# Day 11 Advance Git & GitHub for DevOps Engineers: Part-2

[](https://hashnode.com/@DarshanaT)

[**Darshana Mahesh Takawale**](https://hashnode.com/@DarshanaT)

**·**[Apr 11, 2024](https://techwithdarshana.hashnode.dev/day-11-advance-git-github-for-devops-engineers-part-2)**·**

3 min read

**Git Stash:**

Git stash is a command that allows you to temporarily save changes you have made in your working directory, without committing them. This is useful when you need to switch to a different branch to work on something else, but you don't want to commit the changes you've made in your current branch yet.

To use Git stash, you first create a new branch and make some changes to it. Then you can use the command git stash to save those changes. This will remove the changes from your working directory and record them in a new stash. You can apply these changes later. git stash list command shows the list of stashed changes.

You can also use git stash drop to delete a stash and git stash clear to delete all the stashes.

**Cherry-pick:**

Git cherry-pick is a command that allows you to select specific commits from one branch and apply them to another. This can be useful when you want to selectively apply changes that were made in one branch to another.

To use git cherry-pick, you first create two new branches and make some commits to them. Then you use git cherry-pick <commit\_hash> command to select the specific commits from one branch and apply them to the other.

**Resolving Conflicts:**

Conflicts can occur when you merge or rebase branches that have diverged, and you need to manually resolve the conflicts before git can proceed with the merge/rebase. git status command shows the files that have conflicts, git diff command shows the difference between the conflicting versions and git add command is used to add the resolved files.

**Task-01**

* Create a new branch and make some changes to it.

COPY

COPY

$ git branch stashlearn

$ git checkout stashlearn

$ git branch



COPY

COPY

$ touch change1.txt

$ touch git add change1.txt

$ git stash

* Use git stash to save the changes without committing them.



* Switch to a different branch, make some changes and commit them.

COPY

COPY

$ git switch dev

$ touch change2.txt

$ git add change2.txt

$ git commit -m " Added change2 file in commit"

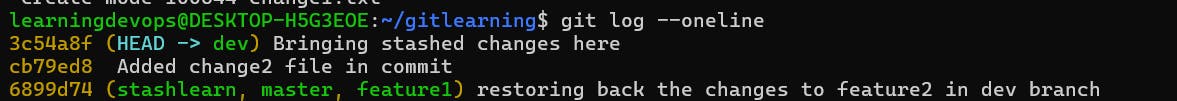
* Use git stash pop to bring the changes back and apply them on top of the new commits.

COPY

COPY

$ git stash pop

$ git commit -m "Bringing stashed changes here"



**Task-02**

* In version01.txt of development branch add below lines after “This is the bug fix in development branch” that you added in Day10 and reverted to this commit.

COPY

COPY

$ git switch development

* Line2>> After bug fixing, this is the new feature with minor alteration”

Commit this with message “ Added feature2.1 in development branch”

* Line3>> This is the advancement of previous feature

Commit this with message “ Added feature2.2 in development branch”

* Line4>> Feature 2 is completed and ready for release

Commit this with message “ Feature2 completed”



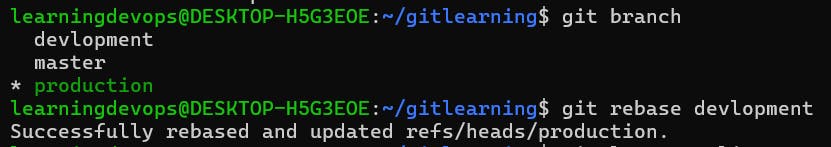
* All these commits messages should be reflected in Production branch too which will come out from Master branch (Hint: try rebase).

COPY

COPY

$ git checkout master

$ git checkout -b production





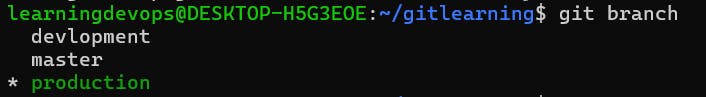
**Task-03**

* In Production branch Cherry pick Commit “Added feature2.2 in development branch”

COPY

COPY

$ git branch



Cherry pick commit from development branch by taking its commit id.

https://cdn.hashnode.com/res/hashnode/image/upload/v1712776236121/6427d37d-54a0-4a41-a460-c639f6016b2c.png?auto=compress,format&format=webp

and add below lines in it:

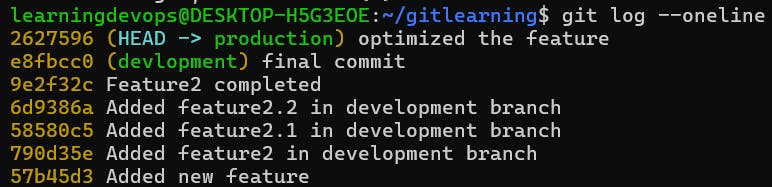
* Line to be added after Line3>> This is the advancement of previous feature
* Line4>>Added few more changes to make it more optimized.
* Commit: Optimized the feature

COPY

COPY

$ vi version01.txt

$ git commit -am "Optimized the feature"



Thanks for reading!!!

Happy Learning...!!!!