Git分布式版本控制工具

什么是Git?

- 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 与 SVN 区别

  Git 不仅仅是个版本控制系统，它也是个内容管理系统(CMS)，工作管理系统等。

  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。
- 我们写的代码需要使用Git进行管理。
- 有必要，因为人工的去处理不同的版本，做相应备份会很麻烦。
- Git是linux之父当年为了维护linux---linus之前也是手动维护合并把文件发给Linus
- linus自己写了一个版本管理的工具(Git)

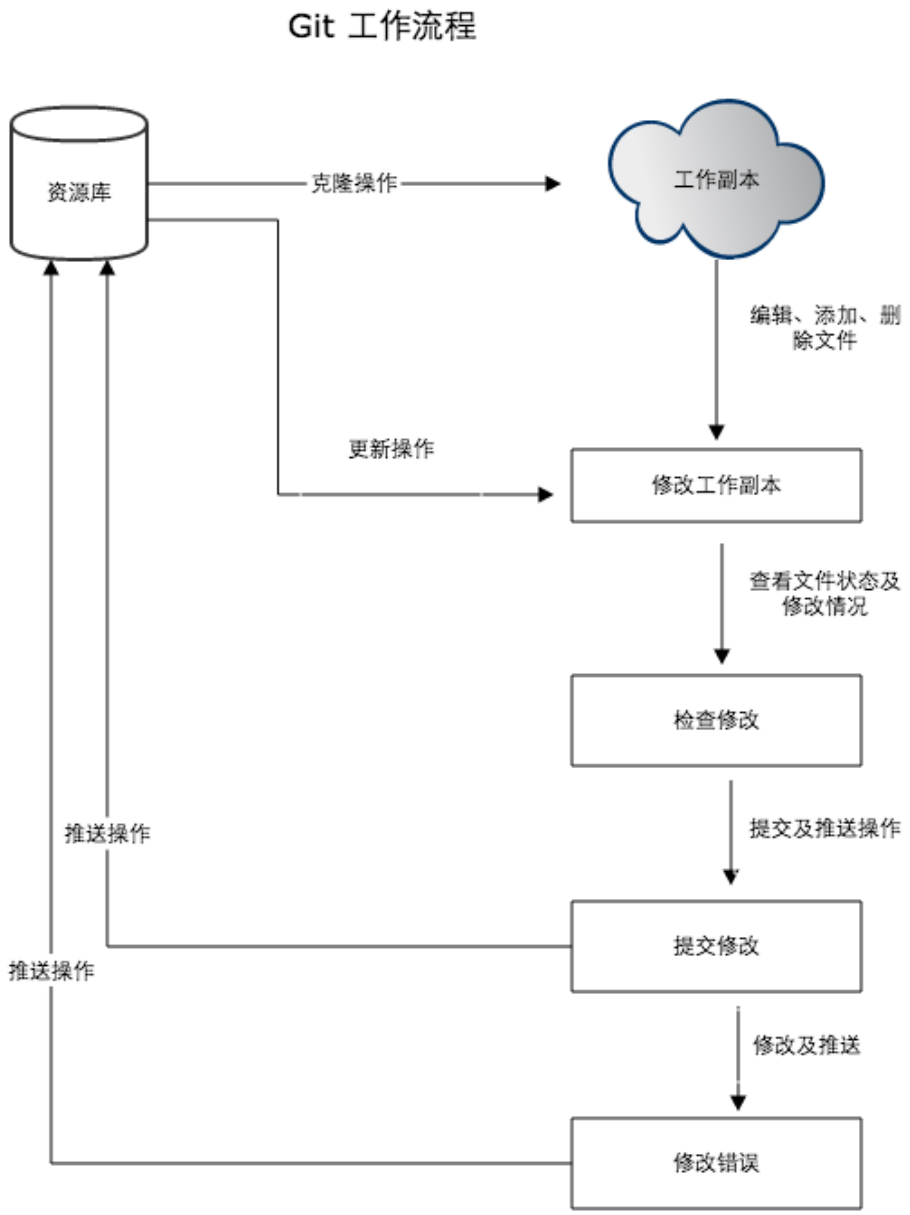
Git 工作流程

本章节我们将为大家介绍 Git 的工作流程。

一般工作流程如下：

- 克隆 Git 资源作为工作目录。
- 在克隆的资源上添加或修改文件。
- 如果其他人修改了，你可以更新资源。
- 在提交前查看修改。
- 提交修改。
- 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。

下图展示了 Git 的工作流程：



Git安装 傻瓜式下一步安装

初始化Git仓储/(仓库)

- 这个仓库会存放，git对我们项目代码进行备份的文件
- 在项目目录右键打开 git bash
- 命令:

```
git init
```

自报家门

- 就是在git中设置当前使用的用户是谁
- 每一次备份都会把当前备份者的信息存储起来

如何配置git的ssh key。

首先用如下命令（如未特别说明，所有命令均默认在Git Bash工具下执行）检查一下用户名和邮箱是否配置（github支持我们用用户名或邮箱登录）：

```
git config --global  --list
```

