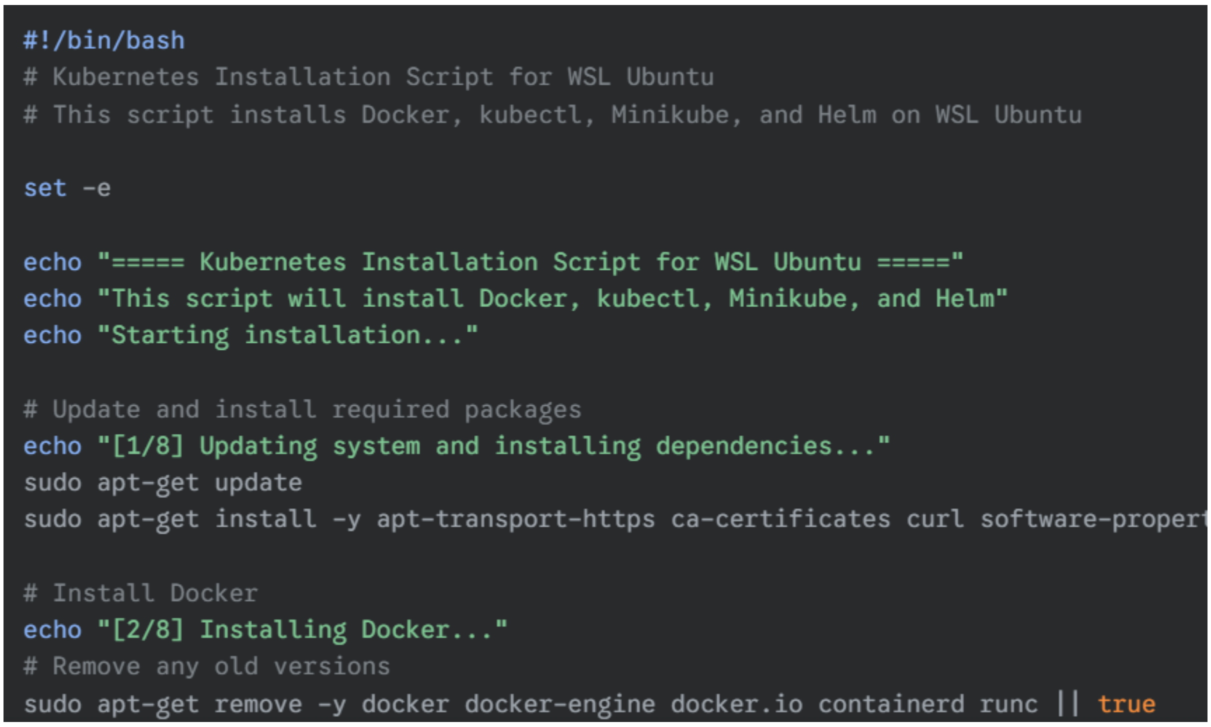
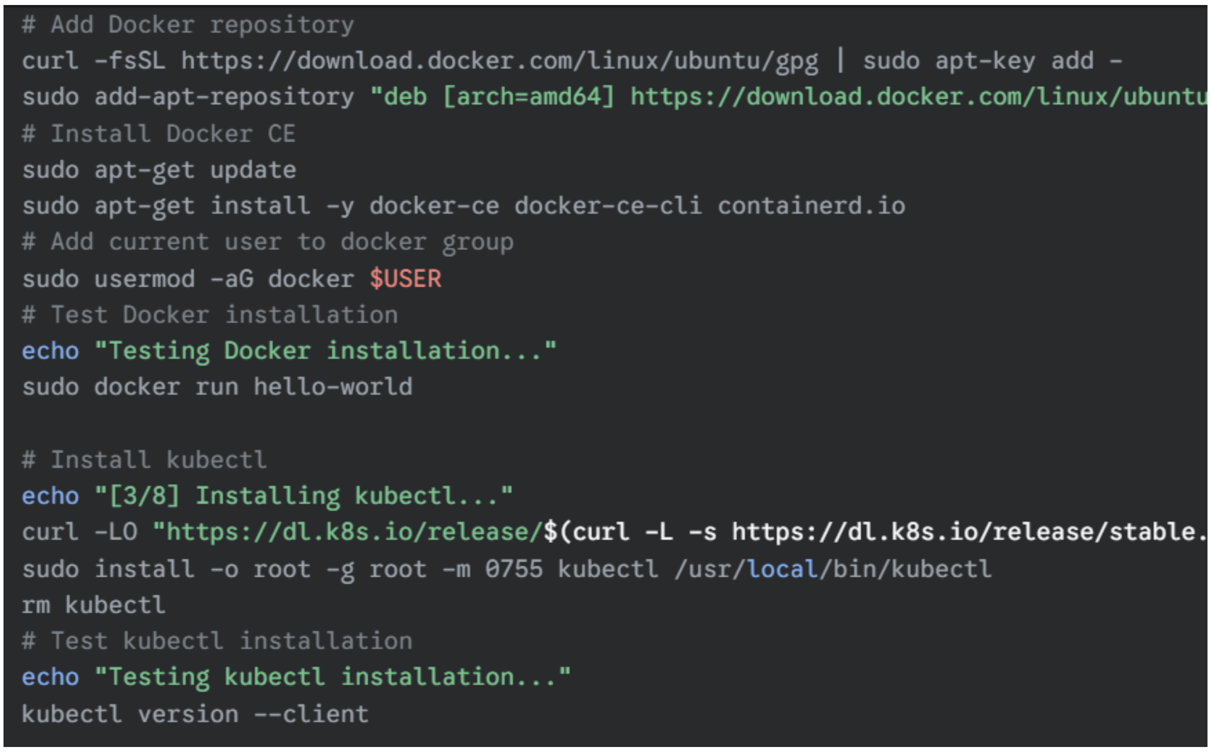
Kubernetics:

