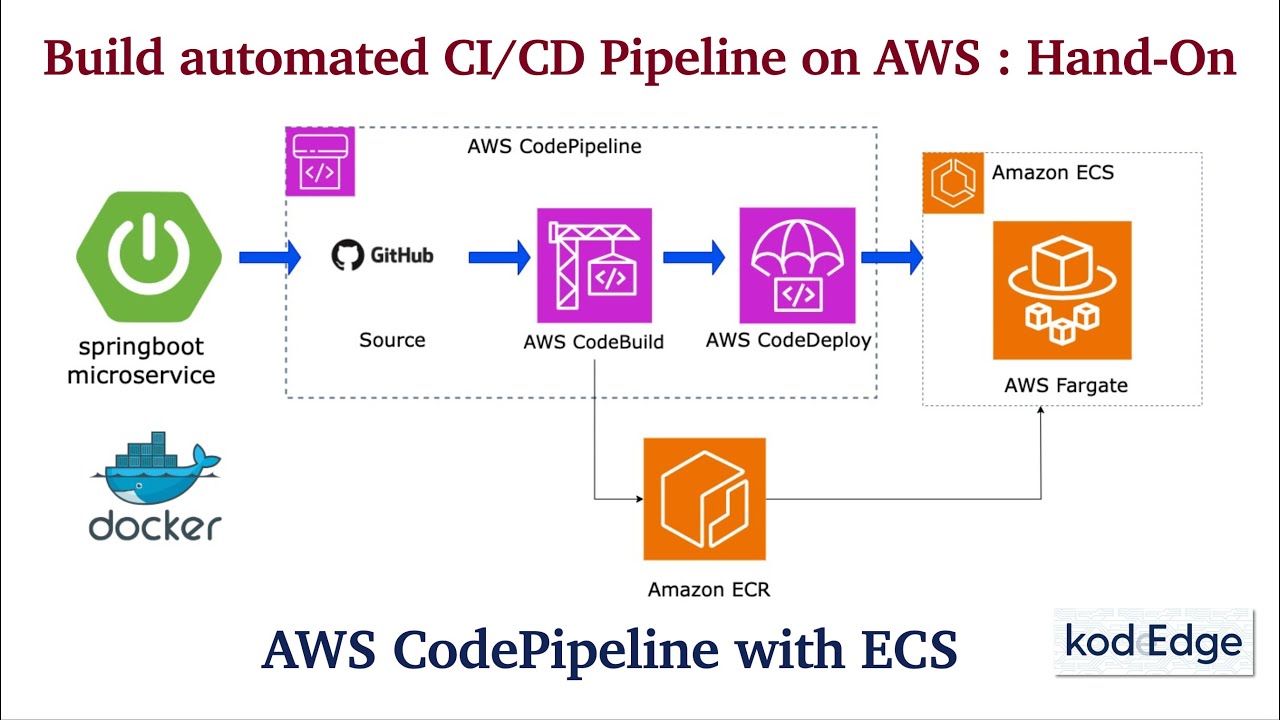
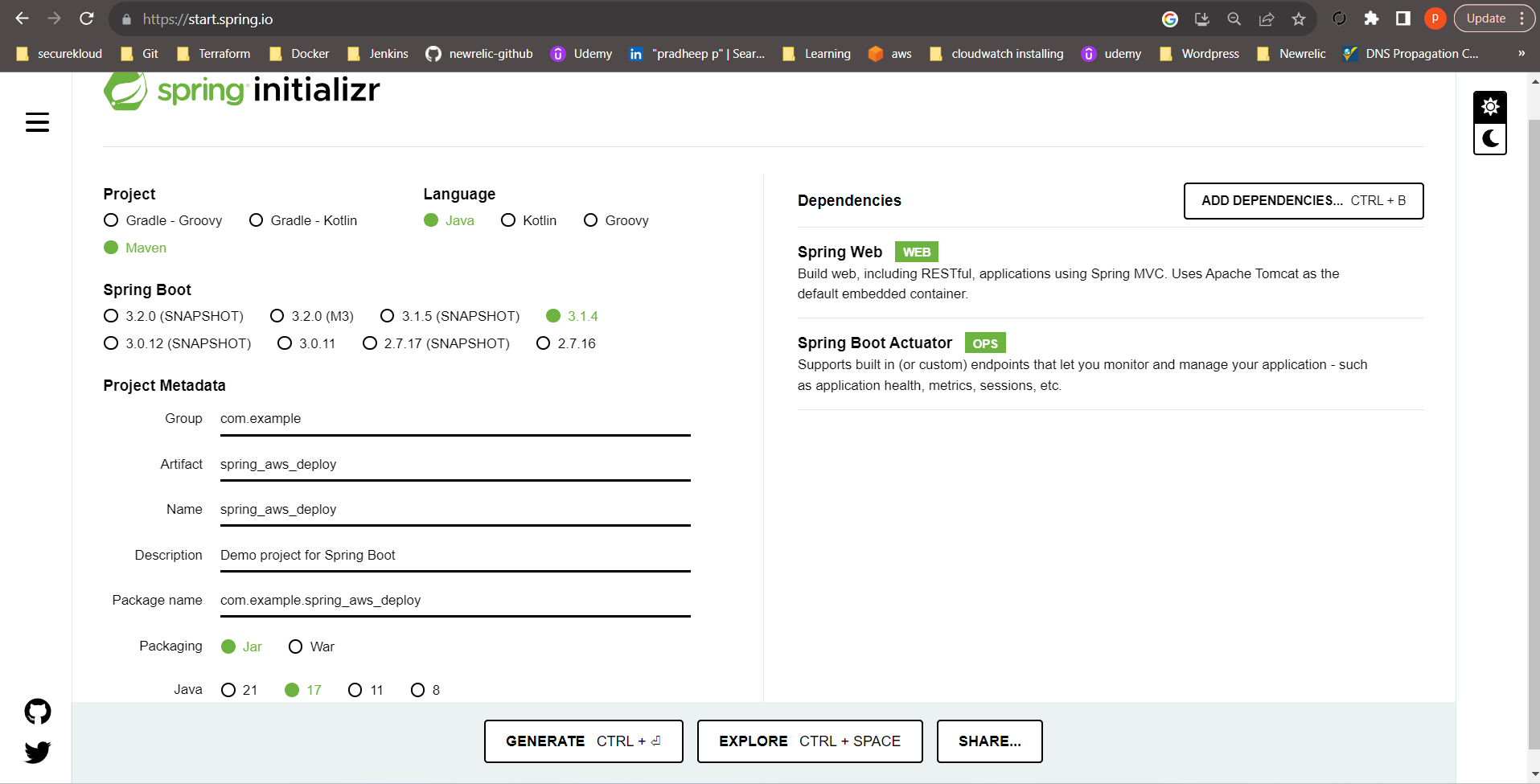
**Intro**

This post describes how to implement a deploy Springboot application on ECS container with CodePipeline and GitHub [AWS]  
  
Below tools and technologies used to build this application.

* SpringBoot Application
* [AWS ECS](https://aws.amazon.com/api-gateway/)
* Code Build
* Code Pipeline
* [GitHub](https://aws.amazon.com/dynamodb/)
* Maven

**Architecture:**

**Step 1:  
 \*** Visit spring initializer to set up the application structure with Dependecies like Spring Web and Spring Boot Actuator

Visit <https://start.spring.io/> and download in local device

**Step 2:**

Update the TestController.java file in spring\_aws\_deploy/src/main/java/com/example/spring\_aws\_deploy

/TestController.java  
  
package com.example.spring\_aws\_deploy;

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

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

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

@RestController

@RequestMapping("/demo")

public class TestController {

@GetMapping("/data")

public String getData() {

return "First message from AWS ECS";

}

}  
  
And Also edit in pom.xml with <finalName>Spring\_Aws\_Deploy</finalName>  


**Step 3:**Create a Dockerfile:  
FROM eclipse-temurin:17-jdk-alpine

RUN apk add curl

VOLUME /tmp

EXPOSE 8080

ADD target/SpringAwsDeploy-service.jar SpringAwsDeploy-service-service.jar

ENTRYPOINT ["java","-jar","/SpringAwsDeploy-service-service.jar"]

**Step 4:**

Inside the directory we need to install java and Maven   
with help of this need to build the project

**Step 5:**  
Create buildspec.yml  
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)

- aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 290191426851.dkr.ecr.ap-south-1.amazonaws.com

- # Replace with this to your repository URI

- REPOSITORY\_URI=290191426851.dkr.ecr.ap-south-1.amazonaws.com/spring\_demo\_ecr

