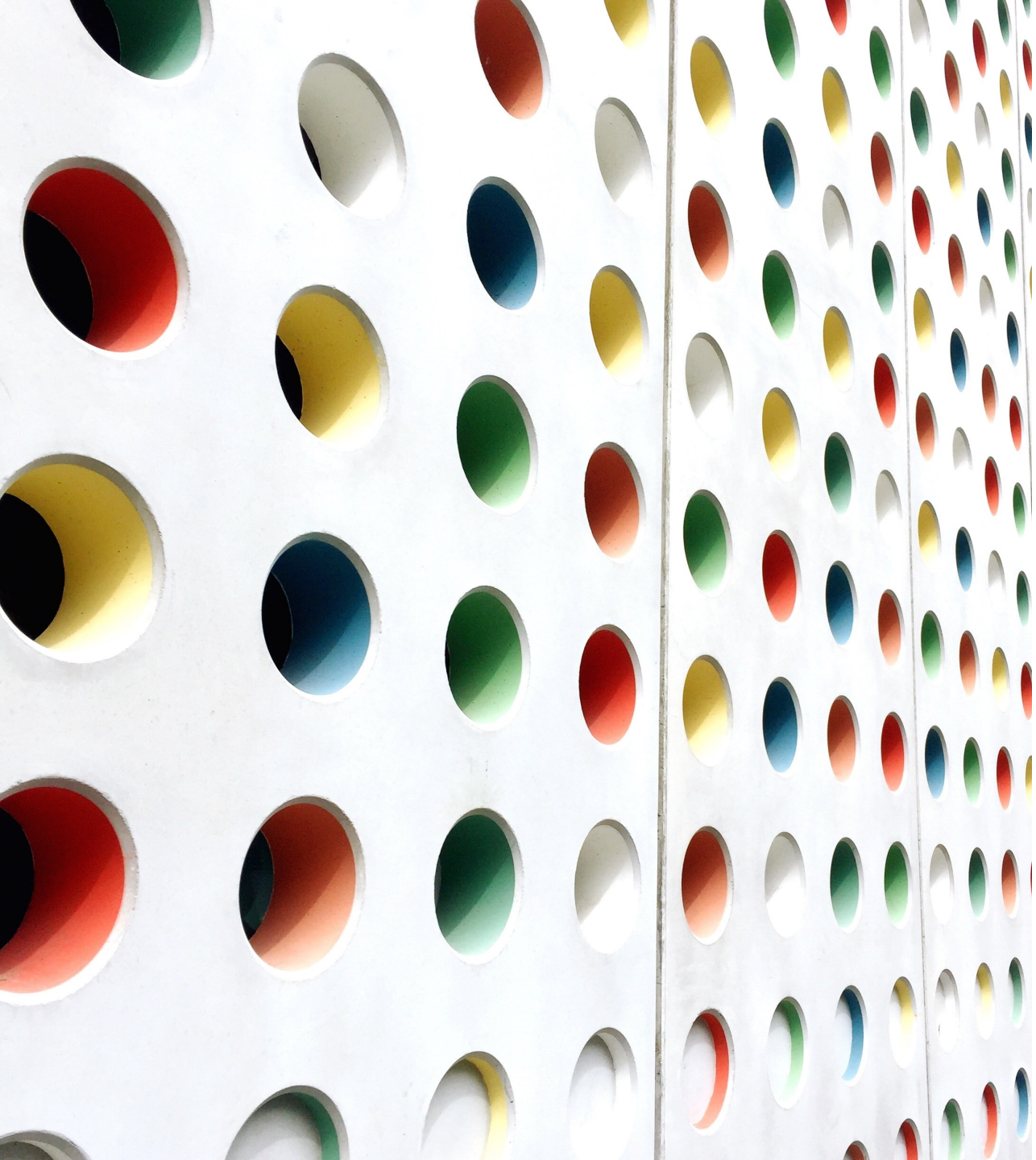


Git & Github

What will be the outcome after this workshop?

what is git?

