Agenda

1. Session – I
    1.1. Introduction.
    1.2. Installation.
    1.3. Remote Setup.
    1.4. Local Setup.
    1.5. Repositories.
2. Session – II
    2.1. Typical Workflow.
    2.2. Moving across commits.
    2.3. Pushing code.
    2.4. Pulling code.
    2.5. Branches.

SESSION - I

Introduction

1. What is Git?

- distributed version control software.

2. Why do you need Git?

- Snapshots Across Time.

3. Who created Git?

- Linus Torvalds, circa 2005.

4. Who uses Git?

- Anybody can use git.
- But importantly, it is used by most of the Free and Open-Source community.
- Git helps democratise code.

Installation

1. **Linux:**

   [Debian/Ubuntu/Debian-derivatives]

   ```
   sudo apt install git-all
   ```

2. **Windows:**

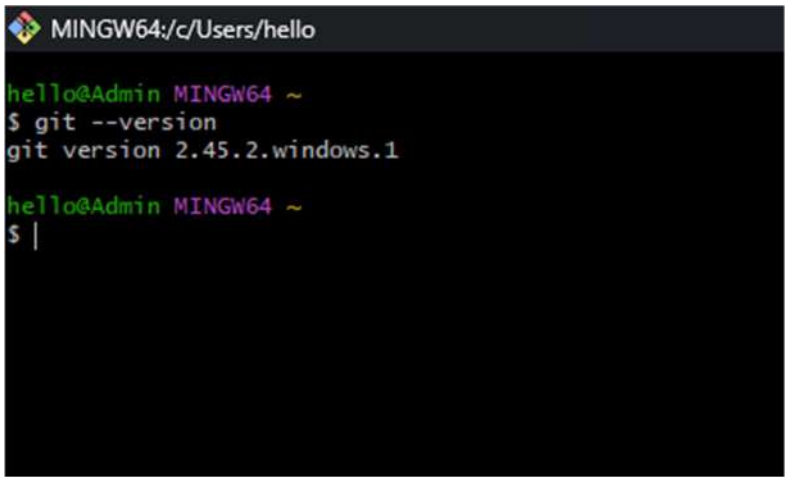
   a. **Goto** https://gitforwindows.org/

   b. **Click on Download** to download the Executable file.

   c. **Run** the executable file as an administrator.

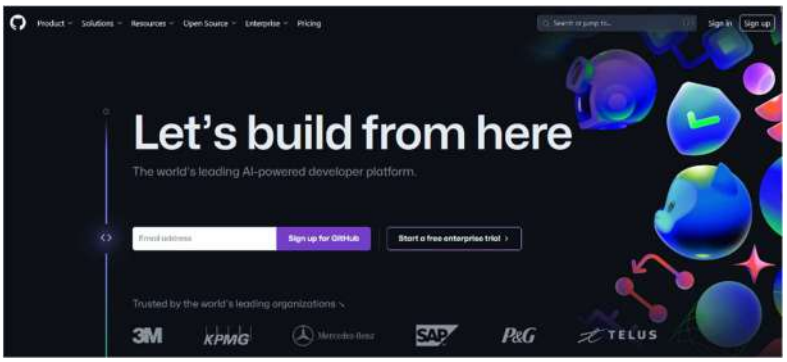
   d. **Open Git-Bash** and type the following command.

   ```
   git --version
   ```



Remote Setup

1. Git is distributed:
    - Git allows for multiple users to edit and modify versions, concurrently over a network.
    - Git supports remotes.
2. Git as a Service:
    - GitHub.
    - GitLab.
    - GitTea.
    - SourceHut.
3. Account Setup Git:
    a. **Goto** https://github.com/



    b. **Sign Up** for a new account.

