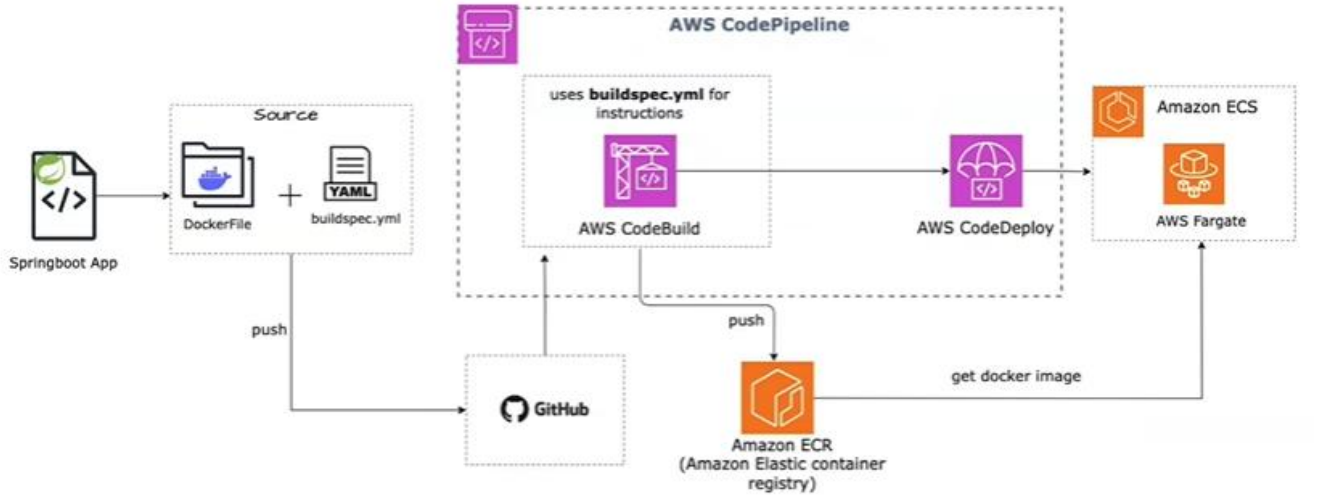
Spring ECS Automate

Project Created By: AMAN DUGGAL
LinkedIn: https://www.linkedin.com/in/aman-duggal-4591bb146



Project Introduction:

The project aimed to streamline the deployment process of a Spring Boot application on Amazon ECS through automation using AWS services.

Objective:

To create an automated pipeline for deploying a Spring Boot application on Amazon ECS efficiently.

Key AWS Services Used:

1. **AWS CodePipeline:** Orchestrated the entire deployment process, from source code changes to deployment on Amazon ECS.

2. **AWS CodeBuild:** Used to build the Spring Boot application and Docker image.

3. **Amazon ECR (Elastic Container Registry):** Stored Docker images.

4. **AWS CodeDeploy:** Deployed the Docker image onto Amazon ECS.

5. **Amazon ECS (Elastic Container Service):** Hosted and managed the Docker containers.

6. **AWS ChatBot:** Created notification rules in CodePipeline to notify Slack for each stage execution.

Implementation Steps:

1. Code Setup:

- Developed a Spring Boot application with a REST API and Dockerfile.
- Pushed code to GitHub for version control.

2. AWS CodeBuild:

- Created an ECR repository.
- Configured AWS CodeBuild with GitHub as a source provider.
- Defined build instructions in `buildspec.yaml`.
- Built the project, created Docker image, and pushed it to ECR.

3. AWS CodeDeploy:

- Created an ECS cluster and task definition.
- Configured AWS CodeDeploy to deploy the Docker image onto ECS.

4. AWS CodePipeline:

- Created a CodePipeline.
- Added source, build, and deploy stages.
- Integrated GitHub as a source provider.
- Specified CodeBuild project for building and deploying the application.

5. SLACK Notifications:

- Integrated AWS Chatbot with Slack.
- Created notification rules in CodePipeline to notify Slack for each stage execution.

Outcome:

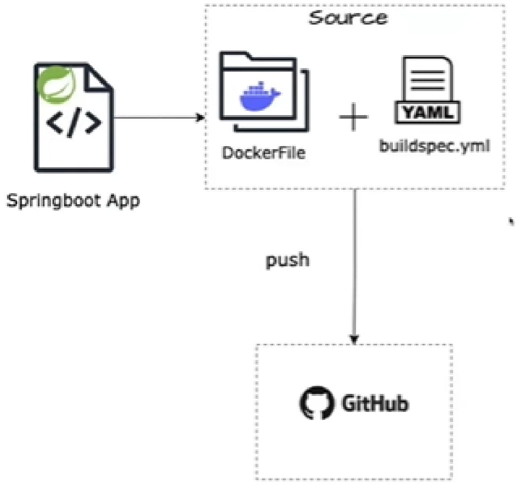
A fully automated pipeline that builds and deploys the Spring Boot application onto Amazon ECS whenever changes are pushed to the code repository.

Purpose:

Showcase the efficiency and power of automation in deployment processes, leveraging AWS services for rapid and seamless deployments.

Step 1: Code Setup

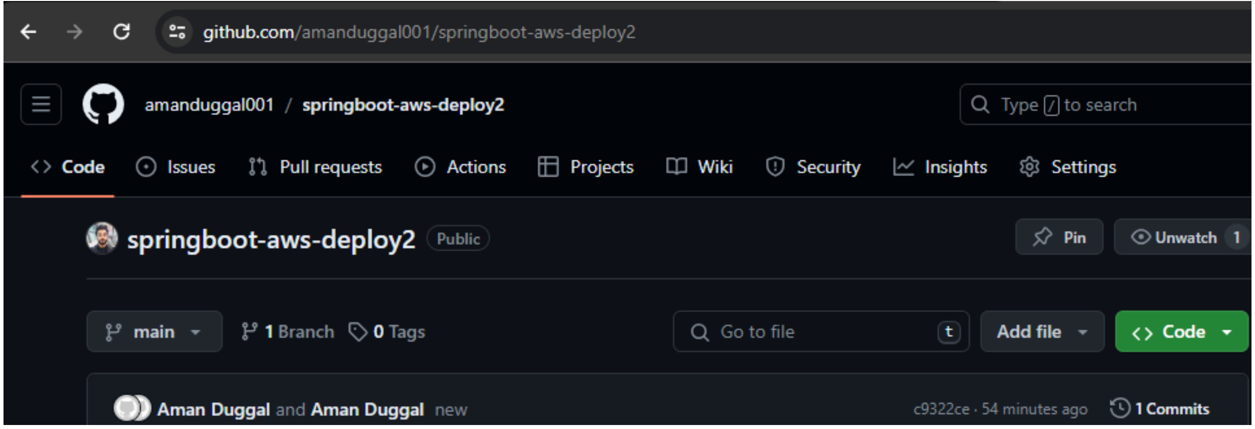
Step 1 - Code Setup



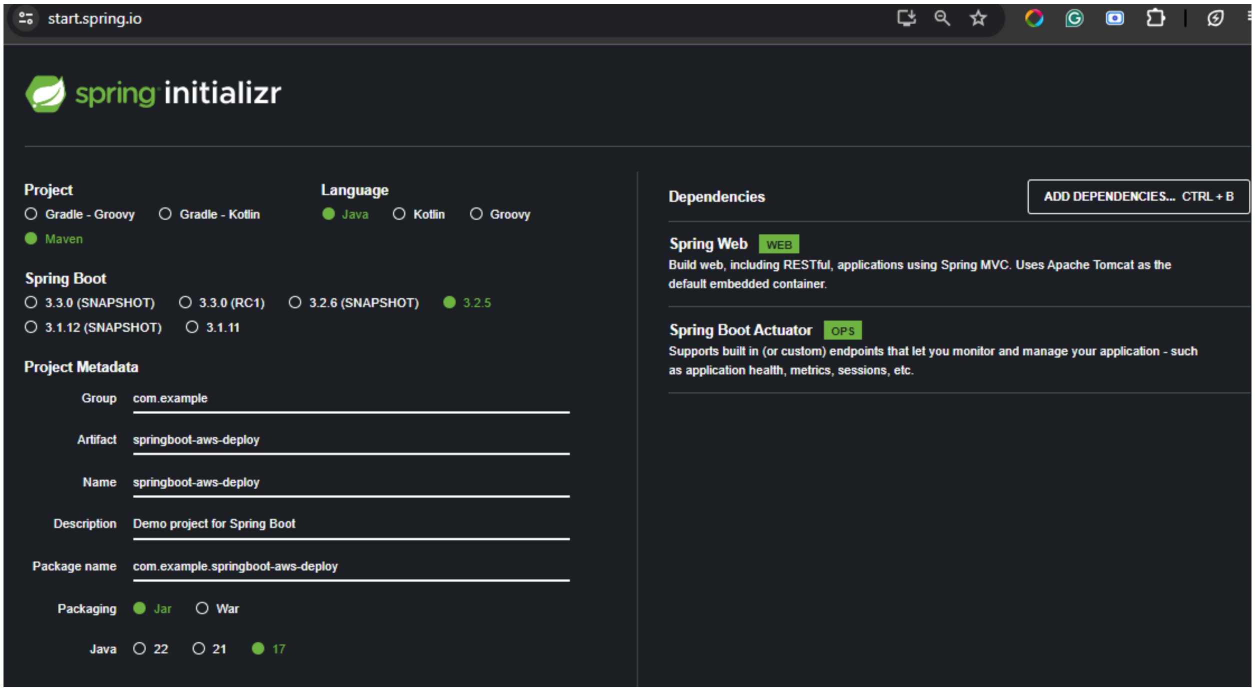
1. Create repository in Github
2. Create Microservice
3. Docker File
4. Push code in GitHub

- Create a new repository on your GitHub account.

https://github.com/amanduggal001/springboot-aws-deploy2



- Open https://start.spring.io/ and generate the source code.



Project
○ Gradle - Groovy ○ Gradle - Kotlin ● Maven

Language
● Java ○ Kotlin ○ Groovy

Spring Boot
○ 3.3.0 (SNAPSHOT) ○ 3.3.0 (RC1) ○ 3.2.6 (SNAPSHOT) ● 3.2.5
○ 3.1.12 (SNAPSHOT) ○ 3.1.11

Project Metadata

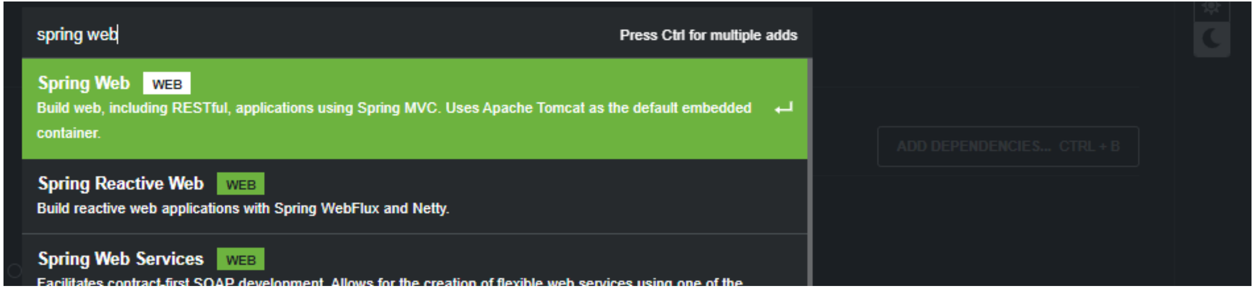
Group	com.example
Artifact	springboot-aws-deploy
Name	springboot-aws-deploy
Description	Demo project for Spring Boot
Package name	com.example.springboot-aws-deploy
Packaging	● Jar ○ War
Java	○ 22 ○ 21 ● 17

Dependencies

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Boot Actuator OPS
Supports built in (or custom) endpoints that let you monitor and manage your application - such as application health, metrics, sessions, etc.

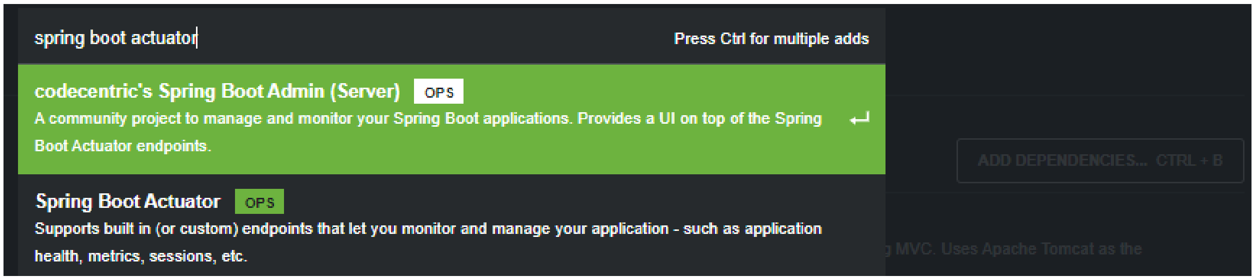
- Click on Add Dependencies > Add **'Spring Web' & 'Spring Boot Actuator'** > Generate


spring web

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Reactive Web WEB
Build reactive web applications with Spring WebFlux and Netty.

Spring Web Services WEB
Facilitates contract-first SOAP development. Allows for the creation of flexible web services using one of the


spring boot actuator

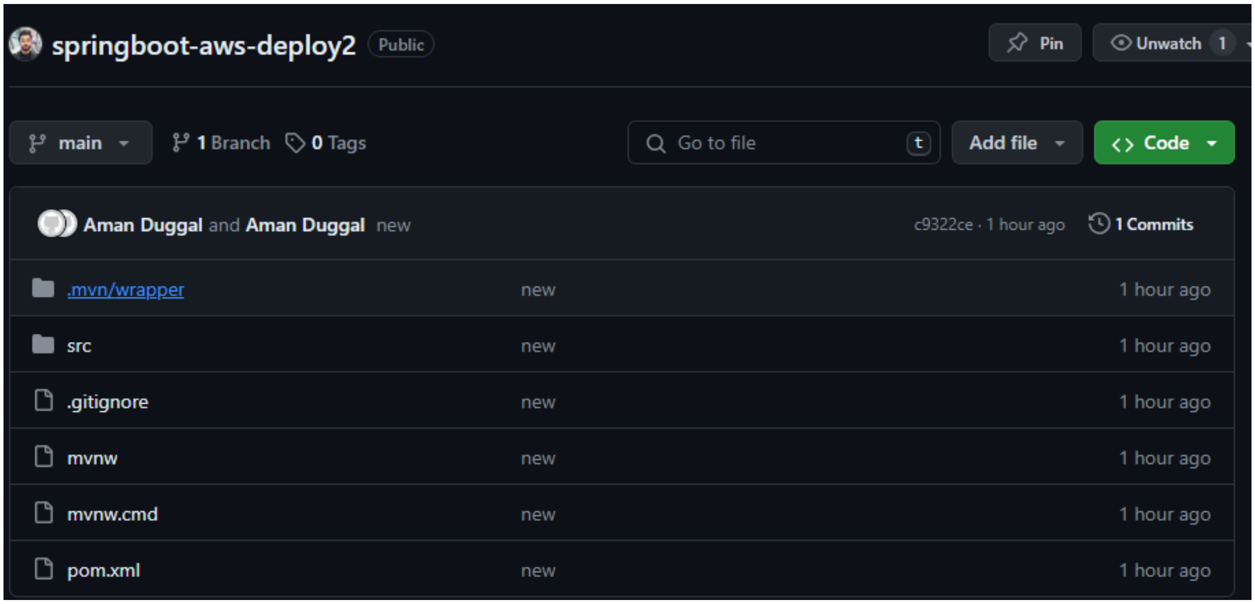
codecentric's Spring Boot Admin (Server) OPS
A community project to manage and monitor your Spring Boot applications. Provides a UI on top of the Spring Boot Actuator endpoints.

Spring Boot Actuator OPS
Supports built in (or custom) endpoints that let you monitor and manage your application - such as application health, metrics, sessions, etc.

- Once the Spring Boot code is downloaded, open the Git bash and push the code in our git hub repository.

```
unzip springboot-aws-application.zip
cd springboot-aws-application
git init
```

git remote -v --#To check the existing repository added in the git
git remote add origin https://github.com/amanduggal001/springboot-aws-deploy2 --#To add the repository in git.
git status
git add .
git commit -m "Added files"
git push origin master



Aman Duggal and Aman Duggal new c9322ce · 1 hour ago 1 Commits

.mvn/wrapper	new	1 hour ago
src	new	1 hour ago
.gitignore	new	1 hour ago
mvnw	new	1 hour ago
mvnw.cmd	new	1 hour ago
pom.xml	new	1 hour ago

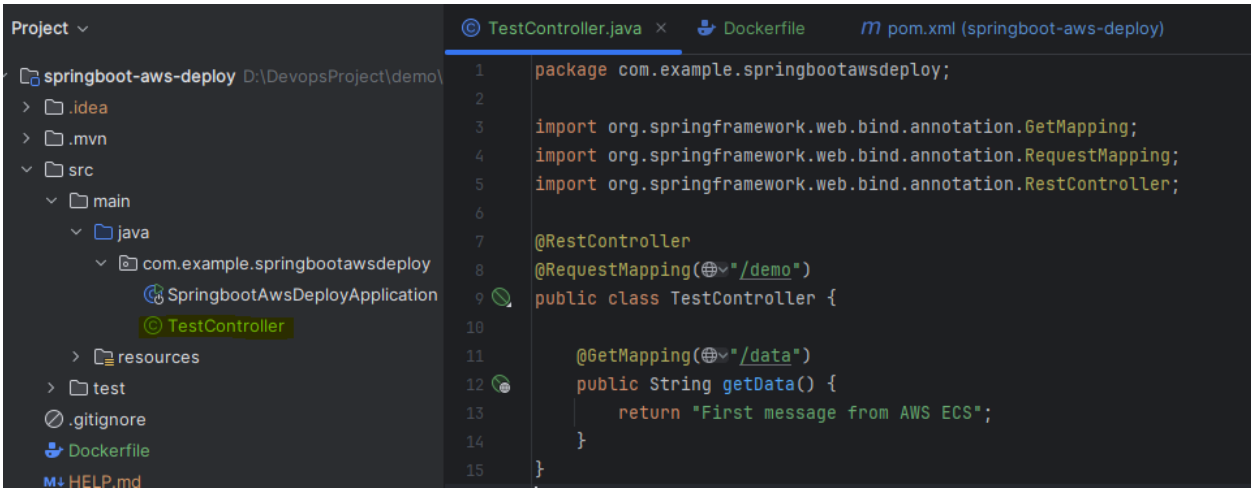
- Now, Add One controller (Java Script) to our application and name it a Testcontroller.
- Create file in **springboot-aws-application\src\main\java\com\example\springbootawsapplication or use intelliJ.**

Testcontroller.java

```
packagecom.example.springbootawsdeploy;

importorg.springframework.web.bind.annotation.GetMapping;
importorg.springframework.web.bind.annotation.RequestMapping;
importorg.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/demo")
publicclassTestController{

@GetMapping("/data")
publicStringgetData(){
return"FirstmessagefromAWSECS";
}
}
```

Basically, this controller defines a single endpoint ("/demo/data") that returns a static message when accessed via an HTTP GET request. It's a basic example demonstrating how to create a RESTful endpoint in a Spring Boot application.

So, we have just created a very simple rest API which will return a string message that is the first message from AWS ECS.


Project
springboot-aws-deploy D:\DevopsProject\demo\
.idea
.mvn
src
main
java
com.example.springbootawsdeploy
SpringbootAwsDeployApplication
TestController
resources
test
.gitignore
Dockerfile
HELP.md
TestController.java
Dockerfile
pom.xml (springboot-aws-deploy)

```java
package com.example.springbootawsdeploy;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/demo")
public class TestController {

    @GetMapping("/data")
    public String getData() {
        return "First message from AWS ECS";
    }
}
```

- Now to make our application dockerize we will add a Docker file to our application.

Dockerfile

```
FROMeclipse-temurin:17-jdk-alpine
RUNapkaddcurl
VOLUME/tmp
EXPOSE8080
ADDtarget/springboot-aws-deploy-service.jarspringboot-aws-deploy-service.jar
ENTRYPOINT["java","-jar","/springboot-aws-deploy-service.jar"]
```

1. Base Image (`FROM eclipse-temurin:17-jdk-alpine`):
 - Specifies the base image as `eclipse-temurin:17-jdk-alpine`, which provides Java 17 runtime environment on Alpine Linux.

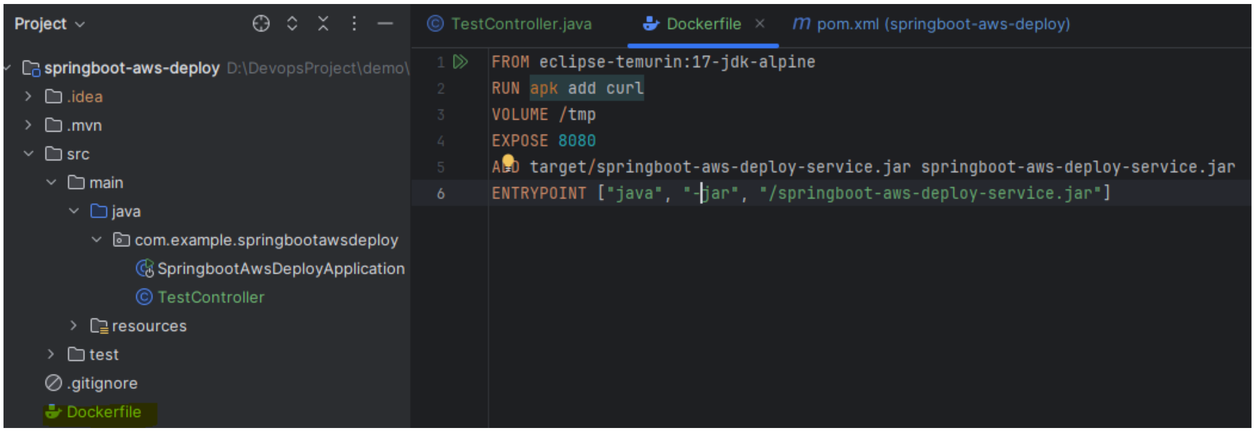
2. Installing Dependencies (`RUN apk add curl`):
 - Installs `curl` package using Alpine package manager (`apk`).

3. Volume Declaration (`VOLUME /tmp`):
 - Declares a volume at `/tmp`, allowing files to be shared between the container and the host.

