**#This File is Written by Sayali Jadhav**

**Day11: Advance Git & GitHub for DevOps Engineers: Part-2**

**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 the 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, the git diff command shows the difference between the conflicting versions and the git add command is used to add the resolved files.

**Task-01**

**1)Create a new branch and make some changes to it. Use git stash to save the changes without committing them.**

Text

Description automatically generated

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

Text

Description automatically generated

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

Text

Description automatically generated

**Task-02**

* In version01.txt of development branch add the below lines after “This is the bug fix in the development branch” that you added in Day10 and reverted to this commit.
* Line2>> After bug fixing, this is the new feature with minor alterations
* Commit this with the message “Added feature2.1 in development branch”
* Line3>> This is the advancement of the previous feature
* Commit this with the message “ Added feature2.2 in development branch”
* Line4>> Feature 2 is completed and ready for release
* A picture containing text

  Description automatically generatedCommit this with the message “ Feature2 completed”

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

* Here, I have changed from dev to production and used git rebase, now all the commits are merged in production.

Text

Description automatically generated

 **Task-03**

* In the Production branch Cherry pick Commit “Added feature2.2 in development branch” and added the below lines in it:

Text

Description automatically generated

Text

Description automatically generated

* The line to be added after Line3>> This is the advancement of the previous feature
* Line 4>>Added a few more changes to make it more optimized.
* Commit: Optimized the feature

