FROM MANUAL DEPLOYMENT TO CI/CD AUTOMATION

Docker Deployment on AWS ECS

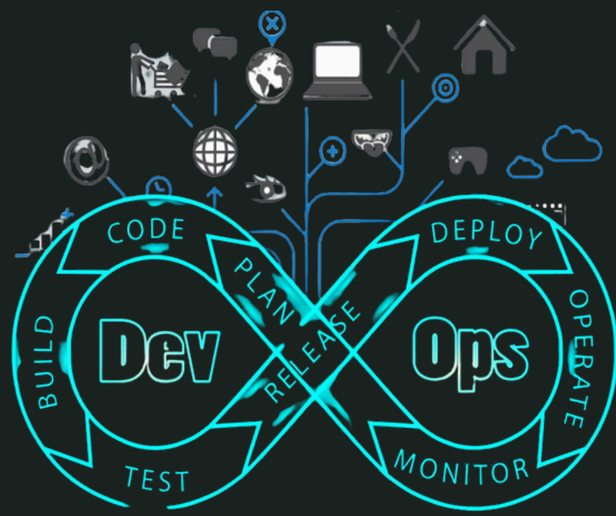
GitHub Repository:https://github.com/DHAVANISHAJ/Manual-to-CI-CD-Automation--Docker-Deployment-on-AWS-ECS.git

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.

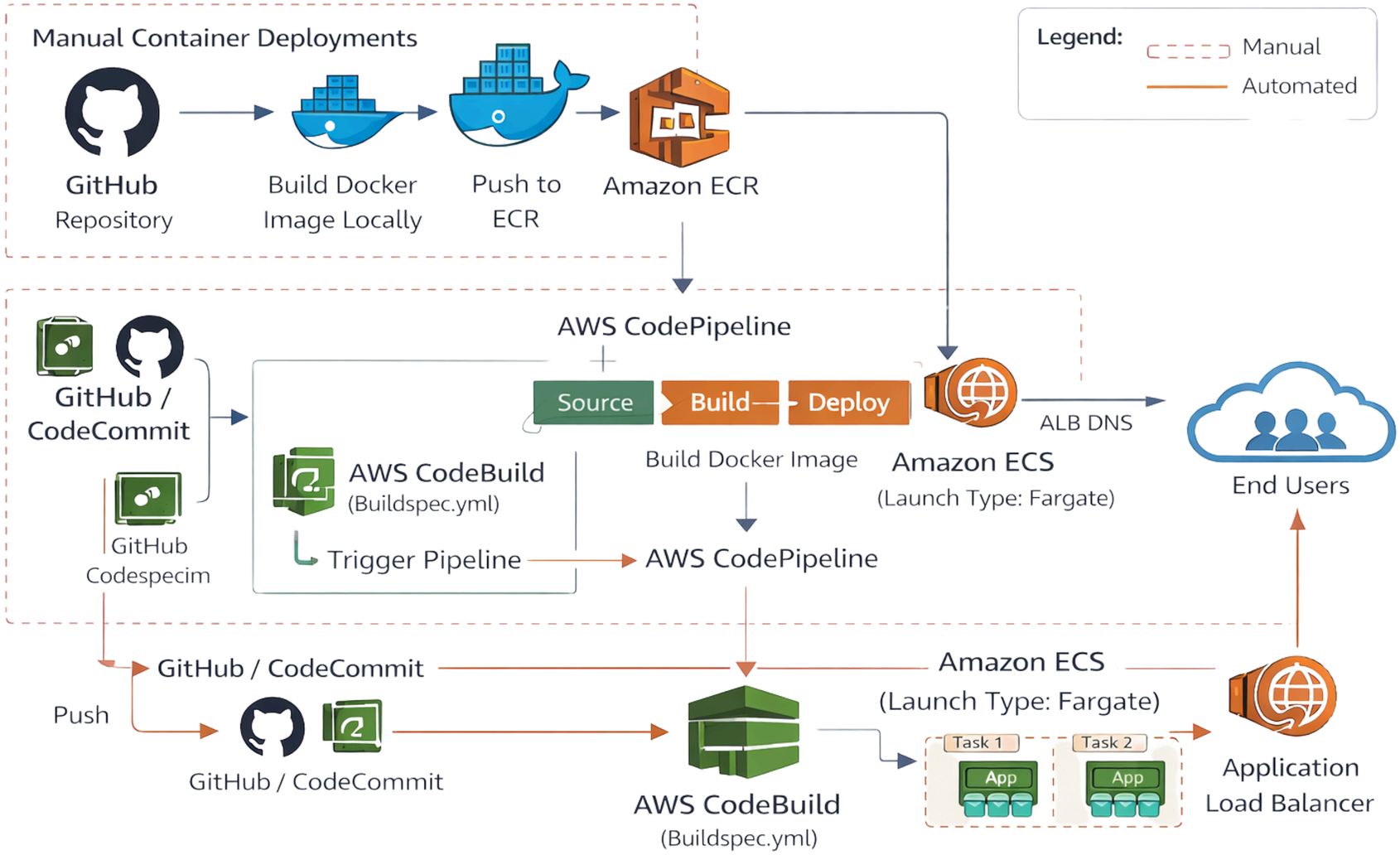
Tools & Technologies Used



- 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

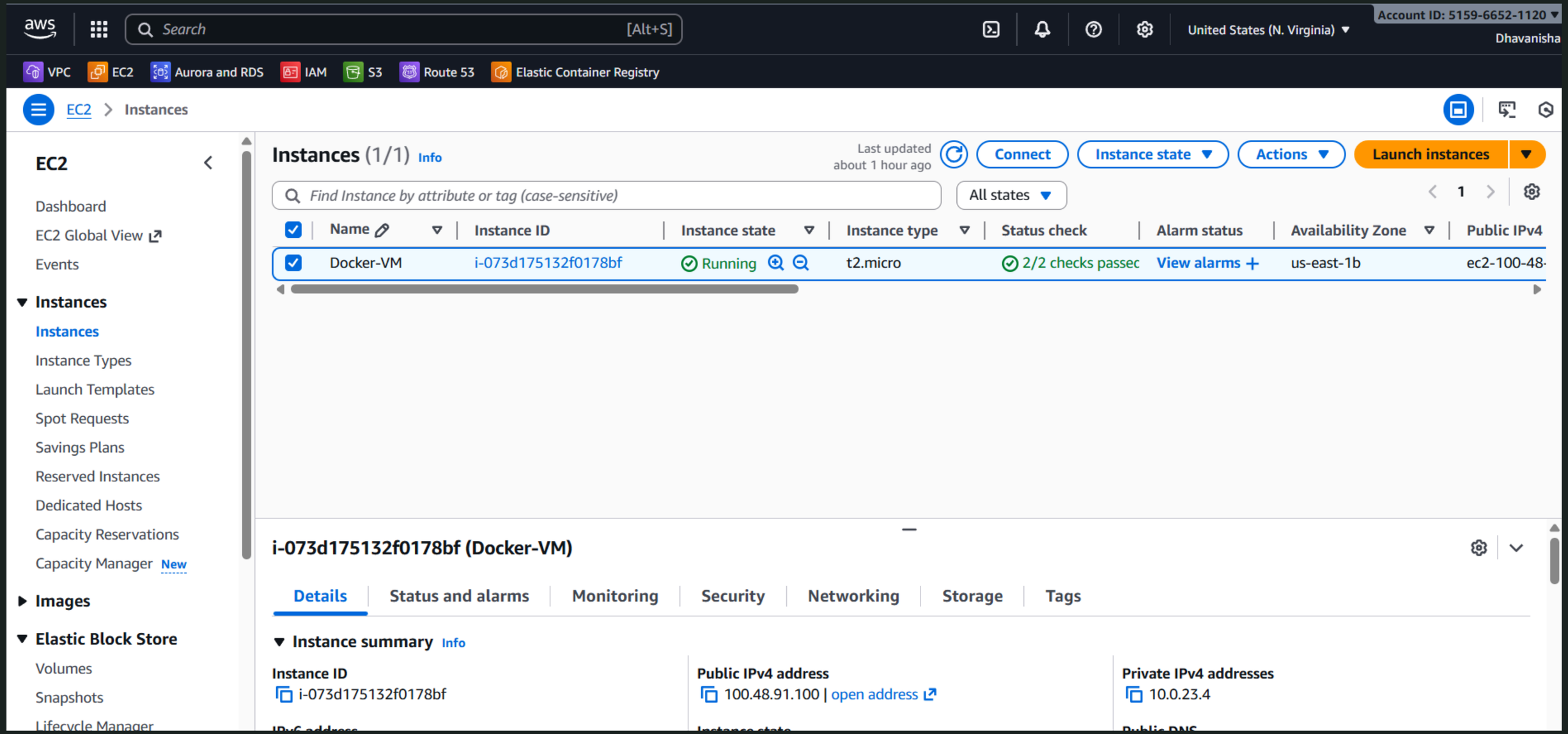
Amazon ECS (Fargate):

- Zero EC2 management
- Stateless container deployment

Automated deployment validation:

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

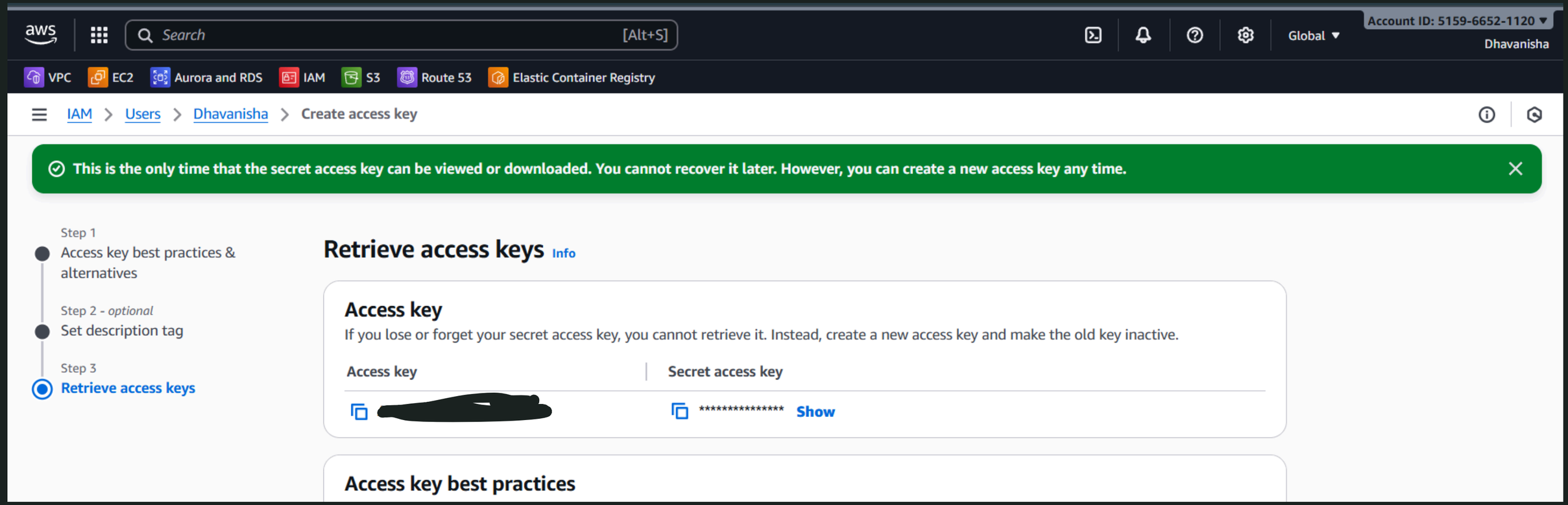
EC2 instances were launched inside the VPC and Docker was installed to run containers.



The 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          containerd-2.1.5-1.amzn2023.0.1.x86_64          docker-25.0.13-1.amzn2023.0.2.x86_64
  iptables-libs-1.8.8-3.amzn2023.0.2.x86_64              iptables-nft-1.8.8-3.amzn2023.0.2.x86_64        libcgroup-3.0-1.amzn2023.0.1.x86_64
  libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64     libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64       libnftnl-1.2.2-2.amzn2023.0.2.x86_64
  pigz-2.5-1.amzn2023.0.3.x86_64                         runc-1.3.3-2.amzn2023.0.1.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 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]: [redacted]
AWS Secret Access Key [None]: [redacted]
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:alpine@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:alpine@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.

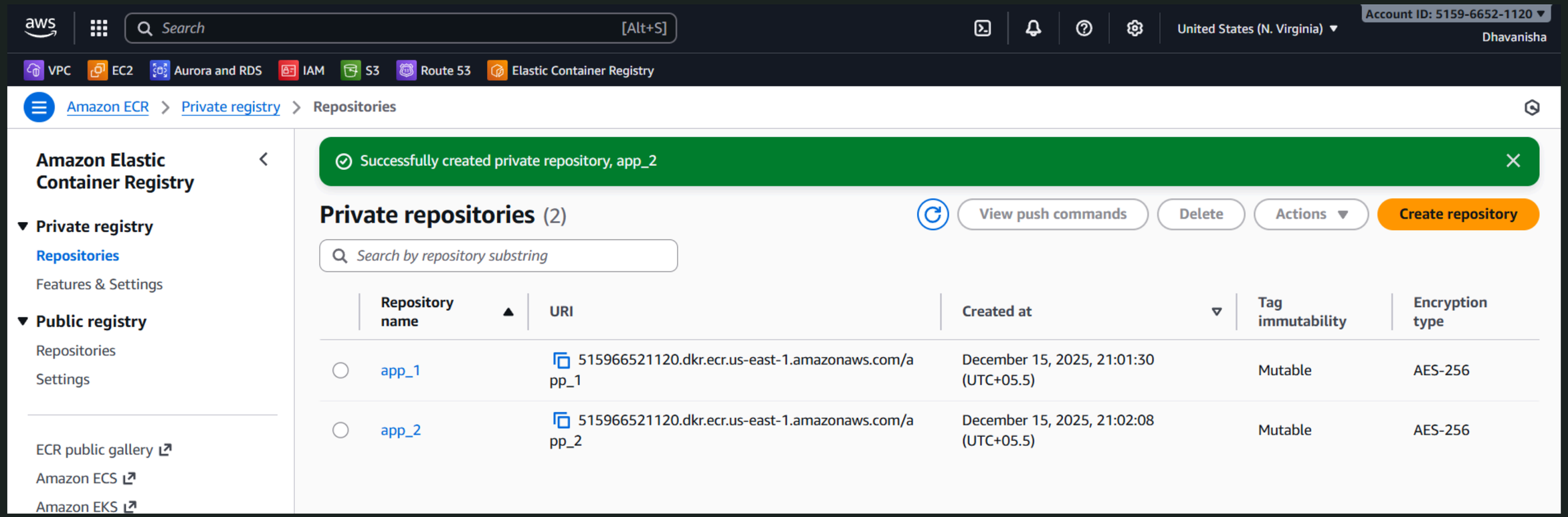
```
[root@ip-10-0-23-4 app_2]# docker run -it -p 4000:80 app_1:latest
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
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
15506f09e32a05a09d9413fd9f15dc72c4ba88b0af87b9f17027614cfd8c0cec
[root@ip-10-0-23-4 ~]#
```

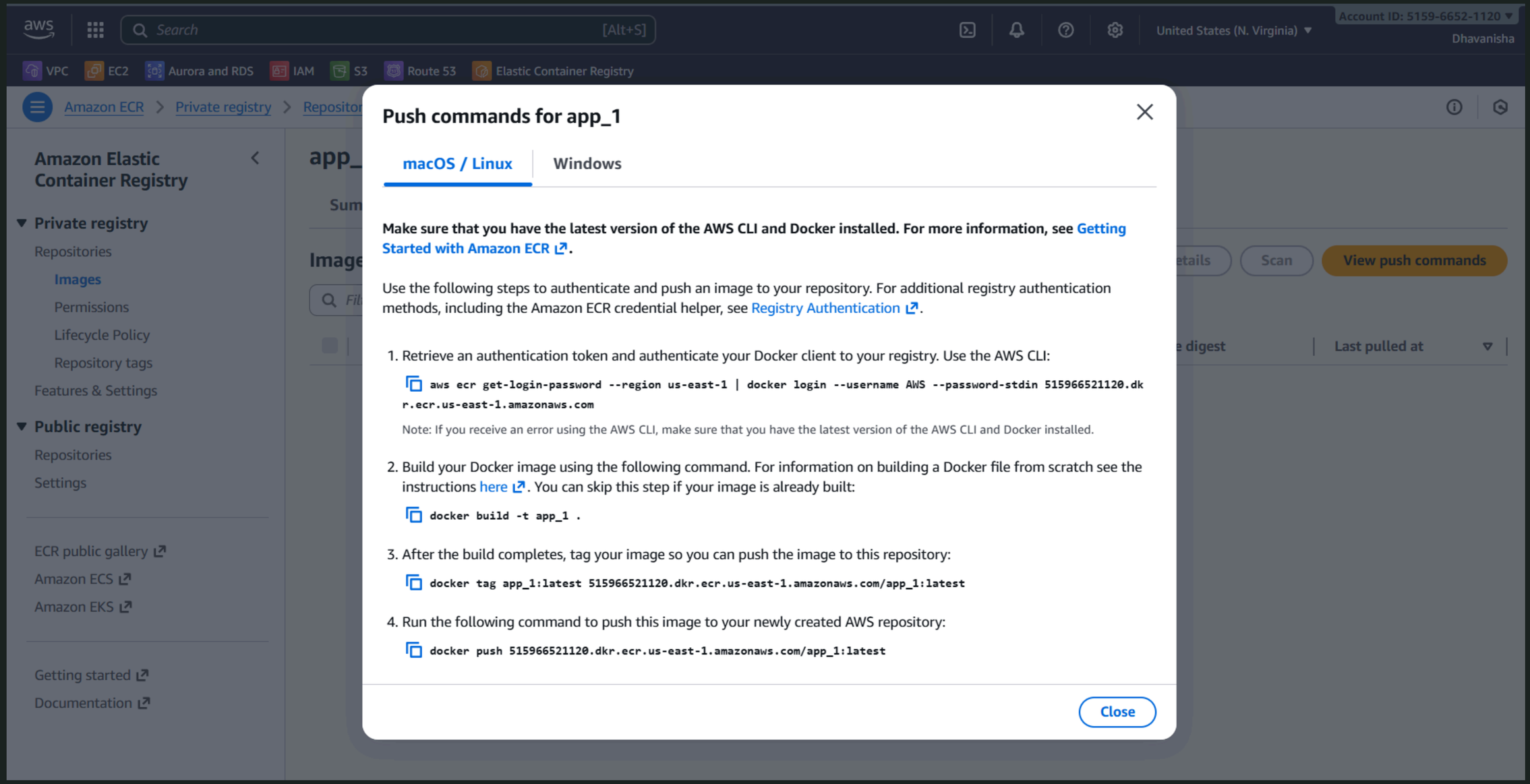
Verified Application 1 running successfully in the browser using public ip.

Verified Application 2 running successfully in the browser.

ECR repositories were created.

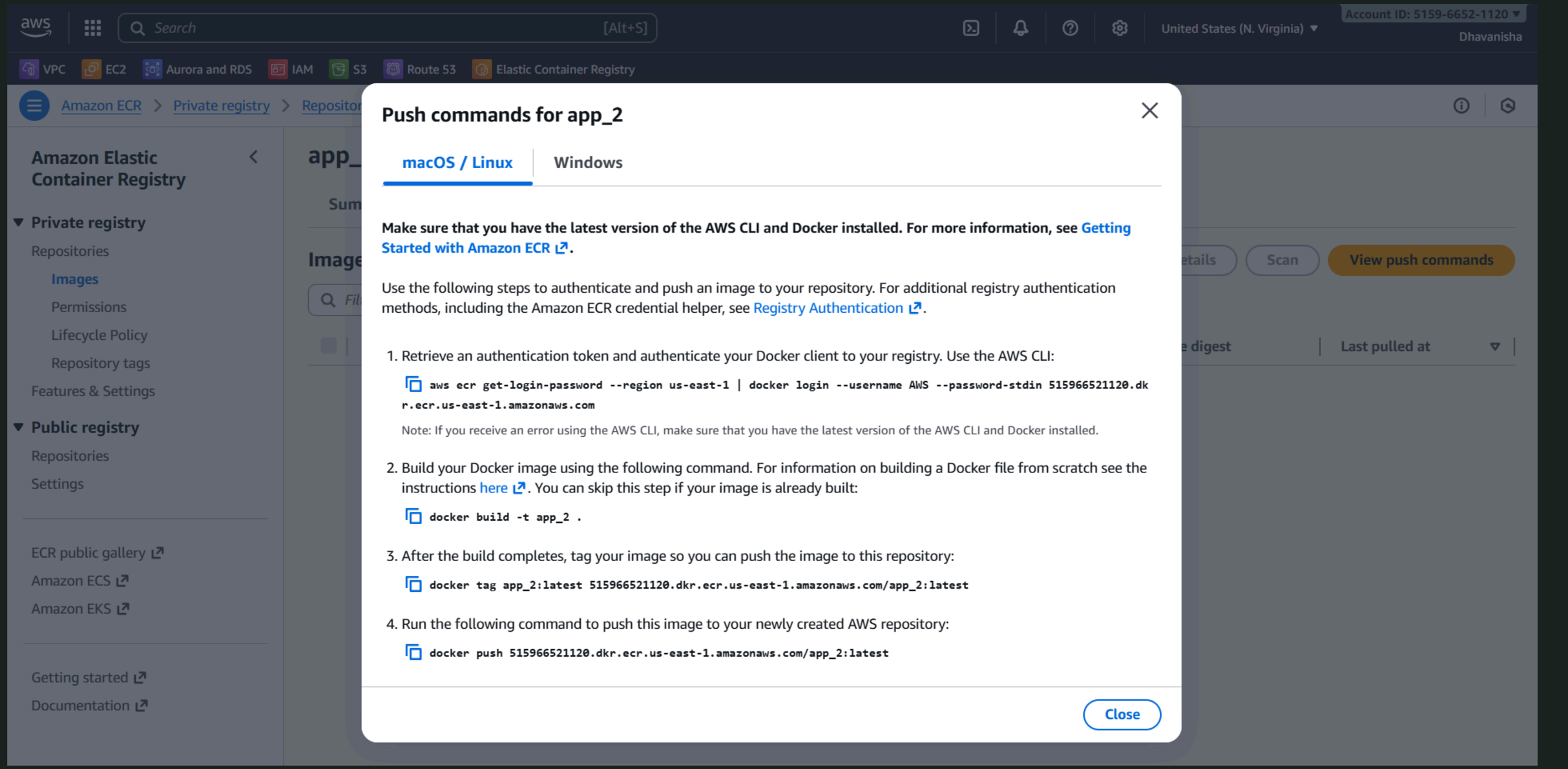


Used AWS-provided commands to authenticate Docker with ECR for both Repositories.

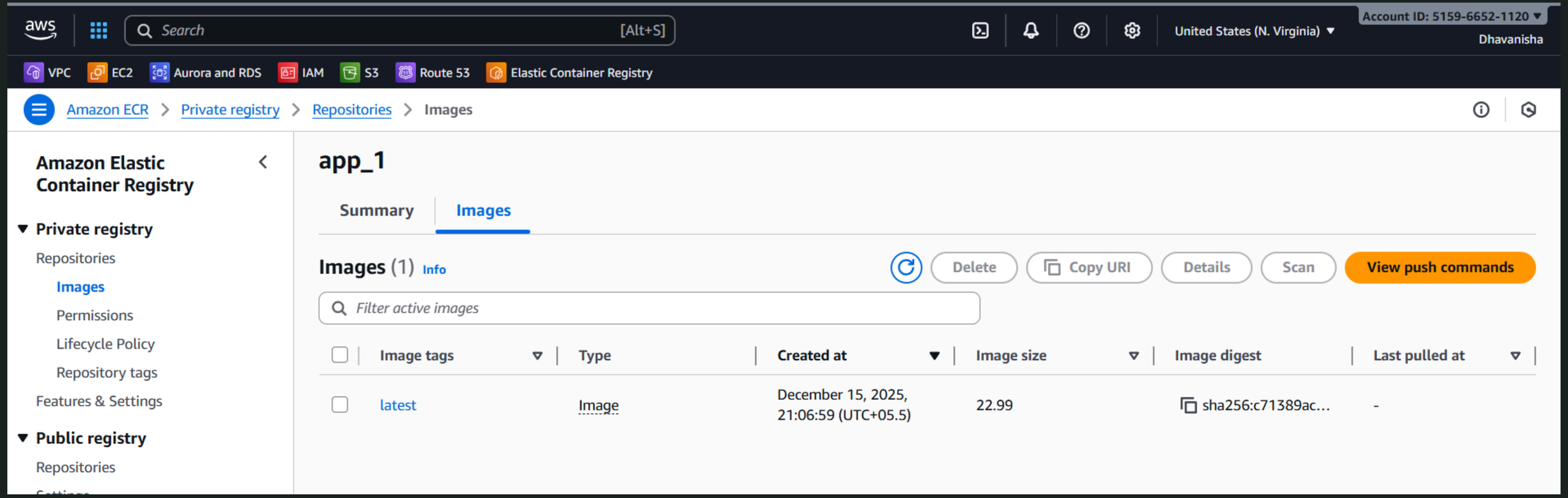


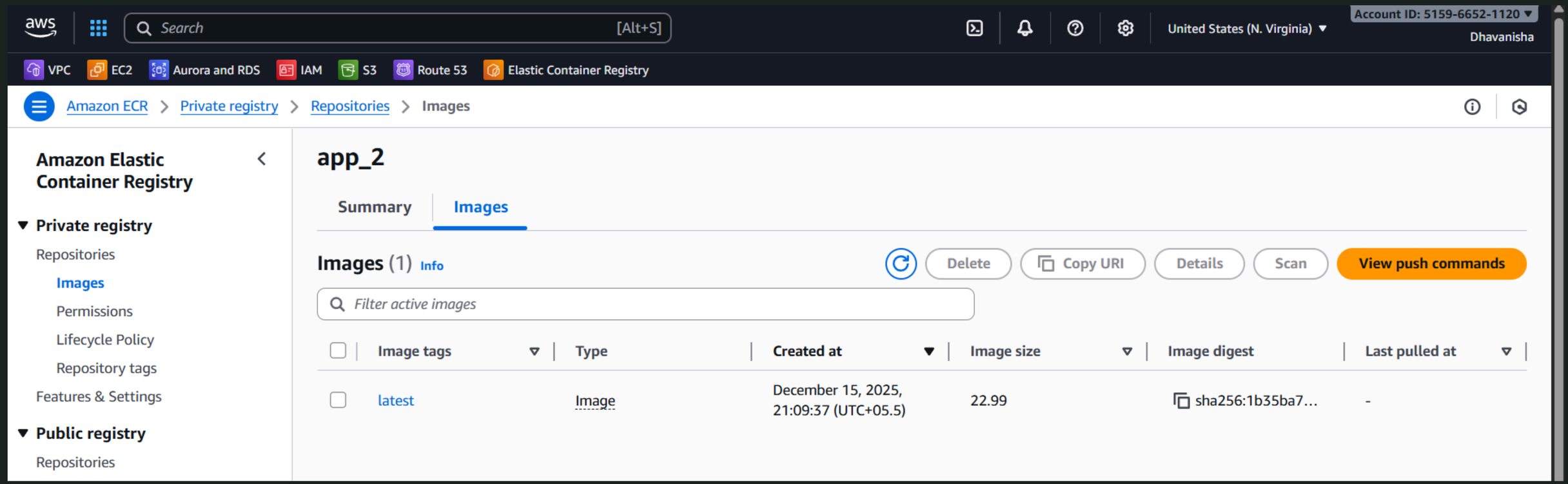
Tagged and pushed Docker images to Amazon ECR successfully.

```
[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 ~]#
```

