**Git vs GitHub**

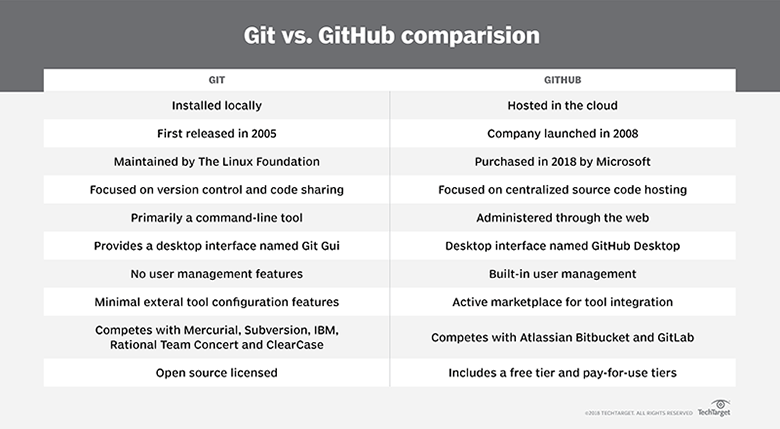
**An optional part for your extension reading**

**Git vs GitHub**

This course uses GitHub as a platform for team work service, so you have to mention git, which is very similar to GitHub.

Git is a distributed version control tool that developers should install locally on their computers and can manage the source code history of development projects, while GitHub is an online service for storing code pushed to it from the computer running the Git tool , Is a cloud-based platform built around Git tools.

The main difference between Git and GitHub is that Git is an open source tool that developers install locally to manage source code, while GitHub is an online service where developers using Git can connect and upload or download resources.



GitHub provides the following features that Git tools do not have:

* lightweight task tracking;
* the GitHub Desktop GUI tool for managing GitHub repositories;
* project wikis with extensive emoji support;
* online editing of files;
* gists for storing code snippets;
* a marketplace for third-party tool integration;
* branch protection rules;
* forking capabilities;
* organizational tools for managing teams;
* email notifications and alerting;
* secure shell and GNU Privacy Guard security and encryption key management;
* project insights regarding traffic, code commit frequency and dependency graphs;
* interaction limits for when arguments about software development heat up.

These are all features that Git does not have, but it is very important for enterprise software development teams.

GitHub repository URL

The GitHub URL uniquely identifies the Git repository.

After creating the repository, developers can install Git locally and then clone the repository. Developers can then make local changes to the project file and then push the submission back to GitHub. To obtain updates to the central GitHub repository, developers can use the pull and fetch commands.

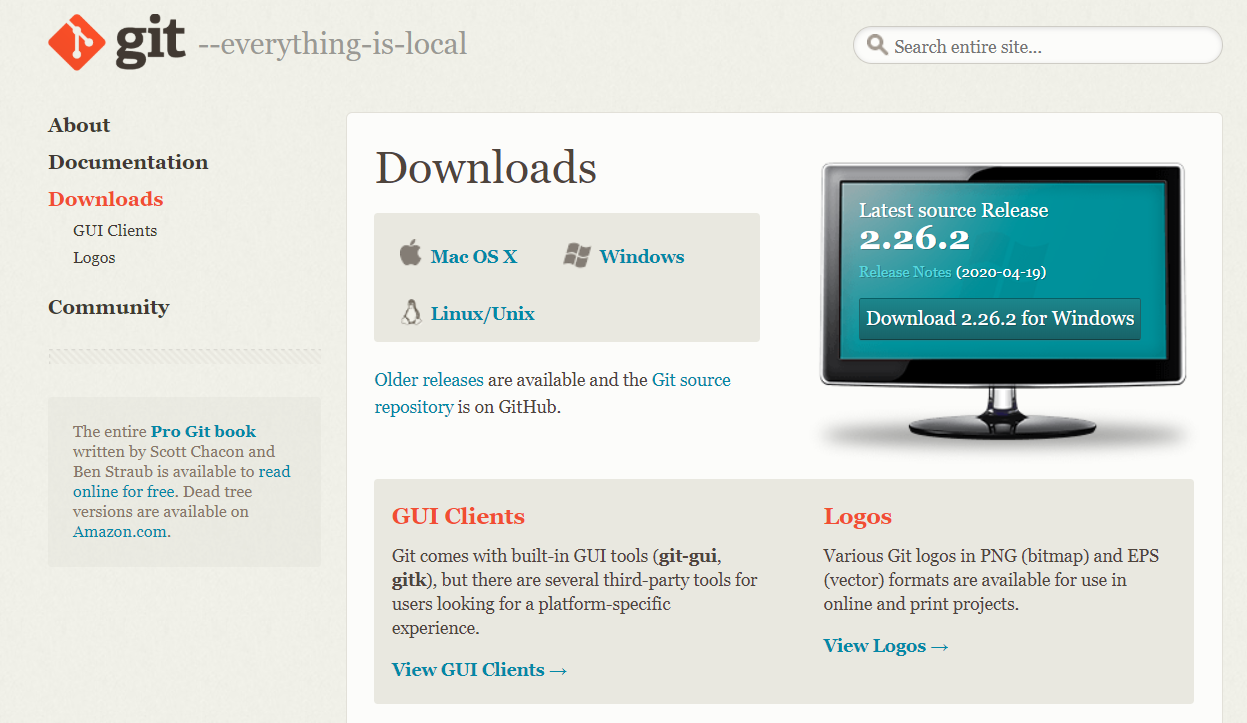
As you can see, there is no need to oppose these two tools. These two tools are complementary. This is not to argue about the advantages and disadvantages of Git and GitHub. After all, when Git and GitHub are configured to work together, the best DevOps integration can be achieved.

