What is the Namespace? How many namespaces are there in Kubernetes?

A namespace is used to work with multiple teams or projects spread across. It is used to divide the cluster resources for multiple users.

Mention different kinds of Namespaces in Kubernetes.

The namespaces are of three kinds. They are:

Default: The default namespace that when the cluster comes out of the box with no other namespaces

Kube-system: The namespace for objects created by Kubernetes.

Kune-public: The namespace that can create automatically and is visible and readable publicly throughout the whole cluster. The public aspect of this namespace is only convenient and reserved for cluster usage.

what is the bind mount in docker?

In bind mount a file or directory is going to be created on the host machine and it is directly mounted into a container.

\* we need to use a separate Linux command to create a directory

\* we need to use seperate Linux commands for backup

Due to this Draw back mostly we can use docker volume.

what is a docker container?

container is a running state of your image. once you run your image we will get a container. you can start a container,stop a container,you can check logs of a container and you can go inside a docker container. you can attach volume to your container and awsellas you can attach network to your container.

1.How can you convert a public subnet to a private subnet?

\*Remove IGW & add NAT Gateway then Associate subnet in Private route table

2.Is it possible to reduce a ebs volume?

\* No it’s not possible, we can increase it but not reduce them

3. One of my s3 buckets is deleted but i need to restore is there any possible way?

\* If versioning was enabled, deleted objects are retained as previous versions, and we can restore them

\*If Cross-Region Replication was configured for your bucket, the deleted objects might still exist in the replicated bucket in another region

4.When I try to launch an ec2 instance i am getting the Service limit exceeding, how to fix the issue?

\*AWS offers service limit of 20 running instances per region, to fix the issue we need to contact AWS support to increase the limit based on the requirement

5.Differentiate Block storage and File storage?

\*In block storage, data is stored in blocks, whereas, in file storage, data is stored as files in a single piece.

\*we can access block storage data over any operating system, which is not the case with file storage.

\*Block storage is like a hard drive in the server, so it is more flexible than file storage.

\*File storage is ideal for data sharing within small teams.

\*Block storage is popular as a networking architecture used by enterprises for business-critical applications.

What’s the difference between a public hosted zone and a private hosted zone in Amazon Route 53?

Answer: Main differences between public and private hosted zones:

Visibility: Public hosted zones are visible to the entire internet, while private hosted zones are limited to specific VPCs or connected networks.

Access Control: Public hosted zones allow access and resolution by any client on the internet, while private hosted zones restrict access to resources within the specified VPC(s) or connected networks.

Resource Accessibility: Public hosted zones enable access to resources from anywhere on the internet, whereas private hosted zones are primarily used for resolving domain names for internal resources within a private network.

Define and explain the three basic types of cloud services and the AWS products that are built based on them?

The three basic types of cloud services are:

Computing

Storage

Networking

Here are some of the AWS products that are built based on the three cloud service types:

Computing - These include EC2, Elastic Beanstalk, Lambda, Auto-Scaling, and Lightsat.

Storage - These include S3, Glacier, Elastic Block Storage, Elastic File System.

Networking - These include VPC, Amazon CloudFront, Route53

What is Kubernetes?

Kubernetes is an open-source container orchestration system for deploying, scaling, and managing automated applications. It offers an excellent community and works with all cloud providers. Hence, it is a multi-container management solution.

What is the purpose of the expose and publish commands in Docker?

Expose

• Expose is an instruction used in Dockerfile.

• It is used to expose ports within a Docker network.

• It is a documenting instruction used at the time of building an image and running a container.

• Expose is the command used in Docker.

• Example: Expose 8080

Publish

• Publish is used in a Docker run command.

• It can be used outside a Docker environment.

• It is used to map a host port to a running container port.

• --publish or –p is the command used in Docker.

• Example: docker run –d –p 0.0.0.80:80

What are the cloud platforms that support Docker?

The following are the cloud platforms that Docker runs on:

• Amazon Web Services

• Microsoft Azure

• Google Cloud Platform

• Rackspace

What is a Dockerfile used for?

• A Dockerfile is used for creating Docker images using the build command.

• With a Docker image, any user can run the code to create Docker containers.

• Once a Docker image is built, it's uploaded in a Docker registry.

• From the Docker registry, users can get the Docker image and build new containers whenever they want.

Explain the architecture of Docker.

• Docker uses a client-server architecture.

• Docker Client is a service that runs a command. The command is translated using the REST API and is sent to the Docker Daemon (server).

• Docker Daemon accepts the request and interacts with the operating system to build Docker images and run Docker containers.

• A Docker image is a template of instructions, which is used to create containers.

• Docker container is an executable package of an application and its dependencies together.

• Docker registry is a service to host and distribute Docker images among users.

Explain the difference between a centralized and distributed version control system (VCS).

Centralized Version Control System

• All file versions are stored on a central server

• No developer has a copy of all files on a local system

• If the central server crashes, all data from the project will be lost

Distributed Control System

• Every developer has a copy of all versions of the code on their systems

• Enables team members to work offline and does not rely on a single location for backups

• There is no threat, even if the server crashes

What is the use of SSH?

SSH stands for Secure Shell and is an administrative protocol that lets users have access and control the remote servers over the Internet to work using the command line.

SSH is a secured encrypted version of the previously known Telnet which was unencrypted and not secure. This ensured that the communication with the remote server occurs in an encrypted form.

SSH also has a mechanism for remote user authentication, input communication between the client and the host, and sending the output back to the client.

Why DevOps has become famous?

These days, the market window of products has reduced drastically. We see new products almost daily. This provides a myriad of choices to consumers but it comes at a cost of heavy competition in the market. Organizations cant afford to release big features after a gap. They tend to ship off small features as releases to the customers at regular intervals so that their products don't get lost in this sea of competition.

Customer satisfaction is now a motto to the organizations which has also become the goal of any product for its success. In order to achieve this, companies need to do the below things:

• Frequent feature deployments

• Reduce time between bug fixes

• Reduce failure rate of releases

• Quicker recovery time in case of release failures.

• In order to achieve the above points and thereby achieving seamless product delivery, DevOps culture acts as a very useful tool. Due to these advantages, multi-national companies like Amazon and Google have adopted the methodology which has resulted in their increased performance.

Who is a DevOps engineer?

A DevOps engineer is a person who works with both software developers and the IT staff to ensure smooth code releases. They are generally developers who develop an interest in the deployment and operations domain or the system admins who develop a passion for coding to move towards the development side.

In short, a DevOps engineer is someone who has an understanding of SDLC (Software Development Lifecycle) and of automation tools for developing CI/CD pipelines.

DevOps Tools

DevOps is a methodology aimed at increased productivity and quality of product development. The main tools used in this methodology are:

• Version Control System tools. Eg.: git.

• Continuous Integration tools. Eg.: Jenkins

• Continuous Testing tools. Eg.: Selenium

• Configuration Management and Deployment tools. Eg.:Puppet, Chef, Ansible

• Continuous Monitoring tool. Eg.: Nagios

• Containerization tools. Eg.: Docker

What is DevOps?

DevOps stands for Development and Operations. It is a software engineering practice that focuses on bringing together the development team and the operations team for the purpose of automating the project at every stage. This approach helps in easily automating the project service management in order to aid the objectives at the operational level and improve the understanding of the technological stack used in the production environment.

This way of practice is related to agile methodology and it mainly focuses on team communication, resource management, and teamwork. The main benefits of following this structure are the speed of development and resolving the issues at the production environment level, the stability of applications, and the innovation involved behind it.

What is the git command that downloads any repository from GitHub to your computer?

The git command that downloads any repository from GitHub to your computer is git clone.

What is the role of AWS in DevOps?

AWS has the following role in DevOps:

• Flexible services - Provides ready-to-use, flexible services without the need to install or set up the software.

• Built for scale - You can manage a single instance or scale to thousands using AWS services.

• Automation - AWS lets you automate tasks and processes, giving you more time to innovate

• Secure - Using AWS Identity and Access Management (IAM), you can set user permissions and policies.

• Large partner ecosystem - AWS supports a large ecosystem of partners that integrate with and extend AWS services.

Mention some of the core benefits of DevOps.

The core benefits of DevOps are as follows:

Technical benefits

• Continuous software delivery

• Less complex problems to manage

• Early detection and faster correction of defects

Business benefits

• Faster delivery of features

• Stable operating environments

• Improved communication and collaboration between the teams

What are the different phases in DevOps?

The various phases of the DevOps lifecycle are as follows:

• Plan - Initially, there should be a plan for the type of application that needs to be developed. Getting a rough picture of the development process is always a good idea.

• Code - The application is coded as per the end-user requirements.

• Build - Build the application by integrating various codes formed in the previous steps.

• Test - This is the most crucial step of the application development. Test the application and rebuild, if necessary.

