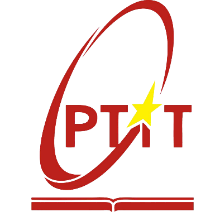
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA AN TOÀN THÔNG TIN**

-------🙞🙜🕮🙞🙜-------



**CÁC KỸ THUẬT GIẤU TIN**

**Đề bài: Tìm hiểu về chuẩn mã hóa video H264-AVC**

|  |  |
| --- | --- |
| **HỌ TÊN:** | **PHẠM HẢI DƯƠNG** |
| **MÃ SINH VIÊN:** | **B21DCAT072** |

Giảng viên hướng dẫn: PGS.TS. Đỗ Xuân Chợ

**HÀ NỘI, 03/2025**

**MỤC LỤC**

[1. Giới thiệu chung về bài thực hành 5](#_Toc196590796)

[2. Nội dung và hướng dẫn bài thực hành 5](#_Toc196590797)

[1.1. Mục đích 5](#_Toc196590798)

[1.2. Yêu cầu đối với sinh viên 6](#_Toc196590799)

[1.3. Nội dung thực hành 6](#_Toc196590800)

[3. Phân tích yêu cầu của bài thực hành 8](#_Toc196590801)

[4. Cài đặt và cấu hình máy ảo 9](#_Toc196590802)

[5. Tích hợp và triển khai 14](#_Toc196590803)

[5.1. Docker Hub 14](#_Toc196590804)

[5.2. GitHub 14](#_Toc196590805)

[6. Thử nghiệm và đánh giá 15](#_Toc196590806)

**DANH MỤC HÌNH ẢNH**

[Hình 1: Giao diện Labedit 9](#_Toc196590811)

[Hình 2: Cấu hình result 10](#_Toc196590812)

[Hình 3: Cấu hình Result 10](#_Toc196590813)

[Hình 4: Cấu hình Goal 11](#_Toc196590814)

[Hình 5: Docker container bob 11](#_Toc196590815)

[Hình 6: Docker container bob 12](#_Toc196590816)

[Hình 7: Docker container alice 13](#_Toc196590817)

[Hình 8:Docker container alice 13](#_Toc196590818)

[Hình 9: Docker 14](#_Toc196590819)

[Hình 10: Đẩy lab lên github 15](#_Toc196590820)

# **Giới thiệu chung về bài thực hành**

Bài thực hành này được thiết kế nhằm giúp sinh viên hiểu rõ cách thức ẩn và trích xuất dữ liệu trong luồng video H264-AVC bằng cách sử dụng trường SEI (Supplemental Enhancement Information). Sinh viên sẽ được rèn luyện kỹ năng thao tác trực tiếp trên cấu trúc NALU của video, chỉnh sửa các payload SEI, đóng gói và giải mã video thông qua các công cụ như FFmpeg và các script Python chuyên dụng.

H264-AVC là một chuẩn mã hóa video phổ biến, không chỉ cung cấp khả năng nén hiệu quả mà còn hỗ trợ nhúng thông tin phụ trợ vào luồng video thông qua các gói SEI. Các trường SEI cho phép chèn metadata, watermark, hoặc dữ liệu bí mật mà không làm ảnh hưởng trực tiếp đến nội dung hình ảnh. Kỹ thuật này đặc biệt hữu ích trong các ứng dụng như truyền phát video an toàn, giấu thông tin trong video, hoặc hỗ trợ phân tích video nâng cao.

Trong bài thực hành này, sinh viên sẽ trải qua toàn bộ quy trình: từ việc tách luồng raw H.264 từ file MP4, tìm kiếm các gói SEI có sẵn, sửa đổi nội dung SEI để giấu dữ liệu, đóng gói trở lại định dạng MP4 chuẩn và xác minh khả năng trích xuất thông tin sau khi truyền tải. Thông qua quá trình này, sinh viên sẽ hiểu rõ cơ chế hoạt động của các gói SEI, cách mã hóa và đóng gói video tuân thủ chuẩn H264-AVC, và các ràng buộc kỹ thuật cần lưu ý để bảo toàn dữ liệu khi truyền đi.

Sau khi hoàn thành bài thực hành, sinh viên sẽ có cái nhìn tổng quan về kỹ thuật nhúng dữ liệu trong video H.264, nắm vững quy trình chỉnh sửa trường SEI, hiểu rõ các yếu tố ảnh hưởng đến tính toàn vẹn của dữ liệu nhúng, cũng như áp dụng các công cụ mã hóa/giải mã video trong các tình huống thực tế.

# **Nội dung và hướng dẫn bài thực hành**

## **Mục đích**

* Hiểu tổng quan về quy trình mã hóa và giải mã video theo chuẩn H264-AVC.
* Nắm vững các bước thực hành sử dụng công cụ và các thuật toán liên quan đến mã hóa video trong chuẩn H264-AVC.
* Áp dụng kỹ thuật giấu tin bằng cách chèn thông tin vào trường SEI (Supplemental Enhancement Information) của luồng H264.
* Thực hành đóng gói video H264 thành định dạng MP4, đảm bảo dữ liệu giấu không bị mất trong quá trình truyền tải và giải mã.

