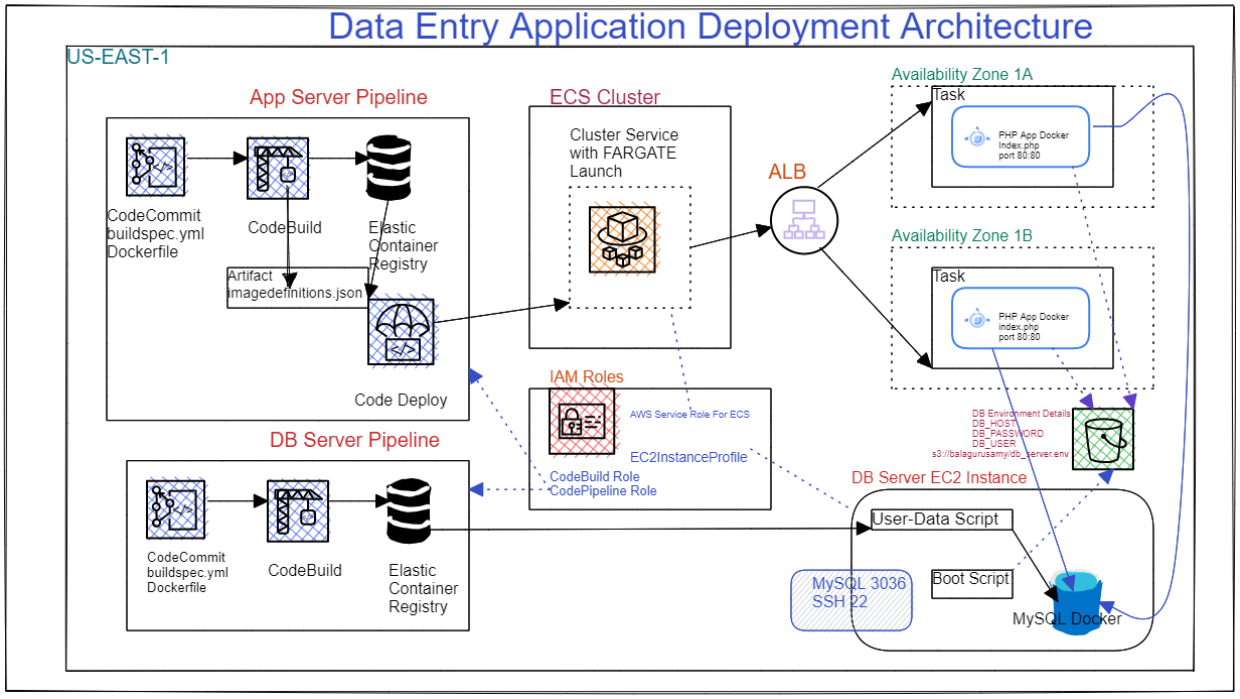


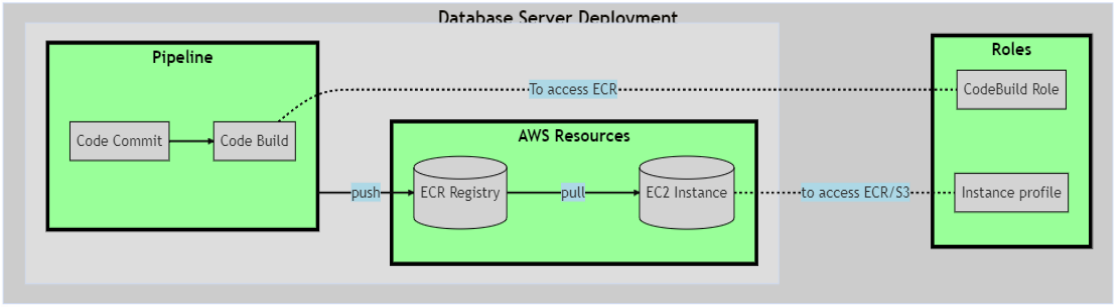
Deployment of the PHP Data Entry Application can be performed in 2 stages.

1. **Deployment of the backend MySQL database server**
   The DB Server docker file generates a MySQL Docker container with custom schema and data given in CRUD.SQL. Dockerfile and buildspec.yml are stored in CodeCommit. The CodeBuild generates the custom MySQL docker image and stores in ECR. A two-stage pipeline has been implemented to connect the CodeCommit repository with the CodeBuild project.
   A user-data startup script (db_ec2_inst_prereq.sh) has been implemented to install Docker and CLI tools during the DB Server EC2 instance initialization. The script also pulls the docker image from the ECR and builds the MySQL container with the 'always' restart policy and port mapping of 3603:3603. A boot-time script **copy_db_config** is also created to copy the DB Environment information (DB_HOST, DB_USER, DB_PASSOWRD) to an S3 object. (**s3://balagurusamy/db-server.env**). These environment variables are passed to the PHP application containers by the ECS service while creating tasks. This S3 object is specified during the task definition of PHP App Server Containers.
2. **Deployment of the PHP Application server**
   The app-server docker file pulls the PHP image from the docker hub, installs and enables the **mysqli** extension, configures the **php.ini** file. The **backend.php** file has been modified to get DB credentials from the environment variables DB_HOST, DB_USER, DB_PASSWORD. Dockerfile ,buildspec.yml and modified PHP application source files are

checked in to CodeCommit. The CodeBuild stage generates the docker image and stores it in ECR and creates an artifact file: **imagedefintions.json** This artifact file is used by the CodeDeploy service to identify the containers for the generated images. A Three-Stage CodePipeline has been implemented to automate the build and deployment process.

A load balancer is created with a target group with port mapping 80:80. A task definition has been created for the PHP container with the environment file: **s3://balagurusamy/db-server.env (DB_HOST, DB_USER, DB_PASSWORD)** generated by Database Server Instance. An ECS cluster has been created along with a FARGATE service to use the load balancer and its target group for running the tasks.

STAGE 1: Deployment of MySQL Database Container



Step 1: Implementation of Dockerfile

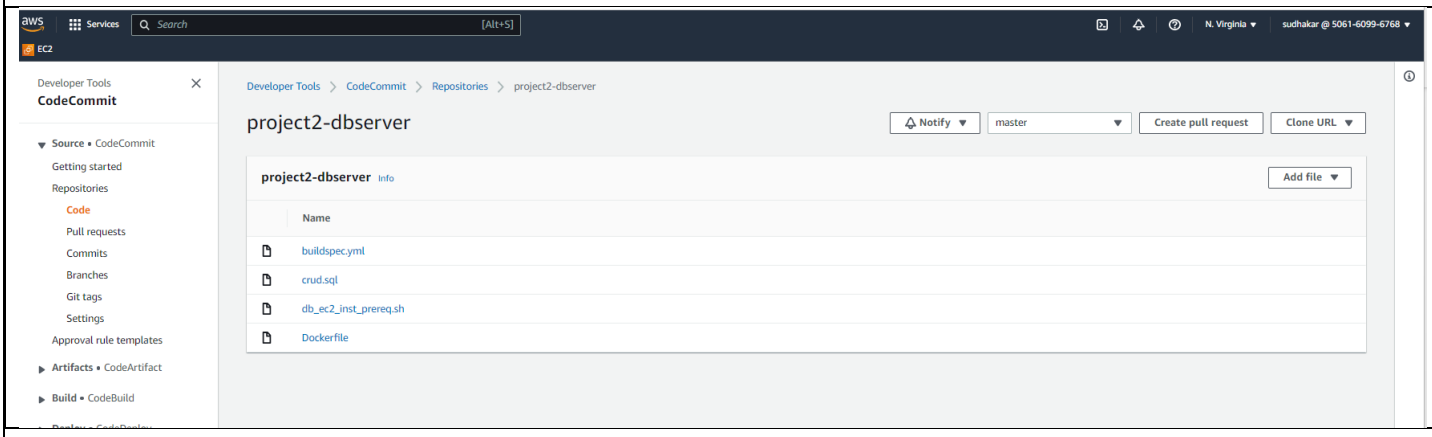
Contents of Dockerfile

FROM mysql:8.0.32 COPY ./crud.sql /docker-entrypoint-initdb.d	This pulls mysql:8.0.32, copies the **crud.sql** to **/docker-entrypoint-initdb.d**. The files from this folder will be executed when the container is **run very first time.**

Step 2: Implementation of buildspec.yml

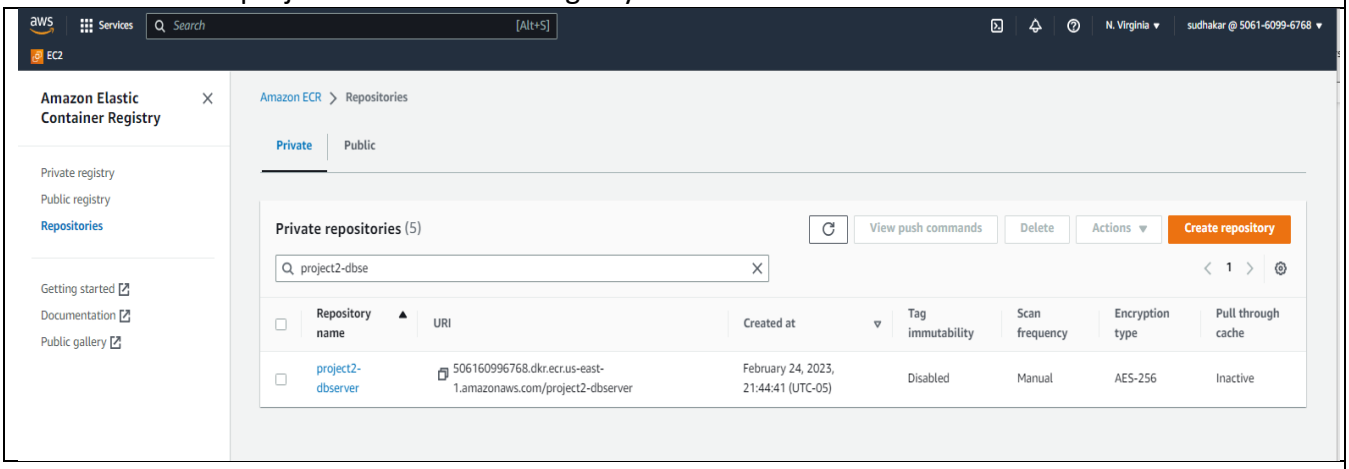
```
version: 0.2

phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
#This logs in to AWS ECR
      - aws ecr get-login-password --region $AWS_DEFAULT_REGION | docker login --username AWS --password-stdin $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
#To build the docker container and store it in ECR
      - docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG .
      - docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG
      - printf '[{"name":"project2-db-docker","imageUri":"%s"}]' $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG > imagedefinitions.json
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
      - docker push $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG
artifacts:
  files:
    - imagedefinitions.json
```

Step 3: Create a CodeCommit repository project2-dbserver and check in Dockerfile, buildspec.yml, crud.sql ,db_ec2_inst_prereq.sh (startup script for DB Server EC2 instance)



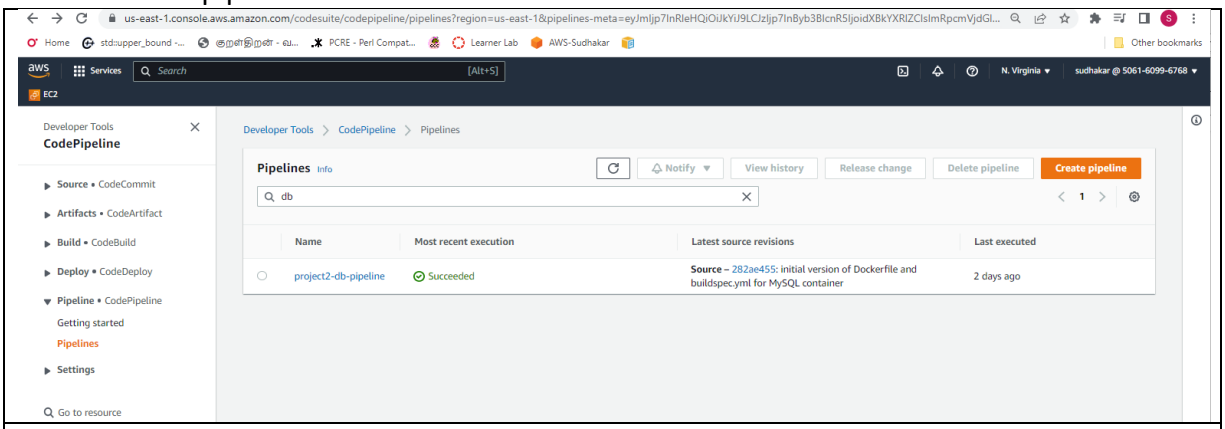
Step 4: Create a Repository in Elastic Container Registry: project2-dbserver

