Git

Module Objective

1. Basic Concept about Git
2. Git Diagram
3. Git Installation
4. Git Configuration Initially
5. Git Initialization
6. Add new Files, Update Files and Remove Current Changes
7. Git Commit

Basic Concept about Git

Git is a popular version control system designed to make it easier to have multiple versions of a code base, sometimes across multiple developers or teams. It allows us to track changes we make to our files over time and easily revert them.

Git is installed and maintained on your local system rather than in the cloud. Git Repository is a place where Git can store versions of our files.

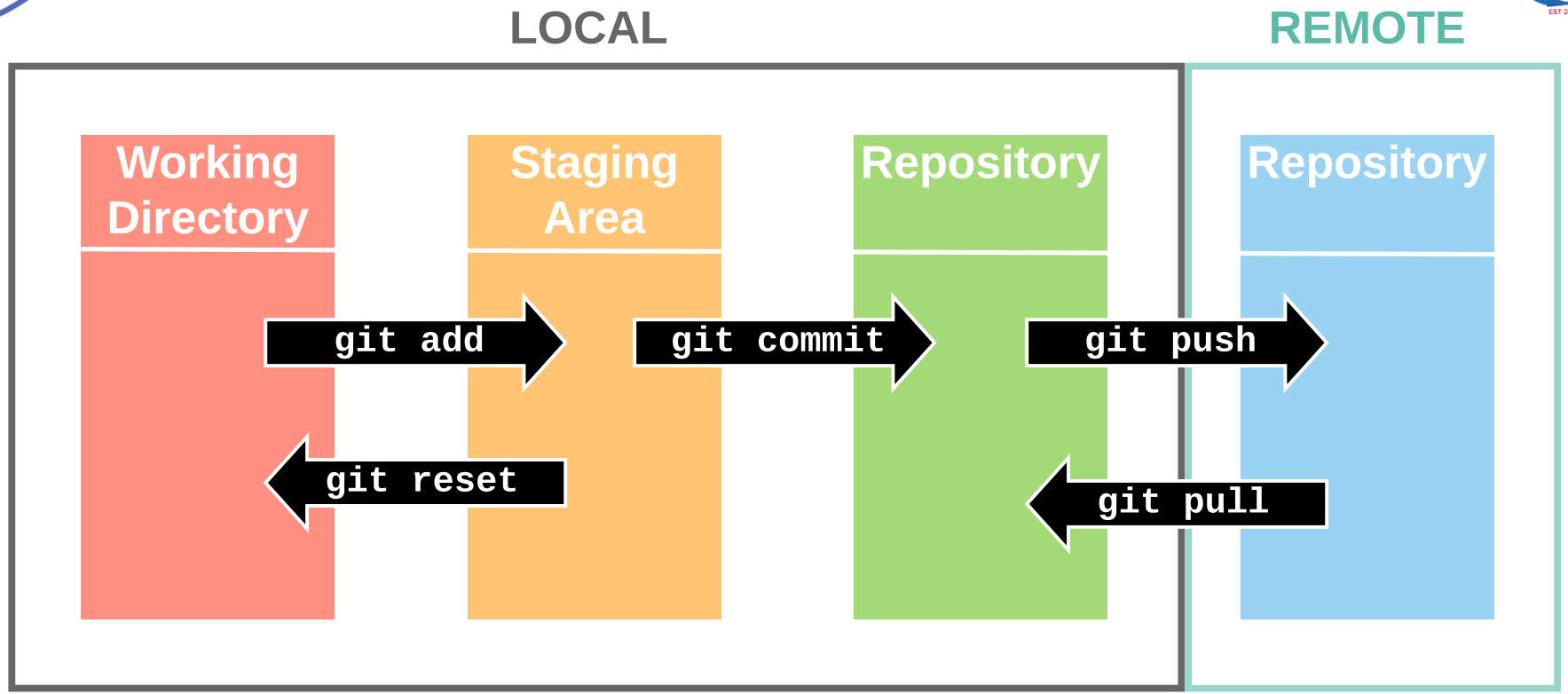
It was created by Linus Torvalds in 2005 and has been maintained by Junio Hamano since then.

Basic Concept about Git

It is used for:
Tracking code changes
Tracking who made changes
Coding collaboration with other developers

Why Git?
Over 70% of developers use Git!
Developers can work together from anywhere in the world.
Developers can see the full history of the project.
Developers can revert to earlier versions of a project.

