**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN – ĐHQG TPHCM**

# **KHOA CÔNG NGHỆ THÔNG TIN**

**Homework 2**

**Môn học: Các Chủ Đề Nâng Cao Trong**

**Công Nghệ Phần Mềm**

**Lớp: 18\_3**

Sinh viên thực hiện:

* Lê Nguyên Hào –18120355
* Nguyễn Văn Lâm – 18120433
* Phạm Thị Anh Thư – 18120582

TP Hồ Chí Minh, tháng 06 năm 2022

*Mục lục*

[1. Yêu cầu 1: CI/CD là gì? Hãy nêu vai trò và sự cần thiết của CI/CD trong phát triển các hệ thống phần mềm 3](#_Toc105079637)

[2. Yêu cầu 2: Để thực hiện CI/CD hoàn chỉnh, một sự án phần mềm cần nhiều công cụ khác nhau trong một hệ thống CI/CD (hay còn gọi là chuỗi các công cụ - toolchain – cho CI/CD). Hãy tìm và mô tả ngắn gọn các công cụ cho một hệ thống CI/CD hoàn chỉnh. 3](#_Toc105079638)

[3. Yêu cầu 3: Cài đặt và viết hướng dẫn ngắn gọn cho một hệ thống CI/CD như thế (các bước cài đặt và tích hợp các công cụ) 5](#_Toc105079639)

[3.1. Mô tả các công cụ sử dụng: 5](#_Toc105079640)

[3.2. Mô tả flow hệ thống CI/CD: 5](#_Toc105079641)

[3.3. Hướng dẫn cài đặt: 6](#_Toc105079642)

[3.3.1. Unit test (mocha): 6](#_Toc105079643)

[3.3.3. Katalon testOps 7](#_Toc105079644)

[3.3.4. Host deploy (render.com) 8](#_Toc105079645)

[3.3.5. CircleCI 9](#_Toc105079646)

1. **Yêu cầu 1: CI/CD là gì? Hãy nêu vai trò và sự cần thiết của CI/CD trong phát triển các hệ thống phần mềm**

CI (Continuous Integration): tích hợp liên tục. Quá trình hoạt động cho phép các thành viên trong một team liên tục lưu trữ những mã mới vào một kho nhất định. Nhờ vào số dữ liệu này, CI sẽ tự động build và test.

CD (Continuous Delivery): chuyển giao liên tục. CD là trường hợp mở rộng của CI. CD tự sửa code đã được build và test nếu phát hiện lỗi, ngoài ra nó cũng tự động thay đổi môi trường testing hoặc staging để nâng cao chất lượng kiểm tra.

Vai trò và sự cần thiết của CI/CD:

* Giảm thiểu rủi ro khi triển khai code. Nhanh chóng phát hiện ra những lỗi mắc phải, sớm khắc phục và rút kinh nghiệm cho những lần sau.
* Tăng tính chia sẻ giữa các thành viên, các thành viên cũng sẽ có những kế hoạch triển khai phù hợp để hoàn thành phần việc của mình.
* Giảm thiểu các thao tác build thủ công.

