LSC - lab 6 - Kubernetes

Author: Michał Skałka

GitHub repository: https://github.com/Skalakid/large-scale-computing
Run the lab using run.sh script: ./run.sh

config.sh U

large-scale-computing > lab6 > config.sh

```
brew install minikube
brew install kind

minikube start

kubectl version --clientkubectl get nodes

curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 | bash
helm repo add stable https://charts.helm.sh/stable
helm repo update

helm install nfs-server stable/nfs-server-provisioner \
  --set persistence.enabled=true \
  --set persistence.size=1Gi \
  --set storageClass.name=nfs-storage

kubectl get storageclass


kubectl apply -f pvc.yaml
kubectl apply -f nginx-deployment.yaml
kubectl apply -f nginx-service.yaml
kubectl apply -f content-job.yaml

minikube service nginx-service
```

1. *Create a k8s cluster using Minikube*

```
~ minikube start
😄  minikube v1.35.0 on Darwin 15.3 (arm64)
✨  Automatically selected the docker driver
📌  Using Docker Desktop driver with root privileges
👍  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...:  314.92 MiB / 314.92 MiB  100.00% 5.01 Mi
    > gcr.io/k8s-minikube/kicbase...:  452.84 MiB / 452.84 MiB  100.00% 5.29 Mi
🔥  Creating docker container (CPUs=2, Memory=4600MB) ...
🐳  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: default-storageclass, storage-provisioner
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" name
space by default
```

2. *Using Helm, install an NFS server and provisioner in the cluster.*

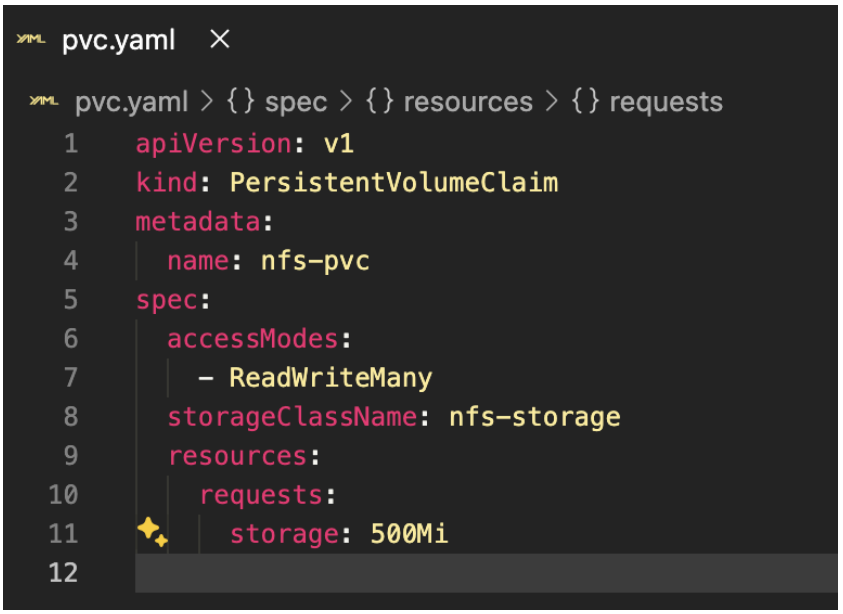
   The NFS (Network File System) server provides shared storage that can be accessed by multiple pods. We installed it in the cluster using a Helm chart called nfs-server-provisioner. It dynamically provisions Persistent Volumes (PVs) on demand. The Persistent Volume is a storage resource in the cluster provided by the NFS server. It abstracts the actual storage (NFS in this case) from the Kubernetes applications.

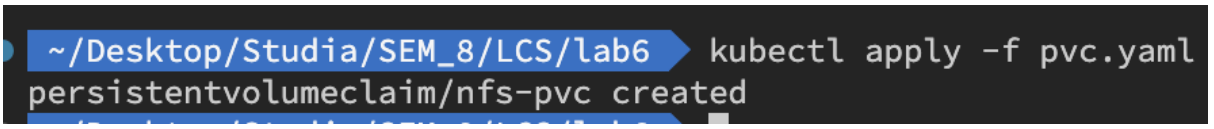
```
~/Desktop/Studia/SEM_8/LCS/lab6 curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 | bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 11913  100 11913    0     0  74823      0 --:--:-- --:--:-- --:--:-- 74456
Downloading https://get.helm.sh/helm-v3.17.3-darwin-arm64.tar.gz
Verifying checksum... Done.
Preparing to install helm into /usr/local/bin
Password:
helm installed into /usr/local/bin/helm
~/Desktop/Studia/SEM_8/LCS/lab6 helm repo add stable https://charts.helm.sh/stable
"stable" has been added to your repositories
~/Desktop/Studia/SEM_8/LCS/lab6 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!⎈
~/Desktop/Studia/SEM_8/LCS/lab6 helm install nfs-server stable/nfs-server-provisioner \
  --set persistence.enabled=true \
  --set persistence.size=1Gi \
  --set storageClass.name=nfs-storage
WARNING: This chart is deprecated
NAME: nfs-server
LAST DEPLOYED: Tue Apr 15 20:16:12 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
~/Desktop/Studia/SEM_8/LCS/lab6
```

```
~/Desktop/Studia/SEM_8/LCS/lab6 kubectl get storageclass
NAME                  PROVISIONER                                    RECLAIMPOLICY   VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
nfs-storage           cluster.local/nfs-server-nfs-server-provisioner   Delete          Immediate           true                   94s
standard (default)    k8s.io/minikube-hostpath                       Delete          Immediate           false                  3m50s
~/Desktop/Studia/SEM_8/LCS/lab6
```

