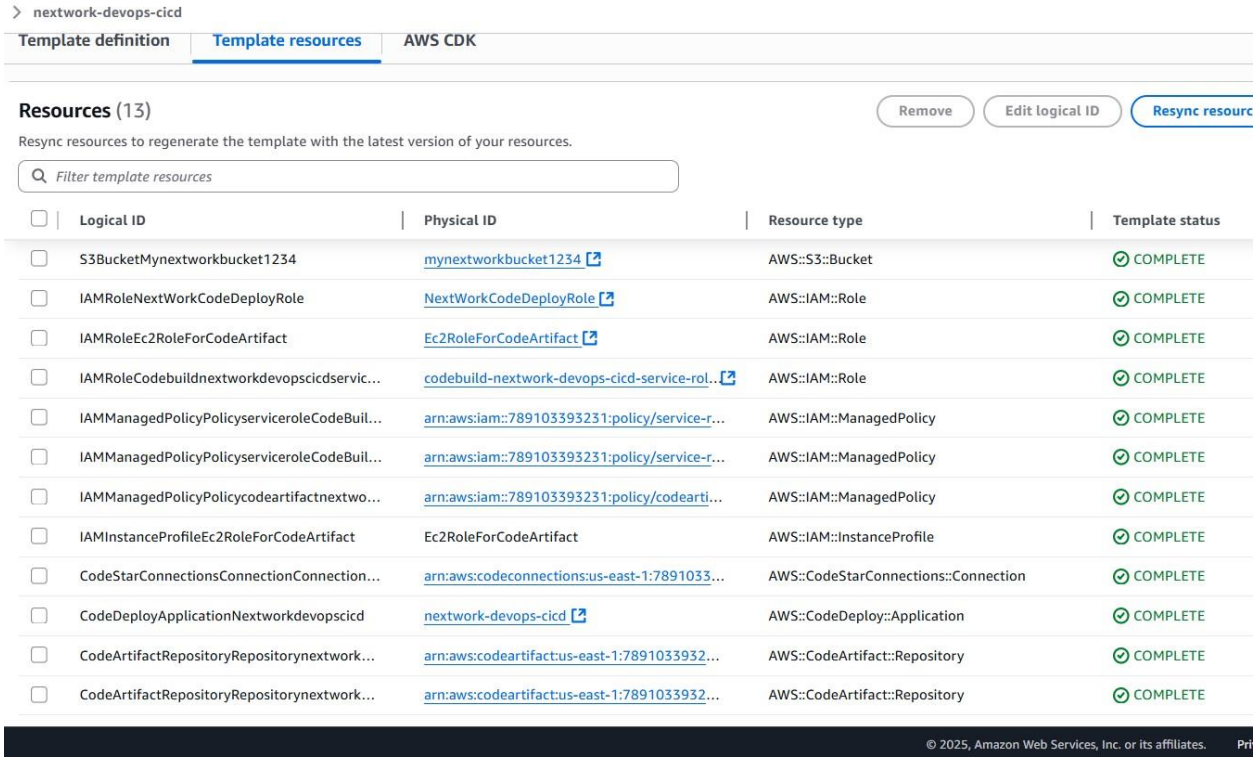
Gursimran Singh
NextWork Student

nextwork.org

Infrastructure as Code with CloudFormation



Introduction to the Project

This project demonstrates the setup of a comprehensive AWS architecture using a CloudFormation template. The primary objective was to leverage Infrastructure as Code (IaC) principles to define and provision a CI/CD pipeline. By the conclusion of this project, a suite of resources—including CodeArtifact, CodeBuild, CodeDeploy, and CodeStar Connections—were successfully codified into a single, deployable template.

Key Tools and Concepts

The implementation of this project relied on AWS CloudFormation and the AWS developer tools suite. The core concepts explored included the fundamentals of Infrastructure as Code, the practical application of the CloudFormation IaC generator, and strategies for resolving resource creation order issues. A key part of the project involved learning to troubleshoot deployment failures, such as dependency errors, and manually authoring resource definitions to extend the template's functionality.

