EDD AWS CodePipeline and CodeBuild Deployment Process

Version 1.02

May 28, 2025

Initial Draft

Document Revision History

Version	Date	Author	Change
1.00	4/21/2025	Karl Thomas	Initial draft
1.01	4/21/2025	Bill Franki	Internal Review, Input, Formatting, Validation, QC
1.02	5/28/2024	Bill Franki	Client Review, Input, Formatting, Validation, QC

Approval Record

Name	Position	Version	Approval Date

Table of Contents

List of Figures ..... **ii**
List of Tables ..... **ii**
1. The EDD Deployment Process **1**
1.1 Accounts 1
1.2 Repositories 3
1.3 EDD Connect 3
1.4 Virtual Callback 3
1.5 Lambda Deployments and Supporting Resources 3
1.6 Connect and Lex Deployments 4
1.7 Cross Account Code Pipeline 5
1.8 Provisioning a Pipeline 6
1.9 How Pipelines Are Used to Deploy Code 7
1.10 Creating a New Environment 7

List of Figures

Figure 1: Accounts Detail 1
Figure 2: AWS CodeCommit Repositories 3
Figure 3: Connect and Lex Deployment Detail 4
Figure 4: Cross Account Code Pipeline Detail 5
Figure 5: AWS Connect instance and Lex Console Export Pipeline 6
Figure 6: Code Deployment Pipeline 7

List of Tables

Table 1: AWS Account Numbers 2

1. The EDD Deployment Process

We use AWS CodePipeline and AWS CodeBuild to deploy code from a central account we call the **Shared Account**. The pipelines are created with CloudFormation using AWS Service Catalog.

1.1 Accounts

We currently have a number of accounts. We deploy to these accounts when code is pushed to the associated branches.

