**MÔ TẢ HỆ THỐNG**

1. **NodeJS:**

Node.js là một nền tảng chạy trên môi trường V8 JavaScript runtime - một trình thông dịch JavaScript cực nhanh chạy trên trình duyệt Chrome. Bình thường thì bạn cũng có thể tải bộ V8 và nhúng nó vào bất cứ thứ gì; Node.js làm điều đó đối với các web server. JavaScript suy cho cùng cũng chỉ là một ngôn ngữ - vậy thì không có lý do gì để nói nó không thể sử dụng trên môi trường server tốt như là trong trình duyệt của người dùng được.

1. **RXJava**:

RxJava cơ bản là một thư viện cung cấp các sự kiện không đồng bộ được phát triển theo Observer Pattern. Bạn có thể tạo luồng dữ liệu không đồng bộ trên bất kỳ thread nào, thay đổi dữ liệu và sử dụng dữ liệu bằng Observer. Thư viện RxJava cung cấp nhiều loại Operator tuyệt vời như map, combine, merge , filter và nhiều thứ khác có thể được áp dụng cho luồng dữ liệu.

1. **Okhttp**:

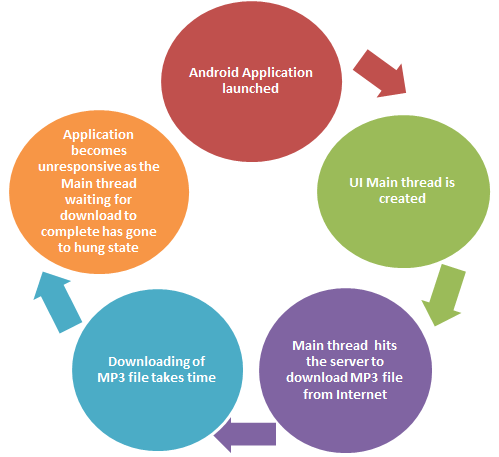
Bỏ qua định nghĩa khó hiểu từ nhà phát triển thì nó đơn giản là một thư viện cho phép kết nối internet sử dụng giao thức Http một cách dễ dàng và nhanh chóng hơn.

* Kiểm soát kết nối tới server
* Kiểm soát các kết nối không tốt và thử kết nối lại khi có thể
* Nó sẽ thử thay thế server IP address nếu kết nối tới một IP nào đó bị thất bại vào IP thay thế được chuẩn bị sẵn
* Giảm độ trễ của request, giảm size của file cần download
* Tránh lặp lại các request đã được hoàn thành

1. **Async task**

AsyncTask là một abstract Android class, giúp ứng dụng Android xử lý main UI thread hiệu quả hơn. AsyncTask trong Android cho phép chúng ta thực hiện những tác vụ dài mà không ảnh hưởng đến main thread

Sơ đồ trạng thái dưới đây cho thấy một loạt các hoạt động sẽ diễn ra khi bạn chạy ứng dụng



Trong khi chờ nhận file MP3 từ máy chủ, ứng dụng sẽ bị treo vì main thread vẫn đang chờ tác vụ tải xuống hoàn tất.

Để khắc phục điều này, chúng ta có thể tạo thread mới và thực hiện các tác vụ trên thread mới đó. Do đó giao diện người dùng sẽ không bị ảnh hưởng và treo nữa

Nhưng việc xử lý với thread riêng biệt có thể tạo ra một số vấn đề như việc cập nhật giao diện người dùng. Bạn sẽ cần phải cập nhập trạng thái download được bao nhiêu % file đó, và khi kết thúc tải thì cũng phải cập nhập cho người dùng biết. Nếu bạn sử dụng Thread đơn giản của java thì việc cập nhập này sẽ khá phức tạp.

1. **Thread**

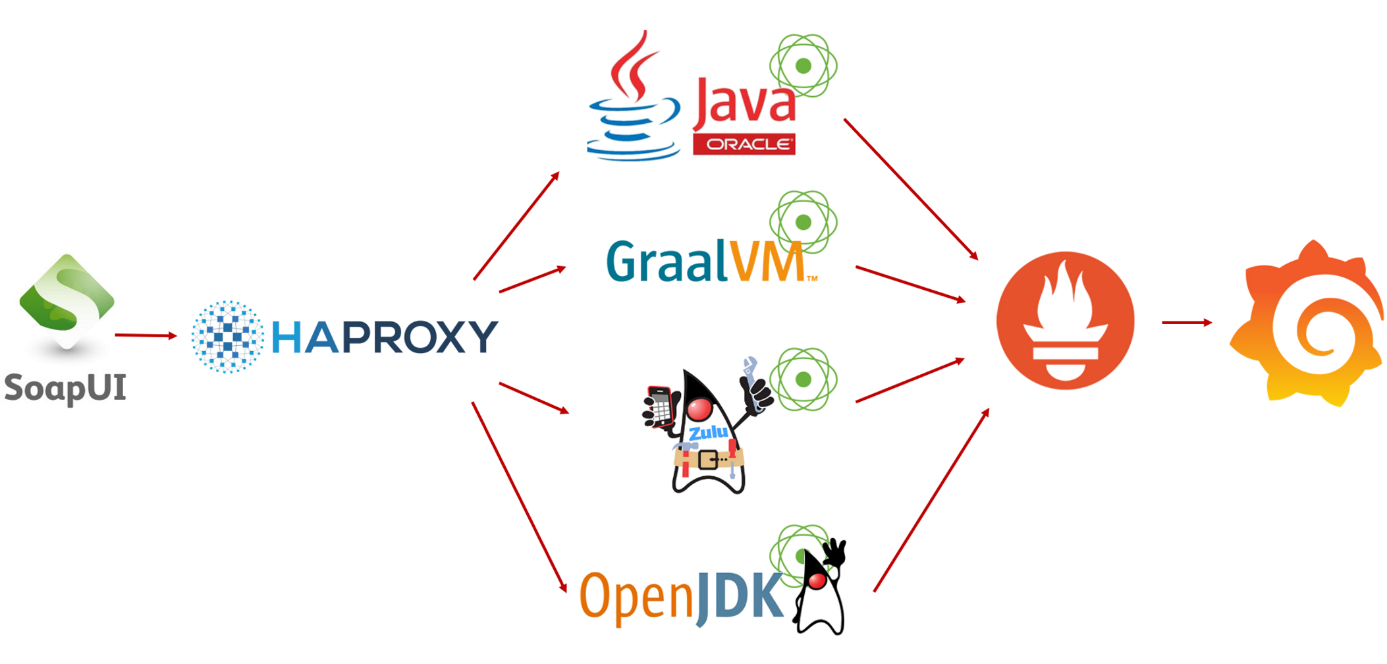
Một Thread là một chủ đề thực hiện trong một chương trình. Máy ảo Java cho phép một ứng dụng có nhiều luồng thực thi chạy đồng thời.