- IMAGE\_TAG=build-$(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

- DOCKER\_CONTAINER\_NAME=spring\_cont

- printf '[{"name":"%s","imageUri":"%s"}]' $DOCKER\_CONTAINER\_NAME $REPOSITORY\_URI:$IMAGE\_TAG > imagedefinitions.json

- echo $DOCKER\_CONTAINER\_NAME

- echo printing imagedefinitions.json

- cat imagedefinitions.json

artifacts:

files:

- imagedefinitions.json

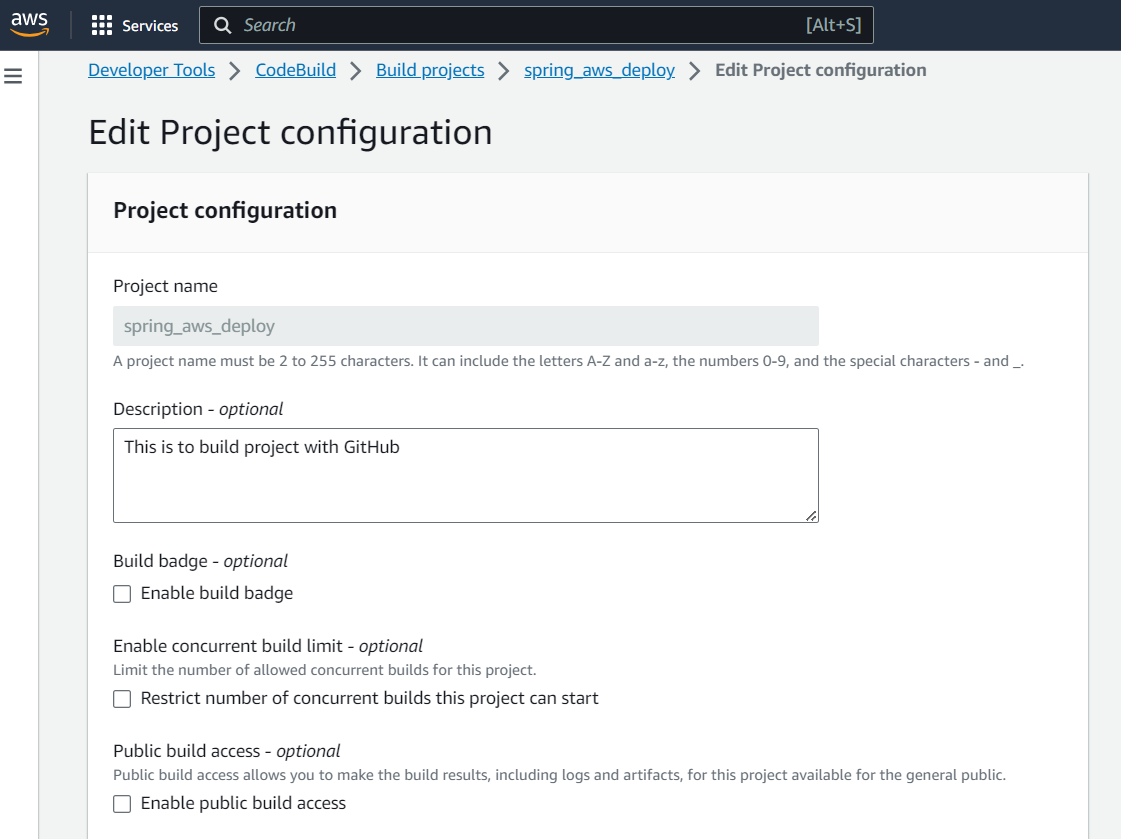
- target/springboot-aws-deploy.jar

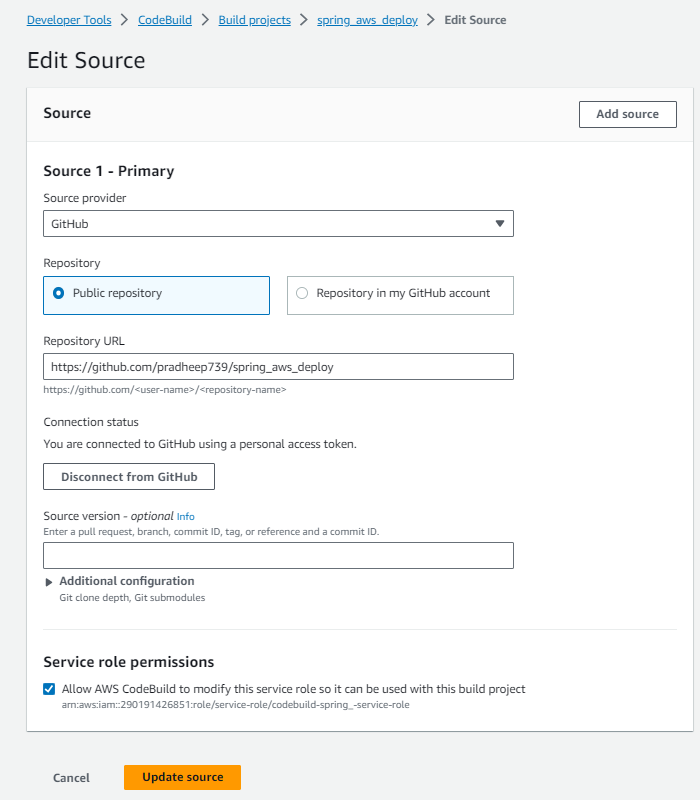
**Step 6:**

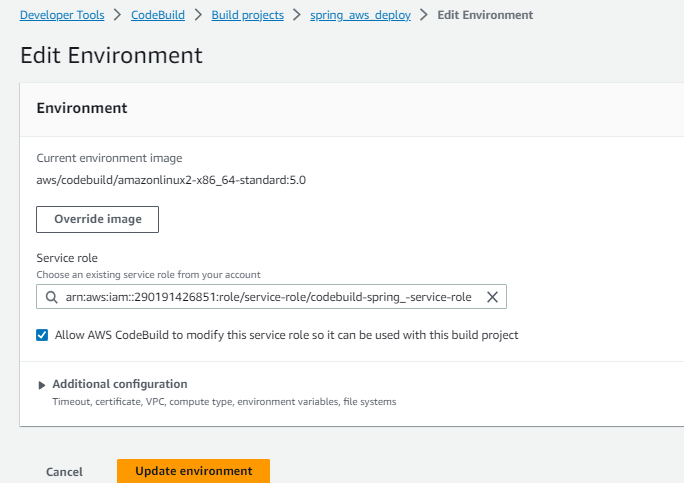
Need to push this entire directory to the github repo  
  
**Step 7:**

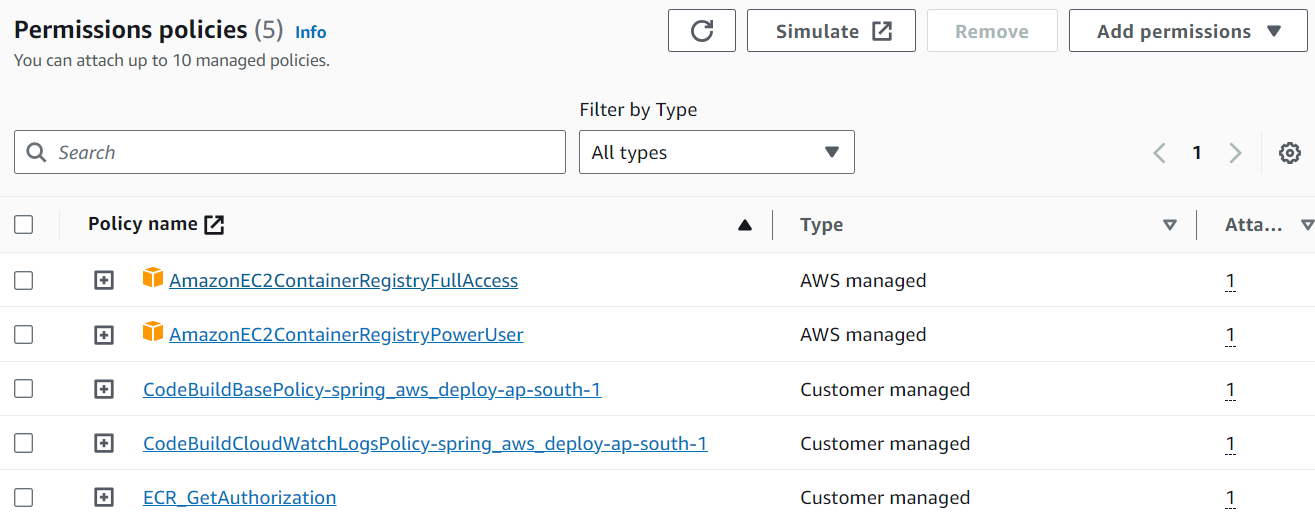
Create a private repo in ECR and need to update the REPOSITORY URI in buildspec.yml