Git Diagram



Download and Install Git

In order to use Git, we have to install it on our computer. We can download the latest version of Git for free from the following official website. You can download for your operating system from the options given.

https://www.git-scm.com/downloads

Download and Install Git

For Windows, you can use Git bash, which comes included in Git for Windows. For Mac and Linux, you can use the built-in terminal.

[Ubuntu] Install at your workstation, follow the default options.

```
$ sudo apt -get update & $ sudo apt -get install git-core
```

To verify the Git installation, we can run this command. This will show you the current version installed on your PC.

```
$ git --version
```

Configure Git Initially

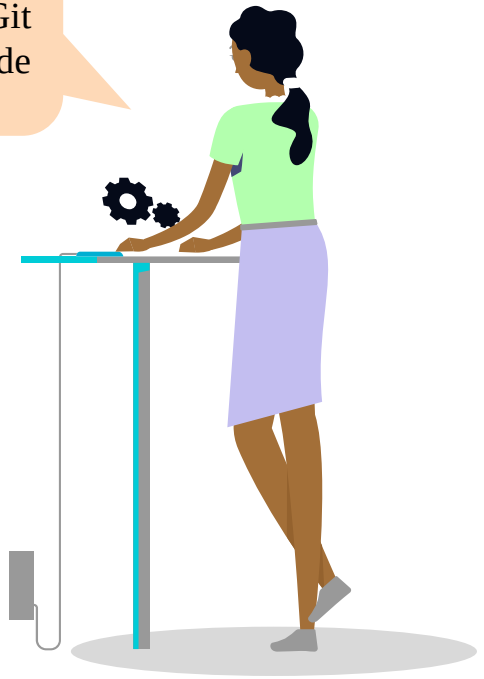
Now let Git know who you are. This is important for version control systems, as each Git commit uses this information. You need to register your username and email address in your Git repository. Git will use this information to identify who made specific changes to files.

```
$ git config --global user.name "Nahid Hasan“

$ git config --global user.email "nahid.hasan@bjitgroup.com"
```

// To verify these are set

```
$ git config –list
```



Create and Initialize a Project in Git

It is now time to create our project. This will tell Git to get ready to start watching our files for every change that occurs.

We can navigate to our new project's location and create new project folder:

```
$ mkdir test-project
```

// Navigate to project working directory.

```
$ pwd
$ cd test-project
```

Create and Initialize a Project in Git

Initialize Git

Once you have navigated to the correct folder, you can initialize Git on that folder:

```
$ git init
```

This will create a .git directory in your current directory. Then you can commit files in that directory into the repository.

// Check our current working directory

```
$ ls -a
```

Add new Files, Update Files and Remove Current Changes

Let's add/create some files in the current working directory. Then we can check the Git status and see if it is a part of our repository:

```
$ touch test.txt
$ git status
```

Now Git is aware of the file, but has not added it to our repository!

Files in your Git repository folder can be in one of 2 states:

- Tracked - files that Git knows about and are added to the repository
- Untracked - files that are in your working directory, but not added to the repository.

When you first add files to an empty repository, later you will change/add files, they are all untracked. To get Git to track them, you need to stage them, or add them to the staging environment.

Add new Files, Update Files and Remove Current Changes

Every time before we commit changes, we must add it to the staging area. Only staged files will be committed. Now, we can add files to the Staging Environment.

// Add a single file to the Staging Environment

```
$ git add test.txt
```

// Add multiple files to the Staging Environment

```
$ git add index.html bluestyle.css
```

// Add all files (new, modified, and deleted) in the current directory to the Staging Environment

```
$ git add --all
```

Add new Files, Update Files and Remove Current Changes

// Add all tracked files. Please note that this will not add your untracked or new files.
$ git add -a

// Add all tracked and untracked files
$ git add -A

The file should be Staged. Let's check the status:
$ git status

// To see what is modified but unstaged
$ git diff

// To see a list of staged changes
$ git diff --cached

// To remove the staged file, but keep the file in working directory
$ git rm --cached <filename>

Git Commit

When all of our changes are added in the staging area, we can commit using the following command.

$ git commit -m "Your commit message here. Refs #22333"

The first part of the command git commit tells Git that all the files staged are ready to be committed. The second part -m "first commit" is the commit message. -m is shorthand for message while the text inside the parenthesis is the commit message.

//To add all of your tracked files into stage and commit in a single command
$ git commit -a -m "Your commit message here. Refs #22333"

// To see a log of all changes in your local repository
$ git log or git log --oneline (shorter version)

// To show only the 5 most recent updates
$ git log -5

// To update your last commit without making a new one, use the following command
$ git commit -a --amend -m "Your amend message here"

Thank You