1 GitHub Workshop Guide: Basic Git Commands and Workflow

Welcome to the practical part of our workshop, where we'll dive into the Basic Git Commands and Workflow. This guide is designed to be followed step-by-step as you learn the core operations of Git, the widely used version control system. By the end of this session, you'll have hands-on experience with creating a repository, making changes, and syncing those changes with a remote repository on GitHub.

1.1 Objective

You will learn how to:

- Create a new repository on GitHub.
- Clone the repository to your local machine.
- Make changes, stage, and commit them.
- Push changes back to GitHub.

1.2 Introduction to Basic Git Commands

Before we begin, let's understand why we use Git. Git helps us track changes in our projects, allowing multiple collaborators to work on the same project without overlapping each other's work. It's an essential tool for version control and collaboration.

1.3 Creating a New Repository on GitHub

Create the Repository:

- Visit [GitHub](https://github.com) and sign in to your account.
- Click on the "New repository" button.
- Name your repository (e.g., “workshop-demo”) and provide a brief description.
- Select "Public" and check "Initialize this repository with a README".
- Click "Create repository".

1.4 Install Git

- Go to https://git-scm.com/ and find “Download for Windows” or “Download for Mac” based on your operating system.
- After downloading, install “Git” following the instructions in the installation wizard.

1.5 Cloning the Repository

Clone the Repository:

- On your repository's page on GitHub, click the "Code" button and copy the HTTPS URL.
- Open your terminal or command prompt.
- Type “git clone [URL]”, replacing “[URL]” with the URL you copied, then press Enter.
- Navigate into the cloned repository with “cd workshop-demo”.

1.6 Making Changes Locally

Create a New File:

- Inside the “workshop-demo” directory, create a new file named “participants.txt”.
- Open “participants.txt” in a text editor, add your name, and save the file.

Check the status of files

- Return to your terminal.
- Type “git status” to see the status files.

Stage the Changes:

- Return to your terminal.
- Type “git add participants.txt” to stage your new file.

Commit the Changes:

- Commit your staged changes by typing “git commit -m "Add participants list"“.

Pushing Changes to GitHub

Push Your Changes:

- In the terminal, type “git branch” to find the branches in the repository.
- Type “git push origin main” to push to “main” branch
- This command sends your changes to GitHub.

Verify on GitHub:

- Refresh your GitHub repository page to see the new “participants.txt” file.

2 GitHub Hands-on Exercise Guide: First Commit and Collaboration

This exercise is designed to give you practical experience with making your first commit and collaborating on a shared project using GitHub. You'll learn how to fork a repository, make changes, and then submit those changes through a pull request. Let’s get started!

2.1 Objective

By the end of this exercise, you will have:

- Forked a repository to your GitHub account.
- Cloned the forked repository to your local machine.
- Made changes to a file and committed those changes.
- Pushed your changes to GitHub.
- Created a pull request to merge your changes into the original repository.

2.2 Fork the Repository

- Fork the Repository: Navigate to the URL: https://github.com/CDAC-lab/BUS5001-workshop-demo.git . Look for the "Fork" button at the top right corner of the page and click it. This action creates a copy of the repository in your GitHub account, allowing you to make changes independently.

Forking a repository allows you to freely experiment with changes without affecting the original

project.

2.3 Clone and Modify the Repository

Clone Your Fork: Once the fork is complete, you'll be taken to your copy of the repository. Click the green "Code" button and copy the URL. Open your terminal or Git Bash, navigate to where you want the repository to be on your local machine, and type:

git clone [URL]

Replace “[URL]” with the URL you copied. This command downloads your forked repository to your computer.

Navigate to the Repository: Change into the directory of the cloned repository by typing:

cd [repository-name]

Modify the “participants.txt” File: Open the “participants.txt” file in your favourite text editor, add your name or a unique identifier to the file, and save your changes.

2.4 Commit and Push Changes

Stage Your Changes: To prepare your changes for a commit, use the command:

git add participants.txt

This command adds your changes to the staging area.

Commit Your Changes: Now, commit your staged changes with a meaningful message by typing:

git commit -m "Add my name to participants list"

Commit messages help you and others understand what changes were made at a glance.

Push Your Changes: Upload your committed changes to your GitHub fork with the command:

git push origin master

Or if your default branch is named “main”, use:

git push origin main

This updates your fork on GitHub with your latest changes.

2.5 Creating a Pull Request (PR)

Navigate to Your Fork on GitHub: Open your web browser and go to your fork of the repository on GitHub.

Initiate the Pull Request: Click the "Pull Request" button near the top of the page. GitHub will automatically direct you to the original repository and highlight the changes you've made.

Create the Pull Request: Click the "Contibute" button and then “Open Pull Request”. Add a title that summarizes your contribution and a brief description in the provided fields. Once complete, submit your pull request by clicking the final "Create Pull Request" button.

Purpose of Pull Requests: PRs allow project maintainers to review your contributions before merging

them into the main project. They are a fundamental tool for collaboration and code review in the world of open-source software.

You've successfully navigated the process of making a contribution to a project on GitHub. You forked a repository, made changes, committed those changes, pushed them to GitHub, and opened a pull request. These steps are the backbone of collaboration in many software projects.

3 GitHub Hands-on Syncing Your Fork with the Original Repository Using GitHub's Web UI

Welcome to this workshop segment on how to keep your forked repository up to date with the original repository using GitHub's Web UI. This essential skill ensures that your project stays current with the latest changes, making collaboration and contribution more straightforward. This guide will take you step-by-step through the process as outlined in GitHub's documentation.

