***GIT document***

Contents

[***What is Git?*** 2](#_Toc105965565)

[***How Git Works?*** 2](#_Toc105965566)

[***Installing Git on your Machine*** 3](#_Toc105965567)

[***Git Setup (Login)*** 3](#_Toc105965568)

[***Cloning the Repository*** 3](#_Toc105965569)

[***How to Prepare your Project Folder in Git*** 4](#_Toc105965570)

[***How to Collaborate with Other Developers Using Git*** 5](#_Toc105965571)

[***What is a Git Commit?*** 5](#_Toc105965572)

[***A Typical Git Workflow*** 6](#_Toc105965573)

# ***What is Git?***

Git is a version control system that allows multiple developers to contribute to a project simultaneously. It is a command-line application with a set of commands to manipulate commits and branches

# ***How Git Works?***

***Branching and Commits***

One of Git's biggest advantages is its branching capabilities. This allows someone to branch off of the master branch (where the production-quality code typically remains) and work on a feature or fix independently of the rest of the project. Once they finish their work, they can merge the branch back into the master branch with the click of a button.

FEATURE o-----o-o---------o-----o

/ \

MASTER <---o----o------o--o-------------------o------>

Feature branches provide an isolated environment for the developer to work. They are also useful for organizing different levels of granularity. For instance, in what is known as the Git-Flow, there are three important branches: master, staging, and develop. Master is the parent branch, representing the state of the project that is released to the public (deployed). The staging branch is where pre-release QA is done. Develop is where developers work on the project. All feature branches are branched from develop.

FEATURE1 o-----o-o---------o-----o

/ \

DEVELOP <-------o-----o---o-----------o-------o------>

\ \

STAGING <---------------o---------------o----o-------->

\ \

MASTER <------------------o-------------------o------>

In the diagram above, commits are represented by o in the branches. A commit is an individual piece of work. It's a bit like a milestone--when a small piece of the task is completed

***Workspace, Staging, Local, and Remote***

A Git repository is also split into four levels. Each level represents a state of the codebase. Each state is isolated from each other, but git commands allow you to transfer data between them.

* The ***workspace*** is the folder on your computer. This is where all the work happens.
* The ***staging*** area is where changed files are placed before making a commit. It allows you to essentially select only the files you want committed.
* The ***local*** is your copy of the repository. No one else will see this except you.
* The ***remote*** is the public repository. Everyone sees the same remote.

So, evidently, moving changes from one workspace to another will involve passing from ***workspace1 > staging1 > local1 > remote > local2 > staging2 > workspace2***. It is also important to note that all communication must pass through the remote repository.

# ***Installing Git on your Machine***

If you're using UNIX, you're in luck! Git come pre-installed on macOS and on most distributions of Linux. Type git --version in the terminal to check if you have Git installed.

Git does not come pre-installed on Windows.

[Git Bash](https://git-for-windows.github.io/) is a great option; it emulates a UNIX shell environment. You can also download a version of [Git](https://git-scm.com/download/win) that you can use in the Command Prompt. [Smart git](https://www.syntevo.com/smartgit/download/) is the graphical user interface

# ***Git Setup (Login)***

Let's get you setup to use Git. First, you need to identify yourself. When you first install Git, it's important to set your username and email address because every commit is signed with this information. In the terminal:

git config --global user.name "John Doe"

git config --global user.email johndoe@example.com

# ***Cloning the Repository***

You will need to clone the repository to your local machine. Open the terminal (or Git Bash if using Windows) and navigate to the folder in which you wish to clone your repository. Now, run the following command:

***git clone <REPO CLONE URL>***.

You can get the repository clone url from the Github or Gitlab repository. You have the option of cloning with HTTPS or with SSH.

This should create a new child directory for the project sources. From the new project working directory, you are able to run your git commands.



