

Advance Git & GitHub for DevOps Engineers

Day 10 : 90Days of DevOps Challenge

SS

Shivraj Salunkhe · Apr 3, 2023 · 📖 5 min read

What is Git Branching ?

1. Git branching is a feature in the Git version control system that allows developers to create and manage multiple branches of code within a single repository.
2. A branch is essentially a separate line of development that diverges from the main codebase, allowing developers to work on new features or fixes without affecting the main codebase until they are ready to merge their changes back in.
3. Git branching enables teams to collaborate on a project more efficiently, as it allows for parallel development and the ability to work on different features simultaneously.
4. It also provides a way to experiment with new ideas and to easily switch between different versions of the code.

Example :

- let's say you are working on a software project with a team of developers. You have a main branch called "**master**" that represents the stable version of the codebase.
- You want to work on a new feature that will take some time to develop, but you don't want to interfere with the stability of the "**master**" branch. So, you create a new branch called "**feature-x**" and start making your changes there.
- You can commit your changes to the "**feature-x**" branch as often as you want without affecting the "**master**" branch. Once you have completed the feature, you can merge the "**feature-x**" branch back into the "**master**" branch to incorporate your changes into the main codebase.
- Meanwhile, other developers on your team can work on their **own** branches for other features or fixes, all in parallel with your work on the "**feature-x**" branch.

What is Git Revert and Reset ?

Git revert and **reset** are two important commands in the Git version control system that allow developers to undo changes in their repository.

1.Git revert creates a new commit that undoes the changes made in a previous commit, effectively rolling back the changes. It does not delete the previous commit but rather creates a new one that negates its changes.

-> Here is an example of how to use `git revert`:

Let's say you have made a commit with a message "added new feature to the app", but you realize later that this feature caused some bugs and you want to undo the changes introduced in that commit. You can use the following command:

```
git revert <commit-hash>
```

COPY 

This will create a new commit that undoes the changes made in the specified commit.

2.Git reset moves the current branch pointer to a specified commit, effectively resetting the repository to that point in time. It can be used to undo local changes that have not yet been committed or to undo one or more commits in the repository. However, it should be used with caution as it can remove commits permanently and make them unrecoverable.

-> Here is an example of how to use `git reset`:

Let's say you have made some changes to a file but have not yet committed them, and you want to undo those changes. You can use the following command:

```
git reset <file-name>
```

COPY 

This will reset the file to the state it was in at the last commit.

Another example is if you want to undo the last two commits and revert the repository to the state it was in before those commits were made. You can use the following command:

```
git reset HEAD~2 --hard
```

COPY 

This will move the current branch pointer back two commits and discard any changes made in those commits. The `--hard` flag indicates that any changes made since the specified commit will be permanently deleted.

What is Git Rebase and Merge ?


Git rebase and **merge** are two ways of **integrating changes** from one branch to another in Git.

-> **Git rebase** is a command that allows you to apply the changes made in one branch onto another branch as if they were made on top of the other branch. It essentially moves the entire branch to a new starting point and replays the commits from the original branch on top of it.

Here is an example of how to use `git rebase`:

Let's say you have created a feature branch to work on a new feature and have made several commits. Meanwhile, another developer has made some changes to the main branch. You want to incorporate these changes into your feature branch before you merge it back into the main branch. You can use the following command:

```
git checkout feature  
git rebase main
```

COPY 

This will move the feature branch to the head of the main branch and replay the commits made on the feature branch on top of the main branch.

-> **Git merge** is a command that combines the changes made in one branch with another branch. It creates a new merge commit that contains the changes made in both branches. It is a non-destructive way of integrating changes and preserves the history of both branches.

Here is an example of how to use `git merge`:

Let's say you have created a feature branch to work on a new feature and have made several commits. Meanwhile, another developer has made some changes to the main branch. You want to incorporate these changes into your feature branch before you merge it back into the main branch. You can use the following command:

```
git checkout feature  
git merge main
```

This will create a new merge commit that combines the changes made in both the feature and main branches.

In summary, `git rebase` and `git merge` are both ways of integrating changes between branches in Git, but they do so in different ways. `git rebase` replays the changes from one branch on top of another, while `git merge` creates a new merge commit that combines the changes from both branches.

Thank You for Reading the Blog.. 🙌

connect with me : [linkedin.com/in/shivraj-salunkhe-5881141a4](https://www.linkedin.com/in/shivraj-salunkhe-5881141a4)

follow my blog channel : shivrajofficial.hashnode.dev



Subscribe to my newsletter

Read articles from directly inside your inbox.
Subscribe to the newsletter, and don't miss out.

Enter your email address

SUBSCRIBE

Devops

GitHub

Git

DevOps Journey

DevOps trends



WRITTEN BY

Shivraj Salunkhe

 Follow

Dedicated and hardworking undergraduate student pursuing a degree in Information Technology with a passion for learning, leadership experience, and real-world skills through internships and part-time jobs in IT sector. Actively looking for new Opportunities and committed to do personal and professional growth.

MORE ARTICLES

Shivraj Salunkhe

**Python Libraries for
DevOps**

Shivraj Salunkhe

**Python Data Types and
Data Structures for
DevOps**

Shivraj Salunkhe

**Basics of Python
Important for DevOps**

-> Reading JSON in Python To read JSON files in Python, you can use the json module. import json # ...

-> Data Types In DevOps, Python is a widely used programming language for scripting, automation, an...

What is Python ? Python is a high-level, interpreted programming language that is popular for its s...

©2023 Shivraj Salunkhe's Blog

[Archive](#) · [Privacy_policy](#) · [Terms](#)



Publish with Hashnode

Powered by [Hashnode](#) - Home for tech writers and readers