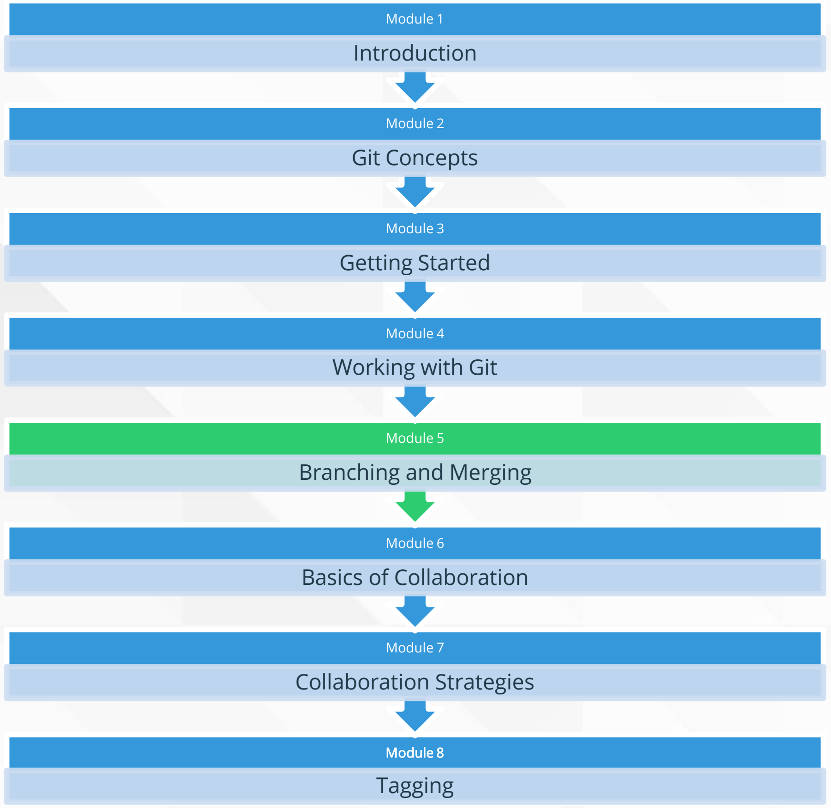
Module 5

Branching and Merging

Module Contents

Module Contents

- Branching
- Switching Branches
- Merging
- Merge Conflicts
- Merge Tools
- Removing Branches
- Branch Management



Module 1
Introduction

Module 2
Git Concepts

Module 3
Getting Started

Module 4
Working with Git

Module 5
Branching and Merging

Module 6
Basics of Collaboration

Module 7
Collaboration Strategies

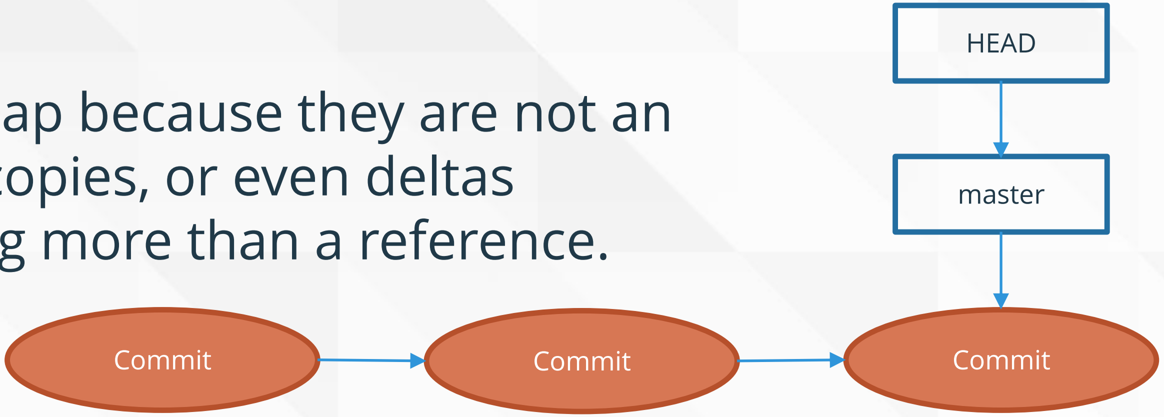
Module 8
Tagging

Branches

Branching is very easy and cheap in Git and different to most other version control systems.

