

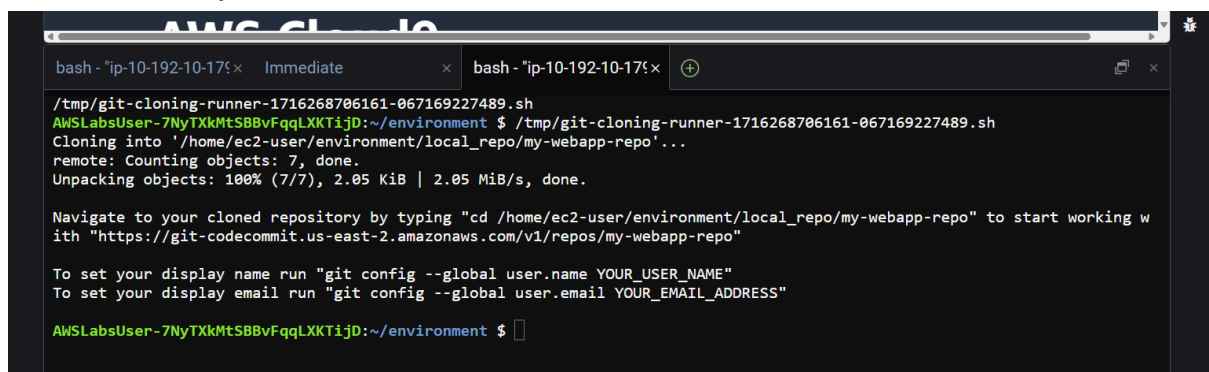
Lab 5: Performing Blue/Green Deployments with CI/CD Pipelines and Amazon ECS

Objectives: Task unclear to me - ECS, and how to configure it.

1. Build custom container images using AWS CodeBuild and store them in Amazon ECR.
2. Set up a CI/CD to build and automate application container image creation using AWS CodePipeline.
3. Host and run a containerized web application using Amazon ECS and AWS Fargate.
4. Configure AWS CodeDeploy to perform a blue/green deployment.
5. Set up a CI/CD to automate the build and deployment of a containerized web application.

Objective 1 - Set up the source files

1.1 Open Cloud9 from URL in the lab, and source files will be updated automatically



```
bash - "ip-10-192-10-17" x Immediate x bash - "ip-10-192-10-17" x +
/tmp/git-cloning-runner-1716268706161-067169227489.sh
AWSLabsUser-7NyTXkMtSBBvFqQLXKTjD:~/environment $ /tmp/git-cloning-runner-1716268706161-067169227489.sh
Cloning into '/home/ec2-user/environment/local_repo/my-webapp-repo'...
remote: Counting objects: 7, done.
Unpacking objects: 100% (7/7), 2.05 KiB | 2.05 MiB/s, done.

Navigate to your cloned repository by typing "cd /home/ec2-user/environment/local_repo/my-webapp-repo" to start working w
ith "https://git-codecommit.us-east-2.amazonaws.com/v1/repos/my-webapp-repo"

To set your display name run "git config --global user.name YOUR_USER_NAME"
To set your display email run "git config --global user.email YOUR_EMAIL_ADDRESS"

AWSLabsUser-7NyTXkMtSBBvFqQLXKTjD:~/environment $
```

Objective 2 - Use AWS CodeBuild and Docker to build your web application image and push the image to Amazon ECR.

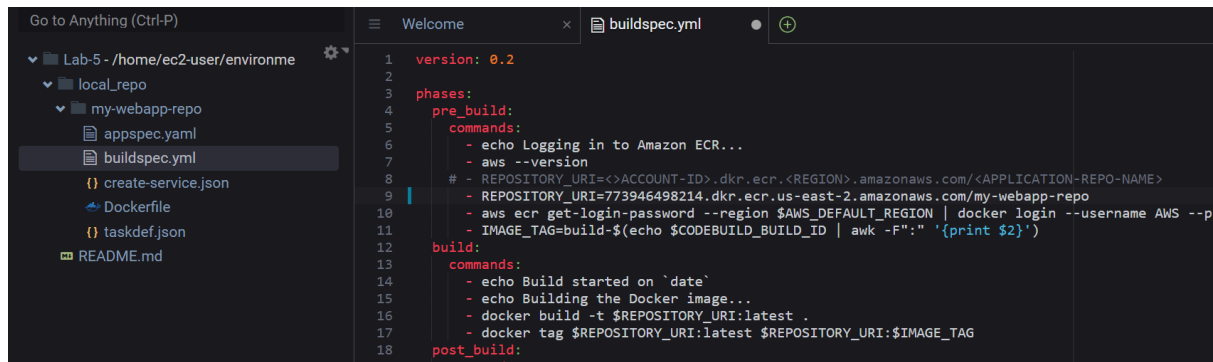
The Docker image built is based on the Dockerfile provided to you, which installs all basic dependencies needed for your application to host a simple web page in the blue background color.

