Experiment-2
Explore Git and GitHub commands

Version Control System

Version Control is the **process of tracking and managing changes to software code**. It's also called **revision control** or **source code control**. These systems help teams manage code versions, collaborate, and keep history of changes.

Why Use a Version Control System?

1. **Maintain Multiple Versions**
   - You can store different versions of code and switch between them based on client or project needs.
2. **Metadata Tracking**
   - Each commit stores information like:
     - Who made the change , When it was made , What was changed ,Why it was changed (via commit message)
3. **Code Sharing Made Easy**
   - Developers can easily share code with their teammates.
4. **Collaboration**
   - Multiple developers can work on the same project at the same time.
5. **Parallel Development**
   - Developers can work on different features or fixes without interfering with each other.
6. **Access Control**
   - You can define **who can view** or **modify** the code in the repository.
7. **Error Recovery**
   - If something breaks, you can **revert** back to a previous working version.

VCS Terminology

- **Working Directory**
  - This is where developers **create or edit files** on their local machines.
  - **Version control does not apply here yet.**
  - No version tags like version-1 or version-2 are used at this point.
- **Repository**
  - This is where files and **their version history** are stored.
  - **Version control is active here.**
  - You can talk about versioned snapshots: version-1, version-2, etc.
- **Commit**
  - The process of **saving changes** from the working directory into the repository.
- **Checkout**
  - The process of **retrieving files** from the repository into the working directory.
  - Used when switching versions or downloading code to edit.

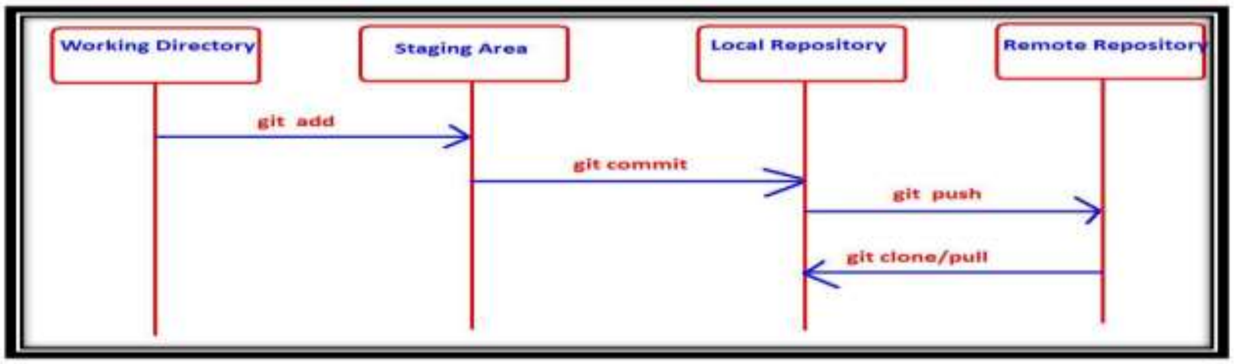
Git

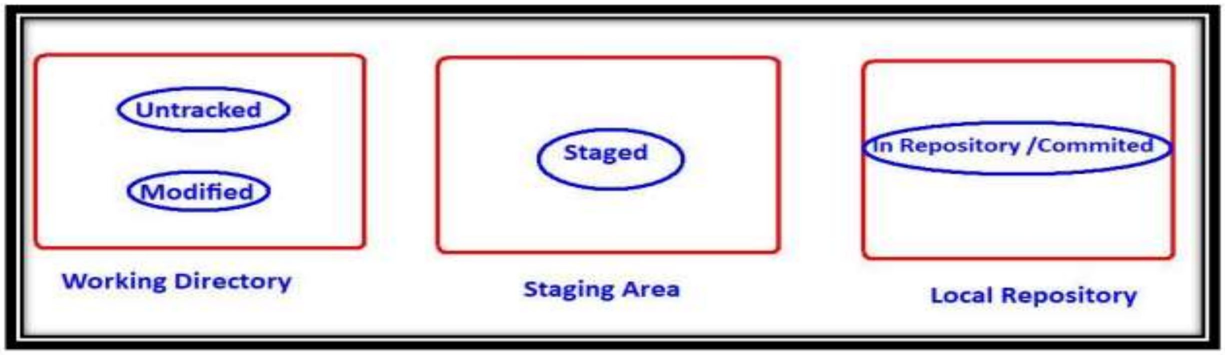
- Git is a Distributed Version Control System Tool.
- Git is not acronym and hence no expansion. But most of the people abbreviated as
    - "Global Information Tracker''.
- GIT is developed by Linus(lai-nuhs taw vldz) Torvalds(Finish software engineer), who also developed Linux Kernel.

Features of Git

- ✓ Every Developer has its own local repository. All the operations can be performed locally. Hence local repo and remote repo need not be connected always.
- ✓ All operations will be performed locally, and hence performance is high when compared with other VCS. i.e it is very speed
- ✓ Most of the operations are local. Hence we can work offline most of the time.
- ✓ There is no single point failure as Every Developer has his own local repository.
- ✓ It enables parallel development & automatic-backups.

- **Git Architecture**





K.Harshini
DevOps(Cse)

Repositories

Git contains 2 types of repositories:

1. **Local Repository(Git)**
2. **Remote Repository(GitHub)**

1.Local Repository:

It is a version-controlled directory on your own computer where Git tracks changes to files and directories. It allows you to manage code, track history, and collaborate with others without needing an internet connection (until you're ready to push to a remote).

Key Components of a Git Local Repository

- ✓ **Working Directory**: The actual files and folders you work on.
- ✓ **Staging Area (Index)**: A place where you prepare changes before committing. You add changes here using git add.
- ✓ **Local Repository (.git directory)**: A hidden folder (.git/) that stores all version history, branches, commits, etc.

2.Remote Repository:

In Git is a version of your project hosted on the internet or another networked server.

It's typically used to **collaborate with others** or **back up your code**.
The most common remote hosts are platforms like **GitHub**, **GitLab**, **Bitbucket**, or a private server.

K.Harshini
DevOps(Cse)

GitHub

- GitHub is a Git repository hosting service.
- GitHub also facilitates with many of its features, such as access control and collaboration. It provides a Web-based graphical interface.
- GitHub is an American company. It hosts source code of your project in the form of different programming languages and keeps track of the various changes made by programmers.
- It offers both distributed version control and source code management (SCM) functionality of Git. It also facilitates with some collaboration features such as bug tracking, feature requests, task management for every project.

➢ Benefits of GitHub:

GitHub can be separated as the Git and the Hub.
GitHub service includes access controls as well as collaboration features like task management, repository hosting, and team management.

The key benefits of GitHub are as follows.

