Brain-Tasks-App Deployment Guide

# 1. Project Overview

This guide describes the step-by-step process to deploy the Brain-Tasks-App using Amazon EC2, Docker, Amazon ECR, Kubernetes (EKS), CodePipeline, CodeBuild, and CloudWatch. The app is a static React site, and the deployment includes full CI/CD using AWS services.

# 2. Prerequisites

- AWS account with admin permissions  
- IAM user/roles set up for CodePipeline, EKS, and CodeBuild  
- EKS cluster created and working (v1.29+)  
- AWS CLI and kubectl configured  
- EC2 Amazon Linux 2 instance to run commands  
- Docker installed on EC2 instance

# 3. Deployment Steps

## Step 1: Clone the Repository

Clone the React static site repository:

git clone https://github.com/Vennilavan12/Brain-Tasks-App.git

## Step 2: Create Dockerfile

Create a Dockerfile inside the root of your project folder with the following content:

FROM nginx:alpine  
COPY ./dist /usr/share/nginx/html  
EXPOSE 80  
CMD ["nginx", "-g", "daemon off;"]

## Step 3: Build Docker Image and Push to ECR

Run the following commands from your EC2 instance:

aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 977099019525.dkr.ecr.ap-south-1.amazonaws.com/brain-tasks-app  
docker build -t brain-tasks-app .  
docker tag brain-tasks-app:latest 977099019525.dkr.ecr.ap-south-1.amazonaws.com/brain-tasks-app:latest  
docker push 977099019525.dkr.ecr.ap-south-1.amazonaws.com/brain-tasks-app:latest

## Step 4: Kubernetes Setup (EKS)

Create `deployment.yaml` and `service.yaml` files and apply them using kubectl:

kubectl apply -f deployment.yaml -f service.yaml

## Step 5: Configure CodeBuild

Create a `buildspec.yml` file with Docker build and push steps. Ensure the role has access to ECR and S3. Use `privileged: true` in the build environment.

## Step 6: CodePipeline Setup

Create a pipeline with 3 stages:  
1. Source (GitHub)  
2. Build (CodeBuild)  
3. Deploy (EKS via kubectl)  
Attach the required IAM role and add the role to EKS aws-auth config map.

## Step 7: IAM Role for Pipeline Access to EKS

Edit aws-auth config map to add pipeline IAM role under `system:masters` group so kubectl can deploy via pipeline.

