**AWS Global Infrastructure \***

* Region – Geographical location
* Availability Zone – Data center, 2 or more ny region
* Edge Location – Cashing, Cloud Front, CDN- Content Delivery Network

**Compute \***

* **EC2** – vm
* EC2 container service
* **Elastic BeanStalk**
* **Lamda**
* Lightsail – Virtual private service, fixed IP address
* Batch – not important, Batch computing in cloud

**Storage \***

* S3 – simple storage service, object based stored, bucket
* EFS – Elastic File System - Network based storage
* Glacier – Data Archival
* Snowball – bring large data into datacenter
* Storage Gateway – Virtual appliance, replicates date to S3

**Database \***

* RDS – Relational Db service, mysql, postgress, oracle
* DynamoDB – Non relational
* Elasticache - caching data
* Red Shift – Data Warehousing

**Migration**

* AWS Migration Hub – track the migration
* Application Discovery service - discover the dependencies. Eg: sharepoint or dependency with DB
* Database migration service
* Server migration service
* Snowball

**Networking & Content Delivery\***

* **VPC** – virtual privacy cloud. Configure your own private cloud
* **CloudFront** – CDN, stores your image/video closer to your AZ, edgelocation
* **Route53** – DNS service
* **API Gateway –**
* **Direct Connect** – direct line from hdq to amazon cloud

**Developer Tools**

* Not important
* CodeStar – colabrating code
* CodeCommit – store code. Version controller
* Code Build
* Code Deploy
* CodePipeline
* X-Ray - debug
* Cloud9 – IDE, browser

**Management Tools\***

* **Cloud watch –**
* **Cloudformation** –
* **CloudTrail** – log the changes
* **Config** – monitor the configuration, visual
* OpsWorks
* Service Catalog
* Systems Manager – patch all the EC2
* **Trusted Advisor** – advice security, risks, capacity
* Managed services

**Media Sevices**

* Elastic Transcoder – resize video to android /ios

**Machine Learning**

* Not important
* Sagemaker –
* Comprehend
* Deeplens – physical hardware – camara
* Lex
* Machine learning
* **Polly** – text to voice
* Rekognition – upload image / video, will tell you what it is
* Amazon translate – translator
* Amazon Transcribe – speech recognition – speech to text

**Analytics \***

* Athena
* **EMR** – big data solution. To process large data
* Cloudsearch
* ElasticSearchService
* **Kinesis** – bigdata, ingesting large amount of data in cloud
* Kinesis Video Streams
* Quicksight – BI tool
* **Data Pipleline** – moving the data bw to amazon services
* Glue – ETL

**Security & Identity & Compliance \***

* **IAM** – Identify Access Management
* **Cognito –** Auth service, gives temp access to AWS
* GuardDuty – malicious activity
* **Inspector** – governance tool, check the vulnerability
* **Macie –** scans S3 for PII data
* **Certificate Manager –** manage SSL
* **CloudHSM** – Hardware security Module - stores encryption keys
* **Directory Service** – integrating Active directory
* **WAF** – Web Application Firewall – prevent XSS, Sql injection
* **Shield** – prevent Ddos attacks
* Artifact – compliance report, PCI report

**Mobile Service**

* Mobile Hub – management console for mobile apps,
* PinPoint – push notifications
* AWS Appsync – automatically updates the data (offline).
* Device Farm
* Mobile Analytics
* Not important

**AR / VR**

* Sumerian – code name

**Application Integration\***

* Step functions
* Amazon MQ
* **SNS** – Notification service
* **SQS -**  Decoupling
* **SWF** – Simple Workflow Service

**Customer Engagement**

* Amazon Connect
* SES - Simple Email Service

**Business Productivity**

* Alexa for Business
* Chime – Video conference like xoom
* Work Docs
* WorkMail – like office 365

**Desktop and App Streaming**

* Workspaces
* AppStream 2.0 like citrix

**IOT**

* IOT device management
* Amazon FreeRtos – OS for microcontroller
* GreenGrass

**Game Development**

* GameLift

**IAM**

* Identity Access Management
* Its Global
* Allows to manage users and their level of access to AWS console
* Users
* Groups - collection of users under one set of permissions
* Role
* Policy – document defines one or more permissions
  + Managed policies
    - Created and administrated by AWS
    - Can be attached to multiple users, roles, groups with aws or different aws accounts
    - You cannot change the permissions
    - Recommended
  + Customer Managed policies
    - Policy that you create
    - Can be attached to multiple users, roles, groups, but only with in your own aws account
  + Inline policies
    - Policy embedded with user, role or group
    - When you delete, user, role or group, inline policy also will be deleted
* Users – Programmatic, console
* Programmatic - Access Key Id & Secret Access Key
* Console – User & Password
* For region specific settings, add conditions as part of policy

**STS – Security Token Service**

* Grants users limited & temp access to AWS resources
* Federation – SAML, AD, Single sign on
* Federation Mobile – FB, amazon, google or openId to login
* Cross account access – one aws account to access another resource
* Federation – Combining list of users from one domain with list of users in another domain. Like from IAM to AD or FB
* Identity Broker – service that allows you take an identity from point A and join (federate) it with point B
* Identity Store – AD, FB, Google
* Identity – user of service like FB
* Identity Broker always authenticate with federation (AD, LDAP, FB, Google) first then STS

**Active Directory Federation**

* AssumeRoleWithSAML API
* Always authenticated with AD first then AWS
* SAML – Security Assertion Markup language

**Web Identity Federation**

* Authenticate using Google, FB, LinkedIn
* AssumeRoleWithWebIdentity API
* ARN – Amazon Resource Name

**Cognito**

* Web identity federation service
* User first authenticated with web identity provider (FB). Received auth token, exchanged for temp. aws credentials allowing them to assume an IAM role
* Provide signup / signin for your apps
* Access for guest users
* Acts identity broker between your app and web ID providers (FB or Google)
* Recommended for mobile apps runs on aws
* Uses Push synchronization – to push updates and synchronize user data across multiple devices
* SNS is used to silent push notification to all the devices associated with the given user identity
* User pools - A user pool is a user directory in Amazon Cognito. With a user pool, your users can sign in to your web or mobile app through Amazon Cognito. Your users can also \*sign in through social identity providers\* like Facebook or Amazon, and through SAML identity providers.
* Identity pools - Identity pools provide AWS credentials to grant your users access to other AWS services. To enable users in your user pool to access AWS resources, you can configure an identity pool to exchange user pool tokens for AWS credentials

**EC2**

* Elastic Compute Cloud
* Allowing only to pay capacity you used
* **OnDemand** – Pay a fixed rate by hour or second, low cost, flexible, short time, unpredictable workload, first time, no upfront payment, no long-term commitment
* Linux by second, windows are by hour
* **Reserved** – Reserve 1 year to 3 year terms, steady state, predictable usage, and reserved capacity, standard RI, convertible RI, and Scheduled RI. (RI – Reserved Instances)
* **Spot** – bid price, flexible start and end times
  + If you terminate the instance, you have pay for the partial hour
  + If amazon terminate the instance (bid range), you don’t have to pay
* **Dedicated** – Physical, use existing software licenses.
* Public IP or Elastic IP address
* Evaluation logic – default deny
* For penetration test – request to amazon and get the approval
* Test allowed (after approval) only for EC2 and RDS
* Vulnerability and penetration test to Other resources are prohibited
* If the instance is terminated, you can find the reasons under ‘State transition reason’ label
* By default all accounts are limited to 5 elastic ip addresses per region

**Instance Families**

* D2 – Density – File servers / Dataware / Hadoop
* R4 – RAM – Memory optimized Apps/DBs
* M5 – main choice General – app servers
* C5 – CPU – CPU intensive Apps / DNs
* G3 – Graphics – Video Encoding, 3D
* I3 – IOPS, high speed storage – NoSQL, DB, DataWare
* F1 – FPGA – Field Programmable Gate Array – Hardware Acceleration
* **T2 – cheap general purpose – web servers, small DB’s**
* P3 – Graphics (pics) – Machine Learning, Bit Coin
* X1 – Extreme Memory – SAP HANA, Apache spark
* H1 – High disk throughput - HDFS
* DIRTMCGFPX – dirty Melbourne ground (old)
* **DR Mc GIFT PX** - 2017 acronym
* **FIGHT DR Mc PX**
* F - FPGA
* I – IOPS
* G – Graphics
* H – High Disk Throughput
* T – Cheap general purpose (Think T2 micro)
* D – Density
* R – RAM
* M – Main choice for general purpose
* C – Compute
* P – Graphics (think pics)
* X – Extreme memory

