AWS SERVICES

Basics

1. most services will have to region scoped and some do not have some services.
2. pricing does vary from region to region.
3. regions - each availability zone is linked together and 2+ linked together they will form a region. each az is a discrete data center.
   edge locations: they are additional servers to serve location near main regions. e.g. chennai had multiple edge locations.
4. IAM - identity and access management in aws.

IAM: IDENTITY AND ACCESS MANAGEMENT

1. root account is created by default and should only be used once.
2. users and groups can be assigned JSON documents called policies. - there are numerous policies available including custom policies that we can upload.
3. user can belong mutiple groups and policies and have accces to all the benefits of each policy.
4. version, id, statement.
   statement: {
   sid:
   effect: allow/deny
   principal: who it targets,
   action: what it allows. e.g. s3:getObject
   resource: any resouce e.g. a bucket.
   }
5. this permissons can be very specific and also add request conditions if we wanted to. there is also a visual editor that we can use to edit the json using a checkbox style editor. this is a json generator.

multifactor authentication + password policy

- a password policy.
- MFA - password + token

Options

1. Universal 2nd factor authentication using a 3rd party usb or a hardware key fob device or gov has its seperate cloud functions.
2. Virtual MFA device e.g. an email.
3. Did not establish mfa since we do not have an android phone.

Access key for authentication for code and cli (computer) + SDK + CLI

1. access keys are secret and function similar to a password. access key is the username e.g. accesskeyID:accesskey would be the key value pair.
2. with aws cli we have direct access to all api's including s3 and an alternative to using the aws management console.
   3.SDK - language specific code. node + python, mobile sdk and iot sdk is also available.

we are using the cli -

aws iam list-users ; this returns all the users and their associated info in the aws account. if we dont have access we are sent nothing.

Some AWS services

some aws services will have to perform actions on your behalf so we need to assign some roles to them.

roles are like people policies but they are assigened to aws services rather than people. i.e. we are giving permissions to services instead of people.

service and the role become one entity and they perform actions against aws.

common roles include

1. ec2 instance roles
2. cloud formation or lamda functions.

iam credential report and iam access advisor are 2 reports that we can see who is doing what with what. the principle is the least priviledged is the norm.

never share the access keys or the iam users. we could get large bills.

Billing

billing is very detailed and the it can be broken down into regions and their services cost.

EC2 - General

It consists of several capabilities including:

1. renting virtual machines (ec2)
2. storing data in (ebs)
3. distribute the load across machines (elb)
4. scaling the service using an auto scaling group (asg)

ec2 comes with a lot of config options.

EC2 user data is a script that is run whenever the system starts and only when the system starts. It can include anything and is run only once. e.g. install software.

trade off - are betweeen compute, memory and networking. general purpose instances will have a good balance between compute network and balancing.
e.g. media servers will need high compute requirements and db will need memory optimised or bi will beed in-memory (ram) optimisation

firewalls - security in the aws cloud, control traffic into and out of the instance. The security group will have rules regarding the instance. e.g. access to ports, IP ranges or inbound or outbound network calls. this security policy is the firewall. ----- security groups lives outside the ec2 instance

so if a traffic is blocked then the instance will not know about it. time out is a security group issue and error is a program error.
e.g. if we do not allow http then all http requests will be stopped. it can be accessed by ssh if it is permitted.

connection: ec2 instance connection options - in the ec2 webrowser or ssh are the available methods. we need the key pair for ssh. both methods require port 22 access as both are versions of ssh.

ROLES

IAM roles for ec2. this is the safer alternative to the aws cli access we have in the home desktop.

types of instances : on demand, reserved instances, convertible reserve instances, schedules reserved reserved instances, spot instances, dedicated hosts, dedicated instances.
reserved instances get up to 75% discount.
spot instances can lose the work load at any time so we only use them when we are resilient to failure. e.g. any distributed workloads.

EC2 - Storage is ebs

EBS - Elastic block storage. this is what allows for persisent storage.

types: ssd, mission critical workloads and st1.

