**Nhóm SV tìm hiểu một công nghệ mới và ứng dụng công nghệ này vào xây dựng phần mềm. Nêu rõ các bước trong quy trình phát triển phần mềm.**

**1.1 Vì sao cần DevOps?**

Thông thường vòng đời của một sản phẩm phần mềm gồm 2 giai đoạn chính: giai đoạn phát triển và giai đoạn vận hành. Giai đoạn phát triển bao gồm phần việc của Business Analyst, UI/UX Designer, Developer, QA/QC Engineer, v.v... Giai đoạn vận hành có sự tham gia của System Engineer, System Administrator, Operation Executive, Release Engineer, Database Administrator, Network Engineer, Security Engineer, v.v... Ở mỗi giai đoạn đều có những đội nhóm và công việc tách biệt. Sự tách biệt này càng phân hóa rõ ràng ở những công ty có quy mô từ trung bình trở lên.

Mục tiêu của nhóm phát triển là phát triển các tính năng mới liên quan đến những thay đổi đối với môi trường hiện có. Điều đó có nghĩa là những thay đổi mới được đưa vào một môi trường mà trên thực tế một môi trường có tính ổn định có thể bị mất ổn định do những thay đổi được đưa ra điều này mâu thuẫn trực tiếp với mục tiêu của các nhóm vận hành. Sự mâu thuẫn này hình thành nên khái niệm Wall of confusion - Bức tường của sự hiểu lầm. Nó sẽ là rào cản kìm hãm sự phát triển phần mềm khiến cho nỗ lực làm việc của nhân viên cũng như chất lượng phần mềm ngày càng giảm sút.

Về mặt sản phẩm, với sự phát triển của công nghệ kéo theo sự bùng nổ về quy mô của các công ty cũng như sự đa dạng các sản phẩm công nghệ khiến cho quy mô của các hệ thống cũng mở rộng theo cấp số nhân. Từ một vài server, hệ thống có thể phát triển lên đến hàng chục, hàng trăm, hàng nghìn, hoặc thậm chí hàng triệu server (ví dụ như trường hợp của Google, Facebook). Đồng thời ngành phát triển phần mềm cũng dịch chuyển theo một hướng khác – microservices - một sản phẩm lớn được chia tách ra thành rất nhiều các service nhỏ, các service này liên kết với nhau tạo thành một sản phẩm hoàn chỉnh. Trước nhu cầu phát triển và cải tiến sản phẩm liên tục, những thách thức mới, những bài toán mới được đặt ra:

**Về mặt quy trình, hay là khía cạnh con người, làm thế nào để các bộ phận hợp tác thuận lợi hơn? Làm thế nào để quy trình làm việc được trơn tru, suôn sẻ hơn?**

Về mặt sản phẩm, làm thế nào để các service kết nối và giao tiếp với nhau theo những nguyên tắc hiệu quả, cũng như đảm bảo việc mở rộng quy mô được dễ dàng và êm ái hơn?

Khái niệm DevOps ra đời nhằm giải quyết những vấn đề này. DevOps giúp tối ưu hóa chu trình phát triển phần mềm, giúp sản phẩm phần mềm được cập nhật, phát hành nhanh và thường xuyên hơn.

**Lợi ích của chính DevOps** đó là việc hoàn thiện, chuyển đổi quy trình phát triển và vận hành của phần mềm từ mô hình truyền thống Waterfall (thác nước) sang dạng mô hình CI/CD (phát triển liên tục) [1] [2].

Ngoài ra, DevOps còn có một số lợi ích khác như:

Tốc độ: hoạt động với tốc độ nhanh và tần suất cao giúp cải thiện khả năng phục vụ khách hàng, thích nghi linh hoạt với thị trường thay đổi liên tục và đạt được hiệu quả kinh doanh đáng kể. Và DevOps là công cụ hỗ trợ cho các nhà phát triển và nhóm nghiệp vụ để đạt được những mục tiêu này.

