Git

第一章 初识Git

1.1 什么是Git

Git是一个分布式的版本控制软件

- 版本控制，类似于毕业论文、写文案等，需要反复修改和保留原历史记录
- 软件，类似于Office、QQ等安装到电脑上才能使用的工具
- 分布式
    - 文件夹拷贝
    - 本地版本控制
    - 集中式版本控制
    - 分布式版本控制

1.2 为什么要做版本控制

每次开发新版本，保留之前版本，防止线上版本出错，可以进行回滚和修改

1.3 安装git

https://git-scm.com/book/zh/v2/%E8%B5%B7%E6%AD%A5-%E5%AE%89%E8%A3%85-Git

第二章 Git Start

2.1 第一阶段：单枪匹马开始干

先进入git bash 界面（相当于linux）

在windows中，右键：Open Git Bash here

想要让git对一个目录进行版本控制需要以下步骤：

- 进入要管理的文件夹

```
cd "文件路径"
```

- 执行初始化命令

```
git init
```

- 管理目录下的文件状态

```
git status

注：新增的文件和修改过后的文件都是红色
```

- 管理指定文件（红变绿）

```
git add 文件名
git add .
```

- 生成版本

```
git commit -m '描述信息'
```

- 查看版本记录

```
git log
```

2.2 第二阶段：拓展新功能

```
git add
git commit -m "第一个新功能"
```

2.3 第三阶段：发生事件，版本出问题

- 回滚至之前版本

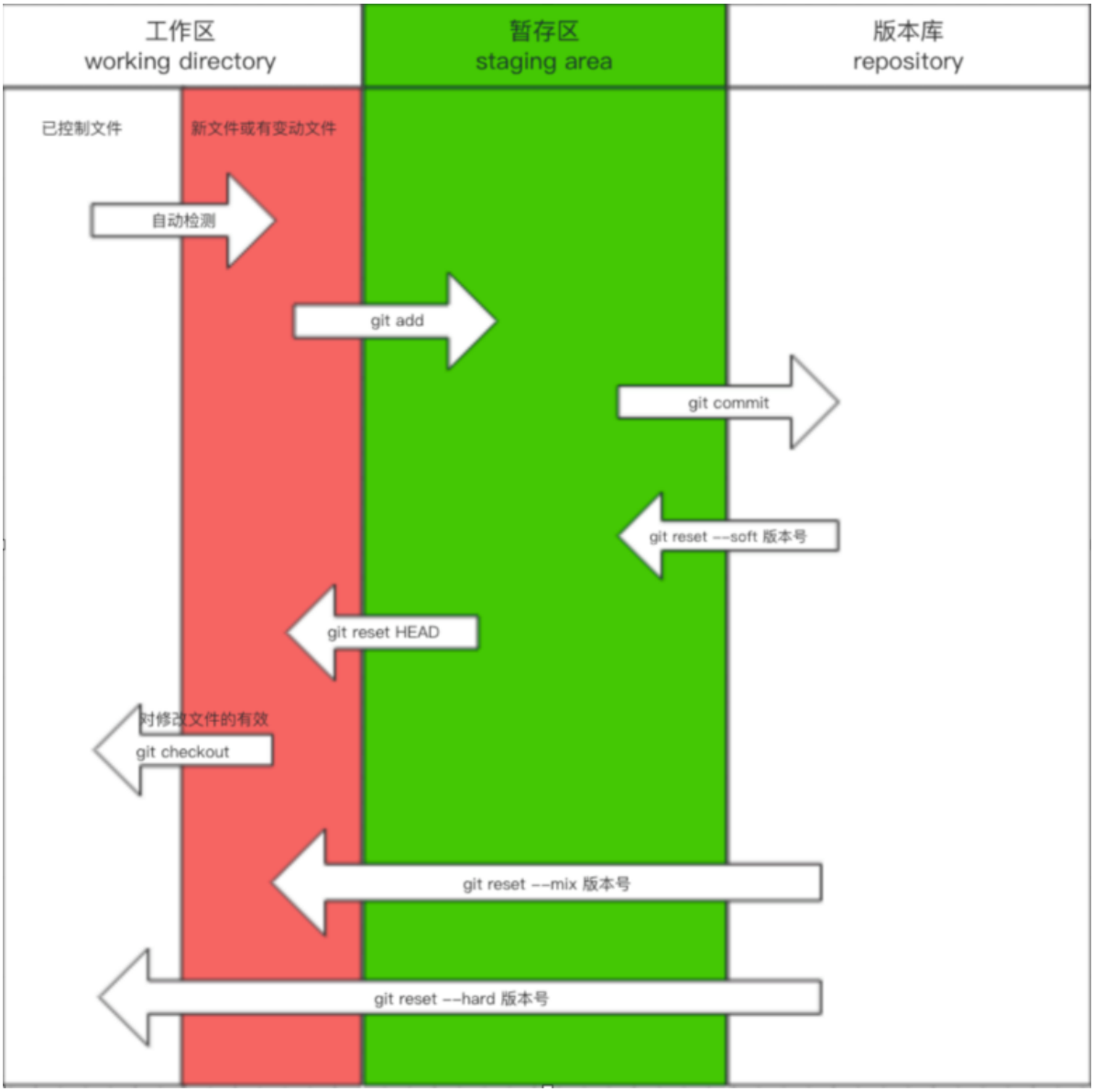
```
git log
git reset --hard 版本号
```

- 回滚至之后版本

```
git reflog
git reset --hard 版本号
```

2.4 小总结

```
git init
git add
git commit
git log
git reflog
git reset --hard 版本号
```

