Lokalise DevOps/SRE take-home assignment

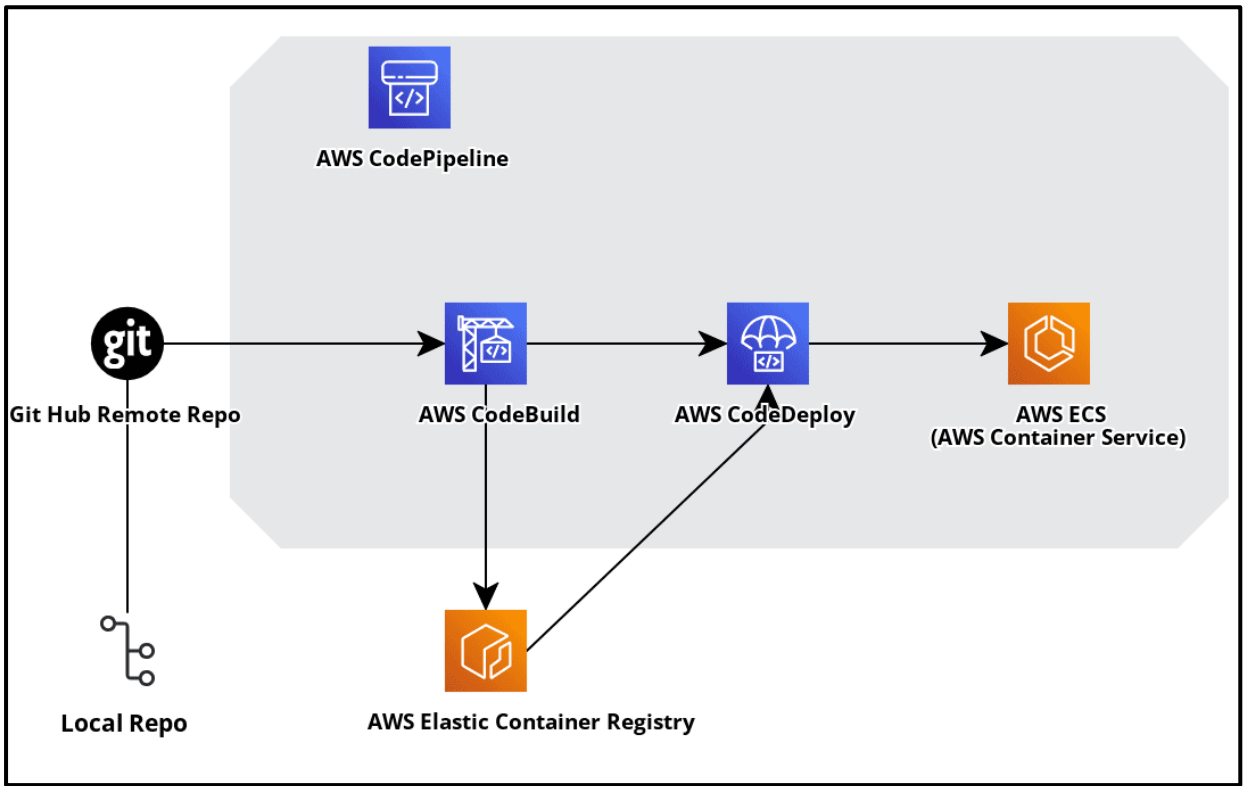
Goal:

- To build a simple proof-of-concept of a blue-green deployment for a web application

Result:

- Got the desired output as one of many alternatives.

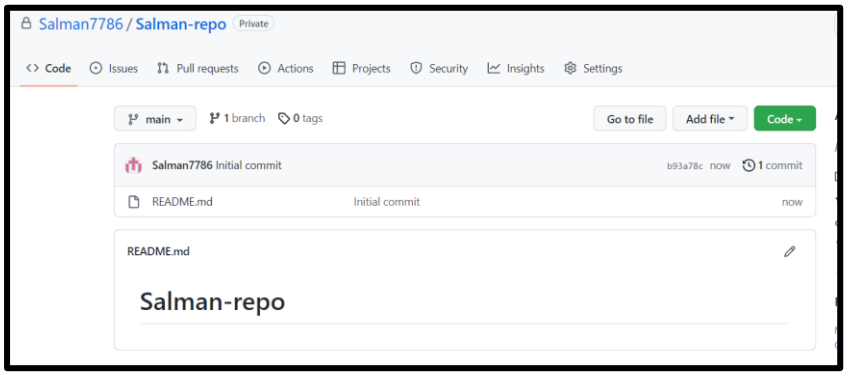
Overview of the Solution:



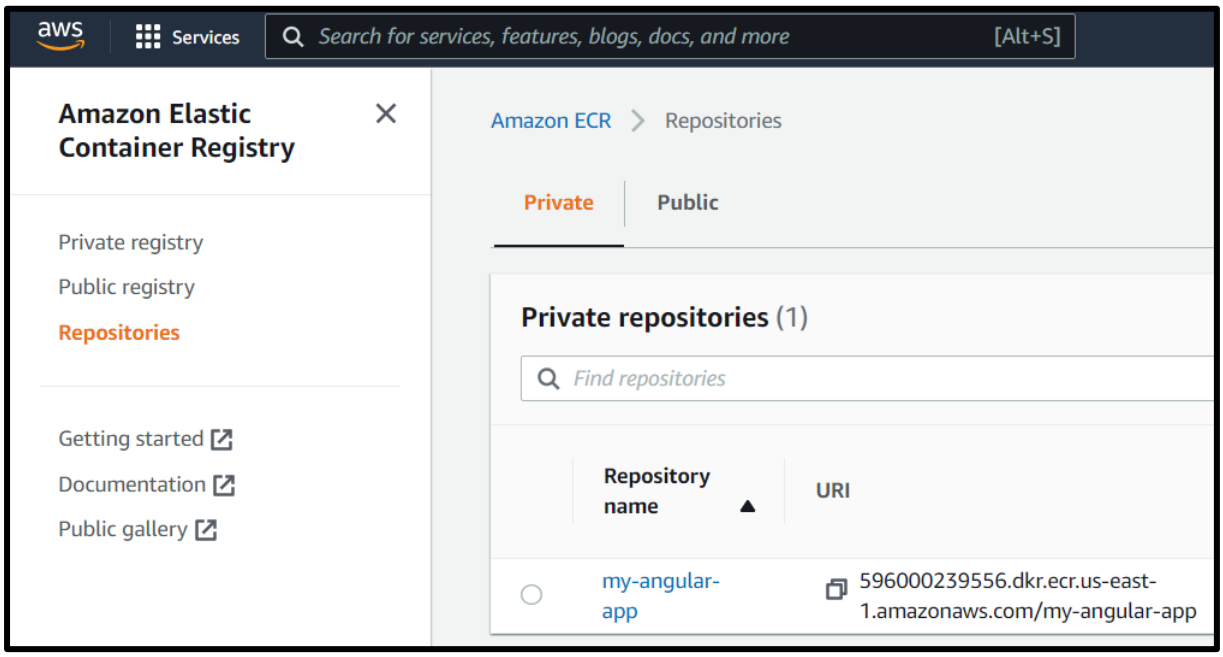
Technology Stack used:

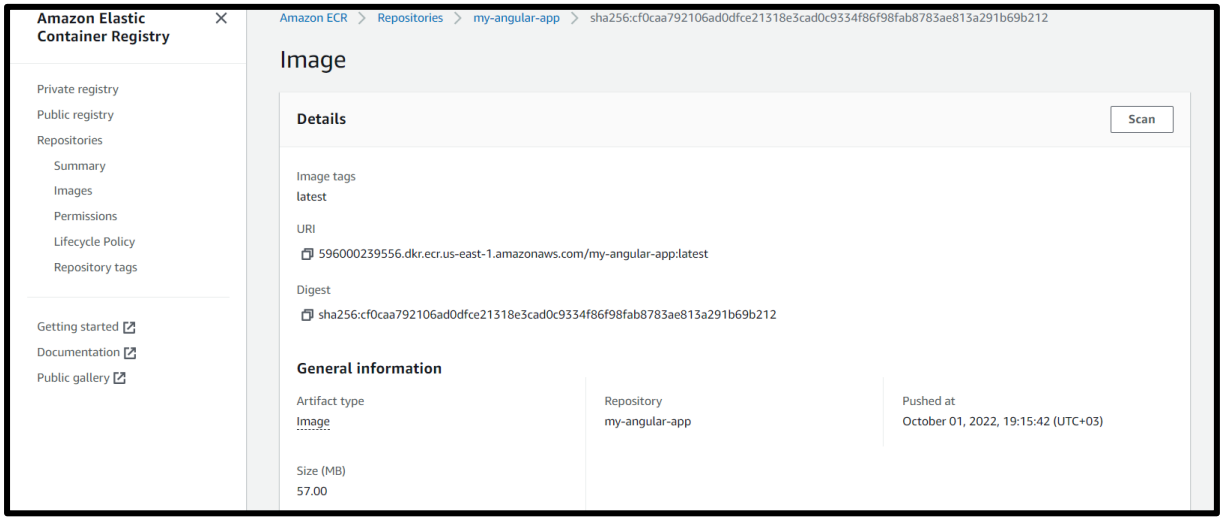
- AWS Cloud
- CloudFormation as IaaC
- Build Docker Container image
- Elastic Container Repository and DockerHUB
- AWS Parameter Store for storing the secrets
- Container Service AWS ECS
- AWS Application Load balancer
- CICD tool used – AWS: CodePipeline, CodeBuild, CodeDeploy.
- For Blue Green deployments, Created Deployment Group with ECS Service along with ALB which routes traffic to Listener target group as per the deployment cycle.s

1. Create Fresh GitHub Remote Repository



2. AWS ECR Private registry for storing docker container Images





3. Git Local machine - initialization

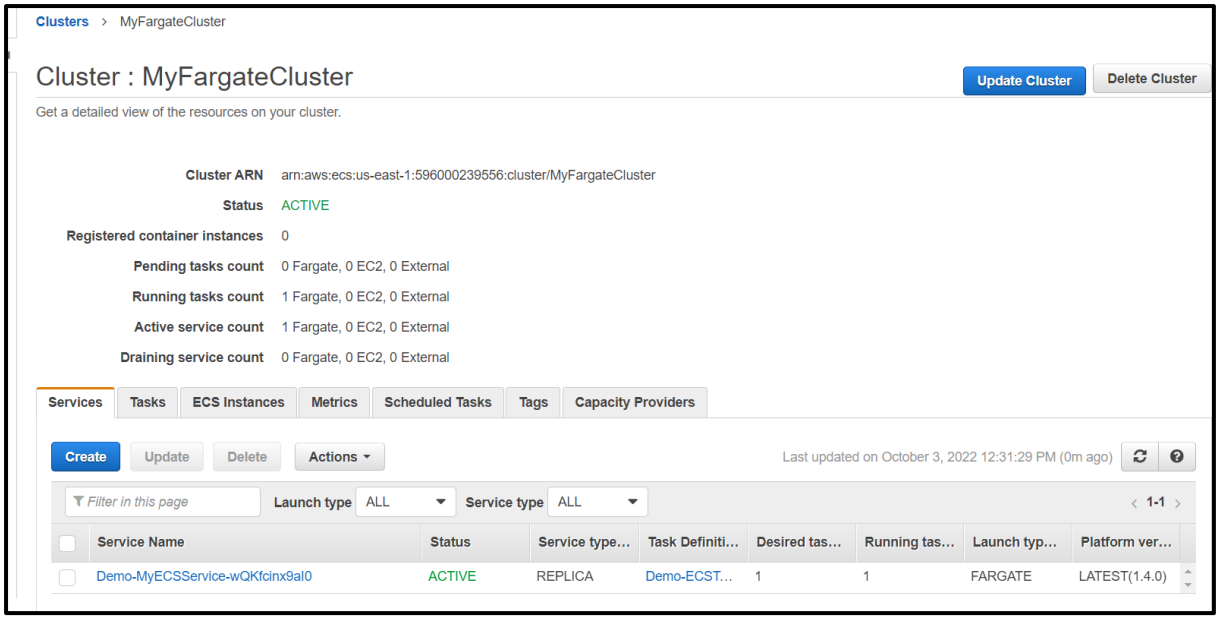
- Inside my sample application folder
- git init
- git remote add origin https://github.com/<repo-name>
- git add .
- git status
- git commit -m "Docker and build Spec were added."
- git push origin master

4. ECSCloudFormation.yaml

The CloudFormation code is creating below resources.

- Using the existing VPC and two Subnet
- ECS Task Definition
- ECS Cluster
- Application Load Balancer
- Target Group for ALB
- Listener
- Security Groups for ALB and Security Group for Containers
- ECS Service.
- Using the Custom docker image from the ECR repository.