Git is a version control system that allows you to track changes to your files and collaborate with others. It is used to manage the history of your code and to merge changes from different branches.


#CHICAGOFIRE
NBC

Did you get it right?

Don’t worry if that sounded a bit jargony! Let’s break it down with some familiar terms.

What is git?

- **Version Control = Saving Your Progress**
  Think of it like saving your game in a video game. You can always go back to a previous point if you make a mistake!
- **Collaborating = Group Projects**
  Remember working on group assignments? Git helps everyone contribute without stepping on each other's toes, ensuring everyone's input is tracked!
- **History = Your Project Timeline**
  Imagine your project is like a timeline of your school year, with important events noted. Git keeps a detailed history of your project's development, so you can look back anytime!
- **Merging = Combining Ideas**
  It's like combining different recipes into one dish. Git allows you to merge your ideas with those of your teammates seamlessly!

Git and Github are different !

- Git is a version control system that is used to track changes to your files.
- It is a free and open-source software that is available for Windows, macOS, and Linux. Remember, GIT is a software and can be installed on your computer.
- Github is a web-based hosting service for Git repositories.
- Github is an online platform that allows you to store and share your code with others.
- It is a popular platform for developers to collaborate on projects and to share code.
- It is not that Github is the only provider of Git repositories, but it is one of the most popular ones.

Install Git

- To install Git, you can use command line or you can visit official website and download the installer for your operating system. Git is available for Windows, macOS, and Linux and is available at https://git-scm.com/downloads.
- Also just go on the github.com and create one account there.

Terminology

- Git and people who use it talk in a different terminology. For example they don't call it a folder, they call it a repository. They don't call it alternative timeline, they call it branch.

Check your git version

- To check your git version, you can run the following command:
  git —version
- This command will display the version of git installed on your system

Repository

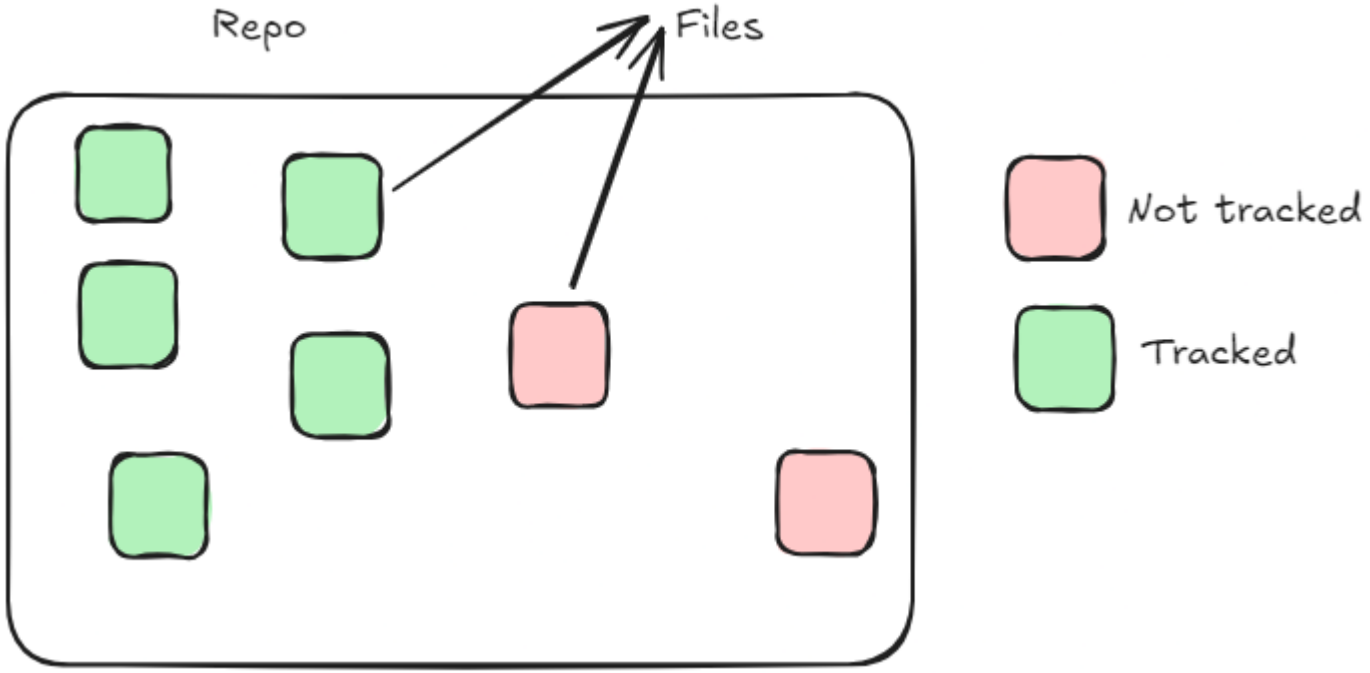
- A repository is a collection of files and directories that are stored together.
- It is a way to store and manage your code.
- A repository is like a folder on your computer, but it is more than just a folder.
- It can contain other files, folders, and even other repositories.
- You can think of a repository as a container that holds all your code.