Mỗi chủ đề có một ưu tiên. Các chủ đề có mức độ ưu tiên cao hơn được thực hiện theo mức độ ưu tiên cho các chủ đề có mức độ ưu tiên thấp hơn. Mỗi luồng có thể hoặc không cũng được đánh dấu là một daemon. Khi mã chạy trong một số luồng tạo một Thread đối tượng mới , luồng mới có mức ưu tiên ban đầu được đặt bằng mức độ ưu tiên của luồng tạo và là một luồng daemon khi và chỉ khi luồng tạo là một daemon.

Khi Máy ảo Java khởi động, thường có một luồng không phải trình nền (thường gọi phương thức được đặt tên main của một số lớp được chỉ định). Máy ảo Java tiếp tục thực thi các luồng cho đến khi một trong hai điều sau xảy ra:

* Các exit phương pháp của lớp Runtime đã được gọi và quản lý bảo mật đã cho phép các hoạt động xuất cảnh sẽ diễn ra.
* Tất cả các luồng không phải là luồng daemon đã chết, bằng cách quay lại từ lệnh gọi đến Run phương thức hoặc bằng cách ném một ngoại lệ lan truyền ngoài Run phương thức.

1. **AdoptJDK**

* **AdoptJDK là gì?**
  + AdoptJDK là một phiên bản của OpenJDK, AdoptJDK được phát triển bởi nhóm cộng đồng Java User Group (JUG) bao gồm Azul, Amazon, GoDaddy, IBM, jClarity (acquired by Microsoft), Microsoft, New Relic, Pivotal and Red Hat
  + AdoptJDK hoàn toàn miễn phí (lưu ý nhiều người hiểu nhầm cứ mã nguồn mở là miễn phí, không phải nguồn mở nào cũng miễn phí, và còn có nhiều yếu tố liên quan tới giấy phép mã nguồn mở)
  + Có thể chạy trên nhiều nền tảng hệ điều hành khác nhau: Windows, Linux, Mac, Unix,… Tạo nên sự vô cùng linh động cho nhà phát triển, dễ dàng lựa chọn nền tảng để triển khai
  + AdoptJDK support 2 loại máy ảo là  [OpenJDK HotSpot](https://openjdk.java.net/groups/hotspot/" \t "_blank) or [Eclipse OpenJ9](https://www.eclipse.org/openj9/) VM

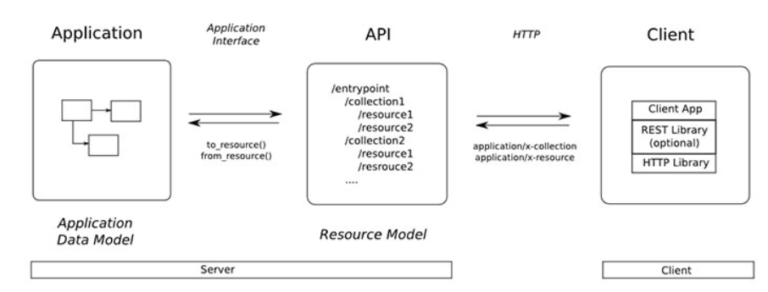
.

Hình 1: Các phiên bản JDK

* **AdoptJDK làm được những gì?**
* OpenJDK làm được gì thì AdoptJDK làm được như thế .
* Hỗ trợ 2 loại máy ảo khác nhau, mỗi loại máy ảo sẽ có chức năng khác nhau
* AdoptJDK có thể truy vấn, thêm, sửa, xóa dữ liệu trong nhiều hệ Cơ sở dữ liệu như: MySql, MongoDB, Microsoft SQL Server, PostgreSQL.
* AdoptJDK không bị ràng buộc bởi Oracle, chủ sở hữu hiện tại của Oracle JDK, công ty này đã từng kiện Google rất nhiều lần do liên quan đến các vấn đề “Giấy Phép”
* Và nhiều khả năng khác nữa, xem thêm trên . <https://adoptopenjdk.net/index.html>

1. **Thư viện Json Project**: Patch các chuỗi Json thành các Modle
2. **Rest API**

* **REST API là gì?**

****

Hình 2: Mô hình REST API

* + **REST** là viết tắt của **Representational State Transfer**. Giải thích đơn giản, REST là một loạt hướng dẫn và dạng cấu trúc dùng cho việc chuyển đổi dữ liệu. Thông thường, REST hay được dùng cho ứng dụng web, nhưng cũng có thể làm việc được với dữ liệu phần mềm.
  + **API** là viết tắt của **Application Programming Interface**, phương thức kết nối với các thư viện và ứng dụng khác. Windows có nhiều API, và Twitter cũng có web API, tuy chúng thực hiện các chức năng khác nhau, với mục tiêu khác nhau.
  + Nhìn chung, REST API là những API đi theo cấu trúc REST.
* **REST hoạt động như thế nào?**
  + REST hoạt động dựa trên giao thức HTTP. Như hình trên chúng ta có thể thấy được trình tự hoạt động của nó khi Client request đến Server.
    - **GET (SELECT)**: trả về một resource hoặc danh sách resource
    - **POST (CREATE)**: tạo mới một resource
    - **PUT (UPDATE)**: cập nhật thông tin cho resource
    - **DELETE**: xóa một resource
* **Các ưu điểm của REST**
  + Giúp cho ứng dụng trở nên rõ ràng hơn.
  + REST URL đại diện cho resource chứ không phải là hành động.
  + Dữ liệu được trả về với nhiều định dạng khác nhau như: xml, html, rss, json …
  + Code đơn giản và ngắn gọn.
  + REST chú trọng vào tài nguyên hệ thống.
  1. **AWS**