2.1 There is a buildspec.yml file to your source code repository to tell CodeBuild how to build the application image.

The buildspec.yml file tells AWS CodeBuild how to build and push your Docker image:

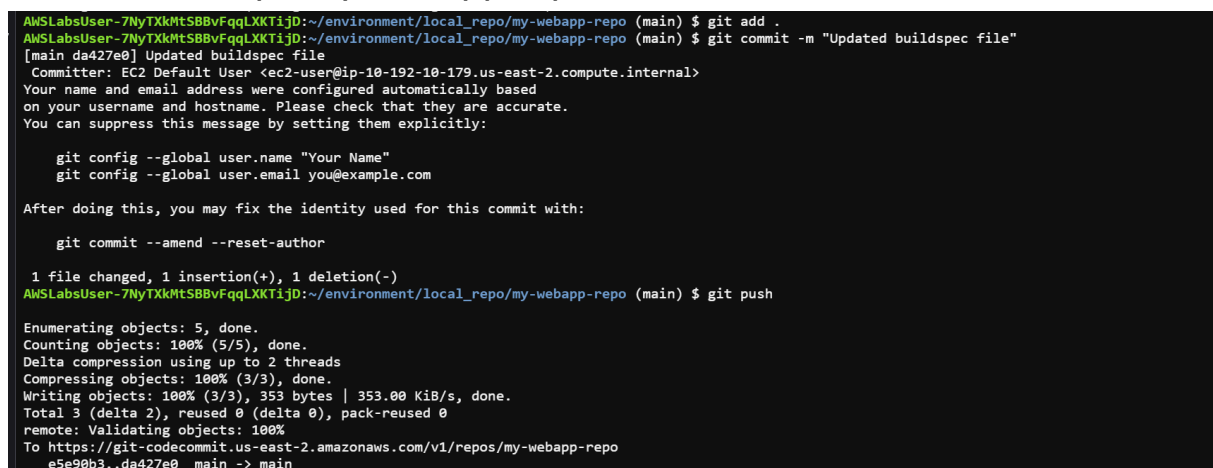
1. Pre-build stage: Logs in to Amazon ECR and sets up variables.
2. Build stage: Builds and tags the Docker image.
3. Post-build stage: Pushes the image to Amazon ECR.

2.2 Change the URI value in that file from the value in lab



```
1 version: 0.2
2
3 phases:
4   pre_build:
5     commands:
6       - echo Logging in to Amazon ECR...
7       - aws --version
8       # - REPOSITORY_URI=<>ACCOUNT-ID>.dkr.ecr.<REGION>.amazonaws.com/<APPLICATION-REPO-NAME>
9       - REPOSITORY_URI=773946498214.dkr.ecr.us-east-2.amazonaws.com/my-webapp-repo
10      - aws ecr get-login-password --region $AWS_DEFAULT_REGION | docker login --username AWS --p
11      - IMAGE_TAG=build-$(echo $CODEBUILD_BUILD_ID | awk -F":" '{print $2}')
12
13   build:
14     commands:
15       - echo Build started on `date`
16       - echo Building the Docker image...
17       - docker build -t $REPOSITORY_URI:latest .
18       - docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG
19
20   post_build:
21     commands:
```

2.3 push the changes you made to the build specification file to the codecommit repo my-webapp-repo.



```
ANSLabsUser-7NyTXkMtS8BvFqgLXKtjD:~/environment/local_repo/my-webapp-repo (main) $ git add .
ANSLabsUser-7NyTXkMtS8BvFqgLXKtjD:~/environment/local_repo/my-webapp-repo (main) $ git commit -m "Updated buildspec file"
[main da427e0] Updated buildspec file
Committer: EC2 Default User <ec2-user@ip-10-192-10-179.us-east-2.compute.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

1 file changed, 1 insertion(+), 1 deletion(-)
ANSLabsUser-7NyTXkMtS8BvFqgLXKtjD:~/environment/local_repo/my-webapp-repo (main) $ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 353 bytes | 353.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.us-east-2.amazonaws.com/v1/repos/my-webapp-repo
e5e90b3..da427e0 main -> main
```

Objective 3 - Create a CI/CD pipeline named my-webapp-pipeline for automating the creation of the application image based on the build specification file.