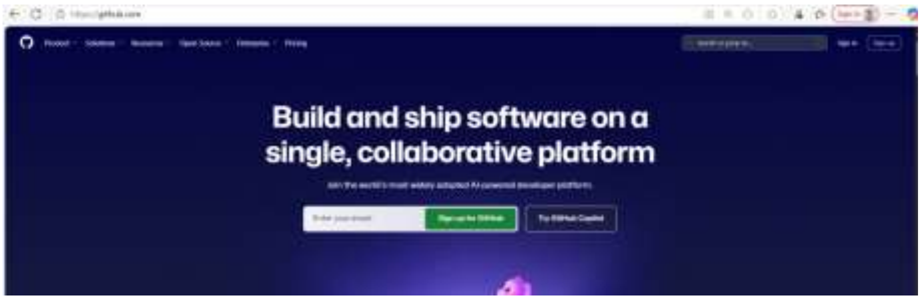
- ✓ It is easy to contribute to open source projects via GitHub.
- ✓ It helps to create an excellent document.
- ✓ You can attract recruiter by showing off your work. If you have a profile on GitHub, you will have a higher chance of being recruited.
- ✓ It allows your work to get out there in front of the public.
- ✓ You can track changes in your code across versions

Next we are going to Create a GitHub account(Remote repository) and Setup (localRepository) and using git commands we will make necessary changes and push from local repository to remote repository

<u>**Creation of GitHub Account and Repository**</u>

1.Go to github.com

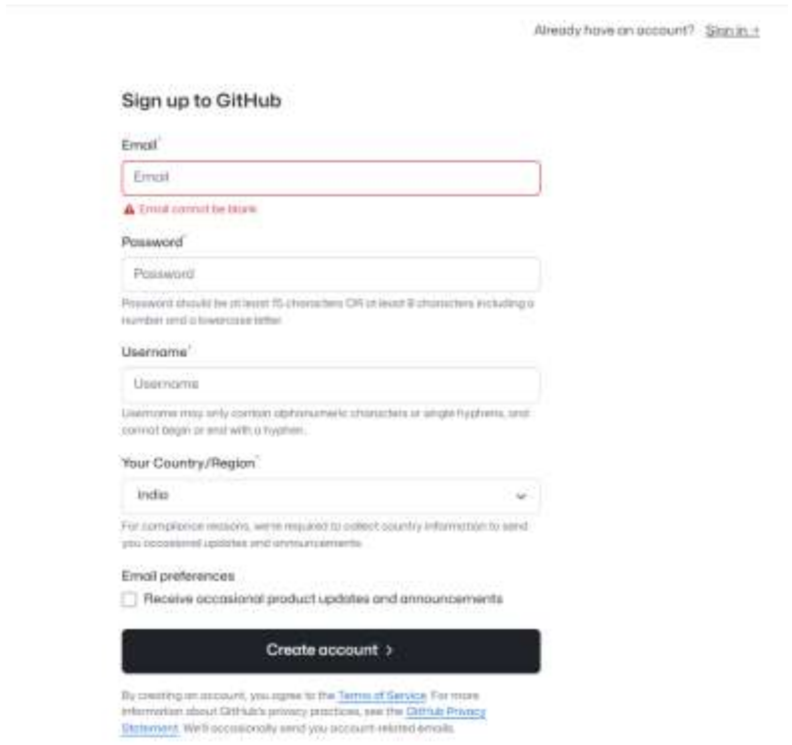
2.Select 'Sign Up' to create a new account. If you already have an account, click on 'SignIn'.


Build and ship software on a single, collaborative platform

Join the world's most widely adopted AI-powered developer platform.

3. Considering as Sign-Up for new account:

Provide necessary details in the signup page with Emailid, NewPassword, Username and Create account


Already have an account? Sign in →

Sign up to GitHub

Email*

Email

Email cannot be blank

Password*

Password

Password should be at least 15 characters OR at least 8 characters including a number and a lowercase letter.

Username*

Username

Username may only contain alphanumeric characters or single hyphens, and cannot begin or end with a hyphen.

Your Country/Region*

India

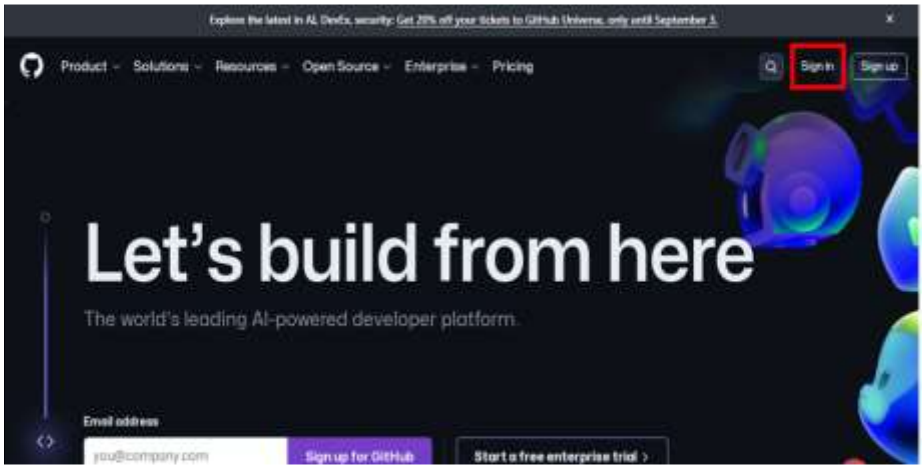
For compliance reasons, we're required to collect country information to send you occasional updates and announcements.

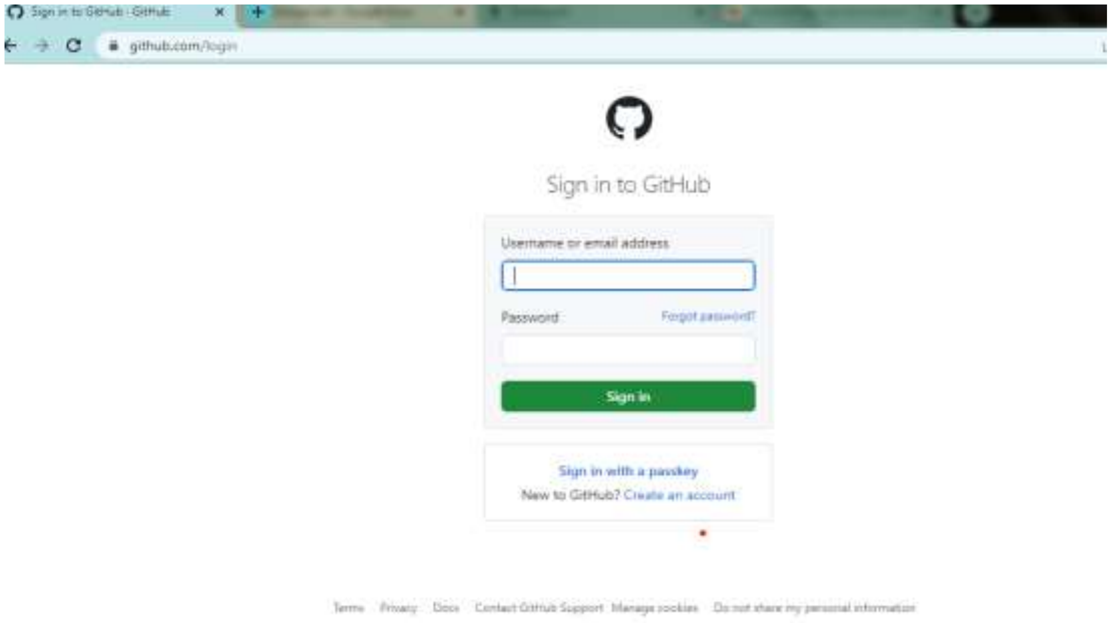
Email preferences

☐ Receive occasional product updates and announcements

By creating an account, you agree to the Terms of Service. For more information about GitHub's privacy practices, see the GitHub Privacy Statement. We'll occasionally send you account-related emails.

