# Exam Breakdown

* Deployment - 22%
* Security - 26%
* Development - 30%
* Refactoring - 10%
* Monitoring & Troubleshooting - 12%

\*\*The developer exam is much more serverless focus, less info on EC2.

\*\* Blue green deployments whitepaper (read)

\*\*Minimum passing score of 720

# S3 - Simple Storage Service

Summary -

* **Object based** storage, NOT block storage. (No databases or OS)
* Files can be from 0 Bytes to 5 TB
* Limited to 100 buckets per account
* Unlimited storage
* **Must have unique DNS name** (all lower case) even though buckets are region specific

Core Fundamentals of S3 Object

* Key Value Store {name: “hi.txt”, data: “Hello World”}
* Version ID (for versioning)
* Metadata
* Subresources - bucket specific configuration
  + Policies, Access Control Lists (IAM stuff)
  + CORS
  + **Transfer Acceleration** - **Use CloudFront to optimize transfers**

Data Consistency

* **Read after write** consistency for **PUTS** of **new** objects
* **Eventual** consistency for overwrites **UPDATES**(PUT existing)and **DELETES** (can take some time to see changes)
* 5TB max for one PUT 5GB max for PUTs

Storage Tiers

* **S3 Standard** - 99.99% availability, 99.(9\*9) durability
* **S3 Infrequently Accessed** - 99.9% availability, Lower fee than S3 but you are charged a retrieval fee, good for files accessed every 1 - 6 months
* **S3 One Zone IA** - Same as IA but only stored in one availability zone 99.5% availability cost is 20% less though
* **Reduced Redundancy** Storage - Less durability 99.99%, designed for data that can be recreated if lost. AWS recos not to use this
* **Glacier** - VERY cheap, used for archiving, takes 3-5 hours to access

Charged For

* Storage (per GB)
* Requests (Get, Copy, Put)
* Storage management (Analytics, Tagging)
* Downloading data

Security

* By default buckets are private
* Bucket policies - applied at **bucket** level
* Access Control lists (ACL) - applied at **object** (file) level
* S3 buckets can be configured to create logs (even to another bucket) for access control audits
* Two ways to access file, using open and clicking link (clicking link will be error if file isn’t public)
* When turned into a website must be http**, can turn into https with cloudfront**
  + Bucket:  https://s3-eu-west-1.amazonaws.com/ankittest
  + Website:  http://ankittestsite.s3-website-eu-west-1.amazonaws.com

**Encryption**

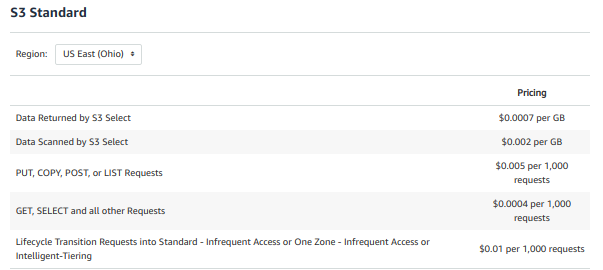
* In Transit - (HTTPS)
* At Rest -
  + SSE-S3 - S3 managed keys (click on object and press encrypt)
    - USING THE API use x-amz-server-side-encryption”:”**AES256**” as an HTTP header
  + SSE-KMS - AWS Key Management Service, additional charge, but has audit trail
    - USING THE API use x-amz-server-side-encryption”:**”aws:kms**” as an HTTP header
  + SSE-C Customer provided keys
  + Client Side Encryption (encrypted before uploaded)
* Enforcing
  + Bucket Policy that denies PUT request if header doesn’t have **x-amz-server-side-encryption AES256**

Tips

* Should Use Multipart Uploading for files bigger than 5 GB
* Objects stored in alphabetical order, so to optimize and spread load across S3 don’t name files similarly (**3,500 put req per second, 5000 get**)
* Users getting **403? Create a bucket policy**
* The versioning state applies to all (never some) of the objects in that bucket

Pricing:

* .023 per GB per month storage



# IAM - Identity & Access Management

Summary - Allows you to manage users and access. IAM is global.

New users begin with NO permissions.

* Users - End user, via console or CLI
* Groups - A collection of users HR, IT, Admins, DBA
* Roles - Used to define set of permissions, can make an EC2-S3 Access role
* Policies - Defines permissions that can be attached to any of the above.

Policies:

* Managed Policies - IAM policy created and managed by AWS (the ones i usually use)
* Customer Managed Policies - Policy that you create and manage
* Inline Policies - Embedded with user, group, or role that it is for. Not reco

Authorization Types:

* Lambda Authorizer - when you want to implement custom auth (uses OAuth/SAML)
* Cognito User Pools
* IAM permissions with sigv4

Ex: Super Admin

{

"Version": "2012-10-17",

"Statement": [{

“Effect”: “Allow”,

“Action”: “\*”,

“Resource”: “\*”

}]

}

Cross-Account Access

* Account A creates an IAM Policy that grants account B a certain level of access. Account B can then give access to its resources up to the amount of permissions it has
* Always give users Least Privilege Access. They give developers Admin :)
* Use Groups and assign perms to groups, not to individual users
* Create one key per developer, not one for all of the developer group
* **Roles are always preferred** instead of Access Key IDs and Secret Keys
  + But roles can only be used in your AWS environment, i.e EC2. Can’t assign a role to my computer
  + Must use AWS Keys (Access Key ID and Secret Key) for programmatic access
* Changing a policy on a role takes immediate effect, no need to restart EC2

Policy Logic

* Explicit Denies override any Allow
* SCP (Service Control Policies) - Specify the max permissions for an org

# STS - Security Token Service

Summary

* Web service that enables you to request temporary, limited-privilege credentials for AWS IAM users that you authenticate (federated users)
* A web service that enables you to request temporary, limited-privilege credentials for AWS Identity and Access Management (IAM) users or for users that you authenticate (federated users). By default, AWS Security Token Service (STS) is available as a global service, and all AWS STS requests go to a single endpoint at <https://sts.amazonaws.com>. NOT supported with API Gateway
* Credentials that are created by using account credentials can range from 900 seconds (15 minutes) up to a maximum of 3,600 seconds (1 hour), with a default of 1 hour. Hence you need to renew the credentials post expiry.

