Table of Contents

4. Security and Disaster Recovery
    - Advanced IAM
    - Encryption
    - SSM Parameter Store
    - Secrets Manager
    - CloudHSM
    - Shield
    - WAF - Web Application Firewall
    - Shared Responsibility Model
    - Disaster Recovery
    - DMS - Database Migration Tool
    - AWS DataSync
5. Other Services
    - CICD - Continuous Integration / Continuous Delivery
    - CloudFormation
    - Containers in the Cloud
    - Data Analytics
    - Other Services (OpsWorks, Elastic Transcoder, Workspaces, Appsync)
    - More Solutions Architecture
6. Whitepapers
    - Well Architectured Framework
    - AWS Truster Advisor
    -

AWS Regions

- AWS Regions are clusters of data centers
- Example of regions: us-east-1, eu-west-1, etc.
- Most services provided by AWS are regions scoped. This means a data for a service used in one region is not replicated in another region

AWS Availability Zones (AZ)

- AWS Regions are divided into availability zones
- Each region has between 2 and 6 AZs. Usually they have 3
- Each AZ is one or more discrete data center with redundant power, networking an connectivity
- All AZs from a region are geographically separated from each other

- Even if they are separated, they have high bandwidth connection between them

EC2

- It mainly consists of the following capabilities:
    - Renting virtual machines in the cloud (EC2)
    - Storing data on virtual drives (EBS)
    - Distributing load across multiple machines (ELB)
    - Scaling the services using an auto-scaling group (ASG)

Introduction to Security Groups (SG)

- Security Groups are the fundamental of networking security in AWS
- They control how traffic is allowed into or out of EC2 machines
- Basically they are firewalls

Security Groups Deep Dive

- Security groups regulate:
    - Access to ports
    - Authorized IP ranges - IPv4 and IPv6
    - Control of inbound and outbound network traffic
- Security groups can be attached to multiple instances
- They are locked down to a region/VPC combination
- They do live outside of the EC2 instances - if traffic is blocked the EC2 instance wont be able to see it
- *It is good to maintain one separate security group for SSH access*
- If the request for the application times out, it is most likely a security group issue
- If for the request the response is a "connection refused" error, then it means that it is an application error and the traffic went through the security group
- By default all inbound traffic is blocked and all outbound traffic is authorized
- A security group can allow traffic from another security group. A security group can reference another security group, meaning that it is no need to reference the IP of the instance to which the security group is attached

Elastic IP

- When an EC2 instance is stopped and restarted, it may change its public IP address
- In case there is a need for a fixed IP for the instance, Elastic IP is the solution
- An Elastic IP is a public IP the user owns as long as the IP is not deleted by the owner
- With Elastic IP address, we can mask the failure of an instance by rapidly remapping the address to another instance
- AWS provides a limited number of 5 Elastic IPs (soft limit)
- Overall it is recommended to avoid using Elastic IP, because:
    - They often reflect pool arhcitectural decisions
    - Instead, us e a random public IP and register a DNS name to it

EC2 User Data

- It is possible to bootstrap (run commands for setup) an EC2 instance using EC2 User data script
- The user data script is only run once at the first start of the instance
- EC2 user data is used to automate boot tasks such as:
    - Installing update
    - Installing software
    - Downloading common files from the internet
    - Any other start-up task
- THe EC2 user data scripts run with root user privileges

EC2 Instance Launch Types

- On Demand Instances: short workload, predictable pricing
- Reserved: known amount of time (minimum 1 year). Types of reserved instances:
    - Reserved Instances: recommended long workloads
    - Convertible Reserved Instances: recommended for long workloads with flexible instance types
    - Scheduled Reserved Instances: instances reserved for a longer period used at a certain schedule
- Spot Instances: for short workloads, they are cheap, but there is a risk of losing the instance while running
- Dedicated Instances: no other customer will share the underlying hardware
- Dedicated Hosts: book an entire physical server, can control the placement of the instance

EC2 On Demand

- Pay for what we use, billing is done per second after the first minute
- Hast the higher cost but it does not require upfront payment
- Recommended for short-term and uninterrupted workloads, when we can't predict how the application will behave

EC2 Reserved Instances

- Up to 75% discount compared to On-demand
- Pay upfront for a given time, implies long term commitment
- Reserved period can be 1 or 3 years
- We can reserve a specific instance type
- Recommended for steady state usage applications (example: database)
- Convertible Reserved Instances:
    - The instance type can be changed
    - Up to 54% discount
- Scheduled Reserved Instances:
    - The instance can be launched within a time window
    - It is recommended when is required for an instance to run at certain times of the day/week/month

EC2 Spot Instances

- We can get up to 90% discount compared to on-demand instances
- It is recommended for workloads which are resilient to failure since the instance can be stopped by the AWS if our max price is less then the current spot price
- Not recommended for critical jobs or databases
- Great combination: reserved instances for baseline performance + on-demand and spot instances for peak times

EC2 Dedicated Hosts

- Physical dedicated EC2 server
- Provides full control of the EC2 instance placement
- It provides visibility to the underlying sockets/physical cores of the hardware
- It requires a 3 year period reservation
- Useful for software that have complicated licensing models or for companies that have strong regulatory compliance needs

EC2 Dedicated Instances

- Instances running on hardware that is dedicated to a single account
- Instances may share hardware with other instances from the same account
- No control over instance placement
- Gives per instance billing

EC2 Spot Instances - Deep Dive

- With a spot instance we can get a discount up to 90%
- We define a max spot price and get the instance if the current spot price < max spot price
- The hourly spot price varies based on offer and capacity
- If the current spot price goes over the selected max spot price we can choose to stop or terminate the instance within the next 2 minutes
- Spot Block: block a spot instance during a specified time frame (1 to 6 hours) without interruptions. In rare situations an instance may be reclaimed
- Spot request - with a spot request we define:
    - Maximum price
    - Desired number of instances
    - Launch specifications
    - Request type:
        - One time request: as soon as the spot request is fulfilled the instances will be launched an the request will go away
        - Persistence request: we want the desired number of instances to be valid as long as the spot request is active. In case the spot instances are reclaimed, the spot request will try to restart the instances as soon as the price goes down
- Cancel a spot instance: we can cancel spot instance requests if it is in open, active or disabled state (not failed, canceled, closed)
- Canceling a spot request does not terminate the launched instances. If we want to terminate a spot instance for good, first we have to cancel the spot request and the we can terminate the associated instances, otherwise the spot request may relaunch them

Spot Fleet

- Spot Fleet is a set of spot instances and optional on-demand instances
- The spot fleet will try to meet the target capacity with price constraints

- AWS will launch instances from a launch pool, meaning we have to define the instance type, OS, AZ for a launch pool
- We can have multiple launch pools from within the best one is chosen
- If a spot a fleet reaches capacity or max cost, no more new instances are launched
- Strategies to allocate spot instances in a spot fleet:
    - lowerPrice: the instances will be launched from the pool with the lowest price
    - diversified: launched instances will be distributed from all the defined pools
    - capacityOptimized: launch with the optimal capacity based on the number of instances

EC2 Instance Types

- R: applications that needs a lot of RAM - in-memory cache
- C: applications that need good CPU - compute/database
- M: applications that are balanced - general / web app
- I: applications that need good local I/O - databases
- G: applications that need GPU - video rendering / ML
- T2/T3 - burstable instances
- T2/T3 unlimited: unlimited burst

Bustable Instances (T2/T3)

- Overall the performance of the instance is OK
- When the machine needs to process something unexpected (a spike load), it can burst and CPU can be very performant
- If the machine bursts, it utilizes "burst credits"
- If all the credits are gone, the CPU becomes bad
- If the machine stops bursting, credits are accumulated over time
- Credit usage / credit balance of a burstable instance can be seen in CloudWatch
- CPU credits: bigger the instance the faster credit is earned
- T2/T3 Unlimited: extra money can be payed in case the burst credits are used. There wont be any performance loss

AMI

- AWS comes with lots of base images
- Images can be customized ar runtime with EC2 User data
- In case of more granular customization AWS allows creating own images - this is called an AMI
- Advantages of a custom AMI:
    - Pre-install packages
    - Faster boot time (on need for the instance to execute the scripts from the user data)
    - Machine configured with monitoring/enterprise software
    - Security concerns - control over the machines in the network
    - Control over maintenance
    - Active Directory out of the box
- An AMI is built for a specific region (NOT GLOBAL!)

Public AMI

- We can leverage AMIs from other people
- We can also pay for other people's AMI by the hour, basically renting the AMI form the AWS Marketplace
- Warning: do not use AMI which is not trustworthy!

AMI Storage

- An AMI takes space and they are stored in S3
- AMIs by default are private and locker for account/region
- We can make our AMIs public and share them with other people or sell them on the Marketplace

Cross Account AMI Sharing

- It is possible the share AMI with another AWS account
- Sharing an AMI does not affect the ownership of the AMI
- If a shared AMI is copied, than the account who did the copy becomes the owner
- To copy an AMI that was shared from another account, the owner of the source AMI must grant read permissions for the storage that backs the AMI, either the associated EBS snapshot or an associated S3 bucket
- Limits:
    - An encrypted AMI can not be copied. Instead, if the underlying snapshot and encryption key where shared, we can copy the snapshot while

re-encrypting it with a key of our own. The copied snapshot can be registered as a new AMI
    - We cant copy an AMI with an associated billingProduct code that was shared with us from another account. This includes Windows AMIs and AMIs from the AWS Marketplace. To copy a shared AMI with billingProduct code, we have to launch an EC2 instance from our account using the shared AMI and then create an AMI from source

Placement Groups

- Sometimes we want to control how the EC2 instances are placed in the AWS infrastructure
- When we create a placement group, we can specify one of the following placement strategies:
    - Cluster - cluster instances into a low-latency group in a single AZ
    - Spread - spread instances across underlying hardware (max 7 instances per group per AZ)
    - Partition - spread instances across many different partitions (which rely on different sets of racks) within an AZ. Scale to 100s of EC2 instances per group (Hadoop, Cassandra, Kafka)

Placement Groups - Cluster

- Pros: Great network (10Gbps bandwidth between instances)
- Cons: if the rack fails, all instances fail at the time
- Use cases:
    - Big data job that needs to complete fast
    - Application that needs extremely low latency and high network throughput

Placement Groups - Spread

- Pros:
    - Can span across multiple AZs
    - Reduces risk for simultaneous failure
    - EC2 instances are on different hardware
- Cons:
    - Limited to 7 instances per AZ per placement group
- Use case:
    - Application that needs to maximize high availability

    - Critical applications where each instance must be isolated from failure

Placement Groups - Partitions

- Pros:
    - Up to 7 partitions per AZ
    - Can have hundreds of EC2 instances per AZ
    - The instances in a partition do not share racks with the instances from other partitions
    - A partition failure can effect many instances but they wont affect other partitions
    - Instances get access to the partition information as metadata
- Use cases: HDFS, HBase, Cassandra, Kafka

Elastic Network Interfaces - ENI

- Logical component in a VPC that represents a virtual network card
- An ENI can have the following attributes:
    - Primary private IPv4 address, one or more secondary IPv4 addresses
    - One Elastic IP (IPv4) per private IPv4
    - One Public IPv4
- ENI instances can be created independently from an EC2 instance
- We can attach them on the fly to an EC2 instances or move them from one to another (useful for failover)
- ENIs are bound to a specific available zone
- ENIs can have security group attached to them
- EC2 instances usually have a primary ENI (eth0). In case we attach a secondary ENI, eth1 interface will be available. The primary ENI can not be detached.

EC2 Hibernate

- We can stop or terminate EC2 instances:
    - If an instance is stopped: the data on the disk (EBS) is kept intact
    - If an instance is terminated: any root EBS volume will also gets destroyed
- On start, the following happens in case of an EC2 instance:
    - Fist start: the OS boots and EC2 User data script is executed
    - Following starts: the OS boots

    - After the OS boot the applications start, cache gets warmed up, etc. which may take some time
- EC2 Hibernate:
    - All the data from RAM is preserved on shut-down
    - The instance boot is faster
    - Under the hood: the RAM state is written to a file in the root EBS volume
    - The root EBS volume must be encrypted
- Supported instance types for hibernate: C3, C4, C5, M3, M4, M5, R3, R4, R5
- Supported OS types: Amazon Linux 1 and 2, Windows
- Instance RAM size: must be less then 150 GB
- Bare metal instances do not support hibernate
- Root volume: must be EBS, encrypted, not instance store. And it must be large enough
- Hibernate is available for on-demand and reserved instances
- An instance can not hibernate for more than 60 days

EC2 for Solution Architects

- EC2 instances are billed by the second, t2.micro is free tier
- On Linux/Mac we can use SSH, on Windows Putty or SSH
- SSH is using port 22, the security group must allow our IP to be able to connect
- In cas of a timeout, it is most likely a security group issue
- Permission for SSH key => chmod 0400
- Security groups can reference other security groups instead of IP addresses
- EC2 instance can be customized at boot using EC2 User Data
- 4 EC2 launch modes:
    - On-demand
    - Reserved
    - Spot
    - Dedicated hosts
- We can create AMIs to pre-install software
- An AMI can be copied through accounts and regions
- EC2 instances can be started in placement groups:
    - Cluster
    - Spread
    - Partition

Elastic Load Balancers

Scalability and High Availability

- Scalability means that an a system can handle greater loads by adapting
- We can distinguish two types of scalability strategies:
    - Vertical Scalability (scale up/down)
        - Increase the size of the current instance (ex. from a t2.micro instance migrate to a a t2.large one)
        - Vertical scalability is common for non distributed systems, such as databases
        - RDS, ElastiCache are services that can scale vertically
    - Horizontal Scalability (scale out/in)
        - Increase the number of instances on which the application runs
        - Horizontal scaling implies having a distributed system
        - It's easy to scale horizontally thanks to could offerings such as EC2
- High Availability means running our application in at least 2 data centers (AZs)
    - The goal of high availability is to survive a data center loss
- High Availability can be:
    - Passive
    - Active
- Load balancers can scale but not instantaneously - contact AWS for a "warm-up"
- Troubleshooting:
    - 4xx errors are client induced errors
    - 5xx errors are application induced errors (server side errors)
    - Error 503 means that the load balancer is at capacity or no registered targets can be found
    - If the load balancer can't connect to the application, it most likely means that the security group blocks the connection
- Monitoring:
    - ELB access logs will log all the access requests to the LB
    - CloudWatch Metrics will give aggregate statistics (example: connections counts)

