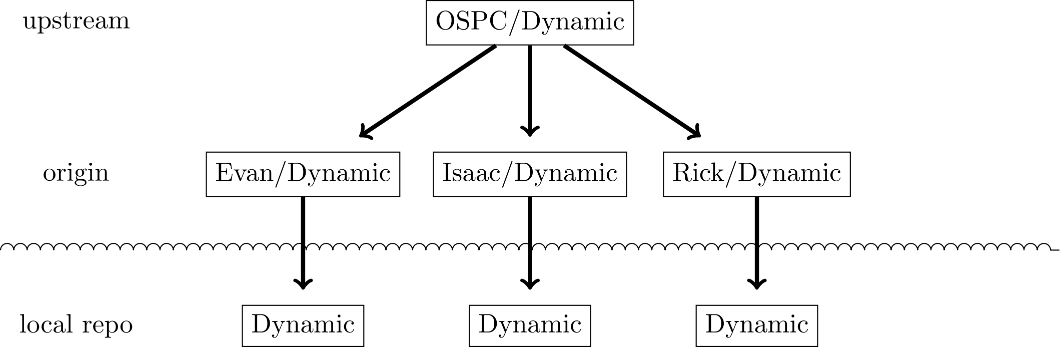
How to Git Properly

Isaac Swift and Evan Magnusson

Overview: OSPC has the main Dynamic repo. We don't want to make changes directly to that – things can go wrong, it's messy, harder to collaborate, etc. So, we make a copy (fork) of the OSPC repo in our own GitHub account (this is the remote repo). We then clone the copy onto our machine (this is the local repo). Then, on our computer, we can make branches (different projects or portions of the code you are working on) and commit the changes to our repo. Once we have finished one of our projects, then we can go to the GitHub website and submit a pull request to the OSPC repo and ask them to look at our changes and accept them into the main repo. As people do this, we can update our local and remote repos to keep track of the new updates.

- Getting started
  - Get Git
  - Get a GitHub user account
  - If you already cloned the OSPC dynamic repo, delete it from your computer.
  - Fork the repository of your choice
    - Go to the OpenSourcePolicyCenter/dynamic repository online.
    - Fork it (there is a button you click)
      - If you have multiple accounts or affiliations, it will ask you which one you want to fork to. Just do your basic account.
    - Now, go to your account, and there should be a *YourUsername*/dynamic repository
    - In your fork, go to the HTTPS clone URL area and copy the link
    - Go to your command line on your computer and clone the repo
      - git clone *TheLinkYouCopied*
      - It is important that you clone the fork, not the original repo from OSPC.
  - To make it easier to push and pull your work to the main OSPC repo, go back to that repository online and copy that link (that you would normally use to clone it)
    - Type the following in your command prompt (in the dynamic repo directory)
      - git remote add upstream *TheLinkYouCopied*
        - what this does is allow you to reference the OSPC repo later on without having to type the entire URL each time
  - If you type: git remote –v, in your command prompt you should now see 2 origin (the name of your remote repo) and 2 upstream (the name of the OSPC repo) lines

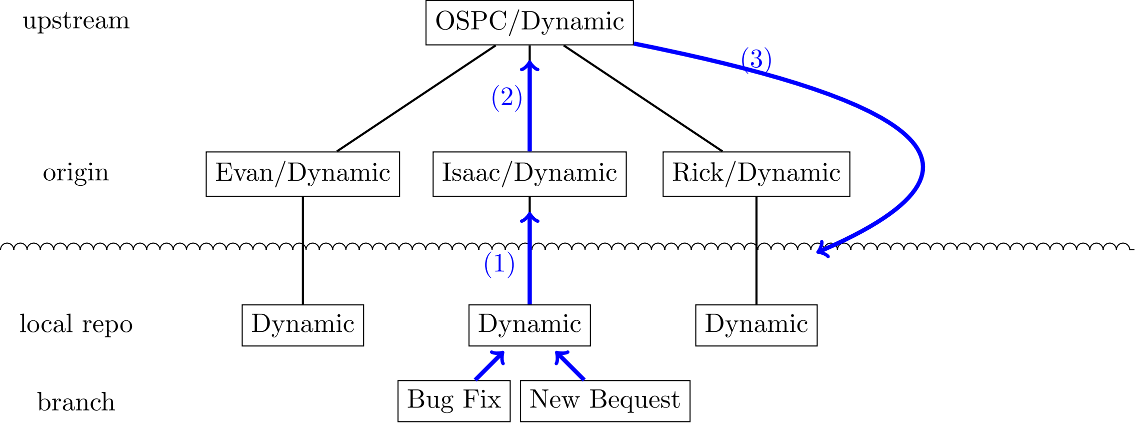
The following GIF is an animation of forking and cloning. Click to view animation.


