

版本控制神器之Git深入介绍

lecturer:波波老师

Git简介

Git (全局信息追踪器)。

Git是一个分布式版本控制工具，Git的使用中央仓库不是必须的，用户本地就是一个完整的版本仓库，代码的前进、回退、删除等等操作都可以直接在本地进行，不需要中央仓库。但是，在实际操作中，为了能够和其他同事快速沟通以及合并代码，一般还是会搭建一个中央仓库。Git对分支的管理非常友好，可以快速创建或者合并分支。

Svn集中式的版本控制工具，Svn中，必须要有中央仓库，所有的版本信息都保存在中央仓库中，代码的前进、回退、删除等等操作都需要在中央仓库中进行，用户本地保存的只是版本仓库的一个副本，Svn中的分支非常臃肿。

Git安装

<https://gitforwindows.org/>

链接：<https://pan.baidu.com/s/18xouiB4LcRCbkp0c7zzWyA>


提取码：bobo

配置个人信息

```
git config --global user.name 'bobo'
```

```
git config --global user.email 'dengpbs@163.com'
```

在多个客户端的情况下用身份识别的

 MINGW64:/c/Users/admin/Desktop

```
admin@DESKTOP-TBA12IL MINGW64 ~/Desktop
$ git config --global user.name 'bobo'

admin@DESKTOP-TBA12IL MINGW64 ~/Desktop
$ git config --global user.email 'dengpbs@163.com'

admin@DESKTOP-TBA12IL MINGW64 ~/Desktop
$ |
```

基本操作

1.创建版本库

指定一个文件夹位置即可

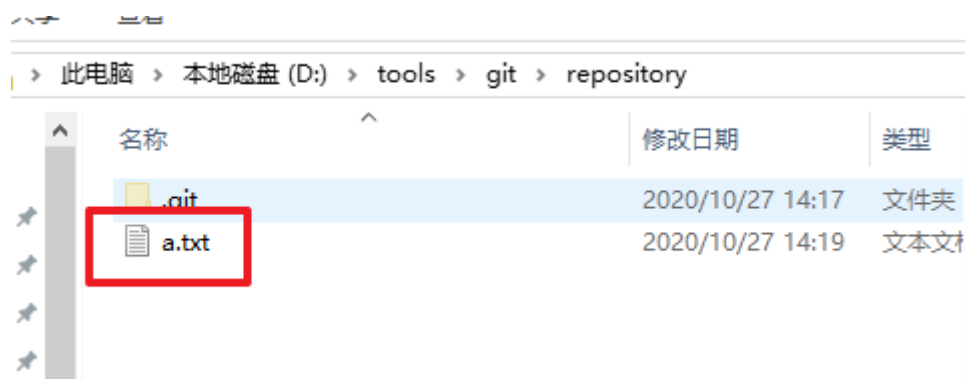
2.初始化操作

我们要想将某个文件夹作为我们的版本库还需要通过 `git init` 命令来初始化。



3.添加文件到版本库中

想要把某个文件管理起来，首先创建一个文件



然后通过 `git add` 命令添加到版本库中

```
admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git add a.txt
warning: LF will be replaced by CRLF in a.txt.
The file will have its original line endings in your working directory
```

再去执行 `git commit -m '备注'` 命令，来提交

```
admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git commit -m '第一次提交'
[master (root-commit) 427b5dc] 第一次提交
1 file changed, 1 insertion(+)
create mode 100644 a.txt
```

`commit -m`的-m后面跟的是本次操作的备注说明信息。最好是有意义的，也就是下次看到这个说明就清楚提交了什么内容。

为什么Git添加文件需要add，commit一共两步呢？因为commit可以一次提交很多文件，所以你可以多次add不同的文件

```
admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ vi b.txt

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ vi c.txt

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git add b.txt
warning: LF will be replaced by CRLF in b.txt.
The file will have its original line endings in your working directory

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git add c.txt
warning: LF will be replaced by CRLF in c.txt.
The file will have its original line endings in your working directory

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git commit -m '第二次提交'
[master 93fa093] 第二次提交
2 files changed, 2 insertions(+)
create mode 100644 b.txt
create mode 100644 c.txt
```

