Andhra Pradesh State Skill Development Corporation

Source Code Management Using Git & GitHub

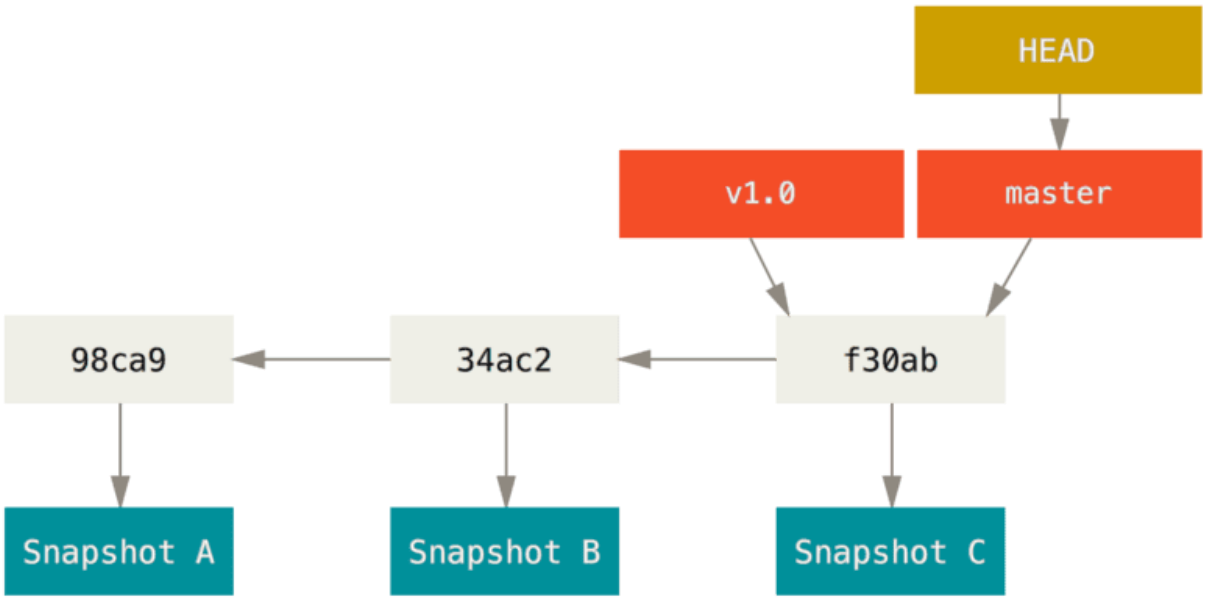
Git Branches

Git Branching

Nearly every VCS has some form of branching support. Branching means you diverge from the mainline of development and continue to do work without messing with that mainline. In many VCS tools, this is a somewhat expensive process, often requiring you to create a new copy of your source code directory, which can take a long time for large projects.

Some people refer to Git's branching model as its "killer feature," and it certainly sets Git apart in the VCS community. Why is it so special? The way Git branches are incredibly lightweight, making branching operations nearly instantaneous, and switching back and forth between branches generally just as fast. Unlike many other VCSs, Git encourages workflows that branch and merge often, even multiple times in a day. Understanding and mastering this feature gives you a powerful and unique tool and can entirely change the way that you develop.

A branch in Git is simply a lightweight movable pointer to one of these commits. The default branch name in Git is master. As you start making commits, you're given a master branch that points to the last commit you made. Every time you commit, the master branch pointer moves forward automatically.