**EBS**

* Elastic Block Store - allows to create to storage volumes and attach with EC2 instance
* Virtual disk
* Think of disk in the cloud attached to EC2
* Automatically replicated in AZ
* IOPS
* **General Purpose SSD - GP2 –** upto 10000 IOPS
* **Provisional IOPS SSD (IO1)** – more than 10000 IOPS – designed for I/O intensive applications, large relational or nosql DB, Can provision upto 20000 IOPS per volumn
* IOPS – input output operation per second
* **Throughput Optimized HDD (ST1) –** magnetic not SSD
  + Big Data
  + Frequently accessed workloads
  + Data warehouse
  + Cant boot volume
* **Cold HDD (SC1)**
  + Lowest cost
  + Less Frequently accessed workloads
  + File server
  + Cant boot volume
* **Magnetic (standard)** 
  + Bootable
  + HDD
  + Cheap
  + Data accessed Infrequently
* Cannot mount 1 EBS volume to multiple EC2 instances – instead us EFS
* Can transfer reserved instance from one AZ to another

**Instance Store**

* Called as Ephemeral storage
* Volume cannot be stopped. If the underlying host fails you will lose the data. But EBS backed instance can be stopped. You will not lose the data if the instance stopped.
* We can reboot both. Will not lose the data
* By default root volume will be deleted on termination. But EBS we can have option

**EC2 Lab**

* **AMI** – Amazon Machine Instance (when you create AMI, registerImage is the final process)
* AMI can be shared to other AWS account
* AMI can be changed to public
* If you make AMI, it will not be immediately available across all regions
* AMI’s can be only be shared within region. For other regions do copy
* Instance type – t2 micro
* Default VPC
* One subnet always will be in one AZ. Same subnet range will not be shared across AZ.
* Termination protection – by default its Off
* Monitoring – cloud watch
* Tenancy – dedicated
* Advanced Details – user input, Boot instructions (install PHP SDK, apache)
* Storage – Root, Boot volume
* By default Root volume can’t be encrypted. Either bit locker, or encrypt when creating AMI or API.
* By Default Root EBS volume will be deleted on termination. Option can be changed
* Tags
* Security Group, SSH, SSL, HTTP.
* Source – Custom, Anywhere, My IP. Anywhere means open to world
* Key Pair – Pem file
* Same private key can be used for multiple EC2
* Status check –System Status check, Instance Status check
* Monitoring, Cloud watch – basic monitoring, CPU. Disk,Nw
* Standard monitoring – 5 mins
* Private key should be protected from read/write from other users.
* Unprotected private key file – do chmod 400
* Unix
  + Change the permission chmod 400 \*.pem
  + ssh ec2-user@<ip> -i <pem>
  + sudo yum update -y -> to update security packages, -y to force update
  + yum install httpd –y –> install apache
  + cd /var/www/html
  + nano – text editor
  + nano index.html
  + Add something
  + Ctrl X to exit and save
  + service httpd start
  + chkconfig httpd on - starts automatically with boot
* Windows
  + Putty Keygen to convert pem to ppk

**Security Group**

* Its virtual firewall
* State Full
* 1 instance can have multiple security groups
* 1 security group can be assigned to multiple instances.
* Rules will be applied immediately. Restarting the instance not required.
* Inbound rule will automatically add outbound – State full
* In VPC, Network Access control list, State less, we need to add both inbound and outbound. State less
* Can’t deny the traffic. Everything denied by default. Add rule to allow traffic.
* Can’t block specific IP address, instead use Network access control
* All inbound traffic are blocked by default
* All Outbound traffic are allowed by default

**EBS Volumn**

* Its virtual hard disk
* Snapshot – point in time copy of volume.
* Volumns exists on EBS
* Snapshots exists in S3.
* Snapshots are incremental. This means only the blocks changed since last snapshot will be moved to S3
* First snapshot will takes time
* Snapshots encrypted automatically
* Volumes restored from encrypted snapshots also encrypted automatically
* You can share snapshots, only if unencrypted. This can be shared to other AWS accounts or public
* EC2 and EBS Volumes has to be in same AZ. Always
* Root volume – snapshot
* Modify volume - no downtime
* EBS volume can be changed on the fly. Including Size and Storage type
* From SSD, can’t change to HDD, but HDD can be changed to both SSD and HDD
* Standard (Magnetic) - you can’t modify the volume, others can
* To create / move volume to another AZ, create snapshot of root volume
* From snapshot, create volume, image, copy
* With Copy, you can move it to another region
* TO create snapshot for root, you should stop the instance. however you can create the snapshot when instance is running
* From Snapshot, both volume and image can be created
* AMI are regional, AMI can be launched where it’s stored.
* AMI can be copied to another region, using console, cli and aws EC2 api’s
* Snapshot of RAID array – take an application consistent snapshot. Stop the application and flush all cashes to disk. (Freeze the file system 0r unmount the RAID array or stop EC2 before snapshot) – stop any kind of IO

**Encrypt EBS Volume**

* While creating volume, option to encrypt. Volumes can be attached or detached to running EC2 (Volume and EC2 should be in same AZ)
* Volume from encrypted snapshot will be automatically encrypted
* Volume from non-encrypted snapshot will be not be encrypted
* Root volume will not be encrypted by default
* To encrypt root volume, 2 ways
* 1. OS level like bit locker
* 2. Create snapshot, Copy, while copying to do encrypt, from encrypted snapshot, create image and launch Image. Launching will not supported in free tier
* Encryption – Master Key – Default – aws/ebs
* lsblk – List the block devices
* file –s {/dev/xvdf} - to view the filesystem
* create file system before mount
* create file system – mkfs –t ext4 /dev/xvdf
* to mount go to root and - mount {/dev/xvdf} /filesystem (create filesystem dir in root)
* to unmount – umount –d {/dev/xvdf}

**RDS**

* Relational Database Service
* SQLServer
* Oracle
* MySQL server
* PostgreSQL
* Amazon Aurora
* MariaDB
* Elastic Cache – cache the frequently accessed data
* Supports Memcached, Redis (open source caching engines)
* RDS us OLTP
* RedShift – OLAP
* RDS will always give you DNS. No ip address
* Assign security group. Important to access RDS from EC2
* RDS backup - 1. Automatic backup 2. DB Snapshot
* Automatic Backup - by default, point in time backup in a regular interval, can recover to any point within retention period, retention period is 1 to 35 days. It will take full daily snapshot and will store transactions throughout the day. So when we recover, aws choose most recent daily backup and apply transactions relevant to that day.
* Backup data stored in S3. You will free storage space in S3 equal to your DB size
* During backup window, there will be some latency
* Snapshot – Manual, user initiated. They are stored even after RDS is deleted. (in Automatic, backups will be deleted in RDS is deleted)
* When you restore it will be new RDS instance with new RDS endpoint. (both auto and snapshot)
* Encryption – all RDS supports encryption
* Encryption is done using KMS – AWS Key Management Service
* Once RDS is encrypted, data from snapshot or back up also will be encrypted.
* We can’t encrypt existing RDS instance. we need take snapshot, copy, encrypt and restore
* Multi AZ-RDS – it allows you have an exact copy of your DB in another AZ. AWS does the replication for you. in case of DR, aws automatically failover to backup RDS.
* RDS endpoint is always a DNS name
* **Multi AZ is only for DR not for performance**
* Multi AZ is synchronous
* For performance go for read replica
* Read replica – it allows you to have read only copy of production DB
* Read replica - Asynchronous
* Read Replica – must have auto backup turned on, can have upto 5 read replicas, can have read replica’s of read replica (latency), each replica will have own endpoint, read replica’s can have multi AZ, can be promoted has own DB but this will break replication, can have read replica in second region.
* Read replica can be encrypted
* RDS cannot be paused or stopped. Take the snapshot for future use and terminate
* Supports
  + Auto backup
  + Auto software patching
  + Auto failure detection and recovery
* Scaling is not automated. User has to do some clicks
* When creating RDS, user must specify, multi AZ or not

