Suite Script Developer Notes.

What is branching?

Branching essentially allows us to maintain multiple copies of our file structure so that different contributors can work on different tasks within the project. They could even be working on the exact same files without impacting each other this way. However, the goal of a branch is not to keep it separate and isolated forever. What we want to happen is for those contributors to complete their tasks in their own branch, then \*merge\* their work into the main codebase (i.e. the `master` branch).We don't want to use branches to \*permanently\* segregate our files, but rather temporarily, while we are actively working on a feature or fix. Then once the task is completed, we want to incorporate everything back into the main codebase on the `master` branch. In that way, the `master` branch is sort of "sacred"; it is the Production-ready code we \*never\* want to introduce broken code onto `master`. We write our code in a branch, we test it thoroughly, and then we merge to `master`, usually via a Pull Request.(More on pull requests)

[https://help.github.com/articles/about-pull-requests/](https://help.github.com/articles/about-pull-requests/%20)   
 <https://www.atlassian.com/git/tutorials/making-a-pull-request>

Question : What do you call your Branches?

Great question; so in general I classify the tasks I'm working on into three different categories:

1. feature - any brand new development or improvement on an existing feature

2. bugfix - any break fix

3. chore - any internal work like building tools or optimization or refactoring

Then, my branch names are basically: `type-of-task/name-of-task`

[10:36]

There are also `release` branches, where I name them `release/versionNumber`, as I"ve done in this repo:   
<https://gitlab.com/stoicsoftware/netsuite-sku-analytics/branches>

So for example, if I'm building a new sales order approval feature, then I might name the branch `feature/so-approval`or fixing an error with a search filter, maybe `bugfix/search-filter-error` Many git clients will use the `/` in the name and use it to group your branches together and visualize them as folders, so it's a nice visualization of your branch structure And one final caveat. When I am working out of some kind of issue tracking system that provides unique IDs for tasks, then i'll use the unique ID in the name of the branch insted

so for instance, we used to work out of JIRA, which would have issue IDs like "RR-2148". So if I was working on RR-2148, a bug, then I would name the branch `bugfix/RR-2148`; that way I can very easily connect the code for a specific task with its branch “`master` has gotten ahead of your test branch, so you want to incorporate the latest changes on `master` into `test `You can do that however you wish; there are a few methods in git for updating feature branches `merge` is one of them and is the one I prefer. what you want to do before you merge a feature branch up to `master`is update the feature branch with `master`'s code. So what it looks like is Develop on feature branch > merge `master` into feature branch > (clean up any conflicts/oddities) > Create Pull Request > Merge Pull Request once approved

GitHub Links:

<https://www.atlassian.com/git/tutorials/using-branches>

<https://www.atlassian.com/git/tutorials/git-merge>

Lastly, this is a nice visualization of various methods for multiple collaborators to utilize branches: <https://www.atlassian.com/git/tutorials/comparing-workflows>  
  
I tend to prefer the "Feature Branch Workflow" when I am working on a project individually, and the "Gitflow Workflow" when I am working in a team setting