# Tasks

tree of homework

.  
├── homework-tonytech.md  
├── k8s  
│   ├── task1  
│   │   ├── cm.yaml  
│   │   ├── create\_cm\_dinamically.sh  
│   │   ├── main.conf  
│   │   ├── main.crt  
│   │   ├── main.key  
│   │   ├── pod.yaml  
│   │   ├── port.conf  
│   │   └── svc.yaml  
│   ├── task2  
│   │   ├── be-pod-statefulset.yaml  
│   │   ├── be-pvs.yaml  
│   │   ├── be-service.yaml  
│   │   ├── fe-pod-deployment.yaml  
│   │   └── fe-service.yaml  
│   └── Vagrantfile  
└── pictures  
 ├── pic-1.png  
 ├── pic-2.png  
 └── pic-3.png

Try to solve the following set of tasks:

* Using Vagrantfile with 3 nodes and one nfs server

# -\*- mode: ruby -\*-  
# vi: set ft=ruby :  
  
$common = <<SCRIPT  
echo '\* Add hosts ...'  
echo '127.0.0.1 localhost' > /etc/hosts  
echo '' >> /etc/hosts  
echo 'ff02::1 ip6-allnodes' >> /etc/hosts  
echo 'ff02::2 ip6-allrouters' >> /etc/hosts  
echo '' >> /etc/hosts  
echo '192.168.99.101 node1.k8s.lab node1' >> /etc/hosts  
echo '192.168.99.102 node2.k8s.lab node2' >> /etc/hosts  
echo '192.168.99.103 node3.k8s.lab node3' >> /etc/hosts  
echo '192.168.99.104 nfs-server.k8s.lab nfs-server' >> /etc/hosts  
  
SCRIPT  
  
$podnet = <<SCRIPT  
# Possible values are none, flannel, calico, and antrea  
PODNETWORK='flannel'  
  
if [ $PODNETWORK == 'flannel' ]; then  
 echo "\* Installing Pod Network plugin (Flannel) ..."  
 wget -q https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml -O /tmp/kube-flannel.yaml  
 sed -i '/--kube-subnet-mgr/ a CHANGEME' /tmp/kube-flannel.yaml  
 sed -i "s/CHANGEME/ - --iface=$(ip a | grep 192.168.99.101 | tr -s ' ' | cut -d ' ' -f 8)/" /tmp/kube-flannel.yaml   
 kubectl apply -f /tmp/kube-flannel.yaml  
elif [ $PODNETWORK == 'calico' ]; then   
 echo "\* Installing Pod Network plugin (Calico) ..."  
 kubectl create -f https://raw.githubusercontent.com/projectcalico/calico/v3.28.2/manifests/tigera-operator.yaml  
 wget -q https://raw.githubusercontent.com/projectcalico/calico/v3.28.2/manifests/custom-resources.yaml -O /tmp/custom-resources.yaml  
 sed -i 's/192.168.0.0/10.244.0.0/g' /tmp/custom-resources.yaml  
 kubectl create -f /tmp/custom-resources.yaml  
elif [ $PODNETWORK == 'antrea' ]; then   
 echo "\* Installing Pod Network plugin (Antrea) ..."  
 kubectl apply -f https://raw.githubusercontent.com/antrea-io/antrea/main/build/yamls/antrea.yml  
else  
 echo "\* WARNING: No Pod Network has been configured."  
fi  
  
SCRIPT  
  
$k8scp = <<SCRIPT   
  
echo "\* Initialize Kubernetes cluster ..."  
kubeadm init --kubernetes-version=$(kubeadm version --output=short) --apiserver-advertise-address=192.168.99.101 --pod-network-cidr 10.244.0.0/16 --token abcdef.0123456789abcdef  
  
echo "\* Save the hash to a file ..."  
openssl x509 -pubkey -in /etc/kubernetes/pki/ca.crt | openssl rsa -pubin -outform der 2>/dev/null | openssl dgst -sha256 -hex | sed 's/^.\* //' > /vagrant/hash.txt  
  
echo "\* Copy configuration for root ..."  
mkdir -p /root/.kube  
cp -i /etc/kubernetes/admin.conf /root/.kube/config  
chown -R root:root /root/.kube  
  
echo "\* Copy configuration for vagrant ..."  
mkdir -p /home/vagrant/.kube  
cp -i /etc/kubernetes/admin.conf /home/vagrant/.kube/config  
chown -R vagrant:vagrant /home/vagrant/.kube  
  
SCRIPT  
  
$k8swk = <<SCRIPT  
  
echo "\* Join the worker node ..."  
kubeadm join 192.168.99.101:6443 --token abcdef.0123456789abcdef --discovery-token-ca-cert-hash sha256:`cat /vagrant/hash.txt`  
  
