# Git基础入门

## 一、安装与配置

### 1.git介绍

什么是git? Git是一个免费开源的分布式版本控制系统。

版本管理 就比如git是一个档案馆，它可以记录你在什么时候写了什么代码

分布式 主要是区别于集中式的版本管理系统 比如sv

集中式&分布式 主要区别在于 集中式他只有一个档案馆 而分布式是每个人都有一个档案馆 可以在自己的档案馆里 处理自己的版本

假如说你自己的版本觉得已经做好了 你就可以拿着自己的一个版本去和别人的版本进行一个合并操作

这个版本管理主要是方便大家进行一个协同合作或者是以后出事故找人背锅



Github gitlab gitee git 简单来说前者是后者的一个托管平台 当你把自己的代码仓库做好以后上传到github 或 gitlab、gitee上。如果你的代码仓库是公开的话。其他人可以对你的仓库进行一些贡献。

### 2.下载与安装:

<https://git-scm.com/download/>

### 3.使用入口

win：右键菜单—git bash

mac：终端窗口

### 4.基础配置

a.首次使用添加身份说明，使用以下两个命令：

  $ git config --global user.name "你的昵称"

  $ git config --global user.email [邮箱@example.com](mailto:邮箱@example.com)

只是一个说明性的 其实想填啥就填啥 只是留个标记而已

b.创建仓库

①在项目文件夹下使用git bash输入$git init 文件夹会变成一个仓库

②使用他人项目创建仓库 克隆

clone项目

$git clone 项目url

## 二、状态与提交版本

前言 当我们创建一个仓库后 我们会赋予这个仓库的每一个文件都有个状态

如果是自己新建的仓库的话所有文件都是一个未被跟踪的状态

### 1.跟踪

a.跟踪文件

$ git add <name>

b.跟踪当前目录

$ git add .

### 2.取消跟踪

a.rm删除

$ git rm <name>

b.保留但不跟踪

$ git rm-cache <name>

### 3.文件状态修改

修改后缓存 / 取消缓存

a.将修改文件缓存

git add <file-name>

b.取消缓存

git reset HEAD <name>

c.提交缓存的修改

git commit

d.文件状态总结

一个文件的四个状态



e.git commit 具体操作

commit 操作

①git commit 进入提交界面,

按" i "键进入输入模式后输入本次提交详情,

然后esc退出编辑模式, 按" : "进入命令栏, 输入"wq"保存并退出

②git commit -m ' 你对提交内容的描述 '

commit 的简洁操作

③git commit -a

连带未暂存文件一起提交

git commit -am '提交描述'

commit -a和-m

④git reset head~ --soft

取消本次提交

使用该命令取消本次提交, 但是首次提交不可撤回.

f.查看状态

查看文件状态

①git status

红色代表 已修改, 未暂存

绿色代表 已暂存

提交后, 则不显示

②git diff

更详细地查询修改

详细查看文件的第几行第几个字母被修改了

③git log

提交历史

查看提交历史信息

查看所有分支的提交

git log --all

查看所有分支的提交

结合graph食用更佳

图形化查看所有分支的提交

git log --all –graph

美化输出界面

git log--pretty

git log--pretty=oneline

自定义格式 $ $ git log --pretty=format:"%h - %an, %ar : %s"