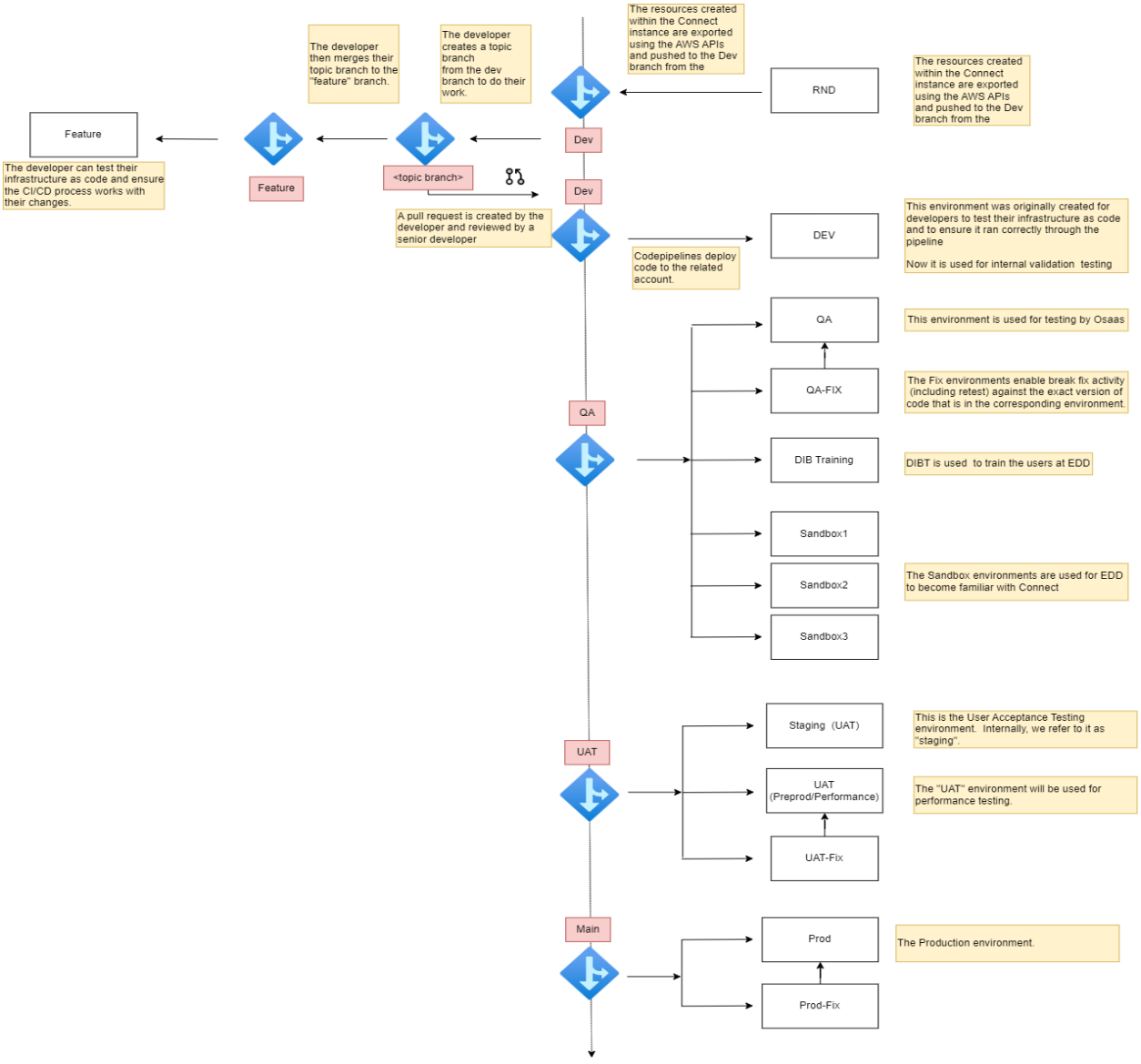

Figure 1: Accounts Detail

1.1.1 AWS Account Numbers

Table 1: AWS Account Numbers

Account #	IV Name	EDD Name	Purpose
318484875904	Shared	n/a	Contains all AWS CodePipeline, CodeBuild, and CodeCommit resources
90541811020	RND	n/a	Developers create components in the AWS Console
441184076731	Feature	n/a	Developers use this account to test their infrastructure as code and deployments
658456346551	Dev	n/a	Used for internal testing and validation
640079793484	QA	QA	Used for testing by Osaas
137271411486	QA-FIX	QA-FIX	Patch/bugfix environment fot QA
559450564007	Sandbox1	Sandbox1	R&D environment for EDD's IT department
123372620210	Sandbox2	Sandbox2	R&D environment for EDD's IT department
990703426931	Sandbox3	Sandbox3	R&D environment for EDD's IT department
414275752728	DIBT	DIBT	Training environment for call center workers
443054495397	Staging	UAT	Used for customer validation
635425321918	UAT	Performance	Used for performance testing
058264286667	UAT-FIX	UAT-FIX	Patch/bugfix environment fot UAT
559550955475	Prod	Prod	Contains production Connect instance
891376957702	Prod-FIX	Prod-FIX	Patch/bugfix environment for Prod

1.2 Repositories

This implementation is contained in 6 AWS CodeCommit repositories.

