Intro to Git

> 参考资料:Git-book

前言

Git是什么

简而言之：Git 是目前使用最广泛的版本控制系统之一。

~~但是，这谁听得懂？~~

版本是什么

版本：文件或代码的版本，例如某个项目的初步版本、最终版本。

一般的项目开发流程会涉及到项目不断的迭代（迭代是指**对文件的更新**），每一次更新文件后即产生一个新的**版本**。

版本控制是什么

对项目开发过程中产生的不同版本进行**跟踪与记录**，并更够查找、回退到历史版本。

例如你对某个文件进行更新后发现自己写了一个bug,你想将文件**回退**到之前没有bug的版本，此时就需要版本控制。

为什么要用Git

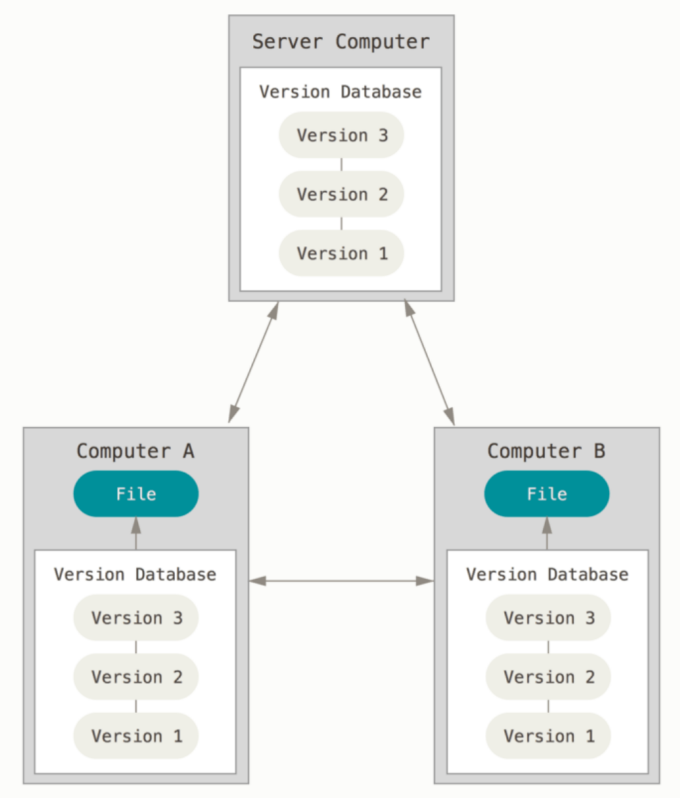
- Git是使用广泛的**分布式**版本控制系统

分布式是什么意思

分布式只多台电脑同时保存有项目的代码，换句话说，项目的代码**分布于**多台电脑之中。

为什么要设计分布式的版本控制系统

项目的代码分布于多台电脑之中，这些电脑的主人可以在自己的电脑上对代码做出不同的更新，随后可以将这些不同的更新合并到同一台**主电脑**上，每个工作人员可以通过主电脑获取其他人更新后的代码。分布式的目的是**分工协作**。



安装

窗户系统

从网上下载Git软件包，安装到电脑上，**并设置环境变量中的PATH变量**（设置路径）。

环境变量是什么

> 当场讲

~~高贵的~~Linux系统

用相应系统的安装包管理器进行安装。

- ~~高贵的~~Archlinux：`pacman`
- Ubuntu：`apt`

用Linux系统的安装包管理器进行安装一般不需要设置环境变量。（为什么？）

配置

Git 根据配置文件的应用范围，将配置文件分为不同的等级，其中较常用的有两个级别：

1. 适用于当前用户的全局配置文件，该用户操作本系统上的所有仓库时都会查询该配置文件。
2. 适用于当前文件夹的配置文件。

使用`git config`命令进行配置。

设置用户信息

安装 Git 后，第一件事情就是设置你的用户名和邮箱。这些信息在每次提交时都会用到。

```
$ git config --global user.name "<your name>"
$ git config --global user.email "<your email>"
```

