



FROM MANUAL DEPLOYMENT TO CI/CD AUTOMATION

Docker Deployment on AWS ECS



GitHub Repository:<https://github.com/DHAVANISHAJ/Manual-to-CI-CD-Automation--Docker-Deployment-on-AWS-ECS.git>

Presented by **Dhavanisha J**

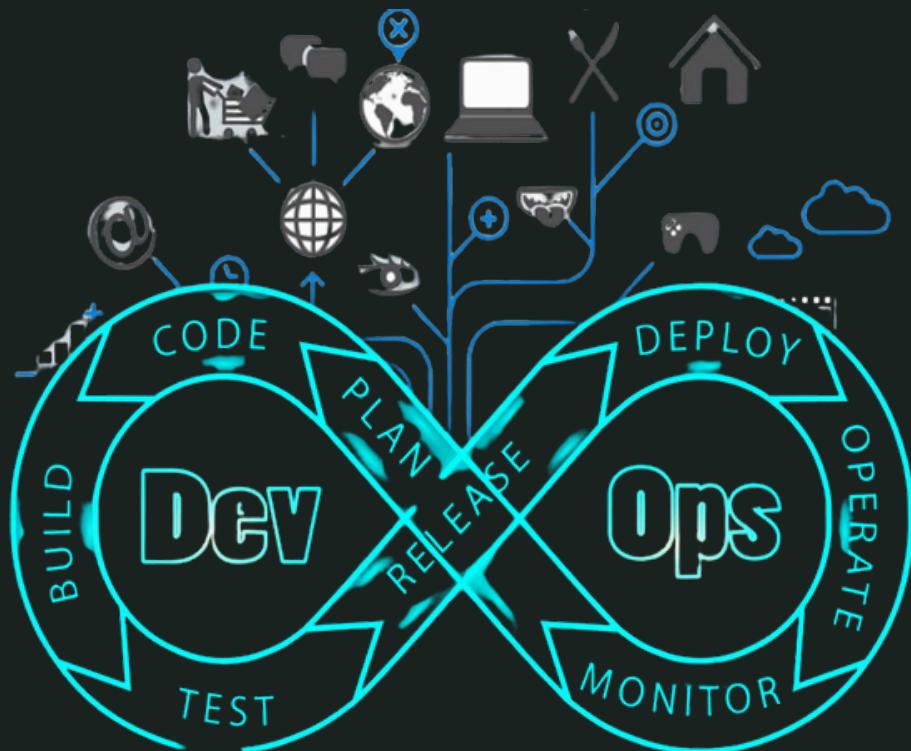


INTRODUCTION

This project focuses on automating the deployment of Dockerized applications on AWS ECS using a CI/CD pipeline. The deployment was first validated through a manual setup to understand the architecture and behavior, followed by full automation using AWS CodeCommit, CodePipeline, and CodeBuild. This approach reduces manual effort, improves reliability, and aligns with modern DevOps best practices.



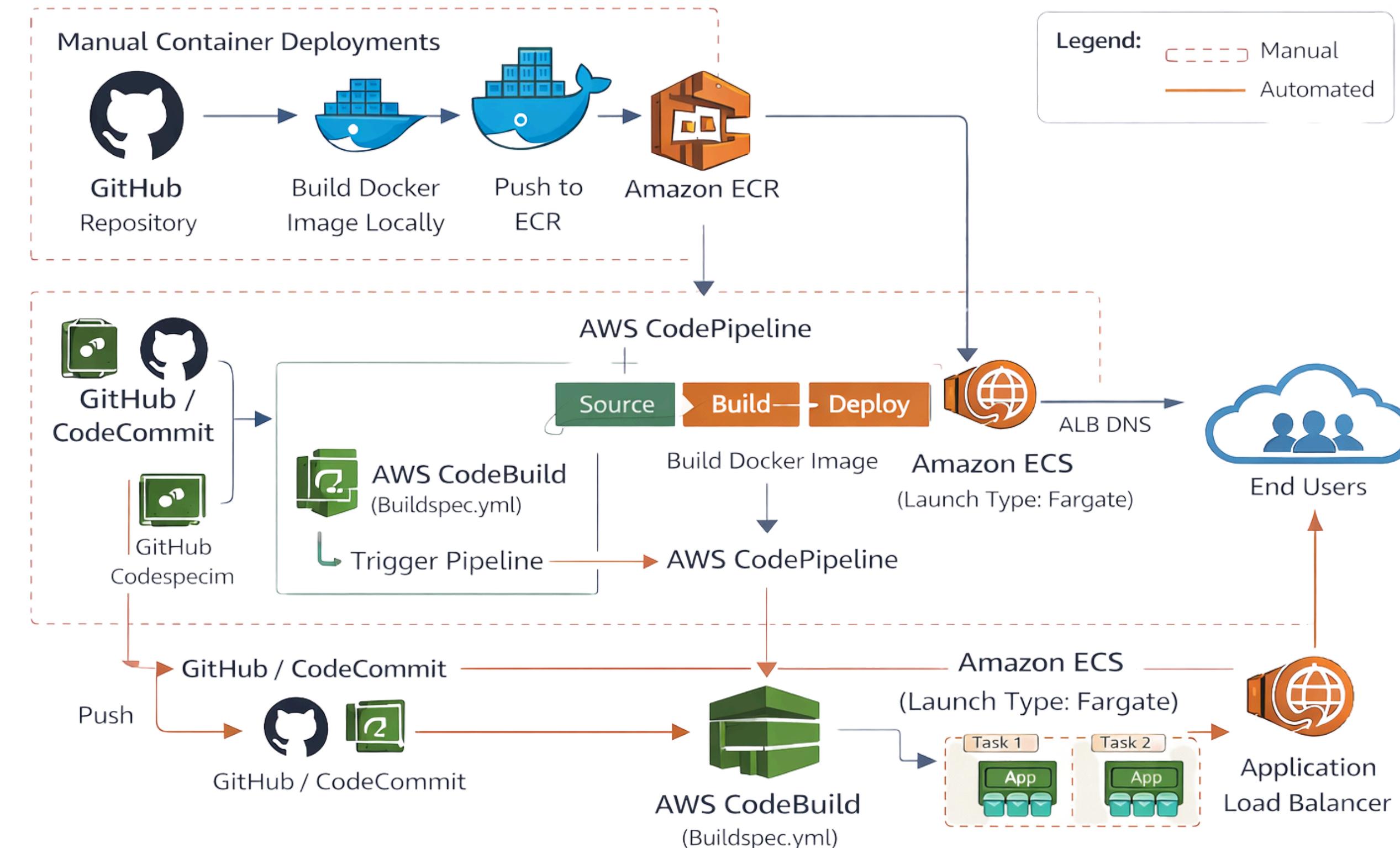
Tools & Technologies Used



- IAM - IAM role was configured with permissions for ECR, ECS, and CodeCommit to securely interact with AWS services through the AWS CLI.
- VPC - Custom networking setup
- Security Group - Allows required traffic between the Load Balancer, ECS tasks and services.
- EC2 - Compute for ECS and Docker
- ECR - Amazon ECR was used to store Docker images securely. The CI/CD pipeline pushed built images to ECR, and ECS pulled images from ECR during deployment.
- ECS - Amazon ECS was used to run and manage Docker containers. ECS task definitions and services controlled how containers were deployed, scaled, and maintained.
- Application Load Balancer - An Application Load Balancer was used to route incoming user traffic to ECS tasks. It ensured high availability and distributed traffic across running containers.
- CodeCommit - AWS CodeCommit was used as the source code repository. Any code changes in CodeCommit triggered the CI/CD pipeline, starting the automated build and deployment process.

ARCHITECTURE

MANUAL TO CI/CD AUTOMATION: DOCKER DEPLOYMENT ON AWS ECS



Phase 1: Manual Setup & Validation (Foundation Layer)

Purpose: Understand, validate, and de-risk the architecture

ECS on EC2 launch type

Manual creation of:

- VPC, subnets, security groups
- EC2 instances with Docker
- ECR repositories
- ECS task definitions & services
- Application Load Balancer & target groups

Applications tested using:

- EC2 public IP
- ALB DNS

This phase proves infrastructure understanding and debugging skills



Phase 2: Fully Automated CI/CD Deployment (Production Layer)

Purpose: Eliminate manual steps and enable repeatable deployments

Source control via AWS CodeCommit

AWS CodePipeline orchestrates the flow

- AWS CodeBuild:
- Builds Docker images
- Pushes to Amazon ECR

Amazon ECS (Fargate):

- Zero EC2 management
- Stateless container deployment

Automated deployment validation:

- Target group health checks
- Old tasks draining
- New tasks serving traffic via ALB



EC2 instances were launched inside the VPC and Docker was installed to run containers.

The screenshot shows the AWS Management Console interface for the EC2 service. The top navigation bar includes the AWS logo, a search bar, and account information (Account ID: 5159-6652-1120, United States (N. Virginia), Dhavanisha). The main menu bar has links for VPC, EC2, Aurora and RDS, IAM, S3, Route 53, and Elastic Container Registry. The current view is under the EC2 menu, specifically the Instances section.

The main content area displays the 'Instances (1/1)' page. A summary table shows one instance:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
Docker-VM	i-073d175132f0178bf	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-100-48-

Below the table, the details for the instance i-073d175132f0178bf (Docker-VM) are shown. The 'Details' tab is selected, followed by Status and alarms, Monitoring, Security, Networking, Storage, and Tags. The Instance summary section displays the following information:

Instance ID	Public IPv4 address	Private IPv4 addresses
i-073d175132f0178bf	100.48.91.100 open address ↗	10.0.23.4

The left sidebar contains navigation links for EC2 (Dashboard, Global View, Events), Instances (Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager), Images, and Elastic Block Store (Volumes, Snapshots, Lifecycle Manager).

The EC2 instance was accessed via SSM, Docker was installed, and its installation was verified successfully.

Installed:

```
container-selinux-4:2.242.0-1.amzn2023.noarch
iptables-libs-1.8.8-3.amzn2023.0.2.x86_64
libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64
pigz-2.5-1.amzn2023.0.3.x86_64
```

```
containerd-2.1.5-1.amzn2023.0.1.x86_64
iptables-nft-1.8.8-3.amzn2023.0.2.x86_64
libnftnetlink-1.0.1-19.amzn2023.0.2.x86_64
runc-1.3.3-2.amzn2023.0.1.x86_64
```

```
docker-25.0.13-1.amzn2023.0.2.x86_64
libcgroup-3.0-1.amzn2023.0.1.x86_64
libnftnl-1.2.2-2.amzn2023.0.2.x86_64
```

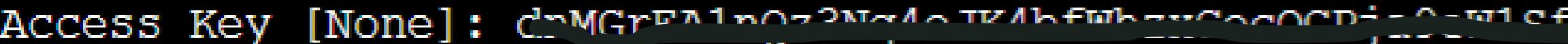
Complete!

```
[root@ip-10-0-23-4 ~]# service docker start
Redirecting to /bin/systemctl start docker.service
[root@ip-10-0-23-4 ~]# docker --version
Docker version 25.0.13, build 0bab007
```

An IAM role was configured with permissions for ECR, ECS, and CodeCommit to securely interact with AWS services through the AWS CLI.

The screenshot shows the AWS IAM 'Create access key' interface. At the top, there's a navigation bar with the AWS logo, search bar, and various service links like VPC, EC2, Aurora and RDS, IAM, S3, Route 53, and Elastic Container Registry. The IAM link is highlighted. Below the navigation is a breadcrumb trail: IAM > Users > Dhavanisha > Create access key. On the left, a sidebar lists three steps: Step 1 (Access key best practices & alternatives), Step 2 - optional (Set description tag), and Step 3 (Retrieve access keys), with Step 3 currently selected. A green banner at the top states: "This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time." The main content area is titled 'Retrieve access keys' and contains sections for 'Access key' and 'Secret access key'. Both fields are displayed as redacted black rectangles. A 'Show' link is visible next to the secret access key field. Below this, a section titled 'Access key best practices' provides instructions: "If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive." The top right corner of the interface shows the account ID (5159-6652-1120) and the user name (Dhavanisha).

The IAM access key was generated and configured on the server using the aws configure command.

```
[root@ip-10-0-23-4 ecs_node]# aws --version
aws-cli/2.32.1 Python/3.9.25 Linux/6.1.158-180.294.amzn2023.x86_64 source/x86_64.amzn.2023
[root@ip-10-0-23-4 ecs_node]# aws configure
AWS Access Key ID [None]: 
AWS Secret Access Key [None]: 
Default region name [None]: us-east-1
Default output format [None]: [root@ip-10-0-23-4 ecs_node]#
[root@ip-10-0-23-4 ecs_node]# █
```

Two separate application directories were created, and Dockerfile and index.html files were added for each application.

```
[root@ip-10-0-23-4 ~]# mkdir app_1
[root@ip-10-0-23-4 ~]# cd app_1/
[root@ip-10-0-23-4 app_1]# vim Dockerfile
[root@ip-10-0-23-4 app_1]# vim index.html
[root@ip-10-0-23-4 app_1]# ls
Dockerfile  index.html
[root@ip-10-0-23-4 app_1]# cd ..
[root@ip-10-0-23-4 ~]# mkdir app_2
[root@ip-10-0-23-4 ~]# cd app_2/
[root@ip-10-0-23-4 app_2]# vim Dockerfile
[root@ip-10-0-23-4 app_2]# vim index.html
[root@ip-10-0-23-4 app_2]# ls
Dockerfile  index.html
[root@ip-10-0-23-4 app_2]# █
```

Built Docker images for both applications successfully.

```
[root@ip-10-0-23-4 ~]# cd app_1/
[root@ip-10-0-23-4 app_1]# docker build -t app_1:latest .
[+] Building 0.3s (7/7) FINISHED                                            docker:default
=> [internal] load build definition from Dockerfile                      0.0s
=> => transferring dockerfile: 323B                                         0.0s
=> [internal] load metadata for docker.io/library/nginx:alpine           0.1s
=> [internal] load .dockerignore                                         0.0s
=> => transferring context: 2B                                           0.0s
=> [internal] load build context                                         0.0s
=> => transferring context: 236B                                         0.0s
=> CACHED [1/2] FROM docker.io/library/nginx@sha256:052b75ab72f690f33debaa51c7e08d9b969a0447a133eb2b99cc905d9188cb2b 0.0s
=> [2/2] COPY index.html /usr/share/nginx/html/                           0.1s
=> exporting to image                                                 0.0s
=> => exporting layers                                              0.0s
=> => writing image sha256:69915efa822eee1159318854eacc4ff83ea633ec93551c98f6b2af6f54f5e5de 0.0s
=> => naming to docker.io/library/app_1:latest                         0.0s
[root@ip-10-0-23-4 app_1]# cd ..
[root@ip-10-0-23-4 ~]# cd app_2/
[root@ip-10-0-23-4 app_2]# docker build -t app_2:latest .
[+] Building 0.3s (7/7) FINISHED                                            docker:default
=> [internal] load build definition from Dockerfile                      0.0s
=> => transferring dockerfile: 323B                                         0.0s
=> [internal] load metadata for docker.io/library/nginx:alpine           0.1s
=> [internal] load .dockerignore                                         0.0s
=> => transferring context: 2B                                           0.0s
=> [internal] load build context                                         0.0s
=> => transferring context: 88B                                         0.0s
=> [1/2] FROM docker.io/library/nginx@sha256:052b75ab72f690f33debaa51c7e08d9b969a0447a133eb2b99cc905d9188cb2b 0.0s
=> CACHED [2/2] COPY index.html /usr/share/nginx/html/                     0.0s
=> exporting to image                                                 0.0s
=> => exporting layers                                              0.0s
=> => writing image sha256:191918751b975018f9dc8c43f44801a00fa966ee91374d5b5381d3a5fbff076e 0.0s
=> => naming to docker.io/library/app_2:latest                         0.0s
[root@ip-10-0-23-4 app_2]# docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
app_1           latest   69915efa822e  52 seconds ago  53.7MB
app_2           latest   191918751b97  5 minutes ago   53.7MB
[root@ip-10-0-23-4 app_2]# █
```

The first application container was run and exposed on port 4000, and the second application container was run and exposed on port 8080.

```
[root@ip-10-0-23-4 app_2]# docker run -it -p 4000:80 app_1:latest
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/12/15 14:04:46 [notice] 1#1: using the "epoll" event method
2025/12/15 14:04:46 [notice] 1#1: nginx/1.29.4
2025/12/15 14:04:46 [notice] 1#1: built by gcc 15.2.0 (Alpine 15.2.0)
2025/12/15 14:04:46 [notice] 1#1: OS: Linux 6.1.158-180.294.amzn2023.x86_64
2025/12/15 14:04:46 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 32768:65536
2025/12/15 14:04:46 [notice] 1#1: start worker processes
2025/12/15 14:04:46 [notice] 1#1: start worker process 30
103.6.156.129 - - [15/Dec/2025:14:05:07 +0000] "GET / HTTP/1.1" 200 141 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
```

```
[root@ip-10-0-23-4 ~]# docker run -d -p 8080:80 app_2:latest
15506f09e32a05a09d9413fd9f15dc72c4ba88b0af87b9f17027614cf8c0cec
[root@ip-10-0-23-4 ~]#
```

Verified Application 1 running successfully in the browser using public ip.



Verified Application 2 running successfully in the browser.



ECR repositories were created.

