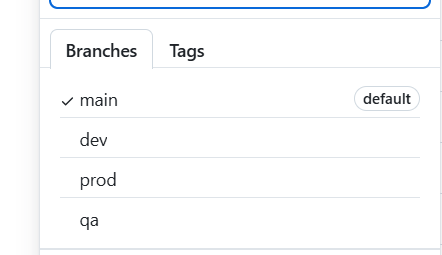
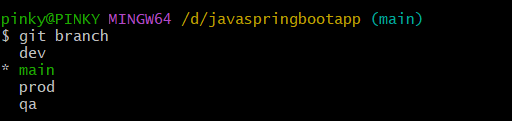
**Case Study-1:**

**Problem Statement:** You have been asked to set up a multi-environment **declarative/yaml based** deployment pipeline for a web application running on Elastic Kubernetes Service (EKS). A company controls the code for an application in **source code repository**. Containerized the web application using Docker and captured vulnerabilities by **image scanning** before pushing to registry. The application needs to be deployed **automatically to multiple EKS** environments (dev, qa, prod etc.) whenever changes are made to the respective branch code. Changes occur frequently every day, so the pipeline must be as responsive as possible. Prod pipeline must wait for **manual approval** before deployment. You are expected to integrate various DevOps/DevSecOps tools to perform given activities - **build management, test coverage, code coverage** etc and execute code and test coverage stages in **parallel manner**. Also trigger **notification** if coverage is less than 80% or any critical observation found and **terminate** the pipeline.

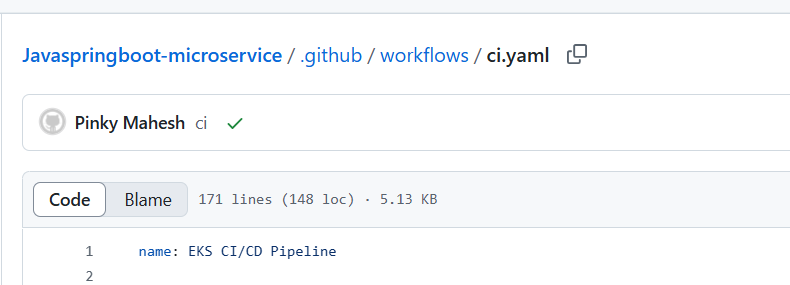
**GitHub Repo Structure:**





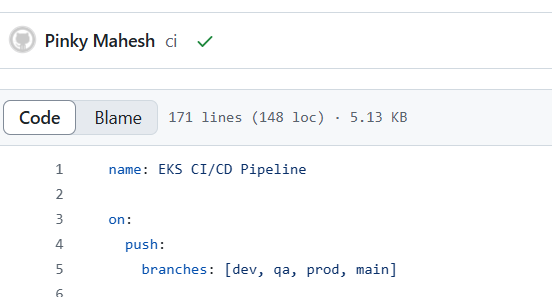
**\*Used GitHub actions as CICD tool and also used GitHub hosted runners for that.**

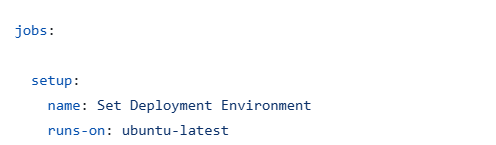
**2. GitHub Actions Workflow File**



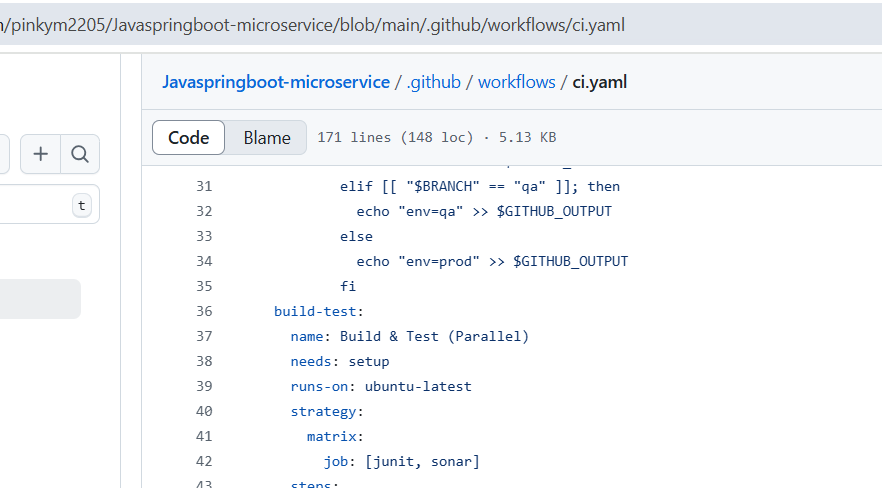
**Used GitHub action cicd pipeline to configure Multi-environment pipeline**

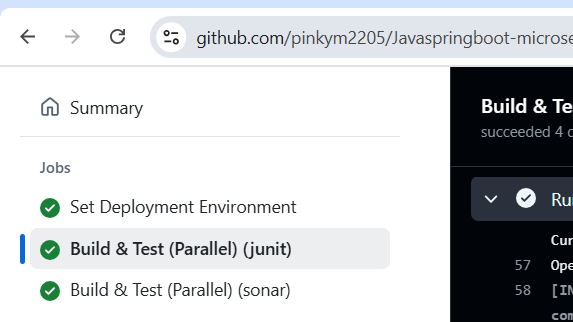
**my .github/workflows/ci.yaml contains:**



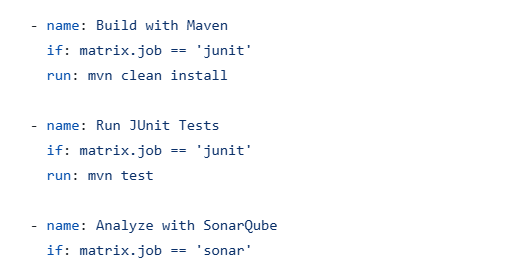


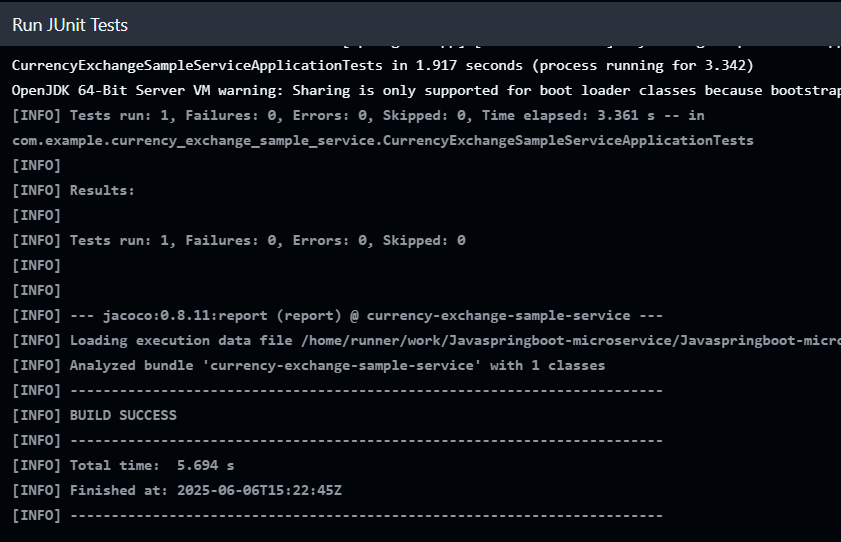
**Paralell execution of Junit test and Sonar**