3. *Create a Persistent Volume Claim which will bind to a NFS Persistent Volume provisioned dynamically by the provisioner installed in the previous step*

   Persistent Volume Claim (PVC) is a request for storage by a user/application. It will dynamically bind to the NFS Provisioner provisioned in the previous step.





4. *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*

   The Deployment manages the lifecycle of a set of identical pods (like the HTTP server). In this case, it deploys an nginx web server and ensures that the pod is running at all times. The deployment uses the PVC to mount the shared volume inside the pod.

pvc.yaml | nginx-deployment.yaml

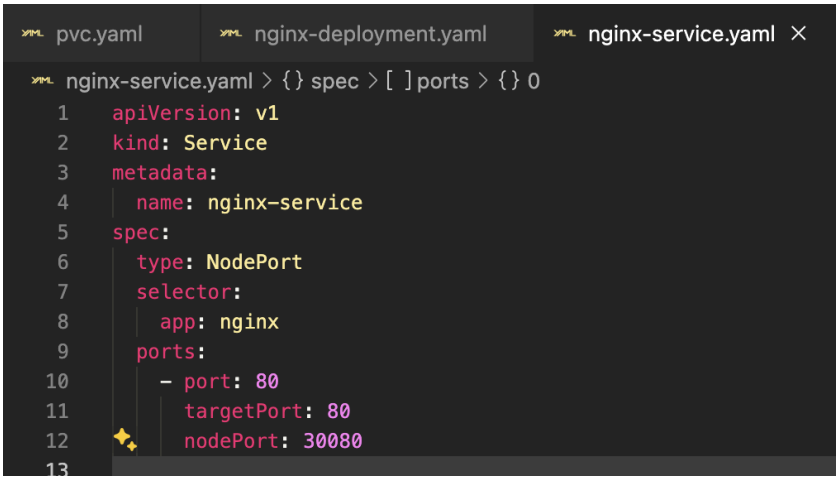
nginx-deployment.yaml > {} spec > {} template > {} spec > [ ] vol

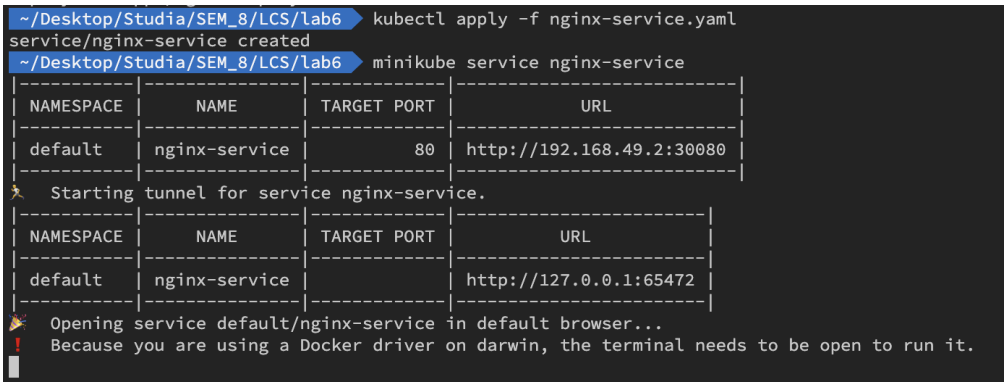
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx
          ports:
     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

```

