



ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỀ TÀI

CONTINUOUS INTEGRATION

TRAVIS CI

Nhóm thực hiện:	Họ tên	MSSV
	1. Nguyễn Thái Hòa	1642019
	2. Hà Nguyễn Thái Học	1642021
	3. Dương Tấn Huỳnh Phong	1642049
	4. Nguyễn Xuân Phúc	1642051
Lớp:	16HCB	
Môn học:	Công cụ kiểm chứng phần mềm	

Mục lục

I. GIỚI THIỆU TRAVIS CI.....	3
II. PHẠM VI HỖ TRỢ.....	3
1. Hệ điều hành.....	3
2. Ngôn ngữ lập trình.....	3
III. CHỨC NĂNG CHÍNH	4
IV. Demo Travis CI.....	5
1. Integrate Travis CI cho Github.....	5
2. Integrate Travis CI với Slack.....	15
3. Automatically deploy cho heroku với Travis CI	18

I. GIỚI THIỆU TRAVIS CI

Travis CI là một CI (continuous integration – tích hợp liên tục) giúp build, chạy unit test, deploy lên server một cách tự động mỗi khi push code mới hoặc tạo pull request.

Travis CI thường được tích hợp chung với GitHub và Heroku để giúp trong việc build, chạy unit test và deploy sau khi build và chạy unit test thành công.

II. PHẠM VI HỖ TRỢ

1. Hệ điều hành

- Ubuntu
- MacOS

2. Ngôn ngữ lập trình

- Android
- C/C++
- C#
- Elixir
- Go
- Ruby
- Java
- JavaScript (Node.js only)
- Objective-C
- PHP
- Python
- Rust
- Scala
- Swift
- Visual Basic

- Perl
- Perl6
- F#
- Clojure
- Crystal
- D
- Dart
- Erlang
- Groovy
- Haskell
- Haxe
- Julia
- Nix
- Smalltalk

III. CHỨC NĂNG CHÍNH

- Build Project
 - Travis CI sẽ build project khi push code mới hoặc tạo pull request.
- Run Unit Test
 - Travis CI sẽ run unit test khi push code mới hoặc tạo pull request.
 - Thông thường sẽ được cài đặt chạy sau khi build thành công.
- Deploy Project
 - Travis CI sẽ deploy project lên host đã được cài đặt sẵn.
 - Thông thường sẽ được cài đặt chạy sau khi build và chạy unit test thành công.
- Cron job
 - Travis CI cho phép tạo thời gian biểu (schedule) để chạy tự động một tiến trình nào đó trong một thời gian cụ thể.

– Integration & Notification

- Travis CI cho phép tích hợp vào bên thứ 3 như **Slack**, **GitHub**, **Heroku**, ... và thông báo kết quả build, chạy unit test đến các bên thứ 3.

IV. Demo Travis CI

Demo Travis CI bằng Ruby on Rails.

1. Integrate Travis CI cho Github

– Bước 1 – Tạo repository trên Github

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

 zgid123 ▾

Repository name

/ demo-travis-ci

Great repository names are short and memorable. Need inspiration? How about **glowing-spork**.

Description (optional)

