介绍

devops：5台服务器；centos7.9操作系统，64位

- gitlab, 代码仓库
- jenkins, 发布服务
- harbor, docker仓库服务
- 这里以发布hyperf 服务到两台机器为演示

资源准备

机器	服务	说明
10.4.7.100	gitlab	代码仓库，安装gitlab
10.4.7.101	jenkins	jenkins服务，安装jenkins, docker
10.4.7.102	harbor	docker仓库, 安装docker,harbor
10.4.7.103	hyperf服务	正常程序服务机器
10.4.7.104	hyperf服务	正常程序服务机器

gitlab安装

1. 安装相关依赖

```
yum -y install policycoreutils openssh-server openssh-clients postfix
```

2. 启动ssh服务&设置为开机启动

```
systemctl enable sshd && sudo systemctl start sshd
```

3. 设置postfix开机自启，并启动，postfix支持gitlab发信功能

```
systemctl enable postfix && systemctl start postfix
```

4. 开放ssh以及http服务，然后重新加载防火墙列表

```
firewall-cmd --add-service=ssh --permanent
firewall-cmd --add-service=http --permanent
firewall-cmd --reload
```

如果关闭防火墙就不需要做以上配置

5. 下载gitlab包，并且安装
   在线下载安装包：

```
wget https://mirrors.tuna.tsinghua.edu.cn/gitlab-ce/yum/el6/gitlab-ce-12.4.2-ce.0.el6.x86_64.rpm
```

安装：

```
rpm -i gitlab-ce-12.4.2-ce.0.el6.x86_64.rpm
```

6. 修改gitlab配置

```
vi /etc/gitlab/gitlab.rb
```

修改gitlab访问地址和端口，默认为80，我们改为82

```
external_url 'http://10.4.7.100:82'
nginx['listen_port'] = 82
```

7. 重载配置及启动gitlab

```
gitlab-ctl reconfigure
gitlab-ctl restart
```

8. 把端口添加到防火墙

```
firewall-cmd --zone=public --add-port=82/tcp --permanent
firewall-cmd --reload
```

启动成功后，看到以下修改管理员root密码的页面，修改密码后，然后登录即可

jenkins安装

安装docker

> 参考辅助安装里的docker安装

安装jenkins

安装jenkins

1）安装JDK

Jenkins需要依赖JDK，所以先安装JDK1.8

```
yum install java-1.8.0-openjdk* -y
```

安装目录为：/usr/lib/jvm

2）获取jenkins安装包

下载页面：https://jenkins.io/zh/download/

http://mirrors.jenkins-ci.org/redhat-stable/

安装文件：jenkins-2.190.3-1.1.noarch.rpm

3）把安装包上传到10.4.7.101服务器，进行安装

```
wget http://mirrors.jenkins-ci.org/redhat-stable/jenkins-2.190.3-1.1.noarch.rpm
rpm -ivh jenkins-2.190.3-1.1.noarch.rpm
```

4）修改Jenkins配置

```
vim /etc/sysconfig/jenkins
```

修改内容如下：

```
JENKINS_USER="root"

JENKINS_PORT="8888"
```

5）启动Jenkins

```
systemctl start jenkins
```

6）打开浏览器访问

http://10.4.7.101:8888注意：本服务器把防火墙关闭了，如果开启防火墙，需要在防火墙添加端口

```
war包
/usr/lib/jenkins/jenkins.war
配置文件  （端口号、jenkins_home目录等）
/etc/sysconfig/jenkins
默认的JENKINS_HOME目录
/var/lib/jenkins/
Jenkins日志文件地址
/var/log/jenkins/
真正的服务文件
/etc/init.d/jenkins
```

7）获取并输入admin账户密码

```
cat /var/lib/jenkins/secrets/initialAdminPassword
```

8）跳过插件安装

因为Jenkins插件需要连接默认官网下载，速度非常慢，而且经过会失败，所以我们暂时先跳过插件安装

9）添加一个管理员账户，并进入Jenkins后台

安装jenkins的常用插件

Jenkins国外官方插件地址下载速度非常慢，所以可以修改为国内插件地址：

Jenkins->Manage Jenkins->Manage Plugins，点击Available

这样做是为了把Jenkins官方的插件列表下载到本地，接着修改地址文件，替换为国内插件地址

```
cd /var/lib/jenkins/updates
sed -i 's/http:\/\/updates.jenkins-
ci.org\/download/https:\/\/mirrors.tuna.tsinghua.edu.cn\/jenkins/g' default.json && sed
-i 's/http:\/\/www.google.com/https:\/\/www.baidu.com/g' default.json
```