## **Yêu cầu đối với sinh viên**

* Có kiến thức cơ bản về kỹ thuật giấu tin trong đa phương tiện (steganography), đặc biệt là trên video.
* Hiểu rõ khái niệm, cấu trúc luồng dữ liệu và các ứng dụng của chuẩn video H264-AVC trong thực tế.
* Thành thạo việc sử dụng các công cụ xử lý media và các script Python hỗ trợ xử lý dữ liệu H264.

## **Nội dung thực hành**

Sinh viên khởi động bài lab

Chạy lệnh:

*labtainer -r stego\_h264\_video\_sei\_code*

*(Chú ý: sinh viên sử dụng <TÊN\_TÀI\_KHOẢN> của mình để nhập thông tin người thực hiện bài lab khi có yêu cầu, để sử dụng khi chấm điểm.)*

Sau khi khởi động bài lab,có 2 container hiện lên là bob và alice. Sinh viên thực hiện làm lab theo yêu cầu trên 2 container này

***TASK 1: Mã hóa video chuẩn h264 và chuyển dổi thành luồng h264 thô***

* Sau khi khởi động xong một terminal ảo sẽ xuất hiện, terminal sử dụng để thực hành bài lab
* Trong terminal bob đã có sẵn 1 file video.mpeg.
* Sinh viên sử dụng công cụ ffmpeg để nén video mẫu thành 1 video theo chuẩn mã hóa H264 :

***ffmpeg -i video.mpeg -c:v libx264 -profile:v high -x264opts cabac=1 -crf 28 video\_h264.mp4***

* Câu lệnh này sử dụng ffmpeg để chuyển đổi video từ định dạng mpeg sang video H.264 với các tùy chọn mã hóa:
* Video mới được tạo có tên là video\_h264.mp4. Sau khi có được video này, ta tiến hành chuyển đổi nó thành dạng raw h264,phục vụ cho việc giấu tin sau này

***ffmpeg -i video\_h264.mp4 -c:v copy -bsf:v h264\_mp4toannexb raw.h264***

***TASK 2: Phân tích luồng h264 thô để tìm thành phần giấu tin***

* SEI (Supplemental Enhancement Information) là một loại dữ liệu đặc biệt trong chuẩn mã hóa video H264-AVC, được thiết kế để mang theo các thông tin phụ trợ bổ sung cho quá trình giải mã hoặc xử lý video.Thông tin SEI không ảnh hưởng đến nội dung hình ảnh của video và thường được dùng để truyền:
  + Metadata mô tả video,
  + Thông tin thời gian (timing),
  + Các chỉ dẫn hỗ trợ giải mã,
  + Các thông tin mở rộng như watermarking hoặc giấu dữ liệu bí mật.
* Trong cấu trúc SEI NALU có phần payload Data. Nếu là user\_data\_unregistered, phần đầu là UUID (16 bytes), theo sau là nội dung dữ liệu.
* Đặc điểm quan trọng:
  + Các thiết bị hoặc trình phát video sẽ bỏ qua SEI nếu không hiểu nội dung, nên nó không phá vỡ luồng video.
  + SEI rất thích hợp để thực hiện kỹ thuật giấu tin mà không làm thay đổi khung hình video.
  + UUID trong user\_data\_unregistered cho phép phân biệt các loại dữ liệu ẩn khác nhau.
* Vậy nên ta sẽ tìm phần này trong raw.h264 để giấu thông tin.
* Dùng Python script (find\_sei.py) để tìm SEI #0 (UUID gốc và thông điệp hiện tại) : ***python3 find\_sei.py raw.h264.***
* Trên termial sẽ hiển thị thông tin xem có tìm được SEI trong raw.h264 không, vị trí SEI, UUID và thông điệp gốc trong SEI đó

***TASK 3: Giấu tin vào nội dung SEI***

* Sau khi đã tìm được thông tin về SEI#0, ta tiến hành giấu thông điệp vào trong nó: ***python3 patch\_sei\_message.py raw.h264 patched.h264 "MSV***" (sinh viên điền mã sinh viên của mình làm thông điệp cần giấu)
* Sau khi sử dụng Python script này, nội dung trong SEI#0 đã được chèn thêm thông điệp và ghi lại thành 1 dạng thô mới là patched.h264
* Tiến hành đóng gói file thô patched.h264 thành 1 video dạng mp4: ***ffmpeg -f h264 -i patched.h264 -c:v copy final\_output.mp4***
* Ta tiến hành xem thử video mới này xem có gì lỗi không (nếu giấu tin không đúng cách có thể phá vỡ cấu trúc video, khiến video bị lỗi và có thể thấy được nhưng thay dổi bằng mắt thường): ***ffplay -autoexit final\_output.mp4***

***TASK 4: Giãi mã thông điệp***

* Ta khởi động dịch vụ ssh bên máy alice: ***sudo service ssh start***
* Gửi video vừa tạo được sang cho alice qua ssh : ***scp final\_output.mp4 ubuntu@"ip máy alice":~***
* Để giải mã được thông điệp, alice cũng cần chuyển video thành luông thô h264: ***ffmpeg -i final\_output.mp4 -c:v copy -bsf:v h264\_mp4toannexb raw.h264*** rồi tiến hành giải mã thông điệp giấu trong video bằng cách sử dụng python script***: python3 find\_sei.py raw.h264***
* Thông điệp được dấu trong SEI sẽ được ghi vào file extract\_data.txt. đọc file này sẽ tìm được thông điệp bob đã giấu: ***cat extract\_data.txt***

