K8S环境中通过HELM 部署应用（MYSQL）

**硬件要求：**

虚拟机：2 台（1 台 master + 1 台 node，或单节点兼作 master 和 node）

系统：CentOS 7.9（64 位，最小化安装）

资源：每台至少 2 核 CPU、4GB 内存、20GB 磁盘

网络：VMware 使用「NAT 模式」，确保虚拟机可联网（需配置静态 IP）

**软件依赖：**

Docker（容器运行时）

kubeadm、kubelet、kubectl（K8s 核心组件）

Helm（包管理工具）

网络插件（flannel）

以VMware为例

**第一步下载 CentOS 镜像**

<https://vault.centos.org/7.9.2009/isos/x86_64/>（选择CentOS-7-x86\_64-Minimal-2009.iso）

**第二步： 创建虚拟机（以 1 台 master 为例，node 节点步骤一致）**

打开 VMware → 新建虚拟机 → 典型配置 → 选择下载的 ISO 镜像

操作系统选择「Linux → CentOS 7 64 位」

虚拟机名称：k8s-master（node 节点命名为k8s-node1）

磁盘大小：20GB，勾选「将磁盘存储为单个文件」

自定义硬件：CPU≥2 核，内存≥4GB，网络适配器选择「NAT 模式」

启动虚拟机，安装 CentOS（设置 root 密码：例如123456）

#### **第三步：配置静态 IP（每台虚拟机都需配置）**

查看网卡名称

ip addr

编辑网络配置文件：

vi /etc/sysconfig/network-scripts/ifcfg-ens33

修改配置（按实际网络调整 IP，确保与 VMware NAT 网段一致）：

BOOTPROTO=static # 静态IP

ONBOOT=yes # 开机启动IPADDR=192.168.159.130 # master节点IP（示例）NETMASK=255.255.255.0 # 子网掩码GATEWAY=192.168.159.2 # 网关（VMware NAT网关，在「编辑→虚拟网络编辑器」中查看）

DNS1=114.114.114.114 # DNS服务器

重启网络：systemctl restart network

验证联网：ping baidu.com # 能通则正常

#### **第4步：基础环境配置（每台虚拟机执行）**

##### **（1）关闭防火墙、SELinux、Swap**

# 关闭防火墙

systemctl stop firewalld

systemctl disable firewalld

# 关闭SELinux（临时+永久）

setenforce 0

sed -i 's/SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config

# 关闭Swap（临时+永久）

swapoff -a

sed -i '/swap/s/^/#/' /etc/fstab # 注释swap行

##### **配置主机名和 hosts**

# master节点设置主机名

hostnamectl set-hostname k8s-master

# node节点设置主机名（在node节点执行）

hostnamectl set-hostname k8s-node1

# 编辑hosts文件（所有节点都添加）

cat >> /etc/hosts << EOF

192.168.159.130 k8s-master

192.168.159.--- k8s-node1 # 替换为node节点实际IP

EOF

##### **配置时间同步**

# 安装chrony

yum install -y chrony

# 启动并设置开机自启

systemctl start chronyd

systemctl enable chronyd

# 同步时间

chronyc sources

##### **配置内核参数（K8s 要求）**

# 添加内核模块

cat >> /etc/modules-load.d/k8s.conf << EOF

overlay br\_netfilter

EOF

# 加载模块

modprobe overlay modprobe br\_netfilter

# 配置内核参数

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

net.ipv4.ip\_forward = 1

EOF

# 生效配置

sysctl --system

第四步安装docker(所有节点）

# 安装依赖

yum install -y yum-utils device-mapper-persistent-data lvm2

# 添加Docker源（阿里云）

yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo # 安装Docker（指定版本，避免兼容性问题）

yum install -y docker-ce-19.03.15 docker-ce-cli-19.03.15 containerd.io

# 启动Docker并设置开机自启

systemctl start docker systemctl enable docker

# 配置Docker镜像加速器（阿里云，提升拉取速度）

mkdir -p /etc/docker cat >> /etc/docker/daemon.json << EOF { "registry-mirrors": ["https://b9pmyelo.mirror.aliyuncs.com"] } EOF

# 重启Docker

systemctl daemon-reload systemctl restart docker

### 第五步：****部署 K8s 集群****

#### **1. 安装 kubeadm、kubelet、kubectl（所有节点）**

# 添加K8s源（阿里云）

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

[kubernetes] name=Kubernetes baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86\_64/

enabled=1

gpgcheck=1

repo\_gpgcheck=1 gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg

EOF

# 安装组件（指定版本，与Docker兼容）

yum install -y kubeadm-1.23.6 kubelet-1.23.6 kubectl-1.23.6

# 启动kubelet并设置开机自启

systemctl start kubelet systemctl enable kubelet

#### **2. 初始化 Master 节点（仅在 master 执行）**

# 初始化集群（指定阿里云镜像仓库，避免墙内拉取失败）

kubeadm init \

--image-repository registry.aliyuncs.com/google\_containers \ --kubernetes-version v1.23.6 \ --pod-network-cidr=10.244.0.0/16 # flannel网络需要的网段