3.1 Open CodePipeline, create a new pipeline with following values

Pipeline settings

Pipeline name

Enter the pipeline name. You cannot edit the pipeline name after it is created.

No more than 100 characters

3.2 By adding this stage, any changes you make to the source code is automatically detected and continuously integrated into the pipeline workflow.

Source

Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

AWS CodeCommit ▼

Repository name
Choose a repository that you have already created where you have pushed your source code.

Q my-webapp-repo X

Branch name
Choose a branch of the repository

Q main X

3. 3 add a stage to build your application artifacts using AWS CodeBuild. AWS CodeBuild refers to the source artifacts and builds the application based on the commands you provided in build specification file

Build - optional

Build provider
This is the tool of your build project. Provide build artifact details like operating system, build spec file, and output file names.

AWS CodeBuild ▼

Region

US East (Ohio) ▼

Project name
Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.

Q my-webapp-build-project X or [Create project](#)

✔ Successfully created my-webapp-build-project in CodeBuild. X

Environment variables - optional
Choose the key, value, and type for your CodeBuild environment variables. In the value field, you can reference variables generated by CodePipeline. [Learn more](#)

[Add environment variable](#)


Build type

☒ **Single build**
Triggers a single build.

☐ **Batch build**
Triggers multiple builds as a single execution.

In creating a new project section, fill the following values

- For **Project name** , choose [Create project](#)

 **Note:** The browser opens a new tab for creating a CodeBuild project.

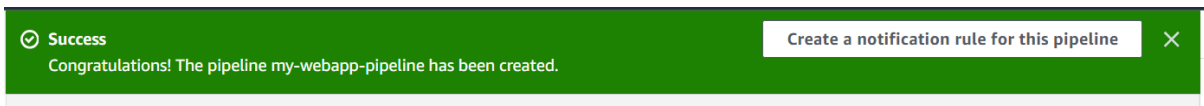
21. On the **Create build project** page, in the **Project configuration** section:

- For **Project name** , enter  `my-webapp-build-project`

22. In the **Environment** section:

- For **Environment image** , select **Managed image**
- For **Operating system** , choose **Ubuntu**
- For **Runtime(s)** , choose **Standard**
- For **Image** , choose **aws/codebuild/standard:7.0**
- For **Image version** , choose **Always use the latest image for this runtime**
- For **Privileged** , select the check box **Enable this flag if you want to build**
- For **Service role** , select **Existing service role**
- For **Role ARN** , choose **CodeBuild_Service_Role**

3.4 Click on create pipeline



Pipeline created

Objective 4 - Preparing the ECS cluster to support application deployment

4.1 Create ECS task definition

Run the following command to create the task definition, it takes the input from taskdef.json file, pre-existing in the Cloud9 IDE.

```
aws ecs register-task-definition --cli-input-json file://taskdef.json
```

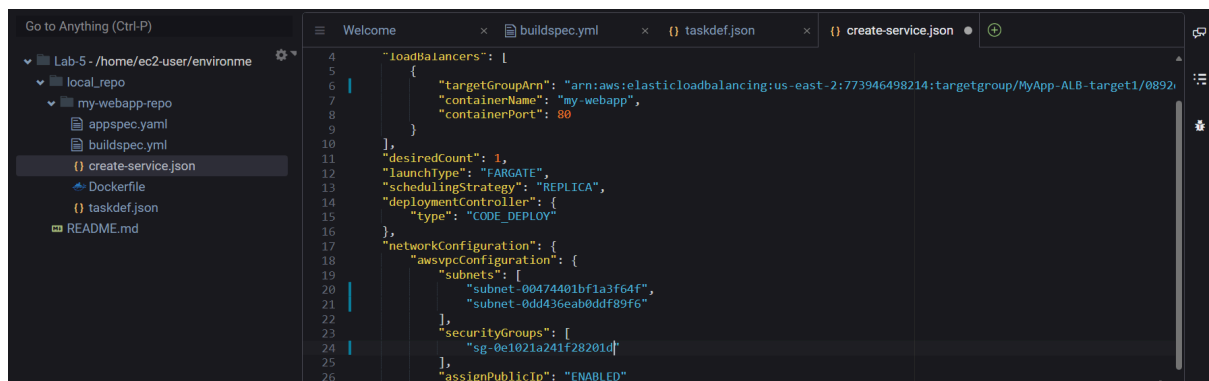
```
{
  "taskDefinition": {
    "taskDefinitionArn": "arn:aws:ecs:us-east-2:773946498214:task-definition/my-webapp:1",
    "containerDefinitions": [
      {
        "name": "my-webapp",
        "image": "773946498214.dkr.ecr.us-east-2.amazonaws.com/my-webapp-repo",
        "cpu": 0,
        "portMappings": [
          {
            "containerPort": 80,
            "hostPort": 80,
            "protocol": "tcp"
          }
        ],
        "essential": true,
        "environment": [],
        "mountPoints": [],
        "volumesFrom": [],
        "systemControls": []
      }
    ],
    "family": "my-webapp",
    "executionRoleArn": "arn:aws:iam::773946498214:role/Task_Definition_Role",
    "networkMode": "awsvpc",
    "revision": 1,
    "volumes": [],
    "status": "ACTIVE",
    "requiresAttributes": [

