

# ECS Fargate Blue-Green Deployment using Code pipeline

## **Objective:**

Deploy a containerized Node.js application using Amazon ECS Fargate with Blue/Green deployment strategy via AWS CodePipeline, CodeBuild, and CodeDeploy.

## **Resources**

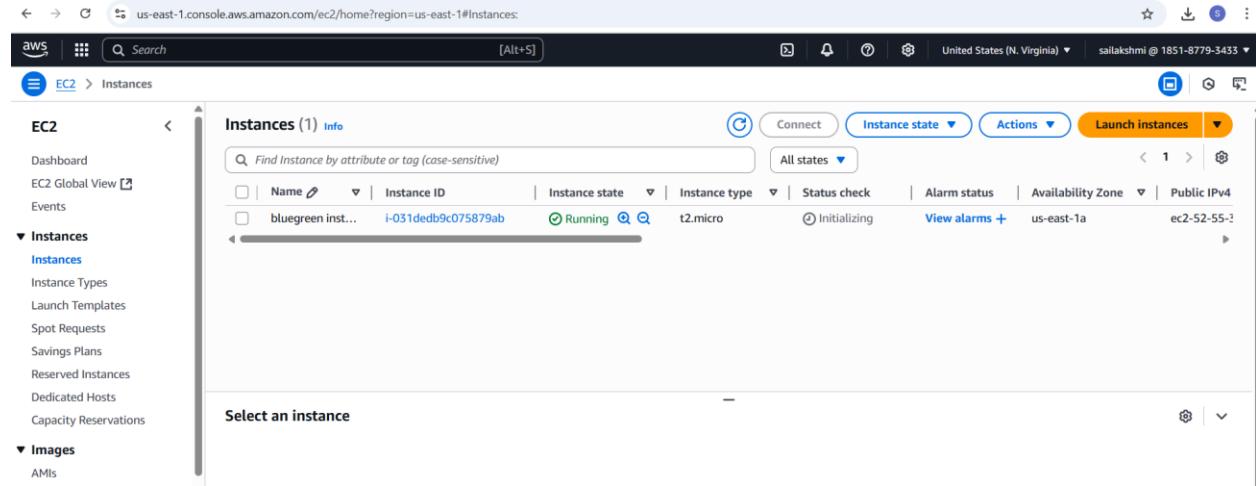
- ECS Cluster
- Fargate Task Definition & Service (with deployment\_controller as CODE\_DEPLOY)
- ECR Repository
- CodePipeline with CodeBuild and CodeDeploy stages
- IAM Roles
- ALB with target groups for blue/green

## **How Blue/Green Deployment Works**

- ECS Service is created with deployment controller set to CODE\_DEPLOY.
- CodePipeline updates the ECS Task Definition with the new image.
- CodeDeploy shifts traffic between Blue (old) and Green (new) targets via ALB.
- ALB target groups manage routing for each version.
- Zero-downtime deployment using weighted traffic shifting.

## Step-by-Step Deployment

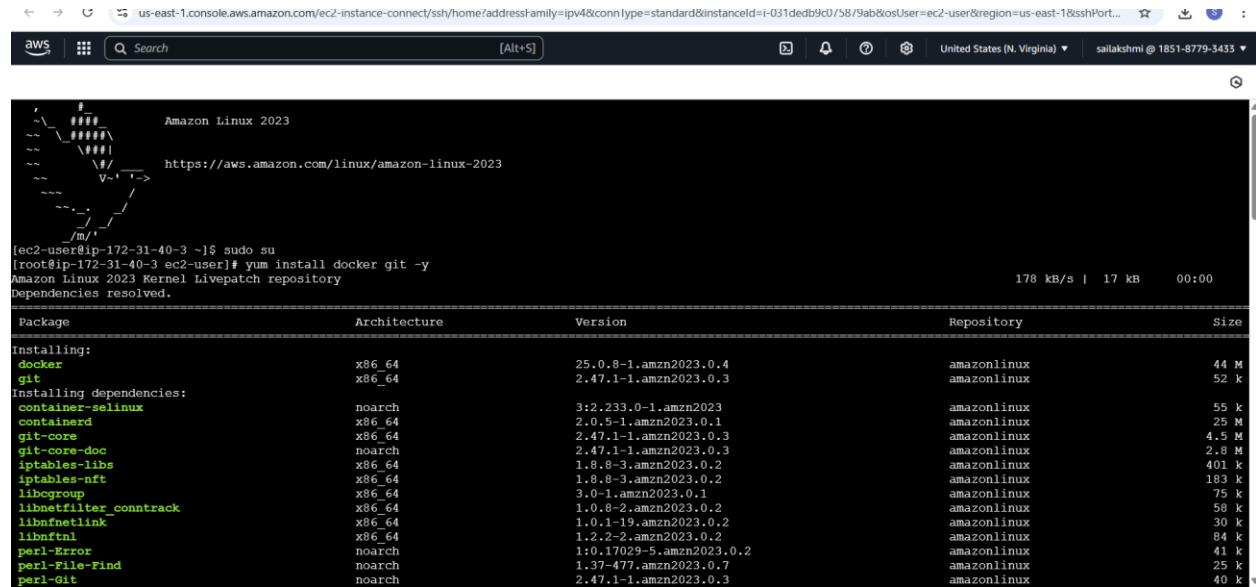
### Step:1 Create an ec2 instance



The screenshot shows the AWS Management Console EC2 Instances page. On the left, there's a navigation sidebar with 'EC2' selected. The main area displays a table titled 'Instances (1) Info' with one row. The instance is named 'bluegreen inst...', has an ID of 'i-031dedb9c075879ab', is 'Running', is a 't2.micro' type, and is in the 'Initializing' status. It's located in the 'us-east-1a' availability zone with a public IPv4 of 'ec2-52-55-2'. A search bar at the top allows filtering by name or instance ID.