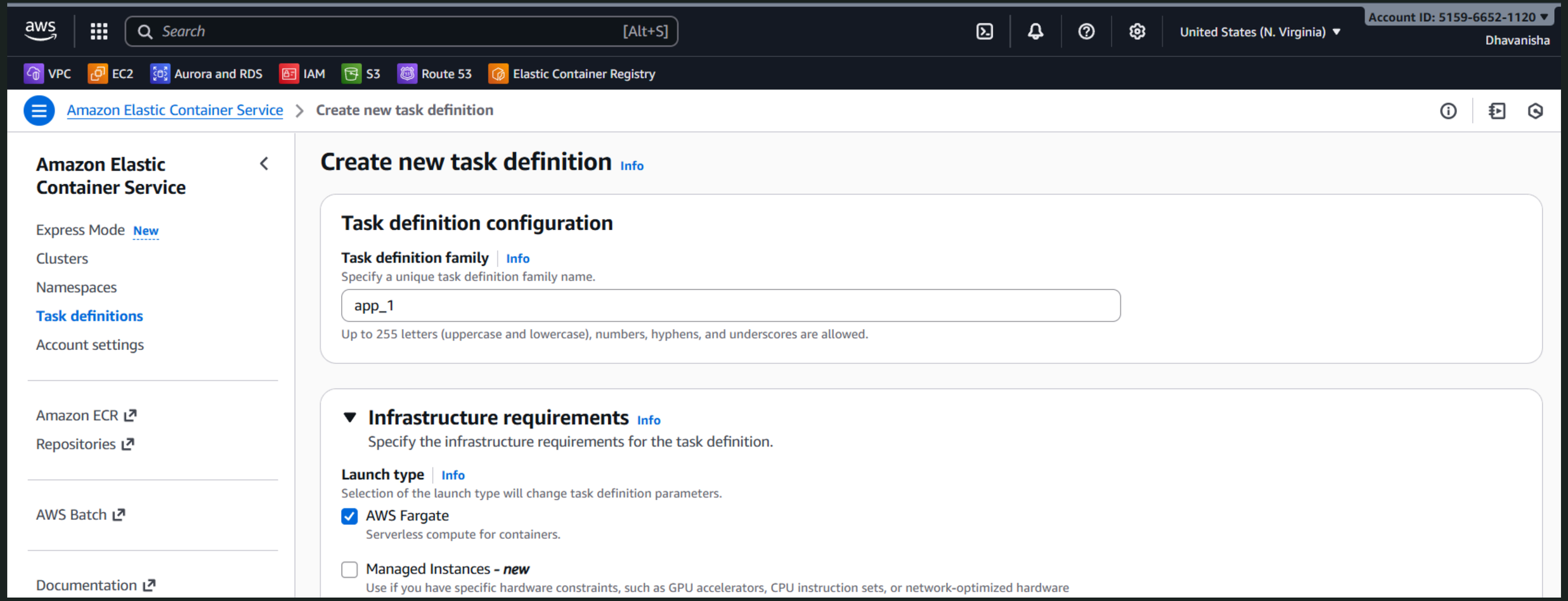

aws
Search
[Alt+S]
United States (N. Virginia)
Account ID: 5159-6652-1120
Dhavanisha
VPC
EC2
Aurora and RDS
IAM
S3
Route 53
Elastic Container Registry
Amazon ECR
Private registry
Amazon Elastic Container Registry
Private registry
Repositories
Images
Permissions
Lifecycle Policy
Repository tags
Features & Settings
Public registry
Repositories
Settings
ECR public gallery
Amazon ECS
Amazon EKS
Getting started
Documentation
Scan
View push commands
Last pulled at
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
5aa68bbbc67e: Pushed
latest: digest: sha256:1b35ba79f921af77d9c10b30a217165830a0ce624f5fa761ea9de5f6254d1118 size: 2196
[root@ip-10-0-23-4 ~]#
```

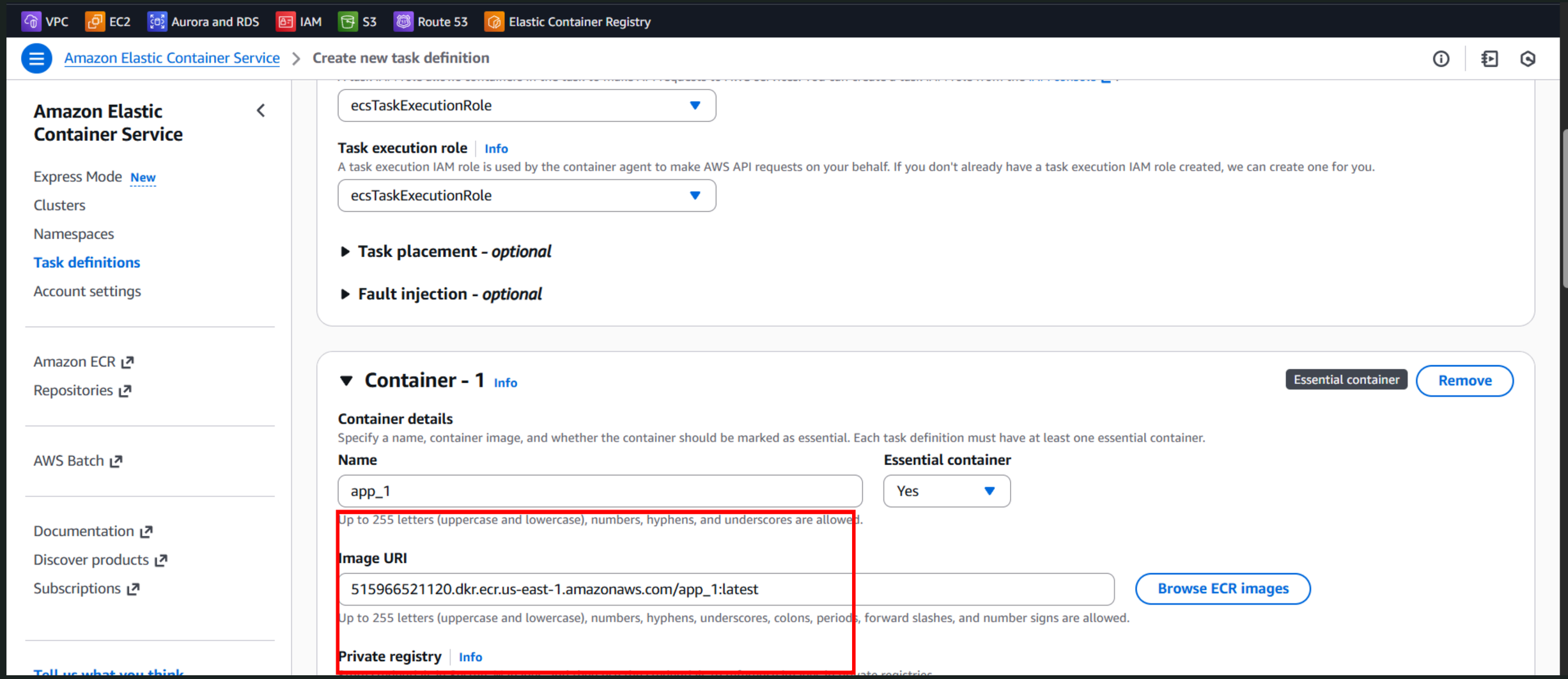

aws
Search
[Alt+S]
United States (N. Virginia)
Account ID: 5159-6652-1120
Dhavanisha
VPC
EC2
Aurora and RDS
IAM
S3
Route 53
Elastic Container Registry
Amazon ECR
Private registry
Repositories
Images
Amazon Elastic Container Registry
Private registry
Repositories
Images
Permissions
Lifecycle Policy
Repository tags
Features & Settings
Public registry
Repositories
app_1
Summary
Images
Images (1) Info
Delete
Copy URI
Details
Scan
View push commands
Filter active images
Image tags
Type
Created at
Image size
Image digest
Last pulled at
latest
Image
December 15, 2025, 21:06:59 (UTC+05.5)
22.99
sha256:c71389ac...
-


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 ECR
Private registry
Repositories
Images
Amazon Elastic Container Registry
Private registry
Repositories
Images
Permissions
Lifecycle Policy
Repository tags
Features & Settings
Public registry
Repositories
app_2
Summary
Images
Images (1) Info
Delete
Copy URI
Details
Scan
View push commands
Filter active images
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.



copy paste the ECR repostory URI here



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.

- [x] 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

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

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

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

Essential container

Yes

Image URI

515966521120.dkr.ecr.us-east-1.amazonaws.com/app_2:latest

Browse ECR images

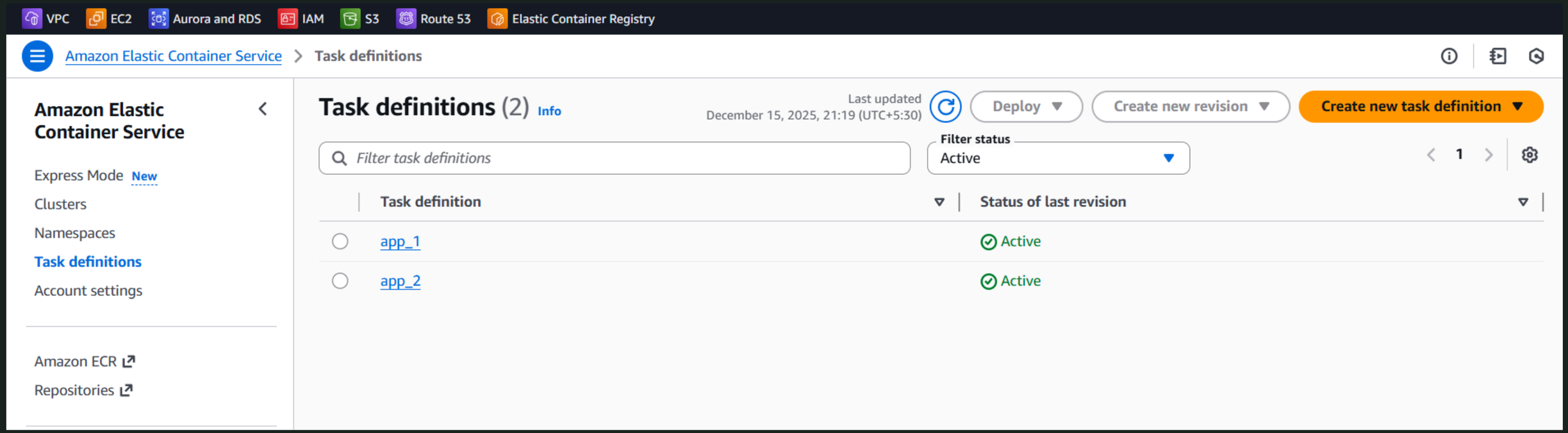
Up to 255 letters (uppercase and lowercase), numbers, hyphens, underscores, colons, periods, forward slashes, and number signs are allowed.

Private registry Info

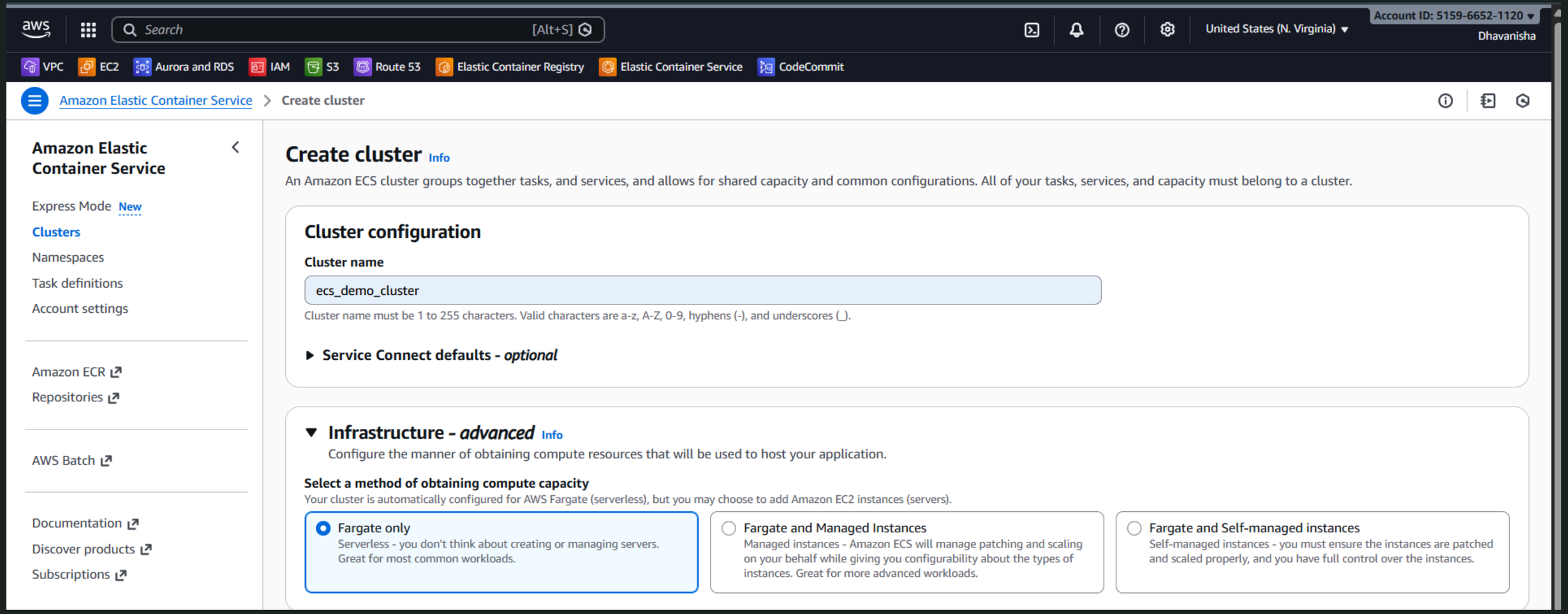
Store credentials in Secrets Manager, and then use the credentials to reference images in private registries.

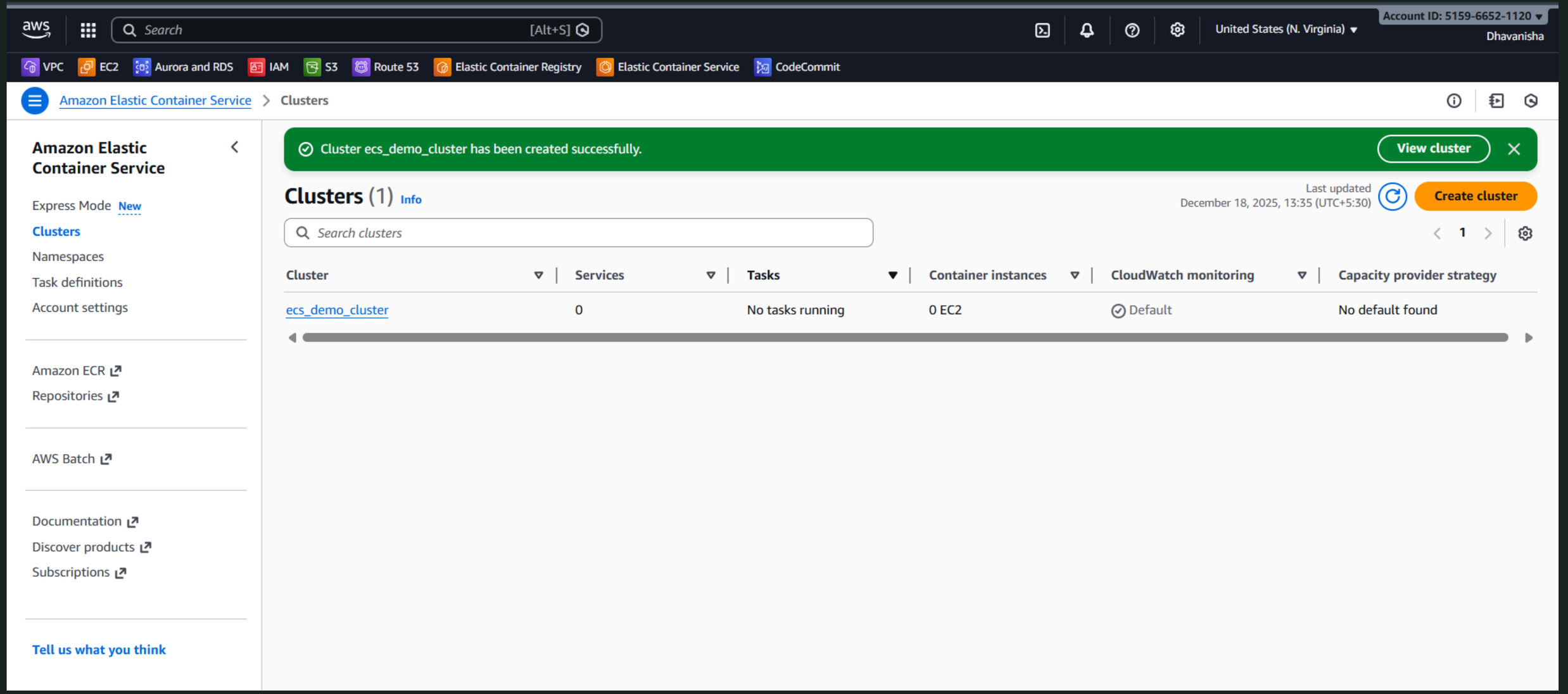
Private registry authentication

Task definition has been created for both application

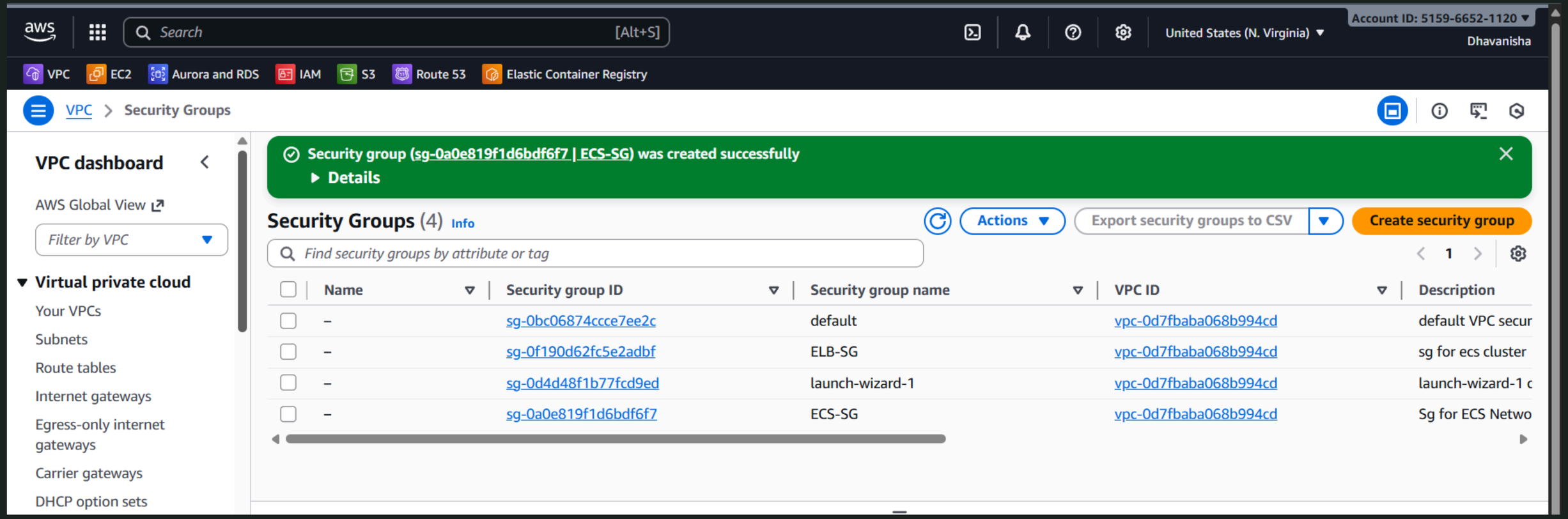


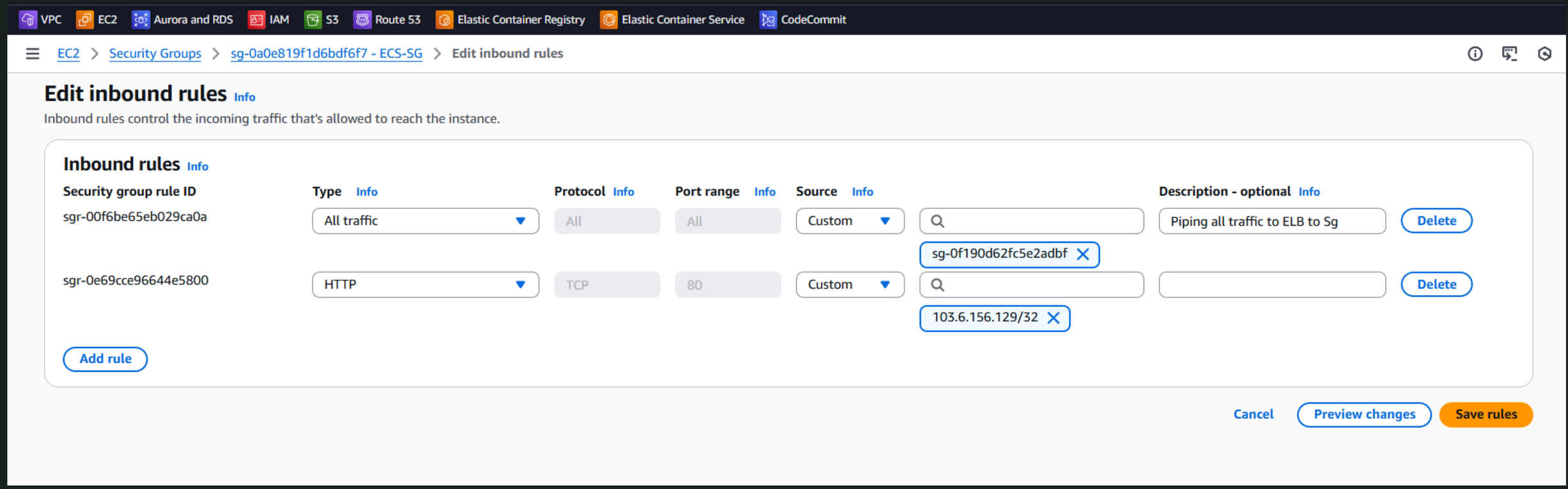
Created a new ECS cluster named ecs_demo_cluster.




