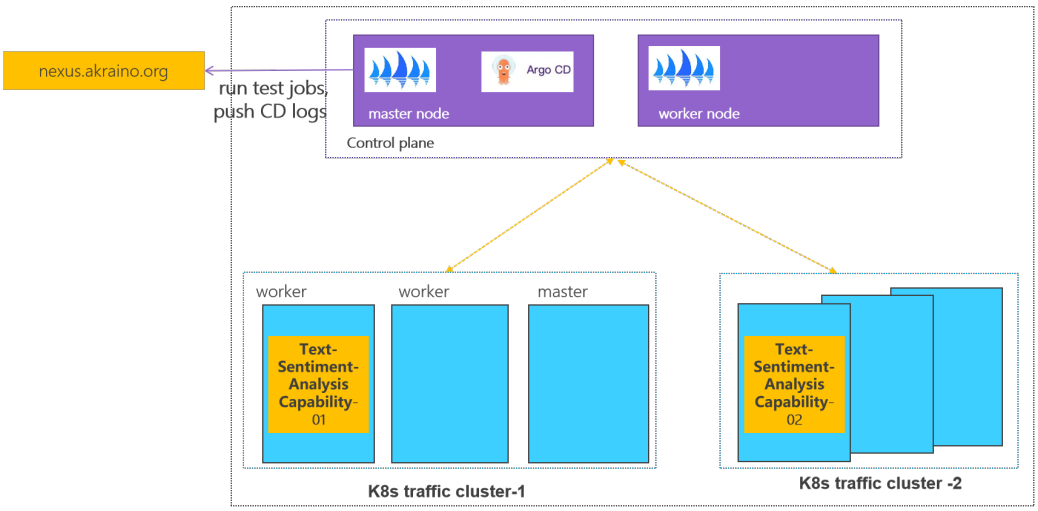
R7 Installation Document

Introduction

This document describes steps required to deploy a sample environment and test for CFN (Computing Force Network) Ubiquitous Computing Force Scheduling Blueprint.

Deployment Architecture



Control plane: one k8s cluster is deployed in private lab.

Traffic plane: two K8s clusters are deployed in private lab.

Pre-Installation Requirements

- Hardware Requirements

64-bit CentOS 7

- Software Perequisites

docker-ce-20.10.11

kubelet-1.23.7

kubeadm-1.23.7

kubectl-1.23.7

- Database Perequisites

schema scripts: N/A

- Other Installation Requirements
    - Jump Host RequirementsN/A
    - Network Requirements: N/A
    - Bare Metal Node Requirements N/A
    - Execution Requirements (Bare Metal Only) N/A

Installation control plane cluster

0 Environmental description

At least two CentOS machines are required, one as the master node and the other as the worker node. The installed k8s version is 1.23.7.

There will be a comment like #master in front of each bash command, which is used to indicate which type of machine the command is used on. If there is no comment, the bash command needs to be executed on both types of machines

This document contains the operation and execution process, you can compare the screenshots of the document during the installation process.

1 Basic environment preparation

Both master node and worker node need to execute.

Preparing the basic environment to ensure the normal execution of subsequent operations.

1.1 Confirm your OS

Confirm that the operating system of the current machine is CentOS 7 .

Execute command

```
cat /etc/redhat-release
```

Execute screenshot

```
[root@worker1 ~]# cat /etc/redhat-release
CentOS Linux release 7.6.1810 (Core)
```

1.2 Set hostname

If the name is long, it is recommended to use a combination of letters and dashes, such as "aa-bb-cc", here directly set to master and worker1.

Execute command

```
# master
hostnamectl set-hostname master
hostnamectl
# worker
hostnamectl set-hostname worker1
hostnamectl
```

Execute screenshot

```
[root@cluster1-node5 ~]# hostnamectl set-hostname master
[root@cluster1-node5 ~]# hostnamectl
   Static hostname: master
         Icon name: computer-vm
           Chassis: vm
```

```
[root@cluster1-node1 bin]#  hostnamectl set-hostname worker1
[root@cluster1-node1 bin]# hostnamectl
   Static hostname: worker1
         Icon name: computer-vm
           Chassis: vm
```

The changed host name needs to take effect after reboot.

Execute command

```
reboot
```

1.3 Set address mapping

Set address mapping, and test the network.

Execute command

```
cat <<EOF>> /etc/hosts
${YOUR IP} master
${YOUR IP} worker1
EOF
ping master
ping worker1
```

Execute screenshot

```
[root@master ~]# cat <<EOF>> /etc/hosts
> 36.137.235.80 master
> 36.137.236.195 worker1
> EOF
[root@master ~]# ping master
PING master (36.137.235.80) 56(84) bytes of data.
64 bytes from master (36.137.235.80): icmp_seq=1 ttl=53 time=0.978 ms
64 bytes from master (36.137.235.80): icmp_seq=2 ttl=53 time=0.340 ms
^C
--- master ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.340/0.659/0.978/0.319 ms
[root@master ~]# ping worker1
PING worker1 (36.137.236.195) 56(84) bytes of data.
64 bytes from worker1 (36.137.236.195): icmp_seq=1 ttl=52 time=1.36 ms
64 bytes from worker1 (36.137.236.195): icmp_seq=2 ttl=52 time=0.551 ms
^C
--- worker1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.551/0.958/1.365/0.407 ms
```

```
[root@worker1 ~]# cat <<EOF>> /etc/hosts
> 36.137.235.80 master
> 36.137.236.195 worker1
> EOF
[root@worker1 ~]# ping master
PING master (36.137.235.80) 56(84) bytes of data.
64 bytes from master (36.137.235.80): icmp_seq=1 ttl=52 time=1.40 ms
64 bytes from master (36.137.235.80): icmp_seq=2 ttl=52 time=0.436 ms
^C
--- master ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.436/0.920/1.404/0.484 ms
[root@worker1 ~]# ping worker1
PING worker1 (36.137.236.195) 56(84) bytes of data.
64 bytes from worker1 (36.137.236.195): icmp_seq=1 ttl=51 time=0.857 ms
64 bytes from worker1 (36.137.236.195): icmp_seq=2 ttl=51 time=0.490 ms
^C
--- worker1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.490/0.673/0.857/0.185 ms
```

1.4 Turn off firewall, selinux and swap

Execute command

```
systemctl stop firewalld
systemctl disable firewalld
setenforce 0 sed -i "s/^SELINUX=enforcing/SELINUX=disabled/g" /etc/selinux/config
swapoff -a
sed -i 's/.*swap.*/#&/' /etc/fstab
```

Execute screenshot

```
[root@master ~]# systemctl stop firewalld
[root@master ~]# systemctl disable firewalld
[root@master ~]# setenforce 0
[root@master ~]# sed -i "s/^SELINUX=enforcing/SELINUX=disabled/g" /etc/selinux/config
[root@master ~]# swapoff -a
[root@master ~]# sed -i 's/.*swap.*/#&/' /etc/fstab
```

```
[root@worker1 ~]# systemctl stop firewalld
[root@worker1 ~]# systemctl disable firewalld
[root@worker1 ~]# setenforce 0
[root@worker1 ~]# sed -i "s/^SELINUX=enforcing/SELINUX=disabled/g" /etc/selinux/config
[root@worker1 ~]# swapoff -a
[root@worker1 ~]# sed -i 's/.*swap.*/#&/' /etc/fstab
```

1.5 Download the new yum source

Execute command

```
rm -rf /etc/yum.repos.d/* ;wget ftp://ftp.rhce.cc/k8s/* -P /etc/yum.repos.d/
ls /etc/yum.repos.d/
```