这里的`--global`表示修改的是全局配置，即该设置对当前用户下的所有仓库均有效。如果不添加`--global`选项，则会默认修改当前仓库下的配置文件。

显示配置

可以通过`git config -l`列出当前已经设置的所有配置参数。使用 `git config --global -l`可以列出所有全局配置。

添加ssh公钥和密钥

将文件上传到github需要和远程仓库建立连接，一般使用ssh协议进行远程连接。

什么是协议

> 协议：计算机网络中，两台计算机之间进行通信时必须遵守的规则。

ssh是计算机网络协议的一种。

什么是公钥和密钥

他们是逻辑上的钥匙，用于识别你的身份。

将本地文件上传到github上需要让github知道你是谁。github会判断**公钥**和你的**密钥**是否匹配成功，如果匹配成功则允许你上传文件。

生成公钥和密钥

```
ssh-keygen -t rsa
```

将公钥上传到github

点击头像，选择`Settings`，选择`SSH and GPG keys`，点击`New SSH key`，将公钥粘贴到`Key`中，点击`Add SSH key`。

基本概念

工作区(working directory)

工作区是你在本地存放项目代码的📁**文件夹**（directory：目录、文件夹）。你将在这里修改或者更新你的代码。

使用`git init`在当前目录下创建并初始化一个git工作区。`git init`指令将会在当前目录下创建一个`.git`文件夹，其中记录了有关版本控制的信息。

暂存区(stage area)

暂时存放代码的地方。注意，暂存区更多是一个**逻辑上的概念**，在Git中仅作为一个过度的区域。例如当你更新了一部分的代码，但是又不想把代码作为一个新的版本进行提交的时候，可以将你修改的这部分代码先放入暂存区。

当你修改了`file.txt`文件中的某些代码后，可以使用`git add file.txt`将修改后的`file.txt`放入暂存区。

本地仓库(local repository)

当你确认自己已经完成了所需要进行的代码更新后，你可以将文件从暂存区**提交**到本地仓库。本地仓库是位于你自己的电脑上的仓库，**任何提交到本地仓库的代码将会被视为一个新的版本**。

