**Tuần 2 : Tìm hiểu chung về quy trình phát triển phần mềm và DevOps cùng các thứ liên quan**

**1: Tìm hiểu chung về quy trình phát triển phần mềm**

1. **Khái niệm**

**Quy trình phát triển phần mềm (SDLC) là một tập hợp các bước được thực hiện để tạo ra phần mềm. SDLC giúp đảm bảo rằng phần mềm được phát triển một cách hiệu quả và đáp ứng được nhu cầu của người dùng.**

**b) Tại sao nên áp dụng quy trình phát triển phần mềm?**

**Chúng ta nên áp dụng quy trình phát triển phần mềm là vì:**

* **Quy trình phát triển phần mềm giúp xác định rõ ràng các mục tiêu, phạm vi, nguồn lực, thời gian và chi phí của dự án, giảm thiểu rủi ro và tăng khả năng thành công.**
* **Quy trình phát triển phần mềm giúp cải thiện giao tiếp và hợp tác giữa các bên liên quan, như khách hàng, nhà phát triển, người dùng và nhà quản lý.**
* **Quy trình phát triển phần mềm giúp áp dụng các tiêu chuẩn, kỹ thuật và công cụ chuyên nghiệp để tạo ra các sản phẩm phần mềm chất lượng cao, an toàn và bảo mật.**
* **Quy trình phát triển phần mềm giúp theo dõi và kiểm soát tiến độ, chất lượng và chi phí của dự án, đánh giá hiệu suất và đưa ra các biện pháp khắc phục khi cần.**
* **Quy trình phát triển phần mềm giúp duy trì và nâng cấp phần mềm theo thời gian, đáp ứng nhu cầu thay đổi của khách hàng và thị trường.**

**6 giai đoạn của quy trình phát triển phần mềm**



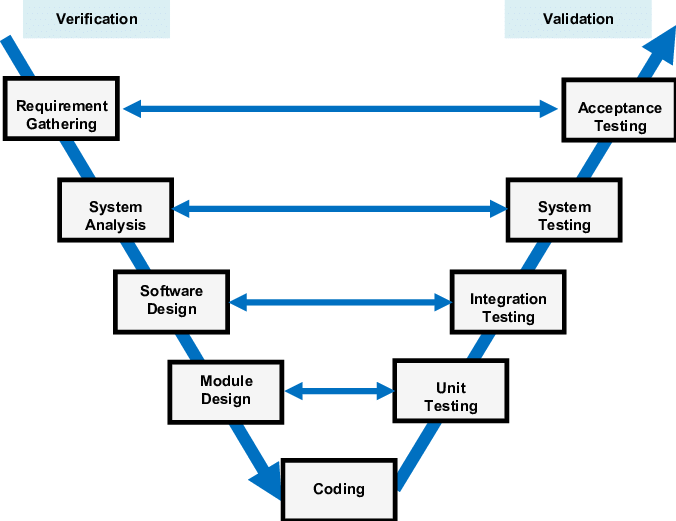
**Một quy trình phát triển phần mềm thông thường sẽ gồm 6 giai đoạn chính sau:**

* **Lên kế hoạch và phân tích yêu cầu - Analysis.**
* **Thiết kế phần mềm - Design.**
* **Tiến hành coding - Development.**
* **Kiểm thử - Testing.**
* **Triển khai - Deployment.**
* **Bảo trì - Maintenance.**

**d) Các mô hình phát triển phần mềm.**

**Trên thực tế, có rất nhiều mô hình phát triển phần mềm được áp dụng để thực hiện các giai đoạn của quy trình phát triển phần mềm. Mỗi mô hình có những ưu điểm, nhược điểm và phù hợp với những loại dự án khác nhau. Dưới đây là 5 mô hình phổ biến nhất:**

