K8s installation in one ECS server of Aliyun(V1.11.2)

K8s Installation

Install on single one server, and it acts as Master and Worker node as well.

Environment
OS: CentOS Linux release 7.4 (Core)

Kubernetes: v1.11.2

Preparation
1. Add entries into /etc/hosts for master and every worker node, the entry likes:
<private ip> test01

2. Stop and Disable firewalld service
```
# systemctl stop firewalld && systemctl disable firewalld
```

3. Disable SELinux
```
#setenforce 0
#vi /etc/selinux/config
```

4. Disable swap
```
#echo "vm.swappiness = 0">> /etc/sysctl.conf
```

Check ali YUM source for K8s and configure if necessary
```
# cat /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=0
```

And update repo files with the attached .repo files (repo.tar)

```
# yum -y install epel-release

# yum clean all

# yum makecache
```

Install kubeadm and tools

```
# yum install docker kubelet kubeadm kubectl kubernetes-cni
```

Load images if there is any blocker on network to google for image pulling
<*.tar files>
And change the tag

Install Kubernetes with kubeadm

```
# kubeadm init --kubernetes-version=v1.11.2 --pod-network-cidr=10.244.0.0/16
[init] using Kubernetes version: v1.11.2
[preflight] running pre-flight checks
I0902 11:43:49.771080   20293 kernel_validator.go:81] Validating kernel version
I0902 11:43:49.771205   20293 kernel_validator.go:96] Validating kernel config
[preflight/images] Pulling images required for setting up a Kubernetes cluster
[preflight/images] This might take a minute or two, depending on the speed of your internet
connection
[preflight/images] You can also perform this action in beforehand using 'kubeadm config
images pull'
[kubelet] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-
flags.env"
[kubelet] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[preflight] Activating the kubelet service
[certificates] Generated ca certificate and key.
[certificates] Generated apiserver certificate and key.
[certificates] apiserver serving cert is signed for DNS names [test01 kubernetes
kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs
[10.96.0.1 172.31.65.91]
[certificates] Generated apiserver-kubelet-client certificate and key.
[certificates] Generated sa key and public key.
[certificates] Generated front-proxy-ca certificate and key.
[certificates] Generated front-proxy-client certificate and key.
[certificates] Generated etcd/ca certificate and key.
[certificates] Generated etcd/server certificate and key.
[certificates] etcd/server serving cert is signed for DNS names [test01 localhost] and IPs
[127.0.0.1 ::1]
[certificates] Generated etcd/peer certificate and key.
[certificates] etcd/peer serving cert is signed for DNS names [test01 localhost] and IPs
[172.31.65.91 127.0.0.1 ::1]
[certificates] Generated etcd/healthcheck-client certificate and key.
[certificates] Generated apiserver-etcd-client certificate and key.
[certificates] valid certificates and keys now exist in "/etc/kubernetes/pki"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/admin.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/kubelet.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/controller-manager.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/scheduler.conf"
[controlplane] wrote Static Pod manifest for component kube-apiserver to "/etc/kubernetes/
manifests/kube-apiserver.yaml"
[controlplane] wrote Static Pod manifest for component kube-controller-manager to "/etc/
kubernetes/manifests/kube-controller-manager.yaml"
[controlplane] wrote Static Pod manifest for component kube-scheduler to "/etc/kubernetes/
manifests/kube-scheduler.yaml"
[etcd] Wrote Static Pod manifest for a local etcd instance to "/etc/kubernetes/manifests/
etcd.yaml"
[init] waiting for the kubelet to boot up the control plane as Static Pods from directory "/etc/
kubernetes/manifests"
[init] this might take a minute or longer if the control plane images have to be pulled
[apiclient] All control plane components are healthy after 40.002134 seconds
[uploadconfig] storing the configuration used in ConfigMap "kubeadm-config" in the "kube-
system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config-1.11" in namespace kube-system with the
```

```
configuration for the kubelets in the cluster
[markmaster] Marking the node test01 as master by adding the label "node-
role.kubernetes.io/master=''"
[markmaster] Marking the node test01 as master by adding the taints [node-
role.kubernetes.io/master:NoSchedule]
[patchnode] Uploading the CRI Socket information "/var/run/dockershim.sock" to the Node
API object "test01" as an annotation
[bootstraptoken] using token: c8ne1o.hh60w2lrsumq2khh
[bootstraptoken] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order
for nodes to get long term certificate credentials
[bootstraptoken] configured RBAC rules to allow the csrapprover controller automatically
approve CSRs from a Node Bootstrap Token
[bootstraptoken] configured RBAC rules to allow certificate rotation for all node client
certificates in the cluster
[bootstraptoken] creating the "cluster-info" ConfigMap in the "kube-public" namespace
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes master 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/

You can now join any number of machines by running the following on each node
as root:

  kubeadm join 172.31.65.91:6443 --token c8ne1o.hh60w2lrsumq2khh --discovery-token-ca-
cert-hash
sha256:e3d8b356c8457dff16b5a39a6c358493f8734fed54fa22380b58bbc4f355de34
```