# KMS - Key Management Service

Summary:

* Used to create and control encryption keys
* KMS allows you to centrally manage keys known as customer master keys (CMKs)
* CMKs are protected by HSM (Hardware Security Modules)
* KMS generates **data keys** which are **used to encrypt the data** and then are **encrypted themselves**. (Envelope encryption) Unencrypted data keys are never stored
* Logged in CloudTrail
* Encryption Keys are **REGIONAL**

Usage:

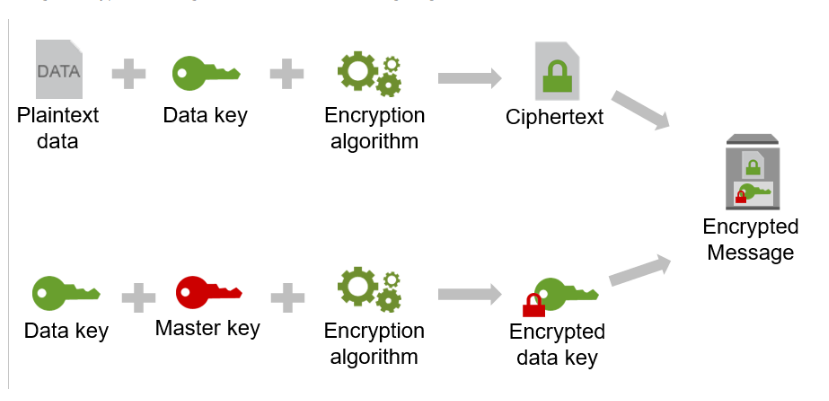
* KMS APIs - directly encrypt/decrypt data using master keys in KMS
* AWS Services can encrypt your data (sometimes as easy as checking a box)
* AWS Encryption SDK that can encrypt it
* MAX SIZE: **4KB,** can’t encrypt anything larger, must use another method

API Calls

* Use key-d for commands
* Aws kms encrypt, decrypt, re-encrypt, enable-key-rotation

You can import your own 256-bit symmetric keys

Envelope encryption is

****

# Cognito

Summary:

* Amazon Cognito lets you add user sign-up, sign-in, and access control to your web and mobile apps quickly and easily
* Can use Google, FB, Amazon using AD and SAML

User Pools (Directory):

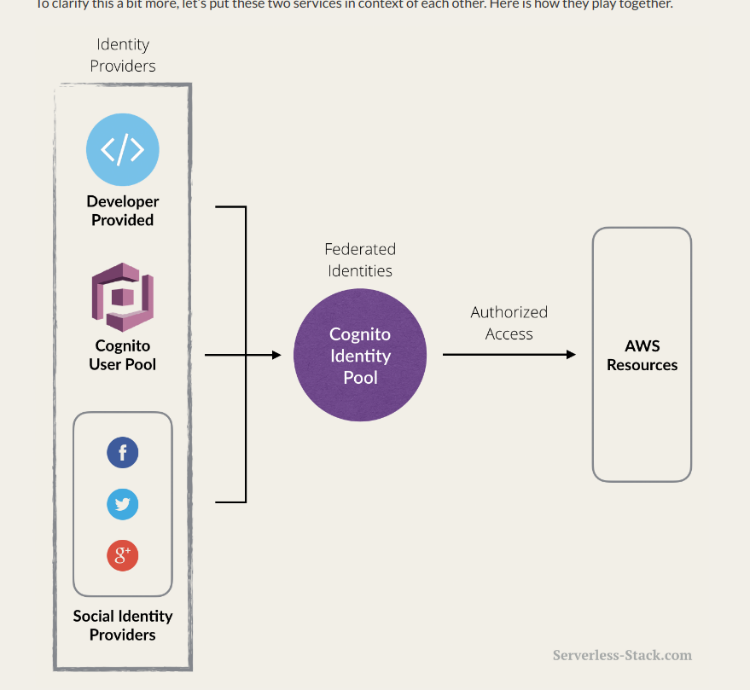
* Sign Up and Sign in directly, or through identity providers
* A directory for all your users, a UI for users to sign up and store profiles
* User tokens are returned to your app
* Token handling for your app is provided by SDK
* Allows manually decoding and verifying JWT tokens as well
* Successful auth generates **JWT**

Identity Pools (Federated Identities)

* **Roles** and permissions
* Control access to AWS resources to control access.
* Can give different **permissions** and even **temporary** AWS credentials
* Unique ID for users and authenticate them with web ID providers??

Cognito Sync

* Enables cross-device syncing of application-related user data
* Uses **SNS** to do this

****

# Trusted Advisor

Summary:

* Tool that offers a snapshot of your service and helps identify misconfigurations, suggestions for improvement, and underutilized resources

AWS Guard Duty  is a threat detection service

* Integrates with Cloudwatch Events

# AWS Systems Manager

·        Parameter Store

·        Provides configuration and password management including passwords, database strings, and license stores

·        Provided at no cost

·

·        Similar to Secrets Manager

·

# API Gateway

Summary

* Can import API defs using Swagger syntax
* You can use it for SOAP passthrough
* CORS enforced by the client

Logs

* Must grant API Gateway IAM permissions to read and write to CloudWatch

Auth

* Usage plans and API Keys -  allow you to distribute your API to customers. Ex. Taleo
* Lambda authorizer - Use lambda functions to control access. Allows auth strategies such as JWT and OAuth.
* Cognito User Pools
* IAM roles and policies
* CORS
* Client-Side SSL certs
* Resource Policies - Allow or deny access based on IP addr or VPC

Mapping Templates

* Defines a set of variables with models and mapping templates
* $context variable for lambda

Canary Deployment

* Able to release the new version of the API concurrently with the old version (in a preconfigured ratio) and rollover when you’re ready

Caching - This is a thing. Get requests, controlled by TTL, 0-3600. Client can invalidate using Header **Cache-Control: max-age=0**

Throttling

* Default is 10,000 requests per second, 5000 can process at once
* Max concurrent requests across all APIs on account

# Lambda

Summary:

* FAAS
* Languages: Node, Java, Go, C#, Python
* Layers allow you to put unchanging code (libraries) across functions
* Triggers: S3, Dynamo, Kinesis, SNS, SES, SQS, Cognito, Cloudformation, Cloudwatch Logs/Events, CodeCommit, Scheduled Events, Config, Alexa, Lex, API Gateway, IOT Button, CloudFront, Kinesis Data Firehouse
* Scales OUT not up

Aliases

* Allow you to have more than one version of your function that you can use. They each have unique ARNs
* Only difference between aliases and versions is that ALIASES ARE MUTABLE (they can change)

Security:

* If you need 1 MB file encrypted you must use **envelope encryption** and store as file within the code

Deployments are always blue/green

If a lambda is invoked asynchronously and retries twice before they fail and send to DLQ

Lambda@Edge is a feature of Amazon CloudFront that lets you run code closer to users of your application, which improves performance and reduces latency.

 you don't have to provision or manage infrastructure in multiple locations around the world. You pay only for the compute time you consume - there is no charge when your code is not running.

Limits:

* 1000 concurrent execs
* Function RAM 128-3008
* Timeout - 15 min
* Environment var - 4KB
* Deployment package size - **50MB** zipped**, 250** MB unzipped
* /tmp dir storage 512 MB

# SWF - Simple Workflow Service

You should use this if you require external signals to be a part of your workflow, ot you want to launch child processes that return a result to their parent. Otherwise use AWS STEP

# STEP

Summary - a way to visualize serverless applications. Reco over SWF for new stuff

* Uses CFN to provision resources
* Application & Integration
* Uses Amazon State Language (JSON)

# EC2 Elastic Compute Cloud

Pricing

* **On Demand** - Fixed rate, by the hour/sec with no commitment. When you want low cost and flexibility, best for short term and testing.
* **Reserved** - 1 to 3 year commitment extreme discount. Different types that give you discounts up to 75%
  + Scheduled - enable you to purchase capacity reservations that recur on a daily/weekly/monthly basis. For stuff that runs on a schedule
* **Spot** - server auto spins up and shuts off based on price, like stock market, applications that are only feasible at very low compute prices.  (bitcoin mining?)
* **Dedicated Hosts** - Physical EC2 server dedicated for your use. For regulatory and licensing that require dedicated server, stupid oracle.

Can run custom scripts via **USER DATA**

To retrieve instance metadata use<http://169.254.169.254>/latest/meta-data  This metadata includes public and private IP apparently

Need to know instance types for - Solution Architect Professional and SysOps Administrator Associate and Specialty Exams (FIGHT DR MC PX)

Security

* Encryption - Must create a snapshot of the EC2 volume. Create a copy of that volume (checking box to enable encryption). Create an AMI of the copied snapshot then  redeploy
* ec2config - For windows instances this service sets a random admin password and encrypts instance using the Ec2 public key
* Apps that run on Ec2 retrieve temp creds from the instance metadata

Elastic Beanstalk can be used to provision multiple identical environments

**Common exam Q:** If spot price goes over threshold amazon auto terminates and you don’t get charged for partial hour. But if you terminate it yourself you will be charged for the hour

Troubleshooting:

* Your ALB can’t connect to Ec2, it times out. The problem is your **security groups**. You need to add a security group rule that allows inbound traffic
* **Detailed** monitoring is available at **1 min intervals** for upcharge **(default mon is 5 min)**
* A security group acts as a virtual firewall for your instance to control inbound and outbound traffic.

# EBS - Elastic Block Store

Summary - Storage volumes that you attach to EC2 instances.

Types (SSD, Magnetic)

* General Purpose SSD (GP2) - General purpose, <10,000 IOPS this is best price for performance
* Provisioned IOPS SSD (O1) - Optimized for extreme performance, >10,000 IOPs

ex. Gigantic DB

* Throughput Optimized HDD (ST1) - used for big data, can’t be boot volume
* Cold HDD (SC1) - Lowest cost, used for data that isnt accessed often, file server, can’t be boot volume
* Magnetic (Standard) - Lowest cost that is bootable LEGACY, don’t really use this, maybe in dev

**EBS Volumes are AVAILABILITY ZONE locked!** One of the only things that are AZ specific

# ASG - Auto Scaling Groups

Summary

* Contains a collection of Amazon Ec2 instances that share similar characteristics
* If an instance is unhealthy it is terminated and replaced
* You can also autoscale other services using the Application Auto Scaling API
* Enabled by **CLOUDWATCH**

Application Auto-Scaling API

* ECS, EMR clusters, Appstream, DynamoDB Capacity, Aurora Replicas, SageMaker
* ASG vs EC2 Auto Scaling - Use EC2 auto scaling if you only want to autoscale EC2 instances.
* Predictive Scaling - Uses historic information about your usage to schedule future usage, uses ML

# ECR - Elastic Container Registry

Summary

* Docker container registry that allows you to store docker images
* ECR has no costs
* Automatically encrypts containers at rest and in transit over HTTPS
* Can control access to images using IAM

# ECS - Elastic Container Service

Summary

* Container management service that makes it easy to run, stop, and manage Docker containers on a cluster of Amazon EC2 instances.
* To prepare your application to run on Amazon ECS, you create a task definition

ECS\_ENABLE\_TASK\_IAM\_ROLE=true enables IAM role

**Task Definition**

* Specifies
  + Containers to use
  + Ports to use
  + Launch Types to use
  + Data Volumes to use
* Ex: task definition

{

   "family": "webserver",

   "containerDefinitions": [

       {

           "name": "web",

           "image": "nginx",

           "memory": "100",

           "cpu": "99"

       },

   ],

   "requiresCompatibilities": [

       "FARGATE"

   ],

   "networkMode": "awsvpc",

   "memory": "512",

   "cpu": "256",

}

* ECR - When using ECR image in definition use full registry/repository:tag

*aws\_account\_id*.dkr.ecr.*region*.amazonaws.com/*my-web-app*:*latest*

# Elastic Beanstalk

Summary

* Service for deploying and scaling web applications and services
* Supports Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, Nginx, Passenger, and IIS
* Makes it so developer don’t have to worry about underlying resources
* Compute Service