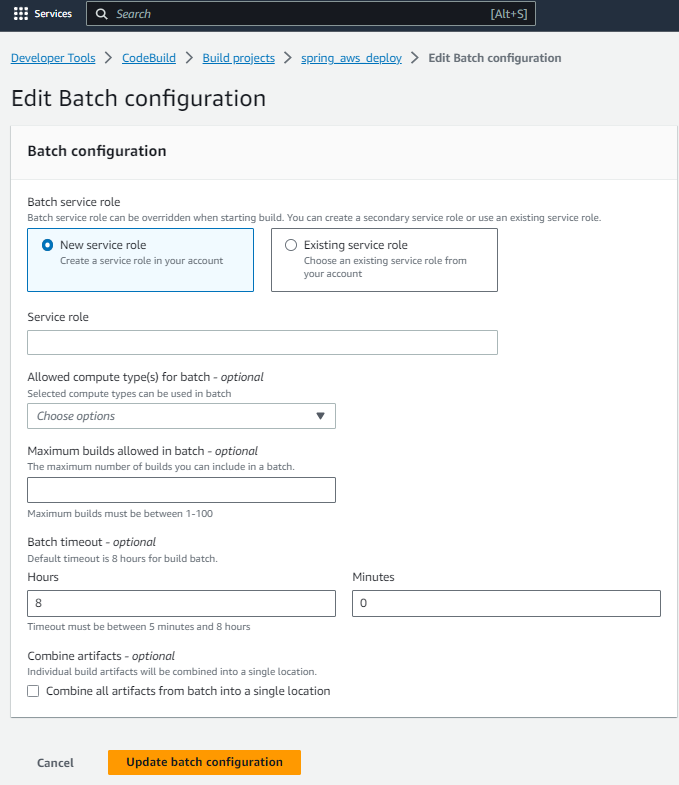
**Step 8:**

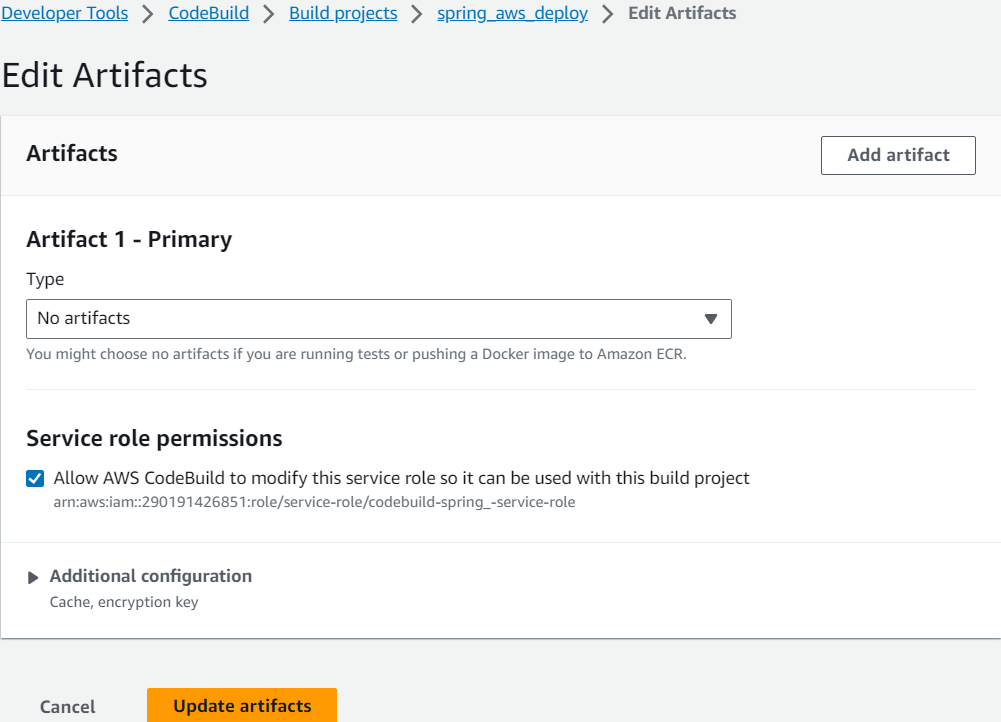
Create CodeBuild Project in AWS and attach policy  
  


