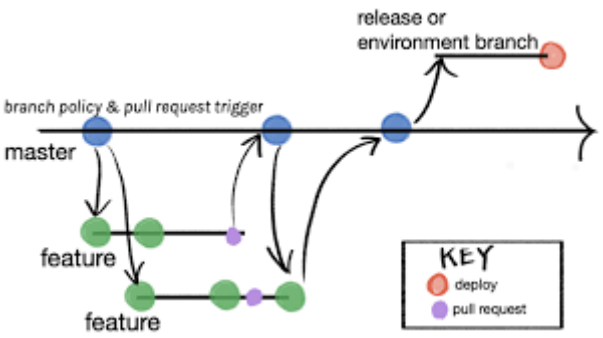
Git Basic Branching



The document is an in-depth review of the git branch command and a discussion of the overall Git branching model. Branching is a feature available in most modern version control systems. Branching in other VCS's can be an expensive operation in both time and disk space.

Git branches are effectively a pointer to a snapshot of your changes. When you want to add a new feature or fix a bug - no matter how big or small your feature needs to have it's own branch. This makes it harder for unstable code to get merged into the main code base, and it gives you the chance to clean up your future history before merging it into the main branch.

The git branch command lets you create, list, rename, and delete branches. It doesn't let you switch between branches or put a forked history back together again. For this reason, git branch is tightly integrated with the git checkout and git merge commands.

Common Options

```
git branch
```

List all of the branches in your repository. This is synonymous with `git branch --list`.

```
git branch <branch>
```

Create a new branch called <branch>. This does not check out the new branch.

```
git branch -d <branch>
```

Delete the specified branch. This is a safe operation in that git prevents you from deleting the branch if it has unmerged changes.

```
git branch -m <branch>
```

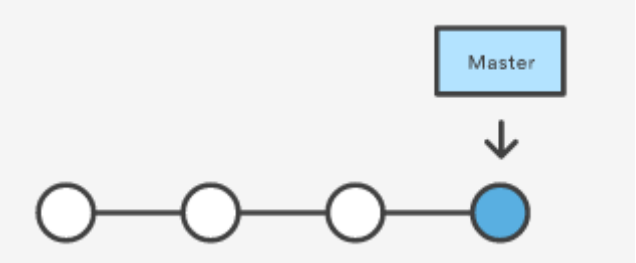
Rename the current branch to <branch>

```
git branch -a
```

List all remote branches.

Creating Branches

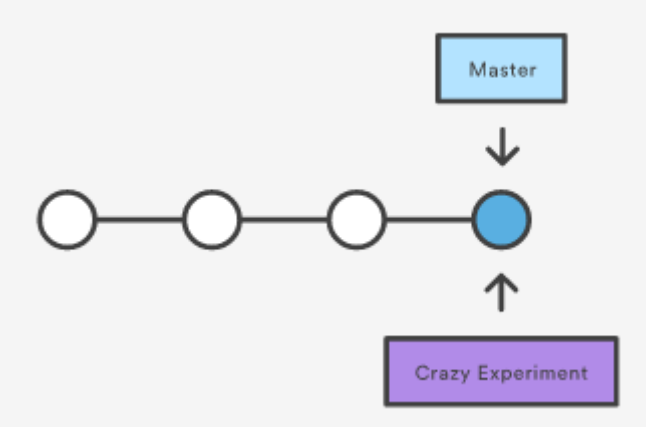
It's important to understand that branches are just pointers to commits. When you create a branch, all Git needs to do is create a new pointer, it doesn't change the repository in any other way. If you start with a repository that looks like this:



Then, you create a branch using the following command:

```
git branch crazy-experiment
```

The repository history remains unchanged. All you get is a new pointer to the current commit:



Note that this only creates the new branch. To start adding commits to it, you need to select it with `git checkout`, and then use the standard `git add` and `git commit` commands.

So far these examples have demonstrated local branch operations. The git branch also works on remote branches. In order to operate on remote branches, a remote repo must first be configured and added to the local repo config.

```
git remote add new-remote-repo https://<repo-link-here>
```

The above command adds remote repo to local repo config.

```
git push <new-remote-repo> crazy-experiment~
```

The above command pushes the crazy-experiment branch to new-remote-repo