Configure kubectl

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

Install and configure flannel network

```
# mkdir -p /etc/cni/net.d/

# cat /etc/cni/net.d/10-flannel.conf
{
“name”: “cbr0”,
“type”: “flannel”,
“delegate”: {
```

```
"isDefaultGateway": true
}
}
```

```
# mkdir /run/flannel

# cat /run/flannel/subnet.env
FLANNEL_NETWORK=10.244.0.0/16
FLANNEL_SUBNET=10.244.1.0/24
FLANNEL_MTU=1450
FLANNEL_IPMASQ=true

# wget https://raw.githubusercontent.com/coreos/flannel/v0.9.1/Documentation/kube-flannel.yml
# kubectl apply -f kube-flannel.yml
```

Verify the installation

```
# kubectl get nodes --all-namespaces
NAME     STATUS   ROLES    AGE    VERSION
test01   Ready    master   20m    v1.11.2
[root@test01 config]# kubectl get pods --all-namespaces
NAMESPACE     NAME                               READY   STATUS    RESTARTS   AGE
kube-system   coredns-78fcdf6894-5bctd           1/1     Running   0          20m
kube-system   coredns-78fcdf6894-fc7bd           1/1     Running   0          20m
kube-system   etcd-test01                        1/1     Running   0          19m
kube-system   kube-apiserver-test01              1/1     Running   0          19m
kube-system   kube-controller-manager-test01     1/1     Running   0          19m
kube-system   kube-flannel-ds-lqvwr              1/1     Running   0          55s
kube-system   kube-proxy-7b5bm                   1/1     Running   0          20m
kube-system   kube-scheduler-test01              1/1     Running   0          19m
```

Dashboard installation
Option 1 without authentication

```
# kubectl apply -f kubernetes-dashboard-http.yaml
# kubectl apply -f admin-role.yaml
# kubectl apply -f kubernetes-dashboard-admin.rbac.yaml
```

Checking with private IP:
```
# curl http://<private-ip>:31000
```

Configure in 云服务器 ECS --> 安全组列表 to enable the income through port of 31000 for public IP

Login with:
http://<public-IP>:31000

Option 2 with token authentication:

```
# kubectl apply -f /opt/config/dashboard2/dashboard.yaml
```

Get token by:
```
# kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep admin-user | awk '{print $1}')
```

Config 云服务器 ECS --> 安全组列表 enable the income through port of 30000 ports for public IP

Login https://<public-IP>:30000/#!/login with providing token

Additional information:
API

Config 云服务器 ECS --> 安全组列表 to open 6443 port for public IP

https://<public-ip>:6443/api/v1

The used images are exported with following commands:
with command:
```
# for i in `docker images | grep -v "REPOSITORY" | awk -F" " '{ print $1":"$2 }'`
do
    file=`echo $i | sed 's/\//@/g' `
    docker save -o ${file}.tar $i
done
```

and they can be loaded with "docker load " command.