upstream
OSPC/Dynamic
origin
Evan/Dynamic
Isaac/Dynamic
Rick/Dynamic
local repo
Dynamic
Dynamic
Dynamic

- Editing Code
    - Keep your local and remote repos up to date with the OSPC repo.
        - Go to the *dynamic* directory (from here on, always assume you're in the *dynamic* directory)
        - Download the current state of the OSPC dynamic repo
            - git fetch upstream
        - Swap to the master branch of your local repo
            - git checkout master
            - This is very important, especially if you have more of your own branches
        - Update your master branch locally with the current state of the OSPC repo
            - git merge upstream/master
            - This will not delete any changes you've made to the repo. You may have to merge files if you've edited the same ones that others have.
        - Update your remote repo to match your local repo
            - git push
    - Making new branches
        - git checkout –b *NewBranchName*
    - Editing
        - Swap to the desired branch where you want to work
            - git checkout BranchName
            - Note: *master* is already a branch name from the beginning
            - Make sure you're in *master* or *OtherBranch* when you want to be – you don't want to find out you committed changes from the wrong branch!
        - Edit files.
        - Commit the changes you've made
            - View what files have been changed
                - git status
            - Either add all these files, or choose which ones, to commit
                - git add *filename*
                    - add individual files or folders
                - git add –u
                    - add all files/folders within this directory
            - Commit the changes with a description of what you did
                - git commit -m '*description*'
        - Push you're local branch to your remote repo
            - git push origin *BranchName*
- Submit a pull request
    - Now, go online to your fork of the *dynamic* repo
    - There should now be a line that asks if you want to submit a pull request
        - You may not want to. Sometimes you want to wait to do a pull request if you aren't finished with what you wanted to do in your branch, but you wanted to push changes to the remote repo in order to back up your changes in case someone blows up your computer.

- Make sure you are up to date with the current OSPC repo
    - Fetch and merge your local and remote repos with the upstream/master, as instructed above
- Click the submit a pull request button
    - A box will come up where you can provide a much more accurate description of what you changed. Be thorough here.
    - Once you submit the pull request, you can go into the pull request and see all the insertions and deletions you made to each file.
        - This is a good way for you (and other team members) to make sure you didn't mess anything up before your changes get merged into the main OSPC repo.
        - You can put comments in on each insertion/deletion to explain exactly what you were doing and why you did it.
- Ask other team members to review and accept your changes.
    - Wait for feedback, and fix anything on your local repo that they say is wrong.
        - Commit these changes again, as above.
        - Until a pull request is accepted, any changes you commit from a branch that already has a pull request are updated on the pull request.
    - Eventually, they accept the changes.
    - You can accept the changes yourself, but you shouldn't do this usually.

The following GIF is an animation of fetching, pushing, and pull requesting. Click to view animation.



(1). add, commit, push
(2). pull request
(3). fetch, merge

- Tips
  - Merge often
    - Keep all of your branches (not just master) up to date with the OSPC repo by fetching and merging into each of them
      - Simply do what you did above, but checkout into a different branch
  - Commit often
    - Committing often, without submitting a pull request, even when you aren't finished is good
      - It prevents you from losing work if your computer crashes
      - It makes the reset command more effective (see below) because you can reset exactly where you want to
      - Don't submit a pull request, though, if you don't need to, because you don't want to have people accept something you aren't done with
  - Deleting branches once you finish them
    - Once you finish a branch you've been working on (meaning that you finished that problem/project you were doing and your pull request(s) for that branch have all been accepted), you should delete the branch.
      - This helps with organization/cleanliness
    - At the bottom of a merged/closed pull request online, there is a delete branch button (deletes the branch remotely)
      - Or, you can delete branches from the Branches page on Github.
      - If you do this on accident, there is a Restore branch button as well.
    - To delete the local branch:
      - git branch -d *BranchName*
      - using a capital D will force the deletion of the branch, even if it isn't merged with your other branches (meaning you lose all the work you made on the branch)
- Commands
  - Resetting commits
    - git reset *NameOfCommit*
    - Resets your local repo to the name of some commit you did in the past. Good in case you accidently messed something up, and want to go back.
    - Use sparingly, you can lose all your work this way.
  - List all branches
    - git branch
    - The * indicates your current branch
  - List the last commit for each branch
    - git branch -v
  - List branches that you have or have not yet merged into with your current branch
    - git branch --merged
    - git branch --no-merged

- Uses: make sure *master* is up to date with upstream. Go into *master,* and then do the command to see which of your other branches are not merged with master. Then, go and merge those branches with master.
- Also, you cannot delete a branch unless it is merged, so this is a good way to make sure it is before you try to delete it.