

# Large Scale Computing

## Lab 6 - Kubernetes

Author: Adrian Żerebiec

### Table of contents

Architecture diagram .....	2
Github link .....	2
Used commands .....	3
Download and install minikube .....	3
Download and install kubectl .....	3
Run Minikube in docker .....	3
Download and install Helm .....	4
Install NFS Server/Provisioner with Helm .....	4
Applying PVC/nginx-development/nginx-service/job.....	5
Start HTTP server .....	5
Final effect .....	5
Files.....	6
pvc.yaml.....	6
nginx-deployment.yaml.....	6
nginx-service.yaml .....	7
job.yaml .....	7

# Architecture diagram

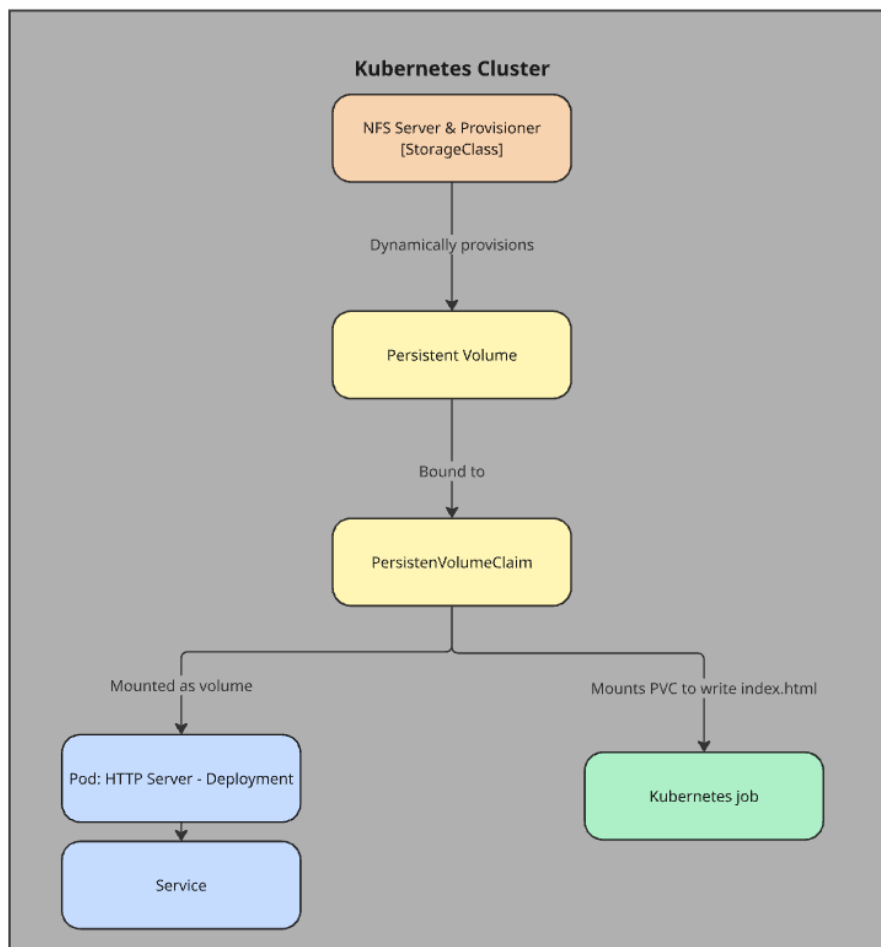


Photo 1: Diagram of the created application

**NFS Server** - Installation of the NFS server in the cluster, which will create Persistent Volumes.

**Persistent Volume** - A resource that stores data, created by the NFS Server.

**Persistent Volume Claim** - A request for storage space, used by applications to store data.

**Pod: HTTP Server** – A Kubernetes Pod (the smallest deployable unit that can run containers) running a nginx server that serves an HTML page from files stored on a mounted PVC.

**Service** - Kubernetes Service of type NodePort that allows external access to the nginx server.

**Dynamically provisions** - NFS automatically creates the PV

**Bound to** - PVC is bound to the corresponding PV

**Mounted as volume** - PVC is mounted as a volume by the job to write data

**Mounted PVC to write index.html** - PVC is mounted as a volume by the job to write data

## Github link

Link to LSC repository: [AGH-LSC](#)

Link to Lab 6 task: [Lab 6 - Kubernetes](#)

# Used commands

## Download and install minikube

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

```
sudo apt install -y curl
```

```
adrian2300@LAPTOP-LQ0MESR8:/mnt/c/Users/adria/OneDrive/Pulpit/Lab 6 - LSC$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
nstat minikube-linux-amd64 /usr/local/bin/minikube % Total % Received % Xferd Average Speed Time Time Time Current
100 119M 100 119M 0 0 9869K 0 0:00:12 0:00:12 --:--:-- 10.4M
adrian2300@LAPTOP-LQ0MESR8:/mnt/c/Users/adria/OneDrive/Pulpit/Lab 6 - LSC$ sudo install minikube-linux-amd64 /usr/local/bin/minikube
[sudo] password for adrian2300:
Sorry, try again.
[sudo] password for adrian2300:
Sorry, try again.
[sudo] password for adrian2300:
adrian2300@LAPTOP-LQ0MESR8:/mnt/c/Users/adria/OneDrive/Pulpit/Lab 6 - LSC$ sudo apt install -y curl
0 "https://dl.k8s.io/release/$(curl -s https://dl.k8s.io/releacurl -LO "https://dl.k8s.io/release/$(curl -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
sudo install kubect1 /usr/local/bin/kubect1
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
curl is already the newest version (7.81.0-1ubuntu1.20).
curl set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 92 not upgraded.
```

