AWS Kubernetes Lab Guide: HPA, Cluster Autoscaler, Prometheus & Grafana

AWS Kubernetes Class Notes: Horizontal Pod Autoscaling (HPA)

Recap: Previous Class

- Deployed an application (e.g., Node.js app) on Kubernetes.
- Connected the app to a domain purchased from GoDaddy.

Request Flow:

User → Load Balancer → Ingress → Service → Deployment → Pods

Clone Kubernetes Autoscaling Manifests

GitHub: https://github.com/sibasish934/kubernetes-manifests.git

Navigate to: auto-scalling

What is HPA (Horizontal Pod Autoscaler)?

Automatically scales pods based on observed CPU, memory, or custom metrics.

Step-by-Step Lab Implementation

1. Cluster Creation: EKS cluster already created with nodes and pods running.

2. Load Generation: Use BusyBox pod to simulate CPU load.

3. Check Resource Usage: Use `kubectl top pods`.

4. Configure HPA: Define CPU threshold in hpa.yaml.

5️⃣ Apply HPA: `kubectl apply -f hpa.yaml`

6️⃣ Monitor Scaling: Use `kubectl get hpa` and `kubectl get pods --watch`

7️⃣ Trigger Load Again: Observe scaling up and down.

☑️ Conclusion

- HPA optimizes performance by auto-scaling pods.
- Requires metrics-server.
- Can be extended to memory or custom metrics.

☸️ Cluster Autoscaler Setup (EKS)

🔐 Attach IAM Policy to user

🆔 Set up OIDC provider using eksctl

🔐 Create IAM role for autoscaler

📝 Update YAML with role ARN & cluster name

📦 Apply YAML

📄 Check logs with `kubectl logs`

🧭 Verify in AWS Console (Auto Scaling Groups)

🧩 Optional: Apply additional test deployments.

🧹 Clean Up

Delete HPA test deployment and service:

- `kubectl delete -f hpa-testing-deploy.yaml`
- `kubectl delete -f hpa-testing-service.yaml`

📊 Monitoring with Prometheus & Grafana

🔎 Prometheus: collects metrics

📈 Grafana: displays dashboards

Monitoring Setup Lab

1. Check Metrics API: `kubectl get --raw /metrics`
2. Ensure AWS Load Balancer Controller is installed
3. Install EBS CSI Driver: for persistent volumes

Installing Prometheus

- Create `prometheus` namespace
- Add Helm repo: prometheus-community
- Install with Helm and gp2 storage class
- Verify pods
- Port-forward to view Prometheus UI at http://localhost:9090

