4/26/2021

OFTEIN++ UM Documentation

Table of Contents

[1 Outline 2](#_Toc70846417)

[2 Cluster Information 2](#_Toc70846418)

[2.1 Cluster Architecture 2](#_Toc70846419)

[2.2 IP Addressing 2](#_Toc70846420)

[2.3 Master nodes 3](#_Toc70846421)

[2.4 High-Availability (HA) Cluster 3](#_Toc70846422)

[2.5 Cluster Storage 3](#_Toc70846423)

[2.6 Networking 3](#_Toc70846424)

[3 Cluster setup 4](#_Toc70846425)

[3.1 System requirements 4](#_Toc70846426)

[3.2 Creating new cluster or Joining new node 4](#_Toc70846427)

[4 Cluster Services 5](#_Toc70846428)

[4.1 Nginx Ingress Controller 5](#_Toc70846429)

[4.2 NFS StorageClass 6](#_Toc70846430)

[4.3 Prometheus Grafana Monitoring 7](#_Toc70846431)

[5 Cluster maintenance 8](#_Toc70846432)

[5.1 Kubectl 8](#_Toc70846433)

[5.2 UM Oftein DNS Server 8](#_Toc70846434)

[5.3 Synology Data Storage 9](#_Toc70846435)

[5.4 Helm Chart 10](#_Toc70846436)

[5.5 Renewing/Upload HTTP SSL certificates 11](#_Toc70846437)

[5.6 Port proxy with socat; Nginx Ingress Controller 11](#_Toc70846438)

[6 Container Deployment Example 12](#_Toc70846439)

[6.1 Deploy container from yaml file 12](#_Toc70846440)

[6.2 Deploy application through helm chart 12](#_Toc70846441)

[7 BGP service deployment 13](#_Toc70846442)

[7.1 ASN & IP PREFIX 13](#_Toc70846443)

[7.2 Topology 14](#_Toc70846444)

[7.3 Monitoring Stack 14](#_Toc70846445)

[7.4 Setup guide 14](#_Toc70846446)

# Outline

In Section 2, current UM-site Kubernetes Cluster information is showed. Kubernetes HA control-plane machines is hosted as VM running together in UM-playground-tower-2 physical machine. The worker nodes joining this cluster include nodes from Oftein++ members machine, such as Myanmar, Indonesia, Taiwan, Bhutan, and Malaysia. One storage machine in UM is used as the NFS Storage backened for the Kubernetes NFS Storage. In this cluster, Calico is used as the underlying Kubernetes Networking CNI.

In Section 3, system requirements tested during deployment are showed. Instructions to setup new node, cluster or joining a new node to existing cluster are documented.

In Section 4, the cluster services deployed in the current Kubernetes Cluster are introduced with their usage guidelines.

In Section 5, tools and tasks related in cluster maintenance are documented.

In Section 6, examples about deploying container in the cluster through yaml file or helm chart are showed.

In Section 7, BGP service deployment, an experiment carried in the cluster is documented.

# Cluster Information

## Cluster Architecture

Application

Description automatically generated with medium confidence

## IP Addressing

|  |  |
| --- | --- |
| Machine | IP address |
| smartx-microbox-drukren-1 | 103.133.216.15 |
| smartx-microbox-itb-1 | 167.205.51.41 |
| smartx-microbox-monash-1 | 203.80.21.39 |
| smartx-microbox-rub-1 | 103.133.216.211 |
| smartx-microbox-ucsm-1 | 103.47.184.186 |
| smartx-microbox-ucsy-1 | 103.137.86.4 |
| um-k8s-master1 | 203.80.21.29(203.80.21.30 Master) |
| um-k8s-master2 | 203.80.21.26 |
| um-ncku-worker1 | 140.116.215.202 |
| um-ncu-worker1 | 140.115.80.164 |
| um-sandbox-3 | 203.80.21.9 |

## Master nodes

|  |  |
| --- | --- |
| Virtual machine | Physical machine |
| um-k8s-master1 | Um-playground-tower-2(203.80.21.17) |
| um-k8s-master2 | Um-playground-tower-2(203.80.21.17) |

## High-Availability (HA) Cluster

Softwares used for HA feature:

Keepalived Version: 1:1.3.9-1ubuntu0.18.04.2

HAProxy Version: 1.8.8-1ubuntu0.11

|  |  |
| --- | --- |
| Master node | Keepalived role |
| um-k8s-master1 | Master |
| um-k8s-master2 | Slave |

