

## 1. Clone the Repository

```
git clone https://github.com/fmerchant31/k8s-assignment.git
cd eks-k8s-monitoring
```

## 2. Prerequisites

Ensure the following tools are installed:

```
aws --version
eksctl version
kubectl version --client
helm version
```

You'll also need:

- AWS account
- IAM permissions for EKS, EC2, and CloudFormation
- AWS CLI configured

```
aws configure
```

Enter:

- AWS Access Key ID
- AWS Secret Access Key
- Default region name (e.g., `us-east-1`)
- Default output format (`json`)

## 3. Create EKS Cluster

Use **eksctl** for simplicity.

```
eksctl create cluster \
  --name dream-eks-cluster \
  --region us-east-1 \
  --nodegroup-name worker-nodes \
  --node-type t3.medium \
  --nodes 2 \
  --nodes-min 2 \
  --nodes-max 4 \
  --managed
```

This may take 10–15 minutes.

Once the cluster is ready, update kubeconfig:

```
aws eks update-kubeconfig --name dream-eks-cluster --region us-east-1  
kubectl get nodes
```

You should see 2 worker nodes in **Ready** state.

## 4. Deploy Metrics Server (for HPA)

HPA requires the metrics-server to collect CPU/memory usage.

```
kubectl apply -f  
https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml  
kubectl get deployment metrics-server -n kube-system
```

Check metrics availability:

```
kubectl top nodes  
kubectl top pods
```

## 5. Deploy the Sample Application

Apply the application manifest files:

```
kubectl apply -f deployment.yaml  
kubectl apply -f service.yaml
```

Verify:

```
kubectl get pods  
kubectl get svc
```

## 6. Set Up Ingress Controller

Install **NGINX Ingress Controller** using Helm:

```
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx  
helm repo update
```

```
helm install nginx-ingress ingress-nginx/ingress-nginx \
  --namespace ingress-nginx --create-namespace
```

Apply the ingress resource:

```
kubectl apply -f ingress.yaml
kubectl get ingress
```

Wait until an **External IP or DNS** is assigned (few minutes).

Test in browser:

```
http://<external-IP>/
```

Optional: For local DNS testing, add to `/etc/hosts`:

```
<external-IP> sample-app.local
```

## 7. Configure Horizontal Pod Autoscaler (HPA)

Deploy HPA:

```
kubectl apply -f hpa.yaml
kubectl get hpa
```

Check scaling behavior:

```
kubectl describe hpa
```

Simulate load:

```
kubectl run -i --tty load-generator --image=busybox --restart=Never --
/bin/sh
# Inside the shell:
while true; do wget -q -O- http://sample-app-service; done
```

Monitor scaling:

```
kubectl get pods --watch
```

## 8. Integrate Datadog for Monitoring

## Step 1: Create Datadog API Key

1. Log in to [Datadog](#)
2. Go to **Integrations** -> **APIs**
3. Copy your **API Key**

## Step 2: Create Kubernetes Secret

Replace `<API_KEY>` with your actual Datadog key:

```
kubectl create secret generic datadog-secret --from-literal  
api-key=<API_KEY>
```

## Step 3: Add Helm Repo & Install Datadog Agent

```
helm repo add datadog https://helm.datadoghq.com  
helm repo update  
helm install datadog-agent datadog/datadog -f datadog-values.yaml
```

### Example `datadog-values.yaml`

```
datadog:  
  apiKeyExistingSecret: datadog-secret  
  site: datadoghq.com  
  clusterName: test-eks-cluster  
  
agents:  
  containers:  
    agent:  
      env:  
        - name: DD_COLLECT_KUBERNETES_EVENTS  
          value: "true"
```

Verify installation:

```
kubectl get pods | grep datadog
```

## 9. View Metrics & Logs in Datadog

After 5–10 minutes, go to [Datadog Dashboard](#):

- **Infrastructure -> Containers -> Kubernetes**  
-> View cluster, nodes, and pod metrics.
- **Dashboards -> New Dashboard -> Add Widgets**
  - `kubernetes.memory.usage`
  - `kubernetes.cpu.usage.total`
  - `kubernetes.pods.running`
- **Monitors -> New Monitor**
  - Alert if CPU > 80%
  - Alert if Pod restarts > 3
- **Logs -> Live Tail**
  - Filter by `kube_namespace` or `pod_name`

## 10. Verify Everything

Run:

```
kubectl get all
kubectl top pods
kubectl describe hpa
kubectl get ingress
```

You should observe:

- Pods autoscaling between 2–5
- App reachable via ingress URL
- Metrics visible in Datadog
- Alerts firing based on thresholds

## 11. Cleanup

When done, clean up all resources:

```
kubectl delete -f hpa.yaml
kubectl delete -f ingress.yaml
kubectl delete -f service.yaml
kubectl delete -f deployment.yaml
helm uninstall nginx-ingress -n ingress-nginx
helm uninstall datadog-agent
```

```
eksctl delete cluster --name dream-eks-cluster --region us-east-1
```