

Docker: إطار افتراضية على مستوى نظام التشغيل وتطبيقاته في المكتبات

جون فينك

عرض تاريخي لـ Docker في 2014. احتُفظ بالشفيرات والأسماء والإصدارات القديمة بوصفها جزءًا من الأصل؛ ليست تعليمات تركيب محدثة. نُشر الأصل على الويب دون ترقيم، ونسخة مصدره هنا طباعة أرشيفية من 7 صفحات. أعدت الترجمة لاحقًا ضمن استكمال الأرشيف.

مقدمة:

إذا عملت في تقنية المكتبات في الألفية الماضية، فستذكر غرفة الخوادم: أبراج حواسيب تشغل Novell Network متصلة بمصفوفات أقراص CD-ROM ضخمة، وصناديق Sun بحجم الثلاجة، و AlphaServers من Digital لتشغيل الفهرس. واحتاجت أعمال المكتبة الجادة معدات كبيرة، فازدحمت الغرف بالرفوف والأسلاك والتكيف. وصغرت قليلًا مع Linux على الحواسيب الصغيرة، لكن التحول الحقيقي جاء أوائل الألفية الجديدة بانتشار افتراضية سهلة التنفيذ [1] تشغل نظم تشغيل مستقلة متعددة على جهاز واحد.

تعود الافتراضية بمعناها الحديث إلى الستينيات وآلات [2] IBM System/360، وتضمنت شريحتا 286 و386 أشكلاً منها [3]، لكن انطلاقها المرتبطة بـ Linux تسارعت مع منتجات VMware لمعمارية x86 في 2001. ومع [4] KVM و [5] Xen محليًا و [6] Amazon Web Services و [7] OpenStack سحابيًا، كادت الغرفة المزدحمة تنقرض، وأصبح بالإمكان إنشاء آلات افتراضية كثيرة من عتاد قليل.

يعرض الأمر التالي سيع آلات افتراضية على خادم واحد في مكان عمل المؤلف، وكلها قيد التشغيل:

```
virsh # list
Id Name      State
-----
1 trotsky    running
2 eris       running
3 funhouse   running
4 gorgar     running
5 shoah      running
6 balder2    running
7 asgard     running
```

الافتراضية في المكتبات:

تختلف الحاجات كثيرًا؛ فقد يكفي وحدة تقنية صغيرة أو متوسطة قليلة التطوير بضع آلات: للموقع ونظام المكتبة المتكامل وإدارة صور محطات المستفيدين. وتكون بدائل مباشرة للخوادم القديمة قدر الإمكان. وقد تُقيد الأعراف أو التراخيص تثبيت نظام مكتبة بأقل انحراف عن التقليدي، وقد لا يستحق تخصيص أنظمة أخرى جهدًا لا يعوضه تحسن الأداء أو الراحة.

أما فرق التطوير فلها حاجات أعقد، لأن العمل تكرر دائم: شيفرة واختبارات وأعطال ثم شيفرة واختبارات جديدة. وتخيّل إعداد آلة جديدة يدويًا، ولو افتراضية، كلما أردت اختبارًا من الصفر. وأي توفير لهذا الوقت يتيح للمطور عمله المتخصص. فـ [8] VirtualBox مثلاً ذو توجه رسومي قوي [9]، فُتني حوله إطار كامل [10] لجعل الإنشاء والإزالة قابليْن للبرمجة النسية.

تستخدم ترتيبات المكتبات عادة افتراضية مستوى الآلة. وتحاكي KVM و VMware و Xen و VirtualBox و DOSBox، المخصص لألعاب [11] DOS، بيئة الحوسبة برمجيًا بما فيها الأقراص والذاكرة والرسوم والمعالج. وقد تحاكي ARM، مثل [12] Raspberry Pi، على جهاز Intel. لكنها قد تستهلك الموارد؛ فتستنزف الآلات المتعددة التخزين أو الذاكرة المخصصة. وتُحل بعض المشكلات بسهولة نسبيًا، كتركيب أقراص خارجية كمشاركات شبكية، وتصعب أخرى كالذاكرة المحدودة.