Execute screenshot

```
--2022-11-26 16:00:07--  ftp://ftp.rhce.cc/k8s/k8s.repo.bak
           => '/etc/yum.repos.d/k8s.repo.bak'
==> CWD not required.
==> PASV ... done.    ==> RETR k8s.repo.bak ... done.
Length: 276

100%[===================================================================================================>] 276         --.-K/s   in 0s

2022-11-26 16:00:07 (1.81 MB/s) - '/etc/yum.repos.d/k8s.repo.bak' saved [276]

[root@master ~]#
[root@master ~]#
[root@master ~]#
[root@master ~]# ls /etc/yum.repos.d/
CentOS-Base.repo  docker-ce.repo  docker-ce.repo.bak  epel.repo  k8s.repo  k8s.repo.bak
```

1.6 Set iptables

Execute command

```
cat <<EOF> /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
sysctl -p /etc/sysctl.d/k8s.conf
```

Execute screenshot

```
[root@master ~]# cat <<EOF> /etc/sysctl.d/k8s.conf
> net.bridge.bridge-nf-call-ip6tables = 1
> net.bridge.bridge-nf-call-iptables = 1
> net.ipv4.ip_forward = 1
> EOF
[root@master ~]# sysctl -p /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
```

1.7 Make sure the time zone and time are correct

Execute command

```
timedatectl set-timezone Asia/Shanghai
systemctl restart rsyslog
```

Execute screenshot

```
[root@master ~]# timedatectl set-timezone Asia/Shanghai
[root@master ~]# systemctl restart rsyslog
```

2 Install docker

Both master node and worker node need to execute

The main content is to install docker-ce, and configure the cgroup driver of docker as systemd, confirm the driver.

2.1 Uninstall old docker

Execute command

```
yum -y remove docker docker-client docker-client-latest docker-common docker-latest docker-latest-logrotate
docker-logrotate docker-engine docker-ce docker-ce-cli
```

Execute screenshot

```
[root@master ~]#  yum -y remove docker docker-client docker-client-latest docker-common docker-latest docker-latest-logrotate docker-logrotate dock
er-engine   docker-ce docker-ce-cli
Loaded plugins: versionlock
No Match for argument: docker
No Match for argument: docker-client
No Match for argument: docker-client-latest
No Match for argument: docker-common
No Match for argument: docker-latest
No Match for argument: docker-latest-logrotate
No Match for argument: docker-logrotate
No Match for argument: docker-engine
Resolving Dependencies
--> Running transaction check
---> Package docker-ce.x86_64 3:20.10.21-3.el7 will be erased
--> Processing Dependency: docker-ce for package: docker-ce-rootless-extras-20.10.21-3.el7.x86_64
```

```
Removed:
  docker-ce.x86_64 3:20.10.21-3.el7                                     docker-ce-cli.x86_64 1:20.10.21-3.el7

Dependency Removed:
  docker-ce-rootless-extras.x86_64 0:20.10.21-3.el7

Complete!
```

2.2 Install docker

Execute command

```
yum -y install docker-ce
```

Execute screenshot

```
Installed:
  docker-ce.x86_64 3:20.10.21-3.el7

Dependency Installed:
  docker-ce-cli.x86_64 1:20.10.21-3.el7                                 docker-ce-rootless-extras.x86_64 0:20.10.21-3.el7

Complete!
```

2.3 Set docker to boot and confirm docker status

Execute command

```
systemctl enable docker
systemctl start docker
systemctl status docker
```

Execute screenshot

```
[root@worker1 ~]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/etc/systemd/system/docker.service; disabled; vendor preset: disabled)
  Drop-In: /etc/systemd/system/docker.service.d
           └─docker-options.conf
   Active: active (running) since Sat 2022-11-26 12:12:58 CST; 2h 15min ago
     Docs: http://docs.docker.com
 Main PID: 5504 (dockerd)
```

2.4 Configure the driver of docker's cgroup

The driver of docker's cgroup needs to be consistent with that of kubelet. It is recommended to set it to systemd.

Execute command

```
docker info | grep -i cgroup
```

Check the current configuration, if it is the system in the figure below, skip the follow-up and go directly to the third section

Execute screenshot

```
[root@master ~]#  docker info | grep -i cgroup
 Cgroup Driver: systemd
 Cgroup Version: 1
```

If it is cgroupfs, add the following statement

Execute command

```
vim /etc/docker/daemon.json
# add this
{
  "exec-opts": ["native.cgroupdriver=systemd"]
}
```

Restart to check for changes

Execute command

```
systemctl daemon-reload && systemctl restart docker
docker info | grep -i cgroup
```

```
[root@node1 ~]#  docker info | grep -i cgroup
 Cgroup Driver: cgroupfs
 Cgroup Version: 1
[root@node1 ~]# vim /etc/docker/daemon.json
[root@node1 ~]# systemctl daemon-reload && systemctl restart docker
[root@node1 ~]# docker info | grep -i cgroup
 Cgroup Driver: systemd
 Cgroup Version: 1
[root@node1 ~]#
```

3 Install k8s basic components

Both master node and worker node need to execute

The main content is to install the 1.23.7 version of the component kubeadm kubectl kubelet

3.1 Check kubeadm kubectl kubelet

If it is the version inconsistent, you need to uninstall it through yum remove ${name}.

Execute command

```
yum list installed | grep kube
```

Execute screenshot

```
[root@master ~]# yum list installed | grep kube
[root@master ~]#
```

3.2 Install kubelet kubeadm kubectl version 1.23.7

Execute command

```
yum -y install kubelet-1.23.7 kubeadm-1.23.7 kubectl-1.23.7
```

Execute screenshot

```
[root@worker1 yum.repos.d]# yum -y install kubelet-1.23.7-0 kubeadm-1.23.7-0 kubectl-1.23.7-0
Loaded plugins: versionlock
Resolving Dependencies
--> Running transaction check
---> Package kubeadm.x86_64 0:1.23.7-0 will be installed
--> Processing Dependency: cri-tools >= 1.19.0 for package: kubeadm-1.23.7-0.x86_64
--> Processing Dependency: kubernetes-cni >= 0.8.6 for package: kubeadm-1.23.7-0.x86_64
---> Package kubectl.x86_64 0:1.23.7-0 will be installed
```

```
Installed:
  kubeadm.x86_64 0:1.23.7-0                      kubectl.x86_64 0:1.23.7-0                      kubelet.x86_64 0:1.23.7-0

Dependency Installed:
  cri-tools.x86_64 0:1.25.0-0                                        kubernetes-cni.x86_64 0:1.1.1-0

Complete!
```

3.3 Verify installation

Execute command

```
kubelet --version
kubeadm version
kubectl version
```

Execute screenshot

```
[root@master ~]# kubelet --version
Kubernetes v1.23.7
[root@master ~]# kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"23", GitVersion:"v1.23.7", GitCommit:"42c05a547468804b2053ecf60a3bd15560362fc2", GitTreeState:"cle
an", BuildDate:"2022-05-24T12:29:44Z", GoVersion:"go1.17.10", Compiler:"gc", Platform:"linux/amd64"}
[root@master ~]# kubectl version
Client Version: version.Info{Major:"1", Minor:"23", GitVersion:"v1.23.7", GitCommit:"42c05a547468804b2053ecf60a3bd15560362fc2", GitTreeState:"clean
", BuildDate:"2022-05-24T12:30:55Z", GoVersion:"go1.17.10", Compiler:"gc", Platform:"linux/amd64"}
The connection to the server localhost:8080 was refused - did you specify the right host or port?
```

4 Initialize the master

Execute only on the master node

The main content is to pull the image of version 1.23.7, initialize the master node, and configure the cilium network plugin for the master node

4.1 Pull the k8s image Pull the k8s image

Execute command

```
# master
kubeadm config images list --kubernetes-version=v1.23.7 |sed -e 's/^/docker pull /g' -e 's#k8s.gcr.io#registry.
aliyuncs.com/google_containers#g' |sh -x
docker pull registry.aliyuncs.com/google_containers/coredns:v1.8.6
docker images
```

