目录

4.1创建rancher目录

4.2安装

5.安装jdk（如果有jdk，此步骤可忽略）

5.1上传文件

5.2安装

6.安装Jenkins（安装前请确认系统已安装jdk）

6.1上传文件

6.2安装

6.3启动jenkins

6.4jenkins重启和停止命令如下

问题：

启动jenkins时报错：

解决方法：

初衷
此次发表该博客主要是由于博主在一周的rancher离线搭建的过程中，踩了很多的坑，由于网络上没有一个完整的搭建过程，顾写下这篇博客供大家参考。

前期准备
开始搭建之前推荐大家根据以下链接下载博主准备好的一整套的安装文件包，当然你也可以使用自己下载的安装文件。

🔗如下：

https://pan.baidu.com/s/1Vp8R0Ac8KLHw2KlOiqtK8A
提取码：n46d

注意：所有安装脚本建议开发者在linux系统中自己创建脚本文件，不建议直接上传使用部署文件夹中的脚本，因为Windows系统编辑的脚本文件在linux系统中可能存在不兼容现象。

1.安装docker
1.1.在服务器上创建docker目录
mkdir /home/tomcat/docker
--这个地方需要改成 mkdir -p /home/tomcat/docker 这样父目录不存在的情况下也都会创建对应的目录

1.2.进入docker目录
cd /home/tomcat/docker

1.3. 准备docker离线包
下载需要安装的docker版本，我此次下载的是docker-19.03.4.tgz，你只需要将该文件夹的中docker-19.03.4.tgz上传至创建好的docker目录

1.4.准备docker.service 系统配置文件
docker.service
[Unit]
Description=Docker Application Container Engine
Documentation=https://docs.docker.com
After=network-online.target firewalld.service
Wants=network-online.target
[Service]
Type=notify
# the default is not to use systemd for cgroups because the delegate issues still
# exists and systemd currently does not support the cgroup feature set required
# for containers run by docker
ExecStart=/usr/bin/dockerd
ExecReload=/bin/kill -s HUP $MAINPID
# Having non-zero Limit*s causes performance problems due to accounting overhead
# in the kernel. We recommend using cgroups to do container-local accounting.
LimitNOFILE=infinity
LimitNPROC=infinity

```
LimitCORE=infinity
# Uncomment TasksMax if your systemd version supports it.
# Only systemd 226 and above support this version.
#TasksMax=infinity
TimeoutStartSec=0
# set delegate yes so that systemd does not reset the cgroups of docker containers
Delegate=yes
# kill only the docker process, not all processes in the cgroup
KillMode=process
# restart the docker process if it exits prematurely
Restart=on-failure
StartLimitBurst=3
StartLimitInterval=60s
[Install]
WantedBy=multi-user.target
```

1.5.安装脚本

```
#!/bin/sh
echo '解压tar包...'
tar -xvf $1
echo '将docker目录移到/usr/bin目录下...'
cp docker/* /usr/bin/
echo '将docker.service 移到/etc/systemd/system/ 目录...'
cp docker.service /etc/systemd/system/
echo '添加文件权限...'
chmod +x /etc/systemd/system/docker.service
echo '重新加载配置文件...'
systemctl daemon-reload
echo '启动docker...'
systemctl start docker
echo '设置开机自启...'
systemctl enable docker.service
echo 'docker安装成功...'
docker -v
```

1.6.卸载脚本 uninstall.sh

```
#!/bin/sh
echo '删除docker.service...'
rm -f /etc/systemd/system/docker.service
```

```
echo '删除docker文件...'
rm -rf /usr/bin/docker*
echo '重新加载配置文件'
systemctl daemon-reload
echo '卸载成功...'
```

1.7.安装

此时目录为：（只需要关注docker-19.03.4.tgz、docker.service、install.sh、uninstall.sh即可）

执行脚本 sh install.sh docker-19.03.4.tgz

待脚本执行完毕后，执行 docker -v

出现如图所示，说明docker安装成功。

1.8.卸载

如果你想卸载docker，此时执行脚本 sh uninstall.sh 即可。

2.安装docker-compose

docker-compose的版本需要换成高版本的才行，要不然回提示报错，还有就是缺少对应的docker-compose.yml文件

2.1.上传安装文件

将安装文件夹中的docker-compose-Linux-x86_64文件上传到/home/tomcat/docker

2.2.安装

2.2.1执行命令，将安装文件复制到/usr/local/bin/目录下

```
cp docker-compose-Linux-x86_64 /usr/local/bin/docker-compose
```

