Git Commands

Git Commands with Explanations and Representations

Git is free and open-source software for distributed version control: tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development.

Try It:

git –version

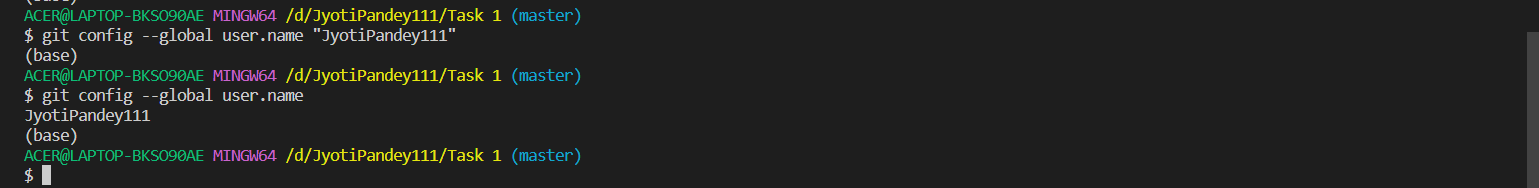


# Configure Git: SETUP

1. Configuring user information used across all local repositories. It is a good idea to introduce yourself to Git with your name and public email address before doing any operation.

Try It:

git config --global user.name “[firstname lastname]”



set a name that is identifiable for credit when review version history

git config --global user.email “[valid-email]”



set an email address that will be associated with each history marker

1. SETUP & INIT: Open the terminal in your project and execute the below command.

Try It:

git init



Initialize an existing directory as a Git repository

You’ve now initialized the working directory—you may notice a new directory created, named ".git".

1. To list changes that you’ve made but not yet added in stagging or committed

Try It:

git status



Files are untracked.

1. Next, tell Git to take a snapshot of the contents of all files under the current director

Try It:

git add .

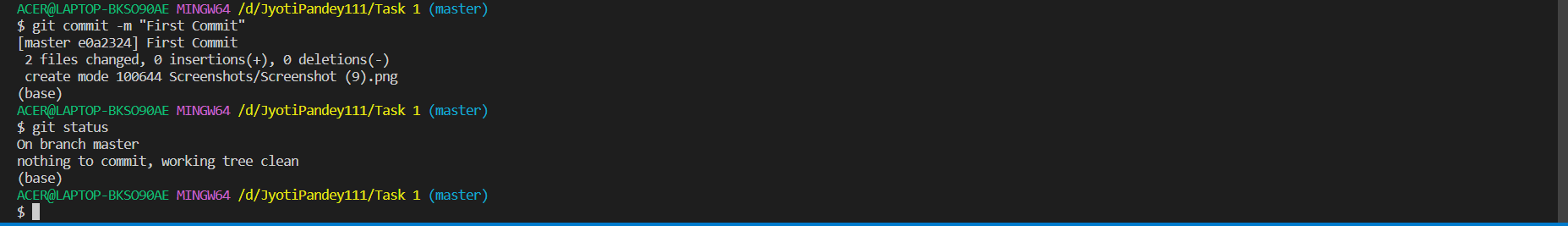


This snapshot is now stored in a temporary staging area which Git calls the "index". Here we have tracked all the file.

1. You can permanently store the contents of the index using the below command.

Try It:

git commit -m "commit message"



Files went into staging.

1. Viewing project history

Try It:

git log

# Managing Branches:

A single Git repository can maintain multiple branches of development.

1. To create a new branch named "branch\_name", use.

Try It:

git branch branch\_name

1. To list all branches

Try It:

git branch

Output: Default branch name “master” and newly created branch will be visible

1. To start working on another branch you can switch branches using the below command.

Try It:

git switch branch\_name

Now make any changes in your repo, add changes to staging, and finally commit your changes then move to the default master branch. Now you will observe that changes made by you will no longer be available in the default “master” branch.

1. To bring changes into the master branch you have to merge your created branch with the master using the below command

Try It:

git merge branch\_name

1. To see all commit history using nice graph representation.

Try It:

git log –graph

1. To delete the branch.

Try It:

git branch -d branch\_name

Note: This command ensures that the changes in the experimental branch are already in the current branch.

1. If you have experimented with something in a branch and you don’t want to include those changes in the main branch then

Try It:

git branch -D branch\_name

Using Git for Collaboration

1. Clone a repo

Try It:

git clone repo\_url

1. Now change your directory to the newly cloned repo

Try It:

cd path\_to\_cloned\_repo

1. Now make any changes and commit it

Try It:

git commit -a -m "commit\_message"

1. Push a repo

Try It:

git push origin "branch\_name"

If more than one person is working in same repo then you might get error during push because other developer might have made changes in github repo which is not available in your local repo

1. To fix that issue you can use git fetch or git pull command git pull is equivalent git fetch and git merge git pull bring changes from remote repo into local and merge the change into your local repo

Try It:

git pull

git fetch bring changes from remote repo and allow us to review and resolve any conflict and after that we can manually commit the new changes.

Try It:

git fetch

git merge

1. Now you can push your changes to github

Try It:

git push origin main