AWS ECS Fargate Infrastructure with GitHub Actions

Table of Contents

Overview

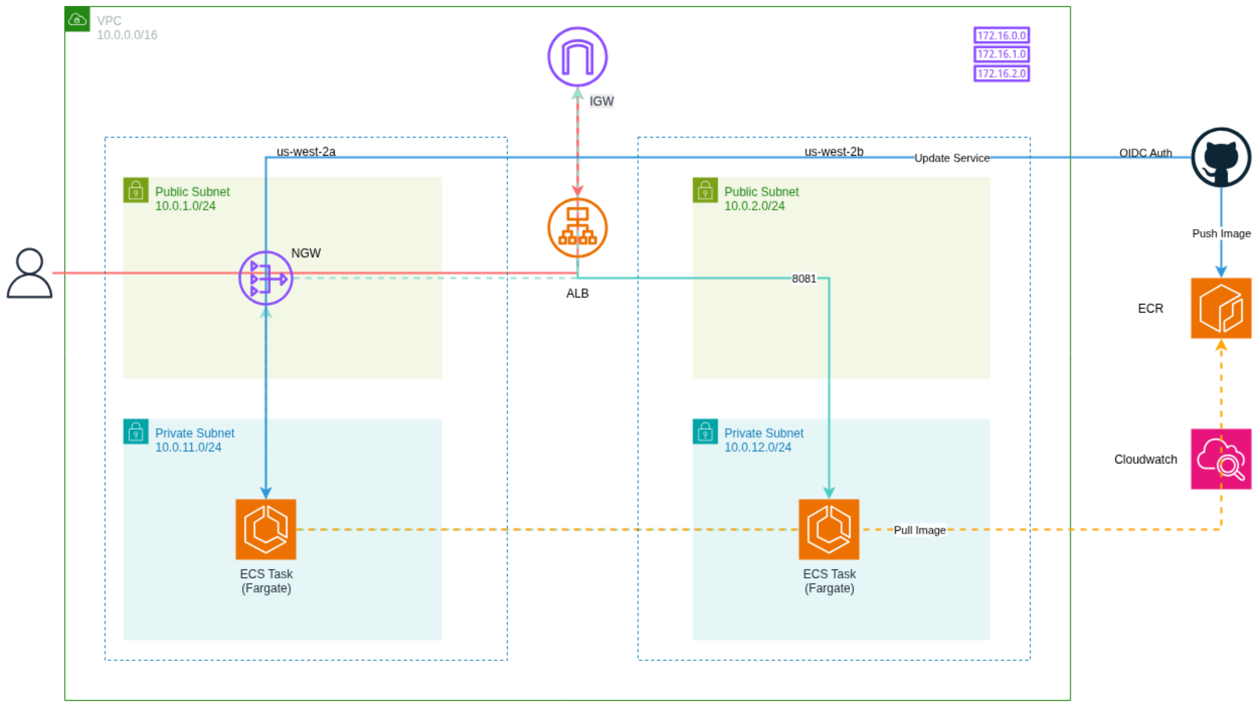
This guide demonstrates deploying a containerized Node.js application on AWS ECS Fargate using the AWS Management Console. The deployment follows a multi-tier architecture with an Application Load Balancer distributing traffic to Fargate tasks running in private subnets across two availability zones.

The infrastructure includes a custom VPC with public and private subnets, security groups implementing least-privilege access, and an automated CI/CD pipeline using GitHub Actions with OIDC authentication. Container images are stored in Amazon ECR, and application logs are centralized in CloudWatch Logs.

Key Components:

• Multi-AZ VPC with public and private subnets
• Application Load Balancer for traffic distribution
• ECS Fargate for serverless container orchestration
• ECR for container image storage
• GitHub OIDC for secure, keyless CI/CD authentication
• IAM roles with least privilege access

Architecture Diagram



Task 12.1: Create Networking Infrastructure

Create VPC

- Navigate to VPC Console
- Create VPC: Name = wu-node-app-vpc, IPv4 CIDR = 10.0.0.0/16
- Enable DNS hostnames and DNS resolution

Create Subnets

- Create Public Subnet 1: Name = wu-node-app-public-subnet-1, AZ = us-west-2a, CIDR = 10.0.1.0/24
- Create Public Subnet 2: Name = wu-node-app-public-subnet-2, AZ = us-west-2b, CIDR = 10.0.2.0/24
- Create Private Subnet 1: Name = wu-node-app-private-subnet-1, AZ = us-west-2a, CIDR = 10.0.11.0/24

- Create Private Subnet 2: Name = wu-node-app-private-subnet-2, AZ = us-west-2b, CIDR = 10.0.12.0/24
- Enable auto-assign public IPv4 for both public subnets