Keepalived/ Kubernetes API endpoint IP address: 203.80.21.30

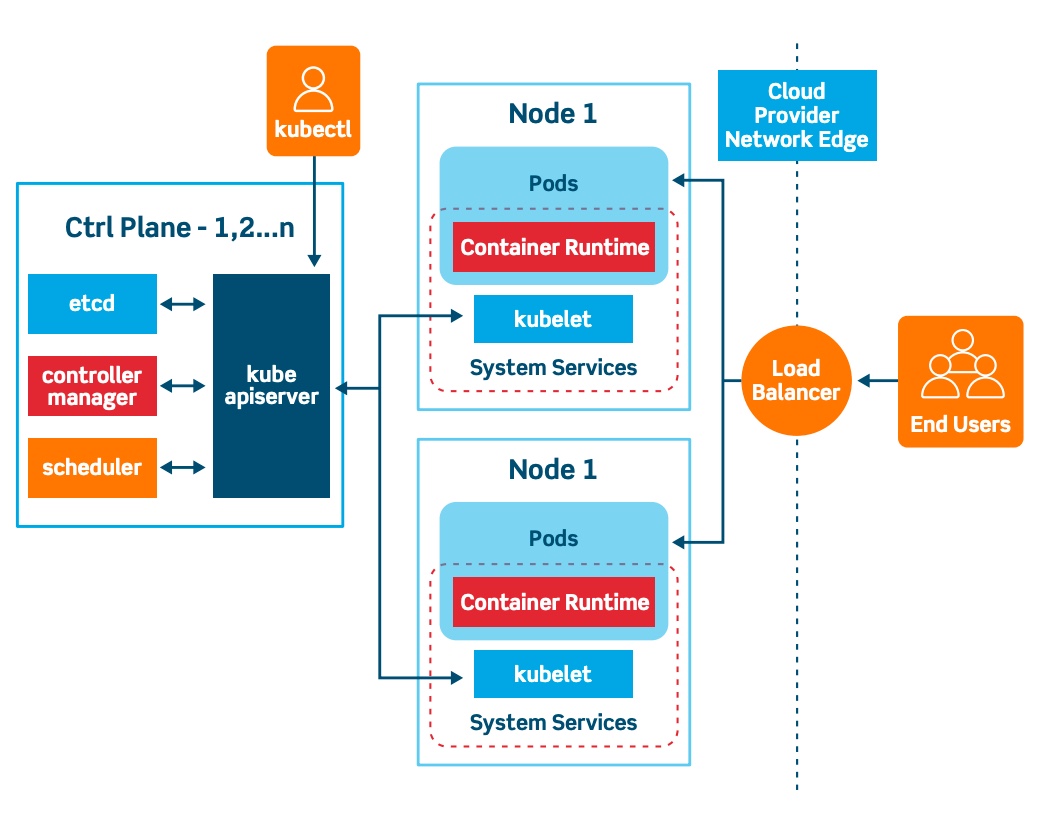
## Cluster Storage

NFS Storage is provided through Synology OFTEIN-DataLake at synology.oftein.fsktm.um.edu.my

## Networking

Calico is used as the underlying Kubernetes Network CNI. Calico nodes/pods are created at every Kubernetes worker node. They read Cluster Nodes information from the Kubernetes API EndPoints, which eventually read the information in the etcd database. A Calico node/pod (bigger IP address number) will form BGP neighbors with another Calico node/pod (smaller IP address number). Eventually forming a full mesh BGP peering between the nodes/pods.

# Cluster setup



## System requirements

System tested : Ubuntu 18.04 and Ubuntu 20.04

Ip Address : IP address should avoid NAT to prevent

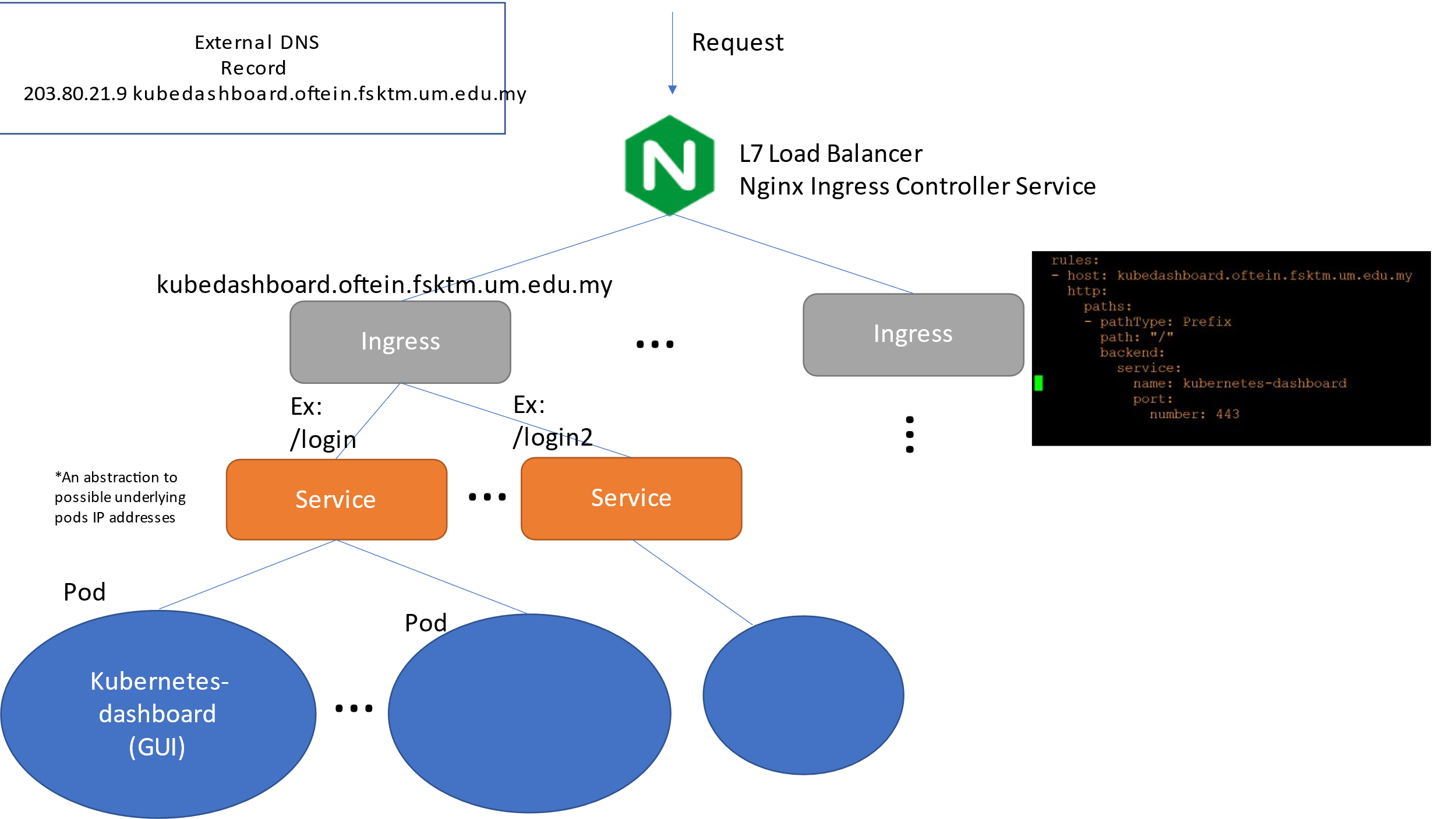
Kubernetes version tested : 1.19.03

## Creating new cluster or Joining new node

Refer to [Kubernetes Installation](https://github.com/skywood123/kubernetes-installation) or README.md file in /Kubernetes-Installation directory.