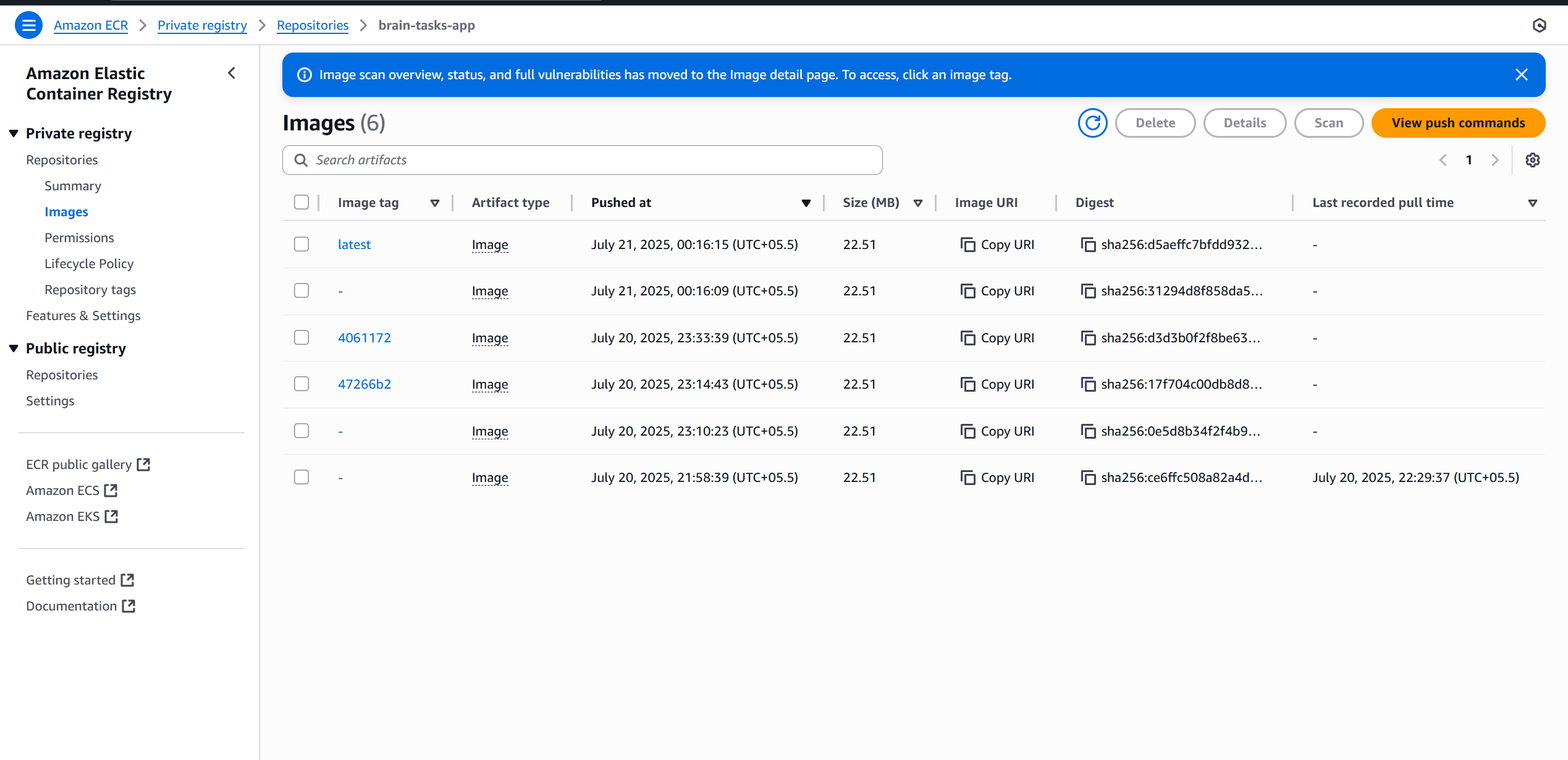
## Step 8: Setup CloudWatch

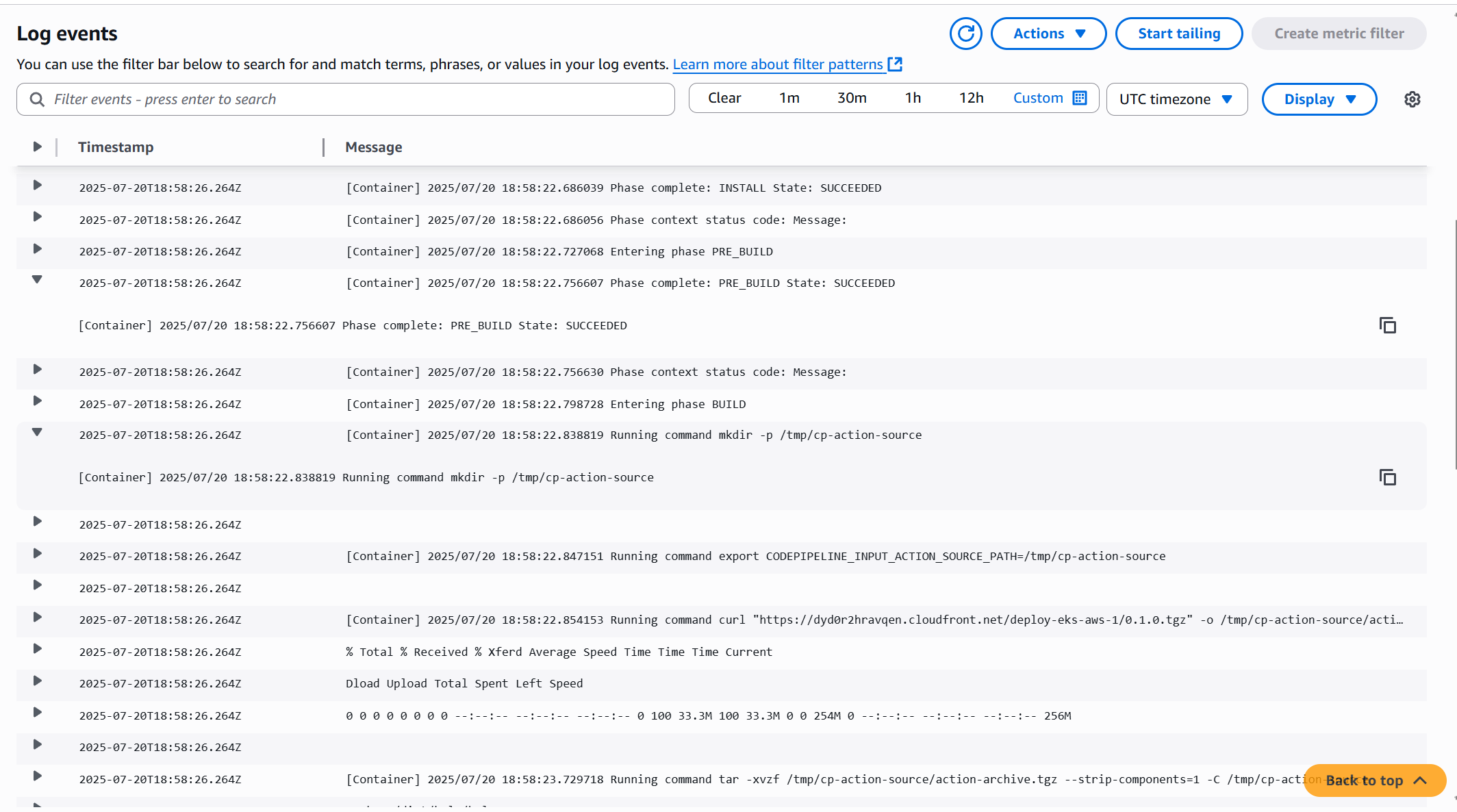
