GIT基础

何吉轩 软03

Github

什么是Git

By Wiki

Git 是一个开源的**分布式版本控制系统**，用于敏捷高效地处理任何或小或大的项目。

Git 是 Linus Torvalds 为了帮助管理 Linux 内核开发而开发的一个开放源码的版本控制软件。

在2002年以前，Linux社区中世界各地的志愿者把源代码文件通过diff的方式发给Linus，然后由Linus本人通过手工方式合并代码。

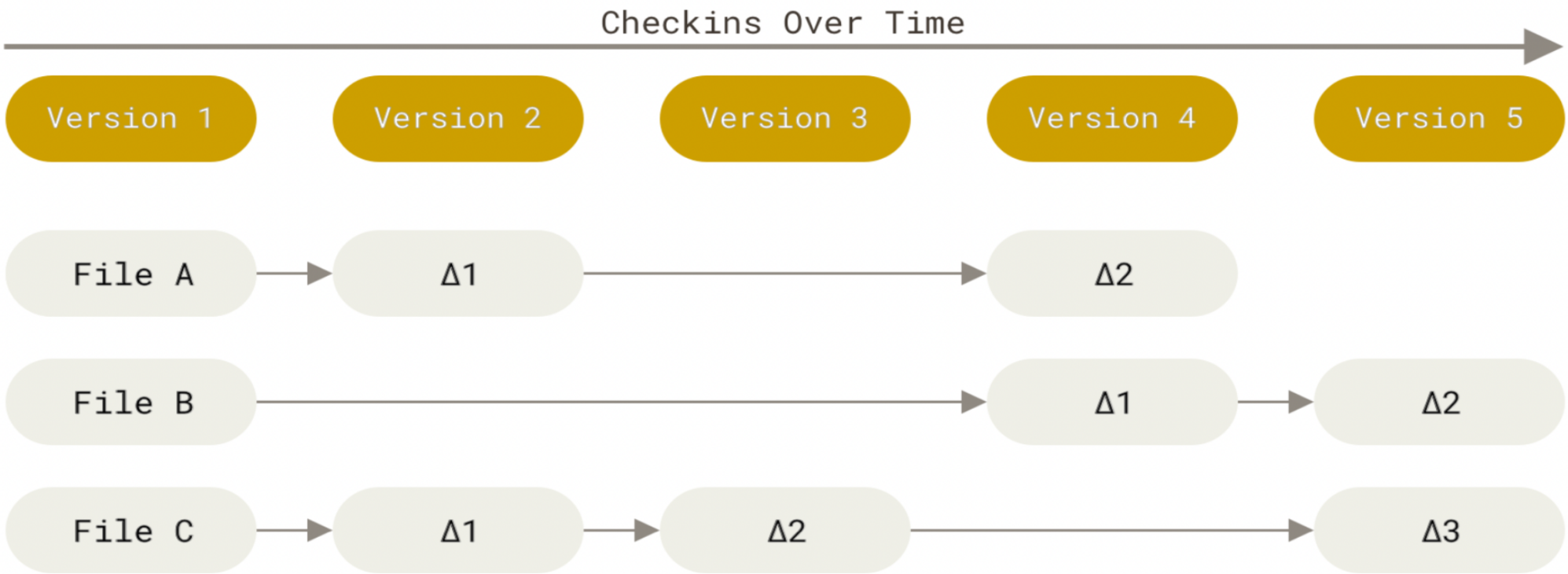
疑问

1. 何谓分布式？好理解
2. git如何实现版本控制？

一个集中版本控制的例子

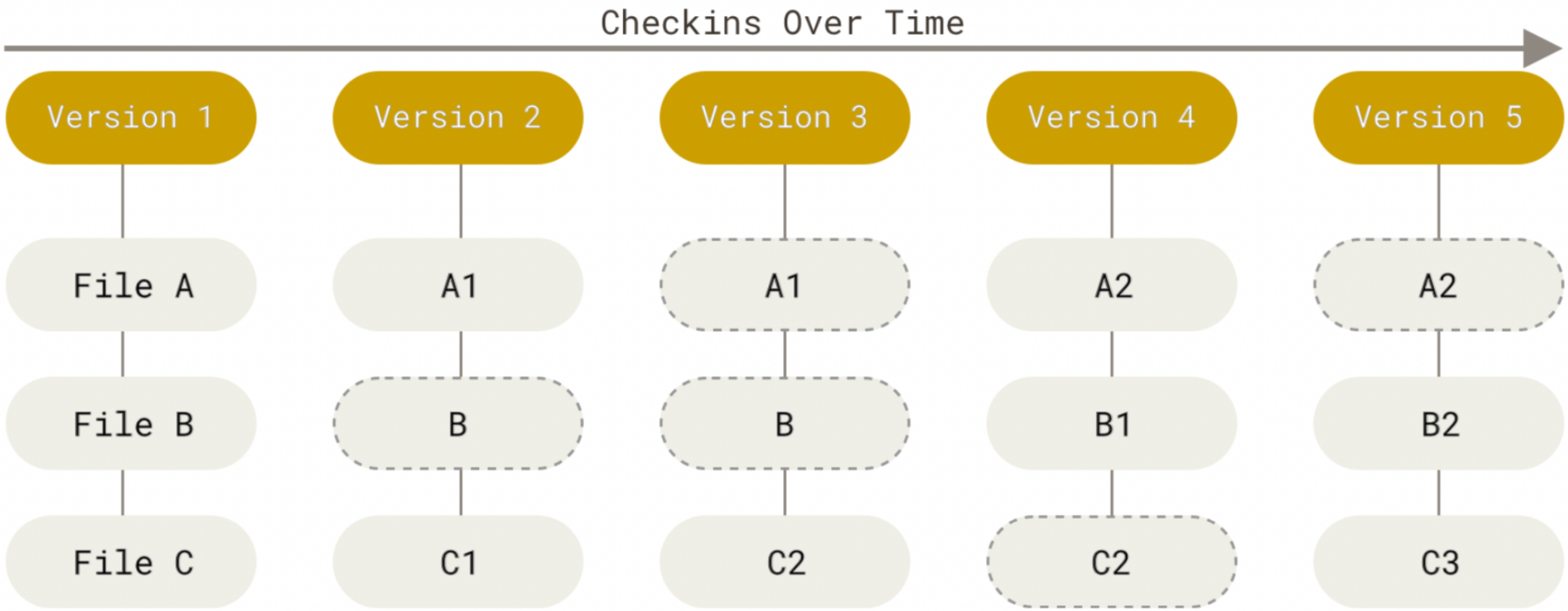
Subversion

把版本看成是一组文件随时间累积的结果，基于差异来控制版本。



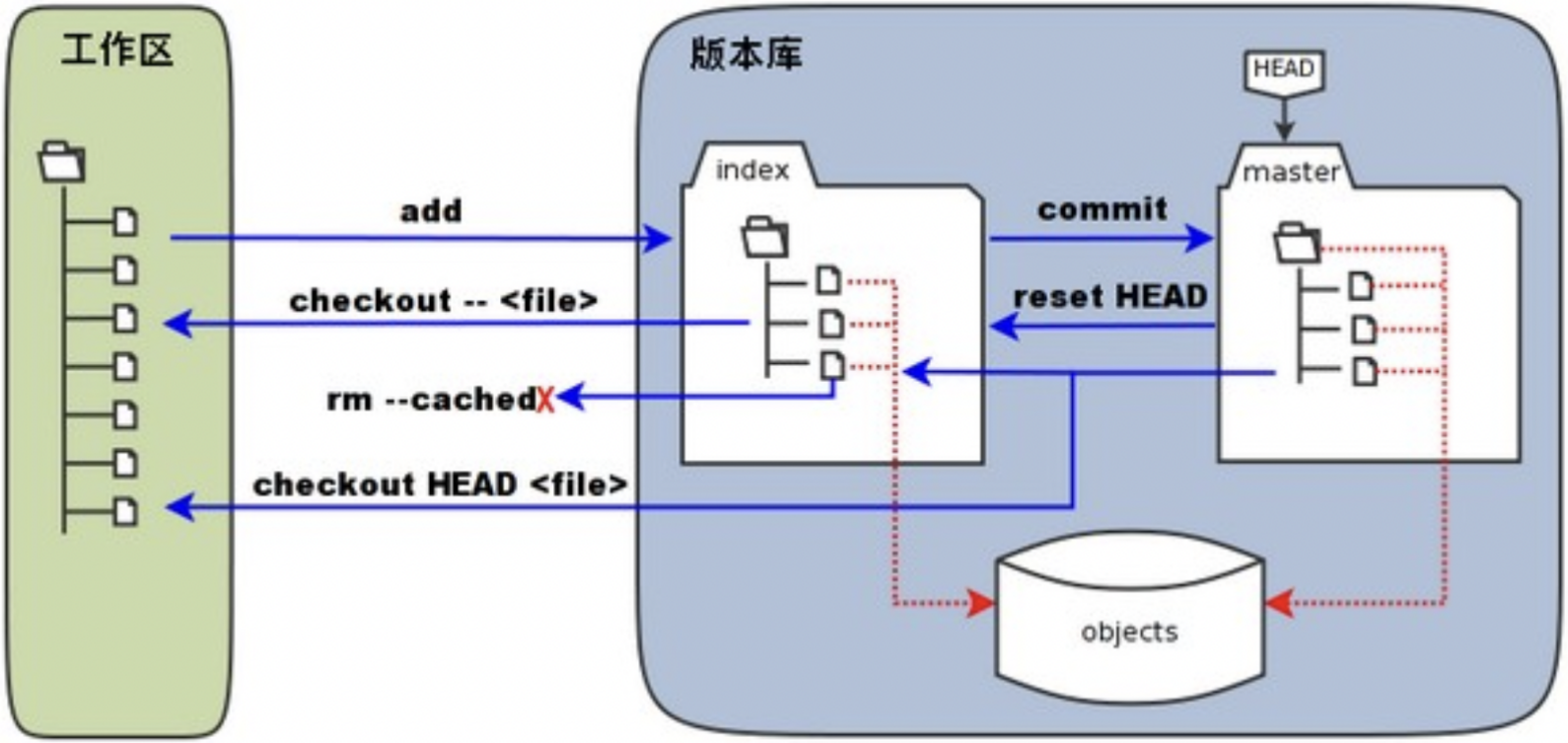