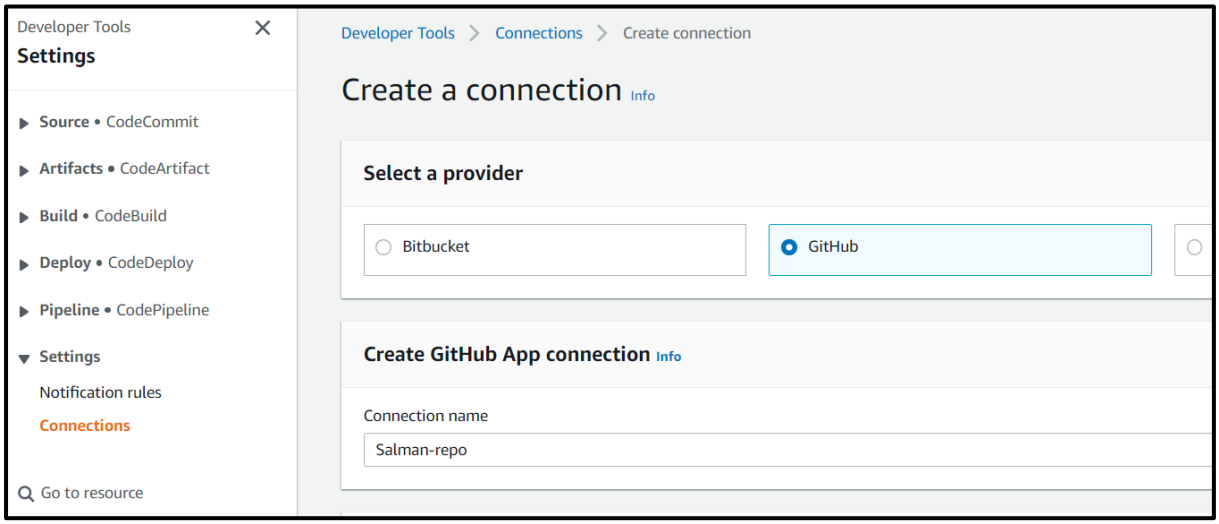
At this point I can access my web application through Application Load balancer URL.
It's showing the current version and 1.0

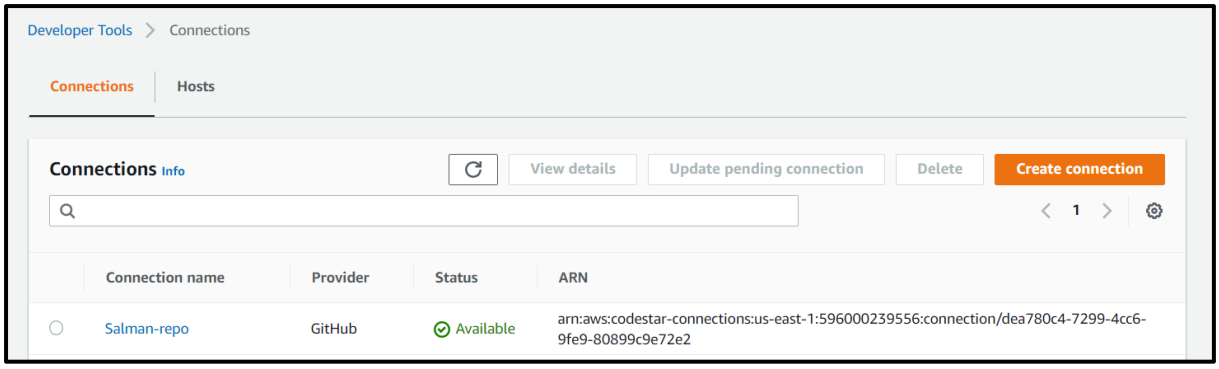


5. Create Service Link IAM Role for CodePipeline, CodeBuild, and CodeDeploy or we can dynamically use existing AWS managed policy.

6. CICD using CodePipeline

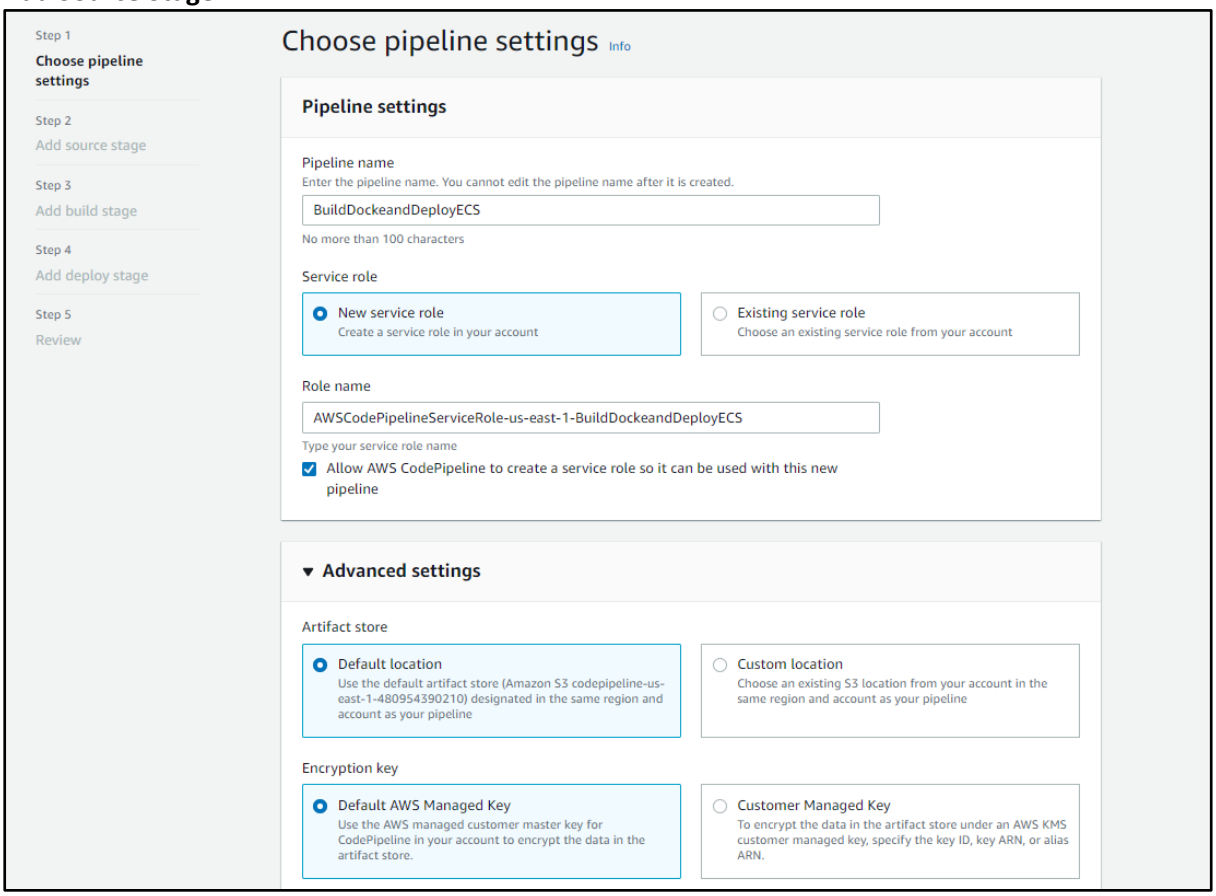
6.1 We need to integrate the GitHub repository with our AWS CodePipeline so we need to create a new connection in AWS console.





