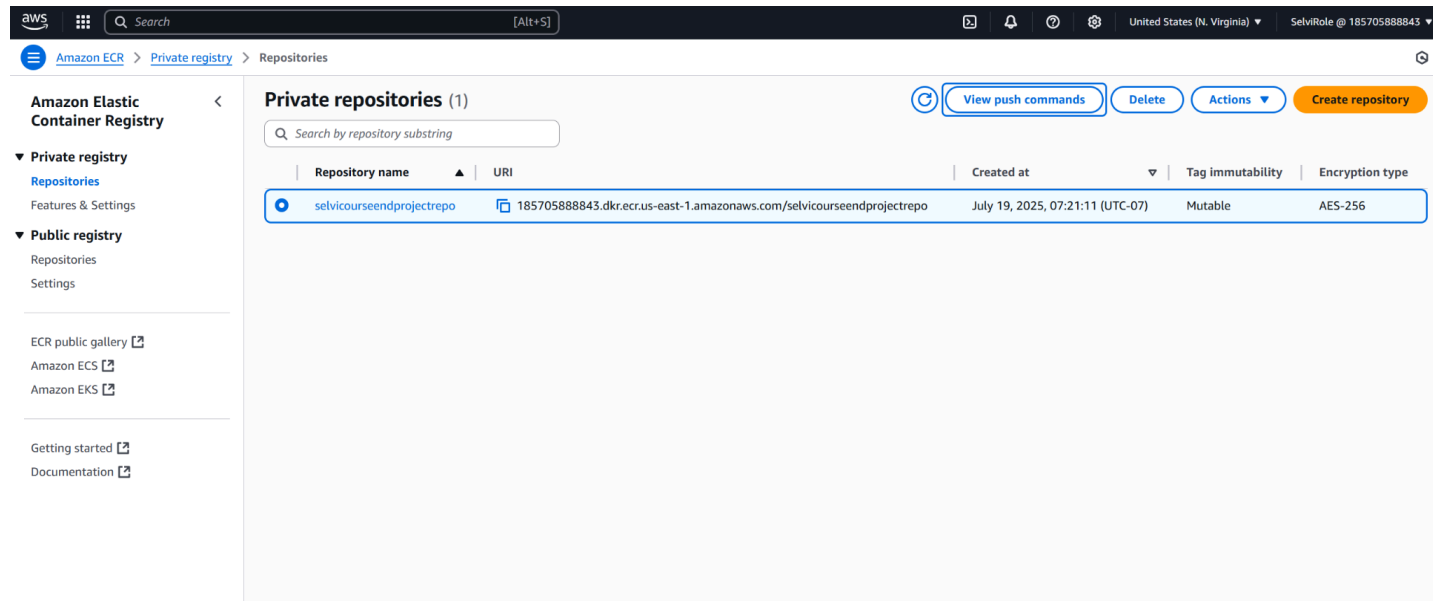
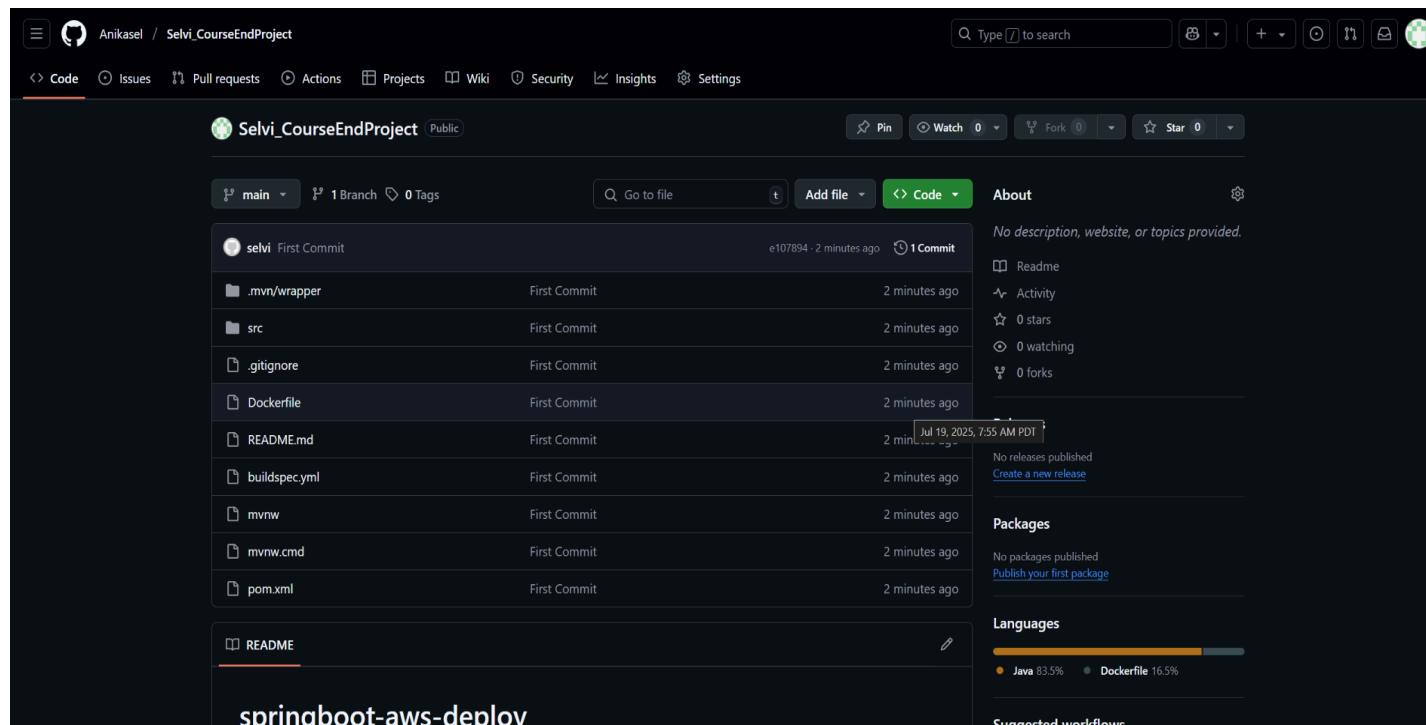


