

ماهو الـ **Docker** ؟

إعداد

د. سلطان بن سعود القحطاني

أستاذ مساعد بكلية علوم الحاسب والمعلومات

جامعة الإمام محمد بن سعود الإسلامية

مقدمة

كما نعرف بأن سفن الشحن هي إحدى الطرق الرئيسية لتوزيع البضائع حول العالم. قديما، تكلفة نقل البضائع عبر تلك السفن مكلفة نوعا ما لأن كل شحنة لديها خصائص تختلف عن الشحنات المرافقة لها على نفس السفينة. فمثلا، شحن صندوق من السمك عبر المحيطات يحتاج رعاية خاصة من ناحية التبريد وطريقة التحميل والتفريغ كما أنه يحتاج نوع مخصص من الكريانات (الرافعات)، وعليه فلا بد أن تكون هناك سفن مخصصة لذلك. بينما في حالة شحن سيارته عبر المحيطات فلا يحتاج نفس العناية بصندوق السمك، بل يحتاج نوع مخصص من الرافعات وكذلك عناية خاصة بشحن السيارات، وبالتالي يحتاج سفينة مخصصة لذلك. وقياسا على الأمثلة السابقة، كل نوع من الشحنات يحتاج عناية خاصة لا بد أن تتوفر له سفينة مخصصة. وهذا مكلف جدا. إذا ما هو الحل؟

الحاويات (containers)، هي الحل لمشكلة التنوع في الشحنات. الحاويات لها شكل واحد وتختلف في الخصائص، فمثلا حاويات الأسماك هي نفسها حاويات السيارات من ناحية الشكل ولكن تختلف من ناحية الخصائص. فمثلا حاويات الأسماك تكون مزودة بأدوات تبريد، مضادات للبكتيريا، وغيرها. بينما في حاويات السيارات لا تحتاج مثل تلك الخصائص. العامل المشترك بين تلك الحاويات هو الشكل مما يساعد على شحنها في سفينة شحن واحدة، وأيضا آلية التحميل والتفريغ موحدة (الرافعات ليست مخصصة لكل حاوية، بل تستطيع العمل على جميع الحاويات).

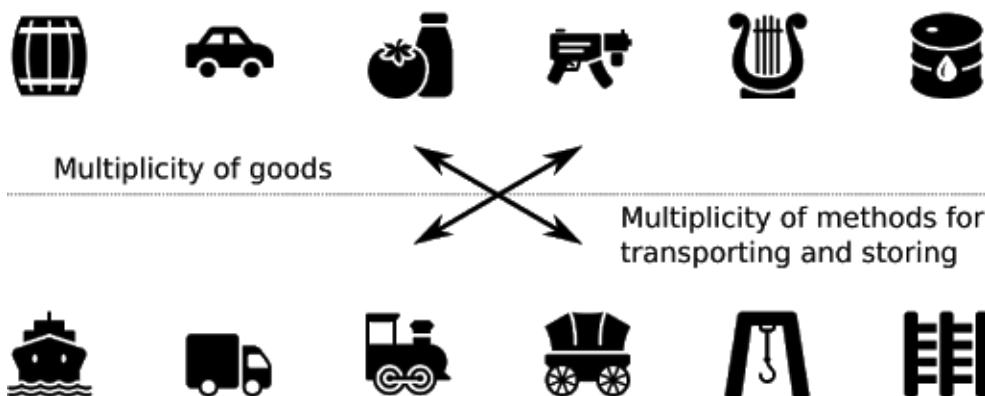


Figure1 : تنوع الطرق للشحن والتخزين على حسب نوع المادة أو البضاعة

نفس المفهوم يحدث في عالم تطوير البرمجيات. دوكر (Docker) أصبح أداة توصيل البرامج بشكل عام بغض النظر عن هيكلية تلك البرامج أو اعتماد البرامج على برمجيات أخرى (dependencies) أو طريقة تثبيتها.

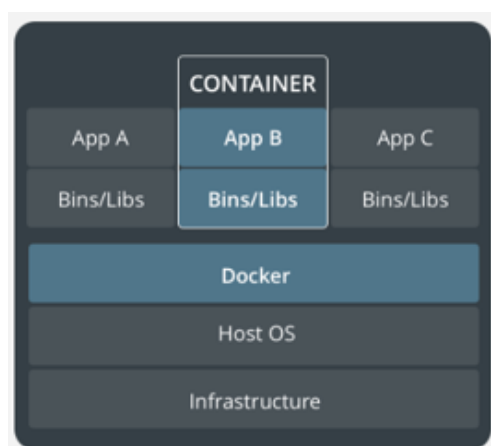


Figure2 : container's components

بداية المشكلة والحل لها

الـ Docker هو مشروع مفتوح المصدر تمت برمجته باستخدام لغة الـ Go, لغة برمجته من انتاج شركة Google. وأصبح الـ Docker من اكبر المشاريع مفتوحة المصدر في الوقت الحالي.

إذا ماهي المشكلة الكبيرة التي من أجلها الـ Docker اتى ليقوم بحلها في عالم هندسة البرمجيات (تطوير البرامج)؟ فمثلا, لو نظرنا للتطبيق في الصورة التالي, تطبيق يعتمد على لغة الجافا كلغة برمجته, ويعتمد على الـ MySQL كقاعدة بيانات , ويستخدم قاعدة بيانات للذاكرة مثل redis. اذا لدينا ثلاث مكونات من التقنيات المختلفة التي نستخدمها لبناء التطبيق. الان, لنفترض اننا نريد نعمل Deployment للتطبيق بعد الانتهاء منه على خادم (server) وليكن هذا الـ Server يعمل عليه نظام Linux ولكي يعمل التطبيق على هذا الخادم, فلا بد ان تتوفر جميع التقنيات المستخدمة في التطبيق على السيرفر لكي يعمل بالشكل الصحيح, فمثلا لابد ان تكون التقنيات التي تخص الـ java مثل الـ maven, Gradel وغيرها متوفرة, وأيضا لو كان التطبيق يعمل على الويب فلا بد ان يتوفر الـ web server مثل الـ Apache Webserver وأيضا الـ MySQL server لتشغيل قواعد البيانات الخاصة بالتطبيق وهكذا. وأيضا نقطه مهمه هي الإصدارات للتقنيات هذه لابد ان تكون متوافقة مع الإصدارات المستخدمة في التطبيق لكي لا يحصل تعارض ونحافظ على الـ compatibility (مثلا Java 5 تختلف في إصدارها عن Java 8). أي بمعنى لابد ان يحتوي السيرفر على جميع الـ Dependencies والـ Frameworks التي يحتاجها التطبيق الخاص بي لكي يعمل بالشكل الصحيح.

