

# GIT

BY VIGNESH S





# Introduction

**GIT** is a distributed version control system (DVCS) that helps developers manage and track changes to their code. It is an essential tool for software development

**Version Control:** Git allows you to keep track of changes made to your code over time.

**Collaboration:** Git enables multiple developers to work on the same project simultaneously.

**Backup:** With Git, you always have a backup of your code.





# Local Repositories

A local repository is the copy of your project stored on your computer.

It contains all the files and the complete history of your project.

You can work on your code, make changes, and commit them to your local repository.

**Clone:** To start working with Git, you first clone a remote repository to create a local copy on your machine.






# Local Repository

**Edit:** You make changes to the code in your local repository.

**Commit:** After making changes, you commit them to your local repository. This creates a snapshot of your code at that point in time.

**Branch:** You can create branches to work on different features or fixes independently.

**Merge:** Once you're satisfied with your changes, you merge them into the main branch.





# Remote Repository

A remote repository is a copy of your project stored on a server or in the cloud.

It serves as a centralized location for collaboration and backup.

**Push:** After committing changes to your local repository, you push them to the remote repository.

This makes your changes available to other team members.





# Remote Repositories

**Pull:** To get the latest changes made by others, you pull from the remote repository to update your local copy.

## Fork and Pull Request:

In open-source projects,

contributors often fork a repository, make changes in their forked version, and then create pull requests to suggest changes to the original project.



## Remote Repository

```
<html>
  <body>
    <h1>My Website!</h1>
  </body>
</html>
```



## Local Repository



```
<html>
  <body>
    <h1>My Website!</h1>
  </body>
</html>
```

## Local Repository



```
<html>
  <body>
    <h1>My Website!</h1>
    <p>This is some more text!</p>
  </body>
</html>
```

