ECS Fargate Blue-Green Deployment using Code pipeline

Objective:

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

Resources

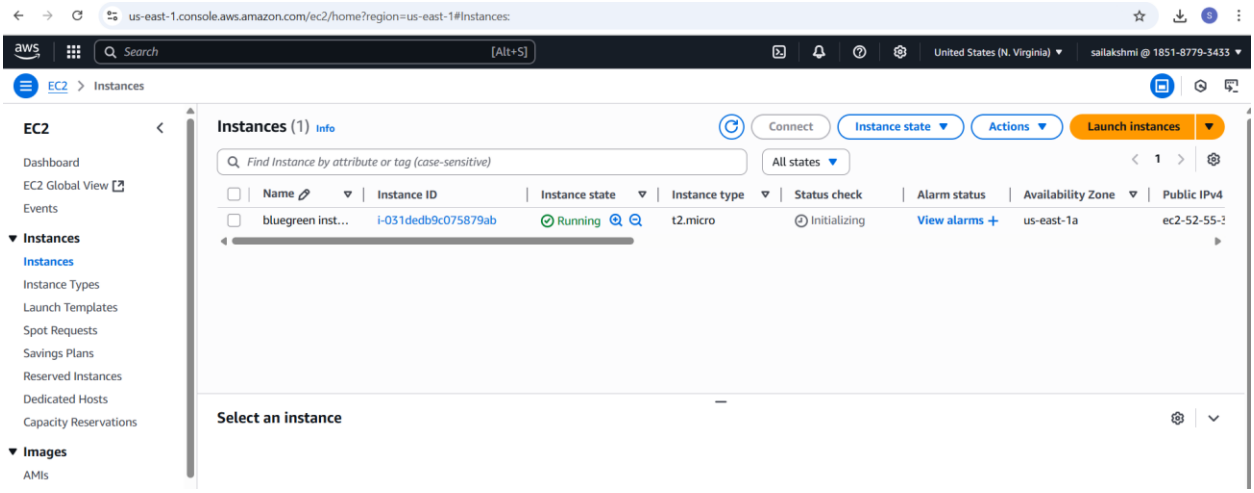
- ECS Cluster
- Fargate Task Definition & Service (with deployment_controller as CODE_DEPLOY)
- ECR Repository
- CodePipeline with CodeBuild and CodeDeploy stages
- IAM Roles
- ALB with target groups for blue/green

How Blue/Green Deployment Works

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

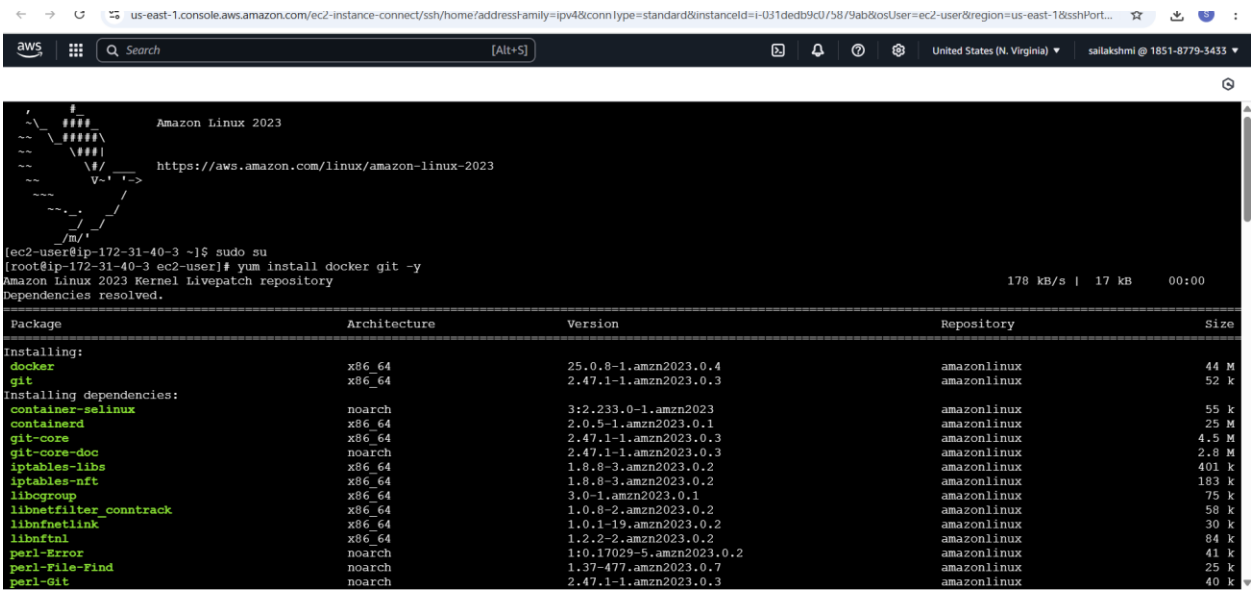
Step-by-Step Deployment

Step:1 Create an ec2 instance

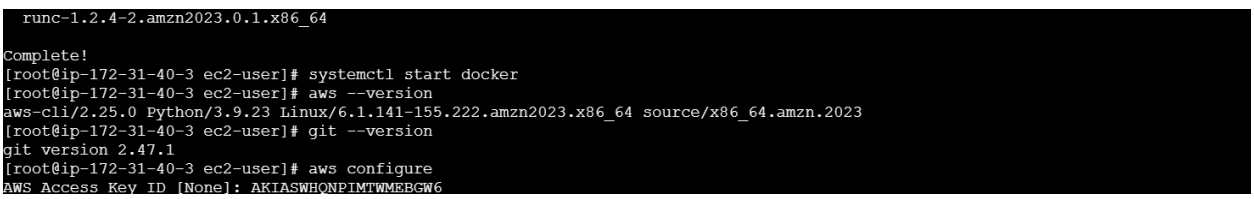


Connect to ec2 instance

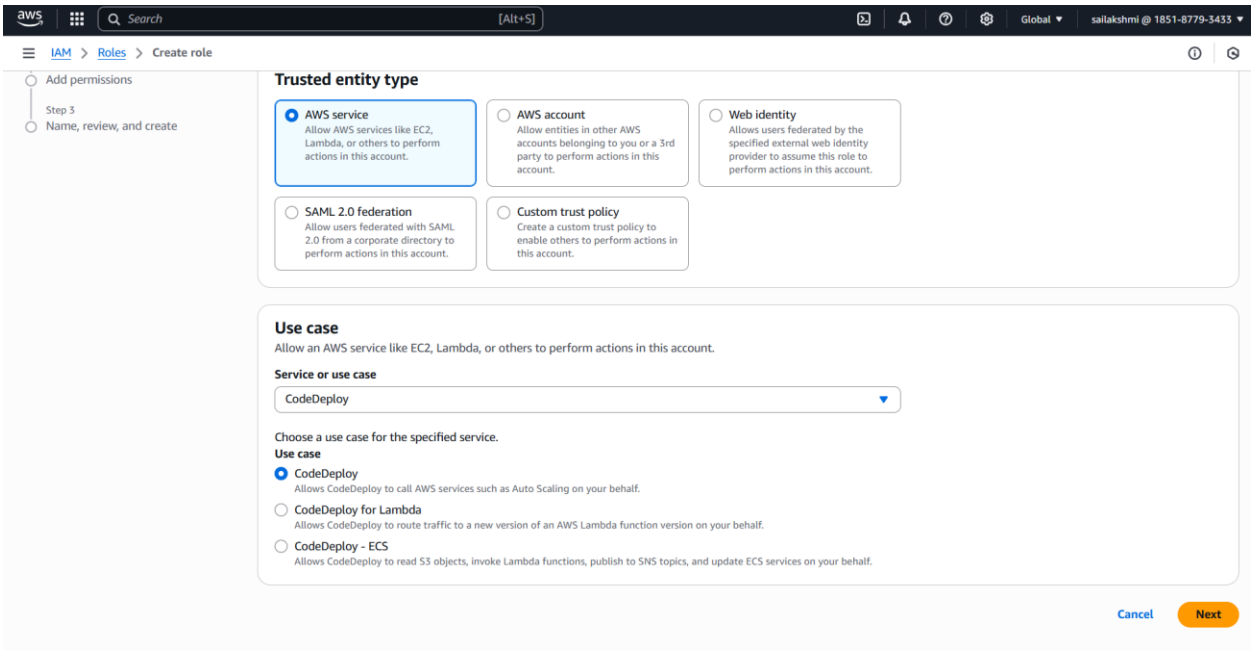
Install docker in it

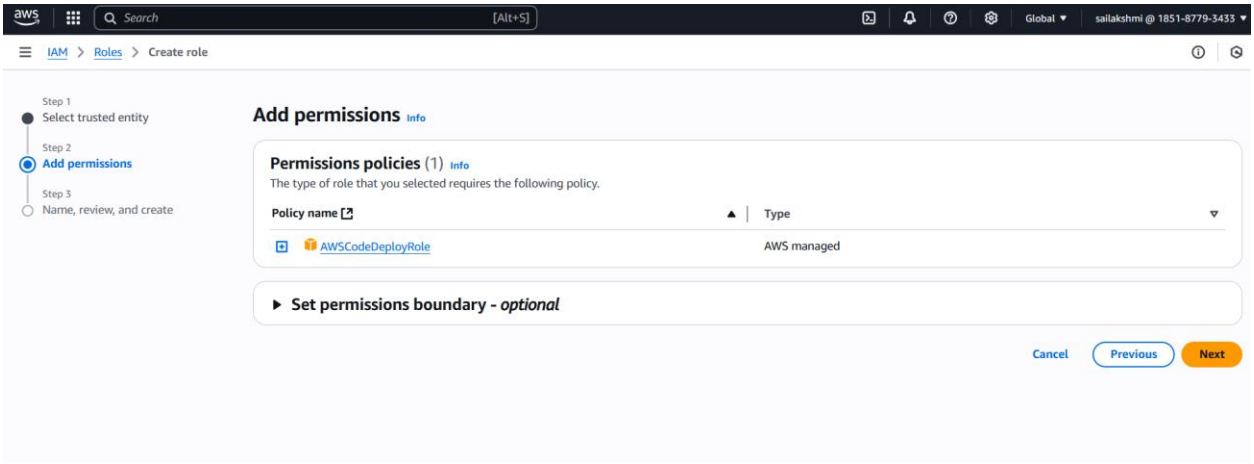


Start docker

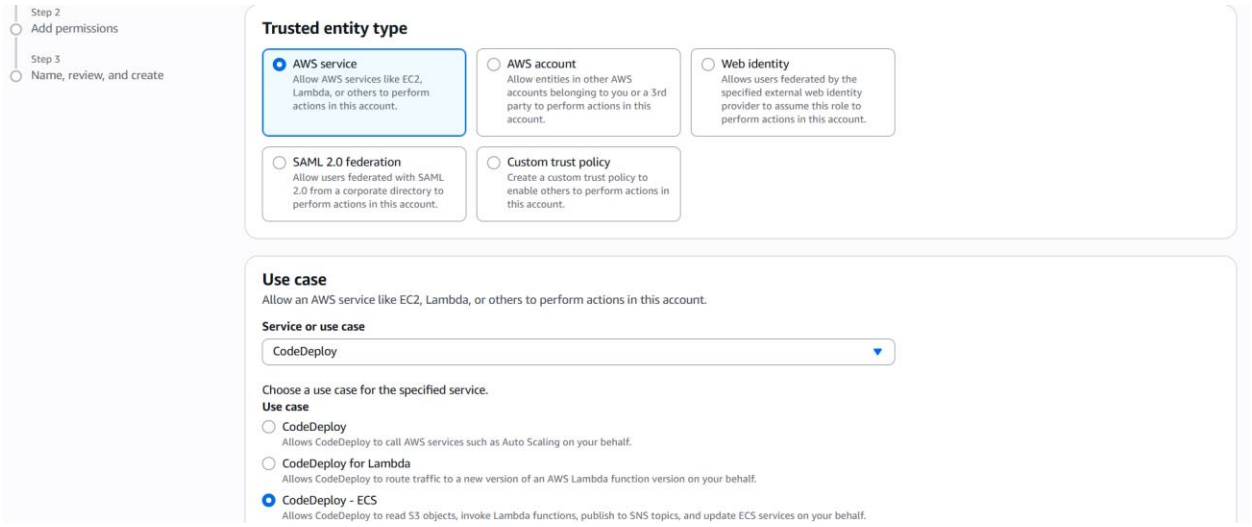


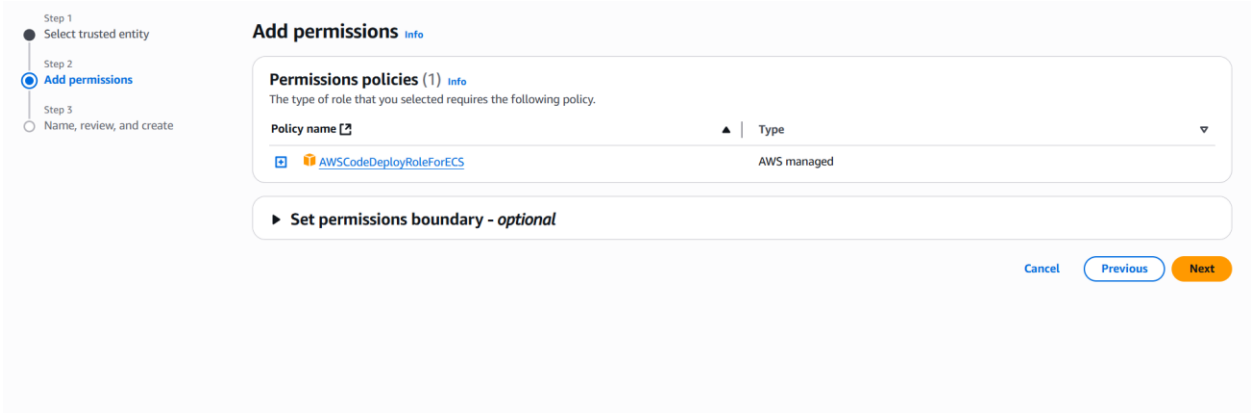
Step:2 Create a IAM role for code deploy, ECS