Git的版本控制

把每次提交都保存为系统文件的快照。如果没有修改，则直接引用前一版本。



如果把某些很大的文件（例如视频等二进制文件）也加入到这个版本控制系统的管理中，会发生什么？

Git仓库的结构



图中左侧为工作区，右侧为版本库。在版本库中标记为 "index" 的区域是暂存区(stage/index)，标记为 "master" 的是 master 分支所代表的目录树。

每一棵目录树，就是上一张图里的一个节点。

安装Git

Linux(以Ubuntu为例)

试着输入 `git`，检查系统是否已经安装Git

```
$ git
The program 'git' is currently not installed. You can install it by typing:
sudo apt-get install git
```

执行 `sudo apt-get install git` 以安装Git。或前往Git官网安装。

macOS

使用homebrew安装（https://brew.sh/）

```
$ brew install git
```

Windows

前往Git官网安装 https://git-scm.com/downloads

注意:在windows下，安装完成后最好在git bash中执行之后的命令;如果想在cmd/powershell中使用，则需要在 安装时选中“把Git可执行文件添加到PATH”这一复选框。

安装好之后，还需要进行一步设置，在命令行中输入

```
$gitconfig--globaluser.name"YourName"
$gitconfig--globaluser.email"email@example.com"
```

注意 `git config` 命令的 `--global` 参数，这表示这台机器上所有的Git仓库都会使用这个配置，当然也可以对各个仓库指定不同的用户名和Email地址。

Let’s try it!

