CI/CD Pipeline using AWS CDK

# Overview

The purpose of this project is to build a pipeline that creates two stacks using CDK. First codepipeline stack and second application stack which creates ASG, RDS, LB, Route53 entry and app deployment using userdata on EC2 instances. The deployment is broken into 2 distinct phases, that can be modularized depending on the requirements. All of this can be configured in a central configuration file found in the application repository. Codepipeline stack is secluded from application pipeline to enable multiple application deployment using same codepipeline repo. This document is meant to give a high-level technical overview of what is contained in the project, specific implementation and deployment details can be found in the code’s README file.

The initial deployment needs access to central config file to setup the deployment pipeline which can be done by copying the file locally. Then the individual deployment of application is pulled from GITHUB which allows for multiple applications to be deployed from same codepipeline codebase. The Architecture is designed around Python app so build stage is setup after test.

# Features

* CDK/Codepipeline as separate GIT repo so we can use the same pipeline codebase with multiple applications.
* Separate Unit test scripts for each application residing in application repo.
* Centralized Yaml file for all application configurations.
* Integration with Slack
* Integration with Route53 to create A record for the app based on the domain name provided as part of the config file.
* RDS and ASG setup to enable high availability.

# Deployment

The initial deployment commands will trigger codepipeline creation and will subsequently trigger CDK stack for application gathering all required information from the central config file. Application stack deployment is completely automated. The one point of complication is understanding which environment variables will impact the CDK initial deployment. All other updates are handled by CodePipeline and will only be updated when changes are pushed to GITHUB. The rule of thumb between the two is anything impacting the pipeline will require a manual deploy, while anything being changed in application stack will be handled with the post git commit hook set up by CodePipeline.

# CodePipeline using CDK

The pipeline created by initially deploying CDK stack is show in the diagram below. The deployment is split in to 4 stages. Items that can be run in parallel are run in a single stage. Most of the steps are dependent on outputs from previous items so they do need to be run in parallel. If changes are made to the structure it is recommended to rebuild to avoid any issues.

