AWS Certified Solutions Architect

SAA-C01

# Domain 1: Design Resilient Architectures

## Choose reliable/resilient storage.

### S3 Storage Tiers

* S3 Standard Tier
  + General purpose
  + High durability = 11 9s
  + Multi-AZ
    - Sustain 2 concurrent failures
  + 99.99 availability
  + Use Cases
    - Big Data Analytics
    - Mobile gaming
* S3 IA
  + Infrequent Access
  + Infrequent but fast access
  + Durability = 11 9s
  + Multi-AZ
  + Lower Cost
  + Pay retrieval fee
* S3 1 Zone IA
  + Infrequent access no replication
  + Single Zone
  + Even lower cost
  + Durability = 11 9s
  + Loose data if AZ is destroyed
  + 99.95 availability
* S3 Intelligent Tiering
  + Same as S3 standard
  + Monthly monitoring fee
  + Objects auto moved between other tiers depending on access
  + Multi-AZ
  + 99.9 Availability
* Glacier
  + Archiving and Backups
  + Large files
  + Not access often
  + Long term storage
  + Alternative to on prem tape
  + Durability = 11 9s
  + Lowest cost + retrieval cost
  + Vaults = buckets
  + Retrieval options
    - Expidited
      * 1 – 5 minutes
    - Standard
      * 3-5 hours
    - Bulk
      * 5-12 hours

#### S3 Lifecycle Rules

* Rules created to move objects to diff tiers
* Transition Actions
  + Defines when an object moved to another storage class (tier)
* Expiration Actions
  + Expire (delete) object after specified time
* Can be used to delete incomplete multipart uploads

## Determine how to design decoupling mechanisms using AWS services.

### Overview

* Allows apps to talk to each other
* Patterns of communication
  + synchronous
    - App to app
    - Problem if sudden spikes in traffic
    - One app can overload the other
    - Uses API Gateway or Load balancer
  + Asynchronous
    - Event based
    - Queue sits in the middle and helps communicate
    - decoupled

### SQS

* Simple queue service
* message queues
* producers send messages to queues
  + each queue can handle multiple producers
* Consumer polls messages from queue
  + Can have multiple consumers per queue
* Types
  + Standard Queue
    - Oldest offering of AWS
    - Fully managed
    - Scales from 1 message per second to 10000 per second
    - Auto scales
    - Default retention of message of 4 – 14 days
    - No limit to messages in queue
    - Low latency
    - Duplicate messaged can exist
    - Can have out of order messages
    - Limitation of 256KB per message
    - Delay Queue
      * Can delay messages up to 15 minutes
      * Can override default with DelaySeconds parameter
  + FIFO Queue
    - Name must end in .fifo
    - Lower throughput
    - Messages are processed in order
    - Send exactly once
    - No per message delay ( can set queue level delay )
    - Dedup in queue
      * 5 minute intervals
    - Message groups
      * Possible to group messages for FIFO ordering
      * Only one worker can be assigned per message group so that messages are processed in order
      * Message group is an extra tag on message
* Producing messages
  + Body
    - 256 KB
    - String
  + MetaData
    - Key value pair
    - Message attributes
  + Delay Delivery
* Get Back from SQS
  + Message ID
  + MD5 Hash
* Consume Messages
  + Poll SQS for messages
  + Receive up to 10 per time
  + Process message within visibility timeout
  + Delete the message using the message ID and receipt handle
* Visibility Timeout
  + If Consumer gets message, it is invisible to other consumers for a period of time
  + 0 seconds to 12 hours
  + Default 30 seconds
  + Can be changed with ChangeMessageVisibility API
* Delete Message API to tell queue message was processed
* Dead Letter Queue
  + Retry timeout if consumers keep retrying and cant process then the message is moved here
* Long Polling
  + Wait if no message is in queue for certain time
  + Decrease number API calls
  + Increase efficiency and latency
  + 1-20 sec wait time
  + 20 recommended
  + API WaitTimeSeconds or set by default for queue

### SNS

* Simple Notification Service
* Assists with Sending messages to many receivers
* SNS Topic
  + Container for the message
  + 100000 topic limit
* Event Producer
  + Creates the message and sends to event SNS Topic
* Event Receiver
  + Subscribes to SNS Topic
  + 10000000 subscriptions per topic
  + Sqs queue
  + HTTP/HTTPS endpoint
  + Lambda
  + Email
  + Sms messages
  + Mobile notifications

#### SNS – Publish

* Topic Publish
  + Within aws server using SDK
  + Create topic
  + Create subscription
  + Publish the topic
* Direct Publish for mobile app SDK
  + Create platform app
  + Create platform endpoint
  + Publish platform endpoint

### SNS and SQS Fan Out

* Push once to SNS and SQS queues can subscribe to it.
* Fully decoupled
* No data loss
* Ability to add additional recievers
* Sqs allows for delayed processing
* Sqs allows for retries of work

### Kinesis

* Managed alternative to Apache Kafka
* Streaming tool to collect logs metrics on any real time big data
* Stream processing frameworks
* Does not scale infinately
* Replicated to 3 AZ
* Big data, real time
* Shards
  + Streams are divided into shards or partitions
  + More shards = more throughput for stream
  + Data retention is 1 – 7 days (1 by default)
  + Can reprocess and replay data
  + Multiple applications can consume the same stream
  + Once data is inserted in Kinesis it can’t be deleted
    - Immutable
  + 1 mb/s or 1000 messages/s at write per shard
  + 2 mb/s at read per shard
  + Billing is per shard
  + Batch available or per message calls
  + Shards can scale up / down
  + Records are ordered per shard
* Put Records
  + Send data to Kinesis
  + Data + message key(partition key)
    - Message key
  + Sequence number to order messages in shard
  + Choose partition key to be highly distributed
  + Batching
    - Reduce cost and increase throughput
  + ProvisionedThroughputExceeded
    - Use retrys
    - Make sure you don’t have a hot shard (shard busier than others)
* Consumer
* Kinesis Security
  + Control access / authorization using IAM policies
  + Encryption in flight with HTTPS Endpoints (SSL)
  + Encryption at rest with KMS
  + Possible to encrypt / decrypt data client side
  + VPC endpoints available for Kinesis

#### Kinesis Data Streams ( kinesis )

* Streams at scale with low latency
* Build custom apps that process or analyze data
* Stores data for later processing by apps

#### Kinesis Analytics

* Real time analytics on streams using SQL
* Auto scaling
* Managed
* Pay for consumption rate
* Can create streams from real-time queries

#### Kinesis Firehose

* Easiest way to Load streams into S3, Redshift, ElasticSearch, etc
* Managed
* Near real time ( 60 second latency )
* Auto scaling
* Support for many data formats ( pay for conversion)
* Pay for the amount of data going through Firehose
* Immediately delivers data to AWS resource

### Amazon MQ

* MQ uses open source protocols unlike SNS and SQS
* Allows you to migrate on prem apps that use these into AWS without rewriting app
* Managed Apache ActiveMQ
* Doesn’t scale as much as SNS / SQS
* Runs on dedicated machine
* Runs with HA and failover
* Has both a queue and topic feature

## Determine how to design a multi-tier architecture solution.

### Disaster Recovery

#### Backup and Restore

* Data is copied to AWS
* Access can happen over internet
* Storage Gateway service

#### Pilot Light

* Critical core service running in AWS
* Scaled when DR event
* Typically use pre installed software on AMIs to deploy other services

#### Warm Standby

* All services running on scaled down system
* Will need to be scaled up to handle full production load

#### Multi-Site

* Full scale services running
* Active-active
* Weighted DNS route

### UserData

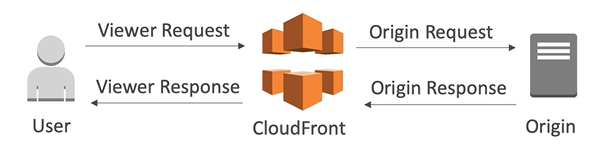
* Can bootstrap instances using EC2 user data script
  + Bootstrap =
    - launch command when instance starts
    - Only run once at first start
* Used to automat boot tasks such as
  + Install updates
  + Install software
  + Etc
* Runs as root user

