



Twelve-factor app Review

Here is where your presentation begins

Tổng quan

Twelve-factor app là một phương pháp luận (methodology) được dùng trong thiết kế phần mềm ứng dụng web. Được đề xuất bởi 1 nhóm lập trình viên thuộc Heroku (một nền tảng đám mây), Twelve-factor nhằm mục đích giúp các lập trình viên:

- Hỗ trợ tự động hóa (automation), giảm thời gian và công sức.
- Quy ước và cấu trúc, giúp chương trình linh hoạt - có thể chuyển đổi giữa các nền tảng khác nhau.
- Phù hợp với nền tảng đám mây, giảm thiểu yêu cầu về server và quản trị hệ thống.
- Giảm thiểu sự khác biệt giữa các môi trường, giúp tăng tốc và có thể phát triển liên tục.
- Có khả năng mở rộng.

Ngoài ra, dù được thiết kế chủ yếu cho các ứng dụng web, Twelve-factor app có thể ứng dụng cho bất kì loại ứng dụng nào.



Codebase

Dependencies

Config

Backing services

Build, release and run

Processes

Port binding

Concurrency

Disposability

Dev/prod parity

Logs

Admin processes

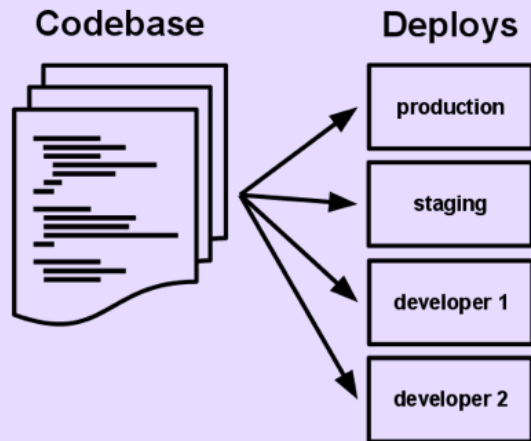


Codebase

Tạm dịch: Mã nguồn cơ sở.

Nội dung: Về cơ bản, nội dung này có 2 phần chính:

- Nên có một mã nguồn cơ sở được quản lý bởi một hệ thống quản lý phiên bản.
- Nên có nhiều triển khai (deployment).



Codebase

Giải thích: Một mã nguồn cơ sở có nhiều mục đích: Người lập trình có một cơ sở để thiết kế sản phẩm một cách nhất quán (inconsistent), dễ dàng truy ngược nếu có sai sót... Ngoài ra, trong một nhóm, việc có một mã nguồn cơ sở còn giúp cho các thành viên tránh các trường hợp mâu thuẫn...

Một hệ thống quản lý phiên bản là để người thiết kế có thể theo dõi các phiên bản theo thời gian của ứng dụng, dễ dàng hơn trong việc thao tác và chỉnh sửa.

Mỗi triển khai sẽ là một 'phiên bản' (instance) của ứng dụng với các điều kiện khác nhau (môi trường, trạng thái...) nhằm các mục đích như cô lập - isolation hay mở rộng - scalability...

Ví dụ: Git là một hệ thống quản lý phiên bản rất tốt cho nội dung này. Bằng cách quy ước một nhánh (branch) như một mã nguồn cơ sở, ta có thể triển khai nhiều nhánh khác để phát triển phần mềm, sử dụng commit như phiên bản... Rồi dựa theo đó mà thiết kế phần mềm.





Dependencies


Tạm dịch: Phụ thuộc. Phụ thuộc ám chỉ những gì mà ứng dụng cần để chạy. Nó bao gồm các thư viện, framework, tools...

Nội dung: Phụ thuộc nên được khai báo rõ ràng và cô lập.

Giải thích: Phụ thuộc có vai trò rất quan trọng trong tất cả giai đoạn của ứng dụng, vì vậy.

- Phụ thuộc cần được khai báo rõ ràng để dễ dàng thao tác và chỉnh sửa (như cập nhật). Ngoài ra, việc các phụ thuộc được khai báo rõ ràng giúp cho việc thiết lập lại (cho thành viên mới tham gia nhóm thiết kế hay cho những người khác ngoài nhóm). Các phụ thuộc nên được khai báo chi tiết: tên, phiên bản...
- Phụ thuộc cần được cô lập giữa các phiên bản để đảm bảo các phiên bản (của các trạng thái) chạy một cách nhất quán, không lệ thuộc lẫn nhau dẫn đến mâu thuẫn.