Connect to ec2 instance

Install docker in it



The terminal session on the Amazon Linux 2023 instance shows the following commands and output:

```
sudo su
yum install docker git -y
Dependencies resolved.
Package           Architecture Version       Repository      Size
Installing:
  docker          x86_64    25.0.8-1.amzn2023.0.4   amazonlinux   44 M
  git             x86_64    2.47.1-1.amzn2023.0.3   amazonlinux   52 k
Installing dependencies:
  container-selinux noarch    3:2.233.0-1.amzn2023   amazonlinux   55 k
  containerd      x86_64    2.0.5-1.amzn2023.0.1   amazonlinux   25 M
  git-core        x86_64    2.47.1-1.amzn2023.0.3   amazonlinux   4.5 M
  git-doc         noarch    2.47.1-1.amzn2023.0.3   amazonlinux   2.8 M
  iptables-libc   x86_64    1.8.8-3.amzn2023.0.2   amazonlinux   401 k
  iptables-nft   x86_64    1.8.8-3.amzn2023.0.2   amazonlinux   183 k
  libcgroup       x86_64    3.0-1.amzn2023.0.1   amazonlinux   75 k
  libnetfilter_conntrack x86_64    1.0.8-2.amzn2023.0.2   amazonlinux   58 k
  libnetfilter_inetlink x86_64    1.0.1-19.amzn2023.0.2  amazonlinux   30 k
  libnftnl        x86_64    1.2.2-2.amzn2023.0.2   amazonlinux   84 k
  perl-Error     noarch    1:0.17029-5.amzn2023.0.2  amazonlinux   41 k
  perl-File-Find noarch    1.37-477.amzn2023.0.7   amazonlinux   25 k
  perl-Git        noarch    2.47.1-1.amzn2023.0.3   amazonlinux   40 k
```

Start docker

```
runc-1.2.4-2.amzn2023.0.1.x86_64

Complete!
[root@ip-172-31-40-3 ec2-user]# systemctl start docker
[root@ip-172-31-40-3 ec2-user]# aws --version
aws-cli/2.25.0 Python/3.9.23 Linux/6.1.141-155.amzn2023.x86_64 source/x86_64.amzn.2023
[root@ip-172-31-40-3 ec2-user]# git --version
git version 2.47.1
[root@ip-172-31-40-3 ec2-user]# aws configure
AWS Access Key ID [None]: AKIASWHQNPIMTWMEBGW6
```

## Step:2 Create a IAM role for code deploy, ECS

Trusted entity type

- AWS service
  - Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account
  - Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity
  - Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation
  - Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy
  - Create a custom trust policy to enable others to perform actions in this account.

**Use case**  
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

CodeDeploy

Choose a use case for the specified service.

Use case

- CodeDeploy
  - Allows CodeDeploy to call AWS services such as Auto Scaling on your behalf.
- CodeDeploy for Lambda
  - Allows CodeDeploy to route traffic to a new version of an AWS Lambda function version on your behalf.
- CodeDeploy - ECS
  - Allows CodeDeploy to read S3 objects, invoke Lambda functions, publish to SNS topics, and update ECS services on your behalf.

Cancel Next

Add permissions

Step 1 Select trusted entity

Step 2 **Add permissions**

Step 3 Name, review, and create

**Permissions policies (1)**

The type of role that you selected requires the following policy.

Policy name ▾ Type

AWSCodeDeployRole AWS managed

▶ Set permissions boundary - optional

Cancel Previous Next

Another role for ECS

Step 2 Add permissions

Step 3 Name, review, and create

### Trusted entity type

- AWS service Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy Create a custom trust policy to enable others to perform actions in this account.

### Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

Choose a use case for the specified service.

Use case

- CodeDeploy Allows CodeDeploy to call AWS services such as Auto Scaling on your behalf.
- CodeDeploy for Lambda Allows CodeDeploy to route traffic to a new version of an AWS Lambda function version on your behalf.
- CodeDeploy - ECS Allows CodeDeploy to read S3 objects, invoke Lambda functions, publish to SNS topics, and update ECS services on your behalf.

Step 1 Select trusted entity

Step 2 **Add permissions**

Step 3 Name, review, and create

### Add permissions Info

Permissions policies (1) Info

The type of role that you selected requires the following policy.

Policy name	Type
<input checked="" type="checkbox"/> AWSCodeDeployRoleForECS	AWS managed

▶ Set permissions boundary - optional

[Cancel](#) [Previous](#) **Next**

Step 1 Select trusted entity

Step 2 Add permissions

Step 3 **Name, review, and create**

### Name, review, and create

**Role details**

**Role name**  
Enter a meaningful name to identify this role.

**Description**  
Add a short explanation for this role.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: \_+-. @~/[]!#\$%^&`~`

**Step 1: Select trusted entities** [Edit](#)

**Trust policy**

```

1< [{
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "",
6       "Effect": "Allow",
7       "Principal": {
8         "Service": "codedeploy.amazonaws.com"
9       },

```

Another role for CodeBuild

aws Search [Alt+S] Global sailakshmi @ 1851-8779-3433

IAM > Roles > Create role

Role bgcodedeployecrole created. View role X

Step 1 Select trusted entity Step 2 Add permissions Step 3 Name, review, and create

### Select trusted entity Info

**Trusted entity type**

- AWS service Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account Allows entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation Allows users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy Create a custom trust policy to enable others to perform actions in this account.

**Use case**  
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

**Service or use case**  
CodeBuild

Choose a use case for the specified service.  
**Use case**

- CodeBuild Allows CodeBuild to call AWS services on your behalf.

aws Search [Alt+S] Global sailakshmi @ 1851-8779-3433

IAM > Roles > Create role

Role bgcodedeployecrole created. View role X

Step 1 Add permissions Step 2 Name, review, and create

### Permissions policies (2/1066) Info

Choose one or more policies to attach to your new role.

Filter by Type

Policy name	Type	Description
<input checked="" type="checkbox"/>  AmazonEC2ContainerRegistryFullAccess	AWS managed	Provides administrative access to Amazon ECR...
<input type="checkbox"/>  AmazonEC2ContainerRegistryPowerUser	AWS managed	Provides full access to Amazon EC2 Container Registry...
<input type="checkbox"/>  AmazonEC2ContainerRegistryPullOnly	AWS managed	Provides access to pull images from Amazon ECR...
<input type="checkbox"/>  AmazonEC2ContainerRegistryReadOnly	AWS managed	Provides read-only access to Amazon ECR...
<input checked="" type="checkbox"/>  AmazonElasticContainerRegistryPublicFullAccess	AWS managed	Provides administrative access to Amazon ECR...
<input type="checkbox"/>  AmazonElasticContainerRegistryPublicPowerUser	AWS managed	Provides full access to Amazon ECR Power User...
<input type="checkbox"/>  AmazonElasticContainerRegistryPublicReadonlyAccess	AWS managed	Provides read-only access to Amazon ECR...
<input type="checkbox"/>  AmazonSageMakerModelRegistryFullAccess	AWS managed	This is a new managed policy for SageMaker Model Registry...
<input type="checkbox"/>  AWSGlueSchemaRegistryFullAccess	AWS managed	Provides full access to the AWS Glue Schema Registry...
<input type="checkbox"/>  AWSGlueSchemaRegistryReadonlyAccess	AWS managed	Provides readonly access to the AWS Glue Schema Registry...
<input type="checkbox"/>  AWSServiceCatalogAppRegistryFullAccess	AWS managed	Provides full access to Service Catalog App Registry...

← → ⚙️ us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#roles/create?trustedEntityType=AWS\_SERVICE&selectedService=CodeBuild&selectedUseCase=AWS+CodeBuild&policies=arn%3... ☆ ↴

aws Search [Alt+S] Global ▾ sailkshmi @ 1851-8779-3433 ⓘ

IAM > Roles > Create role

Step 1 Select trusted entity

Step 2

Step 3

Name, review, and create

### Name, review, and create

#### Role details

**Role name**  
Enter a meaningful name to identify this role.  
Maximum 64 characters. Use alphanumeric and '+-=.\_@-' characters.

