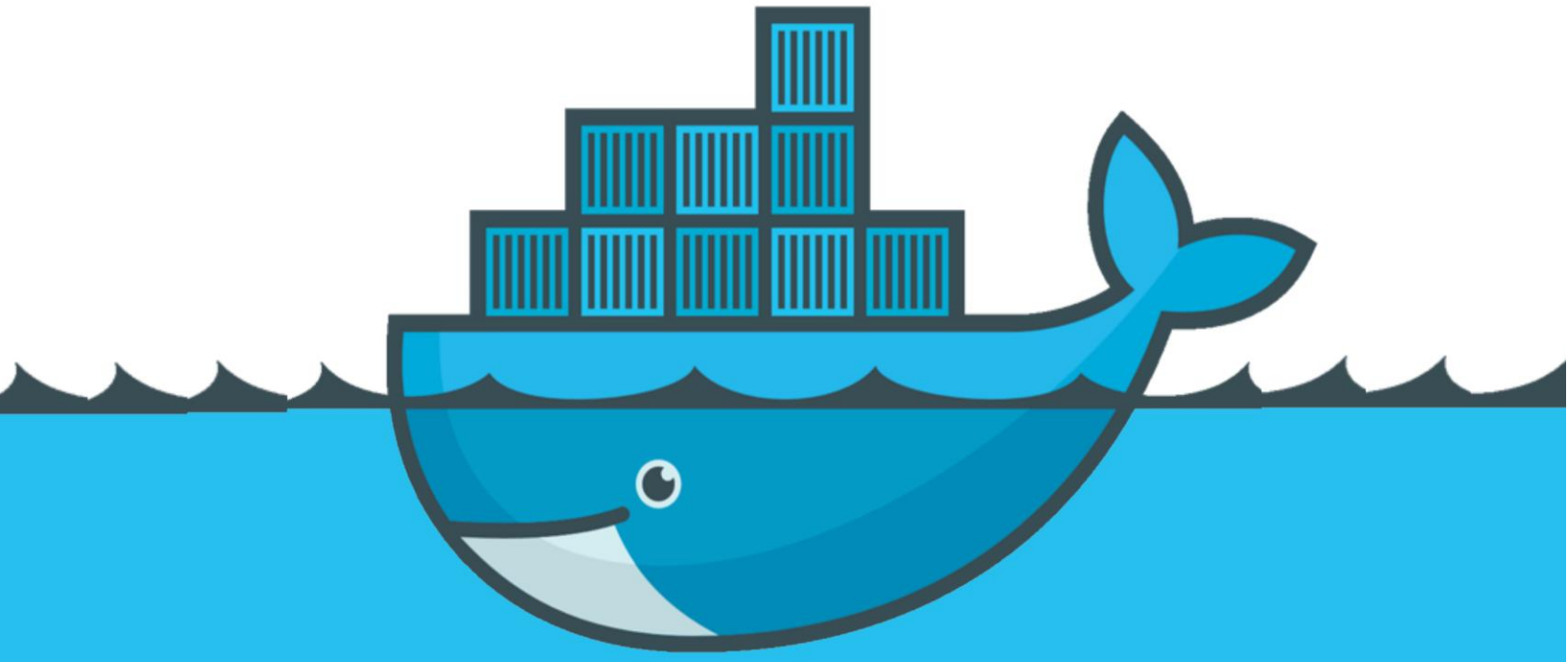
# خطواتك الأولى في

# Docker



وجدي عصام عبدالرحيم

# الفهرس


مقدمة ..... 4

الفصل الأول: مقدمة حول دوكر **Docker** ..... 5

لماذا نحتاج لطريقة جديدة في التعامل مع البرمجيات؟..... 5

ما الفرق بين ال Containers وال Virtual Machines؟ ..... 15

ما هو دوكر Docker؟..... 16

الفصل الثاني: البداية مع دوكر **Docker** ..... 18

تحميل دوكر على نظام تشغيل ويندوز ..... 18

تنصيب دوكر على لينوكس ..... 23

تجربة دوكر بدون تثبيت أي برامج ..... 25

البداية في أوامر Docker..... 26

مثال ثاني..... 33

تجربة ال Windows Containers..... 35

أنواع العزل Isolation في دوكر ..... 38

مكونات ال Image ..... 41

عزل العمليات Process Isolation ..... 43

عزل نظام الملفات File System Isolation..... 44

عزل الشبكات Network Isolation ..... 44

خلاصة..... 46

الفصل الثالث: مشاركة المجلدات مع ال **Containers** ..... 47

استخدام ال Volumes لمشاركة جزء من نظام الملفات ..... 49

تشغيل برنامج ffmpeg للتعامل مع ملفات الفيديو ..... 53

خلاصة..... 54

الفصل الرابع: بناء ال **Docker Images** لاستضافة المواقع ..... 55

تشغيل الموقع عن طريق ال Mounting..... 55

تشغيل الموقع عن طريق نسخ الملفات لل Container ..... 56

تشغيل الموقع عن طريق وضع كامل الملفات على شكل Image..... 57

استخدام ال Dockerfile بدلاً من إعادة كتابة الأوامر ..... 59
رفع ال Image على ال Docker hub ..... 61
الفصل الخامس: تشغيل قواعد البيانات في دوكر ..... 63
تشغيل MS-SQL Server بداخل ال Container ..... 63
تشغيل MySQL بداخل ال Container ..... 65
حفظ البيانات في ال Volumes ..... 66
الحذف – إيقاف ال Containers ..... 68
الحذف – حذف ال Containers ..... 69
الحذف – حذف ال Volumes ..... 70
الحذف – حذف ال Images ..... 71
خلاصة ..... 72
الفصل السادس: تشغيل البرامج بال **Docker Compose** ..... 73
أهمية ال Docker Compose ..... 73
استخدام ال Docker Compose ..... 75
خلاصة ..... 81
خاتمة ..... 82

# مقدمة

تقنية الدوكر Docker تعتبر من التقنيات الحديثة نسبياً إلا أنها أحدثت ضجة كبيرة عند ظهورها، حيث عززت فكرة ال Containerization والتي تسمح لك بتشغيل البرامج (حالياً ال Server-Side Applications) بداخل Container تكون معزولة عن البرامج أو ال Containers الأخرى في نظام التشغيل، بالإضافة الى تقديم البرامج على شكل Image بدلاً من أن تكون بأشكال مختلفة، مثلاً بداخل ملف مضغوط أو Jar أو exe وغيرها. وهذه المفاهيم سهلت لنا طريقة تطوير البرمجيات، طريقة نشرها على السيرفرات المختلفة وحتى ال Clouds، طريقة تشغيلها بسهولة في بيئات مختلفة.

ولأن تقنية الدوكر كبيرة وبها العديد من الجوانب فقد يكون من الصعب لأي مبتدئ أن يبدأ بها، فهناك مصادر تتحدث عنها ولكن باستخدام لغة أو تقنية معينة مثلاً ASP.NET & Docker أو Python & Docker وقد يكون من الصعب لمبتدئ لا يتقن تلك اللغة أن يستخدم هذه المصادر. وأحياناً تكون المصادر أيضاً موجهة بلهجة مدراء الأنظمة ويكون من الصعب مثلاً للمطورين البدء معها، أو تكون مفصلة للغاية بحيث تعرض كل التفاصيل والأوامر الدقيقة وقد لا تناسب من يريد البدء أو تجربة أول أمثلة بها.

