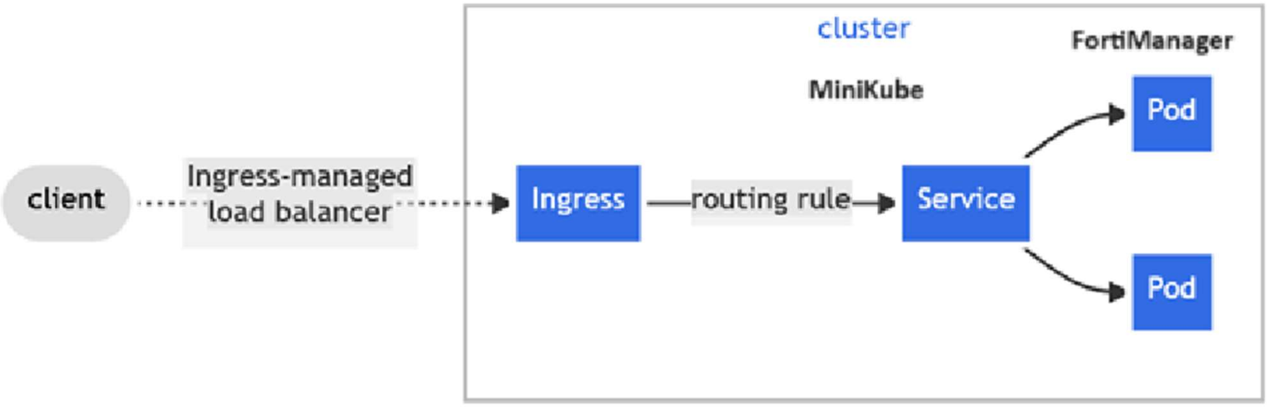
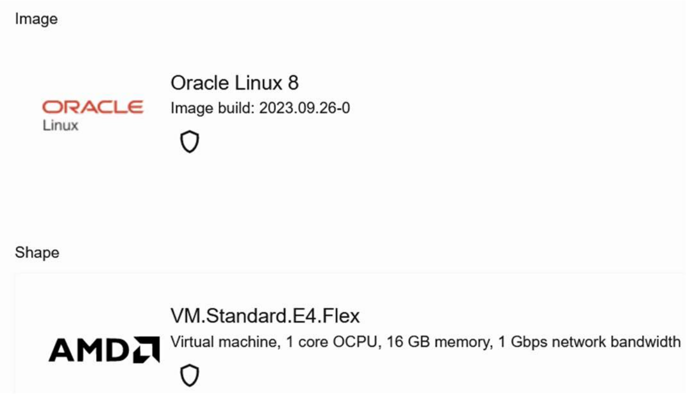
Disclaimer: This tutorial is created for the purpose of understanding basic K8S and creating a simple containerized FortiManager and FortiAnalyzer using Minikube Cluster and is not intended for production purposes.

Architecture Design



You can perform this hands-on lab on either your MacOS, Windows or any CSPs of your choice. In this demo, I am using the below specifications. It is best recommended to have at least 2vCPU, 4GB RAM, 20GB Disk.



Image

Oracle Linux 8
Image build: 2023.09.26-0

Shape

VM.Standard.E4.Flex
Virtual machine, 1 core OCPU, 16 GB memory, 1 Gbps network bandwidth

1. Login to your Linux and open up your favourite terminal e.g. MobaXTerm. Ensure Linux is update to date. This might take awhile, grab a cup of coffee while waiting.

   sudo dnf -y update

2. Install Podman and other required packages. Hold on, what's Podman? It's basically an alternative of Docker which is a daemonless, open source, Linux native tool designed to make it easy to find, run, build, share and deploy applications using Open Containers Initiative (OCI) Containers and Container Images. In short, it's like "Driver" for Container Runtime. There are also other alternative such as Docker, KVM2, VirtualBox, QEMU and etc. However, for testing purpose, we are going to use Podman.

   sudo dnf install -y podman podman-docker conntrack

3. Ensure curl is installed as we are going to use curl to install our first Minikube Cluster

   sudo dnf -y install curl

4. Now, we are going to begin our first Kubernetes using a lighter weight version called MiniKube.

Let’s download the binary first

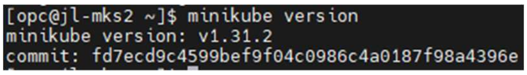
cd ~; curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64

Next, we will install it.

sudo install minikube-linux-amd64 /usr/local/bin/minikube

Let’s verify if our Minikube is installed successfully?

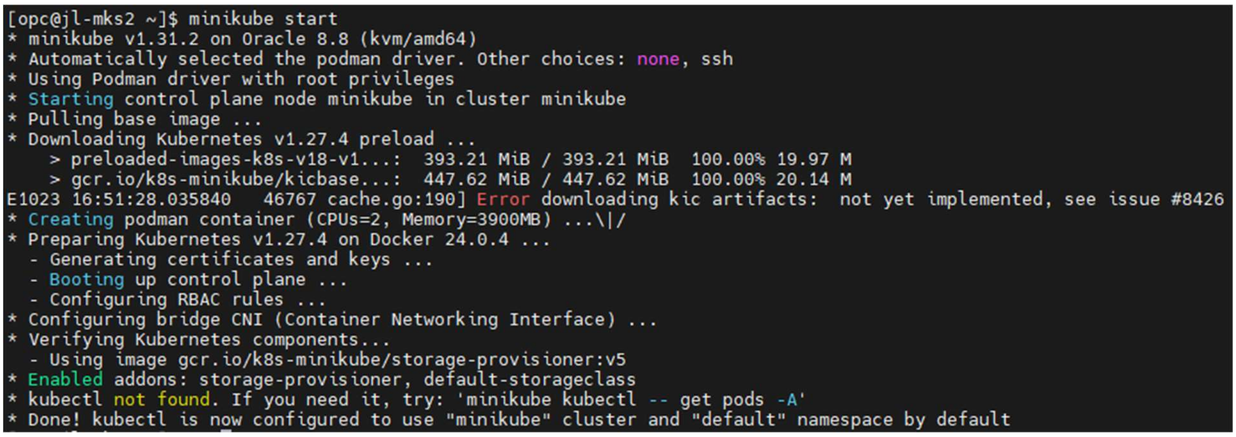
minikube version



```
[opc@jl-mks2 ~]$ minikube version
minikube version: v1.31.2
commit: fd7ecd9c4599bef9f04c0986c4a0187f98a4396e
```

5. Since we are using Podman Driver, we are going to start the Minikube Cluster with Podman driver so it will be a complete set. Again, this might take a minute to start.

