Lesson 03 Demo 01

Managing and Administering a Kubernetes Cluster

Objective: To verify cluster certificates, create a namespace, and access clusters using the Kubernetes API

Tools required: kubeadm, kubectl, kubelet, and containerd

Prerequisites: A Kubernetes cluster should already be set up (refer to the steps provided in Lesson 02, Demo 01 for guidance).

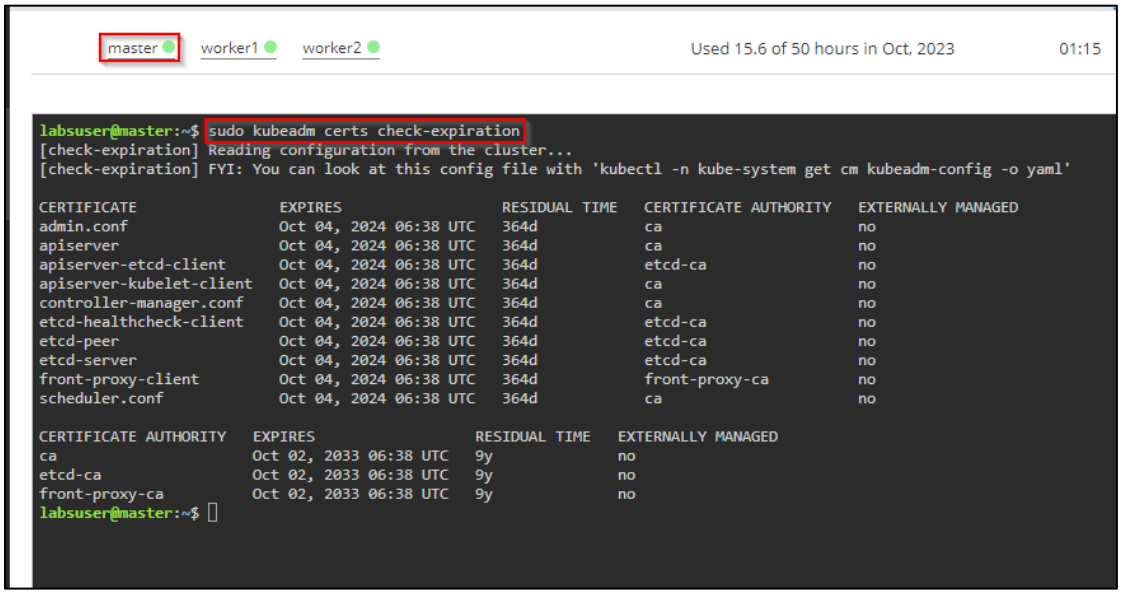
Steps to be followed:

1. Verify the certificates of the cluster
2. Create a namespace
3. Access clusters using the Kubernetes API

Step 1: Verify the certificates of the cluster

1.1 To check the expiration date of the certificate as a regular user, use the following command:

sudo kubeadm certs check-expiration



1.2 On the master node, enter the following command to review cluster information:
kubectl cluster-info

```
[check-expiration] Reading configuration from the cluster...
[check-expiration] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'

CERTIFICATE                EXPIRES                  RESIDUAL TIME   CERTIFICATE AUTHORITY   EXTERNALLY MANAGED
admin.conf                 Oct 04, 2024 06:38 UTC   364d            ca                      no
apiserver                  Oct 04, 2024 06:38 UTC   364d            ca                      no
apiserver-etcd-client      Oct 04, 2024 06:38 UTC   364d            etcd-ca                 no
apiserver-kubelet-client   Oct 04, 2024 06:38 UTC   364d            ca                      no
controller-manager.conf    Oct 04, 2024 06:38 UTC   364d            ca                      no
etcd-healthcheck-client    Oct 04, 2024 06:38 UTC   364d            etcd-ca                 no
etcd-peer                  Oct 04, 2024 06:38 UTC   364d            etcd-ca                 no
etcd-server                Oct 04, 2024 06:38 UTC   364d            etcd-ca                 no
front-proxy-client         Oct 04, 2024 06:38 UTC   364d            front-proxy-ca          no
scheduler.conf             Oct 04, 2024 06:38 UTC   364d            ca                      no

CERTIFICATE AUTHORITY   EXPIRES                  RESIDUAL TIME   EXTERNALLY MANAGED
ca                      Oct 02, 2033 06:38 UTC   9y              no
etcd-ca                 Oct 02, 2033 06:38 UTC   9y              no
front-proxy-ca          Oct 02, 2033 06:38 UTC   9y              no
labsuser@master:~$ kubectl cluster-info
Kubernetes control plane is running at https://172.31.23.240:6443
CoreDNS is running at https://172.31.23.240:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
labsuser@master:~$
labsuser@master:~$
```