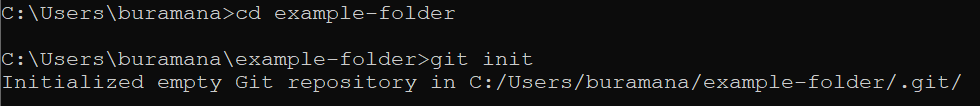
# ***How to Prepare your Project Folder in Git***

Need to create one directory in local machine

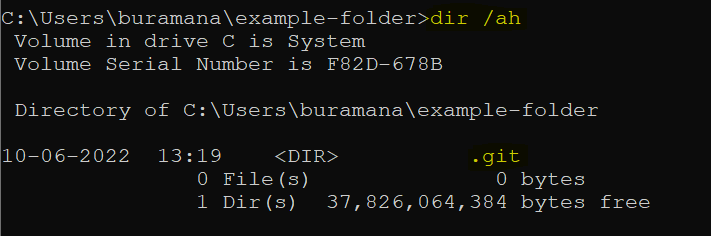


Then, go inside newly created folder and initiate the git repository locally

git init : to initialize the current folder/directory as git repository.



To view hidden files in windows you can use ***dir /ah*** command. Then you can be able to see. git folder



***. git*** folder inside our project folder. It will track all the changes made to the files in our project and record that history over time.

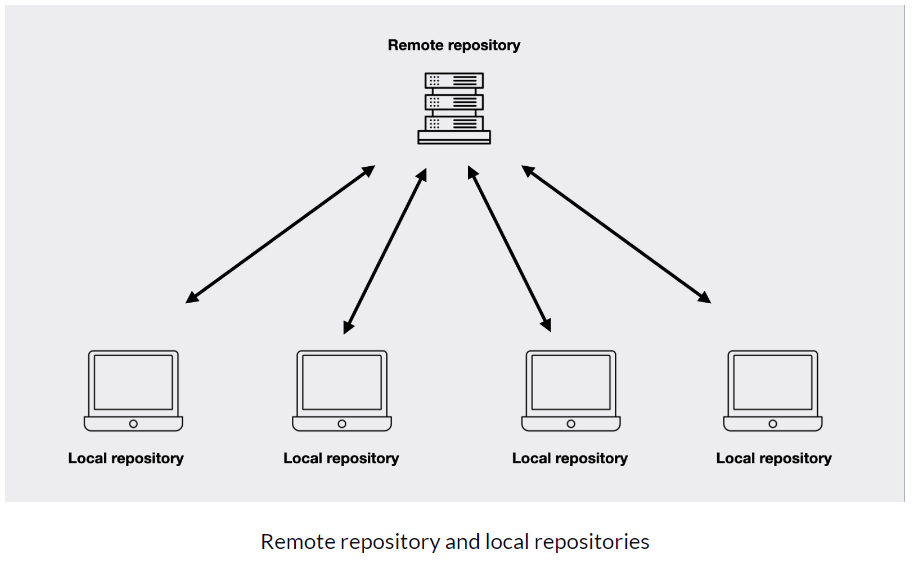
When we push (in other words upload) our local repository to one of **github or gitlab or azure repos or bitbucke**t, then the repository that resides in this service in the cloud is referred to as the **remote repository**

# ***How to Collaborate with Other Developers Using Git***

If another developer wants to collaborate with us on our project then they can clone (or in other words download) the remote repository from the hosting service you uploaded it to their computer.

This allows them to have the project on their computer as well. The project on their computer is then also referred to as a local repository.

In a project with multiple developers, each one has a local repository on their computer. And there is one remote repository that they all contribute to and they use it to share their work.

