**Git**

**参考文献**[**http://www.runoob.com/git/git-tutorial.html**](http://www.runoob.com/git/git-tutorial.html)

**<https://www.yiibai.com/git/git_init.html>**

# Git简介



官网地址：https://git-scm.com/

Git是一个开源的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 是 Linus Torvalds 为了帮助管理 Linux 内核开发而开发的一个开放源码的版本控制软件。

Git 与常用的版本控制工具 CVS, Subversion 等不同，它采用了分布式版本库的方式，不必服务器端软件支持。

**GIT优点：**

大部分操作在本地完成，不需要联网

完整性保证

尽可能添加数据而不是删除或修改数据

分支操作非常快捷流畅

与 Linux命令全面兼容

# 什么是版本控制

版本控制最主要的功能就是追踪文件的变更。它将什么时候、什么人更改了文件的什么内容等信息忠实地了已录下来。每一次文件的改变，文件的版本号都将增加。除了记录版本变更外，版本控制的另一个重要功能是并行开发。[软件开发](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%BC%80%E5%8F%91/3448966)往往是多人协同作业，版本控制可以有效地解决版本的同步以及不同开发者之间的开发通信问题，提高协同开发的效率。并行开发中最常见的不同版本软件的错误(Bug)修正问题也可以通过版本控制中分支与合并的方法有效地解决。

# 版本控制工具应该具备的功能

**协同修改**

多人并行不悖的修改服务器端的同一个文件。

**数据备份**

不仅保存目录和文件的当前状态，还能够保存每一个提交过的历史状态。

**版本管理**

在保存每一个版本的文件信息的时候要做到不保存重复数据，以节约存储空

间，提高运行效率。这方面 SVN采用的是增量式管理的方式，而 Git采取了文

件系统快照的方式。

**权限控制**

对团队中参与开发的人员进行权限控制。

对团队外开发者贡献的代码进行审核——Git独有。

**历史记录**

查看修改人、修改时间、修改内容、日志信息。

将本地文件恢复到某一个历史状态。

**分支管理**

允许开发团队在工作过程中多条生产线同时推进任务，进一步提高效

# 版本控制简介

## 版本控制

工程设计领域中使用版本控制管理工程蓝图的设计过程。在 IT开发过程中也可以

使用版本控制思想管理代码的版本迭代。

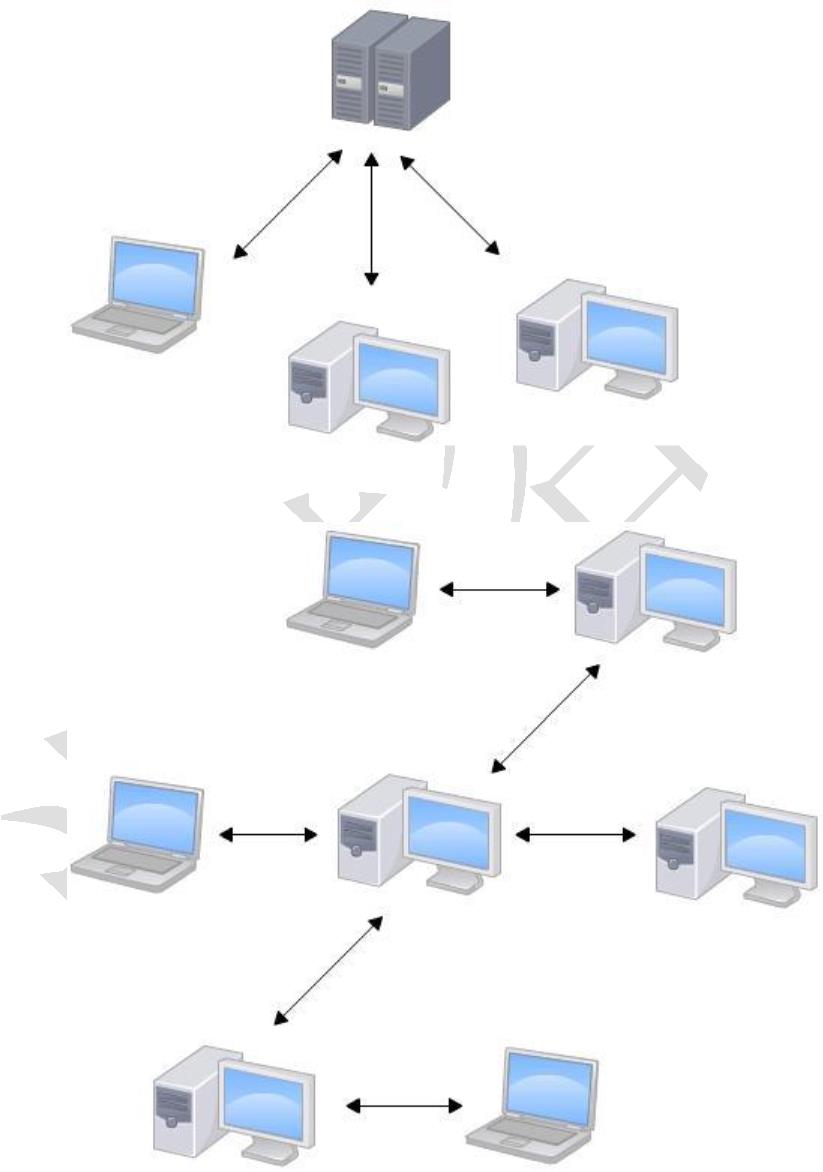
## 版本控制工具

思想：版本控制

实现：版本控制工具

**集中式版本控制工具**：

CVS、**SVN**、VSS……



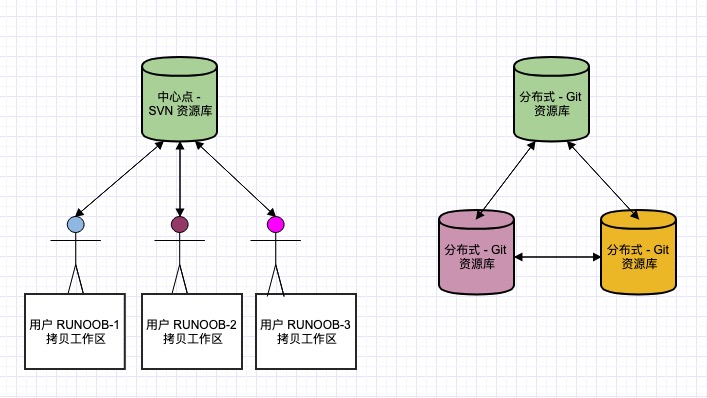
分布式版本控制工具：

**Git**、Mercurial、Bazaar、Darcs……

# Git 与 SVN 区别

**Git 与 SVN 区别点：**

* **1、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。
* **2、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。
* **3、Git 分支和 SVN 的分支不同：**分支在 SVN 中一点都不特别，其实它就是版本库中的另外一个目录。
* **4、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。
* **5、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。



