Chuyên đề DevSecOps

BẢO MẬT VÀ VẬN HÀNH HỆ THỐNG

CUSC G2B - phucuong@ctu.edu.vn

NỘI DUNG

1. Giới thiệu DevOps, CI/CD

DevOps là gì?

DevOps là văn hóa làm việc mà ở đó đề cao sự hợp tác, hướng đến việc giúp cho giai đoạn phát triển, vận hành có thể xích lại gần hơn.

DevOps là **sự kết hợp của 2 khái niệm Development và Operations** – là 2 giai đoạn chính của một Software Development Life Cycle (chu trình phát triển phần mềm):

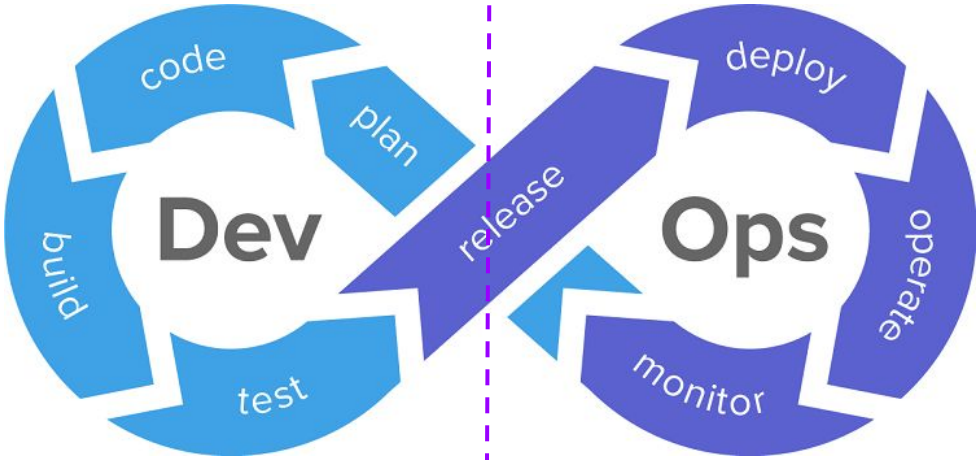
- **Development**: Giai đoạn phát triển dự án, phần mềm như các công việc QA (Quality Assurance), QC (Quality Control), developer, designer, ...

- **Operations**: Giai đoạn vận hành của dự án bao gồm sự tham gia của system administrator, system engineer, release engineer, operation executive, network engineer, DBA, ...

DevOps là gì?

Phát triển (Development)

1. Plan
2. Code
3. Build
4. Test



Chu trình hoạt động của DevOps
(từ giai đoạn phát triển đến vận hành)

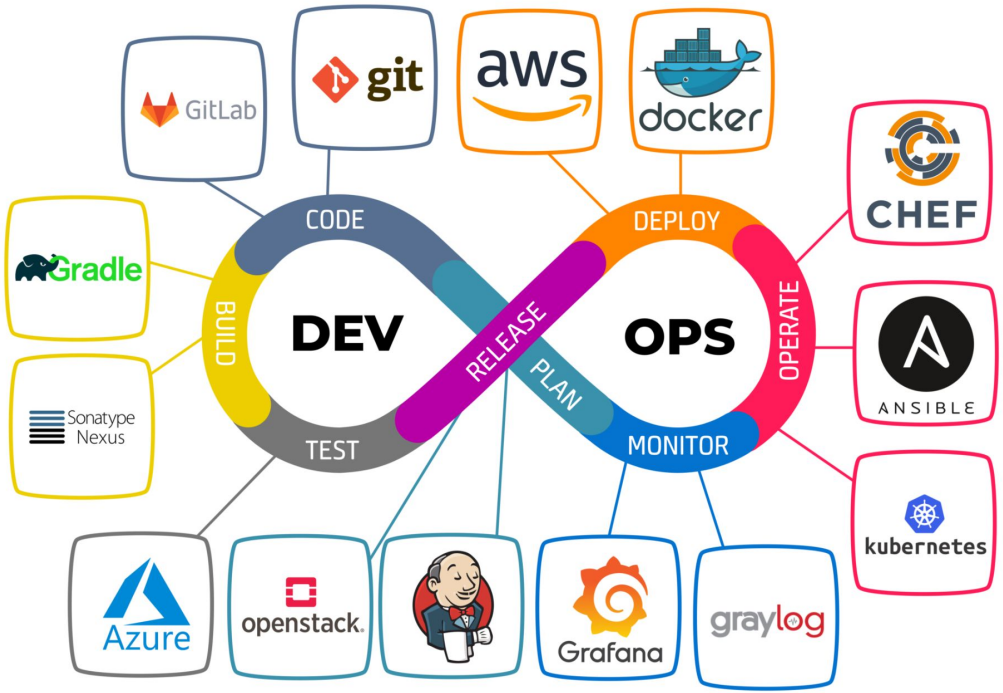
Vận hành (Operations)

1. Release
2. Deploy
3. Operate
4. Monitor

DevOps là gì?