They are network availability zones and we get 30 gb of free tier storage.
It is a network drive that is attached to the drive and uses the network throughput.
we can attach more than one storage volume to each instance and a volume can exist without being attached.

they can be deleted on root volume and can be setup so that the volume is deleted when the instance is deleted. They can be transfered across region by taking a snap shot.

A snap shot can be transferred to another region.

AMI - amazon machine image - is a customisation of an ec2 instance that allows us to configured to prepackage certain software etc. They can also be copied across regions.
or we can use it from the aws market place for ami e.g. jitzi server in digital ocean.

EC2 instance store is the option for storing information that is more high requirement however it will be deleted when the instance is relaunched. it is ephemeral so buffer or cache. These are very high performance volumes.

EFS - elastic file system

it is available across all regions and instances across multiple locations will connect to same file system. this does not work for windows. this works for webserving or cms and is parellel. the bills can be very expensive but all the instances will have access in high availability to all their file systems. This is for cdn's.

ELB AND ASG - elastic load balancing and auto scaling groups

users connect to a load balancer and then connect to a instance via the load balancer.

vertical scalability: increase the size of the instances.
horizontal scalability: add more instances/ systems.
high availability - surivive disaster and other service disruptions and goes hand in horizontal scaling and genrally run in multiple availability zones.

load balancers - diverts traffic to the networked instance and balances the load across instances.

elb: checks the load on a instance and we normally jus use aws services for this.

HEALTH CHECKS : a healthy instance can take on more requests and are crucial for load balacing across instances. Poor health means we need to allows for vertical or horizontal scaling.

Types of loadbalancers

1. classic load balancers
2. applications load balancers - allows load balancing to the same ec2 instance (containers) and only works on layer 7 (HTTP) and also supports redirects.
3. route routing is available e.g. /inventory is different from /home or query within the url. so micro services and container based applications can be established with this.
    1. ALB can be an ec2 isntance or ecs tasks or lamda function.
    2. health checks are done at the target health group level.
       we can set rules for the load balancers.

4. network load balancers - tcp or udp based network calls. so its very low latency and can handle millions of requests.

It can be internal or external load balancer.
ELB also offers logs for monitoring and trouble shooting.

load balancer stickiness

allows us to redirect the same client to the same instance e.g. the same session is maintained but this circuvents the health based routing of the load balancers

Connection Draining

The instance will last till the requests to the instances is completed before it is drained.

Auto scaling groups

we can setup a system where we can set max and min instances that will automatically increase the no of instances to fullfill the request requirements but we need to set this out.

1. target tracking scaling - expansion is based on target cpu usage.
2. simple and step scaling - rule based scaling e.g. launch one more istance when we reach 70% health.
3. scheduled expansion in the number of instances e.g. Friday evening.

we can also establish alarms to email or notify when there is a over usage on our resources. We can also add capacity units and it is done automatically once the alarm is triggered.

> we install a package called STRESS to hog cpu resources and trigger the alarm.

Relational Database Service

we access to the following services in the RDS:

1. continuous back ups - 5 mins ago and 30 days snapshots are available.
2. read replicas are available - helps us increase read transactions speeds. The syncs between db's are async and will eventually be consistent. Write is only done to one

database. The reporting application is generally connected to the read replicas.
3. upgrades are also done automatically.

RDS is auto scaling is also available in vertical and storage criteria. This can also be rule based e.g. expand x if free space is less than 10% of allocated space. To leverage the read replicas the application must update the connections to leverage the read replicas individually.

- data transfer within a region is free and is part of the managed database service.
- we can also establish a standby in an other region/az and a read replicas can also be setup as a standby db.

RDS security

we can encrypt the master and the read replicas and the encrpytion has to be defined during the creation phase. The transit encrytion has to be defined around ssl certificates.

- inflight encryption is done at the database level.
    - mysql - inthe db command
    - at the group level security command

IAM help control who can mange the database e.g. create/delete databases and tables. This can work with mysql and postgres can be done with IAM authenticatio through an api which returns a token. so this is a token based authentication. The token is valid fot 15 mins.