• Integrate - Multiple codes from different programmers are integrated into one.

• Deploy - Code is deployed into a cloud environment for further usage. It is ensured that any new changes do not affect the functioning of a high traffic website.

• Operate - Operations are performed on the code if required.

• Monitor - Application performance is monitored. Changes are made to meet the end-user requirements.

What is the concept behind sudo in Linux OS?

Sudo stands for ‘superuser do’ where the superuser is the root user of Linux. It is a program for Linux/Unix-based systems that gives provision to allow the users with superuser roles to use certain system commands at their root level.

What can be a preparatory approach for developing a project using the DevOps methodology?

The project can be developed by following the below stages by making use of DevOps:

• Stage 1: Plan: Plan and come up with a roadmap for implementation by performing a thorough assessment of the already existing processes to identify the areas of improvement and the blindspots.

• Stage 2: PoC: Come up with a proof of concept (PoC) just to get an idea regarding the complexities involved. Once the PoC is approved, the actual implementation work of the project would start.

• Stage 3: Follow DevOps: Once the project is ready for implementation, actual DevOps culture could be followed by making use of its phases like version control, continuous integration, continuous testing, continuous deployment, continuous delivery, and continuous monitoring.

The flow is as follows:

• Developer works on completing a functionality.

• Developer deploys his code to the test environment.

• Testers work on validating the feature. Business team can intervene and provide feedback too.

• Developers work on the test and business feedback in continuous collaboration manner.

• The code is then released to the production and validated again.

How is DevOps different than the Agile Methodology?

DevOps is a practice or a culture that allows the collaboration of the development team and the operations team to come together for successful product development. This involves making use of practices like continuous development, integration, testing, deployment, and monitoring of the SDLC cycle.

DevOps tries to reduce the gap between the developers and the operations team for the effective launch of the product.

Agile is nothing but a software development methodology that focuses on incremental, iterative, and rapid releases of software features by involving the customer by means of feedback. This methodology removes the gap between the requirement understanding of the clients and the developers.

What are the advantages of DevOps?

Technical benefits:

• Continuous software delivery

• Less complex problems to fix

• Faster resolution of problems

Business benefits:

• Faster delivery of features

• More stable operating environments

• More time available to add value (rather than fix/maintain)

How does AWS contribute to DevOps?

AWS stands for Amazon Web Services and it is a well known cloud provider. AWS helps DevOps by providing the below benefits:

• Flexible Resources: AWS provides ready-to-use flexible resources for usage.

• Scaling: Thousands of machines can be deployed on AWS by making use of unlimited storage and computation power.

• Automation: Lots of tasks can be automated by using various services provided by AWS.

• Security: AWS is secure and using its various security options provided under the hood of Identity and Access Management (IAM), the application deployments and builds can be secured.

What is Amazon Relational Database Service (Amazon RDS)?

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.

What is an AMI? How do we implement it?

AMI is Amazon Machine Image, which basically is a copy of your root file system. It feeds the information required to launch an instance.

We implement AMI by specifying an AMI whenever we want to launch an instance. Multiple instances can be launched from a single AMI with the same configuration.

In the case of launching instances with different configurations, we would need to launch different AMIs.

AMI includes one or more snapshots of your EBS volumes, in the case of instance-store backed AMIs, along with a template for the root volume of your instance (like an operating system, an application server, and applications).

It launches the permissions that decide which AWS accounts can use the AMI for launching instances. It also needs a block device mapping for specifying the volumes in order to attach them to the instances whenever they are launched.

What are the Cloud Storage Levels?

Files

Blocks

Datasets

Objects

Mention the layers of PaaS architecture.

Cloud Controller

Automatically creates virtual machines and controllers

Deploys applications

Connects to services

Automatically scales up and down

Storage Services

Object

NoSQL

Relational

Block storage

Applications Stored in Storage Services

Simple-to-scale applications

Easier recovery from failure

What are the cloud service models?

Infrastructure as a Service (IaaS)

Platform as a Service (PaaS)

Software as a Service (SaaS)

What are the benefits of Cloud Computing?

The main benefits of Cloud Computing are:

Data backup and storage of data

Powerful server capabilities

Incremented productivity

Cost-effective and time-saving

What is Cloud Computing?

It is an advanced-stage technology implemented so that the cloud provides its services globally as per the user requirements. It provides a method to access several servers worldwide.

What is a Cloud?

A cloud is a collaboration of networks, hardware, services, storage, and interfaces that help in delivering computing as a service. It has three users:

1. End users

2. Business management users

3. Cloud service providers

How to run a Docker container?

The Docker run command manages the running of containers in Docker.

Running a container under a specific name:

The command for running a container under a specific name is:

docker container run --name [container\_name] [docker\_image]

Running a container in the background in the detached mode:

The command for running a container in the background is:

docker container run -d [docker\_image]

Running a container interactively:

The following command is run for running a container interactively:

docker container run -it [docker\_image] /bin/bash

Running a container and publishing container ports:

We have to include -p to the docker run command, along with the following:

-p [host\_ip]:[host\_port]:[container\_port]

Here, host\_ip is optional. It is not mandatory to specify this while we run the command.

Running a container and mounting host volumes:

The docker container run command looks like this:

docker container run -v [/host/volume/location]:[/container/storage] [docker\_image]

What is Docker Hub?

Docker Hub helps with linking to code repositories. This cloud-based registry enables the building, testing, and storing of images in Docker Cloud. Images can also be deployed to the host with it.

Q. Name and explain the states of a Docker container.

1. Created: We see this Docker container state when a container is newly created.

2. Restarting: When the Docker container is restarted due to any issues, this state is observed.

3. Running: It is the main state for the container after it has started.

4. Paused: When a running Docker container is temporarily stopped via docker pause, this is the status that we will see.

5. Exited: If a container has stopped due to some issue or stopped manually, this will be the state of the container.

6. Dead: When the daemon has tried but failed to stop a container (mostly because of a busy device or resource), this state will be seen.

Q. Name and explain the states of a Docker container.

1. Created: We see this Docker container state when a container is newly created.

2. Restarting: When the Docker container is restarted due to any issues, this state is observed.

3. Running: It is the main state for the container after it has started.

4. Paused: When a running Docker container is temporarily stopped via docker pause, this is the status that we will see.

5. Exited: If a container has stopped due to some issue or stopped manually, this will be the state of the container.

6. Dead: When the daemon has tried but failed to stop a container (mostly because of a busy device or resource), this state will be seen.

5. Automation

AWS helps you use automation so you can build faster and more efficiently. Using AWS services, you can automate manual tasks or processes such as deployments, development & test workflows, container management, and configuration management.

6. Secure

Use AWS Identity and Access Management (IAM) to set user permissions and policies. This gives you granular control over who can access your resources and how they access those resources.

7. Large Partner Ecosystem

AWS supports a large ecosystem of partners which integrate with and extend AWS services. Use your preferred third-party and open source tools with AWS to build an end-to-end solution. Visit here to learn more about our DevOps Partner Solutions.

8. Pay-As-You-Go

With AWS purchase services as you need them and only for the period when you plan to use them. AWS pricing has no upfront fees, termination penalties, or long term contracts. The AWS Free Tier helps you get started with AWS. Visit the pricing pages of each service to learn more.

Q. Why AWS for DevOps?

Answer:

1. Get Started Fast

Each AWS service is ready to use if you have an AWS account. There is no setup required or software to install.

2. Fully Managed Services

These services can help you take advantage of AWS resources quicker. You can worry less about setting up, installing, and operating infrastructure on your own. This lets you focus on your core product.

3. Built for Scale

You can manage a single instance or scale to thousands using AWS services. These services help you make the most of flexible compute resources by simplifying provisioning, configuration, and scaling.

4. Programmable

You have the option to use each service via the AWS Command Line Interface or through APIs and SDKs. You can also model and provision AWS resources and your entire AWS infrastructure using declarative AWS CloudFormation templates.

What is the Application Load Balancer (ALB)?

The Classic ELB and the ALB share commonalities in function, but the ALB has been specialized to provide users with enhanced capabilities. In this article, we’re going to investigate the unique functionality offered by each and consider cases where you would decide to implement one instead of the other.

VPC Peering

When you allow to connect one VPC with another VPC through some of direct network route and using some private IP addresses, then this connection is called VPC Peering. and Instances will behave as if they are on the same private network. Users can peer between the regions. From VPC Peering, users can easily peer the VPC’s with another AWS accounts and also users can peer the VPCs in the same account.

Amazon VPC Networking Components:

Amazon Virtual Private Cloud:

It is a networking service that you can use to establish boundaries around your AWS resources. So, in simpler words, Amazon Virtual Private Cloud (Amazon VPC) enables the users to define some virtual network and then launch the AWS resources into that virtual network. It gives you full control over various network environments, resources, connectivity, and security. Moreover, it defines how a network should communicate across different Availability Zones or regions. Users have a option of easy customization of the network configuration for their Amazon Virtual Private Cloud(VPC).