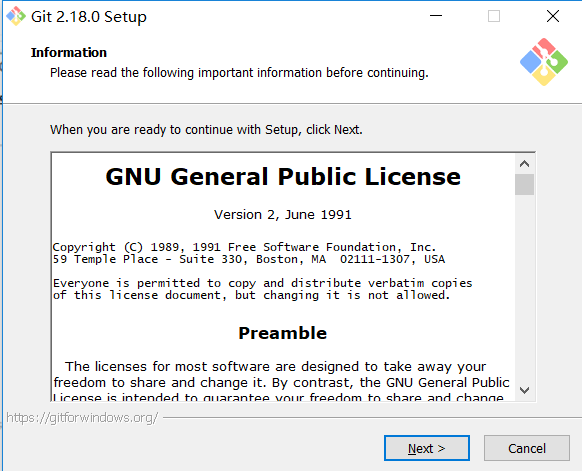
# Git安装

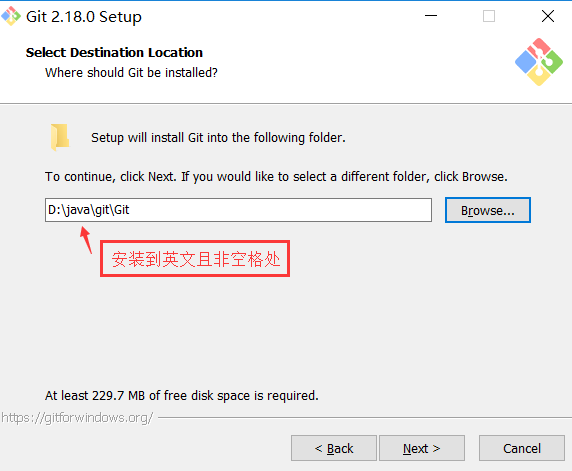
官网下载：

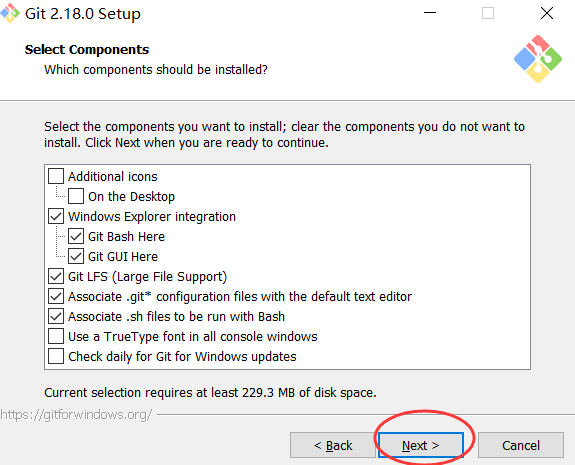
https://git-scm.com/downloads

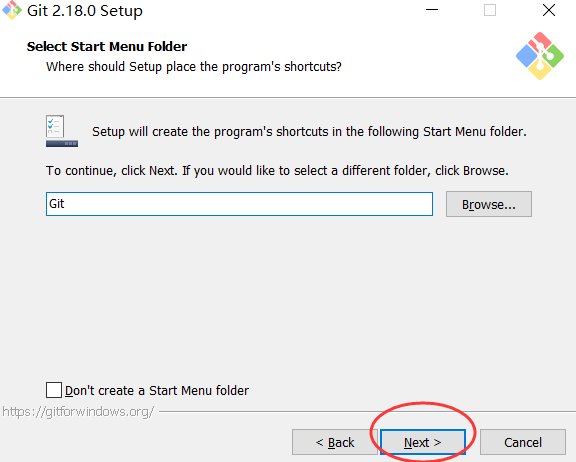
## Windows 平台上安装

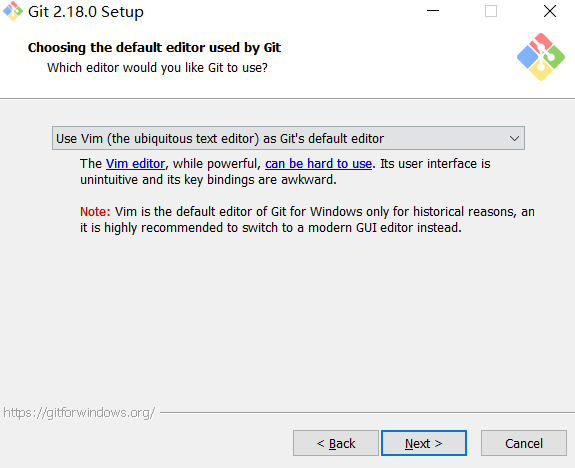




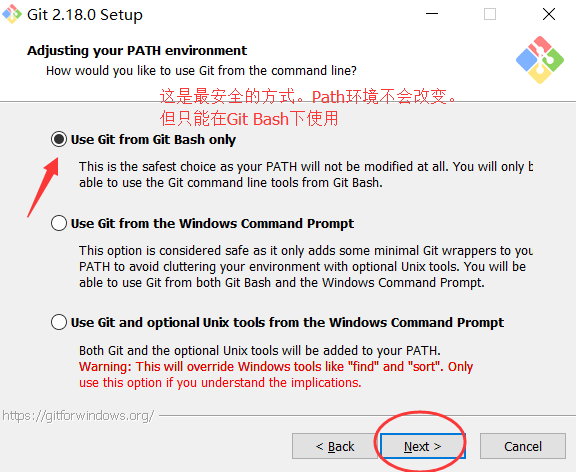


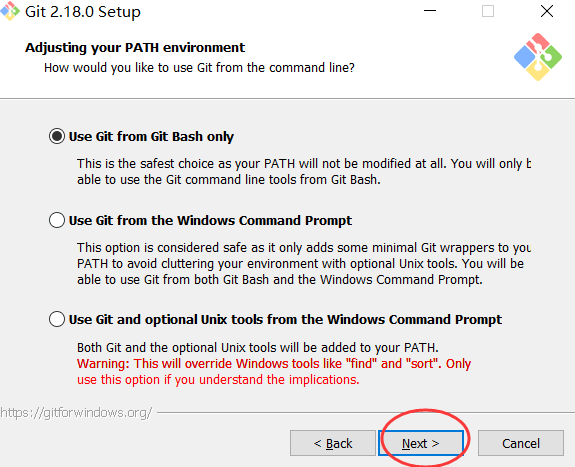


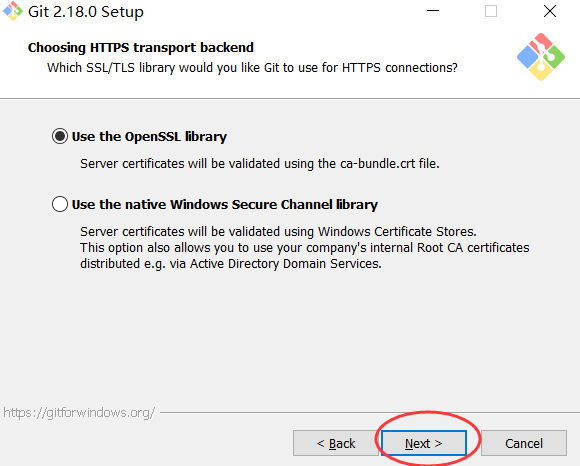


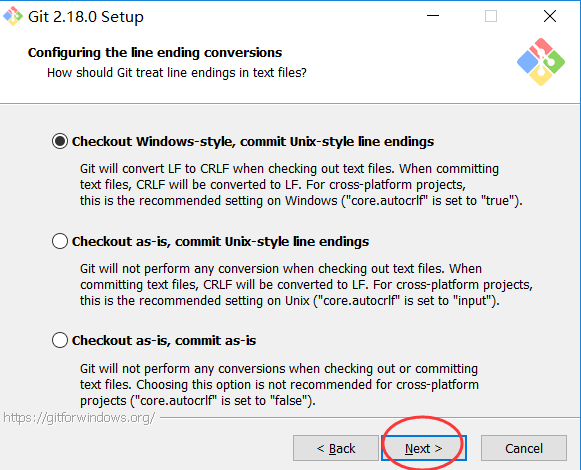


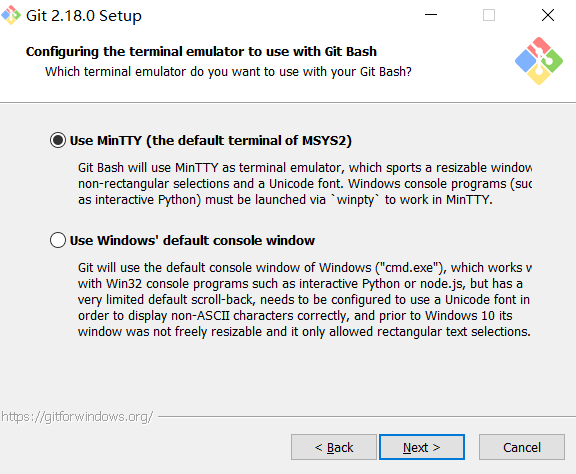
完全不修改Path环境变量。但只能在GitBash下使用

