nsd1904_devops_day04

CI / CD：持续集成 / 持续交付

流行的自动化运维平台：docker / k8s / ansible / git / jenkins

准备三台机器

- node4.tedu.cn: 程序员PC
- node5.tedu.cn: gitlab服务器 -> 4GB以上内存
- node6.tedu.cn: jenkins

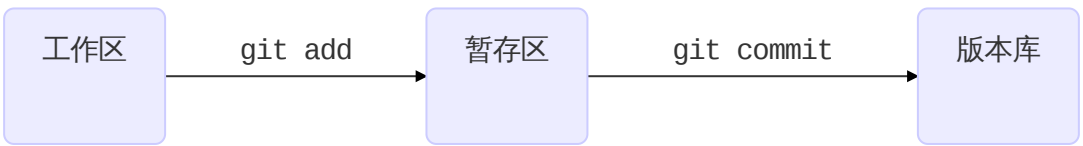
git

- 软件版本管理系统
- 分布式

```
# 安装
[root@node4 ~]# yum install -y git
# 配置基础信息
[root@node4 ~]# git config --global user.name "zzg"
[root@node4 ~]# git config --global user.email "zzg@tedu.cn"
[root@node4 ~]# git config --global core.editor vim
# 查看
[root@node4 ~]# git config --list
[root@node4 ~]# cat ~/.gitconfig
```

git的工作区域

- 工作区
- 暂存区
- 版本库



初始化版本库

- 项目之初，直接构建

```
[root@node4 ~]# git init mypro
初始化空的 Git 版本库于 /root/mypro/.git/
[root@node4 ~]# cd mypro/
[root@node4 mypro]# ls -A
.git
```

- 已有项目

```
[root@node4 ~]# mkdir myweb
[root@node4 ~]# cd myweb
[root@node4 myweb]# echo '<h1>Hello World</h1>' > index.html
[root@node4 myweb]# git init
初始化空的 Git 版本库于 /root/myweb/.git/
[root@node4 myweb]# ls -A
.git  index.html
```

添加文件到暂存区

```
# 查看状态
[root@node4 myweb]# git status
[root@node4 myweb]# git status -s
?? index.html

# 添加到暂存区
[root@node4 myweb]# git add .
[root@node4 myweb]# git status
[root@node4 myweb]# git status -s
A  index.html

# 从暂存区撤出文件
[root@node4 myweb]# git rm --cached index.html
[root@node4 myweb]# git status -s
?? index.html
```

确认至版本库

```
[root@node4 myweb]# git add .
[root@node4 myweb]# git status -s
A  index.html
[root@node4 myweb]# git commit  # 在跳出的vim中写日志
[root@node4 myweb]# git status
# 位于分支 master
无文件要提交，干净的工作区

# 继续提交
[root@node4 myweb]# cp /etc/hosts .
[root@node4 myweb]# git status -s
?? hosts
[root@node4 myweb]# git add hosts
[root@node4 myweb]# git status -s
```

```
A  hosts
[root@node4 myweb]# git commit -m "add hosts"
[root@node4 myweb]# git status
# 位于分支 master
无文件要提交，干净的工作区
```

文件改名

```
[root@node4 myweb]# git mv hosts zhuji
[root@node4 myweb]# ls
[root@node4 myweb]# git status
[root@node4 myweb]# git status -s
R  hosts -> zhuji
[root@node4 myweb]# git commit -m "mv hosts zhuji"
[root@node4 myweb]# git status
# 位于分支 master
无文件要提交，干净的工作区
```

删除文件

```
[root@node4 myweb]# git rm zhuji
[root@node4 myweb]# git status
[root@node4 myweb]# git commit -m "rm zhuji"
[root@node4 myweb]# git status
# 位于分支 master
无文件要提交，干净的工作区
```