**Description**  
Add a short explanation for this role.  
Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: \_+=., @\_/\{\}\#\$%^&`~^`

### Step 1: Select trusted entities

Edit

#### Trust policy

```
1 * [{  
2     "Version": "2012-10-17",  
3     "Statement": [  
4         {  
5             "Effect": "Allow",  
6             "Action": [  
7                 "sts:AssumeRole"  
8             ],  
9             "Principal": {  
10                "Service": [  
11                    "codebuild.amazonaws.com"  
12                ]  
13            }  
14        }  
15    }]
```

## Step 3: Create ECR Repositories

- Sign in to the AWS Console → go to **ECR** (Elastic Container Registry).
  - Click Create repository
    - Repository name:
  - Save the **ECR URI** (e.g., 123456789012.dkr.ecr.us-east-1.amazonaws.com/bluegreen)

**Repository name**  
Enter a concise name. Repositories support namespaces, which you can use to group similar repositories.  
185187793433.dkr.ecr.us-east-1.amazonaws.com/ **bluegreen**

9 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, and special characters .-/-.

**Image tag mutability**  
Choose the tag mutability setting.  
 **Mutable**  
Image tags can be overwritten.  
 **Immutable**  
Image tags can't be overwritten.

**Encryption settings** Info

**Encryption configuration**  
By default, repositories use the industry standard Advanced Encryption Standard (AES) encryption. You can optionally choose to use a key stored in the AWS Key Management Service (KMS) to encrypt the images in your repository.

**AES-256**  
Industry standard Advanced Encryption Standard (AES) encryption  
 **AWS KMS**  
AWS Key Management Service (KMS)

**Image scanning settings - deprecated**

**Create**

Repository name	URI	Created at	Tag immutability	Encryption type
bluegreen	185187793433.dkr.ecr.us-east-1.amazonaws.com/bluegreen	July 10, 2025, 22:58:29 (UTC-04)	Mutable	AES-256

## Step 4: Push Initial Docker Images

### Git clone

```
git@ip-172-31-40-3: ~ % git clone https://github.com/sailakshmi-d/blue-green-ecs
Cloning into 'blue-green-ecs'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 10 (delta 1), reused 10 (delta 1), pack-reused 0 (from 0)
Receiving objects: 100% (10/10), done.
Resolving deltas: 100% (1/1), done.
[git@ip-172-31-40-3: ~] % cd blue-green-ecs/
[git@ip-172-31-40-3: blue-green-ecs] %
```

### Create a docker image and push image to ECR

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-  
stdin 185187793433.dkr.ecr.us-east-1.amazonaws.com
```

```
[root@ip-172-31-40-3 blue-green-ecs]# aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 185187793433.dkr.ecr.us-east-1.amazonaws.com  
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.  
Configure a credential helper to remove this warning. See  
https://docs.docker.com/engine/reference/commandline/login/#credentials-store  
Login Succeeded  
[root@ip-172-31-40-3 blue-green-ecs]#
```

```
docker build -t bluegreen .
```

```
docker tag bluegreen:latest 185187793433.dkr.ecr.us-east-  
1.amazonaws.com/bluegreen:latest
```

```
docker push 185187793433.dkr.ecr.us-east-1.amazonaws.com/bluegreen:latest
```