**Elastic Cache**

* Elastic Cache – cache the frequently accessed data
* Good if your app / db is read heavy work loads – social networking, gaming
* Supports Memcached, Redis (open source caching engines)
* Memcached – Memory object caching system, No persistence, can grow or shrink simila to Ec2 auto scaling, individual nodes are expandable, automatic node replacement and auto discovery. Multi-threaded, no multi AZ capability
* Memcached usecases – caching is primary goal or offload your DB or simple cashing model as possible or planning to run large cache nodes and require multithreaded performance with utilization of multiple cores or you want to ability to scale you cache horizontally as you grow then use Memcached.
* Redis – In memory key value store, supports data structures like sorted set and List. It supports master slave replication and Multi AZ. So it’s used to achieve cross AZ redundancy.
* Redis – replication and persistence features
* Elastic Cache manages redis as a relational database, stateful, supports failover
* Redis usecases – more advance datatypes as lists, hashsets, sets or sorting and ranking datasets in memory like **leaderboards** or persistence of keystore is important or pub sub capabilities or you want to run in multiple AZ with failover then go with Redis.
* Caching strategies
  + Lazy loading
  + Write through
* Lazy loading – loads the data into cache only when necessary. Only at the time its requested
* Cache hit – returns the data
* Cache miss or expired – returns null – so your application needs to get the data form DB and update to elastic cache
* Write through – adds or update to the cache whenever data is written to the database
* Data In the cache is never stale

**EFS – Elastic File System**

* File storage service
* Elastic – grow and shrink based on the need.
* Can share EFS across multiple EC2. EC2 instances should be in same security group as EFS
* Block based storage (S3 is object based storage)
* Read after write consistency
* Pay only what you use
* Scale upto petabytes
* Supports NFSv4
* Supports 1000 of concurrent NFS connections
* Data stored in multiple AZ within the region

**AWS – CLI**

* aws configure
* cd ~, cd .aws, credentials and config file
* aws <service> help
* aws ec2 describe-instances 🡪 will show terminated accounts also
* aws ec2 describe-images 🡪 will show all the available images (both public and private)
* aws ec2 run-instances 🡪 To provision the instance
* aws ec2 terminate-instances –-instance-ids <id>
* role can be assigned to running instance
* With role, acceskey and password not required. Just configure with region
* aws s3 cp –recursive s3://<s3\_bucket> <destination> --region <region where S3 is located>
* Best practice is to use region. Some will work without region

**Bash Scripting**

#!/bin/bash

yum update -y

yum install httpd -y

service httpd start

chkconfig httpd on

aws s3 cp s3://indexbucket-186/index.html /var/www/html --region ap-south-1

**PHP SDK**

* Aws PHP SDK install
* <https://github.com/acloudguru/s3>

**Instance Meta data and User Data**

* curl <http://169.254.169.254/latest/meta-data/>
* ami-id
* ami-launch-index
* ami-manifest-path
* block-device-mapping/
* hostname
* iam/
* instance-action
* instance-id
* instance-type
* local-hostname
* local-ipv4
* mac
* metrics/
* network/
* placement/
* profile
* public-hostname
* **public-ipv4**
* public-keys/
* reservation-id
* security-groups
* services/
* <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html>
* <https://github.com/acloudguru/metadata>

**Cloud Watch**

* Monitoring, Cloud watch – basic monitoring, CPU. Disk,Nw
* Standard monitoring – 5 mins
* Detailed monitoring – for every 1 min (not free)
* Cloud Watch – performance monitoring
* Cloud Trail – Audit trail
* Dashboards
* Alarms
* Events
* Logs

**Elastic Load Balancer - ELB**

* No public IP. Only DNS name. Amazon manages public IP
* Health check. In service, out of service
* It will cost if its running
* Classic load balancer – TCP/IP layer or Http/Https, layer 7 or 4, legacy, not recommended. Important for exam
* Application Load balancer – application layer, operates at layer 7, best suited for http/https, Can look at till application level to decide routing
* Network Load balancer – level 4, most expensive, TCP, extreme performance, can handle millions of requests per second
* Healthy threshold – no of success healthy checks before declare the instance as in service
* UnHealthy threshold – no of failure healthy checks before declare the instance as out of service
* Timeout
* Interval
* Error code 504 – Gateway Time out error. Issue is with application (web server or DB)
* X-forwarded-For Header - Load balancer sending user’s source IP to EC2 in X forwarded for header.
* If you ipv4 public address of end user, look for x-forwarded-for header in classic load balancer.
* To maintain session state or even distribution session in ELB – use Elastic Cashe to store transient session data
* ELB use ports from 1024 to 65535
* You can specify only one subnet per AZ
* Atleast 2 subnets must be specified
* Subnet should have internet access. Private Subnet will not be allowed
* If doing load test on ELB,
  + Ensure to re-resolve DNS before every request
  + Send requests from globally distributed clients or multiple test clients

**Route 53**

* DNS service
* Map domain names to EC2, Load balancer and S3
* Global

**SDK**

* Software Development Kit
* <https://aws.amazon.com/tools/>
* Java, .Net, Node.js, PHP, Python, Ruby, Browser, GO, C++
* AWS IOT
* Android, IOS, React Native, Mobile web, AWS Mobile
* Default Region US-East-1, North Virginia
* Some have default regions (java)
* Some do not (Node.js)

**Lambda**

* Serverless (lambda, S3, DynamoDB, Api gateway). EC2 & RDS is not
* Its compute service, where you can upload the code and create lambda function
* Aws lambda takes care of provisioning and managing servers
* We don’t have to worry about OS, patching, scaling etc
* AWS Lambda – data center, hardware, protocols, OS
* Lamda events can trigger other lamda events
* Can communicate to aws services
* Scale up – increase RAM, memory, storage
* Scale out – spawn EC2 instance based on load
* Lambda scales out automatically. If your lambda run out of memory, you need to upgrade the configuration
* Lambda triggers (eg: api gatway, S3, Cloudfront, dynamoDB) important exam topic, memories all the triggers
* Event Driven – when u upload Meme to S3 lambda can be triggered to add text. After adding text another lambda can be triggered to show to user and another lambda to replicate the meme in another S3
* Api Gateway – when http request comes to api gateway, lambda can be triggered to give the response
* If 1 million requests to API gateway, I million lambda functions will be triggered. Automatically scale out
* Languages – Java, Node.js, python, Go and C#
* Lambda priced based on no of requests
* 1st 1 million requests are free. $.20 per 1 million there after
* Also priced based on duration. Max threshold is 5 mins. (300 seconds) If the function takes more than 5 mins, split into multiple functions default timeout is 3 seconds
* No servers
* Super cheap
* Amazon echo is lambda
* AWS x-ray to debug lambdas
* Lambda can do things globally
* Lambda versioning – version control
* Can have multiple versions of lambda function
* Lambda versioning immutable. Once published can’t be changed
* $LATEST – latest version
* Qualified ARN – the function ARN with the version suffix
* UnQualified ARN – the function ARN without the version suffix
* Alias – can create alias and map the version behind the screen
* Can split traffic using alias to different version. Alias can 50 % to v1 and 50% traffic to v2
* Cannot spilt traffic with $latest
* Lamda default timeout – 3 seconds
* You can set memory in 64 MB increments from 128 MB to 3 GB
* If you choose 256 MB, it allocates approximately twice as much CPU power to your lambda function as requesting 128 MB of memory and half as much CPU power as choosing 512 MB of memory

**API Gateway**

* Fully manages service used to publish, maintain, monitor, and secure api’s at any scale \
* Low cost and scales automatically
* Can throttle API gateway to prevent attacks
* Can log results to cloudwatch
* Configure API gateway :
  + Define an API (container)
  + Define resources and nested resources (URL paths)
  + For each resource, select HTTP method, security, target (EC2,lamabda) and request response transformation
  + Deploy API to STAGE & PROD
* Uses API gateway domain by default. Can use custom domain
* Now supports AWS certificate manager (free SSL TLS certs)
* API caching – cache your endpoint response for the specific TTL (time to live) period in seconds
* Same Origin policy – under the policy, web browser permits, scripts in first web page to access data in second web page. But only if both the web pages have the same origin. This is to prevent cross site scripting (XSS)
* CORS policy – CORS is enforced by the client
* Can import API from an external definition file
* Currently (import) supports swagger v2.0
* API Throttling
  + By default API gateway limits 10000 rps (requests per second)
  + By default Max concurrent requests is 5000 across all api’s with in aws account
  + If you go over 10000 rps or 5000 concurrent requests – you will get 429 –Too many requests
* API gateway can be configured as soap webservice pass through

