git practices | Yessmine

Steps to Push Changes from `develop` to `main`

1. Switch to the `develop` Branch

First, ensure you are on the `develop` branch where your changes are located. If you're not already on `develop`, switch to it using the following command:

```
git checkout develop
```

2. Pull Latest Changes (Optional)

It's a good practice to fetch and merge any changes from the remote repository (`origin`) before proceeding with your own changes. This helps in avoiding conflicts and keeps your local repository up-to-date.

```
git fetch origin
git merge origin/develop
```

3. Add and Commit Changes

Stage your changes (if any) and commit them to the `develop` branch. Replace `<files>` with the specific files or directories you want to include in this commit.

```
git add <files>
git commit -m "Commit message describing your changes"
```

4. Push Changes to Remote `develop`

Push the committed changes from your local `develop` branch to the remote `develop` branch (`origin/develop`).

```
git push origin develop
```

This command sends your local commits from `develop` to the remote repository (`origin`).

5. Switch to the `main` Branch

Now, switch to the `main` branch where you want to merge your changes from `develop`.

```
git checkout main
```

6. Merge `develop` into `main`

Merge the `develop` branch into `main` to integrate your changes. This step combines the changes from `develop` into `main`.

```
git merge develop
```

7. Push `main` to Remote

Finally, push the merged `main` branch to the remote repository (`origin`). This step updates the `main` branch on the remote repository with your changes from `develop`.

```
git push origin main
```

Notes:

- **Branch Switching**: Use `git checkout <branch>` to switch between branches (`develop`, `main`, etc.).
- Avoid Committing Directly to `main` , use instead either the `develop` branch or create a new one.
- **Atomic Commits:** Make each commit a logically separate change that represents a single unit of work. This makes it easier to review, revert, and understand changes over time.
- **Commit Messages:** Write clear and descriptive commit messages that explain the what and why of each change.
- Do not use `git push` from the rbpi directly to avoid merging problems. Only use `git pull` to test your code.