Screenshot of the project2-dbserver ECR registry

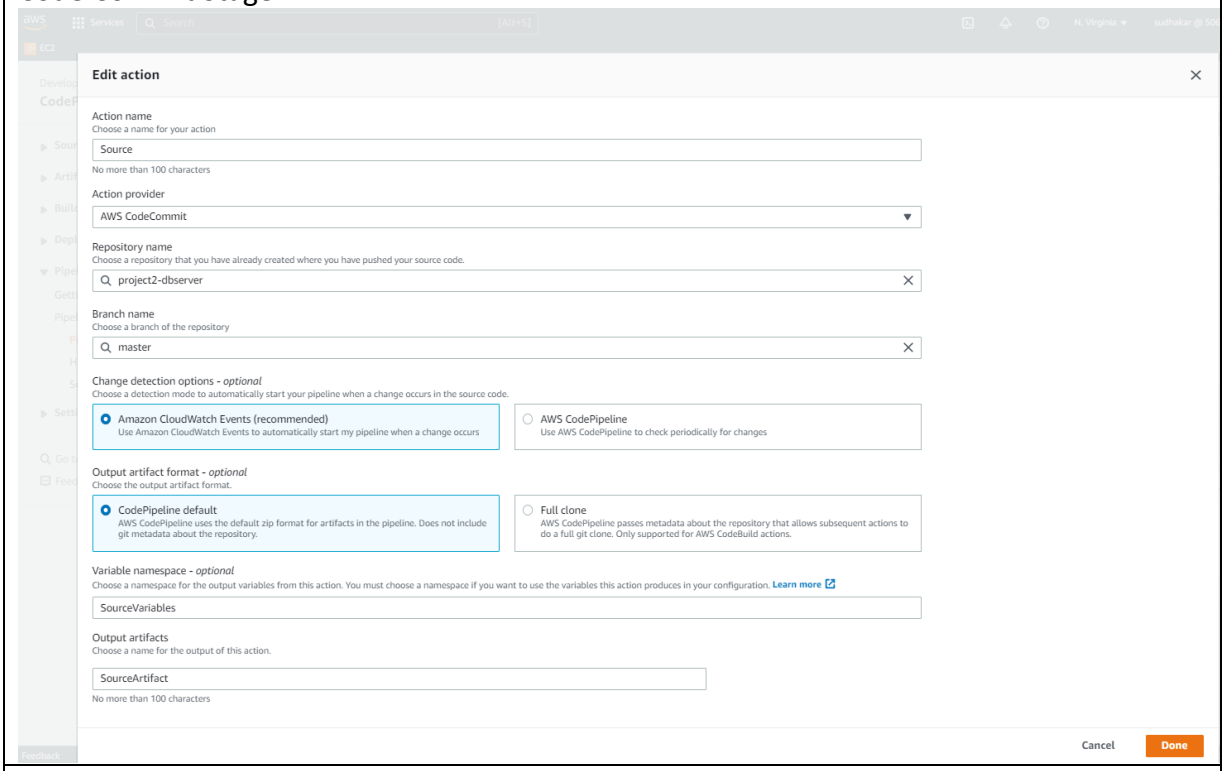


Step 5: Create a CodePipeline: project2-db-pipeline

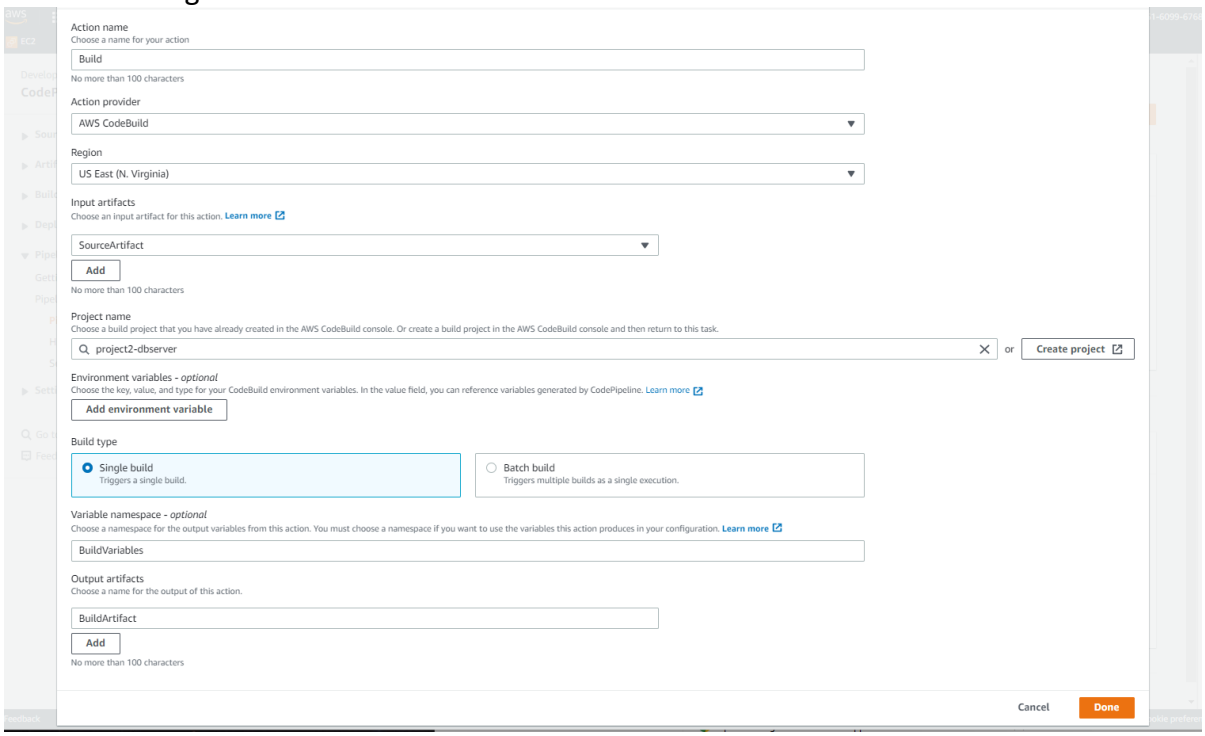
Screenshots of pipeline and its details



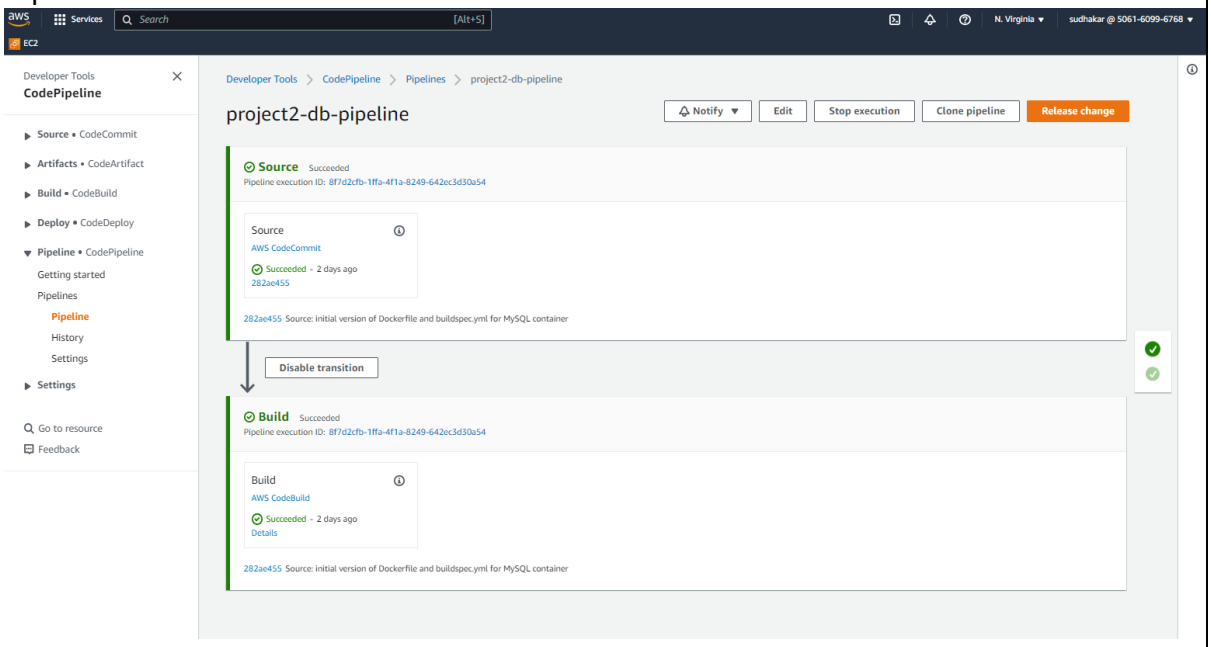
Code Commit Stage



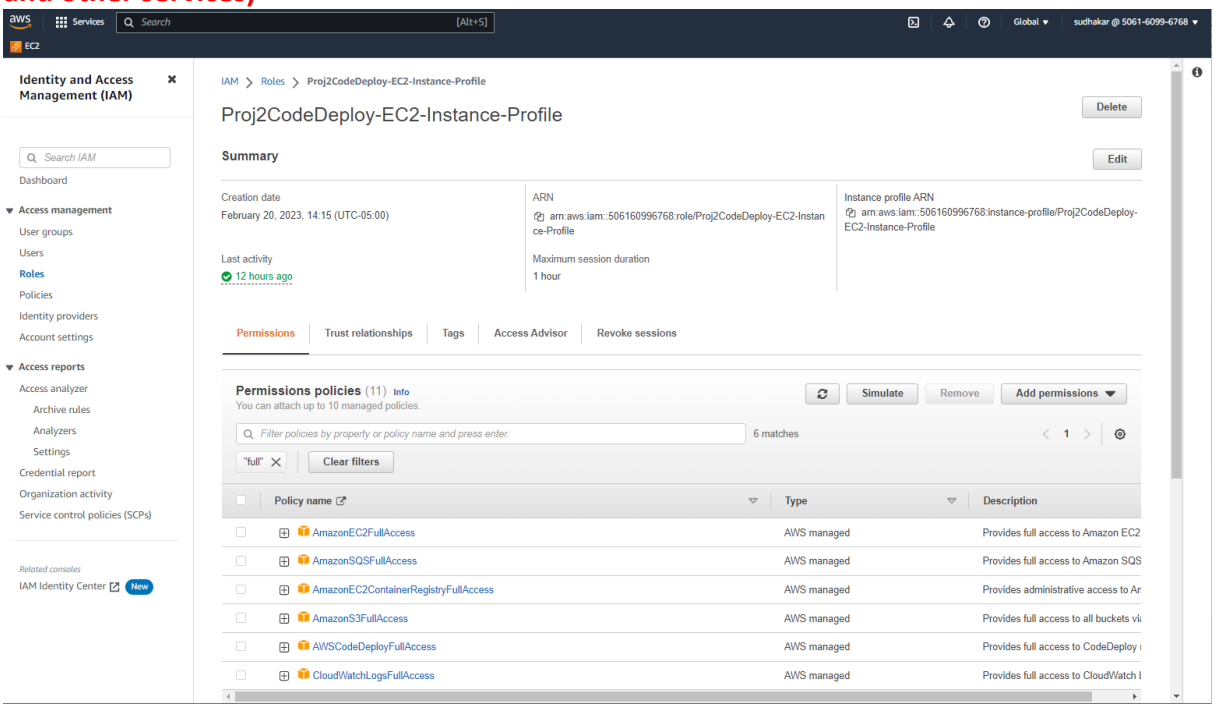
CodeBuild Stage



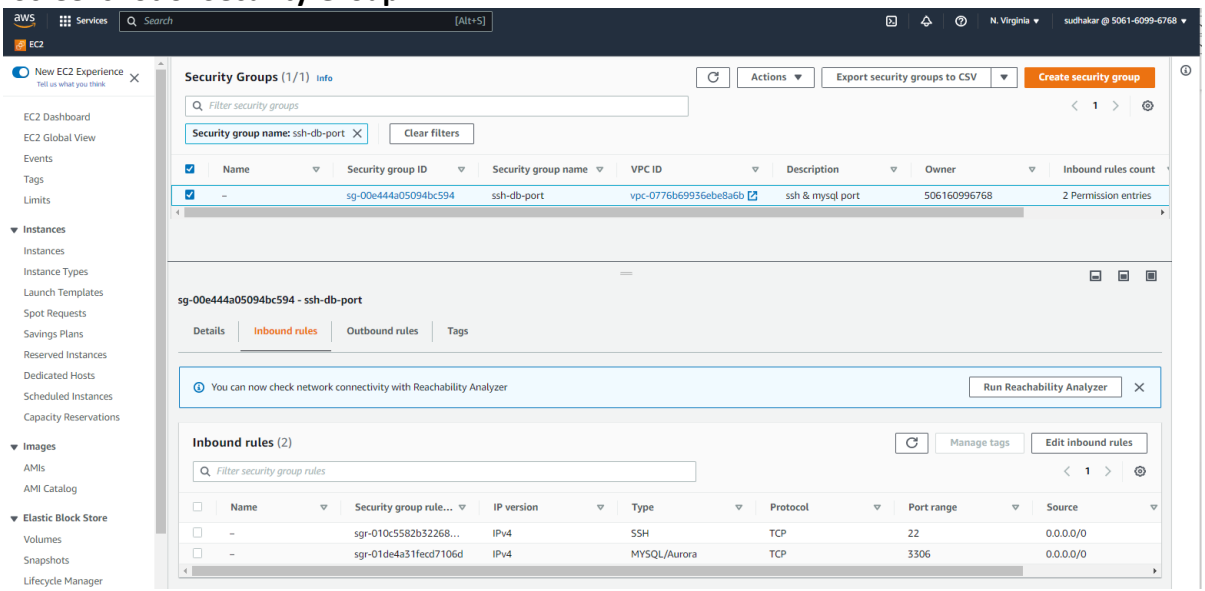
Successful pipeline workflow. This creates MySQL docker image in the project2-dbserver repo in ECR



Step 6: Create an IAM Role for the DB Server EC2 Instance (to access ECR, S3,CodeDeploy and other services)



Step 7: Create Security Group: ssh-db-port (to grant access to SSH 22 & MySQL 3306 ports) -Screenshot of Security Group



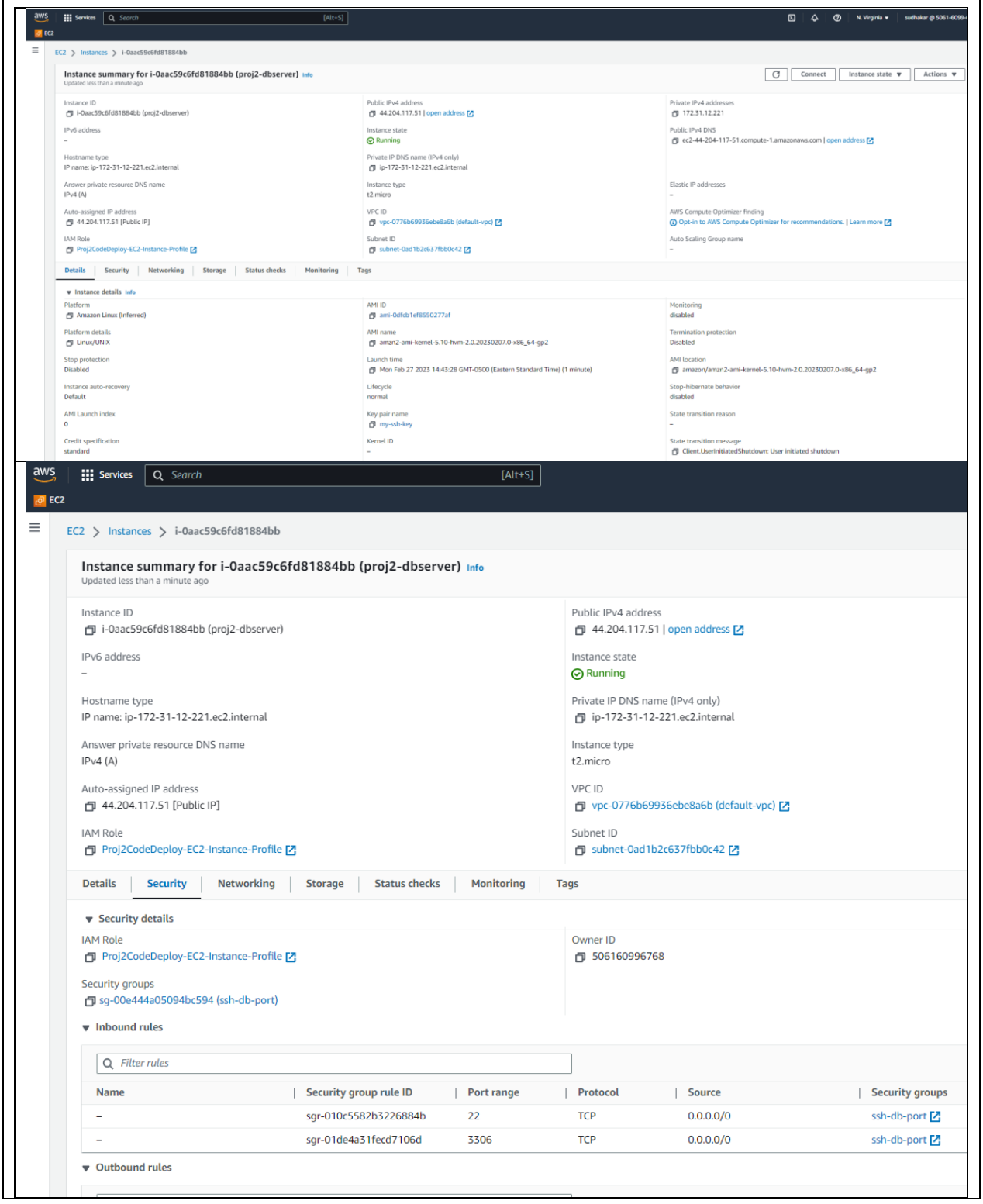
Step 8: Implementation of startup script for the DB Server EC2 instance

This script will be executed when the EC2 instance is created. This installs the docker, AWS CLI, creates copy_db_config script to copy the DB environment variables (DB_HOST,DB_USER,DB_PASSWORD) to an S3 object. (s3://balagurusamy/db-server.env). This will be used by the PHP application container.

Contents of the script: db_ec2_inst_prereq.sh

```
#!/bin/bash
yum update -y
#install docker
amazon-linux-extras install -y docker
systemctl enable docker
usermod -aG docker ec2-user
#install CLI
yum install awscli -y
#build the docker container with restart policy-always
docker run --name db-docker --restart always -e MYSQL_ROOT_PASSWORD="bluehills" -
d -p 3306:3306 ACCTNO.dkr.ecr.us-east-1.amazonaws.com/project2-dbserver:latest
#Create a boot-time script: copy_db_config to store the db credentials in S3
cat << EOM > /var/lib/cloud/scripts/per-boot/copy_db_config
#!/bin/bash
public_ip=\$(curl -s http://169.254.169.254/latest/meta-data/public-ipv4)
echo "DB_HOST=\$public_ip" > /tmp/db-server.env
echo "DB_USER=root" >> /tmp/db-server.env
echo "DB_PASSWORD=bluehills" >> /tmp/db-server.env
echo "DB_DATABASE=user" >> /tmp/db-server.env
aws s3 cp /tmp/db-server.env s3://balagurusamy
EOM
#Execute the copy_db_config
chmod +x /var/lib/cloud/scripts/per-boot/copy_db_config
/var/lib/cloud/scripts/per-boot/copy_db_config
```

Step 9: Launch an EC2 instance for DB Server with ssh/mysql ports enabled and initialize the instance with the user-data script: db_ec2_inst_prereq.sh



Step 10: Log in to DB server and verify the MySQL container installation.
Login to EC2 instance

```
root@UrsaMajor: ~   root@ip-172-31-12-221:/home
(mypython) sudhakar@UrsaMajor:~$ ssh -i ".ssh/my-ssh-key.pem" ec2-user@ec2-44-204-117-51.compute-1.amazonaws.com
Last login: Mon Feb 27 20:08:49 2023 from c-73-159-171-229.hsd1.ma.comcast.net

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-12-221 ~]$ sudo -s
```

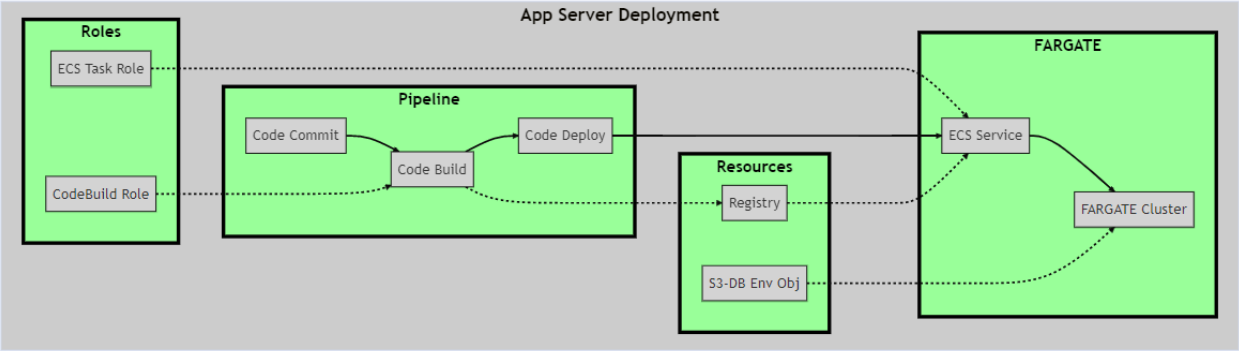
Check the installationstatus using grep **/var/log/cloud-init-output.log** for docker, ecr and db-server.env

```
[root@ip-172-31-12-221 log]# grep docker cloud-init-output.log
Cleaning repos: amzn2-core amzn2extra-docker amzn2extra-kernel-5.10
---> Package docker.x86_64 0:20.10.17-1.amzn2.0.2 will be installed
--> Processing Dependency: runc >= 1.0.0 for package: docker-20.10.17-1.amzn2.0.2.x86_64
--> Processing Dependency: libcgroup >= 0.40.rc1-5.15 for package: docker-20.10.17-1.amzn2.0.2.x86_64
--> Processing Dependency: containerd >= 1.3.2 for package: docker-20.10.17-1.amzn2.0.2.x86_64
--> Processing Dependency: pigz for package: docker-20.10.17-1.amzn2.0.2.x86_64
 docker         x86_64     20.10.17-1.amzn2.0.2     amzn2extra-docker      39 M
 containerd     x86_64     1.6.8-1.amzn2.0.1        amzn2extra-docker      27 M
 runc           x86_64     1.1.4-1.amzn2.0.1        amzn2extra-docker     2.9 M
  Installing : docker-20.10.17-1.amzn2.0.2.x86_64                           5/5
  Verifying  : docker-20.10.17-1.amzn2.0.2.x86_64                           4/5
  docker.x86_64 0:20.10.17-1.amzn2.0.2
Installing docker
 20  docker=latest           enabled      \
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
[root@ip-172-31-12-221 log]# grep ecr cloud-init-output.log
Unable to find image '506160996768.dkr.ecr.us-east-1.amazonaws.com/project2-dbserver:latest' locally
Status: Downloaded newer image for 506160996768.dkr.ecr.us-east-1.amazonaws.com/project2-dbserver:latest
[root@ip-172-31-12-221 log]# grep db-server cloud-init-output.log
upload: tmp/db-server.env to s3://balagurusamy/db-server.env
upload: tmp/db-server.env to s3://balagurusamy/db-server.env
upload: tmp/db-server.env to s3://balagurusamy/db-server.env
upload: tmp/db-server.env to s3://balagurusamy/db-server.env
[root@ip-172-31-12-221 log]#
```

Execute bash script on the MySQL container to check the execution status of crud.sql

```
[root@ip-172-31-12-221 ec2-user]# docker exec -it db-docker bash
bash-4.4# mysql -pbluehills
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.32 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| user               |
+--------------------+
5 rows in set (0.00 sec)

mysql> select * from user.crud;
+----+-----------------------+--------------------+------------+-----------------+
| id | name                  | email              | phone      | city            |
+----+-----------------------+--------------------+------------+-----------------+
| 40 | Mahatma gandhi        | gandhi@gandhi.com  | 9191919191 | New Delhi       |
| 42 | Issac Newton          | newton@newton.com  | 4444444    | London          |
| 44 | sudhakar balagurusamy | whitesnow@aol.com  | 7814921488 | South Weymouth  |
+----+-----------------------+--------------------+------------+-----------------+
3 rows in set (0.00 sec)

mysql>
```

Stage 2: Deployment of the Data Entry PHP application server



Step 1: Implementation of Dockerfile

Implement a Dockerfile to pull the latest PHP image,
configure the php.ini,
install the MYSQLI extension,
add the PHP pages

Contents of Dockerfile

FROM php:8.1.16-apache *# Copy the Application files to temporary folder /tmp/app* *COPY ./configure ./index.php /tmp/app/* *COPY ./ajax/ /tmp/app/ajax/* *COPY ./backend/ /tmp/app/backend/* *COPY ./css/ /tmp/app/css/* *# Run the configure script to enable and install mysqli extension* *# confgure script also copies the application files to appropriate html folders* *RUN /bin/bash /tmp/app/configure*	This pulls the php:8.1.16, copies the application files and the configure script to temporary folder and runs the configure script.