demo test travis ci

☒  **Public**

Anyone can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

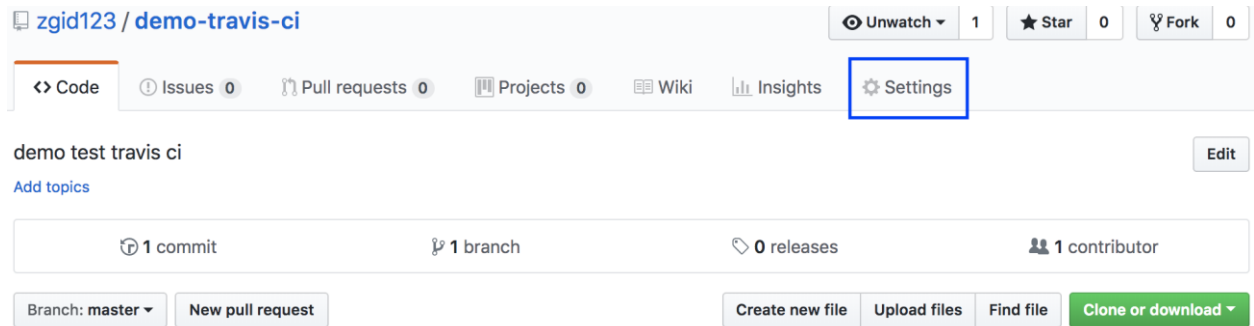
Add a license: **None** ▾



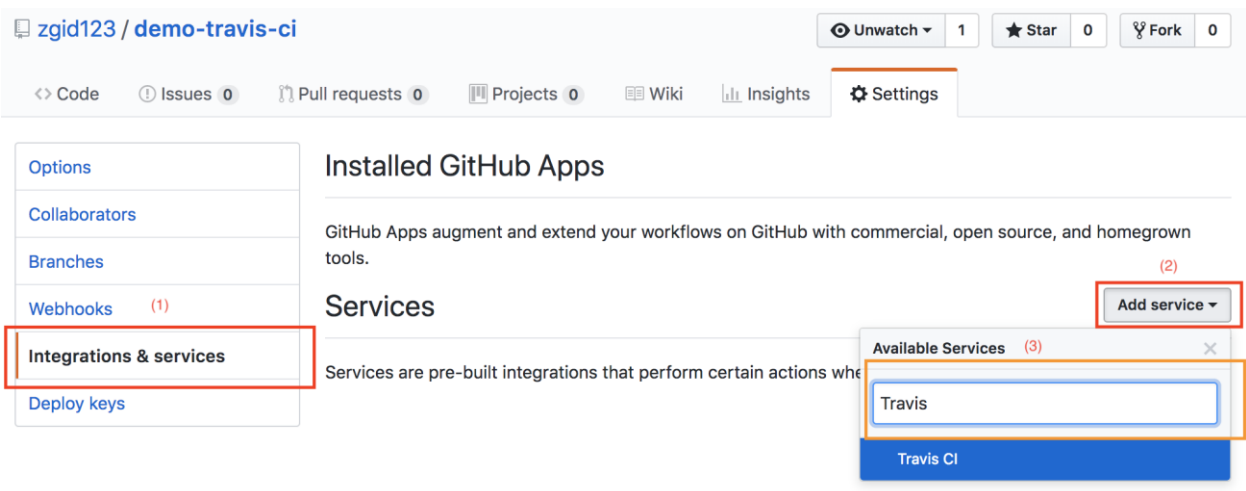
Create repository

Sau đó push project lên Github.

– Bước 2 – Integrate Travis CI cho Github



Chọn mục Settings.



Chọn (1), sau đó chọn (2) và gõ **Travis** vào (3) và chọn kết quả search.

Options

Collaborators

Branches

Webhooks

Integrations & services

Deploy keys

Services / Add Travis CI

Travis CI is a distributed continuous integration service.

By enabling this hook, Travis CI will listen to push and pull request events to trigger new builds.

For more details about Travis, go to <http://docs.travis-ci.com>.

Automatic configuration from Travis CI

We recommend using the Travis profile page at <https://travis-ci.org/profile> to manage your hooks.
For private repositories, use <https://travis-ci.com/profile>.

Travis CI Status

Travis CI status page: <http://status.travis-ci.com>

User

Token

Domain

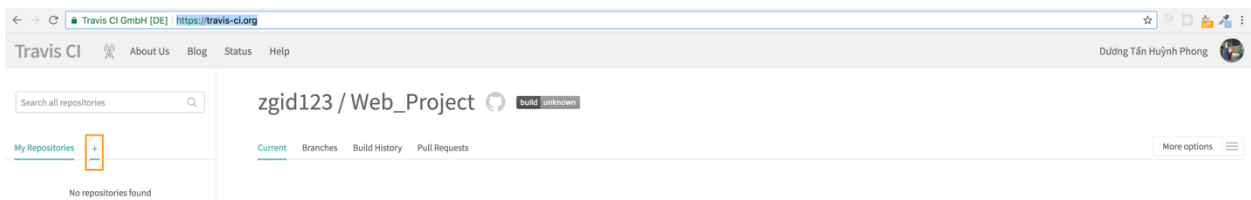
☒ Active

We will run this service when an event is triggered.

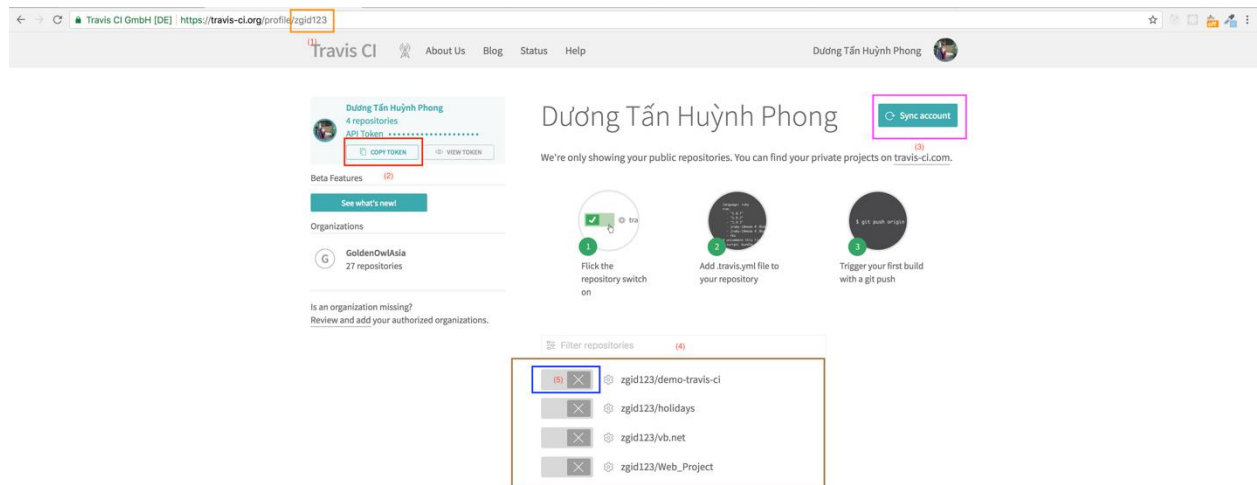
Add service

Để lấy **User** và **Token**, thực hiện các bước sau:

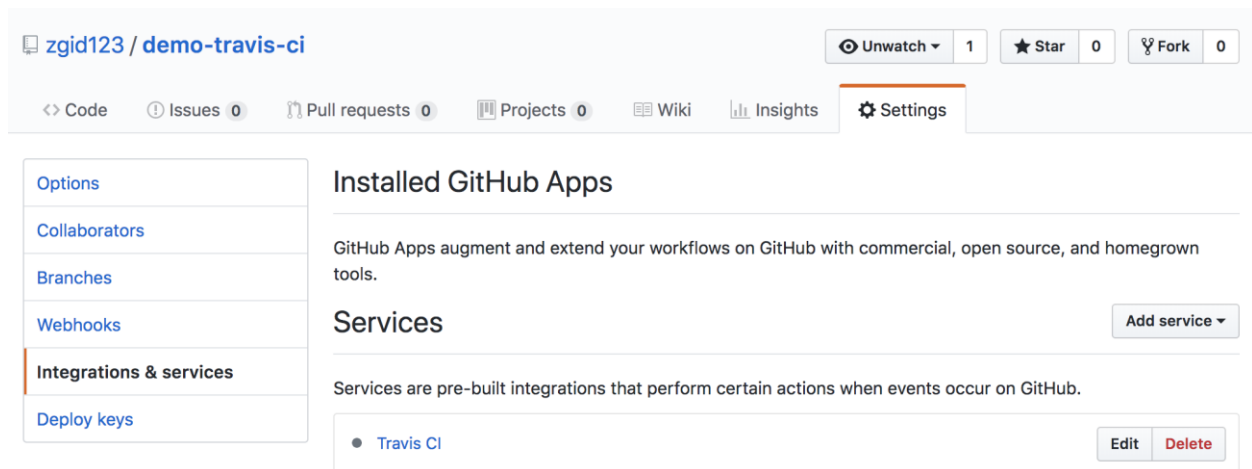
- Vào trang <https://travis-ci.org/>, đăng nhập tài khoản Travis.
- Chọn **Add New Repository** như hình.



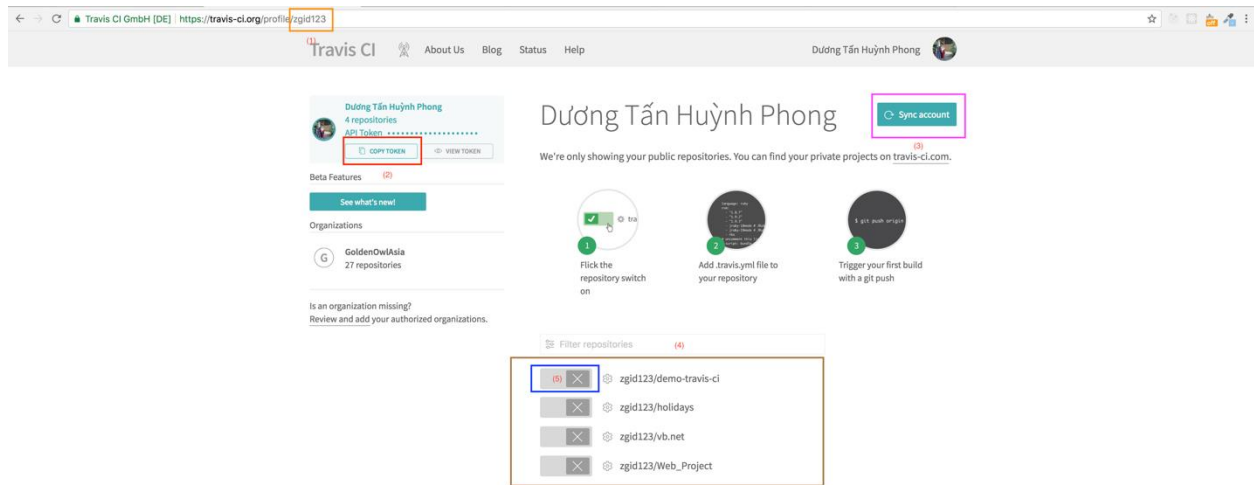
- Giá trị (1) là **User**, nhấn vào (2) để copy **Token**.