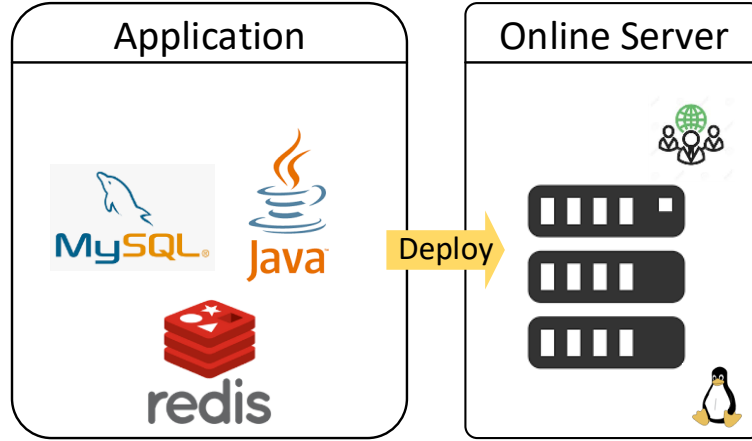


Figure3 : المشكلة في حالة رفع التطبيق على السيرفر سيحتاج توفير جميع الإصدارات المتوافقه مع المكونات المستخدمه لكل تطبيق

ولك ان تتخيل كم من الجهد المطلوب لعمل كل الـ configuration للسيرفر لكي يتوافق مع التقنيات المستخدمة في التطبيق ولكي يعمل بالشكل الصحيح. كل هذا العمل مجهد ومستهلك للوقت.

ومع تطور تطوير البرمجيات ومفاهيم هندسة البرمجيات اصبح تطوير البرمجيات يحتاج عمل وجهد كبير جدا وأيضا التطبيقات المستهدفة ذات حجم كبير جدا وقد تحتوي على ما يسمى مايكرو سيرفيس (micro-services) وكل micro-service يعمل على نسخته معينه عن الـ micro-service الاخر وجميعها لابد ان يكونو على نفس الخادم, وهذا يخلق مشكله توافر النسخ المتعدده وتعارضها مع بعضها البعض لعدم وجود الـ isolation بينهم (او مانسميع بالعزل).

الحل لكل هذه المشاكل, هو الـ Containerisation. فمثلا كما هو مبين في الصورة التآليه, كل مكون من مكونات البرنامج عبارة عن image (صورة) فمثلا, الـ java وجميع ما يتعلق بها من تقنيات (libraries, frameworks, etc.) انا احتاجها لبرمجة التطبيق معمول لها packaging داخل هذه الـ image وبنفس الطريقه لـ MySQL وredis, ولكي نعمل الان deployment للتطبيق على الخادم فقط نحتاج الـ Docker محمل على السيرفر والذي بدوره يقوم بتشغيل جميع الـ images فقط من دون الاهتمام بأي تقنية كل image تعمل وكيف تعمل ومال الى اخره. والسبب بأن الـ images انشأ لها instances وهذه الـ instances تسمى الـ containers ويمكن ننشأ اكثر من instances من image وحده (مشابه لمفهوم الـ objects من الـ class في البرمجة الشيئية Object-oriented programming). وبالتالي اصبح التطبيق بإمكانه العمل على الخادم بشكل فعال بحيث كل مكون من التطبيق عبارة عن image ومعزول عن الـ images الأخرى في الخادم, وبالتالي لا اهتم بمشكله الـ compatibility التي ذكرتها سابقا.

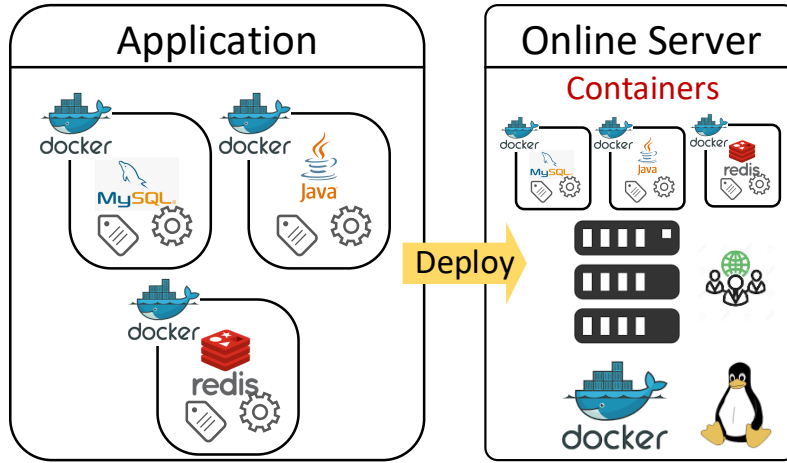


Figure4 : الحل باستخدام الدوكر

في نهاية المطاف, كل container الان في السيرفر يشترك في نفس الـ kernel لنظام التشغيل الموجود على السيرفر مع الـ containers الأخرى. طبعا لكي يشتغل الـ Docker لابد ان يكون نظام التشغيل Linux, ولمن يتسأل ماذا عن الـ Windows؟ نفس الطريقة ولاكن يحتاج تشتغل على Virtual Machine مثل VMware ومن ثم تعمل الـ Docker.

اذا بهذه الطريقة, قمنا بإيجاد حل لمشكلة الـ deployment التي عرضناها سابقا.

