Aws questions—

Types of Cloud Services-

1. IaaS- Infrastructure as a service- The client have full access to control the infrastructure. It is same as the traditional data center without physically maintaining/ managing the servers. Clients can access the servers and db but virtually.

Thing to be maintained by client- Application, runtime, middleware, data, OS

Things to be maintained by cloud provider- Networking, servers, storage, hard drives, virtualization.

Advantages- very flexible

* Pay as you use
* Highly scalable
* Client has complete control over the infrastructure.

1. PaaS- Platform as a Service- The client is provided with the software and hardware and it is managed by the provider while the client is only concerned about the application and data.

It is idle for developers. All the servers, networking, OS, runtime are maintained by the provider/third party provider.

Advantages-

* Simple and cost effective development ans deployment principle
* Scalable
* Highly available
* Reduced amount of coding needed

1. SaaS- Software as a service- The software hardware as well as the application is maintained by 3rd party itself. No need to download or install anything on client side. Everything is readily available on internet.

Advantages-

* Managed from central location
* Hosted on remote server
* Accessible over internet
* User is not responsible for hardware and software.

3 Basic types of cloud services-

1. Compute- Ec2, lambda, autoscaling
2. Storage- S3, Glacier, EBS,EFS
3. Networking- VPC, R53, CloudFront

Difference between ami and ec2-

AMI is a templete of OS/ it is like a cd through which we can install OS or softwares to our computer

EC2 is a phyical entity/system that can be aceessed virtually

Diff between scalibility and elasticity

Scalibility- Ability of a system to handle increased load by increasing the size of the CPU/RAM-- no of machine is not increased/only specification is varried

Elasticity-Work overload is handled by increasing the no. of resources/machines

Difference b/w CloudWatch n CloudTrail

Diff between CLB and ALB

CLB- distributes loads equally in round robin fashion

ALB- distributes loads based on the required URL/path.

Difference between cloud watch and cloud trail

Cloud watch-- real time monitoring of aws resources and applications like ec2, rds, s3 etc

Cloud trail-- it helps in governing the aws account.Monitors and logs the activities performed in aws account

EIP cost is incurred only when it is attcahed to stopped instance or it is allocated but not associated or more than 2 EIP is attached to an instance

Different types of ec2-instances purchasing

1.On demand- pay as you demand

2.Spot- request for unused instances and bid a price

3.reservered- make a commitment of constant ec2 configurations for long period and get discounted price unlike on demand

4. dedicated- instanced that run on single tenant hardware--blocked for use by single customer

5.Capacity reservation- reserve capacity of ec in a particular region and AZ for any duration

6. Saving plans- commitment on usage per hr. in USD per hr. for long duration

move data over long distances using the internet, for instance across countries or continents to your Amazon S3 bucket--Amazon transfer acceleration

Amazon RDS- db management service for structured data

Dynamo DB- db mangemnet service for unstructured data

Lifecycle hook in Auto scaling-- they are used to put additional wait time when scaling (scale in /scale out) event is going on

Due to some issue the auto scaling group has failed to launch a single instance for more than 24 hours --then it will suspend scaling process

Types of ec2-

1. General purpose- balance of compute memory and networking resoources—m, t

2. Compute obtimized-- used for delivering high performance--c

3. Memory optimised-- used for deliveringg fast processing to the the works that process large data in memory--r

4. Accelerated computing-- uses processors in much efficient way than a normal instance does-- used for grapic processing or pattern matching—g/p

5. Storage optimized-- used for workloads that require high read and write access in large data sets on local storage.-- i

Diff between vpn nd vpc

---------------------------------------------------------------------------------------------------------------

All About EC2- Elastic Cloud Compute

1.difference between AMI and instance

Best security practices for ec2-

- use IAM to control access to AWS resources

- whitelist only the ports which are used to connect to that instance

- the security group associated should be restrictive

- disable passwrd based authentication for remote login

2. Regions and AZ-- Regions are different physical locations(countries/continents) where large AWS datastores are present

AZ is the subdivision of a region --they are isolated locations within each region

3. Types of ec2 instances-

- General purpose

- Memory optimised

- Storage optimised

- compute optimised

- accelerated computing

4. Types of ec2 on the basis of price-

- On demand

- Spot

- dedicated

- reserved

- saving plans

-capacity reservation

5. Security group-

- it is attched to network interface(eth0)

- it filters the traffic on the basis of port and protocol

- it is stateful that is-- if a request is sent outside from the ec2, then response will come irrespective to the inbound restrictions

- we can attach multiple security grps to a instance

- if we have more than one rule for a port, the most permissive one is considered

6. Elastic IPs are region specific

All AWS accounts are limited to 5 Elastic IP addresses per Region

7. Elastic Block Storage

- block level storage volume that behaves like a raw unformatted block device which can be mounted

- independent of the life of instance

- if you want same volume in another region- make a snapshot of it....shift the snapshot to another region and then create a volume out of it

- types of EBS volumes-

1. general purpose SSD vol- balance between price n performance. can be used for various purposes .can be used for development and testing env

2. Provisioned IOPS SSD vol- it can provide upto 64000 IOPS and 1000MiB/s throughput

3. Throughput optimized HDD vol- low cost magnetic storage whose performance is measured in terms of throughput.can be used for large sequential work loads such as log processing

4. Cold HDD vol- low cost magnetic storage device whose performance is measured in terms of throughput. Used where frequent access of data is not required.

8. Instance State

1. Started

2. Stopped

3. Hibernate- when we hibernate an instance the contents present in RAM of ec2 is loaded to EBS and then the instance is stopped. can be used with specific instance types only

4. Reboot- its like OS reboot.

5. Retire - an instace is retired when AWS finds out an irreparable failure of underlying hardware. AWS itself stops or terminates such instance

6. Terminate- normal shutdown process where all the volumes are deattached and instance cant be recovered or started again

------------------------------------------------------------------------------------------------------------

AWS AUTO SCALING

We can scale up or down the no. of instance required as per usage. 3 paramaters- max/min/required no of instances

Benefits-1. Better fault tolerance 2.Better Cost managemnet 3.Better availibility

Lauch Config/ Launch Templete

- includes AMI, instance type key paie information,security grp,user data, EBS mappings

Launch templete is similar to launch config but it has some additional features over launch config like version maintianing. With launch temnplete all the latest features of Autoscaling is compatible but not with launch config

How does scaling policy works- Autoscaling group continuously monitors a CW metrics and scaling policy defines what action is to be done when

Types of Scaling policy--

1. target scaling--> you define a target value of the Matrix and the ASG adjusts no. of instances based on that. create an ASG then create a target tracking scaling policy.

