What Is Git Bash; Working with Git Bash Commands

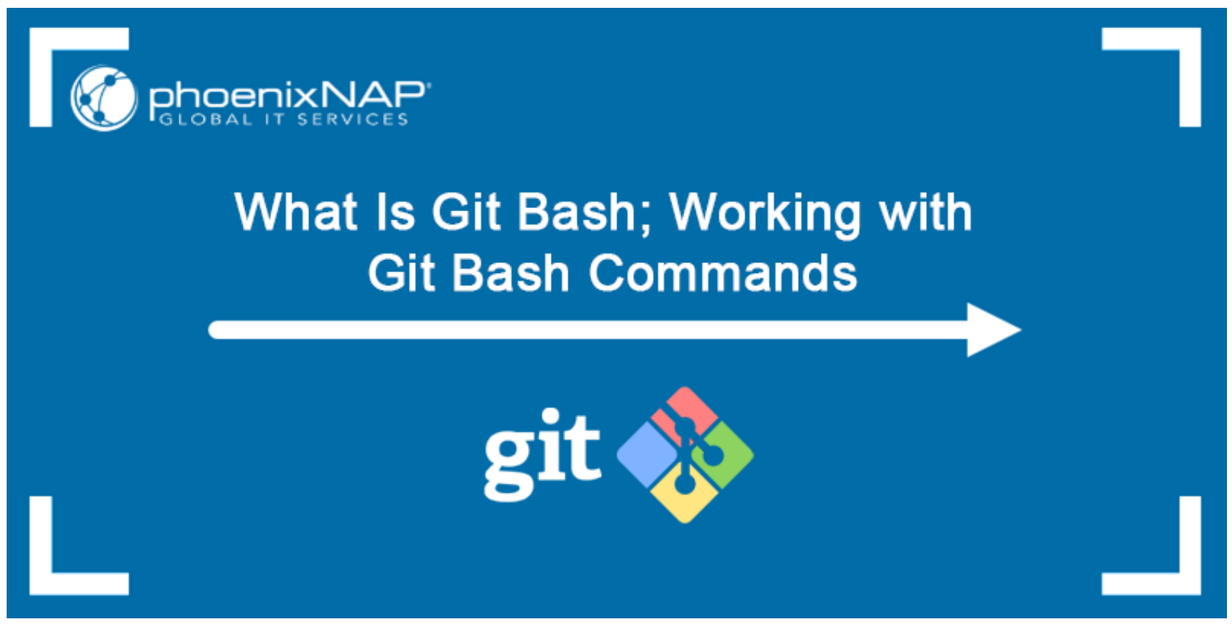
September 8, 2021 BASH COMMANDS GIT

Contents

Introduction

Git is a version control system that lets developers track source code changes during software development. Git Bash is an application for Microsoft Windows, allowing developers to use Git in a command-line interface.

In this article, you will learn what Git Bash is and how to use it.



Prerequisites

- A system running Windows
- A network connection

What Is Git Bash

Git Bash is a Microsoft Windows application with a Git command-line shell experience and utilities, such as Secure Shell Protocol (SSH), Secure Copy Protocol (SCP), CAT (concatenate command), etc. Bash is an acronym for Bourne Again Shell, which is a GNU Project shell.

A shell is used to interface with an operating system by executing commands, and Bash is the default shell used on Linux and macOS.

Note: For a detailed overview of basic Git functionalities, read our Beginner's guide for using Git.

What Is Git Bash Used For

Git Bash emulates a bash environment on Windows, allowing users to use the Bash shell and most of the standard Unix commands on a Windows OS. Users can interact with a repository and Git elements by running commands in Git Bash.

How to Install and Set Up Git Bash (A Step-by-Step Guide)

Follow the steps below to install and set up Git Bash.

Step 1: Download and Install Git Bash

First, you must install Git on your machine. Follow the steps outlined in the tutorial to download and install Git on Windows systems.

Step 2: Launch Git Bash

After installing Git, search for *Git Bash* in the start menu. Press **Enter** to launch the app.

Alternatively, to run Git Bash from a location where you want to keep your project files, press the right mouse button and click the **Git Bash Here** option from the dropdown menu.

Step 3: Connect to a Repository

The syntax to configure your local Git installation to use your GitHub credentials is:

```
git config --global user.name "github_username"
git config --global user.email "email_address"
```

Replace **`github_username`** and **`email_address`** with your GitHub credentials.

If you already have a repository on GitHub, you can clone the repository to your local machine. Use the following syntax:

```
git clone [repository_url]
```

Find your **`repository_url`** in the Code section of your GitHub page:

Use it with the **clone** command:

Our Beginner's guide on using Git offers more information on creating a new local repository or repository on GitHub.

How to Use Git Bash

The following section explains the basic functionalities of Git Bash and the available commands.

- **Initialize**

The **git init** command creates an empty .git repository or reinitializes an existing one.

Note: Running **git init** when you already have an existing repository doesn't overwrite your existing files but **adds new templates**.

For example:

- **Navigate**

The **cd** command allows you to change the directory in which Git Bash operates. The syntax is:

```
cd [directory-name]
```

For example:

If you want to see all the files and subdirectories in the current directory, run:

```
ls
```

For example:

- **Status**

The **git status** command lists all the modified files ready to be added to the local repository.