قد يسأل شخص, ما لفرق بين الـ Containerisation و الـ Virtualisation؟

في حالة الـ Virtualisation, اي اننا نستخدم الـ virtual tools المعروفة مثل VirtualBox و VMware, وغيرها, نحتاج لكل مكون (component) نصعنه داخل virtual box نظام تشغيل خاص فيه, وذلك لكي نحقق مفهوم الـ isolation التي ذكرناها سابقا بحيث تكون كل مكون معزولة عن الأخر.

ولاكن في حالة الـ Containerisation, لانحتاج نظام تشغيل لكل container مثل ما عملنا مع الـ virtualization, بل أن جميع الـ containers تتشارك في نفس الـ kernel المستخدم لنظام تشغيل الـ server. فقط نحتاج لنظام الـ Docker كطبقة ترتيب المهام لعمل الـ containers.

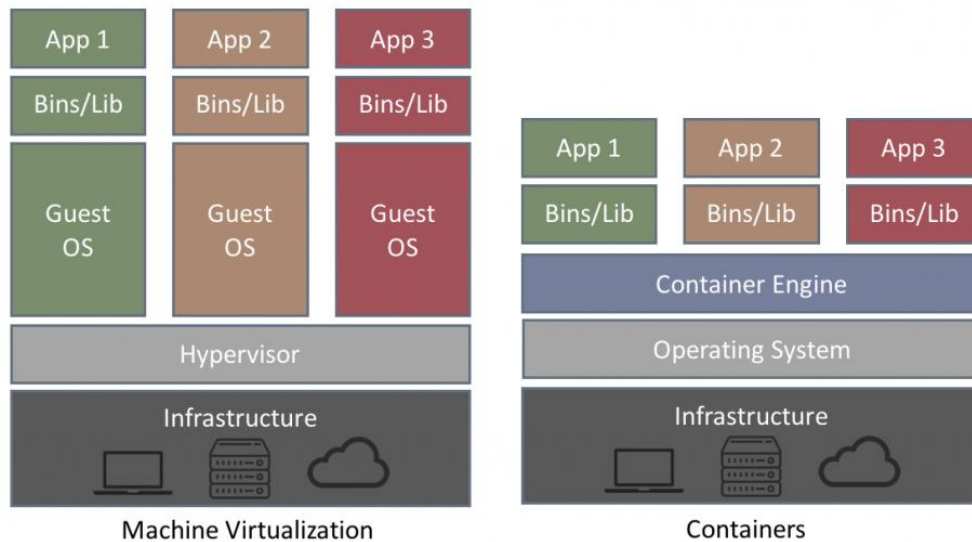


Figure5 : Docker vs. Virtual machines

وكما هو واضح، الـ virtualization يحتاج resources كثير، لان كل نظام تشغيل في كل virtual box مثلا يحتاج إلى حيز في الذاكرة، والمعالج والـ hard drive، وهذا يستهلك وقت أكثر في عملية الـ booting (التشغيل). وكل هذه المشاكل نستطيع ان نقول ان الـ containerisation تتغلب عليها، لاننا نعمل على ذاكرة واحدة، ونظام تشغيل واحد، وكذلك نستخدم او نتشارك في نفس الـ resources، وبالتالي، سرعة الأداء وكفاءة العمل على الـ Docker تتفوق.

لاكن هناك ميزه يجب ان لا نغفل عنها وتميز الـ virtualization عن الـ containerisation الا وهي الـ security. لماذا؟ كما ذكرنا، بأنه في الـ containerization جميع الحاويات تشترك على نفس الكيرنل لنظام التشغيل على السيرفر، اي انها تعمل على نظام التشغيل الأساسي للسيرفر، فلو حصل ان يكون هناك ثغره في نظام التشغيل، الخاص بالسيرفر، فمعناه ستكون الثغره مؤثره على جميع الكونتينرز التي تعمل على السيرفر. لكن في الـ virtualization، كل كوبونتت لها الـ security configuration الخاص بها، لانها لا تتشارك في نفس نظام التشغيل بعكس الـ containerization، وبالتالي، اي ثغره على نظام التشغيل على السيرفر لن يؤثر على اي من الـ virtual boxes لانها منعزله بنظام تشغيل مستقل بها.

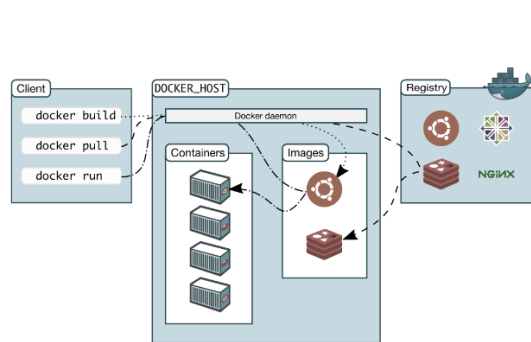
في الجزء القادم سوف نتكلم بشيء من التفصيل عن الـ Docker engine (architecture).

Docker Engine أو محرك نظام الـ Docker

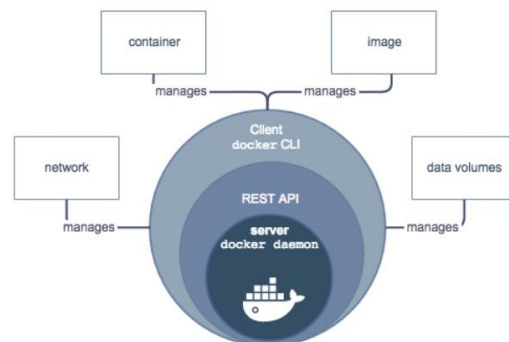
الـ Docker engine مبني على مفهوم الـ client-server architecture ويتكون ثلاث مكونات (components) رئيسية كالتالي:

1. Docker daemon: هو القلب المحرك للـ Docker او الـ code المحرك للنظام، وهو مسؤول عن الـ Docker objects مثل images, containers, networks و volumes. من ناحية إضافية، وتعديل، وحذف وغيرها.
2. Representation State Transfer Application Program Interface (REST API): الهدف الأساسي من الـ REST API في الـ Docker هو توفير الية التخاطب للطلبات المرسله لـ Docker daemon وهي الـ http requests بين الـ clients و الـ server وتحديد الية التعامل معها.
3. Command line interface client (CLI): الـ CLI تستخدم الـ Docker REST API للتحكم والتعامل مع الـ Docker daemon من خلال الأوامر عن طريق الـ command line او بإستخدام الـ scripts.

