Kubernetes (v1.18.3) Deployment

Prerequisites

For both master and edge node:

Network

Make sure all you nodes can have access to each other.

Installing runtime (docker), kubeadm, kubelet, kubectl

Note: the version installed here must meet the version of the Kubernetes control plane that will be installed by `kubeadm`.

Version used:

- Docker: 19.03.12
- kubectl / kubelet / kubeadm: v1.19.0

To install `kubeadm`, `kubelet`, `kubectl`,

```
# make apt support ssl
apt-get update && apt-get install -y apt-transport-https
# get gpg
curl https://mirrors.aliyun.com/kubernetes/apt/doc/apt-key.gpg | apt-key add -
# add mirror source of k8s
cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb https://mirrors.aliyun.com/kubernetes/apt/ kubernetes-xenial main
EOF
# update
apt-get update
apt-get install -y kubelet kubeadm kubectl
```

Check cgroup driver

Docker and kubelet need the same cgroup driver. To modify docker's, change in file `/etc/docker/daemon.json` as `"exec-opts": ["native.cgroupdriver=cgroupfs"],` and check by `docker info | grep Cgroup`. Then restart docker `systemctl restart docker`

For kubelet, modify `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf`, add

```
`Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf --cgroup-driver=cgroupfs"`
```

and restart kubelet

```
systemctl daemon-reload
systemctl restart kubelet
```

Turn swap off

Temporary turn off: run `swapoff -a`, disable immediately

Initialization on master

This part only for master node.

Pull Image

Before pulling image, you should check what kinds of images that you should install by runing `kubectl config images list`.

As the images needed could not be pulled directly, first need to pull using mirror source, with specified version, which correspond to kubernetes version v1.18.3.

```
docker pull mirrorgooglecontainers/kube-apiserver:v1.13.2
docker pull mirrorgooglecontainers/kube-controller-manager:v1.13.2
docker pull mirrorgooglecontainers/kube-scheduler:v1.13.2
docker pull mirrorgooglecontainers/kube-proxy:v1.13.2
docker pull mirrorgooglecontainers/pause:3.1
docker pull mirrorgooglecontainers/etcd:3.2.24
docker pull coredns/coredns:1.2.6
```

Then tag the corresponding images

```
docker tag mirrorgooglecontainers/kube-apiserver:v1.13.2 k8s.gcr.io/kube-apiserver:v1.13.2
docker tag mirrorgooglecontainers/kube-proxy:v1.13.2 k8s.gcr.io/kube-proxy:v1.13.2
docker tag mirrorgooglecontainerso/kube-controller-manager:v1.13.2 k8s.gcr.io/kube-controller-manager:v1.13.2
docker tag mirrorgooglecontainers/kube-scheduler:v1.13.2 k8s.gcr.io/kube-scheduler:v1.13.2
docker tag mirrorgooglecontainers/coredns:1.2.6 k8s.gcr.io/coredns:1.2.6
docker tag mirrorgooglecontainers/etcd:3.2.24 k8s.gcr.io/etcd:3.2.24
docker tag mirrorgooglecontainers/pause:3.1 k8s.gcr.io/pause:3.1
```

- Troubleshooting:
    - If you use a different version of k8s, you can find mirror images on Dockerhub and then tag them as `k8s.gcr.io/`

kubeadm init

To apply pod network `flannel` to the cluster, `--pod-network-cidr=10.244.0.0/16` is needed.

run on master

```
sudo kubeadm init \
--pod-network-cidr=10.244.0.0/16 \
--kubernetes-version v1.18.3
```

When see

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

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

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 192.168.0.68:6443 --token 88mamu.oed0ryk6q1ln99im \
    --discovery-token-ca-cert-hash sha256:6d3ad6f127b16420025bba5c91e409d05b332d668e3a8dad4c2a7de6d2f5752f
```

The initialization succeed. Remember save the output following `kubeadm join`.

- Troubleshooting:
    - `docker` or `kubelet` is not running. Check status: `systemctl status [docker|kubelet]`. To start: use `systemctl start [docker|kubelet]`.
    - To reinitiate it, first run `kubeadm reset` and the folowing command

      ```
      systemctl stop kubelet
      systemctl stop docker
      rm -rf /var/lib/cni/
      rm -rf /var/lib/kubelet/*
      rm -rf /etc/cni/
      ifconfig cni0 down
      ifconfig flannel.1 down
      ifconfig docker0 down
      ip link delete cni0
      ip link delete flannel.1
      systemctl start docker
      systemctl start kubelet
      ```

      othereise reinitialization will not work.

kubectl configration

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

Use `kubectl get nodes` to check node status. Due to no network addons installed, `master` node should be `NotReady`.

- Troubleshooting:
    - if there is no such file `/etc/kubernetes/admin.conf` on edge node, you can use `scp` to copy the file on your cloud node to edge node.

- Use `kubectl get pods -n kube-system` to check pods' status. All should show running. To see the error in detail, use `kubectl describe node [pod's name] -n kube-system`. The error might caused by the image pulling and the name could be seen from `describe` command. Download it and tranfer to the node.

flannel configration

Run

```
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/a70459be0084506e4ec919aa1c114638878db11b/Documentation/kube-flannel.yml
```

When it shows

