Lesson-End Project

Fetching Cluster-Specific Configuration

Project agenda: To retrieve cluster-specific configurations from a running Kubernetes cluster, ensuring detailed insights into nodes, API versions, and operational statuses for optimal functionality and deployment readiness

Description: Your team lead has given you the task of accessing a Kubernetes cluster to gather specific details about it. You need to report on the available nodes and their IP addresses, determine the API versions supported by the server, check the status of the control plane and CoreDNS, and assess the status of the pods within the kube-system namespace.

Tools required: kubeadm, kubectl, kubelet, and containerd

Prerequisites: A Kubernetes cluster (refer to Demo 01 from Lesson 01 for setting up a cluster)

Expected deliverables: A Kubernetes cluster with high availability enabled

Steps to be followed:

1. List the available nodes and their IP addresses
2. Identify supported API versions
3. Examine the control plane and CoreDNS status
4. Review the status of the pods in the kube-system namespace

Step 1: List the available nodes and their IP addresses

1.1 Use the command below to display the nodes and their statuses within the cluster:
kubectl get nodes

```
labsuser@master:~$ kubectl get nodes
NAME                       STATUS   ROLES           AGE    VERSION
master.example.com         Ready    control-plane   5h23m  v1.28.2
worker-node-1.example.com  Ready    <none>          5h22m  v1.28.2
worker-node-2.example.com  Ready    <none>          5h19m  v1.28.2
labsuser@master:~$
```

1.2 Use the following command to retrieve detailed information about the nodes and their IP addresses:
kubectl get nodes -o wide

```
labsuser@master:~$ kubectl get nodes -o wide
NAME                       STATUS   ROLES           AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VE
RSION    CONTAINER-RUNTIME
master.example.com         Ready    control-plane   5h23m   v1.28.2   172.31.18.220   <none>        Ubuntu 22.04.3 LTS   6.2.0-101
2-aws    containerd://1.6.8
worker-node-1.example.com  Ready    <none>          5h22m   v1.28.2   172.31.13.202   <none>        Ubuntu 22.04.3 LTS   6.2.0-101
2-aws    containerd://1.6.8
worker-node-2.example.com  Ready    <none>          5h20m   v1.28.2   172.31.1.23     <none>        Ubuntu 22.04.3 LTS   6.2.0-101
2-aws    containerd://1.6.8
labsuser@master:~$
```

Step 2: Identify supported API versions

2.1 Execute the following command to find the resources available in the cluster and their respective versions:
kubectl api-versions

```
labsuser@master:~$ kubectl api-versions
admissionregistration.k8s.io/v1
apiextensions.k8s.io/v1
apiregistration.k8s.io/v1
apps/v1
authentication.k8s.io/v1
authorization.k8s.io/v1
autoscaling/v1
autoscaling/v2
batch/v1
certificates.k8s.io/v1
coordination.k8s.io/v1
crd.projectcalico.org/v1
discovery.k8s.io/v1
events.k8s.io/v1
flowcontrol.apiserver.k8s.io/v1beta2
flowcontrol.apiserver.k8s.io/v1beta3
networking.k8s.io/v1
node.k8s.io/v1
policy/v1
rbac.authorization.k8s.io/v1
scheduling.k8s.io/v1
storage.k8s.io/v1
v1
labsuser@master:~$
```

2.2 Use the commands below to check the versions of both kubeadm and kubectl:
kubeadm version
kubectl version

```
labsuser@master:~$ kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"28", GitVersion:"v1.28.2", GitCommit:"89a4ea3e1e4ddd7f7572286090359983e0387b2f", G
itTreeState:"clean", BuildDate:"2023-09-13T09:34:32Z", GoVersion:"go1.20.8", Compiler:"gc", Platform:"linux/amd64"}
labsuser@master:~$ kubectl version
Client Version: v1.28.2
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Server Version: v1.28.2
labsuser@master:~$
```

Step 3: Examine the control plane and CoreDNS status

3.1 Use the commands below to gain insights into a Kubernetes cluster's configuration:
kubectl config view

```
labsuser@master:~$ kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://172.31.18.220:6443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data: DATA+OMITTED
    client-key-data: DATA+OMITTED
labsuser@master:~$
```

3.2 Run the following command to find the operational status of the control plane and CoreDNS:
kubectl cluster-info

```
labsuser@master:~$ kubectl cluster-info
Kubernetes control plane is running at https://172.31.18.220:6443
CoreDNS is running at https://172.31.18.220:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

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

This command provides URLs where the control plane components and CoreDNS are operational.

Step 4: Review the status of the pods in the kube-system namespace

4.1 Use the command below to list all the available pods within the kube-system namespace:
kubectl get pods -n kube-system

```
labsuser@master:~$ kubectl get pods -n kube-system
NAME                                          READY   STATUS    RESTARTS        AGE
calico-kube-controllers-7ddc4f45bc-fwr4c      1/1     Running   11 (13m ago)    5h27m
calico-node-bs6rd                             1/1     Running   3 (13m ago)     5h24m
calico-node-kcd7x                             1/1     Running   7 (13m ago)     5h27m
calico-node-njqwj                             1/1     Running   3 (13m ago)     5h27m
coredns-5dd5756b68-8nbtm                      1/1     Running   4 (13m ago)     5h27m
coredns-5dd5756b68-fbmdl                      1/1     Running   8 (13m ago)     5h27m
etcd-master.example.com                       1/1     Running   36 (13m ago)    5h26m
kube-apiserver-master.example.com             1/1     Running   41 (13m ago)    5h28m
kube-controller-manager-master.example.com    1/1     Running   43 (13m ago)    5h26m
kube-proxy-nntss                              1/1     Running   3 (13m ago)     5h27m
kube-proxy-wn9v5                              1/1     Running   3 (13m ago)     5h24m
kube-proxy-z8dkr                              1/1     Running   13 (13m ago)    5h27m
kube-scheduler-master.example.com             1/1     Running   38 (13m ago)    5h26m
labsuser@master:~$
```

From the output, you can observe that multiple pods operate under the kube-system namespace.

By following these steps, you have successfully obtained detailed configurations and insights into your Kubernetes cluster, ensuring its optimal functionality and readiness for deployment tasks.