使用`git commit`将暂存区内的文件提交大本地仓库。

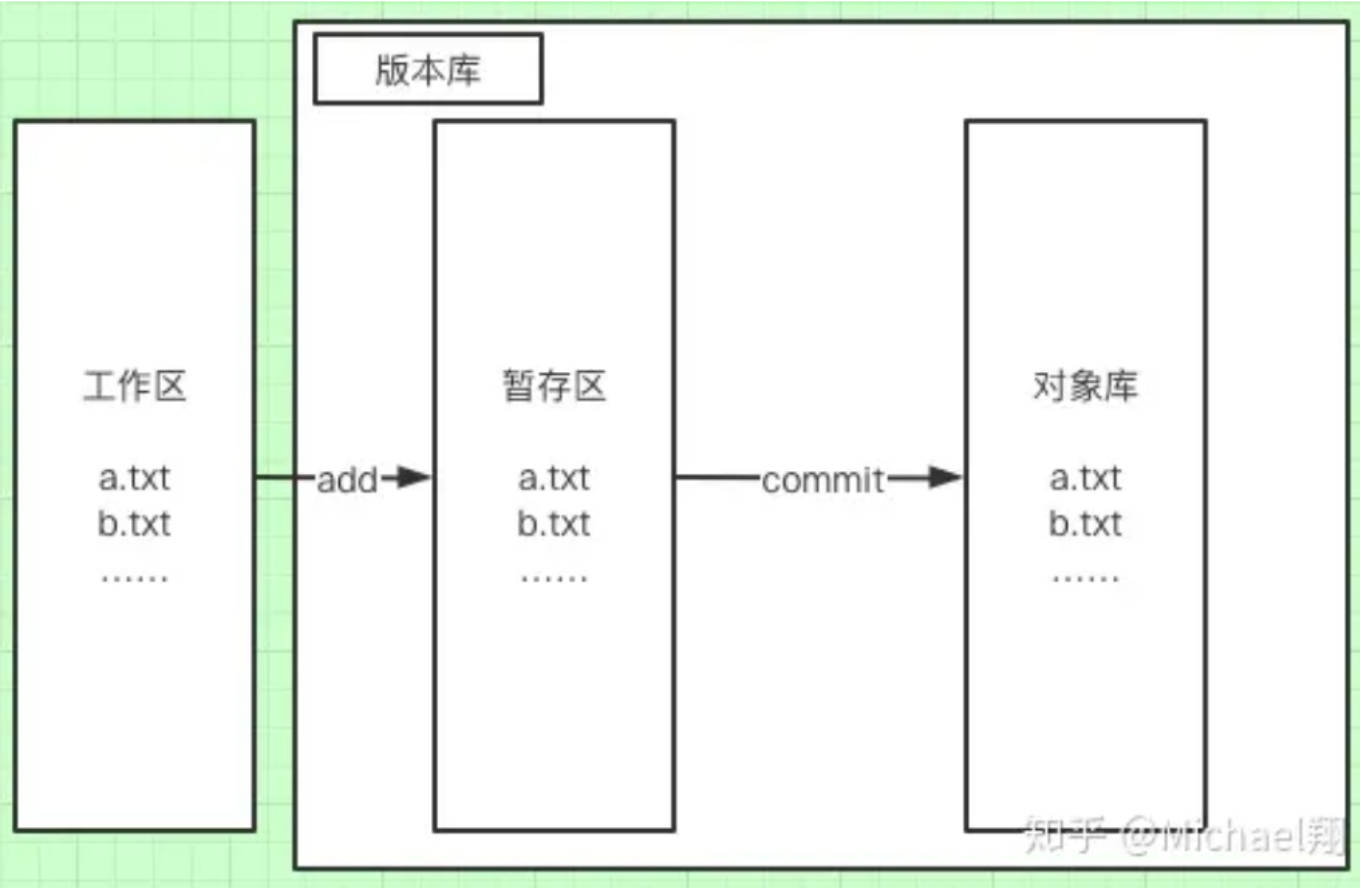
远程仓库(remote repository)

这是一个位于**主服务器**上的仓库，当你把自己要更新的代码更新好并提交到本地仓库时，接着就可以推送到主服务器上。其他同时可以根据通过拉取主服务器上的代码获取你更新的部分代码。