Chuyển giao nhanh chóng: Việc tăng tần suất và nhịp độ release là cần thiết để cải thiện và nâng cấp sản phẩm một cách nhanh chóng. Khi các tính năng mới được release sẽ giúp khắc phục các sự cố kỹ thuật một cách nhanh chóng hơn, đáp ứng được các nhu cầu của khách hàng và tạo được lợi thế cạnh tranh.

Độ tin cậy: Mặc dù tốc độ chuyển giao được đẩy nhanh nhưng DevOps vẫn sẽ giúp đảm bảo được độ tin cậy, chất lượng của sản phẩm và trải nghiệm của người dùng cuối thông qua việc áp dụng CI /CD, Monitoring và Logging.

Khả năng mở rộng: Nhờ sự tự động hóa và tính nhất quán của DevOps sẽ giúp quản lý hiệu quả những hệ thống phức tạp hay những hệ thống thường xuyên thay đổi ở mức rủi ro thấp nhất.

Cải thiện khả năng cộng tác: Việc xây dựng các nhóm làm việc hiệu quả hơn theo mô hình DevOps nhấn mạnh vào giá trị của sự tự chủ và trách nhiệm đối với công việc. Các nhà phát triển và nhóm nghiệp vụ cùng hợp tác chặt chẽ, chia sẻ trách nhiệm và phối hợp thực hiện các quy trình công việc. Điều này giúp giảm thiểu sự chậm trễ và tiết kiệm thời gian.

Bảo mật: DevOps giúp quá trình được đẩy nhanh mà không ảnh hưởng đến tính bảo mật nhờ các chính sách tuân thủ tự động hóa, các công cụ kiểm soát và các kỹ thuật quản lý.

**1.2 Giới thiệu về DevOps**

**1.2.1 Khái niệm DevOps**

DevOps không có một khái niệm cụ thể được xác định. Theo định nghĩa mô hình DevOps của AWS, "DevOps là sự kết hợp các triết lý văn hoá, thực tiễn và công cụ giúp nâng cao khả năng của tổ chức trong việc cung cấp các ứng dụng và dịch vụ với tốc độ cao, phát triển và cải tiến sản phẩm với tốc độ nhanh hơn các tổ chức sử dụng quy trình quản lý cơ sở hạ tầng và phát triển phần mềm truyền thống. Tốc độ này cho phép các tổ chức phục vụ khách hàng tốt hơn và cạnh tranh hiệu quả hơn trên thị trường" [3]. Đơn giản hơn, DevOps là viết tắt của Development (Dev) và Operations (Ops), là một quá trình đề cao sự hợp tác, kéo hai giai đoạn phát triển và vận hành xích lại gần nhau hơn.

Có thể coi DevOps là sự kết hợp của ba yếu tố Tư tưởng mới, Công cụ mới và Kỹ năng mới [1].

Tư tưởng mới chính là DevOps Engineer cần cần đặt lợi ích doanh nghiệp, lợi ích sản phẩm lên hàng đầu. Trong DevOps, các nhóm không hoạt động tách biệt – ít nhất là các nhóm phát triển và và các nhóm vận hành hỗ trợ sản phẩm. DevOps phải chỉ ra rằng cả nhóm phát triển và nhóm vận hành đều thuộc cùng một nhóm, họ không có những mục tiêu khác nhau. Toàn bộ nhóm có một bộ mục tiêu duy nhất áp dụng cho tất cả các vai trò trong nhóm, cùng chia sẻ lợi ích cũng như rủi ro. Do đó, DevOps Engineer phải là người đứng giữa, có tư tưởng của cả hai phía để tăng cường sự hợp tác, giải quyết mâu thuẫn, không chỉ phá vỡ bức tường mà còn làm cho toàn bộ quá trình phát triển và hỗ trợ trở nên dễ dàng và liền mạch.

