# 安装 k3s

## 安装规划

|  |  |  |
| --- | --- | --- |
| 角色 | IP | 备注 |
| K3s-server | 192.168.80.25 |  |
| K3s-agent | 192.168.80.26 |  |

## 安装前提

# 修改主机名

sudo hostnamectl set-hostname k3s-server/k3s-agent

sudo vi /etc/hosts

192.168.80.25 k3s-server

192.168.80.26 k3s-agent

# 安装ssh 服务

sudo apt-get install openssh-server

sudo apt-get install curl

## 安装 (脚本方式)

安装选项参考：<https://docs.rancher.cn/docs/k3s/installation/install-options/_index>

# Ubuntu 切换到 root 用户

sudo su -

# 执行安装脚本

curl -sfL http://rancher-mirror.cnrancher.com/k3s/k3s-install.sh | INSTALL\_K3S\_MIRROR=cn sh -

# 节点状态

sudo kubectl get nodes

NAME STATUS ROLES AGE VERSION

k3s-server Ready control-plane,master 78s v1.20.5+k3s1

# 卸载

/usr/local/bin/k3s-uninstall.sh

k3s 相关命令说明：

|  |  |
| --- | --- |
| **命令** | **描述** |
| k3s server | 运行 K3s server，它还将启动 Kubernetes control-plane 组件，如 API server, controller-manager, 和 scheduler。 |
| k3s agent | 运行 K3s agent 节点。这将使 K3s 作为工作节点运行，启动 Kubernetes 节点服务kubelet和kube-proxy。 |
| k3s kubectl | 运行嵌入式[kubectl](https://kubernetes.io/docs/reference/kubectl/overview/) CLI。如果没有设置KUBECONFIG环境变量，当启动 K3s 服务器节点时，将自动尝试使用在/etc/rancher/k3s/k3s.yaml创建的配置文件。 |
| k3s crictl | 运行一个嵌入式[crictl](https://github.com/kubernetes-sigs/cri-tools/blob/master/docs/crictl.md)。这是一个用于与 Kubernetes 的容器运行时接口（CRI）交互的 CLI。对调试很有用。 |
| k3s ctr | 运行一个嵌入式的[ctr](https://github.com/projectatomic/containerd/blob/master/docs/cli.md)。这是为 containerd（K3s 使用的容器守护进程）提供的 CLI。对调试很有用。 |
| k3s help | 显示一个命令的命令列表或帮助。 |

# 集群配置

## k3s server 节点

# 启动项

sudo vi /etc/systemd/system/multi-user.target.wants/k3s.service

ExecStart=/usr/local/bin/k3s server --no-deploy traefik --write-kubeconfig-mode 664 --tls-san 192.168.80.25 --kube-apiserver-arg="advertise-address=192.168.80.25" --kube-apiserver-arg="external-hostname=192.168.80.25"

# 重启

sudo systemctl daemon-reload

sudo systemctl restart k3s

# 获取 k3s server 的 token

sudo cat /var/lib/rancher/k3s/server/node-token

K1083cd9ee544e37db02d6d6ac67397ae26898ff5dba0f3b3a4c8ce668bc66f04a0::server:e32090ba0c4ec71a13846143bfa623e7

## k3s agent 节点

# 启动项

sudo vi /etc/systemd/system/multi-user.target.wants/k3s.service

ExecStart=/usr/local/bin/k3s agent --server https://192.168.80.25:6443 --node-external-ip 192.168.80.25 --token K1083cd9ee544e37db02d6d6ac67397ae26898ff5dba0f3b3a4c8ce668bc66f04a0::server:e32090ba0c4ec71a13846143bfa623e7

sudo systemctl daemon-reload

sudo systemctl restart k3s

另外，还可通过指定参数的方式安装agent：

curl -sfL http://rancher-mirror.cnrancher.com/k3s/k3s-install.sh | INSTALL\_K3S\_MIRROR=cn K3S\_URL=https://192.168.80.25:6443 K3S\_TOKEN=K1083cd9ee544e37db02d6d6ac67397ae26898ff5dba0f3b3a4c8ce668bc66f04a0::server:e32090ba0c4ec71a13846143bfa623e7 sh -

