

CI/CD环境搭建

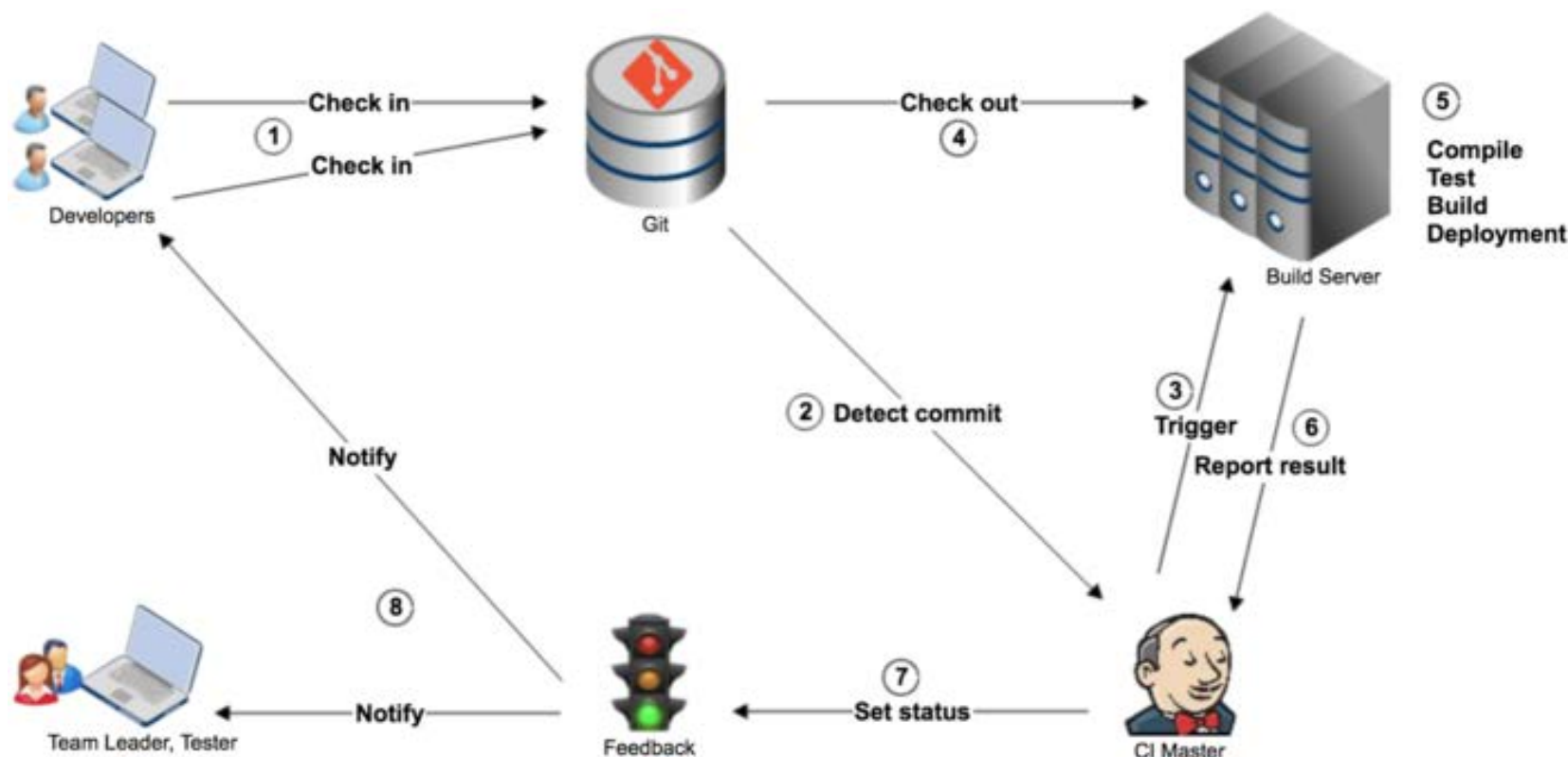
冯二虎 朱锦昊 马轲 谢添翼 曹金坤

CI是什么？

CI (Continuous Integration)

- 持续集成是一种软件开发实践，即团队开发成员经常集成他们的工作，通常每个成员每天至少集成一次，这就意味着每天可能会发生多次集成。

CI是什么?