4. Port Exposition (`EXPOSE 8080`):
 - Exposes port 8080 to allow external access to the Spring Boot application.

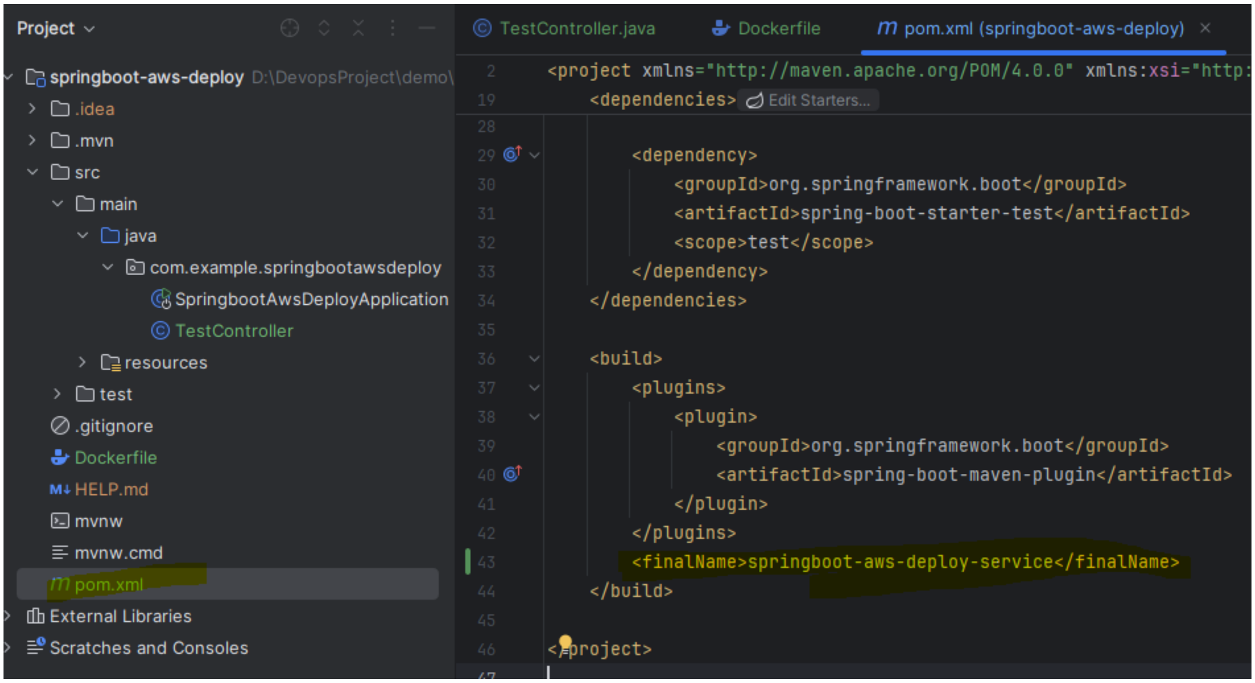
5. Application Setup (`ADD target/springboot-aws-application-service.jar springboot-aws-application-service.jar`):
 - Adds the Spring Boot application JAR file (`springboot-aws-application-service.jar`) from the `target` directory to the image.

6. Entrypoint Definition (`ENTRYPOINT ["java","-jar","/springboot-aws-application-service.jar"]`):
 - Defines the entrypoint command to execute the Spring Boot application using the `java` command with the `-jar` option and specifying the application JAR file.


Project
TestController.java
Dockerfile
pom.xml (springboot-aws-deploy)
springboot-aws-deploy D:\DevopsProject\demo\
.idea
.mvn
src
main
java
com.example.springbootawsdeploy
SpringbootAwsDeployApplication
TestController
resources
test
.gitignore
Dockerfile

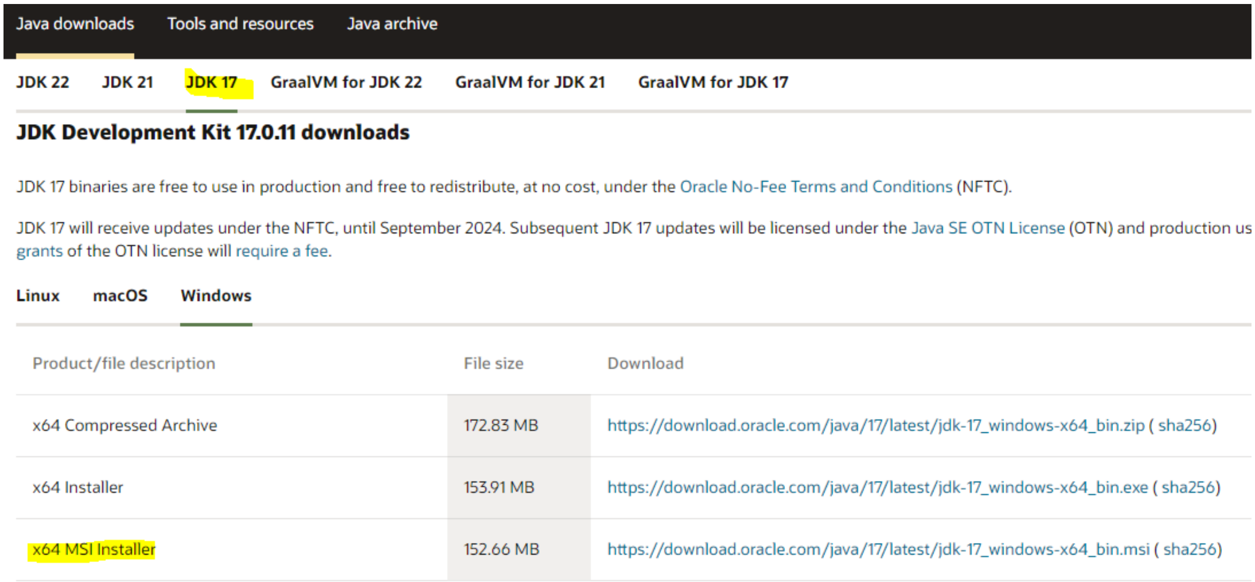
```
FROM eclipse-temurin:17-jdk-alpine
RUN apk add curl
VOLUME /tmp
EXPOSE 8080
ADD target/springboot-aws-deploy-service.jar springboot-aws-deploy-service.jar
ENTRYPOINT ["java", "-jar", "/springboot-aws-deploy-service.jar"]
```

- Add the below line in the POM.xml file.
<finalName>springboot-aws-deploy-service</finalName>



```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http:
    <dependencies>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
        <finalName>springboot-aws-deploy-service</finalName>
    </build>

</project>
```

- **Note:** Install the JDK17 in your windows OS. https://www.oracle.com/in/java/technologies/downloads/#jdk17-windows



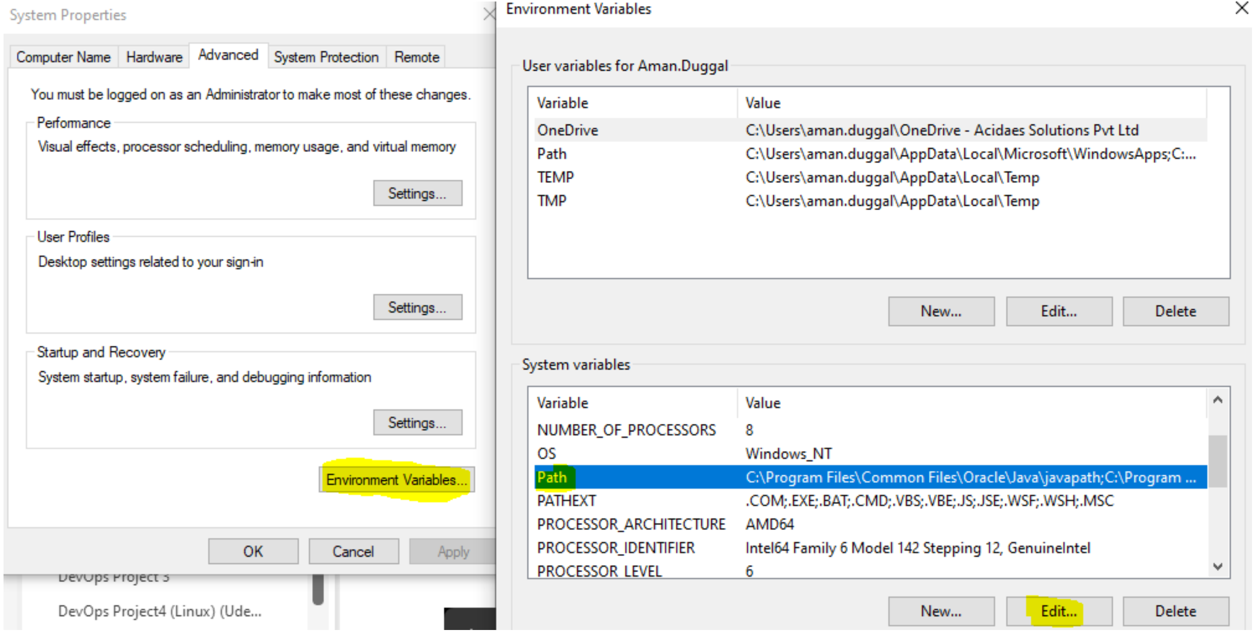
JDK Development Kit 17.0.11 downloads

JDK 17 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions (NFTC).

JDK 17 will receive updates under the NFTC, until September 2024. Subsequent JDK 17 updates will be licensed under the Java SE OTN License (OTN) and production us grants of the OTN license will require a fee.

Product/file description	File size	Download
x64 Compressed Archive	172.83 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.zip (sha256)
x64 Installer	153.91 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.exe (sha256)
x64 MSI Installer	152.66 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.msi (sha256)

- Now, add Java path to Environment variables.



You must be logged on as an Administrator to make most of these changes.

Performance

Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles

Desktop settings related to your sign-in

Startup and Recovery

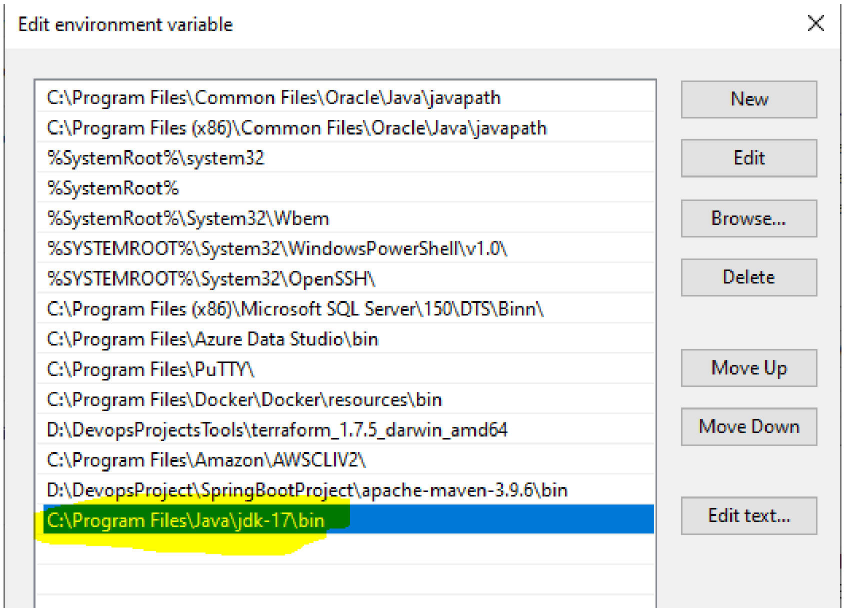
System startup, system failure, and debugging information

User variables for Aman.Duggal

Variable	Value
OneDrive	C:\Users\aman.duggal\OneDrive - Acidaes Solutions Pvt Ltd
Path	C:\Users\aman.duggal\AppData\Local\Microsoft\WindowsApps;C:...
TEMP	C:\Users\aman.duggal\AppData\Local\Temp
TMP	C:\Users\aman.duggal\AppData\Local\Temp

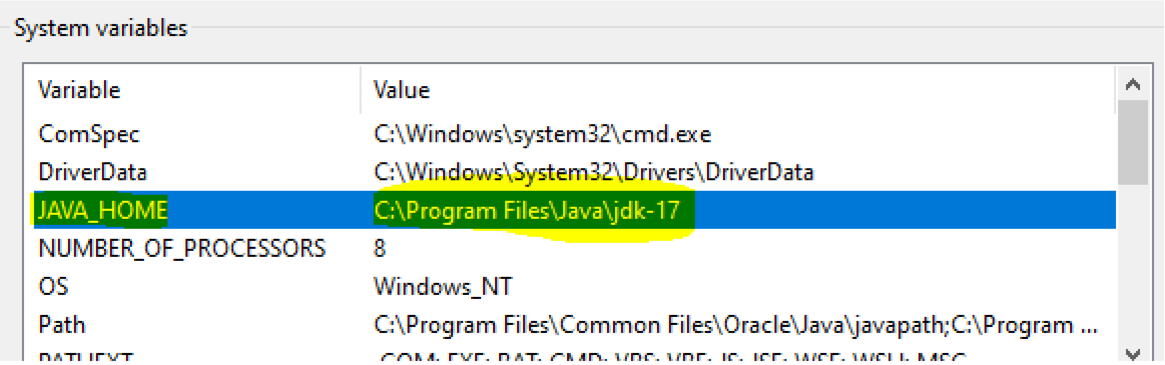
System variables

Variable	Value
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\Program Files\Common Files\Oracle\Java\javapath;C:\Program ...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 142 Stepping 12, GenuineIntel
PROCESSOR_LEVEL	6



```
C:\Program Files\Common Files\Oracle\Java\javapath
C:\Program Files (x86)\Common Files\Oracle\Java\javapath
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files (x86)\Microsoft SQL Server\150\DTS\Binn\
C:\Program Files\Azure Data Studio\bin
C:\Program Files\PuTTY\
C:\Program Files\Docker\Docker\resources\bin
D:\DevopsProjectsTools\terraform_1.7.5_darwin_amd64
C:\Program Files\Amazon\AWSCLIV2\
D:\DevopsProject\SpringBootProject\apache-maven-3.9.6\bin
C:\Program Files\Java\jdk-17\bin
```

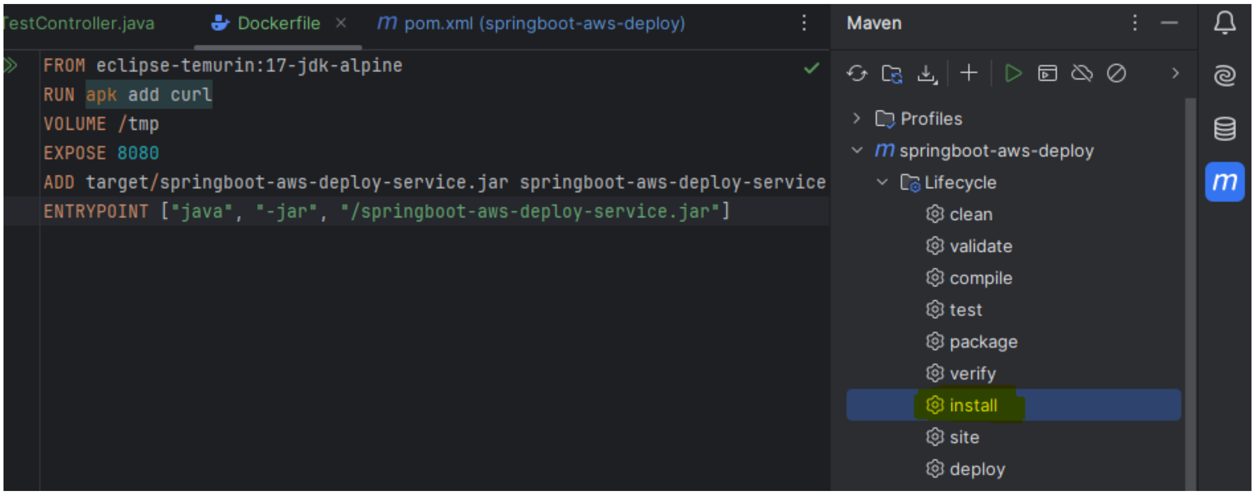
- Now, create a New System Variable with the name **'JAVA_HOME'** and provide the path.



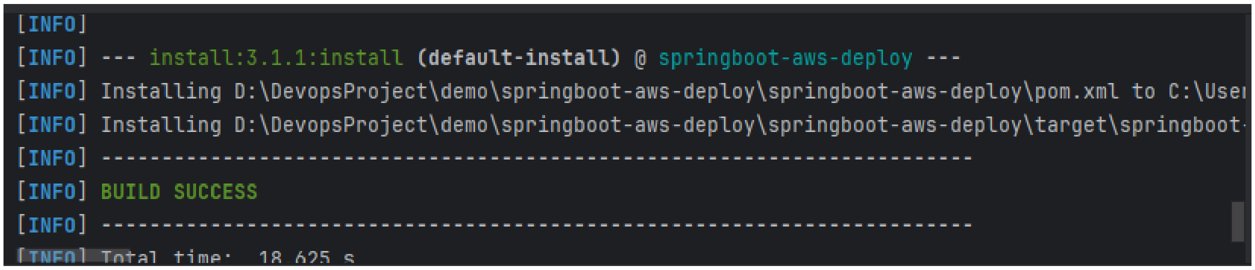
System variables

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
JAVA_HOME	C:\Program Files\Java\jdk-17
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\Program Files\Common Files\Oracle\Java\javapath;C:\Program ...

- Now, build the project from **intelliJ** or any other software application.



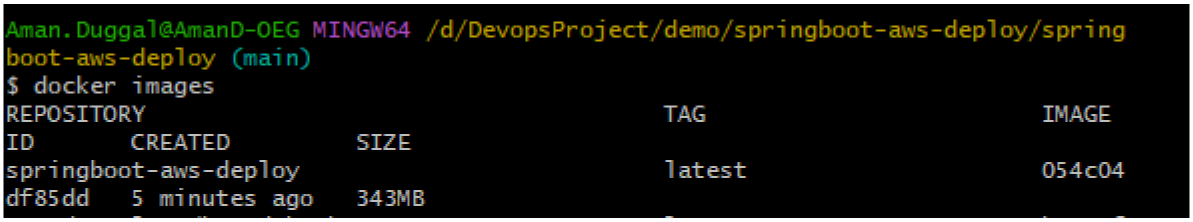
```
FROM eclipse-temurin:17-jdk-alpine
RUN apk add curl
VOLUME /tmp
EXPOSE 8080
ADD target/springboot-aws-deploy-service.jar springboot-aws-deploy-service
ENTRYPOINT ["java", "-jar", "/springboot-aws-deploy-service.jar"]
```



```
[INFO]
[INFO] --- install:3.1.1:install (default-install) @ springboot-aws-deploy ---
[INFO] Installing D:\DevopsProject\demo\springboot-aws-deploy\springboot-aws-deploy\pom.xml to C:\Use
[INFO] Installing D:\DevopsProject\demo\springboot-aws-deploy\springboot-aws-deploy\target\springboot
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  18.625 s
```

- Once the Build is successful, now create the image from the Dockerfile.

docker build -t springboot-aws-deploy .
docker images



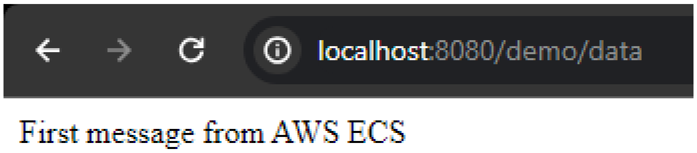
```
Aman.Duggal@AmanD-OEG MINGW64 /d/DevopsProject/demo/springboot-aws-deploy/spring
boot-aws-deploy (main)
$ docker images
REPOSITORY                                    TAG                        IMAGE
ID        CREATED          SIZE
springboot-aws-deploy                         latest                     054c04
df85dd    5 minutes ago    343MB
```

- Now, create the container from the image.

docker run -p 8080:8080 springboot-aws-deploy

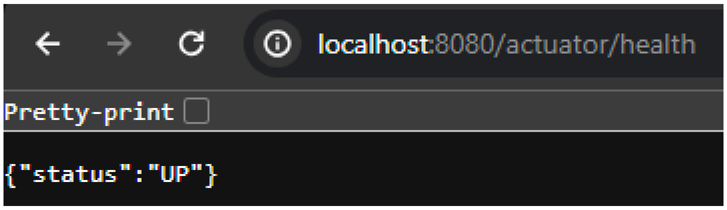
- Once done, now test the API. Enter the below url on browser to see the message.

localhost:8080/demo/data
Our message will be reflected.



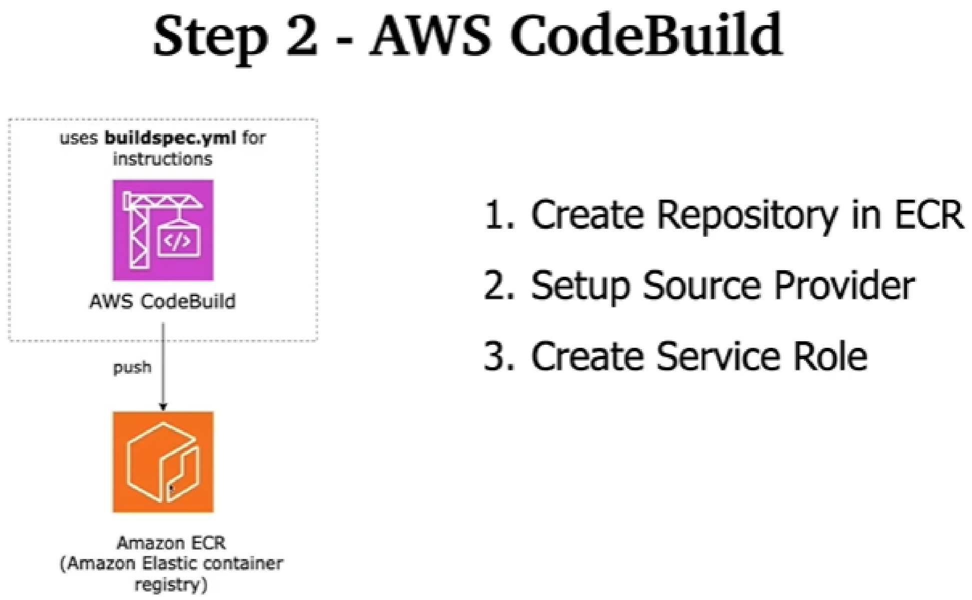
First message from AWS ECS

To check Health.
localhost:8080/actuator/health



```
{"status":"UP"}
```

Step 2: AWS CodeBuild.


uses **buildspec.yml** for instructions
AWS CodeBuild
push
Amazon ECR
(Amazon Elastic container registry)

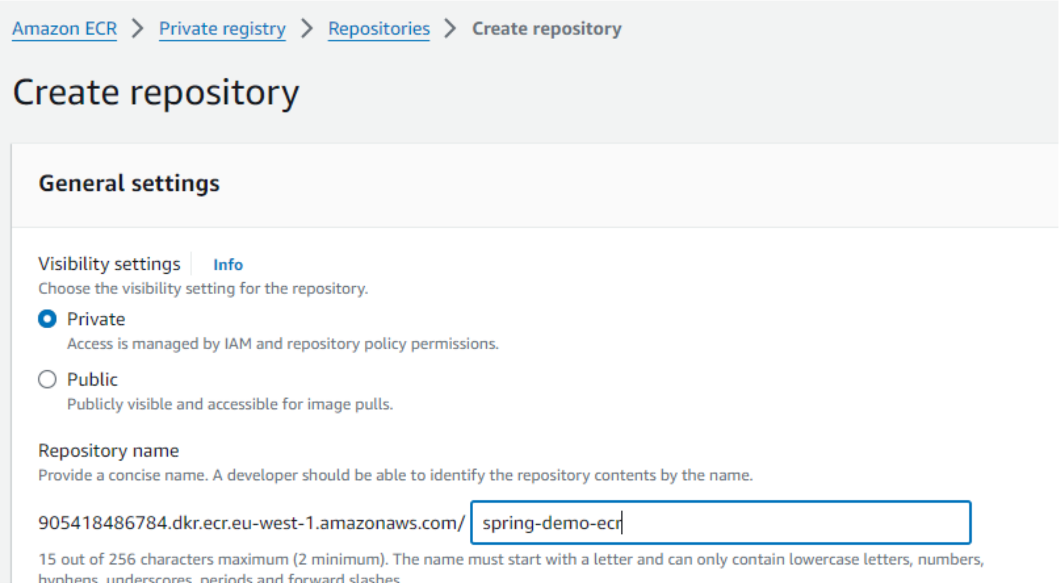
Step 2 - AWS CodeBuild

1. Create Repository in ECR
2. Setup Source Provider
3. Create Service Role

- In step 2 we will be configuring AWS code build here, for that we will first create a repository into Amazon ECR i.e **'Elastic Container Registry'**.
- After that, we will configure the code build in which we will set up our source provided as a GitHub.
- Then we will add a **buildspec.yaml** file into our application which AWS codebuild will use for the project build.
- At the end we will create a service role for the codebuild because codebuild will require permission to push Docker image into Amazon ECR.