# Cluster Services

## Nginx Ingress Controller



*Traffic flow sequence with Kubernetes Ingress*

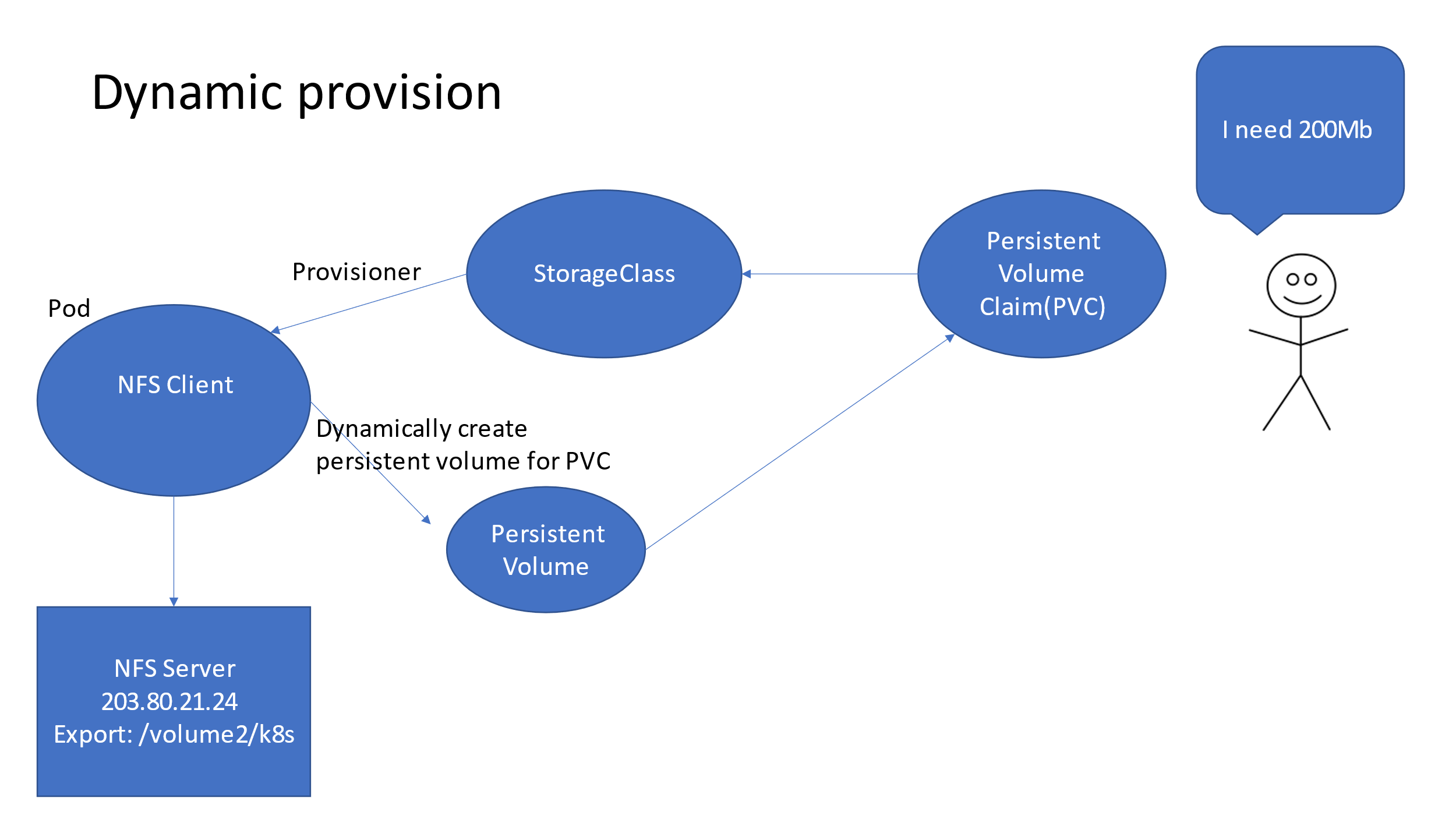
Function:

* Allow traffic load balancing between Kubernetes Services
* Allow external traffic to reach pods with the use of domain name stated in Ingress resource

How to use:

* Users deploy application pods and Kubernetes Service to reach the application
* Create Ingress yaml file and deploy Ingress object in the same namespace with the application deployed
* Add DNS record for the Ingress domain name in the DNS server

## NFS StorageClass



*Logic flow with PVC requests*

Function:

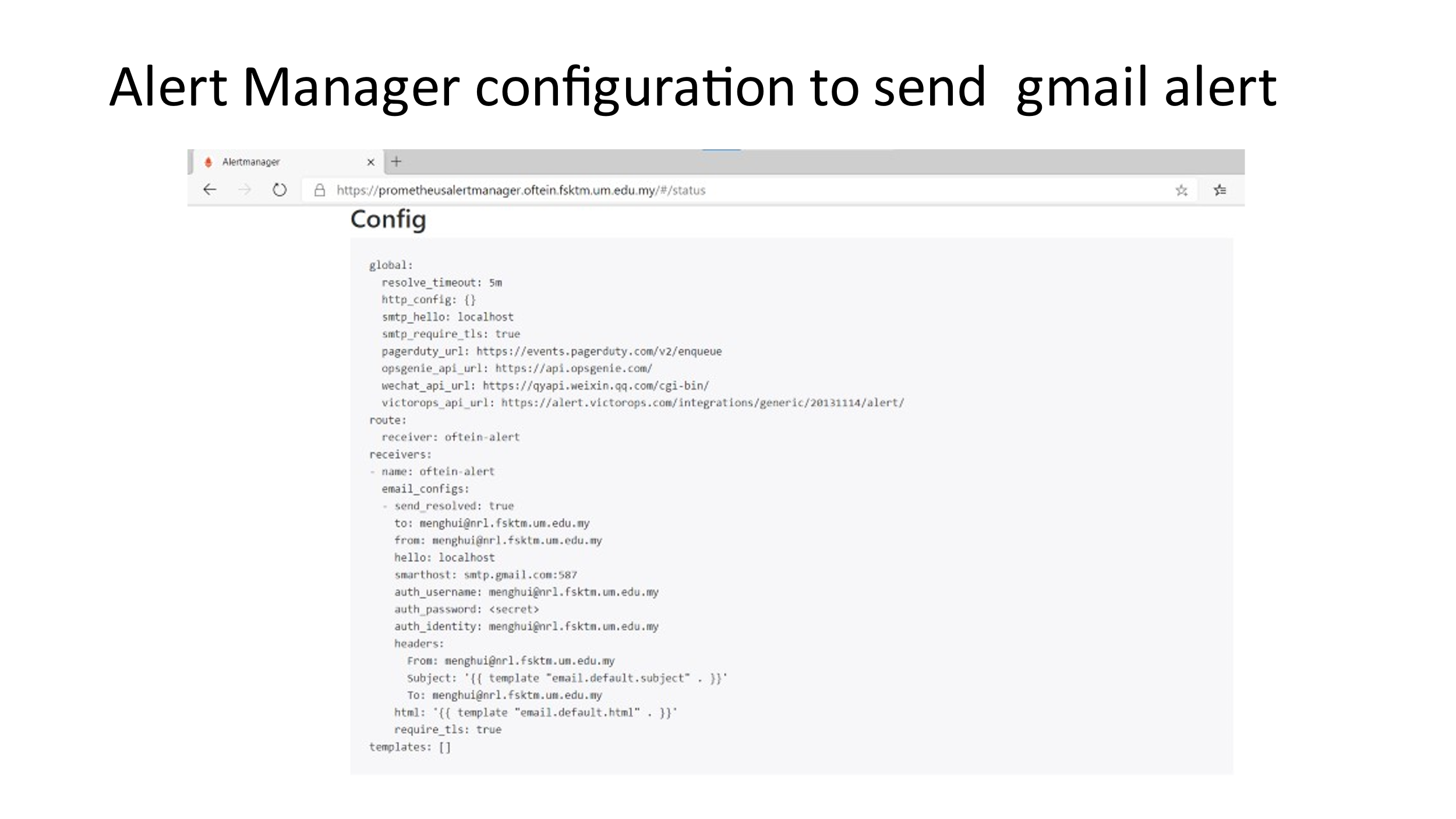
* Allow dynamic provision of persistent volume (PV) for permanent storage

How to use:

* User deployment yaml configuration file attached a persistent volume claim (PVC) to request PV from the storageclass **nfs-client**
* StorageClass is responsible to dynamically provision persistent volume for the deployment

## Prometheus Grafana Monitoring





Function:

* Monitor Kubernetes resources used: CPU, network traffic, memory, persistent volume, etc.
* Data are stored in Persistent Volume for 10 days by default.

How to use:

* Access <https://k8sgrafana.oftein.fsktm.um.edu.my/> to view monitoring data
* Prometheus alert-manager allows notification through email when threshold hit
* Loki can be used as the data source which aggregate static log file scraping from promtail

# Cluster maintenance

## Kubectl

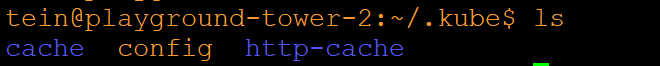
Kubectl is the tool to interact with the Kubernetes API endpoint in order to manage the cluster.

A **config** file is required to provide access information to the cluster.

Admin config file have full access to the cluster.

It is possible to create service accounts config file and distribute out for restricted cluster access, such as restricted to a specific namespace.

Admin config file is backup in the /credential folder



Example of kubectl interaction to the cluster

* $ kubectl get pods # get pods information in this namespace
* $ kubectl get pods -o wide # get pods full information in this namespace
* $ kubectl get pods -n rpki # get pods information in rpki namespace
* $ kubectl get nodes # get Kubernetes nodes(machine)
* $ kubectl get namespaces # get the list of namespaces
* $ kubectl apply -f helloworld.yaml # submit yaml file to the cluster
* $ kubectl delete -f helloworld.yaml # delete the yaml file from the cluster
* $ kubectl describe pod helloworld # describe more information about a pod
* $ kubectl get all # list resources generally in this namespace