Ensure that the DB is configured to only allowed to connect to ssl connections.

Amazon Aurora

It is compatible with mysql or postgres and it is faster for cloud based services.

- failover is faster, and high availabilty.
- higher read and write speeds.
- makes 6 copies across 3 az.
- it has self healing.
- it has 15 read replicas.

It has a reader and writer endpoint and also allows for IAM token based authentication.

Elastic Cache overview

This is for redis or memcached databases. We store the response from the db in cache and if we get a cache then we do not query the db.

-this can make out application stateless, twitter.

Route 53 - Managed DNS

ROUTE 53 is a managed domain name system. The most common records are:

1. A: hostname to IPV4 - we can provide multiple IP's to allow for client side load balancing.
2. AAAA: hostname to IPV6
3. CNAME: hostname to hostname
4. Alias: hostname to aws resource

These resolve to a particular ip address or hostname.

- CNAME: points a hostname to another hostname however it only works for non root domain. e.g. something.mydomain.com
- ALIAS: points a hostname to an AWS resource. They work for root and non-root domain. e.g. mydomain.com. ALIAS is an aws domain service only.

Allows load balancing across instances via IP allocation and across regions.
TTL - time to live i.e. it is the time that the request ip is cached in the browser and then used to relocate the request. e.g. 300 seconds to max of 24 hrs.

Types of routing policies

1. Simple routing to another IP
2. Weigted routed policy - a % of all requests will meet a particular endpoint. A new version of the app can be set in a particular IP and then we can direct a certain % of the all applicants to that version.
3. Latency routing Policy - routing is based on which ever is IP is faster.
4. Health checks: can also be established and it will not route to IP's that are not

healthy.

5. Failover routing: this is based on health check and if the health check fails then ROUTE 53 will automatically redirect to the secondary ip incase of failover.
6. Geolocation routing: this routing is based on the user location.
7. Geoproximity: route based on resources and the users based on biases. The bias is based on the user location relative to the distance between the two servers. Think of it as a line between the two severs that splits all the services requests between them.
8. Multivalue: this is used when we want to route traffic to multiple resources and want to asssociate a ROUTE 53 health check records. It is ambivalent about routing. This allows for client side routing.

VPC - VIRTUAL PRIVATE CLOUD

VPC and SUBNETS

VPC: is a private network to deploy your resources. This is region bound.

SUBNET: this allows us to partition your network inside your VPC(availability zone level)

PUBLIC SUBNET: This is a network that is accessible from the internet. This has an access to an interner gateway. for private instance that need access to the internet we can have a NAT gateway that allows them to access the internet while being private.

To define access to the internet and the subnets we use **route tables**. A VPC has set of ip ranges and we can have a private and public subnets in each availabililty zone.

Network ACL

- It is a firewall which controls traffic from and to a subnet.
    - can have allow and deny rules.
    - are attached at a subnet level.
    - rules only include IP addresses.
- security groups: is a firewall that controls traffic to and from an ENI/EC2 instance.
    - this can only have allow rules.
    - rules include ip addresses and other security groups.

VPC Endpoints: allow us to connect to aws services using a private network instead of the public www network. e.g. s3 and dynamodb

Site to Site VPN & Direct Connect

Site to site VPN: this connect an on premises vpn to aws and it goes over the public internet.
Direct connect: establish a physical connection between on-premise and aws. this goes over a private network.

TYPICAL Three-tier solutions Architecture

We have an elastic load balancer and we use route 53 to find the address of the elastic load balancer. Then we can access the load balancer and connect to an instance in private subnet inside a auto-scaling group. the provate subnet will have access to the data subnet i.e. rds or s3 storage which is only accessible by the internet.

This the path that req will take - **client > route 53 > elb > private subnet > data subnet**

the lamp stack in ec2 is available.

S3: INFINITLY SCALING STORAGE

All operations are consistent after that it is consistent. ir. earlier writes will be reflected in subsequent reads there will be no time delay.

what are buckets and objects?

S3 allows person to store files(objects) in buckets and buckets have globally unique name.