The screenshot shows the AWS ECR console interface. On the left, there is a navigation sidebar with sections for Amazon Elastic Container Registry, Private registry (selected), Public registry, ECR public gallery, Amazon ECS, and Amazon EKS. The main content area displays a success message: "Successfully created private repository, app_2". Below this, the "Private repositories (2)" section is shown, featuring a search bar and a table with columns: Repository name, URI, Created at, Tag immutability, and Encryption type. The table contains two rows: app_1 and app_2. Both repositories were created on December 15, 2025, at different times (21:01:30 and 21:02:08 UTC+05.5). They are both mutable and encrypted with AES-256.

Repository name	URI	Created at	Tag immutability	Encryption type
app_1	51966521120.dkr.ecr.us-east-1.amazonaws.com/app_1	December 15, 2025, 21:01:30 (UTC+05.5)	Mutable	AES-256
app_2	51966521120.dkr.ecr.us-east-1.amazonaws.com/app_2	December 15, 2025, 21:02:08 (UTC+05.5)	Mutable	AES-256

Used AWS-provided commands to authenticate Docker with ECR for both Repositories.

The screenshot shows the AWS ECR console with a modal window titled "Push commands for app_1". The modal is divided into two tabs: "macOS / Linux" (selected) and "Windows".

macOS / Linux

Make sure that you have the latest version of the AWS CLI and Docker installed. For more information, see [Getting Started with Amazon ECR](#).

Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see [Registry Authentication](#).

1. Retrieve an authentication token and authenticate your Docker client to your registry. Use the AWS CLI:
`aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 515966521120.dkr.ecr.us-east-1.amazonaws.com`
2. Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions [here](#). You can skip this step if your image is already built:
`docker build -t app_1 .`
3. After the build completes, tag your image so you can push the image to this repository:
`docker tag app_1:latest 515966521120.dkr.ecr.us-east-1.amazonaws.com/app_1:latest`
4. Run the following command to push this image to your newly created AWS repository:
`docker push 515966521120.dkr.ecr.us-east-1.amazonaws.com/app_1:latest`

Close

The background shows the ECR console with a repository named "app_1" selected. The "View push commands" button is highlighted in orange.

Tagged and pushed Docker images to Amazon ECR successfully.

```
[root@ip-10-0-23-4 ~]# aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 515966521120.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-10-0-23-4 ~]# docker tag app_1:latest 515966521120.dkr.ecr.us-east-1.amazonaws.com/app_1:latest
[root@ip-10-0-23-4 ~]# docker push 515966521120.dkr.ecr.us-east-1.amazonaws.com/app_1:latest
The push refers to repository [515966521120.dkr.ecr.us-east-1.amazonaws.com/app_1]
13b7950a53e4: Pushed
7a44d12423b9: Pushed
a2ec75146a7f: Pushed
dcfd04fc2e09: Pushed
45178d7e9376: Pushed
9f71ae529e9d: Pushed
44227ec39841: Pushed
e4531687ef8e: Pushed
5aa68bbbc67e: Pushed
latest: digest: sha256:c71389ac0aab87a42425ad7f4c92d1ca6cc6c4ca366648ada14e2fdd0e22719b size: 2196
[root@ip-10-0-23-4 ~]# █
```

AWS | Search [Alt+S] | Account ID: 5159-6652-1120 ▾ Dhavanisha | United States (N. Virginia) ▾

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry

Amazon ECR > Private registry > Repository

Amazon Elastic Container Registry

Private registry

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

Public registry

- Repositories
- Settings

ECR public gallery ↗

Amazon ECS ↗

Amazon EKS ↗

Getting started ↗

Documentation ↗

Push commands for app_2

macOS / Linux Windows

Make sure that you have the latest version of the AWS CLI and Docker installed. For more information, see [Getting Started with Amazon ECR ↗](#).

Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see [Registry Authentication ↗](#).

1. Retrieve an authentication token and authenticate your Docker client to your registry. Use the AWS CLI:
 `aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 515966521120.dkr.ecr.us-east-1.amazonaws.com`

Note: If you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.

2. Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions [here ↗](#). You can skip this step if your image is already built:
 `docker build -t app_2 .`
3. After the build completes, tag your image so you can push the image to this repository:
 `docker tag app_2:latest 515966521120.dkr.ecr.us-east-1.amazonaws.com/app_2:latest`
4. Run the following command to push this image to your newly created AWS repository:
 `docker push 515966521120.dkr.ecr.us-east-1.amazonaws.com/app_2:latest`

Close

```
[root@ip-10-0-23-4 ~]# docker tag app_2:latest 515966521120.dkr.ecr.us-east-1.amazonaws.com/app_2:latest
[root@ip-10-0-23-4 ~]# docker push 515966521120.dkr.ecr.us-east-1.amazonaws.com/app_2:latest
The push refers to repository [515966521120.dkr.ecr.us-east-1.amazonaws.com/app_2]
ce3a41100a11: Pushed
7a44d12423b9: Pushed
a2ec75146a7f: Pushed
dcfd04fc2e09: Pushed
45178d7e9376: Pushed
9f71ae529e9d: Pushed
44227ec39841: Pushed
e4531687ef8e: Pushed
5aa68bbc67e: Pushed
latest: digest: sha256:1b35ba79f921af77d9c10b30a217165830a0ce624f5fa761ea9de5f6254d1118 size: 2196
[root@ip-10-0-23-4 ~]#
```

AWS | Search [Alt+S] Account ID: 5159-6652-1120
VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry United States (N. Virginia) Dhavanisha

Amazon ECR > Private registry > Repositories > Images

Amazon Elastic Container Registry < app_1

Private registry

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

Public registry

Repositories

Images (1) Info

Filter active images

Image tags Type Created at Image size Image digest Last pulled at

Image tags	Type	Created at	Image size	Image digest	Last pulled at
<input type="checkbox"/> latest	Image	December 15, 2025, 21:06:59 (UTC+05.5)	22.99	<input type="checkbox"/> sha256:c71389ac...	-

Summary Images

View push commands

aws | Search [Alt+S] Account ID: 5159-6652-1120
United States (N. Virginia) Dhavanisha

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry

Amazon ECR > Private registry > Repositories > Images

Amazon Elastic Container Registry < app_2

Private registry

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

Public registry

Repositories

Summary Images

Images (1) Info

Filter active images

C Delete Copy URI Details Scan View push commands

Image tags	Type	Created at	Image size	Image digest	Last pulled at
latest	Image	December 15, 2025, 21:09:37 (UTC+05.5)	22.99	sha256:1b35ba7...	-

Create task definition to run the Docker application from ECR on ECS Fargate without using EC2 servers for both Application.

The screenshot shows the AWS Amazon Elastic Container Service (ECS) console. The top navigation bar includes the AWS logo, a search bar, and account information (Account ID: 5159-6652-1120, United States (N. Virginia), Dhavanisha). Below the navigation bar, there are links for VPC, EC2, Aurora and RDS, IAM, S3, Route 53, and Elastic Container Registry. The left sidebar for the Amazon Elastic Container Service shows options like Express Mode (New), Clusters, Namespaces, Task definitions (selected), and Account settings. The main content area is titled "Create new task definition" and contains two sections: "Task definition configuration" and "Infrastructure requirements". In the "Task definition configuration" section, the "Task definition family" is set to "app_1". In the "Infrastructure requirements" section, the "Launch type" is selected as "AWS Fargate", which is described as "Serverless compute for containers".

aws | Search [Alt+S] Account ID: 5159-6652-1120
VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry United States (N. Virginia) Dhavanisha

Amazon Elastic Container Service > Create new task definition

Amazon Elastic Container Service

Express Mode New

Clusters

Namespaces

Task definitions

Account settings

Amazon ECR

Repositories

AWS Batch

Documentation

Create new task definition Info

Task definition configuration

Task definition family Info

Specify a unique task definition family name.

app_1

Up to 255 letters (uppercase and lowercase), numbers, hyphens, and underscores are allowed.

▼ Infrastructure requirements Info

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.

Managed Instances - new
Use if you have specific hardware constraints, such as GPU accelerators, CPU instruction sets, or network-optimized hardware

copy paste the ECR repository URI here

The screenshot shows the AWS Elastic Container Service (ECS) interface for creating a new task definition. On the left, there's a sidebar with links like 'Amazon Elastic Container Service', 'Task definitions', and 'Amazon ECR'. The main area is titled 'Create new task definition' and shows a 'Task execution role' dropdown set to 'ecsTaskExecutionRole'. Below it, under 'Container - 1', there's a 'Name' field with 'app_1' and an 'Essential container' dropdown set to 'Yes'. The 'Image URI' field contains '515966521120.dkr.ecr.us-east-1.amazonaws.com/app_1:latest'. A red box highlights this 'Image URI' field and the entire 'Container - 1' section below it.

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry

Amazon Elastic Container Service > Create new task definition

ecsTaskExecutionRole

Task execution role | Info

A task execution IAM role is used by the container agent to make AWS API requests on your behalf. If you don't already have a task execution IAM role created, we can create one for you.

ecsTaskExecutionRole

▶ Task placement - *optional*

▶ 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: app_1

Essential container: Yes

Up to 255 letters (uppercase and lowercase), numbers, hyphens, and underscores are allowed.

Image URI: 515966521120.dkr.ecr.us-east-1.amazonaws.com/app_1:latest

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

Private registry | Info

Browse ECR images

Essential container Remove

AWS | Search [Alt+S] Account ID: 5159-6652-1120 ▾ Dhavanisha

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry

Amazon Elastic Container Service > Create new task definition

Amazon Elastic Container Service

Express Mode [New](#)

Clusters

Namespaces

Task definitions

Account settings

Amazon ECR ↗

Repositories ↗

AWS Batch ↗

Documentation ↗

Discover products ↗

Subscriptions ↗

Tell us what you think

Create new task definition [Info](#)

Task definition configuration

Task definition family [Info](#)

Specify a unique task definition family name.

app_2

Up to 255 letters (uppercase and lowercase), numbers, hyphens, and underscores are allowed.

▼ Infrastructure requirements [Info](#)

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.

Managed Instances - new
Use if you have specific hardware constraints, such as GPU accelerators, CPU instruction sets, or network-optimized hardware (offloads scaling, patching, and instance management to AWS).

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](#)

Linux/X86_64

Network mode [Info](#)

awsvpc

Task size [Info](#)

AWS | Search [Alt+S] | Account ID: 5159-6652-1120 | United States (N. Virginia) | Dhavanisha

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry

Amazon Elastic Container Service > Create new task definition

A task IAM role allows containers in the task to make API requests to AWS services. You can create a task IAM role from the [IAM console](#).
ecsTaskExecutionRole

Task execution role | [Info](#)
A task execution IAM role is used by the container agent to make AWS API requests on your behalf. If you don't already have a task execution IAM role created, we can create one for you.
ecsTaskExecutionRole

▶ Task placement - *optional*

▶ Fault injection - *optional*

▼ Container - 1 [Info](#) Essential container Remove

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 app_2 **Essential container** Yes

Up to 255 letters (uppercase and lowercase), numbers, hyphens, and underscores are allowed.

Image URI 515966521120.dkr.ecr.us-east-1.amazonaws.com/app_2:latest [Browse ECR images](#)
Up to 255 letters (uppercase and lowercase), numbers, hyphens, underscores, colons, periods, forward slashes, and number signs are allowed.

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

Tell us what you think

Task definition has been created for both application

The screenshot shows the Amazon Elastic Container Service (ECS) Task definitions page. The top navigation bar includes links to VPC, EC2, Aurora and RDS, IAM, S3, Route 53, and Elastic Container Registry. The left sidebar for the Amazon Elastic Container Service lists Express Mode (New), Clusters, Namespaces, Task definitions (which is selected and highlighted in blue), and Account settings. Below the sidebar, there are links to Amazon ECR and Repositories. The main content area displays the Task definitions section with the following details:

Task definition	Status of last revision
app_1	Active
app_2	Active

At the top right of the main content area, there are buttons for Deploy, Create new revision, and Create new task definition. A filter status dropdown is set to Active. The last update timestamp is December 15, 2025, 21:19 (UTC+5:30). Navigation controls for pages 1 and 2 are also present.

Created a new ECS cluster named ecs_demo_cluster.

The screenshot shows the AWS Elastic Container Service (ECS) Create cluster wizard. The top navigation bar includes the AWS logo, a search bar, and account information (Account ID: 5159-6652-1120, United States (N. Virginia), Dhavanisha). The main menu bar has links for VPC, EC2, Aurora and RDS, IAM, S3, Route 53, Elastic Container Registry, Elastic Container Service, and CodeCommit. The left sidebar for the Amazon Elastic Container Service (Clusters, Namespaces, Task definitions, Account settings, Amazon ECR, Repositories, AWS Batch, Documentation, Discover products, Subscriptions) is visible. The main content area is titled "Create cluster" and describes an ECS cluster as grouping tasks and services. It shows the "Cluster configuration" section where the "Cluster name" is set to "ecs_demo_cluster". Below it is a "Service Connect defaults - optional" section. The "Infrastructure - advanced" section is expanded, showing options for obtaining compute capacity: "Fargate only" (selected), "Fargate and Managed Instances", and "Fargate and Self-managed instances".

Amazon Elastic Container Service

Clusters

Namespaces

Task definitions

Account settings

Amazon ECR

Repositories

AWS Batch

Documentation

Discover products

Subscriptions

Search [Alt+S]

United States (N. Virginia)

Account ID: 5159-6652-1120

Dhavanisha

Create cluster Info

An Amazon ECS cluster groups together tasks, and services, and allows for shared capacity and common configurations. All of your tasks, services, and capacity must belong to a cluster.

Cluster configuration

Cluster name

ecs_demo_cluster

Cluster name must be 1 to 255 characters. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

► Service Connect defaults - *optional*

▼ Infrastructure - *advanced* Info

Configure the manner of obtaining compute resources that will be used to host your application.

Select a method of obtaining compute capacity

Your cluster is automatically configured for AWS Fargate (serverless), but you may choose to add Amazon EC2 instances (servers).

Fargate only
Serverless - you don't think about creating or managing servers.
Great for most common workloads.

Fargate and Managed Instances
Managed instances - Amazon ECS will manage patching and scaling on your behalf while giving you configurability about the types of instances. Great for more advanced workloads.

Fargate and Self-managed instances
Self-managed instances - you must ensure the instances are patched and scaled properly, and you have full control over the instances.

AWS | Search [Alt+S] Account ID: 5159-6652-1120 ▾ Dhavanisha

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry Elastic Container Service CodeCommit

Amazon Elastic Container Service Clusters

Amazon Elastic Container Service

Express Mode New

Clusters Namespaces Task definitions Account settings

Amazon ECR Repositories

AWS Batch

Documentation Discover products Subscriptions

Tell us what you think

Cluster ecs_demo_cluster has been created successfully.

Clusters (1) Info Last updated December 18, 2025, 13:35 (UTC+5:30)

Create cluster

Cluster	Services	Tasks	Container instances	CloudWatch monitoring	Capacity provider strategy
ecs_demo_cluster	0	No tasks running	0 EC2	<input checked="" type="checkbox"/> Default	No default found

Configured security groups to allow required traffic between the Load Balancer, ECS tasks and services.

The screenshot shows the AWS VPC Security Groups page. At the top, there is a success message: "Security group (sg-0a0e819f1d6bdf6f7 | ECS-SG) was created successfully". Below this, the "Security Groups (4)" table is displayed. The table has columns for Name, Security group ID, Security group name, VPC ID, and Description. The rows show the following data:

Name	Security group ID	Security group name	VPC ID	Description
-	sg-0bc06874ccce7ee2c	default	vpc-0d7fbaba068b994cd	default VPC secur
-	sg-0f190d62fc5e2adbf	ELB-SG	vpc-0d7fbaba068b994cd	sg for ecs cluster
-	sg-0d4d48f1b77fcfd9ed	launch-wizard-1	vpc-0d7fbaba068b994cd	launch-wizard-1 c
-	sg-0a0e819f1d6bdf6f7	ECS-SG	vpc-0d7fbaba068b994cd	Sg for ECS Netwo

Edit inbound rules [Info](#)

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules [Info](#)

Security group rule ID

sgr-00f6be65eb029ca0a

Type [Info](#)

All traffic

Protocol [Info](#)

All

Port range [Info](#)

All

Source [Info](#)

Custom



Description - optional [Info](#)

Piping all traffic to ELB to Sg

[Delete](#)

sgr-0e69cce96644e5800

Type [Info](#)

HTTP

Protocol [Info](#)

TCP

Port range [Info](#)

80

Source [Info](#)

Custom



Description - optional [Info](#)

sg-0f190d62fc5e2adbf

[Delete](#)

103.6.156.129/32

[Delete](#)

[Add rule](#)

[Cancel](#)

[Preview changes](#)

[Save rules](#)

AWS Search [Alt+S] Account ID: 5159-6652-1120 ▾ United States (N. Virginia) ▾ Dhavanisha

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry Elastic Container Service CodeCommit

EC2 > Security Groups > sg-0f190d62fc5e2adb - ELB-SG > Edit inbound rules

Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type <small>Info</small>	Protocol <small>Info</small>	Port range <small>Info</small>	Source <small>Info</small>	Description - optional <small>Info</small>
sgr-0d10fe4136a84661a	Custom TCP	TCP	8080	Custom	0.0.0.0/0 <small>X</small>
sgr-0dfb0bdf0db7f8901	Custom TCP	TCP	8080	Custom	::/0 <small>X</small>
sgr-0b6af2ca2e24e5232	HTTP	TCP	80	Custom	::/0 <small>X</small>
sgr-035a8930a22ce7dab	HTTP	TCP	80	Custom	0.0.0.0/0 <small>X</small>

Add rule

Cancel Preview changes Save rules

After configuring the security group, the ECS task was executed in the cluster to verify the application.