**Installing and Using Git on Your Machine**

Git is an open source distributed version control system, which can effectively and quickly handle version management from small to very large projects. The specific installation steps are as follows:

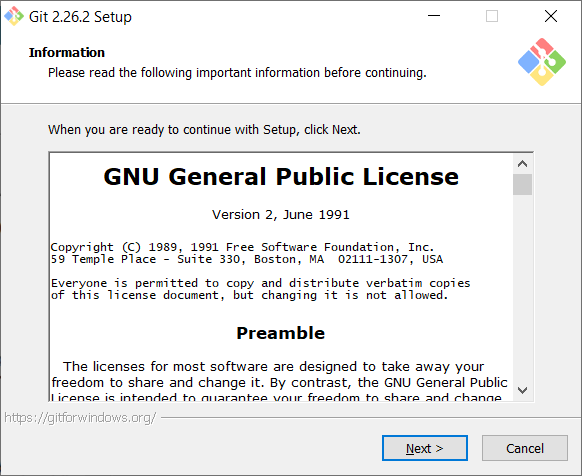
**Step 1:** Download the latest version of Git from the official website.

<https://git-scm.com/downloads>

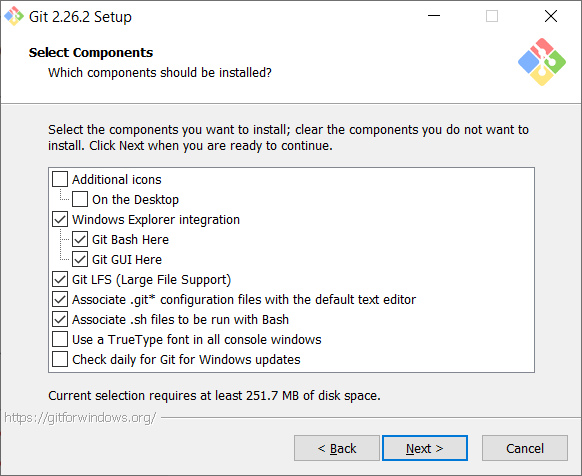


You can choose the appropriate version according to the computer's operating system.

**Step 2:** Double-click the downloaded git installation package, a prompt box will pop up, as shown below:

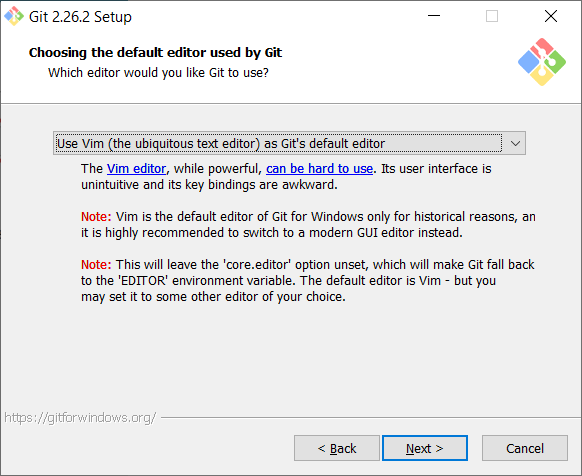


**Step 3:** Click "Next" directly to enter the next step, select the installation path, and after selecting the installation path, click "Next" to enter the next step, the installation configuration window pops up, including the git command line, git graphics window, etc., as shown below As shown:



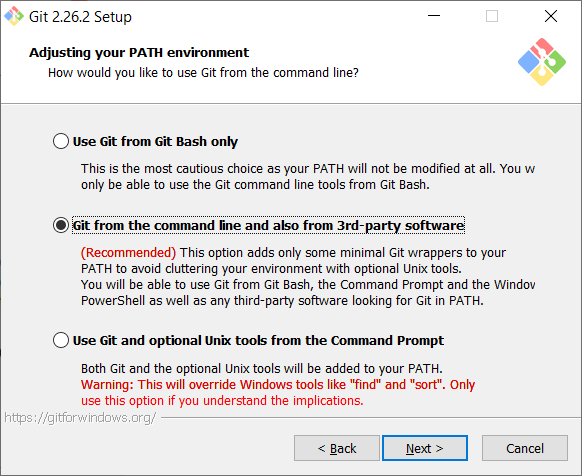
Basically, there is no need to change the default options. Of course, you can add the functions, if you need.

**Step 4:** Just follow the default path, click "Next" directly to enter the next step, enter the "Choose the default editor used by Git" window, as shown below:



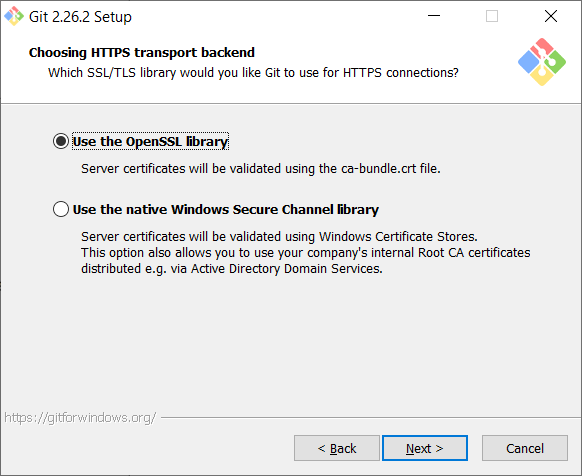
You can choose the editor according to your needs. If there is no special requirement, the default vim editor is generally recommended.

**Step 5:** Click "Next" to enter the next step and enter the "Adjust Path Environment Variables" window, as shown below:

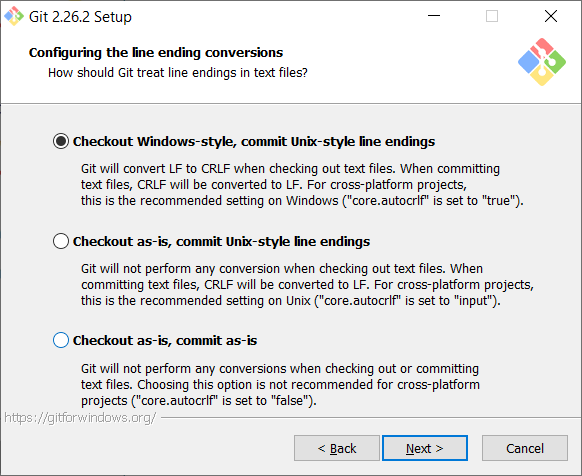


Basically, all steps can directly select the default options until the installation is complete. as shown below:

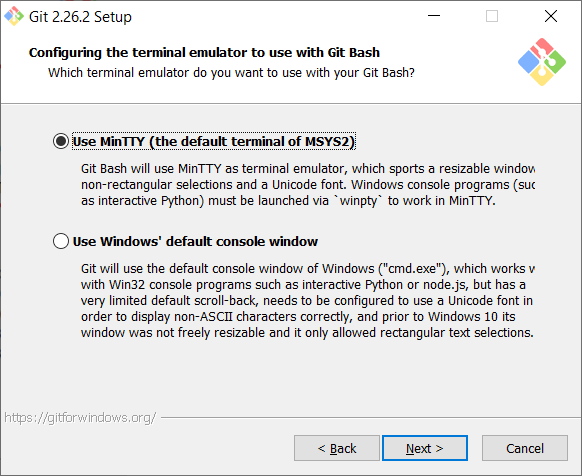
**Step 6:**



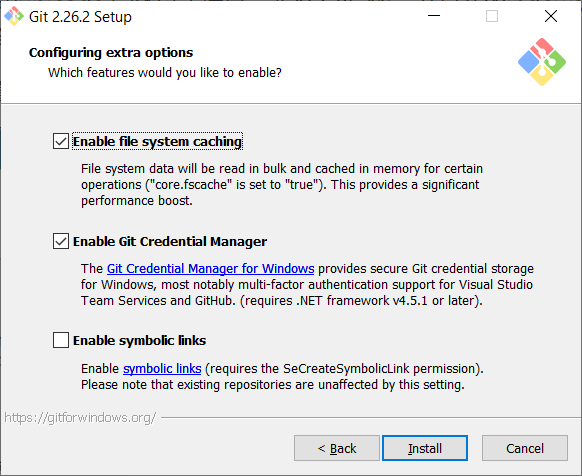
**Step 7:**



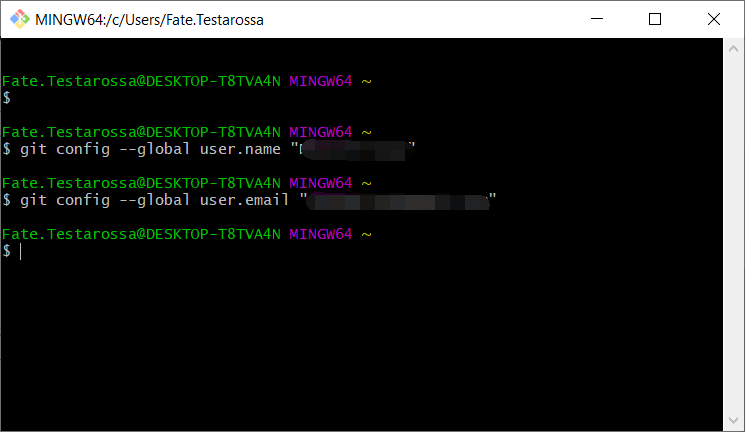
**Step 8:**



**Step 9:**



**Step 10:** After the installation is