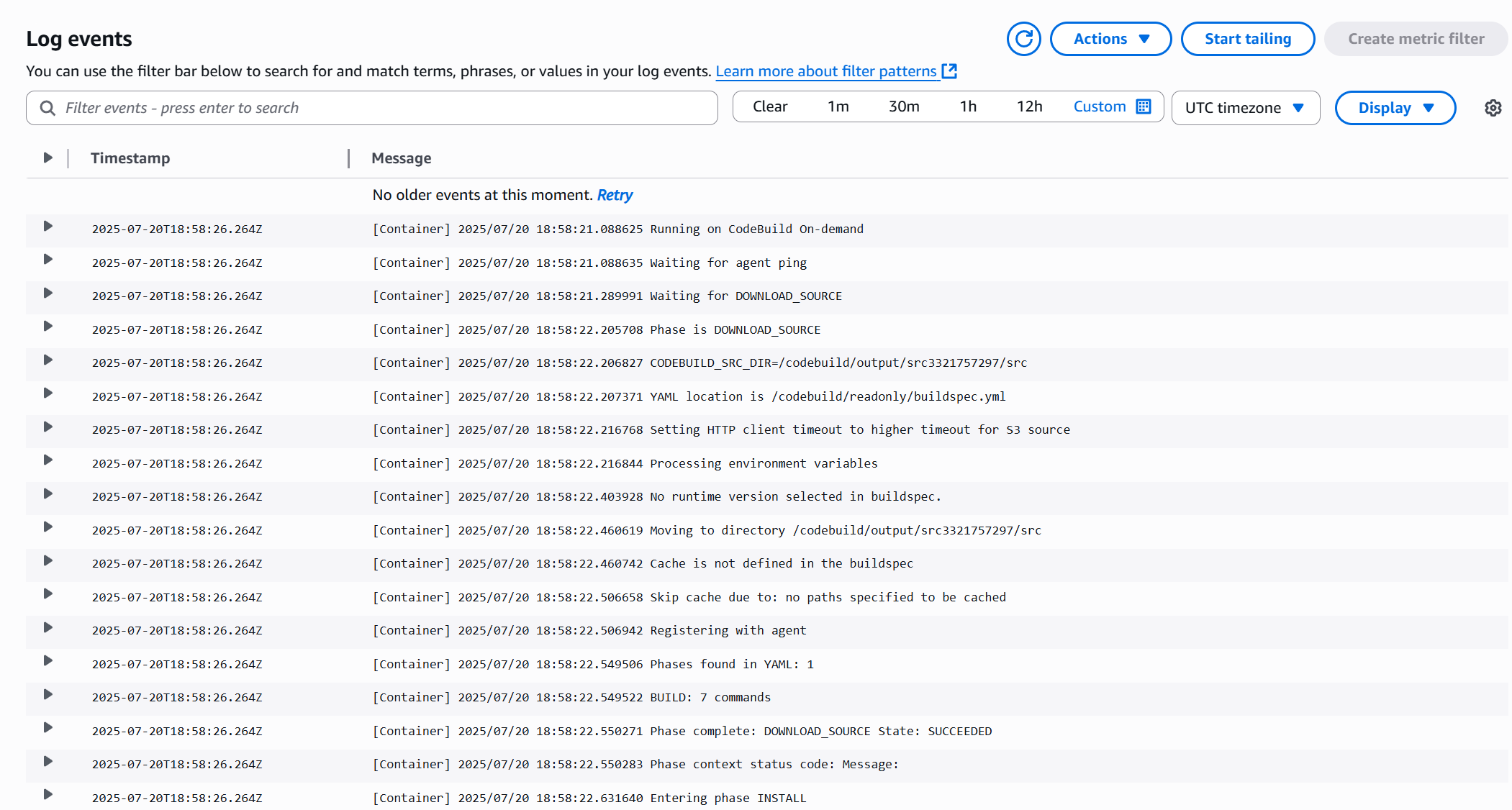
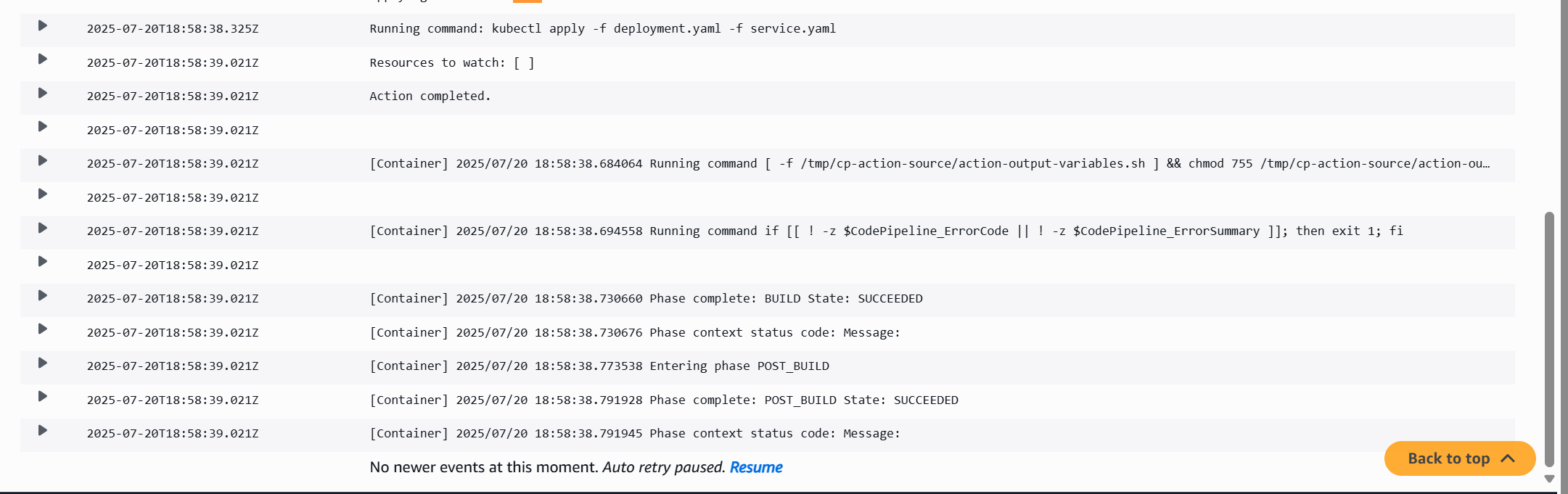
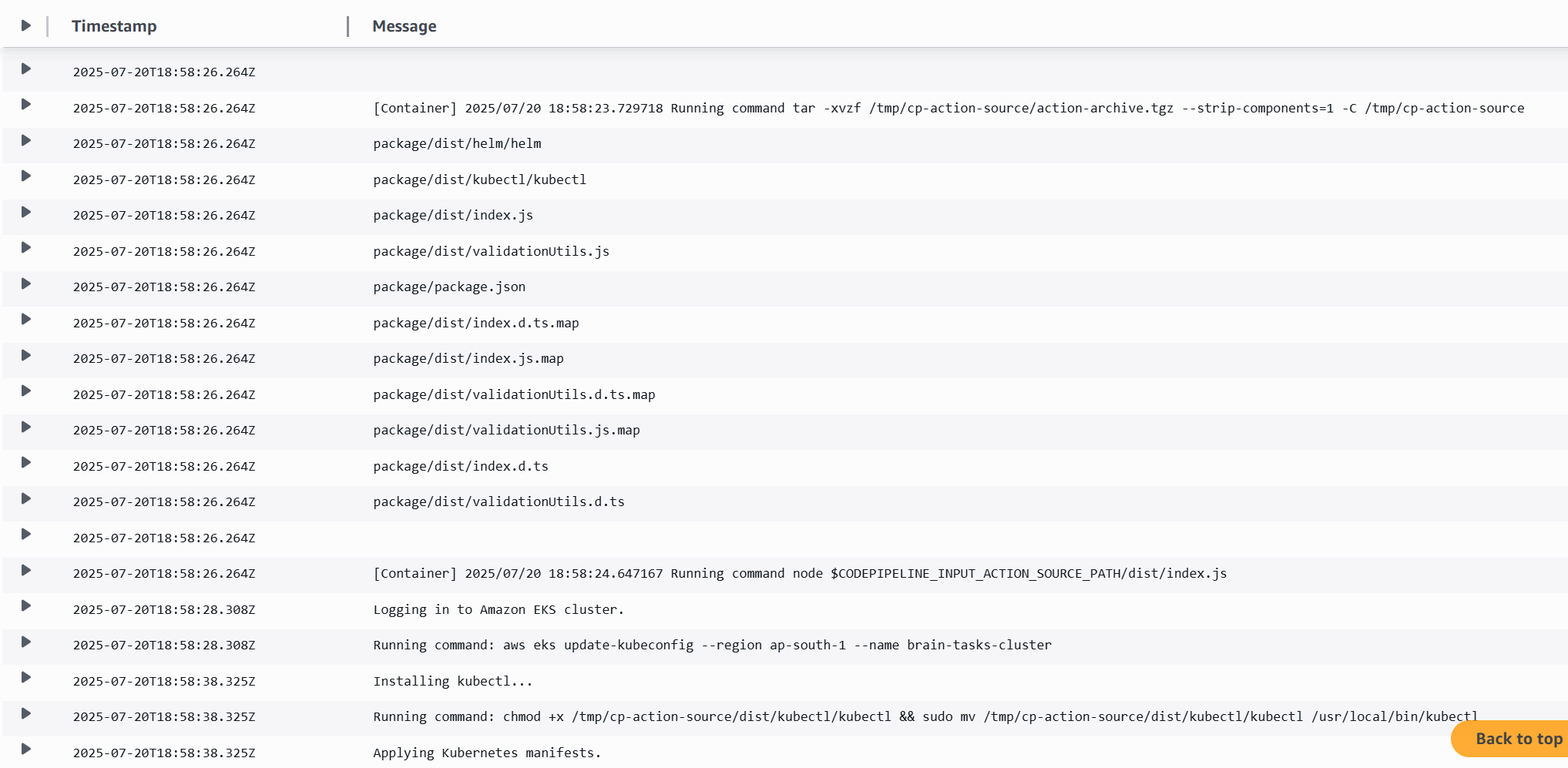
Ensure logs from CodeBuild are streamed to CloudWatch. Check build history and failures in the AWS Console.

