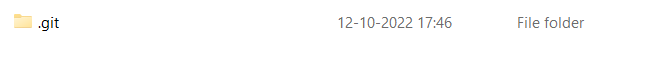
1. **Configuring git for the first time:**

**$ git config –global user.name “<Enter your user name>”**

**$ git config –global user.email “<Enter your email id>”**

1. **Initializing Git:**

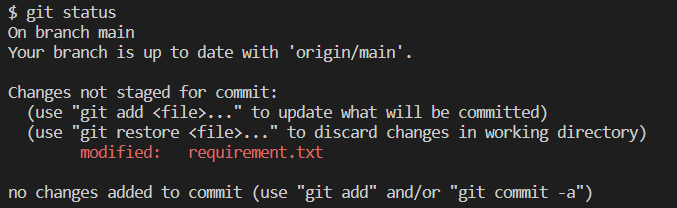
**$ git init**



This command creates a hidden folder ( .git) to keep track of changes. It can help us to convert normal project folder into Git local repository.

1. **Status of files:**

**$ git status**



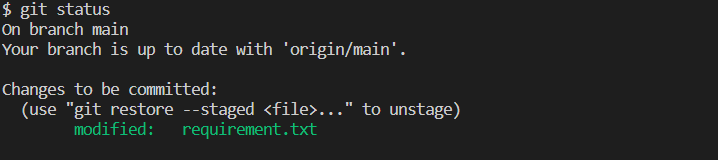
This command shows the status of the git repository. It returns the status of the local working directory and the staging area.

The output shows, the user has made some modifications in the requirement.txt file but the file is not yet added to the staging area.

1. **Adding files:**

**$ git add <file> or $ git add <file1> <file2> or $ git add .**

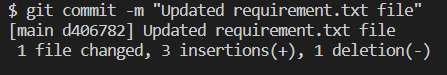
After adding the modified file to the staging area then the repository status looks like below:

****

1. **Git commit:**

**$ git commit -m “Enter your message”**

This command helps to commit the modified files which are available in the staging area. It keeps track of our progress and changes as we work. Git considers each commit as a change point or as a save point. Git assigns a 40 character string known as SHA or SHA# for each commit, which helps to track the previous commits.

****

Sometimes, when we make small changes, using the staging environments to commit seems like a waste of time. It is possible to commit changes directly by skipping the staging environment. Command for the direct commit is:

**$ git commit -a -m “Enter your message”**

1. **Git log:**

**$ git log**

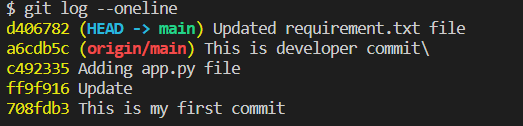
This command is used to check the history of commits for a repository.



Shorter version of log is:

**$ git log –oneline**

This returns all the commit history in a shorter way.



1. **Connecting local repo to the remote repo:**

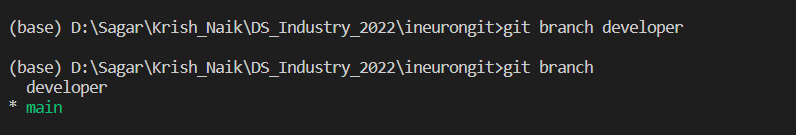
**$ git remote add origin <URL>**

This specifies that we are adding a remote repo, with the specified URL, as an origin to our local Git repo.

1. **Creating a new branch:**

Branch is a separate version of the main repo. Branches allow you to work on different part of a project without impacting the main branch. When the work is complete, a branch can be merged with the main project.

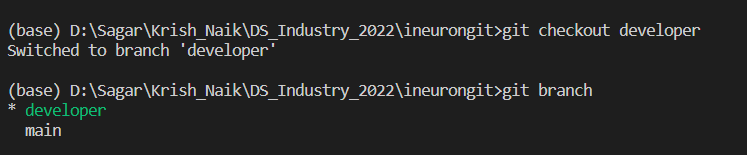
**$ git branch <name of the branch>**

****

This command creates a new branch “developer”.

1. **Switching to other branches:**

**$ git checkout <branch name>**

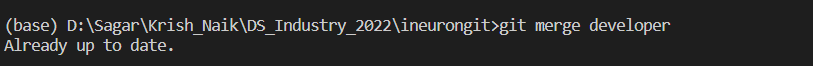
****

Switching from “main” branch to “developer” branch.

1. **Merging two branches:**

**$ git merge <branch name>**

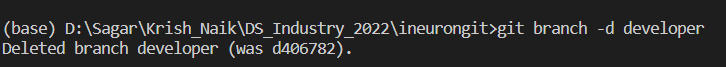
Before merging any branch to the main branch, its preferred to switch to the main branch.