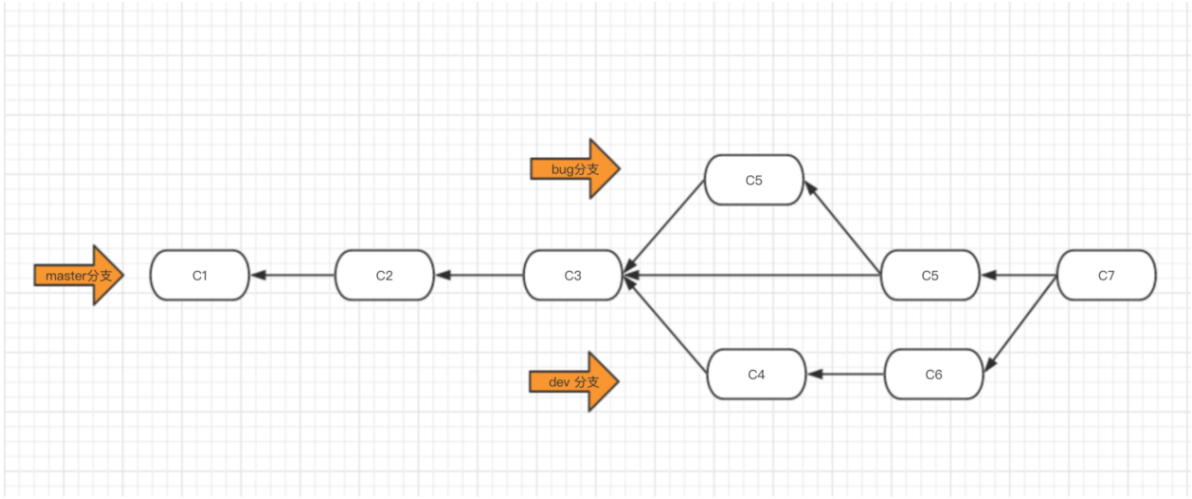


2.5 第四阶段：商城&紧急修复bug

2.5.1 分支

分支可以给使用者提供多个环境，意味着可以把工作从开发主线上分离开来，以免影响开发主线。

2.5.2 紧急修复bug方案



2.5.3 命令总结

- 查看分支

```
git branch
```

- 创建分支

```
git branch 分支名称
```

- 切换分支

```
git checkout 分支名称
```

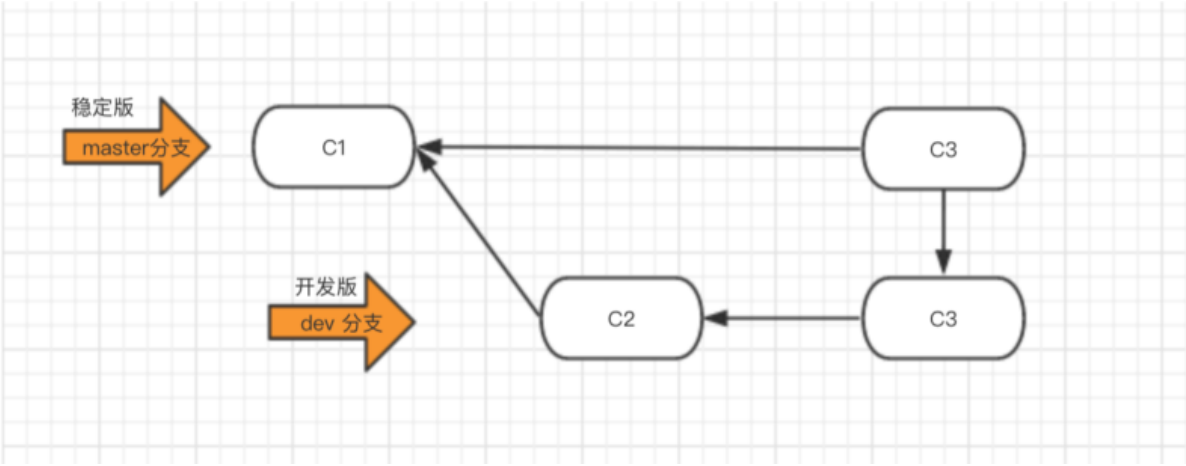
- 分支合并（可能产生冲突）

```
git merge 要合并的分支

注意：切换分支再合并
```