Ví dụ: package.json của Node.js, requirements.txt của Python... Và các công cụ như pip của Python, npm của Node.js... Với việc cô lập, ta thường sử dụng các môi trường ảo như .venv hay anaconda (Python), node_modules (Node.js), docker...



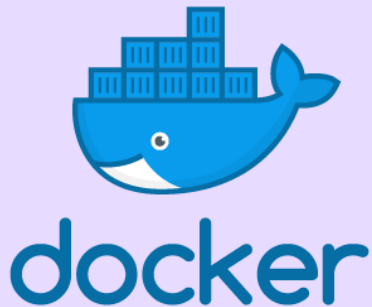
Config

Tạm dịch: Cấu hình. Cấu hình là các tham số mà chương trình dựa theo đó để chạy như: Database, các dịch vụ (đám mây như AWS, Azure...), các giá trị ứng với phiên bản như cổng chạy (PORT)...

Nội dung: Cấu hình nên được đặt trong các file môi trường (thường là .env).

Giải thích: Khác với các giá trị tham số khác, cấu hình sẽ có thể (và thường) thay đổi giữa các phiên bản hay trạng thái của ứng dụng, vì vậy việc thêm vào trong mã nguồn cơ sở là không nên. Các file môi trường thường không được thêm vào mã nguồn cơ sở (.env thường nằm trong .gitignore) mà thay vào đó, tùy vào môi trường mà người thiết kế sẽ xây dựng khác đi.

Ví dụ: .env, docker...



Backing service

Tạm dịch: Dịch vụ hỗ trợ, dịch vụ hậu cần. Thông thường là các dịch vụ sử dụng mạng như database, tin nhắn, mail...

Nội dung: Coi các dịch vụ này là đính kèm, có thể thay thế.

Giải thích: Các dịch vụ này thuộc về cấu hình, vì vậy việc thêm vào mã nguồn là không nên. Ngoài ra, các dịch vụ này có thể thay đổi rất nhiều, đến từ việc sử dụng nội bộ (trong giai đoạn thiết kế), sử dụng các bên thứ ba (khi triển khai sản phẩm), hay thậm chí khi thay đổi công nghệ... Nên mã nguồn của những lần thay đổi này nên không được thay đổi.

Ví dụ: Các dịch vụ database như MySQL, MongoDB...



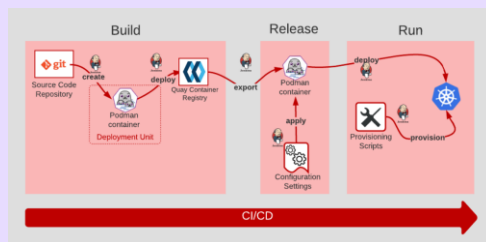
Build, release, run

Tạm dịch: Xây dựng, phát hành, chạy. Đây là ba trạng thái chính của ứng dụng mà cụ thể:

- Xây dựng: Mã nguồn của chương trình nói chung cùng với các phụ thuộc sẽ được biên dịch.
- Phát hành: Thêm cấu hình vào xây dựng.
- Chạy: Ứng dụng sẽ được chạy trong một môi trường vận hành.

Nội dung: Cả ba trạng thái này phải được tách rời.

Giải thích: Các trạng thái sẽ rất khác nhau, vì vậy sẽ cần cách giải quyết khác nhau. Ví dụ: Trạng thái xây dựng là trạng thái thay đổi nhiều nhất: Sửa lỗi, cập nhật công nghệ... Trạng thái phát hành là nơi mã nguồn được 'thử nghiệm', vì vậy nên được tách rời cho một số tính năng như là quay trở lại các phiên bản trước... Cuối cùng, lỗi tại trạng thái chạy có thể khiến cả ứng dụng bị hỏng, vì vậy các thao tác tại ứng dụng chạy nên được đơn giản hóa để hạn chế lỗi.





Processes

Tạm dịch: Tiến trình. Chi tiết hơn, các tiến trình ở đây được coi như là các tiến trình phi trạng thái (không biến).

Nội dung: Ứng dụng được vận hành như một hoặc nhiều các tiến trình phi trạng thái.

