**CI/CD pipeline for Amazon ECS with GitHub Actions & AWS Code Build Tests**

**Creating ECS Infrastructure**

To build the architecture described in the solution overview, you will need the following ECS components:

* ECR Repository: store versioned application container images
* ECS Cluster: provides compute power to run application container instances
* ECS Task Definition: specifies application container image version and environment considerations
* ECS Service: specifies how task definition will be deployed onto underlying compute resources

To build this infrastructure, we will be using the [AWS Cloud Development Kit (CDK)](https://aws.amazon.com/cdk/) and Python 3.7.

**Source Code:**

[**https://github.com/bhupathis/ecs\_project.git**](https://github.com/bhupathis/ecs_project.git)

**To create our application infrastructure:**

1. [Configure your AWS CLI](https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-configure.html) with an IAM user that has permissions to create the resources (VPC, ECS, ECR, IAM Role) described in the template below.

2. Install the ‘cdk’ using nodejs.

Run the following commands to initialize your CDK project

# Create a project directory

mkdir ecs-devops-sandbox-cdk

# Enter the directory

cd ecs-devops-sandbox-cdk

# Use the CDK CLI to initiate a Python CDK project

cdk init --language python

# Activate your Python virtual environment

# NOTE: For Windows users, replace with ".env\Scripts\activate.bat"

source .env/bin/activate

# Install CDK Python general dependencies

pip install -r requirements.txt

# Install CDK Python ECS dependencies

pip install aws\_cdk.aws\_ec2 aws\_cdk.aws\_ecs aws\_cdk.aws\_ecr aws\_cdk.aws\_iam

3. Replace the contents of the file ecs\_devops\_sandbox\_cdk/ecs\_devops\_sandbox\_cdk\_stack.py (automatically created by the CDK) with the code below

<https://github.com/bhupathis/ecs_project/blob/main/ecs_devops_sandbox_cdk_stack.py>

4. Run the following command from the root directory of your CDK project

# Create the CloudFormation stack

cdk deploy

The end result of this step should be a Cluster, ecs-devops-sandbox-cluster, running a Service, ecs-devops-sandbox-service, that consists of a Task Definition, ecs-devops-sandbox-task-definition.

**Create a Code Build Project to execute our application tests:**

CodeBuild is used to execute our application tests and provide the status of these tests to GitHub.  If the tests do not pass, CodeBuild marks the build as Failed and this status is reported to GitHub.  In our architecture, we rely on webhooks that are automatically created by CodeBuild to trigger a build of our application code every time there is a commit to the master branch of the GitHub repository.

To create a CodeBuild Project with GitHub integration:

1.     Navigate to AWS CodeBuild and select Create build project

2.     Under Project Configuration, for Project name, enter ecs-devops-sandbox

3.     Under Source, for Source Provider, select GitHub

4.     Under Source, for Repository, select Connect using OAuth and select Connect to GitHub

5.     At the pop-out window, log into the GitHub account that owns the repository you wish to use

6.     Under Source, for Repository, select Repository in my GitHub account

7.     Under Source, for GitHub repository, select the repository you configured earlier

8.     Under Source – Additional configuration, for Build Status – *optional*, check the box

9.     Under Primary source webhook events, for Webhook – *optional*, check the box

10. Under Environment, for Environment image, select Managed image

11. Under Environment, for Operating system, select Ubuntu

12. Under Environment, for Runtime, select Standard

13. Under Environment, for Image, select aws/AWS CodeBuild/standard:4.0

14. Leave remaining default values and select Create build project

**Create a GitHub Workflow with Actions:**

AWS has provided a starter GitHub workflow that takes advantage of the AWS open-source GitHub Actions to build and deploy containers on ECS for each commit to master branch of the repository.

To add the starter GitHub workflow to your GitHub repository:

1.     Under the Actions tab, select New workflow

2.     Search for Deploy to Amazon ECS and select Set up this workflow

3.     Follow the instructions outlined in the starter workflow file for using this workflow with your application.  Note that we have already created an ECR repository, ECS task definition, ECS cluster, and ECS service

<https://github.com/bhupathis/ecs_project/blob/main/.github/workflows/aws.yml>

**Results:**

With the AWS CodeBuild project and GitHub workflow in place, we can add new features to our application that automatically deploy to Amazon ECS if all tests are passed.

This example shows a modification to the application code (app.py) that does not pass the unit test.

<https://github.com/bhupathis/ecs_project/blob/main/app.py>

When this code is checked into the repository, we can see that Code Build is automatically invoked via webhook and executes the application tests.

======================================================================