    c. Provide the necessary information to **create an account.**

Local Setup

1. **Testing Git**: Enter the following command.

```
git --help
```

2. **Initial Git Configuration**: Open the Git-bash and enter the following

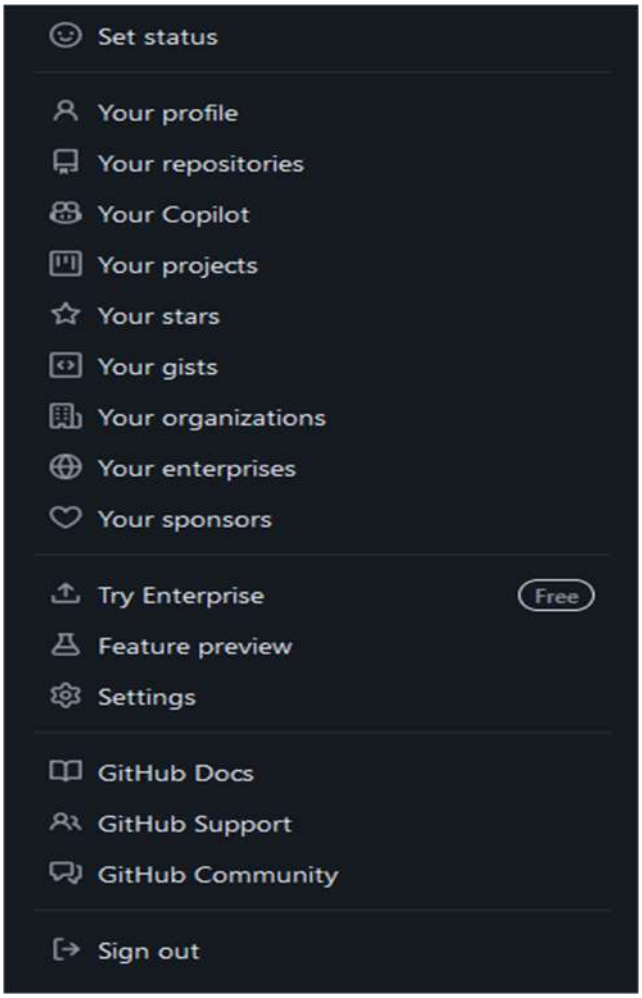
```
#Telling Git your name
git config --global user.name "Your Name!"

#Telling Git your email
git config --global user.email "your-email@addre.ss"

#Showing the basic configurations
git config --list
```

Repositories

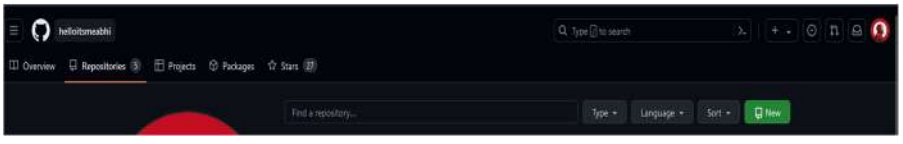
1. What are Repositories?
    - Environments.
    - projects.
    - what git deals with.
    - A repository is a 'collection of source code files' that can be hosted and collaboratively edited.
2. Repositories are made up of 3 layers:

3. Two ways of getting a repository:
    - create your own.
    - use existing.

4. Creating a repository:
    a. Goto your repositories in right side bar and click on **Your Repositories**.



    b. Click on new to create a repository.



    c. Check the necessary options and create a repository.