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
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.
View cluster
Clusters (1)
Info
Last updated
December 18, 2025, 13:35 (UTC+5:30)
Create cluster
Search clusters
1
Cluster
Services
Tasks
Container instances
CloudWatch monitoring
Capacity provider strategy
ecs_demo_cluster
0
No tasks running
0 EC2
Default
No default found

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




VPC
EC2
Aurora and RDS
IAM
S3
Route 53
Elastic Container Registry
Elastic Container Service
CodeCommit
EC2 > Security Groups > sg-0a0e819f1d6bdf6f7 - ECS-SG > Edit inbound rules
Edit inbound rules Info
Inbound rules control the incoming traffic that's allowed to reach the instance.
Inbound rules Info
Security group rule ID
Type Info
Protocol Info
Port range Info
Source Info
Description - optional Info
sgr-00f6be65eb029ca0a
All traffic
All
All
Custom
sg-0f190d62fc5e2adbf
Piping all traffic to ELB to Sg
Delete
sgr-0e69cce96644e5800
HTTP
TCP
80
Custom
103.6.156.129/32
Delete
Add rule
Cancel
Preview changes
Save rules

Edit inbound rules Info

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

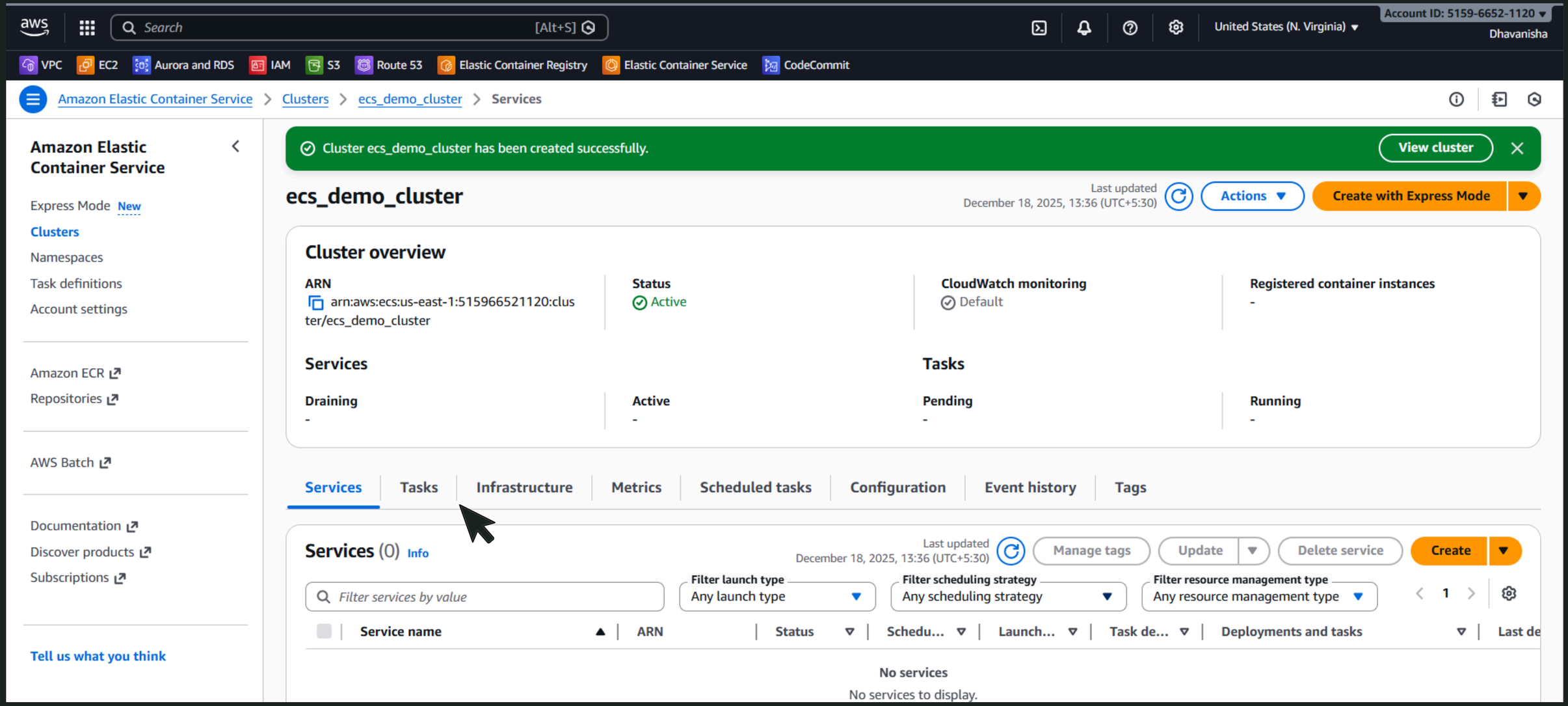
Inbound rules Info

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

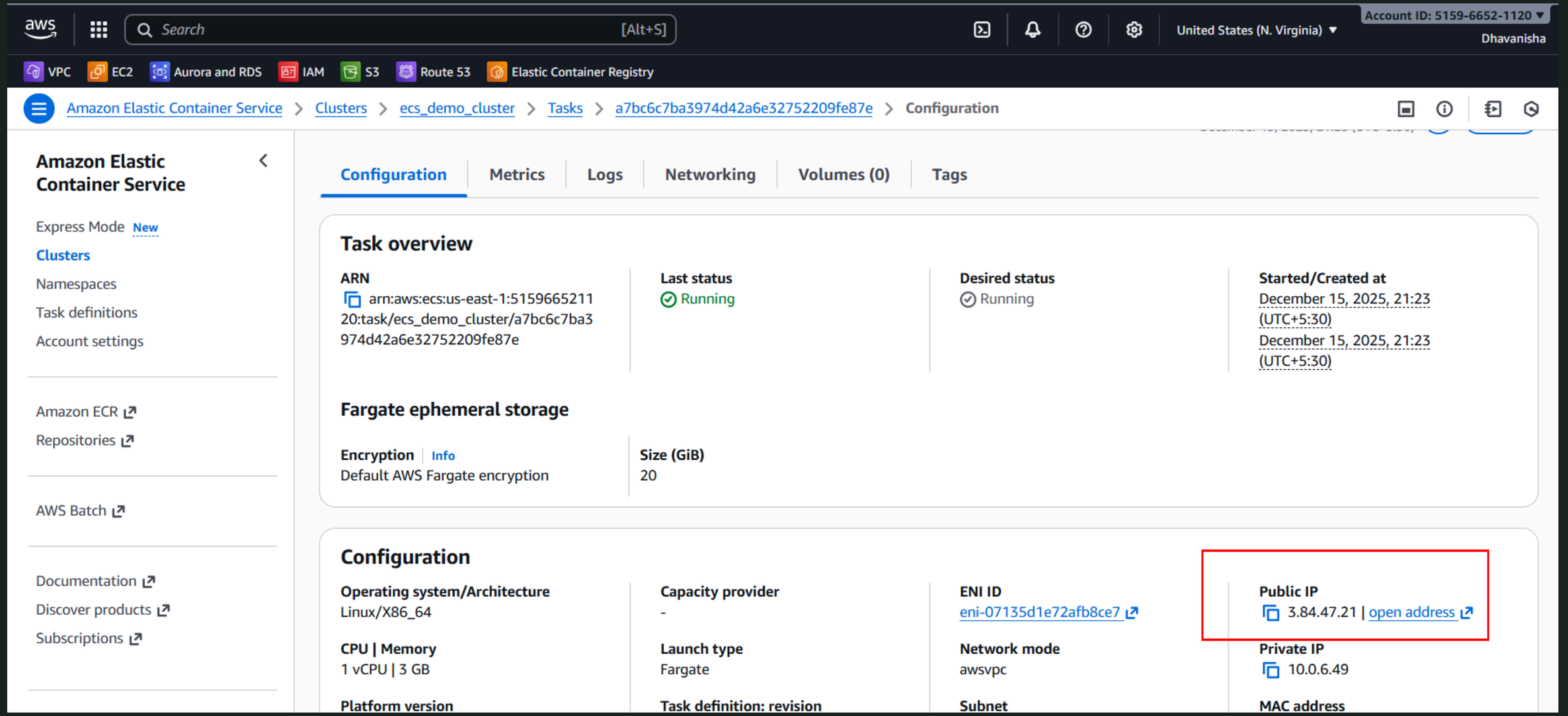
Add rule

Cancel | Preview changes | Save rules

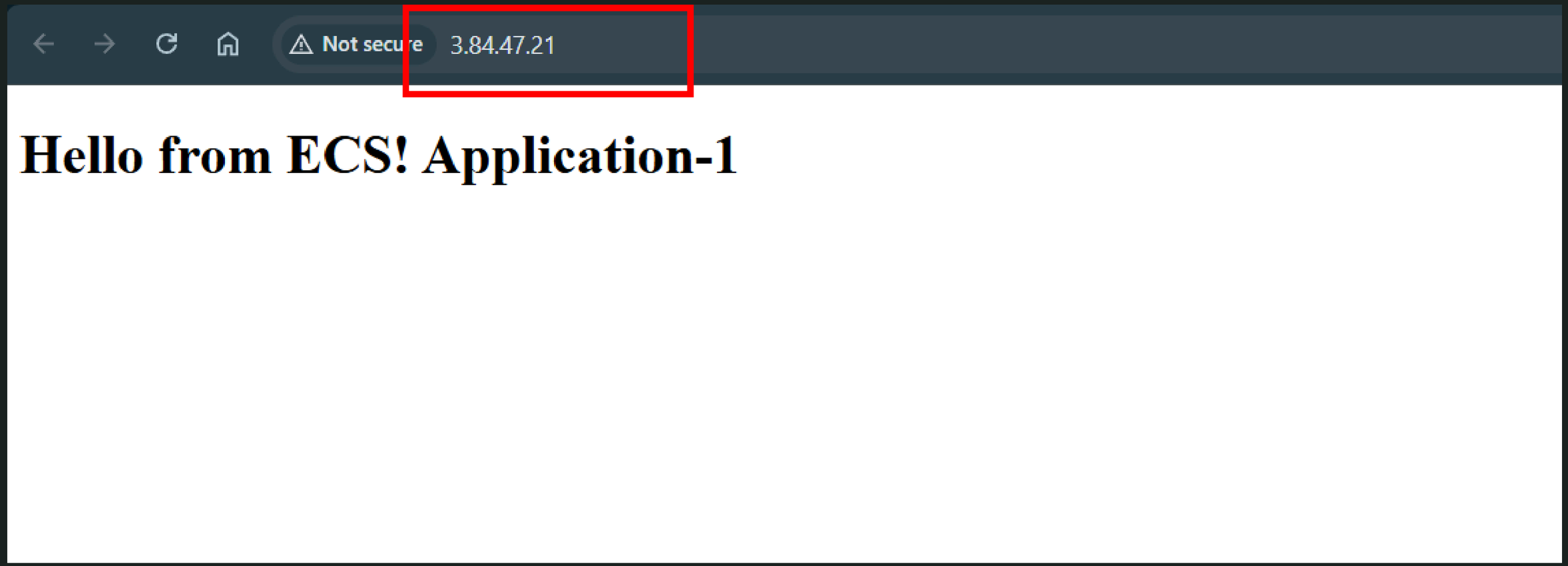
After configuring the security group, the ECS task was executed in the cluster to verify the application.

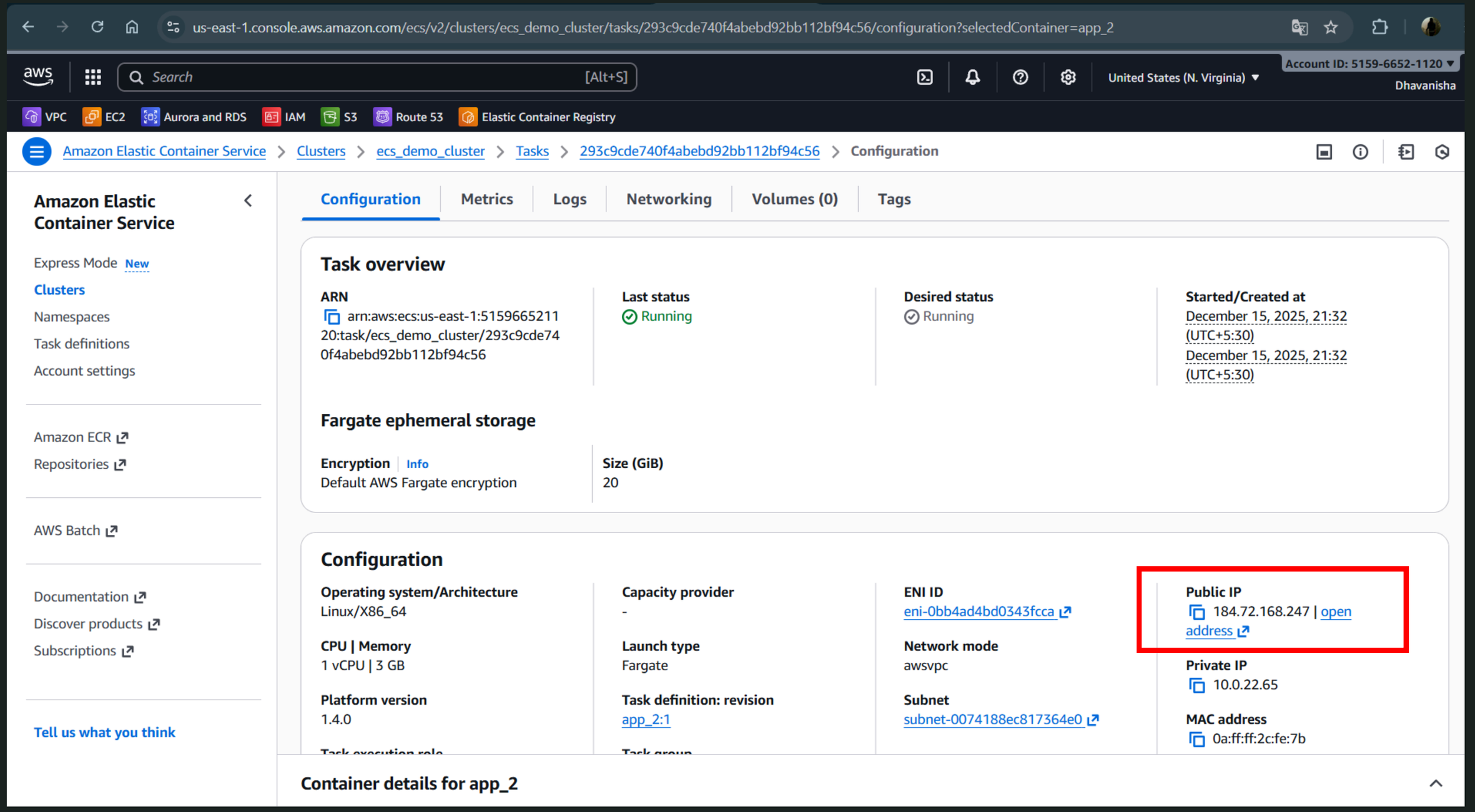


Use this public IP to test the both application.



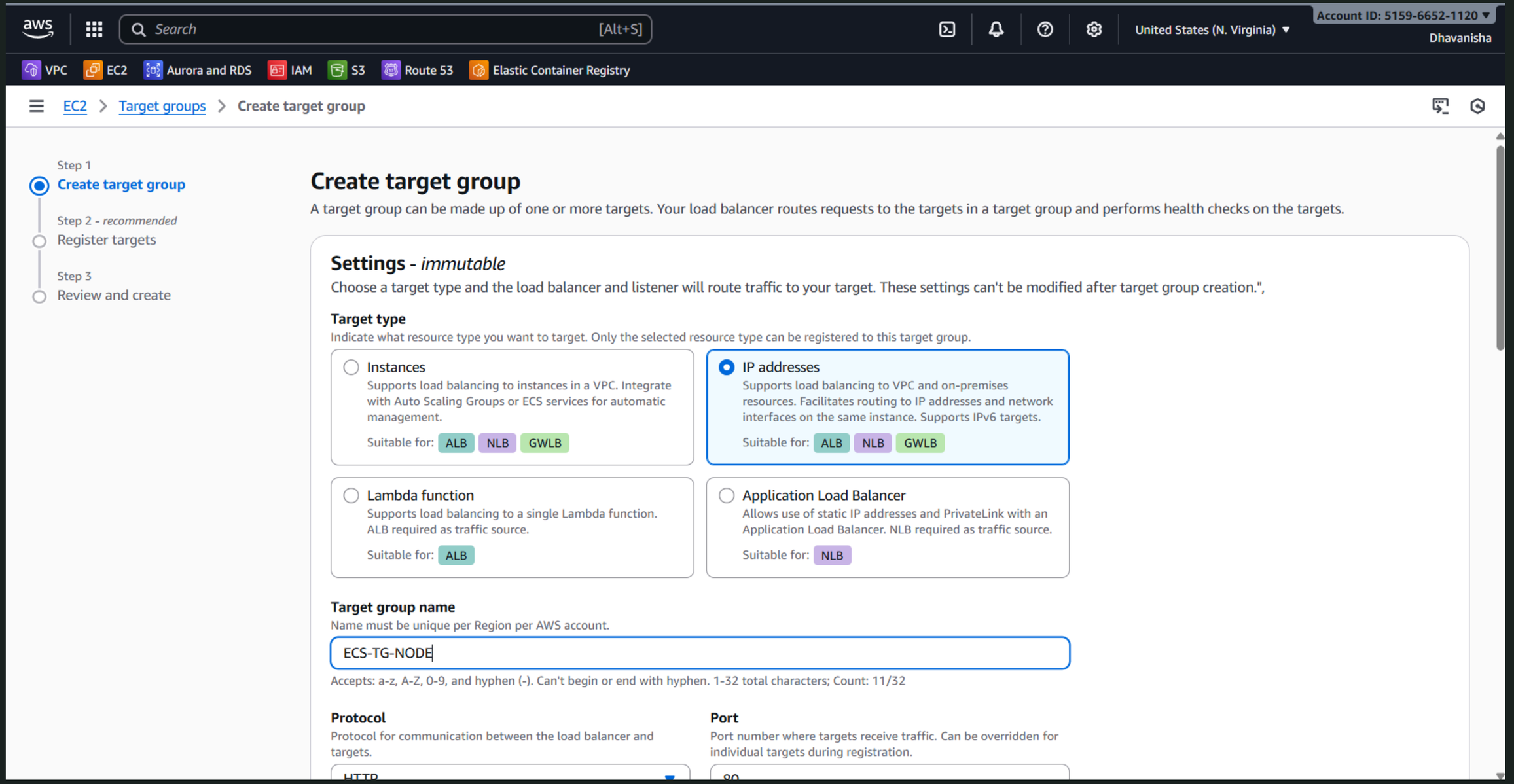
Verified Application 1 running successfully in the browser.




us-east-1.console.aws.amazon.com/ecs/v2/clusters/ecs_demo_cluster/tasks/293c9cde740f4abebd92bb112bf94c56/configuration?selectedContainer=app_2
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 > 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
Metrics
Logs
Networking
Volumes (0)
Tags
Task overview
ARN
arn:aws:ecs:us-east-1:515966521120:task/ecs_demo_cluster/293c9cde740f4abebd92bb112bf94c56
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
CPU | Memory
1 vCPU | 3 GB
Platform version
1.4.0
Capacity provider
-
Launch type
Fargate
Task definition: revision
app_2:1
ENI ID
eni-0bb4ad4bd0343fcca
Network mode
awsvpc
Subnet
subnet-0074188ec817364e0
Public IP
184.72.168.247 | open address
Private IP
10.0.22.65
MAC address
0a:ff:ff:2c:fe:7b
Container details for app_2

Verified Application 2 running successfully in the browser.

Create a target group for the load balancer.



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	Target type	Protocol : Port	Protocol version
ECS-TG-NODE	IP	HTTP: 80	HTTP1

VPC	IP address type
vpc-0d7fbaba068b994cd	IPv4

Health check details