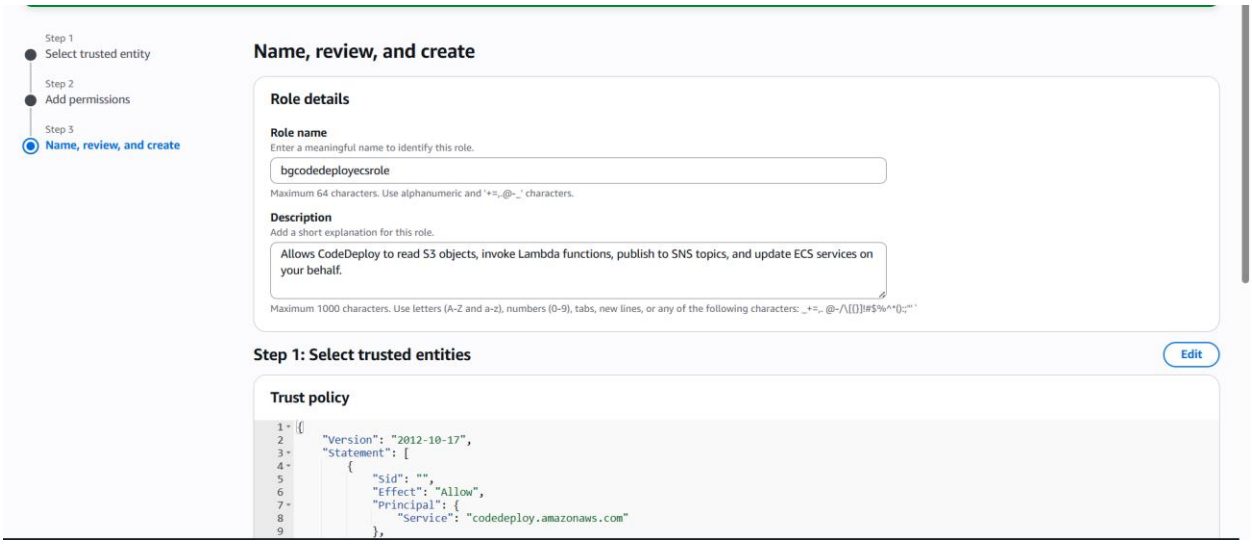




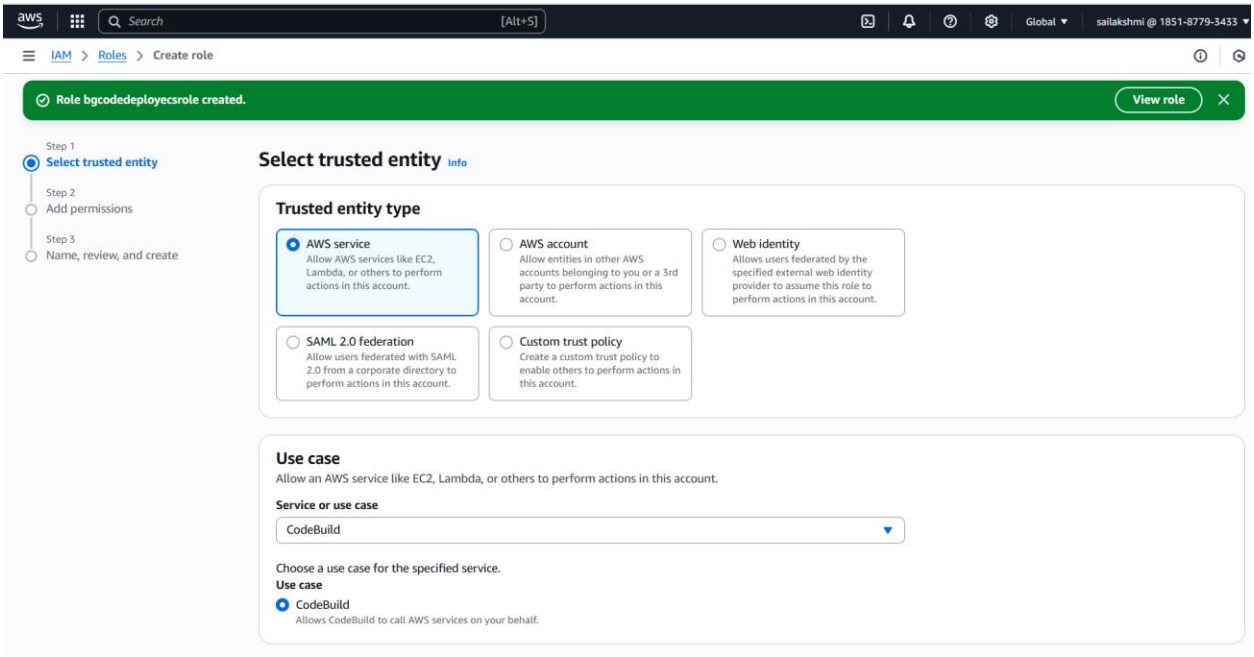
Another role for ECS

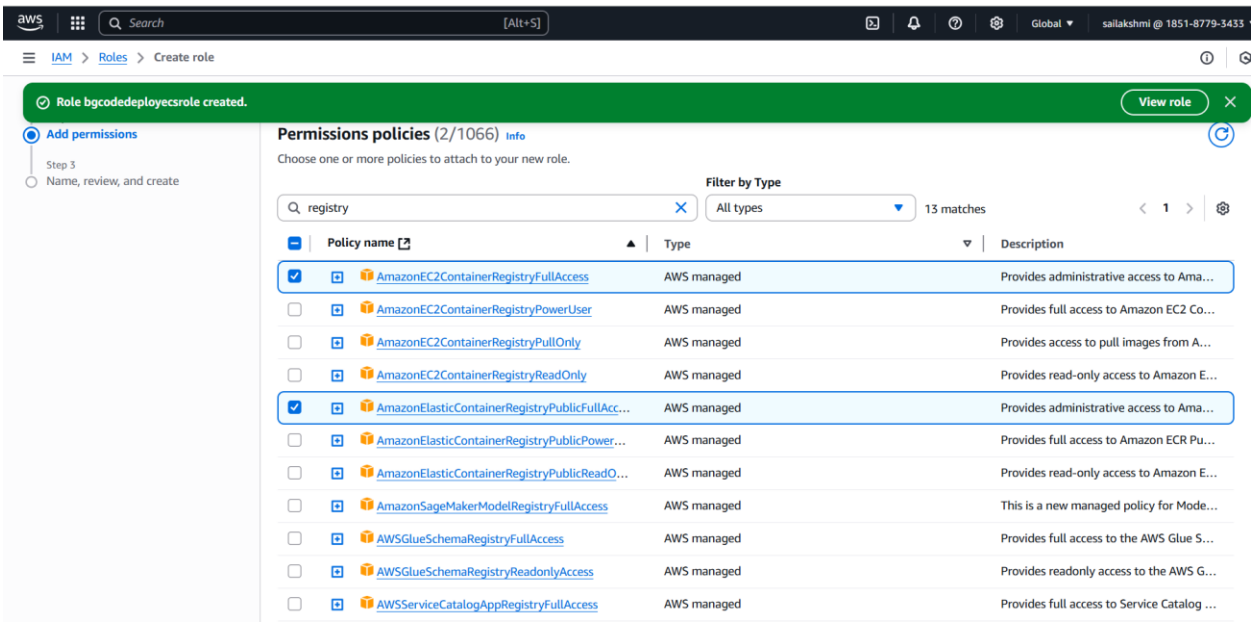


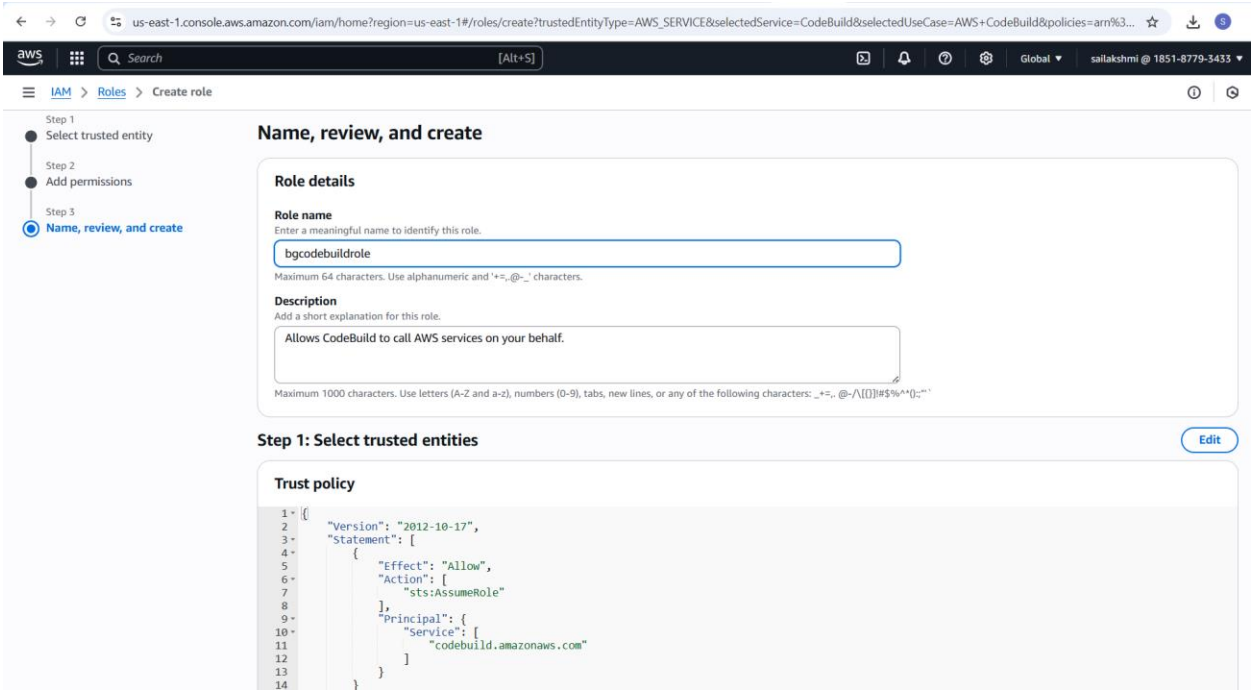




Another role for CodeBuild

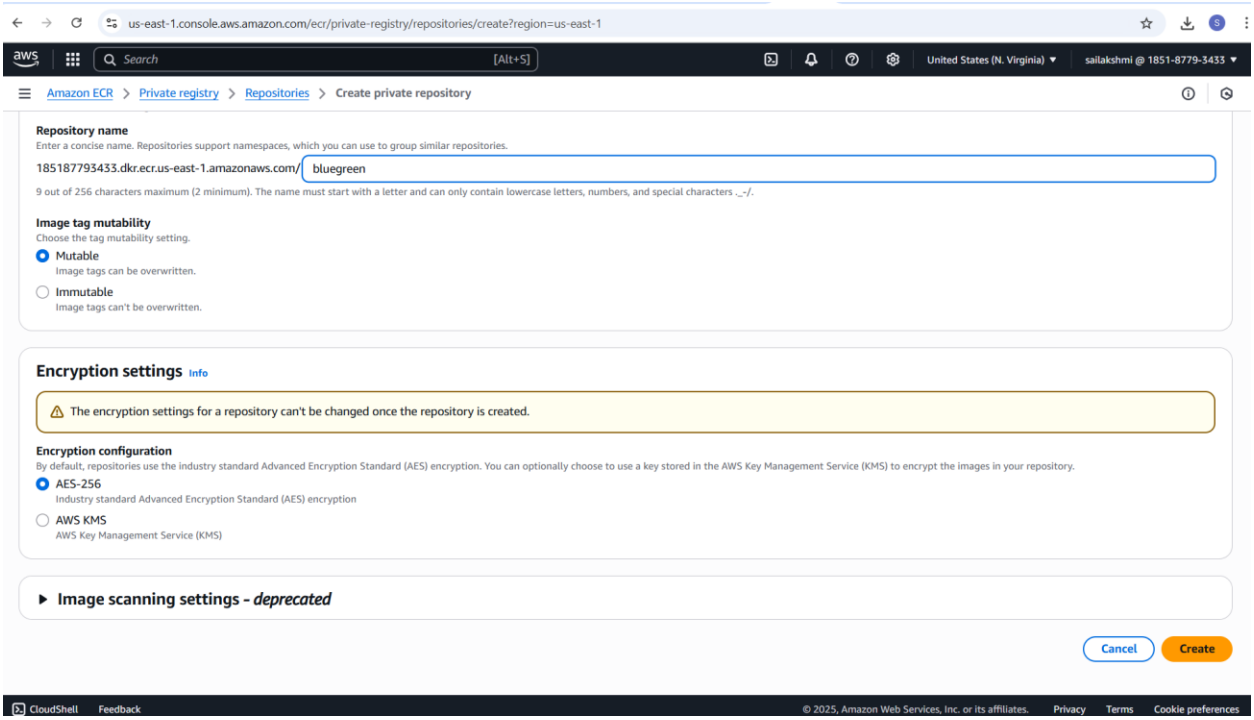

Search
[Alt+S]
Global
sailakshmi @ 1851-8779-3433
IAM > Roles > Create role
Role bgcodedeployecsrole created.
View role
Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create
Select trusted entity Info
Trusted entity type
AWS service
Allow AWS services like EC2, Lambda, or others to perform actions in this account.
AWS account
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
Web identity
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
SAML 2.0 federation
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
Custom trust policy
Create a custom trust policy to enable others to perform actions in this account.
Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.
Service or use case
CodeBuild
Choose a use case for the specified service.
Use case
CodeBuild
Allows CodeBuild to call AWS services on your behalf.

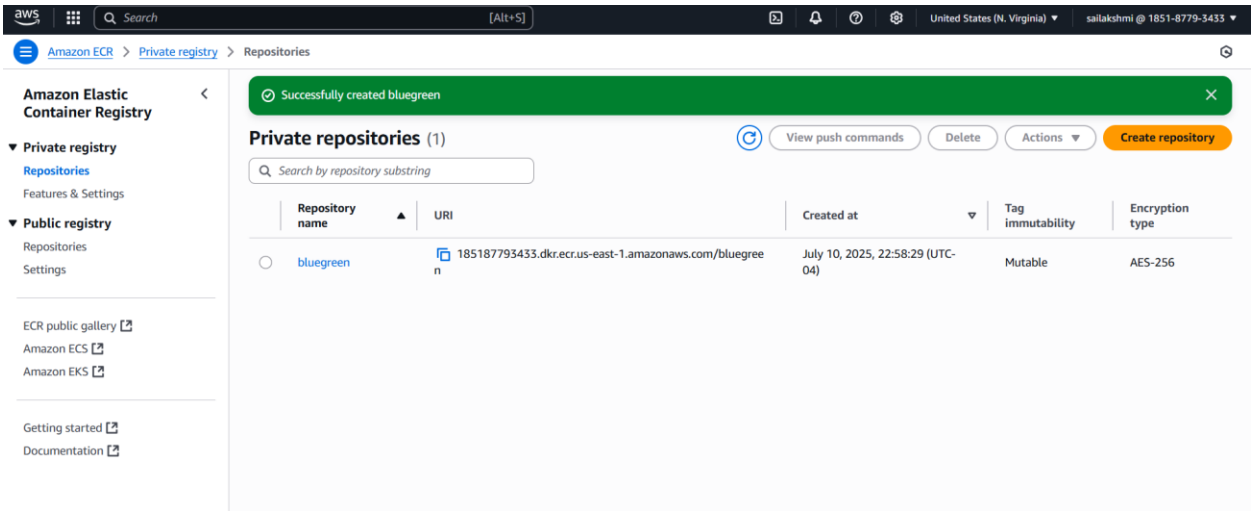

Search
[Alt+S]
Global
sailakshmi @ 1851-8779-3433
IAM > Roles > Create role
Role bgcodedeployecsrole created.
View role
Add permissions
Step 3
Name, review, and create
Permissions policies (2/1066) Info
Choose one or more policies to attach to your new role.
Filter by Type
registry
All types
13 matches
1
Policy name
Type
Description
AmazonEC2ContainerRegistryFullAccess
AWS managed
Provides administrative access to Ama...
AmazonEC2ContainerRegistryPowerUser
AWS managed
Provides full access to Amazon EC2 Co...
AmazonEC2ContainerRegistryPullOnly
AWS managed
Provides access to pull images from A...
AmazonEC2ContainerRegistryReadOnly
AWS managed
Provides read-only access to Amazon E...
AmazonElasticContainerRegistryPublicFullAcc...
AWS managed
Provides administrative access to Ama...
AmazonElasticContainerRegistryPublicPower...
AWS managed
Provides full access to Amazon ECR Pu...
AmazonElasticContainerRegistryPublicReadO...
AWS managed
Provides read-only access to Amazon E...
AmazonSageMakerModelRegistryFullAccess
AWS managed
This is a new managed policy for Mode...
AWSGlueSchemaRegistryFullAccess
AWS managed
Provides full access to the AWS Glue S...
AWSGlueSchemaRegistryReadonlyAccess
AWS managed
Provides readonly access to the AWS G...
AWSServiceCatalogAppRegistryFullAccess
AWS managed
Provides full access to Service Catalog ...



