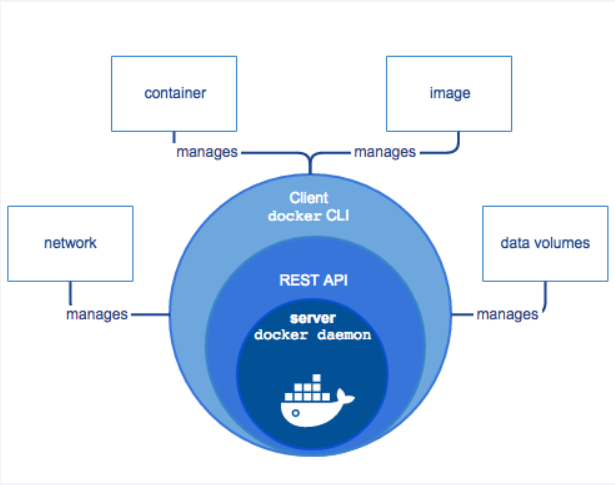
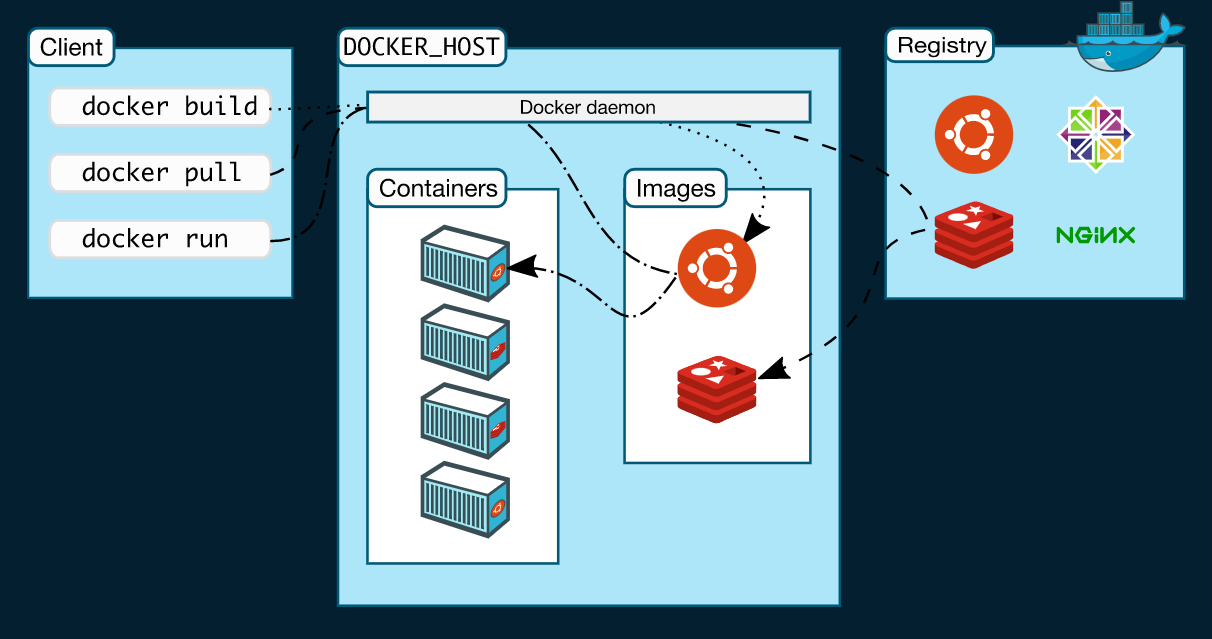
Báo cáo công việc tuần 1 ( 24/10/2022 – 30/10/2022)

Docker là gì ?

* Để một source code được khởi động và hoạt động trơn tru ở một máy tính đòi hỏi máy tính đó phải có đầy đủ tài nguyên như môi trường , các biến , một số thư viện , cũng như một số công cụ để hỗ trợ cho ngôn ngữ và công nghệ đó hoạt động. Với Docker bạn có thể quản lý các tài nguyên của source code một cách dễ dàng và giảm độ trễ giữ việc viết code với việc chạy chúng trong sản phẩm của bản thân.
  + VD: Nếu một máy tính đang chạy một phiên bản môi trường ví dụ như NodeJs ở phiên bản 16 nhưng ở một máy của người đồng nghiệp lại chạy NodeJs ở phiên bản cao hơn hoặc thấp hơn điều này sẽ dẫn đến việc là khi source code chạy ở máy mình hoạt động bình thường nhưng khi sang máy của bạn hoặc đồng nghiệp lại không được vì không hỗ trợ cách viết JS đó hoặc thiếu gì đó và dẫn đến mấy thời gian rất nhiều
* Chính vì thế thì Docker( Container ) được sinh ra để giải quyết những vấn đề này
  + Docker là một công cụ để cung cấp cho việc Build , Deploy, và Run source code ứng dụng hay là một trang web nào đó một cách dễ dàng. Và docker ban đầu được viết bằng ngôn ngữ Python và sau này là GoLang
  + Để thực hiện các việc như Build , Deploy , Run như thế thì Docker áp dụng 1 thuật ngữ là Containers . Containers cho phép người lập trình viên đóng gói source code với tất cả các điều kiện mà source code cần để chạy được như là thư viện , môi trường , hay là công cụ hỗ trợ.
  + Dựa vào Container sau khi ta khai báo và đóng gói các package cần thiết thì ta có thể đưa nó vào máy chủ chạy bất kì hệ điều hành nào cũng được và phổ biến nhất ở đấy là Linux, Ubuntu , Centos,…. Và có thể là Windows của Microsoft tùy thuộc vào máy chủ mà người tập trình viên đó thao tác với nó
  + Sự độc lập và bảo mật cho phép bạn chạy riêng biệt nhiều Container cùng một máy chủ. Các container hoạt động chiếm một lượng dung lượng của máy chủ tùy theo độ lớn nhỏ của dự án mà ta đưa vào nó.
  + Docker cung cấp các công cụ và nền tảng để quản lý quá trình hoạt động và vòng đời container của bạn.
    - Phát triển ứng dụng của bạn và các thành phần hỗ trợ trong contrainers
    - Container cũng đóng vai trò là đơn vị cung cấp và thử nghiệm ứng dụng hay trang web của bạn
    - Nếu bạn có nhu cầu đưa ứng dụng hay trạng web của mình lên môi trường sản phẩm và đưa ra cho mọi người sử dụng dưới dạng là 1 container hoặc là một dịch vụ nào đó cho dù môi trường đó là nằm ở máy của bạn hay là nằm trên một máy chủ khác ở điện toán đám mây hoặc có thể là sử dụng cả hai.
* Docker Engine là gì ?
  + Một tiến trình Docker daemon có nhiệm vụ các bước ban đầu để tạo và quản lý các container, image , volume , network
  + Docker Client sẽ giao tiếp với Docker daemon thông qua API theo chuẩn REST ( tức có nghĩa sẽ có các phương thức GET , POST , PUT , DELETE ) để yêu cầu Docker daemon thực thi các lệnh mà người dùng gọi đến
  + Docker CLI được xem là Client hay được hiểu là giao diện tương tác giữa người dùng ở đây là sử dụng các công cụ hỗ trợ như là Command Line (CMD), Window Terminal, …. Để gửi các yêu cầu tương ứng đến docker daemon thông qua API

