BÁO CÁO THỰC HÀNH

Bài thực hành số 01: Automating Everything as Code (DevOps/CI-CD/Git)

**Môn học:** Lập trình an toàn và khai thác lỗ hỏng phần mềm

**Lớp:** NT521.N11.ATCL

**THÀNH VIÊN THỰC HIỆN (Nhóm Ra dẻ):**

|  |  |  |
| --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** |
| 1 | Vũ Hoàng Thạch Thiết | 20521957 |
| 2 | Lê Viết Tài Mẫn | 20521593 |

|  |
| --- |
| **Điểm tự đánh giá** |
|  |

MỤC LỤC

[**A.** BÁO CÁO CHI TIẾT 2](#_Toc116322169)

[1. Quản lý phiên bản phần mềm với git 2](#_Toc116322170)

[a. Thiết lập git repository 2](#_Toc116322171)

[b. Staging và committing một tập tin trên Repository 3](#_Toc116322172)

[c. Sửa đổi tập tin và theo dõi các thay đổi 4](#_Toc116322173)

[d. Branches và Merging 6](#_Toc116322174)

[e. Xử lý xung đột khi merge 8](#_Toc116322175)

[f. Thiết lập hệ thống quản lý phần mềm Git kết hợp với Github 10](#_Toc116322176)

[2. Xây CI/CD Pipeline bằng jenkins 11](#_Toc116322177)

[a. Commit sample app lên github 11](#_Toc116322178)

[b. Sửa đổi sample app và push thay đổi lên git 12](#_Toc116322179)

[c. Tải và thiết lập chạy jenkins docker image 14](#_Toc116322180)

[d. Cấu hình Jenkins 15](#_Toc116322181)

[e. Sử dụng Jenkins để chạy úng dụng đã dựng 17](#_Toc116322182)

[f. Dùng Jenkins để kiểm tra bản dựng 18](#_Toc116322183)

[g. Tạo Pipiline trong Jenkins 19](#_Toc116322184)

[3. Test python function với unittest 22](#_Toc116322185)

[**B.** TÀI LIỆU THAM KHẢO 25](#_Toc116322186)

# BÁO CÁO CHI TIẾT

## Quản lý phiên bản phần mềm với git

### Thiết lập git repository

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Dòng 1, 2: Cấu hình thông tin nhóm là “Ra de” sử dụng và liên kết tài khoản trong local repository và email đại diện của nhóm
* Dòng 3: Kiểm tra thông tin người sử dụng
* Dòng 4: Tạo thư mục tên nhóm “ra de” và di chuyển đến thư mục vừa tạo
* Dòng 5: Trong thư mục vừa tạo, tiếp tục tạo thư mục git-intro và di chuyển vào
* Dòng 6: Khởi tạo thư mục dưới dạng Git-repository
* Dòng 8: Xem trạng thái của repository cục bộ

### Staging và committing một tập tin trên Repository

Ảnh có chứa văn bản

Mô tả được tạo tự động

- Dòng 1: kiểm tra tập tin vừa tạo trong thư mục

- Dòng 2: Kiểm tra nội dung trong file vừa tạo

- Dòng 3: Kiểm tra trạng thái repository sau khi tạo tập tin

\*Nhưng tìm thấy tập tin nhưng chưa ở chế độ giám sát

\*Nếu muốn ở chế độ giám sát thì phải dùng lệnh git add tên\_file

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Dòng 4: đưa tập tin vào vùng staging, tạo một snapshot cho tệp nên mọi thay đổi của tệp sẽ được giám sát để chuẩn bị commit
* Dòng 5: Kiểm tra trạng thái repository sau khi “stage” tập tin
* Dòng 6: Commit tập tin, Đối với các thay đổi đã được đưa vào vùng ‘staging’, cần thực hiện tiếp bước Commit để Git thay đổi những thay đổi đó
* Dòng 7: Xem lịch sử commit, lệnh này sẽ hiển thị tất cả các commit trong lịch sử của nhánh hiện tại. Và theo mặc định, tất cả các commit là được thực hiện cho nhánh master

### Sửa đổi tập tin và theo dõi các thay đổi

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Dòng 1: Sửa đổi tập tin bằng cách thêm 1 dòng ”I am beginning to understand Git”
* Dòng 3: Xem lại nội dung đã thay đổi
* Dòng 4: Kiểm tra thay đổi với repository