## UM Oftein DNS Server

IP: 203.80.21.33/oftein.fsktm.um.edu.my

$ sudo -i

$ cd /etc/bind/

$ nano db.oftein # add/delete DNS record

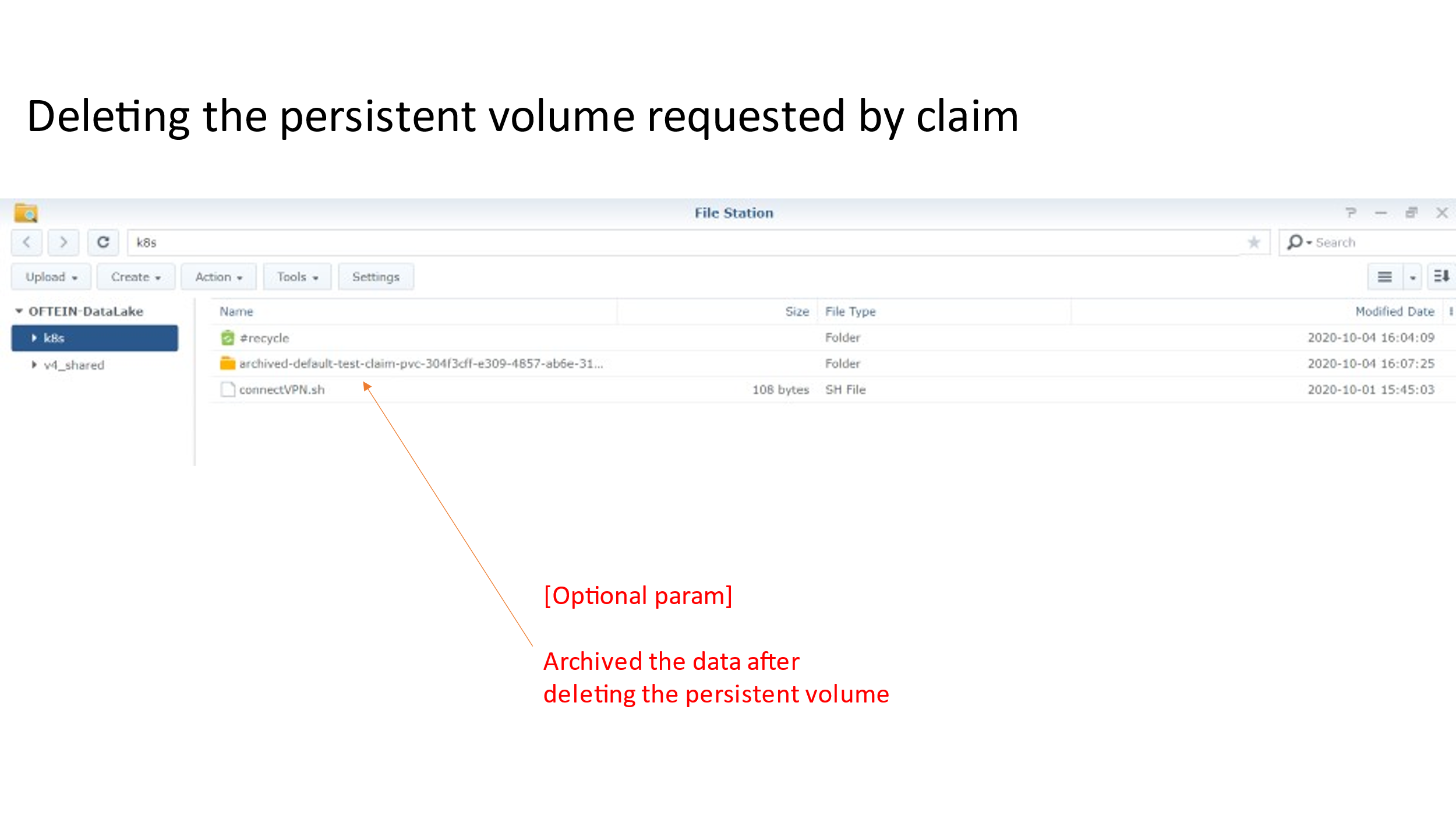
$ systemctl restart bind9 #restart DNS name server to update the database records

$ systemctl status bind9 #make sure status is active and running

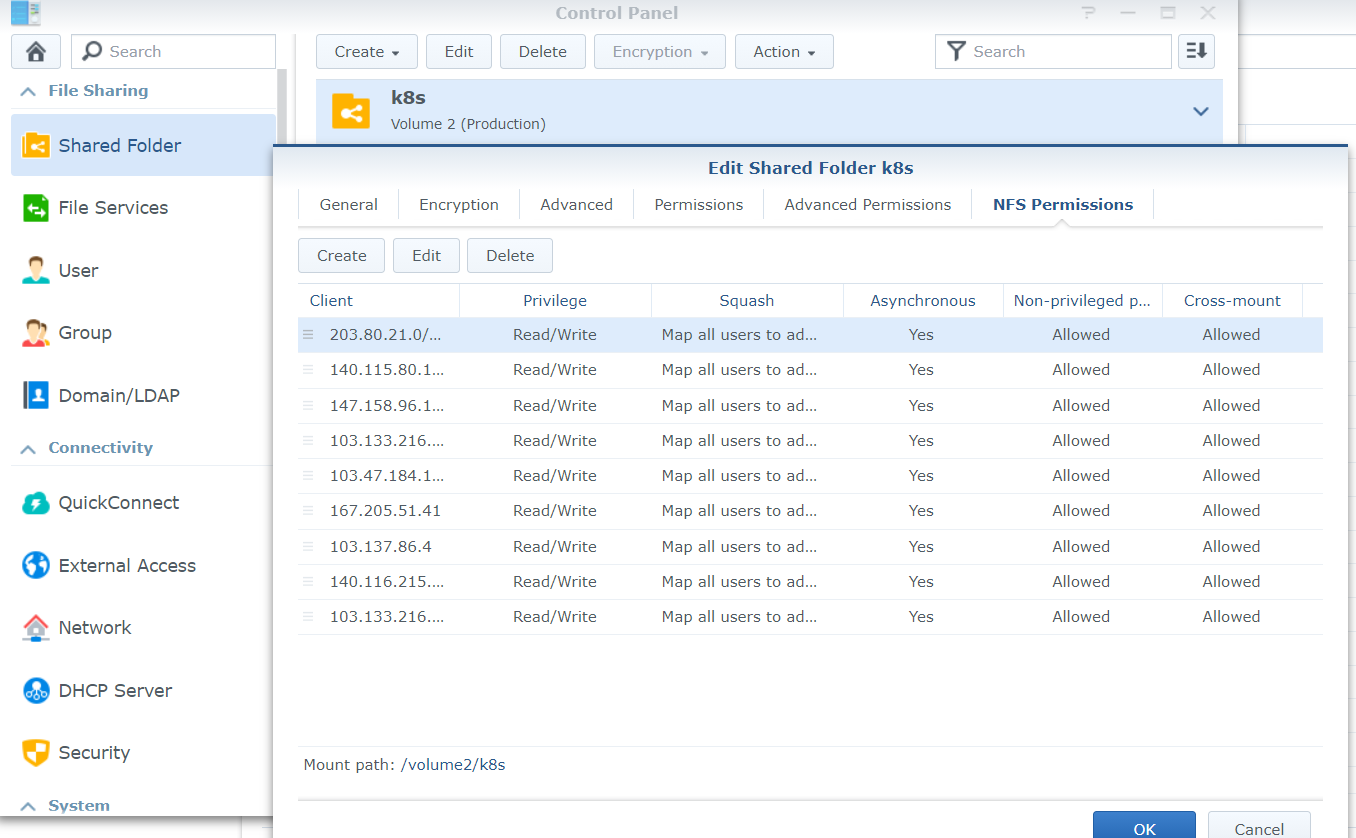
## Synology Data Storage

IP: 203.80.21.24/synology.oftein.fsktm.um.edu.my

* Persistent Volume folder can be archived even though the PV is deleted.