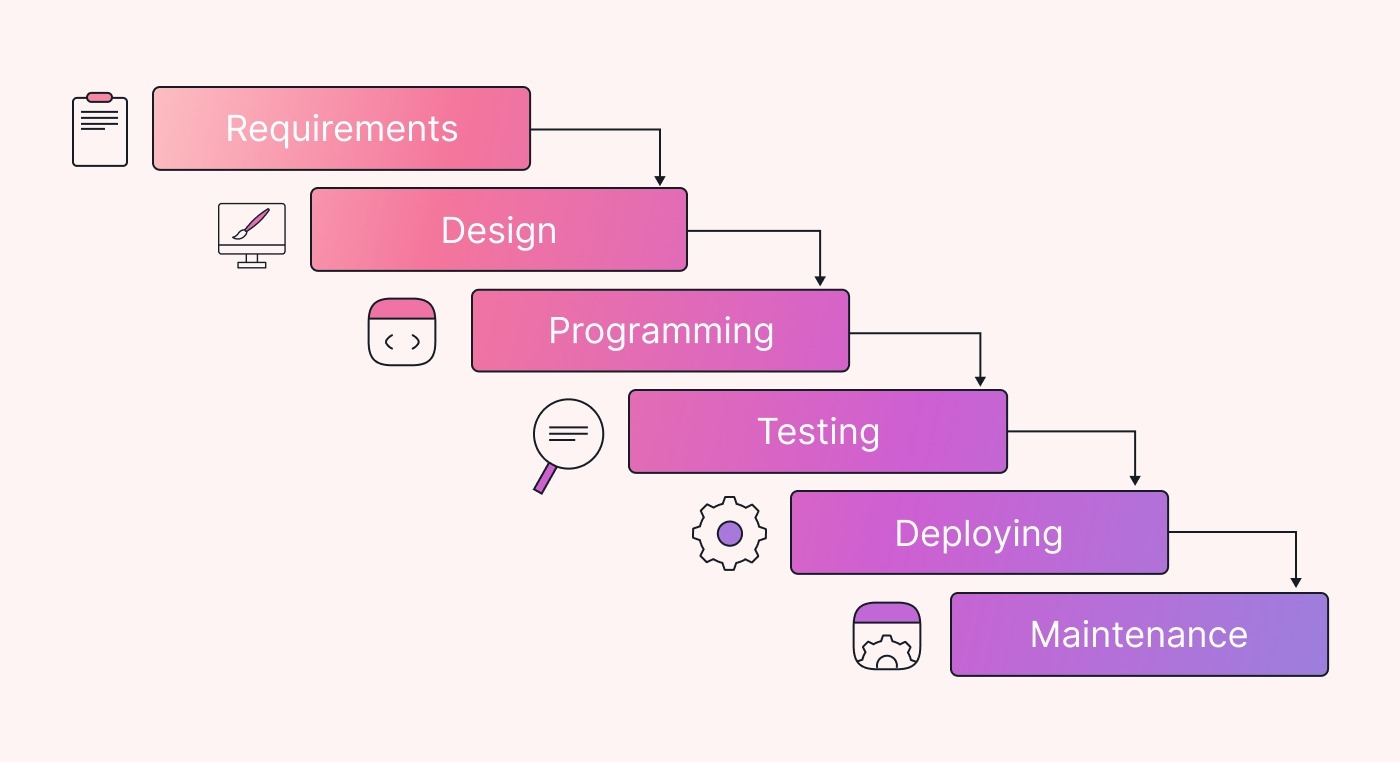
**Mô hình chữ V - V Model**

****

**Mô hình chữ V là một mô hình tuyến tính, theo chiều xuống từ trên xuống dưới. Mô hình này gồm hai nhánh song song: nhánh bên trái là các giai đoạn lên kế hoạch, phân tích yêu cầu và thiết kế; nhánh bên phải là các giai đoạn coding, kiểm thử và triển khai. Mỗi giai đoạn bên trái sẽ tương ứng với một giai đoạn bên phải theo nguyên tắc xác minh (verification) và xác nhận (validation).**

**Mô hình này có ưu điểm là rõ ràng, có kỷ luật cao và dễ kiểm soát. Tuy nhiên, mô hình này có nhược điểm là khó thích ứng với thay đổi yêu cầu, không linh hoạt và có nguy cơ cao khi lỗi xảy ra ở cuối quy trình.**

**Mô hình thác nước - Waterfall Model**

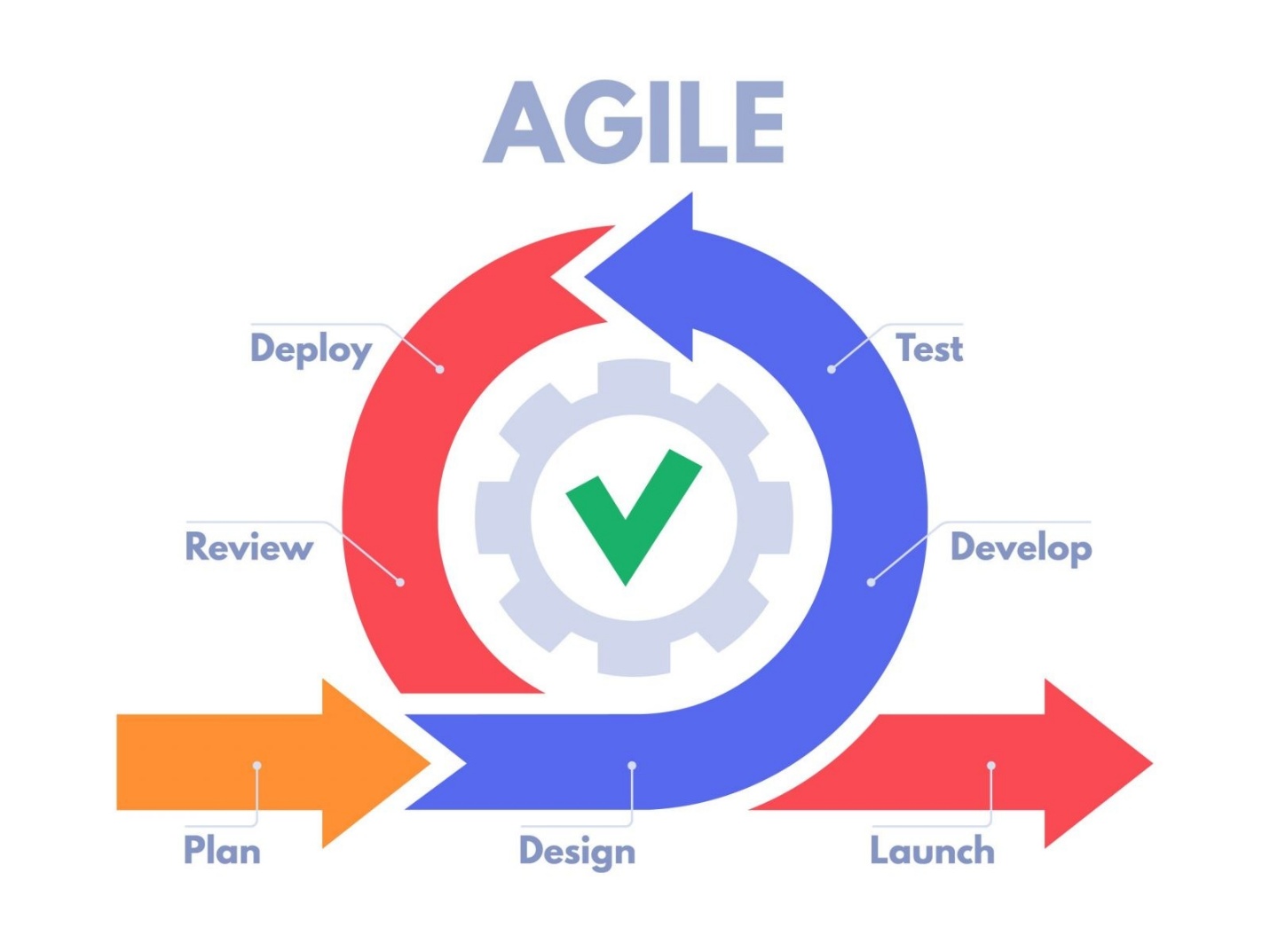
****

**Mô hình thác nước là một trong những mô hình quy trình phát triển phần mềm cổ điển và đơn giản nhất. Mô hình này được đặt tên theo cách các giai đoạn của quy trình được sắp xếp tuần tự từ trên xuống dưới, giống như dòng nước chảy từ đỉnh xuống chân của thác nước. Các giai đoạn của mô hình thác nước bao gồm:**