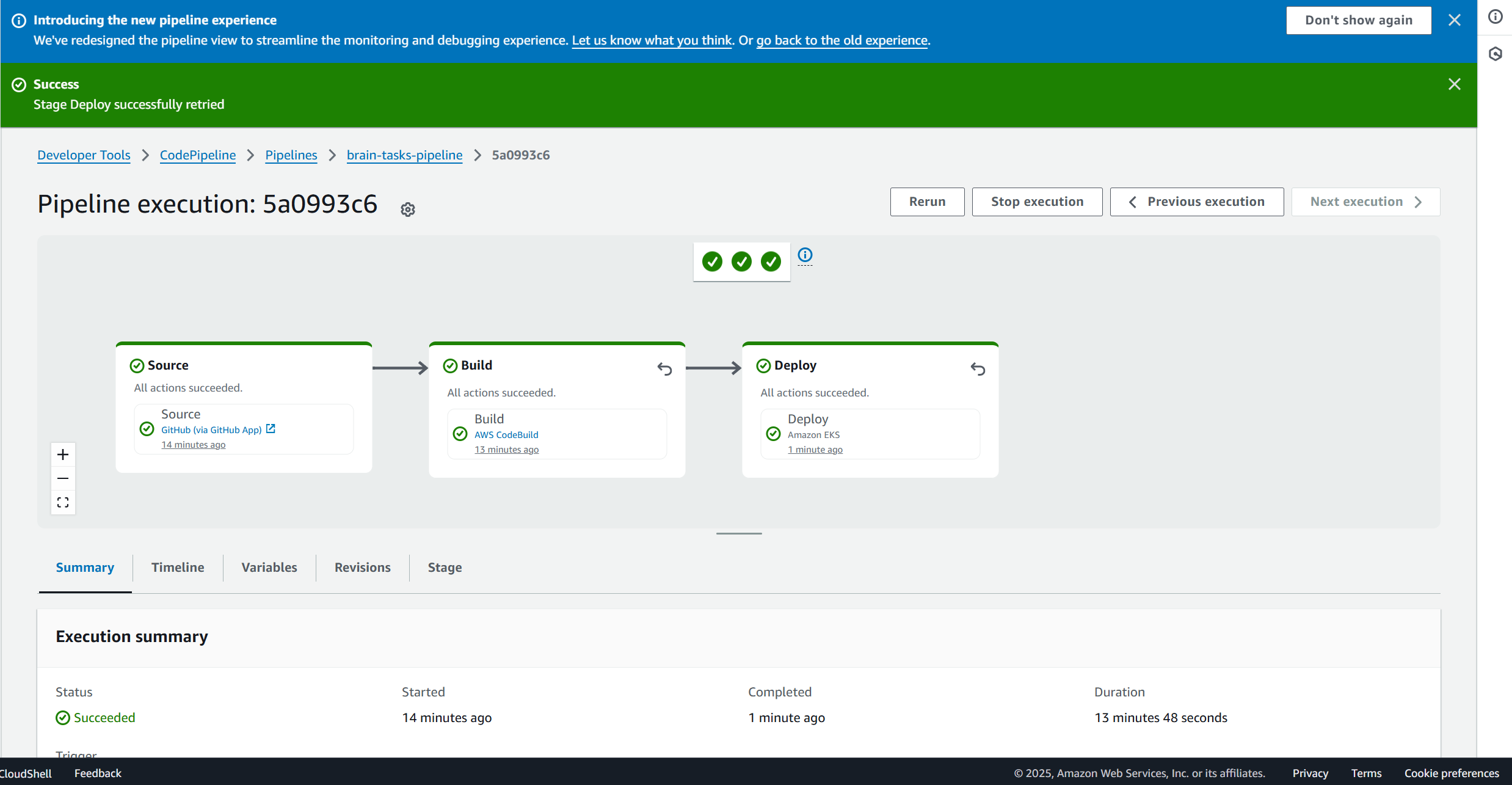
## Step 9: Verification

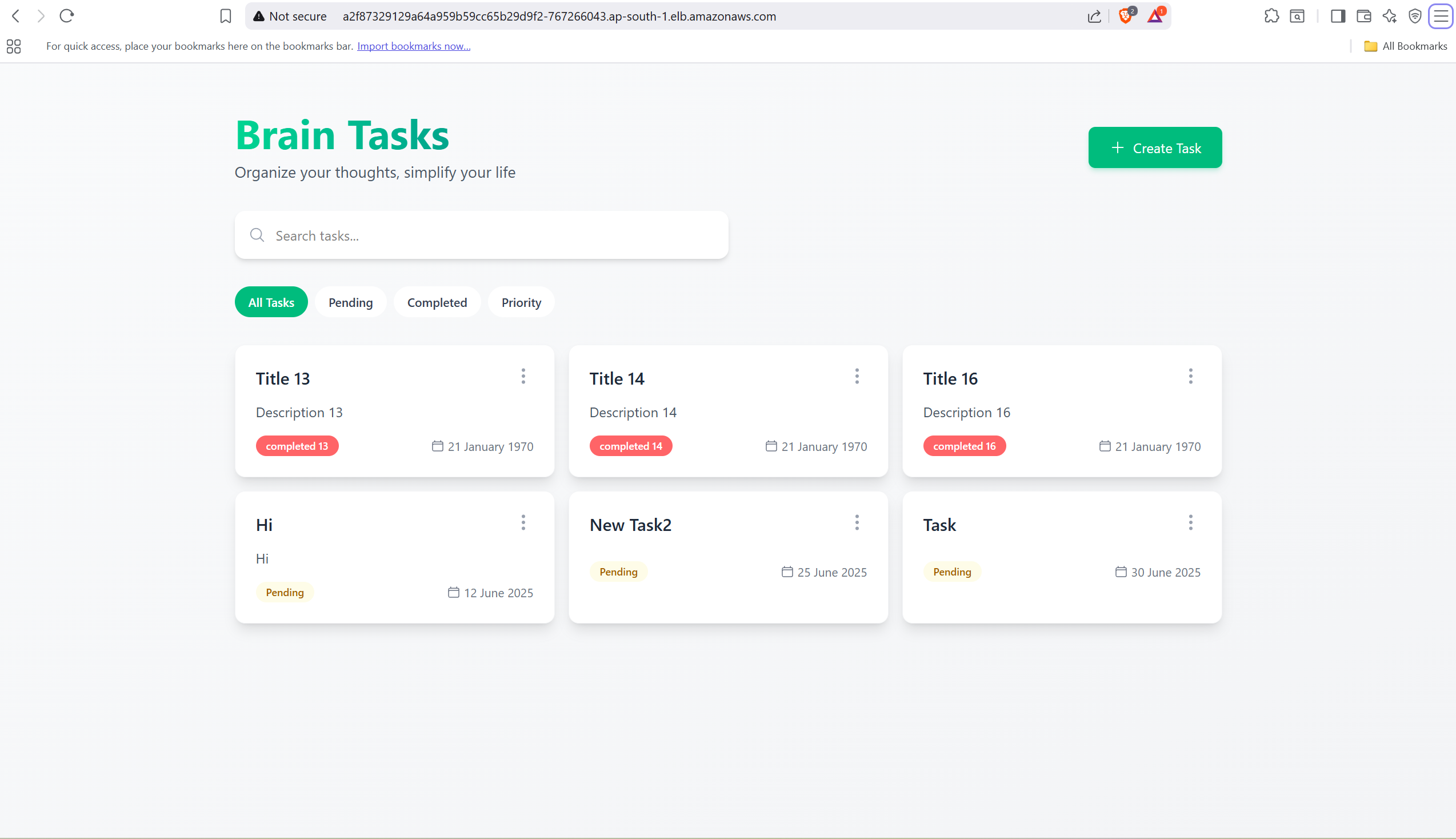
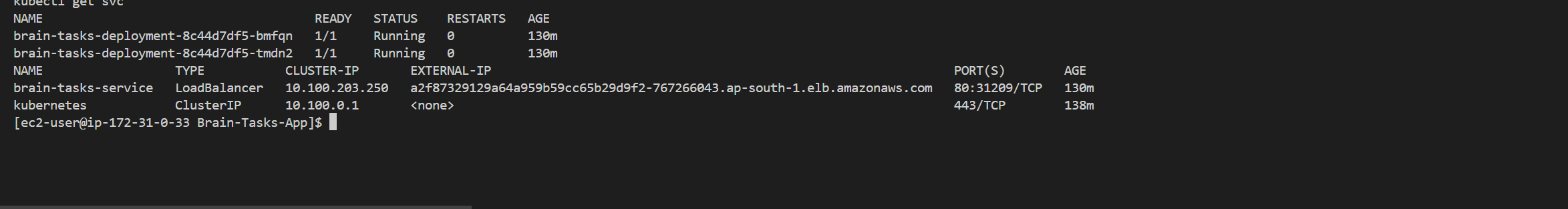
Visit the LoadBalancer URL from your `service.yaml` output to verify the deployed app is accessible.

