**KHOA CÔNG NGHỆ THÔNG TIN**Logo, company name

Description automatically generated

**BÁO CÁO CUỐI KỲ**

**MÔN ĐIỆN TOÁN ĐÁM MÂY**

***ĐỀ TÀI:***

**VIẾT ỨNG DỤNG**

**KẾT HỢP NHIỀU DOCKER VỚI NHAU**

**GVHD: TS. HUỲNH XUÂN PHỤNG**

**NHÓM SINH VIÊN THỰC HIỆN:**

**1. PHAN DUY LÂM - 20110322**

**2. HOÀNG CÔNG THÀNH NHÂN - 20110687**

**3. TRẦN VĂN QUẢNG - 20110160**

**TP. HỒ CHÍ MINH – 11/2022**

**LỜI CẢM ƠN**

Đầu tiên nhóm chúng em xin được phép gửi lời cảm ơn chân thành đến với Khoa Công Nghệ Thông Tin – Trường Đại Học Sư Phạm Kỹ Thuật Thành Phố Hồ Chí Minh đã tạo điều kiện tốt nhất cho nhóm chúng em được học tập, phát triển nền tảng kiến thức sâu sắc từ đó có thể được đề tài kết thúc môn học này.

Bên cạnh đó nhóm chúng em xin gửi đến thầy Huỳnh Xuân Phụng lời cảm ơn sâu sắc nhất trong suốt thời gian dài học tập cũng như thực hiện đề tài trong thời gian qua.

Với sự chỉ bảo nhiệt tình, giảng dạy tận tình và đầy đủ kiến thức của thầy Huỳnh Xuân Phụng, nhóm chúng em đã học tập và hiểu được những kiến thức cơ bản về AWS EC2. Một lần nữa, nhóm xin cảm ơn thầy!

Tuy nhiên lượng kiến thức là vô tận và với khả năng còn nhiều hạn chế, chúng em đã cố gắng hết sức để hoàn thành đề tài một cách tốt nhất có thể. Chính vì vậy, việc xảy ra những thiếu sót là điều khó có thể tránh khỏi. Nhóm chúng em hi vọng nhận được sự góp ý, nhận xét và phê bình tận tình của quý thầy (cô)để có thể rút ra được các bài học kinh nghiệm từ đó hoàn thiện và cải thiện nâng cấp lại đề tài của mình một cách tốt nhất có thể.

Nhóm chúng em xin chân thành cảm ơn quý thầy(cô)!

**Sinh viên thực hiện đồ án**

Mục lục

[PHẦN MỞ ĐẦU 4](#_Toc122289603)

[PHẦN NỘI DUNG 5](#_Toc122289605)

[CHƯƠNG 1. CƠ SỞ LÝ THUYẾT 5](#_Toc122289606)

[1.1. Các nền tảng phát triển ứng dụng 5](#_Toc122289607)

[1.2. Các service cần thiết 5](#_Toc122289608)

[CHƯƠNG 2. THIẾT KẾ VÀ CÀI ĐẶT ỨNG DỤNG 12](#_Toc122289609)

[Thiết kế và cài đặt môi trường 15](#_Toc122289610)

[PHẦN KẾT LUẬN 22](#_Toc122289612)

# PHẦN MỞ ĐẦU

## Tính cấp thiết của đề tài

Docker là nền tảng phần mềm cho phép ta dựng, kiểm thử và triển khai ứng dụng một cách nhanh chóng. Docker đóng gói phần mềm vào các đơn vị tiêu chuẩn hóa được gọi là container có mọi thứ mà phần mềm cần để chạy, trong đó có thư viện, công cụ hệ thống, mã và thời gian chạy. Bằng cách sử dụng Docker, bạn có thể nhanh chóng triển khai và thay đổi quy mô ứng dụng vào bất kỳ môi trường nào và biết chắc rằng mã của bạn sẽ chạy được. Việc chạy Docker trên AWS đem đến cho các nhà phát triển và quản trị viên một phương thức dựng, vận chuyển và chạy ứng dụng phân phối ở quy mô bất kỳ có chi phí thấp và độ tin cậy cao.

1. **Đối tượng nghiên cứu**

Trong phạm vi đề tài, nhóm sẽ tập trung nghiên cứu lý thuyết nền tảng liên quan tới Docker, áp dụng kiến thức đã học được từ môn Điện toán đám mây về AWS để hoàn thành ứng dụng quản lý sinh viên cơ bản.

1. **Phạm vi nghiên cứu**

* Lý thuyết cách hoạt động của EC2.
* Lý thuyết cách hoạt động của Docker.
* Cách ứng dụng quy trình CICD vào phát triển phần mềm.

1. **Kết quả dự kiến đạt được**

* Hiểu được cách dùng Docker và kết hợp Docker cùng các nền tảng để triển khai ứng dụng.

# PHẦN NỘI DUNG

## CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

### Các nền tảng phát triển ứng dụng

* Front end: ASP.NET
* Backend: Java, JSP, Servlet
* Database: MySQL
* Github
* Github Action

### Các service cần thiết

#### Docker

* + - 1. **Docker là gì?**

Docker chính là một nền tảng được sử dụng để cung cấp cho ta cách building, deploying và cả running ứng dụng một cách dễ dàng hơn. Docker sẽ hoạt động thông qua cách sử dụng những containers ở trên nền tảng ảo hóa. Lúc đầu, chúng sẽ được viết bằng Python và hiện tại thì đã chuyển sang Golang.