```
[> => extracting sha256:5634f043553bc14a2dd0ad7d54bf0ecd8b5e283ce5961d00493b3433323f5c87 0.05  
=> => extracting sha256:98423917571b47a9645b52efb84d20ca5fba2aad821d73b41dfe4fe4aaeb4 0.25  
=> => extracting sha256:66c4cd472d0a7dfb23708b47f95ba9ce7c003e6c5c76d9c71b41ea9ae5218033 0.05  
=> [internal] load build context 0.05  
=> => transferring context: 51.55kB 0.05  
=> [2/5] WORKDIR /usr/src/app 0.05  
=> [3/5] COPY package*.json . 0.05  
=> [4/5] RUN npm install 1.9s  
=> [5/5] COPY . . 0.1s  
=> exporting to image 0.1s  
=> exporting layers 0.1s  
=> => writing image sha256:bf865ba6e92ccdee5633371e026199f2e62f2fef831aaaef8a5fabab7c1e731b 0.05  
=> => naming to docker.io/library/bluegreen 0.05  
[root@ip-172-31-40-3 blue-green-ecs]# docker tag bluegreen:latest 185187793433.dkr.ecr.us-east-1.amazonaws.com/bluegreen:latest  
[root@ip-172-31-40-3 blue-green-ecs]# docker push 185187793433.dkr.ecr.us-east-1.amazonaws.com/bluegreen:latest  
The push refers to repository [185187793433.dkr.ecr.us-east-1.amazonaws.com/bluegreen]  
eba59144031b: Pushed  
584207d11cd6: Pushed  
f579a944bd09f: Pushed  
5f70bf18a096: Pushed  
71e33b7c9ff1: Pushed  
d278df77c3b2: Pushed  
3dec07644066: Pushed  
e474609372a6: Pushed  
6b84dc94dd6: Pushed  
82d5847241f7: Pushed  
b4911aa1197: Pushed  
55c1577568ad: Pushed  
5f22a16e7f43: Pushed  
e15f10fc563f: Pushed  
365de8b4c116: Pushed  
95e78699c821: Pushed  
latest: digest: sha256:5edb97004b65c103435e6319b9a7f2c26be7b5769524ef74635132b54b6b0046 size: 3660  
[root@ip-172-31-40-3 blue-green-ecs]#
```

Now you see the image in ECR

Amazon ECR > Private registry > Repositories > bluegreen

Amazon Elastic Container Registry

Private registry

- Repositories
- Summary
- Images**
- Permissions
- Lifecycle Policy
- Repository tags
- Features & Settings

Public registry

- Repositories
- Settings

ECR public gallery

Amazon ECS

Amazon EKS

Images (1)

Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Last recorded pull time
latest	Image	July 10, 2025, 23:17:08 (UTC-04)	232.92	Copy URI	sha256:5edb97004b65c1...	-

## Step 5: Set Up the Load Balancer

- Go to EC2 → Target Groups
- Create two target groups: blue-target and green-target
- Go to Load Balancers → Create ALB
  - Choose Application Load Balancer
  - Set listeners for port 8080
  - Register the two target groups
  - Add listener rules to forward traffic accordingly

Screenshot of the AWS EC2 Target Groups creation interface:

**Step 1 Specify group details**

Your load balancer routes requests to the targets in a target group and performs health checks on the targets.

**Basic configuration**

Settings in this section can't be changed after the target group is created.

**Choose a target type**

- Instances
  - Supports load balancing to instances within a specific VPC.
  - Facilitates the use of [Amazon EC2 Auto Scaling](#) to manage and scale your EC2 capacity.
- IP addresses
  - Supports load balancing to VPC and on-premises resources.
  - Facilitates routing to multiple IP addresses and network interfaces on the same instance.
  - Offers flexibility with microservice based architectures, simplifying inter-application communication.
  - Supports IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.
- Lambda function
  - Facilitates routing to a single Lambda function.
  - Accessible to Application Load Balancers only.
- Application Load Balancer
  - Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
  - Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.

**Target group name**

TG1

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

**Protocol**

Protocol for load balancer-to-target communication. Can't be modified after creation.

HTTP

**Port**

Port number where targets receive traffic. Can be overridden for individual targets during registration.

8080

**IP address type**

Only targets with the indicated IP address type can be registered to this target group.

IPv4

IPv6

**VPC**

Select the VPC that hosts the load balancer. Only VPCs that support the IP address type selected above are available in this list. On the [Register targets](#) page, you can register IP addresses from this VPC, or from private IP addresses located outside of this load balancer's VPC (such as a peered VPC, EC2-Classic, or on-premises targets that are reachable over Direct Connect or VPN).

vpc-05f0e32fcf899db09  
IPv4 CIDR: 172.31.0.0/16

**Protocol version**

HTTP1

Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.

Create another target group

- Facilitates routing to a single Lambda function.
- Accessible to Application Load Balancers only.

Application Load Balancer

- Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
- Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.

**Target group name**

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

**Protocol**

Protocol for load balancer-to-target communication. Can't be modified after creation.

HTTP

**Port**

Port number where targets receive traffic. Can be overridden for individual targets during registration.

8080

1-65535

**IP address type**

Only targets with the indicated IP address type can be registered to this target group.

IPv4

IPv6

**VPC**

Select the VPC that hosts the load balancer. Only VPCs that support the IP address type selected above are available in this list. On the [Register targets](#) page, you can register IP addresses from this VPC, or from private IP addresses located outside of this load balancer's VPC (such as a peered VPC, EC2-Classic, or on-premises targets that are reachable over Direct Connect or VPN).

vpc-05f0e37cf899db09

aws Search [Alt+S]

EC2 > Target groups

Target groups (2) <span style="color: blue;">Info</span>						
<span style="float: left;">Filter target groups</span> <span style="float: right;"> <a href="#">Actions</a> <span style="color: orange;">▼</span> <a href="#">Create target group</a> </span>						
	Name	ARN	Port	Protocol	Target type	Load balancer
<input type="checkbox"/>	TG2	arn:aws:elasticloadbalancin...	8080	HTTP	IP	<span style="color: blue;">None associated</span>
<input type="checkbox"/>	TG1	arn:aws:elasticloadbalancin...	8080	HTTP	IP	<span style="color: blue;">None associated</span>

**0 target groups selected**

Select a target group above.

Now create a load balancer

Screenshot of the AWS CloudFormation console showing the creation of a new AWS Lambda function named "HelloWorld".