Installation:



```bash
#!/bin/bash
# Kubernetes Installation Script for WSL Ubuntu
# This script installs Docker, kubectl, Minikube, and Helm on WSL Ubuntu

set -e

echo "===== Kubernetes Installation Script for WSL Ubuntu ====="
echo "This script will install Docker, kubectl, Minikube, and Helm"
echo "Starting installation..."

# Update and install required packages
echo "[1/8] Updating system and installing dependencies..."
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl software-propert

# Install Docker
echo "[2/8] Installing Docker..."
# Remove any old versions
sudo apt-get remove -y docker docker-engine docker.io containerd runc || true
```



```bash
# Add Docker repository
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
# Install Docker CE
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io
# Add current user to docker group
sudo usermod -aG docker $USER
# Test Docker installation
echo "Testing Docker installation..."
sudo docker run hello-world

# Install kubectl
echo "[3/8] Installing kubectl..."
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
rm kubectl
# Test kubectl installation
echo "Testing kubectl installation..."
kubectl version --client
```

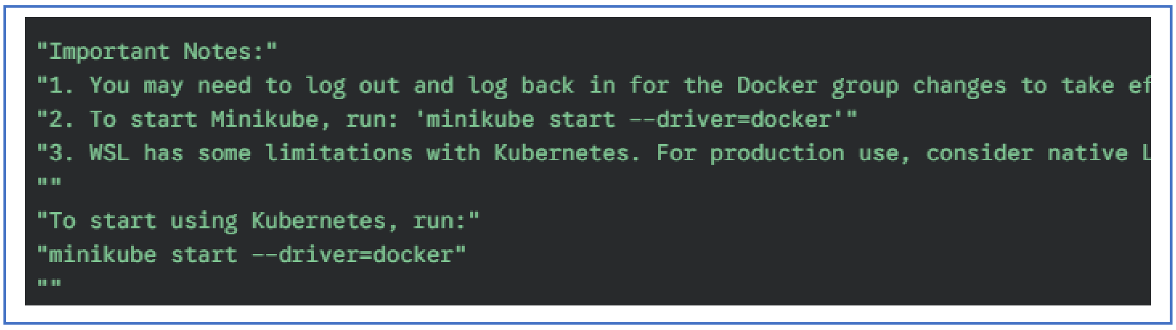
```bash
# Install Minikube
echo "[4/8] Installing Minikube..."
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-am
sudo install minikube-linux-amd64 /usr/local/bin/minikube
rm minikube-linux-amd64

# Install Helm
echo "[5/8] Installing Helm..."
curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash

# Configure Docker to start on WSL startup
echo "[6/8] Configuring Docker to start on WSL startup..."
echo '# Start Docker daemon automatically when WSL starts
if [ -z "$(ps -ef | grep dockerd | grep -v grep)" ]; then
    sudo service docker start
fi' >> ~/.bashrc

# Create directory for Kubernetes config
echo "[7/8] Creating Kubernetes config directory..."
mkdir -p ~/.kube
```

```bash
# Final setup and verification
echo "[8/8] Performing final setup and verification..."
echo '# Kubernetes aliases
alias k="kubectl"
alias kgp="kubectl get pods"
alias kgs="kubectl get services"
alias kgd="kubectl get deployments"
alias kgn="kubectl get nodes"
alias kga="kubectl get all"' >> ~/.bashrc

# Print installation summary
echo ""
echo "===== Installation Complete! ====="
echo "Docker version:"
docker --version
echo "kubectl version:"
kubectl version --client
echo "Minikube version:"
minikube version
echo "Helm version:"
helm version
```

```bash
echo ""
echo "Important Notes:"
echo "1. You may need to log out and log back in for the Docker group changes to t
echo "2. To start Minikube, run: 'minikube start --driver=docker'"
echo "3. WSL has some limitations with Kubernetes. For production use, consider na
echo ""
echo "To start using Kubernetes, run:"
echo "minikube start --driver=docker"
echo ""
echo "===== Happy Kuberneting! ====="
```

