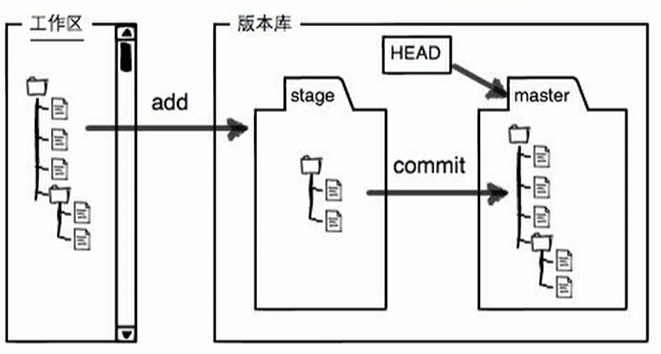
# git version

# 代码开发流程

1. 系统开发，编写代码
2. 提交代码给Git本地库
3. 将代码提交到git远程库，分享给团队其他人
4. 从远程库获取最新代码
5. 继续修改编写代码
6. 重复第二步及以后的操作

# Git基础概念



1. 本地工作文件夹
2. Git索引区（stage）
3. Git库（repository）
   1. Local： 本地库
   2. Remote：远程库

# Git使用

## 建立git库

# git init 在本地当前文件夹下创建一个git库；可以对当前文件夹下的文件进行版本管理

## 设置Git基础信息

# git config -l #查看git配置信息

# git config –global user.name “Stone”

# git config –global user.email [stone@test.com](mailto:stone@test.com)

# git config –global color.ui true

# git config –l # 验证配置

## 获得git帮助信息

# git config help #windows在浏览器打开帮助文件

# 或git help config, git help commit, git help branch 等

## Git提交（commit）

* + 建立、修改本地文件（在本地工作文件夹中）
  + 追加文件到索引区（stage）
  + 提交文件到本地库

# git status #查看本地工作文件夹的状态

# git add index.html #将文件添加到索引区, 或 git add . 添加所有文件到索引区

# git status查看状态，当有新文件或文件更改后，会有提示

# git commit –m “message, 注释” #将索引区内容提交到本地库

# git log #查看提交历史

# git log -3 #只显示最近3次提交历史

# git log –oneline #缩短每次提交的记录为一行

# git log –p #显示详细内容，包括更改的内容

# git log –stat #显示每次更改的统计信息

# git log –help , git help log

## Git diff [--cached]比较修改内容

#git diff #文件add到stage之前，可以使用git diff查看更改内容

# git diff --cached #文件add到stage之后，必须加—cached选项才能查看更改内容

## Git的文件操作

* + # git add [file1 file2 file3 …] #添加 文件到stage
  + # git add . #添加所有文件及子目录到stage
  + # git mv <filename1> <filename2> #add到stage后，给文件改名
  + # git rm <filename> # 将文件从stage中删除，相当于git reset, git checkout 两步操作

## Git忽略管理

* + 设置git忽略的文件，这些文件不参与git库的提交和管理
  + # 编辑 .gitignore 保存在git的主工作目录下, 增加一行：\*.tmp 表示忽略所有以tmp结尾的文件，包括子目录下的tmp文件； 文件中注释行使用#符号

## Git更新最后一次的提交

* + Git commit以后，发现有错误，需要重新更改并提交，但不想在git log中生成一条新的记录；
    - 可以使用 git commit –amend
    - # git commit –am “message” [--amend] #追加message信息当上次提交信息

## Git 取消文件更改

更改文件后，可以使用git checkout返回原来状态

* + 如果还没有add到stage，可以使用git checkout -- <filename> 放弃更改
  + 如果已经add到stage, 可以使用git reset HEAD <filename> 将更从索引区删除

## Git返回到过去

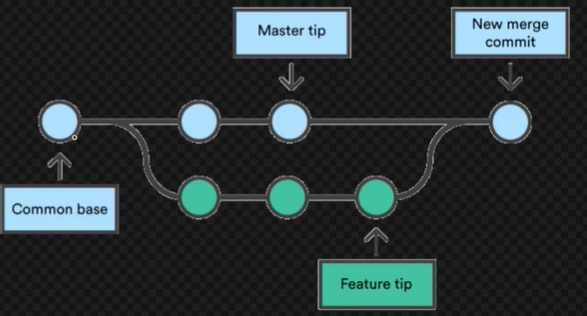
### Git reset

* + # git reset –hard HEAD #使用该命令将本地文件变化、和stage中的变化恢复到本地仓库head的内容；本地更改将丢失（需要提前备份）
  + # git reset –hard HEAD~ #相当于HEAD~1
  + # git reset –hard HEAD~n
  + # git reset –hard <id> #通过commit­\_id恢复

### Git reflog

# git reflog列出所有commit的commit-id，可以使用git restart –hard <id> 恢复

## Git使用分支branch



### 创建、切换分支

* + # git branch #列出所有分支，星号标注为当前分支
  + # git branch <branch\_name> #创建一个分支
  + # git checkout <branch\_name> #切换到某个分支
  + # git checkout –b <branch\_name> #创建一个分支，并同时切换到该分支

### 合并分支

# git merge <branch\_name># 把别的分支合并到当前分支

# git branch –d <branch\_name> # 合并完后，将某个分支删除

### 制造分支冲突

当多人同时在不同分支编辑一个文件，当merge时会出现冲突

### 解决分支冲突

在主分支master中更改文件，然后commit

## Git使用tag进行版本管理

# git tag #显示当前所有tag

# git tag <tag\_name> [commit\_id] #给当前commit, 或某个commit\_id 加tag

# git show <tag\_name>

# git tag –d <tag\_name>

## Git使用别名alias

在git中将经常使用的命令以别名方式使用；如：

# git config –global alias.co checkout #使用co代替checkout

# git config –global alias.st status # 使用st代替status

## Git连接远程服务器

### 本地生成ssh key

# git bash 桌面右键点击git bash

# 输入ssh-keygen, 会在C:\Users\Stone\.ssh\ 目录生成ssh key

# copy 生成的id\_rsa.pub文件中的内容，copy到github的sshkey中；profile>settings>SSH and GPG keys > new SSH keys

### 连接远程git仓库

# git clone [url] #把远程库的master分支克隆到本地文件夹；自动设置git remote 服务器

# git remote add origin [https://](NULL) #连接远程git仓库， 在本地已经创建了git仓库后，执行git remote add连接远程git仓库；

# git remote –v #列出连接的远端服务器

# git remote remove/rm [remote\_name] #断开与远程库的连接

### 与远程git仓库同步文件

# git branch dev remotes/orgin/dev #以远端dev分支为基础，创建本地dev分支；缺省以本地Master为基础

# git branch –a # -a=all, 列出所有分支，包括远端分支，不加-a只显示本地分支

# git push [origin] [master] #将本地当前分支 push到远端的origin库的master分支

# git status 也会列出与远程库的差异

# git pull [origin] [master] #从远程git仓库下拉内容到本地仓库