- Now, go to AWS console > ECR > Create New repository.

Name: spring-demo-ecr



Amazon ECR > Private registry > Repositories > Create repository

Create repository

General settings

Visibility settings Info
Choose the visibility setting for the repository.

● Private
Access is managed by IAM and repository policy permissions.

○ Public
Publicly visible and accessible for image pulls.

Repository name
Provide a concise name. A developer should be able to identify the repository contents by the name.

905418486784.dkr.ecr.eu-west-1.amazonaws.com/ spring-demo-ecr

15 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.

- Create **buildspec.yml** file.

```
version:0.2
phases:
pre_build:
commands:
-echoLogginginto AmazonECR....
-aws--version
#-$(awsecrget-login--regionap-south-1--no-include-email)
-awsecrget-login-password--regioneu-west-1|dockerlogin--usernameAWS--password-
stdin905418486784.dkr.ecr.eu-west-1.amazonaws.com                --#Replace this
command from the ECR repository push command. You will get it from AWS
console.
-#ReplacewiththistoyourrepositoryURI
-REPOSITORY_URI=905418486784.dkr.ecr.eu-west-1.amazonaws.com/spring-demo-ecr
--#Get URI from the AWS ECR you created.
-IMAGE_TAG=build-$(echo$CODEBUILD_BUILD_ID|awk-F":"'{print$2}')
build:
commands:
-echoBuildstartedon`date`
-echobuildingtheJarfile
-mvncleaninstall
-echoBuildingtheDockerimage...
-dockerbuild-t$REPOSITORY_URI:latest.
-dockertag$REPOSITORY_URI:latest$REPOSITORY_URI:$IMAGE_TAG
post_build:
commands:
-echoBuildcompletedon`date`
-echopushingtorepo
-dockerpush$REPOSITORY_URI:latest
```

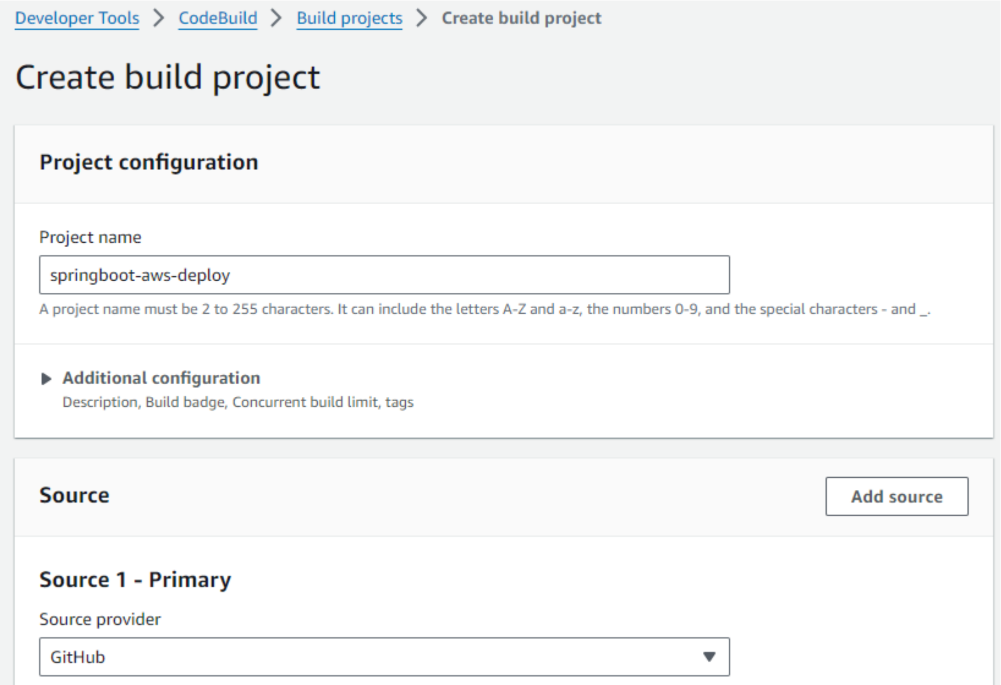
```
-dockerpush$REPOSITORY_URI:$IMAGE_TAG
-echoWritingimagedefinitionsfile...
#Giveyourcontainername
-DOCKER_CONTAINER_NAME=spring-demo-ecr
-
printf'[{"name":"%s","imageUri":"%s"}]'$DOCKER_CONTAINER_NAME$REPOSITORY_URI
:$IMAGE_TAG>imagedefinitions.json
-echo$DOCKER_CONTAINER_NAME
-echoprintingimagedefinitions.json
-catimagedefinitions.json
artifacts:
files:
-imagedefinitions.json
-target/springboot-aws-deploy.jar
```

- Now, push the changes in the git hub.

```
git add .
git commit -m "New commit"
git push -u origin main
```

- Now search for **CodeBuild** in AWS console > **Create Project**.

- **Project Name:** springboot-aws-deploy
- **Source provider:** GitHub
- **Repository:** Connect with a GitHub personal access token
- **GitHub personal access token: --#Generate the Personal Access Token from the GitHub and Save Token.**
- **GitHub repository**: https://github.com/amanduggal001/springboot-aws-deploy2.git
- **Operating system:** Amazon Linux
- **Runtime(s):** Standard
- **Image:** ...Standard:5.0
- **Image version:** Always use the latest image
- **Service role:** New service role
- **Role name:** codebuild-springboot-aws-deploy-service-role **(Auto-populate)**
- **Privileged:** **Check mark** (Enable this flag if you want to build Docker images or want your builds to get elevated privileges)
- **Build specifications:** Use a buildspec file
- **CloudWatch:** Enable CloudWatch logs
- **Group name:** Spring-demo



Developer Tools > CodeBuild > Build projects > Create build project

Create build project

Project configuration

Project name: springboot-aws-deploy

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 _.

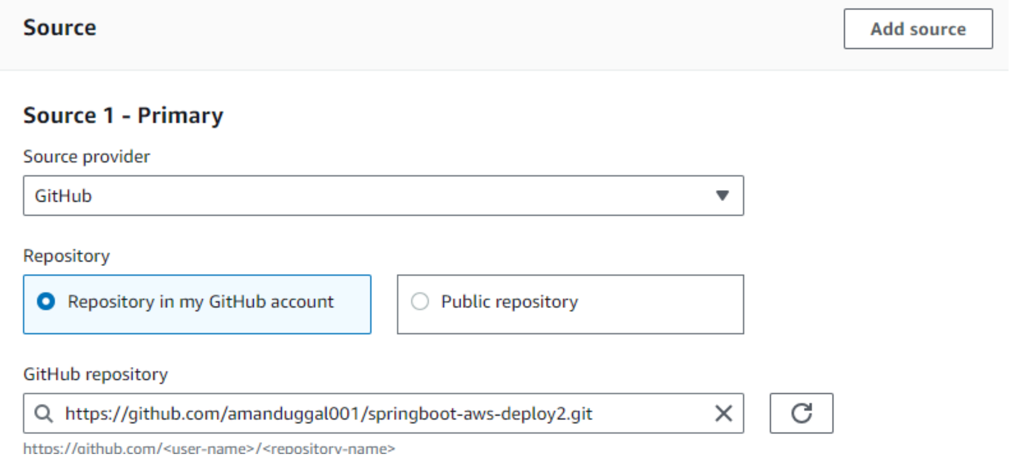
▶ Additional configuration

Description, Build badge, Concurrent build limit, tags

Source

Source 1 - Primary

Source provider: GitHub



Source

Source 1 - Primary

Source provider: GitHub

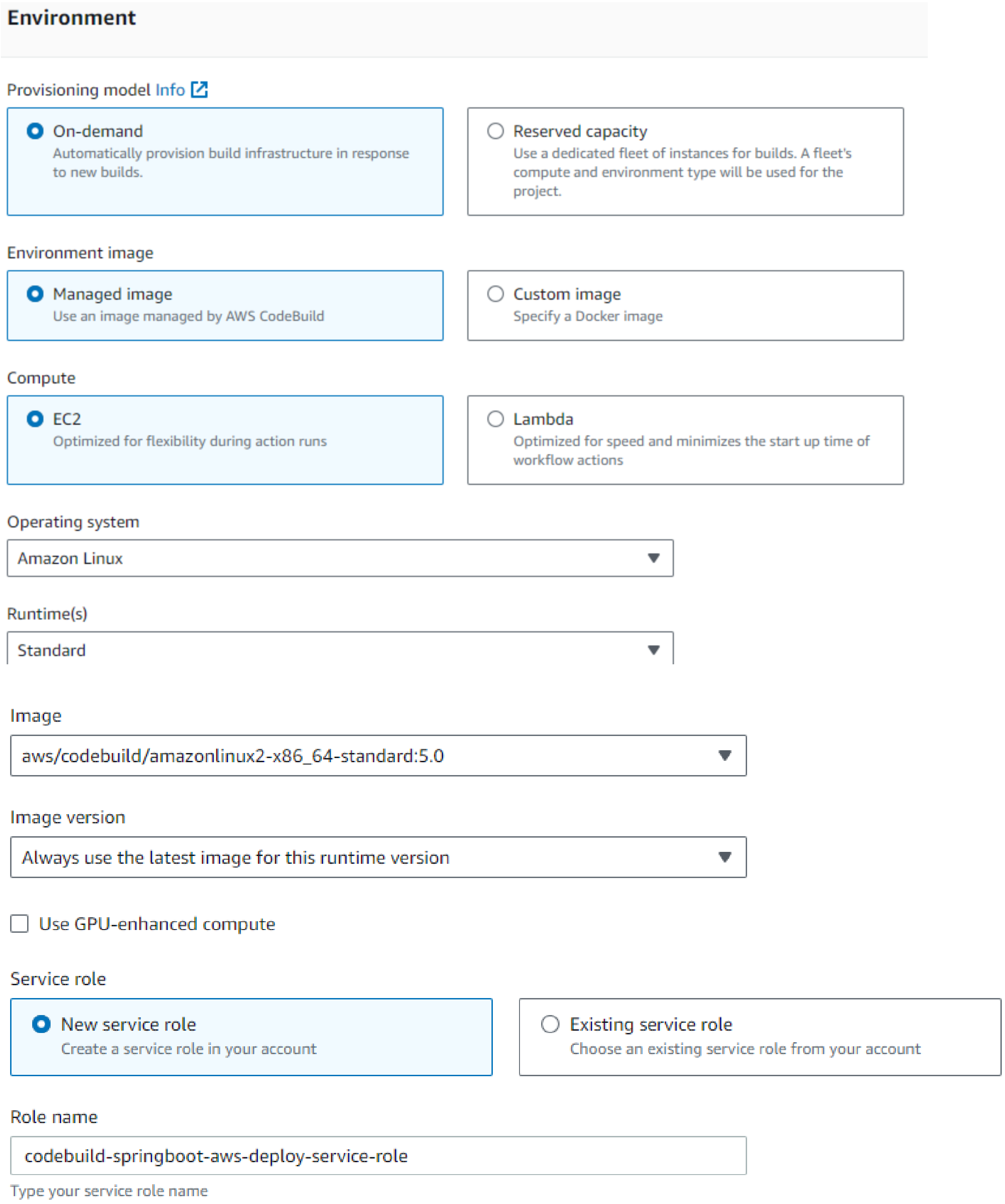
Repository

● Repository in my GitHub account

○ Public repository

GitHub repository: https://github.com/amanduggal001/springboot-aws-deploy2.git

https://github.com/<user-name>/<repository-name>



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

Operating system

Amazon Linux ▼

Runtime(s)

Standard ▼

Image

aws/codebuild/amazonlinux2-x86_64-standard:5.0 ▼

Image version

Always use the latest image for this runtime version ▼

□ Use GPU-enhanced compute

Service role

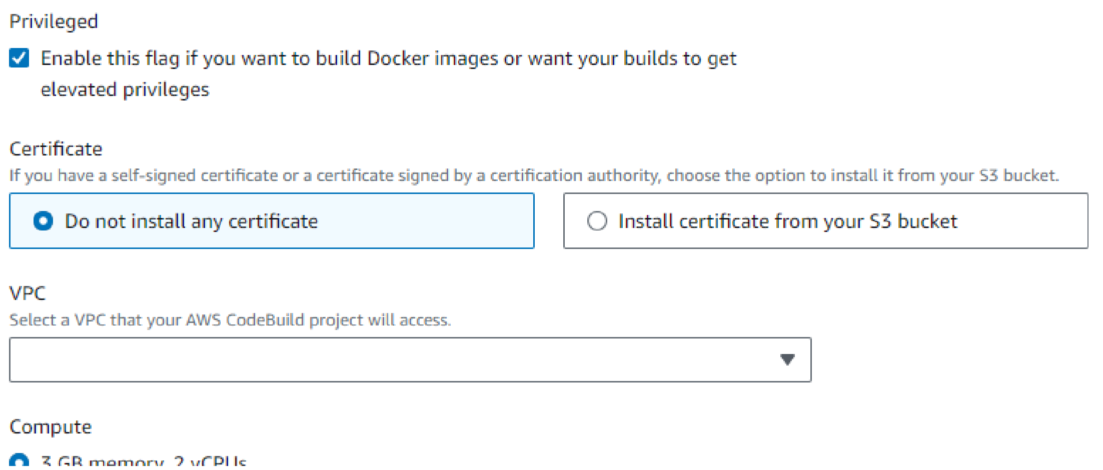
● New service role
Create a service role in your account

○ Existing service role
Choose an existing service role from your account

Role name

codebuild-springboot-aws-deploy-service-role

Type your service role name



Privileged

☑ Enable this flag if you want to build Docker images or want your builds to get elevated privileges

Certificate

If you have a self-signed certificate or a certificate signed by a certification authority, choose the option to install it from your S3 bucket.

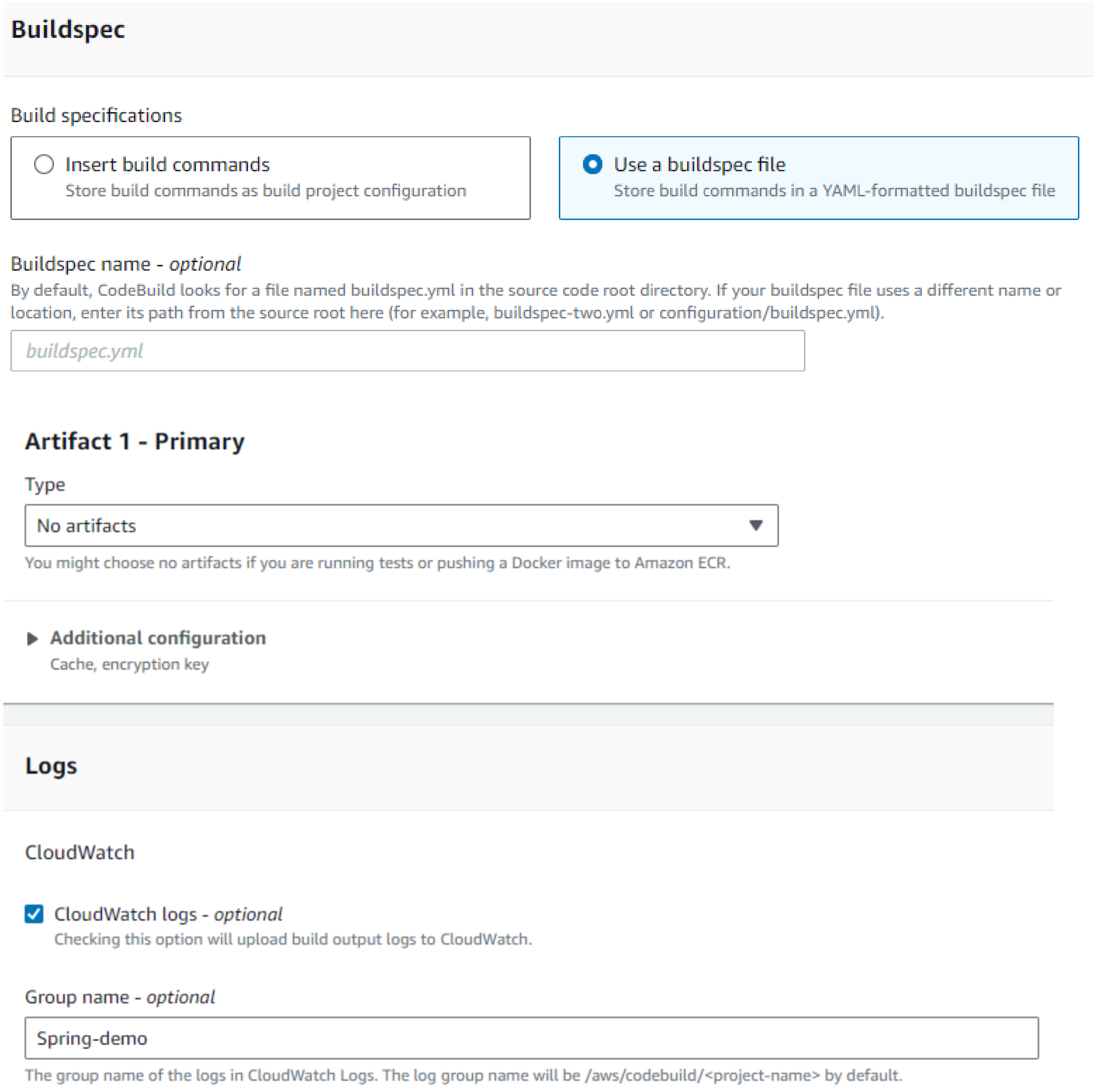
● Do not install any certificate   ○ Install certificate from your S3 bucket

VPC

Select a VPC that your AWS CodeBuild project will access.

Compute

● 3 GB memory, 2 vCPUs



Buildspec

Build specifications

○ Insert build commands
Store build commands as build project configuration

● Use a buildspec file
Store build commands in a YAML-formatted buildspec file

Buildspec name - *optional*

By default, CodeBuild looks for a file named buildspec.yml in the source code root directory. If your buildspec file uses a different name or location, enter its path from the source root here (for example, buildspec-two.yml or configuration/buildspec.yml).

buildspec.yml

Artifact 1 - Primary

Type

No artifacts

You might choose no artifacts if you are running tests or pushing a Docker image to Amazon ECR.

▶ **Additional configuration**
Cache, encryption key

Logs

CloudWatch

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

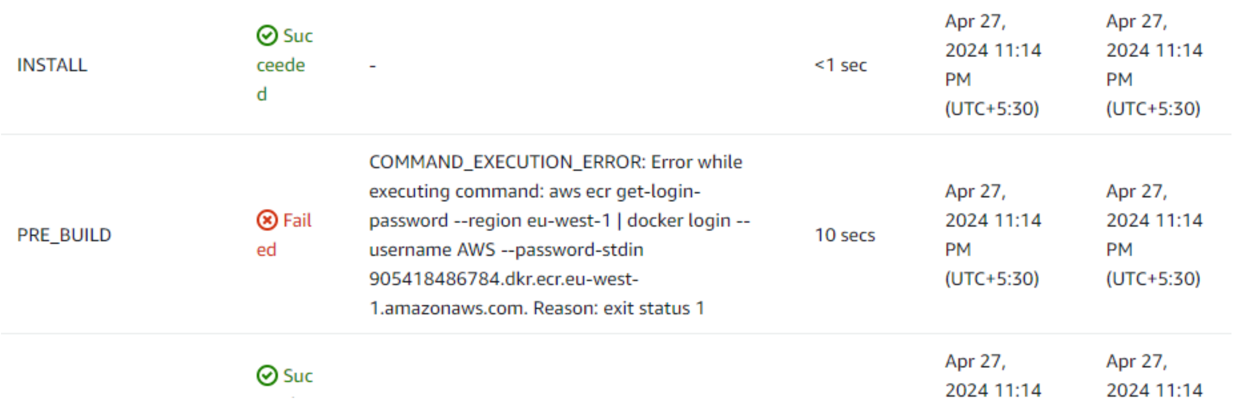
Group name - *optional*

Spring-demo

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

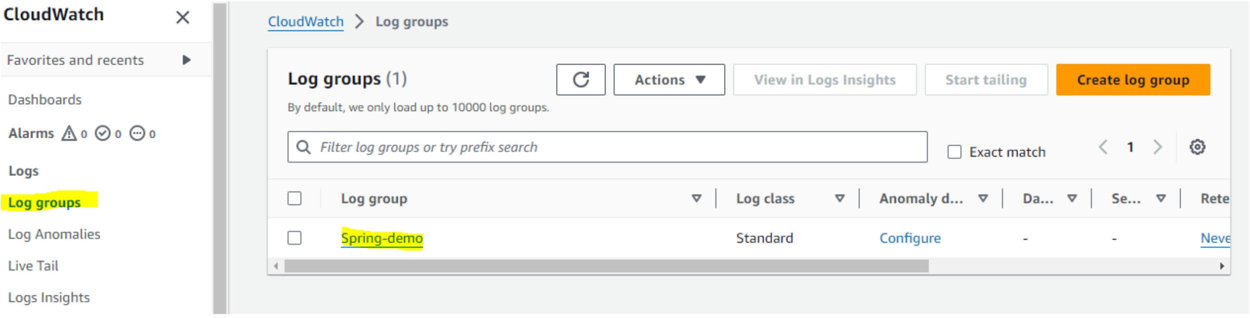
- Now, once the project is created, start the build manually.

After executing we see that the build has failed since the **CodeBuild** is trying to log into AWS ECR and the code build doesn't have permission to log into ECR.

