

L3 ASSIGNMENT 1

STEPS TO CREATE KUBERNETES KIND CLUSTER:-

- `curl -Lo ./kind https://kind.sigs.k8s.io/dl/v0.14.0/kind-linux-amd64`, use this command to install kind on linux
- Change the binary's permissions to make it executable by using “`chmod +x ./kind`” command
- Move Kind to an application directory, such as **/bin**: `sudo mv ./kind /bin/kind`
- Create a yaml file to configure the cluster with any name “kind.yaml”
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
- role: worker
- Use the command “`kind create cluster`” command to create a cluster

```
root@ip-172-31-13-144:~# chmod +x ./kind
root@ip-172-31-13-144:~# sudo mv ./kind /usr/local/bin/kind
root@ip-172-31-13-144:~# vi kind.yaml
root@ip-172-31-13-144:~# kind create cluster --config kind-config.yaml --name=demo
ERROR: failed to create cluster: error reading file: open kind-config.yaml: no such file or directory
root@ip-172-31-13-144:~# kind create cluster --config kind.yaml --name=demo
Creating cluster "demo" ...
 ✓ Ensuring node image (kindest/node:v1.25.0)
 ✓ Preparing nodes
 ✓ Writing configuration
 ✓ Starting control-plane
 ✓ Installing CNI
 ✓ Installing StorageClass
 ✓ Joining worker nodes
Set kubectl context to "kind-demo"
You can now use your cluster with:
```

STEPS TO CONTAINERIZE APPLICATION AND DEPLOY USING KIND

- Create an account on docker hub
- Create a docker file such as :
FROM python:3.8-buster
RUN pip install --upgrade pip
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY src/ .
CMD ["python", "trtest.py"]
Save and close the file
- Build the docker image :`docker build -t myimage`
- Tagging the image as per requirement: `docker image tag myimage Gourab/myimage:latest`
- Pushing the image to docker hub : `docker image push Gourab/myimage:latest`

L3 ASSIGNMENT 1

Create a deployment manifest for the application named `deployment.yml` in your home directory:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapp
  labels:
    app: myapp
spec:
  replicas: 3
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
      - name: myimage
        image: myimage:latest
        ports:
        - containerPort: 80
```

- Command to create deployment : `kubectl apply deployment.yml`
- Now to check deployment use command : `kubectl get deployments`
- Output in this format:

NAME		READY	UP-TO-DATE	AVAILABLE	AGE
webapp	0/3	0	0	1s	