Git Workflow Blueprint

1. Create a New Feature/Development Branch

Whenever you begin work on a new feature or change, start by creating a new branch off the latest version of `master`.

```
git checkout -b <branch_name>
```

- This command creates a new branch named `<branch_name>` and switches to it immediately.
- Make sure to use a descriptive branch name, like `feature/currency-add` or `bugfix/fix-login-issue`, to make branch purposes clear. Also Include JIRA Ticket in the branch name.

---

2. Make Code Changes in the Branch

- While in your branch, make your code changes as needed.
- After making changes, add those changes to the staging area:

```
git add .
```

- This command stages all changes in the current directory.
- You can also specify specific files if you don't want to add all files.

---

3. Commit Changes Locally

After staging your changes, commit them locally with a descriptive message:

```
git commit -m "Add currency feature"
```

- Each commit message should clearly describe the purpose of the change.
- Repeat Steps 2 and 3 for each set of changes you make.

---

4. Sync with the Latest `master` Branch

Before merging your branch, make sure your feature branch is up to date with the latest changes from the `master` branch. This helps avoid conflicts during the final merge.

1. **Switch to the `master` branch**:

   ```
   git checkout master
   ```

2. **Pull the latest changes from the remote `master` branch**:

   ```
   git pull origin master
   ```

   - This command updates your local `master` branch with any new commits from the remote repository.
   - **Important**: Always ensure you have the latest changes from `master` before merging.

3. **Switch back to your feature branch**:

```
git checkout <branch_name>
```

4. **Merge the latest `master` changes into your branch**:

```
git merge master
```

   - This step integrates any new changes from `master` into your branch, helping prevent conflicts in the final merge.

---

5. Finalize and Merge Changes into `master`

Once your work is complete, and you've tested your changes, follow these steps to merge your branch into `master`.

1. **Switch to the `master` branch**:

```
git checkout master
```

2. **Merge your feature branch into `master`**:

```
git merge <branch_name>
```

   - This command applies all changes from your branch into `master`.
   - Resolve any conflicts that may arise during the merge.

---

6. Push Changes to Remote `master` Branch

Finally, push the updated `master` branch to the remote repository so that others can see the changes.

```
git push origin master
```

- This command sends your local `master` branch to the remote repository.

Summary of Commands

1. **Create a branch**: `git checkout -b <branch_name>`
2. **Add changes**: `git add .`
3. **Commit changes**: `git commit -m "Commit message"`
4. **Sync with latest `master`**:
   - `git checkout master`
   - `git pull origin master`
   - `git checkout <branch_name>`
   - `git merge master`
5. **Merge to `master`**:
   - `git checkout master`
   - `git merge <branch_name>`
6. **Push to remote `master`**: `git push origin master`

Best Practices for Collaboration

1. **Keep Branches Up-to-Date**: Regularly merge changes from `master` into your branch to avoid large conflicts.
2. **Use Descriptive Commit Messages**: Commit messages should make it clear what each change accomplishes.
3. **Pull from Remote Regularly**: Always pull from `master` before starting work and before merging.
4. **Review Changes Before Pushing**: Make sure code is tested and reviewed by team members before final merges into `master`.