* **Phân tích yêu cầu: Là giai đoạn thu thập và xác định các yêu cầu của khách hàng và người dùng cho sản phẩm phần mềm. Kết quả của giai đoạn này là tài liệu yêu cầu phần mềm (Software Requirement Specification - SRS).**
* **Thiết kế: Là giai đoạn xây dựng kiến trúc và thiết kế chi tiết cho sản phẩm phần mềm, bao gồm các thành phần, giao diện, thuật toán, cấu trúc dữ liệu, ... Kết quả của giai đoạn này là tài liệu thiết kế phần mềm (Software Design Document - SDD).**
* **Lập trình: Là giai đoạn viết mã nguồn cho sản phẩm phần mềm dựa trên tài liệu thiết kế. Kết quả của giai đoạn này là mã nguồn hoàn chỉnh và có thể biên dịch được.**
* **Kiểm thử: Là giai đoạn kiểm tra chất lượng của sản phẩm phần mềm, bao gồm kiểm tra các chức năng, hiệu năng, bảo mật, khả năng sử dụng,... Kết quả của giai đoạn này là danh sách các lỗi và khuyết điểm cần khắc phục.**
* **Bảo trì: Là giai đoạn duy trì và cập nhật sản phẩm phần mềm sau khi đã triển khai cho khách hàng và người dùng. Các hoạt động trong giai đoạn này bao gồm sửa lỗi, nâng cấp tính năng, tối ưu hóa hiệu suất, ...**

**Ưu điểm của mô hình thác nước là rõ ràng, dễ hiểu và dễ quản lý. Nhược điểm của mô hình này là khó thích ứng với những thay đổi trong yêu cầu của khách hàng và người dùng, do không có sự giao tiếp liên tục giữa các bên liên quan. Ngoài ra, việc kiểm thử chỉ được thực hiện ở cuối quy trình khiến cho việc phát hiện và sửa lỗi trở nên khó khăn và tốn kém.**

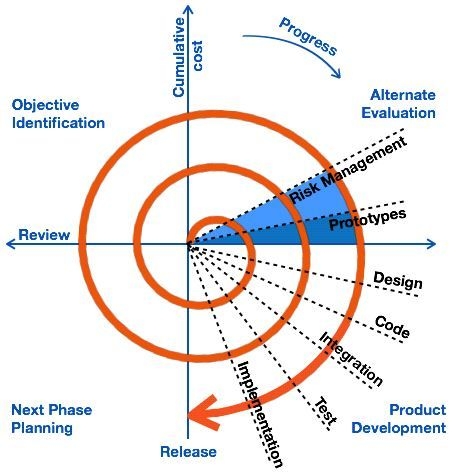
**Mô hình Agile - Agile Model**

****

**Mô hình Agile là một trong những mô hình quy trình phát triển phần mềm hiện đại và linh hoạt nhất. Mô hình này được đặt tên theo nguyên tắc Agile, là một tập hợp các giá trị và nguyên tắc nhằm tạo ra các sản phẩm phần mềm chất lượng, đáp ứng nhanh chóng và linh hoạt các yêu cầu thay đổi của khách hàng và người dùng. Mô hình Agile không có một cấu trúc cụ thể, mà được thực hiện theo các chu kỳ lặp (iteration) ngắn.**

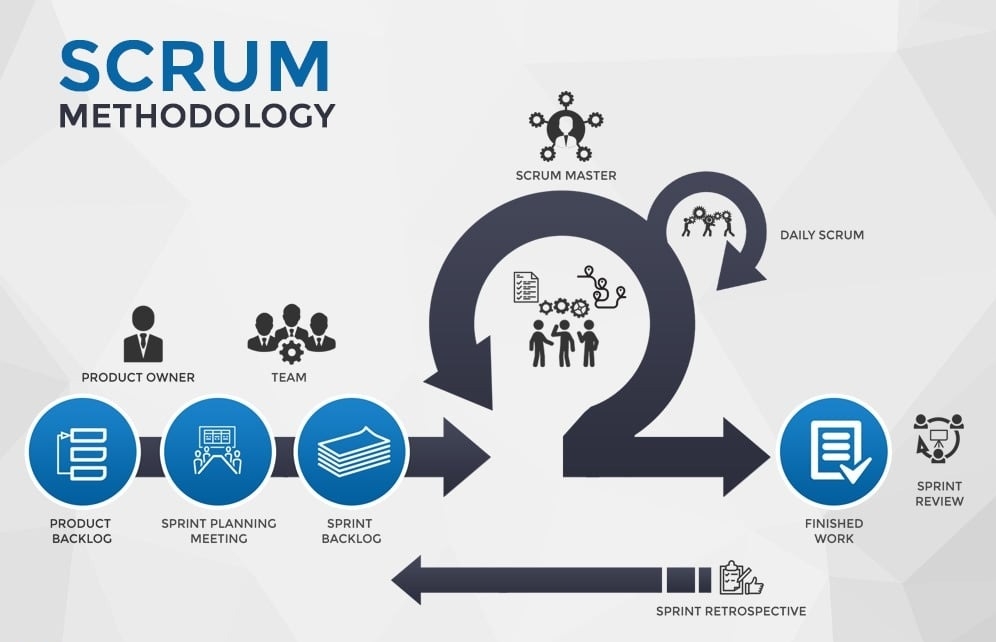
**Ưu điểm của mô hình Agile là linh hoạt, nhanh chóng và tiết kiệm chi phí. Nhược điểm của mô hình này là khó kiểm soát, yêu cầu sự tham gia tích cực của khách hàng và người dùng, và có thể dẫn đến sự thiếu ổn định của sản phẩm phần mềm.**

**Mô hình xoắn ốc - Boehm Spiral Model**

****

**Mô hình xoắn ốc là một mô hình quy trình phát triển phần mềm do Barry Boehm đề xuất vào năm 1986. Mô hình này kết hợp các ưu điểm của mô hình tuần tự (waterfall) và mô hình lặp (iterative). Mỗi vòng xoắn ốc gồm bốn giai đoạn: xác định mục tiêu, phân tích rủi ro, thực hiện và đánh giá. Mỗi vòng xoắn ốc tương ứng với một giai đoạn của dự án, từ khảo sát nhu cầu đến triển khai và bảo trì. Mô hình xoắn ốc cho phép xử lý các rủi ro trong quá trình phát triển, thích nghi với các thay đổi của yêu cầu và cải tiến liên tục sản phẩm.**

