

# Lab 6 – Kubernetes

## Assignments

Create a Kubernetes application (7p)

- a) Create a k8s cluster using Amazon Elastic Kubernetes Service (EKS)

I created EKS cluster on AWS

**lsc-node-1**

[Refresh](#) [Edit](#) [Delete](#)

Node group configuration <a href="#">Info</a>		
Kubernetes version 1.32	AMI type <a href="#">Info</a> AL2023_x86_64_STANDARD	Status <span>✓ Active</span>
AMI release version <a href="#">Info</a> 1.32.1-20250403	Instance types t3.medium	Disk size 20 GiB

[Details](#) [Nodes](#) [Health issues 0](#) [Kubernetes labels](#) [Update config](#) [Kubernetes taints](#) [Update history](#) [Tags](#)

Details			
<b>Node group ARN</b> <a href="#">arn:aws:eks:us-east-1:135464447074:nodegroup/lsc-cluster-1/lsc-node-1/d4cb1d63-ac0b-de4b-444c-e4cc8364619e</a>	<b>Autoscaling group name</b> <a href="#">eks-lsc-node-1-d4cb1d63-ac0b-de4b-444c-e4cc8364619e</a>	<b>Capacity type</b> On-Demand	<b>Subnets</b> <a href="#">subnet-04e3d7c8a20849798</a> <a href="#">subnet-09cfad25a585ac43f</a>
<b>Created</b> <span>2 minutes ago</span>	<b>Node IAM role ARN</b> <a href="#">arn:aws:iam::135464447074:role/LabRole</a> <a href="#">View in IAM</a>	<b>Desired size</b> 2 nodes	<b>Configure remote access to nodes</b> off
		<b>Minimum size</b> 2 nodes	
		<b>Maximum size</b> 2 nodes	

**lsc-cluster-1**

[Refresh](#) [Delete cluster](#) [View dashboard](#)

Cluster info <a href="#">Info</a>			
<b>Status</b> <span>✓ Active</span>	<b>Kubernetes version</b> <a href="#">Info</a> 1.32	<b>Support period</b> <a href="#">Standard support until March 21, 2026</a>	<b>Provider</b> EKS
<b>Cluster health issues</b> <span>✓ 0</span>	<b>Upgrade insights</b> <span>✓ 4</span>	<b>Node health issues</b> <span>✓ 0</span>	

[Overview](#) [Resources](#) [Compute](#) [Networking](#) [Add-ons 1](#) [Access](#) [Observability](#) [Update history](#) [Tags](#)

Details		
<b>API server endpoint</b> <a href="#">https://2290C9140005D18A1612482A2B0C78F6.gr7.us-east-1.eks.amazonaws.com</a>	<b>OpenID Connect provider URL</b> <a href="#">https://oidc.eks.us-east-1.amazonaws.com/id/2290C9140005D18A1612482A2B0C78F6</a>	<b>Created</b> <span>an hour ago</span>
<b>Certificate authority</b> <a href="#">LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSOtLS0tck1USURCVENDQWUyZ0F3SUJBZ0Rk1G5...</a>	<b>Cluster IAM role ARN</b> <a href="#">arn:aws:iam::135464447074:role/LabRole</a> <a href="#">View in IAM</a>	<b>Cluster ARN</b> <a href="#">arn:aws:eks:us-east-1:135464447074:cluster/lsc-cluster-1</a>
		<b>Platform version</b> <a href="#">Info</a>

- b) Using Helm, install an [NFS server and provisioner](#) in the cluster.

```

(agh_env) mikolajpajor@MacBookAir ~ % helm install nfs-server stable/nfs-server-provisioner --set storageClass.name=nfs --set persistence.enabled=false --set persistence.size=10Gi
WARNING: This chart is deprecated
NAME: nfs-server
LAST DEPLOYED: Tue Apr 15 20:51:27 2025
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The NFS Provisioner service has now been installed.

A storage class named 'nfs' has now been created
and is available to provision dynamic volumes.

You can use this storageclass by creating a 'PersistentVolumeClaim' with the
correct storageClassName attribute. For example:

---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: test-dynamic-volume-claim
spec:
  storageClassName: "nfs"
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi

```

- c) Create a [Persistent Volume Claim](#) which will bind to a NFS Persistent Volume [provisioned dynamically](#) by the provisioner installed in the previous step.

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: nfs-pvc
spec:
  storageClassName: nfs # Musi pasować do nazwy StorageClass
  accessModes:
    - ReadWriteMany # NFS obsługuje RWX
  resources:
    requests:
      storage: 1Gi
~
Documents Figure_3.png Music config srv.yaml
(agh_env) mikolajpajor@MacBookAir ~ % kubectl apply -f pvc.yaml
persistentvolumeclaim/nfs-pvc created

```

- d) Create a [Deployment](#) with a HTTP server (e.g., apache or nginx). The web content directory should be mounted as a volume using the PVC created in the previous step.

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:alpine
          ports:
            - containerPort: 80
          volumeMounts:
            - name: nfs-volume
              mountPath: /usr/share/nginx/html
      volumes:
        - name: nfs-volume
          persistentVolumeClaim:

```

- e) Create a [Service](#) associated with the Pod(s) of the HTTP server Deployment.

```

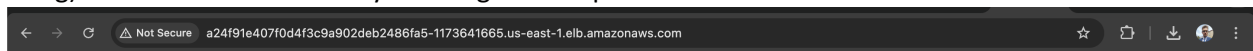
# nginx-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  type: LoadBalancer
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80

```

- f) Create a [Job](#) which mounts the PVC and copies a sample content through the shared NFS PV.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: copy-job
spec:
  template:
    spec:
      containers:
      - name: busybox
        image: busybox
        command: ["/bin/sh", "-c", "echo '<h1>Hello from NFS!</h1>' > /mnt/index.html"]
        volumeMounts:
        - name: nfs-volume
          mountPath: /mnt
      volumes:
      - name: nfs-volume
        persistentVolumeClaim:
          claimName: nfs-pvc
        restartPolicy: Never
      backoffLimit: 2
```

g) Test the HTTP server by showing the sample web content in a browser.



Files and script available on: [https://github.com/Pejdzor/large\\_scale\\_computing\\_k8s\\_nginx](https://github.com/Pejdzor/large_scale_computing_k8s_nginx)