

Centos 7 搭建 Gitlab 伺服器超詳細（搭建成功）

<https://www.cnblogs.com/zhangycun/p/10963094.html>

一、安裝並組態必要的依賴關係

在 CentOS 系統上安裝所需的依賴：ssh，防火牆，postfix(用於郵件通知)，wget，以下這些命令也會打開系統防火牆中的 HTTP 和 SSH 連接埠訪問。

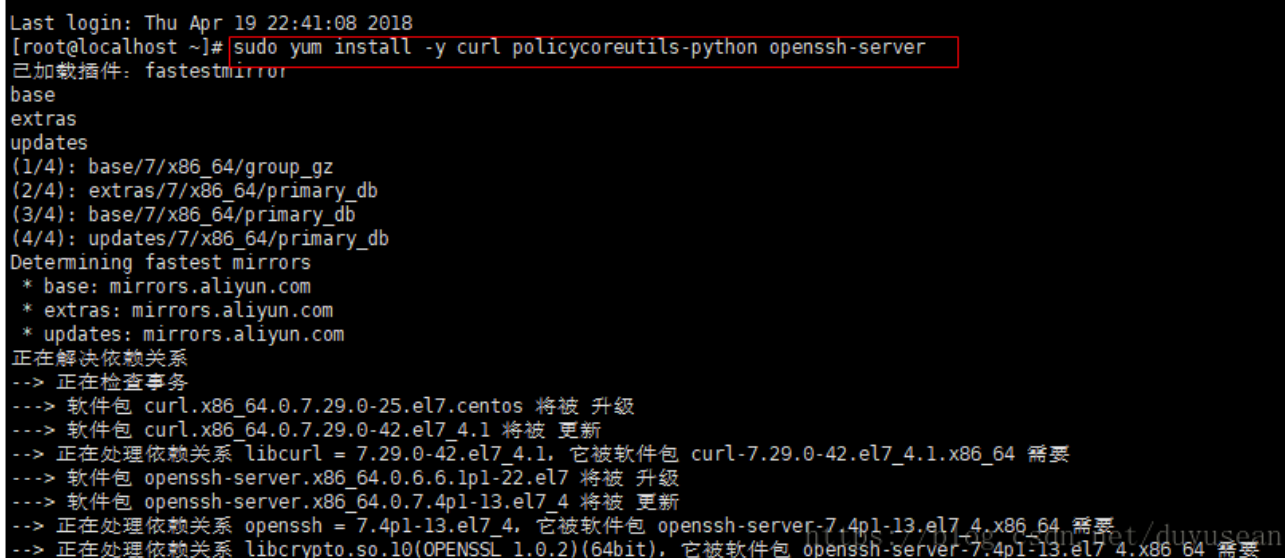
注意：使用者不是管理員權限，出現如下警告

使用者不在 sudoers 檔案中此事將被報告

可以使用 su root 切換 root 權限

1. 安裝 ssh

```
sudo yum install -y curl policycoreutils-python openssh-server
```



```
Last login: Thu Apr 19 22:41:08 2018
[root@localhost ~]# sudo yum install -y curl policycoreutils-python openssh-server
已加载插件: fastestmirror
base
extras
updates
(1/4): base/7/x86_64/group_gz
(2/4): extras/7/x86_64/primary_db
(3/4): base/7/x86_64/primary_db
(4/4): updates/7/x86_64/primary_db
Determining fastest mirrors
 * base: mirrors.aliyun.com
 * extras: mirrors.aliyun.com
 * updates: mirrors.aliyun.com
正在解决依赖关系
--> 正在检查事务
--> 软件包 curl.x86_64.0.7.29.0-25.el7.centos 将被 升级
--> 软件包 curl.x86_64.0.7.29.0-42.el7_4.1 将被 更新
--> 正在处理依赖关系 libcurl = 7.29.0-42.el7_4.1, 它被软件包 curl-7.29.0-42.el7_4.1.x86_64 需要
--> 软件包 openssh-server.x86_64.0.6.6.1p1-22.el7 将被 升级
--> 软件包 openssh-server.x86_64.0.7.4p1-13.el7_4 将被 更新
--> 正在处理依赖关系 openssh = 7.4p1-13.el7_4, 它被软件包 openssh-server-7.4p1-13.el7_4.x86_64 需要
--> 正在处理依赖关系 libcrypto.so.10(OPENSSL_1.0.2)(64bit), 它被软件包 openssh-server-7.4p1-13.el7_4.x86_64 需要
```