SCRIPT  
  
  
Vagrant.configure("2") do |config|  
 config.vm.box = "shekeriev/kubernetes-on-debian"  
 config.vm.box\_version = "1.30.3"  
  
 config.vm.define "node1" do |node1|  
 node1.vm.hostname = "node1.k8s.lab"  
 node1.vm.network "private\_network", ip: "192.168.99.101"  
 node1.vm.synced\_folder "task1/", "/vagrant/task1"  
 node1.vm.synced\_folder "task2/", "/vagrant/task2"  
 node1.vm.provision "shell", inline: $common  
 node1.vm.provision "shell", inline: $k8scp  
 node1.vm.provision "shell", inline: $podnet  
 end  
  
 config.vm.define "node2" do |node2|  
 node2.vm.hostname = "node2.k8s.lab"  
 node2.vm.network "private\_network", ip: "192.168.99.102"  
 node2.vm.provision "shell", inline: $common  
 node2.vm.provision "shell", inline: $k8swk  
 end  
  
 config.vm.define "node3" do |node3|  
 node3.vm.hostname = "node3.k8s.lab"  
 node3.vm.network "private\_network", ip: "192.168.99.103"  
 node3.vm.provision "shell", inline: $common  
 node3.vm.provision "shell", inline: $k8swk  
 end  
  
 config.vm.define "nfs-server" do |nfs|  
 nfs.vm.hostname = "nfs-server.k8s.lab"  
 nfs.vm.network "private\_network", ip: "192.168.99.104"  
 nfs.vm.provision "shell", inline: $common  
 end  
  
end

### 1. Configuration maps and secrets

#### a. Create a ConfigMap resource hwcm that:

##### i. has two key-value pairs (k8sver and k8sos) initialized as literals that hold your Kubernetes version and the name of the OS where Kubernetes is running

* prepare script to create ConfigMap hwcm

#!/bin/bash  
  
# Get Kubernetes version  
K8S\_VERSION=$(kubectl version | grep "Server Version" | awk '{print $3}')  
  
# Get OS name  
K8S\_OS=$(cat /etc/os-release | grep ^ID= | awk -F'=' '{print $2}')  
  
# Create ConfigMap yaml file  
cat > cm.yaml << EOF  
apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: hwcm  
data:  
 k8sver: "${K8S\_VERSION}"  
 k8sos: "${K8S\_OS}"  
EOF

* Execute create\_cm\_dinamically.sh

$ ./create\_cm\_dinamically.sh  
  
$ cat cm.yaml  
apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: hwcm  
data:  
 k8sver: "v1.30.3"  
 k8sos: "debian"

* Apply cm.yaml

$ kubectl apply -f cm.yaml  
configmap/hwcm created  
  
$ kubectl describe cm hwcm  
'Name: hwcm  
Namespace: default  
Labels: <none>  
Annotations: <none>  
  
Data  
====  
k8sos:  
----  
debian  
k8sver:  
----  
v1.30.3  
  
BinaryData  
====  
  
Events: <none>

##### ii. has two more key-value pairs (main.conf and port.conf) initialized from files. The first one (main.conf) should contain:

# main.conf   
name=homework   
path=/tmp   
certs=/secret

And the second one (port.conf):

8080

* Delete ConfigMap because I can’t figure out how to attach files inside manifest file

kubectl delete -f cm.yaml  
configmap "hwcm" deleted

* create ConfigMap interactively

$ export K8S\_VERSION=$(kubectl version | grep "Server Version" | awk '{print $3}')  
$ export K8S\_OS=$(cat /etc/os-release | grep ^ID= | awk -F'=' '{print $2}')  
$ kubectl create configmap hwcm --from-literal=k8sver=$K8S\_VERSION --from-literal=k8sos=$K8S\_OS --from-file=main.conf --from-file=port.conf  
  
$ kubectl describe cm hwcm  
Name: hwcm  
Namespace: default  
Labels: <none>  
Annotations: <none>  
  
Data  
====  
port.conf:  
----  
8080  
k8sos:  
----  
debian  
k8sver:  
----  
v1.30.3  
main.conf:  
----  
# main.conf\r  
name=homework\r  
path=/tmp\r  
certs=/secret  
  
BinaryData  
====  
  
Events: <none>

#### b. Create a Secret resource hwsec that:

* Generate main.key file

$ openssl genrsa -out main.key 4096  
  
$ ls -al main.key  
-rwxrwxrwx 1 vagrant vagrant 3272 Nov 3 11:39 main.key

* Create main.crt file

$ openssl req -new -x509 -key main.key -out main.crt -days 365 -subj /CN=www.hw.lab  
  
$ ls -al main.crt  
-rwxrwxrwx 1 vagrant vagrant 1809 Nov 3 11:39 main.crt

##### i. Has two data entries – main.key and main.crt created from files

##### ii. The content for the above two generate by using the openssl utility. For example:

openssl genrsa -out main.key 4096   
openssl req -new -x509 -key main.key -out main.crt -days 365 -subj /CN=www.hw.lab

* Generate main.key file

$ openssl genrsa -out main.key 4096  
  
$ ls -al main.key  
-rwxrwxrwx 1 vagrant vagrant 3272 Nov 3 11:39 main.key

* Create main.crt file

$ openssl req -new -x509 -key main.key -out main.crt -days 365 -subj /CN=www.hw.lab  
  
$ ls -al main.crt  
-rwxrwxrwx 1 vagrant vagrant 1809 Nov 3 11:39 main.crt

* Create Secret object

$ kubectl create secret generic hwsec --from-file=main.key --from-file=main.crt  
  
$ kubectl describe secret hwsec  
Name: hwsec  
Namespace: default  
Labels: <none>  
Annotations: <none>  
  
