**ELASTIC-CLOUD-COMPUTE :** Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It allows us to create, configure, and manage virtual servers (known as instances) in the AWS cloud.

We can launch instances based on pre-configured Amazon Machine Images (AMIs). AMI is a pre-configured template that contains the information like operating system, applications, data, and configuration settings.Also we can create custom AMIs by taking snapshot of instnace.These instances run within virtual private clouds (VPCs) and can be configured with various resources like CPU, memory, storage, and networking.

We can launch different types of instances such as general-purpose(m5.large-2,8GiB), memory-optimized(r4.xlarge-4,30.5GiB), compute-optimized(c5.xlarge-4,8GiB), storage optimized(i3.large-2,15.25GiB with NVMe SSD 475 GiB), GPU optimized(p3.2xlarge-8,16GiB) & Burstable performance(t3.medium-2,4GiB temporarily increase its CPU performance beyond its baseline performance level).

To scale comupting power of EC2 machine we have to change the instance type. First we need to stop the instance after that we change instance type & start the instance.

As per pricing model we can have On-demand instances, Reserved instances & spot instances.

- On-Demand Instances: Pay-as-you-go pricing with no upfront commitment.

- Reserved Instances: Provides capacity reservation at a lower cost in exchange for a commitment.

- Spot Instances: Allows users to bid on unused EC2 capacity, potentially leading to significantly lower costs.

We can place instances in multiple Availability Zones (AZs) within a region. This helps ensure redundancy and fault tolerance.

We can also enhance the security of EC2 instances by using security groups, Network ACLs, key pairs, and configuring firewalls. Additionally, implementing multi-factor authentication (MFA) is recommended for account access.

EIP : We can assign public when launching instance but it can change if the instance is stopped and started. An Elastic IP is a static public IP address which provide a consistent/permanent public IP even after stopping and starting the instance the public IP will not change.

ENI : Elastic Network Interfaces are static private ip address. Whenever we launch a instance, it creates a ENI in the background which gets released automatically after instance termination. We can also create our own eni to keep our private ip address static. ENI is responsible for network level properties like security group , subnet , private ip.We can attach 3 ENI max to a single instance

We can scale our application horizontally by adding more instances. Amazon EC2 Auto Scaling helps you automatically adjust the number of instances based on demand.

EBS : Amazon Elastic Block Store (EBS) provides persistent block storage volumes for EC2 instances. EBS volumes can be attached to instances and used as data storage. We can encrypt EBS volumes using Amazon EBS encryption or KMS encryption.We can create encrypted volumes during instance launch or we can encrypt existing volumes also.

EBS volume types :

Provisioned IOPS (SSD) Volumes (io1): Designed for I/O-intensive workloads

Throughput Optimized HDD Volumes (st1): Designed for frequently accessed, large, and streaming workloads.

Cold HDD Volumes (sc1): Designed for infrequently accessed, cost-effective workloads.

Magnetic (Standard) Volumes (standard): The original EBS volume type, designed for workloads with balanced I/O performance and cost.

NVMe SSD Volumes (nvme): Introduced for the I3 and I3en instance families.

Fast Snapshot Restore (fsr): allows you to quickly restore EBS volumes from snapshots for instances that support it.

GP3 Volumes (gp3): An upgrade to the gp2 volumes, offering a balance of price and performance.

Snapshots & AMI :

Since EC2 is IAAS, we have to configure the backups ourselves.Snapshots are point in time & cold backup of EBS volumes without impacting the running instances. we can create snapshots of EBS volumes, which serve as backups. These snapshots can be used to create new EBS volumes or restore existing ones. Snapshots cannot be directly attached to a instance , you either need to create a volume or image out of it in order to use it.Snapshots can be used to quickly restore new volumes across the Availability Zones. We can automate snaphot management using Data Lifecycle Manager (DLM). Lifecycle manager lets us schedule backups for our instances.These backups are incremental & we can mention retention period.If we give retention period of 7 days, on 8th backup it will delete oldest one.Incremental means first backup it will be taken from scratch & in the next back up it will record only changes.

AMI : Snapshot approach can become tedious if there are multiple ebs volumes attached to an instance. This limitation is the reason companies stick with AMI. We can create our own AMI, which can be used as a backup or can be used as template to avoid reinstallation of service or packages. Since we create AMI from the instance , it records every volume that is attached to the instance and gives us the complete bundle so we do not have to create seperate snapshots.

Instance store instances use ephemeral/temporary storage that is directly attached to the instance, providing high I/O performance. EBS-backed instances use EBS volumes for storage, offering persistent data storage & we can detach or attach EBS volumes.

Instance metadata provides information about an instance, such as its instance ID, IP address, instance type, public/private Ips, security groups and IAM role.We can access metadata by using URL publicIP/latest/metadata.User data is information that you can pass during launch of an instance.It is a bootstrapping process.It is used for running scripts, install software, configure settings.

When launching instances, we can choose a specific VPC and subnet. This ensures that the instances are launched within the defined network environment.

An EC2 security group acts as a virtual firewall for instances to control inbound and outbound traffic. You can specify rules to allow or deny traffic based on IP addresses and ports.

You can use AWS CloudFormation to create and manage a collection of related AWS resources, including EC2 instances. This allows you to define the infrastructure as code.

**SYSTEMS-MANAGER :** AWS Systems Manager is a service that provides centralized management for AWS resources, helping you automate tasks, manage configurations, and improve overall operational efficiency.Host management

By using Run Command we can remotely execute commands on a group of managed instances. We can automate the patching of managed instances using Patch Manager.We can store confidential data like password, keys inside Parameter Store.

AWS Systems Manager is an AWS service that you can use to view and control your infrastructure on AWS.We can view operational data from multiple AWS services & automate tasks across AWS resources.It helps you to maintain security & compliance by scanning your managed instances & reporting on any policy voilations.

Systems Manager also helps you configure and maintain your managed instances. Managed instance is the instance configured with system manager.It supports different machines like EC2, on-premise servers & virtual machines in other cloud environments.It uses SSM agent to update, manage & configure these resources.

We can manage configuration of managed instances remotely by using RUN command of system manager.Instances must be managed by system manager to use RUN command. We can save output of run command in

**AWS-CLI :** The AWS Command Line Interface (CLI) is a unified tool that allows you to interact with various AWS services using command-line commands.

**IAM :** AWS IAM is a service that allows you to manage users, groups, and permissions for accessing AWS resources. It provides centralized control over authentication and authorization.

Identity-Based Policies, Resource-Based Policies, Inline Policies, Managed Policies,MFA

We can grant access to IAM users or groups in AWS by attaching policies. These policies specify what actions are allowed or denied on AWS resources.Also IAM roles are used to grant access to one AWS services to access other AWS service.E.g EC2 instance to read files from s3 bucket.To enhance security we can use MFA multi factor authentication.We can also create inline policies which are specific to one user or group or role only.In each policy we define statements.In each statement we have 3 fields effect, Action & Resource. Effect is either allow or deny, Action is actual permission like listbucket, deletebucket & Resource is arn of aws resource.

**S3 :** Amazon Simple Storage Service (Amazon S3) is a scalable object storage service designed to store and retrieve any amount of data from anywhere on the web. It's commonly used to store files, backups, images, videos, and more.

It provides various storage classes like S3 standard for frequent data access, S3 Intelligent-Tiering for optimizing cost, S3 Standard-IA less frequent access, S3 Glacier, S3 Glacier deep archieve. We can apply lifcycle policies on objects which automatically move objects from one storage class to other.Also we can use ACLs to control object level access & bucket policies to control bucket level access.By using versioning we can keep multiple version of single file.We can host static website using S3.We can replicate bucket object by cross region replication.For temporary access to S3 object we can use presigned URL.

**VPC :** Amazon VPC is a logically isolated section of the AWS Cloud where you can launch resources in a virtual network that you define. It allows you to control your network environment, including IP addresses, subnets, and security settings.

We can create subnets to segregate VPC network.These subnets are associated with different availability zones.To control traffic between these subnets we use route tables.To secure VPC network we can use security groups & NACLs.Security groups provide security at instance level while NACl provide security at subnet level.Using internet gateway our resources in VPC can access the internet.We can also have elastic ip which permanent public IP to EC2 instance.We can configure VPC peering to connect 2 VPCs without internet.Using VPN & AWS dirrect connect we can connect on-premise servers with VPC.We can configure VPC flow logs which records ingress & egress traffic of VPC.

**ELB :** An Elastic Load Balancer (ELB) is a managed AWS service that automatically distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, containers, or IP addresses, to ensure high availability and fault tolerance.

There are three types of Elastic Load Balancers: Application Load Balancer (ALB), Network Load Balancer (NLB), and Gateway Load Balancer (GWLB).

ALB operates at the application layer and supports advanced routing, including content-based routing and path-based routing. ALB is the distribution of requests based on application-layer data.It makes routing decisions based on application-level data such as HTTP/HTTPS requests, URL paths, headers and content types. Applications are load balanced based on their peculiar behavior and not solely on server (operating system or virtualization layer) information. Content-based routing looks at the content or data within a request to make routing decisions, while path-based routing focuses on the URL path to determine where traffic should be sent.

NLB is distribution of traffic based on network variables such as IP address and destination ports. NLB operates at the layer 3 & 4 which is network & transport layer and it provides ultra-low latency and high throughput which is useful fot gaming applications and real-time streaming applications. The network load balancer just forward requests whereas the application load balancer examines the contents of the HTTP request header to determine where to route the request.

