Large Scale Computing - Kubernetes

Marcin Szwed

Minikube kubernetes cluster was created.

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

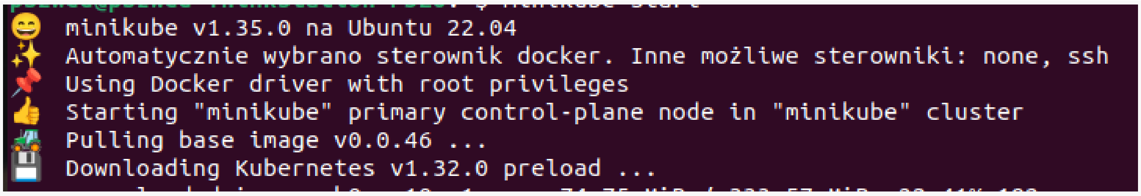
```
  curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"
echo "$(cat kubectl.sha256)  kubectl" | sha256sum --check
```

```
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

Minikube:

```
curl -LO
https://storage.googleapis.com/minikube/releases/latest/minikube_latest_a
md64.deb
sudo dpkg -i minikube_latest_amd64.deb
```

```
minikube start
```



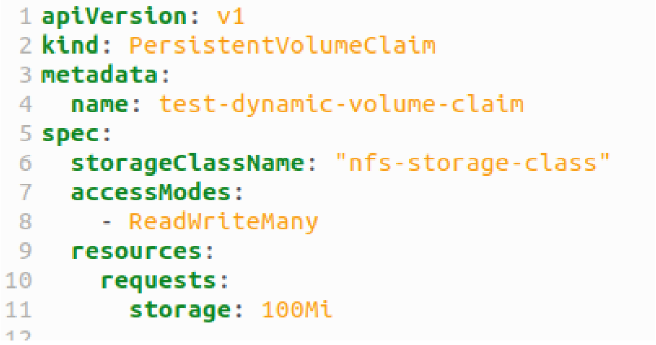
```
😄  minikube v1.35.0 na Ubuntu 22.04
✨  Automatycznie wybrano sterownik docker. Inne możliwe sterowniki: none, ssh
📌  Using Docker driver with root privileges
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.46 ...
💾  Downloading Kubernetes v1.32.0 preload ...
```

Then Helm was installed:

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

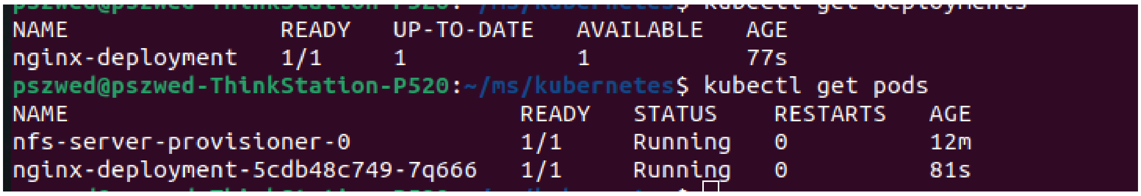
```
helm install nfs-server-provisioner stable/nfs-server-provisioner \
--set nfs.server=nfs-server-provisioner \ --set nfs.path=/export/nfs \
--set storageClass.name=nfs-storage-class
```

Nfs-pvc.yaml was created and executed.



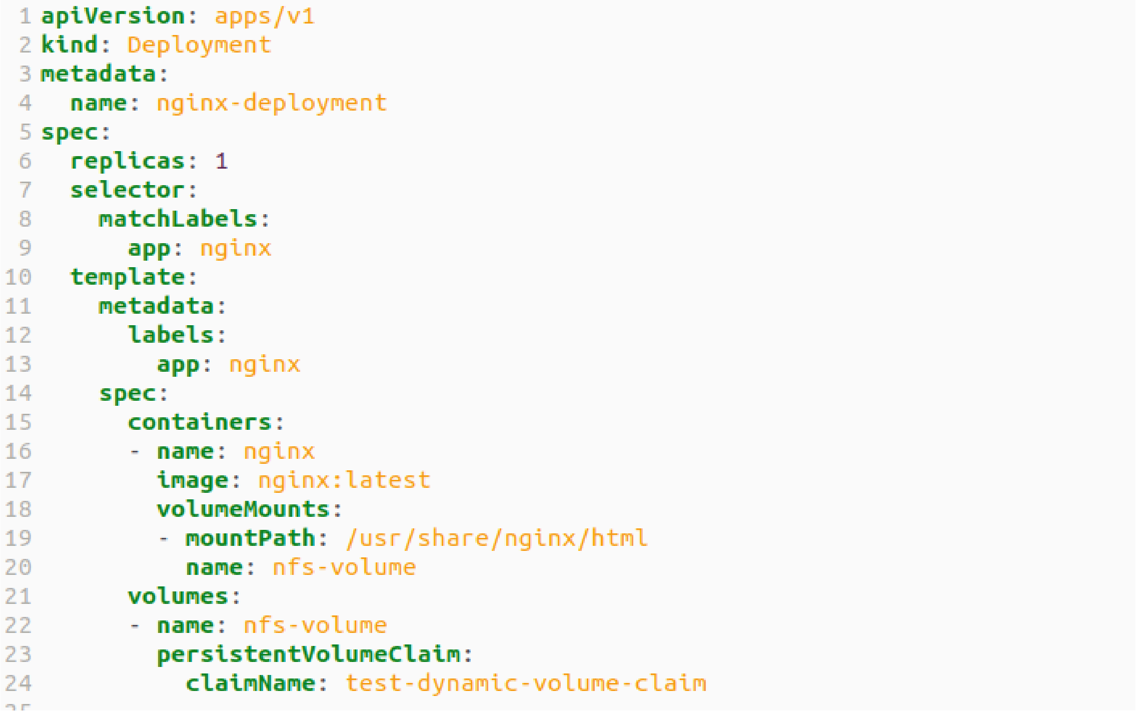
```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: test-dynamic-volume-claim
spec:
  storageClassName: "nfs-storage-class"
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Mi
```

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



