

Kubernetes on AWS EKS with Monitoring & Autoscaling

20 Points Possible

12/10/2025

Attempt 1  In Progress
NEXT UP: Submit Assignment  Add Comment

Unlimited Attempts Allowed

12/4/2025

▼ Details

Kubernetes on AWS EKS with Monitoring & Autoscaling

Goal:

Students will deploy a simple application on EKS, configure **Horizontal Pod Autoscaling (HPA)** based on CPU, and use **CloudWatch/metrics** to monitor scaling behavior.

Suggested format:

- **In-class (60–90 min):** Deploy CloudFormation, connect kubectl, deploy app & service.
- **Homework:** Configure HPA, generate load, capture monitoring screenshots, submit deliverables.

Learning Objectives

By the end, students should be able to:

1. Launch an **EKS cluster** and node group using **CloudFormation**.
2. Configure kubectl to connect to the cluster.
3. Deploy a sample app and expose it via a **Kubernetes Service**.
4. Configure **Metrics Server** and an **HPA** to auto-scale the deployment.
5. Use **CloudWatch / kubectl top** to monitor CPU and scaling events.

Prerequisites

Students should have:

- An AWS account with permission to create:
 - EKS clusters, node groups, IAM roles, CloudFormation stacks.
- An existing **VPC** with at least **two subnets with internet access** (public subnets or private subnets with NAT).
- AWS CLI (v2), kubectl, and eksctl optional but not required.
- IAM user/role with appropriate privileges configured in AWS CLI.

You'll give them:

- CloudFormation template (below).
- Kubernetes manifests (Deployment, Service, HPA).

Part 1 – Create EKS Cluster with CloudFormation

1. Provide this CloudFormation template

Deploy eks-networking.yaml (you can find this in Canvas)

Deploy eks-basic-cluster.yaml (you can find this in Canvas)

This template assumes an existing VPC and at least two subnets. Update the parameters to include two public subnets and the corresponding VPC.

Student tasks

1. In CloudFormation → Create stack:

- Upload eks-basic-cluster.yaml (you can find this in Canvas)
- Provide:
 - ClusterName: e.g., student-eks-<netid>.
 - VpcId: from their existing VPC.
 - SubnetIds: choose **two subnets** (preferably in different AZs).
- Wait for stack status **CREATE_COMPLETE**.

2. Verify cluster exists:

```
aws eks list-clusters
```

Configure kubectl and Validate Cluster

- Configure kubeconfig:

```
aws eks update-kubeconfig \
--name <ClusterName> \
--region <your-region>
```

- Verify connectivity:

```
kubectl get nodes
```

```
kubectl get ns
```

You should see the worker nodes as Ready

- Provide a screenshot of the worker nodes. `kubectl get nodes`
- Example:

Deploy Sample App and Service

- Deploy this deployment & service manifest

File: `deployment-service.yaml` (you can find this in Canvas)

```
## run this first
```

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

```
## then run these commands separately
```

```
kubectl get pods
```

```
kubectl get svc
```

- Wait for the LoadBalancer service to get an **EXTERNAL-IP**.
- Curl or browse to the external IP to confirm it returns a response.

Screenshots required:

- `kubectl get svc`
- Browser or curl output showing app response.

Install Metrics Server & Create HPA

- Install Metrics Server

Use the official manifests:

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

- Wait a minute, then verify:

```
kubectl get apiservices | grep metrics
```

```
kubectl top nodes
```

```
kubectl top pods
```

Deploy the HPA manifest `hpa.yaml`(found in Canvas):

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: cpu-demo-hpa
spec:
```

```
scaleTargetRef:  
apiVersion: apps/v1  
kind: Deployment  
name: cpu-demo  
minReplicas: 1  
maxReplicas: 5  
metrics:  
- type: Resource  
  resource:  
    name: cpu  
    target:  
      type: Utilization  
      averageUtilization: 50
```

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

- **Screenshot required:** kubectl get hpa

Generate Load & Observe Autoscaling

- Get service URL

```
export SVC_URL=$(kubectl get svc cpu-demo-svc -o jsonpath='{.status.loadBalancer.ingress[0].hostname}')
```

```
echo $SVC_URL
```

- **Generate load from a pod inside the cluster (recommended)**

Create a simple busybox pod for load generation:

```
kubectl run load-generator \  
--image=busybox \  
--restart=Never \  
-- /bin/sh -c "while true; do wget -q -O- $SVC_URL > /dev/null; done"
```

- **Observe scaling (in separate terminals):**

```
kubectl get hpa cpu-demo-hpa -w
```

```
kubectl get pods -l app(cpu-demo) -w
```

You should see CPU utilization rise and **replicas scale up** to meet the 50% target, then scale down after load stops (they can delete the load-generator pod).

Screenshot required:

- kubectl get hpa showing increased replicas and CPU utilization.
- kubectl get pods showing increased pod count.

Monitoring with CloudWatch:

1. In AWS Console, go to **CloudWatch** → **Metrics** → Search EC2 or CPUUtilization.

2. Identify metrics for:

- Node CPU utilization and/or
- Network In/Out for worker nodes.

3. Create a **simple dashboard**:

- Add at least **two widgets** (e.g., CPUUtilization and NetworkIn for the node group or instances).

4. Capture a screenshot during or shortly after the load test showing resource usage.

Screenshot required: CloudWatch dashboard with at least 2 widgets relevant to the cluster or nodes.

Deliverables:

1. **CloudFormation proof:**

- Screenshot of CloudFormation stack in **CREATE_COMPLETE**.



2. **Cluster status:**

- Screenshot of kubectl get nodes.

3. **App & service:**

- Screenshot of kubectl get svc.
- Screenshot of app response (curl or browser).

4. **Autoscaling evidence:**

- Screenshot of kubectl get hpa showing scaled replicas.
- Screenshot of kubectl get pods showing multiple cpu-demo pods.

5. **Monitoring dashboard:**

- Screenshot of CloudWatch dashboard with at least 2 widgets.

6. **Short reflection (3–5 sentences):**

- Explain how HPA decided to scale the deployment.
- Mention one monitoring metric you would track in production and why.

✓ **View Rubric**

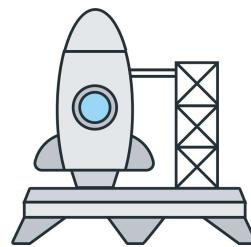
Kubernetes on AWS EKS with Monitoring & Autoscaling

Criteria	Ratings	Pts
EKS Cluster via CloudFormation view longer description	5 pts Full Marks	0 pts No Marks / 5 pts
Application Deployment & Service view longer description	4 pts Full Marks	0 pts No Marks / 4 pts
Metrics Server & HPA Configuration view longer description	5 pts Full Marks	0 pts No Marks / 5 pts
Autoscaling in Action (Pods) view longer description	3 pts Full Marks	0 pts No Marks / 3 pts
Monitoring Dashboard & Reflection view longer description	3 pts Full Marks	0 pts No Marks / 3 pts

Total Points: 0

Choose a submission type





Choose a file to upload

or

Webcam Photo

Canvas Files

1

(<https://umd.instructure.com/courses/1389234/modules/items/13944263>)

7

Assignment
<https://umd>