ALB supports path-based routing.We can configure rules for listener that forward requests based on the URL in the request.This structures our application as smaller services and route requests to the correct service based on the content of the URL. Primarily for http/https requests.NLB has ability to handle volatile workloads and scale to millions of requests per second. Listener decides on which port the load balancer will listen to requests.

user initiates an HTTP request > ALB listener listens for incoming request > based on listener rules defined request is forwarded to target group > instances in target group handle the request & generate response > ALB send response back to user.

ALB create steps : Name ALB > Configure Basic Settings like listener settings listener protocol and port, selecting a VPC, choosing availability zones > Configure Routing – define target groups, Set up listener rules for routing such as URL paths, hostnames, or query parameters > enable stickiness in target group attribute > Register Targets – register instances in target group > Configure Load Balancing Algorithms - round robin or least outstanding requests > Configure Health Checks - define the health check protocol, port and path as well as healthy and unhealthy thresholds > Add tags .

A target group is a logical grouping of targets (such as EC2 instances, IP addresses or AWS Lambda functions) registered with a load balancer. ALB and NLB use target groups to route requests to registered targets.

Elastic Load Balancers perform health checks on registered targets to ensure they are available to receive traffic. These health checks performed by configuring specific thresholds. If we set the healthy threshold to 3, it means that a target(instance) must pass three consecutive health checks to be marked as healthy & same for unhealthy threshold.Unhealthy targets are temporarily removed from rotation.

Cross-zone load balancing is a feature that evenly distributes traffic across all registered targets in all availability zones, helping to achieve even distribution and better resource utilization.

**CLOUDWATCH** **:** Amazon CloudWatch is a monitoring and observability service that provides insights into your AWS resources and applications by collecting and tracking metrics, logs, and events.

It uses metrics like CPU utilization, memory usage etc for monitoring AWS resources.These metrics are organized within namespace.Dimensions are used to filter the metrics using attributes like instance IDs or resource names.

Metrics are collected over specified time intervals & aggreated.Cloudwatch alarms gets triggered when a specific metric value goes above or below the thresold limit.Cloudwatch logs collect, store & monitor data from AWS resources and applications.Cloudwatch event provides real time stream of events of AWS resources.To visualize & analyze data we can create dashboard in cloudwatch.We can set data retention policy which automatically deletes log data after certain period.Using insights we can query data from cloudwatch logs.

CLOUDTRAIL : AWS CloudTrail is a service that provides governance, compliance, and audit capabilities by recording and storing API calls made on your AWS account.

The recorded data is stored in JSON format called as events. Events are categorized as read-only or write-only events.Read-only events are about reading or viewing AWS resources, while write-only events involve creating, modifying, or deleting AWS resources.These events can cover a wide range of activities, from launching an EC2 instance to creating an S3 bucket.We can use event selector to filter events.

**RDS :** Amazon RDS is a managed relational database service that simplifies database setup, operation, and scaling. It supports various database engines like MySQL, PostgreSQL, Oracle, SQL Server, and Amazon Aurora.

DB parameter group allows us to customize database settings & DB option group allows us to customize database instnace settings.We can achieve high availability through multi-AZ deployments also we can create read replica of database to improve its performance.We can use DB snapshot which is point in time copy of database instance.Access control of database instance is managed by DB security group.We can monitor and optimize performance using features like DB events and Performance Insights.

**DYNAMODB :** Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. It's designed to handle massive amounts of structured data across various use cases. It organizes data into tables, with each table containing multiple items.Each item represents data and has attributes.Max item size is 400kb. The primary key is used to uniquely identify items within a table & it consists of partition key and sort key. Partition keys are important for distributing and retrieving data, while sort keys help us to query data efficiently within partition.It offers two consistency models: eventually consistent reads and strongly consistent reads, allowing users to balance performance and data consistency.It uses DAX DynamoDB accelerator which is in-memory caching service to improve read performance.Also Global Tables used to enable multi-region, active-active replication of DynamoDB tables, providing high availability and disaster recovery.

**ELASTIC-BEAN-STALK :** AWS Elastic Beanstalk is a platform-as-a-service (PaaS) offering that simplifies application deployment and management. It handles infrastructure provisioning, deployment, monitoring, and scaling, allowing developers to focus on writing code.

**ROUTE53 :** Amazon Route 53 is a scalable and highly available Domain Name System (DNS) web service that helps route end-user requests to AWS resources or external endpoints.

DNS is a global system that translates domain names into IP address.It is a highly available Domain Name System (DNS) web service that plays a pivotal role in managing domain names and routing traffic across the internet. Its key components include hosted zones, record sets, and various DNS record types like A, AAAA, CNAME, MX, and TXT.

A record maps domain name to ipv4 address, AAAA record maps domain name to ipv4 address, CNAME creates alias for domain names, MX for mail server, TXT for txt data, NS record for name servers for domain, SOA record contains administrative information about the zones. Route 53 allows you to route traffic to different resources based on routing policies and health checks.Routing policies includes policies such as Simple Routing, Weighted Routing, Latency Routing, Failover Routing, Geolocation Routing, Geoproximity Routing, Multivalue Answer Routing.In checks the health of each resource & routes the traffic to healthy resource.Alias record allows you to route the traffic to AWS redources like S3 bucket, cloudfront, ELB.TTL(Time to Live) decides how DNS records are managed and how quickly changes to these records are distributed across the internet.

**LAMBDA-FUNCTIONS :** AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers. It automatically scales and manages the infrastructure required to run your code in response to events.

In lambda function there is a default function called lambda handler which is an entrypoint to lambda.It takes two arguments event & context both are provided by the resource/service that invoke lambda function. For e.g cloudwatch invokes lambda function so the event is cloudwatch.Whatever invokes or trigger lambda function is event.It is in JSON format To see what is inside the event we use print(event).By using these event details we write logic of python code.Copy these details from log group & paste in any json formatter.

import boto3

def get\_vol\_id\_from\_arn(volume\_arn):

arn\_parts = volume\_arn.split(':')

volume\_id = arn\_parts[-1].split(/)[-1]

return volume\_id

def lambda\_handler(event, context):

# TODO implement

print(event)

{

"version":"0",

"id":"b446a0d7-d6c8-d421-c353-6c9edf854dee",

"detail-type":"EBS Volume Notification",

"source":"aws.ec2",

"account":"214083310546",

"time":"2023-10-26T09:17:46Z",

"region":"us-east-1",

"resources":[

"arn:aws:ec2:us-east-1:214083310546:volume/vol-04b61bbabb7ca3678"

],

"detail":{

"result":"available",

"cause":"",

"event":"createVolume",

"request-id":"cd072adb-0d17-4382-adb5-d8c8bc3330d5"

}

}

volume\_arn = event['resources'][0]

ebs\_volume\_id = get\_vol\_id\_from\_arn(volume\_arn)

ec2\_client = boto3.client('ec2')

response = ec2\_client.modify\_volume(

VolumeId=ebs\_volume\_id,

VolumeType='gp3'

)

return {

'statusCode': 200,

'body': json.dumps('Hello from Lambda!')

}

Create lambda function > Create event > print event in lambda-handler > copy info from log group > paste in json formattor > build logic on basis of this info >>

parse json data to get arn & store it in variable volume\_arn

create a function get\_vol\_id with volume\_arn as parameter to get only volume id (function returns volume\_id)

call the function get\_vol\_id with volume\_arn as argument in lambda\_handler function & store it in volume\_id which was returned from get

from arn parse json data to get arn which is input to get\_volume\_id function >

To interact with AWS services we use client object.It provides programmatic access to AWS services.We can interact to AWS services using client object which allows us to make API calls, send requests & manage resources.

AWS provides different runtime environments to run lambda function such as Node.js, Python, Java, C#, Ruby etc.Lambda function gets executed in response to event source.These events sources originate from various AWS services like Amazon S3, AWS API Gateway, or Amazon Kinesis.These events received by the lambda function which invokes the specified handler function. Then, the Lambda function executes the handler code, processes the event and performing actions based on the event's content.Code execution can start with a "Cold Start," involving initialization and resource allocation, or a "Warm Start," if the function is reused without an idle period.Throughout execution, logs are generated and sent to Amazon CloudWatch Logs for monitoring and debugging.After that a response is sent back to the event source.

Execution Role defines what other AWS services a Lambda function can interact with.

Concurrency is the number of requests that your function is handling simultaneously. AWS Lambda automatically scales the concurrency based on the number of requests in the queue.

Invocation Type: Specifies how a function is invoked, either "Event" for asynchronous or "RequestResponse" for synchronous invocation.

Dead Letter Queue (DLQ): An Amazon SQS queue or Amazon SNS topic that you configure as a destination for unprocessed events when a Lambda function fails.

AWS Step Functions: A serverless orchestration service that allows you to coordinate multiple AWS services, including Lambda functions, into serverless workflows.

**CLOUDFRONT :** Amazon CloudFront is a Content Delivery Network (CDN) service provided by AWS that accelerates content delivery by distributing it across a network of edge locations.

These locations cache and serve content, ensuring that users' requests are routed to the nearest edge location for optimal performance. There are two types of distributions: web distributions for websites and RTMP distributions for media streaming. The origin is the source of the content that CloudFront distributes. It can be an Amazon S3 bucket, an EC2 instance, or other HTTP(S) servers.The cache stores frequently accessed content in edge locations. It enables faster content delivery. TTL (Time to Live) is the parameter controlling the duration of content caching, specifying how long content remains valid before a refresh. When a user requests content, it initiates a viewer request, which CloudFront fulfills by performing an origin request, fetching the content from the designated origin server. The result, viewer response, is the content returned to the user.