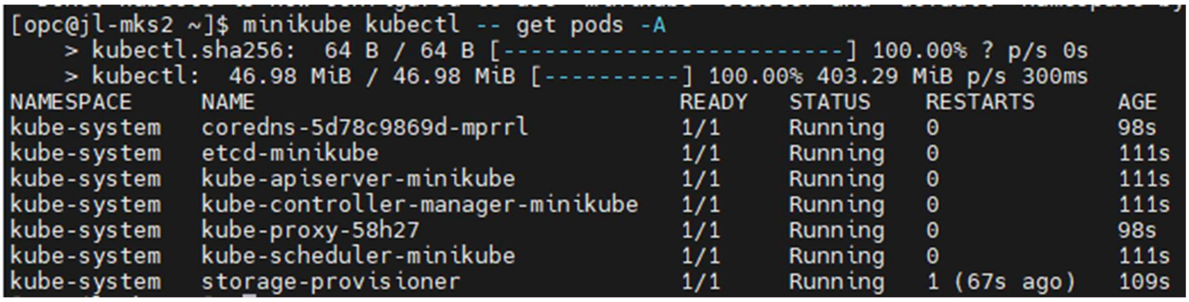
   minikube start --driver=podman



```
[opc@jl-mks2 ~]$ minikube start
* minikube v1.31.2 on Oracle 8.8 (kvm/amd64)
* Automatically selected the podman driver. Other choices: none, ssh
* Using Podman driver with root privileges
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Downloading Kubernetes v1.27.4 preload ...
    > preloaded-images-k8s-v18-v1...:  393.21 MiB / 393.21 MiB  100.00% 19.97 M
    > gcr.io/k8s-minikube/kicbase...:  447.62 MiB / 447.62 MiB  100.00% 20.14 M
E1023 16:51:28.035840   46767 cache.go:190] Error downloading kic artifacts:  not yet implemented, see issue #8426
* Creating podman container (CPUs=2, Memory=3900MB) ...\|/
* Preparing Kubernetes v1.27.4 on Docker 24.0.4 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* kubectl not found. If you need it, try: 'minikube kubectl -- get pods -A'
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Note: Error downloading kic artifacts: not yet implemented, see issue #8426 is a known issue in the current version of Podman, which the developer plans to resolve in a future version (This doesn’t impact most deployment scenarios) so you can ignore it.

6. Confirm that all the installed systems are functioning properly
   minikube kubectl -- get pods -A



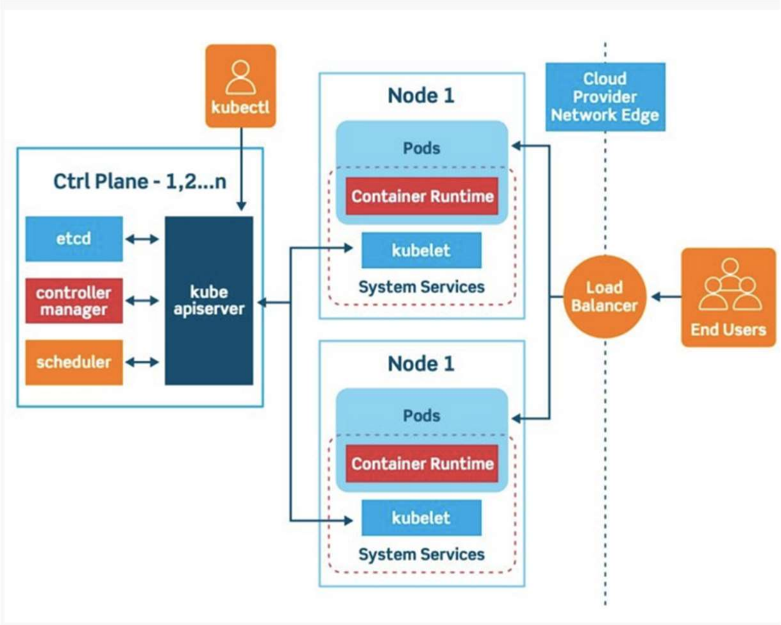
```
[opc@jl-mks2 ~]$ minikube kubectl -- get pods -A
    > kubectl.sha256:   64 B / 64 B [-------------------------] 100.00% ? p/s 0s
    > kubectl:  46.98 MiB / 46.98 MiB [----------] 100.00% 403.29 MiB p/s 300ms
NAMESPACE     NAME                               READY   STATUS    RESTARTS      AGE
kube-system   coredns-5d78c9869d-mprrl           1/1     Running   0             98s
kube-system   etcd-minikube                      1/1     Running   0             111s
kube-system   kube-apiserver-minikube            1/1     Running   0             111s
kube-system   kube-controller-manager-minikube   1/1     Running   0             111s
kube-system   kube-proxy-58h27                   1/1     Running   0             98s
kube-system   kube-scheduler-minikube            1/1     Running   0             111s
kube-system   storage-provisioner                1/1     Running   1 (67s ago)   109s
```

Wait, you might have seen before kubectl? It’s basically pretty much like a CLI command to begin with. Something like FortiGate “exec”. To make things easier, we will be adding an alias for “Kubectl” instead of keep repeating the lengthy “minikube kubectl” command at the very start

echo 'alias kubectl="minikube kubectl --"' >> ~/.bashrc
source ~/.bashrc

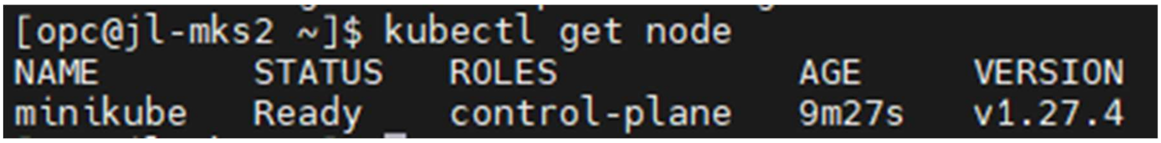
7. Alright, now we have a small K8S cluster with a single control-plane (typically called Master Node) for testing. In real-life, there will be multiple Master and Worker Nodes to serve different purposes. You can read up online https://platform9.com/blog/kubernetes-enterprise-chapter-2-kubernetes-architecture-concepts/ on what's the purpose of Master and Worker as well, it has many components within it. For now, you can continue with this hands-on to get a feel of it first.

   Overview of K8S Architecture

   Architectural overview of Kubernetes

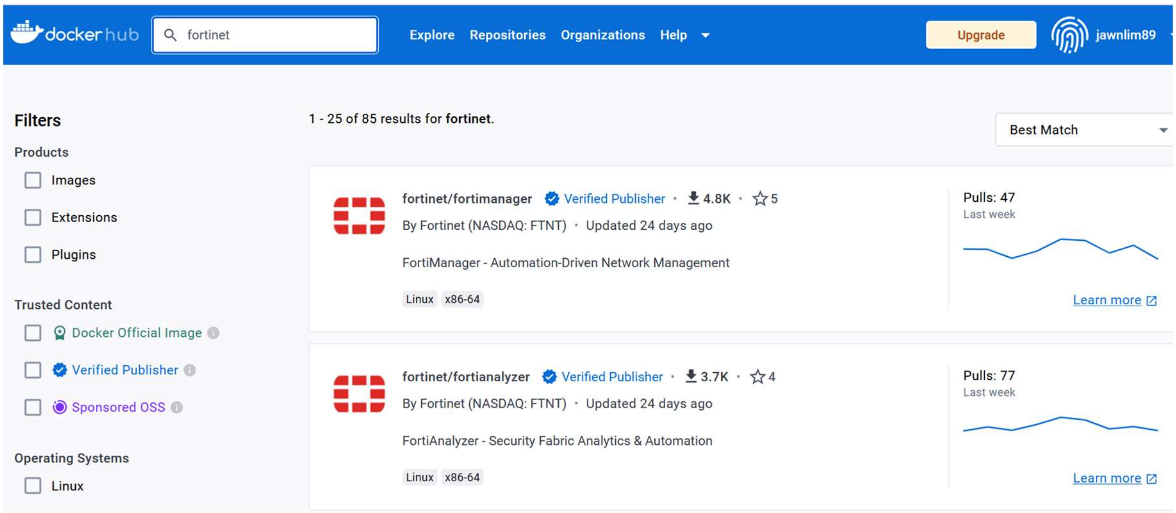
   You can start to view your master node (control-plane) with **kubectl get nodes**

