal Instances / Nominal AZs = Optimal Number of Instances per AZ
* Highest Number is always optimal
* Example 6 Instances, minimum required also 6
  + 1 AZ is bad
  + 2 AZ would need 6 in each Zone 🡪 Not cost effective (100% over Provision)
  + 3 AZ, 3 instances in each 🡪 More Cost efficient (50% over Provision)

Subnets & Tiers

* Traditional applications: Presentation Tier, Logic Tier, Data Tier with own networking and security configuration with Firewalls in-between
* 1 Tier design using NACLs, also with multiple AZs
* 2 Tiers design with public and private subnets and security groups and NACLs, also with multiple AZs, separating of routing with route tables
* Internet-facing ALBs can communicate with private instances
* Number of APP subnets \* AZs = minimum number of subnets needed

**Storage Services**

FSx for Windows File Server

* Fully managed native windows file shares
* Integrates with Directory Service or Self-Managed AD
* Single or Multi AZ within a VPC, On-demand and scheduled Backups
* Accessible using VPC, Peering, VPN, Direct Connect
* VSS User Driven Restores
* Accessible over SMB, Windows permission model
* Integrates with DS and own directory

FSx for Lustre

* Managed Lustre, Designed for High Performance Computing, Linux
* Machine Learning, Big Data, Financial Modelling
* 100s GBs throughput & sub milliseconds latency
* Persistent(HA, self healing) or Sratch(short term, no replication, fast)
* Accessible over VPN or Direct Connect

EFS

* NFSv4 Filesystems that can be mounted in Linux(only)
* Shared between EC2 instances
* Private service via mount targets inside a VPC
* Can be accessed from on premises VPN or DX
* General Purpose or Max I/O Performance Modes
* Bursting or Provisioned Throughput Modes
* Standard vs Infrequent Access(IA) Class, Lifecycle policies available

S3

* Storage Classes
  + Standard:
    - **For frequently accessed data which is important**
    - Objects are replicated across at least 3 AZs
    - Billed GB/m fee and for transfer OUT and per 100 requests
    - No retrieval fee, no min duration, no min size
  + Standard-IA:
    - **Used for Long-lived important data with infrequent access**
    - Cheaper Storage Cost but additional retrieval fee
    - Min duration of 30 days, min capacity charge 128 KB
  + One-Zone-IA:
    - **Used for Long-lived non-critical data with infrequent access**
    - Cheaper than Standard-IA but Objects are not replicated in AZs
  + Glacier Instant Retrieval:
    - **Used for Long-lived important data with infrequent access(once per quarter)**
    - Like Standard-IA but min duration of 90 days and cheaper
  + Glacier Flexible Retrieval:
    - **Used for Archival Data, where access isn’t needed frequently(yearly)**
    - Cheaper but Cold Storage(cannot made public) and require retrival process
      * Expedited(1-5min), Standard(3-5h), bulk(5-12h)
  + Glacier Deep Archive:
    - **Used for Archival Data, which rarely or never needs to be accessed (legal or regulation data storage)**
    - Even cheaper with more restrictions(180 days min duration)
      * Standard(12h), Bulk(48h)
  + Intelligent Tiering:
    - **Monitors and automatically moves objects between Tiers**
    - Contains 5 different Tiers (similar to the other Classes)
      * Frequent, Infrequent, Archive Instant, Archive, Deep Archive
* S3 Select and Glacier Select
  + Lets you use SQL-Like statements to Select parts of objects 🡪 faster and cheaper
  + CSV, JSON, Parquet, BZIP2
* Lifecycle Configuration
  + Set of rules consisting of actions on a Bucket or group of Objects
  + Transition or Expiration Actions
* S3 Replication
  + Cross Region Replication or Same Region Replication between 2 Buckets
  + Cross Account possible as well
  + All Objects or a subset, Default is to maintain Storage Class, Ownership can be changed (for cross account replication), RTC – SLA for 50 min replication
  + By Default Not retroactive & Versioning needs to be ON
  + Batch replication can be used, One-way replication
  + Unencrypted or SSE-S3 or SSE-KMS or SSE-C
  + Source bucket owner needs permissions to objects
  + NO system events, Glacier or Deep Archive and NO Deletes are replicated
* S3 Encryption
  + Buckets aren’t encrypted, objects are
  + Encryption at Transit
  + Server Side Encryption(SSE) is mandatory
    - Objects aren’t initially encrypted, getting encrypted in S3
    - SSE with Customer-Provided Keys(SSE-C)
      * Customer manages Keys, S3 manages Encryption
    - SSE with Amazon s3-Managed Keys(SSE-S3) – default
      * S3 manages Keys and Encryption
    - SSE with KMS Keys(SSE-KMS)
      * KMS manages Keys and Encryption, KMS generates encryption keys
      * Customer managed KMS keys possible, Key rotation, role separation
  + Client Side Encryption
    - Objects are encrypted by client before uploading
    - Key is owned by Client
  + S3 Bucket Keys
    - Normal SSE-KMS can have cost and throttling problems
    - Time limited Bucket Key is used for each object inside the bucket
    - CloudTrail KMS now shows bucket ARN
    - Works with replication
* S3 Presigned URLs
  + URL created for an object to give matching permissions of entity(IAM user) that generated the URL
  + Don’t generate with a IAM role (URL stops worling when temporary creds expire)
  + For downloads(GET) and uploads(PUT)
* S3 Access Points
  + Simplify managing access to S3
  + Many access points with different policies with different network access control with own endpoint address
  + Granular permissions for bucket/objects
  + Tied to VPC over VPC endpoint
* S3 Object Lock
  + Can be enabled on new buckets and can’t be disabled
  + Write-Once-Read-Many – No Delete, No Overwrite
  + Requires Versioning
  + Retention Period and Legal Hold can be defined
* Amazon Macie
  + Data Security and Privacy Service for S3
  + Automated Discover, Monitor and Protect Data in S3 (Keys, Finance Data, etc.)
  + Managed Data Identifiers using ML and pattern matching
  + Custom Data Identifiers – Regex Based
  + Integrates with SecurityHub and EventBridge
  + Discovery Jobs get Findings
  + Can be automated with EventBridge and Lambdas