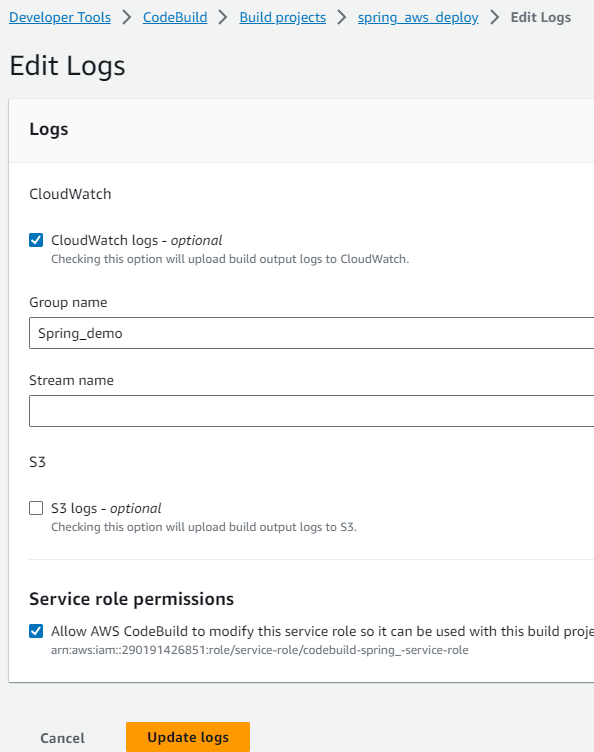








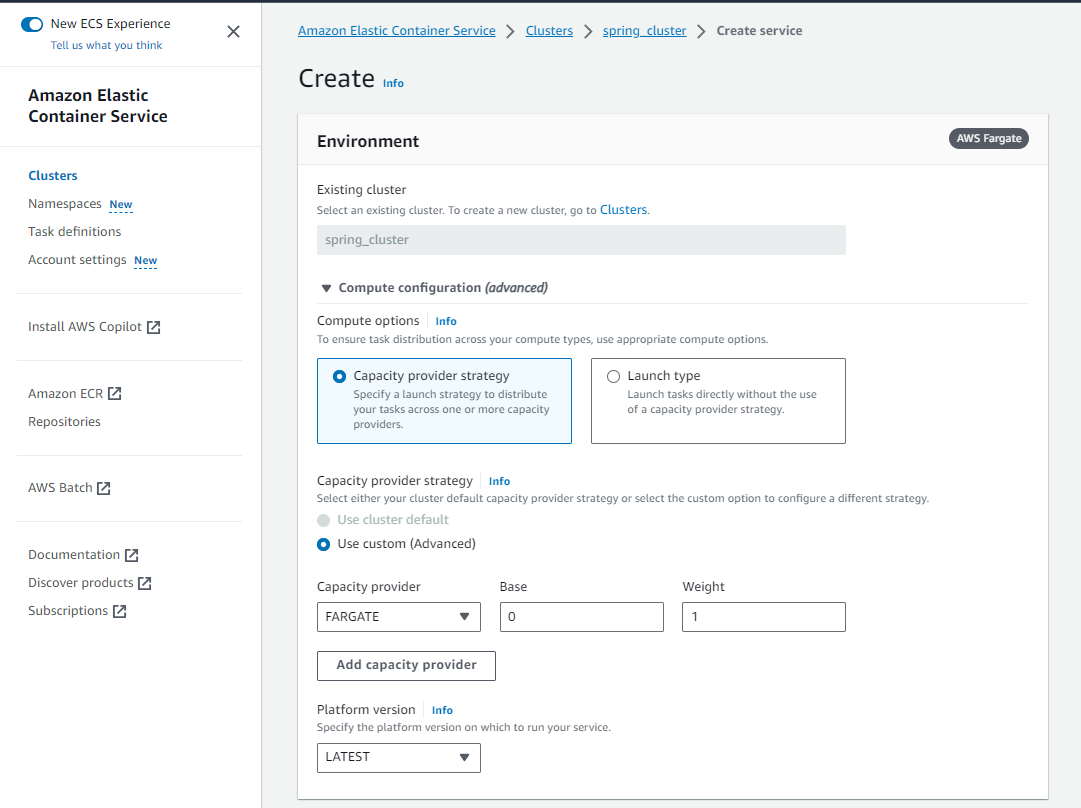


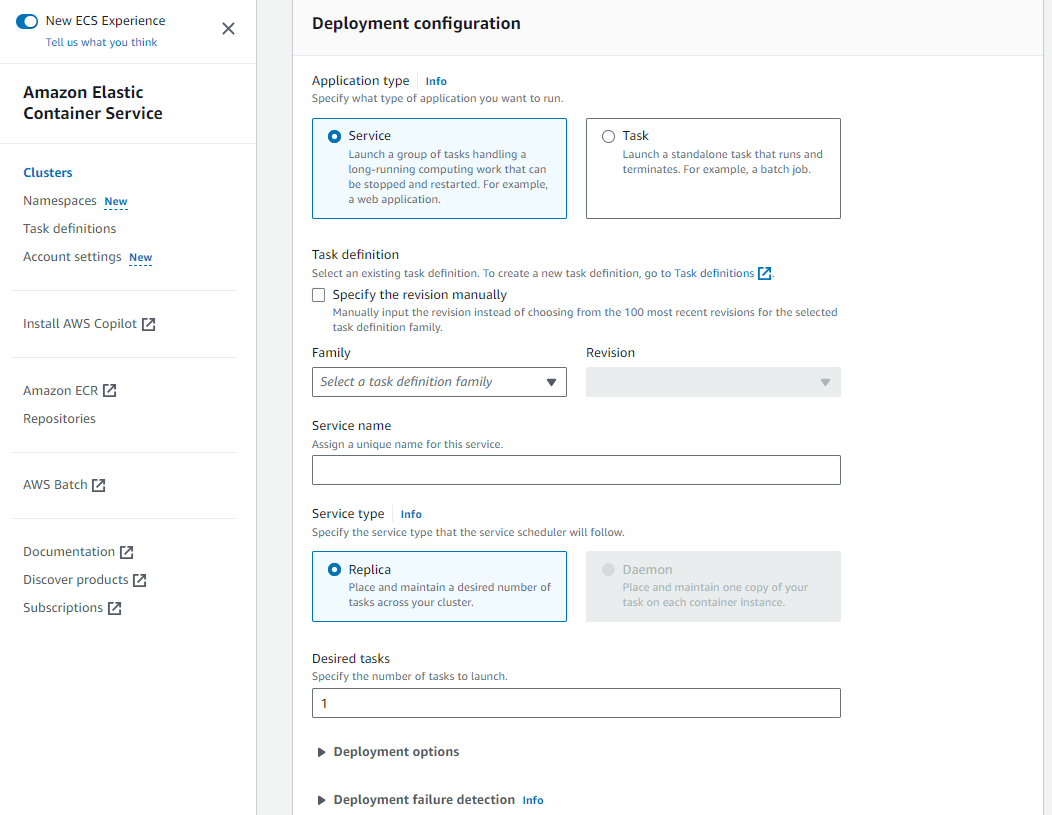


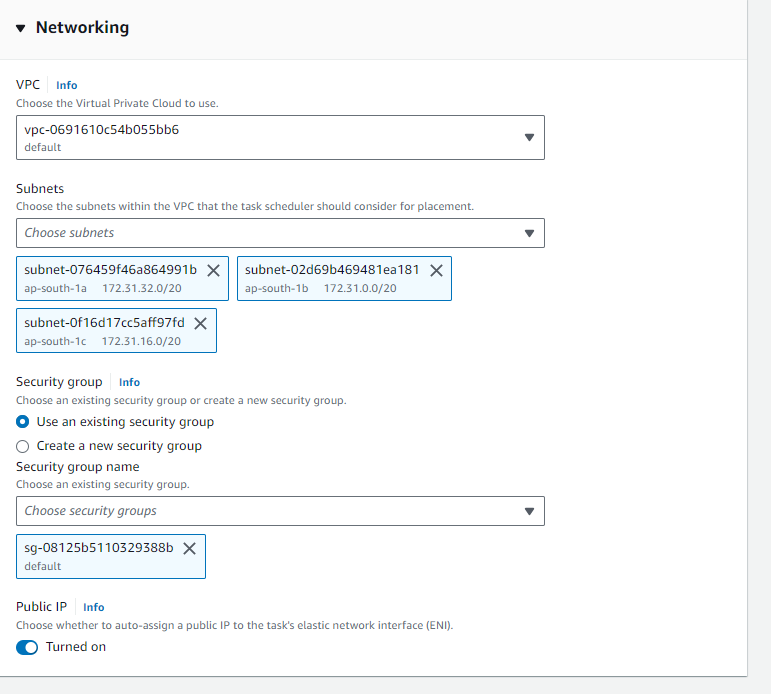
Also need to configure with Github only for specific repo using GitHub Access Token

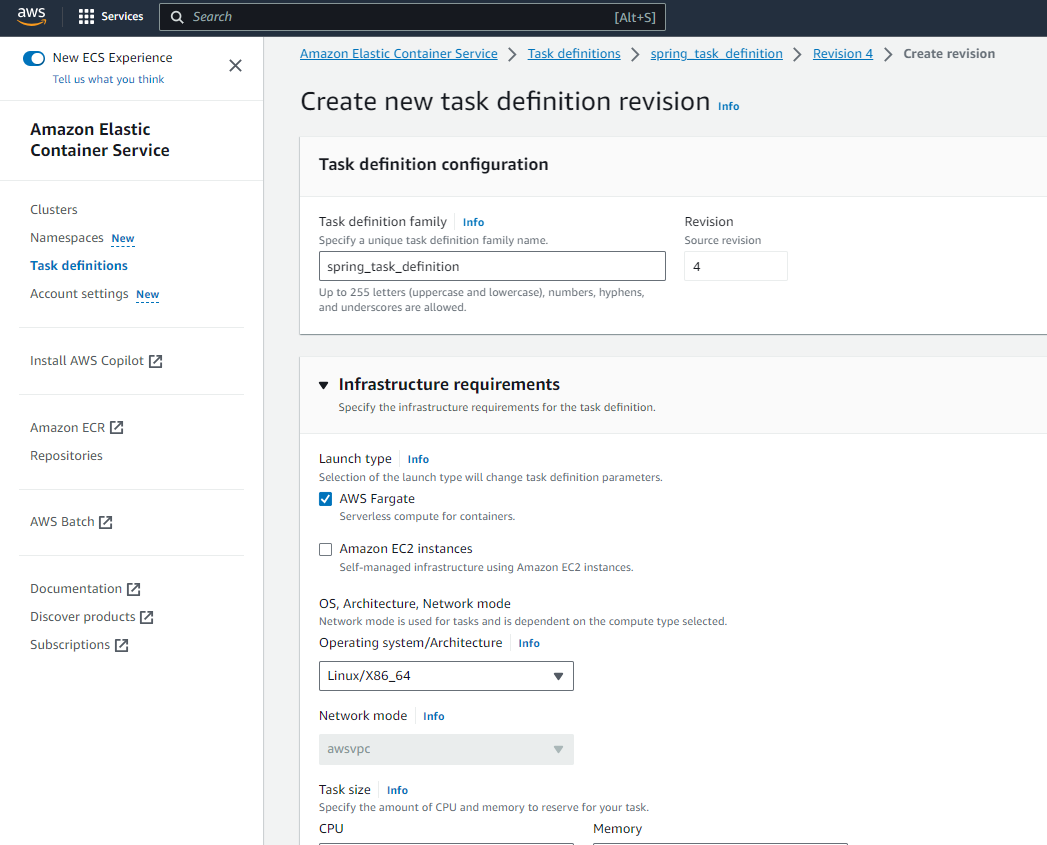
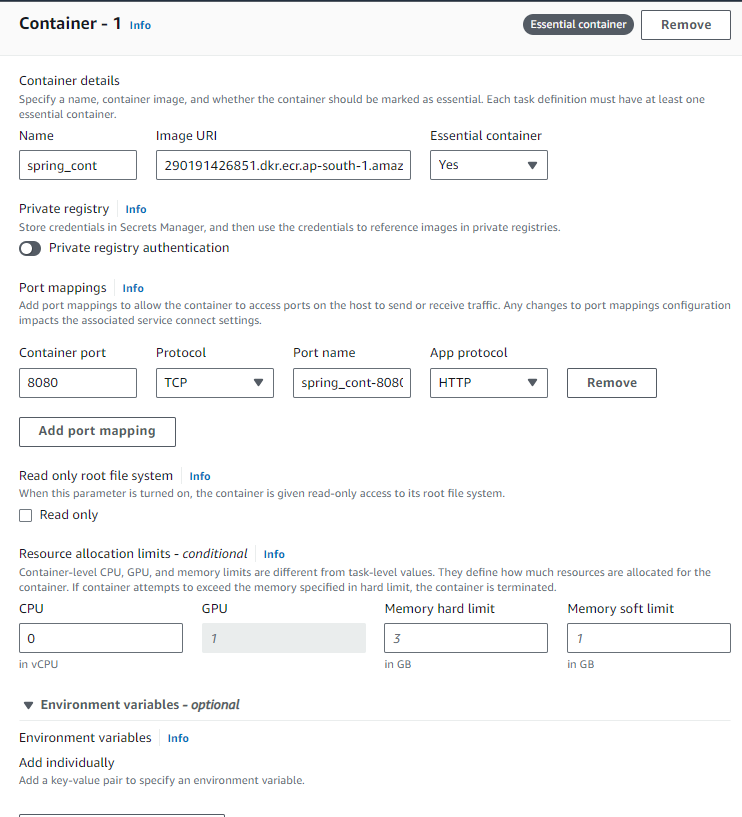
**STEP 9:**