Custom Domain Name (CNAME) feature allows you to associate a branded or custom URL with the content delivered.

**CLOUDFORMATION :** AWS CloudFormation is a service that allows you to define and provision infrastructure as code, by enabling you to create, update, and manage AWS resources in a declarative and automated way.

It uses JSON or YAML template which contains infrastructure specifications.We declare resources that we want to create in these templates.Then these templates creates stacks, which are collections of AWS resources treated as a single unit.Parameters allow users to input values during stack creation. Outputs provides result of resource creation.Mappings enable value lookups, and conditions used to create resource based on specified logic.

Drift detection & change sets are like a terraform plan command which identify the differences between desired and actual resource configurations and review update before application. Stack policies define permissions for updates & resource types categorize resources.

**CODE-PIPELINE :** AWS CodePipeline is a fully managed continuous integration and continuous delivery (CI/CD) service that automates the release process of software applications. It enables developers to build, test, and deploy their code changes automatically and efficiently.It automates the entire software release process.  
Pipeline defines workflow for your application. Each pipeline is composed of one or more stages.

In the source stage application's source code is retrieved from a source repository, such as AWS CodeCommit or GitHub.Then the build stage compiles and packages the source code into a deployable artifact. These artifacts are temporarily stored in an artifacts store, often an Amazon S3 bucket.After build stage we perform test stage of application.In the deploy stage he artifact is deployed to the target such as AWS EC2 instances, Lambda functions, or Elastic Beanstalk. Throughout this process, IAM roles are assigned to pipeline actions, ensuring the correct permissions to interact with AWS services.

Pipeline can get triggered by webhooks of GitHub or it can also gets triggered by CloudWatch events. We can create cloudWatch Event rule to detect changes in S3 bucket or in CodeCommit repository which triggers pipeline.

**CODE-BUILD :** AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software artifacts, such as executable files or application packages.

First we create project in the codebuild.Then We write buildspec file which contains build phases like install, pre-build, build, post-build, and artifacts.Mostly we write buildspec in YAML format. Then we define build environment which provides the runtime environment for the build such as operating system, runtime version or any custom docker image. CodeBuild executes the build process according to the instructions in the Buildspec file.Once the build is completed successfully it creates artifacts of different types such as ZIP files, TAR, WAR which further stored in S3 bucket.We can check logs also if anything goes wrong.

**CODE-DEPLOY :** AWS CodeDeploy is a fully managed deployment service that automates software deployments to a variety of compute platforms, including Amazon EC2 instances, AWS Lambda functions, and on-premises servers.

In a CodeDeploy Deployment is the process of distributing and updating your application across designated target environments such as EC2 instances or on-premiser servers. Deployment Groups allows you to group target instances logically for code deployment purposes. Deployment Config defines how a deployment proceeds, such as the percentage of instances that must remain healthy during deployment.We can use hooks which are custom scripts we run before & after deployment.We can use 2 deployment strategies Blue/Green Deployment or In-place deployment. Blue/Green Deployment creates a new environment (green) to deploy the updated code while keeping the old environment(blue).Traffic shifted gradually to new environment. In-Place Deployment directly updates the existing instances, potentially stopping and starting services during the process during deployment.We can revert to a previous version of the application code by using Rollback. Deployment Monitor provides insights into deployment status, including success and failure metrics.

**CLOUD-MIGRATION :** Cloud migration refers to the process of moving applications, data, and workloads from on-premises environments or one cloud provider to another.

**ECR :** Amazon Elastic Container Registry (ECR) is a fully managed Docker container registry that makes it easy to store, manage, and deploy Docker container images.

Containers are created from docker Images and ECR acts as a repository to store these images.We can version these images by using image tags.ECR repositories are private by default. We can use IAM policies to control who can push and pull images from the repository.ECR automatically scans container images for software vulnerabilities during the push process.We can pull image from ECR repository & create container out of it.We can apply lifecycle policies to ECR repo which automate the removal of old images based on defined rules.Also we can monitor image and repository activities using cloudwatch logs.

**ECS :** Amazon Elastic Container Service (Amazon ECS) is a fully managed container orchestration service that allows you to run, manage and scale Docker containers on a cluster of Amazon EC2 instances or AWS Fargate.

In the Task Definition we can specify details such as the Docker image to be used, CPU and memory requirements, container ports, and other settings.Then we create ECS Cluster which is logical grouping of container instances.For running our containers we create Container Instances inside cluster.We also need to install & start ECS agent on these instances. ECS agent used for the communication between the ECS control plane and the containers running on your EC2 instances. For long-running applications we create "ECS Service”. ECS Service maintains a specified number of tasks and monitors their health. If a task fails or becomes unhealthy, ECS automatically replaces it to maintain the desired task count. To handle varying workloads, you can configure "Auto Scaling" for your ECS services. For high-traffic applications, we can use "Elastic Load Balancing (ELB)" to distribute incoming traffic evenly across the tasks within your ECS service.For monitoring the performance and health of containers we can use AWS cloudwatch.

**EKS :** Amazon Elastic Kubernetes Service (Amazon EKS) is a fully managed Kubernetes service that makes it easier to deploy, manage, and scale containerized applications using Kubernetes.

Kubernetes is an open-source container orchestration platform used for automating the deployment, scaling, and management of containerized applications. EKS Cluster consists of EC2 instances that are used to run containerized applications.We can interacting with Kubernetes clusters using kubectl.The master node is the control plane of the Kubernetes cluster and it is managed by EKS itself.It is responsible for managing the overall state of the cluster, including scheduling, scaling, and maintaining applications and nodes.Node Group is is a group of EC2 instances or worker nodes within an EKS cluster which are used to run and manage Kubernetes workloads. Kubeconfig file contains configuration information for establishing a connection to the EKS cluster.

A pod is the smallest deployable unit in Kubernetes. It can contain one or more containers, which share the same network namespace and storage volumes

A Kubernetes service is an abstraction that defines a logical set of pods. It provides a consistent endpoint for accessing the pods, regardless of the number or location of the underlying pods.

A Kubernetes deployment is a resource that defines the desired state of a set of pods. It allows for scaling, rolling updates, and self-healing of applications.

Namespaces provide a way to divide cluster resources into multiple virtual clusters within the same physical cluster. They are useful for managing different workloads and environments.

Cluster Autoscaler is a Kubernetes component that automatically adjusts the size of the node group in your EKS cluster based on resource requirements.

**AWS-TERRAFORM :** Terraform is an open-source Infrastructure as Code (IaC) tool that allows you to define, manage, and provision infrastructure resources using declarative code.

**Amazon GuardDuty :** Amazon GuardDuty is a threat detection service that continuously monitors your AWS accounts and workloads for malicious activities and unauthorized behavior. It uses machine learning and anomaly detection to identify security threats.

When GuardDuty detects irregular activities, it generates Findings, which are security alerts for threats.These findings are of different types such as CryptoCurrency, or UnauthorizedAccess.It uses CloudTrail logs and VPC Flow Logs, which help in monitoring user and network activities.It offers Recommendations on how to mitigate the detected threats, providing actionable insights for improving the security posture.

**AWS Key Management Service (KMS) :** AWS KMS is a managed encryption service that makes it easier to create and control the encryption keys used to secure your data. It supports key generation, rotation, and management.

**AWS WAF (Web Application Firewall) :** AWS WAF is a web application firewall that helps protect web applications from common web exploits and attacks, such as SQL injection and cross-site scripting (XSS).

A Web ACL is a set of rules that controls how WAF inspects and controls incoming web requests to your web application. A rule is a condition-action pair that specifies what to do when certain conditions are met. Conditions are criteria that define what to look for in web requests such as IP match condition like specific IP address, String Match Condition like block requests with the keyword "admin", GeoMatch Condition checks the geographic origin of incoming requests. Actions determine what WAF should do when a request matches a rule such as blocking the request. A web request is any request sent by a user's browser or application to your web application.

**Amazon Inspector :** Amazon Inspector is an automated security assessment service that helps you identify vulnerabilities and security issues in your EC2 instances and applications.

First we create assessment templates which contains AWS resources or assessment targets such as EC2 instances, S3 buckets, RDS databases and set of security rules or rules packages. Once we have created and configured an Assessment Template,we can run assessments based on that template. After an assessment run is complete, AWS Inspector provides findings and reports that detail the security and compliance issues discovered during the assessment.

**AWS Secrets Manager :** AWS Secrets Manager helps you protect access to your applications, services, and resources by managing sensitive information such as API keys, passwords, and database credentials.

AWS Secrets Manager allows you to store and manage these secrets securely.

**AWS Single Sign-On (SSO) :** AWS SSO is a cloud-based identity service that enables users to sign in to multiple AWS accounts and third-party applications using a single set of credentials.

**AWS Organizations :** AWS Organizations helps you centrally manage and govern multiple AWS accounts by providing features for policy-based control and consolidated billing.

**AWS Direct Connect :** AWS Direct Connect establishes dedicated network connections from on-premises data centers to AWS, providing more reliable and consistent network performance.

**Amazon SNS (Simple Notification Service) :** Amazon SNS is a messaging service that allows you to send messages, notifications, and alerts to distributed components and services.

**Amazon SQS (Simple Queue Service) :** Amazon SQS is a fully managed message queuing service that helps decouple and distribute workloads in a distributed system.

**AWS Storage Gateway :** AWS Storage Gateway is a hybrid cloud storage service that connects on-premises environments with cloud storage, bridging the gap between your data center and the cloud.