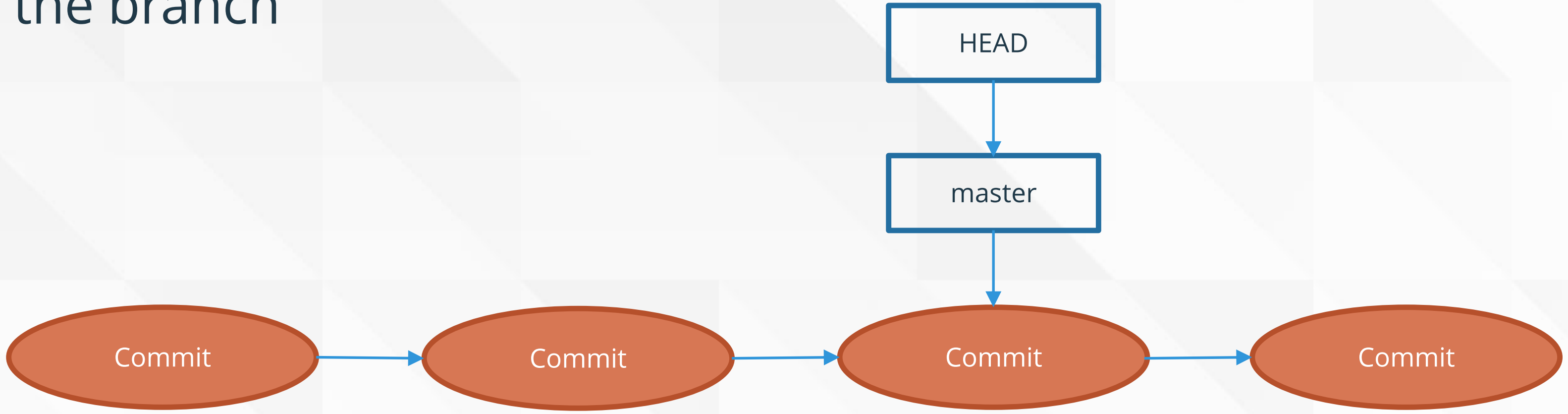
A Git branch in essence is a file stored in the .git/refs/head directory that contains the hash value of the most recent commit on that branch.

Branches are cheap because they are not an actual set of file copies, or even deltas
- they are nothing more than a reference.



Working on Branches

Whenever you make a change in Git you record that change through a commit, each reference points to the last commit made while working on that branch.
Each ref then updates itself should changes be made while working on the branch



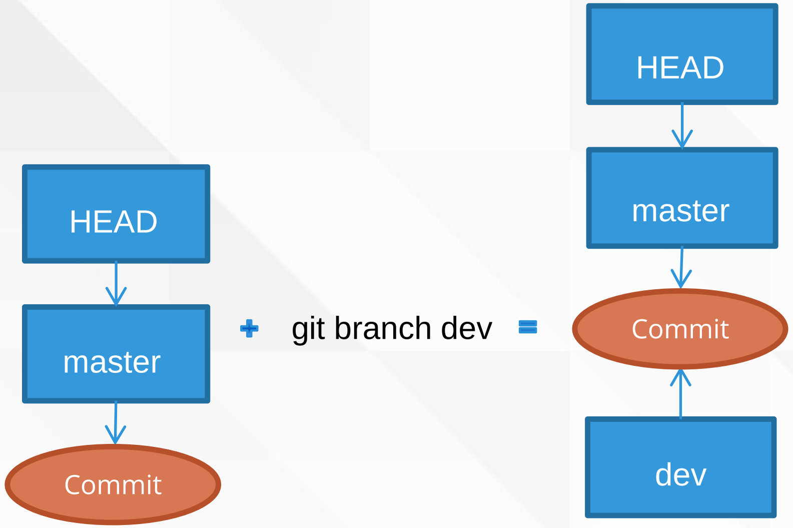
Creating Branches

By default every git repository starts with a master branch, that points to the last commit made. This will track your commits moving forward automatically with each commit.