\*Create ECS cluster with Task definition and Services  
\*And also add the actuator URL to define its status that it is healthy or not.

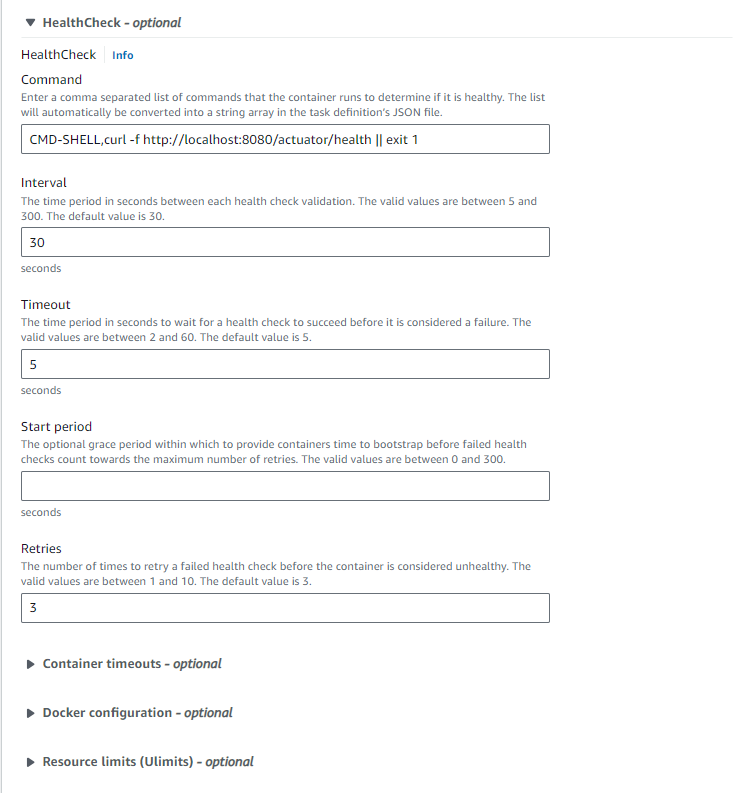




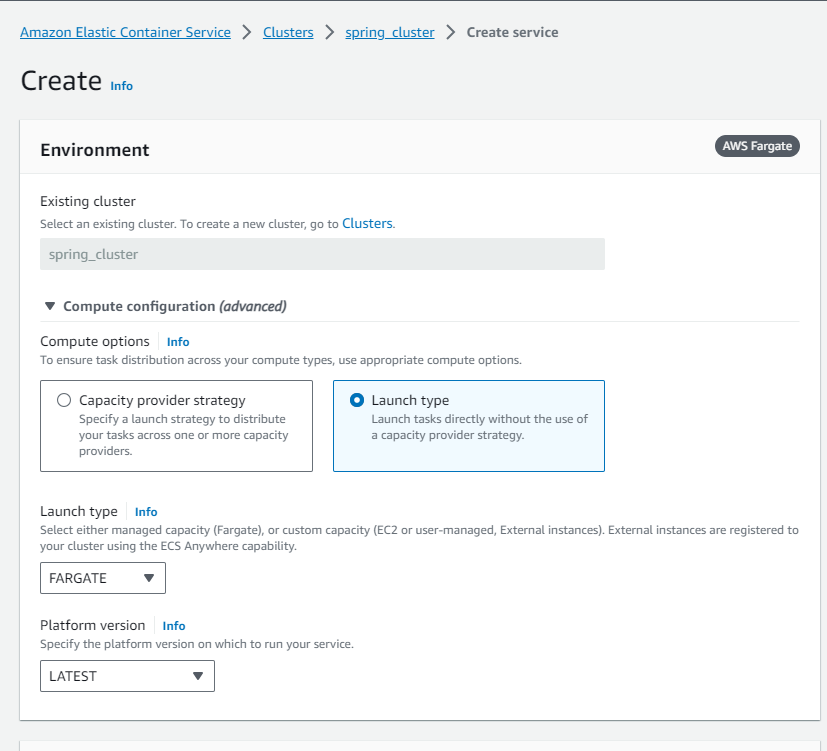


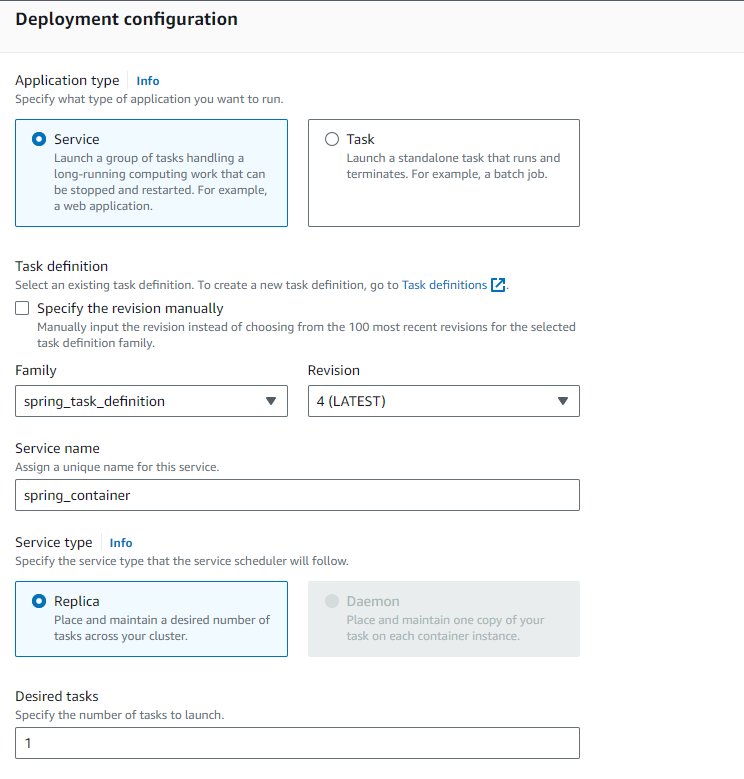
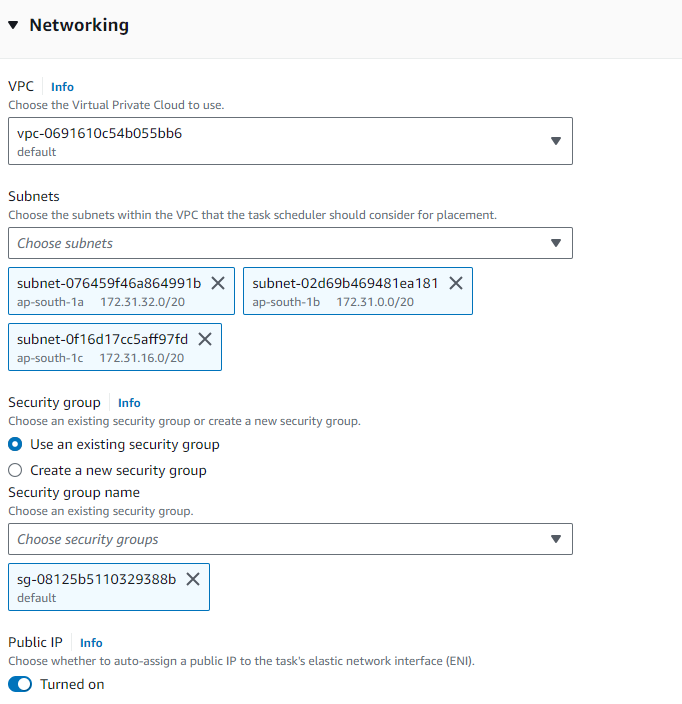
In Task Definition,  
  


