



Firebase: CI/CD

Continuous integration (CI) i **Continuous Delivery (CD)** to zbiór praktyk, które pozwolą nam na częste (niemalże automatyczne) dostarczanie sprawdzonych (przetestowanych) zmian w kodzie.

- Proces CI odpowiada za integrowanie oraz weryfikowanie dodanych zmian w kodzie.
- Proces CD odpowiada za automatyczne wgrywanie dodanych i sprawdzonych wcześniej zmian.



Źródło: <https://www.redhat.com/>



Pokażę Ci w jaki sposób skonfigurować projekt korzystając z Firebase, żeby po każdej zmianie w kodzie dodanej do repozytorium na GitLabie, Twój projekt budował się automatycznie (być może uruchamiał też testy) i jeżeli wszystko pójdzie pomyślnie - wgrywał zmiany bezpośrednio na środowisko produkcyjne!

KROK 0: Wymagania wstępne

1. Posiadasz już utworzony projekt Firebase.

2. Projekt Firebase działa prawidłowo przy lokalnym uruchamianiu. Tzn. działa prawidłowo przy wywołaniu polecenia `firebase serve`.

KROK 1: Umieść projekt w repozytorium na GitLab

Jeżeli masz już utworzone repozytorium z projektem (oraz wgranym najnowszym kodem) na GitLabie możesz spokojnie pominąć cały ten krok 😊

1. Utwórz repozytorium na GitLab.

F Firebase-ARPFront1PL 🔒

Project ID: 36416016

6 Commits 1 Branch 0 Tags 41 KB Project Storage

main Find file Web IDE Clone

Arkadiusz Sas authored 3 minutes ago 292b0b78

README CI/CD configuration Add LICENSE Add CHANGELOG Add CONTRIBUTING Add Kubernetes cluster

Configure Integrations

Name	Last commit	Last update
public	:)	3 minutes ago
.firebaserc	Init the project	40 minutes ago
.gitignore	Init the project	40 minutes ago
.gitlab-ci.yml	:)	8 minutes ago
README.md	Initial commit	48 minutes ago
firebase.json	Init the project	40 minutes ago
package-lock.json	Init the project	40 minutes ago
package.json	Init the project	40 minutes ago

2. Sklonuj repozytorium na lokalny dysk:

```
git clone <link do repozytorium>.git
```

3. Dodaj pliki z Twoim projektem do powstałego folderu.

4. Wgraj pliki z Twoim projektem do zdalnego repozytorium GitLab:

```
git add .
git commit -m "Added the project files"
git push
```

KROK 2: Pobranie tokenu

Krok zakłada, że Twój projekt jest lokalnie poprawnie skonfigurowany oraz posiadasz zainstalowane `firebase-tools`. Jeżeli używane przez nas podczas wcześniejszych lekcji polecenie `firebase serve` Ci działało to znaczy, że wszystko będzie okej! 😊

1. W folderze z projektem uruchom poniższe polecenie. Firebase przekieruje Cię do przeglądarki, gdzie będziesz mieć możliwość zalogowania się na swoje konto. W wyniku działania tego polecenia otrzymasz swój token CI (continuous integration). **Zapisz go - będzie nam jeszcze potrzebny.**

```
firebase login:ci
```

```
PS C:\Users\konta\OneDrive\Pulpit\Szkolenia\Szkolenie Firebase\firebase-arpfront1pl> firebase login:ci

Visit this URL on this device to log in:
https://accounts.google.com/o/oauth2/auth?client_id=(...)

Waiting for authentication...

+ Success! Use this token to login on a CI server:

<<TU BĘDZIE TWÓJ TOKEN DO UWIERZETLNIENIA CI>>

Example: firebase deploy --token "$FIREBASE_TOKEN"
```

KROK 3: Konfiguracja GitLab CI

1. W repozytorium na GitLab przejdź do `Settings → CI/CD → Variables` oraz rozwiń tę sekcję.
2. Dodaj klucz `FIREBASE_DEPLOY_KEY` oraz **token** otrzymany w Terminalu.