```
[opc@jl-mks2 ~]$ kubectl get node
NAME       STATUS   ROLES           AGE     VERSION
minikube   Ready    control-plane   9m27s   v1.27.4
```

8. Next, having a Minikube Cluster UP and RUNNING is as good as having an Empty VMWARE, or Hyper-V waiting for applications to be installed right? So, we need to first, find the image of Fortinet (In this case, FortiManager and FortiAnalyzer). Let's start with FortiManager. In Docker, we use docker image (similar like how you import ISO image to VMWARE). If you visit DockerHub https://hub.docker.com/search?q=fortinet, you should be able to find the image with the path of

   - fortinet/fortimanager
   - fortinet/fortianalyzer

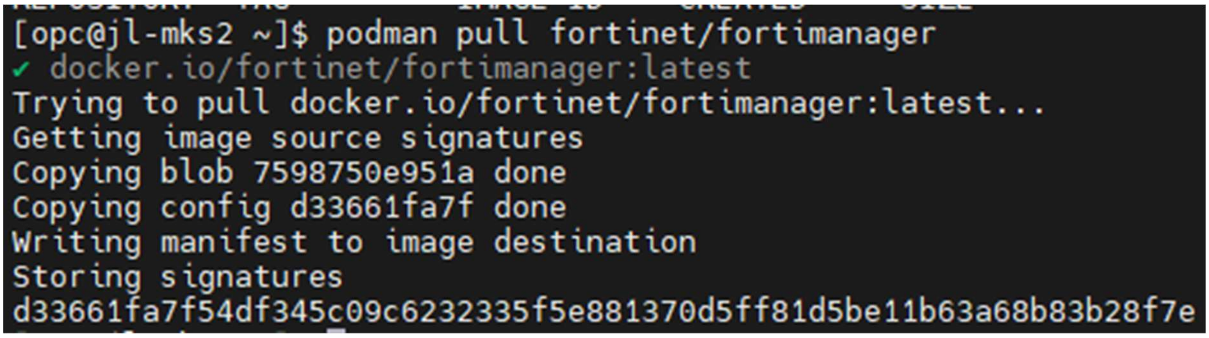


1 - 25 of 85 results for **fortinet**.

fortinet/fortimanager · Verified Publisher · 4.8K · 5
By Fortinet (NASDAQ: FTNT) · Updated 24 days ago
FortiManager - Automation-Driven Network Management
Linux x86-64
Pulls: 47
Last week
Learn more

fortinet/fortianalyzer · Verified Publisher · 3.7K · 4
By Fortinet (NASDAQ: FTNT) · Updated 24 days ago
FortiAnalyzer - Security Fabric Analytics & Automation
Linux x86-64
Pulls: 77
Last week
Learn more

9. [Optional -> This step is optional as during the deployment of container, it can also grab the image from the internet even without importing the image first. However, it would be good for you to learn this as in real-life, customers will have their own custom or golden image which they will want to pull from their own repository instead of the cloud repository] We need to import this image in, how do we do that? In docker, we use the command **docker pull fortinet/fortimanager,** however, I am using podman, you can just switch to **podman pull fortinet/fortimanager** and choose the correct image when prompted. (Use Up, Down Key and press enter)

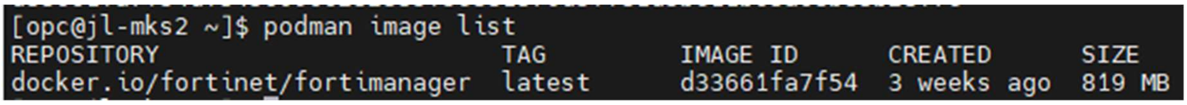


```
[opc@jl-mks2 ~]$ podman pull fortinet/fortimanager
✓ docker.io/fortinet/fortimanager:latest
Trying to pull docker.io/fortinet/fortimanager:latest...
Getting image source signatures
Copying blob 7598750e951a done
Copying config d33661fa7f done
Writing manifest to image destination
Storing signatures
d33661fa7f54df345c09c6232335f5e881370d5ff81d5be11b63a68b83b28f7e
```

Let's verify that the image is successfully pull over to your Minikube Cluster.

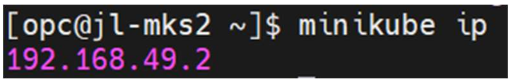
Podman image list

Alternatively, if you want a specific version, please specify the version at the end during the pull action. e.g podman **pull fortinet/fortimanager:7.2.2**



```
[opc@jl-mks2 ~]$ podman image list
REPOSITORY                    TAG         IMAGE ID      CREATED      SIZE
docker.io/fortinet/fortimanager  latest   d33661fa7f54  3 weeks ago  819 MB
```

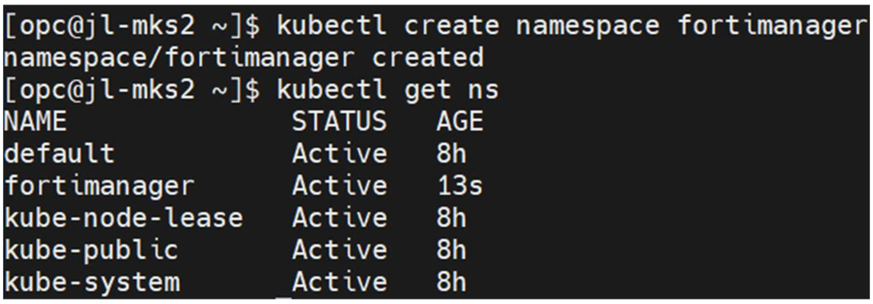
You can also check your **"minikube ip"** with this command



```
[opc@jl-mks2 ~]$ minikube ip
192.168.49.2
```

10. Let's create a namespace with the name of Fortimanager. What's namespace? If you are familiar with Linux namespace, it's basically partitioning kernel resources such that one set of processes sees one set of resources. In short, it's like separating environments.

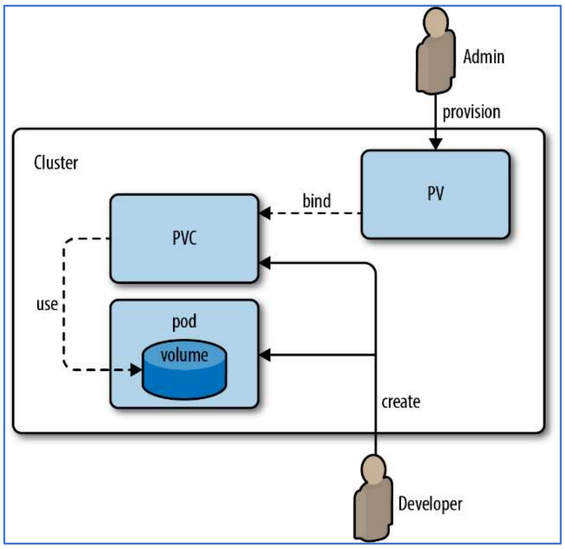
kubectl create namespace fortimanager



```
[opc@jl-mks2 ~]$ kubectl create namespace fortimanager
namespace/fortimanager created
[opc@jl-mks2 ~]$ kubectl get ns
NAME              STATUS   AGE
default           Active   8h
fortimanager      Active   13s
kube-node-lease   Active   8h
kube-public       Active   8h
kube-system       Active   8h
```

11. Next, we will need to create a storage so our container can use right? In container, we typically call it Persistent Volume (PV) and Persistent Volume Claim (PVC). What's the difference between these 2? PV is basically a BIG storage for the Cluster to use, it can be created either in dynamic or static manner. PV can be in many forms (e.g. NFS, ISCSI, CSP Cloud Disks). On the other hand, A Persistent Volume Claim (PVC) is a request for storage by a user. PVC is a subset of PV thus whenever a user request PVC, it will kind of slice from the PV thus both have tight dependencies. One thing to take note, if you don't have PV/PVC, even if you are able to create container with an application running inside, if the container got terminated or recreated, data will be deleted as it is just a temporary disk. Thus, it is very IMPORTANT to have PV/PVC bound to container.

Architecture of the relationship between PV and PVC

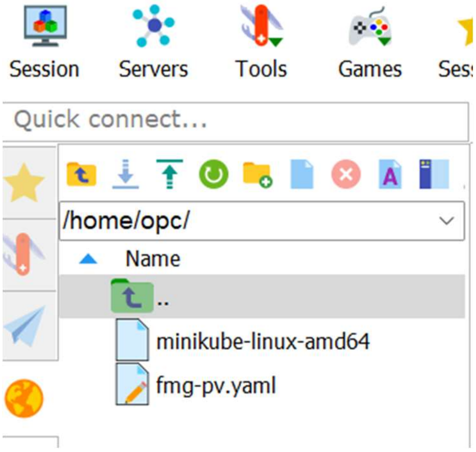


Let's begin by creating 2 PVC in a single yaml file, one for /var and one for /data. Remember I mentioned that PV can be created dynamically, below script will dynamically create PV even though I didn't specify PV in the script. Only PVC is specified.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: fmgvar
  namespace: fortimanager
spec:
  resources:
    requests:
      storage: 20Gi
  accessModes:
    - ReadWriteOnce
---
apiVersion: v1
kind: PersistentVolumeClaim
```