Deployments

* **All at once** - default, Deploy the new version to all instances at the same time. All instances are out of service for a short time (**Downtime**)
* **Rolling** - Elastic Beanstalk splits the environment into batches and updates them one batch at a time. Some instances will serve old version and some new until deployment is complete
* **Rolling deployment with an additional batch** - Will launch a new batch of instances prior to taking any instances out of service. Use if you want to maintain full capacity during deployments. After deployment additional batch is terminated
* **Immutable** - Launches full set of new instances running new version alongside old instances. If EC2 instances don’t pass health checks, they get terminated and originals stay. If you want to allow instances passing with lower status you can modify the **Healthy threshold**. Optionally you can override with **ignore health check** option
* Blue/Green - Use if you want to manually do immutable

.ebextensions/

* **.ebextensions/settings.config** - This file allows you to configure your environment
* Any resources created in ebextensions will be deleted if environment is terminated
* If you want them to last you must define them externally

Tips:

* Uses Cloud Formation to deploy applications.
* Worker environments can be used for task that take a long time to complete. you can offload those tasks to a dedicated worker environment.
* For RDS, deploy it by itself first, then connect to Beanstalk. Must create security group and connection string
* Can do Cron jobs Enable periodic tasks by including a YAML file in your source code that defines the schedule and path for each task.

Lifecycle Policy

* **A lifecycle policy tells Elastic Beanstalk to delete application versions that are old**, or to delete application versions when the total number of versions for an application exceeds a specified number.
* applies an application's lifecycle policy each time you create a new application version

# ELB - Elastic Load Balancer

Summary - Balances load across web servers.

Types:

* **Application** - Can see all the way up to Application Layer and cleverly route. Suited for HTTP/HTTPS/Websocket traffic. (Tesla Model S)
* **Network** - Most expensive, best performance. Suited for routing TCP traffic, can handle millions of request per second, also allows static IPS. (Tesla Roadster)
* **Classic** - Legacy, can load balance at application layer or network layer. However it’s application load balancing isn’t intelligent, uses X-Forwarding and sticky sessions

Developer Exam test mostly on Classic even though legacy. (Old Beater)

Troubleshooting

* **504 Gateway Timeout Erro**r - (Not the load balancer itself having an error) means you need to troubleshoot **webserver** or **db**.
* 4XX Error - This is a Client side error
* **Stickiness** will route a user to the first instance they went to. This prevents relogin

User -> Load Balancer -> EC2 Instance

On server if you need to know user’s public IPV4 address look in the X-Forwarded-For header.

To get certs can use AWS Cert Manager or generate your own and upload them to Cert Manager or IAM

# RDS - Relational Database Service

Summary: It is Online Transaction Processing **(OLTP**). I.e A relational database.

Types

* SQL Server
* Oracle (gross)
* MySQL Server
* PostgreSQL
* Aurora (Amazon’s SQL database technology)
* MariaDB

Back Ups

* Automated - can recover db any time in your retention period 1 - 35 (default 7) days. These backups are stored on S3, you automatically get a bucket = to your db size
* Snapshots - Done manually, stored even after RDS instance is deleted.
* Restored db will be new RDS instance with new DNS endpoint.

Encryption

* Encryption at rest is supported using AWS Key Management.
* Everything will be encrypted from snapshots to backups to db.
* Can’t encrypt existing db, **must take snapshot then encrypt that**. Then deploy a new RDS instance

Multi-AZ

* For Disaster Recovery ONLY
* Available for all SQL dbs in AWS (Built into Aurora by default)
* Any changes to db, its “backup” will also get these changes
* If for any reason main db goes down it auto fails over to backup. DNS addr updates automatically
* Minimal downtime, a minute or so
* Synchronous (supposedly not important for any exam but professional)

Read Replicas

* Meant for performance improvement, allows read only replicas to be created
* When db is written to, then it replicates across its replicas.
* Benefit is you can have some of your EC2 instances point to the replicas for just reading
* This is scaling out your database (Up to 5)
* You can get read replica of a read replica (mind-blown)
* Available for all RDS except SQL Server and Oracle
* Async (supposedly not important for any exam but professional)
* Each Read Replica has its own dns endpoint

You never deal with IPs with RDS, always DNS. This is important for disaster recovery.

Potential question: How to improve performance: Use ElastiCache.

Potential question: Troubleshooting - You have an ec2 instance and you have an instance of RDS. They are in two different security groups by default. So the ec2 instance won’t be able to talk to RDS. You must go to the RDS security group and allow inbound connection on 3306 from your webserver security group.

Potential question: When to use Read Replica vs Multi A-Z

# DynamoDB

Summary:

* Serverless
* Stored on SSD
* Conditions on IAM Policy to restrict data that isn’t your own using dynamodb:LeadingKeys

LSI (Local Secondary Indexes)

* Has to be created as you’re creating table
* Has same primary key but can have different sort key

GSI (Global Secondary Indexes)

* Use these when you want to query by something that isn’t the partition key
* Keys in a GSI don’t need to be unique
* The tradeoff to projecting every attribute into an index is the cost to store and write
* If you need to access most of the non-key attributes often project the entire base table, but storage cost will double

Queries:

* Default is eventually consistent but can be strongly consistent
* ProjectionExpression to refine results

Scan:

* Hits whole table
* Can set **smaller page size**
* Use **parallel scans**

Read/Write Capacity Modes:

* On Demand - autoscaling
  + Use if workloads are unknown, unpredictable traffic, want to pay as you go
* Provisioned (free-tier)

Calculating Throughput

* Read (4 KB)
  + One Read Request Unit = 1 4KB strongly consistent read or 2 4KB eventually consistent reads
  + Transactional read requests require 2 Read Request Units
* Write (1 KB)
  + One Write Request Unit = 1 1 KB write
  + Transactional write requests require 2 Write Request Units

Limits:

* 256 tables per region

TTL:

* Can set data to expire
* Expressed in epoch time
* Will be deleted within 48 hours of being marked for deletion

Transactions:

* transactions simplify the developer experience of making coordinated, all-or-nothing changes to multiple items both within and across tables. Transactions provide atomicity, consistency, isolation, and durability (ACID) in DynamoDB, helping you to maintain data correctness in your applications.
* ACID transactions, all or nothing
* Finances

Tips

* **DAX -** DynamoDB Accelerator (caching service): delivers fast response times for accessing eventually consistent data. Reduces response time from milliseconds to microseconds. Useful for read heavy or bursty workloads. Supports encryption. Don’t use if app requires strongly consistent reads, write intensive, or are caching a different way. (Write through)
* **Atomic Counter** - Allows all write requests to be applied in order to increment/decrement value
* Hot partitions are a thing
* By default writes are **unconditional**, but you can have **conditional** writes, which can be idempotent
* Streams - Logs are stored for 24 hours, tune-ordered seq of modifications
* ProvisionedThroughputExceedException - Use exponential backoff, reduce request frequency. Used in every **AWS SDK**

# Elasticache

Summary - Web service that allows an **in memory cache** in the cloud. Used for speeding up applications. A way of caching frequent queries from database. Improve latency and throughput for read heavy applications.