若出現以下圖片的資訊則表示成功

```

验证中      : libselinux-2.2.2-6.el7.x86_64

已安装:
policycoreutils-python.x86_64 0:2.5-17.1.el7

作为依赖被安装:
audit-libs-python.x86_64 0:2.7.6-3.el7  checkpolicy.x86_64 0:2.5-4.el7      libcgroup.x86_64
python-IPy.noarch 0:0.75-6.el7          setools-libs.x86_64 0:3.3.8-1.1.el7

更新完毕:
curl.x86_64 0:7.29.0-42.el7_4.1  dracut.x86_64 0:033-502.el7_4.1  openssh-server.x86_64 0:7.9p1-8.el7_4.1

作为依赖被升级:
audit.x86_64 0:2.7.6-3.el7          audit-libs.x86_64 0:2.7.6-3.el7      dracut-core.x86_64 0:033-502.el7_4.1
libcurl.x86_64 0:7.29.0-42.el7_4.1  libgudev1.x86_64 0:219-42.el7_4.10  libselinux.x86_64 0:2.2.2-6.el7_4.1
libsemanage.x86_64 0:2.5-8.el7      libsepol.x86_64 0:2.5-6.el7          openssh.x86_64 0:7.9p1-8.el7_4.1
openssl.x86_64 1:1.0.2k-8.el7      openssl-libs.x86_64 1:1.0.2k-8.el7  policycoreutils-python.x86_64 0:2.5-17.1.el7
systemd-libs.x86_64 0:219-42.el7_4.10  systemd-sysv.x86_64 0:219-42.el7_4.10

完毕!
[root@localhost ~]#

```

<https://blog.csdn.net/duyusean>

2. 將 SSH 服務設定成開機自啟動

安裝命令：`sudo systemctl enable sshd`

3. 啟動 SSH 服務

安裝命令：`sudo systemctl start sshd`

```

完毕!
[root@localhost ~]# sudo systemctl enable sshd
[root@localhost ~]# sudo systemctl start sshd
[root@localhost ~]#

```

<https://blog.csdn.net/duyusean>

4. 安裝防火牆

（如果已經安裝了防火牆並且已經在運行狀態，則可直接進行第 6 步）

`yum install firewalld systemd -y`

```
[root@localhost ~]# yum install firewalld systemd -y
```

若出現“完畢！”的字樣，則表示安裝成功

```
作为依赖被升级：
NetworkManager-libnm.x86_64 1:1.8.0-11.el7_4
NetworkManager-wifi.x86_64 1:1.8.0-11.el7_4
libnl3.x86_64 0:3.2.28-4.el7

替代：
NetworkManager.x86_64 1:1.0.6-27.el7

完毕！ https://blog.csdn.net/duyusean
[root@localhost ~]#
```

5. 開啟防火牆

安裝命令：`service firewalld start`

```
[root@localhost ~]# service firewalld start
Redirecting to /bin/systemctl start firewalld.service
```

6. 新增 http 服務到 firewalld

`permanent` 表示永久生效，若不加 `--permanent` 系統下次啟動後就會失效。

安裝命令：`sudo firewall-cmd --permanent --add-service=http`

```
[root@localhost ~]# sudo firewall-cmd --permanent --add-service=http
success
```

7. 重啟防火牆

安裝命令：`sudo systemctl reload firewalld`

```
[root@localhost ~]# sudo systemctl reload firewalld
[root@localhost ~]#
[root@localhost ~]#
```

