nsd1906_devops_day04

ansible

如果你连接远程主机是个普通用户，怎么执行管理任务？

```
# vim ansible.cfg
[defaults]
inventory = inventory
remote_user = zhangsan

[priviledge_escalation]    # 提权
become = yes               # 需要切换用户
become_method = sudo       # 切换的方式是sudo（另一种方式是su）
become_user = root         # 切换成管理员
become_ask_pass = no       # 不询问切换密码

# 被管理的服务器，需要配置sudo
# visudo
zhangsan    ALL=(ALL)       NOPASSWD: ALL
```

ansible-cmdb

- 将ansible收集的主机信息转换成web页面

```
# 收集远程主机的信息
(nsd1906) [root@room8pc16 myansible]# ansible all -m setup --tree /tmp/nsd1906out

# 安装ansible-cmdb
(nsd1906) [root@room8pc16 myansible]# pip install /var/ftp/pub/zzg_pypkgs/ansible-
cmdb_pkgs/*
# 或在线安装
(nsd1906) [root@room8pc16 myansible]# pip install ansible-cmdb

# 生成web页面
(nsd1906) [root@room8pc16 myansible]# ansible-cmdb /tmp/nsd1906out > /tmp/hosts.html
(nsd1906) [root@room8pc16 myansible]# firefox /tmp/hosts.html &
```

git

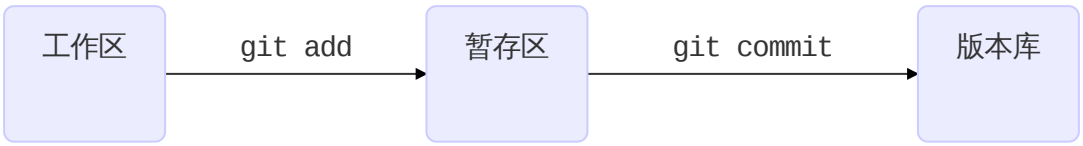
- 它是一个分布式的软件版控制系统

```
# 安装
[root@node4 ~]# yum install -y git
# 配置基本信息
[root@node4 ~]# git config --global user.name "Mr.Zhang"
```

```
[root@node4 ~]# git config --global user.email "zzg@tedu.cn"
[root@node4 ~]# git config --global core.editor vim
# 查看信息
[root@node4 ~]# git config --list
user.name=Mr.Zhang
user.email=zzg@tedu.cn
core.editor=vim
[root@node4 ~]# cat ~/.gitconfig
[user]
    name = Mr.Zhang
    email = zzg@tedu.cn
[core]
    editor = vim
```

