Workflow

1. Create a new, **blank** private repository on your own github account, and name it as `<project_name>_<issue_branch_name>_sync`(eg. `template_issue2_chipsplot_sync`).
2. Clone the development branch `<issue_branch_name>` to mirror repository(so it becomes a `master` branch in that repo by default).
3. Pull the `master` branch of mirror repository into Overleaf, creating a mirror project, make edits and push back to that `master` branch.
4. Merge the `master` branch in mirror repository back to development branch `<issue_branch_name>`.
5. Repeat step 3 and 4 when there is demand for edit. Merge the development branch to the main branch and delete the mirror repository as well as the (mirror) Overleaf project after you close that issue.

Collaboration

To collaborate with your collaborators, you only need to share the Overleaf project. By sharing with your collaborators, they can commit changes to the document in real-time(more details are provided in Step 3 below), without needing permissions for the associated mirror repository on GitHub.

Step-by-Step Instructions

1. Create a new private repository

Log in your personal account, click on the "+" sign at the top right corner of the page. From the dropdown menu, select "New repository." You'll be taken to the "Create a new repository" page. Please make sure you create the repository via Github Online, instead of Github Desktop.

Fill in the following details:

- Repository Name: Choose the name `<project_name>_<issue_branch_name>_sync` for your repository.
- Visibility: Choose "Private." This will ensure that only you can access the repository's content.
- Initialize this repository with: Leave this option unchecked. We want to create a blank repository without any initial files.
- Add .gitignore: You can skip this step by scrolling down.
- Add a License: Since we're creating a blank repository, you won't be adding any code initially. You can skip this step by scrolling down.

Once you've filled in the necessary details, click the "Create repository" button at the bottom of the page.

2. Clone the development branch to the new mirror repository

- First switch to the branch you want to sync with:

```
git checkout <issue_branch_name>
```

- Add `<project_name>_<issue_branch_name>_sync` as a new remote repository in local repository:

```
git remote add <remote_repo_name> git@github.com:
<your_username>/<project_name>_<issue_branch_name>_sync
```

- Push the branch to the remote repo `<project_name>_<issue_branch_name>_sync`:

```
git push <remote_repo_name> <issue_branch_name>
```

- If (in the future) there's further pushes on the original development branch, we can *navigate to the mirror repo directory*, and sync to the mirror repo by executing these commands:

```
git fetch origin <issue_branch_name>
git remote add origin_<issue_id> git@github.com:<your_username>/<project_name>
git fetch origin_<issue_id>
git checkout -b origin_<issue_id> origin_<issue_id>/<issue_branch_name>
git checkout <issue_branch_name>
git merge origin_<issue_id>
git push origin <issue_branch_name>
```

3. Pull the master branch of the new mirror repository into overleaf, make edits and push back

- Open a new Overleaf project and import from GitHub:
  Go to the Overleaf Website and log in to your account. In your Overleaf, click on the "New Project" at the top left corner. Select "Import from GitHub" from the dropdown menu. If you haven't connected your Overleaf account to GitHub before, you might need to refer here to authorize the connection. If connected, a window will pop up showing your GitHub repositories. Choose the new mirror repository (the one where you pushed the development branch earlier). Click the "Import to Overleaf" button. Overleaf will start importing the repository and its files.
- Edit in Overleaf:
  Once the import is complete, you'll see the files and you can make edits directly in the Overleaf editor. As you make edits, Overleaf will automatically save your changes. You can preview your changes by clicking the "Recompile" button or the "Preview" button in Overleaf.
- Commit and push changes back to GitHub:
  After you've made your edits and are satisfied with the changes, go back to the "Menu" and click on "GitHub Sync." Select "Push Overleaf Changes to GitHub." Overleaf will create a commit with your changes and push it to the new mirror repository of development branch. Then you can view Changes on GitHub:

4. Merge the master branch in the mirror repository back to the development branch.

- Fetch the remote branch:
  In your terminal, navigate to the directory of your local repository (the original repository). Fetch the the `master` branch from the `<remote_repo_name>` remote mirror repository.

```
git fetch <remote_repo_name> <issue_branch_name>
```