- all object has a key which is the full path to the file. the key consists of two component the prefix and the object file.txt. the object values are the content of the body.
    - metadata cane be added on each object e.g. time of updation etc.
    - tags can alos be added for security or lifecycle reasons.
    - version id are also available.

the public url will have its access denied if we just copy the object url. we need to set permissions to allow it to work properly. we can access it only if we have a presigned url. The open button generates a presigned url and that is why we are able to access it.

Creating Folders

Folders can be created using the create folder button and they are also objects that can link to other objects. It is just a key value pair with the illusion of creating folder for our organisation.

Versioning

versioning services are available however it has to be enabled at the bucket level. so if we reupload a file with a same key then it will create a version of it and then store it. Both version will be available. Versioning is activated in the properties section.

- suspending version will keep the previvous version.
- if we start versioning after the old files will not have a version therfore it will be null.

The version will start based on the key value. i.e. if the file has the same name then it will be replaced. If we delete a file it will add a delete marker to it. (paranoid delete) and it will be available in the version history. if we delete the 'delete marker' then it will be permanently deleted. This is what allow for version control.

Encryption

Protection of the files - 4 methods.

1. encryption can be handled by aws - sse-s3 - keys are with aws and server side encryption.
2. sse-kms - is a key managed service and gives us control over who has access to what and we also need to set the header for kms.
3. sse-c : we will provide the key that is used to encrypt the data. the key will be discarded every single time. https is mandatory.
4. client side encryption - we encrypt using an sdk and then send.

sse-s3 is done by setting the header to:

```
Must set header: “x-amz-server-side-encryption": "AES256"
```

The server encrypts it and stores it.

Security and Bucket Policies

User based

- IAM policies - which API calls should be allowed for a specific user from the IAM console
- Resource Based policies
    - Bucket Policies - bucket wide rules from the S3 console - allows cross account access to the bucket.
    - Object Access Control List (ACL) – finer grain
    - Bucket Access Control List (ACL) – less common

An iam principle can access it the user policy allows it or the resouce policy allows it and there is no explicit deny in the options.

Policies: use the json objects. Grant public access or grant access to another another account.

Buckets can be stopped from being public in their entirety.

User security include a

1. mfa for delete
2. pre-signed url that are valid only for a limited period of time.

SECURITY POLICY: we set one up so that all files uploaded has to have a encryption allowed and the test succeeded and it has to be of that type of encryption stated.

We can also block access to all the bucket with the account level bucket access console.

Websites

we can store a website and have it be accessible to the public. the URL will be

> *bucket-name*.s3-website.*AWS-region*.amazonaws.com

to allow public access we have to remove the block public access in the permission tab and also allow public access in the bucket policy statement.

CORS EXPLAINED

We cannot make requests to other website if the origin is different. It is the cross origin website that provided the access and the access is provided to the browser as well. so we need to enable cors headers in our s3 bucket. The cors headers are defined in the cross origin bucket and not in the same origin bucket. We can allows for a specific origin or a cross origin.

we provide the access in the cross origin bucket.

```
[
  {
    "AllowedHeaders": ["Authorization"],
    "AllowedMethods": ["GET"],
    "AllowedOrigins": [
      "<url of first bucket with http://...without slash at the end>"
    ],
    "ExposeHeaders": [],
    "MaxAgeSeconds": 3000
  }
]
```

S3 ADVANCED

MFA delete

This requires versioning in the bucket and it has to use root credentials to set it up.

Access logs

for audit purposes - this tracks all the requests made to the bucket and also allows storage in another bucket.

S3 REPLICATION CRR and SRR

ASYNC replication across buckets in different location to maintain a common access pool. There is no chaining of replication between buckets. Replication rules do not work on a delete.