8. 安裝 Postfix 以傳送通知郵件

安裝命令：`sudo yum install postfix`

```

[root@localhost ~]# sudo yum install postfix
已加载插件: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirrors.aliyun.com
 * extras: mirrors.aliyun.com
 * updates: mirrors.aliyun.com
软件包 2:postfix-2.10.1-6.el7.x86_64 已安装并且是最新版本
无须任何处理
[root@localhost ~]#

```

9. 將 postfix 服務設定成開機自啟動

安裝命令：`sudo systemctl enable postfix`

10. 啟動 postfix

安裝命令：`sudo systemctl start postfix`

```

[root@localhost ~]# sudo systemctl enable postfix
[root@localhost ~]# sudo systemctl start postfix
[root@localhost ~]#

```

在安裝 Postfix 期間，可能會出現組態螢幕。選擇“Internet Site”並按 enter 鍵。使用您的伺服器的外部 DNS 以“mail name”並按 enter。如果出現額外的螢幕，繼續按 enter 鍵接受預設值。

11. wget 用於從外網上下載外掛

檢查系統中是否已經安裝 wget，使用命令若出現下圖 wget 相關版本描述則說明系統中已經安裝 wget 若報系統找不到命令說明 wget 未安裝

```
[root@localhost ~]# wget -V
GNU Wget 1.14 在 linux-gnu 上编译。

+digest +https +ipv6 +iri +large-file +nls +ntlm +opie +ssl/openssl

Wgetrc:
  /etc/wgetrc (系统)
字符集: /usr/share/locale
编译: gcc -DHAVE_CONFIG_H -DSYSTEM_WGETRC="/etc/wgetrc"
      -DLOCALEDIR="/usr/share/locale" -I. -I../lib -I../lib -O2 -g -pipe
      -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fstack-protector-strong
      --param=ssp-buffer-size=4 -grecord-gcc-switches -m64 -mtune=generic
链接程序: gcc -O2 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions
          -fstack-protector-strong -param=ssp-buffer-size=4
          -grecord-gcc-switches -m64 -mtune=generic -lssl -lcrypto
```

若 wget 未安裝則進行安裝，安裝命令：`yum -y install wget`

```
[root@localhost ~]# yum -y install wget
已加载插件: fastestmirror
base
extras
https://packages.gitlab.com/gitlab/gitlab-ce/el/7/x86_64/repo/da
-accelerate.amazonaws.com/7/8/el/7/x86_64/repo/da/6b03db6762ed2
AccessKeyId=AKIAJ74R7IHMTQVGFCEA&Signature=8lkMGH7Va3pKvfHnouLCC
milliseconds with 0 out of 0 bytes received')
正在尝试其它镜像。
```

12. 安裝 vim 編輯器

安裝命令：`yum install vim -y`

二、新增 GitLab 鏡像源並安裝 gitlab 伺服器

1. 新增 gitlab 鏡像

`wget https://mirrors.tuna.tsinghua.edu.cn/gitlab-ce/yum/el7/gitlab-ce-10.0.0-ce.0.el7.x86_64.rpm`

```
[root@localhost ~]# wget https://mirrors.tuna.tsinghua.edu.cn/gitlab-ce/yum/el7/gitlab-ce-10.5.7-ce.0.el7.x86_64.rpm
--2018-04-19 17:20:44-- https://mirrors.tuna.tsinghua.edu.cn/gitlab-ce/yum/el7/gitlab-ce-10.5.7-ce.0.el7.x86_64.rpm
正在解析主机 mirrors.tuna.tsinghua.edu.cn (mirrors.tuna.tsinghua.edu.cn)... 101.6.8.193, 2402:f000:1:408:8100::1
正在连接 mirrors.tuna.tsinghua.edu.cn (mirrors.tuna.tsinghua.edu.cn)|101.6.8.193|:443... 已连接。
已发出 HTTP 请求，正在等待回应... 200 OK
长度: 427374190 (408M) [application/x-redhat-package-manager]
正在保存至: "gitlab-ce-10.5.7-ce.0.el7.x86_64.rpm"

6% [====>] https://mirrors.tuna.tsinghua.edu.cn/gitlab-ce/yum/el7/gitlab-ce-10.5.7-ce.0.el7.x86_64.rpm 927KB/s 剩余 7m 15s
```