complete, open git bash, as shown below:

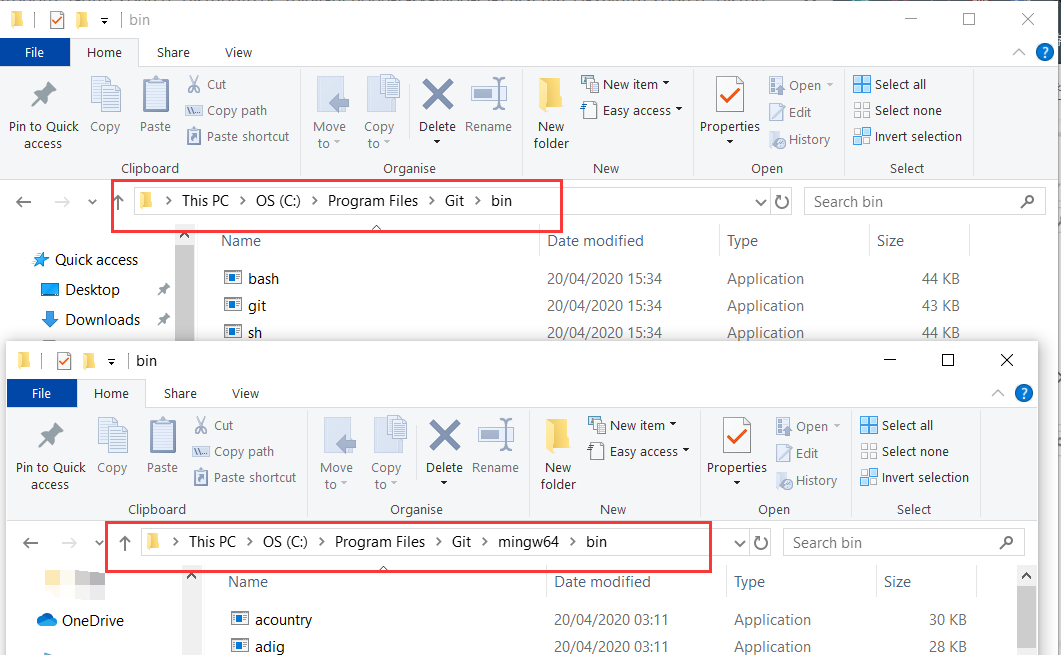


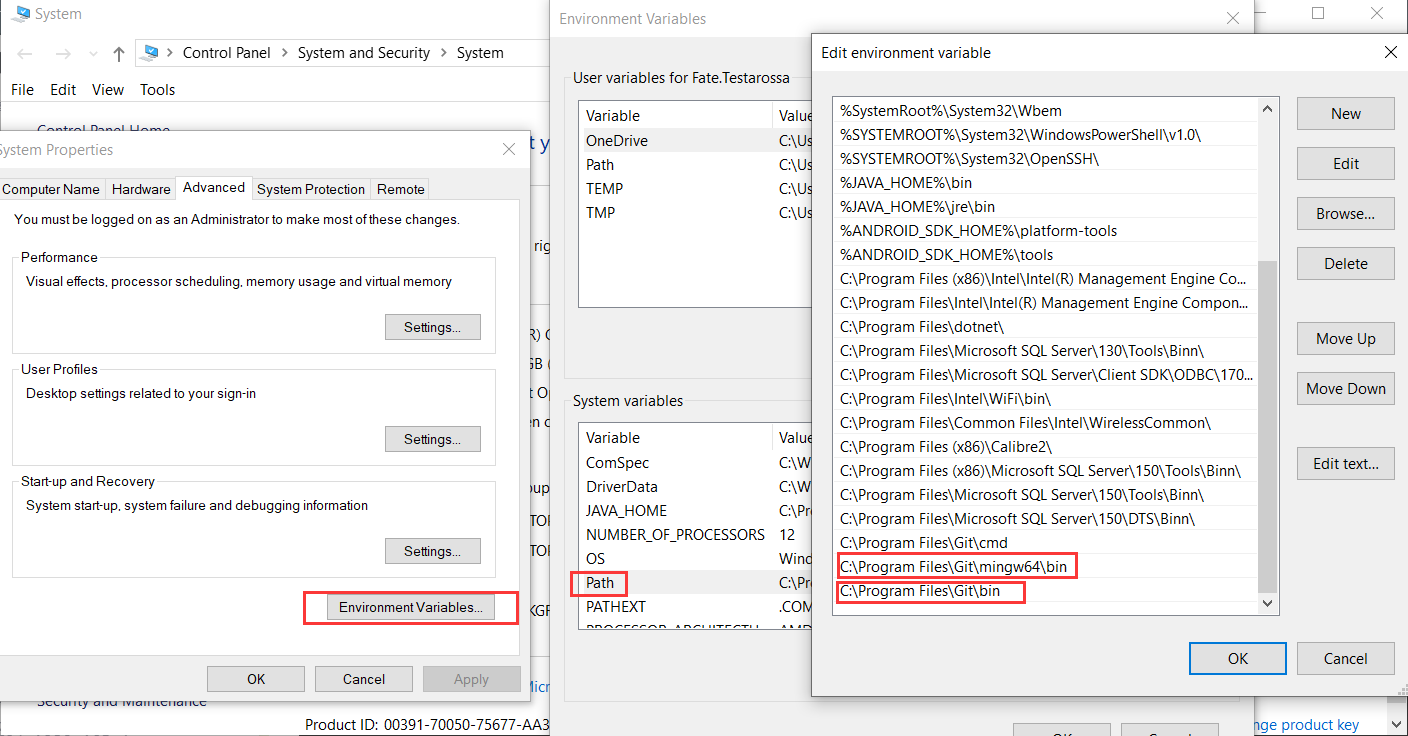
Enter your username and email when registering your GitHub account.

git config --global user.name "userName"