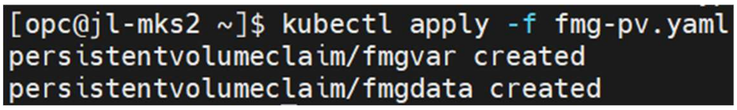
```
metadata:
  name: fmgdata
  namespace: fortimanager
spec:
  resources:
    requests:
      storage: 10Gi
  accessModes:
    - ReadWriteOnce
```

Upload your yaml (fmg-pv.yaml) or create a new yaml and copy and paste the above script.



Apply the yaml file so it will take effect.

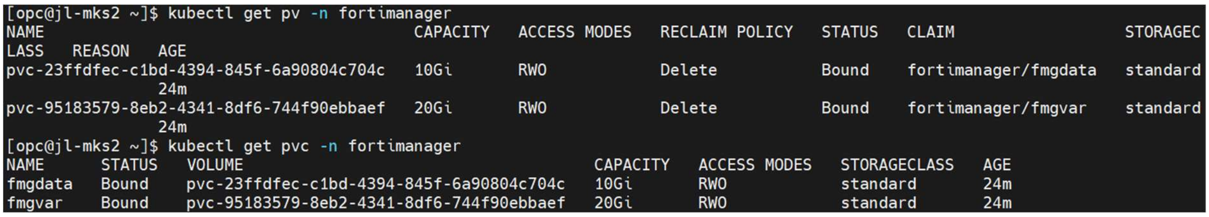
kubectl apply -f fmg-pv.yaml



```
[opc@jl-mks2 ~]$ kubectl apply -f fmg-pv.yaml
persistentvolumeclaim/fmgvar created
persistentvolumeclaim/fmgdata created
```

Verify that the PV and PVC are created successfully.

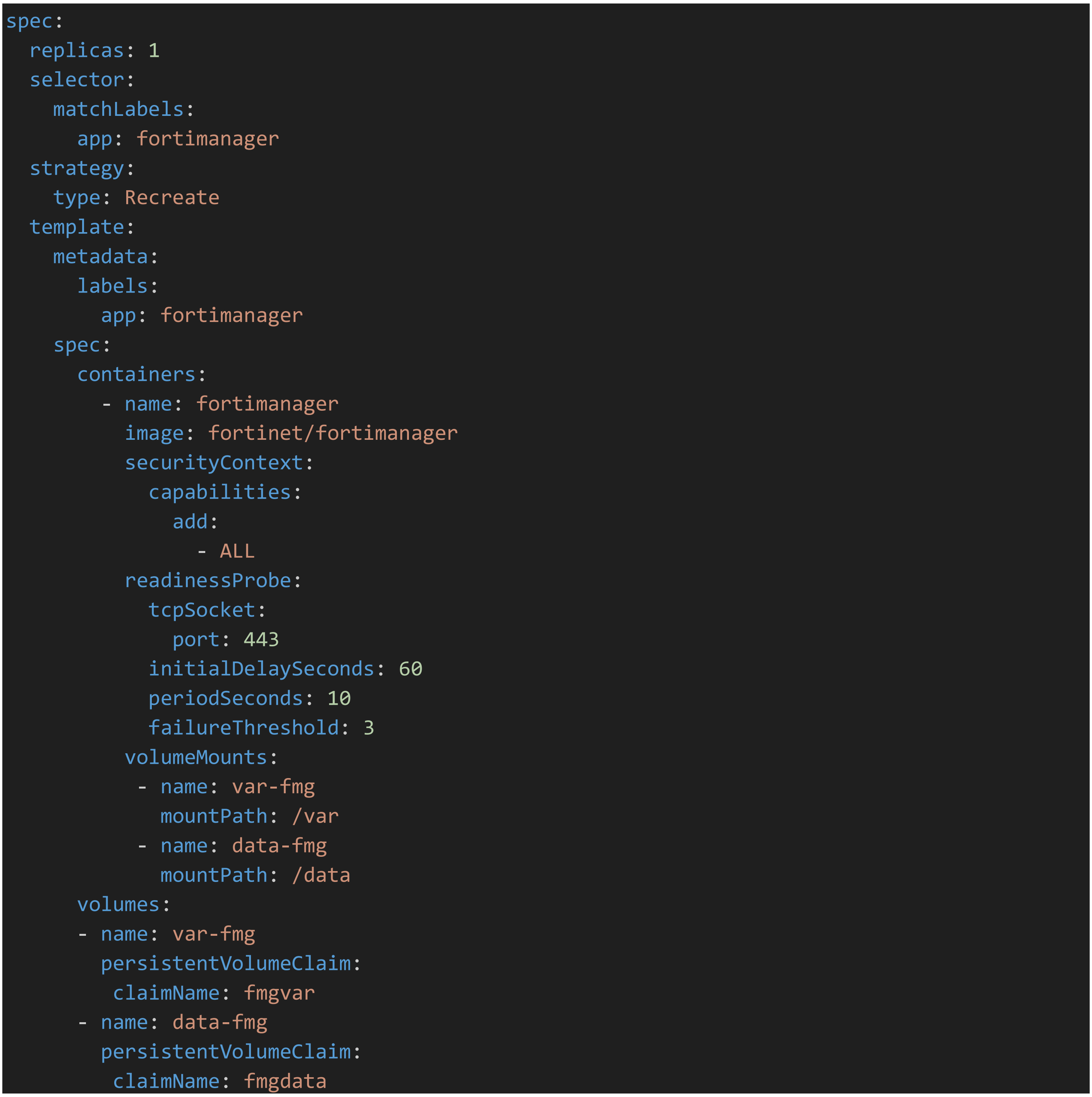
kubectl get pv or pvc -n fortimanager



```
[opc@jl-mks2 ~]$ kubectl get pv -n fortimanager
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                      STORAGEC
LASS    REASON   AGE
pvc-23ffdfec-c1bd-4394-845f-6a90804c704c   10Gi       RWO            Delete           Bound    fortimanager/fmgdata   standard
                 24m
pvc-95183579-8eb2-4341-8df6-744f90ebbaef   20Gi       RWO            Delete           Bound    fortimanager/fmgvar    standard
                 24m
[opc@jl-mks2 ~]$ kubectl get pvc -n fortimanager
NAME      STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
fmgdata   Bound    pvc-23ffdfec-c1bd-4394-845f-6a90804c704c   10Gi       RWO            standard       24m
fmgvar    Bound    pvc-95183579-8eb2-4341-8df6-744f90ebbaef   20Gi       RWO            standard       24m
```