#### Các khái niệm cơ bản trong Docker:

* Docker Engine: là thành phần chính của Docker, như một công cụ để đóng gói ứng dụng.
* Docker Hub: là một “github for docker images”. Trên Docker Hub có hàng ngàn public images được tạo bởi cộng đồng cho phép bạn dễ dàng tìm thấy những image mà bạn cần. Và chỉ cần pull về và sử dụng với một số config mà chúng ta mong muốn.
* Images: Docker image là một file bất biến - không thay đổi, chứa các source code, libraries, dependencies, tools và các files khác cần thiết cho một ứng dụng để chạy. Images chỉ là các mẫu, chúng ta không thể start hoặc run chúng. Những gì chúng ta có thể làm là sử dụng mẫu đó làm cơ sở để xây dựng một container. Một container cuối cùng chỉ là một image đang chạy. Khi tạo một container, nó sẽ thêm một lớp có thể ghi (writable) lên trên image bất biến, nghĩa là bây giờ ta có thể sửa đổi nó.
* Container: là một instance của một image. Ta có thể create, start, stop, move or delete container dựa trên Docker API hoặc Docker CLI.
* Docker Client: là một công cụ giúp người dùng giao tiếp với Docker host.
* Docker Daemon: lắng nghe các yêu cầu từ Docker Client để quản lý các đối tượng như Container, Image, Network và Volumes thông qua REST API. Các Docker Daemon cũng giao tiếp với nhau để quản lý các Docker Service.
* Dockerfile: là một tập tin bao gồm các chỉ dẫn để build một image.
* Volumes: là phần dữ liệu được tạo ra khi container được khởi tạo.
  + - 1. Docker Container

Các containers cho phép lập trình viên đóng gói một ứng dụng với tất cả các phần cần thiết, chẳng hạn như thư viện và các phụ thuộc khác, và gói tất cả ra dưới dạng một package.

Bằng cách đó, nhờ vào container, ứng dụng sẽ chạy trên mọi máy Linux khác bất kể mọi cài đặt tùy chỉnh mà máy có thể có khác với máy được sử dụng để viết code.

Theo một cách nào đó, Docker khá giống virtual machine. Nhưng tại sao Docker lại phát triển, phổ biến nhanh chóng? Đây là những nguyên nhân:

* Tính dễ ứng dụng: Docker rất dễ cho mọi người sử dụng từ lập trình viên, sys admin… nó tận dụng lợi thế của container để build, test nhanh chóng. Có thể đóng gói ứng dụng trên laptop của họ và chạy trên public cloud, private cloud… Câu thần chú là “Build once, run anywhere”.
* Tốc độ: Docker container rất nhẹ và nhanh, bạn có thể tạo và chạy docker container trong vài giây.
* Môi trường chạy và khả năng mở rộng: Bạn có thể chia nhỏ những chức năng của ứng dụng thành các container riêng lẻ. Ví dụng Database chạy trên một container và Redis cache có thể chạy trên một container khác trong khi ứng dụng Node.js lại chạy trên một cái khác nữa. Với Docker, rất dễ để liên kết các container với nhau để tạo thành một ứng dụng, làm cho nó dễ dàng scale, update các thành phần độc lập với nhau.

Với xu hướng dịch chuyển sang microservices của các hệ thống lớn, Docker đang làm một thành phần cực kỳ quan trọng, làm cho nó trở thành một phần của nhiều công cụ DevOps. Hiện tại thế giới bắt đầu sử dụng thêm một công cụ quản lý container tiên tiến khác là Kubernetes (Đọc thêm bài Kubernetes là gì?)

* + 1. AWS

AWS (Amazon Web Service) là một nền tảng về điện toán đám mây nổi tiếng của Amazon và đang chiếm thị phần lớn nhất tính đến thời điểm hiện tại. Nó cung cấp cho doanh nghiệp những giải pháp về storage, computing power, database hay developer tools,…

Hiện nay, AWS đã có hơn 175 dịch vụ trên nền tảng của họ. Sử dụng AWS bạn sẽ có thể sử dụng một trung tâm dữ liệu với công nghệ điện toán đám mây ở bất cứ đâu trên toàn thế giới. Bởi với AWS thì đám mây không chỉ co cụm trong phạm vi của một quốc gia nữa.

#### Amazon EC2

Amazon EC2 cung cấp các ứng dụng máy tính ảo hoá có thể mở rộng về khả năng xử lý cùng các thành phần phần cứng ảo như bộ nhớ máy tính (RAM), vi xử lý, linh hoạt trong việc lựa chọn các phân vùng lưu trữ dữ liệu ở các nền tảng khác nhau và sự an toàn trong quản lý dịch vụ bởi kiến trúc ảo hoá đám mây mạnh mẽ của AWS.