```

Successfully created an ECS task to define your container's configurations.

4.2 Create an ECS service MyApp-Web-service based on the task you defined in the previous task.

Open create_service.json file and insert the values for variables, that is suggested in lab manual



4.3 To create the service for your ECS cluster, run the following command

```
aws ecs create-service --service-name MyApp-Web-service --cli-input-json file://create-service.json
```

4.4 successfully created an Amazon ECS service and defined how to run it.

4.5 Push Changes to the source CodeCommit Repo

```

AWSLabsUser-7NyTXkMtSBBvFqgLXKTjD:~/environment/local_repo/my-webapp-repo (main) $ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   create-service.json
        modified:   taskdef.json

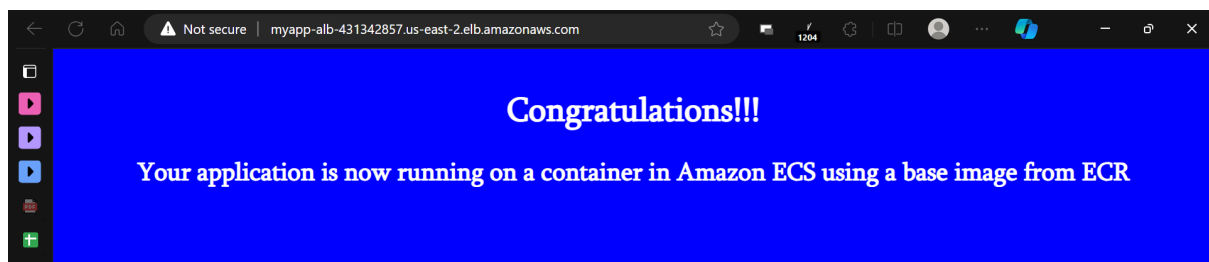
```

Successfully configured your ECS cluster to support the deployment of a demonstration application. ECS task and ECS service are now in effect to deploy application automatically in the next task.

NOTE - By making the required application source code available and configuring ECS service in the previous tasks, you have manually deployed the application

Objective 5 - Review the current running state of your web application

5.1 Put **LoadBalancerDNSName** value from lab manual in new tab, and webapp opens



Objective 6 - Configure the Blue/green deployment

1. add the Deploy stage to the pipeline.
2. current running state of the application (blue).
3. you make a code change to start the pipeline to initiate an automatic application deployment. You notice AWS CodeDeploy creating the new application environment (green)

6.1 CREATE A CODEDEPLOY APPLICATION AND DEPLOYMENT GROUP

Open CodeDeploy, create a new application as follows:

Create application

Application configuration

Application name

Enter an application name

100 character limit

Compute platform

Choose a compute platform


Tags

[Cancel](#)

[Create application](#)


Create a new deployment group
Input the following values

58. On the **Create deployment group** page, for the **Deployment group name** section:

- For **Enter a deployment group name** , enter  MyApp-Web-ECS-Group

59. In the **Service role** section:

- For **Enter a service role** , choose **CodeDeploy_Service_Role**

 **Note:** Automatically updates with the **ARN** .

60. In the **Environment configuration** section:

- For **Choose an ECS cluster name** , choose **my-webapp-cluster**
- For **Choose an ECS service name** , choose **MyApp-Web-service**