1.3 On the master node, enter the following command to view complete cluster information
kubectl cluster-info dump

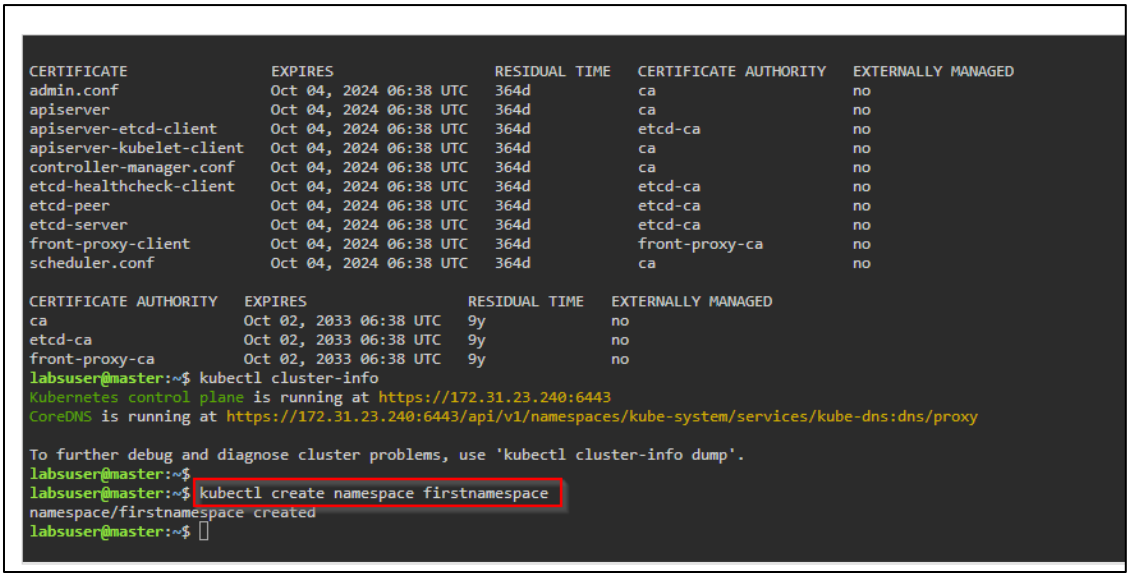

```
        },
        "schedulerName": "default-scheduler",
        "tolerations": [
            {
                "key": "node.kubernetes.io/not-ready",
                "operator": "Exists",
                "effect": "NoExecute",
                "tolerationSeconds": 300
            },
            {
                "key": "node.kubernetes.io/unreachable",
                "operator": "Exists",
                "effect": "NoExecute",
                "tolerationSeconds": 300
            }
        ],
        "priority": 0,
        "enableServiceLinks": true,
        "preemptionPolicy": "PreemptLowerPriority"
    },
    "status": {
        "phase": "Pending",
        "conditions": [
            {
                "type": "PodScheduled",
                "status": "False",
                "lastProbeTime": null,
                "lastTransitionTime": "2023-10-31T07:38:01Z",
                "reason": "Unschedulable",
```

```
==== END logs for container redis-server of pod default/redis-cache-8478cbdc86-wldjq ====
==== START logs for container test-pod of pod default/test-pod ====
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2023/10/31 11:44:30 [notice] 1#1: using the "epoll" event method
2023/10/31 11:44:30 [notice] 1#1: nginx/1.25.3
2023/10/31 11:44:30 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2023/10/31 11:44:30 [notice] 1#1: OS: Linux 6.2.0-1014-aws
2023/10/31 11:44:30 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1024:524288
2023/10/31 11:44:30 [notice] 1#1: start worker processes
2023/10/31 11:44:30 [notice] 1#1: start worker process 29
2023/10/31 11:44:30 [notice] 1#1: start worker process 30
==== END logs for container test-pod of pod default/test-pod ====
==== START logs for container web-app of pod default/web-server-55f57c89d4-8lnnb ====
==== END logs for container web-app of pod default/web-server-55f57c89d4-8lnnb ====
==== START logs for container web-app of pod default/web-server-55f57c89d4-kh5st ====
==== END logs for container web-app of pod default/web-server-55f57c89d4-kh5st ====
==== START logs for container web-app of pod default/web-server-55f57c89d4-rbxrf ====
==== END logs for container web-app of pod default/web-server-55f57c89d4-rbxrf ====
```