Type: Opaque  
  
Data  
====  
main.key: 3272 bytes  
main.crt: 1809 bytes

#### c. Mount the above resources to a pod created from the shekeriev/k8s-environ image (used during the practice) by

##### ii. k8sver and k8sos should be mounted as environment variables with prefix HW\_

##### ii. main.conf should be mounted as a volume to the /config folder inside the container

##### iii. port.conf should be mounted as an environment variable HW\_PORT

##### iv. main.key and main.crt should be mounted as a volume to the /secret folder inside the container

* Create pod.yaml file

$ cat pod.yaml  
apiVersion: v1  
kind: Pod  
metadata:  
 name: pod-task1  
 labels:  
 app: environ-task1  
spec:  
 containers:  
 - image: shekeriev/k8s-environ  
 name: task1  
 envFrom:  
 - configMapRef:  
 name: hwcm  
 prefix: HW\_  
 env:  
 - name: HW\_PORT  
 valueFrom:  
 configMapKeyRef:  
 name: hwcm  
 key: port.conf  
 volumeMounts:  
 - name: config-volume  
 mountPath: /config  
 - name: secret-volume  
 mountPath: /secret  
  
 volumes:  
 - name: config-volume  
 configMap:  
 name: hwcm  
 items:  
 - key: main.conf  
 path: main.conf  
 - name: secret-volume  
 secret:  
 secretName: hwsec  
  
$ kubectl apply -f pod.yaml  
pod/pod-task1 created

* Create svc.yaml file

$ cat svc.yaml  
apiVersion: v1  
kind: Service  
metadata:  
 name: svc-environ  
spec:  
 type: NodePort  
 ports:  
 - port: 80  
 nodePort: 30001  
 protocol: TCP  
 selector:  
 app: environ-task1  
  
$ kubectl apply -f svc.yaml  
service/svc-environ created

* Inspect pods and services

$ kubectl get pod,svc  
NAME READY STATUS RESTARTS AGE  
pod/pod-task1 1/1 Running 0 7m29s  
  
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  
service/kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 87m  
service/svc-environ NodePort 10.96.214.219 <none> 80:30001/TCP 12m

* Picture

A screenshot of a computer

Description automatically generated

* Entire output in browser

Environment Landscape  
APACHE\_CONFDIR => /etc/apache2  
APACHE\_ENVVARS => /etc/apache2/envvars  
APACHE\_LOCK\_DIR => /var/lock/apache2  
APACHE\_LOG\_DIR => /var/log/apache2  
APACHE\_PID\_FILE => /var/run/apache2/apache2.pid  
APACHE\_RUN\_DIR => /var/run/apache2  
APACHE\_RUN\_GROUP => www-data  
APACHE\_RUN\_USER => www-data  
GPG\_KEYS => 1729F83938DA44E27BA0F4D3DBDB397470D12172 BFDDD28642824F8118EF77909B67A5C12229118F  
HOSTNAME => pod-task1  
HW\_PORT => 8080  
HW\_k8sos => debian  
HW\_k8sver => v1.30.3  
KUBERNETES\_PORT => tcp://10.96.0.1:443  
KUBERNETES\_PORT\_443\_TCP => tcp://10.96.0.1:443  
KUBERNETES\_PORT\_443\_TCP\_ADDR => 10.96.0.1  
KUBERNETES\_PORT\_443\_TCP\_PORT => 443  
KUBERNETES\_PORT\_443\_TCP\_PROTO => tcp  
KUBERNETES\_SERVICE\_HOST => 10.96.0.1  
KUBERNETES\_SERVICE\_PORT => 443  
KUBERNETES\_SERVICE\_PORT\_HTTPS => 443  
LANG => C  
PATH => /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin  
PHPIZE\_DEPS => autoconf dpkg-dev file g++ gcc libc-dev make pkg-config re2c  
PHP\_ASC\_URL => https://www.php.net/distributions/php-8.0.12.tar.xz.asc  
PHP\_CFLAGS => -fstack-protector-strong -fpic -fpie -O2 -D\_LARGEFILE\_SOURCE -D\_FILE\_OFFSET\_BITS=64  
PHP\_CPPFLAGS => -fstack-protector-strong -fpic -fpie -O2 -D\_LARGEFILE\_SOURCE -D\_FILE\_OFFSET\_BITS=64  
PHP\_INI\_DIR => /usr/local/etc/php  
PHP\_LDFLAGS => -Wl,-O1 -pie  
PHP\_SHA256 => a501017b3b0fd3023223ea25d98e87369b782f8a82310c4033d7ea6a989fea0a  
PHP\_URL => https://www.php.net/distributions/php-8.0.12.tar.xz  
PHP\_VERSION => 8.0.12  
PWD => /var/www/html  
SHLVL => 0  
SVC\_ENVIRON\_PORT => tcp://10.96.214.219:80  
SVC\_ENVIRON\_PORT\_80\_TCP => tcp://10.96.214.219:80  
SVC\_ENVIRON\_PORT\_80\_TCP\_ADDR => 10.96.214.219  
SVC\_ENVIRON\_PORT\_80\_TCP\_PORT => 80  
SVC\_ENVIRON\_PORT\_80\_TCP\_PROTO => tcp  
SVC\_ENVIRON\_SERVICE\_HOST => 10.96.214.219  
SVC\_ENVIRON\_SERVICE\_PORT => 80  
  
  
Folder (/config) Dump  
/config/main.conf  
# main.conf  
name=homework  
path=/tmp  
certs=/secret  
  
