# Cloud & Automation Class

Minikube II

Adam Lahbib - Skander Soltane - Mouna Rouini - Mariem Cherif

3/27/2023 @ INSAT

# Contents

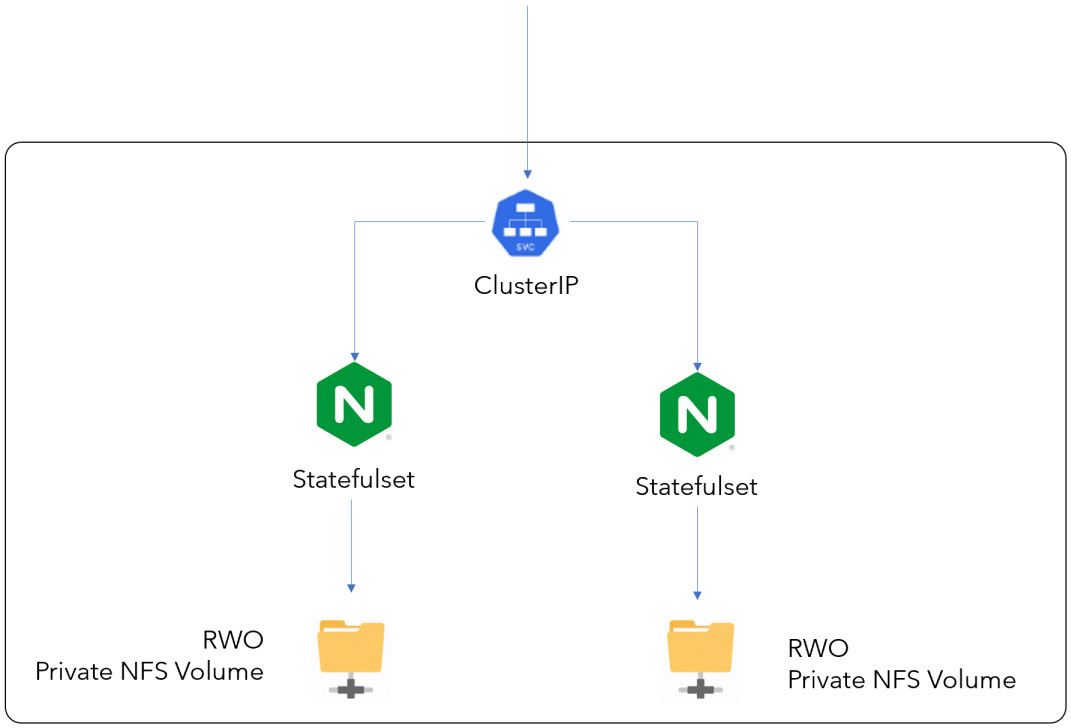
# 1 Lab Introduction

## 1.1 Objectives

This lab shows a step-by-step guide to create a web front-end in Kubernetes using a statefulset and a service. Designed to help a developer create a work environment from scratch. This lab emphasizes the features and characteristics of Kubernetes as an orchestrator for container-based workloads.

## 1.2 Architecture

- A service is created to expose the pods to the outside world.
- A statefulset is created to create two pods connecting to the volume claim template.
- HTML files are stored in a volume claim template, two ReadWriteOnce private NFS volumes.