Repository

- There is a difference between a software on your system vs tracking a particular folder on your system.
- At any point you can run the following command to see the current state of your repository:

```
git status
```

Repository



- Not all folders are meant to be tracked by git.
- Here we can see that all green folders are projects are getting tracked by git but red ones are not.

Your config settings

- Github has a lot of settings that you can change.
- You can change your username, email, and other settings.
- Whenever you checkpoint your changes, git will add some information about your such as your username and email to the commit.
- There is a git config file that stores all the settings that you have changed.

Your config settings

- Let's setup your email and username in this config file. I would recommend you to create an account on github and then use the email and username that you have created.
  git config --global user.email "your-email@example.com".
  git config --global user.name "Your Name"
- Now you can check your config settings:
  git config --list

Creating a repository

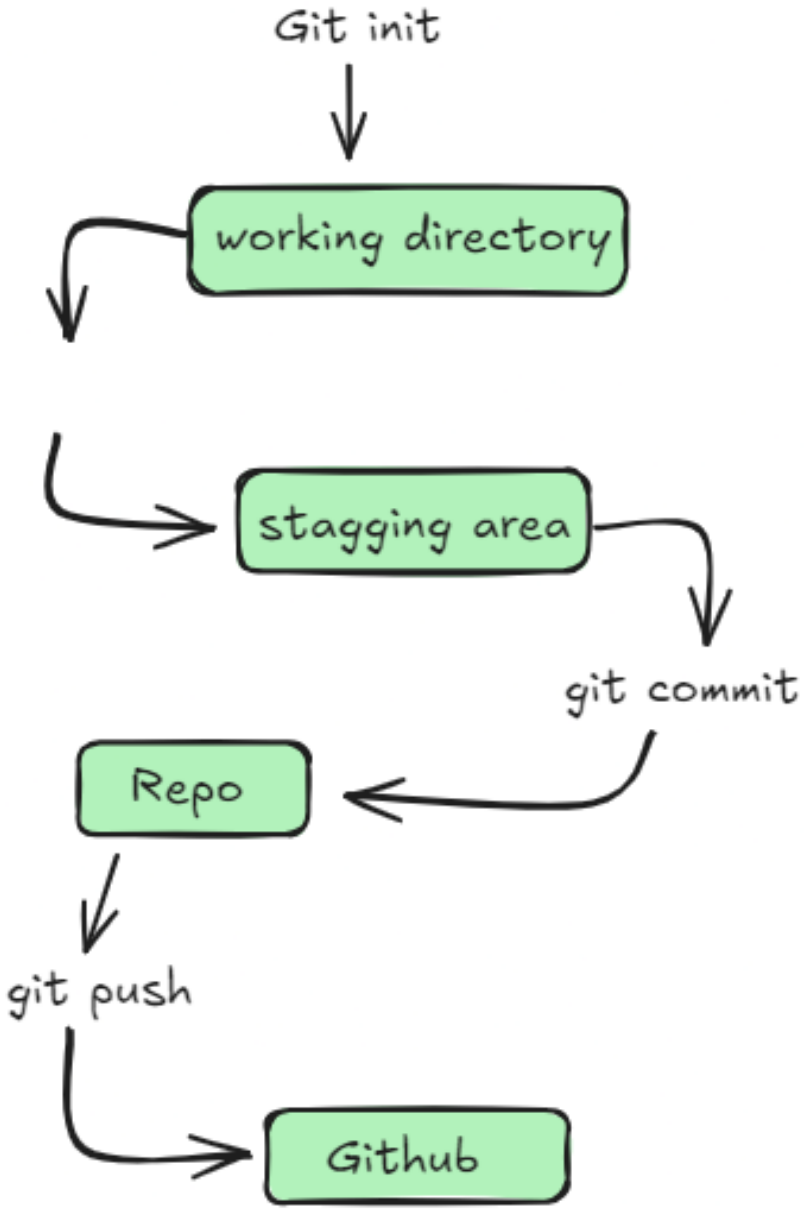
- Creating a repository is a process of creating a new folder on your system and initializing it as a git repository.
- It's just regular folder to code your project, you are just asking git to track it. To create a repository, you can use the following command:

  git init

- This command will create a new folder on your system and initialize it as a git repository. This adds a hidden .git folder to your project.

Commit

- Commit is a way to save your changes to your repository.
- It is a way to record your changes and make them permanent. You can think of a commit as a snapshot of your code at a particular point in time.
- When you commit your changes, you are telling git to save them in a permanent way. This way, you can always go back to that point in time and see what you changed.
- Usual flow looks like this:



Complete Flow of Git

Stage

- Stage is a way to tell git to track a particular file or folder. You can use the following command to stage a file:

  git init

  git add <file> <file2>

  git status

- Here we are initializing the repository and adding a file to the repository.
- Then we can see that the file is now being tracked by git.
- Currently our files are in staging area, this means that we have not yet committed the changes but are ready to be committed.

Commit

git commit -m "commit message"

git status

- Here we are committing the changes to the repository. We can see that the changes are now committed to the repository.
- The –m flag is used to add a message to the commit. This message is a short description of the changes that were made.
- You can use this message to remember what the changes were.

Logs

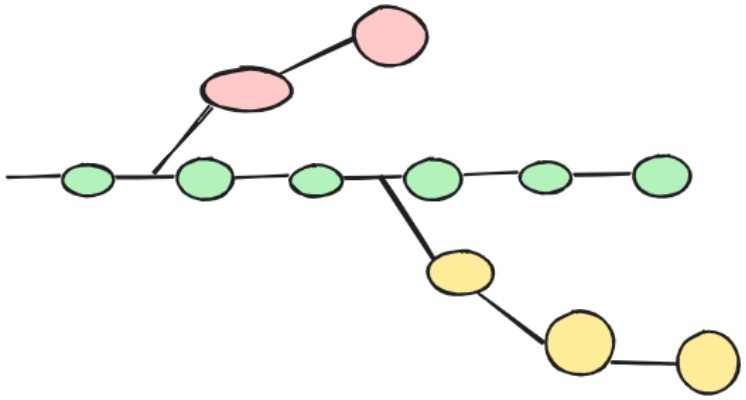
git log

- This command will show you the history of your repository.
- It will show you all the commits that were made to the repository.
- You can use the --online flag to show only the commit message.
- This will make the output more compact and easier to read.

gitignore

- Gitignore is a file that tells git which files and folders to ignore.
- It is a way to prevent git from tracking certain files or folders.
- You can create a gitignore file and add list of files and folders to ignore.