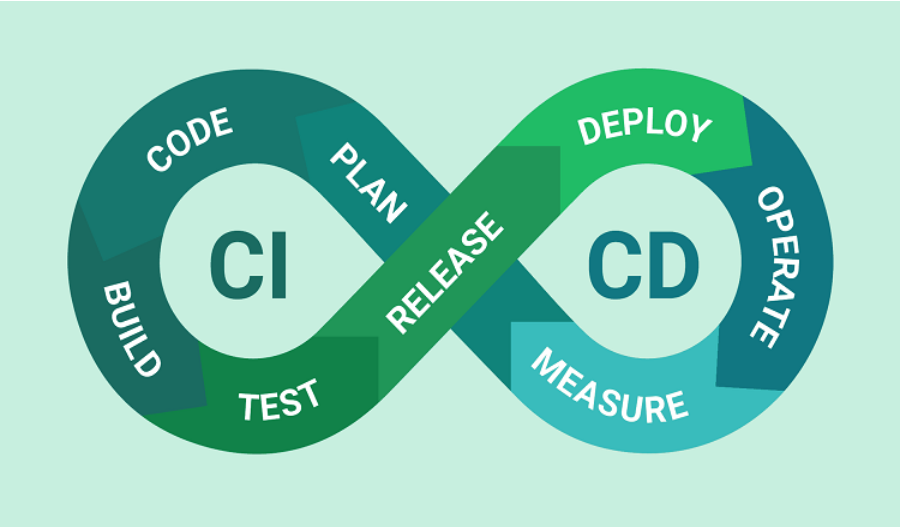


* Kiến trúc của Docker
  + Docker có kiến trúc là Client – Server và giao tiếp qua REST API (Cổng mạng và một số phương thức khác) để có thể gửi các yêu cầu đến Docker daemon. Việc giao tiếp này thực hiện các công việc như Run , Deploy , Build các docker container. Ta có thể cấu hình cho Docker Client – Docker Daemon trên cùng 1 máy , hoặc có thể dùng Docker Client để giao tiếp từ xa với Docker Daemon. Và minh họa cho kiến trúc của docker như sau