**Basic configuration**

**Load balancer name**  
Name must be unique within your AWS account and can't be changed after the load balancer is created.  
ALB

**Scheme** | **Info**  
Scheme can't be changed after the load balancer is created.

**Internet-facing** (selected)

- Serves internet-facing traffic.
- Has public IP addresses.
- DNS name resolves to public IPs.
- Requires a public subnet.

**Internal**

- Serves internal traffic.
- Has private IP addresses.
- DNS name resolves to private IPs.
- Compatible with the IPv4 and Dualstack IP address types.

**Load balancer IP address type** | **Info**  
Select the front-end IP address type to assign to the load balancer. The VPC and subnets mapped to this load balancer must include the selected IP address types. Public IPv4 addresses have an additional cost.

**IPv4** (selected)  
Includes only IPv4 addresses.

**Dualstack**  
Includes IPv4 and IPv6 addresses.

**Dualstack without public IPv4**  
Includes a public IPv6 address, and private IPv4 and IPv6 addresses. Compatible with **internet-facing** load balancers only.

**Network mapping** | **Info**  
The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

**VPC** | **Info**  
The load balancer will exist and scale within the selected VPC. The selected VPC is also where the load balancer targets must be hosted unless routing to Lambda or on-premises targets, or if using VPC peering. To confirm the VPC for your targets, view [target groups](#). For a new VPC, [create a VPC](#).

vpc-05f0e32fcf899db09  
IPv4 VPC CIDR: 172.31.0.0/16

**Subnet**  
Only CIDR blocks corresponding to the load balancer IP address type are used. At least 8 available IP addresses are required for your load balancer to scale efficiently.  
subnet-05af9b523844db2d7  
IPv4 subnet CIDR: 172.31.64.0/20

**Security groups** | **Info**  
A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can [create a new security group](#).

**Security groups**  
Select up to 5 security groups

default  
sg-01dac83717121cd74 VPC: vpc-05f0e32fcf899db09

**Listeners and routing** | **Info**  
A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

**Listener HTTP:80**  
Protocol: HTTP | Port: 80 | Default action: Forward to TG1 (HTTP)

**Listener tags - optional**  
Consider adding tags to your listener. Tags enable you to categorize your AWS resources so you can more easily manage them.

Go to listener tab and add listner with port 8080

**Successfully created load balancer: ALB**  
It might take a few minutes for your load balancer to fully set up and route traffic. Targets will also take a few minutes to complete the registration process and pass initial health checks.

**Application Load Balancers now support public IPv4 IP Address Management (IPAM)**  
You can get started with this feature by configuring IP pools in the Network mapping section.

**ALB**

**Details**

Load balancer type	Status	VPC	Load balancer IP address type
Application	Provisioning	vpc-05f0e32fcf899db09	IPv4
Scheme	Hosted zone	Availability Zones	Date created
Internet-facing	Z35SXDOTRQ7X7K	subnet-06547f6b14f25be5f (us-east-1e (use1-az3)) subnet-06053826720daabc2 (us-east-1b (use1-az1)) subnet-01d02e41622b80257 (us-east-1d (use1-az4)) subnet-0860021ca5eb65b690 (us-east-1a (use1-az6)) subnet-03b5002f5d76246ba (us-east-1c (use1-az2)) subnet-05af9b523844db2d7 (us-east-1f (use1-az5))	July 10, 2025, 23:23 (UTC-04:00)
		DNS name <a href="#">Info</a>	
	Load balancer ARN		

**Listeners and rules (2) [Info](#)**  
A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

Protocol:Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate
HTTP:8080	Forward to target group • TG2 (100%) • Target group stickiness: Off	1 rule	<a href="#">ARN</a>	Not applicable	Not applicable
HTTP:80	Forward to target group • TG1 (100%) • Target group stickiness: Off	1 rule	<a href="#">ARN</a>	Not applicable	Not applicable

## Step 6: Create ECS Cluster

- Console → ECS → Clusters → Create Cluster → Networking only (Fargate) → Name it ECS cluster created.

**Cluster configuration**

**Cluster name**  
bluegreen\_ecscluster

**Service Connect defaults - optional**

**Infrastructure - optional** Info Serverless  
Your cluster is automatically configured for AWS Fargate (serverless) with two capacity providers. Add Amazon EC2 instances.

**AWS Fargate (serverless)**  
Pay as you go. Use if you have tiny, batch, or burst workloads or for zero maintenance overhead. The cluster has Fargate and Fargate Spot capacity providers by default.

**Amazon EC2 instances**  
Manual configurations. Use for large workloads with consistent resource demands.

i External instances using **ECS Anywhere** can be registered after cluster creation is complete.

**Monitoring - optional** Info  
CloudWatch Container Insights is a monitoring and troubleshooting solution for containerized applications and microservices.

**Encryption - optional**  
Choose the KMS keys used by tasks running in this cluster to encrypt your storage.

**Tags - optional** Info  
Tags help you to identify and organize your clusters.

i On June 25, 2025, Amazon ECS changed the default log driver mode from blocking to non-blocking to improve application availability during CloudWatch outages. [Learn more](#)

**Clusters (1) Info**

View in CloudFormation

Cluster	Services	Tasks	Container instances	CloudWatch monitoring	Capacity provider
bluegreen_ecscluster	0	No tasks running	0 EC2	Default	No default found

## Step 7: Create Task Definitions

- ECS → Task Definitions → Create new:
- Launch type: FARGATE, set CPU (256), Memory (512).
- Add container:
  - Image: ECR URI
  - Port mapping: 8080
- Output: Task definition revision is created (service-a-taskdef:1).

Amazon Elastic Container Service > Create new task definition

On June 25, 2025, Amazon ECS changed the default log driver mode from blocking to non-blocking to improve application availability during CloudWatch outages. [Learn more](#)

## Create new task definition Info

### Task definition configuration

**Task definition family** [Info](#)  
Specify a unique task definition family name.  
  
Up to 255 letters (uppercase and lowercase), numbers, hyphens, and underscores are allowed.

### Infrastructure requirements

Specify the infrastructure requirements for the task definition.

**Launch type** [Info](#)  
Selection of the launch type will change task definition parameters.  
 AWS Fargate  
Serverless compute for containers.  
