Sprawozdanie Jenkins pipeline

Kamil Kruczek, Informatyka Techniczna, GCL04

Cel projektu

Celem projektu było utworzenie automatycznego pipeline'u, który buduje kod, testuje go, odtwarza i publikuje. Osiągnięcie tego celu zostało przeprowadzone przy pomocy konteneryzacji Dockerem obsługiwanej przez Jenkinsa.

Pipeline

1. **Prebuild**

```
pipeline {
    agent any
    parameters {
        booleanParam(name: 'PROMOTE', defaultValue: true, description: '')
    }
    stages {
        stage("Prebuild") {
            steps {
                script {
                    sh 'mkdir -p io_dir'
                }
            }
            post {
                success {
                    sh 'echo Prebuild successful'
                }
                failure {
                    sh 'echo Prebuild failed'
                }
            }
        }
```

Powyższy fragment pliku *Jenkinsfile* ustawia parametr typu logicznego - *PROMOTE* - i ustawia jego wartość domyślną na *true*. Następnie zaczyna się pierwszy krok pipeline'u: *Prebuild*. Krok ten jest odpowiedzialny za stworzenie folderu służącego jako wolumin wyjściowy i wejściowy kontenerów. Jesteśmy również informowani o statusie zakończenia każdego z kroków.

2. **Build**

```
stage('Build') {
    steps {
        script {
            sh '''
            echo Build

            docker build -f dockerbuild -t devbuilder . --no-cache
            docker run -v \$(pwd)/io_dir:/docker_vol devbuilder
            '''
        }
    }
    post {
        success {
            sh 'echo Build successful'
        }
        failure {
            sh 'echo Build failed'
        }
    }
}
```

Następnym krokiem jest *Build*, który odpowiada za budowanie obrazu Dockera z pliku *dockerbuild*:

```
1 FROM node
2 RUN git clone https://github.com/stemmlerjs/simple-typescript-starter.git
3 WORKDIR /simple-typescript-starter/
4 RUN npm install && npm run build
5 RUN npm pack
6 RUN mkdir /artifacts
7 RUN cp typescript-starter-1.0.0.tgz /artifacts
8 CMD cp typescript-starter-1.0.0.tgz /docker_vol
```

Na obraz kopiowane jest wybrane do projektu repozytorium, a następnie instalowane są jego dependencje oraz uruchamiany jest build. Pomyślnie zbudowany kod zostaje zapakowany przez polecenie *npm pack*, wysłany do folderu *artifacts* i folderu z woluminem.

3. **Test**

```
stage('Test') {
    steps {
        script {
            sh '''
            echo Test
            docker build . -f dockertest -t devtester
            docker run devtester
            '''
        }
    }
    post {
        success {
            sh 'echo Tests successful'
        }
        failure {
            sh 'echo Tests failed'
        }
    }
}
```

W tym etapie zostają przeprowadzone testy kodu za pomocą komendy *npm run test* na obrazie zbudowanym na podstawie obrazu z poprzedniego kroku. Plik *dockertest*:

```
1 FROM devbuilder:latest
2 WORKDIR /simple-typescript-starter/
3 RUN npm run test
```

4. **Deploy**

```
stage('Deploy') {
    steps {
        script {
            sh '''
            echo Deploy

            docker build . -f dockerdeploy -t devdeploy
            docker run -d -v \$(pwd)/io_dir:/docker_vol devdeploy
            '''
        }
    }
    post {
        success {
            sh 'echo Deploy successful'
        }
        failure {
            sh 'echo Deploy failed'
        }
    }
}
```

W tym etapie, jeśli poprzednie kroki zakończyły się sukcesem, kod zostaje przygotowany do publikacji, tworzony jest obraz na podstawie obrazu z kroku Build. Plik *dockerdeploy*:

```
1 FROM devbuilder:latest
2 RUN mkdir /deploy
3 RUN cp /artifacts/typescript-starter-1.0.0.tgz /deploy/
```

5. **Publish**

```
stage('Publish') {
    when
        {
            expression {return params.PROMOTE}
        }
    steps {
        script {
                archiveArtifacts artifacts: 'io_dir/typescript-starter-1.0.0.tgz', fingerprint: true
        }
    }
}
```

Gdy parametr logiczny *PROMOTE* ma wartość *true*, a wszystkie poprzednie kroki zakończyły się powodzeniem, artefakt zbudowany w kroku Build zostaje publikowany.

Pipeline devops

Dodaj opis

Wyłącz projekt

Last Successful Artifacts

typescript-starter-1.0.0.tgz 2.56 KB view

Recent Changes

Stage View

	Declarative: Checkout SCM	Prebuild	Build	Test	Deploy	Publish
Average stage times: (Average full run time: ~3min 16s)	1s	1s	2min 47s	13s	9s	423ms
#12 Jun 17 13:48 No Changes	1s	973ms	2min 49s	15s	4s	1s

Diagram aktywności