# Working with AWS CodeCommit

## Overview

AWS CodeCommit is a highly scalable, managed source control service that hosts private Git repositories. CodeCommit stores your data in Amazon S3 and Amazon DynamoDB giving your repositories high scalability, availability, and durability. You simply create a repository to store your code. There is no hardware to provision and scale or software to install, configure, and operate.

This hands-on lab gives you practice with AWS CodeCommit, part of AWS Developer Tools. In this lab, you will first create a code repository in AWS CodeCommit. Then you will create a local repository on a Linux instance running in EC2. After you create the local repo, you will make some changes to it. Then you will synchronize (commit) your changes to the AWS CodeCommit repository.

### Topics covered

This lab demonstrates how to:

- Create a code repository using AWS CodeCommit via the Amazon Management Console
- Create a local code repository on the Linux instance using git
- Synchronize a local repository with an AWS CodeCommit repository

### Prerequisites

Students should have some development experience and understand the principles of source code repositories, and has some prior development experience. Students should be comfortable with making SSH connections to instances running in Amazon EC2, and using Linux commands and editors from the command line in Linux. Students should have taken at a minimum *Introduction to Amazon Elastic Compute Cloud (EC2)* prior to taking this lab.

## Task 1: Create an AWS CodeCommit Repository

To get started creating your first AWS CodeCommit Repository, you will access AWS CodeCommit in the AWS management console.

1. In the **AWS Management Console**, on the Services menu, click **CodeCommit**.

2. Click Get Started

If a Dashboard page appears instead of the welcome page, click **Create repository**.

1. On the **Create repository** page, configure:

    - **Repository name:** `My-Repo`

- **Description:** `My first repository`
- Click Create

An empty repository called *My-Repo* will be created in AWS CodeCommit. Connection details will be displayed.

# Task 2: Connect to Your EC2 Instance

You can create new EC2 instance or use exisitng one. In this task, you will connect to the instance.

## Windows Users: Using SSH to Connect

These instructions are for Windows users only.

If you are using Mac or Linux, skip to the next section.

1. **Create/Download PPK** using AWS Console.

2. Save the file to the directory of your choice.

You will use PuTTY to SSH to Amazon EC2 instances.

If you do not have PuTTY installed on your computer, download it here.

1. Open PuTTY.exe

2. Configure the PuTTY to not timeout:

- Click **Connection**
- Set **Seconds between keepalives** to `30`

This allows you to keep the PuTTY session open for a longer period of time.

1. Configure your PuTTY session:

- Click **Session**
- **Host name (or IP address):** Copy and paste the **EC2PublicIP** shown to the left of these instructions
- In the **Connection** list, expand **SSH**
- Click **Auth** (don't expand it)
- Click Browse
- Browse to and select the PPK file that you downloaded
- Click Open to select it
- Click Open

1. Click Yes to trust the host and connect to it.

2. When prompted **login as**, enter: `ec2-user`

This will connect to your EC2 instance.

1. Windows Users: skip ahead to the next task.

## Mac and Linux Users

These instructions are for Mac/Linux users only. If you are a Windows user, skip ahead to the next task.

1. **Create/Download PEM** using AWS Console.
2. Save the file to the directory of your choice.
3. Copy this command to a text editor:

```
chmod 400 KEYPAIR.pem

ssh -i KEYPAIR.pem ec2-user@EC2PublicIP
```

1. Replace *KEYPAIR.pem* with the path to the PEM file you downloaded.
2. Replace *EC2PublicIP* with the value of EC2PublicIP shown to the left of these instructions.
3. Paste the updated command into the Terminal window and run it.
4. Type `yes` when prompted to allow a first connection to this remote SSH server.

Because you are using a key pair for authentication, you will not be prompted for a password.

# Task 2: Create a Local Repository using Git

This task provides an example of how you would use AWS CodeCommit to synchronize to any local code repository that you might create in your normal production development environment.

1. Install the git client:

```
sudo yum install -y git
```

1. Run these commands to use the Git credential helper with the AWS credential profile, and enable the Git credential helper to send the path to repositories:

```
git config --global credential.helper '!aws codecommit credential-helper $@'
git config --global credential.UseHttpPath true
```

You will now need to obtain the HTTPS URL of your AWS CodeCommit repository.