2. Step scaling-The alarm has to be defined, scaling policy has to be defined and what action has to be taken when is also defined.Create an ASG-->create a scaling policy-->create a CW alarm

3. Simple scaling- smae as step scaling.

Difference- When scaleup/down is going on in simple scaling,the policy must wait till the activity is over to response to another alarm but the same is not true for step scaling.Create an ASG-->create a scaling policy-->create a CW alarm

Lifecycle hook-- enables us to perform custom actions by putting instance in wait state during a scaling event

LOAD BALANCERS

Types-

1. Classic load balancer/Elastic Load balancer- Provides routing at layer 4(tcp/udp) and layer 7(http/https) also

- doesnt provide path based routing. Routing is done on port no.

- cant forward traffic on more than 1 port per instance.

- it doesnt support forwarding to an elastic IP. Specific instance has to be defined.

- doesn’t support target groups

2. Application LB - Provides routing on layer 7

- provides path based routing

- it can provide user authentication

- it can be configured to give a fixed response

- it doesnt support forwarding to an elastic IP. Specific instance has to be defined.

- can send an instance request from many ports

3. Network LB- works on layer 4

- it can be assigned with EIP unlike ALB n ELB

- high performance

- it preserves source IP unlike ELB n ALB

---------------------------------------------------------------------------------------------

All About VPC(Virtual Private Cloud)

VPC is a private virtual network which you define and you can launch ur resources here.

While creating VPC you need to define a range of IPs that can be allocated to resources inside the VPC.

Important components of a VPC-

Subnet- Small range of IP addresses in your VPC. there can be multiple subnets in a VPC.

Subnets can be Public or private.

Resources used to connect to internet are launched in public subnet.like- proxy server.

Resources not used to cnnect to internet are launched in private subnet.like- db

Route tables- It is a set of rules which defines where the traffic has to go.

Accessing Internet- accessed with help of internet gateway which is associated to Public routr table which in turn is connected to public subnet.

Accessing a corporate or home network-

Site to site VPN connection is established to do so. Virtual private gateway is attached to VPC

Accessing services through aws private link-

We can access various AWS services without connecting over internet. It connects to services via VPC endpoints

VPC peering connection allows you to communicate to instances in another VPC privately as if they are within same network.

Monitoring VPC--

1.Flow logs- it captures the info about IP traffic going in and out of the VPC. We can create flow logs for VPC, subnets, ENI

2.Monitoring NAT gateways- we can monitor NAT gateways for incoming and outgoing traffics

VPC FlowLog- It can publish flow log data directly to the CW

Need to have an IAM role and trust relationship to "vpc-flow-logs.amazonaws.com"

Elastic Network Interface-

-every instance comes up with an NI attached to it which cant be reattached to another instance

-an instance can have more than 1 ENI attached to it

-ENi has 1 primary IP and we can assign multiple secondary IP to it.

Route Table- it contains a set of rules which determines where your traffic has to be directed.

Main RT- default RT created at the time of VPC creation. If a subnet is not attached to any RT, then this RT is responsible for directing traffic.

Custom RT- custom defined RT.

- A subnet can be attached to only 1RT but a RT can have multiple subnets attached to it

-each routes have a Destination and target

-Every route table contains a local route for communication within the VPC.This route is added by default to all route tables.

-if RT has more than 1 subnet association, then the longest prefix matching is used to route the traffic

-longest prefix match mechanism is used to route the traffic

Prefix List- Its a set of ip addresses or CIDR ranges, which are frequently used by you and all of them are routed to same destination

VPC Endpoints- It enables private connection between VPC and AWS services like secrets manager, S3 bucket, Dynamo DB-

Types-

1.Interface Endpoint- It creates a private DNS for the service which we want to use.

2.Gateway endpoint- In this we create route to the AWS enpoints via entry on RT. GAteway endpoint gives us a Prefix list which has to be associated with RT

AWS Network Firewall- A firewall where we can eliminate unwanted traffic on VPC level.It is stateful.

Difference between VPN and VPC-

1.VPC- its a whole network which is private and no one except authenticated users can enter.

2.VPN- Virtual provate network- It is a virtual peer to peer connection using combination of dedicated connections and encryption protocols. Its like a tunnel through which the traffic passes.

3.VPS- Virtual Private Sharing- Its a technology by which we can host our applications on a virtual servers which is a shared resource. All the partitions are separate and independent of each other in a virtual server.

------------------------------------------------------------------------------------------------------------

Amazon Simple Storage Service(s3)

S3 is a Simple storage service which is very simple to use and highly robust. We can store our data on S3.

Each s3 bucket can store a data upto 5TB

We can access the data files stored in S3 by API calls

Buckets- Its a container which contains objects stored in it. It can store upto 5TB of data. It is specific to region and it doesnt leave that region

Objects- It is the entity stored in S3 bucket. Objects consists of object data and metadata. Object data is the data that id stored in the file and metadata is the name value pair that describe the object.

Key- It is the unique identifier for the object in the bucket.

Storage classes in S3-

1.s3 Standard- used for general purpose storage of frequently accessed data

2.S3 Standarad\_AI- used for storing data for long term which are less frequently accessed

3. S3 Glacier- used for long term archieve

## The combination of bucket, key and version Id uniquely define an object

S3 are highly avialable. Whenever we put an object in s3, the data/object is automatically stored in multiple server to make it highly available.

Bucket policies- The policies that is applied on the bucket. Add grants to the entire bucket/all components of bucket.

Access control list- Used to grant access to individual objects in a bucket.

S3 provides both REST and SOAP interface

Properties of S3 bucket-

1. Bucket versioning- It maintains the version of objects, so that if we replace an object by mistake then we can recover the previous object. It is by default disabled in S3 bucket

2. Default encryption- server side encryption- It encrypts the data before saving it and decrypts is when we download it

3. Server access logging- Get detailed record of who is making calls to the bucket.

3. Transfer acceleration- Helps in easy fast and secure transfer of files over long distance. Uses edge location to perform fast transfers. We can use this feature when we have to transfer huge amt of data frequently.

4. Object lock- Locks the object so that it doesn’t get deleted or overwritten by anyone for specified period of time

5. Static website hosting

-We can delete a bucket having objects from console, otherwise we can delete an empty bucket

Setting default bucket encryption- It uses AWS managed key to encrypt the objects. We can also use Customer Managed key for encryption with same or cross account.

- to encrypt the existing objects, use S3 Batch Operations which can encrypt many objects at a time

Object= Data+ metadata

Types of metadata-

1.System defined metadata-Maintained by system like when the object was crested, last modified date and time.

2.User defined metadata-Maintained by user,like key pair value. When using rest API- header x-amz-meta- is the header.

