Lesson 09 Demo 08

Handling Component Failure Threshold

Objective: To view the nodes within a cluster and gather detailed health information for ensuring a proper functioning of the nodes

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. Check the cluster health information

Step 1: Check the cluster health information

1.1 Execute the following command to check the nodes in the cluster:

kubectl get nodes

```
labsuser@master:~$ kubectl get nodes
NAME                        STATUS   ROLES           AGE   VERSION
master.example.com          Ready    control-plane   49m   v1.28.2
worker-node-1.example.com   Ready    <none>          47m   v1.28.2
worker-node-2.example.com   Ready    <none>          47m   v1.28.2
labsuser@master:~$
```

1.2 To check the health information of a cluster and verify its content, execute the following commands:

kubectl cluster-info dump > dump.json

vi dump.json

```
labsuser@master:~$ kubectl get nodes
NAME                       STATUS   ROLES           AGE    VERSION
master.example.com         Ready    control-plane   4h42m  v1.28.2
worker-node-1.example.com  Ready    <none>          4h41m  v1.28.2
worker-node-2.example.com  Ready    <none>          4h41m  v1.28.2
labsuser@master:~$ kubectl cluster-info dump > dump.json
labsuser@master:~$ vi dump.json
labsuser@master:~$
```


```
{
    "kind": "NodeList",
    "apiVersion": "v1",
    "metadata": {
        "resourceVersion": "23821"
    },
    "items": [
        {
            "metadata": {
                "name": "master.example.com",
                "uid": "cf00285b-4007-49e0-ac04-b6b2b7f758d9",
                "resourceVersion": "23512",
                "creationTimestamp": "2023-10-17T10:46:38Z",
                "labels": {
                    "beta.kubernetes.io/arch": "amd64",
                    "beta.kubernetes.io/os": "linux",
                    "kubernetes.io/arch": "amd64",
                    "kubernetes.io/hostname": "master.example.com",
                    "kubernetes.io/os": "linux",
                    "node-role.kubernetes.io/control-plane": "",
                    "node.kubernetes.io/exclude-from-external-load-balancers": ""
                },
                "annotations": {
                    "kubeadm.alpha.kubernetes.io/cri-socket": "unix:///var/run/containerd/containerd.sock",
                    "node.alpha.kubernetes.io/ttl": "0",
                    "projectcalico.org/IPv4Address": "172.31.35.149/20",
                    "projectcalico.org/IPv4IPIPTunnelAddr": "172.16.204.64",
                    "volumes.kubernetes.io/controller-managed-attach-detach": "true"
                }
            },
"dump.json" 12784L, 1309856B                                                  1
```


As shown in the screenshot above, **kubectl cluster-info dump > dump.json** generates a cluster information dump and redirects the output to a file named **dump.json**.

Note: Examine the **dump.json** file to get the details of cluster's health

By following these steps, you have successfully enlisted a comprehensive collection of diagnostic information about the Kubernetes cluster, including details about the cluster's configuration, resources, and status.