2.2.2执行命令，将该文件赋为可执行文件：

```
chmod +x /usr/local/bin/docker-compose
```

2.2.3执行命令创建软链：

```
ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

2.2.4测试是否安装成功：

docker-compose --version

2.2.5如果想要卸载docker-compose只需执行命令：

rm -rf /usr/local/bin/docker-compose

3.安装Harbor私有仓库

3.1上传文件

将文件夹中的harbor-offline-installer-v1.9.3.tgz压缩文件上传到/home/tomcat目录

3.2安装

3.2.1解压harbor-offline-installer-v1.9.3.tgz文件

tar zxvf harbor-offline-installer-v1.9.3.tgz

3.2.2修改harbor.yml文件

1进入解压harbor-offline-installer-v1.9.3.tgz文件后出现的harbor文件夹

cd /home/tomcat/harbor

2修改harbor.yml文件，将hostname改成服务器的IP地址,port改成8800（防止与rancher端口冲突即可），可选择性修改harbor_admin_password属性：admin账户密码；测试环境使用http协议，不使用CA证书

3.2.3执行命令启动harbor

./install.sh

3.2.4启动完成后即可通过http://xxx.xxx.xxx.xxx:8800登录harbor图形化界面

3.2.5登录harbor，新增用户

3.2.6设置管理员

3.2.7修改daemon.json文件

vim /etc/docker/daemon.json
daemon.json文件中添加如下json(xxx.xxx.xxx.xxx为你的ip地址)

{
"registry-mirrors":["https://docker.mirrors.ustc.edu.cn"],
"insecure-registries":["https://xxx.xxx.xxx.xxx:8800"]
}
执行命令保存并重启容器

systemctl daemon-reload
systemctl restart docker
docker-compose up -d
进入harbor目录，执行命令刷新harbor配置

cd /home/tomcat/harbor
./prepare
./install.sh
使用命令登录docker

docker login xxx.xxx.xxx.xxx:8800
Username和Password输入刚刚创建的用户和密码即可
--输入密码的时候是看不到输入效果的，只需要正常输入自己的密码后点击回车即可

如果想要登出，可执行如下命令

docker logout xxx.xxx.xxx.xxx:8800
4.安装rancher
4.1创建rancher目录
mkdir /home/tomcat/rancher

4.2安装

4.2.1进入rancher目录

cd /home/tomcat/rancher

4.2.2准备部署文件，本次安装使用rancher2.4.2

1.将文件夹中的rancher-images.txt和rancher-images.tar.gz压缩文件上传到该目录

2.创建rancher-load-images.sh脚本

```
#!/bin/bash
images="rancher-images.tar.gz"
list="rancher-images.txt"
windows_image_list=""
windows_versions="1903"
usage () {
echo "USAGE: $0 [--images rancher-images.tar.gz] --registry my.registry.com:5000"
echo " [-l|--image-list path] text file with list of images; one image per line."
echo " [-i|--images path] tar.gz generated by docker save."
echo " [-r|--registry registry:port] target private registry:port."
echo " [--windows-image-list path] text file with list of images used in Windows. Windows image mirroring is skipped when this is empty"
echo " [--windows-versions version] Comma separated Windows versions. e.g., \"1809,1903\". (Default \"1903\")"
echo " [-h|--help] Usage message"
}
push_manifest () {
export DOCKER_CLI_EXPERIMENTAL=enabled
manifest_list=()
for i in "${arch_list[@]}"
do
manifest_list+=("$1-${i}")
done
echo "Preparing manifest $1, list[${arch_list[@]}]"
docker manifest create "$1" "${manifest_list[@]}" --amend
docker manifest push "$1" --purge
}
```

```
while [[ $# -gt 0 ]]; do
key="$1"
case $key in
-r|--registry)
reg="$2"
shift # past argument
shift # past value
;;
-l|--image-list)
list="$2"
shift # past argument
shift # past value
;;
-i|--images)
images="$2"
shift # past argument
shift # past value
;;
--windows-image-list)
windows_image_list="$2"
shift # past argument
shift # past value
;;
--windows-versions)
windows_versions="$2"
shift # past argument
shift # past value
;;
-h|--help)
help="true"
shift
;;
*)
usage
exit 1
;;
esac
```