笔者的机器显示信息如下（已配置）：

如未配置，则执行以下命令进行配置：

```
git config --global  user.name "这里换上你的用户名"

git config --global user.email "这里换上你的邮箱"
```

然后执行以下命令生成秘钥：

```
ssh-keygen -t rsa -C "这里换上你的邮箱"
```

执行命令后需要进行3次或4次确认：

确认秘钥的保存路径（如果不需要改路径则直接回车）；
如果上一步置顶的保存路径下已经有秘钥文件，则需要确认是否覆盖（如果之前的秘钥不再需要则直接回车覆盖，如需要则手动拷贝到其他目录后再覆盖）；
创建密码（如果不需要密码则直接回车）；
确认密码；

在指定的保存路径下会生成2个名为id_rsa和id_rsa.pub的文件：

再打开你的github，进入配置页：

选择SSH and GPG keys项：

之前生成的是ssh秘钥，所以下面选择New SSH key（笔者这里已经配置了一个key，如果是未配置秘钥的用户，这里应该是空的）：

然后用文本工具打开之前生成的id_rsa.pub文件，把内容拷贝到key下面的输入框，并为这个key定义一个名称（通常用来区分不同主机），然后保存：

把大象放到冰箱要几步

1. 打开冰箱门
2. 放大象
3. 关上冰箱

把代码存储到.git仓储中

- 1.把代码放到仓储的门口

  ```
  git add ./readme.md 所指定的文件放到大门口
  git add ./ 把所有的修改的文件添加到大门口
  ```

- 2.把仓储门口的代码放到里面的房间中去

```
git commit -m "这是对这次添加的东西的说明"
```

可以一次性把我们修改的代码放到房间里(版本库)

```
git commit --all -m "一些说明"

=> --all 表示是把所有修改的文件提交到版本库
```

查看当前的状态

- 可以用来查看当前代码有没有被放到仓储中去
- 命令:

```
git status
```

git中的忽略文件

- .gitignore,在这个文件中可以设置要被忽略的文件或者目录。
- 被忽略的文件不会被提交仓储里去.
- 在.gitignore中可以书写要被忽略的文件的路径，以/开头，
  一行写一个路径，这些路径所对应的文件都会被忽略，
  不会被提交到仓储中
  - 写法
    - `/.idea` 会忽略.idea文件
    - `/js` 会忽略js目录里的所有文件
    - `/js/*.js` 会忽略js目录下所有js文件

查看日志

```
git log 查看历史提交的日志

git log --oneline 可以看到简洁版的日志
```

回退到指定的版本

- `git reset --hard Head~0`
  - 表示回退到上一次代码提交时的状态
- `git reset --hard Head~1`
  - 表示回退到上上次代码提交时的状态
- `git reset --hard [版本号]`
  - 可以通过版本号精确的回退到某一次提交时的状态
- `git reflog`
  - 可以看到每一次切换版本的记录:可以看到所有提交的版本号

分支

- 默认是有一个主分支master

创建分支

- `git branch dev`
  - 创建了一个dev分支
  - 在刚创建时dev分支里的东西和master分支里的东西是一样的

切换分支

- `git checkout [name]`
  - 切换到指定的分支,这里的切换到名为dev的分支
    `git branch` 可以查看当前有哪些分支

合并分支

- `git merge [name]`
  - 合并分支内容,把当前分支与指定的分支(dev),进行合并
  - 当前分支指的是`git branch`命令输出的前面有*号的分支
- 合并时如果有冲突，需要手动去处理，处理后还需要再提交一次.

删除分支

```
git branch -d [name]
```

GitHub

- https://github.com
- 不是git,只是一个网站
- 只不过这个网站提供了允许别通过git上传代码的功能

服务器地址操作

- 查看服务器地址：

  ```
  git remote -v
  ```

- 设置服务器地址:

  ```
   git remote origin set-url [url]
  ```

- 删除后添加:

  ```
  git remote rm origin
  git remote add origin [url]
  ```

提交代码到github(当作git服务器来用)

- `git push [地址] master`
- 示例: `git push origin master`
- 会把当前分支的内容上传到远程的master分支上
- `git fetch [地址] master`：抓取指令就是将仓库里的更新都抓取到本地，不会进行合并

- 示例: `git fetch origin master`
- `git pull [地址] master`: 拉取指令就是将远端仓库的修改拉到本地并自动进行合并，等同于fetch+merge
- 示例: `git pull origin master`
- 会把远程分支的数据得到:(*注意本地要初始一个仓储*)
- `git clone [地址]`
- 会得到远程仓储相同的数据,如果多次执行会覆盖本地内容。

模拟两个或多个用户一起操作

git一直输入密码在本地设置缓存

```
git config --global credential.helper store
```

gitee的使用与github相同