Basic about Git

Git is the name of a distributed version management system (Distributed Version Control System - DVCS) is one of the most popular distributed version management systems today.

VCS - version management system

Version Control System (VCS) is a software that helps software programmers work together and maintain a complete history of the work they have done.

Here are the functions of a VCS:

Allow software developers to work together

Do not allow overwriting each other's changes

Maintain a history of all versions.

Here are the types of VCS:

Centralized version control system (CVCS).

Control / distribution system (DVCS).

In this chapter we will focus only on distributed version management systems and specifically on Git.

Control version distribution system

Centralized version control system (CVCS) uses a server to store all files and allow teams to collaborate. But the biggest drawback of CVCS is also its failure point, that is, the failure of central servers. Unfortunately, if the central server is broken for an hour, during that time no one can collaborate with anyone. And even in the worst case, if the central server's disk is broken and backup is not done, you will lose the entire history of the project. Here, the distributed version management system appears.

DVCS clients not only check the latest snapshots of folders, but they also observe all the repository of the project. If the server is down, the repository of the client can copy a full copy to the server to restore it. Git does not depend on the central server and that is why you can perform many operations when you are offline. You can delegate changes, create branches, view logs and perform other operations when you are offline. You need to network only to publish your changes and bring the latest changes to the project.

So what is Git?

Git is the name of a distributed version management system (Distributed Version Control System - DVCS) is one of the most popular distributed version management systems today.

The advantages of Git

Free and open source

Git is published under the GPL open source license. It is available for free online. You can use Git to manage appropriate projects without paying any money. As an open source, you can download its source code and also make changes according to your requirements.

Fast and compact speed

When most operations are done internally, it brings great speed benefits. Git does not depend on the server, which is why it does not require interaction with the remote server (git server) for all operations. The core part of Git is written in C, which can avoid runtime-related costs of other high-level languages. Although Git reflects the entire repository, the size of the data on the client is small. This shows the effectiveness of Git in compressing and storing data on clients.

Backup (backup) hidden

Data loss is rare when there are many copies of it. Data is present at any client, so it can be used in case of failure or stop at the server.

High safety

Git uses a generic cryptographic hash function, called a secure hash function (SHA1), to name and identify objects in its database. Each file and commit is summarized and obtained results at the time of inspection. This implies that it cannot change files, dates and commit messages and any other data from the Git database without knowing Git.

Does not require a strong hardware

In the case of CVCS, the central server needs to be strong enough to serve the needs of the entire team. For small teams, it is not a problem, but when the team size grows, the limitations of the server hardware can make the performance of the work change in a negative direction. In the case of DVCS, developers do not interact with the server unless they need to announce changes that have been made. All things happen on the client, so the server hardware can really be a simple problem.

Branching easier

CVCS uses cheap copy technology, if we create a new branch, it will copy all the code to the new branch, so it's time-consuming and inefficient. Apart from us, deleting and merging of branches in CVCS is complicated and time consuming. But branch management with Git is very simple. It only takes a few seconds to create, delete,

and import branches.

Terms of DVCS

Internal commit

Each VCS tool provides a private workplace as a working copy. Developers made changes in their own workplace and after their commits, the changes became part of the repository. Git takes a further step by providing them with a separate copy of the entire repository. Users can perform many operations with this repository such as adding, moving, renaming files, changing commits and many other operations.

Working directory and Staging or Index

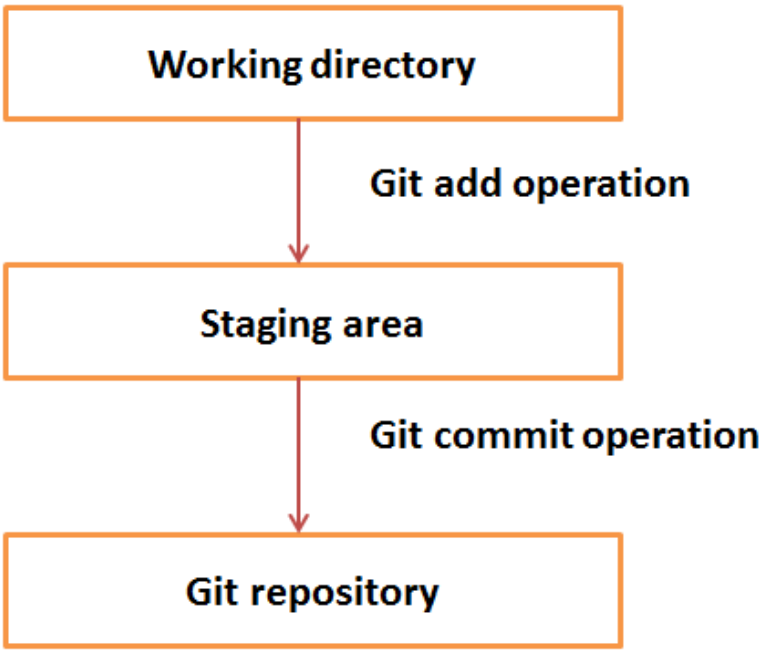
The working directory is where the files are checked. In CVCS, developers often make changes and commit their changes directly to the repository. But Git uses a different method. Git does not track each modified file. Whenever the user commits an operation, Git searches for files in the staging area. Only when the files present in this organization area are considered to commit, not all files are modified.

