

Project 01 - 1 Hour

Deploying a Scalable Web Application with Persistent Storage and Advanced Automation

Objective:

Deploy a scalable web application using Docker Swarm and Kubernetes, ensuring data persistence using a single shared volume, and automate the process using advanced shell scripting.

Overview:

1. **Step 1:** Set up Docker Swarm and create a service.
 2. **Step 2:** Set up Kubernetes using Minikube.
 3. **Step 3:** Deploy a web application using Docker Compose.
 4. **Step 4:** Use a single shared volume across multiple containers.
 5. **Step 5:** Automate the entire process using advanced shell scripting.
-

Step 1: Set up Docker Swarm and Create a Service

1.1 Initialize Docker Swarm

```
# Initialize Docker Swarm
docker swarm init
```

1.2 Create a Docker Swarm Service

```
# Create a simple Nginx service in Docker Swarm
docker service create --name nginx-service --publish 8080:80 nginx
```

```
einfochips@AHMLPT1474: ~/Day6
einfochips@AHMLPT1474:~/Day6$ sudo docker swarm init
Swarm initialized: current node (8p1fosf53vzc0pozmkfa8fgks) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-1ne8tnjudxl8rah8kvwee61ih7p0bf9j0kehkycn5
cfbr9uhan-3cpr2k9a8pxpaouc8rku5j0bz 192.168.232.24:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow
the instructions.

einfochips@AHMLPT1474:~/Day6$ sudo dokcer service create --name nginx-service --
publish 8080:80 nginx
sudo: dokcer: command not found
einfochips@AHMLPT1474:~/Day6$ sudo docker service create --name nginx-service --
publish 8080:80 nginx
nk1egja7zwdzszm7t1vinru18
overall progress: 1 out of 1 tasks
1/1: running
verify: Waiting 1 seconds to verify that tasks are stable...
█
```