4.status和diff命令

命令	介绍
status	查看当前库的状态
diff	是difference的简写，是用来查看文件的变化的【工作区和版本库】

5.版本回退

5.1 log命令

查看文件的历史版本

```
admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git log a.txt
commit 89091e3522a83c548792f5563d204821f39ee85f (HEAD -> master)
Author: bobo <dengpbs@163.com>
Date: Tue Oct 27 14:43:17 2020 +0800

    123

commit 2ef2f3ec3ca36aec2d28aa8a726047b98823e095
Author: bobo <dengpbs@163.com>
Date: Tue Oct 27 14:42:30 2020 +0800

    又修改了a.txt文件

commit b5958446bbe3b5c801234d381291c3e1bb3addc8
Author: bobo <dengpbs@163.com>
Date: Tue Oct 27 14:32:34 2020 +0800

    修改了a.txt文件

commit 427b5dc1d243918d1ac6ec9b451fcc0614651932
Author: bobo <dengpbs@163.com>
Date: Tue Oct 27 14:22:07 2020 +0800

    第一次提交
```

简化日志文件的显示方式 --pretty=oneline

```
admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git log --pretty=oneline a.txt
89091e3522a83c548792f5563d204821f39ee85f (HEAD -> master) 123
2ef2f3ec3ca36aec2d28aa8a726047b98823e095 又修改了a.txt文件
b5958446bbe3b5c801234d381291c3e1bb3addc8 修改了a.txt文件
427b5dc1d243918d1ac6ec9b451fcc0614651932 第一次提交
```

5.2 版本回退

回退到上一个版本

```
git reset --hard HEAD^
```

```
admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ cat a.txt
okokoko
nonononono
cccccc
123123123

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git reset --hard HEAD^
HEAD is now at 2ef2f3e 又修改了a.txt文件

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ cat a.txt
okokoko
nonononono
cccccc
```

回退到指定的版本

```
git reset --hard 版本编号
```

```
admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git reset --hard 427b5dc
HEAD is now at 427b5dc 第一次提交

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ cat a.txt
okokoko

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$
```

回退到最新版本

一种方式就是可以查看之前记录的最新版本的编号，但是log是看不到最新的日志记录的哦

```
admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git log a.txt
commit 427b5dc1d243918d1ac6ec9b451fcc0614651932 (HEAD -> master)
Author: bobo <dengpbs@163.com>
Date: Tue Oct 27 14:22:07 2020 +0800
```

```

admin@DESKTOP-TBA12IL MINGW64 //tools/git/repository (master)
$ git reset --hard 89091e35
HEAD is now at 89091e3 123

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ cat a.txt
okokoko
nonononono
cccccc
123123123

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)

