Kubernetes lab

Authors: Olivia Manz and Samuel Roland in **Group D**

The whole project is available on this Github repository: https://github.com/samuelroland/CloudSys-labs/tree/main/lab5

Introduction

We have just done the Kubernetes exercise where the forecasting was deployed to a “Kind” cluster.

For the lab, you will repeat the Kubernetes exercise but deploy all the services to a k3s cluster instead of a “Kind” Cluster.

We will reuse the files from the previous exercise and deploy them on AWS.

Create AWS instances

Sign in at AWS EC2 Console and create 4 instances:

- 1x: 2 CPU, 4GB RAM instance, 50GB, named `GroupD-control-pane`
- 3x: 1CPU, 2GB RAM at most, 30GB, named `control-worker-1`, `control-worker-2`, `control-worker-3`

A security group `launch-wizard-114` is automatically created. We will use the same key pair `Gd-control-plane.pem` for all instances.

To allow k3s to communicate between the master and worker nodes during installation, add a rule in the security group to allow TCP on port `6443`. We can prepare the futur by adding:

- `UDP 8472` to allow the forecast service to push data to Redis (node-to-node communication)
- `TCP 32000` to allow access to the Grafana dashboard from outside the cluster. To minimize exposure, restrict the source to the security group itself for `UDP 8472` and `TCP 6443`.

Install k3s on control-pane

Connect to the `GroupD-control-plane` instance via SSH. The IP address (or SSH command) is available in the AWS console:

```
ssh -i "Gd-control-plane.pem" ec2-user@ec2-54-204-109-236.compute-1.amazonaws.com
```

Install k3s:

```
curl -sfL https://get.k3s.io | sh -
```

Verify the installation

```
$ sudo k3s kubectl get nodes
NAME                          STATUS   ROLES                  AGE   VERSION
ip-172-31-29-6.ec2.internal   Ready    control-plane,master   70s   v1.33.5+k3s1
```

Display the node token for adding workers later:

```
sudo cat /var/lib/rancher/k3s/server/node-token
```

Install k3s on workers

Connect to each worker via SSH and install k3s in agent mode, using the control plane IP and token:

```
curl -sfL https://get.k3s.io | K3S_URL=https://<IP-control-plane>:6443 K3S_TOKEN=<token-node> sh -
```

Repeat for all workers. On the control plane, verify the nodes:

```
$ sudo k3s kubectl get nodes -o wide
NAME                           STATUS   ROLES                  AGE     VERSION        INTERNAL-IP     EXTERNAL-IP   OS-
IMAGE                       KERNEL-VERSION                   CONTAINER-RUNTIME
ip-172-31-23-123.ec2.internal   Ready    <none>                 3m17s   v1.33.5+k3s1   172.31.23.123   <none>
Amazon Linux 2023.9.20251027   6.1.156-177.286.amzn2023.x86_64   containerd://2.1.4-k3s1
ip-172-31-24-250.ec2.internal   Ready    <none>                 2m22s   v1.33.5+k3s1   172.31.24.250   <none>
Amazon Linux 2023.9.20251027   6.1.156-177.286.amzn2023.x86_64   containerd://2.1.4-k3s1
ip-172-31-27-254.ec2.internal   Ready    <none>                 2m46s   v1.33.5+k3s1   172.31.27.254   <none>
Amazon Linux 2023.9.20251027   6.1.156-177.286.amzn2023.x86_64   containerd://2.1.4-k3s1
ip-172-31-29-6.ec2.internal     Ready    control-plane,master   7m57s   v1.33.5+k3s1   172.31.29.6     <none>
Amazon Linux 2023.9.20251027   6.1.156-177.286.amzn2023.x86_64   containerd://2.1.4-k3s1
```

Prepare nodes

As we should install grafana, redis and data-retrieval on the master and forecast on the workers, we label each node:

```
sudo k3s kubectl label node ip-172-31-29-6.ec2.internal role=control-plane
sudo k3s kubectl label node ip-172-31-23-123.ec2.internal role=worker
sudo k3s kubectl label node ip-172-31-24-250.ec2.internal role=worker
sudo k3s kubectl label node ip-172-31-27-254.ec2.internal role=worker
```