Các tools sử dụng

- OpenStack, Jenkins
- Git, GitLab, GitHub
- Gradle, Sonatype Nexus
- Azure Test
- AWS, Docker
- Chef, Ansible, Kubernetes
- Graylog, Grafana
- ...



Chu trình hoạt động của DevOps
(từ giai đoạn phát triển đến vận hành)

Quy trình thủ công (chưa có CI/CD)

Lập trình viên thông thường sẽ cần làm các công việc:

Code -> build -> zip -> upload server -> test/check

1. Developer code xong -> build trên máy tính/laptop cá nhân -> đợi chờ code build hoàn tất.
2. Sau đó nén (zip) lại source build -> truy cập vào server -> upload code lên server -> chờ đợi code upload hoàn tất.
3. Sau đó tiến hành test (tự test hoặc thông báo kiểm định test) xem đã vận hành được hay chưa?

Bất lợi của quy trình thủ công:

- Nếu có bất kỳ sai sót (upload thiếu file, code lỗi, code thiếu chức năng...) thì toàn bộ công việc trên phải làm lại từ đầu.
- Cũng có khả năng developer phát triển chức năng mới làm ảnh hưởng/hỏng các chức năng cũ -> chỉ có khi deploy mới phát hiện ra.

Quy trình tích hợp & triển khai tự động (CI / CD)

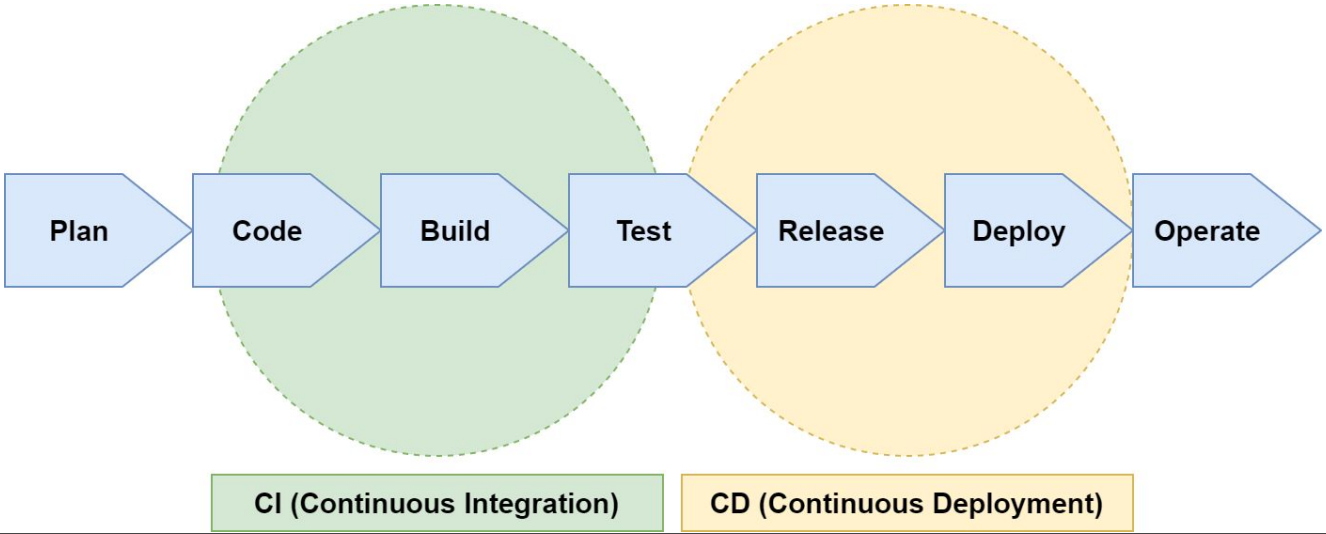
Khái niệm CI/CD thường đề cập đến việc **tự động hóa trong quy trình phát triển phần mềm và chuyển giao sản phẩm**, giúp cho việc tích hợp diễn ra nhanh hơn và sản phẩm hoàn thiện được chuyển đến người dùng trong thời gian ngắn nhất.