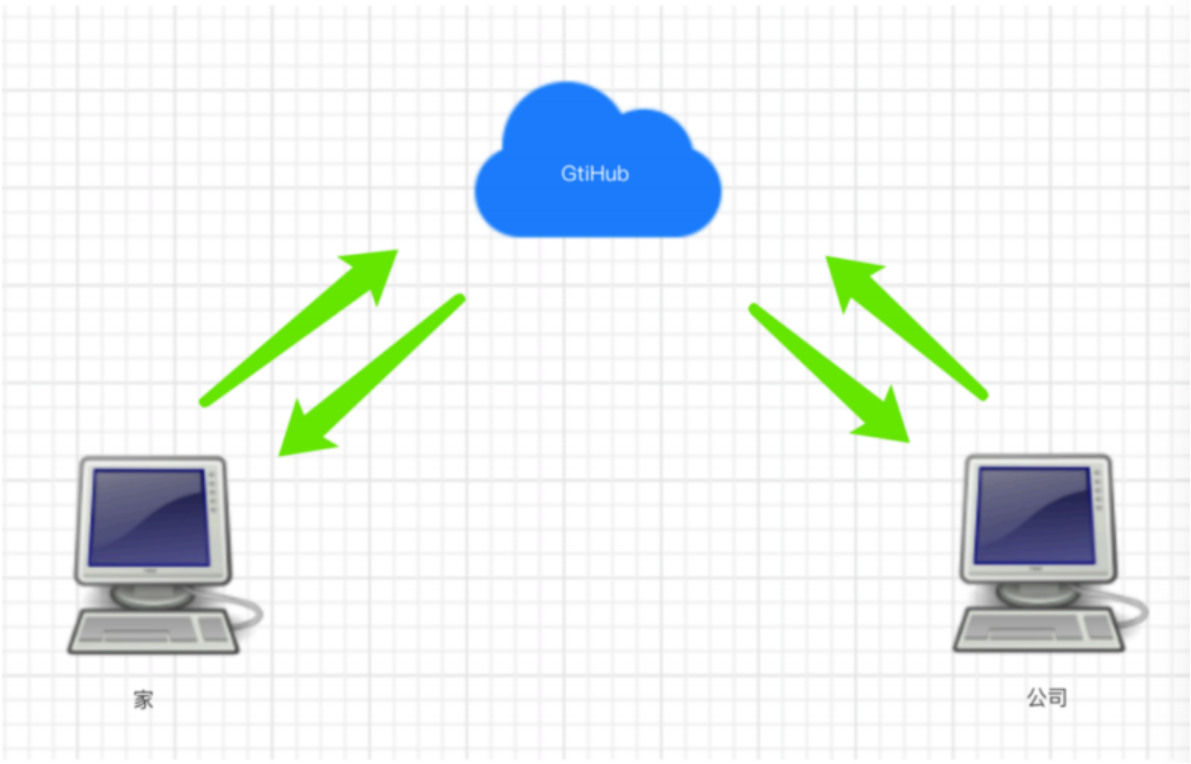
- 删除分支

```
git branch -d 分支名称
```

2.5.4 工作流



2.6 第五阶段：使用代码仓库协助



2.6.1 第一天上班前在家上传代码

首先，需要注册github账号，并创建远程仓库，然后再执行如下命令，将代码上传到github

```
1. 给远程仓库起名
    git remote add origin 远程仓库地址
2.向远程推送代码
    git push -u origin 分支
```

2.6.2 初次在公司新电脑下载代码

```
1. 克隆远程仓库代码
    git clone 远程仓库地址 (内部已实现git remote add origin 远程仓库地址)
2. 切换分支
    git checkout 分支
```

在公司下载完代码后，继续开发

```
1. 切换到dev分支进行开发
    git checkout dev
2. 把master分支合并到dev [仅一次]
    git merge master
3. 修改代码
4. 提交代码
    git add .
    git commit -m 'xx'
    git push origin dev
```

2.6.3 下班回家继续写代码

```
1. 切换到dev分支进行开发
    git checkout dev
2. 拉代码
    git pull origin dev
3. 继续开发

4. 提交代码
    git add .
    git commit -m 'xx'
    git push origin dev
```

2.6.4 到公司继续开发

```
1. 切换到dev分支进行开发
    git checkout dev
2. 拉最新代码(不必再clone, 只需要通过pull获取最新代码即可)
    git pull origin dev
3. 继续开发

4. 提交代码
    git add .
    git commit -m 'xx'
    git push origin dev
```

开发完毕，要上线

```
1. 将dev分支合并到master,进行上线
    git checkout master
    git merge dev
    git push origin master
2. 把dev分支也推送到远程
    git checkout dev
    git merge master
    git push origin dev
```

2.6.5 有一天在公司忘记提交代码

```
1. 拉代码
    git pull origin dev
2. 继续开发

3. 提交代码
    git add .
    git commit -m 'xx'

注： 忘记push了
```

