Large Scale Computing

Lab 6

Adrian Madej

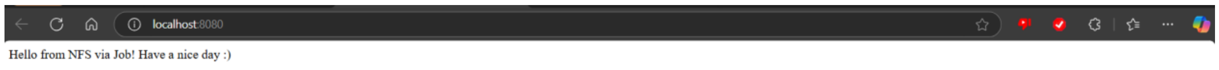
1. Short description of running the application

As part of the project, a web application was deployed in a local Kubernetes cluster using kind. The application includes the following components:

- An NFS server with a dynamic provisioner, installed via Helm,
- A PersistentVolumeClaim with ReadWriteMany access mode,
- A Pod running an nginx HTTP server that mounts the NFS volume,
- A Kubernetes Job that writes a sample index.html file to the shared volume,
- A NodePort Service that exposes the HTTP server for external access.

The entire configuration and commands used to deploy the application are included in the GitHub repository - Large-Scale-Computing/lab_6 at main · ltmollo/Large-Scale-Computing .

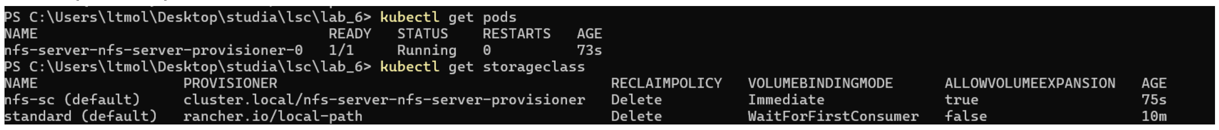
A screenshot of a web page:



localhost:8080

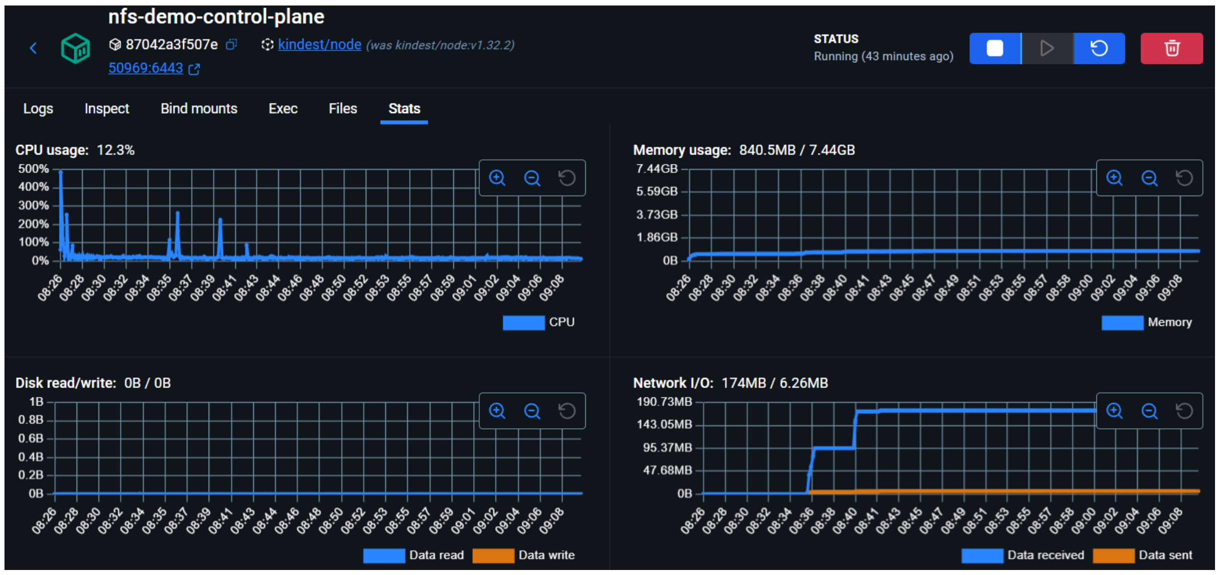
Hello from NFS via Job! Have a nice day :)

Example outputs:



```
PS C:\Users\ltmol\Desktop\studia\lsc\lab_6> kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
nfs-server-nfs-server-provisioner-0   1/1     Running   0          73s
PS C:\Users\ltmol\Desktop\studia\lsc\lab_6> kubectl get storageclass
NAME                 PROVISIONER                                       RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
nfs-sc (default)     cluster.local/nfs-server-nfs-server-provisioner   Delete          Immediate              true                   75s
standard (default)   rancher.io/local-path                             Delete          WaitForFirstConsumer   false                  10m
```

Stats



nfs-demo-control-plane

87042a3f507e kindest/node (was kindest/node:v1.32.2)

50969:6443

STATUS
Running (43 minutes ago)