Contents of the script: configure.

# Copy the application files to /var/www/html folder cp -r /tmp/app/ajax/ /var/www/html cp -r /tmp/app/backend/ /var/www/html cp -r /tmp/app/css/ /var/www/html # Copy the modified index.php cp /tmp/app/index.php /var/www/html/ # Create the php.ini configuration file cp /var/www/html/php.ini-production /var/www/html/php.ini # Enable mysqli extension to access the MySQL database server sed -i 's/^;extension=mysqli/extension=mysqli/g' /var/www/html/php.ini # Install the mysqli extension docker-php-ext-install mysqli	This copies the application files to html folder and enables and installs mysqli extension module.

Contents of backend/database.php

<?php
 	
// Create connection
$conn = mysqli_connect(getenv("DB_HOST"), getenv("DB_USER"), getenv("DB_PASSWORD"), getenv("DB_DATABASE"));
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}
?>
	Database.php has been modified to retrieve DB INFO from the environment variables: DB_HOST, DB_USER, DB_PASSWORD.

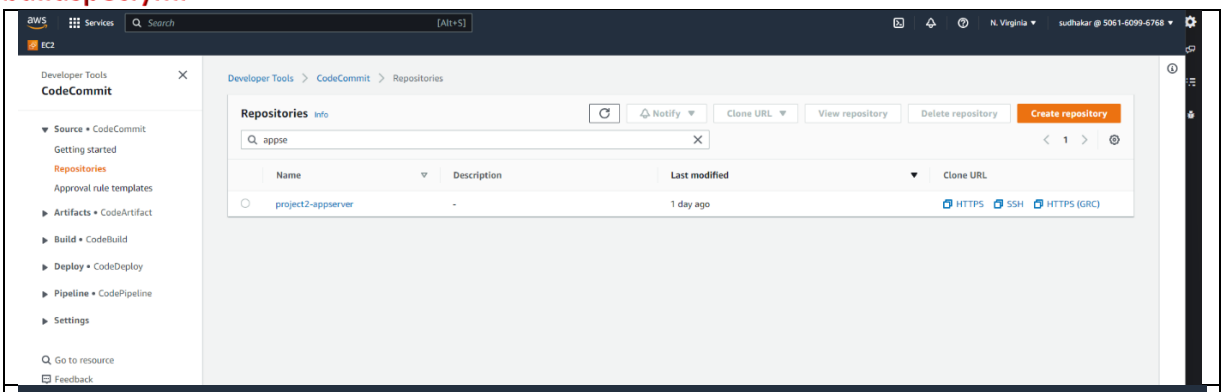
Step 2: Implementation of buildspec.yml

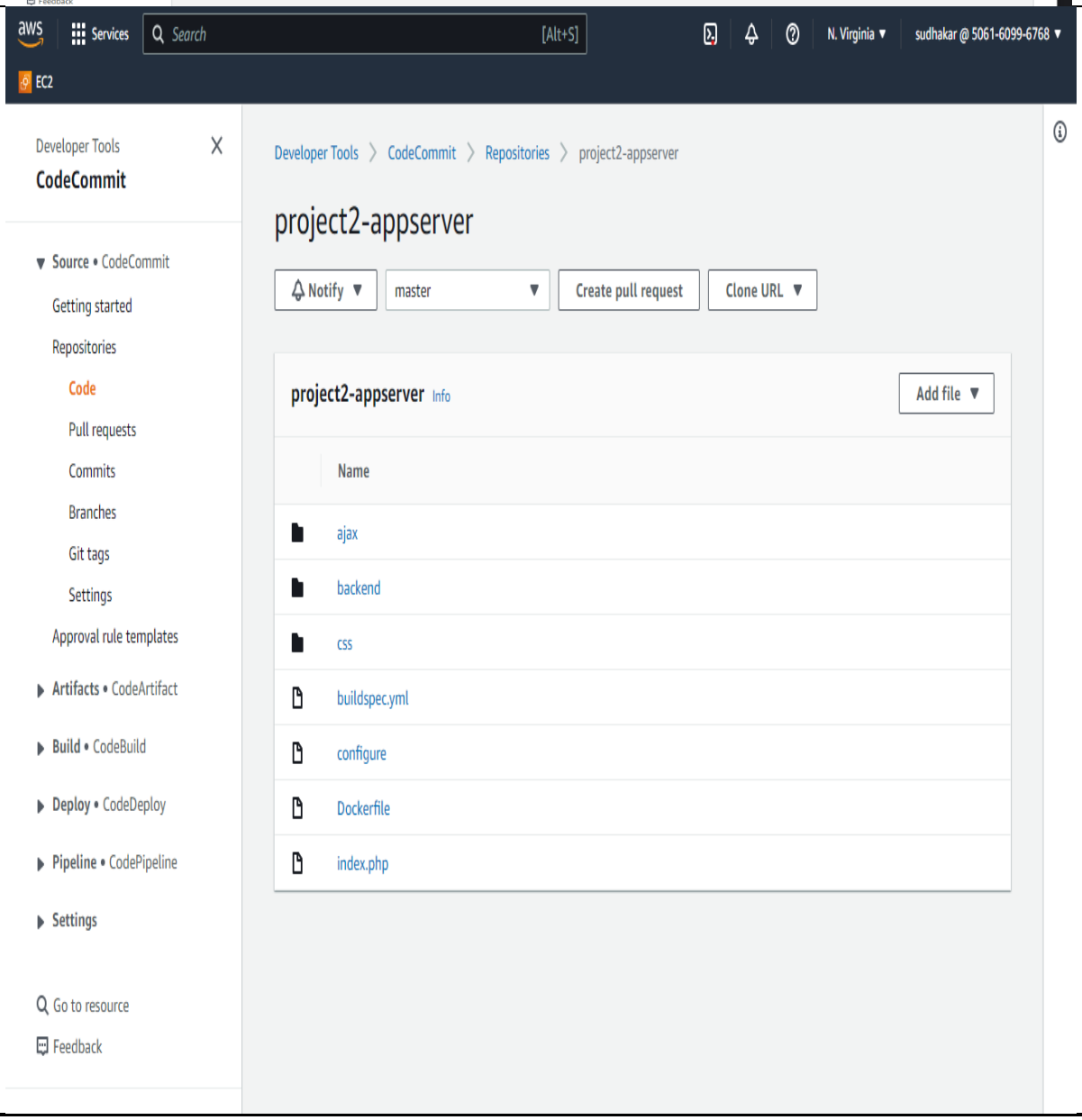
Contents of buildspec.yml

This is used by CodeBuild application to build the container and store it in the ECR. The **IAM role** used for the CodeBuild application needs to have write access to ECR (Elastic Container Registry). This also generates an artifact file **imagedefinitions.json** used by the Elastic Container Service to deploy the generated image to the appropriate FARGATE cluster containers.

```
version: 0.2
phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws ecr get-login-password --region $AWS_DEFAULT_REGION | docker login --username AWS --
password-stdin $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG .
      - docker tag $IMAGE_REPO_NAME:$IMAGE_TAG
$AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG
      - printf '[{"name":"project2-docker","imageUri":"%s"}]'
$AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG
> imagedefinitions.json
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
      - docker push
$AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG
artifacts:
  files:
    - imagedefinitions.json
```

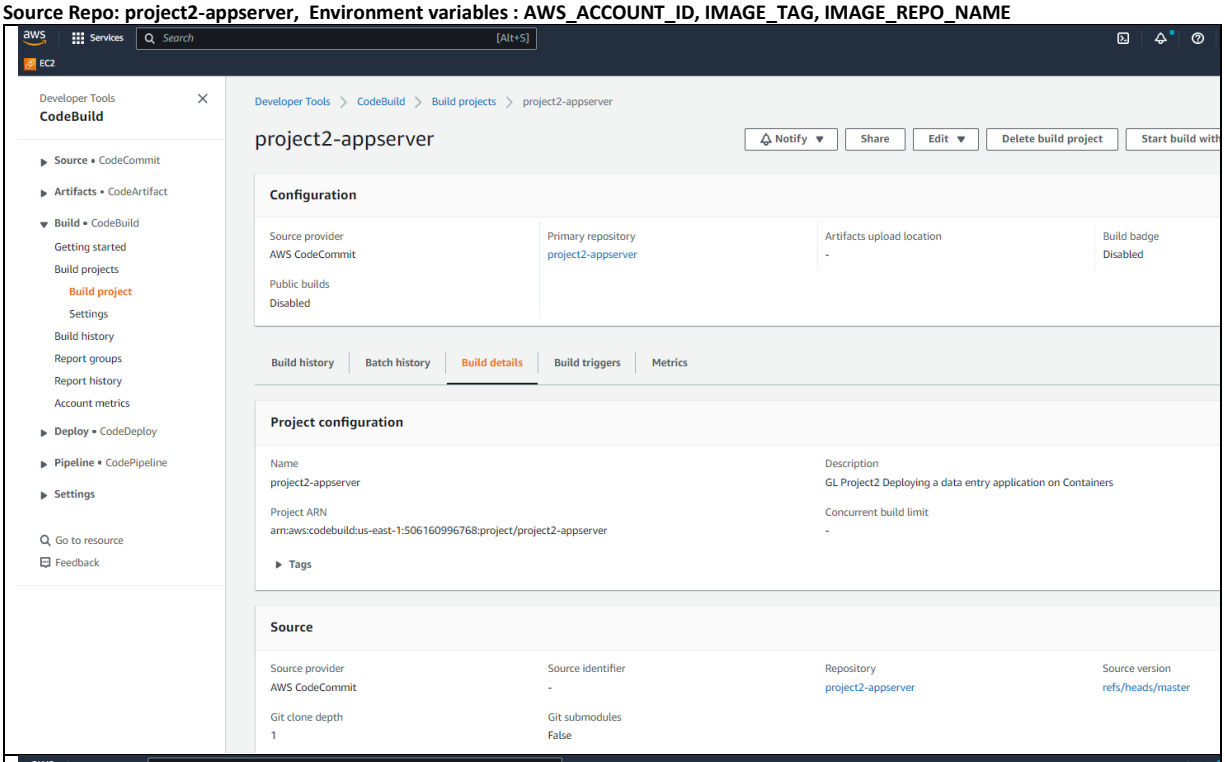
Step 3: Create a CodeCommit repository and check in the Application files, Dockerfile and buildspec.yml

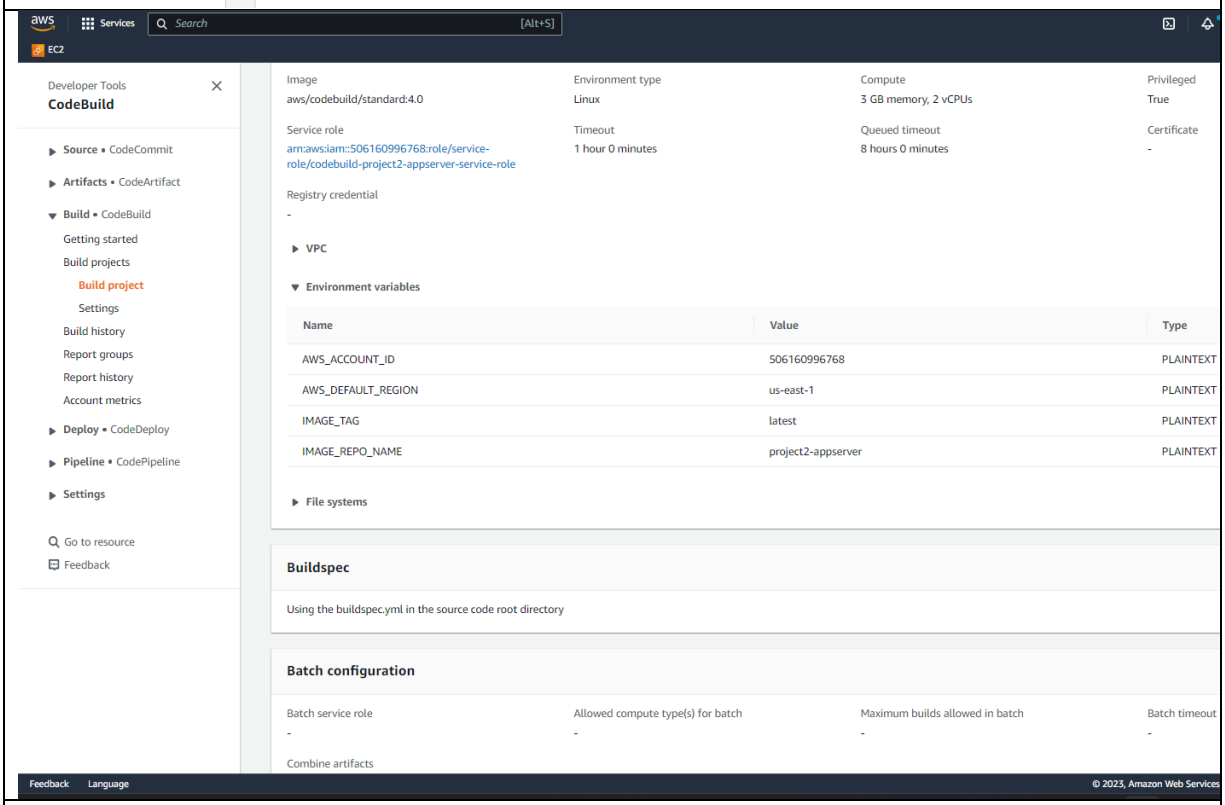




Step 4: Create a CodeBuild Project: project2-appserver.

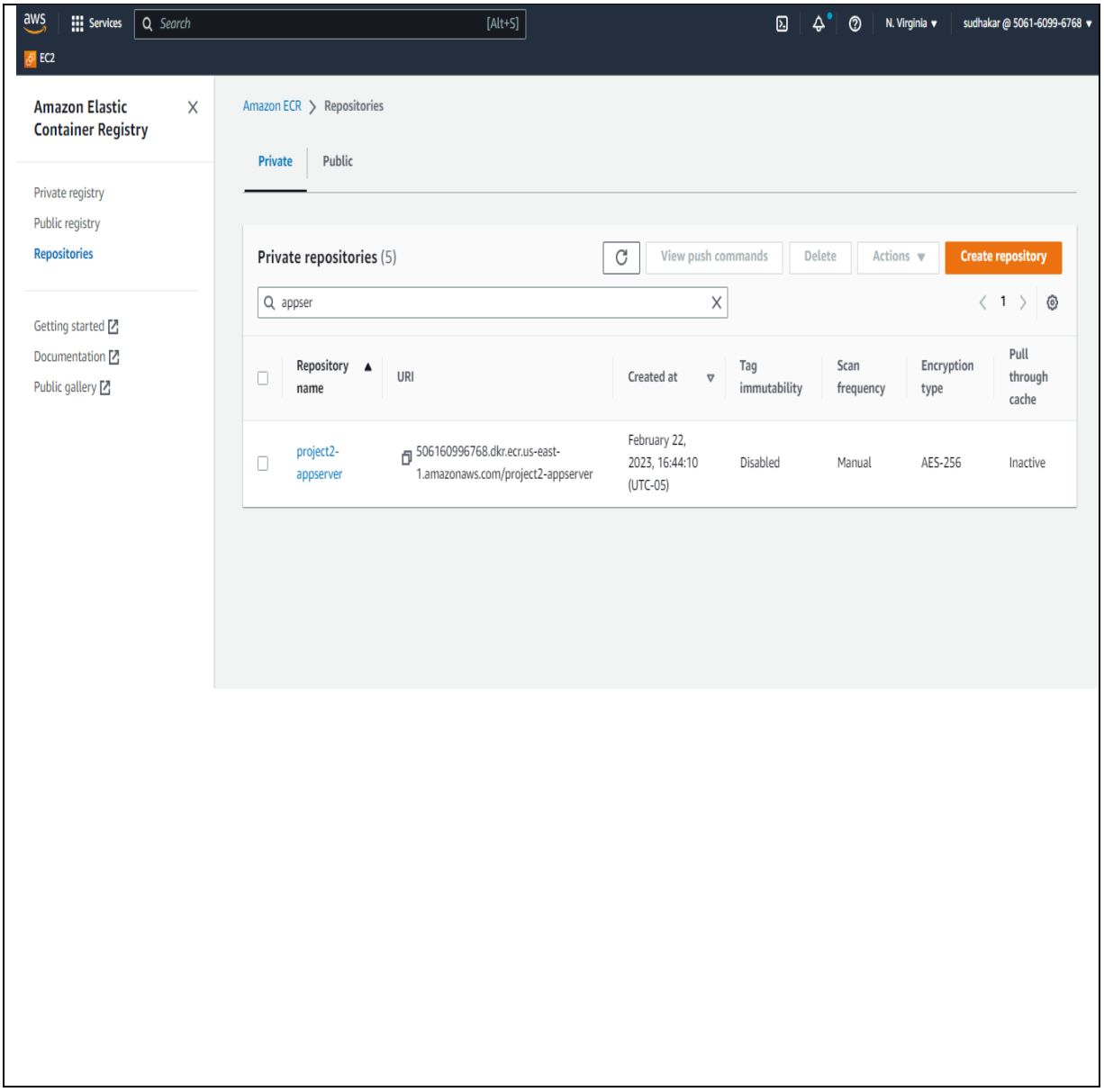
Source Repo: project2-appserver, Environment variables : AWS_ACCOUNT_ID, IMAGE_TAG, IMAGE_REPO_NAME





Step 5: Create an Elastic Container Registry Repository: project2-appserver.