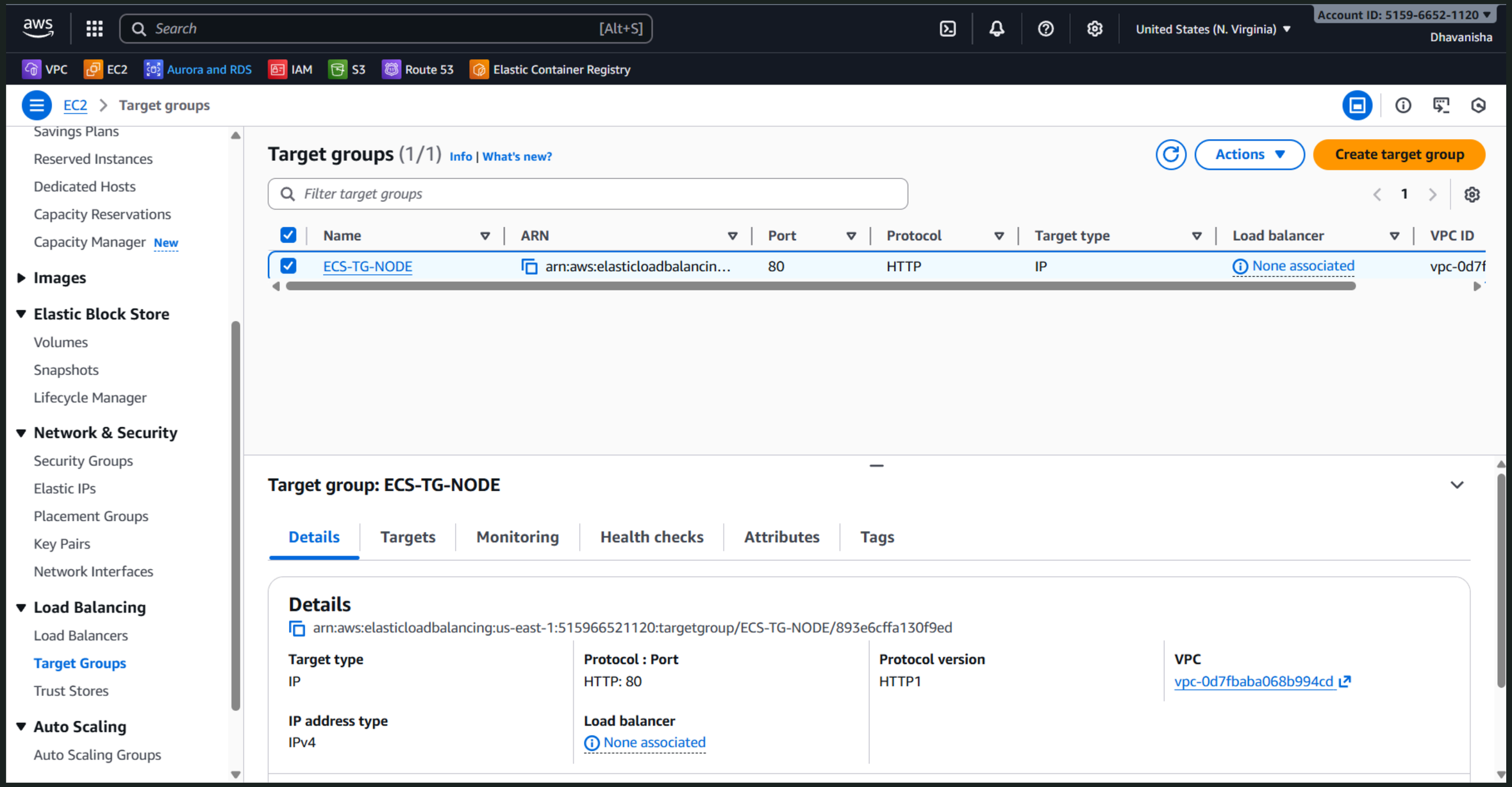
Health check protocol	Health check path	Health check port	Interval
HTTP	/	traffic-port	30 seconds

Timeout	Healthy threshold	Unhealthy threshold	Success codes
5 seconds	5	2	200

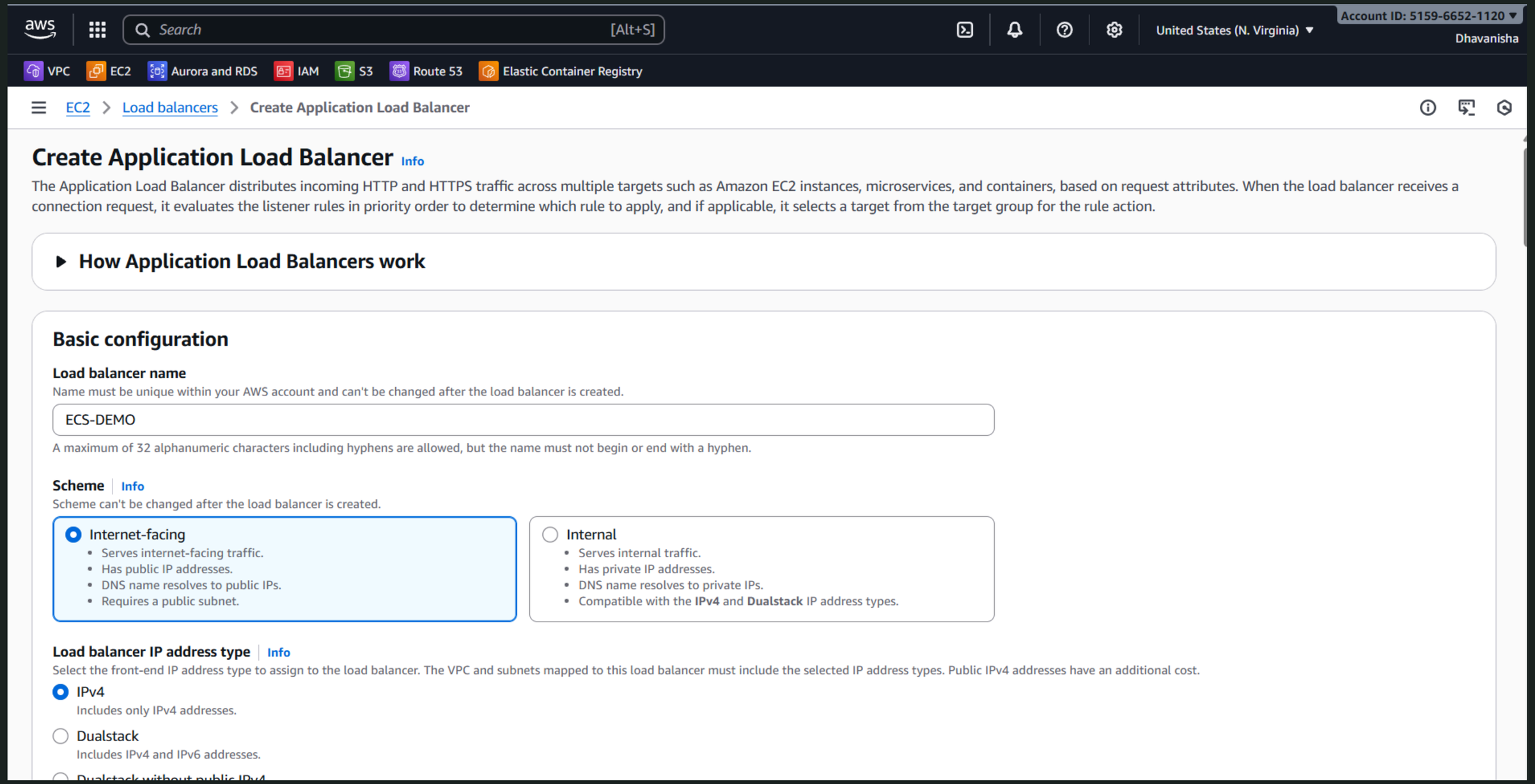
Step 2: Register targets

Edit

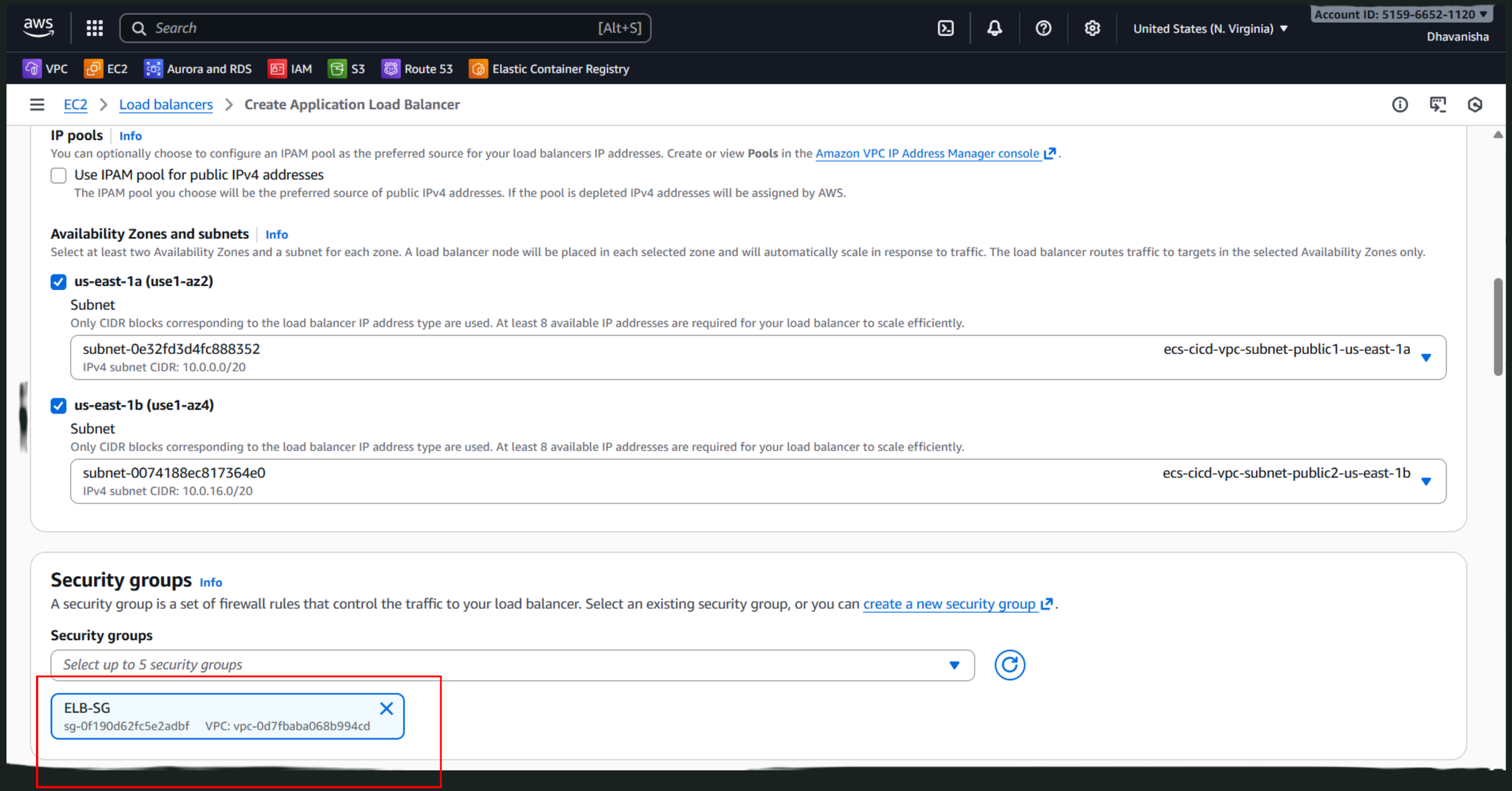
Targets (0)


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
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?
Actions
Create target group
Filter target groups
1
Name
ARN
Port
Protocol
Target type
Load balancer
VPC ID
ECS-TG-NODE
arn:aws:elasticloadbalancin...
80
HTTP
IP
None associated
vpc-0d7f
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

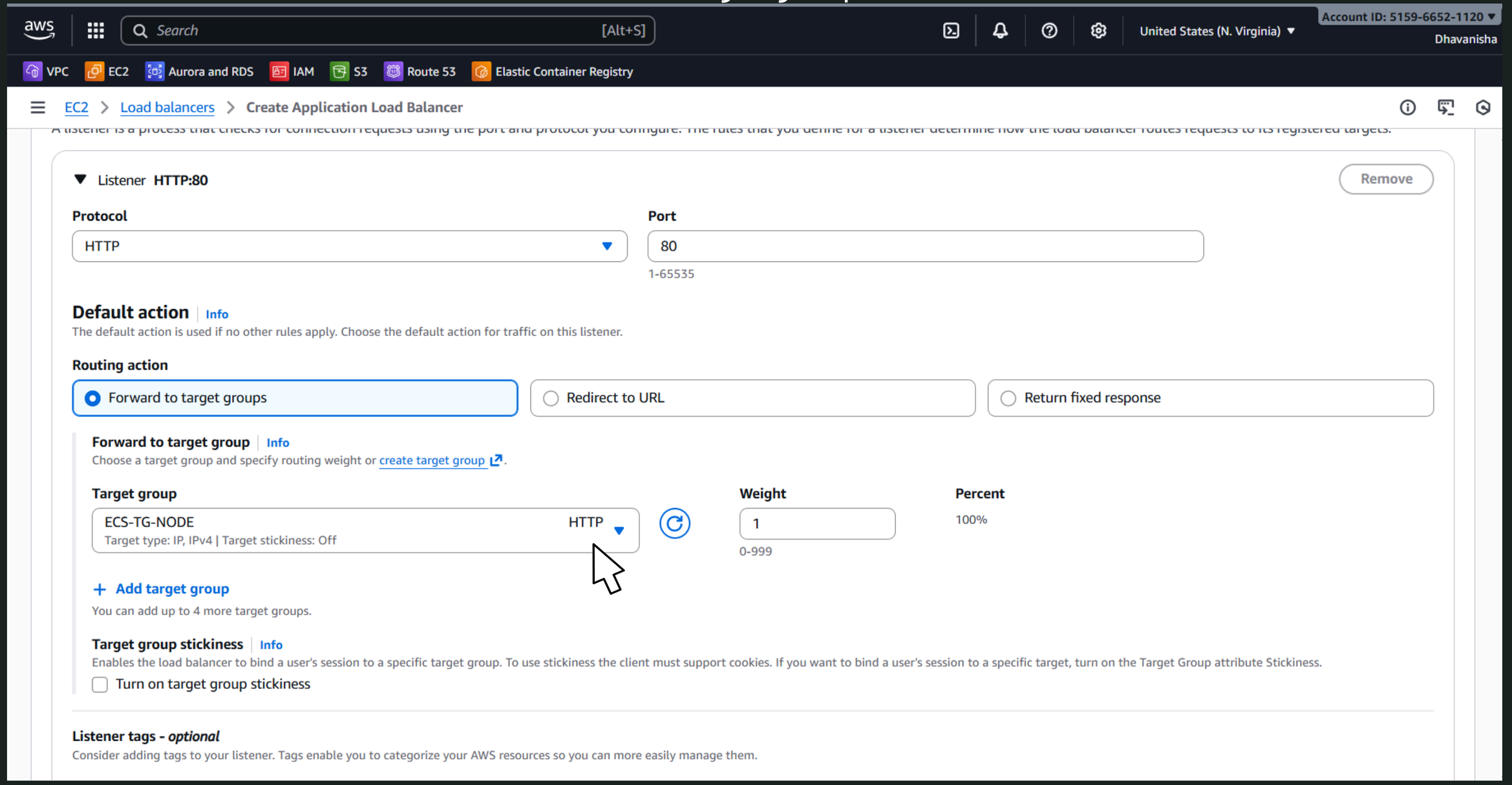
Create an internet-facing load balancer.



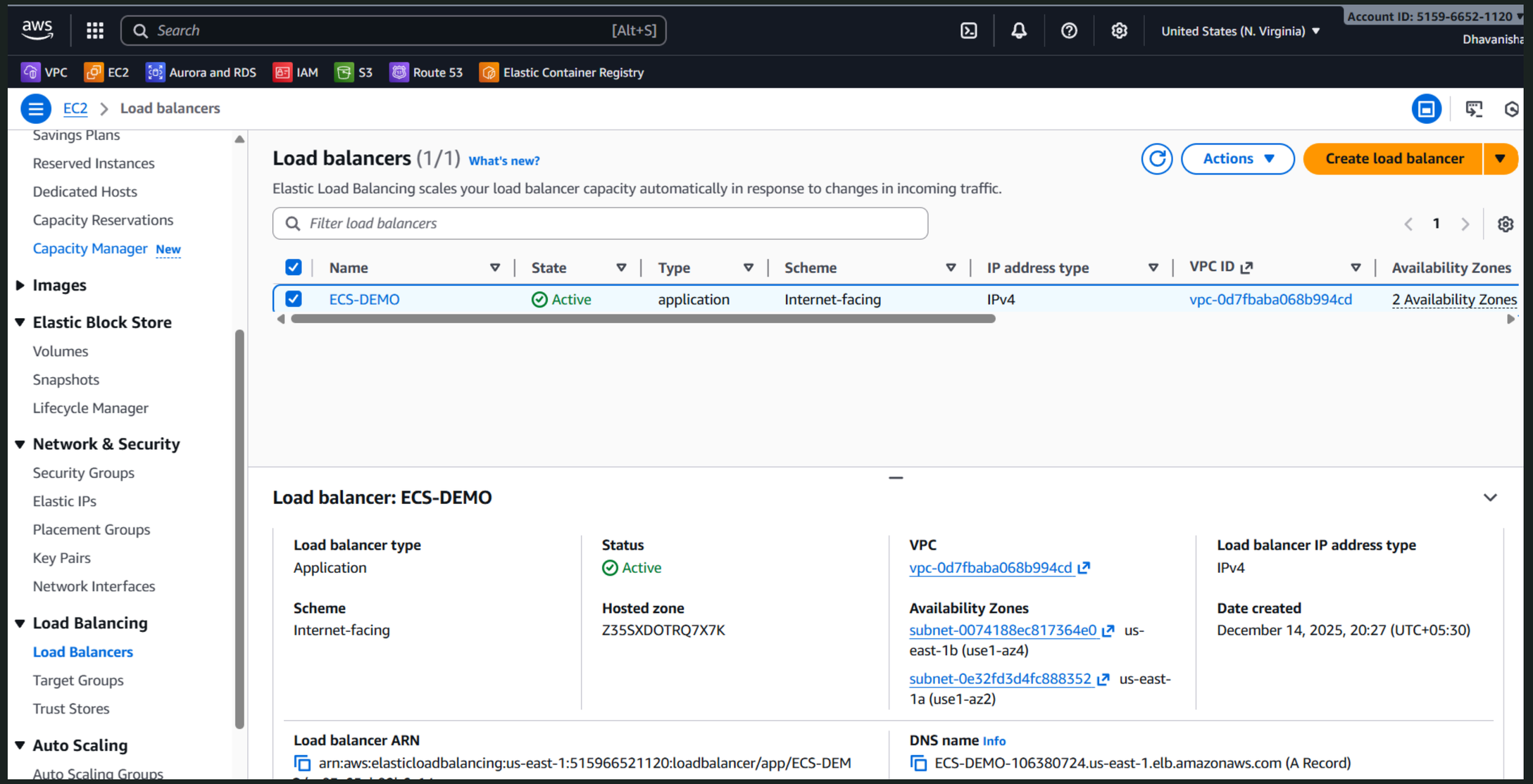
Select the security group created for ELB



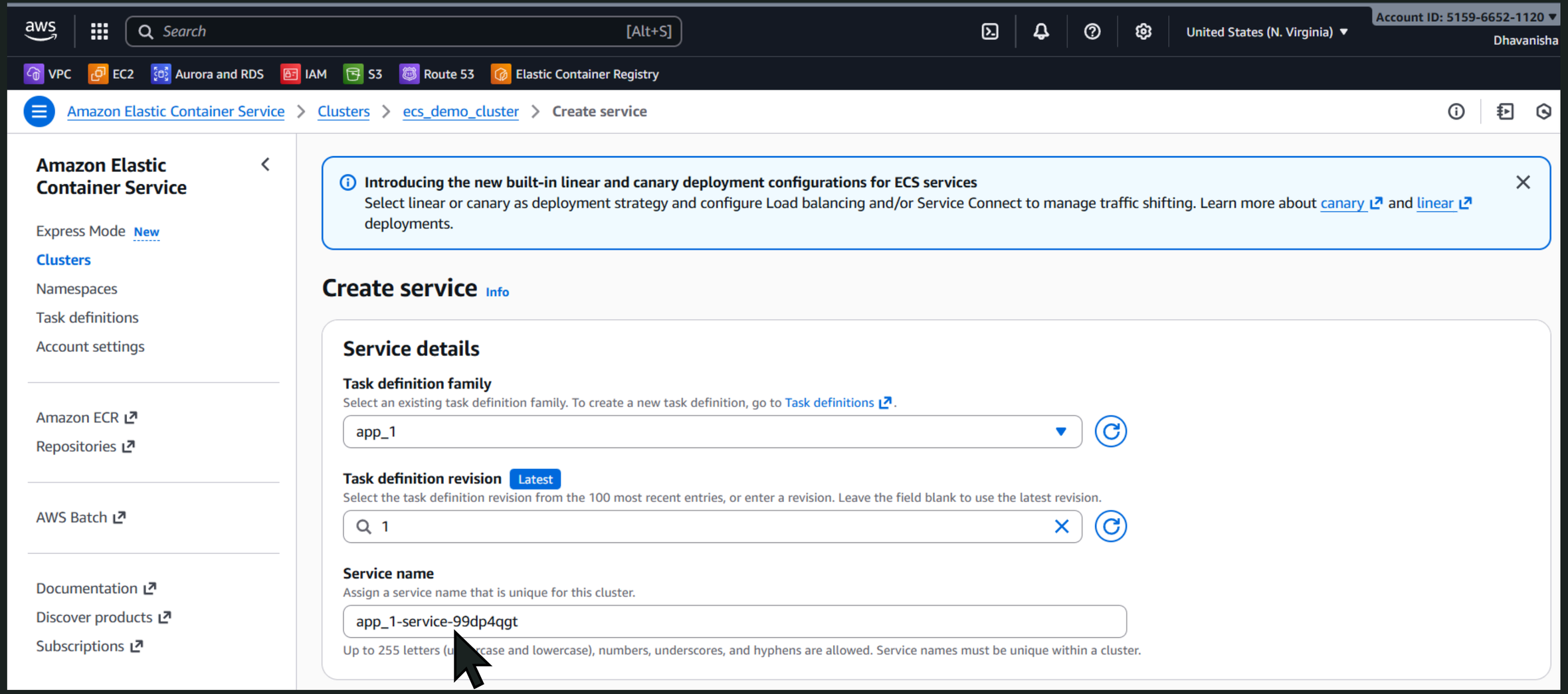
Select theTarget group.



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.



Create a service and Use a unique service name.



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.

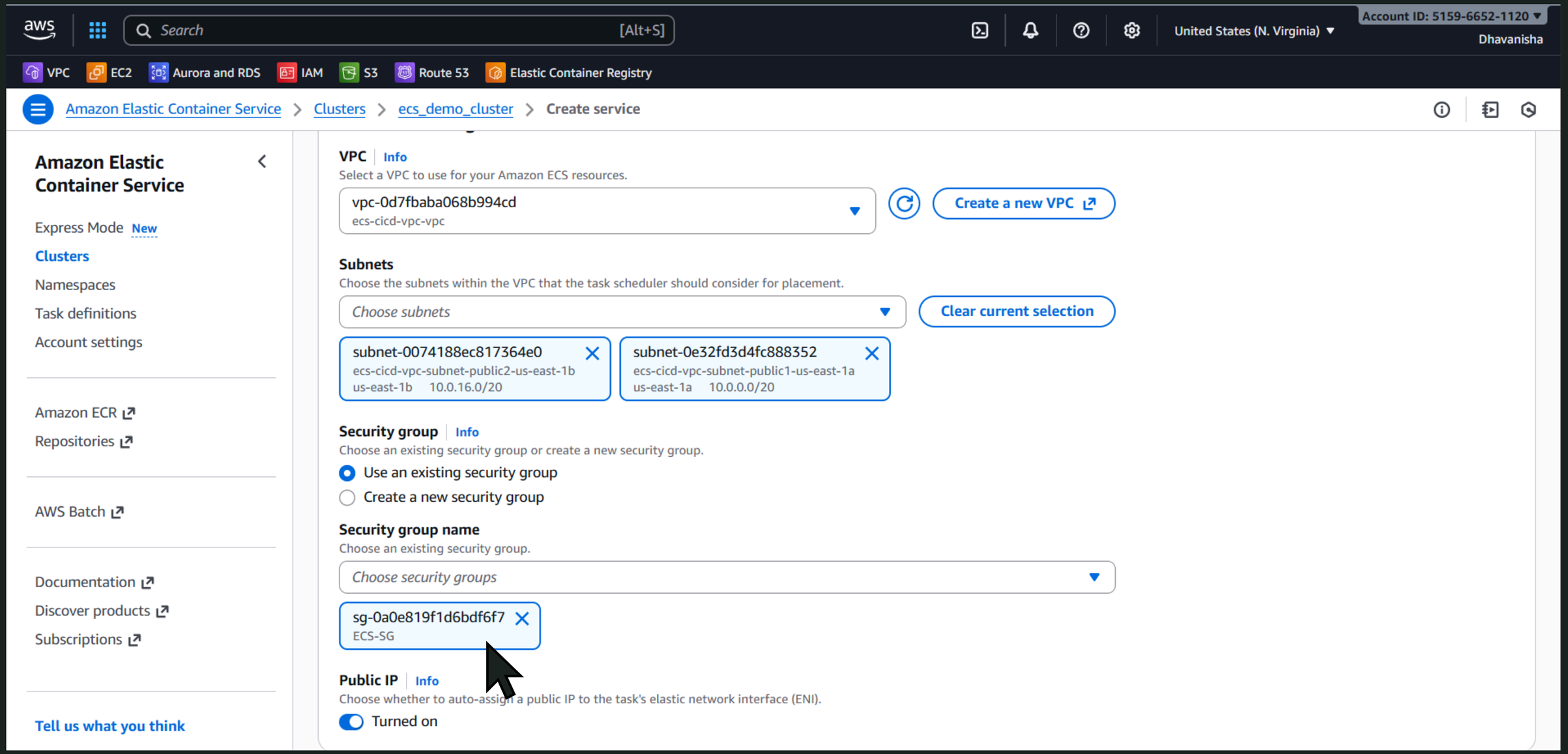
Desired tasks

Specify the number of tasks to launch.

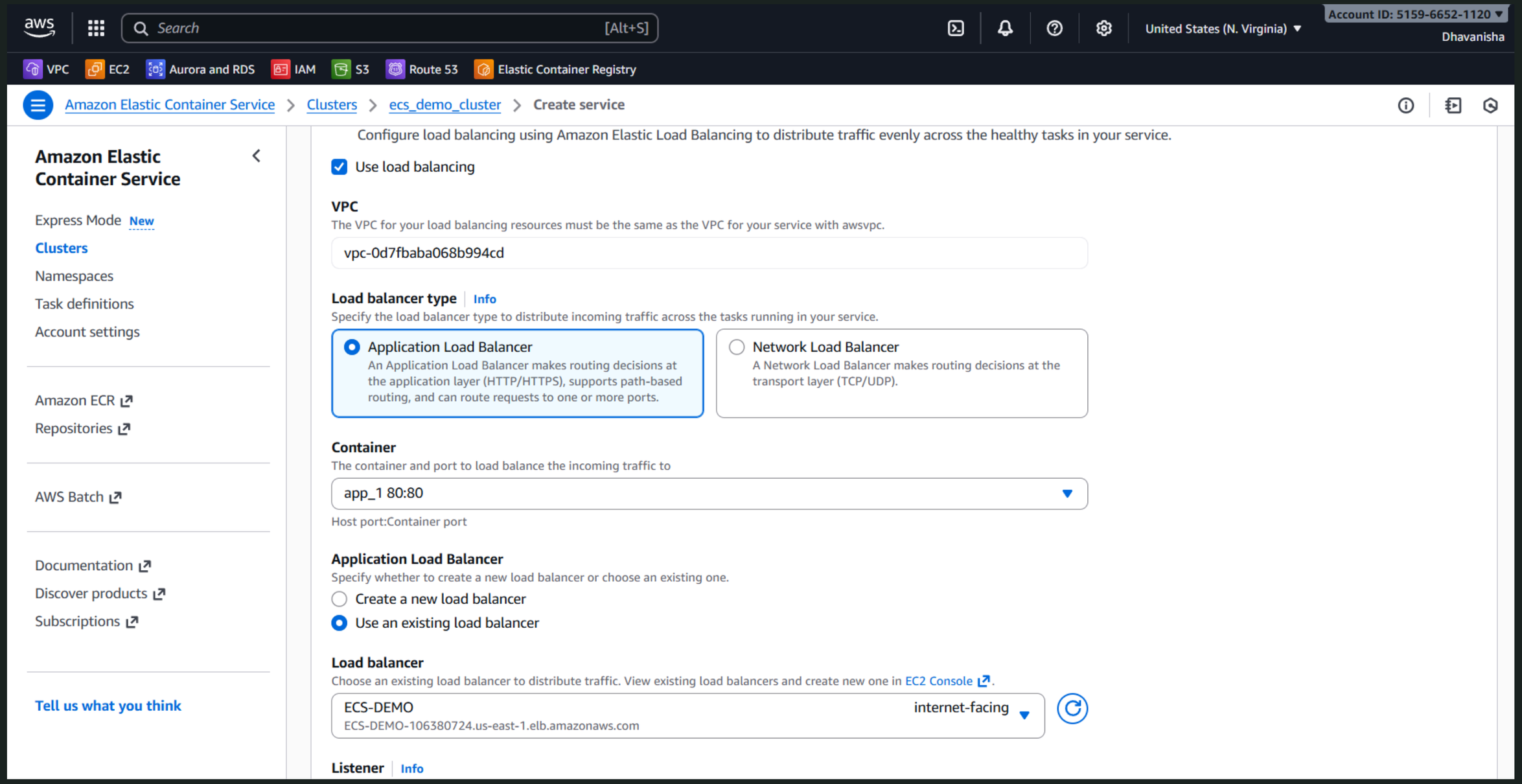
2

Availability Zone rebalancing Info

Select the security group created for ECS.



Attached the existing Application Load Balancer using port 80



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

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

- Create new listener
- Use an existing listener

Listener

HTTP:80

Listener rules for 80:HTTP (1)

Traffic received by the listener is routed according to its rules. Rules are evaluated in priority order, from the lowest value to the highest value. The default rule is evaluated last.

1

Priority	Rule path	Target group
default	/	ECS-TG-NODE

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

ECS-TG-NODE

Health check path

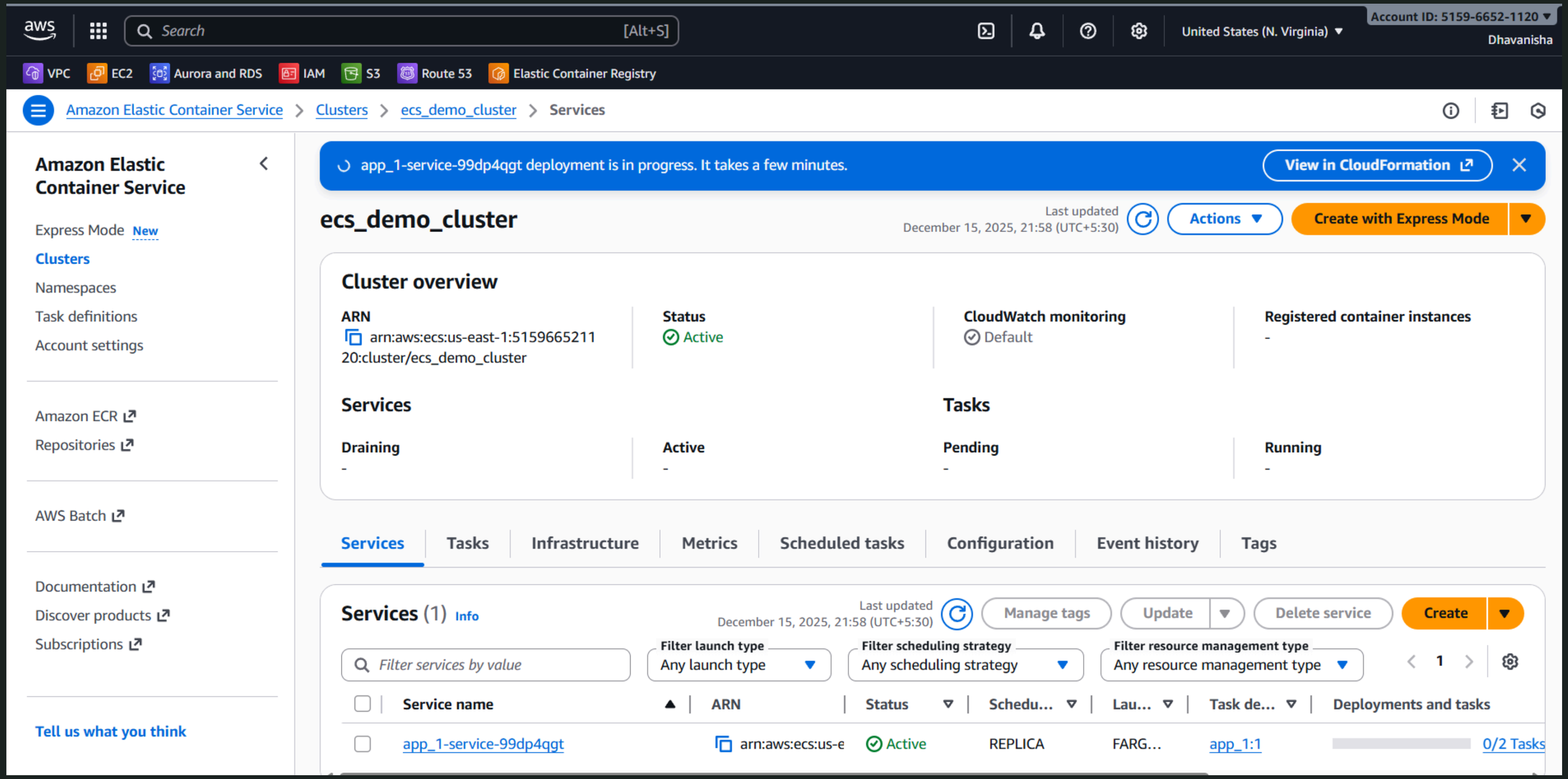
/

Health check protocol Info

HTTP

VPC Lattice - *optional* Info

Services has been Created for application 1.



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 has been deployed successfully. View service

app_1-service-99dp4qgt Info

Last updated December 15, 2025, 22:00 (UTC+5:30) | 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	Tasks (2 Desired)	Task definition: revision	Deployment status
Active	0 Pending | 2 Running	app_1:1	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

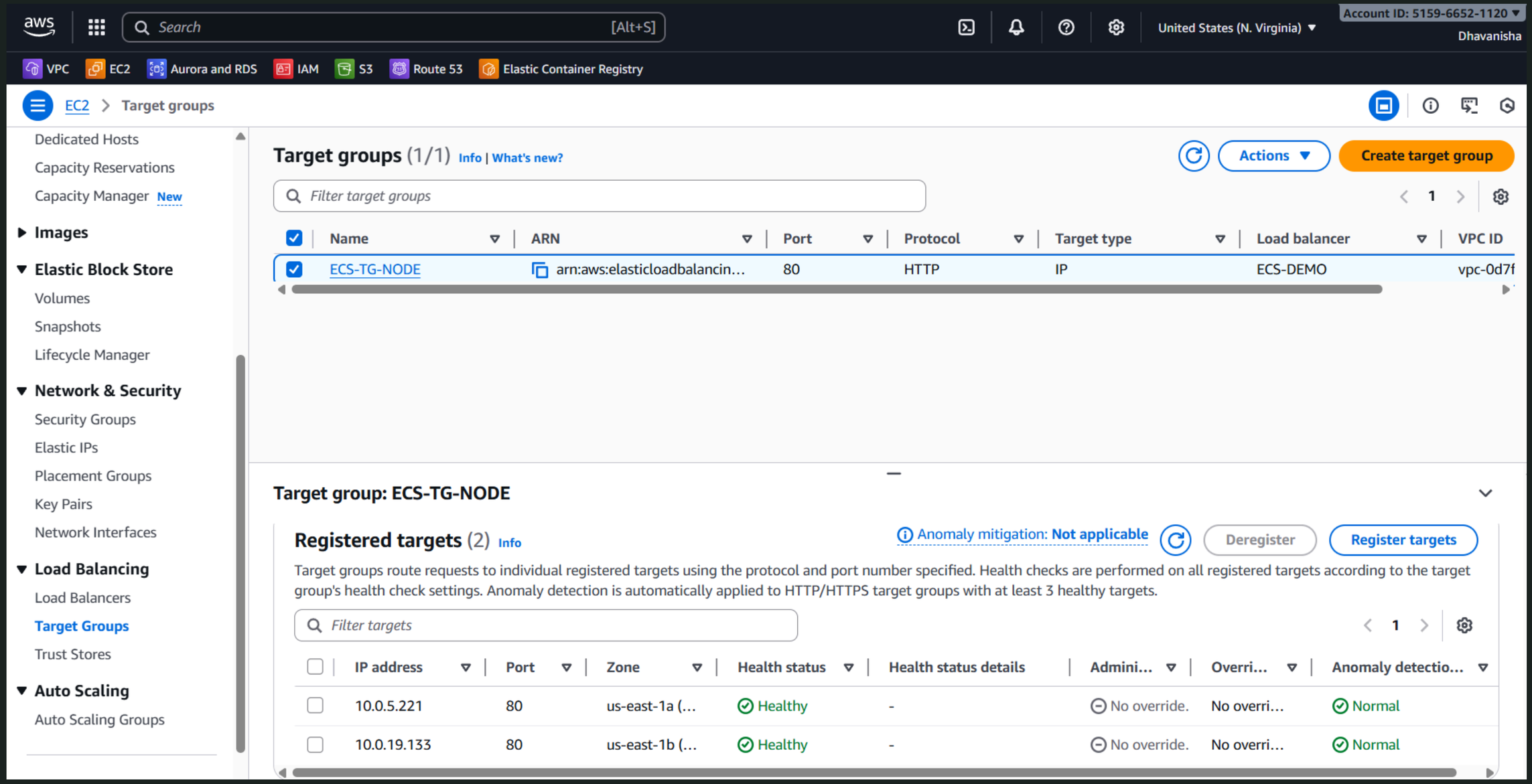
Deployments current state
2 Completed tasks

Created at
December 15, 2025, 21:58 (UTC+5:30)

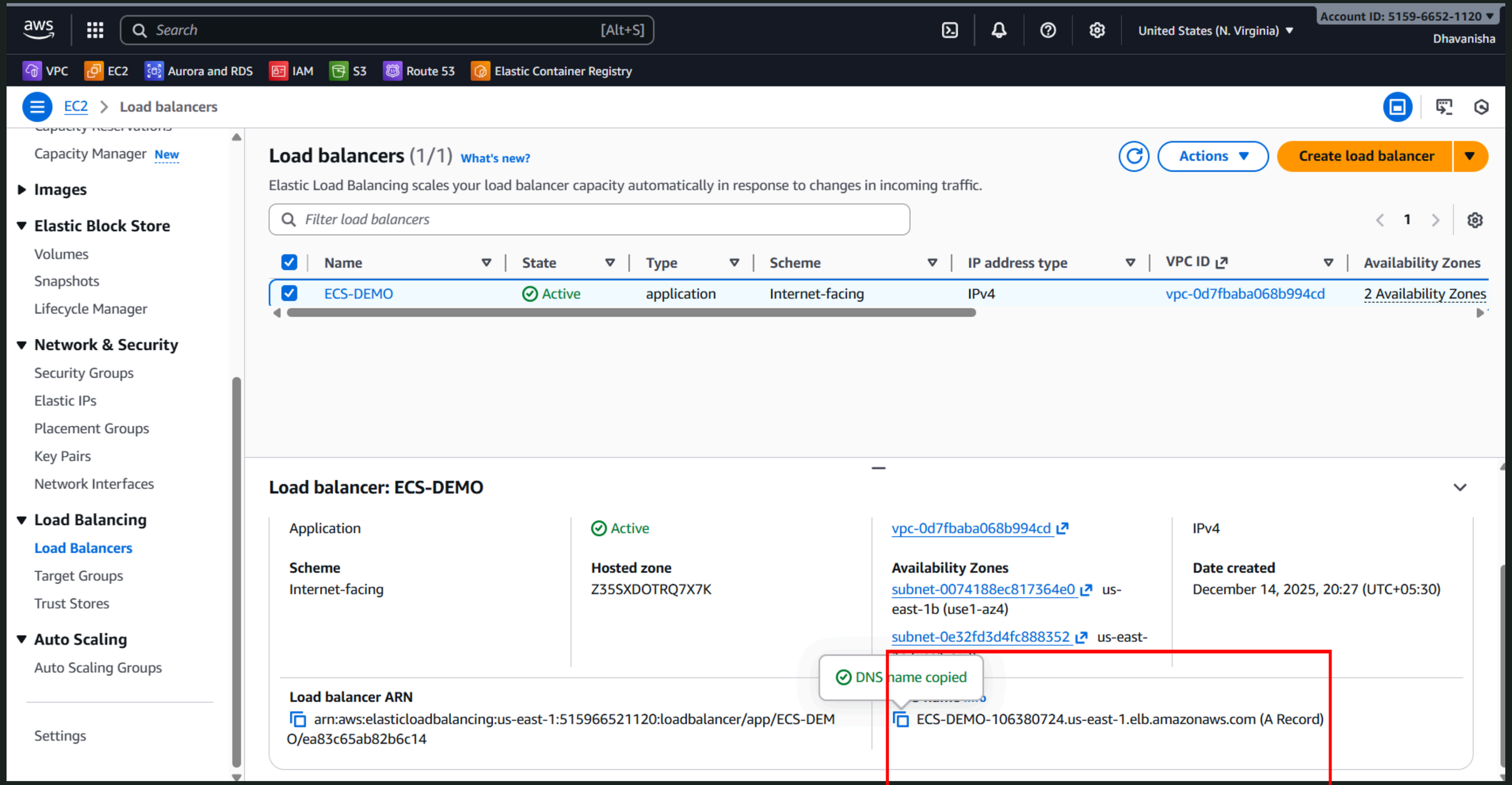
Health check grace period
0 seconds

▼ Load balancer health

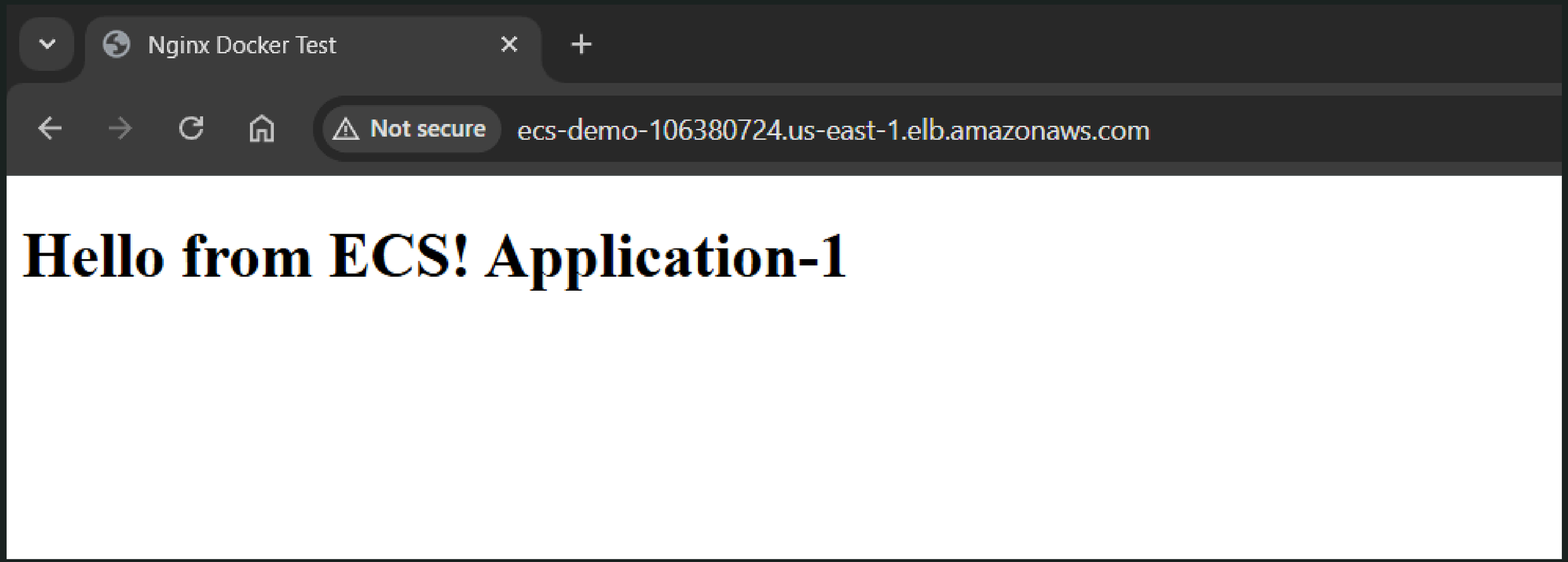
The target group shows all registered targets as healthy and ready to receive traffic on port 80.