### Route53

* DNS
* Records
  + A = URL to IPv4
  + AAAA = Url to IPv6
  + CNAME = URL to URL
    - Only for non root domain
      * Must be something.domain.com
      * Not domain.com
  + Alias = URL to AWS resource
    - AWS Resources
      * Expose AWS URL
    - Works for root domain and non root domain
    - Free of charge
    - Native health checks
* Public names
  + DNS domains you buy
  + On internet
* Private Names
  + Internal to VPC
  + Only used within VPC
* Advanced Features
  + Load Balancing
  + Health Checks
    - If route is unhealthy IP will not be returned from Route53
    - Unhealthy if fails x number of checks in a row (3 default)
    - Health if passes x in a row ( 3 default
    - Default health check interval is 30 s ( fast health check are 10s = higher cost)
    - Can use HTTP, TCP, HTTPS (no SSL Verification)
    - Can use cloudwatch alarms as heath check
  + Route Policy
    - Simple
      * Maps a domain to one URL
      * Redirect to single resource
      * Cannot use Health checks
      * Random chosen if multiple IPs returned
    - Weighted
      * Controls % of the requests that go to specific IP
      * Can use health checks
    - Latency
      * Redirect to IP with least latency, usually close to us
      * Latency is determined based on user to designated region
    - Failover
      * Mandatory health checks (Secondary does not need a healthcheck)
      * Auto failover to secondary when primary fails
    - GeoLocation
      * Based on user location (not latency)
      * Should be routed to specific IP
      * Need a default policy in case no match
      * Can use countries or continent as location
    - Multi-Value
      * Route to multiple resources
      * Health checks with records
      * Up to 8 healthy records
      * Not a replacement for ELB
* Time To Live
  + TTL
  + Local DNS cache will hold query response for TTL time
  + So until TTL expires the local machine reads cache
  + Quicker response
* Third party domains (non AWS resources)
  + Can register DNS names with Route53
  + To register Name with another Domain registrar
    - Buy name from third party
    - Create hosted zone in Rout53
    - Update NS Records on 3rd party website to use route53 name servers

### Lambda

* Deploy functions
* Serverless
* Limited by time – short execution
* Run on demand
* Scaling is automated
* Pricing
  + Pay per request
  + Pay for compute time
* Integrated with whole AWS Stack
* Configuration
  + Timeout
    - How long function run before failing
    - Default 3 s
    - Max 15 minutes (900 s)
  + Environment variables
    - Key pairs
    - Access from code
    - Allows you to parameterize code
  + Allocate memory
    - 128 MB to 3 GB
  + Deploy within VPC and assign Security group to functions
  + IAM execution role must be attached to function
* Limits
  + Execution
    - Memory 128 MB to 3008MB (64MB increments)
    - Max execution is 15 minutes (900 s)
    - Disk capacity in a function container ( in /tmp ) = 512 MB
    - Concurency limit = 1000
      * Update via support ticket
  + Deployment
    - Function compressed zip file size < 50 MB
    - Uncompressed code + dependencies < 250 MB
    - Can use the /tmp directory to load other files at startup
      * Good if you want to go over the uncompressed limit
    - Environment vars < 4KB
* Lambda@edge
  + Use Case:
    - Allows you to run global aws lambda alongside cloudfront
    - Implement request filtering before reaching app on cloudfront
    - Website security and privacy
    - Dynamic web application at the edge
    - Search engine optimization ( SEO )
    - Intelligently route across origins and data centers
    - BOT mitigation at the edge
    - Real-Time image transformation
    - A/B Testing
    - User authentication and authorization
    - User Prioritization
    - User tracking and analytics
  + Global lambda function
  + Benefits
    - Build more responsive apps
    - Deployed globally
    - Customize CDN Content
    - Only pay for what you use
  + Can change Cloud front requests and responses
    - Viewer Request
      * After cloudfront receives request from viewer
    - Origin Request
      * Before cloudfront forwards the request to the origin
    - Origin Response
      * After cloudfront receives the response from the origin
    - Viewer response
      * Before cloudfront forwards the response to the viewer



* Lambda functions can also generate responses to the viewers without sending request to origin

## 1.4 Determine how to design high availability and/or fault tolerant architectures.

### Architecture

* + Regions
    - Geographic region around world
  + Availability Zone
    - physical Data Centers within a Region
    - separated from one another for fault tolerance

### Scalability

* handle greater loads by adapting
* Virtical Scalablity
  + Scale up / down
  + Adding additional HW into server
  + Increasing instance size
* Horizontal Scalability
  + Elasticity
  + Scale out / in
  + Adding additional server in parallel
  + Distributed systems
  + Increase number of instances

### High Availability

* Running app in at least two AZ
* Goal = survive failures
* Can be active or passive

### EC2

* VM
* Elastic Block Storage (EBS)
  + Virtual disk for the VM
* Elastic Load Balancer
  + Distributes load across VMs
* Auto Scaling Group
  + ASG
  + Auto scale up or down VMs

#### EC2 Instance Metadata

* Allows ec2 instances to learn about themselves without using a role
* URL = <http://169.254.169.254/latest/meta-data>/
  + Only works from ec2 instance
* Retrieve IAM Role Name but not content

### ECS

* Elastic Container Service
* Securing
  + Can only apply one IAM role to a Task Definition
  + Task Definition
    - Required to run Docker containers in AWS
    - Attach IAM role (task role) to task definition for permissions to services
* Runs on EC2 instances unless using fargate launch method
  + Fargate requires the use of Elastic Container Registry

### EBS Volume

* Terminating EC2 instance deletes the root volume
* Keep main data on attached volumes separate from root
* Data persists even when instance is terminated
* Can only be attached to an instance in the same AZ
* Can be moved
* Privisioned specify size and IOPS
* Drive can be increased over time
* EBS types
  + GP2
    - SSD
    - Balanced
    - Boot volume
    - Recommended for most workloads
    - Least expensive
    - 1 GB – 16 TB
    - 3 IOPS per GB
    - If lower that 1000 IOPS it can burst to 3000
    - Increase GB to increase IOPS
  + IO1
    - SSD
    - High performance
    - Mission critical low latency high throughput apps
    - Boot volume
    - If you need more than 16000 IOPS
    - 4 GB to 16TB
    - Provisioned IOPS
    - Max 32000
    - 50 IOPs per GB
  + ST1
    - HDD
    - Frequent access throughput intensive
    - Streaming workloads
    - Cannot be boot
    - 500GB to 16TB
    - Max IOPS of 500
  + SC1
    - HDD
    - Less frequent accessed workloads
    - Max IOPS 250
    - Slower and cheaper that ST1

### Instance Store

* Ephemeral storage
* Drive physically connected to VM
* Benefits
  + Better IO performance
  + Good for cache
  + Data survives reboots
* Cons
  + Stop or termination data is lost
  + You can’t resize
  + Can’t snapshot

### Load Balancing

* Device sitting in front of application or servers to distribute traffic
* Expose single DNS access point to application
* Handles failures of downstream servers
  + Health checks to make sure server is up
    - Lets ELB know which server is working correctly
    - 200 = OK
    - 4xx = client errors
    - 5xx = application errors
* Security groups can block ELB access.
* SSL termination for websites
  + Cert lives here
  + AWS Certificate Manager – creastes and manages certs
  + You can upload own cert
  + HTTPS Listener
    - Must specify default cert
    - Clients can use Server Name Indication (SNI) to reach specific hostname
      * Allows for multiple certificates on one load balancer
      * Optional list of certs to support multiple domains
    - Can add a security policy to support older versions of SSL/TLS on older browsers
* Enforce stickiness
  + Session will always connect to same server
  + Same client will always be redirected to same instance
  + Classic and application load balancers
  + Uses cookies
    - Has expiration date
  + Ensure users do not loose session data
  + My induce imbalance
* Across AZ
* Separate public and private traffic
* All load balancers have static hostname.
* Perfect Forward Secrecy
  + Provides additional safeguards against eavesdropping of encrypted data through use of unique random session key
* Can scale but not quickly

#### Application Load Balancer

* Exposes URL
* Layer 7
  + HTTP/HTTPS level
* Multiple applications on same machine
* Load balance by route in URL
* Load balance by Hostname in URL
* Microservices and container based apps
* Can redirect to any dynamic port on backend (port map)
  + Good with container services
* Target group
  + Group of services that are load balanced by ELB
* Benefits
  + Stickiness
    - Enabled at target group level
    - Handled by ELB not application
  + HTTP/HTTPS and web sockets
  + Application servers don’t see the IP of client directly
    - Must use X-Forwared-For to insert the IP into the header seen by app
* Call via DNS
* Dynamic Port Mapping
  + Allow multiple tasks from the same service can use the Load balancer
* Path-based routing
  + Redirects traffic to correct service

#### Classic Load Balancer

* Older style
* Layer 4/7
* Recommended not to use (Depricated?)