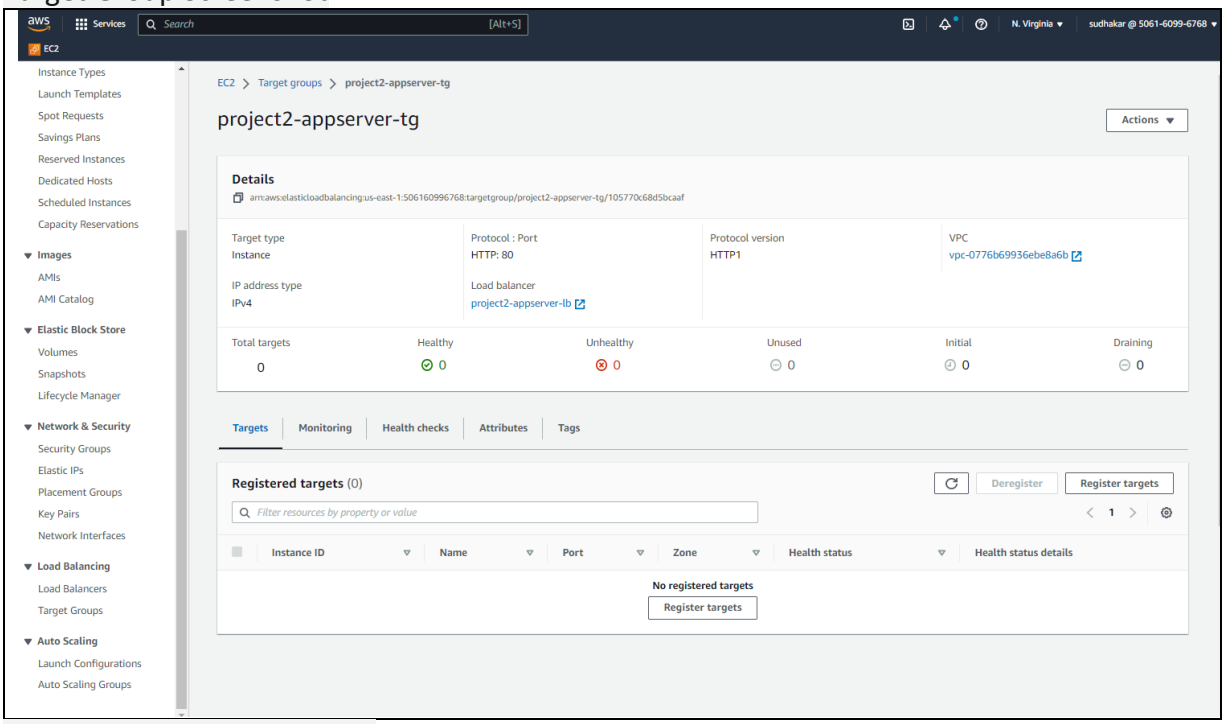
ACCTNO.dkr.ecr.us-east-1.amazonaws.com/project2-appserver



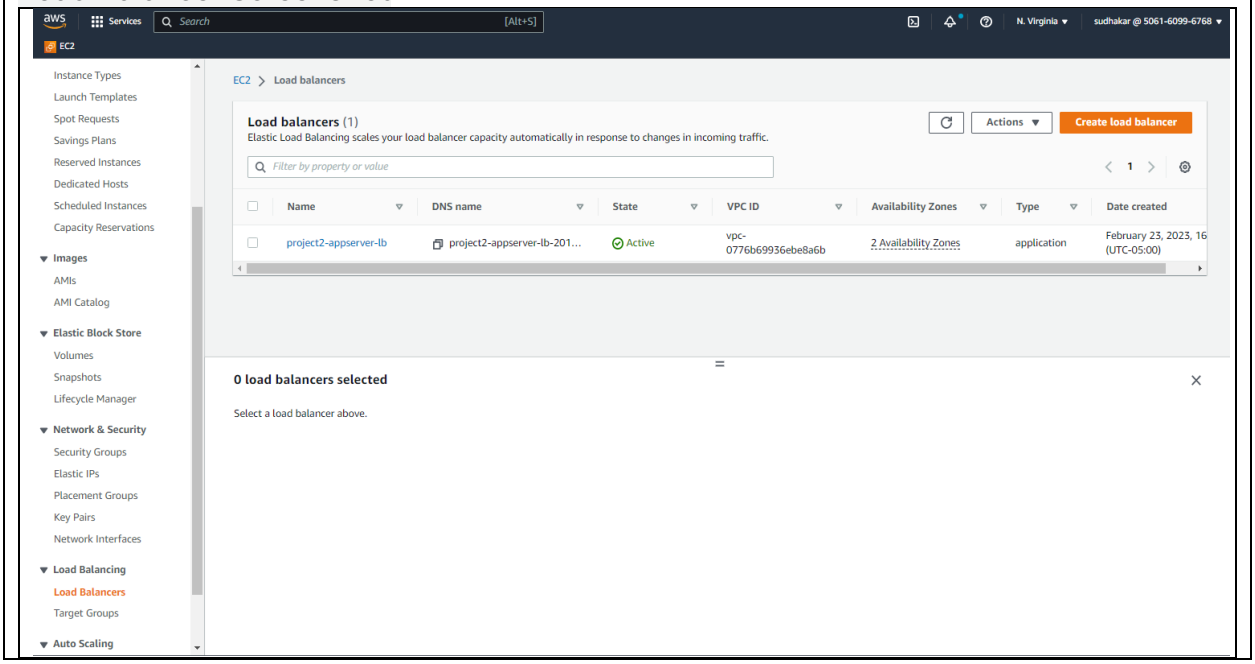
Step 6: Create a target group: project2-appserver-tg

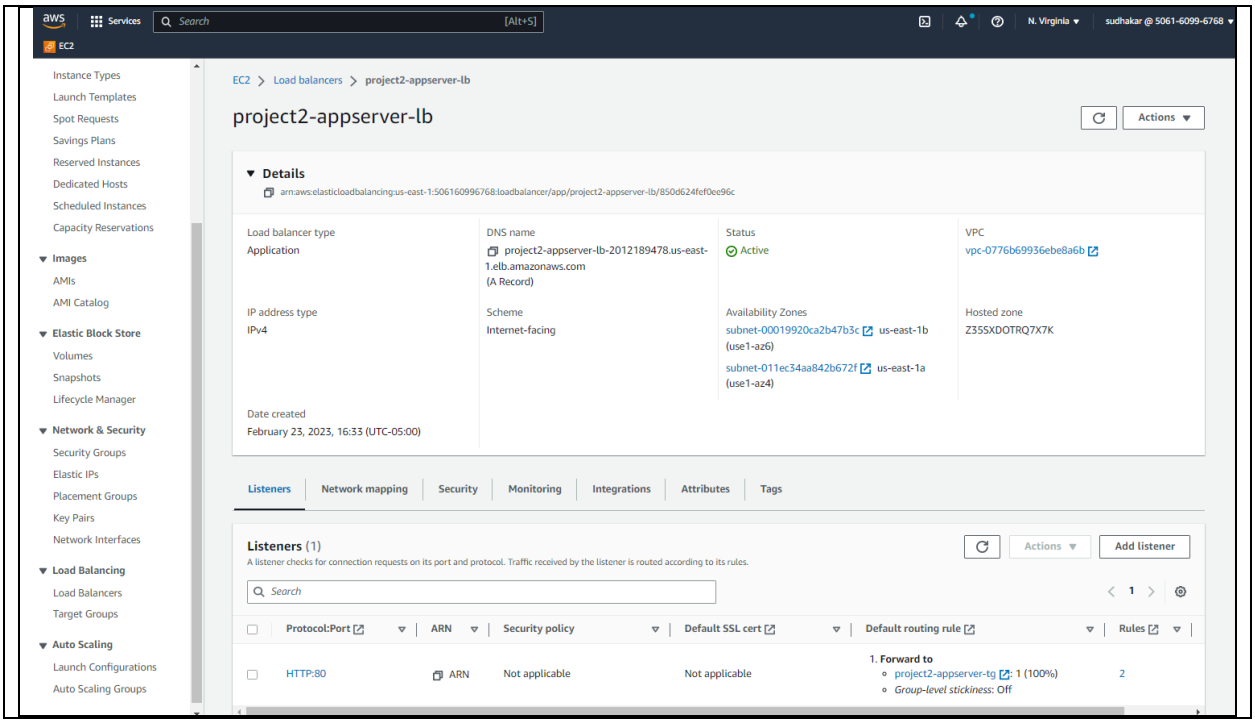
Create a load balancer in 2 Availability zones: project2-appserver-lb

Target Group Screenshot

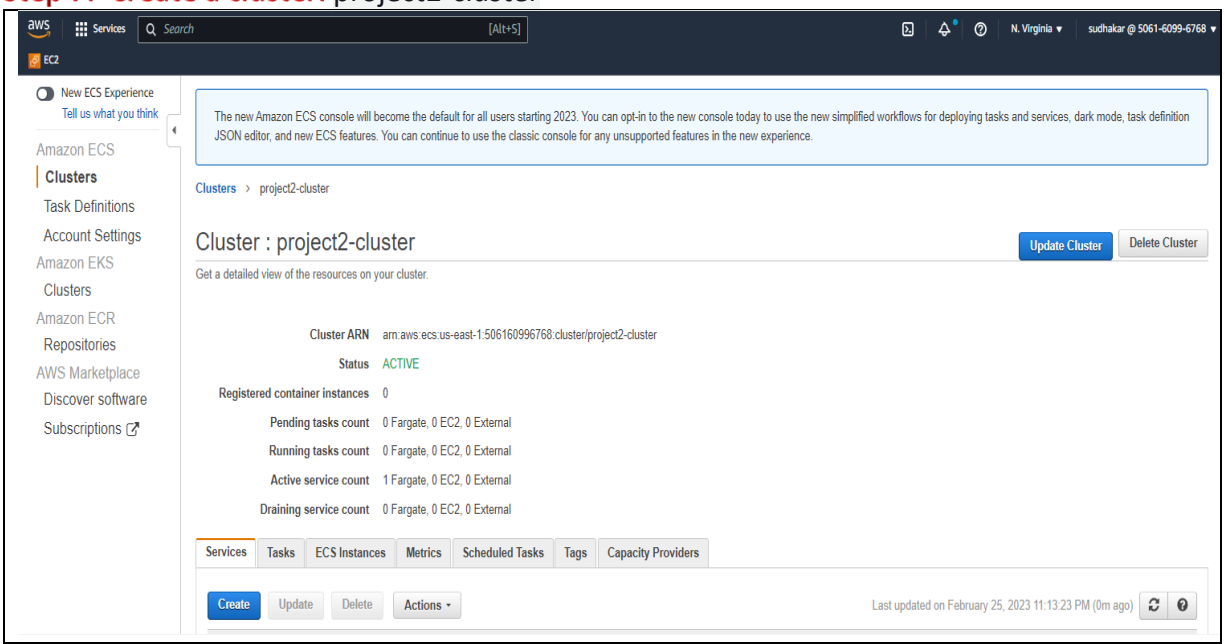


Load Balancer Screenshot





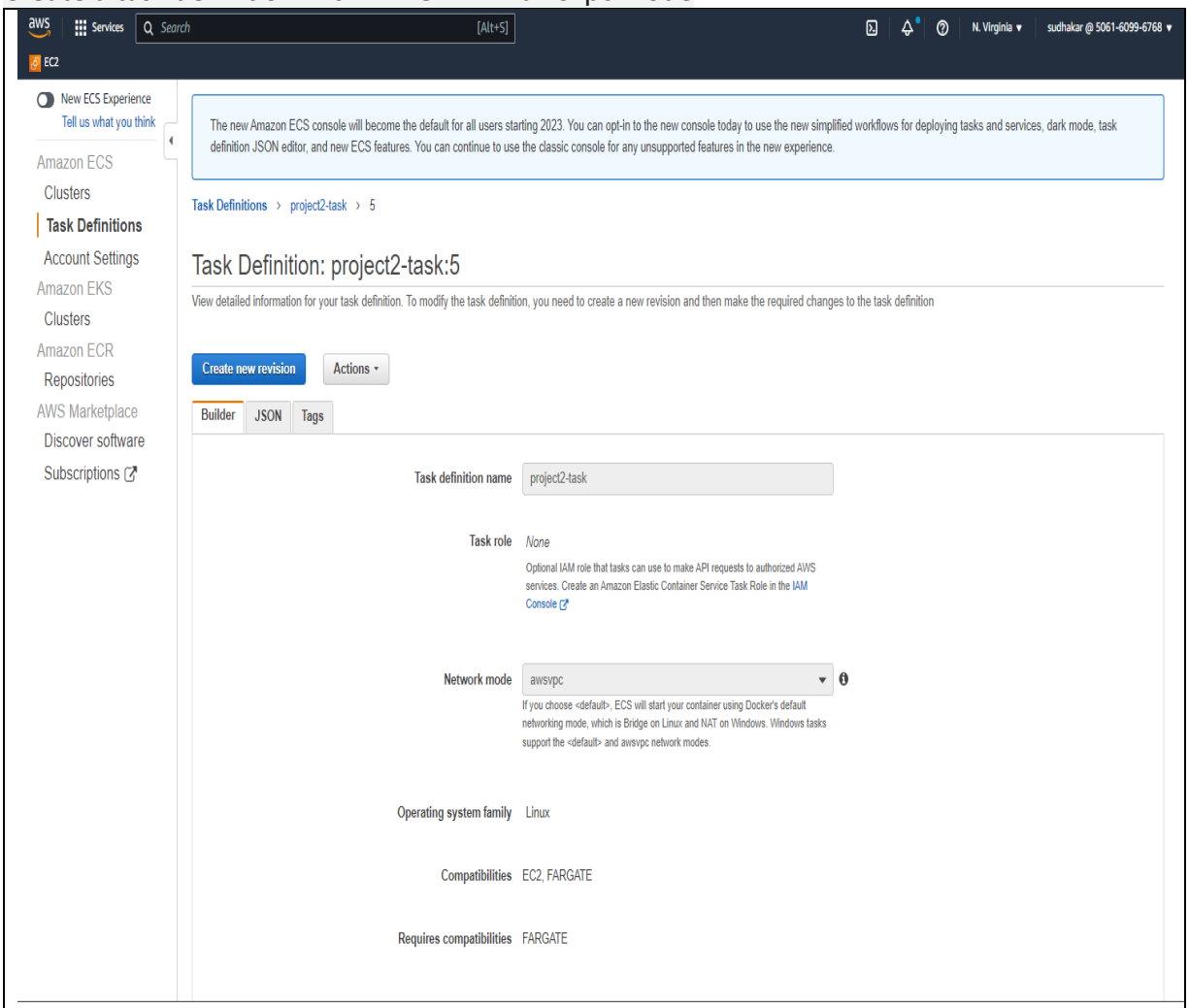
Step 7: Create a cluster: project2-cluster