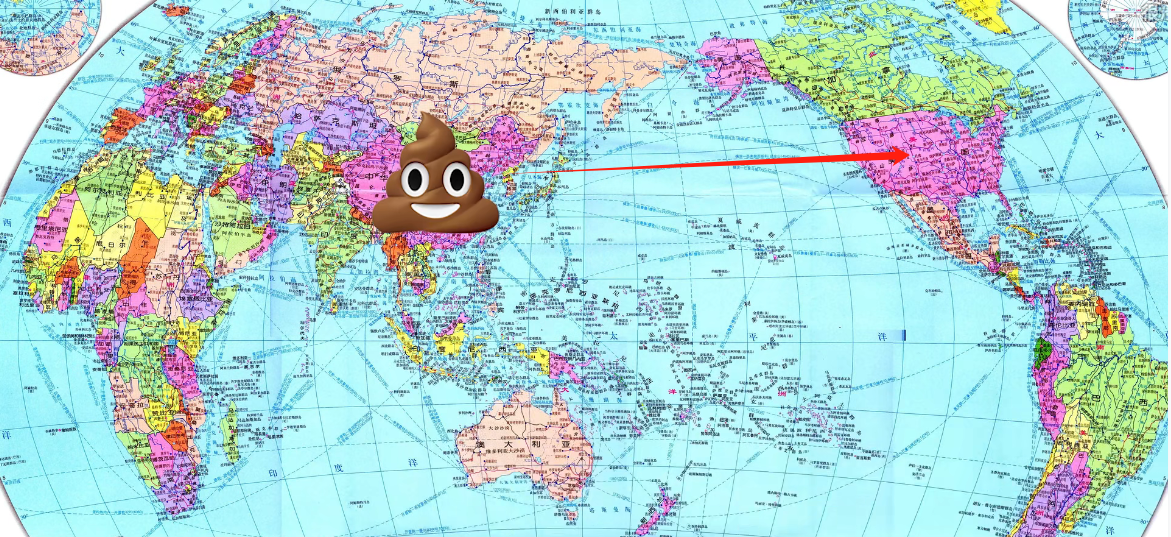
④git log--graph

图形化界面查看(结合分支)

分支的概念

## 三、远程仓库

冲出亚洲!



注册比较简单, 就不教了

新建远程仓库

### 1.链接远程仓库到本地﻿

本地添加远程仓库

git remote add origin 远程仓库链接

### 2.重命名仓库

重命名

git remote rename 目标仓库名 修改内容

### 3.推送到远程仓库

a.﻿本地代码推送到远程仓库

git push 仓库名 分支名

b.GitHub已禁止使用用户名与密码验证

①使用token令牌验证

登录验证

②简单方式:ssh鉴权

进入用户ssh目录 cd ~/.ssh

Ssh-keygen -t rsa -b 4396 -C “12970398154@qq.com”