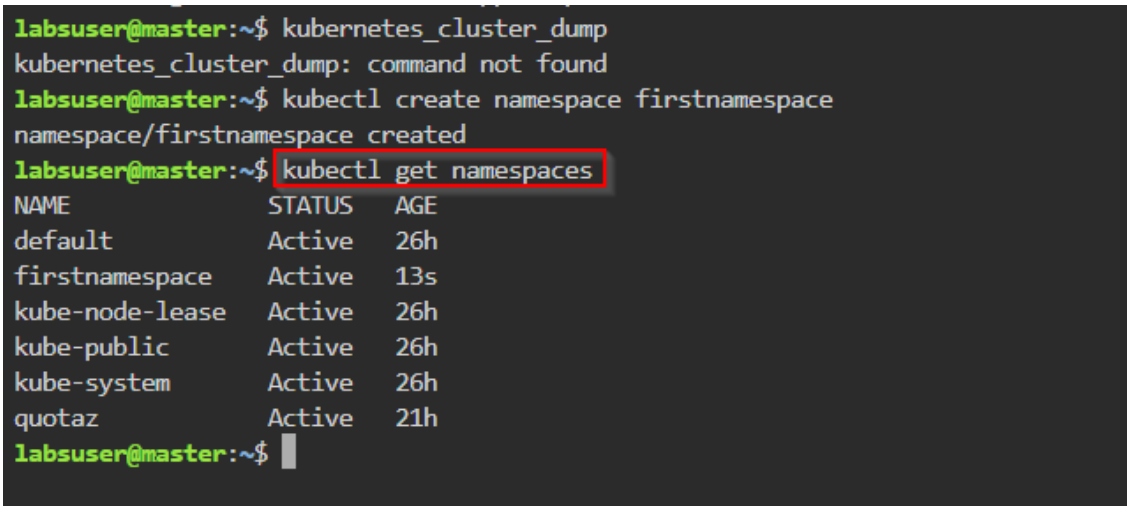
Note: You can also export the dump to a file,
kubectl cluster-info dump > kubernetes_cluster_dump.

