Ghi chú của một coder

Vũ Anh

Tháng 01 năm 2018

Mục lục

Phần I

Linh tinh

Chương 1

Nghiên cứu

01/11/2017
Không biết mình có phải làm nghiên cứu không nữa? Vừa kiêm phát triển, vừa đọc paper mỗi ngày. Thôi, cứ (miễn cưỡng) cho là nghiên cứu viên đi.

1.1 Các công cụ

1.1.1 Google Scholar & Semantic Scholar

Google Scholar vẫn là lựa chọn tốt

- Tìm kiếm tác giả theo lĩnh vực nghiên cứu và quốc gia: sử dụng filter label: + đuôi
  - ví dụ: danh sách các nhà nghiên cứu Việt Nam thuộc lĩnh vực xử lý ngôn ngữ tự nhiên
- danh sách này đã sắp xếp theo lượng trích dẫn

Bên cạnh đó còn có semanticscholar (một project của allenai) với các killer features

- Tìm kiếm các bài báo khoa học với từ khóa và filter theo năm, tên hội nghị
- Xem những người ảnh hưởng, ảnh hưởng bởi một nhà nghiên cứu, cũng như xem co-author, journals và conferences mà một nhà nghiên cứu hay gửi bài

1.1.2 Mendeley

Mendeley rất tốt cho việc quản lý và lưu trữ. Tuy nhiên điểm hạn chế lại là không lưu thông tin về citation

1.1.3 Hội nghị và tạp chí

Các hội nghị tốt về xử lý ngôn ngữ tự nhiên

- Rank A: ACL, EACL, NAACL, EMNLP, CoNLL

- Rank B: SemEval

Các tạp chí

- Computational Linguistics (CL)

1.1.4 Câu chuyện của Scihub

Sci-Hub được tạo ra vào ngày 5 tháng 9 năm 2011, do nhà nghiên cứu đến từ Kazakhstan, Alexandra Elbakyan
Hãy nghe chia sẻ của cô về sự ra đời của Sci-Hub
> Khi tôi còn là một sinh viên tại Đại học Kazakhstan, tôi không có quyền truy cập vào bất kỳ tài liệu nghiên cứu. Những bai bao tôi cần cho dự án nghiên cứu của tôi. Thanh toán 32 USD thi thât la điên rô khi bạn cần phải đọc lướt hoặc đọc hàng chục hoặc hàng trăm tơ để làm nghiên cứu. Tôi có được những bai bao nhơ vao trôm chúng. Sau đó tôi thấy có rất nhiều và rất nhiều nhà nghiên cứu (thậm chí không phải sinh viên, nhưng các nhà nghiên cứu trường đại học) giống như tôi, đặc biệt là ở các nước đang phát triển. Họ đã tạo ra các cộng đồng trực tuyến (diễn đàn) để giải quyết vấn đề này. Tôi là một thành viên tích cực trong một cộng đồng như vậy ở Nga. Ở đây ai cần có một bài nghiên cứu, nhưng không thể trả tiền cho nó, có thể đặt một yêu cầu và các thành viên khác, những người có thể có được những giấy sẽ gửi nó cho miễn phí qua email. Tôi có thể lấy bất cứ bai nao, vì vậy tôi đã giải quyết nhiều yêu cầu và người ta luôn rất biết ơn sự giúp đỡ của tôi. Sau đó, tôi tạo Sci-Hub.org, một trang web mà chỉ đơn giản là làm cho quá trình này tự động và các trang web ngay lập tức đã trở thành phổ biến.
Về phần mình, là một nhà nghiên cứu trẻ, đương nhiên phải đọc liên tục. Các báo cáo ở Việt Nam về xử lý ngôn ngữ tự nhiên thì thường không tải lên các trang mở như arxiv.org, các kỷ yếu hội nghị cũng không public các proceedings. Thật sự scihub đã giúp mình rất nhiều.
Scihub bị chặn
Vào thời điểm này (12/2017), scihub bị chặn quyết liệt. Hóng được trên page facebook của scihub các cách truy cập scihub. Đã thử các domain khác như .tw, .hk. Mọi chuyện vẫn ổn cho đến hôm nay (21/12/2017), không thể truy cập vào nữa.
Đành phải cài tor để truy cập vào scihub ở địa chỉ http://scihub22266oqcxt.onion.
Và mọi chuyện lại ổn.

1.2 Làm sao để nghiên cứu tốt

- Làm việc mỗi ngày
- Cập nhật các kết quả từ các hội nghị, tạp chí
- Viết nhật ký nghiên cứu mỗi tuần (tổng kết công việc tuần trước, các ý tưởng mới, kế hoạch tuần này)

1.3 Sách giáo khoa

Machine Learning Yearning, by Andrew Ng

1.4 Lượm lặt

Review các khóa học Deep Learning

Chương 2

Trở thành giảng viên

Hơi bị hay Ai cũng có thể trở thành giảng viên giỏi nếu của anh Cường tại techmaster. Tưởng phải mua, hóa ra lại được set free. A hihi.

- Lên ý tưởng
- Chuẩn bị nội dung
- Code mẫu
- Xây dựng kịch bản
- Trình bày đơn giản, dễ hiểu

Một bài giảng

2.1 Khác biệt giữa dạy online và offline

Dạy online
Ưu điểm

1. Khả năng mở rộng rất tốt
2. Chi phí thấp
3. Số lượng học viên không giới hạn

Nhược điểm

- Tỷ lệ bỏ học rất cao
- Tương tác chưa thực sự tốt

2.1.1 Làm sao để giảm tỷ lệ bỏ học trực tuyến?

- Video ngắn $<$ 10 phút
- Cấp chứng chỉ 70-90 USD lấy 1 chứng chỉ cho khóa học 4-6 tuần
- Chấm bài tập

- Facebook group kết nối giảng viên học viên
- Bài tập đủ dễ: chia nhỏ dự án khó ra
- Gamification: học như chơi
- Chịu khó email thông báo cho lớp

2.2 Marketing

Các hình thức marketing
1. Mạng xã hội: Facebook, forum Phải boost quảng cáo. 500000 - 2000000 VND cho mỗi khóa học
2. SEO - Google Search
3. Qua hội thảo, meetup hiệu quả thấp do số lượng người tham gia < 40 người
Conversation rate: số người mua hàng / số người tham quan
Nếu đến từ organic search CR = 1 / 200
Nếu qua chia sẻ bài viết mạng xã hội CR = 1 / 4000
Nếu qua giới thiệu, tư vấn người quen CR = 1 / 2

2.3 Flip Learning

Lớp học đảo ngược
Vấn đề học online

- Bỏ học cao
- Tương tác face 2 face kém
- Thực hành không có, không kiểm soát

Vấn đề học offline

- Chi phí cao
- Giao thông không thuận tiện
- Tuyển sinh khó

Flip Learning lớp học đảo ngược, kết hợp ưu điểm học trực tuyến và thực hành phòng lab.
- Khuyến khích học viên xem trước bài giảng video hướng dẫn giảng viên
- Đọc tìm hiểu thêm, trả lời trắc nghiệm qua Internet
- Tại buổi học, dành tối đa thời gian để **thực hành, hỏi đáp, hợp tác**
Khi đến lớp, học viên đã có kiến thức, biết rõ mình sẽ làm gì.
- Không thụ động, rèn tính tự học, tự đọc
- Không chống cằm nhìn giảng viên, không ghi chép
- Hỏi, làm, nói, giúp đỡ nhau
Flip Learning:
- Giảng viên phải chuẩn bị bài giảng
- Đến lớp trao đổi, hướng dẫn

2.4 Thuyết trình

Tự nhiên hôm nay (22/01/2018) lại đọc được bài You suck at PowerPoint, thấy hay quá.
Sau đây là 10 lỗi thường gặp khi làm bài thuyết trình

1. Quá nhiều chữ trong 1 slide.
2. Màu chữ và màu nền không tương phản với nhau.
3. Dùng clip art, word art.
4. Hình ảnh sử dụng trong slide chất lượng kém, scale sai tỉ lệ.
5. Sử dụng nhiều font chữ trong 1 slide.
6. Lạm dụng quá nhiều hiệu ứng (animation/transition).
7. Bài presentation không có cấu trúc.
8. Slide không ăn nhập gì với nội dung trình bày.
9. Không ghi rõ nguồn khi sử dụng tài liệu, hình ảnh của người khác.
10. Ý thức của người làm slide