**Mô hình Scrum - Scrum Model**

****

**Mô hình Scrum là một mô hình quy trình phát triển phần mềm thuộc dạng Agile (linh hoạt). Mô hình Scrum gồm ba vai trò chính: Product Owner (chủ sản phẩm), Scrum Master (quản lý dự án) và Development Team (nhóm phát triển). Mỗi dự án được chia thành các sprint (chu kỳ), thường kéo dài từ hai tuần đến một tháng.**

**Trong mỗi sprint, nhóm phát triển sẽ hoàn thành một số tính năng của sản phẩm theo danh sách ưu tiên (backlog) do Product Owner xác định. Mỗi ngày, nhóm phát triển sẽ tổ chức một cuộc họp ngắn gọn (daily stand-up) để báo cáo tiến độ, gặp phải vấn đề gì và cần giúp đỡ gì. Cuối mỗi sprint, nhóm phát triển sẽ thực hiện kiểm thử, trình bày sản phẩm cho khách hàng và người dùng để nhận phản hồi và cải tiến.**

**Mô hình Scrum giúp tăng hiệu quả làm việc của nhóm, tạo ra sản phẩm chất lượng cao và thỏa mãn khách hàng.**

**\*Kết luận:**

**Quy trình phát triển phần mềm là một quá trình quan trọng để tạo ra các sản phẩm phần mềm chất lượng cao. Tùy vào từng loại dự án, ngân sách, thời gian và yêu cầu của khách hàng, chúng ta có thể lựa chọn một mô hình quy trình phát triển phần mềm phù hợp.**

**DEVOPS tất cả mọi thứ liên quan**

**1 Devops là gì, vai trò của devops trong dự án và trong quy trình phát triển phần mềm.**

**a)**

**-Khái niệm:**

**DevOps viết tắt của Development (phát triển) và Operations (vận hành), là một phương pháp làm việc nhằm tối ưu hóa quá trình phát triển và triển khai phần mềm. Thay vì hoạt động riêng lẻ, DevOps tạo ra sự liên kết chặt chẽ giữa đội ngũ lập trình viên và nhóm vận hành, giúp cải thiện cả hiệu suất và chất lượng của sản phẩm.**

**Phương pháp này không chỉ tập trung vào công nghệ mà còn nhấn mạnh sự thay đổi trong văn hóa làm việc, sử dụng các công cụ tự động hóa và thúc đẩy quy trình cộng tác liên tục. Với DevOps, các doanh nghiệp có thể tăng tốc chu kỳ phát triển phần mềm, đảm bảo độ tin cậy cao và mang lại giá trị nhanh chóng cho khách hàng.**

**-Lợi ích của Devops**

**DevOps mang lại nhiều lợi ích vượt trội, không chỉ cho đội ngũ kỹ thuật mà còn cho doanh nghiệp và người dùng cuối. Những lợi ích nổi bật của DevOps bao gồm:**

**1. Tăng tốc độ phát triển và triển khai phần mềm**

**Nhờ các quy trình tự động hóa như Tích hợp liên tục (**[**CI**](https://viettelidc.com.vn/tin-tuc/ci-cd-la-gi)**) và Triển khai liên tục (CD), DevOps giúp rút ngắn thời gian từ khâu phát triển đến khi sản phẩm được đưa vào sử dụng. Điều này đặc biệt hữu ích trong các thị trường có tính cạnh tranh cao, ưu tiên tốc độ triển khai tính năng mới.**

**2. Nâng cao chất lượng sản phẩm**

**DevOps tích hợp kiểm thử tự động vào quy trình làm việc, giúp phát hiện và khắc phục lỗi ngay từ giai đoạn đầu. Ngoài ra, việc giám sát liên tục sau khi triển khai giúp nhanh chóng xử lý các vấn đề phát sinh, đảm bảo sản phẩm vận hành ổn định và đáp ứng tốt nhu cầu người dùng.**

**3. Tăng cường hiệu quả làm việc và tối ưu hóa nguồn lực**

**Các công cụ DevOps như Jenkins,**[**Kubernetes**](https://viettelidc.com.vn/tin-tuc/kubernetes-la-gi-chuc-nang-tu-dong-hoa)**,**[**Docker**](https://viettelidc.com.vn/tin-tuc/docker-la-gi-va-tai-sao-chung-ta-lai-su-dung-no-khi-phat-trien-phan-mem)**,... giúp tự động hóa những công việc lặp đi lặp lại, giảm tải cho đội ngũ kỹ thuật. Nhờ đó, nhân viên có thể tập trung vào các nhiệm vụ sáng tạo và quan trọng hơn, tối ưu hóa hiệu quả sử dụng nguồn lực.**

**4. Thúc đẩy sự hợp tác và giao tiếp giữa các đội ngũ**

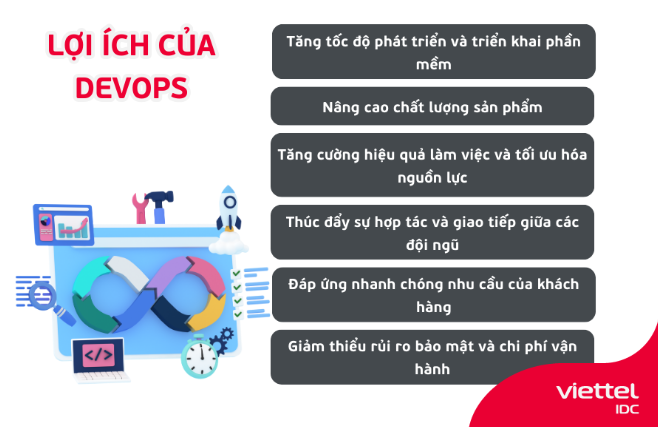
**DevOps tạo ra môi trường làm việc liên kết chặt chẽ giữa nhóm phát triển và nhóm vận hành. Các rào cản giao tiếp được loại bỏ, giúp hai bên dễ dàng chia sẻ thông tin, hiểu rõ mục tiêu chung và phối hợp hiệu quả để đạt được kết quả tốt nhất.**