Kết thúc lab:

* Trên terminal khởi động lab, sinh viên sử dụng lệnh:

***Stoplab***

* Khi bài lab kết thúc, một tệp lưu kết quả được tạo và lưu vào một vị trí được hiển thị bên dưới stoplab. Sinh viên cần nộp file .lab để chấm điểm.
* Để kiểm tra kết quả khi trong khi làm bài thực hành sử dụng lệnh:

***checkwork <tên bài thực hành>***

* Sinh viên cần nộp file .lab để chấm điểm.
* Kiểm tra kết quả trong quá trình làm bài:

***checkwork <tên bài lab>***

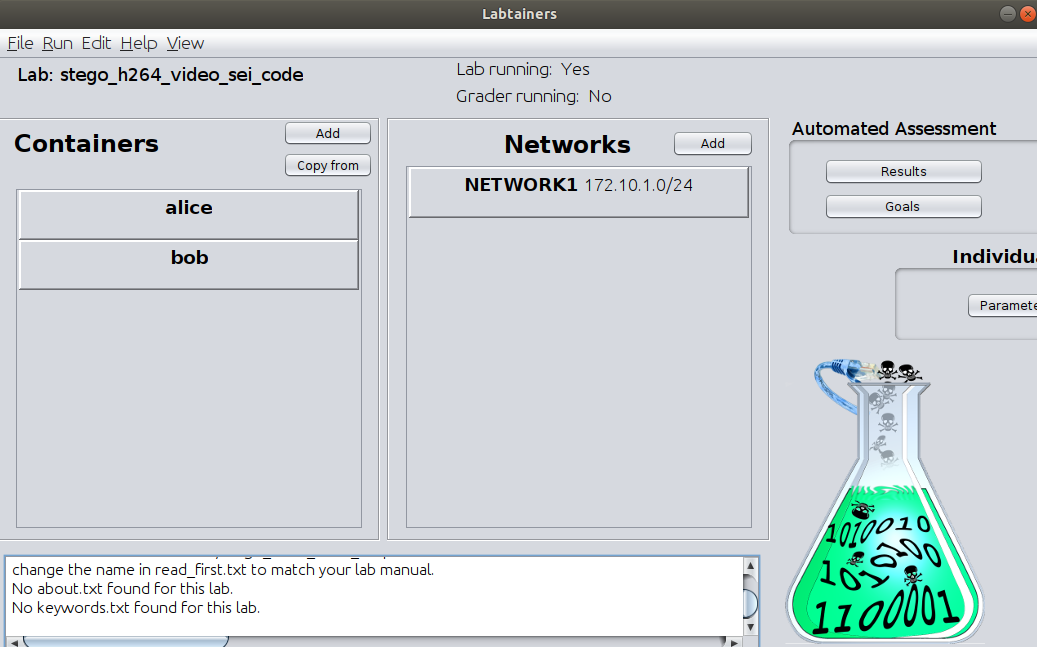
* Khởi động lại bài lab: Trong quá trình làm bài sinh viên cần thực hiện lại bài lab, dùng câu lệnh:

***labtainer -r stego\_h264\_video\_encoding***

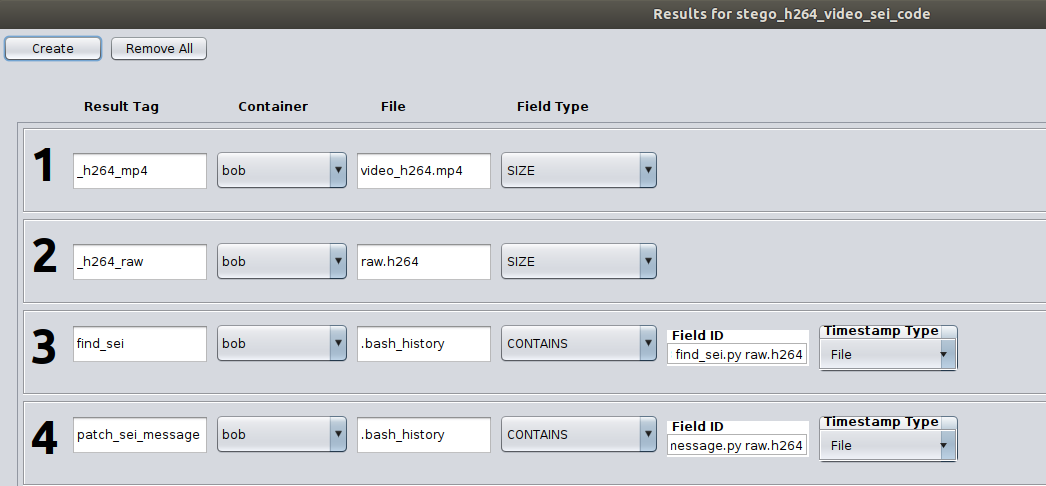
# **Phân tích yêu cầu của bài thực hành**