2.5 Xác định đối tượng học tập

Học viên Techmaster là ai?
1- Sinh viên CNTT năm đầu, cần thực hành: 302- Người thất nghiệp: 403- Lập trình cần nâng cao kỹ năng 304- Hầu hết là từ nông thôn 705- Thời gian có hạn, cần tìm việc ngay
Họ cần gì?
1- Bài giảng thực tế, có nhiều ví dụ sinh động 2- Rất khác với ở trường đại học: + Không phải thi + Lý thuyết ít, thực hành nhiều, trừu tượng ít + Ví dụ phải cool + Có người hỗ trợ ngay 3- Trung tâm giới thiệu việc làm sau khi học

2.6 Sai lầm cỗ hữu của giảng viên

1- Nghĩ ai cũng biết như mình.
Học viên 80Giảng viên dùng toàn từ chuyên ngành tiếng Anh mà không giải thích cặn kẽ
2- Nói nhiều, lý thuyết nhiều, ít có ví dụ, thí nghiệm minh họa. Học viên đã rất chán kiểu dạy ở đại học
3- Khô cứng: giải thích về kế thừa
4- Ba hoa, bốc phét như bán hàng đa cấp. Lesson online giới hạn 10 phút, học offline chỉ có 140 phút.
5- Dùng nhiều tính từ, đại ngôn: rất, cực.. mà không có con số.
6- Cẩu thả: không chuẩn bị kỹ slide, slide toàn chữ 12 trang.
7- Mã nguồn, dự án ví dụ vô duyên, khó hiểu

```
*
**
***
****
*****

*     *
 *   *
  * *
   *
```

8- Chia nhỏ các bài tập

2.7 Xây dựng khung giáo trình

1- Top Down tốt hơn Bottom Up
2- Tham khảo sách ebook, Udemy, PluralSights, Lynda...
3- Người bình thường không thể nhớ quá 5 mục trong một buổi học: + Offline: 1 buổi dạy 1 chủ đề chia thành 4 minitask + Online: 1 section gồm 4-5 video, mỗi video $<$ 10 phút
4- Sử dụng MindMap

2.8 Viết kịch bản nói

1- Viết chi tiết chính xác đến từng câu, rồi đọc lại + Nói chưa lưu loát $>$ viết chi tiết 2- Liệt kê ý chính, vừa demo, vừa nói + Nói tốt + lười + Hậu kỳ phải cắt bỏ nhiều đoạn nói nhịu, ề à Câu từ dài dòng, rườm ra, tỷ lệ tiếp thu càng thấp Học trực tuyến, học viên dùng mắt đọc - xem là chính 70%, nghe là phụ, 30%. Video qua 5 phút gây buồn ngủ.
Nhắc kịch bản
- Không nên in ra giấy ! - Màn hình rộng Dell Utra, để mở cửa sổ nhắc kịch bản - Mở rộng thêm 1 màn hình mới - Gõ kịch bản ra iPad, dựng iPad lên thành màn hình phụ

2.9 Đồ nghề tạo giáo trình trực tuyến

1- Web Cam
+ Logitech C920 hoặc Xioami Camera loại 650,000VND. + Tỷ lệ bỏ lửng bài video giảm 30% nếu học viên thấy mặt giảng viên. + Mặt giảng viên đẹp trai, hấp dẫn, hài hước càng tốt + Mở sẵn cửa sổ ứng dụng cần demo. Giảm tối đa thời gian chết
2- Camtasia
+ Camtasia Studio trên Windows có nhiều chức năng hơn bản Camtasia Mac + Phiên bản Camtasia Mac không nén được video định dạng H264 phải dùng FFmpeg để nén + Thu ở khung hình 1280 * 720 pixels. Trên Mac có công cụ xScope để căn kích thước màn hình thu
3- Phòng thu âm cần yên tĩnh

+ Đóng chặt cửa sổ, cửa phòng khi quay chấp nhật nóng trong 10 phút quay.
+ Bật chế noise remove trong Camtasia
4- Sublime Text 5- Ink2Go 6- Lucid Chart 7- Adobe Photoshop 8- Skitch 9- PowerPoint 10- Slides.com

2.10 Kinh nghiệm khi quay video

Phòng cần yên tĩnh: nên tắt quạt hay các thiết bị điện tử gây ồn. Tắt điện thoại tránh cuộc gọi đến khi đang ghi hình Để micro xa bàn phím để không lẫn tiếng lạch cạch
Tuyệt đối không nên để background wallpaper lòe loẹt, học viên không chú ý vào demo của bạn mà xem background thì hiệu quả truyền đạt rất tệ. Tốt nhất để màn hình background là đen 100
Tắt Skype, chat, loại bỏ các icon không cần thiết trên màn hình desktop
Nên dùng web camera Logitech C920 để xa mồm để không ghi tiếng thở hổn hển của giảng viên
Nên quay cả mặt giảng viên, nhưng cut crop để không quay background của phòng thu. Hãy để học viên tập trung vào bài giảng
Không sử dụng âm thanh nên có tiếng hát hoặc giai điệu nhanh, phức tạp khiến học viên không tập trung
Lưu ý rằng:
Học viên thích xem demo sinh động và đọc chữ trên màn hình rất nhanh, đừng nói quá dài dòng, phức tạp.

2.11 Thiết kế để học trực tuyến

1- Một video $<$ 10 phút. Tốt nhất 3 - 5 phút
2- Ví dụ là một dự án mẫu chạy được. Đơn giản tối đa
3- Học viên phấn khích khi thấy kết quả cuối cùng
4- Chia nhỏ task top-down:
+ Sản phẩm này làm gì?
+ Có chức năng gì? chỉ nên 1 hoặc 2
+ Màn hình chính

2.12 Upload video và tạo bài giảng

Tạo một lesson như thế nào?
- Có một video dài $<$ 10 phút
- Tối thiểu 3 câu hỏi quiz
- Nên bổ sung mã nguồn hoặc slide

2.13 Tính logic cấu trúc bài giảng

Không nên:
- Tùy tiện theo ý thích hoặc kinh nghiệm cá nhân giảng viên - Giập khuôn theo sách hoặc khóa học có sẵn Hậu quả học nhiều, nhưng kỹ năng, kiến thức lộn xộn. Thà học ít mà dùng được nhiều, hiệu quả còn hơn

Nên:
- Thiết kế top down dùng Mindmap - Các mẩu kiến thức có liên kết, sâu chuỗi
- Trước - Sau: thủ tục -> hướng đối tượng -> design pattern array -> dictionary -> linked list
SQL raw query -> Objection Relation Mapping
- Kế thừa: Cha - Con, Chị - Em
UIView > UIScollView > UITableView, UICollectionView
- Đối lập - so sánh:
Postgresql <> MongoDB, FireBase <> RethinkDB Apple Push Notification <> Google Cloud Messaging
- Phân nhóm theo tiêu chí: Dễ học, dùng thường xuyên, học viên sướng. Sướng quyết định tất cả
Ví du:
- Lẻ tẻ, minh họa cụ thể từng cú pháp, API function. Học viên không thu hoạch được nhiều - Một dự án kết hợp các bước sẽ thú vị hơn, dài hơn. Hay lúc đầu, buồn ngủ lúc sau.

Chương 3

Nghề lập trình

Chân kinh con đường lập trình: [Teach Yourself Programming in Ten Years. Peter Norvig](http://norvig.com/21-days.html)
Trang web hữu ích
* Chia sẻ thú vị: [15 năm lập trình ở Việt Nam](https://vozforums.com/showthread.php?t=3431312) của Blanic (vozfourm) * Trang web chứa cheatsheet so sánh các ngôn ngữ lập trình và công nghệ http://hyperpolyglot.org/
01/11/2017
Vậy là đã vào nghề (đi làm full time trả lương) được 3 năm rưỡi rồi. Thời gian trôi qua nhanh như *ó chạy ngoài đồng thật. Tâm đắc nhất với câu trong một quyển gì đó của anh lead HR google. Có 4 level của nghề nghiệp. 1 là thỏa mãn được yêu cầu cả bản. 2 là dự đoán được tương lai. 3 là cá nhân hóa (ý nói là tận tình với các khách hàng). 4 là phiêu diêu tự tại. Hay thật! Bao giờ mới được vào mức 4 đây.

