

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

The screenshot shows the Amazon ECR console. On the left, there's a sidebar with navigation options like 'Amazon Elastic Container Registry', 'Private registry', 'Public registry', and links to 'Getting started' and 'Documentation'. The main area is titled 'Private repositories (1)' and lists a single repository: 'selvicourseendprojectrepo'. The repository details are as follows:

Repository name	URI	Created at	Tag immutability	Encryption type
selvicourseendprojectrepo	18570588843.dkr.ecr.us-east-1.amazonaws.com/selvicourseendprojectrepo	July 19, 2025, 07:21:11 (UTC-07)	Mutable	AES-256

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

The screenshot shows a GitHub repository page for 'Selvi_CourseEndProject'. The repository is public and has one branch ('main') and no tags. The commit history shows the following files were added in the first commit:

File	Commit Message	Time
.mvn/wrapper	First Commit	2 minutes ago
src	First Commit	2 minutes ago
.gitignore	First Commit	2 minutes ago
Dockerfile	First Commit	2 minutes ago
README.md	First Commit	2 minutes ago
buildspec.yml	First Commit	2 minutes ago
mvnw	First Commit	2 minutes ago
mvnw.cmd	First Commit	2 minutes ago
pom.xml	First Commit	2 minutes ago

The repository also includes sections for 'About', 'Packages', and 'Languages'.

```
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/Anikasel/Selvi_CourseEndProject.git
 * [new branch]      main -> main

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

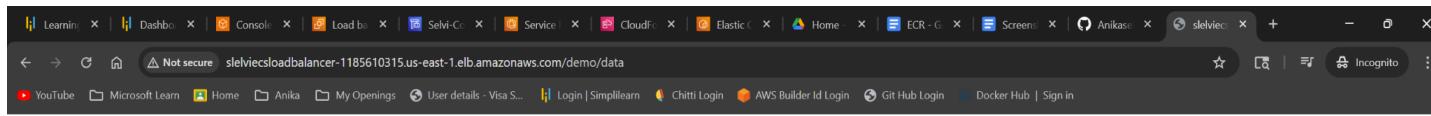
Step 3: Create a Code Build project

The screenshot shows the AWS CodeBuild console. On the left, a sidebar menu under 'Developer Tools' has 'CodeBuild' selected. The main 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 build ID 'Selvi-CodeBuildProject:441d022e-eb52-44bf-9841-19682b0e16c2' is shown. The 'Build status' section details the build's status as 'Succeeded', initiated by 'SelviRole/odl_user_1798140', with a build ARN. It also shows the start time as Jul 19, 2025 8:14 AM (UTC-7:00) and end time as Jul 19, 2025 8:15 AM (UTC-7:00), build number 1. The 'Phase details' tab is selected, showing three phases: SUBMITTED, QUEUED, and PROVISIONING, all in 'Succeeded' status. The 'Build logs' tab is also present.

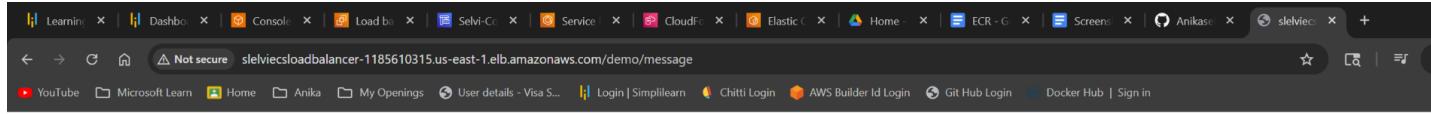
Step 4: Establish an ECS cluster

The screenshot shows the AWS Elastic Container Service (ECS) console. The left sidebar menu under 'Amazon Elastic Container Service' has 'Clusters' selected. The main area shows a cluster named 'SelviECSCluster'. The 'Cluster overview' section displays the ARN (arn:aws:ecs:us-east-1:18570588843:cluster/SelviECScluster), status (Active), CloudWatch monitoring (Default), and registered container instances (none). The 'Services' tab is selected, showing 0 services. A search bar at the bottom allows filtering by service name, ARN, status, and other metrics.

Step 5: Validate the Application Deployment



First message from AWS Ecs



Second message from AWS Ecs

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