```

我们还可以借助另一个命令 `git reflog`

```

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git log a.txt
commit 427b5dc1d243918d1ac6ec9b451fcc0614651932 (HEAD -> master)
Author: bobo <dengpbs@163.com>
Date: Tue Oct 27 14:22:07 2020 +0800

    第一次提交

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git reflog a.txt
427b5dc (HEAD -> master) HEAD@{0}: reset: moving to 427b5dc1d
89091e3 HEAD@{1}: reset: moving to 89091e35
427b5dc (HEAD -> master) HEAD@{2}: reset: moving to 427b5dc
2ef2f3e HEAD@{3}: reset: moving to HEAD^
89091e3 HEAD@{4}: commit: 123
2ef2f3e HEAD@{5}: commit: 又修改了a.txt文件
b595844 HEAD@{6}: commit: 修改了a.txt文件
427b5dc (HEAD -> master) HEAD@{9}: commit (initial): 第一次提交

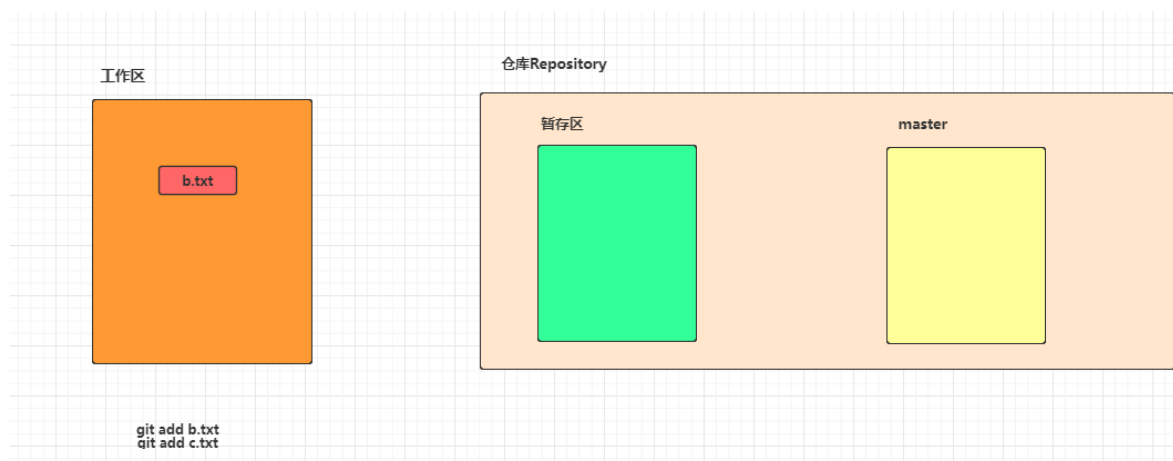
```

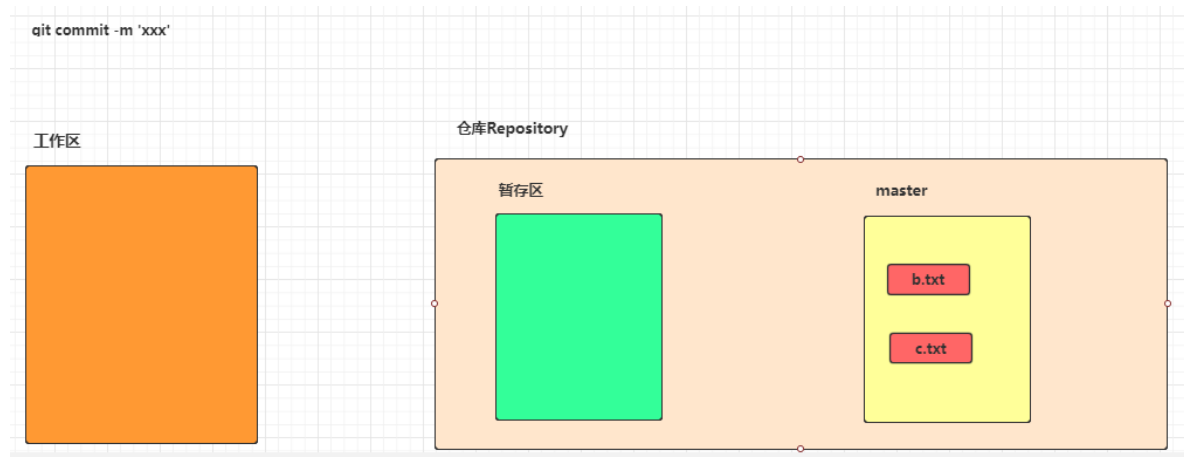
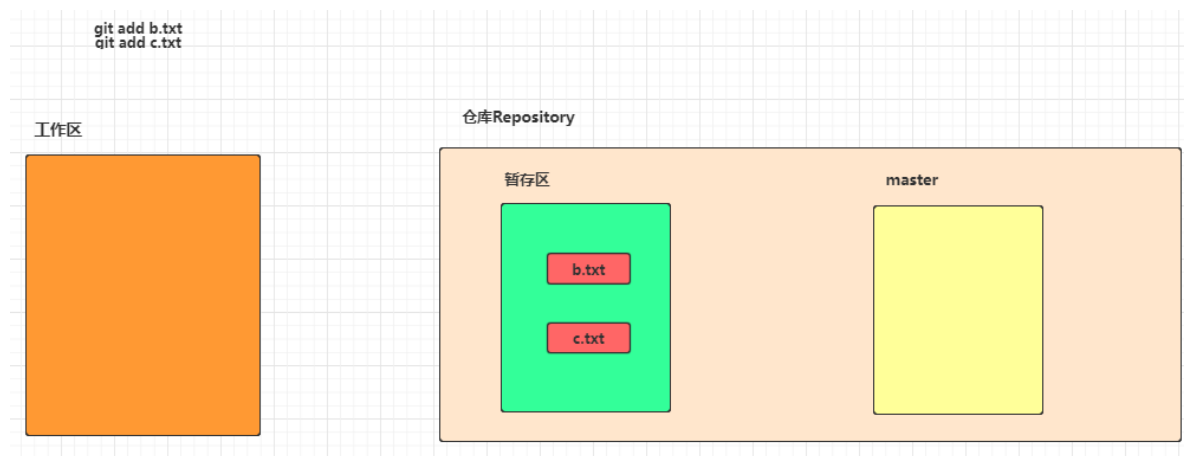
在这个命令中会记录过往的所有的操作，包括版本的切换回退。

