所属项 目默认项目

标签(未配置)

实例名称samctfu01

登录方式关联密钥（skey-rtewhlnb | yummyliu\_eks）

安全加固免费开通

云监控免费开通

注意：请牢记您所设置的密钥，设置密钥登录后将无法使用密码登录，如遗忘可登录CVM控制台重新关联SSH密钥。

<https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64>

<http://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64r>

<http://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg>

<https://packages.cloud.google.com/yum/doc/yum-key.gpg>

配置kubernetes的yum源

cat <<EOF > /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86\_64

enabled=1

gpgcheck=0

repo\_gpgcheck=0

gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg

https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

EOF

wget <https://dl.k8s.io/v1.12.1/kubernetes-server-linux-amd64.tar.gz>

wget <https://github.com/etcd-io/etcd/releases/download/v3.3.10/etcd-v3.3.10-linux-amd64.tar.gz>

cat > /etc/hosts << EOF

10.1.1.102 k8s-master 192.168.116.130 worker1 192.168.116.131 worker2

1q2w3e4!Q@W#E$

scp C:\Users\Administrator\Desktop\kubernetes-bins.tar.gz root@10.1.1.102:/home

kubeadm init --image-repository registry.aliyuncs.com/google\_containers --kubernetes-version v1.16.0 --apiserver-advertise-address 10.1.1.102 --pod-network-cidr=10.244.0.0/16 --token-ttl 0

kubeadm init --image-repository registry.aliyuncs.com/google\_containers --kubernetes-version v1.16.0 --apiserver-advertise-address 10.1.1.206 --pod-network-cidr=10.244.0.0/16 --token-ttl 0

kubeadm join 192.168.29.137:6443 --token 5w6qwh.8n0ektfrjdct3ib4 --discovery-token-ca-cert-hash sha256:af9e070ea723dd2281c2ae2414c932832a012d40bc55dc9c747bb00e68602388

kubeadm join 10.1.1.102:6443 --token 6bw0zz.neijxfemgd1pori2 --discovery-token-ca-cert-hash sha256:59e5944b5fe45434bd57ba5fe6af98b30ac6f9b63c7ebfda742e8f6bcd74d912

dsv139.mchkyxusf9w6zh00

kubeadm join 10.1.1.102:6443 --token 6bw0zz.neijxfemgd1pori2 --discovery-token-ca-cert-hash sha256:59e5944b5fe45434bd57ba5fe6af98b30ac6f9b63c7ebfda742e8f6bcd74d912

root@VM-1-149

/home/myscript/cluster-a6.sh

[reset] Reading configuration from the cluster...

[reset] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -oyaml'

[preflight] Running pre-flight checks

[reset] Removing info for node "vm-1-37-ubuntu" from the ConfigMap "kubeadm-config" in the "kube-system" Namespace

[reset] Stopping the kubelet service

[reset] Unmounting mounted directories in "/var/lib/kubelet"

[reset] Deleting contents of config directories: [/etc/kubernetes/manifests /etc/kubernetes/pki]

[reset] Deleting files: [/etc/kubernetes/admin.conf /etc/kubernetes/kubelet.conf /etc/kubernetes/bootstrap-kubelet.conf /etc/kubernetes/controller-manager.conf /etc/kubernetes/scheduler.conf]

[reset] Deleting contents of stateful directories: [/var/lib/etcd /var/lib/kubelet /var/lib/dockershim /var/run/kubernetes /var/lib/cni]

The reset process does not clean CNI configuration. To do so, you must remove /etc/cni/net.d

The reset process does not reset or clean up iptables rules or IPVS tables.

If you wish to reset iptables, you must do so manually by using the "iptables" command.

If your cluster was setup to utilize IPVS, run ipvsadm --clear (or similar)

to reset your system's IPVS tables.

The reset process does not clean your kubeconfig files and you must remove them manually.

Please, check the contents of the $HOME/.kube/config file.

