Git使用指导书 V1.0



华为技术有限公司

2017年7月

修订记录Revision record

|  |  |  |  |
| --- | --- | --- | --- |
| 日期  Date | 修订版本Revision version | 修改描述  change Description | 作者  Author |
| 2017-07-26 | V1.0 |  | 蔡圣哲 |
| 2017-08-07 | V1.1 | 实际开发中使用的详细说明 | 蔡圣哲 |
|  |  |  |  |

**目 录**

[Git使用指导书 V1.0 1](#_Toc489883659)

[1 Git概述 4](#_Toc489883660)

[1.1 Git概述 4](#_Toc489883661)

[1.2 Git工作流程 5](#_Toc489883662)

[1.3 Git存储机制 5](#_Toc489883663)

[2 Git的安装与配置 6](#_Toc489883664)

[2.1 软件安装 6](#_Toc489883665)

[2.2 Git配置 6](#_Toc489883666)

[3 Git使用及常见命令 7](#_Toc489883667)

[3.1创建 7](#_Toc489883668)

[3.2 克隆仓库 7](#_Toc489883669)

[3.3 基本快照 7](#_Toc489883670)

[3.4 提交和重置 8](#_Toc489883671)

[3.5 查看历史 9](#_Toc489883672)

[4 分支和标签 10](#_Toc489883673)

[4.1 分支管理 10](#_Toc489883674)

[4.2 标签 10](#_Toc489883675)

[5 实际使用 11](#_Toc489883676)

# Git概述

## Git概述

分布式版本控制系统，用于敏捷高效地处理任何或小或大的项目。

**Git的功能特性：**

从一般开发者的角度来看，git有以下功能：

* 从服务器上克隆完整的Git仓库（包括代码和版本信息）到单机上。
* 在自己的机器上根据不同的开发目的，创建分支，修改代码。
* 在单机上自己创建的分支上提交代码。
* 在单机上合并分支。
* 把服务器上最新版的代码fetch下来，然后跟自己的主分支合并。
* 生成补丁（patch），把补丁发送给主开发者。
* 看主开发者的反馈，如果主开发者发现两个一般开发者之间有冲突（他们之间可以合作解决的冲突），就会要求他们先解决冲突，然后再由其中一个人提交。如果主开发者可以自己解决，或者没有冲突，就通过。
* 一般开发者之间解决冲突的方法，开发者之间可以使用pull 命令解决冲突，解决完冲突之后再向主开发者提交补丁。

从项目管理者的角度看，git有以下功能：

* 查看邮件或者通过其它方式查看一般开发者的提交状态。
* 打上补丁，解决冲突（可以自己解决，也可以要求开发者之间解决以后再重新提交，如果是开源项目，还要决定哪些补丁有用，哪些不用）。
* 向公共服务器提交结果，然后通知所有开发人员。

**Git 与 SVN 区别点：**

* GIT是分布式的，SVN不是：这是GIT和其它非分布式的版本控制系统，例如SVN，CVS等，最核心的区别。
* GIT把内容按元数据方式存储，而SVN是按文件：所有的资源控制系统都是把文件的元信息隐藏在一个类似.svn,.cvs等的文件夹里。
* GIT分支和SVN的分支不同：分支在SVN中一点不特别，就是版本库中的另外的一个目录。
* GIT没有一个全局的版本号，而SVN有：目前为止这是跟SVN相比GIT缺少的最大的一个特征。
* GIT的内容完整性要优于SVN：GIT的内容存储使用的是SHA-1哈希算法。这能确保代码内容的完整性，确保在遇到磁盘故障和网络问题时降低对版本库的破坏。

## Git工作流程

一般工作流程如下：

1.克隆 Git 资源作为工作目录。

2.在克隆的资源上添加或修改文件。

3.如果其他人修改了，你可以更新资源。

4.在提交前查看修改。

5.提交修改。

6.在修改完成后，如果发现错误，可以撤回提交并再次修改并提交。

## Git存储机制

**工作区：**就是你在电脑里能看到的目录。

**暂存区：**英文叫stage, 或index。一般存放在 ".git目录下" 下的index文件（.git/index）中，所以我们把暂存区有时也叫作索引（index）。

**版本库：**工作区有一个隐藏目录.git，这个不算工作区，而是Git的版本库。

# Git的安装与配置

## 软件安装

* + - 1. Windows平台用户到 GitHub 的页面上下载.exe安装文件并运行，访问<https://git-for-windows.github.io/>
      2. 在Windows平台，Git会找寻用户主目录下的.gitconfig文件。主目录即 $HOME 变量指定的目录，一般都是 C:\Documents and Settings\$USER。

## Git配置

用户信息

 $ git config --global user.name "cWX491729"

$ git config --global user.email caishengzhe@hauwei.com

**--global** ，以后你所有的项目都会默认使用这里配置的用户信息。

如果要在某个特定的项目中使用其他名字或者电邮，只要去掉 --global 选项重新配置即可。

文本编辑

$ git config --global core.editor emacs

一般使用Vi 或者 Vim

查看配置信息

$ git config --list

或：

vim ~/.gitconfig

$ git config user.name

# Git使用及常见命令

## 3.1创建

git init

初始化

git init 指定目录

在指定目录创建Git仓库

## 克隆仓库

$ git clone <仓库的repo> <指定目录>

克隆仓库，到指定目录，eg.：

$ git clone <https://github.com/Stevenchooo/Flink-test.git> flink-test

## 基本快照

$ git status -s

查看项目的当前状态

$ git diff

git diff 命令显示已写入缓存与已修改但尚未写入缓存的改动的区别

$ git commit -m '第二次版本更新'