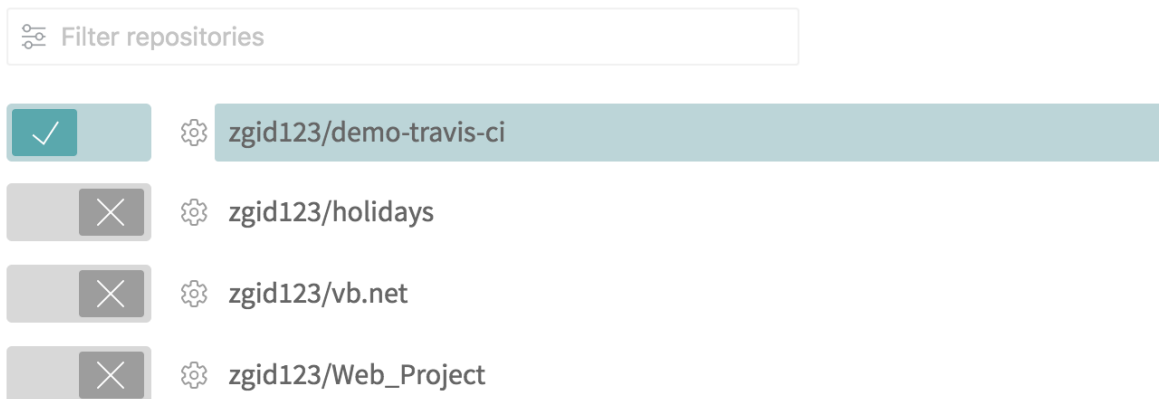
- Điền **User** và **Token** vào, sau đó nhấn nút **Add service**.



– Bước 3 – Enable repository của Github trên Travis CI



(4) là danh sách các repositories của Github, chọn (5) ngay dòng của repository cần integrate với Github.



Lưu ý: Trường hợp nếu chưa thấy repository cần integrate, chọn (3) sau đó reload trang để refresh toàn bộ repositories, nếu vẫn chưa thấy thực hiện lại cho đến khi thấy được repository đó.

– Bước 4 – Config Travis CI cho project và cài đặt unit test và functional testing

Thực hiện các bước sau:

- Tạo file **.travis.yml** và thêm config sau:

```
language: ruby
gemfile:
  - gemfiles/rails_4.2.0.gemfile
rvm:
  - 2.4.2
before_install:
  - gem update bundler
  - gem update --system
matrix:
  include:
    - rvm: 2.4.2
      gemfile: gemfiles/rails_5.0.0.gemfile
services:
  - postgresql
script:
  - bundle install
  - bundle exec rspec
before_script:
  - psql -c 'create database test_travis_ci_test;' -U postgres
  - cp config/database.yml.sample config/database.yml
```

- (i) Language: là ngôn ngữ lập trình project đang sử dụng.
- (ii) Before_install: các lệnh chạy trước khi thực hiện Script.
- (iii) Services: là các dịch vụ chạy kèm, trong ví dụ là postgresql để chạy unit test và functional testing với data của postgresql.
- (iv) Script: là các lệnh chính cần chạy, trong ví dụ gồm **bundle install** – build project và **bundle exec rspec** – chạy unit test và functional testing.
- (v) Before_script: là các lệnh chạy trước **script** và sau **before_install**.

- Viết unit test

```
describe User do
  let!(:user) { create(:user, :admin) }

  describe '#admin?' do
    subject { user.admin? }

    context 'is admin' do
      it { is_expected.to eq true }
    end

    context 'is not admin' do
      before { user.update(role: User.roles[:vip]) }

      it { is_expected.to eq false }
    end
  end

  describe '#vip?' do
    subject { user.vip? }

    context 'is vip' do
      before { user.update(role: User.roles[:vip]) }

      it { is_expected.to eq true }
    end

    context 'is not vip' do
      it { is_expected.to eq false }
    end
  end

  describe '#user?' do
    subject { user.user? }

    context 'is user' do
      before { user.update(role: User.roles[:user]) }

      it { is_expected.to eq true }
    end

    context 'is not user' do
      it { is_expected.to eq false }
    end
  end

  describe '#short_info' do
    subject { user.short_info }

    it { is_expected.to eq "#{user.email} - #{user.name}" }
  end
end
```

- Viết functional testing

```
require 'rails_helper'

RSpec.describe HomeController, type: :controller do
  let!(:admin) { create(:user, admin: true, name: 'Hihi') }
  let!(:user) { create(:user, admin: false, name: 'Kaka') }

  describe '#index' do
    context 'access to index' do
      before do
        get :index
      end

      it { expect(assigns(:users)).to eq [admin, user] }
      it { expect(response).to render_template :index }
      it { expect(response.status).to eq 200 }
    end
  end
end
```

– Bước 5 – Push code mới lên Github

Sau khi thực hiện các bước trên, push code mới lên để Travis CI build project, chạy unit test và functional testing.