وهنا افتراضية مستوى نظام التشغيل [13] التي تشارك الموارد بدل محاكاة العتاد كله. فتشارك المثلثات الذاكرة والتخزين والنواة، ويقل احتمال استنفاد موارد المضيف مقارنة بعدد مماثل من الآلات الكاملة. لكن مشاركة النواة تعني نوع المعالج ونظام التشغيل نفسه [14]؛ فلا تشغل Raspberry Pi افتراضياً على x86 أو Windows داخل مضيف Linux بهذه الطريقة. ويمكن مع ذلك اختلاف توزيعات Linux، مثل مضيف Red Hat على Ubuntu. ورغم القيد، يجذب خفة النشر وسهولته التطوير.

الافتراضية باستخدام Docker:

قائمة، كنظم الملفات Linux تنفيذ مفتوح المصدر على مستوى نظام التشغيل يزداد انتشاره بين المطورين. ويجمع تقنيات [15] Docker مع قابلية نقل النشر وإدارة الإصدارات وإعادة الاستخدام وقابلية [17] Linux وحاويات [16] AUFS الاتحادية بالنسخ عند الكتابة، وغالباً التكرار [18]. وتميزه هذه الخصائص عن الآلات التقليدية المتمسكة باستعارة العتاد.

وتحول تجهيز البيئة من نموذج مستهلك للوقت محوره مدير النظام إلى سير عمل محوره المطور [19]، كما يظهر في إدارة الإصدارات المشابهة لـ [20] git بفروقتها ووسومها.

جوهر Docker تشغيل التطبيقات، لا محاكاة العتاد. فالآلة الافتراضية تعيد إنشاء الذاكرة وعدد المعالجات وبطاقات الشبكة وغيرها، بينما تركز افتراضية النظام التطبيق [21]. وغالباً لا يهمن أثناء التطوير والاختبار والإطلاق العتاد الدقيق، عدا محاكاة الأجهزة التاريخية [22] والحالات الحديثة. وعندما نطلب مساعدة في Drupal لا نبدأ عادة بسرد نموذج Dell وأربعة معالجات و16GB من الذاكرة وتفاصيل البطاقات، إلا بدليل على ارتباط المشكلة بالعتاد؛ بل نركز البرمجيات، كما يفعل Docker. ولا نحدد عادة له ذاكرة أو قرصاً أو حصة معالج، مثلما لا نفعل لتشيفرتنا على الحاسوب المكتبي.

يعني ذلك تشغيل مثيلات متزامنة أكثر. فعلى حاسوب ضعيف من 2007 عانى المؤلف لتشغيل مثيلين من VirtualBox معاً، لكنه شغل على المضيف نفسه أكثر من 40 حاوية Docker دون تراجع ملحوظ في الأداء.

تشغيل Docker ومثال قائمة الحاويات:

بعد تثبيت [23] Docker يتاح البرنامج الرئيس docker لبدء الحاويات وإيقافها واستيرادها وتصديرها وعمليات أخرى [24]. ويعرض docker ps الحاويات. وفي المثال الأصلي كانت السجلات التالية، وقد أعيد تنسيقها لتجنب جدول عريض؛ كل سجل يحفظ المعرف والصورة والأمر ووقت الإنشاء والحالة والمنافذ والاسم:

السجل 1: المعرف 1bc191f4cddb؛ الصورة bedework:latest؛ الأمر supervisor -n؛ أنشئ قبل 11 يوماً، وقيد التشغيل منذ 11 يوماً؛ المنافذ 0.0.0.0:8080<-tcp/8080؛ الاسم sick_newton.

السجل 2: المعرف b58946da298c؛ الصورة papyrus-demo:port6000؛ الأمر bin/bash؛ أنشئ قبل 13 يوماً، وقيد التشغيل منذ 13 يوماً؛ المنافذ 6001/tcp/3000->0.0.0.0:6001/tcp، 6000/tcp؛ الاسم drunk_bell.

السجل 3: المعرف c90c0a6be88f؛ الصورة saucy-csclub:latest؛ الأمر bin/bash؛ أنشئ قبل 13 يوماً، وقيد التشغيل منذ 13 يوماً؛ المنافذ 0.0.0.0:9999<-tcp/9999؛ الاسم angry_shockley.

السجل 4: المعرف e5a0f8a71f7e؛ الصورة papyrus-demo:in-process؛ الأمر start.sh؛ أنشئ قبل أسبوعين، وقيد التشغيل منذ 13 يوماً؛ المنافذ 0.0.0.0:3000<-tcp/3000؛ الاسم mad_poincare.