Công cụ mới là việc DevOps Engineer phải thường xuyên tìm hiểu các công cụ mới, lựa chọn, đánh giá sau đó áp dụng vào hệ thống cơ sở hạ tầng của công ty. Nhiệm vụ quan trọng của DevOps Engineer là tự động hóa hệ thống. Thậm chí DevOps Engineer tự phát triển các công cụ tự động hóa cho công ty. Kỹ năng mới xuất phát từ việc DevOps là một vị trí mới với vai trò mới. Công việc chính của DevOps Engineer rất gần với công việc của System Administrator, bao gồm: deploy (triển khai), optimizing (tối ưu hóa), monitoring (giám sát), analysis (phân tích), v.v... Điểm khác biệt là DevOps Engineer đòi hỏi nhiều kĩ năng mềm hơn, đồng thời phải biết dùng coding (kỹ năng lập trình), scripting (kỹ năng viết kịch bản) để tự động hóa hệ thống. DevOps Engineer cũng cần tìm hiểu về bộ công nghệ mà sản phẩm công ty đang sử dụng, để có thể cùng đánh giá lỗi, viết những unit test thông thường, và để khi phát triển quy trình CI/CD có thể deploy dễ dàng hơn.

***Quy trình thực hành DevOps bao gồm các phương pháp:***

CI – Tích hợp liên tục: hợp nhất các thay đổi code để đảm bảo phiên bản mới nhất có sẵn cho các nhà phát triển.

CD – Phân phối liên tục và triển khai liên tục: tự động hóa quá trình release cá bản cập nhật để tăng hiệu quả.

Microservices: xây dựng một ứng dụng dưới dạng một tập hợp các dịch vụ nhỏ.

IaC – Cơ sở hạ tầng dưới dạng mã: Nhu cầu thiết kế, triển khai và quản lý cơ sở hạ tầng ứng dụng thông qua code.

**1.2.2 Sự hình thành của DevOps [4] [5]**

Những đề xuất kết hợp các phương pháp phát triển phần mềm với khái niệm triển khai và vận hành bắt đầu xuất hiện vào cuối những năm 80, đầu những năm 90. Năm 2007, Patrick Debois - một nhà tư vấn, quản lý dự án và là một chuyên gia Agile người Bỉ bắt đầu thực hiện một dự án di chuyển trung tâm dữ liệu cho một bộ của chính phủ Bỉ. Ông nhận thấy rằng những đội ngũ phát triển (Dev) và những đội ngũ vận hành (Ops) không phối hợp tốt với nhau. Trong khi những khoảng cách và xung đột giữa Dev và Ops luôn khiến ông lo lắng, thì việc liên tục chuyển đổi qua lại trong một dự án di chuyển trung tâm dữ liệu lớn nơi ông chịu trách nhiệm thử nghiệm đã khiến ông đặc biệt thất vọng.

Năm 2008, tại hội nghị Agile, Andrew Shafer đã tổ chức một cuộc họp để thảo luận về “Cơ sở hạ tầng Agile”. Andrew không nghĩ sẽ có ai đến nên bản thân ông ấy cũng không đến dự cuộc họp của mình. Patrick Debois xuất hiện và đi tìm Andrew vì ông ấy muốn nói về cơ sở hạ tầng Agile là giải pháp giúp các hoạt động trở nên Agile giống như các nhà phát triển. Đây là nơi DevOps bắt đầu.

Năm 2009, tại hội nghị Velocity, John Allspaw và Paul Hammond đã nói về

“10+ deploys per day - Dev and Ops Cooperation at Flickr” và ý tưởng này bắt đầu thu hút được sự chú ý. Buổi nói chuyện này khiến mọi người chú ý đến những gì có thể thực hiện được bằng cách áp dụng các phương pháp thực hành DevOps ban đầu này. Ngoài ra, vào tháng 10 năm 2009, Patrick đã tổ chức hội nghị DevOpsDays đầu tiên ở Ghent, Bỉ. Nó được mô tả là “Hội nghị gắn kết sự phát triển và vận hành lại với nhau”.