```
pszwed@pszwed-ThinkStation-P520:~/ms/kubernetes$ kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   1/1     1            1           77s
pszwed@pszwed-ThinkStation-P520:~/ms/kubernetes$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nfs-server-provisioner-0            1/1     Running   0          12m
nginx-deployment-5cdb48c749-7q666   1/1     Running   0          81s
```

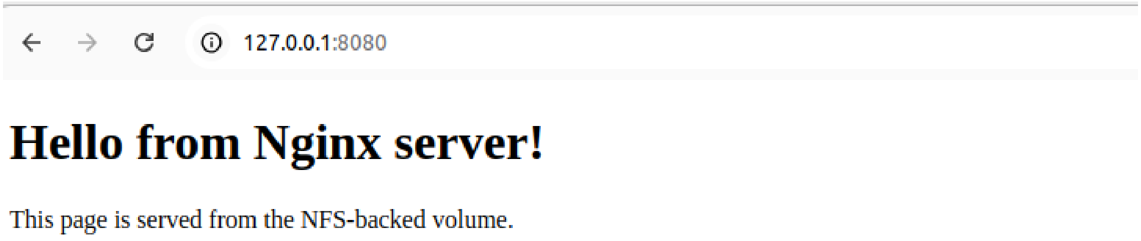
Nginx-deployment.yaml was created.