We can check the result:

```
$ sudo k3s kubectl get nodes --show-labels
NAME                           STATUS   ROLES                  AGE     VERSION        LABELS
ip-172-31-23-123.ec2.internal   Ready    <none>                 3m28s   v1.33.5+k3s1   beta.kubernetes.io/
arch=amd64,beta.kubernetes.io/instance-type=k3s,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/
hostname=ip-172-31-23-123.ec2.internal,kubernetes.io/os=linux,node.kubernetes.io/instance-type=k3s,role=worker
ip-172-31-24-250.ec2.internal   Ready    <none>                 2m33s   v1.33.5+k3s1   beta.kubernetes.io/
arch=amd64,beta.kubernetes.io/instance-type=k3s,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/
hostname=ip-172-31-24-250.ec2.internal,kubernetes.io/os=linux,node.kubernetes.io/instance-type=k3s,role=worker
ip-172-31-27-254.ec2.internal   Ready    <none>                 2m57s   v1.33.5+k3s1   beta.kubernetes.io/
arch=amd64,beta.kubernetes.io/instance-type=k3s,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/
hostname=ip-172-31-27-254.ec2.internal,kubernetes.io/os=linux,node.kubernetes.io/instance-type=k3s,role=worker
ip-172-31-29-6.ec2.internal     Ready    control-plane,master   8m8s v1.33.5+k3s1   beta.kubernetes.io/
arch=amd64,beta.kubernetes.io/instance-type=k3s,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/
hostname=ip-172-31-29-6.ec2.internal,kubernetes.io/os=linux,node-role.kubernetes.io/control-plane=true,node-
role.kubernetes.io/master=true,node.kubernetes.io/instance-type=k3s,role=control-plane
```

As asked in the instructions, we want to have all the pods on the control-plane node and except for the Forecast pods on the worker nodes. To achieve this, we changed the deployment yaml files and to add a `nodeSelector` that makes it possible to select the node by role.

```
spec:
  template:
    spec:
      nodeSelector:
        role: control-plane
```

For `forecast-deployment.yaml`, we did the same thing with `role: worker`.

For Grafana, we changed the service type to `NodePort` to allow external access:

```
apiVersion: v1
```

```yaml
kind: Service
metadata:
  name: grafana-service
spec:
  type: NodePort
  ports:
  - port: 3000
    targetPort: 3000
    nodePort: 32000   # port available from outside
  selector:
    app: grafana-pod
```

Deploying the Services

We decide to copy deployment files on control-plane node and run these commands locally to upload the folder `deployment` under `/home/ec2-user/`.

```
scp -i Gd-control-plane.pem -r deployment ec2-user@54.204.109.236:/home/ec2-user/
```

On the control-plane node, we deployed Redis:

```
sudo k3s kubectl apply -f /home/ec2-user/deployment/redis-deployment.yaml
```

Like before, we created a secret for AWS credentials

```
kubectl create secret generic <name-of-the-secrets> \
  --from-literal=AWS_ACCESS_KEY_ID=<Your access key> \
  --from-literal=AWS_SECRET_ACCESS_KEY=<your secret key>
```

We then deployed Data-Retrieval:

```
sudo k3s kubectl apply -f /home/ec2-user/deployment/data-retrieval-deployment.yaml
```

We check that data-retrieval has finished its execution before deploying the forecast module.

```
sudo k3s kubectl apply -f /home/ec2-user/deployment/forecast-deployment.yaml
```