2.6.6 回家继续写代码

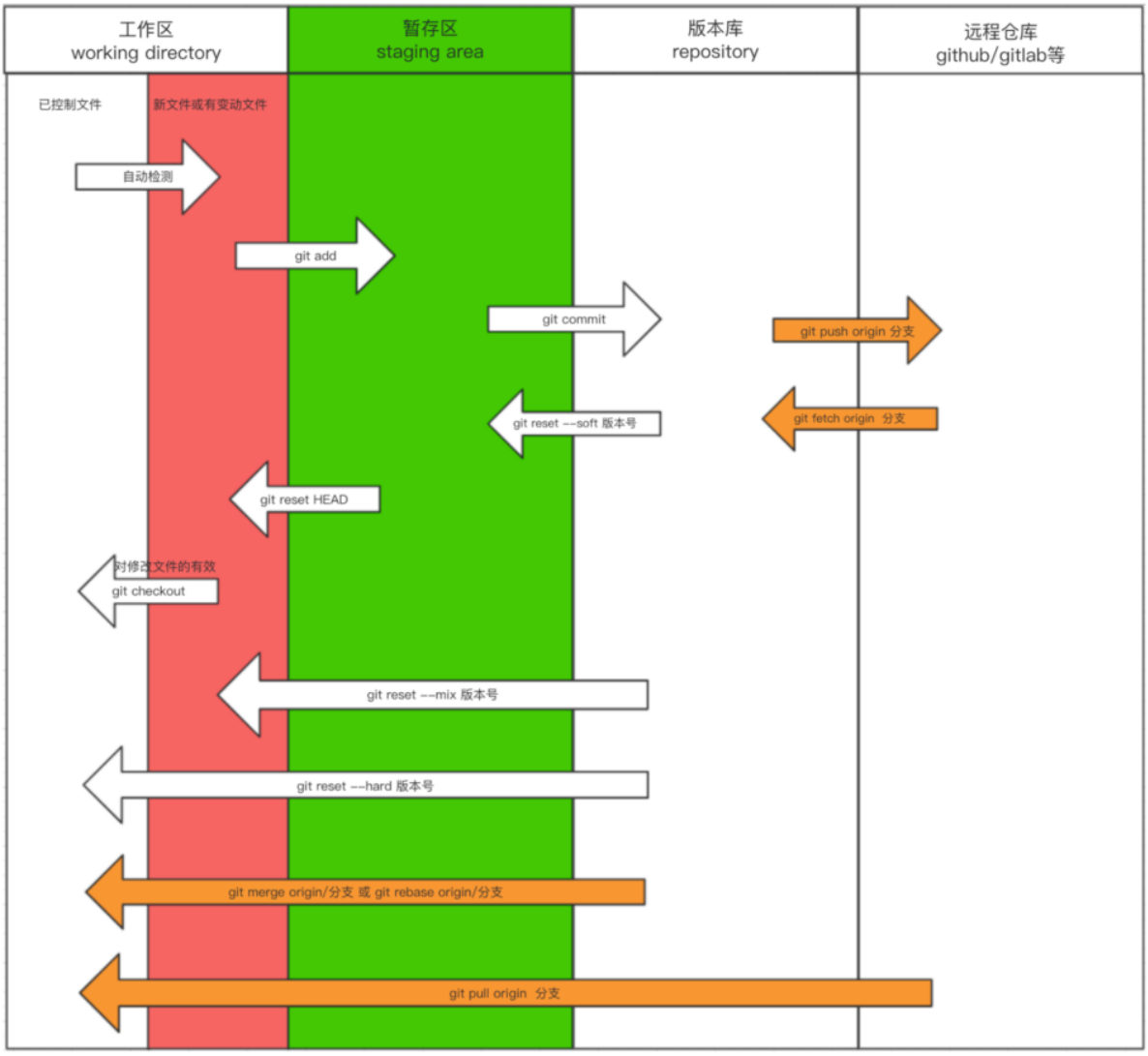
```
1. 拉代码，发现在公司写的代码忘记提交...
    git pull origin dev

2.继续开发其他功能

3.把dev分支也推送到远程
    git add .
    git commit -m 'xx'
    git push origin dev
```

2.6.7 到公司继续写代码

```
1. 拉代码，把晚上在家写的代码拉到本地(有合并、可能产生冲突)
    git pull origin dev

2. 如果有冲突，手动解决冲突

3. 继续开发其他功能

4. 把dev分支也推送到远程
    git add .
    git commit -m 'xx'
    git push origin dev
```

2.6.8 其他

```
git pull origin dev
等价于
git fetch origin dev
git merge origin/dev
```



2.6.9 rebase的作用

rebase可以保持提交记录简洁，不分叉

2.6.10 快速解决冲突

1. 安装beyond compare
2. 在git中配置

```
git config --local merge.tool bc3
git config --local meigetool.path '/user/local/bin/bcomp'
git config --local mergetool.keepBackup false
```

3. 应用beyond compare 解决冲突

```
git mergetool
```

2.7 小总结

- 添加远程连接

```
git remote add origin 地址
```

- 推送代码

```
git pull origin dev
```

- 下载代码

```
git clone 地址
```

- 拉取代码

```
git pull origin dev
等价于
git fetch origin dev
git merge origin/dev
```

- 保持代码提交整洁（变基）rebase

```
git rebase 分支
```

- 记录图形展示

```
git log --graph --pretty=format:"%h %s"
```

第三章 其他

3.1 配置

- 项目配置文件：项目/.git/config

```
git config --local user.name 'justtry'
git config --local user.email 'name@xx.com'
```