Components of Amazon VPC:

Subnet: A subnet in VPC is something a range of IP addresses. It is a section of a VPC that can contain resources such as Amazon EC2 services and shares a common address component. Public Subnet where resources are exposed to the internet through Internet Gateway and Private Subnet where resources are not exposed to the outside world.

Route Table: They are the set of rules used to decide where the network traffic has to be managed. It specifies the destination i.e IP address and target. The target can be Internet gateway, NAT gateway, Virtual private gateway, etc. With the use of route tables, users can determine where the network traffic will be directed from your subnet or gateway.

Virtual Private Gateway: It is the VPN(Virtual Private Network) hub on the Amazon side of the VPN connection to have a secure transaction. Users can attach it to the VPC from which they want to create the VPN connection.

NAT Gateway: Network Address Translation (NAT) Gateway is used when higher bandwidth, availability with lesser management effort is required. It updates the routing table of the private subnet such that it sends the traffic to the NAT gateway. It supports only UDP, TCP, and ICMP protocols.

VPC Peering: A VPC peering connection allows you to route traffic between two Virtual Private Clouds using IPv4 or IPv6 private addresses. Users can create a VPC peering connection between their own VPC with a VPC in another AWS account. This connection helps you to smoothly transfer the data.

Security Groups: It consists set of firewalls rules that control the traffic for your sample. You can have a single security group associated with multiple instances.

Elastic IP: It is a static IP address which is a reserved public IP address that can be assigned to any Instance in a particular region and never changes.

Network Access Control Lists (NACL): It is an optional layer of security for your VPC that acts as a firewall for controlling traffic in and out of one or more subnets. It adds an additional layer of security to your VPC.

Customer Gateway: VPN connection links your network (or data) to your Amazon VPC (virtual private cloud). A customer gateway is a presenter on your side of that connection. It can be a physical or software appliance.

Network Interface: It’s a connection between private and public networks. Network traffic is automatically shifted to the new instance if you move it from one instance to the other.

VPC Endpoints: It allows VPC to make a connection with other services of AWS without using the internet. They are of two types, Interference endpoints, and Gateway endpoints. They are scaled, redundant, and highly available VPC components.

IP addressing: With the IP Addressing, you can assign your VPCs and subnets, the IPv4 addresses and IPv6 addresses.

Elastic IP address basics

The following are the basic characteristics of an Elastic IP address:

An Elastic IP address is static; it does not change over time.

An Elastic IP address is for use in a specific Region only, and cannot be moved to a different Region.

An Elastic IP address comes from Amazon's pool of IPv4 addresses, or from a custom IPv4 address pool that you have brought to your AWS account.

To use an Elastic IP address, you first allocate one to your account, and then associate it with your instance or a network interface.

When you associate an Elastic IP address with an instance, it is also associated with the instance's primary network interface. When you associate an Elastic IP address with a network interface that is attached to an instance, it is also associated with the instance.

When you associate an Elastic IP address with an instance or its primary network interface, the instance's public IPv4 address (if it had one) is released back into Amazon's pool of public IPv4 addresses. You cannot reuse a public IPv4 address, and you cannot convert a public IPv4 address to an Elastic IP address. For more information, see Public IPv4 addresses.

You can disassociate an Elastic IP address from a resource, and then associate it with a different resource. To avoid unexpected behavior, ensure that all active connections to the resource named in the existing association are closed before you make the change. After you have associated your Elastic IP address to a different resource, you can reopen your connections to the newly associated resource.

A disassociated Elastic IP address remains allocated to your account until you explicitly release it. We impose a small hourly charge for Elastic IP addresses that are not associated with a running instance.

When you associate an Elastic IP address with an instance that previously had a public IPv4 address, the public DNS host name of the instance changes to match the Elastic IP address.

We resolve a public DNS host name to the public IPv4 address or the Elastic IP address of the instance outside the network of the instance, and to the private IPv4 address of the instance from within the network of the instance.

When you allocate an Elastic IP address from an IP address pool that you have brought to your AWS account, it does not count toward your Elastic IP address limits. For more information, see Elastic IP address limit.

When you allocate the Elastic IP addresses, you can associate the Elastic IP addresses with a network border group. This is the location from which we advertise the CIDR block. Setting the network border group limits the CIDR block to this group. If you do not specify the network border group, we set the border group containing all of the Availability Zones in the Region (for example, us-west-2).

An Elastic IP address is for use in a specific network border group only.

What is Amazon Route 53?

Amazon Route 53 is a highly available and scalable Domain Name System (DNS) web service. You can use Route 53 to perform three main functions in any combination: domain registration, DNS routing, and health checking.

If you choose to use Route 53 for all three functions, be sure to follow the order below:

1. Register domain names

Your website needs a name, such as example.com. Route 53 lets you register a name for your website or web application, known as a domain name.

For an overview, see How domain registration works.

For a procedure, see Registering a new domain.

For a tutorial that takes you through registering a domain and creating a simple website in an Amazon S3 bucket, see Getting started with Amazon Route 53.

2. Route internet traffic to the resources for your domain

When a user opens a web browser and enters your domain name (example.com) or subdomain name (acme.example.com) in the address bar, Route 53 helps connect the browser with your website or web application.

For an overview, see How internet traffic is routed to your website or web application.

For procedures, see Configuring Amazon Route 53 as your DNS service.

For a procedure on how to route email to Amazon WorkMail, see Routing traffic to Amazon WorkMail.

3. Check the health of your resources

Route 53 sends automated requests over the internet to a resource, such as a web server, to verify that it's reachable, available, and functional. You also can choose to receive notifications when a resource becomes unavailable and choose to route internet traffic away from unhealthy resources.

For an overview, see How Amazon Route 53 checks the health of your resources.

For procedures, see Creating Amazon Route 53 health checks and configuring DNS failover.

Q. What are the types of load balancing?

Answer:

We can classify load balancing into three main categories depending on what the load balancer checks in the client request to redirect the traffic.

Application load balancing

Complex modern applications have several server farms with multiple servers dedicated to a single application function. Application load balancers look at the request content, such as HTTP headers or SSL session IDs, to redirect traffic.

For example, an ecommerce application has a product directory, shopping cart, and checkout functions. The application load balancer sends requests for browsing products to servers that contain images and videos but do not need to maintain open connections. By comparison, it sends shopping cart requests to servers that can maintain many client connections and save cart data for a long time.

Network load balancing

Network load balancers examine IP addresses and other network information to redirect traffic optimally. They track the source of the application traffic and can assign a static IP address to several servers. Network load balancers use the static and dynamic load balancing algorithms described earlier to balance server load.

Global server load balancing

Global server load balancing occurs across several geographically distributed servers. For example, companies can have servers in multiple data centers, in different countries, and in third-party cloud providers around the globe. In this case, local load balancers manage the application load within a region or zone. They attempt to redirect traffic to a server destination that is geographically closer to the client. They might redirect traffic to servers outside the client’s geographic zone only in case of server failure.

DNS load balancing

In DNS load balancing, you configure your domain to route network requests across a pool of resources on your domain. A domain can correspond to a website, a mail system, a print server, or another service that is made accessible through the internet. DNS load balancing is helpful for maintaining application availability and balancing network traffic across a globally distributed pool of resources.

What are the benefits of load balancing?

Load balancing directs and controls internet traffic between the application servers and their visitors or clients. As a result, it improves an application’s availability, scalability, security, and performance.

Application availability

Server failure or maintenance can increase application downtime, making your application unavailable to visitors. Load balancers increase the fault tolerance of your systems by automatically detecting server problems and redirecting client traffic to available servers. You can use load balancing to make these tasks easier:

Run application server maintenance or upgrades without application downtime

Provide automatic disaster recovery to backup sites

Perform health checks and prevent issues that can cause downtime

Application scalability

You can use load balancers to direct network traffic intelligently among multiple servers. Your applications can handle thousands of client requests because load balancing does the following:

Prevents traffic bottlenecks at any one server

Predicts application traffic so that you can add or remove different servers, if needed

Adds redundancy to your system so that you can scale with confidence

Application security

Load balancers come with built-in security features to add another layer of security to your internet applications. They are a useful tool to deal with distributed denial of service attacks, in which attackers flood an application server with millions of concurrent requests that cause server failure. Load balancers can also do the following:

Monitor traffic and block malicious content

Automatically redirect attack traffic to multiple backend servers to minimize impact

Route traffic through a group of network firewalls for additional security

Application performance

Load balancers improve application performance by increasing response time and reducing network latency. They perform several critical tasks such as the following:

Distribute the load evenly between servers to improve application performance

Redirect client requests to a geographically closer server to reduce latency

Ensure the reliability and performance of physical and virtual computing resources

What is load balancing?