ومن هنا كانت فكرة الكتاب، حيث هو يمثل الخطوة الأولى لك في عالم الدوكر Docker، وسوف نستعرض فيه بعض المفاهيم الأساسية فيه، والفوائد منه، بحيث تكون مناسبة سواءً للمطورين بغض النظر عن لغة البرمجة وأيضاً لمدراء الأنظمة أو لأي شخص يعمل في مجال تقنية المعلومات.

بالطبع لن تجد كل شيء هنا، لا من ناحية شرح الأوامر، ولا من ناحية تغطية كل جوانب الاستخدام، فهذا الكتاب ليس الا الخطوة الأولى فقط، وسوف يتبعه كتب مصغرة أخرى في عدة مواضيع مختلفة، مثلاً:

1. استخدام ال Docker مع تقنيات الويب المختلفة مثلاً ASP.NET أو NodeJS،
2. استخدام ال Docker Swarm Mode لعمل Clustering مكون من عدة سيرفرات وعشرات ال Containers
3. متابعة ال Healthy وال Logs لل Containers في دوكر باستخدام منصات لذلك مثلاً ELK وغيرها
4. عمل ال Continues Integration وال Continuous Delivery باستخدام الدوكر.
5. سوف نختم السلسلة ببناء مشروع متكامل يضم كافة الجوانب لبناء خدمات عالية الجودة والأداء High Availability and Scalability بهذه التقنية، ونطبق مفاهيم ال Microservices فيه وطرق التواصل الداخلي المختلفة مثلاً HTTP أو ببروتوكولات أخرى كال gRPC .

اعتمدت في إعداد هذا الكتاب على دورة Getting Started with Docker on Windows ل Wes Higbee وهي دورة رائعة في موقع Pluralsight الشهير بالإضافة الى خبرتي الشخصية في العمل على دوكر في ال Productions لعدة تطبيقات وخدمات ويب سواءً لشركات ناشئة أو لجهات ومؤسسات أخرى.

وجدي عصام عبدالرحيم
Email: wajdyessam@hotmail.com
Twitter: @wajdyessam
Website: https://informatic-ar.com
Questions: https://io.hsoub.com/docker
2018-11-24

# الفصل الأول: مقدمة حول دوكر Docker

يعتبر مشروع دوكر Docker من المشاريع الإبداعية والتي تغير طريقة التعامل مع البرمجيات سواءً من ناحية تنزيل هذه البرمجيات Installation، وطريقة نشرها وتشغيلها Deployment وأيضاً أثناء التطوير والاختبارات Building للمطورين.

ولأن دوكر Docker يعتبر أحد المشاريع التي تستخدم مفهوم ال Containers وال Images، وهناك غيره من المشاريع التي تستخدم نفس المفاهيم مثل Rkt، فمن المهم أن تكون هذه المقدمة عن ماهية هذه المفاهيم (Containers & Images) وما هي أهميتها، وما الدور الذي جعلت المشاريع التي تستخدمها مثل Docker or Rkt من أنجح المشاريع حالياً. وسوف نناقش في هذا الفصل ما يلي:

- لماذا نحتاج الى طريقة جديدة في التعامل مع البرمجيات
- الفرق بين ال Container وال Virtual Machine
- ما هو ال Container وال Images
- ما هو Docker

## لماذا نحتاج لطريقة جديدة في التعامل مع البرمجيات؟

لكي نفهم ال Containers والحوجة لها وما الذي اضافته لنا، فسوف نقوم بعمل مقارنة بين الأسلوب القديم عندما نعمل على البرمجيات من أكتر من منظور مع ال Containers، وذلك من النواحي التالية:

- عند تحميل البرامج – كمستخدم أو شخص يريد تحميل أدوات لجهازه
- أثناء البرمجة والتطوير ونشر البرمجيات على السيرفر –وكيف يتم توحيد بيئة التطوير والاختبارات وتسهيل عملية نشر البرمجيات

وسنبدأ بأول نقطة وهي في العمل على البرمجيات: ولنفرض أنك أردت تحميل أي برنامج لكي تستخدمه على جهازك، وليكن مثلاً MySQL أو RabbitMQ أو أي برنامج اخر، فسوف تجد أن هناك خطوات عليك القيام بها:

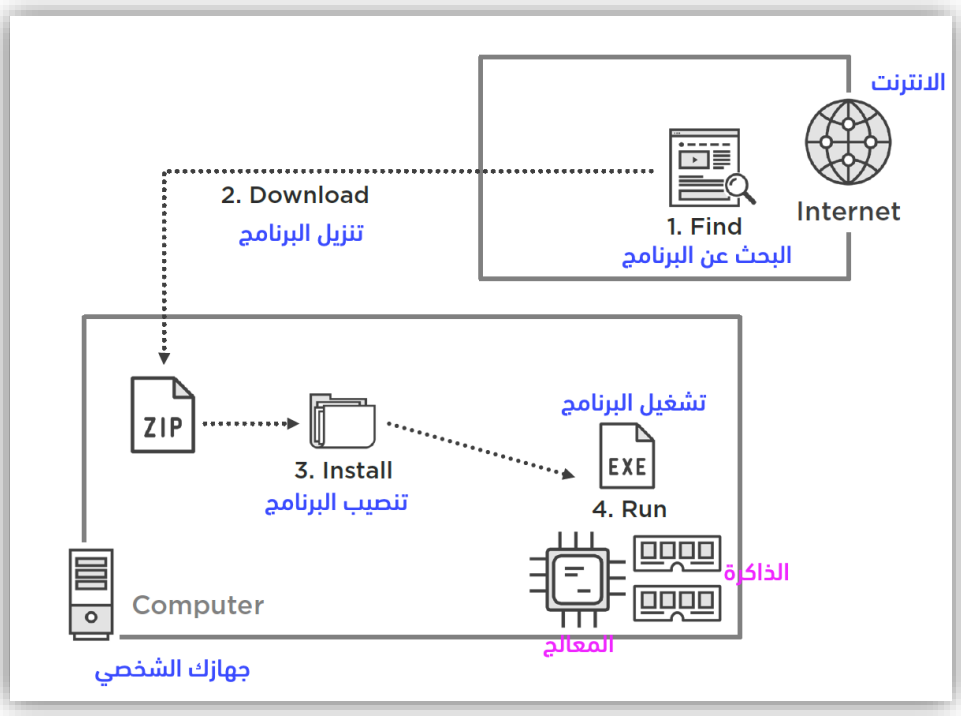
1- البحث **Finding**: عن مكان لتحميل البرنامج، سواءً من موقعه الأصلي، أو من مواقع أخرى

2- التنزيل **Downloading**: فتح صفحة التحميل وقراءتها بتركيز لتحميل البرنامج المطلوب، على حسب نظام التشغيل، وهل هو 32bit/64bit والتأكد من تحميل النسخة الصحيحة

3- التنصيب **Installing**: قرائه صفحة الاعتماديات وهل هناك متطلبات لتحميل برامج وأدوات أخرى مطلوبة، ووضع متغيرات في النظام Environment Variables يحتاجها البرنامج في عمله

4- التشغيل **Running**: تشغيل البرنامج بشكل صحيح والقدرة على إيقافه وتشغيله متى ما أردت.