2. 安裝 gitlab

安裝命令：`rpm -i gitlab-ce-10.0.0-ce.0.el7.x86_64.rpm`

安裝過程需要些時間，如果出現下圖，則說明安裝成功。（個人在安裝時並未出現，但是也是成功的）

[illegible]

4. 修改 gitlab 組態檔案指定伺服器 ip 和自訂連接埠：

```
vim /etc/gitlab/gitlab.rb
```



```
1 ## Latest options listed at https://gitlab.com/gitlab-org/omnibus-gitlab/blob/master/files/gitlab-
  config-template/gitlab.rb.template
2
3 ## Url on which GitLab will be reachable.
4 ## For more details on configuring external url see:
5 ## https://gitlab.com/gitlab-org/omnibus-gitlab/blob/629def0a7a26e7c2326566f0758d4a27857b52a3/READ
  ME.md#configuring-the-external-url-for-gitlab
6 external_url 'http://localhost:8080'
```

進入編輯器後按“i”鍵進入編輯狀態，修改完畢後，按 ESC 鍵退出編輯狀態

然後退出並保存，命令輸入“:wq”

ps:注意這裡設定的連接埠不能被佔用，默認是 8080 連接埠，如果 8080 已經使用，請自訂其它連接埠，並在防火牆設定開放相對應得連接埠

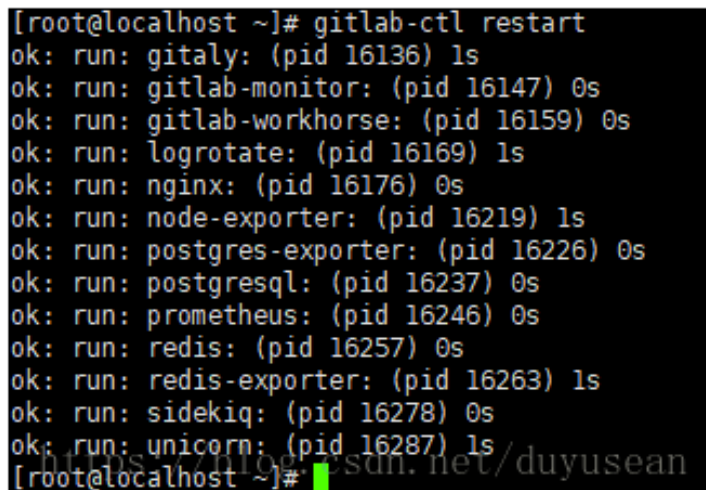
5.重設並啟動 GitLab

執行：

```
gitlab-ctl reconfigure
```

```
gitlab-ctl restart
```

提示 "ok: run:"表示啟動成功



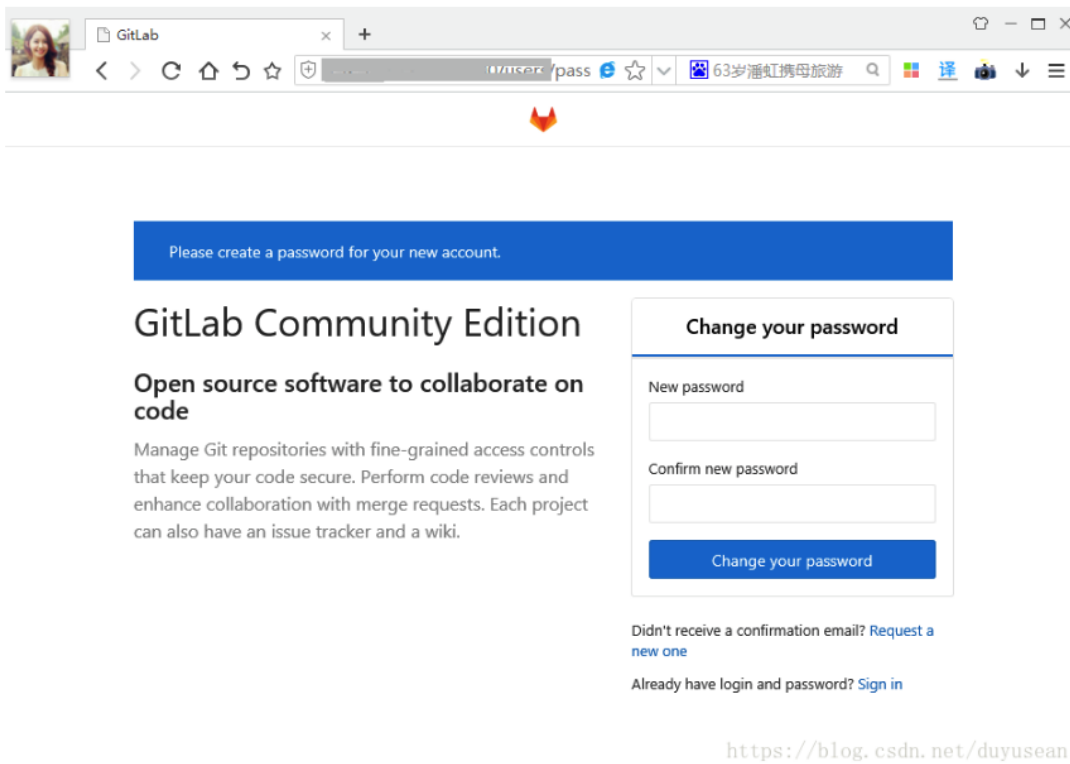
```
[root@localhost ~]# gitlab-ctl restart