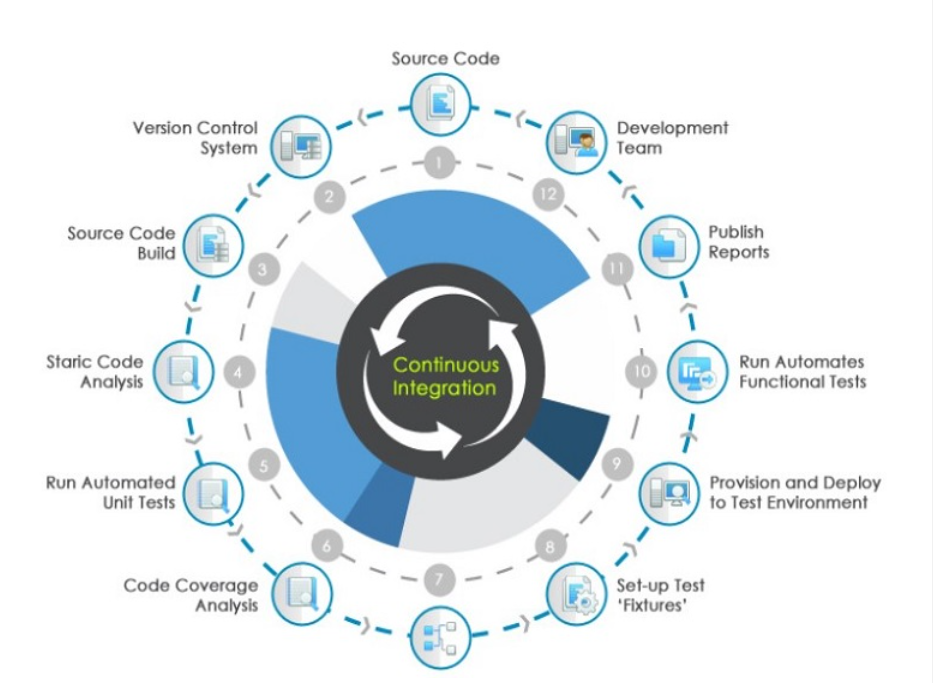


* Docker daemon
  + Daemon Docker (dockerd) lắng nghe các yêu cầu của người dùng thông qua Docker API và quản lý các Docker đối tượng như image, container, network và volume. Một daemon cũng có thể giao tiếp với các daemon khác để quản lý dịch vụ Docker.
* Docker Client
  + Docker client (docker) is the way that many user is match with Docker. Khi bạn sử dụng các câu lệnh như docker run, client thông qua Docker API sẽ gửi các câu lệnh này đến dockerd, thực hiện chúng. Docker client có thể giao tiếp với nhiều hơn một daemon Docker.
* Docker Registry
  + Docker registry sẽ là nơi lưu trữ các hình ảnh Docker. Docker Hub là nơi lưu trữ Docker image public (public registry) mà bất kỳ ai cũng có thể sử dụng và Docker được định cấu hình mặc định để tìm kiếm hình ảnh trên Docker Hub. Ngoài ra, người dùng có thể cấu hình các tư vấn đăng ký khác nhau để lưu trữ hình ảnh Docker.
* Khi người dùng sử dụng câu lệnh “docker pull” hoặc “docker run”, các image only sẽ được tải về dựa trên registry đã được cấu hình trước đó. Khi người dùng sử dụng câu lệnh “docker push”, image cũng sẽ được tải lên registry mà người dùng đã cấu hình từ trước đó.
* Docker image
  + Docker Image là template read-only (chỉ cho phép đọc) với các hướng dẫn để tạo Docker container. Image sẽ được sử dụng để đóng gói các ứng dụng và các thành phần đi kèm của ứng dụng, được lưu trữ ở server hoặc trên registry. Ví dụ bạn có thể sử dụng Dockerfile để tạo ra một Docker image sử dụng hệ điều hành Ubuntu và cài đặt Apache server với những cài đặt, cấu hình tùy chỉnh của riêng mình.
* Docker container
  + Container là 1 “runable instance” của image. Bạn có thể khởi tạo, dừng hoặc xóa container bằng cách sử dụng Docker API hoặc CLI. Bạn có thể kết nối container đến 1 hoặc nhiều network, thư mục lưu trữ, hoặc thậm chí tạo ra 1 image mới dựa trên tình trạng hiện tại của container. Mặc định 1 container được “cách ly” với các container và server nếu người dùng không có các cài đặt gì thêm.
* Docker volume
  + Volume được thiết kể để làm nơi lưu trữ các dữ liệu độc lập với vòng đời của container.
* Docker network
  + Cung cấp một private network mà chỉ tồn tại giữa container và server, giúp các container có thể giao tiếp được với nhau một cách dễ dàng.
* Docker service
  + Docker Service cho phép bạn mở rộng các contaner thông qua nhiều Docker daemon, chúng giao tiếp với nhau thông qua swarm cluster bao gồm nhiều manager và worker. Mỗi một node của swarm là 1 Docker daemon giao tiếp với nhau bằng cách sử dụng Docker API. Theo mặc định thì service được cân bằng tải trên các node.
* Cùng với Docker thì một khái niệm và là công cụ thường được sử dụng cùng với docker thì ta có CI/CD vậy CI/CD là gì

CI/CD là gì ?



* CI ( Continous Integration ) là cách thức phát triển phần mềm bằng cách tích hợp liên tục có nghĩa là nếu trong một nhóm những người cùng một dự án khi thực hiện một tính năng hay tham gia vào việc bảo trì một tính năng nào đó và sinh ra các đoạn mã mới thay cho đoạn mã cũ và đưa đoạn mã trên nhánh của người đó lên nhánh tổng ( master ) hoặc là nhánh develop thì ở đây CI sẽ thực hiện công việc của nó là tự động chạy toàn bộ source code cùng với đoạn mã của người viết đoạn mã vừa mới đưa lên và kiểm tra và check xem độ chính xác về mặt quy tắc viết mã của ngôn ngữ mà source đó sử dụng hay không. Nếu không thì sẽ tiếp tục bước tiếp theo nhưng khi có vẫn đề xảy ra thì sẽ báo lỗi cho người viết đoạn mã đó chưa khớp với đoạn mã trên nơi tổng mà các nhánh của các thành viên đổ về đó, và cũng như đưa ra những gợi ý cho người viết mã là một đề xuất để sửa. CI cũng đảm bảo source code của bạn sau gom mã của thành viên trong nhóm lại sẽ có thể hoạt động ở bất kì điều kiện nào và không xảy ra lỗi. Quá trình Build Run và Deploy sẽ được chạy và cho người quản lý source code tiến trình nó xảy ra như thế nào.
* Quá trình đưa mã code từ nhánh cá nhân lên nhánh chính ( develop hoặc master) thì quá trình CI hoạt động sẽ được hoạt động tự động nếu người quản lý source code cấu hình và cài đặt trước . Nến quá trình gom source code -> kiểm tra -> phản hồi sẽ diễn ra liền mạch và tự động. Người xây dựng mã người có thể tiết kiệm thời gian để xây dựng tính năng khác hoặc sửa lỗi nếu có và sau khi quá trình kết thúc thì người quản lý có thể xem được quá trình build và test.

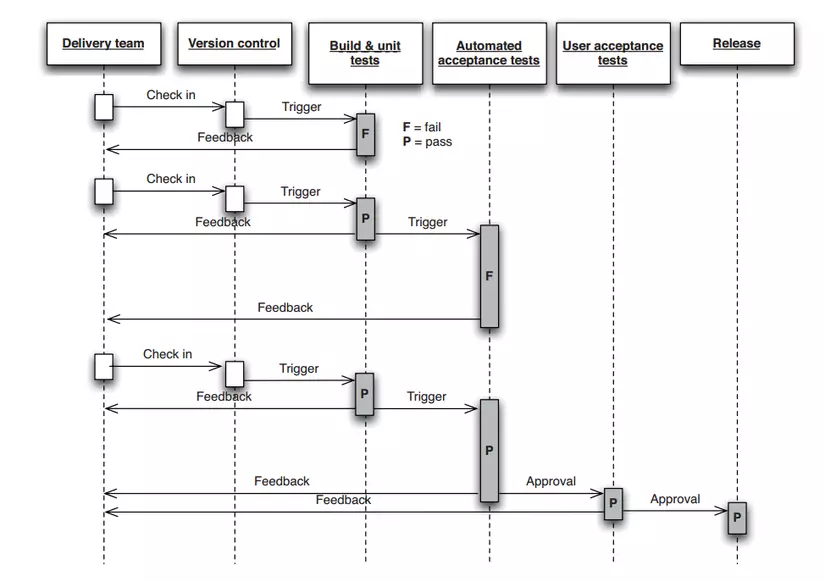


Lợi ích của việc sử dụng CI

* Giảm thiểu rủi ro nhờ việc phát hiện lỗi và fix sớm, tăng chất lượng phần mềm nhờ việc tự động test và inspect (đây cũng là một trong những lợi ích của CI, code được inspect tự động dựa theo config đã cài đặt, đảm bảo coding style, chẳng hạn một function chỉ được dài không quá 10 dòng code ...)
* Giảm thiểu những quy trình thủ công lặp đi lặp lại (build css, js, migrate, test...), thay vì đó là build tự động, chạy test tự động
* Sinh ra phần mềm có thể deploy ở bất kì thời gian, địa điểm

CD là gì : Thì đi kèm với CI là việc tích hợp liên tục thì còn một quá trình nữa gọi là chuyển giao liên tục thì nếu source code qua quá trình CI không có lỗi xảy ra thì sẽ sang một giai đoạn gọi là CD để tiến hành kiểm tra những thay đổi về mã nguồn đã được build lên và mã nguồn trong môi trường kiểm thử trước đó. CD cho phép người viết mã nguồn cài đặt những chu kì tự động ngắn và liên tục . Từ đó nhanh chống phát hiện ra lỗi sai và tiến hành chỉnh sửa ngay trong những phần tương tự.