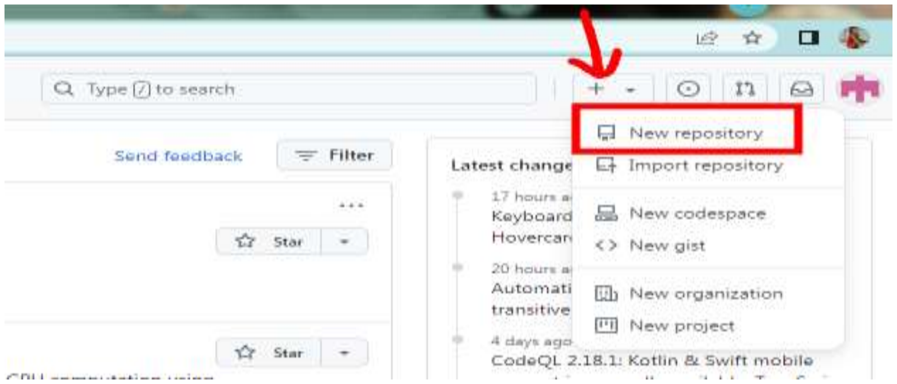
4-After Account Creation Sign-in to the account with previously created credentials

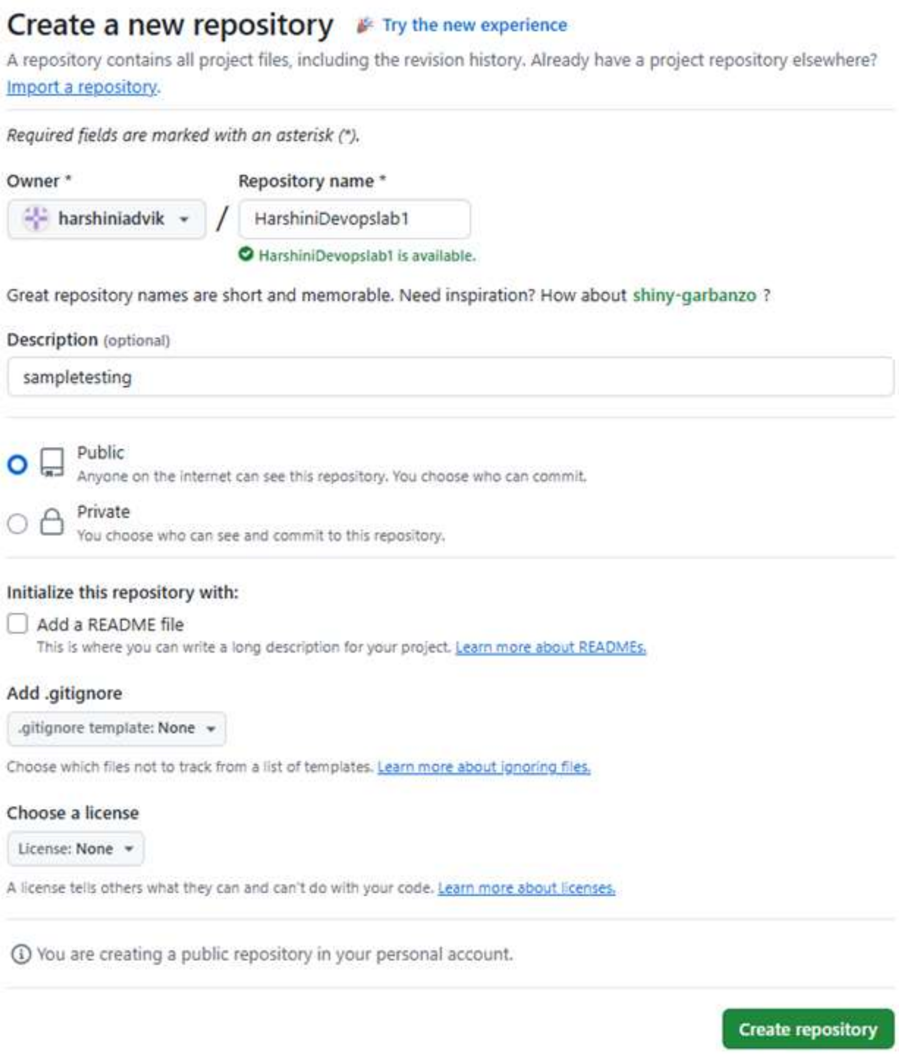




K.Harshini
DevOps(Cse)

5-Create New-Repository with a ProjectName




Create a new repository Try the new experience

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Owner * harshiniadvik / Repository name * HarshiniDevopslab1

HarshiniDevopslab1 is available.

Great repository names are short and memorable. Need inspiration? How about shiny-garbanzo ?

Description (optional)

sampletesting

[✓ Public] Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Initialize this repository with:

☐ Add a README file
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. Learn more about ignoring files.

Choose a license

License: None

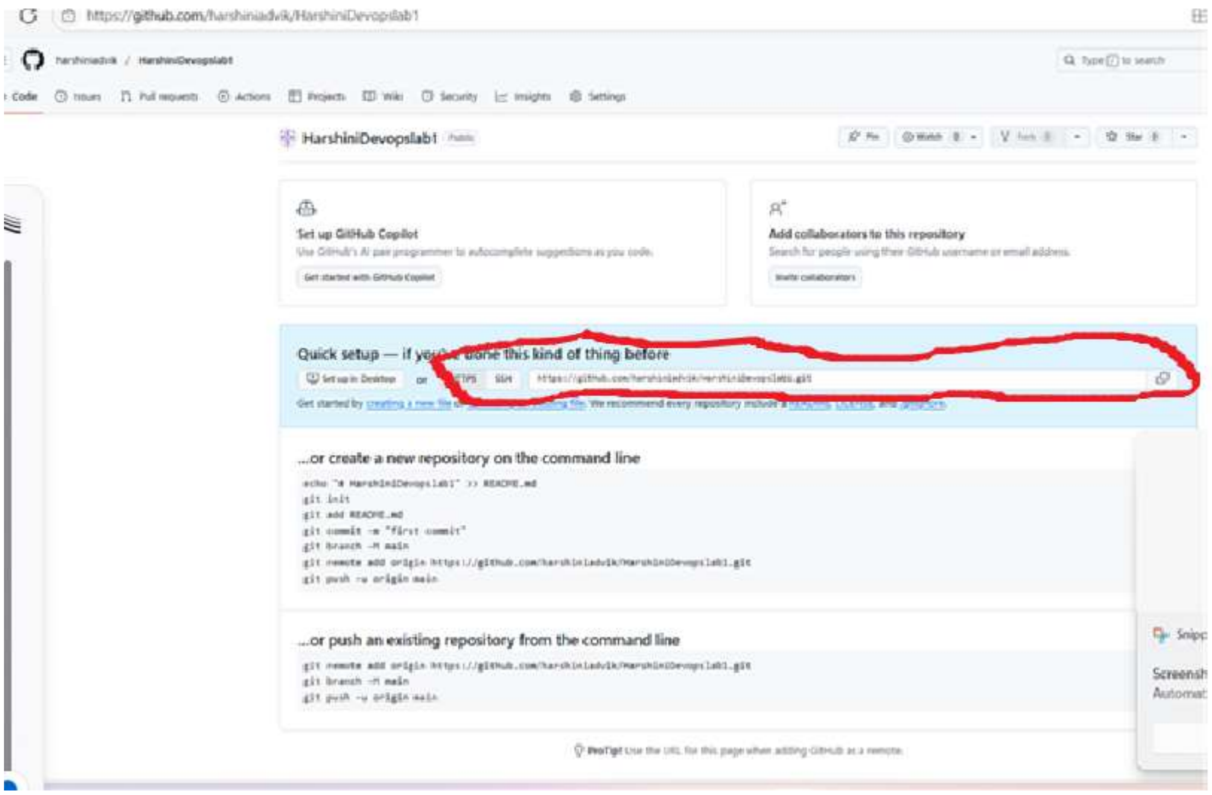
A license tells others what they can and can't do with your code. Learn more about licenses.

You are creating a public repository in your personal account.