5. Using an existing repository:

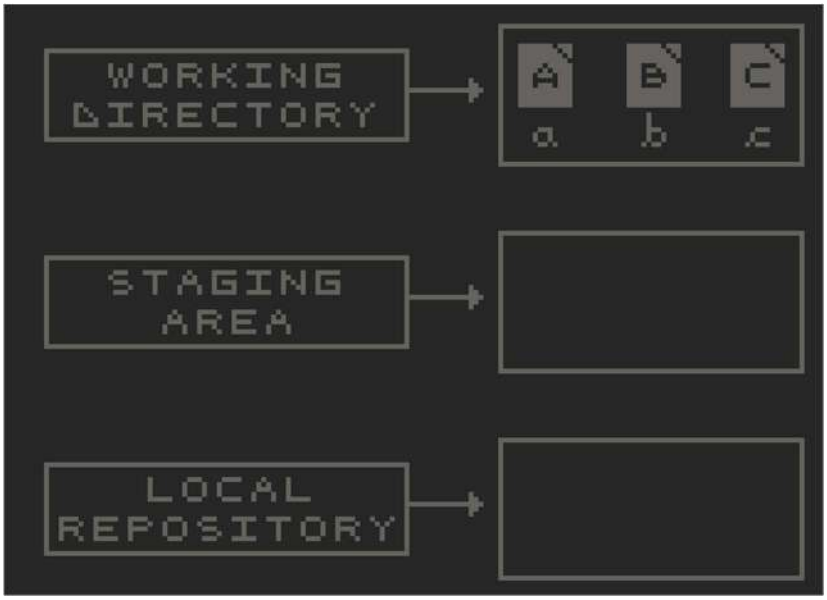
   a. Use the following commands to use the existing repository.

```
#Cloning your repository
git clone gitlab.com/<username>/first-repo

#Navigate
cd first-repo

#Checking current status
git status
```

```
C:\Users\hello\OneDrive\Desktop>git clone https://github.com/helloitsmeabhi/first-repo
Cloning into 'first-repo'...
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 13 (delta 2), reused 10 (delta 2), pack-reused 0
Receiving objects: 100% (13/13), done.
Resolving deltas: 100% (2/2), done.

C:\Users\hello\OneDrive\Desktop>cd first-repo

C:\Users\hello\OneDrive\Desktop\first-repo>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

   b. Creating Files.

```
#Create file called 'a'
echo "A" > a


#Create file called 'b'
echo "B" > b


#Create file called 'c'
echo "C" > c


#Checking status
git status
```

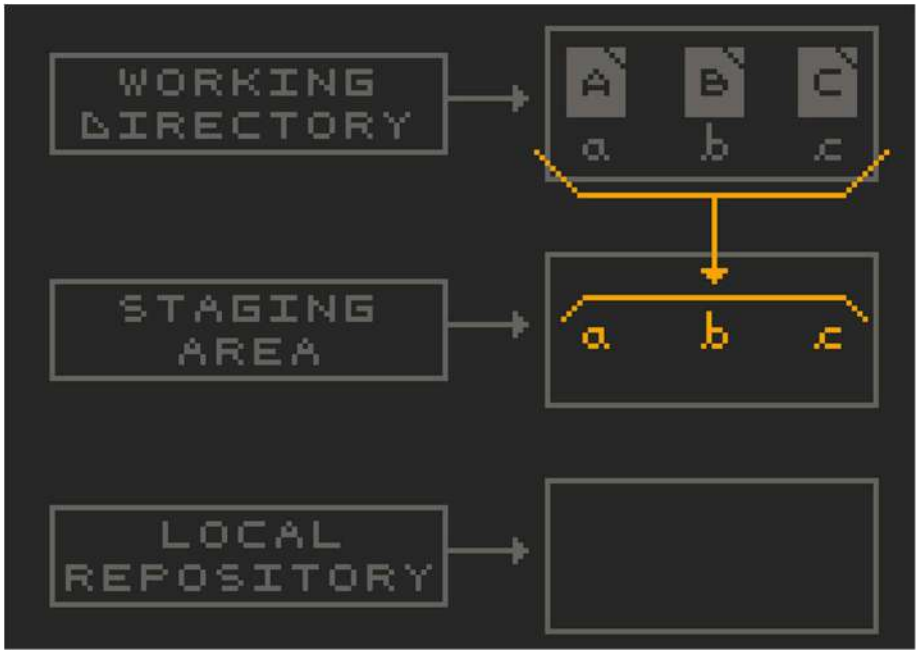


c. Staging: Enter the following commands for staging.

```
#Add file

git add a

git add b

#Check Status

git status

#Remove File

git rm --cached b
```

d. Adding all to staging area.