* If adding new worker node or ip address changes, need to manually add in the synology NFS folder for permission



## Helm Chart

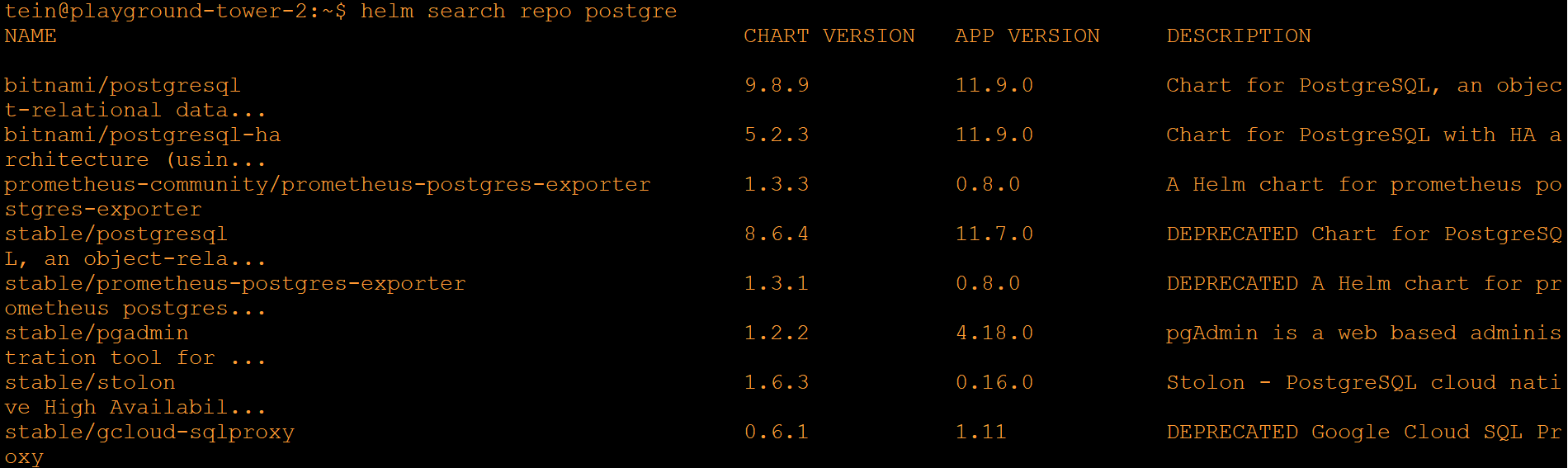
Helm Chart allow simpler way in deploying application

* Helm installation

<https://helm.sh/docs/intro/install/>

* $ helm search repo postgre

#Search result shows list of available postgre database from distributers and their versions

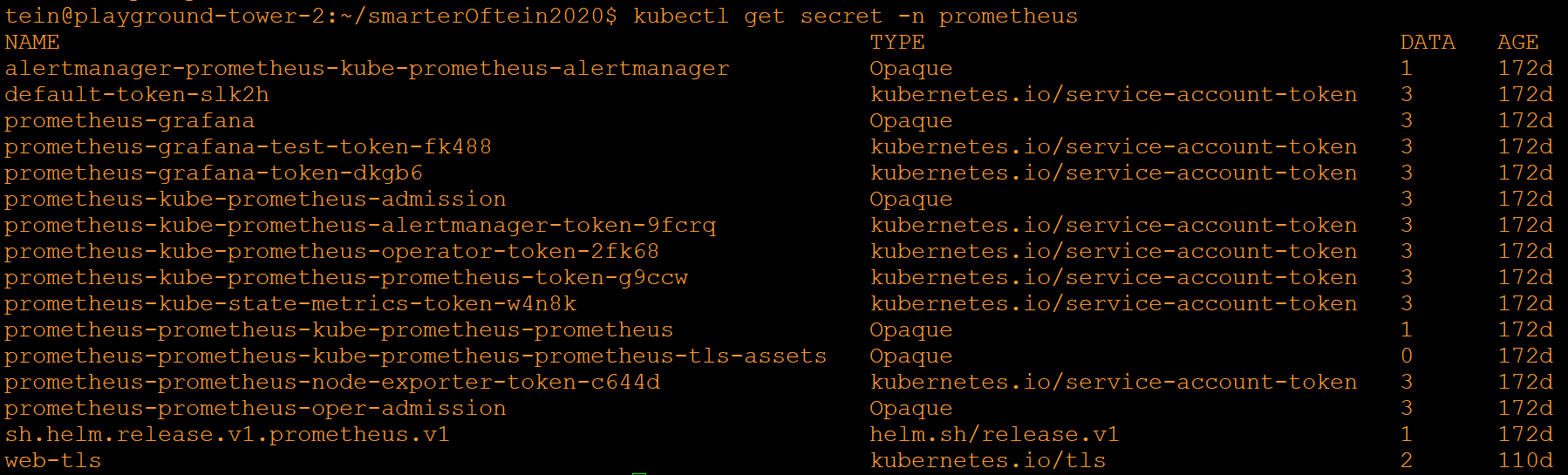


* $ helm install –name mypostgresqldemo -namespace hello bitnami/postgresql

#Install directly in the cluster

* Application in the helm normally comes with a values.yaml file allows us to fine tune and supply parameters to it. A way is to pull the application down locally, modify the values.yaml file, and only install in the cluster

