安装GIT

MAC 安装包下载地址 https://git-scm.com/download/mac

申请帐号

JIRA流程： 《帐号管理流程使用手册》

现在有两套运行的GIT服务器。
老的 git@lib.tap4fun.com
新的 git@git.tap4fun.com

老服务器是目前线上项目都在使用的仓库，使用gitolite 管理
新服务器用gitlab搭建，帐号密码使用公司域密码。后面会逐渐从老服务器迁移到新服务器。

GIT 简介

1. 版本控制系统

2. 本地提交
GIT不需要中心服务器来管理仓库，每一份GIT代码都是一个本地仓库，多个仓库之间可以互相同步代码。这样的好处是可以离线提交，降低对中心仓库的依赖。另外GIT在本地保留这项目所有历史信息，在获取历史信息时是直接在本地读取，而不需要访问中心服务器。

3. 使用指针管理分支
分支本质上是一个指向索引对象的可变指针，而每一个索引对象又指向文件快照。特别是像我们服务器这种需要频繁创建分支的非常方便，快速

教程 http://rogerdudler.github.io/git-guide/index.zh.html

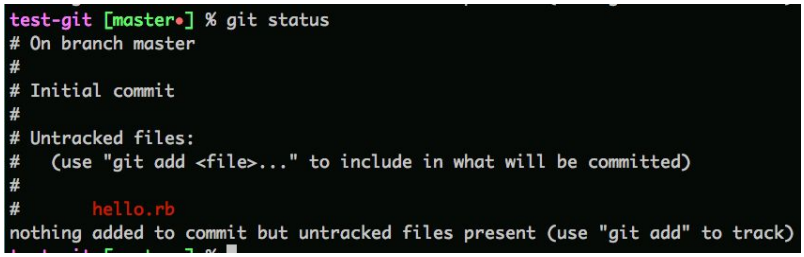
基础练习

在本地创建仓库

```
mkdir test-git
cd test-git
echo "puts 'hello git'" > hello.rb
```

这时test-git是一个普通的包含文件的目录
git init # 初始化git仓库
这时会在本目录下多一个.git目录，包含所有本地仓库所有的git信息。

git status #查看当前状态



如图，“Untracked files” 表示没有被GIT 纳入版本控制的文件。这里要了解文件的三种状态，详情点击http://git-scm.com/book/zh/%E8%B5%B7%E6%AD%A5-Git-%E5%9F%BA%E7%A1%80#文件的三种状态。

git add hello.rb #把hello.rb 纳入git 管理
git status
echo "puts 'i am modified'" >> hello.rb
git status
git commit -a -m ‘test commit’ #提交所做修改
git status
git show
git log

清除修改
如果你想清除当前所做但是还没提交的修改，则可以使用git checkout — . 使所有代码回退到最新的一次提交。

此时你已经可以使用git管理你的代码了。

分支和标签

- **分支基础**

不同的分支可以管理不同版本的文件。默认情况下git工作在master分支上，所做的任何更改都只在master上保存。如果这时需要在当前基础上同时开发另外一个版本，则需要创建新分支，并在新分支上进行新版本开发。

使用 git branch 命令可以管理分支

git branch #查看分支列表
git branch new-feature ＃在当前master基础上创建 new-feature分支

git checkout new-feature #从master分支切换到 new-feature分支
也可以使用 git checkout -b new-feature，直接创建并切换到new-feature分支

每创建一个分支就创建一个新的指针，在git中创建分支是非常轻量的，我们可以大量的创建分支，而不用担心速度。但是我们要养成即使清理不用分支的习惯，避免过多的分支给造成版本上的不确定。

echo "puts 'i am at new-feature branch'" >> hello.rb
git commit -a ‘test-branch'

git checkout master 查看 hello.rb的内容

也可在当前分支下使用 git diff master 查看变更内容

- **合并分支**

合并new-feature的代码到master上
git checkout master 先把分支切换到master上
git merge new-feature 把new-feature上的更改合并到matser上。

合并完成之后如果不再对new-feature进行开发则应该删除new-feature分支。
git branch -d new-feature .如果在删除有更改的分支前没有合并到其他分支上则会报错，可以强制删除
git branch -D new-feature 强制删除一个分支

标签

标签就是各种历史版本的别名，因为git的版本号是40 个十六进制字符组成的字符串，使用标签可以使用很易懂的标签名方便的找到需要的版本。
git tag ＃查看标签列表
git tag test-tag ＃在当前版本上创建一个名叫test-tag的标签

分支管理规范：《T4F-WI-QA-009-服务端代码管理规范V01.00》
在合并新分支前建议在当前master分支的基础上新建一个标签，以便于回退代码。
比如当前日期是2016-10-27，则tag名为stable-20161027
git checkout master #切回到master
git pull origin master #更新最新的代码到本地
git tag stable-20161027 #设置标签
git push origin stable-20161027 #上传分支
如果部署时需要回退到合并之前的版本，这时就可以执行 git checkout stable-20161027

常用命令-基本

- init 初始化一个仓库
- add 添加一个新文件
- branch –d 删除 –m 重命名
- chckout 切换到指定的分支（指针）
    - –b 切换并创建分支

    - -- . 清除修改
- commit 提交跟新
    - --amend 改写上次提交的注释
- log 查看当前分支的日志
    - –author= --before= --after=
    - --branches= --tags= --remotes=
- status 查看当前状态
- merge 合并分支
    - --no-ff 创建一个新的分支，即使只是简单的向前移动指针
- stash 临时保存

远程协作

以上操作都是在本地进行的，所有变更都保存在本地机器上。要想多人同时修改同一仓库，则需要使用到git的远程命令。
git 远程命令有

- clone 克隆远程仓库
- fetch 获取远程仓库的分支信息
- pull 获取远程仓库信息并合并分支
- push 把本地分支信息推送到远程仓库中
- remote 查看远程仓库信息

为了方便大家练习，我在公司的git 上新建了一个公共仓库用于大家练习
git@git.tap4fun.com:base/test-git.git 需要大家先申请git 帐号才能clone

```
git clone git@git.tap4fun.com:base/test-git.git #从远程仓库下载代码
cd test-git
git checkout -b my_test_branch  #基于当前分支新建一个新的分支
echo “ test 1” >>  test1
echo “test 2” >> test2
git commit -a -m ‘ save’  #提交变更到你的分支上
git push origin my_test_branch #把你的分支推送到远程服务器
git pull origin my_test_branch #从服务器上拉取最新代码并合并到你的本地分支
```

服务器上部署

- **配置hosts**

在公司外网服务器需要配置hosts
具体请看 http://git.tap4fun.com/base/gitlab/wikis/home

- **拉取代码**

git 协议是基于ssh的，因此认证也是基于ssh的。只要你将你本地的公钥签名一起登录服务器就能直接克隆。

具体步骤，

ssh-add #添加你的公钥摘要
ssh-add -l #查看当前已经加载的公钥摘要
ssh -A developer@hosts # 登录服务器
git clone git@git.tap4fun.com/repo.git

错误的用法：
sudo git clone xxxxxxx
sudo su - otheruser && git clone xxxxx

- **限制**

1. 只能在公司内网和服务器网段内拉取代码
2. git镜像的代码只能读取，不能push