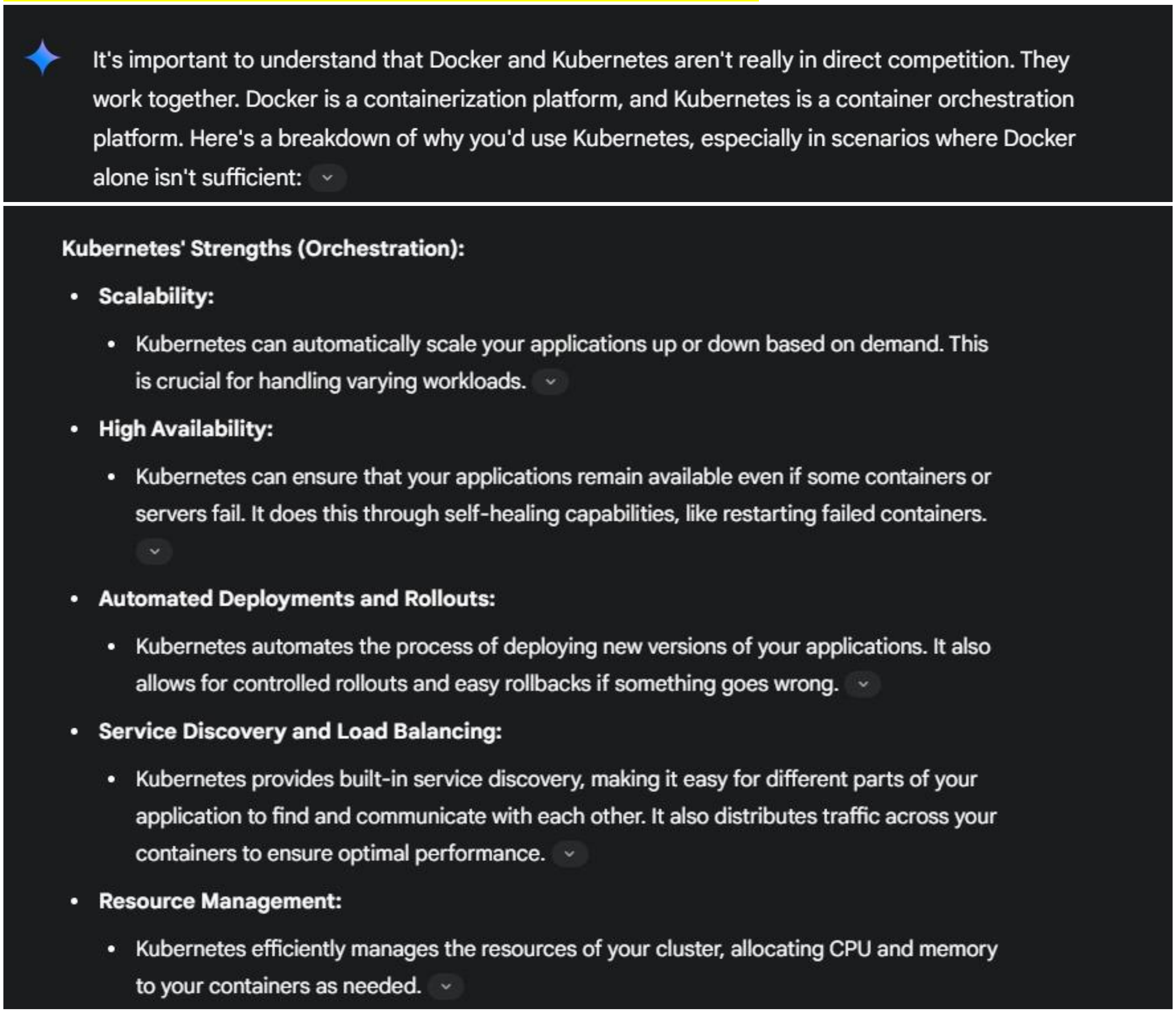
```
#!/bin/bash

# Kubernetes Installation Script for WSL Ubuntu

# This script installs Docker, kubectl, Minikube, and Helm on WSL Ubuntu

set -e

echo "===== Kubernetes Installation Script for WSL Ubuntu ====="

echo "This script will install Docker, kubectl, Minikube, and Helm"

echo "Starting installation..."

# Update and install required packages

echo "[1/8] Updating system and installing dependencies..."

sudo apt-get update

sudo apt-get install -y apt-transport-https ca-certificates curl software-properties-common
gnupg2 conntrack

# Install Docker

echo "[2/8] Installing Docker..."

# Remove any old versions

sudo apt-get remove -y docker docker-engine docker.io containerd runc || true

# Add Docker repository

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable"

# Install Docker CE

sudo apt-get update

sudo apt-get install -y docker-ce docker-ce-cli containerd.io

# Add current user to docker group

sudo usermod -aG docker $USER

# Test Docker installation

echo "Testing Docker installation..."

sudo docker run hello-world

# Install kubectl

echo "[3/8] Installing kubectl..."

curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

```
rm kubectl

# Test kubectl installation

echo "Testing kubectl installation..."

kubectl version --client

# Install Minikube

echo "[4/8] Installing Minikube..."

curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64

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

rm minikube-linux-amd64

# Install Helm

echo "[5/8] Installing Helm..."

curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash

# Configure Docker to start on WSL startup

echo "[6/8] Configuring Docker to start on WSL startup..."

echo '# Start Docker daemon automatically when WSL starts

if [ -z "$(ps -ef | grep dockerd | grep -v grep)" ]; then

   sudo service docker start

fi' >> ~/.bashrc

# Create directory for Kubernetes config

echo "[7/8] Creating Kubernetes config directory..."

mkdir -p ~/.kube

# Final setup and verification

echo "[8/8] Performing final setup and verification..."

echo '# Kubernetes aliases

alias k="kubectl"

alias kgp="kubectl get pods"

alias kgs="kubectl get services"

alias kgd="kubectl get deployments"

alias kgn="kubectl get nodes"