Load Balancing Basics

- Load balancers are servers that forward internet traffic to multiple other servers (most likely EC2 instances)
- Why use load balances?
    - Spear load across multiple downstream instances
    - Expose a single point of access (DNS) to the application
    - Seamlessly handle failures of downstream instances (by using health checks)
    - Do regular health checks to registered instances
    - Provide SSL termination (HTTPS) for the website hosted on the downstream instances
    - Enforce stickiness for cookies
    - High availability across availability zones (load balancer can be spread across multiple AZs, not regions!!!)
    - Cleanly separate public traffic from private traffic
- An ELB (Elastic Load Balancer) is a managed load balancer which means:
    - AWS guarantees that it will be working
    - AWS takes care of upgrades, maintenance and high availability
    - An ELB provides a few configuration options for us also
    - It costs less to setup our custom load balancer, but it will be a lot more effort to maintain on the long run
    - An ELB is integrated with many AWS offering/services, it will be more flexible than a custom LB

Health Checks

- They enable for a LB to know if an instance for which traffic is forwarded is available to reply to requests
- The health checks is done using a port and a route (usually /health)
- If the response is not 200, then the instance is considered unhealthy

Types of Load Balancers on AWS

- AWS provides 4 type of load balancers:
    - Classic Load Balancer (v1 - old generation): supports HTTP, HTTPS and TCP
    - Application Load Balancer (v2 - new generation): supports HTTP, HTTPS and WebSockets
    - Network Load Balancer (v2 - new generation): supports TCP, TLS (secure TCP) and UDP

    - Gateway Load Balancer (new generation - see VPC section of the notes)
- It is recommended to use the new versions
- We can setup internal (private) and external (public) load balancers on AWS

Classic Load Balancers (CLB)

- They support 2 types of connections: TCP (layer 4) and HTTP(S) (layer 7)
- Health checks are either TCP or HTTP based
- CLBs provide a fixed hostname: XXX.region.elb.amazonaws.com

Application Load Balancers (ALB)

- They are a layer 7 type load balancers (only HTTP or HTTPS)
- They allow load balancing to multiple HTTP applications across multiple machines (target groups). Also they allow to load balance to multiple applications on the same EC2 instance (useful in case of containers)
- They have support for HTTP2 and WebSockets.
- They support redirects, example for HTTP to HTTPS
- They provide routing tables to different target groups:
    - Routing based on path in URL
    - Routing based on the hostname
    - Routing based on query strings an headers
- ALBs are great fit for micros-services and container based applications
- ALBs have port mapping features to redirect to dynamic ports in case ECS
- Target groups can contain:
    - EC2 instances (can be managed by an Auto Scaling Group)
    - ECS tasks (managed by ECS itself)
    - Lambda Functions - HTTP request is translated to a JSON event
    - IP Addresses - must be private IP addresses
- ALBs also provide a fixed hostname (same as CLBs): XXX.region.elb.amazonaws.com
- The application servers behind the LB can not see the IP of the client who accessing them directly, but they can retrieve for X-Forwarded-For header. The port can be fetched from X-Forwarded-Port and the protocol from X-Forwarded-Proto

Network Load Balancers (NLB)

- Network load balancers (layer 4) allow to:

    - Forward TCP and UDP traffic to the registered instances
    - Handle millions of requests per second
    - Less latency ~100ms (vs 400ms for ALB)
- NLBs have one static IP per AZ and supports Elastic IPs (can be used when whitelisting is necessary)
- Use case for NLBs: NLBs are used for extreme performance in case of TCP or UDP traffic (example: video games)
- Instances behind an NLB don't see traffic coming from the load balancer, they see traffic as it was coming from the outside world => no security group is attached to LB => security group attached to the target EC2 instance should be changed to allow traffic from the outside (example: 0.0.0.0/0, on port 80)

Stickiness

- It is possible to implement stickiness in case of CLB and ALB load balancers
- Stickiness means that the traffic from the same client will be forwarded to the same target instance
- Stickiness works by adding a cookie to the request which has an expiration date for controlling the stickiness period
- Possible use case for stickiness: we have to make sure that the user does not lose his session data
- Enabling stickiness may bring imbalance to the load over the downstream target instances

Cross-Zone Load Balancing

- With Cross-Zone Load Balancing enabled each LB instance distributes traffic evenly across multiple AZs
- Otherwise, ech LB node distributes requests evenly only in the AZ where it is registered
- Classic Load Balancer:
    - Cross-zone load balancing is disabled by default
    - No additional charges for cross zone load balancing if the feature is enabled
- Application Load Balancer:
    - Cross-zone load balancing is always on, can not be disabled
    - No charges applied for cross zone load balancing
- Network Load Balancer:
    - Cross-zone load balancing is disabled by default

    - Additional charges apply if the feature is enabled

SSL/TLS Certificates

- An SSL certificate allows traffic to be encrypted between the clients and the load balancers. This ia called encryption in transit or in-flight encryption
- SSL - Secure Socket Layer
- TLS (newer version of SSL) - Transport Layer Security
- Nowadays TLS are mainly used, but we are still referring to it as SSL
- Public SSL certificates are issued by a Certificate Authority
- SSL certificates have an expiration dates and they must be renewed
- SSL termination: client can talk with a LB using HTTPS but internal traffic can be routed to a target using HTTP
- Load balancer can load an X.509 certificate (which is a SSL/TLS server certificate)
- We can manage certificates in AWS using ACM (AWS Certificate Manager)
- HTTPS Listener:
    - We must specify a default certificate
    - We can add an optional list of certificates to support multiple domains
    - Clients can use SNI (Server Name Indication) to specify which hostname want to reach
    - Ability to specify a security policy to support older versions of SSL/TLS (for legacy clients like Internet Explorer 5 lol:))

SNI - Server Name Indication

- SNI solves the problem of being able to load multiple SSL certificates onto one web server
- There is a newer protocol which requires the client to indicate the hostname of the target server in the initial SSL handshake
    - In case of AWS this only works for ALB, NLB and CloudFront (no CLB!)

ELB - Connection Draining

- Feature naming:
    - In case of a CLB is called Connections Draining
    - If we have a target group: (ALB, NLB) it is called Deregistration Delay

- Connection draining is the time to complete in-flight requests while the instance is de-registering or unhealthy. Basically it allows the instance to terminate whatever it was doing
- The LB will stop sending new requests to the target instance which is in progress of de-registering
- The time period of the connection draining can be set between 1 seconds to 3600 seconds
- It also can be disabled (set the period to 0 seconds)

Auto Scaling Groups

- Applications may encounter different amount of load depending on their usage span
- In cloud we can create and get rid of resources (servers) quickly
- The goal of an Auto Scaling Group (ASG) is to:
    - Scale out (add more EC2 instances) to match the increased load
    - Scale in (remote EC2 instances) to match a decreased load
    - Ensure we have a minimum and a maximum number of machines running
    - Automatically register new instances to a load balancer

ASG Attributes

- Launch configuration - consists of:
    - AMI + Instance Type
    - EC2 User Data
    - EBS Volumes
    - Security Groups
    - SSH Key Pair
- Min size, max size, initial capacity
- Network + subnets information
- Load balancer information
- Scaling policies - what will trigger a scale out/scale in

Auto Scaling Alarms

- It is possible to scale an ASG based on CloudWatch alarms

- An alarm monitors a metric (such as Average CPU)
- Metrics are computed for the overall ASG instances
- Based on alarms we can create:
    - Scale-out policies
    - Scale-in policies

Auto Scaling New Rules

- It is possible to define "better" auto scaling rules managed directly by EC2 instances, for example:
    - Target Average CPU Usage
    - Number of requests on teh ELB per instance
    - Average Network In/Out
- These rules are easier to set up and to reason about then the previous ones

Auto Scaling Based on Custom Metrics

- We can scale based on a custom metric (ex: number of connected users to the application)
- In order to do this we have to:
    i. Send a custom metric request to CloudWatch (PutMetric API)
    ii. Create a CloudWatch alarm to react to values of the metric
    iii. Use the CloudWatch alarm as a scaling policy for ASG

ASG Summary

- Scaling policies can be on CPU, Network, etc. and can even be based on custom metrics or based on a schedule
- ASGs can use launch configurations or launch templates (newer version)
    - Launch configurations allow to specify one instance type
    - Launch templates allow to use a spot fleet of instances
- To update an ASG, we must provide a new launch configuration/template. The underlying EC2 instances will be replaced over time
- IAM roles attached to an ASG will be assigned to the launched EC2 instances
- ASGs are free. We pay for the underlying resources being launched (EC2 instances, attached EBS volumes, etc.)
- Having instances under an ASG means that if they get terminated for any reason, the ASG will automatically create new ones as a replacement

- ASG can terminate instances marked as unhealthy by a load balancer and obviously replace them
- Health checks - we can have 2 types of health checks:
    - EC2 health checks - instances is recreated if the EC2 instance fails to respond to health checks
    - ELB health checks - instance is recreated if the ELB health checks fail, meaning that the application is down for whatever reason

ASG Scaling Policies

- Target Tracking Scaling
    - Most simple and easy to setup
    - Example: we want the average ASG CPU to stay around 40%
- Simple/Step Scaling
    - Example:
        - When a CloudWatch alarm is triggered (example average CPU > 70%), then add 2 units
        - When a CloudWatch alarm is triggered (example average CPU < 30%), then remove 1 unit
- Scheduled Actions
    - Can be used if we can anticipate scaling based on known usage patterns
    - Example: increase the min capacity to 10 at 5 PM on Fridays

Scaling Cool-downs

- The cool-down period helps to ensure that our ASG doesn't launch or terminate additional instances before the previous scaling activity takes effect
- In addition to default cool-down for ASG we can create cool-downs that apply to specific *simple scaling policy*
- A scaling-specific cool-down overrides the default cool-down period
- Common use case for scaling-specific cool-downs is when a scale-in policy terminates instances based in a criteria or metric. Because this policy terminates instances, an ASG needs less time to determine wether to terminate additional instances
- If the default cool-down period of 300 seconds is too long, we can reduce costs by applying a scaling-specific cool-down of 180 seconds for example
- If our application is scaling up and down multiple times each hour, we can modify the ASG cool-down timers and the CloudWatch alarm period that triggers the scale

Suspend and Resume Scaling Processes

- We can suspend and then resume one or more of the scaling processes for our ASG. This can be useful when we want to investigate a configuration problem or other issue with our web application and then make changes to our application, without invoking the scaling processes.
- We can manually move an instance from an ASG and put it in the standby state
- Instances in standby state are still managed by Auto Scaling, are charged as normal, and do not count towards available EC2 instance for workload/application use. Auto scaling does not perform health checks on instances in the standby state. Standby state can be used for performing updates/changes/troubleshooting etc. without health checks being performed or replacement instances being launched.

ASG for Solutions Architects

- ASG Default Termination Policy
    - Find the AZ which has to most number of instances
    - If there are multiple instances to choose from, delete the one with the oldest launch configuration
- Lifecycle Hooks
    - By default as soon as an instance is launched in an ASG, the instance goes in service
    - ASGs provide the ability to perform extra steps before the instance goes in service
    - Also, we have he ability to perform some actions before the instance is terminated
    - Lifecycle hooks diagram: https://docs.aws.amazon.com/autoscaling/ec2/userguide/lifecycle-hooks.html
- Launch Templates vs. Launch Configurations
    - Both allow to specify the AMI, the instance type, a key par, security groups and the other parameters that we use to launch EC2 instances (tags, user-data, etc.)
    - Launch Configurations are considered to be legacy:
        - They must be recreated every time
    - Launch Templates:
        - They can have multiple versions

        - They allow parameter subsets used for partial configuration for re-use and inheritance
        - We can provision both On-Demand and Spot instances (or a mix of two)
        - We can use the T2 unlimited burst feature
        - Recommended by AWS

EBS - Elastic Block Storage

- An EC2 instances loses its root volume when it is manually terminated
- Unexpected terminations may happen
- Sometimes we need a way to store instance data somewhere
- An EBS (Elastic Block Store) Volume is a network drive which can be attached to an EC2 instance
- It allows for the instances to persist date
- EBS is a network drive:
    - It uses the network to communicate with the instance, which can introduce latency
    - It can be detached from an EC2 and attached ot another
- EBS volumes are locked to an AZ
- To move a volume across, we need to create a snapshot
- EBS volumes have a provisioned capacity (size in GB and IOPS)
- Billing is done for all provisioned capacity even if the capacity is not fully used

EBS Volume Types

- EBS Volumes can have 4 types:
    - GP2 (SSD): general purpose SSD volume that balances price and performance
    - IO1 (SSD): highest performance SSD volume for mission-critical low-latency or high-throughput workloads
    - ST1 (HDD): low cost HDD volume designed for frequent access, throughput-intensive workloads
    - SC1 (HDD): for less frequently accessed data
- EBS Volumes are characterized in Size | Throughput | IOPS (I/O Operations per second)

- Only GP2 and IO1 can be used as boot volumes

EBS Volume Types - Deep Dive

GP2

- Recommended for most workloads
- Can be system boot volume
- Can be used for virtual desktops, low-latency applications, development and test environments
- Size can range from 1GiB to 16TiB
- Small GP2 volumes can burst IOPS to 3000
- Max IOPS is 16000
- We get 3 IOPS per GiB, which means at 5334GiB we are the max IOPS size

IO1

- Recommended for business critical applications which require sustained IOPS performance, or more than 16000 IOPS per volume
- Recommended for large database workloads
- Size can be between 4Gib and 16 TiB
- The maximum ratio of provisioned IOPS per requested volume size is 50:1
- Max IOPS for IO1/2 volumes is 64000 IOPS for instances built on Nitro System and 32000 for other type of instances

ST1

- Recommended for streaming workloads
- It has fast throughput at low price
- Can not be a root volume
- Size can be between 500Gib and 16TiB
- Max IOPS is 500
- Max throughput 500 MiB/Sec

SC1

- Throughput oriented storage for large volumes of data which is infrequently accessed
- Can not be a boot volume
- Max IOPS is 250, max throughput 250MiB/sec

Limits

- SSD, General Purpose – gp2 – Volume size 1 GiB – 16 TiB – Max IOPS/volume 16,000
- SSD, Provisioned IOPS – i01 – Volume size 4 GiB – 16 TiB – Max IOPS/volume 64,000 – HDD, Throughput Optimized – (st1) – Volume size 500 GiB – 16 TiB
    - Throughput measured in MB/s, and includes the ability to burst up to 250 MB/s per TB, with a baseline throughput of 40 MB/s per TB and a maximum throughput of 500 MB/s per volume
- HDD, Cold – (sc1) – Volume size 500 GiB – 16 TiB.
    - Lowest cost storage – cannot be a boot volume – These volumes can burst up to 80 MB/s per TB, with a baseline throughput of 12 MB/s per TB and a maximum throughput of 250 MB/s per volume: HDD, Magnetic – Standard – cheap, infrequently accessed storage – lowest cost storage that can be a boot volume