Create repository

K.Harshini
DevOps(Cse)

6- Once Repository is created redirect to that directory


https://github.com/harshiniadvik/HarshiniDevopslab1
harshiniadvik / HarshiniDevopslab1
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
HarshiniDevopslab1
Public
Set up GitHub Copilot
Use GitHub's AI pair programmer to autocomplete suggestions as you code.
Get started with GitHub Copilot
Add collaborators to this repository
Search for people using their GitHub username or email address.
Invite collaborators
Quick setup — if you've done this kind of thing before
Set up in Desktop
HTTPS
SSH

...or create a new repository on the command line

```
echo "# HarshiniDevopslab1" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/harshiniadvik/HarshiniDevopslab1.git
git push -u origin main
```

...or push an existing repository from the command line

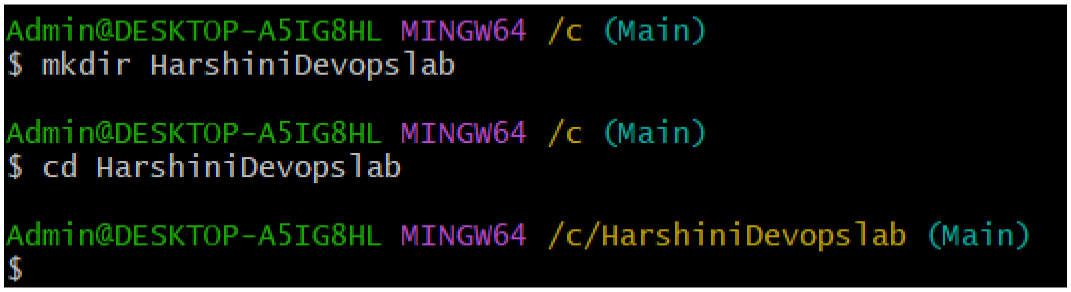
```
git remote add origin https://github.com/harshiniadvik/HarshiniDevopslab1.git
git branch -M main
git push -u origin main
```

This is the url of the remote repository
https://github.com/harshiniadvik/HarshiniDevopslab1.git
By using this url only we can communicate with remote repository.

Set Up Local Repository

Using **Gitbash** or **VisualStudio**

First create directories



```
Admin@DESKTOP-A5IG8HL MINGW64 /c (Main)
$ mkdir HarshiniDevopslab

Admin@DESKTOP-A5IG8HL MINGW64 /c (Main)
$ cd HarshiniDevopslab

Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (Main)
$
```

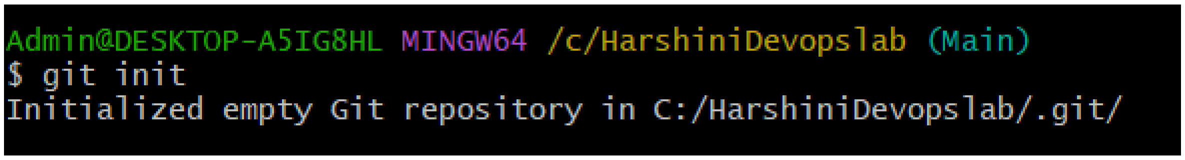
Now **HarshiniDevopslab** directory acts as a working directory.we have to request to version control for this directory.for this we have to use the **git init** command.

Git Commands Starts from here

➢ git init:

- ❖ Once we create a workspace, if we want version control, then we require a local repository. To create that local repository we have to use the git init command.
- ❖ .git is an empty repository, which is a hidden directory.

  Syntax: git init

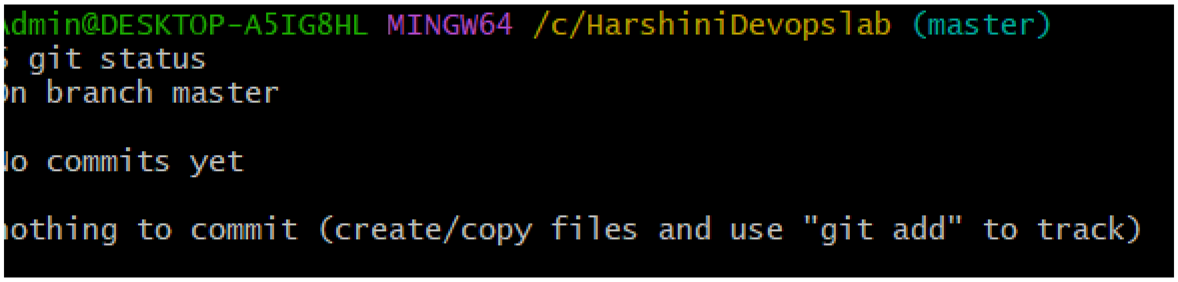


```
Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (Main)
$ git init
Initialized empty Git repository in C:/HarshiniDevopslab/.git/
```

➢ git status:

- ❖ It shows the current status of all files in each area, like which files are untracked, which are modified, which are staged etc

  Syntax: git status



```
dmin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
 git status
n branch master

o commits yet

othing to commit (create/copy files and use "git add" to track)
```

K.Harshini
DevOps(Cse)

➢ **git config**:

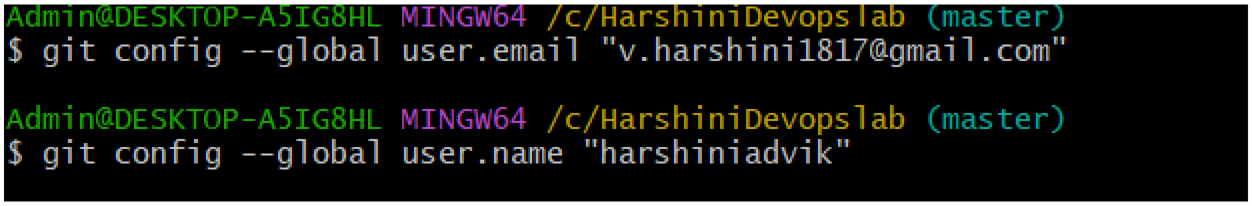
❖ **Git Configurations before 1st Commit:**

❖ Before the first commit, we have to configure username and mail id, so that git can use this information in the commit records. We can perform these configurations with the following commands.

Syntax:

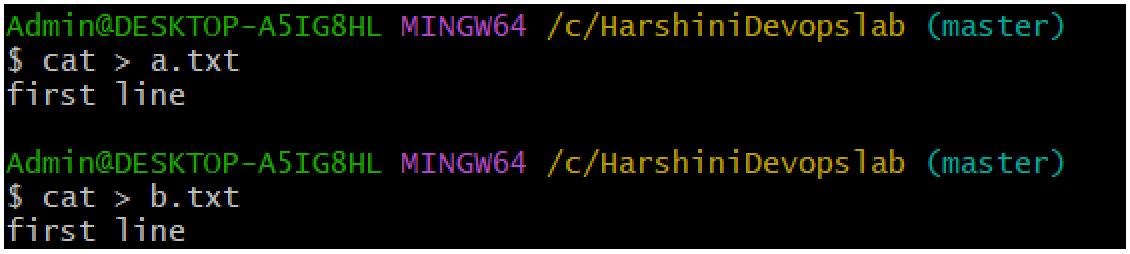
git config --global user.name "Your Name"

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



```
Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ git config --global user.email "v.harshini1817@gmail.com"

Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ git config --global user.name "harshiniadvik"
```

