INTRODUCTION

We will deploy a demo application on ECS Fargate and update the application using the Blue/Green deployment capability of CodeDeploy and ECS.

Blue/Green deployment is a technique for releasing an application by shifting the traffic between two identical environments running different versions of the same application. Blue/Green deployment is recommended for critical workloads since it mitigates the risks associated with deploying software, such as downtime and rollback capability.

Traditionally, with in-place upgrades, it was difficult to validate your new application version in a production deployment while also continuing to run your old version of the application. After you deploy the green environment, you have the opportunity to validate it. If you discover the green environment is not operating as expected, there is no impact on the blue environment. You can route traffic back to it, minimising impaired operation or downtime, and limiting the blast radius of impact.

This ability to simply roll traffic back to the still-operating blue environment is a key benefit of blue/green deployments. You can roll back to the blue environment at any time during the deployment process.

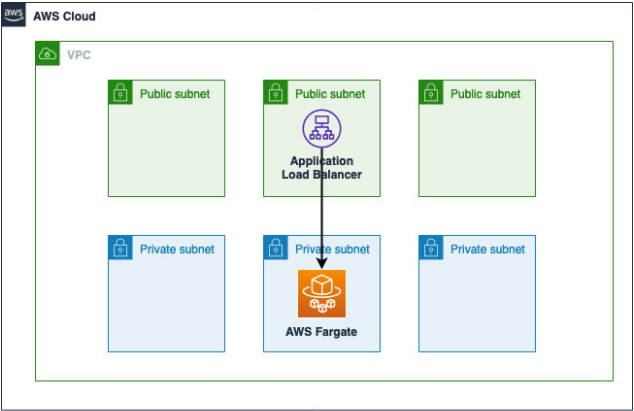
We will be deploying a demo application to ECS Fargate.

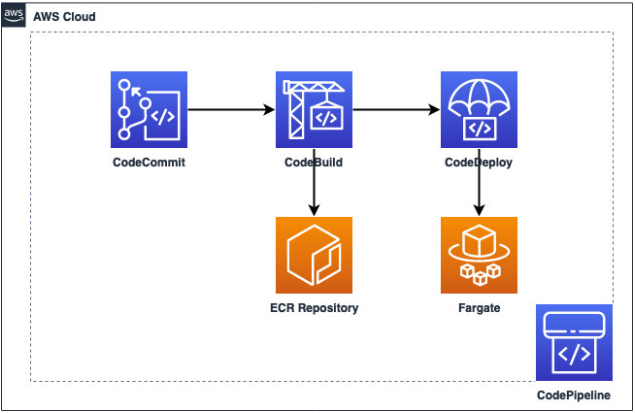
• This application will be a static web page running on NGINX as a Fargate service.

• CodePipeline will be used for executing Blue/Green deployment using CodeCommit, CodeBuild and CodeDeploy

• The container images will be stored in the Elastic Container Registry.

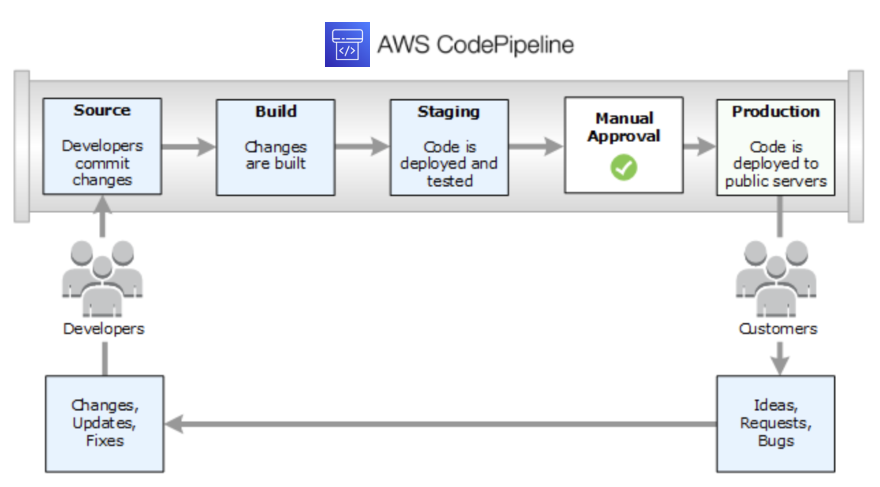
Below is a diagram of the environment we will be building:



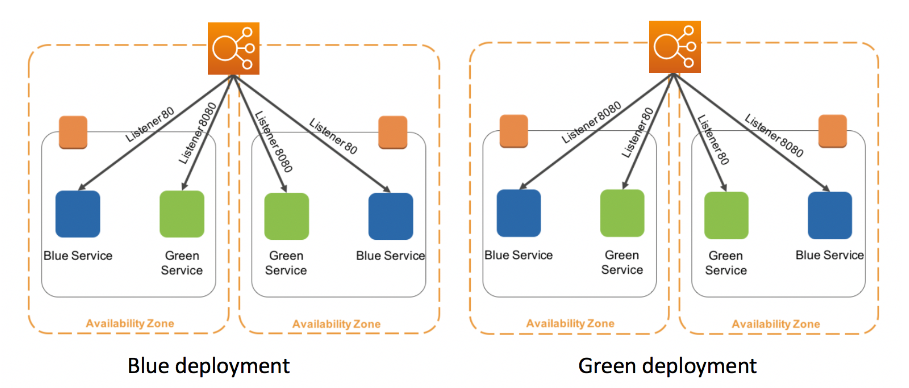


**CODEPIPELINE PRIMER:**

AWS CodePipeline is a fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates. CodePipeline automates the build, test, and deploy phases of your release process every time there is a code change, based on the release model you define.



