# 建立版本库

## git init

|  |
| --- |
| **作用：**  建立本地版本库 |
| **格式：**  git init <dir> |
| **示例一：**  $ mkdir repo1  $ cd repo1  $ git init  **示例二：**  $ git init repo1  示例一与示例二等价，均是在文件夹repo1下建立版本库。 |
| **进阶：**  在仓库中建立一个“.git”文件夹。  建立 master分支 |

## git clone

|  |
| --- |
| **作用：**  从远程主机克隆一个版本库到本地 |
| **格式：**  git clone [<options>] [--] <repo> [<dir>] |
| **示例一：**  $ git clone git@github.com:chtgit070920/first.git  **示例二**  $ git clone –o origin1 git@github.com:chtgit070920/first.git  **示例三**  $ git clone –o origin1 [git@github.com:chtgit070920/first.git](mailto:git@github.com:chtgit070920/first.git) first\_ |
| **进阶：**  该命令相当于：  Git init first //可使用“dir”自定义，默认和远程保持一致  Git remote add origin [**git@github.com:chtgit070920/first.git**](mailto:git@github.com:chtgit070920/first.git%20//) //主机名可使用“-o <主机名>”自  //定义，默认origin  Git fetch origin |

**多种协议:**

|  |
| --- |
| $ git clone http[s]://example.com/path/to/repo.git/  $ git clone ssh://example.com/path/to/repo.git/  $ git clone git://example.com/path/to/repo.git/  $ git clone /opt/git/project.git  $ git clone file:///opt/git/project.git  $ git clone ftp[s]://example.com/path/to/repo.git/  $ git clone rsync://example.com/path/to/repo.git/  ssh 还可如下写：  $ git clone [user@]example.com:path/to/repo.git/ |

# 远程管理

## 主机

### git remote add

|  |
| --- |
| **作用：**  添加远程主机 |
| **格式：**  git remote add <主机名> <网址> |
| **示例一：**  $ git remote add origin1 [git@github.com:chtgit070920/first.git](mailto:git@github.com:chtgit070920/first.git) |

### git remote rename

|  |
| --- |
| **作用：**  修改远程主机名 |
| **格式：**  git remote rename <主机名> |
| **示例一：**  $ git remote rename origin1 origin11 |

### git remote rm

|  |
| --- |
| **作用：**  删除远程主机 |
| **格式：**  git remote rm <主机名> |
| **示例一：**  $ git remote rm origin1 |

### git remote

|  |
| --- |
| **作用：**  列出所有的远程主机 |
| **格式：**  git remote |
| **示例一：**  $ git remote  origin1  origin2 |

### git remote -v

|  |
| --- |
| **作用：**  列出所有的远程主机的详细信息 |
| **格式：**  git remote –v |
| **示例一：**  $ git remote -v  origin1 git@github.com:chtgit070920/first.git (fetch)  origin1 git@github.com:chtgit070920/first.git (push)  origin2 git@github.com:chtgit070920/first.git (fetch)  origin2 git@github.com:chtgit070920/first.git (push) |

### git remote show

|  |
| --- |
| **作用：**  列出某远程主机的详细信息 |
| **格式：**  git remote show <主机名> |
| **示例一：**  $ git remote show origin1  \* remote origin1  Fetch URL: git@github.com:chtgit070920/first.git  Push URL: git@github.com:chtgit070920/first.git  HEAD branch: master  Remote branch:  master new (next fetch will store in remotes/origin1) |

## 抓取

### git fetch

|  |
| --- |
| **作用：**  远程主机的更新 |
| **格式：**  git fetch <主机名> [分支名] |
| **示例一：**  $ git fetch origin // 抓取远程所有的更新  **示例二：**  $ git fetch origin master // 抓取远程master分支的更新 |

### git pull

|  |
| --- |
| **作用：**  列出某远程主机的详细信息 |
| **格式：**  git remote show <主机名> |
| **示例一：**  $ git remote show origin1  \* remote origin1  Fetch URL: git@github.com:chtgit070920/first.git  Push URL: git@github.com:chtgit070920/first.git  HEAD branch: master  Remote branch:  master new (next fetch will store in remotes/origin1) |

## 推送

# git log

## git log

|  |
| --- |
| 列出所有历史记录，最近的排在最上方，显示提交对象的哈希值，作者、提交日期、和提交说明。如果记录过多，则按Page Up、Page Down、↓、↑来控制显示；按q退出历史记录列表。 |

## git log –n

|  |
| --- |
| 如果不想向上面那样全部显示，可以选择显示前N条。 |

## **git log --stat –n**

|  |
| --- |
| -n 同上，前n条，可省略。  显示简要的增改行数统计,每次提交文件的变更统计， |
|  |

## **git log -p –n**

|  |
| --- |
| -p : patch **[pætʃ]** 补丁，补片；  按补丁格式显示每个更新之间的差异。 |
| 同**git log --stat –n，不过显示的更全** |

## git log pretty=oneline

|  |
| --- |
| 一行显示，只显示哈希值和提交说明。 |

## **git log --pretty=format:" "**

|  |
| --- |
| 控制显示的记录格式，常用的格式占位符写法及其代表的意义如下：  选项 说明  %H 提交对象（commit）的完整哈希字串  %h 提交对象的简短哈希字串  %T 树对象（tree）的完整哈希字串  %t 树对象的简短哈希字串  %P 父对象（parent）的完整哈希字串  %p 父对象的简短哈希字串  %an 作者（author）的名字  %ae 作者的电子邮件地址  %ad 作者修订日期（可以用 -date= 选项定制格式）  %ar 作者修订日期，按多久以前的方式显示  %cn 提交者(committer)的名字  %ce 提交者的电子邮件地址  %cd 提交日期  %cr 提交日期，按多久以前的方式显示  %s 提交说明 |
| $ git log --pretty=format:"%h -%an,%ar : %s" -3  d0b9a20 -BeginMan,24 hours ago : ok  8c186cd -BeginMan,24 hours ago : mi  b2a3100 -BeginMan,24 hours ago : what?  显示了前3条的信息，简单的哈希值，作者，提交时间，提交说明。  为了方面使用，可以做个别名：  $ git config alias.logs "log --pretty=format:'%h -%an,%ar:%s'"  $ git config alias.logs log --pretty=format:'%h -%an,%ar:%s' |

## **git lot –graph**

|  |
| --- |
| ASCII 字符串表示的简单图形，形象地展示了每个提交所在的分支及其分化衍合情况 |