Load balancing is the method of distributing network traffic equally across a pool of resources that support an application. Modern applications must process millions of users simultaneously and return the correct text, videos, images, and other data to each user in a fast and reliable manner. To handle such high volumes of traffic, most applications have many resource servers with duplicate data between them. A load balancer is a device that sits between the user and the server group and acts as an invisible facilitator, ensuring that all resource servers are used equally.

WHAT EXACTLY ARE SNS AND SQS?

Ans: Amazon Simple Notification Service (SNS) is a web service that manages user notifications sent from any cloud platform. From any cloud platform, manage and distribute messages or notifications to users and consumers.

Amazon Simple Queue Service (SQS) administers the queue service, which allows users to move data whether it is running or active.

Q. What is the difference between Scalability and Elasticity?

Answer:

Scalability is the ability of a system to increase its hardware resources to handle the increase in demand. It can be done by increasing the hardware specifications or increasing the processing nodes.

Elasticity is the ability of a system to handle increase in the workload by adding additional hardware resources when the demand increases(same as scaling) but also rolling back the scaled resources, when the resources are no longer needed. This is particularly helpful in Cloud environments, where a pay per use model is followed.

If you want to launch Amazon Elastic Compute Cloud (EC2) instances and assign each instance a predetermined private IP address you should:

Launch the instance from a private Amazon Machine Image (AMI).

Assign a group of sequential Elastic IP address to the instances.

Launch the instances in the Amazon Virtual Private Cloud (VPC).

Launch the instances in a Placement Group.

Answer C.

Explanation: The best way of connecting to your cloud resources (for ex- ec2 instances) from your own data center (for eg- private cloud) is a VPC. Once you connect your datacenter to the VPC in which your instances are present, each instance is assigned a private IP address which can be accessed from your datacenter. Hence, you can access your public cloud resources, as if they were on your own network.

What Is Amazon Virtual Private Cloud (VPC) and How Does It Work?

A VPC is the most efficient way to connect to your cloud services from within your own data centre. When you link your datacenter to the VPC that contains your instances, each instance is allocated a private IP address that can be accessed from your datacenter. As a result, you may use public cloud services as if they were on your own private network.

Can S3 be used with EC2 instances, if yes, how?

Yes, it can be used for instances with root devices backed by local instance storage. By using Amazon S3, developers have access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of web sites. In order to execute systems in the Amazon EC2 environment, developers use the tools provided to load their Amazon Machine Images (AMIs) into Amazon S3 and to move them between Amazon S3 and Amazon EC2.

Another use case could be for websites hosted on EC2 to load their static content from S3.

What exactly is Amazon S3?

Explanation S3 stands for Simple Storage Service, and Amazon S3 is the most extensively used storehouse platform. S3 is an object storehouse service that can store and recoup any volume of data from any position. Despite its rigidity, it’s basically measureless as well as cost-effective because it’s on- a demand storehouse. Away from these advantages, it provides new situations of continuity and vacuity. Amazon S3 aids in data operation for cost reduction, access control, and compliance.

Q. Is one Elastic IP address enough for every instance that I have running?

Answer:

Depends! Every instance comes with its own private and public address. The private address is associated exclusively with the instance and is returned to Amazon EC2 only when it is stopped or terminated. Similarly, the public address is associated exclusively with the instance until it is stopped or terminated. However, this can be replaced by the Elastic IP address, which stays with the instance as long as the user doesn’t manually detach it. But what if you are hosting multiple websites on your EC2 server, in that case you may require more than one Elastic IP address.

What are the best practices for Security in Amazon EC2?

There are several best practices to secure Amazon EC2. A few of them are given below:

Use AWS Identity and Access Management (IAM) to control access to your AWS resources.

Restrict access by only allowing trusted hosts or networks to access ports on your instance.

Review the rules in your security groups regularly, and ensure that you apply the principle of least

Privilege – only open up permissions that you require.

Disable password-based logins for instances launched from your AMI. Passwords can be found or cracked, and are a security risk.

Mention and explain the many types of Amazon EC2 instances.

The various instances available on Amazon EC2 General-purpose Instances:

1. They are used to compute a wide range of tasks and aid in allocating processor, memory, and networking resources.

2. Instances optimized for computing: These are suitable for compute-intensive workloads. They can handle batch processing workloads, high-performance web servers, machine learning inference, and a wide range of other tasks.

3. Memory-optimized: They process and provide tasks that manage massive datasets in memory.

4. Computing speed: It accelerates the execution of floating-point number calculations, data pattern matching, and graphics processing.

5. Optimized Storage: They conduct operations on local storage that need sequential read and write access to big data sets.

Q. Your application is running on an EC2 instance. When your instance’s CPU consumption reaches 80%, you must lower the load on it. What method do you employ to finish the task?

A. Setting up an autoscaling group to deploy new instances when an EC2 instance’s CPU consumption exceeds 80% and distributing traffic among instances via the deployment of an application load balancer and the designation of EC2 instances as target instances can do this.

Q. How is stopping and terminating an instance different from each other?

Starting, stopping and terminating are the three states in an EC2 instance, let’s discuss them in detail:

Answer:

a) Stopping and Starting an instance: When an instance is stopped, the instance performs a normal shutdown and then transitions to a stopped state. All of its Amazon EBS volumes remain attached, and you can start the instance again at a later time. You are not charged for additional instance hours while the instance is in a stopped state.

b) Terminating an instance: When an instance is terminated, the instance performs a normal shutdown, then the attached Amazon EBS volumes are deleted unless the volume’s deleteOnTermination attribute is set to false. The instance itself is also deleted, and you can’t start the instance again at a later time.

What are the featured services of AWS?

AWS’s key components are as follows:

a) Elastic compute cloud (EC2): It is a computing resource that is available on demand for hosting applications. In times of uncertain workloads, EC2 comes in handy.

b) Route 53: It is a web-based DNS service.

c) Simple Storage Device S3: This is a storage device service that is widely used in AWS Identity and Access Management.

d) Elastic Block Store: It allows you to store constant volumes of data and is integrated with EC2. It also allows you to persist data.

e) Cloud watch: It allows you to monitor the critical areas of AWS and even set a reminder for troubleshooting.

f) Simple Email Service: It allows you to send emails using regular SMTP or a restful API call.

What are the various layers of cloud computing? Explain their work.

Cloud computing categories have various layers that include

Infrastructure as a Service (IaaS) is the on-demand provision of services such as servers, storage, networks, and operating systems.

Platform as a Service (PaaS) combines IaaS with an abstracted collection of middleware services, software development, and deployment tools.

PaaS also enables developers to create web or mobile apps in the cloud quickly.

Software as a Service (SaaS) is a software application that has been delivered on-demand, in a multi-tenant model

Function as a Service (FaaS) enables end users to build and execute app functionalities on a serverless architecture.

\*\*\*What is Git & VCS (version contorl system)

 Git is a free open-source version control system.

 Version control software keeps track of every modification of the code in a special kind

of database.

 Vcs system helps term of rollback to previous version in case of any issue with specific

version.

\*\*\* Need of version control system

types of vcs :- Basically two types of vcs.

1. Centralized version control system (CVCS)

2. Distributed version control system

Centralized version control system :- CVCS uses a central server to store all files and enables

team collaboration.

CVCS works on a single repository to which users can directly access a control server.

Centralized version control system (CVCS)

Repository (server)

Working Copy

Workstation pc1

 Here each workstation is connected with the central code repository.

 Drawbacks: - It is not locally available.

 Crash of cvcs will result in losing the entire data of the project.

Distributed version control system :-

 In distributed vcs every contributor has a local copy or "clone" of the main repository.

 Users can change and commit local repo without any interference.

 Users can update their local repo from the central server.

 Users can update the central server from their repo.

 Operations in dvcs are fast.

 New changes can be made locally without manipulation.

 If the central server gets crashed at any point of time, the lost data can be easily

recovered from any one-off contributors’ local repositories.

GIT Key Terminology: -

Git Repository: - Git repository contains files, history, config managed by GIT.

The git repository

Working dir

Repository

~/projects/my.project

./git

Text

V1 txt, v2 text

Stages of GIT:-

Working dir: - Area of live files also known as untracked area of GIT.

Staging Area: - Staging area is when git starts tracking and saving changes that occur in files.

Git Direcotry:- Also called "local repo" is yours .git repo its area where git save everything.

Remote Repository: - GITHUB

Remote repository is stored on a code hosting service like GitHub or on an internal server.

Branch In Git:- Branch in Git is a way to keep developing and coding a new feature or

modification to the software and still not effecting the main part of the project.

