

AWS DevOps Course-end Project 1

Automating CI/CD Pipeline for Spring Boot Application Deployment on AWS

Objectives

To implement a CI/CD pipeline using AWS services for automating the deployment of a Spring Boot application on Amazon ECS with Docker, integrating CodePipeline, CodeBuild and ECR for seamless updates.

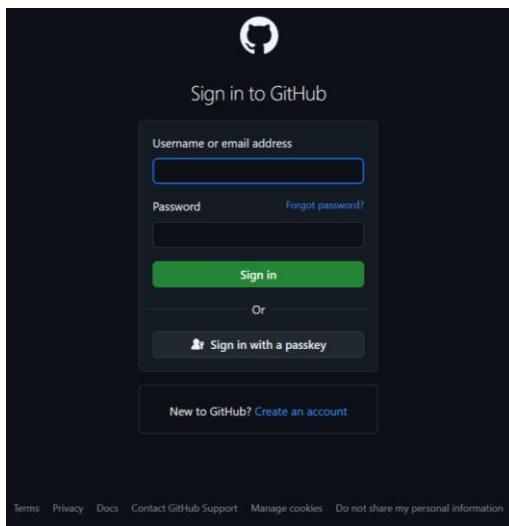
Tasks The following tasks outline the CI/CD pipeline creation process:

1. Fork the Spring Boot project GitHub repository and create an ECR to store the Docker images
2. Modify the build configuration file in the GitHub repository to include the ECR image URI
3. Create a CodeBuild project to pull code from the GitHub repository
4. Create an ECS cluster using the Fargate launch type and configure an ECS service to run tasks on the cluster
5. Deploy the application
6. Configure CodePipeline to trigger on GitHub changes, integrating CodeBuild and ECS for automated deployment
7. Test and monitor the pipeline execution in AWS CodePipeline

Step 1: GitHub Repository and ECR

1.1 Connect to GitHub Repository

To start the project, log-in into your GitHub account.



To host a Spring Boot application's source code in a GitHub repository, you will need to either create your own GitHub repository with a connection to Docker. Or use an established GitHub repository with Docker and Spring Boot Application.

The repository will be the source for the CI/CD pipeline. This means when a developer commits and pushes changes to the GitHub repository, it triggers the pipeline in AWS. We will also be using Docker through AWS.

For this project, we will use an established GitHub repository: <https://github.com/akshu20791/aws-devops-springbootapp>

A screenshot of a GitHub repository page. The top navigation bar shows the repository path "akshu20791 / aws-devops-springbootapp" and includes links for Code, Issues, Pull requests, Actions, Projects, Security, and Insights. There is also a search bar and various repository management icons. The main content area shows the repository details: "aws-devops-springbootapp" (Public), "Code" tab selected, "master" branch, 1 branch, 0 tags. The commit history lists 9 commits from "akshu20791" made 3 weeks ago, all labeled "my first commit" and dated 4 months ago. Below the commits is a "README" file. On the right side, there are sections for "About" (no description, 95 forks), "Releases" (no releases), "Packages" (no packages), "Contributors" (2 contributors: akshu20791 and aarrffrftfr), and "Languages" (no languages listed).

Commit	Author	Message	Date
Update buildspec.yml	akshu20791	my first commit	4 months ago
.mvn/wrapper		my first commit	4 months ago
src		my first commit	4 months ago
.gitignore		my first commit	4 months ago
Dockerfile		my first commit	4 months ago
README.md		Update README.md	4 months ago
buildspec.yml		Update buildspec.yml	3 weeks ago
mvnw		my first commit	4 months ago
mvnw.cmd		my first commit	4 months ago
pom.xml		my first commit	4 months ago

First establish a connection of the Spring Boot App by “forking” it to your account.

The screenshot shows a GitHub repository page for 'akshu20791/aws-devops-springbootapp'. At the top, there are buttons for 'Watch' (1), 'Fork' (95), and 'Star' (1). A tooltip above the 'Fork' button says 'Fork your own copy of akshu20791/aws-devops-springbootapp'. Below the header, there's a search bar ('Go to file'), an 'Add file' button, and a 'Code' dropdown menu. To the right of the code dropdown is an 'About' section with the message 'No description, website, or topics provided.' Below this are links for 'Readme', 'Activity', '1 star', '1 watching', and '95 forks'. On the left, there's a list of commits:

Commit	Date
ommit	4 months ago

Click on “Fork”.

Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks](#).

Required fields are marked with an asterisk (*)

Owner * Repository name *

Choose an owner / aws-devops-springbootap

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

Copy the `master` branch only

Contribute back to [akshu20791/aws-devops-springbootapp](#) by adding your own branch. [Learn more](#).

Create fork

Select your GitHub account under Owner. Then select “Create fork.”

Popular repositories

- test** Public
- apachewebsite** Public

Forked from [akshu20791/apachewebsite](#)

● CSS
- aws-devops-springbootapp** Public

Forked from [akshu20791/aws-devops-springbootapp](#)

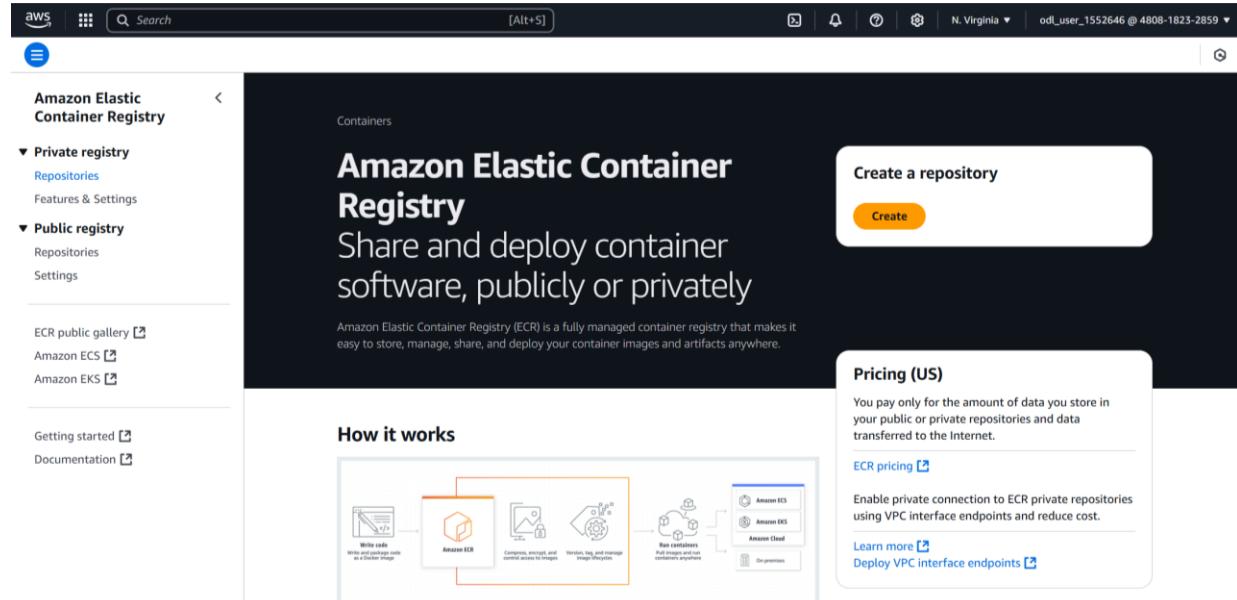
● Java

The GitHub repository “Spring Boot App” will appear as one of your repositories in your GitHub account. You can now begin the project in AWS.

1.2 AWS ECR

Log-in to your AWS account. On the Homepage, search for Elastic Container Repository (ECR).

To create a private repository from Github, go to ECR. Click on Private Registry > Repository.



The screenshot shows the AWS ECR homepage. On the left, there's a sidebar with navigation links for 'Amazon Elastic Container Registry', 'Private registry', 'Public registry', 'ECR public gallery', 'Amazon ECS', 'Amazon EKS', 'Getting started', and 'Documentation'. The main content area has a dark background with white text. It features the heading 'Amazon Elastic Container Registry' and the subtext 'Share and deploy container software, publicly or privately'. Below this is a paragraph about ECR being a managed container registry. To the right, there's a large 'Create a repository' button with an orange 'Create' button underneath it. At the bottom, there's a 'How it works' diagram showing a flow from writing code to running containers, and sections for 'Pricing (US)' and 'ECR pricing'.

In private repository, click on Create Repository.