```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:latest
        volumeMounts:
        - mountPath: /usr/share/nginx/html
          name: nfs-volume
      volumes:
      - name: nfs-volume
        persistentVolumeClaim:
          claimName: test-dynamic-volume-claim
```

Then it was performed, first using port-forward.

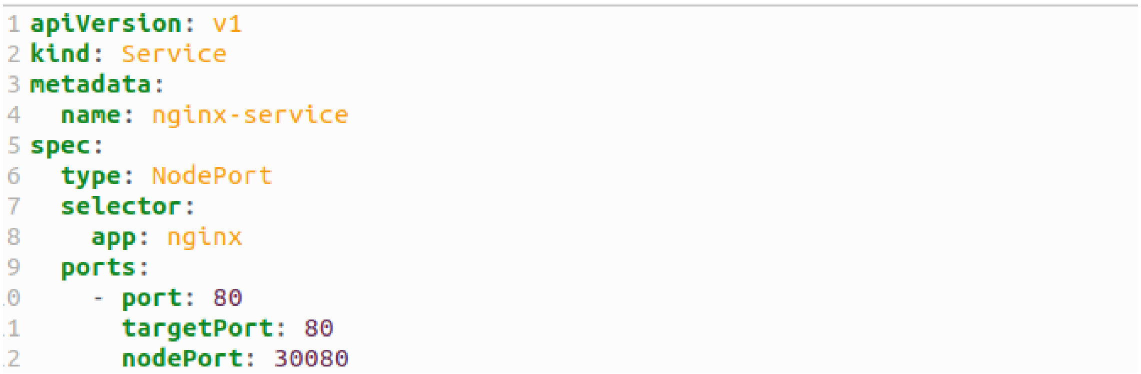
```
kubectl apply -f nginx-deployment.yaml
kubectl port-forward deployment/nginx-deployment 8080:80

kubectl cp ./index.html
nginx-deployment-5cdb48c749-7q666:/usr/share/nginx/html/index.html
```



Hello from Nginx server!

This page is served from the NFS-backed volume.

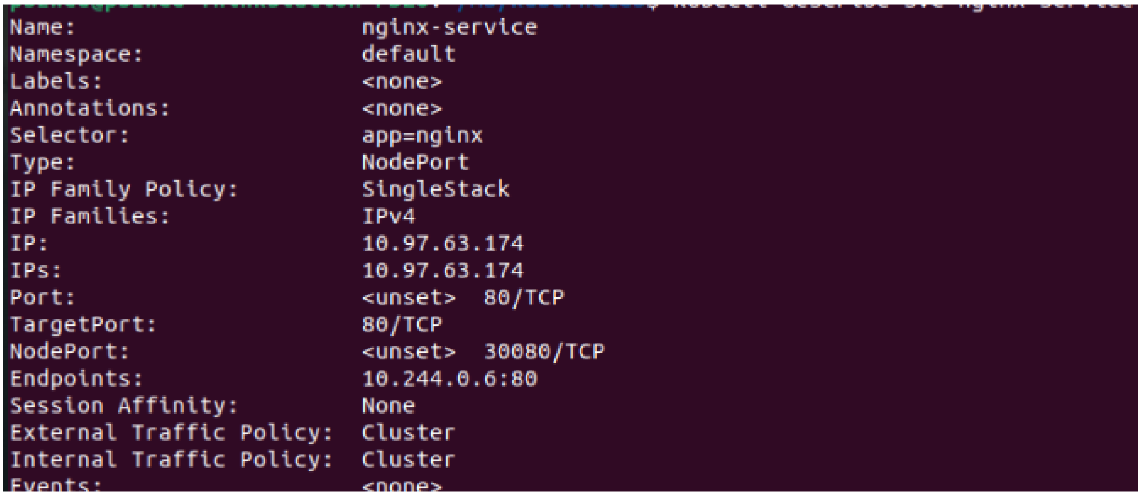
Nginx-service.yaml was created



```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  type: NodePort
  selector:
    app: nginx
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30080
```

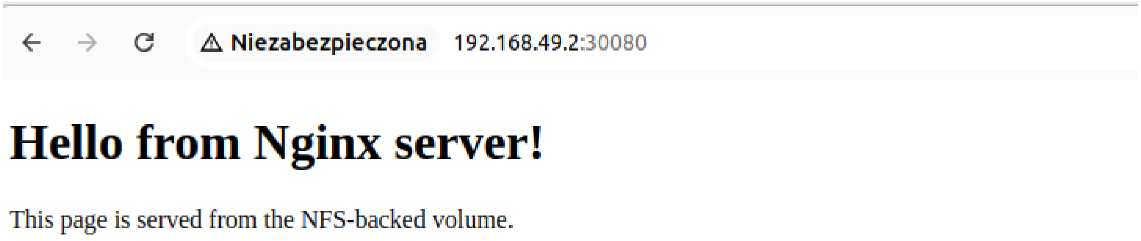
```
kubectl apply -f nginx-service.yaml
minikube ip
192.168.49.2
```

Service was running: `kubectl describe svc nginx-service`