alias kga="kubectl get all"' >> ~/.bashrc
```



```
"Important Notes:"
"1. You may need to log out and log back in for the Docker group changes to take ef
"2. To start Minikube, run: 'minikube start --driver=docker'"
"3. WSL has some limitations with Kubernetes. For production use, consider native L
""
"To start using Kubernetes, run:"
"minikube start --driver=docker"
""
```

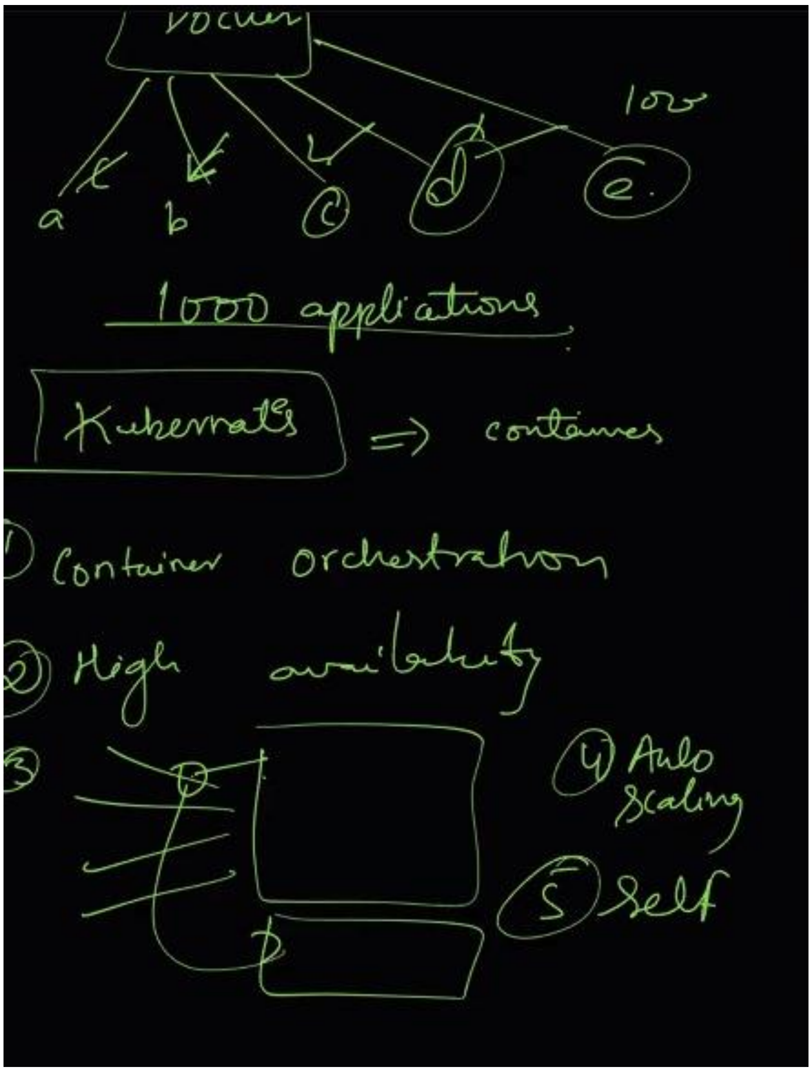
Difference between docker and Kubernetes:



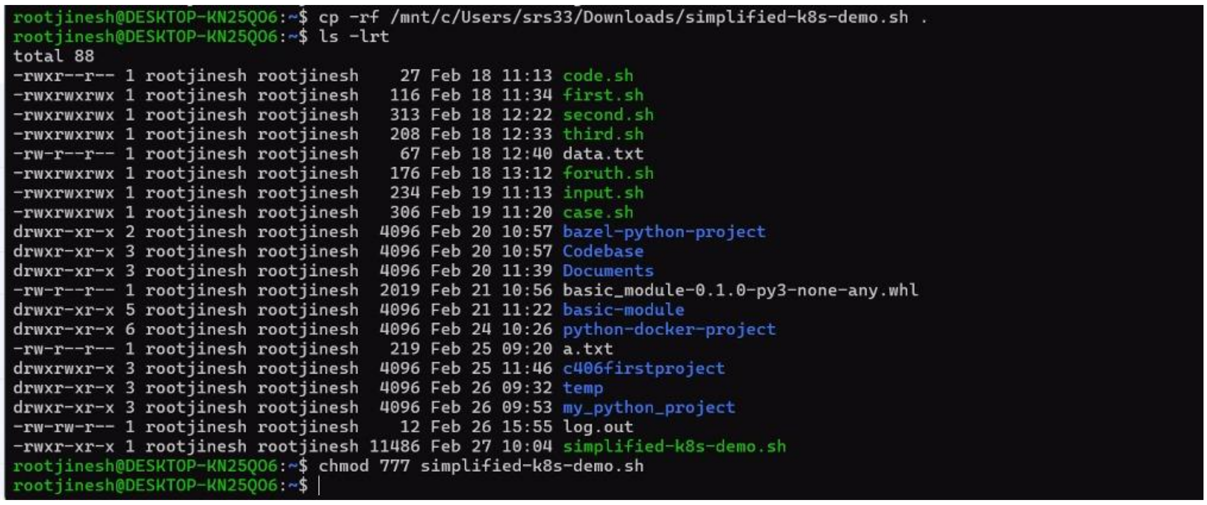
It's important to understand that Docker and Kubernetes aren't really in direct competition. They work together. Docker is a containerization platform, and Kubernetes is a container orchestration platform. Here's a breakdown of why you'd use Kubernetes, especially in scenarios where Docker alone isn't sufficient:

Kubernetes' Strengths (Orchestration):

- **Scalability:**
  - Kubernetes can automatically scale your applications up or down based on demand. This is crucial for handling varying workloads.
- **High Availability:**
  - Kubernetes can ensure that your applications remain available even if some containers or servers fail. It does this through self-healing capabilities, like restarting failed containers.
- **Automated Deployments and Rollouts:**
  - Kubernetes automates the process of deploying new versions of your applications. It also allows for controlled rollouts and easy rollbacks if something goes wrong.
- **Service Discovery and Load Balancing:**
  - Kubernetes provides built-in service discovery, making it easy for different parts of your application to find and communicate with each other. It also distributes traffic across your containers to ensure optimal performance.
- **Resource Management:**
  - Kubernetes efficiently manages the resources of your cluster, allocating CPU and memory to your containers as needed.

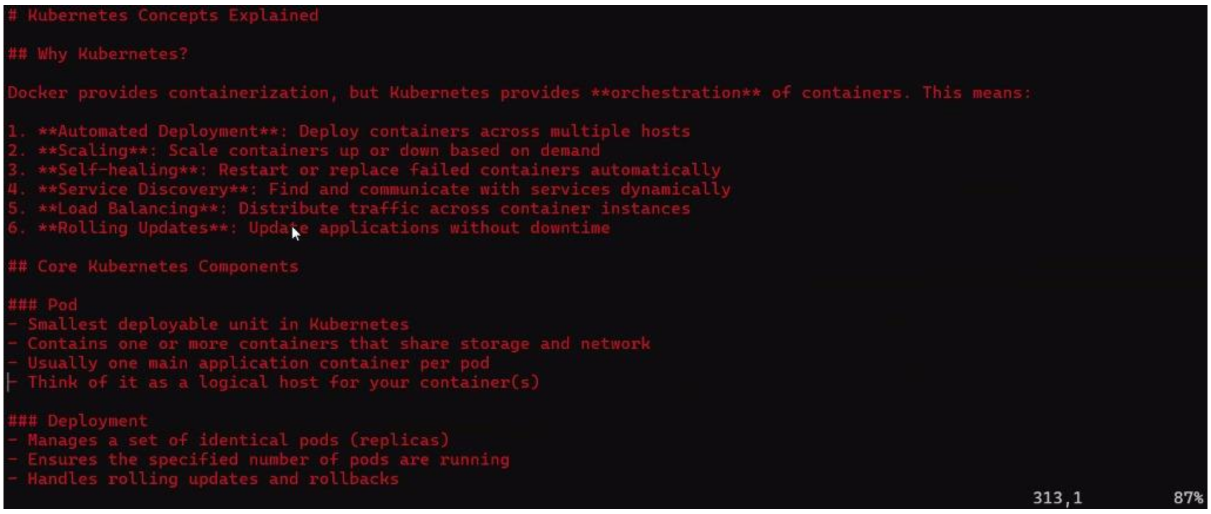


- Now docker some time faced issue related to orchestration, when we had multiple containers we use kubernetics to overcome it.
- As it provides higher availability.
- Load balancer to handle large cloud.
- parallel container can run in kubernetics.