EBS

* General Purpose SSD
  + GP2:
    - 1GB – 16TB
    - IO Credit=16KB, Capacity of 5.4 million credits(starting full), 100 credits per second + 3 per GB of volume (max 16k IO per second), Burst up to 3000 IOPS
    - For boot volumes, low-latency apps, dev & test
  + GP3:
    - 3000 IOPS & 125 MiB/s – Standard
    - Extra Cost for up to 16k IOPS or 1000 MiB/s
    - 20% Cheaper
    - For virtual desktops, medium sized single instance DBs, low latency apps, dev & tes, boot volumes
* Provisioned IOPS
  + IOPS can be adjusted independently of size
  + Per Instance Performance are 4 Volumes
  + IO1: 64k IOPS or 1000 MiB/s, 4GB – 16TB
  + IO2: 64k IOPS or 1000 MiB/s
  + IO2-Block Express: 256K IOPS or 4000 MiB/s, 1GB – 64TB
  + For High Performance, latency sensitive workloads, intensive NoSQL & SQL
* HDD-Based
  + Cheaper than GP, 125GB -16TB
  + St1: Throughput Optimized, Max 500 IOPS or 500 MB/s
    - For Big Data, data Warehouse, Log Processing
  + Sc1: Cold HDD, cheapest EBS, Max 250 IOPS or 250 MB/s
    - For Colder data requiring fewer scans per day

Instance Store Volumes

* Block Storage Devices, Physically connected to ONE EC2 Host
* Instances on that host can access them
* Highest storage performance
* Ephemeral Volumes, Data can get lost on move, resize, hw failure 🡪 for temporary data
* **Attached at launch**
* **More IOPS and Throughput vs EBS**

Instance Store vs EBS

* Persistence 🡪 EBS
* Resilience 🡪 EBS
* Storage isolated from instance lifecycle 🡪 EBS
* Resilience w In built replication 🡪 depends
* High performance needs 🡪 depends
* Super high performance needs 🡪 Instance Store
* Cost 🡪 Instance Store

AWS Transfer Family

* Provides Managed file transfer service
* Managed servers which support protocols(FTP, FTPS, SFTP, AS2)
* Managed File Transfer Workflows – serverless file workflow engine
* Endpoint Types: Public, VPC – Internet, VPC - Internal

**Compute, Scaling & Load Balancing**

Regional and Global AWS Architecture

* Global:
  + Service Location & Discovery
  + Content Delivery and Optimization
  + DNS, Health checks & Failover
* Regional:
  + Entry points
  + Scaling & Resilience
  + Application services and components

EC2 Purchase Options

* On-Demand (Default):
  + Isolated but multiple customers on same hardware
  + Per second billing for running instance
* Spot:
  + AWS selling unused EC2 host capacity for up to 90% discount
  + Non time critical, Anything that can be rerun, Anything stateless
* Reserved:
  + Commitment for 1 or 3 years of resources
  + Partial coverage of larger instances
  + No-Upfront(least discount), Full-Upfront(greatest discount), Partial-Upfront
  + Scheduled Reserved Instances:
    - For long term usage which doesn’t run constantly
  + Capacity Reservations:
    - Regional Reservation Provides billing discount in region
    - Zonal Reservation only applies to one AZ
* EC2 Savings Plan:
  + Hourly commitment for general compute(EC2, Fargate, Lambda) or specific EC2
  + Products have on-demand rate and savings plan rate
  + Beyond commitment on-demand is used
* Dedicated Hosts:
  + Physical Host is bought and instances on the host don’t have extra cost
  + Licensing based on Sockets/Cores
* Dedicated Instances:
  + Dedicated Hardware on Host
  + Extra charges for instances

EC2 Networking

* EC2 have primary ENI which cannot be removed, additional ENIs can be added from other subnets(Not in other AZs)
* Multi-ENI offers multiple security zones or traffic types
* Each ENI can also be protected by NACL around its subnet
* 1 Primary static private IPv4 address, 1 or more secondary private IPv4 addresses
* 1 Public IPv4 address which can change
* 1 Elastic IP per private IPv4

Bootstrapping vs AMI Baking

* Elastic(Autoscaling + LB) with Scaling(Launch, Install, Configure, Test) 🡪 Ready/Complete or Fail
* Bootstrapping:
  + Provision EC2, Add Script to User Data
  + Flexible but takes time
* AMI Baking
  + Launch master EC2, perform tasks, create AMI from it
  + AMI can be used to quickly provision new instances
* Both approaches can be combined

ELB

* Configured to run in 2+ Azs
* Each ELB is configured with alias A record
* Listeners accept traffic on port and protocol
* Internet facing vs Internal(only private Ips)
* EC2 doesn’t need to be public
* Session State:
  + Persistent Server-side piece of information(Shopping Cart, Login state, etc.)
* Session Stickiness:
  + Generates a cookie which locks device to single backend instance for duration
  + Creates AWSALB held by client
  + Sessions move on expiry or instance failure
  + Useful if app doesn’t use external sessions
* ELB evolution: avoid v1(Classic LB), lacking features use vs(ALB, NLB)
* ALB:
  + Only Layer 7(HTTP/s)
  + Traffic terminated at LB and new connection made to Application
  + Must have SSL cert
  + Slower than NLB
  + Evaluate application health
  + Rules direct connections which arrive at listener
    - Priority order
    - Rule conditions based on headers etc.
    - Actions like forward
* NLB:
  + Layer 4, TCP, TLS, UDP
  + No visibility or understanding of HTTP/s, no headers, cookies, stickiness
  + Really fast, can have static IPs
  + For SMTP, SSH, Game Servers, financial apps
  + Health checks not application aware
  + Forward TCP to instances unbroken
  + Can be used with Privatelink

Auto Scaling Groups

* Are free, only resources created are billed
* Cooldowns to avoid rapid scaling
* Use more, smaller instances 🡪 granularity
* Use with ALBs 🡪 abstraction
* AGS define WHEN and WHERE, LT defines WHAT
* Scaling Policies: Manual, Scheduled, Dynamic(Simple, Stepped, Target Tracking)
* Lifecycle Hooks:
  + Custom Actions on instances during ASG actions(instance launch or terminate)
  + Until a timeout (CONTIUNUE or ABANDON)
  + Used with Eventbridge or SNS notifications
* Health Checks: EC2(EC2 state), ELB(application aware) & Custom(external system)
* Connection Draining: Allows in-flight requests to complete before instance removal(only CLB)
* Deregistration Delay: ALB, NLB defined on Target group until connections complete naturally or delay is reached

X-Forwarded-For & Proxy Protocol

* Connection is made from LB and has no customer IP
* X-Forwarded-For: HTTP header containing customer IP and LBs, not supported with NLB
* Proxy: Layer 4 header, works with CLB and NLB, E to E encryption

EC2 Placement Groups

* Cluster: Pack instances close together(direct connection), one AZ only, best network perf
* Spread: Keep instances separated, for isolation
* Partition: groups of instances spread apart, topology aware application

Gateway Load Balancer

* Run and scale 3rd party appliances(Firewalls, intrusion detection etc.)
* Inbound and Outbound traffic
* GWLB endpoints
* GWLB balances across multiple backend appliances using GENEVE protocol