الهيكل والية العمل في نظام الـ Docker



مكونات الـ Docker engine



لأكثر تفاصيل عن الـ Docker architecture انصح بالرجوع إلى <https://docs.docker.com/get-started/overview/#docker-architecture>.

بعد ان اطلعنا على مقدمه بسيطه عن الـ Docker ومما يتكون وكيفية عمله نظريا. ننتقل في الجزء القادم إلى التطبيق العملي وكيفية استخدام الـ Docker وانشاء الـ images وطريقة نشرها ومشاركتها. سوف نتعرف على

الـ Docker hub, registry, كيفية انشاء الـ images وتشغيل الـ container وكذلك نشر الـ images على منصة الـ Docker hub.

ما هو الـ Docker Hub

الـ Docker Hub او ما يسمى بالـ Docker registry هو عبارة عن مخزن كبير للـ images التي يتم صنعها من قبل المبرمجين والمطورين على الإنترنت ويتم مشاركتها للاستفادة منها. كل مطور برمجيات يستطيع ان ينشأ image ويعمل عليها بشكل locally ومن ثم يستطيع ان يضعها خاصه (private) لا يمكن الاطلاع عليها الا اشخاص معينين او عامه (public) يستطيع ان يستفيد أي شخص منها على الـ Docker Hub, سوف نطلع على أمثله بالتفصيل لاحقا.

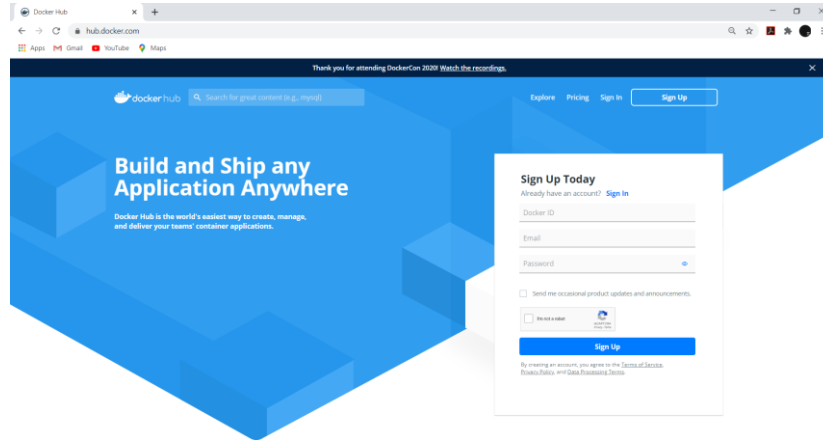


Figure6 : الصفحة الرئيسية لمنصة الدوكر

على نفس المنصة هناك شركات تتيح التطبيقات الخاصة بها على منصة Docker Hub بحيث لا يحتاج المطور ان يعمل configuration لهذه التطبيقات كما يتم في الماضي. مثلا, MySQL من الصورة التالي نشاهد ان هذا التطبيق الان جاهز للإستخدام وكل ما في الأمر هو عمل امر pull ومن ثم تشغيله على الجهاز والاستفادة منه في بناء قواعد البيانات التي احتاج اصممها للتطبيق الخاص بي.

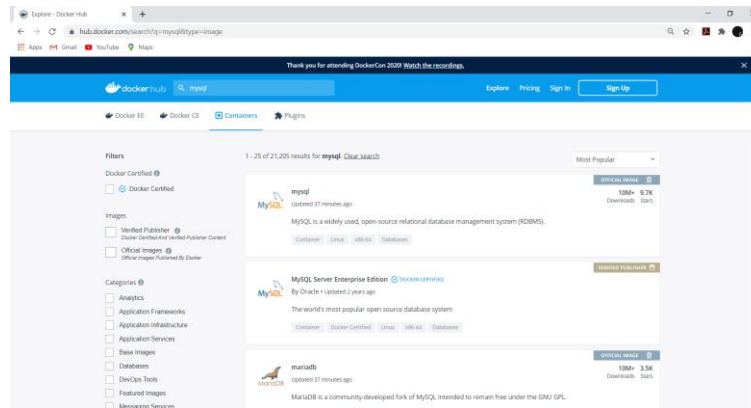


Figure7 : MySQL official image

أيضا نلاحظ بان انه يوضع علامة official image او verified publisher أي ان شركة MySQL هي من وضعت هذه الـ image وعليه يستطيع المستخدم ان يثق في التطبيق بعكس بعض التطبيقات التي قد تحتوي على مشاكل في الجودة او في الأمن.

مثال على استخدام الـ Docker Hub عن طريق الـ CLI

لكي يتم استخدام نظام الـ Docker بالشكل الصحيح, لابد ان يتم تحميل النظام وتتبع الخطوات من هذا الرابط:
بعد ان يتم انزال نظام الـ Docker على نظام Ubuntu او اي اصدار من توزيعات Linux, تستطيع ان تتحقق باستخدام الأمر `$docker --version`.

```
File Edit View Search Terminal Help
sultan@ubuntu1804:~$ docker --version
Docker version 19.03.6, build 369ce74a3c
sultan@ubuntu1804:~$
```

بعد ان تأكدنا بأن الـ Docker يعمل على الجهاز بالشكل الصحيح, نبدأ بعملية الربط عن طريق CLI بمنصة الـ Docker Hub باتباع الخطوات التالية:

- 1- اولاً لابد ان يكون لديك حساب في الـ Docker Hub (اسم مستخدم وكلمة مرور). للتسجيل استخدم الرابط التالي: <https://hub.docker.com/signup>
- 2- عن طريق الـ terminal نقوم باستخدام الأمر `$docker login docker.io` وسطلب الـ username و الـ password الخاصة بك على Docker hub. بعد ذلك ومن مجرد الحصول على Login Succeeded تعتبر الان متصل بجهازك المحلي بمنصة Docker Hub وجاهز لتحميل التطبيقات المساعدة لمشروعك او ايضا تحميل او رفع التطبيقات الخاصة بك إلى المنصة.

```
File Edit View Search Terminal Help
sultan@ubuntu1804:~$ sudo docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: tsultan
Password:
WARNING! Your password will be stored unencrypted in /home/sultan/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
sultan@ubuntu1804:~$
```

بناء الـ Docker image

الـ Docker image نستطيع مقارنتها بـ git repository. استخدام الـ git من الممكن استضافتها (hosting) على الـ GitHub او الـ Bitbucket ومن الممكن على استضافتها على private repository. بنفس المفهوم, نستطيع استضافة الـ Docker image على Docker repository service وهو ما يسمى الـ Docker Hub.