* Do sự tiện lợi của 2 quá trình này thì người quản lý mã nguồn thường tích hợp cả hai CI/CD với nhau để tiết kiệm thời gian , chi phí và tài nguyên bộ nhớ cho 2 quá trình này, CD sẽ sử dụng giai đoạn cuối cùng của CI để bắt đầu quá trình chuyển giao của chính nó
* Trong quá trình CD thực hiện tiến trình của mình thì có thể mã nguồn sẽ trải qua một số phần testing được người quản lý mã nguồn thiết lập và một số bài test bao gồm UI Testing , API Testing, …. CD sử dụng một thuật ngữ được gọi là Deployment Pipeline giúp chia quy trình chuyển giao thành các giao đoạn. Mỗi giai đoạn sẽ đáp ứng những mục đích riêng bao gồm xác minh chất lượng của các tính năng từ một số góc độ khác nhau để kiểm định chức năng và tránh lỗi ảnh hưởng đến người dùng. Pipeline sẽ thể hiện phản hồi cho nhóm trong việc cung cấp tính năng mới hoặc sửa chữa các tính năng đã có sẵn. Deployment Pipelien là quy trình để chuyển phần mềm từ version control đến tay người dùng.
* Việc kết hợp 2 quá trình này thì ta sẽ được chu trình hoạt động của CICD gói gọn trong 3 bước là Xây dựng -> Kiểm tra -> Triển khai



Đặc điểm của CI/CD là gì?

Lợi ích của CI/CD

* **Giảm thiểu rủi ro không đáng có**

Đây có thể là một lợi ích vô cùng hữu ích của CI/CD, nó cho phép làm giảm thiểu đi những rủi ro nhờ việc phát hiện và sửa lỗi sớm, giúp tăng chất lượng sản phẩm nhờ khả năng tự động kiểm tra và quan sát. Không những vậy, những quy trình thủ công lặp đi lặp lại hằng ngày cũng được giảm tải, thay vào đó là xây dựng và kiểm thử tự động mà không cần đến sự giúp đỡ của con người. Một đặc điểm nữa của CI CD chính là có thể deploy, triển khai phần mềm ở bất cứ địa điểm và thời gian nào

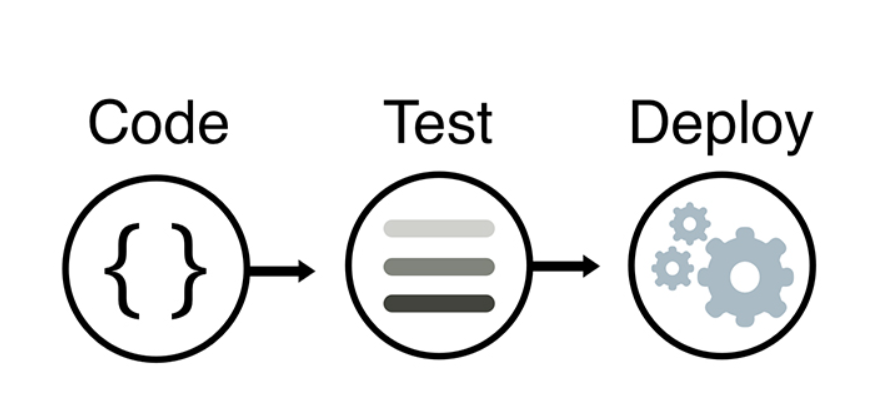
* **Thay đổi code nhỏ**

Một lợi ích vô cùng lớn của CI/CD chính là cho phép chúng ta tích hợp nhiều loại mã nhỏ cùng một lúc. Những thay đổi mã này được thực hiện một cách đơn giản và xử lý nhanh hơn so với những đoạn mã khổng lồ, từ đó làm giảm đi khả năng sinh ra những vấn đề liên quan đến việc thay đổi sau này.

Những sự thay đổi mã nhỏ này có thể được thực kiểm tra ngay sau khi chúng được tích hợp vào kho mã. Các nhà phát triển có thể dễ dàng nhận ra vấn đề trước khi lượng lớn công việc tăng lên một cách chóng mặt. Đây thực sự là một lợi thế đối với những nhóm phát triển lớn hoặc người làm việc từ xa giao tiếp được hiệu quả hơn.

* **Hạn chế những ảnh hưởng của lỗi**

CI/CD được thiết kế với hệ thống sao cho khi có lỗi nào đó xảy ra thì những kết quả tiêu cực sẽ bị giới hạn trong phạm vi ảnh hưởng nhất định nào đó. Việc hạn chế các vấn đề này giúp làm giảm khả năng hư hỏng từ đó làm cho hệ thống được bảo trì và xử lý một cách dễ dàng hơn.Với hệ thống CI CD, có thể đảm bảo cho việc cách ly lỗi sẽ được phát hiện một cách nhanh chóng và dễ dàng thực hiện hơn. Chính vì vậy, hậu quả của các lỗi trong ứng dụng sẽ được giới hạn trong phạm vi ảnh hưởng của nó.

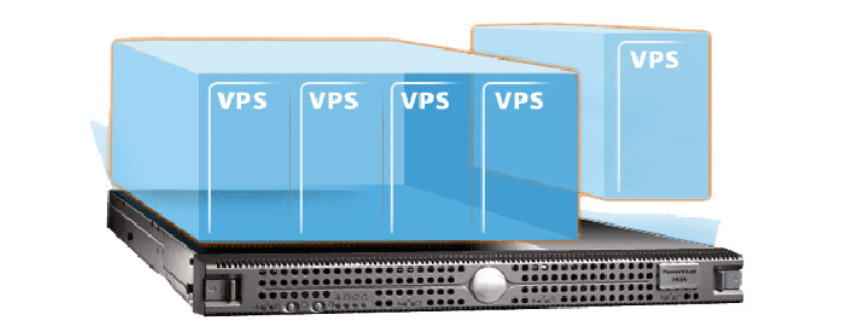


**Nhược điểm của CI/CD là gì ?**

* Bên cạnh các ưu điểm thì hệ thống CI và CD vẫn tồn tại một số điểm trừ nhất định. Vì CI/CD thường xuyên được nâng cấp để tối ưu hơn nên đòi hỏi các nhà lập trình phải nhanh chóng học hỏi các kiến thức mới.
* Ngoài ra, để quá trình tự động hóa diễn ra trơn tu, người sử dụng phải hiểu rõ cách vận hành của chương trình. Và điều này hoàn toàn không đơn giản bởi CI và CD sở hữu bộ kỹ năng khá phức tạp. Để hoàn toàn sử dụng trơn tru được CI/CD đòi hỏi quá trình rèn luyện lâu dài.
* Để có thể ứng dụng được CI và CD vào công việc đòi hỏi nhà sản xuất phải đầu tư trước một khoản phí khá lớn. Không nhiều công ty dám mạo hiểm bởi như đã nói ở trên, việc sử dụng CI và CD không dễ dàng. Và nhược điểm cuối cùng là một số hệ thống lập trình không cho phép hỗ trợ thêm CI và CD. Nếu không thể thay đổi hệ thống thì khó lòng sử dụng được chương trình này.