Create Internet Gateway

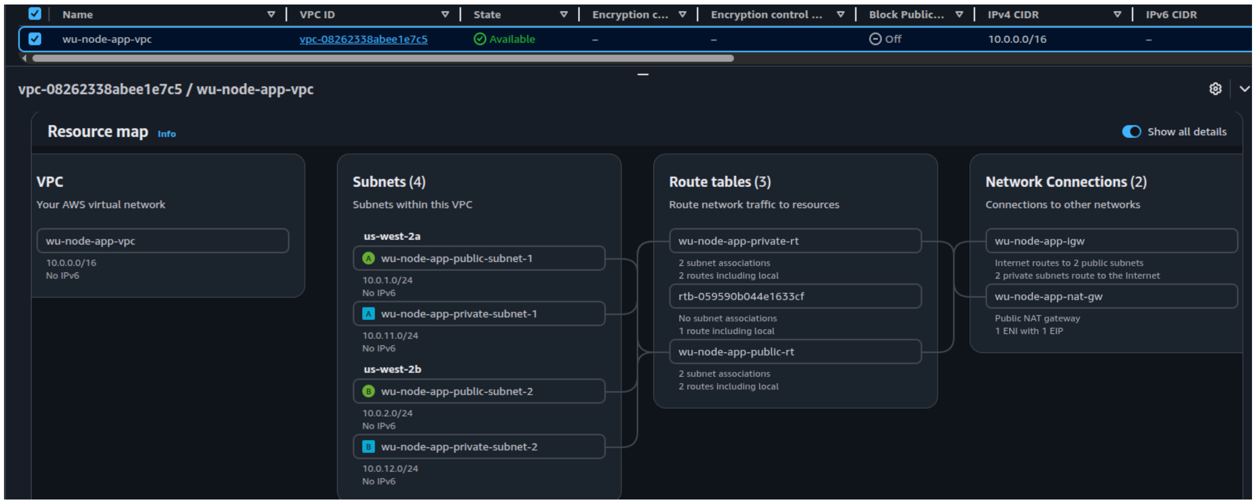
- Create Internet Gateway: Name = wu-node-app-igw
- Attach to wu-node-app-vpc

Create NAT Gateway

- Allocate Elastic IP
- Create NAT Gateway: Name = wu-node-app-nat, Subnet = public-subnet-1
- Associate Elastic IP

Configure Route Tables

- Create Public Route Table: Name = wu-node-app-public-rt
- Add route: 0.0.0.0/0 → Internet Gateway
- Associate with both public subnets
- Create Private Route Table: Name = wu-node-app-private-rt
- Add route: 0.0.0.0/0 → NAT Gateway
- Associate with both private subnets



Name	VPC ID	State	Encryption c...	Encryption control ...	Block Public...	IPv4 CIDR	IPv6 CIDR
wu-node-app-vpc	vpc-08262338abee1e7c5	Available	–	–	Off	10.0.0.0/16	–

vpc-08262338abee1e7c5 / wu-node-app-vpc

Resource map

VPC
Your AWS virtual network
wu-node-app-vpc
10.0.0.0/16
No IPv6

Subnets (4)
Subnets within this VPC
us-west-2a
wu-node-app-public-subnet-1
10.0.1.0/24
No IPv6
wu-node-app-private-subnet-1
10.0.11.0/24
No IPv6
us-west-2b
wu-node-app-public-subnet-2
10.0.2.0/24
No IPv6
wu-node-app-private-subnet-2
10.0.12.0/24
No IPv6

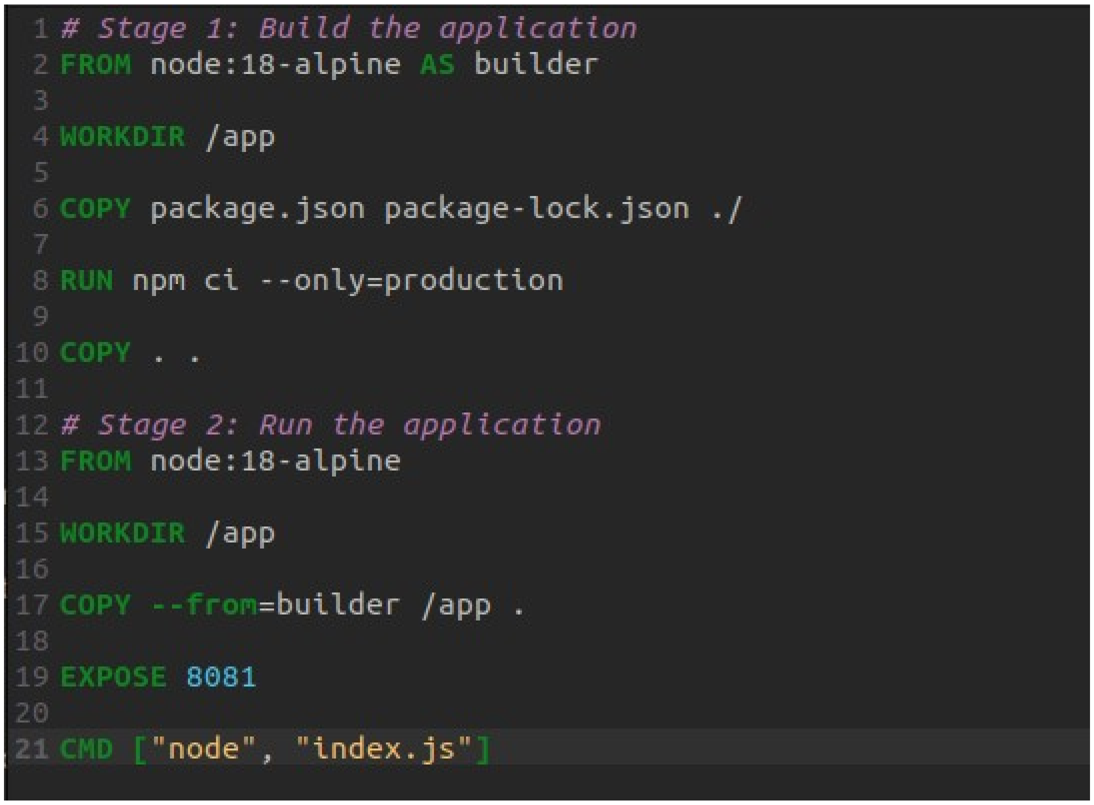
Route tables (3)
Route network traffic to resources
wu-node-app-private-rt
2 subnet associations
2 routes including local
rtb-059590b044e1633cf
No subnet associations
1 route including local
wu-node-app-public-rt
2 subnet associations
2 routes including local