The screenshot shows the AWS Amazon Elastic Container Service (ECS) Cluster Overview page for the cluster 'ecs_demo_cluster'. The top navigation bar includes the AWS logo, search bar, and account information (Account ID: 5159-6652-1120, United States (N. Virginia)). The main content area displays the cluster overview with the following details:

- ARN:** arn:aws:ecs:us-east-1:515966521120:cluster/ecs_demo_cluster
- Status:** Active
- CloudWatch monitoring:** Default
- Registered container instances:** -

The 'Services' tab is selected, showing the following service status summary:

Draining	Active	Pending	Running
-	-	-	-

Below the summary, there is a table for managing services, with the 'Services' tab selected. The table header includes columns for Service name, ARN, Status, Schedu..., Launch..., Task de..., Deployments and tasks, and Last de... . The table body shows 'No services' and 'No services to display.'

The left sidebar contains links for Express Mode (New), Clusters, Namespaces, Task definitions, Account settings, Amazon ECR, Repositories, AWS Batch, Documentation, Discover products, Subscriptions, and a 'Tell us what you think' feedback link.

Use this public IP to test the both application.

The screenshot shows the AWS Elastic Container Service (ECS) Configuration page for a specific task. The task is part of the 'ecs_demo_cluster' and has the ID 'a7bc6c7ba3974d42a6e32752209fe87e'. The 'Configuration' tab is selected, displaying various details about the task's setup.

Task overview

ARN	Last status	Desired status	Started/Created at
arn:aws:ecs:us-east-1:5159665211:task/ecs_demo_cluster/a7bc6c7ba3974d42a6e32752209fe87e	Running	Running	December 15, 2025, 21:23 (UTC+5:30)
			December 15, 2025, 21:23 (UTC+5:30)

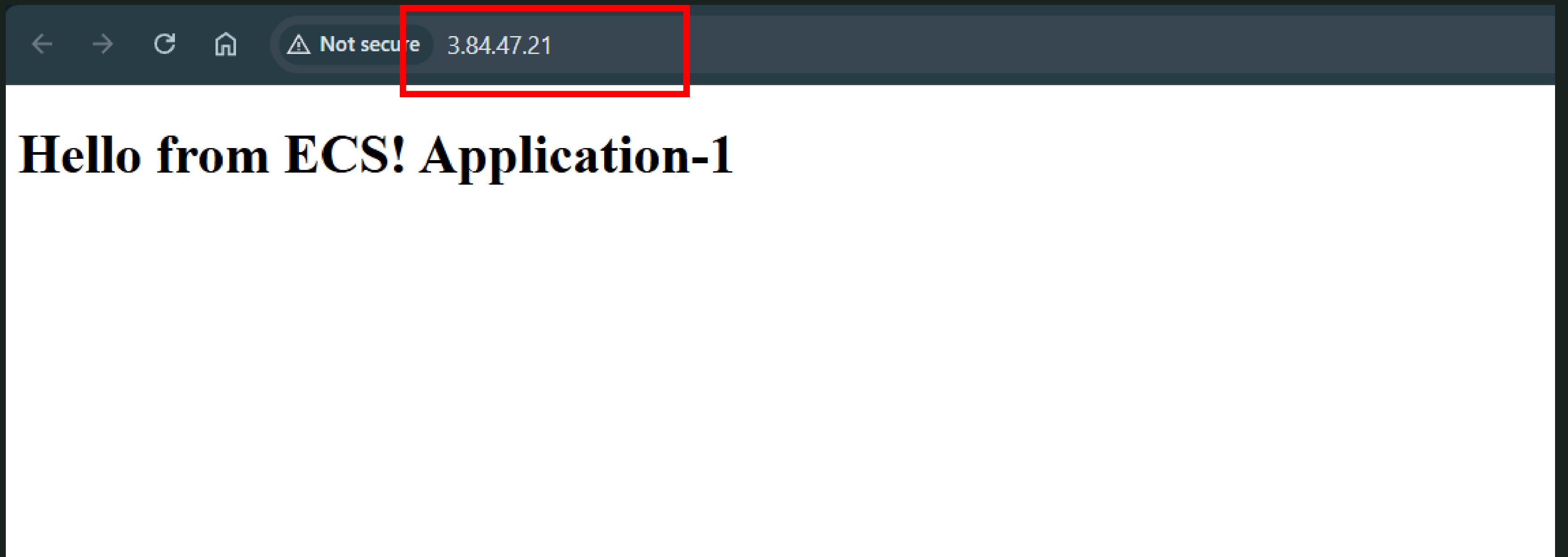
Fargate ephemeral storage

Encryption	Size (GiB)
Info Default AWS Fargate encryption	20

Configuration

Operating system/Architecture Linux/X86_64	Capacity provider -	ENI ID eni-07135d1e72afb8ce7
CPU Memory 1 vCPU 3 GB	Launch type Fargate	Network mode awsvpc
Platform version	Task definition: revision	Subnet
Public IP 3.84.47.21 open address		
Private IP 10.0.6.49		
MAC address		

Verified Application 1 running successfully in the browser.



us-east-1.console.aws.amazon.com/ecs/v2/clusters/ecs_demo_cluster/tasks/293c9cde740f4abebd92bb112bf94c56/configuration?selectedContainer=app_2

Account ID: 5159-6652-1120 | Dhavanisha

aws | Search [Alt+S]

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry

Amazon Elastic Container Service > Clusters > ecs_demo_cluster > Tasks > 293c9cde740f4abebd92bb112bf94c56 > Configuration

Amazon Elastic Container Service

Express Mode [New](#)

Clusters

- Namespaces
- Task definitions
- Account settings

Amazon ECR ↗

Repositories ↗

AWS Batch ↗

Documentation ↗

Discover products ↗

Subscriptions ↗

Tell us what you think

Configuration

Task overview

ARN arn:aws:ecs:us-east-1:5159665211 20:task/ecs_demo_cluster/293c9cde74 0f4abebd92bb112bf94c56	Last status Running	Desired status Running	Started/Created at December 15, 2025, 21:32 (UTC+5:30)
			December 15, 2025, 21:32 (UTC+5:30)

Fargate ephemeral storage

Encryption Info Default AWS Fargate encryption	Size (GiB) 20
--	-------------------------

Configuration

Operating system/Architecture Linux/X86_64	Capacity provider -	ENI ID eni-0bb4ad4bd0343fcca
CPU Memory 1 vCPU 3 GB	Launch type Fargate	Network mode awsvpc
Platform version 1.4.0	Task definition: revision app_2:1	Subnet subnet-0074188ec817364e0
Task execution role	Task group	

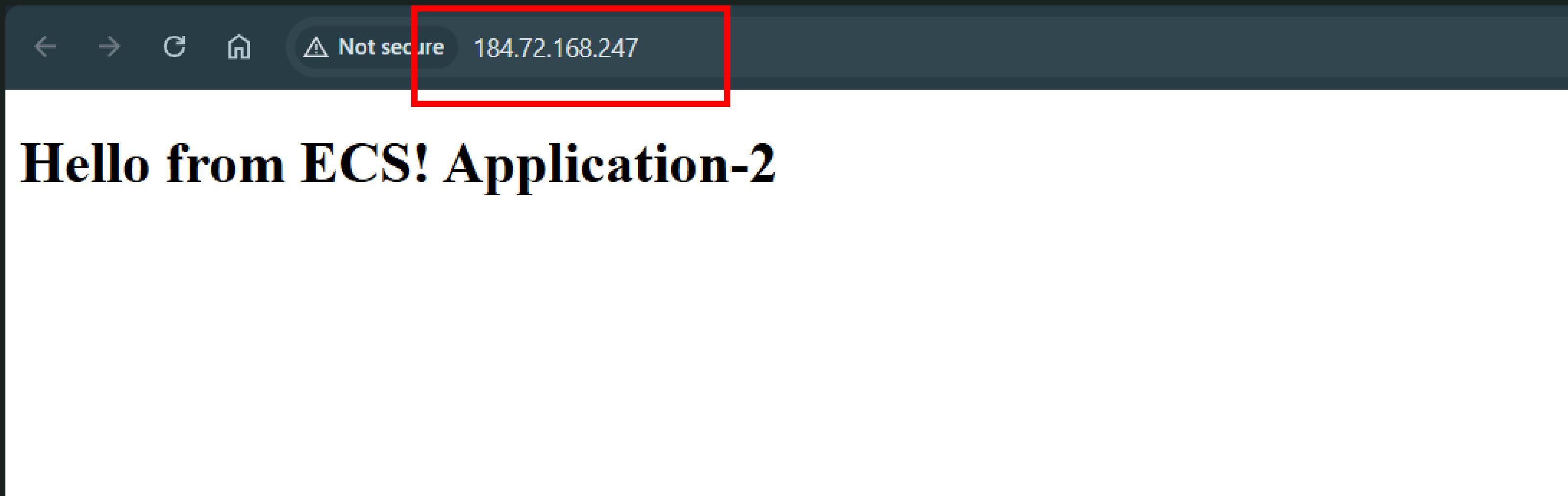
Container details for app_2

Public IP
[184.72.168.247](#) | [open address](#)

Private IP
[10.0.22.65](#)

MAC address
[0a:ff:ff:2c:fe:7b](#)

Verified Application 2 running successfully in the browser.



Create a target group for the load balancer.

Screenshot of the AWS EC2 Target Groups creation wizard.

The navigation bar shows: Account ID: 5159-6652-1120, United States (N. Virginia), Dhavanisha.

The breadcrumb trail is: EC2 > Target groups > Create target group.

The left sidebar shows the steps:

- Step 1: **Create target group** (selected)
- Step 2 - recommended: Register targets
- Step 3: Review and create

Create target group

A target group can be made up of one or more targets. Your load balancer routes requests to the targets in a target group and performs health checks on the targets.

Settings - immutable

Choose a target type and the load balancer and listener will route traffic to your target. These settings can't be modified after target group creation.

Target type

Indicate what resource type you want to target. Only the selected resource type can be registered to this target group.

Instances
Supports load balancing to instances in a VPC. Integrate with Auto Scaling Groups or ECS services for automatic management.
Suitable for: ALB, NLB, GWLB

IP addresses
Supports load balancing to VPC and on-premises resources. Facilitates routing to IP addresses and network interfaces on the same instance. Supports IPv6 targets.
Suitable for: ALB, NLB, GWLB

Lambda function
Supports load balancing to a single Lambda function. ALB required as traffic source.
Suitable for: ALB

Application Load Balancer
Allows use of static IP addresses and PrivateLink with an Application Load Balancer. NLB required as traffic source.
Suitable for: NLB

Target group name
Name must be unique per Region per AWS account.

Accepts: a-z, A-Z, 0-9, and hyphen (-). Can't begin or end with hyphen. 1-32 total characters; Count: 11/32

Protocol
Protocol for communication between the load balancer and targets.

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

aws | Search [Alt+S] | Account ID: 5159-6652-1120 ▾
VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry | United States (N. Virginia) ▾ Dhavanisha

EC2 > Target groups > Create target group

Step 1
Create target group
Step 2 - recommended
Register targets
Step 3
Review and create

Review and create

Review your target group configuration before creating

Step 1: Target group details

[Edit](#)

Target group details

Name ECS-TG-NODE	Target type IP	Protocol : Port HTTP: 80	Protocol version HTTP1
VPC vpc-0d7fbaba068b994cd ↗	IP address type IPv4		

Health check details

Health check protocol HTTP	Health check path /	Health check port traffic-port	Interval 30 seconds
Timeout 5 seconds	Healthy threshold 5	Unhealthy threshold 2	Success codes 200

Step 2: Register targets

[Edit](#)

Targets (0)

AWS | Search [Alt+S] Account ID: 5159-6652-1120 ▾ Dhavanisha

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry

EC2 Target groups

Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Capacity Manager New

Images

Elastic Block Store
Volumes
Snapshots
Lifecycle Manager

Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces

Load Balancing
Load Balancers
Target Groups
Trust Stores

Auto Scaling
Auto Scaling Groups

Target groups (1/1) [Info](#) | [What's new?](#)

Filter target groups

Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
ECS-TG-NODE	arn:aws:elasticloadbalancing:us-east-1:515966521120:targetgroup/ECS-TG-NODE/893e6cffa130f9ed	80	HTTP	IP	None associated	vpc-0d7fbaba068b994cd

Target group: ECS-TG-NODE

Details Targets Monitoring Health checks Attributes Tags

Details

arn:aws:elasticloadbalancing:us-east-1:515966521120:targetgroup/ECS-TG-NODE/893e6cffa130f9ed

Target type IP	Protocol : Port HTTP: 80	Protocol version HTTP1	VPC vpc-0d7fbaba068b994cd
IP address type IPv4	Load balancer None associated		

Create an internet-facing load balancer.

Screenshot of the AWS Management Console showing the process to create an Application Load Balancer (ALB). The user is in the 'Create Application Load Balancer' wizard.

The top navigation bar shows the AWS logo, search bar, account ID (5159-6652-1120), and region (United States (N. Virginia)). The breadcrumb trail indicates the current step: EC2 > Load balancers > Create Application Load Balancer.

Create Application Load Balancer Info

The Application Load Balancer distributes incoming HTTP and HTTPS traffic across multiple targets such as Amazon EC2 instances, microservices, and containers, based on request attributes. When the load balancer receives a connection request, it evaluates the listener rules in priority order to determine which rule to apply, and if applicable, it selects a target from the target group for the rule action.

▶ How Application Load Balancers work

Basic configuration

Load balancer name
Name must be unique within your AWS account and can't be changed after the load balancer is created.
 A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

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

Internet-facing
• 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
Includes only IPv4 addresses.

Dualstack
Includes IPv4 and IPv6 addresses.

Dualstack without public IPv4

Select the security group created for ELB

The screenshot shows the 'Create Application Load Balancer' wizard in the AWS Management Console. The top navigation bar includes the AWS logo, search bar, and account information (Account ID: 5159-6652-1120, United States (N. Virginia)). The main navigation bar has links for VPC, EC2, Aurora and RDS, IAM, S3, Route 53, and Elastic Container Registry. The current page is 'EC2 > Load balancers > Create Application Load Balancer'. The 'Availability Zones and subnets' section shows two selected zones: 'us-east-1a (use1-az2)' and 'us-east-1b (use1-az4)'. Each zone has a subnet listed: 'subnet-0e32fd3d4fc888352' (IPv4 subnet CIDR: 10.0.0.0/20) and 'subnet-0074188ec817364e0' (IPv4 subnet CIDR: 10.0.16.0/20). The 'Security groups' section at the bottom lists 'ELB-SG' (sg-0f190d62fc5e2adbf, VPC: vpc-0d7fbaba068b994cd), which is highlighted with a red box.

IP pools | [Info](#)
You can optionally choose to configure an IPAM pool as the preferred source for your load balancers IP addresses. Create or view [Pools](#) in the [Amazon VPC IP Address Manager console](#).

Use IPAM pool for public IPv4 addresses
The IPAM pool you choose will be the preferred source of public IPv4 addresses. If the pool is depleted IPv4 addresses will be assigned by AWS.

Availability Zones and subnets | [Info](#)
Select at least two Availability Zones and a subnet for each zone. A load balancer node will be placed in each selected zone and will automatically scale in response to traffic. The load balancer routes traffic to targets in the selected Availability Zones only.

us-east-1a (use1-az2)
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-0e32fd3d4fc888352 ecs-cicd-vpc-subnet-public1-us-east-1a
IPv4 subnet CIDR: 10.0.0.0/20

us-east-1b (use1-az4)
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-0074188ec817364e0 ecs-cicd-vpc-subnet-public2-us-east-1b
IPv4 subnet CIDR: 10.0.16.0/20

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

ELB-SG
sg-0f190d62fc5e2adbf VPC: vpc-0d7fbaba068b994cd

Select the Target group.

Screenshot of the AWS CloudFormation console showing the creation of a new Application Load Balancer (ALB). The page displays the configuration for a listener named 'HTTP:80'.

Listener Configuration:

- Protocol:** HTTP
- Port:** 80
- Default action:** Forward to target groups (selected)
- Routing action:** Redirect to URL (radio button), Return fixed response (radio button)

Forward to target group: Choose a target group and specify routing weight or [create target group](#).

Target group	Weight	Percent
ECS-TG-NODE Target type: IP, IPv4 Target stickiness: Off	1	100%
0-999		

Add target group: You can add up to 4 more target groups.

Target group stickiness: Enables the load balancer to bind a user's session to a specific target group. To use stickiness the client must support cookies. If you want to bind a user's session to a specific target, turn on the Target Group attribute Stickiness.

Turn on target group stickiness

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

Created an internet-facing Application Load Balancer named ECS-DEMO, which is active and deployed across two availability zones to distribute incoming traffic to the application.