6.2 Create CodePipeline: So that every Push (git push origin master) will invoke the AWS **CodePipeline** and trigger the subsequent event as creating docker images using **CodeBuild** and pushing it to docker **registry**. And Automated Deployment to **ECS** using **CodeDeploy**.

Add Source Stage:



Step 2
Add source stage

Step 3
Add build stage

Step 4
Add deploy stage

Step 5
Review

Source

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

GitHub (Version 2)

New GitHub version 2 (app-based) action
To add a GitHub version 2 action in CodePipeline, you create a connection, which uses GitHub Apps to access your repository. Use the options below to choose an existing connection or create a new one. Learn more

Connection
Choose an existing connection that you have already configured, or create a new one and then return to this task.

arn:aws:codestar-connections:us-east-1:596000239556:connection/dea780c

or **Connect to GitHub**

Ready to connect
Your GitHub connection is ready for use.

Repository name
Choose a repository in your GitHub account.

Salman7786/Salman-repo

<account>/<repository-name>

Branch name
Choose a branch of the repository.

master

Change detection options

☑ Start the pipeline on source code change
Automatically starts your pipeline when a change occurs in the source code. If turned off, your pipeline only runs if you start it manually or on a schedule.

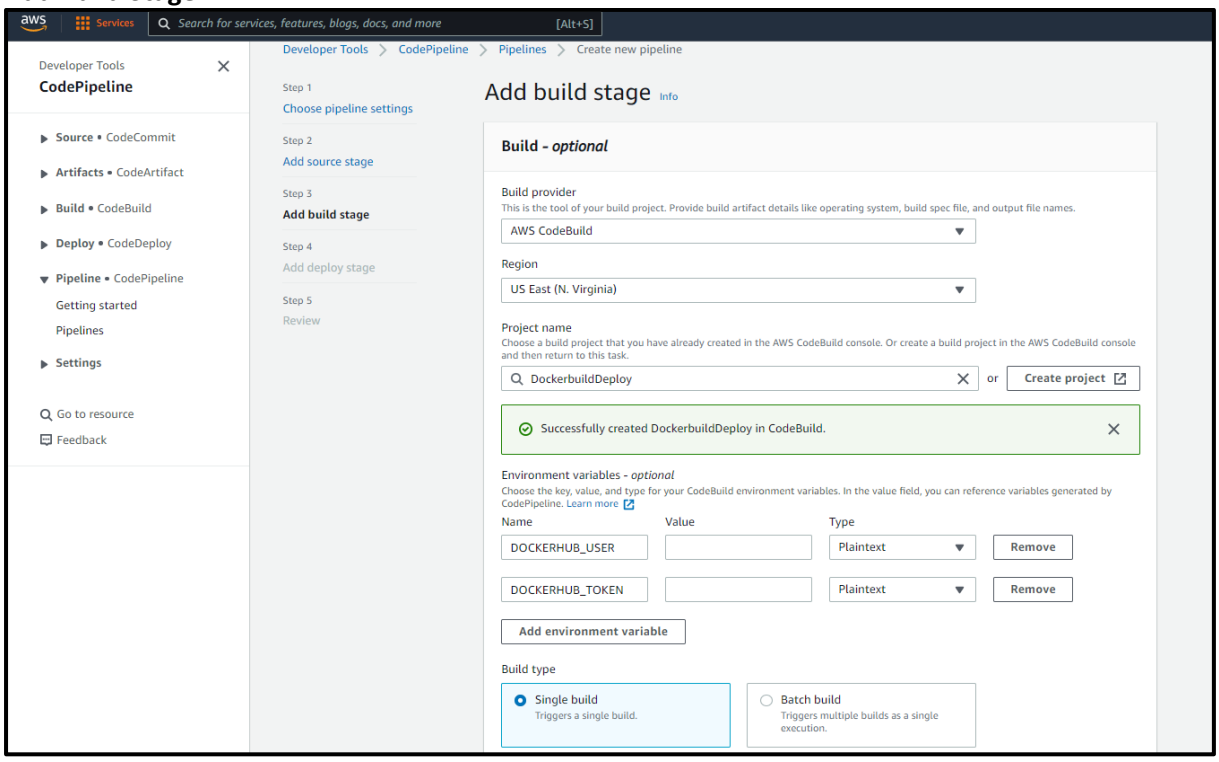
Output artifact format
Choose the output artifact format.

◉ CodePipeline default
AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include git metadata about the repository.

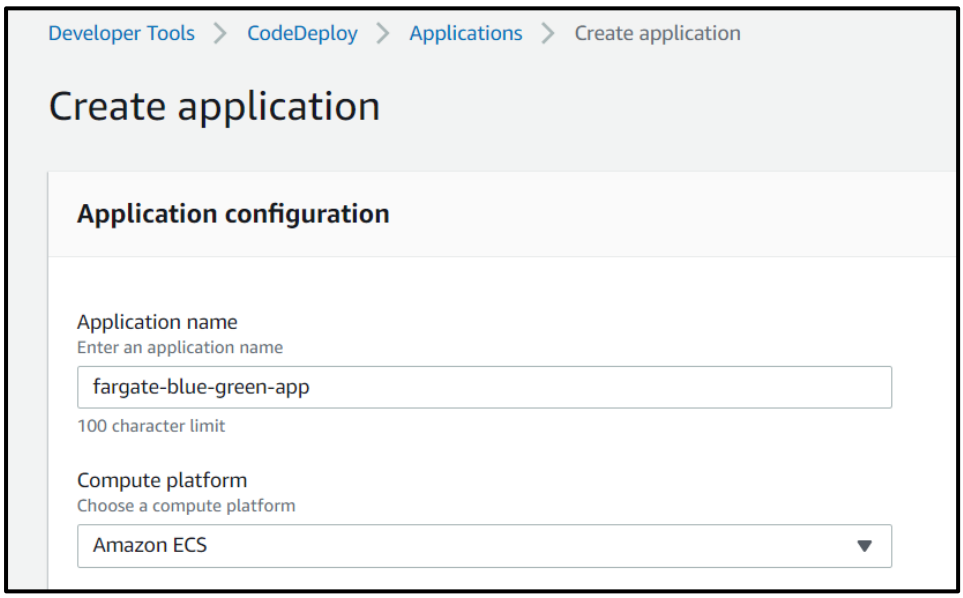
○ Full clone
AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full git clone. Only supported for AWS CodeBuild actions.