Installing Grafana

- Create `grafana` namespace
- Add Helm repo: grafana
- Install via Helm (set admin password)
- Port-forward to http://localhost:3000
- Login: admin/admin123
- Import Dashboard ID 1860 for node metrics

```
PS C:\WINDOWS\system32> aws configure
AWS Access Key ID [****************PSBR]: AKIAT5VFFIODDCIR6VNC
AWS Secret Access Key [****************u1AW]: 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
Default region name [ap-south-1]:
Default output format [json]:
PS C:\WINDOWS\system32> aws sts get-caller-identity
{
    "UserId": "269855572870",
    "Account": "269855572870",
    "Arn": "arn:aws:iam::269855572870:root"
}
2025-06-30 17:16:18 [ℹ]  nodegroup "standardworkers" has 2 node(s)
2025-06-30 17:16:18 [ℹ]  node "ip-192-168-28-21.ap-south-1.compute.internal" is ready
2025-06-30 17:16:18 [ℹ]  node "ip-192-168-90-79.ap-south-1.compute.internal" is ready
2025-06-30 17:16:18 [ℹ]  waiting for at least 1 node(s) to become ready in "standardworkers"
2025-06-30 17:16:18 [ℹ]  nodegroup "standardworkers" has 2 node(s)
2025-06-30 17:16:18 [ℹ]  node "ip-192-168-28-21.ap-south-1.compute.internal" is ready
2025-06-30 17:16:18 [ℹ]  node "ip-192-168-90-79.ap-south-1.compute.internal" is ready
2025-06-30 17:16:18 [✔]  created 1 managed nodegroup(s) in cluster "dev-cluster"
2025-06-30 17:16:19 [ℹ]  kubectl command should work with "C:\\Users\\Ayush Singh\\.kube\\config", try 'kubectl get nodes'
2025-06-30 17:16:19 [✔]  EKS cluster "dev-cluster" in "ap-south-1" region is ready
PS C:\WINDOWS\system32> aws eks update-kubeconfig --region ap-south-1 --name demo-cluster

An error occurred (ResourceNotFoundException) when calling the DescribeCluster operation: No cluster found for name: demo-c
luster.
PS C:\WINDOWS\system32> aws eks update-kubeconfig --region ap-south-1 --name dev-cluster
Added new context arn:aws:eks:ap-south-1:269855572870:cluster/dev-cluster to C:\Users\Ayush Singh\.kube\config
PS C:\WINDOWS\system32> kubectl get nodes
NAME                                           STATUS   ROLES    AGE     VERSION
ip-192-168-28-21.ap-south-1.compute.internal   Ready    <none>   3m26s   v1.32.3-eks-473151a
ip-192-168-90-79.ap-south-1.compute.internal   Ready    <none>   3m32s   v1.32.3-eks-473151a
PS C:\WINDOWS\system32\kubernetes-manifests\auto-scalling> kubectl apply -f hpa-testing-service.yaml
service/hpa-demo-deployment created
PS C:\WINDOWS\system32\kubernetes-manifests\auto-scalling> kubectl get svc
NAME                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
hpa-demo-deployment   ClusterIP   10.100.169.39   <none>        80/TCP    13s
kubernetes            ClusterIP   10.100.0.1      <none>        443/TCP   19m
PS C:\WINDOWS\system32\kubernetes-manifests\auto-scalling> kubectl run -i --tty load-generator --rm --image=busybox --restart=N
ever -- /bin/sh -c "while sleep 0.01; do wget -q -O- http://hpa-demo-deployment; done"
If you don't see a command prompt, try pressing enter.
OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!O
K!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK
!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!O
K!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK
!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!O
K!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK
!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
PS C:\WINDOWS\system32\kubernetes-manifests\auto-scalling> kubectl top pods
NAME                                   CPU(cores)   MEMORY(bytes)
hpa-demo-deployment-58955667b9-flmn2   145m         12Mi
load-generator                         22m          0Mi
PS C:\WINDOWS\system32\kubernetes-manifests\auto-scalling> kubectl get hpa
NAME                  REFERENCE                        TARGETS         MINPODS   MAXPODS   REPLICAS   AGE
hpa-demo-deployment   Deployment/hpa-demo-deployment   cpu: 156%/50%   1         10        1          2m9s
PS C:\WINDOWS\system32\kubernetes-manifests\auto-scalling> kubectl get hpa
NAME                  REFERENCE                        TARGETS         MINPODS   MAXPODS   REPLICAS   AGE
hpa-demo-deployment   Deployment/hpa-demo-deployment   cpu: 105%/50%   1         10        4          2m33s
PS C:\WINDOWS\system32\kubernetes-manifests\auto-scalling> kubectl get pods --watch
NAME                                   READY   STATUS    RESTARTS   AGE
hpa-demo-deployment-58955667b9-27glg   1/1     Running   0          40s
hpa-demo-deployment-58955667b9-2jt5l   1/1     Running   0          9s
hpa-demo-deployment-58955667b9-f68m5   1/1     Running   0          9s
hpa-demo-deployment-58955667b9-flmn2   1/1     Running   0          13m
hpa-demo-deployment-58955667b9-fq647   1/1     Running   0          40s
hpa-demo-deployment-58955667b9-v87gh   1/1     Running   0          9s
hpa-demo-deployment-58955667b9-vg6sr   1/1     Running   0          40s
load-generator                         1/1     Running   0          53s
```

```
PS C:\users\Ayush Singh\downloads\kubernetes-manifests\auto-scalling> kubectl apply -f cluster-auto-scaler.yaml
serviceaccount/cluster-autoscaler created
clusterrole.rbac.authorization.k8s.io/cluster-autoscaler created
role.rbac.authorization.k8s.io/cluster-autoscaler created
clusterrolebinding.rbac.authorization.k8s.io/cluster-autoscaler created
rolebinding.rbac.authorization.k8s.io/cluster-autoscaler created
deployment.apps/cluster-autoscaler created
PS C:\users\Ayush Singh\downloads\kubernetes-manifests\auto-scalling> kubectl get pods -n kube-system
NAME                                  READY   STATUS    RESTARTS   AGE
aws-node-6nk8f                        2/2     Running   0          51m
aws-node-n97mx                        2/2     Running   0          51m
cluster-autoscaler-64b685b46b-75529   1/1     Running   0          27s
coredns-6799d65cb-fncjt               1/1     Running   0          53m
coredns-6799d65cb-v4s76               1/1     Running   0          53m
kube-proxy-cnkcj                      1/1     Running   0          51m
kube-proxy-wptf4                      1/1     Running   0          51m
metrics-server-6c7659898f-4679w       1/1     Running   0          53m
metrics-server-6c7659898f-wwq29       1/1     Running   0          53m
```

Auto Scaling group updated successfully

Auto Scaling groups (1/1) Info

Last updated 1 minute ago | Launch configurations | Launch templates | Actions | Create Auto Scaling group

Search your Auto Scaling groups

Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max
eks-standardworkers-e2cbe053-53a9-6cc7-fb72-2a389c257ee4	eks-e2cbe053-53a9-6cc7-fb72-2a389c25	2	-	2	1	5

```
PS C:\users\Ayush Singh\downloads\kubernetes-manifests\auto-scalling> helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n kube-system --set cluste
rName=test-cluster --set serviceAccount.create=false --set serviceAccount.name=aws-load-balancer-controller --set region=ap-south-1 --set vpcId=0646886ff346a9488
NAME: aws-load-balancer-controller
LAST DEPLOYED: Mon Jun 30 18:27:19 2025
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
AWS Load Balancer controller installed!
PS C:\users\Ayush Singh\downloads\kubernetes-manifests\auto-scalling> eksctl create addon --name aws-ebs-csi-driver --cluster dev-cluster --region ap-south-1 --service-acco
unt-role-arn arn:aws:iam::269855572870:role/eks-cluster-autoscaler-role --force
2025-06-30 18:29:49 [ℹ]  Kubernetes version "1.32" in use by cluster "dev-cluster"
2025-06-30 18:29:49 [ℹ]  IRSA is set for "aws-ebs-csi-driver" addon; will use this to configure IAM permissions
2025-06-30 18:29:49 [!]  the recommended way to provide IAM permissions for "aws-ebs-csi-driver" addon is via pod identity associations; after addon creation is completed,
run `eksctl utils migrate-to-pod-identity`
2025-06-30 18:29:49 [ℹ]  using provided ServiceAccountRoleARN "arn:aws:iam::269855572870:role/eks-cluster-autoscaler-role"
2025-06-30 18:29:49 [ℹ]  creating addon: aws-ebs-csi-driver
```