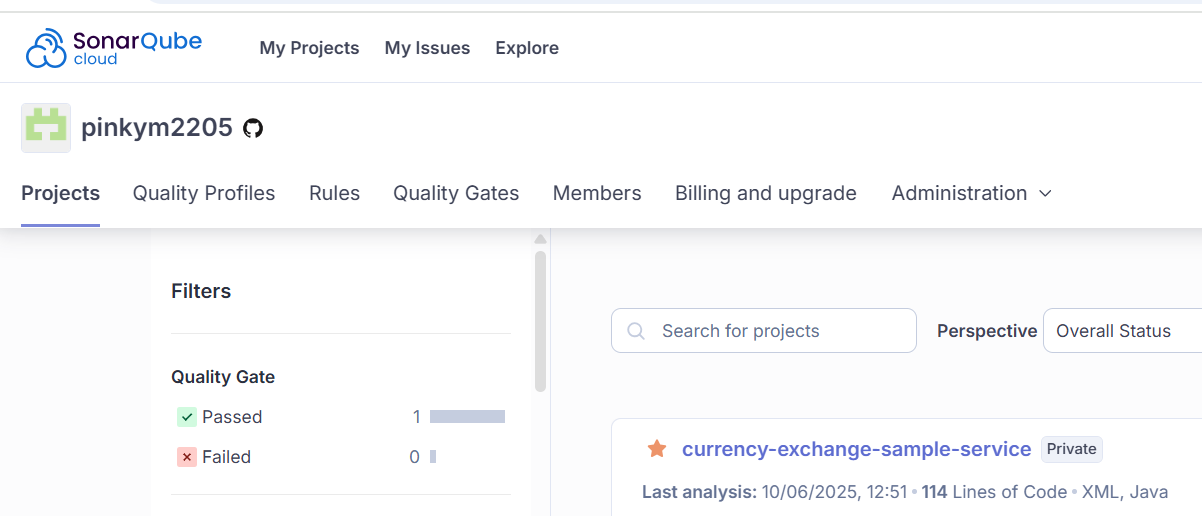


In a single job definition; used the matrix to split the tasks; Run both tasks in parallel to reduce total pipeline time





Sonar cloud setup:



Logged into <https://sonarcloud.io>

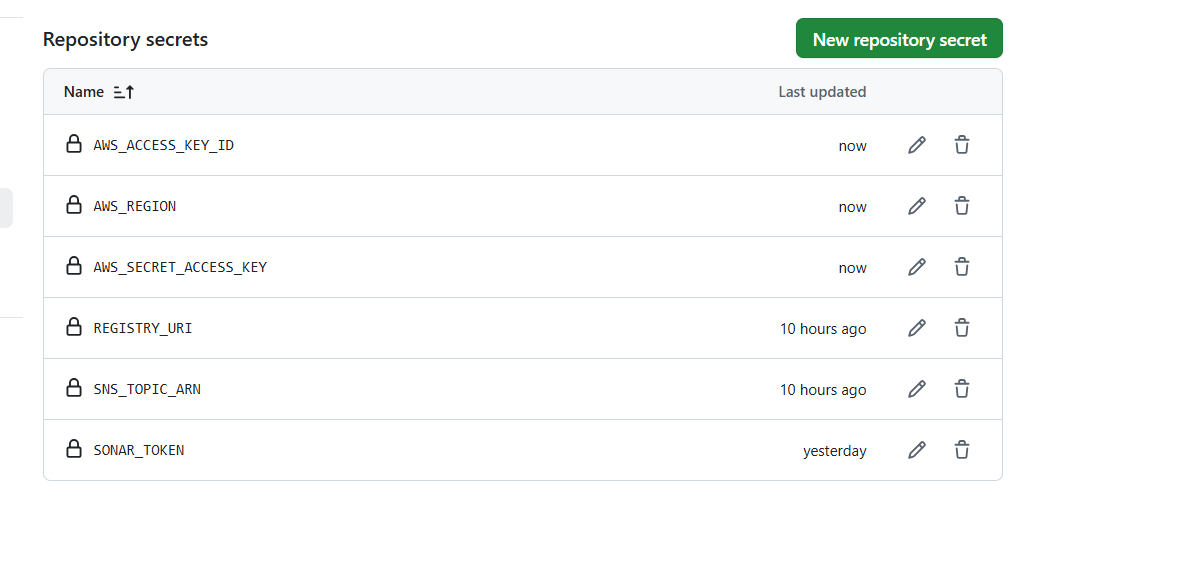
Created my Project on SonarCloud

Generated a SonarCloud Token

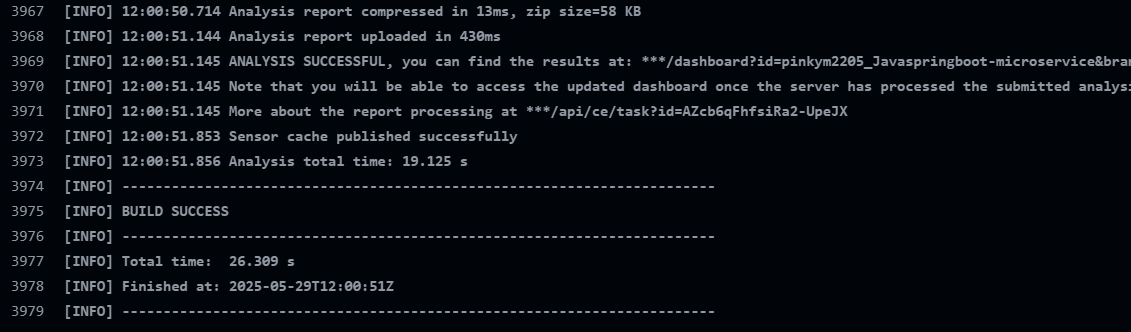
Added the Token to GitHub Repo Secrets

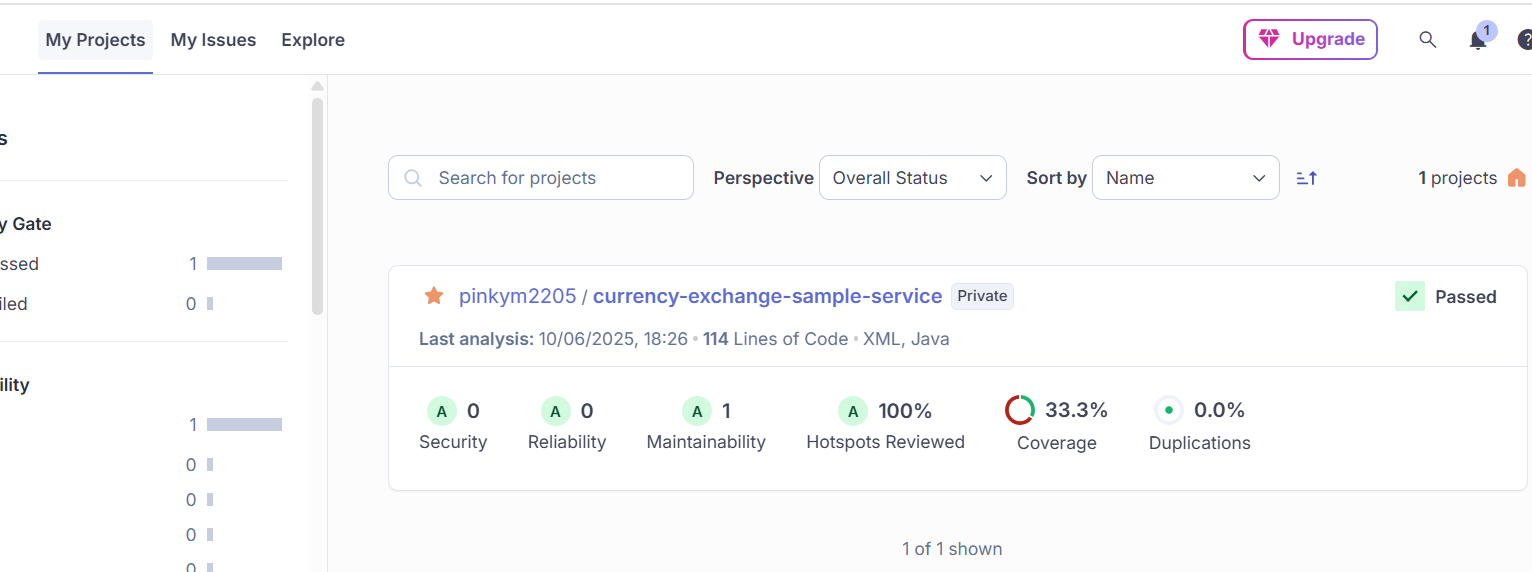
Add sonar-project.properties File to Your Repo