Đây là nơi thuật ngữ "DevOps" lần đầu tiên được sử dụng. DevOpsDays ngày nay đã trở thành hội nghị được tổ chức quốc tế và lan rộng ra khắp các quốc gia, thành phố.

Năm 2010, Jez Humble và David Farley đã viết một cuốn sách mang tính đột phá có tên là Continuous Delivery, trong đó đặt ra các nguyên tắc và thực tiễn kỹ thuật cho phép phân phối nhanh chóng. Thông qua tự động hóa các quy trình build, deploy và test, cùng với sự cộng tác được cải thiện giữa các đội nhóm developers, testers, operations và delivery có thể tạo ra sự thay đổi trong vài giờ, thậm chí là vài phút bất kể quy mô của dự án hay mức độ phức tạp.

Năm 2013, Gene Kim, Kevin Behr và George Spafford xuất bản cuốn sách The Phoenix Project, dựa trên cuốn sách The Goal của Eliyahu Goldratt. The Phoenix Project kể về một cửa hàng công nghệ thông tin trong một công ty sắp phá sản và những gì cần làm để đưa nó hoạt động trở lại. Câu chuyện này nói về việc áp dụng các nguyên tắc sản xuất tinh gọn vào việc phát triển và phân phối phần mềm.

Vào năm 2015, Tiến sĩ Nicole Forsgren, Gene Kim và Jez Humble đã thành lập một công ty khởi nghiệp có tên DORA (DevOps Research and Assessment), nơi tạo ra nghiên cứu DevOps lớn nhất cho đến nay được gọi là Báo cáo State of DevOps. Nicole là Giám đốc điều hành và là một nhà thống kê tài giỏi. Thông qua nghiên cứu này, cô nhận thấy rằng việc áp dụng phương pháp thử nghiệm để phát triển sản phẩm có thể cải thiện hiệu suất tổ chức và hiệu suất IT, đồng thời các tổ chức có hiệu suất cao đang vượt trội hơn hẳn so với các tổ chức có hiệu suất thấp hơn về mặt thông lượng. Nghiên cứu cho thấy rằng việc thực hiện sáng kiến chuyển đổi công nghệ có thể giúp tiết kiệm chi phí đáng kể ở bất kỳ tổ chức nào.

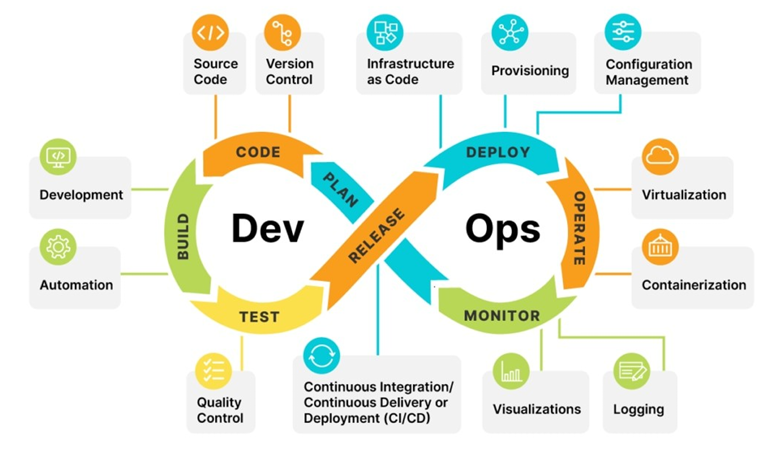
The DevOps Handbook được xuất bản vào năm 2016. Nó được viết bởi Gene Kim, Jez Humble, Patrick Debois và John Willis như phần tiếp theo của The Phoenix Project và đóng vai trò là hướng dẫn thực tế về cách triển khai các khái niệm được giới thiệu trong cuốn sách đó. Nhân tiện, John Willis ngày đó đã làm việc tại Docker và Chef là điều phối viên cho DevOpsDays sau khi tham gia DevOpsDays ban đầu ở Ghent năm 2009 cùng với Patrick Debois.

