****

**AWS DEVOPS TEST**

**By Văn Công Lê Ca**

**Table of Contents**

[1. General 3](#_Toc182992796)

[1.1. Glossary 4](#_Toc182992797)

[1.2. CI/CD Workflow 5](#_Toc182992798)

[1.2.1. Operation Workflow 5](#_Toc182992799)

[1.2.2. Development Workflow 7](#_Toc182992800)

[2. Document 8](#_Toc182992801)

[2.1. Environment Deployment 8](#_Toc182992802)

[2.1.1. IAM Roles & Policies 8](#_Toc182992803)

[2.1.2. Components To Be Deployed Manually 14](#_Toc182992804)

[2.2. Project Images 16](#_Toc182992805)

[2.2.1. Project Github Repository 16](#_Toc182992806)

[2.2.2. Project Demo Images 16](#_Toc182992807)

**Table of Tables**

[**Table 1.** Glossary. 4](#_Toc182992834)

[**Table 2**. Roles & Policies 14](#_Toc182992835)

**Table of Figures**

[**Figure 1.** Project overview. 3](#_Toc182992826)

[**Figure 2.** Operator workflow. 5](#_Toc182992827)

[**Figure 3.** Development workflow 7](#_Toc182992828)

[**Figure 4.** Infrastrutures pipeline 16](#_Toc182992829)

[**Figure 5.** Application pipeline 17](#_Toc182992830)

[**Figure 6.** dev.json (configuration for dev environment) 17](#_Toc182992831)

[**Figure 7.** IaC for buildspec.yml 18](#_Toc182992832)

[**Figure 8.** Application buildspec.yml 18](#_Toc182992833)

# General



**Figure 1.** Project overview.

This project focuses on deploying a Java Tomcat application on AWS Cloud across three environments: dev, test, and prod, using AWS CDK (Python) for Infrastructure as Code (IaC). The solution includes hosting the application on EC2 instances managed by an Auto Scaling Group and distributing traffic via an Elastic Load Balancer (ELB). Two pipelines are implemented using AWS CodePipeline: one for the CI/CD process to build and deploy the application with zero downtime and another for deploying the CDK source code to manage infrastructure. The setup supports environment-specific configurations such as instance types and scaling policies, integrates security checks using cdk-nag, and ensures a modular, secure, and scalable architecture tailored for enterprise-grade deployments.

# Glossary

|  |  |
| --- | --- |
| **Term** | **Definition** |
| AWS | Amazon Web Services |
| CDK | Cloud Development Kit |
| IaC | Infrastructure as Code |
| CI/CD | Continuous Integration and Continuous Deployment |
| CodePipeline | AWS service for automating CI/CD workflows |
| CodeBuild | AWS service for compiling source code, running tests, and producing deployment artifacts |
| CloudFormation | AWS service for defining and managing infrastructure as code using templates |
| VPC | Virtual Private Cloud |
| Subnet | Subnetwork within a VPC |
| ALB | Application Load Balancer |
| ASG | Auto Scaling Group |
| ELB | Elastic Load Balancer |
| EC2 | Elastic Compute Cloud (AWS virtual servers) |
| S3 | Simple Storage Service for storing and retrieving any amount of data |
| Parameter Store | AWS Systems Manager service for securely storing and accessing parameters like secrets and configuration data |
| IAM | Identity and Access Management (for managing permissions and roles in AWS) |
| Operator | Individual/Team responsible for setting up and managing the environment |
| Dev | Individual/Team responsible for developing the application |
| Application | Java Tomcat-based software to be deployed on AWS EC2 |
| Infrastructure | AWS resources such as VPCs, Subnets, Security Groups, EC2, ALB, ASG, etc. |
| Pipeline | An automated workflow for building, testing, and deploying applications |
| cdk-nag | A tool to check AWS CDK code for compliance with security best practices |
| WAR | Web Application Archive (Java-based application package format) |