 Amazon EC2 instances  
Self-managed infrastructure using Amazon EC2 instances.

**OS, Architecture, Network mode**  
Network mode is used for tasks and is dependent on the compute type selected.  
**Operating system/Architecture** [Info](#) **Network mode** [Info](#)

**Task size** [Info](#)  
Specify the amount of CPU and memory to reserve for your task.

us-east-1.console.aws.amazon.com/ecs/v2/create-task-definition?region=us-east-1

## Fault injection - optional

### Container - 1 [Info](#)

**Container details**  
Specify a name, container image, and whether the container should be marked as essential. Each task definition must have at least one essential container.

**Name**  **Image URI**   
Up to 255 letters (uppercase and lowercase), numbers, hyphens, underscores, colons, periods, forward slashes, and number signs are allowed.

**Essential container**  Yes

**Private registry** [Info](#)  
Store credentials in Secrets Manager, and then use the credentials to reference images in private registries.  
 Private registry authentication

**Port mappings** [Info](#)  
Add port mappings to allow the container to access ports on the host to send or receive traffic. For port name, a default will be assigned if left blank.

Container port	Protocol	Port name	App protocol
80	TCP	8080	HTTP

[Add port mapping](#)

**Read only root file system** [Info](#)  
When this parameter is turned on, the container is given read-only access to its root file system.  
 Read only

**Resource allocation limits - conditional** [Info](#)  
Container-level CPU, GPU, and memory limits are different from task-level values. They define how much resources are allocated for the container. If container attempts to exceed the memory specified in hard limit, the container is terminated.

CPU	GPU	Memory hard limit	Memory soft limit
-----	-----	-------------------	-------------------

The screenshot shows the AWS Elastic Container Service (ECS) Task Definitions page. In the top navigation bar, the path is: Amazon Elastic Container Service > Task definitions > bluegreen\_taskdef > Revision 1 > Containers. A success message box is displayed: "Task definition successfully created" with the note "bluegreen\_taskdef:1 has been successfully created. You can use this task definition to deploy a service or run a task." Below this, the task definition name "bluegreen\_taskdef:1" is shown with three buttons: Deploy, Actions, and Create new revision. The main content area is titled "Overview" and includes fields for ARN (arn:aws:ecs:us-east-1:185187793433:task-definition/bluegreen\_taskdef:1), Status (ACTIVE), Time created (July 10, 2025 at 23:38 (UTC-4:00)), App environment (Fargate), Task role (ecsTaskExecutionRole), Task execution role (ecsTaskExecutionRole), Operating system/Architecture (Linux/X86\_64), and Network mode (awsvpc). A "Containers" tab is selected, showing a table with columns: Containers, JSON, Task placement, Volumes (0), Requires attributes, and Tags. The "Task size" section shows Task CPU (256 units (0.25 vCPU)) and Task memory (512 MiB (0.5 GB)).

## Step 8: Create ECS Services (Blue/Green Enabled)

- ECS → Clusters → select cluster → Create → FARGATE service.
- Service name, Task Definition, Set number of tasks and VPC/subnets/security as needed.
- Under Deployment type, choose Blue/Green (CodeDeploy) and select an Application Load Balancer:
  - Set container name/port = 8080.
  - Select target groups before you create.
- Click **Next** and create the service.
- *Output:* Each service shows "Active (GREEN)", target groups in ALB created.

us-east-1.console.aws.amazon.com/ecs/v2/clusters/bluegreen\_ecscluster/create-service?region=us-east-1

Amazon Elastic Container Service > Clusters > bluegreen\_ecscluster > Create service

**Compute configuration (advanced)**

**Compute options** [Info](#)  
To ensure task distribution across your compute types, use appropriate compute options.

Capacity provider strategy  
Specify a launch strategy to distribute your tasks across one or more capacity providers.

Launch type  
Launch tasks directly without the use of a capacity provider strategy.

**Launch type** [Info](#)  
Select either managed capacity (Fargate), or custom capacity (EC2 or user-managed, External instances). External instances are registered to your cluster using the ECS Anywhere capability.

FARGATE

**Platform version** [Info](#)  
Specify the platform version on which to run your service.

LATEST

**Deployment configuration**

**Service type** [Info](#)  
Specify the service type that the service scheduler will follow.

Replica  
Place and maintain a desired number of tasks across your cluster.

Daemon  
Place and maintain one copy of your task on each container instance.

**Desired tasks**  
Specify the number of tasks to launch.  
1

**Availability Zone rebalancing** [Info](#)  
Turn on Availability Zone rebalancing

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

aws Search [Alt+S] United States (N. Virginia) sailakshmi @ 1851-8779-5433

Amazon Elastic Container Service > Clusters > bluegreen\_ecscluster > Create service

**Deployment type** [Info](#)  
Select a deployment controller type for the service.

Rolling update

Blue/green deployment (powered by AWS CodeDeploy)

**Deployment configuration** [Info](#)  
CodeDeployDefault.ECSAllAtOnce

**Service role for CodeDeploy**  
Specify the IAM role the service uses to make API requests to authorized AWS services. Create a service role for CodeDeploy in IAM console. [Learn more](#)

arn:aws:iam::185187793433:role/bgcodedeployrole

**Networking**

**Service Connect - optional** [Info](#)  
Service Connect allows for service-to-service communications with automatic discovery using short names and standard ports.

**Service discovery - optional**  
Service discovery uses Amazon Route 53 to create a namespace for your service, which allows it to be discoverable via DNS.

**Load balancing**  
Configure load balancing using Amazon Elastic Load Balancing to distribute traffic evenly across the healthy tasks in your service.

Use load balancing

VPC

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Amazon Elastic Container Service > Clusters > bluegreen\_ecscluster > Create service

### Networking

VPC | Info Select a VPC to use for your Amazon ECS resources.

vpc-05f0e32fcfb99db09 default

**Subnets** Choose the subnets within the VPC that the task scheduler should consider for placement.

subnet-06547f6b14f25be5f us-east-1a 172.31.48.0/20  
