Git使用基础

zliu@kalengo.com(leo)

Git使用基础

- Git 介绍
- Git 基础
- Git 工作流
- Gitlab and Github

Git 介绍

- 分布式版本控制系统
  - 不只最新版本的文件快照，是把**代码仓库完整地镜像**下来
  - 服务器发生故障，可以用任何一个镜像的本地仓库恢复
  - 可以和若干不同的远端代码仓库进行交互
- **Linus Torvalds** 为了更好地管理linux内核开发而创立
- 强大的分支管理

Git 基础(安装)

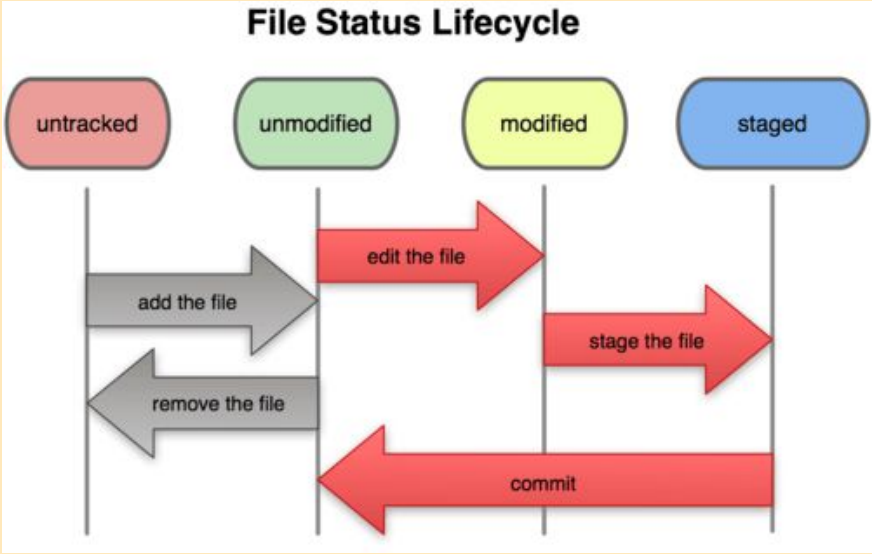
- install
  - linux
    - sudo yum install git
    - sudo apt-get install git
  - mac & windows
    - download form http://git-scm.com/download/
    - brew install git (mac)

Git 基础(配置)

- config
  - git config --global user.name “leo”
  - git config --global user.email “zliu@kalengo.com”
  - git config --global color.ui true
  - git config --list (show current git config)
- config file
  - --system (For writing options: write to system-wide $(prefix) /etc/gitconfig)
  - --global (For writing options: write to global ~/.gitconfig file)
  - --local (For writing options: write to the repository .git/config file)
  - for git project config (local > global > system)

Git Command (local)

- git init (To initialize a Git repository)
- git status (To see what the current state of our project)
- git add (add it to the staging area)
- git diff (Show changes between two files)
- git commit (commit file from staging area)
- git rm (remove file from git repository)
- git commit -a (git add && git commit)
- git log (show history)



Git Command (branch)

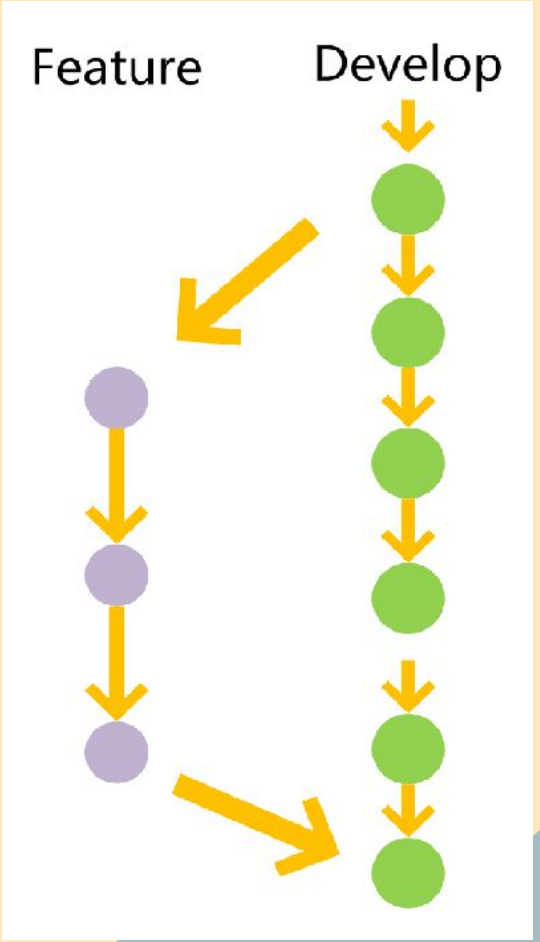
- git branch
  - List, create, or delete branches
- git checkout
  - Switch branches or restore working tree files
- git merge
  - Join two or more development histories together