3.1 Objective

By the end of this workshop, you will be able to:

- Understand the importance of keeping your fork synced with the upstream repository.
- Use GitHub's Web UI to fetch and merge changes from the original repository into your fork.

3.2 Understanding Fork Syncing

A fork is a copy of a repository that you manage. It allows you to freely experiment with changes without affecting the original project.

Over time, the original repository (upstream) may be updated with new changes or improvements. Syncing ensures your fork stays updated, which is crucial for contributing or keeping your project aligned with the upstream developments.

3.3 Configuring the Upstream Repository

Before syncing, ensure the upstream repository is configured correctly in your forked project. GitHub's new Web UI features simplify this process, allowing it to be done entirely through the website without command-line tools.

Navigate to Your Fork on GitHub: Open your web browser and go to your forked repository on GitHub.

Checking for Upstream Repository: In the "Code" tab of your fork, look for the "Sync Fork" option. If you see it, your fork is already aware of the upstream repository. If not, you'll need to set it manually by clicking on "Settings" > "Branches" > "Add more" under "Branch protection rules", and then configure the upstream repository details as required.

4 GitHub Hands-on Syncing Your Fork via the Web UI

4.1 Fetch Upstream Changes:

Go to the "Code" tab of your forked repository.

Click on the "Sync Fork" dropdown. Here, GitHub displays the current status of your fork compared to the upstream repository.

Click "Update Branch" to merge the latest upstream changes into your main branch. This step won't affect your local changes unless there are conflicts between your fork and the upstream changes.

4.2 Reviewing the Changes:

After clicking "Update Branch", GitHub will show you a summary of the merge, including any conflicts that need resolution.

If there are no conflicts, your fork will be automatically updated to match the upstream repository.

In case of conflicts, GitHub will guide you through resolving them online.

4.3 Keeping Your Fork Updated

Regular Maintenance:

It's good practice to fetch and merge changes from the upstream repository regularly, especially before starting new work or making contributions.

Resolving Conflicts:

If you encounter merge conflicts during the update, GitHub's Web UI provides tools to resolve these directly online. Resolve conflicts by choosing which changes to keep, then commit the merge. Detailed instructions and support are available if you run into complex conflict scenarios.

You've successfully learned how to keep your forked repository up-to-date with the original repository using GitHub's Web UI! This skill is vital for maintaining a healthy workflow when working on projects that are collaborative or continuously evolving.

Remember, regularly syncing your fork can save you a lot of time and hassle, especially when preparing to make contributions. Stay proactive about merging changes, and you'll find your collaboration efforts going much more smoothly.

5 GitHub Hands-on Creating Your CV on GitHub

Welcome to our workshop on creating and hosting your CV (Curriculum Vitae) on GitHub! This hands-on session will guide you through creating a repository for your CV, adding your CV content, and showcasing it directly on your GitHub profile. This is an excellent way for developers, designers, and technologists to share their professional journey in a way that's accessible to collaborators, recruiters, and peers.

5.1 Objective

- Have a GitHub repository dedicated to your CV.
- Understand how to create a README file that effectively showcases your CV.
- Know how to leverage GitHub features to enhance your professional visibility.

5.2 Setting Up Your GitHub Repository

5.2.1 Create a New Repository

Log into your GitHub account.

Click on the "+" icon in the upper-right corner and select "New repository".

Name your repository something relevant, like “your-username-CV” (replace “your-username” with your actual GitHub username).

Choose "Public" so that your CV can be viewed by anyone.

Initialize the repository with a README file. This file will serve as the primary document for your CV.

5.2.2 Understanding the README

The README file is displayed by default when someone visits your repository. It’s the perfect place to present your CV content in a clear, markdown-formatted document.

5.3 Adding Your CV to the README.md File

5.3.1 Edit the README.md File

In your new repository, click on the README.md file.

Click the pencil icon (Edit this file) to start editing.

You're now ready to add your CV content.

5.3.2 Formatting Your CV with Markdown

Markdown is a lightweight markup language that allows you to add formatting elements using plain text. Here's a brief overview of how to structure your CV:

Headers: Use “#” for main headers (e.g., “# Student Name”) and “##” for subsections (e.g., “ Education”).

Bullet Points: Use “-” or “*” for bullet points (e.g., “- Bachelor's Degree in Computer Science”).

Links: If you want to link to your projects or social profiles, use “[text](URL)” format (e.g., “[My Portfolio](http://example.com)”).

Italic and Bold: Use “*” for italic and “**” for bold (e.g., “*italic*”, “**bold**”).

5.3.3 Inserting Your CV Content

Start with a brief introduction about yourself.

List your professional experience, starting with the most recent position.

Include your education background.

Highlight any projects, contributions to open source, publications, or awards.

Consider adding a section for skills and another for hobbies or interests if relevant.

5.3.4 Preview and Save

Use the "Preview" tab to review your formatting and make sure everything looks as expected.

Once satisfied, scroll down, add a commit message like "Initial CV upload", and click "Commit changes".

5.4 Making Your CV Visible on Your GitHub Profile

Create a Special Repository for Your GitHub Profile:

GitHub allows you to create a special repository that is used to add a README to your GitHub profile.

The repository must be named exactly like your username. For example, if your username is “username001”, your repository name should be “username001”.

Initialize this repository with a README file.

Link to Your CV from Your Profile README:

Edit the README.md file in this special repository.

Add a section that introduces yourself and then links to your CV repository. For example:

Hi there, I'm Student Name!

I'm a software developer with a passion for open-source projects. Check out my [CV](https://github.com/johndoe/johndoe-CV) for more information about my professional background.

Commit your changes. This README will automatically appear on your GitHub profile, directing visitors to your CV.