Day 3:

MINIKUBE:

Minikube is an open-source tool that allows you to run a single-node Kubernetes cluster locally on your machine. It’s a great option for developers and learners who want to experiment with Kubernetes without needing a full-fledged cloud environment.

Purpose: Minikube is primarily used for learning Kubernetes concepts, testing applications locally, and developing on Kubernetes.

Ease of Setup: Minikube simplifies running Kubernetes by creating a lightweight virtual machine or container that contains the Kubernetes environment.

Features:

Supports Kubernetes add-ons (e.g., ingress, metrics-server, and dashboard).

Offers multi-cluster support for testing multiple Kubernetes clusters simultaneously.

Provides a built-in Docker daemon, eliminating the need for separate Docker installations.

Allows configuration of resource limits like CPU and memory.

Cross-Platform: It works on various operating systems, including Windows, macOS, and Linux.

Use Cases:

Learning Kubernetes basics in a local environment.

Testing CI/CD pipelines and Kubernetes deployments.

Debugging Kubernetes-related issues.

Integration: Minikube integrates well with Kubernetes CLI tools like kubectl

MINIKUBE INSTALLATION:

Download Minikubecurl -LO https://github.com/kubernetes/minikube/releases/latest/download/minikube-linux-amd64

```
# Install Minikube

sudo install minikube-linux-amd64 /usr/local/bin/minikube && rm minikube-linux-amd64



# Start Minikube

minikube start



# Check Minikube status

minikube status



# Get running pods

kubectl get pod



# Get deployments

kubectl get deploy



# Get replicas

kubectl get replica



# Get detailed pod information

kubectl get pod -o wide
```

DOCKER COMPOSE:

Docker Compose is a tool that allows you to define and manage multi-container Docker applications. It simplifies the process of running multiple containers, their configurations, and their interdependencies. Compose uses a YAML file to define the services, networks, and volumes required for your application.

Docker Compose is a tool which is used to manage multi container-based applications.

Using Docker Compose we can easily setup & deploy multi container-based applications.

We will give containers information to Docker Compose using YML file (docker-compose.yml)

Docker Compose YML should have all the information related to containers creation.

Docker Compose YML File Looks Like:

download docker compose

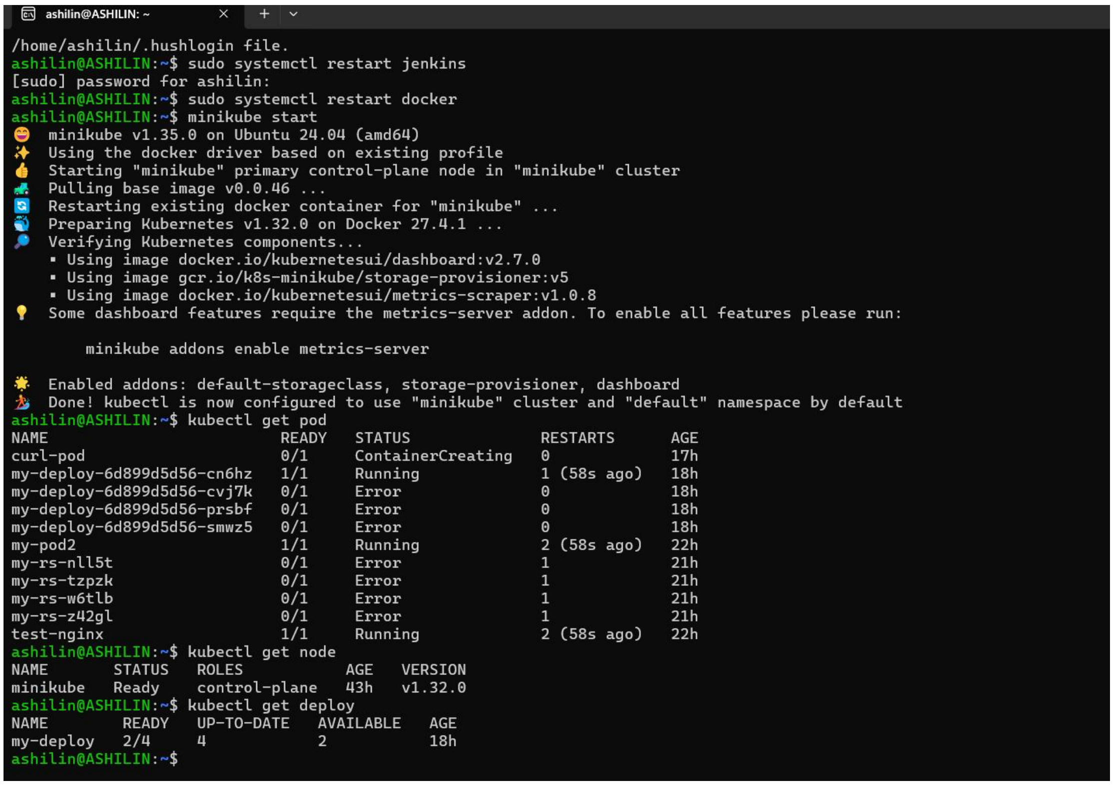
```
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

DOCKER COMPOSE COMMANDS:

```
# Install Docker Compose

sudo apt install docker-compose -y



# Download the latest version of Docker Compose

sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose



# Make Docker Compose executable

sudo chmod +x /usr/local/bin/docker-compose



# Check Docker Compose version

docker-compose --version



# Example docker-compose.yml file

version: '3'
```

```
services:
  web:
    image: nginx:latest
    ports:
      - 80:80

  db:
    image: mysql:latest
    environment:
      - MYSQL_ROOT_PASSWORD=secret

# Start services using Docker Compose
docker-compose up -d

# Execute a shell inside the database container
docker exec -it david-db-1 /bin/bash

# Access MySQL inside the container
mysql -u root -p
 version: '3'
services:
  web:
    image: nginx:latest
    ports:
```

```
    - 80:80

db:

  image: mysql:latest

  environment:

    - MYSQL_ROOT_PASSWORD=secret

docker exec -it david-db-1 /bin/bash

 mysql -u root -p
```



```
/home/ashilin/.hushlogin file.
ashilin@ASHILIN:~$ sudo systemctl restart jenkins
[sudo] password for ashilin:
ashilin@ASHILIN:~$ sudo systemctl restart docker
ashilin@ASHILIN:~$ minikube start
😄  minikube v1.35.0 on Ubuntu 24.04 (amd64)
✨  Using the docker driver based on existing profile
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.46 ...
🔄  Restarting existing docker container for "minikube" ...
🐳  Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
🔎  Verifying Kubernetes components...
    ▪ Using image docker.io/kubernetesui/dashboard:v2.7.0
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
    ▪ Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
💡  Some dashboard features require the metrics-server addon. To enable all features please run:

        minikube addons enable metrics-server

🌟  Enabled addons: default-storageclass, storage-provisioner, dashboard
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
ashilin@ASHILIN:~$ kubectl get pod
NAME                          READY   STATUS              RESTARTS      AGE
curl-pod                      0/1     ContainerCreating   0             17h
my-deploy-6d899d5d56-cn6hz    1/1     Running             1 (58s ago)   18h
my-deploy-6d899d5d56-cvj7k    0/1     Error               0             18h
my-deploy-6d899d5d56-prsbf    0/1     Error               0             18h
my-deploy-6d899d5d56-smwz5    0/1     Error               0             18h
my-pod2                       1/1     Running             2 (58s ago)   22h
my-rs-nll5t                   0/1     Error               1             21h
my-rs-tzpzk                   0/1     Error               1             21h
my-rs-w6tlb                   0/1     Error               1             21h
my-rs-z42gl                   0/1     Error               1             21h
test-nginx                    1/1     Running             2 (58s ago)   22h
ashilin@ASHILIN:~$ kubectl get node
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   43h   v1.32.0
ashilin@ASHILIN:~$ kubectl get deploy
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
my-deploy   2/4     4            2           18h
ashilin@ASHILIN:~$
```