**5. Giảm thiểu rủi ro bảo mật và chi phí vận hành**

**Quy trình DevOps với khả năng kiểm thử và giám sát liên tục giúp phát hiện sớm các vấn đề, giảm thiểu rủi ro về bảo mật và sự cố hệ thống. Đồng thời, việc tự động hóa giảm thiểu các lỗi do con người gây ra cũng giúp tiết kiệm chi phí và nguồn lực cho doanh nghiệp.**

**6. Đáp ứng nhanh chóng nhu cầu của khách hàng**

**Với DevOps, các doanh nghiệp có thể triển khai và cập nhật sản phẩm một cách linh hoạt, nhanh chóng thích nghi với thay đổi trong nhu cầu của thị trường và khách hàng. Điều này giúp tăng cường trải nghiệm người dùng, đồng thời nâng cao lợi thế cạnh tranh cho doanh nghiệp.**

****

**-Cách thức hoạt động của Devops**

**DevOps hoạt động dựa trên sự kết hợp chặt chẽ giữa nhóm phát triển (Development) và nhóm vận hành (Operations) cùng với việc áp dụng các công cụ tự động hóa và tích hợp liên tục. Quy trình DevOps thường được thực hiện qua các bước chính sau:**

**1. Lập kế hoạch (Planning)**

**Bước đầu tiên, doanh nghiệp cần xác định các mục tiêu, yêu cầu và tính năng của phần mềm. Các nhóm liên quan sẽ hợp tác để xây dựng kế hoạch phát triển và triển khai chi tiết, đồng thời thiết lập các chỉ số đo lường hiệu quả (KPIs).**

**2. Viết mã (Development)**

**Các lập trình viên tiến hành viết mã và phát triển tính năng theo kế hoạch. Giai đoạn này áp dụng các công cụ quản lý mã nguồn như Git để đảm bảo kiểm soát phiên bản, hợp tác nhóm và tích hợp mã một cách hiệu quả.**

**3. Tích hợp và kiểm thử (Integration & Testing)**

**Mã nguồn từ các lập trình viên được tích hợp liên tục (CI) vào một hệ thống trung tâm. Quy trình kiểm thử tự động được kích hoạt để phát hiện lỗi, đảm bảo chất lượng phần mềm ngay từ giai đoạn đầu.**

**4. Triển khai (Deployment)**

**Sản phẩm sau khi kiểm thử thành công sẽ được triển khai tự động (CD) đến các môi trường khác nhau, từ thử nghiệm đến sản xuất. DevOps đảm bảo việc triển khai diễn ra nhanh chóng, hạn chế tối đa rủi ro.**

**5. Vận hành và giám sát (Operations & Monitoring)**

**Sau khi triển khai, các nhóm DevOps theo dõi hiệu suất ứng dụng thông qua các công cụ giám sát như Prometheus hoặc Grafana. Dữ liệu thu thập được sử dụng để phát hiện sớm các vấn đề và cải thiện hệ thống.**

**6. Phản hồi và cải tiến liên tục (Feedback & Continuous Improvement)**

**Dựa trên thông tin từ khách hàng và kết quả giám sát, nhóm DevOps thực hiện điều chỉnh và cải tiến sản phẩm. Quy trình này khép kín, tạo nên chu kỳ cải tiến không ngừng để đảm bảo sản phẩm luôn đáp ứng nhu cầu thị trường.**

****

**b)Vai trò**

**\* Trong dự án:**

**Devops giúp đảm bảo sản phẩm được phát triển, kiểm khử, triển khai và vận hành một cách hiệu quả, cụ thể:**

* **Tăng tốc độ phát triển:giảm thời gian giữa các lần phát hành bằng cách tự động hóa quy trình build, test,và triển khai.**
* **Đảm bảo chất lượng: Kết hợp CI/CD để kiểm tra chất lượng phần mềm liên tục.**
* **Tối ưu tài nguyên: Sử dụng Infrastructure as code (IaC) để quản lý hạ tầng dễ dàng hơn, giảm chi phí vận hành.**
* **Giảm rủi ro: Phát hiện lỗi sớm thông qua giám sát liên tục, giúp dự án tránh được những vấn đề nghiêm trọng khi triển khai.**
* **Tăng cường cộng tác: Xây dựng môi trường làm việc nơi DEV và OPS phối hợp chặt chẽ tránh tình trạng “Dev xong rồi quăng cho Ops chạy”.**