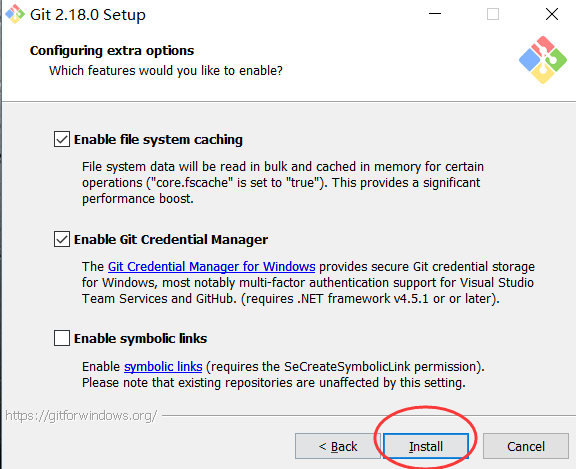


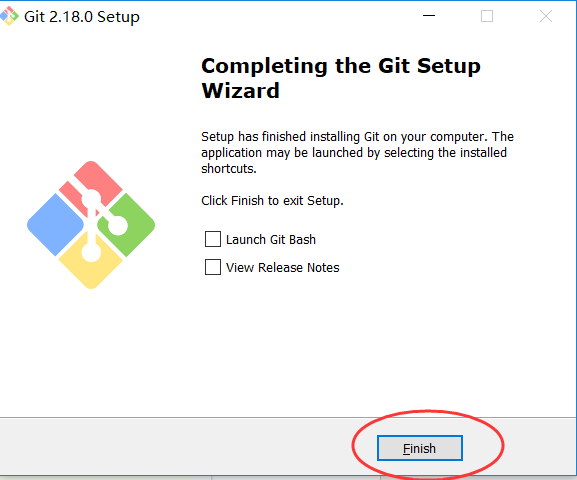


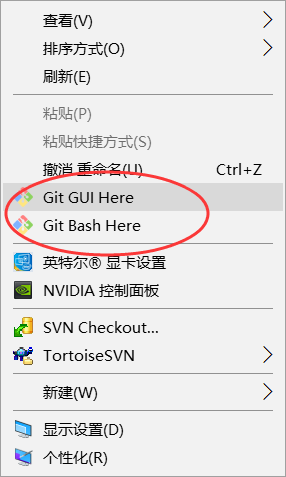












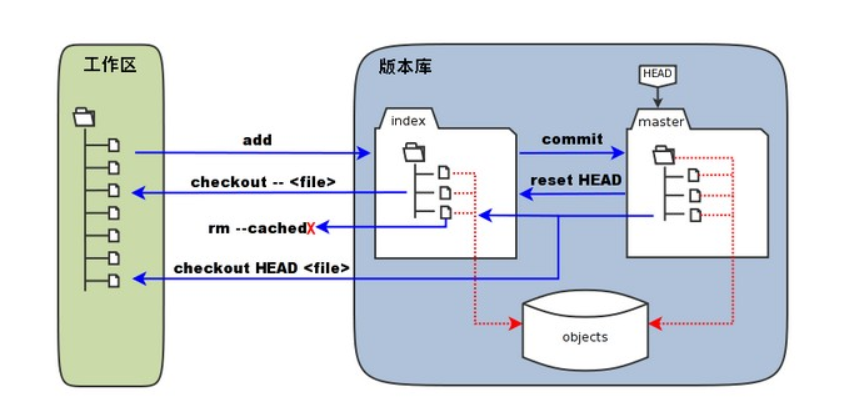
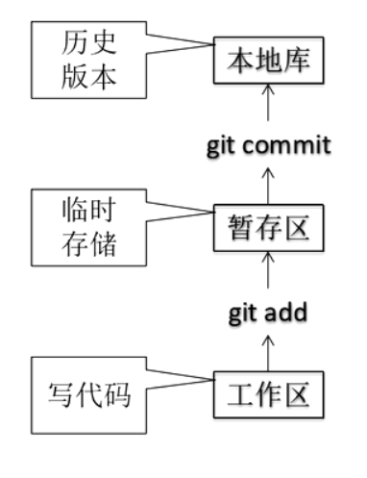
# Git结构

Git 工作区、暂存区和本地（版本）库

## 基本概念

我们先来理解下Git 工作区、暂存区和版本库概念

* **工作区：**就是你在电脑里能看到的目录(**写的代码存在工作区**）。
* **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）。（**计划要提交的文件**）
* **版本库：（本地库）**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和代码托管中心

## 代码托管中心

代码托管中心的任务：维护远程库