==========================================================================================================================================================================================================================================================================================================**DEVOPS**

**GIT :**

Git is a **distributed version control system** that enables **collaboration and history tracking** in software development. **Git repository is the centralized storage for a project's files** and it contains a detailed record of changes, known as commits.**Commits are like a snapshots of the project at specific points in time**, and **they come with descriptive messages to explain the changes made**.**Developers work on Git repositories through branches, which are separate lines of development.**The **main branch or master branch, represents the latest stable version of the project.** Remote repositories provide collaboration between local and remote copies.**Pulling updates changes from remotes into your local** branch, while **pushing sends your commits back to the remote**.**Merging is the process of combining the changes from one branch into another**. **fast-forward merge is a simple linear move of the branch pointer when there's no divergence, while a three-way merge is used to combine changes from branches with separate and divergent histories.Conflicts can occur when multiple people change the same part of a file or when one person deletes a file or line while someone else is trying to change it.** **When conflict occurs we will see messages like "Merge conflict in xyz file".When we open the file with conflict the code between less than & equal to is changes made by your or current branch & the code between equal to & greater than is changes made by other branch.** **To resolve the conflict, you need to manually decide which changes to keep and which to discard.After saving file we do git add & git commit.** Forks create copies of repositories. A pull request is a request to merge changes from one branch into another branch. It's commonly used for code review and collaboration. Git also employs unique commit hashes, tags, and the HEAD pointer, while features like gitignore, stash, rebase, and cherry-picking provide various tools for managing and customizing your development workflow. In this way, Git offers a comprehensive set of tools and concepts for version control and collaborative software development.

Git rebase is a way of moving the changes from one branch onto another branch.It reorganizes the commit history by replaying changes. fast-forward merge creates a linear history without new commits, and three-way merge reconciles/integrates changes with an explicit merge commit that incorporates the common ancestor.

**In our organization we use 5 type of branches in our branching strategy.Master branch contains production ready code.Then for ongoing development work we use develop branch where developers create feature branches from develop branch to work on a new feature or tasks.feature branches merged into develop branch & Develop branch regularly merged into master when feature is complete & tested.Release branch is created from develop branch for new releases.Once release branch is ready it is merged into both develop & master with release tag.And hotfix branches are created to address critical issues in production.After fixing the issue it is merged both into master & develop branches.**

**In our organization, we have a branching strategy that consists of five types of branches. The main branch is called the master branch, which contains the code that is ready for production.For ongoing development work, we use the develop branch.From the develop branch, developers create feature branches to work on new features or tasks.These feature branches are then merged back into the develop branch once the feature is complete and tested.**

**Additionally, we have a release branch that is created from the develop branch for new releases. Once the release branch is ready, it is merged into both the develop and master branches with a release tag. Finally, we have hotfix branches that are created to address critical issues in production. After fixing the issue, the hotfix branch is merged into both the master and develop branches.**

**git clone is used to create a local copy of a Git repository, while "fork" is a term used in the context of remote Git repository hosting platforms, where you create your own copy of someone else's repository**

**git log division : this will give changes between master & division branch simillar to git checkout division && git log**

**Cherry pick used to pick and apply a specific commit from one branch and apply it to another branch.**

**git merge –abort, git rebase –abort**

**fetch vs pull**

**git fetch is used to retrieve changes from a remote repository and update your local repositories while git pull is a command that combines a git fetch with a subsequent git merge or git rebase. It's used to fetch changes from a remote repository and immediately integrate them into your current branch.**

JENKINS

open-source automation server used for building, deploying, and automating software development pipelines

Pipeline is a set of automation steps and tasks Jenkins support both Declarative and Scripted pipeline

We are using master slave architecture of jenkins for load distribution.

Jenkins integration with promethues for alerting high load :

Install plugins promethues metrics, cloudbees disk usage simple

Configuration settings > path : jenkinsIP:8080/promethues > collecting metric periods : 30 seconds

Vi prometheus.yml >

Scrape timeout should be less than scrape interval default is 15 & 10 viz

In scrape config add >

- job name: ‘prometheus’

metrics\_path: /prometheus

static\_config:

- targets: [‘jenkinsIP:8080’]

MAVEN

NEXUS-SONATYPE-REPOSITORY

SONARQUBE

DOCKER

ANSIBLE

KUBERNETES

**GIT**

Question number one is git a distributed or centralized version control system. What is the difference between them?

So basically centralized version control systems are the traditional version control systems. They are not seriously used these days. I mean, very few people might be using it, but mostly they are not used. They are like SVN, CVS. These are the centralized version control systems. And then you have Git, which is a distributed version control system. The major difference is that in **centralized version control system, you have the client and server architecture.** And server, which is a remote repository, has all the copies of your code, **like all the versions of the code are only available with the server.** Whereas in the **distributed version control system, every developer has all the versions of the code, like all the copies of the code are with each and every developer who has cloned the remote repository.** So that's why it's called distributed version control system, because everything is distributed.

Question number two, what are the Git commands that you use to commit changes to your remote repository?

So you basically use three commands **Git add, Git commit and Git push**. So these are the three commands that you have to use.

What is the difference between Git fetch and Git pull?

So basically Git fetch only, you know, **it informs or shows you about the latest changes that are made to your remote repository**. So you want to see what are the changes. So you can simply do Git fetch. It will show you the information about what are the changes that are made, but it does not merge the changes to your local repository. Whereas **Git pull, it does the same, but it will also get the changes to your local repository.Git pull is combination of git fetch & git merge.**

Question number four, what is the difference between Git merge and Git rebase?

So the difference between both of them is so both of them are actually designed to do the same purpose. That means if you want, like you have changes that you want to bring from branch X to branch Y, you can either do Git merge or Git rebase. **Both of them does the same purpose, that is to bring the core changes from one branch to the other branch but the only difference is how they are projecting the history.** So if you want a linear history of commits you use Git rebase. Whereas you are not bothered about the Git history, you can simply do Git merge. So Git merge is the most, I can say that it covers a lot of disadvantages of Git rebase, but if you're worried about your revision history, then you should always go for Git rebase. It gives a linear code history. So that's why Git rebase is preferred for the linear history.

git merge versus git rebase what's the difference **git merge** **creates a new commit that represents a combination of changes made in both the feature branch and the main branch** let's say you Branch off a master and start to work on your feature Branch at the same time other people have added commits to main when you're ready to merge to main a merge commit will be created that combines both your changes and Main togither **git rebase works by moving the entire feature Branch commits to the tip of the main branch latest commits this results in a linear cleaner history without merge commits being created**

Question 5 what is the difference between .git and .gitignore?

So .git is everything in Git. So you know, **dot git folder contains all the information, like project details, all your commit details, remote & local repository details & configuration settings of git**. Everything is present in .git. All the, all the things are stored as objects in Git.**If you delete the .git, then your all the information related to the repository is lost.** **And if you want to ignore anything from .git, you simply go and put that file information into the .gitignore file.** **So let's assume that you don't want to push the files containing API keys, access tokens, passwords, or other sensitive credentials or log & tmp files** from your local repository to remote repository. So **you can simply add the file reference in your .gitignore**.

Question 6 what are pre commit hooks and what are post commit hooks?

So hook is something you want to perform an action before or after something, then it's called a hook. So **pre commit hooks are the actions that are taken before you do Git commit and post commit hooks are the actions that are, that are taken by Git after the commit.** So let's assume that you have certain files like password files, public key, private key that you don't want to commit to git repository. Or you don't want to accidentally push it to git it.So for that, you can configure them in your pre commit hook and you can tell Git that, you know, before every commit, just execute this script. And what Git does is that if you're accidentally even committing the public key or password or something, it executes this pre commit hook that you have given to git before the commit and it says, okay. So I cannot do this because your pre commit hook is preventing it. And then you have post commit hook. So post commit hook is quite opposite. If you want to execute any action after your commit, then that's called post commit hook.

Question 7 what is web hook?

**It allows you to set up notifications and actions in response to events that occur within a GitHub repository.** Webhooks are a way for GitHub to communicate with external services when certain event occurs.**Most common use of webhooks is Triggering CI/CD pipelines when code is pushed to the repository.**

So similar to pre and post commit hooks, there is also another thing called as web hook. So web hook is basically part of your GitHub or Bitbucket or any of your Git implementations. What happens is that web hook is something that let's assume you want to trigger a pipeline or you want to run a Python program after your Git commit is done or you want GitHub to perform an action. So simply you can confer a web hook. That means in GitHub, you can create something called hook using this web hook and GitHub does the action that you told it. So if you simply go to GitHub web hooks, there are a lot of actions that can be taken place, like you can perform web hooks after issues, you can perform web hooks after pull requests, you can perform web hooks after issue commands, everything for every action in GitHub, you can confer a web hook.

Question 8 how to pull and push changes to Git.

If we want to merge latest changes to local repository from remote use git pull & if you want to merge local changes to remote repository use push.

Question 9 what is Git stash and talk about its use cases. Okay.

**Git stash temporarily saves changes that you have made in your working directory.When you're in the middle of working on one branch and need to switch to another you can use git stash to save your changes before the switching.**

So basically, **Git stash temporarily saves changes that you have made in your working directory.** Let's assume you want to, you're working on a code change and you know, immediately there was there is a very serious bug and you want to move from this bug to the other bug. Okay. And you cannot commit this changes because they are not ready. So what you can simply do is that you can stash this changes, which means you are temporarily saving them in your Git and you move to a different bug. You make those code changes and once you're satisfied with them, you push them and then you come back and say, Git stash pop. So whatever the changes that you have stashed, they will be like, you know, you can git those changes back. So that's about Git stash.