**Step Function**

* Allows to visualize and test your serverless applications
* Automatically triggers and tracks each step
* Retries if error
* Logs each step – diagnosis and debug easy
  + Sequential steps
  + Branching steps
  + Parallel steps

**X-Ray**

* AWS X-Ray makes it easy for developers to analyze the behavior of their distributed applications by providing request tracing, exception collection, and profiling capabilities.
* Debugging
* Tracing
* Service Map – Visual map
* X-Ray SDK provides,
  + Interceptor – To add your code to trace incoming http requests
  + Client Handler – to call other AWS services
  + Http Client – to call other internal or external http webservices
* X-Ray integrates with following AWS services
  + Elastic Load balancer
  + Lambda
  + API Gateway
  + Ec2
  + Elastic beanstalk
* X-Ray languages supported – Java, Go, Node.js, Python, Ruby, .Net

**S3**

* Simple storage service
* Object based
* Files can be 0 to 5 TB
* Unlimited storage
* Files stored in buckets
* Not suitable for OS or DB
* S3 is universal. Bucket name should be unique
* <https://s3-us-west-1.amazonaws.com/aacloudguru>
* Read after write consistency for PUTS of new objects - for new uploaded file, can be accessed immediately
* Eventual consistency for overwrite PUTS and DELETES – for update and delete files, retrieve is will take little time.
* After update or delete, retrieve is atomic. Which means it will return either updated file or old file. it will not return partial or corrupted file (reason, sync with all AZ)
* S3 contains
  + Key – name of the object
  + Value –data (sequence of bytes)
  + Version Id – versioning
  + Meta data
  + Sub resources – bucket specific configuration (policies, access control lists), CORS, transfer acceleration
* Built for 99.99 % availability
* Amazon guarantees 99.99% availability
* Amazon guarantees 99.999999999% durability (remember 11 x 9 s)
* Tiered storage available
* Life cycle management
* Versioning
* Encryption
* Secure your data – access control list and bucket policies

S3 storage tiers / classes

* S3
  + 99.99 % availability
  + 99.9999999999 % durability
  + Designed to sustain the loss of 2 facilities concurrently
  + Currently cost effective
* S3 – IA
  + Infrequently accessed
  + But rapid access when needed
  + Lower fee then S2 and but charged retrieval fee (every retrieval)
* S3 – One Zone IA
  + Same has IA but data stored in single AZ
  + 99.5 % availability only
  + 99.9999999999 % durability
  + Cost is 20% less the S3-IA
* Reduced Redundancy Storage
  + 99.99 % availability
  + 99.99 % durability only
  + Used for data that can be created if lost
  + Ex: thumbnails
  + Low cost
  + Not recommended
* Glacier
  + Archiving
  + Not part of S3, but linked
  + Very cheap
  + 99.99 % availability
  + 99.9999999999 % durability
  + Infrequently accessed
  + No real time access
  + it takes 3 to 5 hours to restore data from glacier
* charged for
  + Storage per GB
  + No of request (get, put, copy , delete etc
  + Storage management – inventory, Analytics, Object tags
  + Data Management pricing – Data transferred out of S3
  + Transfer Acceleration – cloudfront
* S3 – Security
  + BY default – private
  + Access control
    - Bucket policies – applied at bucket level, json, applicable to objects in bucket
    - Access control lists – applied at object level
* Can configure to create access logs, which log all requests to S3. These logs can be written into another bucket
* S3 Policies – policy generator
* S3 Encryption
  + In Transit – SSL/TLS
  + Type of block cypher – Advanced encryption standard 256 – AES 256
  + At Rest, Server side encryption
    - SSE – S3 - S3 managed keys, AWS manage the keys and enc/dec – AES256
      * Strong multi factor encryption
      * S3 encrypts each object with unique key
      * It encrypts key itself with master key that it regularly rotates
    - S3-KMS - Key Management Service, AWS manage the keys. Additional benefits. Can track the encryption, decryption activities
    - SSE-C - Customer provided keys. AWS will do enc/dec based on the keys provided by customer
  + Client side encryption – client will encrypt the object and upload
* To enforce encryption, add bucket policy to deny all PUT requests if the header doesn’t have **x-amz-server-side-encryption** tag
* Every upload is PUT request to S3
* **x-amz-server-side-encryption –** if this tag is included in header of PUT request, it tells s3 to encrypt the object with the specified encryption method.
* x-amz-server-side-encryption:AES256 – this means SSE-S3, S3 managed keys
* x-amz-server-side-encryption:ams:kms – this means SSE-KMS, KMS managed keys
* Expect:100-continue – S3 will reject if the content type is invalid
* CORS – Cross Orgin Resource Sharing. To share the objects from one bucket to another
* Static website hosting - <bucket-name>.s3-website-<region>.amazonaws.com
* S3 versioning
  + Once enabled ypu cant disable. You can only suspend it
  + Version no for objects before enabling versioning will be null
* S3 life cycle management
  + Eg : Move the file to S3-IA after 30 days and move to Glacier after 60 days
  + Can delete permenantly after x days
  + Can be applied to current as well as previous version
* Tp upload file more than 5 GB – Multipart upload API (recommended for more then 100MB)
* Multipart allows stop and resume uploads
* Single put max size is 5 GB
* Multi Object Delete – to delete large number of objects, no additional charge
* By default 100 buckets per aws account. Contact aws for more
* S3 Naming
  + 3 to 63 chars long
  + Only lower case
  + Number, period, dash allowed
  + Must start with lower case letter or number
  + Underscore, end with dash, consecutive period, dash adjacent to period not allowed
  + IP address format not allowed
  + Key names are stored lexicographically (alphabetical order)
* S3 bucket ownership is NOT transferable. Owner can grant access to others
* Referrer policy – to allows access from specific domains only
* 409 conflict error – when bucket you trying to delete is not empty through api. But through console, you can delete the non-empty bucket

**CoundFront**

* CDN
* Can be used to deliver static, dynamic, streaming and interactive content
* Geographically disbursed data centers
* Edge location – location where content is cached. Different from Region / AZ
* Edge locations are not just read only. We can put the objects to sync to origin.
* Edge Location will cache the content for the first time use, and serve the content from edge location for the subsequent requests. Request will not goto server. Improve latency
* TTL - time to live – objects are cached for the life of TTL
* Manually clearing cache is allowed. But its chargeable
* Origin – Origin of the files, CDN will distribute. This can be S3 or Ec2 or Elastic Load balancer or Route 53, Can work with non aws services as well
* Distribution – name of CDN, which consists of collection of edge locations
  + Web distribution - for websites, http/https
  + RTMP - for media streaming. Audio/Video
* S3 transfer acceleration make a use of cloudfront to accelerate transfer from S3 to end user. Say for example, if geographically distributed users are uploading multiple files to S3 which is located in UK region. Instead of uploading the files to S3 directly, they can upload to their nearest edge location and then edge locations accelerate the transfer to S3 trough optimized aws network.
* Restrict Bucket Access – bucket can be accessed only via CloudFront
* Origin access identity
* **Grant Read Permission on Bucket. By default No. change it to Yes, update the bucket policy. Important step.**
* Protocol policy – http or https, https only or convert http to https
* Restrict Viewer Access – Signed URL or Signed cookies – for specific paid content
* Price class – use all edge locations or specific
* Alternate domain names
* SSL certificate – default or custom (upload custom cert to aws cert manager)
* Enable IPv6
* Blocklist or whitelist specific geo locations
* Invalidate – manually invalidating the content to remove the content from the cache (chargeable)
* S3 performance optimization
  + GET-intensive workload
    - If most of the workload is GET, use cloud front to optimize the performance
  + Mixed workload
    - Means mix of GET, PUT, DELETE etc
    - Avoid sequential key names for S3 objetcs
    - S3 uses key name to determine the which partition an object stored
    - For heavy workload, this might cause i/o issues
    - So add random prefix to key names, to prevent multiple objects stored in the same partiti
* [**https://aws.amazon.com/s3/faqs/**](https://aws.amazon.com/s3/faqs/)