Chương 4

Latex

15/12/2017:
Hôm nay tự nhiên nổi hứng vẽ hình trên latex. Thấy blog này là một guide line khá tốt về viết blog phần mềm. Quyết định cài latex
Theo [hướng dẫn này](http://milq.github.io/install-latex-ubuntu-debian/)
“‘ sudo apt-get install texlive-full sudo apt-get install texmaker “‘
Tìm được ngay bên này https://www.overleaf.com/ có vẻ rất hay luôn
Hướng dẫn cực kì cơ bản http://www.math.uni-leipzig.de/ hellmund/LaTeX/pgf-tut.pdf
Chương trình đầu tiên, vẽ diagram cho LanguageFlow

```
\documentclass[border=10pt]{standalone}
\usepackage{verbatim}
\begin{comment}
\end{comment}
\usepackage{tikz}
\begin{document}
\begin{tikzpicture}
    \node[draw] (model) at (0, 0) {Model Folder};
    \node[draw] (analyze) at (6, 0) {Analyze Folder};
    \node[draw] (board) at (3,2) {Board};
    \node[draw] (logger) at (3, −2) {Logger};

    \path[−>, densely dotted] (board.east)
        edge [out=0, in=90]
        node[ fill =white, pos=.5] {\tiny (1) init }
        (analyze.north) ;
    \path[−>, densely dotted] (board.south)
        edge [out=−90, in=180]
        node[ fill =white, pos=.3] {\tiny (2) serve}
        (analyze.west) ;
        \path[−>, densely dotted] (logger.west)
        edge [out=180, in=−90]
        node[ fill =white, pos=.7] {\tiny (1) read}
        (model.south) ;
        \path[−>, densely dotted] (logger.east)
```

```
        edge [out=0, in=−90]
        node[ fill =white, pos=.7] {\tiny (2) write}
        (analyze.south) ;
\end{tikzpicture}
\end{document}
```

Doc! Doc! Doc! https://en.wikibooks.org/wiki/LaTeX/PGF/TikZ

Chương 5

Chào hàng

16/01/2018 Bố khỉ. Hôm nay gửi lời mời kết bạn đến một thằng làm research về speech mà nó "chửi" mình không biết pitch. Tổ sư. Tuy nhiên, nó cũng dạy mình một bài học hay về pitch.
Chửi nó là vậy nhưng lần sau sẽ phải đầu tư nhiều hơn cho các lời pitch.
Vẫn không ưa Huyền Chíp như ngày nào, nhưng [bài này](https://www.facebook.com/notes/huyen-chip/k
Tóm lại skill này có 4 phần
1. Ngôn ngữ không trau chuốt 2. Giới thiệu bản thân không tốt 3. Không chỉ ra cho người nhận rằng họ sẽ được gì 4. Không có phương án hành động
Đối với email, thì cần triển khai thế này
* [Chào hỏi] * [Giới thiệu bản thân một cách nào đó để người đọc quan tâm đến bạn] * [Giải thích lý do bạn biết đến người này và bạn ấn tượng thế nào với họ – ai cũng thích được nghe khen] * [Bạn muốn gì từ người đó và họ sẽ được gì từ việc này] * [Kết thúc]

Chương 6

Phát triển phần mềm

* Phát triển phần mềm là một việc đau khổ. Từ việc quản lý code và version, packing, documentation. Dưới đây là lượm lặt những nguyên tắc cơ bản của mình.
Quản lý phiên bản
Việc đánh số phiên bản các thay đổi của phần mềm khi có hàm được thêm, lỗi được sửa, hay các phiên bản tiền phát hành cần thống nhất theo chuẩn của [semversion]. Điều này giúp nhóm có thể tương tác dễ hơn với người dùng cuối.
![](https://raw.githubusercontent.com/magizbox/magizbox/master/wordpress/phat$_t rien_p han_m em/version.$
Đánh số phiên bản
Phiên bản được đánh theo chuẩn của [semversion](https://semver.org/).
* Mỗi khi một bug được sửa, phiên bản sẽ tăng lên một patch * Mỗi khi có một hàm mới được thêm, phiên bản sẽ tăng lên một patch. * Khi một phiên bản mới được phát hành, phiên bản sẽ tăng lên một minor. * Trước khi phát hành, bắt đầu với x.y.z-rc, x.y.z-rc.1, x.y.z-rc.2. Cuối cùng mới là x.y.z * Mỗi khi phiên bản rc lỗi, khi public lại, đặt phiên bản alpha x.y.z-alpha.t (một phương án tốt hơn là cài đặt thông qua github)
Đánh số phiên bản trên git
Ở nhánh develop, mỗi lần merge sẽ được đánh version theo PATCH, thể hiện một bug được sửa hoặc một thay đổi của hàm
Ở nhánh master, mỗi lần release sẽ được thêm các chỉ như x.y1.0-rc, x.y1.0-rc.1, x.y1.0-rc, x.y1.0
Vẫn còn lăn tăn:
* Hiện tại theo workflow này thì chưa cần sử dụng alpha, beta (chắc là khi đó đã có lượt người sử dụng mới cần đến những phiên bản như thế này)
Tải phần mềm lên pypi
Làm theo hướng dẫn [tại đây](http://peterdowns.com/posts/first-time-with-pypi.html)
1. Cấu hình file '.pypirc' 2. Upload lên pypi
"' python setup.py sdist upload -r pypi "'

Chương 7

Phương pháp làm việc

Xây dựng phương pháp làm việc là một điều không đơn giản. Với kinh nghiệm 3 năm làm việc, trải qua 2 project. Mà vẫn chưa produce được sản phẩm cho khách hàng. Thiết nghĩ mình nên viết phương pháp làm việc ra để xem xét lại. Có lẽ sẽ có ích cho mọi người.
Làm sao để làm việc hiệu quả, hay xây dựng phương pháp làm việc hữu ích? Câu trả lời ngắn gọn là "Một công cụ không bao giờ đủ".
<!–more–>
Nội dung
1. [Làm sao để đánh giá công việc trong khoảng thời gian dài hạn?](section1) 2. [Làm sao để quản lý project?](section2) 3. [Làm sao để công việc trôi chảy?](section3) 4. [Làm sao để xem xét lại quá trình làm việc?](section4)
nbsp;
Làm sao để đánh giá công việc trong khoảng thời gian dài hạn?
Câu trả lời OKR (Objectives and Key Results)
 *OKR Framework*
Dầu mỗi quý , nên dành vài ngày cho việc xây dựng mục tiêu và những kết quả quan trọng cho quý tới. Cũng như review lại kết quả quý trước.
Bước 1: Xây dựng mục tiêu cá nhân (Objectives)
Bước 2: Xây dựng các Key Results cho mục tiêu này
Bước 3: Lên kế hoạch để hiện thực hóa các Key Results
nbsp;
Làm sao để quản lý một project
Meistertask
 $*MeisterTask*$
nbsp;
Làm sao để công việc trôi chảy?
Có vẻ trello là công cụ thích hợp
Bước 1: Tạo một team với một cái tên thật ấn tượng (của mình là Strong Coder)
Trong phần Description của team, nên viết Objectives and Key Results của quý này
Sau đây là một ví dụ
“‘ Objectives and Key Results