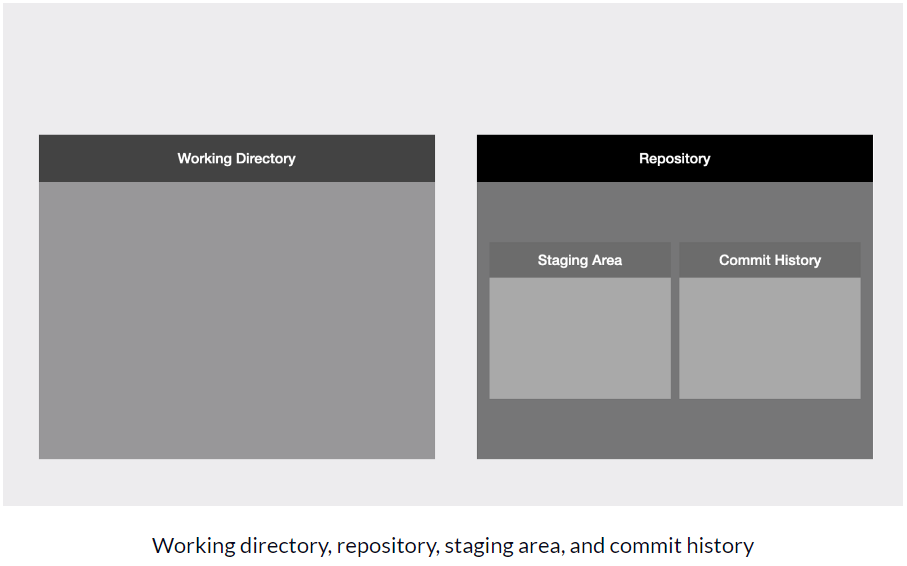


# ***What is a Git Commit?***

A **commit** is a version of your project. It represents a standalone version of your project and has a reference to all the files and folders that are a part of that version.

***How Do I Make a Commit in Git?***

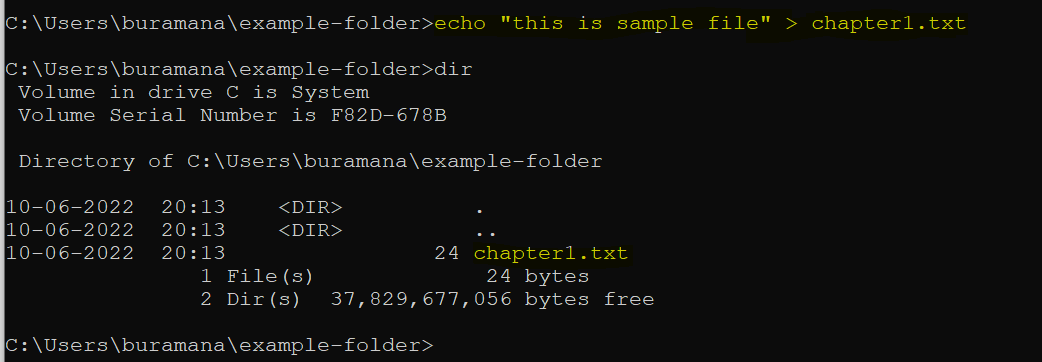
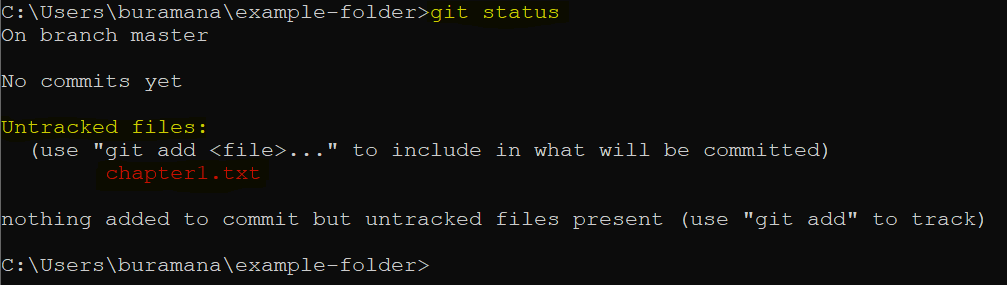
The ***working directory*** is basically represented by the contents of our project folder (hint: a directory is the same thing as a folder). It is sort of like a work bench, where we can add, edit, and delete files in our project. The staging area and commit history are part of our repository. The ***staging area*** is sort of like a rough draft space. It is where we can add updated versions of files or remove files in order to choose what we want to include in our next commit (version of our project). In the. git folder the staging area is represented by a file called index. And finally, the ***commit history*** is basically where our commits live after they’ve been made. In the. git folder the commit history is represented by a folder called objects.