## Step 1: Setup an AWS Elastic Container Registry with a repository



## Step 2: Setup a GitHub repository and clone it to local machine



```
MINGW64/c/Selvi/CourseEndProject/Selvi_CourseEndProject

git config --global user.email "you@example.com"
git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'selvi@Selvi-TUF.(none)')

selvi@Selvi-TUF MINGW64 /c/Selvi/CourseEndProject/Selvi_CourseEndProject (main)
$ git config user.name "selvi"

selvi@Selvi-TUF MINGW64 /c/Selvi/CourseEndProject/Selvi_CourseEndProject (main)
$ git config user.email "selvi@abc.com"

selvi@Selvi-TUF MINGW64 /c/Selvi/CourseEndProject/Selvi_CourseEndProject (main)
$ git commit -m "First Commit"
[main (root-commit) e107894] First Commit
13 files changed, 708 insertions(+)
create mode 100644 .gitignore
create mode 100644 .mvn/wrapper/maven-wrapper.jar
create mode 100644 .mvn/wrapper/maven-wrapper.properties
create mode 100644 Dockerfile
create mode 100644 README.md
create mode 100644 buildspec.yml
create mode 100644 mvnw
create mode 100644 mvnw.cmd
create mode 100644 pom.xml
create mode 100644 src/main/java/com/example/springbootawsdeploy/SpringbootAwsDeployApplication.java
create mode 100644 src/main/java/com/example/springbootawsdeploy/TestController.java
create mode 100644 src/main/resources/application.properties
create mode 100644 src/test/java/com/example/springbootawsdeploy/SpringbootAwsDeployApplicationTests.java

selvi@Selvi-TUF MINGW64 /c/Selvi/CourseEndProject/Selvi_CourseEndProject (main)
$ git push
Enumerating objects: 29, done.
Counting objects: 100% (29/29), done.
Delta compression using up to 16 threads
Compressing objects: 100% (20/20), done.
Writing objects: 100% (29/29), 64.44 KiB | 10.74 MiB/s, done.
Total 29 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Anikase1/Selvi_CourseEndProject.git
 * [new branch]      main -> main

selvi@Selvi-TUF MINGW64 /c/Selvi/CourseEndProject/Selvi_CourseEndProject (main)
$
```

### Step 3: Create a Code Build project

The screenshot shows the AWS CodeBuild console. On the left, the 'Developer Tools' sidebar is open, showing 'CodeBuild' under 'Build'. The main content area displays a green banner at the top stating 'Build started' and 'You have successfully started the following build: Selvi-CodeBuildProject:441d022e-eb52-44bf-9841-19682b0e16c2'. Below this, the breadcrumb navigation is 'Developer Tools > CodeBuild > Build projects > Selvi-CodeBuildProject > Selvi-CodeBuildProject:441d022e-eb52-44bf-9841-19682b0e16c2'. The main heading is 'Selvi-CodeBuildProject:441d022e-eb52-44bf-9841-19682b0e16c2'. Under 'Build status', the status is 'Succeeded'. The initiator is 'SelviRole/odl\_user\_1798140'. The build ARN is 'arn:aws:codebuild:us-east-1:18570588843:build/Selvi-CodeBuildProject:441d022e-eb52-44bf-9841-19682b0e16c2'. The start time is 'Jul 19, 2025 8:14 AM (UTC-7:00)' and the end time is 'Jul 19, 2025 8:15 AM (UTC-7:00)'. The build number is '1'. Below the status, there are tabs for 'Build logs', 'Phase details', 'Reports', 'Environment variables', 'Build details', and 'Resource utilization'. The 'Phase details' tab is selected, showing a table with columns: Name, Status, Context, Duration, and Start time. The table has three rows: 'SUBMITTED' (Succeeded, <1 sec, Jul 19, 2025 8:14 AM (UTC-7:00)), 'QUEUED' (Succeeded, <1 sec, Jul 19, 2025 8:14 AM (UTC-7:00)), and 'PROVISIONING' (Succeeded, 4 secs, Jul 19, 2025 8:14 AM (UTC-7:00)).

**Build status**

Property	Value
Status	✓ Succeeded
Initiator	SelviRole/odl_user_1798140
Build ARN	arn:aws:codebuild:us-east-1:18570588843:build/Selvi-CodeBuildProject:441d022e-eb52-44bf-9841-19682b0e16c2
Start time	Jul 19, 2025 8:14 AM (UTC-7:00)
End time	Jul 19, 2025 8:15 AM (UTC-7:00)
Build number	1

**Phase details**

Name	Status	Context	Duration	Start time
SUBMITTED	✓ Succeeded	-	<1 sec	Jul 19, 2025 8:14 AM (UTC-7:00)
QUEUED	✓ Succeeded	-	<1 sec	Jul 19, 2025 8:14 AM (UTC-7:00)
PROVISIONING	✓ Succeeded	-	4 secs	Jul 19, 2025 8:14 AM (UTC-7:00)

### Step 4: Establish an ECS cluster

The screenshot shows the AWS Amazon Elastic Container Service console. The left sidebar shows 'Amazon Elastic Container Service' with 'Clusters' selected. The main content area displays a blue banner at the top stating 'On June 25, 2025, Amazon ECS changed the default log driver mode from blocking to non-blocking to improve application availability during CloudWatch outages. Learn more'. Below this, the breadcrumb navigation is 'Amazon Elastic Container Service > Clusters > SelviECScluster > Services'. The main heading is 'SelviECScluster'. The cluster overview shows the ARN 'arn:aws:ecs:us-east-1:18570588843:cluster/SelviECScluster', status 'Active', CloudWatch monitoring 'Default', and registered container instances '-'. Below this, there are tabs for 'Services', 'Tasks', 'Infrastructure', 'Metrics', 'Scheduled tasks', 'Configuration', and 'Tags'. The 'Services' tab is selected, showing 'Services (0) Info'. There are filters for 'Filter launch type' (Any launch type) and 'Filter service type' (Any service type). Below the filters, there is a table with columns: Service name, ARN, Status, Service..., Created at, Deployments and tasks, and Last deployment. The table is empty, and a message at the bottom states 'No services to display.'

**Cluster overview**

Property	Value
ARN	arn:aws:ecs:us-east-1:18570588843:cluster/SelviECScluster
Status	✓ Active
CloudWatch monitoring	Default
Registered container instances	-

**Services**

Property	Value
Draining	-
Active	-
Pending	-
Running	-

**Services (0) Info**

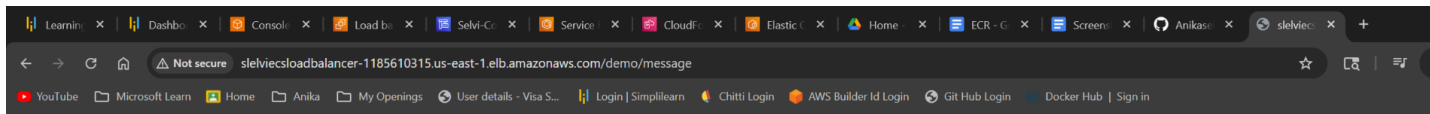
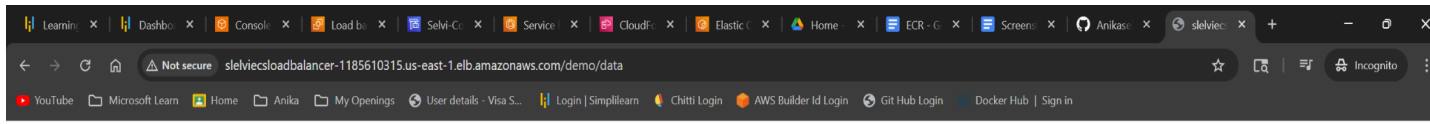
Filter launch type: Any launch type

Filter service type: Any service type

Service name	ARN	Status	Service...	Created at	Deployments and tasks	Last deployment
--------------	-----	--------	------------	------------	-----------------------	-----------------

No services to display.

## Step 5: Validate the Application Deployment



## Step 6. Construct and Execute a CodePipeline to automate the deployment process

The screenshot displays the AWS CodePipeline console for a pipeline named 'SelviCodePipeline'. The interface includes a top navigation bar with the AWS logo, a search bar, and user information. Below the navigation bar, the pipeline's name is shown along with icons for cloning and settings. A row of buttons at the top right allows for 'Edit', 'Stop execution', 'Create trigger', 'Clone pipeline', and 'Release change'. The main view is divided into tabs: 'Pipeline', 'Executions', 'Triggers', 'Settings', 'Tags', and 'Stage'. The 'Pipeline' tab is active, showing a visual representation of the pipeline's stages and actions.

The pipeline consists of three stages, each with a green checkmark indicating successful completion:

- Source Stage:** ID 1f018b18-d92e-44e5-b4e5-e3b90ace5b8b. All actions succeeded. It contains one action named 'Source' using the 'GitHub (via GitHub App)' provider, which completed 17 minutes ago.
- Build Stage:** ID 1f018b18-d92e-44e5-b4e5-e3b90ace5b8b. All actions succeeded. It contains one action named 'Build' using the 'AWS CodeBuild' provider, which completed 16 minutes ago.
- Deploy Stage:** ID 1f018b18-d92e-44e5-b4e5-e3b90ace5b8b. All actions succeeded. It contains one action named 'Deploy' using the 'Amazon ECS' provider, which completed 9 minutes ago.

Each stage also displays a list of actions with their IDs and sources, such as 'e94c2959' from 'Source: second com'.