With a blue/green deployment, you provision a new set of containers on which [CodeDeploy](https://aws.amazon.com/codedeploy)  install the latest version of your application. CodeDeploy then reroutes load balancer traffic from an existing set of containers running the previous version of your application to the new set of containers running the latest version. After traffic is rerouted to the new containers, the existing containers can be terminated. Blue/green deployments allow you to test the new application version before sending production traffic to it.

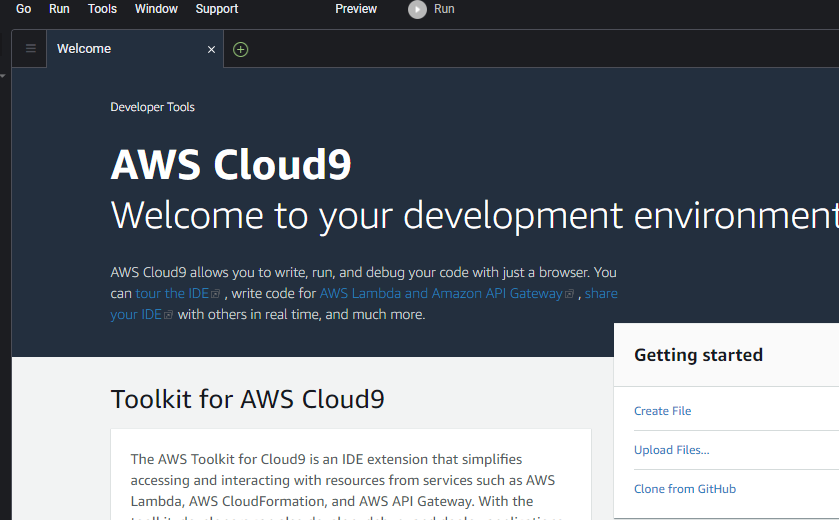


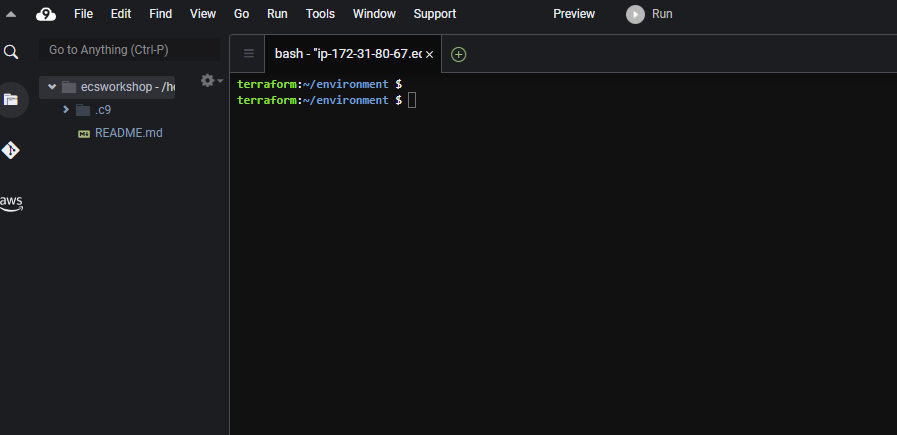
If there is an issue with the newly deployed application version, you can roll back to the previous version faster than with in-place deployments. Additionally, the containers provisioned for the blue/green deployment will reflect the most up-to-date server configurations since they are new.

**4 CREATE A WORKSPACE**

**4.1 Create AWS Cloud9 Environment**

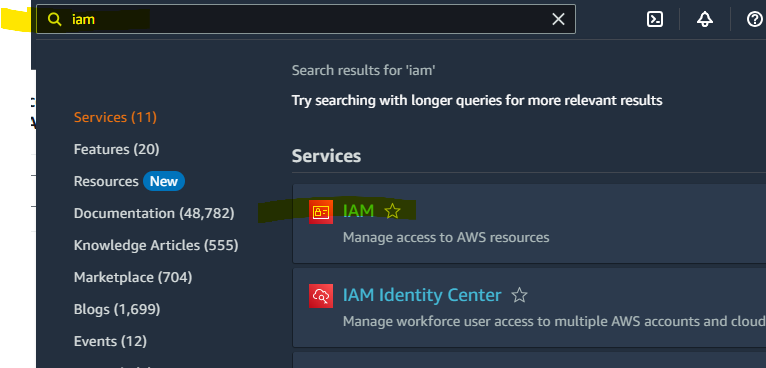
* Sign in to the AWS Management Console and open [*Cloud9 Environment*](https://console.aws.amazon.com/cloud9/home)
* Select **Create environment**
* Name it *ecsworkshop*  
  , and give meaningful **Description**
* Under **Environment type**, select **New EC2 instance**.
* Under New EC2 instance, select Additional instance types and **t3.large** Instance type.
* Stick with rest of the default settings, and select **Create**
* When it comes up, customise the environment by closing the **welcome tab** and **lower work area**, and opening a new **terminal** tab in the main work area:



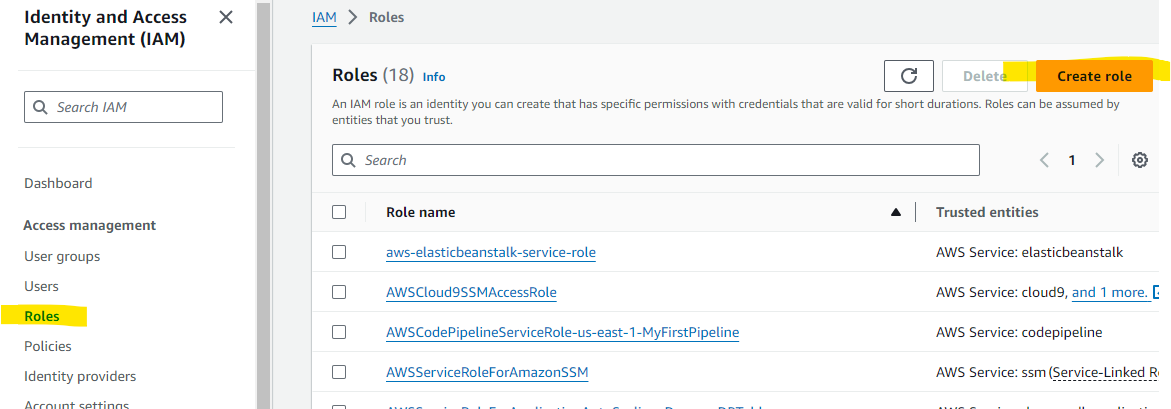


**4.2 Create the IAM Role**

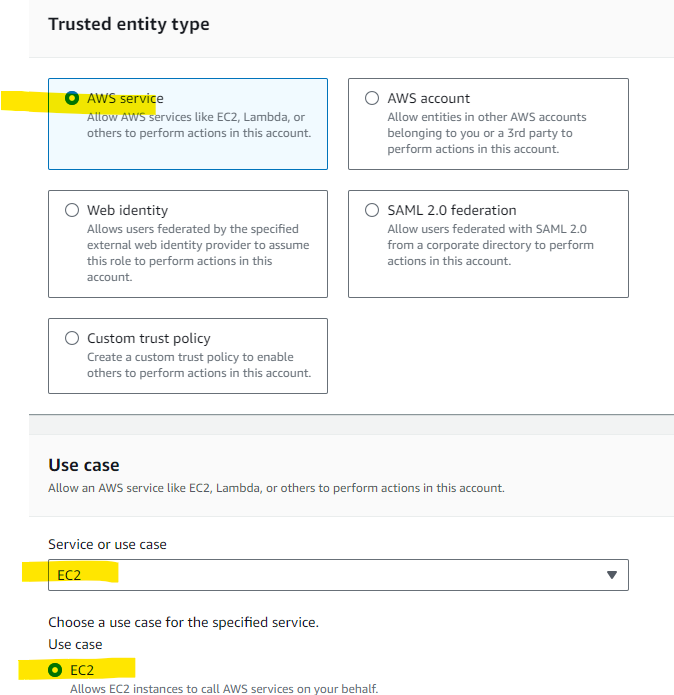
1. Search for IAM and click on it.



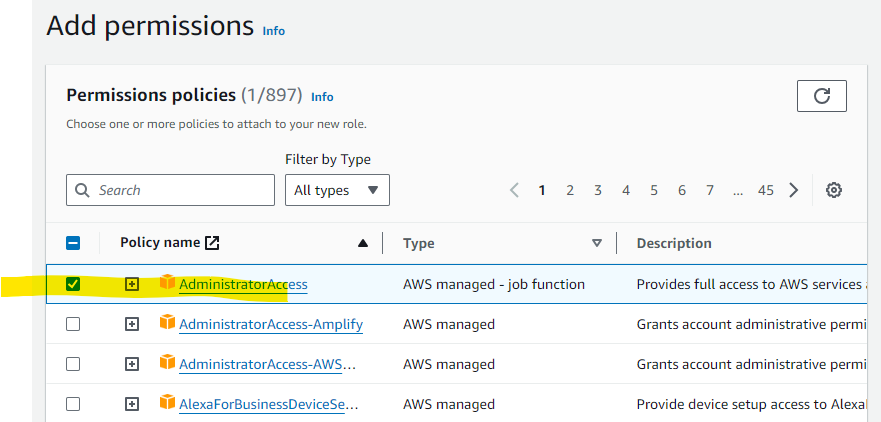
2. From the console dashboard, choose Roles and then click on create Roles.



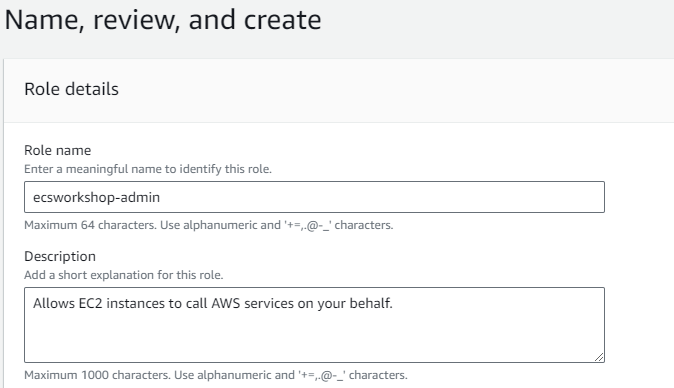
3. Under Select type of trusted entity, select AWS service. Under Choose a use case, select EC2, and then choose Next.