\*\*\*Install GIT on linux machine: -

Git is open source and can be installed on all major os like Linux, mac, windows.

Here we are going to install git in the ubuntu machine.

# sudo su

#apt-get update

To install git

#apt-get install git -y

To check version

#git --version

\*\*\*Install GIT on windowos machine: -

 Go to the browser search download git.

 open the git website and download the file.

 After download run the setup file in machine

 Check the version.

c://Git -version

Note: - If you want to open git then searches gitbash.

Configure User information in GIT:-

Go to windows system open git bash.

Check the version by GIT --version cmd.

Note: - user configuration is global configuration

#git config --global user.name "Raushan kumar"

#git config --global email.name "raushan123@gmail.com"

To check user configuration details: -

#git config --list

Create First GIT Repository: -

#mkdir project

#cd project

Initialize the GIT by

#git init

GIT COMMIT:-

GIT STATUS: - it is a command which provides the details of current working details.

#git add (file name)

#git add first.file.txt

#git status

Note: - with the help of add. cmd we can move all working to staging area.

To this cmd we can commit working dir to staging area.

#git add .

We have other cmd for commit with commits: -

#git commit -m “first file commit "

To diff cmd we can see the updated data in the file.

#git diff

\*\*\* Git tracking & git Logs:-

First check the working dir on git with the help of cmd also we can create new project so

working to create new project.

#touch myfile

#ls

To check the status:-

#git status

After commit the code by commit cmd.

#git commit -m "my commit for second file"

#git status

GIT LOGS: -

With the help of git log we can get to know about t

he commit details or commit file metadata.

#git log :- If we run this cmd then shows the logs of all commit details but in case we want to

check the particular single or author commit then run below cmd.

#git log --author="Raushan kumar"

we can get the details of commit with the help email also.

#git log --author="raushan123@gmail.com"

\*\*\*Explore GIT with GITHUB: -

 Git hub is a website and cloud-based service that helps developers store and manage

their code as well as track and control changes to their code.

 GIT hub is remote repository for your code base.

How to create account in github:-

Login into GitHub ----create a new repository.

Owner

Repository Name

Raushan

Git Training

Description: -

Public

private

Add a README.file

add.gitignore

template gitbook

create.

Note:- if you want to sync with local git to remote git server firstly login into github ---goto the

code area form the will get clone option ,https, ssh, download zip other.

will use https copy that link after that come to the local server check the current path select the

path where you want to crrate a clone of remote server in your machine.

Basically git works an root folder come to the root folder and run this cmd.

#git clone -b main (paste the link here)

Check to remote repo

#ls

#git status

We can create a file an local server that file we will upload to the remote server or central

server.

In that way will use push or pull cmd.

#vi text.file

insert some data in that folder and save it :wq

#git status

Note:- if we want to add or message in cmd area.

This cmd commit with changes & message (commit for track file)

#git commit -am "commit message"

For push the code to central server:-

#git push

First time ask username and password of git account.

\*\*\*Setup authentication in local github:-

Login into your github account -----settings---sshkey and GPG keys. (open this here will get

some option like your personal profile , nes ssh key,new gpg key with the help of ssh & gpg keys

for the setup.

Open ssh keys:- gener ssh key (click on it ) here will get some option like about ssh checking

for existing ssh keys generating a new ssh key and adding it to the ssh-agent adding a new.ssh

key to your github account.

For jenerate the SSH key:-

1.Open gitbash

2. Your github email address.

#ssh-keygen -t ed25519 -c"raushan123@gmail.com"

After that they ask confirmation then just press enter.

will get some details:-

Enter file in which yo save the key (rakumar/.ssh/id\_ed25519) :- press enter

Enter passphare ( empty for no phasspharse)

After that you get username or ssh id like (id\_ed25519) :- enter

jenerate the ssh key and copy:-

firstly goto that location like (rakumar/.sh/id)ed25519)

Copy the sshkey come back to github account ----setting--ssh and gpg keys---inside the ssh

keys click on new ssh key

sshkeyss/addnew

tittle:- windows ssh key

Key :- paste the ssh key----add ssh key

After that we need to test for that again we need to goto github ---setting---ssh and gpg

keys--ssh keys----generate sshkeys (click on that then you will get the option for testing ssh

connection -----click on testing your ssh connection).

Here will get step for the test ssh connection

Open git bash

enter the following for test.

#ssh -T git@github.com

Before running this cmd you should an the clone repo in your local machine.

After verification try to push or pull something from your local repo to remote server (github).

#git push

Branch In Git:-

To check branch in local repo

#git branch

To create new branch:-

# git branch develop (branch name)

To check create branch

#git branch

If you want to switch the branch:-

#git switch branch name

#git branch develop

create some file or dir here in develop branch

#cat file.txt g1 23 e3

check the status (here will get untarck file)

#git status

To cehck file or dir

#ll

To move the file from working dir to staging area.

#git add .

After check the stauts

#git status

To permanent save we have to commit the file by below cmd:--

#git commit -am "commit simple file a

develop"

After that push the code to github server by cmd:-

#git push

First time we got error like (fatal:- the current branch develop has no upstream barnch to push

the current branch and set the remote as upstream use).

then we have to use below cmd:-

#git push origin develop

After run this cmd go to your github server referesh then you will get your develop branch

details like file or dir.

NOTE:- If you want to verify develop branch file or dir refelecting or not in master barnch.

#git switch master

#ll

Here some file are mssing like file.txt g1 23 e3

After creation of develop branch if we create some file or dir in master branch it will not reflect

in develop branch.

create some file in master and verify:-

come to master branch by git switch master

Come to the master branch

#git switch master

Create file and some dir

#touch h1 b3 4

To check the status

#git status

To Move in staging area

#git add .

#git commit -am "test commit"

#git push

After check into the develop branch file will not refelect.

Note:- if we are going to push the code by master branch then we dont need to used git push

origin (mastername).

 All the branches are sync with master branch.

 We can merge the all branch with master branch.

Note:- If you want to merge the code of two diffrent branchs like master or developer.

Go to your github account--verify fire where to where you want to merge the code.

Here we have two branch main or develop so we want to merge the code from develop to

main.

Go to the pullreuqest---create new pull request--compare the changes.

Base:- develop branch ---compare ----main branch

Create pull reuqest

tittle:- New file.txt

create pull reuqest

Review

merge pull reuqest ----confirm

Check the files in develop branch

If you want to work an merge file then we have to pull the latest code from develop branch.

Cmd:- Git pull origin develop

come to develop branch then run this cmd.

branch develop:-

#git pull origin develop

How to add, commit in GIT:-

Create one dir and go inside it.

#sudo su

#Git init

# touch myfile (put some data)

# git status

#Git add .

#git commit -m "1st commit"

#git status

# git log

#git show <commit id>

#git push origin (branch name)

#git pull origin (branch name)

Merge branches in GIT:-

 Isolating features into diffrent branches is crucial practice for any serious developer.

 At some point a pieace of code will reach a state where you will want to integrate it with

the rest of the project. this is where the git merge command come in.

Before merge :-

c0 c1 c2 c3 c4----hotfix

after merge:-

c0 c1 c2 c3 c4--- master

Practically we are going to do that:- go to your develop branch and do ls create fsome file or put

some data.

After that come to master branch run merge cmd

#git branch master

#git merge develop

Here will get all file inside the develop branch

#ls

Note:- if we want to upload this code on github then we have to commit this code.

#git add .

#git status

#git commit

#git push

Revert in previous state in GIT:-

 Revert is all about to undo the changes you did in repo.

 In GIT this can be done via RESET and REVERT

 RESET- Practically user can think of it as a “rollback".

 Reset points local environment back to a previous commit.

 Get reset is a powerful command that is used to undo local changes to the state of a get

repo.

 To reset staging area:-

git reset <filename>

To reset the changes from both staging area and working dir at a time.

If we want to delete the add. file or current working dir file then we can use this cmd

git reset --hard

REVERT:-

 The Net effect of the git revert command is similar to reset, buts its approach is

different.

 The revert command helps you undo an existing commit to.

 Revert adds a new commit at the end of the chain to "cancel" changes.

 REVERT or RESET?

 If User have already pushed commit to the remote repo , a revert is a mice way to

cancel out changes.

 Git workflow works well for picking up additional commits at the end of a branch, but it

can be challenging if a

set of commits is no longer seen in the chain when someone

resets the branch pointer back.

 It commits in local then reset is good, it commits is pushed then revert is good option.

Practical: -

Git reset:-

firstly, check the all oneline: -

to check log:-

git log --oneline

create one file or put some data in that file

touch file1

vi file1

i am raushan.