Amazon EC2 Instance là một Cloud server. Với một tài khoản bạn có thể tạo và sử dụng nhiều Amazon EC2 Instance. Các Amazon EC2 Instance được chạy trên cùng một server vật lý và chia sẻ memory, CPU, ổ cứng...

Tuy nhiên do tính chất của Cloud service nên mỗi một Instance được hoạt động giống như một server riêng lẻ.

EC2 cho phép người dùng xây dựng ứng dụng để tự động mở rộng quy mô theo nhu cầu thay đổi và thời kỳ cao điểm, đồng thời giúp việc triển khai máy chủ ảo và quản lý lưu trữ trở nên đơn giản, giảm thiểu nhu cầu đầu tư vào phần cứng và giúp hợp lý hóa các quy trình phát triển.

Một số lợi ích và tính năng thu hút các nhà phát triển đến EC2 cho điện toán đám mây. Đứng đầu trong số này là:

* Khả năng đáp ứng các yêu cầu về dung lượng thay đổi: Việc mở rộng quy mô dễ dàng của EC2 giúp loại bỏ các trở ngại phát triển xảy ra khi các ứng dụng yêu cầu nhiều tài nguyên hơn.
* Linh hoạt trong cấu hình: Người dùng có thể chọn kích thước bộ nhớ, CPU và kích thước phân vùng khởi động được tối ưu hóa cho hệ điều hành họ chọn.
* Tích hợp: EC2 có thể tích hợp với các dịch vụ AWS khác, chẳng hạn như RDS, SimpleDB và SQS.
* Kiểm soát chính xác: Người dùng có quyền truy cập quản trị vào các phiên bản của họ, có thể dừng và bắt đầu các phiên bản trong khi vẫn giữ lại dữ liệu phân vùng khởi động và có thể truy cập đầu ra bảng điều khiển cho phiên bản đó.
* Bảo mật: Người dùng có thể kiểm soát phiên bản nào vẫn ở chế độ riêng tư và phiên bản nào có khả năng tiếp xúc với Internet. EC2 tận dụng Đám mây riêng ảo của Amazon (VPC) để bảo mật và các doanh nghiệp có thể kết nối cơ sở hạ tầng CNTT an toàn của họ với các tài nguyên trong VPC.
* Chi phí: Trong số một số tùy chọn giá cả, EC2 cung cấp mức giá hàng giờ hợp lý.

#### CICD

Quy trình phát triển phần mềm (SDLC) truyền thống được diễn ra qua 6 giai đoạn cụ thể như lập kế hoạch, thực hiện kế hoạch, kiểm thử, thiết lập thông tin, triển khai và bảo trì. SDLC là mô hình được sử dụng trong nhiều năm qua và nó đã thể hiện được rất nhiều những ưu điểm trong ngành công nghệ thông tin.

Tuy nhiên, đối mặt với nhu cầu của con người ngày càng tăng như hiện nay thì SDLC lại dần bộc lộ những điểm yếu của mình. Nhiều người đã cho rằng SDLC đã không còn phù hợp như trước đây và đã bị thay thế bởi những mô hình mới có những lợi ích về tốc độ, tự động hóa trong công việc trong đó có CI/CD. Vậy CI/CD là gì?

* + - 1. **CI**

Continuous Integration hay CI là một phương pháp phát triển phần mềm. Phương pháp này đòi hỏi các thành viên trong đội cần phải tích hợp công việc với nhau một cách thường xuyên, mỗi ngày cần ít nhất một lần tích hợp. Mỗi lần tích hợp sẽ được xây dựng một cách tự động nhằm mục đích phát hiện ra những lỗi phát sinh một cách nhanh nhất có thể. Khi sử dụng CI sẽ giúp làm giảm những vấn đề về tích hợp và cho phép các developer phát triển phần mềm được nhanh hơn và đúng tiến độ hơn.Quy trình làm việc của CI có thể được diễn ra như sau:

* Các developer sẽ commit code thể lên repo.
* Tiếp theo, CI server sẽ thực hiện giám sát trên repo và kiểm tra xem có bất kỳ sự thay đổi nào trên repo hay không.

Khi xảy ra những thay đổi, CI server sẽ phải hiện ra code mới nhất từ repo và sau đó sẽ build, chạy các unit test và integration test. Tiếp sau đó, CI server sẽ tạo ra những phản hồi và gửi đến cho các thành viên trong dự án. Và CI server lại tiếp tục chờ đợi những thay đổi từ repo.

Mỗi lần mà Dev làm xong các task của mình, họ sẽ chạy trên local trước để kiểm tra trước khi commit code lên repo. Đây là một công đoạn thường xuyên diễn ra tại bất cứ thời điểm nào trong ngày. Việc xây dựng tích hợp sẽ không xảy ra nếu chúng không ảnh hưởng đến repo.

* + - 1. **CD**

Nếu CI đảm nhận nhiệm vụ xây dựng và kiểm tra một cách tự động thì CD lại có nhiệm vụ cao hơn một chút. CD được viết tắt bởi Continuous Delivery - chuyển giao liên tục. Đây là quá trình nâng cao hơn chút đó là kiểm tra tất cả những thay đổi về code đã được build và code trong môi trường kiểm thử. CD cho phép các lập trình viên tự động hóa phần mềm testing, kiểm tra phần mềm qua nhiều thước đo trước khi triển khai.

Những bài test này có thể bao gồm UI testing, integration testing, API testing,... CD sử dụng Deployment Pipeline giúp chia quy trình chuyển giao thành các giai đoạn. Mỗi giai đoạn có những mục tiêu riêng để xác minh chất lượng của các tính năng từ một góc độ vô cùng khác để có thể kiểm định được chức năng và tránh những lỗi phát sinh ảnh hưởng đến người dùng.

* + - 1. **CICD**

Từ hai khái niệm trên ta có thể hiểu một cách đơn giản thì CI/CD chính là quá trình làm việc liên tục và tự động hóa của phần mềm. Để quá trình kiểm thử được diễn ra liên tục thì CI CD phải được tích hợp vào trong vòng đời phát triển phần mềm. Trong những năm gần đây, CI CD đã trở thành thông lệ trong quá trình phát triển phần mềm, đây là chìa khóa kết hợp các khâu kiểm thử và phát triển phần mềm lại với nhau.Ngoài ra, CI/CD cũng trở thành thông lệ tốt nhất trong ngành công nghệ thông tin để giữ cho ứng dụng có thể được triển khai bất cứ lúc nào cũng như đẩy mã triển khai đó vào kho sản xuất khi có mã mới.

* + 1. **Github Action**

GitHub Actions là một event-driven nghĩa là chương trình chạy một loạt các dòng lệnh khi bạn nhận được 1 event, ví dụ mỗi lần một developer nào đó tạo một pull request cho một repository nào đó thì dòng lệnh build được thực thi.

Các thành phần trong github action:

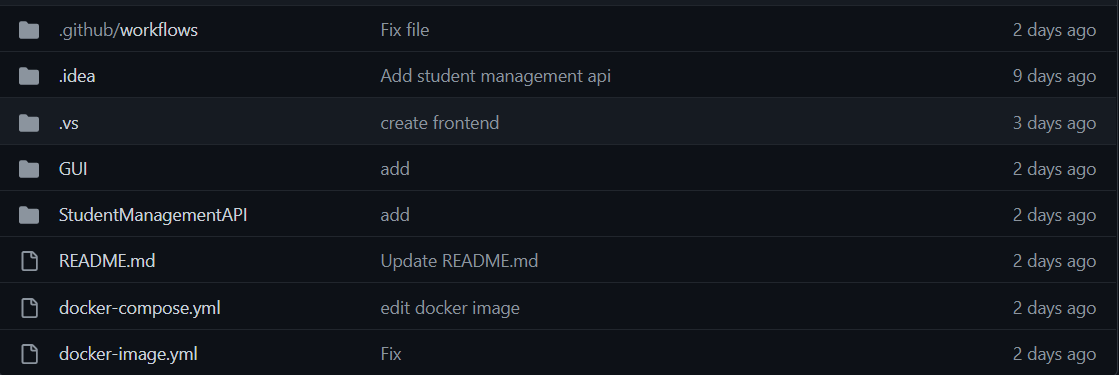
* Workflows: Là một thủ tục tự động, workflows được tạo bởi một hay nhiều job và các job này có thể chạy theo định kỳ hoặc trigger chạy khi nhận được một event nào đó.
* Events: Là một hành động ấn định để trigger một workflow, ví dụ khi một developer tạo một nhánh develop nào đó thì event này được gửi tới workflow, hoặc bạn có thể sử dụng webhook khi nhận một event bên ngoài trigger vào workflow.
* Jobs: Là một tập các step để thực thi trên cùng một runner. Mặc định với một workflow chạy nhiều job thì các job này chạy song song (parallel).
* Steps: Là những tác vụ riêng rẽ để chạy các command (mà còn gọi là action), mỗi step trong một job thực thi trên .
* Actions: Là một building block, có tính portable được build sẵn, bạn có thể tạo riêng một action riêng cho bạn.
* Runners: Là một server mà GitHub Actions cài đặt các ứng dụng trong nó. Bạn có thể sử dụng runner của github hoặc bạn có thể sử dụng server riêng của bạn. Github runner hỗ trợ các hệ điều hành: Ubuntu Linux, Microsoft Windows, and macOS.

## CHƯƠNG 2. THIẾT KẾ VÀ CÀI ĐẶT ỨNG DỤNG

Trong khuôn khổ đề tài, nhóm sẽ thực hiện một phần mềm quản lý sinh viên đơn giản với các chức năng như sau:

* Xem danh sách sinh viên
* Thêm sinh viên
* Xóa sinh viên
* Sửa thông tin sinh viên

1. **Cấu trúc project**



Thư mục GUI chứa source code của project ASP.NET chạy front-end cho ứng dụng Quản lý sinh viên đơn giản.

Thư mục StudentManagementAPI chứa source code của project JAVA Servlet chạy backend cho ứng dụng.

Thư mục .github/workflows chứa file workflows cho Github Action sử dụng để tự động hóa build và push image lên Docker Hub.