Variables

[Collapse](#)

Variables store information, like passwords and secret keys, that you can use in job scripts. [Learn more](#).

Variables can be:

- **Protected:** Only exposed to protected branches or protected tags.
- **Masked:** Hidden in job logs. Must match masking requirements. [Learn more](#).

Environment variables are configured by your administrator to be [protected](#) by default.

Type	↑ Key	Value	Protected	Masked	Environments
Variable	FIREBASE_DEPLOY_KEY	*****	✓	✗	All (default)

[Add variable](#)

[Reveal values](#)

KROK 4: Definiowanie skryptu CI/CD

1. W lokalnym folderze z projektem Firebase utwórz plik [.gitlab-ci.yml](#).
2. Utwórz nazwę środiska dla Twojego projektu Firebase w Terminalu:

```
firebase use --add fir-application-6d29a production
```

3. Wprowadź zawartość skryptu do utworzonego pliku [.gitlab-ci.yml](#).

```
image: node:latest

cache:
  paths:
    - node_modules/

deploy_production:
  stage: deploy
  environment: Production
  only:
    - main
  script:
    - npm install -g firebase-tools
    - npm install
    - firebase use --token $FIREBASE_DEPLOY_KEY <>ID_TWÓJEGO_PROJEKTU_FIREBASE>>
    - firebase deploy -m "Pipeline $CI_PIPELINE_ID, build $CI_BUILD_ID" --non-interactive --token $FIREBASE_DEPLOY_KEY
```

3. Dodaj zmiany do zdalnego repozytorium na GitLab (Patrz: Krok 1 → Punkt 4).
4. Uwaga! Chcąc sprawdzić, czy nasz proces CI/CD działa prawidłowo konieczne jest wcześniejsze podpięcie numerów karty do konta GitLab. Jest to zabezpieczenie przed nadużywaniem zasobów GitLaba, operacja jest w całości darmowa.

5. Jeżeli wszystko zostało skonfigurowane prawidłowo w sekcji CI/CD na GitLab powinniśmy otrzymać prawidłowo zbudowany oraz wydeployowany projekt 🎉

```
48 ✓ hosting[fir-application-6d29a]: version finalized
49 i  hosting[fir-application-6d29a]: releasing new version...
50 ✓  hosting[fir-application-6d29a]: release complete
51 ✓ Deploy complete!
52 Project Console: https://console.firebaseio.google.com/project/fir-application-6d29a/overview
53 Hosting URL: https://fir-application-6d29a.web.app
54 Saving cache for successful job
55 Creating cache default-protected...
56 node_modules/: found 11137 matching files and directories
57 Uploading cache.zip to https://storage.googleapis.com/gitlab-com-runners-cache/project/36416016/default-protected
58 Created cache
59 Cleaning up project directory and file based variables
60 Job succeeded
```

The screenshot shows the GitLab CI/CD pipeline interface. At the top, there are tabs for 'All' (selected), 'Finished' (4), 'Branches', and 'Tags'. On the right, there are buttons for 'Clear runner caches', 'CI lint', and 'Run pipeline'. Below this is a search bar labeled 'Filter pipelines' with a magnifying glass icon and a dropdown for 'Show Pipeline ID'. The main area displays a table with columns: Status, Pipeline, Triggerer, and Stages. A single row is shown, indicating a job that has passed ('passed') and completed ('00:03:47' ago). The pipeline ID is '#546069852', it triggered from 'main' branch with commit 'e7e8dc39', and the stage status is green. There is also a three-dot menu icon.

ZADANIA

1. Utwórz repozytorium na GitLabie dla swojego projektu.
2. Pobierz oraz zapisz w GitLabie swój token do procesu CI/CD.
3. Dodaj proces CI/CD do swojej aplikacji. Możesz skorzystać z kopii swojego projektu, żeby uniknąć sytuacji, że limit czas na GitLabie został przekroczony.