Create activity in Git

In this chapter, we will learn how to create a remote git repository, from which we will mention it as a Git server. We need a Git server to allow the team to collaborate.

In this chapter, we will learn how to create a remote git repository, from which we will mention it as a Git server. We need a Git server to allow the team to collaborate.

Create a new user account on the operating system (here I use Linux)

```
# create a new user group in Linux.  We will create a new account in git to pra
root @ ubuntu: ~ groupadd dev

# create a new user in Linux
root @ ubuntu: ~ # useradd -G dev -d / home / gituser -m -s / bin / bash gituse

# change the password of gituser account
root @ ubuntu: ~ # passwd gituser
```

The above command will produce the following result:

```
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

So we have created a gituser account in the dev group on Linux operating system. Now I ssh on the server again with the new account created by the command ssh gituser@10.10.10.10 (assuming my ip is 10.10.10.10)

Create an empty repository

Let us launch a new repository using the init command followed by the --bare option. It launches the repository which is not a working directory. By convention, this empty repository must be named as .git.

```
gituser @ ubuntu: ~ $ pwd
/ home / gituser

gituser @ ubuntu: ~ $ mkdir QTM.git

gituser @ ubuntu: ~ / QTM.git $

gituser @ ubuntu: ~ / QTM.git $ ls

gituser @ ubuntu: ~ / QTM.git $ git --bare init
```

```
  Initialized empty Git repository in /home/gituser/QTM.git/

  gituser @ ubuntu: ~ / QTM.git $ ls
  branches config description HEAD hooks info objects refs
```

Public / private key pair

Let's take a look at the process of configuring a Git server, the public ssh-keygen / private RSA utility, which we'll use to identify the user.

Open a terminal and enter the following command and press enter for each entry. Once completed, it will create a .ssh folder inside the home directory. I created this key on the client computer (root account), not git server with gituser account

```
  root @ ubuntu: ~ # pwd
  / root

  root @ ubuntu: ~ # ssh-keygen
```

The above command will produce the following result:

```
Generating public/private rsa key pair. Enter file in which to save the key (/roo
 Press Enter Only
 Created directory '/home/tom/.ssh'. Enter passphrase (empty for no passphrase):
---------------> Press Enter Only Enter same passphrase again:
------------------------------> Press Enter Only
 Your identification has been saved in /home/tom/.ssh/id_rsa. Your public key has
```

ssh-keygen has created two keys, first is private (ie, id_rsa) and second is public (ie, id_rsa.pub).

Note : You should never share your Private key with anyone else.

Add keys to authorized keys

Suppose there are two programmers working on a project, named Tom and Jerry. Both generate public keys. Let's see how to use these keys to confirm.

Tom enters his public key to the server using the ssh-copy-id command as follows:

```
  [tom @ CentOS ~] $ pwd
  / home / tom

  [tom @ CentOS ~] $ ssh-copy-id -i ~ / .ssh / id_rsa.pub gituser@git.server.com
```

The above command will produce the following result:

```
  gituser@git.server.com's password:
  Hãy th? ??ng nh?p vào máy ph?c v?, v?
i "ssh 'gituser@git.server.com'", and check in:
  .ssh / authorized_keys
```

```
  ?? t?o sure b?n không thêm ???c khóa thêm mà b?n không th? ki?m k?t.
```

In the same way, Jerry enters the public key to the server using the ssh-copy-id command.

```
  [jerry @ CentOS ~] $ pwd
  / home / jerry

  [jerry @ CentOS ~] $ ssh-copy-id -i ~ / .ssh / id_rsa gituser@git.server.com
```

The above command will produce the following result:

```
  gituser@git.server.com's password:
  Hãy th? ??ng nh?p vào máy ph?c v?, v?
i "ssh 'gituser@git.server.com'", and check in:
  .ssh / authorized_keys
  ?? t?o sure b?n không thêm ???c khóa thêm mà b?n không th? ki?m k?t.
```

Push (Push) the changes to the repository

We have created an empty repository on the server and allow access for two people. From there, Tom and Jerry can push their changes to the repository by adding it as a remote control.

The init command creates a .git directory to keep the metadata about the repository every time it reads the configuration from the .git / config file.

Tom creates a new directory, adds his README, and commit changes as the first commit. After signing, he verifies the commit messages by running git log.

```
  [tom @ CentOS ~] $ pwd
  / home / tom

  [tom @ CentOS ~] $ mkdir tom_repo

  [tom @ CentOS ~] $ cd tom_repo /

  [tom @ CentOS tom_repo] $ git init
  Initialized empty Git repository in /home/tom/tom_repo/.git/

  [tom @ CentOS tom_repo] $ echo 'TODO: Add contents for README'> README

  [tom @ CentOS tom_repo] $ git status -s
  ??  README

  [tom @ CentOS tom_repo] $ git add.

  [tom @ CentOS tom_repo] $ git status -s
  A README

  [tom @ CentOS tom_repo] $ git commit -m 'Initial commit'
```

The above command creates the following result:

```
[master (root-commit) 19ae206] Initial commit
1 files changed, 1 insertions (+), 0 deletions (-)
create mode 100644 README
```

Tom checks the log message by executing the git log command.

```
[tom @ CentOS tom_repo] $ git log
```

The above command will produce the following result:

```
commit 19ae20683fc460db7d127cf201a1429523b0e319
Author: Tom Cat
Date: Wed Sep 11 07:32:56 2013 +0530

Initial commit
```

His commit changes to the local repository. Now is the time to push the changes to the remote repository. But before that, we must add this repository as a remote control, this is a one-time operation. Then he can safely push these changes to the remote repository.

Note : By default, Git only pushes to connected branches: for each branch that exists on the internal side, the remote control side is updated if a branch with the same name already exists on it. In our tutorial, every time we push changes to the origin master branch, we use the branch name exactly.

```
[tom @ CentOS tom_repo] $ git remote add origin gituser@git.server.com: projec

[tom @ CentOS tom_repo] $ git push origin master
```

The above command will produce the result:

```
Counting objects: 3, done.
Writing objects: 100% (3/3), 242 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To gituser@git.server.com: project.git
* [new branch]
master -> master
```

Now, these changes have been successfully committed to the remote control repository.

According to Tutorialspoint

Previous article: Git life cycle

Next lesson: Clone activity in Git