**Tugas Day 13**

Integrasi & Implementasi CI/CD

pada Gitlab Dan Heroku

**Intro**

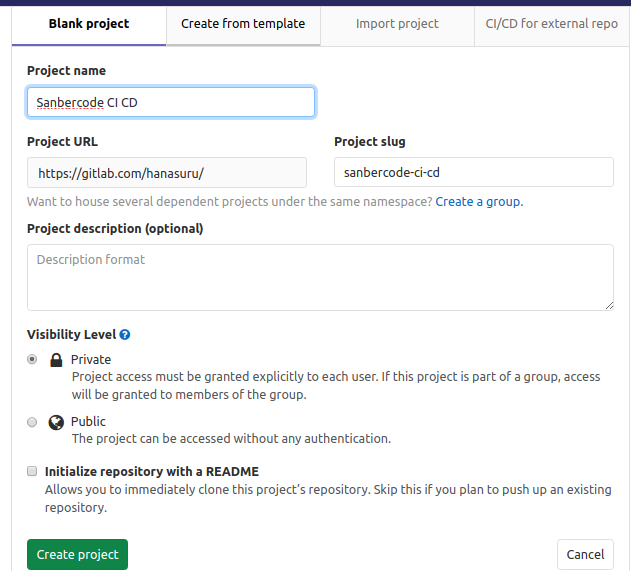
***Continuous integration*** / ***continuous delivery*** (CI/CD) merupakan salah satu *workflow* yang sering diaplikasikan dalam *software* *engineering*. Kedua aspek ini menjembatani kesenjangan yang terjadi pada siklus *development* dengan menekankan otomatisasi pada proses ***building****,* ***testing***, dan***deployment*** dalam sebuah aplikasi.

**Skenario**

Pada skenario kali ini, kita diminta untuk mengaplikasikan **CI/CD** pada platform **Gitlab** dan **Heroku** dengan **Laravel** sebagai base framework yang digunakan. Seperti biasa, kita diberikan sebuah [*repository*](https://gitlab.com/muhammadyaqin/laravel5_sample.git) yang memuat kode dari aplikasi yang nantinya akan kita *deploy*. Adapun berikut langkah-langkahnya:

**Inisialisasi Gitlab Project**

Pertama-tama pastikan untuk membuat akun Gitlab terlebih dahulu. Untuk membuat sebuah Gitlab project, dapat kita lakukan dengan mengakses url **gitlab.com/projects/new**. Dari sini kita isikan nama project, misal saja “*Sanbercode CI CD* “.



Alternatifnya, kita juga dapat melakukan hal serupa melalui ***command-line*** dengan perintah **lab**sebagai berikut

|  |
| --- |
| $ curl -s https://raw.githubusercontent.com/zaquestion/lab/master/install.sh | sudo sh  $ sudo mv /usr/local/bin/lab /usr/bin/lab  $ lab project create -n "Sanbercode CI CD" |

**Menambahkan Project Laravel**

Untuk menambahkan project laravel yang telah ditentukan, dapat digunakan perintah sebagai berikut:

|  |
| --- |
| $ git clone https://gitlab.com/muhammadyaqin/laravel5\_sample.git sanbercode-ci-cd  $ cd sanbercode-ci-cd |

Selanjutnya kita atur ulang **remote origin** dari repository dengan url project yang telah kita buat sebelumnya.

|  |
| --- |
| $ rm -rf .git  $ git init  $ git remote add origin https://gitlab.com/hanasuru/sanbercode-ci-cd.git |

Saatnya melakukan *commit* untuk menyimpan perubahan pada repository

|  |
| --- |
| $ git add .  $ git commit -m "Add Laravel Project" |

**Mengaktifkan Pipelines pada Repository**

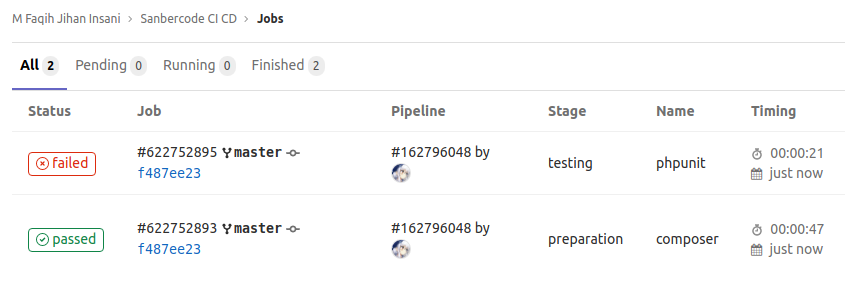
Secara umum, Gitlab CI/CD menggunakan file .gitlab-ci.yml untuk mengaktifkan pipelines yang akan aktif ketika terjadi perubahan pada kode. Pipelines ini terbagi atas beberapa **stage** yang masing-masing menjalankan sebuah **jobs** yang dieksekusi oleh **Github Runner**.