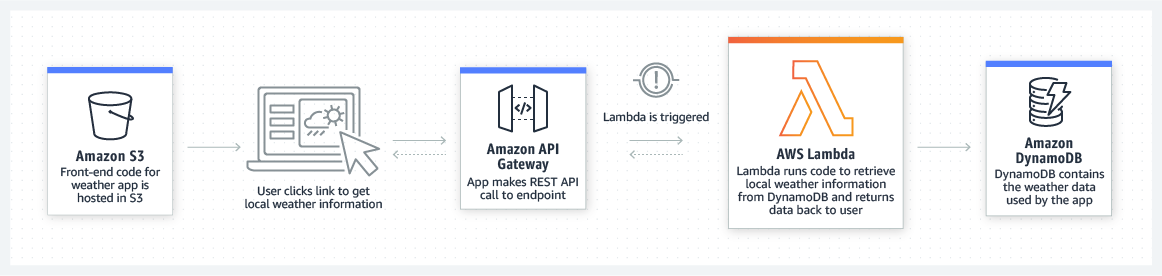
Vào năm 2006, Amazon Web Services (AWS) bắt đầu cung cấp các dịch vụ cơ sở hạ tầng CNTT cho các doanh nghiệp dưới hình thức dịch vụ web -- nay thường được gọi là điện toán đám mây. Một trong những lợi ích chính của điện toán đám mây là cơ hội thay thế chi phí cơ sở hạ tầng trả trước bằng các khoản phí không cố định ở mức thấp có khả năng thay đổi quy mô theo hoạt động kinh doanh của bạn. Với Đám mây, doanh nghiệp không còn cần phải lên kế hoạch và mua máy chủ cũng như các cơ sở hạ tầng CNTT khác trước nhiều tuần hay nhiều tháng. Thay vào đó, họ có thể ngay lập tức đăng ký hàng trăm hoặc hàng nghìn máy chủ chỉ trong vài phút và mang lại kết quả nhanh hơn.

Ngày nay, Amazon Web Services cung cấp nền tảng cơ sở hạ tầng có chi phí thấp, khả năng thay đổi quy mô linh hoạt và độ tin cậy cao trên nền tảng đám mây cung cấp dịch vụ cho hàng trăm nghìn doanh nghiệp tại 190 quốc gia trên khắp thế giới. Với các vị trí đặt trung tâm dữ liệu ở Hoa Kỳ, châu Âu, Brazil, Singapore, Nhật Bản và Úc

* + 1. **Lambda**

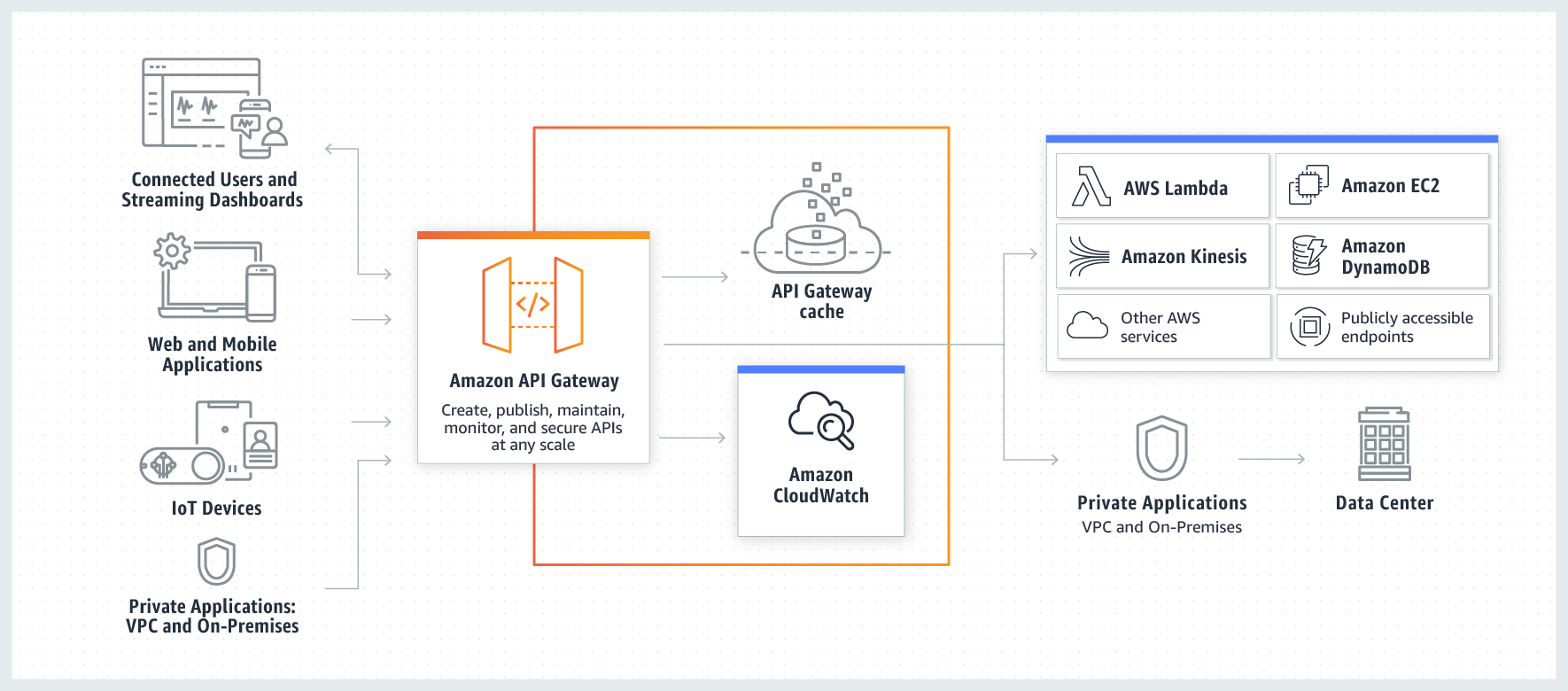
Trong ứng dụng Quản Lý Khách Sạn, em có sử dụng Lambda để làm ứng dụng web, nó giúp tối ưu chi phí viết API, không cần quan tâm về hosting