Types:

* **Memcached** - If no redundancy is necessary use Memcached. Pure caching, no persistence.
  + Simpler
  + Caching is primary goal
  + Scale out (horizontal)
  + Not Multi-AZ
* **Redis** - open source in memory key value store.
  + Use this if you want Multi-AZ redundancy.
  + Managed as a relational db, is stateful
  + Advanced datatypes (lists, hashes, sets)
  + Sorting and ranking datasets (leaderboards)
  + Persistence
  + Across multiple AWS availability zones

Strategies

* Lazy loading
  + Loads the data in cache when necessary only requested data is cached.
  + Doesn’t auto update if database changes, so stale data can happen.
  + Use TTL to make data less stale
* Write Through
  + cache whenever data is written to the database.
  + Adds/Updates cache every time data is put in DB
  + Data is never stale, but must write every time
  + Use if users care more about read speed than write speed
  + Wasteful if most data isn’t used

Curveball, **Redshift** could be the answer to free up a strained database IF the reason for the db stress is because you keep running OLAP (analytics) transactions. Data warehousing.

# SQS - Simple Queueing Service

Summary:

* Message oriented queueing service that allows decoupling of microservices, distributed systems and serverless apps
* NO ORDER - SQS messages can be delivered multiple times in any order
* Design - you can have 2 priority queues for priority-based message one for higher and other for lower priority
* EC2 instances always poll for messages from the queue (**pull** from the queue and not push)
* Long polling can reduce costs, it waits for a message to come in before returning. **Maximum long polling timeout 20 seconds**. This reduces # of empty ReceiveMessageResponse(s) returned
* FIFO queues end in .fifo - offer exactly once processing, support multiple producers but not multiple consumers

Limits

* **NO LIMIT** for number of messages in a SQS queue
* MAX of **120,000** in-flight messages (FIFO queue 20,000)
* Message can contain **1-256KB of tex**t, billed at 64KB chunks (For messages bigger than this use SQS Extended Client)
* Single consumers can have **1 to 10** messages unto maximum of 256KB payload
* Messages in the Queue can be retained from **1 min** up to **14 days**
* 3,000 messages per second with batching
* Messages can have any format

Billing:

* First 1 million request ares free, then $0.50 PER EVERY MILLION REQUESTS

DLQ - Dead Letter Queues:

* Other queues can put messages here when they aren’t processed successfully
* Main purpose is handling failures

VisibilityTimeout

* It is a period of time where it prevents other consumers for processing the message again
* Visibility timeout always start from when the application instance polled the message.
* Visibility timeout expires that means there is a failure somewhere since that message was polled but not processed and hence not deleted so other some other process will poll the message again and visibility timeout starts again.
* Visibility timeout by **default is 30 Seconds** min is **0 seconds**, up to **12 hour** maximum (ChangeMessageVisibility) / maximum visibility
* ChangeMessageVisibility -  Changes the timeout of a message in a queue to a new value.

**Key Terms**

* **Message deduplication ID:** If a message with a particular message deduplication ID is sent successfully, any messages sent with the same message deduplication ID are accepted successfully but aren't delivered during the 5-minute deduplication interval.
* **Message Group ID:** Messages that belong to the same message group are always processed one by one, in a strict order relative to the message group (however, messages that belong to different message groups might be processed out of order).
* **Receive request attempt ID:** The token used for deduplication of ReceiveMessage calls.
* **Sequence Number:** The large, non-consecutive number that Amazon SQS assigns to each message.

# SNS - Simple Notification Service

Summary:

* Pub/Sub messaging for microservices, distributed systems and serverless apps
* SNS can notify to Email, Text / SMS, SQS or any HTTP end point (and lambda functions).
* Lower cost that SES

Tips:

* Using AWS PrivateLink you can publish messages without traversing the internet
* CloudWatch or Autoscaling triggers SNS
* protocols: HTTP, HTTPS, EMAIL, EMAIL-JSON, SQS or Application - messages can be customized for each protocol
* SNS messages are stored redundantly to multiple AZs
* SNS Dataformat - JSON (Subject, Message, TopicArn, MessageId, unsubscribeURL etc..)

Cost:

* $0.50 per 1 million SNS request
* Different price for different recipient types
* to HTTP: $0.06 / 100,000 notifications deliveries
* to EMAIL: $2 / 100,000 notifications deliveries
* to SMS: $0.75 / 100 notifications deliveries

Fanout v Filtering

* Fanout - Occurs when a message is sent to a topic and then replicated and pushed to multiple endpoints
* Filtering - Subscribers can filter out messages they don’t want/need

# SES - Simple Email Service

Summary:

* Email sending service
* Max email size: 10 MB
* Max Recipients: 50
* Encrypted using KMS
* More expensive than SNS, but doesn’t require user opt in to send email

# EMR - Elastic Map Reduce

Summary:

* A container service that allows you to run and scale Spark, **Hadoop**, HBase, Presto, Hive and other **BIG DATA f**rameworks
* An EMR cluster is all the steps needed. They all start with ‘j-’
* A Cluster step is a unit of processing written in Java, Ruby, Perl, Python PHP, R, C++, and example step is count frequency of words in a document
* Input data in S3
* Uses Ec2 Instance(s) for processing

Cluster States:

* Starting - Cluster provisions, starts, Ec2 configured
* Bootstrapping -
* Running - One of the steps  is running
* Waiting - On, but no steps are running
* Terminating
* Terminated
* Terminated with errors