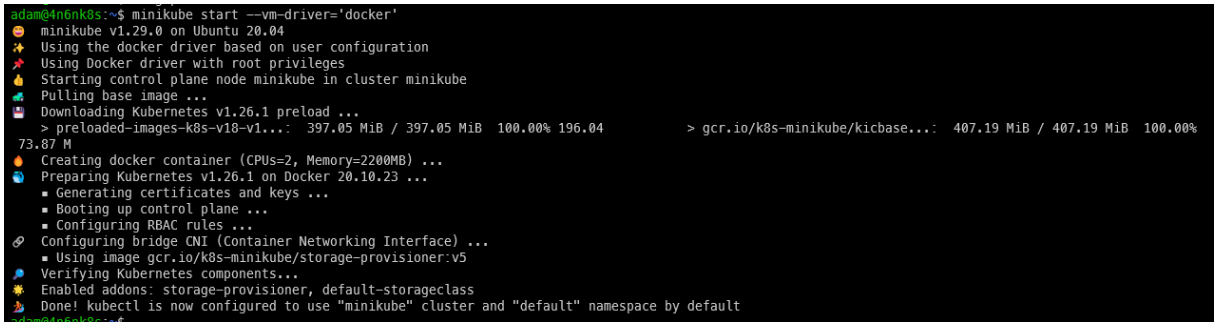


## 1.3 Requirements

- Ubuntu 20.04
- Minikube
- Kubectl
- Docker CLI

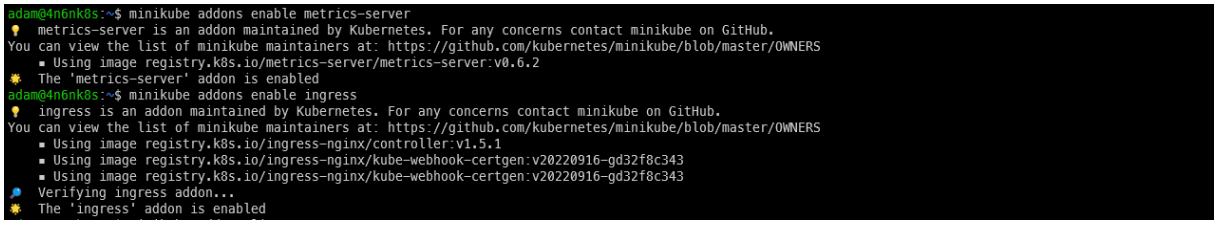
# 2 Lab Walkthrough

### 2.0.1 Starting Minikube




Minikube with the Docker driver is the recommended way to run Kubernetes locally. The Docker driver is the default driver for Minikube. It uses Docker to run Kubernetes components inside a VM on your local machine.

### 2.0.2 Enabling Addons



```
adam@4n6nk8s:~$ minikube addons list
|-----------------------------|----------|--------------|--------------------------------|
|         ADDON NAME          | PROFILE  |    STATUS    |           MAINTAINER           |
|-----------------------------|----------|--------------|--------------------------------|
| ambassador                  | minikube | disabled     | 3rd party (Ambassador)         |
| auto-pause                  | minikube | disabled     | Google                         |
|                             |
| cloud-spanner               | minikube | disabled     | Google                         |
|                             |
| csi-hostpath-driver         | minikube | disabled     | Kubernetes                     |
| dashboard                   | minikube | disabled     | Kubernetes                     |
| default-storageclass        | minikube | enabled ✅   | Kubernetes                     |
| efk                         | minikube | disabled     | 3rd party (Elastic)            |
| freshpod                    | minikube | disabled     | Google                         |
|                             |
| gcp-auth                    | minikube | disabled     | Google                         |
|                             |
| gvisor                      | minikube | disabled     | Google                         |
|                             |
| headlamp                    | minikube | disabled     | 3rd party (kinvolk.io)         |
| helm-tiller                 | minikube | disabled     | 3rd party (Helm)               |
| inaccel                     | minikube | disabled     | 3rd party (InAccel             |
|                             |          |              | [info@inaccel.com])            |
| ingress                     | minikube | enabled ✅   | Kubernetes                     |
| ingress-dns                 | minikube | disabled     | Google                         |
|                             |
| istio                       | minikube | disabled     | 3rd party (Istio)              |
| istio-provisioner           | minikube | disabled     | 3rd party (Istio)              |
| kong                        | minikube | disabled     | 3rd party (Kong HQ)            |
| kubevirt                    | minikube | disabled     | 3rd party (KubeVirt)           |
| logviewer                   | minikube | disabled     | 3rd party (unknown)            |
| metallb                     | minikube | disabled     | 3rd party (MetalLB)            |
| metrics-server              | minikube | enabled ✅   | Kubernetes                     |
| nvidia-driver-installer     | minikube | disabled     | Google                         |
|                             |
| nvidia-gpu-device-plugin    | minikube | disabled     | 3rd party (Nvidia)             |
| olm                         | minikube | disabled     | 3rd party (Operator Framework) |
| pod-security-policy         | minikube | disabled     | 3rd party (unknown)            |
| portainer                   | minikube | disabled     | 3rd party (Portainer.io)       |
| registry                    | minikube | disabled     | Google                         |
|                             |
| registry-aliases            | minikube | disabled     | 3rd party (unknown)            |
adam@4n6nk8s:~$
| storage-provisioner         | minikube | enabled ✅   | Google                         |
|                             |
| storage-provisioner-gluster | minikube | disabled     | 3rd party (Gluster)            |
| volumesnapshots             | minikube | disabled     | Kubernetes                     |
|-----------------------------|----------|--------------|--------------------------------|
adam@4n6nk8s:~$
```

The metrics-server is a cluster-wide aggregator of resource usage data. Metrics Server collects resource metrics from Kubelets and exposes them in Kubernetes apiserver through the metrics API for use by Horizontal Pod Autoscaler and Vertical Pod Autoscaler. Metrics API can also be accessed by kubectl top, allowing Kubernetes users to check the amount of resources used by their applications.