Giải thích: Bộ nhớ đệm của các tiến trình không nhất quán. Bởi vì một ứng dụng có nhiều tiến trình, một tiến trình khác chạy có thể đồng nghĩa với việc bộ nhớ đệm của các tiến trình sẽ bị cài đặt lại, dẫn đến việc truyền thông tin giữa các tiến trình là rất khó nếu chỉ dùng bộ nhớ đệm. Ngoài ra, với việc các tiến trình đã được độc lập với nhau, việc mở rộng ứng dụng cũng trở nên dễ dàng hơn. Cuối cùng, với môi trường khác nhau, các tiến trình có thể có giá trị khác nhau nên việc cô lập chúng là cần thiết.

Lưu ý: Việc 'giao tiếp' giữa các tiến trình với nhau có thể được thực hiện thông qua database.

Ví dụ: Có một loại tiến trình giúp quản lý phiên của người dùng (thường được gọi là sessionInfo). Tiến trình này nên được chia sẻ qua server hoặc trung gian như database.

Port binding

Tạm dịch: Ràng buộc cổng. Cổng ở đây là cổng mạng.

Nội dung: Các dịch vụ (hoặc trong một số trường hợp là ứng dụng) sẽ nhận thông tin đầu vào qua cổng mạng thay vì đường dẫn.

Giải thích: Đường dẫn và địa chỉ IP có thể dễ dàng được thay đổi. Cổng mạng sẽ ổn định hơn và dễ dàng quản lý hơn. Ngoài ra, việc ràng buộc cổng mạng ở đây còn giúp cho ứng dụng có thể trở thành một dịch vụ cho một ứng dụng hoặc dịch vụ khác.

Ví dụ: Đa số các dịch vụ hỗ trợ đều có các cổng mặc định của chúng, như MySQL là 3306, MongoDB là 27017...

COMMON PORTS			
TCP/UDP Port Numbers			
7 Echo	554 RTSP	2745 Radmin	5891-690 Windows Live
19 Chargen	546-547 DHCPv6	2967 Symantec AV	6970 Quicktime
20-21 FTP	560 rmonitor	3050 Interbase DB	7212 GhostSurf
22 SSH/SCP	563 NNTP over SSL	3074 XBOX Live	7648-7649 EU-SeeMe
23 Telnet	587 SMTP	3124 HTTP Proxy	8000 Internet Radio
25 SMTP	591 FileMaker	3127 MyDoom	8080 HTTP Proxy
42 WINS Replication	593 Microsoft DCOM	3128 HTTP Proxy	8086-8087 Kaspersky AV
43 WHOIS	631 Internet Printing	3222 GLBP	8118 Privoxy
49 TACACS	636 LDAP over SSL	3260 iSCSI Target	8200 VMware Server
53 DNS	639 MSDP (PIM)	3306 MySQL	8500 Adobe ColdFusion
67-68 DHCP/BOOTP	646 LDP (IMPLS)	3389 Terminal Server	8767 TeamSpeak
69 TFTP	691 MS Exchange	3689 iTunes	8866 Radmin
70 Gopher	860 iSCSI	3690 Subversion	9100 HP JetDirect
79 Finger	873 rsync	3724 World of Warcraft	9101-9103 Bacula
80 HTTP	902 VMware Server	3784-3785 Jentriilo	9119 Mail
88 Kerberos	989-990 FTP over SSL	4333 mSQL	9800 WebDAV
102 MS Exchange	993 IMAP4 over SSL	4444 Blaster	9898 Dabber
110 POP3	995 POP3 over SSL	4664 Google Desktop	9988 Rbot/Spybot

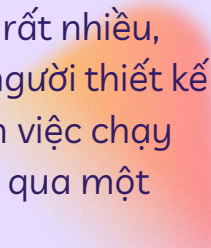


Concurrency

Tạm dịch: Đa nhiệm, đồng bộ, đồng thời. Từ này thường nói về hai việc có thể xảy ra cùng một lúc, nhưng không phải song song.

Nội dung: Ứng dụng nên được mở rộng thông qua mô hình quy trình.

Giải thích: Khác với các ứng dụng thông thường, các ứng dụng web thường có yêu cầu được mở rộng rất nhiều, đến mức những phương pháp thông thường là không khả thi. Vì vậy, khi thiết kế một ứng dụng web, người thiết kế cần chú ý đến việc thiết kế để có thể mở rộng theo chiều ngang (horizontal scaling), tức cân nhắc đến việc chạy ứng dụng bằng nhiều môi trường (nhiều máy, nhiều nguồn)... Các môi trường này được quản lý thông qua một Load balancer.