اذا الـ Docker hub هو خدمه مقدمه من الـ Docker لاستضافة (hosting) الـ images, والبحث والمشاركة لجميع الـ Docker repositories. وكما ذكرنا سابق في مثال الـ git, الـ Docker repository من الممكن ان تكون عامه او خاصه. أخيراً, يطلق على الـ Docker Hub وخدمات استضافة الـ third party repository الـ registries. مثلاً, RedHat لديها السجل (RedHat registry) المخصص لها لاستضافة صور الحاويات (container images) الخاصة بها.

ملحوظه مهمه, بأن السجل الواحد لديه العديد من المخازن (repositories) بينما المخزن الواحد لديه إصدارات متعددة لنفس الصورة. كل هذه السجلات, قد تكون عامه أو خاصه. فالـ Docker Hub مثال على سجل عام (public registry).

الدوكر في حالته العادية هو في الأساس مهيئ على استخدام منصة الدوكر كسجل افتراضي (default registry). بإمكانك تستخدم الأمر `$docker info` للاطلاع على السجل الذي يستخدمه Docker حالياً. بشكل افتراضي, ستلاحظ انه يشير إلى <http://index.docker.io/v1/> وهو موقع السجل على منصة الدوكر (Docker hub).

```
sultan@ubuntu1804:~$ sudo docker info
Client:
 Debug Mode: false


Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 0
 Server Version: 19.03.6
 Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Native Overlay Diff: true
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
 Swarm: inactive
 Runtimes: runc
 Default Runtime: runc
 Init Binary: docker-init
 containerd version:
 runc version:
 init version:
 Security Options:
  apparmor
  seccomp
   Profile: default
 Kernel Version: 5.3.0-61-generic
 Operating System: Ubuntu 18.04.4 LTS
 OSType: linux
 Architecture: x86_64
 CPUs: 1
 Total Memory: 1.94GiB
 Name: ubuntu1804
 ID: RYK...
 Docker Root Dir: /var/lib/docker
 Debug Mode: false
 Registry: https://index.docker.io/v1/
 Labels:
 Experimental: false
 Insecure Registries:
  127.0.0.0/8
 Live Restore Enabled: false
```

الصور الرسمية والغير رسميه (official images vs. non-official)

على منصة الـ Docker، هناك نوعين من الـ images – رسميه وغير رسميه كما ذكرنا سابقا. الـ image الرسمية، هي عبارة عن image موثوقة ومحسنة، ولديها توثيق وشرح مفصل (documentation)، وأيضا تتوفر على صفحتها على الـ Docker hub امثله على كيفية الاستخدام، ومصممه بأن تكون للاستخدامات المتعددة الشاملة والمتعارف عليها.

في الجانب الآخر، الـ image الغير رسميه، هي أي image أنشئت من قبل مستخدم عادي وليس شركه او منظمه برمجيه. وبالتالي، Docker لديه بعض المقاييس لكي يتم التمييز بين النوعين. فالـ image الرسمية تحتوي على `<image_name>` كإسم للـ image، بينما الـ image الغير رسميه تسميتها تكون بـ `<username>/<image_name>`. أيضا الصور الرسمية، يكتب امامها كلمة **official** كما هو مبين في المثال التالي:

Image رسمي من شركة MySQL



mysql

Updated 37 minutes ago

OFFICIAL IMAGE

10M+ Downloads

9.7K Stars

MySQL is a widely used, open-source relational database management system (RDBMS).

Container

Linux

x86-64

Databases

Image غير رسمي من مستخدم mysqlboy



mysqlboy/docker-myddumper

By [mysqlboy](#) • Updated a year ago

10K+ Downloads 3 Stars

docker-myddumper containerizes MySQL logical backup tool myddumper

Container Linux x86-64

تحميل image او pull image من منصة Docker

استخدام البحث عن images على منصة Docker

نستطيع ان نبحث ونجد صور على منصة الدوكر سواء باستخدام محرك البحث على موقع المنصة او باستخدام الأمر التالي:

```
$docker search <image_name>
```