Step 2: Create a namespace

2.1 Use the following command to create a namespace:
kubectl create namespace firstnamespace

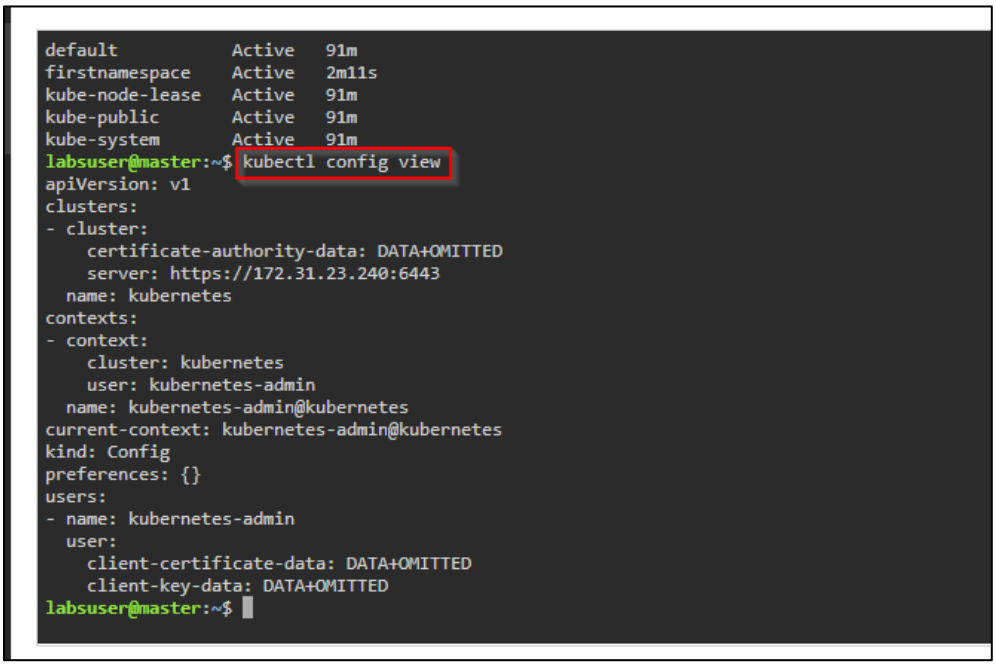


2.2 Confirm the creation of the new namespace with the following command:
kubectl get namespaces

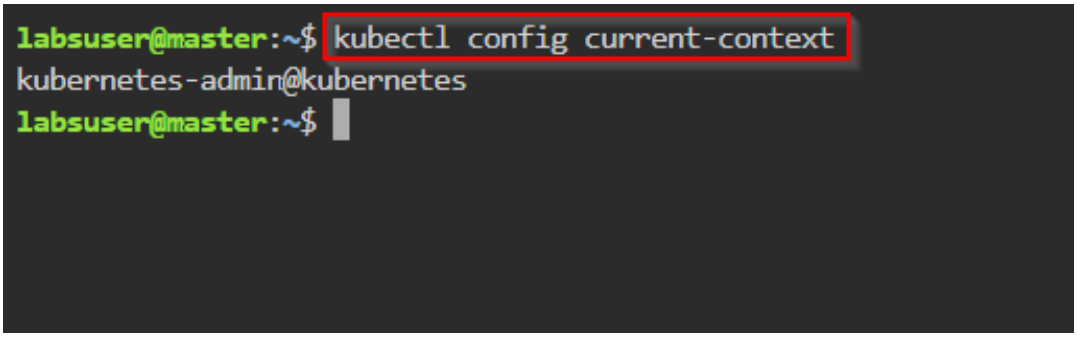