Question 10. What is the difference between Git clone and Git fork?

git clone is used to create a local copy of a remote repository on your machine, while git fork used for creating a copy of a Git repository on a Git.

Okay. So the difference between Git clone and folk is that **Git clone is basically you're creating a copy of whatever remote repository that is, whereas Git folk is that you are gitting the Git repository into your namespace.** So let's assume that there is a GitHub repository on my name. Okay. So I've created a Git GitHub repository and one of my colleagues wants to have the repository in his namespace. That means in his, let's assume my GitHub ID is Abhishek Romala and his ID is XYZ. So he can git the same Git repository from Abhishek Romala to XYZ. So that's not cloning, but he's actually gitting the whole thing into his namespace. And he makes his own changes and someone else, if they like his changes, he can clone his fork. So basically you're creating an entire copy from someone's namespace that is Git folk, whereas in Git clone, you can simply git changes from a remote repository to your laptop. That's it. So that's the difference between Git clone and Git folk.

Question 12 what is cherry pick in Git?

**It allows you to select and apply a specific commit from one branch to another branch. First we execute git log command to copy specific commit id from one branch and then we switch to another branch & execute git cherry-pick command with commit id which apply that commit to current branch**

Okay. So cherry picking in Git is nothing but gitting, let's assume that you have a commit in XYZ branch and you want to git the same commit as it is from XYZ branch to a branch called feature. Okay. So if you know that there are no merge conflicts, okay, instead of manually copying the changes or using Git merge or something, you can simply do Git cherry pick so that the whole commit, if there are no merge conflicts can be taken directly and you know, the commit history will be the same, the person who is committing everything would remain the same. So you are simply cherry picking it from one branch to the other branch.

Question number 11 is how to amend a commit in Git.

**To amend a commit in Git, you can use the git commit --amend command. This command allows you to make changes into the most recent commit or the last commit you made and possibly rewrite its commit message.**

**If you have already pushed the previous commit to a remote repository, you will need to force-push the amended commit.** **This overwrites the previous commit.**

So amending is nothing but you want to add something to the existing commit. Okay. So you can then something wrong with your commit and you have pushed the commit. Okay. And if you want to fix the broken commit, what you can simply do is that you can use Git commit hyphen, hyphen, amending and that way actually amending or you know, it would fix your broken change. So that's about Git commit amend. And the final

Question 12 How do you resolve a merge conflict in Git?

**When two or more developers make changes to the same lines of code or to same file simultaneously from two different branches it creates merge conflict. To resolve this conflict first we need to open that conflicted file then there are conflict markers which shows changes from current branch & changes from incoming branch. We need to decide which changes to keep & remove. Then we do git add, commit & merge.**

**If you used git merge to initiate the merge, you can complete it by using git merge --continue. If you were using git pull, the pull process will automatically continue once you commit the resolved changes.**

So this is a very practical scenario. If you have a merge conflict, how do you fix it basically depends on the merge conflict itself. So how did you run into this merge conflict? Who is the developer that has made the change? Let's assume that you are making a logic change to the calculator. Let's assume you are improving the addition function and then you realize that it is actually creating a conflict in conflict with the existing code. So what you can do is that you can sit with the other developer, try to understand why he has made the change and you can come to a consensus that which change is correct and which change is wrong. So that is a simple way of resolving a merge conflict. So it can be resolved by collaboration, discuss which which change is required, if his change is required or your change is required. And you can you know, with consensus you can come to a common point and you can simply use the fix the merge conflict. So that's the easiest way of fixing the merge conflict. And once you fix the merge conflict, you have different options. It depends on how you are merging it, whether you are using git merge or git rebase. So if you run into a conflict, what you do is you can do git merge-f&f and abort to completely remove your changes. Or if you know that your changes are fixed, you sat with the developer and you understood what has to be changed. So simply add the changes back and do git merge-f&re-store or git rebase-f&re-store. So this is the thing that you can do. Yeah, I think this is all that we have for today. And if you have any further questions on this, please let me know. We can discuss in the comment section or you can also ping me on LinkedIn. Thank you.

**DOCKER : cp dsrapp**

Question 2 What is Docker? So **Docker is an open source containerization platform which is used for developing, shipping, and running applications into the containers..** **Containerization is a method for packaging, distributing, and managing applications and their dependencies in an isolated environment called a container. And Containers are instances of Docker images, which are lightweight, stand-alone, and executable packages that include the application code, runtime, system tools, libraries, and settings needed to run the application.** **Each container runs as an isolated process and does not interfere with other containers** .**Containers share the host's kernel but have their own user space, file system, and networking.** So basically, it provide container runtime there are other containerization platforms which are alternative to Docker like you have Podman and the other things, but Docker is one of the first containerization platforms and it's one of the most popular ones because it's backed up by a great community and it has been for a long time now. Now, because we talked about Docker containers,

Question 2 what is a Docker container and how is it different from a virtual machine?

So container can be considered as a standard executable component, which is a package of your application source code. The dependencies that are required to run your application source code and an operating system or the OS libraries that are required. So we in terms of a virtual machine, you use the complete operating system, whereas with Docker container, you only use the libraries that are required. So it follows a concept called shared libraries. Okay. So because Docker uses shared libraries, so the containers are basically light in wait. They don't have the complete operating system, whereas in terms of virtual machine, you have the complete operating system, you install all the dependencies on your virtual machine and then you install your application application server and the other components that are required. So the main reason why a container is light in wait because of the shared libraries concept that it uses. If you want to learn more about Docker containers, you can go back to my DevOps playlist and watch the video on Docker containers and the concept of Docker, how it works. I also try to put the link in the description.

Question 3, what is a Docker file?

**It is configuration file used in Docker to define the steps and instructions necessary to build a Docker image.**

**A Docker image is a standalone, executable package that includes an application and all its dependencies.Dockerfile contains base image & instructions like run, copy, add, workdir, cmd & entrypoint, expose.**

So if you put in a very simple words, you know, users would require to create a Docker file with set of instructions or commands that defines a Docker image. Like, you know, let's assume that you want to create build a Docker image and then you want to run a Docker container. So for which you have to write a Docker file and inside this Docker file, you have to you have to define like, you know, what is the base operating system that you want to use? And then on top of it, what are the dependencies that you want to install? And then you have to provide like, you know, if you want to copy any files and all these instructions, you write in a file called as Docker file and then you build this Docker file to pro, I mean, to git a Docker image and using this Docker image, you use a Docker CLI command called Docker run. Once you run this Docker run on the image that you created, it generates a container. So basically the whole thing happens in three steps. Firstly, a user writes a Docker file with set of instructions and then you use a Docker CLI command called as Docker build. And this Docker build builds an image. And this image has to be run to create a Docker container. So these are the three steps.

So the next question is also answered using the question for three, I have explained what is a Docker image as well. But if you want the definition, Docker image acts as a set of instructions to build the Docker container. Like I told you, it happens in three steps, step one Docker file, step two Docker image step three is Docker container.

Question 5 what are the different Docker components?

**There are 3 core components of docker docker engine, docker images & docker containers.Docker engine contains docker daemon which is a background service that manages containers on a host system & we can interact with docker daemon using docker cli which is command line tool for docker.Docker image is like a template that contains an application and its dependencies.We create containers from docker images. We can store these images to the docker hub which is cloud based a registry.Docker Containers are instances of Docker images, which are lightweight, stand-alone, and executable packages that include the application code, runtime, system tools, libraries, and settings needed to run the application.**

Okay. So you know, in Docker, you have basically three components that is Docker client. And then you have Docker host and then you have a Docker registry. So Docker client is nothing but your Docker CLI, where you provide all the Docker commands like, you know, you say Docker build Docker run, all the commands that is Docker client. And then you have a Docker host, which has actually has the demon. So for example, I want to execute all this Docker commands. So there has to be someone who who has to listen to this commands and respond, right? So that is nothing but a Docker demon that is present inside the Docker host. And then you have Docker registry, which is basically used to host these images. So let's assume I created a Docker image. Now, I want to share it with other people. So to share it with other people, there should be a common place that is called as a Docker registry. So for example, you have Docker hub, you have create.io. These are all the Docker registries, or you can call them as container registries.

What is a Docker registry? And which registry are you using in the current organization? The same question, right? So once you have Docker images, let's assume that your develops engineer or anybody has built Docker images. So now there has to be a common place where this images has to be hosted, shared within the organization or to your customers. So you can have internal Docker registries or you can have an external Docker registry. It depends on your own. And you can also use publicly hosted Docker registries like Docker hub or create.io. Or if your organization has some secured images, you can create an internal registry and you can store your images there. So depending on what your organization is doing, you can tell the answer accordingly.

Question 7 what is the difference between Docker copy and Docker add?

COPY instruction is used for copying files and directories from the host machine into the Docker image and ADD is used to download specific file from URL into the docker image.

So both of them are used to copy the files inside your Docker image. But if you're using Docker add, it also has the capability of pulling the image, sorry, pulling the files from a URL. Let's assume that you have a doc, you have a file that is stored in GitHub and you want to pull it into your Docker image. So instead of copying that file from the GitHub to your local, you can and then copying it to Docker, you can simply write in your Docker, you know, in your Docker file, you can use the Docker add command. And this ad, once the URL is provided, it can it will simply download the file and it will place it in the Docker image. Whereas copy does not have this capability copy can only copy the files from your host to your Docker image or your Docker container.