W0608 19:40:10.378238 14571 configset.go:348] WARNING: kubeadm cannot validate component configs for API groups [kubelet.config.k8s.io kubeproxy.config.k8s.io]

[init] Using Kubernetes version: v1.19.7

[preflight] Running pre-flight checks

[WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The recommended driver is "systemd". Please follow the guide at https://kubernetes.io/docs/setup/cri/

[WARNING FileExisting-conntrack]: conntrack not found in system path

[WARNING FileExisting-ebtables]: ebtables not found in system path

[WARNING FileExisting-socat]: socat not found in system path

[WARNING SystemVerification]: this Docker version is not on the list of validated versions: 20.10.7. Latest validated version: 19.03

[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 localhost vm-1-37-ubuntu] and IPs [10.45.0.1 10.1.1.37 127.0.0.1 124.156.237.68]

[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 vm-1-37-ubuntu] and IPs [10.1.1.37 127.0.0.1 ::1]

[certs] Generating "etcd/peer" certificate and key

[certs] etcd/peer serving cert is signed for DNS names [localhost vm-1-37-ubuntu] and IPs [10.1.1.37 127.0.0.1 ::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 15.004151 seconds

[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace

[kubelet] Creating a ConfigMap "kubelet-config-1.19" in namespace kube-system with the configuration for the kubelets in the cluster

[upload-certs] Skipping phase. Please see --upload-certs

[mark-control-plane] Marking the node vm-1-37-ubuntu as control-plane by adding the label "node-role.kubernetes.io/master=''"

[mark-control-plane] Marking the node vm-1-37-ubuntu as control-plane by adding the taints [node-role.kubernetes.io/master:NoSchedule]

[bootstrap-token] Using token: r6l73r.vbg0f5bdf5x96gn1

[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles

[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to get nodes

[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials

[bootstrap-token] configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token

[bootstrap-token] configured RBAC rules to allow certificate rotation for all node client certificates in the cluster

[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace

[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key

[addons] Applied essential addon: CoreDNS

[addons] Applied essential addon: kube-proxy

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 10.1.1.37:6443 --token r6l73r.vbg0f5bdf5x96gn1 \

--discovery-token-ca-cert-hash sha256:37e2755062ecbf333616048a67c6d183c3912e5abf8a4c9aa8c120a9078acfa5

Error from server (NotFound): nodes "VM-1-37-ubuntu" not found

node/vm-1-37-ubuntu untainted

configmap/calico-config created

customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created

clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created

clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created

clusterrole.rbac.authorization.k8s.io/calico-node created

clusterrolebinding.rbac.authorization.k8s.io/calico-node created

daemonset.apps/calico-node created

serviceaccount/calico-node created

deployment.apps/calico-kube-controllers created

serviceaccount/calico-kube-controllers created

poddisruptionbudget.policy/calico-kube-controllers created

Successfully created 4 'IPPool' resource(s)

✓ Setting up broker RBAC

✓ Deploying the Submariner operator

✓ Created operator CRDs

✓ Created operator namespace: submariner-operator

✓ Created operator service account and role

✓ Created lighthouse service account and role

✓ Created Lighthouse service accounts and roles

✓ Deployed the operator successfully

✓ Deploying the broker

✓ The broker has been deployed

✓ Creating broker-info.subm file

✓ A new IPsec PSK will be generated for broker-info.subm

The authenticity of host '43.128.232.201 (43.128.232.201)' can't be established.

ECDSA key fingerprint is SHA256:2mD1lkAWMZq8Zvi2qLzIhdP9+SPNiSe4LppgXRRWlTU.

Are you sure you want to continue connecting (yes/no)? no

Host key verification failed.

lost connection

\* broker-info.subm says broker is at: https://124.156.237.68:6443

\* No worker node found to label as the gateway

Discovered network details:

Network plugin: generic

Service CIDRs: [10.45.0.0/16]

Cluster CIDRs: [10.44.0.0/16]

✓ Discovering network details

✓ Validating Globalnet configurations

✓ Discovering multi cluster details

✓ Deploying the Submariner operator

✓ Created Lighthouse service accounts and roles

✓ Creating SA for cluster

✓ Deploying Submariner

✓ Submariner is up and running

root@VM-1-37-ubuntu:~#

root@VM-1-37-ubuntu:~#

root@VM-1-37-ubuntu:~#

root@VM-1-37-ubuntu:~#

root@VM-1-37-ubuntu:~#

root@VM-1-37-ubuntu:~#