Lab ORCH – Kubernetes

Introduction and Prerequisites

In this exercise, you will look at the anatomy of a Kubernetes Cluster and explore the services and functionalities of Kubernetes.

Therefore, we will be using k3d as an orchestrator to spin up lightweight Kubernetes distributions based on k3s. It is using Docker in Docker technology (i.e., running containers within containers) to spin up virtual Kubernetes node containers (similar to VMs or physical servers) and start the required services and applications/pods within these node containers.

The objective of this lab session is to:

- Investigate the anatomy of and Kubernetes cluster.
- Explore the resources and services running in a standard Kubernetes cluster.
- Investigate the network configuration of a cluster and the flow of traffic from the internet to the application running in the pods.
- Verify application resilience provided by replica sets.

Required Resources and Tools

We will use the Lab environment set up in the ARCH Lab. If you do not have set up the environment and tooling, please follow the lab setup instructions of the first lab.

You need to be able to connect to the lab cluster running on the VM (or on your local computer) using `kubectl` (local on the VM or from your computer).

In this lab, no additional resources are required. We will be working with the same resources and configurations provided in the Lab repository on `https://github.zhaw.ch/CCP2/CCP2-ARCH-Lab`.

References

- k3d – run k3s cluster in docker: https://k3d.io/
- Arkade – utility for managing K8s CLI tools: https://github.com/alexellis/arkade
- kubectl – cheat sheet: https://kubernetes.io/docs/reference/kubectl/cheatsheet/
- Docker CLI Reference: https://docs.docker.com/reference/cli/docker/

Task 1 - Inspect the anatomy of the K3D cluster

In the lecture, the anatomy of a Kubernetes cluster was explained. In the first task, you will explore your lab cluster and compare it with the theory.

First, we need to verify your cluster is operational:

1. Connect to your VM using SSH and check the status of your cluster using
   `k3d cluster list`
   If the returned list is empty, you should initialize it using
   `k3d cluster create -c k3d-ccp2-config.yaml`
   If it shows the not all nodes operational, e.g,
   ```
   NAME    SERVERS    AGENTS    LOADBALANCER
   ccp2    0/1        0/2       true
   ```
   then you should start your cluster using `k3d cluster start ccp2`

2. Verify the node status using **`k3d node list`** (STATUS should be running)
    a. What nodes are running? Does it match the architecture in the slides?
    b. What is the function of the different nodes?
3. Compare the k3d nodes with the running docker containers.
   Hint: you can specify a specific format using the format argument, e.g.
   `docker ps --format 'table {{.ID}}\t{{.Image}}\t{{.Names}}\t{{.Ports}}'`
4. Make sure you can connect to the cluster: **`kubectl cluster-info`**
5. Also, kubectl can give you some more information about the nodes using:
   `kubectl get node -o wide`
   What info is shown? (Remember the shown IP addresses for later)

Now, let's have a look inside the cluster:

- By default, 4 namespaces are created.
  A namespace is used to separate projects or applications, for easier management. Also, it is possible to set access policies, to limit access to authorized users / roles.
  `kubectl get namespaces`

○ default	all resources without a specified namespace will be created here
○ kube-node-lease	used internally for monitoring of nodes
○ kube-public	may contain public objects (no authentication required)
○ kube-system	contains the kubernetes system services

  For us, the `default` and `kube-system` namespaces are interesting.

  We will use default namespace to deploy some test applications.