Question 8 what is the difference between CMD and entry point in Docker?

**Both the CMD and ENTRYPOINT instructions are used to specify the command that should be executed when a container is run. When you provide arguments while starting a container with docker run, the CMD instruction is overridden by the user-provided command and arguments. In contrast, the ENTRYPOINT instruction remains the primary command and is not overridden; instead, the user-provided arguments are appended to it.**

So firstly, you need to understand what these things do. So CMD or entry point are basically used to run the command. Let's assume that once you are once you are spinning up your Docker container. Okay. So you want some command to be executed as soon as your container is spun. Let's assume that you want to say Hello world or you want to run a sample script or you want to run a script that has to be executed as soon as your container or a process is started in the container. So this kind of things you can put it either in the command CMD or entry point. So now what is the difference if both of them are doing the same thing? What is the difference between CMD and entry point? So let's assume you are using CMD and you have some shell script that you want to execute when the once the container is created. Sorry, once the container is run. The main disadvantage with the CLI is that if a user is passing some CLI arguments using Docker run, let's assume he starts a container using Docker run followed by the container name followed by some commands he provides manually in the Docker CLI. So these arguments overwrites the arguments that are provided in the CMD instructions. Whereas if you are using entry point, okay, and you're using entry point in the shell mode like in the entry point you have provided a shell command. Okay. So even though user tries to pass some arguments through Docker run, they won't git over written. Okay. So depending upon your use case, you can either choose CMD or entry point.

Question 9 How can you transfer a file from one container to another container? Okay. So let's assume that you have two containers. One is a login container and one is, you know, transaction or the payments container. And as soon as the user user logs in, you want to pass some information from the login container to the payments or the transactions container. Okay. So now how these containers communicate, you want to pass one file from one container to the other container. So for which you have two options, there are other options as well, but popular ones are either you can use Docker CP. So Docker CP, **By using Docker CP, you have to do manual copy from one container to other container in two steps. One step is you copy the file from the login container to your host and from your host, you again, copy it to the payments container.** **So this is a manual process.** **Another way is by creating shared volume. So that both of container can share files to each other on common volume or mount point. You can mount the same volume into both containers, allowing them to read and write data to a common location.**If you want to automate this process, what you have to do is inside your container, you have to, you know, inside your application and the container, you have to ensure that these containers would read this file from a shared volume. So for both of these containers, you would create a volume that is shared between both of them. And whenever application tries to application login tries to pass the information to payments. What it does is login container creates that file in the shared volume and payments container would read this file from the shared volume that is available. So both of them communicate using the shared volume. So these are the two ways either you can do manually using Docker CP or you can automate by creating a shared volume so that information can be passed through that shared volume.

Question 10 what is the difference between Docker file Docker compose and a Docker swarm?

**From dockerfile we can create single container where using docker compose we can create multiple containers. Docker swarm is container orchestration platform which allows you to create cluster of containers. It provides features like load balancing, service discovery, rolling updates, and high availability for containers in a cluster.**

So if you're writing a Docker file and if you're running a Docker file, you can create a single container. Whereas using Docker compose, you can actually create one or more than one containers, but Docker spam altogither is a different one. It's a container orchestration platform, which allows a lot of additional capabilities like, you know, container to container communication or, you know, it's a basically enterprise support for running containers.

Question number 11, what is a multi stage building Docker?

**It allow you to create more efficient and smaller container images by using multiple build stages within a single Dockerfile.It help you reduce the size of the final image.For e.g we create artifacts in first stage of dockerfile using maven & in the second stage we use tomcat image, copy that artifact from first stage & deploy it to this stage.Finally if we launch container from this image it will automatically start the Tomcat server and serve the Java web application.**

Okay. So if this is a new concept that is added to the Docker probably a year ago, where previously, you know, if you want to build a Docker file, so you basically write the Docker file instructions and everything happens in a single stage. But now Docker has come with a new concept that you can create multi stage builds. Like let's assume your application has a UI component, your application has a backend component. And finally, you bundle UI and backend component into a single image and you ship it to your customer. So what you can do is you can create first stage as UI component, a UI component build and second stage as the backend component build. And you can pass the artifacts from first stage and second stage to the final stage. So what happens with this is that in your first stage, you must be using some dependencies like NPM or Jan or anything to build your UI component. And then the second stage, you are using Java or something to build your backend. Okay. So in the third stage, you can simply omit all the dependencies of the UI component and simply copy the artifact from the UI component and copy the artifact from the backend component and simply take those two components in the final, you know, final stage. So your container will become a very light in wait. So this is the concept of multi stage builds.

Question 12 what are distro less images in Docker ? This is also new component. So distrolls images are basically very light in wait Docker images. So you might be asking me like, you know, Docker images are already very light in wait. So why do you want the concept of distrolls images? So let's take a scenario. You have a Docker image, which you run and you know, it generates a Docker container. And this Docker container you have deployed in your production and it's running well. Now let's assume the size of this container is some 600 MB or 500 MB. When you go to the Docker container and see that it has package management libraries, like if it is a centoise, let's assume it's a Ubuntu or centoise container, it has M or apt or some other things, right? But in general, those things are not used or they are not useful because you're already running your content in production. You don't want to use anything like M to upgrade any libraries or something. So what is the basic use of those package management libraries? So if you feel that these things are not useful, you can simply go to the distrolls images in in case of distrolls images, they are basically more light in wait than the actual Docker images because they even eliminate this kind of dependencies from your Docker operating system, whatever the operating system it is. What it does is distrolls images, they simply have your application, your application dependencies and very, very, very minimal operating system libraries. They don't even have the package managers. So this is a concept of distrolls images. So both question number 11 or 12 are related to, you know, lightweight containers. So these are new, both of them are new concepts like introduced one or two years ago in Docker. Yeah, so these are the questions that we have for today. And please follow my channel to, you know, learn more about DevOps interview questions and a lot of new interesting things that are happening in DevOps. So I'll see you in the next video guys. Thank you so much. Take care. Bye.

**KUBERNETES**

Hello everyone. Welcome back to my channel. In this video, we'll take a look at some of the basic Kubernetes interview questions and their answers. So Kubernetes is a very vast topic. I made a lot of videos on Kubernetes on my channel. So firstly, I'll recommend you to go and watch those videos so that you git a context of what is a pod, what is deployment service ingress. We also did a video on the advanced deployment strategies. We did a lot of videos on Kubernetes. I'll try to put all of them in the description, please go and watch those videos so that you git a very clear idea on the components of Kubernetes and a lot of other things that I've discussed on Kubernetes. So this is part one series of interview questions. We will do more videos on Kubernetes interview questions because it would take a lot of time if we do all of them togither. So let's quickly jump and see what we have for today. So these are the basic questions and these are the most commonly asked questions in interviews. You should definitely have an idea and the answers to these questions should be on tip of your tongue because if you try to fumble on these questions, it might, I mean, interview might feel that you do not have proper experience on Kubernetes. So even though you know the answers to these questions, just try to revise it one more time so that you will be able to answer these questions during interview promptly. Okay. Okay,

so question number one that we have today is what is the difference between Docker and Kubernetes? Right? Both of them are very popular technologies in DevOps. So basically Docker is for building and running single container, whereas Kubernetes is a system. So you can consider Kubernetes as a group of virtual machines that are combined togither to form a cluster. So that's the main difference between Docker and Kubernetes. So one is a container platform and the other is container orchestration platform.

So question number two, what are the main components of Kubernetes architectures? Okay. So this will tell the interviewer that do you have the knowledge of Kubernetes or not because there are a lot of components in Kubernetes and these are the components that interact with each other to actually form your whole communication between the containers. So in Kubernetes, you have two major things. One is the control plane or the master and the other one is the node. So there are components which run on the control plane and there are components which run on the node. So firstly, the components that run on the control plane are Cube API server, which is responsible for your API. I mean, it's a API server so that itself tells what it does. It is responsible for everything. The actions that you perform using Cube CTL or, you know, the communications that happen, everything goes to the Cube API server first. And then you have ETCD, which is the Kubernetes, I mean, where Kubernetes stores its objects and then you have Cube Scheduler, which takes care of the scheduling of pods on Kubernetes and then you have controller manager. So controller manager is responsible for the controllers in Kubernetes and then you have the cloud controller manager. So these are the components that run on the control plane. And after that, you have components that run on the node. So these are your Cubelet, which is responsible for running containers in a pod. And then you have Cube proxy, which I'll tell you going ahead. And then you have container runtime, container runtime, like, you know, Kubernetes has supports multiple container run times like Docker, continuity or Creo. So they are different container run times, which are responsible for running the containers again. Okay.

Question number three. What are the main differences between Docker swam and Kubernetes? Okay. So again, both of them are container orchestration environments. Both of them are designed to solve the same purpose. One is designed by Docker. And the other one is by Google, which is Kubernetes. So what basically, I mean, in which aspect basically the differ is if you consider Kubernetes, it has advantage over Docker swam, like, you know, the scalability of the Kubernetes or you can configure a lot of monitoring with Kubernetes. And Kubernetes has something like custom resource definitions with which you can do load balancing, advance load balancing with, with the applications that you have deployed in Kubernetes. Kubernetes also has support for integrating with like the API gateways and you can integrate security web application firewall. So Kubernetes is overall, you know, a complete production grade system. I don't say Docker swam is not, but Kubernetes is even backed up by a great community. CNCF is there and a lot of other companies are contributing to Kubernetes. So which makes Kubernetes overall a better system than Docker swam.