# CloudFormation

Summary:

* Templates that allow you to provision cloud resources. collection of sample templates will help you get started with AWS CloudFormation and quickly build your own templates
* Can save as .json, .yml, .template, or .txt
* To makes changes to a running stack use a change set (allows to see what your changes will do)
* Stored in S3

Intrinsic Functions:

* Function that allows you to assign values to properties not available til runtime.:
* Can use in resource properties, outputs, metadata attributes, update policy attributes and to conditionally render stack resources

Sections:

* Metadata - during stack update you can’t directly edit this
* Parameters - values to pass at runtime. Allows user to specify arguments
* Mappings - Key,value lookup table. Can use Fn::FindInMap intrinsic function
* Conditions
  + Can’t be applied to Parameters
* Transform - Where SAM templates go and other transforms/macros
* Resources - Only required section, defines your stack resources and their properties
* Outputs - Values that are returned when you view your stacks properties.
  + **Names must be unique within a region, can only ref stacks in your region**
  + "Outputs" : {  
       "*Logical ID*" : {  
         "Description" : "*Information about the value*",  
         "Value" : "*Value to return*",  
         "Export" : {  
           "Name" : "*Value to export*"  
         }  
       }  
     }

Since Cloudwatch templates can reference other stacks, always delete them in reverse order that they are referenced. Nested stacks created in resources section

Commands

* Cloudformation package
* Cloudformation deploy

Example Intrinsic functions

* **Fn::Base64** - a group of binary-to-text encoding schemes that represent binary data in an ASCII string format by translating it into a radix-64 representation.
* **Fn::Cidr** - returns an array of CIDR address blocks,

{ "Fn::Cidr" : [*ipBlock*, *count*, *cidrBits*]}

* Conditions - allows conditonal render of resources
  + Fn:And, Fm:Equals, Fn::If, Fn::Not, Fn::Or
* **Fn:GetAtt** - Gets value of an attribute from a resource in the template. Example use case would be to assign an S3 bucket name the value of the name of something else

{ "Fn::GetAtt" : [ "*logicalNameOfResource*", "*attributeName*" ] }

Shorthand -> !GetAtt *logicalNameOfResource*.*attributeName*

* **Fn:GetAZ** - gets availability zones for a region

{ "Fn::GetAZs" : "region" }

* **Fn::ImportValue** - **Returns value of resource exported from another stack**. **Typically used to reference resources across stacks**

{ "Fn::ImportValue" **:** sharedValueToImport }

* **Fn::FindInMap** - Returns value corresponding to keys in a two-level map declared in the Mappings section

{ "Fn::FindInMap" : [ "MapName", "TopLevelKey", "SecondLevelKey"] }

* **Fn::Join** - literally python join

{ "Fn::Join" : **[** "delimiter", [ comma-delimited list of values ] ] }

* **Fn::Select** - Selects object from list by index.

{ "Fn::Select" : [ index, listOfObjects ] }

* **Fn::Split** - Literally python split

{ "Fn::Split" : [ "delimiter", "source string" ] }

* **Fn::Sub** - substitutes variables in a string with values, 2 or 3 params

{ "Fn::Sub": [ "www.${Domain}", { "Domain": {"Ref" : "RootDomainName" }}]}

* **Fn:Transform** - Specifies macros for processing, SAM is one of these

{ "Fn::Transform" : { "Name" : macro name, "Parameters" : {key : value,... }}}

* Ref - Gets value of specified resource. Ex get Ec2 ID

"InstanceId" : { **"Ref" : "MyEC2Instance"** }

# SAM - Serverless Application Model

Summary:

* AWS::Serverless Transform - a macro hosted by AWS CloudFormation
* Takes a SAM template and converts it to a compliant cfn template
* Requires no special permissions

Globals - Lets you defined properties that you can use in all Serverless Functions and APIs. **AWS::Serverless::Function** and **AWS::Serverless::Api**

Resources:

* AWS::Serverless::Function - Creates lambda, IAM execution role, and event source mappings (Triggers), Also creates an API but not as extensible as using AWS::Serverless::Api
* AWS::Serverless::Api - Creates collection of API Gateway resources/methods
* AWS::Serverless::Application - Embeds a serverless app from either AWS serverless app repo (the AWS provided projects) or an S3 bucket
* AWS::Serverless::SimpleTable - Creates a DynamoDB table with one attribute (primary key)
* AWS::Serverless::LayerVersion - The new lambda layers stuff

CLI Commands

* Sam package
* Sam deploy

Ex: template.yaml

<https://www.baeldung.com/aws-serverless>

# CloudWatch

Summary

* **Monitoring** and management service built for developers, sys-ops, site reliability engineers, and IT managers.
* Monitoring service for AWS cloud resources and applications you run on AWS
* EC2, DynamoDB, RDS and any log files you generate
* Main focus in monitoring/automation/debugging
* Management Service

Cloudwatch Logs

* Log Expiration policies - **Default** is set to **never expire**, but can set retention between **20 years** and **one day**
* Logs Agent - Supports IAM policy, is how you monitor logs on EC2

**Cloudwatch Events**

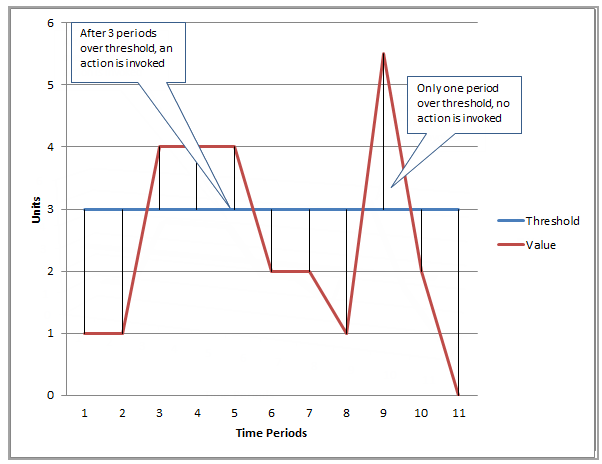
* **Describes changes in AWS Resources**
* Can set up rules to configure notifications, i.e. **email on CodeCommit**
* Difference between this and Cloudtrail is - this is near real-time and you can set rules that perform automated actions. CloudTrail just delivers logs of API calls

