# Project Title: CI/CD - DevOps

**Duration**: 01/09/2025 – 30/09/2025

**Organization**: Sunbeam Institute Of Information Technology

**Submitted by**: Neha Sanjay Kumbharde (**93886**)

Under the guidance of

**Mr. Gajanan Taur Sir**

## DECLARATION

We declare that this written submission represents our ideas in our own words and where others ideas or words have been included; we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed**.**

Place: Pune

Date:

Neha Sanjay Kumbharde (**93886**)

# CERTIFICATE

This is to certify that the project report entitled **“CI/CD - DevOps”**, submitted by **Neha Kumbharde** is the bonafide work completed under our supervision and guidance in partial fulfillment for the Internship Program of Sunbeam Institute of Information Technology, Pune

Place: Pune

Date:

**<u>Day 01 (01/09/2025)</u> :**

**Objectives:**

- Install Docker on Ubuntu/Debian machine.
- Containerize a Flask-based Todo App.
- Run containers for both application and database.

**Tools & Technologies:**

- Docker
- Dockerfile
- Flask (Python)
- PostgreSQL / SQLite (as DB)

**Tasks and Implementation:**

1.Docker Installation:

a) Installed Docker Engine using repository:

```
sudo apt update && sudo apt upgrade -y
sudo apt install apt-transport-https ca-certificates curl software-properties-common -y
sudo apt update
sudo apt install docker-ce -y
sudo systemctl start docker
sudo systemctl enable docker
```

2.Dockerize Todo App:

a) Created a Dockerfile for the Flask application:

```
FROM python:3.9-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
CMD ["python", "app.py"]
```

3. Docker Image Creation:

```
docker build -t todo-app:latest .
```

4. Run Containers:

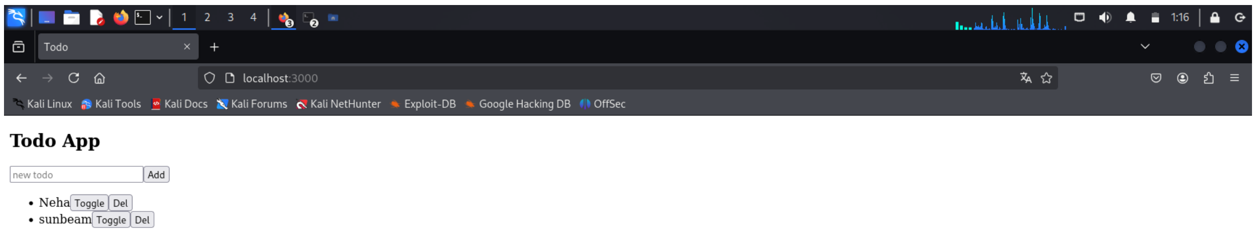
```
Docker network create todonet

Docker run -d –name  db  –network  todonet -e POSTGRES_PASSWORD=pass postgres

Docker run -d –name todoapp –network todonet -p 5000:5000 todoapp
```

**Outcome:**

A basic Flask Todo application is containerized and running in isolated Docker containers, connected via a custom network.



Todo App

new todo Add

- Neha Toggle Del
- sunbeam Toggle Del

---

## Day 02 (09/09/2025):

**Objectives:**

- Create a Kubernetes Cluster using two VMs (1 Master, 1 Worker)

**Tools & References:**

- VirtualBox / Cloud VMs

- kubeadm, kubelet, kubectl

**Tasks and Implementation:**

1) **VM Setup:**

- Two Ubuntu VMs created.
- Required packages installed: containerd, kubeadm, kubectl, kubelet.