Sebagai permulaan, kita akan mendeklarasikan stage **preparation** dan **testing** pada .gitlab-ci.yml

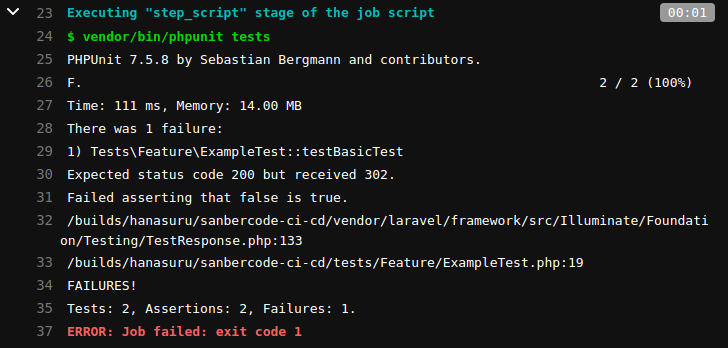
|  |
| --- |
| $ vim .gitlab-ci.yml |
| image: php:7.2-alpine  stages:  - preparation  - testing  composer:  stage: preparation  script:  - curl -sS https://getcomposer.org/installer | php  - mv .env.example .env  - php composer.phar install --prefer-dist --no-ansi --no-interaction --no-progress --no-scripts  - php artisan key:generate  artifacts:  paths:  - vendor/  - .env  expire\_in: 1 days  when: always  cache:  paths:  - vendor/  phpunit:  stage: testing  dependencies:  - composer  script:  - vendor/bin/phpunit tests |

Kemudian, lakukan *commit* kembali untuk menyimpan perubahan dan tiba saatnya untuk melakukan *push* ke Gitlab repository.

|  |
| --- |
| $ git add .  $ git commit -m "Add gitlab CI for preparation & testing"  $ git push origin master |



Hasilnya, kita akan melihat bahwa stage **preparation** berhasil lolos, sedangkan **testing** tidak lolos pengujian. Dari sini, kita akan mengecek terkait penyebab kegagalan dari stage **testing** seperti di bawah ini



Seperti yang kita lihat, bahwa **unit testing** pada file **tests/Feature/ExampleTest.php** tidak berhasil terpenuhi. Untuk itu kita lakukan debugging pada file tersebut

|  |
| --- |
| $ cat tests/Feature/ExampleTest.php |
| ..  ..  public function testBasicTest()  {  **$response = $this->get('/');**  **$response->assertStatus(200);**  }  ..  .. |

Di sini kita mengetahui bahwa, **unit testing** ingin mengecek ketika **route** ‘/ ’ diakses, apakah akan diperoleh response code **200**. Untuk itu kita *trace* kembali route yang didefinisikan pada file **routes/web.php**

|  |
| --- |
| $ cat routes/web.php |
| ..  ..  Route::get('/', function () {  **return redirect('/students');**  });  Route::resource('students','StudentController'); |

Berdasarkan kedua file tersebut, akhirnya kita mengetahui bahwa gagalnya **unit testing** disebabkan oleh fungsi **redirect()** memberikan **temporary redirect** yang ekivalen dengan response code **302.**

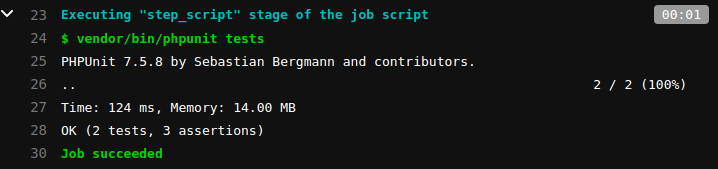
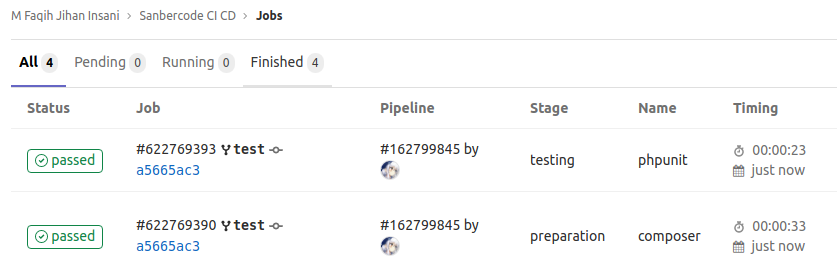
Untuk memperbaiki hal tersebut, kita dapat mengubah lines assertStatus(200) sebagai berikut

|  |
| --- |
| $ vim tests/Feature/ExampleTest.php |
| ..  ..  public function testBasicTest()  {  **$response = $this->get('/');**  **$response->assertRedirect('/students');**  }  ..  .. |

Terakhir lakukan *commit* untuk menyimpan perubahan.

