Study Notes: Complete AWS CI/CD for Spring Boot Deployment

Overview

Tutorial by @Javatechie on deploying Spring Boot applications using AWS CI/CD.

Focus on AWS services: CodeBuild and CodePipeline.

Key Concepts

CI/CD: Continuous Integration and Continuous Deployment processes.

AWS Services:

AWS CodeBuild: Automates the build process.

AWS CodePipeline: Automates the deployment process.

Steps for Deployment

Create a Spring Boot App:

Develop the application and define CRUD operations.

Docker Image Creation:

Create a Dockerfile:

Use OpenJDK 17.

Define working directory and expose relevant ports.

AWS ECR Setup:

Create an Elastic Container Registry (ECR) repository for the Docker image.

Build Specification:

Create a buildspec.yml file:

Define pre-build, build, and post-build phases.

Include commands to build the Docker image and push it to ECR.

Configure AWS CodeBuild:

Create a project in AWS CodeBuild.

Link to the GitHub repository containing the buildspec.yml.

Attach necessary IAM roles/policies for ECR access.

Trigger Build:

Start the build process in CodeBuild to verify CI steps (image creation and pushing to ECR).

Continuous Delivery:

Set up AWS ECS (Elastic Container Service) for deploying the application.

Important Files

Dockerfile: Contains instructions for creating the Docker image.

buildspec.yml: Defines the build process for CodeBuild.

Troubleshooting

Monitor CodeBuild logs for errors during the build phases.

Validate successful image push by checking ECR for the new images.

Conclusion

Successfully automate the CI/CD process for deploying a Spring Boot application on AWS ECS using CodeBuild and CodePipeline.

### Metadata

- Title:Complete AWS CI/CD | Deploy Spring Boot to ECS Using CodeBuild & CodePipeline |

- URL:https://www.youtube.com/watch?v=40X6abe5wv0

### Notes

- ([01:00](https://www.youtube.com/watch?v=40X6abe5wv0&t=60s)) Study Notes: Complete AWS CI/CD for Spring Boot Deployment

Overview

Tutorial by @Javatechie on deploying Spring Boot applications using AWS CI/CD.

Focus on AWS services: CodeBuild and CodePipeline.

Key Concepts

CI/CD: Continuous Integration and Continuous Deployment processes.

AWS Services:

AWS CodeBuild: Automates the build process.

AWS CodePipeline: Automates the deployment process.

Steps for Deployment

Create a Spring Boot App:

Develop the application and define CRUD operations.

Docker Image Creation:

Create a Dockerfile:

Use OpenJDK 17.

Define working directory and expose relevant ports.

AWS ECR Setup:

Create an Elastic Container Registry (ECR) repository for the Docker image.

Build Specification:

Create a buildspec.yml file:

Define pre-build, build, and post-build phases.

Include commands to build the Docker image and push it to ECR.

Configure AWS CodeBuild:

Create a project in AWS CodeBuild.

Link to the GitHub repository containing the buildspec.yml.

Attach necessary IAM roles/policies for ECR access.

Trigger Build:

Start the build process in CodeBuild to verify CI steps (image creation and pushing to ECR).

Continuous Delivery:

Set up AWS ECS (Elastic Container Service) for deploying the application.

Important Files

Dockerfile: Contains instructions for creating the Docker image.

buildspec.yml: Defines the build process for CodeBuild.

Troubleshooting

Monitor CodeBuild logs for errors during the build phases.

Validate successful image push by checking ECR for the new images.

Conclusion

Successfully automate the CI/CD process for deploying a Spring Boot application on AWS ECS using CodeBuild and CodePipeline.

-- With NoteGPT

**Study Notes: AWS CI/CD with Spring Boot on ECS using CodeBuild & CodePipeline**

**Overview**

* **Goal**: Automate CI/CD for deploying a Spring Boot application on AWS ECS using AWS CodeBuild and CodePipeline.
* **Key Concepts**:
  + Continuous Integration (CI)
  + Continuous Deployment/Delivery (CD)

**Sequence of Steps**

1. **Creating the Spring Boot Application**:
   * Develop a simple Spring Boot app with CRUD operations.
2. **Dockerize the Application**:
   * Create a **Dockerfile** to define how to build the Docker image.
     + Use OpenJDK 17, set working directory, copy jar, expose port, and define the command to run the jar.
3. **Push Docker Image to AWS ECR**:
   * Create an **Elastic Container Registry (ECR)** in AWS.
   * Push the Docker image to ECR after building it.
4. **Automate CI with AWS CodeBuild**:
   * Create a **buildspec.yml** file to define the build process:
     + **Pre-build**: Generate jar file, log into ECR, create commit hash and image tag.
     + **Build**: Build the Docker image, tag it.
     + **Post-build**: Push the Docker image to ECR with both ‘latest’ and unique tags.
5. **Set Up AWS CodeBuild**:
   * Create a CodeBuild project linked to the GitHub repository containing the **buildspec.yml**.
6. **Set Up AWS CodePipeline** (not covered in the provided content):
   * To automate deployment to AWS ECS after the build is successful.

**Important Files**

* **Dockerfile**: Defines how to create the Docker image.
* **buildspec.yml**: Contains instructions for CodeBuild on building and pushing the Docker image.

**AWS Services Utilized**

* **AWS CodeBuild**: For building the Docker image and pushing it to ECR.
* **AWS CodePipeline**: For automating the deployment process (to be set up after CodeBuild).

**Key Considerations**

* Ensure that IAM roles are properly configured with necessary permissions for CodeBuild to access ECR.
* Maintain the **buildspec.yml** and **Dockerfile** in the GitHub repository for seamless CI/CD.

**Next Steps**

* Set up **AWS CodePipeline** to automate continuous deployment to AWS ECS once CodeBuild successfully completes the build.

**Resources**

* AWS Documentation for [CodeBuild](https://docs.aws.amazon.com/codebuild/latest/userguide/welcome.html) and [CodePipeline](https://docs.aws.amazon.com/codepipeline/latest/userguide/welcome.html).
* Javatechie’s AWS playlist for foundational concepts.