工作区和暂存区

`git add xxx` 该命令只能将文件从工作区提交到暂存区

`git commit` 只能将文件从暂存区提交到版本库中





撤销管理

1.还未提交到暂存区

git checkout -- fileName

```
Untracked files:
  (use "git add <file>..." to include in what will be committed)

  c.txt

no changes added to commit (use "git add" and/or "git commit -a")

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ cat a.txt
okokoko
123456

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
git checkout -- a.txt

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ cat a.txt
okokoko

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
```

2.提交到了暂存区

git reset HEAD file 移除暂存区回到工作区

git checkout -- fileName 撤销操作

```
admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git reset HEAD a.txt
Unstaged changes after reset:
M   a.txt
M   b.txt
admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
```

3.已经提交到了版本库中

直接回退到上个版本即可

```
admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git reset --hard HEAD^
HEAD is now at 2e37455 修改了b.txt

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ cat a.txt
okokoko
```

git reset --hard HEAD^

删除管理

操作和添加文件是差不多的，这么理解的话就很容易了~！

工作区：是我们直接编辑的地方，比如Eclipse打开的项目，记事本打开的文件，直接操作

暂存区：数据暂时存放的区域，是工作区和版本库之间进行数据交流的纽带

版本库：存放已经提交的数据，push的时候，就是将这个区域中的数据push到远程仓库的

GitHub远程仓库

1.创建SSH Key

ssh-keygen -t ras -C "dengpbs@163.com"

```

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ ssh-keygen -t rsa -C "dengpbs@163.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/admin/.ssh/id_rsa):
Created directory '/c/Users/admin/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/admin/.ssh/id_rsa.
Your public key has been saved in /c/Users/admin/.ssh/id_rsa.pub.
The key's fingerprint is:
SHA256:thNQmPieE6rB75J3gWY0uq0h+MfllvNdZuiFzq8fo dengpbs@163.com
The key's randomart image is:
+---[RSA 2048]-----+
|      .+.      |
|      . +o      |
|      ....      |
|      =+        |
|      o.=S      |
|      o  ==.o=   |
|      .. o++++B  +
|      ... 0000X+o o
|      .o+o..=+=E.
+---[SHA256]-----+

```

一直回车即可

此电脑 > 本地磁盘 (C:) > 用户 > admin > .ssh

名称	修改日期	类型
id_rsa 私钥	2020/10/28 14:34	文件
id_rsa.pub 公钥	2020/10/28 14:34	PUB 文

2.登录GitHub

配置SSH

The screenshot shows the GitHub homepage. At the top, there are navigation links: Pull requests, Issues, Marketplace, and Explore. On the right, a user is signed in as 'q279583842q'. A dropdown menu is open, showing options like 'Set status', 'Your profile', 'Your repositories', 'Your projects', 'Your stars', 'Your gists', 'Upgrade', 'Feature preview', 'Help', 'Settings' (highlighted with a red box), and 'Sign out'. The main content area features a large green box with the text 'Learn Git and GitHub without any code!' and buttons for 'Read the guide' and 'Start a project'. Below this, there's a section for 'Loading activity...' with a circular progress indicator.