最后，Manage Plugins点击Advanced，把Update Site改为国内插件下载地址

```
https://mirrors.tuna.tsinghua.edu.cn/jenkins/updates/update-center.json
```

Sumbit后，在浏览器输入： http://10.4.7.101:8888/restart ，重启Jenkins。

插件列表

Jenkins->Manage Jenkins->Manage Plugins，点击Available，搜索"Chinese"

插件名称	作用
Chinese	汉化
Role-based Authorization Strategy	用户权限管理
Credentials Binding	凭证管理【重点】
Git	拉取代码
Pipeline	流水线式创建项目，也可以在项目里创建Jenkinsfile的方式部署代码
GitLab和Gitlab Hook	gitlab仓库代码更新后，发送通知给jenkins，然后jenkins自动部署代码
Email Extension	构建成功后，发送邮件通知
Publish Over SSH	可以实现远程发送Shell命令
Extended Choice Parameter	支持多选框，构建参数

jenkins服务器

> 安装git
>
> yum install -y git
>
> ssh-keygen -t rsa -C "your_email@youremail.com"
>
> cat /root/.ssh/id_rsa.pub
>
> 将内容拷贝到gitlab服务器上

harbor安装

安装docker

> 参考辅助安装里的docker安装

安装harbor

1）先安装Docker并启动Docker（已完成）

参考之前的安装过程

2）先安装docker-compose

```
sudo curl -L https://github.com/docker/compose/releases/download/1.21.2/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose
```

3）给docker-compose添加执行权限

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

4）查看docker-compose是否安装成功

```
docker-compose -version
```

5）下载Harbor的压缩包（本课程版本为：v1.9.2）

```
https://github.com/goharbor/harbor/releases
```

6）上传压缩包到linux，并解压

```
wget https://github.com/goharbor/harbor/releases/download/v2.2.1/harbor-offline-
installer-v2.2.1.tgz
tar -zxf harbor-offline-installer-v2.2.1.tgz
mkdir /opt/harbor
mv harbor/* /opt/harbor
cd /opt/harbor
```

7）修改Harbor的配置

```
cp harbor.yml.tmpl harbor.yml

vi harbor.yml
```

修改hostname和port

```
hostname: 10.4.7.102
port: 85
```

8）安装Harbor

```
./prepare
./install.sh
```

错误

原因是harbor.yml中默认是配置https的端口及证书路径的。解决办法是把这些配置都注释掉。

```
# https related config
#https:
  # https port for harbor, default is 443
 # port: 443
  # The path of cert and key files for nginx
 # certificate: /your/certificate/path
 # private_key: /your/private/key/path
```

```
vim /etc/docker/daemon.json
{
        "registry-mirrors": ["https://8h1zf55z.mirror.aliyuncs.com"],
        "insecure-registries":["10.4.7.102:5000"]
}
systemctl restart docker
```

9）启动Harbor

```
docker-compose up -d 启动
docker-compose stop 停止
docker-compose restart 重新启动
```

10）访问Harbor

http://10.4.7.102:85

默认账户密码：admin/Harbor12345

11）创建项目和用户

1. 项目：tensquare
2. 用户：harborup/Harborup123
3. 进入 tensquare项目，给这个私有的项目，分配harborup这个用户

jenkins发布到docker集群

安装docker

103, 104 都安装docker，参考辅助安装里的docker安装；并且要配置拉去102服务器的配置

1. 安装docker
2. 修改配置
3. 101, 103, 104都要和102, harbor 服务器进行交互。增加如下配置

```
vim /etc/docker/daemon.json
{
"registry-mirrors": ["https://8h1zf55z.mirror.aliyuncs.com"],
"insecure-registries": ["10.4.7.102:85"]
}
systemctl restart docker
```

步骤说明

> 1.代码上传到gitlab -> 2.从gitlab拉取源码->3.使用Dockerfile编译、生成镜像->4.上传到Harbor镜像仓库->
>
> 5.拉取镜像和发布应用

发布 hyperf

步骤

> 1）项目代码上传到Gitlab
>
> 2）从Gitlab拉取项目源码
>
> 3）使用Dockerfile编译、生成镜像
>
> 4）上传到Harbor镜像仓库
>
> 5)）拉取镜像和发布应用