Step 3: Create ECR Repositories

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





Step 4: Push Initial Docker Images

Git clone

```
[root@ip-172-31-40-3 ec2-user]# git clone https://github.com/sailakshmi-d/blue-green-ecs
Cloning into 'blue-green-ecs'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 10 (delta 1), reused 10 (delta 1), pack-reused 0 (from 0)
Receiving objects: 100% (10/10), done.
Resolving deltas: 100% (1/1), done.
[root@ip-172-31-40-3 ec2-user]# cd blue-green-ecs/
[root@ip-172-31-40-3 blue-green-ecs]#
```

Create a docker image and push image to ECR

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

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

Login Succeeded
[root@ip-172-31-40-3 blue-green-ecs]#
```

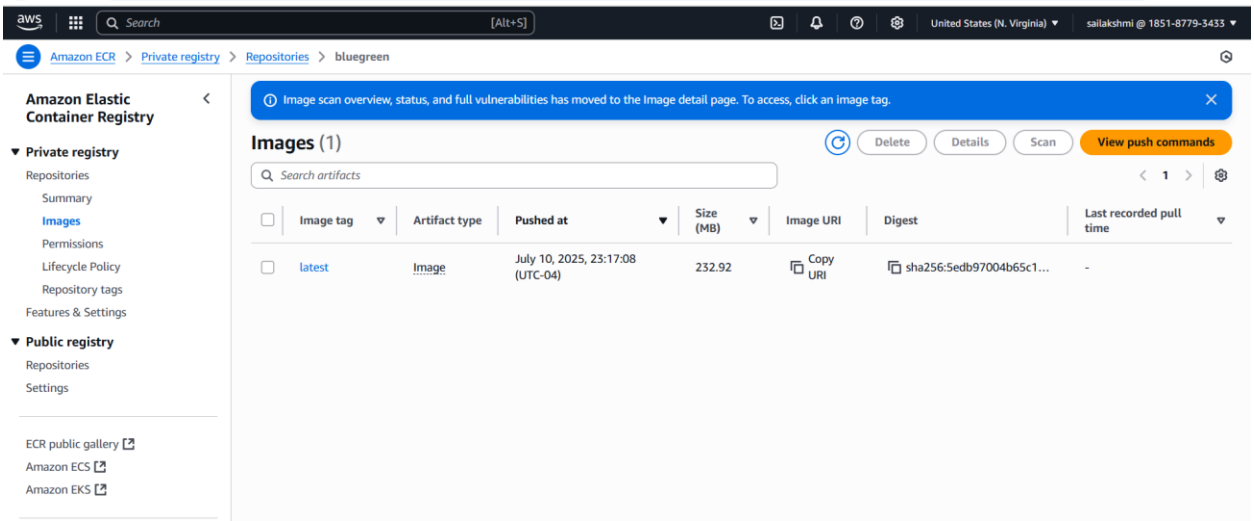
docker build -t bluegreen .

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

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

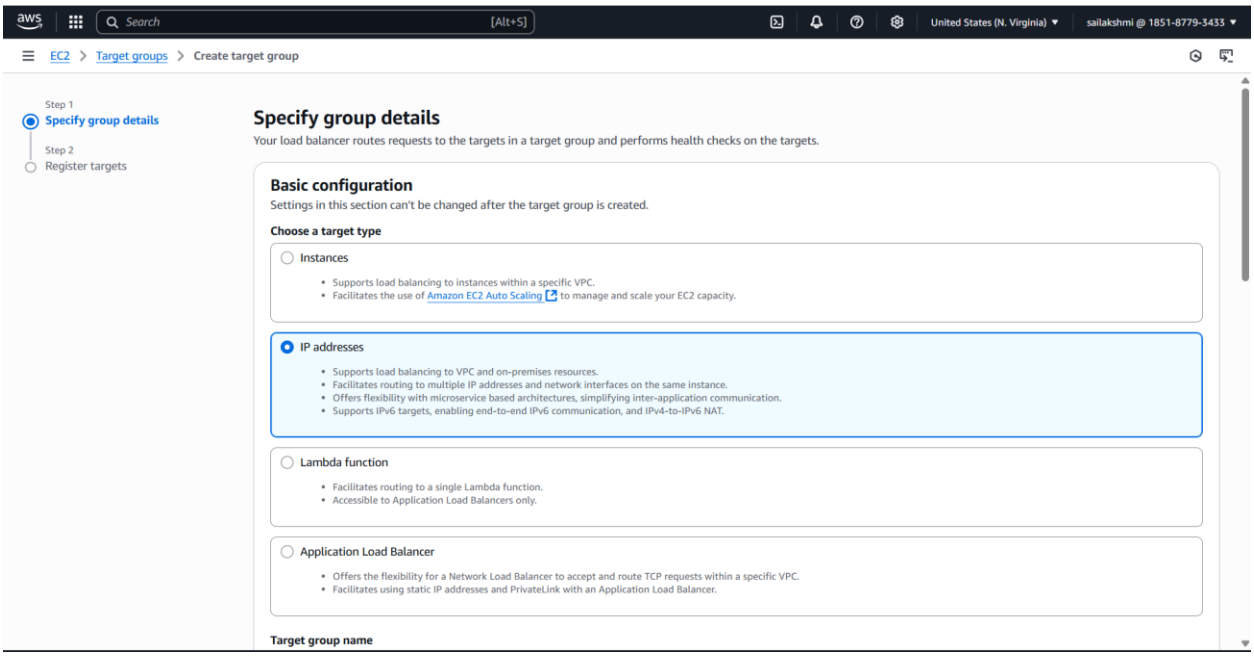
```
 => => extracting sha256:5634f043593bc14a2dd0ad7d54bf0ecd8b5e283ce596d00493b3433323f5c87                                                0.0s
 => => extracting sha256:98423917571b47a9645b52efba84d20caa5fba2aad8821d73b41dfe4fe4aaeb4                                               0.2s
 => => extracting sha256:66c4cda72d0a7dfb23708b47f95ba9ce7c003e6c5c76d9c71b41ea9ae5218033                                               0.0s
 => [internal] load build context                                                                                                       0.0s
 => => transferring context: 51.55kB                                                                                                    0.0s
 => [2/5] WORKDIR /usr/src/app                                                                                                          0.1s
 => [3/5] COPY package*.json ./                                                                                                         0.0s
 => [4/5] RUN npm install                                                                                                               1.9s
 => [5/5] COPY . .                                                                                                                      0.1s
 => exporting to image                                                                                                                  0.1s
 => => exporting layers                                                                                                                 0.1s
 => => writing image sha256:bf865ba6e92ccdee5633371e026199f2e62f2fef831aaaef8a5fabab7c1e731b                                            0.0s
 => => naming to docker.io/library/bluegreen                                                                                            0.0s
[root@ip-172-31-40-3 blue-green-ecs]# docker tag bluegreen:latest 185187793433.dkr.ecr.us-east-1.amazonaws.com/bluegreen:latest
[root@ip-172-31-40-3 blue-green-ecs]# docker push 185187793433.dkr.ecr.us-east-1.amazonaws.com/bluegreen:latest
The push refers to repository [185187793433.dkr.ecr.us-east-1.amazonaws.com/bluegreen]