:wq

git status

git add .

git commit - am "first commit"

again edit file1 and enter some more data along with previous data.

vi file1

i am emplyee of estuate,

:wq

git add

git commit -am "second commit"

To check the latest commit:-

git log --oneline

after commit we fell ilke second commit is not correct in that way we will use RESET for

previous code line or commit.

If you want to reset your second commit then use this cmd firstly find the commit id with the

help of commit.

git log --oneline

copy the second git commit id

git reset --soft (commit id)

git reset --soft 492930e

To soft cmd it will delete from the commit history log only if you want to delete from dir as well

as then we have to used hard cmd like

get reset --hard 492930e

After that check log it is reset or not.

git log --oneline

How to work with GIT REVERT:-

Firstly modify into the file

vi file1

along with previous code.

nikhil is my best friend.

:wq

git commit -am "commit revert"

git push

edit again in file1

vi file1

nikhil dish is chicken.

:wq

git commit -am "commit revert2"

git push

after push check the file in your github server.

git log --oneline

If you want to delete the commit of "commit revert" from remote server use below cmd

To this cmd we can delete the latest commit to the remote server.

git revert HEAD

git push

If we run this cmd they will create another commit with previous commit in that commit they

will delete the content as well as.

git log --oneline

Merge branches in GIT:-

To merge the branches master to develop:-

Git merge develop

To check the details of merge:-

git log --oneline --graph

Merge branch with message cmd:-

git merge branchname -m "message"

git merge main -m "merging main to develop"

If you want to merge develop to main come to develop mode:-

git switch develop

git merge main -m "merge develop to main"

git add .

git commit

git log --oneline --graph

Git Conflict:-

 When same file having different content in different branches if you do merge, conflict occurs

resolve conflict then add and commit.

 Conflict occurs when you merge branches.

Git Stash:-

 Suppose you are implementing a new feature for your product, your code is in progress and

suddenly a customer escalation comes because of this, you have to keep aside your new feature

work few hours.

 you cannot commit your partial code and also cannot throw away your changes so you need

some temporary storage, when you can store your partial charges and later an commit it.

 To stash an item (only applies to modified files nt new files).

GIT stash commands:-

To stash an item:-

git stash

To see stash ite list:-

git stash list

To apply stashed items: -

git stash apply (stash@{0})

To clear the stash items:-

git stash clear/drop

Stash Untracked files & GIT stashed POP:-

Come to terminal run below cmd to check untrack status: -

git status

we can get the details of track or untracked file with the help of below cmd:-

git stash -a

o/p:- stash the data

If you want to continue your work o pending or stash file used another method or cmd also.

git stash pop

To check list:-

git stash list

To pop will retrieve the file or delete from the stash area but apply cmd retrieve the data file noe delete

from the stash.

DOCKER

➢ Docker is an opensource centralized platform designed to create, deploy and run

applications.

➢ Docker uses container on the host OS to run applications. It allows applications to use same

Linux kernel as a system on the host computer rather than creating a whole virtual OS.

➢ We can install docker on any OS but docker engine runs natively on Linux distributions.

➢ Docker written in “GO” programming language.

➢ Docker is a tool that performs OS level virtualization also known as Containerization.

➢ Before docker many users face the problem that a particular code is running in the

developer’s system but not in the user’s system.

➢ Docker was first release in march 2013. It is developed by Solomon Hykes and Sebastian

Pahl.

➢ Docker is a set of Platform-as-a-Service that uses OS level virtualization whereas VMWare

uses hardware level of virtualization.

Advantages of Docker:

➢ No pre allocation of RAM.

➢ CI efficiency: - docker enables you to build a container image and use that same image

across every step of the deployment process.

➢ Less cost.

➢ It is light in weight.

➢ It can run on physical h/w / virtual h/w or on cloud.

➢ You can reuse this image.

➢ It took very less time to create container.

Disadvantages of Docker:

➢ Docker is not a good solution for application that requires rich GUI.

➢ Difficult to manage large number of containers.

➢ Docker doesn’t provide cross platform compatibility means if an application is designed to

run in a docker container in windows than it can’t run in Linux or vice-versa.

➢ Docker is suitable when the development O.S and testing O.S are same. If the O.S are

different then we should use VM.

➢ No solution for data recovery and backup.

Components of Docker:

A. Docker Daemon:

➢ Docker daemon runs on host O.S.

➢ It is responsible for running containers to manages docker services.

➢ Docker daemon can communicate with other daemons.

B. Docker Client:

➢ Docker users can interact with docker through a client.

➢ Docker client uses commands and REST API to communicate with the docker

daemon.

➢ When a client runs any server command on the docker client terminal, the client

terminal sends these docker commands to the docker daemon.

➢ It is possible for docker client to communicate with more than one daemon.

C. Docker Host:

➢ Docker host is used to provide an environment to execute and run applications.

➢ It contains the docker daemon, images, containers, networks and storages.

D. Docker Hub/ Registry:

➢ Docker registry manages and stores the docker image.

➢ There are two types of registries in the docker:

a. Public Registry: it is also called as docker hub.

b. Private Registry: it is used to share image with in the enterprise.

E. Docker Image:

➢ Docker images are the read only binary templates used to create docker containers.

or

➢ Single file with all the dependencies and configuration required to run a program.

Ways to Create an Image:

a. Take image from the docker hub.

b. Create image from docker file.

c. Create image from existing docker containers.

F. Docker Containers:

➢ Containers hold the entire packages that is needed to run the application.

Or

➢ In other words, we can say that the image is a template and the container is a copy

if that template.

➢ Container is like a virtual machine.

➢ Images becomes container when they run on docker engine.

Basic Docker Commands:

To see all images present in your local repo:

# docker images

To find out images in docker hub

# docker search image\_name

To download image from dockerhub to local machine

# docker pull image\_name

To give a name to container

# docker run -it --name new\_name image\_name /bin/bash

To check service start or not (status)

# docker service status

To start: #docker service start

To stop: # docker service stop

To start container

#docker start container\_name

To go inside container

# docker attach container\_name

To see all comtainers

# docker ps -a

To see running containers

# docker ps

To stop container

# docker stop container\_name

To delete a container

# docker rm container\_name

Create container from our own Image:

Log

in into AWS account and start your EC2 instance, access it from putty.

Now we have to create container from our own image. Therefore, create one container first:

#docker run -it –name container\_name image\_name /bin/bash

#cd tmp/

Now create one file inside this tmp directory

# touch myfile

Now if you want to see the difference between the basic image and the changes on it

# docker diff container\_name image\_name

Now create image of this container

# docker commit newcontainer\_name image\_name

#docker images

Now create container from this image

# docker run -it --name newcontanier\_name image\_name /bin/bash

# ls

# cd tmp

# ls (you will get all of your files)

Dockerfile:

Dockerfile is basically a text file. It contains some set of instructions. Automation of docker image

creation.

Dockerfile components:

FROM: for base image, this command must be on the top of the dockerfile.

RUN: to execute commands, it will create a layer in image

MAINTAINER: author/ owner/ description

COPY: copy files from local system (docker vm) we need to provide source, destination (we can’t

download file from internet and any remote repo.)

ADD: similar to copy but it provides a feature to download files from internet, also extract file at

docker image side.

EXPOSE: to expose ports such as port 8080 for tomcat , port 80 for nginx etc.

CMD: execute commands but during container creation.

ENTRYPOINT: similar to CMD but has higher priority over CMD, first commands will be

executed by ENTRYPOIN only.

ENV: environment variables

Dockerfile

➢ Create a file named Dockerfile

➢ Add instructions in Dockerfile

➢ Build dockerfile to create image

➢ Run image to create container

# vi Dockerfile

FROM ubuntu

RUN echo “Nagarjuna hota” > /tmp/testfile

To create image out of Dockerfile

# docker build -t myfile

#docker ps -a

# docker image

Now create container from the above image

#docker run -it --name mycon mying /bin/bash

#cat /tmp/testfile

#vi dockerfile

FROM ubuntu

WORKDIR /tmp

RUN echo “thank you” > /tmp/testfile

ENV myname naga

COPY testfile1 /tmp

ADD test.tar.gz /tmp

Docker Volume:

➢ Volume is simply a directory inside our container.

➢ Finally, we have to declare this directory as a volume and then share volume.

➢ Even if we stop the container still, we can access volume.

➢ Volume will be created in one container.

➢ You can declare a directory as a volume only while creating container.

➢ You can’t create volume from existing container.

➢ You can share one volume across any number of containers.

➢ Volume will not be included when you update an image.

➢ You can map volume in two ways:

a. Container to container

b. Host to container

Benefits of Volume:

➢ Decoupling container from storage.

➢ Share volume among different containers.

➢ Attach volume to containers.

➢ On deleting container volume doesn’t delete.

Creating Volume from Dockerfile:

Create a Dockerfile and write

FROM ubuntu

VOLUME “myvolume”

Then create image from this Dockerfile

#docker build -t myimage

Now create a container from this image and run

# docker run -it --name container1 myimage /bin/bash

Now do ls, you can see myvolume.