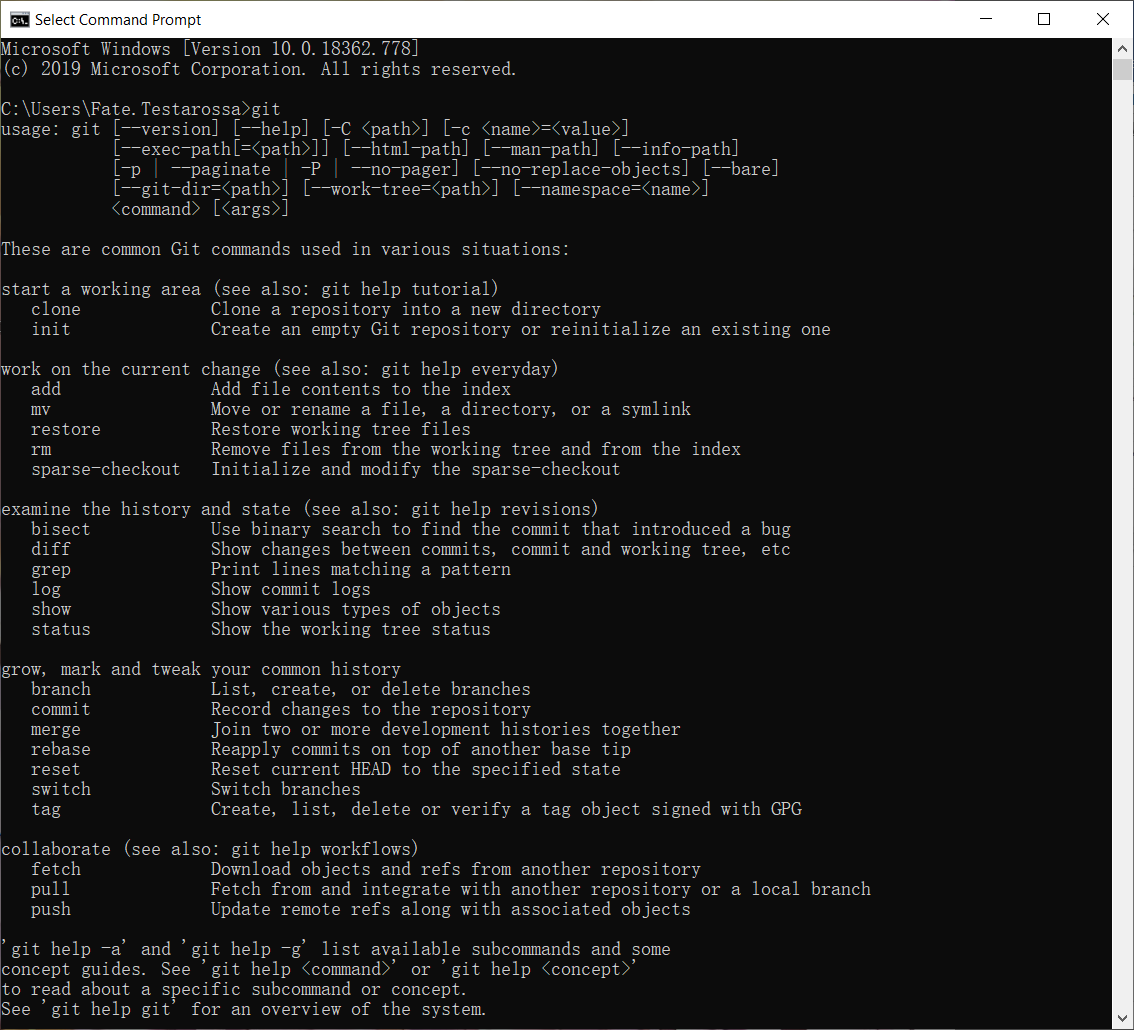
git config --global user.email "userEmail"

**Step 11:** configure the path environment





Finally run the command prompt and enter “git”



**Here is an instruction set for reference**

git init # Initialize local git repository (create new repository)

git config --global user.name "xxx" # Configure username

git config --global user.email "xxx@xxx.com" # Configure email

git config --global color.ui true # Command auto coloring

git config --global color.status auto

git config --global color.diff auto

git config --global color.branch auto

git config --global color.interactive auto

git config --global --unset http.proxy # remove proxy configuration on git

git clone git+ssh://git@192.168.53.168/VT.git # clone remote warehouse

git status # View the current version status (whether to modify）

git add xyz # Add xyz file to index

git add . # Add all changed files in the current subdirectory to index

git commit -m 'xxx' # submit

git commit --amend -m 'xxx' # Merge the last commit (for repeated modifications)

