Git -- Version control

# **Scenario #1**

Let's walk through the git commands that go along with each step in the scenario you just observed in the video above.

#### **STEP 1: You have a local version of this repository on your laptop, and to get the latest stable version, you pull from the develop branch.**

##### ***Switch to the develop branch***

*git checkout develop*

##### ***Pull latest changes in the develop branch***

*git pull*

#### **STEP 2: When you start working on this demographic feature, you create a new branch for this called demographic, and start working on your code in this branch.**

##### ***Create and switch to new branch called demographic from develop branch***

*git checkout -b demographic*

##### ***Work on this new feature and commit as you go***

*git commit -m 'added gender recommendations'*

*git commit -m 'added location specific recommendations'*

*...*

#### **STEP 3: However, in the middle of your work, you need to work on another feature. So you commit your changes on this demographic branch, and switch back to the develop branch.**

##### ***Commit changes before switching***

*git commit -m 'refactored demographic gender and location recommendations '*

##### ***Switch to the develop branch***

*git checkout develop*

#### **STEP 4: From this stable develop branch, you create another branch for a new feature called friend\_groups.**

##### ***Create and switch to new branch called friend\_groups from develop branch***

*git checkout -b friend\_groups*

#### **STEP 5: After you finish your work on the friend\_groups branch, you commit your changes, switch back to the development branch, merge it back to the develop branch, and push this to the remote repository’s develop branch.**

##### ***Commit changes before switching***

*git commit -m 'finalized friend\_groups recommendations '*

##### ***Switch to the develop branch***

*git checkout develop*

##### ***Merge friend\_groups branch to develop***

*git merge --no-ff friends\_groups*

##### ***Push to remote repository***

*git push origin develop*

#### **STEP 6: Now, you can switch back to the demographic branch to continue your progress on that feature.**

##### ***Switch to the demographic branch***

*git checkout demographic*

# **Scenario #2**

Let's walk through the git commands that go along with each step in the scenario you just observed in the video above.

#### **Step 1: You check your commit history, seeing messages of the changes you made and how well it performed.**

##### ***View log history***

*git log*

#### **Step 2: The model at this commit seemed to score the highest, so you decide to take a look.**

##### ***Checkout a commit***

*git checkout bc90f2cbc9dc4e802b46e7a153aa106dc9a88560*

After inspecting your code, you realize what modifications made this perform well, and use those for your model.

#### **Step 3: Now, you’re pretty confident merging this back into the development branch, and pushing the updated recommendation engine.**

##### ***Switch to develop branch***

*git checkout develop*

##### ***Merge friend\_groups branch to develop***

*git merge --no-ff friend\_groups*

##### ***Push changes to remote repository***

*git push origin develop*

# **Scenario #3**

Let's walk through the git commands that go along with each step in the scenario you just observed in the video above.

#### **Step 1: Andrew commits his changes to the documentation branch, switches to the development branch, and pulls down the latest changes from the cloud on this development branch, including the change I merged previously for the friends group feature.**

##### ***Commit changes on documentation branch***

*git commit -m "standardized all docstrings in process.py"*

##### ***Switch to develop branch***

*git checkout develop*

##### ***Pull latest changes on develop down***

*git pull*

#### **Step 2: Then, Andrew merges his documentation branch on the develop branch on his local repository, and then pushes his changes up to update the develop branch on the remote repository.**

##### ***Merge documentation branch to develop***

*git merge --no-ff documentation*

##### ***Push changes up to remote repository***

*git push origin develop*

#### **Step 3: After the team reviewed both of your work, they merge the updates from the development branch to the master branch. Now they push the changes to the master branch on the remote repository. These changes are now in production.**

##### ***Merge develop to master***

*git merge --no-ff develop*

##### ***Push changes up to remote repository***

*git push origin master*

### **Resources**

There's a great article on a successful git branching strategy that you should really read [here](http://nvie.com/posts/a-successful-git-branching-model/).

### **Note on Merge Conflicts**

For the most part, git makes merging changes between branches really simple. However, there are some cases where git will be confused on how to combine two changes, and asks you for help. This is called a merge conflict.

Mostly commonly, this happens when two branches modify the same file.

For example, in this situation, let’s say I deleted a line that Andrew modified on his branch. Git wouldn’t know whether to delete the line or modify it. Here, you need to tell git which change to take, and some tools even allow you to edit the change manually. If it isn’t straightforward, you may have to consult with the developer of the other branch to handle a merge conflict.

You can learn more about merge conflicts and methods to handle them [here](https://help.github.com/articles/about-merge-conflicts/).

[Versioning Data Science](https://shuaiw.github.io/2017/07/30/versioning-data-science.html)

[How to Version Control Your Production Machine Learning Models](https://blog.algorithmia.com/how-to-version-control-your-production-machine-learning-models/)