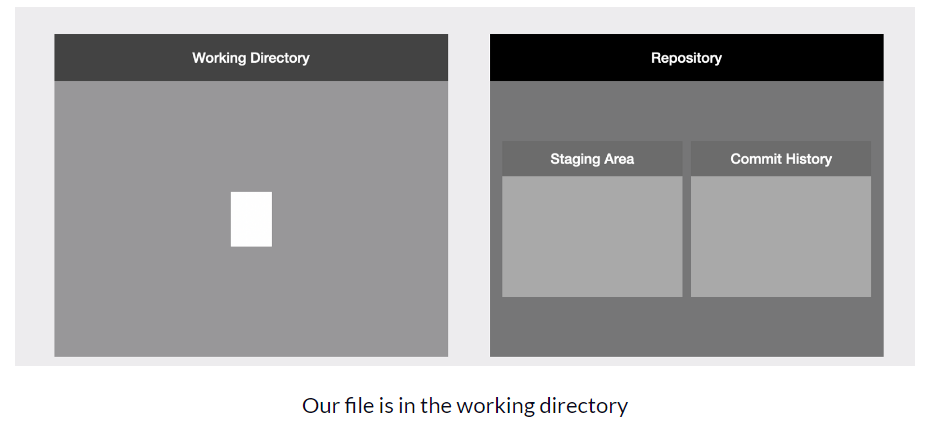


# ***A Typical Git Workflow***

***Step 1 — Edit Files/Add new Files***

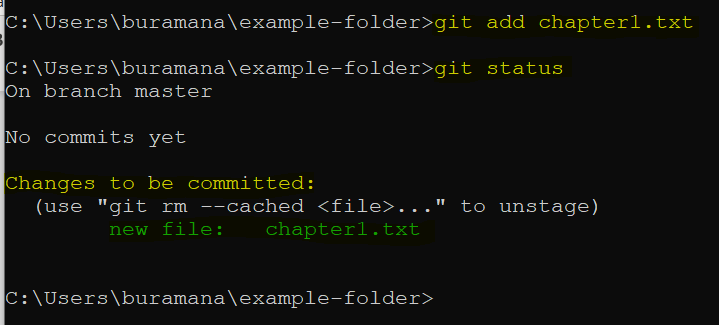
If you have a new project, you will create the very first file in your new project. In our ***example-folder*** project folder we will make a simple text file called ***chapter1***.

If you have an existing project then you will edit some of your existing files, add new files, or delete files.Next, we can use the ***git status*** command. This command will tell us the state of our working directory and staging area and will tell us if there are any differences between the two.In our example, we added just one new file to our new project. When we use the ***git status*** command, Git tells us that we have an untracked file in our working directory and that we need to use the ***git add*** command to include it in what will be committed. Which takes us to ***step 2***.

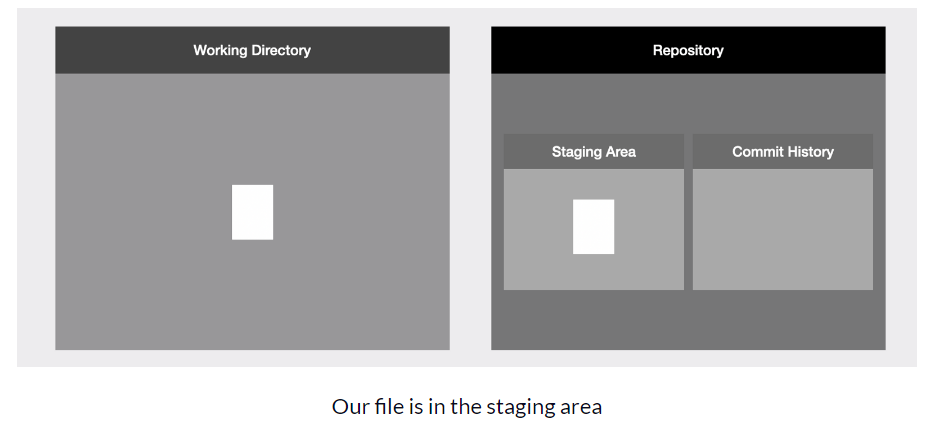


***Step 2 — Add Files to the Staging Area***

We can use the ***git add*** command in order to add new or updated files to the staging area. If we decide we don’t want to include some of the files we changed in our next commit then we simply make sure not to add those particular files to the staging area.