The screenshot shows the AWS ECR interface. On the left, there's a sidebar with navigation links for Amazon Elastic Container Registry, Private registry (Repositories, Features & Settings), Public registry (Repositories, Settings), ECR public gallery, Amazon ECS, Amazon EKS, Getting started, and Documentation. The main content area is titled "Private repositories" and shows a search bar with placeholder text "Search by repository substring". Below the search bar are filter buttons for "Repository name", "URI", "Created at", "Tag immutability", and "Encryption type". A message "No repositories" and "No repositories were found" is displayed. At the top right, there are buttons for "View push commands", "Delete", "Actions", and "Create repository".

In Create Repository, provide a name for your ECR repository.

The screenshot shows the "Create private repository" wizard. The first step, "General settings", requires a repository name (480818232859.dkr.ecr.us-east-1.amazonaws.com/awsdevopsecr) and sets the tag mutability to "Mutable". The second step, "Encryption settings", shows AES-256 selected as the encryption configuration. The third step, "Image scanning settings - deprecated", is collapsed. At the bottom, there are "Cancel" and "Create" buttons.

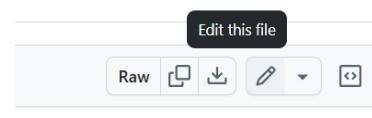
In this case, we will call our private repository “awsdevopsecr”. Leave all default settings and click on Create.

The screenshot shows the AWS ECR console. On the left, there's a sidebar with navigation links for Amazon Elastic Container Registry, Private registry (Repositories, Features & Settings), and Public registry (Repositories, Settings). Below these are links for ECR public gallery, Amazon ECS, and Amazon EKS. At the bottom of the sidebar are Getting started and Documentation links. The main content area has a green success message: "Successfully created awsdevopsecr". Below this is a table titled "Private repositories (1)". The table has columns for Repository name, URI, Created at, Tag immutability, and Encryption type. One row is shown: awsdevopsecr, 480818232859.dkr.ecr.us-east-1.amazonaws.com/awsdevopsecr, December 22, 2024, 15:49:12 (UTC-08), Mutable, AES-256. There are buttons for View push commands, Delete, Actions, and Create repository.

Now to establish first connection with GitHub to your ECR repository.

In the Spring Boot App repository you forked, click the file “buildspec.yml”.

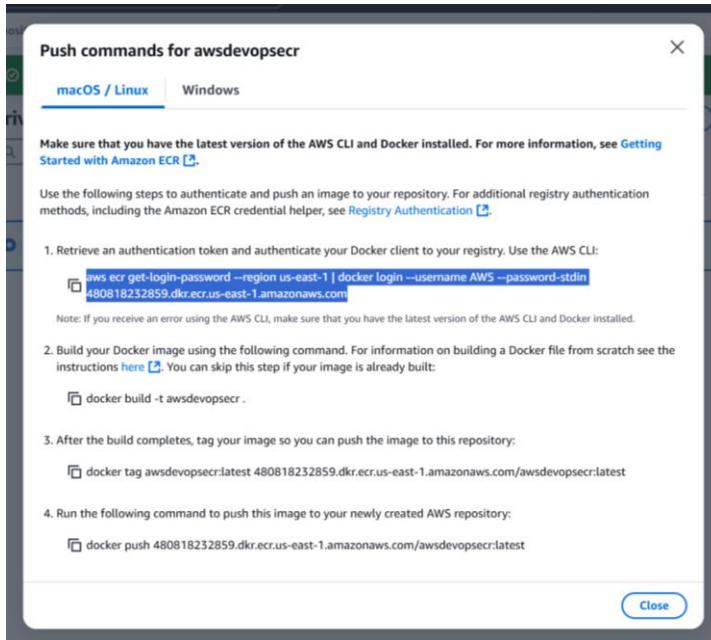
.mvn/wrapper	my first commit	4 months ago
src	my first commit	4 months ago
.gitignore	my first commit	4 months ago
Dockerfile	my first commit	4 months ago
README.md	Update README.md	4 months ago
buildspec.yml	Update buildspec.yml	3 weeks ago
mvnw	my first commit	4 months ago
mvnw.cmd	my first commit	4 months ago
pom.xml	my first commit	4 months ago



In the file, click on the pencil icon to edit file.

In ECR, select your repository and click on “View Push Commands.”

This screenshot shows the AWS ECR console with the same sidebar and repository list as the previous one. The main content area shows the "Private repositories (1)" table with the same data as before: awsdevopsecr, 480818232859.dkr.ecr.us-east-1.amazonaws.com/awsdevopsecr, December 22, 2024, 15:49:12 (UTC-08), Mutable, AES-256. The "View push commands" button is highlighted.



From the first push command. Copy the authentication token for macOS/Linux.

We will not be using Windows.

In editing the `buildspec.yml` file, go to line 10.

```

version: 0.2
phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws --version
      - $(aws ecr get-login --region ap-south-1 --no-include-email)
      - update below
      - aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 042326304859.dkr.ecr.us-east-1.amazonaws.com
      - # Replace with this to your repository URI
      - REPOSITORY_URI=042326304859.dkr.ecr.us-east-1.amazonaws.com/awsdevopsecr
      - IMAGE_TAG=$(echo $CODEBUILD_BUILD_ID | awk -F'-' '{print $2}')
  build:
    commands:
      - echo Build started on `date`
      - echo building the Jar file
      - mvn clean install
      - echo Building the Docker image...
      - docker build -t $REPOSITORY_URI:latest .
      - docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo pushing to repo
      - docker push $REPOSITORY_URI:latest
      - docker push $REPOSITORY_URI:$IMAGE_TAG
      - echo Writing image definitions file...
      - # Give your container name

```

Edit it to match with the push command in ECR.

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin  
480818232859.dkr.ecr.us-east-1.amazonaws.com
```

Next copy the url of your ECR repository.

Amazon ECR > Private registry > Repositories

Amazon Elastic Container Registry

Private registry

Repositories Features & Settings

Public registry

Repositories Settings

ECR public gallery

Private repositories (1)

Search by repository substring

Repository name	URI	Created at	Tag immutability	Encryption type
awsdevopsec	480818232859.dkr.ecr.us-east-1.amazonaws.com/awsdevopsec	December 22, 2024, 15:49:12 (UTC-08)	Mutable	AES-256

Copied URI to clipboard

In repository URL in line 12, copy the url file and edit it in the yml file.

```
version: 0.2
phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      #Checking version
      - aws --version
      # $(aws ecr get-login --region ap-south-1 --no-include-email)
      - update below
      - aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 480818232859.dkr.ecr.us-east-1.amazonaws.com
      # Replace with this to your repository URL
      - REPOSITORY_URI=480818232859.dkr.ecr.us-east-1.amazonaws.com/awsdevopsec
      - IMAGE_TAG=$(echo $CODEBUILD_BUILD_ID | awk -F '-' '{print $2}')
build:
  commands:
    - echo Build started on `date`
```

Now to save your edit, click on Commit Changes in GitHub.

Code Pull requests Actions Projects Wiki Security Insights Settings

aws-devops-springbootapp / buildspec.yml in master

Commit changes

Commit message

Update buildspec.yml

Extended description

Add an optional extended description...

Commit directly to the master branch

Create a new branch for this commit and start a pull request [Learn more about pull requests](#)

Commit changes

As noted, the Spring Boot App comes with a Dockerfile to use.

The screenshot shows a GitHub repository page for 'aws-devops-springbootapp'. At the top, it displays 'master' branch, '1 Branch', and '0 Tags'. Below this, a message says 'This branch is 2 commits ahead of, 1 commit behind akshu20791/aws-devops-springbootapp:master.' There are 'Contribute' and 'Sync fork' buttons. The main area shows a list of files under the 'master' branch, all committed by 'JJ-Soliz' at 'ee4e4eb' 17 minutes ago, with 10 commits. The files listed are: .mvn/wrapper, src, .gitignore, Dockerfile, README.md, buildspec.yml, mvnw, mvnw.cmd, and pom.xml, all showing 'my first commit' and being 4 months ago. On the right, there's a detailed view of the 'Dockerfile' code, which contains the following Dockerfile content:

```

FROM eclipse-temurin:17-jdk-alpine
RUN apk add curl
VOLUME /tmp
EXPOSE 8080
ADD target/springboot-aws-deploy-service.jar springboot-aws-deploy-service.jar
ENTRYPOINT ["java","-jar","/springboot-aws-deploy-service.jar"]

```

With our ECR repository connected to our GitHub repository, we can store the Docker images

2. Create CodeBuild

2.1 AWS CodeBuild

In AWS, search for CodeBuild and click on it.

The screenshot shows the AWS CodeBuild console. The left sidebar has a 'Developer Tools' menu with 'CodeBuild' selected. Under 'Build projects', there are sections for 'Source', 'Artifacts', and 'Build'. 'Build projects' is expanded, showing 'Getting started', 'Build projects' (which is currently selected), 'Build history', 'Report groups', 'Report history', 'Compute fleets New', and 'Account metrics'. The main content area shows the 'Build projects' list with the following header: 'Build projects' (Info), 'Actions', 'Create trigger', 'View details', 'Start build', and 'Create project'. A search bar and a 'Your projects' dropdown are also present. The list table has columns: Name, Source provider, Repository, Latest build status, Description, and Last Modified. The message 'No results' is displayed below the table, followed by 'There are no results to display.'

Click on Create project.

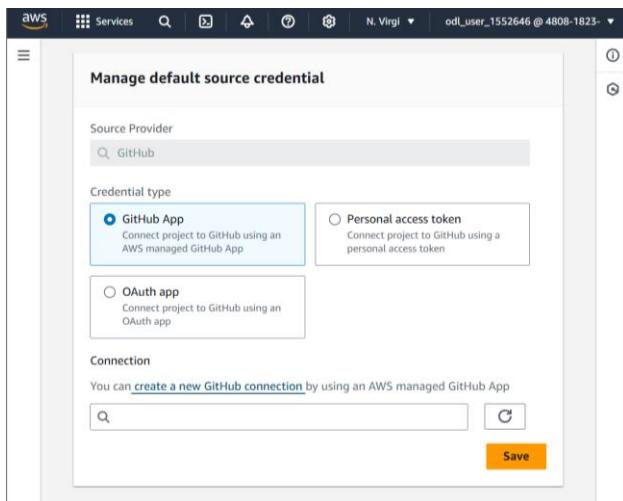
The screenshot shows the 'Create build project' page in AWS CodeBuild. In the 'Project configuration' section, the 'Project name' field contains 'awsdevopscodebuild1'. Below it, under 'Additional configuration', there is a note about project name rules. In the 'Source' section, 'Source 1 - Primary' is selected, and 'GitHub' is chosen as the source provider. Under 'Credential', 'Default source credential' is selected.

In Create build project, write in a project name for your build. In this case, we will call it “awsdevopscodebuild1”.

Under Source, select “GitHub” as your source provider.

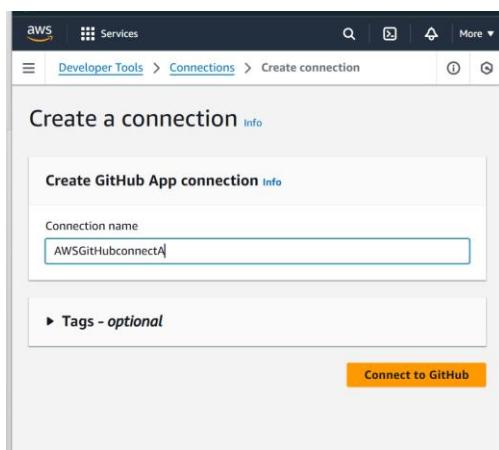
The screenshot shows the 'Source' configuration page. 'GitHub' is selected as the 'Source provider'. Under 'Credential', 'Default source credential' is selected. A note at the bottom left states 'You have not connected to GitHub' and provides a link to 'Manage default source credential'.

Note: your GitHub is not connected. Click on “Manage default source credential” and a new tab will appear.



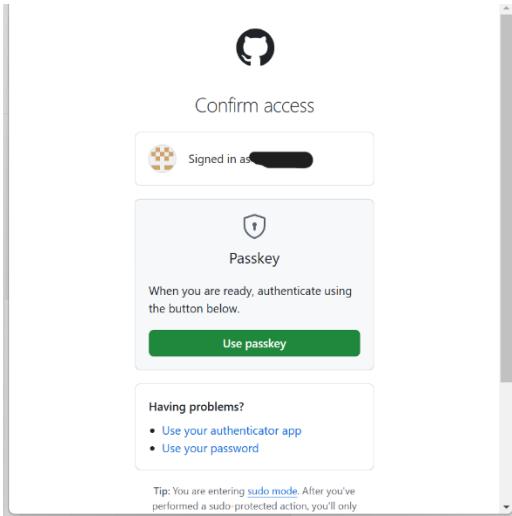
Select “GitHub App” and then click on “create a new GitHub connection”. Be sure you are logged into your GitHub account beforehand.

A pop up will appear.



For connection name, we will use “AWSGitHubconnectA”. Click on Connect to GitHub.

Next click on “Install a new App.” You will be brought to a page to select your GitHub account through AWS Connector.



You have the option of Passkey, Authenticator app, and password. Select one and your Github account will be connected to AWS.

In the window, your Github Account settings will appear.

Under Permissions in Repository access, select repositories and select your Spring Boot App repository that you forked.

Click on Save.

Developer Tools > Connections > Create connection

Connect to GitHub

GitHub connection settings [Info](#)

Connection name: AWSGitHubconnectA

App installation - optional
Install GitHub App to connect as a bot. Alternatively, leave it blank to connect as a GitHub user, which can be used in AWS CodeBuild projects.
Q 57237131 X

▶ Tags - optional

Connect

Your connect is established, click on Connect again.

Manage default source credential

Source Provider: GitHub

Credential type:

- GitHub App
- Personal access token
- OAuth app

Connection:

You can [create a new GitHub connection](#) by using an AWS managed GitHub App

Q AWSdevopsconnectA C AWSGitHubconnectA Save

Click on your connection you created and click on Save.

Credential

- Default source credential
- Custom source credential

Successfully connected by using an AWS managed GitHub App - [open resource](#)

Manage default source credential

Repository

- Repository in my GitHub account
- Public repository
- GitHub scoped webhook

Repository: Q https://github.com/JJ-Soliz/aws-devops-springbootapp X

C

Your GitHub account is fully-connected. Select your GitHub repository and Scroll down.

In Environment, be sure to select New Service Role and copy the Role name of the service role.

codebuild-awsdevopscodebuild1-service-role

The screenshot shows the 'New Service Role' configuration section. It includes fields for Runtime(s) (Standard), Image (aws/codebuild/amazonlinux-x86_64-standard:5.0), Image version (Always use the latest image for this runtime version), and GPU-enhanced compute (unchecked). Under Service role, the 'New service role' option is selected, with a sub-note: 'Create a service role in your account'. The Role name field contains 'codebuild-awsdevopscodebuild1-service-role'. Below this, there's a 'Additional configuration' section with timeout, privileged, certificate, VPC, compute type, environment variables, file systems, auto-retry, and registry credential options. A 'Buildspec' section is also visible at the bottom.

You will need the name when creating IAM roles.