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



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



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

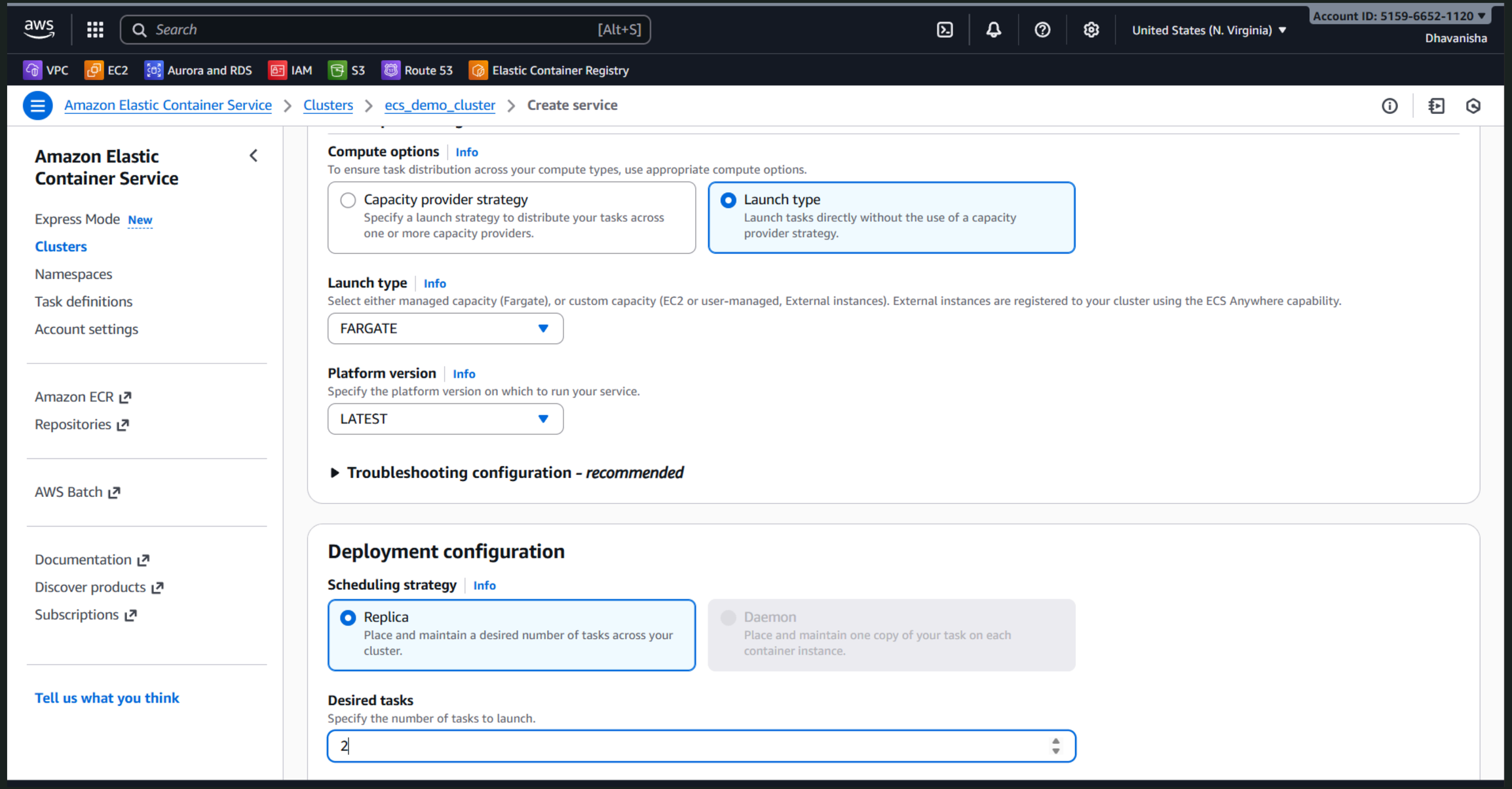
Up to 255 letters (uppercase and lowercase), numbers, underscores, and hyphens are allowed. Service names must be unique within a cluster.

Environment

AWS Fargate

Existing cluster

ecs_demo_cluster


aws
Search
[Alt+S]
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

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

subnet-0e32fd3d4fc888352
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

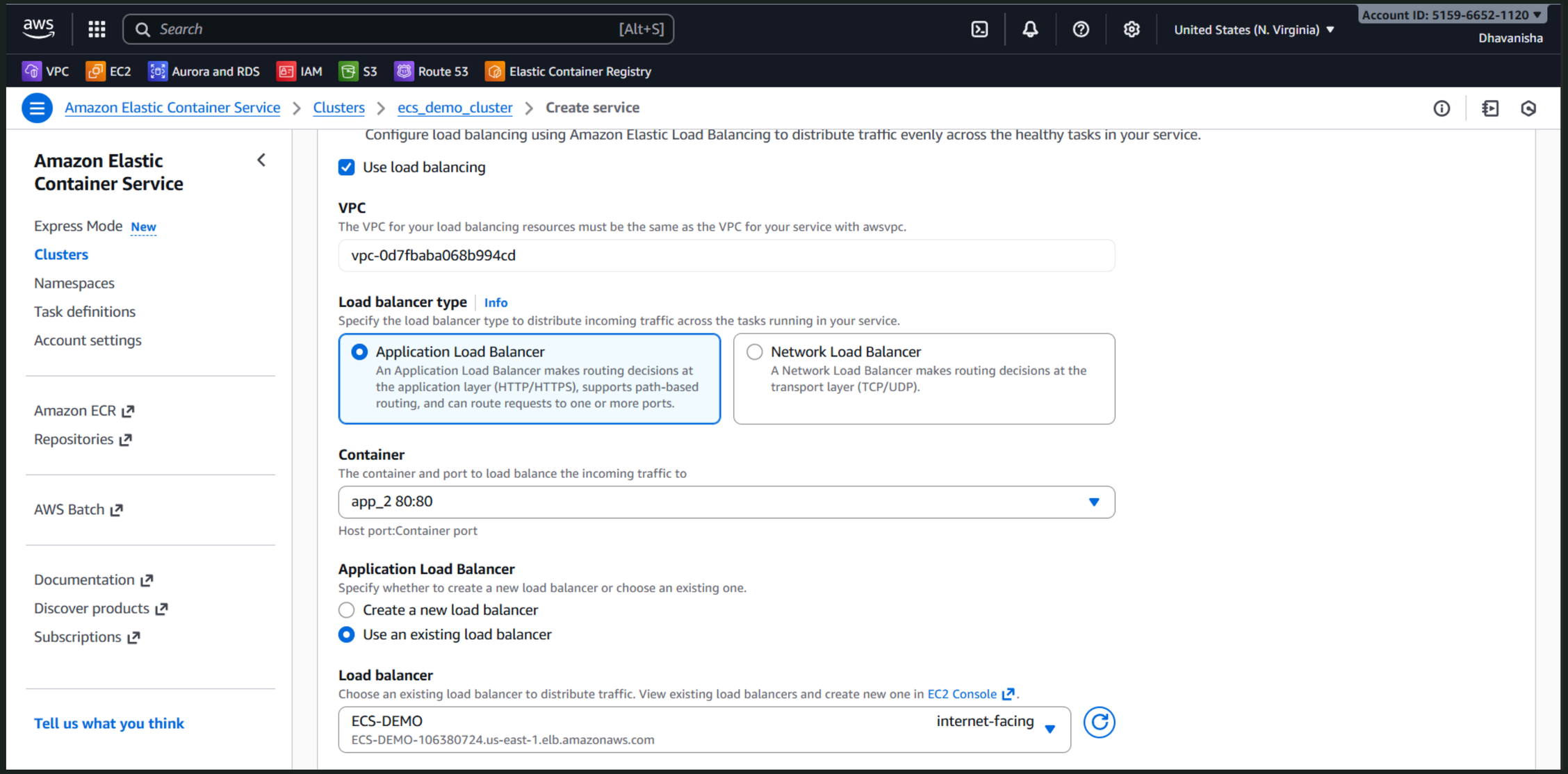
sg-0a0e819f1d6bdf6f7
ECS-SG

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.



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

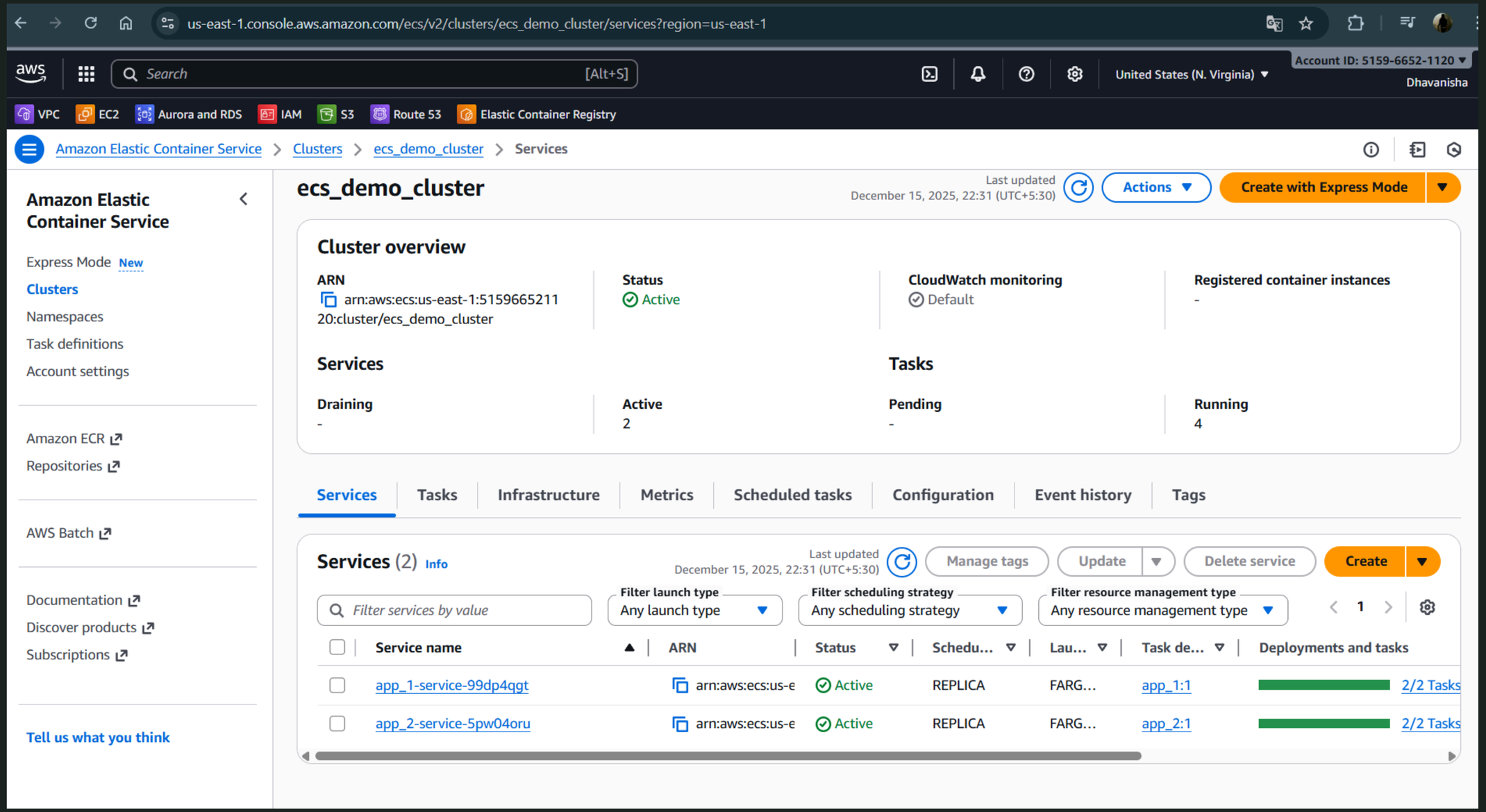
seconds

Health check protocol Info

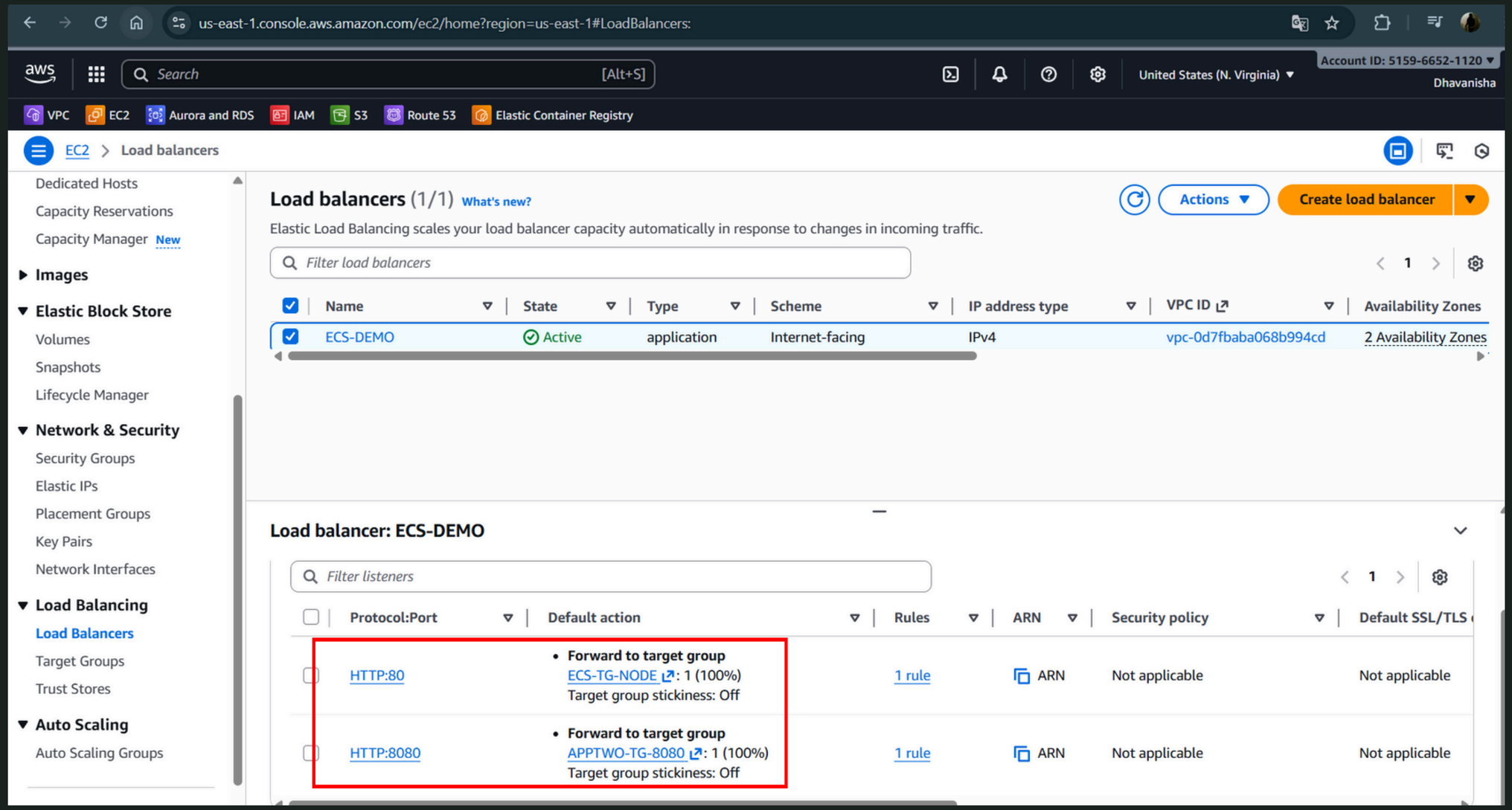
HTTP

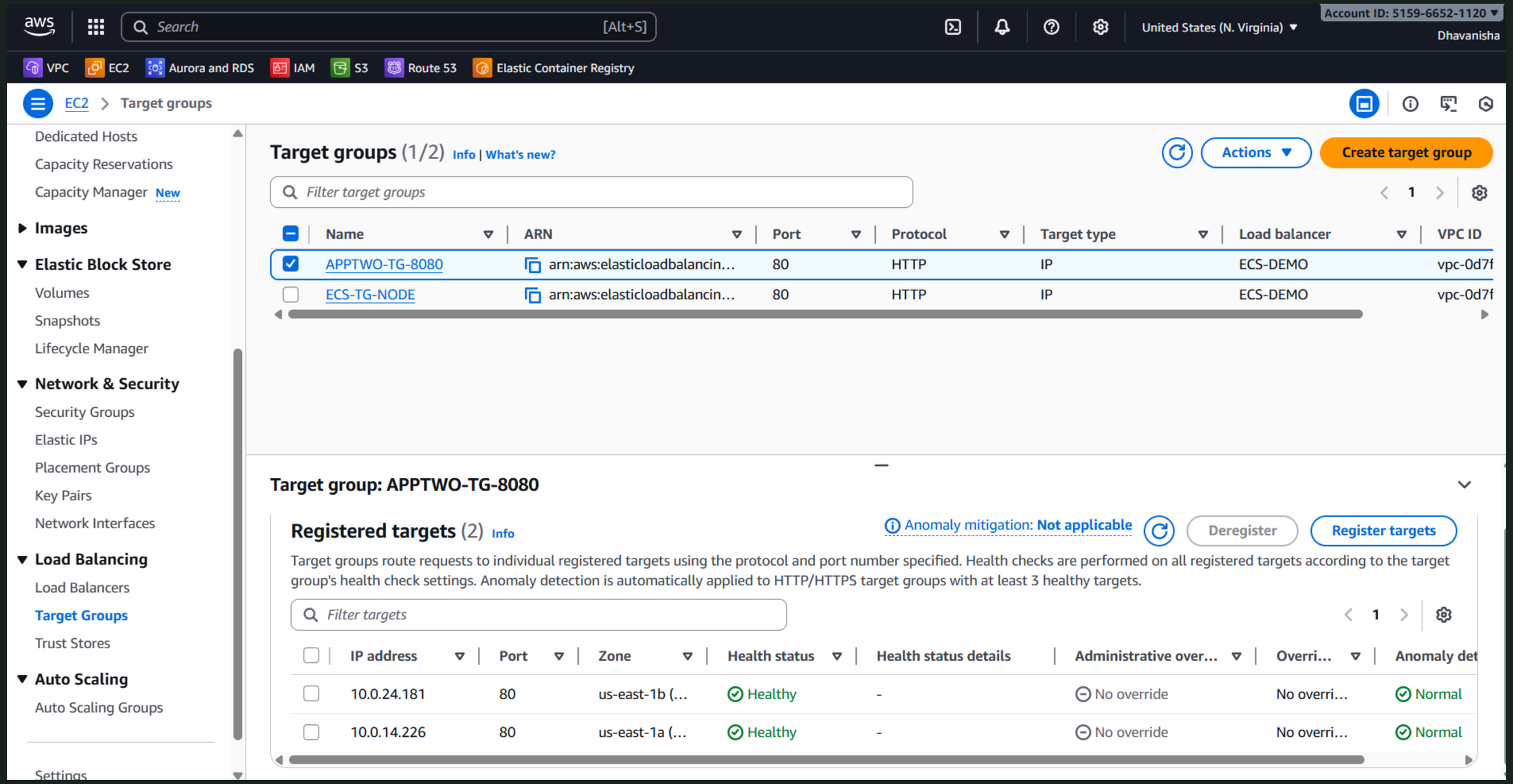
Health check path Info

Two services are created in the cluster.



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




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
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
Settings
Target groups (1/2) Info | What's new?
Actions
Create target group
Filter target groups
1
Name
ARN
Port
Protocol
Target type
Load balancer
VPC ID
APPTWO-TG-8080
arn:aws:elasticloadbalancin...
80
HTTP
IP
ECS-DEMO
vpc-0d7f
ECS-TG-NODE
arn:aws:elasticloadbalancin...
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
1
IP address
Port
Zone
Health status
Health status details
Administrative over...
Overri...
Anomaly det
10.0.24.181
80
us-east-1b (...
Healthy
-
No override
No overri...
Normal
10.0.14.226
80
us-east-1a (...
Healthy
-
No override
No overri...
Normal

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

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	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-1:515966521120:service/ecs_demo_cluster/app_2-service-5pw04oru	2 Completed tasks	December 15, 2025, 22:30 (UTC+5:30)

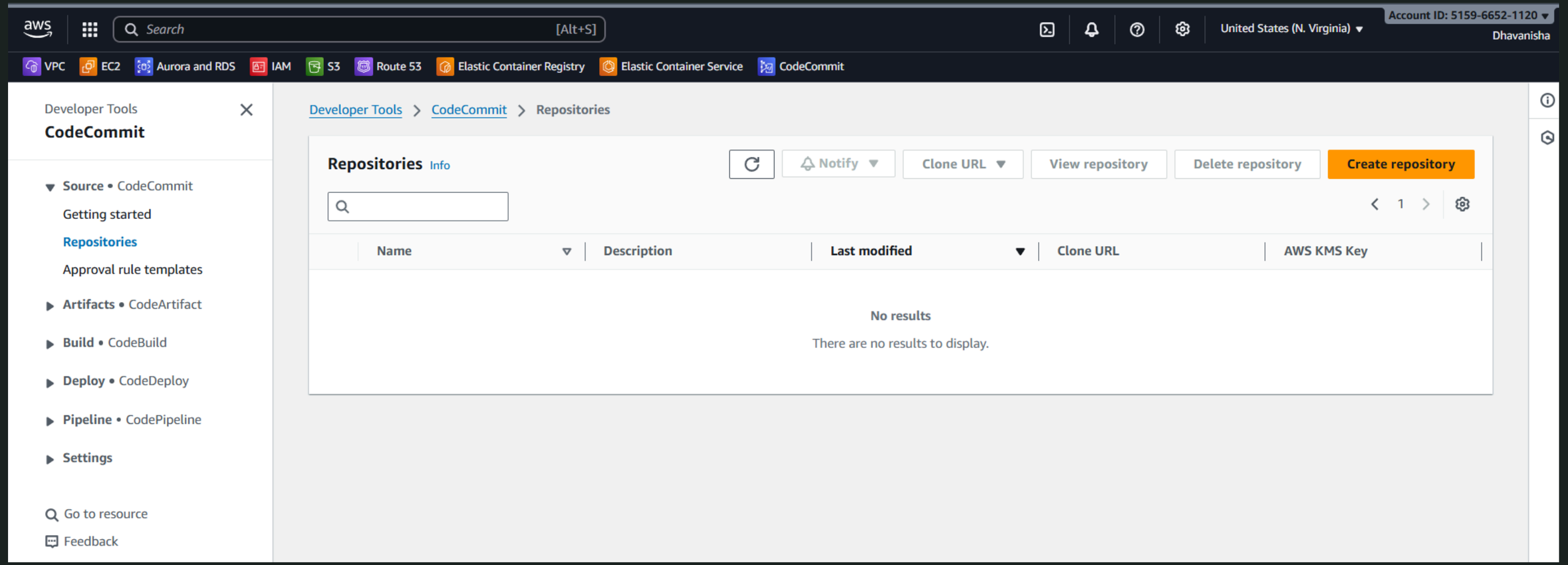
Health check grace period
0 seconds

Load balancer health

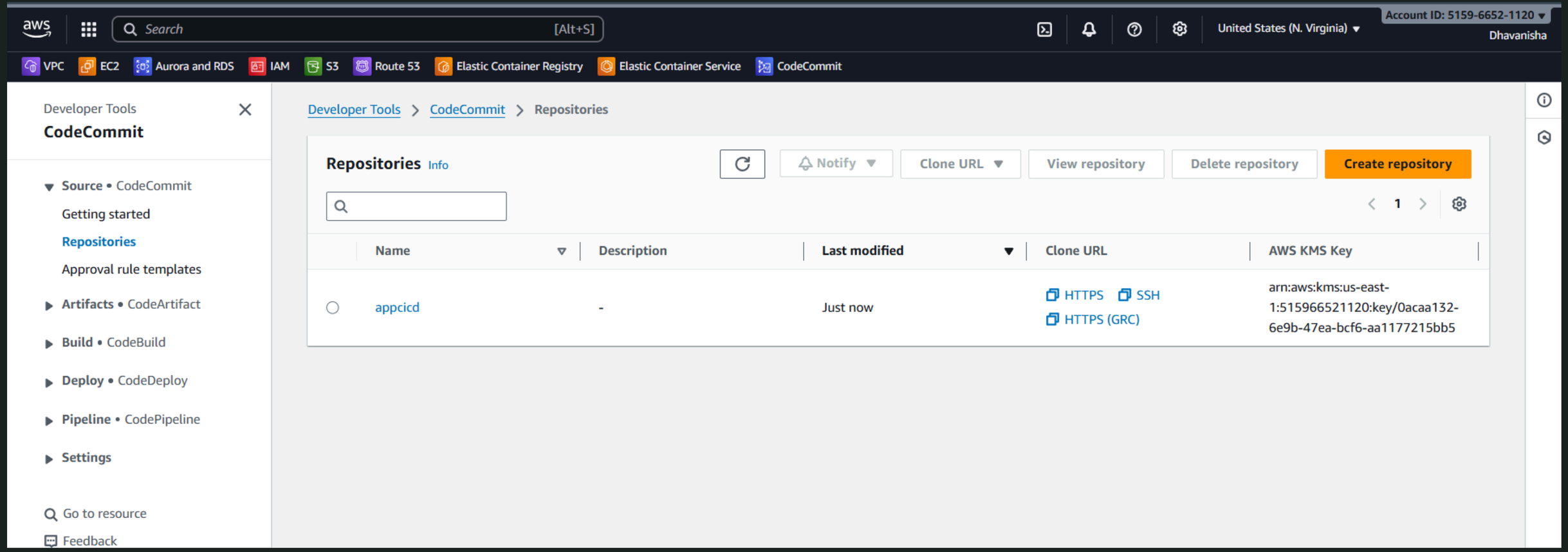
Load balancer	Load balancer type	Container name:port	Listeners	Target group	Target health
ECS-DEMO	Application Load Bal…	app_2:80	HTTP:8080	APPTWO-TG-8080 |…	2 Healthy 0 Unhealthy

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

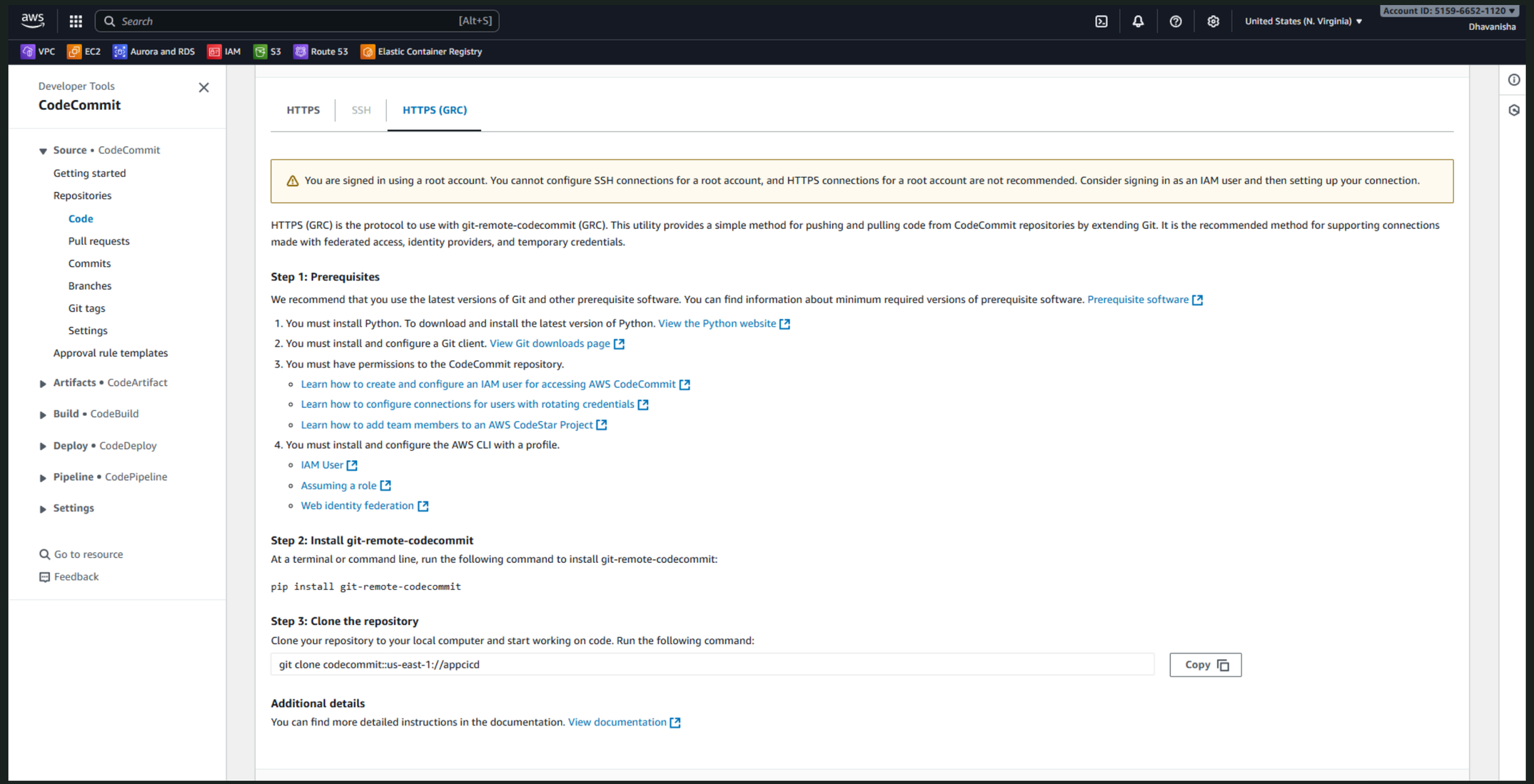
The CI/CD automation process was initiated by setting up a source code repository in AWS CodeCommit.



Repository was created in CodeCommit.



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



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 envir
onment instead: https://pip.pypa.io/warnings/venv
[root@ip-10-0-23-4 ~]#
```

