**Git Workflow**

**Basic Git Workflow**

1. **Initialize a Repository**:

git init

1. **Clone a Repository**:

git clone [url]

1. **Check Repository Status**:

git status

1. **Stage Changes**:

git add [file]

1. **Commit Changes**:

git commit -m "commit message"

1. **Push Changes**:

git push origin [branch-name]

1. **Pull Changes**:

git pull origin [branch-name]

1. **Create a Branch**:

git checkout -b [branch-name]

1. **Switch Branches**:

git checkout [branch-name]

1. **Merge Branches**:

git checkout [target-branch]

git merge [source-branch]

**Advanced Topics**

1. **Rebase**: Rebase is another way to integrate changes from one branch into another. It is often used to keep a linear commit history.

git rebase [branch]

1. **Stashing**: Stashing allows you to save your changes temporarily without committing them.

git stash

git stash apply

1. **Tagging**: Tags are used to mark specific points in history as important, such as releases.

git tag [tag-name]

git push origin [tag-name]

1. **Cherry-Pick**: Cherry-picking is the process of applying a commit from one branch onto another.

git cherry-pick [commit-hash]

1. **Conflict Resolution**: When merging branches, conflicts might occur if changes overlap. Git will prompt you to resolve these conflicts manually.

**Best Practices**

1. **Commit Often**: Frequent commits make it easier to track changes and revert if needed.
2. **Write Meaningful Commit Messages**: Clear messages help understand the history and purpose of changes.
3. **Use Branches**: Isolate features and fixes in separate branches to keep the main branch stable.
4. **Regularly Pull Updates**: Keep your local repository up-to-date with the remote to avoid conflicts.
5. **Review Code Before Merging**: Code reviews help maintain code quality and catch potential issues early.