Quick2Cloud for Cloud Systems

Built on IBM Cloud

Quick2Cloud College - Course 103

Running the Application Locally with a Repository

Repository Branches

When a repository is created, by default, it contains one repository **branch** by the name of **master**. Think of a branch as a pseudo-subdirectory that holds the code for your application□s version. Think of your previous commits as additions and updates to the code in the master branch as you develop features.

In our local repository scenario, master will initially be the branch that contains the sample application code as given to you by *Quick2Cloud* which we will define as **ver00**. Any changes you make in the sample application, from now on, will be treated as a feature to the ver00□s master branch. At some point you will come to the decision that you have added enough features, patiently doing an add & commit after each feature is tested, and now it□s time to leave ver00 behind and onto another version of the application.

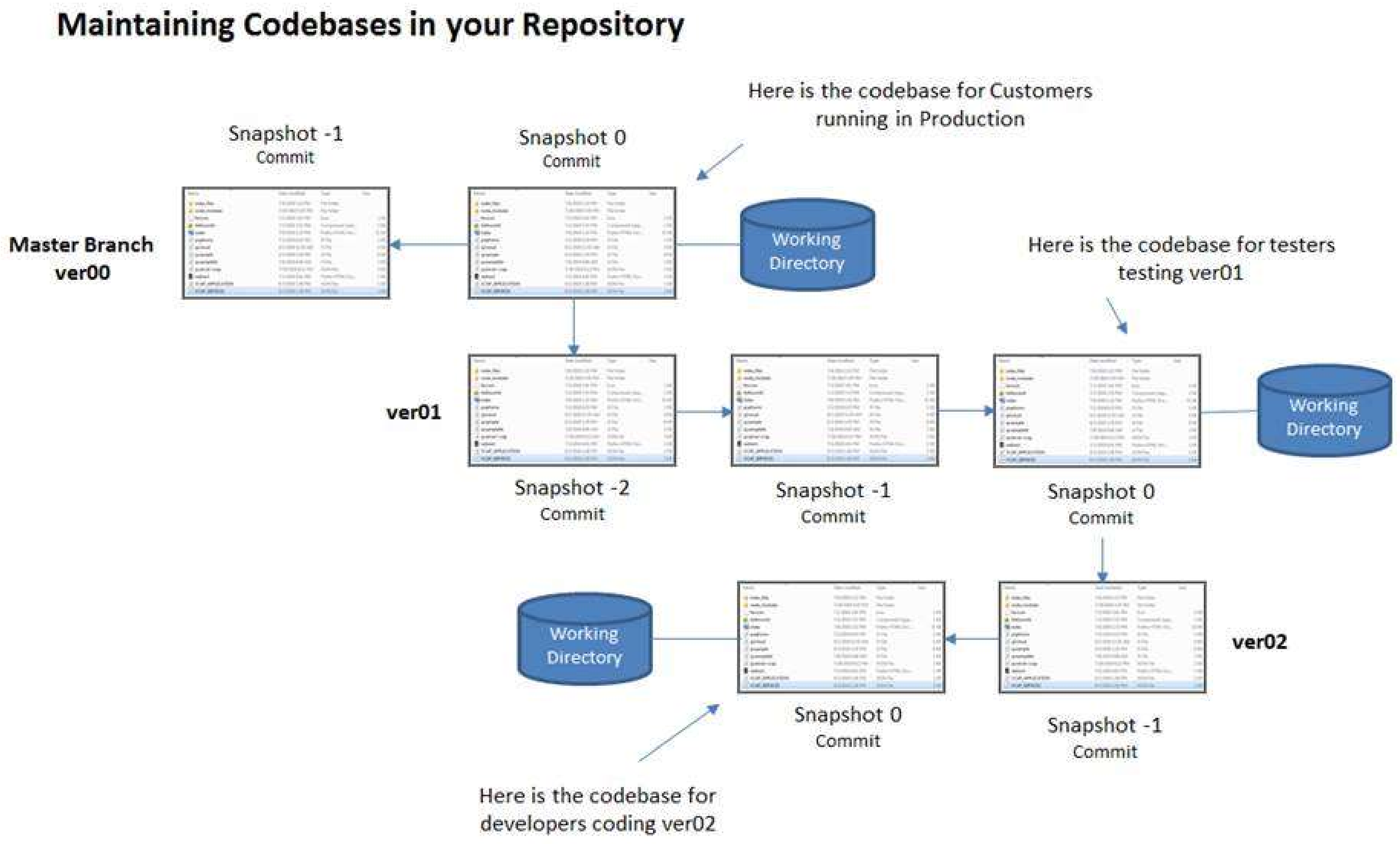
When it is time for that to happen, which is typically when a significant change to what the application does is introduced, you will create a new branch. In keeping with your college course□s methodology, this branch will be called **ver01.** Once this branch is created by a **git branch** command it will contain a copy of the master branch at its most current snapshot (its last commit). At this moment, the code in ver00 and ver01 are identical.