Network Connections (2)
Connections to other networks
wu-node-app-igw
Internet routes to 2 public subnets
2 private subnets route to the Internet
wu-node-app-nat-gw
Public NAT gateway
1 ENI with 1 EIP

Task 12.2: Prepare Application Dockerfile

Create Dockerfile

- Base image: node:18-alpine
- Set working directory to /app
- Copy package files and install dependencies
- Copy application code
- Expose port 8081
- Set CMD to start application



```
# Stage 1: Build the application
FROM node:18-alpine AS builder

WORKDIR /app

COPY package.json package-lock.json ./

RUN npm ci --only=production

COPY . .

# Stage 2: Run the application
FROM node:18-alpine

WORKDIR /app

COPY --from=builder /app .

EXPOSE 8081

CMD ["node", "index.js"]
```

Task 12.3: Create Security Groups

ALB Security Group

- Navigate to EC2 → Security Groups
- Create Security Group: Name = wu-node-app-alb-sg, VPC = wu-node-app-vpc
- Inbound rule: HTTP (80) from 0.0.0.0/0
- Outbound rule: All traffic to 0.0.0.0/0

ECS Tasks Security Group

- Create Security Group: Name = wu-node-app-ecs-sg, VPC = wu-node-app-vpc
- Inbound rule: Custom TCP (8081) from ALB security group
- Outbound rule: All traffic to 0.0.0.0/0

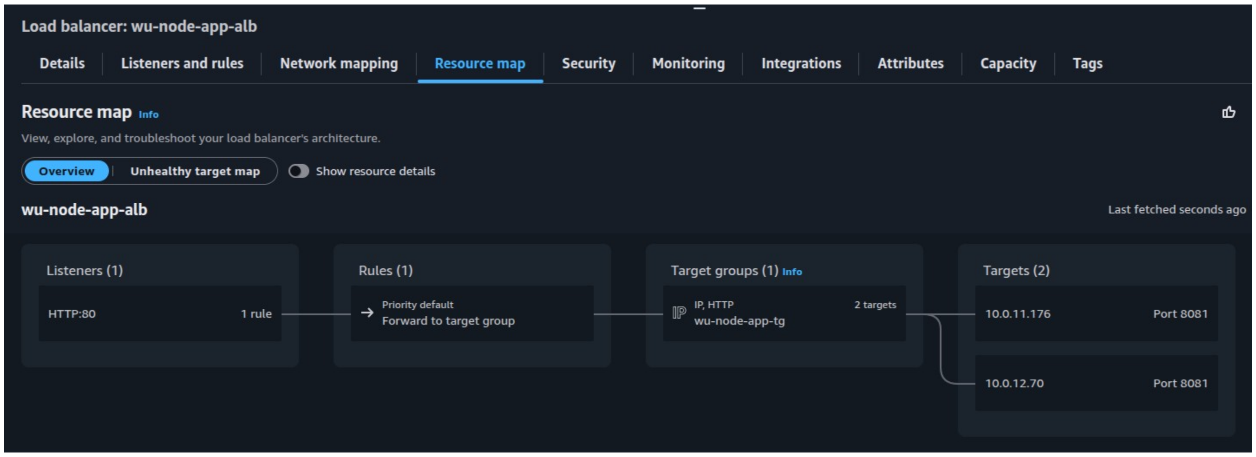
Task 12.4: Create Target Group and Load Balancer

Create Target Group

- Navigate to EC2 → Target Groups
- Target type: IP addresses
- Name = wu-node-app-tg, Protocol = HTTP, Port = 8081
- VPC = wu-node-app-vpc
- Health check path = /
- Create target group (no targets yet)