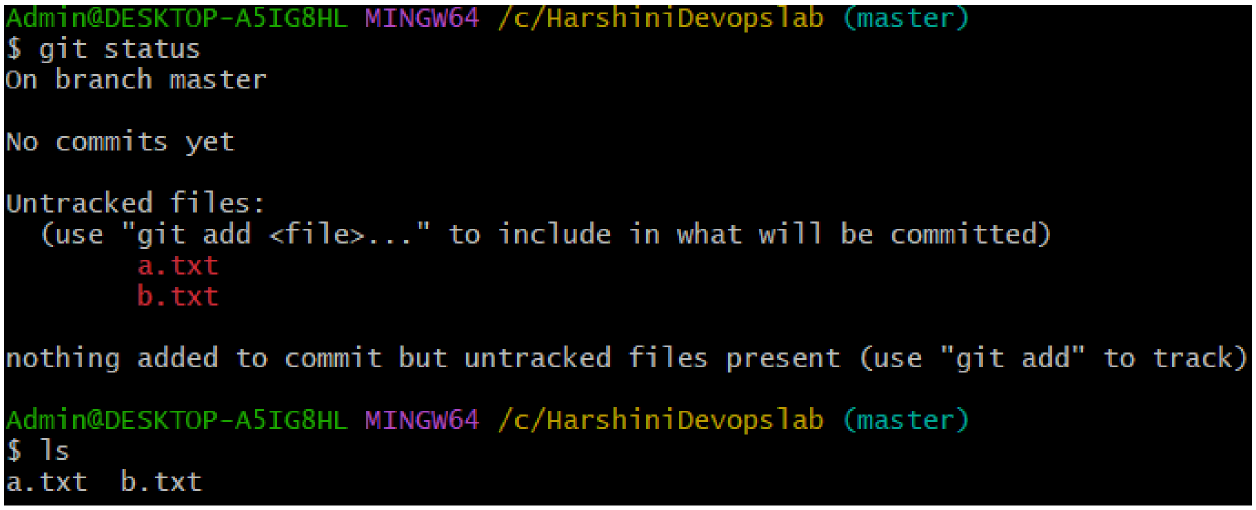
Create a.txt and b.txt with some content



```
Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ cat > a.txt
first line

Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ cat > b.txt
first line
```

After writing the content , press ctrl+D to get back .

Just check again with git status command which will show the untracked files and use command ls to check the files



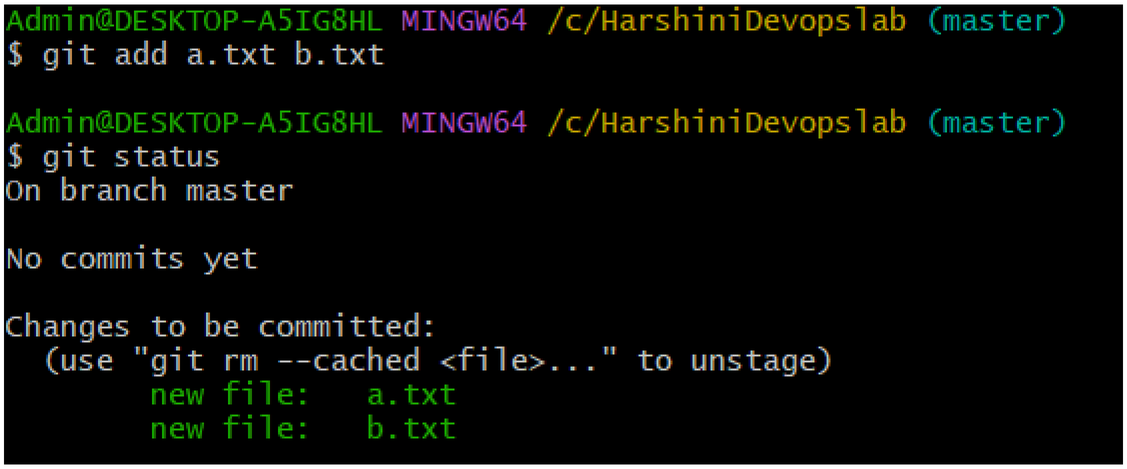
```
Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        a.txt
        b.txt

nothing added to commit but untracked files present (use "git add" to track)

Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ ls
a.txt  b.txt
```

➢ git add:

To add files from the working directory to the staging area for tracking/coming purpose, we have to use this git add command

git add <filename> # Stage a specific file

Ex: **git add a.txt b.txt**



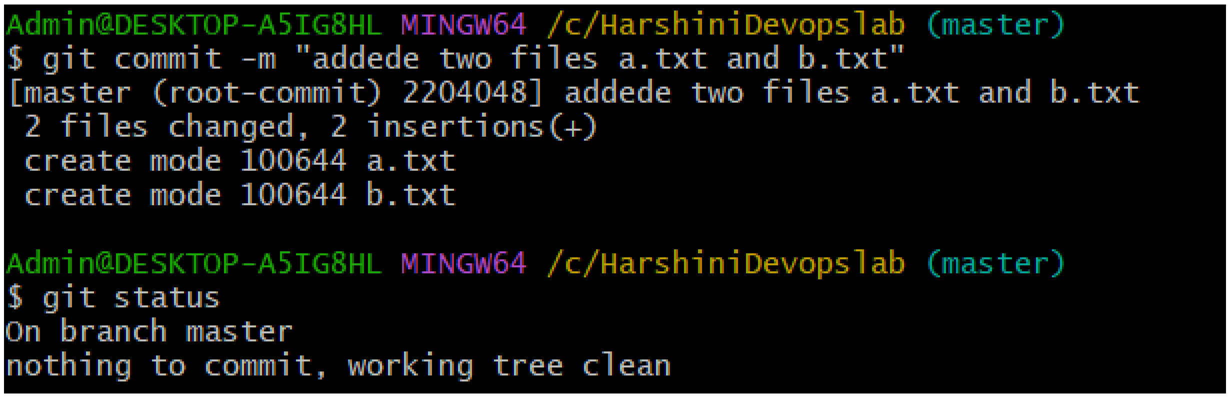
```
Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ git add a.txt b.txt

Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   a.txt
        new file:   b.txt
```

git add . # # Stage all changes in the current directory(Extra data)# Not required to execute now

➢ git commit:

- ❖ If we want to commit staged changes, then we have to use the **git commit** command. For every commit, a unique commit id will be generated.

Syntax: git commit -m "Your message"

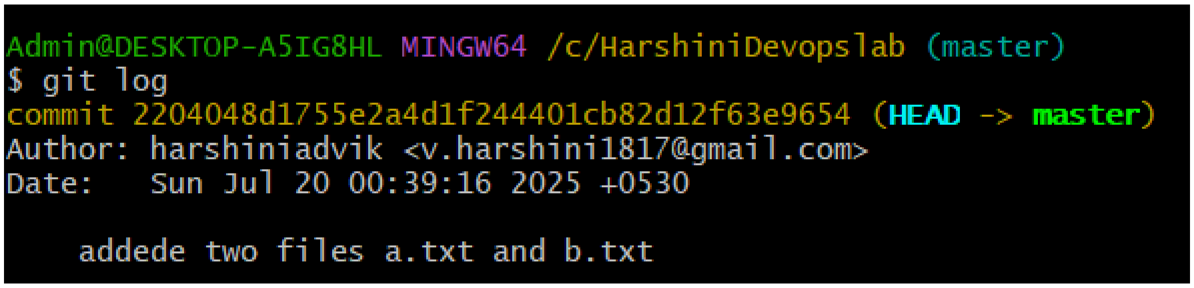


```
Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ git commit -m "addede two files a.txt and b.txt"
[master (root-commit) 2204048] addede two files a.txt and b.txt
 2 files changed, 2 insertions(+)
 create mode 100644 a.txt
 create mode 100644 b.txt

Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ git status
On branch master
nothing to commit, working tree clean
```

➢ git log:

- ❖ It shows the history of all commits.
- ❖ It provides commit id, author name,maild , timestamp and commit message.