Branches in git



- Branches are a way to work on different versions of a project at the same time.
- They allow you to create a separate line of development that can be worked on independently of the main branch.
- This can be useful when you want to make changes to a project without affecting the main branch or when you want to work on a new feature or bug fix.
- Some developers can work on Header, some can work on Footer, some can work on Content, and some can work on Layout.
- This is a good example of how branches can be used in git.

HEAD in git

- The HEAD is a pointer to the current branch that you are working on.
- It points to the latest commit in the current branch. When you create a new branch, it is automatically set as the HEAD of that branch.
- the default branch used to be master, but it is now called main. There is nothing special about main, it is just a convention.

Creating a new branch

To create a new branch, you can use the following command:

```
git branch
```

```
git branch bug-fix
```

```
git switch bug-fix
```

```
git log
```

```
git switch master
```

```
git switch -c dark-mode
```

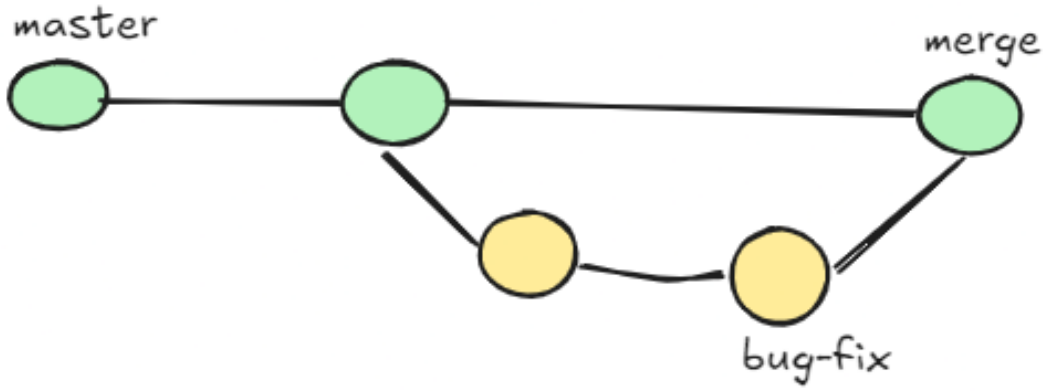
```
git checkout orange-mode
```

Create a new Branch

Some points to note:

- **git branch - This command lists all the branches in the current repository.**
- **git branch bug-fix - This command creates a new branch called bug-fix.**
- **git switch bug-fix - This command switches to the bug-fix branch.**
- **git log - This command shows the commit history for the current branch.**
- **git switch master - This command switches to the master branch.**
- **git switch -c dark-mode - This command creates a new branch called dark-mode. the -c flag is used to create a new branch.**
- **git checkout orange-mode - This command switches to the orange-mode branch.**
- **Commit before switching to a branch**
- **Go to .git folder and checkout to the HEAD file**

Merging branches



- When you are done working on a branch, you can merge it back into the main branch.
- This is done using the following command:

git checkout main

git merge bug-fix

Renaming and deleting a Branch

git branch -m <old-branch-name> <new-branch-name>

git branch -d <branch-name>

Lets Play with Pull Request now.

Fork and Clone the Repository

1. **Fork** the repository on GitHub
2. **Clone** your forked repository**:**

   git clone [your_forked_repo_url]

3. Create a New Branch
   1. Navigate to the project directory:

      cd [project-directory]

Lets Play with Pull Request now.

2. Create and switch to a new branch

    git checkout -b feature/your-branch-name

4. **Make Changes, Stage, and Commit**

    Make changes to the file(s) as needed.

        git add [filename]

        git commit –m “hi this is my commit”

5. Push the branch : git push origin feature/your-branch-name

Lets Play with Pull Request now.

- Add a title and description for your PR, then click **Create Pull Request**.
- Go to the original repository on GitHub.
- Navigate to the **Pull Requests** tab and click **New Pull Request**.
- Select your branch and the base branch for comparison.


ALL
DONE!

Thank you