 Bài thực hành gồm 2 máy tính bob và alice, trong đó bob có chứa 1 video mẫu dạng mpeg .. Để hoàn thành bài thực hành sinh viên cần tìm tạo được 1 video chuẩn h264 từ video gốc, được thành phần SEI trong nó rồi tiến hành giấu tin. Tiếp đến là gửi video sang máy alice rồi giải mã thông điệp

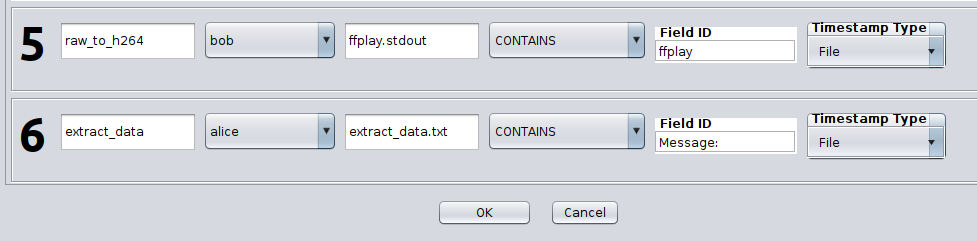
# **Cài đặt và cấu hình máy ảo**



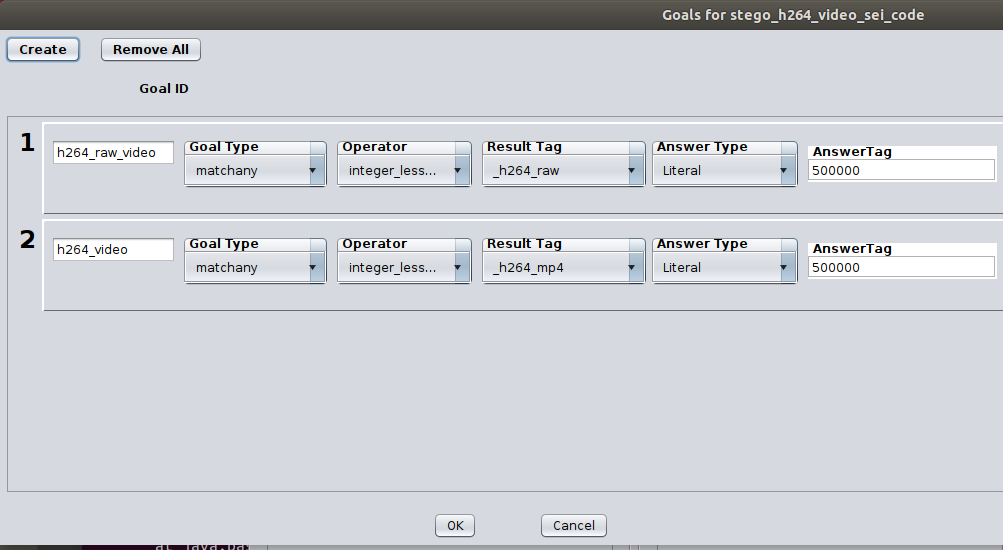