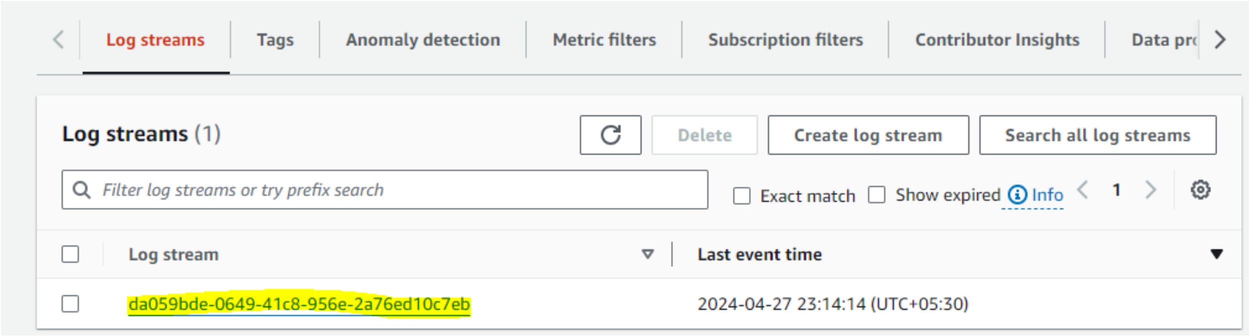


INSTALL	✓ Succeeded	-	<1 sec	Apr 27, 2024 11:14 PM (UTC+5:30)	Apr 27, 2024 11:14 PM (UTC+5:30)
PRE_BUILD	⊗ Failed	COMMAND_EXECUTION_ERROR: Error while executing command: aws ecr get-login-password --region eu-west-1 | docker login --username AWS --password-stdin 905418486784.dkr.ecr.eu-west-1.amazonaws.com. Reason: exit status 1	10 secs	Apr 27, 2024 11:14 PM (UTC+5:30)	Apr 27, 2024 11:14 PM (UTC+5:30)
	✓ Suc			Apr 27, 2024 11:14	Apr 27, 2024 11:14

We can view the logs from the **CloudWatch** also**.**



□	Log group	Log class	Anomaly d...	Da...	Se...	Rete
□	Spring-demo	Standard	Configure	-	-	Neve



□	Log stream	Last event time
□	da059bde-0649-41c8-956e-2a76ed10c7eb	2024-04-27 23:14:14 (UTC+05:30)

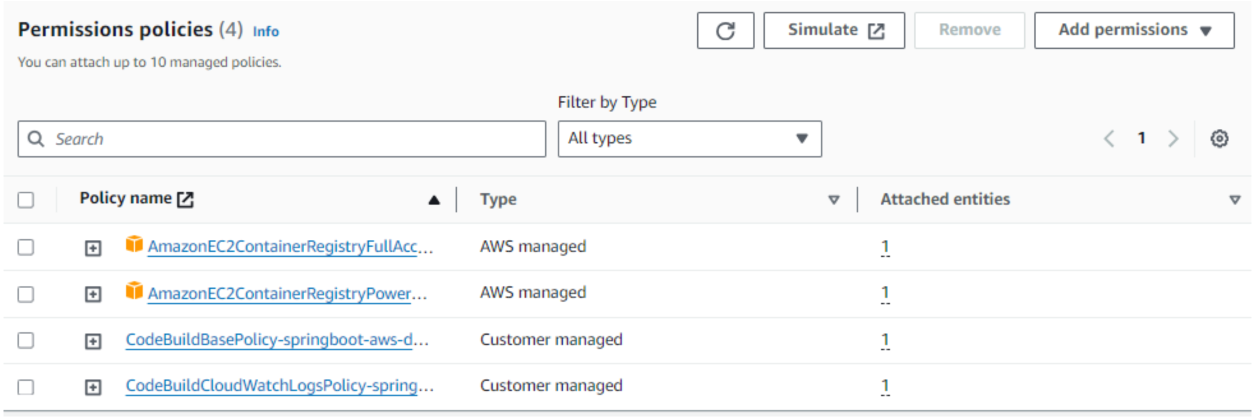
- Now, we provide the permissions to access the ECR to our CodeBuild role i.e, **'codebuild-springboot-aws-deploy-service-role'**

IAM > Roles > Search for Codebuild role name > Add permissions> Attach Policy.

Permissions:

- **AmazonEC2ContainerRegistryFullAccess**

- **AmazonEC2ContainerRegistryPowerUser**

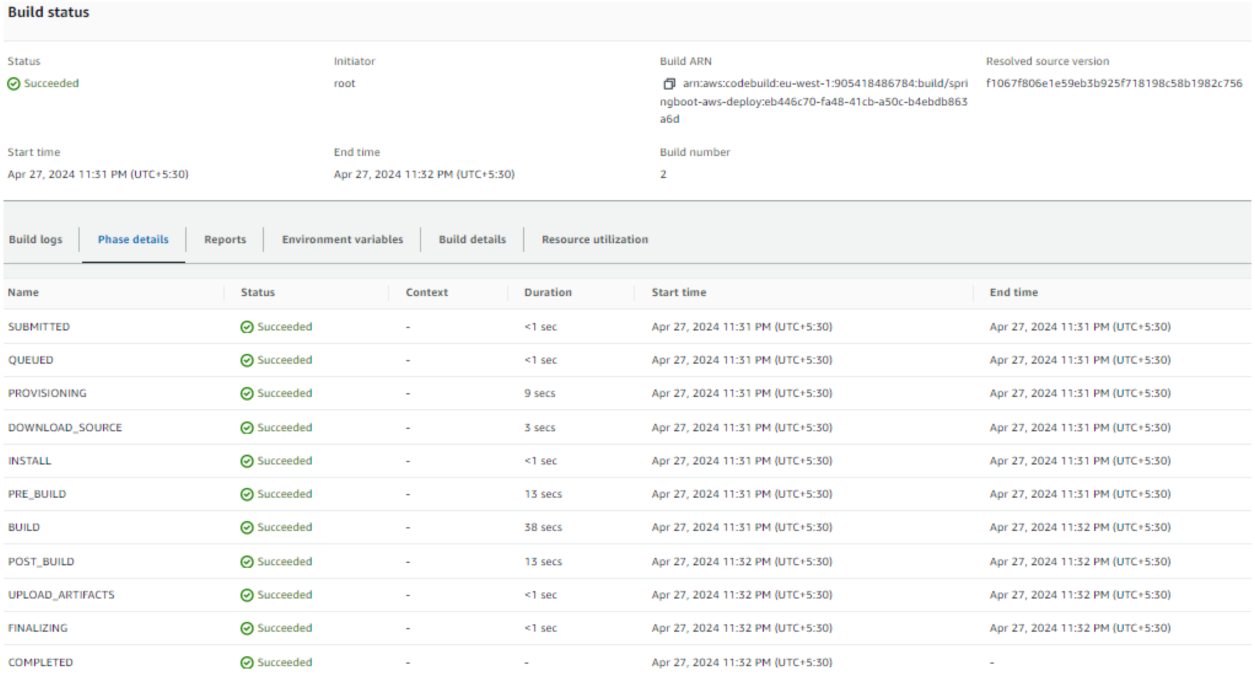


Permissions policies (4) Info

You can attach up to 10 managed policies.

□	Policy name	Type	Attached entities
□	⊞ AmazonEC2ContainerRegistryFullAcc...	AWS managed	1
□	⊞ AmazonEC2ContainerRegistryPower...	AWS managed	1
□	⊞ CodeBuildBasePolicy-springboot-aws-d...	Customer managed	1
□	⊞ CodeBuildCloudWatchLogsPolicy-spring...	Customer managed	1

- Once the permissions are added, Retry the Build.



Build status

Status: ✓ Succeeded

Initiator: root

Build ARN: arn:aws:codebuild:eu-west-1:905418486784:build/springboot-aws-deploy:eb446c70-fa48-41cb-a50c-b4ebdb863a6d

Resolved source version: f1067f806e1e59eb3b925f718198c58b1982c756

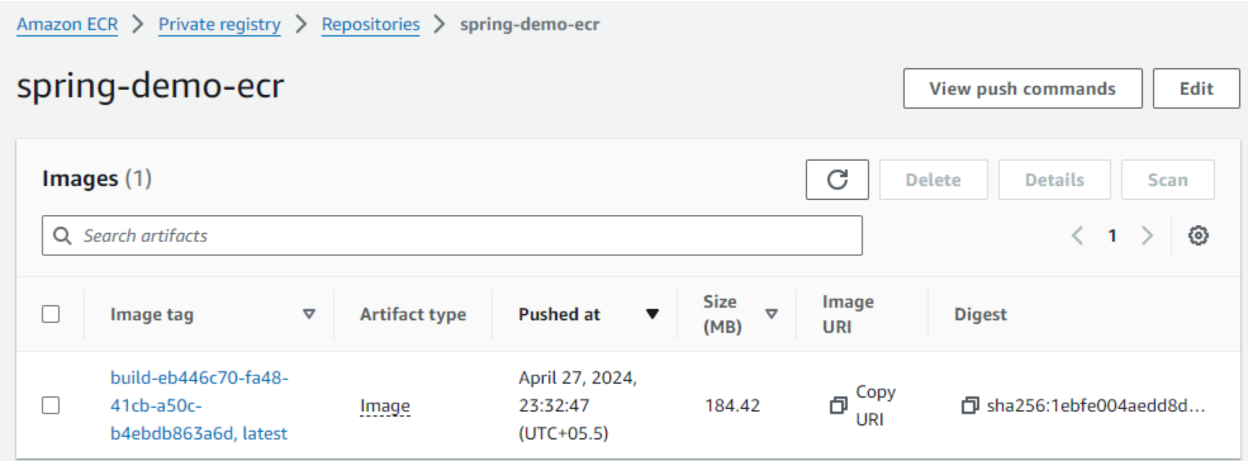
Start time: Apr 27, 2024 11:31 PM (UTC+5:30)

End time: Apr 27, 2024 11:32 PM (UTC+5:30)

Build number: 2

Name	Status	Context	Duration	Start time	End time
SUBMITTED	✓ Succeeded	-	<1 sec	Apr 27, 2024 11:31 PM (UTC+5:30)	Apr 27, 2024 11:31 PM (UTC+5:30)
QUEUED	✓ Succeeded	-	<1 sec	Apr 27, 2024 11:31 PM (UTC+5:30)	Apr 27, 2024 11:31 PM (UTC+5:30)
PROVISIONING	✓ Succeeded	-	9 secs	Apr 27, 2024 11:31 PM (UTC+5:30)	Apr 27, 2024 11:31 PM (UTC+5:30)
DOWNLOAD_SOURCE	✓ Succeeded	-	3 secs	Apr 27, 2024 11:31 PM (UTC+5:30)	Apr 27, 2024 11:31 PM (UTC+5:30)
INSTALL	✓ Succeeded	-	<1 sec	Apr 27, 2024 11:31 PM (UTC+5:30)	Apr 27, 2024 11:31 PM (UTC+5:30)
PRE_BUILD	✓ Succeeded	-	13 secs	Apr 27, 2024 11:31 PM (UTC+5:30)	Apr 27, 2024 11:31 PM (UTC+5:30)
BUILD	✓ Succeeded	-	38 secs	Apr 27, 2024 11:31 PM (UTC+5:30)	Apr 27, 2024 11:32 PM (UTC+5:30)
POST_BUILD	✓ Succeeded	-	13 secs	Apr 27, 2024 11:32 PM (UTC+5:30)	Apr 27, 2024 11:32 PM (UTC+5:30)
UPLOAD_ARTIFACTS	✓ Succeeded	-	<1 sec	Apr 27, 2024 11:32 PM (UTC+5:30)	Apr 27, 2024 11:32 PM (UTC+5:30)
FINALIZING	✓ Succeeded	-	<1 sec	Apr 27, 2024 11:32 PM (UTC+5:30)	Apr 27, 2024 11:32 PM (UTC+5:30)
COMPLETED	✓ Succeeded	-	-	Apr 27, 2024 11:32 PM (UTC+5:30)	-

- Now, check the ECR Repository inside the Repository our image created by CodeBuild, so this is how we have completed step 2.



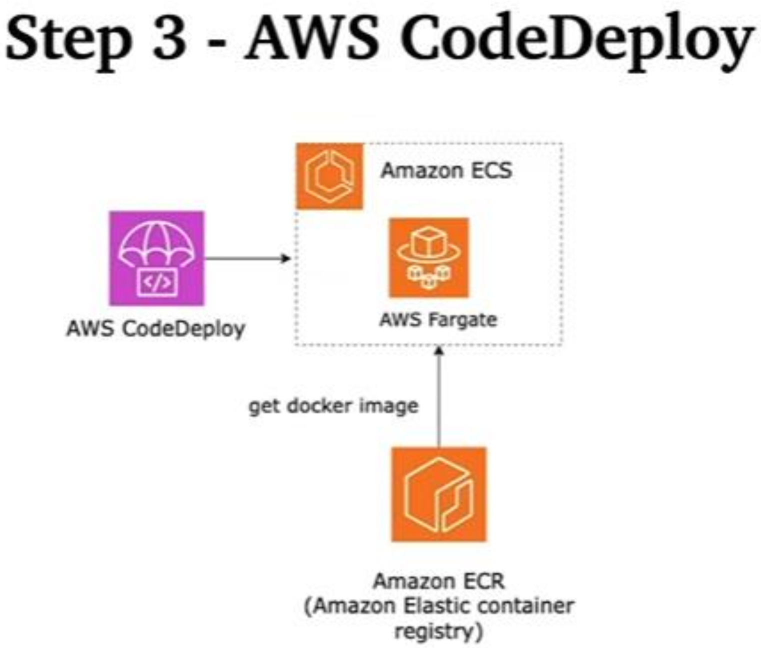
Amazon ECR > Private registry > Repositories > spring-demo-ecr

spring-demo-ecr

Images (1)

□	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
□	build-eb446c70-fa48-41cb-a50c-b4ebdb863a6d, latest	Image	April 27, 2024, 23:32:47 (UTC+05.5)	184.42	Copy URI	sha256:1ebfe004aedd8d…

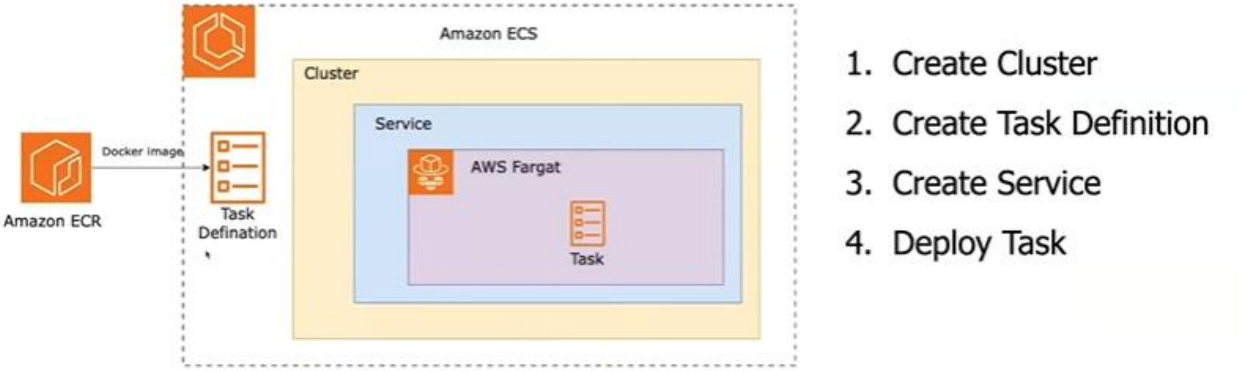
Step 3 - AWS CodeDeploy



Step 3 - AWS CodeDeploy

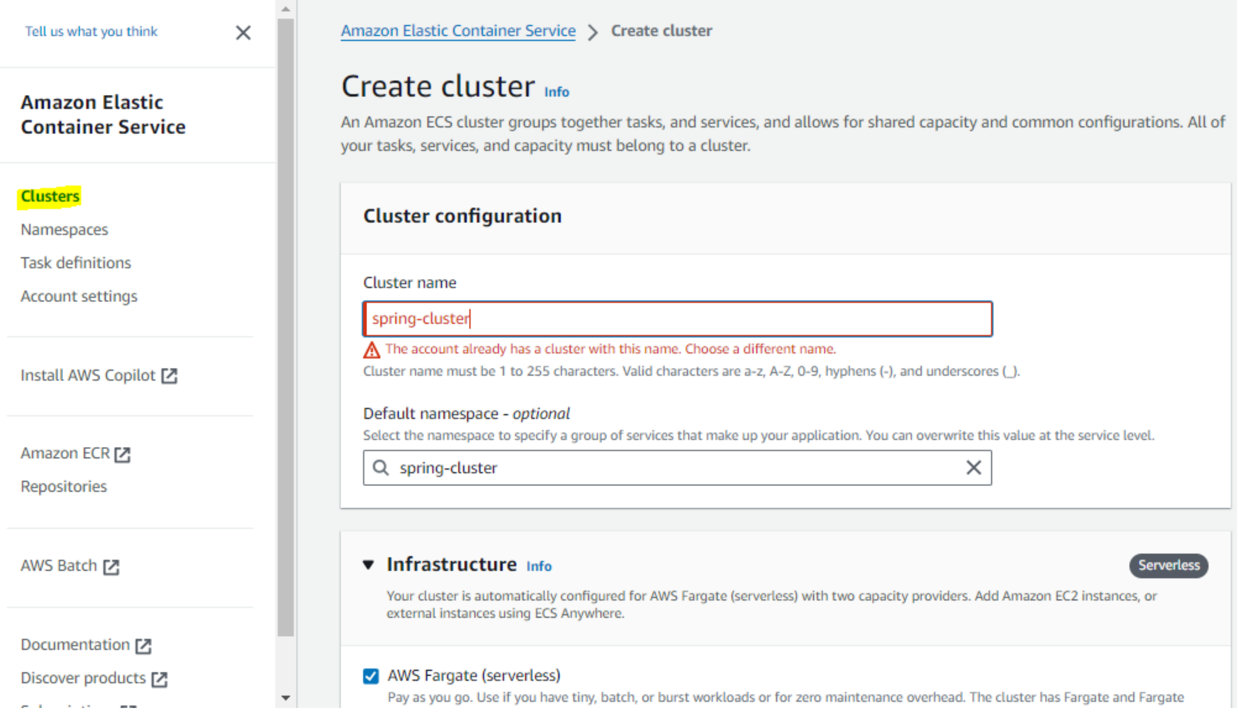
In step 3 we will set **AWS CodeDeploy** in which we will configure code deploy.

If we go deep inside, we are going to create a first cluster. In the cluster we are going to create a service then we are going to create a task definition. Basically, task destination is a kind of a template, and this template is going to be used by AWS Fargate to deploy our application.



1. Create Cluster
2. Create Task Definition
3. Create Service
4. Deploy Task

- Go To **ECS** in **AWS console > Create Cluster.**
  - **Cluster name:** Spring-cluster
  - **Infrastructure:** AWS Fargate (serverless)



Create cluster

An Amazon ECS cluster groups together tasks, and services, and allows for shared capacity and common configurations. All of your tasks, services, and capacity must belong to a cluster.

Cluster configuration

Cluster name

spring-cluster

The account already has a cluster with this name. Choose a different name.

Cluster name must be 1 to 255 characters. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Default namespace - *optional*

Select the namespace to specify a group of services that make up your application. You can overwrite this value at the service level.

spring-cluster

Infrastructure Info Serverless

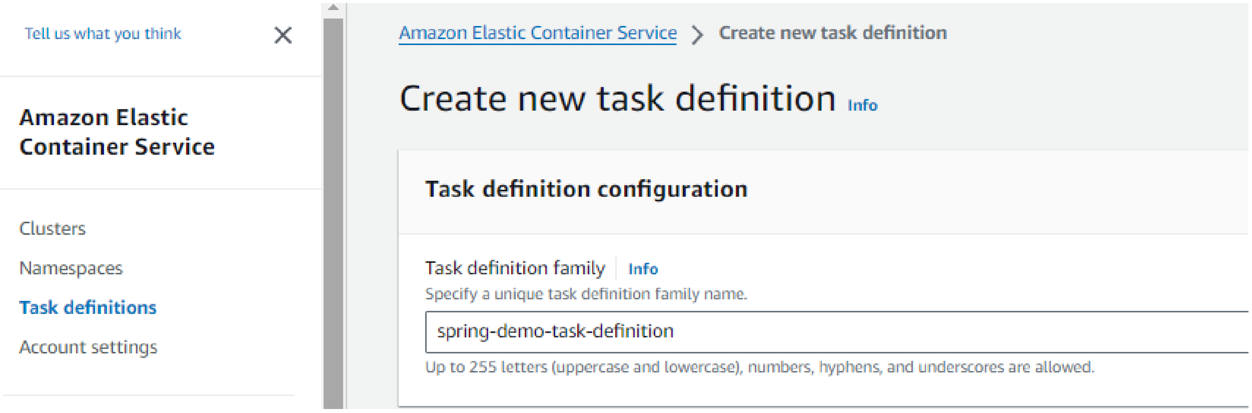
Your cluster is automatically configured for AWS Fargate (serverless) with two capacity providers. Add Amazon EC2 instances, or external instances using ECS Anywhere.

☑ AWS Fargate (serverless)

Pay as you go. Use if you have tiny, batch, or burst workloads or for zero maintenance overhead. The cluster has Fargate and Fargate

- Now, once the cluster is created, create the Task Definition.
  - **Task definition family:** spring-demo-task-definition
  - **Launch type:** AWS Fargate
  - **Container Name:** spring-demo-ecr
  - **Image URI: --#Get from the ECR.**
  - Container port: 8080
  - Protocol: TCP
  - Port Name: spring-demo-ecr-8080-tcp
  - App protocol: HTTP

- Health Check Command: CMD-SHELL,curl -f http://localhost:8080/actuator/health || exit 1



Amazon Elastic Container Service

Clusters

Namespaces

Task definitions

Account settings

Amazon Elastic Container Service > Create new task definition

Create new task definition Info

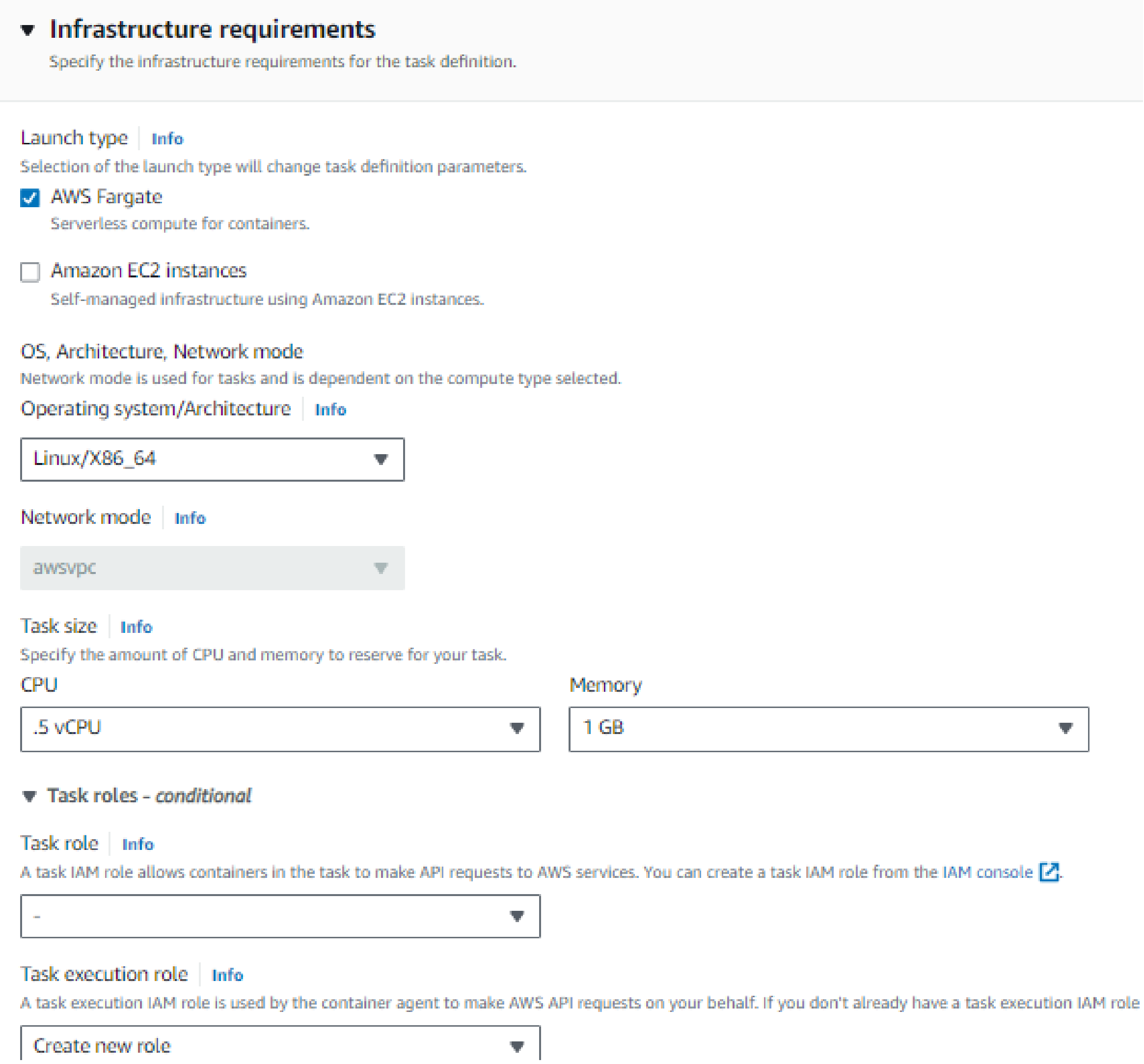
Task definition configuration

Task definition family | Info

Specify a unique task definition family name.

spring-demo-task-definition

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

Specify the amount of CPU and memory to reserve for your task.