1. In the **AWS CodeCommit Management Console**, click **Clone URL** > then click **Clone HTTPS**.

This copies your repository URL to your clipboard. The URL look similar to: *https://git-codecommit.us-east-1.amazonaws.com/v1/repos/My-Repo*

1. Paste the clone URL into your text editor.
2. In your remote SSH session, copy and paste:
3. Replace **URL** with your CodeCommit clone URL that you copied to your text editor.
4. Run the command by pressing **ENTER**.

You should see the following message:

*Cloning into 'my-demo-repo'...*
*warning: You appear to have cloned an empty repository.*

**Congratulations!** You have successfully connected to and synchronized with the AWS CodeCommit repository.

Next, you will conclude with a short demonstration of making a change and synchronizing the repositories. This is a mini example of the workflow of synchronizing code changes during the development process.

## Task 3: Making a Code Change and First Commit to the Repo

In this task, you will create your first commit in your local repo. To do this, you will create two example files in your local repo. You will use Git to stage the change to your local repo and then commit the change to your local repo.

1. Copy and paste these commands to create two files in your local repo:

```
cd ~/My-Repo
echo "The domestic cat (Felis catus or Felis silvestris catus) is a small, usually furry, domesticated, and carnivorous ma
echo "The domestic dog (Canis lupus familiaris) is a canid that is known as man's best friend." >dog.txt
```

- List the directory to see that the files were created:

```
ls
```

- Run this command to stage the change in your local repo:

```
git add cat.txt dog.txt
```

- View the status of your repo:

```
git status
```

You will see that the two files are ready to be committed to the repository.

- Run this command to commit the change in your local repo:

```
git commit -m "Added cat.txt and dog.txt"
```

- View details about the commit you just made:

```
git log
```

Now that you have an initial commit in your local repo, you can push the commit from your local repo to your AWS CodeCommit repository.

## Task 4: Push your First Commit

In this step, you will push the commit from your local repo to your AWS CodeCommit repository.

- Run this command to push your commit through the default remote name Git uses for your AWS CodeCommit repository (origin), from the default branch in your local repo (master):

```
git push -u origin master
```

Once you have pushed code to your AWS CodeCommit repository, you can view the contents using the AWS CodeCommit console.

1. In your **AWS CodeCommit Management Console**, ensure that you are in your **My-Repo** repository.

2. If your are not in your **My-Repo** repository, click your **My-Repo** repository.

3. Refresh your screen.

The two files that you added to your repository should be displayed.

1. Click each file to view its contents.

**Congratulations!** You have created an AWS CodeCommit repository and can now consider how to bring its features and capabilities to your development workflow. These include:

- **Collaboration:** AWS CodeCommit is designed for collaborative software development. CodeCommit allows you to commit, diff, and merge your code allowing you to easily maintain control of your team's projects. You can create a repository from the AWS Management Console, AWS CLI, or AWS SDKs and start working with the repository using Git.

- **Encryption:** You can transfer your files to and from AWS CodeCommit via HTTPS and SSH. Your repositories are also automatically encrypted at rest through AWS Key Management Service using customer-specific keys.

- **Access Control:** AWS CodeCommit uses AWS Identity and Access Management to control and monitor who can access your data as well as how, when, and where they can access it.

- **High Availability and Durability:** AWS CodeCommit stores your repositories in Amazon S3 and Amazon DynamoDB. Your data is redundantly stored across multiple facilities. This architecture increases the availability and durability of your repository data.

- **Unlimited Repositories:** AWS CodeCommit allows you to create as many repositories as you need, with no size limits. You can store and version any kind of file, including application assets such as images and libraries alongside your code.

- **Easy Access and Integration:** You can use the AWS Management Console, AWS CLI, and AWS SDKs to manage your repositories. You can also use Git commands or Git graphical tools to interact with your repository source files. AWS CodeCommit supports all Git commands and works with your existing Git tools. You can integrate with your development environment plugins or continuous integration/continuous delivery systems.

## Conclusion

**Congratulations!** You now have successfully:

- Created a code repository using the AWS CodeCommit Management Console
- Created a local code repository on your Linux instance using git
- Synchronized a local repository with an AWS CodeCommit repository