简单方式: ssh鉴权

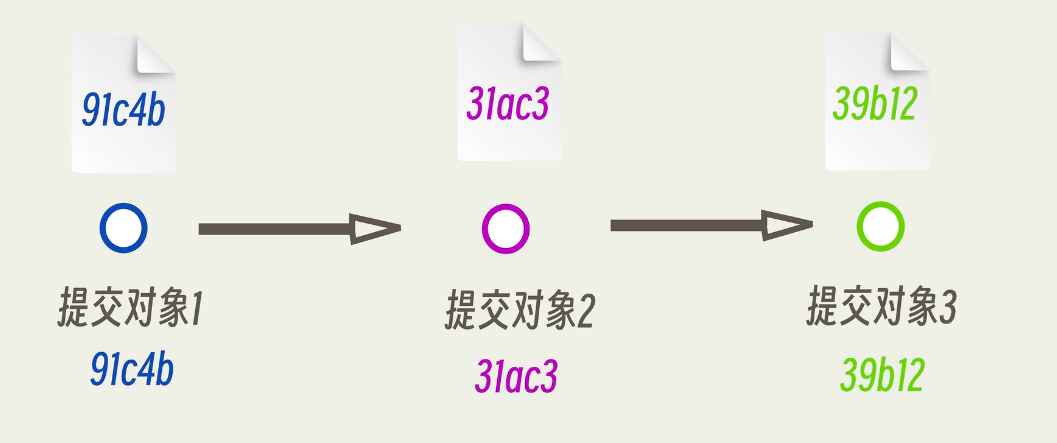
## 四、分支

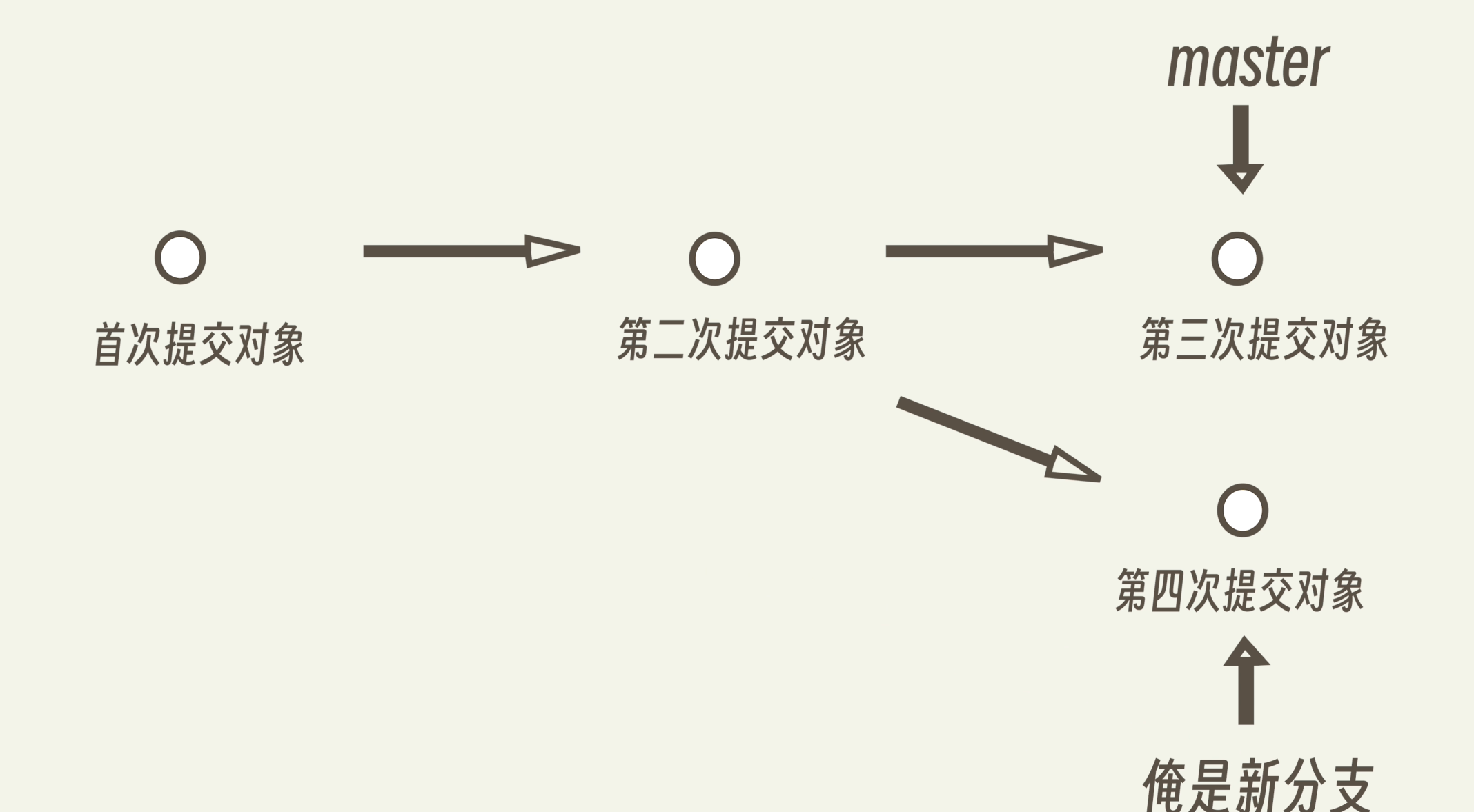
### 1.分支的概念

分支的概念

每次提交生成一个新版本的时候都会生成一个提交对象。每一个提交对象都有一个独一无二的哈希值。其实分支就是一个包含这个哈希值的的一个文件

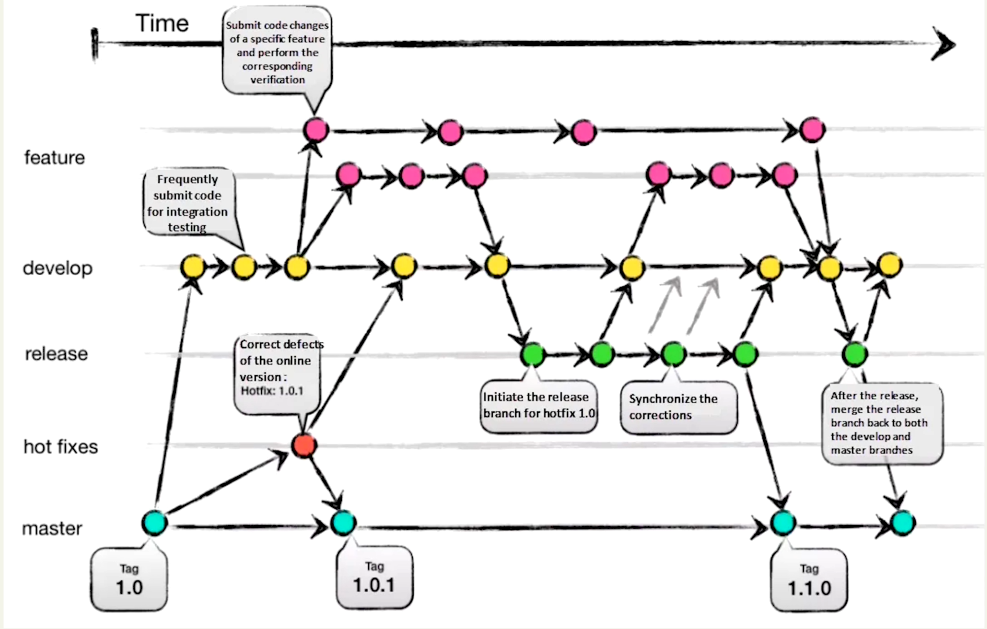
初始化仓库的时候就已经新建了一个master分支 提交分支都是在这个master分支上进行的。当我每进行一次提交,我们的分支也会跟着我们的提交对象进行移动





### 2.经典git模型

git模型



Master 为一个发布版的分支 比如我们开发软件的1.0版本

Feature 开发新特性 再这个分支中进行开发

Develop feature分支进行合并然后对其测试

Release 测试验证没问题 发布至release分支 做过向往验证后将其合并到master分支

Hot fixes 修复bug用的

### 3.创建分支

创建分支

git branch 分支名

新建并切换到该分支

git checkout –b 分支名

### 4.查看分支

git branch --list

### 5.切换分支

切换分支

git checkout 分支名

## 五、分支合并

合并分支

### 1.无冲突合并

在 合并至 的分支使用

git merge 要合并的分支

### 2.分支冲突 merge conflict

18:49

merge conflict

将 分支2 合并到 master分支 时, 与 分支1 冲突了. 原因是 分支1 和 分支2 修改了同一处内容.

19:07

解决冲突

git status 查看哪里有冲突

vi 到冲突文件中, 选择一个分支的内容保留下来, 保存退出

git add 文件名

git commit -m '提交描述'

git log --all --graph 查看合并状态