- Create and switch to a new local temporary branch:
  Create a new local branch named `temp_<issue_id>` that will temporarily hold the changes from the fetched remote branch.

  ```
  git checkout -b temp_<issue_id> <remote_repo_name>/<issue_branch_name>
  ```

- Merge the changes into the development branch and push it to github.
  Switch back to the development branch of the original repository.

  ```
  git checkout <issue_branch_name>
  ```

  Merge the changes from the temp branch into the development branch.

  ```
  git merge temp_<issue_id>
  ```

  Once the merge is complete, push the changes to the `development` branch of the original repository on GitHub.

  ```
  git push origin <issue_branch_name>
  ```

Automation

We can use GitHub Actions workflow automates syncing between a development branch and a mirror repository(Step 2 and Step 4 above). It triggers on pushes to specified branches, copies changes to the mirror repository, and also triggers on push on the mirror repo to merges changes back into the original dev branch. This ensures both repositories stay in sync, simplifying the syncronization process.

Setup

Please strictly follow the instructions below.

Before you setup Github Action, please make sure the mirror repository is not a blank repo(that means you cannot use Automation in your first synchronization, to do your first original -> mirror synchronization, see here).

1. Setup GitHub Actions Secrets

- The owner of the repository(or collaborators if it's a lab project) generate an SSH key without a passphrase on his local machine. (If he has already set up, then skip this.) Add this SSH public key to owner's GitHub account under Settings -> SSH and GPG keys.

- Store the corresponding SSH private key in your GitHub Actions Secrets in *both* mirror repo and original repo, and name it as `SSH_PRIVATE_KEY`, please refer here for details. This allows GitHub Actions to authenticate as you, without needing to enter a passphrase. You should copy the full key, including `-----BEGIN OPENSSH PRIVATE KEY-----`, and `-----END OPENSSH PRIVATE KEY-----`.
- Create a new GitHub Actions Secrets named "email", and add the email of your Github Account into that Secret.

2. Automate Step 2: Clone development branch to new mirror repository

You can set up a GitHub Actions workflow in your original repository. When a new commit is made to the development branch (e.g., <issue_branch_name>), this workflow will automatically push the branch to a new mirror repository (e.g., <project_name><*issue_branch_name*>sync).

Go to your_repository -> Actions -> set up a workflow yourself/New workflow, and paste the lines of `Sync_to_mirror.txt` into the `.yml` workflow. Please rename your `.yml` file as `Sync_to_mirror.yml`.

Sync_to_mirror.txt

After adding that workflow, you *must pull the update* from `master` to your development branch, and push to github, in order to add `.github/workflows/Sync_to_mirror.yml` file. If you have already made edits on the development branch, please do rebase pull.

3. Automate Step 4: Merge the main branch of the mirror repository back to the development branch

You can also set up a GitHub Actions workflow in the mirror repository. When a new commit is made to the main branch, it will automatically merge these changes into the development branch of the original repository.

Go to repository -> Actions -> set up a workflow yourself/New workflow, and paste the lines of `Sync_back_dev_branch.txt` into the `.yml` workflow. Please rename your `.yml` file as `Sync_back_dev_branch.yml`.

Sync_back_dev_branch.txt

Notice

We currently get both "success"/"failed" messages for the Actions even if it is working as expected. You can change the settings to mute the notification of Github Action here in System -> Actions.

Known limitations for Overleaf Git system

- Symlinks: The Overleaf Git system does not handle Symlinks. A symlink can be pushed into an Overleaf project, but will be converted to a regular file, and will over-write the local symlink the next time the project is pulled.
- File Permissions: The Overleaf Git system does not preserve execute permissions. When working with GitHub sync, if any files with the execute bit in GitHub are synced to Overleaf, an automatic commit is triggered by Overleaf to reset the file permissions.
- Git LFS: Overleaf projects do not support Git Large File Storage.

- Connecting an existing project to an existing repository: It is not possible to synchronize an existing Overleaf project with an existing Github repository. That's the reason why we need to create a blank project initially.
- Authorship of pushes to GitHub: The only Github user that Overleaf is aware of is the user who's GitHub account is linked to the Overleaf account and owns that project. All commits are made using a GitHub API which explicitly makes those commits using that users's identity. So all commits will show in the GitHub Git history as coming from the linked user.