VPS là gì ?

* VPS là viết tắt của ( Virtual Private Server) hay còn gọi là máy chủ riêng ảo. Được cung cấp và cho thuê bởi các công ty dịch vụ Internet Hosting được tạo ra bới công nghệ ảo hóa.
* VPS hoạt động như một server để người dùng tạo ra trang web hay ứng dụng trên các tài nguyên ta cài đặt được giành riêng
* Người cài đặt VPS có toàn quyền kiểm soát bằng superuser ( Root , quyền admin ), và thông qua SSH key để điều khiển cài đặt các thư viện, hoặc cấu hình VPS theo ý muốn của người cài đặt.
* Thường thì VPS sẽ được thuê và có những option các hệ điều hành có sẵn khi người muốn thuê lựa chọn các dịch vụ và trả tiền để thành toán phí duy trì của cái máy ảo đó ở máy chủ của công ty hosting đó một số hosting phổ biến như AWS , Digital Ocean , Microsoft Azure , Ở Việt Nam thì có Mắt Bão , ….. trên điện toán đám mây
* VPS an toàn và ổn định tùy theo tình trạng mạng của bạn kết nối và ổn định hơn so với host website vì bạn không chia sẻ không gian lưu trữ với người khác. Và nó cũng có giá thành thấp hơn so với thuê một server riêng



* Thường thì các máy chủ ảo sẽ không có tên miền để thay thế cho việc truy cập đến máy chủ đó mà phải truy cập qua IP của VPS đó được sinh ra khi bạn tiến hành thuê VPS đó
* VPS là một server ảo hoạt động tương tự như một máy chủ vật lý. Máy chủ , hay server, là một máy tinh chứa source code và cơ sở dữ liệu cho ứng dụng của bạn , Môi trường và các thư viện cần thiết. Bất kể khi nào người dùng truy cập vào ứng dụng của bạn trên trình duyệt thì sẽ gửi yêu cầu đến con server của bạn và lúc này dựa vào đường truyền internet các file cần thiết để chạy 1 trang web sẽ được truyền xuống máy tính của người dùng và hiện thị nó trên tình duyệt
* Trên thực tế thì VPS là máy chủ ảo được cấu hình qua mạng như nó sẽ được hình thành trên một server vật lý, và mỗi VPS là sẽ có phần tài nguyên riêng biệt của nó và không bị ảnh hưởng bởi các VPS khác
* Các yếu tố lưu ý đến VPS khi tiến hành thuê
  + **Ram** : Trong máy tính, RAM là loại bộ nhớ chính, nếu VPS của bạn có nhiều RAM thì khả năng truy xuất dữ liệu càng tốt. Bởi vì khi sử dụng VPS bạn sẽ cần RAM để xử lý các vấn đề như xử lý truy vấn nhập xuất của database
  + **SWAP** hay là bộ nhớ ảo có nhiệm vụ lưu lại các hành động xử lý trong quá khứ trong trường hợp bộ nhớ RAM bị quá tải (overload). Hay nói cách dễ hiểu hơn, SWAP là nơi lưu trữ trên ổ cứng thay vì một bộ nhớ độc lập. Tuy nhiên, không phải VPS nào cũng hỗ trợ SWAP
  + **Disk** : Ổ cứng/Ổ đĩa cứng hay còn gọi là Disk là không gian lưu trữ được sử dụng để lưu các file cài đặt của hệ điều hành và các file của mã nguồn website. Có 2 loại ổ đĩa được sử dụng phổ biến, đó là:
    - HDD (Hard Disk Drive): Là loại ổ đĩa thông dụng được sử dụng trên máy tính.
    - SSD (Solid State Drive): SSD hay còn gọi là ổ cứng bán dẫn được sử dụng để lưu trữ dữ liệu nhưng tốc độ truy xuất dữ liệu nhanh hơn so với HDD 300 lần.
    - Do vậy, giá thành của ổ cứng SSD sẽ có đắt hơn so với loại ổ HDD.
  + **CPU Core :** là lõi xử lý của CPU. Một máy chủ riêng sẽ có lượng core nhất định và nó được chia cho các VPS. Thông thường, số core càng cao thì khả năng xử lý dữ liệu càng tốt. Số lượng core trung bình ở các gói VPS là từ 1 core đến 3 core.
  + **Bandwidth và transfer :**  đều có nghĩa là băng thông, chính là lưu lượng được phép truyền tải dữ liệu đi.
  + **IP (Internet Protocol) :** là số lượng địa chỉ IP mà nhà cung cấp dịch vụ VPS sẽ cấp cho bạn.
* Thì các công ty cho thuê dịch vụ máy chủ ảo riêng biệt như thế sẽ cho người thuê toàn quyền quyết định về hệ điều hành, cũng như cho phép chọn cấu hình của phân vùng VPS đó như sử dụng bao nhiều RAM xử lý , Bao nhiêu GB bộ nhớ , Thì Ram và bộ nhớ càng cao ứng dụng của bạn sẽ hoạt động một cách trơn tru hơn nhưng bù lại chi phí của của người thuê bỏ ra để duy trì nó sẽ càng cao hơn.



Ưu điểm của VPS là gì ?