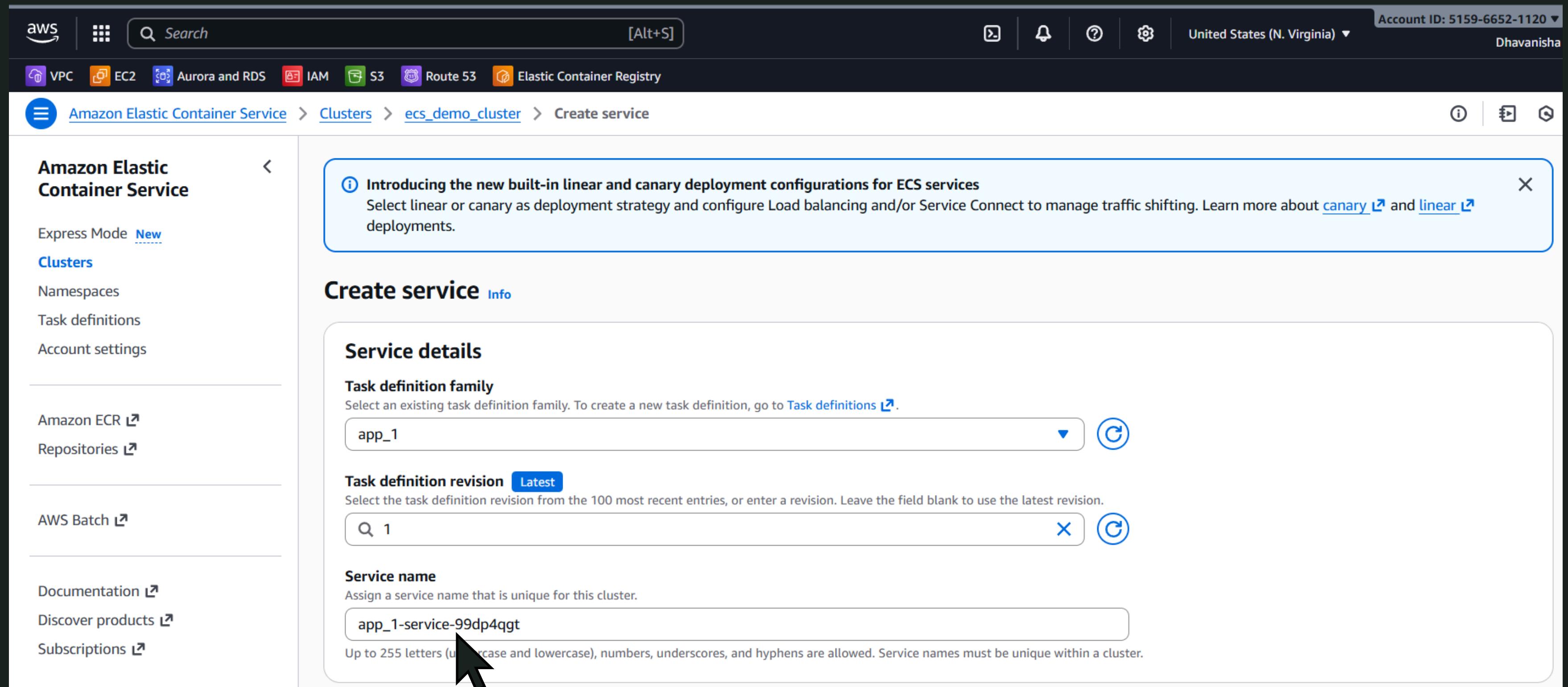
The screenshot shows the AWS Management Console interface for the EC2 service, specifically the Load balancers section. The top navigation bar includes the AWS logo, search bar, and account information (Account ID: 5159-6652-1120, United States (N. Virginia)). The left sidebar contains navigation links for various AWS services like VPC, EC2, Aurora and RDS, IAM, S3, Route 53, and Elastic Container Registry. The main content area displays the 'Load balancers (1/1)' section with the following details:

Name	Status	Type	Scheme	IP address type	VPC ID	Availability Zones
ECS-DEMO	Active	application	Internet-facing	IPv4	vpc-0d7fbaba068b994cd	2 Availability Zones

Below this, the 'Load balancer: ECS-DEMO' configuration is shown with the following settings:

Load balancer type	Status	VPC	Load balancer IP address type
Application	Active	vpc-0d7fbaba068b994cd	IPv4
Scheme	Hosted zone	Availability Zones	Date created
Internet-facing	Z35SXDOTRQ7X7K	subnet-0074188ec817364e0 us-east-1b (use1-az4) subnet-0e32fd3d4fc888352 us-east-1a (use1-az2)	December 14, 2025, 20:27 (UTC+05:30)
Load balancer ARN	DNS name		
arn:aws:elasticloadbalancing:us-east-1:515966521120:loadbalancer/app/ECS-DEM	ECS-DEMO-106380724.us-east-1.elb.amazonaws.com (A Record)		

Create a service and Use a unique service name.



The screenshot shows the AWS Elastic Container Service (ECS) interface. The top navigation bar includes the AWS logo, a search bar, and account information (Account ID: 5159-6652-1120, Dhavanisha). Below the navigation is a toolbar with links to VPC, EC2, Aurora and RDS, IAM, S3, Route 53, and Elastic Container Registry. The main navigation path is "Amazon Elastic Container Service > Clusters > ecs_demo_cluster > Create service".

A sidebar on the left lists "Amazon Elastic Container Service" sections: Express Mode (New), Clusters (selected), Namespaces, Task definitions, Account settings, Amazon ECR (with link to Repositories), AWS Batch, Documentation, Discover products, and Subscriptions.

The main content area is titled "Create service" with an "Info" link. It features a callout box with the message: "Introducing the new built-in linear and canary deployment configurations for ECS services. Select linear or canary as deployment strategy and configure Load balancing and/or Service Connect to manage traffic shifting. Learn more about [canary](#) and [linear](#)".

The "Service details" section contains fields for "Task definition family" (set to "app_1"), "Task definition revision" (set to "Latest"), and "Service name" (set to "app_1-service-99dp4qgt"). A note below the service name field states: "Up to 255 letters (use lowercase and uppercase), numbers, underscores, and hyphens are allowed. Service names must be unique within a cluster." A large black arrow points to the "Service name" input field.

AWS | Search [Alt+S] | United States (N. Virginia) ▾ Account ID: 5159-6652-1120 ▾ Dhavanisha

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry

Amazon Elastic Container Service > Clusters > ecs_demo_cluster > Create service

Amazon Elastic Container Service

Express Mode [New](#)

Clusters

Namespaces

Task definitions

Account settings

Amazon ECR ↗

Repositories ↗

AWS Batch ↗

Documentation ↗

Discover products ↗

Subscriptions ↗

Tell us what you think

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 ▾

▶ **Troubleshooting configuration - recommended**

Deployment configuration

Scheduling strategy | [Info](#)

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.

2

Select the security group created for ECS.

Screenshot of the AWS Amazon Elastic Container Service (ECS) "Create service" wizard.

The left sidebar shows the navigation path: **Amazon Elastic Container Service** > **Clusters** > **ecs_demo_cluster** > **Create service**.

The main configuration screen includes the following fields:

- VPC**: A dropdown menu showing "vpc-0d7fbaba068b994cd" and "ecs-cicd-vpc-vpc". A blue button labeled "Create a new VPC" is adjacent to the dropdown.
- Subnets**: A dropdown menu labeled "Choose subnets" with two selected subnets:
 - subnet-0074188ec817364e0 (with "X" icon)
 - subnet-0e32fd3d4fc888352 (with "X" icon)A blue button labeled "Clear current selection" is next to the dropdown.
- Security group**: A section with two radio button options:
 - Use an existing security group
 - Create a new security group
- Security group name**: A dropdown menu labeled "Choose security groups" with one item selected:
 - sg-0a0e819f1d6bdf6f7 (with "X" icon)
- Public IP**: A section with a toggle switch labeled "Turned on".

A large black arrow points to the "sg-0a0e819f1d6bdf6f7" entry in the security group dropdown.

Attached the existing Application Load Balancer using port 80

Screenshot of the AWS Amazon Elastic Container Service (ECS) console showing the creation of a new service for the 'ecs_demo_cluster'.

The left sidebar shows the navigation path: [Amazon Elastic Container Service](#) > [Clusters](#) > [ecs_demo_cluster](#) > [Create service](#).

The main configuration screen includes the following details:

- Amazon Elastic Container Service** (selected)
- Express Mode** (New)
- Clusters**
- Namespaces**
- Task definitions**
- Account settings**
- Amazon ECR**
- Repositories**
- AWS Batch**
- Documentation**
- Discover products**
- Subscriptions**
- Tell us what you think**

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

Use load balancing

VPC
The VPC for your load balancing resources must be the same as the VPC for your service with awsvpc.

Load balancer type | [Info](#)
Specify the load balancer type to distribute incoming traffic across the tasks running in your service.

Application Load Balancer
An Application Load Balancer makes routing decisions at the application layer (HTTP/HTTPS), supports path-based routing, and can route requests to one or more ports.

Network Load Balancer
A Network Load Balancer makes routing decisions at the transport layer (TCP/UDP).

Container
The container and port to load balance the incoming traffic to

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

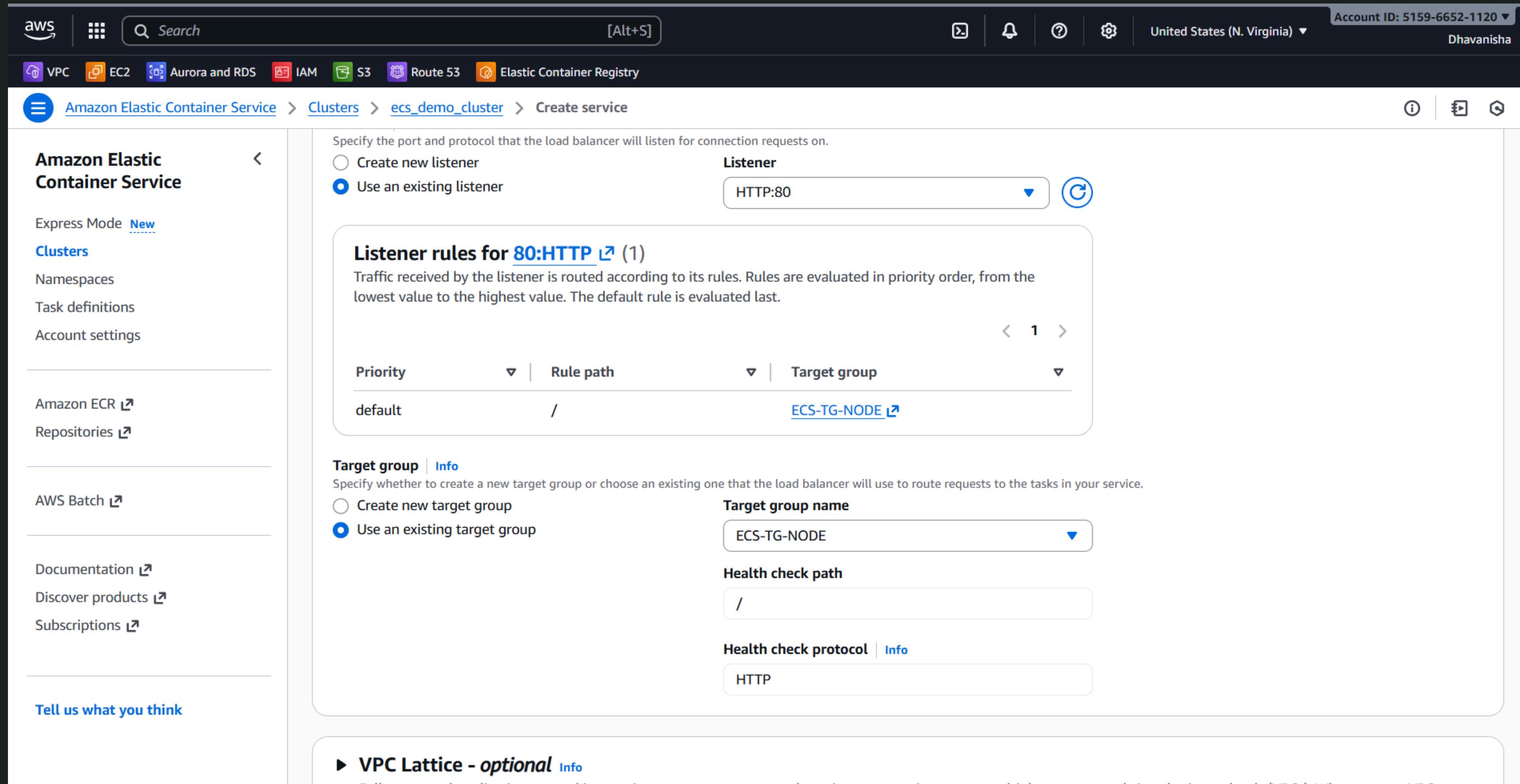
Create a new load balancer

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](#).

internet-facing 

Listener | [Info](#)



Services has been Created for application 1.

AWS | Search [Alt+S] | Account ID: 5159-6652-1120 | United States (N. Virginia) | Dhavanisha

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry

Amazon Elastic Container Service > Clusters > ecs_demo_cluster > Services

Amazon Elastic Container Service <

Express Mode New

Clusters

Namespaces

Task definitions

Account settings

Amazon ECR ↗

Repositories ↗

AWS Batch ↗

Documentation ↗

Discover products ↗

Subscriptions ↗

Tell us what you think

app_1-service-99dp4qgt deployment is in progress. It takes a few minutes. View in CloudFormation X

Last updated December 15, 2025, 21:58 (UTC+5:30)

Actions Create with Express Mode

ecs_demo_cluster

Last updated December 15, 2025, 21:58 (UTC+5:30)

CloudWatch monitoring Default

Registered container instances -

Cluster overview

ARN	Status	CloudWatch monitoring	Registered container instances
arn:aws:ecs:us-east-1:5159665211:cluster/ecs_demo_cluster	Active	Default	-

Services

Draining	Active	Pending	Running
-	-	-	-

Tasks

Active	Pending	Running
-	-	-

Services Tasks Infrastructure Metrics Scheduled tasks Configuration Event history Tags

Services (1) Info

Last updated December 15, 2025, 21:58 (UTC+5:30)

Manage tags Update Delete service Create

Filter launch type Any launch type ▾

Filter scheduling strategy Any scheduling strategy ▾

Filter resource management type Any resource management type ▾

< 1 > ⚙

Service name	ARN	Status	Schedu...	Lau...	Task de...	Deployments and tasks
app_1-service-99dp4qgt	arn:aws:ecs:us-e...	Active	REPLICA	FARG...	app_1:1	0/2 Tasks

AWS | Search [Alt+S] Account ID: 5159-6652-1120 ▾ Dhavanisha

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry

Amazon Elastic Container Service > Clusters > ecs_demo_cluster > Services > app_1-service-99dp4qgt > Health

Amazon Elastic Container Service < View service X

Express Mode New

Clusters Namespaces Task definitions Account settings

Amazon ECR ↗ Repositories ↗ AWS Batch ↗ Documentation ↗ Discover products ↗ Subscriptions ↗ Tell us what you think

app_1-service-99dp4qgt Info Last updated December 15, 2025, 22:00 (UTC+5:30) G Delete service Update service ▾

Introducing the new built-in linear and canary deployment configurations for ECS services Select linear or canary as deployment strategy and configure Load balancing and/or Service Connect to manage traffic shifting. Learn more about [canary ↗](#) and [linear ↗](#) deployments.

Service overview Info

Status Active	Tasks (2 Desired) 0 Pending 2 Running	Task definition: revision app_1:1	Deployment status Success
---------------	---	-----------------------------------	---------------------------

Health and metrics Tasks Logs Deployments Events Configuration and networking Service auto scaling Event his >

Status Info

Service name app_1-service-99dp4qgt	Service ARN arn:aws:ecs:us-east-1:515966521120:service/ecs_demo_cluster/app_1-service-99dp4qgt	Deployments current state 2 Completed tasks	Created at December 15, 2025, 21:58 (UTC+5:30)
-------------------------------------	--	---	--

Health check grace period 0 seconds

Load balancer health

The target group shows all registered targets as healthy and ready to receive traffic on port 80.

The screenshot shows the AWS EC2 Target groups page. On the left, there is a navigation sidebar with sections like Dedicated Hosts, Capacity Reservations, Capacity Manager (New), Images, Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces), Load Balancing (Load Balancers, Target Groups), and Auto Scaling (Auto Scaling Groups). The main content area displays a table of target groups with one entry: "ECS-TG-NODE". The table columns are Name, ARN, Port, Protocol, Target type, Load balancer, and VPC ID. Below this, a detailed view for the "ECS-TG-NODE" target group is shown, listing two registered targets: "10.0.5.221" and "10.0.19.133", both of which are marked as "Healthy".

Target groups (1/1) [Info](#) | [What's new?](#)

Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
ECS-TG-NODE	arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/ECS-TG-NODE/5678901234567890	80	HTTP	IP	ECS-DEMO	vpc-0d7f

Target group: ECS-TG-NODE

Registered targets (2) [Info](#)

Anomaly mitigation: Not applicable

IP address	Port	Zone	Health status	Health status details	Admini...	Overri...	Anomaly detectio...
10.0.5.221	80	us-east-1a (...	Healthy	-	No override.	No overri...	Normal
10.0.19.133	80	us-east-1b (...	Healthy	-	No override.	No overri...	Normal

Copied the load balancer DNS name to verify the application in the browser.