S3 PRESIGNED URL

Users given a presigned url will be given an option to make get or put requests (ttl : 1 hour). We can allow a user to download a specific file or upload a file to a predefined location in the database.

method: cli:

```
<!-- # do not forget to region parameter! (make sure it's the proper region you're choosing) -->
aws s3 presign s3://mybucket/myobject --region my-region

<!-- # add a custom expiration time -->
aws s3 presign s3://mybucket/myobject
<!-- --expires-in 300 --region my-region -->
```

S3 STORAGE CLASSES

1. general purpose
2. standard-infrequent access
3. one zone-infrequent access
4. intellignet tiering
5. glacier
6. glacier deep archive

S3 LIFECYCLE RULES

Moving between storage classes can be done on automatically using a life cycle configuration options this can also include file deletion on command using expirtion action.

- rules can be created for a certain prefix or an object tag.
- delete parts of incomplete multipart uploads.

we can create a lifecycle policy at the bucket level.

Notifications are also available.

ATHENA

It is a serverless service and perform analytics directly against s3 files. We can use sql to query the files about the logs/bi etc. e.g. how many blocked get requests or file not found requests.

CLI, SDK and IAM roles and policies

CLI (S3)

Performing AWS tasks against AWS can be done in:

1. AWS CLI on local and an ec2 instance
2. AWS SDK on local or ec2 instance
3. AWS instance metadata service for ec2.

```
$aws s3 ls <!-- this list all the buckets in the acccount -->
$aws s3 ls s3://asxstudents  <-- this lists all the items in the bucket -->
$aws s3 cp s3://asxstudents/calender.jpeg calender.jpeg
<!-- this downloads the file locally in the current working directory -->
$aws s3 mb s3://asxstudents3
$aws s3 rb s3://asxstudents3
```

commands include cp, ls, mb(make bucket), mv, presign(generates a presigned url), rb(removes buckets), rm(object removal), sync(sync between bucket and local storage or bucket and bucket) and website(setup the website config for the url)

we can also look at the documentation for cli commands for all the aws services.

IAM roles

policies: aws has a lot of policies that are available to us and also we can create our own. An inline policy is one that only exists for that instances policy.

Permission polices: read and get access to all the resources with the amazons3readonlyaccess policy.

there is a visual editor for the amazon editor that we can use to denegerate the policy documents.

--dryrun will simulate api calls but these commands will not be executed. this is available for most but not all commands.

STS COMMAND CLI ERROR DECODER

this sts decodes the error message and that converts it to a more human readable format.

```
~aws sts decode-authorisation-message .... this will decode the error message that we can then understand
```

sts get session token allows for multi factor authentication.

SDK OVERVIEW

It is dependent on the language being used and we have to use the sdk when we make api calls to the aws services e.g. a rds.

if we dont configure a default location then the US will be chosed as the defauly location and call will be made there.

AWS limits

1. api rate limits - ec2 has a 100 calls per second limit while s3 has a 5500 calls per second.
2. service quotas - how much we can run the other services.

EXPONENTIAL BACKOFF: we use this when we have a throttling exception in the api calls.

SIGNED REQUESTS

All requests to aws should be signed and the sdk and cli automatically sign all the requests for us. e.g. to s3.

CLOUDFRONT

Cloudfront is CDN and it is cached in a edge location all over the world and has ddos protection and integration with shield etc and we can expose the external https which talks to an internal backend. It distributes the reads across the world. It is leveraging the s3 bucket as the base storage location.

- allows for enhanced security with cloudfront.
- used an ingress to upload files to s3.
- we can have georestrictions based on a geo-ip database.
- it can seperate between our dynamic requests and static requests.
- invalidation allows us to remove the file from all the edge location incase of an update in the file.
- also allows for signed URL and signed cookie and also to access multiple files with multiple requests. The signed url is similar but not the same as the signed URL.

PRICING: cost of serving is different across regions have price classes based on edge location that we want our content to be loaded in.

ECS, ECR and FARGATE with DOCKER

ECS essentials: Elastic Container Service
ECR: Elastic Container Registry (amazon's docker registry)

we can run docker container in the ec2 instance and internally scale the instance and run a cluster configuration inside the instance.

FARGATE: its all serverless. i.e. we can create the task definition and amazon will run the container for us. So if we wanted to scale then we jus need to increase the task number. There can be an auto scaling groups so the capacity provider will launch an ec2 instance and then execute the task.