```
#Adding files manually can take time
#Sometimes you want to add the whole repo to
staging

git add --all

#or

git add .
```



e. Commits:

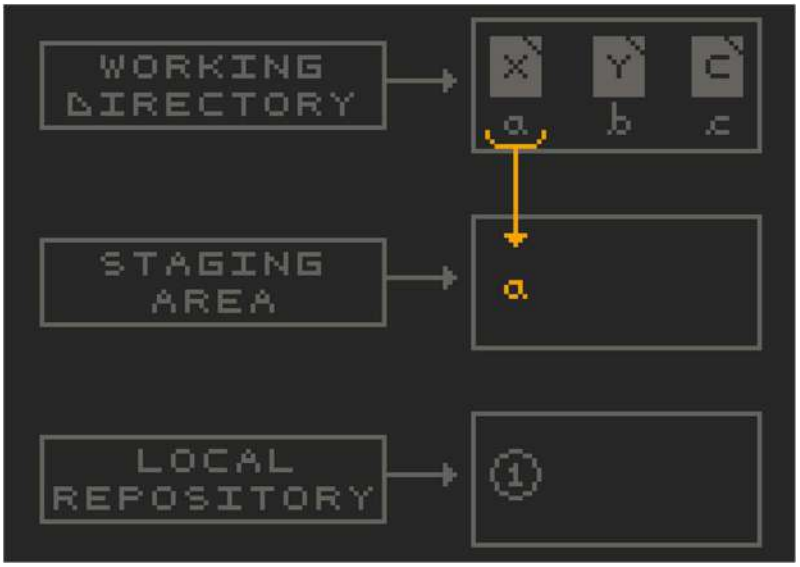
- Savepoints.
- Snapshots.
- Each commit == different version.

f. Creating a commit: use the following commands

```
#Lets recheck git status
git status

#Creating a commit
git commit -m "1 - Start"

#Lets check log
git log
```

g. Making modifications:

```
#Rechecking status
git status

#Modifying a
echo "X" > a

#Modifying b
echo "Y" > b

#Add a
git add a
```

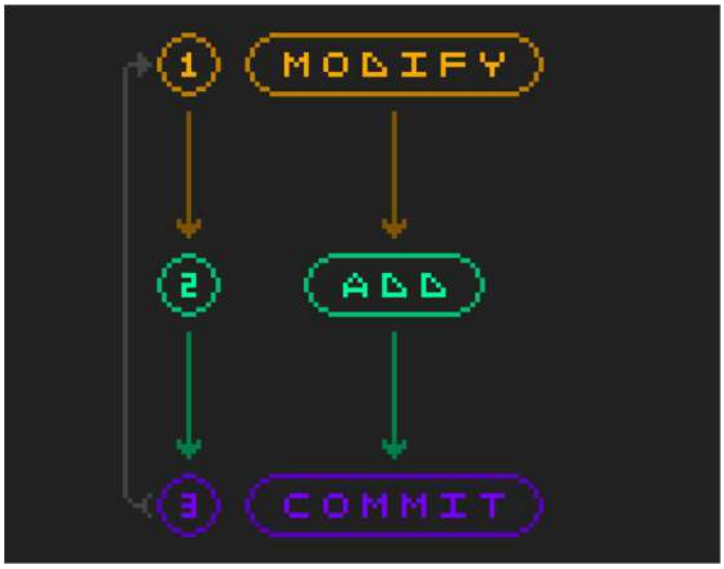


h. Tracked files are in any 1 of 3 states at any given time:

- modified
- staged
- committed

SESSION - II

Typical Workflow



Moving across commits

1. the following commands are used to move across commits

```
#Check Log
git log --graph
#Move to previous commit
git checkout <hash>
#Move back to latest commit
git checkout main
```

```
C:\Users\hello\OneDrive\Desktop\first-repo>git log --graph
* commit 05afd440218f00c132f3ce3633e6f7cae53c5ab1 (HEAD -> main, origin/main, origin/HEAD)
| Author: helloitsmeabhi <helloitsmeabhinav2003@gmail.com>
| Date:   Sat Jun 29 12:14:16 2024 +0530
|
|     2nd commit
|
* commit 814417fd4a6db3afcd0e4529b49e91190a15a593
| Author: helloitsmeabhi <helloitsmeabhinav2003@gmail.com>
| Date:   Sat Jun 29 12:08:18 2024 +0530
|
|     removed b
|
* commit 3cc64ce6f31c81e8ff18b64d5fcec6fecbc89bb2
| Author: helloitsmeabhi <helloitsmeabhinav2003@gmail.com>
| Date:   Sat Jun 29 11:54:45 2024 +0530
|
|     1-start
|
* commit 5c42afb8f4a2216d83f9663b05b3680a3a2d38d5
  Author: helloitsmeabhi <95755633+helloitsmeabhi@users.noreply.github.com>
  Date:   Sat Jun 29 10:48:43 2024 +0530

