# Git Setup and Workflow Guide

## Setting Up Git on a New Machine

Git is a version control system used to track changes in code and collaborate with others. To install Git on a new machine, follow these steps:

1. Update the package list to get the latest available versions:

sudo apt update

2. Install Git by running:

sudo apt install git

Once Git is installed, you can check its version using:

git --version

## Steps to Push Data into Git Repository

After making changes to the code, you need to push those changes to a Git repository. Follow these steps:

1. Add only the necessary files to the staging area (avoid using `git add .` to prevent adding unwanted files):

git add qbank/\* static/\* templates/\*

2. Commit the changes with a meaningful message:

git commit -m "Enter a descriptive commit message"

If you want to enter a detailed commit message, simply run:

git commit

3. Push the committed changes to the remote repository on the master branch:

git push origin master

4. If others have made changes, you should pull the latest updates before pushing:

git pull origin master

## Branch Management

Branches in Git allow developers to work on new features or fixes without affecting the main codebase.

Switch to the `dev` branch:

git checkout dev

Create a new branch for a feature or bug fix:

git branch <branch-name>

Switch to an existing branch:

git checkout <branch-name>

Create a new branch and switch to it immediately:

git checkout -b <branch-name>

Delete a branch when it is no longer needed:

git branch -d <branch-name>

## Merging Branches

After finishing work on a feature or fix, you should merge it into the main branch. To merge a branch into `master` (or another branch), first switch to that branch and run:

git merge <branch-name>

## Checking Git Status and Logs

You can check the current status of your repository with:

git status --short

To view the commit history in a compact format:

git log --oneline

## Stashing Changes

If you need to switch branches but have uncommitted changes, you can stash them temporarily:

git stash

List all stashed changes:

git stash list

Apply the most recent stash:

git stash apply

## Reverting and Resetting Commits

If you need to undo a commit, you have multiple options:

1. To revert a commit without losing history:

git revert <commit-hash>

2. To reset the repository to a specific commit:

git reset <commit-hash>

- `--soft`: Keeps changes staged.

- `--mixed`: Unstages changes but keeps them.

- `--hard`: Deletes changes completely.

3. If you accidentally use `git reset --hard`, you can recover lost commits using:

git reflog

## Fetching and Comparing Changes

To fetch the latest changes from the remote repository without merging them:

git fetch origin dev

To check how many commits your branch is behind the remote branch:

git status

To compare differences between your local and remote branches:

git diff --name-only HEAD origin/<branch>

To see differences in a specific file:

git diff HEAD origin/<branch> -- <file>

## Other Useful Git Commands

To apply a specific commit from another branch onto the current branch:

git cherry-pick <commit-hash>

To edit the last commit message:

git commit --amend -m "New commit message"