The screenshot shows the AWS EC2 Load Balancers console. The left sidebar has sections for Capacity Manager, Images, Elastic Block Store (selected), Network & Security, Load Balancing (selected), Auto Scaling, and Settings. The main area displays 'Load balancers (1/1)' with one item: 'ECS-DEMO' (Active, application, Internet-facing, IPv4, VPC ID: vpc-0d7fbaba068b994cd, 2 Availability Zones). Below it, the 'Load balancer: ECS-DEMO' details show Application (Active), Scheme (Internet-facing), Hosted zone (Z35SXDOTRQ7X7K), Availability Zones (subnet-0074188ec817364e0 us-east-1b (use1-az4), subnet-0e32fd3d4fc888352 us-east-1a), and Date created (December 14, 2025, 20:27 (UTC+05:30)). A red box highlights the 'DNS name copied' message next to the ARN: arn:aws:elasticloadbalancing:us-east-1:515966521120:loadbalancer/app/ECS-DEMO/O/ea83c65ab82b6c14. Another red box highlights the DNS name: ECS-DEMO-106380724.us-east-1.elb.amazonaws.com (A Record).

aws | Search [Alt+S] Account ID: 5159-6652-1120
VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry United States (N. Virginia) Dhavanisha

EC2 > Load balancers

Capacity reservations

Capacity Manager [New](#)

Images

Elastic Block Store

- Volumes
- Snapshots
- Lifecycle Manager

Network & Security

- Security Groups
- Elastic IPs
- Placement Groups
- Key Pairs
- Network Interfaces

Load Balancing

Load Balancers

- Target Groups
- Trust Stores

Auto Scaling

- Auto Scaling Groups

Settings

Load balancers (1/1) [what's new?](#)

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Filter load balancers

Name State Type Scheme IP address type VPC ID Availability Zones

Name	State	Type	Scheme	IP address type	VPC ID	Availability Zones
ECS-DEMO	Active	application	Internet-facing	IPv4	vpc-0d7fbaba068b994cd	2 Availability Zones

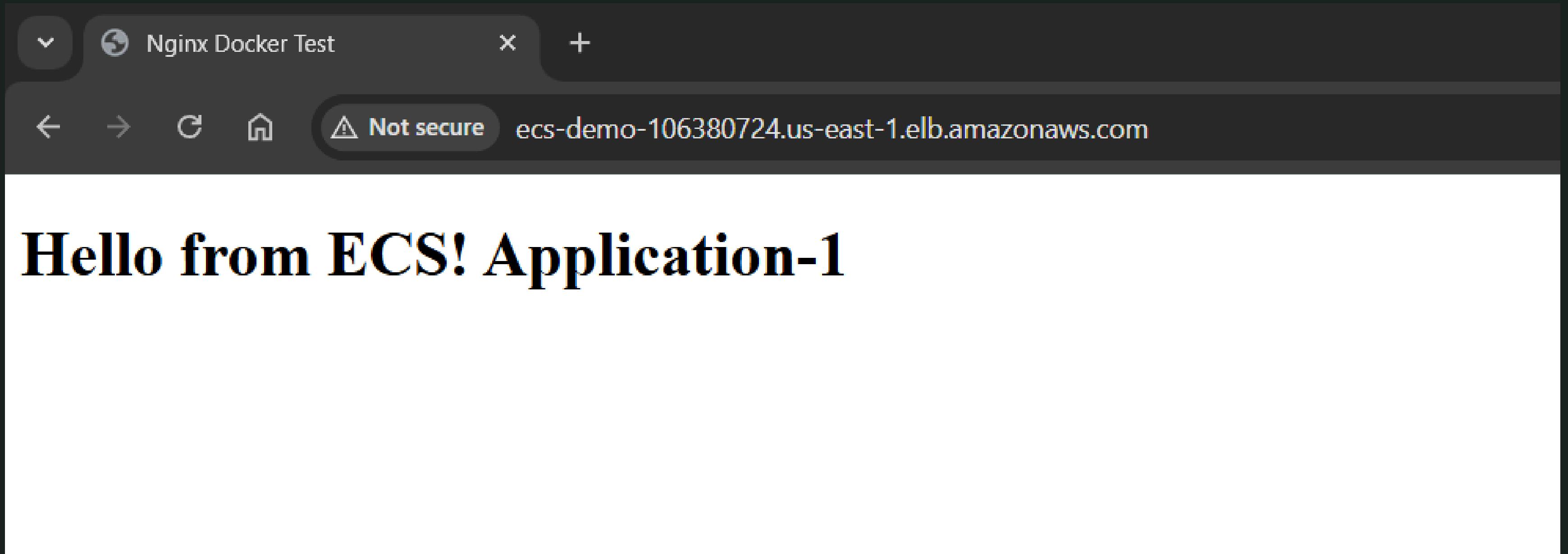
Load balancer: ECS-DEMO

Application	Active	vpc-0d7fbaba068b994cd	IPv4
Scheme	Hosted zone	Availability Zones	Date created
Internet-facing	Z35SXDOTRQ7X7K	subnet-0074188ec817364e0 us-east-1b (use1-az4)	December 14, 2025, 20:27 (UTC+05:30)
		subnet-0e32fd3d4fc888352 us-east-1a	

Load balancer ARN: arn:aws:elasticloadbalancing:us-east-1:515966521120:loadbalancer/app/ECS-DEMO/O/ea83c65ab82b6c14

DNS name copied: ECS-DEMO-106380724.us-east-1.elb.amazonaws.com (A Record)

Verified Application-1 is running successfully using the load balancer DNS name; the same process is followed to validate Application-2.



AWS | Search [Alt+S] Account ID: 5159-6652-1120 ▾ Dhavanisha

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry

Amazon Elastic Container Service > Clusters > ecs_demo_cluster > Create service

Amazon Elastic Container Service

Express Mode [New](#)

Clusters

Namespaces

Task definitions

Account settings

Amazon ECR ↗

Repositories ↗

AWS Batch ↗

Documentation ↗

Discover products ↗

Subscriptions ↗

Tell us what you think

Introducing the new built-in linear and canary deployment configurations for ECS services
Select linear or canary as deployment strategy and configure Load balancing and/or Service Connect to manage traffic shifting. Learn more about [canary ↗](#) and [linear ↗](#) deployments.

Create service [Info](#)

Service details

Task definition family
Select an existing task definition family. To create a new task definition, go to [Task definitions ↗](#).

app_2

Task definition revision [Latest](#)
Select the task definition revision from the 100 most recent entries, or enter a revision. Leave the field blank to use the latest revision.

1

Service name
Assign a service name that is unique for this cluster.

app_2-service-37tot4ew

Up to 255 letters (uppercase and lowercase), numbers, underscores, and hyphens are allowed. Service names must be unique within a cluster.

Environment [AWS Fargate](#)

Existing cluster

ecs_demo_cluster

AWS | Search [Alt+S] | Account ID: 5159-6652-1120 | United States (N. Virginia) | Dhavanisha

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry

Amazon Elastic Container Service > Clusters > ecs_demo_cluster > Create service

Amazon Elastic Container Service

Express Mode New

Clusters

Namespaces

Task definitions

Account settings

Amazon ECR ↗

Repositories ↗

AWS Batch ↗

Documentation ↗

Discover products ↗

Subscriptions ↗

Tell us what you think

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 ▾

▶ Troubleshooting configuration - recommended

Deployment configuration

Scheduling strategy | Info

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.

2 ▾

AWS | Search [Alt+S] Account ID: 5159-6652-1120 ▾ Dhavanisha

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry

Amazon Elastic Container Service > Clusters > ecs_demo_cluster > Create service

Amazon Elastic Container Service <

Express Mode [New](#)

Clusters

Namespaces

Task definitions

Account settings

Amazon ECR ↗

Repositories ↗

AWS Batch ↗

Documentation ↗

Discover products ↗

Subscriptions ↗

Tell us what you think

Networking

VPC | [Info](#)

Select a VPC to use for your Amazon ECS resources.

vpc-0d7fbaba068b994cd
ecs-cicd-vpc-vpc

[Create a new VPC ↗](#)

Subnets

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

[Choose subnets](#) [Clear current selection](#)

subnet-0074188ec817364e0 X
ecs-cicd-vpc-subnet-public2-us-east-1b
us-east-1b 10.0.16.0/20

subnet-0e32fd3d4fc888352 X
ecs-cicd-vpc-subnet-public1-us-east-1a
us-east-1a 10.0.0.0/20

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.

[Choose security groups](#)

sg-0a0e819f1d6bdf6f7 X
ECS-SG

Public IP | [Info](#)

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

Turned on

Created a new target group for Application-2 using port 8080 with HTTP protocol.

The screenshot shows the AWS Amazon Elastic Container Service (ECS) Create service wizard. The top navigation bar includes the AWS logo, search bar, account ID (5159-6652-1120), and region (United States (N. Virginia)). The left sidebar lists navigation options: Express Mode (New), Clusters (selected), Namespaces, Task definitions, Account settings, Amazon ECR, Repositories, AWS Batch, Documentation, Discover products, and Subscriptions. The main content area is titled "Configure load balancing using Amazon Elastic Load Balancing to distribute traffic evenly across the healthy tasks in your service." A checked checkbox "Use load balancing" is present. Under "VPC", the VPC dropdown is set to "vpc-0d7fbaba068b994cd". The "Load balancer type" section contains two options: "Application Load Balancer" (selected) and "Network Load Balancer". The "Container" section shows "app_2 80:80" as the host port:container port mapping. The "Application Load Balancer" section has two options: "Create a new load balancer" and "Use an existing load balancer" (selected). The "Load balancer" section shows an existing load balancer "ECS-DEMO" with the IP "ECS-DEMO-106380724.us-east-1.elb.amazonaws.com" and a dropdown for "internet-facing".

Amazon Elastic Container Service

Express Mode [New](#)

Clusters

Namespaces

Task definitions

Account settings

Amazon ECR ↗

Repositories ↗

AWS Batch ↗

Documentation ↗

Discover products ↗

Subscriptions ↗

Tell us what you think

Search [Alt+S]

Account ID: 5159-6652-1120 ▾
Dhavanisha

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry

Amazon Elastic Container Service > Clusters > ecs_demo_cluster > Create service

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

Use load balancing

VPC
The VPC for your load balancing resources must be the same as the VPC for your service with aws.vpc.

vpc-0d7fbaba068b994cd

Load balancer type [Info](#)
Specify the load balancer type to distribute incoming traffic across the tasks running in your service.

Application Load Balancer
An Application Load Balancer makes routing decisions at the application layer (HTTP/HTTPS), supports path-based routing, and can route requests to one or more ports.

Network Load Balancer
A Network Load Balancer makes routing decisions at the transport layer (TCP/UDP).

Container
The container and port to load balance the incoming traffic to

app_2 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 ↗](#).

ECS-DEMO
ECS-DEMO-106380724.us-east-1.elb.amazonaws.com

internet-facing ▾

AWS | Search [Alt+S] Account ID: 5159-6652-1120 | Dhavanisha

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry

Amazon Elastic Container Service > Clusters > ecs_demo_cluster > Create service

Amazon Elastic Container Service

Express Mode New

Clusters

Namespaces

Task definitions

Account settings

Amazon ECR ↗

Repositories ↗

AWS Batch ↗

Documentation ↗

Discover products ↗

Subscriptions ↗

Tell us what you think

Listener | Info

Specify the port and protocol that the load balancer will listen for connection requests on.

Create new listener
 Use an existing listener

Port
8080

Protocol
HTTP

Target group | Info

Specify whether to create a new target group or choose an existing one that the load balancer will use to route requests to the tasks in your service.

Create new target group
 Use an existing target group

Target group name
app2_tg_8080

Protocol
HTTP

Port
80

Deregistration delay
The amount of time to wait before the state of a deregistering target changes from draining to unused.

300

seconds

Health check protocol | Info

HTTP

Health check path | Info

Two services are created in the cluster.

The screenshot shows the AWS Elastic Container Service (ECS) console interface. The top navigation bar includes the AWS logo, a search bar, and account information (Account ID: 5159-6652-1120, Dhavanisha). The main navigation menu at the top has links for VPC, EC2, Aurora and RDS, IAM, S3, Route 53, and Elastic Container Registry. Below the menu, the breadcrumb navigation shows: Amazon Elastic Container Service > Clusters > ecs_demo_cluster > Services.

The main content area displays the "Cluster overview" for the "ecs_demo_cluster". Key details shown include:

- ARN:** arn:aws:ecs:us-east-1:515966521120:cluster/ecs_demo_cluster
- Status:** Active
- CloudWatch monitoring:** Default
- Registered container instances:** -

The "Services" section shows the following counts:

Draining	Active	Pending	Running
-	2	-	4

The "Tasks" section shows the following counts:

Active	Pending	Running
2	-	4

Below the cluster overview, there is a navigation bar with tabs: Services (selected), Tasks, Infrastructure, Metrics, Scheduled tasks, Configuration, Event history, and Tags.

The "Services" table lists two services:

Service name	ARN	Status	Schedu...	Lau...	Task de...	Deployments and tasks
app_1-service-99dp4qgt	arn:aws:ecs:us-e...	Active	REPLICA	FARG...	app_1:1	2/2 Tasks
app_2-service-5pw04oru	arn:aws:ecs:us-e...	Active	REPLICA	FARG...	app_2:1	2/2 Tasks

At the bottom left, there is a "Tell us what you think" feedback link.

Configured the Application Load Balancer with listeners on port 80 for Application-1 and port 8080 for Application-2.

The screenshot shows the AWS CloudWatch Metrics interface for an Application Load Balancer named "ECS-DEMO". The metrics listed are:

Metric	Value	Unit	Period
HTTP:80	100%	Percent	1 minute
HTTP:8080	100%	Percent	1 minute

Each metric entry includes a "Forward to target group" link and a "Target group stickiness: Off" note.

aws | Search [Alt+S] | Account ID: 5159-6652-1120 | Dhavanisha | United States (N. Virginia) | VPC | EC2 | Aurora and RDS | IAM | S3 | Route 53 | Elastic Container Registry

EC2 > Target groups

Dedicated Hosts
Capacity Reservations
Capacity Manager [New](#)

Images

Elastic Block Store
Volumes
Snapshots
Lifecycle Manager

Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces

Load Balancing
Load Balancers
Target Groups
Trust Stores

Auto Scaling
Auto Scaling Groups

Target groups (1/2) [Info](#) | [What's new?](#)

Filter target groups

Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
APPTWO-TG-8080	arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/APPTWO-TG-8080/1234567890123456	80	HTTP	IP	ECS-DEMO	vpc-0d7f
ECS-TG-NODE	arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/ECS-TG-NODE/1234567890123456	80	HTTP	IP	ECS-DEMO	vpc-0d7f

Target group: APPTWO-TG-8080

Registered targets (2) [Info](#)

Anomaly mitigation: Not applicable

[Deregister](#) [Register targets](#)

Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

Filter targets

IP address	Port	Zone	Health status	Health status details	Administrative over...	Overri...	Anomaly det
10.0.24.181	80	us-east-1b (...)	Healthy	-	<input type="checkbox"/> No override	No overri...	Normal
10.0.14.226	80	us-east-1a (...)	Healthy	-	<input type="checkbox"/> No override	No overri...	Normal

Account ID: 5159-6652-1120 ▾
Dhavanisha

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry

Amazon Elastic Container Service > Clusters > ecs_demo_cluster > Services > app_2-service-5pw04oru > Health

Select linear or canary as deployment strategy and configure Load balancing and/or Service Connect to manage traffic shifting. Learn more about [canary](#) and [linear](#)

Amazon Elastic Container Service

Express Mode [New](#)

Clusters

Namespaces

Task definitions

Account settings

Amazon ECR [↗](#)

Repositories [↗](#)

AWS Batch [↗](#)

Documentation [↗](#)

Discover products [↗](#)

Subscriptions [↗](#)

Tell us what you think

Service overview [Info](#)

Status	Tasks (2 Desired)	Task definition: revision	Deployment status
Active	0 Pending 2 Running	app_2:1	Success

[Health and metrics](#) [Tasks](#) [Logs](#) [Deployments](#) [Events](#) [Configuration and networking](#) [Service auto scaling](#) [Event his](#)

Status [Info](#)

Service name	Service ARN	Deployments current state	Created at
app_2-service-5pw04oru	arn:aws:ecs:us-east-1:51596652:1120:service/ecs_demo_cluster/app_2-service-5pw04oru	2 Completed tasks	December 15, 2025, 22:30 (UTC+5:30)

Health check grace period
0 seconds

▼ Load balancer health

Load balancer	Load balancer type	Container name:port	Listeners	Target group	Target health
ECS-DEMO	Application Load Bal...	app_2:80	HTTP:8080	APPTWO-TG-8080 ...	2 Healthy 0 Unhealthy

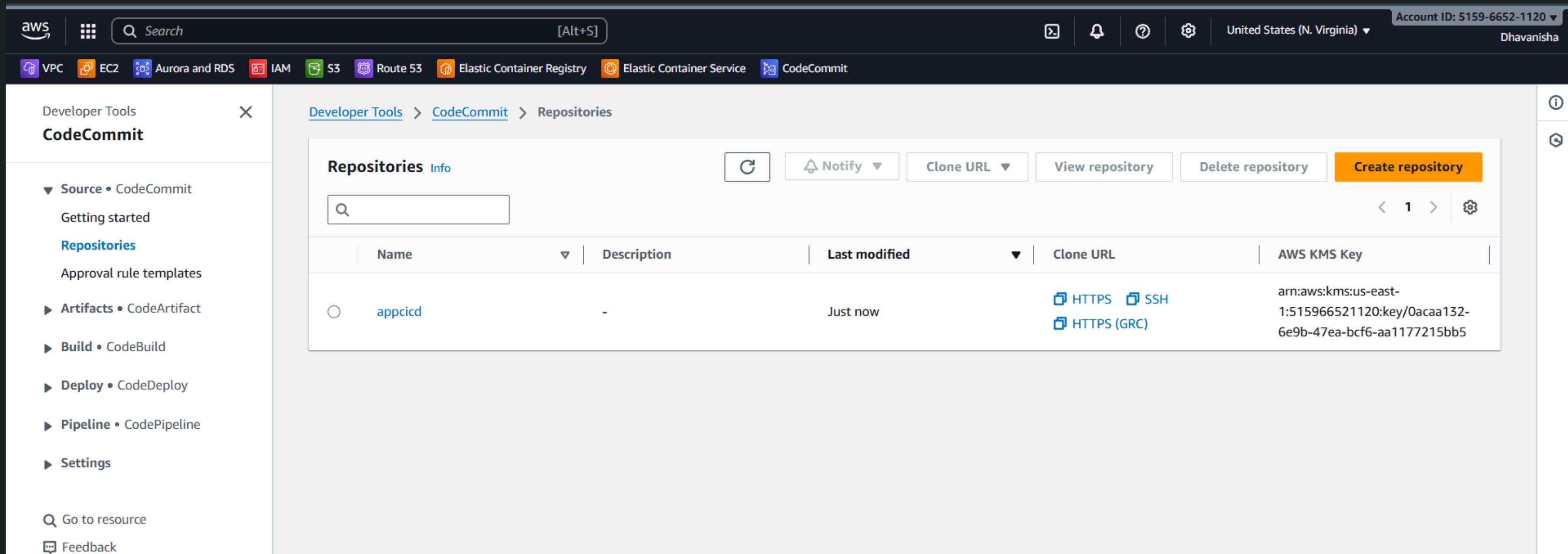
Verified Application-2 is running successfully using the load balancer DNS name.



