When you clone a Git repository or create new features through branches, you need know how upstream branches work and how to set them up.

This article gives an overview of how to set up a Git upstream branch, how to change it and how to have an overview of which Git branch is tracking which upstream branch.

Prerequisites

- Git installed and configured
- A cloned Git repository or your own Git project set up locally

---

Note: To install Git, check out our tutorials:

- How to Install Git on CentOS 7 With Yum or Latest Repository
- How to Install Git on CentOS 8
- How to Install Git on Ubuntu 18.04 / 20.04
- How to Install and Use Git on Windows
- How to Install and Get Started with Git on Mac

What is a Git Upstream Branch?

Using a river analogy to illustrate the flow of data, upstream is sending your data back to where the river stream is coming from. When you send something upstream, you are sending it back to the original authors of the repository.

With `git set upstream`, you can choose where your current local branch will flow. It allows you to change the default remote branch.

How to Set Upstream Branch in Git

There are two ways to set an upstream branch in Git:

- Using `git push`, which is the fastest method if you need to set a single upstream branch.
- Using a short alias command. This method makes sense if you often change the flow of your current branch.

Method 1: Set Upstream Branch Using Git Push

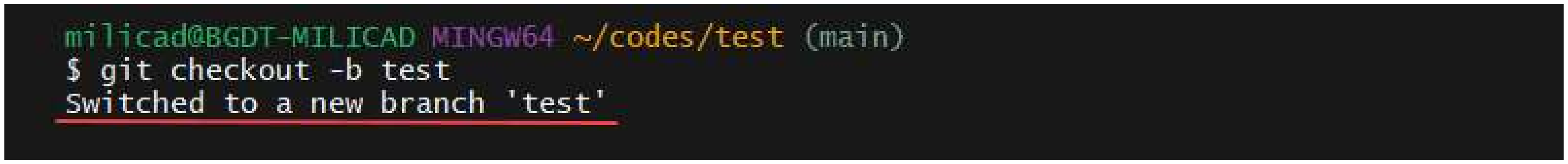
Using `git push` to set an upstream branch is the most straightforward way to set upstream branches in Git.

Note: Forgot how to clone a repository? Freshen up your memory with our Git Commands Cheat Sheet.

1. Create a new branch and give it a name. We named ours *test*. Switch to it using the `checkout` command with the `-b` option:

```
git checkout -b <branch name>
```

A switch branch confirmation appears:



```
milicad@BGDT-MILICAD MINGW64 ~/codes/test (main)
$ git checkout -b test
Switched to a new branch 'test'
```

Note: From this point on, the active branch is listed as *(<branch name>)* instead of *(main)*. In our case, it's *(test)*.

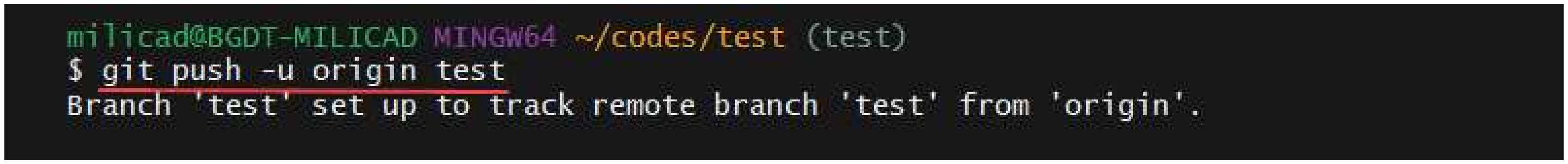
2. Set upstream branch using the **`git push`** command with the **`-u`** extension or use the longer version **`--set-upstream`**. Replace **`<branch name>`** with your branch name.

   ```
   git push -u origin <branch name>
   ```

   Alternatively:

   ```
   git push --set-upstream origin <branch name>
   ```

   You get confirmation that your branch has been set up to track a remote branch:

   ```
   milicad@BGDT-MILICAD MINGW64 ~/codes/test (test)
   $ git push -u origin test
   Branch 'test' set up to track remote branch 'test' from 'origin'.
   ```

   The test branch now has a set upstream branch.