  
Folder (/secret) Dump  
/secret/main.crt  
-----BEGIN CERTIFICATE-----  
MIIFCzCCAvOgAwIBAgIUSRkMSAEM5cR1cVtvtzNW8FQ5LfwwDQYJKoZIhvcNAQEL  
BQAwFTETMBEGA1UEAwwKd3d3Lmh3LmxhYjAeFw0yNDExMDMwOTM5MzhaFw0yNTEx  
MDMwOTM5MzhaMBUxEzARBgNVBAMMCnd3dy5ody5sYWIwggIiMA0GCSqGSIb3DQEB  
AQUAA4ICDwAwggIKAoICAQChQMClbhqVu0Ac/LUBF/Z/+lEhP1eB1QK0IY4UXeGE  
/b+LZ/F4GVzX2aefbtqLVIXOpkKdVQut8QURBPKXRCgjzHwD736ZHbhtV6MjGw6J  
2V/Fc/PX0lcMhBgY7dyI5DWOWWHBm43xEQbJU64jRN3MK3GzfC2PF467cGL+PVdj  
4hOYSg/WreStF8U0J+wzyC8L6LhfgKh94Rbzp1+38LkIghtDaIzui4ETUP3VXKdV  
YBISvcbxW0K45hdmMjGVxiNDOI2PtF9+paNguLICnPutuq7nrnkMom77rZ2ZNQJs  
YD41s0dRNXZyisSt6sC27P3g0wDMX6uYIvQ9faA07ABma9nLp2Hh+GqBNv1I9GvL  
M3om2SUKdJzCpr1Jjs5LPHsaFfPathhp+qNfLevvDJSc3gnr0KjKO9KSEh/iXnDa  
ViudjwLL3huYexb+pSjvp7m7U0axhx9ZBIPZKHyI7n/MMjLL4ARmzzVTTmL9LqEa  
GLDkM+eWU/ampC/8mRormQa9Q3D77zoed/xRfO38GOimy/KXlfhUog7bY4BPHEAJ  
/GXGxa7dmueMpiGkf/3ciwmmmie3gv5IEyf7zFAHojbBTeehmCHqBKGv0GbCnJVi  
mWugJIOCFIp8cxbEgvvt0gkp31apqNsIfBcUXXi/Fm94/RXwlkrWB9rfmJzw7gut  
rQIDAQABo1MwUTAdBgNVHQ4EFgQU7Uo0XbVBBEMJG30cr+yIqYCgtYYwHwYDVR0j  
BBgwFoAU7Uo0XbVBBEMJG30cr+yIqYCgtYYwDwYDVR0TAQH/BAUwAwEB/zANBgkq  
hkiG9w0BAQsFAAOCAgEATHjk4biKX6/wBPyeyEnaybWyKOh7dsinSuUK2xlcLtGX  
LTuL5moWb01FGxCoDpgYxGtai/ZCOVNa+yJy7ac++56Xv/24PgWpJAmw/tdWI727  
usREnhUOrWeRFH9BAOmAttxS7zN6JCGlAdjeBwuxCz6RHnnZGtB1WIM2vHUG6Se2  
q6BrpfUaAGD/vJ9ejsqntF34phVRQ8id7xJl1jTlGtZ+QwjAmuIyFaf4X+iWWrjR  
vw4eoO+BNqWbu+UWNIkqefOv8lu//0/+1OPGnDowUYVeDH8gUnnipfYeym/i9sEP  
jhrP0EQLvFwffXXCP5quFMUr0Wv4eeMGJevf9eXIu92vLNTki4+++9Qv9hHLYxz9  
r+A34GLDC/bSuTBkHnGK1wdp/k6KlNl+pO5qmnjzGSBTE68CbesdC6O900nmuokx  
P5mfq0mb58IpAWdRTK7VCf3jC6GeHk2K40BhHR7XvWi0LwZMIwkGCEiq6ja0eYhJ  
BRL9+2fH/zuNeZmQrqUnv7ZhCbIGtPNZb4/tCQm6/K6BJv266Tp6weTcJ1Yza1CW  
Ims0JlWWOZnzac+nMvfvak5F8ktqDk5nd3R9HboqjfGxGnvSnmEiRYnHm91H6BOl  
+FdqXxyxMdsZ4IZF5FnrApTYc7cDDZHrWfx8V+rS59nWOARTwYo3x1/4i0Y8u4o=  
-----END CERTIFICATE-----  
/secret/main.key  
-----BEGIN PRIVATE KEY-----  
MIIJQwIBADANBgkqhkiG9w0BAQEFAASCCS0wggkpAgEAAoICAQChQMClbhqVu0Ac  
/LUBF/Z/+lEhP1eB1QK0IY4UXeGE/b+LZ/F4GVzX2aefbtqLVIXOpkKdVQut8QUR  
BPKXRCgjzHwD736ZHbhtV6MjGw6J2V/Fc/PX0lcMhBgY7dyI5DWOWWHBm43xEQbJ  
U64jRN3MK3GzfC2PF467cGL+PVdj4hOYSg/WreStF8U0J+wzyC8L6LhfgKh94Rbz  
p1+38LkIghtDaIzui4ETUP3VXKdVYBISvcbxW0K45hdmMjGVxiNDOI2PtF9+paNg  
uLICnPutuq7nrnkMom77rZ2ZNQJsYD41s0dRNXZyisSt6sC27P3g0wDMX6uYIvQ9  
faA07ABma9nLp2Hh+GqBNv1I9GvLM3om2SUKdJzCpr1Jjs5LPHsaFfPathhp+qNf  
LevvDJSc3gnr0KjKO9KSEh/iXnDaViudjwLL3huYexb+pSjvp7m7U0axhx9ZBIPZ  
KHyI7n/MMjLL4ARmzzVTTmL9LqEaGLDkM+eWU/ampC/8mRormQa9Q3D77zoed/xR  
fO38GOimy/KXlfhUog7bY4BPHEAJ/GXGxa7dmueMpiGkf/3ciwmmmie3gv5IEyf7  