NOTE: the master branch will not exist until you create your first commit

Creating a new branch is done via the git branch command

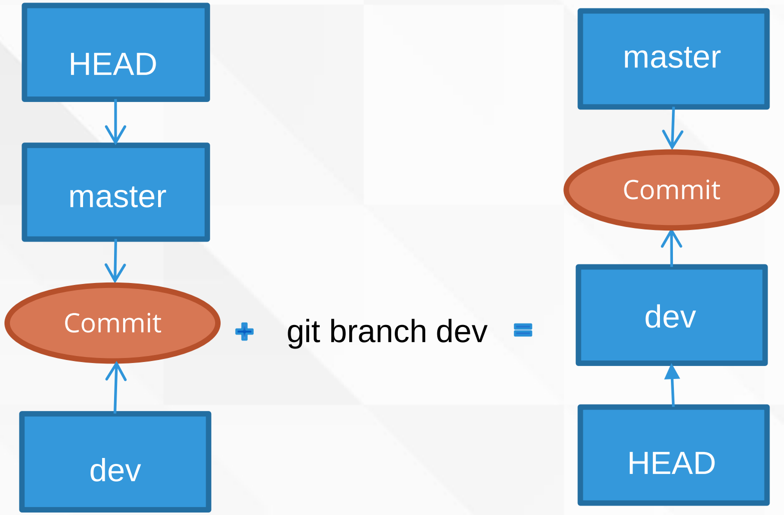
- *git branch <name>*



Switching Branches

To work on your new branch you need to use the checkout command to switch the HEAD to the tip of the new branch.

- *git checkout <branch>*
- *git checkout -b <branch>*
    - *Use this argument to create a new branch*

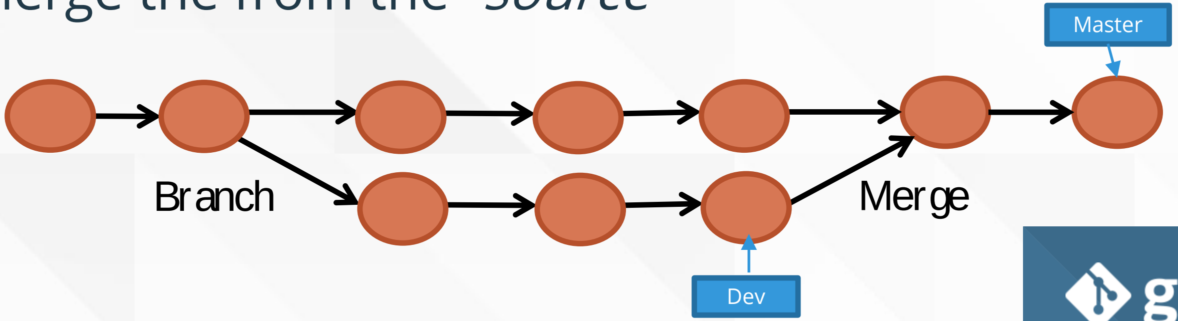


Merging

Branches usually end up with a merging into another development stream, Whenever you branch from a main line you will probably need to return to it.

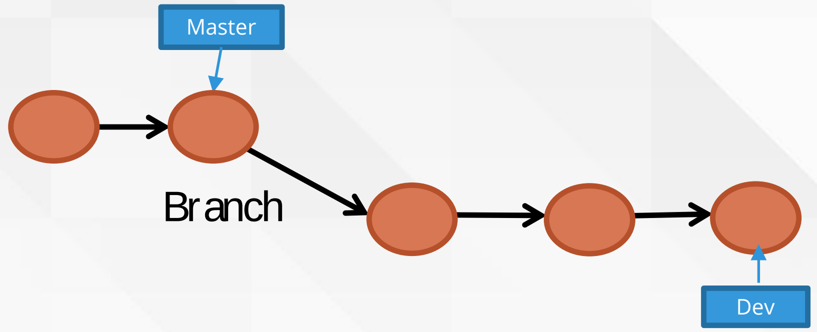
Merging which is carried out using the *git merge* command while working on the branch you wish to merge into. You must always checkout the "*target*" before you can merge the from the "*source*"

- *git checkout master*
- *git merge dev*



Fast Forward

Git has a special term, Fast Forward, for simple merges where there has been no divergence in content since you branched.