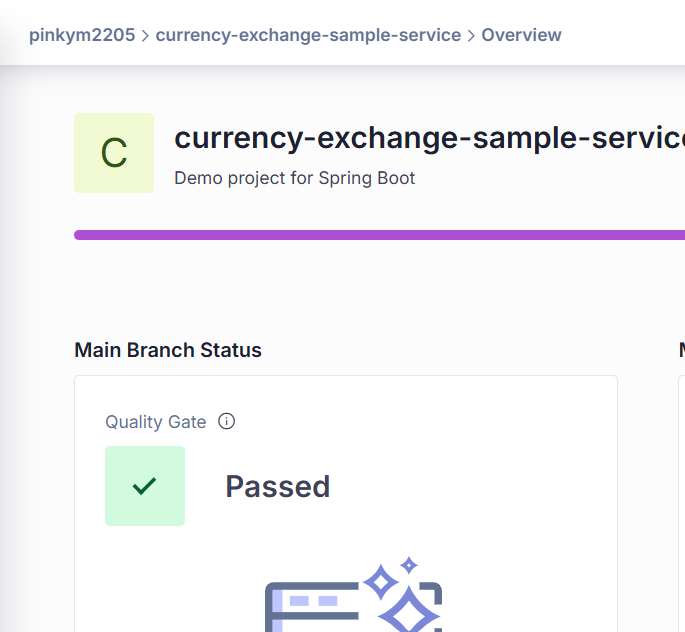
Made sure my pom.xml has the Sonar plugin defined

