Git Notes

Git Command Notes

To become a root user (applicable for Linux only):

sudo su -

To create a working directory:

mkdir myproject

To navigate into the directory:

cd myproject

To create empty files:

touch file1.java file2.java

To create a local repository:

git init

For Git configurations, use the following commands:

Set user name:

git config --global user.name malay

Set user email:

git config --global user.email malay@gmail.com

There are three types of configurations:

Config Variables: user.name and user.email

Local: If the configuration is set with a flag as local, the config variable will apply only to that particular repository.

Global: If the configuration is set with a flag as global, the config variable will apply to every repository created by the current user logged in the machine.

System: If the configuration is set with a flag as system, the config variable will apply to every repository created by every user logged in the machine.

To remove the configurations, use the following commands:

Remove user name:

git config --global --unset user.name

Remove user email:

git config --global --unset user.email

To find out the location where the Git configurations are saved, use the following commands:

For local:

For global: cat ~/.gitconfig

For system:

To add a file to the local repository, use the following commands:

Check status:

git status

Add all files:

git add .

Or

git add -a

Again, check status:

git status

Commit changes:

git commit -m "added file file1 and file2"

To make modifications to an existing file tracked by Git and commit the changes, follow these steps:

Make changes in the file. Edit the file:

nano file1.java

Insert any data. To save and exit, press CTL x, then press y, and finally press the enter key.

Now check Git status:

git status

To check the commit history on a Git repository, use the following commands:

Check log:

git log

Check log in one line:

git log --oneline

Assignments:

1. How can I find the log of Git commits from one date to another date?

Use the following commands:

--since=<date>, --after=<date> Show commits more recent than a specific date.

--until=<date>, --before=<date> Show commits older than a specific date.

1. Move/copy/push your Git log history into a new text file.

To delete an existing file from the local repository and working directory, follow these steps:

List files:

git ls-files

Choose a filename to be deleted.

Execute the Git delete command to delete a file from the local repository and working directory:

git rm <filename>

For example, to delete file1.java:

git rm file1.java

The file will be removed from the local repository and working directory.

List files:

git ls-files

List directory contents:

ls

Commit the deletion of the file:

git commit -m "deletion of file"

Check log in one line:

git log --oneline

To revert changes, use the Git revert command. This command always works on a commit id. It reverts the changes made in the commit back to its original state. When you execute the revert command, Git automatically creates a commit, asking you to enter a message.

Execution steps:

Step 1: Check the commit id that you want to revert.

Take the deletion commit id:

git log --oneline

Copy the seven digits, for example, c9bf927.

Step 2: Execute the revert command:

git revert c9bf927

Press the enter key, it will open a nano editor file asking you to insert some message.

Save the file: press CTL x, press y, and finally press the enter key.

Step 3: The file will be back in the local repository and working directory.

List files:

git ls-files

List directory contents:

ls

.gitignore is used to ignore certain files. In this file, write the names of the files that you want to ignore. Git will not prompt you for adding them to the local repository.

Step 1: Create some files that we will ignore:

touch file.txt file2.xml file3.log file4.class

Step 2: Check the status of the files in Git:

git status

Step 3: We do not want to add or commit them, we want to ignore these files:

nano .gitignore

Insert below text:

.txt
.log
.xml
.class

Save the file and exit by pressing CTL x, then press y, and finally press the enter key.

Step 4: Check the status now and commit .gitignore file:

git status

git add .

git commit -m "done .gitignore"

Remote repository:

To create a GitHub account, follow these steps:

Step 2: Connect the SL lab to the GitHub account so that we can push the data from the local repository to the remote repository.

On the lab terminal, execute the command to generate SSH keys. This will enable passwordless authentication between Git (lab) and GitHub servers using the SSH method.

Execution steps:

Step 1: Generate SSH keys:

Go to the lab terminal:

cd

ssh-keygen

Don't enter anything, just keep pressing enter for three times. Your key will be generated.

The public key can be seen by executing the below command:

cat /root/.ssh/id_rsa.pub

Step 2:

Copy the key carefully.

Step 3: Paste it on GitHub.

Step 4: Create a remote repository.

Pull command:

This operation is performed on the local Git repository to get the files and commits from the remote repository.

git pull origin master

This command will perform two operations:

1. Fetch the changes from remote to local repository, placing the changes in a file called FETCH_HEAD.

2. Add the files into the local repository and working directory, also updating the commit history.

Fetch command:

If we want to just fetch the changes from the remote repository into our local repository to check if there are new changes in the remote which are not in my local branch, then we can use the fetch command.

This command will perform one operation:

1. Fetch the changes from remote to local repository, placing the changes in a file called FETCH_HEAD in the .git folder.

This command will not copy your files into the local repository and working directory, nor update the commit history.

Merge command:

Works on the local repository, between your branches.

To delete the branch feature1:

git branch -d feature1

Delete the commit history, so that I create a new commit history to understand rebasing strategy:

git log --oneline

git reset --hard <last commitid>

git status

git add .

git commit -m "done"

Check log in one line:

git log --oneline

Rebase Command:

This is a merging strategy.

1. Create a new branch feature1 (the base of which is master) and switch to the branch:

git branch

git checkout -b feature1

1. Create new commits on feature 1 branch:

touch file-feature

git add .

git commit -m "done on feature"

1. Switch to master and create new commits:

git checkout master

touch file-master

git add .

git commit -m "done on master"

1. No merging of feature1 on master is done.
2. Switch back to feature1 and do the rebase command (reorganize the base branch commits):

git rebase master

Lesson end project:

Come out of the current directory:

cd

Create a directory with includes the files:

mkdir