CPU	Memory
.5 vCPU ▼	1 GB ▼

▼ Task roles - *conditional*

Task role | Info

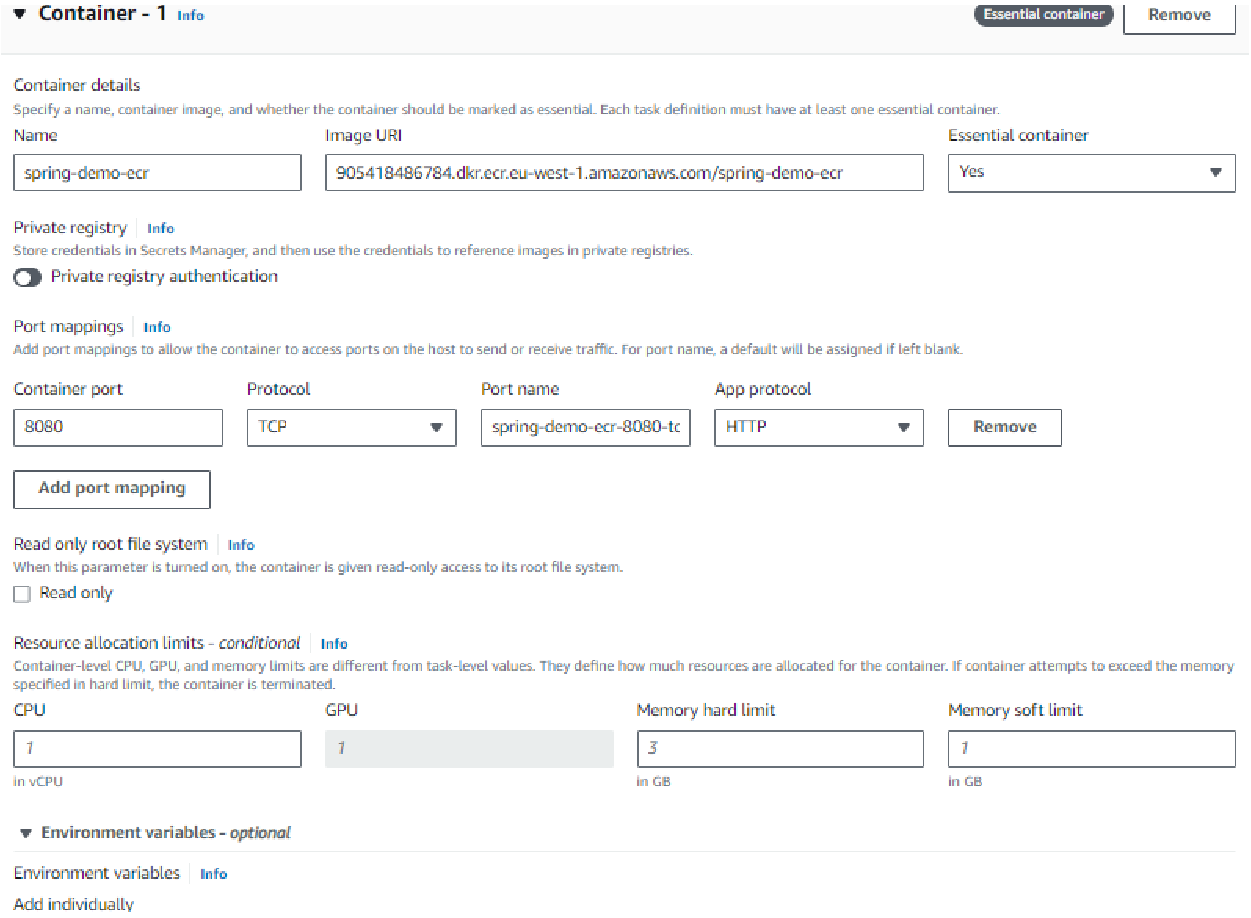
A task IAM role allows containers in the task to make API requests to AWS services. You can create a task IAM role from the IAM console.

- ▼

Task execution role | Info

A task execution IAM role is used by the container agent to make AWS API requests on your behalf. If you don't already have a task execution IAM role

Create new role ▼



▼ **Container - 1** Info

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	Image URI	Essential container
spring-demo-ecr	905418486784.dkr.ecr.eu-west-1.amazonaws.com/spring-demo-ecr	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	Protocol	Port name	App protocol	
8080	TCP ▼	spring-demo-ecr-8080-tc	HTTP ▼	Remove

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

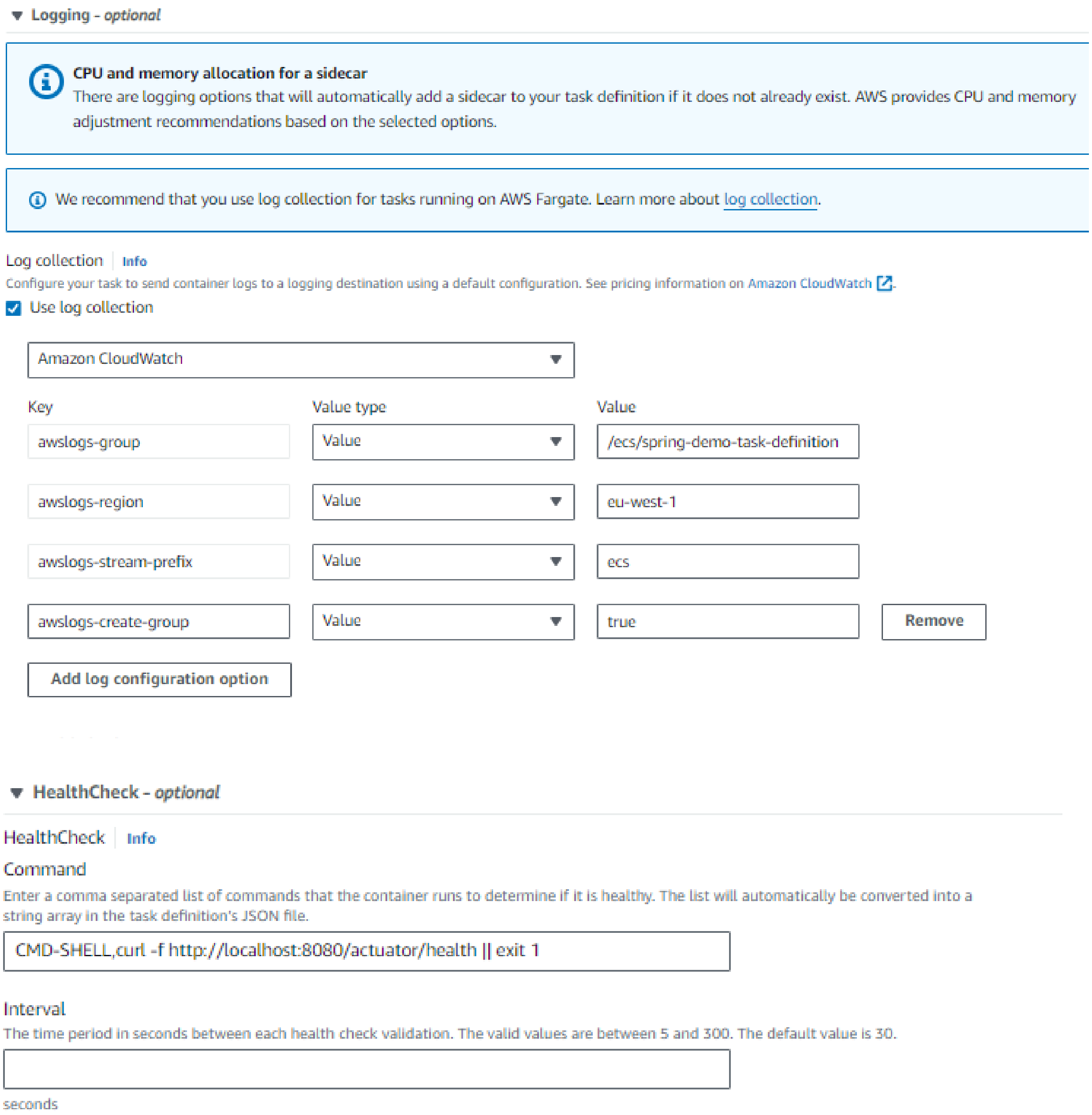
Container-level CPU, GPU, and memory limits are different from task-level values. They define how much resources are allocated for the container. If container attempts to exceed the memory specified in hard limit, the container is terminated.

CPU	GPU	Memory hard limit	Memory soft limit
1	*1*	*3*	*1*
in vCPU		in GB	in GB

▼ **Environment variables - *optional***

Environment variables | Info

Add individually



▼ Logging - *optional*

CPU and memory allocation for a sidecar
There are logging options that will automatically add a sidecar to your task definition if it does not already exist. AWS provides CPU and memory adjustment recommendations based on the selected options.

We recommend that you use log collection for tasks running on AWS Fargate. Learn more about log collection.

Log collection | Info
Configure your task to send container logs to a logging destination using a default configuration. See pricing information on Amazon CloudWatch.
☑ Use log collection

Amazon CloudWatch ▼

Key	Value type	Value
awslogs-group	Value	/ecs/spring-demo-task-definition
awslogs-region	Value	eu-west-1
awslogs-stream-prefix	Value	ecs
awslogs-create-group	Value	true

▼ HealthCheck - *optional*

HealthCheck | Info

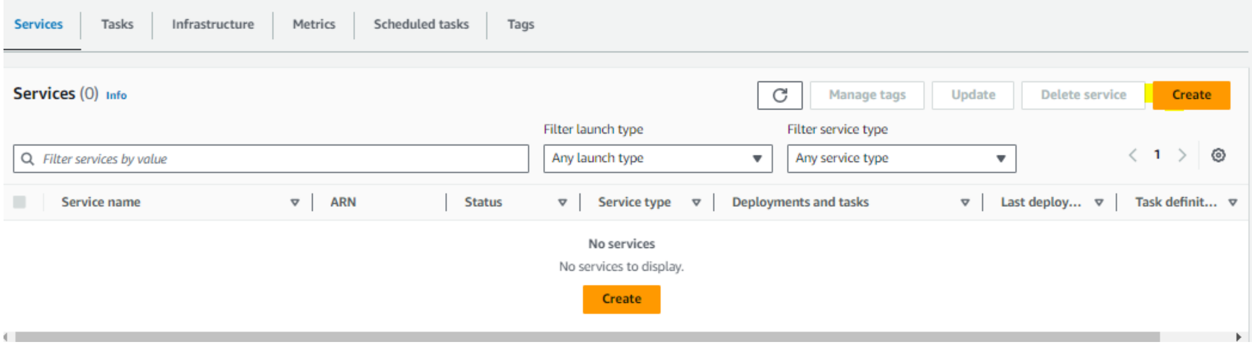
Command
Enter a comma separated list of commands that the container runs to determine if it is healthy. The list will automatically be converted into a string array in the task definition's JSON file.

```
CMD-SHELL,curl -f http://localhost:8080/actuator/health || exit 1
```

Interval
The time period in seconds between each health check validation. The valid values are between 5 and 300. The default value is 30.

seconds

- Once the Task definition is created, create the **Service** in **Cluster**.



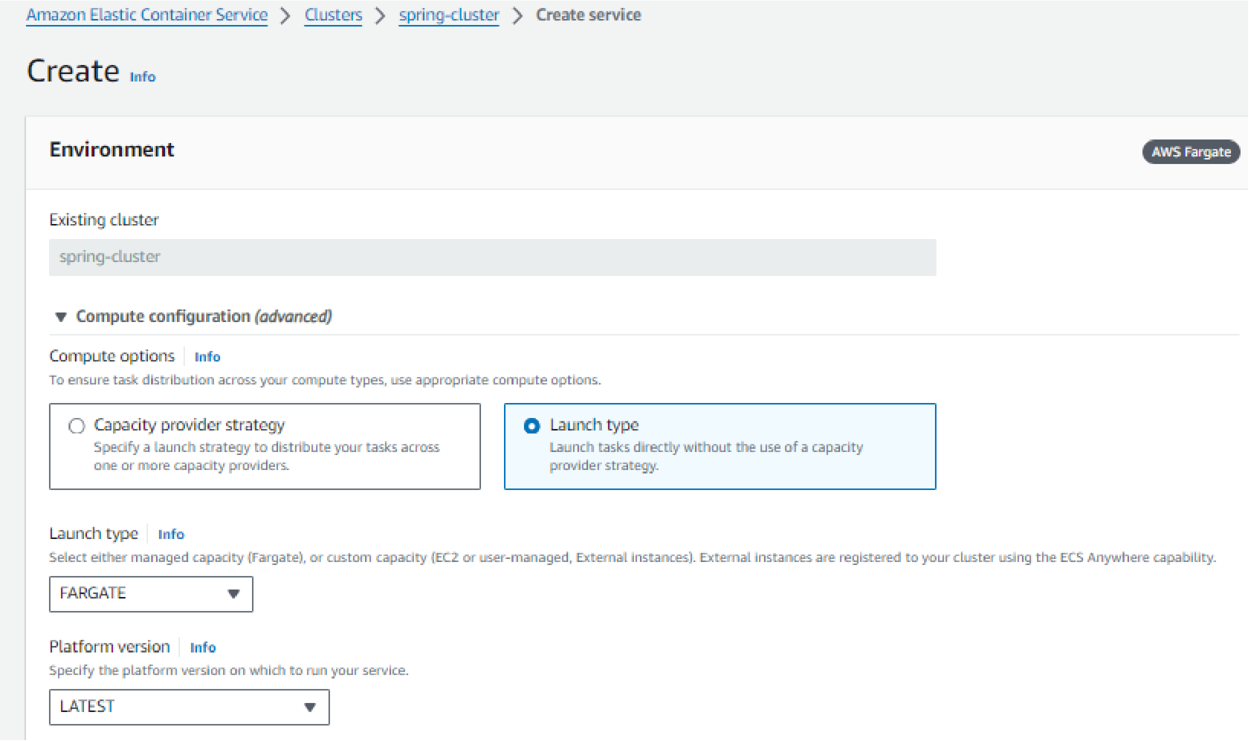
Services (0) Info

Filter services by value
Filter launch type: Any launch type
Filter service type: Any service type

Service name	ARN	Status	Service type	Deployments and tasks	Last deploy...	Task definit...

No services
No services to display.

- **Compute options:** Launch type
- **Application type:** Service
- **Family:** **--#Select the Task definition which we have created.**
- **Service name:** spring-demo-service
- **Security Group:** Create New Security Group > Add port 8080.



Amazon Elastic Container Service > Clusters > spring-cluster > Create service

Create Info

Environment

AWS Fargate

Existing cluster

spring-cluster

▼ 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

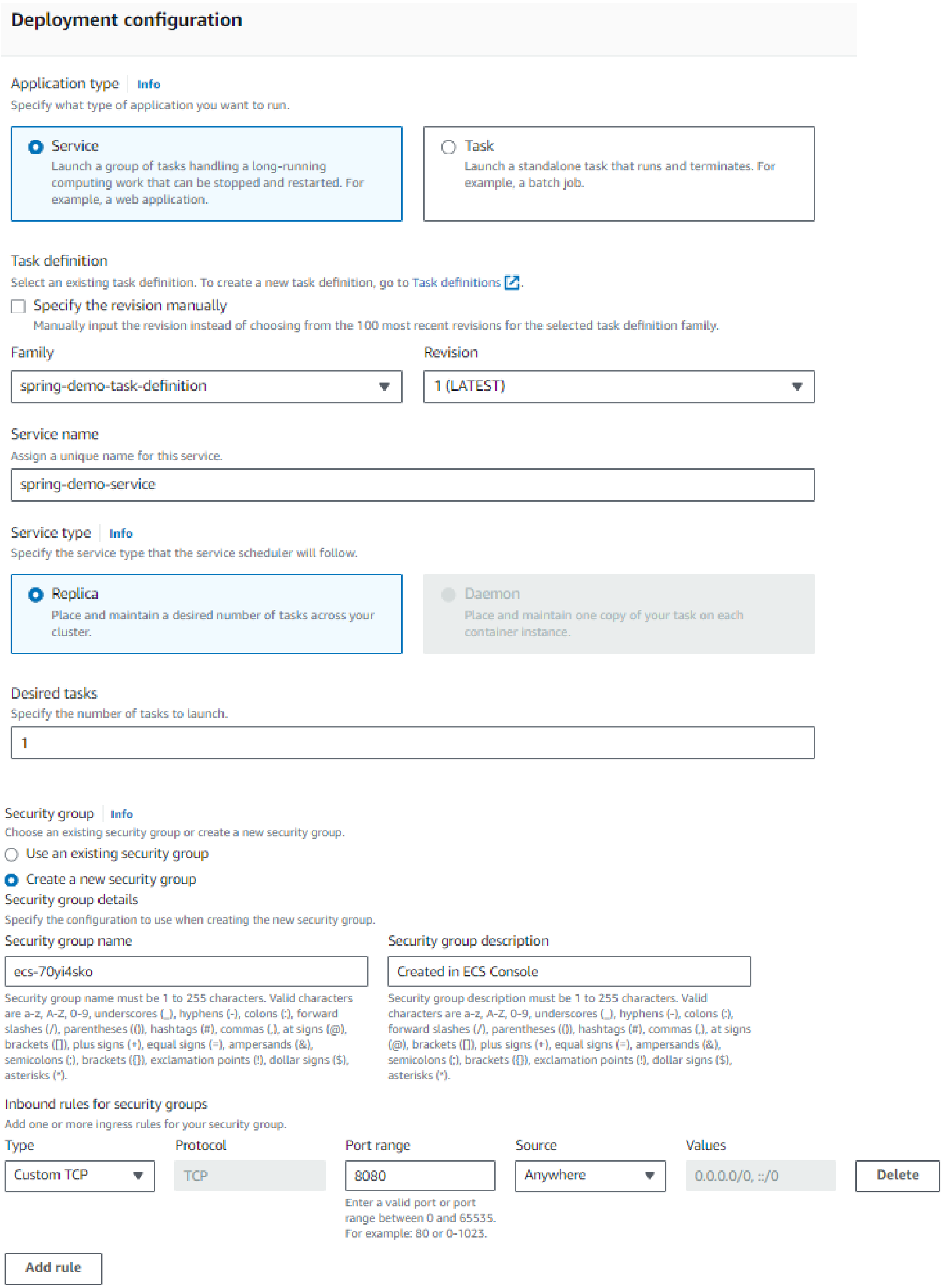
Select either managed capacity (Fargate), or custom capacity (EC2 or user-managed, External instances). External instances are registered to your cluster using the ECS Anywhere capability.

FARGATE ▼

Platform version | Info

Specify the platform version on which to run your service.

LATEST ▼



Deployment configuration

Application type | Info

Specify what type of application you want to run.

● Service
Launch a group of tasks handling a long-running computing work that can be stopped and restarted. For example, a web application.

○ Task
Launch a standalone task that runs and terminates. For example, a batch job.

Task definition

Select an existing task definition. To create a new task definition, go to Task definitions.

□ Specify the revision manually
Manually input the revision instead of choosing from the 100 most recent revisions for the selected task definition family.

Family	Revision
spring-demo-task-definition ▼	1 (LATEST) ▼

Service name

Assign a unique name for this service.

spring-demo-service

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

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 details

Specify the configuration to use when creating the new security group.

Security group name

ecs-70yi4sko

Security group name must be 1 to 255 characters. Valid characters are a-z, A-Z, 0-9, underscores (_), hyphens (-), colons (:), forward slashes (/), parentheses (()), hashtags (#), commas (,), at signs (@), brackets ([]), plus signs (+), equal signs (=), ampersands (&), semicolons (;), brackets ({}), exclamation points (!), dollar signs ($), asterisks (*).

Security group description

Created in ECS Console

Security group description must be 1 to 255 characters. Valid characters are a-z, A-Z, 0-9, underscores (_), hyphens (-), colons (:), forward slashes (/), parentheses (()), hashtags (#), commas (,), at signs (@), brackets ([]), plus signs (+), equal signs (=), ampersands (&), semicolons (;), brackets ({}), exclamation points (!), dollar signs ($), asterisks (*).

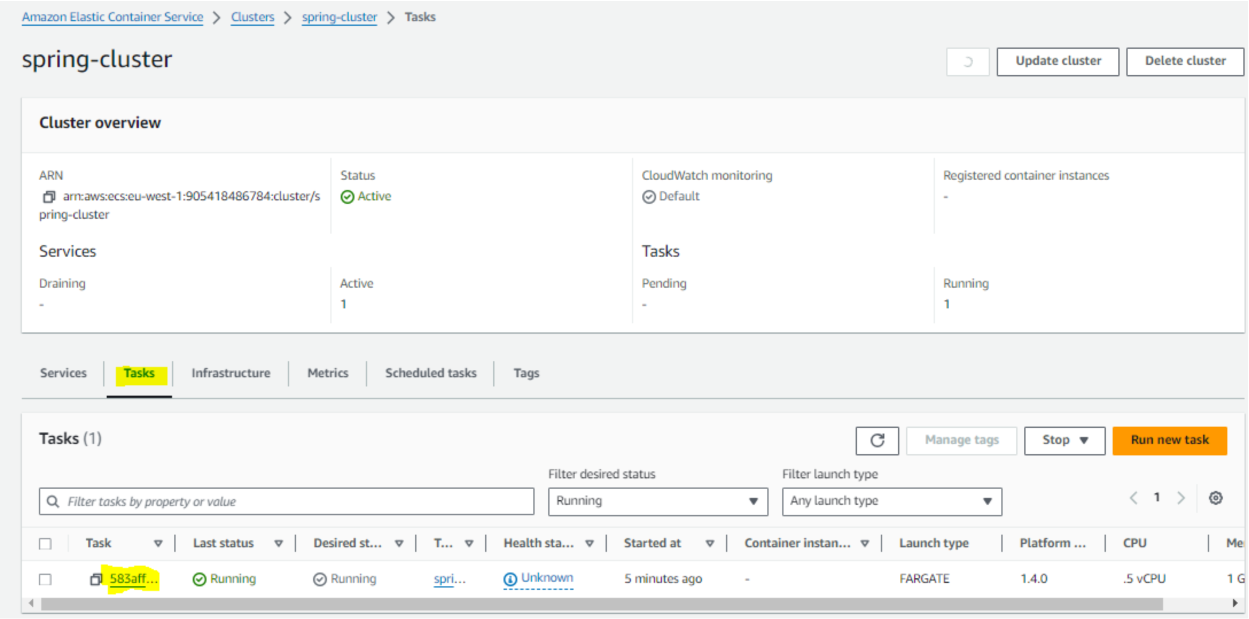
Inbound rules for security groups

Add one or more ingress rules for your security group.