الصورة التالية تبين كل هذه الخطوات سوف تقوم بها حتى تقوم بتحميل البرنامج ومن ثم تشغيله لأول مرة:




وهذه الخطوات ليست بالسهلة وقد تخطئ في أحد الخطوات ولن تستطيع اكمال تنصيب أو تشغيل البرنامج بشكل صحيح، وقد تلغى فكرة استخدام هذا البرنامج من الأساس وتبدأ في البحث عن بديل بسبب هذه المشكلة، ففي كل خطوة قد تجد العديد من المشاكل، على سبيل المثال:

- مرحلة البحث **Searching**:
  - مكان التحميل **Where**: وهل سيتم تحميله ال App Store مثلاً إذا كان يتواجد متجر للتطبيقات في نظام التشغيل، أم قد يكون نظام التشغيل به Package Manager كما في لينوكس وتستطيع تحميل البرنامج مباشرة، ولكن أحياناً تكون بعض البرامج قديمة في ال Package Manager ولا يوجد بها التحديث الأخير، وقد تجد برنامج معين في ال Package Manager في توزيعه لينوكس ما ولن تجده على توزيعه أخرى مثلاً. في بعض الحالات خصوصاً في نظام تشغيل ويندوز قد تجد في الغالب أنك مضطراً لتحميل البرنامج من مواقع ما Standalone website سواءً كانت من الموقع الأصلي أم مواقع أخرى توفر البرنامج.
  - معلومات عن البرنامج **Stats** & **Metadata**: قبل تحميلك للبرنامج قد تحتاج بعض المعلومات لأخذ القرار هل تريد تنزيله أم لا، وقد لا يتوفر في مكان التحميل أي معلومات إحصائية عن عدد التحميلات وأي معلومات أخرى عن البرنامج، مثلاً عدد المشاكل للمستخدمين وغيرها، تاريخ أخر تحديث للبرنامج. وقد تجد بعض المواقع توفر هذه المعلومات والبعض الآخر لا يوفرها أو يوفرها بطريقة مختلفة، تختلف بحسب الموقع.
  - الثقة **Trust**: قد تبحث عن البرنامج من خلال محرك البحث، ومن ثم تصل الى موقع ما يوفر خاصية التحميل، لكن هل هو موقع موثوق، أو موقع مزيف به برمجيات خبيثة ملغمة؟ وهل الموقع يوفر حماية عند تنزيل المشروع باستخدام HTTPS؟ بالإضافة الى هل الموقع يعمل دائماً Availability أم أحياناً به مشاكل ولا تستطيع تحميل البرنامج، وغيرها من المشاكل في هذه المرحلة.

- مرحلة التحميل **Installing**:
  - نظام التشغيل **Operation System**: هل سيعمل على نظام التشغيل الحالي، هل هو Cross-Platform وسوف يعمل على هذه النسخة المعينة بالضبط version/build من نظام التشغيل، هل سيعمل على هذه ال CPU التي تعمل بها والمعمارية التي يعمل بها Architecture.
  - ماهية البرنامج **Format**: هل سيأتي السورس كود فقط وتقوم ببنائه وتشغيله، أو يأتي على شكل executable يعمل مباشرة، أم يأتي به الملف التنفيذي مع كامل المكتبات المطلوبة به في ملف مضغوط Zip؟ أو على شكل Setup به كامل الملفات، وهل سيأتي معه ال Runtime المطلوبة مثل ال JRE أو NET. أم تحتاج لتنصيبها على حدة؟
  - طريقة التحديث **Updates**: كيف ستكون وبأي ألية وهل سيتم مسح القديم كلياً وإعادة تنصيبه، وهل من خلال برنامج اخر أم ما هي التفاصيل لذلك.

- مرحلة التشغيل **Running**:
  - التوثيق للبرنامج **Documentation**: وهل هناك توثيق محدث بحيث تتبع الخطوات فيه أم تحتاج للبحث في الأنترنت عن كيفية تشغيله.
  - المسارات **Path**: هل تحتاج لوضع متغيرات في النظام Environment Variables يحتاجها البرنامج في عمله أم سوف يقوم بها أثناء التنزيل بشكل افتراضي.
  - الاعتماديات **Dependencies**: هل سيقوم البرنامج بتحميلها كاملة أو يحب أن تقوم بتحميلها قبل تحميل البرنامج، وهل تحتاج هذه الاعتماديات الى اعتماديات اخرى
  - كيفية التشغيل والايقاف **Service Registration**: هل من خلال الواجهة للبرنامج تستطيع اغلاق البرنامج، هل تحتاج لوضعه ك Service لكي يعمل مباشرة بعد إعادة التشغيل أو تريد إيقافه وتشغيله مجددا
  - هل هناك تعارض مع البرامج الأخرى: إذا كان البرنامج سوف يفتح بورت معين Port أو يحتاج وصول لملفات معينة فيجب التأكد من عدم استخدامها بواسطة برمجيات أخرى
  - هل التحديثات القادمة سوف تؤثر على البرنامج: مثل تحديثات نظام التشغيل في ال Shared Library التي يستخدمها البرنامج ويعتمد عليها.

لذلك قد تجد أنك تحتاج لساعات عديدة فقط لتنصيب برنامج قبل البدء باستخدامه، خصوصاً البرمجيات الكبيرة، مثلاً وجدت قاعدة بيانات Casandra أو مثلاً Elastic Search وغيرها من البرمجيات التي قد تود أن تتعلمها، فقد تمضي ساعات عديدة وتلغى فكرة تعلمها بسبب واحد هو تعقيد مسألة التعامل مع البرمجيات بشكل عام، بدءاً من البحث، التنزيل، والتشغيل.

فهل توجد آلية ما بحيث نجعل هذه العملية في منتهى البساطة، وتكون موحدة وتعمل في أي مكان، وكل ما في الأمر هو أن تقوم بتحميل البرنامج من مكان موثوق بسهولة، وبأمر واحد يعمل بكافة الاعتمادية المطلوبة ويكون البرنامج معزول عن بقية البرامج الأخرى، ويتم تشغيله بسرعة وايقافه بسرعة وتشغيل أكثر من نسخة بسهولة أيضاً؟ مرحباً بك في ال **Containers & Images** (ودوكر).

