1. Git
   1. 什么是Git

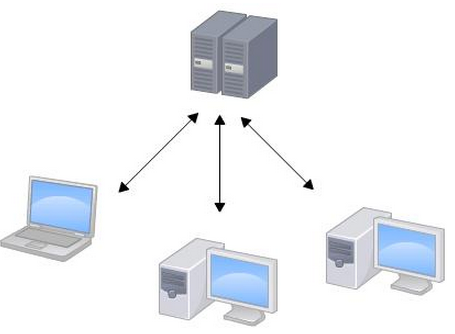
Linus的第二个伟大作品。2005年由于BitKeeper软件公司对Linux社区停止了免费使用权。Linus迫不得己自己开发了一个分布式版本控制工具，从而Git诞生了。

* 1. 哪些公司在使用

目前使用Git作为版本控制的开源软件：Linux kernel，Android, jQuery, Ruby on Rails，Debian…

1. Git和SVN的区别
   1. svn是集中式的版本控制系统

先说集中式版本控制系统，版本库是集中存放在中央服务器的，而干活的时候，用的都是自己的电脑，所以要先从中央服务器取得最新的版本，然后开始干活，干完活了，再把自己的活推送给中央服务器。中央服务器就好比是一个图书馆，你要改一本书，必须先从图书馆借出来，然后回到家自己改，改完了，再放回图书馆。

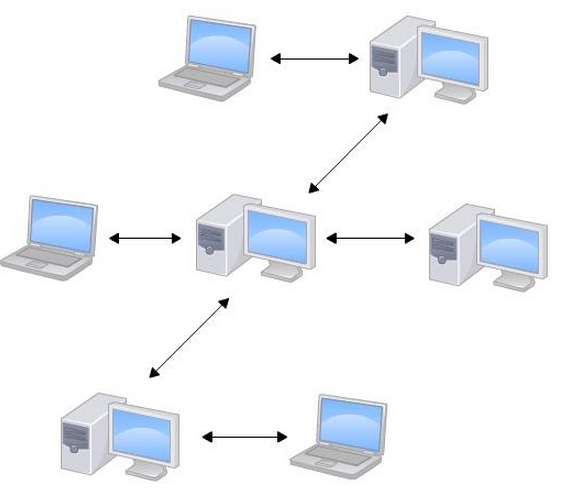


集中式版本控制系统最大的毛病就是必须联网才能工作，如果在局域网内还好，带宽够大，速度够快，可如果在互联网上，遇到网速慢的话，可能提交一个10M的文件就需要5分钟，这还不得把人给憋死啊。

集中式的版本控制系统只有一个版本库(代码的仓库)

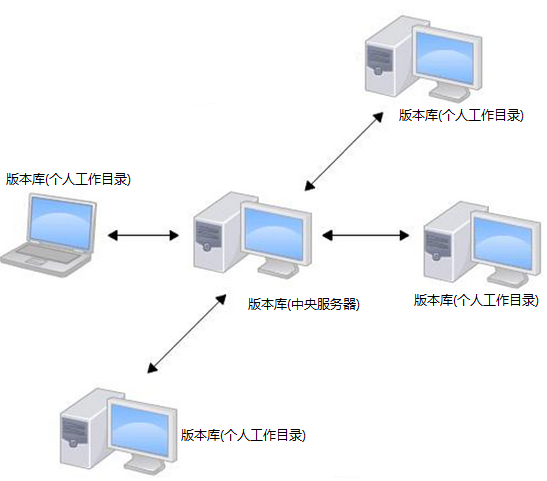
* 1. Git是分布式的版本控制系统

那分布式版本控制系统与集中式版本控制系统有何不同呢？首先，分布式版本控制系统根本没有“中央服务器”，每个人的电脑上都是一个完整的版本库，这样，你工作的时候，就不需要联网了，因为版本库就在你自己的电脑上。既然每个人电脑上都有一个完整的版本库，那多个人如何协作呢？比方说你在自己电脑上改了文件A，你的同事也在他的电脑上改了文件A，这时，你们俩之间只需把各自的修改推送给对方，就可以互相看到对方的修改了。



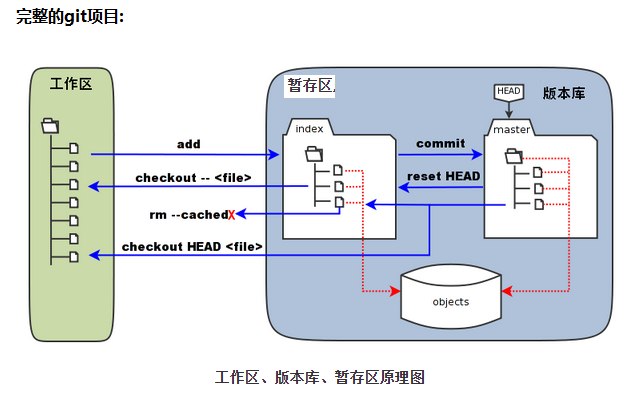
和集中式版本控制系统相比，分布式版本控制系统的安全性要高很多，因为每个人电脑里都有完整的版本库，某一个人的电脑坏掉了不要紧，随便从其他人那里复制一个就可以了。而集中式版本控制系统的中央服务器要是出了问题，所有人都没法干活了。