# 初始化成功后，按提示配置kubectl（普通用户权限）

mkdir -p $HOME/.kube

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

# 验证master状态（此时节点为NotReady，因为未安装网络插件） kubectl get nodes

#### **3. 安装网络插件（flannel，仅在 master 执行）**

# 下载flannel配置文件（国内可访问地址）

kubectl apply -f https://raw.githubusercontent.com/flannel-io/flannel/v0.17.0/Documentation/kube-flannel.yml

# 等待3-5分钟，验证节点状态（变为Ready）

kubectl get nodes

#### **添加 Node 节点到集群（仅在 node 执行）**

会自动生成JOIN和token的命令，如果没有，可以在master节点重新生成kubeadm token create --print-join-command

例子：生成的JOIN和token的命令

kubeadm join 192.168.159.130:6443 \

--token abcdef.0123456789abcdef \ --discovery-token-ca-cert-hash sha256:--------

验证集群节点（在 master 执行）：

kubectl get nodes # 所有节点状态为Ready则成功

**第6步：下载HELM文件，选择合适的版本**

<https://github.com/helm/helm/releases>

# 下载Helm 3.8.0（适合K8s 1.23）

wget https://get.helm.sh/helm-v3.8.0-linux-amd64.tar.gz

# 解压并安装

tar -zxvf helm-v3.8.0-linux-amd64.tar.gz mv linux-amd64/helm /usr/local/bin/helm

# 验证安装

helm version # 输出版本信息则成功

#### **添加 Helm 仓库（存储应用模板）**

# 添加bitnami仓库（包含MySQL等常用应用）

helm repo add bitnami https://charts.bitnami.com/bitnami

# 更新仓库索引

helm repo update

第七步使用helm部署应用MYSQL例子

#### **1. 部署 MySQL（指定 root 密码和存储）**

# 创建命名空间

kubectl create namespace mysql-demo

# 部署MySQL（设置root密码为123456，使用默认存储）

helm install mysql bitnami/mysql \

--namespace mysql-demo \

--set auth.rootPassword=123456 \

--set persistence.size=10Gi # 数据存储大小

#### **2. 验证 MySQL 部署**

# 查看pod（状态为Running则成功）

kubectl get pods -n mysql-demo

# 查看service（获取访问IP和端口）

kubectl get svc -n mysql-demo

# 输出类似：mysql ClusterIP 10.109.xx.xx <none> 3306/TCP ...

# 连接MySQL测试（在master节点临时安装mysql客户端）

yum install -y mysql

mysql -h 10.109.xx.xx -u root -p # 输入密码123456，能登录则成功

流程：VMware 创建虚拟机 → 配置基础环境 → 安装 Docker 和 K8s 组件 → 初始化集群 → 安装 Helm → 部署 MySQL。

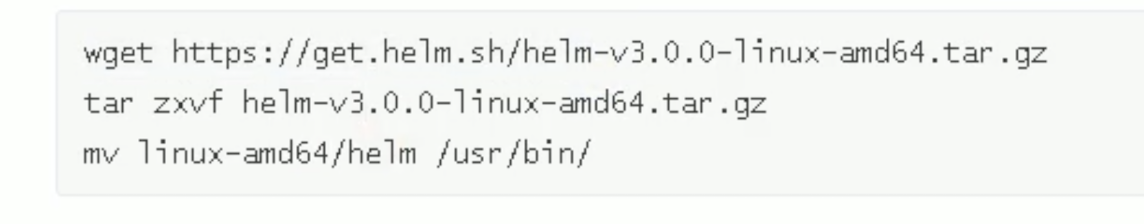
上面是也是chart部署MySQL的方式helm install mysql bitnami/mysql 中的 bitnami/mysql 是 Chart 的 “仓库名 + Chart 名”