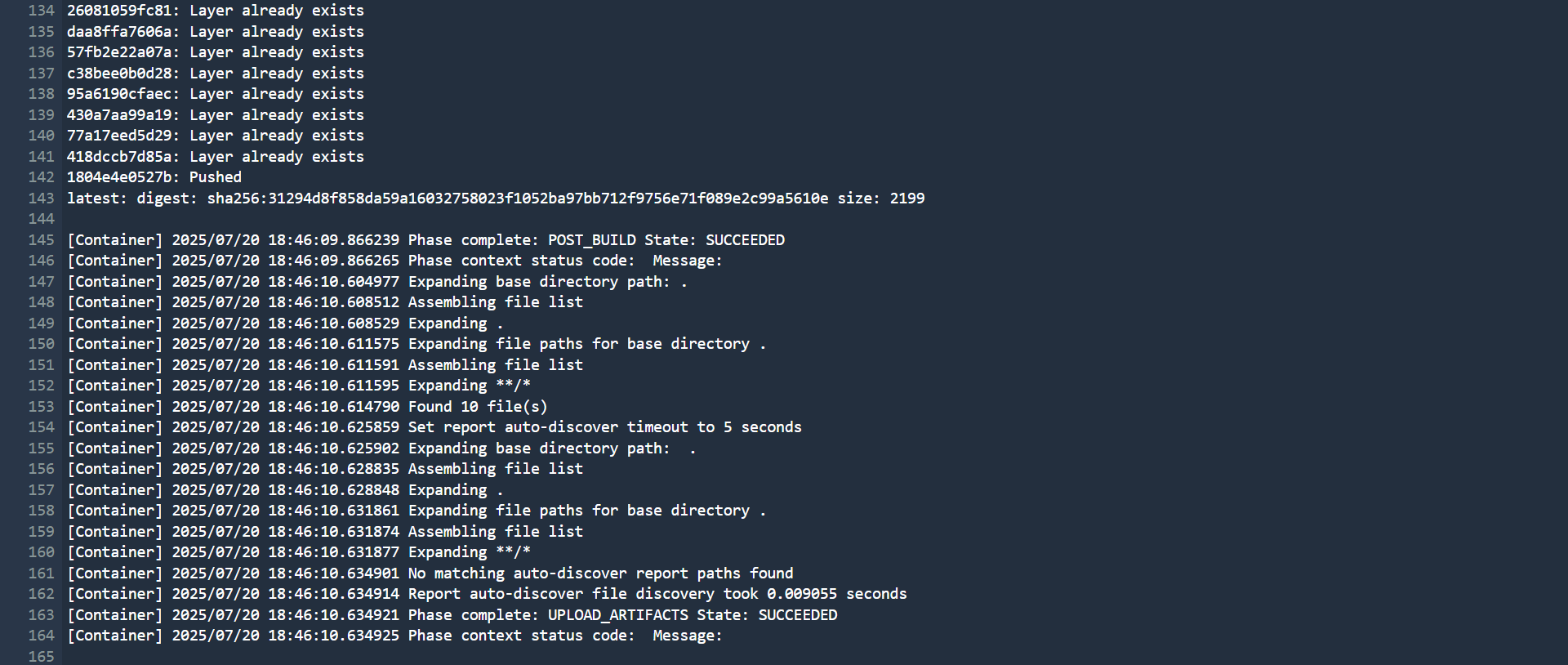
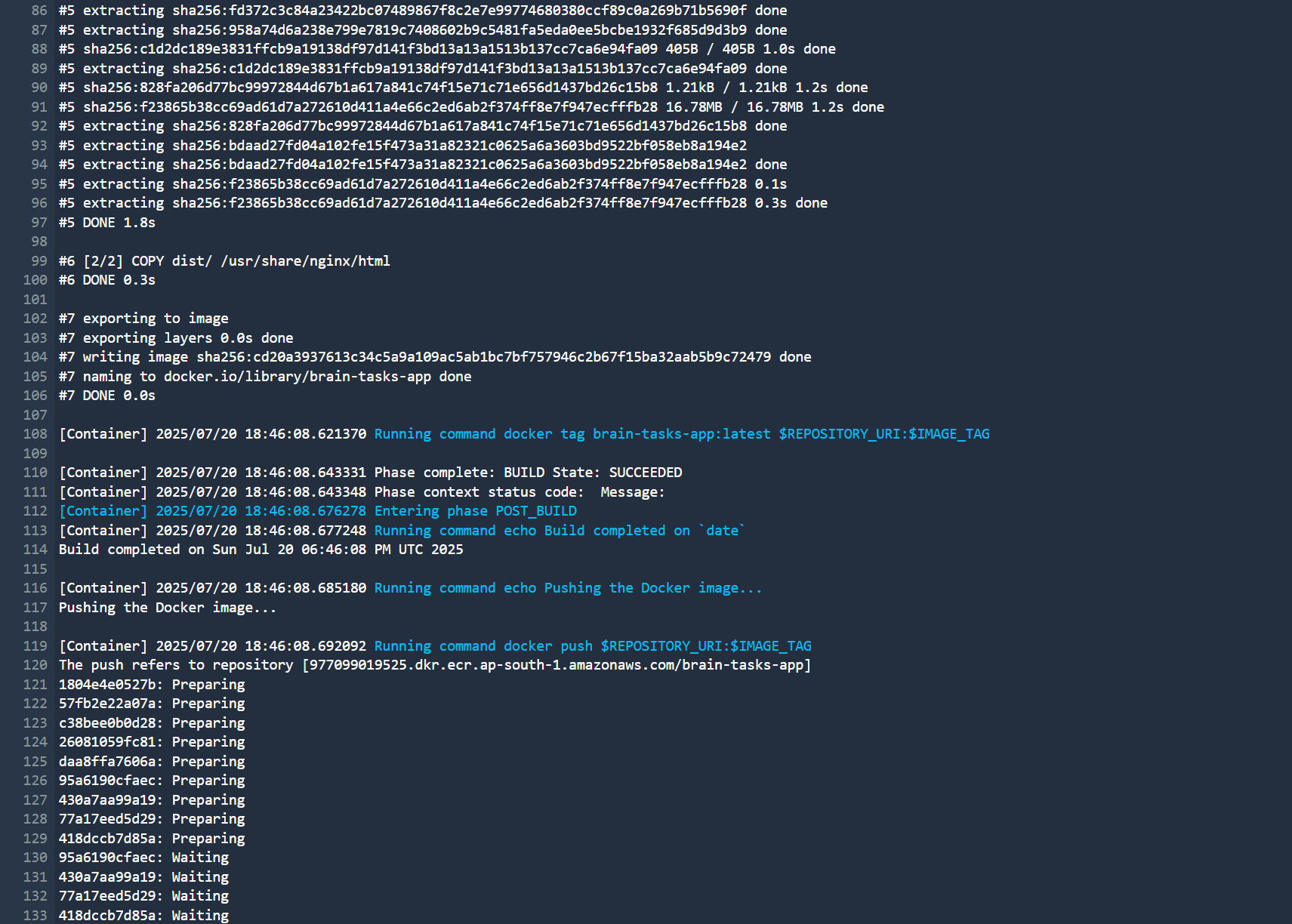
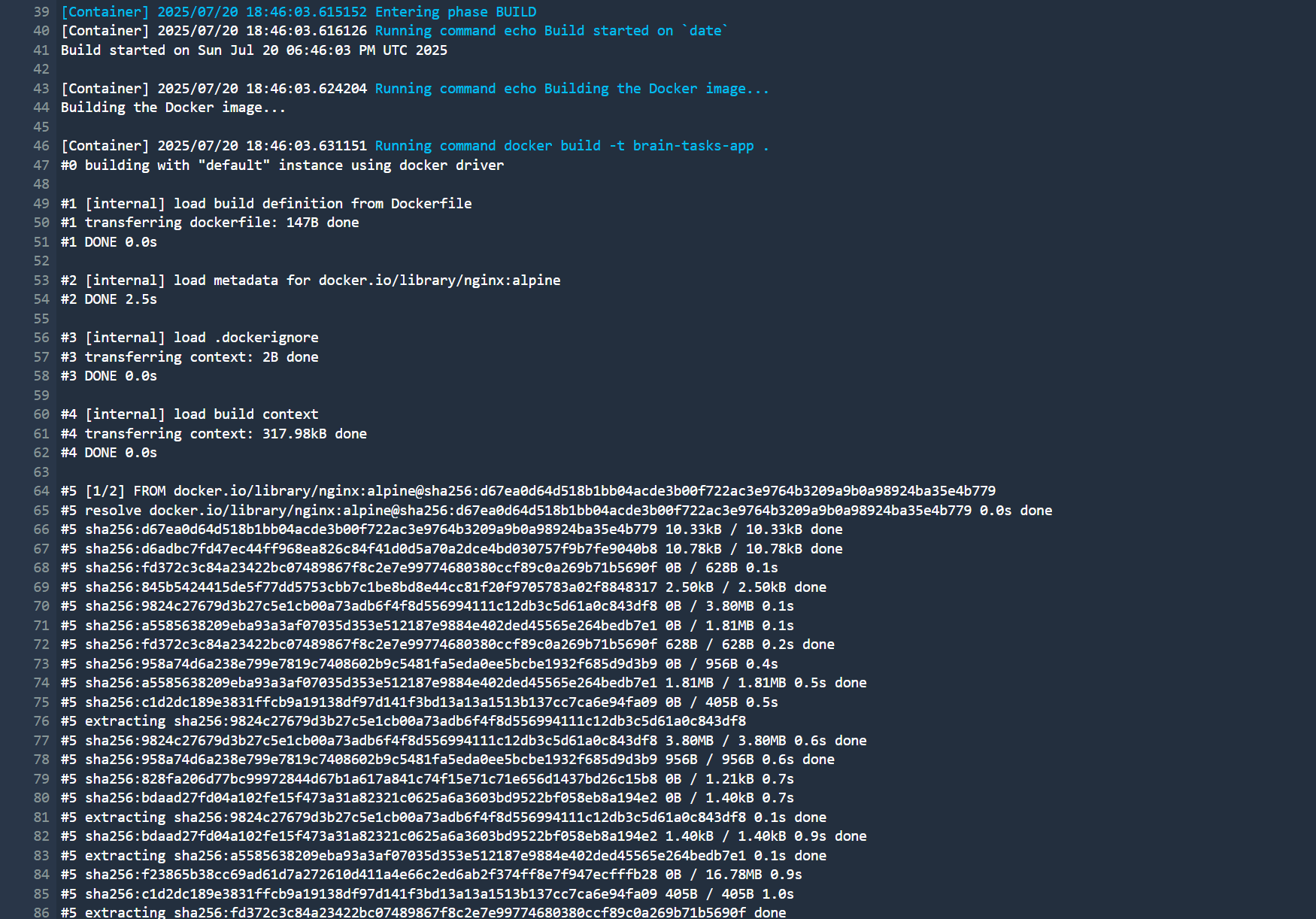
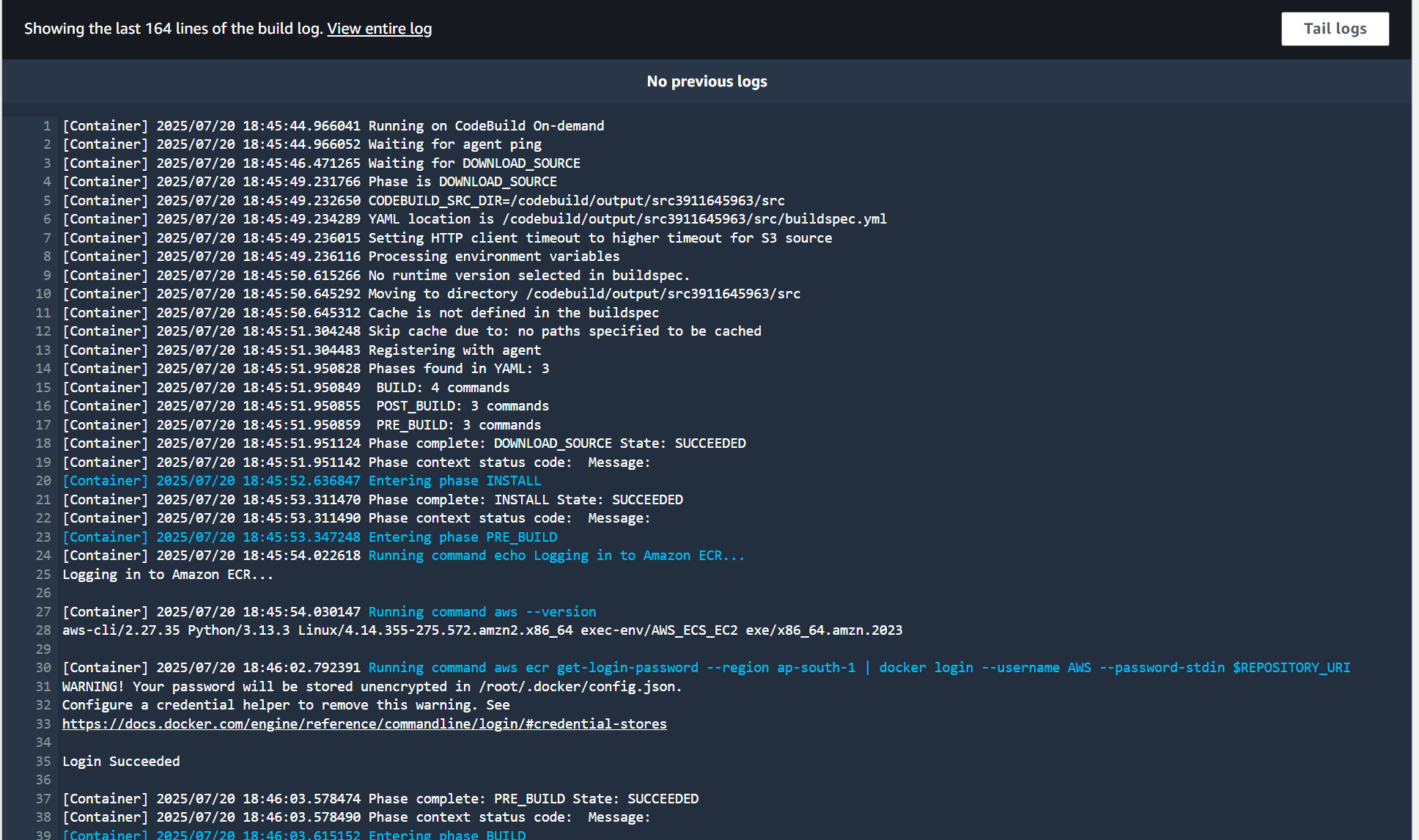
# 10. Screenshots to Include in Report

- Screenshot of ECR repo with pushed image.

- Screenshot of CloudWatch logs.

- Screenshot of CodePipeline execution (all stages green).

- Screenshot of deployed app in browser (LoadBalancer URL).  
- Screenshot of kubectl get pods and services output.

- Screenshot of successful CodeBuild logs.

# 11. Load Balancer

- \*\*DNS Name\*\*: http://a2f87329129a64a959b59cc65b29d9f2-767266043.ap-south-1.elb.amazonaws.com

- \*\*Type\*\*: Classic Load Balancer (created by Kubernetes LoadBalancer service)

- \*\*ARN\*\*: Not applicable (Classic Load Balancer does not expose ARN via ELB API)

> LoadBalancer was auto-created by Kubernetes service of type `LoadBalancer`.