CloudWatch Metrics

* Standard Resolution Metrics - Default: CPU, Network, Disk, Status Check
  + **RAM Utilization** is a custom metric
* High -Resolution Metrics (cost more):
  + Stores it with **resolution** of **one second**
  + Can retrieve/read metric with periods of 1, 5, 10, 30, 60+
* Custom Metric - minimum granularity can have is **1 minute,** minimum **resolution is 1 second**
* Get metric data using GetMetricStatistics API
* Dimensions - In custom metrics you can classify what it is using dimensions {key, value}
* Can’t delete metrics

Cloudwatch Alarms

* Standard Resolution Metrics - 60+
* High Resolution Metrics - **10** or **30** seconds
* Can take actions such as sending email, terminating EC2 instance, or ASG policy
* Pick Metric -> Pick Evaluation period (5 min - 60 min) -> Pick value to measure (avg/max) ->Set a threshold



# CloudTrail

Summary

* Monitor your AWS deployments in the cloud by getting a history of AWS **API** calls for your account, including API calls made via the AWS Management **Console**, the AWS **SDKs**,**CLI**, and higher-level **AWS services**
* Identify Users, Accounts, IPs, Timestamps
* Define S3 bucket for storage
* Can define workflows that execute when security events are detected
* Troubleshoot security and operational issues
* **Auditing** changes to services
* Enables governance, compliance
* Monitor **Authorization** attempts

Can view up to past 90 days just by turning it on, by default logs are stored indefinitely.

By default encrypted with S3 SSE, can also enable MFA for deletion

No cost for management events, but data events do cost. Data events are operations performed on the resource itself

# X-Ray

Summary:

* Helps developers debug and analyze distributed applications (like serverless)
* Lambda X-Ray active tracing must be enabled to trace lambda functions
* Integrates with: ELB, Lambda, API GW, EC2, EBS

X-RAY SDK provides

* Interceptors to add to your code to trace incoming HTTP requests
* Client handlers to instrument AWS SDK clients that your app uses to call other AWS services
* An HTTP client to instrument calls to other internal and external HTTP web services

Sampling - An algorithm applied that makes it so that you get a comprehensive sample without needing to log everyone. Default is first one every second and 5% of each one after.

X-Ray Daemon

* Instead of sending trace data directly to X-Ray, the SDK sends JSON segment documents to a daemon process listening for UDP traffic. The [**X-Ray daemon**](https://docs.aws.amazon.com/xray/latest/devguide/xray-daemon.html) buffers segments in a queue and uploads them to X-Ray in batches. The daemon is available for Linux, Windows, and macOS, and is included on AWS Elastic Beanstalk and AWS Lambda platforms.
* The AWS X-Ray daemon is a software application that listens for traffic on UDP port 2000, gathers raw segment data, and relays it to the AWS X-Ray API. The daemon works in conjunction with the AWS X-Ray SDKs and must be running so that data sent by the SDKs can reach the X-Ray service.

Annotations && Metadata

* Annotations: key-value pairs with string, number, or Boolean values. Annotations are indexed for use with [filter expressions](https://docs.aws.amazon.com/xray/latest/devguide/xray-console-filters.html). **Use annotations to record data that you want to use to group traces in the console, or when calling the**[**GetTraceSummaries**](https://docs.aws.amazon.com/xray/latest/api/API_GetTraceSummaries.html)**API.**
* Metadata: key-value pairs that can have values of any type, including objects and lists, but are not indexed for use with filter expressions. **Use metadata to record additional data that you want stored in the trace but don't need to use with search.**

# CodeCommit

Summary:

* Source Control Service, can host private git repos
* Data is encrypted in transit and at rest
* Can configure notifications via Cloudwatch Event Rules

# CodeBuild

Summary

* Input artifacts - source code input files, also called **build artificats**
* **Input and output files in s3 bucket**
* To watch build use CloudWatch Logs
* To Encrypt specify a KMS key to use
* Can add your own settings in console at buildtime even if you can’t access the buildspec file
* Serverless
* Use S3 caching to cache dependencies

Build\_spec.yml

* Collection of build commands and related settings
* Usually buildspec.yml is stored in root level of project but when storing in s3 it must be in root level of zip
* Ex.

version: 0.2

phases:

 install:

   commands:

     - echo Nothing to do in the install phase...

 pre\_build:

   commands:

     - echo Nothing to do in the pre\_build phase...

 build:

   commands:

     - echo Build started on `date`

     - mvn install

 post\_build:

   commands:

     - echo Build completed on `date`

artifacts:

 files:

   - target/messageUtil-1.0.jar

IAM roles needed to create CodeBuil:

* AWSCodeBuildAdminAccess
* AmazonS3ReadOnlyAccess
* IAMFullAccess

Pricing

# CodeDeploy

Summary:

* A deployment service that automates deployments to EC2 instances, on prem instances and serverless functions
* Can deploy to servers, serverless, and container apps
* Can only deploy **CHANGES**, not applications. Use EBS for deploying apps
* Also does not give application monitoring, use EBS for that
* Application Revisions are stored in S3 Buckets
* Use deployment groups to deploy fractions at a time

Deployment types:

* **Blue/Green (Old, New)**
  + EC2 - Original environment is replaced by new environment
  + Lambda - Traffic is shifted from old lambdas to new ones
  + ECS - Traffic is shifted from one container to the other
  + Rollback is easy with this type
* **In-Place (Also known as rolling)**
  + Applications are stopped and upgraded with new code and then started
  + Must re-deploy

Rollback

* Rolls back by re-deploying previously deployed versions
* Starts with FAILED instances
* Automatic - Can configure automatic rollback on failure or a monitoring threshold is met
* Manual - Done by creating a deployment with a previously deployed application revision

**AppSpec Files** - Application Specification

* Used to manage each deployment as a series of lifecycle event hooks.
* Must have space after key in file
* ECS - Includes name of ECS service, container name, and ports to use for traffic. Also validation tests
* Lambda - Lambda function version to deploy, function to be used as validation tests
* EC2 - On EC2 instances the Appsec file is always YAML formatted