提交，或者使用：

git commit –a

git push

推送到服务器

git pull

取回更新

## 提交和重置

$ git add AnalysKafkaData.scala

将文件纳入版本控制

$ git commit -m '提交时应添加版本信息'

提交到缓存，或

$ git commit -am '修改AnalysKafkaData.scala文件'

$ git reset HEAD -- AnalysKafkaData.scala

重置（取消文件的缓存）

$ git rm AnalysKafkaData.scala

从缓存和硬盘同时删除这个文件，如果想保留硬盘上的文件：

$ git rm --cached AnalysKafkaData.scala

$ git mv AnalysKafkaData.scala A.scala

重命名

## 查看历史

$ git log

列出历史提交记录，如果想查看版本简要信息：

$ git log --oneline

如果想查看分支、合并的逻辑图：

$ git log --oneline --graph

# 分支和标签

## 4.1 分支管理

$ git branch

列出分支

$ git branch testingOne

新建分支

$ git checkout testingTwo

切换分支

git merge

合并分支

git branch -d testingOne

删除分支

## 4.2 标签

$ git tag -a v1.0

在重要的里程碑节点做标记，使用如下命令可查看：

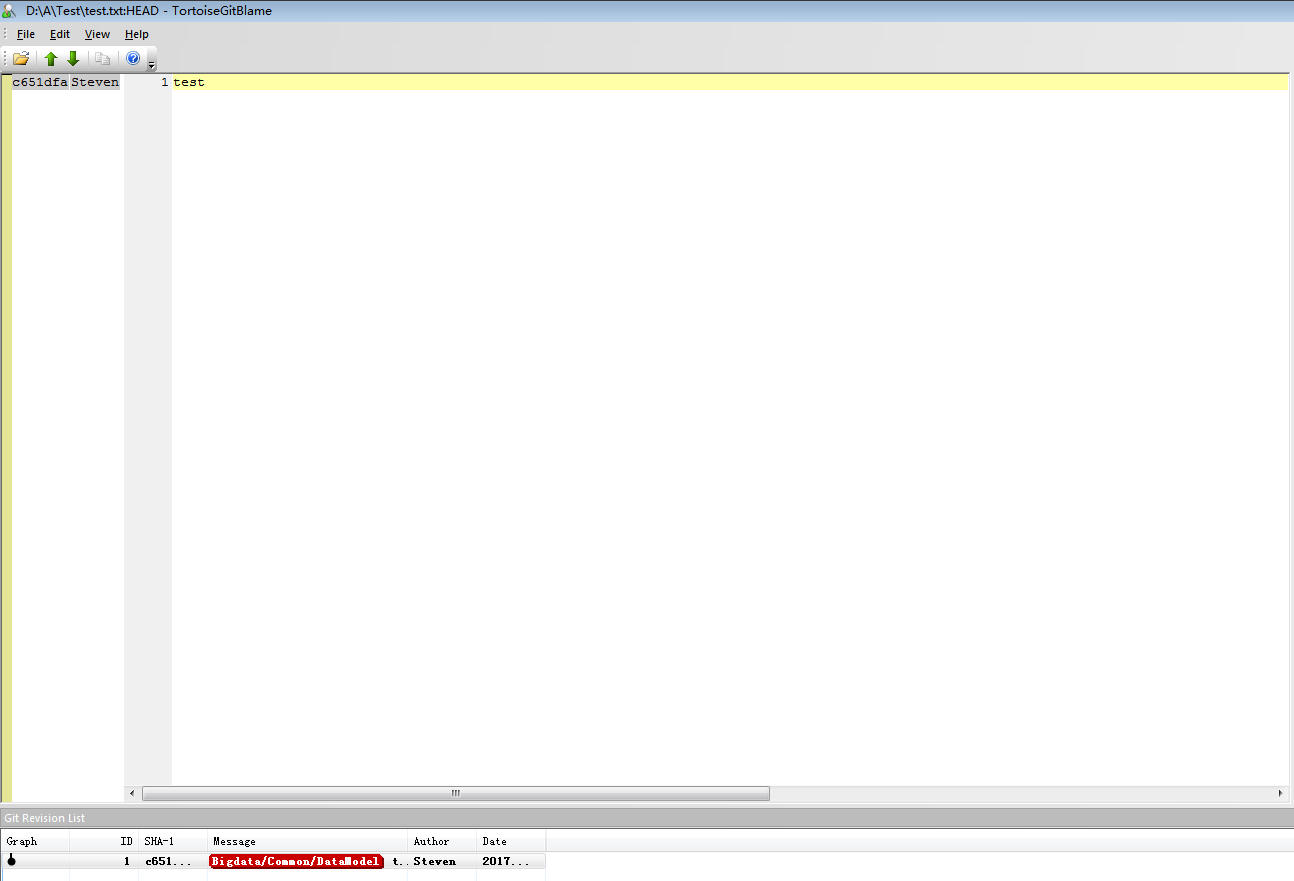
$ git log --oneline --decorate –graph

# 5实际使用

$ git clone http://zwx472924@dgggit01-ge.huawei.com/a/BigData/DataModel/Code -b Bigdata/Common/DataModel D:/Code

使用如上命令，下载代码

使用TortoiseGit，执行上传、提交、更新等操作。



右键点击需要上传的文件进行操作。

设置界面如下：