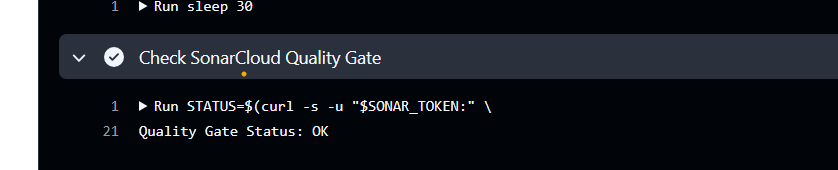
****

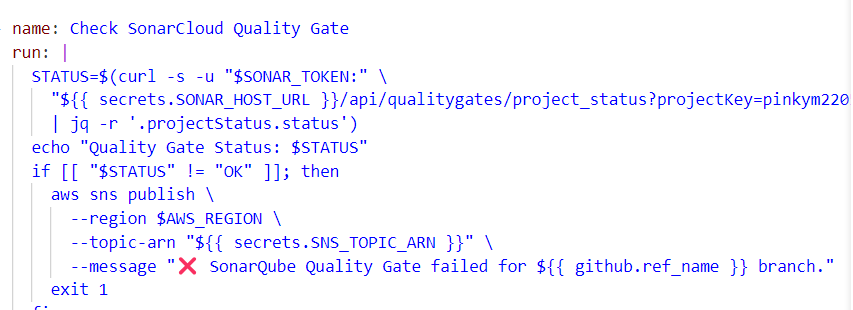


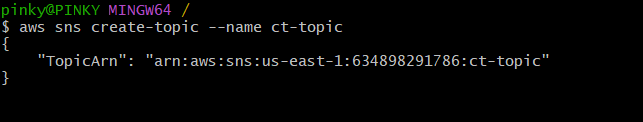


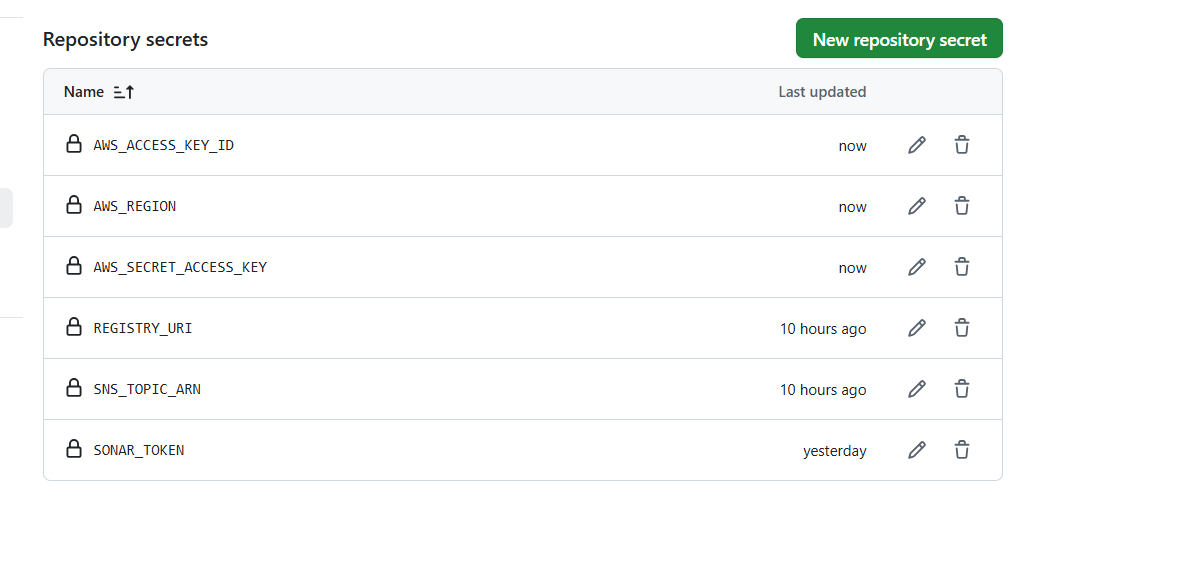


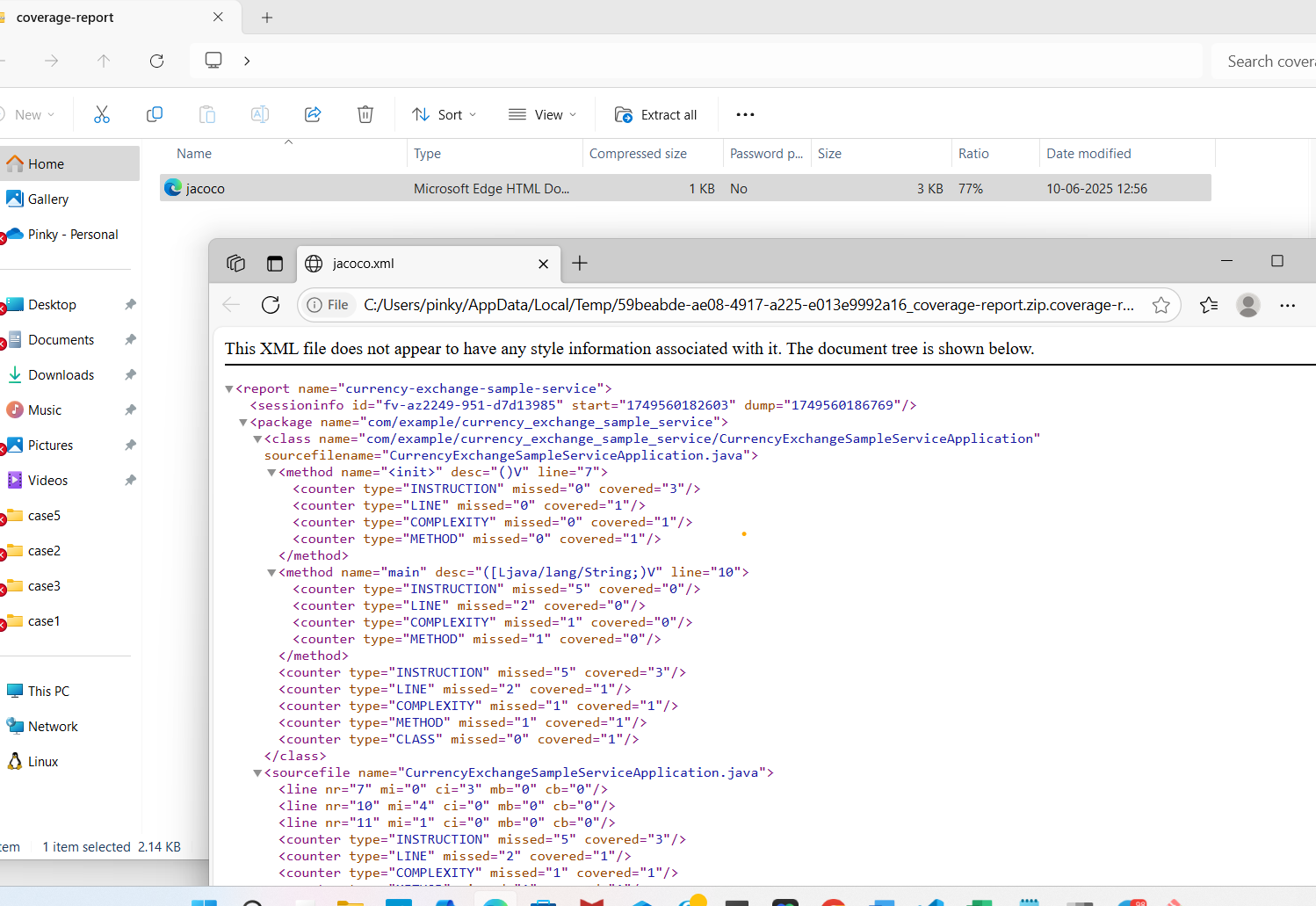




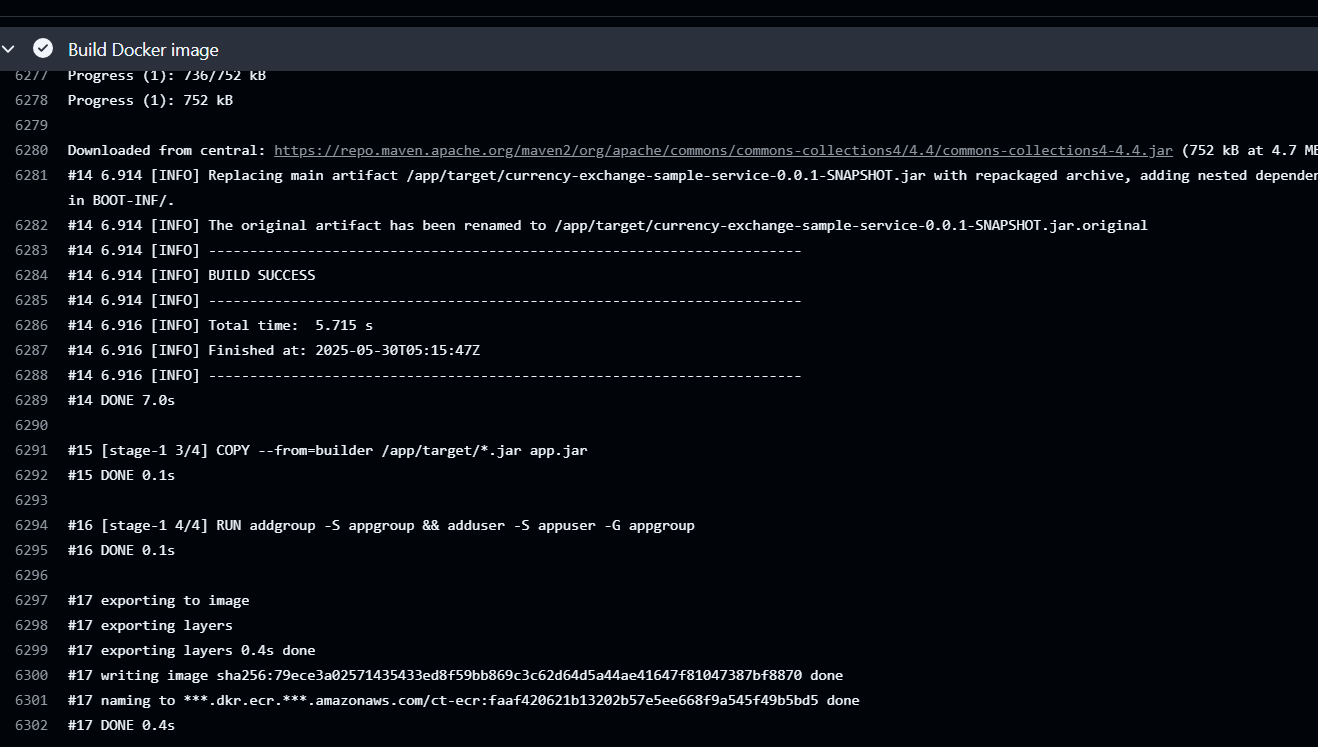


****

****

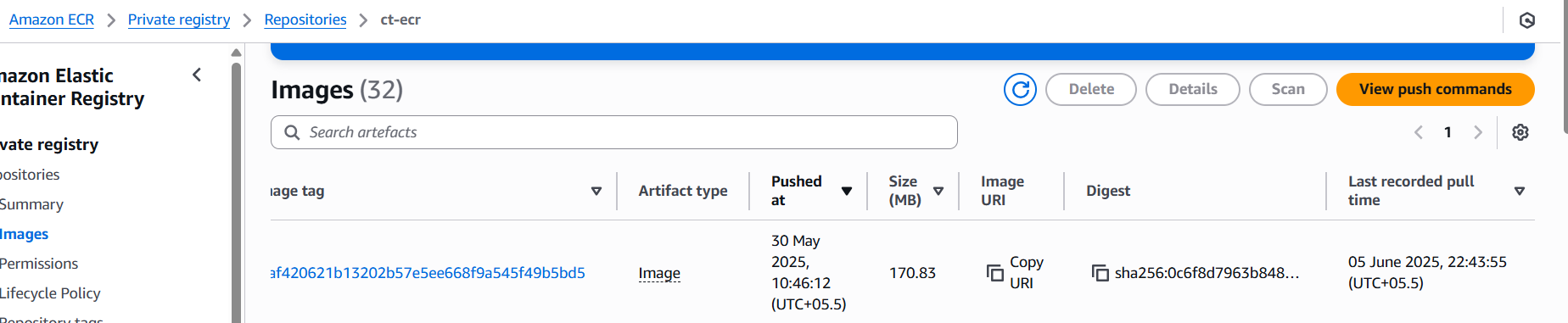


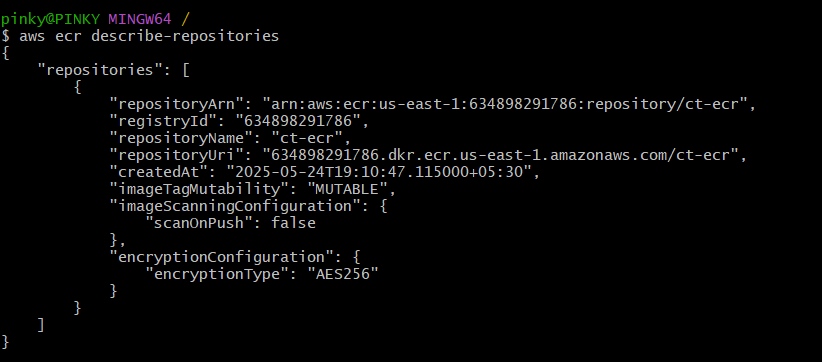