In this example, the **git status** command shows the modified and new files that haven't been added to the index and that Git doesn't track yet. While Git is aware of the files, you have to let Git know you want to track changes to these files.

- **Add**

The **`git add`** command updates the index with the content in the working tree and prepares the content in the staging area for commit.

You can add both files and directories to the index. The syntax is:

```
git add [directory] | [file]
```

For example:

Here, we see that *examplefile.md* has been added to the index and is ready for commit.

If you have several files ready to be committed, you can use the **`git add -A`** command to add all the files from the directory that haven't been added to the index yet.

- **Commit**

After adding files to the staging environment, Git can package the files into a commit via the **`git commit`** command. The **`git commit`** command instructs Git to store that file version. Git only commits the changes made in the repository.

The syntax is:

```
git commit -m "Commit notes"
```

For example:

In this example, Git commits the *examplefile.md*, the only change in the repository.

- **Pull**

The **`git pull`** command fetches changes from a remote repository to your local repository.

Before running the **`git pull`** command, make sure that your central repo is set as origin. Run:

```
git remote add origin [your-repository-link]
```

After setting your origin repository, run:

```
git pull origin master
```

For example:

In this example, Git states that everything is already up to date, and there are no new files to add to the local repository.

- **Push**

The **`git push`** command is the opposite of the **`git pull`** command. This command sends files from the local repository to the remote repository.

Run the following command:

```
git push origin master
```

For example:

- **Branch**

Branching in Git allows every team member to work on a feature they can later merge to the project main branch. The syntax for creating a branch in Git Bash is:

```
git branch [branch-name]
```

For example:

In this example, we create a branch named *new-feature*.

When you are working on several branches, you can easily switch between branches. The syntax is:

```
git checkout [branch]
```

Replace **[branch]** with the branch name you want to switch to.

For example:

- **Merge**

The **git merge** command allows you to merge two branches together.

Important: Ensure you are on the target (merge-receiving) branch when running the **git merge** command. Switch to the target branch using **git checkout**.

The syntax is:

```
git merge [branch-name]
```

For example:

In this example, we merged the *new-feature* branch into the *master* branch, adding a new file.

Conclusion

You now know what Git Bash is and how to use it. While GUI apps for Git may seem more attractive for Windows users, Git Bash allows you to learn how the raw Git methods work.

Using the command line tools instead of a GUI version is beneficial, especially when project collaboration requirements increase with the number of people working on the project.

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, SysAdmin

What Is Git?

July 28, 2021

This article helps you understand what git is, guides you through its features, use cases, and the way git works.

READ MORE

Backup and Recovery, DevOps and Development

How To Resolve Merge Conflicts in Git

June 16, 2021

When working with multiple people or on a big project, version control is a must-have. Joining information from multiple sources sometimes yields merge conflicts. Learn about how they happen and how to resolve merge conflicts.

READ MORE

DevOps and Development, SysAdmin

How to Update Git

May 25, 2021

This tutorial provides an overview of different ways you can update Git on Linux, Windows, and MacOS systems.

READ MORE

DevOps and Development, SysAdmin

How To Unstage Files on Git

September 15, 2020

Unstaging in Git means removing queued changes from the index. This guide covers several different ways to unstage changes.

READ MORE

RECENT POSTS

How to Install Samba in Ubuntu

YUM vs. APT: What's the Difference?

How To Install PHP On Ubuntu 20.04 or 22.04

How to Set Docker Environment Variables

What Is Data Storage? {Definition and Types of Data Storage}

CATEGORIES

SysAdmin
Virtualization
DevOps and Development
Security
Backup and Recovery
Bare Metal Servers
Web Servers
Networking
Databases

COLOCATION

Phoenix
Ashburn
Amsterdam
Atlanta
Belgrade
Singapore

PROMOTIONS

SERVERS

Dedicated Servers
Database Servers
Virtualization Servers
High Performance Computing (HPC) Servers
Dedicated Streaming Servers
Dedicated Game Servers
Dedicated Storage Servers
SQL Server Hosting
Dedicated Servers in Amsterdam
Cloud Servers in Europe
Big Memory Infrastructure

BUY NOW

CLOUD SERVICES

Data Security Cloud
VPDC
Managed Private Cloud
Object Storage

SERVERS

Disaster Recovery
Web Hosting Reseller
SaaS Hosting

INDUSTRIES

Web Hosting Providers
Legal
MSPs & VARs
Media Hosting
Online Gaming
SaaS Hosting Solutions
Ecommerce Hosting Solutions

COMPLIANCE

HIPAA Ready Hosting
PCI Compliant Hosting

NEEDS

Disaster Recovery Solutions
High Availability Solutions
Cloud Evaluation

COMPANY

About Us
GitHub
Blog
RFP Template
Careers

CONNECT

Events
Press
Contact Us

PhoenixNAP Home Blog Resources Glossary GitHub RFP Template

Live Chat Get a Quote Support | 1-855-330-1509 Sales | 1-877-588-5918

Contact Us Legal Privacy Policy Terms of Use DMCA GDPR Sitemap