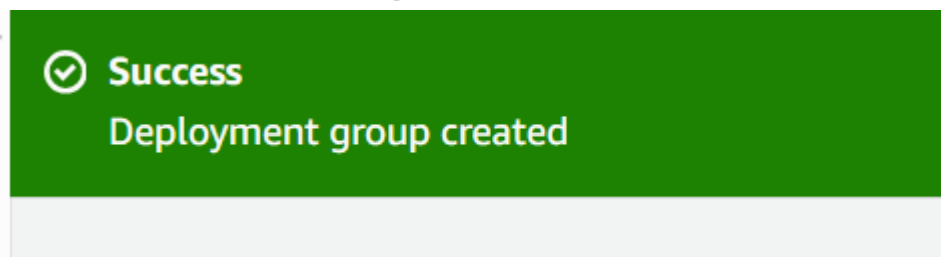
61. In the **Load balancers** section:

- For **Choose a load balancer** , choose **MyApp-ALB**
- For **Production listener port** , choose **HTTP: 80**
- For **Test listener port - *optional*** , leave blank
- For **Target group 1 name** , choose **MyApp-ALB-target1**
- For **Target group 2 name** , choose **MyApp-ALB-target2**

62. In **Deployment Settings** section:

- For **Traffic rerouting** , select **Reroute traffic immediately**
- For **Deployment configuration** , choose **CodeDeployDefault.ECSAllAtOnce**
- For **Original revision termination** section, make the following selections:

Create the deployment group



6.2 Adding Deploy Stage

Open the one given pipeline, and click on edit button


my-webapp-pipeline

 **Notify** ▼  **Edit**  **Stop execution**  **Clone pipeline**  **Release change**


Pipeline type: **V2** Execution mode: **QUEUED**

Using the add stage button, add a new stage - Deploy

Edit: Deploy Cancel Delete Done


 Add action group


☐ Configure automatic rollback on stage failure





 Add stage

For creating the stage, give the following values

68. In the **Edit action** pop-up window:

- For **Action name** , enter  `my-webapp-deploy`
- For **Action provider** , choose **Amazon ECS (Blue/Green)**

 **Note:** The browser refreshes the page with additional options.

- For **Region** , ensure it matches the **AwsRegionCode** value from the **Resources** panel to the
- For **Input artifacts** , choose **SourceArtifact**
- Choose  and add a second artifact:
- For **Input artifacts** , choose **BuildArtifact**
- For **AWS CodeDeploy application name** , choose **MyApp-Web**
- For **AWS CodeDeploy deployment group** , choose **MyApp-Web-ECS-Group**
- For **Amazon ECS task definition** , choose **SourceArtifact** and enter  `taskdef.json`
- For **AWS CodeDeploy AppSpec file** , choose **SourceArtifact** and enter  `appspec.yaml`
- For **Input artifact with image details** , choose **BuildArtifact**
- For **Placeholder text in the task definition** , enter  `IMAGE1_NAME`

Edit action

Action name

Choose a name for your action

my-webapp-deploy

No more than 100 characters

Action provider

Amazon ECS (Blue/Green)

Region

US East (N. Virginia)

Input artifacts

Choose an input artifact for this action. [Learn more](#)

SourceArtifact

BuildArtifact

Add

Remove

No more than 100 characters

AWS CodeDeploy application name

Choose one of your existing applications, or create a new one in AWS CodeDeploy.

MyApp-Web

Create application

AWS CodeDeploy deployment group

Choose one of your existing deployment groups, or create a new one in AWS CodeDeploy.

MyApp-Web-ECS-Group

Amazon ECS task definition

Choose the input artifact where your Amazon ECS task definition file is stored. If other than the default file path, specify the path and filename of your task definition file.

SourceArtifact

taskdef.json

The default path is taskdef.json.

AWS CodeDeploy AppSpec file

Choose the input artifact where your AWS CodeDeploy AppSpec file is stored. If other than the default file path, specify the path and filename of your AppSpec file.

SourceArtifact

appspec.yaml

Dynamically update task definition image - *optional*

You can provide an input artifact and a placeholder name for the container definition image that will be used to dynamically update a task definition. You can specify multiple input artifacts and placeholders.

Input artifact with image details

BuildArtifact

Placeholder text in the task definition

IMAGE1_NAME

Add

Remove

Variable namespace - *optional*

Choose a namespace for the output variables from this action. You must choose a namespace if you want to use the variables this action produces in your configuration. [Learn more](#)

Cancel

Done

Save the pipeline.

