WORKING WITH BRANCHES IN GIT

presented by TOWER — the best Git client for Mac and Windows

Creating a New Branch

This is where the fun starts. Simply type this command to create a new branch, so that you can work on a feature or fix a bug in isolation:

```
$ git branch <new-branch-name>
```

Keep in mind that **you can only create new branches in your local repository** (you will see them remotely when published).

By default, the new branch will be based on your currently checked out revision. If you'd like to start at a specific revision, simply add that revision's hash at the end of the command, like so:

```
$ git branch <new-branch-name> 2b504bee
```

Tip: Make sure you give meaningful names to your branches - something like "feature-being-worked-on" could be a good starting point.

Switching Branches

To switch to a different branch and make it active (**setting it as the HEAD branch**), you can rely on either "switch" or "checkout". We like "switch", as it is self-explanatory and "checkout" can be used for other tasks, but both are valid.

```
$ git switch <other-branch>
```

OR

```
$ git checkout <other-branch>
```

Renaming Branches

This command assumes you've **already checked out the branch** you'd like to rename (your local HEAD branch):

```
$ git branch -m <new-name>
```

If you'd like to rename a **different branch**, you should first insert the current name, like in this example:

```
$ git branch -m <current-name> <new-name>
```

Publishing Branches

While you can't create new branches on remote repositories, you can most definitely publish an existing local branch by typing this command:

```
$ git push -u origin <local-branch>
```

Since **you're not allowed to rename remote branches**, you can delete the old one and then push a new one by typing these two commands:

```
$ git push origin --delete <old-name>
```

```
$ git push -u origin <new-name>
```

Tracking Branches

This is an important command to create **relationships between local and remote branches** - since initially, they don't have any!

The most common example is having a local branch track a remote one, so that you can simply type "git push" or "git pull" without additional parameters to keep everything in sync. This can be quickly achieved by typing the command below:

```
$ git branch --track <new-branch> origin/<base-branch>
```

You can also use the "git checkout" command to achieve this. If you want to name the local branch after the remote one, you only have to specify the remote branch's name:

```
$ git checkout --track origin/<base-branch>
```

WORKING WITH BRANCHES IN GIT

presented by TOWER — the best Git client for Mac and Windows

Comparing Branches

This is a very useful command when you need to decide if you should integrate or delete a branch, as it presents you the commits that were created exclusively in there.

```
$ git log <main>..<feature-branch>
```

Tip: You can also use this command to compare local and remote states, like in the example below:

```
$ git log <origin/main>..<main>
```

Merging Branches

When it is time to **integrate the new changes into your current HEAD branch**, it is time to merge!

Merging consists of two steps: you will need to switch to the branch that will receive the new changes first, and then type the "git merge" command:

```
$ git switch <main>
```

```
$ git merge <feature-branch>
```

Rebasing Branches

Rebasing is **an alternative** to merging - both achieve the same goal, but the Rebase option re-writes the project history, creating a straight line. As a result, you get a linear history, which may be preferred by some teams.

Rebasing consists of two steps: you will need to switch to the feature branch first, and then type the "git rebase" command:

```
$ git switch <feature-branch>
```

```
$ git rebase <main>
```

Deleting Branches

When a **local branch** is no longer needed, you can delete it by typing the command below:

```
$ git branch -d <branch-name>
```

Tip: If that branch contains **unmerged changes**, you might also need the "-f" option to **force** the deletion - if so, **proceed with caution**!

To delete a **remote branch**, keep in mind that the command is totally different:

```
$ git push origin --delete <branch-name>
```

To avoid a messy repository, make sure you periodically look for obsolete branches and delete them.

30-day free trial available at
www.git-tower.com

TOWER
The best Git Client for Mac & Windows