Create Application Load Balancer

- Navigate to EC2 → Load Balancers
- Create Application Load Balancer: Name = wu-node-app-alb
- Scheme: Internet-facing
- Network: Select wu-node-app-vpc, select both public subnets
- Security group: wu-node-app-alb-sg
- Listener: HTTP (80) forward to wu-node-app-tg
- Create load balancer



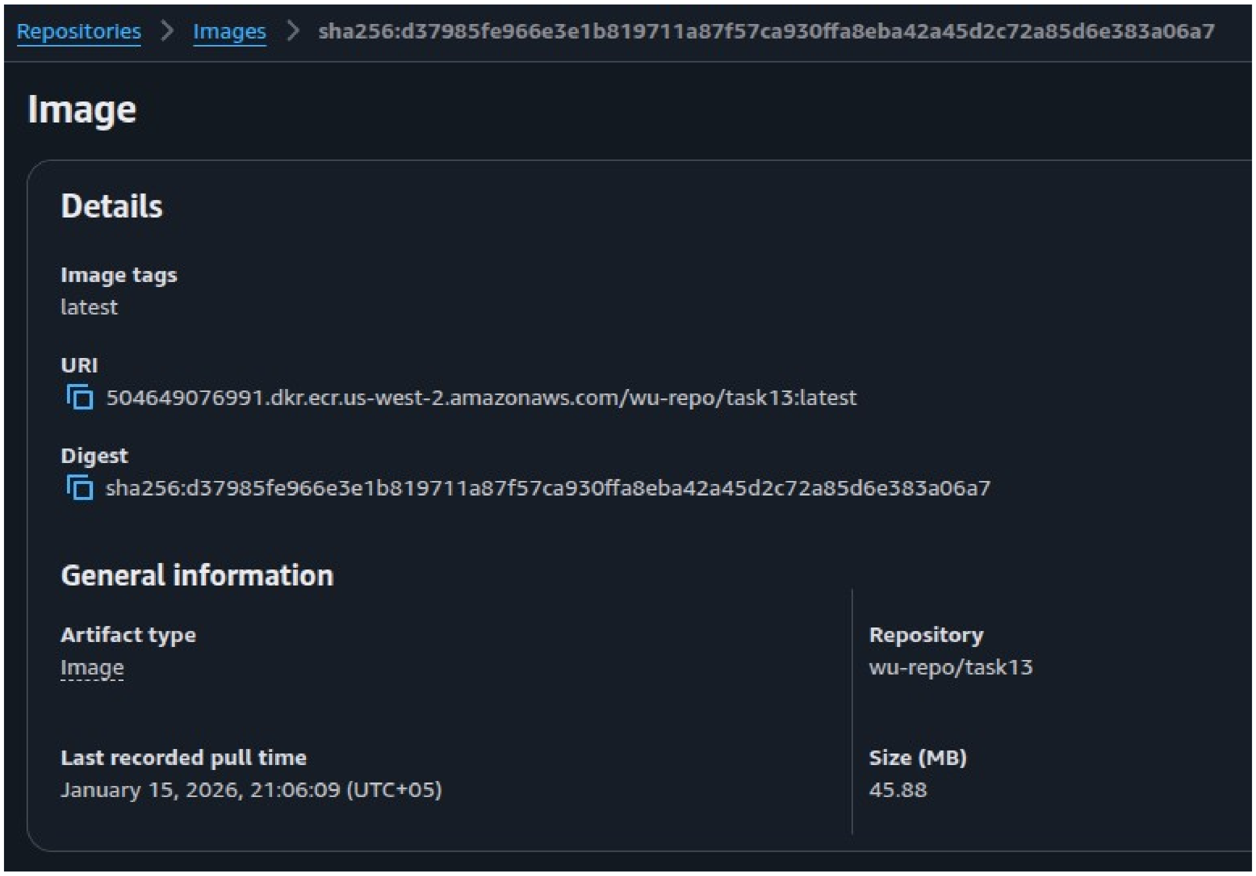
Task 12.5: Create ECR Repository and Push Image

Create ECR Repository

- Navigate to ECR Console
- Create repository: Name = wu-repo/task13
- Image scan on push: Enabled
- Create repository

Build and Push Docker Image

- Authenticate Docker to ECR: aws ecr get-login-password --region us-west-2
- Build image: docker build -t wu-repo/task13:latest .
- Tag image: docker tag wu-repo/task13:latest <account>.dkr.ecr.us-west-2.amazonaws.com/wu-repo/task13:latest
- Push image: docker push <account>.dkr.ecr.us-west-2.amazonaws.com/wu-repo/task13:latest


Repositories > Images > sha256:d37985fe966e3e1b819711a87f57ca930ffa8eba42a45d2c72a85d6e383a06a7

Image

Details

Image tags
latest

URI
504649076991.dkr.ecr.us-west-2.amazonaws.com/wu-repo/task13:latest

Digest
sha256:d37985fe966e3e1b819711a87f57ca930ffa8eba42a45d2c72a85d6e383a06a7

General information

Artifact type
Image

Repository
wu-repo/task13

Last recorded pull time
January 15, 2026, 21:06:09 (UTC+05)

