Continuous Integration & Development (CI/CD)

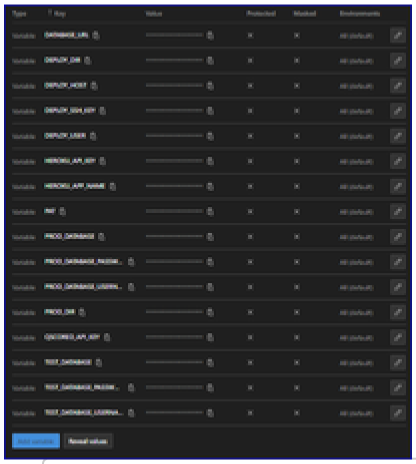
This article serves as a guide for the Continuous Integration & Development (CI/CD) process associated with our application.

📘CI/CD Stages & Jobs

👟 **Runner:**

gitlab-runner 13.0.1 (21cb397c) 2 on autoscale-runner.cs.dal.ca ce9tXy29

🔤 **Variables:**



Type	Key	Value	Protected	Masked	Environments
Variable	DATABASE_URL		×	×	All (default)
Variable	DEPLOY_DIR		×	×	All (default)
Variable	DEPLOY_HOST		×	×	All (default)
Variable	DEPLOY_SSH_KEY		×	×	All (default)
Variable	DEPLOY_USER		×	×	All (default)
Variable	HEROKU_API_KEY		×	×	All (default)
Variable	HEROKU_APP_NAME		×	×	All (default)
Variable	PAT		×	×	All (default)
Variable	PROD_DATABASE		×	×	All (default)
Variable	PROD_DATABASE_PASSW...		×	×	All (default)
Variable	PROD_DATABASE_USERN...		×	×	All (default)
Variable	PROD_DIR		×	×	All (default)
Variable	QSCORED_API_KEY		×	×	All (default)
Variable	TEST_DATABASE		×	×	All (default)
Variable	TEST_DATABASE_PASSW...		×	×	All (default)
Variable	TEST_DATABASE_USERNA...		×	×	All (default)

🎚 **Stages:**

build

test

code_quality

deploy

deploy-prod

Stage: Build

The Build stage merges the source code with its dependencies and compiles a runnable instance of software.

```
build:
  image: maven:3-jdk-8
  stage: build
  script:
    - mvn clean package -DskipTests=true
    - ls
  artifacts:
    paths:
      - target/*.jar
  only:
    refs:
      - feature_configuration
      - feature_UI
      - feature_backend
      - develop
      - merge_requests
      - main
```

Stage: Test

The Test stage includes the execution of automated tests to validate the correctness of code and the behavior of the software. In addition, the stage is responsible for the generation of test reports in XML format. The test stage acts as a safety net that prevents easily reproducible bugs from reaching the end-users. We have executed Unit Tests and Integration Tests for this application.

```
test:
  image: maven:3-jdk-8
  stage: test
  script:
    - mvn clean test
  artifacts:
    when: always
    reports:
      junit:
        - target/surefire-reports/TEST-*.xml
        - target/failsafe-reports/TEST-*.xml
  only:
    refs:
      - feature_configuration
      - feature_UI
      - feature_backend
      - develop
      - merge_requests
      - main
```

Stage: Code_Quality

This stage contains the script to download and execute DesigniteJava on our source code. Subsequently, the resultant XML file is uploaded to QScored to visualize the various types of smells and their corresponding occurrence in the code. Further information about Code Smells, our project's Quality score and Quality rank is available on the Confluence page: Code Smells & Refactoring.

```
code_quality:
  stage: code_quality
  variables:
    UPLOAD_QUALITY_REPORT: 'curl -X PUT -H "Authorization: Token
$QSCORED_API_KEY" -H "repository-link: $CI_PROJECT_URL" + -H "username:
bn489600@dal.ca" -H "Content-Type: mulitpart/form-data" --url
"https://qscored.com/api/upload/file.xml?
is_open_access=off&version=$CI_PIPELINE_IID&project_name=CSCI_5308_GROUP
_16" -F "file=@Designite_output/DesigniteAnalysis.xml"'
  script:
    - wget -O DesigniteJava.jar https://www.dropbox.com/s
/mwizkj8uhplz4x3/DesigniteJava.jar?dl=1
    - echo "/$CI_PROJECT_PATH"
    - ls -lrth
    #    - java -jar DesigniteJava.jar -i . -o Designite_output -f XML
    - java -jar DesigniteJava.jar -ci -repo $CI_PROJECT_PATH -pat $PAT -
host "git.cs.dal.ca"
    - 'eval "$UPLOAD_QUALITY_REPORT"'
  only:
    refs:
      - feature_configuration
      - feature_UI
      - feature_backend
      - develop
      - merge_requests
      - code_quality_danny
      - heroku_configuration
      - ride_share_benny
      - main
```

Stage: Deploy (Pre-Production)

The Pre-production environment is a replica of Production environment with similar server configurations, however, clients are not involved in this environment.

```
deploy:
  image: ruby:latest
  stage: deploy
  #  tags:
  #    - ugrad
  before_script:
    - 'command -v ssh-agent >/dev/null || ( apt-get update -y && apt-
get install openssh-client -y )'
    - eval $(ssh-agent -s)
    - echo "$DEPLOY_SSH_KEY" | tr -d '\r' | ssh-add -
  script:
    - apt-get update -qy
    - apt-get install -y ruby-dev
    - gem install dpl
    - dpl --provider=heroku --app=$HEROKU_APP_NAME --api-
key=$HEROKU_API_KEY
  only:
    - develop
```

Stage: Deploy (Production)

The deploy stage initiates a SSH agent and pushes the source code onto Heroku. Deploys to production can only occur when the latest staging build is green (i.e. all build steps passed without error).

```
deploy-prod:
  image: ruby:latest
  stage: deploy
  before_script:
    - 'command -v ssh-agent >/dev/null || ( apt-get update -y && apt-
get install openssh-client -y )'
    - eval $(ssh-agent -s)
    - echo "$DEPLOY_SSH_KEY" | tr -d '\r' | ssh-add -
  script:
    - apt-get update -qy
    - apt-get install -y ruby-dev
    - gem install dpl
    - dpl --provider=heroku --app=$HEROKU_APP_NAME --api-
key=$HEROKU_API_KEY
#    TODO: Add Production App Environment Variables
  only:
    - main
```

📋 Related articles

Articles relevant to Build & Deployment or Usage Scenarios relevant to the application, can be accessed through the following pages:

- Continuous Integration & Development (CI/CD)

- Usage Scenario
- Build & Deployment