EE379K Enterprise Network Security Lab 2 Report

Student: Sean Wang, szw87
Professor: Mohit Tiwari, Antonio Espinoza
Department of Electrical & Computer Engineering
The University of Texas at Austin

October 2, 2019

Part 3 - Orchestration

3a - Orchestration with Kubernetes

Docker applications

Running the simple PHP and MySQL server example with

```
$ docker-compose up
```

sets up both the web-service and SQL DB are on localhost. The web-service can be accessed through `localhost:8000`, which maps internally to port 80 inside the container. Additionally, the web-service uses port 3306 to access the SQL DB, while port 8082 is exposed to the host. The contents seen on the homepage, `http://localhost:8000`, is due to `src/index.php`.
By going into `docker-compose.yml` and changing the port mapping from `8000:80` to `9000:80`, as shown in Figure 1, the web-server can now be accessed at `http://localhost:9000`, since this changed what port is exposed to the host machine and maps it to the internal port.

Figure 1: Changing port mapping inside docker-compose.yml

Kubernetes

After tagging and pushing the web-service image to the microk8s registry, then the following commands are used to run the web-application in kubernetes:

```
$ microk8s.kubectl apply -f webserver.yaml
$ microk8s.kubectl apply -f webserver-svc.yaml
$ microk8s.kubectl apply -f mysql.yaml
$ microk8s.kubectl apply -f mysql-svc.yaml
```

Then, the different namespaces, shown in Figure 2 and Figure 3, are seen under the `NAMESPACE` column in each of the outputs of the following commands:

```
$ microk8s.kubectl get pods --all-namespaces
$ microk8s.kubectl get services --all-namespaces
```

For example, the `default` namespace refers to the default namespace for objects without any specified namespace. Additionally, Kubernetes creates the `kube-system` namespace, and it includes pods and services like the dashboard. [1]

```
class@class-VirtualBox:~/simplePhpSQL_k8s$ microk8s.kubectl get pods --all-namespaces
NAMESPACE            NAME                                        READY   STATUS    RESTARTS   AGE
container-registry   registry-d7d7c8bc9-xg7wk                    1/1     Running   0          4h34m
default              mysql-5c6d57fc45-zw6fk                      1/1     Running   0          4h30m
default              webserver-77b46fb75f-j7pwh                  1/1     Running   0          4h31m
default              webserver-77b46fb75f-snj6h                  1/1     Running   0          4h31m
default              webserver-77b46fb75f-v8fbm                  1/1     Running   0          4h31m
kube-system          coredns-9b8997588-c2nq4                     1/1     Running   0          4h34m
kube-system          dashboard-metrics-scraper-566cddb686-cvt7r  1/1     Running   0          4h33m
kube-system          heapster-v1.5.2-5c58f64f8b-9vqgv            4/4     Running   1          4h33m
kube-system          hostpath-provisioner-7b9cb5cdb4-2fhr9       1/1     Running   0          4h34m
kube-system          kubernetes-dashboard-678b7d865c-t8qfm       1/1     Running   0          4h33m
kube-system          monitoring-influxdb-grafana-v4-6d599df6bf-pnc4r   2/2     Running   0          4h33m
```

Figure 2: Output of microk8s.kubectl get pods –all-namespaces

```
class@class-VirtualBox:~/simplePhpSQL_k8s$ microk8s.kubectl get services --all-namespaces
NAMESPACE            NAME                        TYPE          CLUSTER-IP       EXTERNAL-IP   PORT(S)                  AGE
container-registry   registry                    NodePort      10.152.183.253   <none>        5000:32000/TCP           4h34m
default              kubernetes                  ClusterIP     10.152.183.1     <none>        443/TCP                  4h39m
default              mysql8-service              NodePort      10.152.183.195   <none>        3306:30765/TCP           4h30m
default              web-service                 LoadBalancer  10.152.183.210   <pending>     80:31670/TCP             4h31m
kube-system          dashboard-metrics-scraper   ClusterIP     10.152.183.250   <none>        8000/TCP                 4h33m
kube-system          heapster                    ClusterIP     10.152.183.228   <none>        80/TCP                   4h33m
kube-system          kube-dns                    ClusterIP     10.152.183.10    <none>        53/UDP,53/TCP,9153/TCP   4h34m
kube-system          kubernetes-dashboard        NodePort      10.152.183.68    <none>        443:32388/TCP            4h33m
kube-system          monitoring-grafana          ClusterIP     10.152.183.139   <none>        80/TCP                   4h33m
kube-system          monitoring-influxdb         ClusterIP     10.152.183.52    <none>        8083/TCP,8086/TCP        4h33m
```