```
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.extensions/kube-flannel-ds-amd64 created
daemonset.extensions/kube-flannel-ds-arm64 created
daemonset.extensions/kube-flannel-ds-arm created
daemonset.extensions/kube-flannel-ds-ppc64le created
daemonset.extensions/kube-flannel-ds-s390x created
```

Then completed.

Also, you could check the pod status through `kubectl get pods -n kube-system`.

- Troubleshooting:
    - if you see some pod showing `ErrImagePulling`, you can always choose to pull those corresponding images by yourself. Make sure tag those images.

Join the node

Node should meet the prerequisites mentioned at first. Run the output saved before

```
kubeadm join 192.168.0.68:6443 --token 88mamu.oed0ryk6q1ln99im \
    --discovery-token-ca-cert-hash
sha256:6d3ad6f127b16420025bba5c91e409d05b332d668e3a8dad4c2a7de6d2f5752f
```

when output shows

```
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 master to see this node join the cluster.
```

then node has joined the cluster.

After that, you should perform the same action as we do for master node or cloud side.

1. **kubectl configration**.

2. **flannel configration**.

Then you can run `kubectl get nodes` and you should see two nodes with status `Ready`.

- Troubleshooting
    - Token valid time: token will only be valid for next 24 hours. If after 24 hours, still need to join nodes, run

```
kubeadm token create    # generate new token
kubeadm token list  # replace the new token into the join command
```

    - `docker` or `kubelet` is not running. Check status: `systemctl status [docker|kubelet]`. To start: use `systemctl start [docker|kubelet]`.
    - You can always choose to reset your cluster by `kubeadm reset`.
    - After execute `kubectl get pods --all-namespaces`, if you see some pod showing `ErrImagePulling`, you can always choose to pull those corresponding images by yourself. Make sure tag those images.

If all you pods are running correctly now, you have successfully installed k8s.

Kubefate(v.1.4.4) Deployment

nginx-ingress-controller installation

You should firstly install ingress controller before installing kubefate, here we choose to use nginx.

Simply we can just run the following command,

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v0.35.0/deploy/static/provider/cloud/deploy.yaml
```

- Troubleshooting:
    - Use `kubectl get pods -n ingress-nginx` to check if every pod is running successfully.
    - If some pod shows `ErrImagePulling`, you can always choose to pull the corresponding image from dockerhub and make sure tag them correctly. By using the command `kubectl describe pod podID -n ingress-nginx`, you can see which image you should pull.

kubefate installation

Create the installation folder of kubefate.

```
mkdir ~/kube-fate && cd ~/kube-fate
```

Then install the latest release file of kubfate, here is the link , and then uncompress those files into `~/kube-fate`.

```
tar -xf kubefate-k8s-1.4.4.tar.gz && ls
```

```
cluster-serving.yaml  config.yaml      fate-9999.yaml  kubefate-k8s-
1.4.4.tar.gz  rbac-config.yaml
cluster.yaml          fate-10000.yaml  kubefate        kubefate.yaml
```

Firstly, apply `rbac-config.yaml`.

```
kubectl apply -f rbac-config.yaml
```

Secondly, apply `kubefate.yaml`.

```
kubectl apply -f kubefate.yaml
```

Also, cp `kubefate` into `/usr/bin`.

```
cp kubefate /usr/bin
```

Also we should add one line in `/etc/hosts`

```
IP'address_of_pod_nginx-ingress-controller kubefate.net
```

where `IP'address_of_pod_nginx-ingress-controller` can be checked by

```
kubectl get pods -n ingress-nginx -o wide |grep controller
```

```
ingress-nginx-controller-77f75945d7-zr7kv   1/1     Running     4
 135m   10.244.0.81   master   <none>           <none>
```

Then, `10.244.0.81` should be the IP's address.

To check if `kubefate` is installed successfully, we can use following command.

```
kubefate version
```

If installation is successful, it should shows

```
* kubefate service version=v1.1.0
* kubefate commandLine version=v1.1.0
```

Cluster Installation Example

```
cd ~/kube-fate
cp cluster.yaml fate-9999.yaml
cp cluster.yaml fate-10000.yaml
```

Create the corresponding namespace in k8s.

```
kubectl namespace create fate-9999
kubectl namespace create fate-10000
```

Then, we should modify the file `fate-9999.yaml` a little bit.

1. `registry: "hub.c.163.com/federatedai"`.
2. delete `Exchange` part.
3. Change the IP address and ports in the file.

Do almost the same thing for `fate-10000.yaml`. For detailed configuration, you can refer to a chinese blog.

Then,

```
kubefate cluster install ./fate-9999.yaml
kubefate cluster install ./fate-10000.yaml
```

Now check your installation by

```
kubefate cluster ls
```

If it shows following message, then your cluster have been installed successfully.

```
UUID                                    NAME            NAMESPACE
REVISION        STATUS  CHART   ChartVERSION    AGE
46d5bce2-5834-4bac-ba3c-937c7782ada6    fate-9999       fate-9999       1
          Running fate    v1.4.4          133m
cc260af3-4c08-4d4a-ba72-893ec0e3fd44    fate-10000      fate-10000      1
          Running fate    v1.4.4          132m
```