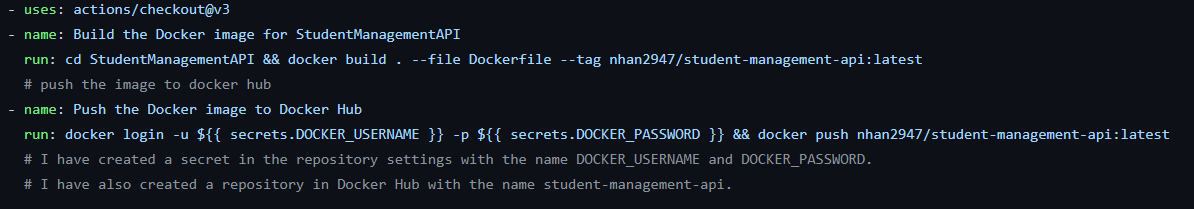
1. **Sử dụng Github Action để tự động hóa quá trình build và push Docker image**

File workflows mà nhóm sử dụng cho Repository này như sau:



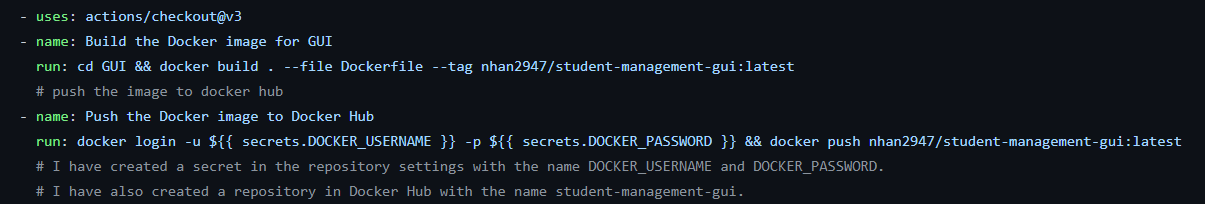
Ta có thể chia workflow này thành 2 step:

Step 1: Tự động build image cho ứng dụng JAVA Servlet, sau đó push lên Docker Hub

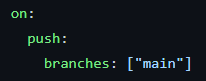


* Quá trình này ta đảm bảo đã cung cấp đầy đủ username và password để đăng nhập vào DOCKER HUB, vì lý do bảo mật, ta lưu các thông tin này thành biến secrets của repo, quản lý bởi github.

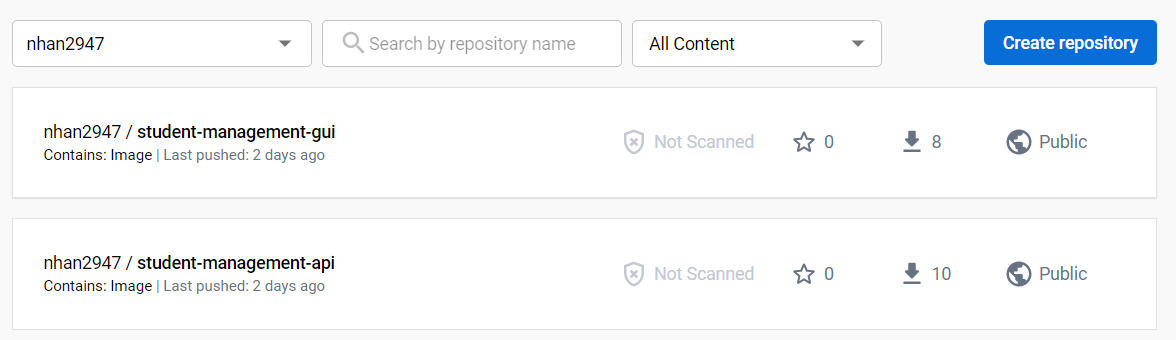
Step 2: Tương tự bước 1, tự động build image cho ứng dụng ASP.NET GUI, sau đó push lên Docker Hub



Quá trình này sẽ xảy ra khi chúng ta đáp ứng điều kiện {on:}, có nghĩa là khi ta push commit mới lên branch main của git repo.



Sau khi 2 step trên thực thi xong thì ta có kết quả như sau trong docker hub:



1. Cài đặt EC2 Instance
2. Chạy docker container trên EC2

# Thiết kế và cài đặt môi trường

## 1. Cấu hình và cài đặt EC2 Instance:

**1.1.** **Mô tả:**

Amazon EC2 Instance là một cloud server được vận hành trên cơ sở hạ tầng AWS, với khả năng cung cấp nhiều tùy chọn cấu hình cho người dùng, đây là một lựa chọn phù hợp để triển khai đồ án với chi phí tốt nhất.

**1.2.** **Tiến hành cấu hình:**

**1.2.1.** **Amazon Machine Image (AMI):** Ubuntu Server 22.04 LTS. So với hệ điều hành Windows thì Ubuntu ít tiêu hao tài nguyên phần cứng hơn, giúp tăng hiệu suất hệ thống khi được vận hành trên cloud server.

**1.2.2.** **Instance type:** t2.micro. Đây là phiên bản tiết kiệm chi phí nhất (Free tier), kết hợp với hệ điều hành Ubuntu là vừa đủ để vận hành đồ án.