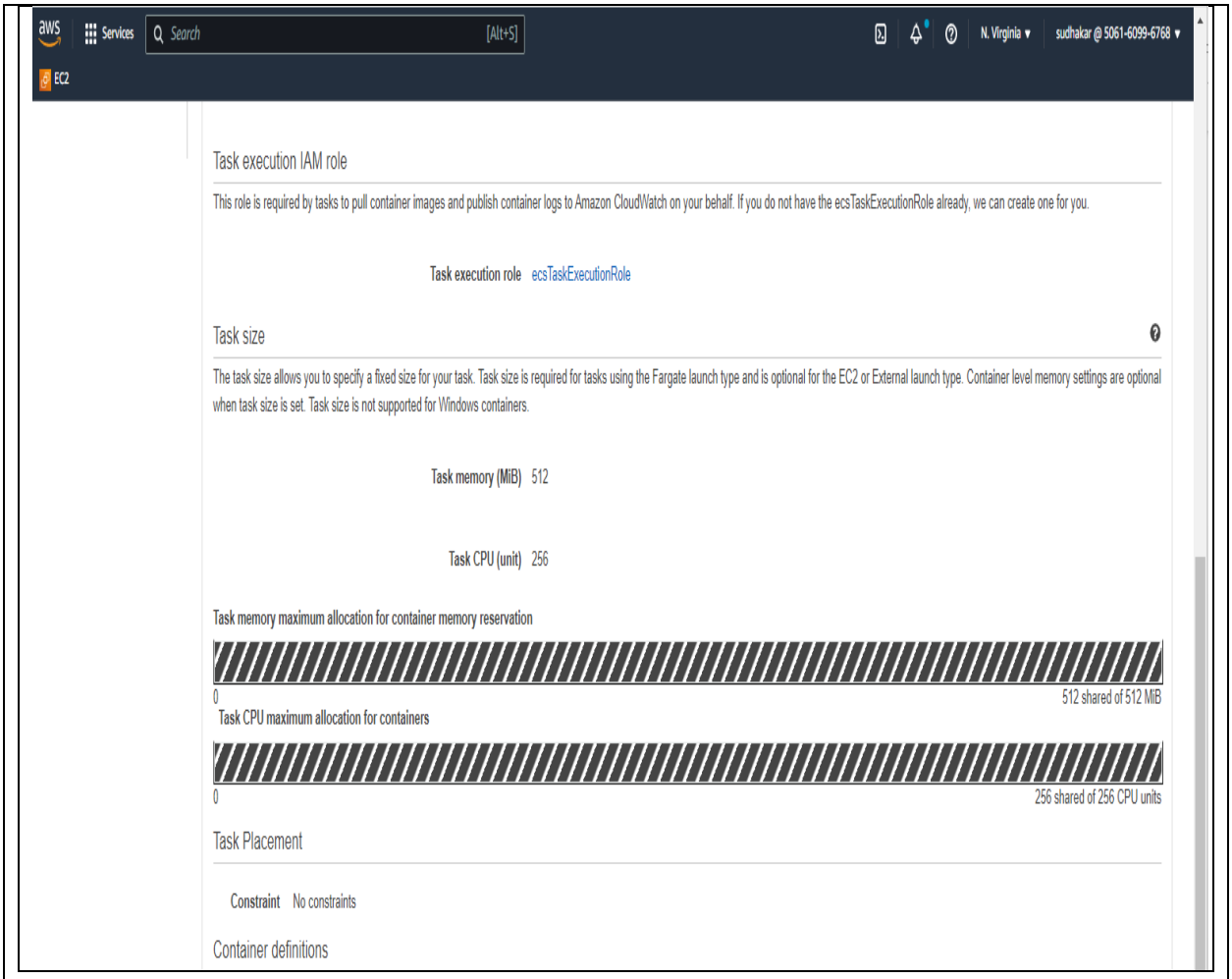


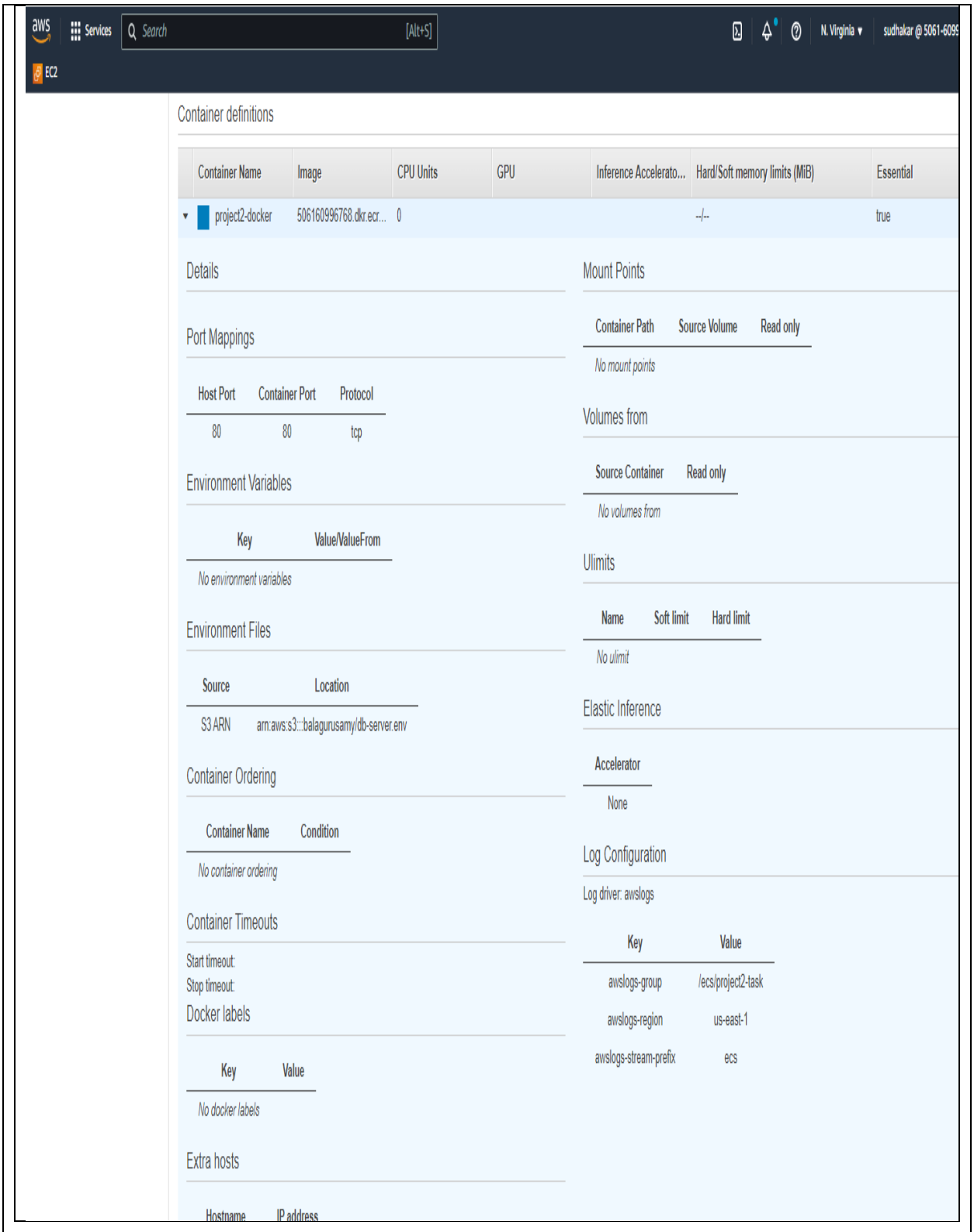
Step 8: Create a Task Definition: project2-task with

Create a Container definition : project2-docker with port mapping 80:80 with environment file

Create a task definition with FARGATE in awsvpc mode

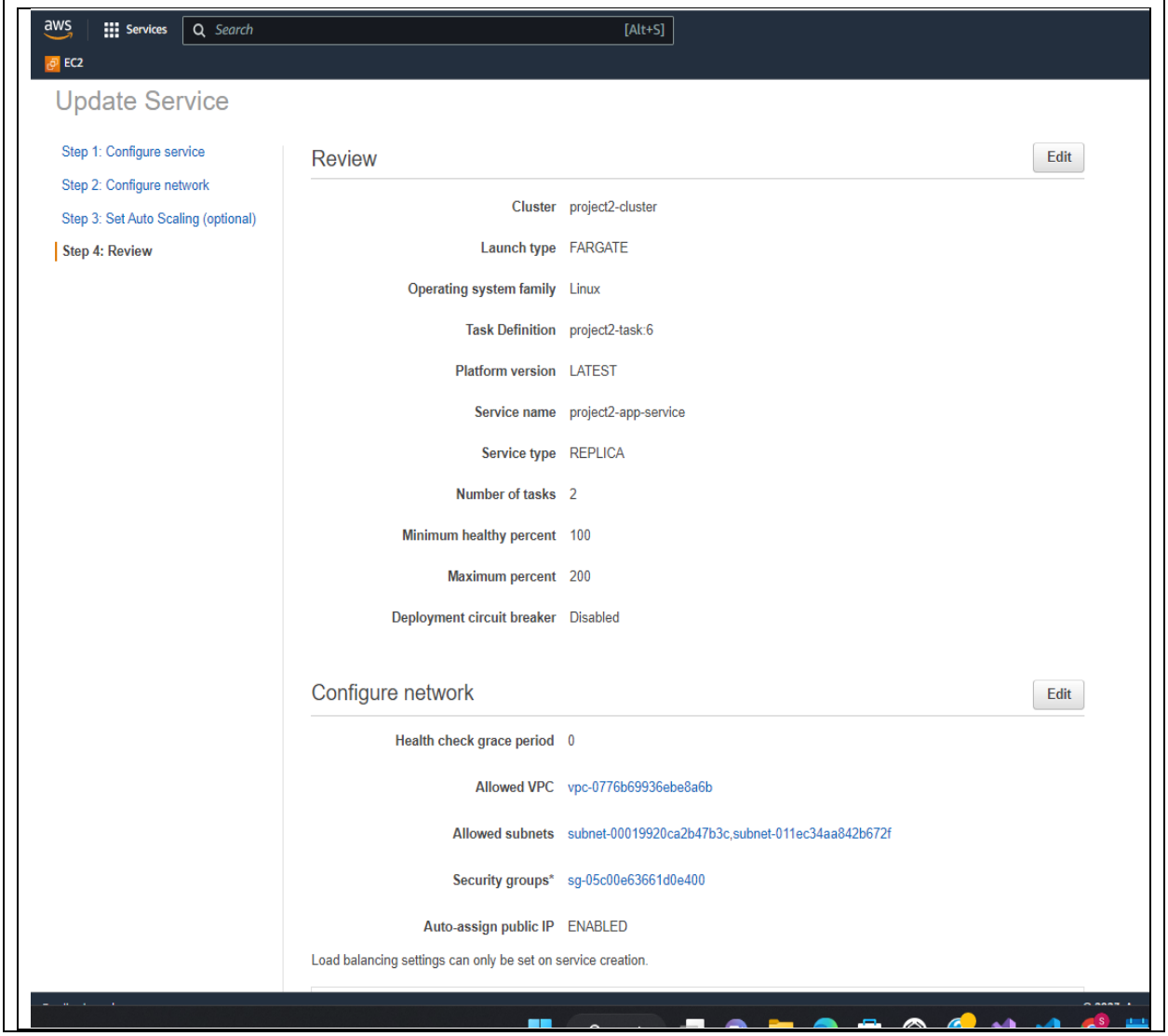



aws
Services
Search
[Alt+S]
N. Virginia
sudhakar @ 5061-6099-6768
EC2
Task execution IAM role
This role is required by tasks to pull container images and publish container logs to Amazon CloudWatch on your behalf. If you do not have the ecsTaskExecutionRole already, we can create one for you.
Task execution role ecsTaskExecutionRole
Task size
The task size allows you to specify a fixed size for your task. Task size is required for tasks using the Fargate launch type and is optional for the EC2 or External launch type. Container level memory settings are optional when task size is set. Task size is not supported for Windows containers.
Task memory (MiB) 512
Task CPU (unit) 256
Task memory maximum allocation for container memory reservation
0
512 shared of 512 MiB
Task CPU maximum allocation for containers
0
256 shared of 256 CPU units
Task Placement
Constraint No constraints
Container definitions


aws
Services
Search
[Alt+S]
N. Virginia
suchakar @ 5061-6099
EC2
Container definitions
Container Name
Image
CPU Units
GPU
Inference Accelerato...
Hard/Soft memory limits (MiB)
Essential
project2-docker
506160996768.dkr.ecr...
0
--/--
true
Details
Port Mappings
Host Port
Container Port
Protocol
80
80
tcp
Environment Variables
Key
Value/ValueFrom
No environment variables
Environment Files
Source
Location
S3 ARN
arn:aws:s3:::balagurusamy/db-server.env
Container Ordering
Container Name
Condition
No container ordering
Container Timeouts
Start timeout:
Stop timeout:
Docker labels
Key
Value
No docker labels
Extra hosts
Hostname
IP address
Mount Points
Container Path
Source Volume
Read only
No mount points
Volumes from
Source Container
Read only
No volumes from
Ulimits
Name
Soft limit
Hard limit
No ulimit
Elastic Inference
Accelerator
None
Log Configuration
Log driver: awslogs
Key
Value
awslogs-group
/ecs/project2-task
awslogs-region
us-east-1
awslogs-stream-prefix
ecs

Step 9: Create a Service in the cluster: project2-app-service

Number of Desired Tasks: 2



Minimum healthy percent 100

Maximum percent 200

Deployment circuit breaker Disabled

Configure network

Edit

Health check grace period 0

Allowed VPC vpc-0776b69936ebe8a6b

Allowed subnets subnet-00019920ca2b47b3c,subnet-011ec34aa842b672f

Security groups* sg-05c00e63661d0e400

Auto-assign public IP ENABLED

Load balancing settings can only be set on service creation.

Load Balancer Name project2-appserver-lb

Container Name: project2-docker

Container Port: 80

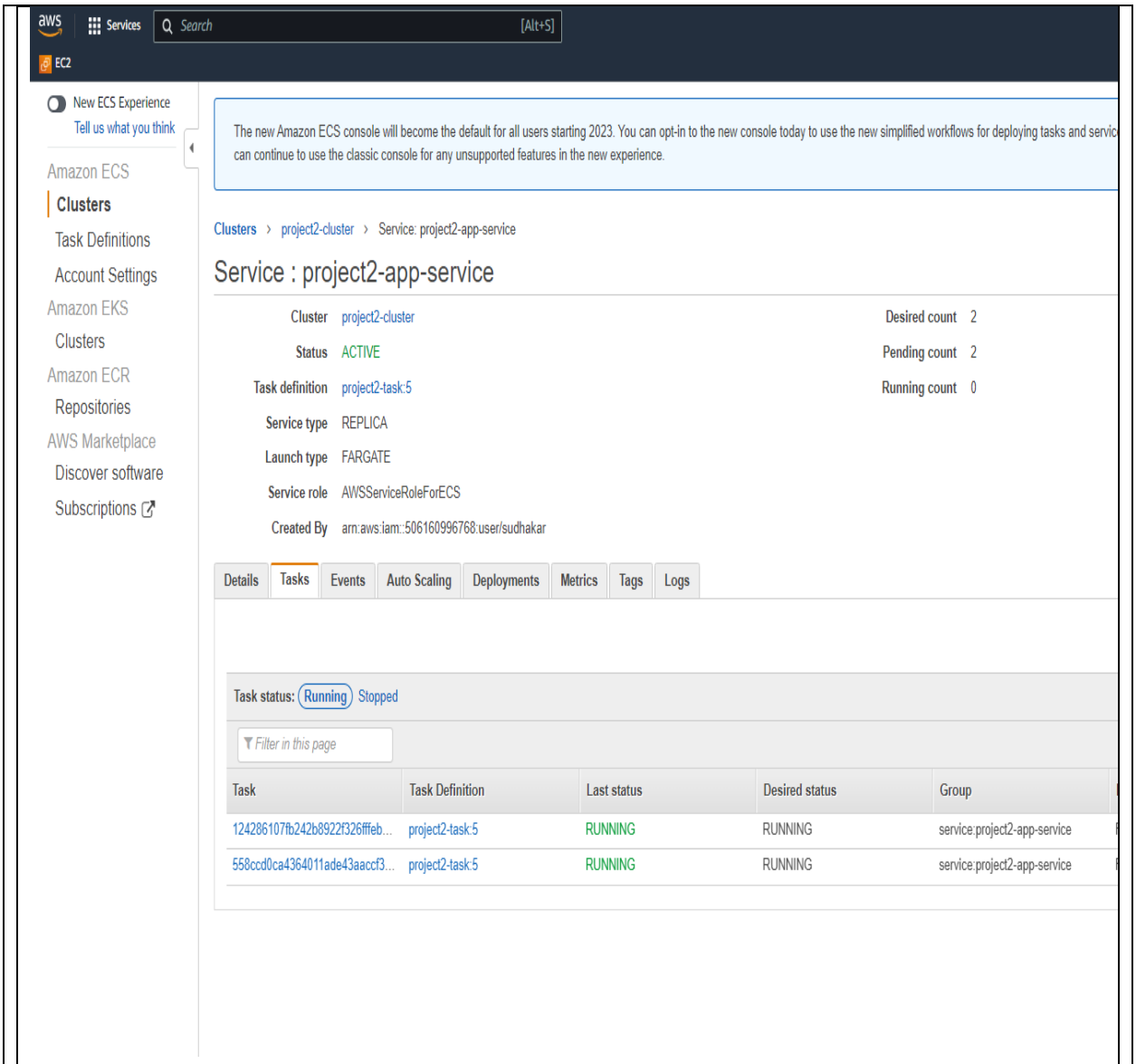
Target Group: arn:aws:elasticloadbalancing:us-east-1:506160996768:targetgroup/ecs-projec-project2-app-service/b6ac3f294a7e468b

Set Auto Scaling (optional)