Project Reflection

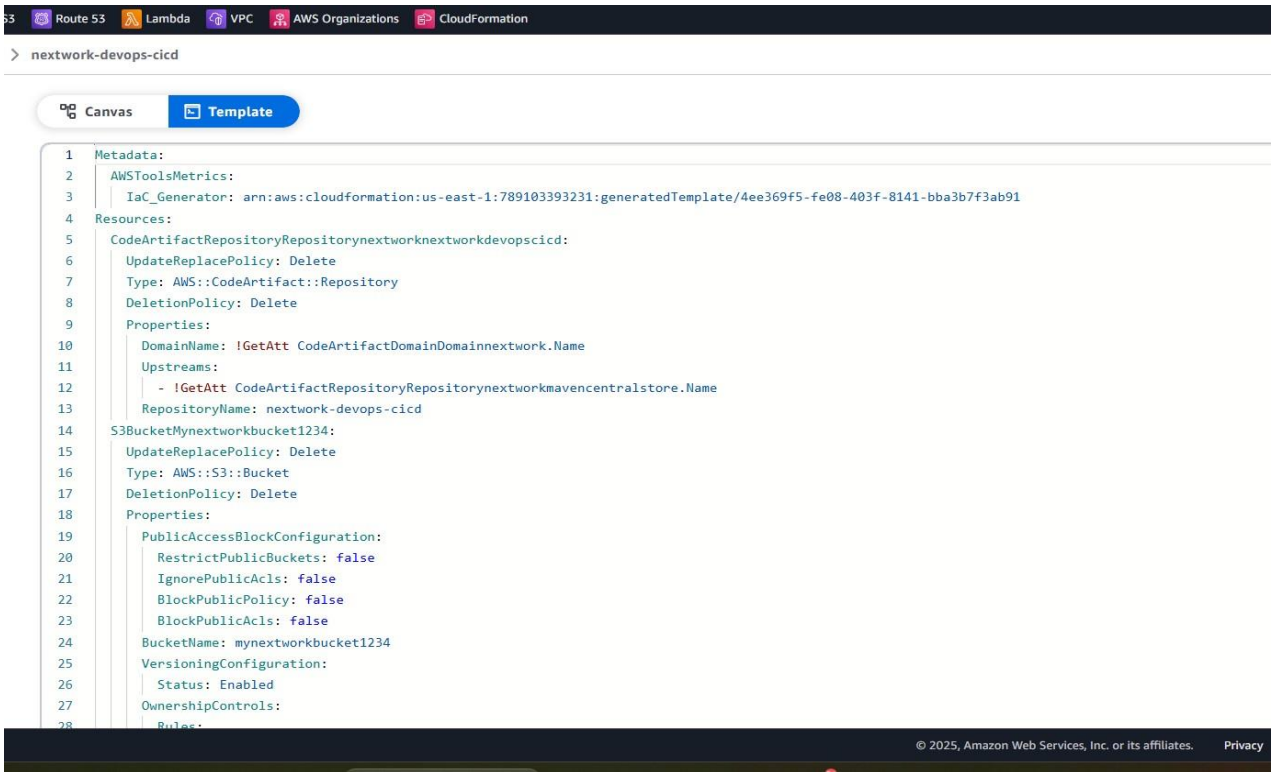
The project, including the demonstration and troubleshooting phases, was completed in approximately three hours. The most significant challenge was the process of editing the CloudFormation template, which required carefully identifying

the logical IDs of various resources to establish correct dependencies. The most rewarding aspect was successfully deploying the final, corrected template and observing the automated creation of all specified resources within the live AWS environment.

