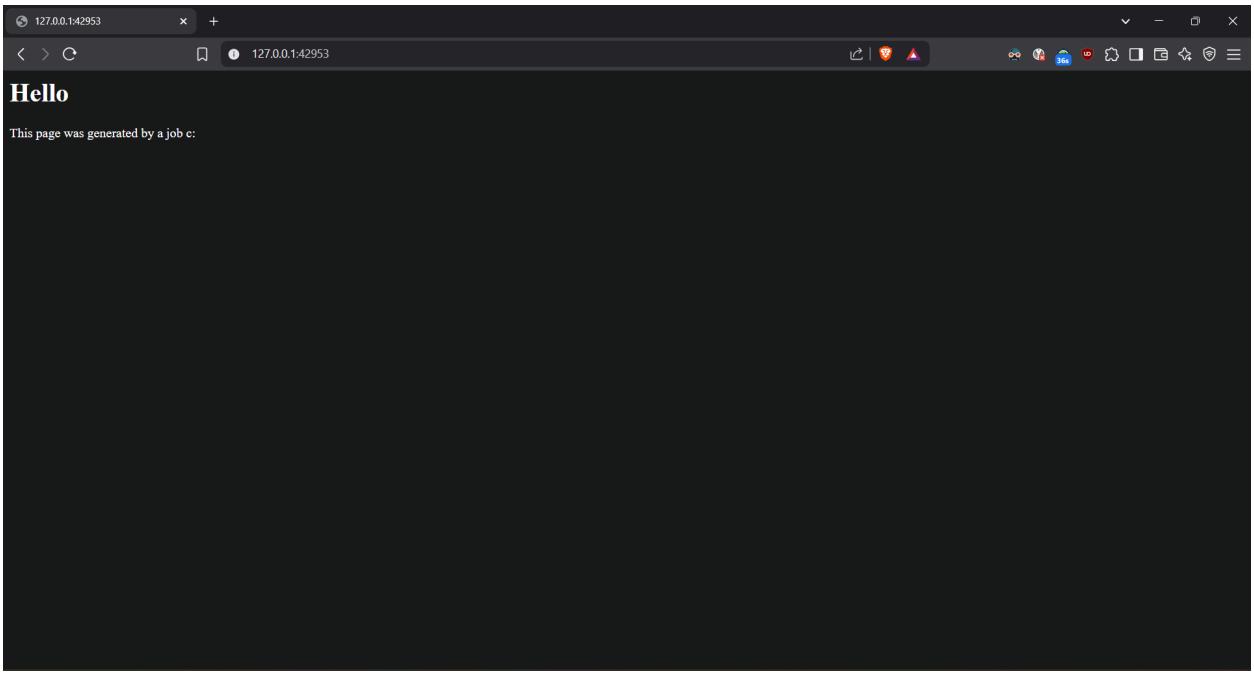
Lab 6 - Kubernetes application

Piotr Wiercigroch

April 11, 2025

1 Results

After deploying the application according to steps provided in *README.md* the result is a working nginx deployment with a load balancer service. The exposed service, list of pods and the web page can be seen on below figures.




```
pedro@MSI:~$ kubectl get svc
NAME                 TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)
                                                                   AGE
kubernetes           ClusterIP      10.96.0.1        <none>        443/TCP
                                                                   16m
lsc-wsd-service      LoadBalancer   10.104.236.120   <pending>     80:31481/TCP
                                                                   15m
test-nfsprovisioner  ClusterIP      10.99.160.242    <none>        2049/TCP,2049/UDP,32803/TCP,32803/UDP,20048/TCP,2004
8/UDP,875/TCP,875/UDP,111/TCP,111/UDP,662/TCP,662/UDP   15m
pedro@MSI:~$ minikube service lsc-wsd-service --url
http://127.0.0.1:42953
❗  Because you are using a Docker driver on linux, the terminal needs to be open to run it.
```

```
pedro@MSI:~/stdia$ kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
lsc-wsd-deployment-697cbc95b4-x2l6j     1/1     Running   0          91m
test-nfsprovisioner-0                   1/1     Running   0          91m
```

2 Explanation of deployment process

The application uses helm to automate the deployment process. If one were to do it by hand the steps would need to be as follows:

2.1 Install minikube and helm

Using the commands presented in *README.md* minikube and helm needs to be installed.

2.2 Install *nfs-server-provisioner*

```
helm install nfs-server-provisioner \
  nfs-ganesha-server-and-external-provisioner/nfs-server-provisioner \
  --set=... \
  --set=... \
```

Where each set argument overrides default values according to **nfsprovisioner** section of `/lsc-wsd/values.yaml`