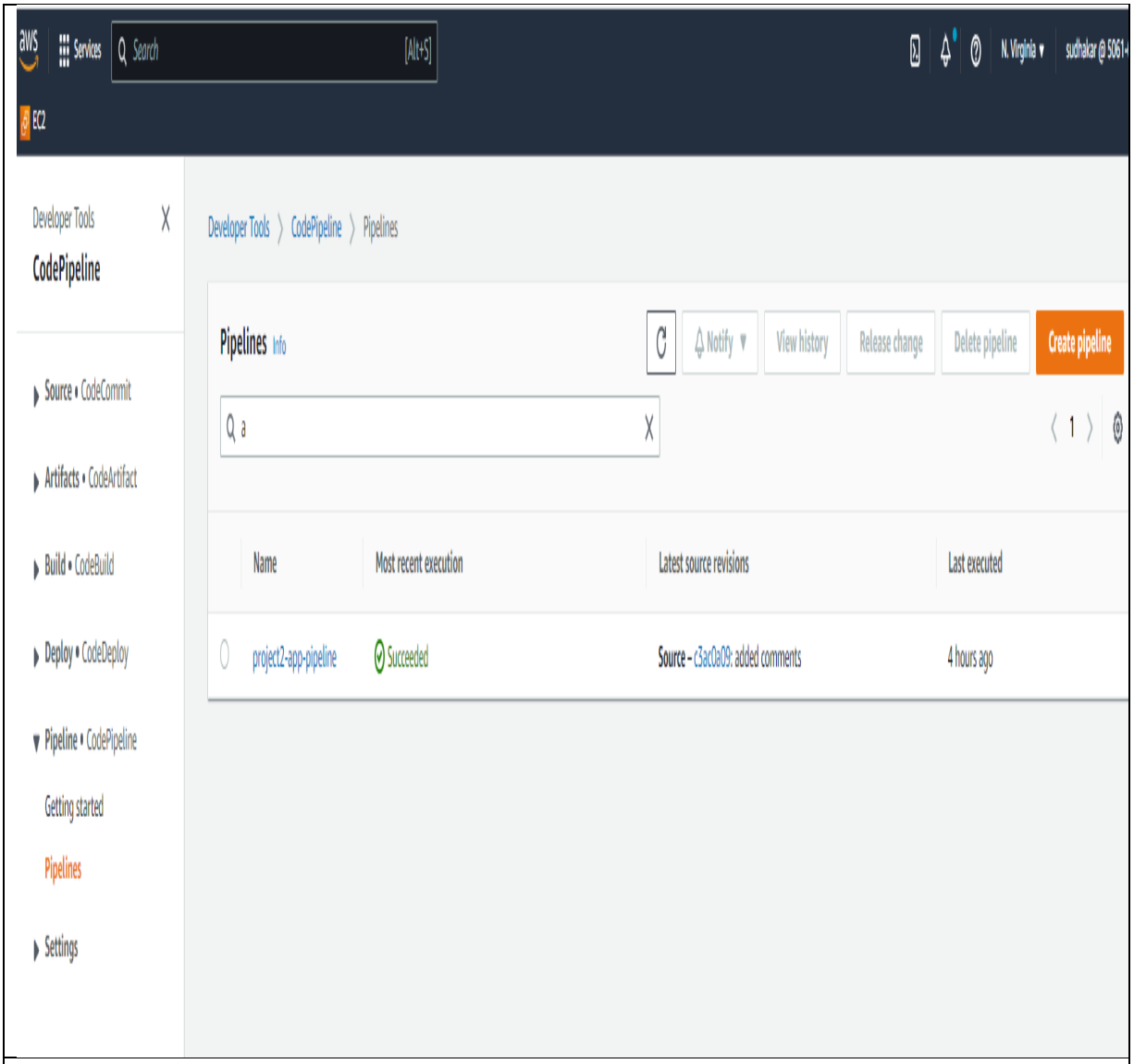
Edit

not configured

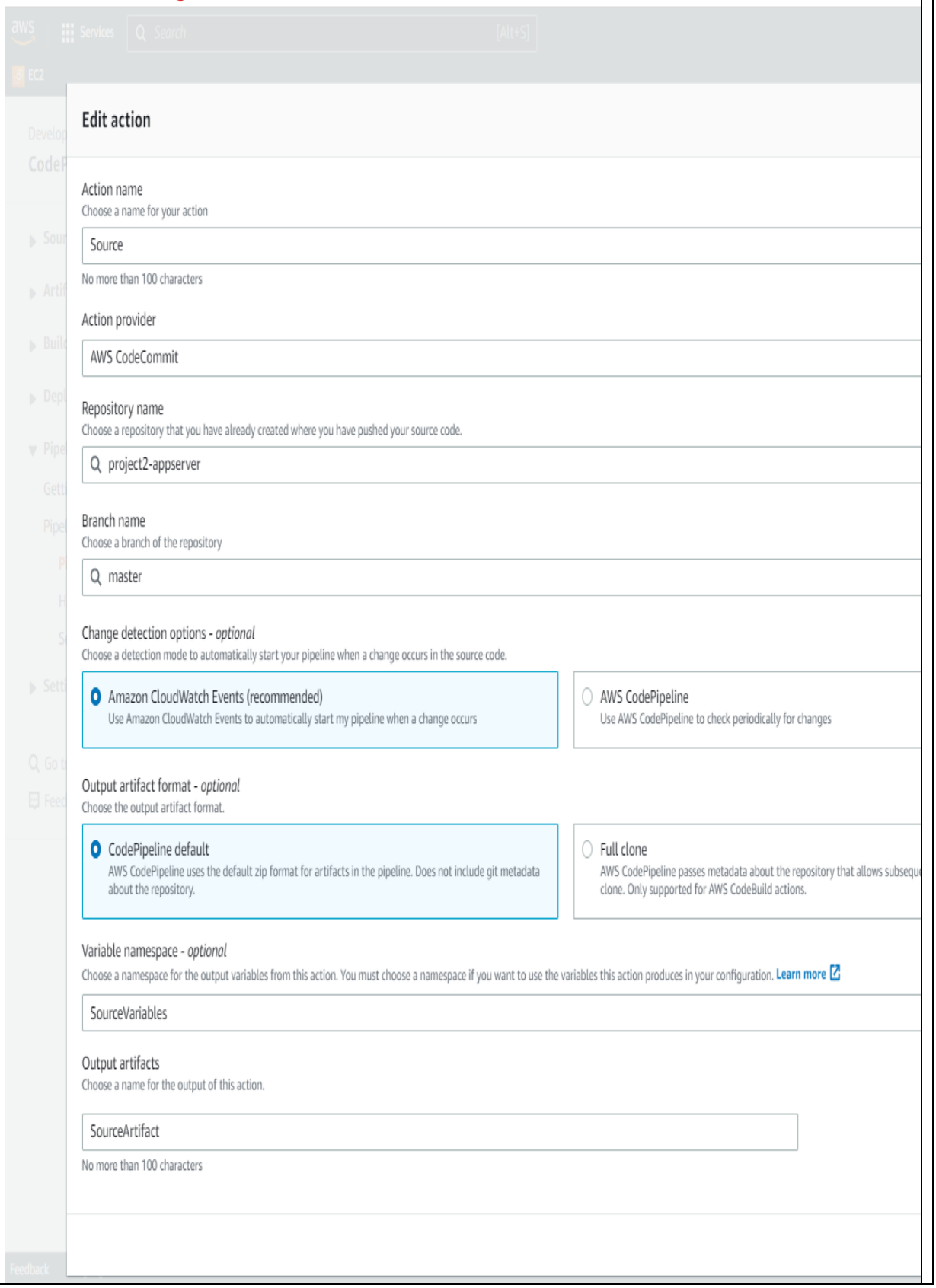
Cancel Previous Update Service


aws
Services
Search
[Alt+S]
EC2
New ECS Experience
Tell us what you think
Amazon ECS
Clusters
Task Definitions
Account Settings
Amazon EKS
Clusters
Amazon ECR
Repositories
AWS Marketplace
Discover software
Subscriptions
The new Amazon ECS console will become the default for all users starting 2023. You can opt-in to the new console today to use the new simplified workflows for deploying tasks and servic
can continue to use the classic console for any unsupported features in the new experience.
Clusters › project2-cluster › Service: project2-app-service
Service : project2-app-service
Cluster project2-cluster
Status ACTIVE
Task definition project2-task:5
Service type REPLICA
Launch type FARGATE
Service role AWSServiceRoleForECS
Created By arn:aws:iam::506160996768:user/sudhakar
Desired count 2
Pending count 2
Running count 0
Details
Tasks
Events
Auto Scaling
Deployments
Metrics
Tags
Logs
Task status: Running Stopped
Filter in this page
Task
Task Definition
Last status
Desired status
Group
124286107fb242b8922f326fffeb...
project2-task:5
RUNNING
RUNNING
service:project2-app-service
558ccd0ca4364011ade43aaccf3...
project2-task:5
RUNNING
RUNNING
service:project2-app-service

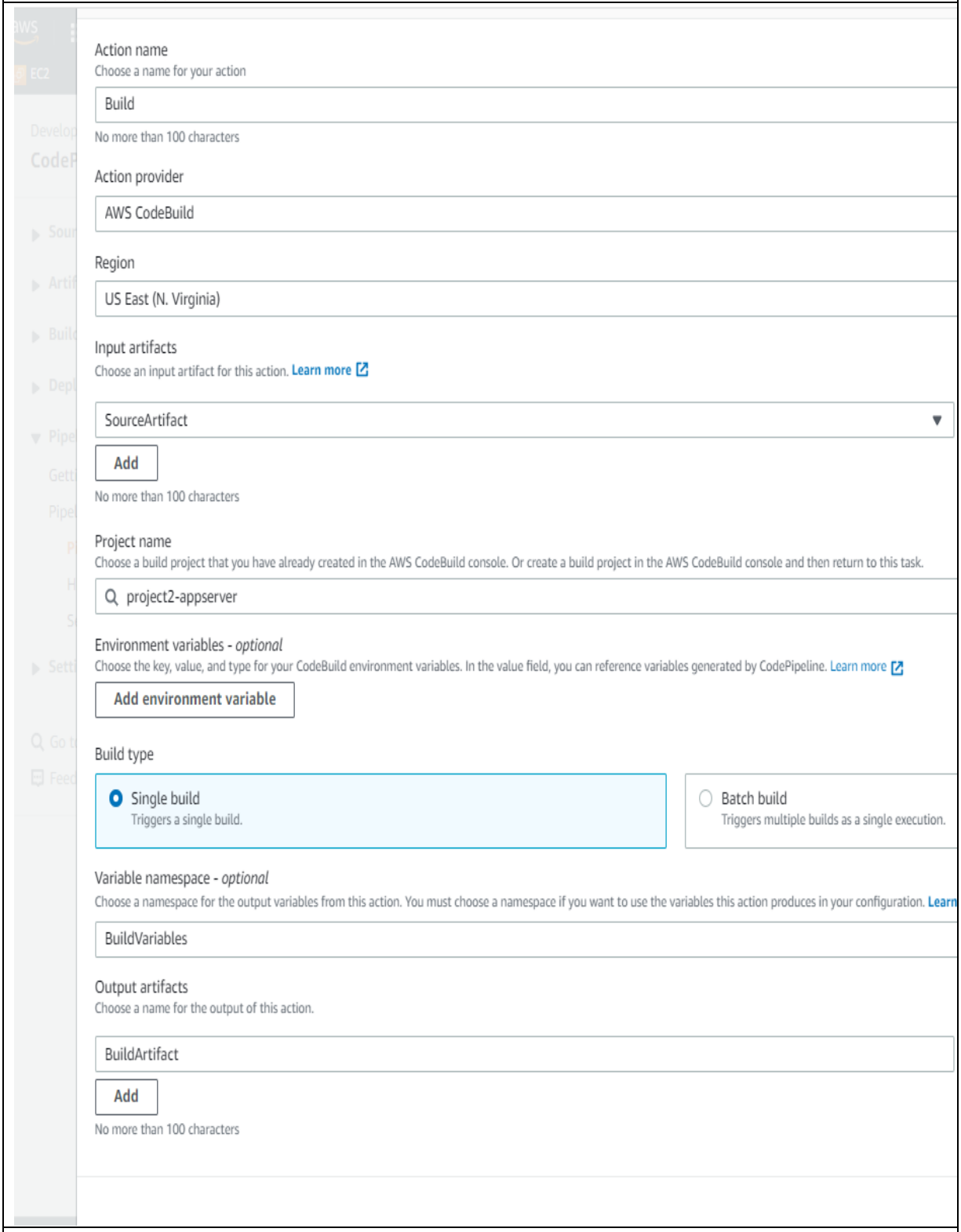
Step 10: Create a CodePipeline: project2-app-pipeline.