Step 2: Set up Kubernetes Using Minikube

2.1 Start Minikube

```
# Start Minikube
minikube start
```

```
einfochips@AHMLPT1474: ~/Day6
PATH
  ■ podman: Not installed: exec: "podman": executable file not found in $PATH
  ■ virtualbox: Not installed: unable to find VBoxManage in $PATH

❌ Exiting due to DRV_NOT_HEALTHY: Found driver(s) but none were healthy. See above f
or suggestions how to fix installed drivers.

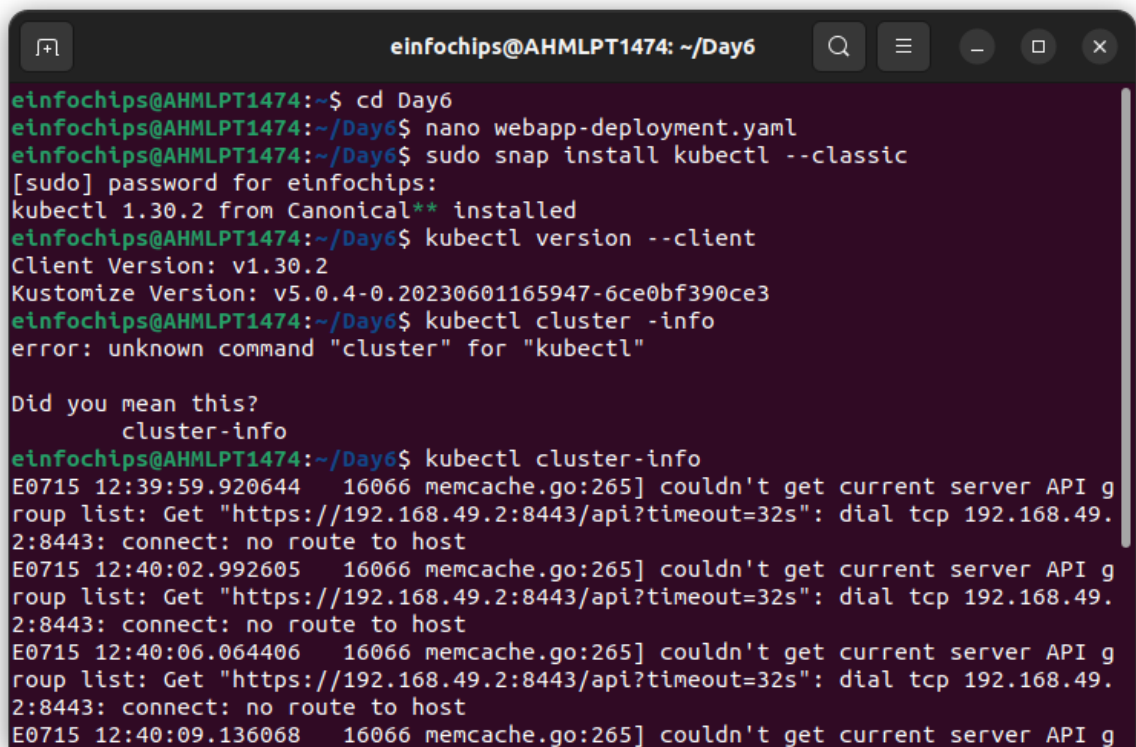
einfochips@AHMLPT1474:~/Day6$ sudo usermod -aG docker $USER && newgrp docker
einfochips@AHMLPT1474:~/Day6$ minikube start
🐹 minikube v1.33.1 on Ubuntu 22.04
🔧 Automatically selected the docker driver. Other choices: ssh, none
🔧 Using Docker driver with root privileges
👍 Starting "minikube" primary control-plane node in "minikube" cluster
📦 Pulling base image v0.0.44 ...
📦 Downloading Kubernetes v1.30.0 preload ...
  > preloaded-images-k8s-v18-v1...: 342.90 MiB / 342.90 MiB 100.00% 156.66
  > gcr.io/k8s-minikube/kicbase...: 481.58 MiB / 481.58 MiB 100.00% 205.27
🔥 Creating docker container (CPUs=2, Memory=3900MB) ...
🔧 Preparing Kubernetes v1.30.0 on Docker 26.1.1 ...
  ■ Generating certificates and keys ...
  ■ Booting up control plane ...
  ■ Configuring RBAC rules ...
🔗 Configuring bridge CNI (Container Networking Interface) ...
🔍 Verifying Kubernetes components...
  ■ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟 Enabled addons: storage-provisioner, default-storageclass
💡 kubectl not found. If you need it, try: 'minikube kubectl -- get pods -A'
🏠 Done! kubectl is now configured to use "minikube" cluster and "default" namespace
by default
einfochips@AHMLPT1474:~/Day6$
```

2.2 Deploy a Web App on Kubernetes

Create a deployment file named `webapp-deployment.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapp
spec:
  replicas: 3
  selector:
    matchLabels:
      app: webapp
  template:
    metadata:
      labels:
        app: webapp
    spec:
      containers:
        - name: webapp
          image: nginx
      ports:
```

- containerPort: 80

A terminal window titled 'einfochips@AHMLPT1474: ~/Day6' with standard window controls. The terminal shows the following commands and output:

```
einfochips@AHMLPT1474:~$ cd Day6
einfochips@AHMLPT1474:~/Day6$ nano webapp-deployment.yaml
einfochips@AHMLPT1474:~/Day6$ sudo snap install kubectl --classic
[sudo] password for einfochips:
kubectl 1.30.2 from Canonical** installed
einfochips@AHMLPT1474:~/Day6$ kubectl version --client
Client Version: v1.30.2
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
einfochips@AHMLPT1474:~/Day6$ kubectl cluster -info
error: unknown command "cluster" for "kubectl"