```
done
if [[ -z $reg ]]; then
usage
exit 1
fi
if [[ $help ]]; then
usage
exit 0
fi
docker load --input ${images}
linux_images=()
while IFS= read -r i; do
[ -z "${i}" ] && continue
linux_images+=("${i}");
done < "${list}"
arch_list=()
if [[ -n "${windows_image_list}" ]]; then
IFS=',' read -r -a versions <<< "$windows_versions"
for version in "${versions[@]}"
do
arch_list+=("windows-${version}")
done
windows_images=()
while IFS= read -r i; do
[ -z "${i}" ] && continue
windows_images+=("${i}")
done < "${windows_image_list}"
# use manifest to publish images only used in Windows
for i in "${windows_images[@]}"; do
if [[ ! " ${linux_images[@]}" =~ " ${i}" ]]; then
case $i in
*/*)
image_name="${reg}/${i}"
;;
*)
image_name="${reg}/rancher/${i}"
;;
```

```
esac
push_manifest "${image_name}"
fi
done
fi
arch_list+=("linux-amd64")
for i in "${linux_images[@]}"; do
[ -z "${i}" ] && continue
arch_suffix=""
use_manifest=false
if [[ (-n "${windows_image_list}") && " ${windows_images[@]}" =~ " ${i}" ]]; then
# use manifest to publish images when it is used both in Linux and Windows
use_manifest=true
arch_suffix="-linux-amd64"
fi
case $i in
*/*)
image_name="${reg}/${i}"
;;
*)
image_name="${reg}/rancher/${i}"
;;
esac
docker tag "${i}" "${image_name}${arch_suffix}"
docker push "${image_name}${arch_suffix}"
if $use_manifest; then
push_manifest "${image_name}"
fi
done
```

3创建rancher-push-images.sh脚本

```
#!/bin/bash
## 镜像上传说明
# 需要先在镜像仓库中创建 rancher 项目
# 根据实际情况更改以下私有仓库地址
# 定义日志
workdir='../pwd'
```

```
log_file=${workdir}/sync_images_$(date +"%Y-%m-%d").log
logger()
{
log=$1
cur_time='['$(date +"%Y-%m-%d %H:%M:%S")']'
echo ${cur_time} ${log} | tee -a ${log_file}
}
images_hub() {
while true; do
read -p "输入镜像仓库地址(不加http/https): " registry
read -p "输入镜像仓库用户名: " registry_user
read -p "输入镜像仓库用户密码: " registry_password
echo "您设置的仓库地址为: ${registry},用户名: ${registry_user},密码: xxx"
read -p "是否确认(Y/N): " confirm
if [ $confirm != Y ] && [ $confirm != y ] && [ $confirm == '' ]; then
echo "输入不能为空，重新输入"
else
break
fi
done
}
images_hub
echo "镜像仓库 $(docker login -u ${registry_user} -p ${registry_password} ${registry})"
images=$(docker images -a | grep -v TAG | awk '{print $1 ":" $2}')
namespace=rancher
docker_push() {
for imgs in $(echo ${images}); do
n=$(echo ${imgs} | awk -F"/" '{print NF-1}')
#如果镜像名中没有/，那么此镜像一定是library仓库的镜像;
if [ ${n} -eq 0 ]; then
img_tag=${imgs}
#namespace=rancher
#重命名镜像
docker tag ${imgs} ${registry}/${namespace}/${img_tag}
#删除原始镜像
#docker rmi ${imgs}
#上传镜像
```

```
docker push ${registry}/${namespace}/${img_tag}
#如果镜像名中有一个/，那么/左侧为项目名，右侧为镜像名和tag
elif [ ${n} -eq 1 ]; then
img_tag=$(echo ${imgs} | awk -F"/" '{print $2}')
#namespace=$(echo ${imgs} | awk -F"/" '{print $1}')
#重命名镜像
docker tag ${imgs} ${registry}/${namespace}/${img_tag}
#删除旧镜像
#docker rmi ${imgs}
#上传镜像
docker push ${registry}/${namespace}/${img_tag}
#如果镜像名中有两个/,
elif [ ${n} -eq 2 ]; then
img_tag=$(echo ${imgs} | awk -F"/" '{print $3}')
#namespace=$(echo ${imgs} | awk -F"/" '{print $2}')
#重命名镜像
docker tag ${imgs} ${registry}/${namespace}/${img_tag}
#删除旧镜像
#docker rmi ${imgs}
#上传镜像
docker push ${registry}/${namespace}/${img_tag}
else
#标准镜像为四层结构，即：仓库地址/项目名/镜像名:tag,如不符合此标准，即为非有效镜像。
echo "No available images"
fi
done
}
docker_push
```

完成以后如下图

4.2.3赋予rancher-load-images.sh，rancher-push-images.sh可执行权限