*2)* **Initialize Master Node:**

*kubeadm init –pod-network-cidr=192.168.0.0/16*

*3)* **Set Up kubeconfig***:*

*mkdir -p $HOME/.kube*
*cp /etc/kubernetes/admin.conf $HOME/.kube/config*

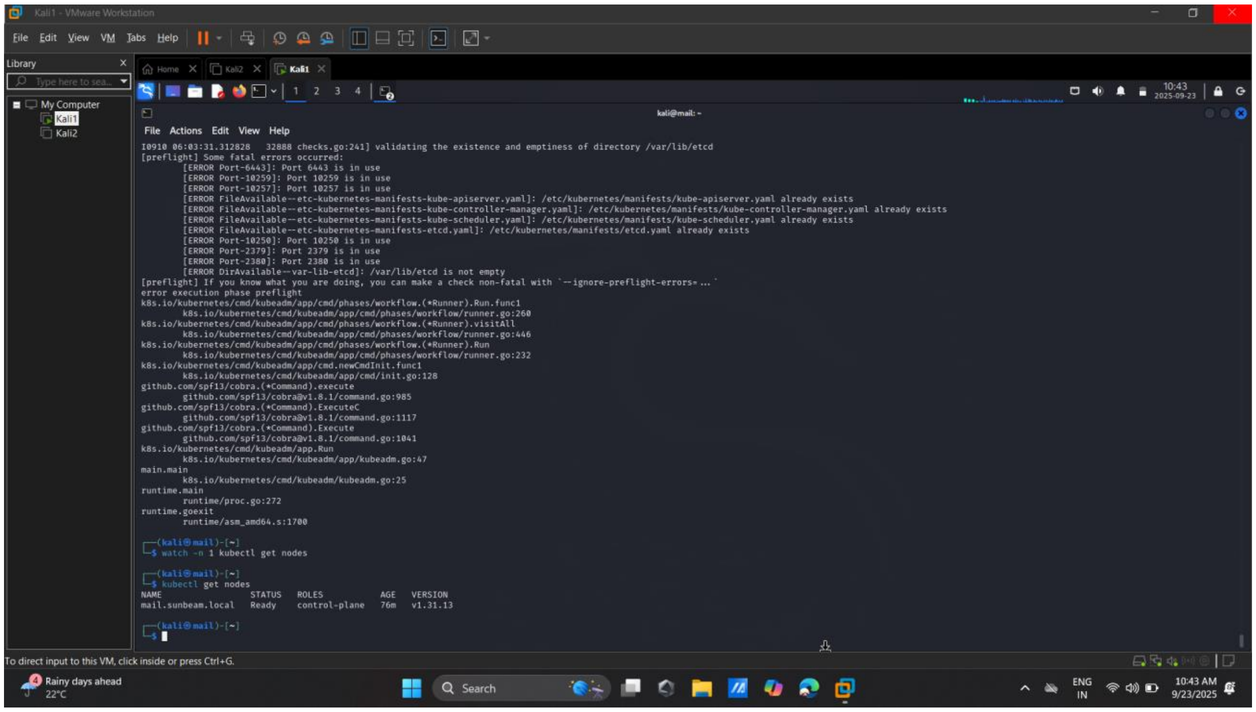
4) **Join Worker Node:**

a ) Used the kubeadm join token command from master to connect worker node.

5) **Deploy Network Plugin (Calico/Flannel)**:

*kubectl apply -f  https://docs.projectcalico.org/manifests/calico.yaml*

**Outcome:** A working Kubernetes cluster with master and worker node is successfully deployed and ready for application orchestration.



```
I0910 06:03:31.312828   32888 checks.go:241] validating the existence and emptiness of directory /var/lib/etcd
[preflight] Some fatal errors occurred:
        [ERROR Port-6443]: Port 6443 is in use
        [ERROR Port-10259]: Port 10259 is in use
        [ERROR Port-10257]: Port 10257 is in use
        [ERROR FileAvailable--etc-kubernetes-manifests-kube-apiserver.yaml]: /etc/kubernetes/manifests/kube-apiserver.yaml already exists
        [ERROR FileAvailable--etc-kubernetes-manifests-kube-controller-manager.yaml]: /etc/kubernetes/manifests/kube-controller-manager.yaml already exists
        [ERROR FileAvailable--etc-kubernetes-manifests-kube-scheduler.yaml]: /etc/kubernetes/manifests/kube-scheduler.yaml already exists
        [ERROR FileAvailable--etc-kubernetes-manifests-etcd.yaml]: /etc/kubernetes/manifests/etcd.yaml already exists
        [ERROR Port-10250]: Port 10250 is in use
        [ERROR Port-2379]: Port 2379 is in use
        [ERROR Port-2380]: Port 2380 is in use
        [ERROR DirAvailable--var-lib-etcd]: /var/lib/etcd is not empty
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
error execution phase preflight
k8s.io/kubernetes/cmd/kubeadm/app/cmd/phases/workflow.(*Runner).Run.func1
        k8s.io/kubernetes/cmd/kubeadm/app/cmd/phases/workflow/runner.go:260
k8s.io/kubernetes/cmd/kubeadm/app/cmd/phases/workflow.(*Runner).visitAll
        k8s.io/kubernetes/cmd/kubeadm/app/cmd/phases/workflow/runner.go:446
k8s.io/kubernetes/cmd/kubeadm/app/cmd/phases/workflow.(*Runner).Run
        k8s.io/kubernetes/cmd/kubeadm/app/cmd/phases/workflow/runner.go:232
k8s.io/kubernetes/cmd/kubeadm/app/cmd.newCmdInit.func1
        k8s.io/kubernetes/cmd/kubeadm/app/cmd/init.go:128
github.com/spf13/cobra.(*Command).execute
        github.com/spf13/cobra@v1.8.1/command.go:985
github.com/spf13/cobra.(*Command).ExecuteC
        github.com/spf13/cobra@v1.8.1/command.go:1117
github.com/spf13/cobra.(*Command).Execute
        github.com/spf13/cobra@v1.8.1/command.go:1041
k8s.io/kubernetes/cmd/kubeadm/app.Run
        k8s.io/kubernetes/cmd/kubeadm/app/kubeadm.go:47
main.main
        k8s.io/kubernetes/cmd/kubeadm/kubeadm.go:25
runtime.main
        runtime/proc.go:272
runtime.goexit
        runtime/asm_amd64.s:1700

┌──(kali㉿mail)-[~]
└─$ watch -n 1 kubectl get nodes

┌──(kali㉿mail)-[~]
└─$ kubectl get nodes
NAME                STATUS   ROLES           AGE   VERSION
mail.sunbeam.local  Ready    control-plane   76m   v1.31.13

┌──(kali㉿mail)-[~]
└─$ 
```

**<u>Day 03 (12/09/2025)</u>:**

**Ojectives:**

- eploy the Todo App and Database to Kubernetes.
- Create Deployment and Service YAML files.

**Tools Used:**

- Kubernetes YAML files (Deployment, Service)
- kubectl

**Tasks and Implementation:**

1. **Created Deployments:**
   a) todoapp-deployment.yaml:

   b) db-deployment.yaml: For PostgreSQL or SQLite depending on your app.
2. **Created Services:**