Every commit (a snapshot) you take going forward will now be in the ver01 branch and no longer changing what is in ver00□s master branch. The significance of this is very important. Master holds the latest ver00 version, usually the production version you continue to run your live application out of. As you develop in **ver01**, that code is quickly becoming a very different codebase and will need a lot of testing before you want to actually rely on it. As you develop new features, you commit them again to ver01 leaving master and your production code alone.

At some point ver01 will be complete and be given over for testing by a testing team or automation. Testing will continue on ver01 for some time until it is blessed as golden. On that blessed occasion you can **push** ver01 into ver00 (master) making master the same code as ver01 and hence; redefining ver00 as your new production code. After that push you will then create another branch called **ver02**, it starts where ver01 left off. Ver02 now becomes the branch for your next big application version and thus; the development cycle repeats itself.

Note: As an alternative, you can make ver00 your production, ver01 your test and ver02 your development. In this way, your Customers, testers and developers will have their own environment to work with and not get affected in changes not intended for them.

Let us look at that graphically:



As the developer of the sample web application you have the ability to develop in any of the branches you wish. Keep in mind; working in the master branch to add new features will affect your production Customers so you have to ask yourself if that is really what you want unless it□s for fixing bugs - that□s all up to you.

The branch named ver01 can also be developed in but that will only influence what the testers are testing which is great if the testers find problems and you want to supply fixes. Plus, you also have the freedom to work in ver02 which will later become your most advanced version to date. If you read these words correctly, you will see that you can switch between branches at will □ but why would you want to do that?

Why switch branches

1. Switch to ver00 fix a production bug
2. Switch to ver01 fix a tester□s bug
3. Switch to ver01 to add requested feature
4. Switch to ver02 to code new version

Here is one of the truly amazing things about the git software, when you switch branches, using a **git checkout** command, your working directory is **restored** to the working directory of that target branches□ snapshot 0 - which is the most recent codebase for that branch. It□s best to finish up and **commit or stash** your current work in your working directory before you switch branches or git won□t allow the switch to take place. This switching feature of the git methodology makes this repository process a powerful way to manage multiple versions of your application.

Stated slightly different

Suppose you find that the latest fix for the testers you did for ver01 did not fix the problem, in fact, it made it worse. To remedy this problem you could switch to branch ver01, have your working directory restored to snapshot 0 then hand back-out all the changes you made, in all spots in the code, but that will require some finger crossing and a prayer in hopes it was backed out correctly.

Or, using git methodology you could switch to and **rewind** to branch ver01 (snapshot -1) which will restore your working directory to the previous snapshot before you put in the broken fix.

Note: Git won□t let you rewind into ver01□s past to fix something and **re-commit** it as ver01□s snapshot 0. You will have to make a new branch and **commit** there **-** then progress forward form that point. In that case you would create a **ver01-a** branch which becomes the continuation of ver01. More information on this interesting topic can be found later in the course.

As you can imagine, you can get quite creative using branches and everyone does get creative when it comes to their specific style of coding.

Let□s take a breather

Quick2Cloud has begun introducing you to a simple repository approach to managing a local application that is made up of various features and versions. Our course will move forward using this approach and when you take **Course 104** on Continuous Delivery then *Quick2Clouds* approach will expand the use of branches to cover how this all works in the Cloud.

Are you sweating...

Wow, that was a lot to learn! But now it is time to put what you learned to work by creating a repository right in the middle of your q2c-web-app working directory so you can mature yourself in how you develop code and forever banish those nasty old coding habits that hits us all, especially when we have more than one developer in the mix.

In the next section you will install a local git repository and get your sample web application Committed in its master branch.

Continue