- auto scaling, if multiple container are created load balancer balances the load and doesn't affect working of our container.
- self healing or auto healing.. when our container fails and our system close, then it would automatically handle that reset the container and set the crux it.
- minikube allows to run kubernetics , without multiple network of computer. on our local machine.



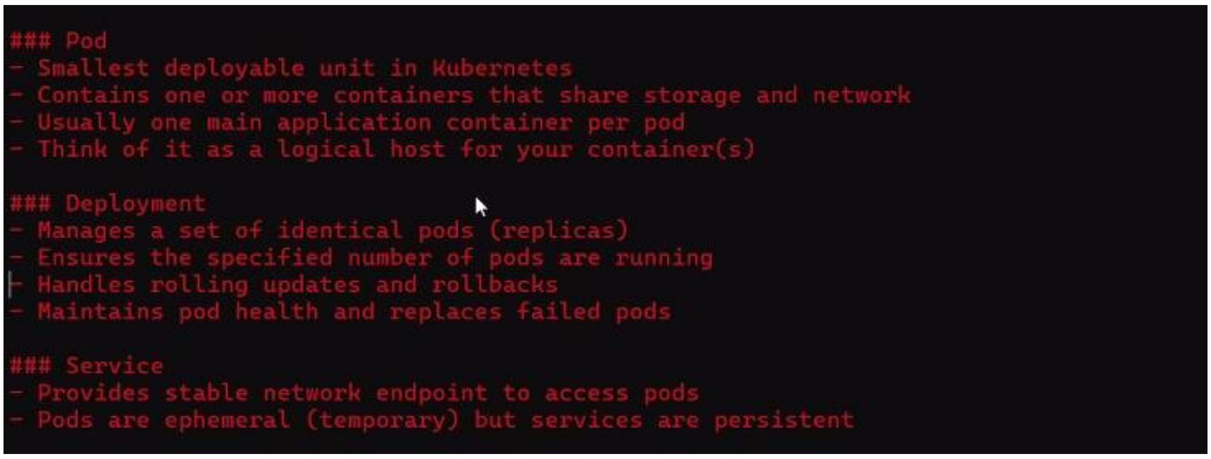
```
rootjinesh@DESKTOP-KN25Q06:~$ cp -rf /mnt/c/Users/srs33/Downloads/simplified-k8s-demo.sh .
rootjinesh@DESKTOP-KN25Q06:~$ ls -lrt
total 88
-rwxr--r-- 1 rootjinesh rootjinesh    27 Feb 18 11:13 code.sh
-rwxrwxrwx 1 rootjinesh rootjinesh   116 Feb 18 11:34 first.sh
-rwxrwxrwx 1 rootjinesh rootjinesh   313 Feb 18 12:22 second.sh
-rwxrwxrwx 1 rootjinesh rootjinesh   208 Feb 18 12:33 third.sh
-rw-r--r-- 1 rootjinesh rootjinesh    67 Feb 18 12:40 data.txt
-rwxrwxrwx 1 rootjinesh rootjinesh   176 Feb 18 13:12 foruth.sh
-rwxrwxrwx 1 rootjinesh rootjinesh   234 Feb 19 11:13 input.sh
-rwxrwxrwx 1 rootjinesh rootjinesh   306 Feb 19 11:20 case.sh
drwxr-xr-x 2 rootjinesh rootjinesh  4096 Feb 20 10:57 bazel-python-project
drwxr-xr-x 3 rootjinesh rootjinesh  4096 Feb 20 10:57 Codebase
drwxr-xr-x 3 rootjinesh rootjinesh  4096 Feb 20 11:39 Documents
-rw-r--r-- 1 rootjinesh rootjinesh  2019 Feb 21 10:56 basic_module-0.1.0-py3-none-any.whl
drwxr-xr-x 5 rootjinesh rootjinesh  4096 Feb 21 11:22 basic-module
drwxr-xr-x 6 rootjinesh rootjinesh  4096 Feb 24 10:26 python-docker-project
-rw-r--r-- 1 rootjinesh rootjinesh   219 Feb 25 09:20 a.txt
drwxrwxr-x 3 rootjinesh rootjinesh  4096 Feb 25 11:46 c406firstproject
drwxr-xr-x 3 rootjinesh rootjinesh  4096 Feb 26 09:32 temp
drwxr-xr-x 3 rootjinesh rootjinesh  4096 Feb 26 09:53 my_python_project
-rw-rw-r-- 1 rootjinesh rootjinesh    12 Feb 26 15:55 log.out
-rwxr-xr-x 1 rootjinesh rootjinesh 11486 Feb 27 10:04 simplified-k8s-demo.sh
rootjinesh@DESKTOP-KN25Q06:~$ chmod 777 simplified-k8s-demo.sh
rootjinesh@DESKTOP-KN25Q06:~$ |
```

Now before running the file we can read it:



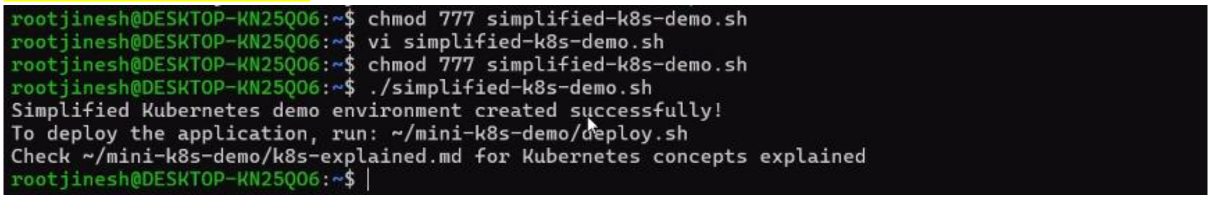
```
# Kubernetes Concepts Explained

