CC Oral questions

Pods are the smallest and simplest unit in the Kubernetes system, which can be used to deploy and manage containerized applications.

- Pods can contain one or more containers and are used to provide an isolated environment for applications to run.
- Pods also provide a way to manage networking, storage, and other resources needed by the containers.

The purpose of pods is to facilitate the deployment and management of containerized applications in Kubernetes.

- Pods provide an abstraction layer between the containers and the underlying infrastructure, making it easier to manage and scale applications.

Five Linux commands with description:

- ls: used to list the contents of a directory
- cd: used to change the current working directory
- mkdir: used to create a new directory
- rm: used to remove files or directories
- cat: used to display the contents of a file

Stages of lifecycle in Terraform:

- Initialization: used to set up a new Terraform working directory
- Plan: used to generate an execution plan for the infrastructure changes
- Apply: used to apply the changes to the infrastructure
- Destroy: used to destroy the resources managed by Terraform
- Refresh: used to update the Terraform state file with the current state of the infrastructure.

Terraform and Ansible are two popular tools used for infrastructure automation, but they differ in their approach and application.

- Terraform is an infrastructure-as-code tool used to provision and manage infrastructure resources such as virtual machines, load balancers, and databases.
- Terraform focuses on defining and managing the state of infrastructure resources, making it ideal for cloud-based environments.

- Ansible is a configuration management tool used to automate software deployment, configuration, and management tasks.
- Ansible focuses on automating tasks on remote systems, making it ideal for managing a large number of servers in a data center or on-premise environment.

In summary, Terraform is used for infrastructure provisioning and management, while Ansible is used for software configuration and management.

Playbooks are used in Ansible for defining and executing a series of tasks and plays across a set of hosts.

- Playbooks allow for the automation of complex multi-step processes, allowing users to define and execute a series of tasks in a specific order.
- Playbooks can also include conditionals and loops, making it easier to manage and scale automation tasks across multiple hosts and environments.

There are two main **types of shell scripting**:

- **Interactive Shell Scripting**: This type of scripting involves writing and executing commands directly in the command-line interface of the shell, one command at a time. Interactive shell scripting is typically used for quick and ad-hoc tasks or for testing code snippets.
- **Batch Shell Scripting**: This type of scripting involves writing a series of commands in a file, which can then be executed as a script. Batch shell scripting is typically used for automating repetitive tasks or for creating more complex and sophisticated scripts that can run for extended periods of time.

The chmod instruction is used to change the permissions of a file or directory in a Linux or Unix-based operating system.

- It is used to set or modify the read, write, and execute permissions for the owner, group, and others who have access to the file or directory.
- This can help to control who can access, modify or execute a file or directory, which is an important aspect of system security and file management.

The ps command is a command-line utility used in Linux and Unix-based operating systems to display information about running processes.

- The ps command can display information such as the process ID (PID), the user who started the process, the status of the process, the amount of memory or CPU usage, and more.
- By default, the ps command displays information about the processes started by the current user, but it can also be used to display information about all processes on the system or about specific processes based on various criteria.

Ansible is an automation tool used to manage and configure systems, deploy applications, and orchestrate workflows.

- Ansible uses a clientless architecture, which means that it does not require any agents or software to be installed on the target system.
- Ansible uses SSH or WinRM to communicate with target systems, and uses YAML-based playbooks to define and execute automation tasks across a set of hosts or devices.
- Ansible modules are used to perform specific tasks such as managing packages, configuring network interfaces, or deploying applications.
- Ansible can be used to automate tasks across a wide range of systems, from small-scale applications to large-scale data center environments.

In a playbook, variables can be used to perform the following tasks:

- Define the values of variables that will be used in the playbook.
- Override the default values of variables defined in roles or playbooks.
- Pass variables as arguments to modules or other tasks within the playbook.
- Use conditional logic to control the execution of tasks based on the values of variables.
- Generate dynamic content based on the values of variables, such as configuration files or scripts.
- Manage the state of resources by using variables to define desired state, and then comparing current state to take appropriate action.
- Enable reuse and modularity of playbooks and roles by defining variables in a centralized location and then referencing them throughout the playbook.