**Next Stage is docker Build and Push to ECR**



**Verify in AWS ECR Console.**

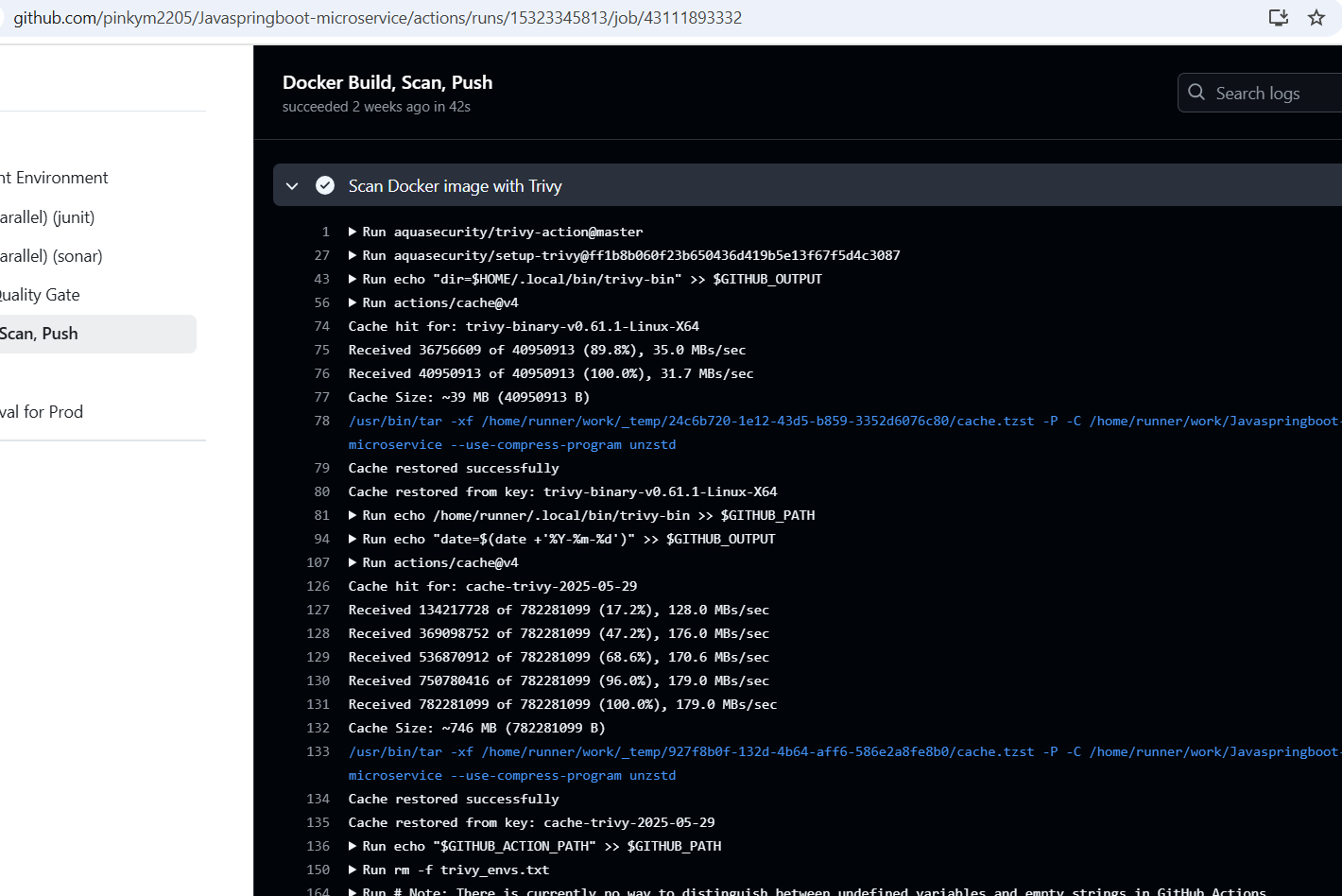
You should see the new image tagged with:

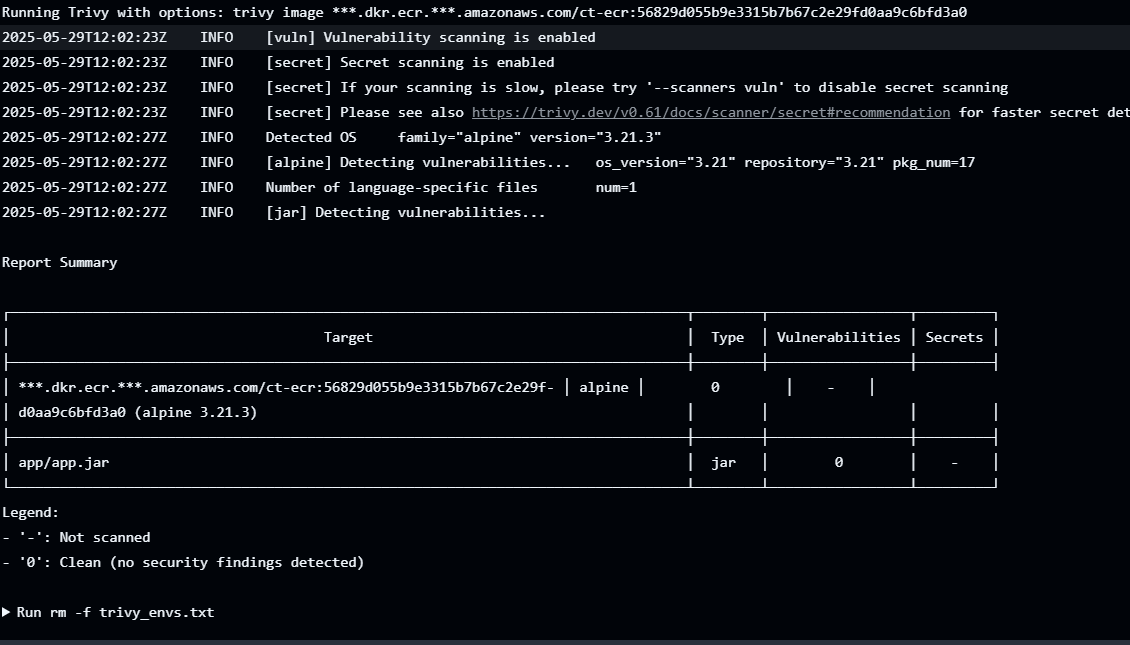


****

**Trivy step**

* **Pulls your Docker image** from ECR using the tag ${{ github.sha }}
* **Installs and runs Trivy** locally within the runner
* **Scans the image** for vulnerabilities (OS packages, JARs, etc.)
* **Prints a report** to the console