# 卸载

/usr/local/bin/k3s-agent-uninstall.sh

## 查看集群状态 (k3s server)

# 查看节点信息

sudo kubectl get nodes

NAME         STATUS   ROLES                  AGE   VERSION

k3s-server   Ready    control-plane,master   44m   v1.20.5+k3s1

k3s-agent    Ready    <none>                 57s   v1.20.5+k3s1

# worker 节点角色默认为none，修改为worker

sudo kubectl label node k3s-agent node-role.kubernetes.io/worker=worker

sudo kubectl get nodes

NAME STATUS ROLES AGE VERSION

k3s-server Ready control-plane,master 45m v1.20.5+k3s1

k3s-agent Ready worker 2m26s v1.20.5+k3s1

# 相关资源检查

## Kubectl 命令

ubuntu@k3s-server:~$ which kubectl

/usr/local/bin/kubectl

ubuntu@k3s-server:~$ ls -l /usr/local/bin/kubectl

lrwxrwxrwx 1 root root 3 Apr  8 01:14 /usr/local/bin/kubectl -> k3

## Kubernetes 版本

ubuntu@k3s-server:~$ kubectl version

Client Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.5+k3s1", GitCommit:"355fff3017b06cde44dbd879408a3a6826fa7125", GitTreeState:"clean", BuildDate:"2021-03-31T06:21:52Z", GoVersion:"go1.15.10", Compiler:"gc", Platform:"linux/amd64"}

Server Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.5+k3s1", GitCommit:"355fff3017b06cde44dbd879408a3a6826fa7125", GitTreeState:"clean", BuildDate:"2021-03-31T06:21:52Z", GoVersion:"go1.15.10", Compiler:"gc", Platform:"linux/amd64"}

## 节点信息

ubuntu@k3s-server:~$ kubectl get node -o wide

NAME         STATUS   ROLES                  AGE     VERSION        INTERNAL-IP     EXTERNAL-IP     OS-IMAGE             KERNEL-VERSION      CONTAINER-RUNTIME

k3s-agent    Ready    worker                 9m29s   v1.20.5+k3s1   192.168.80.26   192.168.80.25   Ubuntu 16.04.6 LTS   4.4.0-142-generic   containerd://1.4.4-k3s1

k3s-server   Ready    control-plane,master   52m     v1.20.5+k3s1   192.168.80.25   <none>          Ubuntu 16.04.6 LTS   4.4.0-142-generic   containerd://1.4.4-k3s1

## 命名空间

ubuntu@k3s-server:~$ kubectl get ns

NAME              STATUS   AGE

default           Active   54m

kube-system       Active   54m

kube-public       Active   54m

kube-node-lease   Active   54m

## K3s 容器

ubuntu@k3s-server:~$ kubectl get pod -n kube-system

NAME                                      READY   STATUS             RESTARTS   AGE

helm-install-traefik-9pcb5                0/1     Completed          0          60m

metrics-server-86cbb8457f-hk2mg           1/1     Running            2          60m

local-path-provisioner-5ff76fc89d-lznqh   1/1     Running            2          60m

svclb-traefik-tbf98                       2/2     Running            5          59m

traefik-6f9cbd9bd4-7jm8l                  1/1     Running            2          59m

svclb-traefik-8rf82                       2/2     Running            0          17m

coredns-854c77959c-59ggw                  0/1     CrashLoopBackOff   21         60m

解决coredns 启动失败问题：worker 节点是克隆的，需要重新安装 k3s，因为克隆的环境，token是一样的

## K3s 进程

controlplane其他服务apiserver/scheduler/controller-manager 等通过 k3s 启动

ubuntu@k3s-server:~$ ps aux | grep "k3s server"

root       8319 14.7 24.9 1307216 506976 ?      Ssl  02:31   2:15 /usr/local/bin/k3s server