eba59144031b: Pushed
584207d11cd6: Pushed
f579d94bd09f: Pushed
5f70bf18a086: Pushed
71e33b7c8ff1: Pushed
d278df77c3b2: Pushed
3decd7644066: Pushed
e474609372a6: Pushed
6be4dc94ddf6: Pushed
82d5847241f7: Pushed
b44911aa1197: Pushed
55c1577568ad: Pushed
5f22a1d1e743: Pushed
e15f10fc563f: Pushed
365dc8b4cf16: Pushed
95e78699c821: Pushed
latest: digest: sha256:5edb97004b65c103435e6319b9a7f2c26be7b5769524ef74635132b54b6b0046 size: 3660
[root@ip-172-31-40-3 blue-green-ecs]#
```

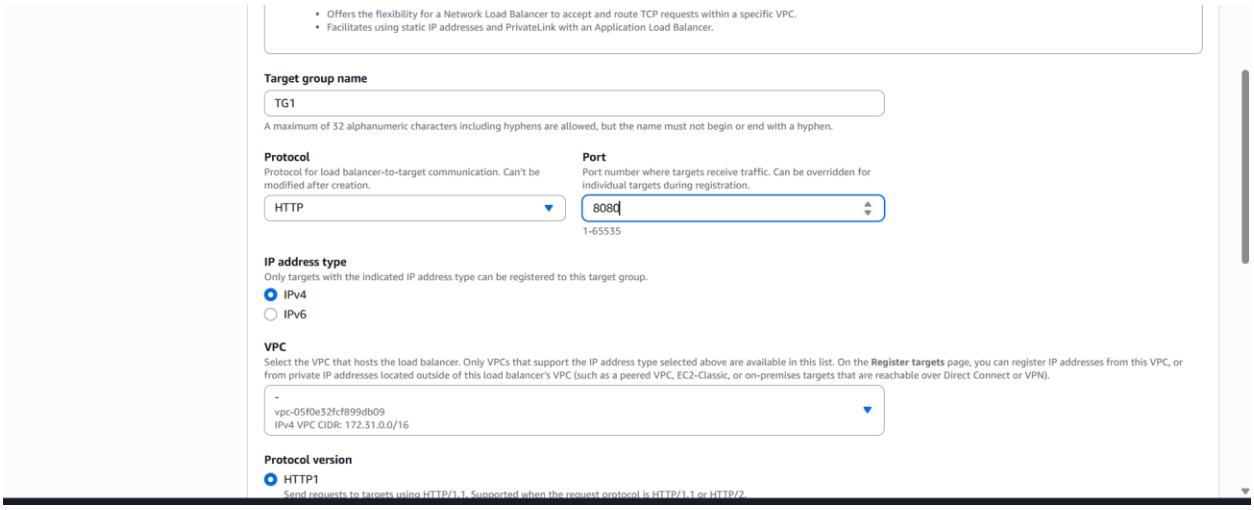
Now you see the image in ECR



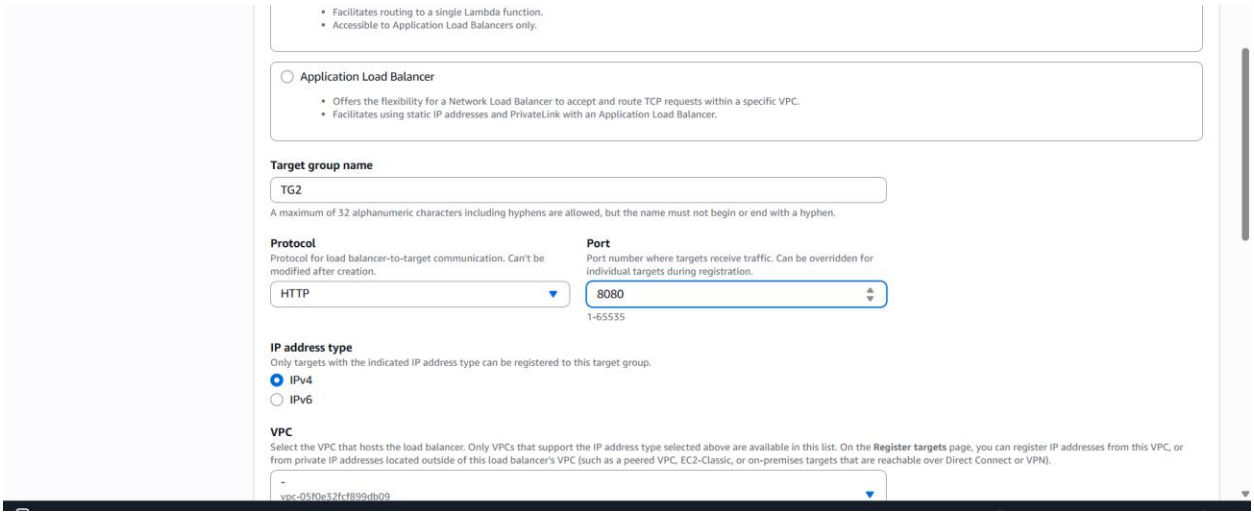
Step 5: Set Up the Load Balancer

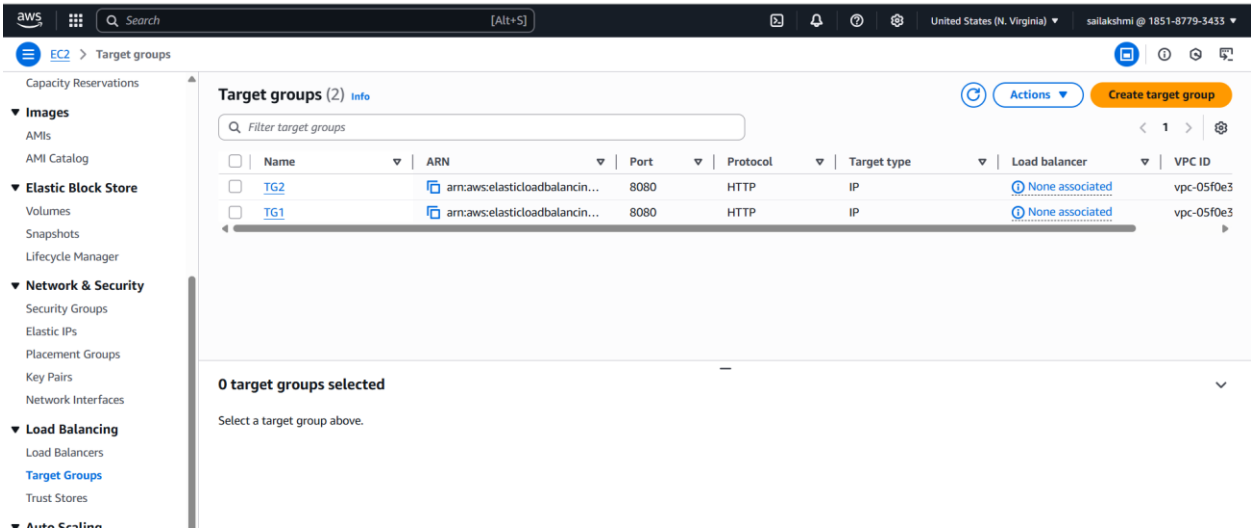
- Go to EC2 → Target Groups
- Create two target groups: blue-target and green-target
- Go to Load Balancers → Create ALB
    - Choose Application Load Balancer
    - Set listeners for port 8080
    - Register the two target groups
    - Add listener rules to forward traffic accordingly



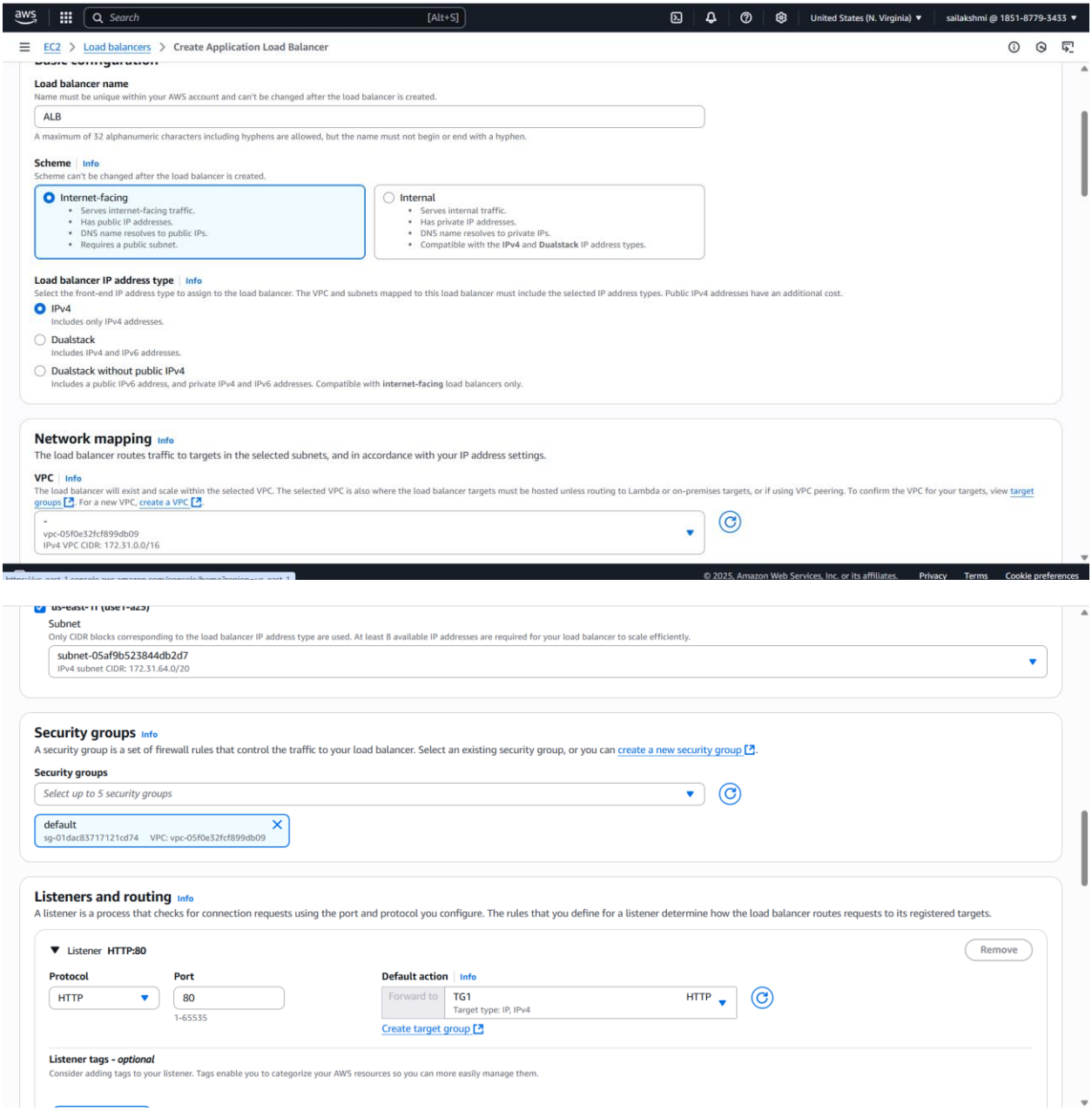


Create another target group

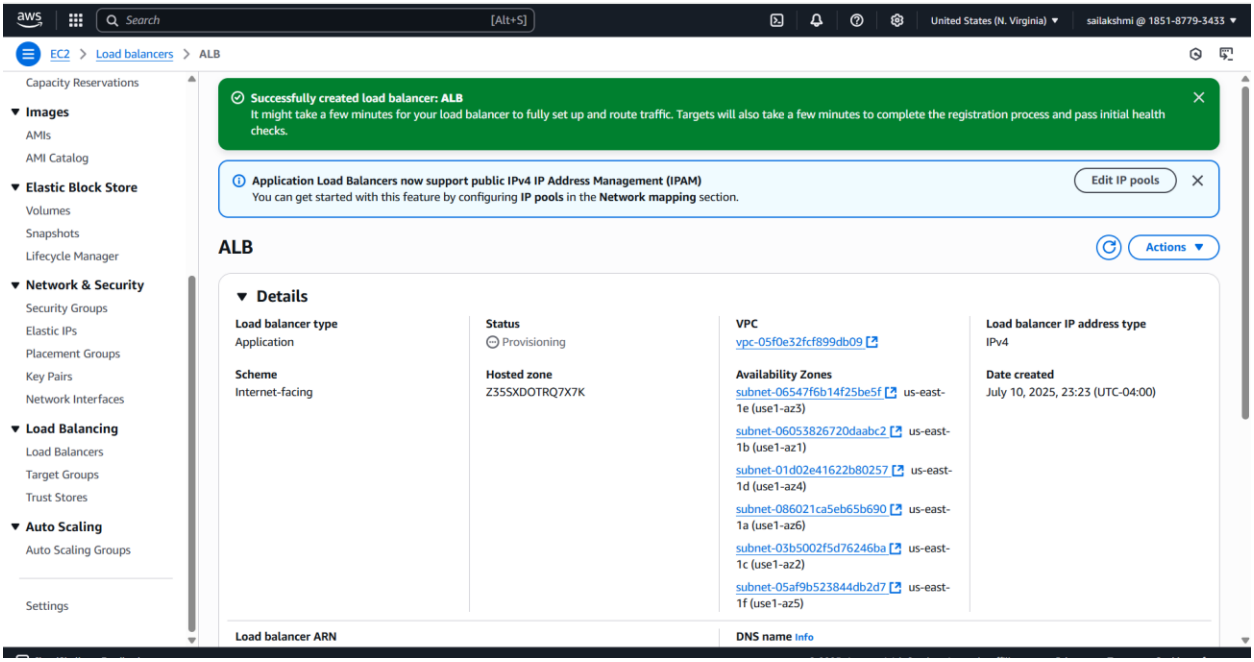


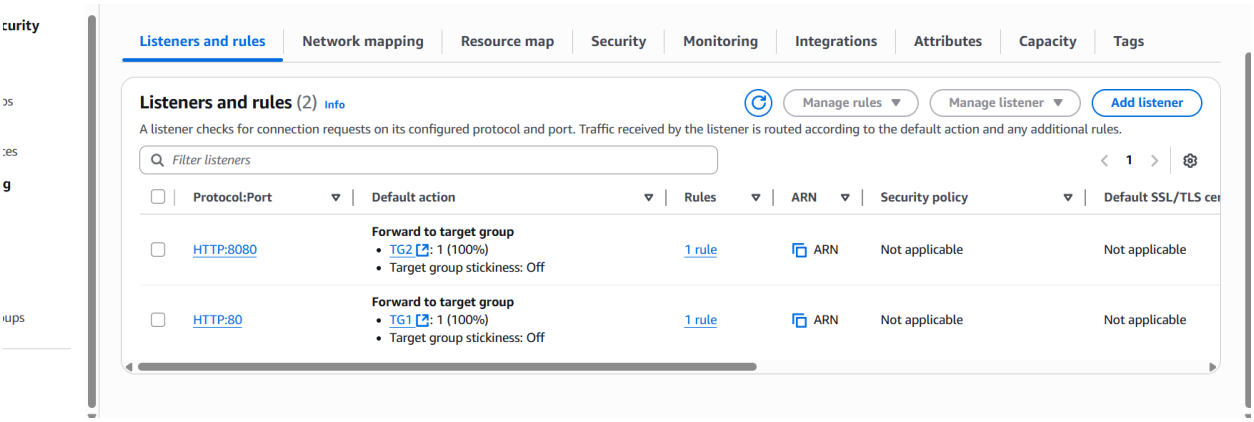


Now create a load balancer



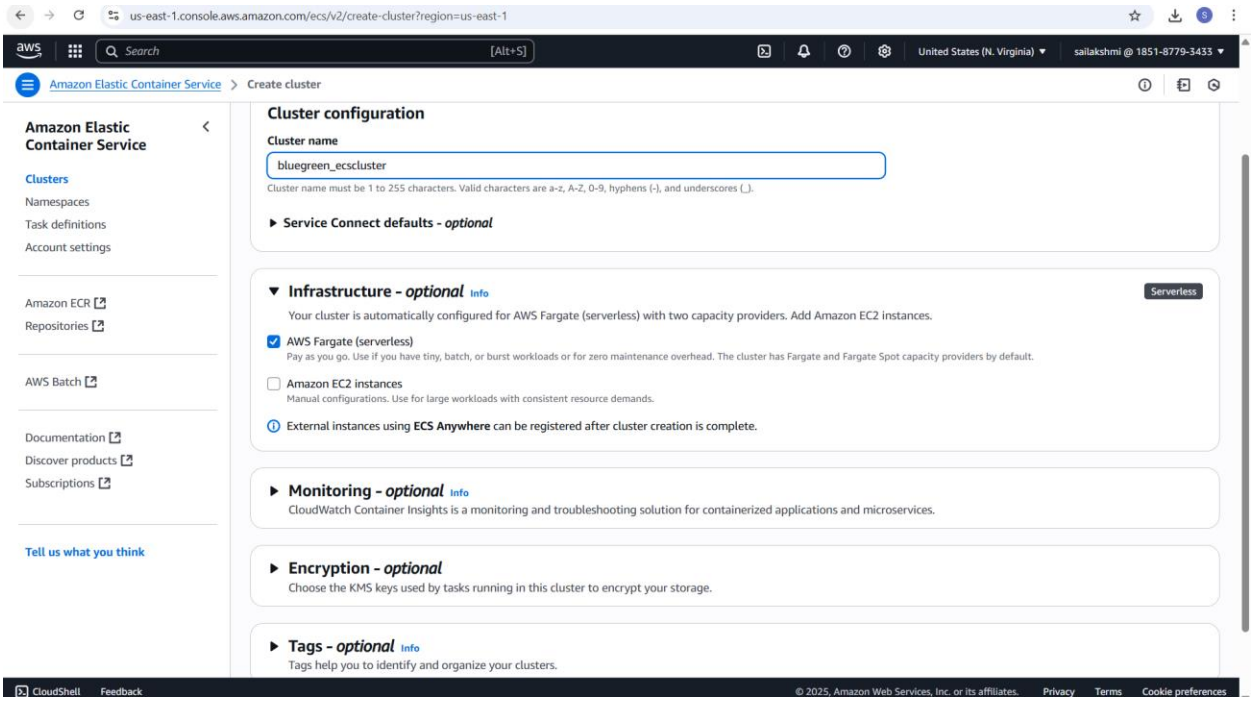
Go to listener tab and add listner with port 8080

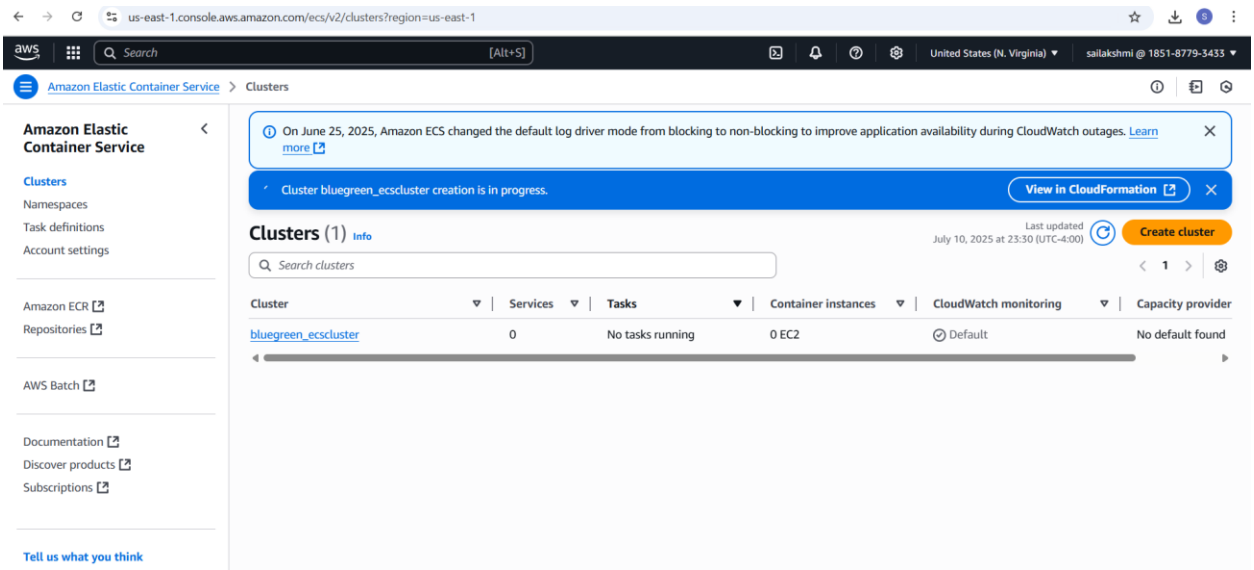




Step 6: Create ECS Cluster

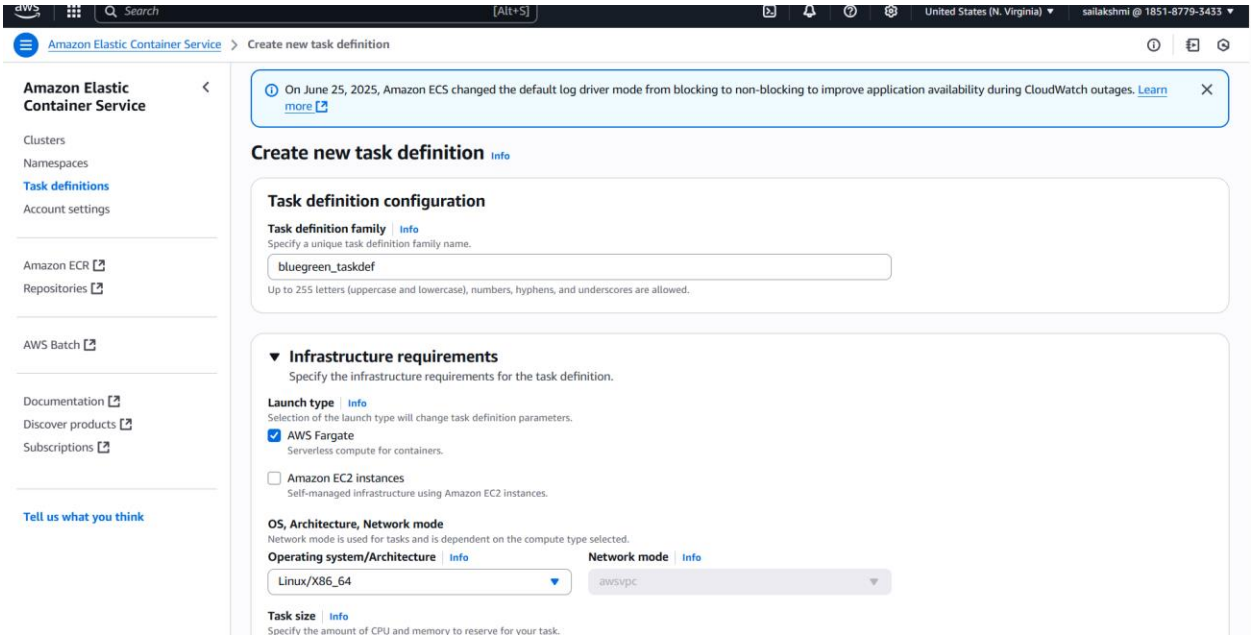
- Console → ECS → Clusters → Create Cluster → Networking only (Fargate) → Name it ECS cluster created.

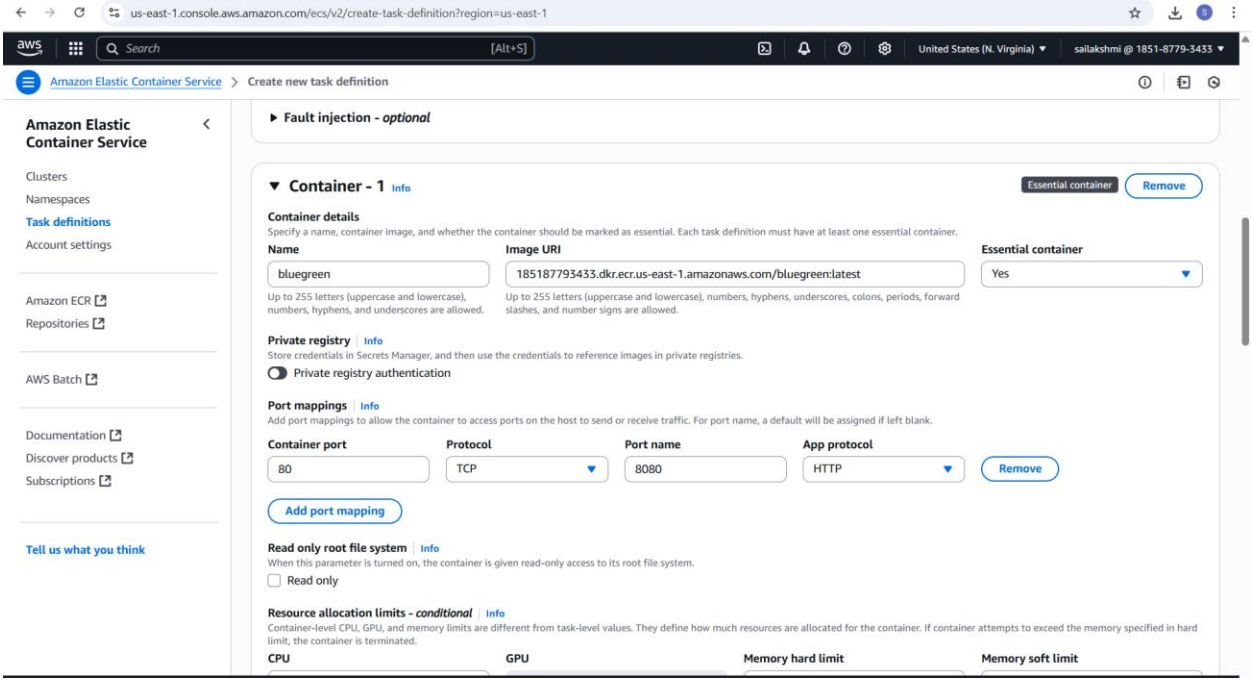


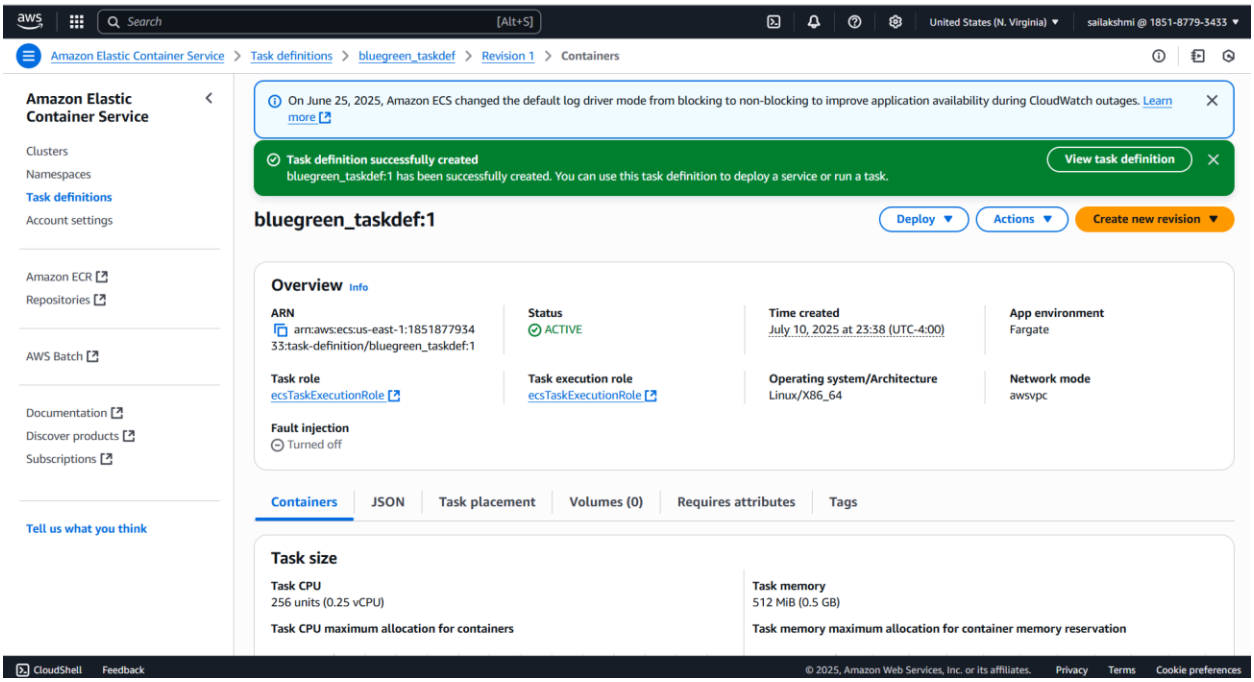


Step 7: Create Task Definitions

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

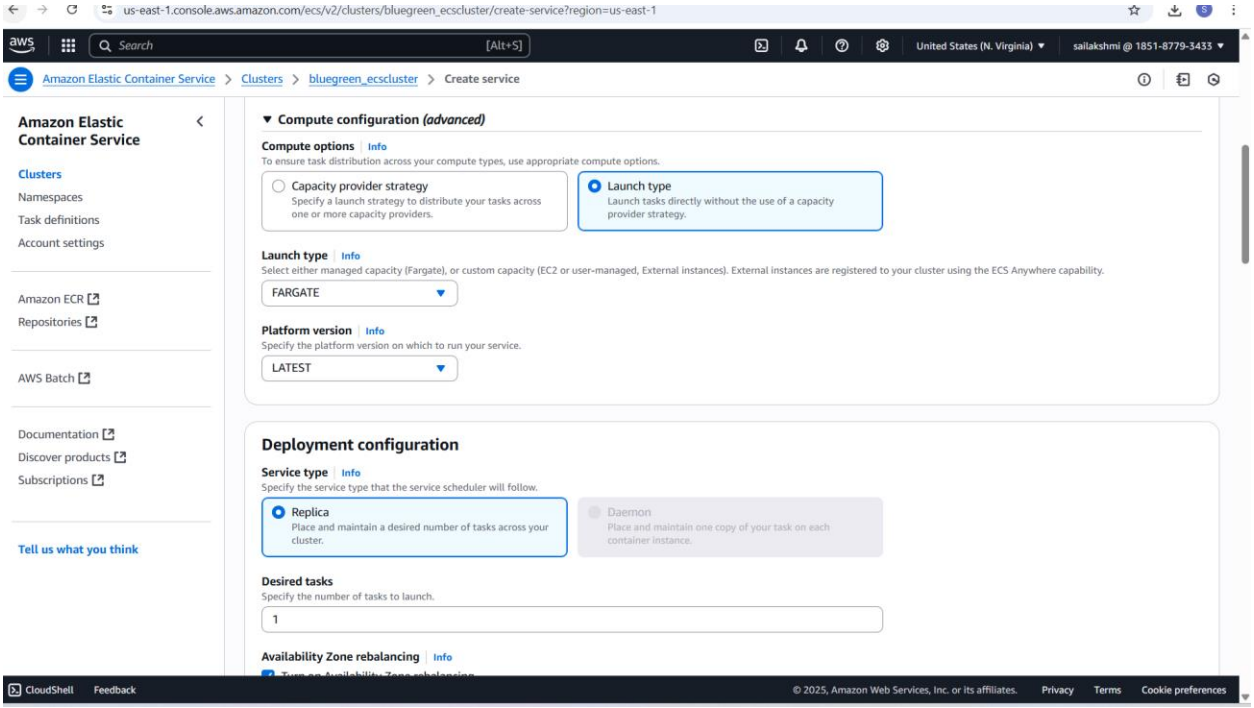

Search
[Alt+S]
United States (N. Virginia)
sailakshmi @ 1851-8779-3433
Amazon Elastic Container Service
Create new task definition
Amazon Elastic Container Service
Clusters
Namespaces
Task definitions
Account settings
Amazon ECR
Repositories
AWS Batch
Documentation
Discover products
Subscriptions
Tell us what you think
On June 25, 2025, Amazon ECS changed the default log driver mode from blocking to non-blocking to improve application availability during CloudWatch outages. Learn more
Create new task definition Info
Task definition configuration
Task definition family Info
Specify a unique task definition family name.
bluegreen_taskdef
Up to 255 letters (uppercase and lowercase), numbers, hyphens, and underscores are allowed.
Infrastructure requirements
Specify the infrastructure requirements for the task definition.
Launch type Info
Selection of the launch type will change task definition parameters.
AWS Fargate
Serverless compute for containers.
Amazon EC2 instances
Self-managed infrastructure using Amazon EC2 instances.
OS, Architecture, Network mode
Network mode is used for tasks and is dependent on the compute type selected.
Operating system/Architecture Info
Linux/X86_64
Network mode Info
awsvpc
Task size Info

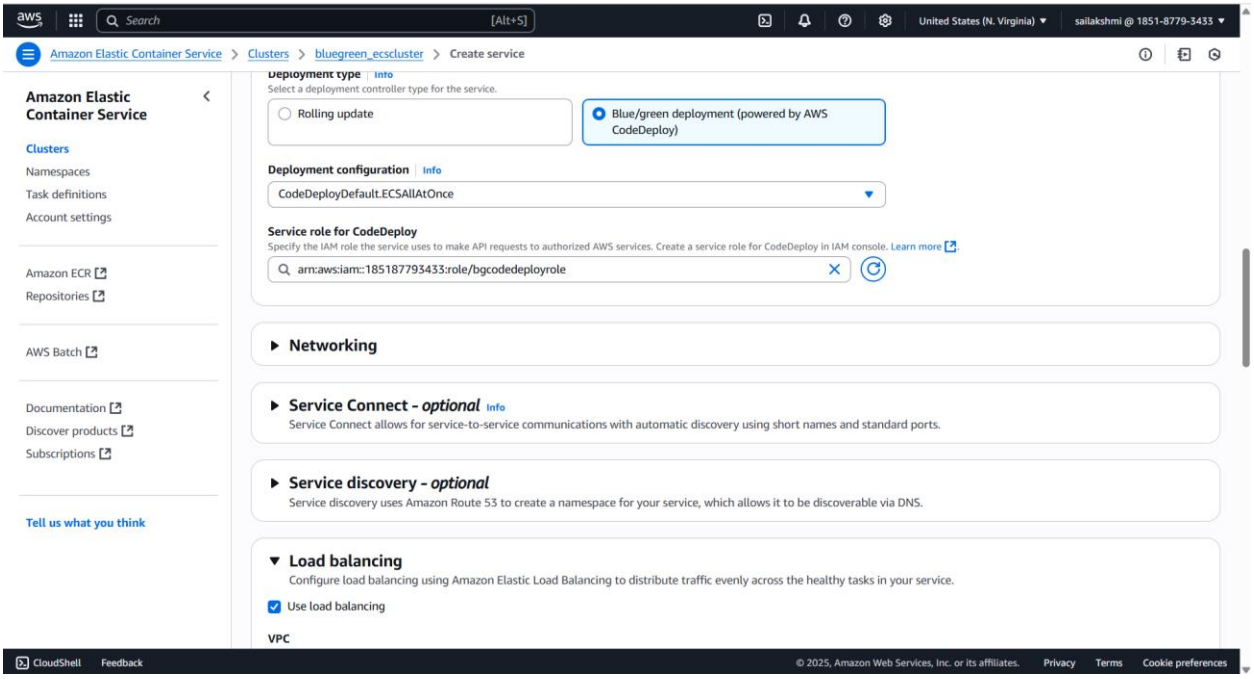

us-east-1.console.aws.amazon.com/ecs/v2/create-task-definition?region=us-east-1
Search
[Alt+S]
United States (N. Virginia)
sailakshmi @ 1851-8779-3433
Amazon Elastic Container Service