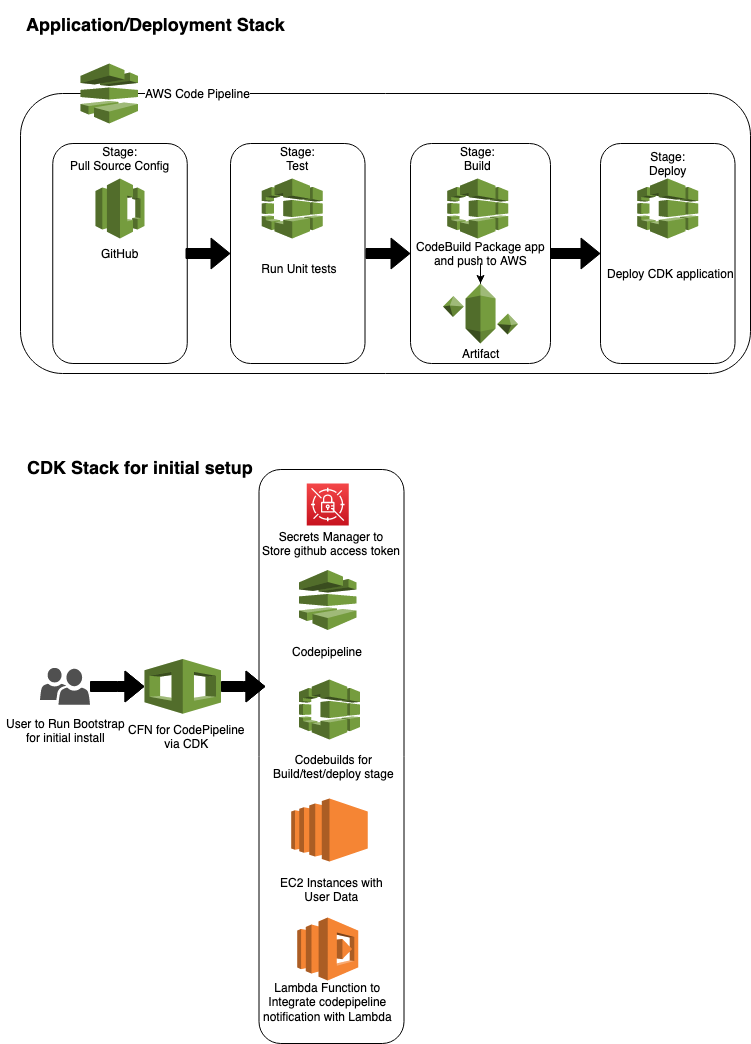


Figure 1 –Architecture Diagram

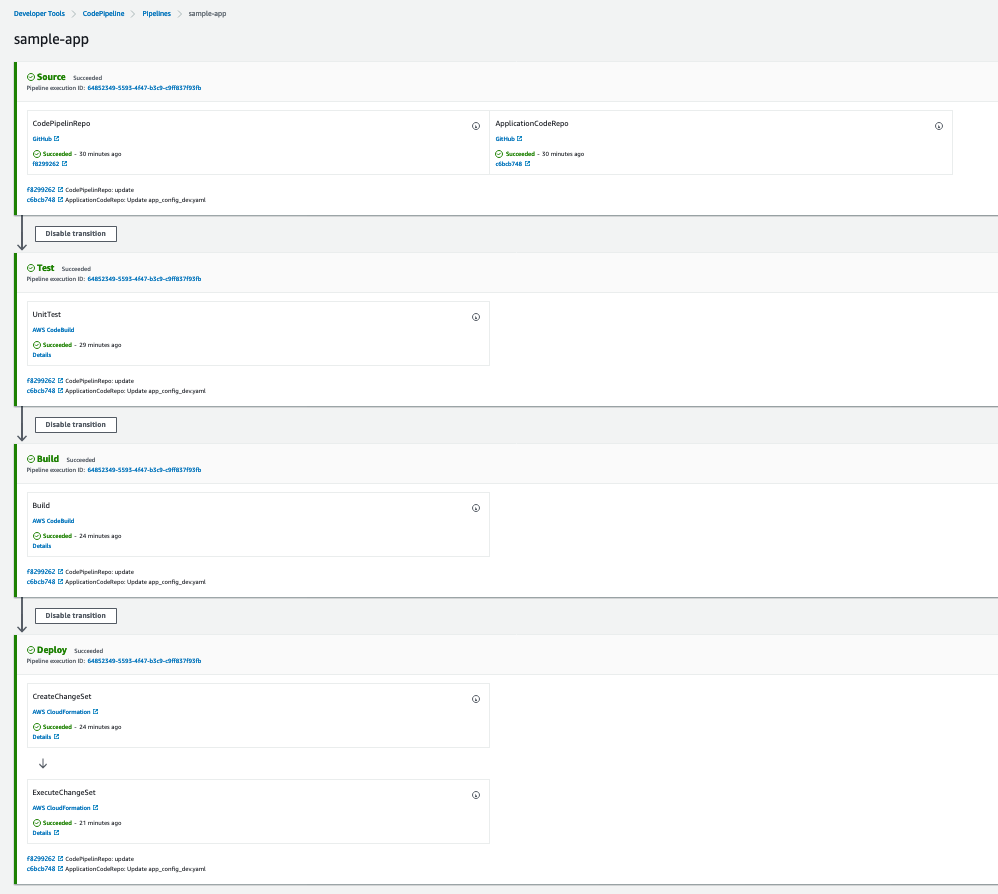
****

Figure 2 – CodePipeline

# Application Stack

Once Codepipeline is created successfully it will trigger application CDK stack creation using codebuild as part of the pipeline Build stage. Application stack consists on RDS instance, AutoScalingGroup(ASG), LoadBalancer(LB), Route53 Update and application deployment. All of these resources can be customized using central config file. Application deployment is done using userdata when ec2 instance is being created. Userdata file resides inside the application repo so it can be customized for different applications. Userdata also provides application teams option to using existing application deployment methods like existing ansible scripts to deploy the ec2 instances.