Add more commits by pushing to the **test** branch on **zgid123/demo-travis-ci**.



Some checks haven't completed yet

[Hide all checks](#)

1 pending check



 **continuous-integration/travis-ci/push** — The Travis CI build is in progress

[Details](#)



This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request



You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Travis CI | About Us | Blog | Status | Help

zgld123 / demo-travis-ci build passing

Current | Branches | Build History | Pull Requests

My Repositories

zgld123/demo-travis-ci # 2

Duration: 1 min 31 sec

Pull Request #1 test

Commit 82ceba3

#1: test

Branch master

Đường Tấn Huỳnh Phong authored and committed

#2 started

Running for 1 min 31 sec

Cancel build

Job log

View config

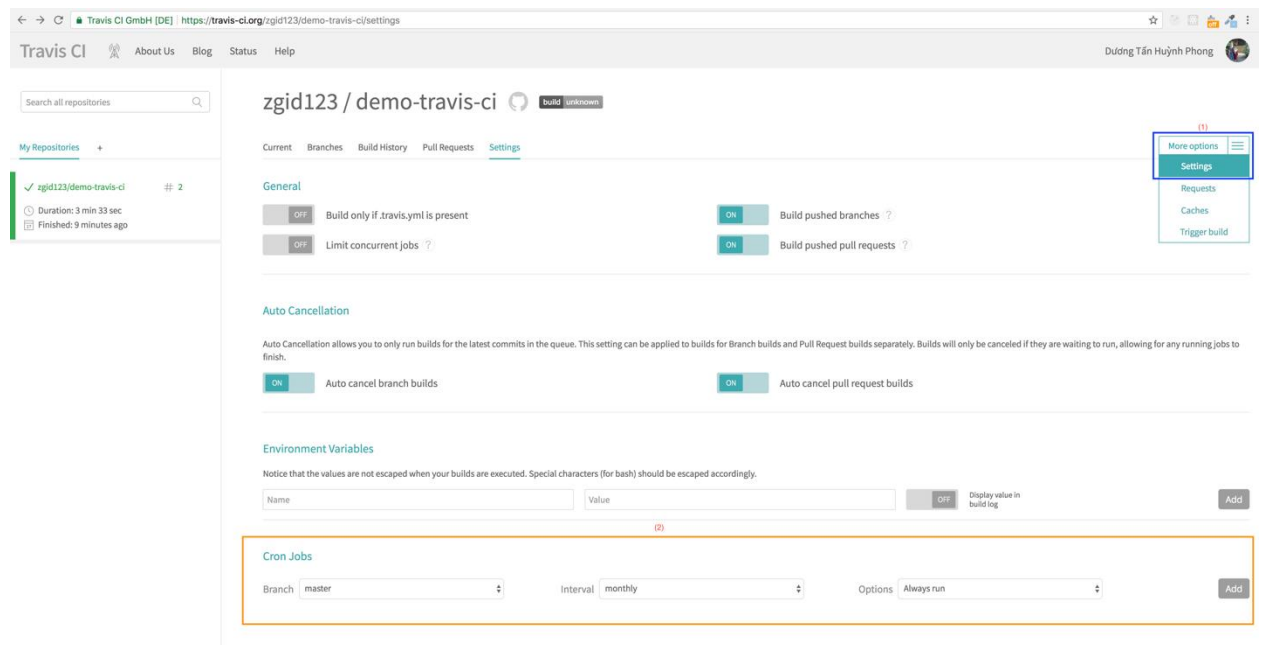
```

400 Worker information
401 mode of '/usr/local/clang-5.0.0/bin' changed from 0777 (rwxrwxrwx) to 0775 (rwxr-xr-x)
402 Build system information
403 removed '/etc/apt/sources.list.d/basho_r1ak.list'
404 Network availability confirmed.
405 127.0.0.1 localhost
406 ::1 ip6-localhost ip6-loopback
407 fe80::0 ip6-localnet
408 ff00::0 ip6-unicastprefix
409 ff02::1 ip6-allnodes
410 ff02::2 ip6-allrouters
411 172.17.0.5 travis-job-zgld123-demo-travis-ci-369417619.travis-ci.net travis-job-zgld123-demo-travis-ci-369417619
412 W: http://ppa.launchpad.net/couchdb/stable/ubuntu/dists/trusty/Release.gpg: Signature by key 158608AF08CC4F3C1E00FC7069548E1C17EAB57 uses weak digest algorithm (SHA1)
413 $ git clone --depth=50 https://github.com/zgld123/demo-travis-ci.git zgld123/demo-travis-ci
414 $ sudo service postgresql start
415
416 Setting environment variables from .travis.yml
417 $ export CC_TEST_REPORTER_ID=31e2972c0f64579da87228f1d9c10a9ee05d4021d040fc2c48c850e4b06f9ac7
418
419 Disabling Gradle daemon
420 $ mkdir -p ~/.gradle && echo "org.gradle.daemon=false" >> ~/.gradle/gradle.properties
421
422 $ rm use 2.4.2 --install --binary --fuzzy
423 $ ruby --version
424 ruby 2.4.2p198 (2017-09-14 revision 59899) [x86_64-linux]
425 $ rm --version
  