-max size of file we can upload to any bucket is 160 GB. To upload larger files use AWS CLI or REST API

MultiPart Upload in S3- It allows to upload a single object in parts. Used for uploading larger files. The files are uploaded as different objects. Once all the parts are uploaded, S3 forms an object out of it which can be accessed like a single object

Data Encryption in S3-

1.Server side encryption- Encryption performed after the data reaches its destination/s3.Types-

a)Encryption with Amazon s3 managed key(SSE-S3)- the data is encrypted with a key managed by s3 which is encrypted by a master key and is rotated regularly.

b)Encryption with CMK- Encryption key is generated by customer and data is encrypted with that

c)Encryption with customer provided key- The key is provided by the customer using which encryption happens

2. Client Side encryption- Encryption before sending the data to s3.

Managing ACL- Under the section permission we have ACL for both object and bucket

CORS- Cross origin resource sharing- Its a way through which a web application loaded with a domain can interact with resources of another domain. S3 supports CORS

Managing Storage in S3-

S3 versioning- each object has a version no associated with it. A bucket can be versioning-enabled, versioning-disabled and versioning-suspended.

By default all buckets have versioning-disabled

Once you move from versioning-disabled to versioning-enabled, you can’t switch back to versioning-disabled. To overcome this limitation, we use versioning-suspended.

Storage classes---

Objects stored in S3 glacier/s3 glacier deep archieve/S3 intelligent-tiering archive access/ deep archieve access tiers, are not accessible in real time.

For archieve acess/deep archieve access tier ---> we have to 1st put restore access and then wait untill the object is moved to frequent access tier.After that if obj is not accessed for 30 days it moves to Infrequent access tier, after 90 days to archieve access tier and after 180 days to deep archieve access tier.

For s3 glacier/deep archieve glacier -->raise a restore request and wait untill a temporary copy of object is available.

1.S3 Standard- It is standard storage for objects frequently accessed. It is highly robust.

2. REduced redundancy- Cna be used to store and access datas which can be recreated since it is not robust.

3. S3 Intelligent tiering- The objects are moved to different tiers according to the uasage/access of the object.

4.S3 Standard IA- Used for long lived, infrequent acessed data. The data is stored in many AZ

5.S3 one Zone IA- Same as standard IA but the data is stored only in one AZ

6.S3 Glacier- used to store data which is rarely accessed. Data is available for acess within 1-5 mins of restore request. The objects has to be stored for minimum of 90 days.

7.S3 Glacier Deep Archieve- used to store very rarely accessed data. Data is available after 12 hrs of restore request. The objets need to be stored for minimum of 180 days.

Managing Lifecycle of objects- Required to store object in a cost efficient manner.There are 2 types of action-

1.Transition- We can define after what time our object has to make a transition from one storage class to another.

2.Expiration- We can define when we have to delete the obj and S3 will do it on your behalf

S3 signed URL- The application can share objects with other by creating a pre –signed URL. By default all the objects are private. The object owner can only access the objects. The owner can share the object with other users by pre-assigned uRL using their own security credentials to grant time limited permission to download the objects.

-----------------------------------------------------------------------------------------------------------------------------------------

AWS CLoudFront-

It is a service which speeds up distribution of the static or dynamic web contents(.html,.css) to the users. It uses the edge location to speed up the loading process

Benefits of using CF-

-accelerated static website content delivery

-serve video on demand or live streaming

-encrypt specific field throughout the system processing

How does CF work?--> when the client sends any request for image files or HTML pages, the request goes to nearest POP or edge location(nearest to client). If the requested page is present in the cache then that is returned else the request goes to the actual server that gets loaded to edge location and then sent to the client.

Regional Edge cache- it sits in between origin server and POP. It is used to store less popular content. If a content becomes less popular then it is moved to regional edge location since it has larger cache as compared to edge location. The content flow is as follows--

Origin server-->Regional Edge location--->POP/Edge location

We create distributions in CF which defines about origin, general configuration for hosting the files.

Geo-targeting- Slight different content is send to the users in different regions.

How it is achieved?—CF recognizes the request is being sent by which country and forwards company code to the server so that the content can be customed for user without changing the url

Settings in Distributions-

1. ORIGIN-

a.Origin Domain name- Domain name where your files are present. Basically the name of s3 bucket URL

b.Origin path- If the files are present inside any subfolder, give the value over here

c.Origin ID- If we configure any other cache settings apart from default, then we need to populate this field

d.Origin connect attempts

e.Origin connection timeout

f.Origin custom headers- if CF has to add custom headers before sending request to the origin

g.Restrict Bucket access- Select yes if you want users to access the contents by using CF URL and not S3 URL/path

h.Origin Access Identity- Option pops up when u choose yes in Restrict Bucket Access

i. Grant resd permission to bucket- for automatically updating the bucket policy or u can choose to do it manually

j.Origin protocol

k.Origin connection timeout

- The no. of caches should be equal to or greater than the no. of distribution origin.

-we can configure the above settings on the basis of path also(different origin)

2. DISTRIBUTION

a.Price Class- Choose the regions in which you can have edge locations according to the price

b.AWS WAF Web Acl

c.Alternate Domain- CF provides a default domain with cert. If you want anothger domain/cname then use this feature.

d.SSL certificate- You need to add custome cert for the cname

e.Custom SSL client support- If you have specified CNAME, then choose how you want CF to serve https requests

f.Security policy-Specify the security policy that you want CF to use for HTTPS connections

g. Supported HTTP version

h.Default root object which is loaded when request is made

--------------------------------------------------------------------------------------------------------------------------------

Identity and Access Management(IAM)

It helps to securely control access to your AWS resources. Benefits of using IAM-

1.you can give access to AWS account without even sharing your password

2.You can give granular permission,like an user can/cant launch ec2,etc.

3.Secure access to AWS resources for applications that run on ec2

4.Multi-factor authentication

Principal- It is a person/application that makes request for an action to the AWS resources. It uses an entity(user/role) to send request.

Resource- The AWS resource on which the actions are to be performed

Identity based policies- The policies which are attached to any role,user or group.Control what actions can be performed under what conditions. Which resources can be used by the user.Types-

1.Managed policies- The standalone policies which can be attached to multiple roles/users at the same time.

2.Inline policies- The policies which is created for that particular role/user and cant be used by any other user/role.

Resource based policy-The policies that are attached to a resource such as S3 or IAM role trust policy.Only supports inline policies. Who can use the resource like S3,SQS,etc.It is attached at the resource side…like in s3 bucket.

ABAC(Attribute BAsed access control) and RBAC(Role based access control)

ABAC- The access is given based on attribute or tag. Suppose a new EC2 instance comes and access to it has to be given. Give a tag which is present in ABAC to that instance and you will be able to access that instance.

RBAC- The access is given based on the role attached to the user/resource. Anew s3 bucket has to be accessed then the admin has to add this bucket in the role.

IAM User- it permits a person/service to interact with AWS services. It has credentials associated with it.

IAM Role- Similar as IAM user except it doesnt have credentials associated with it. A role can be assumed by anyone who needs it.

