Commonly Used Git Commands:

Below are the steps and Git commands you can directly apply to clone a repository, make changes, commit them to a branch, and resolve merge conflicts.

**### Step 1: Clone the repository**

First, you need to clone the repository to your local machine.

```bash

git clone <repository-url>

cd <repository-folder>

```

Replace `<repository-url>` with the URL of the repository you want to clone and `<repository-folder>` with the directory name of the cloned repository.

**### Step 2: Create a new branch**

It's a good practice to create a new branch for the changes you want to make. This way, you don't commit directly to the `main` or `master` branch.

```bash

git checkout -b <your-branch-name>

```

Replace `<your-branch-name>` with a name for your new branch (e.g., `feature/new-feature`).

NOTE: Use below command to find your current branch

**git branch**

**### Step 3: Make changes to a file**

Open the file you want to change in your text editor, make the necessary modifications, and save the file.

**### Step 4: Stage the changes**

After making the changes, stage the modified files to be committed.

```bash

git add <file-name>

```

For all files you've changed, you can use:

```bash

git add .

```

### Step 5: Commit the changes

Now commit the staged changes with a meaningful message.

```bash

git commit -m "Your commit message describing the changes"

```

Use below COmmands in case you get an error, when pushing to remote repository,

**git config --global user.email "gitgit782@gmail.com"**

**git config --global user.name "Sample782"**

### Step 6: Push your changes to the remote repository

Push your branch with the changes to the remote repository.

```bash

git push origin <your-branch-name>

```

### Step 7: Fetch the latest changes from the remote repository (optional but recommended)

Before merging your branch into the main branch, it's a good idea to get the latest changes from the main branch to avoid conflicts.

```bash

git fetch origin

git checkout main

git pull origin main

```

### Step 8: Merge your branch with the latest `main` branch

Switch back to your branch and merge the latest changes from `main` into it to ensure everything is up to date.

```bash

git checkout <your-branch-name>

git merge main

```

If there are no conflicts, it will automatically merge. If there are conflicts, you'll need to resolve them manually.

### Step 9: Resolve merge conflicts (if any)

If there are merge conflicts, Git will mark the files with conflicts. Open each conflicted file, look for sections marked with conflict markers (`<<<<<<<`, `=======`, and `>>>>>>>`), and resolve the conflicts.

Example of a conflict:

```plaintext

<<<<<<< HEAD

This is the content from the main branch.

=======

This is the content from your branch.

>>>>>>> <your-branch-name>

```

You need to decide which changes to keep or how to combine them, then remove the conflict markers. After resolving the conflicts, stage the files again:

```bash

git add <resolved-file>

```

Repeat this for each conflicted file.

### Step 10: Commit the merge

Once all conflicts are resolved and staged, commit the merge.

```bash

git commit -m "Resolved merge conflicts"

```

### Step 11: Push the resolved changes

Finally, push the resolved branch back to the remote repository.

```bash

git push origin <your-branch-name>

```

### Step 12: Create a Pull Request (optional)

If you're using a platform like GitHub, GitLab, or Bitbucket, go to the website and create a pull request (PR) to merge your changes from your branch into the `main` branch.

---

### Summary of commands:

```bash

git clone <repository-url>

cd <repository-folder>

git checkout -b <your-branch-name>

# make changes to a file

git add <file-name>

git commit -m "Your commit message"

git push origin <your-branch-name>

git fetch origin

git checkout main

git pull origin main

git checkout <your-branch-name>

git merge main

# resolve conflicts if necessary

git add <resolved-file>

git commit -m "Resolved merge conflicts"

git push origin <your-branch-name>

```

With these steps, you should be able to clone a repo, make changes, commit them, and resolve merge conflicts when necessary. Let me know if you need further clarification!

**Commands to Merge branch2(feature\_6532) to branch1(main)**

To merge updates from `branch2` into `branch1`, follow these steps:

### Step 1: Switch to `branch1`

You need to be on `branch1` to merge updates from `branch2`.

```bash

git checkout branch1

```

### Step 2: Pull the latest changes (optional, but recommended)

If `branch1` is already tracking the remote repository, make sure it’s up to date by pulling the latest changes.

```bash

git pull origin branch1

```

### Step 3: Merge `branch2` into `branch1`

Now that you're on `branch1`, you can merge the changes from `branch2`:

```bash

git merge branch2

```

- If there are \*\*no conflicts\*\*, Git will automatically merge the changes and you'll see a success message.

- If there are \*\*merge conflicts\*\*, you'll need to resolve them manually (see Step 4).

### Step 4: Resolve merge conflicts (if any)

If there are conflicts, Git will mark the files that need to be resolved. Open the conflicted files in your text editor, and you'll see sections like this:

```plaintext

<<<<<<< HEAD

Changes from branch1

=======

Changes from branch2

>>>>>>> branch2

```

- Resolve the conflicts by deciding which changes to keep, or combine them if needed.

- After resolving the conflicts, \*\*remove the conflict markers\*\* (`<<<<<<<`, `=======`, and `>>>>>>>`).

### Step 5: Stage the resolved files

After resolving the conflicts, stage the changes to mark the conflict as resolved.

```bash

git add <resolved-file>

```

You can also stage all resolved files using:

```bash

git add .

```

### Step 6: Commit the merge

Once all conflicts are resolved and staged, commit the merge.

```bash

git commit -m "Merged branch2 into branch1 and resolved conflicts"

```

### Step 7: Push the updated `branch1` to the remote repository

Finally, push the changes to the remote `branch1` to make sure it's up to date.

```bash

git push origin branch1

```

---

### Summary of Commands:

```bash

git checkout branch1 # Switch to branch1

git pull origin branch1 # Pull the latest changes from remote (optional)

git merge branch2 # Merge changes from branch2 into branch1

# Resolve conflicts (if any)

git add <resolved-file> # Stage resolved files

git commit -m "Merged branch2 into branch1" # Commit the merge

git push origin branch1 # Push the changes to remote

```

With these steps, you'll have successfully merged the updates from `branch2` into `branch1`. Let me know if you need more details!