Size (MB)
45.88

Task 12.6: Create ECS Cluster, Task Definition and Service

Create ECS Cluster

- Navigate to ECS Console
- Create cluster: Name = wu-node-app-cluster
- Infrastructure: AWS Fargate (serverless)

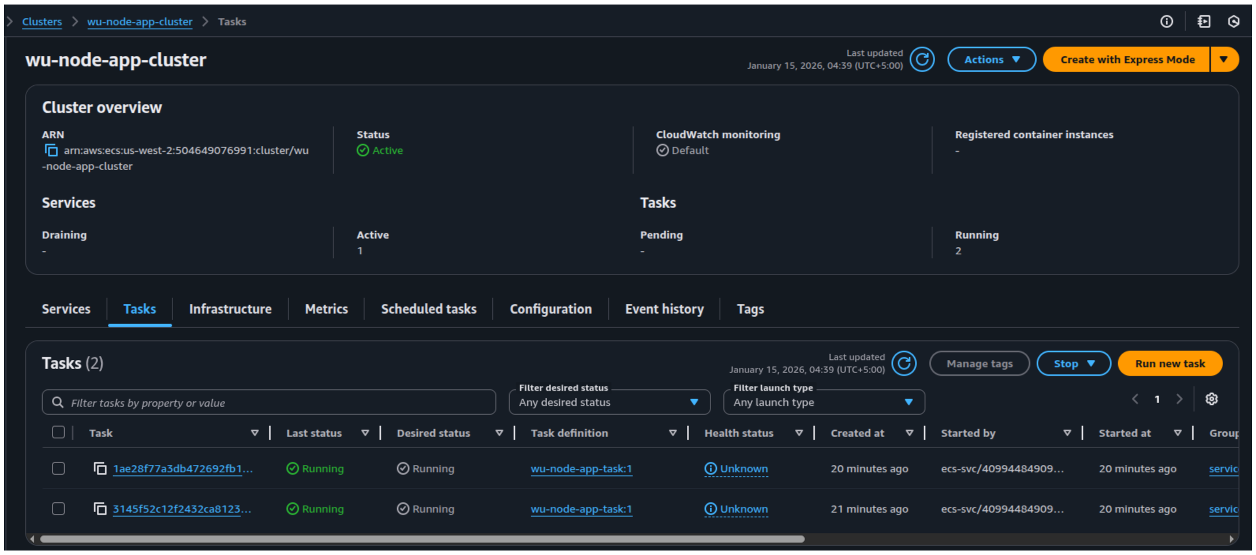
Create Task Definition

- Create new task definition: Name = wu-node-app-task
- Launch type: Fargate
- OS/Architecture: Linux/X86_64
- CPU: 0.25 vCPU (256 units)
- Memory: 0.5 GB (512 MB)
- Task role: Create new role (wu-node-app-ecs-task-role)

- Task execution role: Create new role (wu-node-app-ecs-task-execution-role)
- Container name: wu-node-app-container
- Image URI: <account>.dkr.ecr.us-west-2.amazonaws.com/wu-repo/task13:latest
- Port mapping: 8081 TCP
- Log collection: Enable CloudWatch Logs
- Log group: /ecs/wu-node-app

Create ECS Service

- In cluster, create service: Name = wu-node-app-service
- Launch type: Fargate
- Task definition: wu-node-app-task (latest)
- Desired tasks: 2
- VPC: wu-node-app-vpc
- Subnets: Select both private subnets
- Security group: wu-node-app-ecs-sg
- Public IP: Disabled
- Load balancer: Application Load Balancer you just created.
- Select wu-node-app-alb
- Target group: wu-node-app-tg
- Create service



wu-node-app-cluster

Last updated January 15, 2026, 04:39 (UTC+5:00)

Cluster overview

ARN	Status	CloudWatch monitoring	Registered container instances
arn:aws:ecs:us-west-2:504649076991:cluster/wu-node-app-cluster	Active	Default	-

Services

Draining	Active
-	1

Tasks

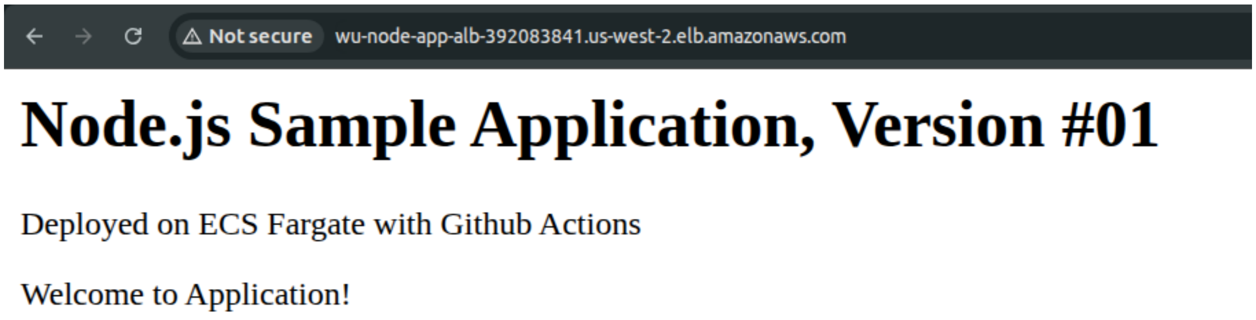
Pending	Running
-	2

Tasks (2)