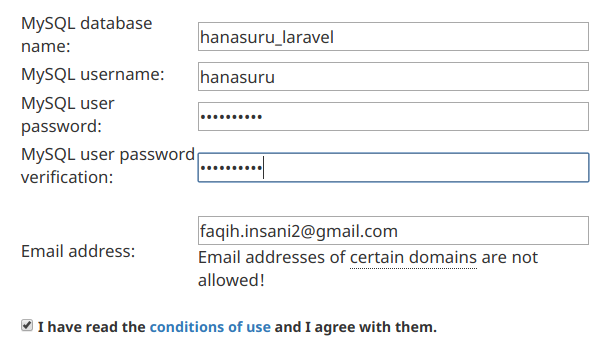
|  |
| --- |
| $ git add .  $ git commit -m "Fix feature test assertion"  $ git push origin master |

Hasilnya, kita mendapati bahwa kedua **stage** lolos pengujian

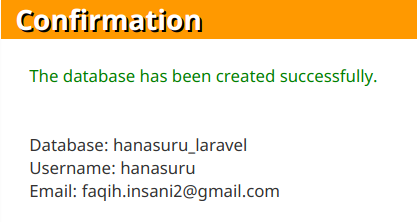


**Inisialisasi Database**

Pada tahap ini, kita akan memanfaatkan layanan db-server, yaitu [db4free](https://db4free.net/signup.php), sebagai pengganti dari MySQL server pada **Heroku**. Di sana, kita cukup memasukkan beberapa informasi terkait **DB\_NAME**, **DB\_USER**,dan **DB\_USER\_PASS** sebagaimana berikut ini



Setelah itu, kita akan menerima email yang berisi link untuk melakukan aktivasi database



Kendati demikian, perlu diingat bahwa layanan ini hanya memberikan **2 buah database** untuk **1 buah email**.

**Inisialisasi Server Staging**

Sesaat setelah **unit testing** lolos pengujian, langkah selanjutnya adalah melakukan deployment ke server **staging**. Dalam konteks ini, **Heroku** akan kita gunakan sebagai server **staging** sebelum nantinya siap untuk dirilis di server **production**.

Di sini kita menggunakan **heroku-cli** untuk menginisasi instance **Heroku APP dan API\_KEY**

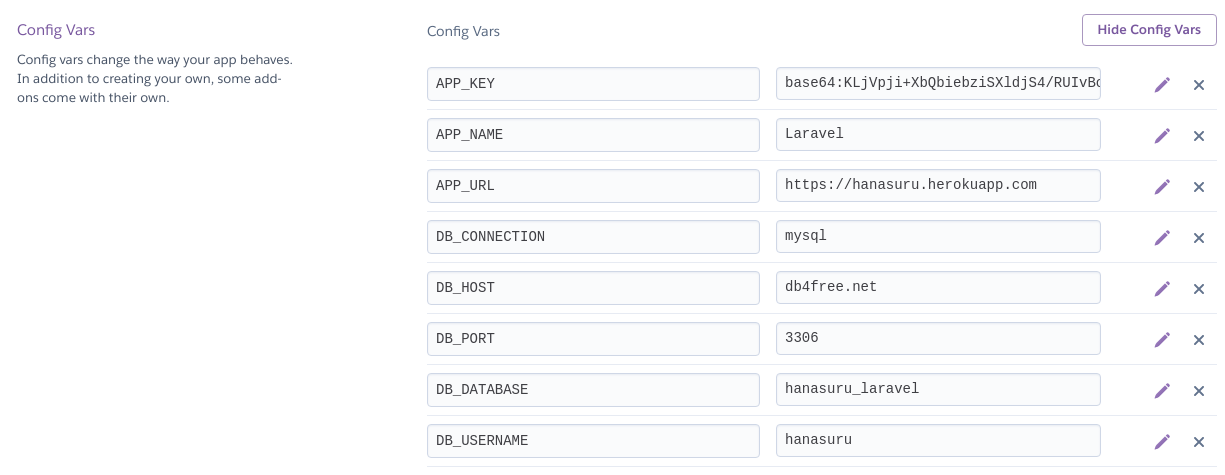
|  |
| --- |
| $ wget https://cli-assets.heroku.com/heroku-linux-x64.tar.gz  $ sudo tar xvzf heroku-linux-x64.tar.gz -C /usr/local/lib  $ sudo ln -s /usr/local/lib/heroku/bin/heroku /usr/bin/heroku  $ heroku create hanasuru  $ heroku auth:token |
|  |