#### Network Load Balancer

* Exposes IP Address
* Layer 4
  + TCP traffic
* High performance millions request per secont
* Static or elastic IP
* Low latency than Application LB
* Application sees client IP
* Must attach elastic IP with public facing
* Call by static IP
* Cross zone load balancing
* SSL termination

### Auto Scaling Group

* Automatically scale out / in
* Ensure you have min / max EC2 instances to handle load
* Auto register new instances to a load balancer
* Concepts
  + Launch Configuration
    - All info needed to create a new instance of the web app
    - AMI+instance type
    - EC2 Userdata
    - EBS Volumes
    - Security Groups
    - SSH Key Pair
    - Immutable
      * Cannot be updated
  + Min size
    - Number of instances to not go below
  + Max Size
    - Max number of instances to grow to
  + Initial capacity
  + Network and Subnets
  + Load balancer / target group info
  + Scaling policies
    - Auto Scaling Alarms
      * Based on cloudwatch alarms
        + Target CPU usage
        + Number of requests
        + Avg network in
        + Avb network out
        + Custom Metric

Example number of connected users

* + - * Alarm based on any metric
        + Average of the overall ASG instances
      * Can be based on schedule ( if you know usage pattern )
  + IAM roles assigned to ASG are applied to new Instances
  + ASG free only pay for what they spin up
  + ASG will restart instances if it is below specific min value
  + ASG will terminate unhealthy instances and create / replace with new
* ASG Default Termination Policy
  + AZ that has the most number of instances
  + Delete the one with oldest launch configuration
  + ASG tries to balance number of instances across AZ
* Scaling Cooldown
  + Ensures no other scaling can happen until cooldown is complete
  + Can modify
  + Lets things level out before adding or removing more instances
* Instance Protection
  + Prevents ASG from terminating specific Instance

### Elastic BeanStalk

* Deploy app template in AWS
* Can deploy a custom AMI
* Managed service
  + Instance config / os handled by AWS
  + Deploy ment strategy configurable but performed by EWS
* Three modles
  + Single instance
    - Dev
  + Load Balancer and Autoscaling group
    - Pre production web apps
  + Autoscaling group
    - Non web apps
* Components
  + Application
  + Application version
  + Environment name
* Deploy app version to environments and can promote application version to next environment
* Rollback feature to previous version
* Full control over lifecyle of environment
* Support for platforms
  + Go
  + Java
  + Java with tomcat
  + .net
  + Node js
  + Etc
* Can write own custom platform

# Domain 2: Define Performant Architectures

## 2.1 Choose performant storage and databases.

### Storage

#### S3

##### S3 Buckets and Objects

* Buckets = folders
* Objects = files
* Globally unique name even tho defined per region
* Naming Convention
  + No uppercase
  + No underscore
  + 3-63 characters long
  + Not an IP
  + Must start with lower case letter or number
* Objects
  + File
  + Identified by key
    - Full path to object
  + Max size 5 TB
  + Metadata
    - Key value pairs
  + Tags are allowed
  + Version ID if enabled
* Multipart upload
  + Uploading objects larger than 5GB require multipart upload
  + Recommended
    - Large objects over stable high bandwidth
      * Parallel uploads
    - Uploading over spotting network
      * Prevents restarting from beginning. Will only need to restart the part that failed.
* Buckets are flat
* But it looks like there are directories
* Use cases
  + Static files
  + Key value for big files
* Well-Architected
  + Operations
    - No operations
  + Security
    - IAM
    - Bucket Policies
    - ACL
    - Encryption
  + Reliability
    - Multi AZ
    - Cross region replication
  + Performance
    - Scales to 1000s read / write
    - Multipart big files
  + Cost
    - Pay for storage used
    - Network cost transfer out
    - requests

##### S3 Versioning

* Enabled at the bucket level
* Overwriting file increments it version
* All versions retained
* Best practice
  + Protect against unintended deletes or bad edits
* Version = null if file was not versioned prior to enable versioning

##### S3 Encryption

* Methods of encrypts
  + SSe-S3
    - Encrypts s3 objects using keys handled and managed by AWS S3
    - Server side
    - AES 256
    - You will not see the keys
    - Must set header when sending info to S3
      * “X-amz-server-side-encryption”:”AES256”
  + SSE-KMS
    - Leverage AWS KMS to manage encryption keys
    - Server side
    - Key managed by KMS
    - More control over KMS key
    - Must set header
      * “X-amz-server-side-encryption”:”aws:kms”
  + SSE-C
    - Manage your own keys
    - Server side
    - Keys fully managed by customer on prem
    - HTTPS must be used
    - Encryption key must be provided in https header for every request
  + Client Side Encryption
    - Encrypted on client side prior to uploading to S3
    - Library such as amazon S3 Encryption client
    - Client must encrypt and decrypt on prem
* Encryption in transit (SSL)
  + HTTP endpoint not encrypted
  + HTTPS endpoint encrypted in flight
* Can be applied to each object individually as it is uploaded
* Can be set as the bucket default
* Bucket policies are evaluated before default encryption policies

##### S3 Security and Bucket Policies

* Types
  + User based
    - IAM policies
      * Which API calls should be allowed for a specific user from IAM console
      * Explicit deny takes precedent over bucket policies
  + Resource based
    - Bucket policies
      * Bucket wide rules from S3 console
      * JSON based policy
      * Resources
        + Buckets and object policy applies to
      * Actions
        + Set of API to Allow or deny
      * Effect
        + Allow or deny
      * Principal
        + Account or user to apply the policy to
      * Use cases
        + Grant public access to bucket
        + Force objects to be encrypted at upload
        + Grant access to another account

Cross account

* + - Object access control list
      * Finer grained
    - Bucket Access control List
      * Less common
  + VPC Endpoints
    - Instances in VPC without internet access can still connect to S3
  + Logging and Audit
    - S3 access logs can be stored in another S3 bucket
    - API calls can be logged in AWS Cloud trail
  + User security
    - MFA can be required in versioned buckets to delete object
    - Signed URL
      * Valid for limited time

##### S3 Websites

* S3 data can be deployed as static websites
* URL is the bucket addres
  + <bucket-name>.s3-website[.|-]<AWS-region>.amazonaws.com
* 403 error on setup means the bucket policy does not allow public reads
* Custom names can be used for websites
  + Bucket must be named the same as the custom name
* HTTP only (not HTTPS)

##### S3 CORS

* Cross Origin Resource Sharing
* Limit the number of websites that can request your files in S3
  + Limits your cost
* When an HTML from one bucket wants to get object from another bucket, with CORS enabled, the ORIGIN of get must be included and allowed.

