`git clone` 命令将存储库克隆到新目录中。

简介

```
git clone [--template=<template_directory>]
      [-l] [-s] [--no-hardlinks] [-q] [-n] [--bare] [--mirror]
      [-o <name>] [-b <name>] [-u <upload-pack>] [--reference <repository>]
      [--dissociate] [--separate-git-dir <git dir>]
      [--depth <depth>] [--[no-]single-branch]
      [--recurse-submodules] [--[no-]shallow-submodules]
      [--jobs <n>] [--] <repository> [<directory>]
```

描述

将存储库克隆到新创建的目录中，为克隆的存储库中的每个分支创建远程跟踪分支(使用 `git branch -r` 可见)，并从克隆检出的存储库作为当前活动分支的初始分支。

在克隆之后，没有参数的普通 git 提取将更新所有远程跟踪分支，并且没有参数的 `git pull` 将另外将远程主分支合并到当前主分支(如果有的话)。

此默认配置通过在 `refs/remotes/origin` 下创建对远程分支头的引用，并通过初始化 `remote.origin.url` 和 `remote.origin.fetch` 配置变量来实现。

执行远程操作的第一步，通常是从远程主机克隆一个版本库，这时就要用到 `git clone` 命令。

```
$ git clone <版本库的网址>
```

比如，克隆 jQuery 的版本库。

```
$ git clone http://github.com/jquery/jquery.git
```

该命令会在本地主机生成一个目录，与远程主机的版本库同名。如果要指定不同的目录名，可以将目录名作为 `git clone` 命令的第二个参数。

```
$ git clone <版本库的网址> <本地目录名>
```

`git clone` 支持多种协议，除了 HTTP(s)以外，还支持 SSH、Git、本地文件协议等，下面是一些例子。

示例

以下是所支持协议的一些示例 -

```
$ git clone http[s]://example.com/path/to/repo.git
$ git clone http://git.oschina.net/yiibai/sample.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 协议还有另一种写法。

```
$ git clone [user@]example.com:path/to/repo.git
```

通常来说，Git 协议下载速度最快，SSH 协议用于需要用户认证的场合。

应用场景示例

从上游克隆下来：

```
$ git clone git://git.kernel.org/pub/scm/.../linux.git mydir
$ cd mydir
$ make # 执行代码或其它命令
```

在当前目录中使用克隆，而无需检出：

```
$ git clone -l -s -n . ../copy
$ cd ../copy
$ git show-branch
```

从现有本地目录借用从上游克隆：

```
$ git clone --reference /git/linux.git
    git://git.kernel.org/pub/scm/.../linux.git
    mydir
$ cd mydir
```

创建一个裸存储库以将您的更改发布给公众：

```
$ git clone --bare -l /home/proj/.git /pub/scm/proj.git
```