ملاحظة: كثيراً من المواضيع عندما تتحدث عن ال Docker & Containers فمباشرة تبدأ بالحديث عن الفرق بين ال Containers وال Virtual Machines وبالرغم من أن هناك فروق وسوف نتحدث عنها، لكن البداية بهذا الشكل قد يعطي الانطباع بأن ال Docker وال Containers هي بديل لل Virtual Machines بحيث تكون Virtualization من نوع آخر، وبالرغم من أن هذا صحيح وأنه يمكن أن

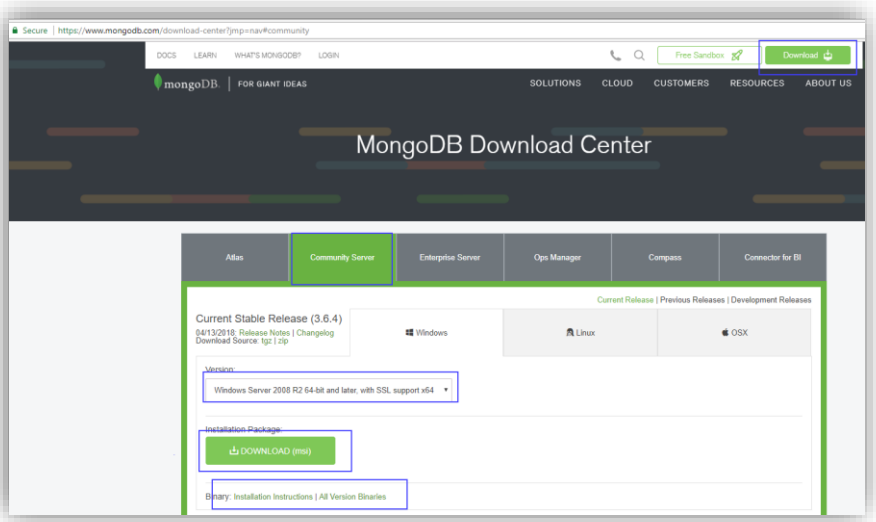
يكون بديل لل Virtualization الا أن ال Containers بشكل عام ليست عن ال Virtualization وانما هي لتسهيل التعامل مع البرمجيات (مثلاً يمكنك اختصارها بأنها لل Software Delivery أو لل Running Software) بدءاً من البحث، التحميل، التنزيل، التشغيل.

بهذه المنظور سوف يسهل عليك العمل على ال Containers وال Docker وسوف تفهمها بالطريقة الصحيحة، ولن تتفاجأ فيما بعد مثلاً بأن هناك بعض البرمجيات أو ال Containers سوف تعمل على ال VM بسبب معين سوف نتحدث عنه فيما بعد (مثلاً تشغيل ال Linux Containers بداخل الويندوز، فهذا يتطلب تشغيل Linux VM ومن ثم تشغيل ال Container بها، ولكن هذا سيتم في ثواني معدودة وفي الخفاء بحيث لا تدري ماذا يحصل داخلياً) لكن الفكرة هو النظر لدوكر على أنه Build, Ship, Delivery وليس بديل لل Virtualization.

## مثال

سوف نقوم الآن بتنصيب برنامج بالطريقة السابقة التي كنا نتعامل معها طوال السنوات السابقة، ومن ثم نجرب الطريقة الحديثة ولنرى كيف أصبحت العملية في منتهى البساطة بها، ويمكنك أن تكمل القراءة بدون تجربة هذا المثال، لأن الفكرة هو اعطائك نظرة عامة وفهم لما يحدث، وسوف نتناول في الفصل الثاني طريقة تنصيب الدوكر خطوة بخطوة، ولكن لنركز على ال Big Picture الآن.

سوف نقوم بتنزيل MongoDB بالطريقة اليدوية، وسوف نبدأ بالبحث في Google عن MonogoDB وسوف تجد خرجت الصفحة الرئيسية، وبعدها سوف ننتقل لصفحة التنزيل، وهناك سوف تجد عدة خيارات سوف يتم اختيار ال Community Server، ومن ثم اختيار نسخة نظام التشغيل الذي اعمل عليه، وأيضاً يتم تحديد هل تريد تحميله مضغوط Zip أو على شكل Installer بالامتداد msi، وسوف تجد رابط لدليل وخطوات التحميل.




بعدها سيتم التحميل بامتداد msi، وأثناء ذلك يمكن فتح دليل التنصيب في الرابط وسوف نجد أن ال Installer يقوم بالعديد من الخطوات، لكن تحتاج الى وضع البرنامج في متغيرات النظام أن أردت تشغيله مباشرة من أي مسار.

بعد أن يتم التحميل يمكن الضغط على التالي Next-Next ليتم تنصيب البرنامج بالإعدادات الافتراضية وبعدها لن تجد شيئاً، وسوف تحتاج أن ترجع للدليل لكي تعرف كيف سيتم تشغيل ال MongoDB server:

**1** **Set up the MongoDB environment.**

MongoDB requires a data directory to store all data. MongoDB's default data directory path is the absolute path **\data\db** on the drive from which you start MongoDB. Create this folder by running the following command in a **Command Prompt**:

```
md \data\db
```

You can specify an alternate path for data files using the **--dbpath** option to **exe**, for example:

```
"C:\Program Files\MongoDB\Server\3.6\bin\mongod.exe" --dbpath d:\test\mongodb\data
```

If your path includes spaces, enclose the entire path in double quotes, for example:

```
"C:\Program Files\MongoDB\Server\3.6\bin\mongod.exe" --dbpath "d:\test\mongo db data"
```

You may also specify the **dbpath** in a configuration file.

**2** **Start MongoDB.**

To start MongoDB, run **exe**. For example, from the **Command Prompt**:

```
"C:\Program Files\MongoDB\Server\3.6\bin\mongod.exe"
```

سوف تجد في الصورة مكان تواجد البيانات الافتراضي هو /data/db وهذا المسار عموماً بطريقة اللينوكس بينما في الويندوز نحتاج لتحديد البارتيشن الذي يتواجد فيه أولاً، وسوف نقوم بفتح مسار ال MongoDB من سطر الأوامر وتنفيذ أمر التشغيل:

```
C:\WINDOWS\system32\cmd.exe

C:\Program Files\MongoDB\Server\3.6\bin>mongod.exe
2018-05-18T08:25:35.852-0700 I CONTROL  [initandlisten] MongoDB starting : pid=9356 port=27017 dbpath=C:\data\db\ 64-bit host=WAJDY-DESKTOP
2018-05-18T08:25:35.853-0700 I CONTROL  [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2018-05-18T08:25:35.854-0700 I CONTROL  [initandlisten] db version v3.6.4
2018-05-18T08:25:35.854-0700 I CONTROL  [initandlisten] git version: d0181a711f7e7f39e60b5aeb1dc7097bf6ae5856
2018-05-18T08:25:35.854-0700 I CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.0.2o-fips  27 Mar 2018
2018-05-18T08:25:35.854-0700 I CONTROL  [initandlisten] allocator: tcmalloc
2018-05-18T08:25:35.854-0700 I CONTROL  [initandlisten] modules: none
2018-05-18T08:25:35.854-0700 I CONTROL  [initandlisten] build environment:
2018-05-18T08:25:35.854-0700 I CONTROL  [initandlisten]     distmod: 2008plus-ssl
2018-05-18T08:25:35.854-0700 I CONTROL  [initandlisten]     distarch: x86_64
2018-05-18T08:25:35.854-0700 I CONTROL  [initandlisten]     target_arch: x86_64
2018-05-18T08:25:35.854-0700 I CONTROL  [initandlisten] options: {}
2018-05-18T08:25:35.855-0700 I STORAGE  [initandlisten] exception in initAndListen: NonExistentPath: Data directory C:\data\db\ not found., terminating
2018-05-18T08:25:35.855-0700 I CONTROL  [initandlisten] now exiting
2018-05-18T08:25:35.855-0700 I CONTROL  [initandlisten] shutting down with code:100

C:\Program Files\MongoDB\Server\3.6\bin>
```