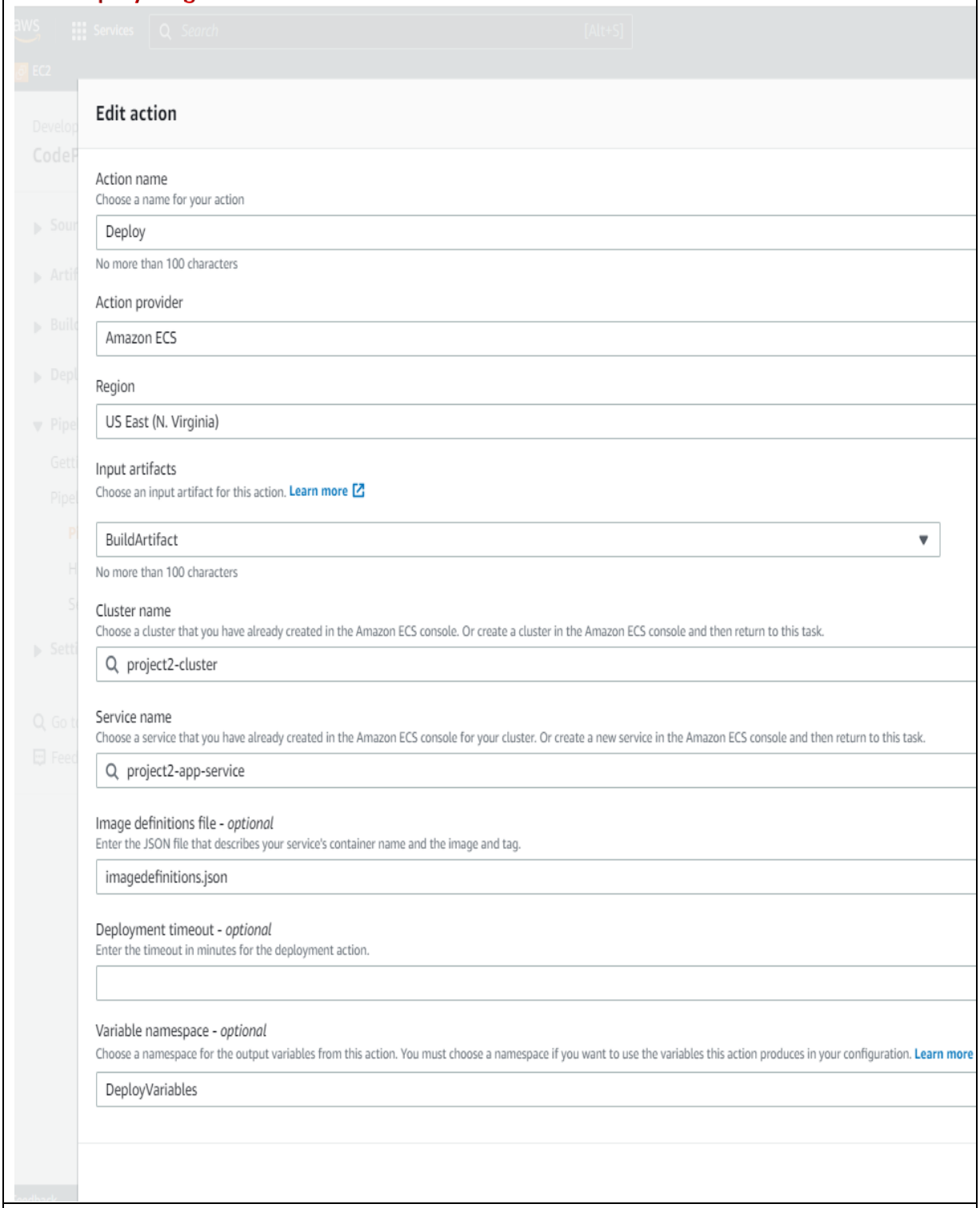
CodeCommit Stage



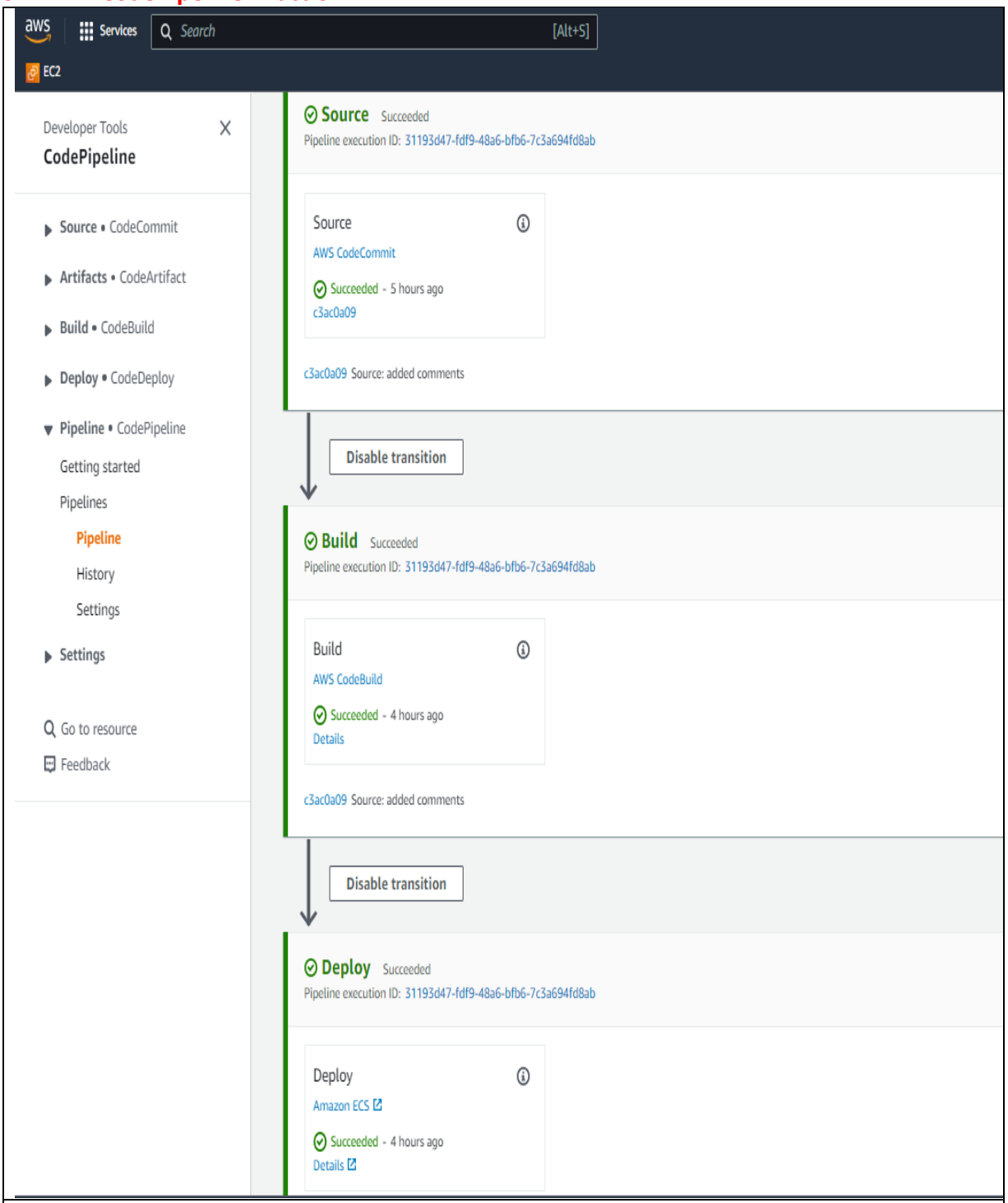
CodeBuild Stage



CodeDeploy Stage

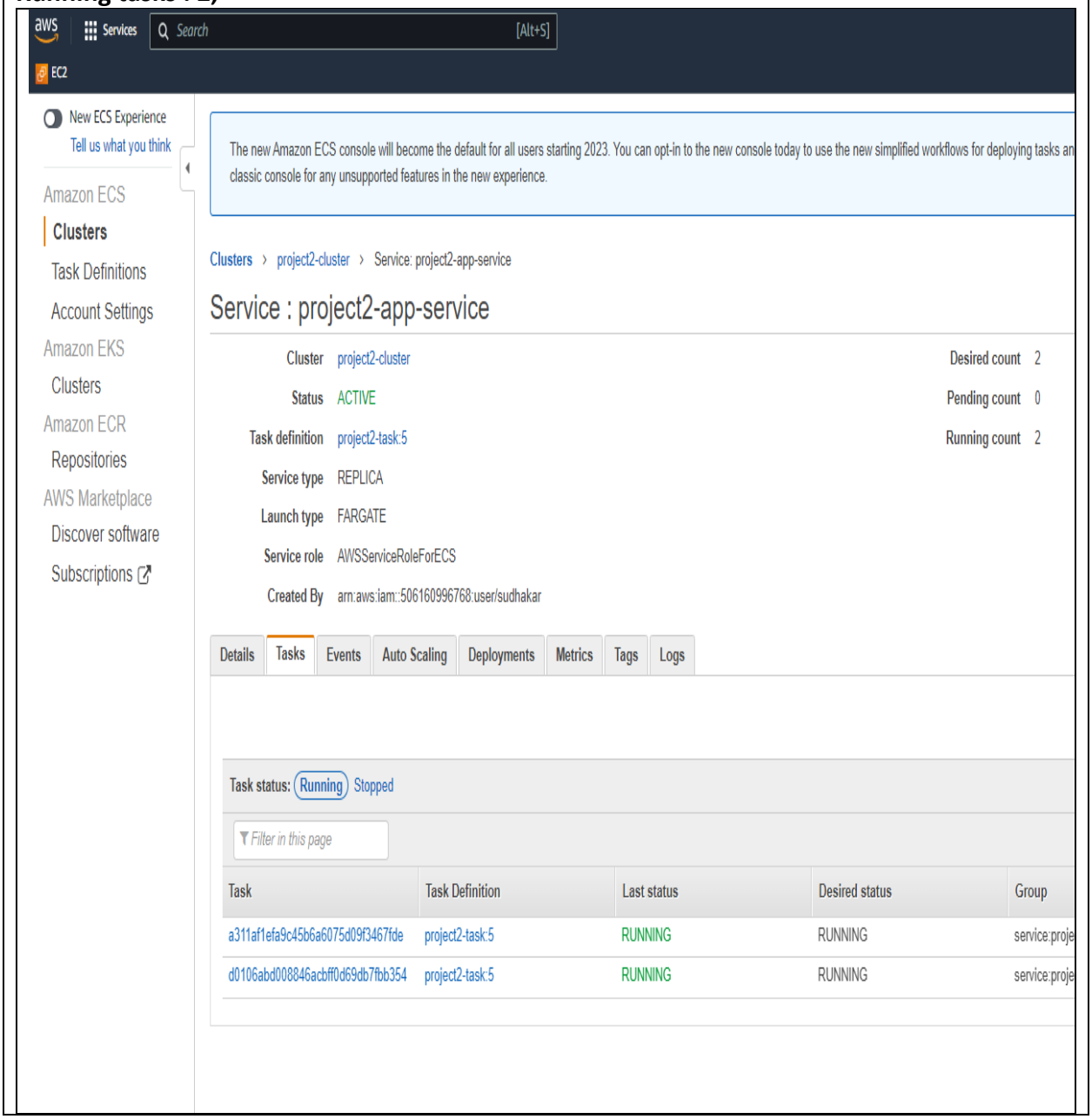


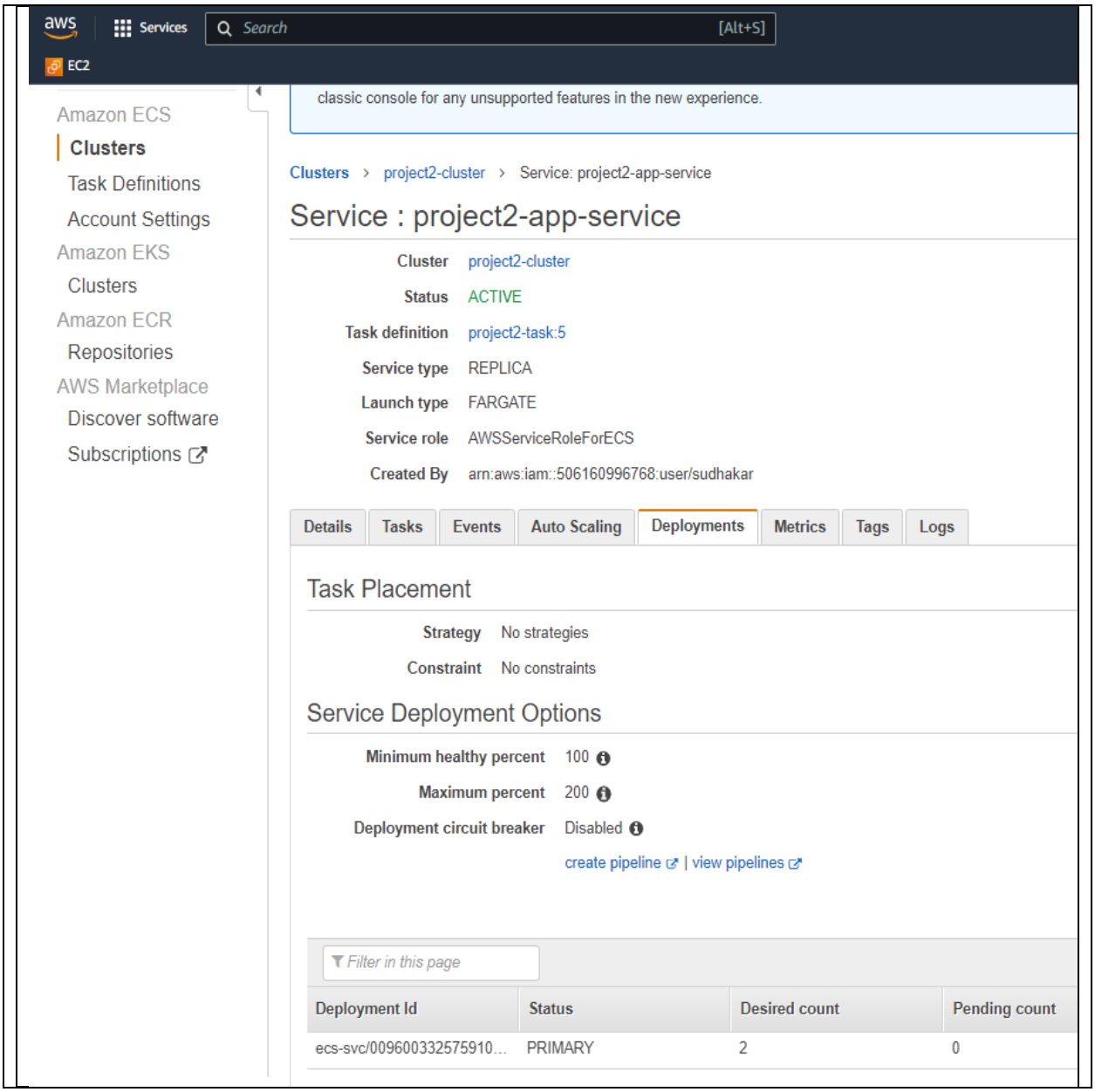
STEP 11: CodePipeline in action

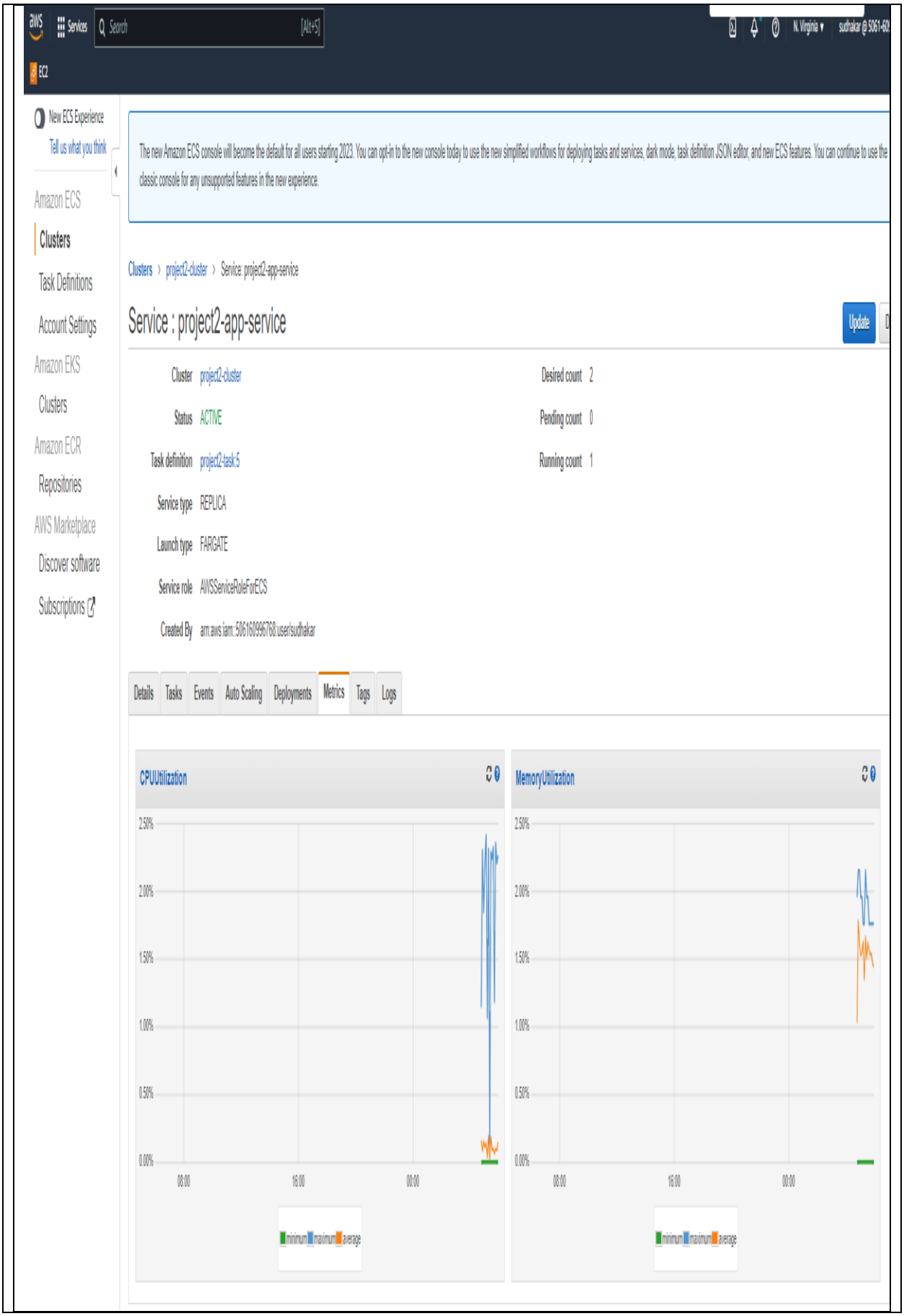


STEP 12: Cluster Deployment Status, Metrics

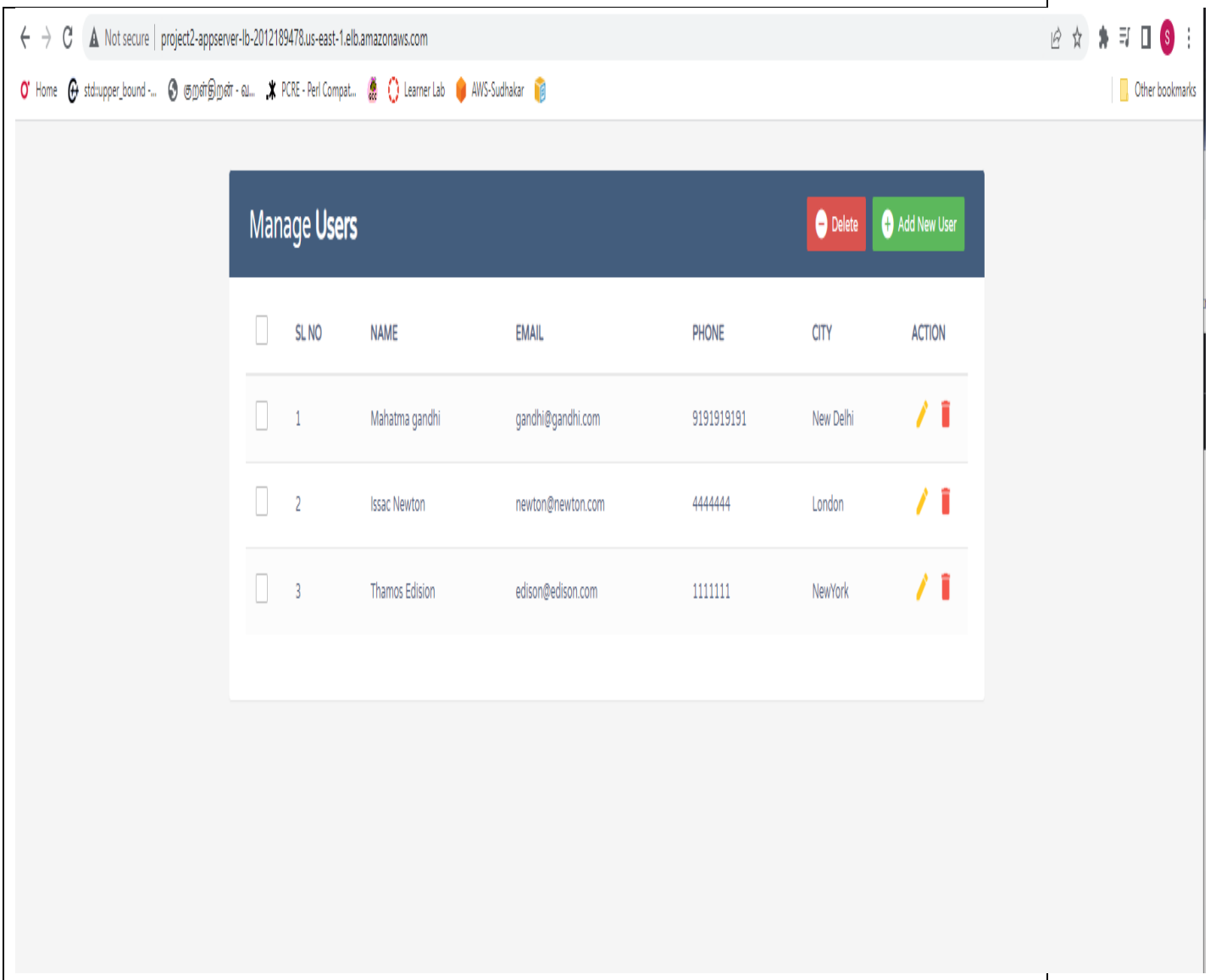
Running tasks : 2,



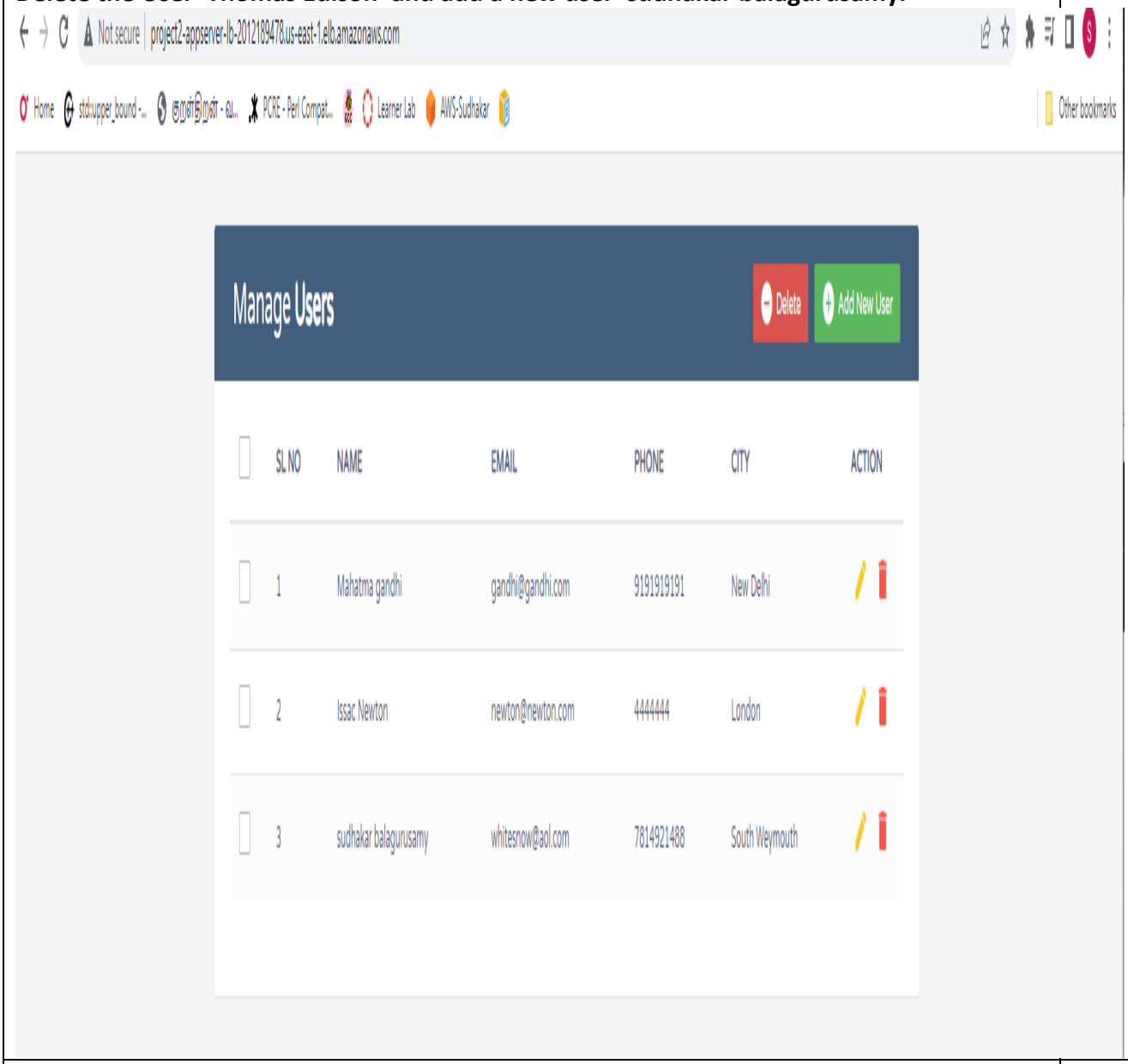

aws
Services
Search
[Alt+S]
EC2
Amazon ECS
Clusters
Task Definitions
Account Settings
Amazon EKS
Clusters
Amazon ECR
Repositories
AWS Marketplace
Discover software
Subscriptions
classic console for any unsupported features in the new experience.
Clusters › project2-cluster › Service: project2-app-service
Service : project2-app-service
Cluster project2-cluster
Status ACTIVE
Task definition project2-task:5
Service type REPLICA
Launch type FARGATE
Service role AWSServiceRoleForECS
Created By arn:aws:iam::506160996768:user/sudhakar
Details
Tasks
Events
Auto Scaling
Deployments
Metrics
Tags
Logs
Task Placement
Strategy No strategies
Constraint No constraints
Service Deployment Options
Minimum healthy percent 100
Maximum percent 200
Deployment circuit breaker Disabled
create pipeline | view pipelines
Filter in this page
Deployment Id
Status
Desired count
Pending count
ecs-svc/009600332575910...
PRIMARY
2
0


Services
Search
[Alt+S]
N. Virginia
EC2
New ECS Experience
Tell us what you think
Amazon ECS
Clusters
Task Definitions
Account Settings
Amazon EKS
Clusters
Amazon ECR
Repositories
AWS Marketplace
Discover software
Subscriptions
The new Amazon ECS console will become the default for all users starting 2023. You can opt-in to the new console today to use the new simplified workflows for deploying tasks and services, dark mode, task definition JSON editor, and new ECS features. You can continue to use the classic console for any unsupported features in the new experience.
Clusters › project2-cluster › Service: project2-app-service
Service : project2-app-service
Update
Cluster project2-cluster
Status ACTIVE
Task definition project2-task:5
Service type REPLICA
Launch type FARGATE
Service role AWSServiceRoleForECS
Created By arn:aws:iam::506160896768:user/sudhakar
Desired count 2
Pending count 0
Running count 1
Details
Tasks
Events
Auto Scaling
Deployments
Metrics
Tags
Logs
CPUUtilization
MemoryUtilization
2.50%
2.00%
1.50%
1.00%
0.50%
0.00%
08:00
16:00
00:00
minimum
maximum
average

STEP 12: Open the Data Entry Application using the load balancer URL

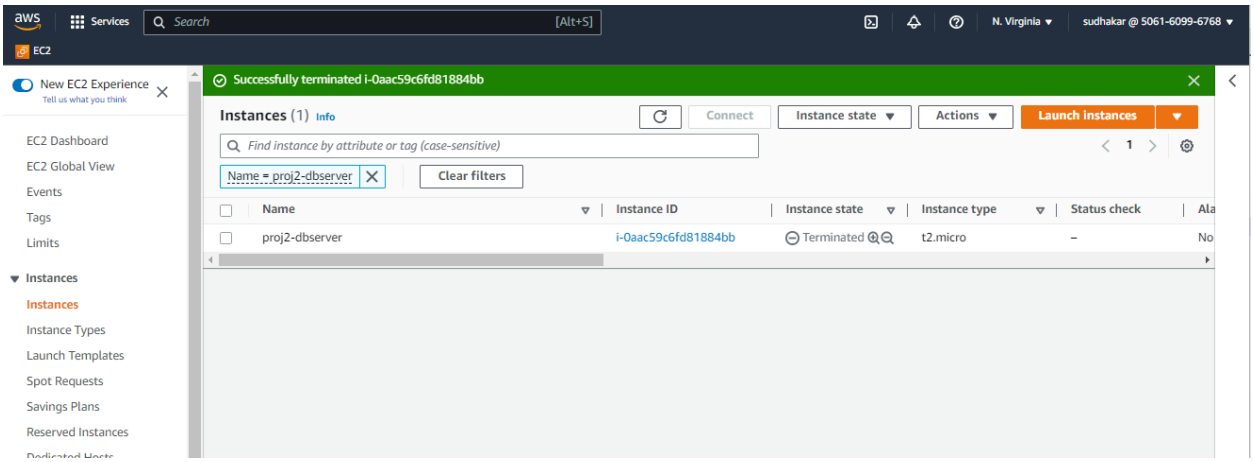


Delete the User ‘Thomas Edison’ and add a new user ‘sudhakar balagurusamy.’