**Table 1.** Glossary.

# CI/CD Workflow

# Operation Workflow



**Figure 2.** Operator workflow.

When an operator commits Infrastructure as Code (IaC) to GitHub, the infrastructure pipeline executes the following steps:

1. **Source Stage**:
   * The pipeline pulls the source code from the specified repository in GitHub.
2. **Branch-Specific Workflow**:
   * For the **dev** branch, the source code proceeds directly to the build stage without additional approvals.
   * For the **test** and **master** branches, administrator approval is required before the code can move to the build stage.
3. **Build Stage**:
   * The **CodeBuild** process creates the infrastructure as defined in the IaC code, including the **VPC**, **Auto Scaling Group (ASG)**, and **Elastic Load Balancer (ELB)**.
   * Once the infrastructure is successfully deployed, the pipeline updates the **/application/dev/latestcommitid** parameter in **AWS Parameter Store** to reflect the latest commit ID.
4. **Deployment Stage**:
   * The newly provisioned infrastructure launches **EC2** **instances** in the **ASG**. These instances are configured to work behind the **ELB** for traffic distribution.
   * The **EC2** **instances** retrieve the **.war file** from an S3 bucket (created during the development phase).
   * The **Tomcat server** on the EC2 instances deploys the .war file, making the application accessible through the ELB.

This workflow ensures a robust CI/CD process by automating infrastructure provisioning with VPCs, ASGs, and ELBs while enforcing branch-specific approvals for stricter workflows in test and production environments.

# Development Workflow



**Figure 3.** Development workflow

When a developer commits application source code to GitHub, the application pipeline follows these steps:

1. **Source Stage**:
   * The pipeline pulls the latest application source code from the GitHub repository.
2. **Build Stage**:
   * The **CodeBuild** process compiles the source code into a **.war file**.
   * The generated .war file is uploaded to the appropriate **S3** **bucket**.
   * Based on the branch, the pipeline updates the corresponding parameter in **AWS Parameter Store**:
     + **dev branch**: Updates /application/dev/helloword to indicate the new .war file.
     + **test branch**: Updates /application/test/ helloword.
     + **prod branch**: Updates /application/prod/helloword.
   * These parameters ensure that the EC2 instances in each environment can identify and download the latest .war file for deployment.
3. **Trigger Infrastructure Pipeline**:
   * After completing the build, the application pipeline triggers the **infrastructure pipeline** to redeploy with the latest update.
   * The triggered infrastructure pipeline uses the **latestcommitid** parameter (which has been updated and saved during the infrastructure phase) for the respective branch to execute the appropriate infrastructure and deploy it. For example:
     + The **dev branch** triggers the parameter application/dev/latestcommitid.
     + The **test branch** triggers /application/test/latestcommitid.
     + The **prod branch** triggers /application/prod/latestcommitid.

This workflow integrates both application and infrastructure updates, ensuring that deployments in each environment are seamless, branch-specific, and always use the most recent application build.