Step 3: Access clusters using the Kubernetes API

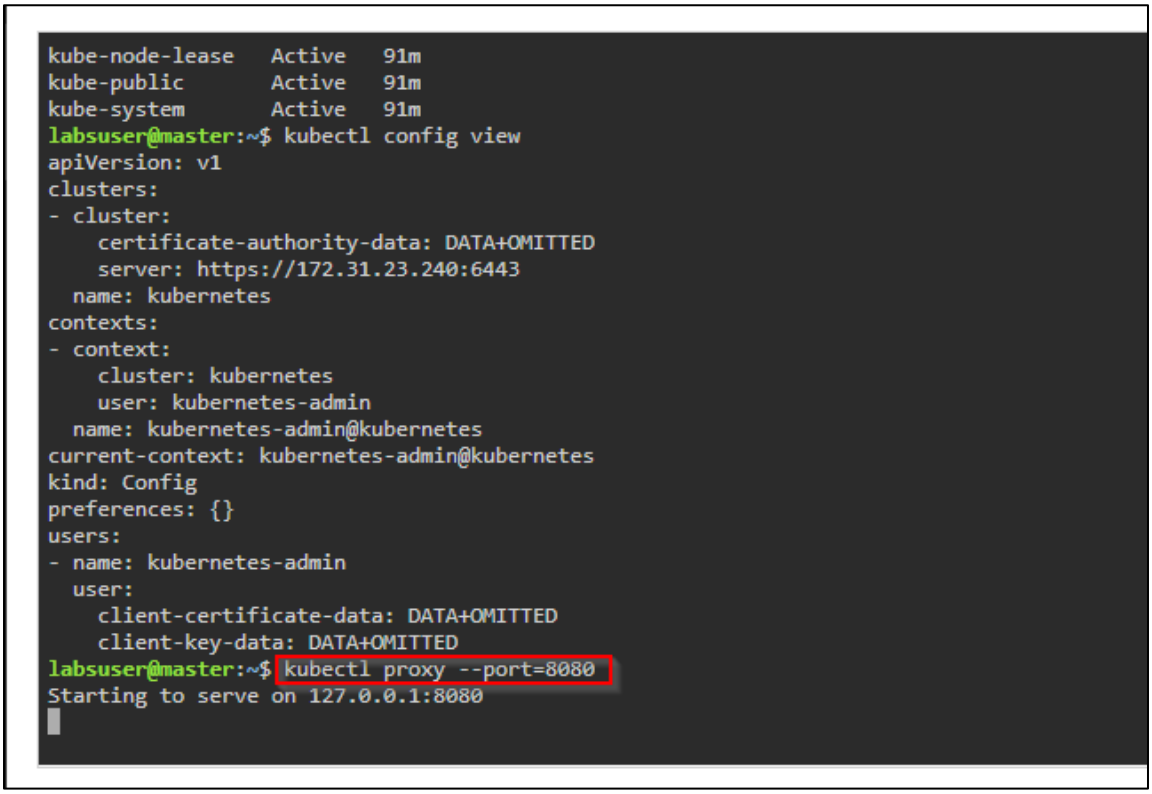
3.1 To view the cluster configuration, use the command below:
kubectl config view



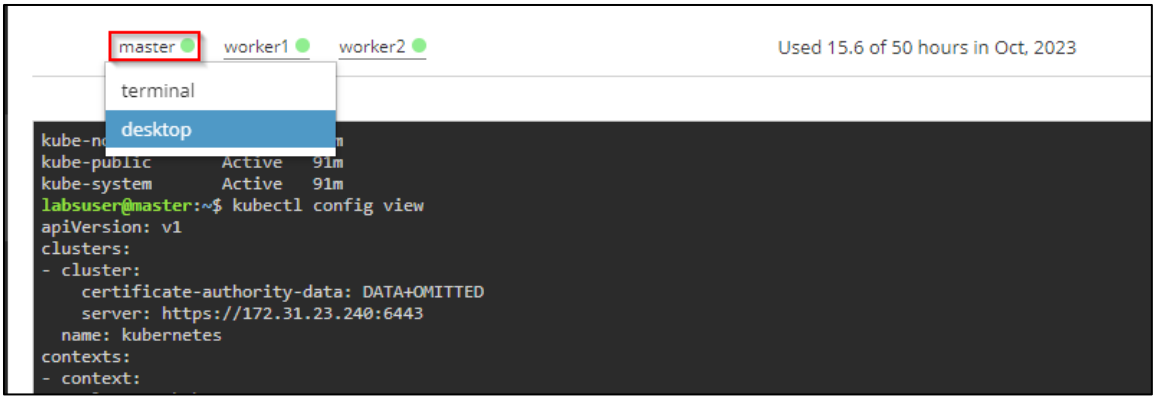
3.2 Run the following command to view the current cluster:
kubectl config current-context