**Monitoring, Logging & Cost Management**

CloudWatch

* Ingest, Store and Manage Metrics
* Namespace = container for metrics
* Datapoint = Timestamp, Value
* Metric = time ordered set of Data points
* Dimension = name/value pair
* Resolution: Standard = 60s, High = 1s
* Retention gets bigger with lower resolutions
  + As Data ages its aggregated with less resolution
* On-prem Integration via Agent/API
* Application Integration via API/Agent
* Alarms: watches metric over time period(ALARM or OK)
  + Value of metric vs threshold
  + One or more actions
* CloudWatch Logs
  + Public Service to Store, Monitor and Access logging data, AWS, On-Prem, IOT, etc.
  + Ingestion:
    - CWAgent system or custom application logging
    - VPC Flow Logs, ELB, ECS, API GW, Lambda, CloudTrail, Route53, etc.
  + Subscription:
    - Export to S3
    - Near realtime persist logs 🡪 Kinesis Firehose
    - Realtime 🡪 Lambda or Kinesis Data Stream
    - Elasticsearch

CloudTrail

* Logs API calls/activities as CloudTrail Event for 90 days by default
* Management(by default) and Data events
* IAM, STS, CloudFront 🡪 Global Service Events
* NOT realtime

AWS X-Ray

* Distributed tracing application – track requests through applications
* Service Graph: JSON detailing services and resources which make up application
* Service Map: Visual Version of Service Graph
* EC2(X-Ray Agent), ECS(Agent in Task), Lambda, Beanstalk, API GW, SNS & SQS
* Requires IAM permissions

Cost Allocation Tags

* Have to be enabled per account or for orga
* Visible in cost reports
* AWS-Generated: aws:createdBy or aws:cloudformation:stack-name
* User-defined: user:something

Trusted Advisor

* Account level, no agent needed
* Cost Optimization, Performance, Security, Fault Tolerance and Service Limits
* 7 core checks with basic & developer support plans, all other checks(115) with Business or Enterprise support
  + Core: S3 Bucket Permissions, Security Groups, IAM Use, Root MFA, EBS/RDS Public snapshots, Service limit checks
* Access via the AWS Support API

**Databases**

RDS

* Database Server as a Service, multiple databases on one Server(instance)
* MySQL, MariaDB, PostgreSQL, Oracle, Microsoft SQL
* Subnet Group to specify which subnets RDS can use
* Dedicated EBS storage per instance
* Asynchronous Replication to Read Replicas
* Cost: Instance Size + Type, Multi AZ, Data transferred, Backup & Snapshots, Licence
* MultiAZ:
  + Instance Deployment: Synchronous Replication to StandBy
    - Snapshots in S3
    - One StandBy ONLY, can’t be used for reads
    - Only same region
  + Cluster Deployment: Synchronous Replication to Readers
    - Data committed when 1+ reader finishes writing
    - Cluster Endpoint, Reader endpoints and Instance Endpoints(nbot recommended)
    - Readers can be used for scaling
    - Faster failover
* Backup & Restore:
  + Snapshots are incremental
  + Automated Backups(Snapshots) in AWS Managed S3 Buckets(0 to 35 days)
  + Manual Snapshots in AWS Managed S3 Buckets
  + Can be replicated to another region, needs to be configured
  + Restore creates a new RDS instance(not fast process) 🡪 Good RPO, RTO not so
* Read Replicas:
  + 5x per DB instance, providing additional instance of performance
  + Read replicas can have replicas as well with more lag
  + RR offer near zero RPO and low RTO but only for failure and not corruption
  + Read only until promoted
* Data Security:
  + SSL/TLS(in transit) available and can be mandatory
  + Supports EBS volume encryption with KMS
  + Data Keys(generated by KMS) used for encryption operations
  + Storage, Logs, Snapshots & replicas are encrypted
  + Encryption can’t be removed
  + MSSQL and Oracle support Transparent Data Encryption(TDE), Oracle with CloudHSM 🡪 Much stronger key controls
  + IAM Authentication: Policy Attached to Users or Roles maps to IAM Identity onto local RDS user
  + Authorization is controlled by DB engine

Aurora

* Uses a Cluster, single primary + 0 or more replicas
* All SSD based, storage based on what’s used
* Replicas can be added and removed without storage provisioning
* No free Tier or Micro size, Offers better value than RDS
* Backup works the same, also allows Backtrack(roll back to previous point in time) and fast clones
* Aurora Serverless: Scalable Aurora Capacity Units(ACU), Min & Max ACU
  + Cluster adjusts on load. Consumption billing per second
  + For infrequently used Applications and New applications, Variable/Unpredictable workloads, Dev and Test DB’s
* Multi-Master: Multiple Write Instances, no load balancing
* Aurora Autoscaling for Read Replicas(not for master)

RDS – Proxy

* Proxy maintains connection pool to Database to not open close connection all the time
* For smaller DBs, in combination with Lambdas, can reduce time for failover
* Fully managed for RDS/Aurora, auto-scaling, HA
* Only accessible from VPC

RDS Custom

* Fills gap between RDS and Database running on EC2
* Can connect using SSH, RDP, Session Manager

DynamoDB

* NoSQL Public Database as a Service, Key/Value & Document
* Manual, Automatic provisioned performance IN/OUT or On-Demand
* Optionally global resilience
* Really fast, SSD based, Backups, point-in-time recovery, encryption at rest
* DynamoDB tables: grouping of Items with same primary key
* On-Demand Backups and Point-in-time- Recovery
* Access via console, CLI, API
* Billed based RCU, WCU, Storage and features
* Reading and Writing On Demand or Provisioned(Every operation consumes at least 1 RCU/WCU)
  + 1 RCU is 4KB read operation per second
  + 1 WCU is 1kb write per second
  + 300 seconds RCU and WCU burst pool
* Eventually vs Strongly Consistent
* DynamoDB Indexes:
  + Query is most efficient operation but only works on 1PK at a time
  + Indexes are alternative views on table data
  + Local Secondary Indexes(LSI): Different Secondary Key
    - Must be created with a table, Shares RCU and WCU with table
  + Global Secondary Indexes(GSI): Different Primary and Secondary Key
    - Can be created at any time, Have own RCU and WCU
  + Some or all attributes(projection)
* DynamoDB Streams:
  + Time ordered list of item changes(Insert, Update, Delete) in a 24 hour rolling window
* DynamoDB Triggers:
  + Allow for actions to take place on Data change based on stream using Lambda(Reporting and Analytics, Aggregation, Notification)
* DynamoDB Accelerator(DAX): Integrated In-Memory cache 🡪 much faster and reduce cost, Scale up and out, supports write-trough
* DynamoDB Global Tables: Multi-master cross region replication
  + Last writer wins, Read and write in any region, sub-second replication
  + Strongly consistent reads only in same region
* DynamoDB TTL: Timestamp for automatic Delete of items

Open-/Elasticsearch

* Managed implementation of ES(open source search product)
* ELK stack: ES(Search and Indexing), Kibana(visualization/ dashboards) and Logstash(needs Logstash agent installed)
* Not serverless

Athena

* Serveless Interactive Querying service, only pay data consumed
* Schema-on-read: translates data to relational-like data when reading S3
* Output can be sent to other services
* For occasional/Ad-hoc queries on S3(Logs, Glue Data Catalog), cost conscious
* Athena Federated Query for other data sources

Neptune

* Managed Graph DB: relationships are as important as data
* Like RDS but for graph data(social media, fraud prevention, dependencies)

Quantum Ledger

* Serverless Immutable, Transparent append-only ledger based DB(blockchain)
* For Finance, Medical, Logistics, Legal, etc.