使用`git push`将本地仓库的文件**同步**到远程仓库。

使用`git pull`将远程仓库的文件**同步**到本地。

使用`git clone`直接将整个远程仓库下载到本地。



分支

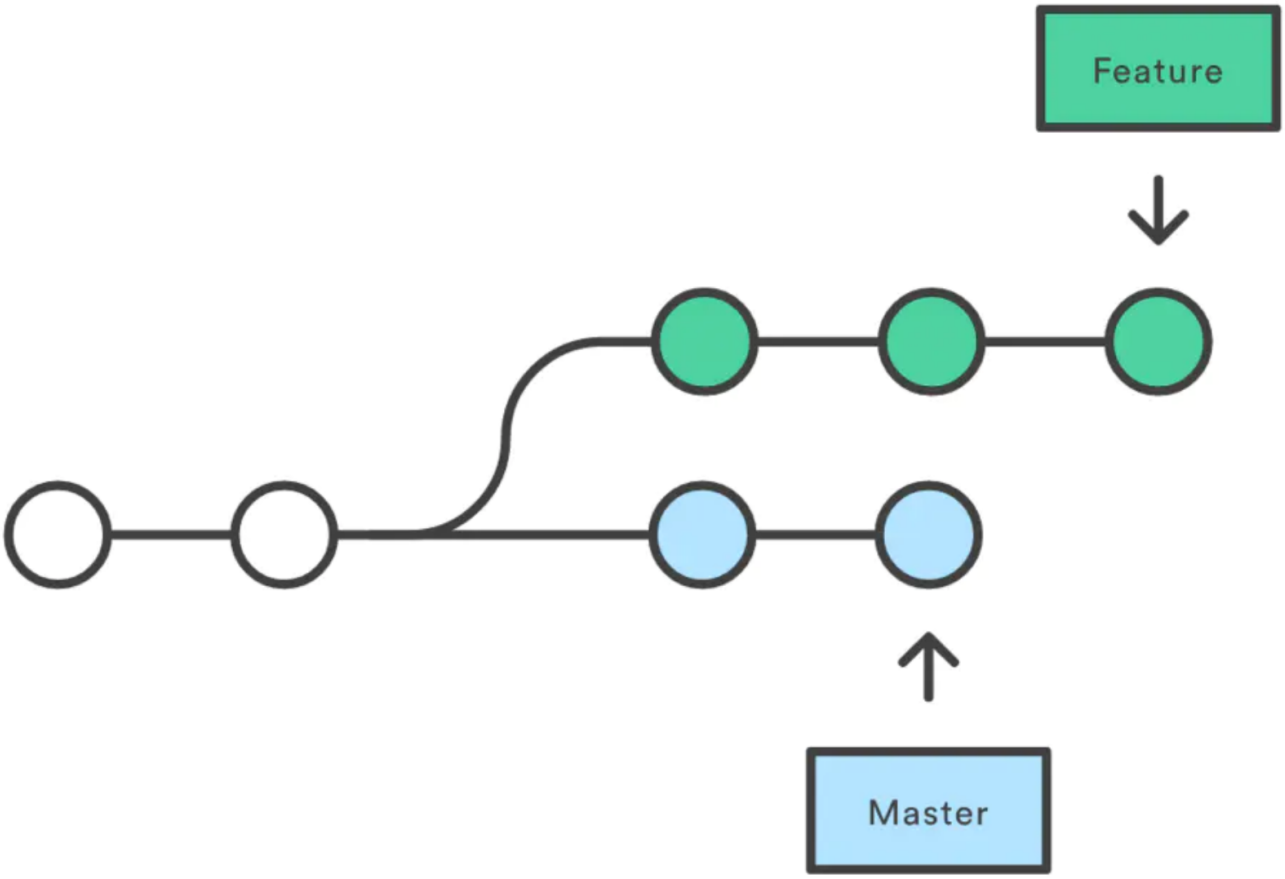
Commit是什么意思

> commit：提交（文件等）

commit会将修改后的代码提交到本地仓库，此时本地仓库会记录**当前版本与前一个版本的区别**，用于版本**回退**。

```
# 新建分支
git branch <name>
# 列出分支
git branch
# 切换分支
git checkout <name>
# 新建并切换分支
git checkout -b <name>
```

假设当前你在feature分支上进行新功能的开发， 与此同时master分支上也合入了其他人提交的代码：



如果采用merge的方式将master上新的代码合并到本地的feature分支上：

```
git merge master feature
```

可视化

```
# 查看当前本地仓库的状态
git status

# 查看工作区文件与暂存区文件的区别
git diff

# 查看当前工作区与某个提交的区别
git diff <commit_id>

# 查看提交(commit)的历史记录
git log
```

Tips

> 这是一些使用命令行或者Git的小技巧

查看帮助文档

你以为我会让你用`man`查看文档？~~什么年代了还在看传统文档。~~

- **Too Long, Don't Read**：`tldr`命令

AI辅助工具

- Github Copilot
- Github Copilot CLI

modern unix

> 链接

- Git 相关：`delta`
    - A viewer for `git` and `diff` output
- 现代化图形界面Git：`lazygit`
- 妙用`alias`：`gitu='git add . && git commit -m "regular update" && git push'`