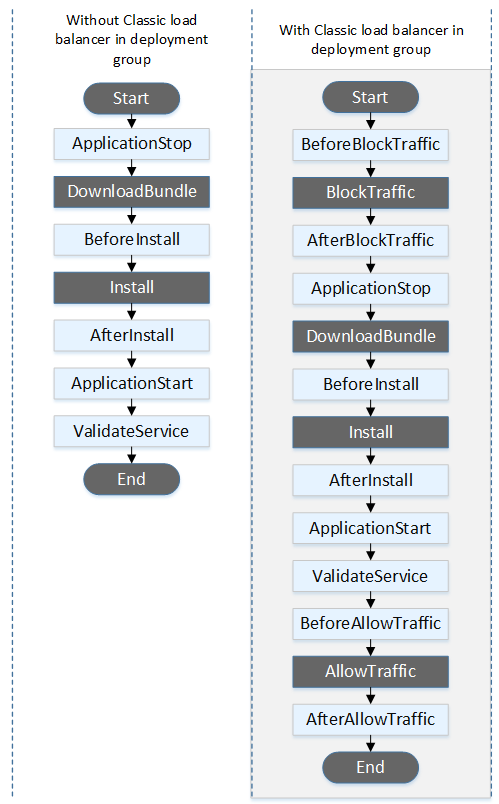
For Below hooks

* All the **\*traffic** events involve before/after it is registered with the load balancer
* **ValidateService** – This is the last deployment lifecycle event for EC2. It is used to verify the deployment was completed successfully.



Hooks for deployment

* ECS: BeforeInstall -> AfterInstall -> AfterAllowTestTraffic -> BeforeAllowTraffic -> AfterAllowTraffic (The app is deployed at this step)
* Lambda: BeforeAllowTraffic -> AfterAllowTraffic
* EC2:



# Code Pipeline

Summary

* Continuous delivery service that helps automate pipelines for applications
* A serverless managed service
* Automates build, test, and deploy phases on every code change
* Integrates with CodeCommit, CodeBuild, CodeDeploy, S3, Elastic Beanstalk, CloudFormation, OpsWorks, ECS, Lambda

Optimization

* Can parallelize steps that can run simultaneously to speed up.
* Can bundle dependencies in the source code at last step of CodeBuild to speed up Beanstalk deployments

Revision - Change made to the source location defined for pipeline. Includes source code, build output, configuration, or data.A pipeline can have multiple at the same time

Stage - group of one or more actions

Action - task performed on a revision

Limits

* 300 max number of pipelines per region per account
* Stages: min 2, max 10
* Actions: min 1, max 50

# AWS VPC

# WAF - Web Application Firewall

Summary:

* Protection for common web attacks
* Configure Web ACLs (Access Control Lists)

AWS Shield gives DDOS protection

# CloudFront

a content delivery network (CDN) service that delivers static and dynamic web content, video streams, and APIs around the world, securely and at scale. By design, delivering data out of CloudFront can be more cost-effective than delivering it from S3 directly to your users.

Summary

* Speeds up distribution of your static and dynamic web content
* Data, Videos, Applications, and APIs
* Objects are cached with a TTL

Protocols supported: HTTP, HTTPS, RTMP (Real-Time Messaging Protocol)

RTMP is used for video streaming via CloudFront

Edge location is a thing, not the same as AZ, where your content is cached

Can use signed URL/Cookie to restrict it

# Route 53

Summary

* DNS Web Service
* Functions: Domain registration, **DNS routing**, and health checking
* Replicates some ELB functionality

Domain Registration

* You need to provide names and contact information for the domain
* Registrar is Amazon Registrar or Gandi

DNS Routing

* Routing Policies:
  + Simple - For a single resource that performs a function for your domain
  + Failover - For **active-passive** failover
  + Geolocation - To route traffic based on **location of your user**
  + Geoproximity - Route traffic based on **location of your resources**, and shift resources from one location to another optionally
  + Latency - When you have resources in multiple regions and want to route traffic to region with best latency
  + Multivalue answer - Makes route 53 respond to queries with up to 8 healthy records…. Selected at random
  + Weighted routing policy - Route traffic to multiple resources in proportions that you specify

Other Info

Data in a region is not replicated outside of that region, we must do that ourselves.

# Kinesis

Summary

* Collect, process, and analyze video and data streams in real time
* Batch records to save cost increase throughput
* Retention period for data record is 24 hours after creation up to 7 days
* Uses AWS KMS for encryption

Terminology:

* Shard - Sequence of data streams (1,000 MB/s)
* Kinesis Data Stream - A set of shards, each shard has a sequence of data records
* Data record - Seq #, partition key, data blob

Kinesis Streams

* Accepts data from data producers stores in shards
* Then data consumers (EC2) do some processing on the data
* Then send data to be stored somewhere

Kinesis Firehouse

* Same as Streams except there’s no shards, streams, or consumers. This is all automated. Analyzed using Lambda
* No retention, in and out to S3.

Kinesis Analytics

* Can run SQL queries off data from Firehouse or Streams

Increase number of shards and allow partition key to take many attributes **in order to scale**

# AWS CLI

**--dry-run** allows you to test the output of a command (Check if you have permissions) without actually running it

Aws iam get-user check your current creds

STS decoded-authorization-message is a thing

Paginate = number the pages of

**Three options to control the number of items included in the output when the AWS CLI calls a service’s API to populate the list.**

* --page-size default 1000 and retrieves all available items. specify that the AWS CLI requests a smaller number of items from each call to the AWS service.
* --max-items: handles pagination with the service as described previously but prints out only the number of items at a time that you specify.
* --starting-token: If the number of items output (--max-items) is fewer than the total number of items returned by the underlying API calls, the output includes a NextToken that you can pass to a subsequent command to retrieve the next set of items.
* --no-paginate: To disable pagination and return only the first page of results, use the --no-paginate

DynamoDB Scan commands

# CodeStar

Summary: enables you to quickly develop, build, and deploy applications on AWS.

* provides a unified user interface, enabling you to easily manage your software development activities in one place.
* you can set up your entire continuous delivery toolchain in minutes, allowing you to start releasing code faster.