zFAHojbBTeehmCHqBKGv0GbCnJVimWugJIOCFIp8cxbEgvvt0gkp31apqNsIfBcU  
XXi/Fm94/RXwlkrWB9rfmJzw7gutrQIDAQABAoICABqqiGEjGNrfhR32DWHBqnYl  
+UIoBSrXhLOwo9wK3Wc+hi1B0m0DtraMTBvBu3rUjq8qj/QQMxb/91BF8u8gaonn  
ADAvbA5am5vbZQNCb3BM5flTcETlyBbuBkUjGrf5k0C318JqXLn7kJV9FNxIwGwJ  
2CrwE2GpKeWl030LPFcsdQSXZNCh36HfPzGdk4JJrbwIr2q9rj9o2ICREnNq59J4  
H5WRZdLx93WwDFX7FUH5WevG/e3+Y0WL6V0xWkFUK5MreD+GOs+JCUBOssNbXgRw  
DYoLMIHuH6/XmWsTWNP+w3Lic54CmxScjaNcIaPug5ppEmX5wTjuuNitXG+8AmGq  
UP/Jit5QMayLM1kyCIlZlU42xqxCGGfOdlzjXJjLRuCcgI7OahFy7ehoD/VB1+R1  
frWstSIRrJeGJgGQ278PXW4I6D3z3RFiDKDLQldiGPCE9uClm/YF5PlE2C6/6ryU  
iX5DqpnH5DaaiZYqyZ6pcGMUJYETz98XSCV9htgf3naYOEeRvhnsmbi4JDerg65n  
d87U7i3O2RieHENviHa5TE+nTzivECn0HHo2WRX9w2AN0TDNdDZyJ03JaRJ4dDuI  
KRCqFuhsoWeSSZsygJQx2673RtnFZQPSl33e4NzwRL3BgrgLlmsgCNYLIU/TXfYU  
3MbVKvxFi7fqI5+WdqlxAoIBAQDW02Ggi1mQWq7plnbaxpAYV6hiW2eUC08QLGkt  
pl7gVOLMxcbEJcvFobWgXQknVIMzsEmxlbvtz4ivOfEMDxWR3aHHaieylPgvwm8l  
t7I/PSeq3vhJ5ZUrLrso0KWOq7eSpaVzJwegRcgBH4nG3JaEGJzBv/IE0McfAz/2  
6BEM/bD1xh1u/RioXAYgpv8dfrjklJlYrdePGyOqNxSK8ZDbTkOTFmsHG87SnNRt  
fD3m2hAZR4KP35nvYdWl6vhji76M3jMHqFhgxbMW/t2fKA3dI3/2sA/27TBiZ9d8  
oE9CpUJtKgNiCZVkfCCCvlT9b5kIpIKgiQSIBrNWZaph5Od9AoIBAQDAKMZKNrEm  
x3hhL9hPNnNTz1u4puaKH2w7uI7AWXiJZDrhDdPoqubLfQQKVAhSdY28iNAAYiCD  
6VgQClsBh/vQMTQaK1Hvun4KdGBCGdgdwmpz+X4vxtnZfHFJBan8AwEHaWp/YwuK  
g9f7XtKnl1Dt70sEkAe5G5CWKJ1LyZ+zQxeK8BD8FHkbYbDYlEHlrhEAUuPX03xM  
v9KloVKeZSAr+XYt9npCYJ0EEVnVu+EORYZCZGHKfz1qu1+P31TO9XcFE0MuNA0z  
i3tJpOV9RRyBwF4LMJzAf8J5tQ/2rICiivF/Kqh6XD+O0nRZKetIiUMyhFJjt8HW  
eKvjFLTiupXxAoIBAH88c3ZDEqkNDMwM8hhhwraoNHMM8CwfBpTtuN3wzADSRgJC  
oZdLS4WiYZC5idJhm9PhdcRYixYu9SXTi5E43e2m90rE9P4wfU7MOkh8pHYQIo5x  
wV0DPXZ2FyP1cBkwW2nf8j5hv4KN0gfYRiIiRIO4kB+xMAbqIQWdRyVWDcND/quS  
y5wRC6mr13U/c5pkk93OpYSAckuwJ0+JkAE7UjCSCjPz8zAiqDzV5ntyOregMCjI  
0euk4+eEaqX6pWNQQWvtrwu6CZ5HeesEyP6bGpr/i6rwXIRDt4SRPG/0Av3WI6HE  
YHyQcIszeY32bSAuGNWlmji67pQiY9WoIWIZ0YECggEBAKfetyO8VdSf681Tg+/n  
jKbJ/sH4pGxD+UTO/X6SaxfzxPA5KUgYjRl19KKlOMP4yELNbpD3KJq0rmcM9OYI  
ivcdFPxLngAE9vlhI3p7o26UGnWInzFT0ZsK+NinJxHu+eYktJHmz/1JItUJyJP6  
QAwswWeR//05VH2pq4i68lYjpZyODvh5VjeSh3EdS9g/0cBrqXox2mQeLALgRRH1  
HD7uZszOh44XYvRbyBRNVtYl7J1xnnDN6bSlTl2c6DUJAdogMs9ZT7MZBjKW6Jkg  
WEAN8tpptHcC5b13bSO9HX9iTIkgliCXGDtIJbaDRXWP3HF0YZmVvbQEpXzy6Pct  
s7ECggEBAKLReV+EeqwMenDsuvz1r09qR09KBy3WhhSF9T9WSfvU6JFmZ9Cxi2/G  
jlMfbjIy/t1LmbjGABztlCbsNid/17Ec5ryueP5n+3+gz2CvcL+jVu4yK2leE+ze  
SI5DFEQbBrHGYunkLc71KrK3dBNTomUPjUF4odxmlqEa/j7ZEoCLuKTVNHfzvF4H  
VRTsfvKmzPqJXmp+DpdUTCyBOF7P2FXZfMWnQ7xMlc8/0ZPjzXYd555uloOSU3FQ  
ciqcAwkiA8AGYjHTgoOG20TpVmcdCvz7DBLxvM7KMr54jFjd1lHZTH+9wqOGRNFM  
x3d2eXK9BVYShTsqwOF8SP7+hWRgp2M=  
-----END PRIVATE KEY-----  
  