subnet-06053826720daabc2 us-east-1b 172.31.0.0/20  
subnet-01d02e41622b80257 us-east-1d 172.31.16.0/20

subnet-086021ca5eb65b690 us-east-1a 172.31.32.0/20  
subnet-03b5002f5d76246ba us-east-1c 172.31.80.0/20  
subnet-05af9b523844db2d7 us-east-1f 172.31.64.0/20

**Security group** | Info Choose an existing security group or create a new security group.

Use an existing security group  
 Create a new security group

**Security group name** Choose an existing security group.

sg-01dac83717121cd74 default

**Public IP** | Info Choose whether to auto-assign a public IP to the task's elastic network interface (ENI).

Turned on

Amazon Elastic Container Service > Clusters > bluegreen\_ecscluster > Create service

### bluegreen 80:80

Host port:Container port

**Application Load Balancer** Specify whether to create a new load balancer or choose an existing one.

Create a new load balancer  
 Use an existing load balancer

**Load balancer** Choose an existing load balancer to distribute traffic. View existing load balancers and create new one in EC2 Console.

ALB ALB-1059582612.us-east-1.elb.amazonaws.com internet-facing

**Listeners** | Info Specify the port and protocol that the load balancer will listen for connection requests on.

**Production listener**

Create new listener  
 Use an existing listener

**Production listener** HTTP:80

**Listener rules for 80:HTTP (1)** Traffic received by the listener is routed according to its rules. Rules are evaluated in priority order, from the lowest value to the highest value. The default rule is evaluated last.

Priority	Rule path	Target group
default	/	TG1 <input type="button" value="Edit"/>

**Test listener**  Add a test listener

The screenshot shows the 'Create service' wizard in the AWS Elastic Container Service (ECS) console. On the left, there's a sidebar with navigation links for Clusters, Amazon ECR, AWS Batch, Documentation, and Tell us what you think. The main form is titled 'Create service' and contains two sections for target groups:

- Target group 1:** Name is set to 'TG1', health check path is '/', and health check protocol is 'HTTP'.
- Target group 2:** Name is set to 'TG2', health check path is '/', and health check protocol is 'HTTP'.

Below the target groups, there are two notes:

- VPC Lattice - optional**: Fully managed application networking service to connect, secure, and monitor your services across multiple accounts and virtual private clouds (VPCs). When you use VPC Lattice, there is a cost associated with it.
- Service auto scaling - optional**: Automatically adjust your service's desired count up and down within a specified range in response to CloudWatch alarms. You can modify your service auto scaling configuration at any time to meet the needs of your application.

## Step 9: Create CodeBuild Project

- CodeBuild → Create build project:
  - Connect GitHub repo
  - Use `buildspec.yml` (see below)
  - Set environment to use Docker runtime
  - Grant necessary IAM permissions

aws | Search [Alt+S]

Developer Tools > CodeBuild > Build projects > Create build project

## Create build project

**Project configuration**

Project name  
bluegreen\_codebuild

A project name must be 2 to 255 characters. It can include the letters A-Z and a-z, the numbers 0-9, and the special characters - and \_.

Project type  
Select what type of project you would like to create. [Info](#)

Default project  
Create a custom CodeBuild project.

Runner project  
Create a CodeBuild managed runner for workflows in GitHub Actions, GitHub Enterprise Actions, GitLab, or Buildkite.

► Additional configuration  
Description, public build access, build badge, concurrent build limit, tags

**Source** [Add source](#)

**Source 1 - Primary**

Source provider  
GitHub

Credential  
 Your account is successfully connected by using an AWS managed GitHub App. [Manage account credentials](#).

Use override credentials for this project only

Connection  
You can [create a new GitHub connection](#) by using an AWS managed GitHub App, [create a new Oauth app token connection](#), or [create a new personal access token connection](#).

arn:aws:codeconnections:us-east-1:185187793433:connection/5... [C](#)

us-east-1.console.aws.amazon.com/codesuite/codebuild/project/new?region=us-east-1

aws Search [Alt+S]

☰ Environment

Provisioning model [Info](#)

On-demand  
Automatically provision build infrastructure in response to new builds.

Reserved capacity  
Use a dedicated fleet of instances for builds. A fleet's compute and environment type will be used for the project.

Environment image

Managed image  
Use an image managed by AWS CodeBuild

Custom image  
Specify a Docker image

Compute

EC2  
Optimized for flexibility during action runs

Lambda  
Optimized for speed and minimizes the start up time of workflow actions

Running mode

Container  
Running on Docker container

Instance  
Running on EC2 instance directly

Operating system

Amazon Linux

Runtime(s)

Standard

Image

aws/codebuild/amazonlinux-x86\_64-standard:5.0

Image version

Always use the latest image for this runtime version

Use GPU-enhanced compute

Service role

New service role |  Existing service role

us-east-1.console.aws.amazon.com/codesuite/codebuild/project/new?region=us-east-1

The screenshot shows the 'Repository' section of the AWS CodeBuild project creation wizard. It includes fields for selecting a GitHub account (radio button selected), entering a GitHub URL (https://github.com/sailakshmi-d/blue-green-ecs), and an optional source version. Below this is an 'Additional configuration' section for Git clone depth, submodules, and build status config.

**Primary source webhook events**

Webhook - optional info: Rebuild every time a code change is pushed to this repository (checkbox checked).

Build type: Single build (radio button selected, triggers single build) or Batch build (Triggers multiple builds as single execution).

**Webhook event filter groups**: A build is triggered if any filter group evaluates to true, which occurs when all the filters in the group evaluate to true. An 'Add filter group' button is present.

**Additional configuration**

**Environment**

Provisioning model info: On-demand (radio button selected, automatically provisions infrastructure) or Reserved capacity (uses a dedicated fleet of instances for builds).