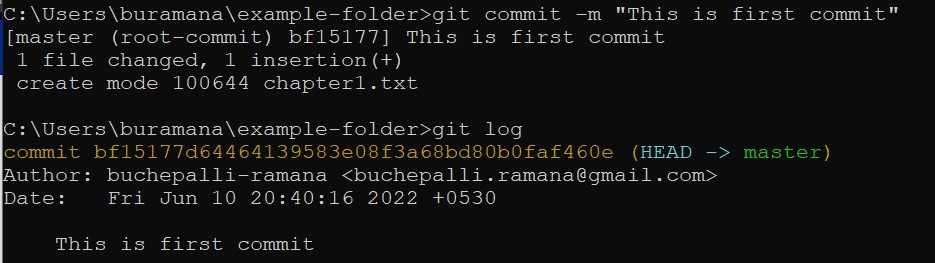
In our example, we add the only file we have in our project to the staging area using the ***git add*** command and passing in the name of the file. Then if we use the ***git status*** command we will see that Git will tell us that we have added our file to the staging area.



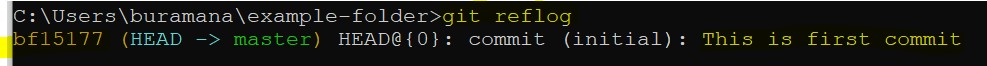
It is also important to note that files **do not** **move** from the working directory to the staging area. Files are **copied over** from the working directory to the staging area.

***Step 3 — Make the Commit***

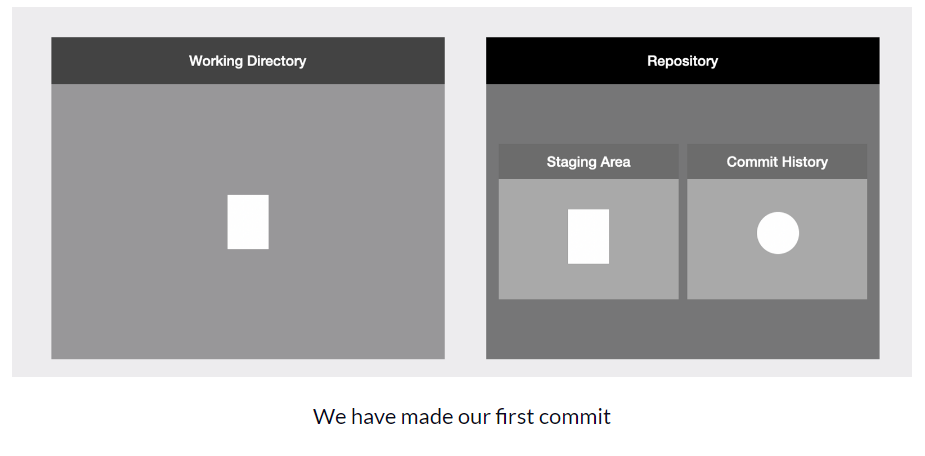
Finally, to make the commit we use the ***git commit*** command with ***-m*** option and pass in a commit message, for example ***git commit -m "this is the first commit"***



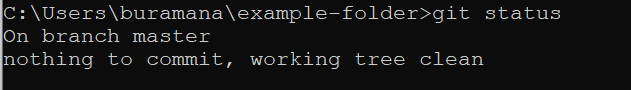
We can get short message by running ***git reflog***



We can then use the ***git log*** command in order to list all the commits we have in our project in reverse chronological order. In our example we only have one commit.



Now if we run ***git status,*** it will display working tree is clean***.***



Now we have made our first commit in our project! In other words we have saved the first version of our project.