Execute screenshot

```
[root@master ~]# docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
[root@master ~]# kubernetes_version="v1.23.7"
[root@master ~]# kubeadm config images list --kubernetes-version=v1.23.7 |sed -e 's/^/docker pull /g' -e 's#k8s.gcr.io#registry.aliyuncs.com/google
_containers#g' |sh -x
+ docker pull registry.aliyuncs.com/google_containers/kube-apiserver:v1.23.7
v1.23.7: Pulling from google_containers/kube-apiserver
36698cfa5275: Pull complete
8ccb9e3bd522: Pull complete
301cac25da41: Pull complete
Digest: sha256:a7206717f4a04b2dbd33505c0c4eb93907b892f5a610488ef7bfe94c6692be99
Status: Downloaded newer image for registry.aliyuncs.com/google_containers/kube-apiserver:v1.23.7
registry.aliyuncs.com/google_containers/kube-apiserver:v1.23.7
+ docker pull registry.aliyuncs.com/google_containers/kube-controller-manager:v1.23.7
v1.23.7: Pulling from google_containers/kube-controller-manager
36698cfa5275: Already exists
8ccb9e3bd522: Already exists
5dac8c9289ed: Pull complete
Digest: sha256:db4970df9c7a657d31299a6bc96b86d9d36c1d91e914b3428a43392fa4299068
```

```
[root@master ~]# docker pull registry.aliyuncs.com/google_containers/coredns:v1.8.6
v1.8.6: Pulling from google_containers/coredns
d92bdee79785: Pull complete
6e1b7c06e42d: Pull complete
Digest: sha256:5b6ec0d6de9baaf3e92d0f66cd96a25b9edbce8716f5f15dcd1a616b3abd590e
Status: Downloaded newer image for registry.aliyuncs.com/google_containers/coredns:v1.8.6
registry.aliyuncs.com/google_containers/coredns:v1.8.6
[root@master ~]#
[root@master ~]#
[root@master ~]# docker images
REPOSITORY                                                    TAG       IMAGE ID       CREATED         SIZE
registry.aliyuncs.com/google_containers/kube-apiserver        v1.23.7   03c169f383d9   6 months ago    135MB
registry.aliyuncs.com/google_containers/kube-controller-manager   v1.23.7   e34d4a6252ed   6 months ago    125MB
registry.aliyuncs.com/google_containers/kube-proxy            v1.23.7   b1aa05aa5100   6 months ago    112MB
registry.aliyuncs.com/google_containers/kube-scheduler        v1.23.7   ed0ccfa052ab   6 months ago    53.5MB
registry.aliyuncs.com/google_containers/etcd                  3.5.1-0   25f8c7f3da61   13 months ago   293MB
registry.aliyuncs.com/google_containers/coredns               v1.8.6    a4ca41631cc7   13 months ago   46.8MB
registry.aliyuncs.com/google_containers/pause                 3.6       6270bb605e12   15 months ago   683kB
```

Please make sure that the above 7 images have been pulled down

4.2 Init master

Execute command

```
# master
kubeadm init --image-repository registry.aliyuncs.com/google_containers --kubernetes-version=v1.23.7 --pod-
network-cidr=10.10.0.0/16
```

Execute screenshot

```
[root@master ~]# kubeadm init --image-repository registry.aliyuncs.com/google_containers --kubernetes-version=v1.23.7 --pod-network-cidr=10.10.0.0/16
[init] Using Kubernetes version: v1.23.7
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local master] and IPs [10.96.0.1 192.168.30.12]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [localhost master] and IPs [192.168.30.12 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [localhost master] and IPs [192.168.30.12 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Starting the kubelet
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests". This can take up to 4m0s
[apiclient] All control plane components are healthy after 5.002161 seconds
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config-1.23" in namespace kube-system with the configuration for the kubelets in the cluster
```

Let kubectl take effect

Execute command

```
# master
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
kubectl get po -A
```

Execute screenshot

```
[root@master ~]#   mkdir -p $HOME/.kube
[root@master ~]#   sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[root@master ~]#   sudo chown $(id -u):$(id -g) $HOME/.kube/config
[root@master ~]#
[root@master ~]# kubectl get po
No resources found in default namespace.
[root@master ~]# kubectl get po -A
NAMESPACE     NAME                                READY   STATUS              RESTARTS   AGE
kube-system   coredns-6d8c4cb4d-jnfvg             0/1     ContainerCreating   0          6m7s
kube-system   coredns-6d8c4cb4d-sv7wg             0/1     ContainerCreating   0          6m7s
kube-system   etcd-master                         1/1     Running             2          6m21s
kube-system   kube-apiserver-master               1/1     Running             6          6m21s
kube-system   kube-controller-manager-master      1/1     Running             2          6m21s
kube-system   kube-proxy-2pc4c                    1/1     Running             0          6m7s
kube-system   kube-scheduler-master               1/1     Running             6          6m21s
```

We can see that the coredns is not ready, so we need configure the network plugin

Note that if an error occurs and you need to reinit, you need to execute the following statement first to ensure that kubeadm is re-executed normally

```
# master, if an error occurs
kubeadm reset -f
rm -rf ~/.kube/
rm -rf /etc/kubernetes/
rm -rf /var/lib/etcd
rm -rf /var/etcd
```

4.3 Configure the cilium network plugin

Here select cilium as the network plugin

Confirm that your current default version of the kernel is above 4.9

Check the current kernel version

Execute command

```
# master
uname -sr
```

If current version ist not satisfied, you need update kernel

Cilium install

Execute command

```
# master
curl -L --remote-name-all https://github.com/cilium/cilium-cli/releases/latest/download/cilium-linux-amd64.tar.gz
tar -zxvf cilium-linux-amd64.tar.gz
mv cilium /usr/local/bin/
cilium version
cilium install
kubectl get po -A
```

Execute screenshot

```
[root@master ~]# curl -L --remote-name-all https://github.com/cilium/cilium-cli/releases/latest/download/cilium-linux-amd64.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
  0     0    0     0    0     0      0      0 --:--:--  0:00:01 --:--:--     0
100 24.6M  100 24.6M    0     0   829k      0  0:00:30  0:00:30 --:--:--  903k
```

```
[root@master ~]# tar -zxvf cilium-linux-amd64.tar.gz
cilium
[root@master ~]# mv cilium /usr/local/bin/
[root@master ~]# cilium version
cilium-cli: v0.12.11 compiled with go1.19.3 on linux/amd64
cilium image (default): v1.12.2
cilium image (stable): unknown
cilium image (running): v1.12.2
```

```
[root@master ~]# cilium install
ℹ Using Cilium version 1.12.2
🔮 Auto-detected cluster name: kubernetes
🔮 Auto-detected datapath mode: tunnel
🔮 Auto-detected kube-proxy has been installed
ℹ helm template --namespace kube-system cilium cilium/cilium --version 1.12.2 --set cluster.id=0,cluster.name=kubernetes,encryption.nodeEncryption=false,kubeProxyReplacement=disabled,operator.replicas=1,serviceAccounts.cilium.name=cilium,serviceAccounts.operator.name=cilium-operator,tunnel=vxlan
ℹ Storing helm values file in kube-system/cilium-cli-helm-values Secret
🔑 Created CA in secret cilium-ca
🔑 Generating certificates for Hubble...
🚀 Creating Service accounts...
🚀 Creating Cluster roles...
🚀 Creating ConfigMap for Cilium version 1.12.2...
🚀 Creating Agent DaemonSet...
🚀 Creating Operator Deployment...
⌛ Waiting for Cilium to be installed and ready...
✅ Cilium was successfully installed! Run 'cilium status' to view installation health
```

```
[root@master ~]# k get po -A
NAMESPACE     NAME                                       READY   STATUS    RESTARTS   AGE
kube-system   cilium-ntkfm                               1/1     Running   0          70s
kube-system   cilium-operator-9dc4b59f7-mf2sj            1/1     Running   0          70s
kube-system   coredns-6d8c4cb4d-7pg6m                    1/1     Running   0          2m22s
kube-system   coredns-6d8c4cb4d-84tsq                    1/1     Running   0          2m22s
kube-system   etcd-master                                1/1     Running   2          2m40s
kube-system   kube-apiserver-master                      1/1     Running   2          2m39s
kube-system   kube-controller-manager-master             1/1     Running   2          2m37s
kube-system   kube-proxy-2mt6c                           1/1     Running   0          2m22s
kube-system   kube-scheduler-master                      1/1     Running   2          2m38s
```

We can see that the all the pod is ready

If an error occur, you can use cilium uninstall to reset.