EBS Snapshots

- Snapshots are incremental - only the changed blocks are backed up
- EBS backups use IO and we should not run them while the application is handling a lot of traffic
- Snapshots are stored in S3 (we are not able to see them)
- It is not necessary to detach the volume to do a snapshot, but it is recommended
- An account can have up to 100k snapshots
- We can make an image (AMI) out of a snapshot, snapshots can be copied across AZs
- EBS volumes restored from snapshots need to be pre-warmed (using `fio` or `dd` commands to read the entire volume)
- Snapshots can be automated using Amazon Data Lifecycle Manager

EBS Migrations

- EBS Volumes are locked to a specific AZ

- To migrate it to a different AZ (or region) we have to do the following:
    - Create a snapshot from the volume
    - (optional) Copy the volume to a different region
    - Create a volume from the snapshot in the AZ of choice

EBS Encryption

- When we create an encrypted EBS volume, we get the following:
    i. Data at rest is encrypted inside the volume
    ii. All the data in flight moving between the instance and the volume is encrypted
    iii. All snapshots are encrypted
    iv. All volumes created from the snapshots will be encrypted
- Encryption and decryption are handled transparently by EBS system
- Encryption may have a minimal impact on latency
- EBS Encryption leverages keys from KMS (encryption algorithm is AES-256)
- Copying an unencrypted snapshot allows encryption
- Encrypt an unencrypted EBS volume:
    i. Create an EBS snapshot from the volume
    ii. Copy the snapshot an enable encryption on the process
    iii. Create a new EBS volume from the snapshot (the volume will be encrypted)
    iv. Attach the encrypted volume to an instance

EBS vs Instance Store

- Some instances do not come with a root EBS volume
- Instead, they come with an instance store (ephemeral storage)
- An instance store is a physically attached to the machine (EBS is a network drive)
- Pros of instance stores:
    - Better I/O performance
    - Good for buffer, cache, scratch data, temporary content
    - Data survives a reboot
- Cons of instance stores:
    - On stop or termination of the instance, the data from the instance store is lost
    - An instance store can not be resized
    - Backups of an instance store must be done manually by the user

- An instance store is:
    - A physical disk form the physical server where the EC2 instance runs
    - Very Hight IOPS disk
    - A disk up to 7.5 TiB, stripped to reach 30 TiB
    - A block storage (just like EBS)
    - Can not be increased in size
    - An ephemeral storage (risk of data loss if hardware fails)

EBS RAID Options

- EBS is already redundant storage (replicated within an AZ)
- If we want to increase IOPS of if we want to mirror an EBS volume we can mount EBS volumes in parallel RAID settings
- RAID is possible as long as the OS supports it
- Some RAID options are:
    - RAID 0
    - RAID 1
    - RAID 5, RAID 6 are not recommended for EBS
- RAID 0: used for increased performance. We can combine to or more volumes and what we get is the total number of disk space and I/O
    - If one of the disks fail, all the logical data is lost
    - Use cases:
        - Applications with lot of IOPS but without the need for fault-tolerance
        - A database with builtin replication
- RAID 1: used for increased fault-tolerance. Mirroring a volume to another.
    - If one the disks fails, the logical volume will still work
    - We have to send the data to two EBS volumes at the same time
    - Use cases:
        - Applications that need increased fault-tolerance
        - Applications which need to service disks

EFS - Elastic File System

- EFS is a managed NFS (network file system) that can be mounted on many EC2 instances
- EFS works with EC2 instances across multi AZs

- EFS is highly available, scalable, but also more expensive (3x GP2) than EBS
- EFS is pay per use
- Use cases: content management, web service, data sharing, Wordpress
- Uses NFSv4.1 protocol
- We can use security groups to control access to EFS volumes
- EFS is only compatible with Linux based AMIs (not Windows)

EFS Performance and Storage Classes

- EFS Scale
    - Thousands of concurrent NFS clients, 10 GB+ per second throughput
    - It can grow to petabyte scale NFS automatically
- Performance mode (can be set at EFS creation time)
    - General purpose (default): recommended for latency-sensitive use cases: web server, CMS, etc.
    - Max I/O - higher latency, throughput, highly parallel, recommended for big data, media processing
- Storage tiers: lifecycle manage feature
    - Standard: for frequently accessed files
    - Infrequent access (EFS-IA): there is a cost to retrieve files, lower price per storage

EBS vs EFS

- EBS volumes
    - Can be attached to only one instance at a time
    - Are locked at the AZ level
    - GP2: IO increases if the disk size increases
    - IO1: can increase IO independently
    - To migrate an EBS across AZ:
        - Take a snapshot
        - Restore the volume from the snapshot
    - Root EBS volumes get terminated by default in the EC2 instance is terminated (this feature can be disabled)
- EFS
    - Can be mounted to multiple instances across AZs via EFS mount targets
    - Available only for Linux instances
    - EFS has a higher price point than EBS
    - EFS is pay per second, we can leverage EFS-IA for cost saving

AWS RDS - Relational Database Service

- It is a managed database service for relational databases
- It allows us to create databases in the cloud that are managed by AWS
- RDS offerings provided by AWS:
    - PostreSQL
    - MySQL
    - MariaDB
    - Oracle
    - Microsoft SQL Server
    - Aurora
- Advantages of AWS RDS over deploying an relational database on EC2:
    - RDS is a managed service, meaning:
        - Automated provisioning, OS patching
        - Continuous backups and restore to specific timestamp (Point in Time Restore)
        - Monitoring dashboards
        - Read replicas
        - Multi AZ setup
        - Maintenance windows for upgrades
        - Scaling capability (vertical and horizontal)
        - Storage backed by EBS (GP2 or IO)
- Disadvantages:
    - No SSH into the instance which hosts the database

RDS Backups

- Backups are automatically enabled in RDS
- AWS RDS provides automated backups:
    - Daily fill backup of the database (during the maintenance window)
    - Transaction logs are backed-up by RDS every 5 minutes which provides the ability to do point in time restores
    - There is a 7 day retention for the backups which can be increased to 35 days
- DB Snapshots:
    - There are manually triggered backups by the users
    - Retention can be as long as the user wants

    - Helpful for retaining the state of the database for longer period of time

RDS Read Replicas

- Read replicas helps to scale the read operations
- We can create up to 5 read replicas
- These replicas can be within AZ, cross AZ or in different regions
- The data between the main database and the read replicas is replicated asynchronously => reads are eventually consistent
- Read replicas can be promoted into their own database
- Use case for read replicas:
    - Production database is up and running taking on normal load
    - There is a new feature for running some reporting for analytics which may cause slow downs and may overload the database
    - To fix this we can create read replicas for reporting
- Read replicas are used for SELECT operations (not INSERT, UPDATE, DELETE)
- Network cost for read replicas:
    - In AWS there is network cost if data goes from one AZ to another
    - In case of cross AZ replication, additional costs may incur because of network traffic
    - To reduce costs, we could have the read replicas in the same AZ

RDS Multi AZ (Disaster Recovery)

- RDS Multi AZ replication is done using synchronous replication
- In case of multi AZ configuration we get one DNS name
- In case of the main database goes down, the traffic is automatically re-routed to the failover database
- Multi AZ is not used for scaling
- The read replicas can be set up as Multi AZ for Disaster Recovery

RDS Security

Encryption

- AWS RDS provides rest encryption: possibility to encrypt the master and read replicas with AWS KMS - AES-256 encryption

    - Encryption has to be defined at the launch time
    - If the master is not encrypted, the read replicas cannot be encrypted
    - Transparent Data Encryption (TDE) is available for Oracle and SQL Server
- In-flight encryption: uses SSL certificates to encrypt data from client to RDS in flight
    - It is required SSL a trust certificate when connecting to database
    - To enforce SSL:
        - PostgeSQL: rds.force_ssl=1 in the AWS RDS Console (Parameter Groups)
        - MySQL: `GRANT USAGE ON *.* To 'user'@'%' REQUIRE SSL;`
- Encrypting RDS backups:
    - Snapshots of un-encrypted RDS databases are un-encrypted
    - Snapshots of encrypted RDS databases are encrypted
    - We can copy an un-encrypted snapshot into an encrypted one
- Encrypt an un-encrypted RDS database:
    - Create a snapshot
    - Copy the snapshot and enable encryption for the snapshot
    - Restore the database from the encrypted snapshot
    - Migrate application from the old database to the new one and delete the old database

Network Security and IAM

- Network security:
    - RDS databases are usually deployed within a private subnet
    - RDS security works by leveraging security groups (similar to EC2), they control who can communicate with the database instance
- Access management:
    - There are IAM policies which help control who can manage an AWS RDS database (through the RDS API)
    - Traditional username/password can be used to login into the database
    - IAM-based authentication can be used to login into MySQL and PostgreSQL
- IAM authentication:
    - IAM database authentication works with MySQL and PostgreSQL
    - We don't need a password to authenticate, just an authentication token obtained through IAM and RDS API calls
    - The token has a lifetime of 15 minutes
    - Benefits:

        - Network in/out must be encrypted using SSL
        - IAM is used to centrally manage users instead of DB credentials
        - We can manage IAM roles and EC2 instance profiles for easy integration

Security Summary

- Encryption at rest:
    - It is done only when the database is created
    - To encrypt an existing database, we have create a snapshot, copy it as encrypted, and create an encrypted database from the snapshot
- Our responsibility:
    - Check the ports/IP/security groups inbound rules
    - Take care of database user creation and permissions or manage them through IAM
    - Create a database with or without public access
    - Ensure parameter groups or DB is configured to only allow SSL connections
- AWS responsibility:
    - DB patching
    - Underlying OS patching and updates

Amazon Aurora

- Aurora is a proprietary technology from AWS (not open sourced)
- It provides support for both PostgreSQL and MySQL drivers in order to be able to connect to it
- Aurora is cloud optimized, it claims 5x performance over MySQL and over 3x performance over PostgreSQL
- Aurora storage automatically grows in increments from 10GB up to 64TB
- Aurora can have 15 replicas (while MySQL has only 5), replication process is faster
- Failover is Aurora is instantaneous
- Aurora costs more than classic RDS (around 20%)
- Aurora availability:
    - Stores 6 copies of the data across 3 AZs - only needs 4 copies for writes and 3 for reads

    - It has self healing properties with peer-to-peer replication
    - Storage is split across 100s of volumes
- In Aurora only one instance is used for writes (master instance)
- In case of master failure, the failover happens in less than 30 seconds
- Provides support for cross region replication

Aurora DB Cluster

- In Aurora we always have one writer endpoint
- We can have up to 15 read replicas
- Read replicas can be inside an auto scaling groups which provides the right read capacity all the time
- Read endpoint: basically does load balancing between read replicas
- Aurora endpoints: we can map each connection to the appropriate instance or group of instances based on an use case. For example, to perform DDL statements we can connect to whichever instance is the primary instance. To perform queries, we can connect to the reader endpoint, with Aurora automatically performing load-balancing among all the Aurora Replicas. For clusters with DB instances of different capacities or configurations, we can connect to custom endpoints associated with different subsets of DB instances
- Custom endpoints: provide load-balanced database connections based on criteria other than the read-only or read-write capability of the DB instances. For example, we might define a custom endpoint to connect to instances that use a particular AWS instance class or a particular DB parameter group. Then we might tell particular groups of users about this custom endpoint. For example, we might direct internal users to low-capacity instances for report generation or ad hoc (one-time) querying, and direct production traffic to high-capacity instances

Aurora Security

- It is similar to basic RDS security, because it is using the same engine under the hood
- Provides encryption at rest using KMS
- Provides automated backups, snapshots and replicas are also encrypted
- Encryption in flight is done using SSL
- Possibility to authenticate using IAM token
- The Aurora instance is protected with security groups (just like basic RDS)
- There is no way to SSH into an Aurora instance

Aurora Serverless

- Provides automated database instantiation and auto-scaling based on actual usage
- Recommended for infrequent, intermittent or unpredictable workloads
- No capacity planning is needed
- Users pay per seconds, can be cost-effective

Global Aurora

- There are two ways to have cross region replication:
    - Aurora Cross Region Read Replicas:
        - Useful for disaster recovery
        - Simple to put in place
    - Aurora Global Database (recommended):
        - We can specify one primary region for read/write
        - We can have up to 5 secondary regions (read-only)
        - Replication lag is bellow 1 second
        - We can have up to 16 read replicas per secondary region
        - In case we need to promote one region, the RTO (Recovery Time Objective) is bellow 1 minute

AWS ElastiCache

- ElastiCache is a managed cache offering which supports Redis and Memcached
- Caches are in-memory databases with high performance and low latency
- The role of a cache is to reduce load from a database by caching query results
- It also can help make an application stateless by storing the application state in memory
- Provides:
    - Write scaling using sharding
    - Read scaling using Read Replicas
    - Multi AZ with Failover Capability
- Since it is a managed solution, AWS takes care of OS maintenance, patching, optimization, setup, monitoring, failure recovery and backups

ElastiCache Solution Architecture

DB Cache

- Application queries the ElastiCache first. If no data is available for the query (cache miss), the application gets the data from RDS and stores it into the cache
- Caching helps relieve the load from the database
- Cache must have an invalidation strategy to make sure only the most current data is stored in the cache

User Session Store

- Uses logs into the application
- The application writes the session data to the cache
- The session data can be reused by other instance of he back-end

Redis vs Memcached

Redis	Memcached
Multi AZ with auto-failover	Multi-node for partitioning data (sharding)
Read replicas to scale reads and have high availability	Non persistent
Data durability and AOF persistance	No backup and restore
Backup and restore features	Multi-threaded architecture

ElastiCache Security

- All caches in ElastiCache:

    - Support SSL in flight encryption
    - Do not support IAM authentication
    - IAM policies on ElastiCache are only used for AWS API-level security
- Redis AUTH
    - We can set a password/token when we create a Redis cluster
    - This is an extra layer of security on top of the security groups
- Memcached
    - Supports SASL-based authentication

Caching Patterns

- Lazy loading: all the reads are cached, data can become stale in cache
- Write Through: adds/updates of data are cached when written to the database
- Session Store: store temporary session data in cache using TTL features

Amazon S3

- Amazon S3 is one of the main building blocks of AWS being advertised as "infinitely scaling" storage

S3 Buckets

- Amazon S3 allows to store objects (files) in "buckets"
- Buckets must have a globally unique name
- Buckets are defined at the region level
- Bucket naming conventions:
    - No uppercase
    - No underscore
    - 3-63 character long names
    - Name must no be an IP
    - Must start with a lowercase letter or number

S3 Objects

- In a bucket an object is a file
- Objects do have a key as an identifier, which is basically the full path of the file:

    - s3://my-bucket/my_file.txt
    - s3://my-bucket/my_folder/another_folder/my_file.txt
- The key is composed of *the prefix* + object name
    - s3://my-bucket/*my_folder/another_folder*/my_file.txt
- In S3 there is no concept of directories within the buckets, just keys with very long names that contain slashes ("/")
- Object values are the content of the body:
    - Max object size in S3 is 5TB
    - In case of a upload bigger than 5GB, we must use multi-part upload
- Each object can have metadata: list of text key/value pairs - system or user added
- Each object can have tags: unicode key/value par, useful for security, lifecycle. A bucket can have up to 10 tags
- If versioning is enabled each object has a version ID

S3 Versioning

- Files can have a version in S3, this should be enabled at the bucket level
- If a files is uploaded with the same key (same filename) the version of the file will be changed, the existing file wont be overridden, we will have both files available with different versions
- It is best practice to version the files, because:
    - The files will be protected against unintended deletes
    - The files can be rolled back to previous versions
- Notes:
    - Any file that is not versioned prior to enabling versioning will have the version "null"
    - Suspending versioning does not delete the previous versions of the file
- Deleting versioned files:
    - When deleting a versioned file adds a delete marker to the file, but the file wont be deleted
    - The file can be restored by deleting the delete marker
    - Deleting the delete marker and the file together is a permanent delete, meaning the file wont be able to be restored

S3 Security

Encryption at Rest

- AWS provides 4 methods of encryption for objects in S3
    - SSE-S3: encrypts S3 objects using keys handled and managed by AWS
    - SSE-KMS: leverage AWS Key Management Service to manage encryption keys
    - SSE-C: the encryption keys are managed by the user
    - Client Side Encryption
- SSE-S3:
    - Keys used for encryption is managed by Amazon S3
    - Objects are encrypted in the server side
    - It uses AES-256 encryption
    - In order to have SSE-S3 encryption clients must set a header, which is x-amz-server-side-encryption": "AES256"
- SSE-KMS:
    - Encryption keys are handled and managed by KMS
    - KMS allows the manage which keys will be used for the encryption, moreover it has audit trails in order to be able to see who was using the KMS key
    - Objects are encrypted in the server-side
    - In order to have SSE-S3 encryption clients must set a header, which is x-amz-server-side-encryption": "aws:kms"
- SSE-C:
    - Server side encryption using data keys provided by the user from the outside of AWS
    - Amazon S3 does not store the encryption key provided by the user
    - Data should be transmitted through HTTPS, because a key is sent the AWS
    - Encryption key must be provided in the header of the request for every request
    - When retrieving the object, the same encryption key must be provided in the header
- Client Side Encryption:
    - Data must be encrypted before sending it to S3
    - This is usually accomplished by using a third party encryption library, like Amazon S3 Encryption Client
    - The user is solely responsible for encryption-decryption
    - The keys and the encryption cycle is managed by the user

Encryption in transit (SSL/TLS)

- Amazon S3 exposes:

    - HTTP endpoint for non-encrypted data
    - HTTPS endpoint for encryption in flight which relies on SSL/TLS
- Most clients for S3 will use HTTPS by default
- In case of SSE-C encryption HTTPS in mandatory

S3 Security Overview

- User based security:
    - Accomplished by using IAM policies: specified which calls should be allowed for a specified sued from IAM console
- Resource based security:
    - Accomplished by using bucket policies which are bucket wide rules from the S3 console. These rules may allow cross account access to the bucket
    - We also have Object Access Control Lists (ACL) and Bucket Access Control Lists which allow finer grain control over the bucket Note: an IAM principle can access an S3 object if:
    - The user IAM permission allows it or the resource policy allows it
    - The is no explicit deny

S3 Bucket Policies

- Bucket policies are JSON based documents
- They can be applied to both buckets and objects in buckets
- The effect of a statement in the bucket policy can be either allow or deny
- The principle in the policy represents the account or the user for which the policy applies to
- Common use cases for S3 bucket policies:
    - Grant public access to the bucket
    - Force objects to be encrypted at the upload
    - Grant access to another account (cross account access)

Bucket Settings for Block Public Access

- Relatively new settings that was created to block public access to buckets and objects if the account has some restrictions:
- S3 provides 4 different kind of block public access settings:
    - new access control lists
    - any access control lists
    - new public bucket or access point policies

    - block public and cross-account access to buckets and objects through any public bucket or access point policies
- These settings were created to prevent company data leaks

S3 Other Security Features

- Networking:
    - S3 supports VPC Endpoints (for instances in VPC without internet access)
- Logging and Audit:
    - S3 Access Logs can be stored in other S3 buckets
    - API calls can be logged in AWS CloudTrail
- User Security:
    - MFA Delete can be required in versioned buckets in order to protect for accidental deletions
    - Pre-Signed URLs: ULRs that are valid only for a limited time

S3 Websites

- S3 can host static websites and have them accessible from the internet
- The website URL will be something like this:
    - `<bucket-name>.s3-website-<AWS-region>.amazonaws.com`
    - `<bucket-name>.s3-website.<AWS-region>.amazonaws.com`
- In case of `403` errors we have to make sure that the bucket policy allows public reads

CORS

- An origin is a scheme (protocol), host (domain) or port
- CORS: Cross Origin Resource Sharing
- CORS is a web browser based mechanism to allow requests to other origins while visiting the main one
- Same origin example: http://example.com/app1 and http://example.com/app2
- Different origins: http://example.com and http://otherexample.com
- The request wont be fulfilled unless the other origin allows for the request, using CORS headers (example: Access-Control-Allow-Origin, Access-Control-Allow-Method)

S3 CORS

- If a client does a cross-origin request on an S3 bucket, the correct CORS headers need to be enabled in order for the request to succeed
- Request can be allowed for a specified origin (by specifying the URL of the origin) or for all origins (by using *)

S3 Consistency Model

- S3 is strong read-after-write consistent

Route 53

- Route53 is a Managed DNS (Domain Name System)
- DNS is a collection of rules and records which helps clients understand how to reach a server through its domain name
- In AWS, the most common records are:
    - A: map a hostname to IPv4
    - AAAA: map a hostname tp IPv6
    - CNAME: map a hostname to another hostname
    - Alias: map a hostname to an AWS resource
- Route 53 can use:
    - public domain names we own
    - private domain name that can be resolved by EC2 instances inside our VPCs
- Route53 has advanced features such as:
    - Load balancing through DNS - also called client load balancing
    - Health checks
    - Routing policy: simple, failover, geolocation, latency, weighted, multi value
- We pay $0.5 per month per hosted zone

DNS Records TTL (Time to Live)

- TTL a is way for web browser adn clients to cache the response of the DNS query
- Reason for caching is not to overload the DNS
- The client will cache the DNS response for the duration of the TTL value (duration value is seconds)

- High TTL (24 hours):
    - Less traffic for DNS
    - Possibility of outdated records
- Low TTL (60 seconds):
    - More traffic on DNS
    - DNS records wont be outdated for longer periods which makes changing DNS records easier
- TTL is mandatory for each DNS record!

CNAME Records vs Alias Records

- AWS resources usually expose an AWS hostname
- CNAME (Canonical Name record):
    - Points a domain name to any another domain name (example: app.mydomain.com => other.domain.com)
    - CNAME records work only for non root domains! (example of a non-root domain: something.rootdomain.com)
- Alias records:
    - Point a hostname to an AWS resource (app.mydomain.com => something.amazonaws.com)
    - Alias records work for both root and non-root domain
    - They are free of charge
    - They can do health checks

Route53 Health Checks

- If an instance is unhealthy, Route53 does not send traffic to it
- An instance is unhealthy if if fails to respond to X (default 3) amount of requests in a row
- It becomes healthy again if passes X (default 3) health checks in a row
- Default health checks interval is 30s (Fast Health Checks: can be set to 10s - higher cost)
- About 15 health checkers will check the end-point's health
- Health check protocols: HTTP, TCP, HTTPS (no SSL verification)
- Can be integrated with CloudWatch

Routing Policies

Simple Routing Policy

- Used to redirect to a single source
- We can not attach health checks to a simple routing policy
- Simple routing policy can return multiple values to a client
- If multiple values are returned the client can chose a random value

Weighted Routing Policy

- Controls the percentage of the requests that goes to a specific end-point
- Helpful to test a certain percentage of traffic on a new application version
- Helpful to split traffic between two regions
- Can be associated with health checks

Latency Routing Policy

- Redirects to a server that has the least latency to the client
- Mainly used when latency is a priority for the clients
- Latency is evaluated in terms of user to designed AWS Region

Failover Routing Policy

- Requires a primary record and a secondary record (disaster recovery)
- If the primary record fails, traffic is routed to the secondary record
- Requires a health check in order to work, failover is detected based on health checks

Geolocation Routing Policy

- It is routing based on the user's location
- We can specify where the traffic should be routed if the user is requesting from an IP that originates from a country
- Requires a default policy (in case there is no match for a location)

Geoproximity Routing Policy

- It is also routing based on the user's location

- We can optionally choose to route more traffic or less to a given resource by specifying a value known as bias
- To use geoproximity routing, we must use Route 53 traffic flow
- We create geoproximity rules for our resources and specify one of the following values for each rule:
    - If we are using AWS resources, the AWS Region that you created the resource in
    - If we are using non-AWS resources, the latitude and longitude of the resource

Multi Value Routing Policy

- Can be used when we need to route traffic to multiple resources and we want to associate Route53 health checks to the records
- It will return up to 8 healthy records for each Multi Value query
- It is not a substitute for Elastic Load Balancer!

Route 53 as a Registrar

- A domain name registrar is an organization that manages the reservation of internet domain names. Some domain registrars are: GoDaddy, Google Domain and also Route 53
- It is possible to use a third party domain registrar with AWS. In order to use a domain bought from the third party, we have to do the following:
    i. Create a hosted zone in Route53
    ii. Update NS records on 3rd party website to use Route 53 name servers

Deep Dive

Advanced S3

S3 MFA-Delete

- To use MFA-Delete we have to enable versioning on the selected bucket
- MFA will be required when:
    - We want to permanently delete an object version
    - We want to suspend the versioning on the bucket
- MFA wont be required when:
    - We want to enable versioning
    - We want to list deleted versions
    - We want to add a delete marker to an object
- MFA-Delete can be enabled/disabled only by the owner of the bucket (root account)!
- MFA-Delete currently can only be enabled using the CLI

S3 Access Logs

- For audit purposes we would want to log all access to S3 buckets
- Any request made to S3, from any account, authorized or denied, will be logged into another S3 bucket
- The data can be analyzed by some data analysis tools or Amazon Athena

Warnings

- We should never set our logging bucket to be the monitored bucket! This may create a logging loop causing the bucket to grow exponentially

S3 Replication

- In order to be able enable replication:
    - We must enable versioning on the source and destination buckets
- There are 2 types of replication:
    - Cross Region Replication (CRR): bucket are in a different region
        - Used for: compliance, lower latency access, replication across accounts
    - Same Region Replication (SRR): buckets are in the same region
        - Used for: log aggregation, live replication between production and test accounts
- In both cases the buckets can be in separate accounts
- Copying between replica buckets happens asynchronously (it is very quick)

- In order to be ably to copy between replicas, an IAM permission has to be assigned to the source bucket

Replication Notes

- Only the new objects are replicated after the replication is activated (no retroactive replication)
- For DELETE operations:
    - For deletion without version ID, a delete marker is added to he object. Deletion is not replicated
    - For deletion with version ID, the object is deleted in the source bucket. Deletion is not replicated
- There is no replication chaining!

S3 Pre-singed URLs

- We can generate pre-signed URLs using the SDK and the CLI
- Pre-signed URLs have a default wait time of 3600 seconds. This can be changed with `--expires-in` argument
- Users given a pre-signed URL will inherit the permissions of the person who generated the ULR

S3 Storage Classes

- Amazon S3 Standard-General Purpose
- Amazon S3 Standard-Infrequent Access (IA)
- Amazon S3 One Zone-Infrequent Access
- Amazon S3 Intelligent Tiering
- Amazon Glacier
- Amazon Glacier Deep Archive
- AmazonS3 Reduced Redundancy Storage (deprecated)

S3 Standard - General Purpose

- High durability (13 nines SLA) of objects across multiple AZ
- SLA: if we store 10 million objects in S3, we can expect to loose on average a single file per 10K years
- 99.99% availability per year

- It can sustain 2 concurrent facility failures

S3 Standard - Infrequent Access

- Suitable for data that is less frequently accessed, but it should be retrieved quickly when it is needed
- Same durability as General Purpose, 99.9% availability
- Lower cost than General Purpose

S3 One Zone - Infrequent Access

- Same as Standard IA, but data is stored in a single AZ
- Same durability as Standard IA. Data can be lost if an AZ goes down
- 99.5% availability per year
- Lower cost than IA

S3 One Zone - Intelligent Tiering

- Automatically moves objects between two access tiers based on access patterns
- Has a small monthly monitoring fee
- Same durability as General Purpose, having availability of 99.9%

S3 Glacier

- Low cost object storage for archiving/backup data
- Data is retained for longer terms (10s of years)
- Alternative to on-premise magnetic tape
- Same durability as General Purpose
- Cost per storage per month is really low, but we pay for data retrieval as well
- Each item in Glacier is called an archive, archives are stored in Vaults
- Provides 3 retrieval options:
    - Expedited (1 to 5 minutes): costs $10
    - Standard (3 to 5 hours)
    - Bulk (5 to 12 hours)
- Minimum storage duration for Glacier is 90 days

S3 Glacier Deep Archive

- For very long term storage - cheaper than S3 Glacier
- Retrieval options:
    - Standard (12 hours)
    - Bulk (48 hours)
- Minimum storage duration is 180 days

S3 - Moving between storage classes

- We can transition objects between storage classes in order to save money
- General rules:
    - Infrequently access documents should be moved to STANDARD_ID
    - Objects that don't need real-time access should be moved to GLACIER or DEEP_ARCHIVE
- Moving objects can be done manually or can be done via a lifecycle configuration
- Transaction actions: they define when should objects be transitioned from one storage to another
- Expiration actions: configuration to delete objects after a given time.\
    - Can be used to delete old versions of files if versioning is enabled on the bucket
    - Can be used to clean-up incomplete multi-part uploads
- Rules can be applied for a certain prefix
- Rules can be created for certain object tags

S3 - Performance

- Amazon S3 automatically scales to high request rates, having latency of 100-200ms to get the first byte out of S3
- We can achieve:
    - 3500 PUT/COPY/POST/DELETE requests per second per prefix in a bucket
    - 5500 GET/HEAD requests per second per prefix in a bucket
- Prefix explained:
    - Example of a file in a bucket: `my-bucket/folder/subfolder/file`
    - Prefix in this case is: `/folder/subfolder/`
    - Request performance will apply to each prefix separately

S3 KMS Limitation

- S3 Performance can be affected by KMS limits
- If encryption is enabled using SSE-KMS, we get additional requests to KMS which will apply to our KMS quota
- KMS could throttle performance, as of today we can not request a quota increase for it

S3 Performance Optimizations

- Multi-Part upload: will split data into smaller chunks and it will upload them in parallel. It is recommended to be used for files bigger than 100MB, it is mandatory for files bigger than 5GB
- S3 Transfer Acceleration: it can increase the transfer speed in case of uploads by using an AWS edge location. Compatible with multi-part upload.
- S3 Byte-Range Fetches: can be used to speed up downloads by parallelizing GET requests. Can be used to retrieve only a part of the file

S3 Select and Glacier Select

- Can be used to retrieve less data using SQL queries to do server side filtering
- We can filter by rows and columns. SQL statements should be simple, we can not have joins
- The purpose of S3 Select is to use less network traffic

AWS Athena

- Serverless service to perform analytics directly against S3 files
- We can use SQL to query data from the files from S3
- It provides a JDBC/ODBC driver
- Pricing: we are charged per query amount of data scanned, we are billed for what are we using
- Supported file formats: csv, json, orc, Avro, Parquet. In the back-end it uses Presto query engine
- Athena uses cases: business intelligence, analytics, reporting, log analysis, etc.