* AWS Lambda là một dịch vụ tính toán nơi mà bạn có thể upload code của mình lên, và dịch vụ AWS Lambda sẽ giúp bạn chạy đoạn code đó bằng việc sử dụng các tài nguyên sẵn có của AWS. Sau khi bạn upload code, và bạn tạo ra một Lambda function, AWS sẽ cung cấp và quản lý các server mà bạn sử dụng để chạy code. Về phần sử dụng AWS, bạn có thể làm như sau:
  + một dịch vụ tính toán hướng sự kiện nơi mà AWS Lambda chạy code của bạn và trả về các sự kiện, những thay đổi về data được đẩy vào một bucket của Amazon S3 hoặc là một bảng của Amazon DynamoDB. ( bạn có thể tìm hiểu thêm về Amazon DynamoDB)
  + Dịch vụ tính toán để chạy code và sẽ trả về các HTTP request sử dụng Amazon API Gateway hoặc API dùng đến AWS SDKs.
* AWS Lambda sẽ chạy code mà bạn upload trên môi trường có tính sẵn sàng cao và vận hành với quyền cao nhất đối với các tài nguyên tính toán, bao gồm server và hệ điều hành. Những gì bạn cần làm là chắc chắn code thuộc một trong những ngôn ngữ mà AWS Lambda đang hỗ trợ: Node.js, Java, Python.
* Để vận hành code được tốt nhất, bạn có thể tham khảo cách xác định và tính toán tài nguyên cần dùng cho AWS Lambda tại Xác định tài nguyên cho AWS Lambda
* AWS Lambda chỉ thực hiện code của bạn chỉ khi cần thiết, và được mở rộng một cách tự động, từ một vài request đến hàng ngàn request/ giây. Với khả năng đó, bạn có thể sử dụng Lambda dễ dàng để build dữ liệu cho các dịch vụ AWS S3 và Amazon DynamoDB, luồng xử lý dữ liệu trong Amazon Kinesis hoặc tạo back end riêng mà AWS thực hiện có quy mô, năng suất và bảo mật.
* Hoạt động:



Hình 4:Mô hình Lambda làm việc với ứng dụng web

* + 1. **API Gateway**
* Amazon API Gateway là dịch vụ được quản lý hoàn toàn giúp các nhà phát triển dễ dàng tạo, phát hành, duy trì, giám sát và bảo vệ API ở mọi quy mô. API đóng vai trò là "cửa trước" cho các ứng dụng để truy cập dữ liệu, logic nghiệp vụ hoặc chức năng từ các dịch vụ backend của bạn. Bằng cách sử dụng API Gateway, bạn có thể tạo các API RESTful và API WebSocket để kích hoạt các ứng dụng giao tiếp hai chiều theo thời gian thực. API Gateway hỗ trợ các khối lượng công việc có trong container và serverless, cũng như các ứng dụng web.
* API Gateway xử lý tất cả các tác vụ liên quan đến tiếp nhận và xử lý lên đến hàng trăm nghìn lệnh gọi API đồng thời, bao gồm quản lý lưu lượng truy cập, hỗ trợ CORS, xác thực và kiểm soát truy cập, điều tiết, giám sát và quản lý phiên bản API. API Gateway không yêu cầu phí tối thiểu hoặc phí ban đầu. Bạn trả tiền cho các lệnh gọi API bạn nhận được cũng như lượng dữ liệu được truyền đi và, với mô hình định giá theo bậc của API Gateway, bạn có thể giảm chi phí khi thay đổi quy mô sử dụng API.



Hình 4: Mô hình hoạt động của API Gateway

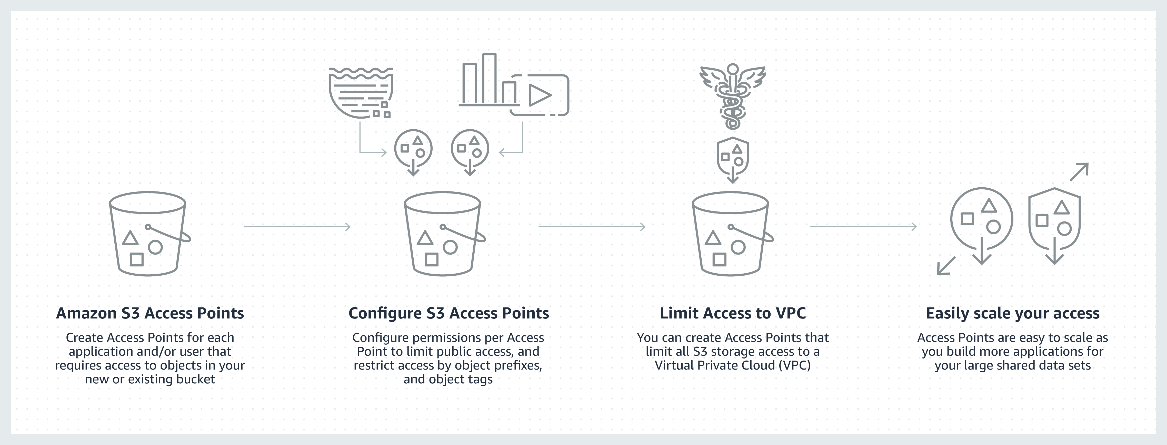
* + 1. **Serverless - Serverless Java container**