5 Initialize workers

The main content is to add worker nodes to the cluster

5.1 Get the join command on the master node

The joining of the worker node needs to initialize the join statement given by the master. If you forget this statement, execute the following statement on the master node machine to get it again.

Execute command

```
# master
kubeadm token create --print-join-command
```

Execute screenshot

```
[root@master ~]# kubeadm token create --print-join-command
kubeadm join 192.168.30.12:6443 --token ixvr7j.ymry4jmzcoililuk --discovery-token-ca-cert-hash sha256:c8cd5ee56ecee1aa2f92ad7856991188d740eb2e443b4c55add98cb
33cb57887
```

5.2 Join the master node

Execute command

```
# worker, change the ip,token and cert-hash to your's
kubeadm join ${YOUR IP} --token ${YOUR TOKEN} --discovery-token-ca-cert-hash ${YOUR CERT HASH}
```

When you have the join statement, copy it and execute it on the worker node

```
[root@worker1 ~]# kubeadm join 192.168.30.12:6443 --token oey5s5.k9xo0ot0be9kzigd --discovery-token-ca-cert-hash sha256:61ba2df
5940423ac7c25bf2c24d83f66f8707db4edc98a20c30733408e894a9a
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Note that if an error occurs and you need to re-init, you need to execute the following statement first to ensure that kubeadm is re-executed normally

```
# worker
kubeadm reset -f
rm -rf ~/.kube/
rm -rf /etc/kubernetes/
rm -rf /var/lib/etcd
rm -rf /var/etcd
```

5.3 Verify the joining of worker nodes

After the worker is added, you can execute the following command on the master machine to confirm the newly added node.

Execute command

```
# master
kubectl get nodes
```

Execute screenshot

```
[root@master ~]# kubectl get nodes
NAME      STATUS   ROLES                  AGE     VERSION
master    Ready    control-plane,master   26m     v1.23.7
worker1   Ready    <none>                 2m53s   v1.23.7
```

6 Install karmada

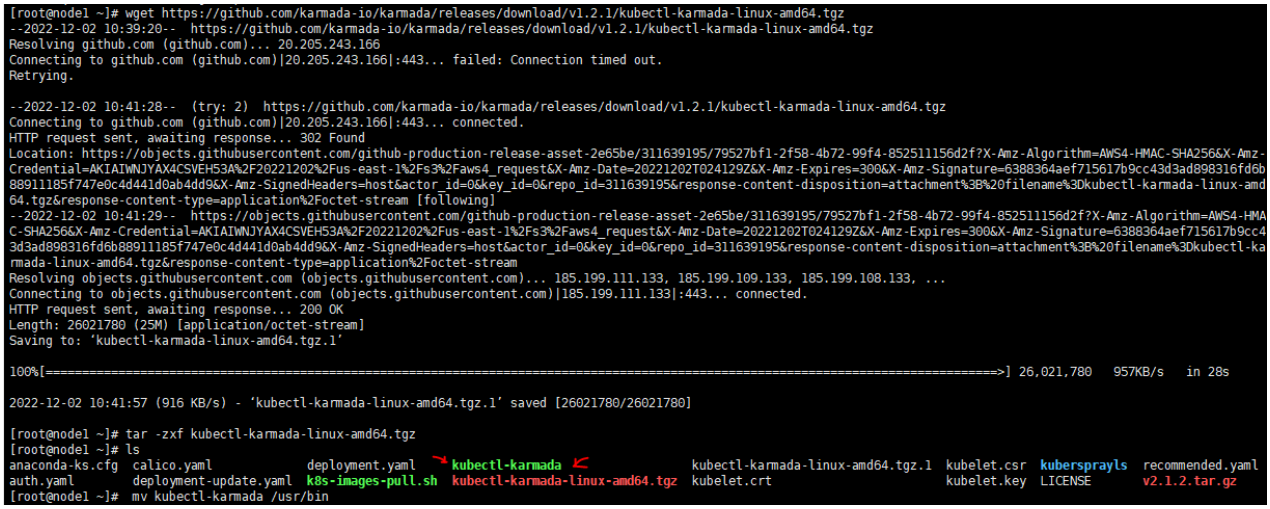
The main content is to install karmada on control plane cluster

6.1 Install the Karmada kubectl plugin

Execute command

```
# master
wget https://github.com/karmada-io/karmada/releases/download/v1.4.0/kubectl-karmada-linux-amd64.tgz
tar -zxf kubectl-karmada-linux-amd64.tgz
mv kubectl-karmada /usr/bin
```

Execute screenshot



```
[root@node1 ~]# wget https://github.com/karmada-io/karmada/releases/download/v1.2.1/kubectl-karmada-linux-amd64.tgz
--2022-12-02 10:39:20--  https://github.com/karmada-io/karmada/releases/download/v1.2.1/kubectl-karmada-linux-amd64.tgz
Resolving github.com (github.com)... 20.205.243.166
Connecting to github.com (github.com)|20.205.243.166|:443... failed: Connection timed out.
Retrying.

--2022-12-02 10:41:28--  (try: 2)  https://github.com/karmada-io/karmada/releases/download/v1.2.1/kubectl-karmada-linux-amd64.tgz
Connecting to github.com (github.com)|20.205.243.166|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/311639195/79527bf1-2f58-4b72-99f4-852511156d2f?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=AKIAIWNJYAX4CSVEH53A%2F20221202%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20221202T024129Z&X-Amz-Expires=300&X-Amz-Signature=6388364aef715617b9cc43d3ad898316fd6b
88911185f747e0c4d441d0ab4dd9&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=311639195&response-content-disposition=attachment%3B%20filename%3Dkubectl-karmada-linux-amd
64.tgz&response-content-type=application%2Foctet-stream [following]
--2022-12-02 10:41:29--  https://objects.githubusercontent.com/github-production-release-asset-2e65be/311639195/79527bf1-2f58-4b72-99f4-852511156d2f?X-Amz-Algorithm=AWS4-HMA
C-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20221202%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20221202T024129Z&X-Amz-Expires=300&X-Amz-Signature=6388364aef715617b9cc4
3d3ad898316fd6b88911185f747e0c4d441d0ab4dd9&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=311639195&response-content-disposition=attachment%3B%20filename%3Dkubectl-ka
rmada-linux-amd64.tgz&response-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.111.133, 185.199.109.133, 185.199.108.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 26021780 (25M) [application/octet-stream]
Saving to: ‘kubectl-karmada-linux-amd64.tgz.1’

100%[===================================================================================================================>] 26,021,780   957KB/s   in 28s

2022-12-02 10:41:57 (916 KB/s) - ‘kubectl-karmada-linux-amd64.tgz.1’ saved [26021780/26021780]