S3 Object Lock and Glacier Vault Lock

- S3 Object Lock: implements WORM (Write Once Read Many Model) model, meaning that it guarantees that a file is only written once and it can not be deleted until the lock is removed
- Glacier Vault Lock: same WORM model is implemented, locket file can not be changed as long as the lock is active
- Helpful for compliance and data retention

AWS CLI, SDK, IAM Roles and Policies

- Performing tasks against AWS can be done in several ways
    - Using AWS CLI from a local machine
    - Using AWS CLI on an EC2 machine
    - Using the SDK from a local machine
    - Using the SDK from an EC2 instance
    - Using the AWS Instance Metadata Service for EC2

AWS CLI Configuration

- In order to use the CLI from a local machine, we must generate access keys. Access keys can be generated from AWS console IAM service
- Access keys are provided as .csv file and they can be downloaded only once
- Set up AWS CLI from terminal:

```
aws configure
```

- This command creates 2 files in `/.aws/config` folder: `config` and `credentials`
- A configuration can be invalidated by deleting the access keys or it can be inactivated

AWS CLI on EC2

BAD WAY - Don't do this

- Never ever put personal credentials on an EC2 instance!
- This means, never put secrets in when running `aws configure` command. Use this for setting some defaults, like region and output format

THE RIGHT WAY

- IAM Roles can be attached to EC2 instances
- IAM Roles can come with a policy authorizing exactly what the EC2 instance should be able to do
- This is the best practice on AWS and should be done every time!

IAM Roles and Policies

- Policies can be managed by AWS or custom managed by users
- AWS provides a huge set of managed policies, if these are not good enough the users can create their own
- Inline policies: policies that are added inline to a role, this make them impossible to add them to another role
- AWS Policy generator: https://awspolicygen.s3.amazonaws.com/policygen.html
- AWS Policy simulator: https://policysim.aws.amazon.com/

EC2 Instance Metadata

- It allows EC2 instance to "lear about themselves" without using an IAM Role
- The URL to get EC2 metadata information is http://169.254.169.254/latest/meta-data. THis URL only works from EC2 instances, since it is an internal IP
- We can retrieve the IAM Role name for the EC2 instance but we can not retrieve the IAM Policy
- Metadata = info about the EC2 instance
- Userdata = launch scripts on the EC2 instance initial startup

AWS SDK

- SDK = Software Development Kit
- Official SDKs are for:
    - Java
    - .NET
    - NodeJS
    - PHP
    - Python (Boto3)
    - Go

    - Ruby
    - C++
- AWS CLI uses Boto3 under the

AWS SDK Credentials Security

- It is recommended to use the default credential provider chain
- The default credential provider chain works seamlessly with:
    - AWS credentials at `~/.aws/credentials` (only on our computers or on premise)
    - Instance Profile Credentials using IAM Roles (for EC2 machines, etc.)
    - Environment variables (AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY)
- Never store AWS credentials in code!
- Best practice is to inherit credentials from the credential provider chain

Exponential Back-off

- Any API that fails because of too many requests needs to be retried with Exponential Back-off strategy
- These applies for rate limited APIs
- SDK usually implements exponential back-off out of the box for requests
- Exponential Back-off example:
    i. First API call fails. After failure we wait 1s and retry
    ii. Second API call fails. We wait 2s
    iii. Third API call fails. We wait 4s
    iv. Forth API call fails. We wait 8s
    v. Etc..until the request succeeds
- Exponential back-off ensures the API is not overloaded

Classic Architecture Discussion

- This section discusses basic architectural solutions combining technical offerings provided by AWS

Stateless Web Apps: WhatIsTheTime.com

- WhatIsTheTime.com: allows people to know what time it is
- No database required
- We want to start small and can accept downtime at the beginning
- We want to be able to fully scale vertically and horizontally with no downtime
- Considerations when building the architecture:
    - Public IP vs private IP in case of EC2 instances
    - Elastic IP vs Route53 vs Load Balancers
    - Route53 TTL A records vs Alias Records
    - Maintaining EC2 instance scaling manually vs using an ASG
    - Multi AZ deployments to have disaster recovery
    - ELB Heath Checks
    - Security Groups Rules
    - Reservation of capacity for cost saving when possible
    - We are considering the 5 pillars of a well architected application: cost, performance, reliability, security and operational excellence

Stateful Web Apps: MyClothes.com

- MyClothes.com: allows people to buy clothes online
- There is a shopping cart for every user
- The application can have hundreds of users using the app in the same time
- We need to scale, maintain horizontal scalability and keep the app as stateless as possible
- Users should not loose their shopping cart while navigating the site
- Users should have their details in a database
- Consideration when building the architecture:
    - ELB sticky sessions
    - Web clients for storing cookies and making the application stateless
    - Session id + session cache which can be ElastiCache or DynamoDB
    - ElastiCache can be used to cache data from RDS
    - Multi AZ for disaster recovery
    - RDS:
        - We can use it for store user data
        - We can have read replicas for scaling reads
        - We can enable Multi AZ for disaster recovery
- Tight security using security groups

Instantiating Applications Quickly

- When launching a full stack application, it can take time to:
    - Install the application
    - Insert initial (or recovery) data
    - Configure everything
    - Launch the application
- In order to solve this we can use:
    - For EC2 Instances:
        - Golden AMI: install the application, OS, dependencies etc. beforehand and launch the EC2 instance from the Golden AMI
        - Bootstrap using User Data: for dynamic configuration we can use User Data scripts
    - For RDS databases we can restore the database from a snapshot
    - For EBS Volumes we can also restore the data from a snapshot

Elastic Beanstalk

- Developers usually don't want to manage infrastructure, they want to deploy code (lol :)
- Elastic Beanstalk is a developer centric view of deploying applications on AWS
- It uses all the previous components: EC2, ASG, ELB, RDS, etc.
- These components can be configured and instantiated via configuration files
- Beanstalk is entirely free, we only pay for the underlying infrastructure
- Beanstalk is managed service, which means:
    - Instance configuration, OS is handled by AWS
    - Deployment strategies are configurable but they are performed by Beanstalk
- The application code is the developer's responsibility (obviously)
- There are 3 architecture models for Beanstalk:
    - Single Instance deployments, recommended for development
    - LB + ASG: great for production or pre-production
    - ASG only: great for non wen applications
- Beanstalk has 3 components:
    - Application
    - Application version: each deployment gets a version
    - Environments
- We deploy application versions to environments and can promote application versions to the next environment
- We can rollback to previous versions of the application

- We have full control over the lifecycle of the application
- Beanstalk has support for many applications:
    - Go
    - Java SE
    - Java with Tomcat
    - .Net on Windows Server with IIS
    - Node.js
    - PHP
    - Python
    - Ruby
    - Packer
    - Single Container Docker
    - Multi-container Docker
    - Pre-configured Docker
- There is an option for custom platform if the stacks above are not what we are looking for

AWS CloudFront

- CloudFront is a content delivery network (CDN)
- It improves read performance, content is cached at the edge locations (currently there are 216 edge locations globally)
- CloudFront also offers DDos protection, integration with Shield and integration with AWS WAF (Web Application Firewall)
- CloudFront allows to expose external HTTPS end-points and it also can talk to internal HTTPS back-ends

CloudFront Origins

- The location of the data which gets distributed by CloudFront can be in:
    - S3 bucket:
        - Recommended for distributing files and caching them at the edge locations
        - It offers enhanced security with CloudFront Origin Access Identity (OAI)
        - CloudFront can be used as an ingress for uploading files to S3
    - Custom Origin (HTTP) which could be the following:
        - Application Load Balancer

        - EC2 instance
        - S3 website (must be enabled the static website functionality on the bucket)
        - Any other HTTP back-end

CloudFront Geo Restriction

- CloudFront can restrict who can access the distribution based on geographic location
- It provides:
    - Whitelisting: allow users to access the content if they from countries which are on the approved list
    - Blacklisting: deny access for users from countries which are listen on the banned list
- The country is determined using 3rd party Geo-IP database

CloudFront vs S3 Cross Region Replication

- CloudFront:
    - Global Edge network
    - Files are cached for a time period (TTL)
    - Recommended fro static content that must be available everywhere in the world
- S3 Cross Region Replication:
    - Must be configured for each region for which we want replication
    - Files are updated in near real time
    - Read only
    - Recommended for dynamic content that needs to be available at low-latency in few regions

CloudFront Signed URL/Signed Cookies

- Used for distributing exclusive content to specific users
- When a signed cookie/URL is created, a policy needs to be attached, which should contain:
    - URL expiration date
    - IP Ranges which can access the data
    - Trusted signers (which AWS accounts can create a signed URL)

- Signed URL time to live:
    - For shared content (movie, music) we should make it short
    - Fro private content we can mae it last longer
- Signed URL: we can get access to individual files
- Signed Cookies: we can get access to multiple files

CloudFront Signed URL vs S3 Pre-Signed URL

- CloudFront Signed URL:
    - Allows access to a path, no matter the origin
    - It is an account wide key-pair, only the root account can manage it
    - It can filter by IP, path, date, expiration
    - We can leverage all the CloudFront caching features
- S3 Pre-Signed URL:
    - It issues requests as the person whe pre-signed the URL
    - It uses the IAM key of the signing IAM principal
    - It has a limited lifetime

AWS CloudWatch

AWS CloudWatch Metrics

- CloudWatch provides metrics for every service in AWS
- A Metric is a variable to monitor: CPUUtilization, NetworkIn, etc.
- Metrics in CloudWatch belong to namespaces
- A Dimension is an attribute of a metric, examples: instance id, environment name, etc.
- We can have up to 10 dimensions per metrics
- Metrics have timestamps
- We can create CloudWatch dashboards from metrics

EC2 Details Monitoring

- EC2 instance metrics are gathered every 5 minutes
- We can enable details metrics (for a cost) which will allow gathering every 1 minute

- We can use detailed monitoring if we want more prompt scale for ASG
- Free tier allows to have 10 details monitoring metrics
- EC2 memory usage by default is not pushed to CloudWatch, we should have a custom metric for it

CloudWatch Custom Metrics

- We have the possibility to send our own custom metrics to CloudWatch
- We can use dimensions (attributes) to segment our metrics
- Metrics resolution by default is 1 minute, but we can have higher resolutions up to 1 second for a higher cost
- We can send metrics by using the PutMetricsData API call
- In case of errors we should use exponential back-off

CloudWatch Dashboards

- Great way to setup dashboards for quick access to key metrics
- Dashboards are global
- Dashboards can include graphs from different regions
- We can change the time zone and time rage for each dashboard
- We can set up automatic refresh (10s, 1m, 2m, 5m, 15m)
- Pricing:
    - 3 dashboards (up to 50 metrics) for free
    - $3/dashboard/month

CloudWatch Logs

- Applications can send logs to CloudWatch using the SDK
- Also, CloudWatch can collects logs from:
    - Elastic Beanstalk: collection of logs from applications
    - ECS: collections of logs from containers
    - AWS Lambda: collection from functions
    - VPL Flow Logs
    - API Gateway
    - CloudTrail based on filter
    - CloudWatch log agents: from EC2 machines
    - Route53: logs for DNS queries
- CloudWatch logs can be saved to:

    - Batch exporting to S3 for archival
    - Stream logs to ElasticSearch cluster for further analytics
- Log storage architecture:
    - Log groups: arbitrary name, usually representing the name of an application
    - Log stream: instances within application/log files/containers
- We can define a log expiration policy: never expire, 30 days, etc.
- Using the AWS CLI we can tail logs
- To send logs to CloudWatch, we have to make sure the IAM permissions are correct
- Logs can be encrypted at group level using KMS

Log Metric Filter and Insights

- CloudWatch Logs can use filter expressions
    - For example, find a specific IP inside of a log
    - Metric filters can be used to trigger alarms
- CloudWatch Logs Insights: can be used to query logs and add queries to CloudWatch Dashboards

CloudWatch Agent

- By default no logs from EC2 machines will go to CloudWatch
- We need to run a CloudWatch agent on EC2 to push the log files to CloudWatch
- We have to make sure the IAM permissions are correct for the EC2 instance
- CloudWatch log agents can be installed to on-premise instances

CloudWatch Logs Agent and Unified Agent

- CloudWatch Logs Agent:
    - Old version of the agent
    - Can only send data to CloudWatch Logs
- CloudWatch Unified Agent:
    - Can collect additional system level metrics
    - Can collect logs and send them to CloudWatch logs
    - Can collect metrics
    - It can have centralized configuration using SSM Parameter Store

CloudWatch Unified Agent Metrics

- Metrics are collected from Linux Servers running on EC2 instances
- Can collect information from:
    - CPU (active, guest, idle, system, user, steal)
    - Disk metrics (free space, used, total)
    - Disk IO (reads, writes, bytes, iops)
    - RAM (free, inactive, used, total, cached)
    - Netstat (number of TCP and UDP connections, net packages)
    - Processes (total, dead, blocked, idle, running, sleep)
    - Swap Space
- Out of the box metrics for EC2 - disk, CPU, network, for more granularity use CloudWatch Unified Agent

CloudWatch Alarms

- Alarms are used to trigger notifications for any metric
- Alarms can go to Auto Scaling, EC2 Actions, SNS notifications
- There are various options for alarm metrics: sampling, percentage, max, min, etc.
- Alarm states:
    - OK
    - INSUFFICIENT_DATA
    - ALARM
- Period:
    - Length of time in seconds to evaluate the metric
    - In case we are using high resolution custom metrics, we can chose between 10 or 30 seconds for firing the alarm