**\*Trong quy trình triển khai phần mềm:**

**-Giai đoạn1: Lập kế hoạch (planning)**

**+Hỗ trợ lập kế hoạch phát triển phần mềm với mô hình Agile.**

**+Kết nối đội Dev vs đội Ops để hiểu rõ yêu cầu về hạ tầng và vận hành.**

**+Thiết lập các công cụ quản lý như Jira, Trello để theo dõi tiến độ.**

**-Giai đoạn 2: Phát triển Development**

**+ Hỗ trợ thiết lập môi trường phát triển đồng bộ giữa các lập trình viên.**

**+ Tích hợp kiểm khử sớm trong lập trình viết Code thông qua CI/CD**

**+ Sử dụng container ( Docker, kubernetes) để đảm bảo ứng dụng chạy được trên nhiều môi trường.**

**-Giai đoạn 3: Kiểm khử ( testing)**

**+ Tự động hóa kiểm khử ( Unit test, Integration Test, UI Test) để phát hiện lỗi sớm.**

**+ Mô phỏng môi trường Production để kiểm tra hiệu suất bảo mật.**

**+ Xây dựng pipeline CI/CD để đảm bảo phần mềm luôn được kiểm khử trước khi phát hành.**

**-Giai đoạn 4: Triển khai ( Development)  
 + Triển khai nhanh chóng và an toàn thông qua CI/CD**

**+ Áp dụng các phương pháp như Canary Deplopment, Blue-green Deplopment để giảm rủi ro khi release phiên bản mới.**

**+ Sử dụng Infrastructure as Code (IaC) để quản lý và cấu hình hạ tầng tự động.**

**-Giai đoạn 5: Vận hành( Operation)**

**+Giám sát hệ thống công bằng các công cụ như prometheus, Grafana, ELK stack.**

**+ Tối ưu hiệu suất, phát hiện lỗi nhanh chóng nhờ Loggin và Monioring.**

**+ Đảm bảo bảo mật thông qua các chính sách kiểm soát truy cập, các mã hóa dữ liệu.**

**-Giai đoạn 6:Phản hồi ( feedback)**

**+ Thu thập phản hồi từ ngườid dùng và giảm sát hệ thống liên tục cải thiện sản phẩm.**

**+ Phối hợp chặt chẽ giữa các Dev, Ops, QA để nhanh chóng sửa lỗi và phát hành bản cập nhập**

* **Devops giúp giải quyết công việc nhanh chóng không chỉ từ một vị trí mà cả một công ty hay một doanh nghiệp: rút ngắn thời gian vòng đời phát triển phần mềm, nâng cao chất lượng và tối ưu hiệu suất, Trong một dự án, Devops giúp giảm xung đột giữa các nhóm, tắng tốc triển khai, giảm lỗi. Trong quy trình phát triển phần mềm, Devops giúp tự động hóa, tối ưu hóa hạ tầng và trải nghiệm của người dùng.**

**Vòng đời DevOps( Devops lifecycle)**

**DevOps là một phương pháp cho phép một nhóm duy nhất xử lý toàn bộ vòng đời ứng dụng, bao gồm phát triển, thử nghiệm, phát hành, triển khai, vận hành, hiển thị và lập kế hoạch. Nó là sự kết hợp của các thuật ngữ "Dev" (để phát triển) và "Ops" (cho hoạt động). Chúng tôi có thể tăng tốc độ cung cấp các ứng dụng và dịch vụ của một doanh nghiệp với sự hỗ trợ của DevOps. Amazon, Netflix và các doanh nghiệp khác đều đã áp dụng DevOps một cách hiệu quả để cải thiện trải nghiệm khách hàng của họ.**

**Vòng đời DevOps là tập hợp các giai đoạn bao gồm**[**DevOps**](https://www.geeksforgeeks.org/introduction-to-devops/)**để tham gia vào các nhiệm vụ nhóm [Phát triển](https://www.geeksforgeeks.org/web-development/)và Vận hành để phân phối chương trình phần mềm nhanh hơn. DevOps tuân theo các kỹ thuật tích cực bao gồm mã, xây dựng, thử nghiệm, phát hành, triển khai, vận hành, hiển thị và lập kế hoạch. Vòng đời DevOps tuân theo một loạt các giai đoạn như phát triển không ngừng nghỉ, tích hợp không ngừng, thử nghiệm không ngừng, giám sát không ngừng và phản hồi không ngừng. Mỗi phân đoạn của vòng đời DevOps có liên quan đến một số thiết bị và khoa học ứng dụng để có được quy trình. Một số công cụ thường được sử dụng là mã nguồn mở và được thực hiện chủ yếu dựa trên các yêu cầu của doanh nghiệp thương mại. Vòng đời DevOps rất dễ dàng để thao tác và nó giúp phân phối thỏa đáng.**

**7 C của DevOps**

1. **Phát triển liên tục**
2. **Tích hợp liên tục**
3. **Kiểm tra liên tục**
4. **Triển khai liên tục/Phân phối liên tục**
5. **Giám sát liên tục**
6. **Phản hồi liên tục**
7. **Hoạt động liên tục**

**1. Phát triển liên tục**

**Trong Continuous Development, mã được viết bằng các bit nhỏ, liên tục thay vì tất cả cùng một lúc, Continuous Development rất quan trọng trong DevOps vì điều này cải thiện hiệu quả mỗi khi một đoạn mã được tạo, nó được kiểm tra, xây dựng và triển khai vào sản xuất. Phát triển liên tục nâng cao tiêu chuẩn của mã và hợp lý hóa quá trình sửa chữa các sai sót, lỗ hổng và lỗi. Nó tạo điều kiện cho các nhà phát triển khả năng tập trung vào việc tạo mã chất lượng cao.**

**2. Tích hợp liên tục**

**Tích hợp liên tục có thể được giải thích chủ yếu trong 4 giai đoạn trong DevOps. Chúng như sau:**

1. **Lấy mã nguồn từ SCM**
2. **Xây dựng mã**
3. **Đánh giá chất lượng mã**
4. **Lưu trữ các tạo phần mềm xây dựng**

**Các giai đoạn được đề cập ở trên là luồng Tích hợp liên tục và chúng ta có thể sử dụng bất kỳ công cụ nào phù hợp với yêu cầu của mình trong từng giai đoạn và các công cụ phổ biến nhất là**[**GitHub**](https://www.geeksforgeeks.org/introduction-to-github/)**để quản lý mã nguồn (SCM) khi nhà phát triển phát triển mã trên máy cục bộ của mình, anh ta đẩy nó đến kho lưu trữ từ xa là GitHub từ đây, người đang có quyền truy cập có thể kéo, sao chép và có thể thực hiện các thay đổi cần thiết đối với mã. Từ đó bằng cách sử dụng**[**Maven**](https://www.geeksforgeeks.org/introduction-apache-maven-build-automation-tool-java-projects/)**, chúng ta có thể xây dựng chúng vào gói cần thiết (war, jar, ear) và có thể kiểm tra các trường hợp Junit.**[**SonarQube**](https://www.geeksforgeeks.org/software-testing-security-testing-tools/)**thực hiện đánh giá chất lượng mã, nơi nó sẽ đo lường chất lượng của mã nguồn và tạo báo cáo ở dạng HTML hoặc PDF. Nexus để lưu trữ các tạo tác xây dựng sẽ giúp chúng ta lưu trữ các [tạo tác](https://www.geeksforgeeks.org/artifact-software-development/) được xây dựng bằng cách sử dụng Maven và toàn bộ quá trình này đạt được bằng cách sử dụng công cụ Tích hợp liên tục**[**Jenkins**](https://www.geeksforgeeks.org/what-is-jenkins/)**.**

