# Version Control with Github

Git commands

# Introduction

Version control is a basic skill that is important to know when it comes to software development. This presentation is about the hands-on practice of Git version control, specifically the command line interface.

# Objectives of the Exercise

- Understand Git version control concepts
- Set up and manage repositories using the command line
- Practice core Git workflows
- Demonstrate collaboration using branches and pull requests
- Present evidence of learning outcomes

# Tools & Environment Used

- Git (Command Line Interface)
- GitHub
- Local development environment
- Text editor (VS Code)

# Configuration of Git

- There are several ways to install git on your machine which depends on your Operating System.
- Take note of the following commands:

  **git config - -global user.name “Your name here” :** This command is used to configure and set up your name on Git bash.

  **git config - -global user.name :** This command is used to check whether your username has been added

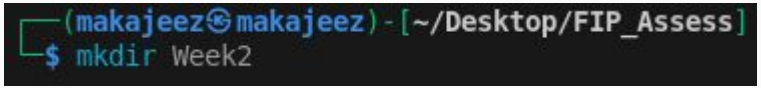
# Configuration of Git

**git config - -global user.email “Your email here” :** This command is used to configure and setup your email on Git bash.

**git config - -global user.email :** This command is used to check whether the email has been added


# Repository Setup (CLI)

- Create a repository in the CLI using the code **mkdir week2**
- Move into the created directory **cd week2**





- Initiate a local repository using **git init**
- Added remote repository using **git remote add origin**
- Initialized main branch and pushed to remote
- Cloned repository to a second directory for practice

# Repository Setup (CLI)

```
┌──(makajeez㉿makajeez)-[~/Desktop]
└─$ git clone https://github.com/makajeez/FIP_Assess
Cloning into 'FIP_Assess' ...
remote: Enumerating objects: 28, done.
remote: Counting objects: 100% (28/28), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 28 (delta 5), reused 21 (delta 4), pack-reuse
d 0 (from 0)
Receiving objects: 100% (28/28), 5.51 KiB | 512.00 KiB/s, d
one.
Resolving deltas: 100% (5/5), done.
```

- **git init** was not used because git clone “URL” was used in the above screenshot, and hence remote is also connected

```
┌──(makajeez㉿makajeez)-[~/Desktop/FIP_Assess]
└─$ git remote -v
origin  https://github.com/makajeez/FIP_Assess (fetch)
origin  https://github.com/makajeez/FIP_Assess (push)
```

```
┌──(makajeez㉿makajeez)-[~/Desktop/FIP_Assess]
└─$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Week2/

nothing added to commit but untracked files present (use "git add" to track)
```

- This show the status of the Local repo, and also that Git sees your new folder and its content

# Branching & Commit Management

- Created feature branches using `git checkout -b`
- Made file changes and staged them with `git add`
- Committed changes using meaningful commit messages
- Renamed branches to reflect purpos

```
┌──(makajeez㉿makajeez)-[~/Desktop/FIP_Assess]
└─$ git checkout -b  week2
```

```
┌──(makajeez㉿makajeez)-[~/Desktop/FIP_Assess]
└─$ git add Week2/

┌──(makajeez㉿makajeez)-[~/Desktop/FIP_Assess]
└─$ git commit -m 'Commit: Readme.md'
[main 9ca3a0a] Commit: Readme.md
 1 file changed, 1 insertion(+)
 create mode 100644 Week2/README.md
```

# Reverting & History Control

- Reverted commits using `git revert`
- Preserved commit history integrity
- Demonstrated safe rollback practices