\*Ta thấy rằng file cập nhật được hiển thị dưới dạng file chưa được theo dõi. Nó hiển thị bằng màu đỏ vì nó chưa được tracked

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Dòng 5: Sử dụng các lệnh để add vào trạng thái tracked để commit
* Dòng 8: Xem lại commit vừa thực hiện trong repository

+ Commit ID: 0dbd9f…426

+ Thời gian commit: 25/9/2022 21:47:21

+ Thông điệp commit: NhomRade Added additional line to file

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Dòng 9: So sánh các commit
* hiện thị thông tin thay đổi giữa thư mục làm việc và vùng index (staging) hoặc với commit cũ, thông tin thay đổi giữa index(staging) và commit, thông tin thay đổi giữa hai nhánh
* diff –git a/README … : hiển thị các nguồn đầu vào của diff là file README>MD
* ---a/README.MD và +++b/README.MD : gán các ký hiệu cho từng nguồn đầu vào
* @@ -2… : Tiêu đề chunk, là bản tóm tắt các thay đổi được thực hiện cho tệp
* Trong ví dụ này thì đã thêm dòng “I am beginning to understand Git”

### Branches và Merging

1. Làm việc trong branch

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Dòng 1: Tạo branch mới
* Dòng 2: Kiểm tra branch hiện tại
* Dòng 3: Chuyển sang branch mới
* Dòng 4: Kiểm tra lại banch hiện tại
* Lệnh git branch không kèm theo tên branch sẽ hiển thị tất cả các branch cho repository này. Trong đó, nhánh có ký hiệu dấu \* phía trước là nhánh hiện tại đang làm việc (đang checkout) và nhánh đang có ký hiệu \* là feature
* Dòng 5: Thêm đoạn text mới “…from branch feature” vào file README.MD
* Dòng 6: Stage tập tin vừa thay đổi trong branch feature
* Dòng 7: Kiểm tra trạng thái repository sau khi stage tập tin
* Dòng 8: Commit tập tin trong branch feature với thông điệp “Added a third line in feature branch”
* Dòng 8: Xem lịch sử commit đã thực hiện
* Ảnh có chứa văn bản

  Mô tả được tạo tự động
* Ta thấy ở phần commit ID so sánh 2 phần commit ở 2 thời gian thì ở dưới có tên branch là master còn commit mới nhất thì có tên branch là feature sau khi chuyển từ nhánh master

1. Merge các thay đổi từ các branch vào brnach master

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Dòng 1 : chuyển từ banch feature sang branch master
* Dòng 2: Kiểm tra file README.MD ở branch master có bị thay đổi nội dung chưa
* Dòng 3: Gộp branch feature với branch master lại với nhau để thêm những tập tin từ branch feature vào branch master
* Dòng 4: Kiểm tra xem nội dung file README.MD có sự thay đổi chưa
* Ta thấy file README.MD đã có sự thay dổi từ lần cuối ở brnach master trước khi gộp. Nó đã có thêm 1 dòng “…from branch feature”
* Dòng 5: Kiểm tra branch feature còn tồn tại
* Dòng 6: Xóa branch feature đi
* Dòng 7: Kiểm tra các branch đã thấy branch feature đã bị xóa

### Xử lý xung đột khi merge

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Dòng 1, 2: Tạo branch test và chuyển sang branch test vừa tạo
* Dòng 3: Chỉnh sửa nội dung của file README.MD
* File trước khi thực thi câu lệnh

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Dòng lệnh này dùng để lưu các thay đổi để thay thế chuỗi trong cùng 1 tệp
* Ta thấy sau khi thực thi câu lệnh thì tất cả từ feature được chuyển thành test
* Dòng 5: Thực hiện stage và commit branch test
* Dòng 6,7: Chuyển sang branch master và thực hiện thay đổi file README.MD tương tự như ở trên
* Dòng 8: Xem nội dung của file README.MD
* Branch master vẫn cưa có thay đổi gì nhưng sau khi thực hiện câu lệnh thì các từ feature được chuyển thành master giống như đã làm ở branch test
* Dòng 9: stage và commit ở branch master
* Dòng 10: Merge 2 branch test và master lại với nhau7 nhưng không được do xảy ra xung đột
* Dòng 11: Tìm xung đột từ log
* HEAD đang là branch master
* File READMME.MD sẽ chưa thông tin tìm ra xung đột. Phiên HEAD là branch master có chứa từ “master” đang xung đột với phiên bản từ branch test là từ “test”
* Ảnh có chứa văn bản