  
Working on pod-task1

### 2. Create and run a set of manifest files to spin the following application:

A diagram of a service

Description automatically generated

Please note that:

* Service FE should be of type NodePort

---  
apiVersion: v1  
kind: Service  
metadata:  
 name: fe-service  
spec:  
 type: NodePort  
 ports:  
 - port: 5000  
 targetPort: 5000  
 nodePort: 30001  
 protocol: TCP  
 selector:  
 app: fe

* Pod FE should use shekeriev/k8s-facts-fe image and should be initialized with two environment variables – FACTS\_SERVER equal to the name of the Service BE and FACTS\_PORT equal to the port of Service BE

---  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: fe-deployment  
spec:  
 replicas: 3  
 selector:  
 matchLabels:  
 app: fe  
 template:  
 metadata:  
 labels:  
 app: fe  
 spec:  
 containers:  
 - name: fe-container  
 image: shekeriev/k8s-facts-fe  
 env:  
 - name: FACTS\_SERVER  
 value: "be-service"  
 - name: FACTS\_PORT  
 value: "5000"  
 ports:  
 - containerPort: 5000

* Pod FE listens on port 5000/tcp
* Service BE should be of type ClusterIP (please note, that this is not the headless service but the “public” one)

---  
apiVersion: v1  
kind: Service  
metadata:  
 name: be-service  
spec:  
 type: ClusterIP  
 ports:  
 - port: 5000  
 targetPort: 5000  
 protocol: TCP  
 selector:  
 app: be

* Pod BE should use shekeriev/k8s-facts image and expects a volume to be mounted at /data folder

---  
apiVersion: apps/v1  
kind: StatefulSet  
metadata:  
 name: be-statefulset  
spec:  
 serviceName: be-service  
 selector:  
 matchLabels:  
 app: be  
 replicas: 3  
 # POD template   
 template:  
 metadata:  
 labels:  
 app: be  
 spec:  
 containers:  
 - name: be-container  
 image: shekeriev/k8s-facts  
 ports:  
 - containerPort: 5000  
 volumeMounts:  
 - name: be-volume  
 mountPath: /data  
 readOnly: false  
 # VolumeClaim template   
 volumeClaimTemplates:  
 - metadata:  
 name: be-volume  
 spec:  
 accessModes: [ "ReadWriteOnce" ]  
 resources:  
 requests:  
 storage: 2Gi

* Pod BE listens on port 5000/tcp
* For the PVs and PVCs use NFS and storage capacity of 2Gi

---  
apiVersion: v1  
kind: PersistentVolume  
metadata:  
 name: pv1  
spec:  
 capacity:  
 storage: 2Gi  
 volumeMode: Filesystem   
 accessModes:  
 - ReadWriteOnce  
 persistentVolumeReclaimPolicy: Recycle  
 mountOptions:  
 - nfsvers=4.1  
 nfs:  
 path: /data/nfs/pv1  
 server: nfs-server  
---  
apiVersion: v1  
kind: PersistentVolume  
metadata:  
 name: pv2  
spec:  
 capacity:  
 storage: 2Gi  
 volumeMode: Filesystem   
 accessModes:  
 - ReadWriteOnce  
 persistentVolumeReclaimPolicy: Recycle  
 mountOptions:  
 - nfsvers=4.1   
 nfs:  
 path: /data/nfs/pv2  
 server: nfs-server  
---  
apiVersion: v1  
kind: PersistentVolume  
metadata:  
 name: pv3  
spec:  
 capacity:  
 storage: 2Gi  
 volumeMode: Filesystem   
 accessModes:  
 - ReadWriteOnce  
 persistentVolumeReclaimPolicy: Recycle  
 mountOptions:  
 - nfsvers=4.1  
 nfs:  
 path: /data/nfs/pv3  
 server: nfs-server