1. **Yêu cầu 2: Để thực hiện CI/CD hoàn chỉnh, một sự án phần mềm cần nhiều công cụ khác nhau trong một hệ thống CI/CD (hay còn gọi là chuỗi các công cụ - toolchain – cho CI/CD). Hãy tìm và mô tả ngắn gọn các công cụ cho một hệ thống CI/CD hoàn chỉnh.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Công cụ | Nguồn gốc | Đặc điểm | Phù hợp với | Đánh giá |
| Jenkins | Mã nguồn mở được viết bằng Java. Có nguồn gốc như là một nhánh của Hudson. | * Là công cụ CI platform và nó cung cấp cấu hình thông qua cả giao diện GUI và các câu lệnh điều khiển. | * Code lưu ở 1 server riêng. * Muốn toàn quyền kiểm soát CI/CD. * Yêu cầu máy chủ tại chỗ. * Có thể chỉ định 1 người hay một nhóm quản lý và duy trì. * Tiết kiệm tiền. | Mạnh mẽ, linh hoạt. |
| Team City | Là máy chủ CI thật đến từ công ty JetBrains. | * Cung cấp toàn bộ các tính năng trong bản miễn phí. * Phải trả phí để sử dụng các chức năng nâng cao. * Hoạt động trên nhiều nền tảng khác nhau. * Bảo mật và cung cấp plugin cực kì ổn định. | * Dùng server riêng . * Chỉ định 1 hay 1 nhóm người quản lý. * Có thể hỗ trợ nhanh chóng. * Có nhiều dự án với cấu hình tương tự, mỗi dự án tiến triển khác nhau. | Phải trả phí, phù hợp với nhu cầu doanh nghiệp và quy mô dự án nhỏ. |
| GitLab CI/CD | Là một phần của dự án mã nguồn mở GitLab. | * Công cụ có sẵn và tích hợp trên gitlab. * Cho phép mở rộng dễ dàng (vì hệ thống nhiều máy chủ). * Cho phép lưu trữ một số tính năng Gitlab trên máy chủ. | * Code lưu trên Gitlab. * Bỏ qua config. * Không cần Plugin. * Khi cần đăng kí tích hợp docker. * Liên tục phát hành những tính năng ổn định. | Cung cấp miễn phí. Là lựa chọn tốt cho các doanh nghiệp. |
| CircleCI | Là một công cụ dựa trên đám mây. | * Khả năng lưu trữ mạnh và quy trình làm việc tuỳ biến cao. * Dễ dang thiết lập và chạy. * Có thể cấu hình và gửi trực tiếp kết quả đến Slack. * CircleCI chỉ hỗ trợ GitHub và BitBucket và các ngôn ngữ {Java, Rupy/Rails, Python, Node.js, PHP, Haskell và Skala} | * Cần hỗ trợ sẵn và song song với việc dev. * Code lưu trữ trên Github hay BitBucket. * Nhóm gồm nhiều nhà phát triển dùng chung một CI. * Ưu tiên tóc độ hơn các thứ khác. * Cần chung 1 quy trình công việc tuỳ biến cao, tương thích nhiều test tool, không giới hạn repo, user… | Nếu cần tốc độ nhanh thì Circle CI là một lựa chọn tốt. |
| Travis CI | Ban đầu là một dự án mã nguồn mở sau chuyển thành dự án mã nguồn đóng. | * Miễn phí cho tất cả các dự án trên Github và cho 100 bản build đầu tiên. * Cho ơheps kiểm tra hệ điều hành Mac và Linux đồng thời * Thiết lập dễ dàng và nhanh chóng | * Code trên github. * Cần hỗ trợ nhiều ngôn ngữ khác nhau. * Không dùng window. * Cần 1 giải pháp linh hoạt. * Cần các máy chủ cơ sở dữ liệu được cài sẵn. * Không yêu cầu tích hợp từ bển thứ 3. | Là giải pháp tốt nhất cho cả máy chủ và On-premises. |

1. **Yêu cầu 3: Cài đặt và viết hướng dẫn ngắn gọn cho một hệ thống CI/CD như thế (các bước cài đặt và tích hợp các công cụ)**
   1. **Mô tả các công cụ sử dụng:**

* Version control: GitHub
* Code: JS – NodeJS
* Test:
  + Unit test: mocha (Unit test nhanh module JS trước khi chính thức merge vào main (pull request))
  + Integration test: Katalon testOps
* CI: CircleCI liên kết các thành phần lại
* Deployment:
  + Deploy staging để chạy integration test: Heroku
  + Deploy thật: Render.com (Đại diện cho một host khác host dùng để staging)
* Issue tracking:
  + CircleCI liên kết với Slack trả về kết quả deploy
  + Katalon testOps liên kết với Slack trả về kết quả chạy integration test
  1. **Mô tả flow hệ thống CI/CD:**

Diagram

Description automatically generated

* 1. **Hướng dẫn cài đặt:**
     1. **Unit test (mocha)**:
* Cài đặt mocha: *npm install mocha*
* Ở root thư mục project, tạo thư mục *test*
* Viết test trong thư mục *test* (vd: *{root project}/test/test.js* (<https://mochajs.org/#getting-started>))