**3. Kiểm tra liên tục**

**Bất kỳ công ty nào cũng có thể triển khai thử nghiệm liên tục bằng cách sử dụng các phương pháp agile và DevOps. Tùy thuộc vào nhu cầu của mình, chúng tôi có thể thực hiện kiểm thử liên tục bằng cách sử dụng các công cụ kiểm thử tự động hóa như Testsigma,**[**Selenium**](https://www.geeksforgeeks.org/selenium-basics-components-features-uses-and-limitations/)**, [LambdaTest](https://www.geeksforgeeks.org/tag/lambdatest/), v.v. Với những công cụ này, chúng ta có thể kiểm tra mã của mình và ngăn chặn sự cố và mùi mã, cũng như kiểm tra nhanh chóng và thông minh hơn. Với sự hỗ trợ của một nền tảng tích hợp liên tục như Jenkins, toàn bộ quy trình có thể được tự động hóa, đây là một lợi ích bổ sung khác.**

**4. Triển khai liên tục / Phân phối liên tục**

**Triển khai liên tục: [Triển khai liên tục](https://www.geeksforgeeks.org/overview-of-continuous-deployment/)là quá trình tự động triển khai ứng dụng vào môi trường sản xuất khi ứng dụng đã hoàn thành các giai đoạn thử nghiệm và xây dựng. Ở đây, chúng tôi sẽ tự động hóa mọi thứ từ lấy mã nguồn của ứng dụng đến triển khai nó.**

**Phân phối liên tục: [Phân phối liên tục](https://www.geeksforgeeks.org/ci-cd-continuous-integration-and-continuous-delivery/) là quá trình triển khai ứng dụng vào máy chủ sản xuất theo cách thủ công khi nó đã hoàn thành các giai đoạn thử nghiệm và xây dựng. Ở đây, chúng tôi sẽ tự động hóa các quy trình tích hợp liên tục, tuy nhiên, vẫn cần sự tham gia thủ công để triển khai nó vào môi trường sản xuất.**

**5. Giám sát liên tục**

**Vòng đời DevOps không hoàn chỉnh nếu không có Giám sát liên tục. Giám sát liên tục có thể đạt được với sự trợ giúp của Prometheus và Grafana, chúng tôi có thể liên tục giám sát và có thể nhận thông báo trước khi có bất kỳ sự cố nào xảy ra với sự trợ giúp của Prometheus, chúng tôi có thể thu thập nhiều biện pháp hiệu suất, bao gồm sử dụng CPU và bộ nhớ, lưu lượng mạng, thời gian phản hồi ứng dụng, tỷ lệ lỗi và các biện pháp khác. Grafana giúp bạn có thể biểu diễn trực quan và theo dõi dữ liệu từ chuỗi thời gian, chẳng hạn như sử dụng CPU và bộ nhớ.**

**6. Phản hồi liên tục**

**Khi ứng dụng được phát hành ra thị trường, người dùng cuối sẽ sử dụng ứng dụng và họ sẽ cung cấp cho chúng tôi phản hồi về hiệu suất của ứng dụng và bất kỳ trục trặc nào ảnh hưởng đến trải nghiệm người dùng sau khi nhận được nhiều phản hồi từ người dùng cuối, nhóm DevOps sẽ phân tích phản hồi do người dùng cuối đưa ra và họ sẽ liên hệ với nhóm nhà phát triển, cố gắng sửa chữa những sai lầm mà họ được thực hiện trong đoạn mã đó Bằng cách này, chúng tôi có thể giảm các lỗi hoặc lỗi mà chúng tôi hiện đang phát triển và có thể tạo ra kết quả hiệu quả hơn nhiều cho người dùng cuối, chúng tôi cũng giảm bất kỳ bước không cần thiết nào để triển khai ứng dụng. Phản hồi liên tục có thể tăng hiệu suất của ứng dụng và giảm lỗi trong mã, giúp người dùng cuối sử dụng ứng dụng một cách suôn sẻ.**

**7. Hoạt động liên tục**

**Chúng tôi sẽ duy trì thời gian hoạt động của ứng dụng cao hơn bằng cách triển khai hoạt động liên tục, điều này sẽ giúp chúng tôi cắt giảm thời gian ngừng hoạt động bảo trì sẽ ảnh hưởng tiêu cực đến trải nghiệm của người dùng cuối. Sản lượng nhiều hơn, chi phí sản xuất thấp hơn và kiểm soát chất lượng tốt hơn là những lợi ích của hoạt động liên tục.**

**\*Các giai đoạn khác nhau của vòng đời DevOps**

1. **Kế hoạch: Các chuyên gia xác định nhu cầu thương mại và thu thập ý kiến của người dùng cuối trong suốt cấp độ này. Trong bước này, họ thiết kế kế hoạch dự án để tối ưu hóa tác động kinh doanh và tạo ra kết quả mong muốn.**
2. **Mã – Trong thời điểm này, mã đang được phát triển. Để đơn giản hóa quy trình thiết kế, nhóm nhà phát triển sử dụng các công cụ DevOps vòng đời và các tiện ích mở rộng như Git để hỗ trợ họ ngăn chặn các vấn đề an toàn và các tiêu chuẩn mã hóa xấu.**
3. **Xây dựng – Sau khi lập trình viên đã hoàn thành nhiệm vụ của mình, họ sử dụng các công cụ như Maven và Gradle để gửi mã đến nguồn mã chung.**
4. **Kiểm tra - Để đảm bảo tính toàn vẹn của phần mềm, trước tiên sản phẩm được chuyển đến nền tảng thử nghiệm để thực hiện các loại sàng lọc khác nhau như kiểm tra khả năng chấp nhận của người dùng, kiểm tra an toàn, kiểm tra tích hợp, kiểm tra tốc độ, v.v., sử dụng các công cụ như JUnit, Selenium, v.v.**
5. **Phát hành – Tại thời điểm này, bản dựng đã được chuẩn bị để triển khai trong môi trường hoạt động. Bộ phận DevOps chuẩn bị các bản cập nhật hoặc gửi một số phiên bản đến sản xuất khi bản dựng đáp ứng tất cả các kiểm tra dựa trên nhu cầu của tổ chức.**
6. **Triển khai – Tại thời điểm này, Cơ sở hạ tầng dưới dạng mã hỗ trợ tạo cơ sở hạ tầng hoạt động và sau đó xuất bản bản dựng bằng các công cụ vòng đời DevOps khác nhau.**
7. **Vận hành – Phiên bản này hiện đã thuận tiện cho người dùng sử dụng. Với các công cụ bao gồm Chef, bộ phận quản lý đảm nhận cấu hình và triển khai máy chủ tại thời điểm này.**
8. **Giám sát – Quy trình làm việc DevOps được quan sát ở cấp độ này tùy thuộc vào dữ liệu thu thập được từ hành vi của người tiêu dùng, hiệu quả ứng dụng và các nguồn khác. Khả năng quan sát môi trường xung quanh hoàn chỉnh hỗ trợ các nhóm xác định các nút thắt cổ chai ảnh hưởng đến hiệu suất của nhóm sản xuất và vận hành.**