Did you mean this?
    cluster-info
einfochips@AHMLPT1474:~/Day6$ kubectl cluster-info
E0715 12:39:59.920644    16066 memcache.go:265] couldn't get current server API group list: Get "https://192.168.49.2:8443/api?timeout=32s": dial tcp 192.168.49.2:8443: connect: no route to host
E0715 12:40:02.992605    16066 memcache.go:265] couldn't get current server API group list: Get "https://192.168.49.2:8443/api?timeout=32s": dial tcp 192.168.49.2:8443: connect: no route to host
E0715 12:40:06.064406    16066 memcache.go:265] couldn't get current server API group list: Get "https://192.168.49.2:8443/api?timeout=32s": dial tcp 192.168.49.2:8443: connect: no route to host
E0715 12:40:09.136068    16066 memcache.go:265] couldn't get current server API group list: Get "https://192.168.49.2:8443/api?timeout=32s": dial tcp 192.168.49.2:8443: connect: no route to host
```

Apply the deployment:

`kubectl apply -f webapp-deployment.yaml`

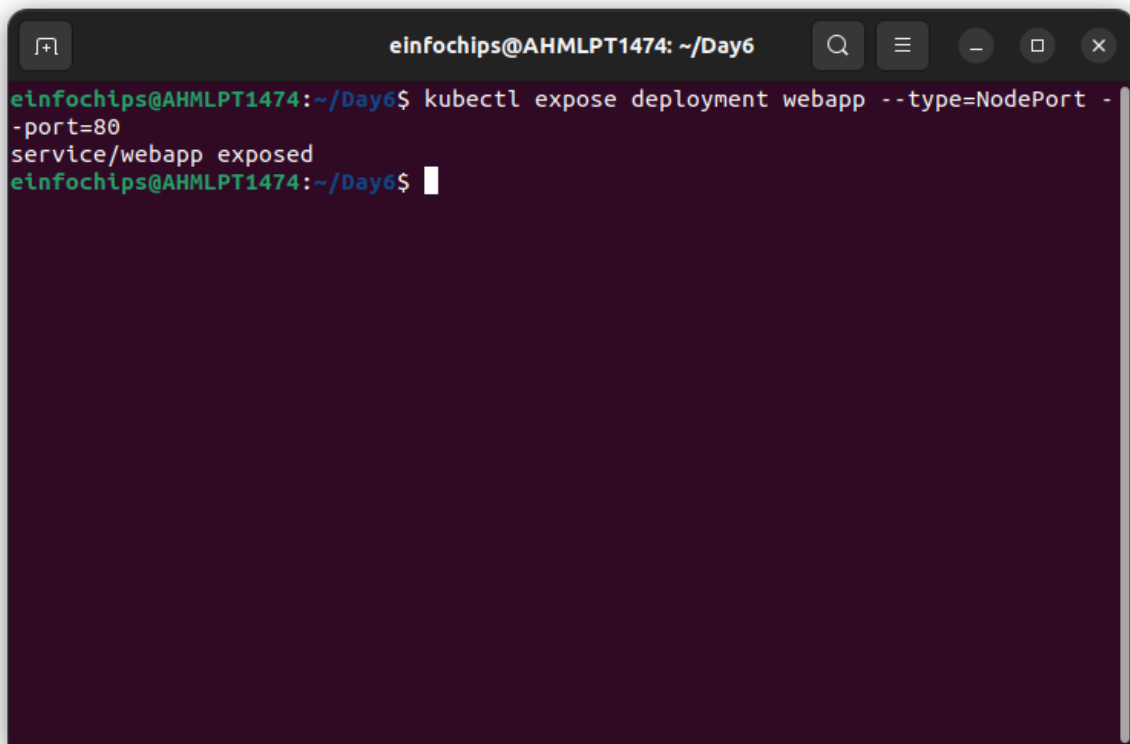
```
einfochips@AHMLPT1474: ~/Day6
einfochips@AHMLPT1474:~/Day6$ kubectl apply -f webapp-deployment.yaml
error: error validating "webapp-deployment.yaml": error validating data: failed
to download openapi: Get "https://192.168.49.2:8443/openapi/v2?timeout=32s": dial
 tcp 192.168.49.2:8443: connect: no route to host; if you choose to ignore thes
e errors, turn validation off with --validate=false
einfochips@AHMLPT1474:~/Day6$ sudo spt-get install virtualbox virtualbox-ext-pac
k
sudo: spt-get: command not found
einfochips@AHMLPT1474:~/Day6$ sudo apt-get install virtualbox virtualbox-ext-pac
k
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package virtualbox
E: Unable to locate package virtualbox-ext-pack
einfochips@AHMLPT1474:~/Day6$ kubectl apply -f webapp-deployment.yaml
error: error validating "webapp-deployment.yaml": error validating data: failed
to download openapi: Get "https://192.168.49.2:8443/openapi/v2?timeout=32s": dial
 tcp 192.168.49.2:8443: connect: no route to host; if you choose to ignore thes
e errors, turn validation off with --validate=false
einfochips@AHMLPT1474:~/Day6$
```