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]#
```

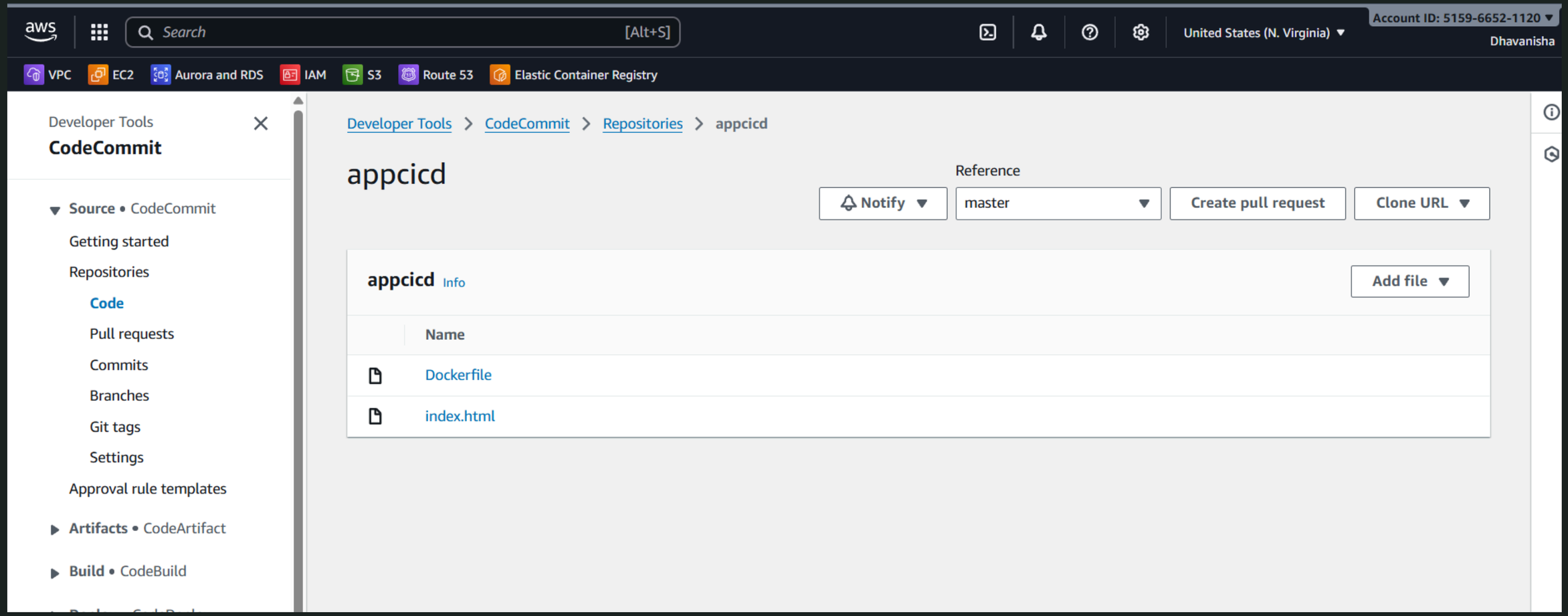
```
[root@ip-10-0-23-4 ~]# cp -r app_1/* cicddemo/appcicd/
[root@ip-10-0-23-4 ~]# cd cicddemo/appcicd/
[root@ip-10-0-23-4 appcicd]# ls
Dockerfile  index.html
[root@ip-10-0-23-4 appcicd]# git status
On branch master

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.

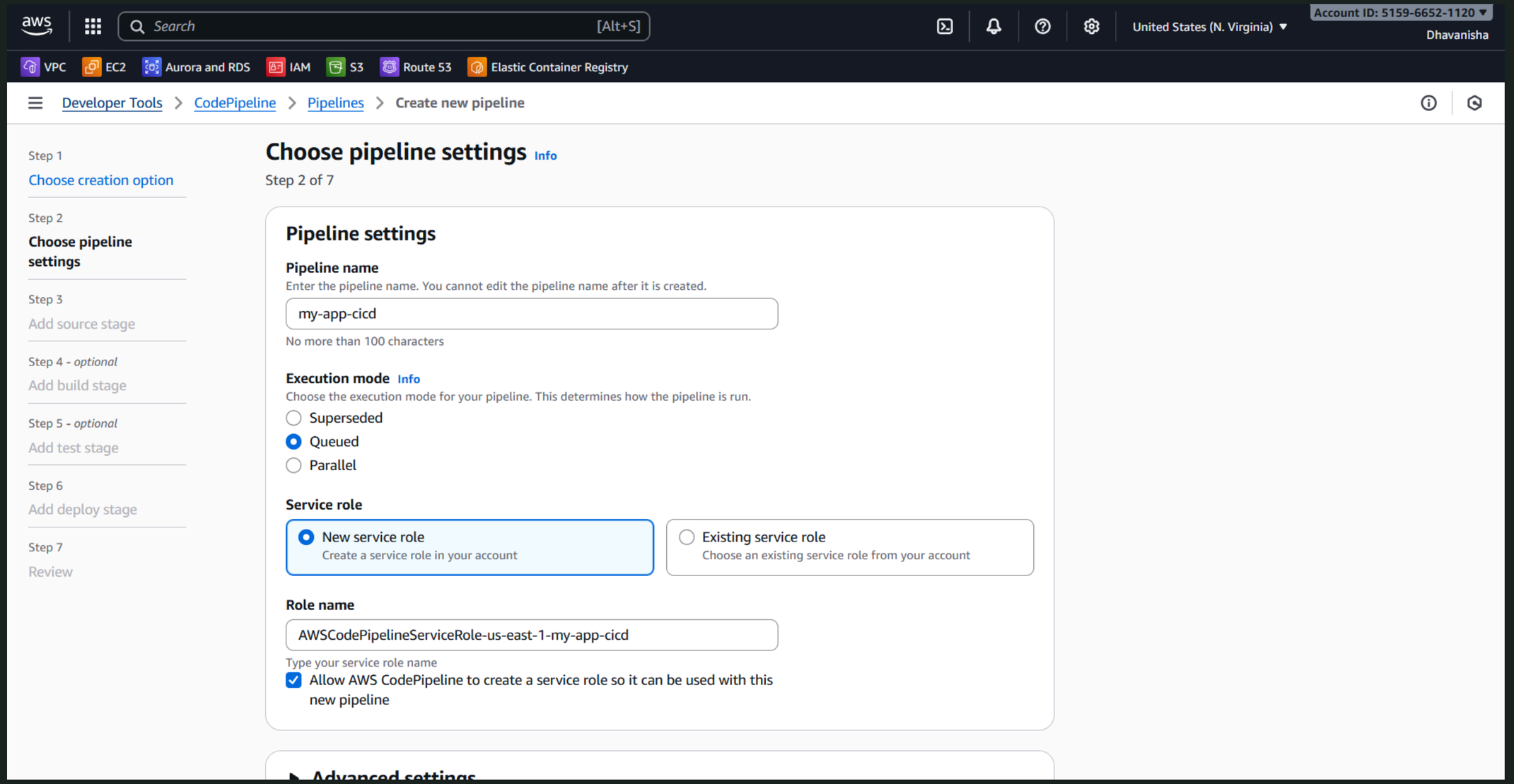
```
[root@ip-10-0-23-4 appcicd]# git status
On branch master

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.

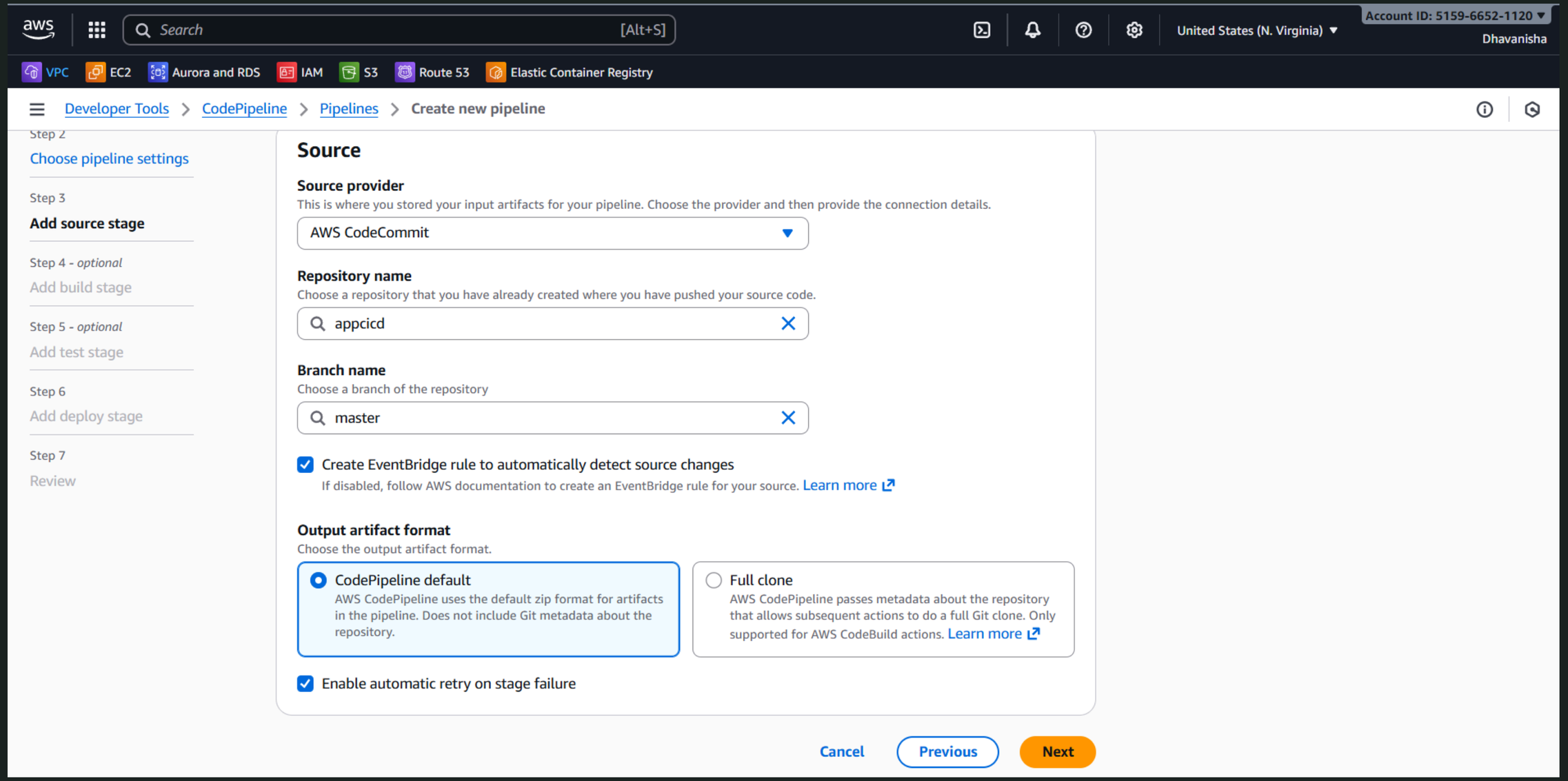
Verified that the application files were successfully uploaded to the CodeCommit repository.



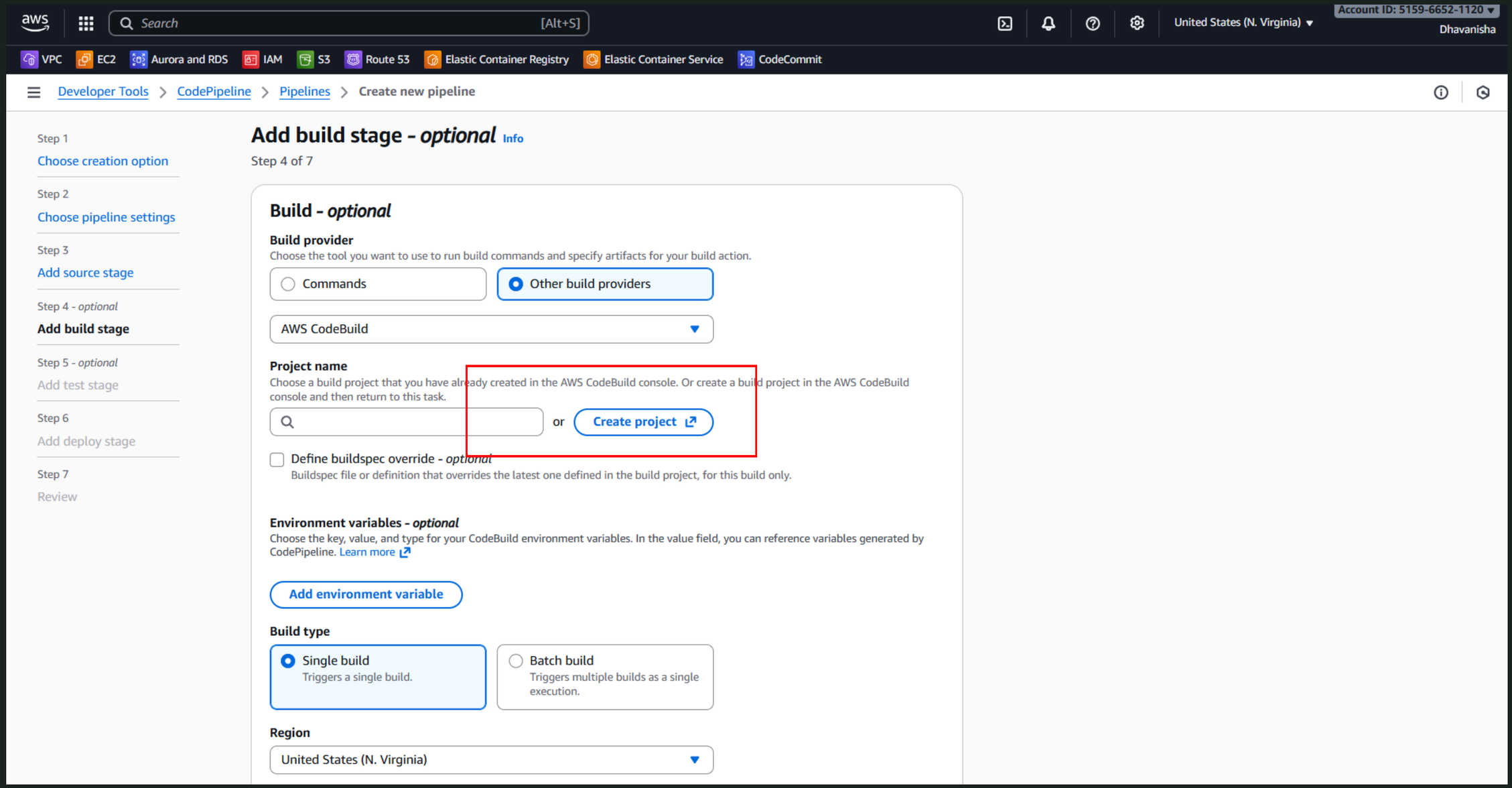
Created a new CodePipeline named my-app-cicd and configured a service role for pipeline execution.



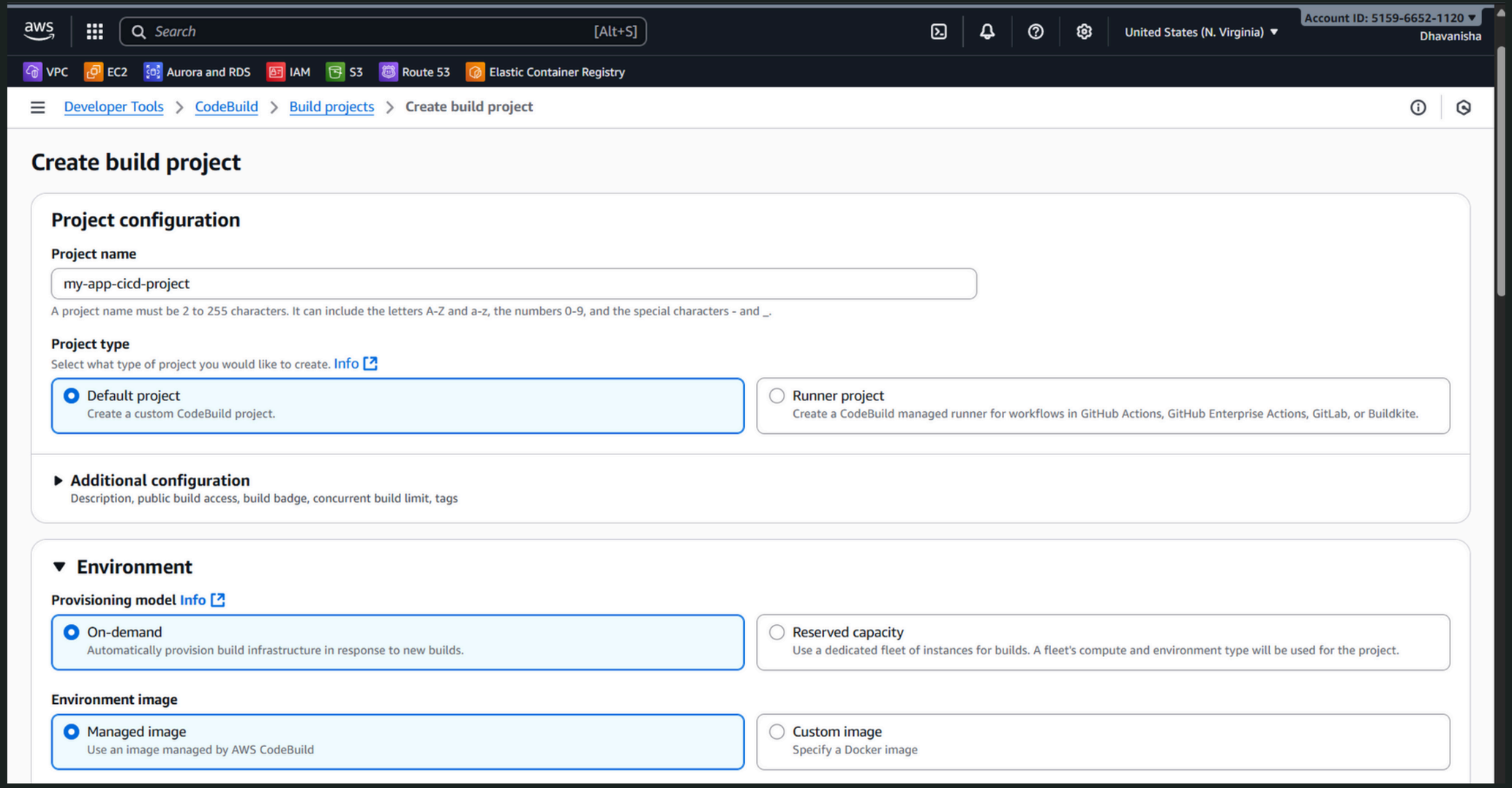
Configured the source stage by connecting CodePipeline to the CodeCommit repository on the master branch.



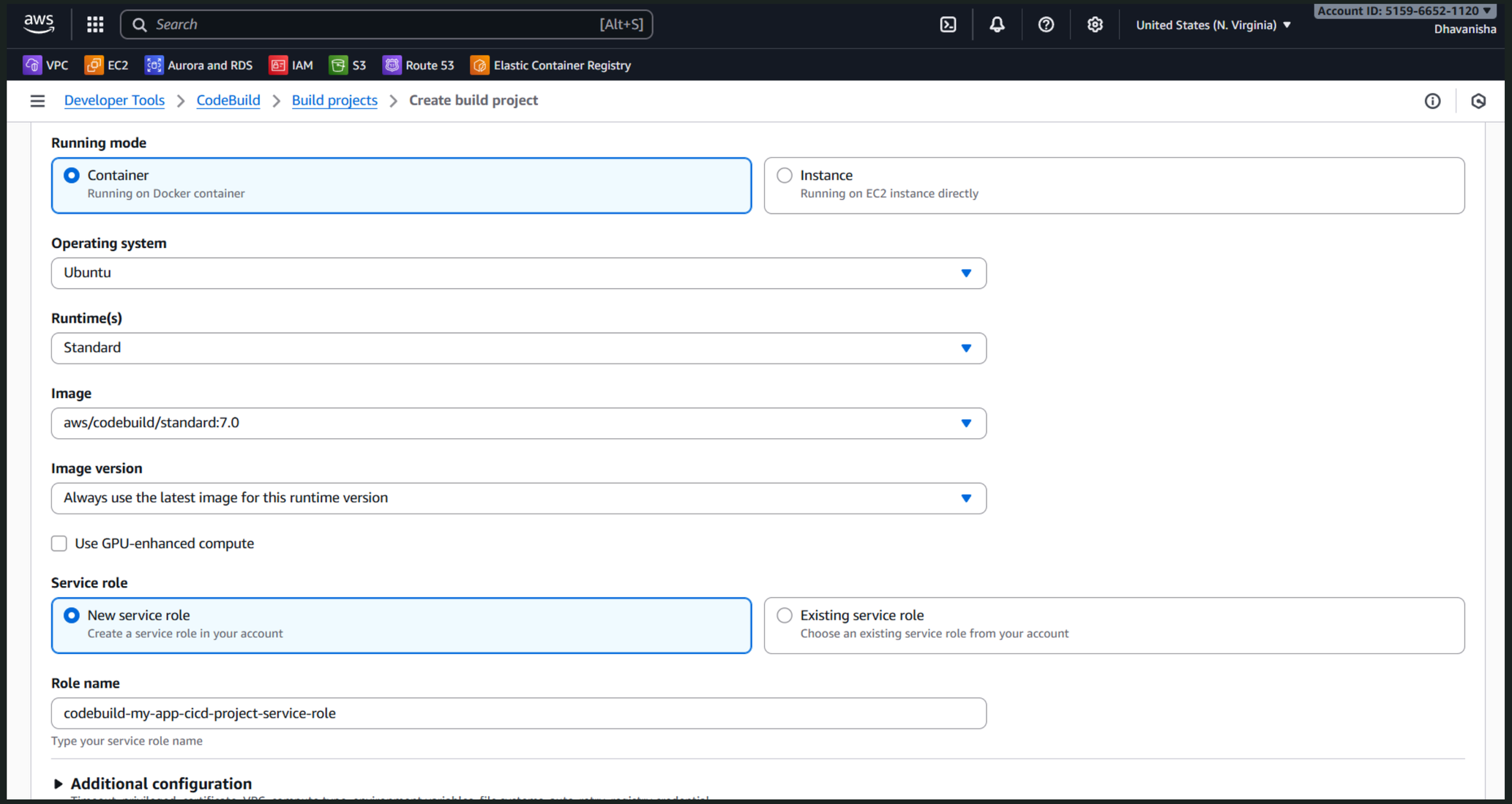
In the build stage, clicked “Create project” to create a new AWS CodeBuild project for the pipeline.



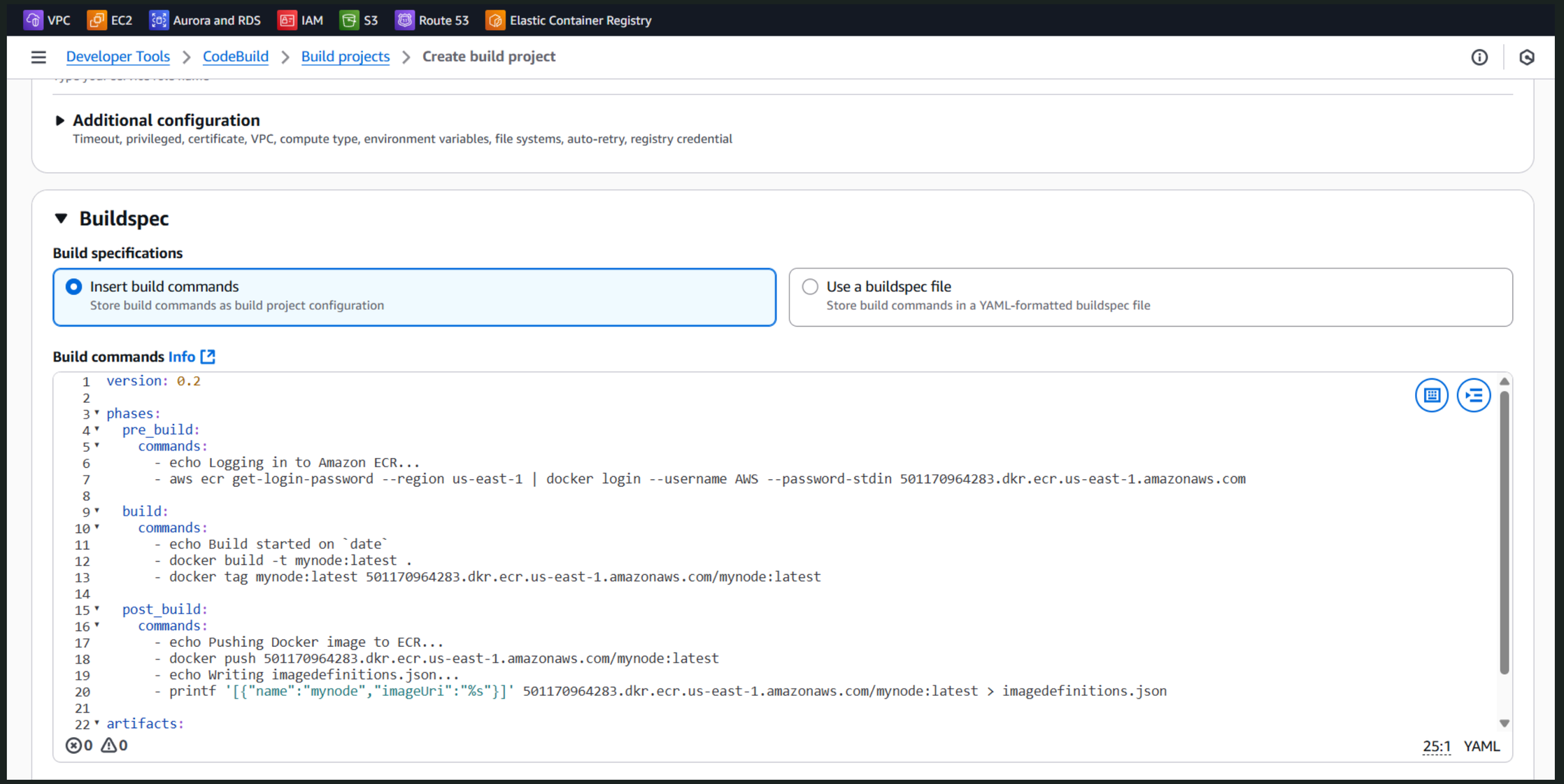
Created a new AWS CodeBuild project named my-app-cicd-project with default build settings.



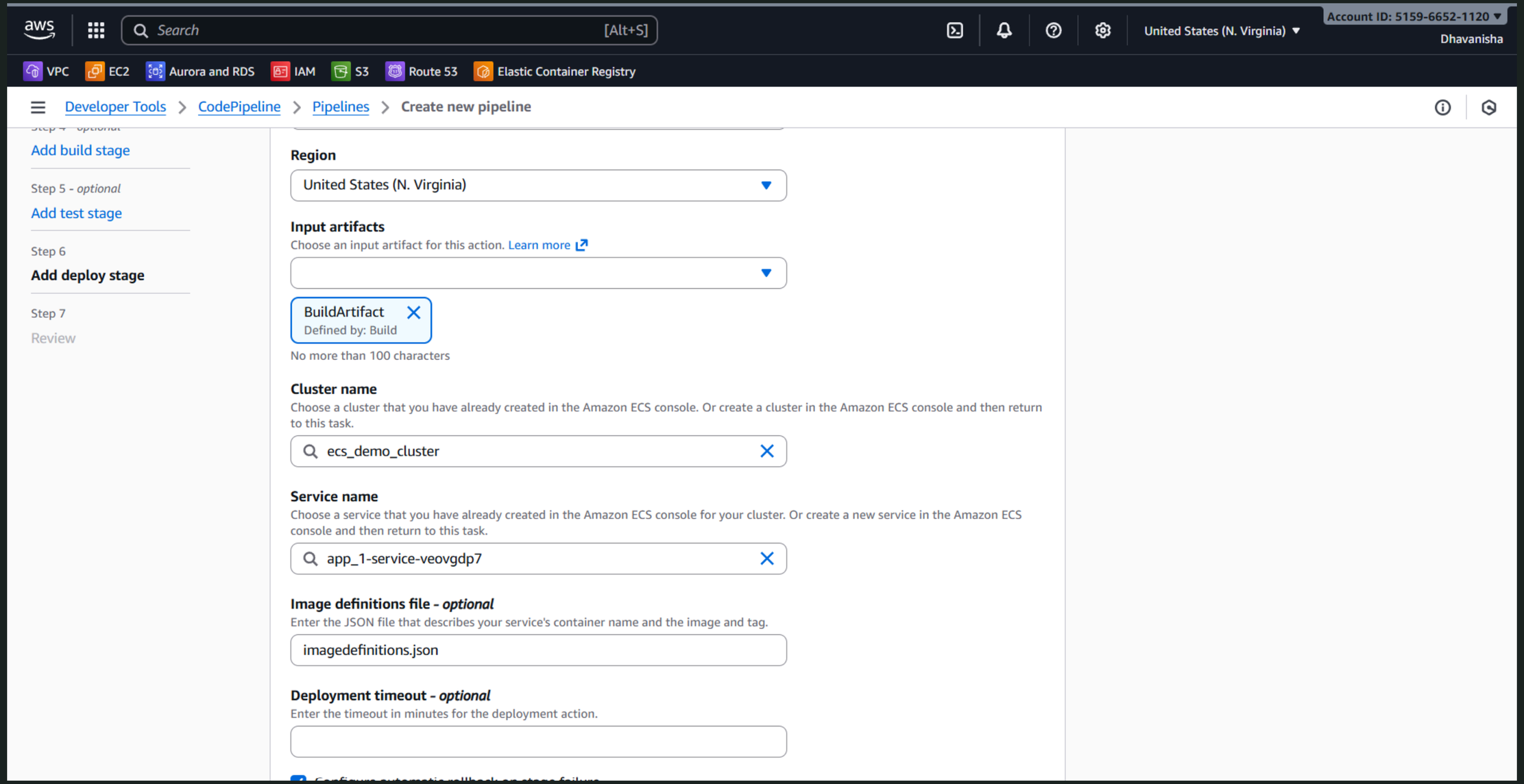
Configured the CodeBuild environment to run builds in a Docker container using the standard Ubuntu image.



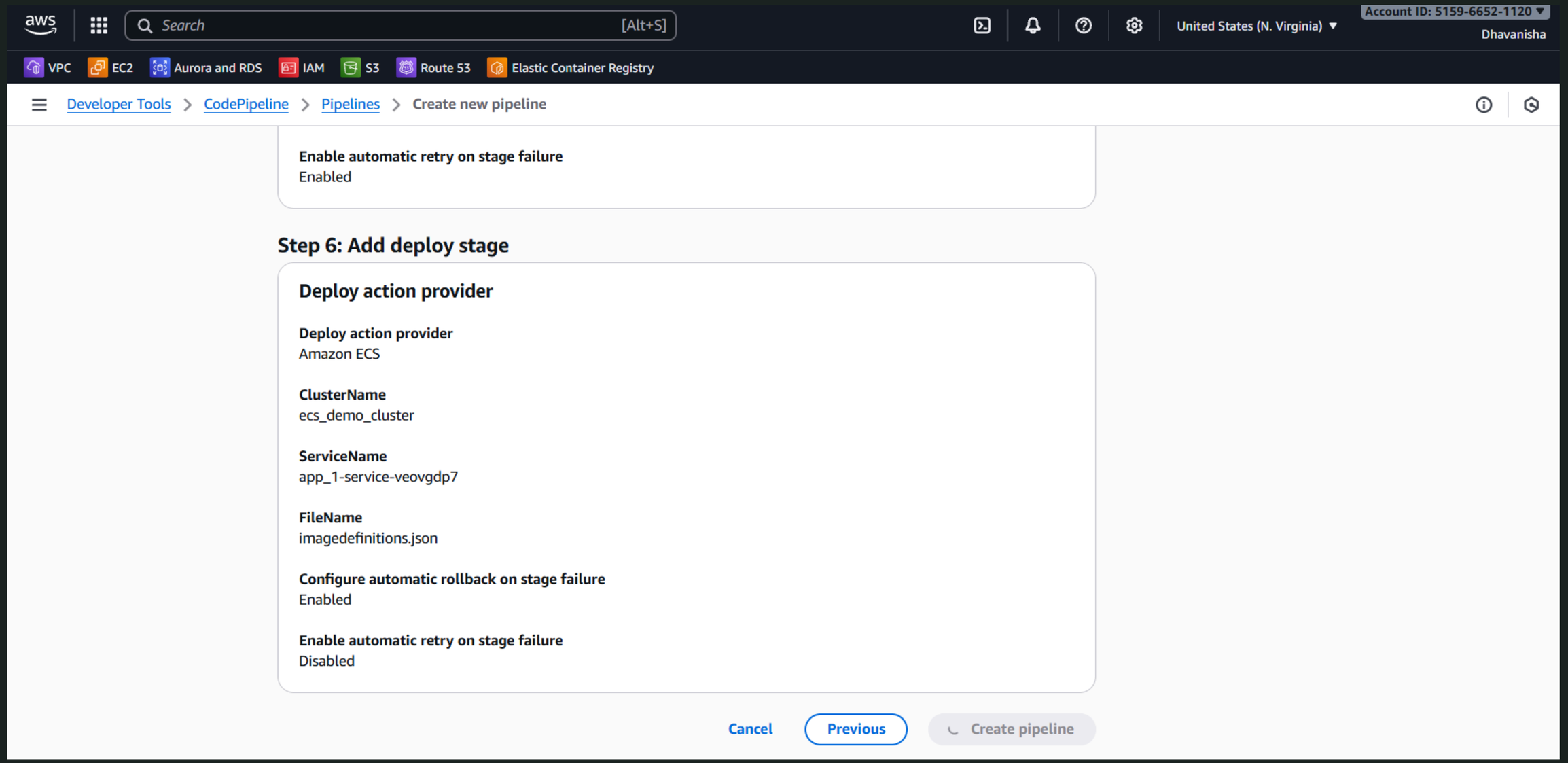
Configured the buildspec to build the Docker image and push it to Amazon ECR using CodeBuild.

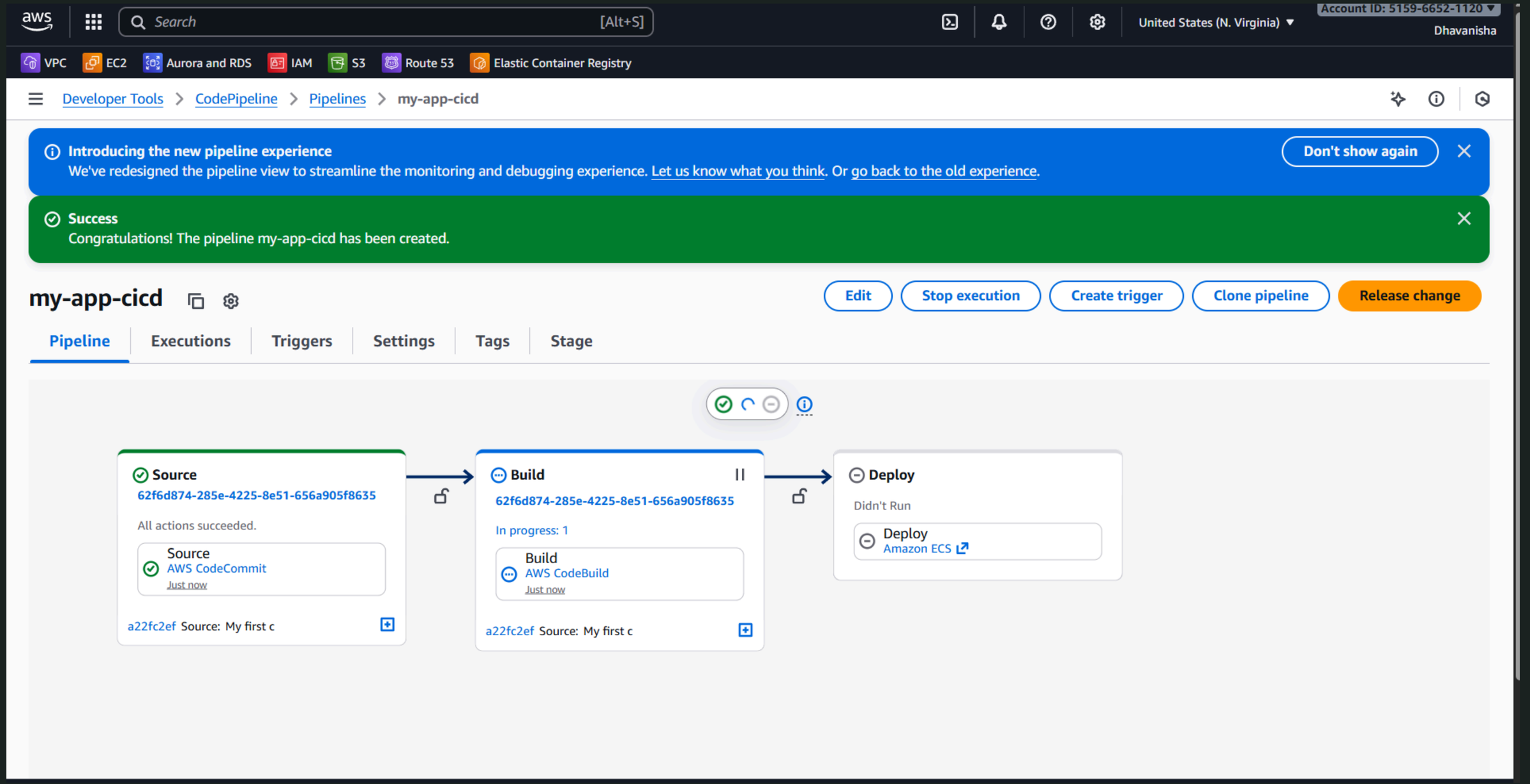


Configured the deploy stage to deploy the build artifact to the ECS service in the ecs_demo_cluster.

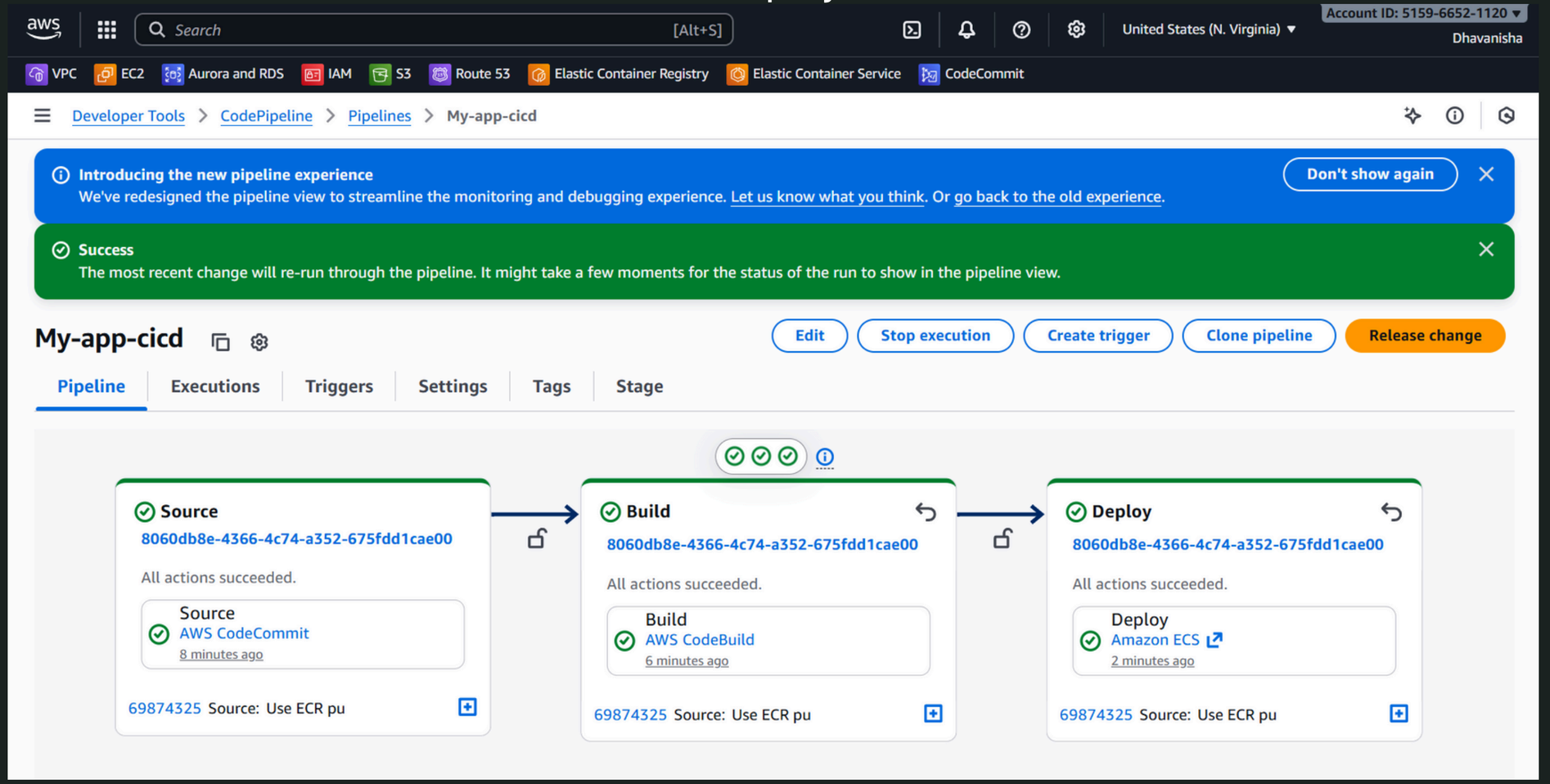


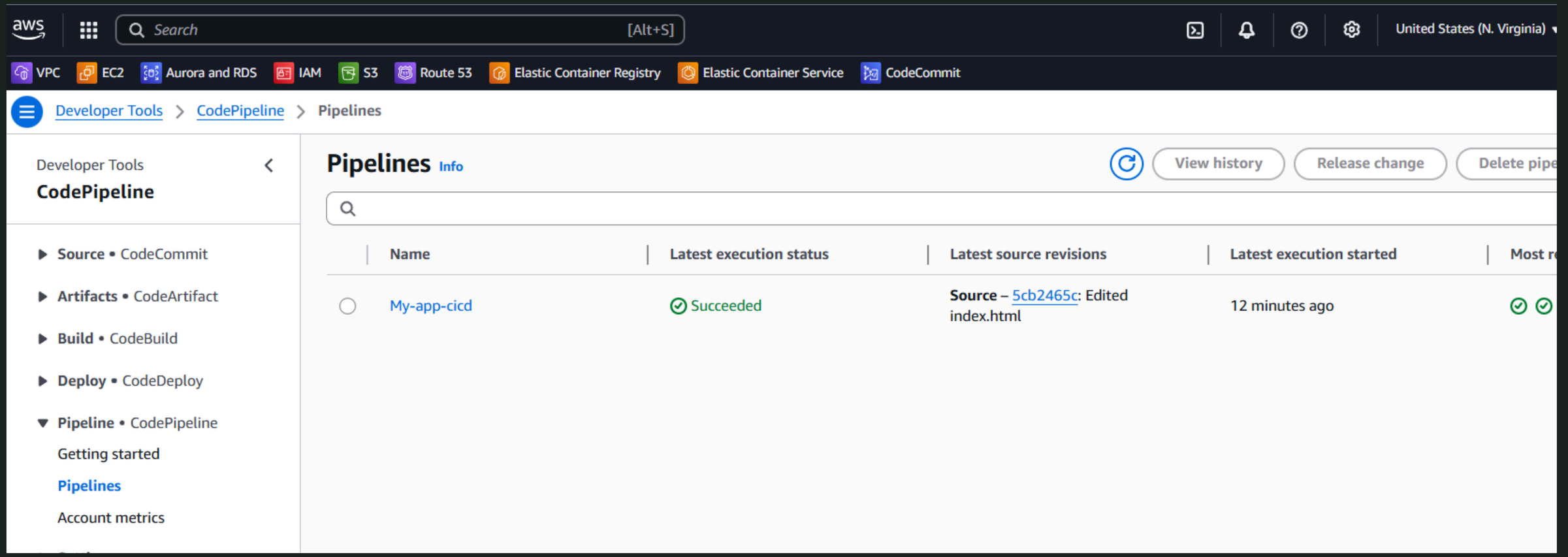
Configured the deploy stage to deploy the application to Amazon ECS with automatic rollback enabled.




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
Developer Tools
CodePipeline
Pipelines
my-app-cicd
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.
Don't show again
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 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
Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Account metrics
Pipelines Info
View history
Release change
Delete pipe
Name
Latest execution status
Latest source revisions
Latest execution started
Most r
My-app-cicd
Succeeded
Source – 5cb2465c: Edited index.html
12 minutes ago

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
Succeeded

Initiator
codepipeline/My-app-cicd

Build ARN
arn:aws:codebuild:us-east-1:515966521120:build/my-app-cicd-project-01:6b74e707-bce0-4300-a879-fbfeb21917a1

Resolved source version
6987432577010d84aa3cd2689ad17554a3858427

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

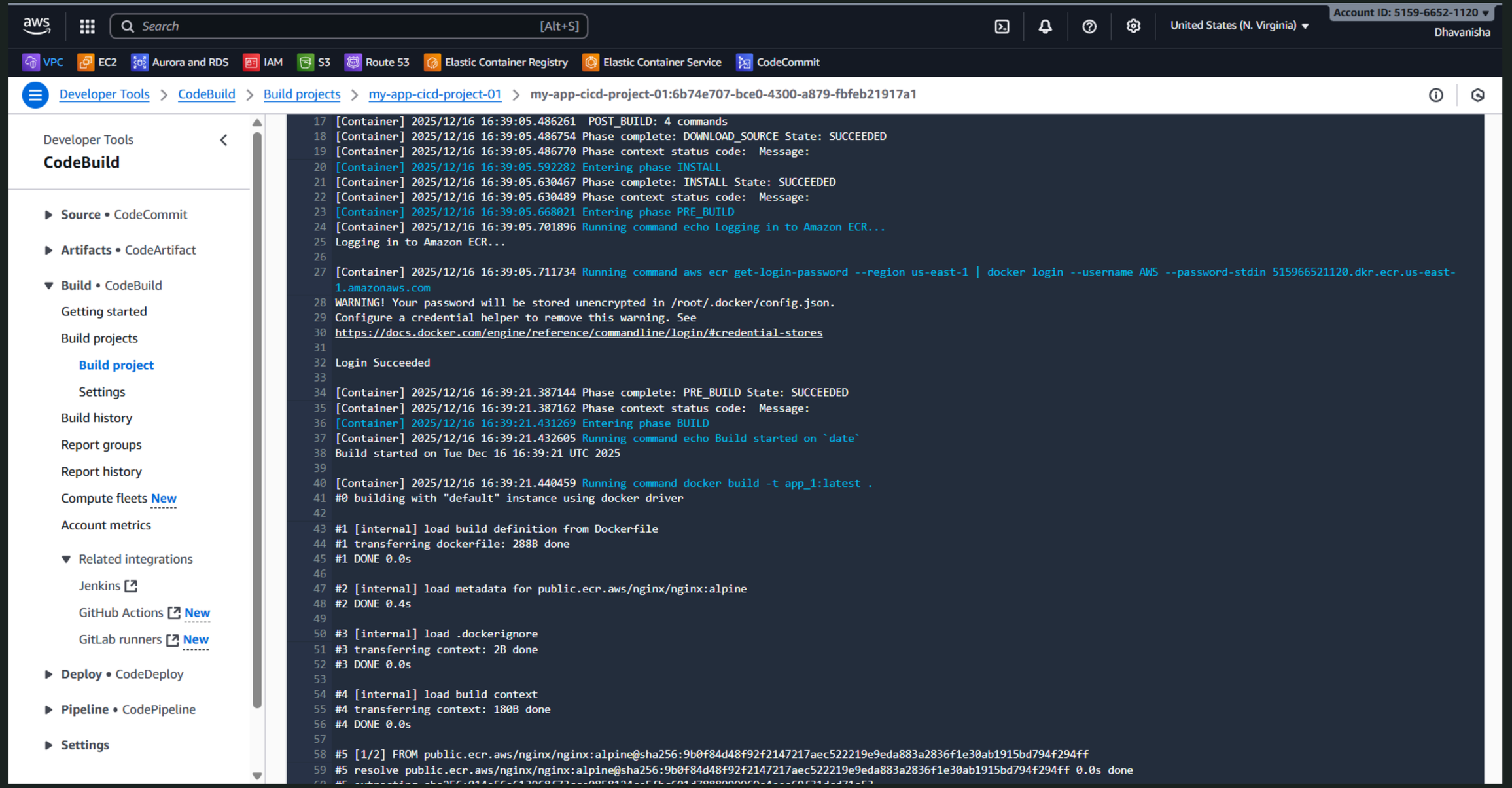
Showing the last 156 lines of the build log. View entire log

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: 4 commands
```

Build logs confirm successful Docker image build and ECR authentication.