  Mô tả được tạo tự động
* Sau đó mở tập tin lên tìm và xóa các đoạn có ký hiệu “<<”, “==” và dòng được thêm ở branch test sau đó lưu lại
* Dòng 12, 13,14: Stage và commit sau đó kiểm tra commit ở branch master
* Đã merge được branch master với branch test

### Thiết lập hệ thống quản lý phần mềm Git kết hợp với Github

* Đã trỏ git repository tới githubrepository thành công
* Và gửi tập tin READMME.ME lên github thành công

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Kết quả trên github

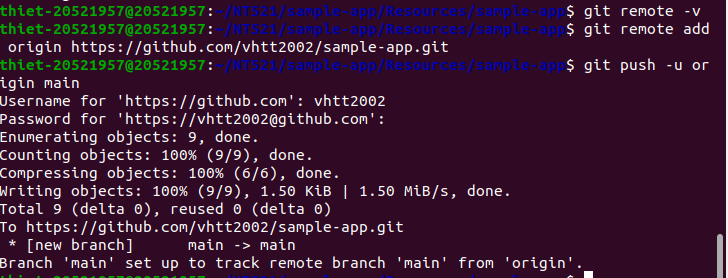
Ảnh có chứa văn bản, ảnh chụp màn hình, đen, màn hình