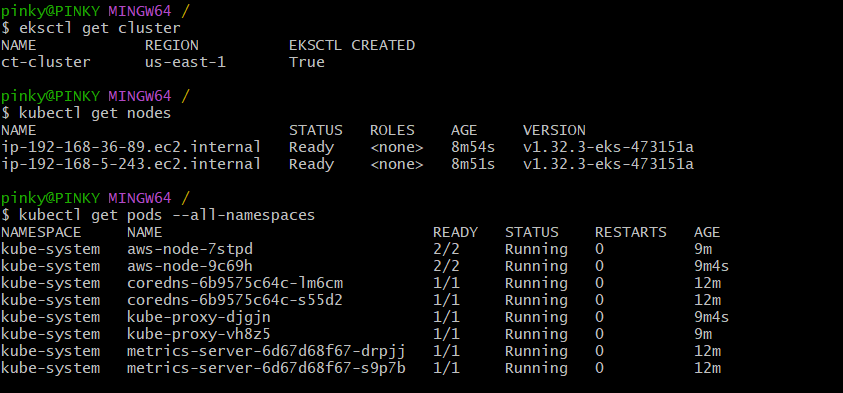


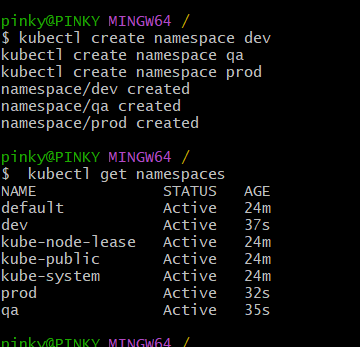
│ \*\*\*.dkr.ecr.\*\*\*.amazonaws.com/ct-ecr:... │ alpine │ 0 │ - │

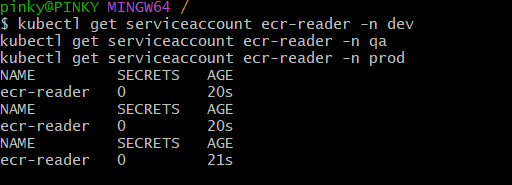
│ app/app.jar │ jar │ 0 │ - │

Vulnerabilities = 0: The Alpine base image and JAR were **scanned** and are **clean**