```
~/Desktop/Studia/SEM_8/LCS/lab6 kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
~/Desktop/Studia/SEM_8/LCS/lab6
```

5. *Create a Service associated with the Pod(s) of the HTTP server Deployment.*

   A Service in Kubernetes is an abstraction that exposes a set of pods as a network service. In this case, the service is configured to expose the HTTP server pod(s) created by the Deployment





6. *Create a Job which mounts the PVC and copies a sample content through the shared NFS PV.*

   A Job in Kubernetes is used to run a task to completion. In this case, the Job is responsible for mounting the same PVC used by the HTTP server and copying sample index.htm into it.

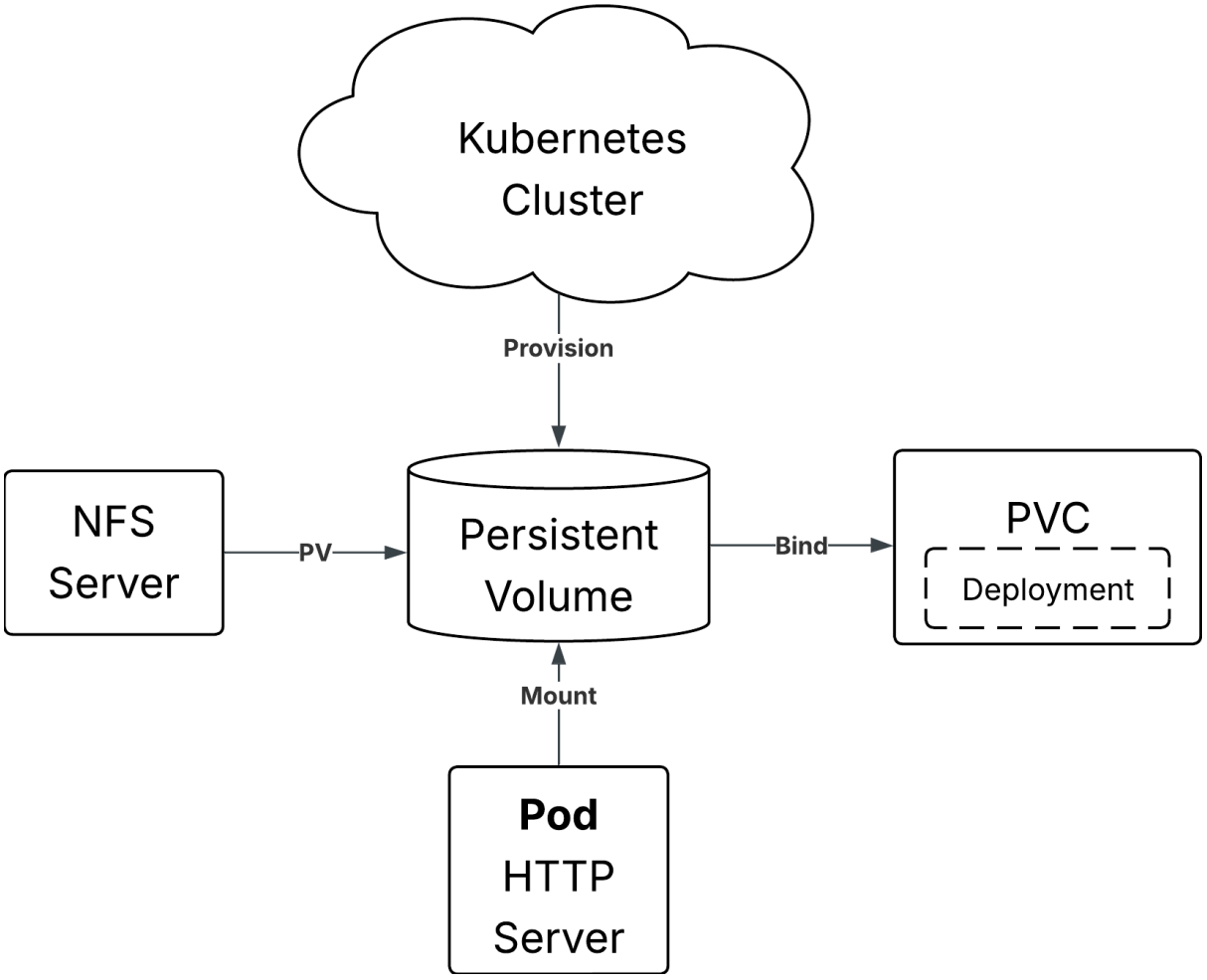
pvc.yaml | nginx-deployment.yaml | nginx-service.yaml | content-job.yaml

content-job.yaml > { } spec

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

```
~/Desktop/Studia/SEM_8/LCS/lab6 kubectl apply -f content-job.yaml
job.batch/copy-content-job created
~/Desktop/Studia/SEM_8/LCS/lab6 minikube service nginx-service
|-----------|---------------|-------------|---------------------------|
| NAMESPACE |     NAME      | TARGET PORT |            URL            |
|-----------|---------------|-------------|---------------------------|
| default   | nginx-service |          80 | http://192.168.49.2:30080 |
|-----------|---------------|-------------|---------------------------|
🏃  Starting tunnel for service nginx-service.
|-----------|---------------|-------------|------------------------|
| NAMESPACE |     NAME      | TARGET PORT |          URL           |
|-----------|---------------|-------------|------------------------|
| default   | nginx-service |             | http://127.0.0.1:65524 |
|-----------|---------------|-------------|------------------------|
🎉  Opening service default/nginx-service in default browser...
❗  Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
```

Diagram:



Provision: NFS provisioner automatically creates a Persistent Volume when a PVC is detected.
Bind: Kubernetes matches and binds the PVC to an available PV.
Mount: The PV is mounted into the pod so the container can use it as a normal directory.