```

```

1255
1256 All examples were filtered out; ignoring {:focus=>true}
1257
1258 HomeController
1259 #index
1260 access to index
1261   should eq [#<User id: 1, email: "user1@example.com", encrypted_password: "", reset_password_token: nil, reset_p...20:22:57", confirmation_sent_at: nil, unconfirmed_email: nil, role: nil,
1262   avatar: nil, admin: false>]
1263   should render template index
1264   should eq 200
1265
1266 User
1267 #admin?
1268 is admin
1269   should eq true
1270   is not admin
1271   should eq false
1272 #vip?
1273 is vip
1274   should eq true
1275   is not vip
1276   should eq false
1277 #user?
1278 is user
1279   should eq true
1280   is not user
1281   should eq false
1282 #short_info
1283   should eq "user1@example.com - Dr. Lolita Schamberger"
1284
1285 Finished in 0.77042 seconds (files took 14.05 seconds to load)
1286 10 examples, 0 failures
  
```

Lưu ý: Travis CI sẽ chạy mỗi lần push code mới lên hoặc merge code vào branch master. Ngoài ra, Travis CI còn cho tạo cron job để build, chạy unit test và functional testing theo từng ngày, hoặc từng tháng hoặc mỗi tuần. Để sử dụng cron job, thực hiện như sau:



- Chọn (1) và chọn **Settings**.

• *Chỉnh thông số ở (2) như: branch – tên branch muốn chạy cron job, interval – lịch biểu chạy cronjob, gồm 3 thông số **monthly** (chạy hàng tháng), **weekly** (chạy hàng tuần) và **daily** (chạy hàng ngày) và options gồm **Always run** – chỉ cần tới interval thì sẽ chạy build, unit test và functional testing, hoặc **Do not run if there has been a build in the last 24h** – nếu trong vòng 24 tiếng đồng hồ có một build nào đó, thì cron job sẽ không chạy khi đến interval.*

2. Integrate Travis CI với Slack


– Bước 1 – Cài đặt **Travis CI** cho slack

Browse Apps

View App Directory

Travis CI


From the App Directory



Travis CI
Test and Deploy your code with confidence

Install

[Browse Apps](#)



Travis CI

App Info Settings

Travis CI is a continuous integration platform that takes care of running your software tests and deploying your apps.

This integration will allow your team to receive notifications in Slack for normal branch builds, and for pull requests, as well.

Install

Post to Channel

Start by choosing a channel where Travis CI build results will be posted.

#travis-ci

or [create a new channel](#)

Add Travis CI integration

Chọn channel để integrate và nhấn nút **Add Travis CI integration**.

– Bước 2 – Copy config vào file **.travis.yml**.

Setup Instructions

[close](#)

Here are the steps necessary to add the Travis CI integration.

Simple notifications

Paste the following code in your **travis.yml** file:

```
notifications:
  slack: lthd2017:St0lXRKIa5FFK1g7be4gKdFK
```

Multi-channel notifications

If you'd like to post notification to multiple Slack channels, you can instead specify them like so:

```
notifications:
  slack:
    rooms:
      - lthd2017:St0lXRKIa5FFK1g7be4gKdFK#general
      - lthd2017:St0lXRKIa5FFK1g7be4gKdFK#random
```



```

language: ruby
gemfile:
  - gemfiles/rails_4.2.0.gemfile
rvm:
  - 2.4.2
before_install:
  - gem update bundler
  - gem update --system
matrix:
  include:
    - rvm: 2.4.2
      gemfile: gemfiles/rails_5.0.0.gemfile
services:
  - postgresql
script:
  - bundle install
  - bundle exec rspec
before_script:
  - psql -c 'create database test_travis_ci_test;' -U postgres
  - cp config/database.yml.sample config/database.yml
  - curl -L https://codeclimate.com/downloads/test-reporter/test-reporter-latest-linux-amd64 > ./cc-test-reporter
  - chmod +x ./cc-test-reporter
  - ./cc-test-reporter before-build
after_script:
  - ./cc-test-reporter after-build
env:
  global:
    - CC_TEST_REPORTER_ID=c1977a02c828284b3adeeae3f1887ed56ae7fd9729d86e7de099ad2908571eea
notifications:
  email: false
  slack:
    on_success: always
    on_failure: always
    rooms:
      - lthd2017:zu5qE0CGIxEv3CI2CdHZRJzH#travis-ci

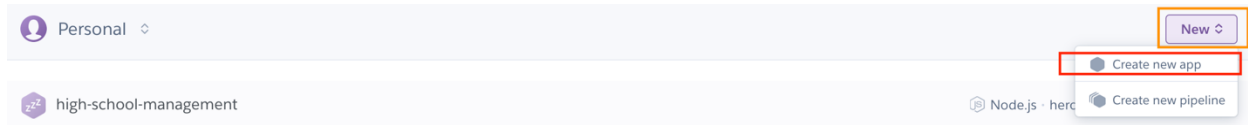
```