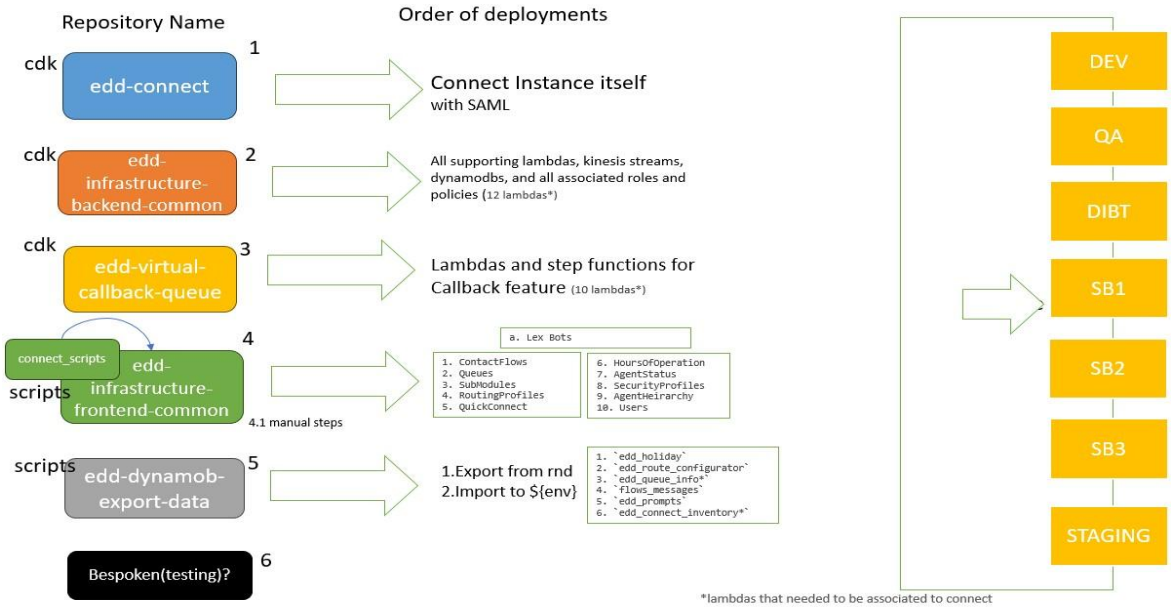

Figure 2: AWS CodeCommit Repositories

1.3 EDD Connect

The *edd-connect* repository contains the CDK application that deploys the base Connect instance.

1.4 Virtual Callback

This is an implementation of the AWS blog post Set up queued callback by creating flows, queues, and routing profiles.

1.5 Lambda Deployments and Supporting Resources

The *edd-infrastructure-backend-common* repository contains:

- Lambda code called by Lex and Connect
- Lambda code that extends CloudFormation via CloudFormation custom resources
- Typescript based Cloudformation Development Kit applications to deploy that Lambda code
- Build and deployment instructions in the buildspec file used to deploy the CloudFormation templates and the CDK application.

1.6 Connect and Lex Deployments

The Connect and Lex deployments use scripts contained in the connect-scripts repository.

These scripts can be used to copy any Connect implementation to another Connect instance.

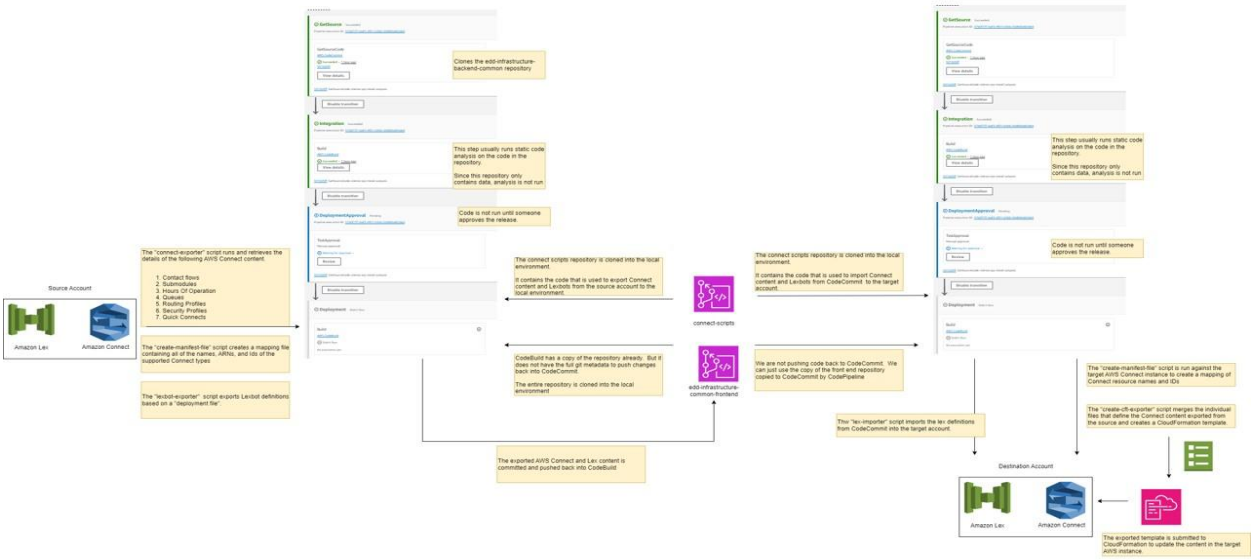

Figure 3: Connect and Lex Deployment Detail

1.7 Cross Account Code Pipeline

All Code and CodePipelines are stored in the shared account.