Photo 2: Downloading and installing minikube

## Download and install kubect1

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubect1"
```

```
sudo install kubect1 /usr/local/bin/kubect1
```

## Run Minikube in docker

```
minikube start --driver=docker
```

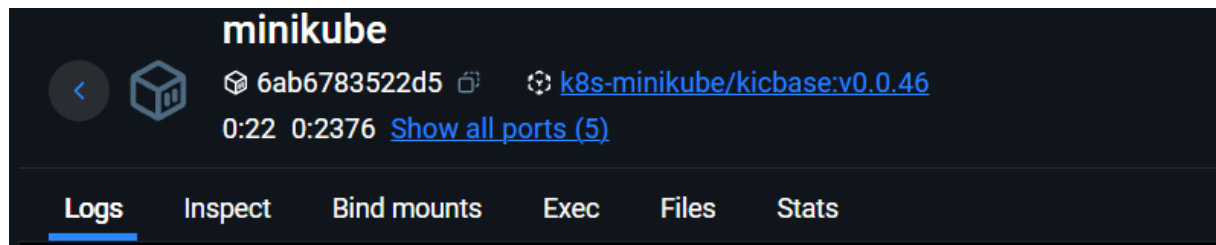


Photo 3: Docker screenshot

```
adrian2300@LAPTOP-LQ0MESR8:/mnt/c/Users/adria/OneDrive/Pulpit/Lab 6 - LSC$ curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubect1"
% Total % Received % Xferd Average Speed Time Time Time Current
100 138 100 138 0 0 563 0 --:--:-- --:--:-- 563
100 54.6M 100 54.6M 0 0 9709K 0 0:00:05 0:00:05 --:--:-- 10.2M
adrian2300@LAPTOP-LQ0MESR8:/mnt/c/Users/adria/OneDrive/Pulpit/Lab 6 - LSC$ sudo install kubect1 /usr/local/bin/kubect1
adrian2300@LAPTOP-LQ0MESR8:/mnt/c/Users/adria/OneDrive/Pulpit/Lab 6 - LSC$ minikube start --driver=docker
minikube v1.35.0 on Ubuntu 22.04 (amd64)
* Using the docker driver based on user configuration
* Using Docker driver with root privileges
! For an improved experience it's recommended to use Docker Engine instead of Docker Desktop.
Docker Engine installation instructions: https://docs.docker.com/engine/install/#server
Starting "minikube" primary control-plane node in "minikube" cluster
Pulling base image v0.0.46 ...
Downloading Kubernetes v1.32.0 preload ...
> preloaded-images-k8s-v18-v1...: 333.57 MiB / 333.57 MiB 100.00% 4.37 Mi
> gcr.io/k8s-minikube/kicbase...: 500.31 MiB / 500.31 MiB 100.00% 5.84 Mi
Creating docker container (CPUs=2, Memory=2200MB) ...
Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
* Generating certificates and keys ...
* Booting up control plane ...
* Configuring RBAC rules ...
Configuring bridge CNI (Container Networking Interface) ...
Verifying Kubernetes components...
* Using image gcr.io/k8s-minikube/storage-provisioner:v5
Enabled addons: storage-provisioner, default-storageclass
Done! kubect1 is now configured to use "minikube" cluster and "default" namespace by default
```

Photo 4: Downloading and instalingl kubect1, starting minikube

## Download and install Helm

```
curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 | bash
```

```
adrian2300@LAPTOP-LQ0MESR8:/mnt/c/Users/adria/OneDrive/Pulpit/Lab 6 - LSC$ curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 | bash
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           % Done    0     0      0         0      0      0     0
100 11913  100 11913    0     0  102k      0  0 --:--:-- --:--:-- --:--:-- 102k
Downloading https://get.helm.sh/helm-v3.17.3-linux-amd64.tar.gz
Verifying checksum... Done.
Preparing to install helm into /usr/local/bin
[sudo] password for adrian2300:
helm installed into /usr/local/bin/helm
```

Photo 5: Downloading and installing Helm

## Install NFS Server/Provisioner with Helm