The ingress addon is a collection of resources that allow you to expose HTTP and HTTPS routes from outside the cluster to services within the cluster. Traffic routing is controlled by rules defined on the Ingress resource.

### 2.0.3 Cluster Ready

```
adam@4n6nk8s:~$ kubectl get no
NAME       STATUS   ROLES           AGE     VERSION
minikube   Ready    control-plane   5m34s   v1.26.1
adam@4n6nk8s:~$
```

## 2.1 Creating NGINX

Provided:

- nginx.yml file

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  type: ClusterIP
  ports:
    - port: 80
      name: web
  selector:
    app: nginx
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  serviceName: "nginx"
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx
          ports:
            - containerPort: 80
              name: web
          volumeMounts:
            - name: www
              mountPath: /usr/share/nginx/html
  volumeClaimTemplates:
    - metadata:
        name: www
      spec:
        accessModes: ["ReadWriteOnce"]
        resources:
          requests:
            storage: 1Gi
        storageClassName: "standard"
```

This file creates a service and a statefulset. The statefulset creates two pods with nginx image. The service is used to access the pods. The service is a clusterIP service, so it is only accessible from inside the cluster!

### 2.1.1 Creating a service and statefulset

```
adam@4n6nk8s:~$ vim nginx.yml
adam@4n6nk8s:~$ kubectl apply -f nginx.yml
service/nginx created
statefulset.apps/web created
adam@4n6nk8s:~$ kubectl get pvc
NAME        STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
www-web-0   Bound    pvc-d9ba09bd-e339-4359-b195-133b12a4a0db   1Gi        RWO            standard       45s
www-web-1   Bound    pvc-2e14dc43-03a9-4a19-847a-4e8c74ae53eb   1Gi        RWO            standard       37s
adam@4n6nk8s:~$ kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM               STORAGECLASS   REASON   AGE
pvc-2e14dc43-03a9-4a19-847a-4e8c74ae53eb   1Gi        RWO            Delete           Bound    default/www-web-1   standard                52s
pvc-d9ba09bd-e339-4359-b195-133b12a4a0db   1Gi        RWO            Delete           Bound    default/www-web-0   standard                60s
adam@4n6nk8s:~$ kubectl get po
NAME    READY   STATUS    RESTARTS   AGE
web-0   1/1     Running   0          67s
web-1   1/1     Running   0          59s
adam@4n6nk8s:~$ kubectl get svc
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP   8m48s
nginx        ClusterIP   10.101.118.186   <none>        80/TCP    76s
adam@4n6nk8s:~$
```

A service is created with the name `nginx` and a statefulset is created with the name `web`. The statefulset creates two pods with the name `web-0` and `web-1`.

### 2.1.2 Adding HTML to pods

#### 2.1.2.1 Curling the service internally

```
adam@4n6nk8s:~$ kubectl exec web-1 -- curl 10.101.118.186:80
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx/1.23.3</center>
</body>
</html>
100   153  100   153    0     0   149k      0 --:--:-- --:--:-- --:--:--  149k
adam@4n6nk8s:~$ kubectl exec web-0 -- curl 10.101.118.186:80
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx/1.23.3</center>
</body>
</html>
100   153  100   153    0     0  21857      0 --:--:-- --:--:-- --:--:-- 21857
adam@4n6nk8s:~$
```

We got 403 forbidden when trying to curl the homepage of the service from both pods, this is because we don't have any html file being served by NGINX.

Let's check `web-0` pod's `nginx/html` contents:

```
adam@4n6nk8s:~$ kubectl exec -it web-0 -- /bin/bash
root@web-0:/# cd /usr/share/nginx/html/
root@web-0:/usr/share/nginx/html# ls -la
total 8
drwxrwxrwx 2 root root 4096 Mar 27 10:12 .
drwxr-xr-x 3 root root 4096 Mar 23 16:00 ..
root@web-0:/usr/share/nginx/html#
```

We can see that the `nginx/html` directory is empty. Let's add some html files to it.

#### 2.1.2.2 Creating a simple hello world homepage

```
adam@4n6nk8s:~$ kubectl exec -it web-0 -- /bin/bash
root@web-0:/# cd /usr/share/nginx/html/
root@web-0:/usr/share/nginx/html# ls -la
total 8
drwxrwxrwx 2 root root 4096 Mar 27 10:12 .
drwxr-xr-x 3 root root 4096 Mar 23 16:00 ..
root@web-0:/usr/share/nginx/html# cat <<EOF > /usr/share/nginx/html/index.html
<html>
<head><title>Hello World from web-0</title></head>
<body>
<center><h1>Hello World from web-0</h1></center>
<hr><center>nginx/1.15.12</center>
</body>
</html>
EOF
root@web-0:/usr/share/nginx/html# exit
exit
adam@4n6nk8s:~$
```

Let's repeat the process for `web-1` pod as well!