Trong đó:
- CI là viết tắt của **Continuous Integration (tích hợp liên tục)**.
- CD là viết tắt của **Continuous Delivery/Deployment (chuyển giao/triển khai liên tục).**

Quy trình CI/CD hoàn chỉnh

Một **quy trình CI/CD hoàn chỉnh** có thể được hình dung như sau:

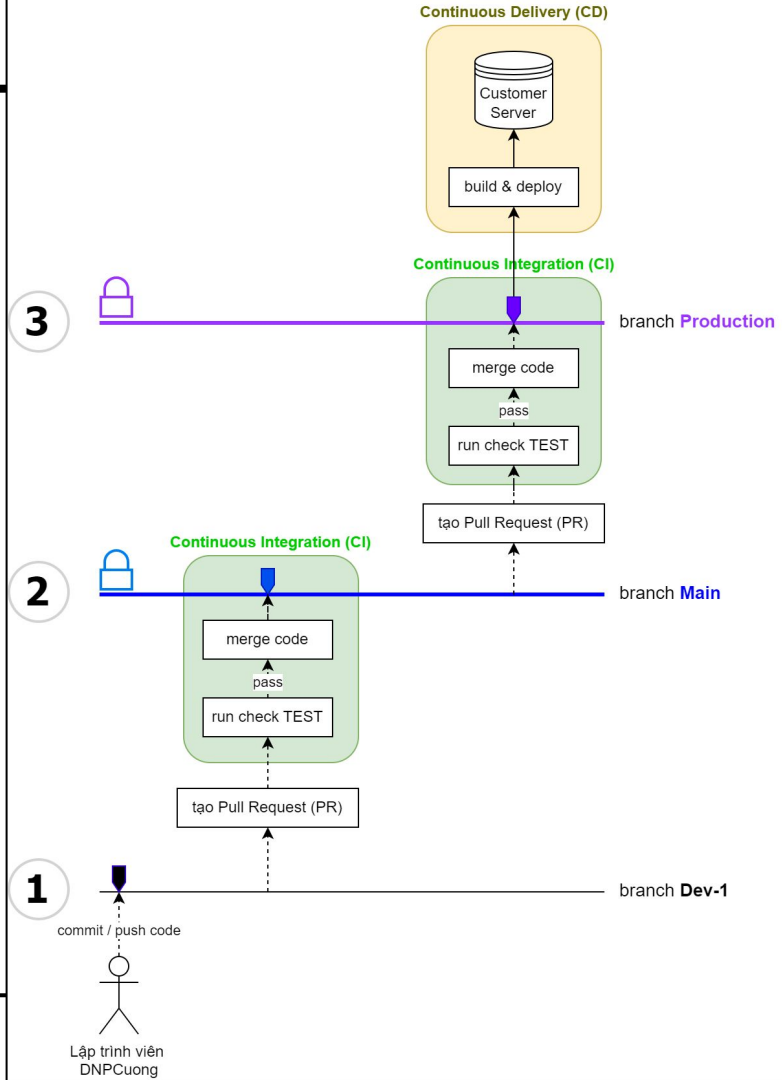
1. Developer commit code (đẩy code lên server).
2. Quy trình **CI**/**CD** sẽ tự động chạy build, chạy test và deploy sản phẩm.
3. Tiếp tục chuyển giao sản phẩm đến người dùng.



Quy trình CI/CD hoàn chỉnh

Ví dụ quy trình CI/CD hoàn chỉnh với GitHub:

1. Lập trình viên commit/push code lên branch “dev-1” mình phụ trách.
2. Tạo PR (pull request) yêu cầu merge code vào branch “main”.
3. Quá trình CI sẽ thực thi kiểm tra.
4. Nếu code pass thì team leader quản lý code sẽ tạo PR merge code qua branch “production”.
5. Quá trình CI sẽ thực thi kiểm tra.
6. Nếu code pass sẽ kích hoạt quá trình CD deploy cho khách hàng.



2. Giới thiệu DevSecOps

DevSecOps là gì?

DevSecOps là viết tắt của **phát triển, bảo mật và vận hành**, là một khuôn khổ **tích hợp bảo mật vào tất cả các giai đoạn** của vòng đời phát triển phần mềm.

