Git Checkout

This page is an examination of the `git checkout` command. It will cover usage examples and edge cases. In Git terms, a "checkout" is the act of switching between different versions of a target entity. The `git checkout` command operates upon three distinct entities: files, commits, and branches. In addition to the definition of "checkout" the phrase "checking out" is commonly used to imply the act of executing the `git checkout` command. In the Undoing Changes topic, we saw how `git checkout` can be used to view old commits. The focus for the majority of this document will be checkout operations on branches.

Checking out branches is similar to checking out old commits and files in that the working directory is updated to match the selected branch/revision; however, new changes are saved in the project history—that is, it's not a read-only operation.

Checking out branches

The `git checkout` command lets you navigate between the branches created by `git branch`. Checking out a branch updates the files in the working directory to match the version stored in that branch, and it tells Git to record all new commits on that branch. Think of it as a way to select which line of development you're working on.

Having a dedicated branch for each new feature is a dramatic shift from a traditional SVN workflow. It makes it ridiculously easy to try new experiments without the fear of destroying existing functionality, and it makes it possible to work on many unrelated features at the same time. In addition, branches also facilitate several collaborative workflows.

The `git checkout` command may occasionally be confused with `git clone`. The difference between the two commands is that clone works to fetch code from a remote repository, alternatively checkout works to switch between versions of code already on the local system.

Usage: Existing branches

Assuming the repo you're working in contains pre-existing branches, you can switch between these

```
$> git branch
main
another_branch
feature_inprogress_branch
$> git checkout feature_inprogress_branch
```

The above example demonstrates how to view a list of available branches by executing the `git branch` command, and switch to a specified branch, in this case, the `feature_inprogress_branch`.

New Branches

`Git checkout` works hand-in-hand with `git branch`. The `git branch` command can be used to create a new branch. When you want to start a new feature, you create a new branch off `main` using `git branch new_branch`. Once created you can then use `git checkout new_branch` to switch to that branch. Additionally, The `git checkout` command accepts a `-b` argument that acts as a convenience method which will create the new branch and immediately switch to it. You can work on multiple features in a single repository by switching between them with `git checkout`.

The above example simultaneously creates and checks out `<new-branch>`. The `-b` option is a convenience flag that tells Git to run `git branch` before running `git checkout <new-branch>`.

```
git checkout -b <new-branch> <existing-branch
```

By default `git checkout -b` will base the `new-branch` off the current `HEAD`. An optional additional branch parameter can be passed to `git checkout`. In the above example, `<existing-branch>` is passed which then bases `new-branch` off of `existing-branch` instead of the current `HEAD`.

Switching Branches

Switching branches is a straightforward operation. Executing the following will point `HEAD` to the tip of `<branchname>`.

```
git checkout <branchname>
```

Git tracks a history of checkout operations in the reflog. You can execute `git reflog` to view the history.

Branch

When collaborating with a team it is common to utilize remote repositories. These repositories may be hosted and shared or they may be another colleague's local copy. Each remote repository will contain its own set of branches. In order to checkout a remote branch you have to first fetch the contents of the branch.

```
git fetch --all
```

In modern versions of Git, you can then checkout the remote branch like a local branch.

```
git checkout <remotebranch>
```

Older versions of Git require the creation of a new branch based on the `remote` .

```
git checkout -b <remotebranch> origin/<remote
```

Additionally you can checkout a new local branch and reset it to the remote branches last commit.

```
git checkout -b <branchname>
```

Detached HEADS

Now that we've seen the three main uses of `git checkout` on branches, it's important to discuss the "`detached HEAD`" state. Remember that the `HEAD` is Git's way of referring to the current snapshot. Internally, the `git checkout` command simply updates the `HEAD` to point to either the specified branch or commit. When it points to a branch, Git doesn't complain, but when you check out a commit, it switches into a "`detached HEAD`" state.

This is a warning telling you that everything you're doing is "detached" from the rest of your project's development. If you were to start developing a feature while in a detached `HEAD` state, there would be no branch allowing you to get back to it. When you inevitably check out another branch (e.g., to merge your feature in), there would be no way to reference your feature:

The point is, your development should always take place on a branch—never on a detached `HEAD`. This makes sure you always have a reference to your new commits. However, if you're just looking at an old commit, it doesn't really matter if you're in a detached `HEAD` state or not.

Summary

This page focused on usage of the `git checkout` command when changing branches. In summation, `git checkout`, when used on branches, alters the target of the `HEAD` ref. It can be used to create branches, switch branches, and checkout remote branches. The `git checkout` command is an essential tool for standard Git operation. It is a counterpart to `git merge`. The `git checkout` and `git merge` commands are critical tools to enabling `git workflows`.

Next up:

Powered By Bitbucket

Recommend

Want future articles?

Site hosted by ATLASSIA

Enter Your Email For Git News