CIRI2 CI/CD Document

Jordy Walraven

Version history:

Version	Author(s)	Description	Date
1	Jordy Walraven	Setup Initial architecture	27-3-2024

Distribution history:

Version	Distributed to	Comments	Date

Table of contents

Context

Ciri2 is a platform designed for PC gaming enthusiasts. At its core, Ciri2 serves as a hub where users can assess their system's capability to run specific games and benchmark their performance against others in the community. Here's an overview of its key functionalities:

Account Creation:

Users can easily create personalized accounts, unlocking access to a host of features tailored to their gaming needs.

PC Rig Configuration:

Within the application, users have the flexibility to construct and customize their PC rigs using intuitive part pickers for various components. This empowers them to tailor their systems to meet the demands of their favorite games.

Game Overview:

Through Ciri2, users gain access to a comprehensive overview of games, including key details such as system requirements, gameplay mechanics, and community-generated performance data. This empowers gamers to make informed decisions about which titles to explore based on their hardware capabilities.

FPS Submission:

Gamers can submit their frames per second (FPS) benchmarks for specific games and graphics presets, allowing them to share and compare their performance metrics with fellow users. This feature fosters a sense of community and healthy competition among players.

Front-end CI/CD flow

Development workflow

Overview

The Development Pipeline is an automated workflow triggered by every push and every pull request in the development environment. This pipeline streamlines the process of ensuring code quality, testing, and analysis before merging changes into the main branch. The pipeline comprises several essential steps to ensure the reliability and efficiency of the application's development cycle.

Steps in the Development Pipeline

1. Install Dependencies

This frontend application uses Nodejs, this means that it needs to install the node_modules before it can run or build the application.

2. Build the Application

In this stage, the application source code is compiled and built into executable code or artifacts. The build process ensures that the code is converted into a deployable format suitable for execution.

3. Run Tests

Testing is a crucial aspect of the development process to ensure that the application functions as expected and meets the defined requirements. Automated tests, including unit tests, integration tests, and end-to-end tests, are executed to validate the behavior and functionality of the application.

4. Submit to SonarCloud for Static Code Analysis

SonarCloud is used for static code analysis to identify code smells, bugs, security vulnerabilities, and other potential issues in the source code. The pipeline submits the codebase to SonarCloud, which performs comprehensive analysis and provides actionable feedback to improve code quality and maintainability.

Release workflow

Overview

The Release Workflow is a structured process for preparing and deploying new versions of the application to production. This workflow ensures that the application is thoroughly tested, analyzed, and packaged before being released to end-users. The primary focus of the Release Workflow is to build the Docker image of the application, push it to DockerHub, and perform Lighthouse analysis to assess performance and accessibility. This workflow is run when creating a new release on github.

Steps in the Release Workflow

1. Build the Dockerfile of the Application

The first step in the Release Workflow involves building the Docker image of the application using the Dockerfile. The Dockerfile contains instructions for building the application environment and dependencies within a containerized environment.

2. Push Dockerfile to DockerHub

Once the Docker image is successfully built, it is pushed to DockerHub, a cloud-based registry service for storing and distributing Docker images. Pushing the Docker image to DockerHub makes it accessible to other developers and deployment pipelines.

3. Pull Dockerfile in Lighthouse Step

In this step, the Docker image previously built and pushed to DockerHub is pulled into the Lighthouse step. Lighthouse is an open-source tool for improving the quality of web pages by auditing performance, accessibility, and other aspects.

4. Run Lighthouse Analysis

With the Docker image loaded, Lighthouse conducts a comprehensive analysis of the application, focusing on performance metrics such as load time, interactivity, and accessibility compliance. The analysis results provide valuable insights into areas for improvement to enhance the user experience and overall quality of the application.