Syntax: **git log**

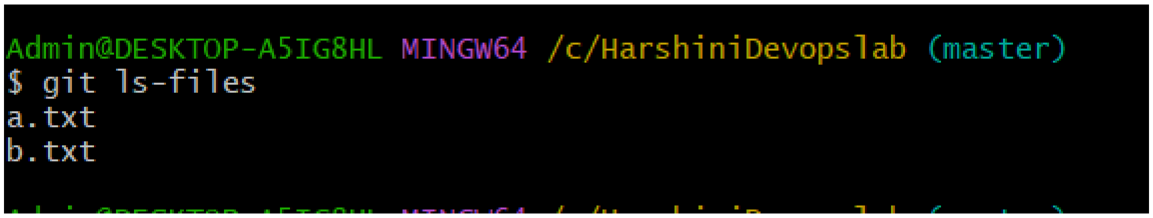


```
Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ git log
commit 2204048d1755e2a4d1f244401cb82d12f63e9654 (HEAD -> master)
Author: harshiniadvik <v.harshini1817@gmail.com>
Date:   Sun Jul 20 00:39:16 2025 +0530

    addede two files a.txt and b.txt
```

➢ git ls-files:

- ❖ This command will list out all files which are tracked by git.

Syntax: git ls-files

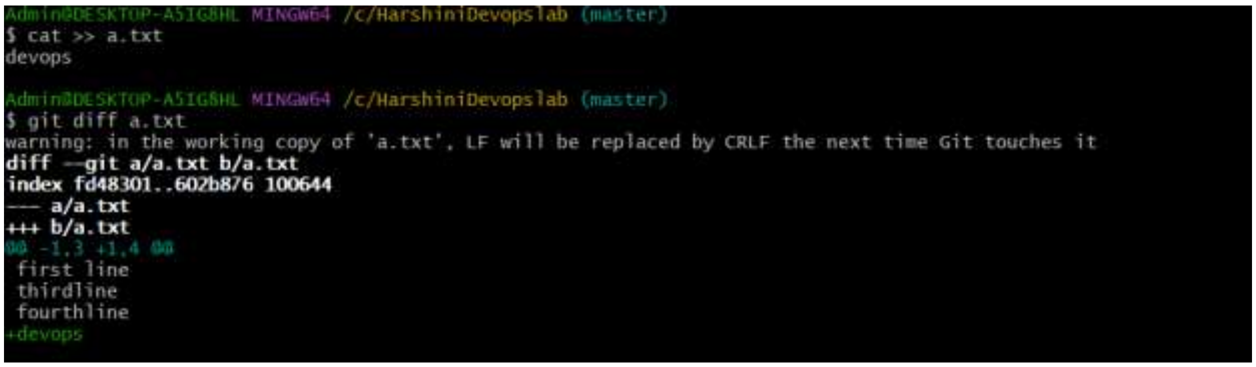


```
Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ git ls-files
a.txt
b.txt
```

➢ git diff :

- ❖ It is a very common requirement to find differences between the content of a particular file or all files.

Syntax:git diff

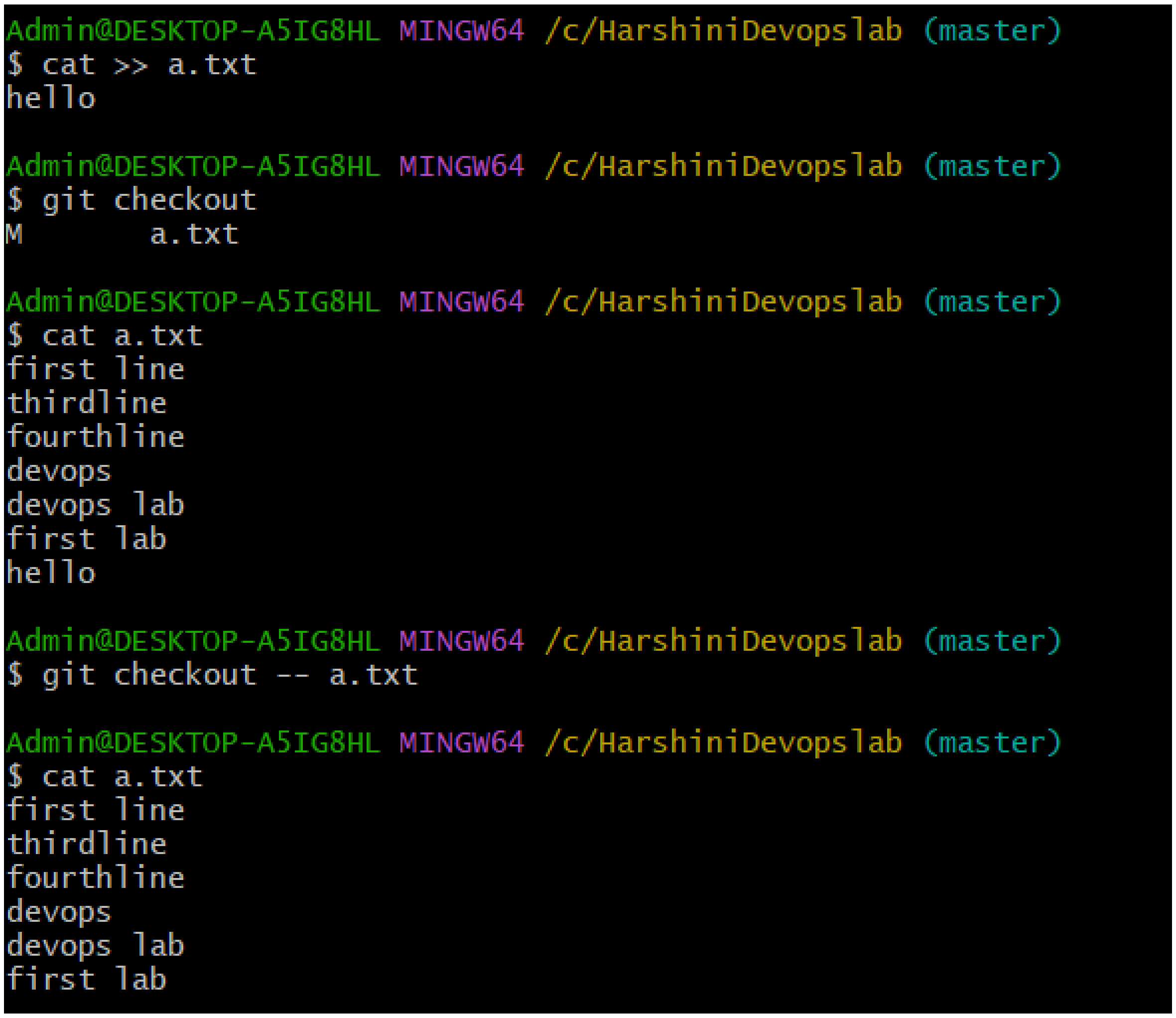


```
Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ cat >> a.txt
devops

Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ git diff a.txt
warning: in the working copy of 'a.txt', LF will be replaced by CRLF the next time Git touches it
diff --git a/a.txt b/a.txt
index fd48301..602b876 100644
--- a/a.txt
+++ b/a.txt
@@ -1,3 +1,4 @@
 first line
 thirdline
 fourthline
+devops
```

➢ git Checkout :

❖

- ❖ We can use the checkout command to discard unstaged changes in the tracked files of the working directory.
  Syntax: git checkout <branch-name>



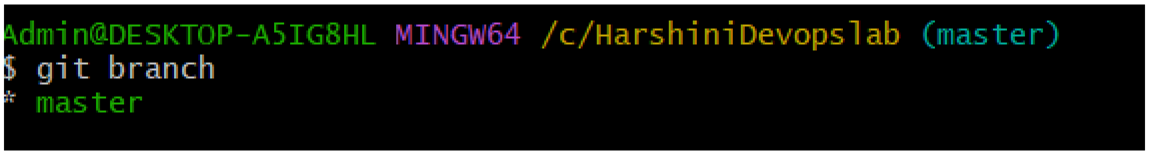
```
Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ cat >> a.txt
hello

Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ git checkout
M       a.txt

Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ cat a.txt
first line
thirdline
fourthline
devops
devops lab
first lab
hello

Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ git checkout -- a.txt

Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ cat a.txt
first line
thirdline
fourthline
devops
devops lab
first lab
```