The screenshot shows the 'Buildspec' configuration section. It includes a 'Build specifications' section where 'Use a buildspec file' is selected, with a note: 'Store build commands in a YAML-formatted buildspec file'. Below this is a 'Buildspec name - optional' field containing 'buildspec.yml'.

Logs

CloudWatch

CloudWatch logs - optional
Checking this option will upload build output logs to CloudWatch.

Group name - optional

The group name of the logs in CloudWatch Logs. The log group name will be /aws/codebuild/<project-name> by default.

Stream name prefix - optional

The prefix of the stream name of the CloudWatch Logs.

S3

S3 logs - optional
Checking this option will upload build output logs to S3.

[Cancel](#) [Create build project](#)

Continue to Scroll Down until you reach the end and click on Create build project.

If you get an error, click on Edit and remove “Cloudwatch logs”. That should fix the issue.

The screenshot shows the AWS CodeBuild console with a success message: "Project details for build project awsdevopscodebuild1 successfully updated." The navigation path is Developer Tools > CodeBuild > Build projects > awsdevopscodebuild1. The main view displays the configuration for the build project "awsdevopscodebuild1". Key settings include:

- Actions:** Create trigger, Edit, Clone, Debug build, Start build with overrides, Start build (highlighted).
- Configuration:**
 - Source provider: GitHub
 - Primary repository: JJ-Soliz/aws-devops-springbootapp
 - Artifacts upload location: -
 - Service role: arn:aws:iam::480818232859:role/service-role/codebuild-awsdevopscodebuild1-service-role
 - Public builds: Disabled
- Project details:** Build history, Batch history, Project details (selected), Build triggers, Metrics.
- Project configuration:**

Name	Description
awsdevopscodebuild1	-
Project ARN	arn:aws:codebuild:us-east-1:480818232859:project/awsdevopscodebuild1
Build badge	Disabled

Your CodeBuild is created.

2.2 IAM Permission Roles

Go to IAM in AWS to establish permissions roles for your Code Build.

The screenshot shows the AWS IAM Dashboard. On the left, there's a sidebar with 'Identity and Access Management (IAM)' and 'Access management' sections. The main area has a 'Security recommendations' section with three items: 'Add MFA for root user', 'Add MFA for yourself', and a note about unused access keys. To the right is the 'AWS Account' section with account ID, alias, and sign-in URL. Below that is a 'Quick Links' section with 'My security credentials'.

Select Roles in the left side of the menu.

The screenshot shows the 'Roles' page under 'Access management'. It lists 20 roles, each with a role name, trusted entities, and last activity. The roles listed include various AWS service roles like 'AWSServiceRoleForAmazonGuardDuty' and 'AWSServiceRoleForAmazonInspector'.

Select the role you created in the Code build.

The screenshot shows the 'Roles' page with a search filter set to 'dev'. It lists two roles: 'codebuild-awsdevopscodebuild1-service-role' and another role. Below the table is a 'Roles Anywhere' section with options for 'Access AWS from your non AWS workloads', 'X.509 Standard', and 'Temporary credentials'.

Select and open.

Identity and Access Management (IAM)

codebuild-awsdevopscodebuild1-service-role

Summary

Creation date: December 22, 2024, 17:34 (UTC-08:00)

Last activity: -

ARN: arn:aws:iam::480818232859:role/service-role/codebuild-awsdevopscodebuild1-service-role

Maximum session duration: 1 hour

Permissions | Trust relationships | Tags | Last Accessed | Revoke sessions

Permissions policies (1) Info

You can attach up to 10 managed policies.

Filter by Type: All types

Attached entities: 1

Policy name: CodeBuildBasePolicy-awsdevopscodebuild1...

Type: Customer managed

Add permissions ▾

Attach policies

Create inline policy

In Add permissions > Select Attach policies.

Search [Alt+S]

IAM > Roles > codebuild-awsdevopscodebuild1-service-role > Add permissions

Attach policy to codebuild-awsdevopscodebuild1-service-role

▶ Current permissions policies (1)

Other permissions policies (1/1022)

Policy name	Type	Description
<input checked="" type="checkbox"/> AdministratorAccess	AWS managed - job function	Provides full access to AWS services an...
<input type="checkbox"/> AdministratorAccess-Amplify	AWS managed	Grants account administrative permis...
<input type="checkbox"/> AdministratorAccess-AWSElasticBeanstalk	AWS managed	Grants account administrative permis...

Other permissions policies (3/1022)

Policy name	Type	Description
<input checked="" type="checkbox"/> AmazonEC2ContainerRegistryFullAccess	AWS managed	Provides administrative access to Ama...
<input checked="" type="checkbox"/> AmazonEC2ContainerRegistryPowerUser	AWS managed	Provides full access to Amazon EC2 Co...
<input type="checkbox"/> AmazonEC2ContainerRegistryPullOnly	AWS managed	Provides access to pull images from A...
<input type="checkbox"/> AmazonEC2ContainerRegistryReadOnly	AWS managed	Provides read-only access to Amazon E...

Select AdministratorAccess, AmazonEC2ContainerRegistryFullAccess, and AmazonEC2ContainerRegistryPowerUser.

Select all three and then select Add Permissions.