[root@node1 ~]# tar -zxf kubectl-karmada-linux-amd64.tgz
[root@node1 ~]# ls
anaconda-ks.cfg  calico.yaml             deployment.yaml   kubectl-karmada                kubectl-karmada-linux-amd64.tgz.1  kubelet.csr  kubersprayls  recommended.yaml
auth.yaml        deployment-update.yaml  k8s-images-pull.sh  kubectl-karmada-linux-amd64.tgz  kubelet.crt                        kubelet.key  LICENSE       v2.1.2.tar.gz
[root@node1 ~]#  mv kubectl-karmada /usr/bin
```

6.2 Install karamda via karmadactl

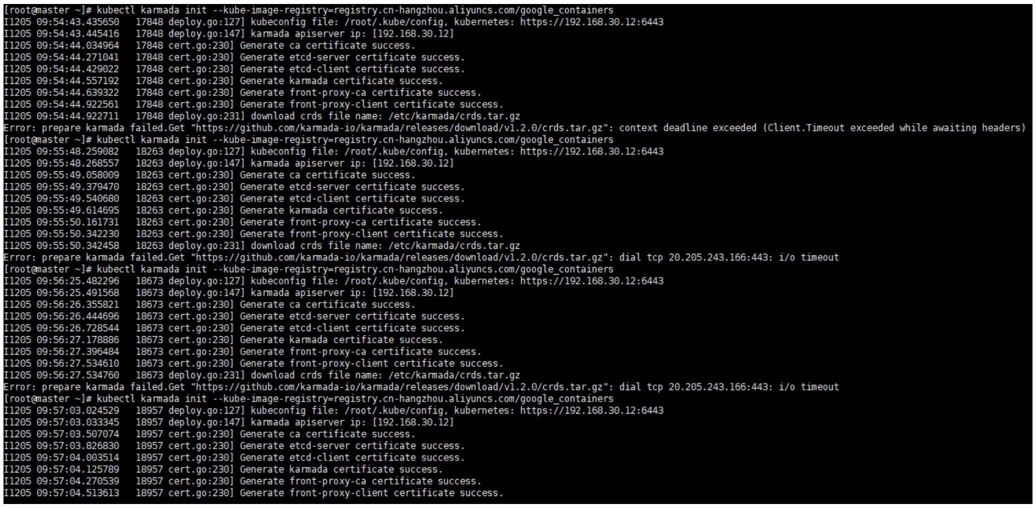
Install karamda via kubectl. China mainland registry mirror can be specified by using kube-image-mirror-country

Execute command

```
kubectl karmada init --kube-image-registry=registry.cn-hangzhou.aliyuncs.com/google_containers
```

Due to network problems, you may need to retry a few times

Execute screenshot



```
[root@master ~]# kubectl karmada init --kube-image-registry=registry.cn-hangzhou.aliyuncs.com/google_containers
I1205 09:54:43.435650   17848 deploy.go:127] kubeconfig file: /root/.kube/config, kubernetes: https://192.168.30.12:6443
I1205 09:54:43.445416   17848 deploy.go:147] karmada apiserver ip: [192.168.30.12]
I1205 09:54:44.034964   17848 cert.go:230] Generate ca certificate success.
I1205 09:54:44.271041   17848 cert.go:230] Generate etcd-server certificate success.
I1205 09:54:44.429022   17848 cert.go:230] Generate etcd-client certificate success.
I1205 09:54:44.557192   17848 cert.go:230] Generate karmada certificate success.
I1205 09:54:44.639322   17848 cert.go:230] Generate front-proxy-ca certificate success.
I1205 09:54:44.922561   17848 cert.go:230] Generate front-proxy-client certificate success.
I1205 09:54:44.922711   17848 deploy.go:231] download crds file name: /etc/karmada/crds.tar.gz
Error: prepare karmada failed.Get "https://github.com/karmada-io/karmada/releases/download/v1.2.0/crds.tar.gz": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
[root@master ~]# kubectl karmada init --kube-image-registry=registry.cn-hangzhou.aliyuncs.com/google_containers
I1205 09:55:48.259082   18263 deploy.go:127] kubeconfig file: /root/.kube/config, kubernetes: https://192.168.30.12:6443
I1205 09:55:48.268557   18263 deploy.go:147] karmada apiserver ip: [192.168.30.12]
I1205 09:55:49.058009   18263 cert.go:230] Generate ca certificate success.
I1205 09:55:49.379470   18263 cert.go:230] Generate etcd-server certificate success.
I1205 09:55:49.540680   18263 cert.go:230] Generate etcd-client certificate success.
I1205 09:55:49.614695   18263 cert.go:230] Generate karmada certificate success.
I1205 09:55:50.161731   18263 cert.go:230] Generate front-proxy-ca certificate success.
I1205 09:55:50.342230   18263 cert.go:230] Generate front-proxy-client certificate success.
I1205 09:55:50.342458   18263 deploy.go:231] download crds file name: /etc/karmada/crds.tar.gz
Error: prepare karmada failed.Get "https://github.com/karmada-io/karmada/releases/download/v1.2.0/crds.tar.gz": dial tcp 20.205.243.166:443: i/o timeout
[root@master ~]# kubectl karmada init --kube-image-registry=registry.cn-hangzhou.aliyuncs.com/google_containers
I1205 09:56:25.482296   18673 deploy.go:127] kubeconfig file: /root/.kube/config, kubernetes: https://192.168.30.12:6443
I1205 09:56:25.491568   18673 deploy.go:147] karmada apiserver ip: [192.168.30.12]
I1205 09:56:26.355821   18673 cert.go:230] Generate ca certificate success.
I1205 09:56:26.444696   18673 cert.go:230] Generate etcd-server certificate success.
I1205 09:56:26.728544   18673 cert.go:230] Generate etcd-client certificate success.
I1205 09:56:27.178886   18673 cert.go:230] Generate karmada certificate success.
I1205 09:56:27.396484   18673 cert.go:230] Generate front-proxy-ca certificate success.
I1205 09:56:27.534610   18673 cert.go:230] Generate front-proxy-client certificate success.
I1205 09:56:27.534760   18673 deploy.go:231] download crds file name: /etc/karmada/crds.tar.gz
Error: prepare karmada failed.Get "https://github.com/karmada-io/karmada/releases/download/v1.2.0/crds.tar.gz": dial tcp 20.205.243.166:443: i/o timeout
[root@master ~]# kubectl karmada init --kube-image-registry=registry.cn-hangzhou.aliyuncs.com/google_containers
I1205 09:57:03.024529   18957 deploy.go:127] kubeconfig file: /root/.kube/config, kubernetes: https://192.168.30.12:6443
I1205 09:57:03.033345   18957 deploy.go:147] karmada apiserver ip: [192.168.30.12]
I1205 09:57:03.507074   18957 cert.go:230] Generate ca certificate success.
I1205 09:57:03.826830   18957 cert.go:230] Generate etcd-server certificate success.
I1205 09:57:04.003514   18957 cert.go:230] Generate etcd-client certificate success.
I1205 09:57:04.125789   18957 cert.go:230] Generate karmada certificate success.
I1205 09:57:04.270539   18957 cert.go:230] Generate front-proxy-ca certificate success.
I1205 09:57:04.513613   18957 cert.go:230] Generate front-proxy-client certificate success.
```

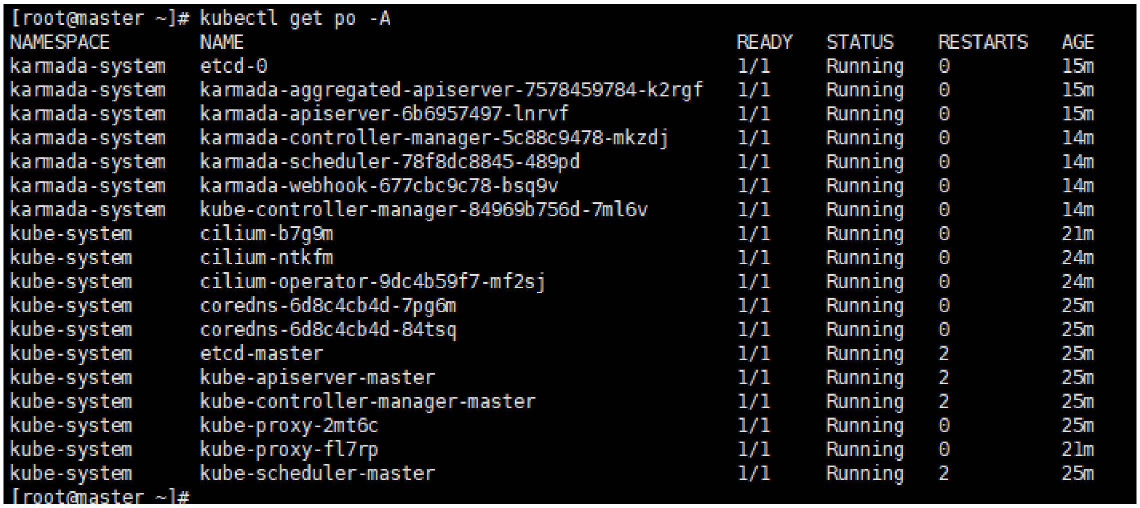
```
I1205 10:01:58.277628   20880 check.go:49] pod: karmada-webhook-677cbc9c78-bsq9v is ready. status: Running