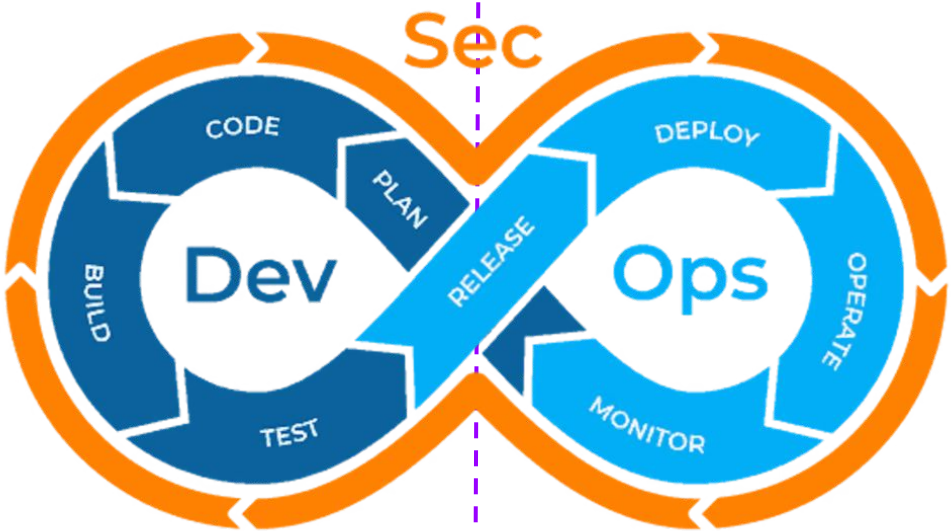
- Các tổ chức áp dụng phương pháp này để **giảm rủi ro phát hành mã có chứa lỗ hổng bảo mật**.
- Thông qua cộng tác, tự động hóa và các quy trình rõ ràng, **các nhóm phát triển dự án có chung trách nhiệm về bảo mật**, **thay vì để đến cuối cùng** khi các vấn đề có thể khó giải quyết và tốn kém hơn nhiều.

DevSecOps là gì?

Bảo mật (Security)

Phát triển (Development)

1. Plan
2. Code
3. Build
4. Test



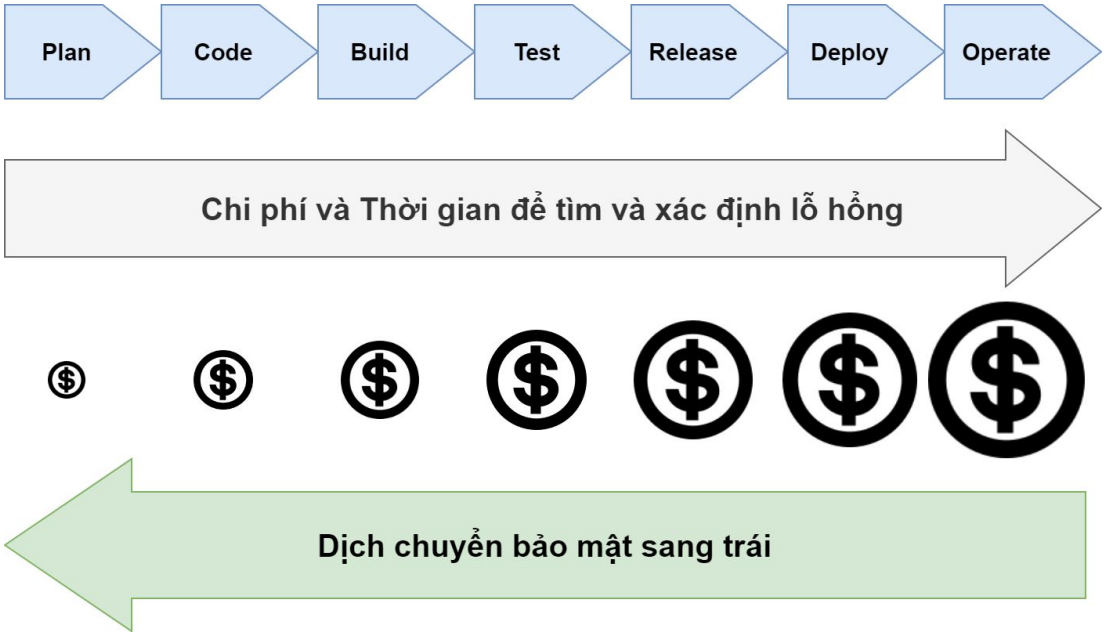
Vận hành (Operations)