The screenshot shows the AWS IAM Roles page. The top navigation bar includes 'Search' and 'Global'. The main title is 'codebuild-awsdevopscodebuild1-service-role'. A green banner at the top states 'Policies have been successfully attached to role.' Below this, it shows the creation date 'December 22, 2024, 17:34 (UTC-08:00)' and the ARN 'arn:aws:iam::480818232859:role/service-role/codebuild-awsdevopscodebuild1-service-role'. The 'Permissions' tab is selected, showing four managed policies: 'AdministratorAccess', 'AmazonEC2ContainerRegistryFullAccess', 'AmazonEC2ContainerRegistryPowerUser', and 'CodeBuildBasePolicy-awsdevopscodebuild...'. Other tabs include 'Trust relationships', 'Tags', 'Last Accessed', and 'Revoke sessions'.

2.3 Create Build

Go back to your Code Build.

The screenshot shows the AWS CodeBuild console. The left sidebar has a 'Developer Tools' section with 'CodeBuild' selected, and a 'Build projects' section with 'awsdevopscodebuild1' highlighted. The main area shows the 'awsdevopscodebuild1' project details. It includes tabs for 'Actions' (Create trigger, Edit, Clone, Debug build, Start build with overrides, Start build), 'Configuration' (Source provider: GitHub, Primary repository: JJ-Soliz/aws-devops-springbootapp, Artifacts upload location: -, Service role: arn:aws:iam::480818232859:role/service-role/codebuild-awsdevopscodebuild1-service-role), and 'Build history' (No results). The bottom navigation bar includes 'CloudShell', 'Feedback', 'Search', and various application icons. The status bar at the bottom right shows '5:56 PM 12/22/2024'.

Select Start Build.

This will take a couple minutes.

The screenshot shows the AWS CodeBuild console. On the left, a sidebar navigation includes: Developer Tools, CodeBuild, Source (CodeCommit), Artifacts (CodeArtifact), Build (CodeBuild), Getting started, Build projects, Build project (selected), Settings, Build history, Report groups, Report history, Compute fleets (New), Account metrics, Related integrations (Jenkins, GitHub Actions, GitLab runners), Deploy (CodeDeploy), Pipeline (CodePipeline), and Settings. The main content area displays a green banner at the top stating "Build started" and "You have successfully started the following build: awsdevopscodebuild1:d12fe526-b05d-43f6-9983-01883476f083". Below this, the build ID is shown as "awsdevopscodebuild1:d12fe526-b05d-43f6-9983-01883476f083". There are "Stop build" and "Retry build" buttons. The "Build status" section shows the status as "Succeeded", initiator as "odl_user_1552646", and build ARN as "arn:aws:codebuild:us-east-1:480818232859:build/awsdevopscodebuild1:d12fe526-b05d-43f6-9983-01883476f083". It also shows start time (Dec 22, 2024 5:57 PM UTC-8:00), end time (Dec 22, 2024 5:58 PM UTC-8:00), build number (1), and reports (awsdevopscodebuild1-JUNITXML-AutoDiscovered). Below this are tabs for Build logs, Phase details, Reports, Environment variables, Build details, and Resource utilization. The "Logs" section indicates CloudWatch logs are disabled, CloudWatch group name is "/aws/codebuild/awsdevopscodebuild1", and CloudWatch stream name is "-".

Build is successful.

3. Elastic Container Service Cluster

3.1 Create ECS Cluster

Go to Elastic Container Service (ECS). Go to Clusters.

The screenshot shows the AWS Elastic Container Service (ECS) Clusters page. The left sidebar includes: Amazon Elastic Container Service, Clusters (Updated), Namespaces, Task definitions, Account settings (Updated), and Install AWS Copilot. The main content area shows a table titled "Clusters (0)" with columns: Cluster, Services, Tasks, Container instances, CloudWatch monitoring, and Capacity p. A message at the bottom states "No clusters" and "No clusters to display". At the top right, there is a "Create cluster" button. The top bar shows the AWS logo, search bar, and user information (odl_user_1552646 @ 4808-1823-2859).

Select Create Cluster.

The screenshot shows the 'Create cluster' wizard in the AWS Elastic Container Service. The current step is 'Cluster configuration'. The 'Cluster name' field contains 'awsdevopsclusterA'. The 'Default namespace - optional' field contains 'awsdevopsclusterA'. Below these fields, there is a section titled 'Infrastructure' with a 'Serverless' tab selected. It includes options for AWS Fargate (selected), Amazon EC2 instances (unchecked), and external instances using ECS Anywhere (unchecked). A note states: 'Your cluster is automatically configured for AWS Fargate (serverless) with two capacity providers. Add Amazon EC2 instances.'

Create a name for your cluster “awsdevopsclusterA”.

The screenshot shows the 'Create cluster' wizard in the AWS Elastic Container Service. The current step is 'Infrastructure'. The 'AWS Fargate (serverless)' checkbox is checked. Other options like 'Amazon EC2 instances' and 'External instances using ECS Anywhere' are unchecked. Below this, there are sections for 'Monitoring - optional', 'Encryption - optional', and 'Tags - optional'. At the bottom right are 'Cancel' and 'Create' buttons.

Be sure to select AWS Fargate in Infrastructure. Then select Create.

The screenshot shows the 'Clusters' page in the AWS Elastic Container Service. A green banner at the top says 'Cluster awsdevopsclusterA has been created successfully.' The main table shows one cluster named 'awsdevopsclusterA'. The table has columns for Cluster, Services, Tasks, Container instances, CloudWatch monitoring, and Capacity provider strategy. The 'Container instances' row shows 0 EC2. The 'Capacity provider strategy' row shows 'Default' and 'No default found'. At the top right are 'View cluster' and 'Create cluster' buttons. At the bottom right are navigation buttons for the table.