3. **Applied Configs:**

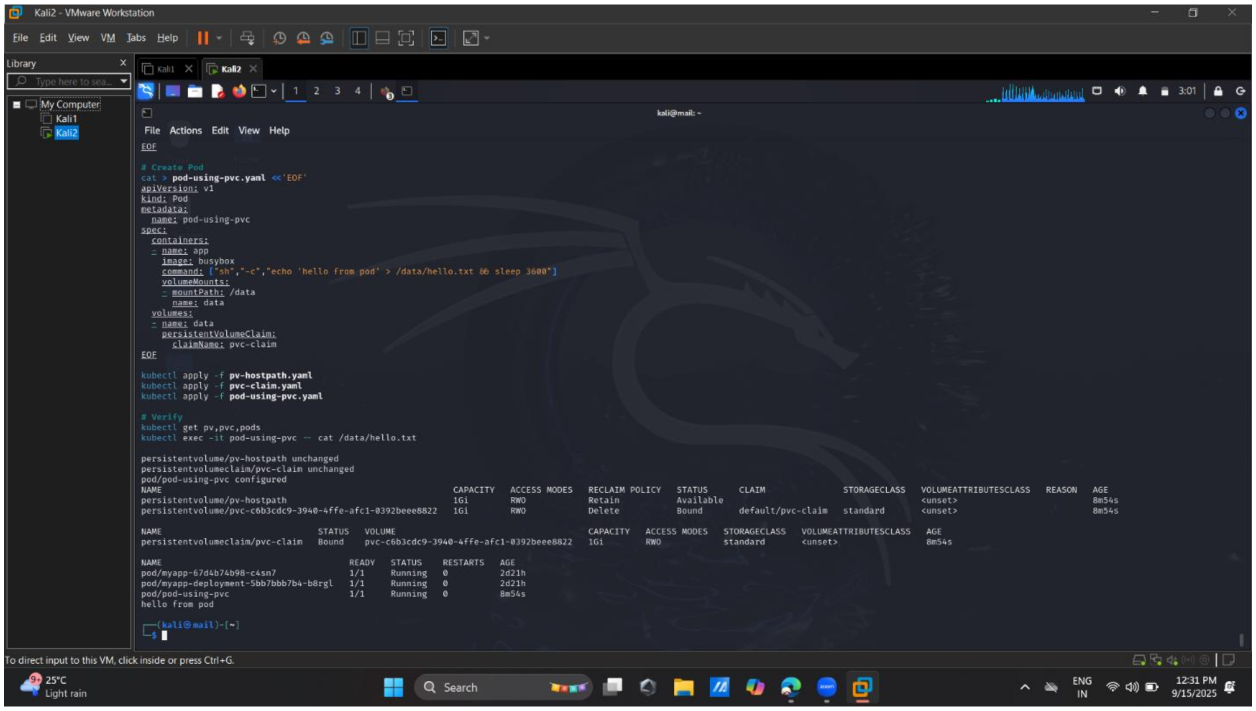
   *kubectl apply -f todoapp-deployment.yaml*

   *Kubectl apply -f db-deployment.yaml*

   *Kubectl apply -f todoapp-service.yam*

**Outcome:**

Todo App is successfully deployed to Kubernetes using declarative manifests.



```
EOF

# Create Pod
cat > pod-using-pvc.yaml <<'EOF'
apiVersion: v1
kind: Pod
metadata:
  name: pod-using-pvc
spec:
  containers:
  - name: app
    image: busybox
    command: ["sh","-c","echo 'hello from pod' > /data/hello.txt && sleep 3600"]
    volumeMounts:
    - mountPath: /data
      name: data
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: pvc-claim
EOF

kubectl apply -f pv-hostpath.yaml
kubectl apply -f pvc-claim.yaml
kubectl apply -f pod-using-pvc.yaml

# Verify
kubectl get pv,pvc,pods
kubectl exec -it pod-using-pvc -- cat /data/hello.txt

persistentvolume/pv-hostpath unchanged
persistentvolumeclaim/pvc-claim unchanged
pod/pod-using-pvc configured
NAME                                                         CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM               STORAGECLASS   VOLUMEATTRIBUTESCLASS   REASON   AGE
persistentvolume/pv-hostpath                                 1Gi        RWO            Retain           Available                                      <unset>                          8m54s
persistentvolume/pvc-c6b3cdc9-3940-4ffe-afc1-0392beee8822   1Gi        RWO            Delete           Bound       default/pvc-claim   standard       <unset>                          8m54s

NAME                              STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
persistentvolumeclaim/pvc-claim   Bound    pvc-c6b3cdc9-3940-4ffe-afc1-0392beee8822   1Gi        RWO            standard       <unset>                 8m54s

NAME                                   READY   STATUS    RESTARTS   AGE
pod/myapp-67d4b74b98-c4sn7             1/1     Running   0          2d21h
pod/myapp-deployment-5bb7bbb7b4-b8rgl  1/1     Running   0          2d21h
pod/pod-using-pvc                      1/1     Running   0          8m54s
hello from pod

┌──(kali㉿mail)-[~]
└─$ 
```