ok: run: gitlab-ctl: (pid 16136) ls
ok: run: gitlab-monitor: (pid 16147) 0s
ok: run: gitlab-workhorse: (pid 16159) 0s
ok: run: logrotate: (pid 16169) 1s
ok: run: nginx: (pid 16176) 0s
ok: run: node-exporter: (pid 16219) 1s
ok: run: postgres-exporter: (pid 16226) 0s
ok: run: postgresql: (pid 16237) 0s
ok: run: prometheus: (pid 16246) 0s
ok: run: redis: (pid 16257) 0s
ok: run: redis-exporter: (pid 16263) 1s
ok: run: sidekiq: (pid 16278) 0s
ok: run: unicorn: (pid 16287) 1s
[root@localhost ~]#
```

6.訪問 GitLab 頁面

如果沒有域名，直接輸入伺服器 ip 和指定連接埠進行訪問

初始帳戶: root 密碼:5iveL!fe

第一次登錄修改密碼



7. 為了安全性考慮，需要建立 public key

建立 key

```
[root@git-node1 demo]# ssh-keygen #一路 Enter
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
48:94:9a:65:cd:0f:f3:17:c6:dc:3c:28:0a:bb:47:98 root@git-node1
The key's randomart image is:
+--[ RSA 2048 ]-----+
|      .+   o +      |
|      .= = . * +    |
|      =. = * o . .   |
|      o.E.o o .     |
```



```

|      .os  .      |
|      . .      |
|      .      |
|      |      |