Cancel **Previous** **Next**

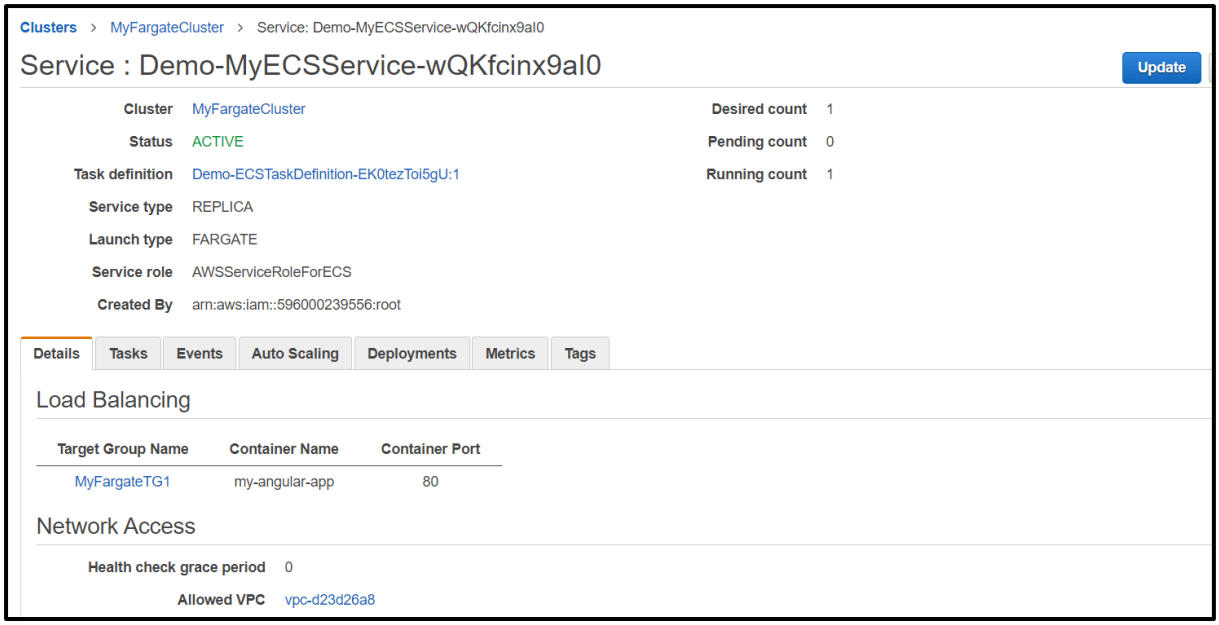
Add Build Stage:



Add CodeDeploy Stage: Create CodeDeploy Application

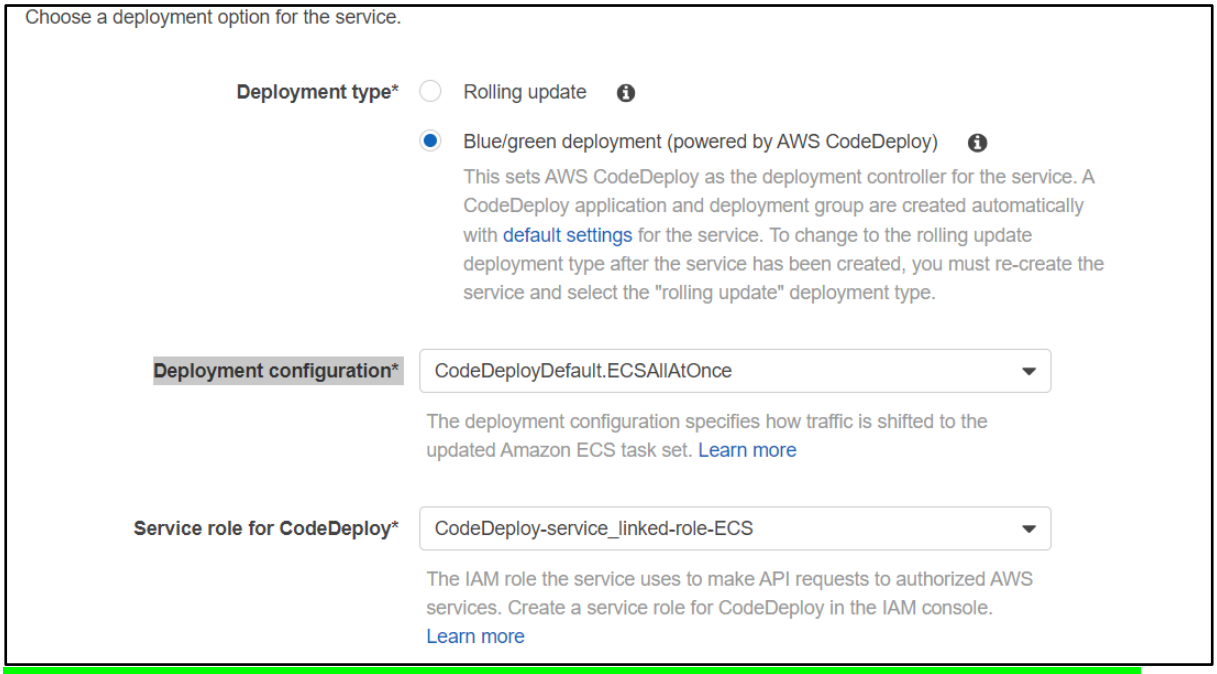


6.3 Update a Service definition in ECS console.



Use below Steps:

- Launch type: FARGATE
- Operating system family: Linux
- Choose a deployment option for the service.
- Blue/green deployment



Above config sets AWS CodeDeploy as the deployment controller for the service. A CodeDeploy application and deployment group are created automatically.

Container to load balance

my-angular-app : 80 Remove ✖

Production listener port* 80:HTTP

Production listener protocol* HTTP

Test listener ☑

An optional test listener is used to test the new application revision before routing traffic to it.