##### S3 Consistency

* S3 is global service
* Read after write consistency for PUTS of new objects
  + As soon as object is written it can be read
  + True if you have not done a GET to check if object exists
    - Makes it eventually consistent.
    - Must wait
* Eventual Consistent for DELETES and PUTS of existing objects
  + If read after update, the older object may still be returned
  + Delete object, may still be able to GET object

##### S3 MFA Delete

* Forces a user to generate MFA code on device before doing Important S3 Operations
  + Permanently delete object version
  + Suspend versioning on the bucket
* Enable versioning
* Only bucket owner can enable / disable MFA Delete
* MFA Delete must be done via CLI

##### S3 Access Logs

* You can log access to another S3 bucket
* Can be analyzed (Athena)

##### S3 Cross Region Replication

* Asynchronous replication S3 from one region to another
* Must have Versioning enabled
* Can be different accounts
* Proper IAM permissions to S3
* Use Cases
  + Compliance
  + Lower latency by replication to user location
  + Replication across accounts

##### S3 Pre-signed URLs

* Can generate using SKD or CLI
  + for downloads CLI is easiest
  + for uploads Must use SDK
* expires after 3600 seconds (1 hr)
* user inherits permissions from user who created pre-signed URL
* Use Cases
  + Allow only logged in users to download premium video from S3 Bucket
  + Allow ever changing list of users to download files by generating URLs dynamically
  + Allow temporary user to upload a file to a precise location in bucket

#### EBS

#### EFS

* Elastic File System
* Managed NFS
* Mounted on as many Instances as needed
* Multi AZ
* Highly available ,scalable, expensive
* Pay for what you use not what you provision
* Only works with Linux based Intances
* Use cases
  + Content management
  + Web serving
  + Data sharing
  + Wordpress
* NFS v4.1
* Security groups control access
* Performance Modes
  + General Purpose (Default)
  + Max I/O
    - used when thousand of instances are using the EFS
* EFS File Sync
  + Sync from on-prem to EFS
* Backup EFS to another EFS incrementally
* Encrypted option at rest

#### Snowball

* Physical transport data in and out of AWS
* Alternative to moving over the network
* Up to PetaBytes ( PB )
* 80TB per device
* Pay per job
* Use case
  + Large cloud migrations
  + Disaster recovery
  + Use if it take more than a week over network

##### Snowball Edge

* Adds computational capacity to device
* Storage optimized
  + 100 TB storage
  + 24 vCPU
* Compute Optimized
  + 42TB + SSD
  + 52 VCPU
* Storage optimized with GPU
  + 42 TB + SSD
  + 52 vCPU, GPU
* Can load custom EC2AMI and Lambda functions
* Can perform calculations while moving
* Use
  + Preprocess data while moving

#### Snowmobile

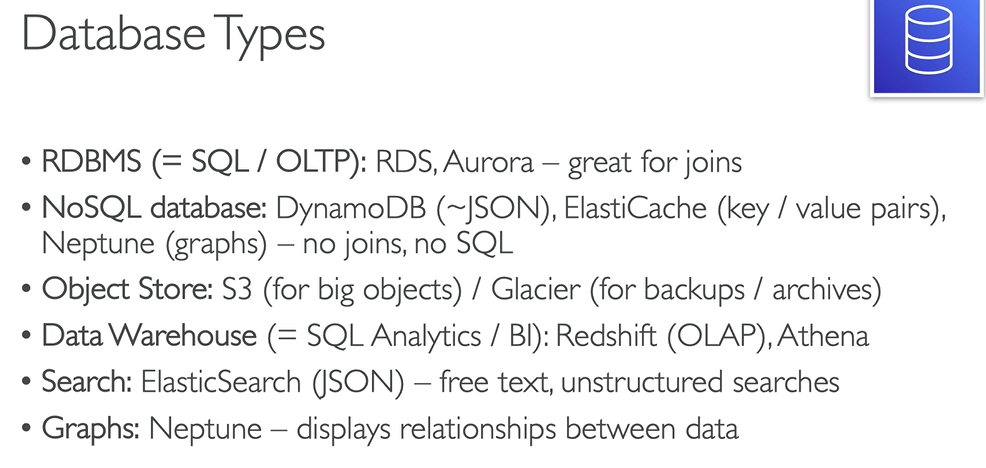
* Exabyte of data
* Container truck
* Better if you transfer more than 10PB per upload

#### S3 Storage Gateway

* Exposes S3 buckets to on-prem servers
* User cases
  + DR
  + Backup restore
  + Tier storage
* Types
  + File Gateway
    - Access via SMB or NFS
    - Each bucket will have its own IAM role
    - Most recently used data will be cached in file gateway
    - Can be mounted on may servers
  + Volume Gateway
    - Block storage
    - Accessed via ISCSI
    - Backed by EBS snapshots
    - modes
      * Cached Volumes
        + low latency access to most recent data
      * Stored Volumes
        + Entire data set on premise scheduled to backup to S3
  + Tape Gateway
    - Virtual tape library
    - Backup ‘tapes’ in the cloud
    - Existing software will backup to tape gateway in S3

### Databases

* Things to consider when determining DB
  + Read heavy, write heavy, balanced
  + Throughput
  + Will these change, does it need to autoscale?
  + How much data will need to be stored, for how long.
  + Object size and will any of these change
  + Durability
  + Latency requirements
  + # of users
  + Data model
    - How will you query
    - Structure
  + Schema
  + License cost



#### Athena

* Serverless
* Combined with S3 they both become a DB
* Analytics directly against S3 files
* Can Use SQL query language
* Charged per query and amount of data scanned
* Supports server and client side encryption
* Use Cases
  + BI
  + Log queries
* Well architected
  + Operations
    - No operations
  + Security
    - IAM + S3 security
  + Reliability
    - Managed service
    - Presto engine
    - Highly available
  + Performance
    - Queries scaled on data size
  + Cost
    - Pay per query
    - Pay per TB of data scanned

#### Aurora

* AWS not open source
* Postgres and mysql
* Cloud optimized
* 5x over SQL
* 3x over SQL
* 6 replicas over 3 AZ
* Backup and recovery
* Industry compliance
* Push button scaling
* Automated patching
* Routine maintenance
* Global Databases
  + Span multiple regions of rDR
    - One primary region
    - Secondary region
      * Lower latency reads
      * Failover
* Cross region read replicas
  + Latency improvement
  + No DR
* Backtrack
  + Point of time without using backups
* Advanced monitoring
* Auto Scaling Incremental storage grows over time
  + Start at 10 GB
  + Up to 64 TB
* 15 replicas
  + Replication is faster
  + Increase reads
  + Increases availability
* HA native
  + Instantaneous failover
* More efficient but costs more (might decrease with efficiency)
* Aurora Serverless option
  + No need to choose size
  + Autoscales
  + Only supports MySQL
  + Postgres beta
  + Helpful if can’t predict workclod
  + Can migrate between serverless and not
  + Aurora capacity units
    - Billed in 5 minut increments
* Great for joins
* Auto healing
* Multi AZ, auto scaling read replicas
* Global read replicas
* Use cases
  + Leans towards enterprise grade
* Well architected
  + Operations
    - Less operations than RDS
    - Auto scaling storage
  + Security
    - AWS does OS security
    - Client does Encryption, security groups, IAM authorizing
  + Reliability
    - Multi AZ
    - Highly avail with 6 replicas
    - Serverless option
  + Performance
    - 5x performance of RDS
    - 15 Read replicas
  + Cost
    - Per hour based on EC2 and EBS type
    - More efficient so possibly better cost

#### DynamoDB

* Serverless
* Managed no sql
* Proprietary for AWS
* Replicated across 3 AZ by default
* NoSQL
* Not relational
* Automatically Scales to massive workloads
* Distributed database
* Integrated with IAM for security, auth and admin
* Low cost and auto scales. Charged for what you use
* Made of tables
* Defaults to eventually consistent, can force to strongly consistent
* Each table has primary key (decided at creation time)
* Max size of item (row) = 400KB
* Data type
  + Scalar types
    - String, Number ,Binary ,Boolan,Null,
  + Document Types
    - List, Map
  + Set Types
    - String Set, Number Set, Binary Set
* Provisioned Throughput
  + Table must have provisioned read and write capacity units
  + Read capacity (RCU)
    - Throughput for read
    - 1 RCU = 1 strongly consistent read for 4 KB per second
    - 1 RCU = 2 eventually consisten read of 4KB per second
  + Write Capacity ( WCU )
    - 1 WCU = 1 write of 1 KB per second
  + option to set up auto-scaling of throughput
  + throughput can be exceeded using burst credits
    - if burst credits are empty then error ProvisionedThroughputException
      * Fix with Exponential back-off retry
* Backup and restore
  + Point in time backups like RDS
  + No performance impact
* Global Tables
  + Replicated everywhere
  + Low latency
* DMS
  + DB migration service
  + Helps with migration
* Can launch local DynamoDB on own computer to dev
* User Cases:
  + Serverless app dev
  + Distributed serverless cache
  + Doesn’t use SQL queries
  + Transactional capability
* Well-Architecture
  + Operations
    - No operations needed
    - Auto scaling
  + Security
    - IAM policies
    - Encryption
  + Reliability
    - Multi AZ
    - backups
  + Performance
    - Single milli second performance
    - Does not degrade when app scales
  + Cost
    - Pay per provisioned capacity and storage usage