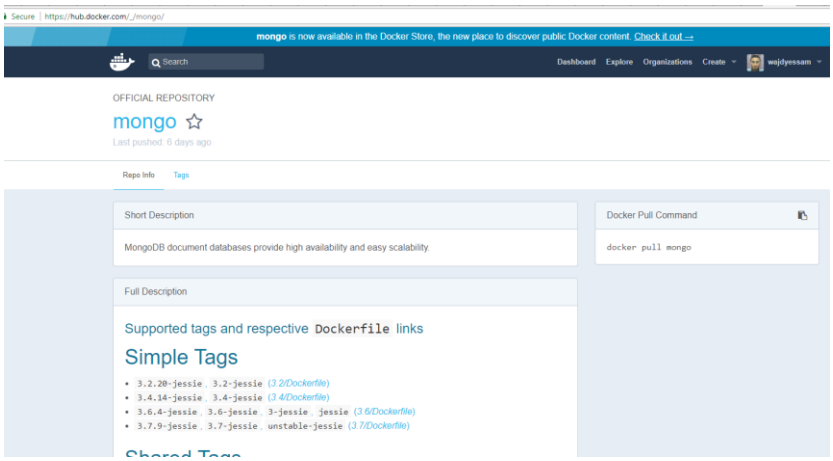
سوف تجد خطأ في المسار، وعليك أن تقوم بإنشائه وسوف ننشئ المجلد على القرص C، وبعدها نقوم بتشغيله مجددا وسوف يعمل الآن، وسوف تجد رسالة بأن السيرفر ينتظر الاتصال على البورت 27017:

```
C:\WINDOWS\system32\cmd.exe - mongod.exe
2018-05-18T08:27:28.463-0700 I CONTROL  [initandlisten] ** WARNING: This server is bound to localhost.
2018-05-18T08:27:28.463-0700 I CONTROL  [initandlisten] **          Remote systems will be unable to connect to this server.
2018-05-18T08:27:28.463-0700 I CONTROL  [initandlisten] **          Start the server with --bind_ip <address> to specify which IP
2018-05-18T08:27:28.463-0700 I CONTROL  [initandlisten] **          addresses it should serve responses from, or with --bind_ip_all to
2018-05-18T08:27:28.463-0700 I CONTROL  [initandlisten] **          bind to all interfaces. If this behavior is desired, start the
2018-05-18T08:27:28.463-0700 I CONTROL  [initandlisten] **          server with --bind_ip 127.0.0.1 to disable this warning.
2018-05-18T08:27:28.463-0700 I CONTROL  [initandlisten]
2018-05-18T08:27:28.464-0700 I CONTROL  [initandlisten]
2018-05-18T08:27:28.464-0700 I CONTROL  [initandlisten] ** WARNING: The file system cache of this machine is configured to be greater than 40% of the tota
l memory. This can lead to increased memory pressure and poor performance.
2018-05-18T08:27:28.464-0700 I CONTROL  [initandlisten] See http://dochub.mongodb.org/core/wt-windows-system-file-cache
2018-05-18T08:27:28.464-0700 I CONTROL  [initandlisten]
2018-05-18T18:27:28.465+0300 I STORAGE  [initandlisten] createCollection: admin.system.version with provided UUID: d6e5b328-9989-4012-b9c0-5a33b22f8332
2018-05-18T18:27:28.476+0300 I COMMAND  [initandlisten] setting featureCompatibilityVersion to 3.6
2018-05-18T18:27:28.479+0300 I STORAGE  [initandlisten] createCollection: local.startup_log with generated UUID: 09994848-de11-4100-949d-ac7239910186
2018-05-18T18:27:28.750+0300 I FTDC     [initandlisten] Initializing full-time diagnostic data capture with directory 'C:/data/db/diagnostic.data'
2018-05-18T18:27:28.752+0300 I NETWORK  [initandlisten] waiting for connections on port 27017
```

سوف نفتح سطر أوامر اخر لكي نتصل بهذا السيرفر الذي ينتظر أي كلاينت يتصل به، وسوف نكتب الأمر mongo.exe لتشغيل الكلاينت، ونكتب أي أمر مثلاً لمشاهدة قواعد البيانات show dbs:

```
> show dbs
admin   0.000GB
local   0.000GB
>
```

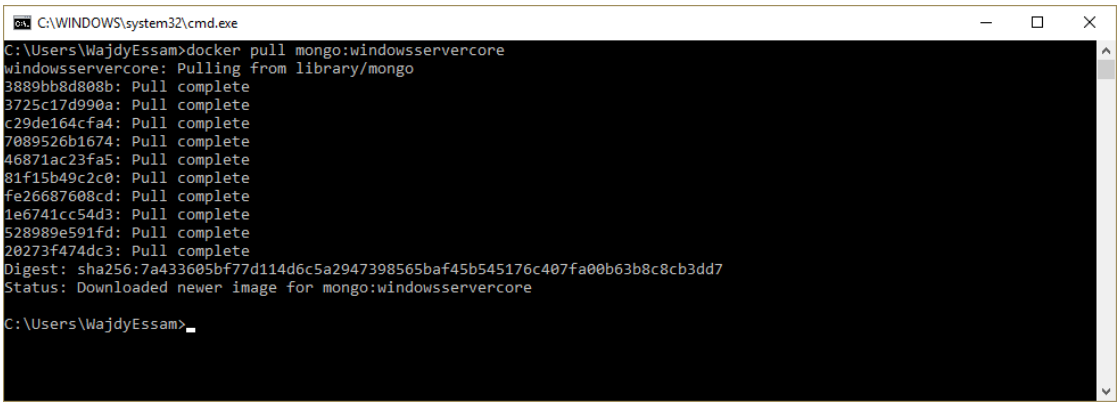
سوف نقوم الآن بنفس التجربة ولكن بمفهوم ال Containers/Images ولنرى الفرق الكبير بينهم، فسوف نبدأ بالبحث عن mongodb في محرك البحث وسوف يظهر أول نتيجة في ال docker hub:




لتحميل ال MongoDB من ال Docker Hub سوف نقوم بعمل ال Pull (وهو أمر التحميل) وسيتم تحميل ما يعرف بال Image (ويمكن تصورها على أنها البرنامج) وهي فعلياً تحتوي على البرنامج مع كل الاعتماديات الأخرى. وسوف تلاحظ وجود ال Official Repository في أعلى الصفحة للدلالة على أن هذا البرنامج (ال Repository) من مصدره الرسمي.