IAM ROLE-

It provides temporary access key and role session which usually lasts for an hour.Anyone can assume roles

Use cases-

1. Different prod and dev env. Access to Dev env user to update bucket in prod env-


        Use a role to delegate permissions to a user in a different account
      

To read—SAML

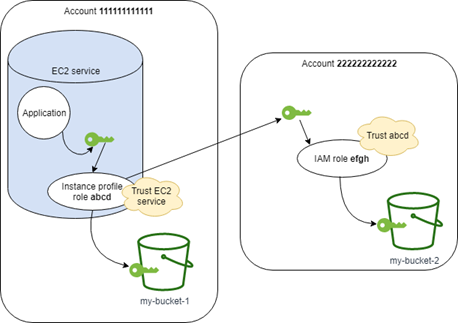
Using the credentials of one role to assume another role is called role chaining.

We cant switch role if we log in as AWS account root user.

How roles work for EC2-


        Application on an EC2 instance accessing an AWS resource
      

Example of using IAM to allow an EC2 to fetch value from a bucket in another account-



**Account 111111111111 *abcd* Role Permissions Policy—**

{

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

"Statement": [

{

"Sid": "AllowAccountLevelS3Actions",

"Effect": "Allow",

"Action": [

"s3:ListAllMyBuckets",

"s3:HeadBucket"

],

"Resource": "\*"

},

{

"Sid": "AllowListAndReadS3ActionOnMyBucket",

"Effect": "Allow",

"Action": [

"s3:Get\*",

"s3:List\*"

],

"Resource": [

"arn:aws:s3:::my-bucket-1/\*",

"arn:aws:s3:::my-bucket-1"

]

},

{

"Sid": "AllowIPToAssumeCrossAccountRole",

"Effect": "Allow",

"Action": "sts:AssumeRole",

"Resource": "arn:aws:iam::222222222222:role/efgh"

}

]

}

The abcd role must trust the Amazon EC2 service to assume the role. To do this, the abcd role must have the following trust policy:

**Account 111111111111 *abcd* Role Trust Policy**

{

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

"Statement": [

{

"Sid": "abcdTrustPolicy",

"Effect": "Allow",

"Action": "sts:AssumeRole",

"Principal": {"Service": "ec2.amazonaws.com"}

}

]

}

**Account 222222222222 *efgh* Role Permissions Policy**

{

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

"Statement": [

{

"Sid": "AllowAccountLevelS3Actions",

"Effect": "Allow",

"Action": [

"s3:ListAllMyBuckets",

"s3:HeadBucket"

],

"Resource": "\*"

},

{

"Sid": "AllowListAndReadS3ActionOnMyBucket",

"Effect": "Allow",

"Action": [

"s3:Get\*",

"s3:List\*"

],

"Resource": [

"arn:aws:s3:::my-bucket-2/\*",

"arn:aws:s3:::my-bucket-2"

]

}

]

}

**Account 222222222222 *efgh* Role Trust Policy**

{

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

"Statement": [

{

"Sid": "efghTrustPolicy",

"Effect": "Allow",

"Action": "sts:AssumeRole",

"Principal": {"AWS": "arn:aws:iam::111111111111:role/abcd"}

}

]

}

AWS IAM Policies-

Types of policies-

1. Identity based policy- Policies that are attached to an user/role. Cant be attached to root user.

2. Resource based policy- Policies attached to an AWS resource like S3,SNS etc. Root user can be mentioned as principal.

3. Permission boundaries- It set a boundary for no. of permissions that an identity based policy can have. It doesn’t apply for resource based policy. Not applicable for root user.

4. Organization SCP- It sets a boundary for no. of permissions that can be associated with an account user. Applies to both identity based and role based policy. Root user is affected.

5. ACL- It is similar to resource based policy with following difference- It is cross account permission policy that grant permission to the specified principal. It cant grant permission to entities within same account. Root user is affected of another account.

6. Session policies- The policies that limit permission for a created session.

Evaluating effective permission with permission boundary-

1. Identity provider with permission boundary-


                Evaluation of identity-based policies and permissions boundaries
            

1. Resource based policy- Permission boundary doesn’t limit permission attached to resources . So no explicit deny for resource based policy


                            Evaluation of a resource-based policy, permissions boundary, and
                                identity-based policy
                        

1. Organization SCP- Permission boundary applies to organization SCP since it controls over the roles/users of an account


                Evaluation of an SCP, permissions boundary, and identity-based
                    policy
            

1. Session policies-


                Evaluation of a session policy, permissions boundary, and identity-based
                    policy
            

-Permission boundary is attached to an user which defines the maximum range of permissions that an user can have. It doesn’t assign permission to user.

Controlling access using IAM tags-

1. Request tag-> aws:RequestTag/key-name🡪 requests for a tag
2. Resource Tag-> aws:ResourceTag/key-name🡪tag attached to a resource
3. Principal Tag->aws:PrincipalTag/key-name🡪tag of the principal
4. Use the **aws:TagKeys** condition key to control whether specific tag keys can be used on a resource, in a request, or by a principal.

Examples-

Specific access during a date range-

{

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

"Statement": [

{

"Effect": "Allow",

"Action": "*service-prefix*:*action-name*",

"Resource": "\*",

"Condition": {

"DateGreaterThan": {"aws:CurrentTime": "2020-04-01T00:00:00Z"},

"DateLessThan": {"aws:CurrentTime": "2020-06-30T23:59:59Z"}

}

}

]

}

Enable/Disable aws region-

{

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

"Statement": [

{

"Sid": "EnableDisableHongKong",

"Effect": "Allow",

"Action": [

"account:EnableRegion",

"account:DisableRegion"

],

"Resource": "\*",

"Condition": {

"StringEquals": {"account:TargetRegion": "*ap-east-1*"}

}

},

{

"Sid": "ViewConsole",

"Effect": "Allow",

"Action": [

"aws-portal:ViewAccount",

"account:ListRegions"

],

"Resource": "\*"

}

]

}

IAM customer managed policy supports versioning. By default it has versioning enabled and upto 5 versions can be stored.

Versioning is not supported in inline policies.

Security best practices in IAM-

* Lock root account access
* Create individual IAM users
* Give least required privileges to the users
* Give customer managed policies rather than inline
* Use access levels to review permissions
* Enable MFA
* Use roles to delegate permissions
* Use policy boundaries

SAML integration-

We can setup identity provider when we want to establish trust relationship with an Idp which follows SAML(Security Assertion Markup Language) 2.0 standards.

To do this-

* We need to register identity provider in the IAM console
* Add provider->choose SAML
* For Metadata go to Idp and generate it and download. Attach this metadata to identity provider in IAM.
* Assign IAM role to your identity provider to let the users access AWS resources.

In this step we have our AWS account trusting the Ipd with the help of metadatas which contains issuer’s name, expiration info and keys to authenticate SAML users.

Now the next step is that our Idp should know about AWS as a service provider. To do this upload the AWS metadata(https://signin.aws.amazon.com/static/saml-metadata.xml) on the IPD provider.