1. Release
2. Deploy
3. Operate
4. Monitor

Chu trình hoạt động của Dev**Sec**Ops
(từ giai đoạn phát triển ↔ bảo mật ↔ vận hành)

Nguyên tắc dịch trái DevSecOps

"Càng phát hiện được các lỗ hổng bảo mật càng sớm thì càng tiết kiệm được thời gian và chi phí"



Lợi ích của việc áp dụng DevSecOps

1. Phát hiện sớm lỗ hổng phần mềm
2. Bảo mật chủ động hơn
3. Rút ngắn thời gian tiếp cận thị trường
4. Đảm bảo khả năng tuân thủ quy định, chịu trách nhiệm?
5. Làm việc hiệu quả hơn

Các loại Test bảo mật trong DevSecOps

Có nhiều loại test bảo mật trong DevSecOps, về cơ bản có các nhóm test như sau:

1. SAST Test: (Static Application Security Testing)
2. SCA Test (Software Composition Analysis)
3. DAST Test (Dynamic Application Security Testing)
4. Image Testing

1. SAST Test (Static Application Security Testing)

Là việc scan các file source code (kiểm tra khi chương trình chưa được chạy - not running) nhằm để kiểm tra các lỗ hổng bảo mật có thể xảy ra trong các file source code:
- Xác định các lỗ hổng trong source code, file cấu hình (configuration).
- Kiểm tra các lỗi thông dụng...
- Rà quét các thông tin nhạy cảm được lưu trữ trong source code (secret scanning) như API Keys, Private keys, Password, Authentication File, Token id_rsa, ...
- SQL Injection
- XSS - Cross Site Scripting
- Client and Server request Forgery; Path Traversal...

=> cần có SAST Tool để rà quét các lỗ hổng này.

2. SCA Test (Software Composition Analysis)

Là việc scan check các gói thư viện bên thứ ba (third-party libraries, open source libraries), các framework, ... được sử dụng trong dự án mà có thể đã dính các lỗ hổng bảo mật.

Các thư viện có thể được lấy về từ các kho thư viện:
- Maven (java, android)
- NPM (javascript)
- PIP (python)
- Composer (php)

=> cần có SCA Tool để rà quét các lỗ hổng này.

3. DAST Test (Dynamic Application Security Testing)

Là việc chủ động giả lập các đợt tấn công để tự rà quét các lỗ hổng bảo mật có thể có chương trình đang chạy.

Thường xây dựng các **kịch bản pentest (kiểm thử xâm nhập)** để rà quét chương trình.

=> cần có đội ngũ test giả lập như là hacker để tấn công hệ thống với các hình thức:

- **Blackbox:** không biết gì về hệ thống (tự đoán mò như là hacker).
- **Greybox:** biết chút ít thông tin về hệ thống (như domain, có thông tin tài khoản user cấp thấp...)
- **Whitebox**: đã biết rõ về hệ thống.

4. Image Testing

Là việc rà quét các file ảnh (image) khi triển khai ứng dụng với dạng container (docker / kubernetes)

- Image có sử dụng quyền “root” user không?
- Image có các lỗ hổng về OS package hay không?
- Image có quá nặng hay không? Đã đầy ổ cứng ?

=> cần có các tool rà quét các images.

3. Các công cụ DevSecOps và Testing Security phổ biến

Top 10 lỗi bảo mật phổ biến nhất trong hệ thống web

OWASP open-source project đã tổng hợp 10 lỗi phổ biến nhất trong hệ thống web:

1. Injection (SQL, Javascript, ...)
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities (XEE)
5. Broken Access Control
6. Security Misconfiguration
7. Cross-Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components With Known Vulnerabilities
10. Insufficient Logging And Monitoring

Các công cụ có thể sử dụng trong DevSecOps

1. Husky:
- Cấu hình các hook gits.
2. Commitlint:
- Chuẩn hóa câu chữ "commit message".
3. Retire.JS:
- Kiểm tra các thư viện Javascript có bị các lỗ hổng bảo mật nào không?
- Top 10 các lỗi nguy hiểm về web: "Using Components with Known Vulnerabilities".
4. SonarQube:
- Công cụ dùng để scan quét đảm bảo code convention, rà soát bug, code...
5. ESLint:
- Kiểm tra việc viết code phải theo chuẩn quy định.