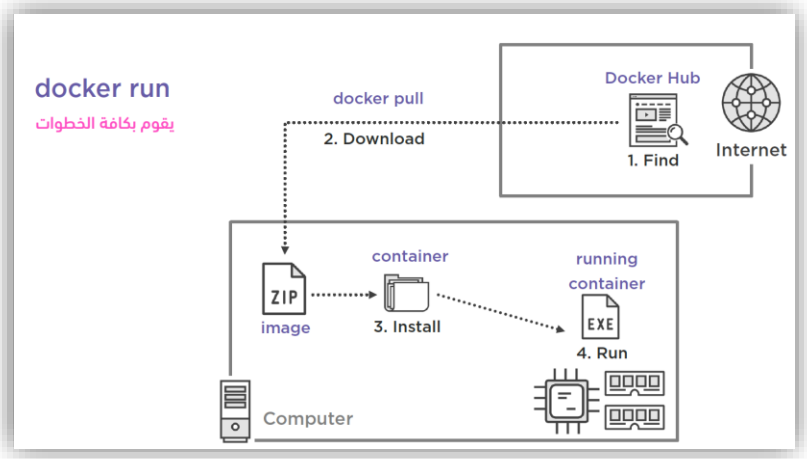
الصورة التالية تبين طريقة التحميل وتم تحديد اسم البرنامج، بالإضافة الى تحديد windowsservercore وذلك لأني أعمل على Windows وأريد تشغيل البرنامج) بداخل نظام التشغيل مباشرة وبالتالي احتاج الى أن استخدم نسخة الويندوز وليس اللينوكس، وبالرغم من أنه يمكن تشغيل نسخة اللينوكس على الويندوز بأي حالة ولكن هذا يستوجب اخبار دوكر واعداده ليشغل نمط ال Linux Container، سوف نتحدث عن هذه النقطة بالتفصيل فيما بعد، الذي يهم الآن هو تحميل ال Image بهذا الأمر:

docker pull mongo:windowsservercore



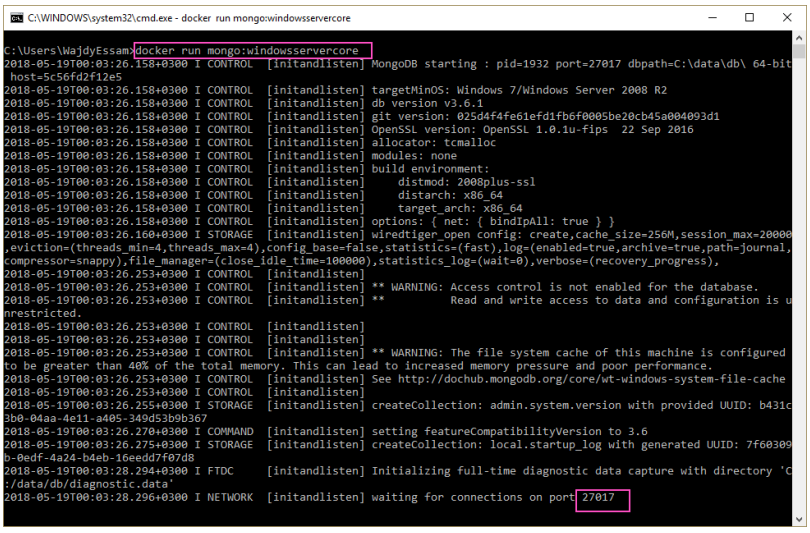
سوف يأخذ تحميل البرنامج بعضاً من الوقت نظراً لأن به كل الاعتماديات بالإضافة الى أن ال Windows Images عموماً أكبر حجماً (وكما ذكرنا يمكن أن نستخدم ال MongoDB Linux Image بداخل الويندوز بسهولة وسيتم توضيحها فيما بعد)، والآن بعد أن ينتهي تحميل ال Image نكون أنهينا الخطوة الثانية وهي التحميل، ولننتقل الى الخطوة الرابعة وهي تنصيب البرنامج، وفي عالم الدوكر يمكن أن نتخيل عملية التنصيب هي ال Container أو وضع ال Image بداخل ال Container، بمعنى أن ال Container هو البرنامج بعد تنزيله والذي يكون جاهزاً للتشغيل، والخطوة الأخيرة هي تشغيل ال Container، وسوف نرى كيف يمكن أن نشغل ال MongoDB Image الآن في ال Container.

الصورة التالية توضح الخطوات لتحميل البرامج باستخدام ال Containers/Images وسوف نجد أنها أصبحت بسيطة للغاية، والنقطة الأهم موحدة لجميع البرامج:




حتى هذه اللحظة فقد قسمنا هذه العملية الى خطوات لأخذ تصور عما يحدث، ولكن الأمر الخاص بالتشغيل سوف يقوم بكل هذه العمليات في أن واحد، بمعنى هو أمر واحد سوف يقوم بتحميل ال Image وتشغيلها في Container مباشرة (سوف يقوم بتحميل ال Image في حال لم تكن موجودة ويقوم داخلياً بعمل Docker Pull، وإذا كانت موجودة فيتم استخدامها) وهو كما يلي:

docker run:mongo:windowsservercore

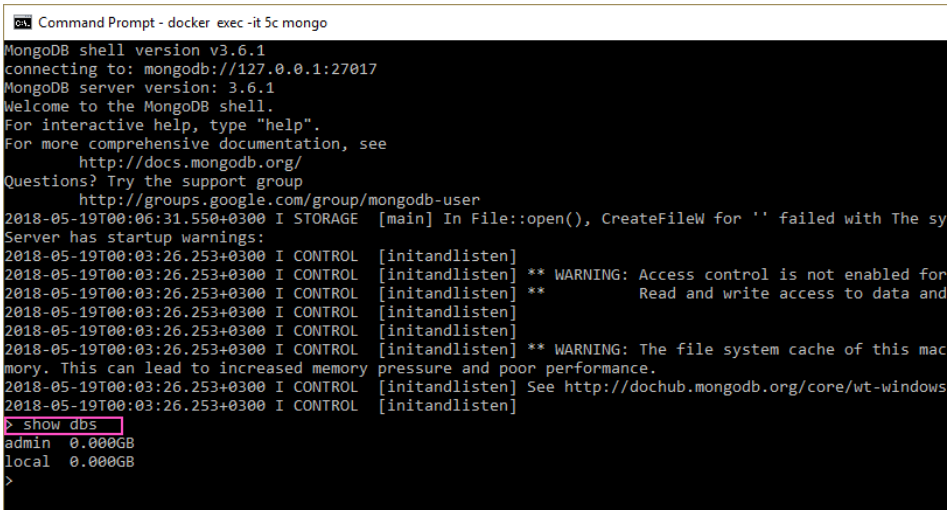


سوف تجد الآن نفس المخرج السابق بالضبط عندما قمنا بتشغيل وتنصيب البرنامج بالطريقة التقليدية ونفس البورت أيضاً.

الآن سوف نجد أن البرنامج يعمل بداخل ال Container (وهذا يسمي بال Attached to Container) ولذلك سوف نفتح سطر أوامر أخر، وسوف نقوم بالدخول على نفس ال Container التي تعمل من خلال هذا الأمر: docker exec –it 27 mongo وهي ستقوم بتشغيل ال mongo client بداخل نفس ال Container السابق، ومن ثم إعطاء الأمر show dbs وسوف تجد نفس المخرج أيضاً

docker exec -it 5c mongo

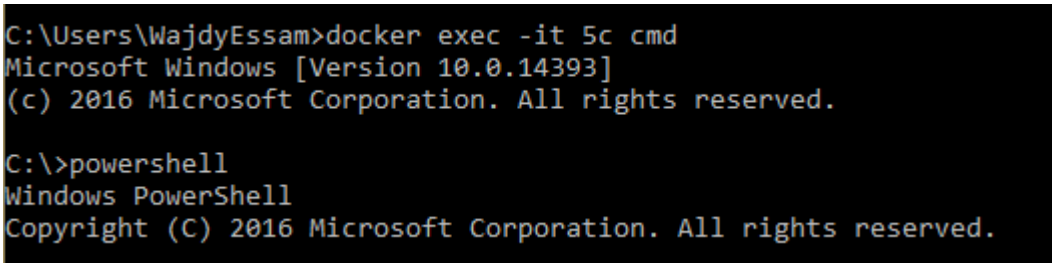
سوف نأتي فيما بعد لوصف الأوامر وشرحها بالتفصيل، ولكن ال 5c في الأمر هو رقم ال container الذي يعمل وتم استخراجه من خلال الأمر docker ps. وسوف ندخل الآن الى داخل ال container التي يعمل بها السيرفر، وقمنا بتشغيل برنامج الكلاينت mongo بها، وسوف نضع نفس الأمر لعرض قاعدة البيانات:




وسوف نجد نفس النتيجة السابقة عندما قمنا بالطريقة التقليدية، الفرق الآن أن الكلاينت والسيرفر يعملان بداخل ال container، وليس بداخل process عادية في نظام التشغيل.