Test listener port* create new 8080

Test listener protocol* HTTP

Load balancer type Application Load Balancer (ALB)
Load balancer name FargateAlb
Container to load balance
Container port 80

Production listener port 80
Target group 1 name MyFargateTG1
Production listener path-pattern /
Production listener health-check path /

Test listener port 8080
Target group 2 name MyFargateTG2
Test listener path-pattern /
Test listener health-check path /

Create Service

Create service: fargate-service

Service created
Service created. Tasks will start momentarily. View: fargate-service

CodeDeploy for blue/green deployments

The CodeDeploy application was created.
CodeDeploy The application and deployment group was created successfully.

CodeDeploy application: AppECS-MyFargateCluster-fargate-service
To view or change deployment settings, go to the CodeDeploy deployment group: DgpECS-MyFargateCluster-fargate-service

The CodeDeploy application was created.
CodeDeploy The application and deployment group was created successfully.

Clusters > MyFargateCluster

Cluster : MyFargateCluster

Update Cl

Get a detailed view of the resources on your cluster.

Cluster ARN	arn:aws:ecs:us-east-1:596000239556:cluster/MyFargateCluster
Status	ACTIVE
Registered container instances	0
Pending tasks count	0 Fargate, 0 EC2, 0 External
Running tasks count	2 Fargate, 0 EC2, 0 External
Active service count	2 Fargate, 0 EC2, 0 External
Draining service count	0 Fargate, 0 EC2, 0 External

Services | Tasks | ECS Instances | Metrics | Scheduled Tasks | Tags | Capacity Providers

Create | Update | Delete | Actions

Last updated on October 3, 2022 1:10:11 P

Filter in this page | Launch type ALL | Service type ALL | 1 selected

Service Name	Status	Service type	Task Definition ...	Desired tasks	Running tasks	Launch type
fargate-service	ACTIVE	REPLICA	Demo-ECSTaskD...	1	1	FARGATE

Developer Tools
CodeDeploy

- Source • CodeCommit
- Artifacts • CodeArtifact
- Build • CodeBuild
- Deploy • CodeDeploy

Developer Tools > CodeDeploy > Applications

Applications

Notify | View details | Deploy application | Create application

Application name	Compute platform	Created
AppECS-MyFargateCluster-fargate-service	Amazon ECS	4 minutes ago

6.4 Stage CodeDeploy

Action name
Choose a name for your action

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

BuildArtifact

Add

No more than 100 characters

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

AppECS-MyFargateCluster-fargate-service | Create application

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

DgpECS-MyFargateCluster-fargate-service

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.

BuildArtifact | taskdef.json

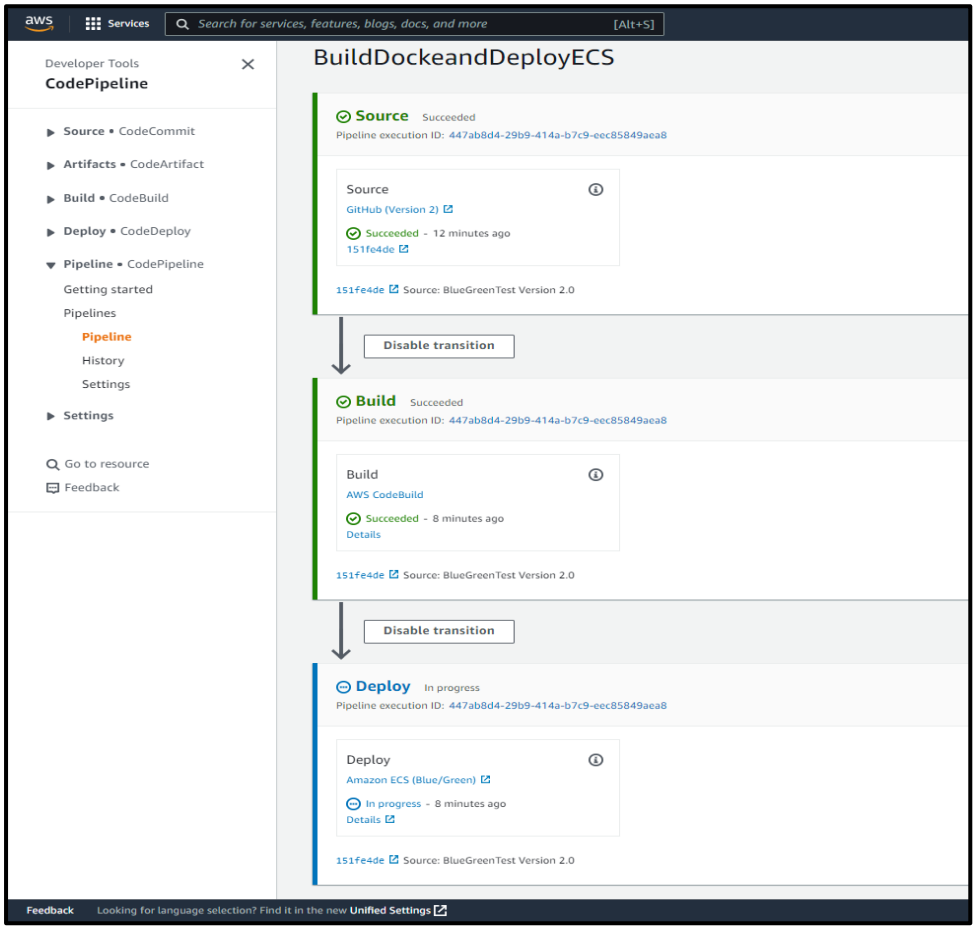
The default path is taskdef.json.

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.