每次集成都通过自动化的构建
(编译->自动化测试->部署)来验证正确性

Build Server 构建

- Check out
- Run build
- Compile
- Test (Unit, Integration, E2E)
- Deploy

Build Server 构建

构建运行完毕，Build Server会输出构建结果，CI Master会根据结果失败与否设置状态（失败：红，成功：绿），最后通知开发人员、QA以及Team Leader。

CD是什么？

CD (Continuous Delivery)


- 持续交付在持续集成的基础上，将集成后的代码部署到更贴近真实运行环境的类生产环境中。持续交付优先于整个产品生命周期的软件部署，建立在高水平自动化持续集成之上。

为什么要构建 CD?


- 快速发布
- 应对业务需求，更快地实现软件价值
- 迭代周期缩短，获得迅速反馈
- 高质量的软件发布标准。
- 整个交付过程进度可视化
- 更先进的团队协作方式

搭建 CI 环境

准备工作

- 拥有  Hub 帐号
- 该帐号下面有一个项目
- 该项目里面有可运行的代码
- 该项目还包含构建或测试脚本

搭建 CI 环境

使用 Github 账户登入  Travis CI , Travis 会列出 Github 上面你的所有仓库, 以及你所属的组织。此时, 选择你需要 Travis 帮你构建的仓库, 打开仓库旁边的开关。一旦激活了一个仓库, Travis 会监听这个仓库的所有变化。

搭建 CI 环境

 fengerhu1 / CI-CD  build error

[Current](#) [Branches](#) [Build History](#) [Pull Requests](#)

More options



✓ master package change

🔗 #4 passed

🔄 Restart build

🔗 Commit 1de1929 [↗](#)

🔗 Compare 5729ef4..1de1929 [↗](#)

🔗 Branch master [↗](#)

