How to Restore a Git Stash

September 27, 2022

GIT

Introduction

A Git stash is unfinished work set aside in a local repository. Developers stash changes when they need to focus on a different issue and don't want to commit the changes yet. Stashed work cannot be seen by other developers and doesn't appear in other repositories.

When ready to continue working on the unfinished code, restore the changes or create a new branch from the Git stash.

In this tutorial, you will learn to restore a Git stash.

Prerequisites

- Git installed (install Git on Ubuntu, macOS, Windows, CentOS 7, or CentOS 8).
- A Git repository.

How to Restore a Git Stash

Git stores the latest stash in *refs/stash*, and names it **stash@{0}**. Each older stash is stored in the reflog of that reference and assigned an index number (e.g., **stash@{1}**, **stash@{2}**, etc.). Restoring a stash requires either the index number, or the stash name, if specified during stashing.

There are two ways to restore a Git stash:

- Using **git stash pop**.
- Using **git stash apply**.

The sections below explain both ways of restoring a Git stash.

1. Restore Git Stash Changes Using Git Stash Apply

The **git stash apply** command restores the stash but doesn't delete it from the reference. This keeps the stash in the reference until you manually delete it. The syntax is:

```
git stash apply stash@{n}
```

- For **{n}**, specify the stash index number. Find the stash index number by running:

```
git stash list
```

```
marij@BOSKO MINGW64 ~/Desktop/Git project (master)
$ git stash list
stash@{0}: WIP on master: 9fd7d12 New page file added
stash@{1}: WIP on master: 9fd7d12 New page file added
```

For example, to restore **stash@{0}**, run the following command:

```
git stash apply stash@{0}
```

```
marij@BOSKO MINGW64 ~/Desktop/Git project (master)
$ git stash apply stash@{0}
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   code.js
        new file:   new-page.html
        new file:   newfile.js
```

The command applies the stashed changes to the repository while keeping the stash in the reference.

Note: Git automatically assigns the stash index and a generic stash name based on the last commit, making it difficult to find the right stash. The solution to keeping stashes organized is to name a Git stash and later retrieve the stash by name.

2. Restore Git Stash Changes Using Git Stash Pop

The **`git stash pop`** command restores the stashed changes and schedules the stash for deletion from the reference. After popping the stash, Git states that the stash has been dropped and outputs the stash SHA value.

However, Git only schedules the stash for deletion, which means you can restore it using the SHA value from the output.

Note: See how to recover a deleted Git stash.

The syntax for popping a Git stash is:

```
git stash pop stash@{n}
```

For example, to pop **`stash@{1}`**, run:

```
git stash pop stash@{1}
```

The command applies the changes and schedules the stash for deletion.

Note: It is also possible to pop or apply stash changes to a new branch in Git. This keeps the working tree clean and avoids any conflicts between the stashed code and the code in the working branch.

Conclusion

This tutorial explained two methods for restoring a Git stash - popping and deleting the stash from the reference or applying the stash and keeping it for future use. Use Git stashes to set aside unfinished work and return to it later without creating a mess in the repository. We suggest also checking out our guide on how to stash untracked files in Git.

For more Git tutorials, read our article about the basics of Git tags, or see how to checkout a Git submodule.

Bosko Marijan

Having worked as an educator and content writer, combined with his lifelong passion for all things high-tech, Bosko strives to simplify intricate concepts and make them user-friendly. That has led him to technical writing at PhoenixNAP, where he continues his mission of spreading knowledge.

Next you should read

DevOps and Development

How to Git Stash Specific Files

September 13, 2022

This tutorial shows how to stash a specific file using Git. Stashing is a great way of putting aside unfinished changes that aren't yet ready to be committed.

READ MORE

DevOps and Development

How To Pull The Latest Git Submodule

August 17, 2022

With submodules, other Git sources can be used in a project copying the code. This tutorial shows how to pull the latest Git submodule.

READ MORE

Backup and Recovery, DevOps and

Development

How to Restore a Git Repository

August 1, 2022

Deleted or overwritten a repository? Don't worry just yet, this guide covers some steps to try and restore the repository.

READ MORE

DevOps and Development

Add, Update, and Remove Git Submodule

September 22, 2022

This tutorial shows how to perform basic submodule management operations - adding, updating, and removing them.

READ MORE