Type	Protocol	Port range	Source	Values	
Custom TCP ▼	TCP	8080	Anywhere ▼	0.0.0.0/0, ::/0	Delete

Enter a valid port or port range between 0 and 65535. For example: 80 or 0-1023.

- Once the Service is created, go to Task and open it.



Amazon Elastic Container Service > Clusters > spring-cluster > Tasks

spring-cluster

Cluster overview

ARN arn:aws:ecs:eu-west-1:905418486784:cluster/spring-cluster	Status Active	CloudWatch monitoring Default	Registered container instances -

Services

Draining -	Active 1

Tasks

Pending -	Running 1

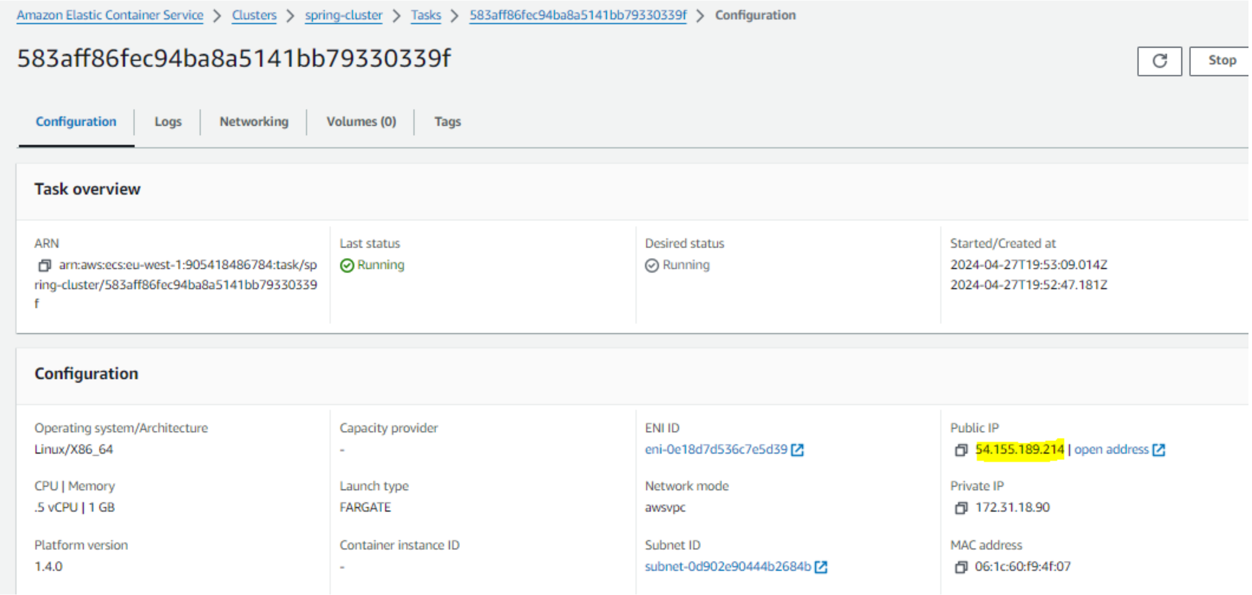
Tasks (1)

Filter tasks by property or value

Filter desired status: Running

Filter launch type: Any launch type

☐	Task	Last status	Desired st...	T...	Health sta...	Started at	Container instan...	Launch type	Platform ...	CPU	Me
☐	583aff...	Running	Running	spri...	Unknown	5 minutes ago	-	FARGATE	1.4.0	.5 vCPU	1 G



Amazon Elastic Container Service > Clusters > spring-cluster > Tasks > 583aff86fec94ba8a5141bb79330339f > Configuration

583aff86fec94ba8a5141bb79330339f

Task overview

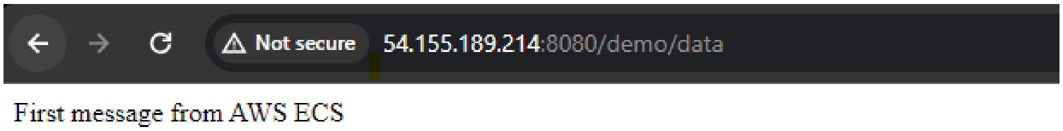
ARN arn:aws:ecs:eu-west-1:905418486784:task/spring-cluster/583aff86fec94ba8a5141bb79330339f	Last status Running	Desired status Running	Started/Created at 2024-04-27T19:53:09.014Z 2024-04-27T19:52:47.181Z

Configuration

Operating system/Architecture Linux/X86_64	Capacity provider -	ENI ID eni-0e18d7d536c7e5d39	Public IP 54.155.189.214 | open address
CPU | Memory .5 vCPU | 1 GB	Launch type FARGATE	Network mode awsvpc	Private IP 172.31.18.90
Platform version 1.4.0	Container instance ID -	Subnet ID subnet-0d902e90444b2684b	MAC address 06:1c:60:f9:4f:07

- Copy the Public IP and test the API, if we can browse it or not.

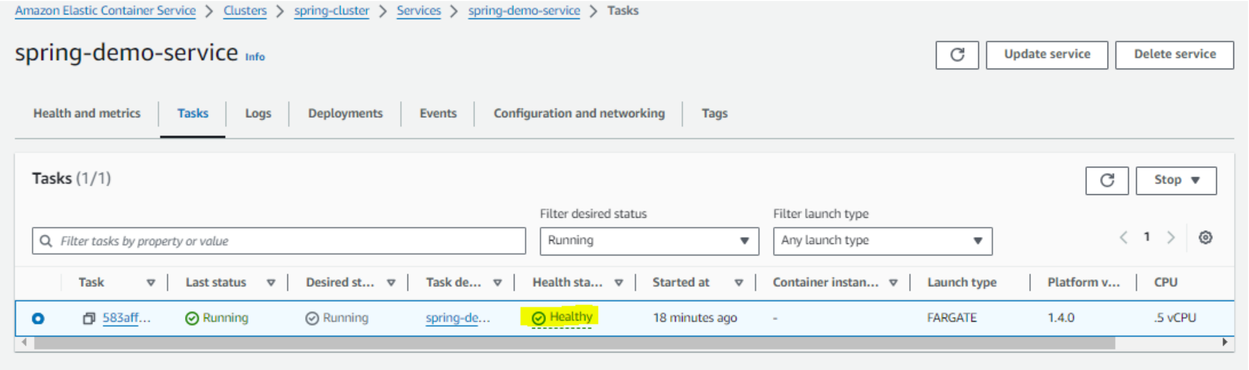
54.155.189.214:8080/demo/data



First message from AWS ECS

So we got the first message from AWS ECS which means our application is deployed successfully onto the ECS as we have received a message from our application.

And if we go back into the services, in the task, we can see we are getting the status as healthy as well. So how is this status getting healthy because it is calling that actuator Health API command which we provided in the configuration otherwise it will not show healthy


Amazon Elastic Container Service > Clusters > spring-cluster > Services > spring-demo-service > Tasks
spring-demo-service Info
Update service
Delete service
Health and metrics
Tasks
Logs
Deployments
Events
Configuration and networking
Tags
Stop ▾
Filter tasks by property or value

Tasks (1/1)

Filter desired status: Running

Filter launch type: Any launch type

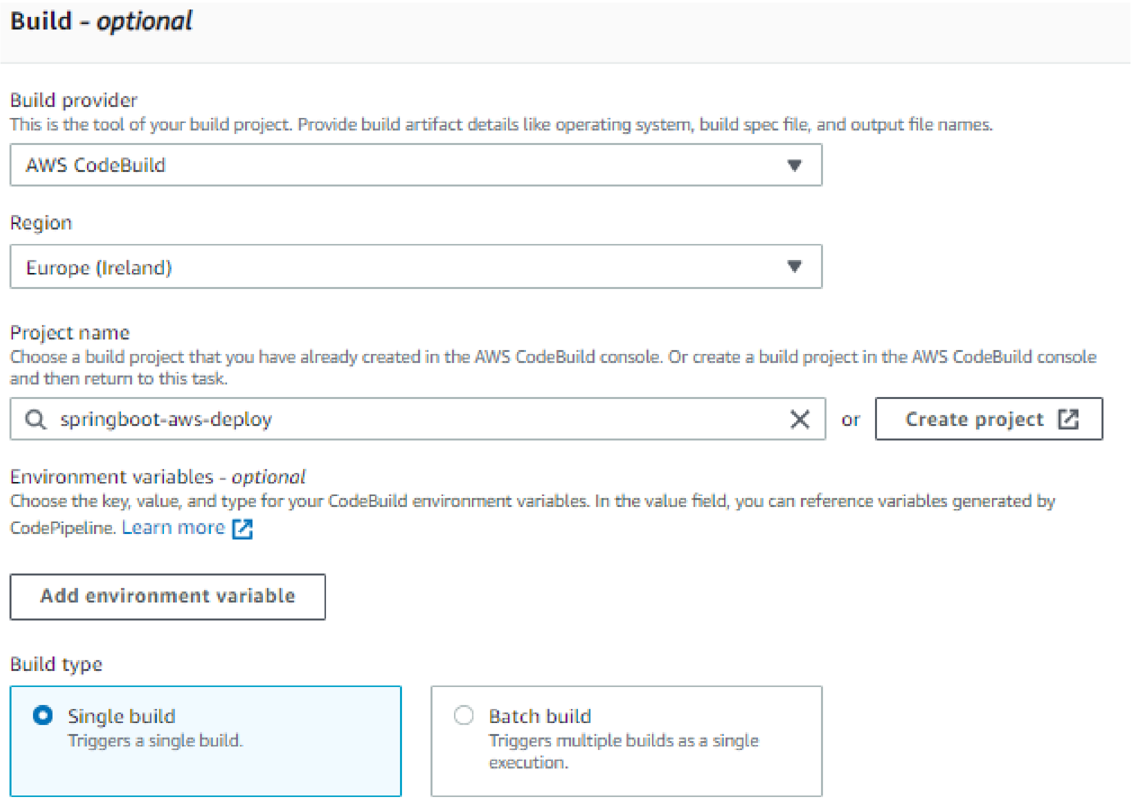
Task	Last status	Desired st...	Task de...	Health sta...	Started at	Container instan...	Launch type	Platform v...	CPU
583aff...	Running	Running	spring-de...	Healthy	18 minutes ago	-	FARGATE	1.4.0	.5 vCPU

So this is how we have completed our step three.

STEP 4: AWS CodePipeline

- Go to **CodePipeline** in **AWS Console > Create Pipeline.**
  - **Pipeline name:** AWS-demo-pipeline
  - **Source provider:** GitHub (Version 2)
  - **Connection:** Connect to GitHub
    - **Connection name:** aws-codepipeline-connection
    - **GitHub Apps: Install New App > Select the Repository and Branch.**

- Add the Build stage and select the Code Build as provider and project name.



Build - *optional*

Build provider
This is the tool of your build project. Provide build artifact details like operating system, build spec file, and output file names.
AWS CodeBuild

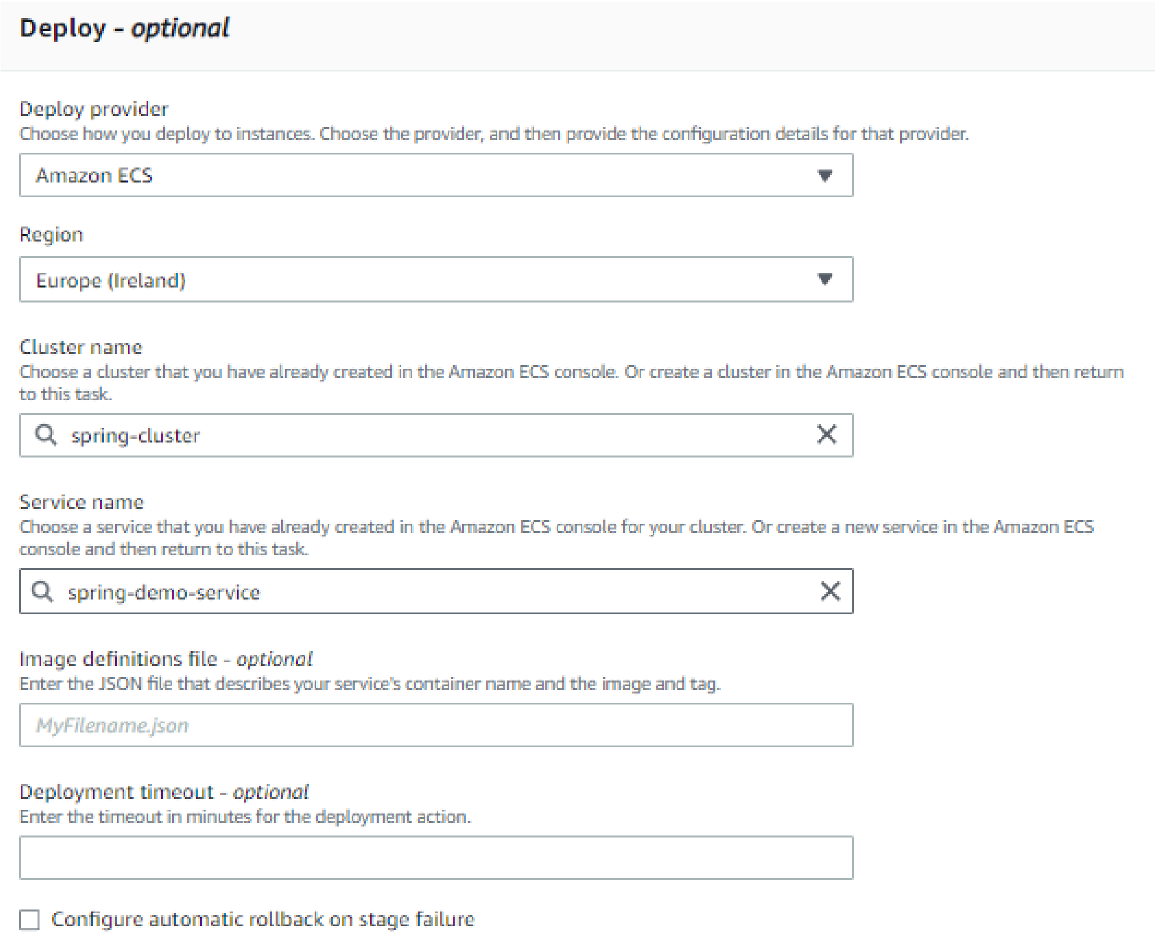
Region
Europe (Ireland)

Project name
Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.
springboot-aws-deploy

Environment variables - *optional*
Choose the key, value, and type for your CodeBuild environment variables. In the value field, you can reference variables generated by CodePipeline. Learn more

Build type
● Single build
Triggers a single build.
○ Batch build
Triggers multiple builds as a single execution.

- Add the Deploy stage and select the AWS ECS as Provider and select the cluster and service name which we have created.



Deploy - *optional*

Deploy provider

Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.

Amazon ECS ▼

Region

Europe (Ireland) ▼

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.

spring-cluster

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.

spring-demo-service

Image definitions file - *optional*

Enter the JSON file that describes your service's container name and the image and tag.

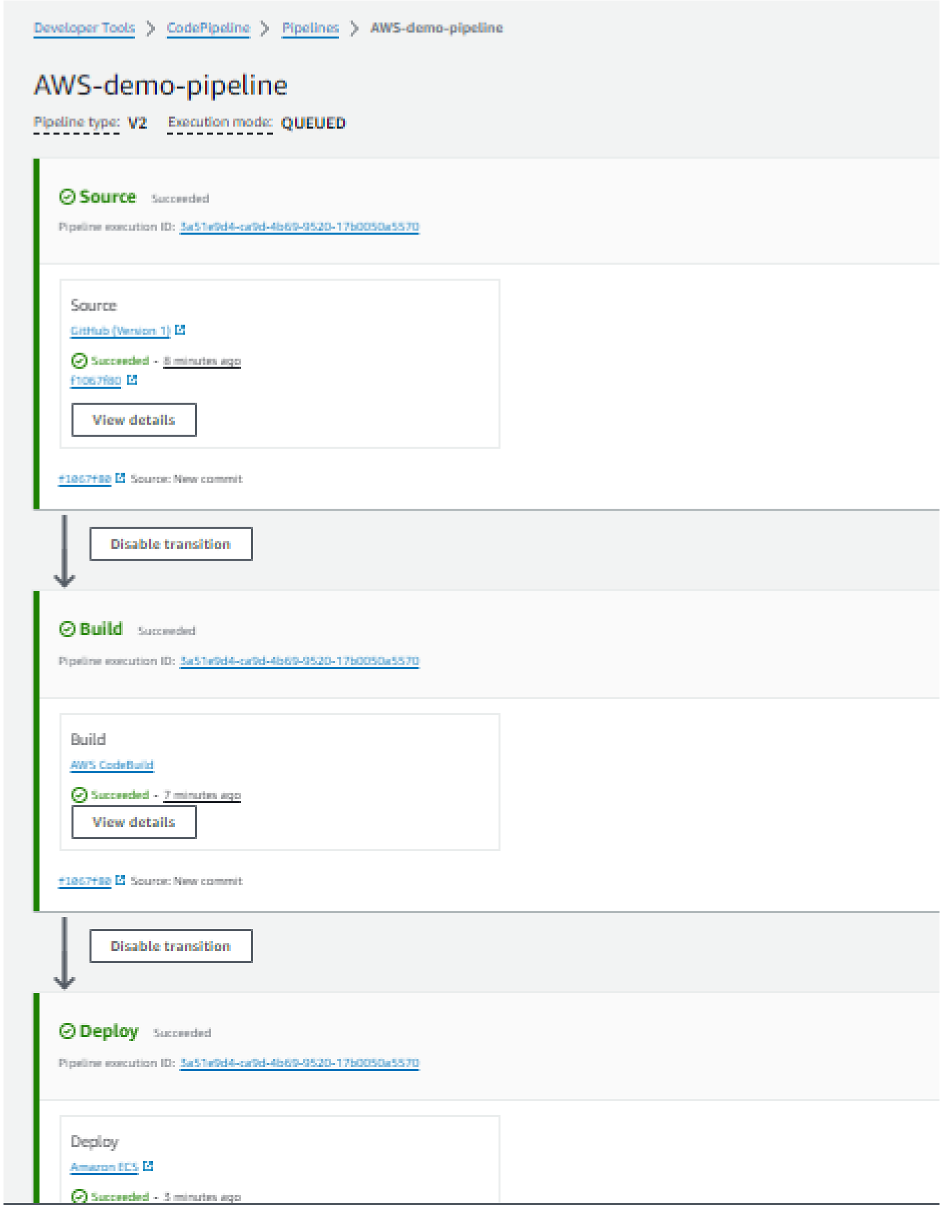
MyFilename.json

Deployment timeout - *optional*

Enter the timeout in minutes for the deployment action.

□ Configure automatic rollback on stage failure

- Now, Review the Pipeline configurations and create it. Once we created the Pipeline automatically started.



Developer Tools > CodePipeline > Pipelines > AWS-demo-pipeline

AWS-demo-pipeline

Pipeline type: V2   Execution mode: QUEUED

Source Succeeded

Pipeline execution ID: 3a51e9d4-ca9d-4b69-9520-17b0050a5570

Source

GitHub (Version 1)

Succeeded - 8 minutes ago

f1067f80

f1067f80 Source: New commit

Build Succeeded

Pipeline execution ID: 3a51e9d4-ca9d-4b69-9520-17b0050a5570

Build

AWS CodeBuild

Succeeded - 7 minutes ago

f1067f80 Source: New commit

Deploy Succeeded

Pipeline execution ID: 3a51e9d4-ca9d-4b69-9520-17b0050a5570

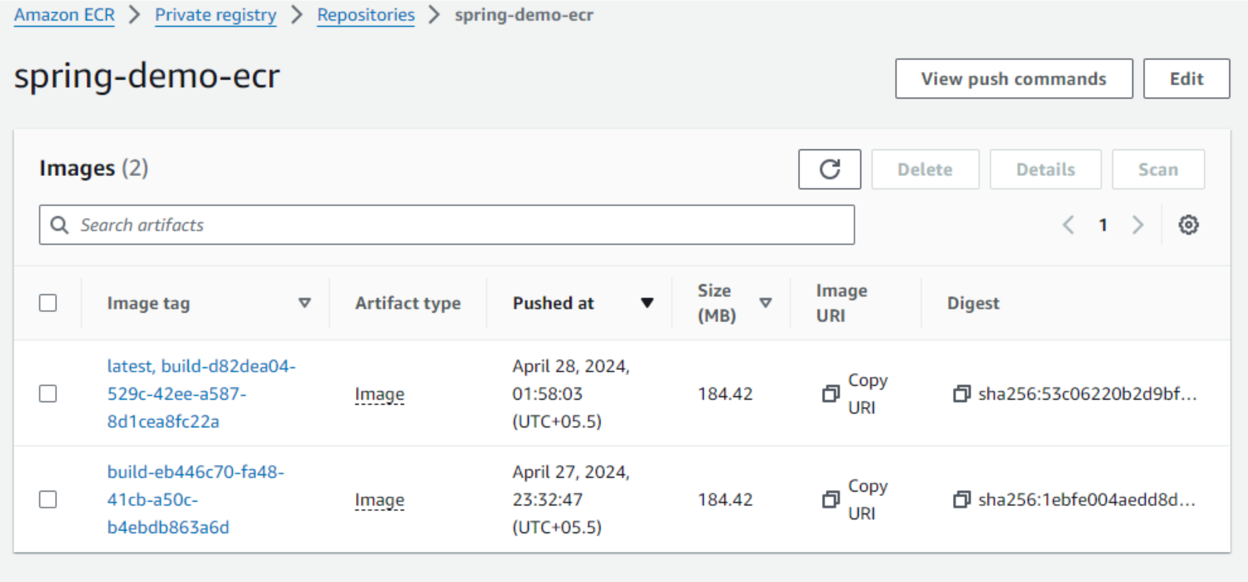
Deploy

Amazon ECS

Succeeded - 3 minutes ago

`

Let's check the ECR, we have 2 images are created. One for the initial time when we created the ECR and another is created after the pipeline execution.
So, whenever the pipeline is getting executed the new image will gets created.


Amazon ECR > Private registry > Repositories > spring-demo-ecr
View push commands
Edit
Delete
Details
Scan
Search artifacts
1

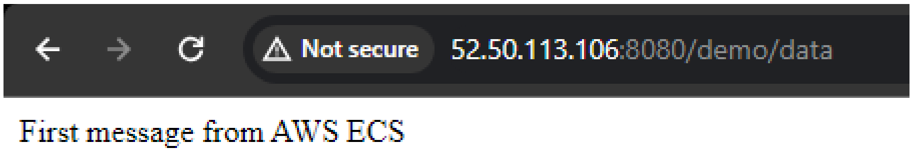
spring-demo-ecr

Images (2)

□	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
□	latest, build-d82dea04-529c-42ee-a587-8d1cea8fc22a	Image	April 28, 2024, 01:58:03 (UTC+05.5)	184.42	Copy URI	sha256:53c06220b2d9bf...
□	build-eb446c70-fa48-41cb-a50c-b4ebdb863a6d	Image	April 27, 2024, 23:32:47 (UTC+05.5)	184.42	Copy URI	sha256:1ebfe004aedd8d...