**<u>Day 04 (15/09/2025) </u>:**

**Objectives:**

- Understand and implement Persistent Volume (PV) and Persistent Volume Claim (PVC)
- Set up CI with GitHub Actions and CD with ArgoCD
- Use Helm to template deployments

**Tools & References:**

- Kubernetes (PV, PVC)
- Helm
- GitHub Actions
- ArgoCD

**Tasks and Implementation:**

1. **Storage with PV/PVC:**
   - pv.yaml and pvc.yaml created to persist DB data.
2. **CI with GitHub Actions:**
   - .github/workflows/docker-image.yml:

3. **CD with ArgoCD:**

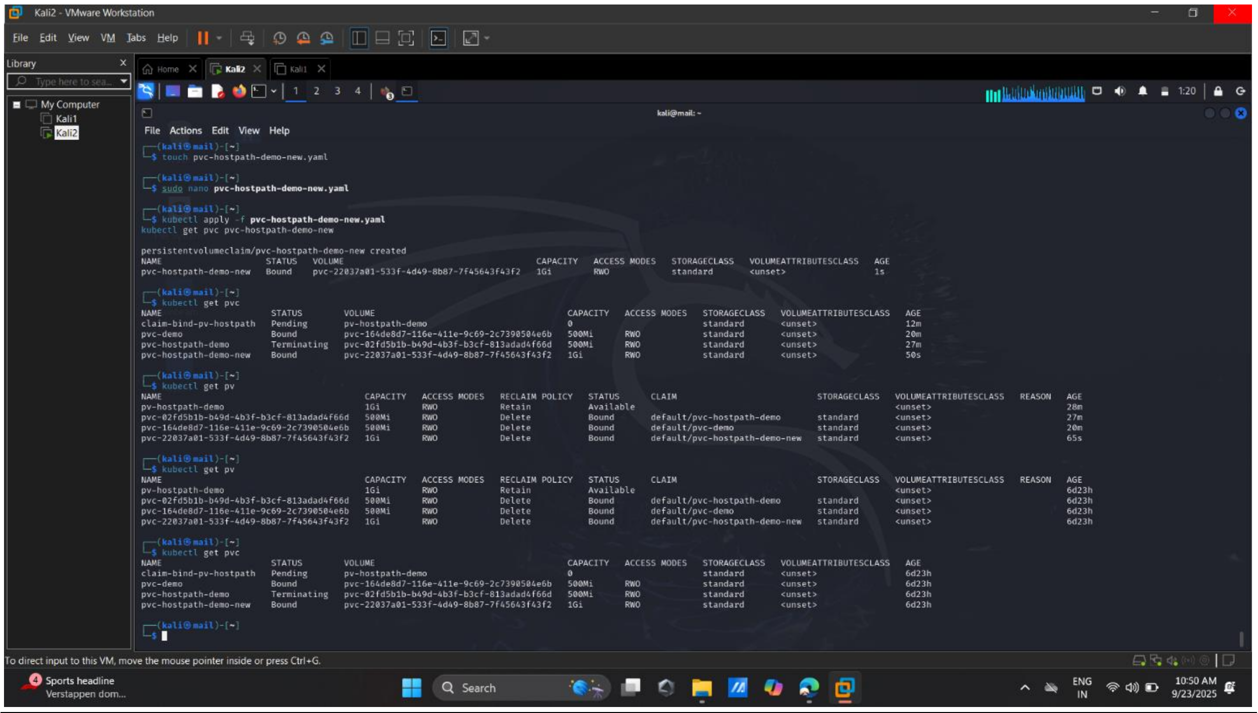
   - ArgoCD installed on cluster.
   - Linked to GitHub repo to sync Helm chart.