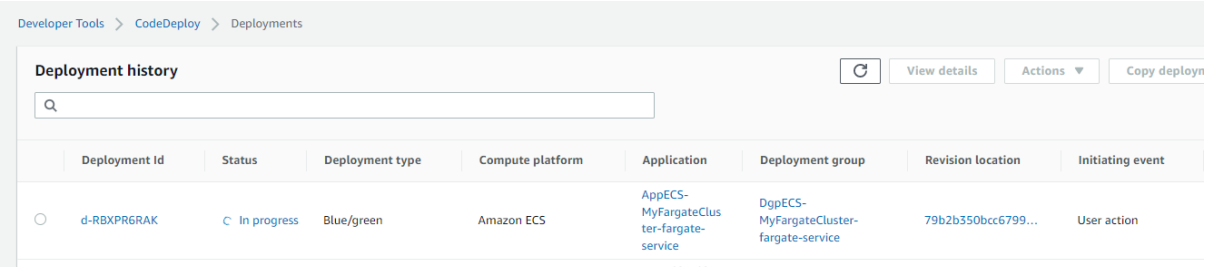
BuildArtifact | appspec.yaml

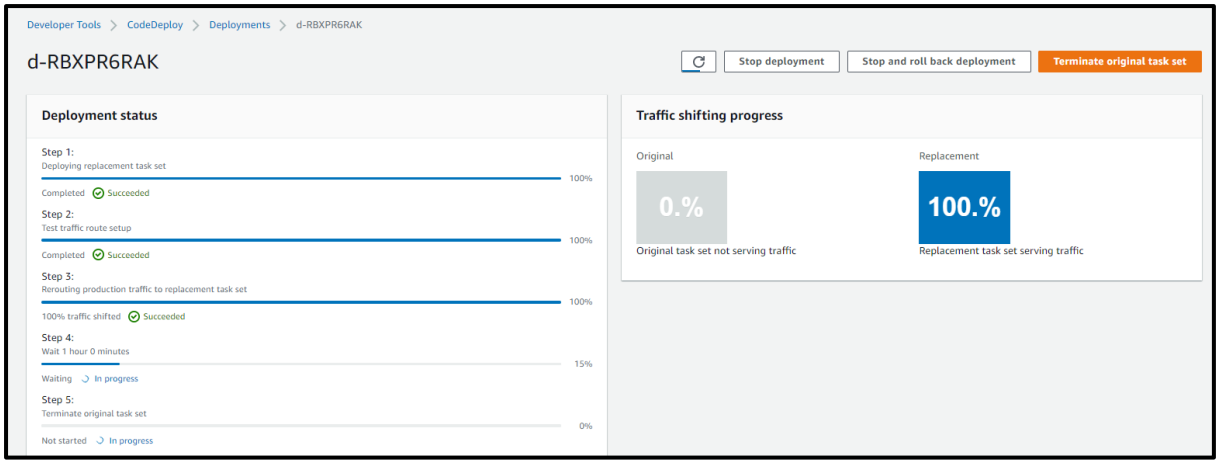
6.5 Commit the code and push

Trigerred our pipeline



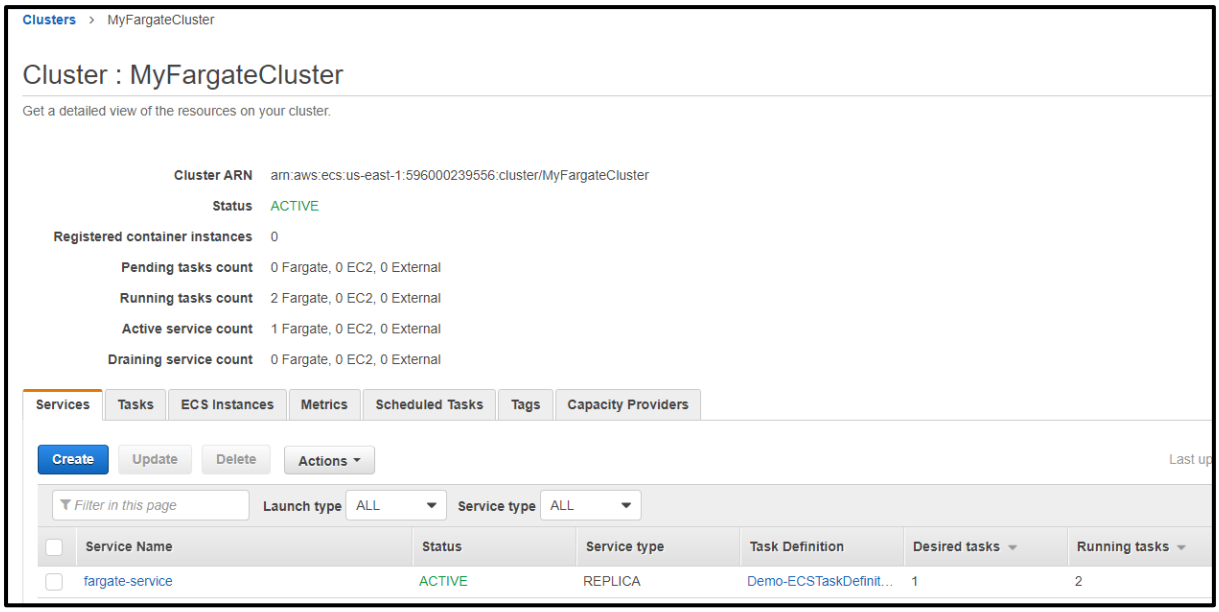
Deployment in Progress





In Deployment Stage, blue green deployment will initiate, and it will spin the additional container based on service and task definition.

And automatically it will register the Target group 2 to our Public Facing Load Balancer.



Clusters › MyFargateCluster › Service: fargate-service

Service : fargate-service

Cluster	MyFargateCluster	Desired count	1
Status	ACTIVE	Pending count	0
Task definition	Demo-ECSTaskDefinition-EK0tezToI5gU:1	Running count	2
Service type	REPLICA		
Launch type	FARGATE		
Service role	AWSServiceRoleForECS		
Created By	arn:aws:iam::596000239556:root		

Details | Tasks | Events | Auto Scaling | Deployments | Metrics | Tags

Task Placement

Strategy	No strategies
Constraint	No constraints

Blue/green deployment

Deployment ID	d-RBXPR6RAK	CodeDeploy deployment group	DgpECS-MyFargateCluster-fargate-service
Type	Blue/green	Start time	2022-10-03 01:42:26
Started by	AWS CodeDeploy	End time	-
Status	InProgress	Deployment history	DgpECS-MyFargateCluster-fargate-service

