

Web & Mobile App Development

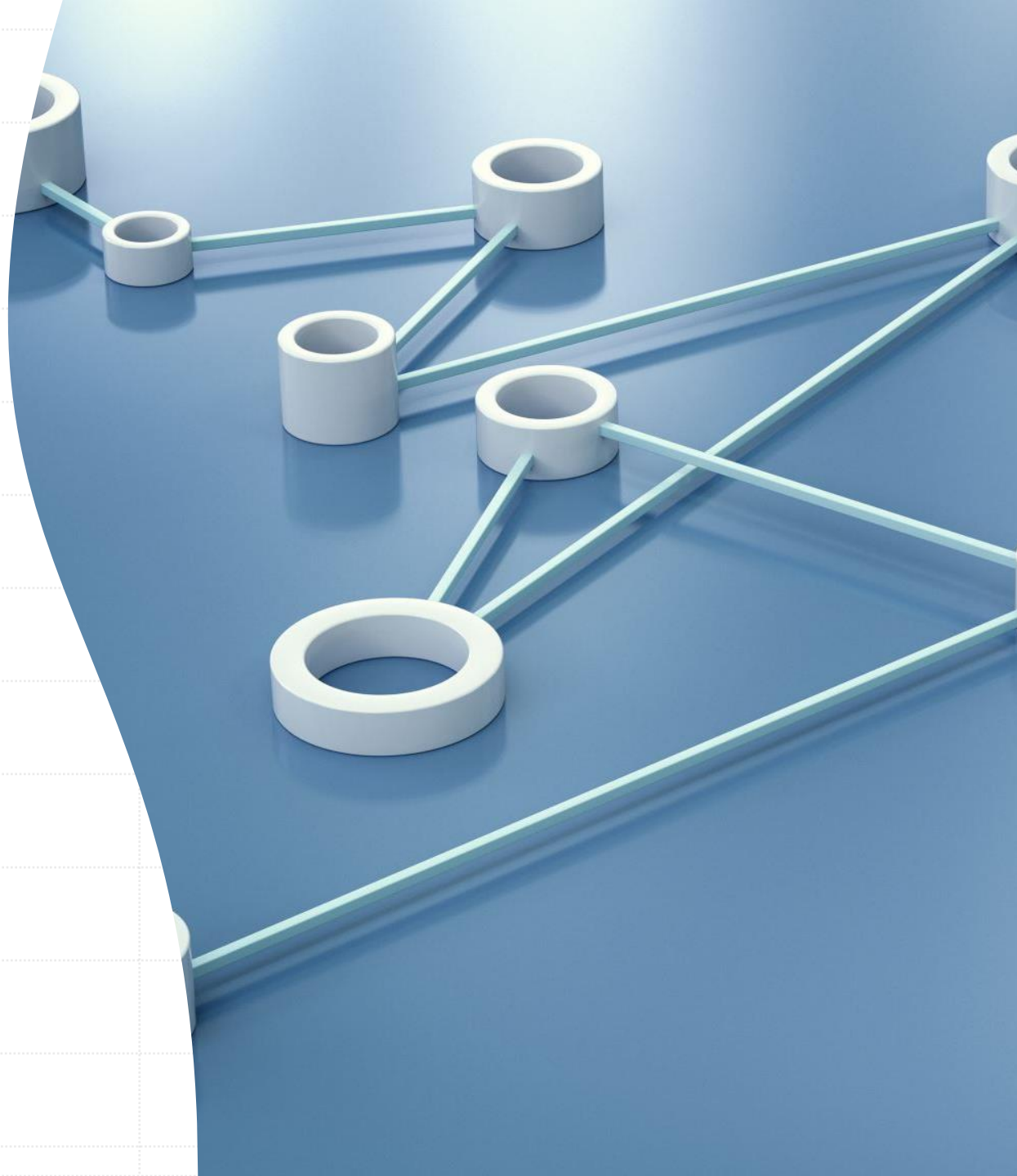
WMA – Lec # 14



Version Control System

Introduction

Version control is a system that **keeps track of changes** made to a set of files over time. It allows developers to easily collaborate on a project, revert changes if necessary, and maintain different versions of the project.



Git




There are several different types of version control systems (VCS), but the most widely used is Git.



Why use it?