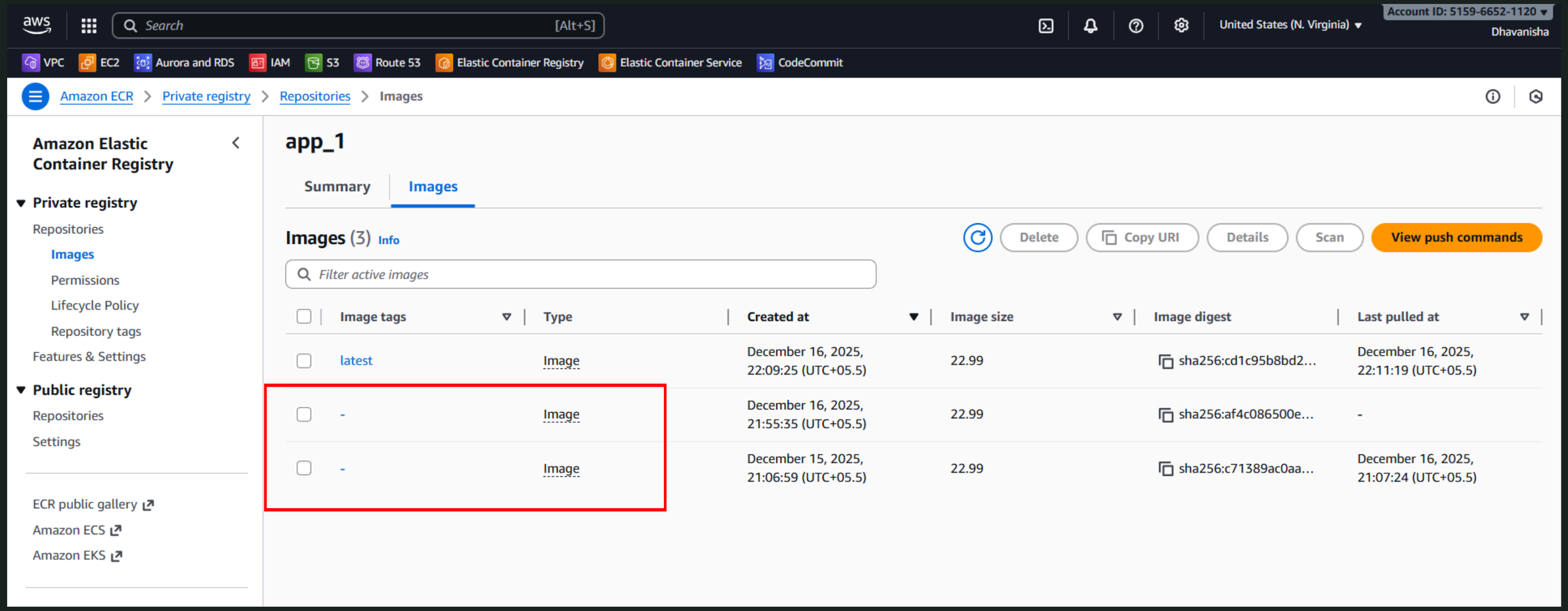


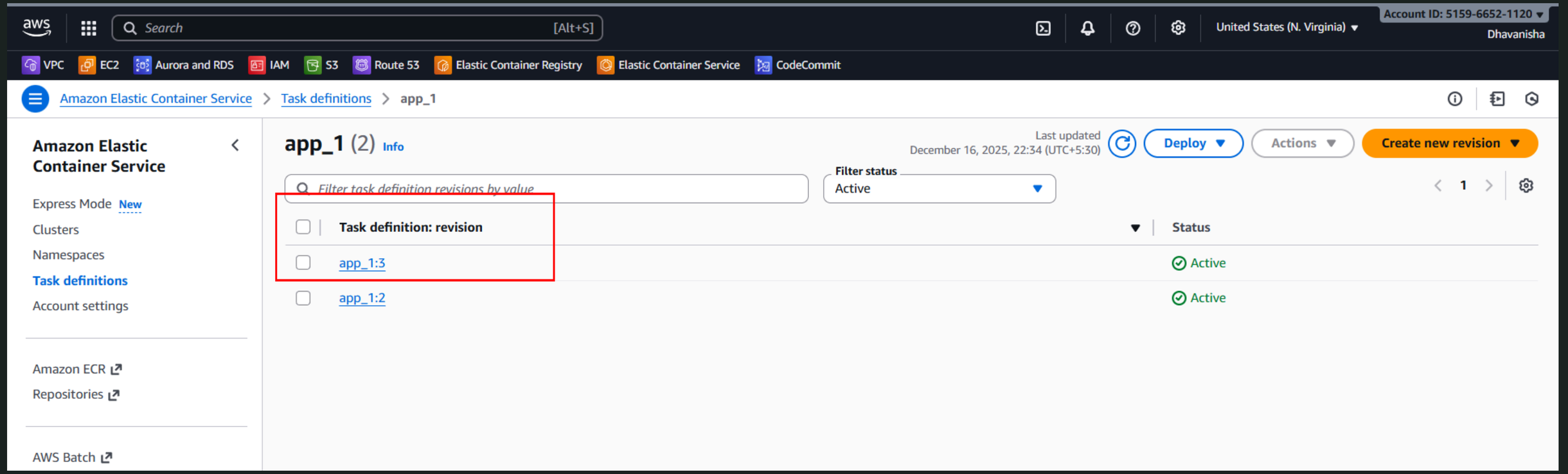
Developer Tools
CodeBuild

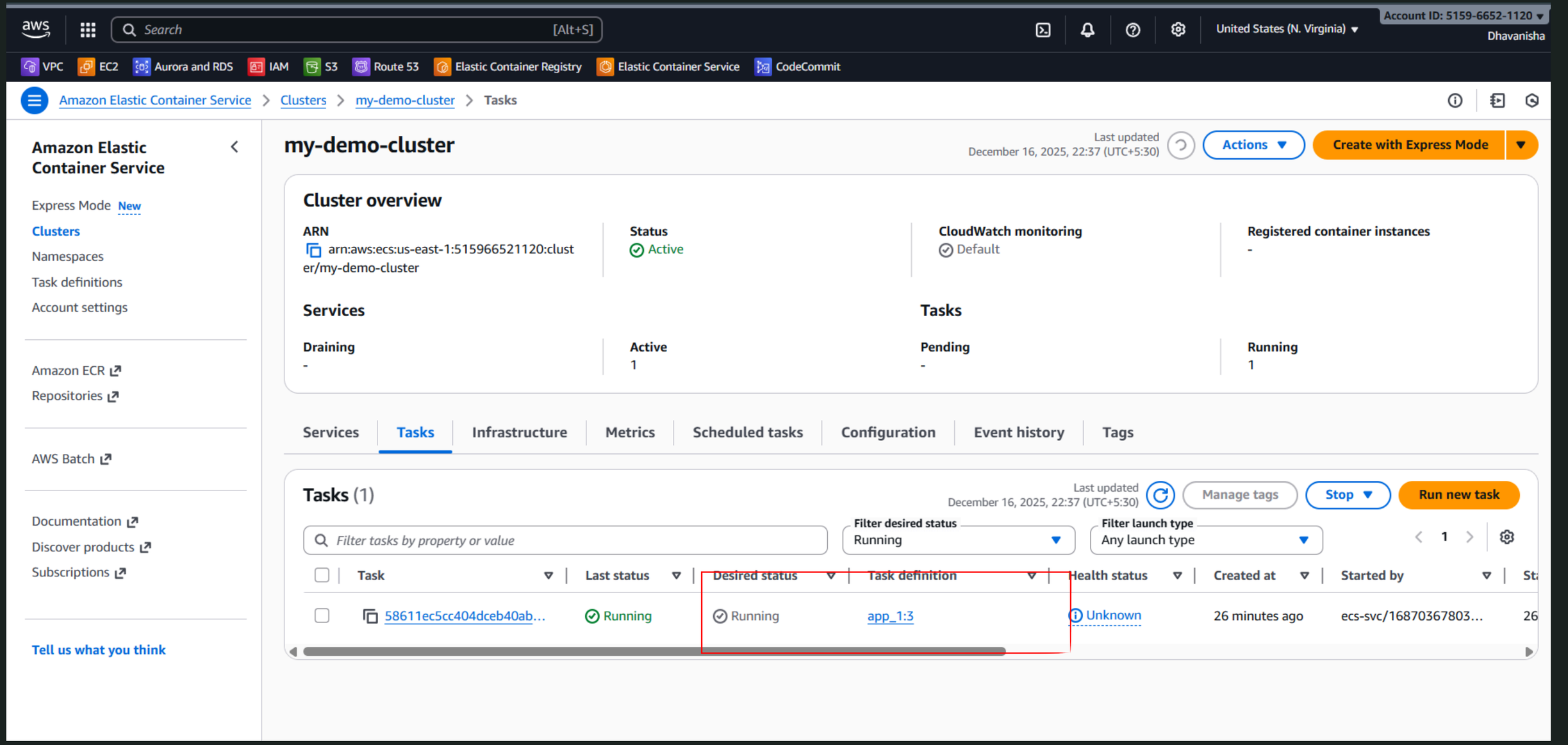
- Source • CodeCommit
- Artifacts • CodeArtifact
- Build • CodeBuild
  - Getting started
  - Build projects
    - Build project
    - Settings
  - Build history
  - Report groups
  - Report history
  - Compute fleets New
  - Account metrics
  - Related integrations
    - Jenkins
    - GitHub Actions 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 dcfd04fc2e09: 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:cd1c95b8bd2aff5a779227cfd54c2881f184b7a03bd3ae299bb0471838376216 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
```

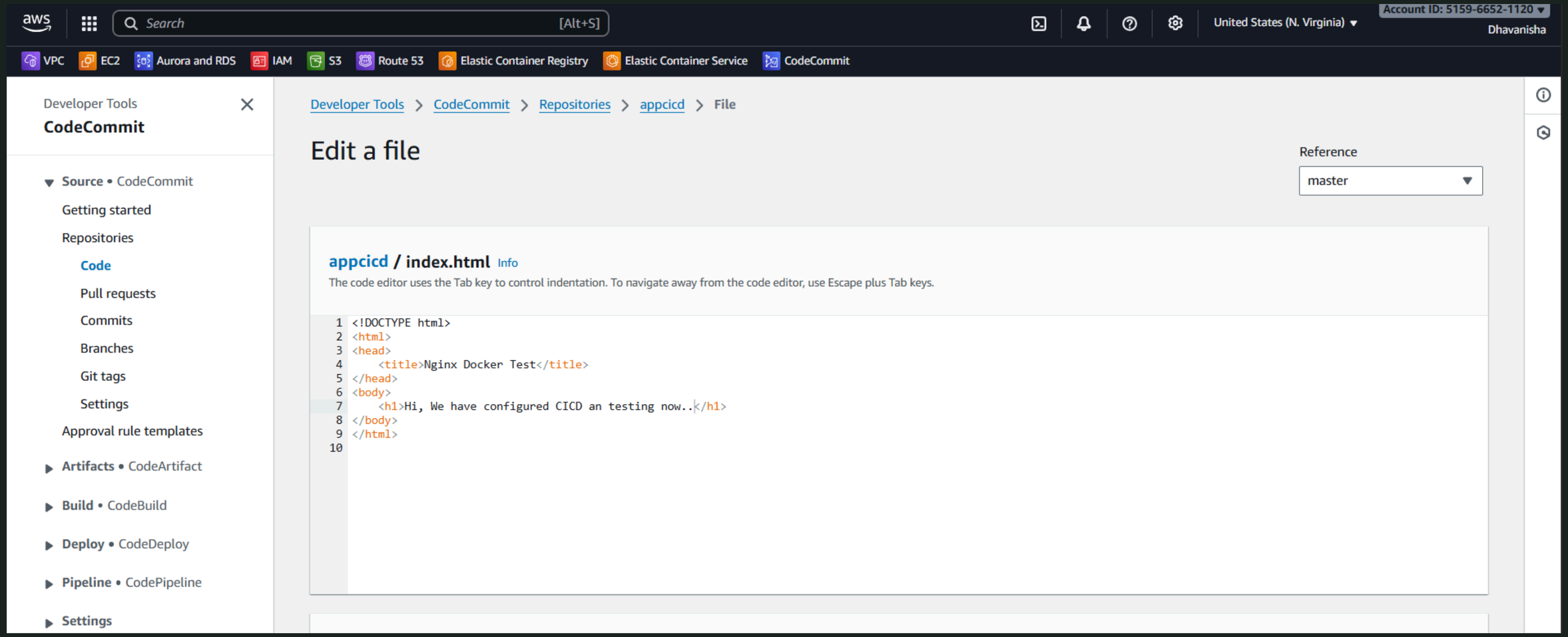
The Docker images were successfully pushed and stored in Amazon ECR for use in the ECS deployment.



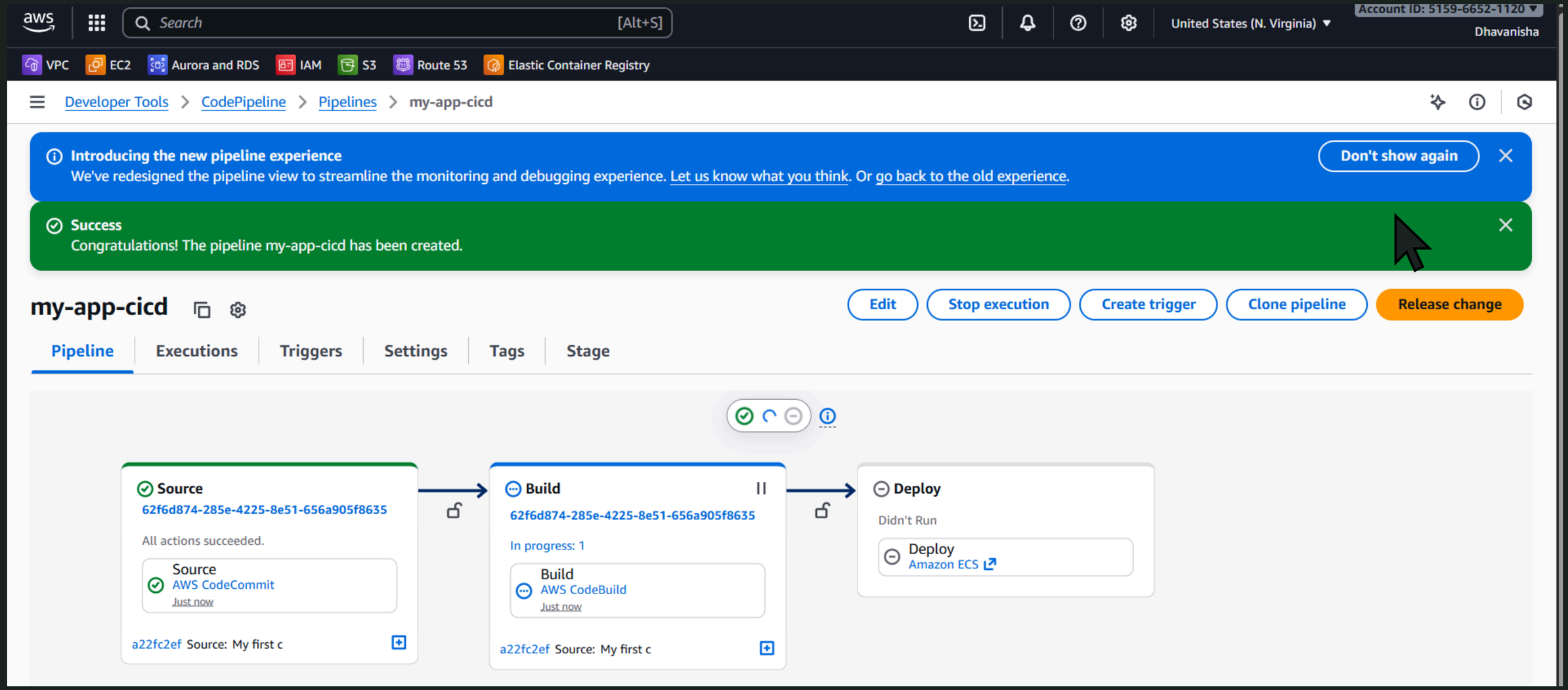

aws
Search
[Alt+S]
United States (N. Virginia)
Account ID: 5159-6652-1120
Dhavanisha
VPC
EC2
Aurora and RDS
IAM
S3
Route 53
Elastic Container Registry
Elastic Container Service
CodeCommit
Amazon Elastic Container Service
Task definitions
app_1
Amazon Elastic Container Service
Express Mode
New
Clusters
Namespaces
Task definitions
Account settings
Amazon ECR
Repositories
AWS Batch
app_1 (2)
Info
Last updated
December 16, 2025, 22:34 (UTC+5:30)
Deploy
Actions
Create new revision
Filter task definition revisions by value
Filter status
Active
1
Task definition: revision
Status
app_1:3
Active
app_1:2
Active


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
Elastic Container Service
CodeCommit
Amazon Elastic Container Service
Clusters
my-demo-cluster
Tasks
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
my-demo-cluster
Last updated
December 16, 2025, 22:37 (UTC+5:30)
Actions
Create with Express Mode
Cluster overview
ARN
arn:aws:ecs:us-east-1:515966521120:cluster/my-demo-cluster
Status
Active
CloudWatch monitoring
Default
Registered container instances
-
Services
Draining
-
Active
1
Tasks
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)
Manage tags
Stop
Run new task
Filter tasks by property or value
Filter desired status
Running
Filter launch type
Any launch type
1
Task
Last status
Desired status
Task definition
Health status
Created at
Started by
58611ec5cc404dceb40ab...
Running
Running
app_1:3
Unknown
26 minutes ago
ecs-svc/16870367803...

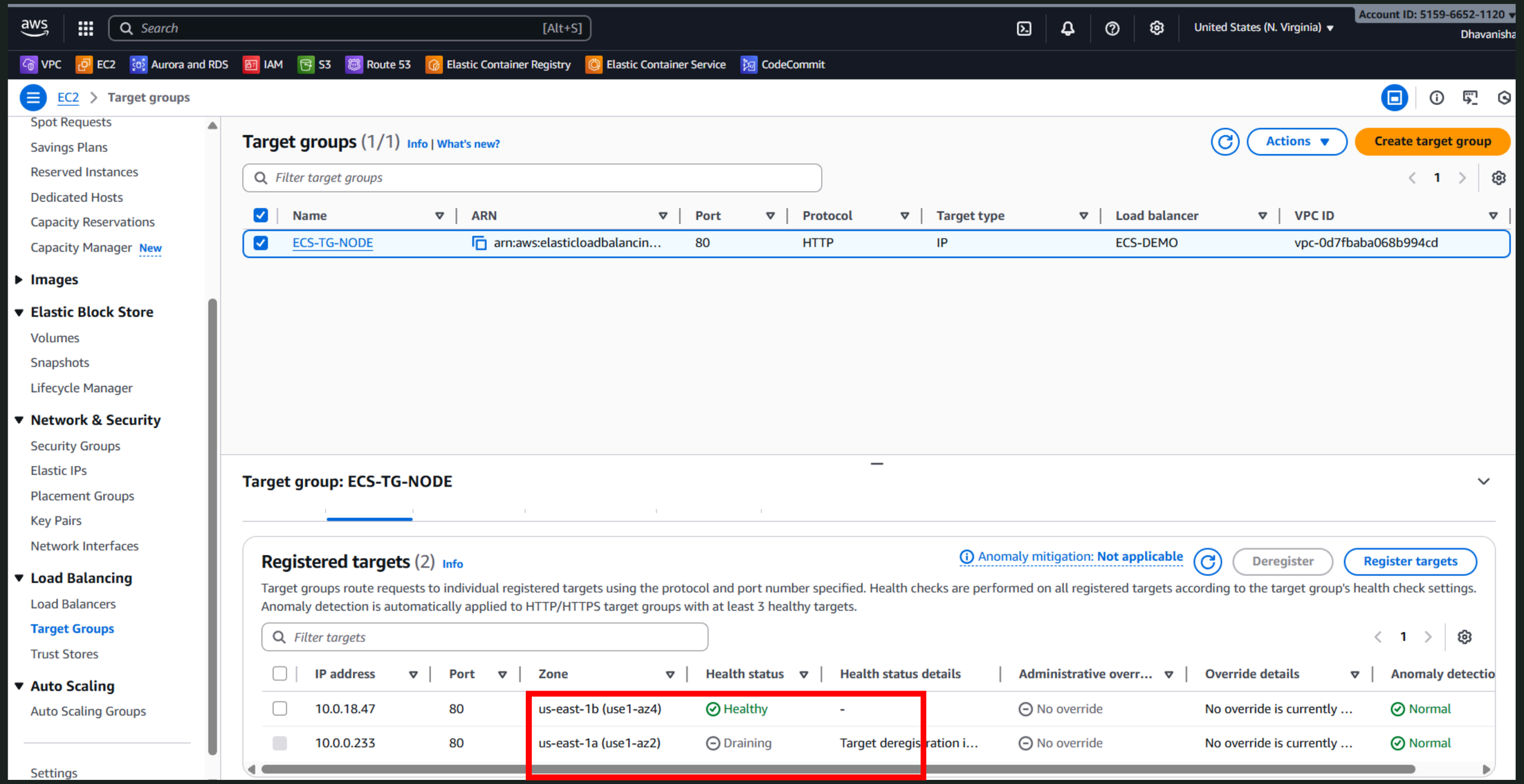
Modified the code to test automated deployment and validated it through the load balancer endpoint.

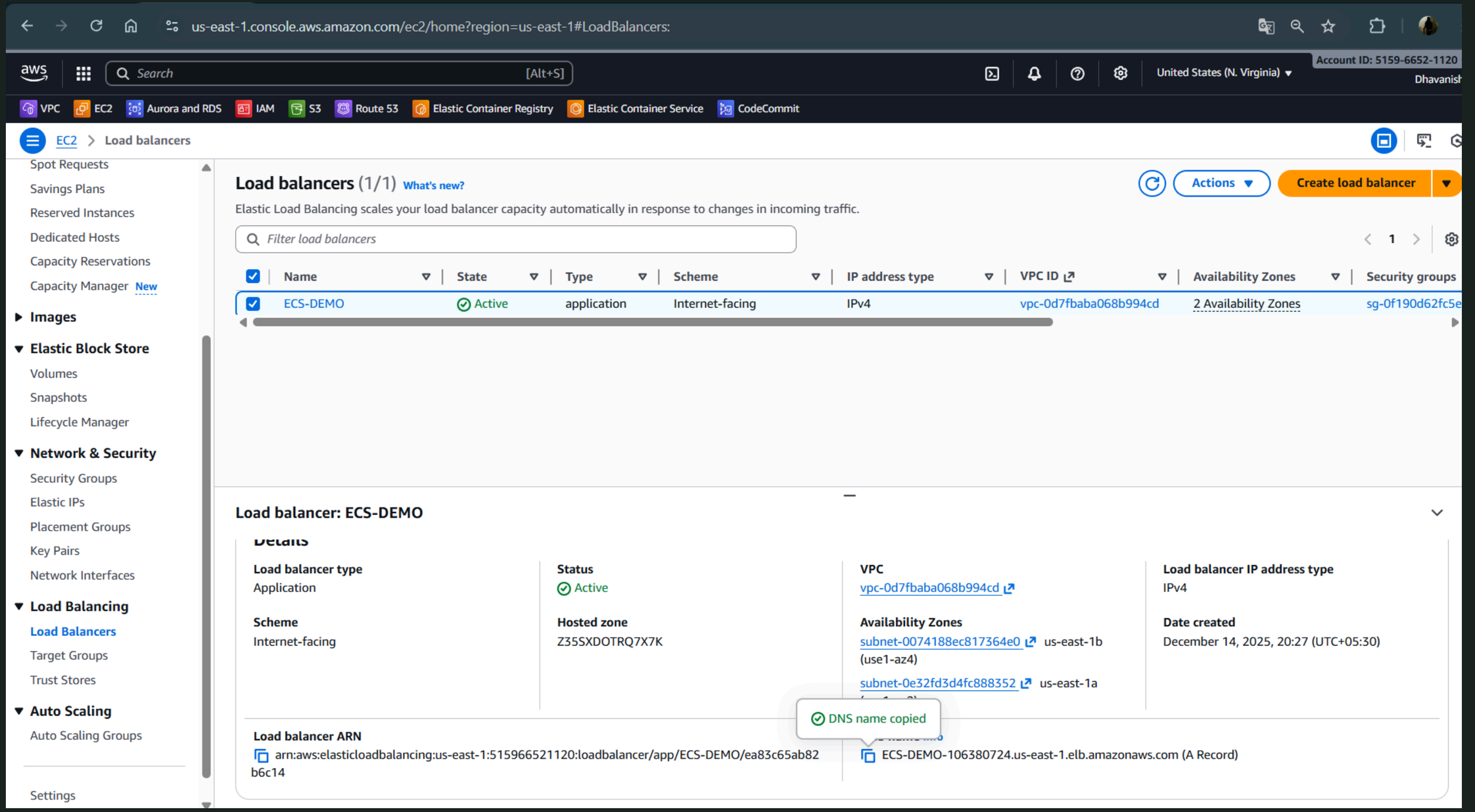


Clicked “Release change” to trigger the pipeline and test the CI/CD automation flow.

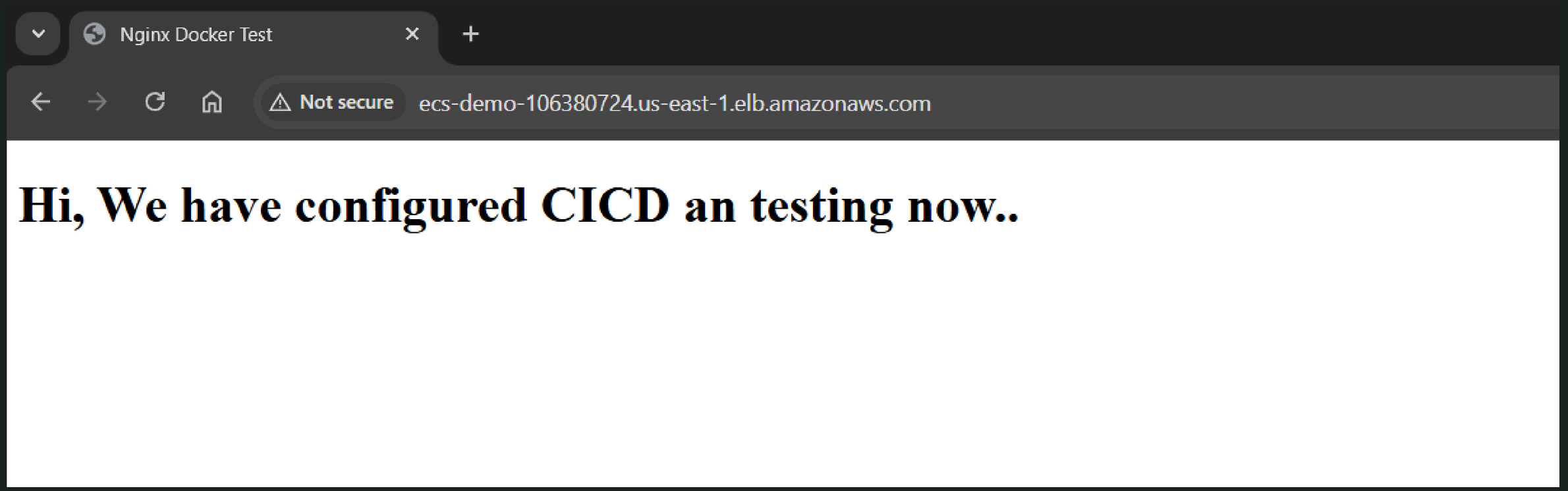


The target group shows the new target as healthy while the old IP is in draining state during deployment.

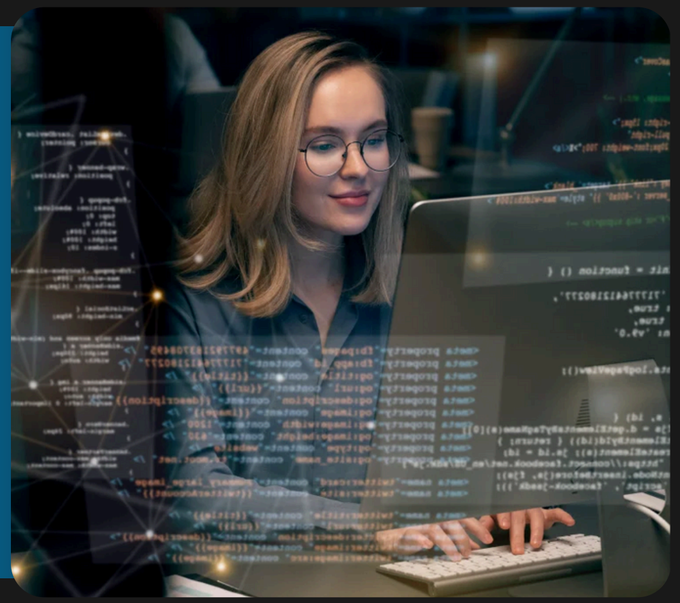



us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LoadBalancers:
Search [Alt+S]
United States (N. Virginia)
Account ID: 5159-6652-1120
Dhavanish
VPC EC2 Aurora and RDS IAM S3 Route 53 Elastic Container Registry Elastic Container Service CodeCommit
EC2 > Load balancers
Spot Requests
Savings Plans
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.
Actions
Create load balancer
Filter load balancers
1
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)
DNS name copied
Load balancer ARN
arn:aws:elasticloadbalancing:us-east-1:515966521120:loadbalancer/app/ECS-DEMO/ea83c65ab82b6c14
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



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