q279583842q
Personal settings

Profile

Account

Account security

Security log

Security & analysis

Emails

Notifications

Billing & plans

SSH and GPG keys

Repositories


Organizations


Saved replies

SSH keys

New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

 **dpb-git**
SHA256: 1Q/omyC6FkcJ0IC3cru0un1Gk0Z0sUv+gFd2FzPcrII
Added on 28 Jan 2019
Last used within the last 11 months — Read/write
Delete

 **bb-git**
SHA256: dUrf2j2I1tQmvs6A2xhAW3FIW60S+21sYDnny+FIPdg
Added on 3 Aug 2019
Last used within the last 10 months — Read/write
Delete

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH Problems](#).

GPG keys

New GPG key

There are no GPG keys associated with your account.

Learn how to [generate a GPG key](#) and add it to your account.

Personal settings

Profile

Account

Account security

Security log

Security & analysis

Emails

Notifications

Billing & plans

SSH and GPG keys

Repositories

Organizations

Saved replies

Title

Gp-git

自定义名称

Key

ssh-rsa
AAAAAB3NzaC1yc2EAAAADAQABAAQAC7hwITxY6XEBkUAXVjAZVkj7NyceoVD69i0JSdF445YDvqN5i1AFw8dUHA
UKS65t4oC03c6b+fB83TSLb4MoLoiEA1gzvGdUGMn+BdgadA/RkjNdt3ENLgmSTVchM8kdd6jPYUjMI8RZVChRsYV
pA3VHy/HK9QarADdWo+k0H5GzoYpmKmkQFL+92XPH95VEIFS+cDf/W947Z+weghuHDEFQqFbdxXGUSdjiw8vCeZ
KOuhZ+KtYwVw3wSDDhEE3XG71cUcg0GSnQJHyWC+9kjD+hCxVtGVUT1BVu8HZxrn/IF0ulucvEqkXS8zWZu2ohP
xdA0H1Amy5eSMDHDO07 dengpbs@163.com

拷贝生成的公钥中的内容即可

Add SSH key

3.创建远程仓库

A repository contains all project files, including the revision history. You may have a project repository elsewhere. [Import a repository.](#)

Owner *

Repository name *

q279583842q ▾

/ gp-repository-01 ✓


定义仓库的名称

Great repository names are short and memorable. Need inspiration? How about [upgraded-couscous?](#)

Description (optional)

Gp 第一个远程仓库


添加描述信息



Public

Anyone on the internet can see this repository. You choose who can commit.

开发的



Private

You choose who can see and commit to this repository.

私有的

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ Add a README file

This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

☐ Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

4.关联远程仓库

Quick setup — if you've done this kind of thing before

Set up in Desktop

 or

HTTPS

SSH

https://github.com/q279583842q/gp-repository-01.git

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

 在本地新创建一个仓库和远程仓库关联

```
echo "# gp-repository-01" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/q279583842q/gp-repository-01.git
git push -u origin main
```

...or push an existing repository from the command line

 直接在本地以及存在的仓库中关联远程仓库

```
git remote add origin https://github.com/q279583842q/gp-repository-01.git
git branch -M main
git push -u origin main
```

...or import code from another repository

 导入其他的

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

```
git remote add origin git@github.com:q279583842q/gp-repository-01.git
```