Karmada is installed successfully.

Register Kubernetes cluster to Karmada control plane.

Register cluster with 'Push' mode

Step 1: Use kubectl karmada join to register the cluster to Karmada control panel. --cluster-kubeconfig is members kubeconfig.
(In karmada)~# MEMBER_CLUSTER_NAME=`cat ~/.kube/config  | grep current-context | sed 's/: /\n/g'| sed '1d'`
(In karmada)~# kubectl karmada --kubeconfig /etc/karmada/karmada-apiserver.config  join ${MEMBER_CLUSTER_NAME} --cluster-kubeconfig=$HOME/.kube/config

Step 2: Show members of karmada
(In karmada)~# kubectl  --kubeconfig /etc/karmada/karmada-apiserver.config get clusters


Register cluster with 'Pull' mode

Step 1:  Send karmada kubeconfig and karmada-agent.yaml to member kubernetes
(In karmada)~# scp /etc/karmada/karmada-apiserver.config /etc/karmada/karmada-agent.yaml {member kubernetes}:~

Step 2:  Create karmada kubeconfig secret
 Notice:
   Cross-network, need to change the config server address.
(In member kubernetes)~#  kubectl create ns karmada-system
(In member kubernetes)~#  kubectl create secret generic karmada-kubeconfig --from-file=karmada-kubeconfig=/root/karmada-apiserver.config  -n karmada-system

Step 3: Create karmada agent
(In member kubernetes)~#  MEMBER_CLUSTER_NAME="demo"
(In member kubernetes)~#  sed -i "s/{member_cluster_name}/${MEMBER_CLUSTER_NAME}/g" karmada-agent.yaml
(In member kubernetes)~#  kubectl apply -f karmada-agent.yaml

Step 4: Show members of karmada
(In karmada)~# kubectl  --kubeconfig /etc/karmada/karmada-apiserver.config get clusters
```

Check all your pod is READY

Execute command

```
kubectl get po -A
```

Execute screenshot



```
[root@master ~]# kubectl get po -A
NAMESPACE        NAME                                          READY   STATUS    RESTARTS   AGE
karmada-system   etcd-0                                        1/1     Running   0          15m
karmada-system   karmada-aggregated-apiserver-7578459784-k2rgf 1/1     Running   0          15m
karmada-system   karmada-apiserver-6b6957497-lnrvf             1/1     Running   0          15m
karmada-system   karmada-controller-manager-5c88c9478-mkzdj    1/1     Running   0          14m
karmada-system   karmada-scheduler-78f8dc8845-489pd            1/1     Running   0          14m
karmada-system   karmada-webhook-677cbc9c78-bsq9v              1/1     Running   0          14m
karmada-system   kube-controller-manager-84969b756d-7ml6v      1/1     Running   0          14m
kube-system      cilium-b7g9m                                  1/1     Running   0          21m
kube-system      cilium-ntkfm                                  1/1     Running   0          24m
kube-system      cilium-operator-9dc4b59f7-mf2sj               1/1     Running   0          24m
kube-system      coredns-6d8c4cb4d-7pg6m                       1/1     Running   0          25m
kube-system      coredns-6d8c4cb4d-84tsq                       1/1     Running   0          25m
kube-system      etcd-master                                   1/1     Running   2          25m
kube-system      kube-apiserver-master                         1/1     Running   2          25m
kube-system      kube-controller-manager-master                1/1     Running   2          25m
kube-system      kube-proxy-2mt6c                              1/1     Running   0          25m
kube-system      kube-proxy-fl7rp                              1/1     Running   0          21m
kube-system      kube-scheduler-master                         1/1     Running   2          25m
[root@master ~]#
```

7 Propagate a deployment by Karmada

Before propagating a deployment, make sure the worker cluster is already working properly And get the latest config currently running

In the following steps, we are going to propagate a deployment by Karmada. We use the installation of nginx as an example

7.1 Join a worker/member cluster to karmada control plane

Here we add the working node cluster through push mode

It is worth noting that /root/.kube/config is Kubernetes host config and the /etc/karmada/karmada-apiserver.config is karmada-apiserver config

Execute command

```
kubectl karmada --kubeconfig /etc/karmada/karmada-apiserver.config join ${YOUR MEMBER NAME} --cluster-kubeconfig=${YOUR MEMBER CONFIG PATH} --cluster-context=${YOUR CLUSTER CONTEXT}
```

```
[root@master nginx]# kubectl karmada --kubeconfig /etc/karmada/karmada-apiserver.config join member1 --cluster-kubeconfig=/root/.kube/member1-config --cluster-context=kubernetes-admin@k
ubernetes
cluster(member1) is joined successfully
```

Here is example command for your information: kubectl karmada --kubeconfig /etc/karmada/karmada-apiserver.config join member1 --cluster-kubeconfig= /root/.kube/member1-config --cluster-context=kubernetes-admin@kubernetes

- `--kubeconfig  specifies the Karmada's kubeconfig file and the CLI`
- --cluster-kubeconfig `specifies the member's config. Generally, it can be obtained from the worker cluster in " /root/.kube/config"`
- `--cluster-context the value of current-context from --cluster-kubeconfig`

If you want unjoin the member cluster, just change the join to unjoin: kubectl karmada --kubeconfig /etc/karmada/karmada-apiserver.config unjoin member2 --cluster-kubeconfig=/root/.kube/192.168.30.2_config --cluster-context=kubernetes-admin@kubernetes

check the members of karmada

Execute command

```
kubectl --kubeconfig /etc/karmada/karmada-apiserver.config get clusters
```

```
[root@master .kube]# kubectl --kubeconfig /etc/karmada/karmada-apiserver.config get clusters
NAME      VERSION   MODE   READY   AGE
member1   v1.23.5   Push   True    3m24s
member2   v1.23.7   Push   True    109s
```

7.2 Create nginx deployment in Karmada

deployment.yaml are obtained through here https://github.com/karmada-io/karmada/tree/master/samples/nginx

Execute command

```
kubectl create -f /root/sample/nginx/deployment.yaml --kubeconfig /etc/karmada/karmada-apiserver.config
kubectl get deployment  --kubeconfig /etc/karmada/karmada-apiserver.config
```

```
[root@master nginx]# kubectl create -f /root/sample/nginx/deployment.yaml
deployment.apps/nginx created
```

```
[root@master nginx]#  kubectl get deployment  --kubeconfig /etc/karmada/karmada-apiserver.config
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   2/2     2            2           21m
```

7.3 Create PropagationPolicy that will propagate nginx to member cluster

propagationpolicy.yaml are obtained through here https://github.com/karmada-io/karmada/tree/master/samples/nginx

Execute command