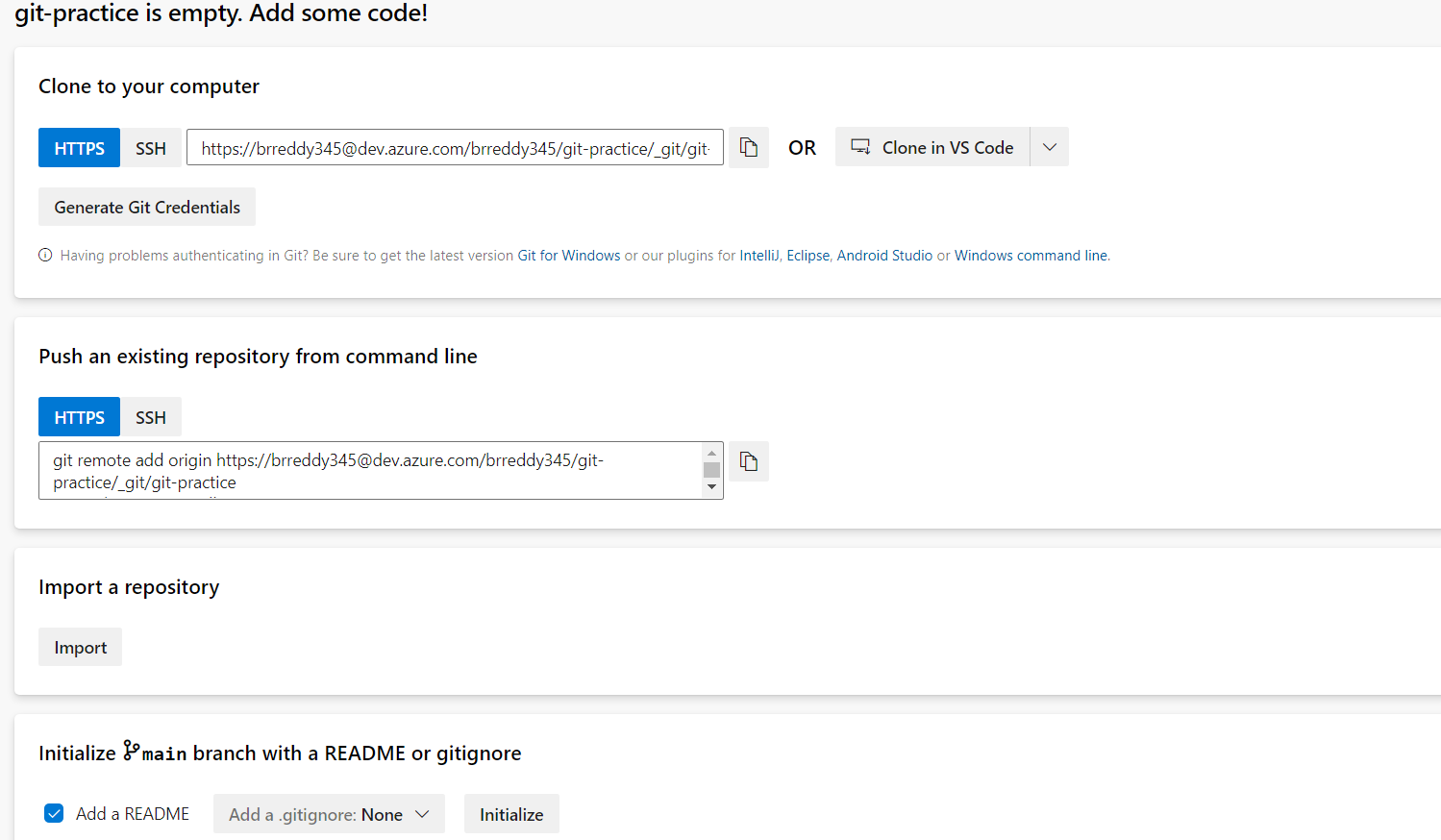
***Step 4— Push Local git repository to Remote repository(I took Azure repos)***

I Have a git repository in github: [*https://github.com/buchepalli-ramana/git-practice.git*](https://github.com/buchepalli-ramana/git-practice.git)

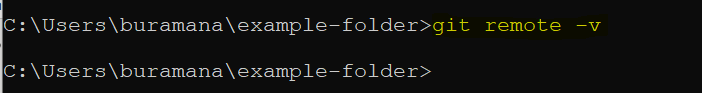
I have git repository in Azure Repos

*https://brreddy345@dev.azure.com/brreddy345/git-practice/\_git/git-practice*

It is empty repository.

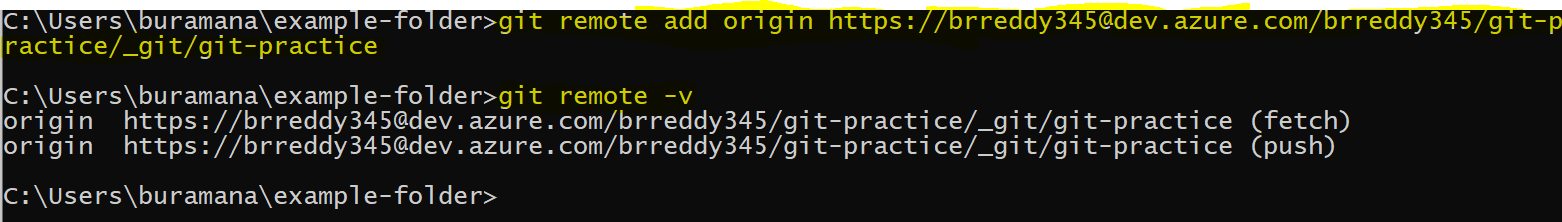


As of now there is no remote added.