```
chmod +x rancher-load-images.sh
chmod +x rancher-push-images.sh
```

4.2.4登录私有仓库

docker logout xxx.xxx.xxx.xxx:8800

4.2.5使用脚本rancher-load-images.sh根据rancher-images.txt提取rancher-images.tar.gz文件中的镜像，根据文件rancher-images.txt中的镜像列表对提取的镜像文件重新打 tag（xxx.xxx.xxx.xxx替换为你的i地址）

./rancher-load-images.sh --image-list ./rancher-images.txt --registry xxx.xxx.xxx.xxx:8800

4.2.6执行脚本rancher-push-images.sh将镜像上传至私有镜像仓库

./rancher-push-images.sh

--执行这个脚本的时候，要先到harbor中创建rancher这个名称的公开项目才行，要不让回报错

denied: requested access to the resource is denied

4.2.7执行脚本启动rancher（注意：启动rancher要确认80端口不被其他线程占用）

docker run -d --restart=unless-stopped -p 80:80 -p 443:443 10.2.33.182:8800/rancher/rancher:v2.4.2

4.2.8使用https://xxx.xxx.xxx.xxx登录rancher

4.2.9首次登录会要求设置密码，按照提示设置即可。

5.安装jdk（如果有jdk，此步骤可忽略）

5.1上传文件

将jdk-8u51-linux-x64.tar.gz压缩文件上传到/home/tomcat

5.2安装

5.2.1在/home/tomcat文件夹下执行命令解压压缩文件

tar -zxvf jdk-8u51-linux-x64.tar.gz

5.2.2将解压后的文件改名为jdk1.8

mv jdk1.8.0_51 jdk1.8

5.2.3配置jdk环境变量

1打开profile文件编辑

vim /etc/profile

2在profile文件中添加环境变量

#设定jdk环境
export JAVA_HOME=/home/tomcat/jdk1.8
export PATH=$JAVA_HOME/bin:$PATH
export CLASSPATH=.:$JAVA_HOME/lib
3刷新系统的环境变量

source /etc/profile
6.安装Jenkins（安装前请确认系统已安装jdk）
6.1上传文件
将文件夹中的jenkins-2.249.3-1.1.noarch.rpm文件上传到/home/tomcat目录

6.2安装
6.2.1执行命令安装jenkins

rpm -ivh jenkins-2.249.3-1.1.noarch.rpm

6.2.2安装完成后编辑环境变量文件

vim /etc/profile
6.2.3在最后一行，添加 /jenkins 为自定义目录，即jenkins工作目录,保存退出

export JENKINS_HOME=/home/tomcat/jenkins
6.2.4刷新环境变量

source /etc/profile
6.2.5安装完成后编辑jenkins的配置文件

vim /etc/sysconfig/jenkins
6.2.6在jenkins配置文件中修改以下内容并保存，此处为修改Jenkins的端口号，你也可以修改为其他。

JENKINS_PORT=”9999”
--很多时候默认的端口都没有开发，要用如下的方法去开放对应的端口才行
https://blog.csdn.net/xiannvbushengqi/article/details/105516504
6.2.7刷新jenkins配置

source /etc/sysconfig/jenkins

6.3启动jenkins

service jenkins start

6.4jenkins重启和停止命令如下

service jenkins restart

service jenkins stop

本次部署到此结束~

问题：

启动jenkins时报错：

Starting jenkins (via systemctl): Job for jenkins.service failed because the control process exited with error code. See "systemctl status jenkins.service" and "journalctl -xe" for details.

解决方法：

1.输入命令查看java环境变量：echo $JAVA_HOME，将查询到的地址复制保存

2.输入vim /etc/init.d/jenkins 在candidates之后添加：刚刚查询到的java环境变量+/bin/java

如：/home/jdk1.8//bin/java

3.输入systemctl start jenkins重启，发现警告：

Warning: jenkins.service changed on disk. Run 'systemctl daemon-reload' to reload units.

4.输入systemctl daemon-reload 刷新配置

5.输入systemctl restart jenkins 重启解决

————————————————