Figure 3: Output of microk8s.kubectl get services –all-namespaces

In the `webserver.yaml` file, there are specifications on how many instances of each application to deploy under `spec/replicas`:

```
apiVersion: apps/v1
kind: Deployment
...
spec:
  replicas: 3
...
```

This value can be changed to change the number of instances of web-servers. For example, if it was changed to 2, then the output of the `microk8s.kubectl get` commands would be the following:

```
class@class-VirtualBox:~/simplePhpSQL_k8s$ microk8s.kubectl get pods --all-namespaces
NAMESPACE            NAME                                        READY   STATUS    RESTARTS   AGE
container-registry   registry-d7d7c8bc9-xg7wk                    1/1     Running   0          4h35m
default              mysql-5c6d57fc45-zw6fk                      1/1     Running   0          4h32m
default              webserver-77b46fb75f-j7pwh                  1/1     Running   0          4h33m
default              webserver-77b46fb75f-v8fbm                  1/1     Running   0          4h33m
kube-system          coredns-9b8997588-c2nq4                     1/1     Running   0          4h35m
kube-system          dashboard-metrics-scraper-566cddb686-cvt7r  1/1     Running   0          4h34m
kube-system          heapster-v1.5.2-5c58f64f8b-9vqgv            4/4     Running   1          4h34m
kube-system          hostpath-provisioner-7b9cb5cdb4-2fhr9       1/1     Running   0          4h35m
kube-system          kubernetes-dashboard-678b7d865c-t8qfm       1/1     Running   0          4h34m
kube-system          monitoring-influxdb-grafana-v4-6d599df6bf-pnc4r   2/2     Running   0          4h34m
```

Figure 4: New output of microk8s.kubectl get pods –all-namespaces

```
class@class-VirtualBox:~/simplePhpSQL_k8s$ microk8s.kubectl get services --all-namespaces
NAMESPACE            NAME                        TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)                  AGE
container-registry   registry                    NodePort       10.152.183.253   <none>        5000:32000/TCP           4h35m
default              kubernetes                  ClusterIP      10.152.183.1     <none>        443/TCP                  4h41m
default              mysql8-service              NodePort       10.152.183.195   <none>        3306:30765/TCP           4h32m
default              web-service                 LoadBalancer   10.152.183.210   <pending>     80:31670/TCP             4h32m
kube-system          dashboard-metrics-scraper   ClusterIP      10.152.183.250   <none>        8000/TCP                 4h34m
kube-system          heapster                    ClusterIP      10.152.183.228   <none>        80/TCP                   4h34m
kube-system          kube-dns                    ClusterIP      10.152.183.10    <none>        53/UDP,53/TCP,9153/TCP   4h35m
kube-system          kubernetes-dashboard        NodePort       10.152.183.68    <none>        443:32388/TCP            4h34m
kube-system          monitoring-grafana          ClusterIP      10.152.183.139   <none>        80/TCP                   4h34m
kube-system          monitoring-influxdb         ClusterIP      10.152.183.52    <none>        8083/TCP,8086/TCP        4h34m
```

Figure 5: New output of microk8s.kubectl get services –all-namespaces

RBAC

For Role Based Access Control, first a service account and role need to be created and then bound together. Then, the following command can be used to set up and run the Kubernetes Dashboard:

```
$ microk8s.kubectl -n kube-system
    edit service kubernetes-dashboard
```

The type must be changed to `NodePort` and the exposed port is given under `ports/nodePort`:

```
...
spec:
  clusterIP: 10.152.183.68
  ports:
  - nodePort: 32388 # port num
    port: 443
    protocol: TCP
    targetPort: 8443
  selector:
    k8s-app: kubernetes-dashboard
  sessionAffinity: None
  type: NodePort # change from ClusterIP to NodePort
...
```

Then, once a secret token is obtained, the dashboard can be opened and a list of all the pods in the default namespace can be seen, like in Figure 6. Only these pods are shown because the namespace of the user-sa account is set to default, which is specified in the `sa-role-bind.yaml` file:

```
...
subjects:
- kind: ServiceAccount
```

```
    name: user-sa
    namespace: default
...
```

