Checkout

One of the most powerful functionalities provided by git is the checkout functionality.

This command can be used to do a number of different things including:

- Creating a new branch
- Switching to a different branch
- Updating the working tree
- Moving to a different commit on the current branch
- Creating a new branch at a previous commit point

I'll go over each of these functionalities and provide a use case for them as well.

Creating a new branch

This can be useful if there are multiple people working on a project where there are subgroups in the project that are working on different parameter settings or different feature extraction methods. Rather than create two different directories where the files live, you can create two separate branches and use the checkout command to switch between them.

To create a new branch use the following syntax

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

Switching to a different branch

Now that this new branch is created though, how do you quickly switch back and forth between them? The answer is to essentially leave out the `-b` flag.

```
git checkout <target branch name>
```

Updating the working tree

The most common case where I use this functionality is to get back a file that I accidentally

deleted from my file system. Running `git checkout <file path>` will bring the latest version of that file into your repo, even if that file has been deleted by mistake.

Moving to a different commit on the current branch

If there's a previous commit that you would like to roll back to, for example, if you need to examine the previous way that some system was running, you can checkout an individual commit from the past. To do this, you will need to obtain the shortened hexidecimal commit hash. One way to obtain this is using the `git log` command. I recommend using the `git hist` alias discussed elsewhere in this chapter.

Another way is to look at the repository in the git web interface on the server where your repository is hosted. There will typically be a place to view the commit history on these web interfaces. Then you can copy the commit code and paste it into the appropriate spot in your command line.

For example,

```
git checkout 79b47a4
```

Where `79b47a4` is an example commit code.

Creating a new branch at the previous commit point

However, the previous command will put your local repository into a detached head state. You can do most git things but you are not currently on a branch terminal so some operations like merging will not work. The solution is to create a new branch when you rewind to a previous commit. This essentially combines the methods from two of the previous sections. Simply run

```
git checkout -b <new branch name> <commit hash>
```

Revision #2
Created Wed, Sep 19, 2018 1:56 AM by kenneth
Updated Wed, Sep 19, 2018 2:24 AM by kenneth