➢ git branch:

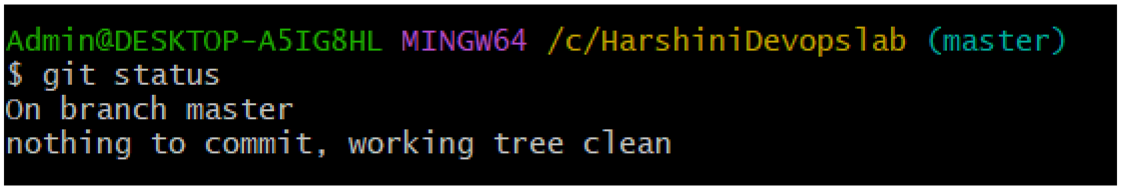
- ❖ It will show all branches in our local repository.
- ❖ By default we have only one branch: master
- ❖ master is the default name provided by GIT

syntax:

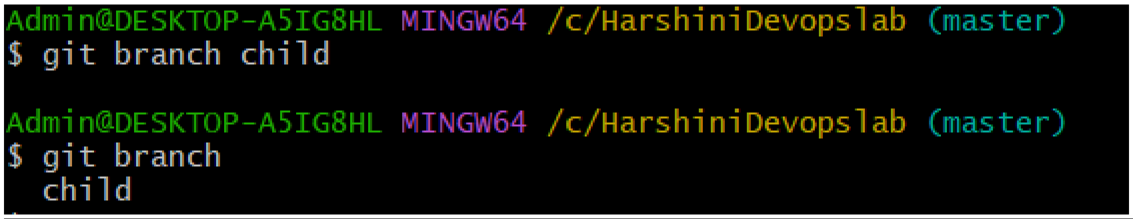
```
git branch    # List branches
git branch new-branch
git checkout new-branch
```



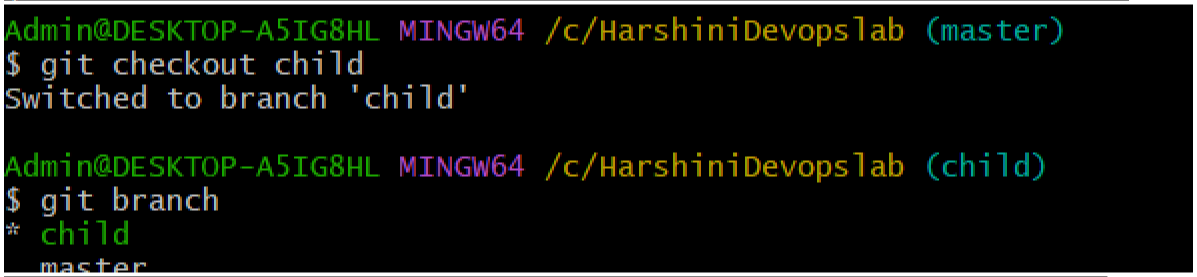
```
Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ git branch
* master
```



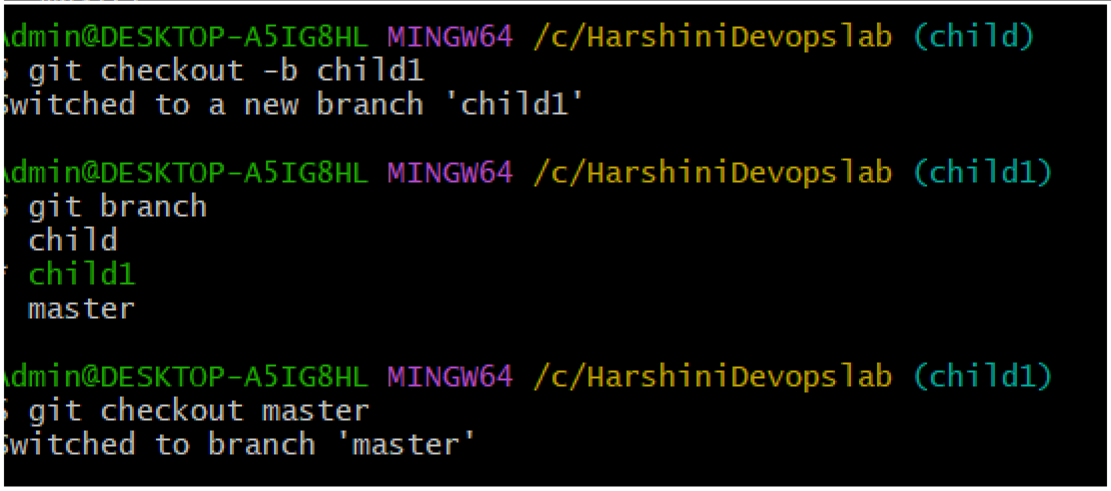
```
Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ git status
On branch master
nothing to commit, working tree clean
```



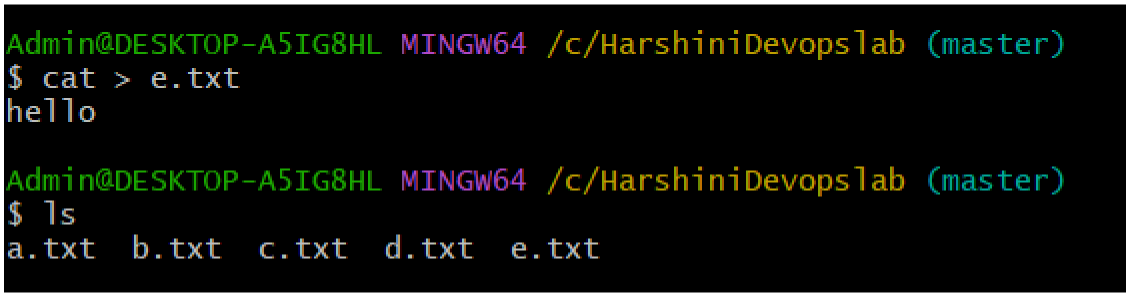
```
Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ git branch child

Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ git branch
  child
```



```
Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ git checkout child
Switched to branch 'child'

Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (child)
$ git branch
* child
  master
```



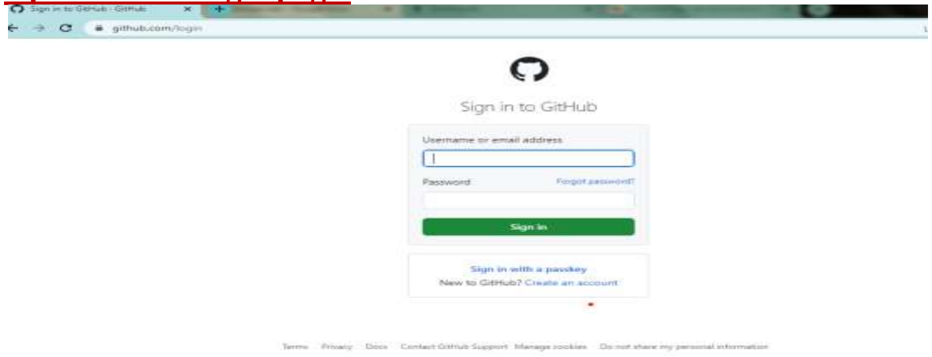
```
dmin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (child)
 git checkout -b child1
witched to a new branch 'child1'

dmin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (child1)
 git branch
 child
 child1
 master

dmin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (child1)
 git checkout master
witched to branch 'master'
```