Created EKS cluster

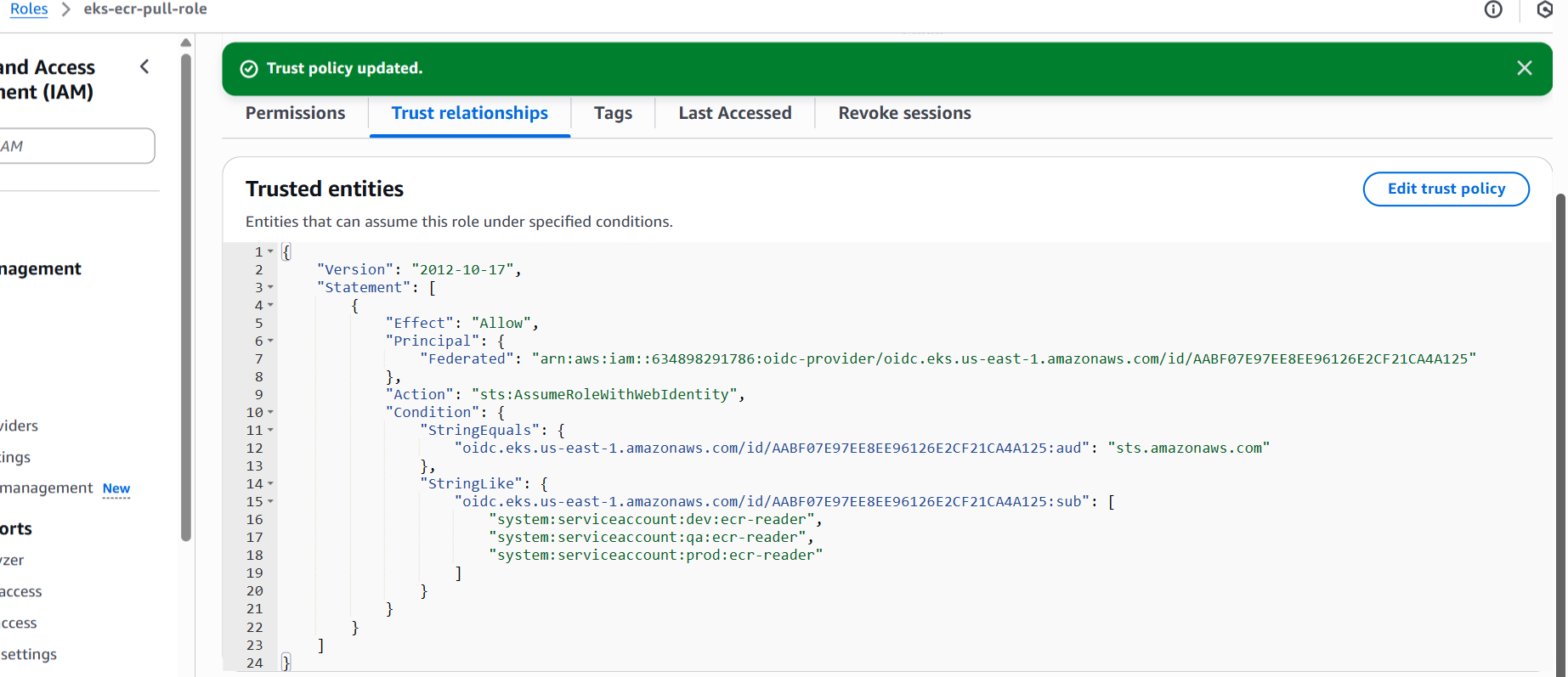


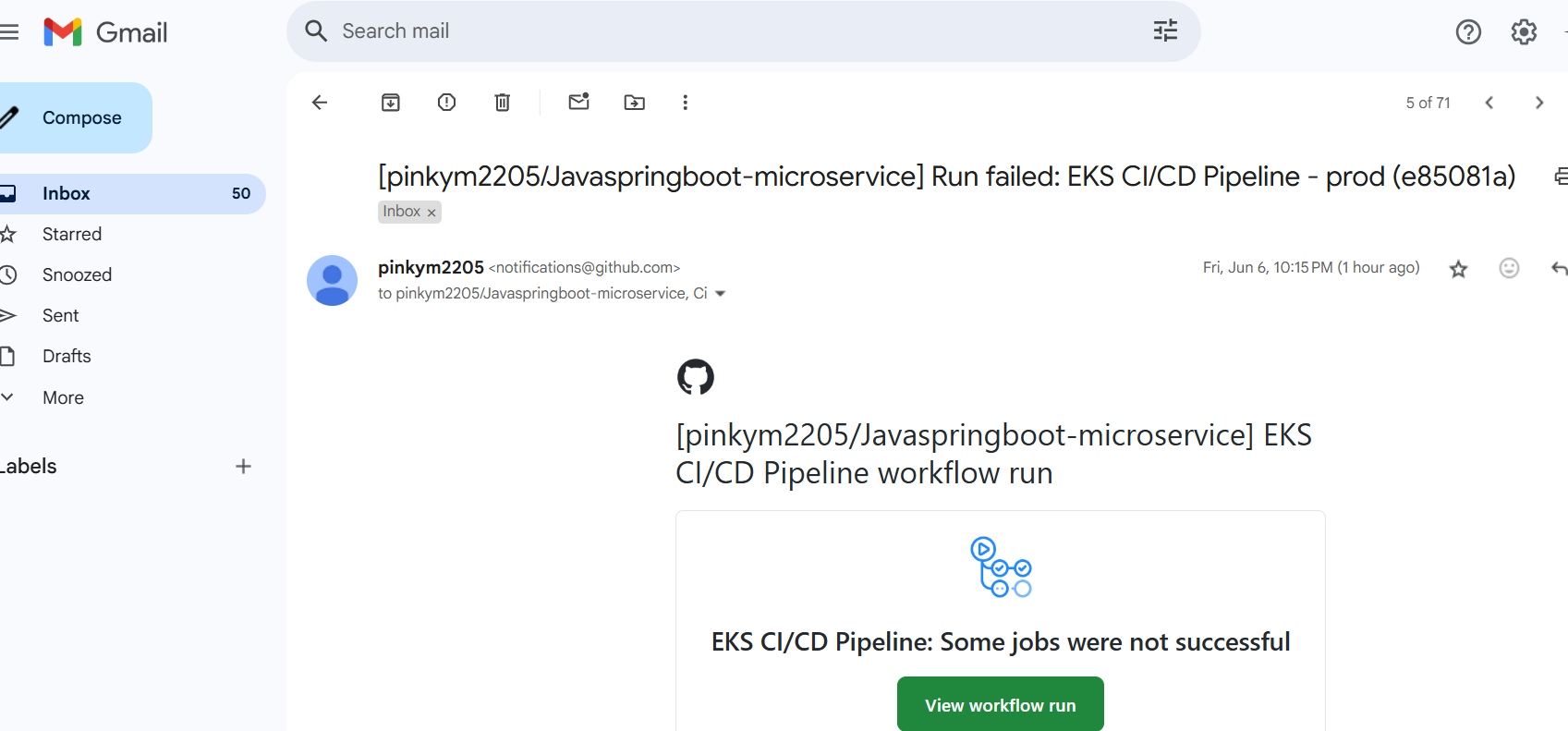


****

**IRSA ROLE NAME :eks-ecr-pull-role**

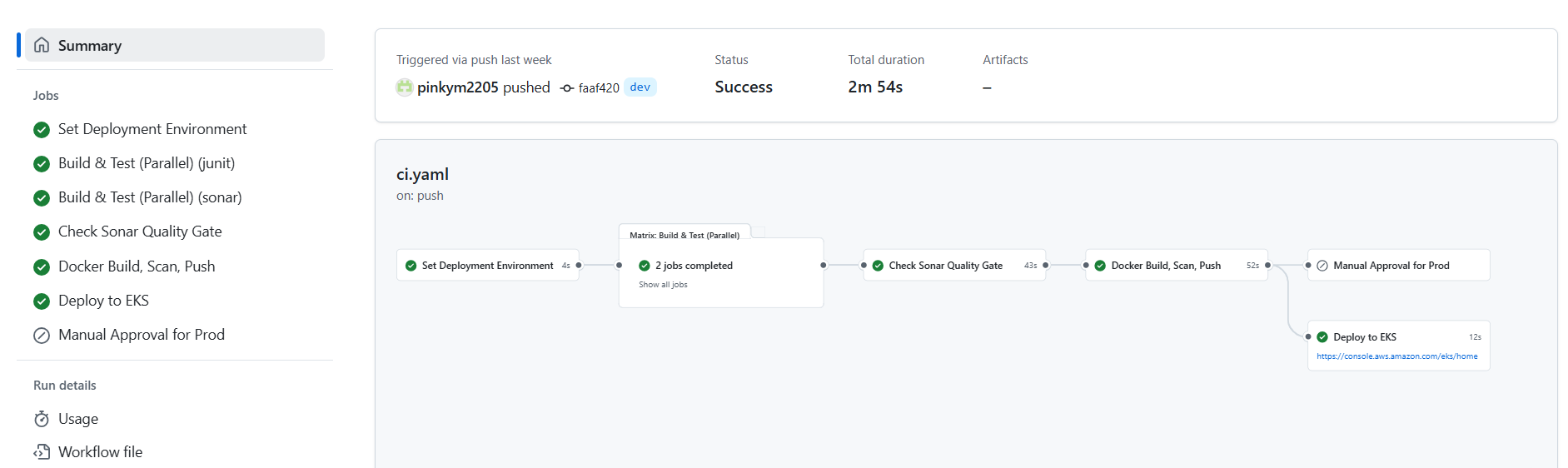
**SA name : ecr-reader (( for all namespaces same name)**