Create new task definition
Amazon Elastic Container Service
Clusters
Namespaces
Task definitions
Account settings
Amazon ECR
Repositories
AWS Batch
Documentation
Discover products
Subscriptions
Tell us what you think
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
bluegreen
Up to 255 letters (uppercase and lowercase), numbers, hyphens, and underscores are allowed.
Image URI
185187793433.dkr.ecr.us-east-1.amazonaws.com/bluegreen:latest
Up to 255 letters (uppercase and lowercase), numbers, hyphens, underscores, colons, periods, forward slashes, and number signs are allowed.
Essential container
Yes
Private registry Info
Store credentials in Secrets Manager, and then use the credentials to reference images in private registries.
Private registry authentication
Port mappings Info
Add port mappings to allow the container to access ports on the host to send or receive traffic. For port name, a default will be assigned if left blank.
Container port
80
Protocol
TCP
Port name
8080
App protocol
HTTP
Remove
Add port mapping
Read only root file system Info
When this parameter is turned on, the container is given read-only access to its root file system.
Read only
Resource allocation limits - conditional Info
Container-level CPU, GPU, and memory limits are different from task-level values. They define how much resources are allocated for the container. If container attempts to exceed the memory specified in hard limit, the container is terminated.
CPU
GPU
Memory hard limit
Memory soft limit

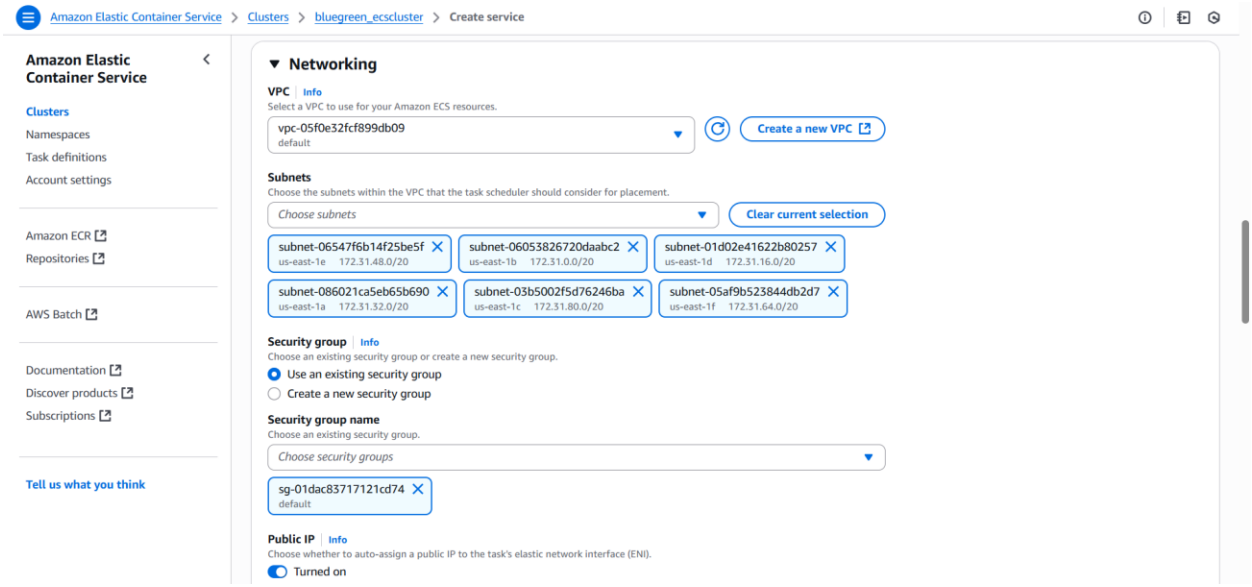


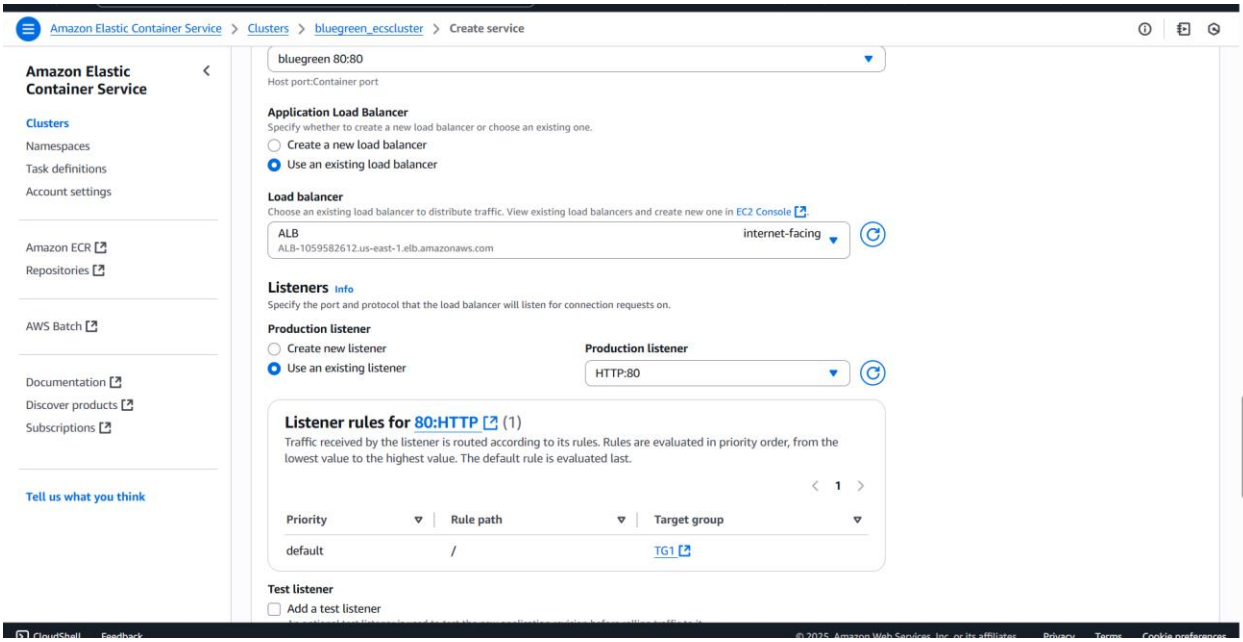
Step 8: Create ECS Services (Blue/Green Enabled)

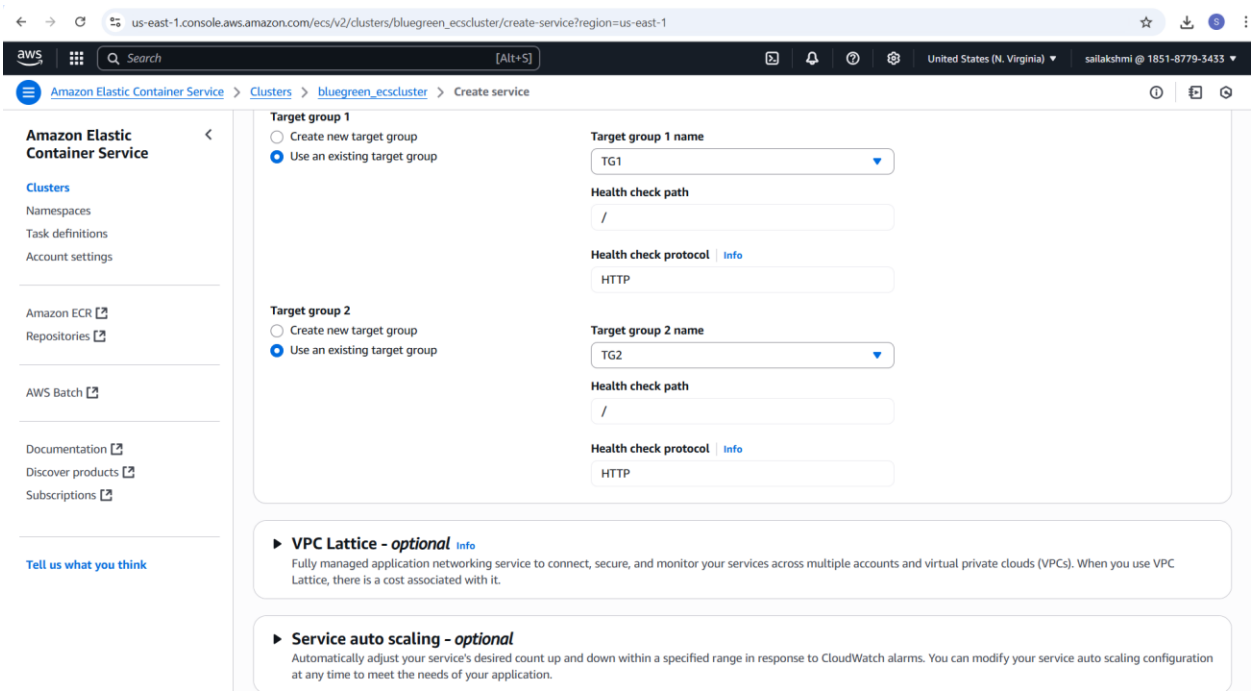
- ECS → Clusters → select cluster → Create → FARGATE service.
- Service name, Task Definition, Set number of tasks and VPC/subnets/security as needed.
- Under Deployment type, choose Blue/Green (CodeDeploy) and select an Application Load Balancer:
    - Set container name/port = 8080.
    - Select target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


us-east-1.console.aws.amazon.com/ecs/v2/clusters/bluegreen_ecscluster/create-service?region=us-east-1
Search [Alt+S]
United States (N. Virginia)
sailakshmi @ 1851-8779-3433
Amazon Elastic Container Service > Clusters > bluegreen_ecscluster > Create service
Amazon Elastic Container Service
Clusters
Namespaces
Task definitions
Account settings
Amazon ECR
Repositories
AWS Batch
Documentation
Discover products
Subscriptions
Tell us what you think
Compute configuration (advanced)
Compute options | Info
To ensure task distribution across your compute types, use appropriate compute options.
Capacity provider strategy
Specify a launch strategy to distribute your tasks across one or more capacity providers.
Launch type
Launch tasks directly without the use of a capacity provider strategy.
Launch type | Info
Select either managed capacity (Fargate), or custom capacity (EC2 or user-managed, External instances). External instances are registered to your cluster using the ECS Anywhere capability.
FARGATE
Platform version | Info
Specify the platform version on which to run your service.