**Data Analytics**

Kinesis

* Data Stream: Scalable public & HA streaming service
  + Producers send data into stream, consumers access data from moving window
  + 24h moving window of data, can be increased to 365 days
  + Designed for huge scale ingestion and multiple consumers(Analytics, Monotoring, etc.)
* Data Firehose: Fully managed service to load data for data lakes/stores and analytics
  + Automatic scaling, serverless, near Real time (~60s)
  + Supports transformation on the fly with lambda
  + Billing based on volume
* Data Analytics: Real time processing of data using SQL(between 2 data streams/firehose)
  + Ingest from Streams or Firehose
  + Destinations: Firehose (S3, Redshift, ES), Lambda, Data Streams
  + Use for: Streaming data needing SQL processing, complex data manipulation

MapReduce

* Data Analysis Architecture – huge scale, parallel processing
* Two Main Phases: Map and Reduce
* Hadoop File System: Highly Fault-tolerant
* Elastic Map Reduce (EMR): Managed Apache Hadoop service, Spark, Hbase, Presto, Hive, Flink
* Can be operated long term or ad-hoc clusters, auto scales
* For Big-data processing, analytics, transformation, etc.

Redshift

* Petabyte scale Data Warehouse, pay as you go
* Server based in one AZ
* Leader Node(query input, planning, aggregation) and Compute Node(performing queries)
* Redshift Enhanced VPC Routing
* OLAP (column based)
* Query S3 directly with Redshift Spectrum
* Direct query other DBs using federated query
* Redshift can copy snapshots for recovery to other regions

AWS Batch

* Managed Compute Service for Data Analytics
* Job: Script, Executable or Docker containers submitted to batch
* Job Definition: Metadata for Job including permissions
* Job Queue: Jobs are submitted to queue with priorities
* Compute Environment:
  + Managed: manage capacity based on load, On-demand or Spot instances, You can determine max spot price, runs in VPC and requires VPC gateways
  + Unmanaged: You manage everything
* Batch vs Lambda:
  + Lambda has execution limit and limited disk space
  + Batch is not serverless(using docker), no time limit or resource limit

AWS Quick Sight

* Business Analytics & Intelligence service (BA/BI)
* Visualisations, Ad-hoc Analysis
* Discovery and Integration with AWS Data sources
* For: Dashboards and Visualisation
* Athena, Aurora, Redshift, S3, IOT, Jira, Github, MSSQL, MySQL, Spark, Snowflake, etc.

**App services, Containers & Serverless**

ECS

* Container Definition(Image & Ports) -> Task Definition(Security/Task Role, Containers, Resources) -> Service Definition(Number of Tasks, HA, Restarts)
* Cluster Types:
  + EC2: Large workload and price concious
  + Fargate: Large workload and overhead concious, or small/burst workloads, batch workloads

EKS

* AWS managed Kubernetes – open source & cloud agnostic
* Control plane scales on multiple AZs
* Integrates with AWS services(ECR, ELB, IAM, VPC)
* EKS cluster = EKS Control Plane & Nodes
* Etcd distributed across multiple AZs
* Nodes: Self managed, managed or Fargate pods

SNS

* Public service that coordinates sending and delivery of messages
* SNS topics are base entity
* Publisher sends messages into Topic
* Topics have subscribers which receive messages
* Deliver Status(Including HTTP, Lambda, SQS)
* Delivery Retrys
* HA and scalable(Region)
* SSE
* Cross Account via Topic Policy

SQS

* Public Fully managed, HA Queues
* Messages up to 256KB
* Received messages are hidden(VisibilityTimeout)
* Billed based on requests
* Short(immediate) vs Long(wait Time In Seconds) Polling
* Encryption at rest(KMS) & in-transit
* Standard: At Least once delivery, Best-Effort Ordering
* FIFO: Exactly once delivery, message order strictly preserved
  + 3000 messages per second with batching, 300 without
* Extended Client Library: Used when handling messages over max SQS size
  + SendMessage uploads to S3 and link in message
  + Receive message loads large payload from S3
  + Delete Message also deletes S3 payload
* Delay Queues: Postpone delivery of messages to consumer
* Dead-Letter Queues: to handle problematic messages with help pf redrive policy and maxRecieveCount

Amazon MQ

* Open source message broker based on Apache ActiveMQ
* Provides Queues and Topics(One to One and One to Many)
* Single Instance or HA Pair, runs in VPC(not public like SQS and SNS)
* No AWS native integration
* For: Migrating from existing system or if protocols like AMQP, MQTT, OpenWire, etc.

Lambda

* Function as a service – short running and focused – piece of code
* Functions use a runtime, environment has direct memory allocation
* Billing based on function running duration
* Python, Ruby, Java, Go. C# and custom runtimes using layers
* 15 min. function timeout(hard limit)
* By default lambdas are public – no access to VPC services
* Private Lambda: running inside VPC with same networking rules
* Lambda resource policy controls what services can invoke lambda
* Lambda execution roles are IAM roles which control permissions the lambda has
* CloudWatch, CloudWatch Logs and X-Ray used for logging – requires permissions
* Synchronous invocation:
  + Result(Success or Failure) returned during request
  + Errors or Retries have to be handled within client
* Asynchronous invocation:
  + Typically used when AWS services invoke lambda
  + Lambda needs to be idempotent reprocessing a result should have the same end state
  + Events can be sent to dead letter queues
  + Supports destinations(SQS,SNS, Lambda) where successful or failed events can be sent
* Event Source Mapping: Typically used on streams or queues
  + Permissions from lambda execution role used by event souce mapping to interact with event source
  + SQS Queues or SNS topics can be used for discarded failed event batches
* Lambda Versions: Lambda functions have immutable versions
  + Unpublished function can be changed and deployed
  + Publishing creates immutable version with unique, qualified ARN
  + $Latest can be used as unqualified ARN
* Aliases is pointer to a function version( like DEV, STAGE, PROD), can be changed
* Startup times:
  + Cold start is full creation and configuration including function code download
  + Warm start happens if lambda is called soon after cold start and reuses execution context
  + Provisioned concurrency can be used to keep contexts warm
* Lambda Handler:
  + Executions have lifecycles – Execution Environment
  + INIT Creates or Unfreezes execution environment
  + INVOKE runs the function handler
  + NEXT INVOKE – Warm start – same environment
  + SHUTDOWN terminates the environment
  + Reusable components(DB connections, Downloads, etc.) should be in init function of the code
* Environment Variables: Key Value pairs, associated with $LATEST or with a version(immutable)
  + Can be accessed within execution env
* Lambda Layers: Allow new runtimes & Libraries to be externalized 🡪 smaller deployment ZIPs with shared libraries between functions
* Lambda Container Images: AWS Lambda Runtime Interface Emulator for local testing
* Lambda can be used in combination with ALB