في الحقيقة سوف نجد أن هذه ال container هي عبارة عن process أيضاً في نظام التشغيل، ولكنها معزولة isolated بحيث يكون لديها نظام الملفات الخاص بها، الشبكات الخاصة بها، قائمة العمليات process الخاصة بها، وكل شيء من داخل ال container سوف يكون كأنه يعمل مفصول تماماً، ولكن من الخارج فهي مجرد process أخرى.

ولكي نتأكد من هذا المفهوم سوف ندخل الى ال container التي تعمل الآن، بنفس الطريقة التي اشغلنا بها برنامج ال Mongo client ولكن هذه المرة لكي نشغل سطر الأوامر فيها، ونرى بعض الأمور، وسنقوم بتنفيذ الأمر: docker exec –it 5c cmd ، وبعدها قمنا بالدخول الى ال PowerShell كما يلي، وكان بالإمكان أن ندخل اليه مباشرة طالما هو برنامج على ذلك ال container:



الآن سنقوم بعرض كافة العمليات التي تعمل على ال container من خلال Get-Process كما يلي:

```
PS C:\> Get-Process

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)      Id  SI ProcessName
-------  ------    -----      -----     ------      --  -- -----------
    104       5     1104       4884       0.06    1512   1 CExecSvc
     42       3     1564       2836       0.02    1056   1 cmd
    186      10     1868       4392       0.41     504   1 csrss
      0       0        0          4                  0   0 Idle
    718      20     3912      11896       0.39     868   1 lsass
    140      13    56968      48052       0.14     640   1 mongo
    218      21    99920      75480       0.63    1932   1 mongod
    188      12     2280       9256       0.08     988   1 msdtc
    545      48    65876      81736       4.25    1952   1 powershell
    175       8     1948       5724       0.25     920   1 services
     48       2      344       1172       3.05      72   0 smss
    289      12     2520       9176       0.13    1084   1 svchost
    256      14     2132       6872       0.02    1120   1 svchost
    272      13     2480      10524       0.20    1204   1 svchost
    296      14     6212      11624       0.23    1224   1 svchost
    930      36    12304      31264       3.33    1284   1 svchost
    185      15     3104      10140       0.06    1304   1 svchost
    512      58     6104      18488       0.44    1384   1 svchost
    362      18     8456      18436       0.50    1476   1 svchost
     89       6     1092       5344       0.02    1488   1 svchost
   4675       0      128        136       2.83       4   0 System
     91       8      900       5152       0.19     712   1 wininit
    140       9     5864      12076       0.83     880   1 WmiPrvSE
```

وكما تلاحظ فأن ال Mongo server وال Mongo Client يعملان الآن بداخل ال container وحتى ال PowerShell الذي قمنا بتشغيله فهو يعمل أيضاً. بقية العمليات هي services خاصة بالويندوز. النقطة المهمة هنا هو أن هذه العمليات معزولة ولن ترى عمليات process من نظام التشغيل الأساسي، وانما فقط العمليات بداخل ذلك ال Container.

من النظام الأساسي ال Host Machine سوف تجد في قائمة العمليات كافة العمليات التي تعمل على النظام الأساسي، بالإضافة الى كافة العمليات التي تعمل أيضاً على ال Container سوف تجدها تعمل كأنها Process عادية، ولكنها كلها تشترك في ال Job ID وجميعهم يكون بنفس الرقم. وهذا يعني أن البرامج التي يتم تشغيلها في ال Container سوف تعمل على النظام الأساسي وكأنه تم تشغيلها مباشرة بفتحها مباشرة من ال host machine.

هكذا سوف نرى أن التعامل مع البرمجيات أصبح أسهل سواءً في التحميل، التنصيب، التشغيل، بالإضافة الى وجود كمية من البرمجيات توجد على شكل Images قابلة للتشغيل ك Containers سواءً في ال Docker Hub (المستودع الأساسي لل Docker Images) أو غيرها من المواقع التي توفر هذه الخدمة.

اذاً يمكن القول بأن ال Image هي القالب Read only Template المكونة من عدة ملفات بشكل Layered مثل ملفات نظام التشغيل، ملفات للفريم وورك مثلاً NodeJS وغيرها من البرمجيات.

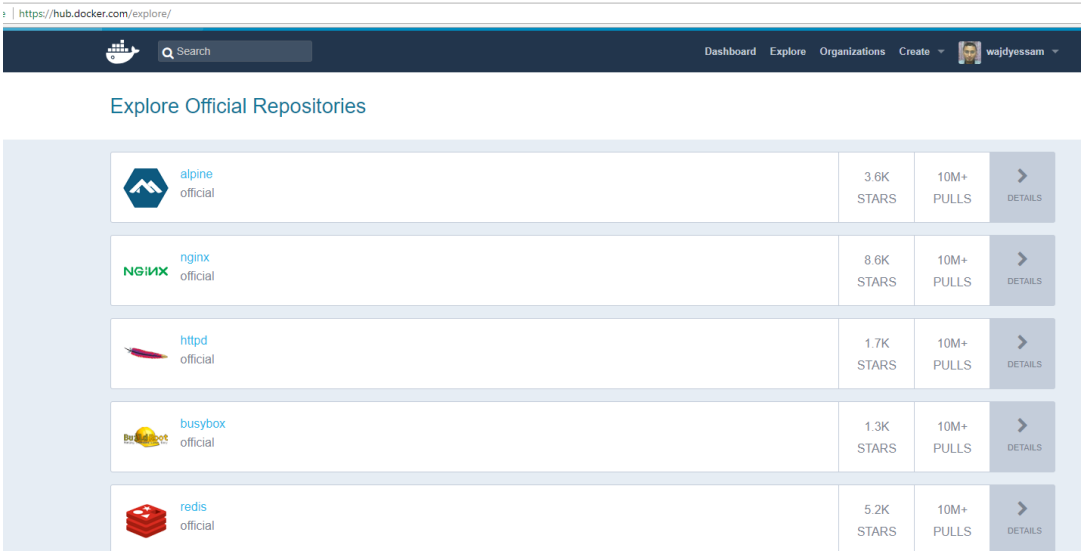
وال Container هو البرنامج المعزول الذي يتم إنشائه من ال Image ويمكن أن يكون قيد العمل، أو متوقف ويمكن أن يتم نقله أيضاً أو حذفه. وهذه العملية سريعة للغاية فيمكن إنشاء وتشغيل وإيقاف ال Containers بسهولة.