**1.2.3.** **Network setting:** Ta thực hiện chỉnh sửa cấu hình mạng như sau:

Mở cổng 22 với giao thức TCP loại SSH cho phép truy cập và kiểm soát server từ xa thông qua internet, hay nói cách khách khi mở cổng này em có thể kết nối trực tiếp vào cửa sổ terminal để vận hành server.

Mở cổng 3306 với giao thức TCP giúp kết nối vào cơ sở dữ liệu thông qua hệ quản trị cơ sở dữ liệu MySQL.

Mở tất cả cổng từ 0-65535 với giao thức TCP, vì đồ án yêu cầu sử dụng Docker tạo container để chạy trên nhiều cổng khác nhau nên việc mở toàn bộ cổng sẽ thuận tiện cho việc cài đặt sau này.

**1.3.** **Cài đặt chương trình:**

**1.3.1.** **Cài đặt Docker:**

Để cài đặt Docker, trước hết ta cần thiết lập Docker repository:

Cập nhật apt package index và cài đặt packages để cho phép apt sử dụng một repository thông qua HTTPS:

$ sudo apt-get update

$ sudo apt-get install \

ca-certificates \

curl \

gnupg \

lsb-release

Thêm offical GPG key cho Docker:

$ sudo mkdir -p /etc/apt/keyrings

$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | \

sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

Tiến hành thiết lập mặc định cho repository:

$ echo \

"deb [arch=$(dpkg --print-architecture) \

signed-by=/etc/apt/keyrings/docker.gpg] \

https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

Cập nhật lại apt package index:

$ sudo apt-get update

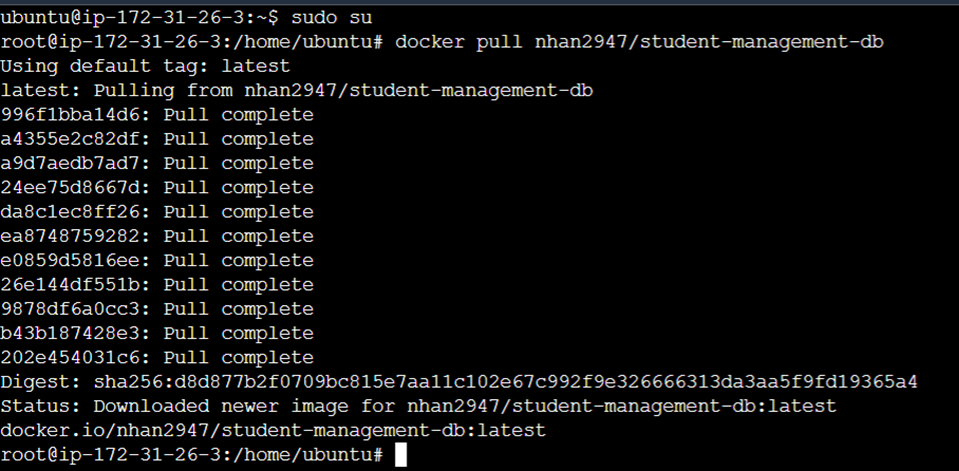
Cài đặt Docker Engine, containerd và Docker Compose:

$ sudo apt-get \

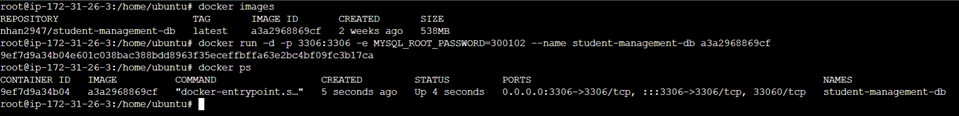
install docker-ce docker-ce-cli containerd.io docker-compose-plugin

**1.3.2.** **Cài đặt MySQL:**

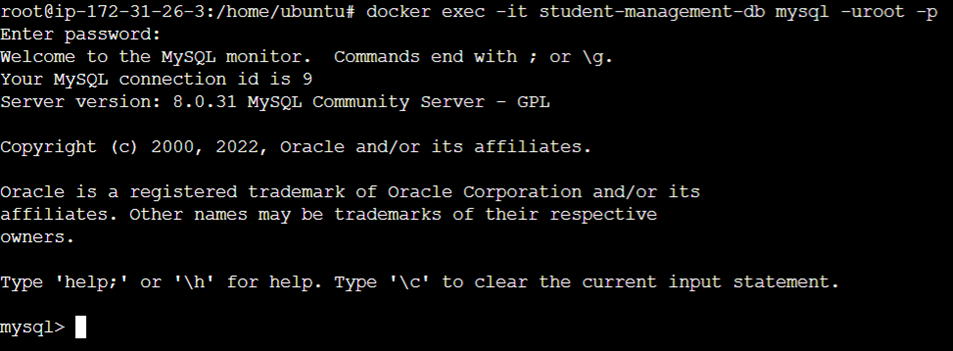
Cài đặt MySQL Image thông qua Docker Hub:



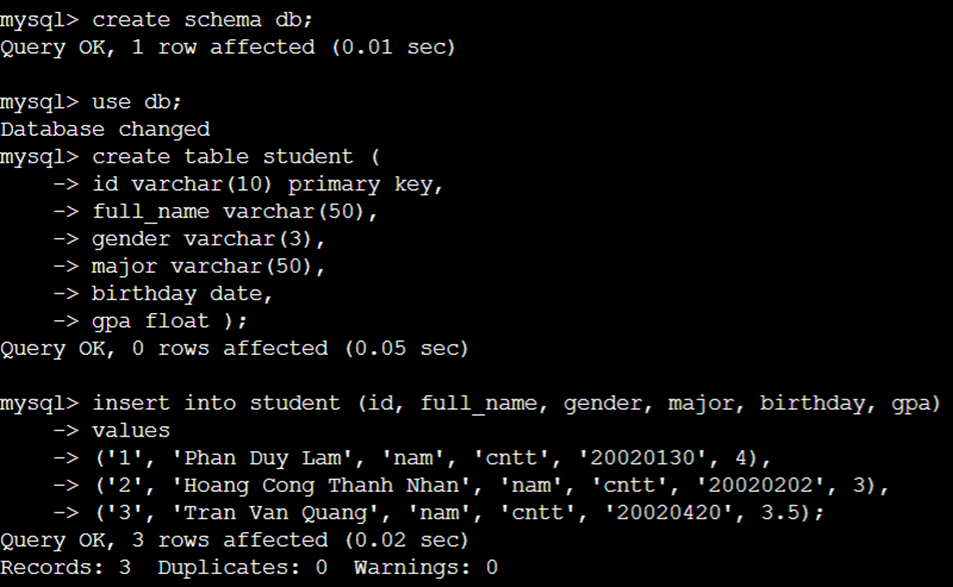
Chạy MySQL Image trên Container:



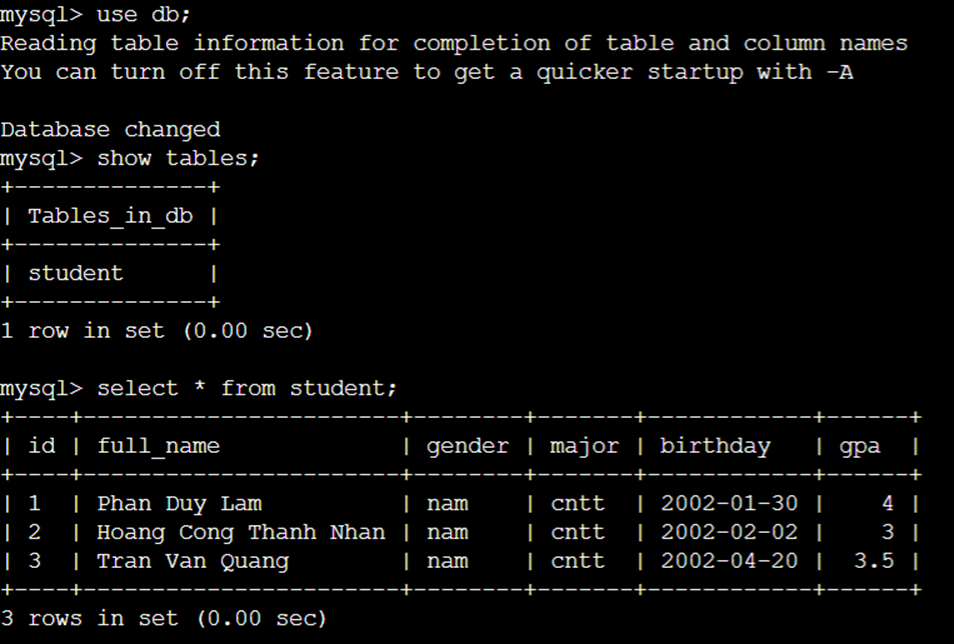
Kết nối vào MySQL:



Tạo bảng và nhập dữ liệu:



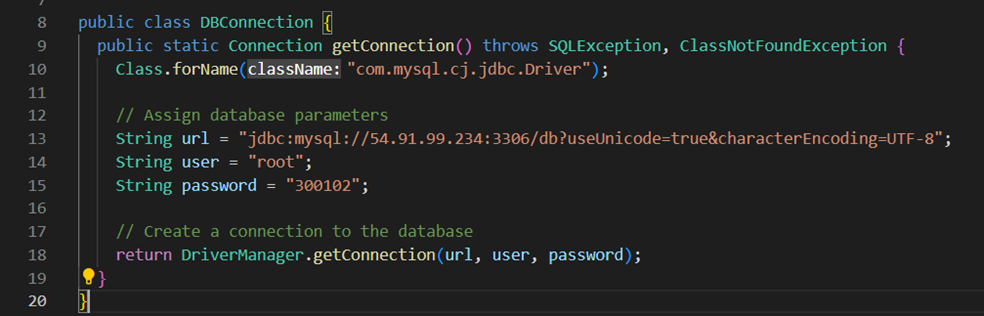
Kiểm tra:



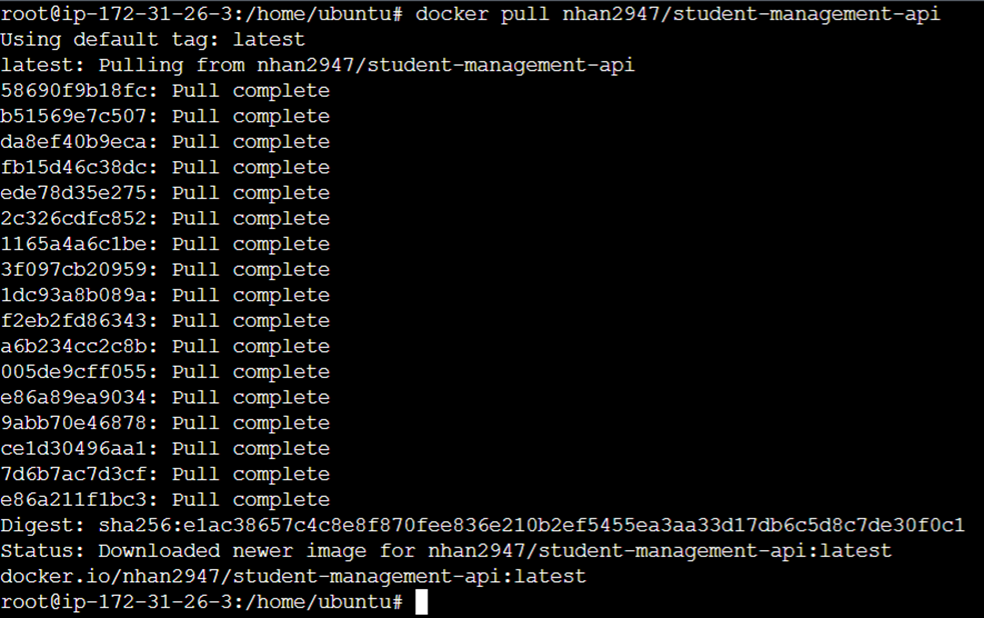
**1.3.3.** **Cài đặt Backend:**

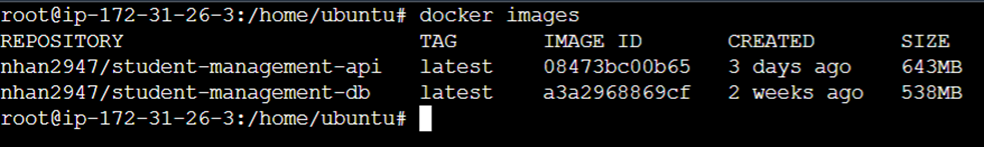
Cấu hình kết nối database:

Với 54.91.99.234 là địa chỉ IPv4 của Instance, 3306 là cổng để kết nối đến cơ sở dữ liệu, db là tên cơ sở dữ liệu.



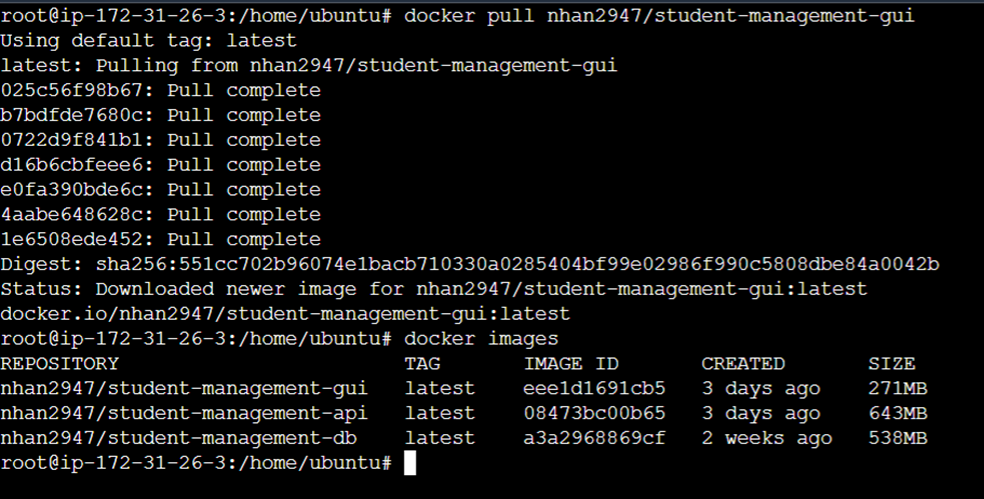
Cài đặt backend vào server thông qua Docker Hub:

****

****

**1.3.4.** **Cài đặt Frontend:**

Cài đặt frontend vào server thông qua Docker Hub:

****

# PHẦN KẾT LUẬN

Điện toán đám mây đang tạo ra một sự thay đổi cơ bản trong kiến trúc máy tính, phát triển phần mềm và các công cụ và tất nhiên cả trong cách chúng ta lưu trữ, phân phối và sử dụng thông tin. Như vậy có thể kết luận rằng điện toán đám mây chính là tập hợp các dịch vụ CNTT được cung cấp qua mạng Internet, mà ở đó người dùng sẽ được trao nhiều quyền chủ động hơn, chẳng hạn như thích gì dùng nấy, dùng bao nhiêu trả bấy nhiêu, được quyền lựa chọn những dịch vụ tốt nhất, có thể dùng bất cứ khi nào và bất cứ ở đâu.