* Both the Deployment and the StatefulSet should spin three replicas

**Task2 step by step**

* Create folders on nfs-server

$ sudo mkdir -p /data/nfs/pv{1..3}  
  
$ ls -al /data/nfs/  
total 20  
drwxr-xr-x 5 root root 4096 Nov 3 15:05 .  
drwxr-xr-x 3 root root 4096 Nov 3 15:05 ..  
drwxr-xr-x 2 root root 4096 Nov 3 15:05 pv1  
drwxr-xr-x 2 root root 4096 Nov 3 15:05 pv2  
drwxr-xr-x 2 root root 4096 Nov 3 15:05 pv3

* Made folders writable by everyone

$ sudo chmod -R 777 /data/nfs/  
  
$ ls -al /data/nfs/  
total 20  
drwxrwxrwx 5 root root 4096 Nov 3 15:05 .  
drwxr-xr-x 3 root root 4096 Nov 3 15:05 ..  
drwxrwxrwx 2 root root 4096 Nov 3 15:05 pv1  
drwxrwxrwx 2 root root 4096 Nov 3 15:05 pv2  
drwxrwxrwx 2 root root 4096 Nov 3 15:05 pv3

* Add folders in /etc/exports

$ sudo exportfs  
/data/nfs/pv1 <world>  
/data/nfs/pv2 <world>  
/data/nfs/pv3 <world>

* Apply fe-service.yaml

$ kubectl apply -f fe-service.yaml  
service/fe-service created

* Apply fe-pod-deployment.yaml

$ kubectl apply -f fe-pod-deployment.yaml  
deployment.apps/fe-deployment created

* Apply be-service.yaml

$ kubectl apply -f be-service.yaml  
service/be-service created

* Apply be-pvs.yaml

$ kubectl apply -f be-pvs.yaml  
persistentvolume/pv1 created  
persistentvolume/pv2 created  
persistentvolume/pv3 created

* Apply be-pod-statefulset.yaml

$ kubectl apply -f be-pod-statefulset.yaml  
statefulset.apps/be-statefulset created

* Verify all objects

$ kubectl get pod,svc,pv,pvc  
NAME READY STATUS RESTARTS AGE  
pod/be-statefulset-0 1/1 Running 0 9m15s  
pod/be-statefulset-1 1/1 Running 0 91s  
pod/be-statefulset-2 1/1 Running 0 84s  
pod/fe-deployment-59bbdb4cd-fq6n9 1/1 Running 0 17m  
pod/fe-deployment-59bbdb4cd-j9fk6 1/1 Running 0 17m  
pod/fe-deployment-59bbdb4cd-rjcvj 1/1 Running 0 17m  
  
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  
service/be-service ClusterIP 10.96.177.203 <none> 5000/TCP 15m  
service/fe-service NodePort 10.98.100.253 <none> 5000:30001/TCP 18m  
service/kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 4h37m  
  
NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS VOLUMEATTRIBUTESCLASS REASON AGE  
persistentvolume/pv1 2Gi RWO Retain Bound default/be-volume-be-statefulset-1 <unset> 15m  
persistentvolume/pv2 2Gi RWO Retain Bound default/be-volume-be-statefulset-2 <unset> 15m  
persistentvolume/pv3 2Gi RWO Retain Bound default/be-volume-be-statefulset-0 <unset> 15m  
  
NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS VOLUMEATTRIBUTESCLASS AGE  
persistentvolumeclaim/be-volume-be-statefulset-0 Bound pv3 2Gi RWO <unset> 14m  
persistentvolumeclaim/be-volume-be-statefulset-1 Bound pv1 2Gi RWO <unset> 91s  
persistentvolumeclaim/be-volume-be-statefulset-2 Bound pv2 2Gi RWO <unset> 84s

* Picture

A screenshot of a computer

Description automatically generated

### NOTE

The backend part is not working, I can’t find why.

* Take logs from be-statefulset-0 one of the BackEnd pods. There is a DNS issue.