Successfully configured AWS CodeDeploy to handle Blue/Green deployments for the ECS environment and added a new stage to the pipeline to automated this.

The pipeline is now ready to automate application deployments using a blue/green deployment type when any changes made to the source code in my-webapp-repo.

Objective 7 - : Testing the CodePipeline automation

You verify that the my-webapp-pipeline is automatically performing blue/green deployments by eliminating the need for manual intervention for building new images and deploying to the containers.

7.1 Change the webpage's background-color to green in the docker file and push the code changes to my-webapp-repo.

The pipeline initiates the release workflow once changes are pushed.

The deploy stage creates the green environment, shifts the traffic to it, and removes the blue environment once deployment is successful.

NOTE - To change color to green, just replace blue with green in the app code

```
argin-top: 40px; background-color:green;}
```

Push this change in code.

Now, open the pipeline window, and we see that the changes are being deployed (see the checklist on right side of screenshot)

my-webapp-pipeline

Pipeline type: V2 Execution mode: QUEUED

Notify Edit Stop execution Clone pipeline Release change

Source Succeeded

Pipeline execution ID: 3c1dbefb-15c6-4d53-a119-dbc14c9b7cf7

Build

AWS CodeBuild

Succeeded - Just now

View details

963afd98 Source: Updated Dockerfile file

Disable transition

Deploy In progress

Pipeline execution ID: 3c1dbefb-15c6-4d53-a119-dbc14c9b7cf7

my-webapp-deploy

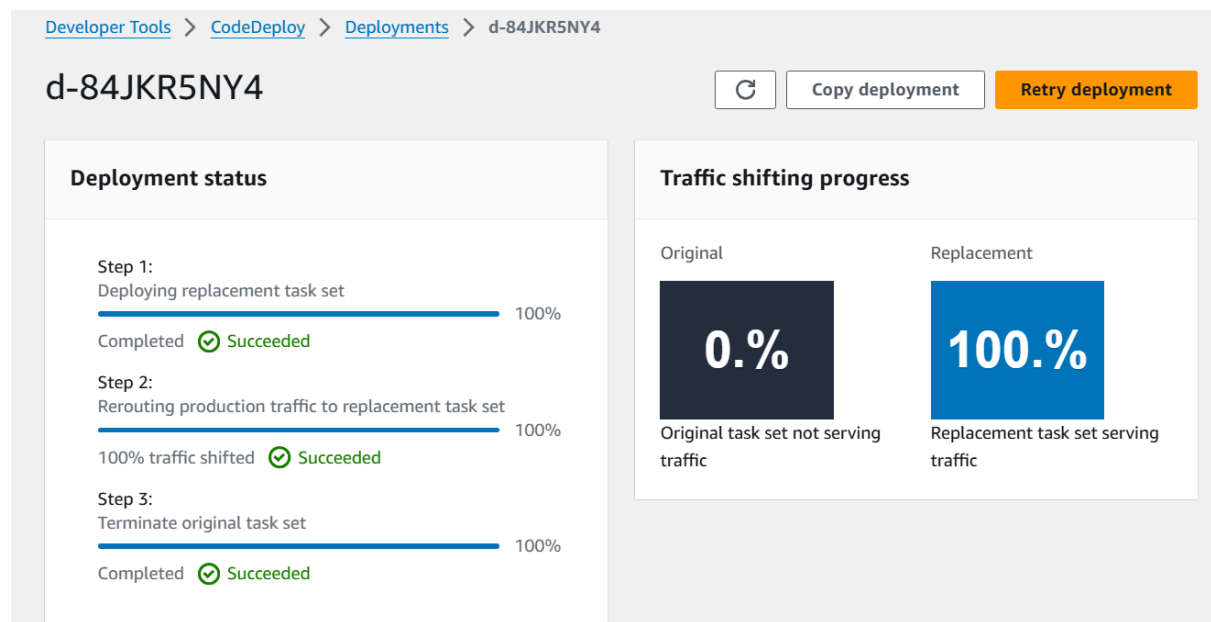
Amazon ECS (Blue/Green)

In progress - Just now

View details

963afd98 Source: Updated Dockerfile file

When deploy stage is being processed, click on the deployment id (in the deployment window), the following is shown



It depicts the deployment status once completed with traffic shifting to the replacement instance.

Now when we visit the web-app, bg color is green