# Document

# Environment Deployment

# IAM Roles & Policies

|  |  |  |
| --- | --- | --- |
| Role/Policy Name | Purpose | JSON |
| IAM Role:  **codebuild-DemoCodebuild-service-role** | Allow AWS CodeBuild to assume the role and perform actions on behalf of the user. | {  "Version": "2012-10-17",  "Statement": [  {  "Effect": "Allow",  "Principal": {  "Service": "codebuild.amazonaws.com"  },  "Action": "sts:AssumeRole"  }  ]  } |
| IAM Policy:  **DemoCodeBuildPolicy** |  | {  "Version": "2012-10-17",  "Statement": [  {  "Sid": "Statement1",  "Effect": "Allow",  "Action": [  "sts:AssumeRole"  ],  "Resource": [  "arn:aws:iam::259642033136:role/cdk-hnb659fds-cfn-exec-role-259642033136-ap-southeast-1",  "arn:aws:iam::259642033136:role/cdk-hnb659fds-deploy-role-259642033136-ap-southeast-1",  "arn:aws:iam::259642033136:role/cdk-hnb659fds-file-publishing-role-259642033136-ap-southeast-1",  "arn:aws:iam::259642033136:role/cdk-hnb659fds-image-publishing-role-259642033136-ap-southeast-1",  "arn:aws:iam::259642033136:role/cdk-hnb659fds-lookup-role-259642033136-ap-southeast-1"  ]  }  ]  } |
| IAM Policy:  **PutAndGetParam** | Allows an entity to assume multiple AWS CDK roles needed for deploying and managing infrastructure through CloudFormation, including tasks like file publishing and image handling. | {  "Version": "2012-10-17",  "Statement": [  {  "Sid": "VisualEditor0",  "Effect": "Allow",  "Action": [  "ssm:GetParameter",  "ssm:PutParameter"  ],  "Resource": [  "arn:aws:ssm:ap-southeast-1:259642033136:parameter/application/dev/\*",  "arn:aws:ssm:ap-southeast-1:259642033136:parameter/application/prod/\*",  "arn:aws:ssm:ap-southeast-1:259642033136:parameter/application/test/\*"  ]  }  ]  } |
| IAM Policy:  **S3GetPut** | Grants permission to read from and write to objects in the demo-elb-access-log-bucket S3 bucket. | {  "Version": "2012-10-17",  "Statement": [  {  "Sid": "VisualEditor0",  "Effect": "Allow",  "Action": [  "s3:PutObject",  "s3:GetObject"  ],  "Resource": "arn:aws:s3:::demo-elb-access-log-bucket/\*"  }  ]  } |
| IAM Role:  **codebuild-ApplicationCodebuild-service-role** | Allows AWS CodeBuild to assume the role and perform actions on behalf of the user. | {  "Version": "2012-10-17",  "Statement": [  {  "Effect": "Allow",  "Principal": {  "Service": "codebuild.amazonaws.com"  },  "Action": "sts:AssumeRole"  }  ]  } |
| IAM Policy:  **RunInfrastructurePipeline** | Grants permission to start the execution of the InfrastructurePipeline in AWS CodePipeline. | {  "Version": "2012-10-17",  "Statement": [  {  "Sid": "VisualEditor0",  "Effect": "Allow",  "Action": "codepipeline:StartPipelineExecution",  "Resource": "arn:aws:codepipeline:ap-southeast-1:259642033136:InfrastructurePipeline"  }  ]  } |
| **CodeBuildBasePolicy-ApplicationCodebuild-ap-southeast-1** | Grants permissions for logging, S3 interactions, and CodeBuild report management for the ApplicationCodebuild project in AWS CodeBuild. | {  "Version": "2012-10-17",  "Statement": [  {  "Effect": "Allow",  "Resource": [  "arn:aws:logs:ap-southeast-1:259642033136:log-group:/aws/codebuild/ApplicationCodebuild",  "arn:aws:logs:ap-southeast-1:259642033136:log-group:/aws/codebuild/ApplicationCodebuild:\*"  ],  "Action": [  "logs:CreateLogGroup",  "logs:CreateLogStream",  "logs:PutLogEvents"  ]  },  {  "Effect": "Allow",  "Resource": [  "arn:aws:s3:::codepipeline-ap-southeast-1-\*"  ],  "Action": [  "s3:PutObject",  "s3:GetObject",  "s3:GetObjectVersion",  "s3:GetBucketAcl",  "s3:GetBucketLocation"  ]  },  {  "Effect": "Allow",  "Action": [  "codebuild:CreateReportGroup",  "codebuild:CreateReport",  "codebuild:UpdateReport",  "codebuild:BatchPutTestCases",  "codebuild:BatchPutCodeCoverages"  ],  "Resource": [  "arn:aws:codebuild:ap-southeast-1:259642033136:report-group/ApplicationCodebuild-\*"  ]  }  ]  } |

**Table 2**. Roles & Policies

# Components To Be Deployed Manually

Infrastructure CodePipeline (using AWS CodePipeline):

* Deploy a pipeline that automates the deployment of infrastructure resources, such as EC2 instances, Autoscaling Group, Load Balancer, and VPC configurations.