```
Name:                     nginx-service
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=nginx
Type:                     NodePort
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.97.63.174
IPs:                      10.97.63.174
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  30080/TCP
Endpoints:                10.244.0.6:80
Session Affinity:         None
External Traffic Policy:  Cluster
Internal Traffic Policy:  Cluster
Events:                   <none>
```

It was possible to connect to the website.



Hello from Nginx server!

This page is served from the NFS-backed volume.

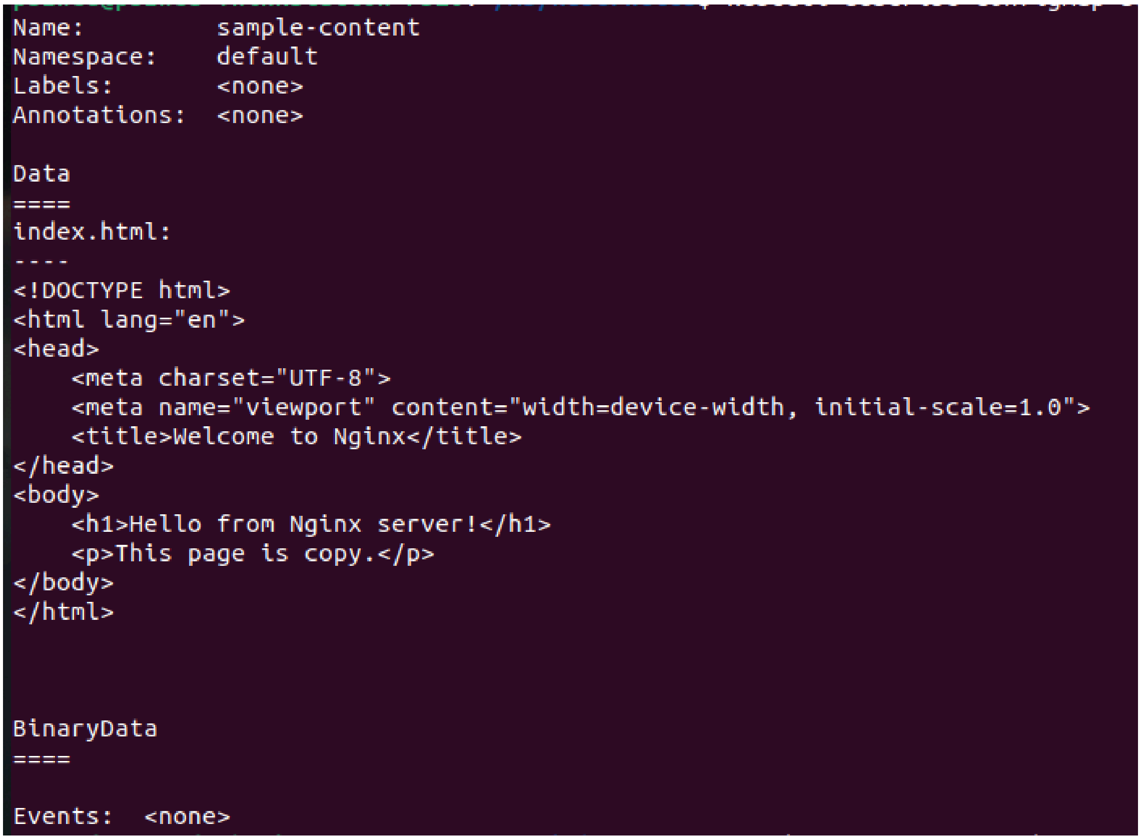
Copy file with a job

Next task was to create a job, which mount the PVC and copies website content.

ConfigMap was created.

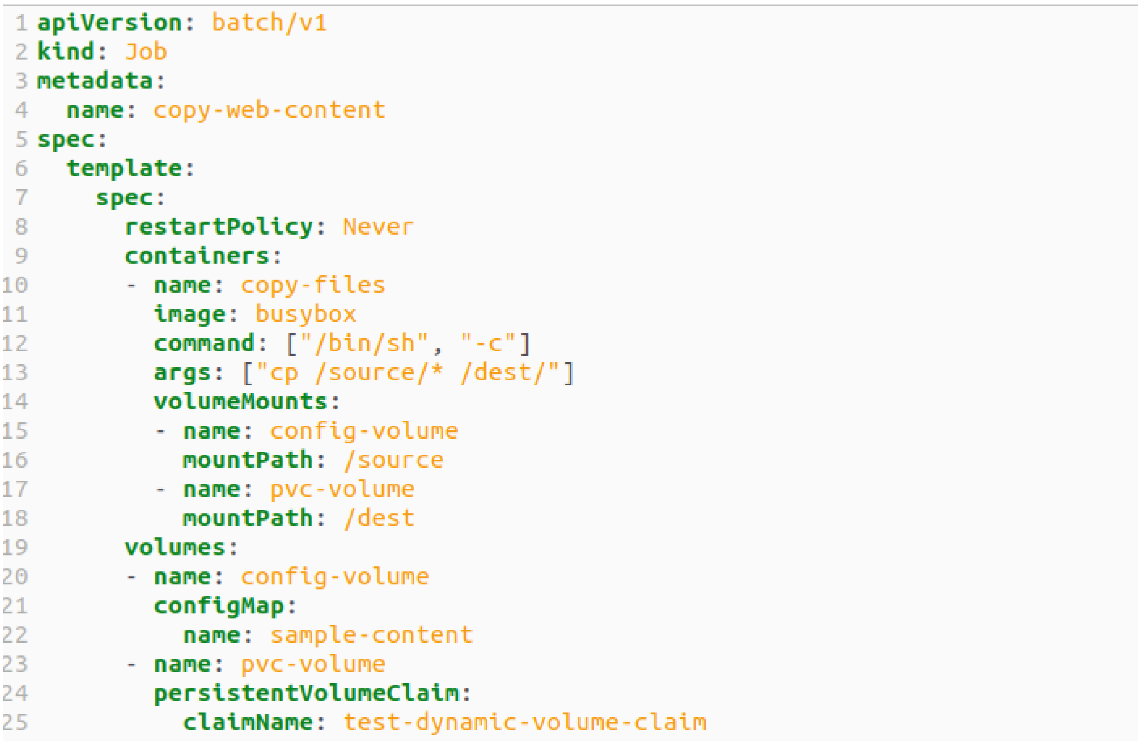
```
kubectl create configmap sample-content --from-file=web-content/
```