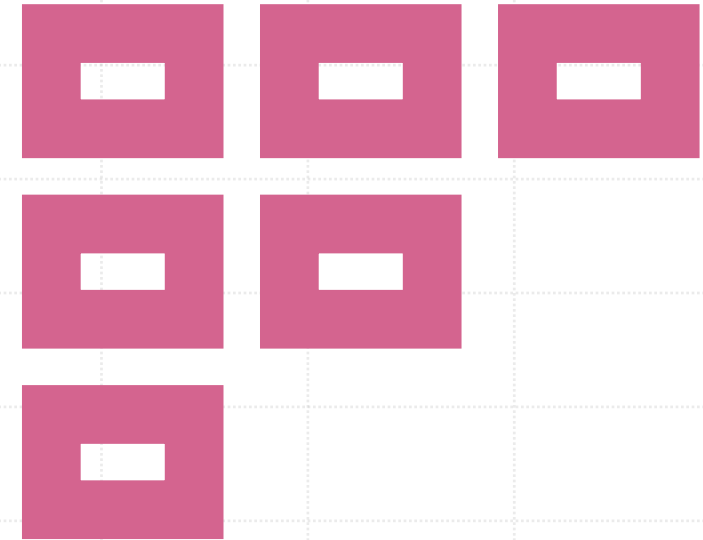
One of the main benefits of using version control is that it allows multiple people to work on the same project at the same time without overwriting each other's changes.



It also helps to keep a **history of changes** made to the project, so developers can easily see what has been changed, who made the changes, and why.

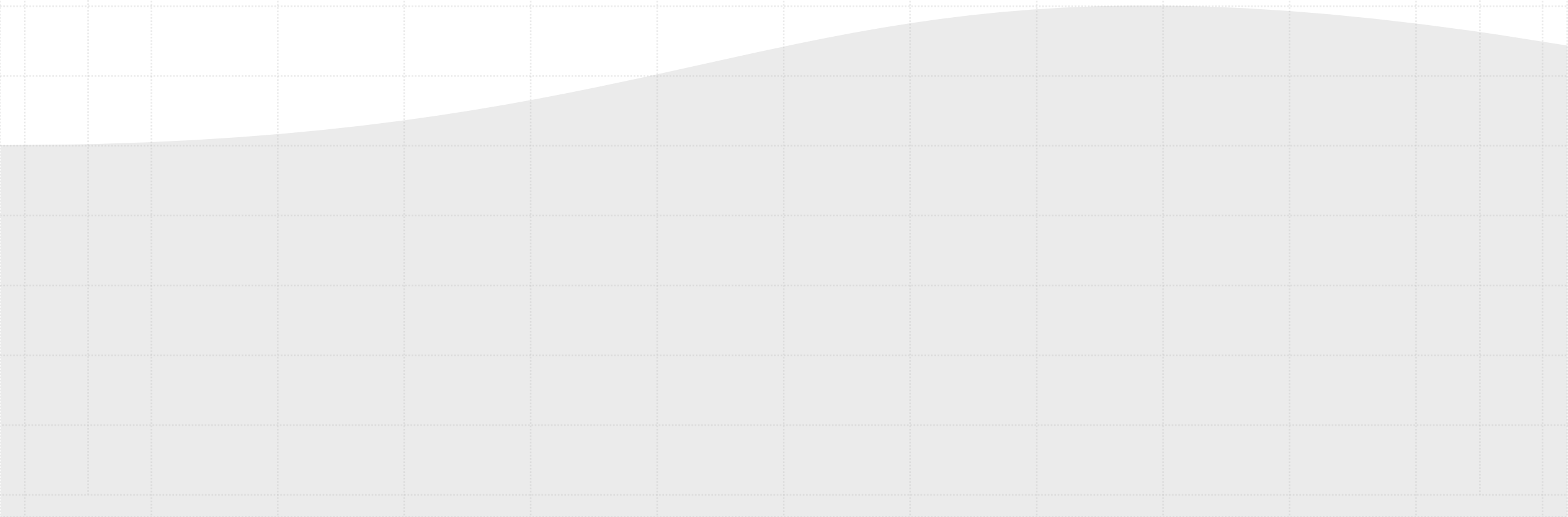
This makes it easy **to revert to a previous** version of the project if a mistake is made or if changes are not working as expected.

It also helps in understanding the **progress of the project** and what is going on in the development process.






Git & Github





Git is a distributed version control system (DVCS) that allows developers to track changes made to their code and collaborate on projects. It was created by Linus Torvalds in 2005.



Git is a command-line tool that allows developers to create local repositories on their computer and track changes made to the files in those repositories.



These changes can then be pushed to a remote repository, such as one hosted on Github.