فلو اردنا ان نبحث عن الـ image بإسم busybox او Debian:

```
File Edit View Search Terminal Help
sultan@ubuntu1804:~$ sudo docker search busybox
NAME                DESCRIPTION                STARS                OFFICIAL                AUTOMATED
busybox             Busybox base image.        1927                [OK]
progrum/busybox     Full-chain, Internet enabled, busybox made f... 71                  [OK]
radial/busyboxplus  Full-chain, Internet enabled, busybox made f... 31                  [OK]
arm32v7/busybox     Busybox base image.        8
yauritux/busybox-curl Busybox with CURL          8
armhf/busybox       Busybox base image.        6
odise/busybox-curl  Busybox base image.        4                  [OK]
arm64v8/busybox     Busybox base image.        3
prom/busybox        Prometheus Busybox Docker base images          2                  [OK]
s390x/busybox       Busybox base image.        2
arm32v6/busybox     Busybox base image.        2
ppc64le/busybox     Busybox base image for ppc64. 2
aarch64/busybox     Busybox base image.        2
joeshaw/busybox-nonroot Busybox container with non-root user nobody 2
l386/busybox        Busybox base image.        2
vukomlr/busybox     busybox and curl          1
spotify/busybox     Spotify fork of https://hub.docker.com/_/bus... 1
ppc64le/busybox     Busybox base image.        1
arm32v5/busybox     Busybox base image.        0
amd64/busybox       Busybox base image.        0
concourse/busyboxplus Busybox                  0
sou856099/busybox   Busybox                  0
emccorp/busybox     Busybox                  0
ggtools/busybox-ubuntu Busybox ubuntu version with extra goodies 0
trollin/busybox     Busybox                  0
sultan@ubuntu1804:~$
```

```
File Edit View Search Terminal Help
sultan@ubuntu1804:~$ sudo docker search debian
NAME                DESCRIPTION                STARS                OFFICIAL                AUTOMATED
ubuntu             Ubuntu is a Debian-based Linux operating sys... 11054              [OK]
debian             Debian is a Linux distribution that's compos... 3523              [OK]
arm32v7/debian     Debian is a Linux distribution that's compos... 66
kiscaro/debian-ssh Debian Jessie              28
arm64v8/debian     Debian is a Linux distribution that's compos... 23
samueldebruyn/debian-git a minimal docker container with debian and g... 22
multiarch/debian-debootstrap multiarch ports of debian-debootstrap          12
l386/debian        Debian is a Linux distribution that's compos... 10
eboraas/debian     Debian base images, for all currently-availa... 8
vergissberlin/debian-development Docker debian image to use for development, _ 6
ppc64le/debian     Debian is a Linux distribution that's compos... 4
amd64/debian       Debian is a Linux distribution that's compos... 4
smartentry/debian  debian with smartentry          4
vicano/debian      Debian docker images for all versions/archit... 3
s390x/debian       Debian is a Linux distribution that's compos... 2
vpgrp/debian       Docker images of Debian.        2
arm32v5/debian     Debian is a Linux distribution that's compos... 2
spritsatl/debian-builder A Docker image based on debian:slim ideal fo... 1
halgerinberg/debian Debian multiarch docker base image          1
dockershelf/debian Repository for docker images of Debian. Test... 1
fleshgrinder/debian Debian base images for production and multis... 0
casept/debian-amd64 A debian image built from scratch. Mostly fo... 0
jdub/debian-sources-resource Concourse CI resource to check for updated D... 0
landinternet/debian-9-nginx-php-7.2-wordpress-4 debian-9-nginx-php-7.2-wordpress-4          0
konstruktoid/debian Debian base image              0
sultan@ubuntu1804:~$
```

كما هو واضح من شاشة الـ terminal جميع المعلومات المهمة عن الـ image اذا هي رسميه سيضاف لها [ok] تحت عبارة official وعدد الـ stars يدل على مدى شعبيتها وقابلية المستخدمين لها, وكذلك وصف بسيط للـ image.

سحب ونسخ image او repository من الـ registry على الجهاز المحلي

لكي نقوم بتحميل image معينه او مجموعه images نستخدم الأمر:

```
$docker <image_name>:<tag_name>
```

إذا tag غير متوفر, فإن محرك دوكر (docker engine) سيتخيم latest: تاق كـtag افتراضي. مثلاً, إذا اردنا ان نقوم بعمل pull او نسخ الـimage ذات الإصدار الأخير من نظام التشغيل الأخير debian نقوم بالتالي:

```
File Edit View Search Terminal Help
sultan@ubuntu1804:~$ sudo docker pull debian
Using default tag: latest
latest: Pulling from library/debian
e9afc4f90ab0: Pull complete

Digest: sha256:46d659005ca1151087efa997f1039ae45a7bf7a2cbb2d17d3dcdba632a3ee9a
Status: Downloaded newer image for debian:latest
docker.io/library/debian:latest
sultan@ubuntu1804:~$
```

إنشاء Docker image

نستطيع ان ننشأ الـimage الخاصه بتطبيقاتنا ونشرها على منصة Docker بطريقتين:

باستخدام Dockerfile لإنشاء image

الـDockerfile هو عبارته عن نص بسيط من مستند يحتوي على سلسلة من الأوامر بحيث يقوم Docker باستخدامها لبناء الـimages. مجموعة الأوامر المدعومة في الـDockerfile هي FROM ,CMD ,ENTRYPOINT ,VOLUME ,ENV , والعديد منها². مثال بسيط على الـDockerfile كالتالي:

```
Open Dockerfile Save
~/dockerproject
FROM busybox:latest
CMD ["date"]
```

ملاحظته: نقطه مهمه لنتذكرها وهي بأن الملف المخصص لصنع الـimage لايد بأن يكون اسمه **Dockerfile**.

أمر الـdocker build يستخدم لبناء الـimage من الـDockerfile. ولكي نقوم ببناء الـimage للملف الذي صنعناه في المثال السابق³ من الـDockerfile نستخدم الأمر التالي:

```
File Edit View Search Terminal Help
sultan@ubuntu1804:~/dockerproject$ sudo docker build -t docker_example_image .
Sending build context to Docker daemon 2.048kB
Step 1/2 : FROM busybox:latest
latest: Pulling from library/busybox
76df9210b28c: Pull complete
Digest: sha256:95cf004f559831017cdf4628aaf1bb30133677be8702a8c5f2994629f637a209
Status: Downloaded newer image for busybox:latest
--> 1c35c4412082
Step 2/2 : CMD ["date"]
--> Running in 151acfe0a08c
Removing intermediate container 151acfe0a08c
--> c21f52ba1d76
Successfully built c21f52ba1d76
Successfully tagged docker_example_image:latest
sultan@ubuntu1804:~/dockerproject$
```

طبعاً هناك بعض الـparameters مع كل امر (commands) نرجو الاطلاع عليها على الرابط⁴.