- Now, to test copy the public ip from **ECS > Cluster > Service > Task.**

52.50.113.106:8080/demo/data



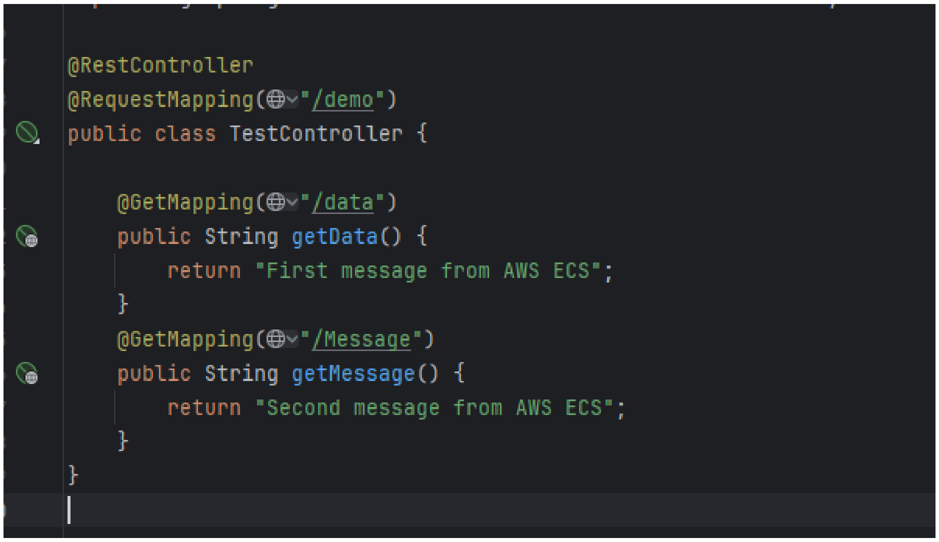
52.50.113.106:8080/demo/data

First message from AWS ECS

- Let's do one more test we will add one more API into our **springboot application**.

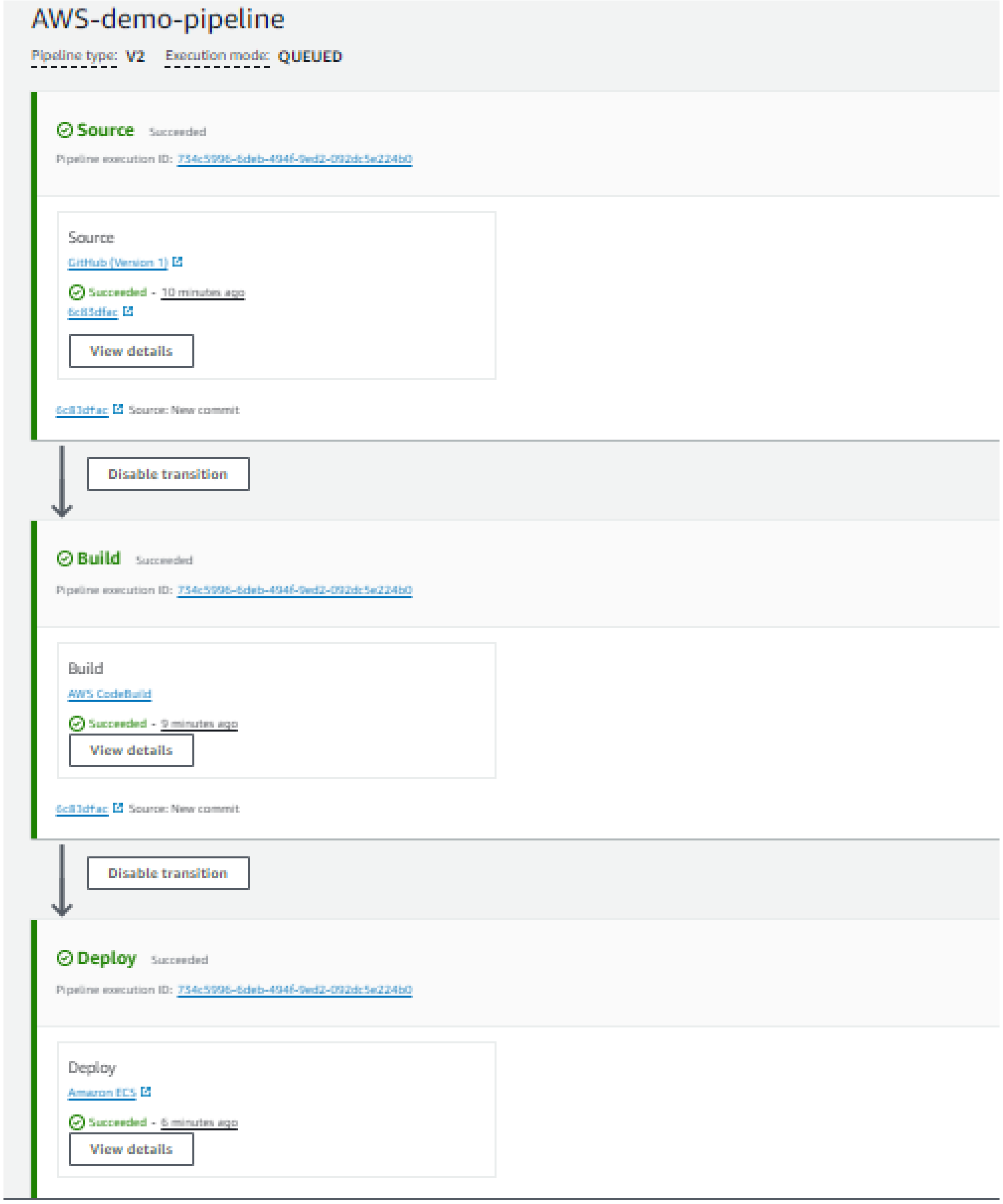
Open the TestController.java and add the below code

```
@GetMapping("/Message")
public String getMessage() {
    return "Second message from AWS ECS";
}
```



```
@RestController
@RequestMapping("/demo")
public class TestController {

    @GetMapping("/data")
    public String getData() {
        return "First message from AWS ECS";
    }
    @GetMapping("/Message")
    public String getMessage() {
        return "Second message from AWS ECS";
    }
}
```

- Now, push these changes to GitHub and once we push the changes the pipeline will automatically get executed.



AWS-demo-pipeline

Pipeline type: V2 Execution mode: QUEUED

✓ Source Succeeded

Pipeline execution ID: 754c5996-6deb-494f-9ed2-092dc5e224b0

Source

GitHub (Version 1)

✓ Succeeded - 10 minutes ago

6c83dfac

6c83dfac Source: New commit

✓ Build Succeeded

Pipeline execution ID: 754c5996-6deb-494f-9ed2-092dc5e224b0

Build

AWS CodeBuild

✓ Succeeded - 9 minutes ago

6c83dfac Source: New commit

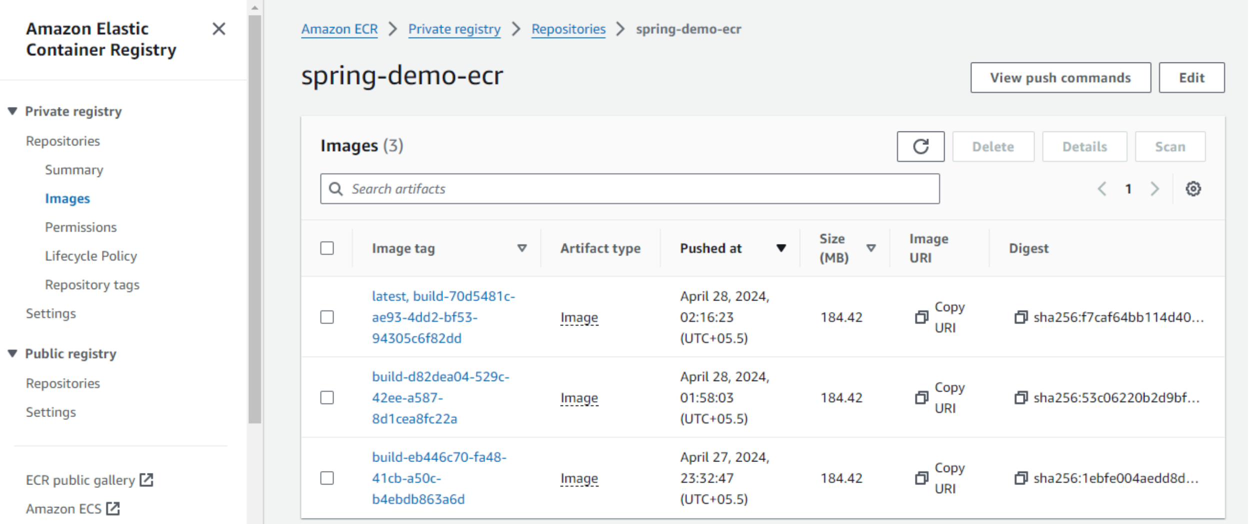
✓ Deploy Succeeded

Pipeline execution ID: 754c5996-6deb-494f-9ed2-092dc5e224b0

Deploy

Amazon ECS

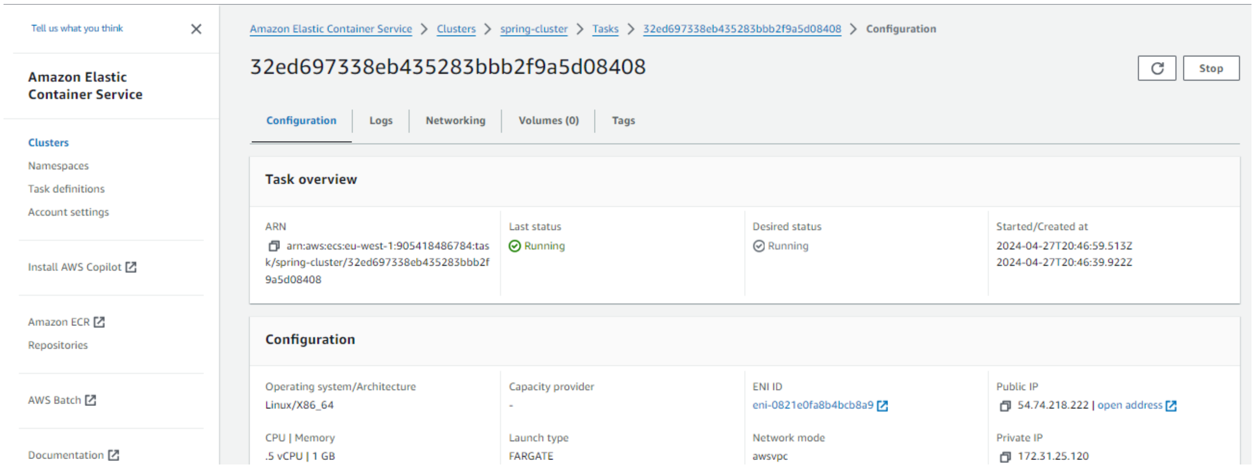
✓ Succeeded - 6 minutes ago



spring-demo-ecr

Images (3)

☐	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
☐	latest, build-70d5481c-ae93-4dd2-bf53-94305c6f82dd	Image	April 28, 2024, 02:16:23 (UTC+05.5)	184.42	Copy URI	sha256:f7caf64bb114d40...
☐	build-d82dea04-529c-42ee-a587-8d1cea8fc22a	Image	April 28, 2024, 01:58:03 (UTC+05.5)	184.42	Copy URI	sha256:53c06220b2d9bf...
☐	build-eb446c70-fa48-41cb-a50c-b4ebdb863a6d	Image	April 27, 2024, 23:32:47 (UTC+05.5)	184.42	Copy URI	sha256:1ebfe004aedd8d...



32ed697338eb435283bbb2f9a5d08408

Task overview

ARN: arn:aws:ecs:eu-west-1:905418486784:task/spring-cluster/32ed697338eb435283bbb2f9a5d08408

Last status: Running

Desired status: Running

Started/Created at: 2024-04-27T20:46:59.513Z 2024-04-27T20:46:39.922Z

Configuration

Operating system/Architecture: Linux/X86_64

Capacity provider: -

ENI ID: eni-0821e0fa8b4bcb8a9

Public IP: 54.74.218.222 | open address

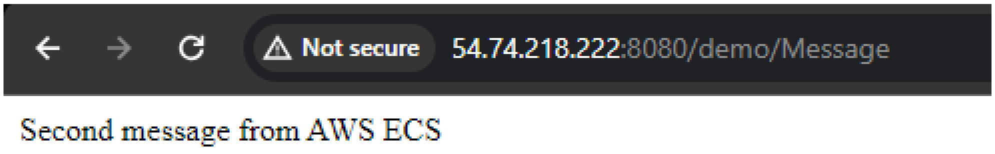
CPU | Memory: .5 vCPU | 1 GB

Launch type: FARGATE

Network mode: awsvpc

Private IP: 172.31.25.120

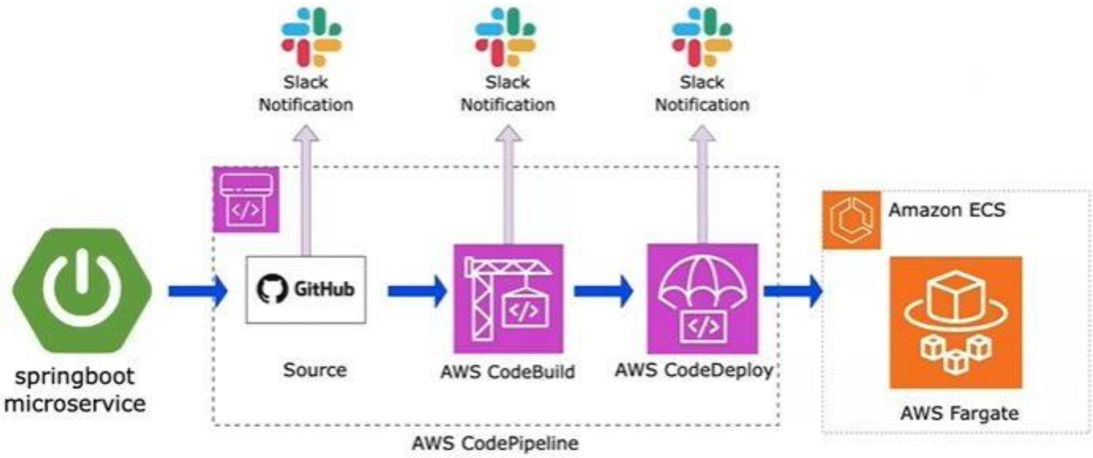
- Our application and accessible properly.



54.74.218.222:8080/demo/Message

Second message from AWS ECS

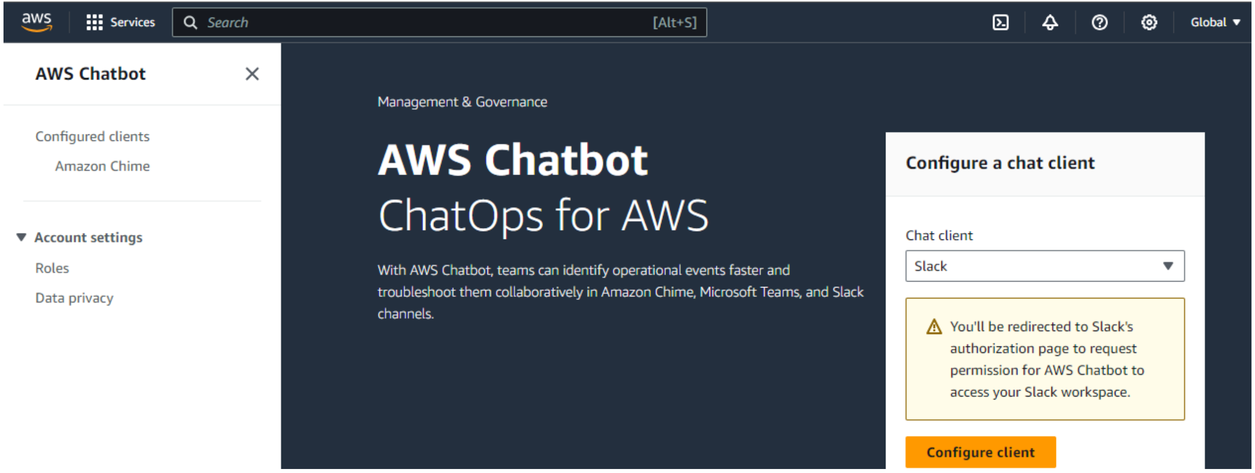
STEP 5 - SLACK Notifications



Now, Let's integrate Slack notifications into the AWS CodePipeline. To add a notification for each stage we can add that notification from Notify option in the Pipeline.

But before configuring that we will require to configure Slack.

- Go to AWS Chatbot in **AWS console > Chat Client > Slack > Configure Client.**



Management & Governance

AWS Chatbot

ChatOps for AWS

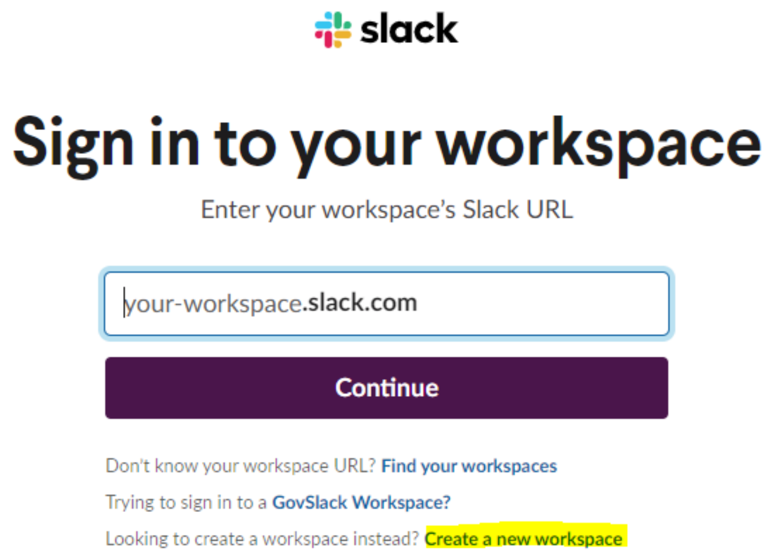
With AWS Chatbot, teams can identify operational events faster and troubleshoot them collaboratively in Amazon Chime, Microsoft Teams, and Slack channels.

Configure a chat client

Chat client: Slack

You'll be redirected to Slack's authorization page to request permission for AWS Chatbot to access your Slack workspace.

- When we click on Configure Client, it will redirects to Slack website. Create a New workspace there and login with your email id and setup and create the Slack Account by filling the basic details.



slack

Sign in to your workspace

Enter your workspace's Slack URL

your-workspace.slack.com

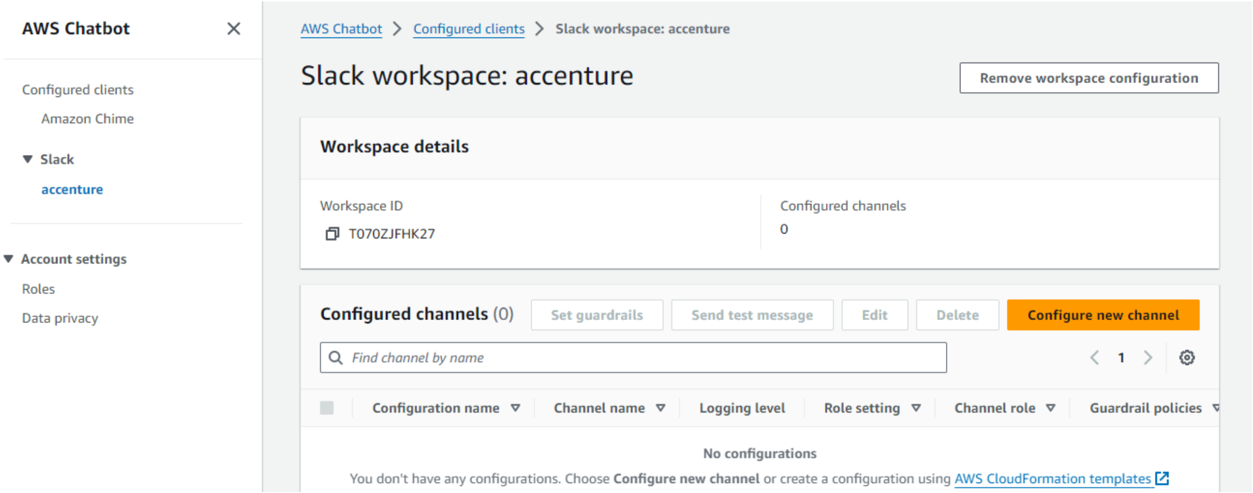
Continue

Don't know your workspace URL? Find your workspaces

Trying to sign in to a GovSlack Workspace?

Looking to create a workspace instead? Create a new workspace

- Once Slack account is created, again go back to AWS chatbot and click on configure client and allow the access permissions.



Slack workspace: accenture

Workspace details

Workspace ID: T070ZJFHK27

Configured channels: 0

Configured channels (0)

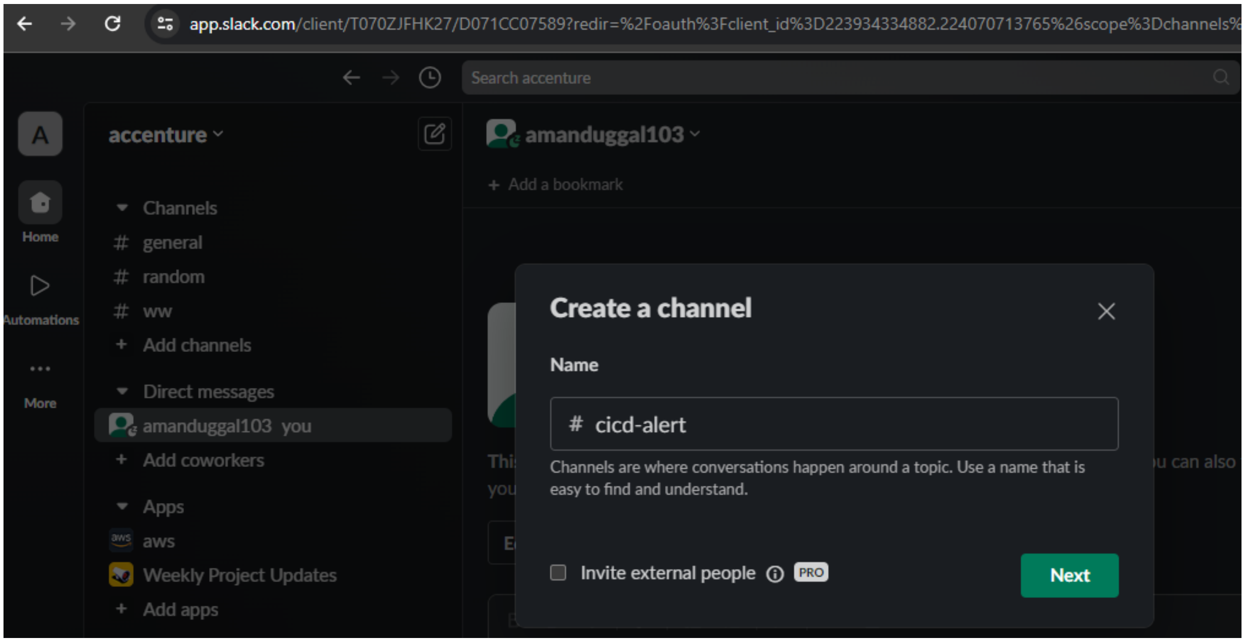
Configuration name	Channel name	Logging level	Role setting	Channel role	Guardrail policies

No configurations

You don't have any configurations. Choose **Configure new channel** or create a configuration using AWS CloudFormation templates

- Create new alert in Slack.

