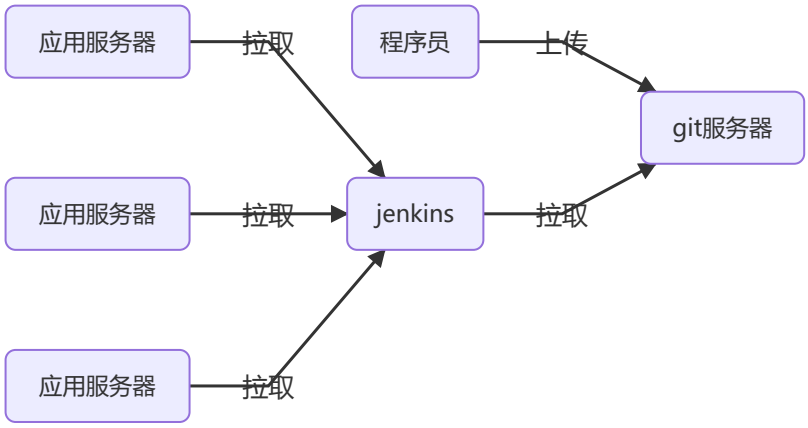
nsd1912-devops-day05

CI/CD，持续集成/持续交付



- 安装

```
[root@jenkins ~]# rpm -qa | grep java
java-1.8.0-openjdk-1.8.0.161-2.b14.el7.x86_64
[root@jenkins ~]# rpm -ivh jenkins-2.222.1-1.1.noarch.rpm
[root@jenkins ~]# systemctl start jenkins
[root@jenkins ~]# systemctl enable jenkins
# 访问 http://x.x.x.x:8080
```

- 在/var/lib/jenkins/secrets/initialAdminPassword取出密码 -> 选择插件来安装 -> 点击中间上面的“无”，不安装任何插件，点击右下角的“安装” -> 点击右下角的“使用admin继续” -> 点击“保存并完成”
  -> 点击"开始使用jenkins"
- 进入jenkins首页后，改admin密码

右上角的"admin" -> configure -> password

- 配置通过国内站点安装插件

```
[root@localhost ~]# ls /var/lib/jenkins/updates/default.json
[root@localhost ~]# sed -i 's/http:\/\/updates.jenkins-
ci.org\/download/https:\/\/mirrors.tuna.tsinghua.edu.cn\/jenkins/g'
/var/lib/jenkins/updates/default.json && sed -i
's/http:\/\/www.google.com/https:\/\/www.baidu.com/g'
/var/lib/jenkins/updates/default.json
[root@localhost ~]# systemctl restart jenkins
```

> 如果/var/lib/jenkins/updates/default.json不存在，可以 Manage Jenkins -> Manage Plugins -> Advanced选项卡 -> Update site : https://mirrors.tuna.tsinghua.edu.cn/jenkins/updates/update-center.json -> 点击submit提交。

插件

- 安装插件

Manage Jenkins -> Manage Plugins -> Available选项卡，按ctrl+f进行搜索并勾选Git Parameter / Localization Chinese(Simplified) / DingTalk / GitLab -> 点击Install without restart -> 勾选Restart Jenkins when installation is complete and no jobs are running

jenkins应用

- 在jenkins服务器上安装git

```
[root@jenkins ~]# yum install -y git
```

- 新建任务

首页 -> 新建Item -> myweb / Freestyle project -> 确定 -> 源码管理 => git / Repository URL: http://192.168.81.134/devops/myweb.git -> 保存

```
# 在jenkins服务器上查看代码目录
[root@localhost ~]# ls /var/lib/jenkins/workspace
ls: 无法访问/var/lib/jenkins/workspace: 没有那个文件或目录
```

点击项目页面左边栏的Build Now(立即构建) -> Build History 下面的#1 -> 左边栏的控制台输出

```
# 在jenkins服务器上查看代码目录
[root@localhost ~]# ls /var/lib/jenkins/workspace
myweb
```

配置机器人发送消息

- 项目构建过程中，可以将整个构建过程通过机器人发送消息。

首页 -> Manage Jenkins -> Configure System -> 系统配置 -> 新增机器人 -> 输入名字、webhook地址以及关键字 -> 点击测试 -> 成功后保存