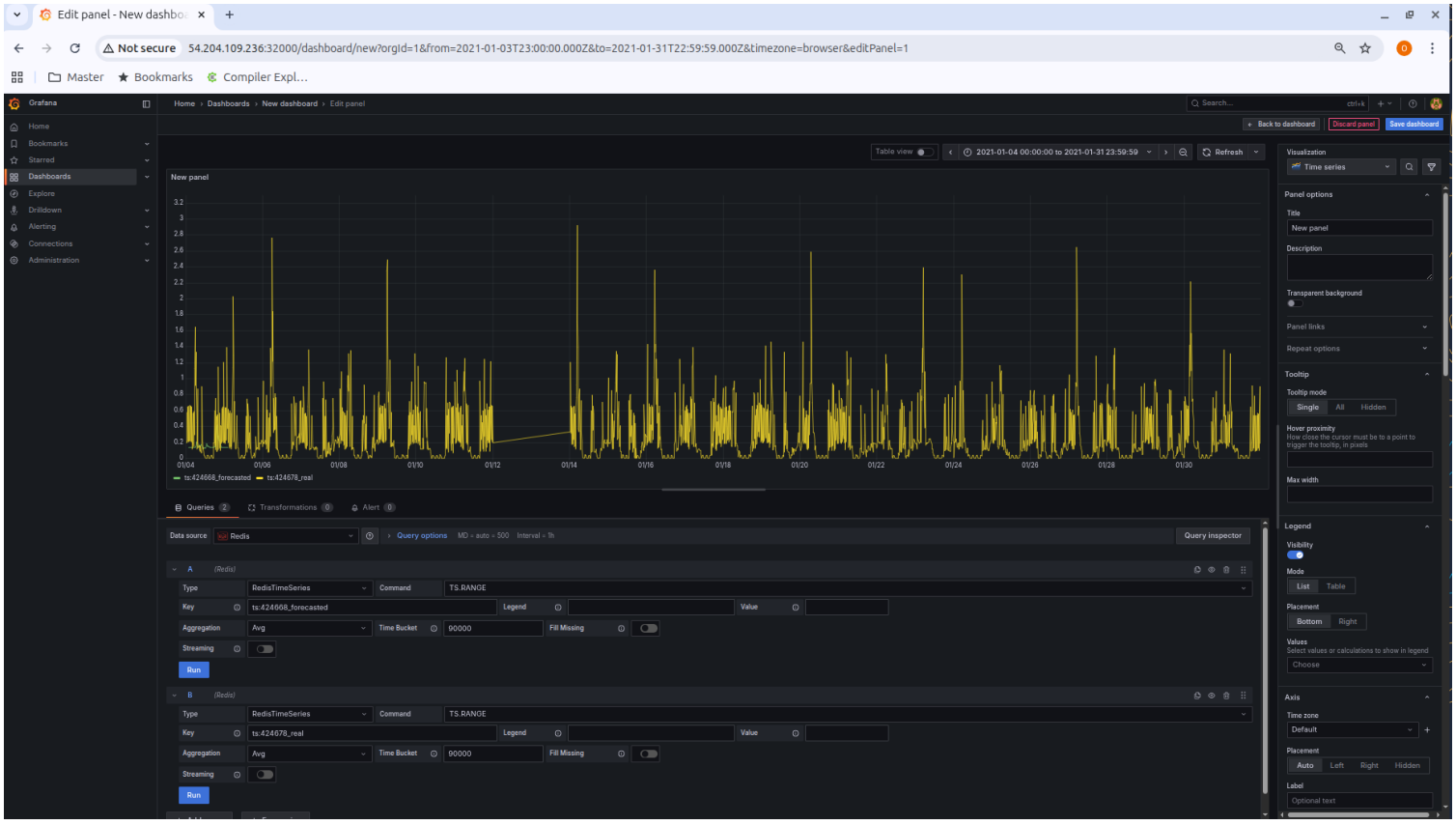
Finally, we deployed Grafana:

```
sudo k3s kubectl apply -f /home/ec2-user/deployment/grafana-deployment.yaml
```