Question number four, what is the difference between Docker container and a Kubernetes pod? Okay. So this is one of the basic questions that interview will ask you to understand if you have knowledge on Kubernetes or you are still working on Docker. So the basic difference is that Docker Kubernetes pod is a group of like, you know, you can call it as one or a group of containers. Okay. It doesn't have to be group of containers all the time. It can be a one single container or group of containers. And the basic advantage is that all the containers that are deployed in a pod, they can have the shared storage, they can also, you know, share the network resources. And it also has a specifications of how these containers have to be done. Like for example, if you're running a Docker container, you pass a lot of parameters like what port it has to bind to the host and what is the volume that it has to use other things. Right. So the same way you put these things in the Kubernetes pod specification, which is a yaml file. So that's the basic difference between a container and a Kubernetes pod.

Question number five, what is the difference between Kubernetes deployment stateful set and the D-Monset? Okay. So these are the three different types of deployments that you can do in Kubernetes, either through a deployment stateful set or a D-Monset. So stateful set is something which you use for maintaining stateful applications. Like, you know, if your application has to maintain a state, if your application is going down and the next time it has to come up, it has to come up with the same data or, you know, it has to use a specific volume. That's persistent volume. So in such cases, you use a stateful set and then D-Monset is basically a controller, okay, which typically runs on all the nodes of your Kubernetes cluster. So if you want your, let's assume you have a Kubernetes deployment and you say replicas as five. So you're not sure where these replicas will fall. It depends on the Kubernetes or if you have configured any rules, but by default, it will go according to the cubes scheduler wherever, wherever it has to deploy the pods, everything can go to a same node or two different nodes. But whereas the D-Monset, the pods are deployed on each and every node of your Kubernetes cluster. Let's say you want an application that has to monitor the nodes of your Kubernetes cluster or it has to check health of each and every node. Your application is performing that thing. So in such case, you need to have a D-Monset. If you're aware of Linux, there are a lot of D-Mons that are running on your Linux. So D-Monset applications are also similar.

What is a namespace in Kubernetes? Okay. So namespace in Kubernetes is basically a logical grouping of networking resources. Let's assume you have a lot of teams in your organization and all of you want to use a Kubernetes cluster. So each of your project gits a namespace. You can consider it, it is used to achieve multi-tenancy or you can consider it is used to achieve multiple teams working in a same Kubernetes cluster. So each of them can be given a Kubernetes namespace or you have a dev namespace, you have staging namespace and you have production namespace for logical isolation of your resources.

Question number seven. What is the role of QProxy? Okay. So QProxy basically takes care of networking in your Kubernetes. Let's assume that you want to, how do I put this? Do you want to talk to an application that is deployed in Kubernetes? QProxy is the one that basically takes care of this thing. So it has, it can be configured in multiple ways. The most common thing is IP tables. So in Kubernetes whenever you deploy a container or whenever you deploy a pod, whatever it is. So IP tables gits updated and QProxy basically takes care of updating these IP tables and whenever someone tries to access your pod from external or within the cluster, they refer to the IP tables, the rules that are written in the IP tables and they take care of the communication. So QProxy is the one basically that is a key component for handling the communication in Kubernetes.

Question number eight. What are the different types of services within Kubernetes? So in a Kubernetes, a service can be implemented in multiple ways. So the commonly used services in Kubernetes are cluster IP mode, you have node port mode and you have the load balancer type mode. So these are the most three commonly used services types in Kubernetes.

And followed up by what are the differences between node port and the load balancer service type. So node port service type, whenever you're deploying a pod and you're creating a service for it or you're deploying a deployment and you're creating a service for it. So what happens is if you create this service as node port service type, your application can be accessed using using the node IP and the port that is allocated by your Kubernetes. So your node, let's assume your node IP is 170.16.3.4. Kubernetes associates, I mean the service gits associated with the node port. So it would be 170.16.3.4 colon, let's say 8,000, I mean, obviously node port should be in the range of 32,000. So it should be 170.16.3.4 colon, 32,174, something like that. So this would be your node port type service. So what is the advantage with it? The advantage is that you are able to access it using the node IP address. So whoever can access this node can be able to access the application. They don't have to have access to the internals of your Kubernetes cluster or that application doesn't have to sit inside your Kubernetes cluster. Even the application that is outside your cluster and has access to your node IP can access the application. What if someone external to your company or someone external to your firewall wants to access this specific thing? I mean, they don't have access to your node or they don't have access to your Kubernetes cluster, but they still want to access your application. That would be any external websites, right? So you are trying to access many websites every day. And what you are typically doing is you are trying to access them using a external IP address. So for that you create a service of type load balancer service. So if you are creating a load balancer service, that should be an underlying, this has to be only deployed on a cloud platform. Or if you are doing it on on-premise, you have to use metal LB. But let's restrict it to let's restrict to cloud provider. If you are deploying it on AWS, what happens is that the AWS cloud load balancer service implementation would create a load balancer IP address or a static IP address for you. And this static IP address can be used to talk to the application that you have deployed in Kubernetes. So it only has to be deployed on cloud platform. Or if you are deploying it on on-premise, you have to definitely install metal LB. It's a project that is being developed. It's not completely, I cannot say that the project is completely ready for production use case, but you can still go and search for metal LB. That's for on-premise clusters.

Question number 10, what is ingress in Kubernetes? Okay, so ingress is also used to achieve the same purpose that load balancer service is doing. And ingress is a resource that has a lot of other advantages. Like it can define you can define the traffic rules for your Kubernetes cluster or your Kubernetes application. Let's say you have a domain called example.com and your example.com has to be only accessed by certain IP address range or you are you want to define rules for your applications. Let's say if the request is coming to slash foo, it has to go to log in application. If the request is going to slash bar, then it has to go to log out application. So ingress is basically used to define rules traffic rules for your application. So there is a lot that you can do with ingress and if you just create an ingress, it wouldn't work because you also have to create an ingress controller. So I made a complete video on ingress ingress controllers, how they work and how you can create a lot of deployment strategies using ingress. So please watch that videos. I'll put it in the description.

So the last question that we have for this basic interview questions is what is the role of Kublitt. Okay. So Kublitt is basically responsible for running containers in your pod. So what Kublitt does is whenever you create a pod in Kubernetes, so Kublitt basically talks to the container runtime. There can be different container runtimes like container D or Docker or any container runtime and Kublitt talks to this container runtime and make sure that your container is running on the defined pod or your container is running on your Kubernetes cluster. So this is responsible for Kublitt. In short, you can say Kublitt is responsible for running pods or running containers in the pod. So this is the basic interview questions and if you have any questions, please post in the comments. I'll make a detailed video on if your question cannot be addressed in a comment. So that's all we have for today guys. Thank you so much.

**ANSIBLE:**

What is configuration management?

Configuration management is process of defining, automating, and maintaining the desired state of your infrastructure. Ansible is open source automation tool which is used to execute tasks remotely across multiple targit hosts. We can remotely install & configure web servers like tomcat nginx, database servers like mysql mongodb, CICD tool like jenkins. Also we can launch instances, docker containers, kubernetes clusters. We just need to write playbook for each of these task.

Question number two, do you think Ansible is better than other configuration management tools? If yes, why?

Ansible is agentless. You don't need to install any agent on targit machine or on ansible controller. Ansible uses existing remote connection like SSH for linux, winrm for windows & API modules for cloud connections.For connection to database it uses python libraries.

Ansible is written in Python which is most popular language & it has lots of modules that ansible uses.Also we write ansible playbooks in YAML which is widely used & easy to understand.

Can you write an Ansible playbook to install HTTPD service ? And git it running ?

How ansible helped your organization?

In one of our project we have automated the server provisioning process using Ansible which reduced the time to set up new servers. Before that we were using shell script for provisioning servers which was time consuming.It also ensures that new servers are configured consistently and securely. Also we have scheduled & automated system updates & patching through ansible.We have used rolling updates with zero downtime to servers which keeps our application running.Also ansible works for on-premise as well as cloud server which makes it easy to manage everything in one go.

Question number five what is ansible dynamic inventory.

Dynamic inventories are particularly useful in scenarios where the inventory of hosts frequently changes, such as when dealing with cloud instances and containers.We have used dynamic inventory for AWS instances.For this we need to install boto3 & download ec2.py from the Ansible GitHub repository.Create ec2.ini with AWS credentials, region, VPC ID, and any group name

Dynamic inventory is used when we have virtual machines are in cloud & specially when those virtual machines are under auto scaling. Coz this machines are dynamically launched & terminated and their IP address keeps changing always. So to keep track of IP addresses we use dynamic inventory. Ansible uses python scripts to dynamic inventory.

Download ec2 python script & ec2.ini to set configuration settings for dynamic inventory

Ansible will support something called as dynamic inventory. So the concept of dynamic inventory is and so will Ansible keeps looking at your AWS account for newly added virtual machines and ansible will auto configure those virtual machines using the concept of dynamic inventory.

What is Ansible tower and have you used it ? If yes, why did you use it ?