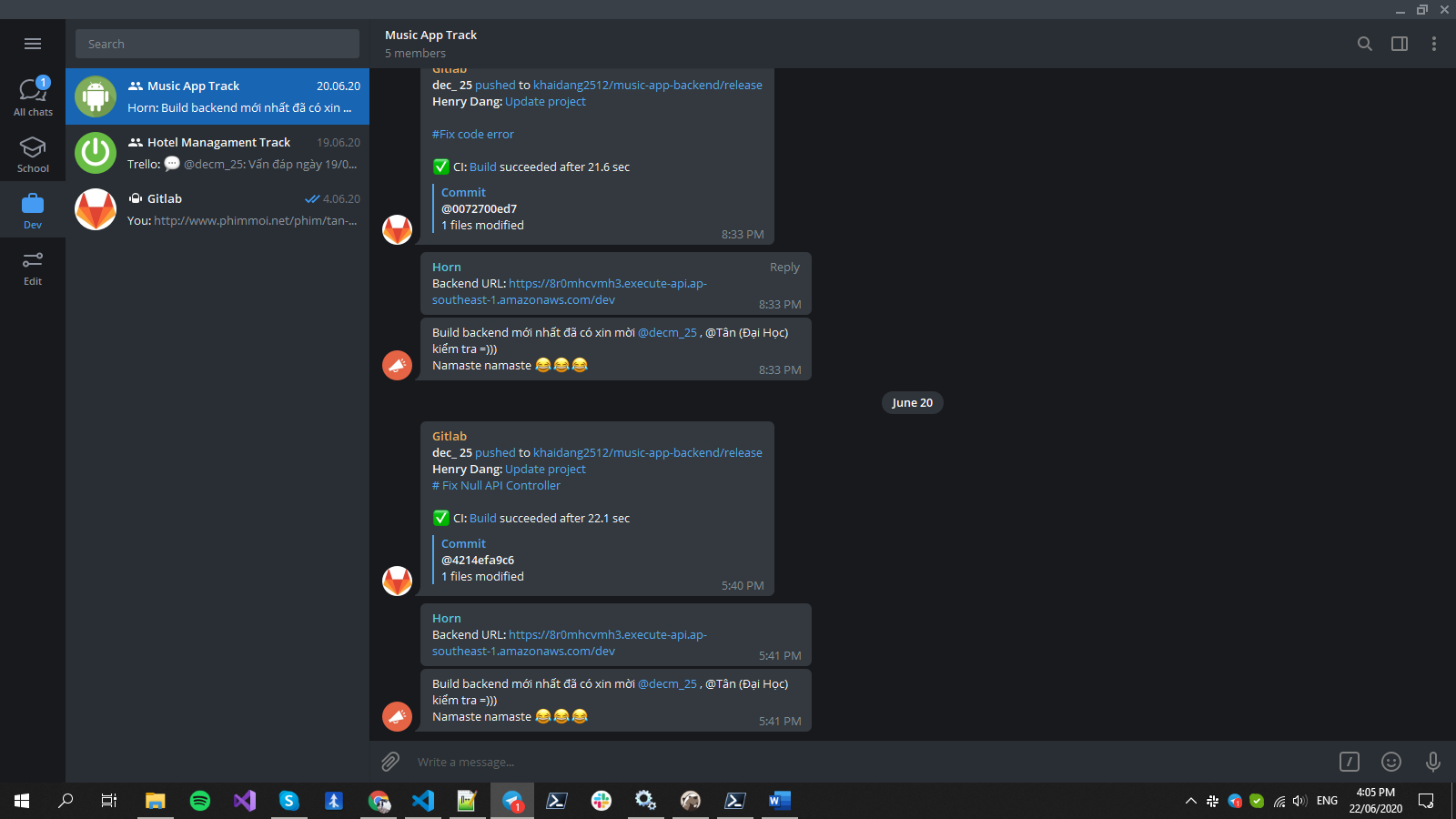
**Serverless là gì?**

* Serverless là môi trường, nền tảng thực thi ứng dụng và dịch vụ mà không phải quan tâm đến máy chủ. Ứng dụng serverless không cần phải quan tâm việc phân bổ, quản lý tài nguyên của hệ điều hành, và bỏ qua các vấn đề về nâng cấp và bảo mật. Với khái niệm là chỉ cần tập trung phát triển sản phẩm, việc còn lại về vận hành sẽ để nền tảng này đảm nhiệm.
* Điều quan trọng và khác biệt nhất trong serverless là bạn chỉ trả tiền khi và chỉ những phần bạn sử dụng. Giả sử bạn có một máy chủ ảo, thì thường sẽ được tính tiền trọn gói bao gồm thời gian chạy 24/7 trong 1 tháng, CPU và RAM, băng thông, lưu trữ. Bạn vẫn sẽ phải trả tiền hàng tháng đều đặn cho dù cái máy chủ ảo đó không chạy, hoặc chỉ sử dụng 5~10% công suất thì bạn vẫn phải trả trọn gói. Hiểu một cách nôm na, thì serverless như gói cước điện thoại được tính theo block giây, gọi bao nhiêu tính tiền bấy nhiêu, còn máy chủ ảo thường thì phải trả tiền thuê bao hàng tháng dù có phải sử dụng hay không.