The screenshot shows the AWS CodeBuild console under the 'Build projects' section. The sidebar includes links for Source, Artifacts, Build, and Deploy. The main area displays a table of build projects:

Name	Source provider	Repository	Latest build status	Description	Last Modified
bluegreen_codebuild	GitHub	sailakshmi-d/blue-green-ecs	-	-	Just now

## Step 10: Create a Unified CodePipeline

- **CodePipeline** → **Create pipeline** → Name
- Connect to your repo (GitHub, etc.).
- Add **Build stage**:
  - Use AWS CodeBuild.

- buildspec.yaml builds both containers, pushes to ECR and writes imagedefinitions.json.
- Add **Deploy stage**:
  - Add **CodeDeploy ECS actions**:

us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipeline/new?region=us-east-1

Choose creation option Step 3 of 7

**Source**

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

GitHub (via GitHub App)

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

Q arn:aws:codeconnectionsus-east-1:185187793433:connection/58e4 X C or Connect to GitHub

Repository name  
Choose a repository in your GitHub account.

Q sailakshni-d/blue-green-ecs X

You can type or paste the group path to any project that the provided credentials can access. Use the format 'group/subgroup/project'.

Default branch  
Default branch will be used only when pipeline execution starts from a different source or manually started.

Q master X

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. Learn more

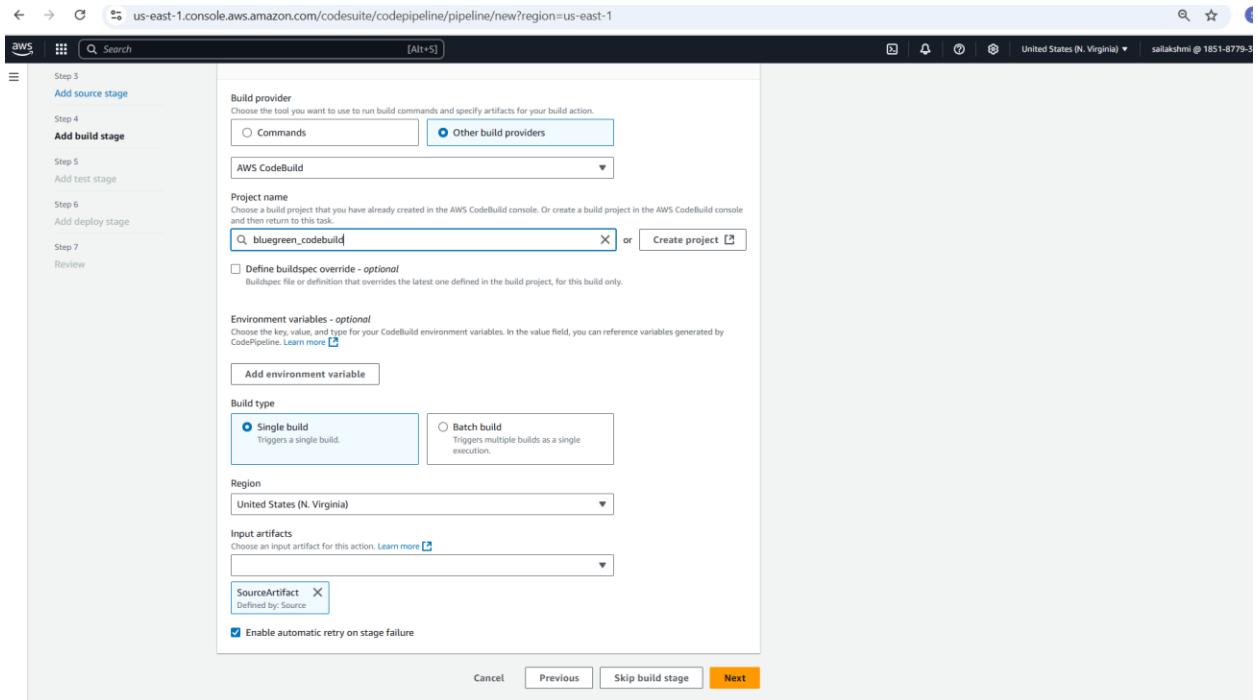
Enable automatic retry on stage failure

**Webhook events**

Webhook - optional  
 Start your pipeline on push and pull request events.

► Webhook event filters - optional

Remove filters



## Create CodeDeploy Applications & Deployment Groups

- Console → CodeDeploy → Applications → Create Application:
  - Choose platform: Amazon ECS
  - Name it.
- Create a Deployment Group:
  - Name it
  - Connect to cluster and service-a-prod service
  - Use the ALB and specify production/green target groups.
  - Choose Blue/Green default traffic shifting.
- Shows each deployment group with Blue/Green traffic shifting enabled.

AWS | Search [Alt+S]

Developer Tools > CodeDeploy > Applications > Create application

## Create application

**Application configuration**

**Application name**  
Enter an application name  
 100 character limit

**Compute platform**  
Choose a compute platform

**Tags**

Cancel

## Deployment group

us-east-1.console.aws.amazon.com/codesuite/codedeploy/applications/bluegreen\_deployapp/deployment-groups/new?region=us-east-1

aws | Search [Alt+S]

Choose an ECS service name

**Load balancers**

Choose a load balancer

Production listener port

Test listener port - optional  
A test listener is required if you want to test your replacement version before traffic reroutes to it

Target group 1 name

Target group 2 name

**Deployment settings**

Traffic rerouting  
Choose whether traffic reroutes to the replacement environment immediately or waits for you to start the rerouting process  
 Reroute traffic immediately  
 Specify when to reroute traffic

Deployment configuration  
Choose from a list of default and custom deployment configurations. A deployment configuration is a set of rules that determines how fast an application is deployed and the success or failure conditions for a deployment.  
 or

aws | Search [Alt+S]

Developer Tools > CodeDeploy > Applications > bluegreen\_deployapp > Create deployment group

## Create deployment group

**Application**

Application  
bluegreen\_deployapp  
Compute type  
Amazon ECS

**Deployment group name**

Enter a deployment group name  
bluegreen\_deploymentgroup  
100 character limit

**Service role**

Enter a service role  
Enter a service role with CodeDeploy permissions that grants AWS CodeDeploy access to your target instances.  
arn:aws:iam::185187793433:role/bcdodedeployecrrole

**Environment configuration**

Choose an ECS cluster name  
bluegreen\_ecscluster

Choose an ECS service name  
bluegreen\_taskdef-service-8hplcza8

---

Go to pipeline and add code deploy

us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipeline/new?region=us-east-1

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1 Choose creation option

Step 2 Choose pipeline settings

Step 3 Add source stage

Step 4 Add build stage

Step 5 Add test stage

**Step 6 Add deploy stage**

Step 7 Review

### Add deploy stage Info

Step 6 of 7

**Deploy - optional**

**Deploy provider**  
Choose how you want to deploy your application or content. Choose the provider, and then provide the configuration details for that provider.

AWS CodeDeploy

**Region**  
United States (N. Virginia)

**Input artifacts**  
Choose an input artifact for this action. Learn more ?

BuildArtifact X  
Defined by: build  
No more than 100 characters

**Application name**  
Choose an application that you have already created in the AWS CodeDeploy console. Or create an application in the AWS CodeDeploy console and then return to this task.

Q bluegreen\_deployapp X

**Deployment group**  
Choose a deployment group that you have already created in the AWS CodeDeploy console. Or create a deployment group in the AWS CodeDeploy console and then return to this task.

Q bluegreen\_deploymentgroup X

Configure automatic rollback on stage failure  
 Enable automatic retry on stage failure

General Previous Next Cancel Save

aws us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipeline/new?region=us-east-1

Enter a service role  
Enter a service role with CodeDeploy permissions that grants AWS CodeDeploy access to your target instances.

Q arn:aws:iam::185187793433:role/bgcodedeployecsrrole X

### Environment configuration

Choose an ECS cluster name  
bluegreen\_ecscluster

Choose an ECS service name  
bluegreen\_taskdef-service-8hplcza8

### Load balancers

Choose a load balancer  
ALB

Production listener port  
HTTP: 8080

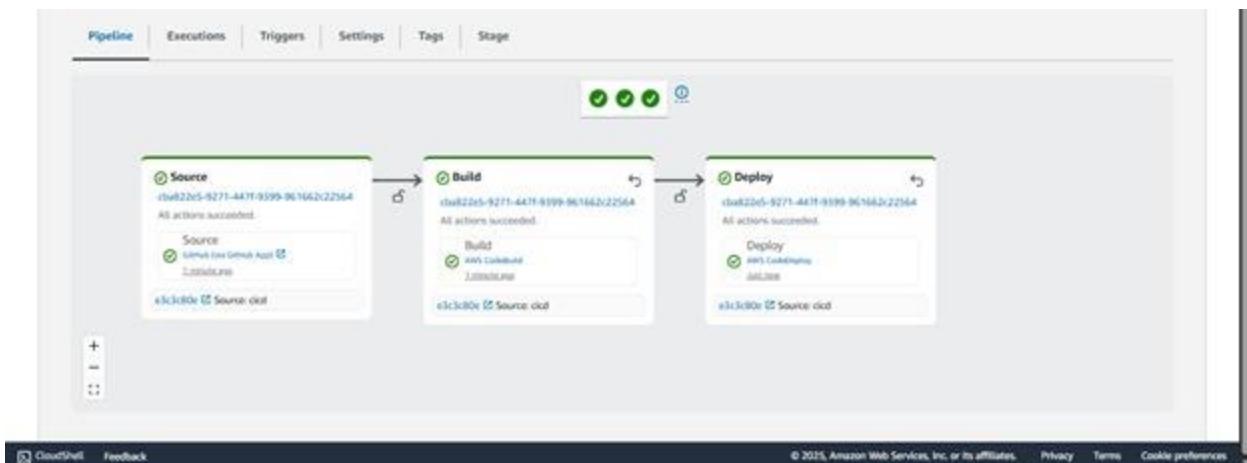
**Test listener port - optional**  
A test listener is required if you want to test your replacement version before traffic reroutes to it.

TG1

Target group 1 name  
TG1

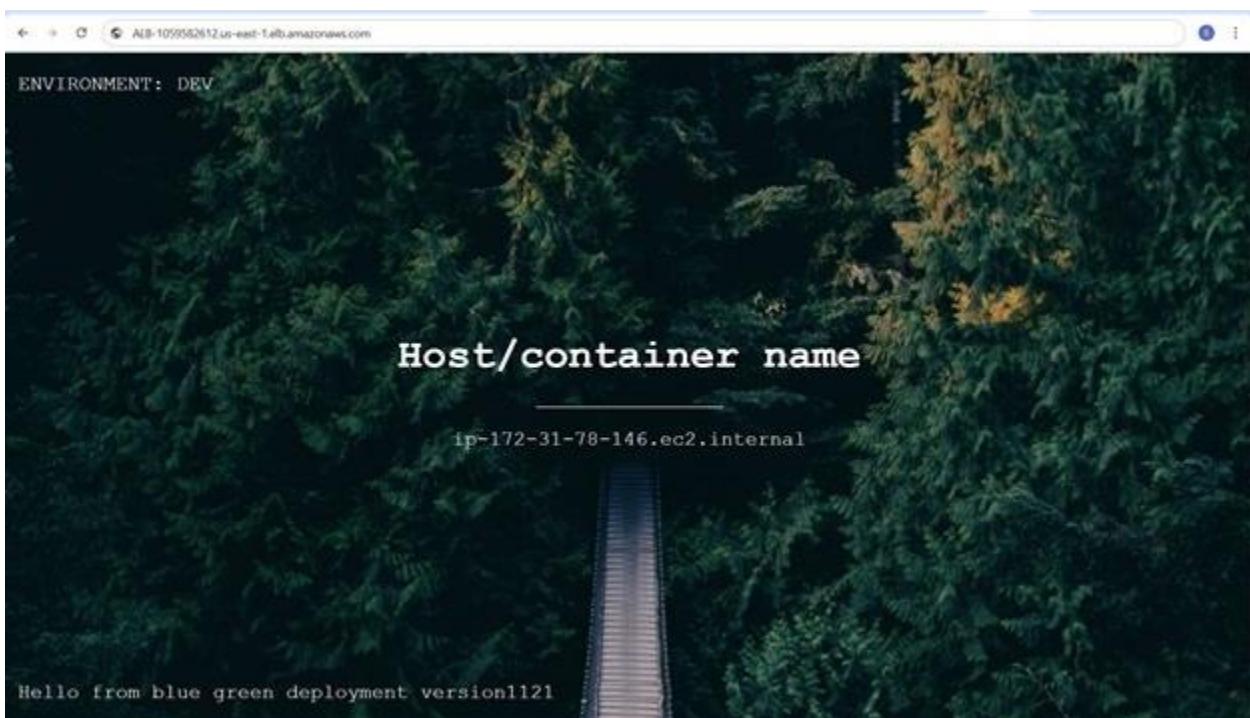
Target group 2 name  
TG2

### Deployment settings



## Step:11 After successful deployment

Check the dns name of load balancer in browser



## Step: 12 Redeployment Flow

- Change the source code in server.js
- Commit and push to GitHub
- Pipeline triggers automatically