**Storage Gateway**

* To backup the data

**SnowBall**

* Snowball
* Snowball Edge
* SnowMobile
* Snowball can import to S3 and export from S3

**Dynamo DB**

* No SQL DB
* You can’t specify AZ when you create DynamoDB table
* Supports both document and Key-Value data models
* Stored SSD storage
* Spread across 3 geo locations
* Choice of 2 consistency models
  + Eventual consistent reads (default) – all copies of data will reach all the data centers within second. Read after short time will always return the updated data – best read performance
  + Strongly consistent reads
* It uses conditional writes for consistency. For PutItem, DeleteItem, UpdateItem operations. Operations succeeds only if attributes meet one or more conditions. Otherwise it returns error.
* It uses optimistic concurrency control – optimistic locking – its strategy – your DB writes are protected from being overwritten by writes of others and vice versa
* Supports atomic counters – all write requests are applied in the order they received
* Tables
* Items – row
* Attributes – key-value
* Json, html or XML
* Stores and retrieves data based on primary key
* 2 types of primary key
  + Partition key – key will determine which partition the data is stored. Unique, eg: userId, productId
  + Composite Key – Partition Key + Sort Key, situations where partition key cannot be unique, E.g.: same user adds multiple entries in online forum. Here composite key will be user id + timestamp when the post posted
* Partition Key – Hash
* Sort Key - Range
* Access control can be managed via AWS IAM. Create user or Role to manage dynamoDb. Can also use special iam condition to restrict user access to only their own records. (Partition id = userId) – IAM policy parameter : dynamodb:LeadingKeys
* Has index features, 2 types of index
  + Local secondary index
    - Can be created only at the time of table creation. Cannot modify later
    - It has same partition key as your original table
    - But different sort key
  + Global secondary index
    - can be created at anytime
    - can have different partition key and different sort key
* scan vs query
* Query –
  + Query by partition key or partition key + sorted key.
  + By default returns all the attributes.
  + You can use the project expression parameter to return only the specific parameters
  + By default sorted by sort key – numeric ascending – 1 2 3 4
  + Sorting order can be reversed by setting ScanIndexForward parameter to false. This is applicable only to query not for scan
  + By default – eventually consistent
  + Query is more efficient then scan
* Scan
  + Returns entire table.
  + By default returns all the attributes.
  + You can use the project expression parameter to return only the specific parameters
  + Filter = to refine the results
  + Scan – dumps the entire table and add the filter – filter is extra step to remove the data
* To improve performance
  + Avoid scan
  + Use page size
  + Parallel scan – by default scan process sequentially, get 1 MB of data and retrieve next 1 MB, can configure parallel scan by logically dividing table or index into segments and scan each segment in parallel. Avoid parallel if your table already busy
* Provisioned throughputs – is measured by capacity units
* Pricing based on the capacity units
* When you create a table, you specify your requirements in terms of read capacity units and write capacity units
* If your application reads or writes larger items it will consume more capacity units and it will cost you more as well
* I read capacity unit represents 1 strongly consistent or 2 eventually consistent read per second for items upto 4 kb
* Write capacity unit –
  + 1 \* write capacity unit = 1 \* 1 KB write per second
* Read capacity unit
  + 1 \* read capacity unit = 2 \* eventually consistent reads of 4kb per second (by default)
  + 1 \* read capacity unit = 1 \* strongly consistent reads of 4kb per second (by default)
* If table with 5 read capacity units and 5 write capacity units then
  + 5 \* 4kb strongly consistent reads = 20 kb per second
  + 5 \* 2 \* 4kb eventually consistent reads = 40 kb per second (twice as strongly consistent)
  + 5 \* 1kb writes = 5 kb per second
* To calculate capacity units
* If your application need to read 80 items (table rows) per second and each item is 3 kb size
  + For strongly consistence reads
    - Calculate how many read capacity required for each read –
    - size of each item /4 kb
    - 3kb / 4kb = 0.75 – round to 1
    - For 80 items = 80 read capacity units required
  + For eventual consistent reads
    - Double the throughput of strongly consistence reads
    - Divide 80/ 2
    - 40 read capacity units required
* If your application want to write 100 items per second and each item is 512 bytes in size
  + Calculate how many capacity units for each write
  + Size of each item / 1 kb
  + 512 bytes / 1 kb = 0.5 = round to 1
  + For 100 items = 100 capacity units required
* Dynamo DB Accelerator – DAX
  + Fully managed, clustered, in-memory cache for DynamoDb
  + More for read only purpose
  + Delivers upto 10x read performance improvement
  + Ideal for read –heavy and burst workloads
  + E.g. auction apps, gaming, retails sites during black Friday promotions
  + It’s a write – through caching service – means data written to the cache and as well as back end store at the same time
  + Allows you to point your dynamoDb api calls at the DAX cluster
  + If the item available in cache – cache hit
  + If the item not available in cache – cache miss
  + In cache miss – DAX performs eventually consistent getItem operation against dynamoDB and update cache and return
  + DAX reduces read load on dynamoDB
  + May be able to reduce provisioned read capacity and save money
  + Not suitable for
    - Strongly consistent reads
    - Write intensive applications
    - Apps do not perform many reads
    - Apps do not require micro second response time
* getItem api
  + returns set of attributes for the item with the given primary key
  + eventual consistent read by default
  + set consistentRead to true for strongly consistent
* for any aws account, limit is 256 tables per region – contact aws for more
* cumulative (concurrent) number of tables and indexes in the CREATING, UPDATING, DELETING state cannot exceed 10 – if so you will receive “LimitExceededException”
* If you want to create more than one table with secondary index you must do sequentially. Create one wait till it became active and proceed to next
* Global secondary index is index with hash and range key that can be different from those on the table
* You can create upto 5 global secondary index when you create table
* Each table can have upto 5 local secondary indexes
* So totally 10 secondary index. Cant increase the limit beyond 10
* Number. String, Boolean data types can be indexed
* Set, list, map types cannot be Indexed
* Max limit of item collection is 10 GB
* Smallest amount of capacity unit can be purchased is 100 (both reads and writes)
* Max size of item in dynamoDB = 400 kb
* Number if attributes item can have = no limit, but total size including attribute names and values should not exclude 400 KB
* Doesn’t support cross table joins (core RDS features not supported)
* ItemCollectionSizeLimitExceededException – for a table with local secondary index, exceeded the maximum size limit of 10 GB
* Scan is always eventually consistent
* Secondary index are optional
* If you do more reads/writes then provisioned capacity units, requests will be throttled and you will receive 400 error code

**KMS**

* Key management service
* Keys are region based.
* Customer Master Key - CMK
  + Alias
  + Creation date
  + Description
  + Key state
  + Key Material (customer provided or aws provided)
* Can never be exported. Keys in HSM can be exported
* aws kms encrypt
* aws kms decrypt
* aws kms re-encrypt
* aws kms enable-key-rotation
* Envelope encryption key – data key – encrypted customer master key (CMK)
* Encrypt the data – get the envelop key by encryption cmk and encrypt the data using envelop key
* Decrypt the data – decrypt the envelop key using CMK and decrypt the data using decrypted master key
* key cannot be deleted immediately. Need to disable the key and schedule for deletion (7 to 30 days)

**SQS**

* simple queue service
* first, oldest aws service
* Pull based system
* Massages can contain upto 1 kb to 256kb of text in any format
* For message size larger then 256Kb use Amazon SQS Extended client library for Java – this library lets you to send a sqs message that contains reference of object in S3 that can be large as 2 GB
* Types
  + Standard queues (default) – order is not guaranteed, message delivered at least once
  + FIFO queues (First – In – First – Out) – order guaranteed. Limited to 300 tps, message delivered once, no duplicates
* Messages can be kept in queue from 1 min to 14 days
* Default retention period is 4 days
* Visibility Timeout – amount of time the message is invisible in sqs queue. If the message once pulled it will be invisible. If the message is processed within visibility timeout, message will be deleted from sqs otherwise message will became visible for next thread to process
* ChangeMessageVisibility – api to extend visibility timeout
* Default visibility timeout is 30 seconds, max is 12 hours
* Short polling – will poll for messages frequently and return immediately with empty response if the sqs is empty
* Long polling – doesn’t return response until the message comes or long poll time out, can save your money
* To enable long polling, set ReceiveMessageWaitTimeSeconds to greater then 0 or equal or less then 20
* Max long poll time out – 20 seconds
* Can use JMS
* No limit on the number of Queues
* Can configure access IAM policy that allows anonymous users to access the queue
* Free tier provides 1 million requests per month at no charge
* SQS is PCI DSS level 1 certified
* When consumes receives and processes message from queue, message remains in the queue after the visibility time out. Amazon doesn’t automatically delete the message. Since it’s a distributed system and there no guarantee component processed the message. It will be deleted only after the retention period.
* So consumer must delete the message from queue – DeleteMessage API
* Dead letter queues – can target for messages which cant processed successfully
* User can delete the queue at any time, whether its empty or not
* Queue Name
  + Limited to 80 characters
  + Alphanumeric, -, \_ allowed
  + Unique within aws account