**Serverless Java container?**

* Là một framework serverless hỗ trợ Java, nó hoạt động như là một handler chính của Lambda, parse request từ api gateway sang request của sring, spring boot hay struts framework.
* Mô hình hoạt động như một container hay docker
* Link github <https://github.com/awslabs/aws-serverless-java-container>
  + 1. **EC2**
* Amazon Elastic Compute Cloud (Amazon EC2) là dịch vụ web cung cấp năng lực điện toán bảo mật và có kích cỡ linh hoạt trên đám mây. Dịch vụ này được thiết kế để giúp các nhà phát triển dễ sử dụng điện toán đám mây ở quy mô web hơn. Giao diện dịch vụ web đơn giản của Amazon EC2 cho phép bạn có được và cấu hình năng lực với sự va chạm tối thiểu. Dịch vụ này cung cấp cho bạn khả năng kiểm soát toàn phần các tài nguyên điện toán của bạn và giúp bạn chạy trên môi trường điện toán đã được kiểm chứng của Amazon.
* Trong dự án em có sữ dụng EC2 như một con Gitlab Runner để chạy CI/CD build code tự đong deploy lên lambda và s3
  + 1. **S3**
* Amazon Simple Storage Service (Amazon S3) là một dịch vụ lưu trữ đối tượng cung cấp khả năng thay đổi theo quy mô, tính khả dụng của dữ liệu, bảo mật và hiệu năng hàng đầu trong lĩnh vực. Điều này có nghĩa là khách hàng thuộc mọi quy mô và lĩnh vực có thể sử dụng dịch vụ này để lưu trữ và bảo vệ bất kỳ lượng dữ liệu nào cho nhiều trường hợp sử dụng khác nhau, chẳng hạn như trang web, ứng dụng di động, sao lưu và khôi phục, lưu trữ, ứng dụng doanh nghiệp, thiết bị IoT và phân tích dữ liệu hớn. Amazon S3 cung cấp các tính năng quản lý dễ sử dụng, nhờ đó, bạn có thể tổ chức dữ liệu và cấu hình các kiểm soát truy cập được tinh chỉnh để đáp ứng yêu cầu cụ thể của doanh nghiệp, tổ chức và yêu cầu về tuân thủ. Amazon S3 được thiết kế để đảm bảo độ bền 99,999999999% (11 9's) và lưu trữ dữ liệu của hàng triệu ứng dụng cho các công ty trên toàn thế giới.
* Ở trong dự án quản lý khách sạn phần frontend em dử sụng angular và em host trên S3
* Mô hình:



* + 1. **AWS SES**
* Amazon Simple Email Service (Amazon SES) là một dịch vụ có quy mô cực kỳ linh hoạt và tiết kiệm chi phí dùng để gửi và nhận email. Amazon SES loại bỏ sự phức tạp cũng như chi phí của việc xây dựng một giải pháp email nội bộ hoặc việc cấp phép, cài đặt và vận hành một giải pháp email của bên thứ ba.
* Amazon SES là giải pháp tuyệt vời cho bất kỳ ai cần một phương thức có độ tin cậy, khả năng thay đổi quy mô và tiết kiệm chi phí để gửi và nhận email. Người dùng của chúng tôi bao gồm nhiều loại hình tổ chức, chẳng hạn như nhà bán lẻ trực tuyến, nhà phát triển ứng dụng và tổ chức tiếp thị kỹ thuật số.
* Trong dự án quản lý khách sạn em sử dụng dịch vụ SES như là một mail provider
  1. **Telegram - Integram**
     1. **Telegram là gì?**
* Telegram là ứng dụng nhắn tin tập trung vào tốc độ và bảo mật, nó siêu nhanh, đơn giản và miễn phí. Bạn có thể sử dụng Telegram cùng lúc trên nhiều thiết bị, tin nhắn được đồng bộ đồng thời trên tất cả các thiết bị: điện thoại, máy tính bảng hay máy tính.
* Với Telegram, bạn có thể gửi tin nhắn, hình ảnh, video, file (bất kỳ loại nào từ doc, zip đến mp3...) cũng như tạo group lên tới 200.000 người hoặc tạo kênh để phát sóng nội dung đến số lượng khán giả không bị giới hạn. Bạn có thể gửi tin nhắn cho các số điện thoại có trong danh bạ, tìm người dùng theo username của họ. Telegram giống như sự kết hợp giữa SMS và email, có thể giải quyết tất cả những nhu cầu nhắn tin cá nhân hoặc công việc của bạn. Ngoài tất cả những tính năng trên, Telegram còn hỗ trợ các cuộc gọi thoại được mã hóa đầu cuối.
  + 1. **Integram là gì**
* Integram là một bot service giúp bạn boost up teamwork collaboration
* Support nhiều bot cho github, gitlab, trello, bit bucket, hoặc là horn mộ service tự custom payload message để gửi vào group hay thông báo.
* Trong dự án quản lý khách sạn em dùng Integram để làm việc với Gitlab, Trello để thông báo số lần commit, CI/CD build passed hay failed, assign task trên trello, gửi tin nhắn vào group để mọi người trong team có thể biết
* Tăng tốc độ làm việc của team lên rất nhiều lần