LATEST
Deployment configuration
Service type | Info
Specify the service type that the service scheduler will follow.
Replica
Place and maintain a desired number of tasks across your cluster.
Daemon
Place and maintain one copy of your task on each container instance.
Desired tasks
Specify the number of tasks to launch.
1
Availability Zone rebalancing | Info
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Search [Alt+S]
United States (N. Virginia)
sailakshmi @ 1851-8779-3433
Amazon Elastic Container Service > Clusters > bluegreen_ecscluster > Create service
Amazon Elastic Container Service
Clusters
Namespaces
Task definitions
Account settings
Amazon ECR
Repositories
AWS Batch
Documentation
Discover products
Subscriptions
Tell us what you think
Deployment type | Info
Select a deployment controller type for the service.
Rolling update
Blue/green deployment (powered by AWS CodeDeploy)
Deployment configuration | Info
CodeDeployDefault.ECSAllAtOnce
Service role for CodeDeploy
Specify the IAM role the service uses to make API requests to authorized AWS services. Create a service role for CodeDeploy in IAM console. Learn more
arn:aws:iam::185187793433:role/bgcodedeployrole
Networking
Service Connect - optional Info
Service Connect allows for service-to-service communications with automatic discovery using short names and standard ports.
Service discovery - optional
Service discovery uses Amazon Route 53 to create a namespace for your service, which allows it to be discoverable via DNS.
Load balancing
Configure load balancing using Amazon Elastic Load Balancing to distribute traffic evenly across the healthy tasks in your service.
Use load balancing
VPC
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

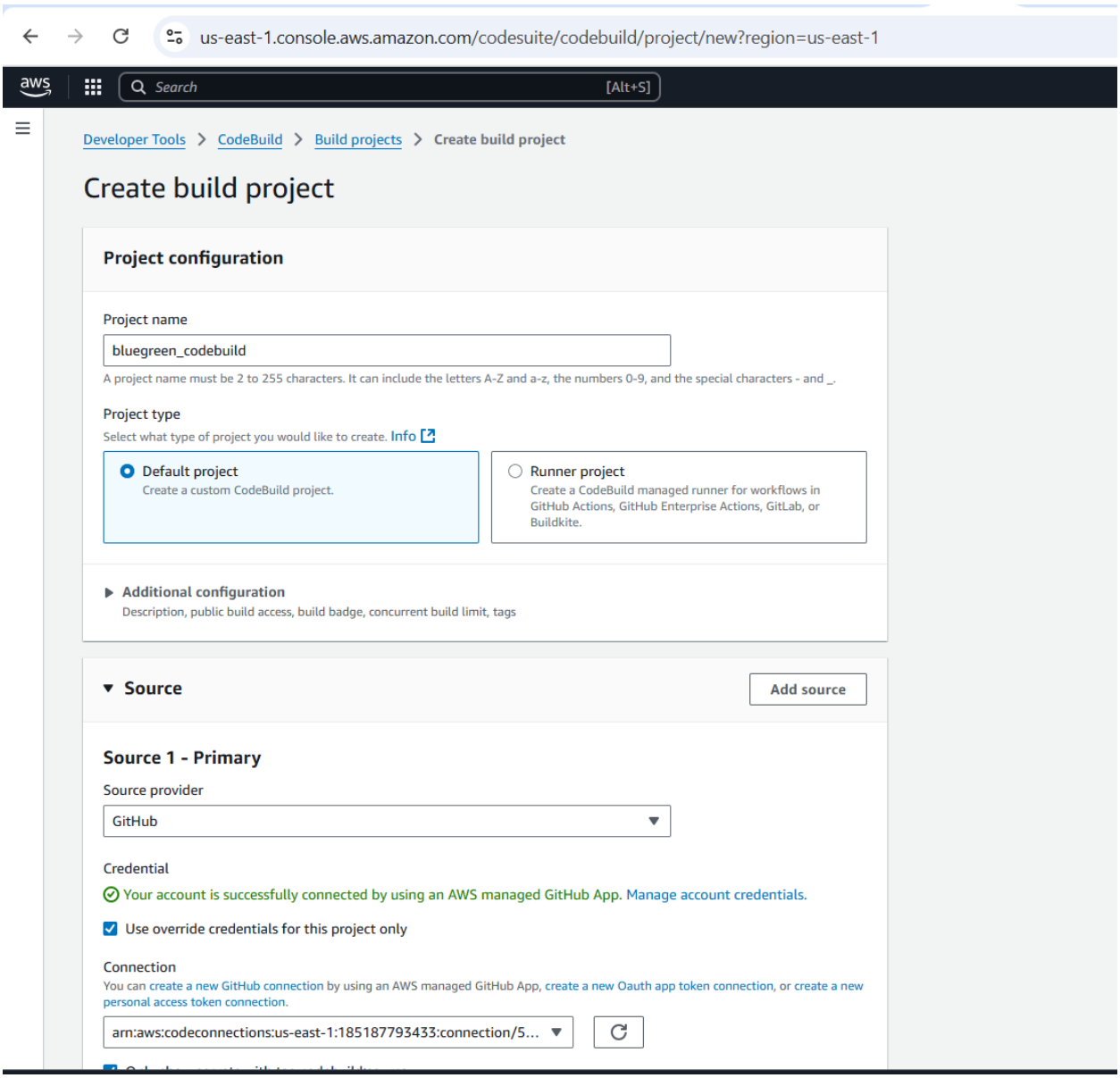

Amazon Elastic Container Service > Clusters > bluegreen_ecscluster > Create service
Amazon Elastic Container Service
Clusters
Namespaces
Task definitions
Account settings
Amazon ECR
Repositories
AWS Batch
Documentation
Discover products
Subscriptions
Tell us what you think
Networking
VPC Info
Select a VPC to use for your Amazon ECS resources.
vpc-05f0e32fcf899db09
default
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-06547f6b14f25be5f
us-east-1e 172.31.48.0/20
subnet-06053826720daabc2
us-east-1b 172.31.0.0/20
subnet-01d02e41622b80257
us-east-1d 172.31.16.0/20
subnet-086021ca5eb65b690
us-east-1a 172.31.32.0/20
subnet-03b5002f5d76246ba
us-east-1c 172.31.80.0/20
subnet-05af9b523844db2d7
us-east-1f 172.31.64.0/20
Security group Info
Choose an existing security group or create a new security group.
Use an existing security group
Create a new security group
Security group name
Choose an existing security group.
Choose security groups
sg-01dac83717121cd74
default
Public IP Info
Choose whether to auto-assign a public IP to the task's elastic network interface (ENI).
Turned on

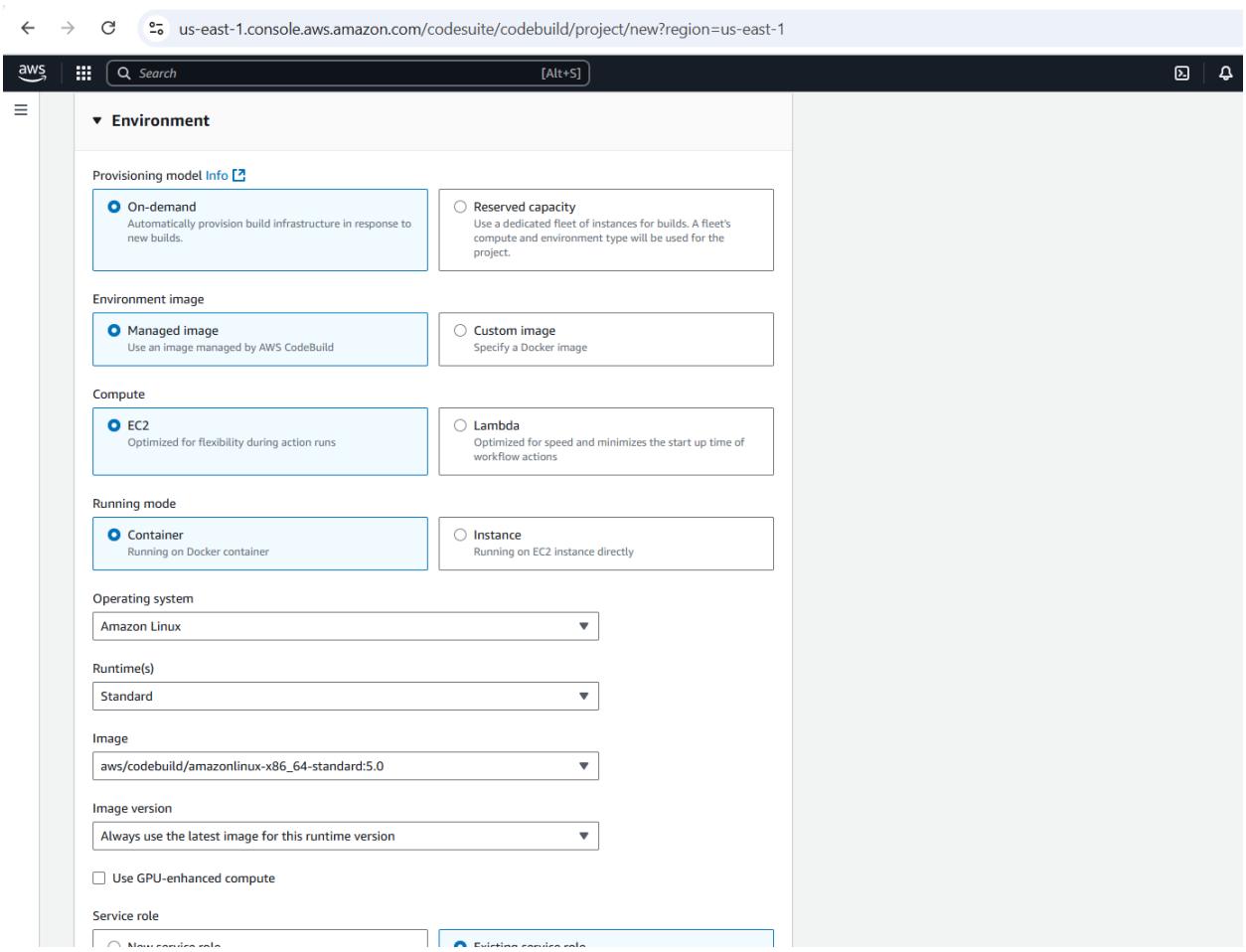

Amazon Elastic Container Service > Clusters > bluegreen_ecscluster > Create service
Amazon Elastic Container Service
Clusters
Namespaces
Task definitions
Account settings
Amazon ECR
Repositories
AWS Batch
Documentation