- Email: Nếu không muốn **Travis CI** gửi mail sau khi chạy build, unit test và functional testing xong, thì set là **false**.
 - Rooms: là token của config phía trên.
 - On_success: set **always** để khi mỗi lần success, sẽ push thông báo lên slack, hoặc **never** để không gửi thông báo, hoặc **change** để thông báo mỗi lần status thay đổi.
 - On_failure: tương tự như **on_success** và áp dụng khi fail.
- Bước 4 – Push code mới lên để kiểm tra kết quả.



3. Automatically deploy cho heroku với Travis CI

– Bước 1 – Tạo Heroku app



Chọn **New** rồi chọn **Create new app**.

App name

test-travis-ci is available

Choose a region

Add to pipeline...

Create app

Nhập tên app rồi nhấn nút **Create app**.

Chạy lệnh **heroku git:remote -a <tên app>** trong terminal để git remote đến app vừa tạo.

– Bước 2 – Thêm config vào file **.travis.yml** như sau:

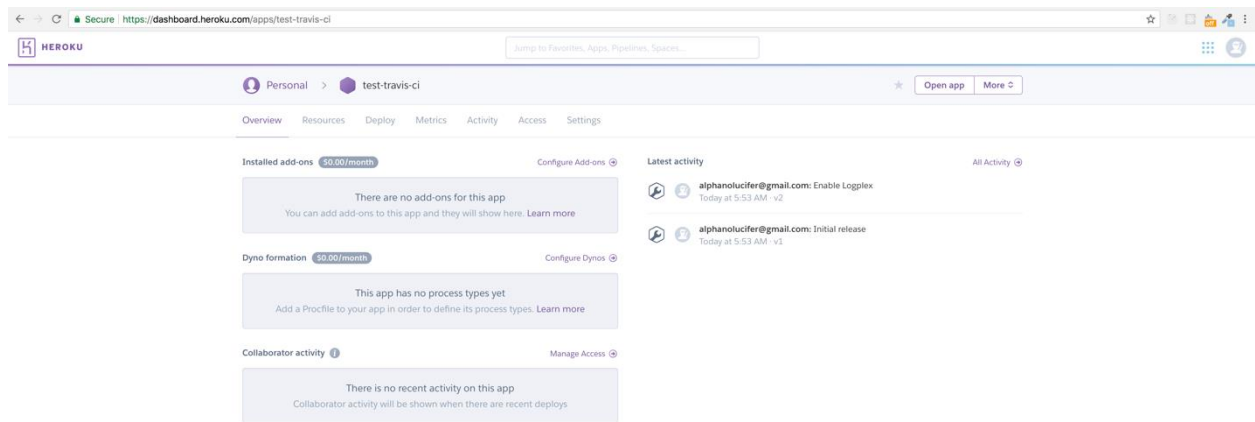
```
language: ruby
gemfile:
  - gemfiles/rails_4.2.0.gemfile
rvm:
  - 2.4.2
before_install:
  - gem update bundler
  - gem update --system
matrix:
  include:
    - rvm: 2.4.2
      gemfile: gemfiles/rails_5.0.0.gemfile
services:
  - postgresql
script:
  - bundle install
  - bundle exec rspec
before_script:
  - psql -c 'create database test_travis_ci_test;' -U postgres
  - cp config/database.yml.sample config/database.yml
  - curl -L https://codeclimate.com/downloads/test-reporter/test-reporter-latest-linux-amd64 > ./cc-test-reporter
  - chmod +x ./cc-test-reporter
  - ./cc-test-reporter before-build
after_script:
  - ./cc-test-reporter after-build
env:
  global:
    - CC_TEST_REPORTER_ID=c1977a02c828284b3adeae3f1887ed56ae7fd9729d86e7de099ad2908571eea
notifications:
  email: false
  slack:
    on_success: always
    on_failure: always
  rooms:
    - lthd2017:St0LXRKIa5FFK1g7be4gKdFK#travis-ci
deploy:
  provider: heroku
  app: test-travis-ci
  api_key: ba3d2552-ba7c-42cb-8882-1b7b39fe733d
```

- Provider: Tên host sẽ deploy lên.
- App: Tên app heroku.
- Api_key: lấy từ câu lệnh **heroku auth:token** từ terminal.