Git Command (remote)

- git clone
  - Clones a repository into a newly created directory
- git remote
  - Manage the set of repositories ("remotes") whose branches you track
- git fetch
  - Fetch branches and/or tags (collectively, "refs") from one or more other repositories
- git pull
  - Incorporates changes from a remote repository into the current branch, git pull is shorthand for git fetch followed by git merge FETCH_HEAD
- git push
  - Updates remote refs using local refs, while sending objects necessary to complete the given refs.

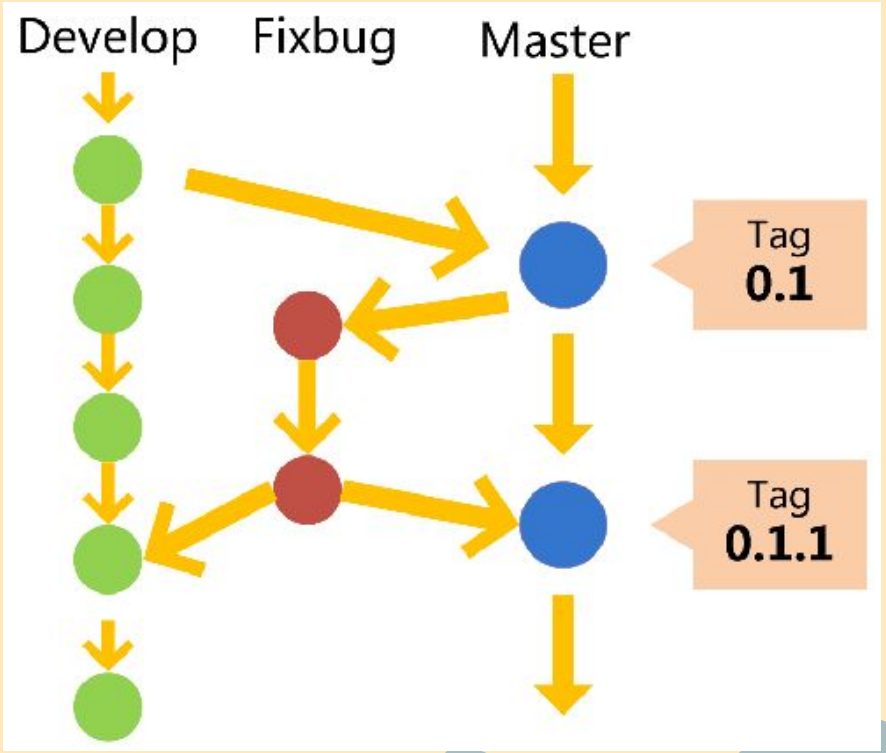
git develop flow

- 创建一个功能分支
  - git checkout -b feature-x develop
- 开发完成后, 将功能分支合并到 develop分支
  - git checkout develop
  - git merge --no-ff feature-x
- 删除feature分支
  - git branch -d feature-x



git fixbug flow

- 创建一个修补bug分支：
  - git checkout -b fixbug-0.1 master
- 修补结束后，合并到master分支：
  - git checkout master
  - git merge --no-ff fixbug-0.1
  - git tag -a 0.1.1
- 再合并到develop分支：
  - git checkout develop
  - git merge --no-ff fixbug-0.1
- 最后，删除"修补bug分支"：
  - git branch -d fixbug-0.1



Git flow practices

1. anything in the master branch is deployable
2. create descriptive branches off of master
3. push to named branches constantly
4. merge only after pull request review
5. deploy immediately after review
6. Everyone commits to the baseline every day

参考

- learn git
  - https://try.github.io/levels/1/challenges/1
- git 简易配置
  - http://rogerdudler.github.io/git-guide/index.zh.html
- a successful git branch model
  - http://nvie.com/posts/a-successful-git-branching-model
- Git远程操作详解
  - **http://www.ruanyifeng.com/blog/2014/06/git_remote.html**
- Git分支管理策略
  - **http://www.ruanyifeng.com/blog/2012/07/git.html**