Disposability

Tạm dịch: Dừng một lần, có thể vứt bỏ

Nội dung: Ứng dụng nên được khởi động và kết thúc một cách gọn gàng.

Giải thích: Nội dung này tập trung vào môi trường chạy là chính. Theo đó, vì môi trường chạy chỉ cho phép can thiệp một cách tối thiểu nên việc khiến cho ứng dụng (hoặc các tiến trình) được gọn gàng sẽ giúp cho việc thiết kế và triển khai ứng dụng được nhanh hơn. Đặc biệt, lối thiết kế như vậy sẽ rất tốt trong trường hợp xuất hiện các lỗi khiến môi trường chạy của ứng dụng gặp lỗi vì nó giúp việc khôi phục lại nhanh và hạn chế các ảnh hưởng.

Ví dụ: Với môi trường ảo Docker, khi một ứng dụng dừng, Docker sẽ gửi về tín hiệu SIGTERM hoặc SIGKILL. SIGTERM có thể bỏ qua nhưng SIGKILL chắc chắn sẽ ngắt môi trường một cách ép buộc. Vì vậy, để hạn chế mất mát dữ liệu hay lỗi, ứng dụng nên được kết thúc sau SIGTERM, hạn chế sử dụng SIGKILL.



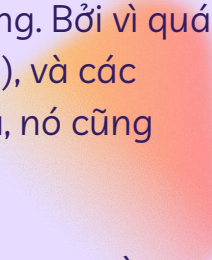
Dev/prod parity

Tạm dịch: Tương đồng phát triển/sản phẩm.

Nội dung: Mã nguồn, môi trường... giữa hai trạng thái đang phát triển và sản phẩm hoàn chỉnh nên có tính tương đồng.

Giải thích: Khác với 'Build, release, run', nội dung này tập trung vào tính ổn định và duy trì của ứng dụng. Bởi vì quá trình thiết kế ứng dụng truyền thống sẽ hình thành nên các khác biệt (thời gian, tính cá nhân, công cụ), và các khác biệt này có thể dẫn tới lỗi. Việc giảm các khoảng cách này sẽ giảm khả năng xảy ra lỗi. Ngoài ra, nó cũng giúp việc phối hợp giữa người thiết kế ứng dụng và người triển khai ứng dụng phối hợp tốt hơn...

Ví dụ: Sử dụng chung hệ điều hành, các dịch vụ database hay các công cụ... sẽ giúp cho sản phẩm và mã nguồn phát triển giống nhau hơn...



Logs

Tạm dịch: Nhật ký.

Nội dung: Xem việc ghi chép nhật ký như luồng sự kiện.

Giải thích: Việc gỡ lỗi của các ứng dụng mạng là rất khó, đặc biệt khi một ứng dụng có nhiều luồng tiến trình, vì vậy việc ghi lại nhật ký là cần thiết. Tuy vậy, để nhật ký có thể thể hiện rõ ràng quá trình chạy của ứng dụng (luồng tất cả các quá trình theo trình tự thời gian), việc thao tác lên quá trình viết nhật ký của ứng dụng là không nên.

Ví dụ: Bundler của Ruby, Pip và Virtualenv (hoặc anaconda) của Python, autoconf của C, package.json của Node.js...




Admin processes

Tạm dịch: Tiến trình quản trị.

Nội dung: Tiến trình quản trị được xem như là một tiến trình tách rời.

Giải thích: Việc quản trị cũng thực hiện thao tác vào các phần của ứng dụng như các tiến trình khác. Vì vậy, để tránh xảy ra lỗi và duy trì sự đồng bộ trong việc thiết kế ứng dụng, quản trị nên được xem như một tiến trình và chạy như một tiến trình. Tuy vậy, bởi vì quyền và phạm vi truy cập, thao tác của tiến trình quản trị khác với các tiến trình thông thường, tiến trình quản trị nên được chạy trên một môi trường riêng biệt.





Cảm ơn vì đã theo dõi

CREDITS: This presentation template was created by **Slidesgo**,
including icons by **Flaticon**, infographics & images by **Freepik**
Contents by **Swetha Tandri**

<https://12factor.net/> <https://12factor.net/vi>

<https://youtu.be/1OhmRmMsGdQ?si=xUbOxriTDkUUGtfD>

<https://viblo.asia/p/the-twelve-factor-app-phan-1-oOVlYzzn58W>

<https://www.redhat.com/architect/12-factor-app>