Các công cụ có thể sử dụng trong DevSecOps

6. GitLeaks:
- Công cụ check kiểm tra các thông tin nhạy cảm có khả năng lộ trong source code (password, api key, ...).
7. NPM Audit:
- Công cụ kiểm tra lỗ hổng của các thư viện.
8. Burp Suite (công cụ được các bên bảo mật sử dụng Viettel, VnCert...)
- Proxy, chặn request, chỉnh sửa, post request, Interception Proxy, Repeater, Intruder, Decoder, Comparer, Extender, Spider & Discover Content, Scanner...

Danh sách công cụ scan/test

STT	Nhóm	Công cụ
1	Công cụ mã nguồn mở	Công cụ Kali Linux: Mestasploit, Nmap, Firefox addons, Grabber, OWASP ZAP, sqlmap, WebScarab, Wireshark, …
		Công cụ dò quét Framework: Drupal, Joomla, WordPress, …
		Công cụ đánh giá ứng dụng mobile: dex2jar, Android SDK, MobSF, Genymotion, APKInspector, IDB, apktool, frida
		Jmeter – Công cụ kiểm thử hiệu năng
2	Công cụ thương mại	Burpsuite – Công cụ kiểm thử xâm nhập ứng dụng
		Nessus Pro – Đánh giá lỗ hổng bảo mật Ứng dụng & Hệ thống
		Hopper disassembler – Công cụ dịch ngược và debug ứng dụng
		Sn1per Pro – Công cụ dò quét lỗ hổng ứng dụng Web
		Softperfect Network Scanner – Công cụ dò quét hệ thống mạng