The CI/CD automation process was initiated by setting up a source code repository in AWS CodeCommit.

The screenshot shows the AWS CodeCommit service within the AWS Management Console. The top navigation bar includes the AWS logo, a search bar, and account information (Account ID: 5159-6652-1120, Dhavanisha). Below the navigation is a horizontal menu with links to VPC, EC2, Aurora and RDS, IAM, S3, Route 53, Elastic Container Registry, Elastic Container Service, and CodeCommit. The CodeCommit link is highlighted. On the left, a sidebar titled 'Developer Tools' contains sections for 'Source • CodeCommit' (Getting started, Repositories, Approval rule templates), 'Artifacts • CodeArtifact', 'Build • CodeBuild', 'Deploy • CodeDeploy', 'Pipeline • CodePipeline', 'Settings', 'Go to resource', and 'Feedback'. The main content area shows the 'CodeCommit > Repositories' page. The title bar has buttons for 'Info', 'Notify', 'Clone URL', 'View repository', 'Delete repository', and 'Create repository'. A search bar is present above the table. The table headers are 'Name', 'Description', 'Last modified', 'Clone URL', and 'AWS KMS Key'. A message at the bottom center states 'No results' and 'There are no results to display.'

Repository was created in CodeCommit.



The screenshot shows the AWS CodeCommit console interface. The top navigation bar includes the AWS logo, search bar, account ID (5159-6652-1120), and region (United States (N. Virginia)). The left sidebar, titled "Developer Tools" and "CodeCommit", contains links for Source (CodeCommit), Getting started, Repositories (Approval rule templates), Artifacts (CodeArtifact), Build (CodeBuild), Deploy (CodeDeploy), Pipeline (CodePipeline), Settings, Go to resource, and Feedback. The main content area shows the "Repositories" page under "CodeCommit > Repositories". The table lists one repository:

Name	Description	Last modified	Clone URL	AWS KMS Key
appcid	-	Just now	HTTPS SSH HTTPS (GRC)	arn:aws:kms:us-east-1:515966521120:key/0acaa132-6e9b-47ea-bcf6-aa1177215bb5

Configured AWS CodeCommit access using HTTPS (GRC) to securely clone and manage the repository.

The screenshot shows the AWS CodeCommit documentation for using HTTPS (GRC). The top navigation bar includes the AWS logo, search bar, and account information (Account ID: 5159-6652-1120, United States (N. Virginia)). The left sidebar has a 'Developer Tools' header and a 'CodeCommit' section with links like 'Getting started', 'Repositories', 'Code' (selected), 'Pull requests', 'Commits', 'Branches', 'Git tags', 'Settings', 'Approval rule templates', 'Artifacts', 'Build', 'Deploy', 'Pipeline', and 'Settings'. The main content area has tabs for 'HTTPS', 'SSH', and 'HTTPS (GRC)' (selected). A warning message states: '⚠ You are signed in using a root account. You cannot configure SSH connections for a root account, and HTTPS connections for a root account are not recommended. Consider signing in as an IAM user and then setting up your connection.' Below this, it says: 'HTTPS (GRC) is the protocol to use with git-remote-codecommit (GRC). This utility provides a simple method for pushing and pulling code from CodeCommit repositories by extending Git. It is the recommended method for supporting connections made with federated access, identity providers, and temporary credentials.' The 'Step 1: Prerequisites' section lists requirements: 1. Install Python (View the Python website), 2. Install and configure a Git client (View Git downloads page), 3. Have permissions to the CodeCommit repository (Learn how to create and configure an IAM user for accessing AWS CodeCommit, Learn how to configure connections for users with rotating credentials, Learn how to add team members to an AWS CodeStar Project), and 4. Install and configure the AWS CLI with a profile (IAM User, Assuming a role, Web identity federation). The 'Step 2: Install git-remote-codecommit' section instructs to run 'pip install git-remote-codecommit'. The 'Step 3: Clone the repository' section provides the command 'git clone codecommit:us-east-1://appcid' with a 'Copy' button. The 'Additional details' section links to 'View documentation'.

HTTPS | SSH | **HTTPS (GRC)**

⚠ You are signed in using a root account. You cannot configure SSH connections for a root account, and HTTPS connections for a root account are not recommended. Consider signing in as an IAM user and then setting up your connection.

HTTPS (GRC) is the protocol to use with git-remote-codecommit (GRC). This utility provides a simple method for pushing and pulling code from CodeCommit repositories by extending Git. It is the recommended method for supporting connections made with federated access, identity providers, and temporary credentials.

Step 1: Prerequisites

We recommend that you use the latest versions of Git and other prerequisite software. You can find information about minimum required versions of prerequisite software. [Prerequisite software](#)

1. You must install Python. To download and install the latest version of Python. [View the Python website](#)
2. You must install and configure a Git client. [View Git downloads page](#)
3. You must have permissions to the CodeCommit repository.
 - o [Learn how to create and configure an IAM user for accessing AWS CodeCommit](#)
 - o [Learn how to configure connections for users with rotating credentials](#)
 - o [Learn how to add team members to an AWS CodeStar Project](#)
4. You must install and configure the AWS CLI with a profile.
 - o [IAM User](#)
 - o [Assuming a role](#)
 - o [Web identity federation](#)

Step 2: Install git-remote-codecommit

At a terminal or command line, run the following command to install git-remote-codecommit:

```
pip install git-remote-codecommit
```

Step 3: Clone the repository

Clone your repository to your local computer and start working on code. Run the following command:

```
git clone codecommit:us-east-1://appcid
```

Copy

Additional details

You can find more detailed instructions in the documentation. [View documentation](#)

Configured CodeCommit support by installing git-remote-codecommit.

```
[root@ip-10-0-23-4 ~]# pip3 install git-remote-codecommit
Collecting git-remote-codecommit
  Downloading git-remote-codecommit-1.17.tar.gz (10 kB)
    Preparing metadata (setup.py) ... done
Collecting botocore>=1.17.0
  Downloading botocore-1.42.9-py3-none-any.whl (14.5 MB)
    |██████████| 14.5 MB 26.6 MB/s
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/lib/python3.9/site-packages (from botocore>=1.17.0->git-remote-codecommit) (0.10.0)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/lib/python3.9/site-packages (from botocore>=1.17.0->git-remote-codecommit) (1.25.10)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3.9/site-packages (from botocore>=1.17.0->git-remote-codecommit) (2.8.1)
Requirement already satisfied: six>=1.5 in /usr/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->botocore>=1.17.0->git-remote-codecommit) (1.15.0)
Using legacy 'setup.py install' for git-remote-codecommit, since package 'wheel' is not installed.
Installing collected packages: botocore, git-remote-codecommit
  Running setup.py install for git-remote-codecommit ... done
Successfully installed botocore-1.42.9 git-remote-codecommit-1.17
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

Cloned the CodeCommit repository successfully into the cicddemo folder.

```
[root@ip-10-0-23-4 ~]# mkdir cicddemo
[root@ip-10-0-23-4 ~]# cd cicddemo/
[root@ip-10-0-23-4 cicddemo]# git clone codecommit::us-east-1://appcicd
Cloning into 'appcicd'...
warning: You appear to have cloned an empty repository.
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
hint:
hint: Disable this message with "git config set advice.defaultBranchName false"
[root@ip-10-0-23-4 cicddemo]# ls
appcicd
[root@ip-10-0-23-4 cicddemo]#
```

```
[root@ip-10-0-23-4 ~]# cp -r app_1/* cicddemo/appcicd/
[root@ip-10-0-23-4 ~]# cd cicddemo/appcicd/
[root@ip-10-0-23-4 appcicd]# ls
Dockerfile index.html
[root@ip-10-0-23-4 appcicd]# git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Dockerfile
    index.html

nothing added to commit but untracked files present (use "git add" to track)
[root@ip-10-0-23-4 appcicd]# git add .
[root@ip-10-0-23-4 appcicd]# git commit -m "My first code"
[master (root-commit) a22fc2e] My first code
  Committer: root <root@ip-10-0-23-4.ec2.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. Run the
following command and follow the instructions in your editor to edit
your configuration file:

  git config --global --edit

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

  git commit --amend --reset-author

  2 files changed, 20 insertions(+)
  create mode 100644 Dockerfile
  create mode 100644 index.html
[root@ip-10-0-23-4 appcicd]# █
```

Copied application files to the repository, added them to Git, and committed the first code successfully.

```
[root@ip-10-0-23-4 appcicd]# git status
On branch master

No commits yet

Untracked files:
(use "git add <file>..." to include in what will be committed)
  Dockerfile
  index.html

nothing added to commit but untracked files present (use "git add" to track)
[root@ip-10-0-23-4 appcicd]# git add .
[root@ip-10-0-23-4 appcicd]# git commit -m "My first code"
[master (root-commit) a22fc2e] My first code
Committer: root <root@ip-10-0-23-4.ec2.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. Run the
following command and follow the instructions in your editor to edit
your configuration file:
git config --global --edit

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

git commit --amend --reset-author

2 files changed, 20 insertions(+)
create mode 100644 Dockerfile
create mode 100644 index.html
[root@ip-10-0-23-4 appcicd]# git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 549 bytes | 549.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Validating objects: 100%
To codecommit::us-east-1://appcicd
 * [new branch]      master -> master
[root@ip-10-0-23-4 appcicd]#
```

Pushed the committed code from the local repository to AWS CodeCommit successfully.

Verified that the application files were successfully uploaded to the CodeCommit repository.

The screenshot shows the AWS CodeCommit interface. The top navigation bar includes the AWS logo, a search bar, and account information (Account ID: 5159-6652-1120, United States (N. Virginia)). The left sidebar has a 'CodeCommit' section with options like Source, Code, Pull requests, Commits, Branches, Git tags, Settings, Approval rule templates, Artifacts, and Build. The main content area shows the repository 'appcid'. The breadcrumb navigation shows Developer Tools > CodeCommit > Repositories > appcid. The repository name 'appcid' is displayed prominently. On the right, there are buttons for Notify (with a dropdown), Reference (set to master), Create pull request, and Clone URL. Below these are sections for 'Info' and 'Code'. The 'Info' section lists the file 'Dockerfile'. The 'Code' section lists the files 'Dockerfile' and 'index.html'. There is also a 'Add file' button.

Created a new CodePipeline named my-app-cicd and configured a service role for pipeline execution.

The screenshot shows the AWS CodePipeline 'Create new pipeline' wizard in progress. The top navigation bar includes the AWS logo, a search bar, and account information (Account ID: 5159-6652-1120, United States (N. Virginia), Dhavanisha). The left sidebar lists steps from Step 1 to Step 7, with 'Step 2' currently selected. The main content area is titled 'Choose pipeline settings' and shows the following configuration:

- Pipeline name:** my-app-cicd
- Execution mode:** Queued (selected)
- Service role:** New service role (selected) - Create a service role in your account. The role name is AWSCodePipelineServiceRole-us-east-1-my-app-cicd. A checkbox allows AWS CodePipeline to create the service role.
- Advanced settings:** A link at the bottom of the form.

Configured the source stage by connecting CodePipeline to the CodeCommit repository on the master branch.

The screenshot shows the AWS CodePipeline 'Create new pipeline' wizard at Step 2: Choose pipeline settings. The left sidebar lists steps from Step 2 to Step 7. The main panel is titled 'Source' and 'Source provider' is set to 'AWS CodeCommit'. The 'Repository name' is 'appcid' and the 'Branch name' is 'master'. A checked checkbox 'Create EventBridge rule to automatically detect source changes' has a note about following AWS documentation. Under 'Output artifact format', 'CodePipeline default' is selected, with a note about it being the default zip format. A second option, 'Full clone', is also listed. At the bottom, a checked checkbox 'Enable automatic retry on stage failure' is present. Navigation buttons 'Cancel', 'Previous', and 'Next' are at the bottom right.

Step 2

Choose pipeline settings

Step 3

Add source stage

Step 4 - optional

Add build stage

Step 5 - optional

Add test stage

Step 6

Add deploy stage

Step 7

Review

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.

appcid

Branch name

Choose a branch of the repository

master

Create EventBridge rule to automatically detect source changes
If disabled, follow AWS documentation to create an EventBridge rule for your source. [Learn more ↗](#)

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

Cancel Previous Next

In the build stage, clicked “Create project” to create a new AWS CodeBuild project for the pipeline.

The screenshot shows the AWS CodePipeline 'Create new pipeline' wizard at Step 4 of 7. The left sidebar lists steps from Step 1 to Step 7, with 'Add build stage' currently selected. The main content area is titled 'Add build stage - optional'. It includes sections for 'Build provider' (set to 'Other build providers' with 'AWS CodeBuild' selected), 'Project name' (with a search bar and 'Create project' button, both highlighted with a red box), 'Define buildspec override - optional' (unchecked), 'Environment variables - optional' (with a 'Add environment variable' button), 'Build type' (set to 'Single build'), and 'Region' (set to 'United States (N. Virginia)').

Step 1
Choose creation option

Step 2
Choose pipeline settings

Step 3
Add source stage

Step 4 - optional
Add build stage

Step 5 - optional
Add test stage

Step 6
Add deploy stage

Step 7
Review

Add build stage - optional Info

Step 4 of 7

Build - optional

Build provider
Choose the tool you want to use to run build commands and specify artifacts for your build action.

Commands Other build providers

AWS CodeBuild

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.

or [Create project](#)

Define buildspec override - optional
Buildspec file or definition that overrides the latest one defined in the build project, for this build only.

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.

Region

United States (N. Virginia)

Created a new AWS CodeBuild project named my-app-cicd-project with default build settings.

The screenshot shows the 'Create build project' wizard in the AWS Management Console. The top navigation bar includes the AWS logo, a search bar, and account information (Account ID: 5159-6652-1120, United States (N. Virginia), Dhavanisha). The breadcrumb trail indicates the path: Developer Tools > CodeBuild > Build projects > Create build project. The main section is titled 'Create build project' and contains the 'Project configuration' step. The 'Project name' field is set to 'my-app-cicd-project'. Below it, a note states: '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 _.' The 'Project type' section offers two options: 'Default project' (selected) and 'Runner project'. The 'Default project' option is described as 'Create a custom CodeBuild project.' The 'Runner project' option is described as 'Create a CodeBuild managed runner for workflows in GitHub Actions, GitHub Enterprise Actions, GitLab, or Buildkite.' The 'Additional configuration' section is collapsed, showing options for description, public build access, build badge, concurrent build limit, and tags. The 'Environment' section is expanded, showing the 'Provisioning model' (On-demand selected, described as 'Automatically provision build infrastructure in response to new builds.') and 'Environment image' (Managed image selected, described as 'Use an image managed by AWS CodeBuild').

Configured the CodeBuild environment to run builds in a Docker container using the standard Ubuntu image.

The screenshot shows the AWS CodeBuild 'Create build project' configuration interface. The 'Running mode' section is set to 'Container' (Running on Docker container). The 'Operating system' is 'Ubuntu'. The 'Runtime(s)' is 'Standard'. The 'Image' is 'aws/codebuild/standard:7.0'. The 'Image version' dropdown is set to 'Always use the latest image for this runtime version'. The 'Service role' section shows 'New service role' selected (Create a service role in your account) and 'Existing service role' (Choose an existing service role from your account) is also present. The 'Role name' is 'codebuild-my-app-cicd-project-service-role'. At the bottom, there is an 'Additional configuration' section with a note about certificates, VPC, compute type, environment variables, file system, and code source.

aws | Search [Alt+S] Account ID: 5159-6652-1120 | United States (N. Virginia) | Dhavanisha

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry

Developer Tools > CodeBuild > Build projects > Create build project

Running mode

Container
Running on Docker container

Instance
Running on EC2 instance directly

Operating system

Ubuntu

Runtime(s)

Standard

Image

aws/codebuild/standard:7.0

Image version

Always use the latest image for this runtime version

Use GPU-enhanced compute

Service role

New service role
Create a service role in your account

Existing service role
Choose an existing service role from your account

Role name

codebuild-my-app-cicd-project-service-role

Type your service role name

Additional configuration

This configuration does not include certificates, VPC, compute type, environment variables, file system, or code source.

Configured the buildspec to build the Docker image and push it to Amazon ECR using CodeBuild.

The screenshot shows the AWS CodeBuild 'Create build project' interface. At the top, there are navigation links: VPC, EC2, Aurora and RDS, IAM, S3, Route 53, and Elastic Container Registry. Below the navigation bar, the breadcrumb trail reads: Developer Tools > CodeBuild > Build projects > Create build project. On the left, there's a sidebar with a 'Additional configuration' section containing timeout, privileged, certificate, VPC, compute type, environment variables, file systems, auto-retry, and registry credential options. The main area is titled 'Buildspec' and contains a 'Build specifications' section. Two options are available: 'Insert build commands' (selected) and 'Use a buildspec file'. The 'Insert build commands' option is described as 'Store build commands as build project configuration'. The 'Use a buildspec file' option is described as 'Store build commands in a YAML-formatted buildspec file'. Below this, the 'Build commands' section displays a YAML buildspec file. The file defines phases: pre_build, build, and post_build. The pre_build phase logs into ECR and performs a docker login. The build phase starts the build and tags the Docker image. The post_build phase pushes the image to ECR and writes an imagedefinitions.json file. The buildspec file is as follows:

```
1 version: 0.2
2
3 phases:
4   pre_build:
5     commands:
6       - echo Logging in to Amazon ECR...
7       - aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 501170964283.dkr.ecr.us-east-1.amazonaws.com
8
9   build:
10    commands:
11      - echo Build started on `date`
12      - docker build -t mynode:latest .
13      - docker tag mynode:latest 501170964283.dkr.ecr.us-east-1.amazonaws.com/mynode:latest
14
15   post_build:
16    commands:
17      - echo Pushing Docker image to ECR...
18      - docker push 501170964283.dkr.ecr.us-east-1.amazonaws.com/mynode:latest
19      - echo Writing imagedefinitions.json...
20      - printf '[{"name":"mynode","imageUri":"%s"}]' 501170964283.dkr.ecr.us-east-1.amazonaws.com/mynode:latest > imagedefinitions.json
21
22 artifacts:
```

At the bottom left, there are status icons for errors and warnings, both of which are currently zero. At the bottom right, it says '25:1 YAML'.

Configured the deploy stage to deploy the build artifact to the ECS service in the ecs_demo_cluster.

The screenshot shows the AWS CodePipeline 'Create new pipeline' interface at Step 7: Add deploy stage. The left sidebar lists optional steps: Add build stage, Step 5 - optional, Add test stage, Step 6, and Step 7. The main panel is titled 'Add deploy stage' and contains the following fields:

- Region:** United States (N. Virginia)
- Input artifacts:** A dropdown menu showing 'BuildArtifact' (Defined by: Build) with a delete icon.
- Cluster name:** A search bar containing 'ecs_demo_cluster'.
- Service name:** A search bar containing 'app_1-service-veovgdp7'.
- Image definitions file - optional:** A text input field containing 'imagedefinitions.json'.
- Deployment timeout - optional:** An empty text input field.

Configured the deploy stage to deploy the application to Amazon ECS with automatic rollback enabled.

The screenshot shows the AWS CodePipeline 'Create new pipeline' wizard at Step 6: Add deploy stage. The configuration details are as follows:

- Deploy action provider**: Amazon ECS
- ClusterName**: ecs_demo_cluster
- ServiceName**: app_1-service-veovgdp7
- FileName**: imagedefinitions.json
- Configure automatic rollback on stage failure**: Enabled
- Enable automatic retry on stage failure**: Disabled

At the bottom, there are navigation buttons: 'Cancel', 'Previous', and 'Create pipeline'.

aws | Search [Alt+S] | Account ID: 5159-6652-1120 | Dhavanisha | United States (N. Virginia) ▾

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry

Developer Tools > CodePipeline > Pipelines > my-app-cicd

① Introducing the new pipeline experience
We've redesigned the pipeline view to streamline the monitoring and debugging experience. [Let us know what you think.](#) Or [go back to the old experience.](#)

② Success
Congratulations! The pipeline my-app-cicd has been created.

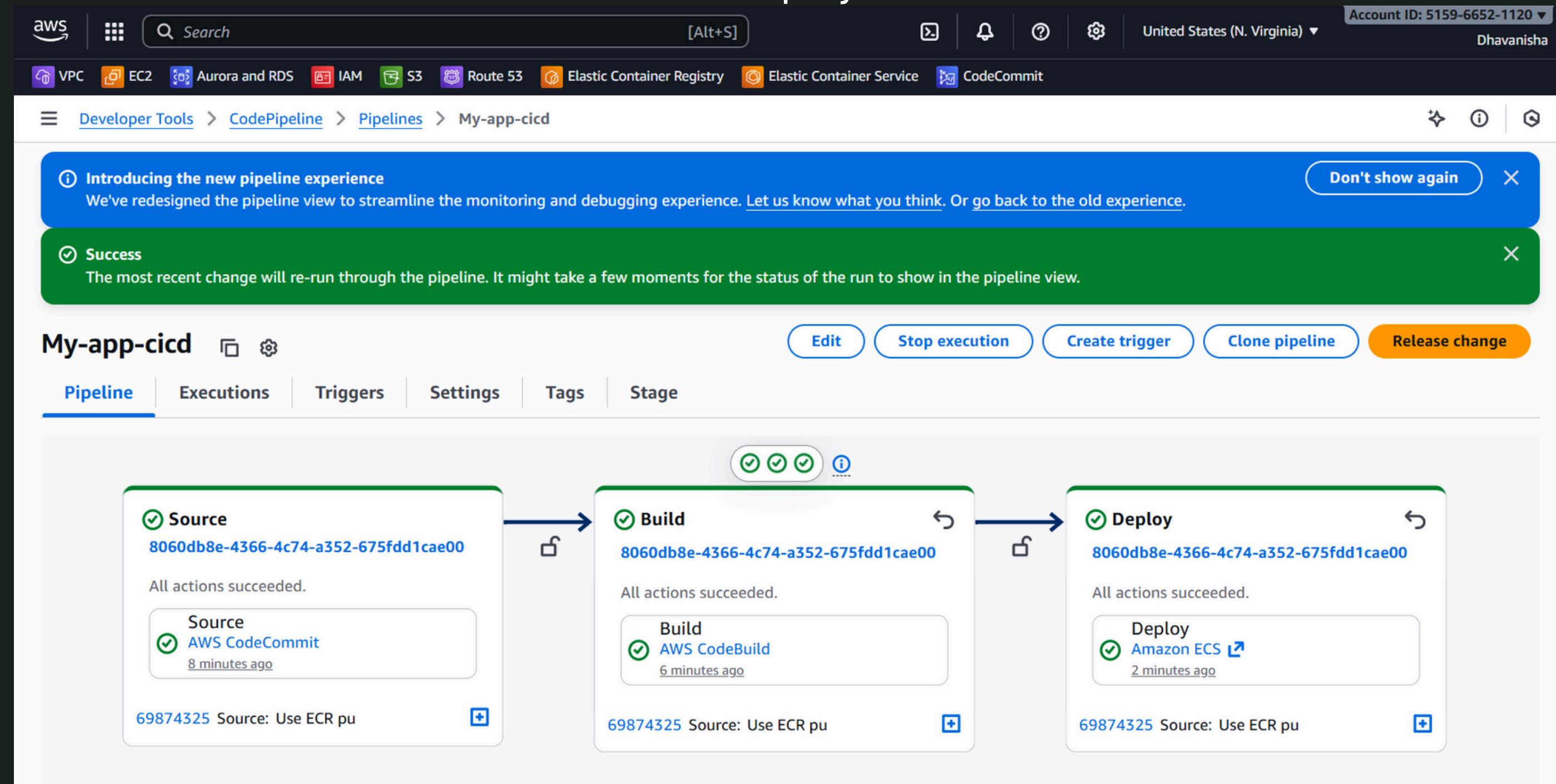
my-app-cicd Edit Stop execution Create trigger Clone pipeline Release change

Pipeline Executions Triggers Settings Tags Stage

The screenshot shows a pipeline named "my-app-cicd" with three stages: Source, Build, and Deploy. The Source stage is completed with a green checkmark, showing an AWS CodeCommit action that succeeded just now. The Build stage is in progress, with one task (AWS CodeBuild) also successful. The Deploy stage has not run yet. Each stage has a "Details" button and a "Logs" button.

```
graph LR; Source[Source: 62f6d874-285e-4225-8e51-656a905f8635] --> Build[Build: 62f6d874-285e-4225-8e51-656a905f8635]; Build --> Deploy[Deploy: Didn't Run]
```

The CI/CD pipeline was successfully executed using AWS CodePipeline, with CodeCommit as the source, CodeBuild for Docker image creation, and Amazon ECS for automated deployment.



aws | Search [Alt+S] | United States (N. Virginia)

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry Elastic Container Service CodeCommit

Developer Tools > CodePipeline > Pipelines

Pipelines Info

C View history Release change Delete pipe

Name	Latest execution status	Latest source revisions	Latest execution started	Most recent
My-app-cicd	Succeeded	Source – 5cb2465c : Edited index.html	12 minutes ago	

Developer Tools <

CodePipeline

- ▶ Source • CodeCommit
- ▶ Artifacts • CodeArtifact
- ▶ Build • CodeBuild
- ▶ Deploy • CodeDeploy
- ▼ Pipeline • CodePipeline
 - Getting started
 - Pipelines**
 - Account metrics

AWS | Search [Alt+S] Account ID: 5159-6652-1120 ▾ United States (N. Virginia) ▾ Dhavanisha

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry Elastic Container Service CodeCommit

Developer Tools > CodeBuild > Build projects > my-app-cicd-project-01 > my-app-cicd-project-01:6b74e707-bce0-4300-a879-fbfeb21917a1 ⓘ ⓘ

Developer Tools CodeBuild

Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Getting started
Build projects
Build project

Settings
Build history
Report groups
Report history
Compute fleets New
Account metrics
Related integrations
Jenkins
GitHub Actions New
GitLab runners New
Deploy • CodeDeploy
Pipeline • CodePipeline
Settings

my-app-cicd-project-01:6b74e707-bce0-4300-a879-fbfeb21917a1

Stop build Debug build Retry build

Build status

Status	Initiator	Build ARN	Resolved source version
Succeeded	codepipeline/My-app-cicd	arn:aws:codebuild:us-east-1:515966521120:build/my-app-cicd-project-01:6b74e707-bce0-4300-a879-fbfeb21917a1	6987432577010d84aa3cd2689ad17554a38584 27

Start time: Dec 16, 2025 10:08 PM (UTC+5:30) End time: Dec 16, 2025 10:09 PM (UTC+5:30) Build number: 6

Build logs Phase details Reports Environment variables Build details Resource utilization

Showing the last 156 lines of the build log. [View entire log](#) Tail logs

No previous logs

```
1 [Container] 2025/12/16 16:39:03.754245 Running on CodeBuild On-demand
2 [Container] 2025/12/16 16:39:03.754254 Waiting for agent ping
3 [Container] 2025/12/16 16:39:03.960448 Waiting for DOWNLOAD_SOURCE
4 [Container] 2025/12/16 16:39:05.201547 Phase is DOWNLOAD_SOURCE
5 [Container] 2025/12/16 16:39:05.202610 CODEBUILD_SRC_DIR=/codebuild/output/src1064323212/src
6 [Container] 2025/12/16 16:39:05.203092 YAML location is /codebuild/readonly/buildspec.yml
7 [Container] 2025/12/16 16:39:05.205412 Setting HTTP client timeout to higher timeout for S3 source
8 [Container] 2025/12/16 16:39:05.205554 Processing environment variables
9 [Container] 2025/12/16 16:39:05.394872 No runtime version selected in buildspec.
10 [Container] 2025/12/16 16:39:05.413538 Moving to directory /codebuild/output/src1064323212/src
11 [Container] 2025/12/16 16:39:05.413564 Cache is not defined in the buildspec
12 [Container] 2025/12/16 16:39:05.453291 Skip cache due to: no paths specified to be cached
13 [Container] 2025/12/16 16:39:05.453695 Registering with agent
14 [Container] 2025/12/16 16:39:05.486232 Phases found in YAML: 3
15 [Container] 2025/12/16 16:39:05.486252 PRE_BUILD: 2 commands
16 [Container] 2025/12/16 16:39:05.486257 BUILD: 3 commands
17 [Container] 2025/12/16 16:39:05.486261 POST_BUILD: 1 commands
```

Build logs confirm successful Docker image build and ECR authentication.

The screenshot shows the AWS CodeBuild console with the build logs for a project named "my-app-cicd-project-01". The logs confirm a successful Docker image build and ECR authentication.

Build Logs:

```
17 [Container] 2025/12/16 16:39:05.486261 POST_BUILD: 4 commands
18 [Container] 2025/12/16 16:39:05.486754 Phase complete: DOWNLOAD_SOURCE State: SUCCEEDED
19 [Container] 2025/12/16 16:39:05.486770 Phase context status code: Message:
20 [Container] 2025/12/16 16:39:05.592282 Entering phase INSTALL
21 [Container] 2025/12/16 16:39:05.630467 Phase complete: INSTALL State: SUCCEEDED
22 [Container] 2025/12/16 16:39:05.630489 Phase context status code: Message:
23 [Container] 2025/12/16 16:39:05.668021 Entering phase PRE_BUILD
24 [Container] 2025/12/16 16:39:05.701896 Running command echo Logging in to Amazon ECR...
25 Logging in to Amazon ECR...
26
27 [Container] 2025/12/16 16:39:05.711734 Running command aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 515966521120.dkr.ecr.us-east-1.amazonaws.com
28 WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
29 Configure a credential helper to remove this warning. See
30 https://docs.docker.com/engine/reference/commandline/login/#credential-stores
31
32 Login Succeeded
33
34 [Container] 2025/12/16 16:39:21.387144 Phase complete: PRE_BUILD State: SUCCEEDED
35 [Container] 2025/12/16 16:39:21.387162 Phase context status code: Message:
36 [Container] 2025/12/16 16:39:21.431269 Entering phase BUILD
37 [Container] 2025/12/16 16:39:21.432605 Running command echo Build started on `date`
38 Build started on Tue Dec 16 16:39:21 UTC 2025
39
40 [Container] 2025/12/16 16:39:21.440459 Running command docker build -t app_1:latest .
41 #0 building with "default" instance using docker driver
42
43 #1 [internal] load build definition from Dockerfile
44 #1 transferring dockerfile: 288B done
45 #1 DONE 0.0s
46
47 #2 [internal] load metadata for public.ecr.aws/nginx/nginx:alpine
48 #2 DONE 0.4s
49
50 #3 [internal] load .dockerignore
51 #3 transferring context: 2B done
52 #3 DONE 0.0s
53
54 #4 [internal] load build context
55 #4 transferring context: 180B done
56 #4 DONE 0.0s
57
58 #5 [1/2] FROM public.ecr.aws/nginx/nginx:alpine@sha256:9b0f84d48f92f2147217aec522219e9eda883a2836f1e30ab1915bd794f294ff
59 #5 resolve public.ecr.aws/nginx/nginx:alpine@sha256:9b0f84d48f92f2147217aec522219e9eda883a2836f1e30ab1915bd794f294ff 0.0s done
60
```

AWS | Search [Alt+S] Account ID: 5159-6652-1120 ▾ United States (N. Virginia) ▾ Dhavanisha

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry Elastic Container Service CodeCommit

Developer Tools > CodeBuild > Build projects > my-app-cicd-project-01 > my-app-cicd-project-01:6b74e707-bce0-4300-a879-fbfeb21917a1

Developer Tools < CodeBuild

- ▶ Source • CodeCommit
- ▶ Artifacts • CodeArtifact
- ▼ Build • CodeBuild
 - Getting started
 - Build projects
 - Build project**
 - Settings
 - Build history
 - Report groups
 - Report history
 - Compute fleets **New**
 - Account metrics
- ▼ Related integrations
 - Jenkins ↗
 - GitHub Actions ↗ **New**
 - GitLab runners ↗ **New**
- ▶ Deploy • CodeDeploy
- ▶ Pipeline • CodePipeline
- ▶ Settings