**SNS**

* Simple notification service
* Topic
* SMS, EMAIL, SQS, email
* Push based system
* To send notifications from cloud
* Push notifications
* Also can deliver notifications by text, email, sqs or http endpoint
* SNS can trigger lambda functions – when message published sns topic that has lambda function subscribed, then lambda function will be invoked with the payload of the message published
* All the messages published to SNS topic will be stored redundantly across multiple availability zones
* Pay as you go model
* Pub – sub model
* O.50$ per 1 million amazon SNS requests
* Fanout pattern – message published to SNS topic is distributed to number of SQS in parallel, by using this pattern, can build the application can process parallel asynchronous processing.
* SNS message body would have
  + Type
  + MessageId
  + Subject
  + Message
  + Signature Version
  + Signature
  + SigningCertURL
  + UnSubscribeURL
* SNS to send notification to mobile endpoints (whether its direct or subscription based, first we need to register the app with aws). To register mobile with AWS,
  + Enter name to represent your app
  + Select platform
  + Provide your credentials for the notification service to platform
  + After registration, create an endpoint for the app and mobile device
  + Then endpoint will be used in SNS to send the notifications.
* If SNS distribute the messages to SQS then, below attributes should not be empty or null
  + Name
  + Value
  + Type
  + MessageBody
* To receive messages published to topic, you have to subscribe to the endpoint of the topic
* Topic name
  + Should be unique within aws account
  + Limited to 256 characters
  + Alphanumeric, -, \_ are allowed
* Subscription requests are valid for = 3 days for confirmation
* Once message published cannot be recalled
* CreatePlatformEndpoint API – to register multiple device tokens
* Platform supported
  + GCM – Google Cloud Messaging
  + APNS – Apple push notification service
  + ADM – Amazon device messaging
  + WNS – Windows Push notification service
  + MPNS – Microsoft Push notification service
  + Baidu cloud push – android in china

**SWF**

* Workers
* Deciders
* They can run independently and scale quickly
* SWF ensures task is assigned only once and never duplicated
* Guarantees the delivery order
* SWF domains – isolate set of types, executions, tasks from others within aws account
* Can register domain by console or registerdomain action in swf api
* Json format
* Maximum workflow can be one year, value always measured in seconds
* Decision task occur when the state of the workflow changes
* Ec2 can perform worker task
* Server resides outside aws can perform EC2 task
* Humans can perform activity task not decision task
* Max 100 SWF domains
* Max 10000 workflow and activity types (in total)
* Use case – Video encoding

**SES**

* Simple Email Service
* Email only
* To send marketing, notification, and transactional emails
* Can also use to receive emails. Automatically delivered to S3 bucket. Can trigger lamba and SNS
* Use cases –
  + Automated emails
  + purchase confirmations, shipping notifications, order status updates
  + Marketing communications, advertisements, newsletters, special offers
* Not subscription based. All you need to know is email address

**Kinesis**

* To send your streaming data
* Kinesis streams –
  + Video streams – securely stream video from connected devices to AWS for analytics and machine learning
  + Data Streams – build custom applications to process data in real time
  + Producer will send the data to Streams (Producers – Ec2, mobile, laptop or dedicated server)
  + Stream will have shards
  + In shards data can be retained for 24 hours (default )
  + Can change upto 7 days
  + And then consumes will consume the data to analyses it (consumer – Ec2)
  + And send it S3 or dynamoDb or Redshit or EMR
* Kinesis Firehose
  + Capture, transform, load data streams into aws data stores for near real time analytics with BI tools
  + No shards, no consumers, no retention period – don’t have manage. Automatic
  + Analyze the data within and automatically uses lambda
  + Analyze is optional – its within firehose
  + Producers will send the data to Firehose
  + Firehose will send to S3 or Red shift
* Kinesis Analytics
  + Allows you run sql queries on the data (sent to Firehose ot Streams) and the result data can be stoted in S3, Redshift

***Elastic Bean Stalk***

* Java, .net, php, node, python, ruby, go and docker
* Apache tomcat, Nginx, passenger , Puma and IIS
* Tomcat for Java
* Apache HTTP server for PHP & Python & Node js
* Nginx also node js
* Passanger or puma for ruby apps
* Microsoft IIS for .Net
* Will handle deployment, capacity provisioning, load balancing, auto scaling and application health
* You still retail full control of underlying amazon resources
* You pay only for the resources required to store and run your application
* Integrated cloud watch and X-Ray
* Deployment policies
  + All the once
    - Deploys the new versions to all instances simultaneously
    - All of your instances will be out of service during the deployment. Outage
    - If the update fails, you need rollback by redeploying the previous version to all the instances
    - Not suitable for critical prod environments
  + Rolling
    - Deploys new version in batches
    - Each batch of instances will be out of service during the deployment
    - Your environment capacity will be reduced by the number of instances in a batch during the deployment
    - If the update fails, you need to perform additional rolling update to revert the changes
    - Not suitable for performance sensitive systems
  + Rolling with additional batch
    - Launces additional batch of instances
    - Deploys the new version in batches
    - Maintains full capacity during the deployment
    - No downtime
    - If the update fails, you need to perform additional rolling update to revert the changes
  + Immutable
    - Deploys the newer version to fresh group of instances in their own auto scaling group
    - When the new instances passed the health checks, they are moved to existing auto scaling group and the old instances are terminated
    - Full capacity during deployment and no down time
    - If the update fails, just terminate the new instances
    - Suitable for mission critical production systems
  + Elastic beanstalk configuration
    - You can define packages to install
    - Create linux users and groups
    - Run shell commands
    - Enable service
    - Configure load balancer
    - Files are written in yaml or json
    - Extension - .config
    - Saved inside the folder .ebextensions
    - Name can be anything
    - .ebextensions folder must be included in the top level directory of your app source code bundle
* RDS can be integrated with EBS is two ways
  + Can launch RDS instance with in EBS. RDS instance will be created with in EBS environement
    - Good option for Dev and Test
    - If you terminate EBS, RDS instances also will be terminated. So not suitable for prod
  + Launch RDS outside EBS and integrate with EBS
    - Decouple RDS and EBS
    - Additional security group must be added
    - Need to provide connection string
* Ec2, RDS, ELB, S3, SNS and Auto scaling group can be deployed in beanstalk (SQS – cant)
* AWS toolkit for eclipse – to update running app in beanstalk

**Code Commit**

* Private GIT repository
* Regional
* Data is encrypted in transit and in rest (TLS)
* Events (pull, comment, commit comments) can be notified through SNS topic
* Connection type – SSH, HTTPS
* As roor cannot configure to connect with SSH
* To connect
  + Install GIT
  + Install AWS CLI
  + IAM user with CodeCommit policy
  + Configure the user access key and secret access key in AWS CLI
  + And GIT commands