Cluster made.

NOTE: You might get an error in creating your cluster. This could be due to the name or need to be redone again. Keep trying until it is successful.

3.2 Create Tasks

Go to Task definitions.

The screenshot shows the AWS Management Console interface for the Amazon Elastic Container Service (Amazon ECR). The left sidebar navigation includes 'Clusters' (Updated), 'Namespaces', 'Task definitions' (selected), 'Account settings' (Updated), 'Install AWS Copilot', 'Amazon ECR', 'Repositories', 'AWS Batch', 'Documentation', 'Discover products', and 'Subscriptions'. The main content area is titled 'Task definitions (0)' and shows a message 'No task definitions to display.' with a 'Create new task definition' button.

Click on Create new task definition.

The screenshot shows the 'Create new task definition' configuration page. The left sidebar navigation is identical to the previous screenshot. The main content area is titled 'Create new task definition' and contains several configuration sections: 'Task definition configuration' (with a 'Task definition family' input field containing 'AWSDevOpsTaskA'), 'Infrastructure requirements' (with 'Launch type' set to 'AWS Fargate' and 'Amazon EC2 instances' unchecked), 'OS, Architecture, Network mode' (with 'Network mode' selected), and 'Operating system/Architecture' (set to 'Linux/X86_64').

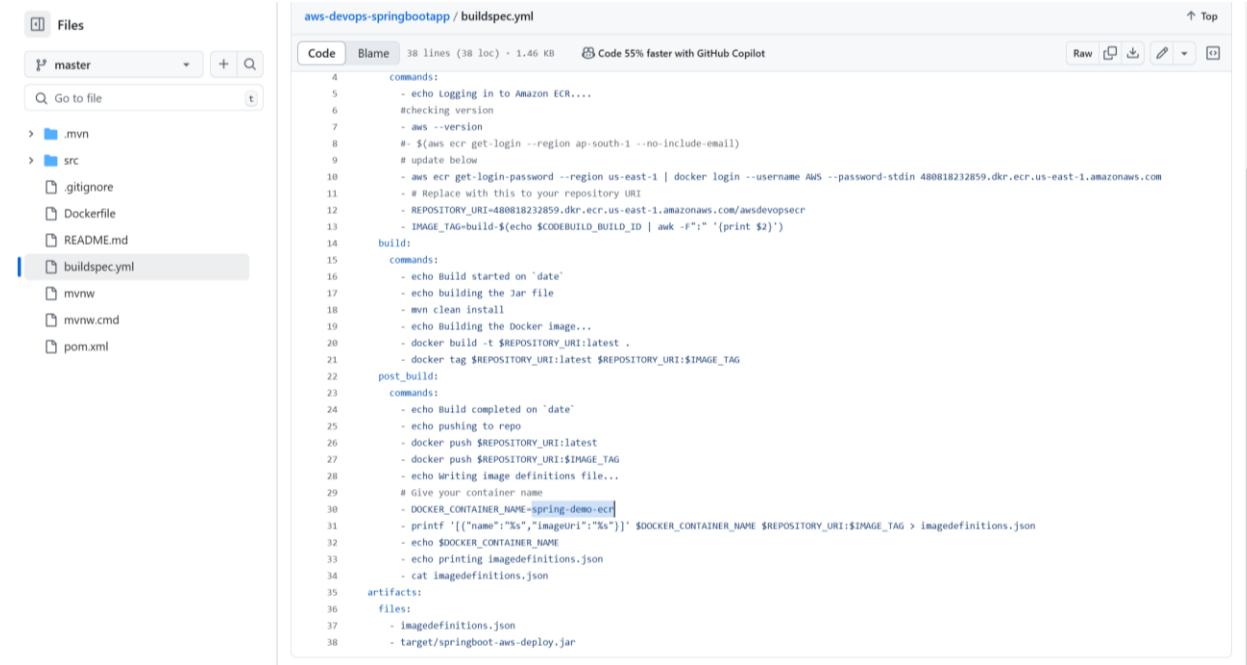
Give a name for the task definition family. Be sure it is an unique name.

In Infrastructure requirements, select AWS Fargate.

Scroll down to Container -1.

Provide the name of the Container: spring-demo-ecr

As shown in the buildspec.yml on GitHub.



The screenshot shows the GitHub interface with the repository 'aws-devops-springbootapp' open. The 'Code' tab is selected, displaying the contents of the 'buildspec.yml' file. The file contains YAML code for building a Docker image and pushing it to ECR. Key parts include defining commands for logging in to ECR, checking version, and building the Docker image. It also includes sections for post-build steps and artifacts like 'imagedefinitions.json'. The file is 38 lines long and 1.46 KB in size.

Repository name	URI	Created at	Tag immutability	Encryption type
awsdevopsecr	480818232859.dkr.ecr.us-east-1.amazonaws.com/awsdevopsecr	December 22, 2024, 15:49:12 (UTC-08)	Mutable	AES-256

Copy the Image URI from the ECR repository you made.

Copy ECR URI as your image URI for Task definition.

Be sure to change the container port to 8080.

This is how you will do Docker through AWS.

Click on Create.

Task created.

3.3 Services

The screenshot shows the AWS ECS Clusters page. A green success message at the top states: "Task definition successfully created" and "AWSDevOpsTaskA:1 has been successfully created. You can use this task definition to deploy a service or run a task." Below this, the "Clusters (1)" section is displayed, showing a single cluster named "awsdevopsclusterA". The cluster details are as follows:

Cluster	Services	Tasks	Container instances	CloudWatch monitoring	Capacity provider
awsdevopsclusterA	0	No tasks running	0 EC2	Default	No default

Go back to ECS and click your cluster.

The screenshot shows the "awsdevopsclusterA" cluster overview page. The "Cluster overview" section displays the following information:

ARN	Status	CloudWatch monitoring	Registered container instances
arn:aws:ecs:us-east-1:480818232859:cluster/awsdevopsclusterA	Active	Default	-

The "Services" section shows:

Draining	Active	Pending	Running
-	-	-	-

The "Encryption" section shows:

Managed storage	Fargate ephemeral storage
-	-

Below the cluster overview, there are tabs for "Services", "Tasks", "Infrastructure", "Metrics", "Scheduled tasks", and "Tags". The "Services" tab is selected, showing a table with one row:

Service name	ARN	Status	Service...	Deployments and tasks	Last deploy...
-	ARN	-	-	-	-

A note at the bottom of the services table says: "No services No services to display."

In Services, click on Create.

The screenshot shows the 'Create service' step in the AWS Elastic Container Service wizard. On the left, a sidebar lists various services like Clusters, Amazon ECR, and AWS Batch. The main area is titled 'Create' and shows the 'Compute configuration (advanced)' section. Under 'Compute options', the 'Launch type' dropdown is set to 'FARGATE'. Under 'Platform version', it is set to 'LATEST'. A note at the top of this section says: 'To ensure task distribution across your compute types, use appropriate compute options.' Below the configuration, there is a note: '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.'

Select Launch Type. Be sure to have Launch type set to Fargate and Latest platform version.

The screenshot shows the 'Deployment configuration' step in the AWS Elastic Container Service wizard. It includes sections for 'Application type', 'Task definition', 'Service name', 'Service type', and 'Desired tasks'. In the 'Application type' section, 'Service' is selected. In the 'Task definition' section, 'Specify the revision manually' is chosen, and the 'Family' dropdown is set to 'AWSDevOpsTaskA' with 'Revision' set to '1 (LATEST)'. The 'Service name' field contains 'AWSdevopssevice1'. In the 'Service type' section, 'Replica' is selected. The 'Desired tasks' field has the value '1'.