CloudWatch Events/EventBride

* If X happens, or Y times do Z
* Default Event bus for the account(only one in CloudWatch Events)
* Rules match incoming events, Route event to targets (Lambda, etc.)
* Event Pattern Rule or Scheduled Rule

API Gateway

* Create and manage APIs, Endpoint/entry-point for applications
* Sits between applications & integrations(services)
* HA, scalable, authorization handling, caching, throttling, CORS, Open API spec
* Can connect to endpoints in AWS or onprem
* HTTP, REST and WebSocket APIs
* Endpoint Types:
  + Edge Optimized: routed to nearest Cloudfront POP
  + Regional: Clients in same region
  + Private: Only accessible within VPC
* Stages: APIs are deployed to stages
  + Stages can be enabled for canary deployments
  + Each stage can have own stage variables
* Errors:
  + 4xx: Client Errors, 400 Bad request, 403 Access Denied, 429 throttling
  + 5xx: Server Errors, 502 Bad gateway, 503 service unavailable(bad lambda output), 504 integration failure/timeout(29s)
* Methods: desired action to be performet(POST, GET, PUT, etc.)
* Resources: points in the API tree (eg. /, /test)
* Integrations: API methonds are integrated with Backend endpoints
  + MOCK: Used for testing, no backend involved
  + HTTP: Backend HTTP Endpoint( with mapping template)
  + HTTP Proxy: pass through integration unmodified
  + AWS: API is exposing AWS services(with mapping template)
  + AWS\_PROXY: Lambda endpoint
  + Mapping templates: Modify or rename Parameters, Modify body or headers, Filtering
* Deployments: Changes in API GW are not live, API needs to be deployed to stage
* Swagger/OpenAPI: Standard for RESTful API’ss, Description format for REST API’s
  + Endpoints and Operations, Input and Output Parameters, Auth Methods

Step Functions

* Long running serverless workflows(Lambda with higher runtime)
* Based on State Machines: START 🡪 STATES 🡪 END
* Max duration 1 year
* Standard Workflow or Express Workflow
* States: SUCCEED, FAIL, WAIT, CHOICE(conditional), PARALLEL, MAP(actions on list), TASK(Lambda, Batch, ECS, SNS, SQS,…)

Simple Workflow Service(SWF)

* Build workflows
* Predecessor to Step Functions but uses Instances
* Activity Task and Activity Worker with Decider
* By default use always Step Functions
* SWF for AWS Flow Framework, External Signals in the process, Launch child flows, more complex decisions, Mechanical Turk

Mechanical Turk

* Managed human task outsourcing – extend app with humans
* Requesters post Human Intelligence Tasks to Marketplace, Humans do these tasks
* Use Human Workers over API

Elastic Transcoder & Elemental MediaConvert

* Serverless file-based video transcoding services
* File loaded from S3, processed and stored on S3
* ET is legacy(supports more legacy media formats, GIF, WebM, MP3) and MediaConvert is successor

IOT

* Suite of Products for managing IOT devices
* Provisioning, Updates & Control
* Provides Device Shadows(digital copies of real devices)

Greengrass

* Extends AWS services to the edge(compute, messaging, data management, sync and ML)
* Locally run Lambdas, Containers, Local IOT Device Shadows, Local Messaging, Local Hardware Access for Lambda

Serverless Application Model(SAM)

* Serverless application consist of:
  + Front end Code & Assets – S3 & CloudFront
  + API Endpoint – API GW
  + Compute – Lambda
  + Database – DynamoDB
  + Event Sources, permissions and more
* AWS SAM template specification
* AWS SAM command line interface to build, test and deploy SAM apps
* SAM is like CloudFormation but points at local resources

**Caching, Delivery and Edge**

CloudFront

* Origin: Source location of content(s3 or Custom)
* Distribution: Configuration unit
* Edge Location: Local cache of data
* Regional Edge Cache: Larger version of Edge location
* CloudFront Behaviours: Settings for TTL, protocol, privacy, etc.
* TTL:
  + More frequent cache hits = lower origin load
  + Default TTL = 24h
  + Minimum and Maximum TTL can be set with Origin Header: Cache-Control max-age and Expires
* Invalidations:
  + Expires all caches for specified path
  + Performed on distribution and applies to all edge locations
  + Versioned file names to not have to use invalidation
* SSL:
  + Supported by default \*.cloudfront.net
  + Alternate Domain Names with Route53 and ACM
* SNI: TLS extension, allowing host to be included, old browsers don’t support it
* Origin Types:
  + S3 origin: designed to work directly with cloudfront
  + Custom origin
* Caching:
  + CACHE HIT = Matching object, delivered from cache
  + CACHE MISS = Matching object not cached
  + Caching works easy for static content (object name only)
  + For dynamic content next to object name also query string parameters, cookies and request headers can be evaluated
  + Forward only what the application needs
  + Cache based on what can change the objects
  + The more things are involved in caching the less efficient it is
* Origin Access Identities(OAI): Type of identity that can be associated with CF Distributions, that OAI can be used in S3 bucket policies
* Custom Headers and IP based firewall blocks can be used to secure custom origins
* Private Distribution requires signed URL or signed Cookie
  + Multiple Behaviours where each is either Public or Private
  + Legacy: Cloudfront Key created by root user added as Trusted Signer
  + New: Trusted Key groups
  + Signed URL provide access to one object
  + Signed Cookies provide access to group of objects
* Geo Restrictions: Restrict which areas can access cloudfront content(Whitelist or Blacklist countries)
  + CloudFront Geo Restriction vs 3rd party Geolocation(completely customizable)
* Field Level Encryption: allows CloudFront to encrypt certain sensitive data at the edge using a public/private key pair
* Lambda@Edge: allows cloudfront to run lambda function at CloudFront edge locations to modify traffic between the viewer and edge location
  + Layers not supported, only Node.js and Python
  + Use cases: A/B testing, Migration between origins, Different Objects based on device, Content by country