git的重要工作区域

- 工作区：编写代码的工作目录
- 暂存区：.git/index，工作区和版本库之间的缓冲地带
- 版本库：工作区中的.git目录



git应用

```
# 创建版本库，方法一：编写项目之初创建
[root@node4 ~]# git init mytest
初始化空的 Git 版本库于 /root/mytest/.git/
[root@node4 ~]# ls -A mytest/
.git
# 创建版本库，方法二：在已存在的项目中创建版本库
[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

# 大多数的操作都可以通过git status得到提示
[root@node4 myweb]# git status
# 位于分支 master
#
```

```
# 初始提交
#
# 未跟踪的文件:
#   (使用 "git add <file>..." 以包含要提交的内容)
#
#   index.html
提交为空，但是存在尚未跟踪的文件（使用 "git add" 建立跟踪）

[root@node4 myweb]# git status -s
?? index.html   # ??表示当前文件的状态未知

# 提交数据到暂存区
[root@node4 myweb]# cp /etc/hosts .
[root@node4 myweb]# git add index.html
[root@node4 myweb]# git status
# 位于分支 master
#
# 初始提交
#
# 要提交的变更：
#   (使用 "git rm --cached <file>..." 撤出暂存区)
#
#   新文件：    index.html
#
# 未跟踪的文件:
#   (使用 "git add <file>..." 以包含要提交的内容)
#
#   hosts
[root@node4 myweb]# git status -s
A  index.html   # A表示新增
?? hosts
[root@node4 myweb]# git add .
[root@node4 myweb]# git status -s
A  hosts
A  index.html

# 从暂存区撤出hosts
[root@node4 myweb]# git rm --cached hosts
rm 'hosts'
[root@node4 myweb]# git status -s
A  index.html
?? hosts

# 创建.gitignore，忽略不需要加入到版本库的文件
[root@node4 myweb]# echo hosts >> .gitignore
[root@node4 myweb]# echo .gitignore >> .gitignore
[root@node4 myweb]# cat .gitignore
hosts
.gitignore
[root@node4 myweb]# git status -s
A  index.html

# 确认文件至版本库
```

```
[root@node4 myweb]# git commit   # 跳出vim写日志，如果为空白日志，则终止提交
[root@node4 myweb]# git commit -m "init data"
[root@node4 myweb]# git status
# 位于分支 master
无文件要提交，干净的工作区

# 删除工作区的文件，并恢复
[root@node4 myweb]# rm -f index.html
[root@node4 myweb]# git status
# 位于分支 master
# 尚未暂存以备提交的变更：
#   （使用 "git add/rm <file>..." 更新要提交的内容）
#   （使用 "git checkout -- <file>..." 丢弃工作区的改动）
#
#   删除：      index.html
#
修改尚未加入提交（使用 "git add" 和/或 "git commit -a"）
[root@node4 myweb]# git checkout -- index.html
[root@node4 myweb]# ls
hosts  index.html

# 删除文件
[root@node4 myweb]# git ls-files   # 查看版本库中的文件
index.html
[root@node4 myweb]# git rm index.html  # 删除文件
rm 'index.html'
[root@node4 myweb]# git status
# 位于分支 master
# 要提交的变更：
#   （使用 "git reset HEAD <file>..." 撤出暂存区）
#
#   删除：      index.html
#
[root@node4 myweb]# git commit -m 'rm index.html'
[root@node4 myweb]# git status
# 位于分支 master
无文件要提交，干净的工作区

# 切换到某一版本的状态
[root@node4 myweb]# git log   # 查看日志
[root@node4 myweb]# git checkout efc9ec25d8a85cdb62d9863819c34a6be4cb80ad
[root@node4 myweb]# ls
hosts  index.html
[root@node4 myweb]# git checkout master   # 回到最新状态
之前的 HEAD 位置是 efc9ec2... init data
切换到分支 'master'
[root@node4 myweb]# ls
hosts
```

git tag标记

- tag可以为某一次提交设置标记
- 如将某一次提交设置为软件的版本号

```
# 查看tag
[root@node4 myweb]# git tag
# 为某一次提交设置tag标记
[root@node4 myweb]# git tag 1.0
[root@node4 myweb]# git tag
1.0
```

分支管理

```
# 创建分支
[root@node4 myweb]# git branch b1
# 查看分支
[root@node4 myweb]# git branch
  b1
* master     # 当前处于master分支

# 在master分支上编写代码并提交
[root@node4 myweb]# cp /etc/passwd .
[root@node4 myweb]# git add .
[root@node4 myweb]# git commit -m 'add passwd'

# 切换到b1分支
[root@node4 myweb]# ls
hosts  passwd
[root@node4 myweb]# git checkout b1
切换到分支 'b1'
[root@node4 myweb]# ls
hosts

# 在b1分支上提交代码
[root@node4 myweb]# echo 'ni hao' > hi.txt
[root@node4 myweb]# ls
hi.txt  hosts
[root@node4 myweb]# git add .
[root@node4 myweb]# git commit -m 'create hi.txt'

# 切换回master
[root@node4 myweb]# git checkout master
切换到分支 'master'
[root@node4 myweb]# ls
hosts  passwd

# 合并分支，将b1汇入主干
[root@node4 myweb]# git branch
  b1
* master
[root@node4 myweb]# git merge b1  # 在跳出的vim中写入日志
[root@node4 myweb]# ls
hi.txt  hosts  passwd

# 分支不再需要时，可以将其删除
[root@node4 myweb]# git help branch   # 查看branch帮助
```

```
[root@node4 myweb]# git branch -d b1
已删除分支 b1（曾为 6fe2b7e）。
[root@node4 myweb]# git branch
* master
```

