Git分布式版本控制工具

# Git概述

## Git历史

Git诞生于一个计算机技术高速发展的年代。Linux内核开源项目有着为数众多的参与者。绝大多数的Linux内核维护工作都花在了提交补丁和保存归档的繁琐事务上（1991－2002年间）。到2002年，整个项目组开始启用一个专有的分布式版本控制系统BitKeeper来管理和维护代码。

到了2005年，开发BitKeeper的商业公司同Linux内核开源社区的合作关系结束，他们收回了Linux内核社区免费使用BitKeeper的权力。这就迫使Linux开源社区（特别是Linux的缔造者LinusTorvalds）基于使用BitKeeper时的经验教训，开发出自己的版本系统。

他们对新的系统制订了若干目标：

* 速度
* 简单的设计
* 对非线性开发模式的强力支持（允许成千上万个并行开发的分支）
* 完全分布式
* 有能力高效管理类似Linux内核一样的超大规模项目（速度和数据量）

## Git与SVN对比

SVN是集中式版本控制系统，版本库是集中放在中央服务器的，而开发人员工作的时候，用的都是自己的电脑，所以首先要从中央服务器下载最新的版本，然后开发，开发完后，需要把自己开发的代码提交到中央服务器。

集中式版本控制工具缺点：

* 服务器单点故障
* 容错性差

Git是分布式版本控制系统（Distributed Version Control System，简称DVCS），分为两种类型的仓库：本地仓库和远程仓库。

* 本地仓库：是在开发人员自己电脑上的Git仓库
* 远程仓库：是在远程服务器上的Git仓库

## Git基本术语

Clone：克隆，就是将远程仓库复制到本地

Push：推送，就是将本地仓库代码上传到远程仓库

Pull：拉取，就是将远程仓库代码下载到本地仓库

## Git工作流程

工作流程如下：

1．从远程仓库中克隆代码到本地仓库

2．从本地仓库中checkout代码然后进行代码修改

3．在提交前先将代码提交到暂存区

4．提交到本地仓库。本地仓库中保存修改的各个历史版本

5．修改完成后，需要和团队成员共享代码时，将代码push到远程仓库

## Git下载与安装

下载地址： https://git-scm.com/download

# Git代码托管服务

## 常用的Git代码托管服务

前面我们已经知道了Git中存在两种类型的仓库，即本地仓库和远程仓库。那么我们如何搭建Git远程仓库呢？我们可以借助互联网上提供的一些代码托管服务来实现，其中比较常用的有GitHub、码云、GitLab等。

gitHub（地址：https://github.com/）是一个面向开源及私有软件项目的托管平台，因为只支持Git作为唯一的版本库格式进行托管，故名gitHub；

码云（地址：https://gitee.com/）是国内的一个代码托管平台，由于服务器在国内，所以相比于GitHub，码云速度会更快；

GitLab（地址：https://about.gitlab.com/）是一个用于仓库管理系统的开源项目，使用Git作为代码管理工具，并在此基础上搭建起来的web服务。

# Git常用命令

## 环境配置

当安装Git后首先要做的事情是设置用户名称和email地址。这是非常重要的，因为每次Git提交都会使用该用户信息

设置用户信息

* Git config –global user.name “\*”
* Git config –global user.email “\*”