The main standard syntax used in a YAML file includes:

- Indentation: YAML uses indentation to define the structure of data, instead of using braces or brackets as in other programming languages.
- Key-Value Pairs: YAML uses a key-value pair format to define data, where the key is separated from the value by a colon and a space (:).
- Lists and Arrays: YAML supports the use of lists and arrays, which are defined using square brackets ([ ]) and can contain a series of values separated by commas.
- Comments: YAML supports comments, which are preceded by a hash symbol (#) and are used to provide additional information or context about the data being defined.
- Scalars: YAML supports the use of scalar values, which can be strings, numbers, booleans, or null values.
- Block Style: YAML supports the use of block style notation, which allows for multi-line strings and nested structures to be defined in a more readable and organized manner.

Shell scripting syntax to add two numbers:

```
# Define variables
num1=5
num2=10

# Calculate the sum
sum=$(($num1 + $num2))

# Display the result
echo "The sum of $num1 and $num2 is $sum"
```

Docker ps command

- In Docker, the ps command is used to list the currently running containers along with their status, resource usage, and other information such as the container ID, image name, and command that was used to start the container.
- The docker ps command is similar to the ps command in Linux, but it is specific to Docker containers. By default, it only shows the running containers, but you can use the -a option to show all containers (both running and stopped).

Why we use only yaml for ansible

Why not other language

- Ansible uses YAML as its primary language for defining automation workflows and playbooks because it is easy to read, write, and maintain. YAML is a human-readable data serialization format that allows for complex data structures to be represented in a simple and concise manner.
- Additionally, YAML is a standardized format that is supported by many programming languages, including Python, which is the language that Ansible is written in. This makes it easy for developers and system administrators who are familiar with Python to use Ansible and write playbooks using YAML.
- While it is technically possible to use other programming languages to define automation workflows in Ansible, using YAML provides several benefits, including readability, maintainability, and compatibility with other Ansible modules and plugins. It also allows Ansible to be easily

integrated into existing infrastructure and workflows, making it a popular choice for configuration management and automation tasks.

Amazon EC2 and Amazon S3 are two different services provided by Amazon Web Services (AWS) that serve different purposes.

- Amazon EC2 (Elastic Compute Cloud) is a web service that provides resizable compute capacity in the cloud. It allows users to launch and manage virtual machines, called instances, on demand. EC2 instances are commonly used for running applications, hosting websites, and processing data.
- Amazon S3 (Simple Storage Service) is a cloud-based object storage service that provides scalable and durable storage for data. S3 can store and retrieve any amount of data from anywhere on the web, making it ideal for storing and distributing static assets such as images, videos, and backups.

Here are some key differences between EC2 and S3:

- Use case: EC2 is typically used for running applications and processing data, while S3 is used for storing and distributing data.
- Data type: EC2 instances store data on their local disk or on attached storage volumes, while S3 is designed for storing and retrieving objects, such as files or data sets.
- Scaling: EC2 instances can be scaled up or down as needed, while S3 storage capacity is automatically scaled as data is added or removed.
- Pricing: EC2 pricing is based on the instance type, operating system, and usage, while S3 pricing is based on the amount of data stored and data transfer.

In summary, use Amazon EC2 when you need to run applications, host websites, or process data, and use Amazon S3 when you need scalable, durable storage for data, such as backups, images, and videos.

AWS provides several security policies and best practices that can help you secure your infrastructure and data. Here are some key policies you can implement to enhance your AWS security:

- Identity and Access Management (IAM) policies: IAM allows you to manage users and their access to AWS resources. You can create IAM policies that define what actions a user or group can perform on specific resources. Use IAM to enforce least privilege access, which means granting users only the permissions they need to perform their jobs.
- Network security policies: Use security groups and network access control lists (ACLs) to control access to your AWS resources. Security groups act as a virtual firewall that controls inbound and outbound traffic, while ACLs control access at the subnet level.
- Encryption policies: AWS offers several encryption options, such as server-side encryption, client-side encryption, and Key Management Service (KMS). Use encryption to protect sensitive data at rest and in transit.
- Compliance policies: AWS provides several compliance frameworks, such as HIPAA, PCI DSS, and SOC 2. Use these frameworks as guidelines to ensure that your infrastructure and applications are compliant with industry regulations.
- Monitoring policies: Use AWS services such as CloudTrail, CloudWatch, and GuardDuty to monitor your infrastructure and detect potential security threats. These services can provide insights into your infrastructure's security posture and help you identify security issues.
- Disaster recovery policies: Implement backup and recovery policies to ensure that your data is protected in the event of a disaster. Use AWS services such as S3, EBS, and Glacier to store backups and implement a disaster recovery plan.