# Local

# Remote

Working  
Directory

Staging  
Area

Git  
Repository

GitHub

clone

setup

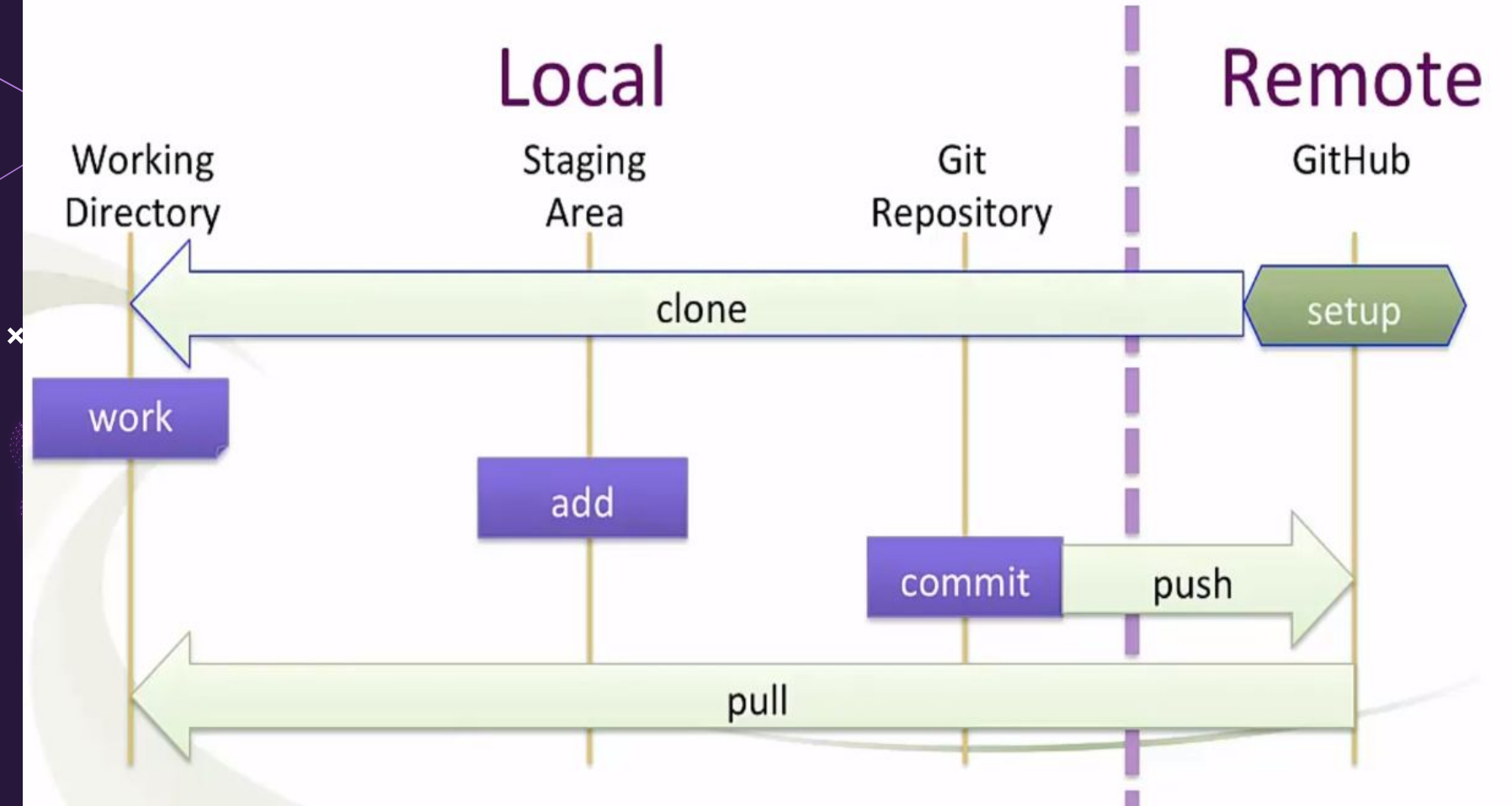
work

add

commit

push

pull





Fixed button CSS

@max



12.23AM

Changed background image

@sara



11.15AM

Added sub title

@tej



11.05AM

Fixed buttons in navigation

@max



10.45AM

Added front page

@sara



10.24AM



# Getting started with GIT

Git -version

Git init

Git status

Git add

Git commit -m <comment>

Git restore

```
Echo "<name>" >  
.gitignore
```

```
Git rm -cached <name>
```



x

## Working Area

Git init

Git Status

## Staging Area

Git add

## Committed Files

story1.txt

Git commit -m

x




# Branches

In Git, branches are a fundamental concept. They are like separate lines of development within your project.

Each branch can have its own set of commits and changes, allowing multiple work streams to progress independently without interfering with each other.

**Main Branch (usually master or main):** This is the default branch when you create a Git repository. It typically represents the stable version of your project.






# Branches

**Feature Branch:** Developers create feature branches when they want to work on a new feature or bug fix. These branches branch off from the main branch, and changes made in the feature branch do not affect the main branch until they are merged.

**Release Branch:** Before releasing a new version of your software, you might create a release branch. This branch allows you to stabilize and prepare the code for release without disrupting ongoing development in other branches.






# HEAD

The term "HEAD" in Git refers to a special pointer or reference that points to the latest commit in the currently checked-out branch or snapshot of your code.

Current Snapshot: HEAD represents the commit that you currently have checked out in your working directory.

Switching Branches: When you switch to a different branch, HEAD updates to point to the latest commit in that branch.



×

```
# Create a new branch
$ git branch sarah
# Switch to an existing branch
$ git checkout sarah

# Create a new branch and Switch to it
$ git checkout -b max

# Delete a branch
$ git branch -d max

# List all branches
$ git branch
```

×



# GIT Push

Git **push** is a command used in Git, a version control system, to send your local changes (commits) to a remote repository.

**Local Repository:** You make changes to files in your local Git repository, such as adding, modifying, or deleting files.

**Staging:** You add these changes to the staging area using `git add`. This tells Git which changes you want to include in the next commit.








# GIT Push

**Commit:** You create a commit using `git commit`, which records the changes along with a commit message explaining what you did.

**Push:** Now, when you want to share your changes with others or update a remote repository, you use `git push`.

This command uploads your local commits to the remote repository, making your changes accessible to others.






# GIT Pull

Git **pull** is used to fetch and merge changes from a remote repository into your local repository. It combines two actions:

**Fetch:** When you run `git pull`, Git contacts the remote repository and retrieves any new commits or changes that have been made there since your last interaction.

**Merge:** After fetching the changes, Git will automatically merge them into your current local branch, if possible. If there are conflicts between your local changes and the remote changes, Git will ask you to resolve them.






# Merging

Merging is a fundamental operation that allows you to combine changes from one branch into another. It's a critical process for integrating new features, bug fixes, or changes made in separate branches back into the main branch.

**Branch Creation:** Developers create separate branches (e.g., feature branches) to work on specific tasks or features without affecting the main codebase.

**Commit Changes:** Developers make changes and commit them to their respective branches. Each branch represents a different line of development.





# Merging

Preparing for Merge: When a feature is complete or a bug is fixed, the changes need to be incorporated into the main branch (e.g., **master or main**). To do this, you first ensure your branch is up-to-date with the latest changes from the main branch by using the **git pull** command.

Initiate Merge: You switch to the main branch where you want to merge the changes (e.g., **git checkout main**). Then, you initiate the merge using the git merge command, specifying the branch you want to merge from (e.g., git merge feature-branch).