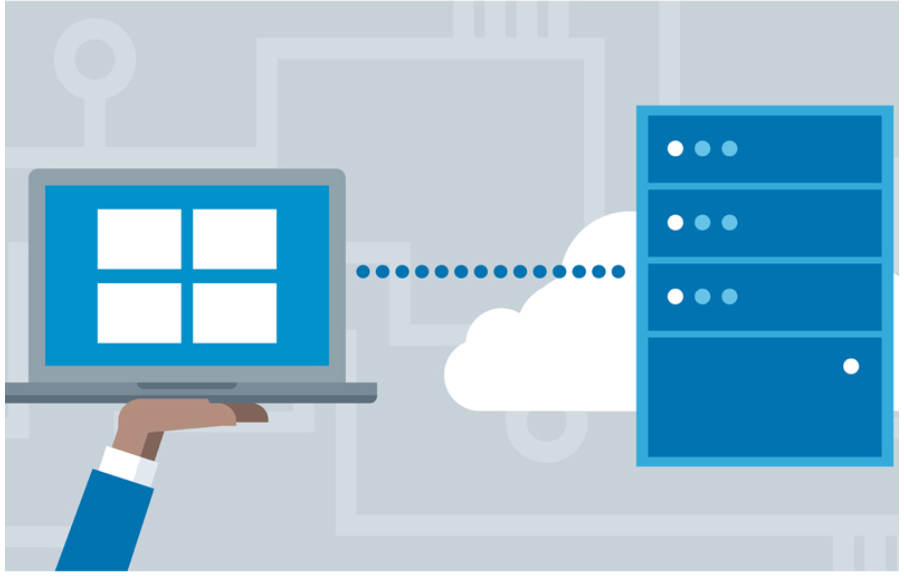
* Nhanh và đáng tin cậy hơn server shared hosting.
* Vì được đảm bảo về thông số server như bộ nhớ và sức mạnh vi xử lý, bạn sẽ không gặp phải vấn đề tài nguyên bị người khác dùng hết.
* Các vấn đề về lượng truy cập đột biến tăng cao không ảnh hưởng đến site của bạn.
* Bạn có quyền superuser (root) trên server.
* Có độ riêng tư cao hơn, vì files và databases bị khóa khỏi hệ thống server của các người dùng khác.
* Dễ dàng nâng cấp. Ngay khi website tăng trưởng, bạn chỉ cần nâng cấp gói hosting để nâng tài nguyên lên mà không phải tốn công chuyển dữ liệu hay chuyển server (RAM, CPU, disk space, bandwidth,…).
* IP riêng. Chính vì điều này khả năng chống DDoS cũng cao hơn.

Nhược điểm

* Giá cao hơn shared hosting.
* Cần ít nhiều kiến thức kỹ thuật để quản lý.
* Cấu hình server không đúng có thể tạo ra lỗ hổng bảo mật.

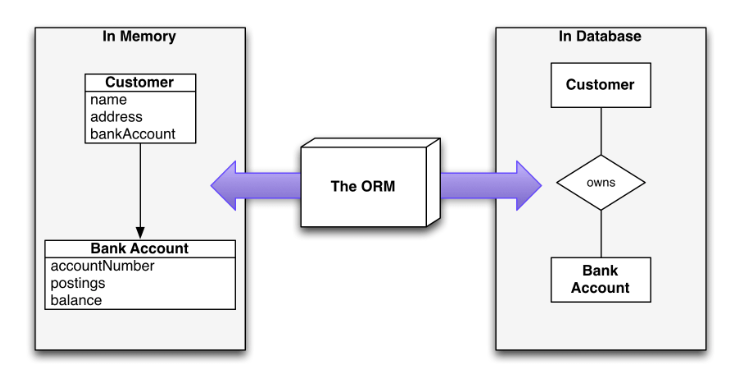
NodeJS ( ExpressJS)

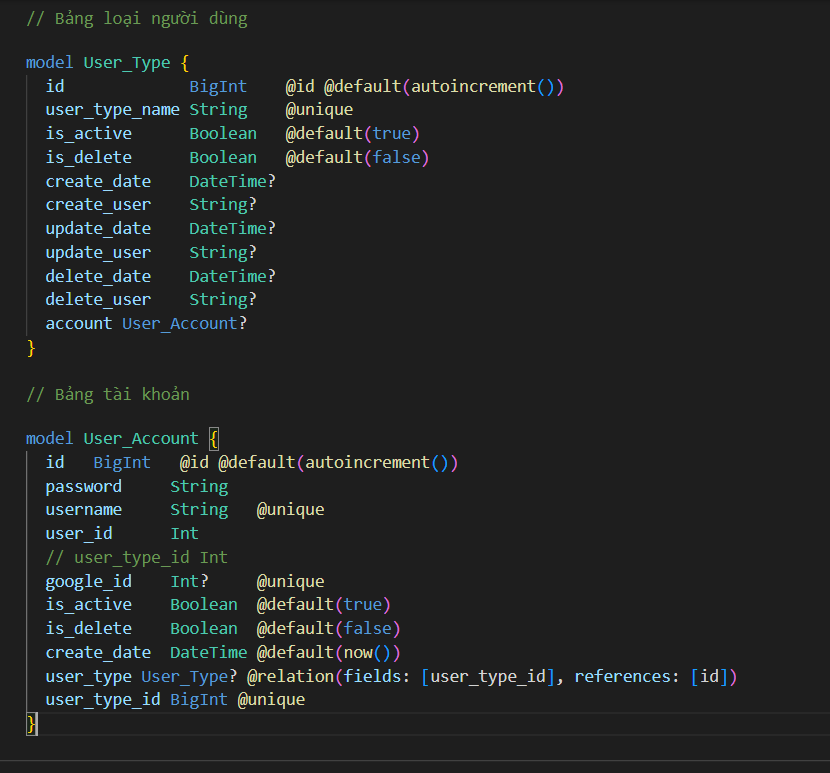
* NodeJS được xem là nền tảng để viết và chạy các mã nguồn thuộc về ngôn ngữ JavaScript, Nó cho biến ngôn ngữ JavaScript từ một ngôn ngữ chỉ hoạt động trên các client web thành một thứ mà các lập trình viên có thể sử dụng nó để tạo ra các ứng dụng độc lập trong các lĩnh vực như
  + Ứng dụng trò chuyện trong thời gian thực: Nhờ vào cấu trúc không đồng bộ đơn luồng, Node.JS rất thích hợp cho mục đích xử lý giao tiếp trong thời gian thực. Nền tảng này có thể dễ dàng mở rộng quy mô và thường dùng để tạo ra các chatbot.
  + Internet of Things (IoT): Các ứng dụng IoT thường bao gồm nhiều bộ cảm biến phức tạp để gửi những phần dữ liệu nhỏ.
  + Truyền dữ liệu: Netflix là một trong số những công ty lớn trên thế giới chuyên sử dụng Node.JS cho mục đích truyền dữ liệu.
  + Các SPA (Single-page application) phức tạp: Trong SPA, toàn bộ ứng dụng được load vào trong một trang duy nhất, do đó sẽ có một số request được thực hiện trong nền.
  + Các ứng dụng REST dựa trên API: JavaScript được sử dụng trong cả frontend lẫn backend của trang. Do đó một server có thể dễ dàng giao tiếp với frontend qua REST API bằng Node.js
* Express JS: là một Framework nhỏ, nhưng linh hoạt được xây dựng trên nền tảng của Nodejs. Nó cung cấp các tính năng mạnh mẽ để phát triển web hoặc mobile
  + Về các package hỗ trợ: Expressjs có vô số các package hỗ trợ nên các bạn không phải lo lắng khi làm việc với Framework này.
  + Về performance: Express cung cấp thêm về các tính năng (feature) để dev lập trình tốt hơn. Chứ không làm giảm tốc độ của NodeJS.
* SQL server hay còn được gọi là Microsoft SQL Server, nó từ viết tắt của MS SQL Server. Đây chính là một loại phần mềm đã được phát triển bởi Microsoft và nó được sử dụng để có thể dễ dàng lưu trữ cho những dữ liệu dựa theo tiêu chuẩn RDBMS.
* SQL Server có khả năng cung cấp đầy đủ các công cụ cho việc quản lý từ giao diện GUI đến sử dụng ngôn ngữ cho việc truy vấn SQL. Điểm mạnh của SQL điểm mạnh của nó là có thể kết hợp và hoạt động với các nền tảng khác nhau
* Và để người viết code giao tiếp với SQL server sẽ được giao tiếp với nhau thông qua một ngôn ngữ gọi là SQL , T-SQL
  + T-SQL là một trong những loại ngôn ngữ thuộc quyền sở hữu của Microsoft và được gọi với cái tên Transact-SQL. Nó thường cung cấp thêm rất nhiều cho các khả năng khai báo biến, thủ tục lưu trữ và xử lý ngoại lệ,...
  + SQL Server Management Studio là một loại công cụ giao diện chính cho máy chủ cơ sở của chính dữ liệu SQL, thông thường thì nó hỗ trợ cho cả môi trường 64 bit và 32 bit.