$ kubectl logs be-statefulset-0  
 \* Environment: production  
 WARNING: This is a development server. Do not use it in a production deployment.  
 Use a production WSGI server instead.  
 \* Debug mode: off  
 \* Running on all addresses.  
 WARNING: This is a development server. Do not use it in a production deployment.  
 \* Running on http://10.244.1.9:5000/ (Press CTRL+C to quit)  
[2024-11-03 14:15:07,995] ERROR in app: Exception on / [GET]  
Traceback (most recent call last):  
 File "/usr/local/lib/python3.9/site-packages/flask/app.py", line 2070, in wsgi\_app  
 response = self.full\_dispatch\_request()  
 File "/usr/local/lib/python3.9/site-packages/flask/app.py", line 1515, in full\_dispatch\_request  
 rv = self.handle\_user\_exception(e)  
 File "/usr/local/lib/python3.9/site-packages/flask/app.py", line 1513, in full\_dispatch\_request  
 rv = self.dispatch\_request()  
 File "/usr/local/lib/python3.9/site-packages/flask/app.py", line 1499, in dispatch\_request  
 return self.ensure\_sync(self.view\_functions[rule.endpoint])(\*\*req.view\_args)  
 File "/app/app.py", line 30, in main  
 records = dns.resolver.resolve('facts.default.svc.cluster.local', 'SRV')  
 File "/usr/local/lib/python3.9/site-packages/dns/resolver.py", line 1305, in resolve  
 return get\_default\_resolver().resolve(qname, rdtype, rdclass, tcp, source,  
 File "/usr/local/lib/python3.9/site-packages/dns/resolver.py", line 1163, in resolve  
 (request, answer) = resolution.next\_request()  
 File "/usr/local/lib/python3.9/site-packages/dns/resolver.py", line 612, in next\_request  
 raise NXDOMAIN(qnames=self.qnames\_to\_try,  
dns.resolver.NXDOMAIN: The DNS query name does not exist: facts.default.svc.cluster.local.  
10.244.2.4 - - [03/Nov/2024 14:15:07] "GET / HTTP/1.1" 500 -  
[2024-11-03 14:15:08,694] ERROR in app: Exception on / [GET]  
Traceback (most recent call last):  
 File "/usr/local/lib/python3.9/site-packages/flask/app.py", line 2070, in wsgi\_app  
 response = self.full\_dispatch\_request()  
 File "/usr/local/lib/python3.9/site-packages/flask/app.py", line 1515, in full\_dispatch\_request  
 rv = self.handle\_user\_exception(e)  
 File "/usr/local/lib/python3.9/site-packages/flask/app.py", line 1513, in full\_dispatch\_request  
 rv = self.dispatch\_request()  
 File "/usr/local/lib/python3.9/site-packages/flask/app.py", line 1499, in dispatch\_request  
 return self.ensure\_sync(self.view\_functions[rule.endpoint])(\*\*req.view\_args)  
 File "/app/app.py", line 30, in main  
 records = dns.resolver.resolve('facts.default.svc.cluster.local', 'SRV')  
 File "/usr/local/lib/python3.9/site-packages/dns/resolver.py", line 1305, in resolve  
 return get\_default\_resolver().resolve(qname, rdtype, rdclass, tcp, source,  
 File "/usr/local/lib/python3.9/site-packages/dns/resolver.py", line 1163, in resolve  
 (request, answer) = resolution.next\_request()  
 File "/usr/local/lib/python3.9/site-packages/dns/resolver.py", line 612, in next\_request  
 raise NXDOMAIN(qnames=self.qnames\_to\_try,  
dns.resolver.NXDOMAIN: The DNS query name does not exist: facts.default.svc.cluster.local.  
10.244.2.3 - - [03/Nov/2024 14:15:08] "GET / HTTP/1.1" 500 -

* The /etc/hosts/ on all VMs

$ cat /etc/hosts  
127.0.0.1 localhost  
  
ff02::1 ip6-allnodes  
ff02::2 ip6-allrouters  
  
192.168.99.101 node1.k8s.lab node1  
192.168.99.102 node2.k8s.lab node2  
192.168.99.103 node3.k8s.lab node3  
192.168.99.104 nfs-server.k8s.lab nfs-server

* All PVs and PVCs are bounded

$ kubectl get pv,pvc  
NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS VOLUMEATTRIBUTESCLASS REASON AGE  
persistentvolume/pv1 2Gi RWO Retain Bound default/be-volume-be-statefulset-1 <unset> 63m  
persistentvolume/pv2 2Gi RWO Retain Bound default/be-volume-be-statefulset-2 <unset> 63m  
persistentvolume/pv3 2Gi RWO Retain Bound default/be-volume-be-statefulset-0 <unset> 63m  
  
NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS VOLUMEATTRIBUTESCLASS AGE  
persistentvolumeclaim/be-volume-be-statefulset-0 Bound pv3 2Gi RWO <unset> 63m  
persistentvolumeclaim/be-volume-be-statefulset-1 Bound pv1 2Gi RWO <unset> 50m  
persistentvolumeclaim/be-volume-be-statefulset-2 Bound pv2 2Gi RWO <unset> 50m

* The source in PV persistentvolume/pv1 looks fine!

$ kubectl describe persistentvolume/pv1  
Name: pv1  
Labels: <none>  
Annotations: pv.kubernetes.io/bound-by-controller: yes  
Finalizers: [kubernetes.io/pv-protection]  
StorageClass:  
Status: Bound  
Claim: default/be-volume-be-statefulset-1  
Reclaim Policy: Retain  
Access Modes: RWO  
VolumeMode: Filesystem  
Capacity: 2Gi  
Node Affinity: <none>  
Message:  
Source:  
 Type: NFS (an NFS mount that lasts the lifetime of a pod)  
 Server: nfs-server  
 Path: /data/nfs/pv1  
 ReadOnly: false  
Events: <none>

* Also PVC persistentvolumeclaim/be-volume-be-statefulset-0 is used from POD be-statefulset-0 as expected.

$ kubectl describe persistentvolumeclaim/be-volume-be-statefulset-0  
Name: be-volume-be-statefulset-0  
Namespace: default  
StorageClass:  
Status: Bound  
Volume: pv3  
Labels: app=be  
Annotations: pv.kubernetes.io/bind-completed: yes  
 pv.kubernetes.io/bound-by-controller: yes  
Finalizers: [kubernetes.io/pvc-protection]  
Capacity: 2Gi  
Access Modes: RWO  
VolumeMode: Filesystem  
Used By: be-statefulset-0  
Events: <none>

I tried quite a few other configurations but couldn’t get it to work. Finally, destroying the cluster and nfs-server :(

vagrant destroy -f  
==> nfs-server: Forcing shutdown of VM...  
==> nfs-server: Destroying VM and associated drives...  
==> node3: Forcing shutdown of VM...  
==> node3: Destroying VM and associated drives...  
==> node2: Forcing shutdown of VM...  
==> node2: Destroying VM and associated drives...  
==> node1: Forcing shutdown of VM...  
==> node1: Destroying VM and associated drives...