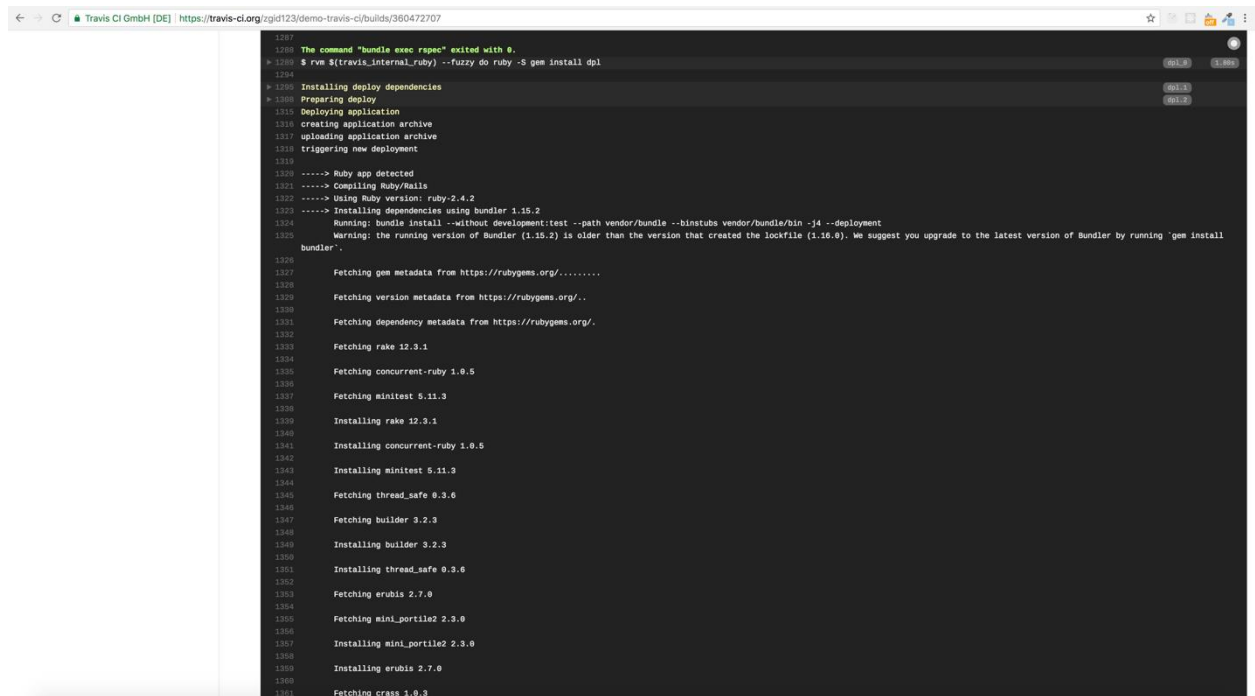
– Bước 3 – Chạy lệnh **heroku labs:enable runtime-dyno-metadata -a <tên app>** để mở dynos meta data của heroku lên (nếu không khi build có thể bị lỗi).

– Bước 4 – Push code mới để xem kết quả.

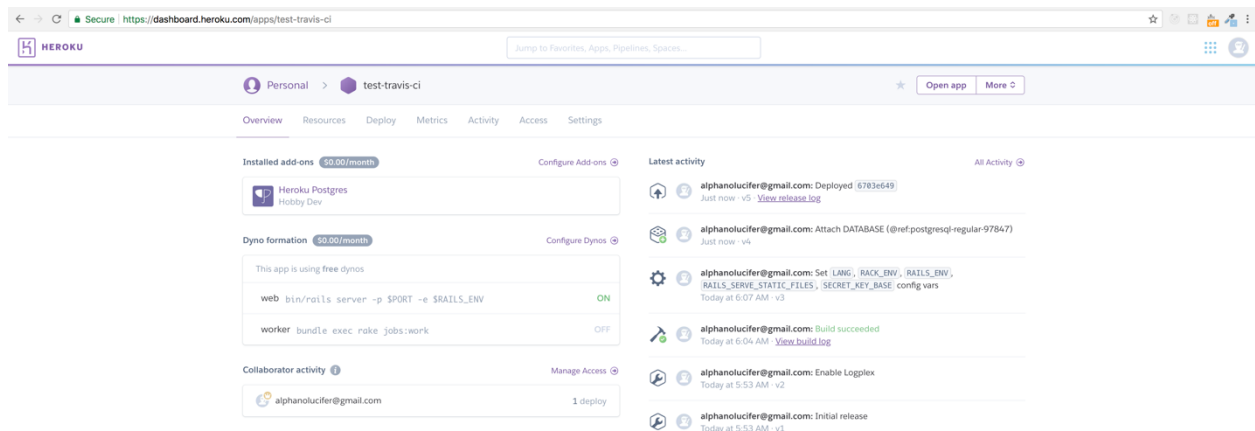
Trước khi deploy tự động:



Travis CI deploy tự động:



Sau khi deploy tự động:



Nguồn tham khảo:

<https://docs.travis-ci.com/user/getting-started/>