EC2 Instance Recovery

- Status Checks:
    - Instance status = check the EC2 VM
    - System check = check the underlying hardware
- If one of these alarms are triggered, we can have an action called Instance Recovery. This will trigger some internal mechanism in AWS to recover the instance
- After an instance recovery we will have the same private, public, elastic IP, same metadata and placement group

- Any data stored on an instance store will not be kept

AWS CloudWatch Events

- CloudWatch events can be:
    - Scheduled: cron job
    - Event pattern: event rules to react to a service doing something
- CloudWatch events can trigger a Lambda function, or can send SQS/SNS/Kinesis messages
- A CloudWatch event creates a small JSON document to give information about the change

AWS CloudTrail

- Provides governance, compliance and audit for an AWS account
- CloudTrail is enabled by default
- CloudTrail provides a history of events/API calls made within an AWS account from:
    - AWS Console
    - SDK
    - CLI
    - AWS services
- Logs from CloudTrail can be put into CloudWatch Logs
- If a resource is deleted in AWS, CloudTrail should contain trace of the operation
- CloudTrail records account activity and service events from most AWS services and logs the following records:
    - The identity of the API caller
    - The time of the API call
    - The source IP address of the API caller
    - The request parameters
    - The response elements returned by the AWS service -Trails can be configured to log data events and management events:
    - Data events: These events provide insight into the resource operations performed on or within a resource. These are also known as data plane operations
    - Management events: Management events provide insight into management operations that are performed on resources in your AWS

account. These are also known as control plane operations. Management events can also include non-API events that occur in the account

AWS Config

- Helps with auditing and recording compliance of AWS resources
- Helps record configurations and changes over time
- Provides the ability of storing the configuration data into S3 where it can be analyzed by Athena
- Problems AWS Config Solves:
    - Check if there is unrestricted SSH access to a security group
    - Check if buckets have public access
    - Find out how did an ALB configuration change over time
- We can receive alerts (SNS notifications) for any change
- AWS Config is a per region service, but it can be aggregated across regions and accounts

AWS Config Resource

- Ability to view the compliance of a resource over time
- Ability to view configuration of a resource over time
- View CloudTrails API calls if enabled

AWS Config Rules

- AWS provides a set of managed config rules (over 75) which can be used by the users
- Users can also make custom config rules (a rule must be defined using AWS Lambda)
- Example of custom rules user can make:
    - Evaluate if each EBS disk is of type GP2
    - Evaluate if each EC2 instance is of type t2.micro
- Rules can be evaluated/triggered:
    - For each config change
    - At regular time intervals
- Evaluation of rules can trigger CloudWatch events if the rule is non-compliant

- Rules can have auto remediations: if a resource is not compliant, the is an option to trigger auto remediation, example: stop instances with non-approved tags
- AWS Config Rules do not prevent actions from happening (no deny)

CloudWatch vs CloudTrail vs Config

- CloudWatch
    - Performance monitoring (metrics, CPU, network, etc.) and dashboards
    - Events and alerting
    - Log aggregation and analysis
- CloudTrail
    - Record API calls made within an account
    - Define trails for specific resources
    - It is a global service
- Config
    - Record configuration changes
    - Evaluate resources against compliance rules
    - Get timeline of changes and compliance

Global Accelerator

Unicast IP vs Anycast IP

- Unicast IP: one server holds one IP address
- Anycast IP: more than one servers are holding the same IP address, the client will be routed to the nearest one

AWS Global Accelerator

- It leverages the AWS internal network to route traffic to our application
- 2 Anycast IPs are created for our application, which are global. The traffic from these IPs is sent directly to Edge Locations which will send it to the application using the AWS global network
- Global Accelerator works with Elastic IP, EC2 instances, ALB, NLB which can either be public or private

- It provides consistent performance by:
    - Intelligent routing to the lowest latency and fast regional failover
    - Not having to deal with client cache (because the IP does not change)
    - It is internal to the AWS network
- Health checks
    - Global Accelerator performs health checks to our applications
    - Failover is less than 1 minutes in case of unhealthy application instance
    - Great for disaster recovery
- Security
    - It provides 2 external IPs which may be needed to be whitelisted
    - DDoS protection thanks to AWS Shield

AWS Global Accelerator vs CloudFront

- The both use the same AWS global network and its edge locations around the world
- Both services integrate with AWS Shield for DDoS protection
- CloudFront:
    - Improves performance for both cacheable content (such as images and videos) and dynamic content (such as API acceleration and dynamic site delivery)
    - Content is served from the edge locations
- Global Accelerator
    - Improves performance for a wide range of applications over TCP or UDP
    - Packages to the application are proxied from the edge locations
    - Good fit for non-HTTP applications such as gaming (UDP), IoT (MQTT) or Voice over IP
    - Good fir tor HTTP in case if it is required to have static IP addresses or deterministic and fast regional failover

Snowball

- It is a physical data transport solution which helps moving terra bytes or peta bytes of data in our out of AWS
- Alternative to move data over network (in case of huge amount of data)
- It is secure, temper resistant, it uses KMS 256 bit encryption
- It has tracking using SNS and text messages. It has an E-ink shipping label
- Uses pay for data transfer jobs

- Use cases: large data cloud migrations, DC decommissions, disaster recovery
- If it takes more than a week to transfer over the network the data, it probably would be recommended to use a Snowball device

Snowball Process

1. Request a snowball device from AWS console for delivery
2. Install the snowball client on the local server
3. Connect the snowball device to the server and copy the files over using the client
4. Shipt back the device when the all the necessary data is transferred to the device
5. The data from the Snowball will be loaded into an S3 bucket
6. Snowball is completely wiped

Snowball Edge

- Snowball Edge adds computational capability to the device
- It can have 100TB of capacity with either:
    - 24 vCPU (Storage optimized)
    - 52 vCPU & optional GPU (Compute optimized)
- Supports a custom EC2 AMI so it can perform processing on the go
- Supports custom Lambda functions
- It is useful for pre-processing data while it is moving
- Use cases: data migration, image collation, IoT capture, machine learning

Snowmobile

- It is truck which can transfer exabytes of data
- Each Snowmobile has 100PT of data storage capacity
- Better than Snowball if more than 10PB of data should be transferred

Snowball into Glacier

- Snowball can not import data directly to Glacier
- We have to use Amazon S3 first, adn an S3 lifecycle policy

Storage Gateway

Hybrid Cloud for Storage

- Hybrid cloud:
    - Part of a company's infrastructure is in the public cloud (like AWS)
    - Part of a company's infrastructure is on-premise
- S3 is a proprietary storage technology (unlike EFS/NFS), it can be exposed to on-premise servers through a storage gateway

Storage Gateway Introduction

- Bridge between on-premise data and cloud data in S3
- Uses cases for storage gateway with S3: disaster recovery, backup and restore, tiered storage
- AWS provides 3 types of storage gateways:
    - File Gateway: allows us to view files from the local files system, but this files are backed by S3
    - Volume Gateway: same as file gateway but with volumes
    - Tape Gateway: used for backup and recovery

File Gateway

- Configured S3 buckets are accessible using NFS and SMB protocols
- Supports S3 Standard, S3 IA, One Zone IA
- Each buckets will have its own IAM roles in order to be accessed by the file gateway
- Most recently used data is cached in the file gateway
- File Gateway can be mounted on many servers (because of the NFS protocol)

Volume Gateway

- Block storage using iSCSI protocol backed by S3
- EBS snapshots are created time to time which are stored in S3, these will help use to restore un-premise volumes
- Cached volumes: low latency access to the most recently used data

- Stored volumes: entire dataset is on premise, scheduled buckets are stored in S3
- Volumes are usually mounter using iSCSI protocol, for on-premise it will look like a local volume

Tape Gateway

- Some companies have backup processes using physical tapes
- With tape gateway these companies can use the same process, but the data will backed into the cloud
- Virtual Tape Library (VTL) backed by Amazon S3 and Glacier
- Backup processes using iSCSI interface will work as well with tape gateway

File Gateway - Hardware Appliance

- Using file gateway means we need virtualization, otherwise we can use a File Gateway Hardware Appliance
- It is an actual hardware which can be bought from amazon.com
- Helpful for daily NFS backup in small data centers

Storage Gateway Summary

- File access / NFS => File Gateway (backed by S3)
- Volumes / Block Storage / iSCSI => Volume Gateway (backed by S3 with EBS snapshots)
- VTL Tape solution / Backup with iSCSI => Tape Gateway (backed by S3 and Glacier)

Amazon FSx

Amazon FSx for Windows

- EFS is a shared POSIX file system for Linux, meaning it can not be used with Windows
- FSx for Windows is a fully managed Windows file system share drive
- Supports SMB protocol and Windows NTFS

- It has Active Directory integration, ACLs and user quotas
- It is built on top of SSD, it can scale up to 10s of GB/s, millions of IOPS and 100s PB of data
- It can accessed from on-premise infrastructure
- It can be configured to be Multi-AZ (high availability)
- Data is backed-up daily to S3

Amazon FSx for Lustre

- Lustre is a type of parallel distributed file system for large-scale computing
- Lustre is derived from "Linux" and "cluster"
- FSx for Lustre is used with machine learning and High Performance Computing (HPC), example: video processing, financial modelling, electronic design automation
- Scales up to 100s GB/s, millions of IOPS, sub-ms latencies
- It has seamless integration with S3
    - Can "read" S3 as a file system
    - Can write the output of the computations back to S3
- It can be used with on-premise servers

Storage Comparison

- S3: Object Storage
- Glacier: Object Archival
- EFS: Network File System for Linux instances, POSIX filesystem
- FSx for Windows: Network File System for Windows servers
- FSx for Lustre: High Performance Computing for Linux systems
- EBS volumes: Network storage for one EC2 instance at a time
- Instance Storage: Physical storage for EC2 instance (high IOPS)
- Storage Gateway
- Snowball / Snowmobile: move large data to the cloud, physical

Networking - VPC

CIDR - Classless Inter-Domain Routing

- CIDR is used for Security Group rules and AWS networking in general
- It is used to define an IP address range
    - ww.xx.yy.zz/32 - one IP address
    - 0.0.0.0/0 - all possible IP addresses
    - 192.168.0.0/26 - range of 192.168.0.0 - 192.168.0.63 64 IP addresses
- CIDR has to component:
    - The base IP: represents an IP from a given the range
    - The subnet mask: defines how many bits can change in the IP
- The subnet mask can take 2 forms:
    - 255.255.255.0 - less common
    - /24 - more common
- The subnet mask basically allows a part of the base IP to get additional next values:
    - /32 allows for 1 IP => 2^0
    - /31 allows for 2 IPS => 2^1
    - /30 allows for 4 IPs => 2^2
    - /29 allows for 8 IPs => 2^3 ...
    - /0 allows for 2^31 IPs, all IPs
- Most used subnet masks:
    - /32 - no IP number can change
    - /24 - last IP number can change (256 IP addresses)
    - /16 - last 2 IP numbers can change (65536 IP addresses)
    - /8 - last three IP numbers can change
    - /0 - all the IP numbers can change
- To calculate CIDR, use: https://www.ipaddressguide.com/cidr

Private vs Public IP Addresses (IPv4)

- Private IP can only allow certain values:
    - 10.0.0.0 - 10.255.255.255 (10.0.0.0/8) - used for big networks
    - 172.16.0.0 - 172.31.255.255 (172.16.0.0/12) - default AWS private VPC
    - 192.168.0.0 - 192.168.255.255 (192.168.0.0/16) - home networks
- All the rest of the IP addresses are considered public

Default VPC

- All new accounts have a default VPC
- New instances are launched into default VPC if no subnet is specified
- Default VPCs have internet connectivity and all instances have public IP address

- We also get a public and a private DNS name

VPC - Virtual Private Cloud

- We can have multiple VPCs in a region (max 5 per region - soft limit)
- Max CIDR per VPC is 5, for each CIDR:
    - Min size is /28 = 16 IP addresses
    - Max size is /16 = 65K IP addresses
- VPC is private => only the private address ranges are allowed
- A newly created VPC CIDR should not overlap with an existing one used another VPC

VPC Subnets

- Subnets are tied to specific AZs
- AWS reservers 5 IP addresses (first 4 and the last 1 address) in each subnet
- These 5 IP addresses are not available and can not be assigned to an instance
- Example, in case of a CIDR block 10.0.0.0/24 the reserver IPs are the following:
    - 10.0.0.0 - Network address
    - 10.0.0.1 - Reserved by AWS for the VPC router
    - 10.0.0.2 - Reserved by AWS for mapping to an Amazon provided DNS
    - 10.0.0.3 - Reserved for future use
    - 10.0.0.255 - Network broadcast address. AWS does not supports broadcast in a VPC, therefore the address is reserved

IGW - Internet Gateways

- Internet gateways helps a VPC instance to connect to the internet
- It scales horizontally and is HA and redundant
- Must be created separately from a VPC
- One VPC can have only one attached internet gateway, one internet gateway can have only one VPC
- Internet Gateway is also a NAT for the instances that have a public IPv4
- Internet Gateways on their one do not allow internet access, route tables also must be configured

NAT Instances - Network Address Translation (outdated)

- Allow instances in private subnets to connect to the internet
- NAT instances must be launched in a public subnet to have internet access
- *Source/Destination Check* flag for EC2 must be disabled
- NAT instanced should have an Elastic IP attached to them
- Route tables must be configured to route traffic from the private subnet to the NAT instance
- In order to set up we must use a pre-configured Amazon Linux AMI
- It is not HA/resilient out of the box => We would need to set up an ASG in multi AZ + resilient user data script
- Internet traffic bandwidth depends on the EC2 instance performance

NAT Gateway

- AWS managed NAT, higher bandwidth, better availability, no administration
- It is pay by hour for usage and bandwidth
- NAT is created in a specific AZ and it uses an Elastic IP
- It can not be used by an instance from the same subnet as the NAT
- Required an IGW
- 5Gbps of bandwidth with automatic scaling up to 45Gbps
- There is no security group to manage
- NAT Gateway is resilient within a single AZ
- We must create multiple NAT Gateways in multiple AZs for fault-tolerance

Comparison between NAT Instance and NAT Gateway

- AWS comparison: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-comparison.html

DNS Resolution in VPC

-DNS Resolutions settings: - enableDnsSupport: - By default is set to true - Helps decide if DNS resolution is supported for the VPC - If true, queries the AWS DNS server at 169.254.169.253 - enableDNSHostName: - By default is set to false for an user

created VPC, true for the Default VPC - Won't do anything unless enableDnsSupport=true - If true, the VPC assigns public host names to EC2 instances

- If we use custom DNS domain names in a private zone in Route53, we must set both of these attributes to true

Network Access Control Lists (NACL) and Security Groups (SG)