**Konfigurasi CI/CD pada Heroku**

Kemudian, buka [dashboard](https://dashboard.heroku.com/apps/) apps untuk mengakses panel setting. Di sini kita akan melakukan input **config vars** yang digunakan sebagai pengganti file **.env**

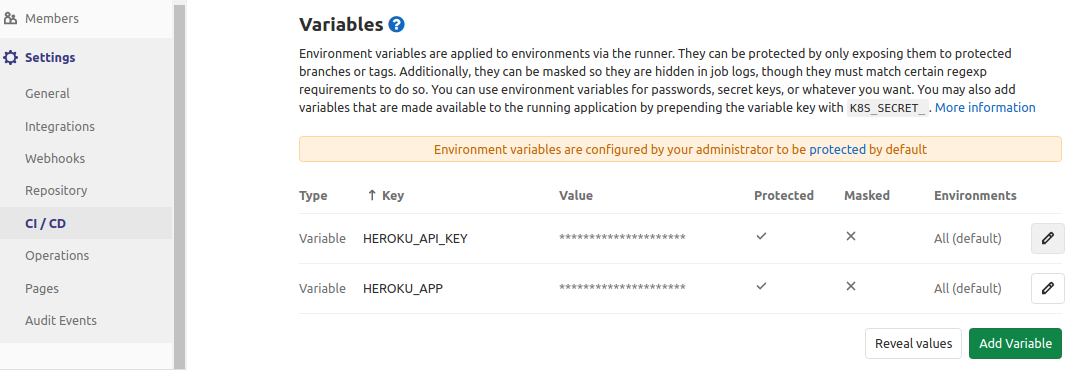
Adapun berikut ini merupakan daftar variabel yang dapat dimasukkan

|  |
| --- |
| APP\_NAME=Laravel  APP\_KEY=base64:KLjVpji+XbQbiebziSXldjS4/RUIvBoZx6zuiCiQxgE=  APP\_URL=https://hanasuru.herokuapp.com  DB\_CONNECTION=mysql  DB\_HOST=db4free.net  DB\_PORT=3306  DB\_DATABASE=hanasuru\_laravel  DB\_USERNAME=hanasuru  DB\_PASSWORD=12345qwert |



**Konfigurasi CI/CD pada Gitlab**

Sebagaimana yang kita lakukan pada **Heroku,** di sini kita juga akan mendefinisikan *environment* *variable* yang dapat digunakan oleh **Github Runner**. Adapun pengaturan ini dapat kita temukan pada **Setting > CI/CD > Variables.** Perlu diingat bahwa, **HEROKU\_APP** merupakan app-name dari **Heroku**; **HEROKU\_API\_KEY** merupakan token yang diperoleh dari perintah heroku auth:key



**Deployment Project**

Setelah semua *environment* variable telah diatur, saatnya kita menambahkan stage **deployment** pada file .gitlab-ci.yml

|  |
| --- |
| $ vim .gitlab-ci.yml |
| image: php:7.2-alpine  stages:  - preparation  - testing  - deployment  composer:  stage: preparation  script:  - curl -sS https://getcomposer.org/installer | php  - mv .env.example .env  - php composer.phar install --prefer-dist --no-ansi --no-interaction --no-progress --no-scripts  - php artisan key:generate  artifacts:  paths:  - vendor/  - .env  expire\_in: 1 days  when: always  cache:  paths:  - vendor/  phpunit:  stage: testing  dependencies:  - composer  script:  - vendor/bin/phpunit tests  heroku:  stage: deployment  dependencies:  - composer  - phpunit  script:  - apk update && apk upgrade  - apk add git ruby  - gem install dpl json\_pure  - dpl --provider=heroku --app=$HEROKU\_APP --api-key=$HEROKU\_API\_KEY  only:  - master |

Terakhir, kita buat file **Procfile** yang digunakan oleh **Heroku** untuk mengeksekusi command tertentu saat aplikasi dijalankan

|  |
| --- |
| $ vim Procfile |
| web: vendor/bin/heroku-php-apache2 public/  release: php artisan migrate --force |

Barulah kemudian, kita lakukan commit dan push untuk menyimpan perubahan terhadap repository

|  |
| --- |
| $ git add .  $ git commit -m "Add Procfile & Update gitlab CI for deployment"  $ git push origin master |

Hasilnya, kita akan mendapati bahwa semua **stage** berhasil lolos dari pengujian dan berhasil di deploy pada **Heroku** ([http://hanasuru.herokuapp.com](http://hanasuru.herokuapp.com/students)/)