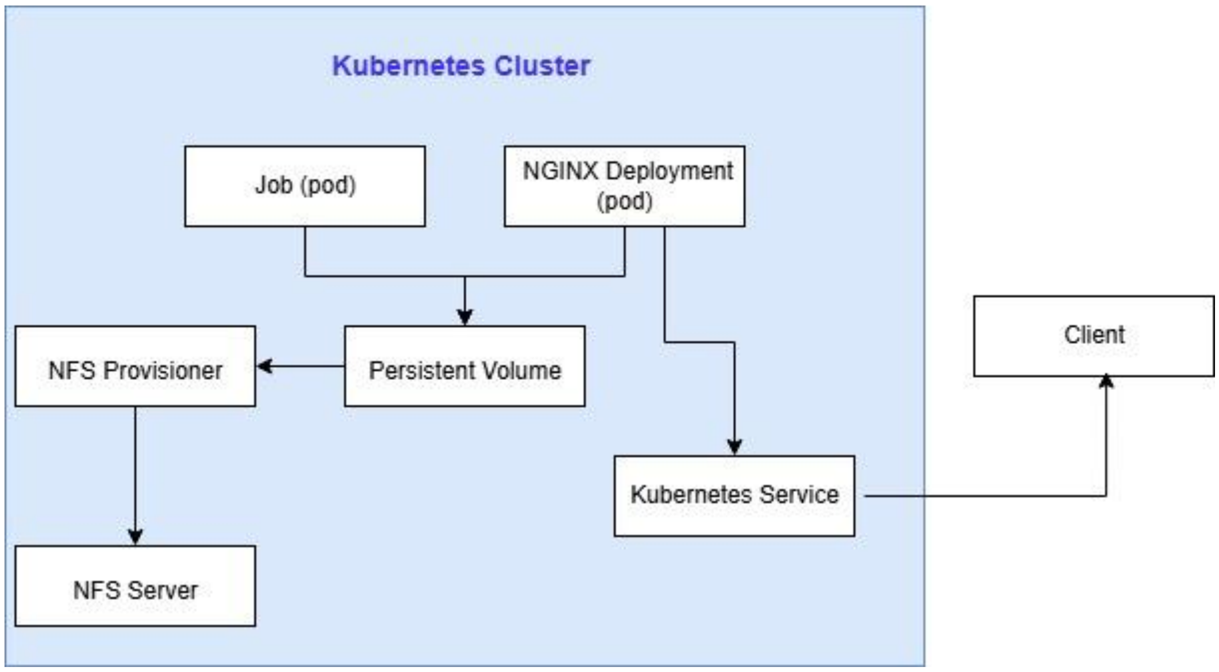
CPU usage: 12.3%

Memory usage: 840.5MB / 7.44GB

Disk read/write: 0B / 0B

Network I/O: 174MB / 6.26MB

2. Architecture diagram of the created application



3. Component roles and connections

- **Job**
  - One-time Kubernetes pod that mounts the shared Persistent Volume Claim (PVC).
  - Writes a sample `index.html` file into the volume.
  - Ensures the web server has content to serve.
- **Deployment**
  - Manages a pod running an `nginx:alpine` container.
  - Mounts the same PVC used by the Job.
  - Serves static web content over HTTP.
- **Persistent Volume (PV)**
  - Dynamically created by the NFS provisioner in response to a PVC request.
  - Backed by the NFS server and supports `ReadWriteMany`.
  - Acts as shared storage between the Job and the NGINX pod.
- **NFS Provisioner**
  - Deployed via Helm.
  - Listens for PVC requests and dynamically provisions NFS-backed PVs.
  - Works with the custom StorageClass `nfs-sc`.
- **NFS Server**
  - Pod running NFS-Ganesha within the Kubernetes cluster.
  - Provides the physical file system backing the dynamically created PVs.
  - Shared between multiple pods via NFS protocol.

- **Kubernetes Service**
    - Type `NodePort` service that exposes the NGINX pod externally.
    - Allows the user to access the web app via `http://localhost:8080`.
    - Routes external HTTP traffic to the correct pod inside the cluster.

Persistent Storage in Kubernetes
• By default, pod storage in Kubernetes is **ephemeral** – it disappears when the pod is deleted or restarted.
• **Persistent Volumes (PVs)** provide long-term, reliable storage that lives independently of pods.
• Applications can **claim** persistent storage by creating a **Persistent Volume Claim (PVC)**.
• In this project, persistent storage is backed by an internal **NFS server**, making it possible to share data between multiple pods.
• The **ReadWriteMany (RWX)** access mode allows the volume to be mounted by more than one pod at the same time.
• This setup ensures that:

- The **Job** can write files (e.g., `index.html`) to the volume
- The **NGINX pod** can serve those files through HTTP
  • Persistent storage is crucial for applications that:
- Need to **retain data** between pod restarts
- Share files across **multiple pods**
- Store user-generated content or configuration