This will merge developer branch with main branch.

1. **Deleting a branch:**

**$ git branch -d <name of the branch>**



This deletes the branch “developer” from the repo.

1. **Adding remote repo:**

First create a Github account and create a remote repository where we can push our files from local repo.

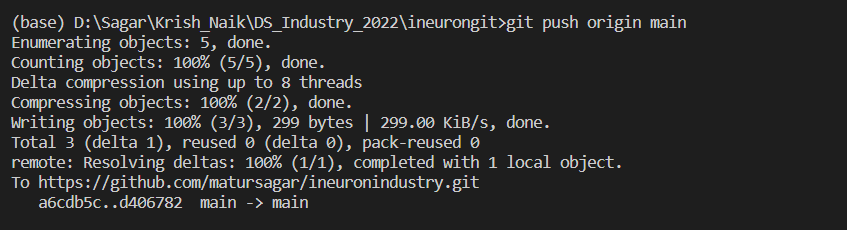
**$ git remote add origin <remote repo URL>**

This specifies that we are adding a remote repo, with the specified URL, as an origin to our local Git repo.

1. **Pushing local repo to the github remote repo:**

Before pushing the code or files, commit all the changes in the local repo.

**$ git push -u origin main**

****

1. **Pull remote repo to the local repo:**

If more then one developers are working in a single remote repo then one might get an error during push because other developer might have made changes in remote repo.

**$ git pull origin**

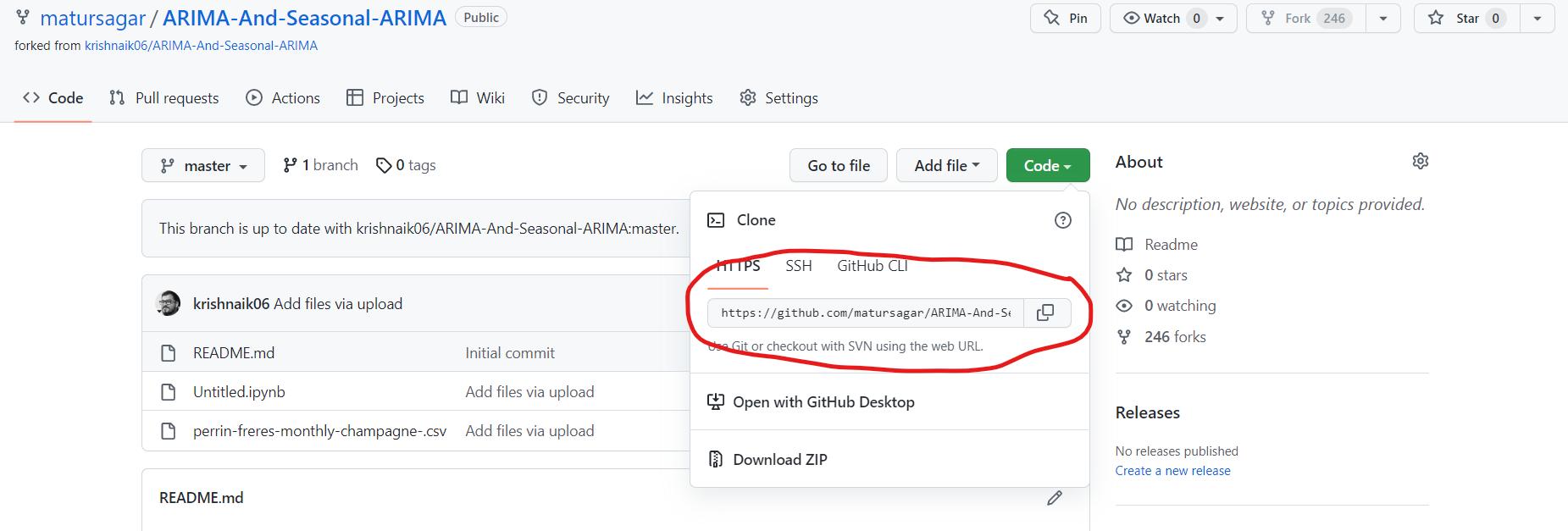
1. **Git clone from Github repo:**

We can clone (or copy) a forked repo from Github on our local repo. A clone is a full copy of a repository.

**$ git clone <URL>**

**$ git clone <URL> <Folder Location>**

This command returns a copy of the github repo by using its URL.



We can also specify the folder location of local machine, where we want to store the copy. Otherwise repo will be copied in the current working directory.