##### DynamoDB Advanced Features

###### DAX

* DynamoDB Accelerator
* Cache
* Write to DB will go thru DAX
* Micro second latency for reads and queries
* Solves the Hot Key problem of too many reads
* 5 minute TTL for each Item
* Up to 10 DAX nodes in a cluster
* Multi AZ
* Secure

###### DynamoDB Streams

* Changes to DynamoDB can end up in a stream
* Change log
* Can be fed to AWS Lambda to process updated / new data
* Must be enabled for cross region replication
* 24 hour of data retention

###### Transactions

* All or nothing type of operation
* Coordinated instert, update, deletes across multiple tables
* Include up to 10 unique items

###### On Demand

* No capacity planning needed for WCU / RCU
* Scales automatically
* More expensive than provisioned capacity
* Helpful when spikes are unpredictable
* Or low throughput

###### DynamoDB Security

* VPC endpoints can access Dynamo DB without internet
* Access controlled by IAM
* Encryption with KMS and SSL

#### ElastiSearch

* Open source
* You can search any field even partially matched
* Common to use as compliment to DB
* Big data apps
* Can have cluster
* Kibana – visualization
* Logstash – log injection
* Well-architected
  + Operations
    - Small downtime when failover or maintenance
    - Read replica scaling
  + Security
    - Cognito, IAM, VPC, KMS, SSL
  + Reliability
    - Multi AZ
    - clustering
  + Performance
    - Petabyte scale
  + Cost
    - Pay per node

EMR

* Elastic Map Reduce
* Hadoop
* Big Data

#### Neptune

* Fully managed graph DB
* When to use
  + High relationship data
  + Social networking
  + Knowledge graphs (Wikipedia)
* Replicate across at least 3 AZ. Upt 15
* Point in time revovery
* KMS and HTTPS encryption
* Well-architected
  + Operations
    - Small downtime when failover or maintenance
    - Read replica scaling
  + Security
    - AWS does OS security
    - Client does Encryption, security groups, IAM authentication
  + Reliability
    - Multi AZ
    - clustering
  + Performance
    - Graph data
  + Cost
    - Pay for node

#### RDS

* Relational Database Service
* Great for joins
* Managed DB running on managed EC2 instance
* Aws patches OS
* Continuous backups
  + Automatically enabled
  + Daily full snapshot
  + Transaction logs in real time
    - Restore to any point in time
  + 7 days retention (to 35)
  + Manual DB snapshots
    - Retention as long as you want
* Monitoring dashboards
* Read Replica
  + Read scalability
  + Increases read throughput
  + Up to 5 replicas
  + Within or across AZ and Regions
  + Async replication
    - Eventually consistent
    - Small lag between write on master and replicated to replicas
  + Replica can be promoted to its own DB
  + Application must change connection string to leverage read replicas
  + Can only do SELECT statements
* Multi AZ
  + Fault tolerance
  + Synchronous replication
  + Application connects to DNS name which is auto failed over to stanby instance of DB
    - (CNAME is updated )
* Scaling manually without auto scale. Must shutdown to change instance size
* Can’t control RDS EC2 Instance
* Transparent Data Encryption (TDE)
  + Only supported with Oracle or SQL
  + Extra layer of encryption on top of KMS
  + Affects performance
* SQL as query language
* Types of DB
  + Postgres
  + Oracle
  + MySQL
  + MariaDB
  + Oracle
  + SQL
  + Aurora
* Encryption
  + Enable at rest with KMS
  + Replicas are encrypted with same key if in same region. Different key in different region
  + SSL encryption in flight
    - To force
      * PostgreS
        + Rds.force\_ssl=1 in RDS console
      * MySQL
        + Within DB
        + GRANT USAGE on \*.\* TO ‘mysqluser’@%’REQUIRE SSL;
    - To Connect
      * Provide SSL Certificate
      * Connect with SSL
  + Can’t encrypt existing DB
    - Create a snapshot, copy it , encrypt copy then build encrypted DB with this copy
* Security
  + Deployed in private Subnet
  + Security group allow connection
  + IAM policies to help control who can manage
  + Traditional username /password can be used to login to the database
    - Or IAM users
* Backup
  + Snapshot
  + Point in time restore
* Monitor with cloudwatch
* Use Cases:
  + Relational datasets
  + SQL Queries
  + Transactional Inserts
  + Update
  + Delete is available
* Well-architected
  + Operations
    - Small downtime when failover or maintenance
    - Read replica scaling
  + Security
    - AWS does OS security
    - Client does Encryption, security groups, IAM authorizing
  + Reliability
    - Multi AZ
  + Performance
    - Depends on EC2 instance
    - Read replicas
    - Doesn’t auto scal
  + Cost
    - Per hour based on EC2 and EBS type

#### Redshift

* Data warehouse
* Must provision cluster
* Based on Postgres
* OLAP – online Analytical Processing
* Data is loaded from
  + S3
  + Dynamo DB
  + DMS
  + Other DB
* Scales from 1 to 128 nodes
  + 160GB space per node
* Leader node
  + Plans query and aggregate results
* Compute Node
  + Perform query and sends result to leader
* Redshift Spectrum
  + Perform query directly against S3
* Redshift enhance VPC routing
  + Copy / unload goes thru VPC not internet
* Use Cases
  + Long complex queries
  + Repeat queries
  + Dashboard
  + Visualization
  + Business Intelligence
  + Analytics
* Well-architected
  + Operations
    - Small downtime when failover or maintenance
    - Read replica scaling
  + Security
    - AWS does OS security
    - Client does Encryption, security groups, IAM authorizing
  + Reliability
    - Highly avail cluster
    - Auto healing
  + Performance
    - 10x perf over other datawarehouse
    - Can use compression
  + Cost
    - Pay per node provisioned

### CloudFront

* Improves read performance
* Content is cached at edge sites
* Query string parameters
  + Forward query strings to origin
* Works with
  + S3
  + EC2
  + Load Balancing
* Protects against network attacks
* SSL encryption
  + Force to use – redirect HTTP to HTTPS
* Supports RTMP protocol
  + Video and media protocol
* Restrict Bucket Access
  + Origin Access Identity
    - Allows access to S3 bucket
    - Can also deny any other users exept this one
    - Can automatically create a S3 Bucket Policy for the ID

#### CloudFront Signed URL

* If S3 can only be accessed via cloudfront you can’t use S3 pre-signed URL
* Attach policy
  + URL expiration
    - Shared content should be short
    - Private can be long
  + IP range access the data
  + Trusted signer
    - Who can create signed URL
  + Must use AWS SDK

#### Cloud Front vs Cross Region Replication

* Great for static content
* S3
  + Must be set up for each region
  + Read only
  + Dynamic content, low latency in few regions

#### Cloudfront GEO restriction

* Restrict who can access based on GEO location (country)
* White list
  + Who can access
* Blacklist
  + Prevent these regions from accessing
* Region is determined by 3rd party geo-ip database to figure out country

## 2.2 Apply caching to improve performance.

### Instance Store

* Ephemeral storage
* Local to Instance
* Used for caching
* Very fast
* Lost when instance stopped
* Retained during reboot

### Aurora

* Parallel query
  + Faster queries

### Elastic Cache

* Managed Redis or Memcached
* In memory DB with high performance low latency
* Can be placed in one region to cache data from another region
* Read from cache not HDD
* Write scaling
  + Sharding
* Read replicas
* Multi AZ
* Place between app and RDS
  + Releave DB load
* User Session Store
  + Write user session data to elasticache
  + User hits another instance of app whcich checks cache to make sure user is logged in
  + Makes app stateless
* IAM authentication not supported
* Patterns
  + Lazy loading
    - All read data is cached
    - Data can become stale
  + Write Through
    - Adds or update data in cache when written to DB
    - No stale data
  + Session Store
    - Store temp session data
    - Using TTL feature
* Redis
  + In memory Key-Value store
  + Super low latency
  + Survives reboots
    - Persistent
  + Use cases
    - User sessions
    - Leader board
    - Distributed states
    - Releave pressure of DB
    - Pub / sub capability for messageing
  + Multi AZ Auto failover
  + Read replicas
  + Redis auth
    - Authenticate to Redis
    - Requires SSL
* MemCached
  + In memory object store
  + Cache doesn’t survive reboots
  + Use cases
    - Quick retrival of objects from memory
    - Cache often accessed objects
  + Not as popular as Redis
  + SASL authentication
* Use Case:
  + Key value store in memory
  + Frequent reads less writes
  + Cache DB queries
  + Store session data
  + Cannot use SQL
* Well-Architected
  + Operations
    - Small downtime when failover or maintenance
    - Read replica scaling
  + Security
    - AWS does OS security
    - Client does Encryption, security groups,
  + Reliability
    - Multi AZ
    - Cluster
  + Performance
    - Sub milisecond
    - Read replicas
    - cache