Elasticache

* Managed in-memory cache which provides a managed implementation of the redis or memcached engines 🡪 High performance, not persistent
* Cache data for Read heavy workloads with low latency
* Reduces database workloads(saving cost)
* Can be used to store session data
* Requires application code changes for the use of caching

**Migrations & Extensions**

6 Rs of Cloud Migration

* Rehosting: Lift and Shift
  + Reduce Admin overhead, Cost savings for low usage applications
  + No full advantage of cloud
* Replatforming: Lift and Shift with Optimization
  + DB 🡪 RDS, Load Balancers 🡪 ELB, etc.
  + Reduce Admin overhead, Cost savings, improved HA, effective backups
* Repurchasing: Move to something new
  + Use XaaS Product( Ms365, Salesforce, etc.)
* Refactoring/Re-architecting: Use Cloud native Services and architectures
  + Cheaper(pay for what you use), scalability, better HA
  + Initially expensive & time consuming
* Retire: Decommission application
* Retain: Keep as it is (Migration to expensive, Risk to high)

Virtual Machine Migrations

* Application Discovery Service: gathers information about on-prem infra
  + Agentless: Measure performance and resource usage
  + Agent: Network, Processes, Usage, Performance
  + Integration with AWS Migration Hub and Athena
* Application Migration Service: Migrate whole VM’s
  + Agentless with a connector
  + Incremental replication of live volumes 🡪 Creates AMIs
  + Integration with AWS Migration Hub

DMS

* Managed DB Migration Service
* Runs using replication instance with Source and Destination Endpoints
* Schema conversion Tool helps with converting to other DB engine
  + Not used when migrating between DBs of same type
* DMS can utilize Snowball for huge DBs

Storage Gateway

* Bridge between onprem storages and AWS
* For Migrations, Extensions, Storage Tiering, DR and Replacement of Backup systems
* Volume Stored Mode: Everything stored onprem
  + Great for full disk backups of servers
  + Assists with disaster recovery
  + Doesn’t improve Datacenter capacity (Main copy is stored on Gateway)
* Volume Cached Mode: All Data stored on AWS managed S3 and cached locally
  + Frequently Accessed Data locally available
  + Onprem Capacity is extended to AWS
* Tape Gateway (VTL Mode): Storage Gateway communicates with storage endpoint and stores on S3 VTL 🡪 Glacier Tape Shelf
* File Gateway (File Mode): Bridges onprem File storage and S3
  + Mount points available via NFS or SMB
  + Files stored into mount point are visible as S3 objects
  + Primary Data held in S3

Snow Family

* Move large amounts of Data in and out of AWS, physical storage
* Snowball: Device Ordered from AWS(50 or 80 TB)
  + Economical Range: 10TB to 10PB(multiple devices)
  + Multiple Devices to multiple premises
  + Only Storage
* Snowball Edge: Storage and Compute
  + Larger Capacity than Snowball
  + Storage Optimized(with EC2), Compute Optimized or Compute with GPU
  + Ideal for remote sites where data processing is needed
* Snowmobile: Portable DC on a Truck
  + Ideal for single location when 10+ PB is required
  + Not economical for multi-site

Datasync

* Data transfer service to and from AWS
* For Migrations, Data Processing Transfers, Archival, Cost Effective Storage or DC/BC
* Designed to work at huge scale, keeps metadata

**Infrastructure as Code(Cloudfromation)**

* Cloudformation Template (json, yml) contains logical resources and are used to create stacks
* Stacks create physical resources from the logical
* If template is changed, physical resources are changed
* If stack is deleted, physical resources are deleted
* Template Parameters: accept console/CLI/API input
  + Can have default, allowed values, min, max, patterns, etc.
* Pseudo Parameters: similar to template parameters but only provided by AWS based on environment when creating the stack
* Intrinsic Functions: built-in functions help manage your stacks
  + assign values to properties that are not available until runtime
* Mappings: matches a key to a corresponding set of named values
* Outputs: declares output values that you can import into other stacks
* Conditions: statements that define the circumstances under which entities are created or configured (evaluated to True or False)
* DependsOn: specify that the creation of a specific resource follows another
* WaitCondition: Wait for x number of success signals with timeout
* Creation Policy: Policy to define timeout and signals
* Cfn-signal: signals AWS CloudFormation to indicate whether Amazon EC2 instances have been successfully created or updated
* Nested Stacks: hierarchy of related templates to be combined to form a single product
  + Stack resource limit of 500 resources
  + Resources in a single stack share a lifecycle
  + Can’t easily reuse resources or reference other stacks
  + Whole templates can be reused in other stacks
  + Use when the stacks form part of one solution – lifecycle linked
* Cross-Stack References: allow one stack to reference another
  + Outputs in one stack reference logical resources or attributes in that stack
  + CFN Stacks are designed to be isolated and self-contained
  + Outputs normally not visible from other stacks
  + Outputs can be exported making them visible
  + Exports must have unique name in region
  + FN::Import Value can be used to reference
  + Use when Service-Oriented & different lifecycles and Stack reuse
* Stack Sets: Deploy CFN stacks across many accounts & regions
  + Containers in an admin account contain stack instances referencing stacks
  + Each stack = 1 Region in 1 account
  + Scenario: Enable AWS Config, Cross account IAM roles, etc.
* Deletion Policy: preserve or (in some cases) backup a resource when its stack is deleted
  + Only applies to delete not to replace
* Stack Roles: allow an IAM role to be passed into the stack via PassRole
  + CFN uses permission of logged in identity by default(you need correct opermissions)
  + With stack role logged in user only needs PassRole
* Cfn-init: include metadata on an Amazon EC2 instance for the cfn-init helper script
* Cfn-hup: Helper daemon that detects changes in resource metadata and runs user-specified actions when a change is detected
* Change sets: allow to preview how proposed changes to a stack might impact your running resources
* Custom Resources: enable to write custom provisioning logic in templates that AWS CloudFormation runs anytime on create, update or delete stacks
  + Integrate CFN with anything it doesn’t support

**Deployment & Management**

Service Code

* Provides end-user portal where products and portfolios can be deployed in a self-service way
* Defined by technical administrators

AWS CodeCommit

* AWS version of github

AWS CodePipeline

* CD tool, controls from source through build towards deployment
* Pipelines are built from Stages and contain Actions
* Artifacts can be loaded and generated
* State changes 🡪 EventBridge