## Why Kubernetes?

Docker provides containerization, but Kubernetes provides **orchestration** of containers. This means:

1. **Automated Deployment**: Deploy containers across multiple hosts
2. **Scaling**: Scale containers up or down based on demand
3. **Self-healing**: Restart or replace failed containers automatically
4. **Service Discovery**: Find and communicate with services dynamically
5. **Load Balancing**: Distribute traffic across container instances
6. **Rolling Updates**: Update applications without downtime

## Core Kubernetes Components

### Pod
- Smallest deployable unit in Kubernetes
- Contains one or more containers that share storage and network
- Usually one main application container per pod
- Think of it as a logical host for your container(s)

### Deployment
- Manages a set of identical pods (replicas)
- Ensures the specified number of pods are running
- Handles rolling updates and rollbacks
                                                                                313,1         87%
```

we can assume pod as a port where we can post multiple containers, they share same place and same network, which overwrites the rule of one application one container one pod. but still it is the best practice to have one container in one pod.



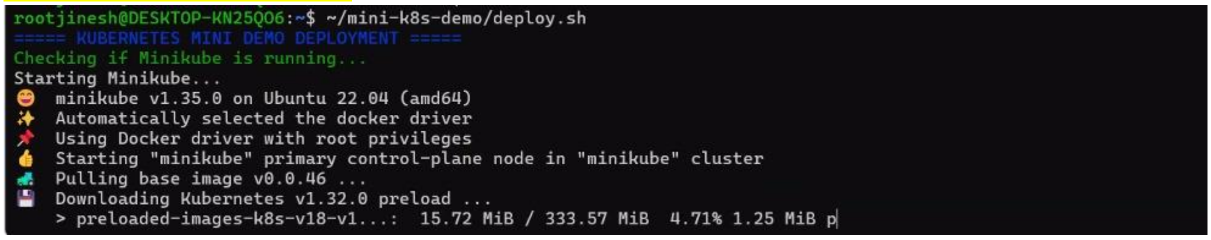
```
### Pod
- Smallest deployable unit in Kubernetes
- Contains one or more containers that share storage and network
- Usually one main application container per pod
- Think of it as a logical host for your container(s)

### Deployment
- Manages a set of identical pods (replicas)
- Ensures the specified number of pods are running
- Handles rolling updates and rollbacks
- Maintains pod health and replaces failed pods

