Git

Previous Next

Git is an open source distributed version control system originally developped by Linus Torvalds to support the development of the linux kernel. Every Git working directory is a full-fledged repository with complete history and full version tracking capabilities, not dependent on network access or a central server.

Installation

The *git* version control system is installed with the following command

```
sudo apt install git
```

Configuration

Every git user should first introduce himself to git, by running these two commands:

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

Basic usage

The above is already sufficient to use git in a distributed and secure way, provided users have access to the machine assuming the server role via SSH. On the server machine, creating a new repository can be done with:

```
git init --bare /path/to/repository
```

> This creates a bare repository, that cannot be used to edit files directly. If you would rather have a working copy of the contents of the repository on the server, ommit the *--bare* option.

Any client with SSH access to the machine can then clone the repository with:

```
git clone username@hostname:/path/to/repository
```

Once cloned to the client's machine, the client can edit files, then commit and share them with:

```
cd /path/to/repository
#(edit some files
git commit -a # Commit all changes to the local version of the repository
git push origin master # Push changes to the server's version of the repository
```

Installing a gitolite server

While the above is sufficient to create, clone and edit repositories, users wanting to install git on a server will most likely want to have git work like a more traditional source control management server, with multiple users and access rights management. The suggested solution is to install *gitolite* with the following command:

```
sudo apt install gitolite
```

Gitolite configuration

Configuration of the *gitolite* server is a little different that most other servers on Unix-like systems. Instead of the traditional configuration files in /etc/, gitolite stores its configuration in a git repository. The first step to configuring a new installation is therefore to allow access to the configuration repository.

First of all, let's create a user for gitolite to be accessed as.

```
sudo adduser --system --shell /bin/bash --group --disabled-password --home /home/git git
```

Now we want to let gitolite know about the repository administrator's public SSH key. This assumes that the current user is the repository administrator. If you have not yet configured an SSH key, refer to SSH Keys

```
cp ~/.ssh/id_rsa.pub /tmp/$(whoami).pub
```

Let's switch to the git user and import the administrator's key into gitolite.

```
sudo su - git
gl-setup /tmp/*.pub
```

Gitolite will allow you to make initial changes to its configuration file during the setup process. You can now clone and modify the gitolite configuration repository from your administrator user (the user whose public SSH key you imported). Switch back to that user, then clone the configuration repository:

```
exit
git clone git@$IP_ADDRESS:gitolite-admin.git
cd gitolite-admin
```

The gitolite-admin contains two subdirectories, "conf" and "keydir". The configuration files are in the conf dir, and the keydir directory contains the list of user's public SSH keys.

Managing gitolite users and repositories

Adding new users to gitolite is simple: just obtain their public SSH key and add it to the keydir directory as $DESIRED_USER_NAME.pub. Note that the gitolite usernames don't have to match the system usernames - they are only used in the gitolite configuration file to manage access control. Similarly, users are deleted by deleting their public key file. After each change, do not forget to commit the changes to git, and push the changes back to the server with

```
git commit -a
git push origin master
```

Repositories are managed by editing the conf/gitolite.conf file. The syntax is space separated, and simply specifies the list of repositories followed by some access rules. The following is a default example

```
repo    gitolite-admin
        RW+     =   admin
        R       =   alice

repo    project1
        RW+     =   alice
        RW      =   bob
        R       =   denise
```

Using your server

To use the newly created server, users have to have the gitolite admin import their public key into the gitolite configuration repository, they can then access any project they have access to with the following command:

```
git clone git@$SERVER_IP:$PROJECT_NAME.git
```

Or add the server's project as a remote for an existing git repository:

```
git remote add gitolite git@$SERVER_IP:$PROJECT_NAME.git
```

Previous Next