4. Search for Administrator Access, select it and click on Next.

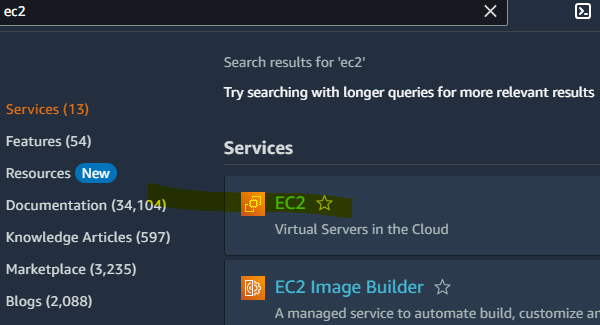


5. Enter a name for the role ( *ecsworkshop-admin*), and then choose Create role.

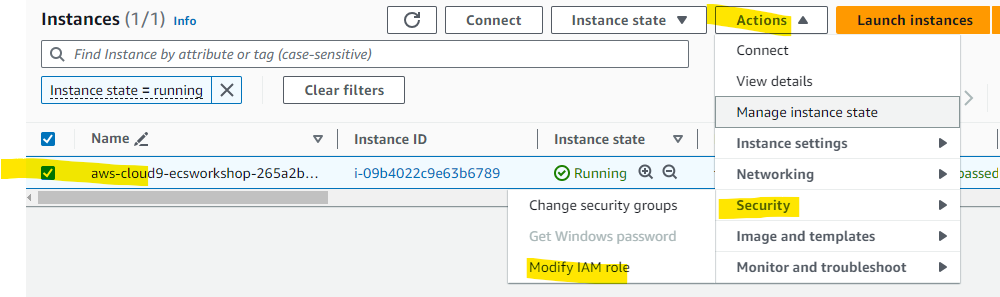


**4.3 Attach IAM Role to Cloud9 EC2 Instance**

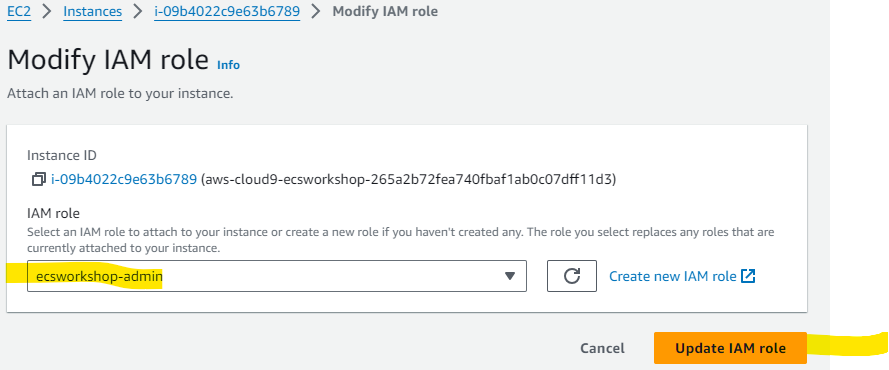
1. Search for EC2 and click on it.



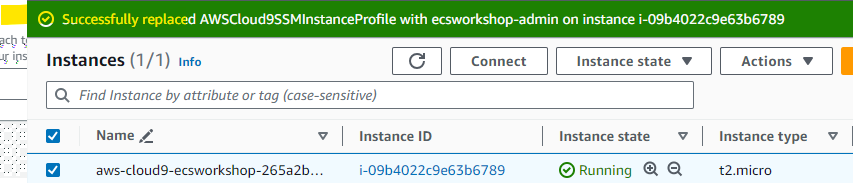
2. Select the EC2 Instance and click on the Action-> Security-> Modify IAM role



3. Select the IAM Role created in the earlier steps and click on Update IAM role.

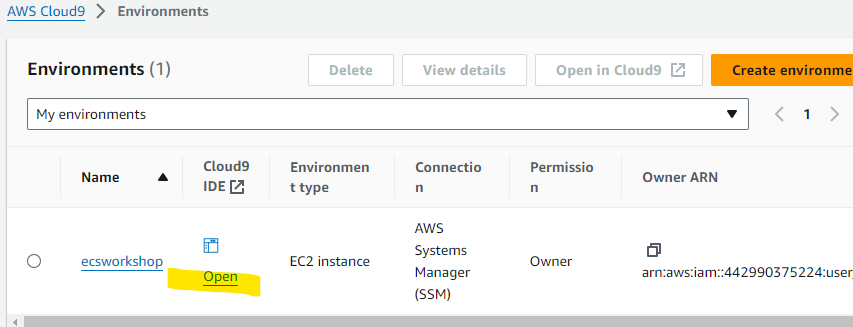


4. IAM Role Updated Successfully



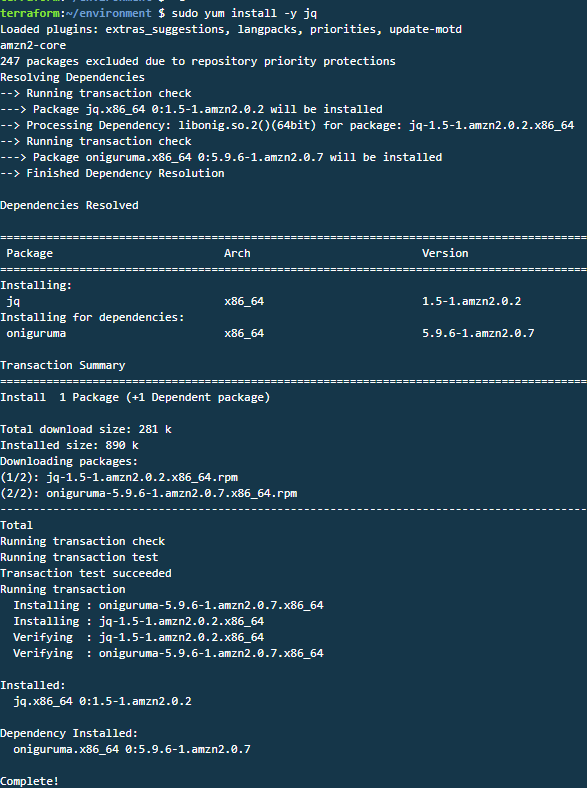
**5 BUILD ENVIRONMENT**

1. Back to the Cloud9 tab, click on **Open**.



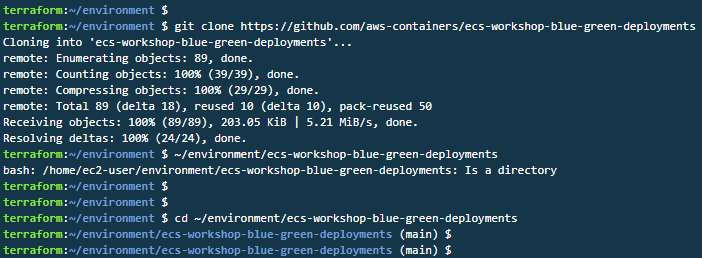
2. Run the following command in the terminal to Install the json.