> web登陆：https://im.dingtalk.com/

- 修改myweb项目，用钉钉机器人发送构建过程

首页 -> 点击项目 - > 左边栏 配置 -> 勾选机器人 -> 保存

构建项目时，构建过程将会通过机器人发送消息

推送代码时自动构建项目

- 修改jenkins项目配置 -> 构建触发器 -> 勾选Build when a change is pushed to GitLab. GitLab webhook URL: http://192.168.181.135:8080/project/myweb -> 点击 高级 -> 点击generate生成 Secret token并复制它 -> 保存
- 修改gitlab配置 -> 点击项目，如myweb -> 左边栏 设置 / 集成 -> 链接url http://192.168.81.135:8080/project/myweb / 安全令牌填写jenkins中生成的Secret token -> 点击增加web钩子。在页面中间部分找到创建的web钩子，点击test -> Push events测试，返回Hook executed successfully: HTTP 200表示成功。
- 测试

```
# 在jenkins服务器上删除构建目录
[root@localhost ~]# rm -rf /var/lib/jenkins/workspace/*

# 程序员推送代码
[root@node2 myprojects]# cd myweb/
[root@node2 myweb]# git mv index.htm index.html
[root@node2 myweb]# git commit -m "mv index.htm index.html"
[root@node2 myweb]# git push
```

程序员推送代码到gitlab服务器后，jenkins项目将会自动构建，并通过机器人发送构建消息。

```
[root@localhost ~]# ls /var/lib/jenkins/workspace/
myweb
```

使用参数git parameter构建某一版本的代码

- 配置jenkins通过tag标签构建

首页 -> 新建Item -> 名字myweb2 / Freestyle project -> 勾选This project is parameterized参数化构建 => 添加参数 => Git Parameter (Git参数) => Name: webver / Parameter Type: Branch or Tag / Default Value: origin/master -> 源码管理 => Git => Repositories => Repository url: http://192.168.81.134/devops/myweb.git / Branches to build: $webver -> 保存

- 构建测试

点击 Build with Parameters -> 选择版本 -> 开始构建

检出代码到子目录

```
# 删除jenkins服务器上的代码目录
[root@localhost ~]# rm -rf /var/lib/jenkins/workspace/myweb2/
```

在jenkins的web页面上点击配置 -> 在源码管理下面找到Addtional Behaviours -> 新增 -> Checkout to a sub-directory(检出到子目录): myweb-$webver -> 保存

- 构建测试：构建两次，每次使用不同版本

点击 Build with Parameters -> 选择版本 -> 开始构建

```
[root@localhost ~]# ls /var/lib/jenkins/workspace/myweb2/
myweb-1.0  myweb-2.0
```

修改项目，实现代码打包

- jenkins服务器通过http协议共享打包后的软件
- jenkins服务器发布当前软件版本和前一版本
- 计算压缩包的md5值

```
# 在jenkins服务器上安装httpd
[root@localhost ~]# yum install -y httpd
[root@localhost ~]# systemctl start httpd
[root@localhost ~]# systemctl enable httpd
# /var/www/html/deploy/live_ver：保存当前版本号
# /var/www/html/deploy/last_ver：保存前一版本的版本号
# /var/www/html/deploy/pkgs：保存软件的压缩包和它的md5值
[root@localhost ~]# mkdir -p /var/www/html/deploy/pkgs
[root@localhost ~]# chown -R jenkins:jenkins /var/www/html/deploy/
```

- myweb2项目 -> 配置 -> 构建 -> 增加构建项目 -> Execute shell

```
pkgs_dir=/var/www/html/deploy/pkgs
# 将下载目录拷贝到web服务器目录
cp -r myweb-$webver $pkgs_dir
cd $pkgs_dir
rm -rf myweb-$webver/.git  # 删除不必要的版本库文件
# 打包
tar czf myweb-$webver.tar.gz myweb-$webver
rm -rf myweb-$webver  # 删除软件目录
# 计算压缩包的md5值
md5sum myweb-$webver.tar.gz | awk '{print $1}' > myweb-$webver.tar.gz.md5
cd ..
# 将live_ver的版本号写到last_ver中
[ -f live_ver ] && cat live_ver > last_ver
echo -n $webver > live_ver  # 将最新版本号写入live_ver
```

