GIT 版本控制器

安装

Linux & CentOS

```
yum -y install git
```

Windows

打开 GIT 官网 下载安装包, 双击安装

概念理解

什么是版本控制器?

> 版本控制最主要的功能就是追踪文件的变更。它将什么时候、什么人更改了文件的什么内容等信息忠实地了已录下来。每一次文件的改变，文件的版本号都将增加

工作区和暂存区?

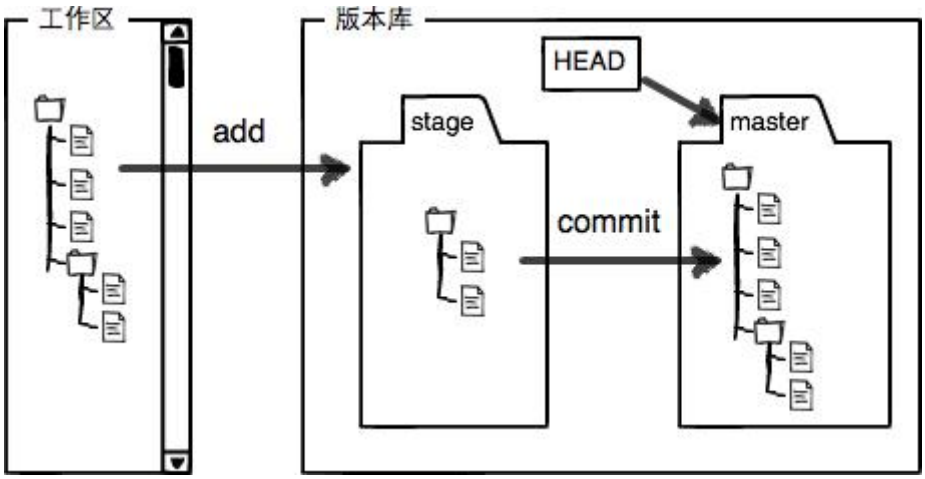
工作区

> 就是你在电脑里能看到的目录

版本库

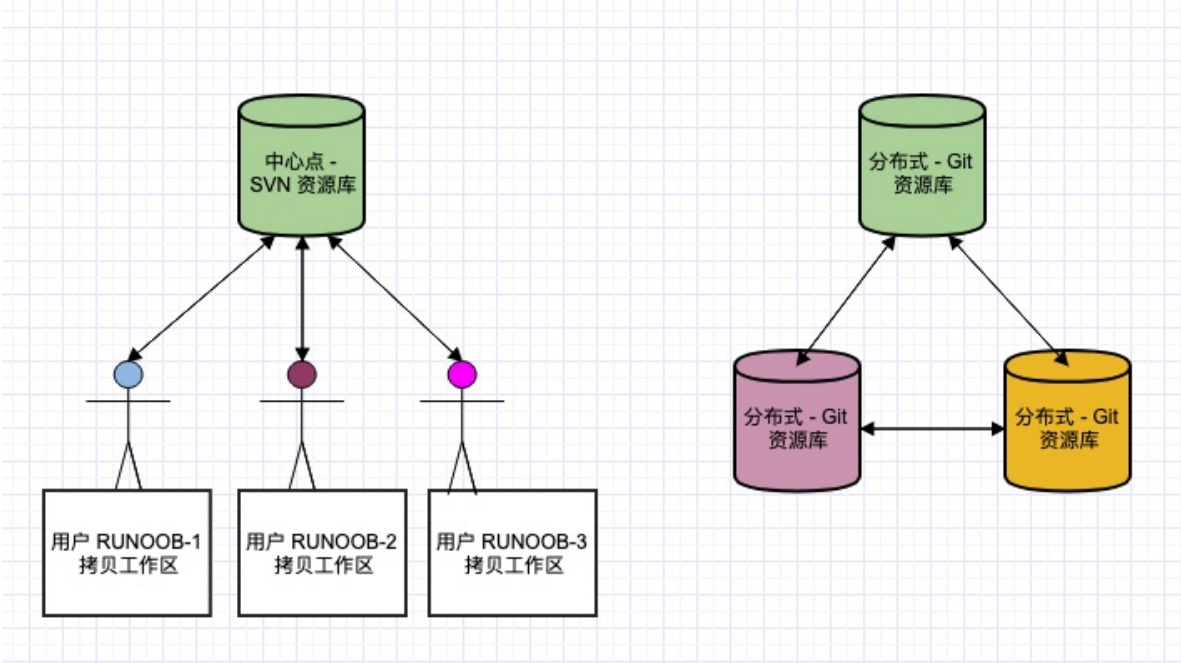
> 工作区有一个隐藏目录`.git`，这个不算工作区，而是Git的版本库。
>
> Git的版本库里存了很多东西，其中最重要的就是称为stage（或者叫index）的暂存区，还有Git为我们自动创建的第一个分支`master`，以及指向`master`的一个指针叫`HEAD`。
>
> 分支 和 HEAD后面说

1. 将修改的文件提交到暂存区
2. 把文件的修改提交到版本库



什么分布式?