Năm 2019, 10 năm sau DevOpsDays đầu tiên ở Ghent, Bỉ, hơn 60 sự kiện DevOpsDays đã được tổ chức tại 21 quốc gia. Patrick Debois đã lãnh đạo DevOpsDays từ khi thành lập vào năm 2009 cho đến năm 2014, và sau đó Bridget Kromhout trở thành người dẫn dắt vào năm 2015. Cô cũng là người đồng dẫn chương trình Podcast rất nổi tiếng, Arrested DevOps. Cô từ chức vào năm 2020 nhưng vẫn ở trong ban cố vấn của DevOpsDays cùng với Patrick.

Những cá nhân được đề cập ở trên là một số người có ảnh hưởng lớn trong phong trào DevOps thời kỳ đầu. Họ không phải là những người duy nhất, nhưng họ đã nỗ lực và tạo nên sự khác biệt. Họ đã cho chúng ta thấy DevOps có thể có tác động như thế nào.

**1.2.3 Các giai đoạn của DevOps trong vòng đời phát triển phần mềm [6]**

Với mô hình DevOps, vòng đời phát triển phần mềm có thể được chia thành 8 giai đoạn chính: Plan, Code, Build, Test, Release, Deploy, Operate, Monitor.



Hình 1.1: Mô hình các giai đoạn của DevOps

Ở giai đoạn Plan - lên kế hoạch và thiết kế sản phẩm, DevOps Engineer sẽ tham gia thảo luận, đánh giá và đưa ra đề xuất về:

Kiến trúc ứng dụng (Architeture): Phần mềm bao gồm những thành phần nào, luồng hoạt động ra sao, yêu cầu về cơ sở hạ tầng như nào để đội ngũ quản trị hạ tầng có thể đáp ứng được?

Những phụ thuộc (Dependencies): Phần mềm yêu cầu những công nghệ kèm theo nào, hạ tầng có sẵn hay không, có dễ maintain hay không?

Kích thước (Sizing): Phần mềm yêu cầu kích thước tài nguyên ra sao để đội ngũ quản trị sắp xếp và cấp phát cho hợp lý?

Ngoài ra tại bước này, DevOps Engineer cũng đưa ra các nguyên tắc, các hướng dẫn để đội ngũ Developer thực hiện nhằm thống nhất quy trình phát triển và vận hành chung, ví dụ:

Git: Tổ chức git branch, phân chia môi trường, v.v... cho hệ thống CI.

Config: Cấu hình ứng dụng như thế nào, đặt ở đâu?

Logging: Định dạng log như thế nào, những gì nên log, log theo các cấp độ INFO, DEBUG, WARNING, ERROR, v.v...

Healthcheck: những ứng dụng nên được triển khai healthcheck nhằm giám sát tình trạng của ứng dụng.

Concurrency: Khả năng đáp ứng đồng thời hỗ trợ cho việc mở rộng quy mô theo chiều ngang (Horizontal Scaling).

Disposability: ứng dụng phải xử lý các tín hiệu hệ thống (system signal) ví dụ SIGHUP, SIGTERM, SIGKILL, v.v... để graceful shutdown.

Tracing: Tích hợp các công cụ truy vết yêu cầu để phát hiện ra lỗi hoặc độ trễ xử lý tại các service trong những ứng dụng phức tạp như các ứng dụng có kiến trúc microservice.

Giai đoạn Code - lập trình phần mềm, ở giai đoạn này DevOps Engineer không trực tiếp tham gia vào lập trình phát triển các chức năng mà sẽ hỗ trợ quản lý git repository, tham gia vào việc đánh giá code.