```
kubectl create -f /root/sample/nginx/propagationpolicy.yaml --kubeconfig /etc/karmada/karmada-apiserver.config
```

```
[root@master nginx]# kubectl create -f /root/sample/nginx/propagationpolicy.yaml --kubeconfig /etc/karmada/karmada-apiserver.config
propagationpolicy.policy.karmada.io/nginx-propagation created
```

7.4 Check the deployment status from Karmada

```
kubectl get po  --kubeconfig /root/.kube/member1-config
kubectl get po  --kubeconfig /root/.kube/member2-config
```

```
[root@master nginx]# k get po  --kubeconfig /root/.kube/member1-config
NAME                     READY   STATUS    RESTARTS   AGE
nginx-85b98978db-84lns   1/1     Running   0          3m45s
[root@master nginx]# k get po  --kubeconfig /root/.kube/member2-config
NAME                     READY   STATUS    RESTARTS   AGE
nginx-85b98978db-l2n2w   1/1     Running   0          3m51s
```

Reference

https://lazytoki.cn/index.php/archives/4/

https://www.cnblogs.com/renshengdezheli/p/16686769.html#%E5%9B%9B%E5%AE%89%E8%A3%85%E9%83%A8%E7%BD%B2kubernetes%E9%9B%86%E7%BE%A4

https://www.cnblogs.com/renshengdezheli/p/16686769.html#%E5%9B%9B%E5%AE%89%E8%A3%85%E9%83%A8%E7%BD%B2kubernetes%E9%9B%86%E7%BE%A4

https://developer.aliyun.com/article/931926

https://ljchen.net/2018/10/23/%E5%9F%BA%E4%BA%8E%E9%98%BF%E9%87%8C%E4%BA%91%E9%95%9C%E5%83%8F%E7%AB%99%E5%AE%89%E8%A3%85kubernetes/

https://zhuanlan.zhihu.com/p/368879345

https://docs.docker.com/engine/install/centos/

https://kubernetes.io/zh-cn/docs/setup/production-environment/container-runtimes/

https://karmada.io/docs/installation/install-kubectl-karmada

https://karmada.io/docs/installation/

https://docs.cilium.io/en/stable/gettingstarted/k8s-install-kubeadm/

https://karmada.io/docs/get-started/nginx-example

Installation worker cluster

0 Environmental

Three centos machines are required, one as the master node and others as the worker node. The installed k8s version is 1.23.5

1 Preparation

The main content is to prepare the basic environment to ensure the normal execution of subsequent operations.

2 Install docker

The main content is to install docker, and configure the cgroup driver of docker as systemd, confirm the driver

2.1 uninstall old docker

Execute command

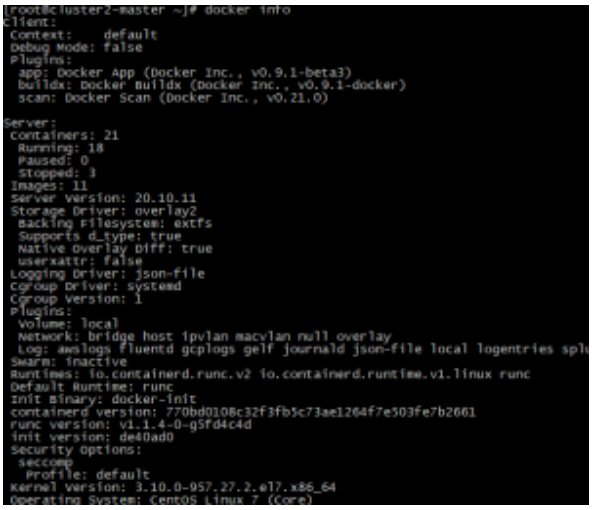
```
 yum -y remove docker docker-client docker-client-latest docker-common docker-latest docker-latest-logrotate docker-logrotate docker-engine docker-ce docker-ce-cli
```

2.2 install docker

Execute command

```
yum -y install docker-ce-20.10.11
```

Execute screenshot



2.3 Set docker to boot and confirm docker status

Execute command

```
systemctl enable docker
systemctl start docker
systemctl status docker
```

2.4 Configure the driver of docker's cgroup

The driver of docker's cgroup needs to be consistent with that of kubelet. It is recommended to set it to systemd. At this time, the execution of subsequent commands may be stuck. Wait for a few minutes or restart, and then try again

Execute command

```
docker info | grep -i cgroup
```

Check the current configuration, if it is the system in the figure below, skip the follow-up and go directly to the third section

If it is cgroupfs, add the following statement

```
vim /etc/docker/daemon.json
# add this
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "insecure-registries": ["0.0.0.0/0"]
}
```

Restart to check for changes

```
systemctl daemon-reload && systemctl restart docker
docker info | grep -i cgroup
```

3 Install k8s

The main content is to install the 1.23.5 version of the component kubeadm kubectl kubelet

3.1 Uninstall old k8s

If it is the version inconsistent, you need to uninstall.

```
kubeadm reset -f
modprobe -r ipip
lsmod
rm -rf ~/.kube/
rm -rf /etc/kubernetes/
rm -rf /etc/systemd/system/kubelet.service.d
rm -rf /etc/systemd/system/kubelet.service
rm -fr /usr/lib/systemd/system/kubelet.service
rm -fr /etc/systemd/system/multi-user.target.wants/kubelet.service
rm -rf /usr/bin/kube*
rm -rf /etc/cni
rm -rf /opt/cni
rm -rf /var/lib/etcd
rm -rf /var/etcd
yum remove kubeadm
yum remove kubelet
yum remove kube*
```

3.2 Download the new yum source

```
rm -rf /etc/yum.repos.d/* ;
wget ftp://ftp.rhce.cc/k8s/* -P /etc/yum.repos.d/ ls /etc/yum.repos.d/
```

3.2 Remove kubeadm kubectl kubelet

If it is the version inconsistent, you need to uninstall it through yum remove.

```
yum -y remove kubelet kubeadm kubectl
```

3.3 Install kubelet kubeadm kubectl version 1.23.5

```
yum -y install kubelet-1.23.5 kubeadm-1.23.5 kubectl-1.23.5
```

3.3 verify installation

```
kubelet --version
kubeadm version
kubectl version
```

3.5 Set kubelet to boot

```
systemctl daemon-reload
systemctl start kubelet
systemctl enable kubelet
```

4 Initialize master

The main content is to pull the image of version 1.23.5, initialize the master node.

4.1 Edit init-config.yaml

```
# init-config.yaml
apiVersion: kubeadm.k8s.io/v1beta3
bootstrapTokens:
- groups:
  - system:bootstrappers:kubeadm:default-node-token
  token: abcdef.0123456789abcdef
  ttl: 24h0m0s
  usages:
  - signing
  - authentication
kind: InitConfiguration
localAPIEndpoint:
  advertiseAddress: 192.168.30.22
  bindPort: 6443
nodeRegistration:
  criSocket: /var/run/dockershim.sock
  imagePullPolicy: IfNotPresent
  name: cluster1-master
  taints: null
---
apiServer:
  timeoutForControlPlane: 4m0s
apiVersion: kubeadm.k8s.io/v1beta3
certificatesDir: /etc/kubernetes/pki
clusterName: kubernetes
controllerManager: {}
dns: {}
etcd:
  local:
    dataDir: /var/lib/etcd
imageRepository: registry.aliyuncs.com/google_containers
kind: ClusterConfiguration
kubernetesVersion: 1.23.5
networking:
  dnsDomain: cluster.local
  serviceSubnet: 10.1.0.0/12
scheduler: {}
```

4.2 Pull the k8s image