AWS CodeBuild

* Fully managed CI service that compiles source code, runs tests, and produces software packages that are ready to deploy (like Jenkins)
* Customized via buildspec.yml
  + Needs to be at root of repository
  + Four main phases in the file
    - Install: installs packages
    - Pre\_build: sign in and install dependencies
    - Build: commands for build process
    - Post\_build: package things, push images, notifications
  + Environment variables
  + Artifacts: what stuff to put where
* Logs 🡪 S3 and cloudwatch

AWS CodeDeploy

* Code Deployment as a Service
* EC2(needs CodeDeploy agent), On-premises, Lambda, ECS
* Integrates with AWS services & AWS Code\* tools
* Appspec.yml or appspec.json
  + Manage Deployments: config + lifecycle event hooks
  + Files(EC2/onprem): Which files should be installed
  + Resources(ECS/Lambda): Name, Version, Target Version, Task Definition, Container Details
  + Permissions(EC2/onprem): Permissions for Files from File section
  + LifecycleHooks: ApplicationStop, DownloadBundle, BeforeInstall, Install, AfterInstall, AppliationStart, ValidateService

Elastic Beanstalk(EB)

* PaaS which can create and manage infrastructure for application code
* Developer Focused, Focus on code, low infrastructure overhead
* Built in languages, docker & custom platforms, Go, Java, Tomcat, Net, Node.js, PHP,…
* Single Container Docker:
  + EC2 with docker,
  + Needs dockerfile or dockerrun.aws.json or docker-compose.ml
* Multicontainer Docker:
  + Creates ECS Cluster
  + Needs dockerrun.aws.json in root directory of application bundle
* Preconfigured Docker for non natively supported applications
* Great for small development teams
* Databases outside of EB, DBs in an ENV are lost if ENV is deleted
  + Environments and RDS: RDS can be created within EB env, Different env = different RDS
* Deployment Policies: How application versions are deployed to environments
  + All at once, Rolling, Rolling with additional batch, Immutable, Traffic Splitting, Blue/Green
* .ebextensions: Customize EB environments
  + YAML or JSON inside application source bundle .ebextensions folder
* For HTTPS SSL cert needs to be applied directly to LB
* Environment Cloning: Create new env by cloning an existing one

OpsWorks

* Provides managed implementations of Puppet and Chef
* Puppet Enterprise: AWS Managed Puppet Master Server
* Chef Automate: AWS Managed Chef Servers
* OpsWorks: AWS Integrated Chef, No servers

AWS Systems Manager(SSM)

* View and control AWS and onprem infrastructure
* Agent based – pre installed on recent windows and linux AMI’s
* Manage Inventory, patch assets, run commands and manage desired state
* Securely connect to EC2
* Parameter store for configuration and secrets
* Run Command: allows for commands to be executed on managed instances at scale
  + No SSH/RDP Access Required
  + Command documents can be reused & have parameters
* Patch Manager: allows for the patching of windows or linux managed instances running in AWS or on-premises
  + For Linux explicitly define patches for distribution
  + Windows:
    - AWS-DefaultPatchBaseline and AWS-WindowsPredefinedPatchBaseline-OS: Critical and Security Updates
    - AWS-WindowsPredefinedPatchBaseline-OS-Applications: also patches MS APP Updates

**Advanced Security and Config Management**

AWS Guard Duty

* Automatic(AI/ML) threat detection service which reviews data from supported services
* Continuous security monitoring service

AWS Config

* Records the configuration of resources over time into configuration histories
* Auditing of changes, compliance with standards

Amazon Inspector

* Automated security assessment service that helps improve the security and compliance of applications (EC2, containers, network assessment)

KMS

* Regional public service to create, store and manage keys
* Symmetric and assymetric keys, encrypt and decrypt
* Keys never leave KMS, AWS Owned or Customer Owned
* Supports Multi-region Keys, Key Rotation, Backing Key and Aliases
* Every Key has Key Policies + IAM Policies

CloudHSM

* Like KMS but true Single Tenant Hardware Security Module
* AWS Provisioned but fully customer managed
* Fully FIPS 140-2 Level 3
* Industry Standard APIs
* No native AWS integration
* Offload SSL/TLS Processing for Web Servers, Transparent Data Encryption for Oracle DBs, Protect Private Keys for a CA

ACM

* Easily provision, manage and deploy public and private SSL/TLS certificates
* Generate or Import(you are responsible for renewal) Certificates
* Certificates can be deployed out to supported services
* Only Supported Services (EC2 not supported)
* Regional Service, certs cannot leave region and need to be in same region as service
* Global services such as Cloudfront use us-east-1 and need cert there

Parameter Store

* Storage and retrieval of parameters - string, stringlist or secure string
* Supports encryption, versioning and can be secured using IAM
* Integrates natively with many AWS services

Secrets Manager

* Manage secrets within AWS, like Parameter store
* Offers automatic Key rotation using Lambda
* Directly integrates with RDS

VPC Flow Logs

* Monitoring of traffic flow to and from interfaces within a VPC
* Only capture Metadata not content
* Can be added at VPC, Subnet or Interface level
* Can be stored in S3 or Cloudwatch Logs
* Not Real Time

Application Layer Firewall

* Capable of inspecting, filtering and even adjusting data up to Layer 7 of the OSI model
* Aware of Layer 7 protocol(HTTP)
* Can identify normal or abnormal requests Protocol specific attacks
* Encryption terminated on L7 Firewall and new Encrypted connection between FW and Backend
* Data can be inspected, blocked, replaced or tagged

WAF

* Helps protect your web applications or APIs against common web exploits and bots
* Allow, Count, Challenge, Captcha or Block traffic matching WEBACL rules
* Rule Groups contain rules, AWS Managed, Custom or Service Owned

AWS Shield

* Managed DDoS protection service
* Standard: protection at the perimeter, region, VPC or edge,
  + Common L3 or K4 layer attacks
* Advanced: Protects CF, R53, Global Accelerator, LBs anything with EIPs
  + Must be explicitely enabled in Shield Advanced
  + Cost protection for unmitigated attacks
  + Proactive Engagement & AWS Shield Response Team

**Disaster Recovery & Business Continuity**

Types of DR

* Backup & Restore: Primary Site constantly backing up data
  + Cheapest but longest time to restore
* Pilot Light: Adding minimal infrastructure for a Secondary site
* Warm Standby: Secondary Site like primary but scaled down
* Active/Active: Secondary Site like Primary

DR Storage

* EBS: Failure of AZ means failure of volume
* S3: regional resilience, with replication between regions
* EFS: regional resilient

DR Compute

* EC2: Failure of AZ failure of Instance, asg to use multiple AZs
* ECS: EC2 mode same as EC2, Fargate ENIs in a VPC
* Lambda: Falure of AZ 🡪 switching to different subnet