Application CodePipeline (using AWS CodePipeline):

* Deploy a separate pipeline that automates the building and deployment of application code (e.g., compiling WAR files) to the EC2 instances or services.

Build Infrastructure Project (using AWS CodeBuild):

* Set up an AWS CodeBuild project for building the infrastructure components. This includes compiling the necessary configuration files, infrastructure templates, and any other resources required for deployment.

Build Application Project (using AWS CodeBuild):

* Set up another AWS CodeBuild project for building the application source code (e.g., building WAR files) and storing the output artifacts in an S3 bucket for later deployment.

Create S3 Bucket for Build Artifacts:

* Create an S3 bucket to store build artifacts (e.g., WAR files) generated by the build project. These artifacts will be used in the deployment process.

Create S3 Bucket for Access Logs:

* Create a dedicated S3 bucket to store access logs generated by the Elastic Load Balancer (ELB) to track incoming traffic and help with monitoring and troubleshooting.

Create Parameters in Parameter Store:

* Set up parameters in AWS Systems Manager Parameter Store to store environment-specific configuration values, such as latestcommitid for application deployment or other custom parameters needed by the application and infrastructure.

Create IAM Roles and Policies:

* Create IAM roles and policies that grant necessary permissions for resources to interact with each other:
* Role for accessing S3: Allow entities like CodeBuild, CodePipeline, or EC2 instances to retrieve or store data in S3 buckets.
* Role for modifying Parameters in Parameter Store: Grant permissions to modify the parameters in the Parameter Store, allowing updates to environment-specific values such as deployment versions or configuration settings.

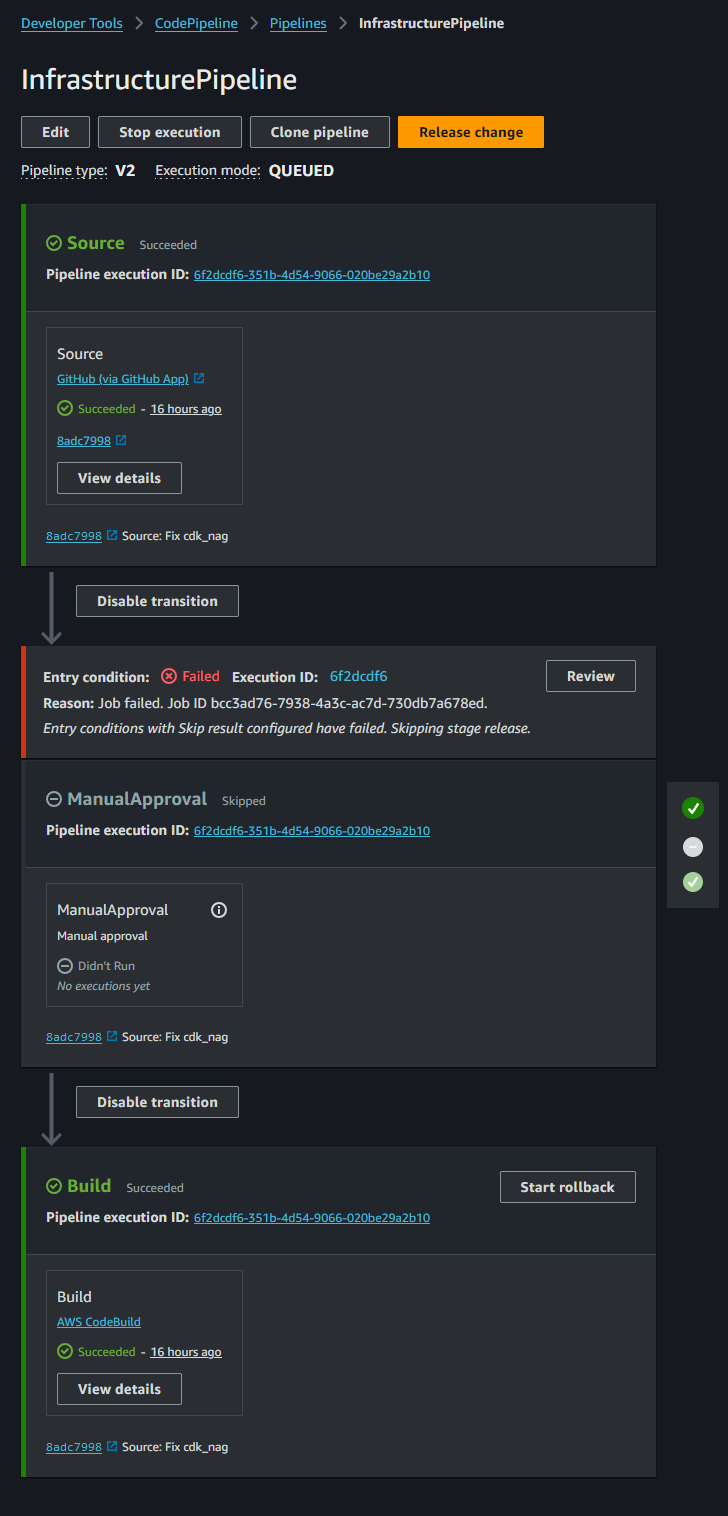
These components will be deployed manually to enable the automation of the application deployment pipeline, manage infrastructure resources, and ensure secure access to necessary services like S3 and Parameter Store.