Channels > Create channel > Enter the Channel Name > Public access.



app.slack.com/client/T070ZJFHK27/D071CC07589?redir=%2Foauth%3Fclient_id%3D223934334882.224070713765%26scope%3Dchannels%

Create a channel

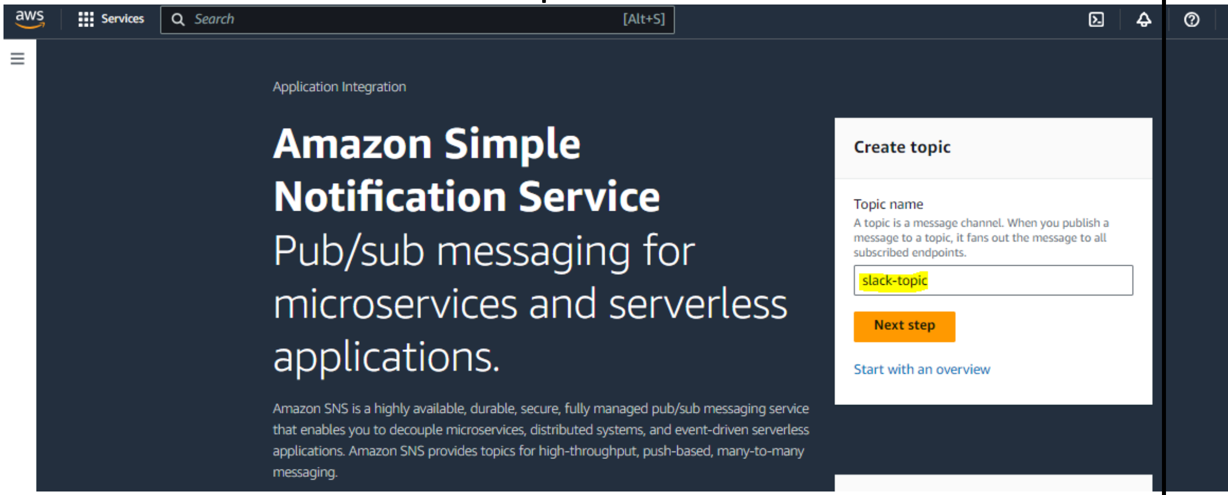
Name

cicd-alert

Channels are where conversations happen around a topic. Use a name that is easy to find and understand.

□ Invite external people PRO

- Now, **Configure New Channel** in AWS Chatbot.
  - **Configuration name:** cicd-alert
  - **Public channel name:** cicd-alert **--#Select the channel name which we have created in the Slack.**
  - **Channel role:** Create an IAM role
  - **Role Name:** slack-channel-role
  - **SNS topics:**
    - First, create the **SNS**.
    - Go to SNS in **AWS console > Create Topic** and remain the default details.



Application Integration

Amazon Simple Notification Service

Pub/sub messaging for microservices and serverless applications.

Amazon SNS is a highly available, durable, secure, fully managed pub/sub messaging service that enables you to decouple microservices, distributed systems, and event-driven serverless applications. Amazon SNS provides topics for high-throughput, push-based, many-to-many messaging.

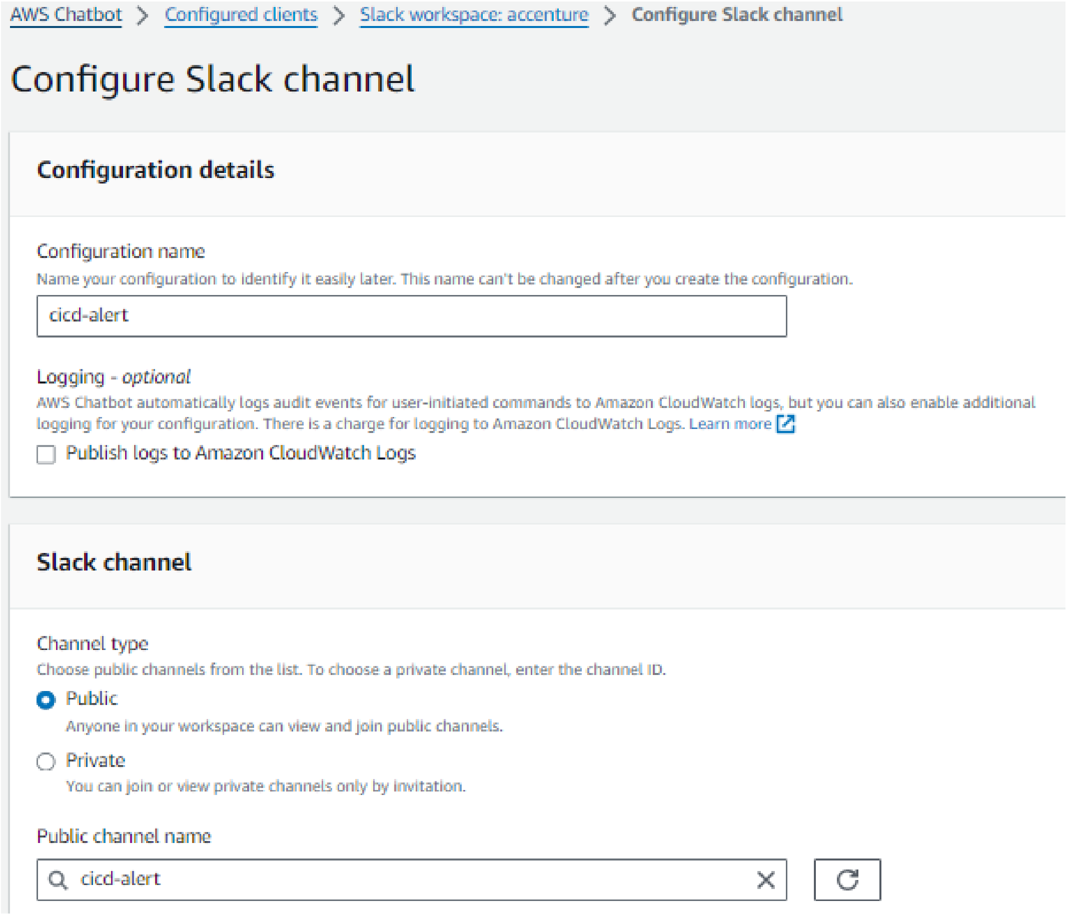
Create topic

Topic name

A topic is a message channel. When you publish a message to a topic, it fans out the message to all subscribed endpoints.

slack-topic

- **Region 1: --#Select the region on which the SNS is created.**
- **Topics 1: --#Select the Topic which we have created.**



AWS Chatbot > Configured clients > Slack workspace: accenture > Configure Slack channel

Configure Slack channel

Configuration details

Configuration name

Name your configuration to identify it easily later. This name can't be changed after you create the configuration.

cicd-alert

Logging - *optional*

AWS Chatbot automatically logs audit events for user-initiated commands to Amazon CloudWatch logs, but you can also enable additional logging for your configuration. There is a charge for logging to Amazon CloudWatch Logs. Learn more

□ Publish logs to Amazon CloudWatch Logs

Slack channel

Channel type

Choose public channels from the list. To choose a private channel, enter the channel ID.

● Public

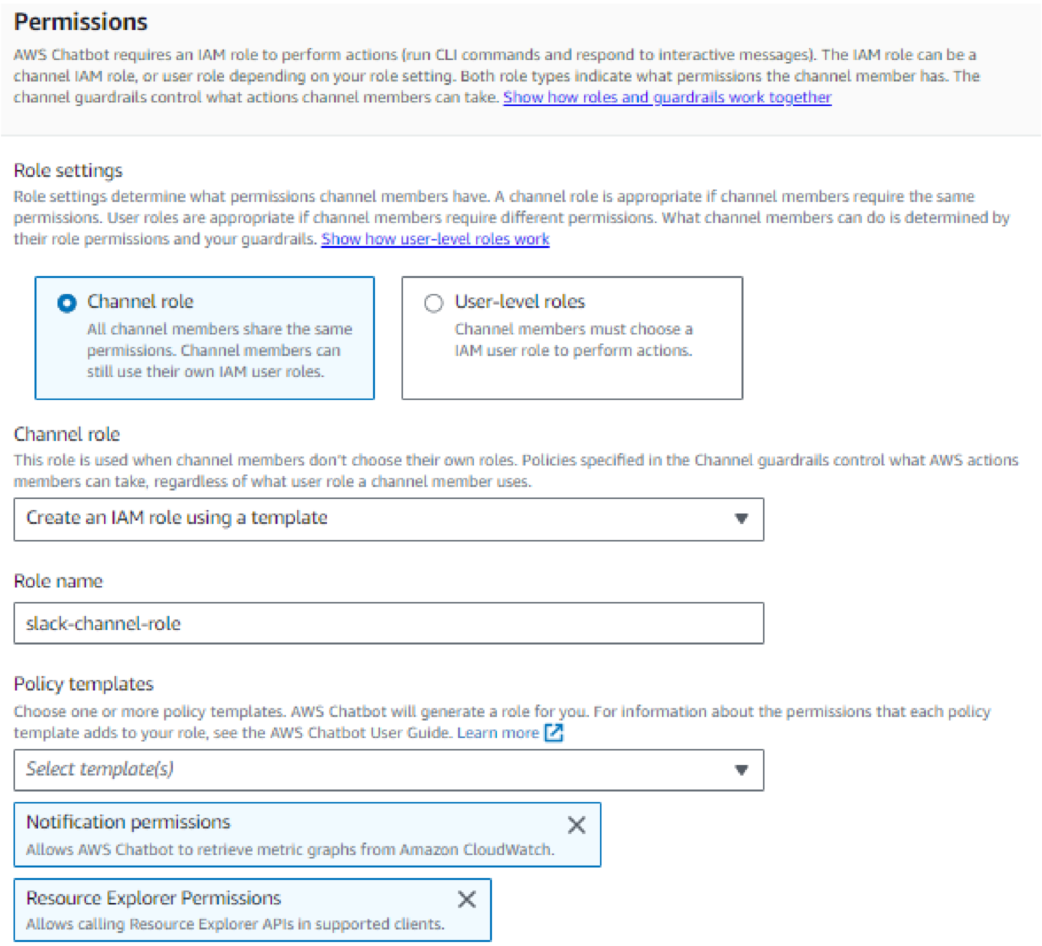
Anyone in your workspace can view and join public channels.

○ Private

You can join or view private channels only by invitation.

Public channel name

cicd-alert



Permissions

AWS Chatbot requires an IAM role to perform actions (run CLI commands and respond to interactive messages). The IAM role can be a channel IAM role, or user role depending on your role setting. Both role types indicate what permissions the channel member has. The channel guardrails control what actions channel members can take. Show how roles and guardrails work together

Role settings

Role settings determine what permissions channel members have. A channel role is appropriate if channel members require the same permissions. User roles are appropriate if channel members require different permissions. What channel members can do is determined by their role permissions and your guardrails. Show how user-level roles work

● Channel role
All channel members share the same permissions. Channel members can still use their own IAM user roles.

○ User-level roles
Channel members must choose a IAM user role to perform actions.

Channel role

This role is used when channel members don't choose their own roles. Policies specified in the Channel guardrails control what AWS actions members can take, regardless of what user role a channel member uses.

Create an IAM role using a template ▼

Role name

slack-channel-role

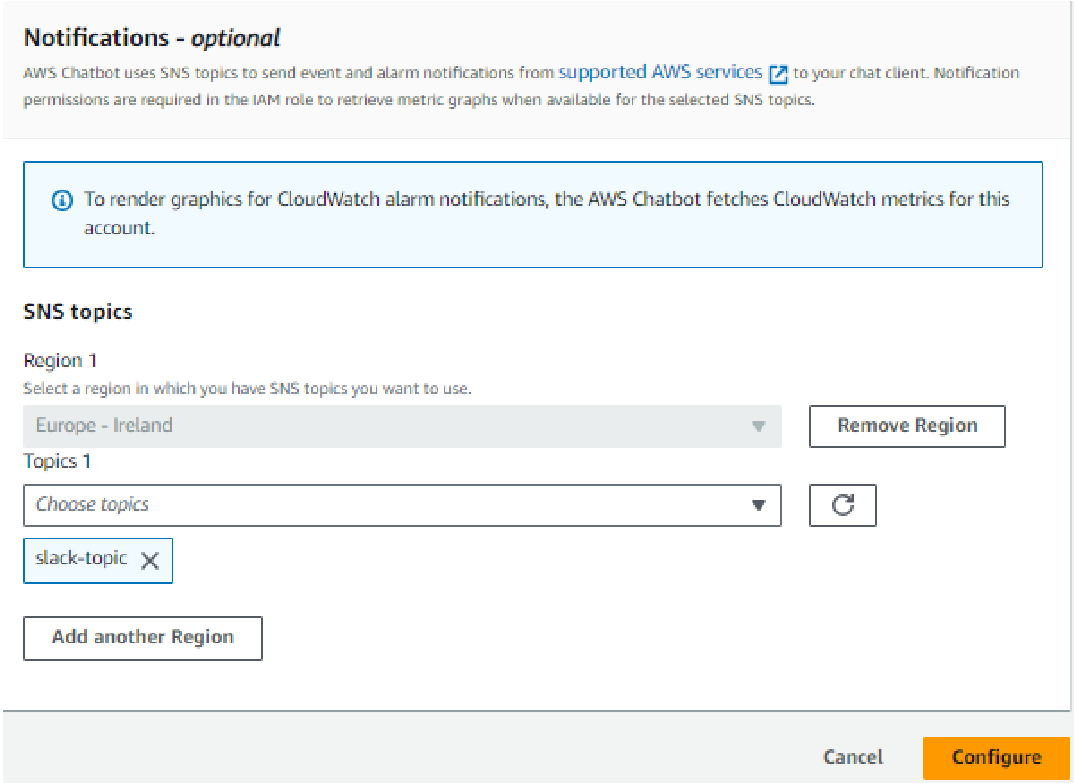
Policy templates

Choose one or more policy templates. AWS Chatbot will generate a role for you. For information about the permissions that each policy template adds to your role, see the AWS Chatbot User Guide. Learn more

Select template(s) ▼

Notification permissions ✕
Allows AWS Chatbot to retrieve metric graphs from Amazon CloudWatch.

Resource Explorer Permissions ✕
Allows calling Resource Explorer APIs in supported clients.



Notifications - *optional*

AWS Chatbot uses SNS topics to send event and alarm notifications from supported AWS services to your chat client. Notification permissions are required in the IAM role to retrieve metric graphs when available for the selected SNS topics.

To render graphics for CloudWatch alarm notifications, the AWS Chatbot fetches CloudWatch metrics for this account.

SNS topics

Region 1

Select a region in which you have SNS topics you want to use.

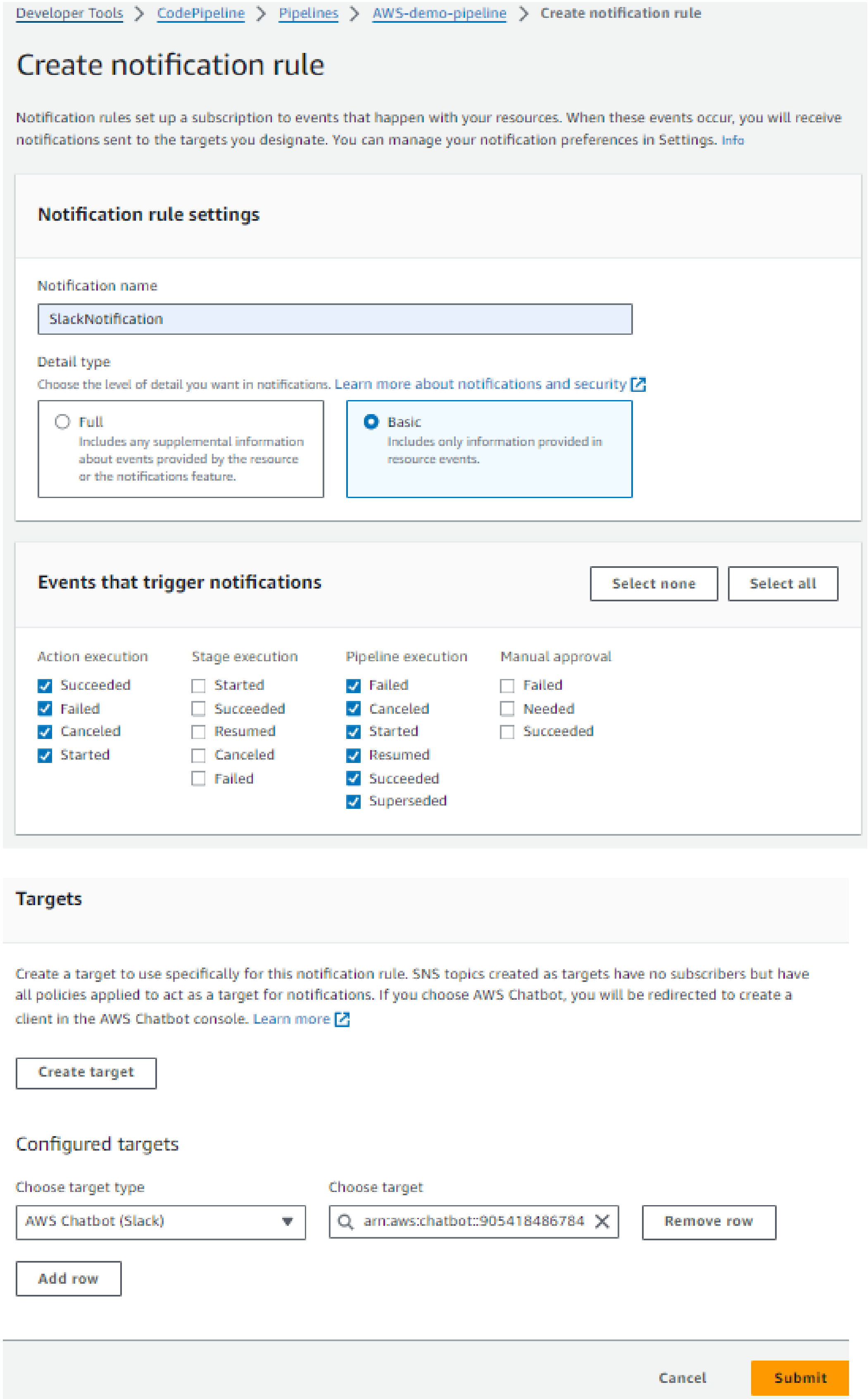
Europe - Ireland

Topics 1

Choose topics

slack-topic

- Now, once the Slack Configuration is done, Go to **CodePipeline and select the Notify option > Create Notification rule**.
    - **Notification name:** SlackNotification
    - **Detail type:** Basic
    - **Events that trigger notifications:** Select **Action** and **Pipeline** Execution.
    - **Configured targets:** AWS Chatbot (Slack)



Developer Tools > CodePipeline > Pipelines > AWS-demo-pipeline > Create notification rule

Create notification rule

Notification rules set up a subscription to events that happen with your resources. When these events occur, you will receive notifications sent to the targets you designate. You can manage your notification preferences in Settings. Info

Notification rule settings

Notification name

SlackNotification

Detail type

Choose the level of detail you want in notifications. Learn more about notifications and security

○ Full
Includes any supplemental information about events provided by the resource or the notifications feature.

● Basic
Includes only information provided in resource events.

Events that trigger notifications

Select none | Select all

Action execution
- ☑ Succeeded
- ☑ Failed
- ☑ Canceled
- ☑ Started

Stage execution
- □ Started
- □ Succeeded
- □ Resumed
- □ Canceled
- □ Failed

Pipeline execution
- ☑ Failed
- ☑ Canceled
- ☑ Started
- ☑ Resumed
- ☑ Succeeded
- ☑ Superseded

Manual approval
- □ Failed
- □ Needed
- □ Succeeded

Targets

Create a target to use specifically for this notification rule. SNS topics created as targets have no subscribers but have all policies applied to act as a target for notifications. If you choose AWS Chatbot, you will be redirected to create a client in the AWS Chatbot console. Learn more

Create target

Configured targets

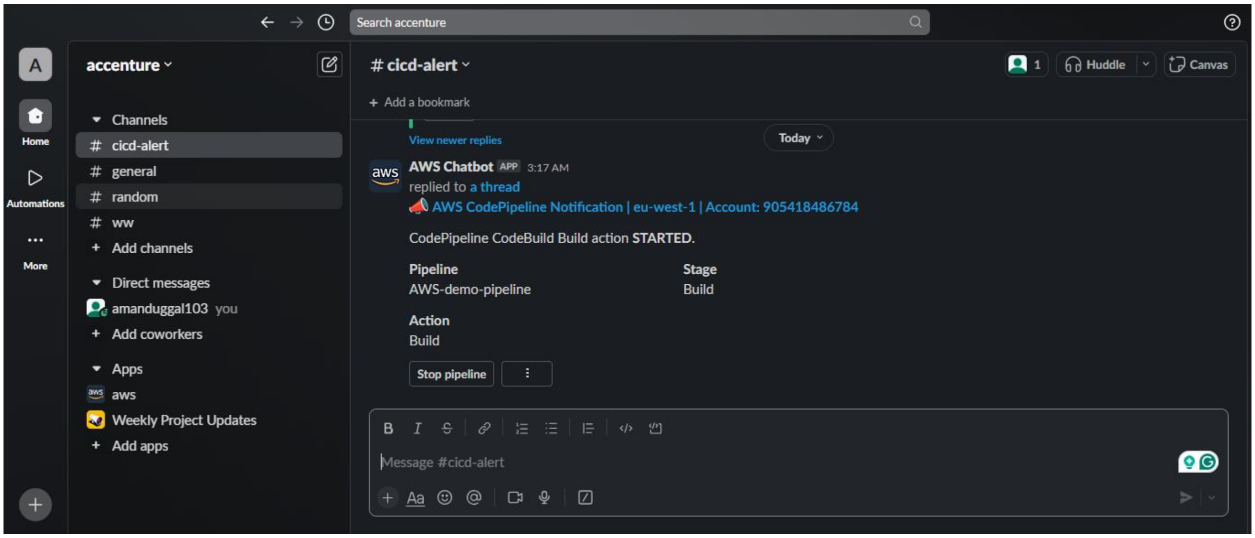
Choose target type	Choose target	
AWS Chatbot (Slack) ▼	arn:aws:chatbot::905418486784 ✕	Remove row

Add row

Cancel | Submit

- Once the setup is done, Release the Pipeline manually and check the Slack Dashboard if you are getting the notifications for every change and action in the pipeline.

We will get the notifications for every action in the slack dashboard.



AWS Chatbot APP 3:17 AM
replied to a thread
📣 AWS CodePipeline Notification | eu-west-1 | Account: 905418486784

CodePipeline CodeBuild Build action STARTED.

Pipeline
AWS-demo-pipeline

Stage
Build

Action
Build