🕒 Ran for 1 min 57 sec

📅 29 minutes ago

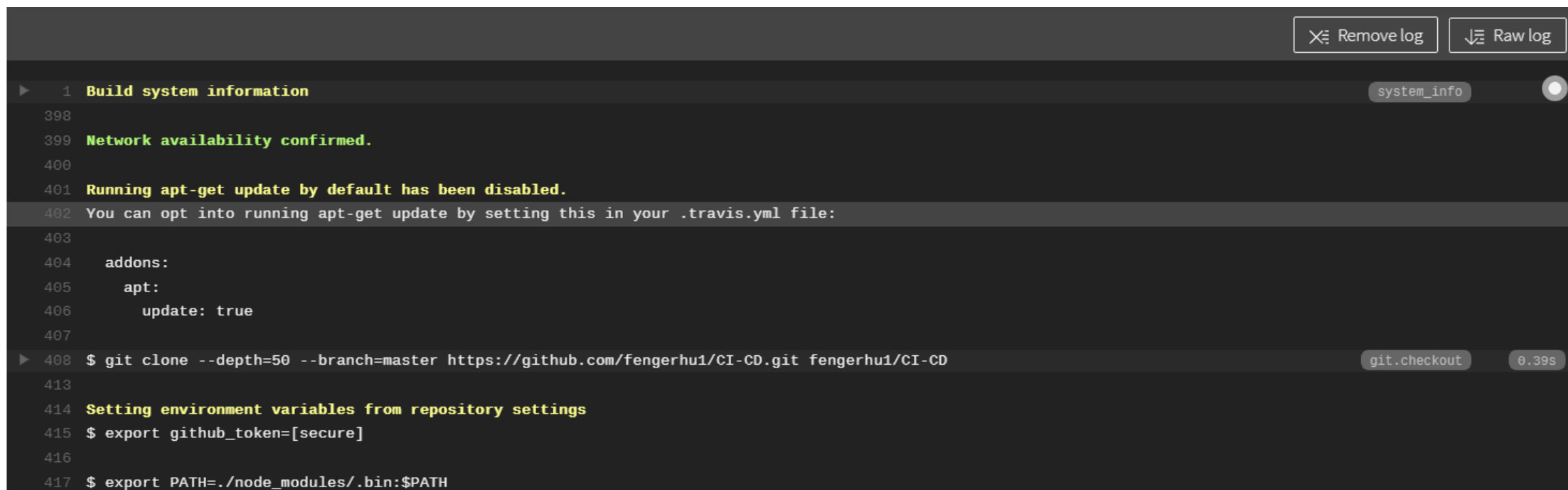
🔌 fengerhu1 authored and committed

搭建 CI 环境

配置.travis.yml

- Travis 要求项目的根目录下面，必须有一个.travis.yml配置文件。
- 一旦代码仓库有新的 Commit，Travis 就会去找这个文件，执行里面的命令。
- script中的脚本是你希望travis-ci帮你跑的测试脚本，可以根据需要自行更改。

搭建 CI 环境



The image shows a screenshot of a Travis CI build log. At the top right, there are two buttons: "Remove log" and "Raw log". The log content is as follows:

```
1 Build system information
398
399 Network availability confirmed.
400
401 Running apt-get update by default has been disabled.
402 You can opt into running apt-get update by setting this in your .travis.yml file:
403
404     addons:
405       apt:
406         update: true
407
408 $ git clone --depth=50 --branch=master https://github.com/fengerhu1/CI-CD.git fengerhu1/CI-CD
413
414 Setting environment variables from repository settings
415 $ export github_token=[secure]
416
417 $ export PATH=./node_modules/.bin:$PATH
```

On the right side of the log, there are two labels: "system_info" and "git.checkout". The "git.checkout" label is associated with a duration of "0.39s".

搭建 CI 环境

在项目根目录中新建 .travis.yml 配置文件:

```
language: node_js
node_js:
  - "stable"
cache:
  directories:
    - node_modules
script:
  - npm test
  - npm run build
deploy:
  provider: pages
  skip_cleanup: true
  github_token: $github_token
  local_dir: build
on:
  branch: master
```

搭建 CI 环境

运行

- `$ npm run build`
- commit之后，在travis-ci中可以看到ci结果。若测试不通过，在job log中也可以看到完整的错误信息。

搭建 CI 环境

运行结果:

```
▶ 2672 store build cache
▶ 2696 $ rvm $(travis_internal_ruby) --fuzzy do ruby -S gem install dpl
2700
▶ 2701 Installing deploy dependencies
2714 Logged in as @fengerhu1 ()
2715 cd /tmp/d20180521-5057-1pe5kps/work
▶ 2716 Preparing deploy
▶ 2718 Deploying application
2742 Done. Your build exited with 0.
```

搭建 CI 环境

简单部署 -- Github Page

- 前端package.json中需要修改, 添加
Github page url
- 添加script

搭建 CI 环境

简单部署 -- 添加github_token认证

- Generate new token - > 勾选Repo

- environmental variables - > 添加token

Environment Variables

Notice that the values are not escaped when your builds are executed. Special characters (for bash) should be escaped accordingly.

github_token

🔒

Name

Value



Display value in
build log

搭建 CD 环境

准备工作

- 科学上网
- 官网注册  **HEROKU** 账号
- 下载 heroku-cli

搭建 CD 环境

命令行中进入项目目录

- `$ heroku login`
- Enter your Heroku credentials.
- Email: user@example.com
- Password:

搭建 CD 环境

创建heroku repo

- \$ heroku create

上传至heroku repo

- \$ git push heroku master

打开部署的网站

- \$ heroku open



Thanks