4. DEMO

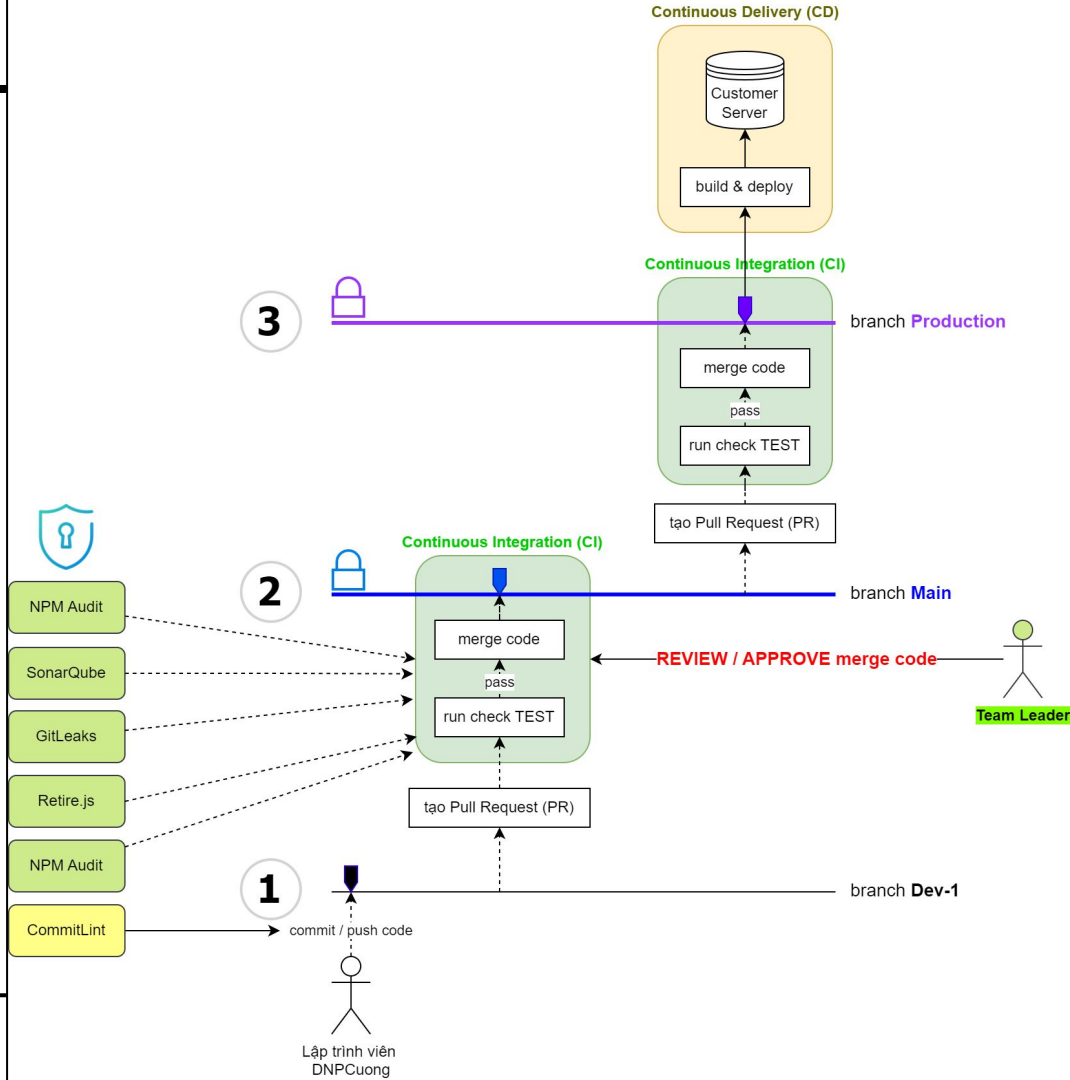
Ví dụ Demo DevSecOps

Ứng dụng demo xây dựng 1 trang web hiển thị tin tức về buổi báo cáo DevSecOps này.

Được viết bằng NextJS, deploy trên GitHub với các cấu hình CI/CD và các bảo mật DevSecOps từ ban đầu dự án.

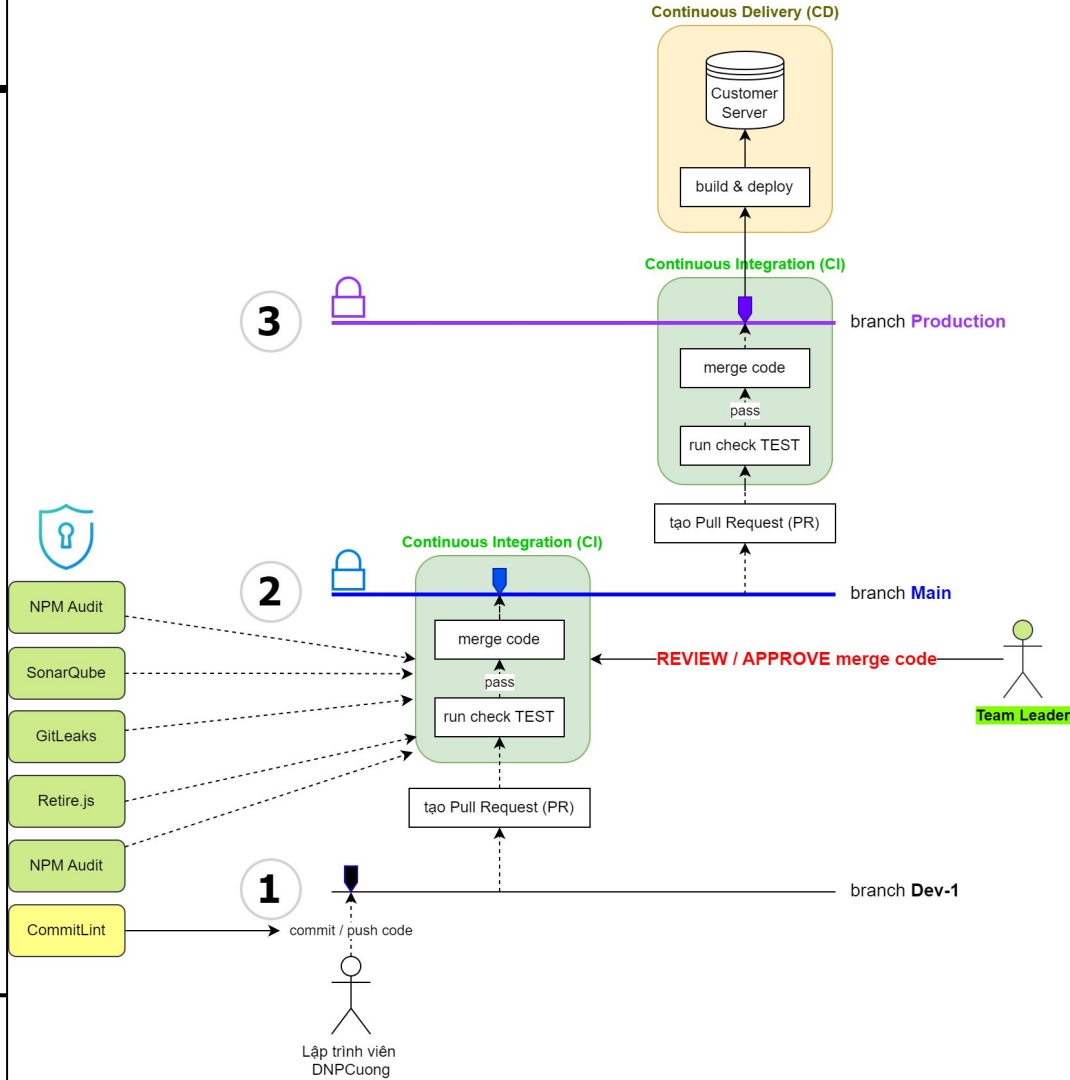
Link demo:

- https://github.com/NenTang/devsecops
- https://nentang.github.io/devsecops/



Ví dụ Demo DevSecOps

1. Phải comment theo đúng quy định thì mới được commit/push code.
2. Code khi push/commit phải pass được các test unit.
3. Code khi push/commit phải được kiểm tra các thư viện đi kèm có dính lỗ hổng hay không?
4. Code khi push/commit sẽ được kiểm tra các thông tin nhạy cảm có bị lộ không?...



XIN CÁM ƠN ĐÃ THEO DÕI !

💖 CUSC G2B - phucuong@ctu.edu.vn 💖

5. THẢO LUẬN

Pentest

Ba khái niệm cơ bản trong bảo mật:
1. Lỗ hổng bảo mật (vulnerabilities) là những điểm yếu bảo mật của một phần mềm, phần cứng, hệ điều hành, hay ứng dụng web cho phép kẻ tấn công một cơ sở để tấn công hệ thống. Lỗ hổng có thể đơn giản như mật khẩu yếu, hay phức tạp như lỗ hổng SQL hoặc tràn bộ nhớ đệm.
2. Khai thác (exploits) là hành động lợi dụng một lỗ hổng, sự cố hay lỗi của phần mềm, đoạn dữ liệu hay một chuỗi các lệnh nhằm gây ra hành vi bất thường không mong muốn xảy ra trên một hệ thống máy tính. Những hành vi đó bao gồm leo thang đặc quyền, đánh cắp thông tin nhạy cảm, tấn công từ chối dịch vụ, v.v.
3. Trọng tải (payloads) là một phần của hệ thống đang tồn tại lỗ hổng và là mục tiêu để khai thác.

Các hình thức pentest

White box Testing: Trong hình thức pentest white box, các chuyên gia kiểm thử được cung cấp đầy đủ thông tin về đối tượng mục tiêu trước khi họ tiến hành kiểm thử. Những thông tin này bao gồm: địa chỉ IP, sơ đồ hạ tầng mạng, các giao thức sử dụng, hoặc source code.
Gray box Testing: Pentest gray box là hình thức kiểm thử mà pentester nhận được một phần thông tin của đối tượng kiểm thử, ví dụ URL, IP address,... nhưng không có hiểu biết đầy đủ hay quyền truy cập vào đối tượng.
Black box Testing: Pentest black box, hay còn gọi là 'blind testing', là hình thức pentest dưới góc độ của một hacker trong thực tế. Với hình thức này, các chuyên gia kiểm thử không nhận được bất kỳ thông tin nào về đối tượng trước khi tấn công. Các pentester phải tự tìm kiếm và thu thập thông tin về đối tượng để tiến hành kiểm thử. Loại hình pentest này yêu cầu một lượng lớn thời gian tìm hiểu và nỗ lực tấn công, nên chi phí không hề rẻ.

Tham khảo

Các yếu tố cần bảo mật:
Cần phải có 1 quy trình bảo mật dự án cụ thể.
https://vnexpress.net/thiet-hai-700-000-usd-vi-nhan-vien-cu-xoa-du-lieu-may-chu-4758312.html

Code test gitleaks

layout.tsx

```
// comment các dòng code nhạy cảm
// const password = "123456@#Abcd";
// const apiKey = "ABCUSC_TOKEN9789";

{/* <p>Mật khẩu: {password}</p> */}
      {/* <p>API Key: {apiKey}</p> */}
```

Code test gitleaks

Config_key.js

```
const password = "123456@#Abcd";
const apiKey = "ABCUSC_TOKEN9789";

if(password & apiKey) {
  console.log("Hack pass");
}
```

Code test gitleaks

Config_key.txt
SecretAccessKey="AKIALALEMEL332320OILE"