### Service
- Provides stable network endpoint to access pods
- Pods are ephemeral (temporary) but services are persistent
```

Now to deploy it:



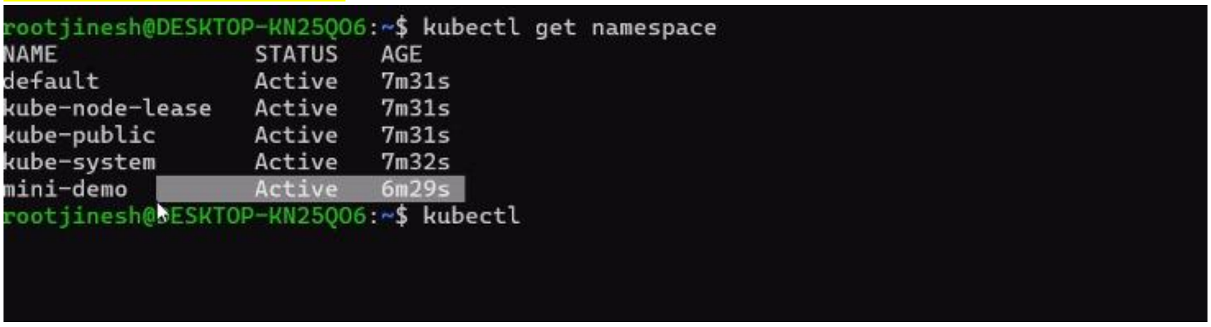
```
rootjinesh@DESKTOP-KN25Q06:~$ chmod 777 simplified-k8s-demo.sh
rootjinesh@DESKTOP-KN25Q06:~$ vi simplified-k8s-demo.sh
rootjinesh@DESKTOP-KN25Q06:~$ chmod 777 simplified-k8s-demo.sh
rootjinesh@DESKTOP-KN25Q06:~$ ./simplified-k8s-demo.sh
Simplified Kubernetes demo environment created successfully!
To deploy the application, run: ~/mini-k8s-demo/deploy.sh
Check ~/mini-k8s-demo/k8s-explained.md for Kubernetes concepts explained
rootjinesh@DESKTOP-KN25Q06:~$ |
```

Now to run our file:



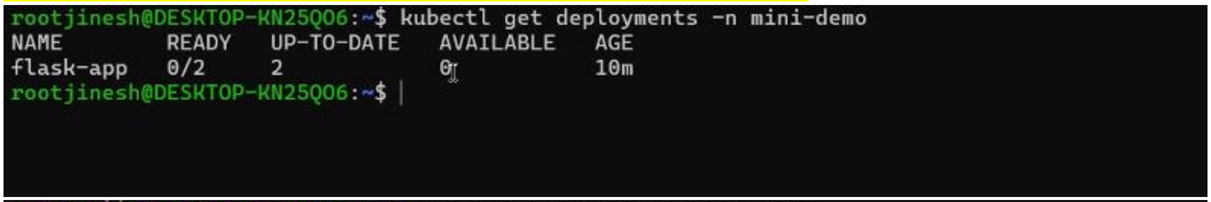
```
rootjinesh@DESKTOP-KN25Q06:~$ ~/mini-k8s-demo/deploy.sh
===== KUBERNETES MINI DEMO DEPLOYMENT =====
Checking if Minikube is running...
Starting Minikube...
😄  minikube v1.35.0 on Ubuntu 22.04 (amd64)
✨  Automatically selected the docker driver
📌  Using Docker driver with root privileges
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.46 ...
💾  Downloading Kubernetes v1.32.0 preload ...
    > preloaded-images-k8s-v18-v1...:  15.72 MiB / 333.57 MiB  4.71% 1.25 MiB p
```

To see name space:

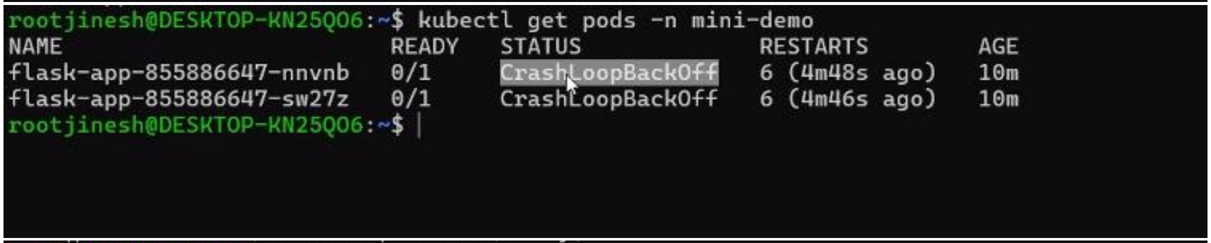


```
rootjinesh@DESKTOP-KN25Q06:~$ kubectl get namespace
NAME              STATUS   AGE
default           Active   7m31s
kube-node-lease   Active   7m31s
kube-public       Active   7m31s
kube-system       Active   7m32s
mini-demo         Active   6m29s
rootjinesh@DESKTOP-KN25Q06:~$ kubectl
```

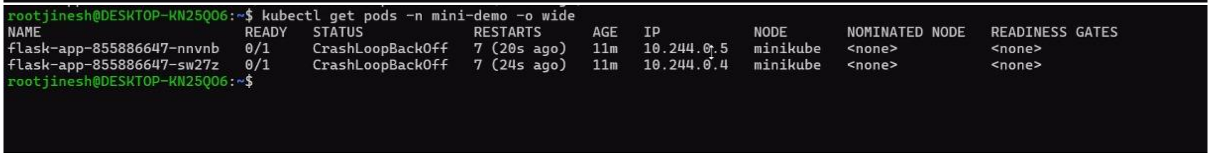
To see if deployed images are available or not:



```
rootjinesh@DESKTOP-KN25Q06:~$ kubectl get deployments -n mini-demo
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
flask-app   0/2     2            0           10m
rootjinesh@DESKTOP-KN25Q06:~$ |
```

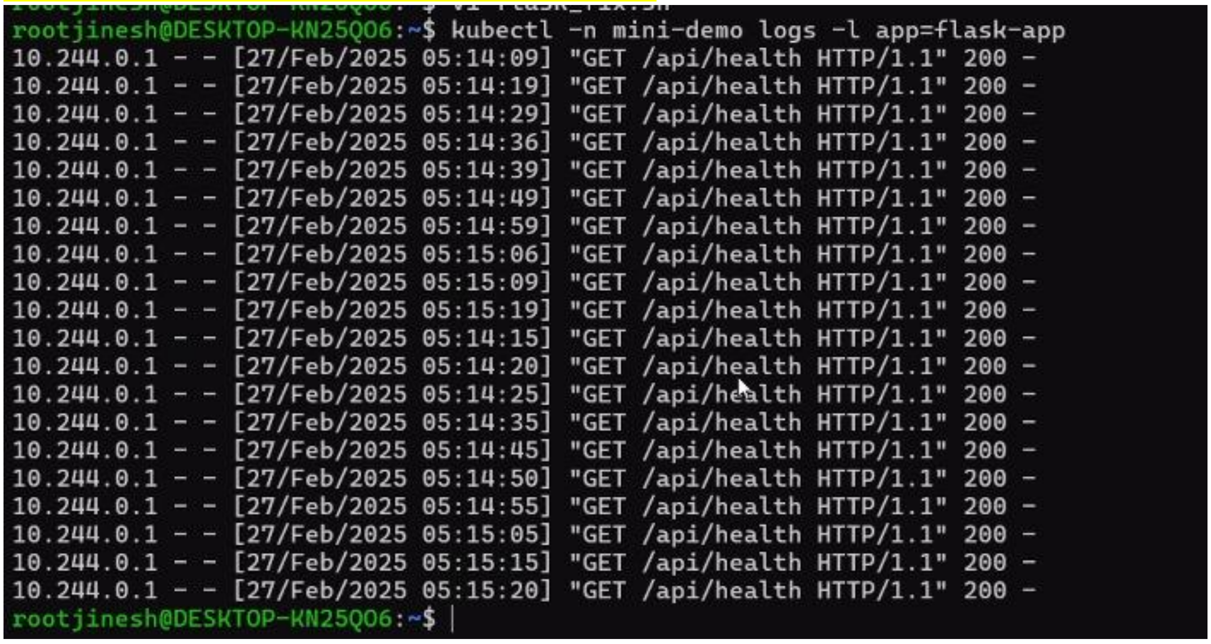