- 全局配置文件：~/.gitconfig

```
git config --global user.name 'justtry'
git config --global user.email 'name@xx.com'
```

- 系统配置文件：/etc/.gitconfig

```
git config --system user.name 'justtry'
git config --system user.email 'name@xx.com'

注意：需要有root权限
```

应用场景：

```
git config --local user.name 'justtry'
git config --local user.email 'name@xx.com'

git config --local merge.tool bc3
git config --local mergetool.path '/user/local/bin/bcomp'
git config --local mergetool.keepBackup false

git remote add origin 地址， 默认添加在本地配置文件中(--local)
```

3.2 免密码登录

- URL中体现

  ```
  原来的地址：https://github.com/justtrys/git_learning.git
  修改的地址：https://用户名：密码@github.com/justtrys/git_learning.git

  git remote add origin https://用户名：密码
  @github.com/justtrys/git_learning.git
  git push origin master
  ```

- SSH实现

  ```
  1. 生成公钥和私钥(默认放在~/.ssh目录下, id_rsa.pub、id_rsa私钥)
      ssh-keygan
      ssh-keygan -t rsa(ed25519/dsa/ecdsa)
  2. 拷贝公钥内容，并设置在github中
  3. 在git本地配置ssh地址
      git remote add origin git@github.com/justtrys/git_learning.git

  4. 以后使用
      git push origin master
  ```

- git自动管理凭证

3.3 git忽略文件

让Git不再管理当前目录下的某些文件

```
*.h
!a.h
files/
*.py[c|a|d]
```

更多参考：https://github.com/github/gitignore

3.4 github任务管理相关

- issues，文档以及任务管理
- wiki，项目文档

版本控制——git管理文件夹

1. 进入要管理的文件夹
2. 初始化
3. 管理
4. 生成版本

git bash here

```
# 先进入要管理的目录
git init   #初始化

git status  #检测当前文件夹中的文件状态

git add index.html

git status   #管理起来的就是绿色的，没有的就是红色

git add .   #全部管理

git commit -m '第一个版本v1'   #生成第一个版本

git status  #这个时候啥也没有了，说明生成了版本
#这个时候修改其中的文件
git status  #出现红色

git add .
git commit -m 'v2'
```

可能会报错

先进行个人信息的配置：用户名、邮箱

然后才能生成版本

```
touch 文件      # 生成文件

git status

git commit -m 'xxx'

git config --global user.email ""
git config --global user.name ""

git commit -m 'xxx'
```

git三大区域

工作区、暂存区、版本库

工作区：

已管理/新增、修改

git add .　　# 提交到暂存区

git commit -m ''　　# 提交到版本库

MAC的终端用iterm

mkdir 新建文件夹

```
git init
git status
git add
git commit -m ''
git log  # 看到版本日志

git reset --hard 版本号
回滚到之前的版本
```

```
git log
git reflog
```

分支

```
git branch  #创建分支

git branch dev #创建分支dev

git checkout dev #切换到dev分支

git status

git commit -m 'c4'

git checkout master

git merge

git branch -d name

git clone

cat filename
vim filename
git branch

git checkout dev
```

```
git pull origin dev
touch a2.py
git add .
git commit -m ''

:wq
```

```
git rebase -i HEAD-3


git branch
git tag -a v1 -m '第一个版本'
git push
```

```
vim .
vim .git/config
vim ~/.git/global
```

Linux

```
ip addr
cmd
远程连接linux
ssh root@ip地址

shutdown -h now #现在关机
shutdown -h 10 #十分钟之后关机
shutdown -r now  == reboot #重启
reboot #重启
exit


ls == list #查看目录下有哪些文件
touch 文件名  touch dms.txt  #新建文件
touch {1..10}.txt
touch {a..f}.txt
rm a.txt    y  #删除文件
rm -f dms.txt -f  force 强制
rm {1..10}.txt

cp == copy  #复制文件
cp  文件  新的文件名


ctrl+c  #打断正在执行的命令

mkdir      #新建目录directory
mv     mv dms smd #重命名目录
cd
cd..
```