So you don't have to lie in this case if you have not used answer bill tower you can upfront tell that you have not used answer bill tower because not everyone works on the enterprise model. So Ansible tower is basically enterprise model or a version of Ansible where it provides you a you know a fancy graphical UI. I mean, fancy graphical user interface and it also extends some additional and simple capabilities like you know it has a very good user management and you can configure it with your I am if you're using and so well on AWS so a lot of fancy stuff that you can do with and so well tower it's an enterprise model it's backed up. So you git a you git support if there is any issue with answer bill. So these are the things that answer bill tower does but if you're not used answer bill tower you can simply say that we are working on answer bill CLI we are only using the open source version of answer bill that is that doesn't make any difference because end of the day you have knowledge of answer bill. So next question is how do you manage our back on answer bill towers so this is again a question on answer bill tower. So, basically you will be tested on in this question you are asked about do you know about role based access control. So let's assume that you are using answer bill tower in your enterprise and you want to segregate access for a specific group or you know you want to distinguish users based on the access let's assume you have developers you have testers you have admins so how do you distinguish these people with their access on the answer bill tower. So for which answer will support something called as role based access control. So using role based access control you can say that you know this specific group call testers or this specific group called as read only has only read only access on the answer bill tower they can only view the playbooks or they can only you view what is happening on the answer bill tower but they cannot execute anything or they cannot write or make any changes to the answer bill tower. So role based access control is basically used for managing access to the answer bill tower and you can also configure answer bill with some external identity providers. Let's assume that your organization is using some SSO like LDAP or key clock or any external identity providers you can also integrate them using answer bill tower or if you are AWS you can also integrate it with your IAM. So these are the different things that you can do with respect to R. And the next question is what is answer bill galaxy command and why is it used so answer bill galaxy command is basically used to bootstrap let's assume that you want to start writing an answer will playbook and you want to create the whole structure. So instead of creating everything by your own answer bill galaxy command can be used to bootstrap the whole structure it creates the folders required files and everything and you can use whatever like you know instead of creating a whole structure by yourself you can directly use it. So this is about answer bill galaxy command. Next question is can you explain me the structure of answer bill playbook using roles. So even I am using I'm seeing these questions after a long time so I'm taking some time to read them. Can you explain me the structure of answer bill playbook using roles. So again it is somewhat related to the previous question that is answer will galaxy command which creates the structure for your answer will playbook and it creates roles it creates all your default handlers everything so in the same way here. The question is about what is the structure of answer will playbook what all falls into the answer bill structure. So so answer will has a standard structure. So if you're creating them you have the default folder you have the handlers you have templates you have metadata you have a task. So all of these things are present in the answer will default structure. So you can say that we use answer will galaxy command to create the structure and these are the different folders that are created by answer will galaxy command and you can also create them manually. You can also describe what each and every folder does like why do you use handlers why do you use defoils or you know why do you templates so you can talk about each of them in detail. And the other thing is you can also talk about the folders like the group variables the folders and what is your structure of a sample answer will playbook that you have written. And the next question is what are what are handlers in answer book and civil and why are they used. Handlers are basically similar to the normal task in answer bill, but they only run when a task contents and notify directive. So let's assume that you have a basic task in your answer bill for example for creation of an engine. You are installing HPD or engine. So the next step is you basically start. So in some cases you only want to start on a conditional basis. So what you can do is you can use this directive called notify and this notify would call the task that you created in the handler and that gits called. So it's basically a way of calling only when it is required. So you can say it as conditional task or you can also call them as tasks that only run during a special notification. So this is about handlers. You can also give an example that I have stated like you know you you have a task that installs in genix and you can use a handler to start engine. Or you know you have a failure in one of your tasks. Then you can use notify to say that okay send a notification using handlers or whatever example that you want to give. So the next question is I would like to run a specific set of tasks only on Windows VMS and not on Linux VMS is it possible. Yeah. So again, this is a pretty common question that you want to run a specific set of tasks on a specific operating system and not on the other operating system. So what you can do is that you can find out the operating system that this specific task is running on. Let's assume that you have 10 tasks and you want to run five on Windows and you know five on Linux VMS. I mean if the operating system is Linux VM. So you can use tags and support something called as tags or you can also use the NVRON and you can find out the environment variable and you can use conditional basis running of tasks in answerable to achieve this use case. Next does answer will support parallel execution. So first of all you need to understand this question. What do you mean by parallel execution? So you can say the interviewer that answer will does parallel execution like you know if there are a fight as what answer will does is it execute task number one on all the configured servers. Let's assume I will wants to run on 10 virtual machines. So it takes task one and it runs parallely task one on all of these virtual machines. And then it runs task by task in a serial way. That means once task one is done execution parallely on all of the host, then it moves to task number two. If you want to run even the task in a parallel way, then you need to you know have multiple instances of answerable. So this is a better way of answering the question. So firstly understand what is parallel execution and then you can easily answer. And if the interviewer is asking about different strategies that are available, then you might talk about the serial and the linear the other strategies that are present in answerable. I'll also try to put a link in the description for this. The other strategies is this that answerable supports. So the question number 13 is what is the protocol that answerable uses to connect to windows VMS. So this is a straightforward question. You can say answerable uses been Rm to connect to windows and it uses SSH to connect to Linux. And the next question is, can you place them in the order of precedence. So you have. Okay, so you have multiple places to store variables in answerable. And here they're asking about group where's. Rule where's and extra. So you have to put them in the order of precedence. So answerable supports a lot of places to store the variables. So you have to be very sure about, you know, the precedence of the variables here even not for the interview, but even when you're trying to execute answerable in real time, you need to know where to place your variable so that precedence acts accordingly. So for this question, I think the order is correct. Like you have group where's rule words and extra words. That is the order of precedence. And I will also put the whole order of precedence of the other variables that you have in answerable. There are a lot of things like you have a wire files, you have cash facts and you have. Playbook host to us. I mean, in the playbook, you can also put variable. So there are many places. So I'll try to put the entire variable precedence in the description so that you can take a look at it. How do you handle secrets in answerable? Okay. So you can configure. So, I mean, hashy cobe supports. I mean hashy cobe has something well less world. So you can store the answerable secrets in the world. That is one of the better solutions that I know if you are aware of any other solutions, you can also post it in the comment section. I'll be more than happy to learn. Can we use answerable for IAC infrastructure as code. Okay. If yes, can you compare it with any other IAC tools like terraforma. Okay. Yeah. Answerable also supports infrastructure as code. I will not say it's an infrastructure as code tool because predominantly answerable is a configuration management tool, but there is a cross of parts. So answerable also achieve. I mean also will can also achieve some of the tasks that terraform can do. But I would say answerable is a configuration management tool. It might achieve some of the things like it can create an easy to instance on your AWS or something like that. But mostly it's used for configuration management. Can you talk about a answerable playbook that you have wrote and how it helped your company. Okay. So this is a scenario based question one more time. And this, I think personally these are the questions that you'll evaluate your real knowledge on answerable because the other questions, you can even Google and try to find the answers. And then the other questions are the real important questions. So talk about a playbook that really makes value to your company or that really made a value to your company. So, for example, you can talk about installation and configuration of a or a kill database or a thing that actually has a lot of steps to do and you used and will effectively reduce the time for example to install and configure an Oracle database. It will take close to 45 minutes for each of the thing. And if you want to do it on 50 to 100 servers, then imagine the amount of time that you're spending in it. So for which you can use an example and you can say that the time has significantly reduced and the entire installation and the chance of failure has reduced a lot because an example is an automated creation and manually you might do some errors. You can say that the time has reduced from 45 minutes into 50 servers to 30 minutes your entire answerable playbook is taking to execute on all of the 50 servers or else you can also talk about some complex installations like, you know, you can say that you have an organization called Kubernetes, or organizations like Kubernetes, you know, installing Kubernetes is not an easy thing because if you're using things like cube adm, which is a basic tool for installing Kubernetes on a cluster. So there are a lot of steps to do you have to attach all the nodes and all so you can also automate the process using Ansible. So, if you don't have any idea like you know you haven't worked on Ansible in real time. So you can go to Ansible Galaxy. So you can see the platform where you have a lot of Ansible playbooks. So looking to that Ansible playbooks and pick one which you feel is a complicated one and, you know, you go through the code of the Ansible Playbook Ansible Galaxy has every most of the Ansible playbooks and the code is also there so pick one which suits you the better and you can project them in your interview. So this is a question that I have added because always people talk about advantages but you know you should also know the real downsides of the tool or that you're working in because every day if you work on a specific tool, you might be feeling that okay this tool can improve in this aspect So here, interviewer will try to judge you if you have really worked on Ansible. This is my personal favorite I keep asking such questions so the best way to answer this is if you're not aware of any drawbacks go to the Ansible GitHub page you can find the issues and you know with your real time experience you can also say that Ansible can you know currently Ansible does not provide option to increase the verbosity of the task like you know what is verbosity like the debug level for task basis so Ansible can improve in that aspect it can provide verbosity on that task basis this is one of the most liked issue on Ansible GitHub page and the other thing is Ansible windows support can be improved. We all know that Ansible supports windows but the support for Linux is much better than Windows so you can say that the Windows support can be improved and you can also say some of the things like you know Ansible can support multiple ideas there are not much ideas today that can do auto correction suggestion so Ansible can work on developing some plugins for those ideas like for example if you're a Python developer you know that there are a lot of ideas like Visual Studio or PyChum which can auto correct auto suggest and it can also you know provide some signed some kind of linting and indentation things with Ansible that's not available today there are some plugins in the VS code as well as the other ideas but they're not up to the mark so Ansible can look at contributing that because so that it will help a lot of Ansible newbies and people who work on Ansible on day to day basis. So this is about Ansible interview questions guys if you have any questions please comment so that I can answer to you on the comment section and you can also chat with me on LinkedIn I'll post my LinkedIn account description link in the description so yeah I'll see you next time. Thanks a lot. Bye