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.

Workflow: Development

This workflow runs on every push or pull request to the repository.

Triggering Events

- **push**: Runs the workflow on every push to the repository.
- **pull_request**: Runs the workflow when a pull request is opened, synchronized, or reopened.

Job: build-test-tag

This job runs on the latest Ubuntu environment and performs the following steps:

1. **Checkout Code**
    - Uses the actions/checkout@v4 action to clone the repository.
    - persist-credentials: false prevents GitHub token from being stored.
    - fetch-depth: 0 ensures the entire history is fetched.
2. **Setup pnpm**
    - Uses pnpm/action-setup@v4 to install pnpm version 8.
3. **Setup Go**
    - Uses actions/setup-go@v5 to install Go version 1.22.
    - Runs go version to verify the Go installation.
4. **Set Git Identity**
    - Configures Git with a specific user name and email for versioning.
    - Sets the remote URL to include a GitHub token for authentication.
5. **Setup Node.js**
    - Uses actions/setup-node@v4 to install Node.js version 20 (from the matrix configuration).
    - Caches pnpm dependencies.
6. **Install Dependencies**
    - Runs pnpm install to install project dependencies.
7. **Install nx**
    - Runs pnpm install -g nx to install nx globally.
8. **Lint Applications**
    - Runs pnpm run lint:affected to check for linting issues in affected applications.
9. **Build Applications**
    - Runs pnpm run build:affected to build the affected applications.
10. **Run Tests**

- Runs pnpm run test:affected to execute tests for the affected applications.

11. **Semantic Versioning**
    - Runs pnpm run version:affected to perform semantic versioning.
    - This step only runs on pushes to the main branch.
    - Uses the GitHub token from secrets for authentication.
12. **Tag Last Release**
    - Tags the latest commit as last-release.
    - Forces the push of the last-release tag to the remote repository.
    - This step only runs on pushes to the main branch.

This setup ensures that every code push or pull request is checked, built, tested, and versioned automatically, maintaining code quality and consistent releases.

Workflow: Deployment

This workflow runs on every published release.

Triggering Events

- **release**: Runs the workflow when a release is published.

Job: dockerhub-release

This job prepares and releases a Docker image on DockerHub.

1. **Checkout Code**
    - Uses the actions/checkout@v4 action to clone the repository.
    - persist-credentials: false prevents GitHub token from being stored.
    - fetch-depth: 0 ensures the entire history is fetched.
2. **Setup pnpm**
    - Uses pnpm/action-setup@v4 to install pnpm version 8.
3. **Setup Go**
    - Uses actions/setup-go@v5 to install Go version 1.22.
    - Runs go version to verify the Go installation.
4. **Setup Node.js**
    - Uses actions/setup-node@v4 to install Node.js version 20 (from the matrix configuration).
    - Caches pnpm dependencies.
5. **Install Dependencies**

- Runs pnpm install to install project dependencies.

6. **Install nx**
   - Runs pnpm install -g nx to install nx globally.
7. **Get Application Name from Tag**
   - Extracts the application name from the release tag.
   - Uses shell commands to parse the tag name and set it as an output variable.
8. **Build Application**
   - Runs nx build "${{ steps.get_tag.outputs.app_name }}" to build the application based on the parsed tag name.
9. **Docker Login**
   - Logs into DockerHub using credentials stored in GitHub secrets.
10. **Dockerhub Application**
    - Runs nx docker "${{ steps.get_tag.outputs.app_name }}" to build and push the Docker image to DockerHub.

Job: deploy

This job deploys the Docker image to Azure Kubernetes Service (AKS).

1. **Needs dockerhub-release**
   - This job depends on the completion of the dockerhub-release job.
2. **Checkout Repository**
   - Uses actions/checkout@v3 to clone the repository.
3. **Set up Azure CLI**
   - Uses azure/login@v1 to set up the Azure CLI with credentials stored in GitHub secrets.
4. **Azure CLI - Get AKS Credentials**
   - Configures the Azure CLI to use the specified subscription.
   - Retrieves AKS cluster credentials to interact with the Kubernetes cluster.
5. **Set up Kubectl**
   - Uses azure/setup-kubectl@v3 to install the latest version of kubectl.
6. **Get Application Name from Tag**
   - Repeats the step to extract the application name from the release tag, ensuring consistency across jobs.
7. **Deploy to AKS**

- Runs kubectl rollout restart deployment/${{ steps.get_tag.outputs.app_name }} to restart the deployment in AKS with the new Docker image.

This setup ensures that every published release triggers a build and deployment process, pushing Docker images to DockerHub and deploying the updated application to AKS, maintaining a streamlined and automated release workflow.

Workflow: E2E Tests

This workflow runs end-to-end (E2E) tests on pull requests to the main branch.

Triggering Events

- **pull_request**: Runs the workflow when a pull request targets the main branch.

Job: run-e2e

This job runs E2E tests using Playwright.

1. **Runs on ubuntu-latest**
   - The job is executed on the latest Ubuntu environment.
2. **Environment Variables**
   - Various environment variables are set from GitHub secrets for Playwright, database connections, and other configurations.
3. **Node.js Version Matrix**
   - The job runs using Node.js version 20 (defined in the matrix configuration).
4. **Steps to Execute**

1. **Checkout Code**
   - Uses the actions/checkout@v4 action to clone the repository.
   - persist-credentials: false prevents GitHub token from being stored.
   - fetch-depth: 0 ensures the entire history is fetched.
2. **Setup pnpm**
   - Uses pnpm/action-setup@v4 to install pnpm version 8.
3. **Setup Go**
   - Uses actions/setup-go@v5 to install Go version 1.22.
   - Runs go version to verify the Go installation.
4. **Setup Node.js**
   - Uses actions/setup-node@v4 to install Node.js version 20.
   - Caches pnpm dependencies.
5. **Install Dependencies**

- Runs pnpm install to install project dependencies.
- Runs npx playwright install --with-deps to install Playwright with dependencies.
- Installs nx globally with pnpm install -g nx.