السجل 5: المعرف f752161937c6؛ الصورة ldap_update_pw:latest؛ الأمر supervisor -n؛ أنشئ قبل خمسة أسابيع، وقيد التشغيل منذ 13 يوماً؛ المنافذ 0.0.0.0:5000<-tcp/5000؛ الاسم distracted_nobel.

السجل 6: المعرف 33cf9eb89073؛ الصورة catmandu:in-process؛ الأمر bin/bash؛ أنشئ قبل ستة أسابيع، وقيد التشغيل منذ 13 يوماً؛ حقل المنافذ فارغ؛ الاسم cranky_mccarthy.

تنقسم مثيلات Docker إلى صور وحاويات؛ فالحاوية مثل جاز للصورة، وقد توجد حاويات متعددة منها أو من تنويعاتها. ويشغل المعرفان b58946da298c و e5a0f8a71f7e حالتين من papyrus-demo بوسمي port6000 و in-process؛ وتعمل الوسوم شبيهاً بوسوم git لتمييز حالات الصورة. ويذكر الأصل أن الصورة دون وسم مميز تحمل latest.

مثال بسيط: المقال حاوية:

أنشأ المؤلف حاوية تحول Markdown الذي كتب به المقال إلى HTML باستخدام [25] pandoc، وتخدمه بوحدة Python المسماة SimpleHTTPServer. وهي متاحة في مستودع المقال على [26] GitHub.

تتكون من Dockerfile يُنفذ عند بناء الصورة، و start.sh يُنفذ كلما أنشئت حاوية منها. ويحتوي c4l-docker-article.md نص المقال الذي يحوله الملف الثاني ويقدمه. وفيما يأتي شرح العناصر.

Dockerfile سطرًا سطرًا:

```
FROM ubuntu:latest
```

يحدد الصورة الأساسية، وهي Ubuntu القياسية لدى Docker.

```
MAINTAINER John Fink <john.fink@gmail.com>
```

يعرف المسؤول عن صيانة الصورة، كما ورد عنوانه المهني العام في المثال الأصلي.

```
RUN echo "deb http://archive.ubuntu.com/ubuntu precise universe" >> /etc/apt/sources.list
```

ينفذ RUN أمرًا داخل الصورة الجاري بناؤها؛ وهنا يضيف مستودع Universe إلى مصادر Ubuntu لتثبيت pandoc لاحقًا.

```
RUN DEBIAN_FRONTEND=noninteractive apt-get update
RUN DEBIAN_FRONTEND=noninteractive apt-get -y install python git pandoc
```

يحدث الأمران النظام ويثبتان الحزم اللازمة.

```
ADD ./start.sh /start.sh
```

يدخل ADD ملفًا من خارج الصورة إلى داخلها.

```
RUN mkdir /article/
ADD ./c4l-docker-article.md /article/c4l-docker-article.md
```

ينشئ RUN المجلد ويضيف ADD ملف المقال إليه.

```
EXPOSE 8888
```

يعلن حاجة برامج الحاوية إلى المنفذ 8888، وهو هنا ل خادم Python البسيط.

```
CMD ["/bin/bash", "/start.sh"]
```

يحدد CMD البرنامج الافتراضي عند عدم إعطاء أمر خاص، وهو تشغيل start.sh عبر /bin/bash.

ملف start.sh وبناء الحاوية:

```
cd /article/
pandoc c4l-docker-article.md -o index.html
python -m SimpleHTTPServer 8888
```

ينتقل الملف إلى المجلد، ويحول Markdown إلى HTML، ثم يشغل الخادم لتقديم المقال.

يبني docker build الصورة بقراءة أسطر Dockerfile وحفظ كل خطوة كتغيير وطبقة فوق السابقة، مما يبسر الرجوع عند الخطأ. وتستخدم عمليات البناء اللاحقة الطبقات السابقة ذاكرة مؤقتة، ولا تعيد إلا ما تغير، فيسر البناء من التعديلات. ورد أمر البناء التالي حرفيًا في الأصل:

```
docker build -t c4l-docker-wordpress git://github.com/jbfink/c4l-docker-article
```

ثم ورد أمر التشغيل:

```
docker run -Pd c4l-docker-article
```

ملاحظة تحريرية: يختلف وسم الصورة في أمر البناء، c4l-docker-wordpress، عن الاسم في أمر التشغيل، c4l-docker-article. أبقى النص كما نُشر؛ ويلزم تطابق الاسمين لتشغيل المثال، وهذا تنبيه إلى تناقض في الأصل وليس تعديلًا صاممًا للشفرة.