Hình 5: Ứng dụng của **Integram**

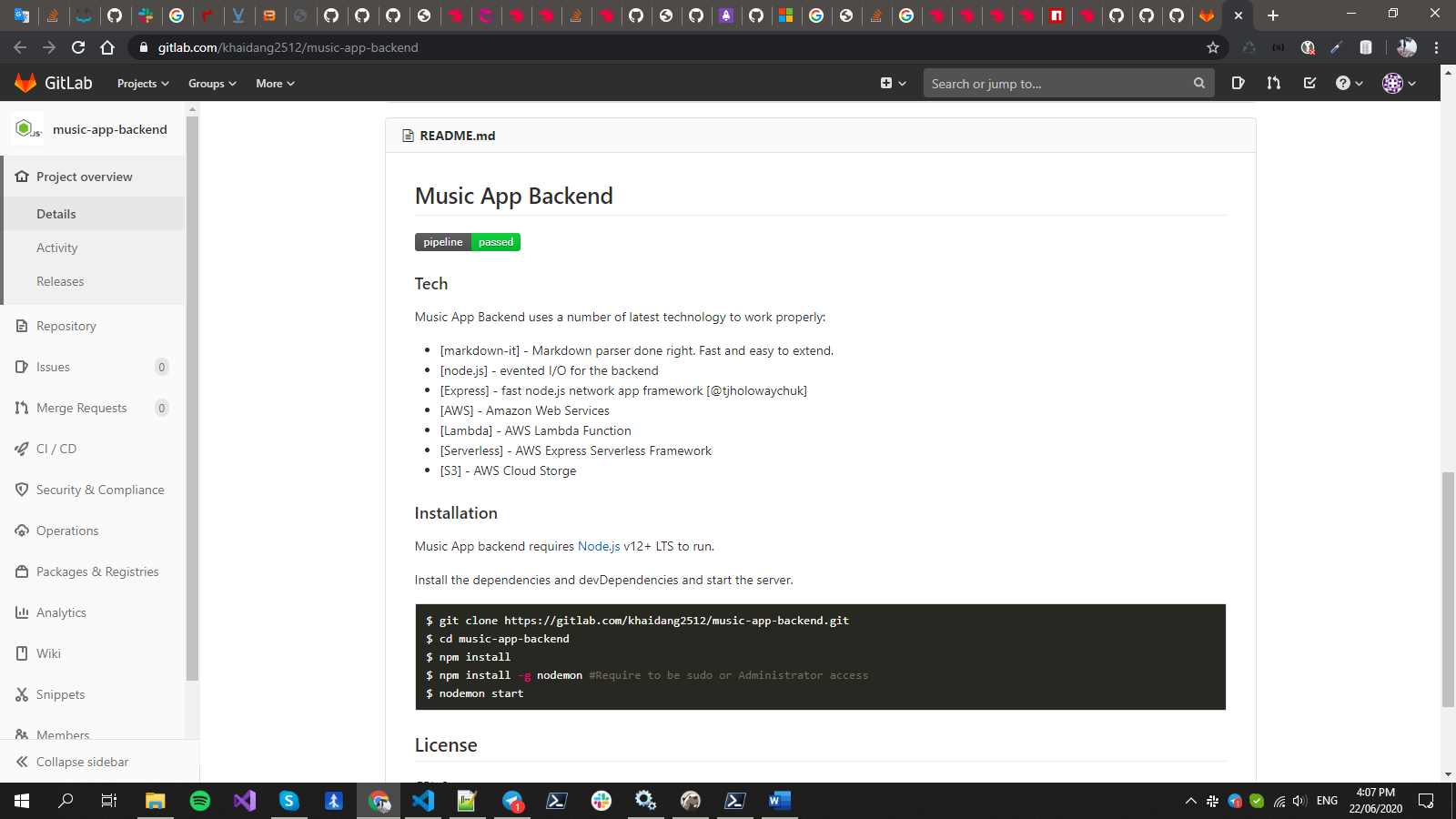
1. **Git – Gitlab**

* **Git là gì?**
  + Git là tên gọi của một Hệ thống quản lý phiên bản phân tán (Distributed Version Control System – DVCS) là một trong những hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay. DVCS nghĩa là hệ thống giúp mỗi máy tính có thể lưu trữ nhiều phiên bản khác nhau của một mã nguồn được nhân bản (clone) từ một kho chứa mã nguồn (repository), mỗi thay đổi vào mã nguồn trên máy tính sẽ có thể ủy thác (commit) rồi đưa lên máy chủ nơi đặt kho chứa chính. Và một máy tính khác (nếu họ có quyền truy cập) cũng có thể clone lại mã nguồn từ kho chứa hoặc clone lại một tập hợp các thay đổi mới nhất trên máy tính kia. Trong Git, thư mục làm việc trên máy tính gọi là Working Tree. Đại loại là như vậy.