In order to create another service account that can access just the kube-system namespace, a new service account must be initialized. This can be done by first creating a service account and then making slight modifications to the `user-role.yaml` and `sa-role-bind.yaml` files, as can be seen in `part3/kube-role.yaml` and `part3/kube-sa-role-bind.yaml`, respectively. After creating the new service account, login to the dashboard with the token for the new account and now nothing can be seen in the `default` namespace, but the pods in the `kube-system` namespace are now visible on the dashboard, as shown in Figure 7. The sequence of commands to set this up are as follows:

```
$ microk8s.kubectl create serviceaccount kube-sa --namespace kube-system
$ microk8s.kubectl apply -f kube-role.yaml
$ microk8s.kubectl apply -f kube-sa-role-bind.yaml
```

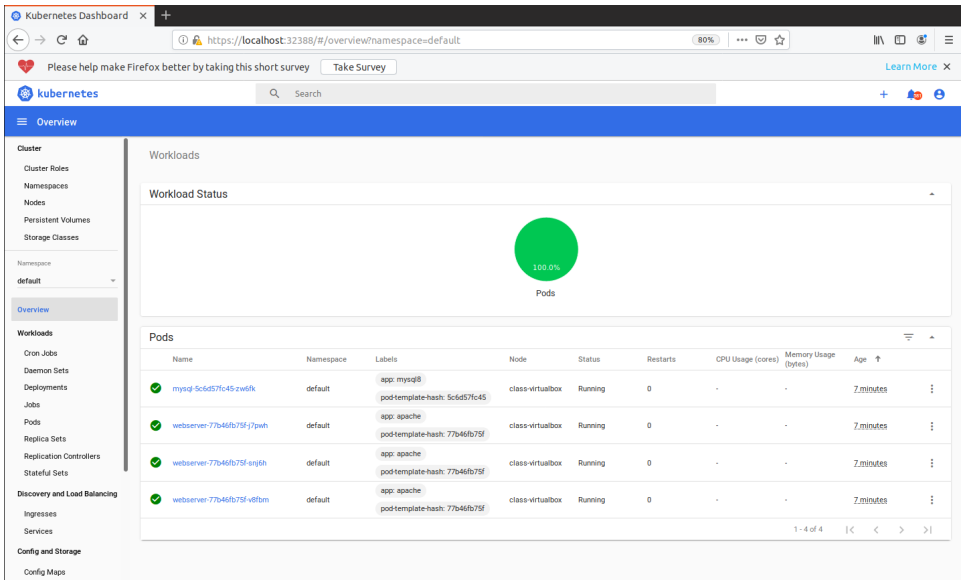

Figure 6: Dashboard view of default namespace visible to user-sa

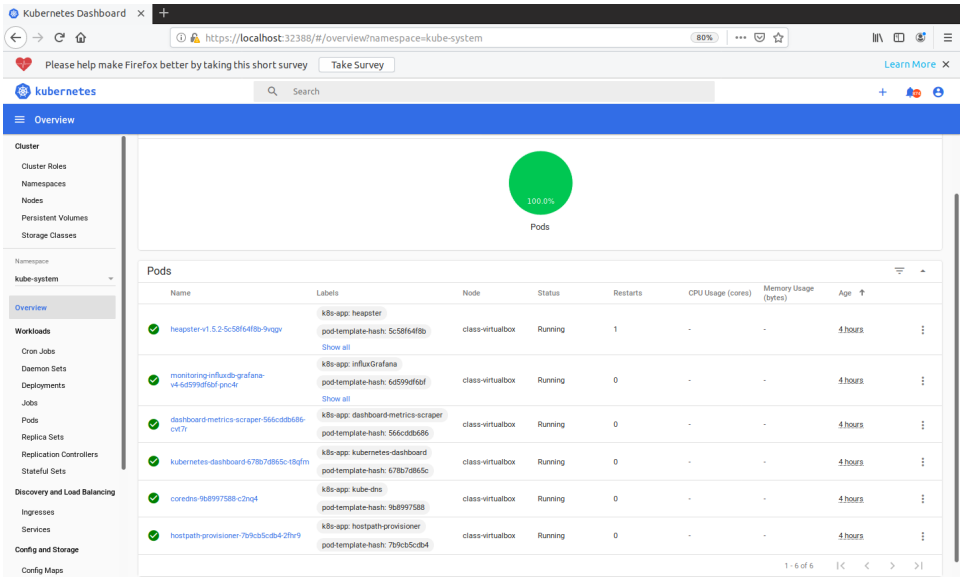

Figure 7: Dashboard view of kube-system namespace visible to kube-sa

References

[1] "Kubernetes Concepts - Namespaces," September 2019.