Text

Description automatically generated

* Thêm script trong *package.json*

*A picture containing shape

Description automatically generated*

* Chạy unit test: *npm test*
  + 1. **Host staging (Heroku)**: theo <https://circleci.com/docs/2.0/deployment-integrations/#heroku>
* Truy cập <https://dashboard.heroku.com/apps> (đăng nhập nếu cần thiết)
* Tạo API Key cho CircleCI

Graphical user interface, application, Teams

Description automatically generated

* Tạo một app mới hay sử dụng app có sẵn
  + Cung cấp biến môi trường (vd: DB Staging,…): Setting -> Config Vars

A picture containing background pattern

Description automatically generated

* Lưu lại link dẫn đến app (vd: <https://staging-cicd-bored-api.herokuapp.com/>)
  + 1. **Katalon testOps**
* Setup Katalon project
* Tạo project Katalon Studio trong thư mục thuộc root project (vd: *{root\_project}/test\_katalon*)
* Thêm một profile (vd: “default”) trong Katalon Studio, tạo một biến lưu link host staging (vd: HOST\_URL) và để test case truy cập host này khi chạy

Graphical user interface, text, application

Description automatically generated

Graphical user interface

Description automatically generated

* + Viết các test case trong katalon và add vào một test suite (vd: *TS\_Staging*)
  + Khi đó, ta có thể chạy test suite từ root project bằng Katalon Runtime Engine bằng câu lệnh: *cd test\_katalon && katalon-execute.sh -statusDelay=30 -retry=0 -testSuitePath="Test Suites/TS\_Staging" -executionProfile="default" -browserType="Chrome (headless)" --apiKey=$KATALON\_API\_KEY*
* Setup Katalon testOps
* Truy cập <https://testops.katalon.io/> và đăng nhập
* Tạo một organization mới -> Tạo API Key (dùng để gọi Katalon Runtime Engine)
  + 1. **Host deploy (render.com)**
* Truy cập <https://dashboard.render.com/> tạo một app mới (Web Service)

Graphical user interface, application, Word

Description automatically generated

* Đặt tên và kết nối đến repository trên gitHub

Timeline

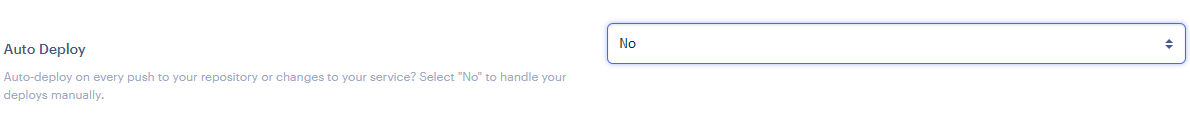
Description automatically generated

* Advanced 🡪 Setup biến môi trường (vd: DB,…)

Graphical user interface

Description automatically generated with low confidence

* Advanced 🡪 Auto deploy 🡪 “No”



* Setting 🡪 Deploy Hook 🡪 Copy (Link này sẽ được dùng để trigger deploy)

Graphical user interface, application

Description automatically generated

* + 1. **CircleCI**

1. Viết *.circleci/config.yml*

* Khai báo jobs
  + Chạy unit test

Text

Description automatically generated

* Deploy staging

Text

Description automatically generated

* Chạy integration test

Text

Description automatically generated

* Deploy thật & gọi thông báo cho Slack

Text

Description automatically generated

* Định nghĩa workflow

Text

Description automatically generated

1. Kết nối circleCI với repository gitHub của project
2. Thiết lập các biến môi trường (*Project Setting -> Environment variables*)

Graphical user interface, text, application

Description automatically generated

1. Thiết lập circleCI kết nối slack như: <https://circleci.com/docs/2.0/slack-orb-tutorial/#step-two-connect-circleci-with-slack>
2. Thiết lập email notification: <https://circleci.com/docs/2.0/notifications/#set-or-change-email-notifications>

Graphical user interface, text, application, email

Description automatically generated

(File .circleci/config.yml đầy đủ)

Text

Description automatically generated