Now share volume with another container

Container to container

# docker run -it --name container2 (new) --privileged=true –volumesfrom container1 ubuntu

/bin/bash

Now after creating container2, myvolume is visible. Whatever you do in one volume, can see from

other volume.

#touch /myvolume/samplefile

#docker start container1

# docker attach container1

#ls/myvolume

You can see sample file here then exit.

Now create volume by using command:

#docker run -it --name container3 -v /volume2 ubuntu /bin/bash

# ls

#cd /volume2

Now create one file cont3file and exit

Now create one more container and share volume2

#docker run -it --name container4 --privileged=true --volumefrom container3 ubuntu /bin/bash

Now you re inside container do ls you can see volume2

Now create one file inside this volume and then check in container3 you can see that file.

Volumes (Host to Container)

Verify files in /home/ec2-user

#docker run -it --name hostcontainer -v /home/ec2-user:/container --privile

ged=true ubuntu

/bin/bash

#cd /container

Do ls, now you can see all files of host machine.

#touch contanerfile (in container) and exit

Now check in EC2 machine you can see this above file.

Some other commands:

#docker volume ls

#docker volume create <volumename>

#docker volume rm <volumename>

#docker volume prune (it removes all unused docker volume)

#docker volume inspect <volumename>

#docker container inspect <containername>

Docker Port Expose:

Login into AWS account, create one linux instance.

Now go to putty -> login as -> ec2-user

#sudo su

# yum update -y

# yum install docker -y

# service docker start

# docker run -td --name techserver -p 80:80 ubuntu

# docker ps

# docker port techserver o/p- 80/tcp – 0.0.0.0/80

# docker exec -it techserver /bin/bash

# apt-get update

# apt-get install apache2 -y

# cd /var/www/html

# echo “write some msg” > index.html

#service apache2 start

# docker run -td --name myjenkins -p 8080:8080 jenkins

Difference between docker attach and docker exec:

➢ Docker ‘exec’ creates a new process in the container’s environment while docker ‘attach’

just connect the standard input/output of the main process inside the container to

corresponding standard input/output error of current terminal.

➢ Docker ‘exec’ is specifically for running new things in an already started container be it a

shell or some other process.

What is the difference between docker expose and publish:

Basically you have three options:

1. Neither specify expose nor -p

2. Only specify expose

3. Specify expose and -p

1. If you specify neither expose nor -p, the service in the container will only be accessible

from inside the container itself.

2. If you expose a port, the service in the container is not accessible from outside docker but

from inside other docker containers so this is good for inter-container communication.

3. If you expose and -p a port, the service in the container is accessible from anywhere even

outside docker.

If you do –p but do not expose docker does an implicit expose. This is because if a port is open to

the public, it is automatically also open to the other docker containers. Hence -p includes expose.

How to push docker image in docker hub:

Go to AWS account – select Amazon linux

Now go to putty – login as – ec2-user

#sudo su

#yum update -y

#yum install docker -y

#service docker start

#docker run -it ubuntu /bin/bash

Now create some files inside container, now create image of this container

#docker commit container1 image1

Now create account in hub.docker.com

Now go to EC2 instance

#docker login

Enter your username and password

Now give tag to your image

#docker tag image1 dockerid/newimage

#docker push dockerid/newimage

Now you can see this image in docker hub account

Now create one instance in another region and pill image from hub

#docker pull dockerid/newimage

#docker run -it --name mycon dockerid/newimage /bin/bash

Some important commands:

Stop all running containers: # docker stop $(docker ps -a -q)

Delete all stopped containers: # docker rm $(docker ps -a -q)

Delete all images: docker rmi -f $(docker images -q)

DOCKER

➢ Docker is an opensource centralized platform designed to create, deploy and run

applications.

➢ Docker uses container on the host OS to run applications. It allows applications to use same

Linux kernel as a system on the host computer rather than creating a whole virtual OS.

➢ We can install docker on any OS but docker engine runs natively on Linux distributions.

➢ Docker written in “GO” programming language.

➢ Docker is a tool that performs OS level virtualization also known as Containerization.

➢ Before docker many users face the problem that a particular code is running in the

developer’s system but not in the user’s system.

➢ Docker was first release in march 2013. It is developed by Solomon Hykes and Sebastian

Pahl.

➢ Docker is a set of Platform-as-a-Service that uses OS level virtualization whereas VMWare

uses hardware level of virtualization.

Advantages of Docker:

➢ No pre allocation of RAM.

➢ CI efficiency: - docker enables you to build a container image and use that same image

across every step of the deployment process.

➢ Less cost.

➢ It is light in weight.

➢ It can run on physical h/w / virtual h/w or on cloud.

➢ You can reuse this image.

➢ It took very less time to create container.

Disadvantages of Docker:

➢ Docker is not a good solution for application that requires rich GUI.

➢ Difficult to manage large number of containers.

➢ Docker doesn’t provide cross platform compatibility means if an application is designed to

run in a docker container in windows than it can’t run in Linux or vice-versa.

➢ Docker is suitable when the development O.S and testing O.S are same. If the O.S are

different then we should use VM.

➢ No solution for data recovery and backup.

Components of Docker:

A. Docker Daemon:

➢ Docker daemon runs on host O.S.

➢ It is responsible for running containers to manages docker services.

➢ Docker daemon can communicate with other daemons.

B. Docker Client:

➢ Docker users can interact with docker through a client.

➢ Docker client uses commands and REST API to communicate with the docker

daemon.

➢ When a client runs any server command on the docker client terminal, the client

terminal sends these docker commands to the docker daemon.

➢ It is possible for docker client to communicate with more than one daemon.

C. Docker Host:

➢ Docker host is used to provide an environment to execute and run applications.

➢ It contains the docker daemon, images, containers, networks and storages.

D. Docker Hub/ Registry:

➢ Docker registry manages and stores the docker image.

➢ There are two types of registries in the docker:

a. Public Registry: it is also called as docker hub.

b. Private Registry: it is used to share image with in the enterprise.

E. Docker Image:

➢ Docker images are the read only binary templates used to create docker containers.

or

➢ Single file with all the dependencies and configuration required to run a program.

Ways to Create an Image:

a. Take image from the docker hub.

b. Create image from docker file.

c. Create image from existing docker containers.

F. Docker Containers:

➢ Containers hold the entire packages that is needed to run the application.

Or

➢ In other words, we can say that the image is a template and the container is a copy

if that template.

➢ Container is like a virtual machine.

➢ Images becomes container when they run on docker engine.

Basic Docker Commands:

To see all images present in your local repo:

# docker images

To find out images in docker hub

# docker search image\_name

To download image from dockerhub to local machine

# docker pull image\_name

To give a name to container

# docker run -it --name new\_name image\_name /bin/bash

To check service start or not (status)

# docker service status

To start: #docker service start

To stop: # docker service stop

To start container

#docker start container\_name

To go inside container

# docker attach container\_name

To see all comtainers

# docker ps -a

To see running containers

# docker ps

To stop container

# docker stop container\_name

To delete a container

# docker rm container\_name

Create container from our own Image:

Log

in into AWS account and start your EC2 instance, access it from putty.

Now we have to create container from our own image. Therefore, create one container first:

#docker run -it –name container\_name image\_name /bin/bash

#cd tmp/

Now create one file inside this tmp directory

# touch myfile

Now if you want to see the difference between the basic image and the changes on it

# docker diff container\_name image\_name

Now create image of this container

# docker commit newcontainer\_name image\_name

#docker images

Now create container from this image

# docker run -it --name newcontanier\_name image\_name /bin/bash

# ls

# cd tmp

# ls (you will get all of your files)

Dockerfile:

Dockerfile is basically a text file. It contains some set of instructions. Automation of docker image

creation.

Dockerfile components:

FROM: for base image, this command must be on the top of the dockerfile.

RUN: to execute commands, it will create a layer in image

MAINTAINER: author/ owner/ description

COPY: copy files from local system (docker vm) we need to provide source, destination (we can’t

download file from internet and any remote repo.)

ADD: similar to copy but it provides a feature to download files from internet, also extract file at

docker image side.

EXPOSE: to expose ports such as port 8080 for tomcat , port 80 for nginx etc.

CMD: execute commands but during container creation.

ENTRYPOINT: similar to CMD but has higher priority over CMD, first commands will be

executed by ENTRYPOIN only.