6. **Build Applications**
   - Runs pnpm run build:all to build all applications.
7. **Run Docker**
   - Runs docker compose up --build -d to build and start Docker containers in detached mode.
8. **Playwright Tests**
   - Runs nx e2e frontend to execute E2E tests using Playwright.
9. **Upload Playwright Report**
   - Uses actions/upload-artifact@v4 to upload the Playwright test report.
   - Uploads the report only if the job is not cancelled.
   - Sets the artifact retention period to 30 days.

This setup ensures that E2E tests are run automatically on pull requests to the main branch, using a containerized environment for consistent testing and thorough reporting.

Workflow: SonarCloud

This workflow analyzes the frontend code using SonarCloud whenever there is a push to any files in the apps/frontend/ directory.

Triggering Events

- **push**: Runs the workflow on every push to files matching the pattern apps/frontend/**.

Job: SonarCloud-frontend

This job performs the SonarCloud analysis for the frontend application.

1. **Runs on ubuntu-latest**
   - The job is executed on the latest Ubuntu environment.
2. **Node.js Version Matrix**
   - The job runs using Node.js version 20 (defined in the matrix configuration).
3. **Steps to Execute**

1. **Checkout Code**
   - Uses the actions/checkout@v4 action to clone the repository.
   - persist-credentials: false prevents GitHub token from being stored.

- fetch-depth: 0 ensures the entire history is fetched.

2. **Setup pnpm**
   - Uses pnpm/action-setup@v4 to install pnpm version 8.
3. **Setup Go**
   - Uses actions/setup-go@v5 to install Go version 1.22.
   - Runs go version to verify the Go installation.
4. **Setup Node.js**
   - Uses actions/setup-node@v4 to install Node.js version 20.
   - Caches pnpm dependencies.
5. **Install Dependencies**
   - Runs pnpm install to install project dependencies.
6. **Install nx**
   - Runs pnpm install -g nx to install nx globally.
7. **Run Tests**
   - Runs nx test frontend --code-coverage to execute tests and generate code coverage for the frontend application.
8. **Fix Code Coverage Paths**
   - Changes directory to ./apps/frontend/coverage/frontend.
   - Runs a sed command to adjust paths in the lcov.info file, removing apps/frontend/ from the paths to match the project structure expected by SonarCloud.
9. **SonarCloud Scan**
   - Uses SonarSource/sonarcloud-github-action@master to run SonarCloud analysis on the frontend code.
   - Sets the projectBaseDir to apps/frontend to specify the root directory for the analysis.
   - Provides necessary environment variables:
     - GITHUB_TOKEN: Needed to get PR information.
     - SONAR_TOKEN: SonarCloud token for authentication, stored in GitHub secrets.

Workflow: Zap Security Scan

This workflow runs a ZAP security scan on pull requests targeting the main branch.

Triggering Events

- **pull_request**: Runs the workflow when a pull request targets the main branch.

Job: run-zap

This job performs a security scan using OWASP ZAP.

1. **Runs on ubuntu-latest**
    - The job is executed on the latest Ubuntu environment.
2. **Environment Variables**
    - Various environment variables are set from GitHub secrets for Playwright, database connections, and other configurations.
3. **Node.js Version Matrix**
    - The job runs using Node.js version 20 (defined in the matrix configuration).
4. **Steps to Execute**

1. **Checkout Code**
    - Uses the actions/checkout@v4 action to clone the repository.
    - persist-credentials: false prevents GitHub token from being stored.
    - fetch-depth: 0 ensures the entire history is fetched.
2. **Setup pnpm**
    - Uses pnpm/action-setup@v4 to install pnpm version 8.
3. **Setup Go**
    - Uses actions/setup-go@v5 to install Go version 1.22.
    - Runs go version to verify the Go installation.
4. **Setup Node.js**
    - Uses actions/setup-node@v4 to install Node.js version 20.
    - Caches pnpm dependencies.
5. **Set Git Identity**
    - Configures Git with a specific user name and email for versioning.
    - Sets the remote URL to include a GitHub token for authentication.
6. **Install Dependencies**
    - Runs pnpm install to install project dependencies.

7. **Install nx**
    - Runs pnpm install -g nx to install nx globally.
8. **Build Applications**
    - Runs pnpm run build:all to build all applications.
9. **Docker Login**
    - Logs into DockerHub using credentials stored in GitHub secrets.
10. **Run Docker**
    - Runs docker compose up --build -d to build and start Docker containers in detached mode.
11. **ZAP Scan Frontend**
    - Uses zaproxy/action-baseline@v0.12.0 to run a ZAP security scan.
    - Configures the scan with the following settings:
        - token: Uses the GitHub token for authentication.
        - docker_name: Specifies the ZAP Docker image to use.
        - target: Sets the target URL for the scan (http://localhost:80).

This setup ensures that a security scan using OWASP ZAP is automatically run for every pull request to the main branch, checking the security of the application in a consistent and automated manner.