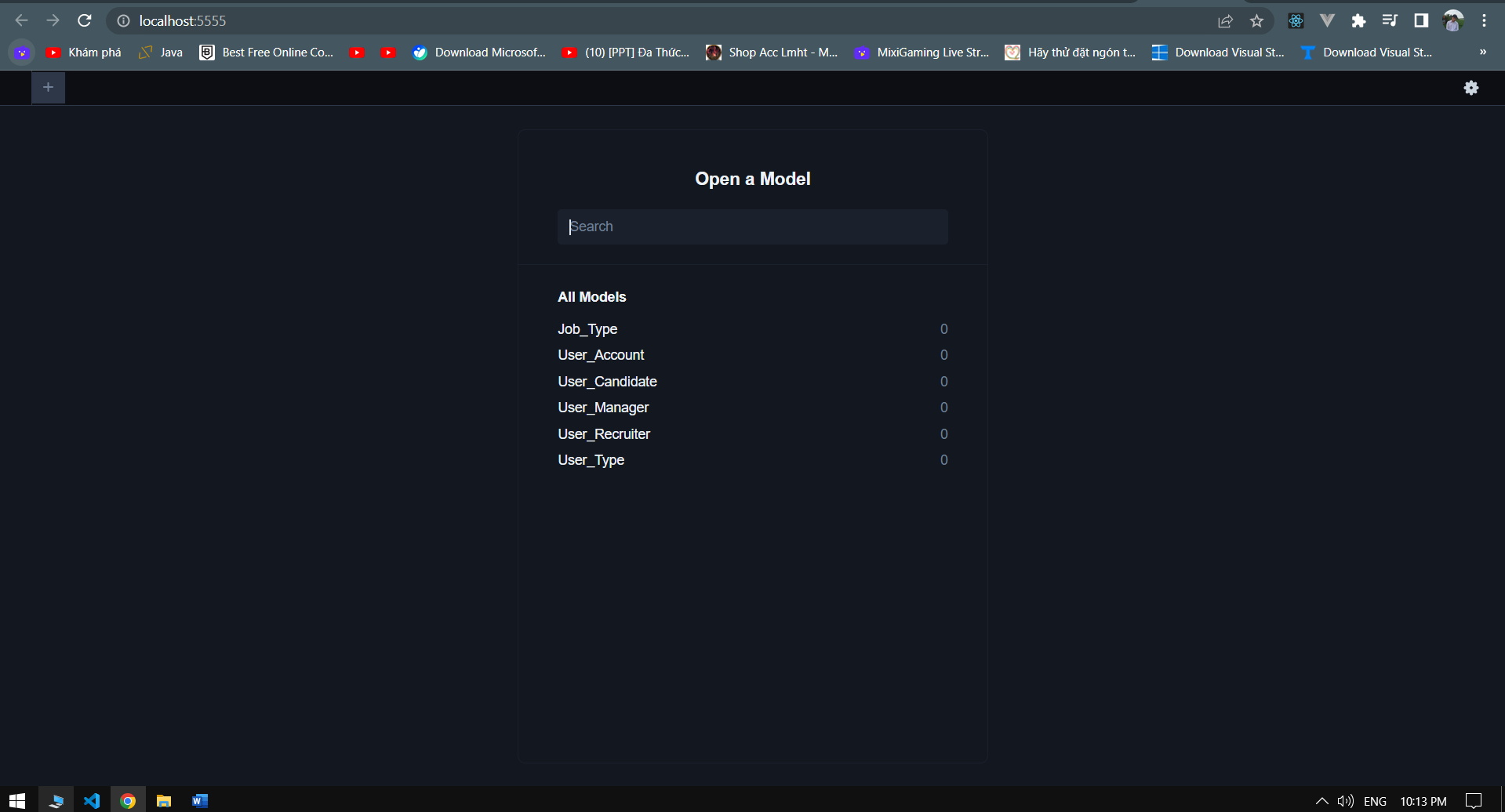
* Thì hiện nay trong thực tế có một số cách thực hiện tạo Database mà không cần thông qua GUI của SQL server mà dùng các đối tượng được định nghĩa và nhờ vào thư viện để ánh xạ các đối tượng đó vào trong cơ sở dữ liệu để hình thành các bảng và mối quan hệ của chúng thì kĩ thuật đó được gọi là ORM
* Vậy ORM là từ viết tắt của (Object Relational Mapping), là một kỹ thuật/cơ chế lập trình thực hiện ánh xạ CSDL sang các đối tượng trong các ngôn ngữ lập trình hướng đối tượng như Java, C# …(các table tương ứng các class, mối ràng buộc giữa các table tương ứng quan hệ giữa các class ‘has a’ , ‘is a’).
* Việc sử dụng ORM cho phép lập trình viên thao tác với database 1 cách hoàn toàn tự nhiên, dễ hiểu thông qua các đối tượng. Lập trình viên không cần tới loại database, kiểu dữ liệu trong database…
* Ưu điểm của ORM:
  + ORM giúp lập trình viên tập trung vào lập trình hướng đối tượng
  + Tính độc lập: Làm việc được với nhiều loại database(hệ quản trị cơ sở dữ liệu), nhiều kiểu dữ liệu khác nhau. Dễ dàng thay đổi loại database hơn. Các câu lệnh SQL không phụ thuộc vào loại database.
  + Đơn giản, dễ sử dụng: Hỗ trợ HSQL, cung cấp nhiều nhiều API truy vấn.
  + Năng suất hơn: viết code ít hơn, dễ hiểu hơn. Phù hợp các case CRUD (Create, Read, Update, Delete)
  + Khả năng sử dụng lại code.
* Nhược điểm của ORM:
  + Khả năng truy vấn bị hạn chế, nhiều trường hợp ta vẫn phải dùng native SQL để truy vấn database.
  + Khó tối ưu câu lệnh SQL (do câu lệnh SQL được ORM tự động sinh ra).
* Một thư viện hỗ trợ việc ánh xạ dữ liệu từ đối tượng trong khi xây dựng back-end bằng NodeJS đó là thư viện Prisma . Thư viện giúp ánh xạ các đối tượng ta khai báo từ dạng model được quy định thành các bảng trong cở sở dữ liệu mà ta muốn thì ở đây chúng em chọn SQL Server
* Prisma là một Object Relational Mapping (ORMs) được dùng để xây dựng các máy chủ như GraphQL Server, SQL Server, RESTful APIs, microservice, ... Prisma đơn giản là 1 layer nằm giữa Webserver và Database. Prisma giúp chúng ta giao tiếp với db một cách dễ dàng hơn.
* Prisma bao gồm ba phần chính:
  + Prisma Client : Trình tạo truy vấn an toàn và được tạo tự động cho Node.js và TypeScript.
  + Prisma Migrate : Một hệ thống di chuyển và mô hình hóa dữ liệu khai báo.
  + Prisma Studio : Một GUI để xem và chỉnh sửa dữ liệu trong cơ sở dữ liệu của bạn.
  + Cách truyền thống mà Webserver giao tiếp với Database là thông qua các câu lệnh SQL Query như SELECT, UPDATE hay DELETE. Giờ đây, nhờ vào các công cụ ORMs nói chung và Prisma nói riêng. Chúng tạo ra một tầng abstraction giữa Webserver và Database. Điều này giúp cho lập trình viên dễ dàng trong việc thao tác với Database. Thay vì viết những câu lệnh SQL khô khan, có thể sai bất cứ lúc nào thì chúng ta có thể viết các hàm tương ứng.