**Code Deploy**

* Automated deployment service
* Allows you deploy the code automatically to EC2, on premises systems and lambda functions
* Auto scales with your infrastructure
* Integrates with various CI/CD tools, Jenkins, GitHub, Atlassian and AWS code pipeline
* Also config management tools like Ansible, Puppet and Chef
* Two deployment approaches,
  + In-place
    - Rolling update
    - Deploys new version in batches
    - Each batch of instances will be out of service during the deployment
    - Your environment capacity will be reduced by the number of instances in a batch during the deployment
    - If the instances are behind the load balancer, configure load balancer to stop the traffic to the instances where the deployment is going on
    - It can only be used in EC2 and on-premise systems
    - Not supported for AWS lambda
    - If the update fails, you need to perform additional rolling update to revert the changes
  + Blue/Green
    - Immutable
    - Blue – active deployment
    - Green – New Release
    - Deploys the newer version to fresh group of instances in their own auto scaling group
    - When the new instances passed the health checks, they are moved to existing auto scaling group (elastic load balancer) nand the old instances are terminated
* Deployment Group – Set of EC2 instances, lambda functions to which new version is to be deployed
* Deployment – process
* Deployment configuration – rules, success / failure conditions
* AppSpec File – defines the deployment actions you want AWS code deploy to execute. Must be placed root directory of your revision
* Revision – everything needed to deploy
* Application
* Lambda deployment – AppSpec file
  + AppSpec file – used to define the parameter will be used for codedeploy deployment – appspec.yml
  + For lambda deployment, AppSpec file – yaml or json
  + Version – reserved for future release, currently only 0.0 is allowed
  + Resources – name and properties of the lambda function to deploy
    - Type
    - Properties
      * Name
      * Alias
      * CurrentVersion
      * TargetVersion
  + Hooks – specifies lambda function to run at set points in the deployment lifecycle to validate the deployment
    - BeforeAllowTraffic – used to specify the tasks or functions you want to run before allowing the traffic to your deployed lambda functions
    - AfterAllowTraffic – vice versa
* EC2 / OnPremises deployment – AppSpec file
  + Supports only yaml
  + Version – reserved for future release, currently only 0.0 is allowed
  + Os – windows or linux
  + Files – location of application files need to be copied and where to be copied,
    - source
    - destination
  + Hooks – specifies scripts to run at set points in the deployment lifecycle to validate the deployment
    - BeforeInstall
      * Location
      * timeout
    - AfterInstall
    - AppicationStart
    - ValidateService
      * Runas
* Hooks order of a code deploy – In place deployment
  + 1
    - BeforeBlockTraffic
    - BlockTraffic
    - AfterBlockTraffic
  + 2
    - ApplictionStop
    - DownloadBundle
    - BeforeInstall
    - Install
    - AfterInstall
    - ApplicationStart
    - ValidateService
  + 3
    - BeforeAllowTraffic
    - Allowtraffic
    - AfterAllowTraffic

**Code Pipeline**

* Fully managed CI CD service
* Pipeline workflow
* Can be integrated with code commit, code deploy etc
* Workflow can be automatically triggered for every push
* Eg : user adds the code to S3 🡪 cloud watch trigger the code pipe line – code pipe line trigger code deploy 🡪 code deploy get the code from S3 and deploy to EC2
* Code pipeline can source the code from S3 or Code Commit or Git Hub – source provider
* Build provider – no build, Jenkins, AWS code build or Solano CI
* Deployment provider – no deployment, ECS, cloudformation, code deploy or elastic beanstalk

**VPC**

* Virtual private Cloud
* By default 5 VPC’s can be created per region. For more need to contact amazon
* Public subnet
* Private subnet
* When we create custom subnet, by default they can talk to each other across AZ
* VPC Subnet can have multiple route tables
* 1 subnet = 1 AZ
* Internet Gate way - only one internet gateway per vpc
* Virtual private gateway – VPN
* Route Table
* Security Group – it exists only with in the VPC, its attached with VPC
* ACL – Access control List
* NACL, SNACL – Subnet Network ACL
* Security Groups – Stateful (if you add inbound rule, automatically outbound is added)
* NACL – StateLess (need to add both inbound and outbound rule)
* Default VPC
  + All subnets in default vpc have route to internet
  + Each EC2 will have private / public ip address. Custom VPC – EC2 have only private ip
* Can have multiple VPC’s in one account
* **http://cidr.xyz/**
* VPC peering
  + Allows you to connect one VPC to other via direct network route using private ip address
  + Can peer VPC’s in other aws accounts also
  + Peering is star configuration, 1 central vpc peers with 4 others
  + **No transitive peering** – all 4 vpc peers thorough centeral vpc. 4 vpc will not not peer directly
* AWS reserves 5 ip addresses in a subnet. First 4 and last one in each subnet CIDR block
  + For subnet with CIDR block 10.0.0.0/24 following 5 are reserved
  + 10.0.0.0 – Network Address
  + 10.0.0.1 – VPC router
  + 10.0.0.2 - DNS
  + 10.0.0.3 – for future
  + 10.0.0.255 – network broadcast
* For public subnet – enable auto assign public ip address to yes
* IGW – internet gateway can be attached with only one VPC – always
* Route Table – attach IGW with IPv4 and IPv6 for internet access – for public subnet
* IPv4 – 0.0.0.0/0
* IPv6 - ::0/0
* For private subnet, attach NAT instance or NAT GateWay In route table
* NAT – Network Address Translation
* Baston server – jump server
* NAT instance
  + EC2 instance search NAT
  + by default EC2 will check source and destination. Disable for NAT instance
  + you have to manage
  + use script to manage failovers
  + attached with security group
  + must be in public subnet
* NAT Gateway – Ipv4
  + In VPC, create NAT GateWay and attach public subnet
  + Highly available
  + Create in each AZ
  + AWS managed
  + Security group not required
  + Scale automatically
  + More secure
  + use ports from 1024 to 65535
* Egress only internet Gateways – ipv6
* NACL
  + Subnet can be associated with one NACL always. But NACL can have multiple subnets
  + If a subnet is associated new another NACL 2, existing association with NACL 1 will be automatically removed
  + NACL can be attached with one VPC
  + Default NACL will be created with VPC – by default it allows all traffic
  + Custom NACL (when you create) – by default it deny all traffic
  + Each subnet in VPC must be associated with NACL. If not subnet will be associated with default NACL
  + Rule 100 – ipv4
  + Rule 101 – ipv6
  + Amazon recommends to start rule from 100
  + StateLess (need to add both inbound and outbound rule)
  + Rules are validated in numerical order. If rule 99 is deny and rule 100 is allows, as per 99 traffic will be denied
  + Can add a rule to deny traffic from particular IP range. Can’t do with security groups
* VPC flow logs
  + Capture information about the IP traffic going to and from – VPC
  + Stored in cloud watch logs
  + Can be created at 3 levels
    - VPC
    - SubNet
    - Network interface level
  + You cannot enable flow log for VPC’s that are peered with your VPC’s unless peer VPC is in your account
  + You cannot tag a flow log
  + After created the flow log, cannot change the configuration. (cannot change the role)
  + Not all IP traffic is monitored. ( below will not be monitored)
    - Traffic generated by your instances when they contact amazon DNS server. If you use your own DNS server, it will be monitored
    - Traffic generated by windows instance for amazon windows license activation
    - Traffic to and from 169.254.169.256 meta data
    - DHCP traffic
* VPC endpoint – to access AWS resources via virtual private gateway (IGW not required)
  + Interface
  + Gateway
* If you want to multiple apps (different ip’s) on a single EC2
  + Launch a vpc instance with 2 network groups
  + Assign elastic IP
  + Assign separate security groups

**CloudFormation**

* Template – template
* Stack – provisioned
* Json and yaml
* Mandatory fields – aws resource
* Options fields
  + input parameters (eg : tags), limit is 60
  + Output parameters (eg:public ip), limit is 60
  + Version – latest template version is 2010-09-09. If you don’t specify, aws will assume the latest version
* Fn:GetAtt – to output data
* Fn::join – appends set of value into single value separated by delimiter, id delimiter is empty, just concat
* Fn::FindInMap – returns the value corresponding to keys in a two level map which is declared in mappings section
* Fn:Select – returns single object from a list of objects by index
* By default, “automatic rollback on error” is enabled
* You are charged for errors
* Cloud formation is free. You need to pay only for resources
* Waitcondition - stacks can wait for apps to be provisioned
* Route53, IAM role is supported
* You can create A records and aliases
* Templates – No limit
* Stacks per account – 200
* Stack can be increased beyond 200 by contacting AWS
* ListStackResources – API to list all the resources in a stack
* ListStacks – list all the stacks including deleted (90 days)
* DescribeStacks – lists only running Stacks
* For deleted Stacks, ListStackResources will returns upto 90 days from the day deleted
* Provides python helper scripts – to install software, start service in EC2. You can call the helper script directly from your template. This will be executed in Ec2 as part of stack creation process
* Can be used to bootstrap Chef and Puppet
* Data can be saved before deleting the stack by defining delete policy