**局域网环境下：**

**GitLab服务器**

**外网环境下**

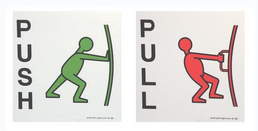
**GitHub https://github.com/**

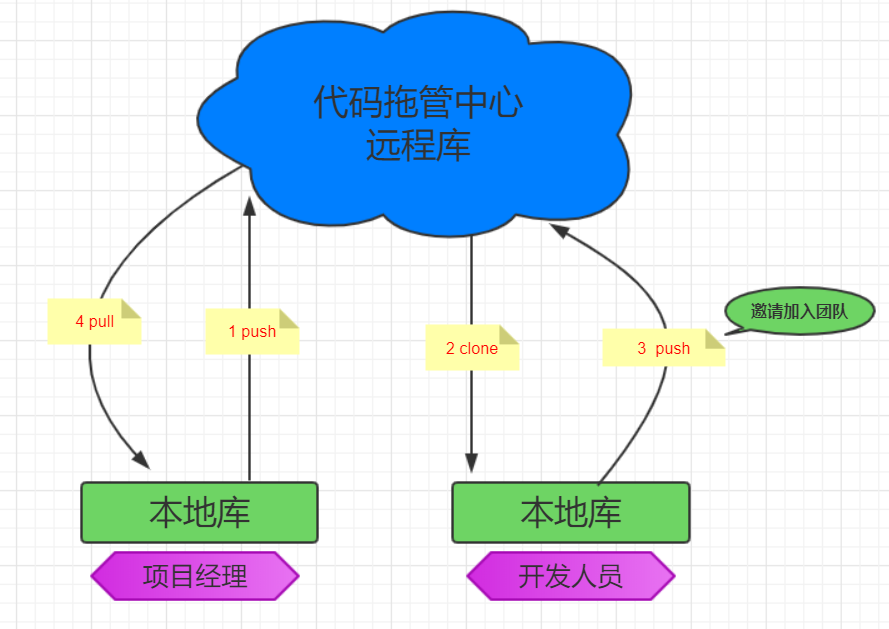
**码云 https://gitee.com/**

## 本地库和远程库

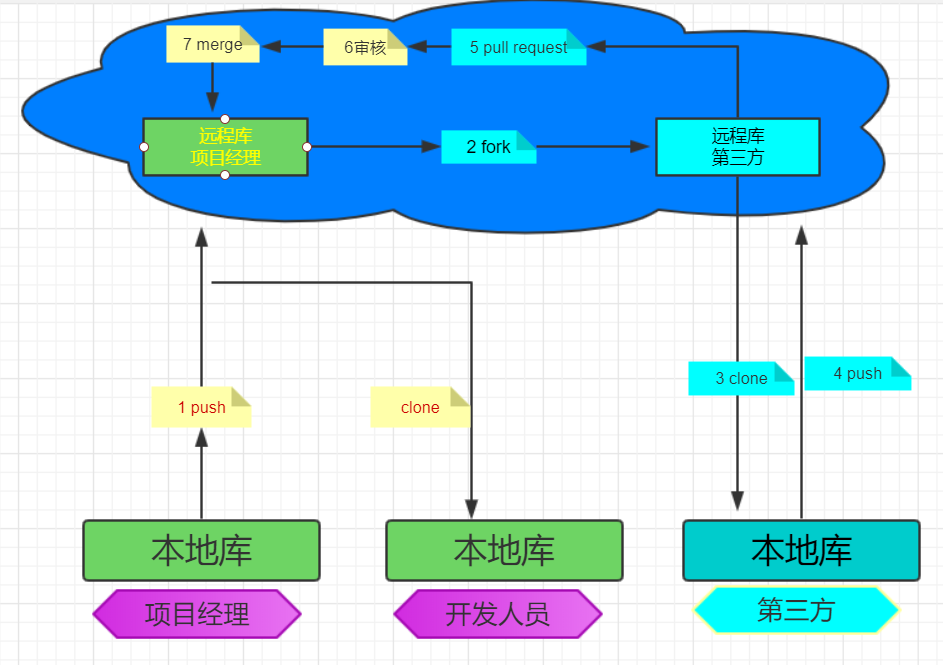
团队内部协作和跨团队协作

### 团队内部协作流程：





### 跨团队协作



# Git命令行



Git Bash命令同时支持Linux相关命令。

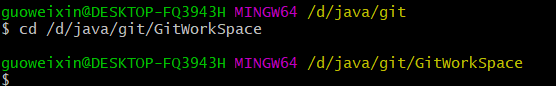
## 创建仓库（本地库）

### 初始化本地仓库

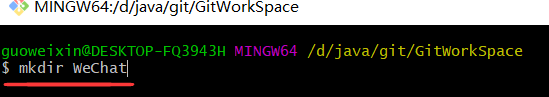
Git 使用 **git init** 命令来初始化一个 Git 仓库，Git 的很多命令都需要在 Git 的仓库中运行，所以 **git init** 是使用 Git 的第一个命令。

