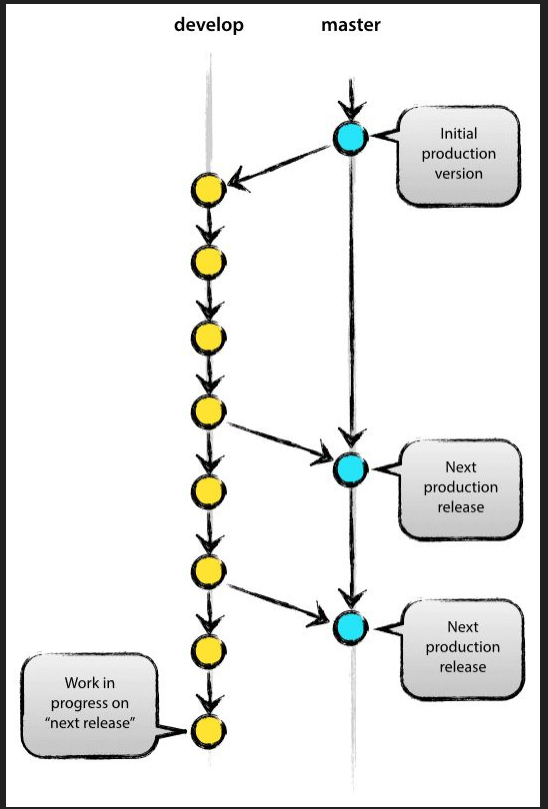
Git Branching

What is Branching?

Branching means you diverge from the main line of development and continue to do work without messing with the main line.

It is very useful when working collaboratively with others - each person can work on their own separate branch, and when ready, merge their work in with the main branch.



Creating a Branch

$ git branch <new branch name>

Important: This does not switch you to the new branch.

To see this new branch (along with all other branches), run

$ git branch

You will see a list of all branches with a star next to the branch you are currently on.

Changing Branches

To switch to another branch, use

$ git checkout <branch name>

When you switch branches, git changes the files to reflect the last commit of that branch.

Also, in most cases, you can’t switch branches if you have uncommitted changes!

Recommended Workflow for Data Question 2

1. Clone you group's repository from GitHub.
2. Create an individual branch for yourself.

   $ git branch <branch name>
3. **Important:** Make sure that you are on your branch each time before you start working.

   $ git branch

   $ git checkout <branch name>
4. When done working, commit your changes locally and then push to Github.

   $ git push origin **<branch name>**

---

5. **Later:** Periodically, merge your work with the master branch on Github by submitting a pull request (we'll talk about those later). Once merged, pull the latest master branch to your local machine.

   $ git pull origin master

Merging Branches

Using the command

$ git merge <branch_name>

will attempt to merge the contents of branch_name with the branch you are currently on.

So if you want to merge other_branch with master, you need to run

$ git checkout master

$ git merge other_branch

Merge Conflicts

Git will do its best to merge your changes, but sometimes it runs into conflicts it can’t resolve. For example, if the same part of a file was changed differently in two different branches.

In this case, you will have a merge conflict, which you need to resolve.

Pull Requests

Demonstration!

Tracking Branches

Checking out a local branch from a remote-tracking branch automatically creates what is called a “tracking branch” (and the branch it tracks is called an “upstream branch”). Tracking branches are local branches that have a direct relationship to a remote branch. If you’re on a tracking branch and type git pull, Git automatically knows which server to fetch from and which branch to merge in.

Fetching

It’s important to note that when you do a fetch that brings down new remote-tracking branches, you don’t automatically have local, editable copies of them. In other words, in this case, you don’t have a new serverfix branch — you have only an origin/serverfix pointer that you can’t modify.

To merge this work into your current working branch, you can run git merge origin/serverfix. If you want your own serverfix branch that you can work on, you can base it off your remote-tracking branch:

Pulling

While the git fetch command will fetch all the changes on the server that you don’t have yet, it will not modify your working directory at all. It will simply get the data for you and let you merge it yourself. However, there is a command called git pull which is essentially a git fetch immediately followed by a git merge in most cases. If you have a tracking branch set up as demonstrated in the last section, either by explicitly setting it or by having it created for you by the clone or checkout commands, git pull will look up what server and branch your current branch is tracking, fetch from that server and then try to merge in that remote branch.

Generally it’s better to simply use the fetch and merge commands explicitly as the magic of git pull can often be confusing.