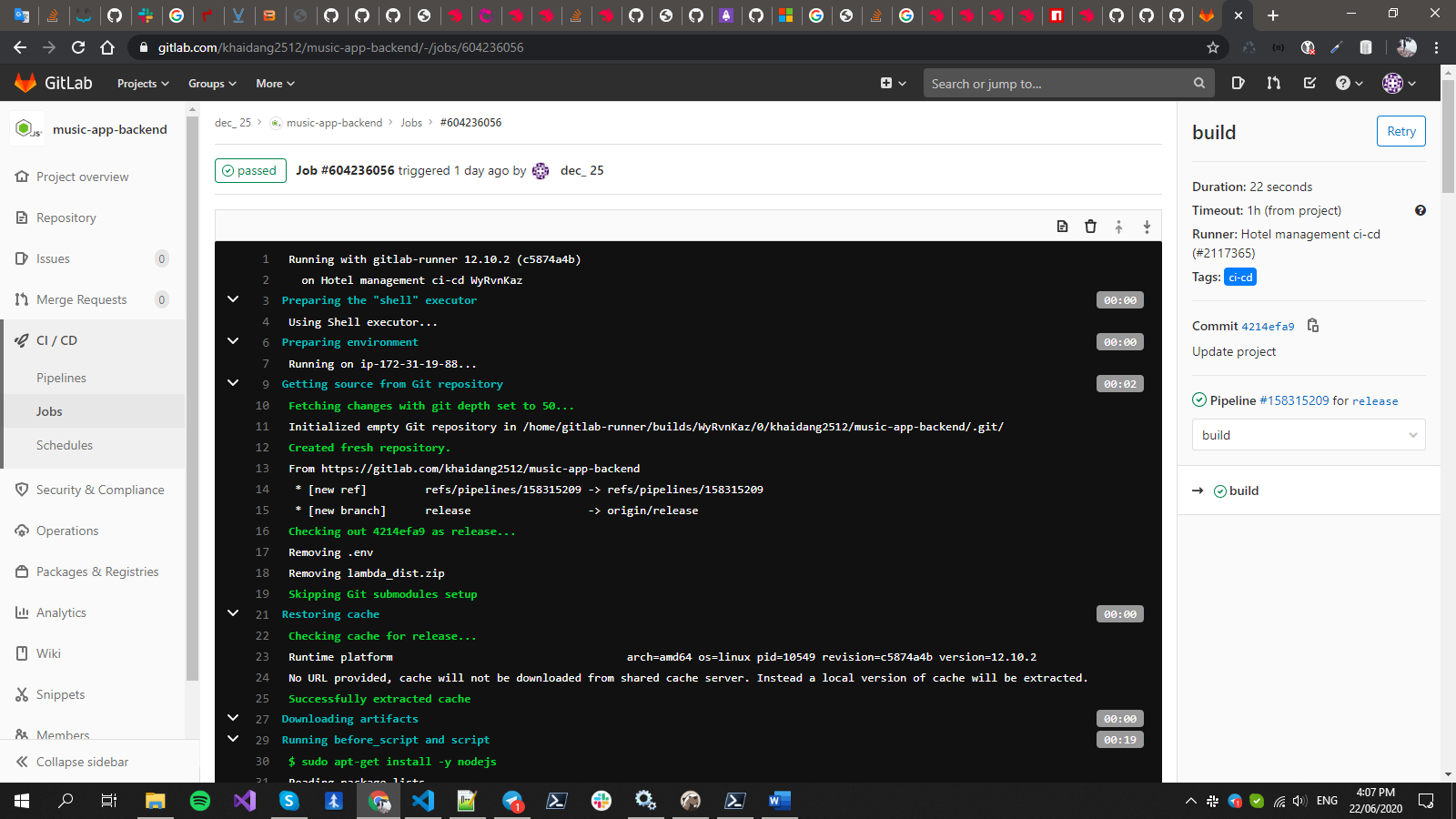
****

Hình 6: Mô hình của Git

* Ngoài ra, có một cách hiểu khác về Git đơn giản hơn đó là nó sẽ giúp bạn lưu lại các phiên bản của những lần thay đổi vào mã nguồn và có thể dễ dàng khôi phục lại dễ dàng mà không cần copy lại mã nguồn rồi cất vào đâu đó. Và một người khác có thể xem các thay đổi của bạn ở từng phiên bản, họ cũng có thể đối chiếu các thay đổi của bạn rồi gộp phiên bản của bạn vào phiên bản của họ. Cuối cùng là tất cả có thể đưa các thay đổi vào mã nguồn của mình lên một kho chứa mã nguồn.
* Cơ chế lưu trữ phiên bản của Git là nó sẽ tạo ra một “ảnh chụp” (snapshot) trên mỗi tập tin và thư mục sau khi commit, từ đó nó có thể cho phép bạn tái sử dụng lại một ảnh chụp nào đó mà bạn có thể hiểu đó là một phiên bản. Đây cũng chính là lợi thế của Git so với các DVCS khác khi nó không “lưu cứng” dữ liệu mà sẽ lưu với dạng snapshot.
* **Gitlab là gì?**
  + GitLab khá nổi tiếng và là một mã nguồn mở của máy chủ Git được thực hiện bởi hơn 50.000 tổ chức. Trong vài năm gần đây Gitlad đã phát triển mạnh mẽ với sự hỗ trợ của cộng đồng mạng, hàng nghìn người sử dụng trên một máy chủ duy nhất hoặc một số máy chủ hoạt động tương tự. Nếu bạn cần thiết lập một máy chủ Git, thì GitLab cung cấp cho bạn một giải pháp hoàn hảo.
  + Gitlad là một hệ thống self-hosted để quản lý mã nguồn của bạn. Bản đầu tiên được phát hành vào tháng 10/2011 và được cập nhật vào ngày 22 hàng tháng. Gitlab được phát hành theo tiêu chuẩn của MIT.
  + Gitlab được thành lập bởi Dmitriy Zaporozhets năm 2013. Dự án bao gồm hai nhóm chính: một bên là “open source core team” và một bên là “GitLab B.V. team” (chi nhánh của công ty Gitlab).
  + GitLab được sử dụng để lư trữ trên Github, nhưng với sự nỗ lực của Dmitriy Zaporozhets suốt một năm làm việc tại GitLab, kể từ tháng 1/2014 mã nguồn được lưu trữ trên sever chính của gitlab là gitlab.com. Các nhánh của GitLab được lưu trữ trên github, sẽ hoạt động như một source, nơi bạn có thể pull, push và merge các yêu cầu.
  + Trong ứng dụng quản lý khách sạn tui em sử dụng Gitlab là máy chủ git vì gitlab support nhiều thứ github không hỗ trợ như build custom một cái máy ảo để build CI/CD



Hình 7: Ứng dụng của Gitlab



Hình 8: CI/CD Build

* **Tại sao nên sử dụng Git?**
  + Có rất nhiều lợi thế để bạn nên sử dụng Git trong việc lập trình ngay từ hôm nay, bất kể là lập trình cái gì đi chăng nữa.
  + Git dễ sử dụng, an toàn và nhanh chóng.
  + Có thể giúp quy trình làm việc code theo nhóm đơn giản hơn rất nhiều bằng việc kết hợp các phân nhánh (branch).
  + Bạn có thể làm việc ở bất cứ đâu vì chỉ cần clone mã nguồn từ kho chứa hoặc clone một phiên bản thay đổi nào đó từ kho chứa, hoặc một nhánh nào đó từ kho chứa.
  + Dễ dàng trong việc deployment sản phẩm.
  + Và nhiều hơn thế nữa.
  + Nếu bạn là một lập trình viên thì Git là một hệ thống bạn cần phải biết cách sử dụng, ít nhất là ngay từ bây giờ.
  + GitHub là một nền tảng tuyệt vời đã thay đổi cách hoạt động của lập trình viên. Tuy nhiên, bất kỳ ai muốn quản lý dự án hữu hiệu cũng có thể tham gia vào GitHub và hợp tác làm chung dự án để đạt hiệu quả cao nhất.
  + Nếu đội ngũ của bạn làm trong một dự án cần cập nhật liên tục và cần giám sát toàn bộ các thay đổi, GitHub là dành cho bạn. Những giải pháp tương tự GitHub khác là GitLab, BitBucket, nhưng chúng tôi vẫn khuyên bạn dùng GitHub.
  + Một đội lập trình viên trong cùng một dự án khi làm việc với nhau sẽ không tránh khỏi sai sót như code bị conflict với nhau, người này đã phát triển tính năng này nhưng người kia không biết, git ra đời để giải quyết các vấn đề đó, giúp kiểm soát mã nguồn tốt hơn cho các lập trinh viên.
  + Giúp code trong dự án chia cấu trúc rõ ràng hơn, nếu như có đội khác tiếp quản dự án cũ có thể dựa vào sơ đồ cảu git thống kê lại các tính năng đã có và phát triển thêm tính năng mới.