在执行完成 **git init** 命令后，Git 仓库会生成一个 .git 目录，该目录包含了资源的所有元数据，其他的项目目录保持不变（不像 SVN 会在每个子目录生成 .svn 目录，Git 只在仓库的根目录生成 .git 目录）。

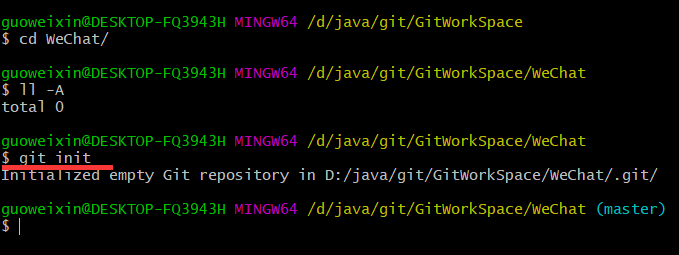
D:\java\git\GitWorkSpace 🡪选择GitWorkSpace当作我Git存放本地库目录



**创建一个项目目录： WeChat**



**使用当前目录作为Git仓库，我们只需使它初始化**

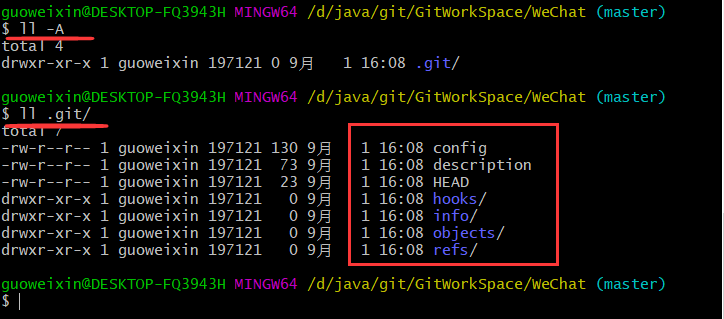


**初始化后，会在 WeChat 目录下会出现一个名为 .git 的目录，所有 Git 需要的数据和资源都存放在这个目录中。**

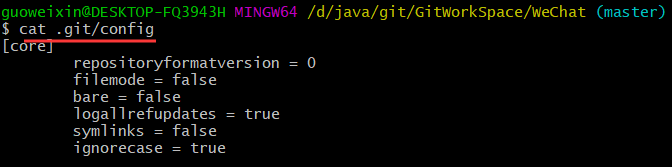
**查看隐藏的.git文件内容：**

**ll –A -a/A 表示查看隐藏文件**

**ll .git/ 表示查看.git 下的文件列表显示**



**例如：查看.git/config 内容**



**注意：.git目录是存放本地库相关的目录和文件，不要对其进行修改和操作。**

### 设置签名

#### 签名概述

**签名的作用：用于标识不同的开发人员（身份）**

**形式：**

用户名：weixin

Email邮箱：[315759265@qq.com](mailto:315759265@qq.com)

**注意**：这里的签名和远程库（代码拖管中心GitHub/GitLab）无任何关系。

**项目级别（仓库）**：仅在当前本地库范围有效

(此处的WeChat)

**系统级别：**登录当前操作系统的用户范围

开始—》窗口—》

**级别优先级：**

就近原则：项目级别优先级大于系统(用户)级别签名

**注意：**(签名级别必须要设置，如果都不设置，GIT可能会导致命令出差。无法辨别)

#### 签名命令

**git config**

**项目级别（仓库）**：仅在当前本地库范围有效

git config user.name weixin\_pro

git config user.email 315759265\_pro@qq.com

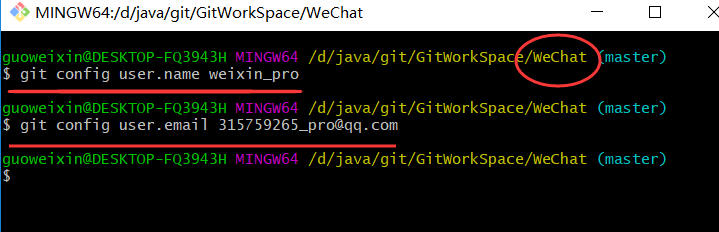
**系统级别：**登录当前操作系统的用户范围

git config **–global** user.name weixin\_global

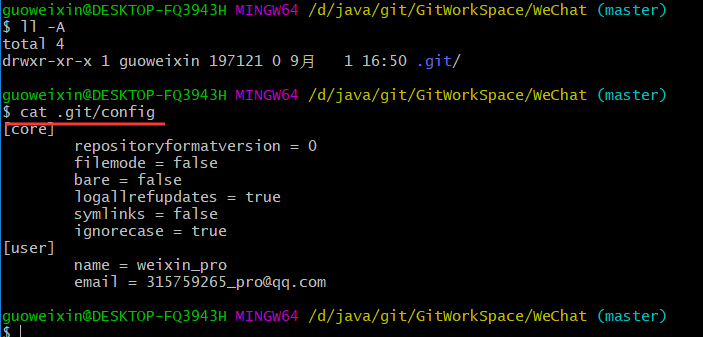
git config **–global** user.email [315759265\_global@qq.com](mailto:315759265_global@qq.com)