*sudo yum install -y jq*

**

3. To clone the Git Repository & navigate to it, run the following command.

*git clone https://github.com/aws-containers/ecs-workshop-blue-green-deployments ~/environment/ecs-workshop-blue-green-deployments*

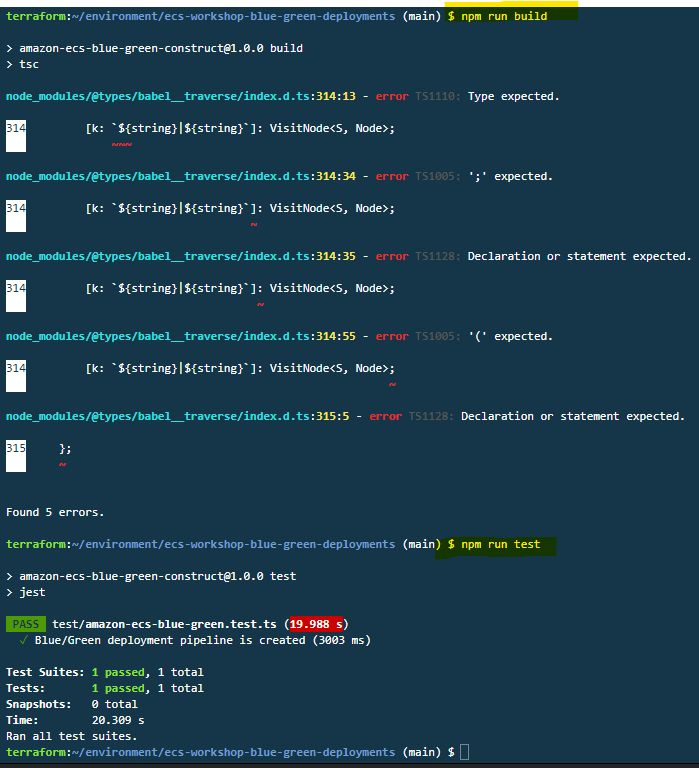
**

4. Run the following command to build the stack.

*npm install*

*npm run build*

*npm run test*

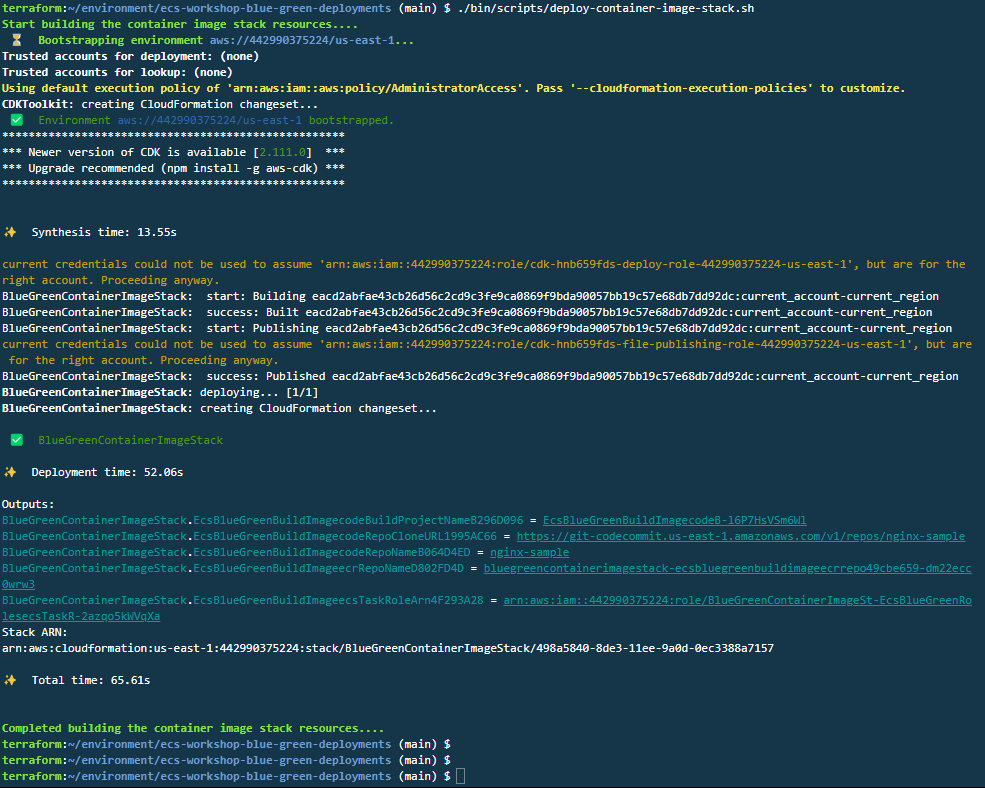
**

5. Run the following command to Deploy the container image stack

• This script bootstraps the CDK in the AWS account.

• Build a CodeCommit repository and a Codebuild project to create container image.