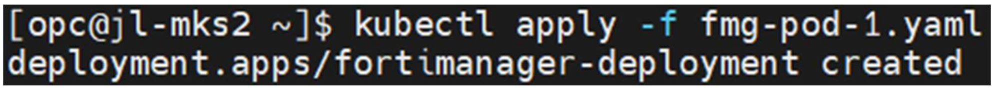
12. After the PV/PVCs are created, we can start to deploy Fortimanager and attach the volume to the pod. Same thing, we will create a yaml (fmg-pod-1.yaml) or you can create a new yaml and copy the below script.
    Notice in this script, we created this Fortimanager in the same namespace so it can recognize the volumes. We also set the replica to 1 so a single container will be created.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: fortimanager-deployment
  namespace: fortimanager
```



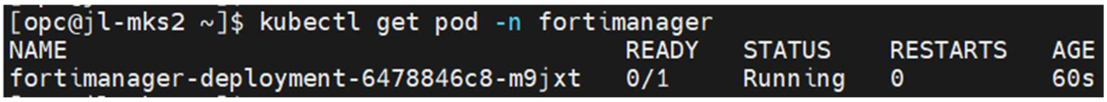
```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: fortimanager
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: fortimanager
    spec:
      containers:
        - name: fortimanager
          image: fortinet/fortimanager
          securityContext:
            capabilities:
              add:
                - ALL
          readinessProbe:
            tcpSocket:
              port: 443
            initialDelaySeconds: 60
            periodSeconds: 10
            failureThreshold: 3
          volumeMounts:
           - name: var-fmg
             mountPath: /var
           - name: data-fmg
             mountPath: /data
      volumes:
      - name: var-fmg
        persistentVolumeClaim:
         claimName: fmgvar
      - name: data-fmg
        persistentVolumeClaim:
         claimName: fmgdata
```

Let’s apply it with **kubectl apply -f fmg-pod-1.yaml**



```
[opc@jl-mks2 ~]$ kubectl apply -f fmg-pod-1.yaml
deployment.apps/fortimanager-deployment created
```

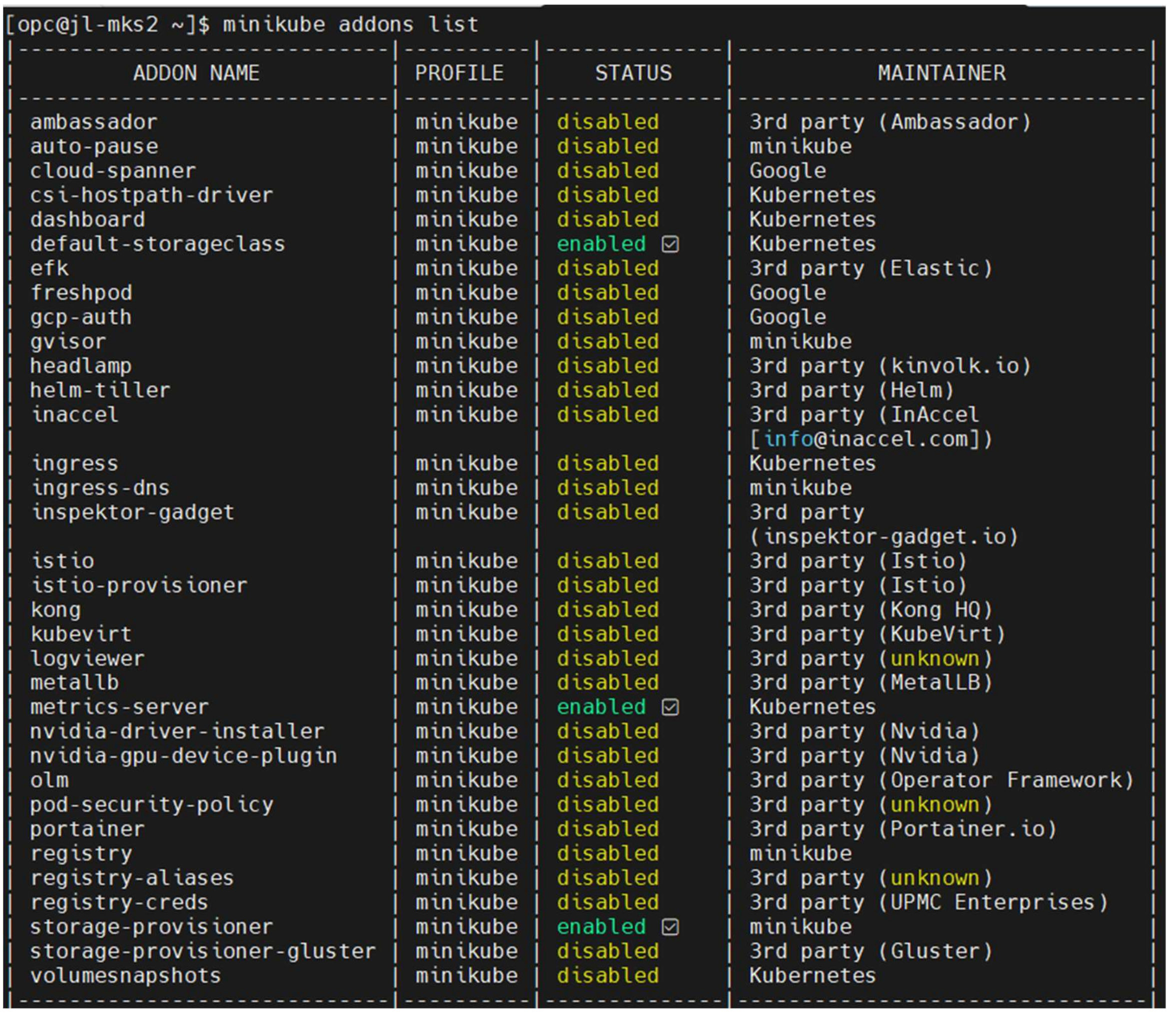
Verify that the pod is successfully created **kubectl get pod -n fortimanager**



```
[opc@jl-mks2 ~]$ kubectl get pod -n fortimanager
NAME                                      READY   STATUS    RESTARTS   AGE
fortimanager-deployment-6478846c8-m9jxt   0/1     Running   0          60s
```

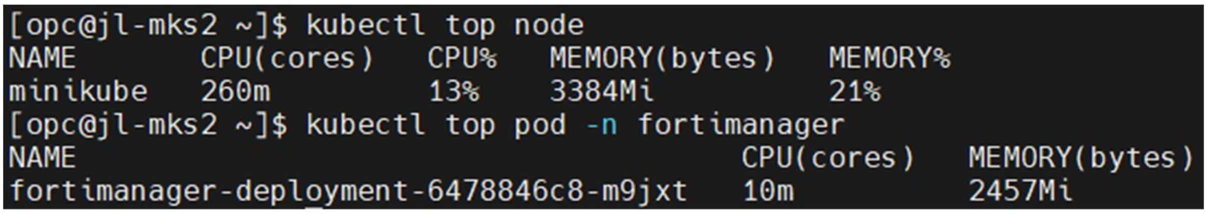
You can also enable the metrics-server to view more metrics of the container.

minikube addons enable metrics-server



