

# Advance Git & GitHub for DevOps Engineers: Part-2

## Day 11 : 90Days of DevOps Challenge

SS

Shivraj Salunkhe · Apr 6, 2023 ·  5 min read

### What is Git Stash?

- Git stash is a feature in Git that allows you to temporarily save changes that are not ready to be committed yet, so you can switch to a different branch or work on something else, and then come back to your changes later.
- The `git stash` command takes the current changes in your working directory and saves them to a stash, which is stored in a stack of stashes.

#### Here's how to use Git stash:

1. Make changes to your files in your working directory.
2. Run `git stash` to save your changes to a stash.
3. Switch to a different branch or work on something else.

4. When you're ready to work on your changes again, switch back to the original branch.
5. Run `git stash apply` to apply your changes back to your working directory.

**Here are some common Git stash commands:**

- `git stash` - Save changes to a stash.
- `git stash list` - List all stashes.
- `git stash apply` - Apply the most recent stash to your working directory.
- `git stash apply stash@{n}` - Apply a specific stash to your working directory.
- `git stash drop` - Remove the most recent stash.
- `git stash pop` - Apply the most recent stash and remove it from the stash stack.

**Here's an example of using Git stash:**

```
# Make some changes to your files
$ echo "Hello, world!" > myfile.txt
$ git add myfile.txt
$ git commit -m "Add myfile.txt"

# Make some more changes
$ echo "Goodbye, world!" > myfile.txt

# Save your changes to a stash
```

COPY 

```
$ git stash

# Switch to a different branch
$ git checkout some-other-branch

# Switch back to the original branch
$ git checkout master

# Apply your changes from the stash
$ git stash apply

# Make another commit
$ git add myfile.txt
$ git commit -m "Update myfile.txt"

# Remove the stash
$ git stash drop
```

## What is Cherry-pick ?

In Git, "cherry-pick" is a command that allows you to apply a specific commit from one branch onto another branch. It essentially takes the changes made in a specific commit and applies them to the current branch, creating a new commit with those changes.

**Here's how to use Git cherry-pick:**

1. Find the commit that you want to apply to the current branch. You can use `git log` to find the commit hash.
2. Switch to the branch that you want to apply the commit to.
3. Run the `git cherry-pick` command followed by the commit hash. This will apply the changes made in that commit onto the current branch.
4. Resolve any conflicts that may arise during the cherry-pick process.
5. Once all conflicts are resolved, commit the changes with `git commit`.

**Here's an example of using Git cherry-pick:**

```
# Switch to the branch you want to apply the commit to
$ git checkout my-branch

# Find the commit you want to apply
$ git log

commit 1234567890abcdef (HEAD -> some-other-branch)
Author: shiv raj <shivraj@example.com>
Date:   Tue Apr 6 11:23:45 2023 -0500

    Add new feature

# Cherry-pick the commit onto your current branch
$ git cherry-pick 1234567890abcdef
```

COPY 

```
# Resolve any conflicts that may arise
$ git status
# Resolve conflicts in myfile.txt
$ git add myfile.txt
$ git cherry-pick --continue

# Commit the changes
$ git commit -m "Cherry-pick commit 1234567890abcdef"
```

It's important to note that while cherry-picking can be useful in certain situations, it should be used with caution. Cherry-picking a commit from another branch can result in conflicts and unintended consequences, especially if the commit contains changes that are dependent on other commits in the original branch.

## What is Resolving Conflicts ?

In Git, conflicts can arise when you try to merge or apply changes from one branch to another and Git is unable to automatically reconcile the differences between the two sets of changes. When this happens, Git will pause the merge or apply process and prompt you to manually resolve the conflicts.

**Here's how to resolve conflicts in Git:**

1. Run `git status` to check which files have conflicts.

2. Open the files with conflicts and locate the conflict markers. These markers look like `<<<<<<`, `=====`, and `>>>>>>`.
3. Edit the file to resolve the conflicts, removing the conflict markers and keeping the changes that you want to keep.
4. Once you have resolved all conflicts in a file, save the file and run `git add` to stage the changes.
5. Continue the merge or apply process with `git merge --continue` or `git apply --continue`.
6. Repeat this process for all files with conflicts.


**Here's an example of resolving conflicts in Git:**

```
# Start a merge
$ git merge some-branch

# Check for conflicts
$ git status

# myfile.txt has conflicts
$ cat myfile.txt

<<<<<< HEAD
This is some text that was already here.
=====
This is some new text that was added in the other branch.
```

COPY 

```
>>>>>> some-branch

# Edit the file to resolve the conflict
$ nano myfile.txt

This is some text that was already here.
This is some new text that was added in the other branch.

# Stage the changes
$ git add myfile.txt

# Continue the merge
$ git merge --continue
```

It's important to note that resolving conflicts can be a complex and time-consuming process, especially if there are many conflicts or if the changes are extensive. It's important to carefully review and understand the changes being merged or applied, and to communicate with your team members if necessary to ensure that everyone is on the same page.

**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](https://shivrajofficial.hashnode.dev)



## Subscribe to my newsletter

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

SUBSCRIBE

GitHub

Git

Devops

DevOps Journey

Devops articles



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

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

**Shivraj Salunkhe**

### Python Data Types and Data Structures for DevOps

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

**Shivraj Salunkhe**

### Basics of Python Important for DevOps

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