## Renewing/Upload HTTP SSL certificates



* SSL Certificate is a Kubernetes Secret tls resource
* With the certificate and key, create a tls secret with the command

$ kubectl create secret tls testsecret --key privkey.pem --cert fullchain.pem

#creating tls secret

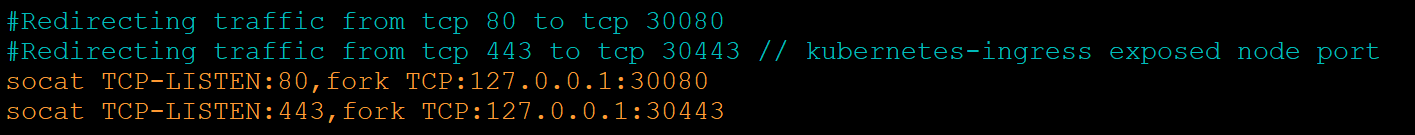
https://kubernetes.github.io/ingress-nginx/user-guide/tls/

kubectl create secret tls ${CERT\_NAME} --key ${KEY\_FILE} --cert ${CERT\_FILE}

#If updating existing secret, just delete it and recreate is fine.

#This is per namespace secret, need to update for the namespaces that using the secret.

## Port proxy with socat; Nginx Ingress Controller



* Nginx Ingress Controller, gained a nodeport access to preconfigured 30080 for http and 30443 for https access (This port range is fixed by Kubernetes 30000-32767)
* To get traffic from standard port 80 and 443, need to redirect traffic from port 80 and 443 to the 30080 and 30443 that nginx ingress controller is listening.
* At the machine where nginx Ingress Controller is running, execute the commands:

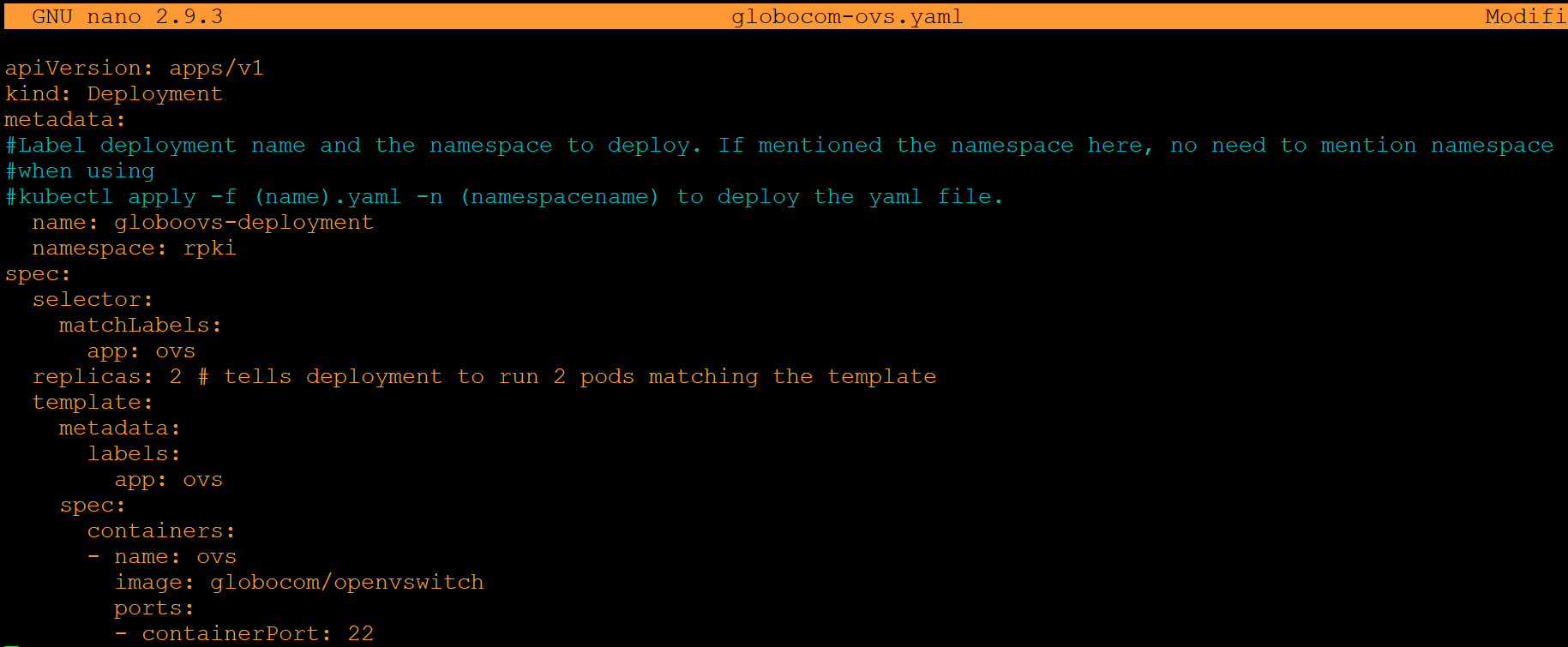
$ socat TCP-LISTEN:80,fork TCP:127.0.0.1:30080

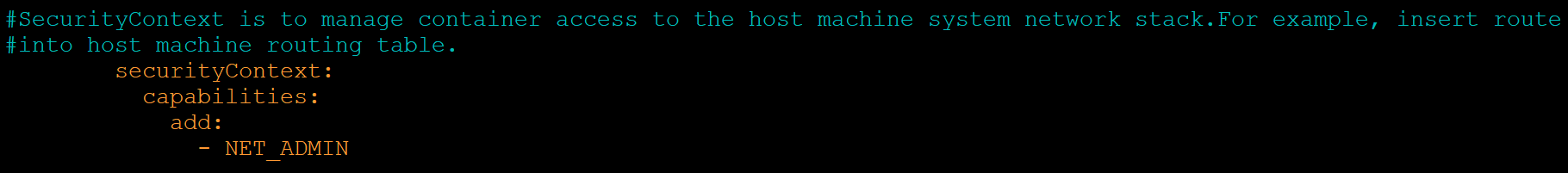
$ socat TCP-LISTEN:443,fork TCP:127.0.0.1:30443