```
rootjinesh@DESKTOP-KN25Q06:~$ kubectl get pods -n mini-demo
NAME                          READY   STATUS             RESTARTS        AGE
flask-app-855886647-nnvnb     0/1     CrashLoopBackOff   6 (4m48s ago)   10m
flask-app-855886647-sw27z     0/1     CrashLoopBackOff   6 (4m46s ago)   10m
rootjinesh@DESKTOP-KN25Q06:~$ |
```



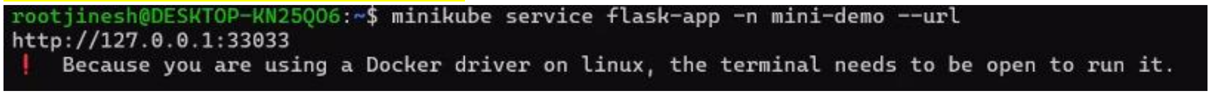
```
rootjinesh@DESKTOP-KN25Q06:~$ kubectl get pods -n mini-demo -o wide
NAME                          READY   STATUS             RESTARTS      AGE   IP           NODE       NOMINATED NODE   READINESS GATES
flask-app-855886647-nnvnb     0/1     CrashLoopBackOff   7 (20s ago)   11m   10.244.0.5   minikube   <none>           <none>
flask-app-855886647-sw27z     0/1     CrashLoopBackOff   7 (24s ago)   11m   10.244.0.4   minikube   <none>           <none>
rootjinesh@DESKTOP-KN25Q06:~$
```

If facing an error we can print logfile:



```
rootjinesh@DESKTOP-KN25Q06:~$ kubectl -n mini-demo logs -l app=flask-app
10.244.0.1 - - [27/Feb/2025 05:14:09] "GET /api/health HTTP/1.1" 200 -
10.244.0.1 - - [27/Feb/2025 05:14:19] "GET /api/health HTTP/1.1" 200 -
10.244.0.1 - - [27/Feb/2025 05:14:29] "GET /api/health HTTP/1.1" 200 -
10.244.0.1 - - [27/Feb/2025 05:14:36] "GET /api/health HTTP/1.1" 200 -
10.244.0.1 - - [27/Feb/2025 05:14:39] "GET /api/health HTTP/1.1" 200 -
10.244.0.1 - - [27/Feb/2025 05:14:49] "GET /api/health HTTP/1.1" 200 -
10.244.0.1 - - [27/Feb/2025 05:14:59] "GET /api/health HTTP/1.1" 200 -
10.244.0.1 - - [27/Feb/2025 05:15:06] "GET /api/health HTTP/1.1" 200 -
10.244.0.1 - - [27/Feb/2025 05:15:09] "GET /api/health HTTP/1.1" 200 -
10.244.0.1 - - [27/Feb/2025 05:15:19] "GET /api/health HTTP/1.1" 200 -
10.244.0.1 - - [27/Feb/2025 05:14:15] "GET /api/health HTTP/1.1" 200 -
10.244.0.1 - - [27/Feb/2025 05:14:20] "GET /api/health HTTP/1.1" 200 -
10.244.0.1 - - [27/Feb/2025 05:14:25] "GET /api/health HTTP/1.1" 200 -
10.244.0.1 - - [27/Feb/2025 05:14:35] "GET /api/health HTTP/1.1" 200 -
10.244.0.1 - - [27/Feb/2025 05:14:45] "GET /api/health HTTP/1.1" 200 -
10.244.0.1 - - [27/Feb/2025 05:14:50] "GET /api/health HTTP/1.1" 200 -
10.244.0.1 - - [27/Feb/2025 05:14:55] "GET /api/health HTTP/1.1" 200 -
10.244.0.1 - - [27/Feb/2025 05:15:05] "GET /api/health HTTP/1.1" 200 -
10.244.0.1 - - [27/Feb/2025 05:15:15] "GET /api/health HTTP/1.1" 200 -
10.244.0.1 - - [27/Feb/2025 05:15:20] "GET /api/health HTTP/1.1" 200 -
rootjinesh@DESKTOP-KN25Q06:~$ |
```

To check the server name:



```
rootjinesh@DESKTOP-KN25Q06:~$ minikube service flask-app -n mini-demo --url
http://127.0.0.1:33033
❗  Because you are using a Docker driver on linux, the terminal needs to be open to run it.
```

Now if facing some error then run trouble shoot and then again try running the system file again.

(for reference those files are saved in email for date 27 feb 2025).