Amazon Elastic Container Services(ECS)

Its is a highly scalable, fast container management service that makes it easy to run,stop and manage containers in a cluster.

Container is a task definition that we use to run an individual task.

We can run these tasks on a serverless infrastructure called AWS Farget or a cluster of Ec2 instances managed by user.

Basic Concepts-

1. Containers and images – contained is an unit/entity which contains all the things that is required to run your application like code, required software etc.

Containers are created from read only template called images.

Images are build from DockerFile which is a plaintext file and includes all the info about what all is present in a container.

Images are stored in registry from where it is downloaded and used.

1. Task Definitions- It’s a text file which that describes one or more containers (upto 10) that form the application. It specifies various parameters (like port, data volumes) for the application.
2. Clusters- It’s a logical grouping of tasks and services.
3. Container Agent- it runs on each container instance within ECS cluster. It sends info about resource’s running tasks. It stops/starts tasks whenever it receives request from ECS.

Docker-

It is technology that provides tools to build, deploy run and test your application on a Linux based container.

Docker Images are used to launch a container. After creating the docker image we push it to registry(ECR/ Docker Hub)

ECS Cluster-

It’s a logical grouping of tasks/services. Cluster concepts-

-it is region specific

-clusters can contain many container instances but each container instance can be registered with only one cluster at a time.

Scaling in and out of instances can be managed by “Capacity Providers” which in turn is associated with an autoscaling group. We can define a “target capacity” to control the capacity of an instance and scale in or out according to that. When using capacity provider, it creates a scaling in and scaling out plan which is monitored by a “target scaling policy” created on your behalf.

Task definition defines parameter that are required to run a task on container

ECS container agent allows container instances to connect to the cluster.

Secrets Manager

AWS service where we can store the secret and rotate it as well. Secret is stored in an encoded format.

SM encrypts the protected text with a KMS customer managed key. This CMK can be default one or created by customer. KMS ensures secure encryption of your secret when at rest.

Whenever SM encrypts a new version of protected secret data, SM requests KMS to generate a new data key from specified CMK.This data key is used for envelope encryption and is stored with the protected data in encrypted format. Whenever secret needs decryption , KMS decrypts with the help pf data key and then immediately dispose the data key.

SM only accepts requests from host using TLS and PFS (Perfect Forward secrecy).

Staging labels helps us to keep track of different versions of secret. Each version can have multiple staging label but the reverse is not true.

Secret Rotation-

We can rotate the secret manually after a time interval. Secret rotation also changes the passwords in the databases which are compatible with the SM service. If not compatible, then we have to build a lambda function which will change the passwords in th required places whenever rotation happens.

By default “AWSCURRENT” labeled version is picked when SM is called until we mention any specific label/version. A label is unique amongst all the version. We cant create 2 labels with same name in 2 different versions.

The following stages occur whenever rotation happens—

-The rotation function creates a new set of credentials to access the db. It stores the new cred with a label AWSPENDING

- Rotation function tests the AWSPENDING version to ensure new version is working fine.

-Once tested and everything goes fine, the AWSPENDING is changed to AWSCURRENT ant the previous password label is changed to AWSPREVIOUS

Staging label- A label given to different versions of secret. A version can have upto 20 labels but the reverse is not true. Label is unique amongst all the version of a secret. A version must always have a staging label attached to it.

Version- Multiple versions of secret can exist which is differentiated by the label attached to it.

When an IAM principal in one account tries to access secret form another account then that secret must be encrypted with a CMK and permission to access both secret and CMK must be given. Default KMS cant work coz it can allow access to principal of its own account. Alternatively we can create a role to access KMS and secret in the account where secret is present and attach it to the user of another account.

Secret Rotation-

When we enable secret rotation by using the “Credentials for RDS”, “Creds for RedShift cluster” and “Creds for DocumentDB” , SM provides you with a lambda which rotates the in both places.

But if we are using any another type of secret then we have to write lambda on our own to rotate the secret in the 3rd party provider.

By default SM endpoints runs on public internet.

Simple System Manager (SSM)

Service which we use to view and control our infrastructure on AWS.

For SSM to operate in our server, we need to have ssm agent installed in our system as well as proper IAM permissions attached to the instances to let the SSM run on it.

SSM agent is a software which lets SSM operations to be run on our instances.

Different types of management under SSM-

1. Operations Management- It helps us to manage AWS resources.
2. Application Management- It helps us to manage the applications running on AWS.
3. Change Management- It helps in changing the AWS resources
4. Node Management- It helps to manage ec2, on premise servers and virtual machines in hybrid environment.
5. Shared resources- It is shared amongst various ec2.

SSM agent has root access on our instance. So we should have proper IAM restrictions to avoid any malicious code to be run in our ec2.

How to reference AWS SM secret through Parameter store parameters-

By using PS we can call out the secrets stored in SM. The following boundaries apply-

* We can retrieve SM secret only by using “getParameter” and “getParameters” and not by any other call like “getParameterByPath”.
* When we retrieve secret from PS then the parameter name should be like-

/aws/reference/secretsmanager/secret\_ID

* If an user doesn’t have access to a secret then he cant retrieve that secret even by using PS.
* PS reference to SM cant use PS versioning feature.

Operational Management-

1.Explorer- It is customizable dashboard that reports information about AWS resources. It helps us to understand the state and operational risks of AWS resources. Eg- ec2 instance CPU utilization, detached EBS volumes, etc. It is a report hub where DevOps managers view summaries of operational data.

2.Opscenter- It provides a central location where engineers can view, investigate and resolve the operational works related to AWS resources. It provides data for each issue and thus reduces the total time of resolution. It is integrated with Amazon Event Bridge and CloudWatch. It provides data collected from different AWS services on a single dashboard.

3.Cloud Watch- It is a customizable dashboard that can monitor resources under a single view even of another region

4. Personal Health dashboard- it monitors the health check of resources in our aws account.

Application Management-

1.Application Manager- It helps to investigate the issue in terms of application. It collects data from various AWS services and display under a single console.

2.AppConfig- used to create, manage and quickly deploy application configurations. We can use app config with applications hosted on ec2, lambda, containers etc.

3. Parameter Store- It is a secure and scalable secret management service with no servers to maintain it. It supports 3 types of parameters- String, String List, secureString.

String- It contains a single string

StringList- Comma separated list of values

SecureString- Any sensitive data that needs to be stored and referenced in a secure manner. The data is encrypted/decrypted using a KMS key.

2 tiers of PM-

1.Standard parameter- size of parameter is 4kb. No. of parameter limit-10000

2.Advanced parameter- size of parameter is 8kb. No. of parameter limit-100000

Advanced parameter cant be converted into standard while the vise versa is possible.

3.Intelligent tiering- It automatically updates PS from Standard to Advanced tier on the basis of size of parameter/ no of parameters stored.