كل إنشاء لحاوية ينفذ CMD، ومن ثم start.sh الذي يحول المقال ويشغل الخادم. ويخصص الخيار P منفذًا عشوائيًا في المضيف يصل إلى 8888 في الحاوية، ويشغل d الحاوية منفصلة في الخلفية. ويعرض docker ps الحاوية ومنفذها، ويمكن فتح عنوان المضيف بذلك المنفذ في المتصفح لقراءة المقال.

مثال واقعي: WordPress:

بدأ المؤلف في ربيع 2013 حاوية WordPress لمشروعات التطوير الداخلية، لأنه واسع الاستخدام والدعم والفهم وسهل العناية نسبيًا وله برمجيات مساندة كثيرة، فيناسب اختبار Docker. وبناها أولًا يدويًا بإطلاق bash وإجراء خطوات التثبيت والتحرير المعتادة، ثم نشرها في فهرس [27] Docker وتلقى أسئلة بالبريد عن بنائها. وفي أغسطس 2013 بدأ [28] docker-wordpress ليقدم بناءً منظمًا يمكن للآخرين تجربته وتطويره. والمشكلة في تجميد تثبيت عادي ثم تشغيله في مكان آخر بقاء إعدادات واحدة، مثل كلمات MySQL وأملاح WordPress ومفاتيحه. والأفضل اختلاف القيم في كل تشغيل أول؛ لذلك يتضمن [29] start.sh أوامر لإنشائها، ومنها المثال الأصلي التالي:

```
sed -e "s/database_name_here/$WORDPRESS_DB/
s/username_here/$WORDPRESS_DB/
s/password_here/$WORDPRESS_PASSWORD/
/'AUTH_KEY'/s/put your unique phrase here/`pwgen -c -n -l 65`/
/'SECURE_AUTH_KEY'/s/put your unique phrase here/`pwgen -c -n -l 65`/
/'LOGGED_IN_KEY'/s/put your unique phrase here/`pwgen -c -n -l 65`/
/'NONCE_KEY'/s/put your unique phrase here/`pwgen -c -n -l 65`/
/'AUTH_SALT'/s/put your unique phrase here/`pwgen -c -n -l 65`/
/'SECURE_AUTH_SALT'/s/put your unique phrase here/`pwgen -c -n -l 65`/
/'LOGGED_IN_SALT'/s/put your unique phrase here/`pwgen -c -n -l 65`/
/'NONCE_SALT'/s/put your unique phrase here/`pwgen -c -n -l 65`/" /var/www/wp-config-sample.php > /var/
www/wp-config.php
```

تضمن هذه العملية قيمًا مختلفة للمتغيرات المهمة عند التشغيل الأول. ومع مخرجات البناء الأول من المصدر [30] يطول البدء، لكن التشغيلات اللاحقة تستغرق عادة أقل من 30 ثانية، ولا يلزم البناء المخزن مؤقتًا غالبًا إلا مع تحديثات كبيرة للمكونات. ويستغرق إنشاء حاوية جديدة نحو ثانية، ثم تُتاح بعنوان داخلي أو منفذ خارجي. ميزة docker-wordpress أنه صورة واحدة وحاوية واحدة سهلة الفهم، لكن ليس نموذجًا جيدًا للإنتاج بصيغته وقت الأصل؛ فالتسجيل العارض للأحداث، الذي قد يحسنه [31] Logstash، ووجود MySQL محلية يضعفان ملاءمته. فلو شغلت 20 نسخة، كان خادم MySQL واحد يخدمها أقل من قاعدة مستقلة في كل واحدة.

خاتمة:

خضع Docker لتطوير مكثف أكثر من سنة وكان منشؤه يشنون عن اعتباره جاهزًا للإنتاج. لكن بلوغه الإصدار 1.0 في 9 يونيو 2014 [32] يجعله، وفق المؤلف آنذاك، جاهزًا لذلك. ويتوافق الإصدار مع السابق، فتستمر المشروعات التي بدأت أثناء تغيره.