ENV: environment variables

Dockerfile

➢ Create a file named Dockerfile

➢ Add instructions in Dockerfile

➢ Build dockerfile to create image

➢ Run image to create container

# vi Dockerfile

FROM ubuntu

RUN echo “Nagarjuna hota” > /tmp/testfile

To create image out of Dockerfile

# docker build -t myfile

#docker ps -a

# docker image

Now create container from the above image

#docker run -it --name mycon mying /bin/bash

#cat /tmp/testfile

#vi dockerfile

FROM ubuntu

WORKDIR /tmp

RUN echo “thank you” > /tmp/testfile

ENV myname naga

COPY testfile1 /tmp

ADD test.tar.gz /tmp

Docker Volume:

➢ Volume is simply a directory inside our container.

➢ Finally, we have to declare this directory as a volume and then share volume.

➢ Even if we stop the container still, we can access volume.

➢ Volume will be created in one container.

➢ You can declare a directory as a volume only while creating container.

➢ You can’t create volume from existing container.

➢ You can share one volume across any number of containers.

➢ Volume will not be included when you update an image.

➢ You can map volume in two ways:

a. Container to container

b. Host to container

Benefits of Volume:

➢ Decoupling container from storage.

➢ Share volume among different containers.

➢ Attach volume to containers.

➢ On deleting container volume doesn’t delete.

Creating Volume from Dockerfile:

Create a Dockerfile and write

FROM ubuntu

VOLUME “myvolume”

Then create image from this Dockerfile

#docker build -t myimage

Now create a container from this image and run

# docker run -it --name container1 myimage /bin/bash

Now do ls, you can see myvolume.

Now share volume with another container

Container to container

# docker run -it --name container2 (new) --privileged=true –volumesfrom container1 ubuntu

/bin/bash

Now after creating container2, myvolume is visible. Whatever you do in one volume, can see from

other volume.

#touch /myvolume/samplefile

#docker start container1

# docker attach container1

#ls/myvolume

You can see sample file here then exit.

Now create volume by using command:

#docker run -it --name container3 -v /volume2 ubuntu /bin/bash

# ls

#cd /volume2

Now create one file cont3file and exit

Now create one more container and share volume2

#docker run -it --name container4 --privileged=true --volumefrom container3 ubuntu /bin/bash

Now you re inside container do ls you can see volume2

Now create one file inside this volume and then check in container3 you can see that file.

Volumes (Host to Container)

Verify files in /home/ec2-user

#docker run -it --name hostcontainer -v /home/ec2-user:/container --privile

ged=true ubuntu

/bin/bash

#cd /container

Do ls, now you can see all files of host machine.

#touch contanerfile (in container) and exit

Now check in EC2 machine you can see this above file.

Some other commands:

#docker volume ls

#docker volume create <volumename>

#docker volume rm <volumename>

#docker volume prune (it removes all unused docker volume)

#docker volume inspect <volumename>

#docker container inspect <containername>

Docker Port Expose:

Login into AWS account, create one linux instance.

Now go to putty -> login as -> ec2-user

#sudo su

# yum update -y

# yum install docker -y

# service docker start

# docker run -td --name techserver -p 80:80 ubuntu

# docker ps

# docker port techserver o/p- 80/tcp – 0.0.0.0/80

# docker exec -it techserver /bin/bash

# apt-get update

# apt-get install apache2 -y

# cd /var/www/html

# echo “write some msg” > index.html

#service apache2 start

# docker run -td --name myjenkins -p 8080:8080 jenkins

Difference between docker attach and docker exec:

➢ Docker ‘exec’ creates a new process in the container’s environment while docker ‘attach’

just connect the standard input/output of the main process inside the container to

corresponding standard input/output error of current terminal.

➢ Docker ‘exec’ is specifically for running new things in an already started container be it a

shell or some other process.

What is the difference between docker expose and publish:

Basically you have three options:

1. Neither specify expose nor -p

2. Only specify expose

3. Specify expose and -p

1. If you specify neither expose nor -p, the service in the container will only be accessible

from inside the container itself.

2. If you expose a port, the service in the container is not accessible from outside docker but

from inside other docker containers so this is good for inter-container communication.

3. If you expose and -p a port, the service in the container is accessible from anywhere even

outside docker.

If you do –p but do not expose docker does an implicit expose. This is because if a port is open to

the public, it is automatically also open to the other docker containers. Hence -p includes expose.

How to push docker image in docker hub:

Go to AWS account – select Amazon linux

Now go to putty – login as – ec2-user

#sudo su

#yum update -y

#yum install docker -y

#service docker start

#docker run -it ubuntu /bin/bash

Now create some files inside container, now create image of this container

#docker commit container1 image1

Now create account in hub.docker.com

Now go to EC2 instance

#docker login

Enter your username and password

Now give tag to your image

#docker tag image1 dockerid/newimage

#docker push dockerid/newimage

Now you can see this image in docker hub account

Now create one instance in another region and pill image from hub

#docker pull dockerid/newimage

#docker run -it --name mycon dockerid/newimage /bin/bash

Some important commands:

Stop all running containers: # docker stop $(docker ps -a -q)

Delete all stopped containers: # docker rm $(docker ps -a -q)

Delete all images: docker rmi -f $(docker images -q)

KUBERNETES

What is Kubernetes?

Kubernetes, often abbreviated as K8s

It is an open-source container orchestration platform.

It is designed to automate the deployment, scaling, and management of containerized applications.

Why we use Kubernetes?

Kubernetes is a powerful tool for managing containerized applications in production environments.

Its features address many challenges associated with deploying and maintaining distributed systems

Key features of Kubernetes include:

Container Orchestration :

Automates deployment and management of containerized applications.

Scalability:

Dynamically scales applications based on demand for optimal resource use.

High Availability:

Ensures reliable application performance by distributing containers across nodes.

Portability:

Provides a consistent platform across diverse infrastructures and environments.

Resource Efficiency:

Optimizes resource allocation and scales applications based on utilization.

Service Discovery and Load Balancing:

Facilitates exposing and balancing traffic for applications.

Community and Ecosystem:

benefits from a large, active open-source community and extensive tooling.

Rolling Updates and Rollbacks:

Supports seamless application updates and automatic rollbacks.

Security and Compliance:

Implements features for securing applications and managing access.

Cost Savings:

contributes to savings through optimized resource use and automation.

Architecture of Kubernetes

Kubernetes Cluster mainly consists of a Control Plane and Worker Machines called Nodes

It is also called as Master-Slave system

We can create single or multiple master nodes for high availability and scalability.

We also can create multiple slave nodes to handle worl load

key components and architecture of Kubernetes:

API Server:

The API Server acts as the central management entity and the frontend interface for the Kubernetes control plane.

It processes RESTful requests to manage cluster resources and updates the corresponding objects in etcd.

It serves as the communication hub for other control plane components to interact and manage the cluster.

ETCD:

etcd is a lightweight, distributed key-value store that securely and consistently stores the critical data of the Kubernetes cluster.

It holds the cluster's configuration and state, including information about nodes, pods, configurations, secrets, and the state of workloads.

Scheduler:

The Scheduler is responsible for assigning workloads, specifically pods, to appropriate nodes.

It selects nodes for new pods based on resource requirements, quality of service requirements, affinity specifications, and other criteria.

Controller Manager:

This component runs various controller processes in the background to regulate the state of the cluster and handle routine tasks.

It helps in maintaining the desired state of the cluster.

kublet:

The kubelet is responsible for executing commands that come from the Kubernetes control plane, including instructions from the scheduler.

It communicates with the control plane and ensures that containers within pods are running as expected.

Kube-proxy:

Kube-proxy maintains network rules on nodes, allowing communication between Pods.

It handles network forwarding, load balancing, and service-related network tasks.

Container Engine:

The container engine on a worker node is responsible for executing and managing containers.

It provides the runtime environment for running containerized applications within the Kubernetes cluster, handling tasks such as container creation, start-up, and resource isolation.

POD:

A Pod is the smallest deployable unit in Kubernetes.

It's like wrapper around container.

The worker node executes and hosts these Pods, running the specified containers within them.

Lifecycle of kubernetes

A user or application initiates a request to create a new pod.

The API server, the central control point, communicates with ETCD, a persistent data store, to gather information about available nodes.

The scheduler, responsible for pod placement, analyzes resource availability and constraints to choose the best node for the pod.

The API server informs the kubelet, the agent running on the selected node, about the pod scheduling decision.

The kubelet creates the pod's containers and manages its lifecycle on that node.

The kubelet sends status updates back to the API server about the pod's progress.

The API server stores the updated cluster state information in ETCD, ensuring a consistent view for all components.

Kubectl Commands

Ref: https://www.bluematador.com/learn/kubectl-cheatsheet

kubectl cluster-info # to get cluster information

kubectl api-resources # to list available k8s objects

kubectl api-version # to list available api versions

kubectl get nodes # to get list of nodes

kubectl get nodes -o wide # to get IP of the nodes

kubectl get pods # to get list of pods

kubectl get pods -o wide # to get IP of the pods