# Container Deployment Example

## Deploy container from yaml file

1. Prepare a yaml file





1. Deploy into a namespace

$ kubectl apply -f globocom-ovs.yaml

1. Access the pod from terminal

$ kubectl exec --stdin --tty quagga-ovs-55dc55c565-2cxrr -- /bin/bash

## Deploy application through helm chart

#Helm Chart is like apt in Ubuntu

#Other distributed the integrated sets of yamls files for applications, ex. Postgresql database, Prometheus, etc.

$ helm install --name my-redis-release stable/redis

# BGP service deployment

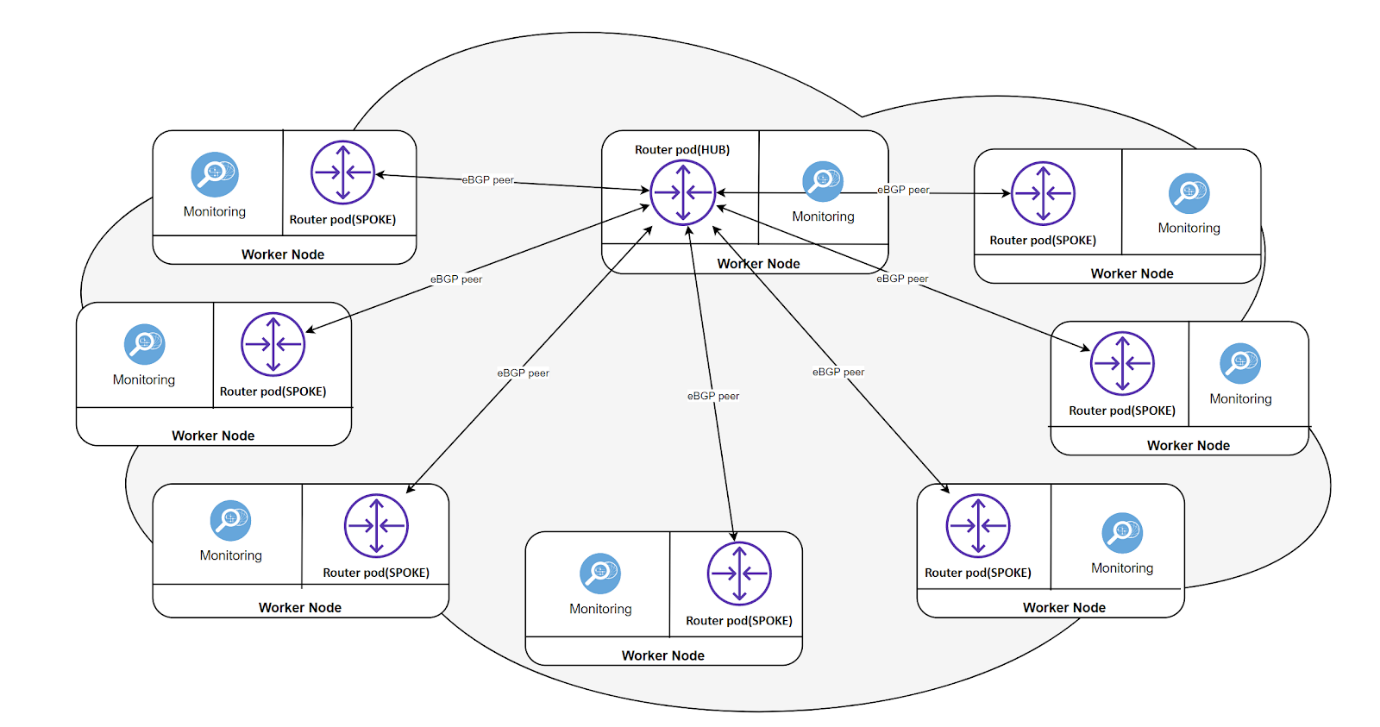
## ASN & IP PREFIX

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Worker node** | **Location** | **AS** | **Advertise network** | **interface** |
| smartx-microbox-monash-1 | Malaysia | 65011 | 172.16.0.0/23 | 192.168.100.4 |
| um-sandbox-3 | Malaysia | 65012 | 172.16.2.0/23 | 192.168.100.1 |
| smartx-microbox-itb-1 | Indonesia | 65021 | 172.16.4.0/23 | 192.168.100.3 |
| smartx-microbox-drukren-1 | Bhutan | 65031 | 172.16.6.0/23 | 192.168.100.2 |
| smartx-microbox-rub-1 | Bhutan | 65032 | 172.16.8.0/23 | 192.168.100.5 |
| smartx-microbox-ucsm-1 | Myammar | 65041 | 172.16.10.0/23 | 192.168.100.8 |
| um-ncu-worker1 | Taiwan | 65051 | 172.16.12.0/23 | 192.168.100.6 |
| um-ncku-worker1 | Taiwan | 65052 | 172.16.14.0/23 | 192.168.100.7 |

## Topology

Hub and Spoke BGP Peering

Hub : um-sandbox-3



## Monitoring Stack

* Router pods (Quagga) log the syslog message into a .log file
* Configure Promtail to scrape this file
* Configure Promtail to reach loki (send the log stream with labeled)
* View aggregated messages from Grafana

A picture containing diagram

Description automatically generated

## Setup guide

* Refer [OFTEIN-Router-and-Monitoring-Pod](https://github.com/OFTEIN-NET/OFTEIN-Router-and-Monitoring-Pod) or /bgp-deployment directory