Giai đoạn Build - xây dựng ứng dụng, DevOps Engineer sẽ dựng các công cụ để xây dựng ứng dụng và thiết kế thành một hệ thống tự động. Các công cụ có thể kể đến như Docker, Kanico, Buildah, ... để đóng gói ứng dụng dưới dạng container; GitLab CI, Jenkins, Circle CI, v.v... để tự động quá quy trình CI, giúp đóng gói ứng dụng một cách nhanh chóng. Hiện nay phổ biến đóng gói ứng dụng bằng Docker trong đó tối ưu Dockerfile là một kỹ thuật không quá khó nhưng rất cần thiết và mang lại lợi ích rấ lớn. Tối ưu Dockerfile hay tối ưu quy trình build ứng dụng có thể:

Giảm thời gian build: tăng tốc phát triển phần mềm.

Giảm kích thước image sau khi build: dễ dàng lưu trữ, phân phối tới các môi trường khác nhau..

Tăng cường bảo mật cho ứng dụng.

Giai đoạn Test - kiểm thử phần mềm, DevOps Engineer có thể tham gia viết các test case, test plan đơn giản. Chuẩn bị và xây dựng môi trường kiểm thử.

Giai đoạn Release - phát hành phần mềm, đây là giai đoạn chuyển giao phần mềm từ các đội ngũ phát triển sang đội ngũ vận hành. DevOps Engineer sẽ yêu cầu đội ngũ phát triển viết Release note để theo dõi những thay đổi mới giúp các đội ngũ khác như kiểm thử, vận hành tập trung theo dõi hoạt động của những thay đổi đó và theo dõi mức độ ảnh hưởng tới những thành phần khác trên hệ thống. Đánh giá những rủi ro phát sinh và lên kế hoạch migrate môi trường cũng như khả năng rollout về phiên bản cũ nếu có sự cố xảy ra; quản lý các release revision.

Giai đoạn Deploy - triển khai phần mềm lên các môi trường, DevOps Engineer tiếp nhận yêu cầu triển khai phần mềm, sau đó chuẩn bị môi trường bao gồm tài nguyên cơ sở hạ tầng và các thành phần phụ thuộc của ứng dụng. Xây dựng hệ thống CD để tự động quy trình triển khai cũng như phân phối các phiên bản ứng dụng tới những môi trường khác nhau.

Giai đoạn Operate - vận hành phần mềm, DevOps Engineer sẽ xây dựng và cung cấp cho đội ngũ phát triển những công cụ để họ có thể tiếp cận sâu hơn vào việc vận hành sản phẩm, phát hiện và sửa lỗi nhanh nhất có thể. Có thể kể đến một số hạng mục cần theo dõi như:

Centralize log: Hệ thống theo dõi log tập trung để phát hiện lỗi.

Resource monitoring: Giám sát lượng tài nguyên ứng dụng sử dụng để đưa ra kế hoạch tối ưu hiệu năng.

CI/CD dashboard: Giao diện quản lý theo dõi quá trình CI/CD để biết tình trạng của ứng dụng.

Tracing: truy vết lỗi hệ thống.

Notification: hệ thống gửi thông báo khi có sự cố bất thường xảy ra với ứng dụng.

Giai đoạn Monitor - Giám sát phần mềm, ở giai đoạn này DevOps Engineer cùng với đội ngũ quản trị, vận hành hệ thống thực hiện giám sát toàn bộ hệ thống không chỉ riêng từng ứng dụng. Giám sát lượng tài nguyên tổng, lên kế hoạch mở rộng quy mô cơ sở hạ tầng, đánh giá rủi ro, lên kế hoạch backup/restore, disaster recovery. Tìm hiểu công nghệ mới để áp dụng tối ưu quy trình hệ thống. Đồng thời cũng triển khai những hệ thống monitoring/alerting khi có sự cố hệ thống xảy ra.