DR Database

* DB on EC2: same as EC2
* RDS: Using standby in different AZ, cross region replication available
* Aurora: Can have replicas in all AZs, global DB available
* DynamoDB: regional resilience, global tables available

DR Networking

* Subnet in one AZ
* VPC regional resilience
* Route 53 global service

Amazon Lex

* Building conversational interfaces into any application using voice and text(powering Alexa)
* Automatic speech recognition - speech to text, Natural Language understanding – intent
* Lambdas can be integrated

Amazon Connect

* Easy to use omnichannel cloud contact center(Call Center)

Kinesis Video Streams

* Stream video from connected devices(Cameras) to AWS for analytics, machine learning (ML), playback, and other processing

AWS glue

* Fully managed extract, transform, and load (ETL) service that makes it easy for customers to prepare and load their data for analytics

Device Farm

* Application testing service that lets you improve the quality of your web and mobile apps by testing them across an extensive range of desktop browsers and real mobile devices

Amazon Comprehend

* Natural-language processing (NLP) service that uses machine learning to uncover valuable insights and connections in text

Amazon Kendra

* Intelligent search service powered by machine learning

Amazon Polly

* Text to speech
* Uses SSML language

Amazon Rekognition

* Deep learning Image and Video Analysis
* Object/Person detection, face analysis and comparison, etc.

Amazon Textract

* Machine learning (ML) service that automatically extracts text, handwriting, and data from scanned documents

Amazon Transcribe

* Automatic speech recognition service that uses machine learning models to convert audio to text

Amazon translate

* Neural machine translation service that delivers fast, high-quality, affordable, and customizable language translation

Amazon Forecast

* Fully managed service that uses statistical and machine learning algorithms to deliver highly accurate time-series forecasts

Amazon Fraud Detector

* Fully managed fraud detection service that automates the detection of potentially fraudulent activities online

Amazon SageMaker

* Fully managed machine learning service.
* With SageMaker, data scientists and developers can quickly and easily build and train machine learning models, and then directly deploy them into a production-ready hosted environment
* S3 not for low-latency retrieval times 🡪 use DynamoDB with TTL to delete records after certain time
* Large Data stored DynamoDB + S3 otherwise just DynamoDB
* DynamoDB has fine grained access control with STS and web identity federation
* Macie only scans S3 not CodeCommit
* On prem tape gateway can’t be connected to Kinesis Video Stream directly 🡪 use AWS storage Gateway first
* Storage Gateway Stored can only store up to 512TB
* Always make sure ELB is using **alias** A record
* API Gateway also has API caching option
* SAM can be used in combination with CodeBuild, CodeDeploy for CI/CD
* Systems Manager Automation is designed to configure and manage instances with custom runbooks or pre-defined runbooks maintained by AWS
* Geo blocking with low latency 🡪 Cloudfront with Geo-restriction enabled
* Cloudfront offers origin failover with multiple origins when HTTP errors happen
* Use Nat-Gateways instead of Nat instances
* NACLs affect all subnets
* Direct connect with encrypted VPN 🡪 needs public VIF
* **Dedicated** network connection onprem to aws 🡪 direct connect
* SQS redrive policy with maxReceiveCount to retry messages from DLQ
* S3 better than EBS for scaling and ha
* Geolocation over geoporximity to send them to specified locations
* A diagram of a computer network

  Description automatically generated
* A diagram of a computer system

  Description automatically generated
* SCP only specifies services and actions not Resources
* AWSServiceRoleForOrganizations service-linked role is primarily used to only allow AWS Organizations to create service-linked roles for other AWS services
* AWS AppSync is not usable for push notifications
* S3 versioning: First version ID with value of null if was there before versioning
* ElastiCache Serverless is a robust cache solution that ensures high availability by automatically replicating data across multiple Availability Zones. Furthermore, it leverages read replicas for RDS to efficiently handle read-heavy workloads and utilizes Multi-AZ configurations to ensure high availability.
* Stateless instances > Stateful instances
* CHAP provides protection against playback attacks by requiring authentication to access storage volume targets for iSCSI
* PowerUser best for Developers
* cannot use the default certificate in CloudFront since the website is using a custom domain
* Glacier > Virtual Tape Shelf
* Redshift is primarily used for OLAP scenarios whereas RDS is used for OLTP scenarios
* FSx for Lustre filesystem can temporarily load the data from S3 and share it among the application tier instances
* AWS IoT Core supports several protocols, including MQTT without the need to provision or manage servers
* Multi-Attach EBS volumes can only be attached on instances within the same Availability Zone
* AWS Config can’t add tags by default
* Submitting comments, etc. 🡪 dynamic website 🡪 no CF
* snapshot copy grant for a master key needed with KMS
* Canary deployment configuration, the traffic is shifted in two increments
* DocumentDb does not offer a native cross-region replication or multi-region operation
* There is no option to Dynamically Allocate the file system size in Workspaces
* DataSync is an online data transfer service that simplifies, automates, and accelerates moving data between on-premises storage systems and AWS storage services
  + Doesn’t capture state and config of VMs
* VPC Flow logs can be in Parquet, which makes Redshift faster on them and saves cost
* RDS Proxy allows applications to pool and share connections established with the database
* Lambda function URL for less complexity than API GW
* AWS Compute Optimizer helps you identify the optimal AWS resource configurations, such as Amazon Elastic Compute Cloud (EC2) instance types, Amazon Elastic Block Store (EBS) volume configurations, task sizes of Amazon Elastic Container Service (ECS) services on AWS Fargate, and AWS Lambda function memory sizes, using machine learning to analyze historical utilization metrics
  + Only for AWS resources
* It is not safe to have a public-facing AWS Transfer family endpoint without any restrictions
* Global Accelerator is a service in which you create accelerators to improve the performance of your applications for local and global users
* API Gateway’s caching and traffic management capabilities may not be as efficient as CloudFront’s edge caching
* Amazon Inspector is not capable of scanning on-premises servers.
* You can’t directly connect two virtual private gateways from two different regions to a private VIF. You will need to have a Direct Connect gateway for this.
* You can create budget alerts specifically for each member account using budget filters
* S3 Replication Time Control (S3 RTC) helps you meet compliance or business requirements for data replication and provides visibility into Amazon S3 replication times
* A diagram of a computer

  Description automatically generated with medium confidence
* SESPort 587
* AWS Elastic Disaster Recovery:
  + When you need to protect critical databases, enterprise applications, and other on-premises or cloud-based workloads from disruptions.
  + When you want to achieve fast, reliable recovery with minimal downtime and data loss.
  + When you want to simplify the disaster recovery process and eliminate the need for specialized skills.