Creating a NewBranch

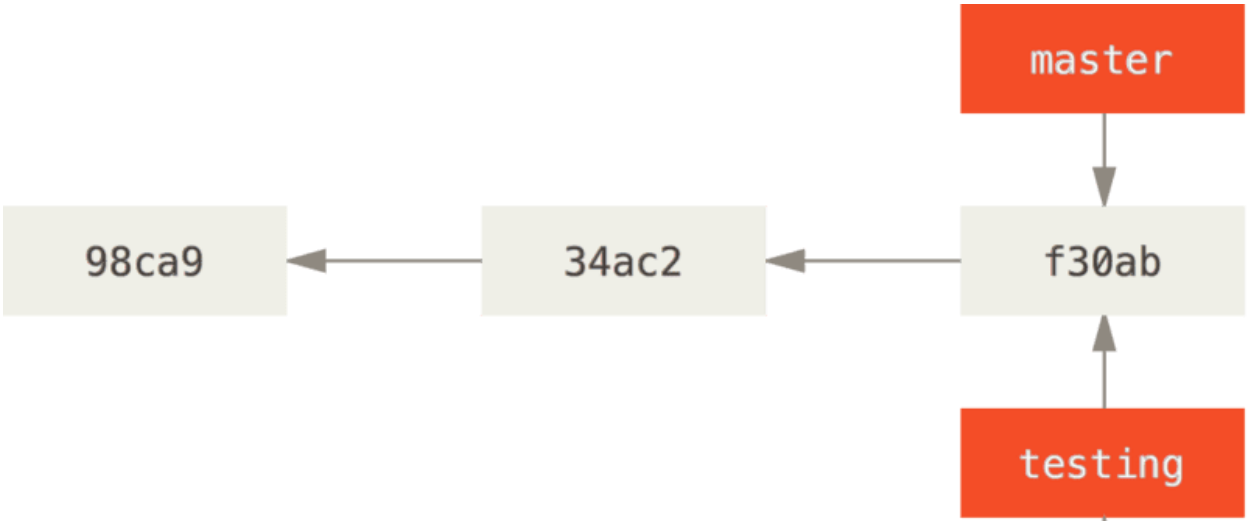
What happens when you create a new branch? Well, doing so creates a new pointer for you to move around. Let's say you want to create a new branch called testing. You do this with the git branch command:

git branch <new branch name>

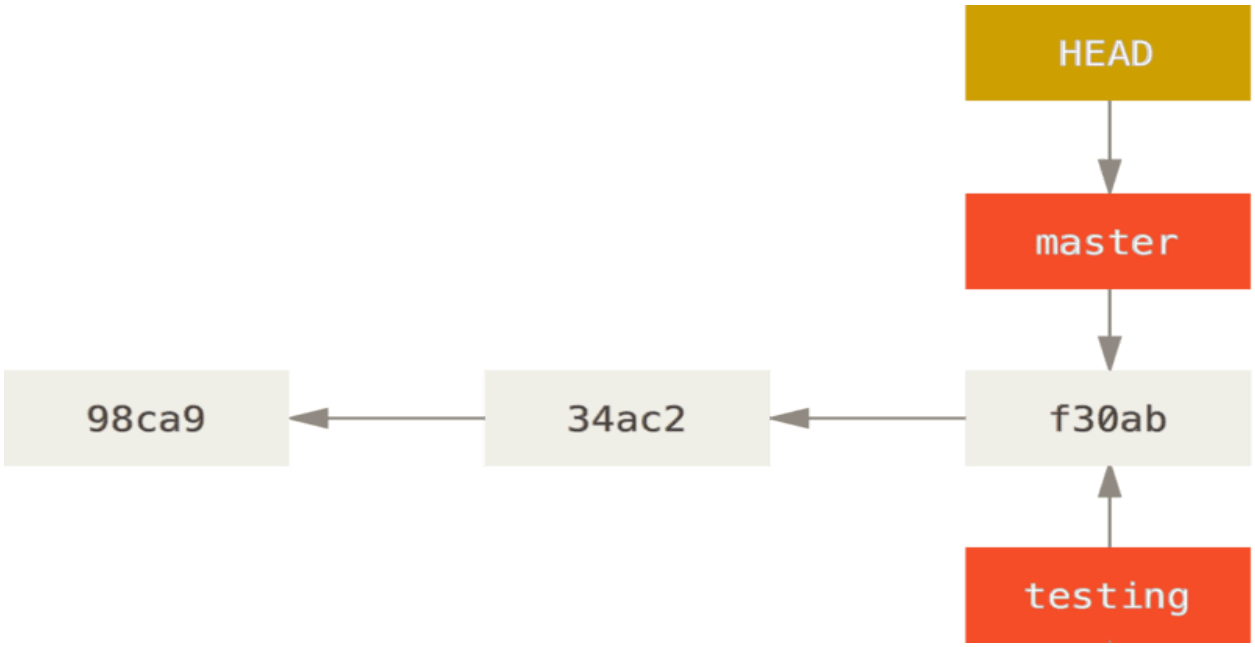
For example, here we are creating a new branch and named as testing

git branch testing

This creates a new pointer to the same commit you're currently on.