  kube-system shows the services required to run Kubernetes:
  `kubectl -n kube-system get all`

```
NAME                                           READY   STATUS      RESTARTS      AGE
pod/helm-install-traefik-crd-5nm9t             0/1     Completed   0             7d3h
pod/helm-install-traefik-76cnk                 0/1     Completed   1             7d3h
pod/svclb-traefik-683c3490-6gnwh               2/2     Running     4 (23m ago)   7d3h
pod/coredns-597584b69b-d8wgr                   1/1     Running     2 (23m ago)   7d3h
pod/svclb-traefik-683c3490-xr27n               2/2     Running     4 (23m ago)   7d3h
pod/metrics-server-5f9f776df5-kwtdb            1/1     Running     4 (23m ago)   7d3h
pod/traefik-66c46d954f-k4zzp                   1/1     Running     2 (23m ago)   7d3h
pod/svclb-traefik-683c3490-x7q9b               2/2     Running     4 (23m ago)   7d3h
pod/local-path-provisioner-79f67d76f8-475zb    1/1     Running     3 (23m ago)   7d3h

NAME                     TYPE           CLUSTER-IP     EXTERNAL-IP                        PORT(S)                      AGE
service/kube-dns         ClusterIP      10.43.0.10     <none>                             53/UDP,53/TCP,9153/TCP       7d3h
service/metrics-server   ClusterIP      10.43.135.35   <none>                             443/TCP                      7d3h
service/traefik          LoadBalancer   10.43.91.185   172.25.0.2,172.25.0.3,172.25.0.4 80:30966/TCP,443:32285/TCP   7d3h

NAME                                    DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/svclb-traefik-683c3490   3         3         3       3            3           <none>          7d3h

NAME                                     READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/traefik                  1/1     1            1           7d3h
deployment.apps/local-path-provisioner   1/1     1            1           7d3h
deployment.apps/coredns                  1/1     1            1           7d3h
deployment.apps/metrics-server           1/1     1            1           7d3h

NAME                                                DESIRED   CURRENT   READY   AGE
replicaset.apps/traefik-66c46d954f                  1         1         1       7d3h
replicaset.apps/local-path-provisioner-79f67d76f8   1         1         1       7d3h
replicaset.apps/coredns-597584b69b                  1         1         1       7d3h
replicaset.apps/metrics-server-5f9f776df5           1         1         1       7d3h

NAME                           COMPLETIONS   DURATION   AGE
job.batch/helm-install-traefik-crd   1/1           30s        7d3h
job.batch/helm-install-traefik       1/1           34s        7d3h
```

Besides the internal services kube-dns (responsible for internal dns resolution) and metrics-server (collecting metrics data), an ingress proxy server (traefik) is installed by k3d and made available via the external load-balancer node (k3d-proxy, remember the k3d docker containers).

You may also notice that a daemonset is maintaining service loadbalancer (svclb-traefik) pods on each node (1 server & 2 agents) for the ingress service.

The local-path-provisioner is a persistent volume service providing mount points to the local file system.

- What is a DaemonSet? What is it used for? Check the Kubernetes Documentation.
- What is the content of the default namespace? (assuming you completed the ARCH lab)

Task 2 - Discover the Kubernetes Network topology

Take a look at your services in the kube-system namespace:
`kubectl -n kube-system get services`

It nicely shows the different IP-ranges of the cluster:

- **172.25.0.0/16** **node IP addresses** (NodePort & LoadBalancer → EXTERNAL-IP)

  The subnet 172.x.0.0/16 may be different for every cluster.
  Remember the **`kubectl get node -o wide`** output?
  Also compare it with the docker networking configuration configured by k3d for your cluster:
  `docker network inspect k3d-ccp2`
  This represents the Infrastructure Subnet provided by the infrastructure provider connecting your cluster nodes. Here it is a virtual network using a docker bridge.

- **10.43.x.y/16** **service IP addresses** (CLUSTER-IP)

  This is the address range used for Kubernetes internal service addresses. Each service will get a cluster-IP address assigned from this address range.

- Finally, you can also discover the pod's IP address range by listing them using the wide output flag:

  `kubectl -n kube-system get pods -o wide`

```
NAME                                  READY   STATUS      RESTARTS     AGE    IP           NODE
helm-install-traefik-crd-5nm9t        0/1     Completed   0            7d4h   10.42.2.2    k3d-ccp2-agent-0
helm-install-traefik-76cnk            0/1     Completed   1            7d4h   10.42.1.2    k3d-ccp2-agent-1
svclb-traefik-683c3490-6gnwh          2/2     Running     4 (64m ago)  7d4h   10.42.0.6    k3d-ccp2-server-0
coredns-597584b69b-d8wgr              1/1     Running     2 (64m ago)  7d4h   10.42.1.14   k3d-ccp2-agent-1
svclb-traefik-683c3490-xr27n          2/2     Running     4 (64m ago)  7d4h   10.42.1.11   k3d-ccp2-agent-1
metrics-server-5f9f776df5-kwtdb       1/1     Running     4 (64m ago)  7d4h   10.42.0.7    k3d-ccp2-server-0
traefik-66c46d954f-k4zzp              1/1     Running     2 (64m ago)  7d4h   10.42.2.14   k3d-ccp2-agent-0
svclb-traefik-683c3490-x7q9b          2/2     Running     4 (64m ago)  7d4h   10.42.2.11   k3d-ccp2-agent-0
local-path-provisioner-79f67d76f8     1/1     Running     3 (64m ago)  7d4h   10.42.2.13   k3d-ccp2-agent-0
```

  10.42.x.y/16 **pod IP addresses**

  Can you detect a schema with the IP addresses regarding the node they are running on? Using this schema, how many nodes could we run and how many pods could we create on each node?