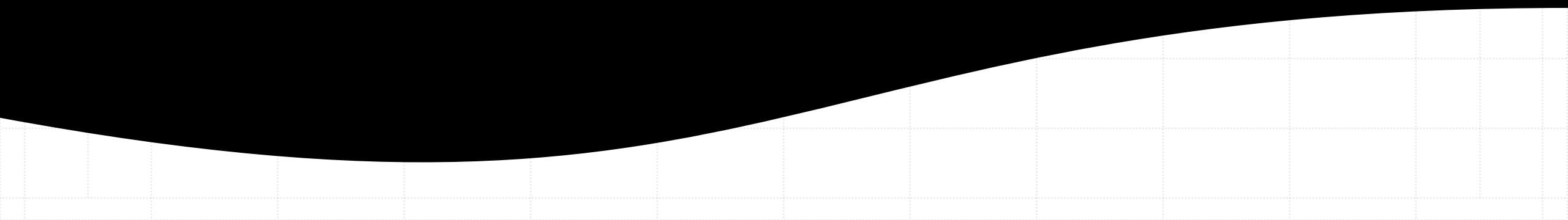
Github is a web-based platform that provides hosting for Git repositories. It was founded in 2008 and was later acquired by Microsoft in 2018.

Github

It provides a user-friendly interface for developers to view, manage, and collaborate on Git repositories.

In short, Git is a version control system that allows developers to track changes in their code, while Github is a web-based platform that allows developers to host and share their Git repositories.

Developers can use Git to manage their code locally, and then use Github to share their code with others.





Setting up Git on your computer

- Git can be installed on a variety of operating systems including Windows, Mac, and Linux.
- Download the latest version of Git from the official website(<https://git-scm.com/downloads>)



Run the installer

Follow the prompts to complete the installation.

To check if Git is installed and to check the version, open Git Bash (installed with Git) and run the command `git --version`.

Creating a Github account

- To create a Github account, go to the Github website (<https://github.com/>) and click on the "Sign up" button.



Creating a new repository on Github

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *

 techloset ▾

Repository name *

/

Great repository names are short and memorable. Need inspiration? How about [super-duper-octo-giggle?](#)

Description (optional)

☐  **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:

Skip this step if you're importing an existing repository.

☐ **Add a README file**

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None ▾

Choose a license

Git Commands

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#) [Settings](#)

Quick setup — if you've done this kind of thing before

Set up in Desktop

 or

HTTPS

SSH

https://github.com/Aroma-Tariq/Affiliate.git

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

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

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

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

```
git remote add origin https://github.com/Aroma-Tariq/Affiliate.git
git branch -M main
git push -u origin main
```



Login

- The command you need to use is '**git config --global user.name "your_username"** to set your username and **git config --global user.email "your_email"** to set your email.



Logout

- `git config --global --unset "your email"`



Git Init

- **git init** is a Git command that is used to initialize a new Git repository on your local computer.
- The command creates a new directory named `.git` in the current working directory, which is used to store the metadata and history of the repository.



Git add

- **git add** is a Git command that is used to stage changes made to the files in a local repository for a commit.
- The command is used to add changes made to the files in the local repository to the "staging area" which is a holding area for changes that are ready to be committed.



Git Commit

- **git commit -m "first commit"** is a Git command that is used to save changes made to the files in a local Git repository.
- The -m option is used to provide a commit message, which is a brief description of the changes that were made in this commit.



Git branches

In Git, a branch is a pointer to a specific commit in the repository's commit history.

By default, when you first create a repository, Git creates a single branch called "master". Branches are used to create different versions of your code, allowing you to work on multiple features or fix bugs without interfering with the main codebase.



Here are a few key points about branches in Git:

- Branches are lightweight and easy to create and switch between.
- Each branch has its own commit history, allowing you to make changes independently of other branches.
- You can merge changes from one branch into another, allowing you to combine multiple branches into a single branch.
- Branches can be used to work on multiple features, fix bugs, and experiment without affecting the main codebase.
- By default, the first branch that is created in a repository is named "master".



Git Branch Command

- **git branch -M main** is a Git command that is used to rename a branch in a local repository. The **-M** option is used to specify that the branch should be renamed, and **main** is the new name of the branch.



Git Remote Origin


In Git, a remote is a named connection to another repository.

By default, when you clone a repository using `git clone`, Git automatically creates a remote named "origin" that points to the URL of the repository you cloned.

The `git remote` command is used to manage and interact with remotes.

Git Clone

'git clone' is a Git command that is used to create a local copy of a remote repository on your computer.



The command creates a new directory on your computer and copies all of the files and commit history from the remote repository to the local copy.



Git push

- **git push** is a Git command that is used to send commits from a local repository to a remote repository.
- The command is used to update the remote repository with the commits that were made in the local repository.



Git pull

git pull is a Git command that is used to retrieve changes from a remote repository and merge them into a local repository.

The command is used to synchronize the local repository with the remote repository by fetching the changes from the remote and merging them into the local repository.