```
helm repo add stable https://charts.helm.sh/stable
```

```
helm repo update
```

```
helm install nfs-provisioner stable/nfs-server-provisioner \
```

```
--set persistence.enabled=true \
```

```
--set storageClass.name=nfs-storage
```

```
adrian2300@LAPTOP-LQ0MESR8:/mnt/c/Users/adria/OneDrive/Pulpit/Lab 6 - LSC$ helm repo add stable https://charts.helm.sh/stable
update
"stable" has been added to your repositories
adrian2300@LAPTOP-LQ0MESR8:/mnt/c/Users/adria/OneDrive/Pulpit/Lab 6 - LSC$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "stable" chart repository
Update Complete. ✎Happy Helming!✎
adrian2300@LAPTOP-LQ0MESR8:/mnt/c/Users/adria/OneDrive/Pulpit/Lab 6 - LSC$ helm install nfs-provisioner stable/nfs-server-provisioner \
> --set persistence.enabled=true \
> --set storageClass.name=nfs-storage
WARNING: This chart is deprecated
NAME: nfs-provisioner
LAST DEPLOYED: Tue Apr 22 15:41:31 2025
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The NFS Provisioner service has now been installed.

A storage class named 'nfs-storage' 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-storage"
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
```

Photo 6: Installing NFS Server/Provisioner with Helm

## Applying PVC/nginx-development/nginx-service/job

```
kubectl apply -f pvc.yaml
```

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

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

```
kubectl apply -f job.yaml
```

```
adrian2300@LAPTOP-LQ0MESR8:/mnt/c/Users/adria/OneDrive/Pulpit/Lab 6 - LSC$ kubectl apply -f pvc.yaml
pvc/nginx-deployment created
adrian2300@LAPTOP-LQ0MESR8:/mnt/c/Users/adria/OneDrive/Pulpit/Lab 6 - LSC$ kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
adrian2300@LAPTOP-LQ0MESR8:/mnt/c/Users/adria/OneDrive/Pulpit/Lab 6 - LSC$ kubectl apply -f nginx-service.yaml
service/nginx-service created
adrian2300@LAPTOP-LQ0MESR8:/mnt/c/Users/adria/OneDrive/Pulpit/Lab 6 - LSC$ kubectl apply -f job.yaml
job.batch/nfs-job created
```

Photo 7: Applying all yamls

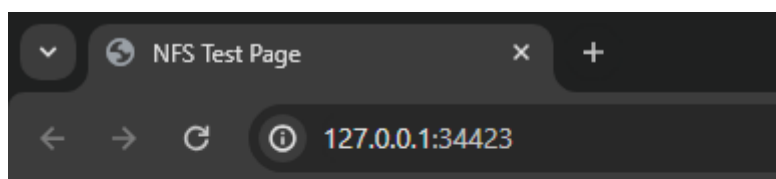
## Start HTTP server

```
minikube service nginx-service
```

```
adrian2300@LAPTOP-LQ0MESR8:/mnt/c/Users/adria/OneDrive/Pulpit/Lab 6 - LSC$ minikube service nginx-service
|-----|-----|-----|-----|
| NAMESPACE | NAME | TARGET PORT | URL |
|-----|-----|-----|-----|
| default | nginx-service | 80 | http://192.168.49.2:31742 |
|-----|-----|-----|-----|
🔗 Starting tunnel for service nginx-service.
|-----|-----|-----|-----|
| NAMESPACE | NAME | TARGET PORT | URL |
|-----|-----|-----|-----|
| default | nginx-service | | http://127.0.0.1:34423 |
|-----|-----|-----|-----|
🔗 Opening service default/nginx-service in default browser...
🔗 http://127.0.0.1:34423
! Because you are using a Docker driver on linux, the terminal needs to be open to run it.
```

Photo 8: Starting service

## Final effect



# Server is working!

This is a test page served from NFS.

Photo 9: Running app

# Files

## pvc.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: nfs-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: nfs-storage
```

Photo 10: pvc.yaml

## nginx-deployment.yaml

```
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
          ports:
            - containerPort: 80
          volumeMounts:
            - mountPath: /usr/share/nginx/html
              name: nfs-volume
      volumes:
        - name: nfs-volume
          persistentVolumeClaim:
            claimName: nfs-pvc
```

Photo 11: nginx-deployment.yaml

## nginx-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  type: NodePort
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

Photo 12: nginx-service.yaml

## job.yaml

```
apiVersion: batch/v1
kind: Job
metadata:
  name: nfs-job
spec:
  template:
    spec:
      containers:
        - name: writer
          image: busybox
          command: ["/bin/sh", "-c"]
          args:
            - |
              echo '<!DOCTYPE html>' > /data/index.html;
              echo '<html>' >> /data/index.html;
              echo '<head><title>NFS Test Page</title></head>' >> /data/index.html;
              echo '<body>' >> /data/index.html;
              echo '<h1>Server is working!</h1>' >> /data/index.html;
              echo '<p>This is a test page served from NFS.</p>' >> /data/index.html;
              echo '</body>' >> /data/index.html;
              echo '</html>' >> /data/index.html;
          volumeMounts:
            - name: nfs-volume
              mountPath: /data
      volumes:
        - name: nfs-volume
          persistentVolumeClaim:
            claimName: nfs-pvc
          restartPolicy: OnFailure
```

Photo 13: job.yaml