初学者可以在VSCode中操作。

创建版本库

创建一个版本库非常简单，例如在某个目录中，通过以下命令把这个目录变成通过Git管理的仓库。

```
$ git init
```

查看一下当前目录下所有文件

```
$ ls -a
.git
```

多出一个隐藏的 `.git` 的目录，这个目录是Git来跟踪管理版本库的，请不要**随意手动修改**这个目录，不然会破坏Git仓库。

问：如何删除一个版本库?

添加文件到版本库

现在我们在 .git 文件所在目录下编写一个 readme.txt 文件，内容如下：

```
Git is a version control system.
Git is free software.
```

1. 用命令 git add 添加文件到暂存区

```
$ git add readme.txt
```

也可以配合 .gitignore 文件使用下面的命令，把所有文件都加入到暂存区

```
$ git add *
```

.gitignore文件的作用?

告诉Git，请忽略掉我列出的这些文件（文件夹、文件类型），不要把他们加入仓库。

通常来说，一个 .gitignore 的编写可以是下面这样子的

```
/node_modules/  # 忽略整个文件夹 注意，在Node.js项目内这几乎是必须的
*.zip           # 忽略所有.zip文件，采用通配符
/src/nonsense.c # 忽略某个具体文件
!node_modules/  # 我偏要上传node_modules
!*.zip          # 不忽略所有.zip文件
!/src/main.cpp  # 不忽略该文件
```

更多模板https://github.com/toptal/gitignore

2. 用命令 `git commit`，把文件提交到仓库

```
$ git commit -m "wrote a readme file"
[master (root-commit) eaadf4e] wrote a readme file
 1 file changed, 2 insertions(+)
 create mode 100644 readme.txt
```

-m 后面输入的是本次提交的说明，可以输入任意内容，当然最好是有意义的，这样你就能从历史记录里方便地找到改动记录。

可以用一个简单的词语描述commit类型，常用词如下：

```
- fix: 修复bug
- feat: feature，实现新功能
- refactor: 重构代码(把不优美的实现改的优美、可维护等)
- ci: 对于CI(持续集成)系统配置的修改。如加入自动构建代码的命令、自动运行代码测试等
- docs: 对于文档的增删
- build: 对构建脚本(如Makefile)等的修改
- test: 增加测试用例，修改测试程序等
- chore: 其他
```

详细规范可以查阅资料。

如果没有输入-m参数?

Git此时会打开命令行EDITOR环境变量指定的编辑器，最常见的是vim，然后等待你输入提交信息。

最最简单的vim使用方法

- 进入vim时为命令模式，此时可以通过 `h,j,k,l` 实现光标的左、下、上、右移动
- 移动到特定位置后，按 `i` 进入输入模式。此时可以像普通的文本编辑器一样使用
- 输入完成后，先按 `esc` 回到命令模式，再按 `:` 进入编辑模式
- 输入 `wq`，回车，即可退出

要随时掌握工作区的状态，可以使用以下命令

```
$ git status
```

如果 `git status` 提示有文件被修改过，可以使用 `git diff` 可以查看修改内容。

*添加文件，我们究竟做了什么？（回到**那一张图**）*

版本回退

查看版本的提交日志，会自动打开vim，按照提交时间由近到远显示

```
$ git log
```

在Git中，用 HEAD 表示当前版本（就是指针），上一个版本就是 HEAD^（6上面的符号），上上一个版本就是 HEAD^^ ，当然如果回退的版本比较多也可以使用数字表示回退的数量，例如 HEAD~100 （1左边）。例如，想要回退一个版本，可以执行以下命令

```
$ git reset --hard HEAD^
```