# Pulling, Pushing & Synchronization

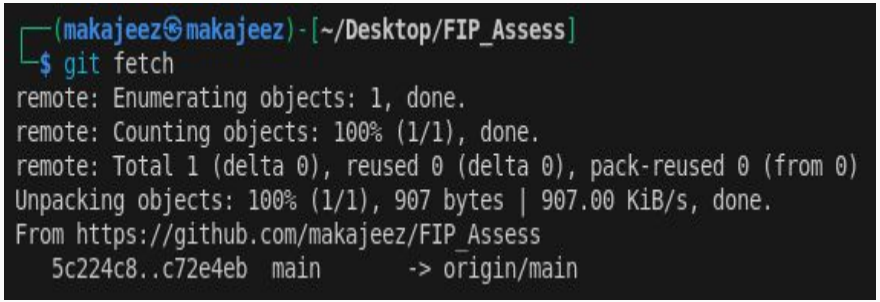
- Pushed changes upstream to remote repository
- Pulled updates downstream from remote
- Practiced working with multiple local copies
- Ensured repository synchronization

```
┌──(makajeez㉿makajeez)-[~/Desktop/FIP_Assess]
└─$ git push -u origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 576 bytes | 576.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/makajeez/FIP_Assess
   8c5df06..5c224c8  main -> main
branch 'main' set up to track 'origin/main'.
```

```
┌──(makajeez㉿makajeez)-[~/Desktop/FIP_Assess]
└─$ git pull
Already up to date.
```

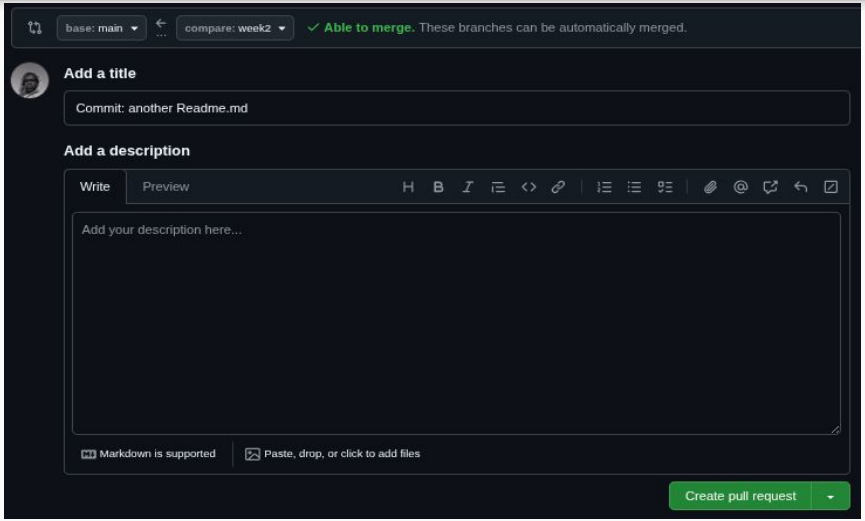
# Fetching & Merging Changes

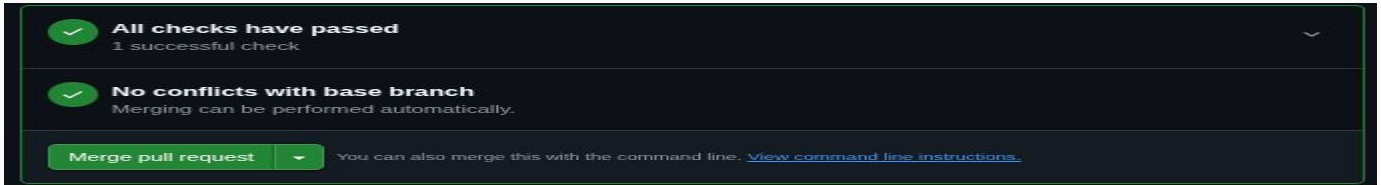
- Used `git fetch` to retrieve remote updates
- Merged branches into main branch
- Resolved merge scenarios



# Pull Request Workflow

- Created pull requests from feature branches
- Added descriptions and reviewed changes
- Merged pull requests into main branch
- Reverted merged pull requests when required



# Conclusion

This hands-on exercise strengthened practical Git version control skills and demonstrated industry-aligned workflows essential for collaborative development and cybersecurity projects.

# References

- Atlassian Git Glossary
- LoginRadius Git Commands Guide