Generating a CloudFormation Template

The initial template was created using the IaC Generator, a feature within the CloudFormation console designed to accelerate template creation. This tool operates on a three-step process: it begins by scanning the existing resources in an AWS account, allows the user to select which resources to include in a new template, and finally provides the option to launch a new stack from that generated template.

The generated template successfully included the CodeArtifact domain and repositories, the CodeDeploy Application, various IAM roles and policies, and an S3 bucket. However, the IaC generator was unable to capture the CodeBuild project and the CodeDeploy deployment group. This limitation exists because these services have complex configurations that the generator cannot currently scan, necessitating that their definitions be written manually.



Template Testing and Initial Failures

Before the template could be tested, all existing resources that would be created by the template had to be manually deleted from the AWS account. This preventative step was necessary to avoid naming conflicts, which would cause the CloudFormation stack deployment to fail.

The first test deployment resulted in a

CREATE_FAILED status. An investigation of the events log revealed that the failure occurred because an IAM Policy resource attempted to attach to an IAM Role that had not yet been created. Since both resources were being created in the same template, a race condition occurred where the policy was ready before its corresponding role existed.

Implementing the "DependsOn" Attribute

To resolve the initial failure, the template was modified in a code editor. The solution was to add the

'DependsOn' attribute, which explicitly instructs CloudFormation to wait for a specific resource to be successfully created before attempting to create the resource that contains the attribute. This attribute was added to four different IAM policies associated with CodeBuild, ensuring they would only be created after their required IAM roles were available. In one instance, a policy for CodeArtifact required dependencies on two separate roles: the CodeBuild service role and the EC2 instance role.

```
    ComputePlatform: "Server"
IAMManagedPolicyPolicyserviceroleCodeBuildBasePolicynextworkdevopscicduseast1:
  UpdateReplacePolicy: "Delete"
  Type: "AWS::IAM::ManagedPolicy"
  DeletionPolicy: "Delete"
  DependsOn: "IAMRoleNextWorkCodeDeployRole"
  Properties:
```

Resolving Circular Dependencies

A second test of the updated template produced a new error: circular dependencies. This type of error occurs when two or more resources are mutually dependent on each other, creating a logical loop that CloudFormation cannot resolve.

An analysis of the template showed that the IAM Roles contained a

'ManagedPolicyArn' section that referenced the policies, while the IAM Policies now had a **'DependsOn'** attribute that referenced the roles. To break this loop, the

'ManagedPolicyArn' sections were removed from all role definitions within the template. This allowed the roles to be created first, after which the policies could be created and attached.

Manual Additions and Template Parameters

To complete the required infrastructure, the template was further extended by manually authoring definitions for two additional resources: a CodeBuild project and a CodeDeploy deployment group. During this process, it was critical to ensure that all internal resource references, such as the S3 bucket ID and IAM service role IDs, were updated to match the logical IDs of the resources defined within the template.

Additionally, the concept of

Parameters was introduced to increase the template's flexibility. Instead of hardcoding values, parameters were created for GitHub account information (account name and repository name), which prompts the user for these values at the time of stack creation.

```
# CodeBuild Project
CodeBuild00nextworkwebcloud formation00:
Type: AWS:: CodeBuild:: Project
DependsOn:
—
1
"IAMRole00codebuild nextworkdevopscicdservice role00QsCVG"
- "S3Bucket00nextworkdevopscicdnatasha0064qd0"
Properties:
Name: nextwork-devops-cicd
Description: Build project for NextWork web application Source:
Type: GITHUB
Location:
Sub "https://github.com/${GitHubRepoOwner}/${GitHubRepo}"
BuildSpec: buildspec.yml
Artifacts:
Type: S3
```

Successful Deployment

After all modifications and manual additions were complete, the CloudFormation stack was deployed one final time. The deployment succeeded, and the status of all resources showed

CREATE_COMPLETE. The successful creation of every resource, including the manually defined CodeBuild project, was verified by clicking the physical ID hyperlinks in the CloudFormation Resources tab and inspecting them in their respective service consoles.