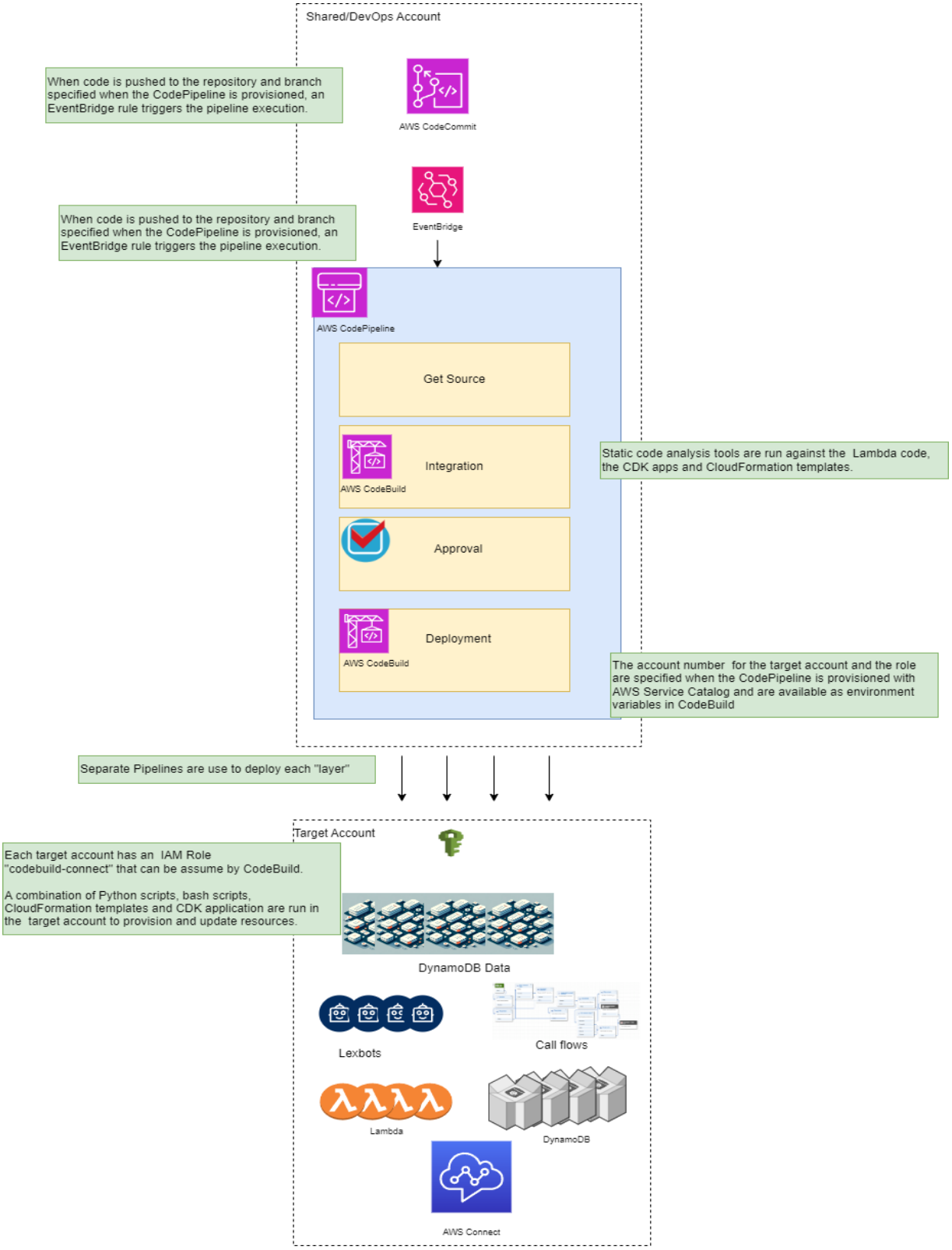

Figure 4: Cross Account Code Pipeline Detail

For the most part we use standard git processes to perform branching, pull requests and merges. However, the process we use to deploy Connect and Lex resources is different.

We have scripts controlled by a separate pipeline that export code directly from the AWS Connect instance and the Lex console to the dev environment.