Last updated January 15, 2026, 04:39 (UTC+5:00)

Filter desired status: Any desired status
Filter launch type: Any launch type

Task	Last status	Desired status	Task definition	Health status	Created at	Started by	Started at	Group
1ae28f77a3db472692fb1…	Running	Running	wu-node-app-task:1	Unknown	20 minutes ago	ecs-svc/40994484909…	20 minutes ago	servic
3145f52c12f2432ca8123…	Running	Running	wu-node-app-task:1	Unknown	21 minutes ago	ecs-svc/40994484909…	20 minutes ago	servic


Not secure wu-node-app-alb-392083841.us-west-2.elb.amazonaws.com

Node.js Sample Application, Version #01

Deployed on ECS Fargate with Github Actions

Welcome to Application!

Task 12.7: Configure IAM for GitHub Actions (OIDC)

Create OIDC Provider

- Navigate to IAM → Identity Providers
- Add provider: OpenID Connect
- Provider URL: https://token.actions.githubusercontent.com
- Audience: sts.amazonaws.com
- Add provider

Create GitHub Actions Role

- Navigate to IAM → Roles → Create Role
- Trusted entity: Web identity
- Identity provider: token.actions.githubusercontent.com
- Audience: sts.amazonaws.com
- GitHub organization: <your-org>
- GitHub repository: <your-repo>
- Role name: wu-node-app-github-actions-role

Attach Permissions to GitHub Actions Role

- ECR permissions: GetAuthorizationToken, BatchCheckLayerAvailability, PutImage, InitiateLayerUpload, UploadLayerPart, CompleteLayerUpload

• ECS permissions: RegisterTaskDefinition, DescribeTaskDefinition, UpdateService, DescribeServices
• IAM permissions: PassRole (for task execution and task roles)

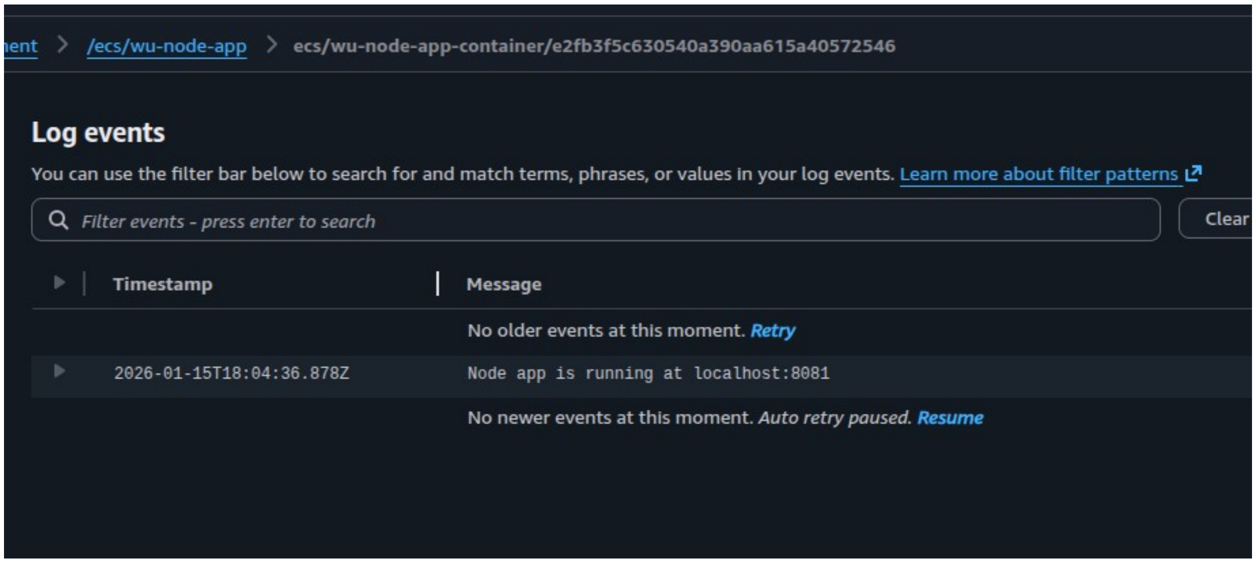
Update ECS Task Execution Role

• Navigate to the auto-created task execution role
• Verify attached policy: AmazonECSTaskExecutionRolePolicy
• Permissions: ECR image pull, CloudWatch Logs write access

Update ECS Task Role

• Navigate to the auto-created task role
• Add permissions: CloudWatch Logs write access for application logs

Later on, you can see the logs being registered in your cloudwatch log group


ent > /ecs/wu-node-app > ecs/wu-node-app-container/e2fb3f5c630540a390aa615a40572546
Log events
Learn more about filter patterns
Filter events - press enter to search
Clear

