Feature Branches

Case: Mix of Live and Dead Code on the Same Branch

Case: Say you want to make a new addition to the code, so you create a fork and start coding right away.

You work for days and days on a new addition and fix some bugs along the way, only to find that your proposed addition is not going to work properly.

But half of the work you have done on the branch was bug fixed, which need to make their way back upstream.

Now you'll have to either manually roll back the additions you have made or do a *git checkout* or *git reset* and fix the same bugs again manually.

Avoiding the Same Fate

To avoid this situation, it is recommended to separate each additional feature or fix for each particular bug out to its own branch.

You can create and switch to a new branch with the following commands:

```
git branch new-branch-name
git checkout new-branch-name
```

OR in a single line:

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

This way, you can make a PR for any modifications separately and once they are ready, instead of needing to manually roll back or redo changes.

Don't forget to switch back to the default/base branch when you are done.

NOTE: In general, we tend to name branches after the key of their associated JIRA tickets. For example: **NDS-101**

About Branch Tracking

It is important to note that whichever branch you are currently on when executing **git branch** or **git checkout -b** will be "tracking branch" of your branch.

This means that your branch stemmed from the tracking branch, and any changes that were present on that branch are now present on yours.

This means that collaborating / integrating with another developer is as easy as creating a new branch, and pulling one or more sets of changes into it for integration testing.

If things go awry, you can always choose to blow away the branch altogether and try again.