##### 项目级别（仓库）：仅在当前本地库范围有效

**命令：**

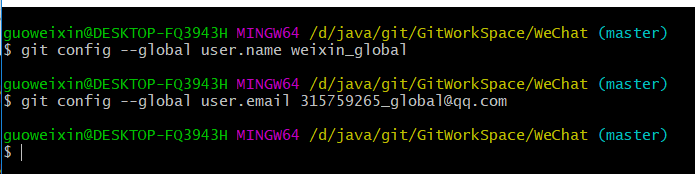


**查看设置结果：**



##### 系统级别：登录当前操作系统的用户范围

**命令：**

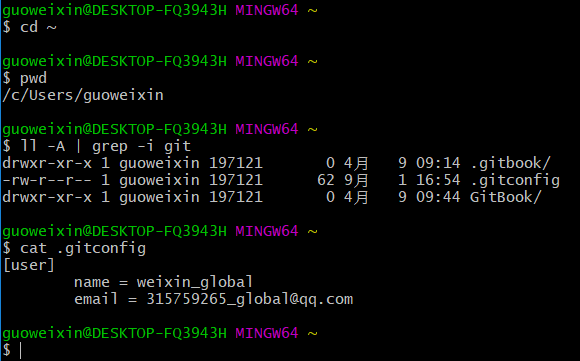


**查看（系统级别）需要到 （家目录查看）：**

**cd ~ 家目录**

**ll –A | grep –i git 查看叫git的文件名称**

**cat .gitconfig 查看文件内容**



**在实际开发过程中，通常只需要设置系统级别即可。如果有特殊需要，再设置项目级别。**

## 常用命令

1. **如何将一个文件在工作区提交到本地库？**
2. **修改呢？**

### Git status

**简介**

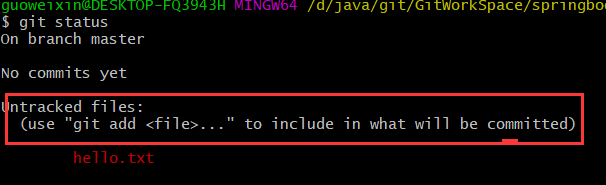
**git status** [<options>…​] [--] [<pathspec>…​]

git status**命令用于显示工作目录和暂存区的状态**。使用此命令能看到那些修改被暂存到了, 哪些没有, 哪些文件没有被Git tracked到。git status不显示已经commit到项目历史中去的信息。看项目历史的信息要使用git log



### Git add

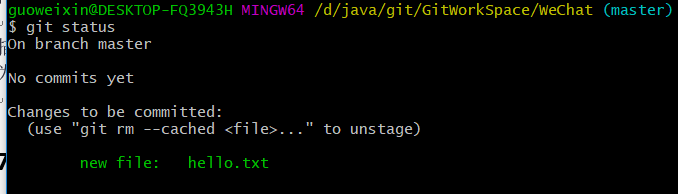
按如上提示，新建【VIM新建文件】，查看git status能看到信息提示：



git add hello.txt命令将文件内容添加到索引(将修改添加到暂存区)。也就是将要提交的文件的信息添加到索引库中。（将工作区写入暂存区）

描述：此命令将要提交的文件的信息添加到索引库中(将修改添加到暂存区)，以准备为下一次提交分段的内容。

**git add 后，再次查询git status**



### Git rm

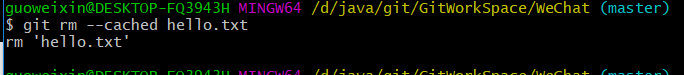
git rm命令用于从工作区和索引中删除文件。

**简介**

git rm [-f | --force] [-n] [-r] [--cached]

[--ignore-unmatch] [--quiet] [--] <file>…

**git rm –cached 文件名称 //用于将暂存区的文件恢复到工作区**



### Git commit

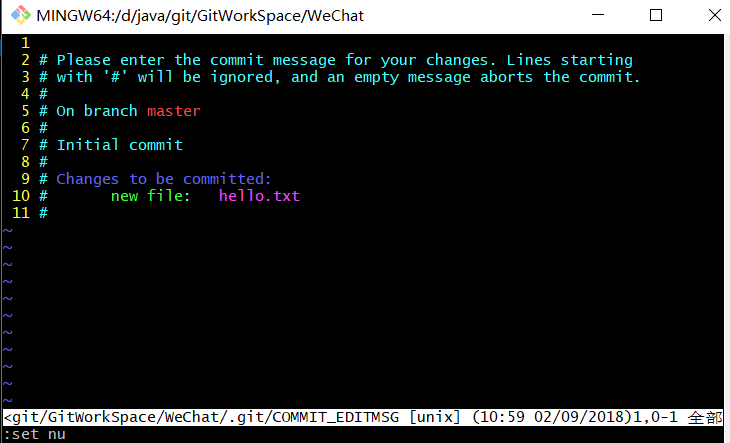
git commit命令用于将更改记录(提交)到存储库。将索引的当前内容与描述更改的用户和日志消息一起存储在新的提交中。

描述

git commit命令将索引的当前内容与描述更改的用户和日志消息一起存储在新的提交中。

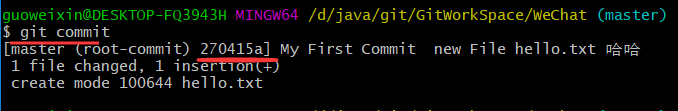
执行命令：git commit 文件名.txt后，显示如下需要输入提交文件的备注或描述。

**可通过:set nu 显示行号方式便利输入备注信息**

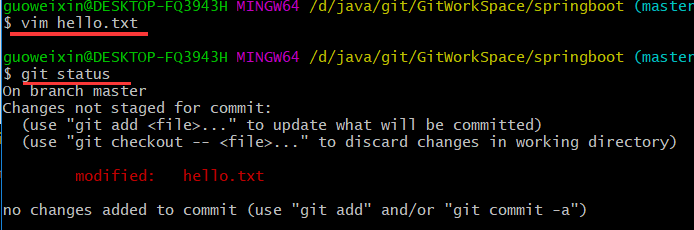


**Vim 加入备注、保存并退出后即可。**

**270415a 相当于svn中的文件版本号**

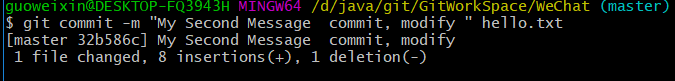


**案例：对hello.txt文件进行修改**



**可能过：**

**git commit -m “备注信息”文件名 //快速加入备注描述**



### Git checkout

git checkout命令用于切换分支或恢复工作树文件。git checkout是git最常用的命令之一，同时也是一个很危险的命令，因为这条命令会重写工作区。

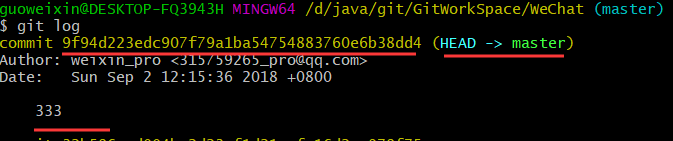
## 常用命令

### Git log

git log命令用于显示提交日志信息。

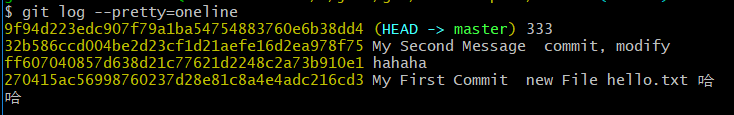
**使用语法**

git log [<options>] [<revision range>] [[\--] <path>…]