```
adam@4n6nk8s:~$ kubectl exec -it web-1 -- /bin/bash
root@web-1:/# cd /usr/share/nginx/html
root@web-1:/usr/share/nginx/html# ls -la
total 8
drwxrwxrwx 2 root root 4096 Mar 27 10:12 .
drwxr-xr-x 3 root root 4096 Mar 23 16:00 ..
root@web-1:/usr/share/nginx/html#
root@web-1:/usr/share/nginx/html#
root@web-1:/usr/share/nginx/html#
root@web-1:/usr/share/nginx/html# cat <<EOF > /usr/share/nginx/html/index.html
<html>
<head><title>Hello World from web-1</title></head>
<body>
<center><h1>Hello World from web-1</h1></center>
<hr><center>nginx/1.15.12</center>
</body>
</html>
EOF
root@web-1:/usr/share/nginx/html#
root@web-1:/usr/share/nginx/html#
root@web-1:/usr/share/nginx/html# exit
exit
adam@4n6nk8s:~$
```

#### 2.1.2.3 Curling the service again

```
adam@4n6nk8s:~$ kubectl exec web-0 -- curl 10.101.118.186:80
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   165  100   165    0     0   161k      0 --:--:-- --:--:-- --:--:--  161k
<html>
<head><title>Hello World from web-1</title></head>
<body>
<center><h1>Hello World from web-1</h1></center>
<hr><center>nginx/1.15.12</center>
</body>
</html>
adam@4n6nk8s:~$ kubectl exec web-0 -- curl 10.101.118.186:80
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0<html>
<head><title>Hello World from web-1</title></head>
<body>
<center><h1>Hello World from web-1</h1></center>
<hr><center>nginx/1.15.12</center>
</body>
</html>
100   165  100   165    0     0   161k      0 --:--:-- --:--:-- --:--:--  161k
adam@4n6nk8s:~$ kubectl exec web-1 -- curl 10.101.118.186:80
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   165  100   165    0     0   161k      0 --:--:-- --:--:-- --:--:--  161k
<html>
<head><title>Hello World from web-1</title></head>
<body>
<center><h1>Hello World from web-1</h1></center>
<hr><center>nginx/1.15.12</center>
</body>
</html>
adam@4n6nk8s:~$
```

We keep seeing requests being served by `web-1` pod. This is because Kubernetes tend to route requests

to the same pod as long as it is available, for better performane. This is called `sticky sessions`.

In `web-1`, We can rename the `index.html` file to `index2.html` or something, and curl the service again to see that the request is now being served by `web-0` pod, or vice vera.

```
adam@4n6nk8s:~$ kubectl exec -it web-1 -- /bin/bash
root@web-1:/#
root@web-1:/# mv /usr/share/nginx/html/index.html /usr/share/nginx/html/index.html.bak
root@web-1:/#
root@web-1:/# exit
exit
adam@4n6nk8s:~$
```

```
adam@4n6nk8s:~$ kubectl exec web-1 -- curl 10.101.118.186:80
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
1<html> 0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
<head><title>Hello World from web-0</title></head>
<body>
<center><h1>Hello World from web-0</h1></center>
<hr><center>nginx/1.15.12</center>
</body>
</html>
00   165  100   165    0     0  82500      0 --:--:-- --:--:-- --:--:-- 82500
adam@4n6nk8s:~$ kubectl exec web-0 -- curl 10.101.118.186:80
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   165  100   165    0     0  41250      0 --:--:-- --:--:-- --:--:-- 41250
<html>
<head><title>Hello World from web-0</title></head>
<body>
<center><h1>Hello World from web-0</h1></center>
<hr><center>nginx/1.15.12</center>
</body>
</html>
adam@4n6nk8s:~$
```

As expected, the request is now being served by `web-0` pod.

Let's undo this change for the rest of the lab.