² <https://docs.docker.com/engine/reference/builder/>

³ التطبيق فقط يطبع تاريخ اليوم من الأمر CMD

⁴ <https://docs.docker.com/engine/reference/commandline/cli/>

بعد ان تتم عملية البناء بالشكل الصحيح سيخرج بأن image تم بناؤها بنجاح وعمل لها tag بإسم image والإصدار. نستطيع ان نعمل تشغيل للحاوية (كما بينا سابقا الحاوية او ال container هي عبارة عن instance من image) وسوف نجد بأن المثال يقوم بطباعة التاريخ كالتالي:

```
File Edit View Search Terminal Help
sultan@ubuntu1804:~/dockerproject$ sudo docker run -it --name docker_app docker_example_image
Mon Jun 29 11:38:44 UTC 2020
sultan@ubuntu1804:~/dockerproject$
```

انشاء image من container

الطريقة الثانية لإنشاء image هي سحب docker image من منصة Docker Hub وانشاء حاوية منها ومن ثم عمل تعديل او تغيير عليها مثال ان نقوم بتحميل التطبيق الخاص بنا على تلك الحاوية. ومن ثم نستخدم الأمر docker commit لعمل انشاء docker commit من الحاوية. لنرى المثال التالي عن كيفية انشاء docker image من حاوية example_appChange it:

```
File Edit View Search Terminal Help
sultan@ubuntu1804:~/dockerproject$ sudo docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
926ed08be72e       docker_example_image "date"             2 minutes ago       Exited (0) 2 mi
4b7c837b4216       docker_example_image "date"             2 minutes ago       Exited (0) 2 mi
d4aa7c318936       docker_example_image "date"             3 minutes ago       Exited (0) 3 mi
sultan@ubuntu1804:~/dockerproject$ sudo docker commit docker_app docker_example_image2:latest
sha256:a548f5d4d54b98361d9f222169c434824787c5167095b40cbe43e93e3c9c42b8
sultan@ubuntu1804:~/dockerproject$
```

فالأمر `docker ps -a` يبين ماهي ال containers التي تعمل وعدد ال instances لكل image. فمثلا، قمت بصناعة 3 containers من ال docker_exmple_image.

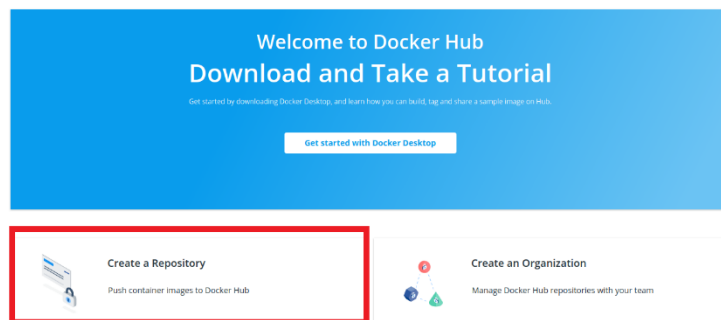
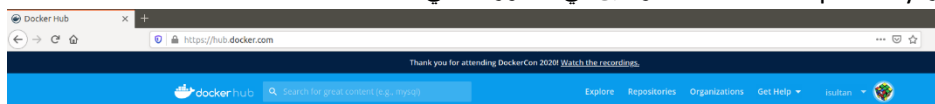
نشر ال images على ال Docker hub

لكي نستطيع نشر docker image الخاصه بنا على منصة ال docker, هناك خطوات لابد ان نتبعها:

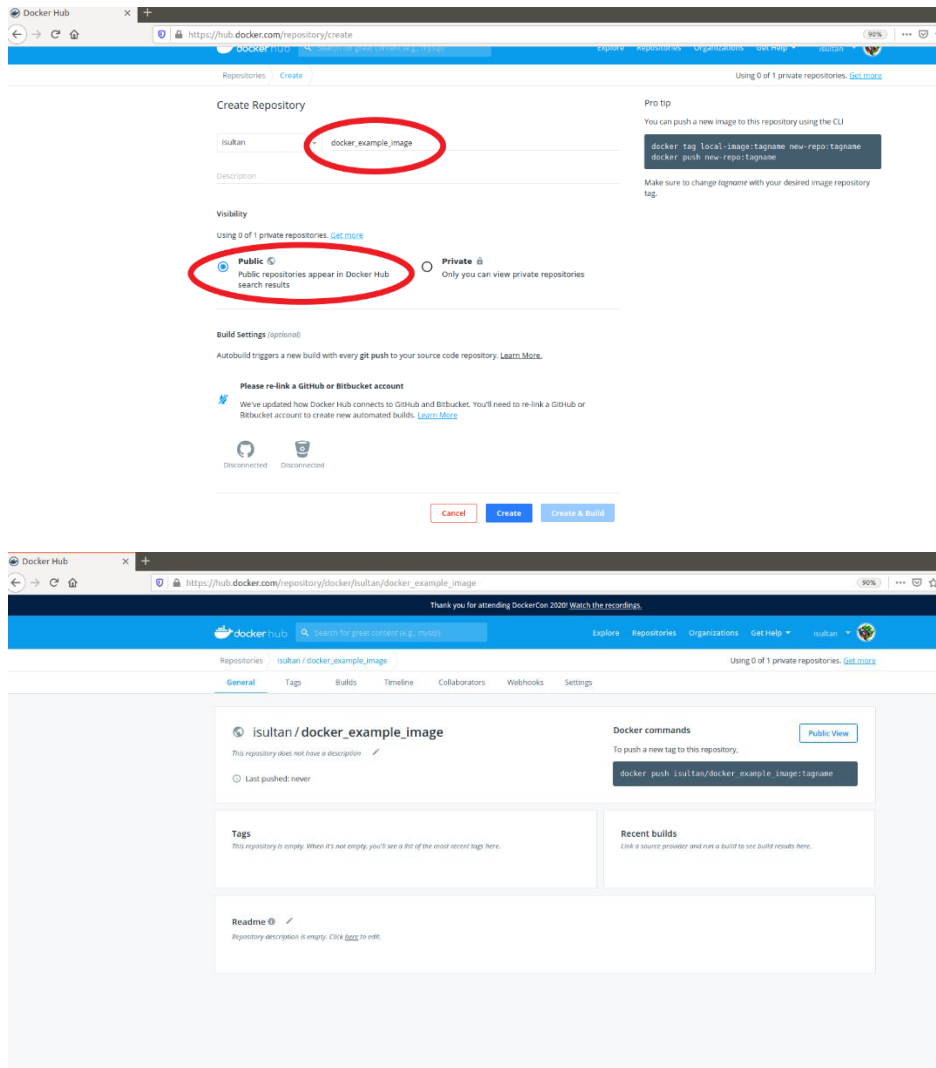
الخطوة الأولى: قبل ان نعمل push للصوره على منصة ال docker, نحتاج أولا ان يكون لدينا حساب على المنصة كما بيناه سابقا.