2.3 Apply yaml files

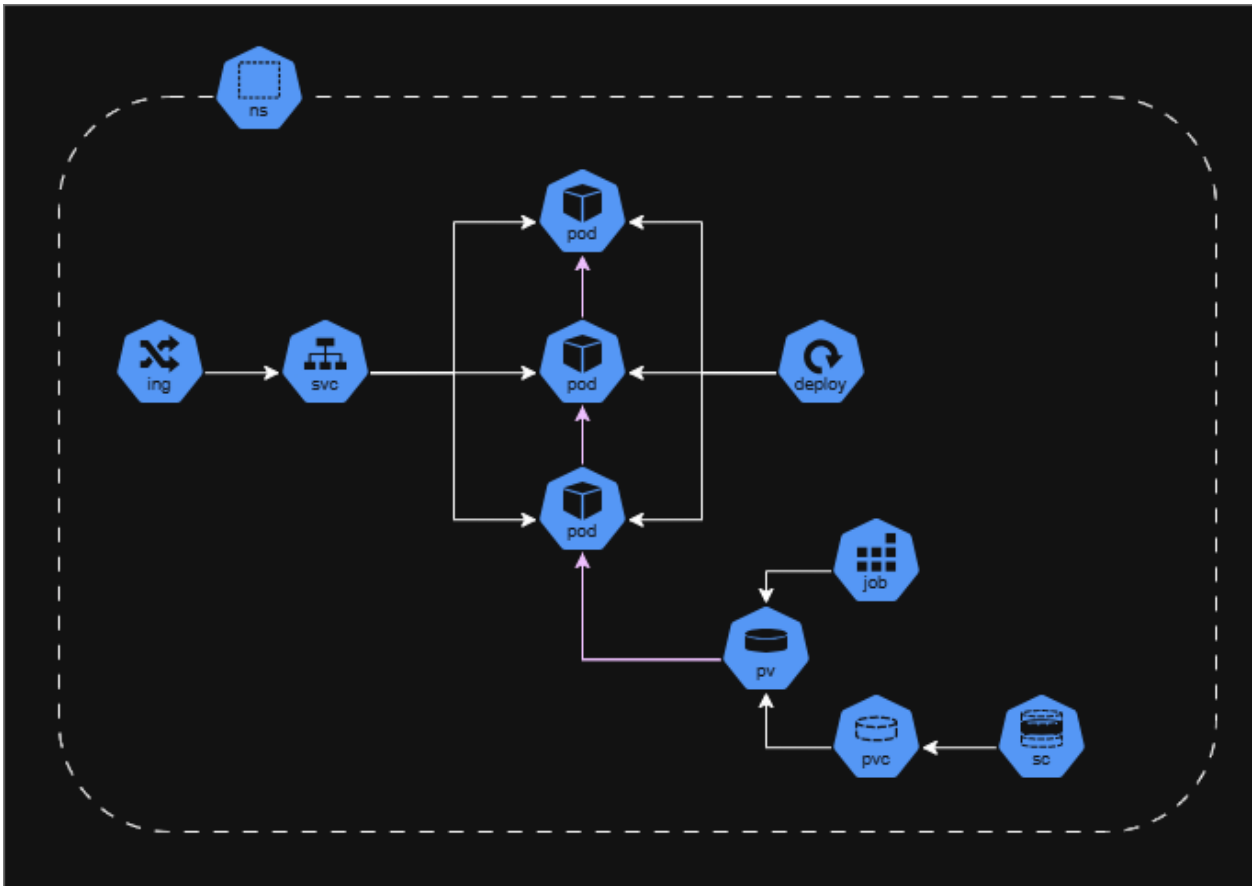
Than each yaml file in directory `/lsc-wsd/templates` needs to be applied to the cluster. Before that they need to be populated by values from `/lsc-wsd/values.yaml`. One can do it by had or use the `helm template lsc-wsd` command which runs just the template engine and outputs a yaml to be applied. The yamls have proper dependencies and hooks, nontheless the preferable order for applying would be:

- pvc.yaml
- deployment.yaml
- job.yaml
- service.yaml

Applying is done by issuing the command `kubectl apply -f <the yaml to apply>`

3 Architecture

3.1 Diagram



3.2 Explanation

3.2.1 Storage Class

Storage Class is an identifyier created by *nfs-server-provisioner*. It denotes a class of storage which is provisioned dynamicly by the provisioner responsible for this Storage Class. When a **PVC** is created with this class as the StorageClass a request to *nfs-server-rpvisioner* is issued to create a **PV** with given parameters.

3.2.2 Persistent Volume

A Persistent Volume is a piece of storage in the cluster that has been provisioned by an administrator or dynamically provisioned using Storage Class. In this application the **PV** is created dynamically.

3.2.3 Persistent Volume Claim

Persistent Volume Claim is a request for storage by a user. Claims can request specific size and access modes (e.g., they can be mounted once read/write or many times read-only).

3.2.4 Pod

A Pod is the basic building block of Kubernetes. Pods run the containers. A Pod encapsulates an application's container (or, in some cases, multiple containers), storage resources, and a network identity.

In this application each pod runs one container with nginx server.

3.2.5 Deployment

A Deployment is a Kubernetes object that manages replicated applications. Deployments manage Pods, creating, updating, and deleting them.

3.2.6 Service

A Service is a way to expose an application running on a set of Pods as a network service. A Service abstracts away the underlying Pods and provides a stable endpoint for accessing the application.

3.2.7 Job

A job is an instance representing some small computations. It is a sort of short lived pod which goal is to perform some one time action.