**DevOps Day2day Workload:**

**Deploy EKS Cluster with Shell-Scripting**

***then***

**Install Prometheus & Grafana with Helm**

**Pre-requisites:**

1. AWS Account.
2. Tools installed on your local machine or bastion server:
   * AWS CLI, kubectl, Helm, eksctl.
3. IAM role with permissions to create EKS and related AWS resources.
4. Basic understanding of Kubernetes and monitoring concepts.

**Phase 1: Set Up Amazon EKS**

**1. Provision the EKS Cluster**

Use eksctl or Terraform to create an EKS cluster.

**eksctl Command:**

eksctl create cluster \

--name monitoring-cluster \

--version 1.25 \

--region us-east-1 \

--nodegroup-name monitoring-nodes \

--node-type t3.medium \

--nodes 3 \

--nodes-min 1 \

--nodes-max 4 \

--managed

**2. Authenticate kubectl**

aws eks --region us-east-1 update-kubeconfig --name monitoring-cluster

kubectl get nodes

**Prometheus & Grafana**

helm repo add **prometheus**-community https://prometheus-community.github.io/helm-charts

helm repo update

kubectl create namespace monitoring

helm **install** prometheus prometheus-community/prometheus \

--namespace monitoring \

--set server.service.type=LoadBalancer \

--set alertmanager.enabled=true \

--set pushgateway.enabled=false \

--set server.global.scrape\_interval="15s" \

--set server.global.scrape\_timeout="10s"

**kubectl get pods -n monitoring**

**kubectl get svc -n monitoring**

(optional)

**vim** values.yaml

server:

global:

scrape\_interval: 15s

scrape\_timeout: 10s

service:

type: LoadBalancer

resources:

requests:

memory: 512Mi

cpu: 500m

limits:

memory: 1Gi

cpu: 1000m

helm **upgrade** prometheus prometheus-community/prometheus \

--namespace monitoring \

-f prometheus-values.yaml

**Grafana**

**helm repo add grafana https://grafana.github.io/helm-charts**

**helm repo update**

helm **install** grafana grafana/grafana \

--namespace monitoring \

--set adminPassword=admin123 \

--set service.type=LoadBalancer

kubectl get pods -n monitoring

kubectl get svc -n monitoring

 Open Grafana in your browser: http://<loadbalancer-ip>:3000

 Login with default credentials:

* Username: admin
* Password: admin123

**Phase 4: Configure Prometheus as a Data Source in Grafana**

1. **Log in to Grafana:** Open the Grafana URL in your browser.
2. **Add Data Source:**
   * Go to **Configuration > Data Sources > Add Data Source**.
   * Select **Prometheus**.
   * Enter the Prometheus service URL:

http://prometheus-server.monitoring.svc.cluster.local

* + Click **Save & Test**.

**Phase 5: Add Kubernetes Dashboards**

1. **Import Prebuilt Dashboards:**
   * In Grafana, go to **Create > Import**.
   * Use Dashboard ID 6417 (Kubernetes Cluster Monitoring).
   * Enter the Prometheus data source and click **Import**.
2. **Verify Dashboards:**
   * Open the imported dashboards to monitor CPU, memory, pods, and nodes.

**Phase 6: Enable Alerting with Prometheus**

**1. Set Up Alertmanager**

Prometheus comes with Alertmanager by default. Verify the alertmanager pod is running:

kubectl get pods -n monitoring

**2. Create Alerting Rules**

vim alert-rules.yaml file:

groups:

- name: InstanceDown

rules:

- alert: InstanceDown

expr: up == 0

for: 1m

labels:

severity: critical

annotations:

summary: "Instance {{ $labels.instance }} is down"

description: "Instance {{ $labels.instance }} has been down for more than 1 minute."

Apply the rules to Prometheus:

kubectl apply -f alert-rules.yaml -n monitoring

**Phase 7: Deploy a Sample Application**

1. **Deploy an Nginx Pod:**

kubectl create deployment nginx --image=nginx

kubectl expose deployment nginx --type=NodePort --port=80

1. **Monitor Metrics:**
   * Use Prometheus to scrape metrics from the Nginx application.
   * View metrics on Grafana dashboards.

**Deleting:**

**kubectl delete all -l app.kubernetes.io/name=grafana -n monitoring**

**kubectl logs -l app.kubernetes.io/name=grafana -n monitoring**

**helm uninstall grafana -n monitoring**

**helm uninstall prometheus -n monitoring**