```
[opc@jl-mks2 ~]$ minikube addons list
|-----------------------------|----------|--------------|--------------------------------|
|         ADDON NAME          | PROFILE  |    STATUS    |           MAINTAINER           |
|-----------------------------|----------|--------------|--------------------------------|
| ambassador                  | minikube | disabled     | 3rd party (Ambassador)         |
| auto-pause                  | minikube | disabled     | minikube                       |
| 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     | minikube                       |
| 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 | disabled     | Kubernetes                     |
| ingress-dns                 | minikube | disabled     | minikube                       |
| inspektor-gadget            | minikube | disabled     | 3rd party                      |
|                             |          |              | (inspektor-gadget.io)          |
| 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     | 3rd party (Nvidia)             |
| 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     | minikube                       |
| registry-aliases            | minikube | disabled     | 3rd party (unknown)            |
| registry-creds              | minikube | disabled     | 3rd party (UPMC Enterprises)   |
| storage-provisioner         | minikube | enabled ☑    | minikube                       |
| storage-provisioner-gluster | minikube | disabled     | 3rd party (Gluster)            |
| volumesnapshots             | minikube | disabled     | Kubernetes                     |
|-----------------------------|----------|--------------|--------------------------------|
```

After enabled metrics-server (will take awhile to see result), you can view current resource utilization of the Node or Pod.



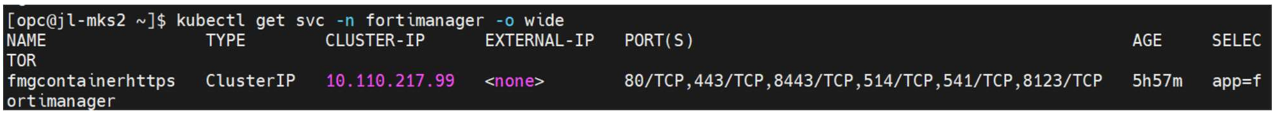
```
[opc@jl-mks2 ~]$ kubectl top node
NAME       CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
minikube   260m         13%    3384Mi          21%
[opc@jl-mks2 ~]$ kubectl top pod -n fortimanager
NAME                                   CPU(cores)   MEMORY(bytes)
fortimanager-deployment-6478846c8-m9jxt   10m          2457Mi
```

13. Next, we will need to create a service to expose the backend container to internet so you will be able to access the Fortimanager console later. This service is very important as it acts a method for exposing a network application that is running as one or more Pods in your cluster. Service can be created in 3 different modes (NodePort, ClusterIP, Load Balancer). We will use Cluster IP for now. Same thing, let's create a yaml (fmg-svc-1.yaml) or you can create a new yaml and copy the below script.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: fmgcontainerhttps
spec:
  sessionAffinity: ClientIP
```

```
ports:
- port: 80
  name: web
  protocol: TCP
  targetPort: 80
- port: 443
  name: webgui
  protocol: TCP
  targetPort: 443
- port: 8443
  name: webgui2
  protocol: TCP
  targetPort: 8443
- port: 514
  name: oftpd
  protocol: TCP
  targetPort: 514
- port: 541
  name: fgfm
  protocol: TCP
  targetPort: 541
- port: 8123
  name: mast
  protocol: TCP
  targetPort: 8123
- port: 8443
  name: webg
  protocol: TCP
  targetPort: 8443
selector:
 app: fortimanager
type: ClusterIP
```

Apply the fmg-svc-1.yaml file by using **kubectl apply -f fmg-svc-1.yaml** and verify the service is successfully created **kubectl get svc -n fortimanager -o wide**

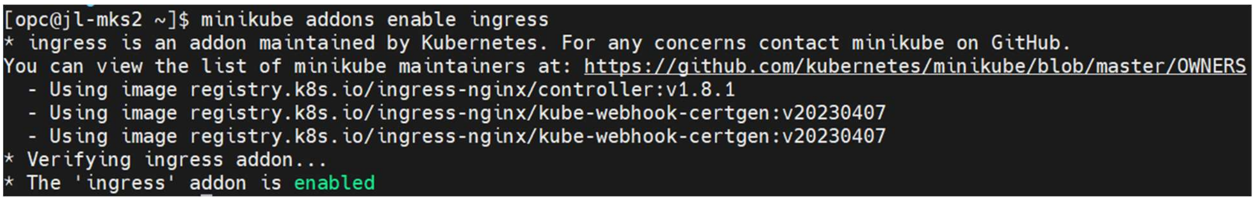


```
[opc@jl-mks2 ~]$ kubectl get svc -n fortimanager -o wide
NAME                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                                           AGE     SELEC
TOR
fmgcontainerhttps    ClusterIP   10.110.217.99   <none>        80/TCP,443/TCP,8443/TCP,514/TCP,541/TCP,8123/TCP   5h57m   app=f
ortimanager
```

14. Next, we need to create an ingress to forward the traffic to the service. Wait, why do we still need ingress here when we already have service which is already a Load Balancer to serve externally? In real life scenario, there will be many services targeting to the PODs which may not feasible or cost efficient if you have too many load balancers as services. By having Ingress, it is at layer 7, one can specify the HTTP or HTTPS request and different path can be set. For instance, www.abc.com/aboutus and www.abc.com/contactus.

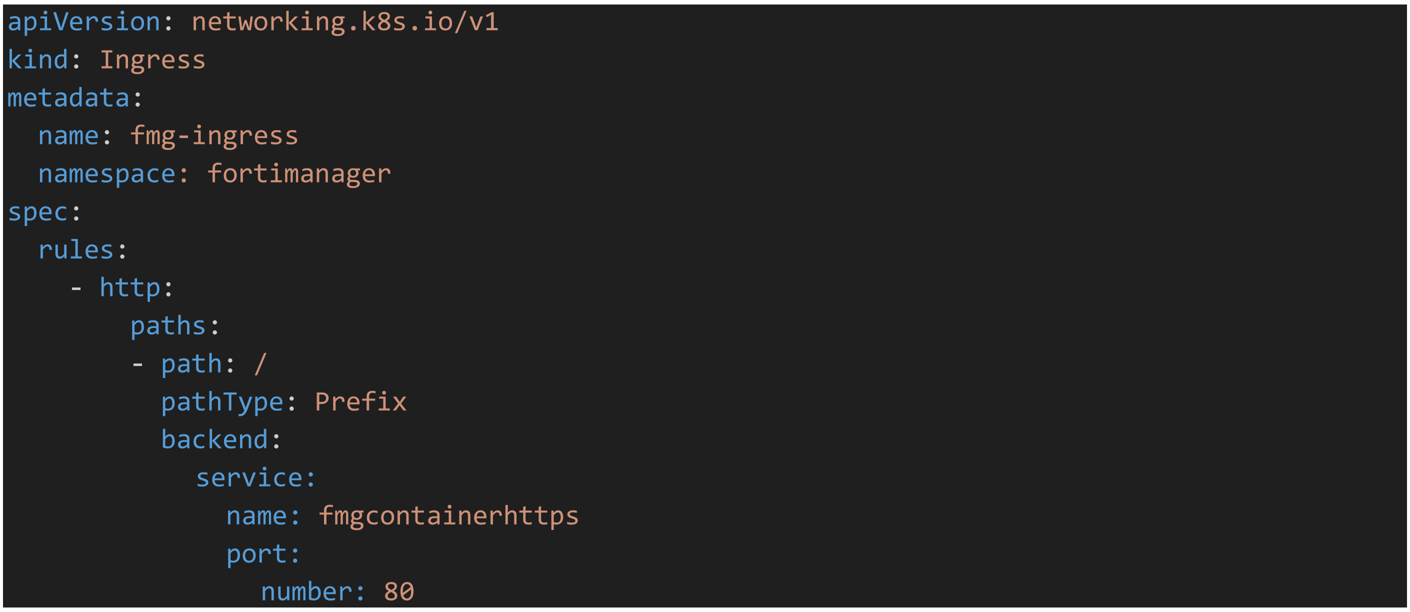
    Let us begin by enabling the minikube add-on on ingress and then we will proceed to add or upload a new yaml file (fmg-ingress.yaml).

    minikube addons enable ingress



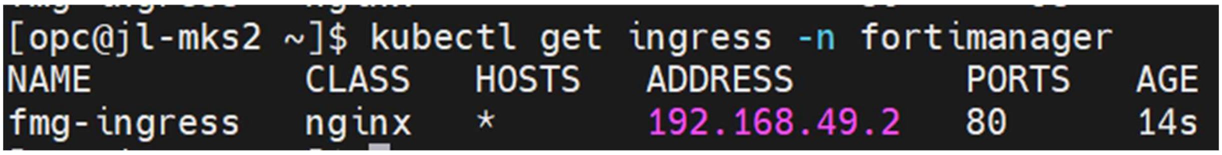
```
[opc@jl-mks2 ~]$ minikube addons enable ingress
* ingress is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
  - Using image registry.k8s.io/ingress-nginx/controller:v1.8.1
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20230407
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20230407
* Verifying ingress addon...
* The 'ingress' addon is enabled
```