同时，Git也支持指定 commit_id ，回退到任何一个已经存在的版本

```
$ git reset --hard 1094adb
```

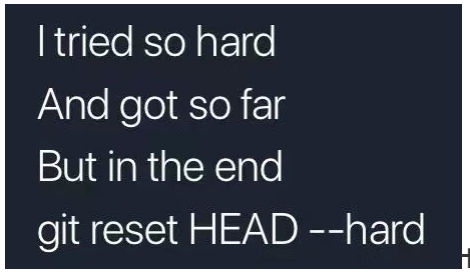
回退时的参数选择

reset 指令有三种参数可以选择，上面选择的是 hard 。它们的用途如下：

```
--hard: 会丢弃工作区和暂存区的修改，未被追踪的文件不受影响；

--mixed: 默认参数，可选择不写，保留当前所有代码，包括工作区和暂存区，并将这些代码一并放入工作区，
只是 HEAD 指向发生了变化，指向命令指定的版本；

--soft: 工作区修改将会原样保留；暂存区中，如果回退之后的版本追踪了该文件，那么所带来的所有的修改，会被保存到暂存区，
如果回退之后的版本未追踪该文件，该文件仍然保留在暂存区，只是变为新增文件，内容为保留最后修改的结果；
未被追踪的文件不受影响。
```



~~remake吧~~

回退之后 commit_id 找不到了怎么办？用下面的命令查看版本撤销记录

```
$ git reflog
```

撤销修改 & 文件删除

执行以下命令以撤销修改

```
$ git checkout -- filename
```

用库中的版本替换工作区的版本，会丢失最近一次修改的内容。

执行以下命令来进行文件删除

```
$ git rm filename
```

删除Git仓库中的某个内容。可以用 `--cached` 命令选择是否保留工作区中的文件。

分支管理

分支的出现，也是解耦思想的一种体现。当执行 `git init` 的时候，默认情况下Git会创建 `master` 分支(或者 `main` 分支)。

创建分支 & 删除分支

- 列出（本地）分支

```
$ git branch
*master
```

此例的意思就是，我们有一个叫做 `master` 的分支，并且该分支是当前分支。

- 创建新分支

```
$ git branch branchname
```

- 切换分支

```
$ git checkout branchname
```

我们也可以在创建时就切换

```
$ git checkout -b branchname
```

- 删除分支

```
$ git branch -d (branchname)
```

分支的内容是与创建该分支时所在分支的内容相同的，使用分支将工作切分开来，从而让我们能够在不同开发环境中做事,并来回切换。

分支合并

可以使用以下命令将任何分支合并到当前分支中去

```
$ git branch
*master
dev
$ git merge dev
```

合并完之后就可以删除分支

```
$ git branch -d dev
Deleted branch dev(wasc1501a2).
```

删除后，就只剩下 master 分支了

```
$ git branch *master
```

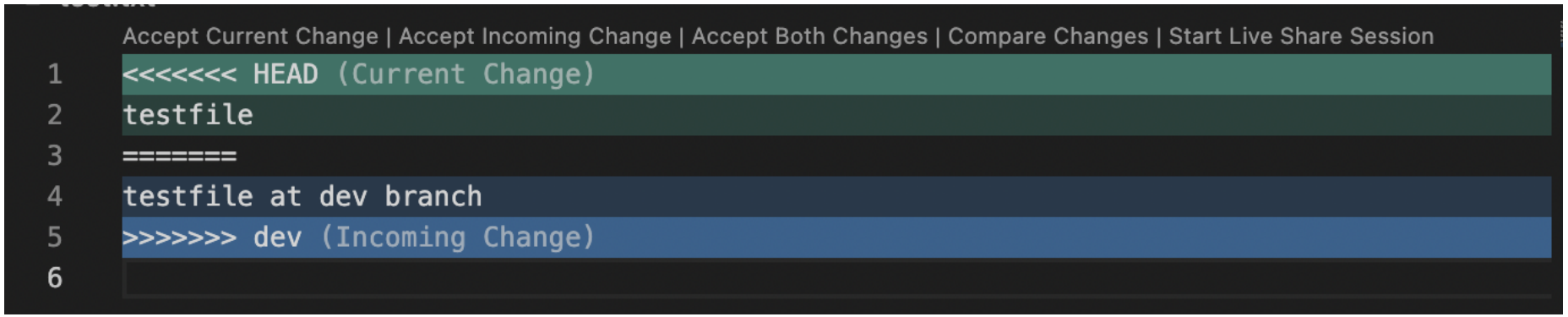
合并冲突

分支合并不仅仅是简单的文件增删操作，Git也会合并修改。然而，当不同分支中的修改存在冲突时，Git便会出现 合并冲突，这时候需要我们手动打开冲突的文件进行修改，修改后 `git add filename` 再进行 `commit` 操作。

虽然我们可以通过命令台解决合并冲突

```
$ git status -sUU newfile.txt
$ git add newfile.txt
$ git status -sM newfile.txt
$ git commit[master 88afe0e] Merge branch 'dev'
```

但是建议使用GUI工具（VSCode和Github Desktop）来操作。



远程仓库

Q: Git和Github是什么关系?