-> Build Vietnamese Sentiment Analysis -> Develop underthesea -> Deep Learning Book “‘
Bước 2: Đầu mỗi tuần, tạo một board với tên là thời gian ứng với tuần đó (của mình là ‘2017 | Fight 02 (11/12 - 16/12)‘)
Board này sẽ gồm 5 mục: "TODO", "PROGRESSING", "Early Fight", "Late Fight", "HABBIT", được lấy cảm hứng từ Kanban Board
*
$TrelloBoardexample*$
* Mỗi khi không có việc gì làm, xem xét card trong "TODO" * [FOCUS] tập trung làm việc trong "PROGRESSING" * Xem xét lại thói quen làm việc với "HABBIT"
Một Card cho Trello cần có
* Tên công việc (title) * Độ quan trọng (thể hiện ở label xanh (chưa quan trọng), vàng (bình thường), đỏ (quan trọng)) * Hạn chót của công việc (due date)
Sắp xếp TODO theo thứ tự độ quan trọng và Due date
nbsp;
Làm sao để xem xét lại quá trình làm việc?
Nhật lý làm việc hàng tuần . Việc này lên được thực hiện vào đầu tuần . Có 3 nội dung quan trọng trong nhật ký làm việc (ngoài gió mây trăng cảm xúc, quan hệ với đồng nghiệp...)
* Kết quả công việc tuần này * Những công việc chưa làm? Lý do tại sao chưa hoàn thành? * Dự định cho tuần tới
Đang nghiên cứu
Làm sao để lưu lại các ý tưởng, công việc cần làm?: Dùng chức năng checklist của card trong meister. Khi có ý tưởng mới, sẽ thêm một mục trong checklist
Làm sao để tập trung vào công việc quan trọng?: Dùng chức năng tag của meister, mỗi một công việc sẽ được đánh sao (với các mức 5 sao, 3 sao, 1 sao), thể hiện mức độ quan trọng của công việc. Mỗi một sprint nên chỉ tập trung vào 10 star, một product backlog chỉ nên có 30 star.
Tài liệu của dự án: Sử dụng Google Drive, tài liệu mô tả dự án sẽ được link vào card tương ứng trong meister.

Chương 8

Làm sao để thành công?

Vòng tròn thành công gồm có 3 phần : ĐAM MÊ, NĂNG LỰC, LỢI NHUẬN

27/01/2018: Hôm nay xem Shark Tank Việt Nam có vụ nói về các yếu tố để thành công hay quá.

ĐAM MÊ: là cái mình yêu thích làm

NĂNG LỰC: là cái mình làm tốt

LỢI NHUẬN: là cái đem lại giá trị cho người khác

Chương 9

Đặt tên email

Email hiện tại là anhv.ict91@gmail.com. Chắc tốt hơn là email brother.rain.1024@gmail.com. Vụ này đã được anh Lê Hồng Phương nhắc một lần rồi.

Chương 10

Agile

View online http://magizbox.com/training/agile/site/

10.1 Introduction

What is Agile? 2 Agile methodology is an alternative to traditional project management, typically used in software development. It helps teams respond to unpredictability through incremental, iterative work cadences, known as sprints. Agile methodologies are an alternative to waterfall, or traditional sequential development.
Agile Manifesto
Scrum Framework
A manifesto for small teams doing important work 4
What is the difference between Scrum and Agile Development? 1 Scrum is just one of the many iterative and incremental agile software development method. You can find here a very detailed description of the process.
In the SCRUM methodology a sprint is the basic unit of development. Each sprint starts with a planning meeting, where the tasks for the sprint are identified and an estimated commitment for the sprint goal is made. A Sprint ends with a review or retrospective meeting where the progress is reviewed and lessons for the next sprint are identified. During each sprint, the team creates finished portions of a product.
In the Agile methods each iteration involves a team working through a full software development cycle, including planning, requirements analysis, design, coding, unit testing, and acceptance testing when a working product is demonstrated to stakeholders.
Agile Stories and Teasers 2011, PRESENTATION: HOW OUR TEAM LIVES SCRUM 2010, The real life of a Scrum team – with photos 2009, How Scrum Helped Our Team Agile Tools 3 Agilefant (4/ 3/ 3) What is the difference between Scrum and Agile Development?
Agile Methodology
agile-tools.net/
A manifesto for small teams doing important work

10.2 Team

Scrum Team 1 Within the Scrum Framework three roles are defined:
Product Owner Scrum Master Development team Each of these roles has a defined set of responsibilities and only if they fulfill these responsibilities, closely interact and work together they can finish a project successfully.
Scrum Roles Stakeholders
Product Owner One of the most important things for the success of scrum is the role of the Product Owner, who serves as an interface between the team and other involved parties (stakeholders). It can be said that in companies that use scrum, the tasks and responsibilities of the particular Product Owner are never the same. Starting with the choice of that person provided with the proper and necessary skills, make them take specific trainings, up to the responsibility they take; the role of the Product Owner –short PO- is the most complex one regarding that procedure.
Often the PO has to "fight" on both sides. Whereas the team can work a certain fraction of time (time boxed) "protected" by the Scrum Master, the Product Owner often needs to deal with marketing, management or the customers in order to be able to present the software requirements (User Stories) quite precisely to the team (see the box "criteria for User Stories).
Furthermore the Product Owner is responsible for the return on investment (ROI). He validates the solutions and verifies whether the quality is acceptable or not from the end-users' point of view. He also has to decide over the importance of single features in order to prioritize these in their treatment and he has to tell the team what the product should look like in the end (product vision). Since one of the teams' tasks is to work effectively, the Product Owner must react fast on call-backs. Hence he fulfills the role of a communicator, as he must be in contact with all stakeholders, sponsors and last but not least the team throughout a project. After all it is his task to coordinate the financial side of the product development, which is successful through his continuous work and prioritizing the advancing tasks (Product Backlog). All these diverse requests demonstrate how important the selection of the "right" person for the role of the PO is for the success of a project.
The nomenclature is definite. The Product Owner is at Scrum not only the manager of a product, but also the Owner and therefore he is the one responsible for the correct creation of a product. Being a Product Owner means:
You are responsible for the success of the outcome of the product delivered by the team. You make Business decisions of importance and priorities. You deliver the vision of the product to the team. You prepare the User Stories for the team of development. You should possess severe domain knowledge. You validate the outcomes and test them for their quality. You react promptly on callbacks. You communicate on a continual base with all Stakeholders, financiers and the team. You control the financial side of the project.
Scrum Master The most obvious difference between a team leader and a Scrum Master is represented by the name itself though. Whereas one is leading the team and sets the tasks, the other one is in charge of observing that the team obeys the rules and realizes the method of Scrum entirely. The Scrum Master does not interfere into the decisions of the team regarding specifically the development, but rather is there for the team as an advisor. He only interferes actively when anybody of the team or any other participant of a project (Stakeholder) does not

obey the rules of Scrum. Whereas a team leader often gives requirements and takes responsibility for the completion of those, an experienced Scrum Master gives only impulses and advises to the team to lead the correct way, to use the right method or to choose the right technology. In fine the Scrum Master acts more like a Team Coach than a team leader.
ScrumMaster and Impediments Another important task of the Scrum Master is to get rid of all possible impediments that might disturb the work of the team. Usually problems can be classified in three different categories. The first one is problems the team cannot solve. E.g. the team cannot do any kind of performance-tests because the hardware is not in place, the IT-department does not provide Bug tracker, or the ordered software just still did not reach the team. Another impediment could be that the marketing or sales manager was there again demanding that another feature gets integrated "quickly".
The second one regards impediments that result through the organizational structure or strategic decisions. Maybe the office is not capable of handling the important meetings or teamwork – e.g. because there is no media. One mistake that occurs quite often regards the problem that the Scrum Master is seen as the personnel responsible for the team members. This is often because of the classical role of a project leader, but using Scrum it only leads to conflicts of interests and is strongly against its major principle: The team owns a management role in the method of Scrum and is therefore coequal with the Scrum Master and the Product Owner. Another aspect can be the insufficient bandwidth of the internet for the new project.
The third problem refers to the individuals. Someone needs a hand with the debugging. Another one cannot solve a task alone and needs someone else for the pair programming. Someone else has to reset his computer....
Even though a Scrum Master can't and shouldn't realize some requirements himself, he is still responsible for solving and getting rid of problems and needs to give proper criteria. This task often takes up a lot of time and requires great authority and backbone. The Scrum Master has to create an optimal working-condition for the team and is responsible for this condition to be retained, in order to meet the goals of every sprint – i.e. for a short sprint the defined requirements.
Scrum Development Team Different from other methods, in Scrum a team is not just the executive organ that receives its tasks from the project leader, it rather decides self dependent, which requirements or User Stories it can accomplish in one sprint. It constructs the tasks and is responsible for the permutation of those – the team becomes a manager. This new self-conception of the team and the therewith aligned tasks and responsibilities necessarily change the role of the team leader/project leader. The Scrum Master does not need to delegate all the work and to plan the project, he rather takes care that the team meets all conditions in order to reach the self-made goals. He cleans off any impediments, provides an ideal working environment for the team, coaches and is responsible for the observation of Scrum-rules – he becomes the so-called Servant Leader.
The changed role perception is one of the most important aspects, when someone wants to understand Scrum and with the intent to introduce it in their own company.
Scrum Roles