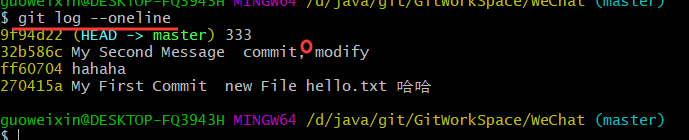


**通过如下方式更优雅的显示(查看历史记录)：**

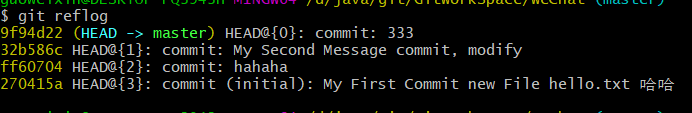
**git log –-pretty=oneline**



**Git log --oneline**



**git reflog //用于显示需要回退的步骤HEAD@{ \* }**



### Git reflog/reset查看(设置)历史记录(之前/之后)：

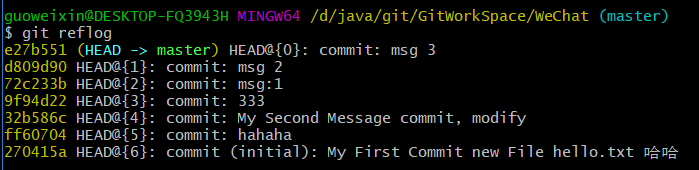
git reset命令用于将当前HEAD复位到指定状态。一般用于撤消之前的一些操作(如：git add,git commit等)。

**HEAD@{移动到当前版本需要多少步}**

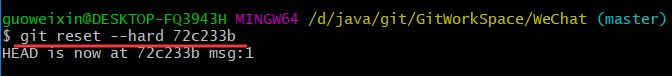
**基于索引值操作[推荐]**

**git reset --hard [局部索引值]**

查看当前版本记录：



通过**git reset --hard [局部索引值]跳到指定版本**



### 永久删除后找回

git rm命令用于从工作区和索引中删除文件。

1. 工作区创建文件，通过命令提交到本地库

**Vim d1.txt git add d1.txt git commit –m ‘上传到本地库d1.txt’ d1.txt**

1. 执行命令进行删除：

**git rm d1.txt** 🡪工作区删除后，提交到本地库**git commit –m ‘删除d1.txt’ d1.txt**

1. 通过日志git reflog 找回

**$ git reset --hard 15d5002**

1. **ll**查看即可。

**注意：git只要在本地库发生操作，都会进行版本记录。**

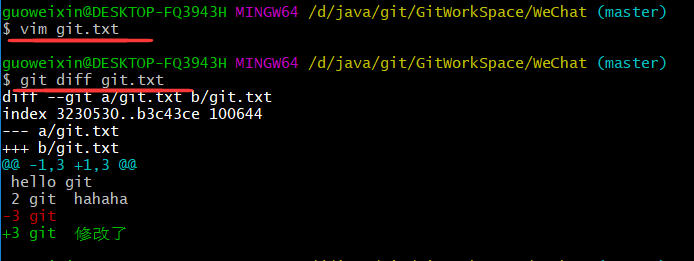
### 文件比较

git diff命令用于显示提交和工作树等之间的更改。此命令比较的是工作目录中当前文件和暂存区域快照之间的差异,也就是修改之后还没有暂存起来的变化内容。**(默认是和暂存区文件进行比较)**

**1、示例：gid diff 将工作区中的文件和暂存区进行比较**

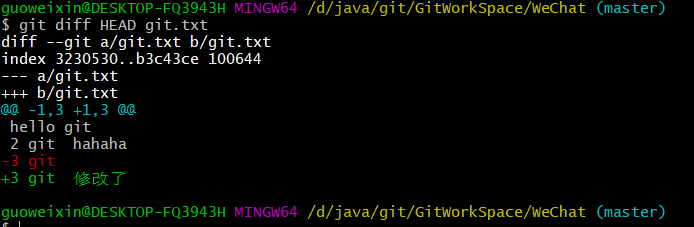
vim git.txt 修改内容

git diff git.txt



* 1. **示例：gid diff 暂存区中的文件和本地库进行比较**

1. **将git.txt 工作区添加到暂存区 git add git.txt**
2. **比较：git diff HEAD git.txt //HEAD代表本地库最新版本指针**