## 六、推拉与远程跟踪分支

### 1.推送

推送

git push 仓库名 分支名

或者

git push -u 仓库名 分支名

第一次使用 -u 指定推送目标后, 此后可直接使用git push

### 2.拉取

拉取分支

git fetch

远程分支本地化

git checkout 远程分支

git checkout -b 本地分支名 远程分支

git checkout --track 远程分支

## 七、贮藏功能

### 1.git stash

stash演示

代码写到一半有13事儿来了, 要切换到其他分支是不允许的, 可以把当前分支修改的东西储藏起来再切换.

### 2.git stash apply

切换回分支后恢复贮存内容

切换回来后, 恢复之前存储的内容

### 3.多次存储

存储多次

a.回看存储记录

git stash list

b.恢复指定记录

恢复某次记录

git stash apply stash@{记录号}

c.恢复并删除记录

git stash pop

①恢复并删除最近一次记录

git stash pop

②删除指定记录

git stash drop @stash{记录号}

## 八、重置与变基

### 1.reset(重置)

a.head

head~ 的含义

head: 当前的提交

head~: 上次的提交

head~2: 倒数第二次的提交

b. --soft

--soft 的含义

仅取消commit操作, 把修改文件暂存.

如果不加 --soft 则表示恢复到暂存前, 修改的内容是存在的.

c. --hard

--hard 的含义

取消暂存, 还取消修改内容, 彻底回到上次提交的状态.

不推荐使用, 可能丢失数据.

2.rebase(变基)

变基(搬家)

a.将B分支的修改移动到A分支

git rebase

git checkout B

git rebase A

b.注意事项

注意远程分支

注意他人在远程分支二次开发时, 审慎使用变基

c.交互式操作

rebase -i