Jenkins

# 安装

也可使用 jenkins/jenkins:lts-centos7 ，但不支持UTF-8。

## Jenkins环境配置

<https://blog.csdn.net/qq_41300350/article/details/90641944>

### 使用宿主主机的.NET环境

在宿主机安装.NET环境

下载以 .tar.gz 为后缀的 .NET Core 3.1 SDK 安装包

<https://dotnet.microsoft.com/download/dotnet-core/thank-you/sdk-3.1.404-linux-x64-binaries>

mkdir -p $HOME/dotnet && tar zxf dotnet-sdk-3.1.404-linux-x64.tar.gz -C $HOME/dotnet

export DOTNET\_ROOT=$HOME/dotnet

export PATH=$PATH:$HOME/dotnet

测试是否安装成功。

dotnet --help

这里使用 jenkins/jenkins:lts 。

# 进入容器

docker exec -it jenkins bash

# 安装依赖

cat > /etc/apt/sources.list << EOF

deb http://mirrors.aliyun.com/debian/ stretch main non-free contrib

deb-src http://mirrors.aliyun.com/debian/ stretch main non-free contrib

deb http://mirrors.aliyun.com/debian-security stretch/updates main

deb-src http://mirrors.aliyun.com/debian-security stretch/updates main

deb http://mirrors.aliyun.com/debian/ stretch-updates main non-free contrib

deb-src http://mirrors.aliyun.com/debian/ stretch-updates main non-free contrib

deb http://mirrors.aliyun.com/debian/ stretch-backports main non-free contrib

deb-src http://mirrors.aliyun.com/debian/ stretch-backports main non-free contrib

EOF

apt update

apt install -y libicu-dev

# 配置临时环境变量

export DOTNET\_ROOT=/root/dotnet

export PATH=$PATH:/root/dotnet

测试是否安装成功。

dotnet --help

### 使用宿主主机的Node.JS环境

在宿主机安装Node.JS环境

下载合适版本以 .tar.gz 为后缀的 Linux 二进制文件 (x64) 安装包

https://nodejs.org/zh-cn/download/

tar xf node-v14.15.1-linux-x64.tar.xz && mv node-v14.15.1-linux-x64 nodejs

# 配置软链接

ln -s /root/nodejs/bin/npm /usr/local/bin/

ln -s /root/nodejs/bin/node /usr/local/bin/

ln -s /root/nodejs/bin/ng /usr/local/bin/

# 安装Angular CLI

npm config set registry https://registry.npm.taobao.org \

&& npm set sass\_binary\_site https://npm.taobao.org/mirrors/node-sass \

&& set NODE\_OPTIONS --max\_old\_space\_size=1024 \

&& npm install -g @angular/cli

测试是否安装成功

node -v

npm -v

ng version

这里使用 jenkins/jenkins:lts 。

# 进入容器

docker exec -it jenkins bash

# 配置软链接

ln -s /root/nodejs/bin/npm /usr/local/bin/

ln -s /root/nodejs/bin/node /usr/local/bin/

ln -s /root/nodejs/bin/ng /usr/local/bin/

测试是否安装成功。

node -v

npm -v

ng -version

### 使用宿主主机的Maven环境

# 首先下载apache-maven-3.6.3

curl -o apache-maven-3.6.3-bin.tar.gz <http://mirrors.tuna.tsinghua.edu.cn/apache/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz>

tar zxf apache-maven-3.6.3-bin.tar.gz && mv apache-maven-3.6.3 maven

## 通过Docker进行安装

docker run \

-u root \

--name jenkins \

--restart=always \

-d \

-p 80:8080 \

-p 50000:50000 \

--dns 192.168.1.254 \

--dns 223.5.5.5 \

-v /etc/localtime:/etc/localtime:ro \

-v /usr/local/jenkins:/var/jenkins\_home \

-v /usr/bin/docker:/usr/bin/docker \

-v /var/run/docker.sock:/var/run/docker.sock \

-v /root/nodejs:/root/nodejs \

-v /root/dotnet:/root/dotnet \

-v /root/maven:/usr/local/software/apache-maven-3.6.3 \

-v /root/jdk1.8.0\_181:/data/jdk1.8.0\_181 \

jenkins/jenkins:lts

# 如果需要备份，备份宿主主机的 /usr/local/jenkins 即可。

## 通过Kubernetes进行安装

参考：<https://www.jianshu.com/p/aaa16f1566d7>

https://www.cnblogs.com/coolops/p/13129955.html

该配置方案配置了名称空间和节点调度策略，在安装之间需要创建相应的名称空间和添加工作节点的标签,建议节点配置8G以上内存。

如果需要使用 .NET 和 NodeJS 环境，需要在每个Job里检测并安装（并不会影响效率）。

kubectl label nodes worker01 jenkins=enabled

kubectl create ns jenkins

mkdir -p /usr/local/kubernetes/jenkins

镜像使用了 jenkins/jenkins:lts

### 创建RBAC

cat > /usr/local/kubernetes/jenkins/jenkins-rbac.yaml << EOF

apiVersion: v1

kind: ServiceAccount

metadata:

name: jenkins-sa

namespace: jenkins

---

apiVersion: rbac.authorization.k8s.io/v1beta1

kind: ClusterRole

metadata:

name: jenkins-cr

rules:

- apiGroups: ["extensions", "apps"]

resources: ["deployments"]

verbs: ["create", "delete", "get", "list", "watch", "patch", "update"]

- apiGroups: [""]

resources: ["services"]

verbs: ["create", "delete", "get", "list", "watch", "patch", "update"]

- apiGroups: [""]

resources: ["pods"]

verbs: ["create","delete","get","list","patch","update","watch"]

- apiGroups: [""]

resources: ["pods/exec"]

verbs: ["create","delete","get","list","patch","update","watch"]

- apiGroups: [""]

resources: ["pods/log"]

verbs: ["get","list","watch"]

- apiGroups: [""]

resources: ["secrets"]

verbs: ["get"]

---

apiVersion: rbac.authorization.k8s.io/v1beta1

kind: ClusterRoleBinding

metadata:

name: jenkins-crd

roleRef:

kind: ClusterRole

name: jenkins-cr

apiGroup: rbac.authorization.k8s.io

subjects:

- kind: ServiceAccount

name: jenkins-sa

namespace: jenkins

EOF

### 创建StatefulSet

cat > /usr/local/kubernetes/jenkins/jenkins-deploy.yaml << EOF

apiVersion: apps/v1

kind: StatefulSet

metadata:

name: jenkins

namespace: jenkins

spec:

replicas: 1

selector:

matchLabels:

app: jenkins

serviceName: jenkins

template:

metadata:

labels:

app: jenkins

spec:

dnsConfig:

nameservers:

- 192.168.1.254

containers:

- image: 'jenkins/jenkins:lts'

imagePullPolicy: IfNotPresent

securityContext:

runAsUser: 0

livenessProbe:

failureThreshold: 12

httpGet:

path: /login

port: 8080

scheme: HTTP

initialDelaySeconds: 600

periodSeconds: 12

timeoutSeconds: 5

name: jenkins

readinessProbe:

failureThreshold: 12

httpGet:

path: /login

port: 8080

scheme: HTTP

initialDelaySeconds: 600

periodSeconds: 12

timeoutSeconds: 5

volumeMounts:

- mountPath: /var/jenkins\_home

name: jenkins-home

- mountPath: /root/nodejs

name: nodejs

- mountPath: /usr/bin/docker

name: docker

- mountPath: /var/run/docker.sock

name: sock

resources:

requests:

memory: 8Gi

nodeSelector:

jenkins: enabled

restartPolicy: Always

serviceAccount: jenkins-sa

serviceAccountName: jenkins-sa

volumes:

- hostPath:

path: /root/nodejs

type: ''

name: nodejs

- hostPath:

path: /usr/bin/docker

type: ''

name: docker

- hostPath:

path: /var/run/docker.sock

type: Socket

name: sock

volumeClaimTemplates:

- metadata:

name: jenkins-home

annotations:

volume.beta.kubernetes.io/storage-class: course-nfs-storage

spec:

accessModes: [ "ReadWriteOnce" ]

resources:

requests:

storage: 100Gi

---

apiVersion: v1

kind: Service

metadata:

labels:

app: jenkins

name: jenkins

namespace: jenkins

spec:

ports:

- name: jenkins

port: 80

protocol: TCP

targetPort: 8080

selector:

app: jenkins

type: ClusterIP

---

apiVersion: networking.k8s.io/v1beta1

kind: Ingress

metadata:

annotations:

app: jenkins

labels:

app: jenkins

name: jenkins

namespace: jenkins

spec:

rules:

- host: jenkins.98.com

http:

paths:

- backend:

serviceName: jenkins

servicePort: jenkins

path: /

EOF

### 创建Jenkins

kubectl apply -f /usr/local/kubernetes/jenkins/jenkins-rbac.yaml

kubectl apply -f /usr/local/kubernetes/jenkins/jenkins-deploy.yaml

## 初始化

### 初始化配置

docker exec -it jenkins bash

cat /var/jenkins\_home/secrets/initialAdminPassword

# 然后设置用户名和密码、邮箱即可

### 插件下载源

将下载源替换为 <https://mirrors.tuna.tsinghua.edu.cn/jenkins/updates/update-center.json>

### 中文设置

如果初始化时没有安装推荐插件，则需要手动安装插件初始化。

在插件中搜索 Localization 和 Localization: Chinese (Simplified) 两者都安装后重启即可。

### 权限管理

在插件中搜索 Role-based Authorization Strategy 安装。

安装完成后在全局安全配置的授权策略选择Role-Based Strategy，即可使用该插件了。找到Manage and Assign Roles即可开始使用。

### 凭证管理

凭证可以存储需要加密保护的数据库密码、Gitlab密码、Docker镜像仓库密码。

在插件中搜索 Credentials Binding 安装。

安装完成后首页已经会出现 凭证 的选项，可以开始使用了。

### 其他插件

Subversion

### 导入CA证书

添加IP解析

cat >> /etc/hosts << EOF

192.168.1.254 harbor.98.com

EOF

## Jenkins迁移与备份

直接压缩打包挂载出的 /usr/local/jenkins 即可。如果是Docker自动生成的目录，使用 docker cp 命令将 /var/jenkins\_home 复制出即可。