Linux Academy

Live! Lab

CodeCommit Basics

Contents

Lab Connection Information

- Labs may take up to five minutes to build
- Access to an AWS Console is provided on the Live! Lab page, along with your login credentials
- Ensure you are using the N. Virginia region
- Labs will automatically end once the alloted amount of time finishes

Related Courses

CodeCommitBasics

Related Videos

What is CodeCommit?

Create, View, Edit, and Delete a Repository

Cloning Repositories, Commits, Push, and Pulls

Branches (local)

Need Help?

Linux Academy Community

... and you can always send in a support ticket on our website to talk to an instructor!

AWS CodeCommit offers users a remote, GitHub-like repository to work with code. Using Git, the Amazon Dashboard, and the Amazon CLI, this lab walks through creating a repository, adding, committing and pushing files to a remote branch, and creating, merging, and deleting branches.

Log into the AWS Dashboard using the details provided on the Live! Lab page. We can now begin.

Create and Clone a Repository

From the AWS Dashboard, select **CodeCommit**, located under the **Developer Tools** heading. Click on **Get Started**.

Give the repository a name; we used *LiveLabRepo*. The description can either be filled out or left empty. **Create repository**.

This takes you do the CodeCommit Dashboard. To learn more about your LiveLabRepo repository, you can select the name from the list. As of now, everything is empty.

Since we want to work from the command line, the lab provides, under the **EC2** service, an Amazon Linux AMI from which to work. Wait until the status checks are finished (it will read *2/2 checks*). From here, you want to note the public IP address of the lab (found in the description section of the EC2 Dashboard), and open the terminal on your workstation.

SSH into the EC2 instance, replacing the IP address with the one provided in your lab:

```
[user@workstation]$ ssh linuxacademy@54.175.178.118
```

The password is *123456*.

This instance has already been set up and configured to work with the AWS CLI. To clone the repository, run the following, using the **HTTPS Clone URL** provided on the **CodeCommit Dashboard**. `local-lab-repo` is the name of the directory into which the repository is cloned. AWS then creates the directory.

```
[linuxacademy@ip-10-0-0-100 ~]$ git clone https://git-codecommit.us-
east-1.amazonaws.com/v1/repos/LiveLabRepo local-lab-repo
Cloning into 'local-lab-repo'...
warning: You appear to have cloned an empty repository.
Checking connectivity... done.
```

Should you view the contents of your home directory, the new directory has been added. Change directories into `local-lab-repo`:

```
[linuxacademy@ip-10-0-0-100 ~]$ cd local-lab-repo/
```

Push Commits and Create Branches

We first want to create a new file in our repository:

```
[linuxacademy@ip-10-0-0-100 local-lab-repo]$ touch newfile.txt
```

Add the file to the repository, using normal Git commands:

```
[linuxacademy@ip-10-0-0-100 local-lab-repo]$ git add newfile.txt
```

And commit the file:

```
[linuxacademy@ip-10-0-0-100 local-lab-repo]$ git commit -m "live lab
commit."
[master (root-commit) 57ec401] live lab commit.
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 newfile.txt
```

We are now ready to push to the *master* brand of our repository. The master branch is the primary branch located within our remote repository.

```
[linuxacademy@ip-10-0-0-100 local-lab-repo]$ git push origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 215 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote:
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/LiveLabRepo
 * [new branch]  master → master
```

If we return to the repository on the CodeCommit Dashboard, we can now see that our "empty" repository is no longer empty, but the new file we created on the command line has been added.

We next want to create a new branch and create a new file in that branch. For this, we return to the command line, where we call our branch *secondbranch*.

```
[linuxacademy@ip-10-0-0-100 local-lab-repo]$ git branch secondbranch
```

To confirm the creation of the branch (or just to see what branches are there), use the `git branch` command:

```
[linuxacademy@ip-10-0-0-100 local-lab-repo]$ git branch
* master
  secondbranch
```

Switch to the new branch using the `checkout` command:

```
[linuxacademy@ip-10-0-0-100 local-lab-repo]$ git checkout secondbranch
Switched to branch 'secondbranch'
```

Add a second file, commit, and push it into the *secondbranch* branch:

```
[linuxacademy@ip-10-0-0-100 local-lab-repo]$ touch secondfile.txt
[linuxacademy@ip-10-0-0-100 local-lab-repo]$ git add secondfile.txt
[linuxacademy@ip-10-0-0-100 local-lab-repo]$ git commit -m "second file"
[secondbranch f039624] second file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 secondfile.txt
[linuxacademy@ip-10-0-0-100 local-lab-repo]$ git push origin
secondbranch
Counting objects: 2, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 245 bytes | 0 bytes/s, done.
Total 2 (delta 0), reused 0 (delta 0)
remote:
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/LiveLabRepo
 * [new branch]  secondbranch → secondbranch
```

Note where, in the `git push` command, we defined the *secondbranch* branch and not the *master*.

To verify the results, return to the CodeCommit Dashboard, where you can now select branches under the *LiveLabRepo* repository.

Merge and Delete Branches

Now, say we want to merge the second branch into the master branch — in practice, this would happen because of a successful bug fix, added or changed code, or a number of other changes.

Switch back to the master branch:

```
[linuxacademy@ip-10-0-0-100 local-lab-repo]$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
```

To merge, use the `git merge` command:

```
[linuxacademy@ip-10-0-0-100 local-lab-repo]$ git merge secondbranch
Updating 57ec401..f039624
Fast-forward
 secondfile.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 secondfile.txt
```

However, if we check the CodeCommit console on AWS, the second branch does not show as merged -- *secondfile.txt* is not present in the master repository. This is because we have yet to *push* the local branch into our remote repository.

```
[linuxacademy@ip-10-0-0-100 local-lab-repo]$ git push origin master
Total 0 (delta 0), reused 0 (delta 0)
remote:
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/LiveLabRepo
   57ec401..f039624  master -> master
```

If you return to the dashboard now, you can see the *secondbranch* file, *secondfile.txt* is now included in the master branch.

Finally, since we have finished with *secondbranch* and our master repository is up to date, we can remove *secondbranch*.

We need to do this twice: On the remote repository, and on our local machine:

```
[linuxacademy@ip-10-0-0-100 local-lab-repo]$ git push origin --delete
secondbranch
remote:
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/LiveLabRepo
 - [deleted] secondbranch
```

Should you refresh the dashboard, the *master* branch is the only branch remaining.

To delete the branch locally:

```
[linuxacademy@ip-10-0-0-100 local-lab-repo]$ git branch -d secondbranch
Deleted branch secondbranch (was f039624).
```

This can be confirmed with the `git branch` command, which will now only output the master branch.