git分支

- 默认git已有一个名为master的主干分支
- 编写新的功能、修复bug时，可以创建一个新分支
- 新分支功能编写完成后，还可以把它汇入主干
- 创建分支、汇入主干、切换分支等务必保证工作区是干净的

```
# 主干可以继续工作
[root@node4 myweb]# cp /etc/passwd mima
[root@node4 myweb]# git add .
[root@node4 myweb]# git commit -m "add mima"

# 查看分支
[root@node4 myweb]# git branch
* master

# 创建分支
[root@node4 myweb]# git branch b1
[root@node4 myweb]# git branch
  b1
* master    # *号表示，当前牌master分支

# 切换分支
```

```
[root@node4 myweb]# git checkout b1
切换到分支 'b1'
[root@node4 myweb]# git branch
* b1
  master

# 在分支中提交数据
[root@node4 myweb]# cp /etc/redhat-release .
[root@node4 myweb]# git add .
[root@node4 myweb]# git commit -m "add redhat-release"
[root@node4 myweb]# ls
index.html  mima  redhat-release

# 切换回master分支
[root@node4 myweb]# git checkout master
切换到分支 'master'
[root@node4 myweb]# git branch
  b1
* master
[root@node4 myweb]# ls
index.html  mima

# 将分支汇入主干
[root@node4 myweb]# git merge b1 -m "merge b1 to master"
[root@node4 myweb]# ls
index.html  mima  redhat-release

# 删除分支
[root@node4 myweb]# git branch -d b1
[root@node4 myweb]# git branch
* master
```

设置被**git**排除的文件

```
[root@node4 myweb]# vim .gitignore
*.swp
a.txt
.gitignore
[root@node4 myweb]# echo 'hello world' > a.txt
[root@node4 myweb]# git status
# 位于分支 master
无文件要提交，干净的工作区
```

返回到以前的状态

```
[root@node4 myweb]# git log
[root@node4 myweb]# git checkout 7582dae74f80eab75cf928ea72dfe9bb5cf70d9c
# 返回到最新状态
[root@node4 myweb]# git checkout master
```

gitlab服务器

- 准备4GB以上的服务器 node5.tedu.cn
- 安装docker软件并启动
- 导入gitlab镜像

启动容器

```
# 因为gitlab容器也需要使用22端口，所以将虚拟机ssh端口改为2022
[root@node5 ~]# vim /etc/ssh/sshd_config
Port 2022
[root@node5 ~]# systemctl restart sshd
# 重新登陆
[root@room8pc16 phase5]# ssh node5 -p2022

# 创建并启动容器
[root@node5 ~]# docker run -d -h gitlab --name gitlab -p 22:22 -p 80:80 -p 443:443 --
restart always -v /srv/gitlab/config:/etc/gitlab -v /srv/gitlab/logs:/var/log/gitlab -v
/srv/gitlab/data:/var/opt/gitlab gitlab_zh

# 查看状态，启动时间较长，当状态显示healthy时，容器才是可用的
[root@node5 ~]# docker ps
```

配置gitlab

1. 访问http://x.x.x.x
2. 第一次访问，需要为root用户设置密码
3. gitlab中重要的概念
    - 项目：软件项目
    - 组：为开发团队创建组
    - 成员：一个组中可以包含多个用户
4. 创建名为devops的公开类型的组
5. 创建名为myweb的公开类型的项目，为群组创建
6. 创建用户，将其设置为myweb项目的主程序员。新建用户时，不能为其设置密码；但是，创建好用户后，可以编辑，为其添加密码。

使用新建用户上传代码

通过**http**协议上传

```
[root@node4 ~]# cd ~/myweb/
[root@node4 myweb]# git remote rename origin old-origin
# 出现以下错误可忽略
error: 不能重命名配置小节 'remote.origin' 到 'remote.old-origin'
[root@node4 myweb]# git remote add origin http://192.168.4.5/devops/myweb.git
[root@node4 myweb]# git push -u origin --all
Username for 'http://192.168.4.5': zzg
Password for 'http://zzg@192.168.4.5':
[root@node4 myweb]# git push -u origin --tags
Username for 'http://192.168.4.5': zzg
Password for 'http://zzg@192.168.4.5':
```

通过**ssh**上传

```
# 生成密钥对
[root@node4 myweb]# ssh-keygen -t rsa -C "zzg@tedu.cn" -b 4096

# 将公钥拷贝到gitlab服务器用户的ssh密码
[root@node4 myweb]# cat ~/.ssh/id_rsa.pub

# 修改本地代码
[root@node4 myweb]# cp /etc/shadow .
[root@node4 myweb]# git add .
[root@node4 myweb]# git commit -m "add shadow"

# 通过ssh推送
[root@node4 myweb]# git push git@192.168.4.5:devops/myweb.git

# 修改默认推送的方式是ssh
[root@node4 myweb]# git remote remove origin   # 删除
[root@node4 myweb]# git remote add origin git@192.168.4.5:devops/myweb.git

[root@node4 myweb]# cp /etc/motd .
[root@node4 myweb]# git add .
[root@node4 myweb]# git commit -m "add motd"
[root@node4 myweb]# git push
```