👫 Buddy Up for Git & GitHub Challenge! 👫

Let's get started! This challenge will help you get comfortable using Git and GitHub with a buddy. You'll learn how to:

- Create and fork repositories
- Manage branches
- Make pull requests
- Merge code changes

Before you start:

1. **Find a partner!** You'll both need GitHub accounts.
2. **Decide who will be Partner A and Partner B.** Partner A will create the original repository.

Part 1: Setting Up the Project (Partner A)

Partner A, it's your turn first!

1. **Create a new repository:**
    - Go to your GitHub homepage and click the green "New" button.
    - Name your repository something fun, like "git-challenge" or "awesome-project".
    - You can keep it public.
    - **(Add Image: Screenshot of creating a new repo on GitHub)**
    - Click "Create repository"!
2. **Clone the repository to your computer:**
    - On your repository page, click the green "Code" button and copy the HTTPS URL.
    - Open your terminal/command prompt.
    - Type `git clone <paste your repository URL here>` and press Enter.
    - **(Add Image: Screenshot of copying the repository URL and using git clone in the terminal)**
3. **Create a `.gitignore` file (Optional but Recommended):**
    - In your terminal, navigate to your new repository's directory: `cd <your repository name>`
    - Create a `.gitignore` file: `touch .gitignore`
    - Open this file in a text editor and add patterns to exclude files you don't want to track (e.g., `*.log`, `node_modules/`). This keeps your repository clean and focused.
    - **(Add Image: Example of a simple .gitignore file)**
4. **Create a new branch for your feature:**
    - In your terminal: `git checkout -b featureA`
    - **Tip:** Use descriptive branch names for real-world projects (e.g., "add-login-feature", "fix-homepage-layout")
    - **(Add Image: Screenshot of terminal showing navigating to the directory and creating a new branch)**
5. **Now, share your repository link with Partner B and relax! They'll take it from here.**

Part 2: Forking and Branching (Partner B)

Alright Partner B, time to shine!

1. **Fork the repository:**
    - Go to your Partner A's repository page on GitHub (they shared the link, right?).
    - Click the "Fork" button at the top right corner.
    - **(Add Image: Screenshot of forking a repository on GitHub)**
    - You now have a copy of the repository under your GitHub account!
2. **Clone your forked repository:**
    - Just like Partner A did, copy the HTTPS URL of **your forked repository**.
    - Open your terminal/command prompt.
    - Type `git clone <paste your forked repository URL here>` and press Enter.
    - **(Add Image: Screenshot of copying the forked repository URL and cloning it in the terminal)**
3. **Create a new branch for your feature:**
    - Navigate to your forked repository's directory: `cd <your forked repository name>`
    - Create a branch: `git checkout -b featureB`
    - **Tip:** Use descriptive branch names, just like Partner A!
    - **(Add Image: Screenshot of terminal showing navigating to the directory and creating featureB branch)**

Part 3: Building Awesome Features (Both Partners)

It's coding time! Both partners will now work simultaneously.

1. **Make your changes:**
    - **Partner A:** Stay on the "featureA" branch and add your amazing code.
    - **Partner B:** Stay on the "featureB" branch and build your fantastic feature.
    - Be creative! You can add new files, modify existing ones, whatever your project needs.
2. **Commit your changes:**
    - **Both Partners:** Remember these commands:
        - `git add <filename>` (to stage changes in a specific file)
        - `git add .` (to stage all changes)
        - `git commit -m "Your descriptive commit message"` (to save your changes with a message)
    - **(Add Image: Screenshot of the terminal showing using git add and git commit commands)**
    - Make frequent commits with clear messages to track your progress.
3. **Push your branch to GitHub:**
    - **Partner A:** `git push origin featureA`
    - **Partner B:** `git push origin featureB`
    - This sends your branch and its changes to your respective GitHub repositories.
    - **(Add Image: Screenshot of the terminal showing using the git push command)**

Part 4: Merging Feature A (Partner A)

Partner A, back to you!

1. **Switch to the main branch:**
    - `git checkout main`
2. **Merge your feature branch:**
    - `git merge featureA`
3. **Push the changes to GitHub:**
    - `git push origin main`
    - Your "featureA" is now part of the main project! 🎉

Part 5: Integrating Feature B (Partner B)

Partner B, let's bring your feature in!

1. **Fetch changes from the original repository:**
    - Add Partner A's original repository as a remote (if you haven't already):
        - `git remote add upstream <Partner A's original repository URL>`
    - Fetch changes from Partner A's main branch:
        - `git fetch upstream main`
    - **(Add Image: Screenshot of terminal showing adding remote and fetching changes)**
2. **Merge the updated main branch into your feature branch:**
    - `git checkout featureB`
    - `git merge upstream/main`
    - This incorporates any changes Partner A made to the main branch while you were working.
3. # Resolve merge conflicts (if any): - What's a merge conflict? Sometimes, when merging branches, Git encounters conflicting changes. It'll mark these conflicts in your files, and you need to decide which changes to keep. - Example: ``` <<<<<<< HEAD This is the change I made on my branch (featureB).

    This is the change Partner A made on the main branch.

    ||||||| *upstream/main*

```
- You need to edit this section to choose one of the changes, or combine them:
```

This is the final resolved change for both branches.

```
-  After fixing conflicts, use `git add <filename>` to stage the resolved
files.
-  Commit the merge: `git commit -m "Merged upstream/main into featureB"`
-  **(Add Image: Screenshot of a merge conflict in a code editor and resolving
it)**
```

4. **Push your merged branch:**
    - `git push origin featureB`
5. **Create a Pull Request:**
    - Go to your forked repository on GitHub.
    - You'll see a banner suggesting to create a pull request for "featureB". Click it!
    - **Best Practices:**
        - **Title:** Clear and concise, summarizing the change (e.g., "Add user authentication")
        - **Description:** Provide more context, explain your changes, and link to related issues if any. Use markdown for formatting!
    - **(Add Image: Screenshot of creating a pull request on GitHub)**

Part 6: Reviewing and Merging the Pull Request (Partner A)

Partner A, one last step!

1. **Review the Pull Request:**
    - Go to the "Pull requests" tab in **your original repository**.
    - You'll see Partner B's pull request.
    - **Careful Review:** Read the changes, leave comments, and discuss any questions or suggestions.
    - **(Add Image: Screenshot of reviewing a pull request on GitHub)**
2. **Merge the Pull Request:**
    - If everything looks good, click the big green "Merge pull request" button.
    - Congratulations! Both of your features are now part of the main project. 🎉

Challenge Completed! 🏆

You and your partner have successfully:

- Created and forked a repository
- Worked on separate branches
- Merged branches and resolved potential conflicts
- Created and merged a pull request

To go even further:

- **Explore different Git commands:** There's a whole world of Git out there! Check out the Git documentation and try new things.

- **Contribute to open-source projects:** Use your newfound Git skills to collaborate on real-world projects. You can find projects to contribute to on platforms like GitHub and GitLab.

Remember: GitHub's interface can change. If something looks different, don't worry! Refer to the official GitHub documentation for the most up-to-date instructions.

Happy coding!