3**、所有文件比较**

git diff //不加文件名

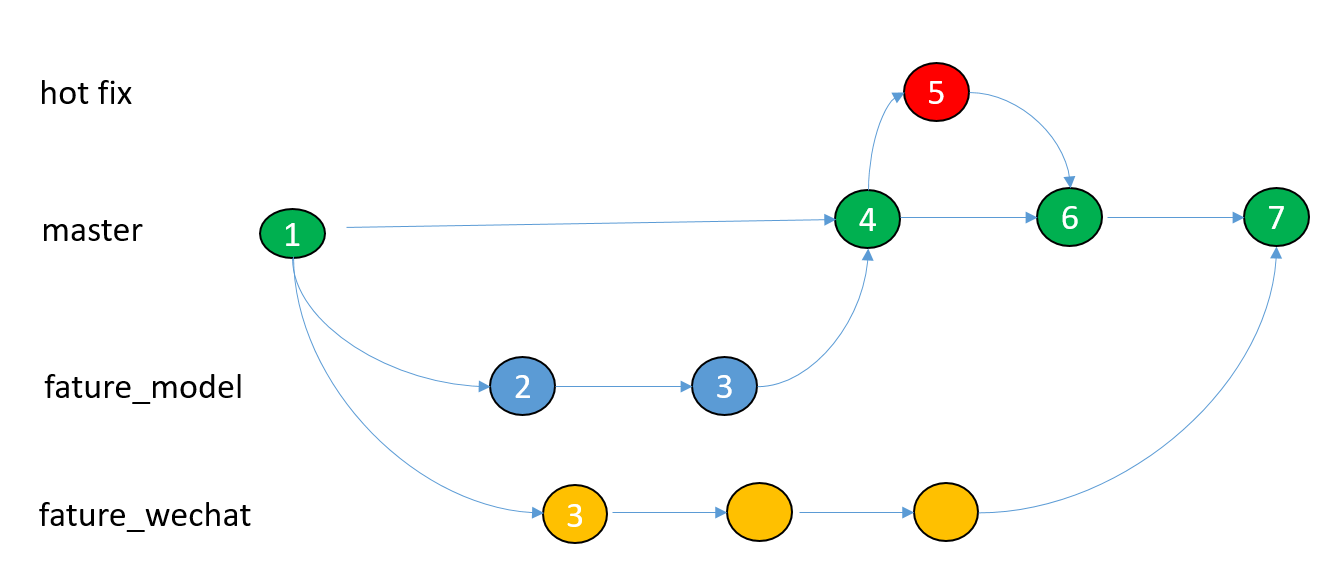
退回修改：git checkout 文件名

# 分支

## 分支概念

**什么是分支？**

在版本控制过程中，使用多条线同时推进多个任务。



**分支的好处？**

同时并行推进多个功能开发，提高开发效率

各个分支在开发过程中，如果某一个分支开发失败，不会对其他分支有任

何影响。

失败的分支删除重新开始即可

## 分支命令（操作）

### Git branch

git branch命令用于列出，创建或删除分支。

git checkout命令用于切换分支或恢复工作树文件。

**1查看当前有哪些分支**

git branch

1. **新建一个分支**

git branch hot\_fix

**3. 切换到指定分支**

git checkout hot\_fix

**4. 查看本地和远程分支**

git branch –a

**示例：合并分支：**

1、在分支hot\_fix 上修改git.txt文件，并提交到本地库

2、需要合并分支。让master上的git.txt同步

**合并分支**：

1、切换到接受修改的分支上master（将hot\_fix内容同步到master）

git checkout master //切换到接受修改的分支上

2、执行merge命令

git merge命令用于将两个或两个以上的开发历史加入(合并)一起。

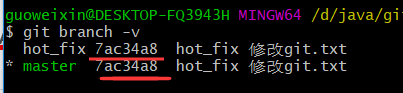
git merge hot\_fix// hot\_fix 被合并的分支名称

## 合并分支（冲突）解决

**问题：将两个分支内的文件分别进行修改.（且改变的是同一位置）进行合并。**

**1、查看分支信息：**

git branch -v

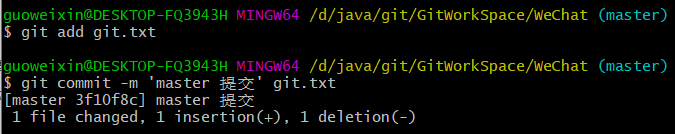


**2、在master下修改某行文件内容。然后进行提交。**

**vim git.txt**

**git add git.txt**

**git commit –m ‘master 提交’ git.txt**



3、查看分支信息

**git branch -v**



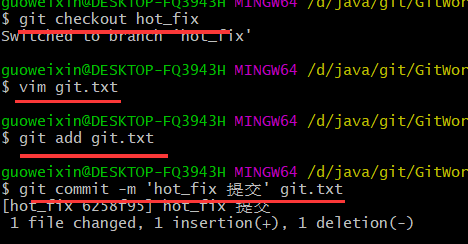
**4、在hot\_fix下修改某行（master）同行文件内容。然后进行提交**

**git checkout hot\_fix**

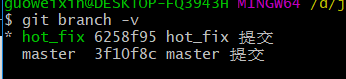
**vim git.txt**

**git add git.txt**

**git commit –m ‘hot\_fix提交’ git.txt**

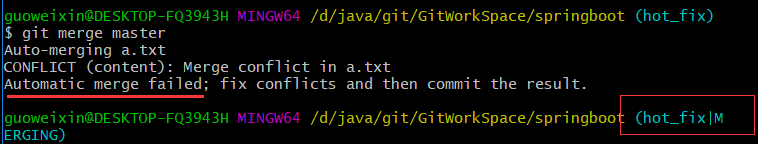


**查看版本信息：**



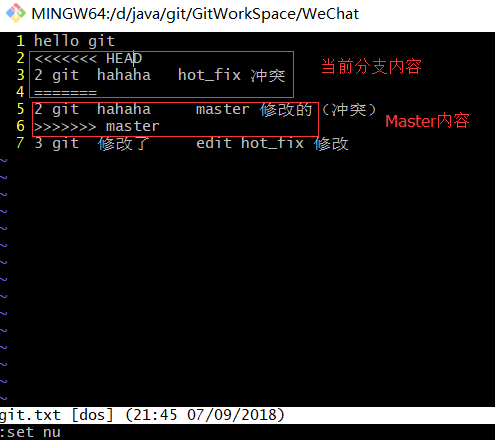
**合并：**

**git merge master //在hot\_fix中向master中合并代码**



**查看文件(冲突的)：**

**cat git.txt**



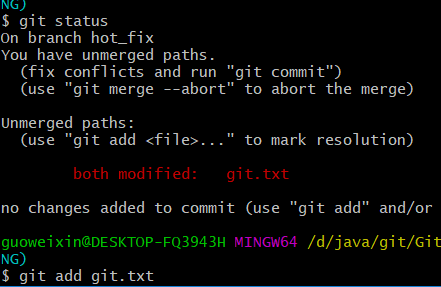
**处理冲突：**

* 1. **由人为沟通解决。**
  2. **进行文件编辑**

**修改文件：**

**vim git.txt (将冲突的符号清除。进行修改内容)**

**git add git.txt**



**git commit –m ‘冲突已修复’//不需要带文件名称**