الخطوة الثانية: انشاء مخزن (repository) على منصة ال Docker, لكي نستطيع تحميل ال images كالتالي:

- 1- تسجيل الدخول على الموقع.
- 2- اختيار create repository كما هو مین في الصورة التالي:



3- قم بإكمال البيانات الخاصه بال repository المراد انشاؤه, اسم المخزن example-image وهو اسم ال image التي انشأناها مسبقا في ال Dockerfile. تحدد خصائص المخزن عام او خاص (في الغالب عام افضل). واخيرا نانشاء المخزن بالضغط على زر create كما هو موضح في الصورة التالية:

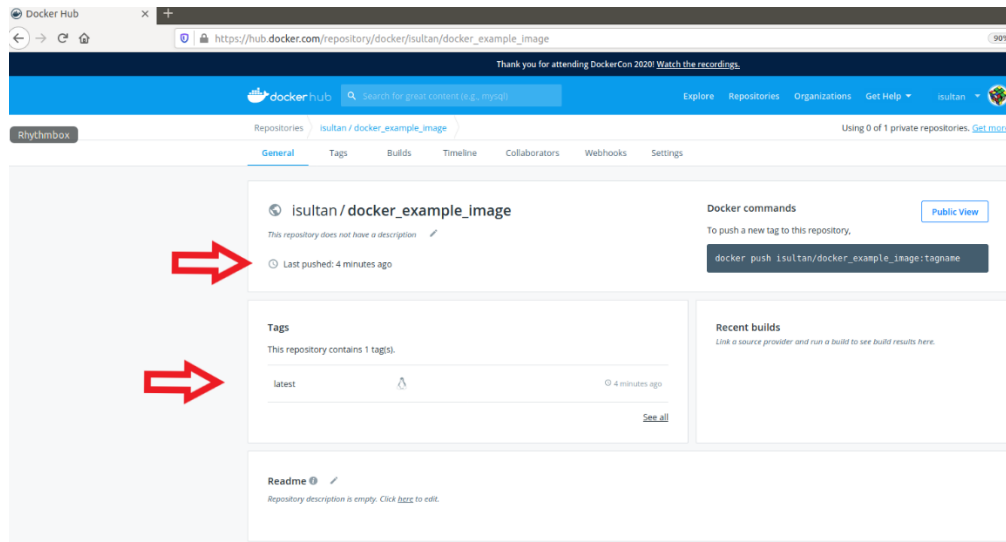


الخطوة الثالثة: عمل push للـ image للـ registry على المنصة كالتالي:

1. نقوم بتسجيل الدخول على حساب الـ docker الخاص بنا عبر نافذة الأوامر CLI باستخدام الأمر `$docker login`
2. نعمل tag للـ image. وهذه الخطوة مهمة جدا، قبل ان نقوم بعمل رفع للـ image على المخزن (repository). كما ذكرنا سابقا، بأن الدوكر يتبع سياسة تسميه لتحديد الـ image الرسمي من غير الرسمية (official or not). وما نقوم ببناءه هو non official image. وبالتالي لابد ان يتم الالتزام بالتسمية الصحيحة على النحو التالي: `<username>/<image_name>:<tag_name>`. في كثير من الحالات نحتاج إلى إعادة التسمية كما `myusername/example_image:latest`.

```
File Edit View Search Terminal Help
sultan@ubuntu1804:~/dockerproject$ sudo docker tag docker_example_image:latest isultan/docker_example_image:latest
sultan@ubuntu1804:~/dockerproject$ sudo docker push isultan/docker_example_image:latest
The push refers to repository [docker.io/isultan/docker_example_image]
1be74353c3d0: Mounted from library/busybox
latest: digest: sha256:2bdb54abb3e30519084ad434d5f2a6ccf0a483fe0953beba1f3e5b74c8dd4ba4 size: 527
sultan@ubuntu1804:~/dockerproject$
```

3. نشر الـ image بعد عمل التاق لها إلى المخزن باستخدام الأمر `$docker push`



بمجرد الانتهاء من الأمر, سنلاحظ بأن الـ image قد تم رفعها على منصة الـ docker بالوقت والتاريخ. هذا كل ما في الأمر, لقد تم بنجاح نشر أول صورة للتطبيق على الـ docker hub. لو اردت ان تقوم باختبار الـ image, نستخدم الأوامر التالية وتفعيل التشغيل للحاوية (container) كالتالي:

```
File Edit View Search Terminal Help
sultan@ubuntu1804:~/dockerproject$ sudo docker pull isultan/docker_example_image:latest
latest: Pulling from isultan/docker_example_image
Digest: sha256:2bdb54abb3e30519084ad434d5f2a6ccf0a483fe0953beba1f3e5b74c8dd4ba4
Status: Image is up to date for isultan/docker_example_image:latest
docker.io/isultan/docker_example_image:latest
sultan@ubuntu1804:~/dockerproject$ sudo docker run -it isultan/docker_example_image:latest

Mon Jun 29 15:04:11 UTC 2020
sultan@ubuntu1804:~/dockerproject$
```

الملخص

في نهاية هذا الشرح, قدمنا ملخص مبسط عن اساسيات نظام الـ Docker وأنه نظام يساعد مطوري البرمجيات على اختبار تطبيقاتهم ومشاركتها بشكل فعال مع الآخرين. وايضا تعرفنا على التعريفات الأساسية لمفاهيم الـ containerization و الفرقها عن الـ virtualization. كما اطلعنا على ما هو الـ container ومما يتكون وكيفية عمله. ايضا تعلمنا كيفية انشاء الـ images وعمل الـ containers على منصة الـ Docker hub والأوامر الأساسية التي يستخدمها المبرمج او المطور للتعامل مع نظام الـ Docker ومنصة الـ Docker hub. هناك تفاصيل كثيرة قد لاتهم المستخدم العادي لنظام الـ docker ومن أراد الاطلاع عليها على الرابط التالي: <https://docs.docker.com/>