Discover products
Subscriptions
Tell us what you think
bluegreen 80:80
Host port:Container port
Application Load Balancer
Specify whether to create a new load balancer or choose an existing one.
Create a new load balancer
Use an existing load balancer
Load balancer
Choose an existing load balancer to distribute traffic. View existing load balancers and create new one in EC2 Console.
ALB
ALB-1059582612.us-east-1.elb.amazonaws.com
internet-facing
Listeners Info
Specify the port and protocol that the load balancer will listen for connection requests on.
Production listener
Create new listener
Use an existing listener
Production listener
HTTP:80
Listener rules for 80:HTTP (1)
Traffic received by the listener is routed according to its rules. Rules are evaluated in priority order, from the lowest value to the highest value. The default rule is evaluated last.
1
Priority
Rule path
Target group
default
/
TG1
Test listener
Add a test listener
CloudShell
Feedback
Privacy
Terms
Cookie preferences

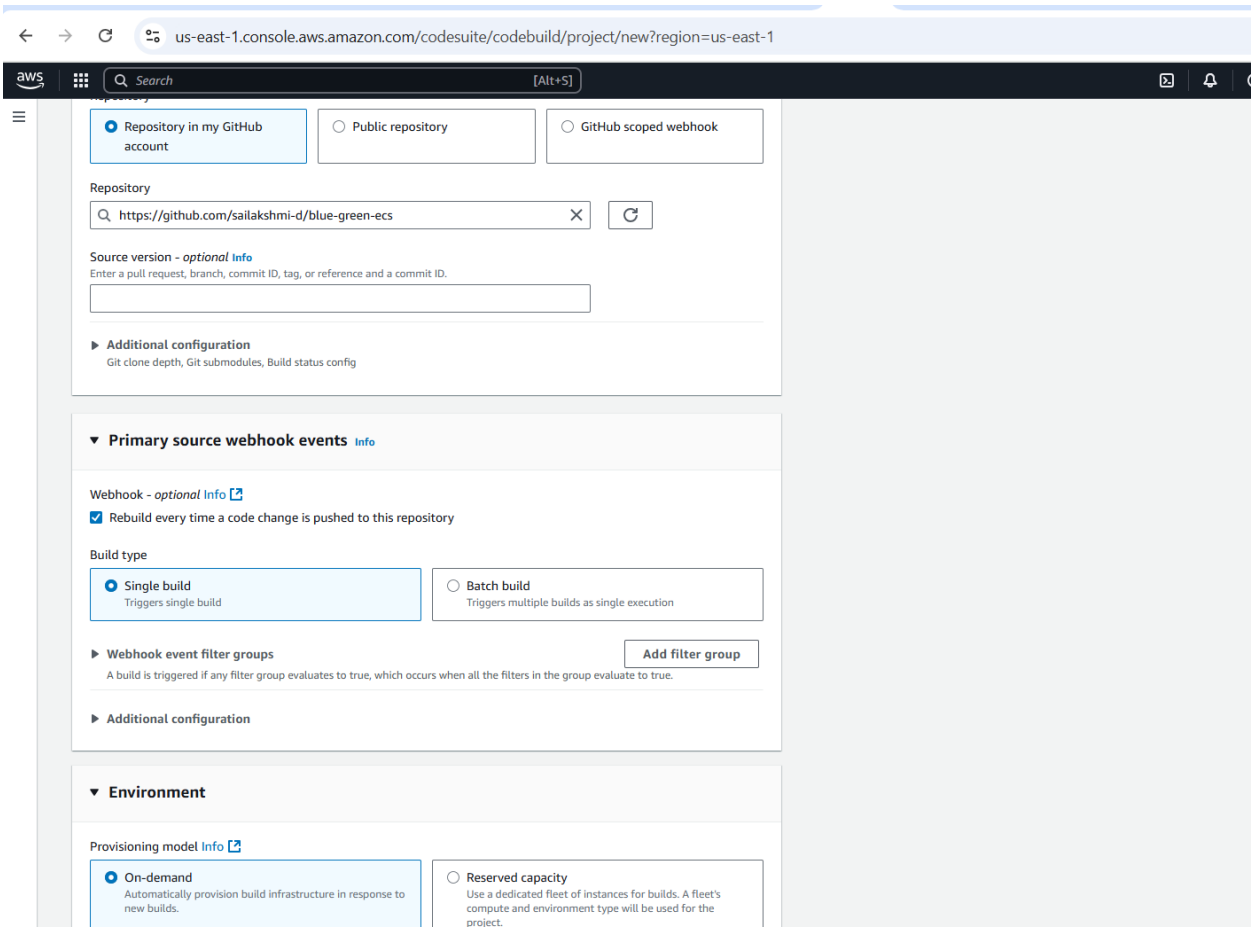


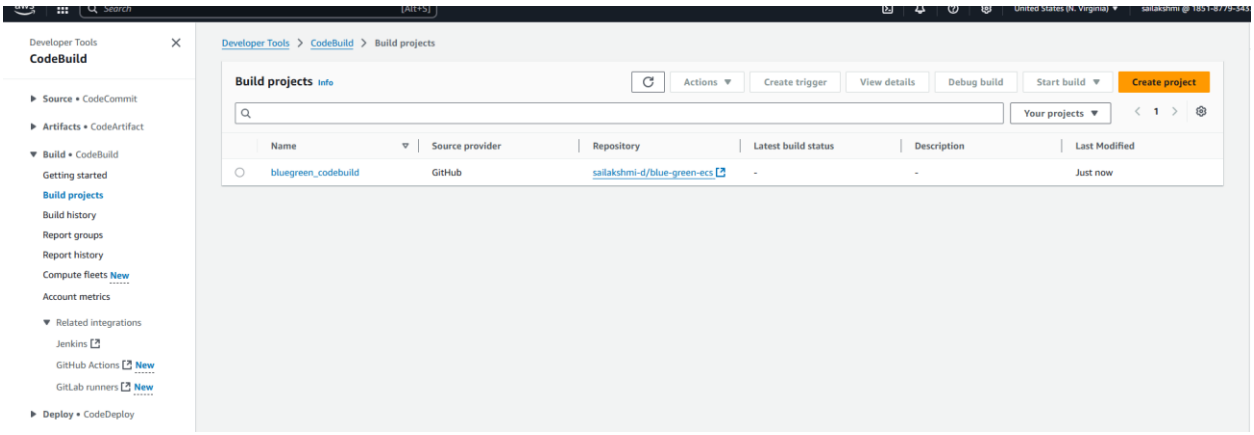
Step 9: Create CodeBuild Project

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


us-east-1.console.aws.amazon.com/codesuite/codebuild/project/new?region=us-east-1
aws
Search
[Alt+S]
Developer Tools > CodeBuild > Build projects > Create build project
Create build project
Project configuration
Project name
bluegreen_codebuild
A project name must be 2 to 255 characters. It can include the letters A-Z and a-z, the numbers 0-9, and the special characters - and _.
Project type
Select what type of project you would like to create. Info
Default project
Create a custom CodeBuild project.
Runner project
Create a CodeBuild managed runner for workflows in GitHub Actions, GitHub Enterprise Actions, GitLab, or Buildkite.
Additional configuration
Description, public build access, build badge, concurrent build limit, tags
Source
Add source
Source 1 - Primary
Source provider
GitHub
Credential
Your account is successfully connected by using an AWS managed GitHub App. Manage account credentials.
Use override credentials for this project only
Connection
You can create a new GitHub connection by using an AWS managed GitHub App, create a new Oauth app token connection, or create a new personal access token connection.
arn:aws:codeconnections:us-east-1:185187793433:connection/5...


us-east-1.console.aws.amazon.com/codesuite/codebuild/project/new?region=us-east-1
aws
Search
[Alt+S]
Environment
Provisioning model Info
On-demand
Automatically provision build infrastructure in response to new builds.
Reserved capacity
Use a dedicated fleet of instances for builds. A fleet's compute and environment type will be used for the project.
Environment image
Managed image
Use an image managed by AWS CodeBuild
Custom image
Specify a Docker image
Compute
EC2
Optimized for flexibility during action runs
Lambda
Optimized for speed and minimizes the start up time of workflow actions
Running mode
Container
Running on Docker container
Instance
Running on EC2 instance directly
Operating system
Amazon Linux
Runtime(s)
Standard
Image