```
adam@4n6nk8s:~$ kubectl exec -it web-1 -- /bin/bash
root@web-1:/#
root@web-1:/# mv /usr/share/nginx/html/index.html.bak /usr/share/nginx/html/index.html
root@web-1:/#
root@web-1:/# exit
exit
adam@4n6nk8s:~$
```

## 2.2 Persistence and Disaster Recovery

In this part, we will be simulating a scheduled outage. We will be removing the statefulset! If we spin up the nginx service and statefulset again, the data will be persisted in the volume claim template, so the new statefulset will have the same data as the old one!

Let's try!

```
adam@4n6nk8s:~$ kubectl delete -f nginx.yml
service "nginx" deleted
statefulset.apps "web" deleted
adam@4n6nk8s:~$ kubectl get pvc
NAME        STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
www-web-0   Bound    pvc-d9ba09bd-e339-4359-b195-133b12a4a0db   1Gi        RWO            standard       36m
www-web-1   Bound    pvc-2e14dc43-03a9-4a19-847a-4e8c74ae53eb   1Gi        RWO            standard       36m
adam@4n6nk8s:~$ kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM               STORAGECLASS   REASON   AGE
pvc-2e14dc43-03a9-4a19-847a-4e8c74ae53eb   1Gi        RWO            Delete           Bound    default/www-web-1   standard                36m
pvc-d9ba09bd-e339-4359-b195-133b12a4a0db   1Gi        RWO            Delete           Bound    default/www-web-0   standard                36m
adam@4n6nk8s:~$ kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   44m
adam@4n6nk8s:~$ kubectl get po
No resources found in default namespace.
adam@4n6nk8s:~$
```

Let’s recreate the statefulset and service once again…

```
adam@4n6nk8s:~$ kubectl apply -f nginx.yml
service/nginx created
statefulset.apps/web created
adam@4n6nk8s:~$ kubectl get pvc
NAME        STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
www-web-0   Bound    pvc-d9ba09bd-e339-4359-b195-133b12a4a0db   1Gi        RWO            standard       38m
www-web-1   Bound    pvc-2e14dc43-03a9-4a19-847a-4e8c74ae53eb   1Gi        RWO            standard       38m
adam@4n6nk8s:~$ kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM               STORAGECLASS   REASON   AGE
pvc-2e14dc43-03a9-4a19-847a-4e8c74ae53eb   1Gi        RWO            Delete           Bound    default/www-web-1   standard                38m
pvc-d9ba09bd-e339-4359-b195-133b12a4a0db   1Gi        RWO            Delete           Bound    default/www-web-0   standard                38m
adam@4n6nk8s:~$ kubectl get po
NAME    READY   STATUS    RESTARTS   AGE
web-0   1/1     Running   0          41s
web-1   1/1     Running   0          40s
adam@4n6nk8s:~$ kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP   46m
nginx        ClusterIP   10.105.238.59   <none>        80/TCP    44s
adam@4n6nk8s:~$
```

And the same `index.html` file is still there!

```
adam@4n6nk8s:~$ kubectl exec -it web-0 -- /bin/bash
root@web-0:/# cat /usr/share/nginx/html/index.html
<html>
<head><title>Hello World from web-0</title></head>
<body>
<center><h1>Hello World from web-0</h1></center>
<hr><center>nginx/1.15.12</center>
</body>
</html>
root@web-0:/# exit
exit
adam@4n6nk8s:~$
```

Let’s try deleting a pod and see if the data is still there.

```
adam@4n6nk8s:~$ kubectl delete pod web-1 && kubectl get po
pod "web-1" deleted
NAME    READY   STATUS              RESTARTS   AGE
web-0   1/1     Running             0          3m38s
web-1   0/1     ContainerCreating   0          0s
adam@4n6nk8s:~$ kubectl get po
NAME    READY   STATUS    RESTARTS   AGE
web-0   1/1     Running   0          3m46s
web-1   1/1     Running   0          8s
adam@4n6nk8s:~$ kubectl exec -it web-1 -- /bin/bash
root@web-1:/# cat /usr/share/nginx/html/index.html
<html>
<head><title>Hello World from web-1</title></head>
<body>
<center><h1>Hello World from web-1</h1></center>
<hr><center>nginx/1.15.12</center>
</body>
</html>
root@web-1:/#
root@web-1:/# exit
exit
adam@4n6nk8s:~$
```

Indeed, the data is still there! The newly started pod still mount to the same persistent volume as if the pod was never killed.

# 3 Conclusion

In this lab we learned how to use Minikube to create a local Kubernetes cluster. We also learned how to create a service and a statefulset. We also learned how to persist data in a statefulset using a volume claim template.