在实际使用分布式版本控制系统的时候，其实很少在两人之间的电脑上推送版本库的修改，因为可能你们俩不在一个局域网内，两台电脑互相访问不了，也可能今天你的同事病了，他的电脑压根没有开机。因此，分布式版本控制系统通常也有一台充当“中央服务器”的电脑，但这个服务器的作用仅仅是用来方便“交换”大家的修改，没有它大家也一样干活，只是交换修改不方便而已。



个人工作目录(本地项目代码)就是一个版本库.

1. Git版本库结构

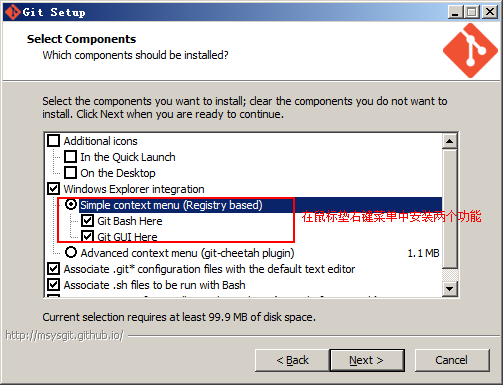


1. Git安装

git是不用单独安装服务器. 只需要安装git软件.



安装开发软件都使用管理员运行.



1. Git命令
   1. 创建仓库步骤:(只需要一次)

应用场景:

项目立项时需要一个git项目作为中心服务器项目供所有的人同步代码使用.需要创建一个新的项目

执行次数: 项目创建时只需要一次.

语法:

* + - 1. 新建项目文件夹
      2. 使用git init --bare命令初始化

--bare说明如果该项目仓库只作为中央服务器项目,不允许使用git直接操作该项目最保存最新的代码让其他人同步



* 1. 克隆项目

应用场景:

新的项目成员需要参与开发该项目, 需要从中心服务器项目从克隆出一份最新的仓库到开发电脑上.

执行次数: 新参与开发成员只需要一次

语法:

git clone 中心服务器项目仓库路径 目标仓库路径



注意:

克隆出来的仓库目录也包含了我们的工作开发目录和本地仓库

* 1. 在项目修改或者添加文件

应用场景:

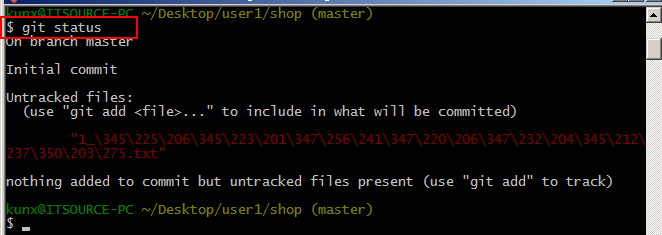
直接在克隆出来的项目中, 开发人员可以添加或者修改代码文件.

执行次数: 多次

* 1. 查看当前状态

应用场景:

想知道当前开发人员的本地仓库.



* 1. 将变动添加到暂存区中

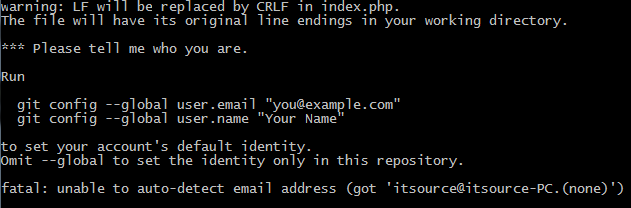
应用场景:

* 1. 暂存区中变动添加到本地版本库中

应用场景:

* 1. 配置当前仓库的身份信息

应用场景:



* 1. 将本地仓库推送到远程

应用场景:

推送语法:

git push origin master

将当前分支推送到主分支上

* 1. 将远程仓库中的改变拉取到本地中

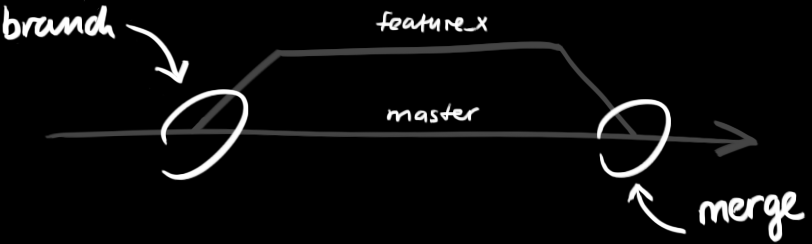
应用场景:

拉取(获取)语法:

git pull

1. 分支(branch)
   1. 什么是分支

分支是用来将特性开发绝缘开来的。在你创建仓库的时候，master 是“默认的”。在其他分支上进行开发，完成后再将它们合并到主分支上。



* 1. 查看分支列表

git branch

前面的\* 代表当前分支

* 1. 创建分支

语法:

git branch 新的分支名称

* 1. 切换分支

语法:

git checkout 分支名

git checkout -b 新分支名 (创建并切换分支)

* 1. 将修改推送分支上

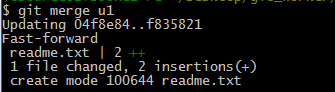
git push origin 分支

* 1. 将远程分支拉取到工作区中

git pull origin u2 拉取远程u2分支代码到工作区

如果发生和本地冲突，就解决冲突

如果想要手动合并多个分支



将本地分支1, 分支2 合并到 当前工作区中.

git merge 分支1 分支2 ....

将分支1 分支2 与当前工作区内容进行合并

如何使用Git上传项目代码到github 此博文包含图片 (2013-07-11 21:18:19)转载▼

标签： it 分类： 计算机技术

这是我第一次应用git，以下仅供git的初学者参考。

github是一个基于git的代码托管平台，付费用户可以建私人仓库，我们一般的免费用户只能使用公共仓库，也就是代码要公开。这对于一般人来说公共仓库就已经足够了。

1.注册账户以及创建仓库

要想使用github第一步当然是注册github账号了。之后就可以创建仓库了（免费用户只能建公共仓库），Create a New Repository，填好名称后Create，之后会出现一些仓库的配置信息，这也是一个git的简单教程。

2.安装客户端tortoiseGit

github是服务端，要想在自己电脑上使用git我们还需要一个git客户端，我这里选用TortoiseGit，他给我们提供了图形界面的操作。在安装之前首先需要安装git，下载地址http://msysgit.github.com/，TortoiseGit下载地址：

http://code.google.com/p/tortoisegit/

装完后右键鼠标会多出一些选项来，在本地仓库里右键选择Git Init Here，会多出来一个.git文件夹，这就表示本地git创建成功。右键Git Bash进入git命令行，为了把本地的仓库传到github，还需要配置ssh key。

3.配置Git

（1） 首先在本地创建ssh key；

$ ssh-keygen -t rsa -C "your\_email@youremail.com"

后面的your\_email@youremail.com改为你的邮箱，之后会要求确认路径和输入密码，我们这使用默认的一路回车就行。成功的话会在~/下生成.ssh文件夹，进去，打开id\_rsa.pub，复制里面的key。回到github，进入Account Settings，左边选择SSH Keys，Add SSH Key,title随便填，粘贴key。

（2）为了验证是否成功，在git bash下输入：

$ ssh -T git@github.com

如果是第一次的会提示是否continue，输入yes就会看到：You’ve successfully authenticated, but GitHub does not provide shell access 。这就表示已成功连上github。

（3）接下来我们要做的就是把本地仓库传到github上去，在此之前还需要设置username和email，因为github每次commit都会记录他们。

$ git config --global user.name "your name"

$ git config --global user.name "your name"$ git config --global user.email "your\_email@youremail.com"

（4）进入要上传的仓库，右键git bash，添加远程地址：

$ git remote add origin git@github.com:yourName/yourRepo.git

后面的yourName和yourRepo表示你再github的用户名和刚才新建的仓库，加完之后进入.git，打开config，这里会多出一个remote “origin”内容，这就是刚才添加的远程地址，也可以直接修改config来配置远程地址。

4.提交、上传

（1）接下来在本地仓库里添加一些文件，比如README，

$ git add README

$ git add README$ git commit -m "first commit"

（2）上传到github：

$ git push origin master

git push命令会将本地仓库推送到远程服务器。

git pull命令则相反。

修改完代码后，使用git status可以查看文件的差别，使用git add 添加要commit的文件，也可以用git add -i来智能添加文件。之后git commit提交本次修改，git push上传到github。

5.gitignore文件

.gitignore顾名思义就是告诉git需要忽略的文件，这是一个很重要并且很实用的文件。一般我们写完代码后会执行编译、调试等操作，这期间会产生很多中间文件和可执行文件，这些都不是代码文件，是不需要git来管理的。我们在git status的时候会看到很多这样的文件，如果用git add -A来添加的话会把他们都加进去，而手动一个个添加的话也太麻烦了。这时我们就需要.gitignore了。比如一般c#的项目我的.gitignore是这样写的：

bin

.suo

obj

bin和obj是编译目录，里面都不是源代码，忽略；suo文件是vs2010的配置文件，不需要。这样你在git status的时候就只会看到源代码文件了，就可以放心的git add -A了。