用的是默认的chart，下面使用chart部署MySQL的方式，是明着部署mysql

第一步下载HELM文件，选择合适的版本

<https://github.com/helm/helm/releases>

第二步：解压到user/bin 目录



第三步下载chart仓库

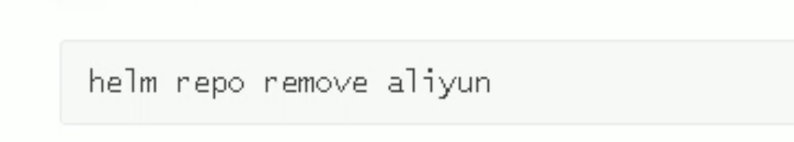


stable aliyun 这些都是名称，随意命名，一般命名stable

查看仓库

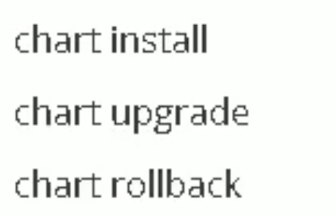


移除仓库

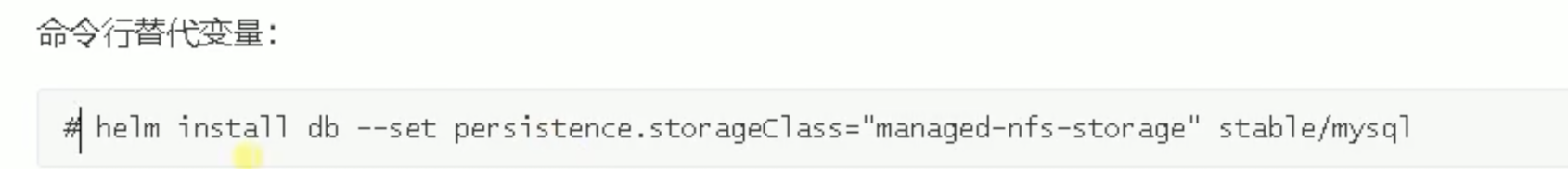


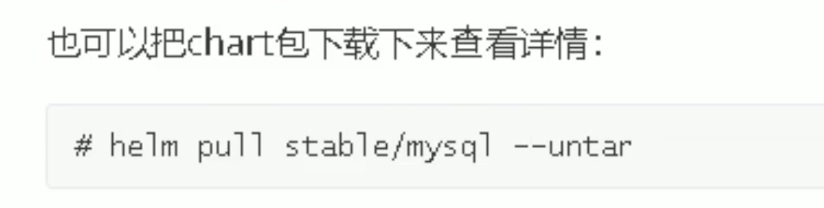
HELM基本使用

这是最基本的一个生命周期









这是部署MYSQL 使用CHART

#### **步骤 1：添加包含 MySQL Chart 的仓库**

Bitnami 仓库提供了成熟的 MySQL Chart：

bash

# 添加Bitnami仓库

helm repo add bitnami https://charts.bitnami.com/bitnami# 更新仓库索引

helm repo update

#### **步骤 2：查看 MySQL Chart 的默认配置（可选）**

了解可自定义的参数（如密码、存储类型、副本数等）：

bash

# 下载Chart到本地查看values.yaml（可选）

helm pull bitnami/mysql --untar# 查看默认配置（重点关注root密码、存储、端口等）cat mysql/values.yaml

关键配置项说明：

* auth.rootPassword：root 用户密码（默认随机生成，建议自定义）。
* primary.persistence.size：数据存储大小（默认 8Gi）。
* service.type：服务暴露方式（默认 ClusterIP，可选 NodePort/LoadBalancer）。

#### **步骤 3：创建自定义配置文件（覆盖默认值）**

新建mysql-values.yaml，指定需要修改的参数（其他参数用默认值）：

yaml

# mysql-values.yamlauth:

rootPassword: "MyRoot123!" # 自定义root密码primary:

persistence:

size: "10Gi" # 数据存储大小改为10Giservice:

type: NodePort # 暴露为NodePort，方便外部访问

nodePort: 30006 # 节点端口（范围30000-32767）

#### **步骤 4：用 Helm 部署 MySQL**

bash

# 安装MySQL，命名为"my-mysql"，使用自定义配置

helm install my-mysql bitnami/mysql -f mysql-values.yaml

输出类似如下（表示部署成功）：

plaintext

NAME: my-mysql

LAST DEPLOYED: ...

NAMESPACE: default

STATUS: deployed

...

#### **步骤 5：验证部署**

bash

# 查看Pod状态（Running表示正常）

kubectl get pods -l app.kubernetes.io/name=mysql

# 查看Service（确认NodePort暴露）

kubectl get svc my-mysql

# 查看PVC（确认存储绑定）

kubectl get pvc

IMG_256

#### **步骤 6：连接 MySQL 测试**

bash

# 进入MySQL容器

kubectl exec -it my-mysql-0 -- bash

# 登录MySQL（使用自定义的root密码）

mysql -u root -p# 输入密码"MyRoot123!"，成功登录则部署正常

****推荐用 Chart 的场景****：主流应用（MySQL、Nginx 等）、需要版本管理 / 回滚、团队协作、应用较复杂（含多组件）。

****不用 Chart 的场景****：无合适 Chart、应用极简、深度定制、学习调试、团队习惯原生 YAML。