**Lợi ích của DevOps (Devops benefits)**

**DevOps có thể tạo ra một môi trường hợp tác nơi các lập trình viên và các nhà điều hành làm việc với nhau và cùng hướng tới mục đích chung. Một cột mốc quan trọng trong quá trình này là việc thực hiện việc tích hợp và triển khai liên tục (**[**continuous integration and continuous delivery – CI / CD**](https://gocit.vn/bai-viet/ci-cd-va-devops-la-gi/)**). Điều này cho phép các team đưa phần mềm ra thị trường nhanh hơn, ít lỗi hơn.**

**Các lợi ích quan trọng của DevOps:**

* **Dự đoán: Các bản release mới có tỷ lệ thất bại thấp hơn.**
* **Bảo trì: Phục hồi các bản release một cách dễ dàng khi có vấn đề.**
* **Khôi phục: Khôi phục các phiên bản trước khi cần thiết.**
* **Chất lượng cao hơn: Kết hợp các vấn đề về cơ sở hạ tầng giúp cải thiện chất lượng phát triển ứng dụng.**
* **Thời gian: Thời gian được giảm lên đến 50%.**
* **Giảm rủi ro: Kết hợp bảo mật vào vòng đời phần mềm giúp giảm rủi ro.**
* **Tiết kiệm chi phí: Tiết kiệm chi phí hơn các phương pháp khác.**
* **Khả năng phục hồi: Hệ thống ổn định, an toàn hơn và các thay đổi có thể kiểm tra được.**
* **Chia nhỏ codebase: DevOps dựa trên phương pháp lập trình nhanh, hỗ trợ chia nhỏ các codebase thành các phần nhỏ, dễ quản lý hơn.**

**So sánh giữa DevOps, Agile và phương pháp truyền thống**

**DevOps có mối liên quan gì và như thế nào đến các phương pháp hay mô hình khác trong lĩnh vực IT, đặc biệt là**[**Agile**](https://opensource.com/article/20/2/devops-vs-agile)**và Waterfall?**

**Phương pháp Agile là tập hợp các nguyên tắc, giá trị và phương pháp dành cho quy phát triển ra phần mềm. Ví dụ: Nếu như bạn có ý tưởng và muốn chuyển đổi ý tưởng thành phần mềm, bạn có thể tận dụng các nguyên tắc và giá trị Agile. Nhưng phần mềm đó chỉ có thể hoạt động trong trong một phạm vi nhỏ hoăc trong môi trường testing. Bạn cần một cách thức để triển khai phần mềm vào môi trường production một cách đơn giản, nhanh chóng và an toàn. Khi đó các công cụ và kỹ thuật DevOps được phát huy tác dụng. Phương pháp phát triển phần mềm Agile tập trung vào các quy trình phát triển, nhưng DevOps sẽ giúp phát triển và triển khai theo cách thức an toàn và đáng tin cậy nhất.**

***Phương pháp Agile. Ảnh: Edureka.co***

**So sánh mô hình waterfall truyền thống với DevOps là cách để hiểu được những lợi ích mà DevOps mang lại. Trong ví dụ sau đây, giả định rằng ứng dụng ​​sẽ hoạt động sau bốn tuần và coding đã hoàn thành được 85%. Đây là lần đầu tiên ứng dụng này được launch và quy trình mua server để triển khai code vừa được bắt đầu.**

|  |  |
| --- | --- |
| **Quy trình truyền thống** | **DevOps** |
| **Sau khi đặt hàng các server, đội ngủ phát triển sẽ tiến hành quá trình kiểm thử. Team điều hành làm các công việc giấy tờ thủ tục cần thiết để triển khai cơ sở hạ tầng.** | **Sau khi đặt hàng các server, team phát triển và điều hành cùng làm việc với nhau để hoàn tất các thủ tục cần thiết để có thể setup các server.  Điều này giúp cho yêu cầu về hạ tầng được hiểu một cách cặn kẽ nhất.** |
| **Chi tiết về failover, dự phòng, các vị trí trung tâm dữ liệu và các yêu cầu về lưu trữ bị sai lệch, vì không có input nào có sẵn từ team phát triển có kiến ​​thức chuyên sâu về ứng dụng.** | **Chi tiết về failover, dự phòng, khắc phục thảm họa, các vị trí trung tâm dữ liệu và yêu cầu lưu trữ được hiểu một cách chính xác do các input từ team phát triển.** |
| **Team điều hành không biết về hoạt động của team phát triển. Team điều hành phát triển kế hoạch giám sát dựa trên sự hiểu biết của họ.** | **Team điều hành biết rõ hoạt động mà nhóm phát triển đang thực hiện. Cả hai team tương tác lẫn nhau và cùng nhau phát triển một kế hoạch giám sát phục vụ cho cả IT và kinh doanh. Họ cũng chia sẻ các công cụ giám sát hiệu suất ứng dụng (Application Performance Monitoring – APM).** |
| **Trước khi go-live, kiểm tải (load test) làm hỏng ứng dụng, làm chậm quá trình release.** | **Trước khi go-live, kiểm tải (load test) làm chậm ứng dụng. Team phát triển nhanh chóng sửa lỗi và ứng dụng được release đúng hạn.** |