We verified the status of all pods and here is the full output of `sudo kubectl get all -o wide` in screenshot.

```
ec2-user@ip-172-31-29-6:~   ec2-user@ip-172-31-23-123:~   ec2-user@ip-172-31-27-254:~   ec2-user@ip-172-31-24-250:~   ~/D/M/A/2/T/g/C/lab5   ~/D/M/A/2/T/g/C/lab5

[ec2-user@ip-172-31-29-6 ~]$
[ec2-user@ip-172-31-29-6 ~]$
[ec2-user@ip-172-31-29-6 ~]$
[ec2-user@ip-172-31-29-6 ~]$
[ec2-user@ip-172-31-29-6 ~]$
[ec2-user@ip-172-31-29-6 ~]$
[ec2-user@ip-172-31-29-6 ~]$
[ec2-user@ip-172-31-29-6 ~]$
[ec2-user@ip-172-31-29-6 ~]$ sudo kubectl get all -o wide
NAME                                        READY   STATUS      RESTARTS   AGE   IP           NODE                          NOMINATED NODE   READINESS GATES
pod/busybox-pod                             1/1     Running     0          33m   10.42.0.11   ip-172-31-29-6.ec2.internal   <none>           <none>
pod/data-retrieval-job-klhc6                0/1     Completed   0          62m   10.42.0.10   ip-172-31-29-6.ec2.internal   <none>           <none>
pod/forecast-deployment-66fb88d988-466tk    1/1     Running     0          19m   10.42.1.7    ip-172-31-23-123.ec2.internal <none>           <none>
pod/forecast-deployment-66fb88d988-5jhnp    1/1     Running     0          19m   10.42.3.9    ip-172-31-24-250.ec2.internal <none>           <none>
pod/forecast-deployment-66fb88d988-6xtdq    1/1     Running     0          19m   10.42.2.8    ip-172-31-27-254.ec2.internal <none>           <none>
pod/forecast-deployment-66fb88d988-h6zsp    1/1     Running     0          19m   10.42.3.10   ip-172-31-24-250.ec2.internal <none>           <none>
pod/forecast-deployment-66fb88d988-hx49r    1/1     Running     0          19m   10.42.2.9    ip-172-31-27-254.ec2.internal <none>           <none>
pod/forecast-deployment-66fb88d988-psbjg    1/1     Running     0          19m   10.42.1.8    ip-172-31-23-123.ec2.internal <none>           <none>
pod/forecast-deployment-66fb88d988-wl6tg    1/1     Running     0          19m   10.42.3.8    ip-172-31-24-250.ec2.internal <none>           <none>
pod/grafana-deployment-68994679dd-b4dnz     1/1     Running     0          18m   10.42.0.14   ip-172-31-29-6.ec2.internal   <none>           <none>
pod/redis-deployment-76df9f5d64-62w2s       1/1     Running     0          63m   10.42.0.9    ip-172-31-29-6.ec2.internal   <none>           <none>

NAME                 TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE   SELECTOR
service/grafana-service   NodePort    10.43.8.37    <none>        3000:32000/TCP   18m   app=grafana-pod
service/kubernetes        ClusterIP   10.43.0.1     <none>        443/TCP          73m   <none>
service/redis             ClusterIP   10.43.210.8   <none>        6379/TCP         63m   app=redis-pod

NAME                                   READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS   IMAGES                                SELECTOR
deployment.apps/forecast-deployment   7/7     7            7           61m   forecast     omanzmaster/forecast:latest           app=forecast-pod
deployment.apps/grafana-deployment    1/1     1            1           18m   grafana      omanzmaster/my-grafana:latest         app=grafana-pod
deployment.apps/redis-deployment      1/1     1            1           63m   redis        redislabs/redistimeseries:latest      app=redis-pod

NAME                                              DESIRED   CURRENT   READY   AGE   CONTAINERS   IMAGES                                SELECTOR
replicaset.apps/forecast-deployment-66fb88d988   7         7         7       19m   forecast     omanzmaster/forecast:latest           app=forecast-pod,pod-template-hash=66fb88d988
replicaset.apps/forecast-deployment-6876545675   0         0         0       39m   forecast     omanzmaster/forecast:latest           app=forecast-pod,pod-template-hash=6876545675
replicaset.apps/forecast-deployment-cc4595c47    0         0         0       61m   forecast     omanzmaster/forecast:latest           app=forecast-pod,pod-template-hash=cc4595c47
replicaset.apps/grafana-deployment-68994679dd    1         1         1       18m   grafana      omanzmaster/my-grafana:latest         app=grafana-pod,pod-template-hash=68994679dd
replicaset.apps/redis-deployment-76df9f5d64      1         1         1       63m   redis        redislabs/redistimeseries:latest      app=redis-pod,pod-template-hash=76df9f5d64

NAME                           STATUS     COMPLETIONS   DURATION   AGE   CONTAINERS        IMAGES                                SELECTOR
job.batch/data-retrieval-job   Complete   1/1           25s        62m   data-retrieval    omanzmaster/data-retrieval:latest     batch.kubernetes.io/controller-uid=35ea212b-f88e-40c9-b5d7-a4a99e67397a
[ec2-user@ip-172-31-29-6 ~]$
```

We accessed Grafana through the public IP of the control-plane node on port 32000. The following screenshot shows the result after the dashboard configuration.



Difficulty

The main difficulty during this lab was knowing whether a rule was necessary to add to the security group. Particularly for the rule allowing the Forecast service to send data to Redis.