```
kubeadm config images list --config=init-config.yaml
# images list
registry.aliyuncs.com/google_containers/kube-apiserver:v1.23.0
registry.aliyuncs.com/google_containers/kube-controller-manager:v1.23.0
registry.aliyuncs.com/google_containers/kube-scheduler:v1.23.0
registry.aliyuncs.com/google_containers/kube-proxy:v1.23.0
registry.aliyuncs.com/google_containers/pause:3.6
registry.aliyuncs.com/google_containers/etcd:3.5.1-0
registry.aliyuncs.com/google_containers/coredns:v1.8.6

# pull images
kubeadm config images pull --config=init-config.yaml
```

Please make sure that the above images have been pulled down

4.2 init master

```
kubeadm init --apiserver-advertise-address=192.168.30.22 --apiserver-bind-port=6443 --pod-network-cidr=10.
100.0.0/16  --service-cidr=10.1.0.0/12 --kubernetes-version=1.23.5 --image-repository registry.aliyuncs.com
/google_containers
```

You can see that the prompt initialization is successful. The prompt executes the following command to use kubectl normally

```
# master
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

5 Initialize workers

The main content is to add worker nodes to the cluster

5.1 Join the master node

```
# worker
kubeadm join 192.168.30.22:6443 --token zb4ydd.afsdnt1k58ow3z8x \
        --discovery-token-ca-cert-hash sha256:a73132eb5e9a911f24697e8550333ddc20d3ae641d35ba17c09580730b607080
```

Note that if an error occurs and you need to re-init, you need to execute the following statement first to ensure that kubeadm is re-executed normally

```
# worker
kubeadm reset -f
rm -rf ~/.kube/
rm -rf /etc/kubernetes/
rm -rf /var/lib/etcd
rm -rf /var/etcd
```

5.3 Verify worker nodes

After the worker is added, you can execute the following command on the master machine to confirm .

```
# master
kubectl get nodes
```

```
Execute screenshot
```

```
[root@cluster1-master ~]# kubectl get nodes
NAME              STATUS   ROLES                  AGE    VERSION
cluster1-master   Ready    control-plane,master   3d6h   v1.23.5
cluster1-node1    Ready    <none>                 3d6h   v1.23.5
cluster1-node2    Ready    <none>                 3d6h   v1.23.5
[root@cluster1-master ~]#
```

5.4 Configure the calico network plugin

Configure the calico network plug-in to make it work. Create the calico.yaml file.

```
# master
curl https://docs.projectcalico.org/v3.11/manifests/calico.yaml -O
vi calico.yaml
#
kubectl apply -f calico.yaml
# loopuntil all the pod's status is running
kubectl get po -A
```

Execute screenshot

```
[root@cluster1-master ~]# kubectl get po -A
NAMESPACE     NAME                                       READY   STATUS    RESTARTS
kube-system   calico-kube-controllers-7f76d48f74-4wr2g   1/1     Running   0
kube-system   calico-node-gmscb                          1/1     Running   0
kube-system   calico-node-tfczc                          1/1     Running   0
kube-system   calico-node-tsvpn                          1/1     Running   0
kube-system   coredns-6d8c4cb4d-nsrz2                    1/1     Running   0
kube-system   coredns-6d8c4cb4d-t5d6s                    1/1     Running   0
kube-system   etcd-cluster1-master                       1/1     Running   0
kube-system   kube-apiserver-cluster1-master             1/1     Running   0
kube-system   kube-controller-manager-cluster1-master    1/1     Running   5
kube-system   kube-proxy-4dfw5                           1/1     Running   0
kube-system   kube-proxy-ctnvs                           1/1     Running   0
kube-system   kube-proxy-t7cjl                           1/1     Running   0
kube-system   kube-scheduler-cluster1-master             1/1     Running   6
```

Deploy Application

0 Environmental

The k8s version is 1.23.5.

1 Preparation

The main content is to prepare the Application image.

```
192.168.30.20:5000/migu/sentiment:latest
```

2 Deployment

The main content is to install sentiment application.

2.1 namespace

```
# namespace.yaml
apiVersion: v1
kind: Namespace
metadata:
  name: migu
```

Execute command

```
kubectl create -f namespace.yaml
```

2.2 image pull secret

Add Harbor Image Registry Pull Secret to Kubernetes.

```
#harborsecret
kubectl create secret generic harborsecret \
    --from-file=.dockerconfigjson=/root/.docker/config.json \
    --type=kubernetes.io/dockerconfigjson \
    -n migu
```

2.3 deployment

```
#deploy.yaml
kind: Deployment
apiVersion: apps/v1
metadata:
  labels:
    app: sentiment
  name: migu-sentiment
  namespace: migu
spec:
  replicas: 2
  selector:
    matchLabels:
      app: sentiment
  template:
    metadata:
      labels:
        app: sentiment
    spec:
      imagePullSecrets:
      - name: harborsecret
      containers:
      - name: sentiment-container
        image: 192.168.30.20:5000/migu/sentiment:latest
        imagePullPolicy: IfNotPresent
        imagePullPolicy: Always
        ports:
        - containerPort: 9600
          protocol: TCP
          name: http
        resources:
          limits:
            cpu: 2
            memory: 4G
          requests:
            cpu: 2
            memory: 4G
```

Execute command

```
kubectl create -f deploy.yaml
```

Execute screenshot

```
[root@cluster1-master migu]#
[root@cluster1-master migu]# kubectl get deploy -n m
NAME              READY   UP-TO-DATE   AVAILABLE   A
migu-sentiment    2/2     2            2           3
[root@cluster1-master migu]# kubectl get pod -n migu
NAME                                READY   STATUS
migu-sentiment-6f94c899fc-47g8d     1/1     Running
migu-sentiment-6f94c899fc-zjlv7     1/1     Running
```

2.3 service

```
#service.yaml
apiVersion: v1
kind: Service
metadata:
  labels:
   app: sentiment
  name: sentiment
  namespace: migu
spec:
  ports:
  - port: 9600
    protocol: TCP
    targetPort: 9600
    nodePort: 30960
  selector:
    app: sentiment
  type: NodePort
```

Execute command

```
kubectl create -f service.yaml
```

Execute screenshot

```
[root@cluster1-master migu]#
[root@cluster1-master migu]# kubectl get svc -n migu
NAME        TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)
sentiment   NodePort   10.15.113.78   <none>        9600:30960/TCP
[root@cluster1-master migu]#
```

2.4 test service

```
## request
curl http://192.168.30.20:30960/health

##response
{"status": "UP"}
```

Execute screenshot

Installation High-Level Overview

- Bare Metal Deployment Guide
  - Install Bare Metal Jump Host : N/A
  - Creating a Node Inventory File: N/A
  - Creating the Settings Files: N/A
  - Running: N/A

- Virtual Deployment Guide
  - Standard Deployment Overview N/A
  - Snapshot Deployment OverviewN/A
  - Special Requirements for Virtual Deployments
  - Install Jump Host: N/A
    - Verifying the Setup - VMsN/A
- Upstream Deployment Guide
  - Upstream Deployment Key Features N/A
  - Special Requirements for Upstream Deployments N/A
  - Scenarios and Deploy Settings for Upstream Deployments N/A
  - Including Upstream Patches with DeploymentN/A
  - Running N/A
  - Interacting with Containerized Overcloud N/A

Developer Guide and Troubleshooting

- Utilization of Images

N/A

- Post-deployment Configuration

N/A

- Debugging Failures

N/A

- Reporting a Bug

N/A

Uninstall Guide

Troubleshooting

1. Network problem: the working cluster uses the default communication mode of calico, and the access between nodes is blocked; After many attempts, calico vxlan is feasible and flannel is feasible at present;

2. Disaster recovery scenario scheduling, test scenario 2, requires the karmada control plane to install the deschedule component;

Maintenance

- Blue Print Package Maintenance
  - Software maintenance N/A
  - Hardware maintenanceN/A
- Blue Print Deployment Maintenance (N/A)

Frequently Asked Questions

N/A

License

N/A

References

N/A

Definitions, acronyms and abbreviations

N/A