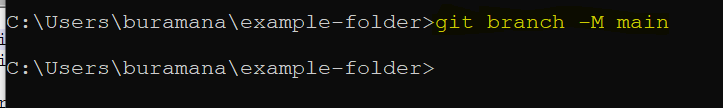


Now I will push the local ***chapter1.txt*** to github repository.

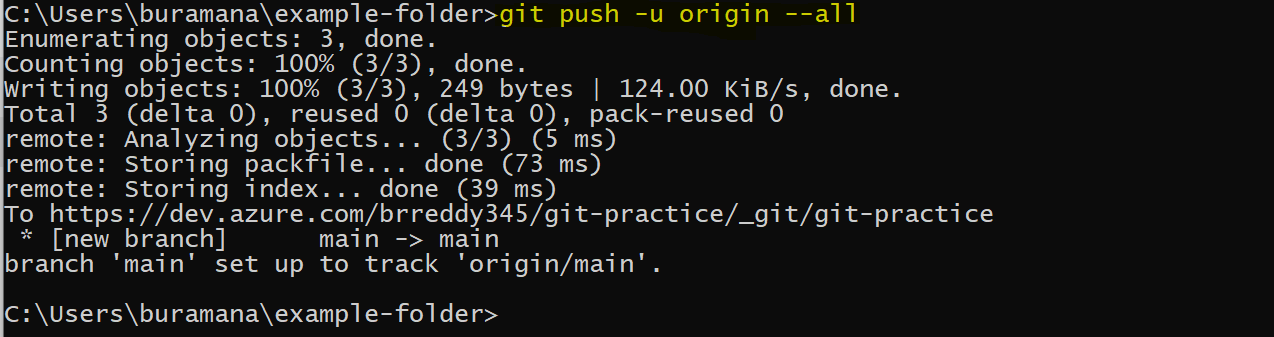
For that first we need add remote repository to local git repo. By running below command we can add it

***Git remote add origin <url of azure devops repository>***

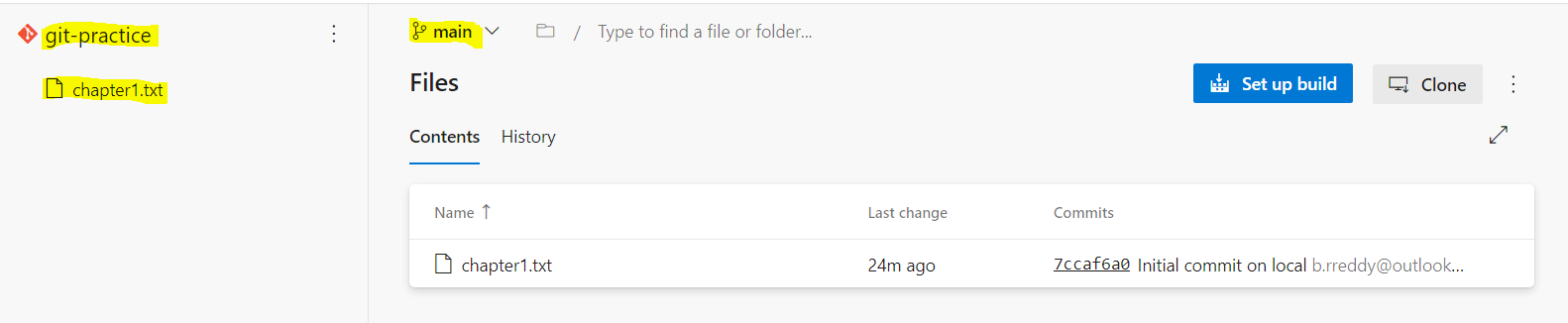
Renaming the master branch name to main branch ***git branch -M main***



Now push the local git repository to remote repository called ***git-practice***



Go to azure DevOps repository and refresh it.



Here is the log information. Origin/main is represents the remote repo i.e, Azure devops repo.