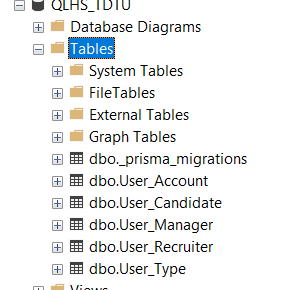




* Thì trên đây là 2 bảng được khởi tạo trong database được khởi tạo thành 2 bảng là User\_type và User\_Account thì các giá trị bắt buộc không được bỏ trống thì sẽ không có dấu ? sau kiểu dữ liệu
* Và cũng như khóa ngoại của bảng User\_account sẽ là id trong bảng User\_type
* Thì với thư viện Prisma ta có thể ánh xạ các model thành các bảng trong cở sở dữ liệu và thư viện này có hỗ trợ một số loại cơ sở dữ liệu phổ biến hiện này như: PostgreSQL , MySQL, SQLite, MongoDB, SQL Server
* Để tạo bảng trong các cơ sở dữ liệu ta chỉ cần thực hiện các thao tác sau
  + npx prisma studio : để mở GUI cho người viết code thấy được giao diện của prisma cùng với các bảng của mình



* + npx prisma migrate : Để khởi tạo các bảng lưu lại quá trình thay đổi cũng như các file có dạng SQL ứng với mỗi migrate sẽ có 1 file SQL sinh ra và Prisma sẽ dựa vào tất cả các file SQL ở từng thư mục trong migrate để khởi tạo cơ sở dữ liệu cũng như là cập nhật các thuộc tính trong quá trình ta thao tác với các model
  + Sau khi chạy câu lệnh này ta trong SQL Server ta khởi tạo và gán địa chỉ vào config để khi tạo từ prisma thì các bảng sẽ vào cơ sở dữ liệu đó sẽ hình thành các bản như các model mà ta đã khai báo trong Prisma ở NodeJS



Link tham khảo NodeJS (ExpressJS) :

<https://vietnix.vn/nodejs-la-gi/>

<https://topdev.vn/blog/express-js-la-gi/>

<https://viblo.asia/p/nodejs-tutorial-phan-4-express-framework-924lJXpNKPM>

Link tham khảo Prisma ORM (SQL Server) : <https://www.prisma.io/docs/concepts/components/prisma-schema>

<https://www.goccuachung.com/orm-la-gi-ma-nha-nha-su-dung/>

<https://hocspringboot.net/2021/08/09/orm-la-gi-tong-quan-ve-orm-framework-2/>

<https://stackjava.com/uncategorized/orm-la-gi-tong-quan-ve-orm-framework.html>

<https://longvan.net/sql-server-la-gi.html>

Link Tham khảo VPS :

<https://vietnix.vn/vps-la-gi/>

<http://fit.tdc.edu.vn/blog/2016/04/vps-la-gi-vps-c-dung--lam-gi>

<https://www.hostinger.vn/huong-dan/vps-la-gi-tat-ca-cac-dieu-can-biet-ve-may-chu-ao>

<https://tenten.vn/tin-tuc/vps-la-gi/>

Link tham khảo (Docker) :

<https://docs.docker.com/get-started/overview/#docker-architecture>

<https://docs.docker.com/get-started/overview/>

<https://topdev.vn/blog/docker-la-gi/>

<https://docs.docker.com/engine/>

<https://tel4vn.edu.vn/blog/gioi-thieu-tong-quan-ve-kien-truc-cua-docker/>

Link tham khảo (CI/CD) :

<https://docs.docker.com/ci-cd/best-practices/>

<https://teky.edu.vn/blog/ci-cd-la-gi/>

<https://itnavi.com.vn/blog/ci-cd-la-gi>

<https://topdev.vn/blog/trien-khai-ci-cd-voi-gitlab/>

<https://viblo.asia/p/ci-cd-va-devops-07LKXYXDZV4>