A: Github是提供Git仓库托管服务的代码托管平台，类似的还有GitLab，TsinghuaGit等。

我们要做的

把本地的Git repo放到Github上，或者，把Github上的仓库下载到本地。

准备工作

1. 创建SSH Key。（一对公钥和私钥）

参考链接：https://docs.qq.com/doc/DZmV6YWZQTGh5eWJX

2. 在Github上配置公钥

KaKituken
Your personal account

Switch to another account

Go to your personal profile

- Public profile
- Account
- Appearance
- Accessibility
- Notifications

Access

- Billing and plans
- Emails
- Password and authentication
- SSH and GPG keys
- Organizations
- Moderation

Code, planning, and automation

- Repositories
- Packages
- GitHub Copilot
- Pages
- Saved replies

SSH keys / Add new

Title

Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

Add SSH key

本地仓库关联远程

1. 在GitHub上创建一个仓库
2. 使用以下命令把本地仓库关联到远程

```
$ git remote add origin git@server-name:path/repo-name.git
```

3. 把本地仓库内容推送到远程

```
$ git push -u origin master
```

之后，可以简写使用 `git push`

4. 可以使用下面的命令同步远程的修改

```
$ git pull
```

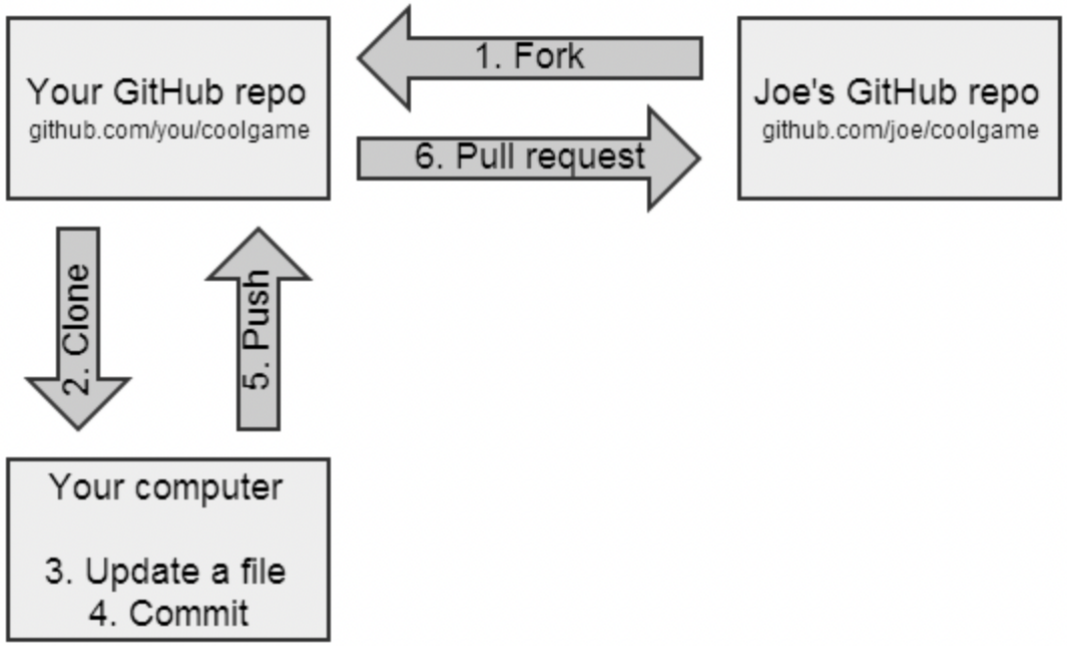
一个好的习惯：在每次 `push` 之前都 `pull` 一下

5. 使用下面的命令删除远程库（解除本地和远程的绑定关系）

```
$ git remote-v
$ git remote rm origin
```

提问：如果你发现了一个别人写好的repo，想把它下载到本地，
