Git

Git: Working with Branches and Merging to Master

1. Setting Up Git and Checking Branches

Before starting, make sure you have Git installed and initialized in your repository.

```
sh
CopyEdit
git init   # Initialize Git (only if it's a new repository)
git status # Check the current status of your repo
git branch # List available branches
```

---

2. Creating a New Branch

To create a new branch and switch to it:

```
sh
CopyEdit
git branch <branch-name>  # Create a new branch
git checkout <branch-name> # Switch to the new branch
```

Alternatively, you can create and switch in one step:

```
sh
CopyEdit
git checkout -b <branch-name>
```

Using Git switch (newer command):

```
sh
CopyEdit
git switch -c <branch-name>
```

---

3. Pushing a Branch to Remote

Once you've made changes in your branch and committed them, push the branch to the remote repository:

```
sh
CopyEdit
git push -u origin <branch-name>
```

This will create a new branch on the remote repository and set up tracking.

---

4. Merging a Branch into Master

Step 1: Switch to the master branch

Before merging, switch to the master/main branch:

```
sh
CopyEdit
git checkout master  # Or use 'main' if your default branch is named 'main'
```

Alternatively, using `switch`:

```
sh
CopyEdit
git switch master
```

Step 2: Merge the Branch into Master

Run the following command to merge the feature branch into the master branch:

```
sh
CopyEdit
git merge <branch-name>
```

Step 3: Push the Changes to Remote

After merging, push the updated master branch to the remote repository:

```
sh
CopyEdit
git push origin master
```

---

5. Merging Master into a Branch

If you want to update your branch with the latest changes from master:

Step 1: Switch to Your Branch

```
sh
CopyEdit
git checkout <branch-name>
```

Step 2: Merge Master into Your Branch

```
sh
CopyEdit
git merge master
```

If there are merge conflicts, Git will prompt you to resolve them manually before completing the merge.

Step 3: Push Updated Branch to Remote

After merging master into your branch, push the updated branch to the remote repository:

```
sh
CopyEdit
git push origin <branch-name>
```

---

6. Resolving Merge Conflicts

If there are conflicts during merging, Git will notify you. Open the conflicted files, resolve the conflicts manually, then use:

```
sh
CopyEdit
git add <file-name>  # Mark file as resolved
git commit -m "Resolved merge conflict"
git push origin master
```

---

7. Deleting a Branch

After merging, you might want to delete the feature branch.

Delete a Local Branch

```
sh
CopyEdit
git branch -d <branch-name>
```

If the branch hasn't been merged yet, use:

```
sh
CopyEdit
git branch -D <branch-name>
```

Delete a Remote Branch

```
sh
CopyEdit
git push origin --delete <branch-name>
```

---

8. Summary of Commands

Action	Command
Create a new branch	`git branch <branch-name>`
Switch to a branch	`git checkout <branch-name>` / `git switch <branch-name>`

Create and switch	`git checkout -b <branch-name>` / `git switch -c <branch-name>`
Push branch to remote	`git push -u origin <branch-name>`
Merge a branch into master	`git checkout master` → `git merge <branch-name>`
Push master to remote	`git push origin master`
Merge master into a branch	`git checkout <branch-name>` → `git merge master`
Delete a local branch	`git branch -d <branch-name>`
Delete a remote branch	`git push origin --delete <branch-name>`

---

9. Rebasing a Branch (Alternative to Merging)

Instead of merging, you can **rebase** your branch onto the latest master to keep a cleaner history:

Step 1: Switch to Your Branch

```
sh
CopyEdit
git checkout <branch-name>
```

Step 2: Rebase onto Master

```
sh
CopyEdit
git rebase master
```

This will apply your branch's commits on top of the latest master branch, avoiding unnecessary merge commits.

If there are conflicts, resolve them, then continue rebase:

```
sh
CopyEdit
git add <file-name>
git rebase --continue
```

Push the updated branch to remote with:

```
sh
CopyEdit
git push origin <branch-name> --force
```

---

10. Stashing Changes Before Switching Branches

Sometimes, you might have uncommitted changes but need to switch branches. You can **stash** your changes:

Step 1: Stash Your Changes

```
sh
CopyEdit
git stash
```

This temporarily saves your uncommitted changes.

Step 2: Switch to Another Branch

```
sh
CopyEdit
git checkout master
```

Step 3: Apply the Stashed Changes

```
sh
CopyEdit
git stash pop
```

This restores your previous changes.

With these extra points, you have an even stronger understanding of Git for real-world development! 🚀 Let me know if you need more details. 😊