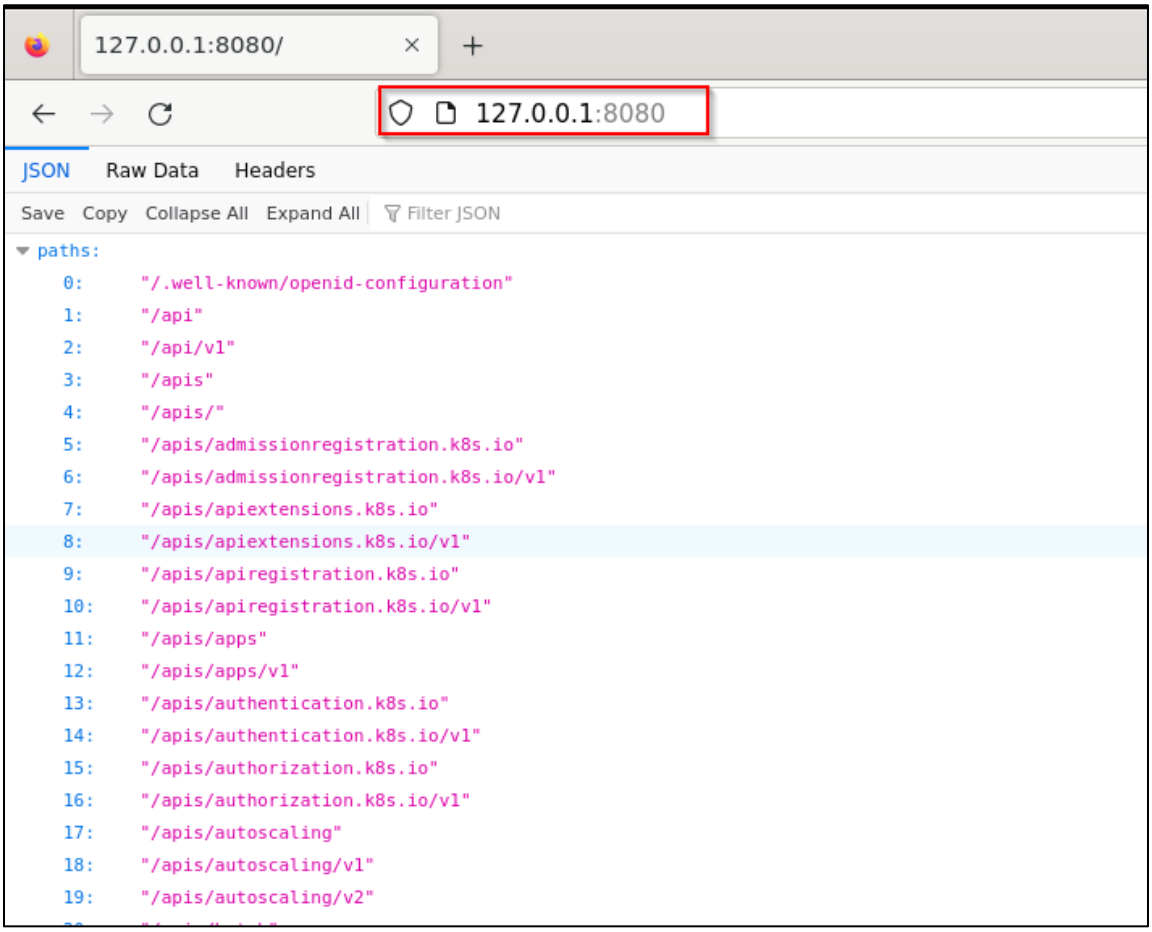
3.3 To identify the API server, execute and copy the **127.0.0.1:8080** port as shown below:
kubectl proxy --port=8080



3.4 In the lab environment, select the **master** tab and choose the **desktop** option



3.5 Navigate to the **desktop** tab and open the **Firefox** browser to access the **API server** by typing the IP address and port mentioned in step 3.2 output



Note: Use **CTRL+C** in the terminal to exit and stop port forwarding.

By following these steps, you have successfully verified cluster certificates, created a namespace, and accessed the Kubernetes API.