- 构建测试

点击 Build with Parameters -> 选择版本 -> 开始构建

- 访问 http://jenkins服务器ip/deploy

部署软件到应用服务器

在web服务器上部署代码

- 下载相应版本的软件包
- 校验下载的软件包是否损坏
- 解压软件包
- 部署软件包到web服务器的文档目录

```
# 在应用服务器上执行以下操作
[root@localhost ~]# yum install -y httpd
[root@localhost ~]# systemctl start httpd
[root@localhost ~]# systemctl enable httpd
[root@localhost ~]# mkdir /var/www/{download,deploy}
# /var/www/download: 用于保存下载的tar包
# /var/www/deploy: 用于保存解压后的目录和live_ver文件
# /var/www/html/nsd1912: web服务器软链接
```

- 编写应用服务器自动上线代码

```
import wget
import requests
```

```
import os
import hashlib
import tarfile

def has_new_ver(ver_url, ver_fname):
    "用于检查是否有新版本，有则返回True，否则返回False"
    # 如果本地没有版本文件，则有新版本
    if not os.path.exists(ver_fname):
        return True

    # 如果本地和远程版本文件不一样，则有新版本
    with open(ver_fname) as fobj:
        local_ver = fobj.read()
    r = requests.get(ver_url)
    if local_ver != r.text:
        return True

    # 如果上述条件都不满足，则没有新版本
    return False


def check_file(md5url, fname):
    "用于检查文件是否损坏，未损坏返回True，否则返回False"
    # 计算本地文件的md5值
    m = hashlib.md5()
    with open(fname, 'rb') as fobj:
        while 1:
            data = fobj.read(4096)
            if not data:
                break
            m.update(data)

    # 取出网上md5值，进行比较
    r = requests.get(md5url)
    if m.hexdigest() == r.text.strip():  # 去除网上md5文件尾部\n
        return True

    # 如果本地和远程的md5值不一样，返回False
    return False

def deploy(app_fname, deploy_dir, dest):
    "用于部署程序到应用服务器"
    # 解压文件到目标
    tar = tarfile.open(app_fname)
    tar.extractall(path=deploy_dir)
    tar.close()

    # 拼接出解压目录的绝对路径
    app_dir = os.path.basename(app_fname)
    app_dir = app_dir.replace('.tar.gz', '')
    app_dir = os.path.join(deploy_dir, app_dir)

    # 创建快捷方式，如果快捷方式已存在，先删除它
    if os.path.exists(dest):
        os.remove(dest)

    os.symlink(app_dir, dest)
```

```
if __name__ == '__main__':
    # 检查是否有新版本，没有新版本退出
    ver_url = 'http://192.168.81.135/deploy/live_ver'
    ver_fname = '/var/www/deploy/live_ver'
    if not has_new_ver(ver_url, ver_fname):
        print('未发现新版本。')
        exit(1)

    # 下载新版本
    r = requests.get(ver_url)
    pkg_url = 'http://192.168.81.135/deploy/pkgs/myweb-%s.tar.gz' % r.text
    app_fname = '/var/www/download/myweb-%s.tar.gz' % r.text
    wget.download(pkg_url, app_fname)

    # 检查新版本程序文件是否损坏，如损坏则删除它并退出
    md5url = pkg_url + '.md5'
    if not check_file(md5url, app_fname):
        print('文件已损坏。')
        os.remove(app_fname)
        exit(2)

    # 部署软件
    deploy_dir = '/var/www/deploy'
    dest = '/var/www/html/nsd1912'
    deploy(app_fname, deploy_dir, dest)

    # 更新本地版本live_ver文件
    if os.path.exists(ver_fname):
        os.remove(ver_fname)
    wget.download(ver_url, ver_fname)
```

完整测试

- 程序员编写新版本代码，并上传

```
[root@node2 myweb]# vim index.html
2nd line
version 2.0
<br>
<img
src="http://n.sinaimg.cn/ent/4_img/upload/075ec3cc/10/w690h920/20200601/8270-iumkapv9072831.jpg">

[root@node2 myweb]# git add .
[root@node2 myweb]# git commit -m "app 4.0"
[root@node2 myweb]# git tag 4.0
[root@node2 myweb]# git push
[root@node2 myweb]# git push --tags
```

- 在jenkins上构建新版本
- 运行deploy.py