*./bin/scripts/deploy-container-image-stack.sh*

**

6. Run the following commands to push the nginx code sample into CodeCommit repository.

*export AWS\_DEFAULT\_REGION=$(aws configure get region)*

*export CODE\_REPO\_NAME=nginx-sample*

*export CODE\_REPO\_URL=codecommit::$AWS\_DEFAULT\_REGION://$CODE\_REPO\_NAME*

*cd ~/environment && git clone $CODE\_REPO\_URL && cd $CODE\_REPO\_NAME*

*cp ~/environment/ecs-workshop-blue-green-deployments/nginx-sample/\* .*

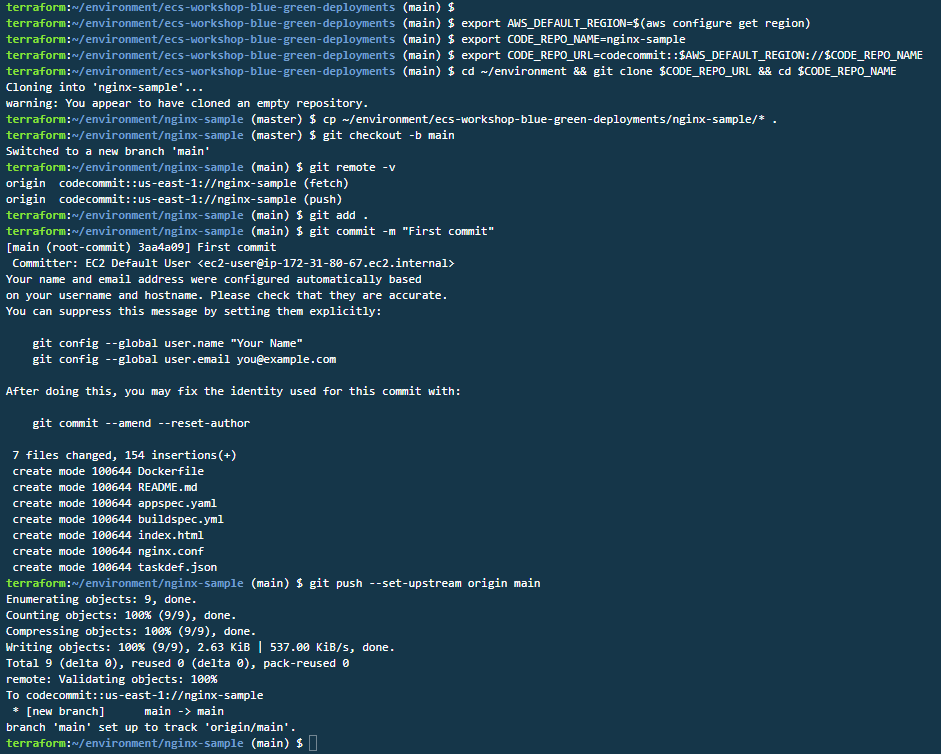
*git checkout -b main*

*git remote -v*

*git add .*

*git commit -m "First commit"*

*git push --set-upstream origin main*

**

7. Run the following command to Deploy the pipeline stack. This script executes below steps:

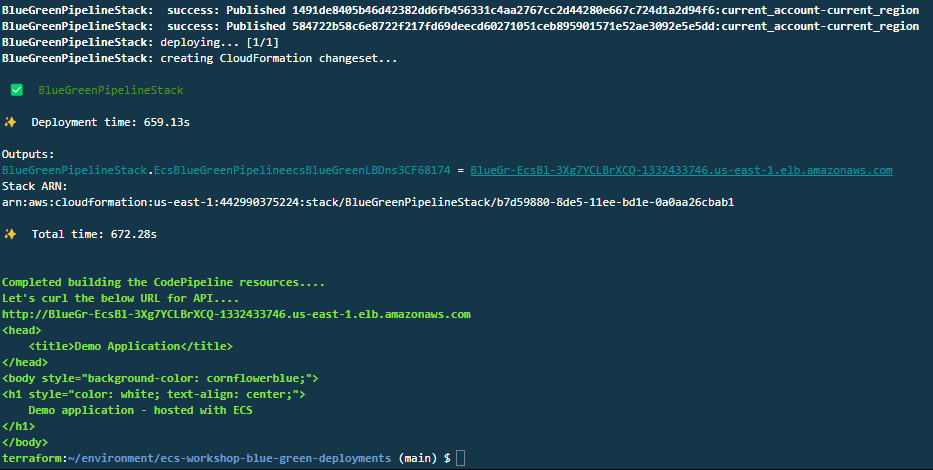
• Build the container image for code in CodeCommit using CodeBuild.

• Deploy the CodeDeploy and CodePipeline resources for blue/green deployment.

• Deploy the AWS Fargate service using the container image.

*cd ~/environment/ecs-workshop-blue-green-deployments*

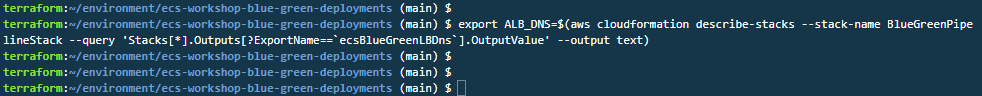
*./bin/scripts/deploy-pipeline-stack.sh*

**

Note: This step will take 10-15 minutes to complete.

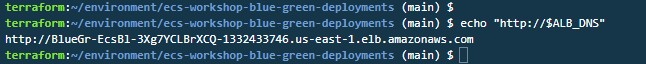
8. Run the following command to export the Load Balancer URL.

*export ALB\_DNS=$(aws cloudformation describe-stacks --stack-name BlueGreenPipelineStack --query 'Stacks[\*].Outputs[?ExportName==`ecsBlueGreenLBDns`].OutputValue' --output text)*

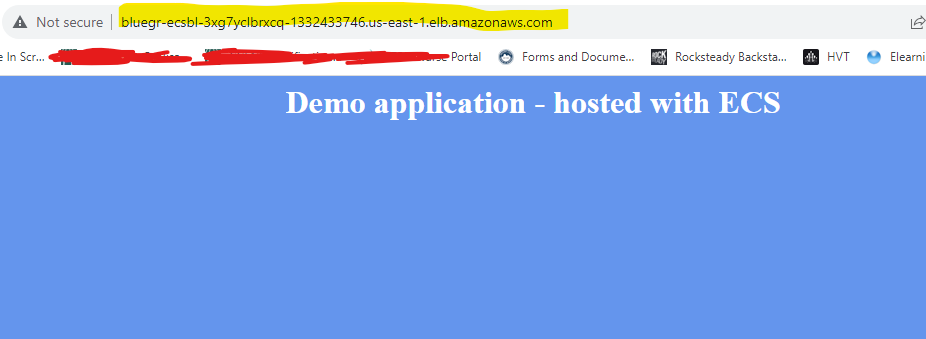
**

9.Run the following command to access the deployed blue version via the load balancer URL.