Select your task definition you created and create a service name for it. AWSdevopssevice1

Under Network.

Create a new security group. Set inbound rules to Type: All traffic and Source: Anywhere.

Now select Create. Give it a few minutes to develop.

Service has been created.

Check tasks to see if it's running. As it is running select your task.

AWSdevopssevice1 has been deployed successfully.

85ba1e75d5f94922a6a02e3044147a16

Configuration | Logs | Networking | Volumes (0) | Tags

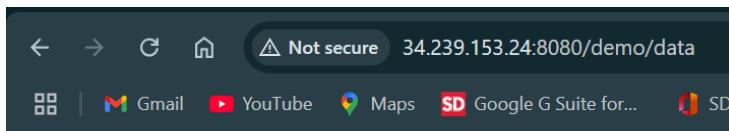
Task overview

ARN arn:aws:ecs:us-east-1:48081823285 9:task/awsdevopscusterA/85ba1e75d5f 94922a6a02e3044147a16	Last status Running	Desired status Running	Started/Created at December 22, 2024 at 18:44 (UTC-8:00) December 22, 2024 at 18:44 (UTC-8:00)
Fargate ephemeral storage	Encryption Info Default AWS Fargate encryption	Size (GiB) 20	

Configuration

Operating system/Architecture Linux/X86_64	Capacity provider -	ENI ID eni-07535fc8c8ad564a	Public IP 34.239.153.24 open address
CPU Memory 1 vCPU 3 GB	Launch type FARGATE	Network mode awsvpc	Private IP 172.31.75.156
Platform version 1.4.0	Container instance ID -	Subnet ID subnet-05db63773a6fb8a68	MAC address 16:ff:dc:fa:e2:87

Copy Public IP to check if it is running. Be sure to add PublicIP:8080/demo/data



First message from AWS Ecs

4. Code Pipeline

4.1 Create Pipeline

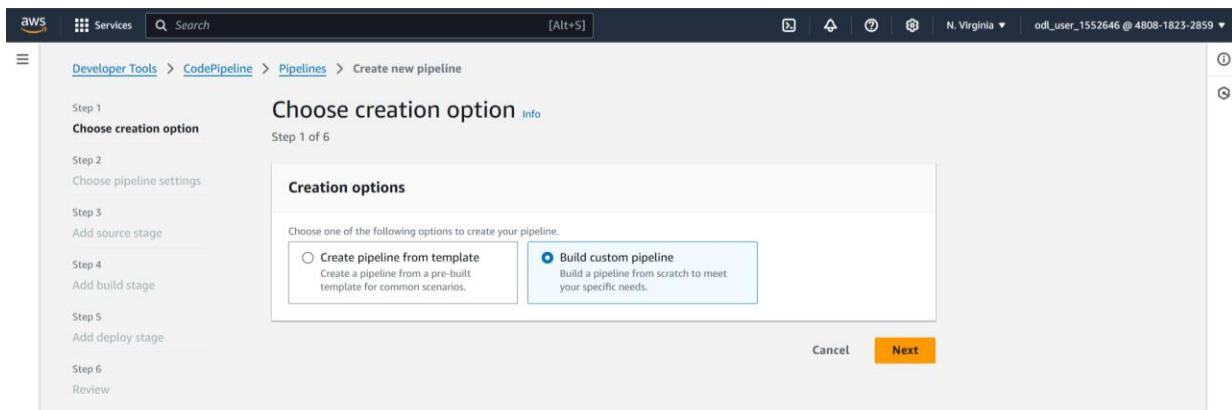
Go to AWS CodePipeline. Select Pipelines.

Developer Tools > CodePipeline > Pipelines

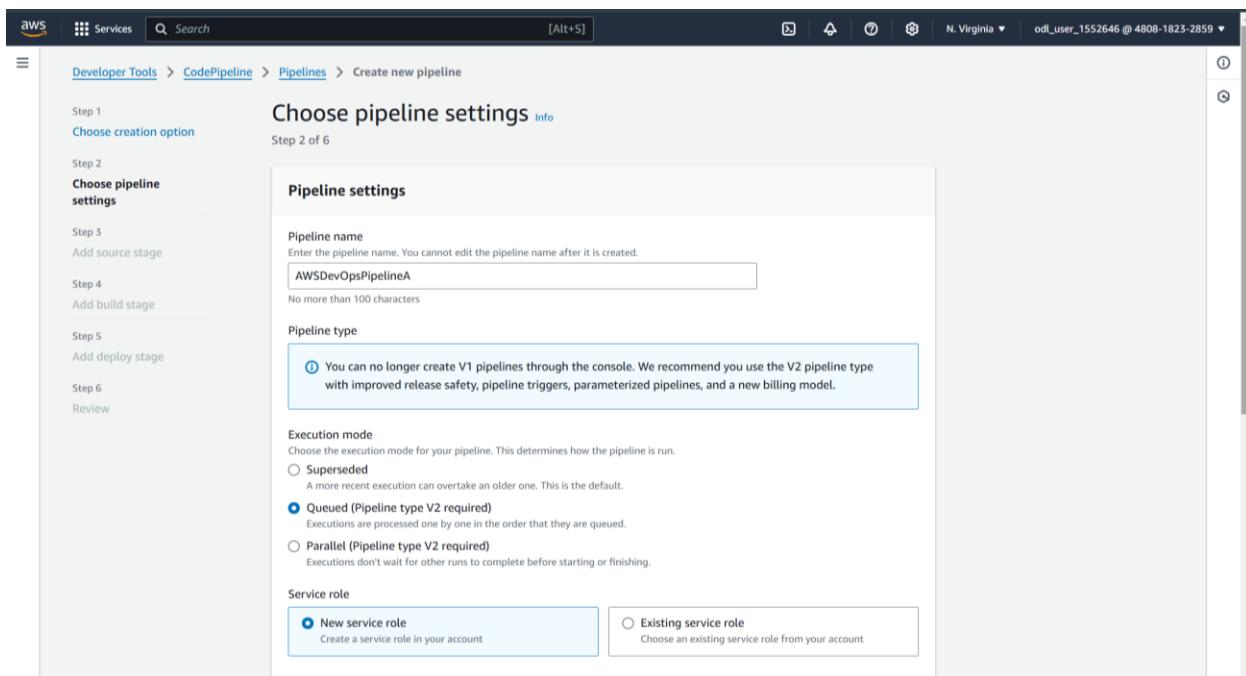
Pipelines info

Name	Latest execution status	Latest source revisions	Latest execution started	Most recent executions
No results There are no results to display.				

Select Create pipeline.



In the first step, select “Build custom pipeline”. Click Next.



In second step, create a name for your pipeline. AWSDevOpsPipelineA.

Leave default settings. Click Next.

In step 3, Add source stage, you have two options on how to connect to Github.

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 deploy stage

Step 6 Review

Add source stage Info

Step 3 of 6

Source

Source provider

This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

AWS CodeCommit
Amazon ECR
Amazon S3
Bitbucket
GitHub (via GitHub App)
GitHub (via OAuth app)
GitHub Enterprise Server
GitLab
GitLab self-managed

Previous Next

Either GitHub App or OAuth app. In this case, we will use OAuth app.

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 deploy stage

Step 6 Review

Add source stage Info

Step 3 of 6

Source

Source provider

This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

GitHub (via OAuth app)

Grant AWS CodePipeline access to your GitHub repository. This allows AWS CodePipeline to upload commits from GitHub to your pipeline.

Connect to GitHub

The GitHub (via OAuth app) action is not recommended

The selected action uses OAuth apps to access your GitHub repository. This is no longer the recommended method. Instead, choose the GitHub (via GitHub App) action to access your repository by creating a connection. Connections use GitHub Apps to manage authentication and can be shared with other resources. [Learn more](#)

Change detection options

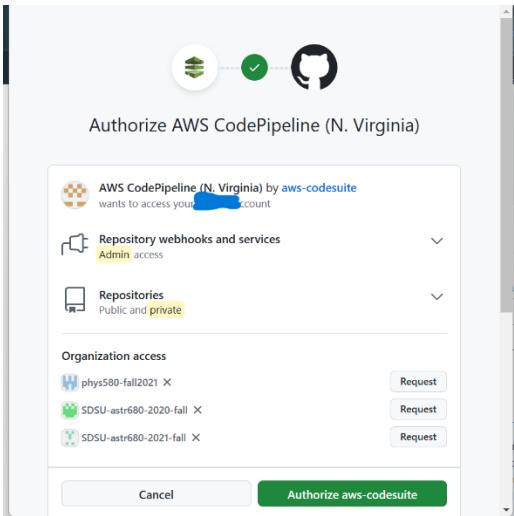
Choose a detection mode to automatically start your pipeline when a change occurs in the source code.

GitHub webhooks (recommended)
Use webhooks in GitHub to automatically start my pipeline when a change occurs

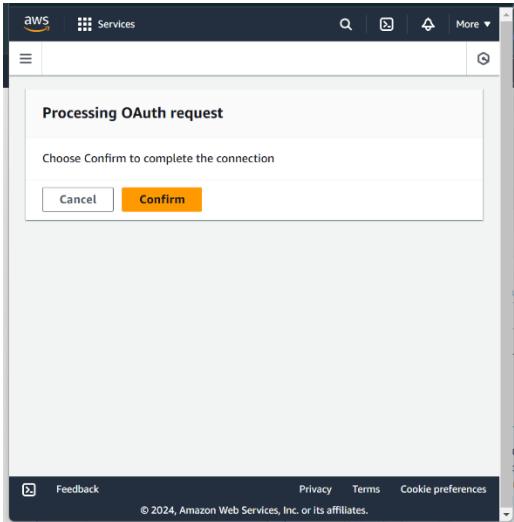
AWS CodePipeline
Use AWS CodePipeline to check periodically for changes

Enable automatic retry on stage failure

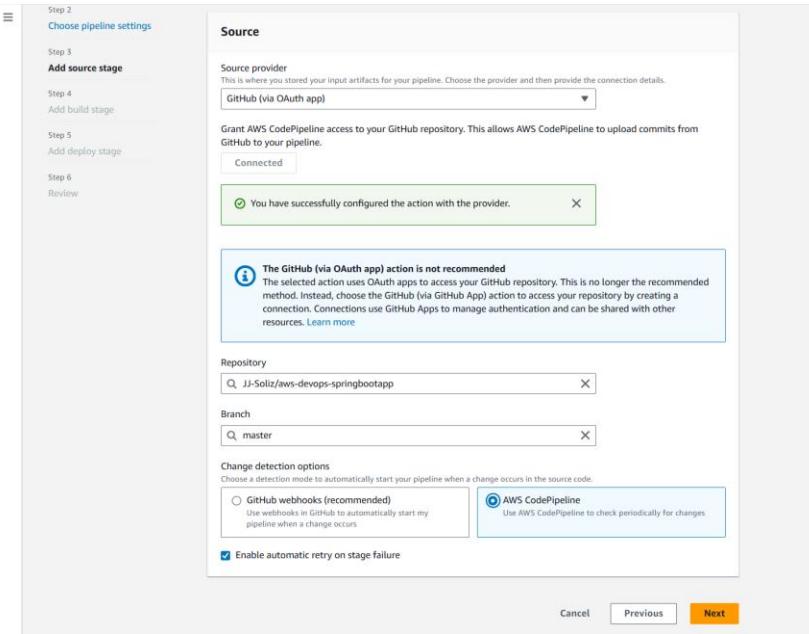
Click on Connect to GitHub. A popup window will appear to connect your GitHub account.



Click Authorize aws-codesuite. You will need to provide an passkey, authorization code or password. Once provide you will be sent to this window.



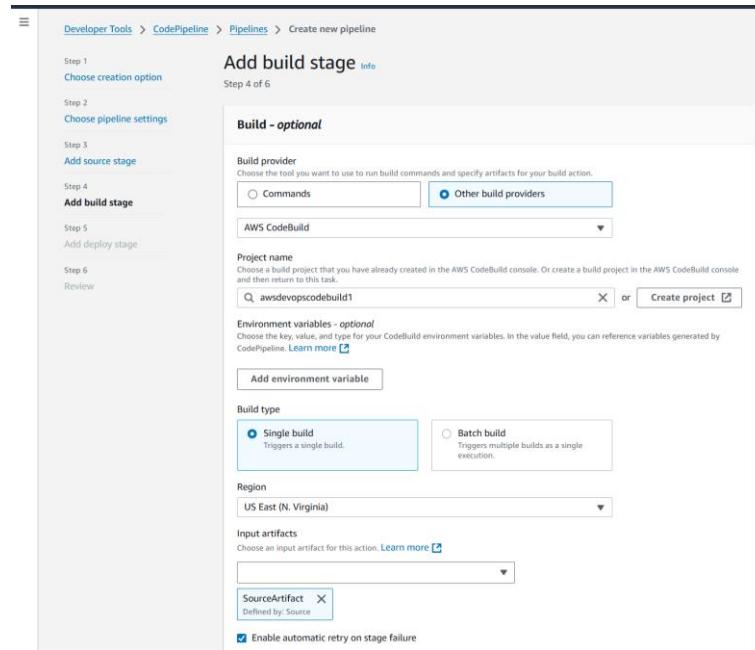
Click Confirm.



With your GitHub account connected, select your Spring Boot App repository and master branch. You have the option on detection mode. In this case, we will have it set to AWS CodePipeline. Select Next.

In step 4, Add build stage.

Select Other build providers. Select AWS CodeBuild then under project name, select your code build you created.



Click Next.

In step 5, Add deploy stage.

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 deploy stage

Step 6 Review

Add deploy stage Info

Step 5 of 6

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.

Amazon ECS

Region
US East (N. Virginia)

Input artifacts
Choose an input artifact for this action. [Learn more](#)

BuildArtifact X
Defined by: Build

No more than 100 characters

Cluster name
Choose a cluster that you have already created in the Amazon ECS console. Or create a cluster in the Amazon ECS console and then return to this task.

awsdevopsclusterA

Service name
Choose a service that you have already created in the Amazon ECS console for your cluster. Or create a new service in the Amazon ECS console and then return to this task.

AWSdevopssevice1

Image definitions file - optional
Enter the JSON file that describes your service's container name and the image and tag.