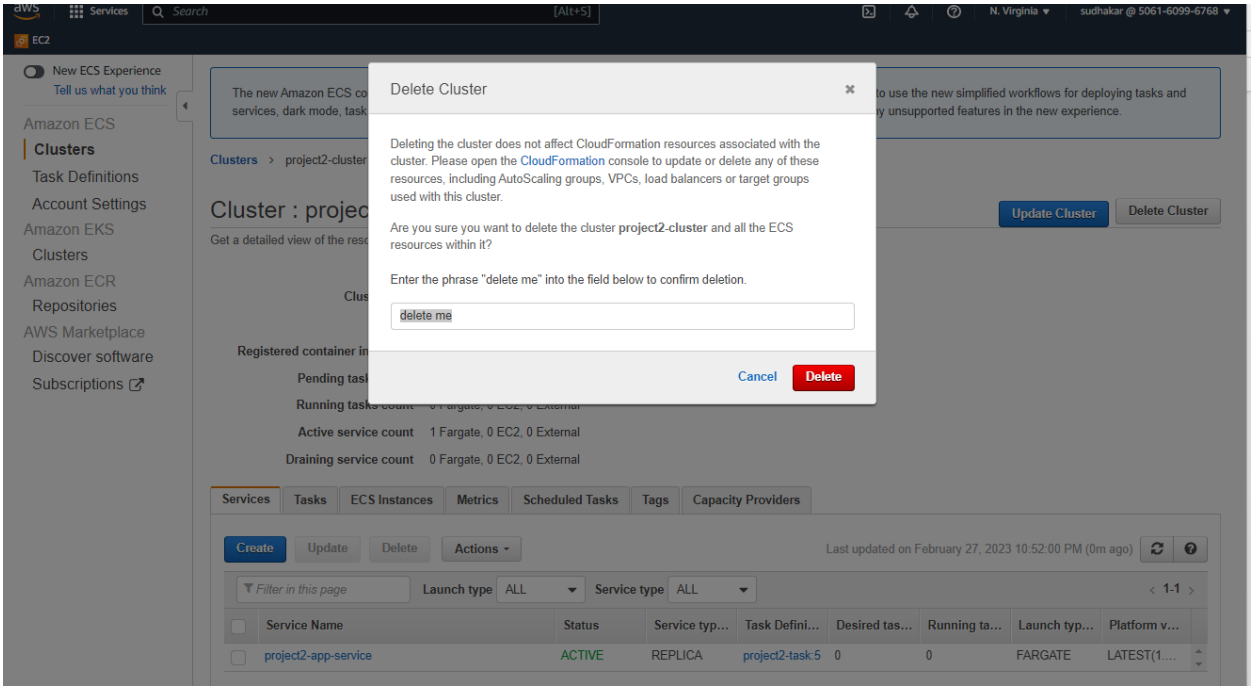


Cleanup of Resources

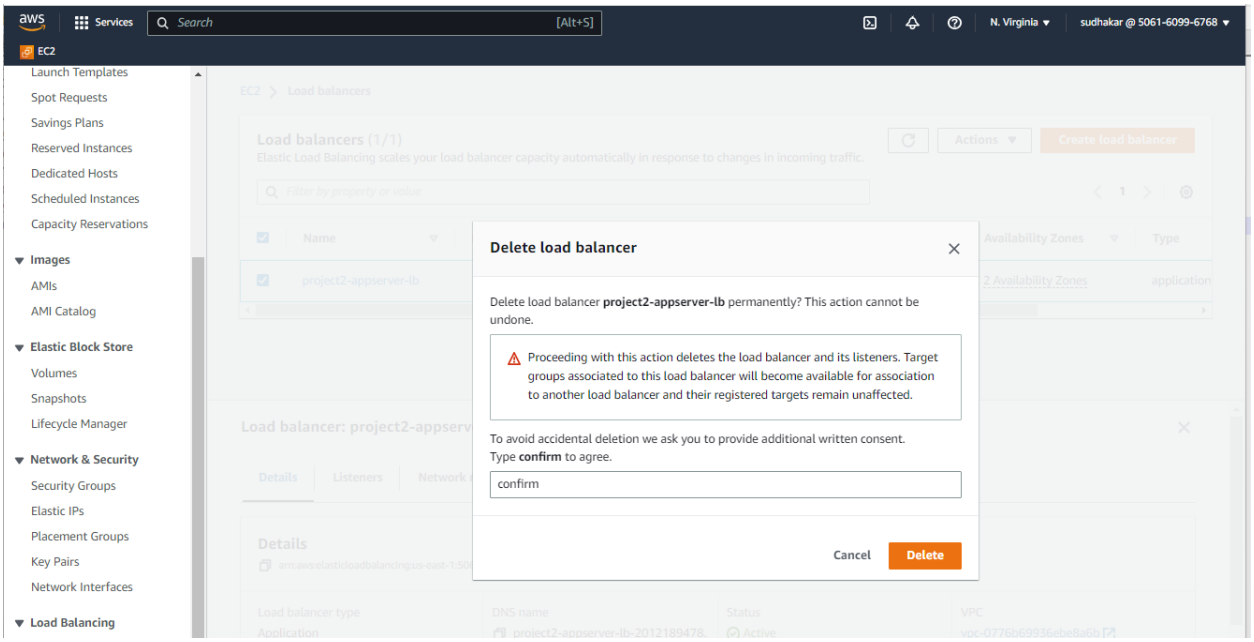
Terminate the DB Server EC2 instance proj2-dbserver



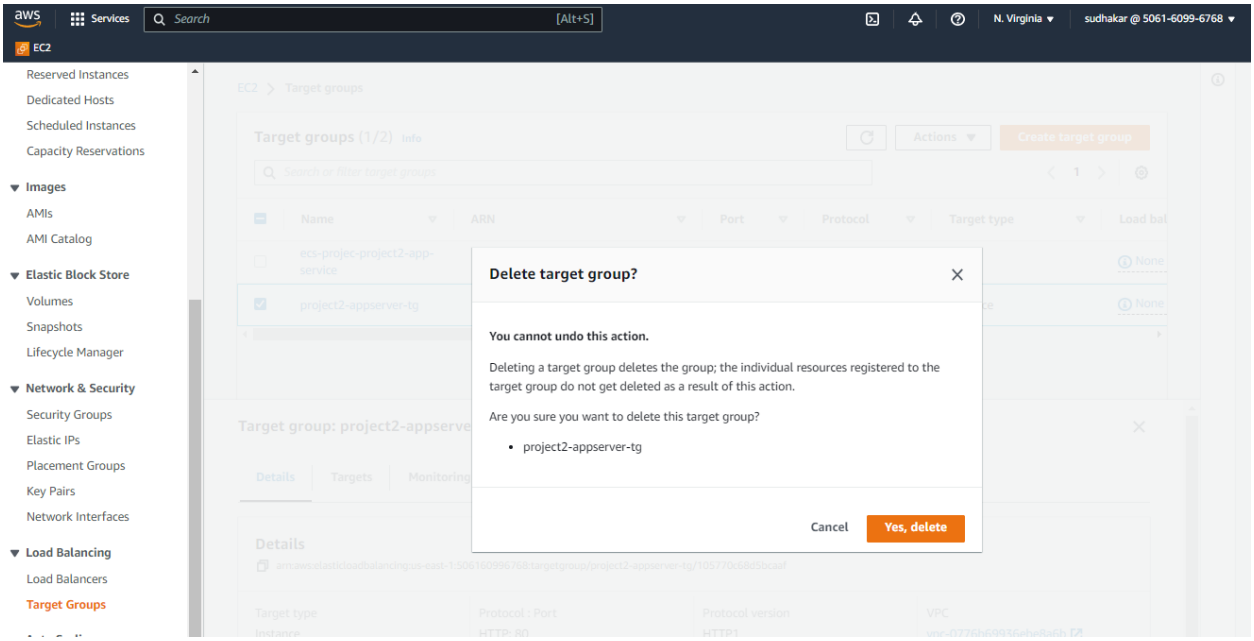
Delete the ECS cluster: project2-cluster.



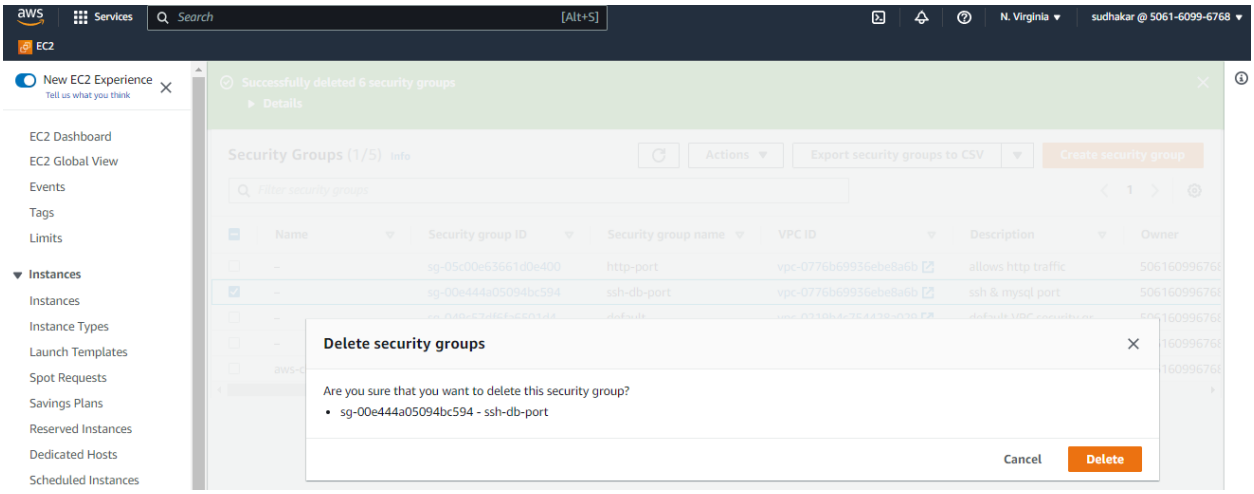
Delete the load balancer.



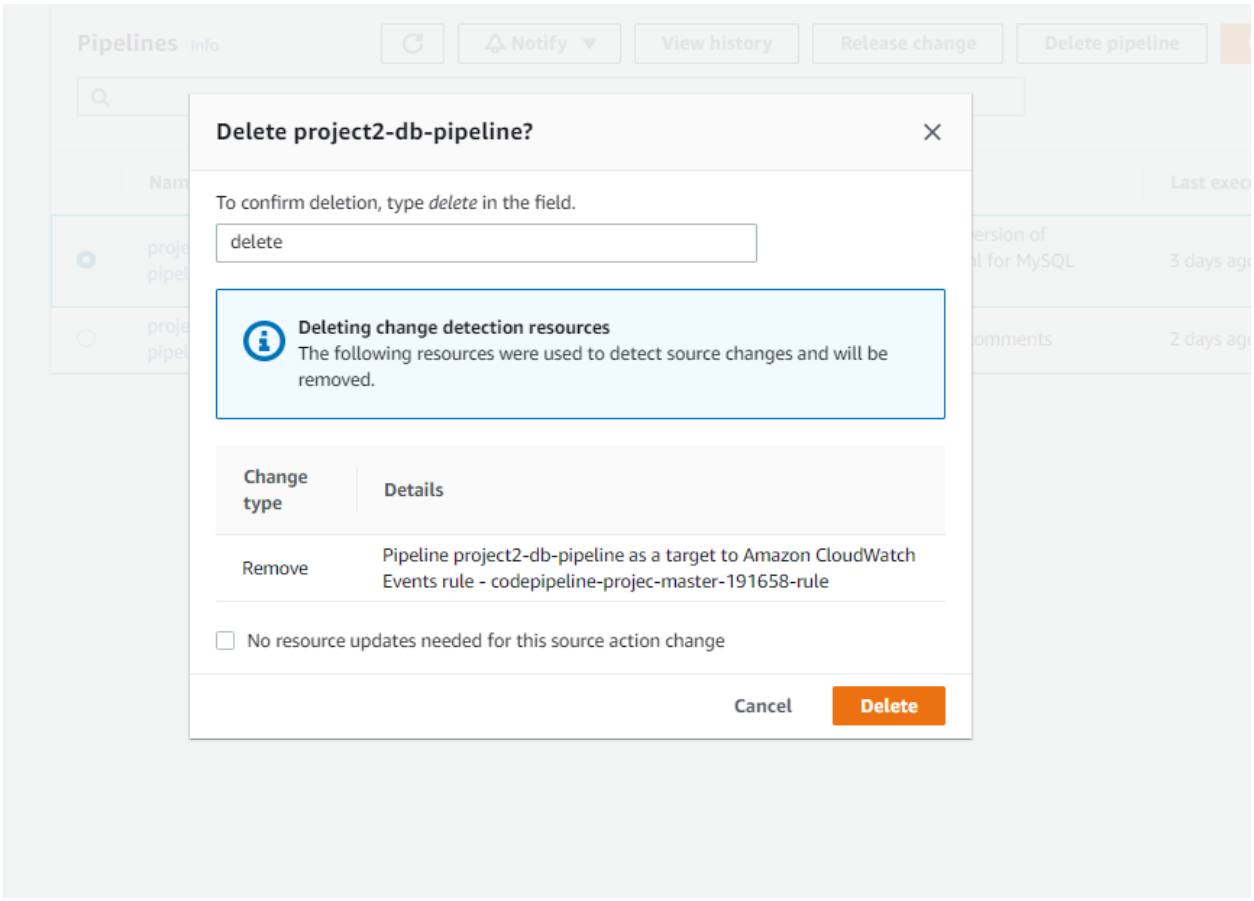
Delete the target group



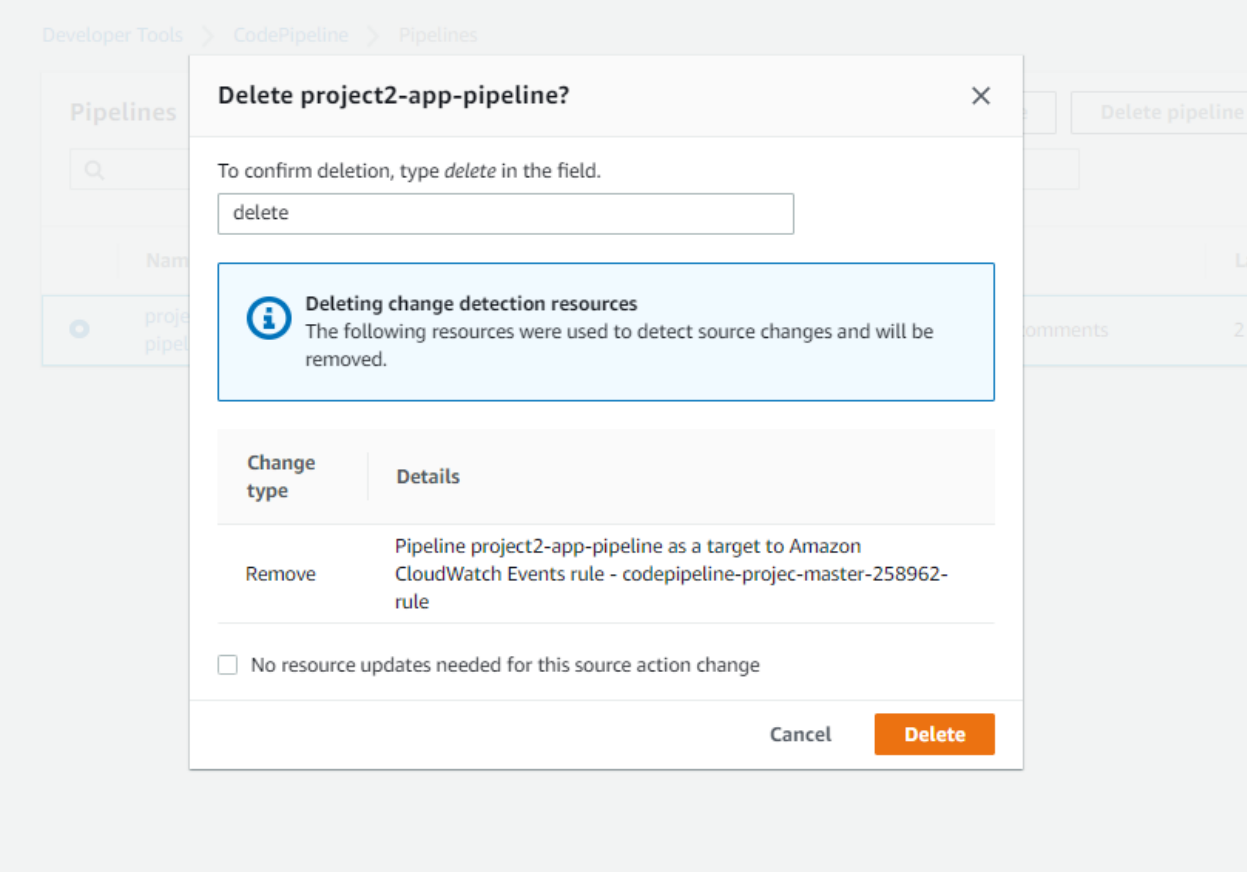
Delete the security group



Delete the DB-sever pipeline



Delete the app-server pipeline



Delete the ECR repositories project2-appserver, dbserver