gitlab服务器

```
# 安装docker并启动
[root@node5 docker_pkgs]# systemctl start docker
[root@node5 docker_pkgs]# systemctl enable docker
# 将镜像导入 (/linux-soft/05/gitlab_zh.tar)
[root@node5 images]# docker load < gitlab_zh.tar

# 将docker宿主机ssh端口号改为2022
[root@node5 ~]# vim /etc/ssh/sshd_config
Port 2022
[root@node5 ~]# systemctl restart sshd
# 再次连接时，需要指定端口号
[root@room8pc16 myansible]# ssh -p2022 node5

# 启动容器
[root@node5 ~]# docker run -d -h gitlab --name gitlab -p 443:443 -p 80:80 -p 22:22 --
restart always -v /srv/gitlab/config:/etc/gitlab -v /srv/gitlab/logs:/var/log/gitlab -v
/srv/gitlab/data:/var/opt/gitlab gitlab_zh:latest
# 查看容器状态，当容器的状态为healthy时，容器才是可用的
[root@node5 ~]# docker ps
```

gitlab应用

1. 登陆服务器http://x.x.x.x，设置root密码
2. gitlab中重要的概念
    1. 群组group：对应一个开发团队
    2. 用户、成员member：对应用户帐户，可以将用户加入到组或项目
    3. 项目：对应软件项目

创建名为devops的组，类型为公开

创建一个用户账号，新建用户时，不能设置密码。创建完成后，可以再编辑，以设置密码。

创建项目：为devops组创建项目，项目名为myweb，类型为公开。

为项目授权：点击项目左边栏的设置。将前一步创建的普通用户加入项目，成为主程序员。

切换为普通用户，上传代码

- 新用户首次登陆时，需要改密码
- 点击项目，因为项目是空的，它将提示你如何操作
    - 创建新版本库：适用于你本地没有任何文件

- 已存在的文件夹：适用于你已经创建了基目目录，但是没有执行过git init实现初始化
- 已存在的 Git 版本库：适用于你已经使用git管理的项目目录

```
[root@node4 ~]# cd myweb/
# 创建仓库，该仓库与一个gitlab项目的url关联
[root@node4 myweb]# git remote add origin http://192.168.4.5/devops/myweb.git
# 推送本地代码到gitlab
[root@node4 myweb]# git push -u origin --all
Username for 'http://192.168.4.5': zzg
Password for 'http://zzg@192.168.4.5':
# 推送tag到gitlab
[root@node4 myweb]# git push -u origin --tags
Username for 'http://192.168.4.5': zzg
Password for 'http://zzg@192.168.4.5':

# 如果在add时，有输入错误，可以把它删掉
[root@node4 myweb]# git remote remove origin
```

通过ssh免密上传代码

1. 点击页面右上角用户->设置
2. 把你的公钥复制到“SSH密钥”（左边栏->ssh密钥）

```
[root@node4 myweb]# ssh-keygen -t rsa -C "zzg@tedu.cn" -b 4096
[root@node4 myweb]# cat ~/.ssh/id_rsa.pub
```

3. 通过ssh上传代码

```
# 将http的方式删除
[root@node4 myweb]# git remote remove origin
# 更新上传方式为ssh
[root@node4 myweb]# git remote add origin git@192.168.4.5:devops/myweb.git
# 上传代码
[root@node4 myweb]# echo 'how are you?' > aaa.html
[root@node4 myweb]# git add .
[root@node4 myweb]# git commit -m 'create aaa.html'
[root@node4 myweb]# git push

# 通过http下载
[root@node4 myweb]# cd /tmp/
[root@node4 tmp]# git clone http://192.168.4.5/devops/myweb.git
[root@node4 tmp]# ls -A myweb/
aaa.html  .git  hi.txt  passwd

# 同步代码
[root@node4 tmp]# cd myweb/
[root@node4 myweb]# git pull
Already up-to-date.
```