**Route 53**

* Global service
* IPv4 – 32 bit
* IPv6 – 128 bit, 340 undecillion addresses
* NS records – Name Server records
* A record – Address record – used by computer to translate the name of the domain to the IP address.
* ELB always use DNSName – no ip
* TTL – Time to live – your computer always cashes the ipv4 address of domain name. by default TTL is 2 days. So everytime you hit DNS, it will search in local cache, and then does dns look up and update the cache. If you planning DNS migration, drop the ttl to 5 mins, so that your computer refresh the cache for every 5 mins
* CName – Canonical Name – used to resolve one domain name to another. You can use <http://mobile.aacloud.com> to <http://m.aacloud.com> so users can use both and its points to same dns
* Alias records
  + Similar to CName
  + Diff is you cant use Cname for naked domain names
  + Naked domain name – zone apex record, no www, <http://acloud.guru>
  + CName is chargeable
* Always choose alias records over CName
* Routing Policy
  + Simple
  + Weighted –
    - let you to split the traffic,
    - 95% traffic to region 1 and 5 % traffic to region 2 or 2 different ELB’s
    - Need to provide weightage 1- 255
  + Latency –
    - let you send traffic to region which has less latency (region near to user) t.
    - Eg 1 elb in india and one in London. If users access from india, latency routing take him to india elb since London elb will have more latency
    - Need to select region
  + Failover
    - Let you to create active passive setup
    - If active region health check fails traffic to passive region
    - Need to add health check of your target (ELB)
  + Geolocation
    - Let you to send traffic based on the geolocation of the user
    - Eg: EU customers will be sent to London and US customers will be sent to Mumbai
    - You need to select location. Continent or Country or state or default (everywhere else)
* Supports SSL termination – all regions
* New instances can be added on the fly. No need to stop ELB

**AutoScaling**

* Free of cost
* Can autoscale
  + EC2 autoscaling groups
  + Aurora DB clusters
  + DynamoDB tables
  + DynamoDB global secondary index
  + ECS
  + Spot fleet requests
* Ways
  + Ec2 auto scaling
  + Application auto scaling api
* EC2 Auto Scaling can also detect when an instance is unhealthy, terminate it, and launch an instance to replace it.

**AWS Regions**

|  |  |  |  |
| --- | --- | --- | --- |
| **Region Name** | **Region** | **Endpoint** | **Protocol** |
| US East (Ohio) | us-east-2 | rds.us-east-2.amazonaws.com | HTTPS |
| US East (N. Virginia) | us-east-1 | rds.us-east-1.amazonaws.com | HTTPS |
| US West (N. California) | us-west-1 | rds.us-west-1.amazonaws.com | HTTPS |
| US West (Oregon) | us-west-2 | rds.us-west-2.amazonaws.com | HTTPS |
| Asia Pacific (Tokyo) | ap-northeast-1 | rds.ap-northeast-1.amazonaws.com | HTTPS |
| Asia Pacific (Seoul) | ap-northeast-2 | rds.ap-northeast-2.amazonaws.com | HTTPS |
| Asia Pacific (Osaka-Local) | ap-northeast-3 | rds.ap-northeast-3.amazonaws.com | HTTPS |
| Asia Pacific (Mumbai) | ap-south-1 | rds.ap-south-1.amazonaws.com | HTTPS |
| Asia Pacific (Singapore) | ap-southeast-1 | rds.ap-southeast-1.amazonaws.com | HTTPS |
| Asia Pacific (Sydney) | ap-southeast-2 | rds.ap-southeast-2.amazonaws.com | HTTPS |
| Canada (Central) | ca-central-1 | rds.ca-central-1.amazonaws.com | HTTPS |
| China (Beijing) | cn-north-1 | rds.cn-north-1.amazonaws.com.cn | HTTPS |
| China (Ningxia) | cn-northwest-1 | rds.cn-northwest-1.amazonaws.com.cn | HTTPS |
| EU (Frankfurt) | eu-central-1 | rds.eu-central-1.amazonaws.com | HTTPS |
| EU (Ireland) | eu-west-1 | rds.eu-west-1.amazonaws.com | HTTPS |
| EU (London) | eu-west-2 | rds.eu-west-2.amazonaws.com | HTTPS |
| EU (Paris) | eu-west-3 | rds.eu-west-3.amazonaws.com | HTTPS |
| South America (São Paulo) | sa-east-1 | rds.sa-east-1.amazonaws.com | HTTPS |

If you don’t explicitly specify endpoint, US West (Oregon) is default endpoint

**Acronym**

IOPS – Input Output per Second

SSD – Slot State Drive

AMI – Amazon Machine Instance

HVM – Hardware Virtual Machine (X – Para virtualization)

NFSv4 – Network File System V4

OTLP – Online Transaction Processing

OLAP – Online Analytics Processing

SAML – Security Assertion Markup language

CORS – Cross Origin Resource Sharing

RTMP – Real Time Messaging Protocol

HSM – Hardware Security Module ??

API – application programming interface

rps - requests per second

DAX - Dynamo DB Accelerator

NACL – Network Access control List

CIDR - Classless Inter-Domain Routing

NAT – Network Address Translation

**Limits**

* S3
  + 100 buckets per account – can increase by contacting AWS
  + No limit
  + 1 object – 0 bytes t0 5 TB
  + Single put max size is 5 GB
  + Amazon recommends Multipart upload for more then 100 MB
* DynamoDB
  + 256 tables per region – can increase by contacting AWS
  + 5 local secondary index – cannot increase secondary index (both local & global)
  + 5 global secondary index (so total is 10)
  + Max limit of item collection is 10 GB
  + Smallest amount of capacity unit can be purchased is 100 (both reads and writes)
  + Max size of item in dynamoDB = 400 kb
  + Number if attributes item can have = no limit, but total size including attribute names and values should not exclude 400 KB
* SWF
  + Max 100 SWF domains
  + Max 10000 workflow and activity types (in total)
  + SWF workflow can live upto 1 year
  + Maximum open activity tasks - 1000
* SNS
  + Topic name
    - Should be unique within aws account
    - Limited to 256 characters
    - Alphanumeric, -, \_ are allowed
  + Subscription requests are valid for = 3 days for confirmation
  + 100,000 topic per account
  + 10 million subscription per topic – contact aws for more for both
* SQS
  + No limit
  + 1 million request per month – free tier
  + Message size, 1 kb to Max 240 KB
  + Retention – 1 min to 14 days
  + Default retention – 4 days
  + Visibility time out 30 seconds to 12 hours
  + Long poling max – 0 to 20 seconds
* CloudFormation
  + Templates – No limit
  + Stacks per account – 200 – can increase by contacting AWS
  + 60 parameters and 60 outputs in a template
* ELB
  + No cost
  + 200 subnets per vpc – call aws for more

**Refresh**

* FAQ - [Amazon Simple Queue Service](https://aws.amazon.com/sqs/faqs/) || [Amazon DynamoDB](https://aws.amazon.com/dynamodb/faqs/) || [Amazon ElastiCache](https://aws.amazon.com/elasticache/faqs/) || [Amazon Kinesis](https://aws.amazon.com/kinesis/data-streams/faqs/) || [AWS Lambda](https://aws.amazon.com/lambda/faqs/) || [Amazon API Gateway](https://aws.amazon.com/api-gateway/faqs/) || [AWS Elastic Beanstalk](https://aws.amazon.com/elasticbeanstalk/faqs/) || [AWS Identity and Access Management](https://aws.amazon.com/iam/faqs/) || [AWS Key Management Service](https://aws.amazon.com/kms/faqs/)
* Dynamo DB billing

Practice Test 1: 9

Practice Test 2: 1 6 19 29 34 35 46 50 59

Practice Test 3: 47 50

Practice Test 4:

Practice Test 5:

Practice Test 6: 2 9 10 19 22 23 24 25 27 28 33 36 38 40 41 42 51 53 54 55 56

<http://tutorialsnation.com/aws-certification-dumps/14>

<https://github.com/okeeffed/cheat-sheets/blob/master/AWS-Certified-Solutions-Architect.md>