添加一个功能再推回去，应该怎么办？

远程仓库下载到本地



1. Fork到自己的Github仓库

直接在创建时选择“从现有仓库创建”即可

2. 把远程仓库克隆到本地

```
$ git clone git@github.com:fsliurujie/test.git --SSH协议
$ git clone git://github.com/fsliurujie/test.git --GIT协议
$ git clone https://github.com/fsliurujie/test.git --HTTPS协议
```

3. 在本地进行开发

4. 将开发后到文件上传并推送

```
$ git commit -m "feat:..."
$ git push
```

5. 提一个pull request，等待合并

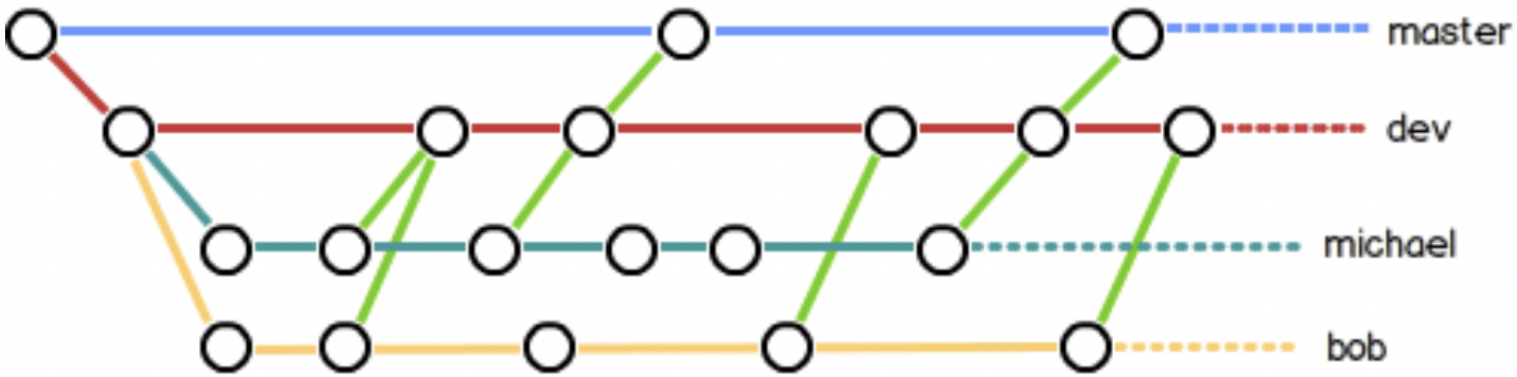
- 什么是pull request?

便于理解的翻译：“我改了一些东西，请拉回去看一看吧”

如何提一个pull request：https://ssast-readme.github.io/ToReader/

除此之外，还可以提issue（也见上）

Git多人协作



多人协作的工作模式通常是这样

1. 克隆Git资源作为工作目录
2. 在本地仓库进行修改和commit操作
3. 可以试图用 `git push origin <branch-name>` 推送自己的修改
4. 如果推送失败，则因为远程分支比你的本地更新，需要先用 `git pull` 试图合并（记得那个好习惯吗？）
5. 如果合并有冲突，则解决冲突，并在本地提交
6. 没有冲突或者解决掉冲突后，再用 `git push origin <branch-name>` 推送就能成功！

- 如果 `git pull` 提示 `no tracking information`，则说明本地分支和远程分支的链接关系没有创建，用命令 `git branch --set-upstream-to <branch-name> origin/<branch-name>` 。

最后，让我们一起逛一下GitHub

更多内容：

- Github CI/CD https://zhuanlan.zhihu.com/p/250534172
- Github Pages https://pages.github.com/

参考资料：《软件学院科协**Git & CI Workshop**》