Apply the below ingress script, **kubectl apply -f fmg-ingress.yaml**



```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: fmg-ingress
  namespace: fortimanager
spec:
  rules:
    - http:
        paths:
        - path: /
          pathType: Prefix
          backend:
            service:
              name: fmgcontainerhttps
              port:
                number: 80
```

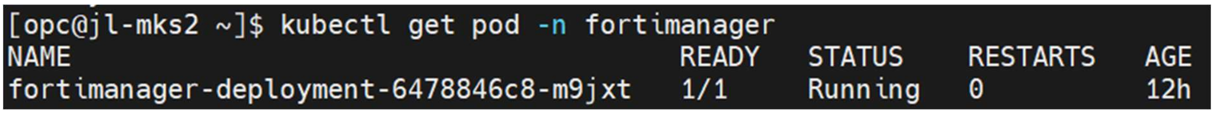
Verify that the ingress is created successfully, **kubectl get ingress -n fortimanager**



```
[opc@jl-mks2 ~]$ kubectl get ingress -n fortimanager
NAME          CLASS   HOSTS   ADDRESS        PORTS   AGE
fmg-ingress   nginx   *       192.168.49.2   80      14s
```

15. Disable the http to https redirect in FortiManager. How do we do that? We need to find out what's our POD name and run a command to CLI into FortiManager command shell.

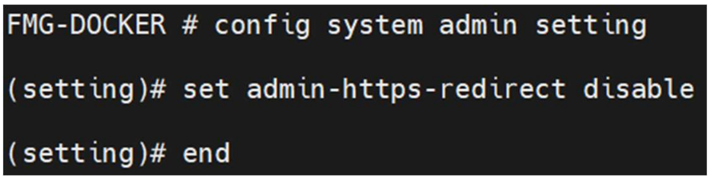
    Kubectl get pod -n fortimanager



```
[opc@jl-mks2 ~]$ kubectl get pod -n fortimanager
NAME                                      READY   STATUS    RESTARTS   AGE
fortimanager-deployment-6478846c8-m9jxt   1/1     Running   0          12h
```

Next, let us remote into the fortimanager and configure accordingly

kubectl exec -it -n fortimanager fortimanager-deployment-6478846c8-m9jxt – cli

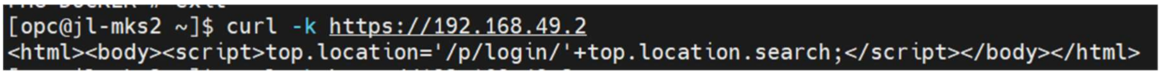


```
FMG-DOCKER # config system admin setting

(setting)# set admin-https-redirect disable

(setting)# end
```

16. Let's try and see if you are able to view the GET
    curl -k https://192.168.49.2



```
[opc@jl-mks2 ~]$ curl -k https://192.168.49.2
<html><body><script>top.location='/p/login/'+top.location.search;</script></body></html>
```

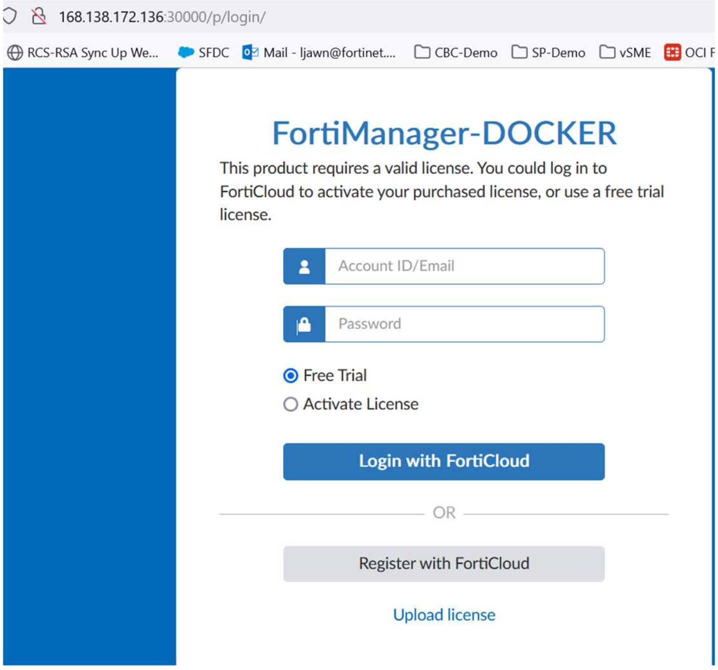
17. As 192.168.49.2 is my minikube internal IP, let's try to access from my public IP. To do that, ensure you have open the ports you would like to listen to and create a port forwarding to direct all traffic to the listening port and eventually landing to the fortimanager port.

    sudo firewall-cmd --permanent --add-port=30000/tcp

    sudo firewall-cmd –reload

    kubectl port-forward -n fortimanager --address 0.0.0.0 svc/fmgcontainerhttps 30000:80

    Now, let's try to access the fortimanager with port 30000.

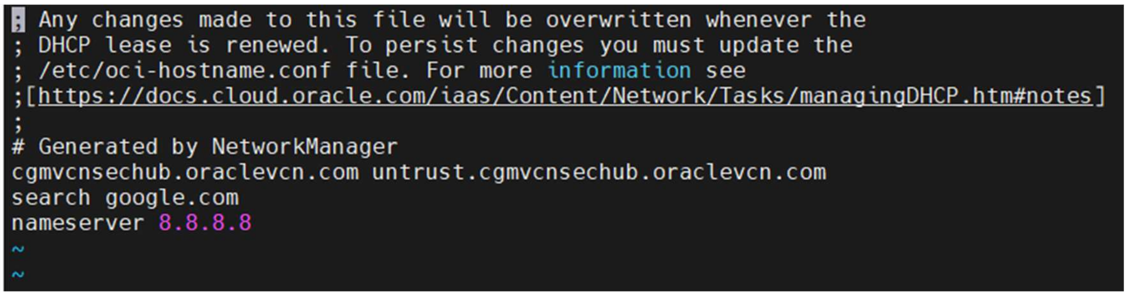


FortiManager-DOCKER

This product requires a valid license. You could log in to FortiCloud to activate your purchased license, or use a free trial license.

18. Let's load the Flex license. Before that, you can also change the nameserver of the Linux to 8.8.8.8 so the FGT can reach to internet service.

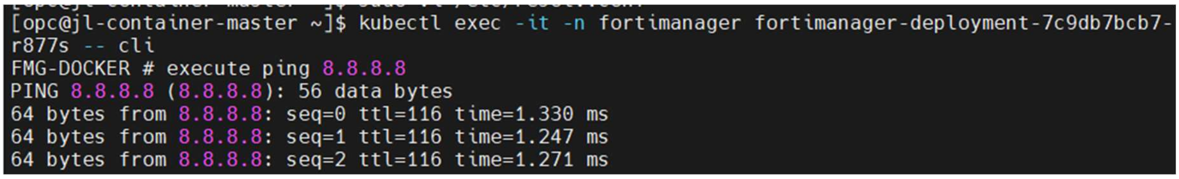
    sudo vi /etc/resolv.conf



```
; Any changes made to this file will be overwritten whenever the
; DHCP lease is renewed. To persist changes you must update the
; /etc/oci-hostname.conf file. For more information see
;[https://docs.cloud.oracle.com/iaas/Content/Network/Tasks/managingDHCP.htm#notes]
;
# Generated by NetworkManager
cgmvcnsechub.oraclevcn.com untrust.cgmvcnsechub.oraclevcn.com
search google.com
nameserver 8.8.8.8
~
~
```

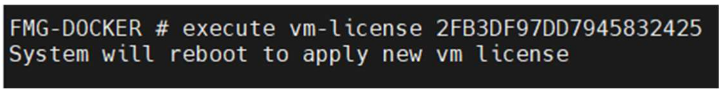
Shell into your FortiManager and try to ping.