ومع ذلك يصعب نقل التطبيقات غير المعتادة. فهو يفضل العمليات في المقدمة، فيحتاج تحويل برامج مثل MySQL وApache من عملها الخلفي المعتاد. ويجعل تركيز التطبيق الواحد لكل حاوية، مع استخدام supervisord عند الحاجة كما في المثال، الأنظمة ذات التركيب المعقد أقل مثالية وربما أنسب للألات التقليدية أو العتاد المباشر وقتها. ورغم العمل المتبقي، يقترب من الاستقرار وبعد بإمكانات للمكتبات. فقد يرفق المطور Dockerfile بشيفرته في git ليختبرها سريعًا على خادم بعيد، أو يعرضها للآخرين دون تعقيد البناء والاعتماديات. وقد بُنى صورة ضمن نسخ موقع المكتبة الاحتياطي للنشر السريع عند الانقطاع. ومن لا يرحب باختزال إجراء تثبيت طويل [33] إلى أمر واحد؟

عن المؤلف:

جون فينك مكتبي البحث الرقمي في مركز شيرمان للبحث الرقمي بجامعة ماكماستر في هاميلتون، أونتاريو. ويعمل مع البرمجيات المفتوحة والمكتبات منذ 1995.

مراجع الأصل:

1. <http://en.wikipedia.org/wiki/Virtualization>
2. http://www.theregister.co.uk/2011/07/14/brief_history_of_virtualisation_part_2/
3. http://www.theregister.co.uk/2011/07/11/a_brief_history_of_virtualisation_part_one/
4. <http://www.linux-kvm.org/>
5. <http://www.xenproject.org/>
6. <http://aws.amazon.com>
7. <http://www.openstack.org>
8. <http://virtualbox.org>
9. <http://www.virtualbox.org/attachment/wiki/Screenshots/gnome.png>

10. <http://vagrantup.com>
11. <http://www.dosbox.com>
12. <http://cronicasredux.blogspot.ca/2011/09/installing-and-running-debian-armel-on.html>
13. http://en.wikipedia.org/wiki/Operating_system-level_virtualization
14. You can run different Linux versions even though you might commonly consider them different OSes; a Red Hat guest could run under an Ubuntu host, for instance.
15. <http://docker.io>
16. <http://www.thegeekstuff.com/2013/05/linux-aufs/>
17. <http://linuxcontainers.org/>
18. <http://stackoverflow.com/a/18208445/380282>
19. <http://radar.oreilly.com/2012/06/what-is-devops.html>
20. <http://git-scm.com/>
21. <https://stackoverflow.com/a/22370529>
22. <http://simh.trailing-edge.com/>
23. https://www.docker.io/gettingstarted/#h_installation
24. <http://docs.docker.io/reference/commandline/cli/>
25. <http://johnmacfarlane.net/pandoc/>
26. <http://github.com/jbfink/c4l-docker-article>
27. <https://index.docker.io/u/jbfink/wordpress/>
28. <http://github.com/jbfink/docker-wordpress>
29. <https://github.com/jbfink/docker-wordpress/blob/master/scripts/start.sh>
30. <https://gist.github.com/jbfink/9054707>
31. <http://logstash.net/>
32. <http://blog.docker.com/2014/06/its-here-docker-1-0/>
33. http://docs.evergreen-ils.org/2.5/_installing_the_evergreen_server.html

البيانات البيوجرافية للمقال الأصلي:

Fink, J. (2014, July 21). Docker: a Software as a Service, Operating System-Level Virtualization Framework. The Code4Lib Journal, 25. <https://journal.code4lib.org/articles/9669>

تاريخ نشر الأصل: 2014-07-21. ترجمة عربية أُعدت بمساعدة الذكاء الاصطناعي ضمن استكمال أرشيف المجلة. حقوق الأصل للمؤلفين المذكورين. ترخيص النص والترجمة: المشاع الإبداعي تسبب المصنّف (CC BY 3.0 US). تُرجمت الشروح وحُفظت الشيفرات والمراجع. حُولت قائمة docker ps العريضة إلى سجلات نصية بكل حقولها. أُضيف تنبيه تحريري إلى اختلاف اسمي الصورة بين أمري البناء والتشغيل في الأصل. بقيت أمثلة الشيفرة التاريخية دون تنفيذ.

<https://creativecommons.org/licenses/by/3.0/us/>