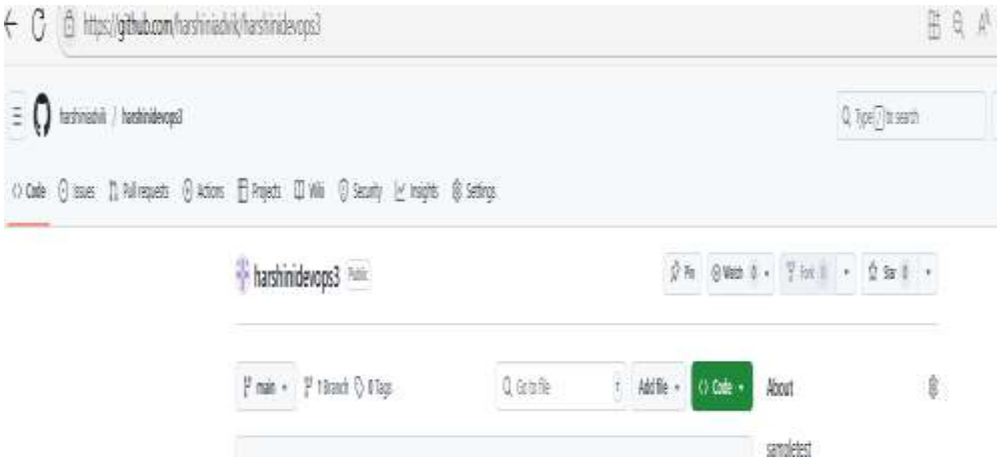
```
Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ cat > e.txt
hello

Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ ls
a.txt  b.txt  c.txt  d.txt  e.txt
```

Below is the process for workingdirectory(local repository)Git data is pushing to (remotedirectory) Github:

Open Git hub login page:



After login the git hub page we have to open repository which we have been created in the initial phase

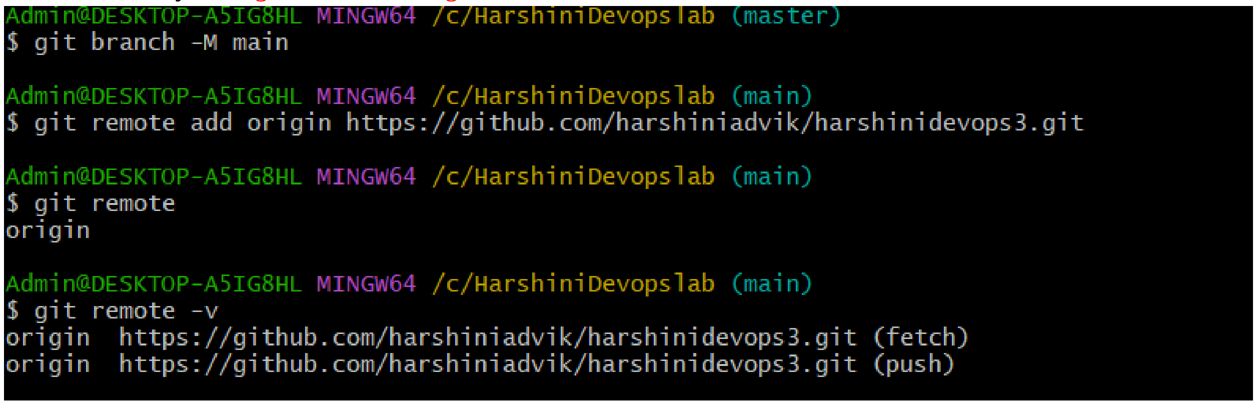


- **git remote:**

❖ We can use git remote command to configure remote repository to our local repository.

❖ It just provides the alias names of remote repositories

Syntax: git remote add origin <url>

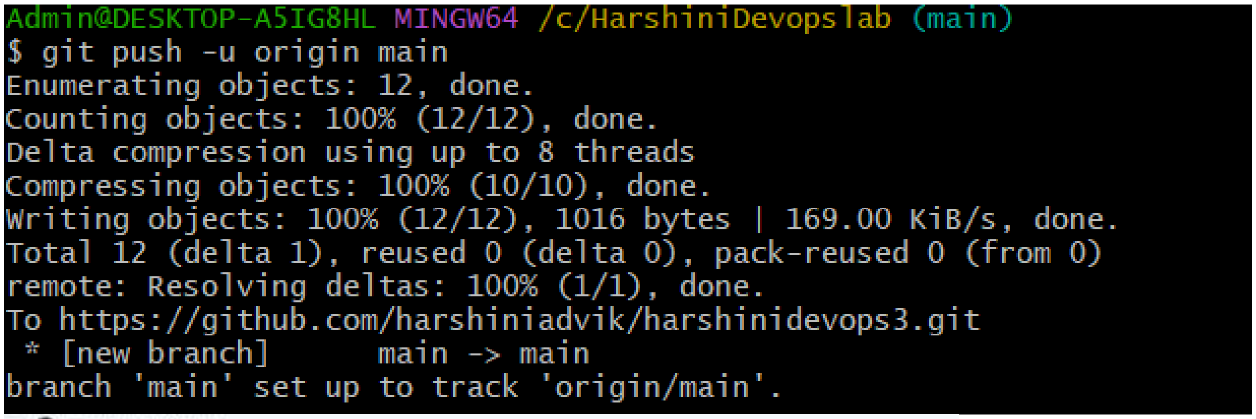


```
Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (master)
$ git branch -M main

Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (main)
$ git remote add origin https://github.com/harshiniadvik/harshinidevops3.git

Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (main)
$ git remote
origin

Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (main)
$ git remote -v
origin  https://github.com/harshiniadvik/harshinidevops3.git (fetch)
origin  https://github.com/harshiniadvik/harshinidevops3.git (push)
```

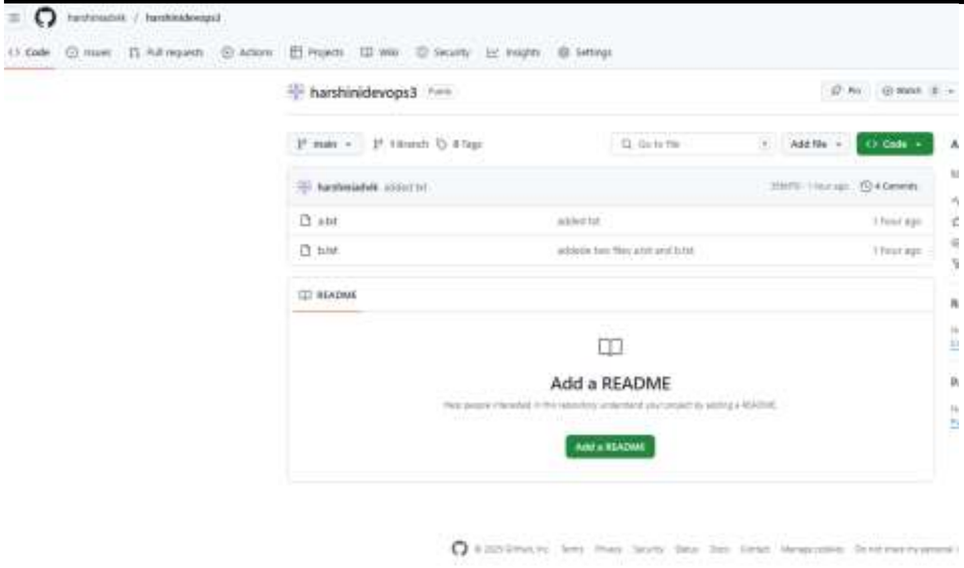
➢ git push:

❖ We can use the git push command to send our changes from local repository to remote repository. ie to push our changes from local repo to remote repo.

Syntax: git push -u origin main



```
Admin@DESKTOP-A5IG8HL MINGW64 /c/HarshiniDevopslab (main)
$ git push -u origin main
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 8 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (12/12), 1016 bytes | 169.00 KiB/s, done.
Total 12 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/harshiniadvik/harshinidevops3.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```



__By this we can see the workingdirectory(localrepository)Git data is pushed to (remotedirectory)-Github__

K.Harshini
DevOps(Cse)