Cost

* + - Per hour based on EC2 and storage used

## 2.3 Design solutions for elasticity and scalability.

### Aurora

* High Available
  + 6 copies of your data across 3 AZ
    - Only needs 4 copies for writes
    - Only 3 copies for reads
    - Self healing peer-to-peer replication
    - Storage is striped across 100s of volumes
  + Only one master to take writes
  + Master up to 15 read replicas
  + Support cross region replication
  + Quick instantaneous failover
  + Writer Endpoint
    - DNS entry that auto connects to Master replcia
* Scalability
  + Autoscales to 64 TB
  + Number of read replicas can autoscale up to 15
  + Reader endpoint
    - DNS entry that connects automatically to read replicas
  + Scales on connections or CPU usage

# Domain 3: Specify Secure Applications and Architectures

### IAM

* Users
  + Physical person
  + 1 per user
* Groups
  + Groups of users
  + Functions etc
  + Contains users
* Roles
  + Internal usage within AWS
  + Roles assigned to machines
  + 1 role per application
    - To add another permission, you must add all permissions to one role
* Policies
  + JSON
  + Define what users / groups / roles can or cannot do
  + There are predefined policies
  + Policies consist of
    - Service
      * Which servers this policy refers to
    - Action
      * Allow or deny
    - Resource
    - Request Condition
  + Inline Policy
    - Added on top of other roles
    - Cannot be added to other roles
* Root Account
  + First account when create AWS account
  + Most power
  + Do not use except for billing
* IAM is global
* MFA
  + Multi factor authentication
* Least privileges principle
  + Only give permissions they need
* IAM Federation
  + Integrate their own repository of users with IAM
  + SSO with company credentials
  + SAML standard
* Never write IAM credentials in code

### Cognito

* Gives users Identity to interact with our apps
* Cognito User Pools
  + Sign in functionality for app users
  + Integrate with API Gateway
  + Serverless DB of users for mobile apps
  + Simple login
  + Possible to verify emails / phone numbers
  + MFA
  + Can use Federated Identiies
    - Facebook
    - Google
    - Etc
  + Sends back JSON web token (JWT)
    - Verifies ID of user
* Cognito Identity Pools (Federated Identity)
  + Provide AWS credentials to users so they can access AWS resources directly
    - Direct access to resources
  + Integrate with Cognito User pools as an identity provider
  + Get Temp AWS credentials from Federated ID Pool
    - Leverages STS for the temp credentials
  + Has IAM policy attached
* Cognito Sync
  + Depricated in favor of AppSync
  + Sync data from device to Cognito
  + Store preferences, configuration and state of app
  + Cross device sync
  + Offline capability and sync when back online
  + Requires Federated Identity pool in Cognito
  + Stored in datasets up to 1 MB
  + Max 20 Datasets

## 3.1 Determine how to secure application tiers.

### SSH

* Ssh -I keyname.pem ec2-user@ipaddress
* Must have Security group attached to instance with port 22 open
* Warning: Unprotected Private Key File
  + EC2 Instances require key pairs downloaded to SSH into them
  + When first downloaded the permissions are not secure enough for use
    - Permission 0644 are too open
  + Fix
    - Chmod 0400 keyname.pem
    - In Windows - Set security on pem file
      * Advanced
      * Owner = yourself
      * Remove all other users except yourself (disable inherited permissions to remove those)
    - Yourself should have full control

### Putty

* Puttygen to convert downloaded keyname.pem to ppk
* Putty hostname = ec2-user@ipaddress
* Connection – auth – browse to PPK file
* Save
* Must have Security group attached to instance with port 22 open

### EC2 Instance Connect

* Must have Security group attached to instance with port 22 open
* Only works with the most recent Amazone 2 Linux AMI (not sure about windows)

### Monitoring

#### Cloudwatch Metics

* Variable that cloudwatch monitors
* Grouped in namespaces
* Dimension is a attribute of the metric
  + Up to 10 per metric
* Timestamp
* EC2 Detailed monitoring
  + Changes sampling of Instances from 5 minutes to 2 minutes
  + Faster auto scaling
* EC2 memory usage is by default not pushed
  + Must be set up inside instance as a custom metric
* Custom metrics
  + Define and send your own metrics to cloudwatch
  + Includes dimensions
  + 1 minute resolution ( sampling )
  + Up to 1 second
  + Must install cloudwatch agent on Instance
  + Memory metrics are custom metrics
* PutMetricData
  + Sends to cloudwatch
* Exponential back off to retry throttling errors

#### Cloudwatch Dashboards

* Access to key metrics
* Global
* Can include graphs from diff regions
* Chang change timezone
* Auto refresh
* Pricing
  + 3 dashboards up to 50 metrics free
  + 3/ dashboard/month

#### Cloudwatch Logs

* Apps can send logs to cloudwatch using SDK
* Can collect logs from
  + Elastic beanstalk – logs from app
  + ECS – container logs
  + AWS lambda – function logs
  + VPC Flow Logs – specific logs
  + API Gateway
  + CloudTrail based on filter
  + Cloudwatch log agents
  + Route 53
* Can send log
  + S3
  + Stream to elastisearch
* Log group
  + Represent app
  + Named what ever
* Log stream
  + Instances within app / log / container
* Log expiration policy
  + How long to keep log files
* Use AWS CLI to tail CloudWatch logs
* IAM permissions
  + Permission for service to write to cloudwatch log
* Filter expressions
  + Used during search
* Metric filter
  + Used to trigger alarms
* Cloudwatch log insights
  + Query logs and add queries into dashboards

#### Cloudwatch Alarms

* Used to trigger notifications
* Can be used to autoscal, actions sns notifications
* States
  + OK
  + Insufficient data
  + Alarm
    - Bad
* Period
  + Length of time to evaluate

#### Cloudwatch Events

* Schedule
  + Trigger events at schedule
* Event pattern
  + Trigger if an event happens

### Audit

#### Cloudtrail

* Governance, compliance and audit
* Enabled by default
* History of events and api calls
* Can put logs from cloudtrail into Cloudwatch logs

## 3.2 Determine how to secure data.

### Aurora

#### Aurora Security

* Encryption at rest KMS
* Auto backups, snapsos and replicas encrypted
* Encryption in flight with SSL
* IAM authentication
* Your responsible
  + Protecting instance with security group
* Can’t ssh

### AWS Shared Responsibility Model

* AWS Responible for
  + Secure of the Cloud
  + Infrastructure, HW, SW Datacenters
  + Managed services
* Client
  + Security IN the Cloud
  + Guest OS – EC2
  + FW – security group
  + Network configuration
  + IAM
  + Data permissions

### Encryption

#### Encryption in Flight ( SSL )

* HTTPS
* Data encrypted before sending and decrypted after receiving
* SSL Certs
* Prevents Man in the middle attack

#### Encryption at Rest

* Server side encryption
  + Data is encrypted after being received at the server
  + Decrypted before being sent back to user.
  + Encrypted with a key
  + Key must be managed somewhere and the server must have access
* Client side Encryption
  + Data encrypted by client
  + Server can’t decrypt
  + Can leverage envelope encryption
    - Encrypting data keys
    - AWS KMS API

### EC2

#### EBS Snapshot

* Incremental only backup what has changed since last snapshot
* Use system IO so do not run during peak hours
* Stored in S3 (but won’t see them)
* Do not need to detach volume but recommended to take snapshot
* 100000 snapshots per volume
* Can copy snapshot across region or AZ
* Can make AMI from snapshot
* Prewarm
  + FIO or DD command to prewarm prior to restore snapshot
* Amazon Data Lifecycle Management
  + Automate snapshots

#### EBS Migration

* EBS volumes locked to AZ
* To migrate
  + Snapshot
  + Copy to diff AZ

#### EBS Encryption

* When you encrypt EBS Volume
  + Data at rese is encrypted
  + Data in flight between volume and instance is encrypted
  + All snapshots are encrypted
  + Volumes created from the snapshot are encrypted
* Encryption minimal impact on latency
* Leverages AWS KMS AEM-256 keys