ubuntu@k3s-server:~$ ps -T 8319

   PID   SPID TTY      STAT   TIME COMMAND

  8319   8319 ?        Ssl    0:00 /usr/local/bin/k3s server

  8319   8323 ?        Ssl    0:21 /usr/local/bin/k3s server

  8319   8324 ?        Ssl    0:14 /usr/local/bin/k3s server

  8319   8325 ?        Ssl    0:00 /usr/local/bin/k3s server

  8319   8326 ?        Ssl    0:00 /usr/local/bin/k3s server

  8319   8327 ?        Ssl    0:02 /usr/local/bin/k3s server

  8319   8328 ?        Ssl    0:02 /usr/local/bin/k3s server

  8319   8330 ?        Ssl    0:00 /usr/local/bin/k3s server

  8319   8331 ?        Ssl    0:21 /usr/local/bin/k3s server

  8319   8332 ?        Ssl    0:21 /usr/local/bin/k3s server

  8319   8333 ?        Ssl    0:00 /usr/local/bin/k3s server

  8319   8334 ?        Ssl    0:00 /usr/local/bin/k3s server

  8319   8335 ?        Ssl    0:00 /usr/local/bin/k3s server

  8319   8469 ?        Ssl    0:00 /usr/local/bin/k3s server

  8319   8480 ?        Ssl    0:09 /usr/local/bin/k3s server

  8319   8481 ?        Ssl    0:02 /usr/local/bin/k3s server

  8319   8544 ?        Ssl    0:14 /usr/local/bin/k3s server

  8319   8550 ?        Ssl    0:22 /usr/local/bin/k3s server

  8319   9601 ?        Ssl    0:01 /usr/local/bin/k3s server

参考代码：<https://github.com/k3s-io/k3s/blob/master/pkg/daemons/control/server.go>

# 使用 docker 替换 containerd

## 安装 docker

# 可能缺少的公共命令

sudo apt-get install software-properties-common

# 证书

sudo curl -fsSL http://mirrors.aliyun.com/docker-ce/linux/ubuntu/gpg | sudo apt-key add -

# 仓库信息

sudo add-apt-repository "deb [arch=amd64] http://mirrors.aliyun.com/docker-ce/linux/ubuntu $(lsb\_release -cs) stable"

# 更新 cache

sudo apt-get update

# 可用版本查询

sudo apt-cache policy docker-ce

# 安装 docker 19.03.15~3-0~ubuntu-xenial

sudo apt-get install docker-ce=5:19.03.15~3-0~ubuntu-xenial

sudo docker version

使用脚本安装：

curl https://releases.rancher.com/install-docker/19.03.sh | sh

## 调整 k3s 容器

# k3s server启动项

sudo vi /etc/systemd/system/multi-user.target.wants/k3s.service

ExecStart=/usr/local/bin/k3s server --docker --no-deploy traefik --write-kubeconfig-mode 664 --tls-san 192.168.80.25 --kube-apiserver-arg="advertise-address=192.168.80.25" --kube-apiserver-arg="external-hostname=192.168.80.25"

# k3s agent启动项 (k3s agent 由systemctl管理)

sudo vi /etc/systemd/system/multi-user.target.wants/k3s.service

ExecStart=/usr/local/bin/k3s agent –-docker --server https://192.168.80.25:6443 --node-external-ip 192.168.80.25 --token K1083cd9ee544e37db02d6d6ac67397ae26898ff5dba0f3b3a4c8ce668bc66f04a0::server:e32090ba0c4ec71a13846143bfa623e7

# 重启

sudo systemctl daemon-reload

sudo systemctl restart k3s

k3s agent 指定参数安装：

# 卸载

/usr/local/bin/k3s-agent-uninstall.sh

# 指定 docker 作为启动容器

curl -sfL http://rancher-mirror.cnrancher.com/k3s/k3s-install.sh | INSTALL\_K3S\_MIRROR=cn INSTALL\_K3S\_EXEC='--docker' K3S\_URL=https://192.168.80.25:6443 K3S\_TOKEN=K1083cd9ee544e37db02d6d6ac67397ae26898ff5dba0f3b3a4c8ce668bc66f04a0::server:e32090ba0c4ec71a13846143bfa623e7 sh -

# 创建资源

## Deployment

cat > kubernetes-bootcamp-v1.yaml <<EOF

apiVersion: apps/v1

kind: Deployment

metadata:

  labels:

    app: kubernetes-bootcamp-v1

  name: kubernetes-bootcamp-v1

spec:

  replicas: 2

  selector:

    matchLabels:

      app: kubernetes-bootcamp-v1

  template:

    metadata:

      labels:

        app: kubernetes-bootcamp-v1

    spec:

      containers:

      - image: jocatalin/kubernetes-bootcamp:v1

        imagePullPolicy: IfNotPresent

        name: kubernetes-bootcamp-v1

EOF

sudo kubectl apply -f kubernetes-bootcamp-v1.yaml

sudo kubectl get pod

NAME                                      READY   STATUS    RESTARTS   AGE

kubernetes-bootcamp-v1-7946548f4f-k4mcz   1/1     Running   0          14m

kubernetes-bootcamp-v1-7946548f4f-kclzs   1/1     Running   0          14m

## Service

cat > kubernetes-bootcamp-svc-v1.yaml <<EOF

apiVersion: v1

kind: Service

metadata:

  labels:

    app: kubernetes-bootcamp-v1

  name: kubernetes-bootcamp-v1

spec:

  ports:

  - port: 8080

    protocol: TCP

    targetPort: 8080

  selector:

    app: kubernetes-bootcamp-v1

  type: ClusterIP

EOF

sudo kubectl apply -f kubernetes-bootcamp-svc-v1.yaml

sudo kubectl get svc

NAME                     TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE

kubernetes               ClusterIP   10.43.0.1      <none>        443/TCP    3h15m

kubernetes-bootcamp-v1   ClusterIP   10.43.53.149   <none>        8080/TCP   6s

curl 10.43.53.149:8080

Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-v1-7946548f4f-vqscx | v=1

## Ingress

不需要单独安装 ingress controller, 因为 k3s 已内置了 Traefik，直接创建 ingress

cat > k3s-ingress.yaml <<EOF

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

  name: "k3s-ingress"

  labels:

    app: k3s-ingress

spec:

  rules:

    - host: k3s.ingress.me

      http:

        paths:

        - path: /v1

          backend:

            serviceName: "kubernetes-bootcamp-v1"

            servicePort: 8080

EOF

sudo kubectl apply -f k3s-ingress.yaml

sudo kubectl get ingress k3s-ingress

NAME          CLASS    HOSTS            ADDRESS         PORTS   AGE

k3s-ingress   <none>   k3s.ingress.me   192.168.80.25   80      9s

sudo vi /etc/hosts

192.168.80.25 k3s.ingress.me

curl http://k3s.ingress.me/v1

Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-v1-7946548f4f-vqscx | v=1

# 网络

k3s 内置 Flannel 网络插件，默认使用 VXLAN 后端，默认IP段为 `10.42.0.0/16`.

内置的 Flannel，支持 VXLAN，ipsec，host-gw, wireguard

支持的CNI：Flannel， Cannal， Calico

ip -o -d link show flannel.1

4: flannel.1: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1450 qdisc noqueue state UNKNOWN mode DEFAULT group default \    link/ether ea:25:1c:f5:2f:6b brd ff:ff:ff:ff:ff:ff promiscuity 0 \    vxlan id 1 local 192.168.80.20 dev ens33 srcport 0 0 dstport 8472 nolearning ageing 300 addrgenmode eui64

# 存储

## 内置存储

k3s 删除了 k8s 内置的 cloud provider 及 storage 插件，内置了 Local Path Provider.

cat > k3s-pvc.yaml <<EOF

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

  name: local-path-pvc

  namespace: default

spec:

  accessModes:

    - ReadWriteOnce

  storageClassName: local-path

  resources:

    requests:

      storage: 2Gi

EOF

cat > k3s-pod.yaml <<EOF

apiVersion: v1

kind: Pod

metadata:

  name: volume-test

spec:

  containers:

  - name: volume-test

    image: jocatalin/kubernetes-bootcamp:v1

    imagePullPolicy: IfNotPresent

    volumeMounts:

    - name: volv

      mountPath: /data

  volumes:

  - name: volv

    persistentVolumeClaim:

      claimName: local-path-pvc

EOF

sudo kubectl create -f  k3s-pvc.yaml

sudo kubectl create -f  k3s-pod.yaml

sudo kubectl get pod volume-test

NAME          READY   STATUS    RESTARTS   AGE

volume-test   1/1     Running   0          39s

sudo kubectl get pvc

AME             STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE

local-path-pvc   Bound    pvc-4901abe1-46ea-4b9d-8b5c-7a09dcfa2232   2Gi        RWO            local-path     20s

sudo kubectl get pv

NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                    STORAGECLASS   REASON   AGE

pvc-4901abe1-46ea-4b9d-8b5c-7a09dcfa2232   2Gi        RWO            Delete           Bound    default/local-path-pvc   local-path              31s

sudo kubectl describe pv pvc-4901abe1-46ea-4b9d-8b5c-7a09dcfa2232

Name:              pvc-4901abe1-46ea-4b9d-8b5c-7a09dcfa2232

Labels:            <none>

Annotations:       pv.kubernetes.io/provisioned-by: rancher.io/local-path

Finalizers:        [kubernetes.io/pv-protection]

StorageClass:      local-path

Status:            Bound

Claim:             default/local-path-pvc

Reclaim Policy:    Delete

Access Modes:      RWO

VolumeMode:        Filesystem

Capacity:          2Gi

Node Affinity:

  Required Terms:

    Term 0:        kubernetes.io/hostname in [k3s-agent]

Message:

Source:

    Type:          HostPath (bare host directory volume)

    Path:          /var/lib/rancher/k3s/storage/pvc-4901abe1-46ea-4b9d-8b5c-7a09dcfa2232\_default\_local-path-pvc

    HostPathType:  DirectoryOrCreate

Events:            <none>