Git Intermediate Workshop

Git Bisect

What is Git Bisect?

Git Bisect is a command that allows you to find the commit that introduced a bug.

When to use Git Bisect?

Git Bisect is useful when you want to find the commit that introduced a bug. For example, you have a branch called `feature` and you want to find the commit that introduced a bug from `feature`. You can use Git Bisect to do that.

How to use Git Bisect?

1. Run `git bisect start`.
2. Run `git bisect bad`. This will mark the current commit as bad.
3. Run `git bisect good <commit hash>`. This will mark the commit as good.
4. Git will checkout to a commit between the good commit and the bad commit.
5. Run `git bisect good` if the commit is good or `git bisect bad` if the commit is bad.
6. Repeat step 6 until you find the commit that introduced a bug.

Git Revert

What is Git Revert?

Git Revert is a command that allows you to undo a commit.

It does not delete the commit, but it creates a new commit that undoes the changes from the previous commit.

When to use Git Revert?

Git Revert is useful when you want to undo a commit. For example, you have a branch called `feature` and you want to undo a commit from `feature`. You can use Git Revert to do that.

How to use Git Revert?

1. Checkout to the branch that you want to undo the commit from. For example, `feature`.
2. Run `git revert <commit hash>`. For example, `git revert 1234567890`.
3. If there is no conflict, you are done. If there is a conflict, resolve the conflict and run `git revert --continue`.

Git Cherry Pick

What is Git Cherry Pick?

Git Cherry Pick is a command that allows you to take a commit from one branch and apply it onto another.

When to use Git Cherry Pick?

Git Cherry Pick is useful when you want to apply a commit from one branch to another branch. For example, you have a branch called `feature` and you want to apply a commit from `feature` to `main`. You can use Git Cherry Pick to do that.

How to use Git Cherry Pick?

1. Checkout to the branch that you want to apply the commit to. For example, `main`.
2. Run `git cherry-pick <commit hash>`. For example, `git cherry-pick 1234567890`.
3. If there is no conflict, you are done. If there is a conflict, resolve the conflict and run `git cherry-pick --continue`.

Git Rebase

What is Git Rebase?

Git Rebase is a command that allows you to change the base of a branch.

When to use Git Rebase?

Git Rebase is useful when you want to change the base of a branch. For example, you have a branch called `feature` and you want to change the base of `feature` to `main`. You can use Git Rebase to do that.

How to use Git Rebase?

1. Checkout to the branch that you want to change the base of. For example, `feature`.
2. Run `git rebase <base branch>`. For example, `git rebase main`.
3. If there is no conflict, you are done. If there is a conflict, resolve the conflict and run `git rebase --continue`.

Git Rebase Interactive

What is Git Rebase Interactive?

Git Rebase Interactive is a command that allows you to change the base of a branch and modify the commits.

When to use Git Rebase Interactive?

Git Rebase Interactive is useful when you want to change the base of a branch and modify its commits. For example, you have a branch called `feature` and you want to change the base of `feature` to `main` and modify the commits. You can use Git Rebase Interactive to do that.

How to use Git Rebase Interactive?

1. Checkout to the branch that you want to change the base of. For example, `feature`.
2. Run `git rebase -i <base branch>`. For example, `git rebase -i main`.
3. A text editor will open. You can change the base of the branch and modify the commits.
4. If there is no conflict, you are done. If there is a conflict, resolve the conflict and run `git rebase --continue`.

Configuring GIT_EDITOR

GIT_EDITOR is an environment variable that allows you to change the text editor that Git uses.

How to configure GIT_EDITOR?

1. Run `git config --global core.editor <text editor>`. For example, `git config --global core.editor "code --wait"`.

Conclusion

Git Bisect, Git Revert, Git Cherry Pick, Git Rebase, and Git Rebase Interactive are useful commands that you can use to make your life easier and to keep your Git history clean.