10.3 Artifacts

Product Backlog 1
Force-ranked list of desired functionality Visible to all stakeholders Any stakeholder (including the Team) can add items Constantly re-prioritized by the Product Owner Items at top are more granular than items at bottom Maintained during the Backlog Refinement Meeting
Product Backlog Item (PBI) 1
Specifies the what more than the how of a customer-centric feature Often written in User Story form Has a product-wide definition of done to prevent technical debt May have item-specific acceptance criteria Effort is estimated by the team, ideally in relative units (e.g., story points) Effort is roughly 2-3 people 2-3 days, or smaller for advanced teams Sprint Backlog 1
Consists of committed PBIs negotiated between the team and the Product Owner during the Sprint Planning Meeting
Scope commitment is fixed during Sprint Execution
Initial tasks are identified by the team during Sprint Planning Meeting
Team will discover additional tasks needed to meet the fixed scope commitment during Sprint execution
Visible to the team
Referenced during the Daily Scrum Meeting
Sprint Task 1
Specifies how to achieve the PBI's what
Requires one day or less of work
Remaining effort is re-estimated daily, typically in hours
During Sprint Execution, a point person may volunteer to be primarily responsible for a task
Owned by the entire team; collaboration is expected
Personal Sprint Board with Sticky Notes (Windows)
Sprint Burndown Chart 1
Indicates total remaining team task hours within one Sprint Re-estimated daily, thus may go up before going down Intended to facilitate team self-organization Fancy variations, such as itemizing by point person or adding trend lines, tend to reduce effectiveness at encouraging collaboration Seemed like a good idea in the early days of Scrum, but in practice has often been misused as a management report, inviting intervention. The ScrumMaster should discontinue use of this chart if it becomes an impediment to team self-organization.
Product / Release Burndown Chart
Tracks the remaining Product Backlog effort from one Sprint to the next May use relative units such as Story Points for Y axis Depicts historical trends to adjust forecasts scrum-reference-card

10.4 Meetings

Meetings Scrum Meetings 1 Sprint Planning Daily Scrum Sprint Review Sprint Retrospective Product Backlog Refinement
Sprint Planning 1
Goals – Discuss and make sure the whole team understands the upcoming Work Items to deliver (quantity, complexity) – Select the work items to achieve during

the Sprint (Create the Sprint Backlog) – Rationally forecast the amount of work and commit to accomplish it – Plan how this work will be done
Attendees 1st part (what?): Whole team 2nd part (how?): Even though the product owner does not attend, he should remain to answer questions. Duration The meeting is time-boxed: 2 hours / week of sprint duration.
Typical meeting roadmap In Scrum, the sprint planning meeting has two parts:
1. What work will be done?
– the Product Owner presents the ordered product backlog items to the development team – the whole Scrum team collaborates to understand the work and select work items starting from the top of the product backlog
2. How will the work be accomplished?
– the development team discusses to define how to produce the next product increment in accordance with the current Definition of Done – The team does just the sufficient design and planning to be confident of completing the work during the sprint – The upcoming work to be done in the early days is broken down into small tasks of one day or less – Work to be done later may be left in larger units to be decomposed later (this is called Just-In-Time planning in Lean) – Final commitment to do the work
Important to know / Good practices – The development team is alone responsible to determine the number of product backlog items that will be "pulled" to the sprint. Nobody else should interfere in that decision, based on the current state of the project, the past performance and the current availability of the team. – It is a good practice to set a sprint goal to keep focus on the big picture and not on the details. – It is necessary for the Sprint planning meeting success that the product backlog is well ordered and refined. This is the Product Owner's responsibility. – The development team is responsible to decide how to do the work (self-organization). – There is no point, especially at the beginning, to try to make hourly estimates of the work and compare them to availability. This habit is inherited from traditional PM methods and may be counterproductive, as reduces ownership of the team's commitments. The best for the team is to intuitively estimate its own capacity to do the work, reduce the amount of committed work to deliver, and get experienced at estimating during the first sprints.
Daily Scrum 1
Goals Ensure, every day, at the same palace, that the team is on track for attaining the sprint goal and that team members are all on the same page. Spot blocks and problems. The Scrum Master can resolve impediments that are beyond the team's control. This is NOT a management reporting meeting to anyone. The team members communicate together as a team. Based on what comes up in the meeting, the development team reorganizes the work as needed to accomplish the sprint goal. Create transparency, trust and better performance. Build the team's self-organization and self-reliance. Attendees Development Team + Scrum Master. The Product Owner presence is not required but it I almost always interesting for him / here to be present, especially to clarify requirements if needed. Other stakeholders can attend to get a good and quick overview of the progress and project status, although having managerial presence may cause a "trigger" effect and pollute the meeting's effectiveness. Duration No more than 15 minutes. A good practice is to allow 2 minutes to each team member. Typical meeting roadmap Every team member answers 3 questions:
What I have accomplished since the last daily Scrum What I plan to accomplish

between now and the next daily Scrum What (if anything) is impeding my progress No discussion during the meeting. Only brief questions to clarify the previous list. Any subsequent discussion should take place after the meeting with the concerned team members.
Important to know / Good practices The Daily Scrum is sometimes called "Daily Stand Up". This name gives a good overview of the tone and shortness of this meeting. Each team member moves the tasks in the taskboard while speaking (if not done before) The Sprint Burndown chart can be updated by the Scrum Master at the end of the meeting Having a "being blocked" task list is useful. Personally I add a dedicated column in the taskboard Only team members speak during the daily Scrum. Nobody else. TheDaily Scrum is a proof of the team's sef-organizing capacity as it shows how much team members collaborate together as they address themselves to the whole team (and not only the Scrum Master) It is quite common to uncover additional tasks during the Sprint to achieve the Sprint Goal Sprint Review Meeting 1
Goals After Sprint execution, the team holds a Sprint Review Meeting to demonstrate a working product increment to the Product Owner and everyone else who is interested. The meeting should feature a live demonstration, not a report. Attendees Product Team Product Owner When After Sprint execution Duration 4 hours given a 30-day Sprint (much longer than anyone recommends nowadays), the maximum time for a Sprint Review Meeting is 4 hours Roadmap Demonstration After the demonstration, the Product Owner reviews the commitments made at the Sprint Planning Meeting and declares which items he now considers done. For example, a software item that is merely "code complete" is considered not done, because untested software isn't shippable. Incomplete items are returned to the Product Backlog and ranked according to the Product Owner's revised priorities as candidates for future Sprints. The ScrumMaster helps the Product Owner and stakeholders convert their feedback to new Product Backlog Items for prioritization by the Product Owner. Often, new scope discovery outpaces the team's rate of development. If the Product Owner feels that the newly discovered scope is more important than the original expectations, new scope displaces old scope in the Product Backlog. The Sprint Review Meeting is the appropriate meeting for external stakeholders (even end users) to attend. It is the opportunity to inspect and adapt the product as it emerges, and iteratively refine everyone's understanding of the requirements. New products, particularly software products, are hard to visualize in a vacuum. Many customers need to be able to react to a piece of functioning software to discover what they will actually want. Iterative development, a value-driven approach, allows the creation of products that couldn't have been specified up front in a plan-driven approach.
Sprint Retrospective Meeting 1 2
Goals At this meeting, the team reflects on its own process. They inspect their behavior and take action to adapt it for future Sprints.
When Each Sprint ends with a retrospective.
Duration 45 minutes
Roadmap
Dedicated ScrumMasters will find alternatives to the stale, fearful meetings everyone has come to expect. An in-depth retrospective requires an environment of psychological safety not found in most organizations. Without safety, the retrospective discussion will either avoid the uncomfortable issues or deteriorate

into blaming and hostility.
A common impediment to full transparency on the team is the presence of people who conduct performance appraisals.
Another impediment to an insightful retrospective is the human tendency to jump to conclusions and propose actions too quickly. Agile Retrospectives, the most popular book on this topic, describes a series of steps to slow this process down: Set the stage, gather data, generate insights, decide what to do, close the retrospective. (1) Another guide recommended for ScrumMasters, The Art of Focused Conversations, breaks the process into similar steps: Objective, reflective, interpretive, and decisional (ORID). (2)
A third impediment to psychological safety is geographic distribution. Geographically dispersed teams usually do not collaborate as well as those in team rooms.
Retrospectives often expose organizational impediments. Once a team has resolved the impediments within its immediate influence, the ScrumMaster should work to expand that influence, chipping away at the organizational impediments.
ScrumMasters should use a variety of techniques to facilitate retrospectives, including silent writing, timelines, and satisfaction histograms. In all cases, the goals are to gain a common understanding of multiple perspectives and to develop actions that will take the team to the next level.
Product Backlog Refinement Meeting 1
How to: A Great Product Backlog Refinement Workshop
PRESENTATION: HOW OUR TEAM LIVES SCRUM

Chương 11

Docker

View online http://magizbox.com/training/docker/site/
Docker is an open platform for building, shipping and running distributed applications. It gives programmers, development teams and operations engineers the common toolbox they need to take advantage of the distributed and networked nature of modern applications.

