Introduction to Git and GitHub

- **What is Git?**
  Git is a version control system that lets you track changes, collaborate with others, and manage the development history of your projects.
- **What is GitHub?**
  GitHub is a cloud-based platform that hosts Git repositories, allowing you to share code, collaborate with others, and showcase your projects.

---

2. Initial Git Setup

1. **Install Git**
   - **Verify Installation**

     ```
     git --version
     ```

     Explanation: Checks if Git is already installed on the system.
2. **Set Username and Email**
   - **Commands**:

     ```
     git config --global user.name "Your Name"
     git config --global user.email "your.email@example.com"
     ```

     Explanation: Configures Git to recognize the user's name and email on each commit, essential for tracking who made changes in collaborative projects.

---

3. Creating a New Repository Locally

1. **Create a New Folder for the Project**
   - **Command**:

     ```
     mkdir my-first-repo
     cd my-first-repo
     ```

Explanation: Creates and navigates into a new folder where the repository will be initialized.

2. **Initialize a New Repository**
   - **Command**:

     ```
     git init
     ```

     Explanation: Initializes a new, empty Git repository, marking the current folder as a tracked project.
3. **Create a README File**
   - **Command**:

     ```
     echo "# My First Repository" > README.md
     ```

     Explanation: Creates a `README.md` file that introduces the repository; a README file is often the first thing viewers see.

---

4. Staging and Committing Changes

1. **Stage Files for Commit**
   - **Command**:

     ```
     git add README.md
     ```

     Explanation: Stages the `README.md` file to be included in the next commit, preparing it to be saved in the repository's history.
2. **Commit Changes**
   - **Command**:

     ```
     git commit -m "Initial commit with README"
     ```

     Explanation: Creates a commit (a snapshot of changes) with a message, logging the state of the staged files.

---

5. Connecting to GitHub and Pushing Changes

1. **Create a New Repository on GitHub**
   Instructions: Go to GitHub, create a new repository, and **don't initialize it** with files

like README or .gitignore.

2. **Link Local Repository to GitHub (Remote)**
   - **Command**:

     ```
     git remote add origin <URL>
     ```

     Explanation: Links the local repository to GitHub, where `<URL>` is the GitHub repository URL.
3. **Push Changes to GitHub**
   - **Command**:

     ```
     git push -u origin main
     ```

     Explanation: Pushes local commits to GitHub, making the project accessible online. The `-u` sets `origin` and `main` as the default push destination.

---

6. Making and Committing Additional Changes

1. **Edit the README File**
   - **Command**:

     ```
     echo "This is my first GitHub project!" >> README.md
     ```

     Explanation: Appends text to the README file.
2. **Check Status of Repository**
   - **Command**:

     ```
     git status
     ```

     Explanation: Displays the current state of the working directory, showing modified, staged, and untracked files.
3. **Stage and Commit Changes**
   - **Commands**:

     ```
     git add README.md
     git commit -m "Update README with project description"
     ```

     Explanation: Stages and commits updates, recording the change with a message.
4. **Push Changes to GitHub**

- **Command**:

  ```
  git push
  ```

  Explanation: Uploads the new commit to GitHub, updating the project’s main branch.

---

7. Viewing History and Undoing Changes

1. **View Commit History**
   - **Command**:

     ```
     git log
     ```

     Explanation: Displays all commits with details like commit ID, author, date, and message.
2. **Undo Changes (Optional)**
   - **Commands**:

     ```
     git restore README.md  # Undo changes in the working directory
     git checkout main       # Reset to the latest commit in main
     branch
     ```

     Explanation: Demonstrates how to revert changes if mistakes are made, with `restore` for uncommitted changes and `checkout` for a clean slate.

---

8. Cloning a Repository

1. **Clone a Repository**
   - **Command**:

     ```
     git clone <URL>
     ```

     Explanation: Clones a remote GitHub repository to their local system, creating a complete copy.

Git Cheat Sheet by Git-SCM

- **Link**: Git-SCM Git Cheat Sheet
- **Details**: Provided by the official Git documentation, this tutorial covers the basics and dives into more advanced commands. Great for beginners and those wanting to explore Git capabilities.

Here’s a collection of additional cheat sheets along with some highly-rated video resources that cover the essentials of Git and GitHub for beginners.

More Cheat Sheets

1. **Git Cheat Sheet by Git-SCM**
   - **Link**: Git-SCM Git Cheat Sheet
   - **Details**: Provided by the official Git documentation, this tutorial covers the basics and dives into more advanced commands. Great for beginners and those wanting to explore Git capabilities.
2. **Git Cheat Sheet by FreeCodeCamp**
   - **Link**: FreeCodeCamp Git Cheat Sheet
   - **Details**: This is a thorough cheat sheet with explanations, covering Git basics, branching, and collaboration. It’s a beginner-friendly resource that’s easy to reference and understand.
3. **Git Command Cheatsheet by DevHints**
   - **Link**: DevHints Git Cheatsheet
   - **Details**: Organized by sections (branching, tagging, pushing, etc.), this cheatsheet has a clean layout for quick reference. Good for those who prefer a web-based guide with structured commands.
4. **Interactive Git Cheat Sheet by Git Tower**
   - **Link**: Git Tower Git Cheat Sheet
   - **Details**: An interactive, well-organized cheat sheet by Git Tower, covering basic and advanced Git commands. It also offers PDF download options.
5. **Comprehensive Git and GitHub Cheatsheet by DigitalOcean**
   - **Link**: DigitalOcean Git and GitHub Cheat Sheet
   - **Details**: A comprehensive cheat sheet, covering everything from Git configuration to rebasing, tagging, and GitHub collaboration.

Video Tutorials and Playlists

1. **Git and GitHub for Beginners by Programming with Mosh**
   - **Link**: Git and GitHub for Beginners - Crash Course

- **Duration**: 1.5 hours
- **Content**: A full crash course that introduces Git and GitHub basics, ideal for beginners who need to start from zero and learn about branches, commits, pushing, and pulling.

2. **Git and GitHub Full Course by FreeCodeCamp**
   - **Link**: FreeCodeCamp Git and GitHub Full Course
   - **Duration**: 2 hours
   - **Content**: This course dives into Git basics, branching, GitHub workflows, and real-world scenarios. It’s a good hands-on approach for understanding Git commands and GitHub collaboration.
3. **Introduction to GitHub by GitHub Learning Lab**
   - **Link**: Introduction to GitHub
   - **Duration**: 10 minutes
   - **Content**: An official GitHub tutorial that covers the essentials of repositories, commits, and pushes. Ideal for getting a quick overview before diving into practical work.
4. **Version Control with Git by Google Developers**
   - **Link**: Google Developers Git Series
   - **Duration**: Series of short videos
   - **Content**: This playlist breaks down Git concepts into manageable chunks. Covers topics from Git installation to working with branches and managing conflicts, making it great for first-year students.
5. **GitHub Pages Guide by Traversy Media**
   - **Link**: Host a Website with GitHub Pages
   - **Duration**: 20 minutes
   - **Content**: Explains how to use GitHub Pages for hosting websites. Good for a project that includes publishing a portfolio site or static webpage.