  Note: This addressing schema depends on the container networking technology used to interconnect the containers in the cluster.
  Kubernetes provides a pluggable Container Networking Interface (CNI). K3S (which is the Kubernetes Distribution used by K3D) uses the Flannel CNI by default.
  Other CNI implementations (like Canal, Calico or Cilium) could be installed → K3S Networking

Let's try to follow the flow of traffic for the hostname application from the ARCH Lab:

1. Make sure you have installed the hostname application to the default namespace.
2. Show the service and pod information: **`kubectl get pod,service -o wide`**

```
NAME                              READY   STATUS    RESTARTS       AGE    IP           NODE
pod/hostname-596df84595-9przb    1/1     Running   2 (117m ago)   7d3h   10.42.1.12   k3d-ccp2-agent-1
pod/hostname-596df84595-psz6q    1/1     Running   2 (117m ago)   7d3h   10.42.1.13   k3d-ccp2-agent-1
pod/hostname-596df84595-l95dz    1/1     Running   2 (117m ago)   7d3h   10.42.2.12   k3d-ccp2-agent-0

NAME                          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE    SELECTOR
service/kubernetes            ClusterIP   10.43.0.1       <none>        443/TCP   7d5h   <none>
service/hostname-service      ClusterIP   10.43.113.109   <none>        80/TCP    7d3h   app=hostname
```

3. Get the configuration of the service: **`kubectl describe service hostname-service`**

```
Name:              hostname-service
Namespace:         default
Labels:            <none>
Annotations:       <none>
Selector:          app=hostname
Type:              ClusterIP
IP Family Policy:  SingleStack
IP Families:       IPv4
IP:                10.43.113.109
IPs:               10.43.113.109
Port:              http  80/TCP
TargetPort:        8080/TCP
Endpoints:         10.42.1.12:8080,10.42.1.13:8080,10.42.2.12:8080
Session Affinity:  None
Events:            <none>
```

4. Next, have to look at the ingress configuration:
   `kubectl describe ingress hostname-ingress`

```
Name:             hostname-ingress
Labels:           <none>
Namespace:        default
Address:          172.25.0.2,172.25.0.3,172.25.0.4
Ingress Class:    traefik
Default backend:  <default>
Rules:
  Host                         Path  Backends
  ----                         ----  --------
  hostname.127.0.0.1.nip.io      /   hostname-service:80 (10.42.1.12:8080,10.42.1.13:8080,10.42.2.12:8080)
  hostname.160.85.253.250.nip.io /   hostname-service:80 (10.42.1.12:8080,10.42.1.13:8080,10.42.2.12:8080)
Annotations:                     ingress.kubernetes.io/ssl-redirect: false
Events:                          <none>
```

This allows to trace the flow of incoming packets for the hostname application:

→ VM 160.85.253.X:80 → serverlb

→ node-ports 172.25.0.2/3/4:80 (svclb-traefik)

→ service 10.43.113.109:80

→ pod(s) 10.42.1.12:8080,10.42.1.13:8080,10.42.2.12:8080

The hostname application uses a service of type ClusterIP. How would the path look like if you modify it using a NodePort service?

172.25.0.2:30xxx / 172.25.0.3:30xxx / 172.25.0.4:30xxx

Task 3 - Exercise resilience using Deployment / Replica Set

In this task you will verify, that replica-sets are restarting pods of a deployment if they fail.

The hostname application is already using a deployment of three instances, which automatically spawns a replica-set.

1. Check the status of the replica set and pods using

   ```
   kubectl get rs,pods -o wide
   ```

   It should show the status of the hostname replica-set (3/3 pods running) and list the 3 pods including IP addresses.

2. Next, you will kill some pods. To see the effect, you need to constantly monitor the status. You can do this using the `watch` command, which is automatically repeating the given command in the specified interval:

   ```
   watch -n 0.5 kubectl get rs,pod -o wide
   ```

3. In a second terminal window, connect to the VM using SSH and start killing pods: e.g,

   ```
   kubectl delete pod/hostname-596df84595-nbzr5
   ```

   (choose a pod name from your deployment). You may also kill multiple pods in one command
   - How does the system react?
   - On which node is it / are they started?
   - What do you discover regarding IP-Addresses?
   - Also try to access the application using the browser, does it work and respond with new hostnames (pod-names) or error messages?

4. You may also change the deployment and scale it to a different number of replicas:

   ```
   kubectl scale deployment hostname --replicas=5
   ```

   - How does the controller / replica-set react?
   - What happens, if you scale it to 0? (also in browser)
   - Can you also scale up again?

Cleanup

Open ports are always a security risk. At the end of the lab-session, you may stop the cluster
- k3d cluster stop ccp2

and restart it when you continue:
- k3d cluster start ccp2