git commit -am 'xxx' # Combine add and commit in one step

git rm xxx # Delete files in index

git rm -r \* # Recursive delete

git log # Show commit log

git log -1 # Display 1 line log -n is n lines

git log -5

git log --stat # Show commit log and related change files

git log -p -m

git show dfb02e6e4f2f7b573337763e5c0013802e392818 # Show details of a submission

git show dfb02 # Only the first few of the commitid can be used

git show HEAD # Show HEAD commit log

git show HEAD^ # Show the commit log of HEAD's parent (previous version) ^^ is the last two versions ^ 5 is the last 5 versions

git tag # Show existing tags

git tag -a v2.0 -m 'xxx' # Add v2.0 tag

git show v2.0 # Show v2.0 logs and details

git log v2.0 # Show v2.0 logs

git diff # Show all changes not added to index

git diff --cached # Show all changes that have been added to the index but not yet committed

git diff HEAD^ # Compare the difference with the previous version

git diff HEAD -- ./lib # Compare the difference with the HEAD version lib directory

git diff origin/master..master # Compare the remote branch master with the local branch master

git diff origin/master..master --stat # Only show the difference files, not the specific content

git remote add origin git+ssh://git@192.168.53.168/VT.git # Add remote definition (for push / pull / fetch)

git branch # Show local branch

git branch --contains 50089 # Show branches with commit 50089

git branch -a # Show all branches

git branch -r # Show all original branches

git branch --merged # Show all branches that have been merged into the current branch

git branch --no-merged # Show all branches that have not been merged into the current branch

git branch -m master master\_copy # Local branch renamed

git checkout -b master master\_copy # Create a new branch master\_copy from the current branch and check it out

git checkout features/performance # Check out the existing features / performance branch

git checkout --track hotfixes/BJVEP933 # Check out the remote branch hotfixes / BJVEP933 and create a local tracking branch

git checkout v2.0

git checkout -b devel origin/develop # Create a new local branch devel from the remote branch develop and check it out

git checkout -- README # Check out the README file of the head version (can be used to modify error rollback）

git merge origin/master # Merge remote master branch to current branch

git cherry-pick ff44785404a8e # Combine and submit the changes of ff44785404a8e

git push origin master # Push the current branch to the remote master branch

git push origin :hotfixes/BJVEP933 # Delete the hotfixes / BJVEP933 branch of the remote warehouse

git push --tags # Push all tags to the remote warehouse

git fetch # Get all remote branches (do not update the local branch, another merge is required)

git fetch --prune # Get all original branches and clear the deleted branches on the server

git pull origin master # Get the remote branch master and merge to the current branch

git mv README README2 # Rename the file README to README2

git reset --hard HEAD # Reset the current version to HEAD (usually used for merge fail back)

git rebase

git branch -d hotfixes/BJVEP933 # Delete the branch hotfixes / BJVEP933 (this branch modification has been merged into other branches)

git branch -D hotfixes/BJVEP933 # Forcibly delete the branch hotfixes / BJVEP933

git ls-files # List files included in git index

git show-branch # Show current branch history

git show-branch --all # Show all branch history

git whatchanged # Show file modification corresponding to commit history

git revert dfb02e6e4f2f7 # Unsubmit dfb02e6e4f2f7

git ls-tree HEAD # Internal command: display a git object

git rev-parse v2.0 # Internal command: display SHA1 HASH for a ref

git reflog # Show all commits, including orphan nodes

git show HEAD@{5}

git show master@{yesterday} # Show the status of the master branch yesterday

git log --pretty=format:'%h %s' --graph # Graphical submission log

git show HEAD~3

git show -s --pretty=raw 2be7fcb476

git stash # Temporarily save the current modification and bring all to the HEAD state

git stash list # View all staging

git stash show -p stash@{0} # Refer to the first temporary storage

git stash apply stash@{0} # Application staging for the first time

git grep "delete from" # Search for the text "delete from" in the file

git grep -e '#define' --and -e SORT\_DIRENT

git gc

git fsck

**Reference**

1.Cameron McKenzie, TechTarget. Git vs. GitHub: What is the difference between them?

<https://www.theserverside.com/video/Git-vs-GitHub-What-is-the-difference-between-them>

2.<https://gist.github.com/guweigang/9848271>

3. <https://git-scm.com/docs>