Git版本管理

何为版本控制/管理?

> 版本控制是一种记录一个或若干文件内容变化，以便将来查阅特定版本修订情况的系统，Git是目前最先进的分布式版本控制系统。

- Git 可以记录每次文件/脚本修改的历史，方便回滚和查找问题。
- 支持团队多人协作开发，避免冲突
- 结合 Jenkins 持续集成工具，实现 **CI/CD 流水线**

Git & SVN对比

- SVN版本集中管理，所有的代码都在一台服务器上
- Git去中心化，每个服务器/本地都有一个完整的代码库

GitHub/GitLab/Gitee对比及区别

- GitHub - 全球最大的面向开源及私有软件项目的托管平台，免费注册并且可以免费托管开源代码。 https://github.com/
- Gitee（码云） - 国内版的GitHub https://gitee.com/
- GitLab - GitLab可以私有化部署在自己的服务器上，代码数据都是掌握在自己手中，适合公司内部团队开发。

> 在使用第三方的平台（GitHub/Gitee）的时候，不要将你们公司项目相关的代码/数据上传平台上面！！！

注册Gitee和Github账号的时候需要，要和你的邮箱地址关联地址

Git安装

Git客户端：https://git-scm.com/downloads

在Git的命令行模式下，我们可以使用非常多的Linux命令操作：ls pwd cd cat ...

for Windows

- 直接下载exe程序包进行安装即可

for Linux：

```
yum -y install git
```

> Cannot find a valid baseurl for repo: base/7/x86_64

安装错误解决方案：

1、检查网络配置

```
# 网络配置文件
vim /etc/sysconfig/network-scripts/ifcfg-ens33
# 重启网络
service network restart
```

2、更新yum源

```
# 备份原来的 YUM 配置
mv /etc/yum.repos.d/CentOS-Base.repo /etc/yum.repos.d/CentOS-Base.repo.bak
# 下载新的阿里云源
curl -o /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-7.repo
# 清理缓存并生成新缓存
yum clean all
yum makecache
# 再次尝试安装
yum install git -y
```

最后使用git --version命令检测是否OK

Git配置步骤

Step1：注册Gitee or GitHub账号

如果是GitLab则找开发/运维申请Gitlab账号

Step2：配置SSH密钥

打开 `Git命令行模式` ，执行以下命令生成密钥：

```
ssh-keygen -t rsa -C "邮箱地址"
```

进入到用户目录下的.ssh目录，复制id_rsa.pub文件内容添加到Gitee或者Github或者GitLab上面

Step3：测试链接是否成功

```
# 测试gitee
ssh -T git@gitee.com
# 测试github
ssh -T git@github.com
```

Step4：配置邮箱和用户名

```
git config --global user.email "你的邮箱地址"
git config --global user.name "你的用户名"
```

配置PyCharm工具支持Gitee的操作

- 下载对应Gitee（需要自己手动下载）/GitHub插件（不需要，pycharm默认有自带了这个插件）
  - 装好了插件之后记得重启你的pycharm，这样才会生效
- 选择VCS->Share Project on GitHub/ Share Project On Gitee

Git常用命令

- 克隆
  git clone 远程仓库地址
- 查看未被追踪的文件：
  git status

- 追踪文件：
  git add

  git add * 追踪当前所有未被追踪的文件
- 提交文件：
  git commit -m “注释”

  需要注意，我们在提交的时候会按照规范写上备注/注释的信息
- 推送到远程：
  git push

  通过git push才会把本地的代码同步到远程的仓库里面
- 删除：
  git rm 文件 ~ 不需要再次使用git add追踪文件
- 查看本地所有分支：
  git branch
- 查看所有的本地及远程分支：
  git branch -a
- 创建分支：
  git checkout -b 分支名
- 将分支推送到远程(第一次)：
  git push --set-upstream origin 分支名
- 切换分支：
  git checkout master

> 分支：主干分支（master/main）、开发分支（develop）、特性分支（feature）、热修复分支（hotfix）、发布分支（release）
>
> 合并feature分支开发的代码 - 正式的项目合并代码流程
>
> - 提交PR（pull request）-GitHub or Gitee平台上面的叫法 /MR（merge request）-GitLab上面的叫法：提交代码合并的请求
> - 将feature分支代码合并到开发（develop）分支里面

- 删除本地分支：
  git branch -d 本地分支名
- 删除远程分支：
  git push origin --delete 分支名
- 拉取远程分支

  git fetch origin 分支名

  git checkout -b 分支名 origin/分支名
- 查看所有操作的版本号
  git reflog
- 版本回退操作：
  git reset --hard 版本号 回退到指定版本

  git reset --hard HEAD^ 回退到上一个版本

  git reset --hard HEAD^^ 回退到上上一个版本