#### EBS RAID

* Software RAID
* EBS is already replicated within AZ
* RAID
  + More IOPS
* RAID 0
  + Increased performance
  + Striped
  + Higher risk
* RAID 1
  + Mirror
  + Increase fault tolerance
* RAID 5
  + Not recommended for EBS
* RAID 6
  + Not Recommended for EBS

### KMS

* Key management service
* Manages encryption keys
* Fully integrated with IAM authorization
* AWS integrated
* Can use CLI / SDK to encrypt on client side
* Cannot retrieve customer managed Key (CMK) as it is managed by KMS. App can use it but we can’t see it
* Key can be rotated in KMS
* KMS can only encrypt up to 4 KB of data
  + If data is > 4KB must use envelope encryption
* Give access to KMS
  + Key policy must allow
  + IAM Policy allows the API calls to KMS
* Fully manage keys
  + Create
  + Rotate
  + Disable
  + Enable
* Audit key usage (cloudtrail)
* Three types of Customer Master Keys (CMK)
  + AWS Managed Service Default CMK : Free
  + User Key created in KMS : 1$ per month
  + User keys imported from outside AWS : 1$ per month
* Pay for calls to KMS
* How does it work
  + API
    - Encrypt
    - Decrypt
  + Migration
    - Throught snapshot / backup
    - EBS volumes
    - RDS
    - Elasticache
    - EFS
  + In-place encryption
    - S3

### RDS

#### RDS Security

* Encryption at rest
  + Can only be enabled during creation of DB
  + Migrating to encrypted DB
    - Snapshot
    - Copy snapshot
    - Encrypt copy
    - Create DB with Encrypted Copy
* Your responsibility
  + Check ports /ip/security groups for DBs
  + In DB permission / users
  + With or without public access
  + Parameter groups
    - SSL connections enable/disabled
* AWS Responibility
  + They SSH
  + They manual DB patching
  + THey manual OS patching
  + They audit underlying instance no way for you to do this
* IAM authentican can be uses vs Username / password
  + Only supported with Postgres and MySQL
  + Lifespan of IAM authentication token is only available for 15 minutes
  + AWS credentials generate tokens
  + Must use SSL
  + Use case
    - EC2 instance with IAM Role to connect to RDS Database

### SSM Parameter Store

* Securely store configuration and secrets
* Optional seamless encryption using KMS
* Serverless
* Scalable
* Durable
* Easy SDK
* Free
* Version tracking of configurations and secrets
* Configuration management using path and IAM
* Notifications with Cloudwatch events
* Integration with cloudformation
* Path = Hierachy similar to folder structure

### STS

* Security Token Service
* Allows you to grant limited and temp access to AWS resources
* Tokens valid up to 1 hour
* User Cases
  + Cross account access
    - Define an IAM Role for another account with permissions to access what they need
    - Define which accounts can access this IAM role
    - AssumeRole API -- Use AWS STS to retrieve credentials and impersonate the IAM role you have access to
    - Receive temp creds for 15 – 1hour
  + Federation with third party providers / Cognito
    - Provide non AWS user with access without IAM Creds
    - Users assume identity provided by access role
    - Allow SSO to log into AWS console without IAM creds
    - Web and mobile apps
    - Third party auth can be
      * LDAP
      * AD
      * SSO
      * Open ID
      * Cognito
        + Makes use of facebook / googl/amazon etc accts
        + Federation Identity pool

Predefined IAM profiles for access

* + - * SAML
        + AD
        + Or other SAML 2.0 account management
        + Trades SAML account for temp AWS Credentials
      * Custom Identity broker Application
        + Custom code that validates local accounts
        + Use only if ID provider is not compatible with SAML 2.0
    - Using federation does not need IAM user accounts

## 3.3 Define the networking infrastructure for a single VPC application.

### Security Group

* Fundamental of network security
* Connected to EC2 instances
* Controls inbound and outbound traffic
* Outbound all allowed by default
* Authorized by IP
* Can be attached to multiple instances
* An EC2 instance can have multiple security groups
* Security groups are locked down to Region / VPC
* Timeout = port probably not open
* Connection refused
  + Service / application not running
* Reference security group from other security groups
  + Allows traffic the is allowed thru one security group to pass this one.

### VPC

* Virtual Private Cloud
* Default VPC
  + New instances launch here by default
  + Has public IP
  + Has internet connectivity
  + Public and private DNS
* Max 5 per region ( can have that increased)
* 5 CIDR addresses per VPC
  + Min = /28
  + MAX = /16
* Only private IP ranges are allowed
* Your CIDR IPs should not overlap with your other networks ( like your corporate network )
* AWS reserves 5 IPs per subnet
  + First 4
    - Network address
    - Reserverd for VPC router
    - Reserved for DNS
    - Reserved for future
  + Last 1
    - Reserved Broadcast
  + Not available for use

#### Bastion Host

* Used to SSH into private instances
* Lives in public subnet
* Best practice is to strongly secure host
  + Only 22 from the IP you need

#### CIDER

* Used for security groups rules
* Defines subnet mask and number of hosts in a range
* IPv4

#### Direct Connect

* Dedicated private connection from a remote network to your VPC
  + Connection must be setup between your DC and AWS Direct Connect Location
* Still need to set up Virtual Private Gateway on VPC
* Access public and private services
* Use cases
  + Increase bandwidth
    - Large datasets
    - Lower cost
  + More consistent network experience
    - Real time data feeds
  + Hybrid environment
* Supports IPv4 and 6
* Direct connect Gateway
  + Set up direct connect to one or more VPC in many diff regions (same account)
  + Does not replace VPC Peer
* 1 Gbps and 10 Gbps

#### DNS Resolution

* EnableDNSSupport
  + Default true
  + Decide if DNS resolution is supported for VPC
  + If true then dns server will be queried
    - 169.254.169.253
* EnableDNSHostName
  + False by default for newly create VPC
  + True by default for default VPC
  + Won’t do anything unless EnableDNSSupport = True
  + If True then assign public hostname to public ec2 instance
* If custom DNS domain names in private zone in route53, you must set both these to true

#### Elastic IP

* ‘static’ public ip for EC2 instance
* Attach to one instance at a time
* Can be remapped to another instance
* Can only have 5 Elastic IP per account.
* Avoid using elastic IP as poor architectural design
* Use DNS with random public IP

#### VPC Endpoints

* Keep traffic internal to AWS
* VPC to AWS Services
* Allows you to access services
* Removes the need for internet gateway to access services
* Can be used to connect to services in another region or in another account
* Types
  + Interface
    - Provide ENI as entrypoint
    - Security group for FW
  + Gateway
    - Provision target
    - S3 and DynamoDB
    - Requires routetable

#### Flow Logs with Athena

* Capture information about IP traffic
* Types
  + VPC flow log
    - All traffic in VPC
  + Subnet Flow log
    - All traffic in specific subnet
  + Elastic Network Interface (ENI) Flow log
    - One interface traffic
* Helps monitor and troubleshoot
* Flow logs can be saved in S3 or cloudwatch log
* Syntax
  + Srcaddr and dstaddr
    - IP
  + Action
    - Success or failure
    - Helps with Secgru or NACL
* Analyz usage patterns or malicious data
* Query flow logs with Athena or cloudwatch log insights

#### Internet Gateway

* Connect VPC with internet
* Scal horizontily
* HA and redundant
* Created separately
* 1 VPC attached to 1 Gateway.
* Gateway is also a NAT for the instances that have a private IPv4
* On their own they do not allow internet access
  + Route tables must be added

#### Egress Only Internet Gateway

* IPv6 only
* Same as NAT (which is for IPv4)
* IPv6 are all public
* This service prevents IPv6 from being public accessable from internet
* Need to edit Route Table

#### NAT Instances

* Old but still on exam
* Allow private subnets to connect to internet
* Must be lauched in the public subnet
* Must disable ec2 flag
  + Source/destination check
* Must have elastic IP attached
* Route table must be configured to go from private to nat instance
* Not HA or redundant automatically
* Depends on EC2 instance for performance
* Must manage security groups and rules