```

8. 複製 id_rsa.pub 公鑰

```
[root@git-node1 demo]# cat ~/.ssh/id_rsa.pub
```

9. 新增公鑰至 gitlab，如圖 1-6-1

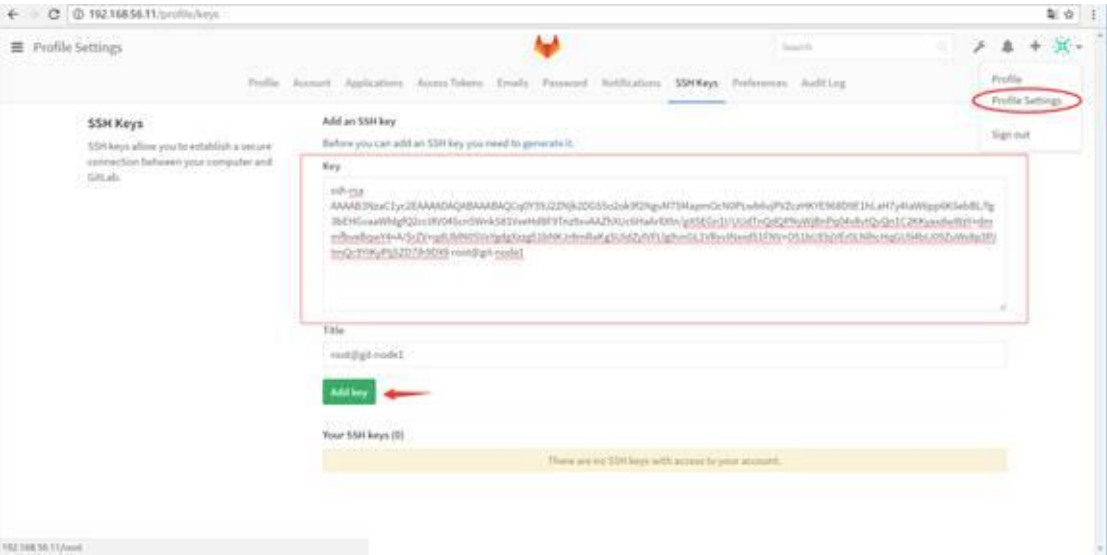


圖 1-6-1 新增伺服器公鑰

三、新增遠端倉庫

1. gitlab 建立倉庫，進行遠端同步

如圖 1-6-2

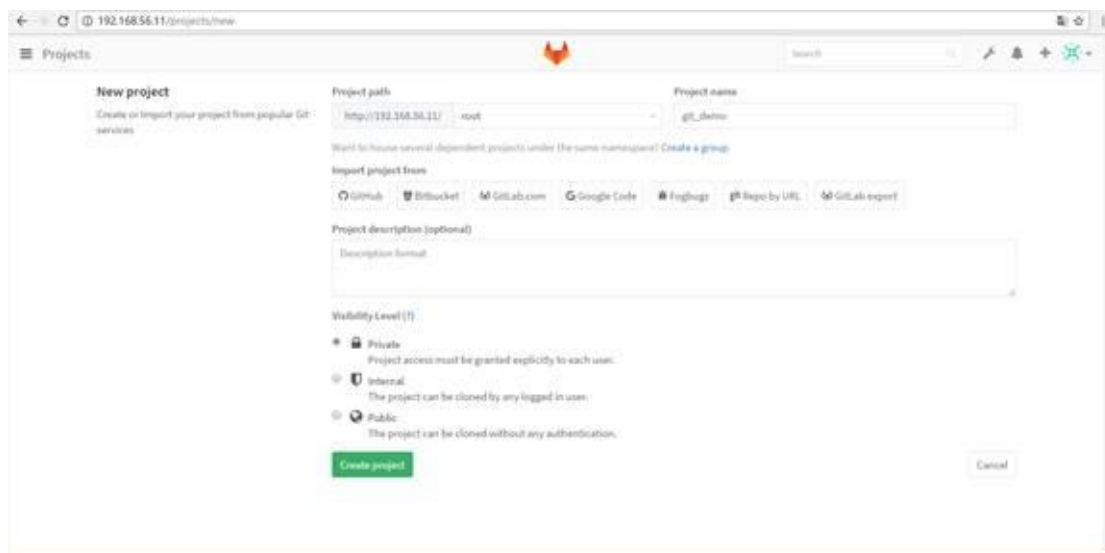


圖 1-6-2gitlab 建立遠端倉庫

2. 使用 git remote 新增遠端倉庫地址,選擇 SSH 方式克隆。

```
[root@ ~]# cd demo/ //必須是 git init 初始化倉庫目錄
```

```
[root@ demo]# git remote add origin git@git-node1:root/git_demo.gitxxx
```