Mô tả được tạo tự động

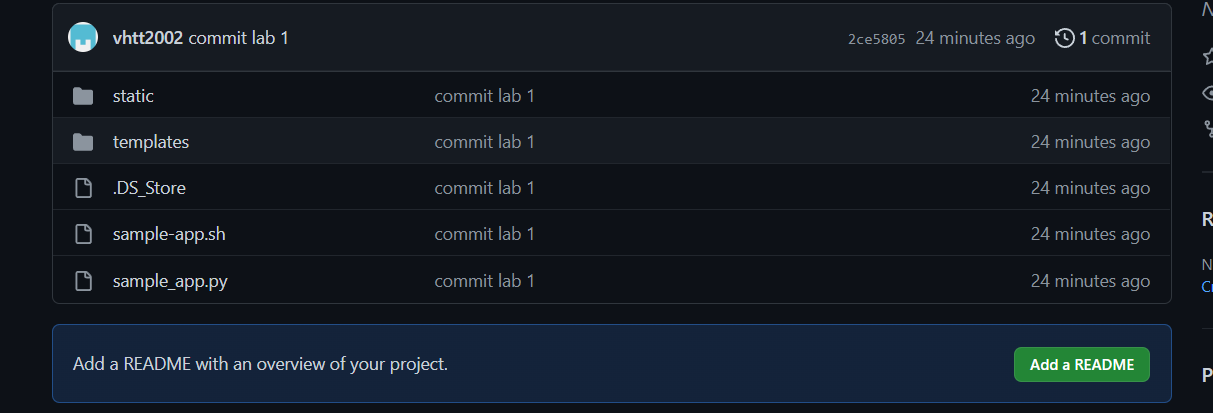
## Xây CI/CD Pipeline bằng jenkins

### Commit sample app lên github

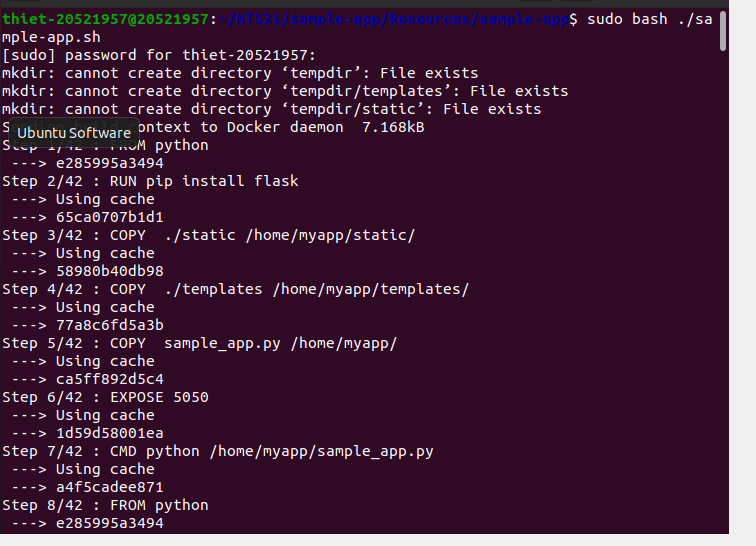




* Kết quả sau khi push lên github



### Sửa đổi sample app và push thay đổi lên git



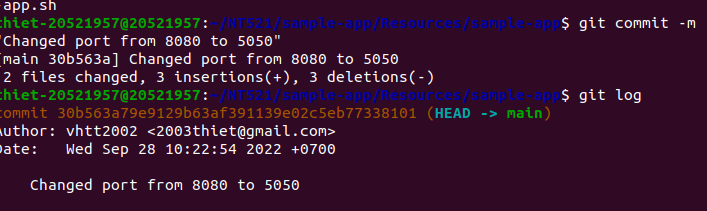
Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Push sự thay đổi lên github
* Commit với thông điệp “Changed port from 8080 to 5050”

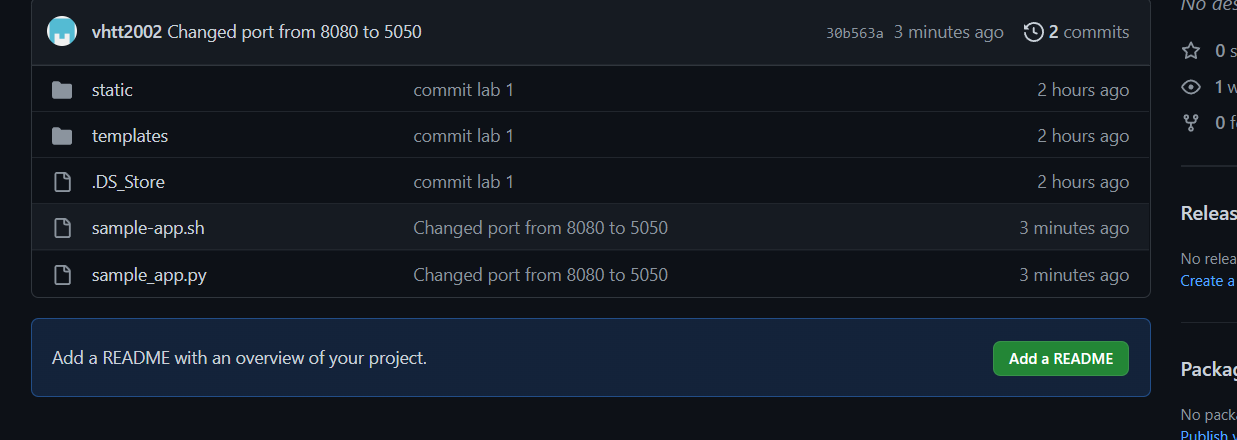


* Push thay đổi lên github

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Kết quả sau khi push thay đổi trên github



### Tải và thiết lập chạy jenkins docker image

Ảnh có chứa văn bản

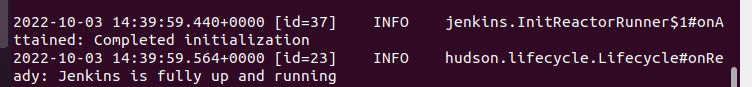
Mô tả được tạo tự động

* Câu lệnh dùng để khởi chạy Jenkins
* Ý nghĩa option -v trong câu lệnh trên:
* Thao tác này sẽ lưu trữ dữ liệu Jenkins trong jenkinsdata:/var/jenkins\_home. Đảm bảo rằng jenkinsdata:/var/jenkins\_home có thể truy cập bởi người dùng jenkins trong container