How does Git know what branch you're currently on? It keeps a special pointer called HEAD. Note that this is a lot different than the concept of HEAD in other VCSs you may be used to, such as Subversion or CVS. In Git, this is a pointer to the local branch you're currently on. In this case, you're still on the master. The git branch command only created a new branch —it didn't switch to that branch.

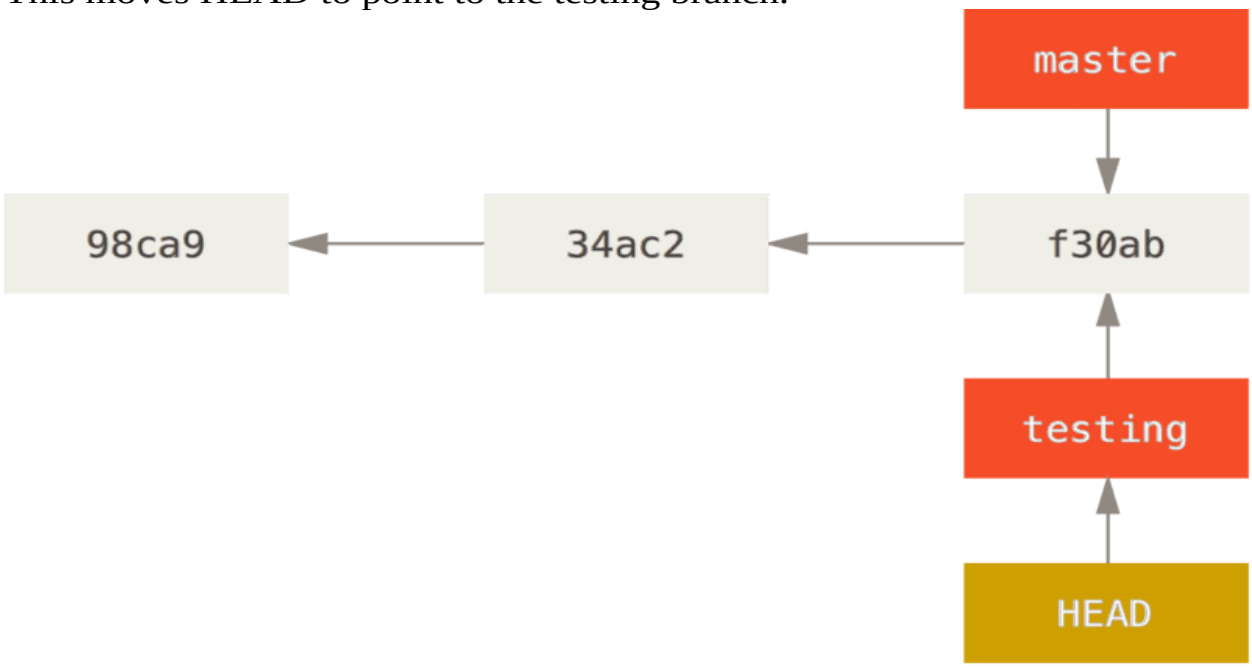


Switching Branches

To switch to an existing branch, you run the git checkout command. Let's switch to the new testing branch:

git checkout testing

This moves HEAD to point to the testing branch.



Merge Branch

Merging is Git's way of putting a forked history back together again. The git merge command lets you take the independent lines of development created by git branch and integrate them into a single branch.

To merge the branch first we should have to check out that branch

git merge <branch name>

Delete the Branch in localrepository

To delete any branch without merging the command is

git branch -D <branch name>

To delete branch after merge

git branch -d <branch name>

To delete the branch in the remote repository

git push <remote name> --delete <branch name>