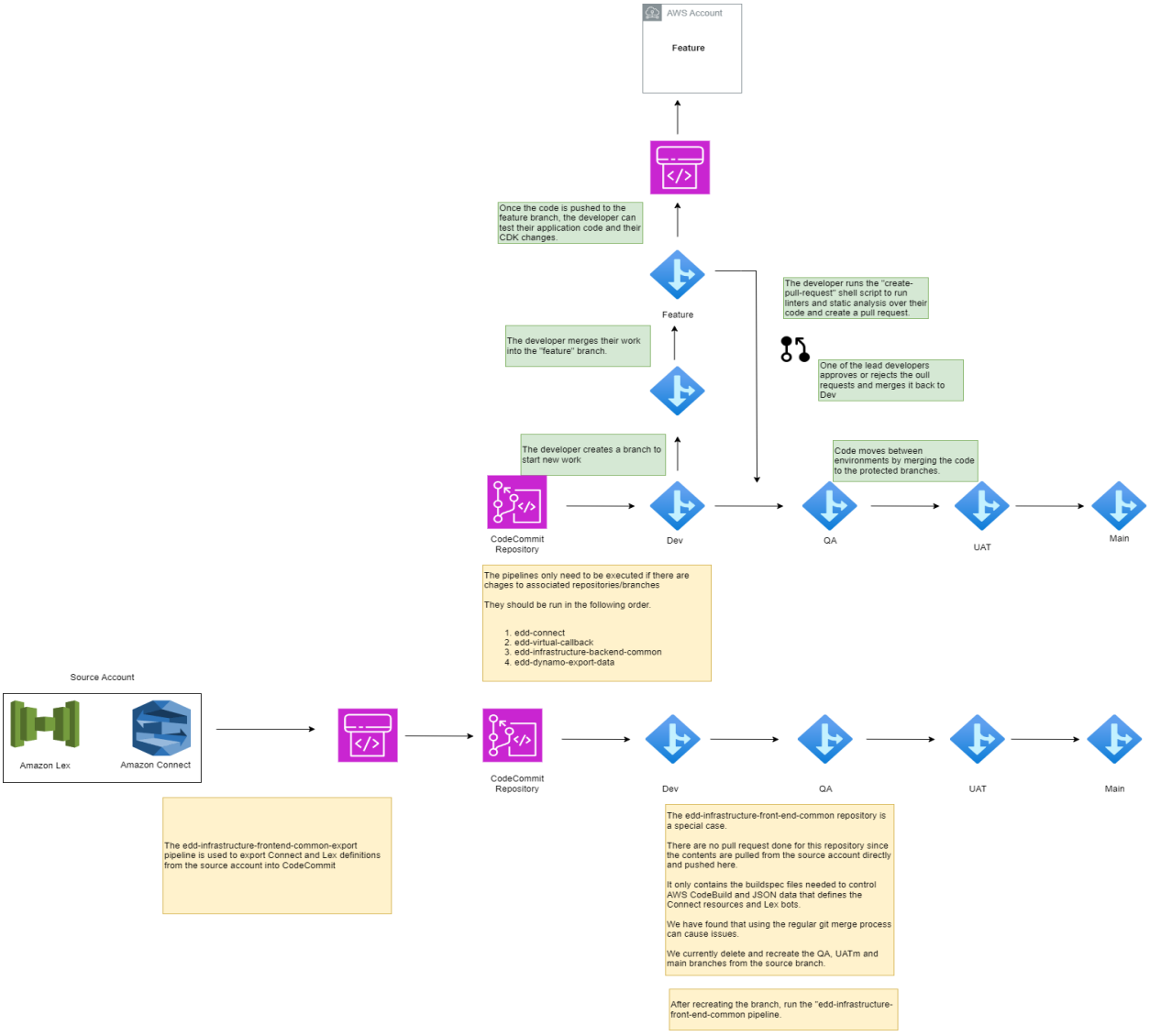

Figure 5: AWS Connect instance and Lex Console Export Pipeline

1.8 Provisioning a Pipeline

We use AWS Service Catalog to create our CodePipelines.

You can find more information in the Service Catalog CodePipeline README.