In summary, AWS provides several security policies and best practices that can help you secure your infrastructure and data. Use IAM policies, network security policies, encryption policies, compliance policies, monitoring policies, and disaster recovery policies to ensure that your AWS infrastructure is secure and resilient.

1.Difference between virtual machine & containerization

2. What are containerization use cases?

3. What is role of Docker Daemon

4. What is Docker Image

5. Difference between exec and run command.

6. What programing language does dockerfile use?

- Virtual machines emulate a complete hardware environment, while containerization virtualizes the operating system, enabling multiple applications to run on the same OS kernel.
- Containerization use cases include application deployment, microservices, continuous integration and deployment, and development and testing environments.
- Docker Daemon is responsible for managing Docker objects such as images, containers, networks, and volumes.
- A Docker image is a read-only template that contains a set of instructions for creating a Docker container.
- The "run" command creates a new container and starts it, while the "exec" command runs a new command in an existing container.
- Dockerfile uses its own domain-specific language for defining instructions and building Docker images.

Terraform is an open-source infrastructure as code (IaC) tool that enables you to define and manage cloud infrastructure using a declarative configuration language. Terraform supports a wide range of cloud providers, including AWS, Azure, and Google Cloud Platform.

- The use of Terraform in AWS is to manage AWS resources using Infrastructure as Code (IaC) principles. You can use Terraform to create and manage AWS resources, such as EC2 instances, VPCs, security groups, and load balancers, in a repeatable and automated manner.
- Compared to Ansible, which is a configuration management tool, Terraform focuses more on provisioning infrastructure and managing resource dependencies. While Ansible can also be used for provisioning, it is primarily designed for configuring systems after they have been provisioned.

Here are some key differences between Terraform and Ansible:

- Declarative vs imperative: Terraform is declarative, which means that you define the desired state of your infrastructure, and Terraform handles the details of creating and managing resources to achieve that state. Ansible is imperative, which means that you define the steps required to configure a system, and Ansible executes those steps in order.
- Resource types: Terraform supports a wide range of resource types for different cloud providers, while Ansible is primarily focused on managing software and system configurations.
- State management: Terraform manages the state of your infrastructure, which allows it to determine the changes that need to be made to achieve the desired state. Ansible does not manage state and relies on idempotent execution to ensure that configurations are applied consistently.
- Complexity: Terraform can be more complex to learn and use than Ansible, particularly when managing large and complex infrastructures. Ansible is generally easier to get started with and can be used for simpler use cases.

In summary, Terraform is an IaC tool that enables you to define and manage cloud infrastructure using a declarative configuration language, while Ansible is a configuration management tool that is primarily focused on managing software and system configurations. The key differences between Terraform and Ansible include their declarative vs imperative approaches, resource types, state management, and complexity.

The architecture of Terraform includes the following components and terms:

- Configuration files: Declarative configuration files written in HashiCorp Configuration Language (HCL) or JSON format that define the desired state of infrastructure.
- Terraform CLI: The command-line interface (CLI) used to interact with Terraform, including commands for initializing, planning, applying, and destroying infrastructure.
- Terraform state: A file that contains the current state of infrastructure managed by Terraform, including resource mappings, dependencies, and metadata.
- Providers: Plugins that interface with different cloud providers, such as AWS, Azure, and Google Cloud Platform, to provision and manage infrastructure resources.

- Resource types: Configurable infrastructure components, such as virtual machines, databases, and networking components, defined by provider plugins.
- Modules: Reusable units of infrastructure code that encapsulate related resources and dependencies, enabling modular and scalable infrastructure design.
- Backends: A remote storage location for Terraform state, such as S3 or Consul, that enables collaboration and state management across teams and environments.

In summary, the architecture of Terraform includes configuration files, the Terraform CLI, Terraform state, providers, resource types, modules, and backends, which work together to define, provision, and manage infrastructure as code.

In YAML, you can create a list by using square brackets [] and separating each item in the list with a comma. Here is an example:

```
fruits:
  - apple
  - banana
  - orange
```

In this example, a list of fruits is defined using the key fruits followed by the list of values enclosed in square brackets. Each item in the list is separated by a comma. You can also use YAML syntax to create nested lists and lists of objects.