四、修改遠端倉庫

由於剛開始新增的遠端倉庫寫錯了 url,現在通過如下命令進行 url 修改

```
[root@ demo]# git remote set-url origin git@git-node1:root/git_demo.git
```

五、查看遠端倉庫

如果已經組態了遠端倉庫伺服器，可以運行 git remote 命令。它會列出你指定每一個遠端伺服器的簡寫。

```
[root@git-node1 demo]# git remote  
Origin
```

也可以指定 -v 選項，會顯示需要讀寫遠端倉庫 git 保存簡寫名稱以及對應的 URL 地址。

```
[root@git-node1 demo]# git remote -v  
origin git@git-node1:root/git_demo.git (fetch)  
origin git@git-node1:root/git_demo.git (push)
```

六、推送遠端倉庫

將本地庫更新內容推送至遠端，用 `git push` 命令，實際上是將當前分支推送至遠端倉庫。

由於遠端庫是新建立空的，我們在第一次推送時候，`git` 默認是不會把本地 `master` 關聯至遠端的 `master`，所以我們需要加上 `-u` 參數，這樣 `git` 不但會把本地的 `master` 分支內容推送至遠端倉庫的 `master` 分支，並且還會將本地的 `master` 分支和遠端 `master` 分支關聯起來。在以後推送或者拉取時就可以簡化命令。

```
[root@git-node1 demo]# git push -u origin master
Counting objects: 5, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (5/5), 432 bytes | 0 bytes/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To git@git-node1:root/git_demo.git
* [new branch] master -> master
```

分支 `master` 設定為跟蹤來自 `origin` 的遠端分支 `master`。

如果推送衝突可以選擇 `--force` 強行推送

```
[root@git-node1 xuliangwei]# git push origin --force
```

如果一次都沒有推送資料，可以選擇 `--all` 一次全部推送至遠端伺服器

```
[root@git-node1 xuliangwei]# git push origin --all
```

七、克隆遠端倉庫

如果現在倉庫已經有開發好的項目，需要加入進來開發，可以先 `clone` 整個項目。

```
[root@git-node1 tmp]# git clone git@git-node1:root/git_demo.git
Clone to 'git_demo'...
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 5 (delta 0), reused 0 (delta 0)
receive: 100% (5/5), done.
```

八、拉取遠端倉庫

簡單的說，這個命令會訪問遠端倉庫，從中取出你還沒有的資料，或者 `git pull` 之後還是沒有的資料。

此前在新增的遠端倉庫的時候指定了倉庫名 `origin`，命令會自動將其新增為遠端倉庫並默認以

`origin` 為簡寫。

所以，`git fetch origin` 相當於從遠端獲取最新版本到本地，然後比較本地 `master` 分支和遠端 `master` 分支差別最後進行合併。

```
[root@git-node1 demo]# git fetch origin //拉取主分支最新版本(可以拉取其他分支)
```

```
[root@git-node1 demo]# git fetch origin dev //獲取 dev 分支最新資料
```

拉取資料，在生產環境中見到比較多的還是 `git pull` 相當於是從遠端獲取最新版本並 `merge` 到本地

```
[root@ xuliangwei]# git pull origin master #拉取主分支最新版本(可以拉取其他分支)
```

```
[root@ xuliangwei]# git pull origin dev //獲取 dev 分支最新資料
```

上述命令其實相當於 `git fetch` 和 `git merge` 在實際使用中，`git fetch` 更安全一些，因為在 `merge` 前，我們可以查看更新情況，然後再決定是否合併

九、更改遠端倉庫

如果想重新命名一個遠端倉庫名稱。將 `test` 重新命名為 `rainbow`, 可以通過 `git remote rename` 進行修改。

注意：這同時會修改你的遠端分支名字。之前引用 `test/master` 的現在會引用 `rainbow/master`

1. 新增新遠端分支，並賦予 `test` 為遠端倉庫名稱

```
[root@git-node1 git_demo]# git remote add test git@git-node1:root/git_demo.git
```

```
[root@git-node1 git_demo]# git remote -v
origin git@git-node1:root/git_demo.git (fetch)
origin git@git-node1:root/git_demo.git (push)
test git@git-node1:root/git_demo.git (fetch)
test git@git-node1:root/git_demo.git (push)
```