1.9 How Pipelines Are Used to Deploy Code

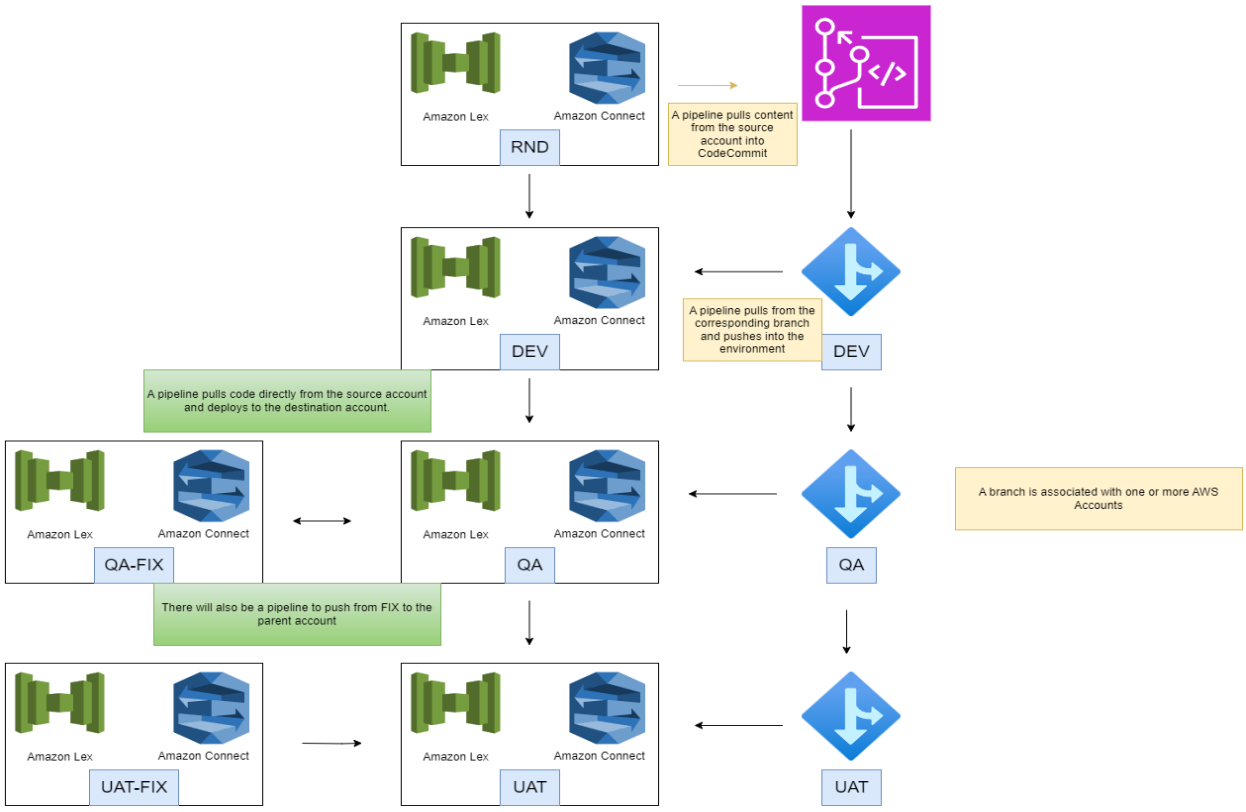

Figure 6: Code Deployment Pipeline

1.10 Creating a New Environment.

1.10.1 Add the Account to the Service Catalog Product

The name of the environment and account number need to be added to the Service Catalog Product.

Log in to the Shared Account ([redacted]) and edit the CodePipeline CloudFormation template.

Add the environment and the AWS Account number of the target account.

```
Parameters:
Repository:
    Type: String
    Description: Name of the CodeCommit repository that stores the application code.
  TargetAccount:
    Type: String
    Description: The destination account
    AllowedValues:
     - dev/[redacted]
     - feature/[redacted]
     - qa/[redacted]
```

After you make the change and save it, the associated CodePipeline will be initiated and the product will be updated.

The environment and account number values will be available as environment variables within CodeBuild - *$Environment* and *$AccountNumber*.

1.10.2 Create Roles in the Target Account

There are two roles required in the target account.

- *codebuild-connect* - for now this account should have the *Administrator Account* policy attached and allow assumed role trust from the DevOps account ([redacted]). This role is used by the CodePipelines to run commands in the account.
- *AWSServiceRoleForLexV2Bots_Common* - this role should have the *AmazonLexV2BotPolicy* attached and is used by imported Lexbots

1.10.3 Create Pipelines to Deploy to the Target Account

A pipeline needs to be created targeting the new account for each dependent repository.

- edd-connect - deploys the AWS Connect instance
- edd-infrastructure-backend-common - deploys the supporting AWS Lambdas and related infrastructure
- edd-virtual-callback-queue - supports scheduled callback
- edd-infrastructure-frontend-common - deploys the Lexbots and Connect content.
  - “full” - if a deployment type of full is specified, both the Lexbots and the Connect content will be deployed.
  - “partial” - if a deployment type of partial is specified, only the Connect content will be deployed.
- edd-dynamod-export-data - populates the target DynamoDB tables that were created as part of the backend common