*echo "http://$ALB\_DNS"*



10.Copy and paste the URL in the New Tab and the endpoint will show the version of the application that you are deployed (which will be blue)



**6 UPDATE CODECOMMIT REPOSITORY**

1. Run the following command to Edit the index.html to change the background colour to green.

*cd ~/environment/nginx-sample/*

*vim index.html*

press I to edit the file and replace the below mentioned code with the existing one.

*<head>*

*<title>Demo Application</title>*

*</head>*

*<body style="background-color: green;">*

*<h1 style="color: white; text-align: center;">*

*Demo application - hosted with ECS*

*</h1>*

*</body>*

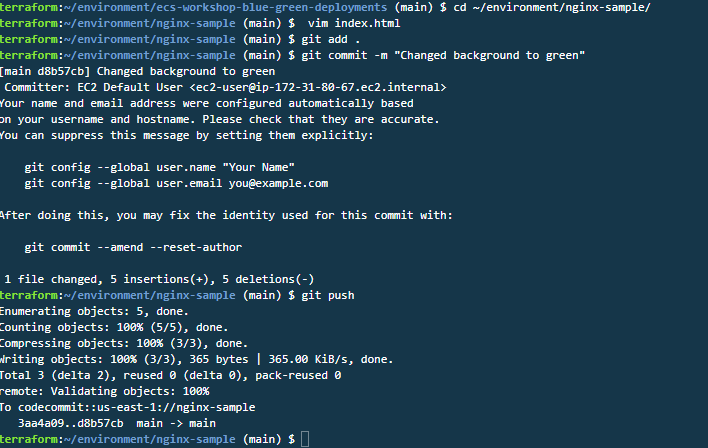
12. Press the Esc key, type ":wq" and press enter to exit the index.html file

13. Run the following command to push the code to the CodeCommit repository.

*git add .*

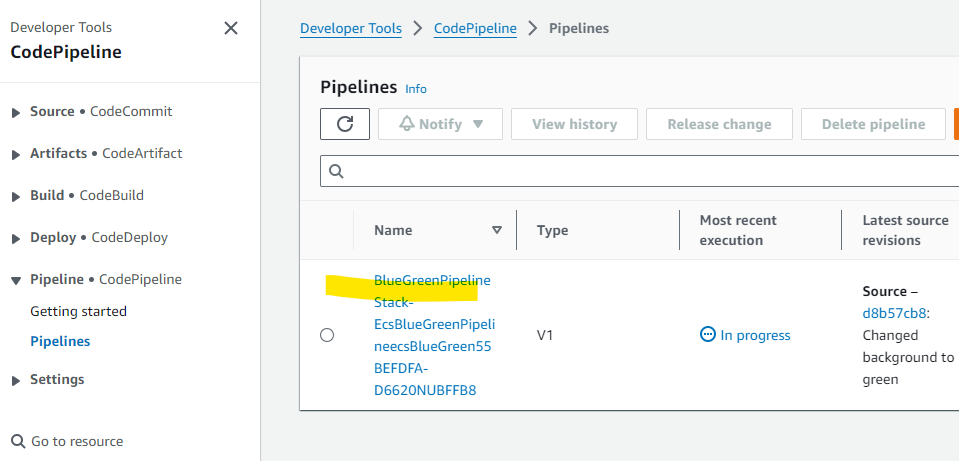
*git commit -m "Changed background to green"*

*git push*

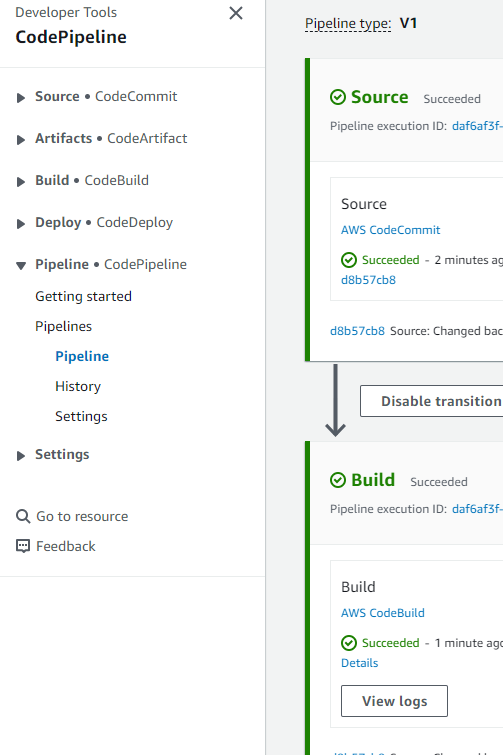
**

14. Search for CodePipeline in the AWS Console search bar and click on it.

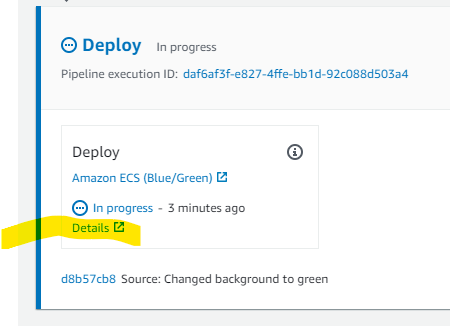
15. Click on the Environment Name.