For HealthCheck URL we need define it as localhost

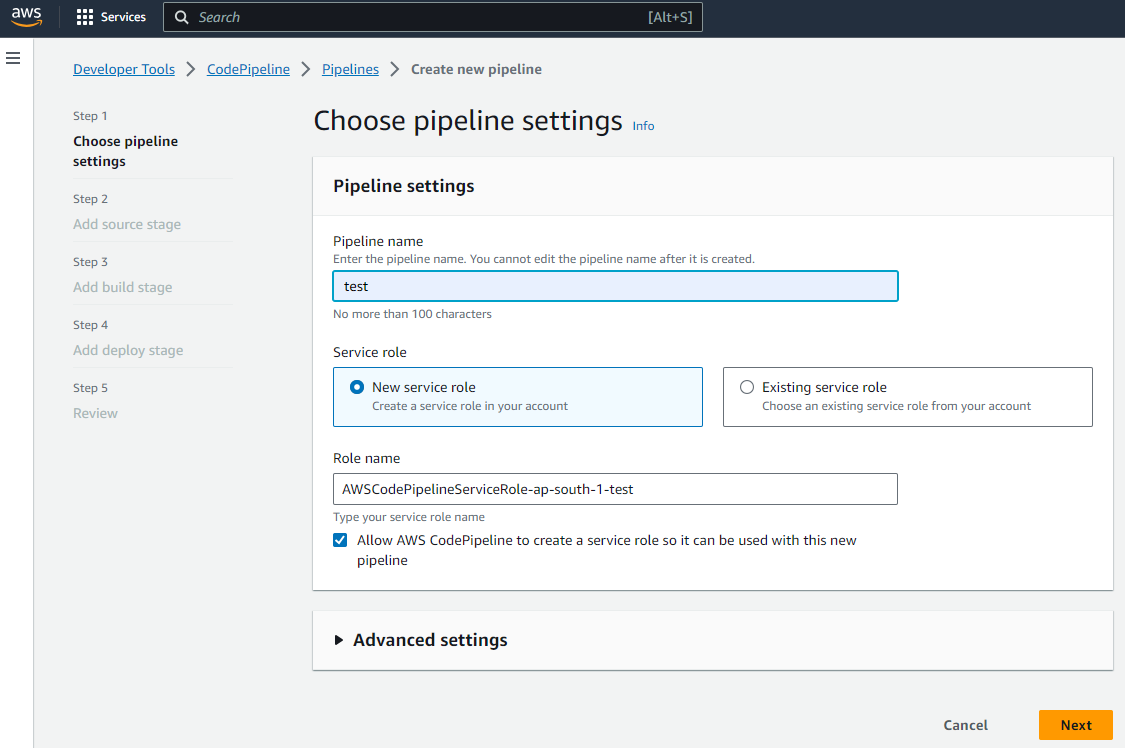


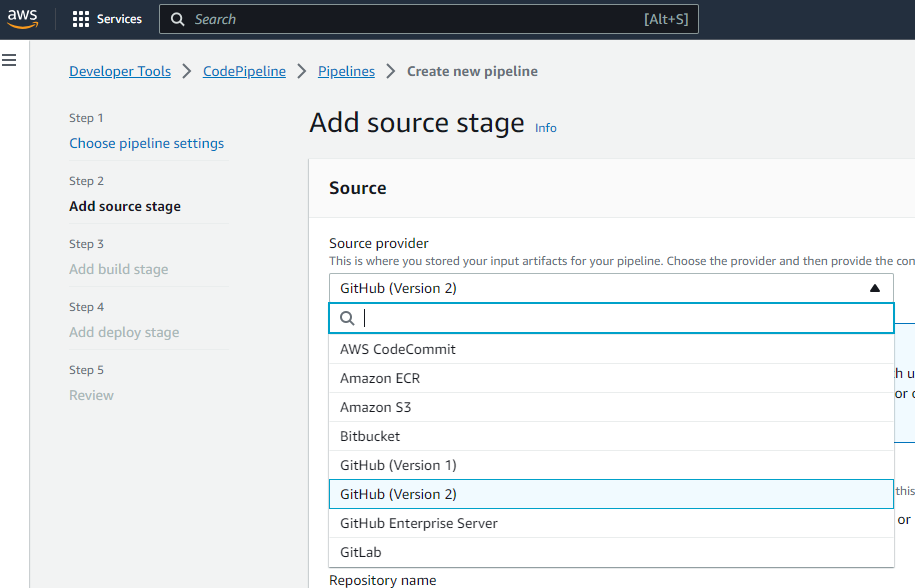
Then click create

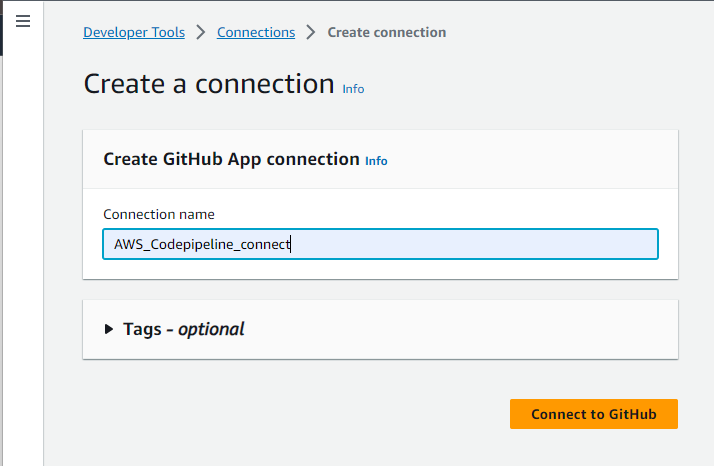
Need to create service inside cluster to run container  


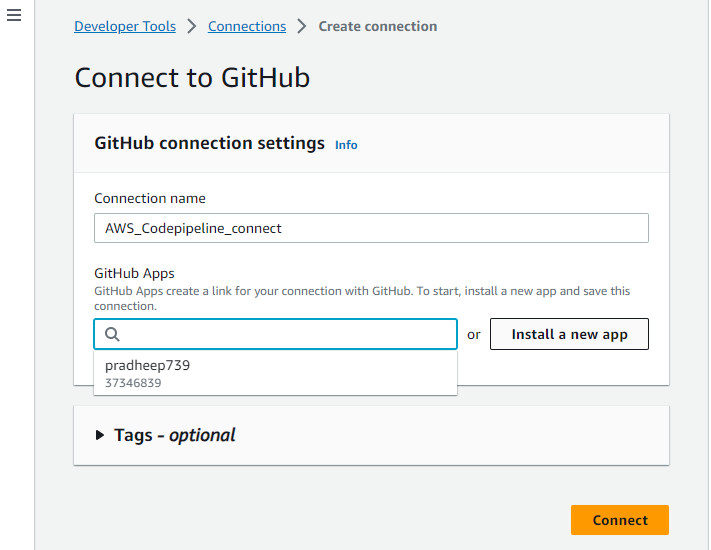
  
We need to turn on Public IP  


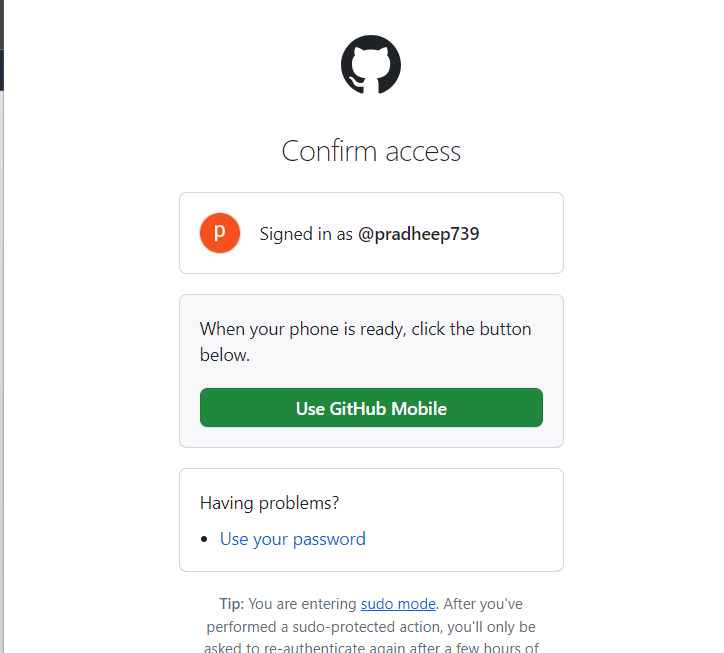
Then Create  
  
  
  