11.1 Get Started

Docker Promise Docker provides an integrated technology suite that enables development and IT operations teams to build, ship, and run distributed applications anywhere.
Build: Docker allows you to compose your application from microservices, without worrying about inconsistencies between development and production environments, and without locking into any platform or language.
Composable You want to be able to split up your application into multiple services.
Ship: Docker lets you design the entire cycle of application development, testing and distribution, and manage it with a consistent user interface.
Portable across providers You want to be able to move your application between different cloud providers and your own servers, or run it across several providers.
Run: Docker offers you the ability to deploy scalable services, securely and reliably, on a wide variety of platforms.
Portable across environments You want to be able to define how your application will run in development, and then run it seamlessly in testing, staging and production.
1. Docker Architecture
Docker Containers vs Virtual Hypervisor Model
In the Docker container model, the Docker engine sits atop a single host operating system. In contrast, with the traditional virtualization hypervisor mode, a separate guest operating system is required for each virtual machine.
Docker Images and Docker Containers
2.1 Docker Images Docker images are the build component of Docker.
For example, an image could contain an Ubuntu operating system with Apache and your web application installed. Images are used to create Docker containers.

Docker provides a simple way to build new images or update existing images, or you can download Docker images that other people have already created.
Built by you or other Docker users
Stored in the Docker Hub or your local Registry
Images Tags
Images are specified by repository:tag The same image my have multiple tags The default tag is lastest Look up the repository on Docker to see what tags are available 2.2 Docker Containers Docker containers are the run component of Docker.
Docker containers are similar to a directory. A Docker container holds everything that is needed for an application to run. Each container is created from a Docker image. Docker containers can be run, started, stopped, moved, and deleted. Each container is an isolated and secure application platform.
2.3 Registry and Repository Docker registries are the distribution component of Docker.
Docker registries
Docker registries hold images. These are public or private stores from which you upload or download images. The public Docker registry is provided with the Docker Hub. It serves a huge collection of existing images for your use. These can be images you create yourself or you can use images that others have previously created. Docker Hub
Docker Hub is the public registry that contains a large number of images available for your use.
2.3 Docker Networking Docker use DNS instead etc/hosts 3. Docker Orchestration Three tools for orchestrating distributed applications with Docker Docker Machine Tool that provisions Docker hosts and installs the Docker Engine on them Docker Swarm Tool that clusters many Engines and schedules containers Docker Compose Tool to create and manage multi-container applications Installation Docker only supports CentOS 7, Windows 7.1, 8/8.1, Mac OSX 10.8 64bit
Windows, CentOS
Usage Lab: Search for Images on Docker Hub Installation: Ubuntu Installation ubuntu 14.04
Prerequisites
apt-get update apt-get install apt-transport-https ca-certificates
apt-key adv –keyserver hkp://p80.pool.sks-keyservers.net:80 –recv-keys 58118E89F3A912897C070ADBF7622
Edit /etc/apt/sources.list.d/docker.list
deb https://apt.dockerproject.org/repo ubuntu-trusty main apt-get update apt-get purge lxc-docker apt-cache policy docker-engine Install docker
apt-get update apt-get install -y –force-yes docker-engine service docker start
Run Docker
docker run hello-world docker info docker version Installation https://www.docker.com/products/docker-toolbox
Go to the Docker Toolbox page. Click the installer link to download. Install Docker Toolbox by double-clicking the installer. Docker: CentOS 7 Installation
Install Docker
make sure your existing yum packages are up-to-date. sudo yum -y update
add docker repo sudo tee /etc/yum.repos.d/docker.repo «-'EOF' [dockerrepo]
name=Docker Repository baseurl=https://yum.dockerproject.org/repo/main/centos/$releasever/enabled = 1gpgcheck = 1gpgkey = https : //yum.dockerproject.org/gpgEOF

install the Docker package. sudo yum install -y docker-engine
start the Docker daemon sudo service docker start
verify sudo docker run hello-world Install Docker Compose
sudo yum install epel-release sudo yum install -y python-pip
sudo pip install docker-compose Docker 1.10.3
Kitematic Port Forwarding Open Virtualbox
Volumes Install Docker for Windows
Self Paced Training, Docker Fundamentals
Understand the architecture
Understand the architecture
Docker Compose and Networking
https://www.docker.com/
Orchestrating Docker with Machine, Swarm and Compose

11.2 Dev

Create New Image Create new image by commit docker commit <container ID> <yourname>/curl:1.0
Build image docker build -t ubuntu/test:1.0 . docker build -t ubuntu/test:1.0 –no-cache . Import/Export Container 1 2 Export the contents of a container's filesystem as a tar archive
docker export [OPTIONS] CONTAINER docker export $red_panda > latest.tar$
docker import file|URL|- [REPOSITORY[:TAG]] docker import http://example.com/exampleimage.tgz
cat exampleimage.tgz | docker import - exampleimagelocal:new Save/Load Image 3 4 Save one or more images to a tar archive
docker save [OPTIONS] IMAGE [IMAGE...] docker save busybox > busybox.tar.gz
docker load [OPTIONS] docker load < busybox.tar.gz Push Images to Docker Hub Login to docker hub docker login –username=yourhubusername –email=youremail@company.com
push docker docker push [repo:tag]
tag an image into a repository docker tag [OPTIONS] IMAGE[:TAG] [REGISTRYHOST/][USERNAME/]NAME[:TAG] Windows 5 Useful Docker Tips and Tricks on Windows
Commandline Reference: export
Commandline Reference: import
Commandline Reference: save
Commandline Reference: load

11.3 Dockerfile

Build a song with Dockerfile
A dockerfile is a configuration file that contains instructions for building a Docker image.
Provides a more effective way to build images compared to using docker commit
Easily fits into your continuous integration and deployment process. Command Line
'FROM' instruction specifies what the base image should be FROM java:7

‘RUN‘ instruction specifies a command to execute RUN apt-get update apt-get install -y curl vim openjdk-7-jdk
CMD Defines a default command to execute when a container is created CMD ["ping", "127.0.0.1", "-c", "30"]
ENTRYPOINT Defines the command that will run when a container is executed ENTRYPOINT ["ping"] Volumes
Configuration Files and Directories**
COPY <src>... <dest> COPY hom* /mydir/ adds all files starting with "hom" COPY hom?.txt /mydir/ ? is replaced with any single character, e.g., "home.txt" Networking
Ports still need to be mapped when container is executed EXPOSE Configures which ports a container will listen on at runtime FROM ubuntu:14.04 RUN apt-get update RUN apt-get install -y nginx
EXPOSE 80 443
CMD ["nginx", "-g", "daemon off;"] 2.4 Linking Containers Example: Web app container and Database container
Linking is a communication method between containers which allows them to securely transfer data from one to another
Source and recipient containers Recipient containers have access to data on source containers Links are established based on container names Create a Link Create the source container first Create the recipient container and use the –link option
Best practice - give your containers meaningful names
Create the source container using the postgres docker run -d –name database postgres
Create the recipient container and link it docker run -d -P –name website –link database:db nginx Uses of Linking
Containers can talk to each other without having to expose ports to the host Essential for micro service application architecture Example: Container with the Tomcat running Container with the MySQL running Application on Tomcat needs to connect to MySQL Operating Systems for Docker 1 2 Operating System Features CoreOS Service Discovery, Cluster management, Auto-updates CoreOS is designed for security, consistency, and reliability. Instead of installing packages via yum or apt, CoreOS uses Linux containers to manage your services at a higher level of abstraction. A single service’s code and all dependencies are packaged within a container that can be run on one or many CoreOS machines The New Minimalist Operating Systems
Top 5 operating systems for your Docker infrastructure