Method 2: Set Upstream Branch Using Alias

Instead of going through these commands every time you create a new branch, set up a short alias command. You can modify your existing Git commands or create a bash command.

1. Configure the global alias command through **`git config`** with the **`--global`** command:

   ```
   git config --global alias.<alias name> “push -u origin HEAD”
   ```

   Or create a bash alias command using **`alias`**:

   ```
   alias <alias name> ='git push -u origin HEAD'
   ```

Note: Pushing to `HEAD` will push to a remote branch with the same name as your current branch.

---

2. Run your global alias by typing:

   ```
   git <alias name>
   ```

   Or your bash alias by typing its name:

   ```
   <alias name>
   ```

How to Change Upstream Branch in Git

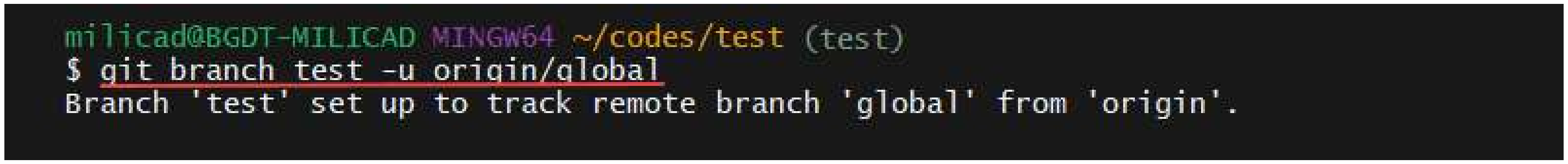
Track a different upstream branch than the one you just set up by running:

```
git branch  -u <remote/branch name>
```

For example:

```
git branch  -u <origin/global>
```

The terminal prints out a confirmation message:

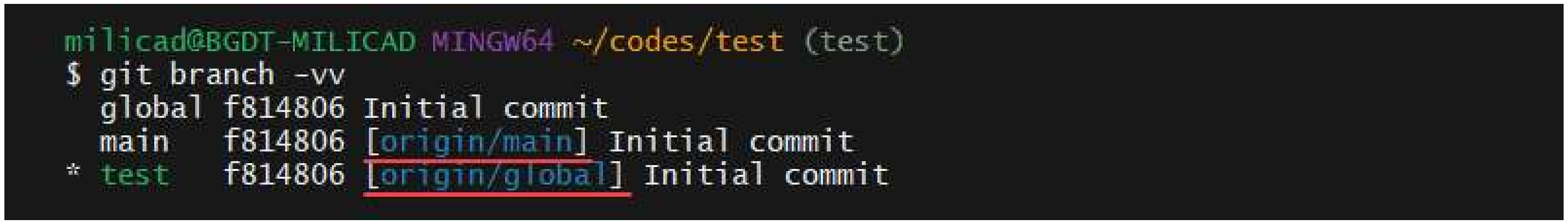


```
milicad@BGDT-MILICAD MINGW64 ~/codes/test (test)
$ git branch test -u origin/global
Branch 'test' set up to track remote branch 'global' from 'origin'.
```

How to Check Which Git Branches Are Tracking Which Upstream Branch

List all your branches and branch tracking by running `git branch` with the `-vv` option:

```
git branch -vv
```



```
milicad@BGDT-MILICAD MINGW64 ~/codes/test (test)
$ git branch -vv
  global f814806 Initial commit
  main   f814806 [origin/main] Initial commit
* test   f814806 [origin/global] Initial commit
```

The *main* branch has a tracking branch of *[origin/main]*. The *test* branch has a tracking branch of *[origin/global]*. The *global* branch has no tracking branches, and therefore no upstream branch.

Note: The current active branch is denoted by the asterisk (*) sign.