  
  
  
  
**Step 10:**

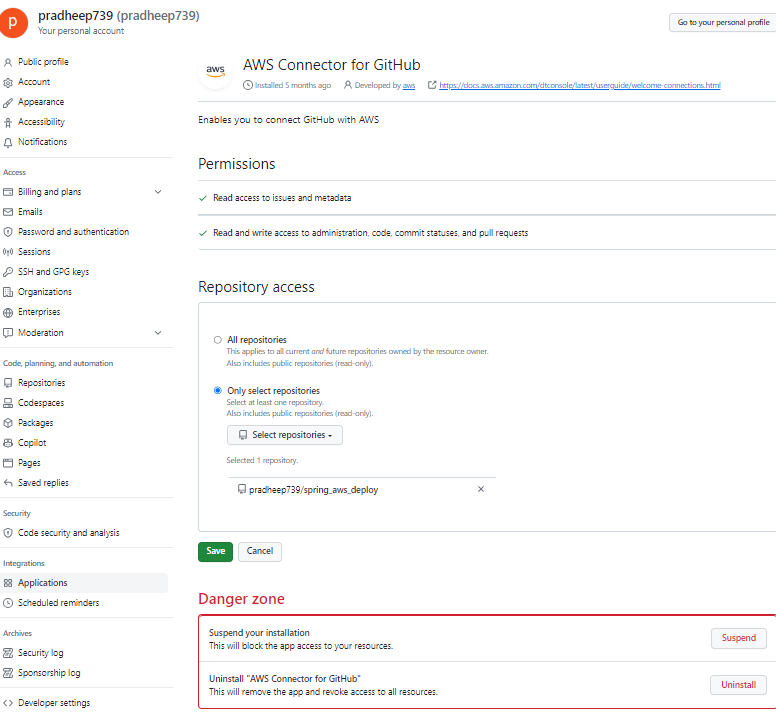
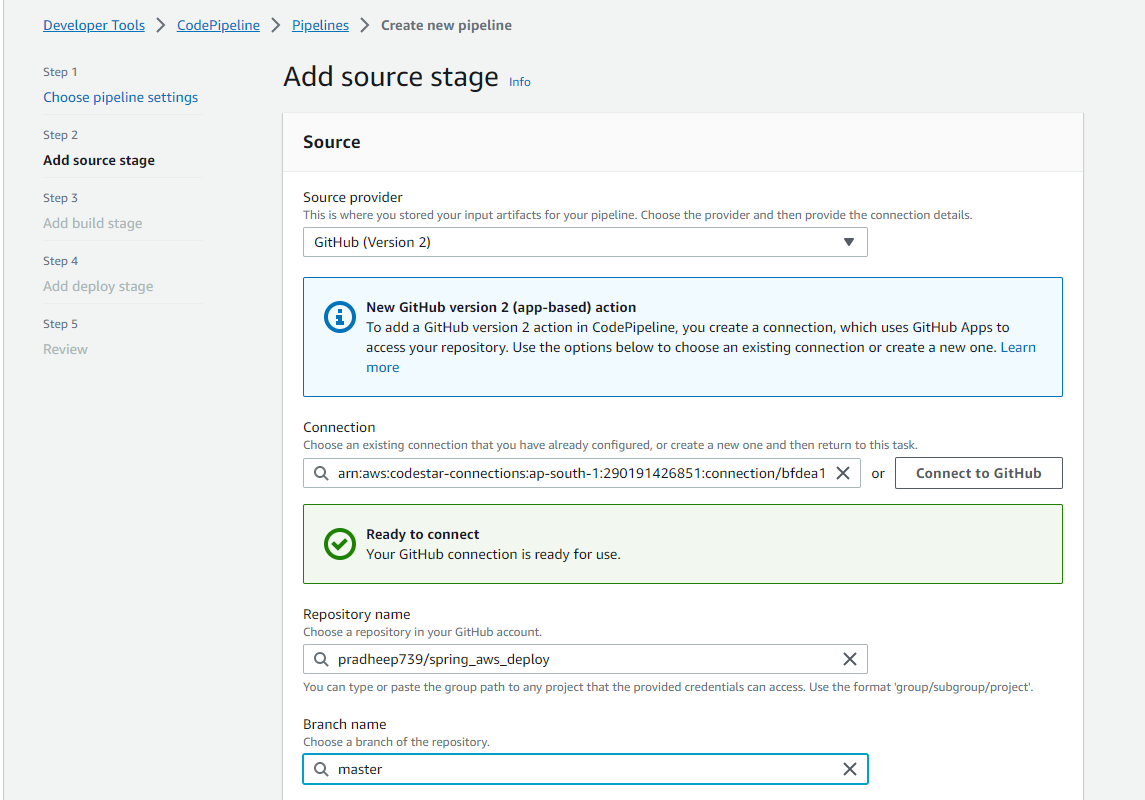
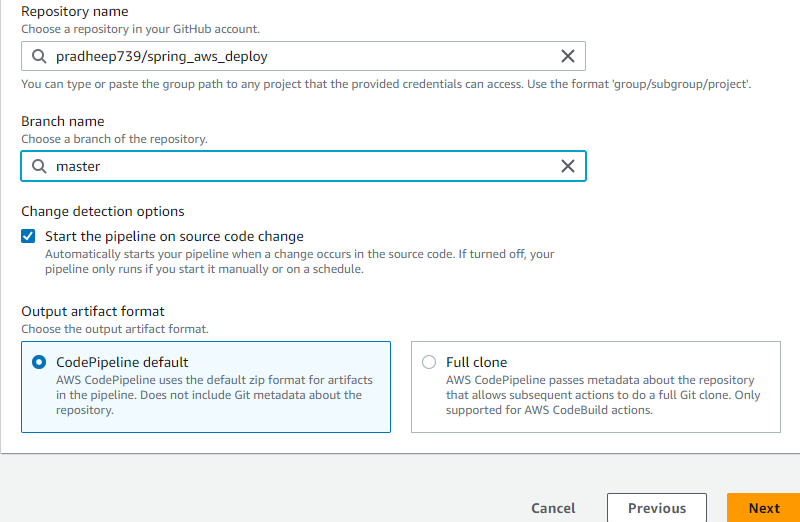
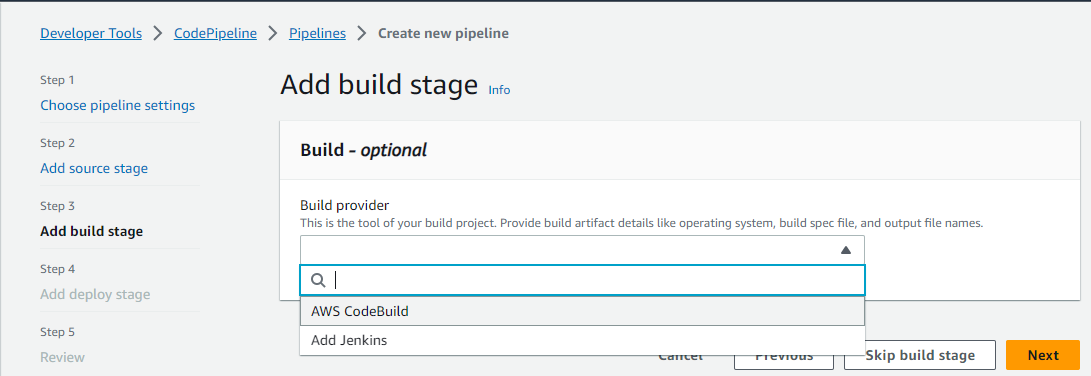
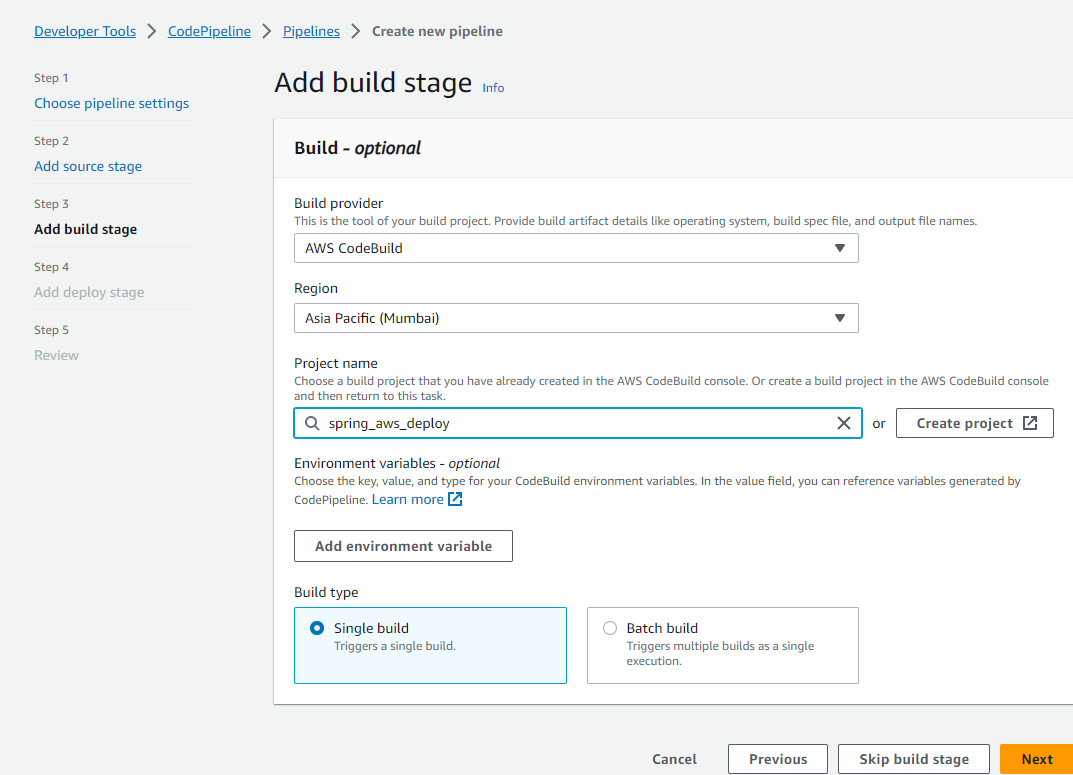
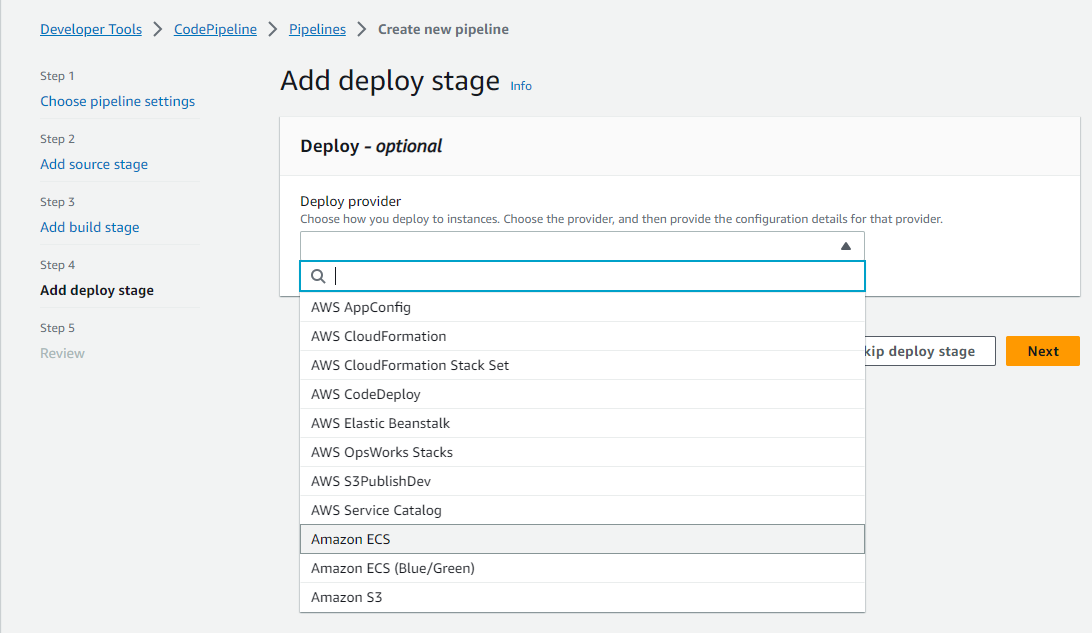
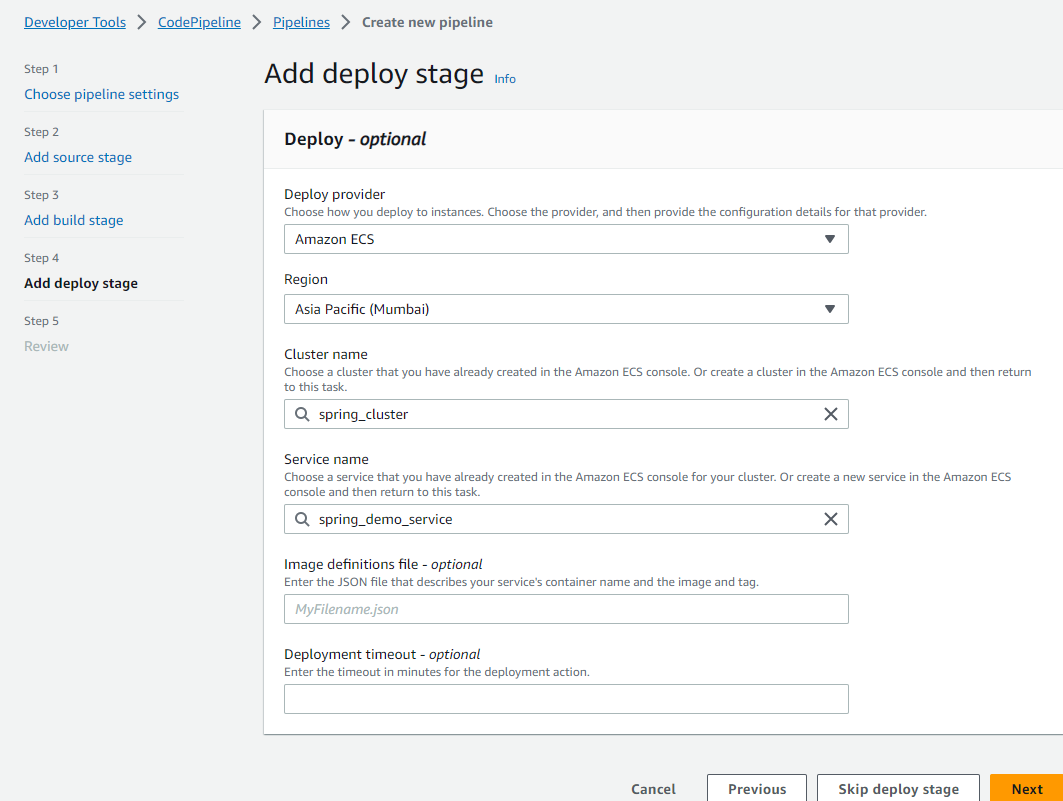
Create Code Pipeline so that our total deployment will be easy  
****

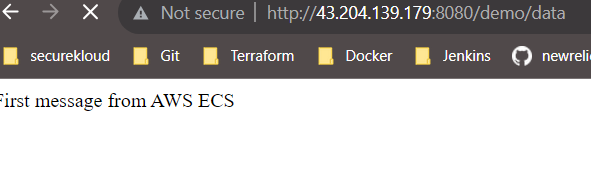
Creating it with default new Role defined by Code Pipeline  
****

****

Select install New app for first time  
****

Need to utheticate with GitHub  
****

****After Installing New App then click connect  
****  ****   

Then Review the things and Click Create Pipeline  
It will automatically build the trigger  
Since this method will run on every changes made in the master branch  
  
Output:  
  
  
  
Reference:  
[**https://www.youtube.com/watch?v=ARGmrYFfv44&ab\_channel=kodEdge**](https://www.youtube.com/watch?v=ARGmrYFfv44&ab_channel=kodEdge)[**https://github.com/kodedge-swapneel/springboot-aws-deploy**](https://github.com/kodedge-swapneel/springboot-aws-deploy)