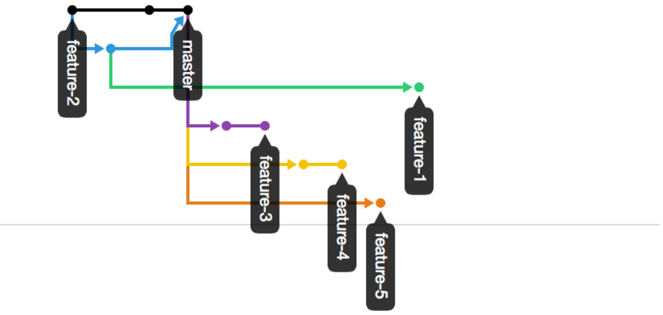
Lab 3: Branching and Merging

This Lab will attempt to demonstrate the concepts of branching merging.

Remember that in git data is stored in a directional acyclic graph.

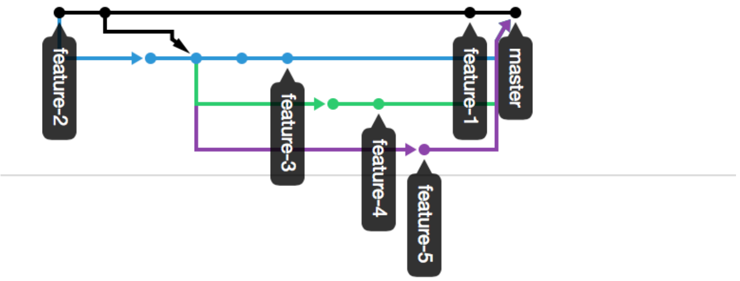
For example consider the following graph with a master branch and multiple feature branches:



A complex set of branches is actually simple to merge (***caveat****: these features are independent*)

```
Git merge feature-1 feature-2 feature-3 feature-4 feature-5
```

Now consider the same repo after merging:



Credit line: images are screenshots of github's web UI for git-flow.

Lab 3: Part 1 - Branching and Merging

BRANCHING:

1. Create a new directory for this lab called lab 3:

```
mkdir ./lab3/
cd ./lab3/
```

2. And initialize the repo

```
git init
```

3. Now create some main content (feel free to create a template repo README for yourself)

```
echo "Read ME" > ./README.txt
git add .
git commit -m "Ansestor version"
```

4. now create another branch with a new "feature" (in this case just a text file)

```
git branch feature-1
git checkout feature-1
echo "feature code" > ./feature-1.txt
git add feature-1.txt
git commit -m "feature 1"
```

5. Now return to the master branch *(Caveat: this step is for the lab)*

```
git checkout master
```

6. Create a second branch

```
git branch feature-2
echo "another feature code" > ./feature-2.txt
git add feature-2.txt
git commit -m "Feature 2"
```

7. Now return to the master branch *(Caveat: this step is for the lab)*

```
git checkout master
```

MERGING

8. Now let's do a multi-branch merge

```
git merge feature-1 feature-2
git log --graph
```

If all goes well you will have a recursive merged repo

9. Now Reset the lab

```
wait ;
cd ../
rm -vR ./lab3
wait ;
# example reset of part 1
mkdir ./lab3
cd lab3/
git init
echo "master code" > ./mastercode.txt
git add .
git commit -m "ansestor version"
```

Checkpoint - Continue to Part 2

LAB 3: Part 2 - Complex Merging

10. Create a couple more complex feature branches (at least 5)
(Caveat: The next part is just to demonstrate the scalability of these decoupled feature branches):

Here is a example loop to create multi-feature branching (otherwise you can create manually as before by incrementing the number for each branch)

```
for FEAT_NUMBER in $(seq 5) ; do
git branch feature-${FEAT_NUMBER}
git checkout feature-${FEAT_NUMBER}
echo "feature code for branch ${FEAT_NUMBER}" > ./feature-${FEAT_NUMBER}.txt
git add feature-${FEAT_NUMBER}.txt
git commit -m "feature ${FEAT_NUMBER}"
echo "Improved feature code for branch ${FEAT_NUMBER}" > ./feature-${FEAT_NUMBER}.txt
git add feature-${FEAT_NUMBER}.txt
git commit -m "feature ${FEAT_NUMBER} improvement"
wait ;
git checkout master
wait ;
done ;
```

11. Once you are satisfied with the feature branches, try to merge all branches at once.

```
git log --graph
git merge feature-1 feature-2 feature-3 feature-4 feature-5
wait ;
git log --graph
```

If you have not created any merge conflicts you're done. Otherwise, for the sake of time just reset this lab and try again with simpler features.