Dockerfile

```
# Default Dockerfile
#
# @link     https://www.hyperf.io
# @document https://hyperf.wiki
# @contact  group@hyperf.io
# @license  https://github.com/hyperf/hyperf/blob/master/LICENSE

FROM hyperf/hyperf:7.4-alpine-v3.10-swoole-v4
LABEL maintainer="Hyperf Developers <group@hyperf.io>" version="1.0" license="MIT"
app.name="Hyperf"

##
# ---------- env settings ----------
##
# --build-arg timezone=Asia/Shanghai
ARG timezone

ENV TIMEZONE=${timezone:-"Asia/Shanghai"} \
```

```
    APP_ENV=prod \
    SCAN_CACHEABLE=(true)

# update
RUN set -ex \
    # show php version and extensions
    && php -v \
    && php -m \
    && php --ri swoole \
    #  ---------- some config ----------
    && cd /etc/php7 \
    # - config PHP
    && { \
        echo "upload_max_filesize=128M"; \
        echo "post_max_size=128M"; \
        echo "memory_limit=1G"; \
        echo "date.timezone=${TIMEZONE}"; \
    } | tee conf.d/99_overrides.ini \
    # - config timezone
    && ln -sf /usr/share/zoneinfo/${TIMEZONE} /etc/localtime \
    && echo "${TIMEZONE}" > /etc/timezone \
    # ---------- clear works ----------
    && rm -rf /var/cache/apk/* /tmp/* /usr/share/man \
    && echo -e "\033[42;37m Build Completed :).\033[0m\n"

WORKDIR /opt/www

# Composer Cache
# COPY ./composer.* /opt/www/
# RUN composer install --no-dev --no-scripts

COPY . /opt/www
RUN composer install --no-dev -o && php bin/hyperf.php

EXPOSE 9501

ENTRYPOINT ["php", "/opt/www/bin/hyperf.php", "start"]
```

Jenkinsfile

```
//gitlab的凭证
def git_auth = "63d52aaa-b5b3-45e0-8ded-eccb650215bd"

def git_url = "git@e.coding.net:jidi/hyperf-test/hyperf-test.git"

//构建版本的名称
def tag = "latest"
```

```
//Harbor私服地址
def harbor_url = "10.4.7.102:85"

//Harbor的项目名称
def harbor_project_name = "tensquare"

//Harbor的凭证
def harbor_auth = "6a955862-9af6-4e81-9f38-177a640b9b4a"

//项目名称
def project_name = "hyperf-test"

node {

    //把选择的服务区信息转为数组
    def selectedServers = "${publish_server}".split(',')

    stage('拉取代码') {
        checkout([$class: 'GitSCM', branches: [[name: '*/${branch}']],
doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [],
userRemoteConfigs: [[credentialsId: "${git_auth}", url: "${git_url}"]]])
    }
    stage('编译，构建镜像，部署服务') {
        //定义镜像名称
        def imageName = "${project_name}:${tag}"

        //准备配置文件
        sh "cp .env.example .env"

        //编译，构建本地镜像
        sh "docker build -t ${imageName} ."

        //给镜像打标签
        sh "docker tag ${imageName} ${harbor_url}/${harbor_project_name}/${imageName}"

        //登录Harbor，并上传镜像
        withCredentials([usernamePassword(credentialsId: "${harbor_auth}",
passwordVariable: 'password', usernameVariable: 'username')]) {
            //登录
            sh "docker login -u ${username} -p ${password} ${harbor_url}"
            //上传镜像
            sh "docker push ${harbor_url}/${harbor_project_name}/${imageName}"

        }

        //删除本地镜像
        sh "docker rmi -f ${imageName}"
        sh "docker rmi -f ${harbor_url}/${harbor_project_name}/${imageName}"
```

```
        //=====以下为远程调用进行项目部署========
        for(int j=0;j<selectedServers.size();j++){
            //每个服务名称
            def currentServer = selectedServers[j]

            //发布到指定的服务器
            sshPublisher(publishers: [sshPublisherDesc(configName: "${currentServer}",
transfers: [sshTransfer(cleanRemote: false, excludes: '', execCommand:
"/opt/jenkins_shell/deploy.sh $harbor_url $harbor_project_name $project_name $tag
$port", execTimeout: 120000, flatten: false, makeEmptyDirs: false, noDefaultExcludes:
false, patternSeparator: '[, ]+', remoteDirectory: '', remoteDirectorySDF: false,
removePrefix: '', sourceFiles: '')], usePromotionTimestamp: false,
useWorkspaceInPromotion: false, verbose: false)])
        }
    }
}
```