****

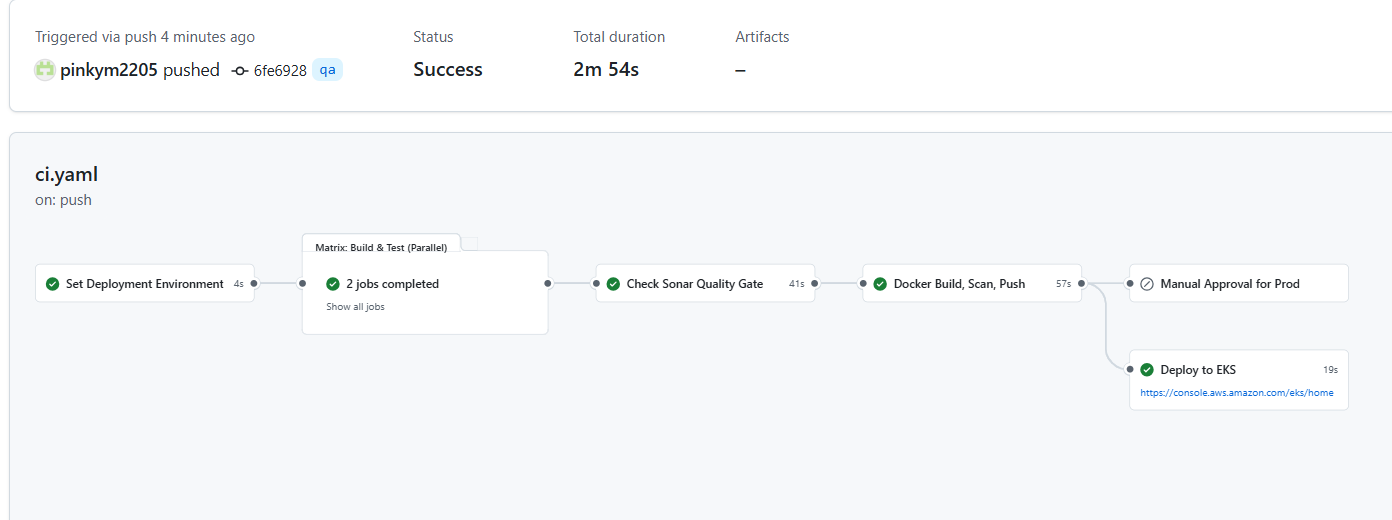


Deploy to EKS which was auto triggered by push event

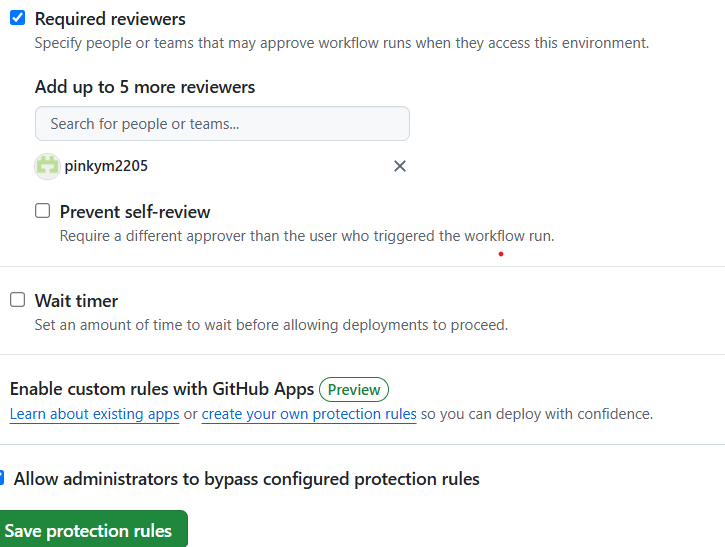
Below is screenshot of deployment to eks cluster dev namespace.



FOR QA namespace:

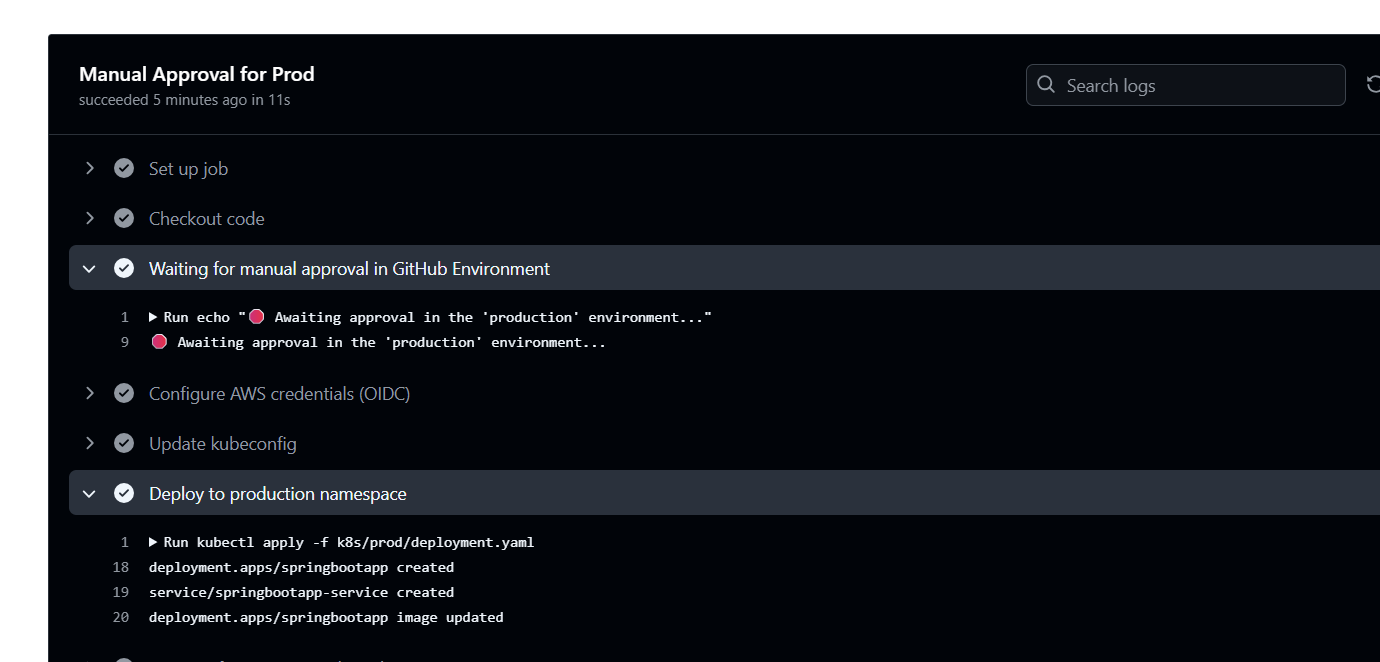


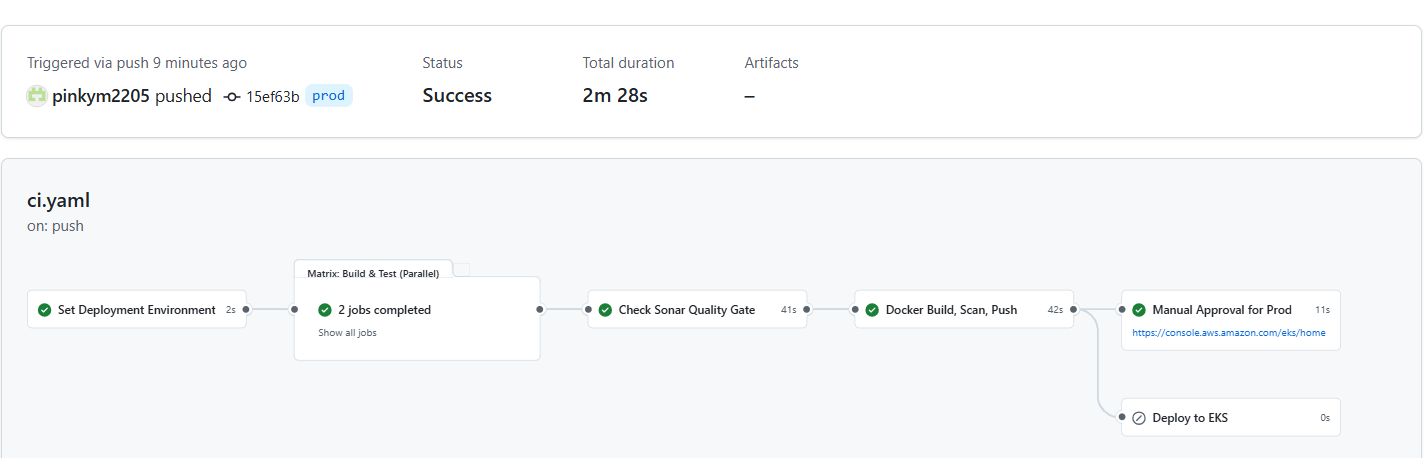
FOR Prod pipeline:

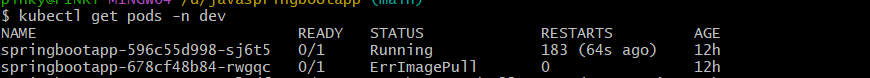










****