We can increase the max no of transition in PS by setting up throughput limit.

About storing parameters-

* Parameters are case sensitive
* Parameters cant include spaces
* Valid characters- a-z A-Z 0-9 \_ . –
* Length of arn can be upto 1011 characters
* Hierarchy depth can be upto 15

Change Management-

Change Manager- It is used for managing the changes to both AWS resources and on premise resources

Automation- It helps in maintenance and deployment tasks of AWS resources.

Change calendar- It helps us to set up date and time ranges actions has to or not to be performed.

Maintenance Window- It helps to define a schedule for when to perform maintenance activities such as patching.

Node Management-

Fleet Manager- we can view health and performance of entire server fleet/group on a single console. We can also perform common tasks on the entire fleet by using this service.

Compliance- we can use this service to scan the fleet of managed instances for patch compliance and configuration in consistencies.

Inventory- It provides visibility into ec2 instances. Using this service we can collect metadata from the instance

Managed instances- List of instance in which AWS SSM is configured

Session manager- It helps to securely login to an instance without need of opening the port or maintaining a bastion host and ssh key. SM creates a TLS1.2 encrypted channel between the user and the machine by looking into the IAM permission of the user

Run Command- It lets us to run any command on the managed instances securely.

State manager- It is a service which automates the process of keeping ec2 in state that we define. It can be used to run commands on the ec2 instance for lifetime. State Manager “association” is the configuration that is to be carried out in the instance. “Schedule” is defined with associate which says when the association has to be applied

Patch Manager- It updates/patches the managed instances with security related and other updates.

Distributor-

Shared resource-

SSM document- It is set of commands written in form of document which can be run in any managed instance.

AWS LAMBDA-

It helps us to run any code without provisioning or managing server. It is also called serverless. We have to pay only for the compute time. No charge for the code which is not running. We can also configure lambda to run in response of an event such as changes of data in s3 or dynamo db.such as changes of data in s3 or dynamo db.

Concepts-

Function- It is a resource that we invoke to run the code.

Execution environment- It provides a secure and isolated runtime env for Lambda function.

Deployment package- We deploy the code using this package. Lambda supports-

* A .zip file archieve
* A container image that is compatible with Open container initiative.

Layer- A .zip file which contains all the libraries and dependencies.

Runtime- The language in which the code is written. It provides language specific env that runs in execution env.

Event- it’s a json formatted document that contains data required for the lambda function to run

Interview questions---

Can we launch instances of multiple type in an autoscaling? – Yes we can. If we do not define any instance type in the launch template, then we can define demand/on spot instances in the autoscaling configurations. We can define the capacity that we can send to demand and on-spot. Also we can choose different instance type and define weightage against it.

Max no of policies that can be attached to IAM user/role- 10 by default and can be increased upto 20.

No 0f groups- 300(default) 500(max)

No of roles in an account- 1000(default) 5000(max)

Access key assigned to IAM user- 2

Access key assigned to root user- 2

Can we associate multiple target group under one auto scaling group?? – Yes

What is maximum size of ebs volume??—16 TiB

How do we access the instance if we lost key pair of it?--> Stop the instance and launch a temporary instance. Detach the root volume from original instance and attach to temporary instance. Login to temp instance and then change the key in authorized\_keys for the original instance’s root volume. Now detach this volume and attach to the original instance.

AWS services which are not region specific- IAM/ Cloudfront/ Route53/ AWS WAF/ S3

Different types 0f AMI-

1. Fully baked AMI- It’s a ready to use AMI in which the OS, logging and required softwares are pre installed.
2. Just OS AMI- Only the OS agent is baked. On top of it we have to deploy the required software/runtimes, logging, monitoring and security. It is very flexible.
3. Hybrid AMI- The OS along with security, logging and monitoring is baked. Runtimes have to installed at time of deployment.

How can we easily delete the snapshots after a certain time period- Using Ops Automator. It allows to create, copy and delete EBS.

Difference between EBS and EFS?

What is burstable instance?

Difference between CF and AWS Beanstalk-

CF- It creates and manages the entire infrastructure including ec2,ecs,etc.

Beanstalk- It provides and manages an env that can be used to deploy and run application

EFS doesn’t support snapshot. The data can be replicated from one EFS to another EFS.

2 types of routing provided by R53-

1. Latency based routing- Its goal is to minimize latency.
2. Geo based routing- Routing is done on the basis off the location from where you are hitting the request.

What is git and why is it preferred over other providers?

Git is a version control system used to maintain different versions of the code and the code can be reverted to any of the previous version anytime. It enables multiple developers to make changes to the code at the same time. It is distributed control system

There are 2 types of version control- centralized and distribution control system.

In centralized sytem, we don’t have local repository rather we have a local working copy. All the commits and push is done in the central repo.

IN distributes control system we have 3 layers- local copy, local repo, remote repo

We have many new files in local repo and we don’t want to add few of them to repo while doing git add. How can we achieve this?—create a .gitignore file and add the names of all th files that isn’t to be commited/pushed. Now run the git add cmd

Diff between git pull and git fetch-

Git pull- Fetches all the changes and merge it with your local branch. Git fetch+ git merge

Git fetch- it just fetches the remote repo.

How to clone a particular branch?- using git checkout or git clone – b <branchname> --single-branch <clone\_link>

Different components/features used in the environment-

Netscaler WAF rules protects from the below mentioned attacks-

1. Injection-

* It is configured to prevent SQL injection. We can also configure custom rules to avoid this kind of injection.
* It auto updates the signature so that signatures are upto date.
* Form Field consistency feature help to learn the pattern and later we can configure it to block/unblock based on pattern.
* Buffer overflow checks are present which determines the length of the request.

1. Broken authentication and session management-

Netscaler keeps track of cookies in use and doesn’t accept the request if the cookies are tampered.

1. Cross site scripting(XSS)-

XSS is tampering with the web page and including malicious code into it. NS field format can be used to prevent this attack

1. Insecure Direct Object Refernces-

NS is configured with start URL checks that allows user access to a predefined whitelist of URLs.

1. Cross site request forgery-

In this attack the attacker makes the user to do changes to its account unintentionally like changing the email id or password and thus the attacker gains full control over users account.

NS comes with CSRF whitelisting feature to avoid this attack.