Kubectl exec -it -n fortimanager <your pod name, can get from kubectl get pod> **-- cli**



```
[opc@jl-container-master ~]$ kubectl exec -it -n fortimanager fortimanager-deployment-7c9db7bcb7-
r877s -- cli
FMG-DOCKER # execute ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=116 time=1.330 ms
64 bytes from 8.8.8.8: seq=1 ttl=116 time=1.247 ms
64 bytes from 8.8.8.8: seq=2 ttl=116 time=1.271 ms
```

execute vm-license <flex-token>



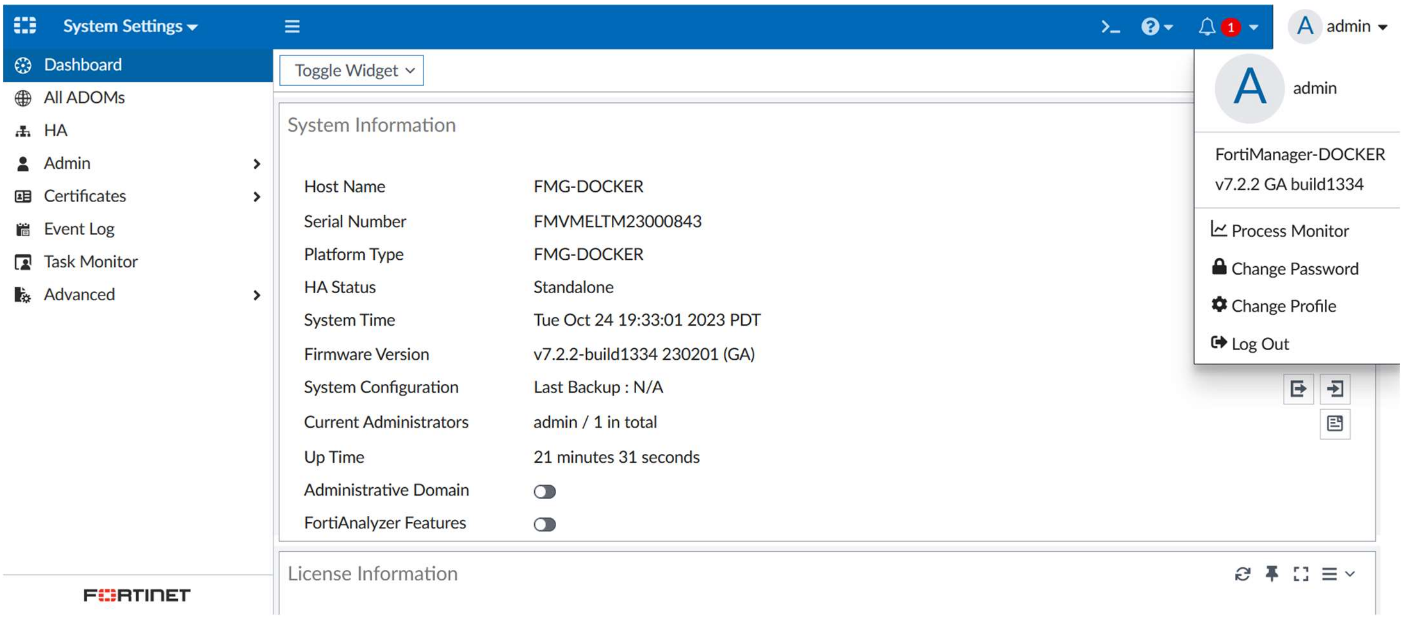
```
FMG-DOCKER # execute vm-license 2FB3DF97DD7945832425
System will reboot to apply new vm license
```

19. Run the **kubectl port-forward -n fortimanager --address 0.0.0.0 svc/fmgcontainerhttps 30000:80** again.

    Congratulations!! You have successfully completed this module. (FMG and FAZ - same concept)

    *Currently using 7.2.2 as there is some glitch in newer version (requirepass went missing right after rebooting FMG/FAZ in /etc/redis.conf

FMG-DOCKER PAGE



admin

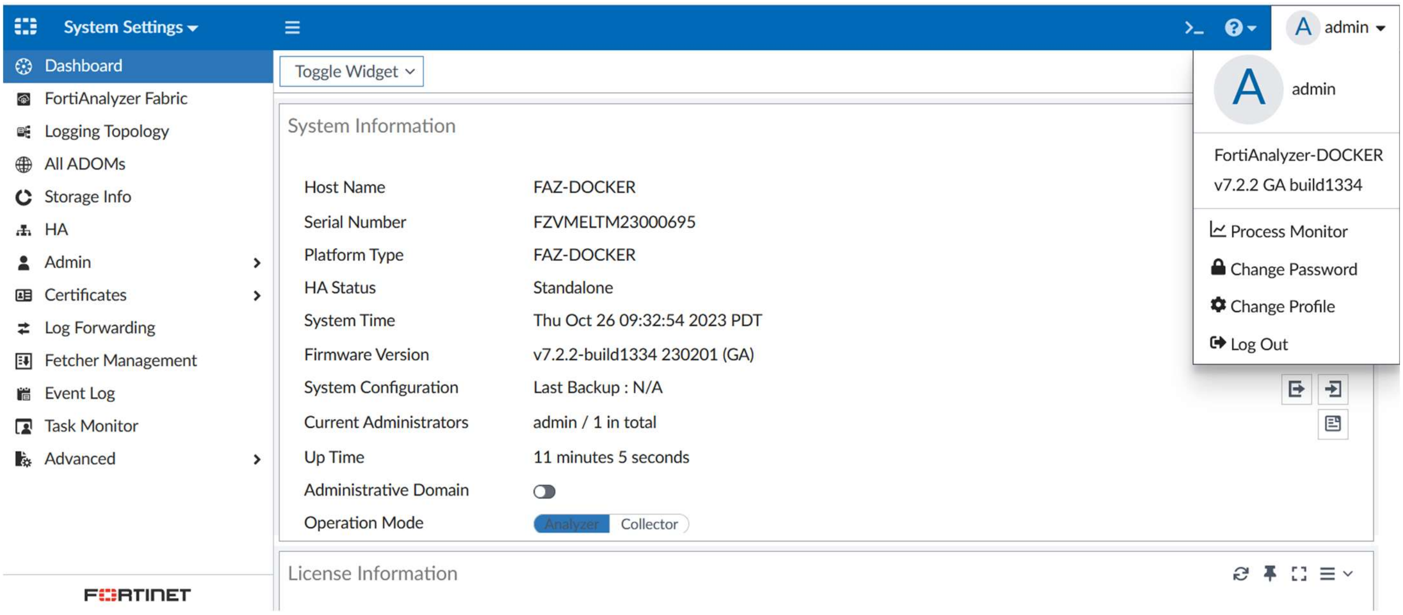
FortiManager-DOCKER
v7.2.2 GA build1334

System Information

Host Name	FMG-DOCKER
Serial Number	FMVMELTM23000843
Platform Type	FMG-DOCKER
HA Status	Standalone
System Time	Tue Oct 24 19:33:01 2023 PDT
Firmware Version	v7.2.2-build1334 230201 (GA)
System Configuration	Last Backup : N/A
Current Administrators	admin / 1 in total
Up Time	21 minutes 31 seconds
Administrative Domain	○
FortiAnalyzer Features	○

License Information

FAZ-DOCKER PAGE



admin

FortiAnalyzer-DOCKER
v7.2.2 GA build1334

System Information

Host Name	FAZ-DOCKER
Serial Number	FZVMELTM23000695
Platform Type	FAZ-DOCKER
HA Status	Standalone
System Time	Thu Oct 26 09:32:54 2023 PDT
Firmware Version	v7.2.2-build1334 230201 (GA)
System Configuration	Last Backup : N/A
Current Administrators	admin / 1 in total
Up Time	11 minutes 5 seconds
Administrative Domain	○
Operation Mode	[✓ Analyzer] Collector

License Information

-----------------------END---------------------------