The following is Git's basic workflow:

Step 1 : You modify a file from the working directory

Step 2 : You add that file to the organizer area

Step 3 : You perform commit operations that move files from the organization area. After the push operation (push), it saves the fixed changes to the Git repository.



Suppose you modify two files, named sort.c and search.c and you want to make two different commits for each activity. You can add a file to the organizer area and commit. After the first commit, redo the same method for the other file.

```
# First commit
[bash] $ git add sort.c

# adds file to staging area
[bash] $ git commit -m "Added sort operation"

# Second commit
[bash] $ git add search.c

# adds file to staging area
[bash] $ git commit -m "Added search operation"
```

Blobs

Blob stands for Binary Large Object. Each version of a file is represented by blob. A blob contains file data but does not contain any file metadata. It is a binary file, and in the Git database, it is placed on the SHA1 hash of the file. In Git, files are not set by name. Everything is addressed by content.

Trees - Trees

Tree is an object, which represents a directory. It keeps blobs as well as other subfolders. A tree is a binary file that holds things related to blobs and trees that are also named SHA1 hash of the tree object.

Commits - Commits

Commit operation keeps the current state of the repository. A commit is also named SHA1 hash. You can consider a commit object as a node of the linked list. Each commit object has a pointer to the original commit object. From a given commit, you can go back by looking at the original cursor point to see the history of that commit. If a commit has multiple original commits, then the specific commits will be created by merging the two branches.

Branches - Branches

Branches are used to create other routes of development. By default, Git has a master branch, which is similar to trunk in Subversion. Usually, a branch is created to work on a new point. Once this point is completed, it is merged with the branch and we delete that branch. Each branch is implied by HEAD, which points to the latest commit in the branch. Whenever you make a commit, HEAD is updated by the latest commits.

- Tags

Tags indicate a meaningful name with a specific version in the repository. Forms are similar to branches, but the difference is that the tags cannot be changed. It means, tags are a branch, but no one intends to fix them. Once a tag is created for specific commits, even if you create a new commit, it will not be updated. Typically, developers create tags for product publication.

Simulation - Clone

Simulation operation creates a copy of the repository. The user can perform many operations with the local repository. This is the time when the internet network needs to be active when the repository instances are being synchronized from the remote repository.

Pull

Pull operation copies changes from a remote instance repository (git server) to the local repository. This operation is used to synchronize between two repository instances. This is similar to update operation in Subversion.

Push

Push (push) operation copies changes from local repository to a remote repository (on git server). It is used to save changes permanently in Git repository. It is similar to commit operation in Subversion.

HEAD

HEAD is a pointer, which usually points to the latest commit in the branch. Whenever you make a commit, HEAD is updated with the latest commit. The heads of branches are stored in .git / refs / heads / directory.

```
[CentOS] $ ls -1 .git / refs / heads /
master

[CentOS] $ cat .git / refs / heads / master
570837e7d58fa4bccd86cb575d884502188b0c49
```

Revision

Revision represents the version of a source code. Revisions in Git are represented by commits. These commits are determined by SHA1 secure hash.

URL

URL represents the location of the Git repository. Git URL is kept in a config file.

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

[tom @ CentOS tom_repo] $ cat .git / config
[core]
repositoryformatversion = 0
filemode = true
bare = false
logallrefupdates = true
[remote "origin"]
url = gituser@git.server.com: project.git
fetch = + refs / heads / *: refs / remotes / origin / *
```

According to Tutorialspoint

Previous article: Math function available in Shell

Next lesson: Git environment installation