aws/codebuild/amazonlinux-x86_64-standard:5.0
Image version
Always use the latest image for this runtime version
Use GPU-enhanced compute
Service role



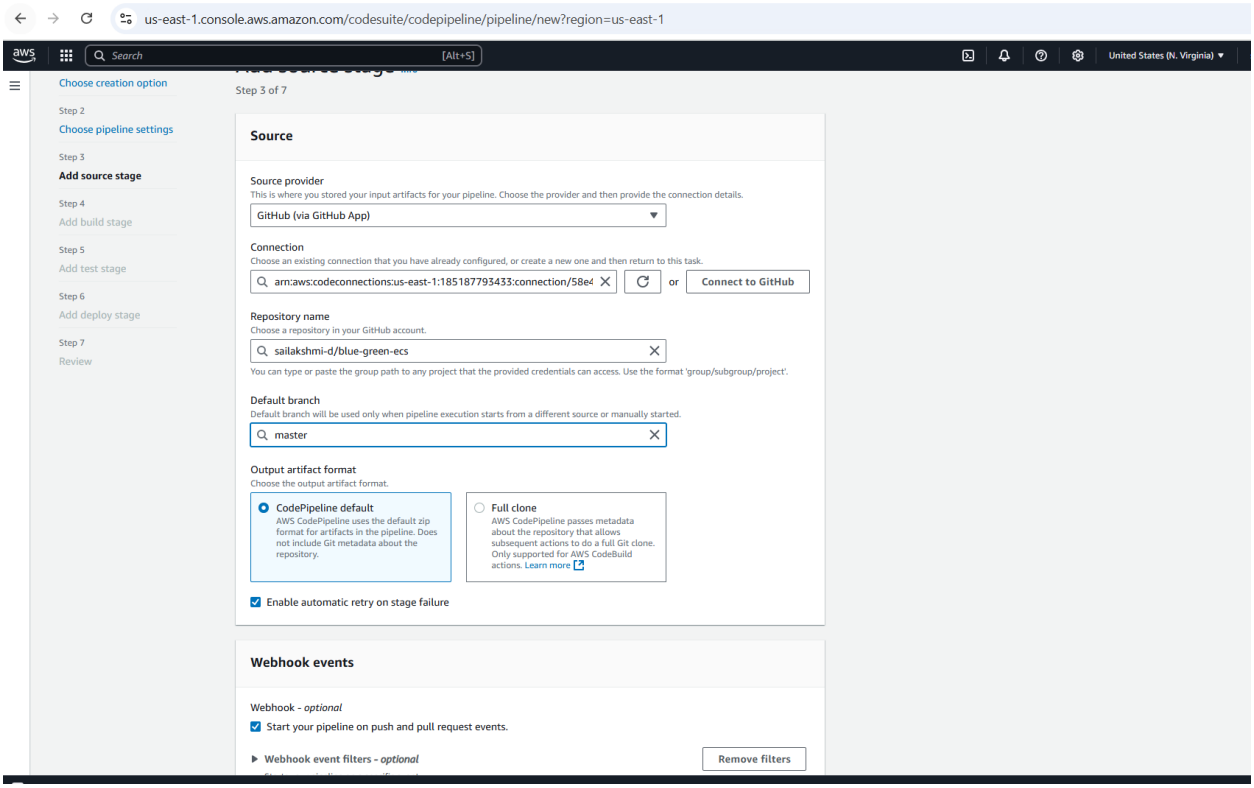


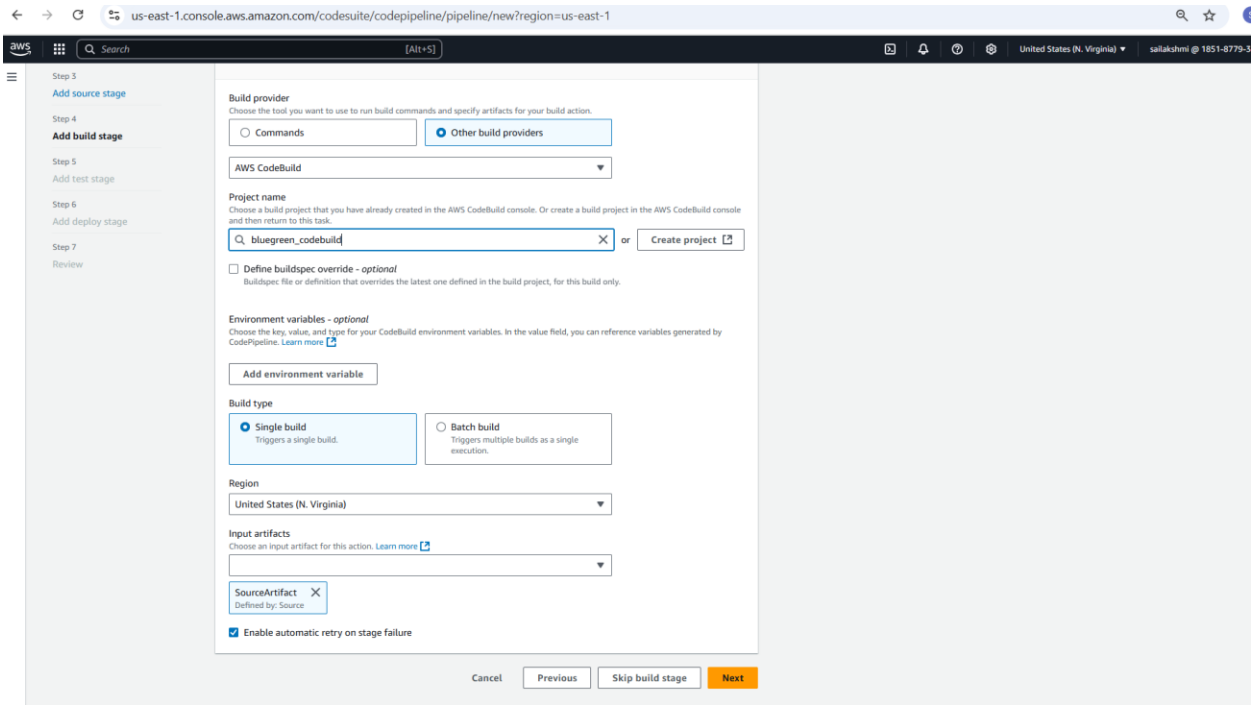
Step 10: Create a Unified CodePipeline

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

- buildspec.yaml builds both containers, pushes to ECR and writes imagedefinitions.json.

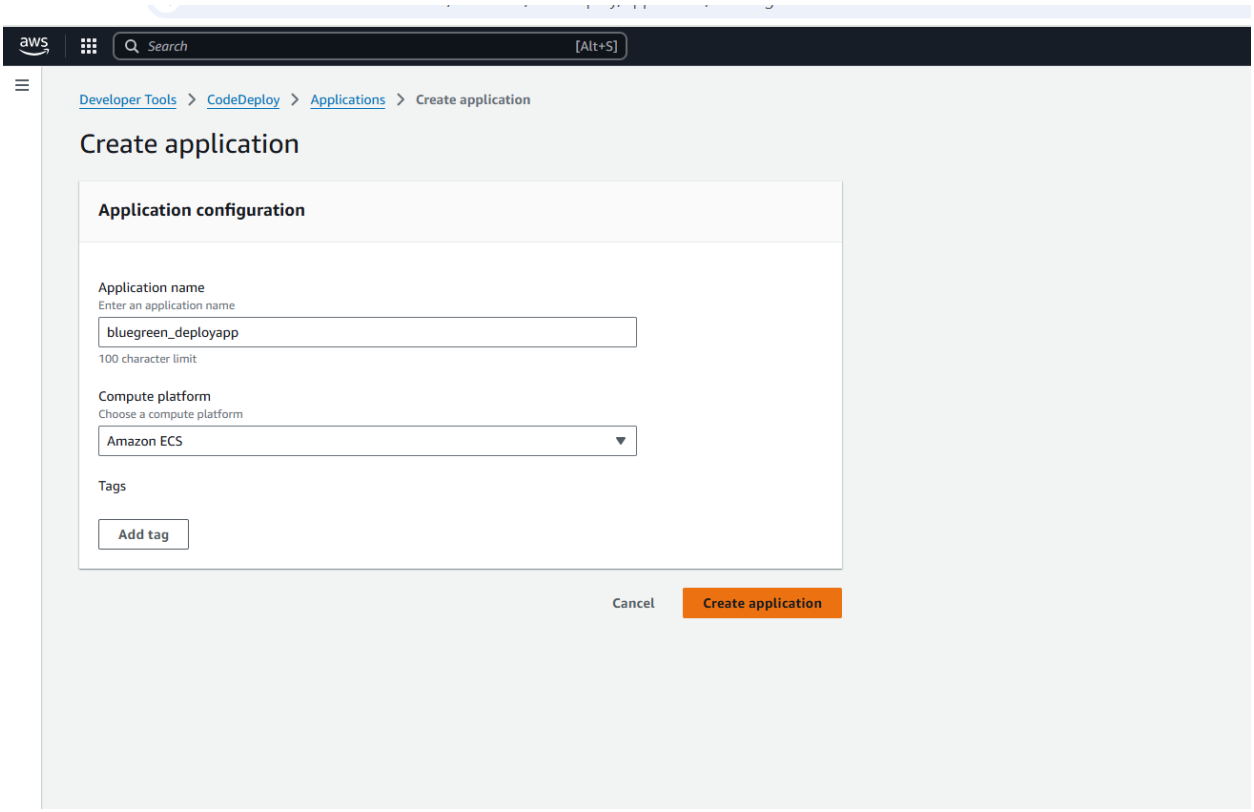
- Add **Deploy stage**:
  - Add **CodeDeploy ECS** actions:



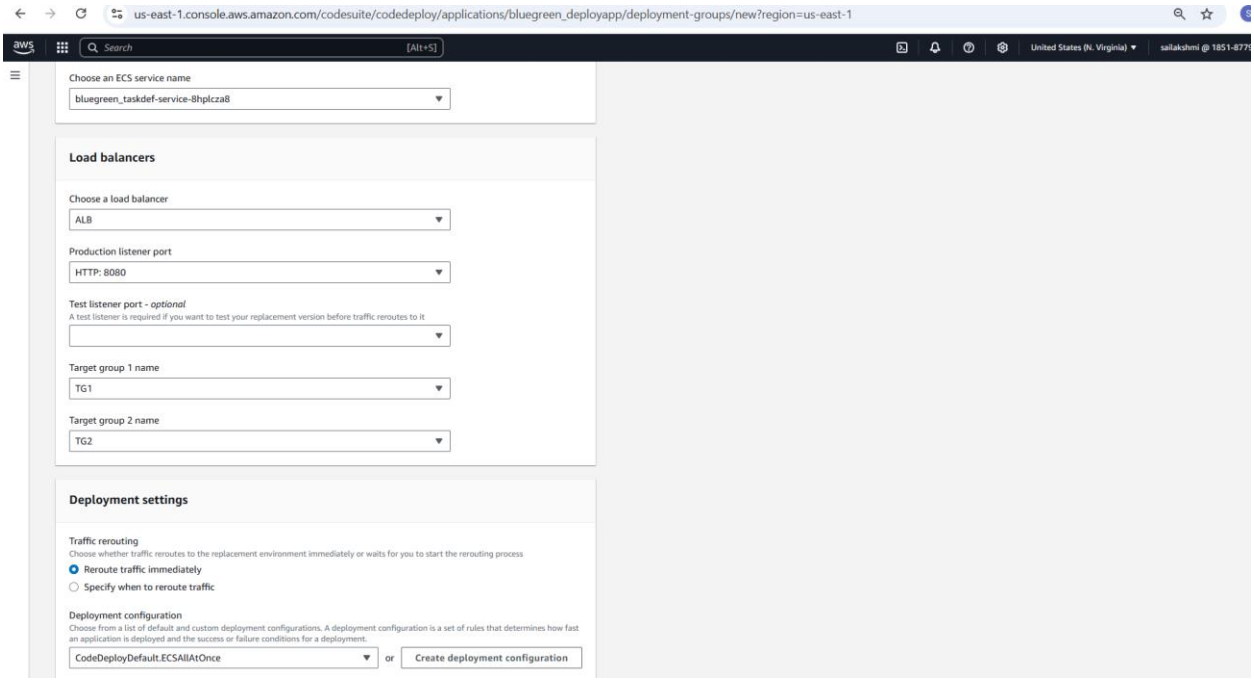


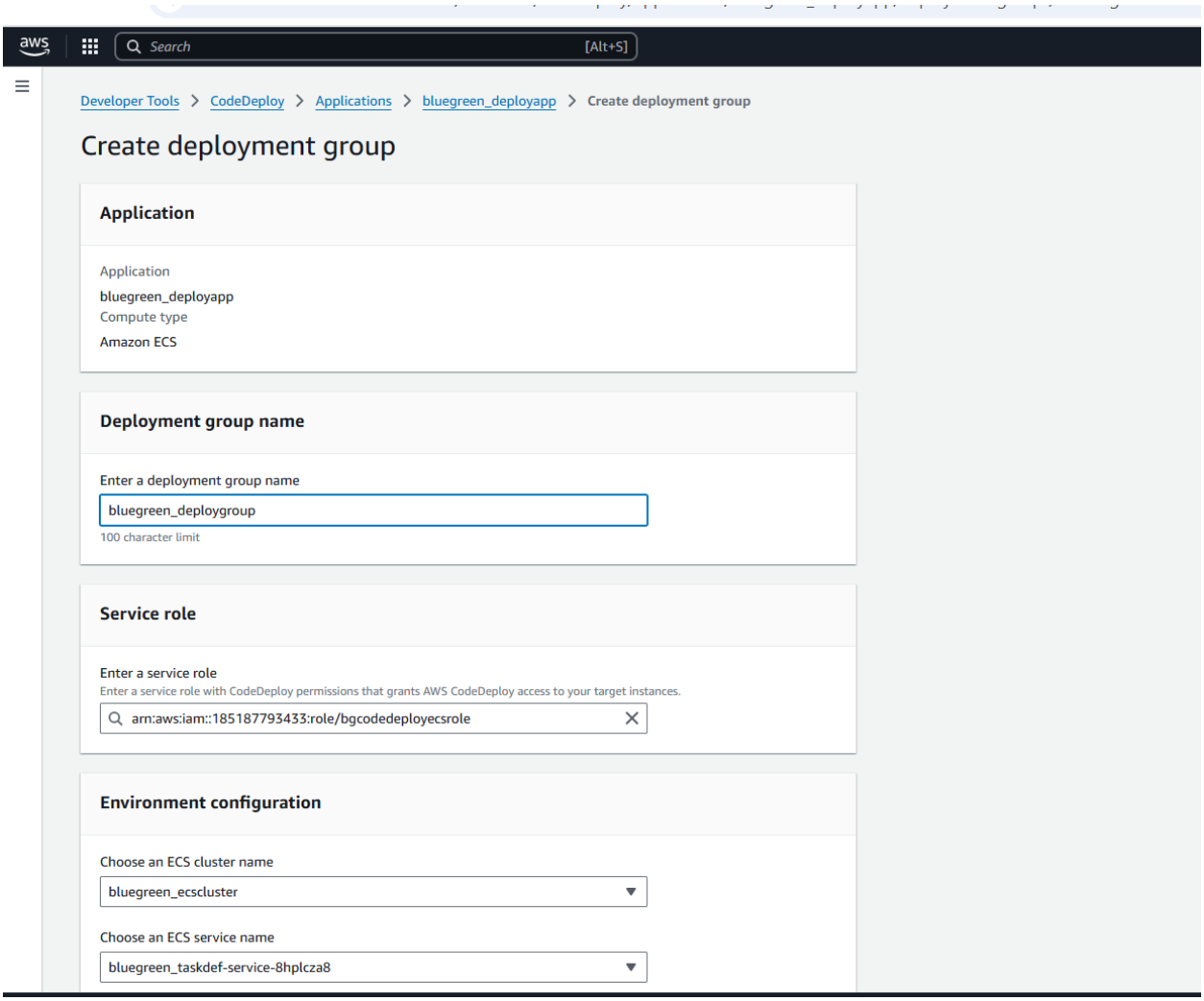
Create CodeDeploy Applications & Deployment Groups

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

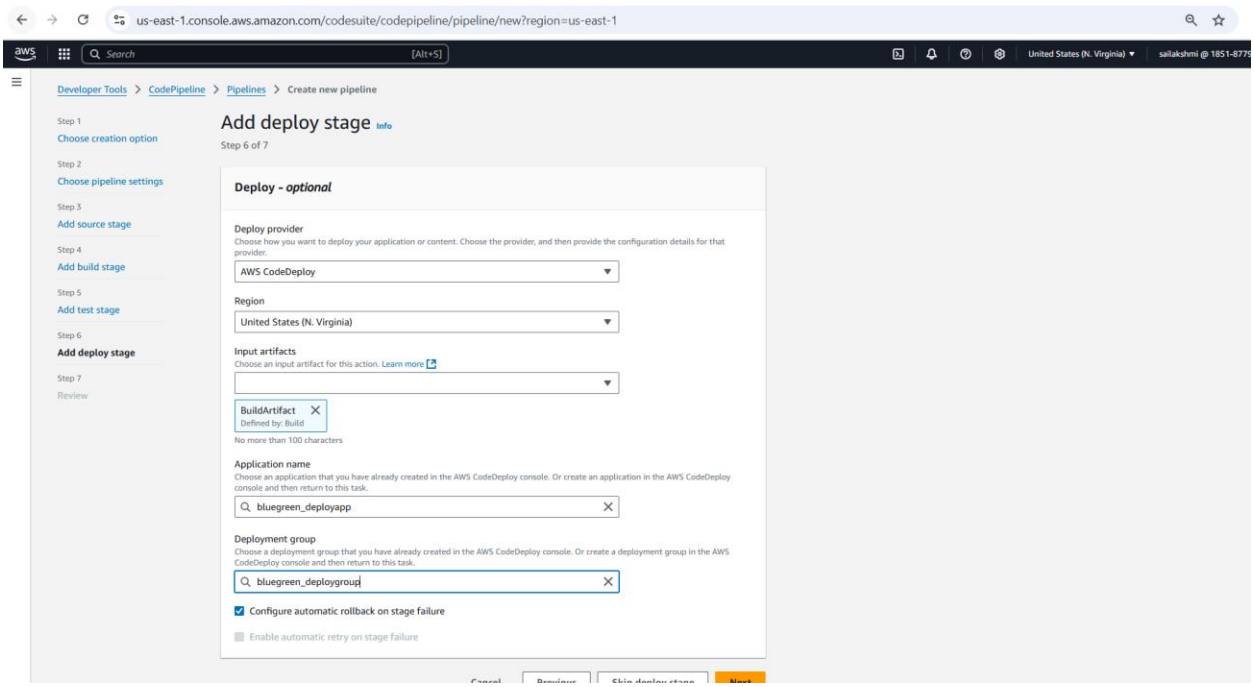


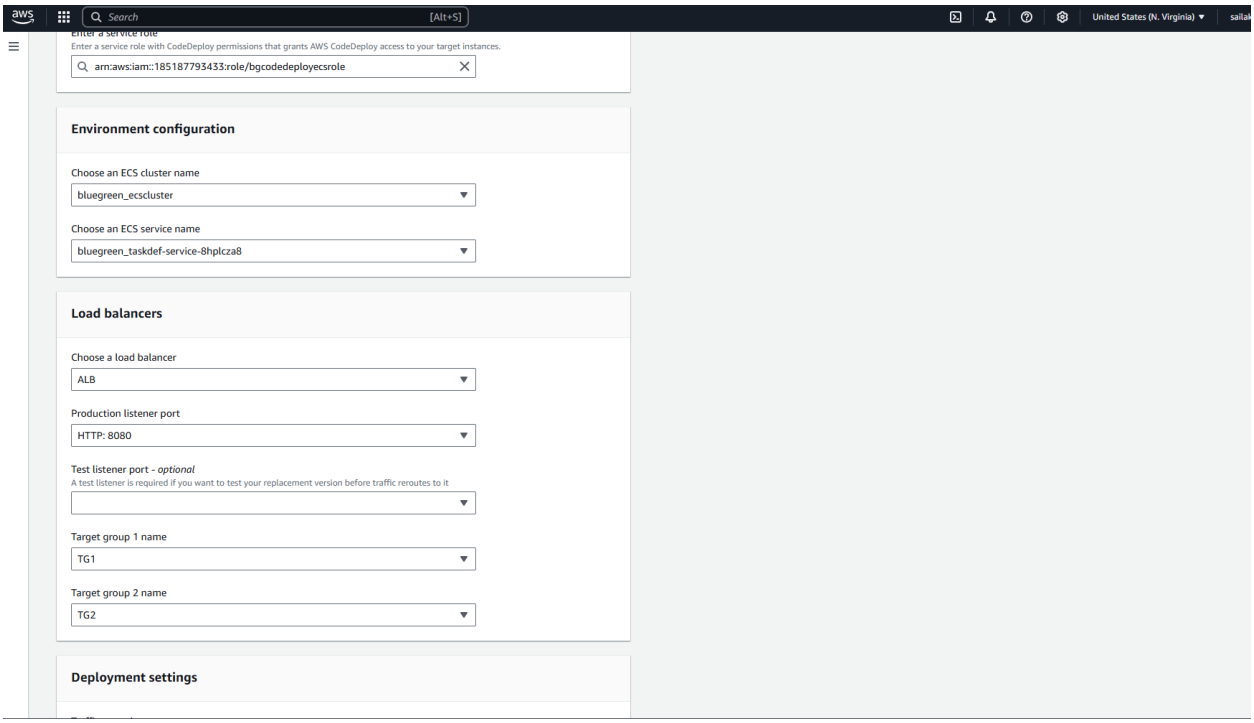
Deployment group

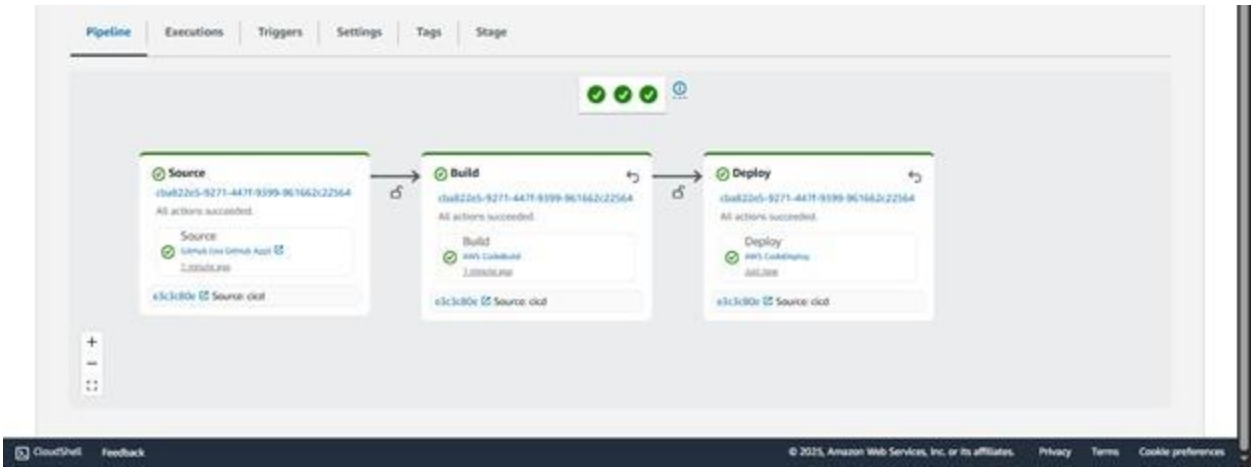




Go to pipeline and add code deploy

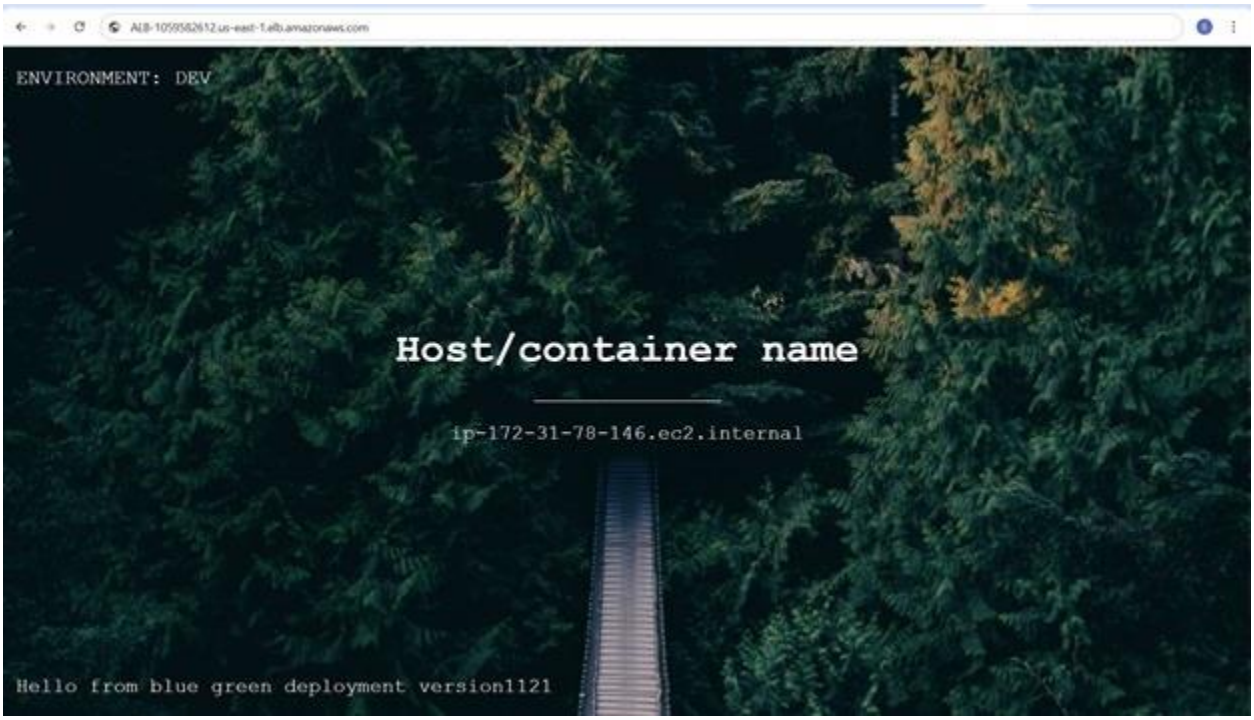

us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipeline/new?region=us-east-1
Search
[Alt+S]
United States (N. Virginia)
Developer Tools > CodePipeline > Pipelines > Create new pipeline
Step 1
Choose creation option
Step 2
Choose pipeline settings
Step 3
Add source stage
Step 4
Add build stage
Step 5
Add test stage
Step 6
Add deploy stage
Step 7
Review
Add deploy stage Info
Step 6 of 7
Deploy - optional
Deploy provider
Choose how you want to deploy your application or content. Choose the provider, and then provide the configuration details for that provider.
AWS CodeDeploy
Region
United States (N. Virginia)
Input artifacts
Choose an input artifact for this action. Learn more
BuildArtifact
Defined by: Build
No more than 100 characters
Application name
Choose an application that you have already created in the AWS CodeDeploy console. Or create an application in the AWS CodeDeploy console and then return to this task.
bluegreen_deployapp
Deployment group
Choose a deployment group that you have already created in the AWS CodeDeploy console. Or create a deployment group in the AWS CodeDeploy console and then return to this task.
bluegreen_deploygroup
Configure automatic rollback on stage failure
Enable automatic retry on stage failure


Search
[Alt+S]
United States (N. Virginia)
Enter a service role
Enter a service role with CodeDeploy permissions that grants AWS CodeDeploy access to your target instances.
arn:aws:iam::185187793433:role/bgcodedeployecsrole
Environment configuration
Choose an ECS cluster name
bluegreen_ecscluster
Choose an ECS service name
bluegreen_taskdef-service-8hplcza8
Load balancers
Choose a load balancer
ALB
Production listener port
HTTP: 8080
Test listener port - optional
A test listener is required if you want to test your replacement version before traffic reroutes to it
Target group 1 name
TG1
Target group 2 name
TG2
Deployment settings



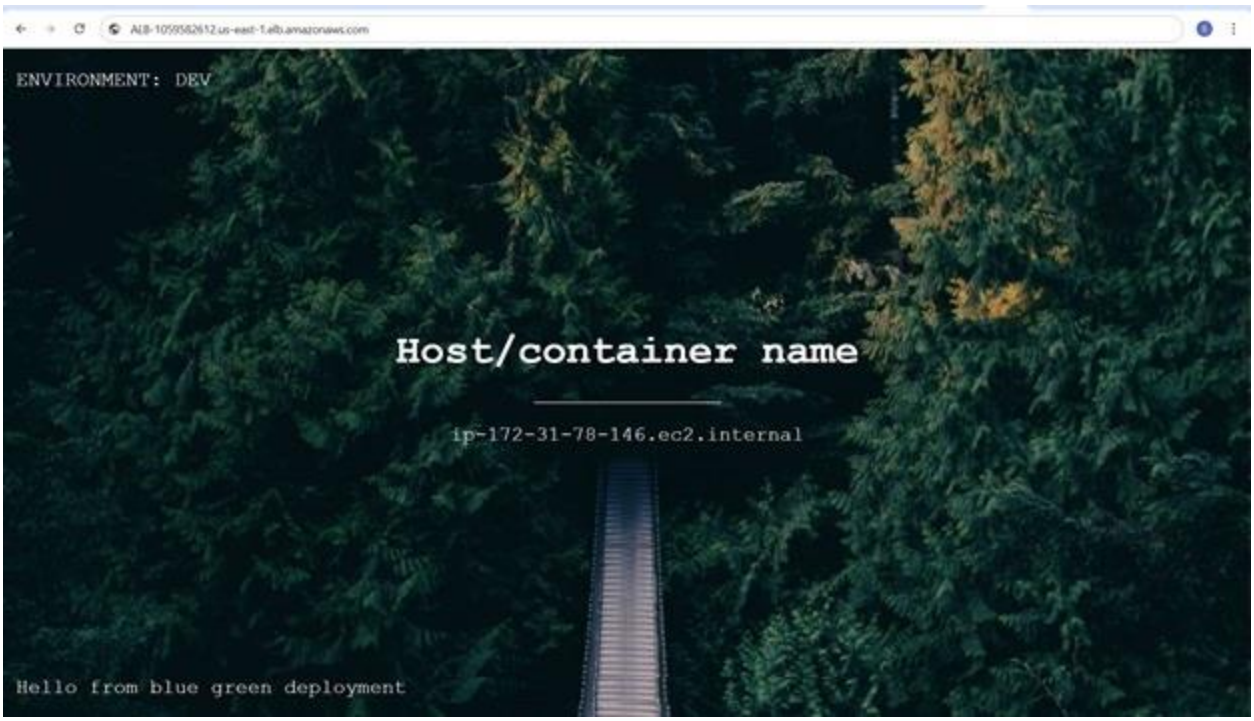
Step:11 After successful deployment

Check the dns name of load balancer in browser



Step: 12 Redeployment Flow

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


ENVIRONMENT: DEV
Host/container name
ip-172-31-78-146.ec2.internal
Hello from blue green deployment