- NACL is a like a firewall which controls traffic which goes outside of comes inside a subnet
- The default NACL allows everything outbound and everything inbound
- We define one NACL per subnet, new subnets are assigned the default NACL
- We can define NACL rules:
    - Rules have a number (1 - 32766) which defines the precedence
    - Rules with lower number have higher precedence
    - Rule with higher precedence wins at the time of evaluation
    - Last rule is has the precedence of asterisk (*) and denies all the requests. This rule is not editable
    - AWS recommends adding rules by increment of 100
- Newly created created NACL will deny everything (last rule)
- NACLs are great way of blocking a specific IP address at the subnet level

Security Group	Network ACL
Operates at the instance level	Operates at the subnet level
Supports allow rules only	Supports allow rules and deny rules
It is stateful: return traffic is automatically allowed	It is stateless: return traffic must be explicitly allowed by the rules
All rules are evaluated before deciding to allow traffic	Rules have a precedence when deciding whether to allow or deny traffic

It is associated to an instance inside of a VPC	Automatically applies to all instances in the subnet

- Example of NACL with ephemeral ports: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-network-acls.html

VPC Peering

- VPC Peering allows to connect two VPCs privately using AWS network. This will make them behave as if they were the same network
- Networks must have non-overlapping CIDRs
- VPC Peering connections is not transitive, which means it must be established for each VPC that needs to communicate with other
- VPC Peering can be done form one AWS account to another
- We must update the route tables in each VPC subnet to ensure instances can communicate with each other
- VPC peering can work inter-region, cross-account. We can reference a security group of a peered VPC (this works cross account also)

VPC Endpoints

- Allow connection to AWS services from VPCs using a private network instead of the public internet
- They scale horizontally
- They remove the need of IGW, NAT, etc. to access AWS services
- There are 2 kinds of VPC Endpoints:
    - Interface: provisions an ENI (private IP address) as an entry point. A security group must be attached to it. Most AWS services use this method
    - Gateway: provisions a target and must be used in a route table. Gateway is used for S3 and DynamoDB
- In case of issues:
    - We have to check DNS settings resolution in the VPC
    - We have to check the route tables

VPC FLow Logs

- Flow logs help capture information about the IP traffic going into the interfaces

- There are 2 kinds of Flow Logs:
    - VPC Flow Logs
    - Subnet Flow Logs
    - Elastic Network Interface Flow Logs
- Flow logs help monitor and troubleshoot connectivity issues
- Flow logs data can go to S3/CloudWatch logs
- It can capture network information from some of the AWS managed interface too: ELB, RDS, ElastiCache, Redshift, AWS Workspaces

Flow Log Syntax

```
<version> <account-id> <interface-id> <srcaddr> <dstaddr> <srcport> <dstport>
<protocol> <packets> <bytes> <start> <end> <action> <log-status>
```

- Srcaddr, dstaddr help identify problematic IP addresses
- Scrport, dstport help identify problematic ports
- Action: success or failure of the request due to SG/NACL
- Flow logs can be used for analytics on usage patterns or malicious behavior
- We can query VPC flow logs using Athena on S3 or CloudWatch Logs Insights

Bastion Hosts

- Bastion hosts are used to SSH into instances from private subnets
- The bastion host is sitting in the public subnet which is connected to the private subnets
- Bastion host SG must be really strict
- We have to make sure the bastion host only has port 22 traffic from the IP we need, not from the security group of other instances

Site to Site VPN

- Virtual Private Gateway:
    - VPN concentrator on the AWS side of the VPN connection
    - Virtual Private Gateway (VPG) is created and attached to the VPC from which we want to create the site-to-site VPN connection
    - Possibility to customize the ASN
- Customer Gateway:

    - Software application or physical device on customer side of the VPN connection
    - IP Address: use static, internet-routable IP address for the customer device. If it is behind a NAT (with NAT-T enabled), use the public IP address of the NAT

Direct Connect (DX)

- Direct Connect provides a dedicated private connection from a remote network to the AWS VPC
- Dedicated connection must be setup between the client's DC and AWS Direct Connect locations
- In AWS we still need to setup a Virtual Private Gateway in the VPC
- DX will allow access to public resources (S3) and private (EC2) on the same connection
- Use cases:
    - Increase bandwidth throughput
    - More consistent network experience - application using real-time data feeds
    - Hybrid Environment
- DX supports both IPv4 and IPv6
- Direct Connect Gateway: if we want to setup a Direct Connect to one or more VPC in many different regions (but in the same account), we must use a Direct Connect Gateway
- Direct Connect Gateway does not replace VPC Peering
- Direct Connect has two connections types:
    - Dedicated Connections: 1 Gbps and 10 Gbps capacity
        - Physical ethernet port dedicated to a customer
        - In order to get this a request has to be made to AWS first, then completed by AWS Direct Connect Partners
    - Hosted Connections: 50 Mbps, 500 Mbps, up to 10 GBps
        - Connection requests are made via AWS Direct Connect Partners
        - Capacity can be added or removed on demand
- Lead times are often longer than 1 month

Direct Connect Encryption

- Data in transit is not encrypted but is private

- In case there is need for encryption we can combine DX with VPN which provides IPsec-encrypted private connection

Egress Only Internet Gateway

- Egress only Internet Gateway works only for IPv6
- Similar function as a NAT, but for IPv6 (NAT is for IPv4)
- All IPv6 addresses are public addresses, therefore all our instances are publicly accessible
- Egress Only Internet Gateway gives our IPv6 instances access to the internet, but they wont be directly reachable by the internet
- After creating an Egress Only Internet Gateway we have to edit the route tables

Exposing Services in a VPC to Other VPC

- Option 1: make it public
    - Goes through the public internet
    - Tough to manage access
- Option 2: VPC peering
    - Must create many peering relations
    - Opens the whole internal network to the other network
- Option 3: AWS Private Link (VPC Endpoint Services)
    - Most secure and scalable way to expose a service to multiple other VPCs
    - The solution does not require VPC peering, IGW, NAT, route tables
    - Requires a network load balancer in the service VPC and an ENI in the customer VPC

EC2-Classic and AWS ClassicLink (deprecated)

- EC2-Classic: instances are running in a single network shared with other customers
- Amazon VPC: instances are running logically isolated in an AWS account
- ClassicLink allows to link EC2-Classic instances to a VPC inside an account. In order to have a link between a VPC and an EC2-Classic instance we have to:
    - Create a ClassicLink and associate a security group to it which will enable private communication
    - ClassicLink removes the necessity to have IPv4 public addresses for this type of communication

AWS VPN CloudHub

- Provides secure communication between multiple customer networks in case we have multiple VPN connections
- Low cost hub-and-spoke model for primary or secondary network connectivity between locations

Transit Gateway

- It is a star connection between all the VPCs and the on-premises infrastructure
- It is for having transitive peering between thousands of VPCs and on-premises and hub-and-spoke connections
- Transit Gateway is a regional resource, but it can work cross-region
- It supports cross-account sharing using Resource Access Manager (RAM)
- We can peer Transit Gateways across regions
- Route tables can limit which VPC can talk with what other VPC
- Works with Direct Connect Gateway, VPN connections
- Supports IP Multicast (not supported by any other AWS services)

VPC Summary

- CIDR: IP Range
- VPC: Virtual Private Cloud => we define a list of IPv4 and IPv6 CIDRs
- Subnets: tied to an AZ, we define it by CIDRs
- Internet Gateway: at the VPC level, provides IPv4 and IPv6 internet access
- Route tables: must be edited to add routes from subnets to the IGW, VPC peering connections, VPC Endpoints, etc.
- NAT Instance: gives internet access to instances in a private subnets. Deprecated, must be setup in a public subnet, Source/Destination flag should be disabled
- NAT Gateway: managed by AWS, provides scalable internet access to private instances, only for IPv4
- Private DNS + Route 53: enables DND Resolution + DNS hostname inside a VPC
- NACL: Stateless, subnet level routes, used for filter inbound and outbound traffic
- Security Groups: Stateful, operate at the instance level
- VPC Peering: connect to VPC with non overlapping CIDRs, non transitive
- VPC Endpoints: provide private access to AWS Services

- VPC FLow Logs: can be configured at VPC/Subnet/ENI level, used for monitoring traffic
- Bastion Hosts: public instances to SSH into, used for connecting to other private instances via SSH
- Site to Site VPN: used for setup a Customer Gateway on DC, a Virtual Private Gateway on VPC and site-to-site VPN over public internet
- Direct Connect: used for setup a VIrtual Private Gateway on VPC and establish a direct private connection to an AWS Direct Connect Location
- Direct Connect Gateway: setup Direct Connect to many VPCs in different regions
- Internet Gateway Egress: similar to NAT, but for IPv6
- Private Link (VPC Endpoint Services): connect services from one VPC to another
- ClassicLink: connect EC2-Classic instances to VPCs
- VPN CloudHub: hub-and-spoke VPN model to connect sites
- Transit Gateway: transitive peering connection for VPC, VPN and DX

Databases

Choosing the Right Database

- There are several databases on AWS from which we can choose from
- Guid line questions to be able to select the best database for the product:
    - Is the workload read-heavy, write-heavy or balanced? What are the throughput needs? Will the throughput change, will fluctuate or will we have to scale the DB over time?
    - How much data do we store and for how long? Will the size of the data grow? What is the average size of an object in the DB? How frequently are these objects accessed?
    - Data durability: should the data be stored for a week or forever? Is the database going to be a source of truth?
    - What are the latency concerns?
    - What is the data model? How will the data be queried? Will the data be structured, semi-structured or unstructured?
    - Do we need strong schema or flexible schema? Do we need reporting? Do we need advanced search?
    - What are the license costs? Can we switch to a cloud native database such as Aurora, DynamoDB, etc?

Database Types on AWS

- RDBMS (SQL/OLTP): RDS, Aurora - great for joins and normalized data
- NoSQL: DynamoDB, ElastiCache (key/value pairs), Neptune (graphs), DocumentDB (json) - no joins, no SQL
- Object Store: S3 (for big objects) / Glacier (for backups, archive)
- Data Warehouse (SQL Analytics / BI): Redshift (OLAP), Athena
- Search: ElasticSearch (JSON) - free text, unstructured searches
- Graphs: Neptune - displays relationships between data

RDS

- Managed PostgreSQL, MySQL, Oracle, SQL Server
- AWS provisions an EC2 instance behind the scenes and EBS Volume
- RDS has support for multi AZ deployment and read replicas
- Security through IAM, security groups, KMS, SSL
- Backup/Snapshot/Point in time restore functionality
- Managed and scheduled maintenance
- Monitoring happens through CloudWatch
- Use cases: store relational datasets, perform SQL queries, transactional inserts, deletes, updates

RDS for Solutions Architects

- Operations: small downtime when a failover happens, when maintenance happens, when scaling of read replicas. Restoring snapshots implies manual intervention
- Security: AWS is responsible for OS security. We are responsible for setting up KMS, SG, IAM policies, authorizing users, using SSL
- Reliability: Multi AZ feature, failover in case of failures
- Performance: depends on the EC2 type, EBS volume type. We have the ability to add Read Replicas. RDS does not auto scale
- Cost: pay per hour based on provisioned EC2 and EBS

Aurora

- Managed by AWS, under the RDS service, having a compatible API with PostgreSQL and MySQL

- Data is held in 6 replicas across 3 AZs
- It has some auto healing capabilities
- Auto scaling read replicas which can be global
- Aurora databases can be global which can be used for disaster recovery purposes or latency purposes
- Auto scaling of storage form 10GB to 64TB
- We define an EC2 instance for Aurora instances
- Aurora has the same security, monitoring and maintenance features as other RDS databases
- Aurora has a serverless option
- Use cases: similar to RDS, but with less maintenance, more flexibly, more performant at a higher cost

Aurora for Solutions Architects

- Operations: less operations, auto scaling for storage
- Security: AWS is responsible for OS security. We are responsible for setting up KMS, SG, IAM policies, authorizing users, using SSL
- Reliability: Multi Az, highly available, Aurora serverless option
- Performance: 5x performance (according to AWS) compared to other RDS types. up to 15 read replicas (only 5 for RDS)
- Cost: pay per hour based on EC2 and storage usage. Possibly lower costs compared to other enterprise grade databases such as Oracle

ElastiCache

- Managed Redis or Memcached
- In-memory data store, sub-millisecond latency
- Must provision and EC2 instance type
- Offers support for clustering (Redis) and Multi AZ, read replicas (sharding)
- Security through IAM, SG, KMS, Redis Auth
- Possibility to do backups, snapshots and point in time restore (for Redis)
- It has managed and scheduled maintenance
- Monitoring happens through CloudWatch
- Uses cases: key/value store, frequent reads, less writes, cache results for DB queries, store session data for websites

ElastiCache for Solutions Architects

- Operations: same as RDS
- Security: same as RDS without the possibility of IAM policies. In the other hand there is Redis Auth for authentication
- Reliability: clustering, multi AZ
- Performance: sub-millisecond performance, read replicas for sharding, popular cache option
- Cost: pay per hour based on EC2 instance type and storage

DynamoDB

- Managed NoSQL database, proprietary to AWS
- Serverless, provisioned capacity, supports auto scaling, provides on demand capacity as well
- Can replace ElastiCache as a key/value store
- Highly available, Multi AZ by default, reads and writes are decoupled, DAX for read cache
- Reads can be eventually consistent or strongly consistent
- Security, authentication and authorization is done through IAM
- We can enable DynamoDB Streams to stream all the changes in DynamoDB and integrate with AWS Lambda
- Provides backup/restore features, Global Table feature
- Monitoring is done through CloudWatch
- The tables can only be queried on primary key, sort key or indexes
- Use cases: serverless application development, distributed serverless cache

DynamoDB for Solutions Architects

- Operations: no operations needed, auto scaling capability out of the box, serverless
- Security: full security through IAM policies, KMS encryption, SSL in flight
- Reliability: Multi AZ, provides backups
- Performance: single digit millisecond performance, DAX for caching reads, performance does not degrade if the application scales
- Cost: pay per provisioned capacity and storage usage (no need to guess in advance any capacity)

S3

- S3 is a key/value store for objects
- Great for big objects, not so great for small objects
- Serverless, scales infinitely, max object size is 5TB
- Eventually consistency for overwrites and deletes, strong consistency for new objects
- Tiers: S3 Standard, S3 IA, S3 One Zone IA, Glacier
- Features: versioning, encryption, cross region replications
- Security: IAM, bucket policies, ACL
- Encryption: SSE-S3, SSE-KMS, SSE-C, client side encryption, SSL in transit
- Use cases: static files, key value store for big files, website hosting

S3 for Solutions Architects

- Operations: no operations needed
- Security: IAM, bucket policies, ACL, encryption, SSL
- Reliability: 99.999999999% durability / 99.99% availability, multi AZ, cross-region replication
- Performance: scales to thousands of reads/writes per second, transfer acceleration / multi-part upload for big files
- Cost: pay per storage usage, network cost, request number