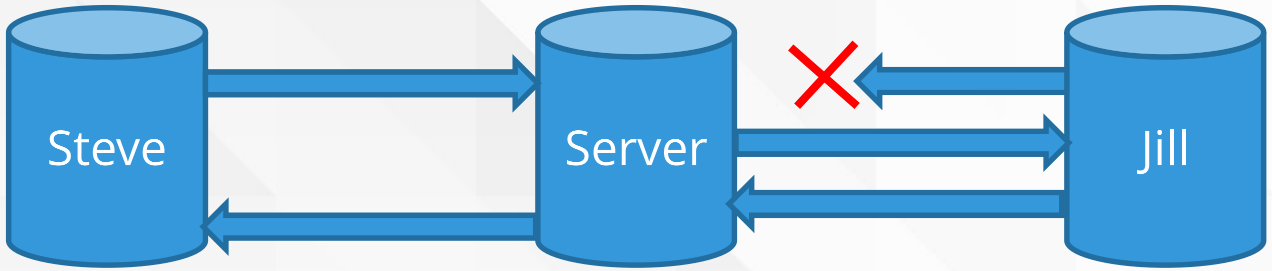


- *git checkout master*
- *git merge dev*

Whenever you attempt to merge your changes with someone else's repository (such as the server) it must always be Fast Forward.

Merging to a server

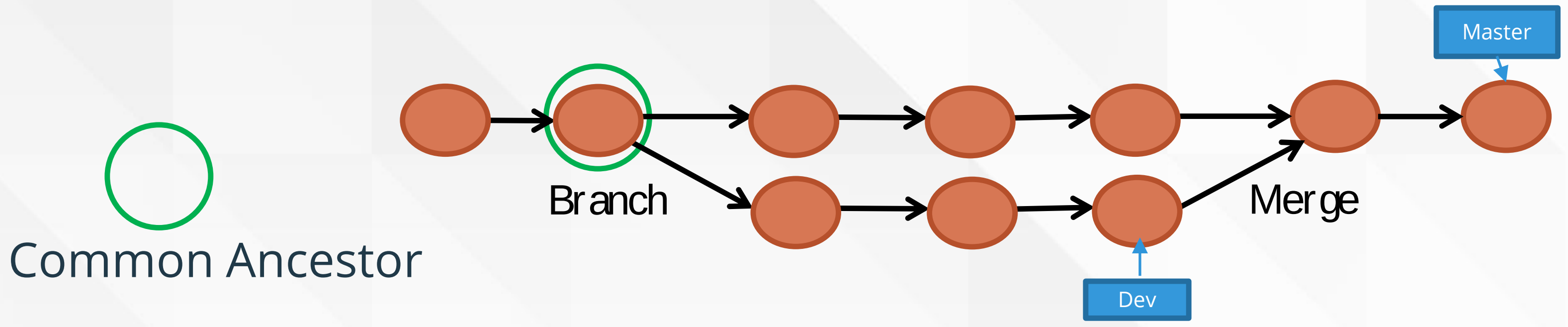
Because of Git's distributed nature you often need to perform a merge between repositories (more on this in the next module)



In this scenario, Jill cannot merge her changes to the server, unless she has locally merged the stages Steve made earlier – this is because it's not a fast forward merge.

3 Way Merging

Often there have been additional changes on the original branch and a fast forward merge is not possible.