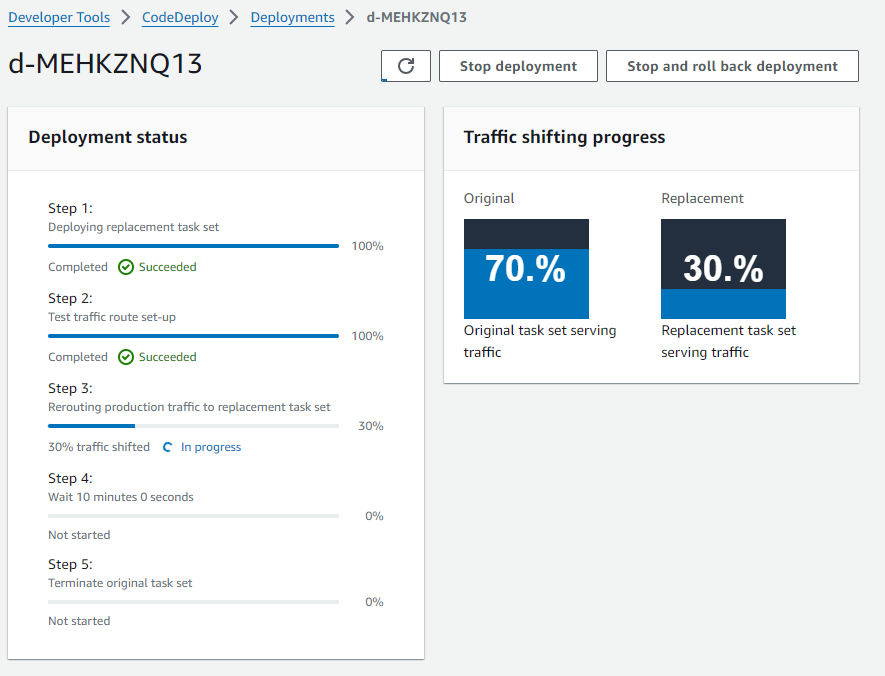
16.Here you can view the stages of the Codepipeline.



17. Click on Details.



18. Here you can see the Deployment in CodeDeploy.

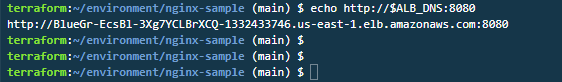


• **Step 1:** will deploy the replacement task set based on the taskdef.json and new image created by the build phase

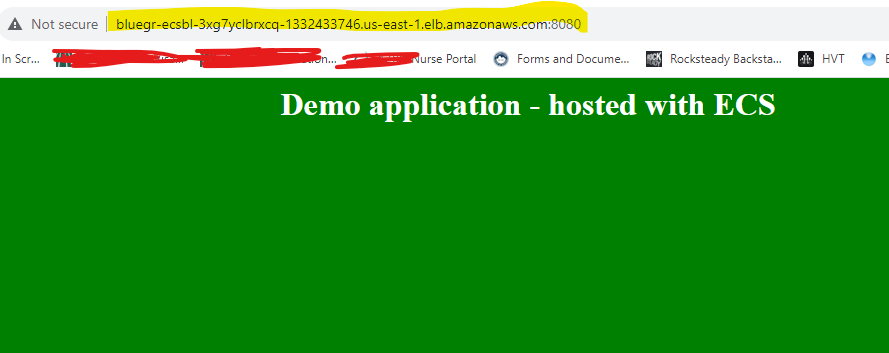
• **Step 2:** will setup the new tasks and enable the test listener on the port 8080

19.Once step 2 is completed, open the service in your browser on the Load Balancer Test listener port 8080 by using the following command.

*echo* [*http://$ALB\_DNS:8080*](about:blank)

**

20.Copy the URL and paste in the browser and you will see the demo application with a changed background colour of green.



**Step 3:** initiates the traffic shifting from Blue to Green deployment.

a) We have a seamless traffic shifting from blue to green using the deployment configuration - CodeDeployDefault.ECSLinear10PercentEvery1Minutes. We shift 10 percent of traffic every minute until all traffic is shifted.

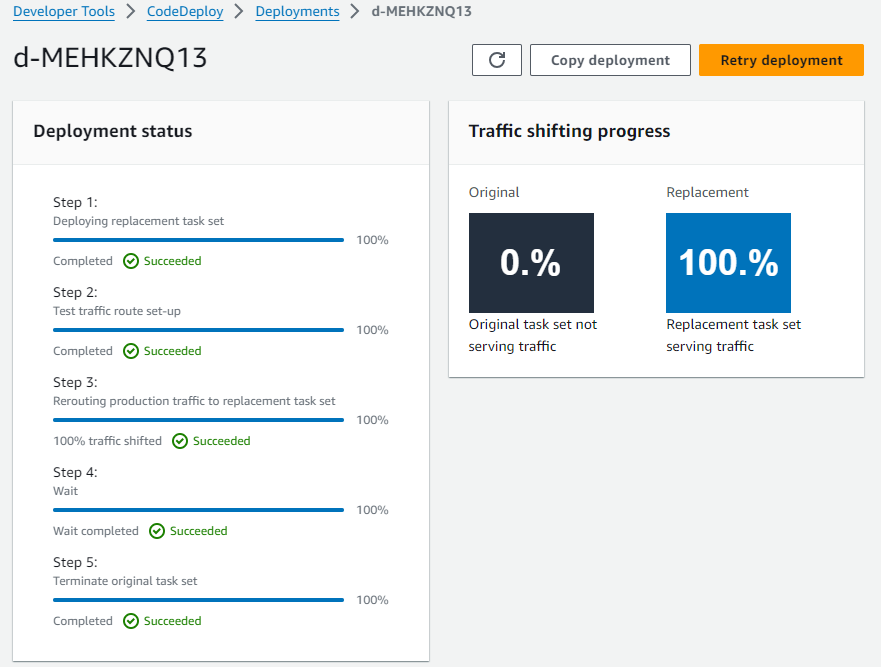
b) Once completed, port 80 will display the green deployment.

c) Run the command to get the URL:

*echo "*[*http://$ALB\_DNS*](about:blank)*"*

**Step 4** is where the old task set is retained for 10 minutes. This time period is configurable via the CDK stack.

**Step 5** will terminate the original task set.



It will take 10 – 15 minutes

We have completed a successful blue/green deployment.

**9 SUMMARY**

We had deployed a demo application to ECS Fargate.

• This application is a static web page running on NGINX as a Fargate service.

• CodePipeline is used for executing Blue/Green deployment using CodeCommit, CodeBuild and CodeDeploy

• The container images are stored in the Elastic Container Registry