5.推送文件到远程库中

我们只能将版本库中文件推送到远程库中，推送的命令是

```
git push -u origin master
```

 第一次推送的时候要加上-u 后面就不用了

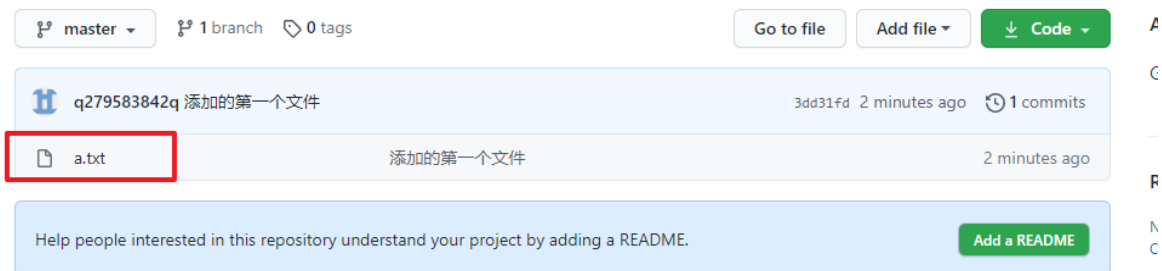
```
admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository1 (master)
$ git add a.txt
warning: LF will be replaced by CRLF in a.txt.
The file will have its original line endings in your working directory

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository1 (master)
$ git commit -m '添加的第一个文件'
[master (root-commit) 3dd31fd] 添加的第一个文件
1 file changed, 2 insertions(+)
create mode 100644 a.txt

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository1 (master)
$ Git status
On branch master
nothing to commit, working tree clean

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository1 (master)
$ git push -u origin master
The authenticity of host 'github.com (13.229.188.59)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWGL7E1IGOCspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,13.229.188.59' (RSA) to the list of known hosts.
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 230 bytes | 230.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:q279583842q/gp-repository-01.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

推送成功后，我们就可以在远程仓库中看到对应的文件



6.克隆远程仓库

```
git clone git@github.com:q279583842q/gp\_repository\_02.git
```

```

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git
$ git clone git@github.com:q279583842q/gp_repository_02.git
Cloning into 'gp_repository_02'...
Warning: Permanently added the RSA host key for IP address '13.250.177.223' to
the list of known hosts.
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git
$ ll
total 4
drwxr-xr-x 1 admin 197121 0 10月 28 15:05 gp_repository_02/
drwxr-xr-x 1 admin 197121 0 10月 28 14:25 repository/
drwxr-xr-x 1 admin 197121 0 10月 28 14:47 repository1/

```

```

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git
$ ll
total 4
drwxr-xr-x 1 admin 197121 0 10月 28 15:05 gp_repository_02/
drwxr-xr-x 1 admin 197121 0 10月 28 14:25 repository/
drwxr-xr-x 1 admin 197121 0 10月 28 14:47 repository1/

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git
$ cd gp_repository_02/

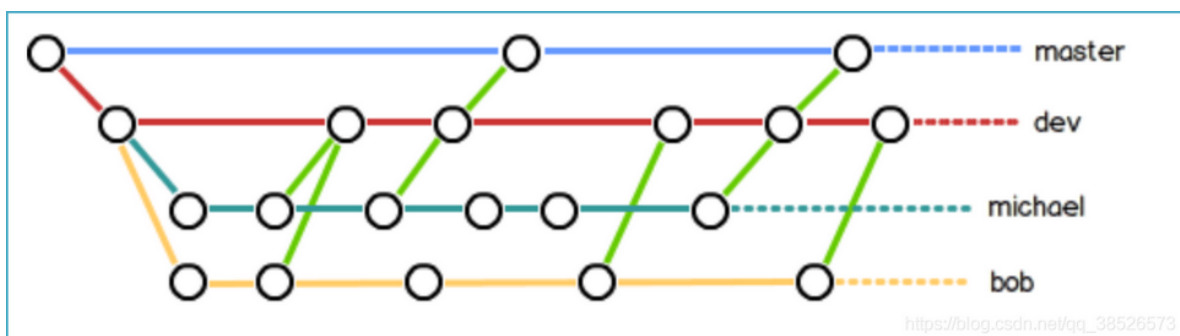
admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/gp_repository_02 (main)
$ ll
total 2
-rw-r--r-- 1 admin 197121 12 10月 28 15:05 hello.txt
-rw-r--r-- 1 admin 197121 43 10月 28 15:05 README.md

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/gp_repository_02 (main)
$ cat hello.txt
你好啊~

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/gp_repository_02 (main)
$

```

分支管理



1.创建及合并分支

git branch 查看分支

```

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/gp_repository_02 (main)
$ git branch
* main

```

git branch dev1 创建分支