```
einfochips@AHMLPT1474: ~/Day6
{
  "kind": "ReplicaSetList",
  "apiVersion": "apps/v1",
  "metadata": {
    "resourceVersion": "626"
  },
  "items": []
}
{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {
    "resourceVersion": "626"
  },
  "items": []
}
einfochips@AHMLPT1474:~/Day6$ kubectl cluster-info
Kubernetes control plane is running at https://192.168.49.2:8443
CoreDNS is running at https://192.168.49.2:8443/api/v1/namespaces/kube-system/se
rvices/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
einfochips@AHMLPT1474:~/Day6$ kubectl apply -f webapp-deployment.yaml
deployment.apps/webapp created
einfochips@AHMLPT1474:~/Day6$
```

2.3 Expose the Deployment

`kubectl expose deployment webapp --type=NodePort --port=80`

A terminal window with a dark background and light green text. The window title is 'einfochips@AHMLPT1474: ~/Day6'. The command 'kubectl expose deployment webapp --type=NodePort -p=80' has been executed, resulting in the output 'service/webapp exposed'. The prompt 'einfochips@AHMLPT1474:~/Day6\$' is visible at the bottom.

```
einfochips@AHMLPT1474: ~/Day6
einfochips@AHMLPT1474:~/Day6$ kubectl expose deployment webapp --type=NodePort -p=80
service/webapp exposed
einfochips@AHMLPT1474:~/Day6$
```

Step 3: Deploy a Web Application Using Docker Compose

3.1 Create a `docker-compose.yml` File

```
version: '3'
services:
  web:
    image: nginx
    ports:
      - "8080:80"
    volumes:
      - webdata:/usr/share/nginx/html

volumes:
  webdata:
```

3.2 Deploy the Web Application

```
# Deploy using Docker Compose
docker-compose up -d
```

Step 4: Use a Single Shared Volume Across Multiple Containers

4.1 Update `docker-compose.yml` to Use a Shared Volume

```
version: '3'
services:
  web1:
    image: nginx
    ports:
      - "8081:80"
    volumes:
      - shareddata:/usr/share/nginx/html
  web2:
    image: nginx
    ports:
      - "8082:80"
    volumes:
      - shareddata:/usr/share/nginx/html

volumes:
  shareddata:
```

4.2 Deploy with Docker Compose

```
# Deploy using Docker Compose
docker-compose up -d
```

Step 5: Automate the Entire Process Using Advanced Shell Scripting

5.1 Create a Shell Script `deploy.sh`

```
#!/bin/bash

# Initialize Docker Swarm
docker swarm init

# Create Docker Swarm Service
docker service create --name nginx-service --publish 8080:80 nginx

# Start Minikube
minikube start

# Create Kubernetes Deployment
kubectl apply -f webapp-deployment.yaml

# Expose the Deployment
kubectl expose deployment webapp --type=NodePort --port=80
```

```
# Deploy Web App Using Docker Compose
docker-compose -f docker-compose-single-volume.yml up -d

echo "Deployment completed successfully!"
```

5.2 Make the Script Executable

```
# Make the script executable
chmod +x deploy.sh
```

5.3 Run the Script

```
# Run the deployment script
./deploy.sh
```