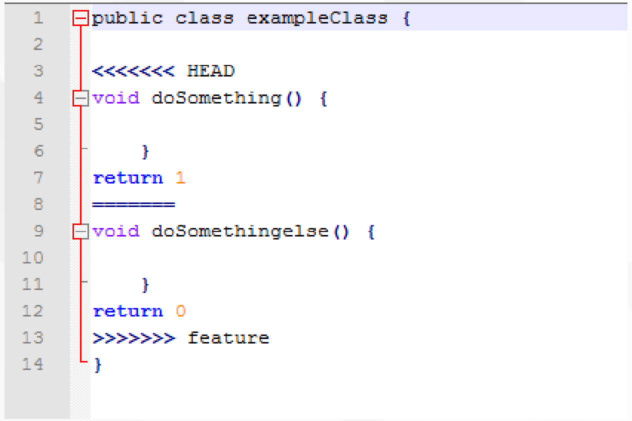
You cannot force another repository to do a three way merge – because there is a chance of conflicts

Merge Conflicts

Git flags that there is a merge conflict and suspends the commit at that point for you to resolve

The Git status command reflects this and shows the file/s that need merging

Git also helps you identify the conflicted areas of the files, by adding the diff information to the conflicted file in your working directory.



```
public class exampleClass {

<<<<<<< HEAD
void doSomething() {

    }
return 1
=======
void doSomethingelse() {

    }
return 0
>>>>>>> feature
}
```

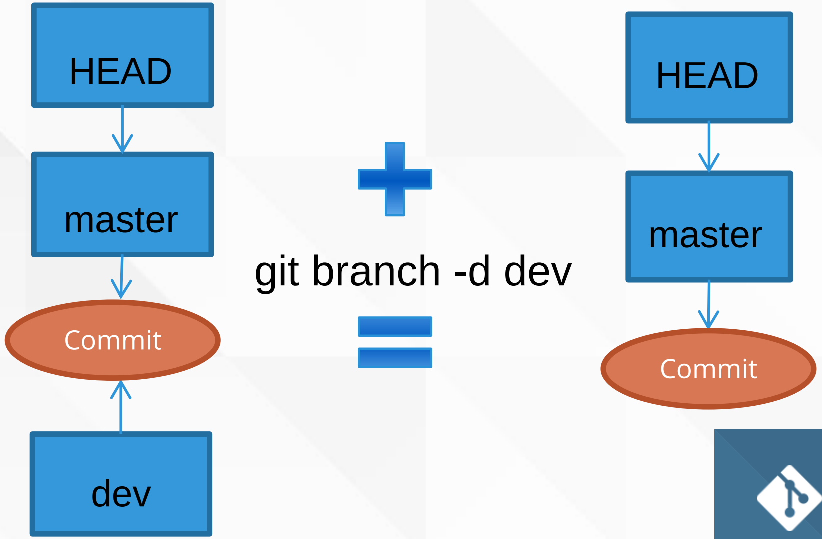
Removing Branches

Once a branch has served its purpose and all work on it has been merged, there is no need to retain it – we are only removing the reference, not the commits.

To delete a branch use:

- *git branch -d <name>*
- *git branch -D <name>*

The -D option forces the delete - even if the branch has not been merged



Branch Management

To help manage your branches and Merges, Git provides a number of additional options to the branch command.

git branch --merged

- List branches that do not contain changes not referenced by the current HEAD

git branch --no-merged

- List branches that contain commits with changes not referenced by the current HEAD

git branch --verbose

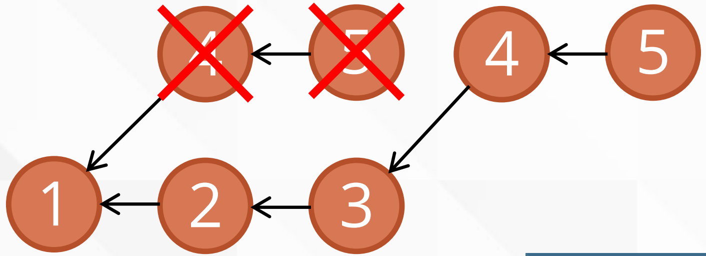
- List additional information such as the commit at the top of the branch

Rebasing Branches

Rebasing is the process of moving a branch to a new base commit – i.e. a new starting point.

Internally, Git accomplishes this by creating new commits and applying them to the specified base literally rewriting your branch history.

- git rebase <base>
- git rebase -i <base>
  - Opens an interactive rebasing session



Lab Exercise

End of Module