You can use the filter bar below to search for and match terms, phrases, or values in your log events.

```
Timestamp                   Message
                            No older events at this moment. Retry
2026-01-15T18:04:36.878Z    Node app is running at localhost:8081
                            No newer events at this moment. Auto retry paused. Resume
```

Task 12.8: Create GitHub Actions Workflow

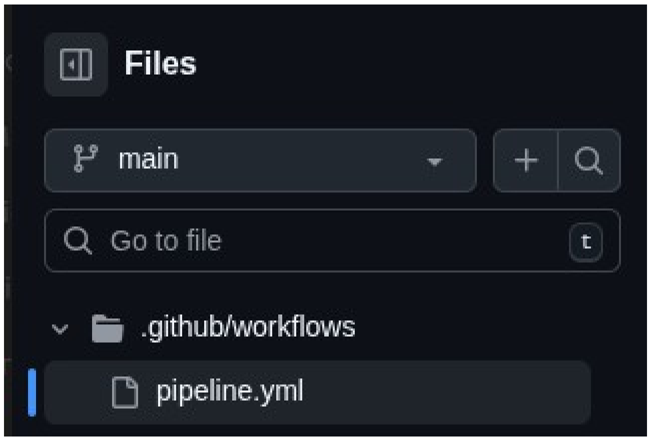
Configure GitHub Secrets and Variables

• Navigate to Repository → Settings → Secrets and variables → Actions
• Add secret: AWS_ROLE_ARN = <GitHub Actions role ARN>

- Add variables: AWS_REGION = us-west-2
- AWS_ACCOUNT_ID = <your-account-id>
- ECR_REPOSITORY = wu-repo/task13
- ECS_CLUSTER = wu-node-app-cluster
- ECS_SERVICE = wu-node-app-service
- CONTAINER_NAME = wu-node-app-container

Create Workflow File

- Create .github/workflows/deploy.yml in repository
- Trigger: Push to main branch
- Permissions: id-token write, contents read
- Configure stages as outlined below



Pipeline Stages

- Stage 1 - Checkout: Clone repository code
- Stage 2 - AWS Authentication: Use OIDC to get temporary credentials
- Stage 3 - ECR Login: Authenticate Docker with ECR
- Stage 4 - Build & Push: Build image with commit SHA tag, push to ECR
- Stage 5 - Download Task Definition: Fetch current task definition from ECS
- Stage 6 - Clean Task Definition: Remove AWS-managed fields using jq
- Stage 7 - Render Task Definition: Update image URI with new image
- Stage 8 - Deploy to ECS: Register new task definition, update service, wait for stability

Task 12.9: Test Pipeline Deployment

Trigger Workflow

- Make a small code change in application
- Commit and push to main branch
- Navigate to Actions tab to monitor workflow

Verify Pipeline Execution

- Check all 8 stages complete successfully
- Verify new image appears in ECR with commit SHA tag
- Check ECS service registers new task definition revision
- Confirm new tasks start and old tasks are replaced

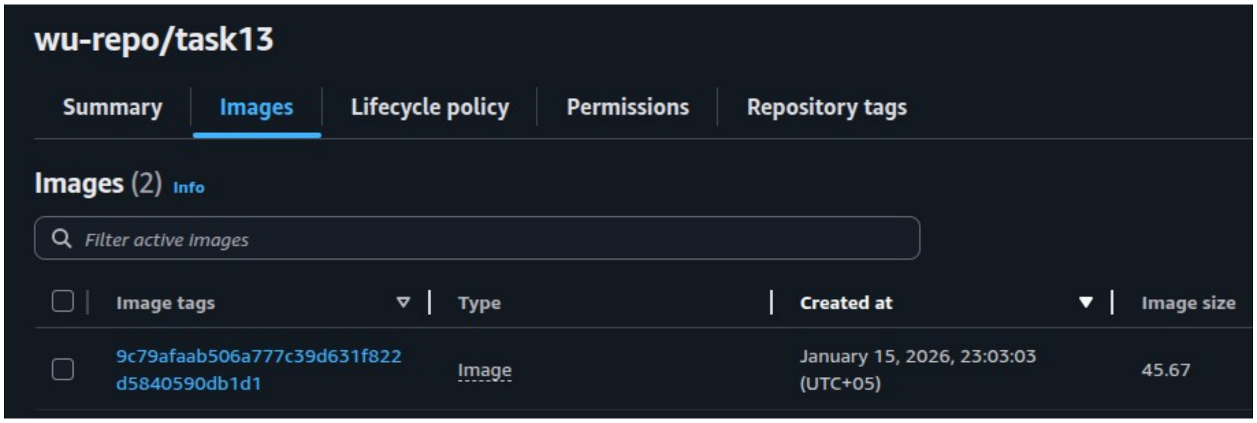
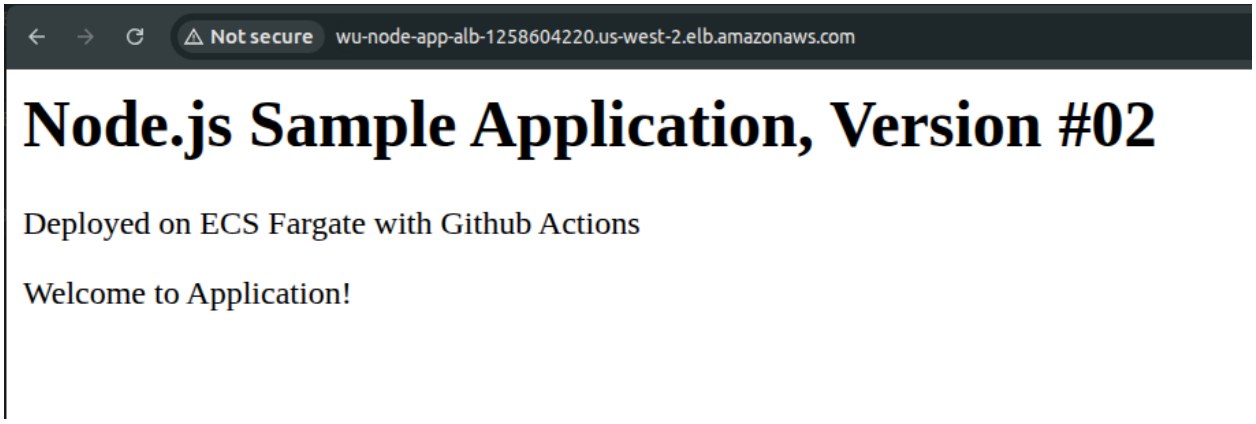