11.4 Container

Lifecycle docker create creates a container but does not start it. docker rename allows the container to be renamed. docker run creates and starts a container in one operation. docker rm deletes a container. docker update updates a container’s resource limits. Starting and Stopping docker start starts a container so it is running. docker stop stops a running container. docker restart stops and starts a container. docker pause pauses a running container, "freezing" it in place. docker unpause will unpause a running container. docker wait blocks

until running container stops. docker kill sends a SIGKILL to a running container. docker attach will connect to a running container. Info docker ps shows running containers. docker logs gets logs from container. (You can use a custom log driver, but logs is only available for json-file and * journald in 1.10). docker inspect looks at all the info on a container (including IP address). docker events gets events from container. docker port shows public facing port of container. docker top shows running processes in container. docker stats shows containers' resource usage statistics. docker diff shows changed files in the container's FS. Import / Export docker cp copies files or folders between a container and the local filesystem. docker export turns container filesystem into tarball archive stream to STDOUT. Example
docker cp foo.txt mycontainer:/foo.txt docker cp mycontainer:/foo.txt foo.txt Note that docker cp is not new in Docker 1.8. In older versions of Docker, the docker cp command only allowed copying files from a container to the host
Executing Commands docker exec to execute a command in container. To enter a running container, attach a new shell process to a running container called foo
docker exec -it foo /bin/bash. Suggest Readings: docker cheatsheet

11.5 Compose

Compose a symphony
Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a Compose file to configure your application's services. Then, using a single command, you create and start all the services from your configuration. To learn more about all the features of Compose see the list of features. Docker Compose Describes the components of an application
Box your component
.component/ docker-compose.yml app-1/ app/ file-1 file-2 Dockerfile run.sh app-2/ app/ file-1 file-2 Dockerfile run.sh Example docker-compose.yml example from docker-birthday-3 3
version: "2"
services: voting-app: build: ./voting-app/. volumes: - ./voting-app:/app ports: - "5000:80" networks: - front-tier - back-tier
result-app: build: ./result-app/. volumes: - ./result-app:/app ports: - "5001:80" networks: - front-tier - back-tier
worker: image: manomarks/worker networks: - back-tier
redis: image: redis:alpine $container_name$: $redis ports$: ["6379"] $networks$: $-back-tier$
db: image: postgres:9.4 $container_name$: $dbvolumes$: $-"db-data$: $/var/lib/postgresql/data"networks$: $-back-tier$
volumes: db-data:
networks: front-tier: back-tier: Networks $network_mode, ports, external_hosts, link, net$
$network_mode$: $"bridge"network_mode$: $"host"network_mode$: $"service$: $[service_name]"network_mode$: $"container$: $[containername/id]"VolumesOpsrebuildyourimagesdocker-composebuild$
build or rebuild services docker-compose build [SERVICE...] docker-compose build –no-cache database
start all the containers docker-compose up -d docker-compose up

stops the containers docker-compose stop [CONTAINER] Keep a container running in compose 2
You can use one of those lines
command: "top" command: "tail -f /dev/null" command: "sleep infinity" Docker Compose Files Version 2
github issue, Keep a container running in compose
docker-compose.yml

11.6 Swarm

Docker Swarm is native clustering for Docker. It turns a pool of Docker hosts into a single, virtual Docker host. Because Docker Swarm serves the standard Docker API, any tool that already communicates with a Docker daemon can use Swarm to transparently scale to multiple hosts 1
Architecture 2
Swarm Manager: Docker Swarm has a Master or Manager, that is a pre-defined Docker Host, and is a single point for all administration. Currently only a single instance of manager is allowed in the cluster. This is a SPOF for high availability architectures and additional managers will be allowed in a future version of Swarm with 598.
Swarm Nodes: The containers are deployed on Nodes that are additional Docker Hosts. Each Swarm Node must be accessible by the manager, each node must listen to the same network interface (TCP port). Each node runs a node agent that registers the referenced Docker daemon, monitors it, and updates the discovery backend with the node's status. The containers run on a node.
Scheduler Strategy: Different scheduler strategies (binpack, spread, and random) can be applied to pick the best node to run your container. The default strategy is spread which optimizes the node for least number of running containers. There are multiple kinds of filters, such as constraints and affinity. This should allow for a decent scheduling algorithm.
Node Discovery Service: By default, Swarm uses hosted discovery service, based on Docker Hub, using tokens to discover nodes that are part of a cluster. However etcd, consul, and zookeeper can be also be used for service discovery as well. This is particularly useful if there is no access to Internet, or you are running the setup in a closed network. A new discovery backend can be created as explained here. It would be useful to have the hosted Discovery Service inside the firewall and 660 will discuss this.
Standard Docker API: Docker Swarm serves the standard Docker API and thus any tool that talks to a single Docker host will seamlessly scale to multiple hosts now. That means if you were using shell scripts using Docker CLI to configure multiple Docker hosts, the same CLI would can now talk to Swarm cluster and Docker Swarm will then act as proxy and run it on the cluster.
Create Swarm Simple Swarm Swarm with CentOS Swarm with Windows Swarm and Compose 5 Create docker-compose.yml file
Example
wget https://docs.docker.com/swarm/swarm$_{a}t_{s}cale/docker-compose.ymlDiscovery4$
Scheduling 3 Docker Swarm: CentOS Prerequisites Docker 1.10.3 CentOS 7 Create cluster 1 Receipts: consult
Configure daemon in each host 6

Edit /etc/systemd/system/docker.service.d/docker.conf
[Service] ExecStart= ExecStart=/usr/bin/docker daemon -H fd:// -D -H tcp://<node$_ip$>: 2375 $--cluster-store=consul://<consul_ip>:8500 --cluster-advertise=<node_ip>:2375 Restart docker$
systemctl daemon-reload systemctl restart docker Verify docker run with config
ps aux | grep docker | grep -v grep Run cluster
In discovery host
run consul docker run -d -p 8500:8500 –name=consul progrium/consul -server -bootstrap In swarm manage host
run swarm master docker run -d -p 2376:2375 swarm manage consul://<consul$_ip$>: 8500/$swarm In swarm node host$
open 2375 port firewalld-cmd –get-active-zones firewall-cmd –zone=public –add-port=2375/tcp –permanent firewall-cmd –reload
run swarm node docker run -d swarm join –addr=<node$_ip$>: 2375$consul://<consul_ip>:8500/swarm Question: Why manager and consul cannot run in the same host? (because docker-proxy?)$
Verify 2
manage docker -H :2376 info export DOCKER$_HOST=tcp://<ip>:12375 docker info docker run -d -P redis$
debug a swarm docker swarm –debug manage consul://<consul$_ip$>: 8500$Docker Swarm: Simple Swarm Simple Swarm in one node with token 1 Change file vi /etc/default/docker$
DOCKER$_OPTS=" -H tcp://0.0.0.0:2375 -H unix:///var/run/docker.sock"$
Restart docker service docker restart
create swarm token docker run –rm swarm create > token$_id$
Run swarm node docker run -d swarm join –addr=ip:2375 token://token$_i$ddocker ps
Run swarm manager docker run -d -p 12375:2375 swarm manage token://token$_id$
Swarm info docker -H :12375 info
export DOCKER$_HOST=tcp://ip:12375 docker info docker run -d -P redis Docker Swarm: Windows Prerequisites Docker 1.10.3 Giithiuvchyth Docker Swarm$
Docker-swarm, Cannot connect to the docker engine endpoint
Docker Swarm Part 3: Scheduling
Docker Swarm Part 2: Discovery
Deploy the application
Configuring and running Docker on various distributions