4. **Helm Chart:**

   - Created Chart.yaml, values.yaml, templates/ directory for Kubernetes manifests.

**Outcome:**

Complete CI/CD pipeline established. GitHub Actions handles automated builds. ArgoCD handles automated deployment from Helm charts.



```
┌──(kali㉿mail)-[~]
└─$ touch pvc-hostpath-demo-new.yaml

┌──(kali㉿mail)-[~]
└─$ sudo nano pvc-hostpath-demo-new.yaml

┌──(kali㉿mail)-[~]
└─$ kubectl apply -f pvc-hostpath-demo-new.yaml
kubectl get pvc pvc-hostpath-demo-new

persistentvolumeclaim/pvc-hostpath-demo-new created
NAME                    STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
pvc-hostpath-demo-new   Bound    pvc-22037a01-533f-4d49-8b87-7f45643f43f2   1Gi        RWO            standard       <unset>                 1s

┌──(kali㉿mail)-[~]
└─$ kubectl get pvc
NAME                     STATUS        VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
claim-bind-pv-hostpath   Pending       pv-hostpath-demo                           0                         standard       <unset>                 12m
pvc-demo                 Bound         pvc-164de8d7-116e-411e-9c69-2c7390504e6b   500Mi      RWO            standard       <unset>                 20m
pvc-hostpath-demo        Terminating   pvc-02fd5b1b-b49d-4b3f-b3cf-813adad4f66d   500Mi      RWO            standard       <unset>                 27m
pvc-hostpath-demo-new    Bound         pvc-22037a01-533f-4d49-8b87-7f45643f43f2   1Gi        RWO            standard       <unset>                 50s

┌──(kali㉿mail)-[~]
└─$ kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM                           STORAGECLASS   VOLUMEATTRIBUTESCLASS   REASON   AGE
pv-hostpath-demo                           1Gi        RWO            Retain           Available                                                  <unset>                          28m
pvc-02fd5b1b-b49d-4b3f-b3cf-813adad4f66d   500Mi      RWO            Delete           Bound       default/pvc-hostpath-demo       standard       <unset>                          27m
pvc-164de8d7-116e-411e-9c69-2c7390504e6b   500Mi      RWO            Delete           Bound       default/pvc-demo                standard       <unset>                          20m
pvc-22037a01-533f-4d49-8b87-7f45643f43f2   1Gi        RWO            Delete           Bound       default/pvc-hostpath-demo-new   standard       <unset>                          65s

┌──(kali㉿mail)-[~]
└─$ kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM                           STORAGECLASS   VOLUMEATTRIBUTESCLASS   REASON   AGE
pv-hostpath-demo                           1Gi        RWO            Retain           Available                                                  <unset>                          6d23h
pvc-02fd5b1b-b49d-4b3f-b3cf-813adad4f66d   500Mi      RWO            Delete           Bound       default/pvc-hostpath-demo       standard       <unset>                          6d23h
pvc-164de8d7-116e-411e-9c69-2c7390504e6b   500Mi      RWO            Delete           Bound       default/pvc-demo                standard       <unset>                          6d23h
pvc-22037a01-533f-4d49-8b87-7f45643f43f2   1Gi        RWO            Delete           Bound       default/pvc-hostpath-demo-new   standard       <unset>                          6d23h

┌──(kali㉿mail)-[~]
└─$ kubectl get pvc
NAME                     STATUS        VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
claim-bind-pv-hostpath   Pending       pv-hostpath-demo                           0                         standard       <unset>                 6d23h
pvc-demo                 Bound         pvc-164de8d7-116e-411e-9c69-2c7390504e6b   500Mi      RWO            standard       <unset>                 6d23h
pvc-hostpath-demo        Terminating   pvc-02fd5b1b-b49d-4b3f-b3cf-813adad4f66d   500Mi      RWO            standard       <unset>                 6d23h
pvc-hostpath-demo-new    Bound         pvc-22037a01-533f-4d49-8b87-7f45643f43f2   1Gi        RWO            standard       <unset>                 6d23h

┌──(kali㉿mail)-[~]
└─$ 
```