Image tags	Type	Created at	Image size
9c79afaab506a777c39d631f822d5840590db1d1	Image	January 15, 2026, 23:03:03 (UTC+05)	45.67

Validate Deployment

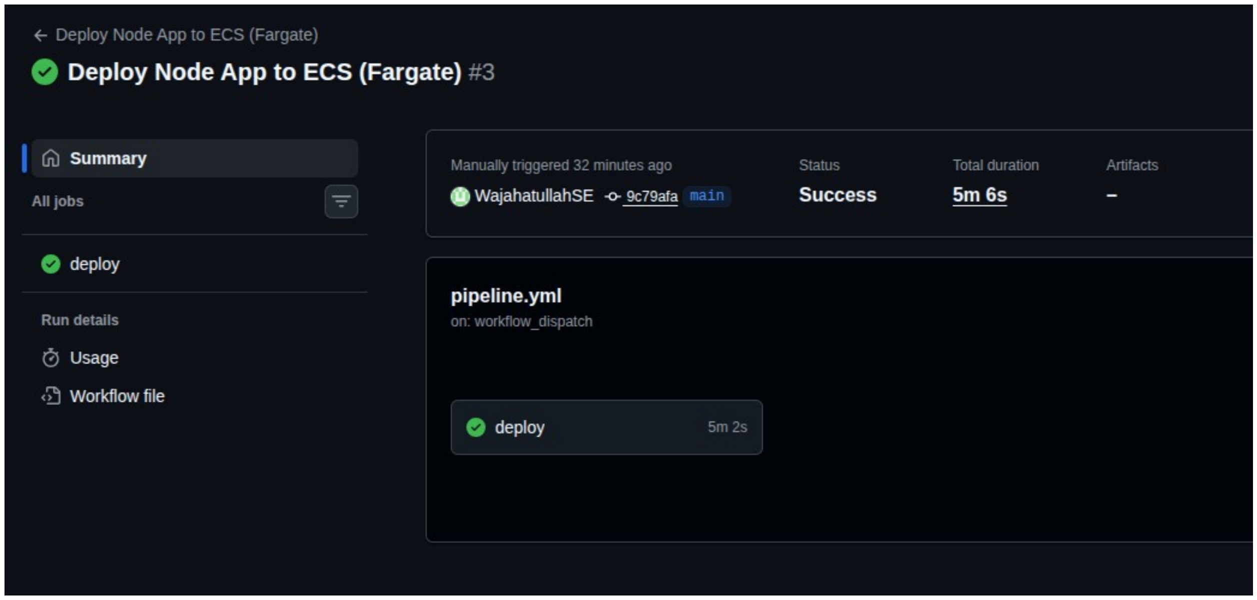
- Access ALB DNS name in browser
- Verify application shows updated code changes
- Check CloudWatch Logs for new log streams
- Confirm target group shows 2 healthy targets


Not secure wu-node-app-alb-1258604220.us-west-2.elb.amazonaws.com

Node.js Sample Application, Version #02

Deployed on ECS Fargate with Github Actions

Welcome to Application!


← Deploy Node App to ECS (Fargate)

Deploy Node App to ECS (Fargate) #3

Manually triggered 32 minutes ago
WajahatullahSE 9c79afa main

Status: Success

Total duration: 5m 6s

Artifacts: –

pipeline.yml
on: workflow_dispatch

deploy 5m 2s