11.7 UCP

Docker Universal Control Plane
Docker Universal Control Plane provides an on-premises, or virtual private cloud (VPC) container management solution for Docker apps and infrastructure regardless of where your applications are running.
[embed]https://www.youtube.com/watch?v=woDzgqtZZKc[/embed]

11.8 Labs

Lab 1:
1. Create a container from an Ubuntu image and run a bash terminal docker run -it ubuntu /bin/bash 2.Inside the container install curl apt-get install curl

3. Exit the container terminal Ctrl-P 4. Run ‘docker ps -a‘ and take not of your container ID 5. Save the container as a new image. For the repository name use <yourname>/curl. Tag the image as 1.0 6. Run ‘docker images‘ and verify that you can see your new image Lab 2: Run Ubntu
FROM ubuntu:14.04 RUN apt-get update apt-get install -y curl vim 1. Go into the test folder and open your Dockerfile from the previous exercise
2. Add the folloing line to the end CMD ["ping", "127.0.0.1", "-c", "30"]
3. Build the image docker build -t <yourname>/testimage:1.1 .
4. Execute a container from the image and observe the output docker run <yourname>/testimage:1.1
5. Execute another container from the image and specify the echo command docker run <yourname>/testimage:1.1 echo "hello world"
6. Observe how the container argument overrides the CMD instruction Lab:
1) In your home directory, create a folder called test 2) In the test folder, create a file called ‘Dockerfile‘ 3) In the file, specify to use Ubuntu 14.04 as the base image FROM ubuntu:14.04 4) Write an instruction to install curl and vim after an apt-get update RUN apt-get update apt-get install -y curl vim 5) Build an image from Dockerfile. Give it the repository <yourname>/testimage and tag it as 1.0 docker build -t johnnytu/testimage:1.0 . 6) Create a container using your newly build image and verify that curl and vim are installed. Lab: Run a container and get Terminal Access
* Create a container using the ubuntu image and connect to STDIN and a terminal docker run -i -t ubuntu /bin/bash * In your container, create a new user using your first and last name as the username addusername username * Exit the container exit * Notice how the container shutdown * Once again run: docker run -i -t ubuntu /bin/bash * Try and find your user ‘cat /etc/passwd‘ * Noice that it does not exist. Lab: Run a web application inside a container
The -P flag to map container ports to host ports docker run -d -P tomcat:7 docker ps check your image details runing go to <server url>:<port number> verify that you can see the Tomcat page Lab Create a Docker Hub Account
Lab Push Images to Docker Hub
Login to docker hub $docker login --username = yourhubusername --email = youremail@company.com Password : WARNING : logincredentials saved in C : docker.json Login Succeeded$
Use ‘docker push‘ command docker push [repo:tag]
Local repo must have same name and tag as the Docker Hub repo
Used to rename a local image repository before pushing to Docker Hub docker tag [image ID] [repo:tag] docker tag [local repo:tag] [Docker Hub repo:tag]
> tag image with ID (trainingteam/testexample is the name of repository on Docker hub) docker tag edfc212de17b trainingteam/testexample:1.0
> tag image using the local repository tag docker tag johnnytu/testimage:1.5 trainingteam/testexample

11.9 GUI

Environment
Ubuntu 14.04 Docker Engine 1.11 note: I can’t make it run on CentOS.
Dockerfile 1 [code]
Base docker image FROM debian:sid MAINTAINER Jessica Frazelle jess@docker.com

ADD https://dl.google.com/linux/direct/google-talkplugin$_{c}urrent_{a}md64.deb/src/google-talkplugin_{c}urrent_{a}md64.deb$
Install Chrome RUN echo 'deb http://httpredir.debian.org/debian testing main' » /etc/apt/sources.list apt-get update apt-get install -y ca-certificates curl hicolor-icon-theme libgl1-mesa-dri libgl1-mesa-glx libv4l-0 -t testing fonts-symbola –no-install-recommends curl -sSL https://dl.google.com/linux/linux$_{s}igning_{k}ey.pub|apt-keyadd- echo"deb[arch = amd64]http : //dl.google.com/linux/chrome/deb/stablemain" > /etc/apt/sources.list.d/google.list apt − getupdateapt − getinstall − y google − chrome−stable −−no−install−recommends dpkg−i′/src/google−talkplugin_{c}urrent_{a}md64.deb′ apt−getpurge−−auto−remove−ycurl rm−rf/var/lib/apt/lists/ rm−rf/src/.deb
COPY local.conf /etc/fonts/local.conf
RUN rm /etc/apt/sources.list.d/google-chrome.list
RUN apt-get update RUN apt-get install -y libcanberra-gtk-module RUN apt-get install -y dbus libdbus-1-dev libxml2-dev dbus-x11
Autorun chrome ENTRYPOINT ["google-chrome"] CMD ["–user-data-dir=/data"] [/code]
Build Image [code] docker build -t chrome . [/code]
Docker Run [code] xhost + docker run -it –net host –cpuset-cpus 0 -v /tmp/.X11-unix:/tmp/.X11-unix -e DISPLAY=unix$DISPLAY$ −vHOME/chrome/data/Downloads:/root/Downloads -v $HOME/chrome/data/$: /data−v/deb/shm : /dev/shm−−device/dev/sndchrome−−user − data − dir = /datawww.google.com[/code]
Known Issue Issue 1: Failed to load module "canberra-gtk-module" 2
[code] Gtk-Message: Failed to load module "canberra-gtk-module" [/code]
Issue 2: Chrome crash 3
Solution: add -v /dev/shm:/dev/shm

- https://hub.docker.com/r/jess/chrome/ /dockerfile/
- http://fabiorehm.com/blog/2014/09/11/running-gui-apps-with-docker/comment-2515573749

Chương 12

Lean

View online http://magizbox.com/training/lean/site/
Lean startup is a method for developing businesses and products first proposed in 2008 by Eric Ries. Based on his previous experience working in several U.S. startups, Ries claims that startups can shorten their product development cycles by adopting a combination of business-hypothesis-driven experimentation, iterative product releases, and what he calls validated learning. Ries' overall claim is that if startups invest their time into iteratively building products or services to meet the needs of early customers, they can reduce the market risks and sidestep the need for large amounts of initial project funding and expensive product launches and failures.

12.1 Lean Canvas

12.2 Workflow

Life's too short to build something nobody wants. What separates successful startups from unsuccessful ones is that they find a plan that works before running out of resources.
Build-Measure-Learn The fundamental activity of a startup is to turn ideas into products, measure how customers respond, and then learn whether to pivot or persevere. All successful startup processes should be geared to accelerate that feedback loop.
Step 1: Document your Plan A. Writing down the initial vision using Lean Canvas and then sharing it with at least one other person
Step 2: Identify the Riskiest Parts Formulate Falsifiable Hypotheses
Falsifiable Hypothesis = [Specific Repeatable Action] will [Expected Measurable Action] The biggest risk is building something nobody wants. The risks can be divided in:
Stage 1: Problem/Solution Fit. Do I have a problem worth solving?
Answer these questions:
Is it something customers want? (must-have) Will they pay for it? If not, who will? (viable) Can it be solved? (feasible) Stage 2: Product/market Fit. Have I built something people want?"

Qualitative metrics (interviews) Quantitative metrics for measuring product/-market fit. Stage 3: Scale. How do I accelerate growth?
After product/market fit, raise a big round of funding to scale the business. Traction is needed! Seek External advice and ask specific questions.
Systematically test your Plan. Based on the scientific method run experiments. First build hypothesis that can be easily disproved and use artifacts in front of customers to "measure" their response using qualitative and quantitative data to derive "learning" to validate or refute the hypothesis.
The Lean Startup is also about being efficient. In all the stages we can take actions to be more efficient: eliminate don't-needs at the MVP, deploy continuously, make feedback easy for customers, don't push for features, etc.
The Lean Startup methodology is strongly rooted in the scientific method, and running experiments is a key activity. Your job isn't just building the best solution, but owning the entire business model and making all the pieces fit.

12.3 Validated Learning

Startups exist not to make stuff, make money, or serve customers. They exist to learn how to build a sustainable business. This learning can be validated scientifically, by running experiments that allow us to test each element of our vision.

Tài liệu tham khảo

Ghi chú