2. 修改 `test` 名稱為 `rainbow` 名稱

```
[root@git-node1 git_demo]# git remote rename test rainbow
[root@git-node1 git_demo]# git remote -v
origin git@git-node1:root/git_demo.git (fetch)
origin git@git-node1:root/git_demo.git (push)
rainbow git@git-node1:root/git_demo.git (fetch)
rainbow git@git-node1:root/git_demo.git (push)
```

十、移除遠端倉庫

因為一些變動不再使用一些特定的鏡像，可以通過 `git remote remove` 遠端倉庫名稱，移除遠端倉庫

1. 查看遠端倉庫

```
[root@git-node1 git_demo]# git remote -v
origin git@git-node1:root/git_demo.git (fetch)
origin git@git-node1:root/git_demo.git (push)
rainbow git@git-node1:root/git_demo.git (fetch)
rainbow git@git-node1:root/git_demo.git (push)
```

2. 移除不再使用的 rainbow 遠端倉庫

```
[root@git-node1 git_demo]# git remote remove rainbow
[root@git-node1 git_demo]# git remote -v
origin git@git-node1:root/git_demo.git (fetch)
origin git@git-node1:root/git_demo.git (push)
```

十一、Git 遠端倉庫小結

要新增一個倉庫，首先必須知道倉庫的地址，然後使用 `git remote add` 命令新增遠端倉庫，也可使用 `git clone` 命令克隆。（Git 支援多種協議，包括 `http`、`https`，但通過 `ssh` 支援的原生 `git` 協議速度最佳。）

要關聯一個遠端庫，使用命令 `git remote add origin git@server-name:path/repo-name.git`，關聯後，使用命令 `git push -u origin master` 第一次推送 `master` 分支的所有內容，此後，每次本地提交後，只要有必要，就可以使用命令 `git push origin master` 推送最新修改

```
# git remote add [remote] [url] #新增(關聯)遠端庫
# git remote set-url [remote] [url] #修改遠端倉庫
# git clone [url] #克隆遠端倉庫項目
# git remote #查看指定遠端倉庫命名簡寫
# git remote -v #查看遠端倉庫詳細資訊以及名稱對應 URL
# git push -u remote master #第一次推送 master 分支的所有內容
# git fetch remote [branch/tag] #下載遠端倉庫的所有變動
```

```
# git pull remote [branch/tag] #拉取主分支最新版本(可以拉取其他分支)
# git push remote [branch/tag] --force #強行推送當前分支至遠端分支,及時衝突
# git push remote [branch/tag] --all #推送所有分支到遠端倉庫
# git remote rename [oldname] [newname] #修改遠端倉庫名稱
# git remote remove [name] #刪除遠端倉庫名稱以及 URL 地址
```

十二、安裝過程遇見的那些坑

在 CentOS 裡面安裝軟體,提示軟體已安裝,但是 `rpm -q` 和 `-e` 都是提示包沒有安裝

查看與 `rpm` 包相關的檔案和其他資訊

`$ rpm -qa | grep 包名`

```
[root@localhost ~]# rpm -qa | grep gitlab-ce-10.0.0-ce.0.el7.x86_64
gitlab-ce-10.0.0-ce.0.el7.x86_64
```

查詢包是否被安裝

命令：`rpm -q 包名`

```
[root@localhost ~]# rpm -q gitlab-ce-10.0.0-ce.0.el7.x86_64
gitlab-ce-10.0.0-ce.0.el7.x86_64
```

刪除軟體包

命令：`rpm -e 包名`

```
[root@localhost ~]# rpm -e gitlab-ce-10.0.0-ce.0.el7.x86_64
```

運行以上三步,把原來的包刪除掉重新下載和安裝