```
kubectl describe configmap sample-content
```



```
Name:         sample-content
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
index.html:
----
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Welcome to Nginx</title>
</head>
<body>
    <h1>Hello from Nginx server!</h1>
    <p>This page is copy.</p>
</body>
</html>



BinaryData
====

Events:  <none>
```

Copy-content-job.yaml was created.

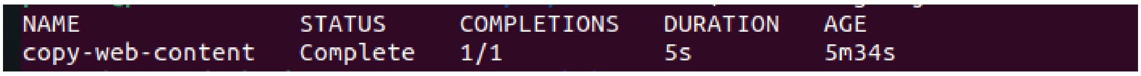


```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: copy-web-content
spec:
  template:
    spec:
      restartPolicy: Never
      containers:
      - name: copy-files
        image: busybox
        command: ["/bin/sh", "-c"]
        args: ["cp /source/* /dest/"]
        volumeMounts:
        - name: config-volume
          mountPath: /source
        - name: pvc-volume
          mountPath: /dest
      volumes:
      - name: config-volume
        configMap:
          name: sample-content
      - name: pvc-volume
        persistentVolumeClaim:
          claimName: test-dynamic-volume-claim
```

And started.

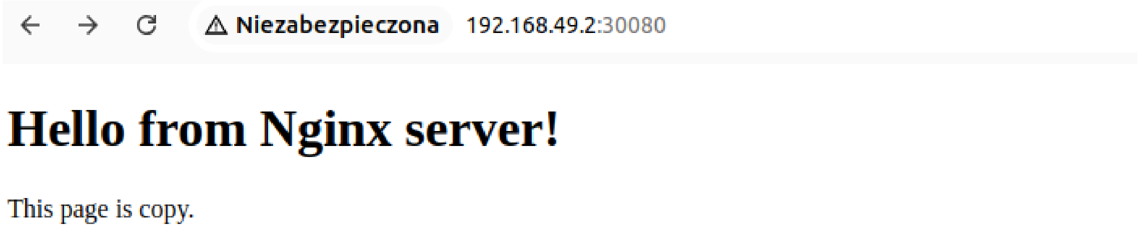
kubectl apply -f copy-content-job.yaml

Kubectl get jobs



```
NAME               STATUS     COMPLETIONS   DURATION   AGE
copy-web-content   Complete   1/1           5s         5m34s
```

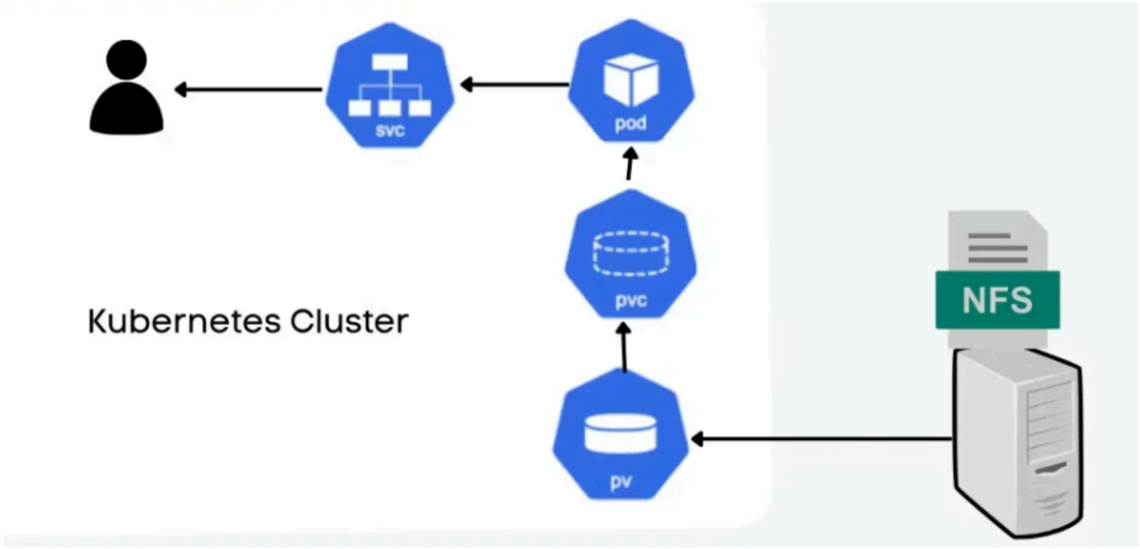
The website files were updated.



Hello from Nginx server!

This page is copy.

Diagram



NFS - protocol that allows multiple machines to share the same filesystem over a network. In this implementation internal NFS Server was used.

Pod - smallest deployable unit in kubernetes, runs one or more containers that share the same network and storage.

PVC - request for persistent storage by a user.

PV - persistent volume, physical volume, provisioned by administrator or dynamically, assigned to PVC.

SVC - service that defines how to access pods, provides endpoint for communication. In this implementation allows to connect via IP:port.