> 也就是说分布式系统背后是由一系列的计算机组成的，但用户感知不到背后的逻辑，就像访问单个计算机一样。



基本操作

初始化一个仓库

> 运行这句命令, 会将当前目录变成 git 版本库

```
$ git init
```

暂存文件

```
$ git add FILENAME
$ git add -A # 暂存所有文件
```

提交文件

```
$ git commit -m "本次版本的备注(必须填写)"
```

查看文件是否被修改

```
$ git status
```

查看文件的内容的变化

```
$ git diff
```

复制一个远程版本库到本地

```
$ git clone URL地址
```

设置用户信息

```
$ git config --global user.name "wzblog"
$ git config --global user.email "443@qq.com"
```

提交本地的版本库到远程

```
$ git push origin master(分支的名字)
```

将远程代码更新到本地

```
$ git pull
```

查看版本日志

```
$ git log
```

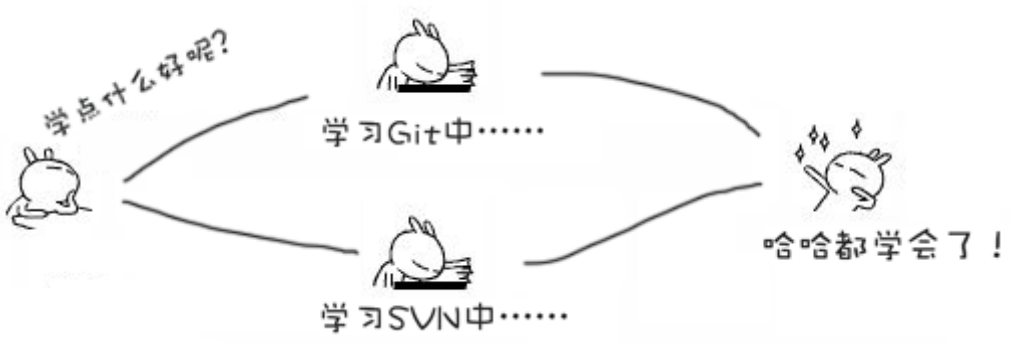
丢弃文件的修改

> 丢弃工作区的文件修改

```
$ git checkout -- FILENAME
```

分支管理

> 分支就是科幻电影里面的平行宇宙，当你正在电脑前努力学习Git的时候，另一个你正在另一个平行宇宙里努力学习SVN。
>
> 如果两个平行宇宙互不干扰，那对现在的你也没啥影响。不过，在某个时间点，两个平行宇宙合并了，结果，你既学会了Git又学会了SVN！



> 分支在实际中有什么用呢？假设你准备开发一个新功能，但是需要两周才能完成，第一周你写了50%的代码，如果立刻提交，由于代码还没写完，不完整的代码库会导致别人不能干活了。如果等代码全部写完再一次提交，又存在丢失每天进度的巨大风险。
>
> 现在有了分支，就不用怕了。你创建了一个属于你自己的分支，别人看不到，还继续在原来的分支上正常工作，而你在自己的分支上干活，想提交就提交，直到开发完毕后，再一次性合并到原来的分支上，这样，既安全，又不影响别人工作。

查看所有分支

```
$ git branch
* master 主分支
```

创建新分支

```
$ git branch dev(分支名) # 创建
$ git checkout -b dev(分支名) # 创建并切换
```

切换分支

```
$ git checkout dev(分支名)
```

删除分支

```
$ git branch -d dev(分支名)
```

合并分支

> 将 `dev` 分支合并到当前分支

```
$ git merge dev(分支名)
```

版本切换(时光机)

> `HEAD` 表示当前版本
>
> `HEAD^` 表示上一个版本, 没多加一个 `^` 则表示前一个版本. `HEAD^^` 则表示回退两个版本
>
> 如果想回退 50 个版本, 那要怎么写呢? 写 50 个 `^`??? 不不不, 我能可以使用 `HEAD~50` 来回退 50 个版本, 当然你也可以将 50 改为其他的数字, 改为几便会回退多少个版本.
>
> 我们还能切换到指定版本, 只需要把参数写成对应的 `commit id` (版本id)

```
$ git reset --hard [commit id | HEAD]
git reflog 查询历史日志
```

1. 切换到你想要修改的版本
2. 新建一个分支用来修改文件, 提交一个版本
3. 修改完成之后先切换回原来的分支, 修改到最新版本
4. 把我们新建的分支合并进来

log

使用方式

如何在不影响他人的情况下开发项目?

使用流程

常用分支

master 主分支

> 这个分支只能从其他分支合并，不能在这个分支直接修改. 这个分支用来保存可以正常运行的项目

dev 主开发分支

> 这个主要用于合并其他分支, 我们需要把不同成员的代码合并, 合并完要测试

成员分支, 分支名称用自己名字来建

> 这个分支用于编写自己的功能代码, 每个人需要一个. 自己写的代码提交在自己的分支上则不会影响到其他人开发项目. 当功能没问题时在合并到dev分支, 同步给其他的小伙伴.

1. master 用来发布完整的项目
2. dev 合并不同的模块用的分支
3. 成员分支

合并代码冲突问题

> 在代码合并时会冲突

参考手册

https://git-scm.com/docs

菜鸟教程

怎么同步代码???

(云端仓库) 不是中心仓库

代码同步: 所有人都要访问的到的地方

github 来做云端仓库 => gayhub 全球最大同行交友网站