Hình 1: Giao diện Labedit



Hình 2: Cấu hình result



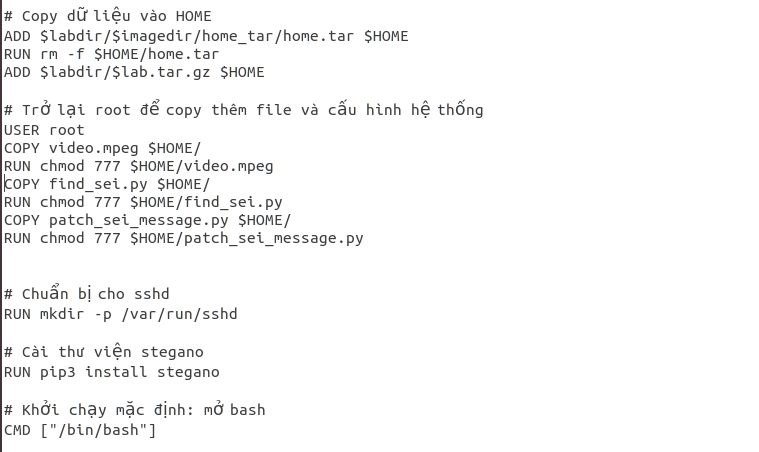
Hình 3: Cấu hình Result



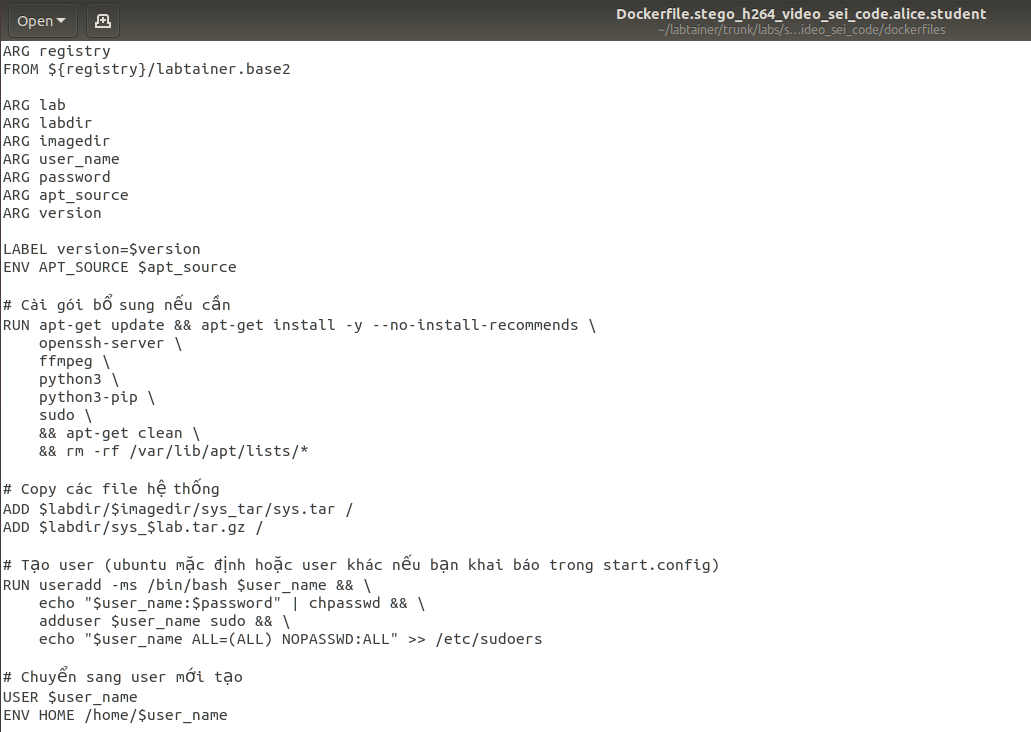
Hình 4: Cấu hình Goal



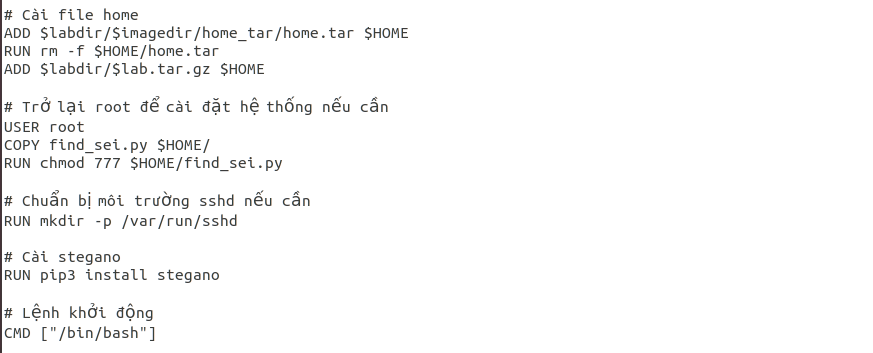
Hình 5: Docker container bob



Hình 6: Docker container bob



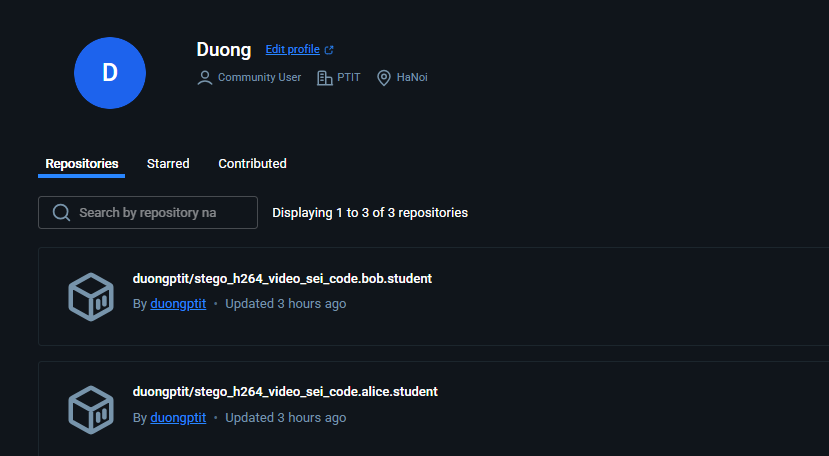
Hình 7: Docker container alice



Hình 8:Docker container alice