```
admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git branch dev1

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git branch
dev1
* master

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ |
```

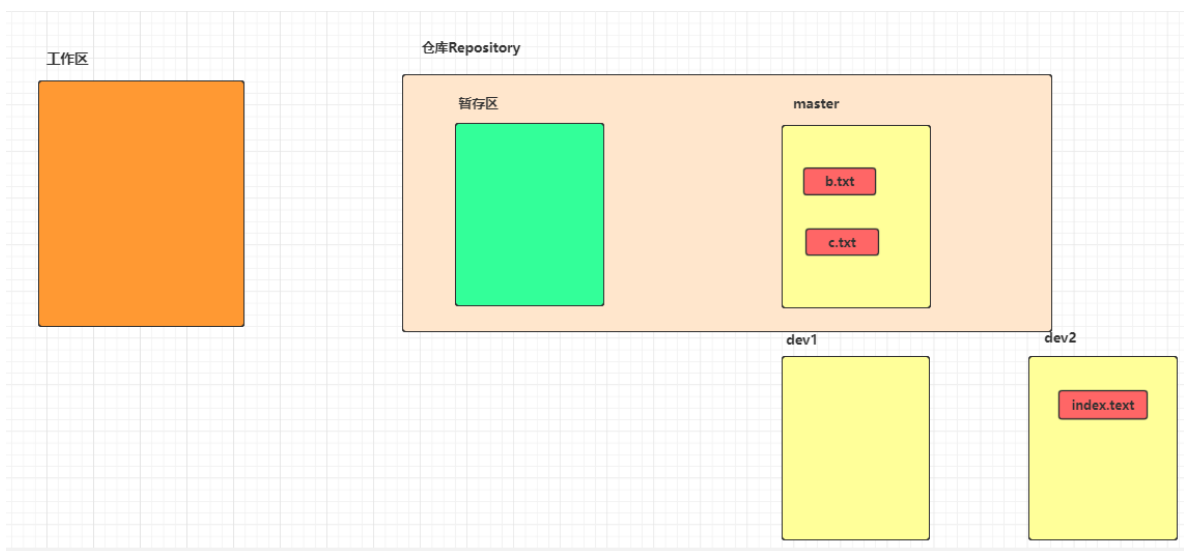
git checkout dev1 切换分支

```
admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git checkout dev1
Switched to branch 'dev1'

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (dev1)
$ git branch
dev1
* master
```

git checkout -b dev2 创建分支并且直接切换到分支中

2.分支的操作



我们在分支上的相关的操作其实是独立的，我们提交到分支对应的版本库中的情况下，其他分支中是查看不到了，包括主分支，如果我们想要在主分支中查看到对应的信息，我们就需要操作分支合并。

首先切换到master分支中然后执行 merge 命令即可

```

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (dev1)
$ git checkout master
Switched to branch 'master'

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git branch
dev1
dev2
* master

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git merge dev2
Updating 2e37455..aed110b
Fast-forward
 index.text | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 index.text

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ ll
total 4
-rw-r--r-- 1 admin 197121  9 10月 28 14:25 a.txt
-rw-r--r-- 1 admin 197121 14 10月 28 14:22 b.txt
-rw-r--r-- 1 admin 197121  5 10月 28 14:06 c.txt
-rw-r--r-- 1 admin 197121  7 10月 28 15:22 index.text

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)

```

3.删除分支

git branch -d dev1

```

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git branch
dev1
dev2
* master

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git branch -d dev1
Deleted branch dev1 (was 2e37455).