#### NAT Gateway

* AWS managed NAT
* Hight bandwidth
* Beter availability
* Pay by hour for usage and bandwidth
* Created in specific AZ and uses an Elastic IP
* Cannot be used by an instance in the same subnet as itself
* Requires Internet Gateway
* 5 Gbps of bandwidth scaling to 45
* No secgru to manage

#### Network ACL

* NACL
* Firewall Applied to VPC Subnet level
* Stateless
  + Outbound rule must be evaluated
  + Return rule has to exist and be allowed
* Default
  + All inbound and outbound
* One per subnet
* Define rules
  + Number
    - Evaluates low numbers first and stops when matched
  + Last rule = \*
    - Deny any thing not specified
* New NACLs deny all
* Use case
  + Block specific IP at subnet level
* Allow and deny rules

#### Peering

* Allows you to connect two VPC privately and make them behave like they were in the same VPC
* Can’t have overlapping subnets
* Not transitive
  + If a can talk to b and b can talk to c. a can’t talk to c without its own peer.
* Requires route table to be updated
* Can peer between accounts
* Can peer between regions

#### Public IP

* Servers can talk to each other over internet
* Can be seen on internet
* Must be unique in internet
* Stop and start instance can change the public IP

#### Private IP

* Non routable IP addresses
  + 192.168.0.0/16
  + 172.16.0.0/12
  + 10.0.0.0/8
* Only can talk within private network
* Use internet gateway to access internet

#### Routing Table

* Provide route to internet

#### Security Group

* Firewall applied to Instance
* Stateful
  + Inbound request passes then outbound will pass
    - Returns allowed
* Allow rules only
* Evaluates all rules

#### VPN

* Connects on-prem Data center to VPC
* Customer Gateway
  + Software or hardware at datacenter
  + IP Address
    - Static public IP
    - Public IP of Datacenter NAT
* VPN Gateway
  + Service in VPC to connect to customer gateway
  + VPN Concentrator
  + Virtual private gateway connected to VPC
  + Customize ASN
* Site to site VPN connection links the two together

# Domain 4: Design Cost-Optimized Architectures

## 4.1 Determine how to design cost-optimized storage.

## 4.2 Determine how to design cost-optimized compute.

### EC2

* Billed by the second

#### Launch Modes

|  |  |
| --- | --- |
| On demand | * Short workloads predictable pricing * Scale up / scale down as needed pay as you go |
| Reserved instances | * 1 or 3 years * Set resources * Pay whether used or not * Cheaper than on demand |
| Convertible reserved instance | * Allows you to change resource |
| Scheduled reserved instance | * Short workload needed for specific times |
| * + Spot | * Bidding on resource * Unused resources * Will be turned off if on demand or reserved need the resources * Up to 90% discount * Can loose if price is higher than what we bid * Batch jobs, big data analysis or resilient workloads * When instance is being taken away you can set the EC2 instances to   + Hybernate   + Terminate   + Stop |
| Dedicated hosts | * Physical ec2 server dedicated to you * Hardware is all yours * Full visibility to hardware * 3 year period * More expensive * Useful for complicated licensing model   + Regulatory compliance |
| Dedicated Instances | * Dedicated hardware but don’t get control of hardware * Can run other instances from same account |

#### Instance Types

|  |  |
| --- | --- |
| R | * Lots of RAM * In memory cache |
| C | * Lots of CPU / Compute * DB |
| M | * Medium / middle ground * Web app * General |
| I | * Good I/O * Disk operation * DB |
| G | * GPU * Video rendering * Machine learning |
| T2/T3 Burstable | * Burstable instances * Up to specific capacity * For short bursts * OK CPU performance * Burst to better performance * Burst credit   + Only allows burst mode if credit exists   + Use credit each time it bursts   + If no burst credit system can burst     - If burst all the time then it prevents bursting   + Cloud watch to see burst credit   + amount of CPU burst credits used per hour and max balance are shown on instance type |
| T2/T3 Unlimited | * Unlimited burst * Unlimited burst credit balance * Pay extra when you burset over your credit balance |

#### Placement Groups

* Control how EC2 instances are deployed within VPC
* Strategies
  + Cluster
    - Instance will be grouped together within a single AZ
    - Same rack, same AZ
    - Can run on same hardware
    - Low-latency
    - Con = if hardware fails, then all instances will fail
    - Use case:
      * Big Data job to complete fast
      * Low latency requirement
  + Spread
    - Spreads across different hardware
    - Minimize failure risk
    - Each instance on different hardware
    - Span across AZ
    - Cons =
      * 7 instances per group per AZ
    - Use Case
      * Max high availability
      * Critical apps
  + Partition
    - Spreads instances across many different partitions( which rely on different sets of racks) within an AZ
    - Max 7 partitions per group
    - Partition failure can affect many but not all
    - Scales to 100s of instances per group
    - Hadoop, Casandra, kafka

#### AMI

|  |  |
| --- | --- |
| Custom AMI | * Template of your prebuilt server |
| AMI Marketplace | * AMI from third parties * Some you have to pay for. |
|  |  |

* Charged for amount of S3 storage
* Since backed by S3 AMIs are regional
* Can be copied to other regions
* Private by default.
* Can be made public
* Create Custom AMI
  + Preinstalled Packages
  + Faster boot time
  + Patches updated

##### COPY AMI

* Sharing AMI does not change ownership unless they copy it to another location then they own the copy
* To copy an AMI you must be granted read permissions for the storage that backs the AMI either EBS snapshot (EBS AMI) or S3 bucket (instance store AMI)
* You can’t copy an encrypted AMI shared with you. If you have encryption key you can copy the snapshot and reencrypt then create new AMI
* Your can’t copy an AMI with associated billing Product (includes Windows AMI or AMI from AMI Market place) code shared from another account
  + You have to launch shared AMI and create AMI from the new running EC2 instance

##### AWS Instance Scheduler

* Configure start/stop schedules for EC2 instances and RDS

# Domain 5: Define Operationally-Excellent Architectures

## 5.1 Choose design features in solutions that enable operational excellence

### API Gateway

* Way to build, deploy, manage API
* Collection of resources and methods integrated with Backend AWS Services
* Throttling
  + Can be configured at multiple levels
    - Global
    - Service Call
* Translates client Rest API requests to Proxy request within AWS
* Handles API versioning
* Handle diff environments (dev, test, qa, prod , etc)
* Auth and authorization
* API keys
* Request throttling
* Swagger / Open API
  + Import to quickly define APIs
* Transform and validate request and responses
* API Cache
  + Cache endpoint responses
  + Can reduce the number of calls to an endpoint improving latency
* Integration
  + Outside VPC
    - Lambda
    - Endpoints on EC2
    - Load balancer
    - AWS Sservices
    - Http Endpoint
  + Inside VPC
    - Lambda
    - EC2 endpoints
* Security
  + IAM Permissions
    - Create an IAM policy authorization and attach to User / Role
    - API Gateway verifies IAM permissions when calling application
    - Goog to provide access within your own infrastructure
    - Leverages SIG v4
    - Great for user / roles already in AWS
  + Lambda Authorizer
    - Uses lambda to validate token in header be passed
    - can cache results
    - helps with 3rd party authentication
      * OAuth
      * SAML
      * Etc
    - Lambda must return an IAM Policy
  + Cognito User Pools
    - Cognito manages user lifecycle
    - No custom lambda functions
    - Cognito only helps with authentication not authorization
    - Backend must provide authorization

### CLI

* Configure CLI
  + Aws configure
    - Add access key / secret key from your IAM account
    - Add region
  + Keep the files it creates safe as they contain in clear txt access key and password
  + Configure CLI on EC2
    - Don’t do it the same way with the access key / secret key
    - Use IAM Roles
      * Combined with policy to allow the EC2 instance to use the CLI with credentials
    - Unable to locate credentials
      * Means credentials (roles) not set up

### SDK

* Software Development Kit
* Allows applications to directly connect to AWS
* If You don’t specify default region it will use us-east-1
* Default Credential Provider Chain
  + Uses /.aws/credentials file to retrieve credentials
  + Instance Profile Credentials
    - If you use a role on your instance
  + Environment variables ( not recommended )
* Never store credentials in code
* Exponential Backoff
  + If API fails because of too many calls needed to be retried
  + Retry mechanism
  + Delays between retries
  + Ensures API is not overloaded