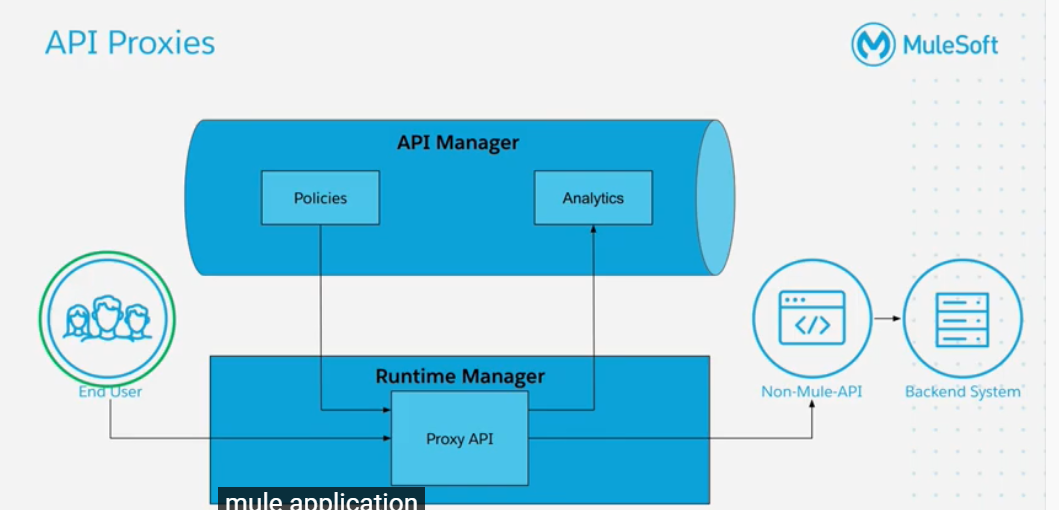
Protection against DDOS attack-

Protection against Layer 3-4 SYN DDos attack- Ns maintains a SYN cookie rather than a half-open connection. Full connection is established only when ACK packet is received. This prevents DDOS attack in SYN.

Mulesoft-

It is used as an API gateway where all the APIs are exposed. It provides secure access to APIs. In our env the APIs are not build on mule rather the security policies are applied on mule for all the APIs. So Mule proxies are deployed so that mule can interact with the APIs hosted in non mule platform.

Working of mule-



End user requests for an API. The request goes to runtime manager which takes the API ID to pull the Policies from API Manger. If there is no violation of the policy then the flow will to to Proxy API. Mule runtime analyses the request and sends back the analytics to Analytics in API Manager. The request goes to non mule API proxy which forwards it to the backend system.

Jenkins—

Jenkins is an open source automation tool written in java with plugins build for continuous integration. It makes it easy for the developers to maintain version and do unit testing.

Work flow of Jenkins- It checks for any change in source code at regular interval. If yes, then it prepares for build and notifies the user. If build fails it notifies the user.

Benefits of using Jenkins-

* Cache build failure
* Automatic build report notification
* Achieves CI
* Automation of maven project
* Easy bug tracking in early stage

How to install Jenkins-

1. Install Java 8
2. Install apache tomcat v 9
3. Download Jenkins war file
4. Deploy Jenkins war file on tomcat.
5. Install plugins

To start/stop Jenkins manually— jenkins.exe start

How can you clone a repo via Jenkins- Enter email and user name. Now execute the cmd- git config

How to setup Jenkins job- From Jenkins dashboard select New Job. Select build free project. Now mentions SCM(where Source code resides).configure optional triggers, build script that builds the project. Now configure the optional options such as archieving the artifact or build results.

Appspec file have hooks in it which we can define different deployment lifecycle events, such as-what has to be done before installation after installation etc. It can outline the files that are required, parameter that needs to be set and commands that needs to be run by codeDeploy agent in your container. Eg-

version: 0.0

os: linux

files:

- source: /

destination: /var/www/html/WordPress

hooks:

BeforeInstall:

- location: scripts/install\_dependencies.sh

timeout: 300

runas: root

AfterInstall:

- location: scripts/change\_permissions.sh

timeout: 300

runas: root

ApplicationStart:

- location: scripts/start\_server.sh

- location: scripts/create\_test\_db.sh

timeout: 300

runas: root

ApplicationStop:

- location: scripts/stop\_server.sh

timeout: 300

runas: root

Buildspec file- It is collection of build commands and related setting. It tells what has to be done in pre-build,build and post build phases. Eg-

version: 0.2

run-as: Linux-user-name

env:

shell: shell-tag

variables:

key: "value"

key: "value"

parameter-store:

key: "value"

key: "value"

exported-variables:

- variable

- variable

secrets-manager:

key: secret-id:json-key:version-stage:version-id

git-credential-helper: no | yes

proxy:

upload-artifacts: no | yes

logs: no | yes

batch:

fast-fail: false | true

# build-list:

# build-matrix:

# build-graph:

phases:

install:

run-as: Linux-user-name

on-failure: ABORT | CONTINUE

runtime-versions:

runtime: version

runtime: version

commands:

- command

- command

finally:

- command

- command

pre\_build:

run-as: Linux-user-name

on-failure: ABORT | CONTINUE

commands:

- command

- command

finally:

- command

- command

build:

run-as: Linux-user-name

on-failure: ABORT | CONTINUE

commands:

- command

- command

finally:

- command

- command

post\_build:

run-as: Linux-user-name

on-failure: ABORT | CONTINUE

commands:

- command

- command

finally:

- command

- command

reports:

report-group-name-or-arn:

files:

- location

- location

base-directory: location

discard-paths: no | yes

file-format: report-format

artifacts:

files:

- location

- location

name: artifact-name

discard-paths: no | yes

base-directory: location

exclude-paths: excluded paths

enable-symlinks: no | yes

s3-prefix: prefix

secondary-artifacts:

artifactIdentifier:

files:

- location

- location

name: secondary-artifact-name

discard-paths: no | yes

base-directory: location

artifactIdentifier:

files:

- location

- location

discard-paths: no | yes

base-directory: location

cache:

paths:

- path

- path

CI Pipeline- Continuous Integration— Code/Build/Test

The code is pushed by API team on bit bucket. In the CI pipeline, the code is retrieved from the bit bucket, unzipped, few values are changed according to the env., the code is then pushed to S3 bucket. At end of CI we have an artifact that is input for CD

CD pipeline- Provision/Deploy/accept test

It compiles the source code, runs tests and produce software package that is ready to deploy

It creates an image and then pushes it to the ECR. It creates a task definition with all the necessary information to run the task. After creating the task definition it creates service in cluster.

Jenkins Setup – Install Jenkins on the server. It will automatically start at port 8080. Login to the instance using http://<publicIP>:8080. It will ask for a password that is present in the mentioned file. Put that password and then it will ask for setting up permanent user and password. Install the recommended plugins.

Master-Slave architecture – Master is just the controller and actual runtime is the slave. All the required softwares for building the project needs to be installed in the slave. We need to configure a new node and do the required settings like provide the IP of the slave, key to login into it.

For key- Generate a key in the master. Copy the public key to the slave instance’s authorized\_key. Try doing ssh from master to the slave. The host name and a cipher will be populated in the known\_hosts file of the master and the master will be able to connect to slave. In the UI the slave will show connected to master.