deploy.sh

发布脚本， 在10.4.7.103机器下；代码在哪台运行，发布在哪台。

```
#! /bin/sh
#接收外部参数
harbor_url=$1
harbor_project_name=$2
project_name=$3
tag=$4
port=$5

imageName=$harbor_url/$harbor_project_name/$project_name:$tag

echo "$imageName"

#查询容器是否存在，存在则删除
containerId=`docker ps -a | grep -w ${project_name}:${tag} | awk '{print $1}'`
if [ "$containerId" != "" ] ; then
  #停掉容器
  docker stop $containerId
    #删除容器
    docker rm $containerId
    echo "成功删除容器"
 fi

 #查询镜像是否存在，存在则删除
 imageId=`docker images | grep -w $project_name | awk '{print $3}'`
 if [ "$imageId" != "" ] ; then
```

```
 #删除镜像
 docker rmi -f $imageId
 echo "成功删除镜像"
fi

# 登录Harbor私服
docker login -u harborup -p Harborup123 $harbor_url

# 下载镜像
docker pull $imageName

# 启动容器
docker run -d -p $port:$port $imageName

echo "容器启动成功"
```

放到 /opt/jenkins_shell下

```
chmod +x /opt/jenkins_shell/deploy.sh
```

pipline发布

> 新建任务->流水线->(1)参数化构建过程 ; 添加参数，字符参数; (2) 流水线, Pipeline script from SCM; git配置项目地址

增加链接用户

> 这里需要增加 gitlab用户, harbor的用户；在jenkins里配置.

配置远程部署服务器

1）拷贝公钥到远程服务器

10.4.7.101，jenkins机器上执行

```
ssh-copy-id 10.4.7.103
```

2）系统管理->系统配置->添加远程服务器

Publish over SSH

```
Path to key 值是 /root/.ssh/id_rsa

SSH Servers 值是 10.4.7.103的服务信息
name         -> master_server
hostname       -> 10.4.7.103
username       -> root
Remote Directory    -> /


Name是给sshserver自定义一个名称
Hostname是ssh服务器的地址
Username是ssh服务器的用户名
Remote Directory是需要上传文件到服务器的远程目录
```

3. 增加第二台服务器

```
ssh-copy-id 10.4.7.104
```

系统管理->系统配置->添加远程服务器-> Publish over SSH

```
SSH Servers 值是 10.4.7.103的服务信息
name         -> slave_server1
hostname       -> 10.4.7.104
username       -> root
Remote Directory    -> /
```

修改Docker配置信任Harbor私服地址

```
vim /etc/docker/daemon.json
{
"registry-mirrors": ["https://8h1zf55z.mirror.aliyuncs.com"],
"insecure-registries": ["10.4.7.102:85"]
}
systemctl restart docker
```

4）添加参数

这样可以支持构建多选框，从而发布多个服务或者发布多台机器

> 项目-> 配置-> 添加参数-> Extended Choice Parameter
>
> 
>
> 效果

添加发布参数

项目-> 参数化构建过程 -> branch:master;port:9501

补充

1. 错误1: jenkins机器上

```
fatal: unable to access 'https://github.com/.../.git': Could not resolve host:
github.com

git config --global --unset http.proxy
git config --global --unset https.proxy
```

2. 初始化镜像拉取

> 1. 发布hyperf时候，因为要先拉官方的镜像：docker pull hyperf/hyperf:7.4-alpine-v3.10-swoole-v4
>
>   可以将上面的镜像在其他服务器上获取到，然后上传到102, harbor服务器上；防止链接外网过慢获取的问题.

辅助安装

安装docker

1）卸载旧版本

```
yum list installed | grep docker 列出当前所有docker的包
yum -y remove docker的包名称 卸载docker包
rm -rf /var/lib/docker 删除docker的所有镜像和容器
```

2）安装必要的软件包

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

3）设置下载的镜像仓库

```
sudo yum-config-manager  --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

4）列出需要安装的版本列表

```
yum list docker-ce --showduplicates | sort -r
```

```
docker-ce.x86_64 3:18.09.1-3.el7 docker-ce-stable
docker-ce.x86_64 3:18.09.0-3.el7 docker-ce-stable
docker-ce.x86_64 18.06.1.ce-3.el7 docker-ce-stable
docker-ce.x86_64 18.06.0.ce-3.el7 docker-ce-stable
......
```

5）安装指定版本（这里使用18.0.1版本）

```
sudo yum install docker-ce-18.06.1.ce -y
```

6）查看版本

```
docker -v
```

7）启动Docker

```
sudo systemctl start docker
sudo systemctl enable docker
```

8）添加阿里云镜像下载地址

```
vi /etc/docker/daemon.json
```

内容如下：

```
{"registry-mirrors": ["https://8h1zf55z.mirror.aliyuncs.com"] }
```

9）重启Docker

```
sudo systemctl restart docker
```