MyFilename.json

Deployment timeout - optional
Enter the timeout in minutes for the deployment action.

Configure automatic rollback on stage failure
 Enable automatic retry on stage failure

[Cancel](#) [Previous](#) [Skip deploy stage](#) **Next**

Select Amazon ECS in Deploy provider.

US East (N. Virginia)

Input artifacts
Choose an input artifact for this action. [Learn more](#)

BuildArtifact X
Defined by: Build

No more than 100 characters

Cluster name
Choose a cluster that you have already created in the Amazon ECS console. Or create a cluster in the Amazon ECS console and then return to this task.

awsdevopsclusterA X

Service name
Choose a service that you have already created in the Amazon ECS console for your cluster. Or create a new service in the Amazon ECS console and then return to this task.

AWSdevopssevice1 X

Image definitions file - optional
Enter the JSON file that describes your service's container name and the image and tag.
MyFilename.json

Deployment timeout - optional
Enter the timeout in minutes for the deployment action.

Configure automatic rollback on stage failure
 Enable automatic retry on stage failure

[Cancel](#) [Previous](#) [Skip deploy stage](#) **Next**

Select your cluster and service you created. Then click Next.

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 deploy stage
Step 6 Review

Review Info Step 6 of 6

Step 2: Choose pipeline settings

Pipeline settings

Pipeline name: AWSDevOpsPipelineA
Pipeline type: V2
Execution mode: QUEUED
Artifact location: A new Amazon S3 bucket will be created as the default artifact store for your pipeline
Service role name: AWSCodePipelineServiceRole-us-east-1-AWSDevOpsPipelineA

Step 3: Add source stage

Source action provider

Source action provider: GitHub (via OAuth app)
PollForSourceChanges: true
Repo: awssdeops-springbootapp
Owner: JI-Soliz
Branch: master
Enable automatic retry on stage failure: Enabled

Step 4: Add build stage

Build action provider

Build action provider: AWS CodeBuild
Project Name: awssdevopscodebuild1
Commands:
-
Enable automatic retry on stage failure: Enabled

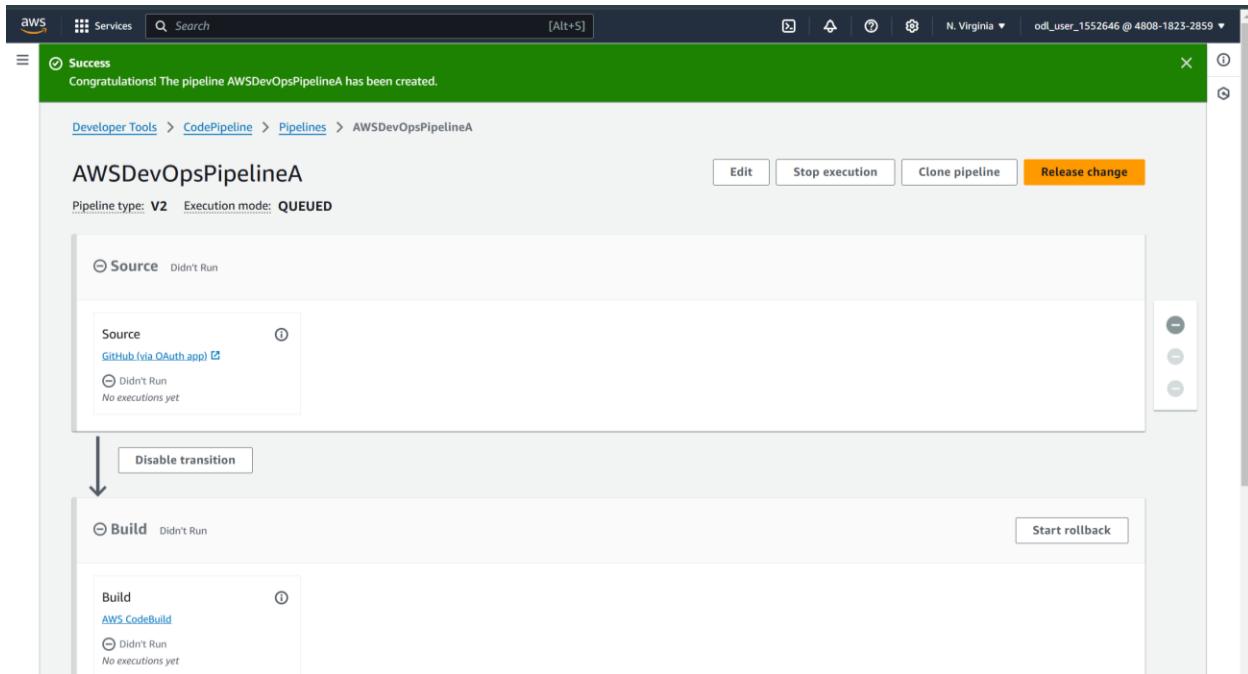
Step 5: Add deploy stage

Deploy action provider

Deploy action provider: Amazon ECS
Cluster Name: awssdevopsclusterA
Service Name: AWSSdevopservice1
Configure automatic rollback on stage failure: Enabled
Enable automatic retry on stage failure: Disabled

Cancel Previous **Create pipeline**

Review your pipeline. Lastly, click on Create pipeline.

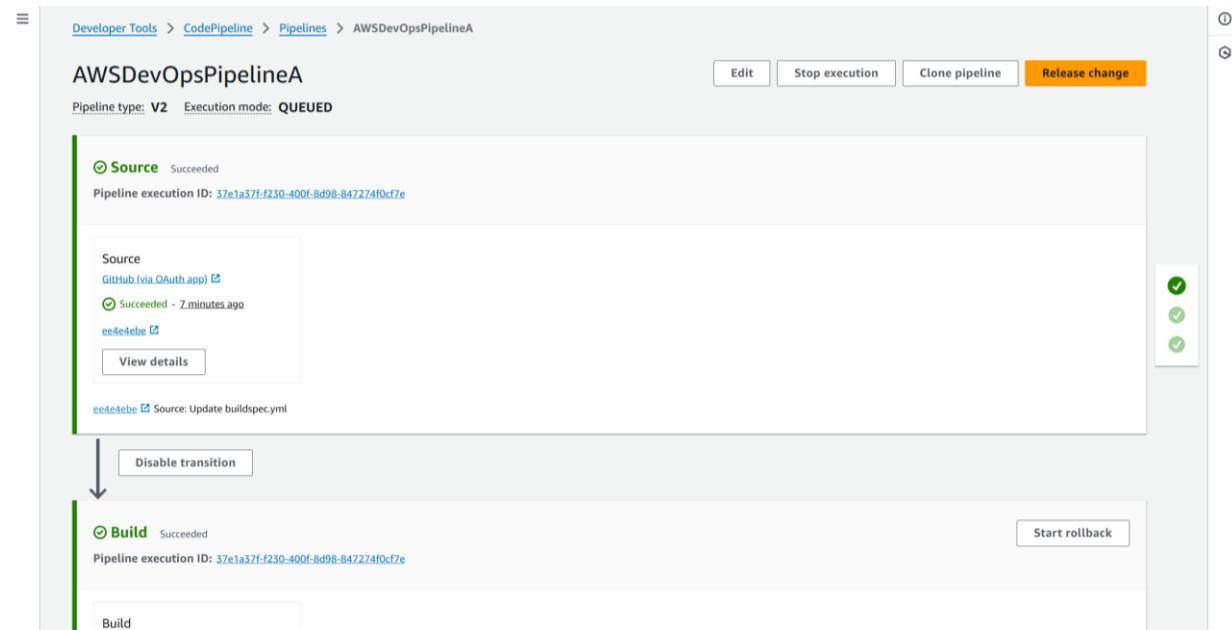


Pipeline created.

4.2 Run Pipeline

Now click on “Release change”.

This will take about 10 minutes to complete.



The screenshot shows a CloudFormation Pipeline named 'My Pipeline' with two stages: 'Build' and 'Deploy'. The 'Build' stage was successful, triggered by a GitHub event, and the 'Deploy' stage was also successful, triggered by an Amazon ECS event. Both stages completed just now. A green checkmark icon is visible on the right side of the pipeline.

Build Succeeded
Pipeline execution ID: [37e1a37f-f230-400f-8d98-847274f0cf7e](#)

Deploy Succeeded
Pipeline execution ID: [37e1a37f-f230-400f-8d98-847274f0cf7e](#)

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Pipeline has been successfully made.

Go to ECR and click on your repository.

The screenshot shows the AWS ECR console with the 'awsdevopsecr' repository selected. Three Docker images have been pushed to the repository, each with a size of 192.58 MB and pushed at December 22, 2024, 19:50:29 UTC-08. The images are listed in a table with columns: Image tag, Artifact type, Pushed at, Size (MB), Image URI, and Digest.

Images (3)

Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
build-3d6b51b9-e492-4846-833a-6e1cd519cc6, latest	Image	December 22, 2024, 19:50:29 (UTC-08)	192.58	Copy URI	sha256:3b13019e41a9c9...
build-fcc3c7bf-0b69-4481-a145-34eadfc79335	Image	December 22, 2024, 19:49:02 (UTC-08)	192.58	Copy URI	sha256:d6f27c3cb8beee4...
build-d12fe526-b05d-43f6-9983-01883476f083	Image	December 22, 2024, 17:58:25 (UTC-08)	192.58	Copy URI	sha256:e4a4ec56a9ccf36...

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

You will see Docker images have been made.

The screenshot shows the AWS ECR interface. On the left, a sidebar navigation includes 'Amazon Elastic Container Registry' (selected), 'Private registry' (expanded), 'Public registry' (expanded), and links to 'ECR public gallery', 'Amazon ECS', 'Amazon EKS', 'Getting started', and 'Documentation'. The main content area is titled 'Image' and displays the following details:

- Details**
 - Image tags**: build-3d6b51b9-e492-4846-833a-6e1cd519cce6, latest
 - URI**: 480818232859.dkr.ecr.us-east-1.amazonaws.com/awsdevopsecr:build-3d6b51b9-e492-4846-833a-6e1cd519cce6
 - Digest**: sha256:3b13019e41a9c980380f4cbde78aec592b26de66608d64b1e50955f1c98e4721
- General information**

Artifact type	Repository	Pushed at
Image	awsdevopsecr	December 22, 2024, 19:50:29 (UTC-08)
- Scanning and vulnerabilities**

Status: Scan not found
- Referrers** [Info](#)

At the bottom, there are links for 'CloudShell', 'Feedback', and copyright information: '© 2024, Amazon Web Services, Inc. or its affiliates.' followed by 'Privacy', 'Terms', and 'Cookie preferences'.

Your CI/CD Pipeline for Spring Boot Application is Ready!

Finish