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Password của Jenkins



* Máy chủ Jenkins đã chạy xong thì sẽ có dòng thông báo như trên

### Cấu hình Jenkins

Ảnh có chứa văn bản

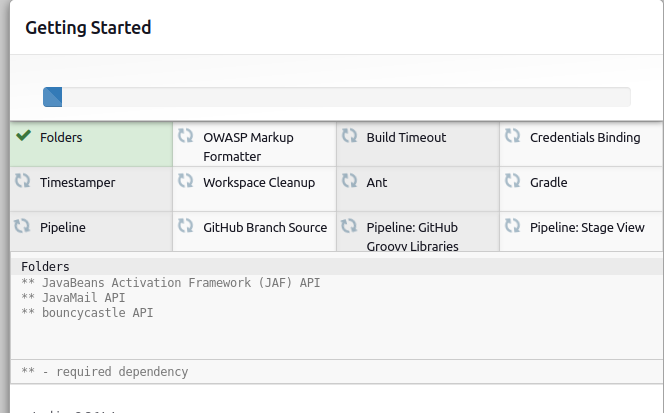
Mô tả được tạo tự động

* Mở trình duyệt truy cập <http://localhost:8080/> và nhập mật khẩu đã lưu như ở trên

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Cài đặt các Jenkins plugins



Ảnh có chứa văn bản

Mô tả được tạo tự động

* Giờ ta có thể bắt đầu sử dụng Jenkins

### Sử dụng Jenkins để chạy úng dụng đã dựng

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Create a job sau đó đạt tên là BuildAppJob và chọn thể loại Freestyle project sau đó chọn OK

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Dùng Jenkins để build ứng dụng bản build thành công với số thứ tự 6

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Xem console output để xem chi tiết bản dựng

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Kết quả build thành công

### Dùng Jenkins để kiểm tra bản dựng

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Tạo Job mới tên TesttAppJob\_NhomRade
* Source code chọn none
* Ở Build Triggers tab, chọn checkbox Build after other projects are built và điền BuildAppJob và Trigger only if build is stable

Ảnh có chứa văn bản

Mô tả được tạo tự động

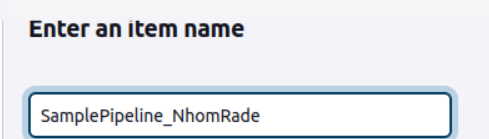
* Bash shell để thực hiện yêu cầu

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Đã build thành công TestAppJob\_NhomRade

### Tạo Pipiline trong Jenkins



* Tạo Job mới tên SamplePipeline\_NhomRade

Ảnh có chứa văn bản

Mô tả được tạo tự động- Chọn kiểu Pipeline

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Cấu hình job SamplePiline\_NhomRade
* Ở tab piline trong phần script ta điền đoạn mã như trên và chọn save

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Kết quả của console output

## Test python function với unittest

Text

Description automatically generated

Text

Description automatically generated

Có thể thấy rằng trong file json gồm có

Dictionary

Trong dictionary có thể có list

Trong list lại có thể dictionary

Và hàm json\_search có chức năng là kiểm tra từng object(input\_object).

Nếu object là dict thì kiểm tra từng cặp k-v.

nếu k trong dict trùng với key thì gán cặp k-v vào ret\_val

Nhưng nếu object v tiếp tục là dict thì đệ quy

Nhưng nếu object v là list thì duyệt từng phần tử trên mảng list

nếu phần tử không phải là string hoặc int thì đệ quy

Nếu input\_objet không phải là string hoặc int (else)

đệ quy

khi ret\_val để trong hàm thì mọi lần gọi đệ quy thì thì ret\_val lại bị gán bằng rỗng

(ret\_val = [])

lỗi ở file này là hàm đệ quy nhưng không có khử đệ quy( điểm dừng).

Đầu tiên, ta thử chép code file json và json\_search ra một file py khác để chạy debug.

Ảnh có chứa văn bản

Mô tả được tạo tự động

Thử đặt return ret\_val sau khi tìm thấy cặp k-v tương ứng

Text

Description automatically generated

Thấy đã tìm được cặp v-k nhưng đệ quy vẫn tiếp tục bị lặp. khi vào hi3, đã tìm thấy cặp k-v và hi1 đã trả giá trị nhưng ở hi3 lại không được gán giá trị cho ret-val và ret\_val lại không được trả về cho các hàm trên của đệ quy

Vì thế để sửa lỗi: Nếu tìm thấy cặp k-v tức là ret-val không rỗng thì trả về cho cái hàm phía trên của đệ quyGraphical user interface, text

Description automatically generated

# TÀI LIỆU THAM KHẢO