Kali2 - VMware Workstation
File Edit View VM Tabs Help
Library
My Computer
Kali1
Kali2
Home
Kali2
Kali1
kali@mail: ~
File Actions Edit View Help
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
Sports headline
Verstappen dom...
Search
ENG IN
10:50 AM
9/23/2025

## Day 05 (22/09/2025) :

**Objectives:**

- Set up **Prometheus** and **Grafana** for monitoring
- Create Dashboards and Alerts

**Tools Used:**

- Prometheus Operator
- Grafana
- Kubernetes Metrics Server
- Alertmanager

**Tasks and Implementation:**

1. **Install Prometheus & Grafana:**
    - Used kube-prometheus-stack Helm chart:

    *Helm repo add Prometheus-community [https://prometheus-community.github.io/helm-charts](https://prometheus-community.github.io/helm-charts)*

    *Helm install monitoring Prometheus-community/kube-prometheus-stack*

2. **Configure Metrics:**
    - Application exposed custom metrics using prometheus_flask_exporter.
    - Prometheus ServiceMonitor created to scrape metrics from Flask app.

3. **Grafana Dashboards:**

- Imported pre-built dashboards (Node Exporter, Pod Metrics, Custom Flask App metrics).
- Added visualizations for:
    - API Response Time
    - Number of Requests
    - Container CPU/Memory

4. **Set up Alerts:**
    - Created rules in prometheusRules.yaml

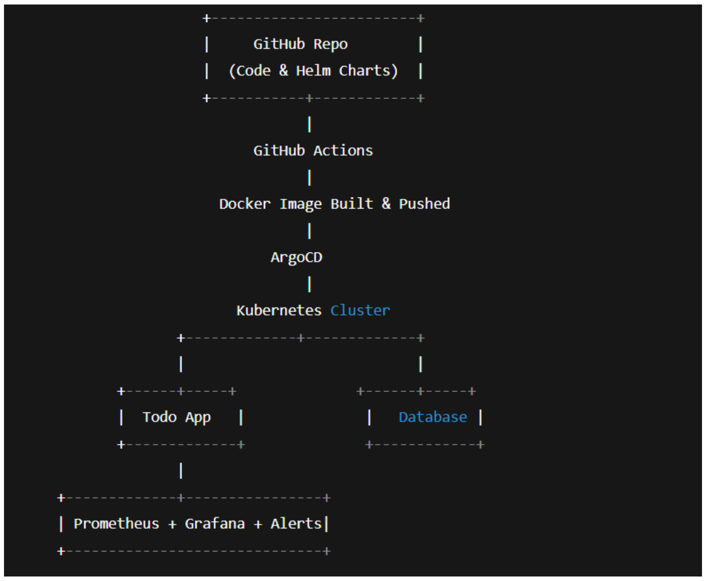
5. **Integrate Monitoring with CI/CD:**

- Alerting integrated with Slack using Alertmanager.
- Health dashboards embedded into deployment review process.

**Outcome:**

- Full-stack monitoring and alerting in place with Prometheus + Grafana. Real-time visibility into app and infrastructure metrics.

**Final Architecture Overview**:



```
                      +------------------------+
                      |     GitHub Repo        |
                      |  (Code & Helm Charts)  |
                      +-----------+------------+
                                  |
                            GitHub Actions
                                  |
                        Docker Image Built & Pushed
                                  |
                               ArgoCD
                                  |
                          Kubernetes Cluster
                   +--------------+-------------+
                   |                            |
            +------+-----+              +------+-----+
            |  Todo App   |              |   Database |
            +-------------+              +------------+
                   |
     +-------------+---------------+
     | Prometheus + Grafana + Alerts|
     +------------------------------+
```