Details | Tasks | Events | Auto Scaling | Deployments | Metrics | Tags

Task Placement

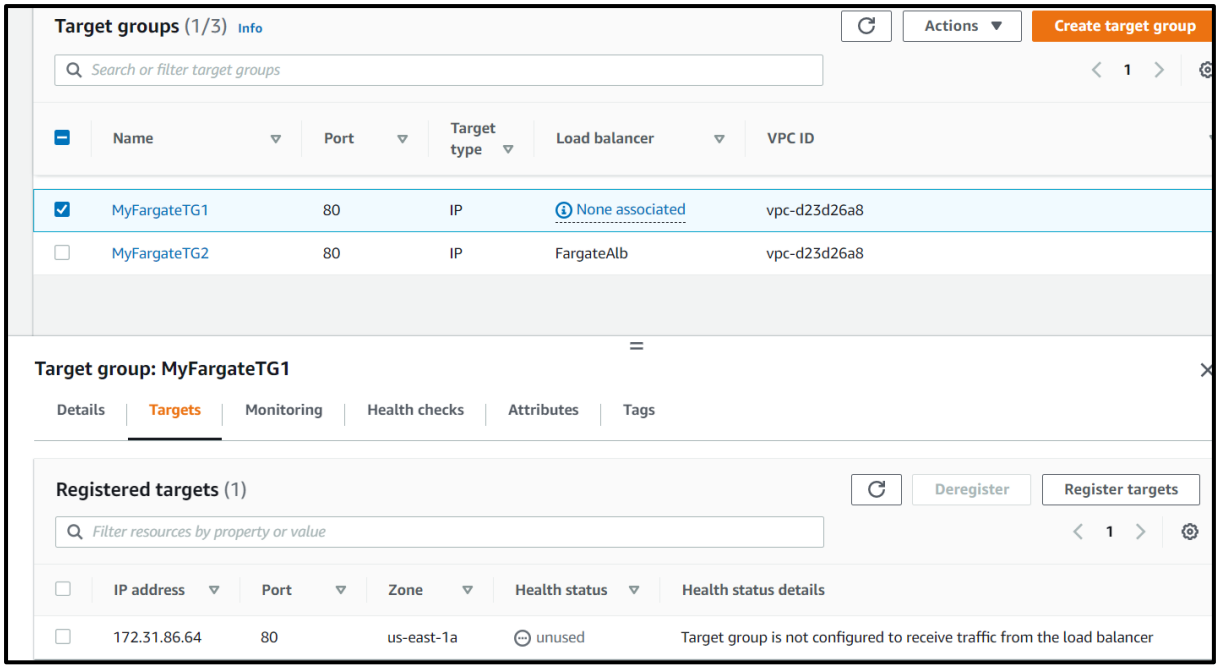
Strategy	No strategies
Constraint	No constraints

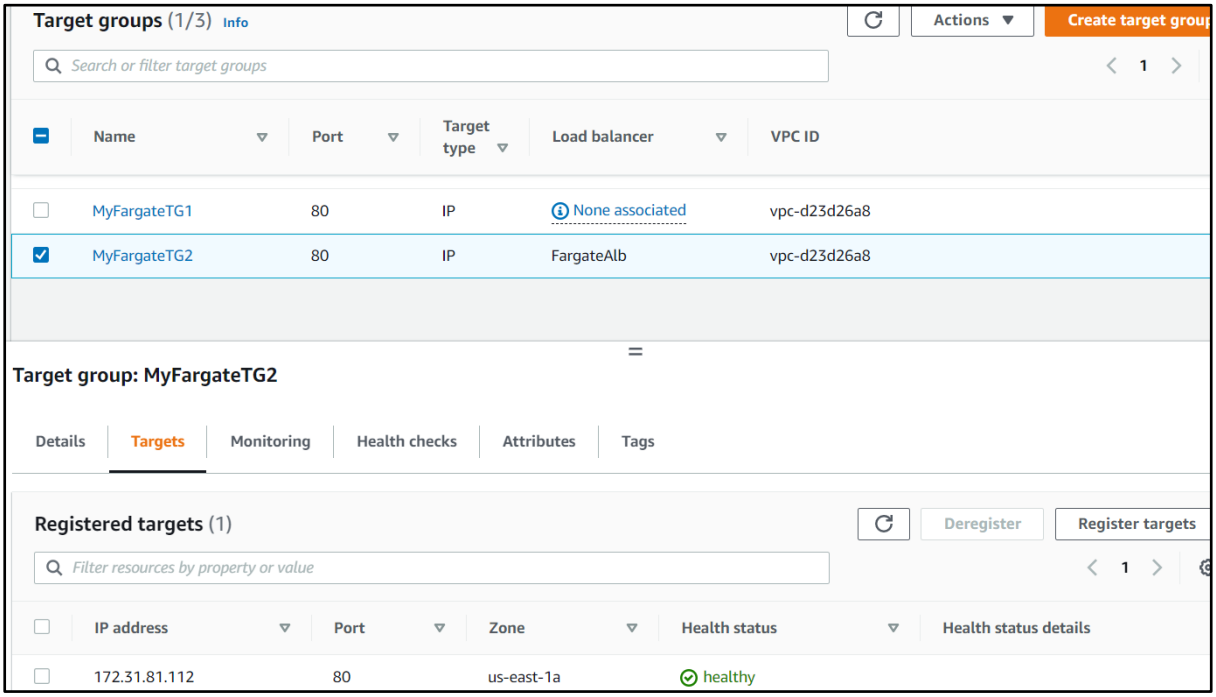
Blue/green deployment

Stop and rollback deployment

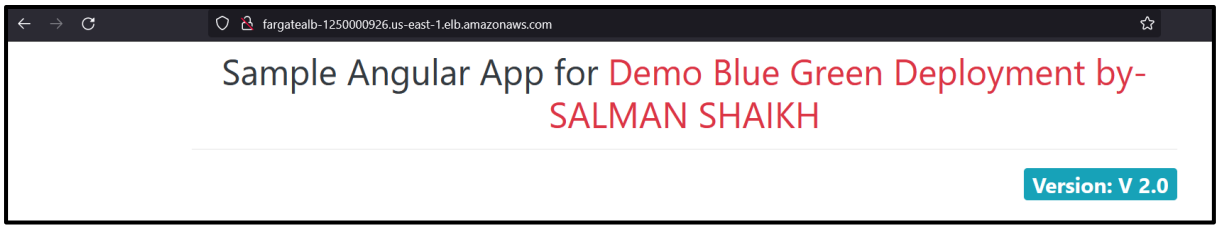
Deployment ID	d-RBXPR6RAK	CodeDeploy deployment group	DgpECS-MyFargateCluster-fargate-service
Type	Blue/green	Start time	2022-10-03 01:42:26
Started by	AWS CodeDeploy	End time	-
Status	InProgress	Deployment history	DgpECS-MyFargateCluster-fargate-service

Task set ID	Environment	Task set status	Traffic	Desired count	Running count	Pending count
ecs-svc/8910503786521628847	Replacement	PRIMARY	100%	1	1	0
ecs-svc/6965737905120952339	Original	ACTIVE	0%	1	1	0





Application Load Balancer URL.



Application is on latest version. Let's say We want to go back to previous state then simply "**Stop and Rollback the Deployment**".

I will continue with the changes and terminate the old container using "**Terminate Original task set**"