Athena

- Fully serverless database (query engine) with SQL capabilities
- Used to query data in S3
- Pay per query
- Output results back to S3
- Secured through IAM
- Use cases: one time SQL queries, serverless queries on S3, log analytics

Athena for Solutions Architects

- Operations: no operations needed, serverless
- Security: IAM + S3 security
- Reliability: managed service, uses Presto engine, highly available
- Performance: queries scale based on data size
- Cost: per per query / per TB of data scanned, serverless

Redshift

- Redshift is based on PostgreSQL, but it's not used for OLTP
- It's OLAP - only analytical processing (analytics and data warehousing)
- 10x better performance than other data warehouses, scales to PBs of data
- Columnar database: data is stored in columns
- Massively parallel query execution (MPP), highly available
- Pay as you go based on the instances provisioned
- Has a SQL interface for performing the queries
- Great to use with BI tools such as AWS Quicksight or Tableau
- Data is loaded form S3, DynamoDB, Database Migration Service (DMS), other DBs
- Can scale from 1 node up to 128 nodes, up to 160 GB of space per node
- Node types: compute nodes for performing queries, result is sent to leader nodes
- Redshift Spectrum: perform queries directly against S3 (no need to load the data)
- Provides backup and restore
- Security is accomplished with VPC/IAM/KMS
- Redshift Enhanced VPC Routing: COPY/UNLOAD goes through VPC

Snapshots and Disaster Recovery

- Snapshots are point-in-time backups of a cluster, stored internally in S3
- Snapshots are incremental (only what has changed is saved)
- Snapshots can be restored into new Redshift clusters
- Snapshots can be automated: every 8 hours, every 5 GB, or on schedule
- Manual snapshots: snapshots are retained until we delete them
- We can configure Redshift to automatically copy snapshots to another regions

Redshift Spectrum

- Data can be queried from S3 without the need to load it into Redshift
- We must have a Redshift cluster available to start the query
- The query is then submitted to thousands of Redshift Spectrum nodes

Redshift for Solutions Architects

- Operations: similar to RDS
- Security: IAM, VPC, KMS, SSL

- Reliability: highly available, auto healing feature
- Performance: 10x performance vs other data warehousing, compression
- Cost: pay per node provisioned, 1/10th of the cost of other warehouses

Neptune

- Fully managed graph database
- Graph use cases:
    - Highly relational data
    - Social networking
    - Knowledge graphs
- Highly available across 3 AZ, up to 15 read replicas
- Provides point-in-time recovery, continuous back-up to S3
- Support for KMS encryption at rest + HTTPS

Neptune for Solutions Architects

- Operations: similar to RDS
- Security: IAM, VPC, KMS, SSL, IAM Authentication
- Reliability: Multi AZ, clustering
- Performance: best suited for graphs, clustering can improve performance
- Cost: similar for RDS, per per node provisioned

ElasticSearch

- Open source technology
- With ElasticSearch we can search any field, we can have partial matches as well
- It's common to use ElasticSearch as a complement to another database
- We can provision a cluster of instances
- Built-in integration with: Amazon Kinesis Data Firehose, AWS IoT and Amazon CloudWatch Logs for data ingestion
- Security through Cognito and IAM, KMS encryption, SSL and VPC
- Comes with Kibana and Logstahs - ELK stack

ElasticSearch for Solutions Architects

- Operations: similar to RDS
- Security: Cognito, IAM, VPC, KMS, SSL

- Reliability: Multi AZ, clustering
- Performance: based on ElasticSearch project, petabyte scale
- Cost: similar for RDS, per per node provisioned

SQS - Simple Queue Service

Integration and Messaging

- Deploying multiple applications will inevitable result in the necessity of a communication layer between them
- There are 2 types of integration communication patterns:
    - Synchronous communication
    - Asynchronous communication
- Application decoupling models:
    - SQS: queue model
    - SNS: pub/sub model
    - Kinesis: real-time streaming model

SQS - Standard Queue

- Oldest offering on AWS (over 10 years old)
- Fully managed service, used to decouple applications
- Attributes:
    - Unlimited throughput, unlimited number of messages in the queue
    - Each message is short leaved: default retention period is 4 days, maximum is 14 days
    - Low latency: <10 ms on publish and receive
    - Limitation for message size: maximum size of a message is 256KB
- SQS Standard Queue can have duplicate messages (at least once delivery)
- It also can have out of order messages (best effort ordering)

SQS Standard - Producing Messages

- Producers send messages to the queue using the SDK (SendMessage API)
- The message is persisted on the queue until a consumer deletes it
- Message retention: default 4 days, up to 14 days
- SQS standard has unlimited throughput

SQS Standard - Consuming messages

- Consumers are applications (running on EC2 instances, other servers or AWS Lambda)
- Consumers poll the queue for messages (they can receive up to 10 messages at a time)
- After the messages are processed the consumers delete the messages from the queue using DeleteMessage API
- Multiple consumers:
    - Consumers receive the messages in parallel
    - Each consumer consumes a fraction of the number of the messages sent
    - We can scale the number of the consumers based on the throughput of processing
- SQS with Auto Scaling Group:
    - We can scale based on the ApproximateNumberOfMessages metric by creating a CloudWatch alarm

SQS Security

- Encryption:
    - In-flight encryption using HTTPS
    - At-rest encryption using KMS
    - Client-side encryption if the client wants to perform encryption/decryption itself
- Access Control: IAM policies to regulate access to the SQS API
- SQS Access Policies:
    - Useful for cross-account access to SQS queues
    - Useful for allowing other services (SNS, S3) to write to an SQS queue

Message Visibility Timeout

- After a message is polled by a consumer, it becomes invisible to other consumers
- Default message visibility timeout is 30 seconds, which means the consumer has 30 seconds to process the message
- After the message visibility timeout is over, the message becomes visible to other consumers

- If the processing is not finished during the visibility timeout, there is a chance the message will be processed twice
- If a consumer knows that the processing wont finish in time, it can use the ChangeVisibility API to request more time
- If the message visibility timeout is high and the processing fails, it may take a long time for the message to be processed again
- If the visibility timeout it too short, we may end up processing the same message twice
- Best practice: the visibility timeout should be set to something appropriate. The consumer must be implemented in a way to use the ChangeVisibility API

Dead Letter Queues

- If a consumer fails to process a message within the visibility timeout, the messages goes back to the queue. This can happen multiple times.
- We can set a MaximumReceives threshold, which denotes how many time a message should be able to go back to the queue
- If the MaximumReceives threshold is exceeded, the message is sent to a dead letter queue
- DLQs are useful for debugging
- We have to make sure the messages are processed in DLQ before expiring. It is not a good idea to set a short expiration time for the DLQ

Delay Queue

- Delaying a message means the consumers wont be able to see the message for a period of time after it was sent. Delay time can be up to 15 minutes
- Delay can be set at a queue level or tt also can be set to message level using the DelaySeconds parameter

FIFO Queues

- FIFO - First In First Out
- The messages will be ordered in the queue, meaning that the messages will be consumed in the same order as they were sent
- FIFO queues have limited throughput: 300 msg/s without batching, 3000 msg/s with batching
- Exactly-once send capability (by activating content-based deduplication)

- The name of the FIFO queue must end with the `.fifo`

SQS With Auto Scaling Group

- Allows scaling the number of EC2 instances based on the available messages in the queue
- In order to accomplish auto scaling we have to create a CloudWatch custom metric representing the number of available messages on the queue divided by the number of EC2 instances. This metric is pushed from an EC2 instance

Data Ordering in SQS

- For standard SQS queues there is no data ordering
- For SQS FIFO, if we don't use a Group ID, messages are consumed in the order they are sent, with only one consumer
- Messages can be group by specifying a Group ID for the message
- For each group we can have different consumers, which will read messages in order

SNS - Simple Notification Service

- Pub/Sub model
- The event produces only sends messages to one SNS topic
- Each subscriber to the topic will get all the messages be default (we can filter them, if we want)
- We can have up to 10 million subscribers per topic
- We cave up to 100K topics
- Subscribers to the topic can be:
    - SQS
    - HTTP/HTTPS
    - Lambda
    - Emails
    - SMS messages
    - Mobile Notifications
- Many different services integrate with SNS for notifications, for example:
    - CloudWatch (for alarms)
    - Auto Scaling Groups notifications

    - S3 (bucket events)
    - CloudFormation (state changes)
    - Etc...
- How to publish?
    - In order to publish we must create a topic using the SDK
    - We may create one or many subscriptions
    - We publish data to the topic
- Direct Publish (for mobile apps SDK)
    - Create a platform application
    - Create a platform endpoint
    - Publish to the platform endpoint
- Direct Publish works with Google GCM, Apple APNS, Amazon ADM

Security

- Encryption:
    - In-flight encryption using HTTPS API
    - At-rest encryption using the KMS keys
    - Client-side encryption if the client wants to perform encryption/decryption itself
- Access Controls: IAM policies to regulate access to the SNS API
- SNS Access Policies (similar to S3 bucket policies):
    - Useful for cross-account access to SNS topics
    - Useful for allowing other services (S3) to write to an SNS topic

SNS + SQS Fan Out

- Send a message to multiple SQS queues using SNS
- Push one in SNS, receive in all SQS queues which are subscribers
- Fully decouples, no data loss
- SQS allows for data persistance, delayed processing and retries of work
- Ability to add more SQS subscribers over time
- SQS queues must have an allow access policy for SNS to be able to write to the queues
- SNS cannot send messages to SQS FIFO queues (AWS limitation)!
- Use case: send S3 events to multiple queues:
    - For the same combination of even type and prefix we can only have one S3 Event rule

    - In case we want to send the same S3 event to many SQS queues, we must use SNS fan-out

AWS Kinesis

- Kinesis is a managed alternative to Apache Kafka
- It is a big data stream tool, which allows to stream application logs, metrics, IoT data, click streams, etc.
- Compatible with many streaming frameworks (Spark, NiFi, etc.)
- Data is automatically replicated to 3 AZ
- Kineses offers 3 types of products:
    - Kinesis Streams: low latency streaming ingest at scale
    - Kinesis Analytics: perform real-time analytics on streams using SQL
    - Kinesis Firehose: load streams into S3, Redshift, ElasticSearch

Kinesis Streams Overview

- Streams are divided in ordered Shards/Partitions
- For higher throughput we can increase the size of the shards
- Data retention is 1 day by default, can go up to 365 days
- Kinesis Streams provides the ability to reprocess/replay the data
- Multiple applications can consume the same stream, this enables real-time processing with scale of throughput
- Kinesis is not a database, once the data is inserted, it can not be deleted

Kinesis Stream Shards

- One stream is made of many different shards
- 1MB/s or 1000 messages at write PER SHARD
- 2MB/s read PER SHARD
- Billing is done per shard provisioned, we can have as many shards as we want as long as we accept the cost
- Ability to batch the messages per calls
- The number of shards can evolve over time (reshard/merge)
- Records are ordered per shard!

Kinesis Streams API - Put Records

- Data must be sent form the PutRecords API to a partition key
- Data with the same key goes to the same partition (helps with ordering for a specific key)
- Messages sent get a sequence number
- Partition key must be highly distributed in order to avoid hot partitions
- In order to reduce costs, we can use batching with PutRecords API
- It the limits are reached, we get a *ProvisionedThroughputException*

Exceptions

- ProvisionedThroughputException Exceptions:
    - Happens when the data value exceeds the limit exposed by the shard
    - In order to avoid the, we have to make sure we don't have hot partitions
    - Solutions:
        - Retry with back-off
        - Increase shards (scaling)
        - Ensure the partition key is good

Consumers

- Consumers can use CLI or SDK, or the Kinesis Client Library (in Java, Node, Python, Ruby, .Net)
- Kinesis Client Library (KCL) uses DynamoDB to checkpoint offsets
- KCL uses DynamoDB to track other workers and share work amongst shards

Security

- Control access / authorization using IAM policies
- Encryption in flight using HTTPS endpoints
- Encryption at rest using KMS
- Possibility to encrypt/decrypt data client side
- VPC Endpoints available for Kinesis to be access within VPCs

Kinesis Data Firehose

- Fully managed service, no administration required, provides automatic scaling, it is basically serverless
- Used for load data into Redshift, S3, ElasticSearch and Splunk

- It is Near Real Time: 60 seconds latency minimum for non full batches or minimum 32 MB of data at a time
- Supports many data formats, conversions, transformation and compression
- Pay for the amount of data going through Firehose

Kinesis Data Streams vs Firehose

- Streams:
    - Requires to write custom code (producer/consumer)
    - Real time (~200 ms)
    - Must manage scaling (shard splitting / merging)
    - Can store data into stream, data can be stored from 1 to 7 days
    - Data can be read by multiple consumers
- Firehose:
    - Fully managed, sends data to S3, Redshift, Splunk, ElasticSearch
    - Serverless, data transformation can be done with Lambda
    - Near real time
    - Scales automatically
    - It provides no data storage

Kinesis Data Analytics

- Can take data from Kinesis Data Streams and Kinesis Firehose and perform some queries on it
- It can perform real-time analytics using SQL
- Kinesis Data Analytics properties:
    - Automatically scales
    - Managed: no servers to provision
    - Continuous: analytics are done in real time
- Pricing: pay per consumption rate
- It can create streams out of real-time queries

Data Ordering with Kinesis

- Data with the same partition key goes to the same shard
- Data is ordered per shard

SQS vs SNS vs Kinesis

- SQS:
    - Consumers pull data
    - Data is deleted after being consumed
    - Can have many consumers as we want
    - No need to provision throughput
    - No ordering guarantee in case of standard queues
    - Capability to delay individual messages
- SNS:
    - Pub/Sub: publish data to many subscribers
    - We can have up to 10 million subscribers per topic
    - Data is not persisted (it is lost if not delivered)
    - Up to 10k topics per account
    - No need to provision throughput
    - Integrates with SQS for fan-out architecture
- Kinesis Data Streams:
    - Consumers "pull data"
    - We can have as many consumers as we want
    - Possibility to replay data
    - Recommended for real-time big data analytics and ETL
    - Ordering happens at the shard level
    - Data expires after X days
    - Must provision throughput

Amazon MQ

- SQS and SNS are cloud-native, they are using proprietary protocols from AWS
- Traditional application running on on-premise may use queues with open protocols such as: MQTT, AMQP, STOMP, Openwire, WSS
- When migrating to cloud instead of re-engineering the application to SQS or SNS, we can use Amazon MQ
- Amazon MQ is basically managed Apache ActiveMQ
- Amazon MQ does not scale as much as SQS/SNS
- It runs on a dedicated machine, can urn in HA with failover
- It has both queue anf topic features