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git branch
dev2
* master

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)

```

4.解决冲突

```

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git merge dev1
Updating aed110b..caf6f16
Fast-forward
 a.txt | 1 +
 1 file changed, 1 insertion(+)

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ cat a.txt
okokoko
123456

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git merge dev2
Auto-merging a.txt
CONFLICT (content): Merge conflict in a.txt
Automatic merge failed; fix conflicts and then commit the result.

```

在合并文件的时候出现了冲突，这时候的解决方案非常简单，直接编辑冲突的文件即可，然后 add commit 操作即可

```

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git merge --no-ff d2 -m '合并了d2分支'
Merge made by the recursive strategy.
 dd | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 dd

admin@DESKTOP-TBA12IL MINGW64 /d/tools/git/repository (master)
$ git log --graph --pretty=oneline --abbrev-commit
*    b6339b0 (HEAD -> master) 合并了d2分支
/
*    ea33edb (d2) d1
/
*    d57e269 (d1) 解决了冲突问题
|
*    ad15347 dev2 修改了a.txt文件
*    caf6f16 dev1 修改了a.txt
/
*    aed110b dev2 提交了index.text
*    2e37455 修改了b.txt
*    79929c7 提交了 b.txt 和 c.txt 两个文件
*    427b5dc 第一次提交

```

Bug分支

针对是突发的任务，我们保持当前分支的工作状态，而去处理临时任务，处理完成后又可以继续回落工作

git stash 存储当前装

git stash list 查看存储的状态列表

git stash apply 恢复之前的状态

多人协作

从远处库中更新最新的文件

`git pull` 更新同步远处库中的内容

`git push origin master` 推送文件到远程库中

`git push origin dev1` 推送到GitHub中的dev1分支中，如果不存在，就会创建dev1分支

`git branch --set-upstream-to=origin/dev1 dev1` 本地分支和远程库中的分支不一致的情况下可以设置对应关系

因此，多人协作的工作模式通常是这样：

1. 首先，可以试图用`git push origin` 推送自己的修改；
2. 如果推送失败，则因为远程分支比你的本地更新，需要先用`git pull`试图合并；
3. 如果合并有冲突，则解决冲突，并在本地提交；
4. 没有冲突或者解决掉冲突后，再用`git push origin` 推送就能成功！
5. 如果`git pull`提示no tracking information，则说明本地分支和远程分支的链接关系没有创建，用命令`git branch --set-upstream-to origin/`。

标签管理

`git`里边默认的版本好不容易记。对于一些里程碑版本，需要记下来，此时可以使用标签，给项目的发布版本打标签，也是标签的一个重要功能之一

常用命令	说明
mkdir XX	(创建一个空目录 XX指目录名)
pwd	显示当前目录的路径。
git init	把当前的目录变成可以管理的git仓库，生成隐藏.git文件。
git add XX	把xx文件添加到暂存区去。
git commit -m "XX"	提交文件 -m 后面的是注释。
git status	查看仓库状态
git diff XX	查看XX文件修改了那些内容
git log	查看历史记录
git reset --hard HEAD^ git reset --hard HEAD~	回退到上一个版本(如果想回退到100个版本，使用git reset --hard HEAD~100)
cat XX	查看XX文件内容
git reflog	查看历史记录的版本号id
git checkout -- XX	把XX文件在工作区的修改全部撤销。
git rm XX	删除XX文件
git remote add origin git@github.com:q279583842/q/gitRepository1.git	关联一个远程库
git push -u(第一次要用-u 以后不需要) origin master	把当前master分支推送到远程库

常用命令	说明
git clone git@github.com:q279583842q/gitRepository1 .git	从远程库中克隆
git checkout -b dev	创建dev分支 并切换到dev分支上
git branch	查看当前所有的分支
git checkout master	切换回master分支
git merge dev	在当前的分支上合并dev分支
git branch -d dev	删除dev分支
git branch name	创建分支
git stash	把当前的工作隐藏起来 等以后恢复现场后继续工作
git stash list	查看所有被隐藏的文件列表
git stash apply	恢复被隐藏的文件，但是内容不删除
git stash drop	删除文件
git stash pop	恢复文件的同时 也删除文件
git remote	查看远程库的信息
git remote -v	查看远程库的详细信息
git push origin master	Git会把master分支推送到远程库对应的远程分支上
git tag <tagname>	创建标签
git tag -a <tagname> -m "blablabla..."	创建标签并且指定信息
git tag	查看所有的标签
git push origin <tagname>	推送一个本地标签到远程
git push origin --tags	可以推送全部未推送过的本地标
git tag -d <tagname>	可以删除一个本地标签
git push origin :refs/tags/<tagname>	可以删除一个远程标签