      Initial commit
```

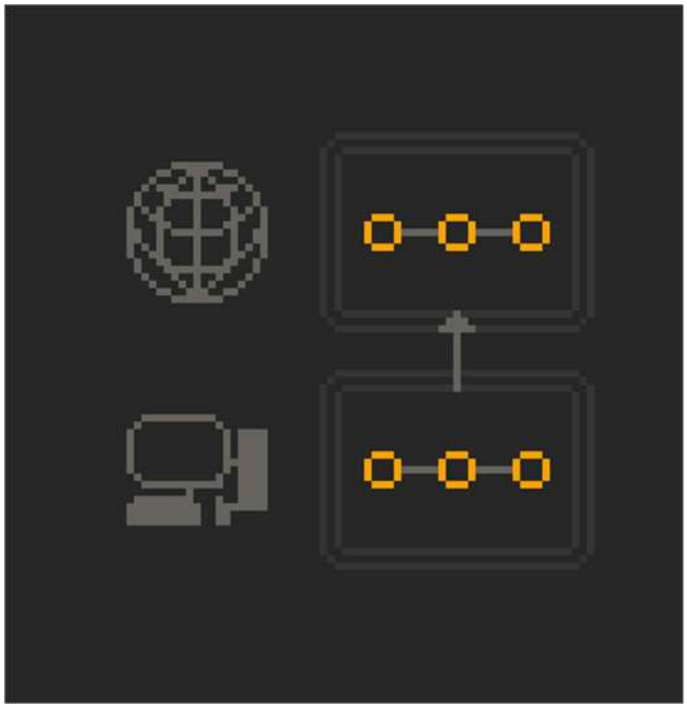
Pushing code

- Pushing code is how you "upload".
- Synchronizing from local to remote.

1. The following command is used to push to remote repository.

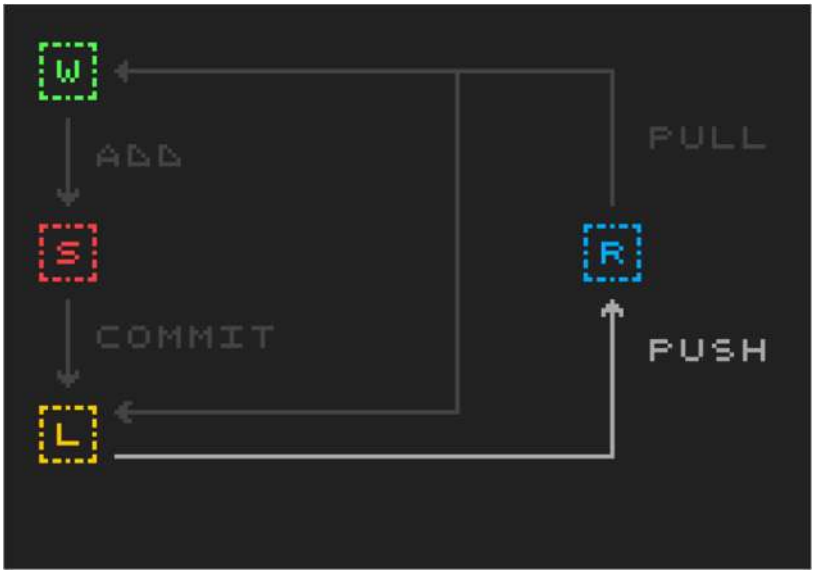
```
Git push
```

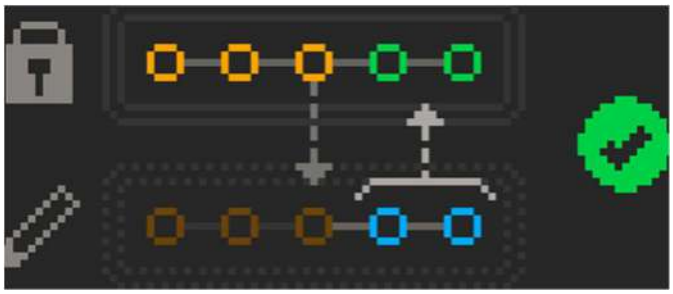
```
C:\Users\hello\OneDrive\Desktop\first-repo>git commit -m "new add"
[main 460ff1e] new add
 6 files changed, 1 insertion(+), 1 deletion(-)
 delete mode 100644 README.md
 delete mode 100644 ab
 create mode 100644 b
 delete mode 100644 hello.txt