Development of pipeline and groovy- We can trigger another job on finishing of the 1st job. This is called pipeline. We can also define this pipeline as a code written in groovy rather than configuring it from console. 2 types of pipeline-

1. Scripted pipeline- Groovy script is written in Jenkins console. Syntax-

node {

stage(‘Build’){

//

}

stage(‘Deploy’){

//

}

}

1. Declarative pipeline- Jenkins file is created and stores in an SCM, pulled and then run. Syntax-

pipeline {

agent any

stages {

stage ('clean workspace') {

steps {

deleteDir()

}

}

stage('Non-Parallel Stage') {

agent {

label "master"

}

steps {

echo 'This stage will be executed first'

}

}

}

How do we build a project with multiple git repo🡪 using a plugin called Multiple SCM, we can configure the job to pull code from several git repo

Difference between monolithic application and microservices—

In an application say Uber, all the module like location notification payments mail, etc were present in the same server and interacted with each other. Single deployment.

In microservices, we have all these module are deployed separately and they interact with each other using json. Now if all the modules were deployed in different instances then we might require many ec2 which wont be fully utilized all the time. So we introduced dockerization where multiple microservices can be run in a single ec2 independent of each other.

Now how do I manage these containers? How do we orchestrate the containers?—Kubernetes/ ECS. Kubernetes gives a lot of flexibility with what we can do with our containers.

Kubernetes is a container orchestration tool used for applications distributed over multiple containers. Its job is to monitor, scale, restart containers automatically

What is blue-green deployment?

How does maven work?—we have a file pom.xml where all the dependencies are defined, which are 1st searched in local repo else central repo. Then in our Jenkins pipeline we define what has to be done like mvn compile, mvn package etc.

Settings.xml in maven—if defines local repo location, alternative remote repo location and authentication required for private repos.

Service- it’s a logical set of pods. It provides an abstraction on top of pod which provides a single IP address and DNS name by which pods can be accessed. It helps in load balancing

Pod- It represents a process running on the cluster. It is collection of container

GIT Fundamentals

2 types of version control-

1. Central version control-



Only 1 central server is responsible for all the changes. If central server goes down no one can contribute to the project

1. Distributed version control-

The do not just copy the latest snapshot of files but they copy the repo including the history. Thus if the central server dies, the client can make changes in its local repo and then push it whenever server comes up.



3 stages in Git-

1. Modified- changed the file but not committed to db
2. Staged- means you have marked modified file in its current version to go into your next commit snapshot.
3. Commit- data is safely stored in local repo

Default branch name is “master”. We can also set diff default branch name. eg- set default branch as main-

Git config --global init.defaultBranch main

Getting a Git repo-

1. You can take a local directory that is currently not under version control, and turn it into a Git repository

Go to the folder which you want to version control.

Type- git init

Git add \*.c

Git add LICENSE

Git commit –m ‘Initial commit’

1. You can **clone** an existing Git repository from elsewhere.

Git clone <repo url>

How to add a new file into repo-

Create the file in local and then

Git add <filename>

How to stage a modified file-\

Use git add <filename>- its multipurpose cmd.

Git commit –m ‘message’

Gitignore- the files which we don’t want to be staged/committed anytime, we put name of those file in .gitignore file

Viewing staged and unstaged files- git status

To see what is changed but not staged- git diff

To see what is staged and will got to next commit- git diff --staged

Moving files- git mv file\_from file\_to

View commit history- git log

We can customize this logging by using several filter out mech

To undo commit- git commit --amend

Upstaging the staged file- git reset HEAD <filename>/ git restore --staged <file\_name>

Unmodifying the modified file- git checkout -- <filename>/ git restore <filename>

Adding remote repo- git remote add <shortname> <url>

Git getch- only downloads the data to local repo, it doesn’t merge it to your local repo

How git commits work- say we had added 3 new files and made a git commit. This is how it happens-



Snapshotting in git- The new tree/commit always points to its parent snapshot as shown-



Creating a new branch- git branch testing

This creates a new pointer to the same commit you currently are in



Now- git checkout testing



Now we make a new file and commit it- vim test.rb

Git commit –a –m “made a change”



Now lets get back to master branch- git checkout master

And add file in master- vim test.rb

Git commit –a –m “made changes in master”



Merging the test branch to master-

Git checkout master

Git merge testing

Git branch –d testing ##delete the branch testing

Merge conflict- if we have changed same part of same file in 2 diff branches and want to merge it, git will create a merge conflict. To solve this issue first check which file has the conflict by “git status”

Changing branch name- git branch --move master main

Rebasing- It creates a copy of the new committed files of other branch into the master branch.



Git checkout experiment

Git rebase master



Git checkout master

Git merge experiment



Git Flow model-

1. A master branch is creater
2. Devlop branch is creates from master branch
3. Release branch and feature branch are created from develop branch
4. When feature is complete its is merged to develop
5. When release branch is complete it is merged to both master and develop
6. If an issue is found in master, hotfix branch is pulled, after fixing the issue it is again merged to master and develop.

How Kubernetes work🡪

It has a master and several nodes to it. Master is for scheduling the nodes, monitor them and keep track of logs

Architecture—

Client-server architecture where master is installed in one machine and nodes on separate linux machines.

Components in master-

1. Etcd- stores config info which can be used by each nodes in the cluster.
2. API Server- Kubenetes is an API server which provides all operations on cluster using the APi

Kubeconfig- A package that can be used for communication. It exposes kubernetes API.

1. Kube-controller-manager—runs controller. 4 kinds of controllers-
2. Node controller- responsible for noticing and responding to when a node goes down.
3. Replication controller- responsible for maintaining correct no of pods
4. Endpoint controller- joins services and pods
5. Service Account and token controllers- create default accounts and API access token for new namespace.
6. Scheduler- responsible for tracking the workload and distributing over diff nodes on basis of availability.
7. Cloud-controller-manager—interacts with the underlying cloud provider.types-
8. Service controller- for creating updating deleting cloud provider load balancer
9. Volume controller
10. Node controller- checks if node has been deleted in cloud
11. Route controller- for setting up the route

Components of node-

1. Container runtime- software responsible for running containers. Kubernetes supports- docker, containerd, etc.
2. Kubelet service- it interacts with etcd store to read configs. It also communicates with master to receive commands and work.
3. Kubernetes proxy service – runs on each node and make services available to external host. It helps in forwarding the request to correct container and does load balancing.

Add-ons-

DNS- a must add-on

Web-UI

Container resource managing

Cluster-level logging

Kubernetes Object- it is a defined no.of objects say replicas which is maintained by kubernetes all the time. To work with kb obj we need to use KB API

Object spec- describes characteristics of the object

Object status- actual state of object that is required.

Kubectl basic commands-

Kubectl create –f <filename>

Kubectl get <resource\_name>

Kubectl expose—to expose an endpoint to the outside world

Kubectl delete – deletes the specified file/dir/pod etc

Kubectl autoscale