查看配置信息

* Git config --list
* Git config user.name

通过上面的命令设置的信息会保存在~/.gitconfig文件中

## 获取Git仓库

要使用Git对我们的代码进行版本控制，首先需要获得Git仓库。获取Git仓库通常有两种方式：

1. 在本地初始化一个Git仓库
2. 从远程仓库克隆

## 在本地初始化一个Git仓库

执行步骤如下：

1.在电脑的任意位置创建一个空目录（例如repo1）作为我们的本地Git仓库

2.进入这个目录中，点击右键打开Git bash窗口

3.执行命令git init

如果在当前目录中看到.git文件夹（此文件夹为隐藏文件夹）则说明Git仓库创建成功。

## 从远程仓库克隆

可以通过Git提供的命令从远程仓库进行克隆，将远程仓库克隆到本地。命令形式为：git clone远程Git仓库地址

## 工作目录、暂存区以及版本库概念

为了更好的学习Git，我们需要了解Git相关的一些概念，这些概念在后面的学习中会经常提到

**版本库：**前面看到的.git隐藏文件夹就是版本库，版本库中存储了很多配置信息、日志信息和文件版本信息等

**工作目录（工作区）：**包含.git文件夹的目录就是工作目录，主要用于存放开发的代码

**暂存区：**.git文件夹中有很多文件，其中有一个index文件就是暂存区，也可以叫做stage。暂存区是一个临时保存修改文件的地方

## Git工作目录下文件的两种状态

Git工作目录下的文件存在两种状态：

1. untracked未跟踪（未被纳入版本控制）
2. tracked已跟踪（被纳入版本控制）
   * ​Unmodified未修改状态
   * ​Modified已修改状态
   * ​Staged已暂存状态

这些文件的状态会随着我们执行Git的命令发生变化

## 本地仓库操作

Git status查看文件状态

也可以使用git status–s使输出信息更加简洁

Git add将未跟踪的文件加入暂存区

Git reset将暂存区的文件取消暂存

Git commit将暂存区的文件修改提交到本地仓库

Git rm删除文件

一般我们总会有些文件无需纳入Git的管理，也不希望它们总出现在未跟踪文件列表。通常都是些自动生成的文件，比如日志文件，或者编译过程中创建的临时文件等。在这种情况下，我们可以在工作目录中创建一个名为.git ignore的文件（文件名称固定），列出要忽略的文件模式。下面是一个示例：

# no .a files

\*.a

# but do track lib.a, even though you're ignoring .a files above

!lib.a

# only ignore the TODO file in the current directory, not subdir/TODO

/TODO

# ignore all files in the build/ directory

build/

# ignore doc/notes.txt, but not doc/server/arch.txt

doc/\*.txt

# ignore all .pdf files in the doc/ directory

doc/\*\*/\*.pdf

然后可以用git log查看日志记录。

## 远程仓库操作

前面执行的命令操作都是针对的本地仓库，本章节我们会学习关于远程仓库的一些操作，具体包括：

### 查看远程仓库

如果想查看已经配置的远程仓库服务器，可以运行 git remote 命令。 它会列出指定的每一个远程服务器的简写。 如果已经克隆了远程仓库，那么至少应该能看到orgin（master），这是 Git 克隆的仓库服务器的默认名字。

### 添加远程仓库

运行git remote add <shortname> <url>添加一个新的远程Git仓库，同时指定一个可以引用的简写

### 从远程仓库克隆

如果你想获得一份已经存在了的Git仓库的拷贝，这时就要用到git clone命令。Git克隆的是该Git仓库服务器上的几乎所有数据（包括日志信息、历史记录等），而不仅仅是复制工作所需要的文件。当你执行git clone命令的时候，默认配置下远程Git仓库中的每一个文件的每一个版本都将被拉取下来。

克隆仓库的命令格式是git clone [url]

### 移除无效的远程仓库

如果因为一些原因想要移除一个远程仓库，可以使用git remoterm

注意：此命令只是从本地移除远程仓库的记录，并不会真正影响到远程仓库

### 从远程仓库中抓取与拉取

Git fetch是从远程仓库获取最新版本到本地仓库，不会自动merge

Git pull是从远程仓库获取最新版本并merge到本地仓库

注意：如果当前本地仓库不是从远程仓库克隆，而是本地创建的仓库，并且仓库中存在文件，此时再从远程仓库拉取文件的时候会报错，解决此问题可以在git pull命令后加入参数--allow-unrelated-histories

### 推送到远程仓库

当你想分享你的代码时，可以将其推送到远程仓库。

命令形式：git push [remote-name] [branch-name]

## Git分支

几乎所有的版本控制系统都以某种形式支持分支。使用分支意味着你可以把你的工作从开发主线上分离开来，以免影响开发主线。Git的master分支并不是一个特殊分支。它跟其它分支没有区别。之所以几乎每一个仓库都有master分支，是因为git init命令默认创建它，并且大多数人都懒得去改动它。

### 查看分支

列出所有本地分支

$gitbranch

列出所有远程分支

$gitbranch-r

\#列出所有本地分支和远程分支

$gitbranch-a

### 创建分支，切换分支等……