```
118 44227ec39841: Waiting
119 e4531687ef8e: Waiting
120 5aa68bbbc67e: Waiting
121 45178d7e9376: Layer already exists
122 7a44d12423b9: Layer already exists
123 dcf04fc2e09: Layer already exists
124 a2ec75146a7f: Layer already exists
125 44227ec39841: Layer already exists
126 e4531687ef8e: Layer already exists
127 9f71ae529e9d: Layer already exists
128 5aa68bbbc67e: Layer already exists
129 fe51c65502b1: Pushed
130 latest: digest: sha256:cd1c95b8bd2aff5a779227cf54c2881f184b7a03bd3ae299bb0471838376216 size: 2196
131
132 [Container] 2025/12/16 16:39:26.189988 Running command echo Writing imagedefinitions.json...
133 Writing imagedefinitions.json...
134
135 [Container] 2025/12/16 16:39:26.196753 Running command printf '[{"name":"app_1","imageUri":"%s"}]' 515966521120.dkr.ecr.us-east-1.amazonaws.com/app_1:latest > imagedefinitions.json
136
137 [Container] 2025/12/16 16:39:26.203512 Phase complete: POST_BUILD State: SUCCEEDED
138 [Container] 2025/12/16 16:39:26.203532 Phase context status code: Message:
139 [Container] 2025/12/16 16:39:26.299508 Expanding base directory path: .
140 [Container] 2025/12/16 16:39:26.301688 Assembling file list
141 [Container] 2025/12/16 16:39:26.301704 Expanding .
142 [Container] 2025/12/16 16:39:26.303891 Expanding file paths for base directory .
143 [Container] 2025/12/16 16:39:26.303904 Assembling file list
144 [Container] 2025/12/16 16:39:26.303908 Expanding imagedefinitions.json
145 [Container] 2025/12/16 16:39:26.305953 Found 1 file(s)
146 [Container] 2025/12/16 16:39:26.309588 Set report auto-discover timeout to 5 seconds
147 [Container] 2025/12/16 16:39:26.309638 Expanding base directory path: .
148 [Container] 2025/12/16 16:39:26.311641 Assembling file list
149 [Container] 2025/12/16 16:39:26.311655 Expanding .
150 [Container] 2025/12/16 16:39:26.313680 Expanding file paths for base directory .
151 [Container] 2025/12/16 16:39:26.313696 Assembling file list
152 [Container] 2025/12/16 16:39:26.313700 Expanding */
153 [Container] 2025/12/16 16:39:26.315810 No matching auto-discover report paths found
154 [Container] 2025/12/16 16:39:26.315827 Report auto-discover file discovery took 0.006239 seconds
155 [Container] 2025/12/16 16:39:26.315906 Phase complete: UPLOAD_ARTIFACTS State: SUCCEEDED
156 [Container] 2025/12/16 16:39:26.315913 Phase context status code: Message:
157
```

The Docker images were successfully pushed and stored in Amazon ECR for use in the ECS deployment.

The screenshot shows the AWS ECR console with the following details:

- Region:** United States (N. Virginia)
- Account ID:** 5159-6652-1120
- Selected Services:** VPC, EC2, Aurora and RDS, IAM, S3, Route 53, Elastic Container Registry, Elastic Container Service, CodeCommit
- Breadcrumbs:** Amazon ECR > Private registry > Repositories > Images
- Repository Name:** app_1
- Images Tab:** Selected
- Image Details:**

Image tags	Type	Created at	Image size	Image digest	Last pulled at
latest	Image	December 16, 2025, 22:09:25 (UTC+05.5)	22.99	sha256:cd1c95b8bd2...	December 16, 2025, 22:11:19 (UTC+05.5)
-	Image	December 16, 2025, 21:55:35 (UTC+05.5)	22.99	sha256:af4c086500e...	-
-	Image	December 15, 2025, 21:06:59 (UTC+05.5)	22.99	sha256:c71389ac0aa...	December 16, 2025, 21:07:24 (UTC+05.5)
- Actions:** Create, Delete, Copy URI, Details, Scan, View push commands

AWS Search [Alt+S] Account ID: 5159-6652-1120 ▾ United States (N. Virginia) ▾ Dhavanisha

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry Elastic Container Service CodeCommit

Amazon Elastic Container Service > Task definitions > app_1

app_1 (2) Info Last updated December 16, 2025, 22:34 (UTC+5:30) Deploy Actions Create new revision

Filter status Active ▾ Status

Task definition: revision app_1:3 Active app_1:2 Active

Express Mode New Clusters Namespaces Task definitions Account settings

Amazon ECR Repositories AWS Batch

The screenshot shows the AWS Elastic Container Service (ECS) Task Definitions page. On the left, there's a sidebar with links for Express Mode, Clusters, Namespaces, Task definitions (selected), Account settings, Amazon ECR, and AWS Batch. The main area displays the 'app_1' task definition with two revisions listed: 'app_1:3' and 'app_1:2'. A red box highlights the first revision, 'app_1:3'. Both revisions are marked as 'Active'. The top right shows the last update time as December 16, 2025, 22:34 (UTC+5:30), and there are buttons for Deploy, Actions, and Create new revision.

aws | Search [Alt+S] | Account ID: 5159-6652-1120 | United States (N. Virginia) | Dhavanisha

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry Elastic Container Service CodeCommit

Amazon Elastic Container Service Clusters my-demo-cluster Tasks

Last updated December 16, 2025, 22:37 (UTC+5:30)

Actions Create with Express Mode

my-demo-cluster

Cluster overview

ARN arn:aws:ecs:us-east-1:515966521120:cluster/my-demo-cluster	Status Active	CloudWatch monitoring Default	Registered container instances -
---	------------------	----------------------------------	-------------------------------------

Services Tasks

Draining	Active 1	Pending	Running 1
----------	-------------	---------	--------------

Services Tasks Infrastructure Metrics Scheduled tasks Configuration Event history Tags

Tasks (1)

Last updated December 16, 2025, 22:37 (UTC+5:30)

Task	Last status	Desired status	Task definition	Health status	Created at	Started by	St
58611ec5cc404dceb40ab...	Running	Running	app_1:3	Unknown	26 minutes ago	ecs-svc/16870367803...	26

Filter tasks by property or value

Filter desired status: Running

Filter launch type: Any launch type

Stop Run new task

Tell us what you think

Modified the code to test automated deployment and validated it through the load balancer endpoint.

The screenshot shows the AWS CodeCommit interface. The top navigation bar includes the AWS logo, a search bar, and account information (Account ID: 5159-6652-1120, United States (N. Virginia)). Below the navigation bar is a toolbar with icons for VPC, EC2, Aurora and RDS, IAM, S3, Route 53, Elastic Container Registry, Elastic Container Service, and CodeCommit. The main left sidebar is titled "Developer Tools" and "CodeCommit". It has a tree view with "Source • CodeCommit" expanded, showing "Getting started", "Repositories", "Code" (which is selected), "Pull requests", "Commits", "Branches", "Git tags", "Settings", and "Approval rule templates". Other sections like "Artifacts • CodeArtifact", "Build • CodeBuild", "Deploy • CodeDeploy", "Pipeline • CodePipeline", and "Settings" are shown with a right-pointing arrow. The central content area shows the breadcrumb path: Developer Tools > CodeCommit > Repositories > appcid > File. The title "Edit a file" is displayed above the code editor. The code editor shows the "appcid / index.html" file with the following content:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Nginx Docker Test</title>
5 </head>
6 <body>
7   <h1>Hi, We have configured CICD an testing now..</h1>
8 </body>
9 </html>
```

The code editor has line numbers on the left and a status bar at the bottom. A reference dropdown on the right shows "master".

Clicked “Release change” to trigger the pipeline and test the CI/CD automation flow.

The screenshot shows the AWS CodePipeline console for a pipeline named "my-app-cicd". A green success message at the top right says "Congratulations! The pipeline my-app-cicd has been created." A cursor arrow points to the "Release change" button in the top navigation bar. The pipeline view shows three stages: Source, Build, and Deploy. The Source stage is completed with a green checkmark, showing an AWS CodeCommit action that succeeded just now. The Build stage is in progress, with one task (AWS CodeBuild) shown as "In progress: 1" and succeeded just now. The Deploy stage has not run yet, indicated by a grey minus sign. The pipeline status is "Success".

aws | Search [Alt+S] Account ID: 5159-6652-1120
United States (N. Virginia) ▾ Dhavanisha

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry

Developer Tools > CodePipeline > Pipelines > my-app-cicd

① Introducing the new pipeline experience
We've redesigned the pipeline view to streamline the monitoring and debugging experience. [Let us know what you think.](#) Or go back to the old experience.

Success
Congratulations! The pipeline my-app-cicd has been created.

my-app-cicd Edit Stop execution Create trigger Clone pipeline Release change

Pipeline Executions Triggers Settings Tags Stage

Source 62f6d874-285e-4225-8e51-656a905f8635 All actions succeeded.
Source AWS CodeCommit Just now
a22fc2ef Source: My first c

Build 62f6d874-285e-4225-8e51-656a905f8635 In progress: 1
Build AWS CodeBuild Just now
a22fc2ef Source: My first c

Deploy Didn't Run
Deploy Amazon ECS

The target group shows the new target as healthy while the old IP is in draining state during deployment.

The screenshot shows the AWS EC2 Target groups console. On the left, there's a navigation sidebar with sections like Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager, Images, Elastic Block Store, Network & Security, Load Balancing, Target Groups, Auto Scaling, and Settings. The main area displays 'Target groups (1/1)' with a single entry: 'ECS-TG-NODE'. Below it, under 'Target group: ECS-TG-NODE', is a table titled 'Registered targets (2)'. The table has columns: IP address, Port, Zone, Health status, Health status details, Administrative override, Override details, and Anomaly detection. The first row shows '10.0.18.47' in the 'us-east-1b (use1-az4)' zone as 'Healthy'. The second row shows '10.0.0.233' in the 'us-east-1a (use1-az2)' zone as 'Draining'. A red box highlights the 'Draining' status of the second target.

IP address	Port	Zone	Health status	Health status details	Administrative override	Override details	Anomaly detection
10.0.18.47	80	us-east-1b (use1-az4)	Healthy	-	No override	No override is currently ...	Normal
10.0.0.233	80	us-east-1a (use1-az2)	Draining	Target deregistration i...	No override	No override is currently ...	Normal

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LoadBalancers:

Account ID: 5159-6652-1120
Dhavanish

VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry Elastic Container Service CodeCommit

EC2 Load balancers

Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Capacity Manager New

Images

Elastic Block Store
Volumes
Snapshots
Lifecycle Manager

Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces

Load Balancing
Load Balancers
Target Groups
Trust Stores

Auto Scaling
Auto Scaling Groups
Settings

Load balancers (1/1) What's new?

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Filter load balancers

Name State Type Scheme IP address type VPC ID Availability Zones Security groups

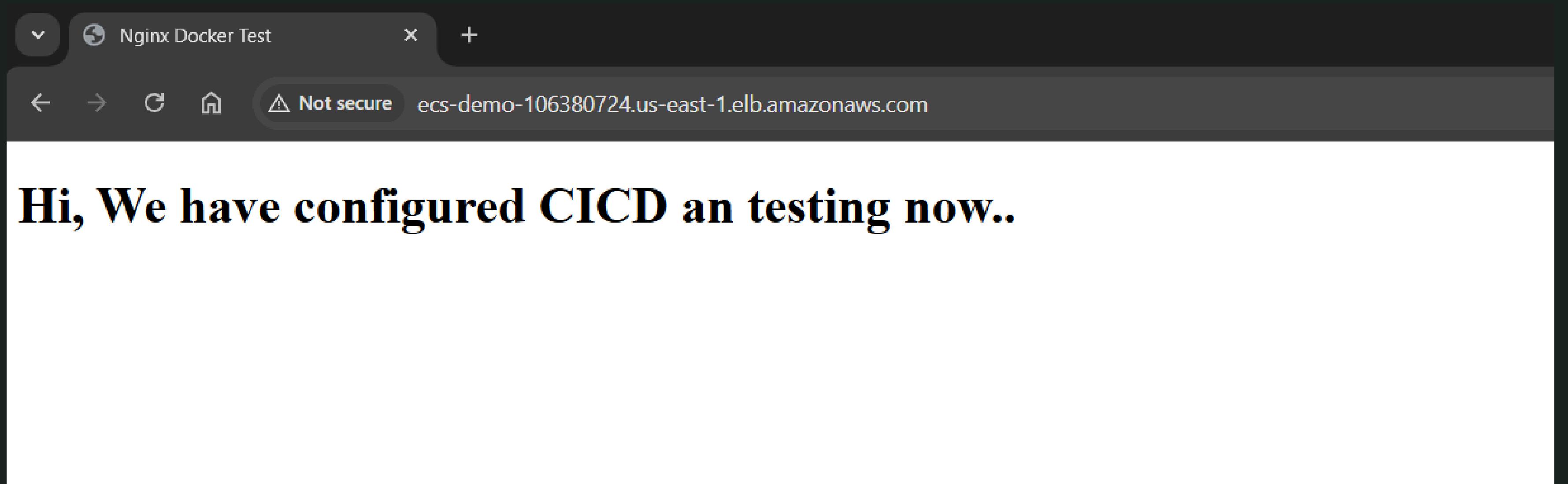
Name	State	Type	Scheme	IP address type	VPC ID	Availability Zones	Security groups
ECS-DEMO	Active	application	Internet-facing	IPv4	vpc-0d7fbaba068b994cd	2 Availability Zones	sg-0f190d62fc5e

Load balancer: ECS-DEMO

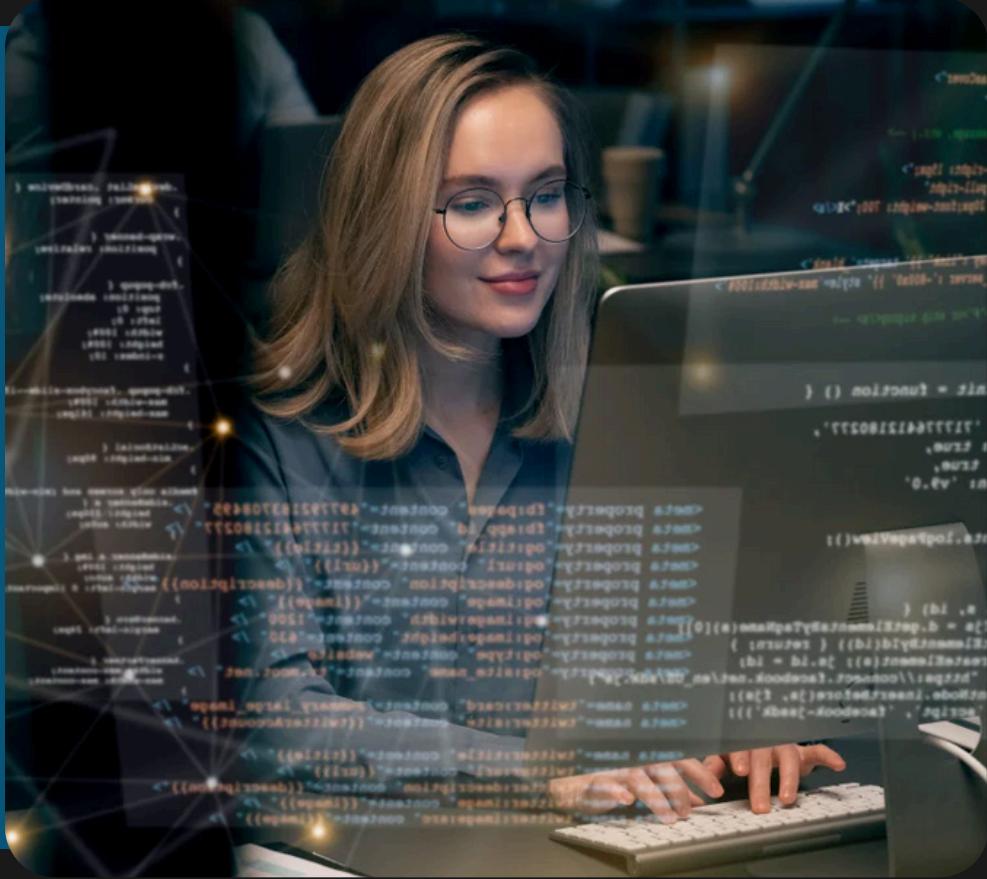
Details

Load balancer type Application	Status Active	VPC vpc-0d7fbaba068b994cd	Load balancer IP address type IPv4
Scheme Internet-facing	Hosted zone Z35SXDOTRQ7X7K	Availability Zones subnet-0074188ec817364e0 us-east-1b (use1-az4) subnet-0e32fd3d4fc888352 us-east-1a	Date created December 14, 2025, 20:27 (UTC+05:30)
Load balancer ARN arn:aws:elasticloadbalancing:us-east-1:515966521120:loadbalancer/app/ECS-DEMO/ea83c65ab2b6c14	DNS name copied ECS-DEMO-106380724.us-east-1.elb.amazonaws.com (A Record)		

Verified the CI/CD automation by accessing the application through the load balancer DNS name and confirming the updated content is live.



CONCLUSION



This project demonstrates the successful transition from manual container deployment to a fully automated CI/CD pipeline on AWS ECS. By validating the infrastructure manually first and then automating the build and deployment using AWS CodeCommit, CodePipeline, and CodeBuild, the project achieves reliable, scalable, and zero-downtime deployments. Overall, it reflects real-world DevOps practices and practical experience in deploying Dockerized applications on AWS.



THANK YOU

GET IN TOUCH



dhavanisha.jp@gmail.com