C:\Users\hello\OneDrive\Desktop\first-repo>git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (5/5), 311 bytes | 62.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/helloitsmeabhi/first-repo
   05afd44..460ff1e  main -> main
```



Pulling code

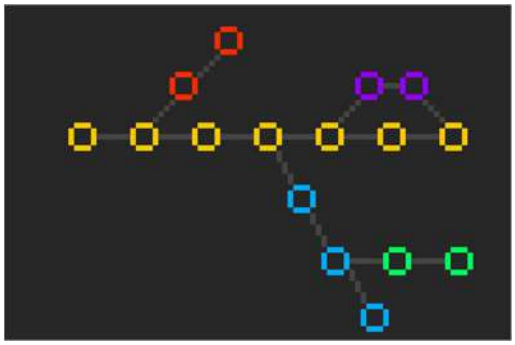
- Synchronizing from remote to local.





Branches

1. What are branches?
   - series of commits.
   - commits that diverge from a common origin.



2. Creating Branches: the following command is used.

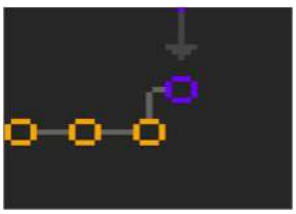
```
#List Branches

git branch -a


#Create Branch

git branch new
```

3. Switching branches:

```
# Switching syntax :

# git switch <branch name>

# git checkout <branch name>

# Switching to the branch we just created

git switch new
```

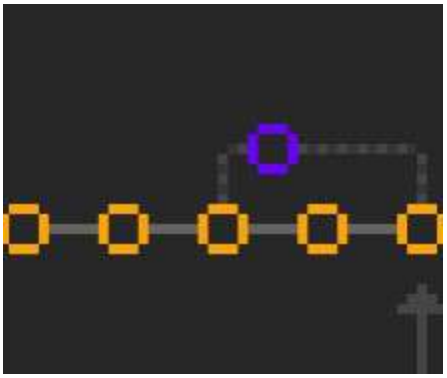
4. Changes to new branch:

```
# Creating C again

echo "C" > c

# Rechecking status

git status

# Adding all to staging

git add .

# Making commit

git commit -m "n0"
```



5. Merging Branches:

```
# Switching back to main
git switch main
# Creating file 'd'
echo "D" > d
# Adding d
git add .;
# Committing
git commit -m "4"
#Merging new branch with main
git merge new -m "5"
```



Photos of the Session


GPS Map Camera
Bengaluru, Karnataka, India
APS COLLEGE OF ENGINEERING Somanahalli, Kanakapura Road, Bengaluru, Karnataka 560082
Lat 12.783406°
Long 77.498351°
29/06/24 10:10 AM GMT +05:30
Google


GPS Map Camera
Bengaluru, Karnataka, India
APS COLLEGE OF ENGINEERING Somanahalli, Kanakapura Road, Bengaluru, Karnataka 560082
Lat 12.783406°
Long 77.498351°
29/06/24 10:10 AM GMT +05:30
Google


GPS Map Camera
Bengaluru, Karnataka, India
APS COLLEGE OF ENGINEERING Somanahalli, Kanakapura Road, Bengaluru, Karnataka 560082
Lat 12.783406°
Long 77.498351°
29/06/24 10:10 AM GMT +05:30
Google


GPS Map Camera
Bengaluru, Karnataka, India
APS COLLEGE OF ENGINEERING Somanahalli, Kanakapura Road, Bengaluru, Karnataka 560082
Lat 12.783406°
Long 77.498351°
29/06/24 10:10 AM GMT +05:30
Google