# Project Images

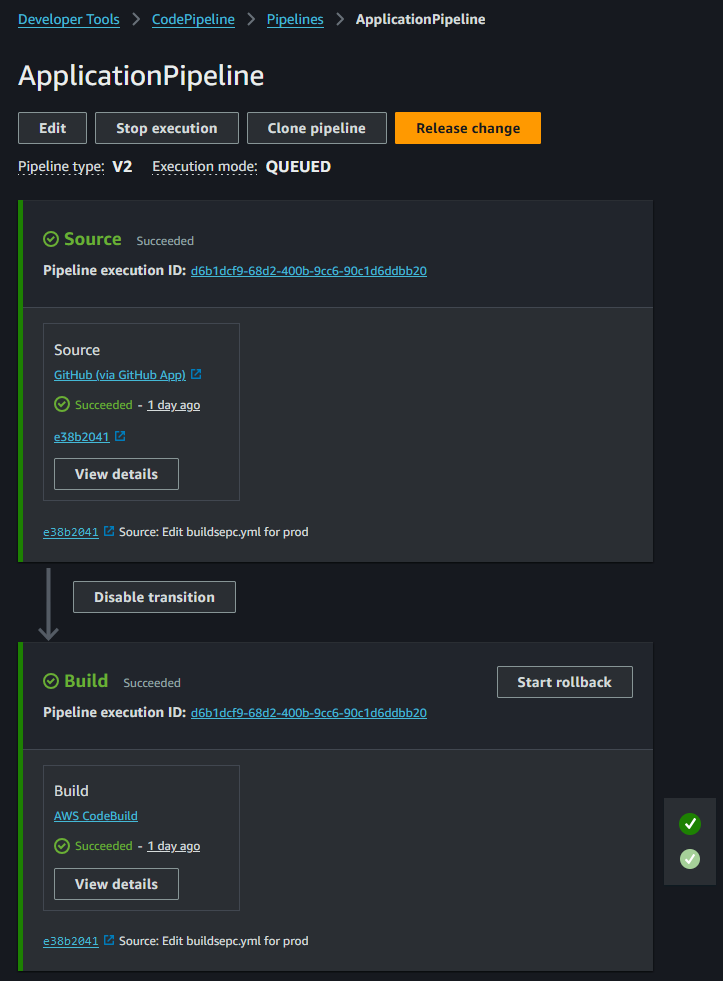
# Project Github Repository

* **Infrastructure as Code (IaC)**: [AWS-Demo-Project](https://github.com/Casca113s2/AWS-Demo-Project)
* **Application**: [tomcat10-jakartaee9](https://github.com/Casca113s2/tomcat10-jakartaee9)