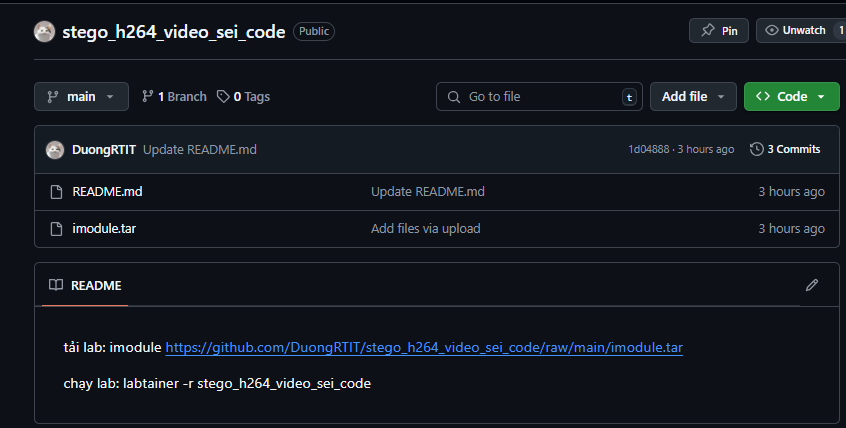
# **Tích hợp và triển khai**

## **Docker Hub**



Hình 9: Docker

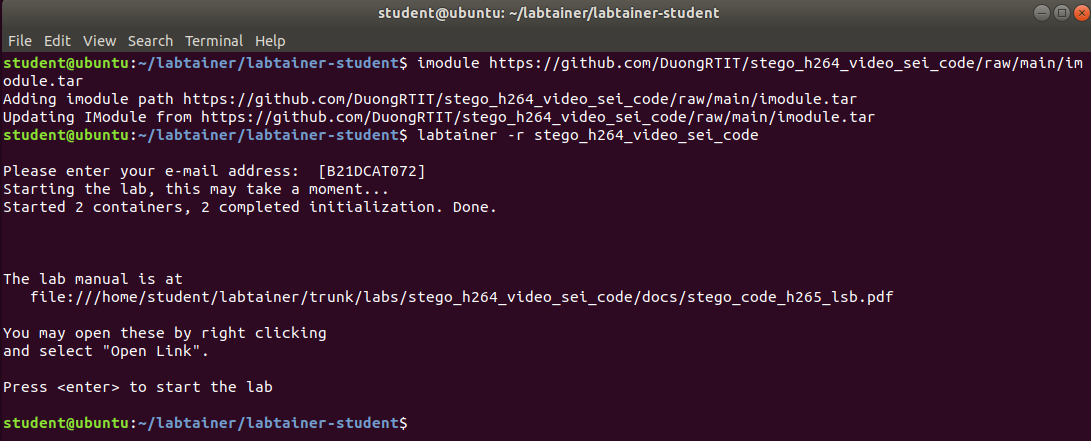
## **GitHub**



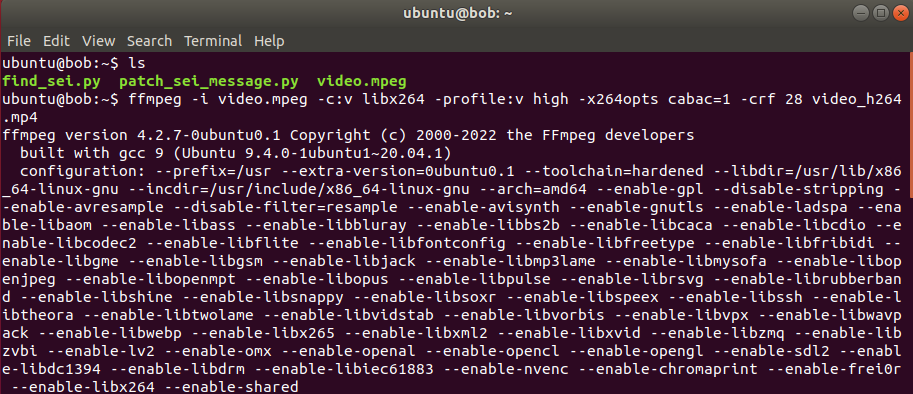
Hình 10: Đẩy lab lên github

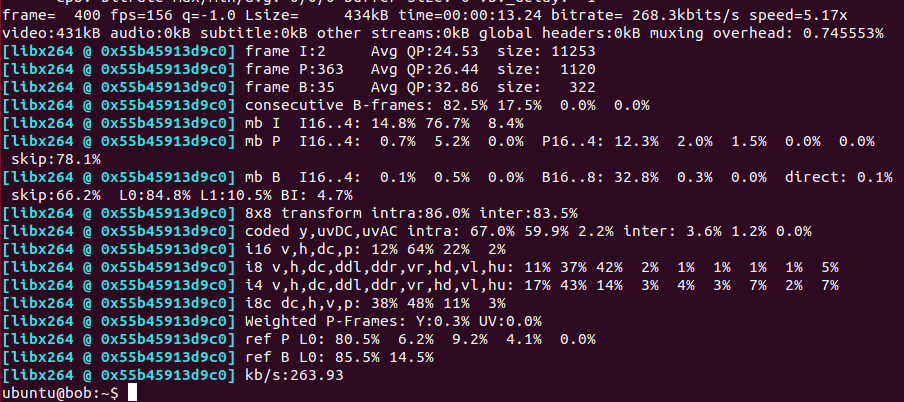
# **Thử nghiệm và đánh giá**

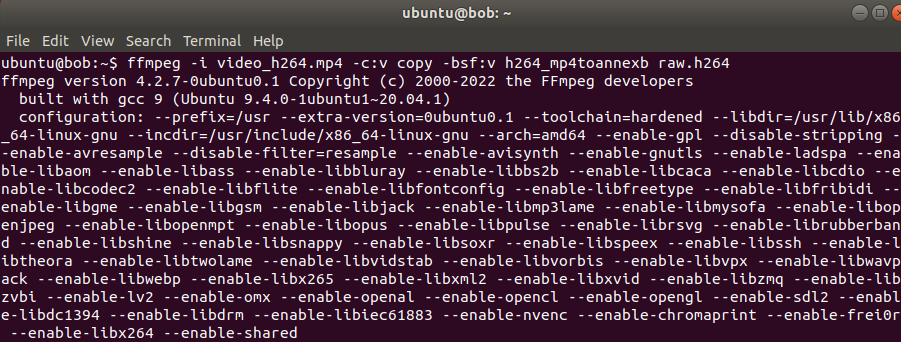
* Tải bài lab về từ Github và chạy bài lab:

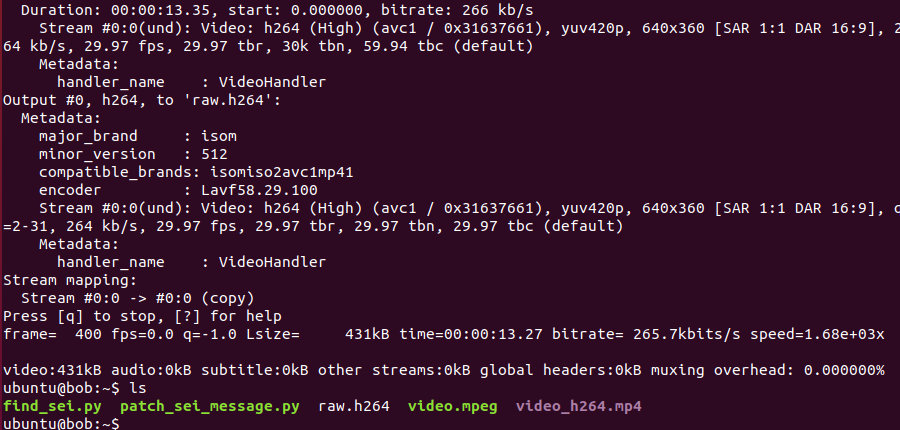


* Thực hiện Task 1:

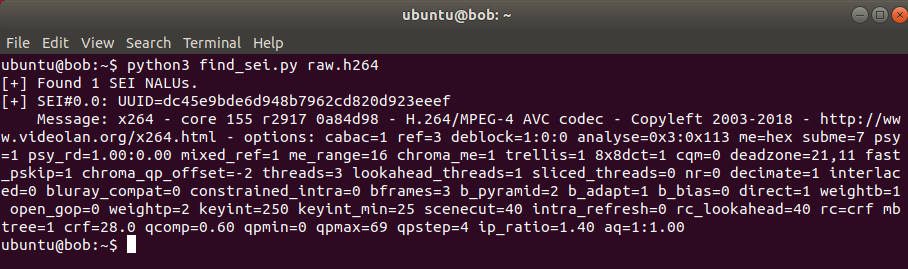




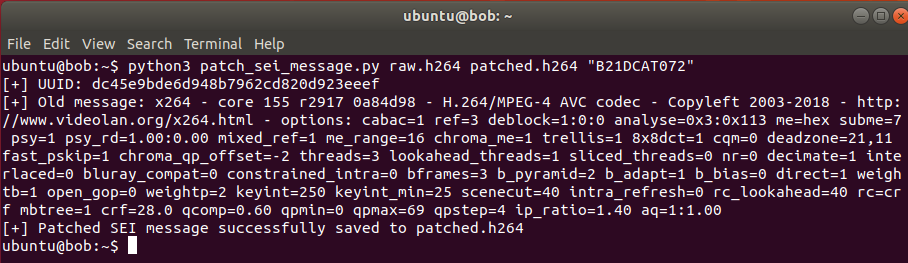


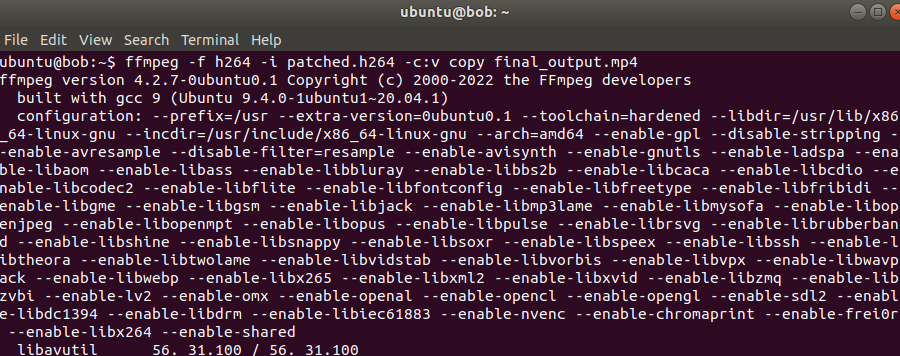


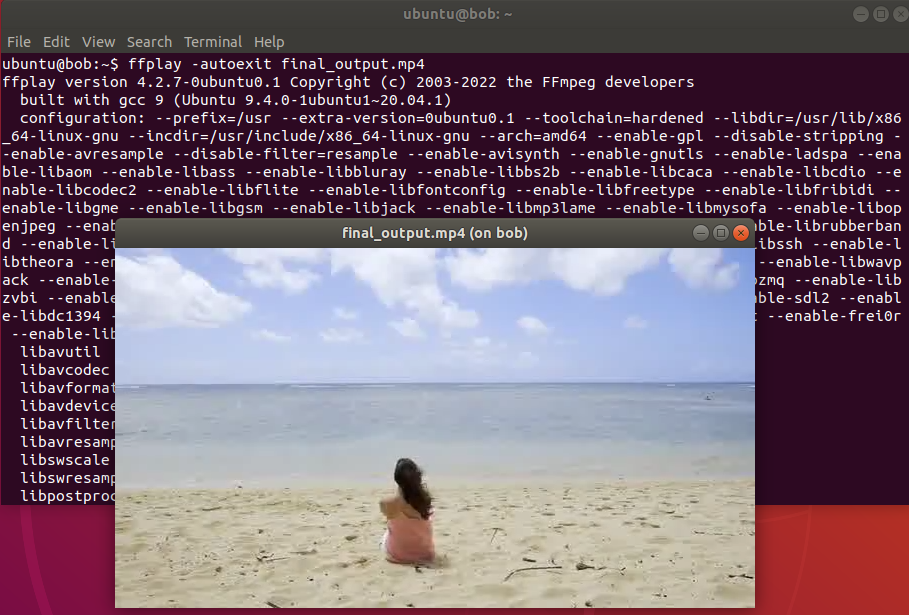
* Thực hiện task 2:

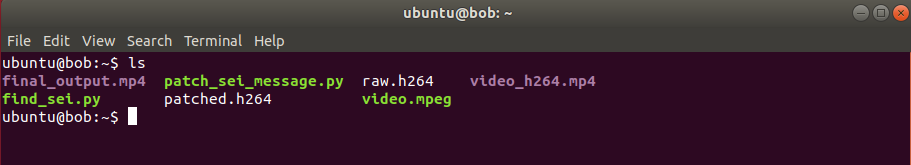


* Thực hiện task 3:

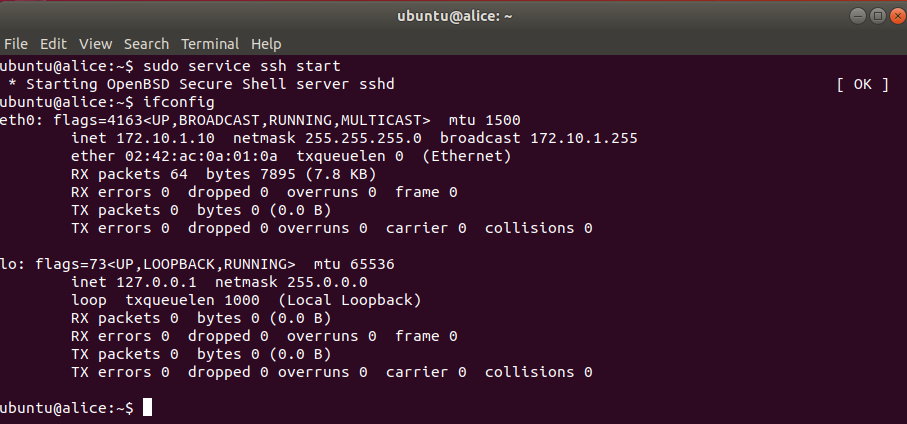


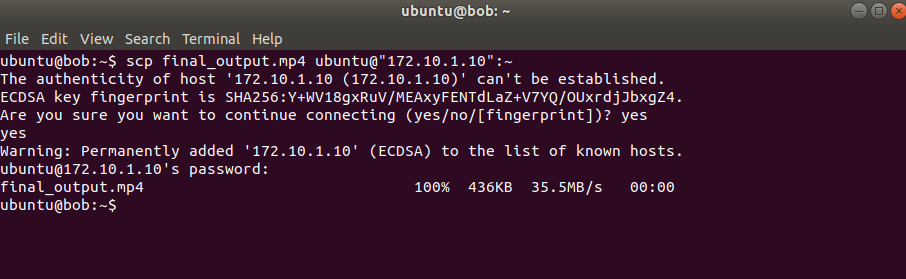




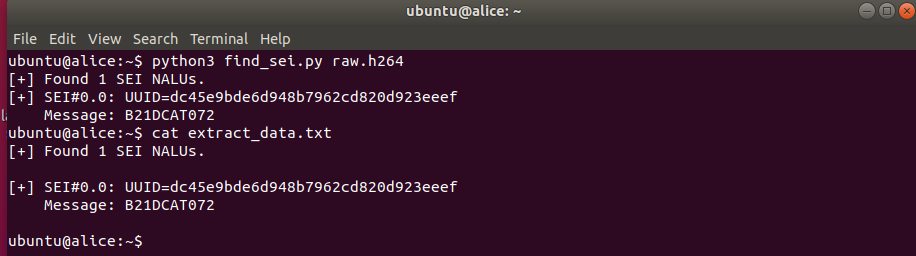


* Thực hiện task 4:









* Checkwork:

