M A M A T H A

1)

1 Provide steps to create a directory inside a directory where the parent directory does not exist.

mkdir /path/to/parentdirectory/subdirectory

2 How to install a package on a Linux server when there is no internet connection?

Download the Package

* On a system with internet access, download the package and all its dependencies to a local directory.

yum - download the package along with dependencies

Install the Package

* On the offline server, navigate to the directory where the package files are located and use the package manager to install the package.

sudo yum install package-name.rpm

3 How to access specific folders of Server A from Server B and Server C?

SSH:

scp user@serverA:/path/to/source/file user@serverB:/path/to/destination/directory

4 How to check all the running processes from a server?

ps aux

5 Provide the command to delete all the files older than X days inside a specific directory.

find /path/to/directory -type f -mtime +X -delete

6 Create a shell script to identify the process ID a. script should as a user input for process ID b. If the process exists script should print the process ID and exit c. If the process doesn't exist script should print the process doesn't exist and asks for another input

#!/bin/bash

while true; do

# Prompt user for input

read -p "Enter the process ID: " pid

# Check if the process exists

if ps -p "$pid" > /dev/null 2>&1; then

echo "Process $pid exists."

exit 0

else

echo "Process $pid doesn't exist. Please enter another process ID."

fi

done

2)

1 What is docker and why do we need it?

* Docker is a platform for packaging, distributing, and running applications within containers, providing consistency across different environments, isolation for applications and their dependencies, portability across various platforms, efficiency in resource utilization, scalability for application deployment, and facilitation of DevOps practices through streamlined deployment pipelines, making it essential for modern software development and deployment workflows.

2 Write docker file for a sample Java/python application.

Java

FROM openjdk:11

WORKDIR /usr/src/app

COPY ./target/my-java-app.jar .

CMD ["java", "-jar", "my-java-app.jar"]

Python

FROM python:3.9

WORKDIR /usr/src/app

COPY . .

RUN pip install --no-cache-dir -r requirements.txt

CMD ["python", "app.py"]

3 what is the docker lifecycle?

Building Images 🡺 Running Containers 🡺 Stopping Containers 🡺 Restarting Containers 🡺 Removing Containers and Images 🡺 Pushing and Pulling Images

Building Images

* The lifecycle starts with building Docker images using Dockerfiles or pulling existing images from a registry. During this stage, Docker fetches the necessary files and dependencies and creates a snapshot of the filesystem to form the image.

Running Containers

* Once an image is built, it can be instantiated into a container using the docker run command. Containers are instances of images that run as isolated processes with their own filesystem, networking, and resources.

Stopping Containers

* Containers can be stopped either manually using the docker stop command or automatically when they complete their tasks or encounter errors. Stopping a container halts its execution but preserves its state and data.

Restarting Containers

* Containers can be restarted after being stopped, either manually or automatically. Docker provides options for configuring container restart policies to ensure high availability and fault tolerance.

Removing Containers and Images

* Containers and images can be removed using the docker rm and docker rmi commands, respectively. Removing a container deletes its filesystem and any data associated with it, while removing an image deletes its filesystem layers and metadata.

Pushing and Pulling Images

* Images can be pushed to and pulled from Docker registries such as Docker Hub or private registries. Pushing an image uploads it to a registry, making it accessible to other users and systems. Pulling an image download, it from a registry to the local system.

Managing Networks and Volumes

* Docker allows for the creation and management of networks and volumes to facilitate communication between containers and persistent storage. Networks enable containers to communicate with each other, while volumes provide durable storage for container data.

4 What is the difference between an image and a container?

* An image is a static snapshot of a filesystem that includes the application code, libraries, and dependencies needed to run an application. It serves as a blueprint for creating containers.
* A container, on the other hand, is a lightweight, executable package that includes everything needed to run a piece of software, including the application code, runtime, system tools, system libraries, and settings.
* Containers are instances of images that are running in an isolated environment with their own CPU, memory, storage, and network resources, allowing for consistent and efficient deployment across different computing environments.
* In essence, while an image is a static representation of an application's dependencies and code, a container is a runtime instance of that image, providing encapsulation and portability for applications.

5 How to check Docker container logs

docker logs -f my\_container

This command will continuously stream the logs of the container named

3)

1 What are different types of services in Kubernetes?

* ClusterIP Service
* NodePort Service
* LoadBalancer Service
* ExternalName Service
* Headless Service

2 what is POD?

* A Pod is the smallest deployable unit in Kubernetes, representing one or more containers that are tightly coupled and share resources such as networking and storage.
* Pods encapsulate an application's containers, storage resources, a unique network IP, and configuration options. They are the basic building blocks for deploying, scaling, and managing applications in Kubernetes clusters.

3 Create a pod with the above created custom image when a pod dies k8s should automatically restart

Yaml:

apiVersion: v1

kind: Pod

metadata:

name: my-custom-pod

spec:

containers:

- name: my-container

image: my-image:latest

restartPolicy: Always

command:

kubectl apply -f custom-pod.yaml

4 How to access the custom application with a specific port?

* In the address bar, we’ve to type the URL of the application followed by a colon and the port number.

http://domain.com:8080

or

http://localhost:3000