# Project Demo Images



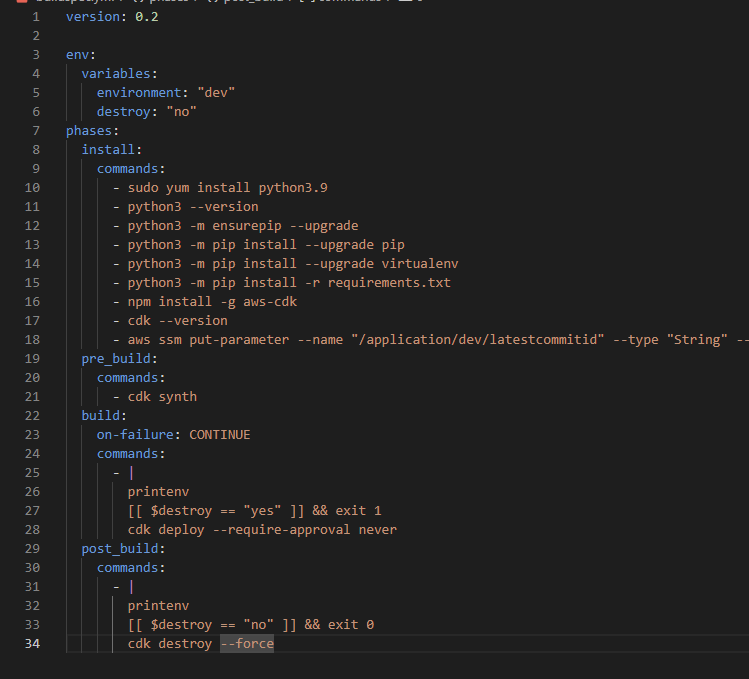
**Figure 4.** Infrastrutures pipeline



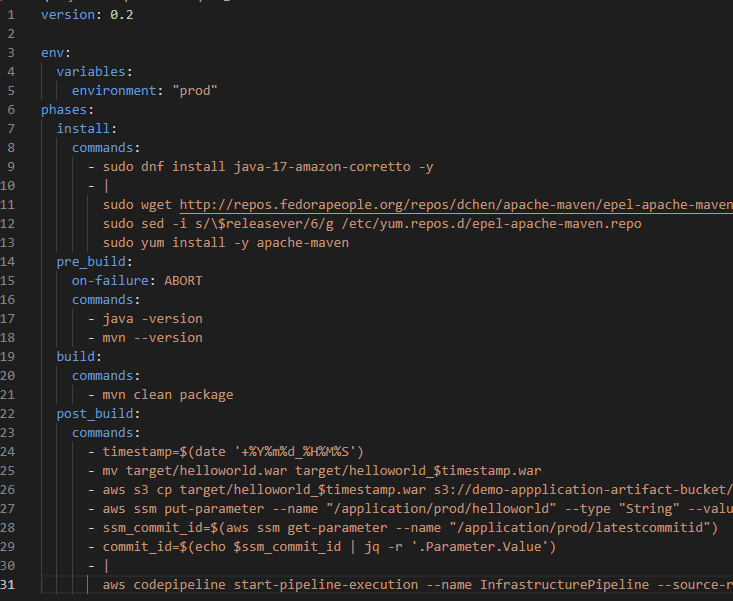
**Figure 5.** Application pipeline



**Figure 6.** dev.json (configuration for dev environment)



**Figure 7.** IaC for buildspec.yml



**Figure 8.** Application buildspec.yml