ملاحظة: يعتبر ال Docker Hub من أحد المواقع لتسجيل ال Docker Images بسهولة بحيث يمكن تحميلها فيما بعد، بمعنى مستودع للبرامج Repositories (مثل متجر تطبيقات Google Play store مثلاً) التي تحتاجها وهو المستودع الرئيسي في مشروع الدوكر. وبه

العديد من البرمجيات مثل MySQL, WordPress, Ubuntu وغيرها، وتستطيع تحميلها pull بسهولة سواءً من جهازك الشخصي، أو من داخل أي سيرفر، ويمكنك بعد ذلك تشغيل ال image بداخل ال container.

وحتى يمكن لك أيضاً أن تعدل على ال container بأي إضافات تريدها، وبعد ترفع ال image المعدلة وبالتالي تكون image جديدة بها التعديلات التي قمت بإضافتها ويمكن لمن يرغب أن يقوم بتحميلها إذا قمت برفعها بشكل عام ومتاح للجميع، وأنت فقط من تستطيع رفع التحديثات على تلك ال repository. أما في حالة ال repositories الخاصة فيمكن لمن لديه صلاحية فقط بتحميلها وأيضاً رفع التحديثات بها وهي مناسبة لكافة أعمالك ومشاريعك الخاصة أو أي مشاريع خاصة للشركات.

ويمكن أيضاً بناء Private Registries بداخل الشركات مثل ال Docker Hub بحيث يكون مستودع لكل برمجيات الشركة داخلياً في الشبكة الداخلية بدلاً من رفعه على أي Online Registry.




أيضاً كما نلاحظ في الصورة أعلاه، سوف نجد بجانب كل برنامج عدد التحميلات والمفضلة وتستطيع الذهاب لصفحة التفاصيل وسوف تجد كيفية التشغيل لهذا البرنامج، بالإضافة الى ال Images الرسمية ستجد محدده بهذا الاسم، وأيضاً سوف تجد نتيجة فحص الأنتي فايروس لكل من ال Images المرفوعة على Docker Hub. وأيضاً سوف تجد رابط ال Docker Store وهو لل Images المدفوعة غير المجانية حتى تقوم بشراء الرخصة والحصول على الترخيص.

ولن تقلق من ناحية نظام التشغيل مجدداً، فاذا كنت تعمل في نظام التشغيل ويندوز فتستطيع تشغيل ال Windows Container وأيضاً تستطيع تشغيل ال Linux Container كما سيأتي بعد ذلك. وأيضاً لن تقلق من ناحية شكل وFormat البرنامج لأنه سيأتي بطريقة موحدة وهي ال Image. وطريقة مسح وتحديث ال Images بسيطة فكل ما عليك بتنزيل الإصدار الجديد وتشغيلها وإيقاف القديمة، وهكذا سنجد أنها حلت أغلب أو كل المشاكل التي كنا نواجها بالطريقة التقليدية سابقاً.

أثناء البرمجة والتطوير: فهناك العديد من الفوائد للمطورين في دوكر:

- تسريع دخول المطورين **Onboarding**: سوف نجد أنه بسهولة يمكن عمل ال Development Environment فمثلاً غالب المشاريع تحتوي على العديد من الأجزاء مثل Database و Queue Server و Caching Server و Indexing Server وستجد من الصعوبة تنصيب هذه الأجزاء وقد يقع أي مطور في مشكلة التنصيب هذه، وحتى للمطور نفسه فأحياناً يعمل من اكثر من

جهاز او بيئة ويحتاج أن يقوم بالعمل عليهم بسهولة، لذلك بعمل ال Images وتشغيلها فمن الممكن توفير الكثير من الوقت والجهد للمطورين، واذا كان المشروع يعمل بفكرة ال Microservices فمن الممكن أن تكون هناك Images أخرى مثلاً لل Identity وواحدة لل Orders مثلاً وهكذا سوف تجد المطور يعمل ويركز على المهام التي لديه ويستخدم المكونات مباشرة كما هي والتي أتت من فرق أخرى.

- لا تصادم بين النسخ **Software Conflicts**: فمن الممكن أن يعمل السيرفر على نسخة معينة من الفريم وورك ويكون به أكثر من برنامج يعتمد عليه، ولن تستطيع تحديثه حتى لا تقع البرامج الأخرى التي تعمل في مشاكل، ولكن بتطبيق فكرة ال Containers فسوف تجد أن كل برنامج يحتوي على ال Framework التي يحتاجها بغض النظر عما يتواجد في الجهاز الأساسي، وهذا يعني أنه لن تكون هناك تصادم بين البرمجيات مجدداً.

- عدم توافق ال **Environment**: مثلاً لو كان لديك بيئة تطوير Development Environment وبيئة للاختبار والتأكد من المشروع قبل رفعه على السيرفر النهائي Staging Servers ومن ثم السيرفر النهائي Production Server، فسوف تجد بدون دوكر فمن الصعوبة وضع كل البرمجيات والمتطلبات على هذه السيرفرات، وقد تحدث مشاكل بسبب اختلاف نوع النظام أو البرامج الموجودة عليه، لكن مع دوكر ستجد من السهولة توحيد تلك الأجهزة Consistency وتستطيع نقل ال Images وتشغيلها على هذه الأجهزة وسيعمل ال containers كما هو عليها جميعاً.

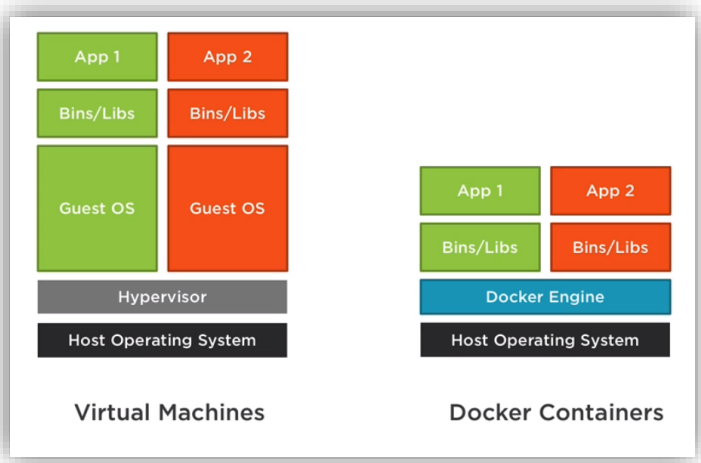
- نشر المشاريع بشكل أسرع **Ship Software Faster**: يمكن الاستفادة من أدوات الأتمتة وبناء Pipeline كامل بمجرد عمل push للسورس كود، فسيتم مباشرة بناء الكود كاملاً، ومن ثم في حال النجاح يتم تشغيل كافة الاختبارات وبعدها في حال النجاح يتم رفع ال images على ال Docker hub (أو على أحد ال Registries) واخيراً تحميلها من السيرفر وعمل Update حتى يعمل التحديث في السيرفر، وكل هذا بمجرد ضغطة زر واحدة فقط Automated Workflow. وهذا الأتمتة مهمة جداً للعديد من المشاريع، خصوصاً في الشركات الناشئة والتي تعمل بمفهوم ال Continues Delivery والتي تضيف وتحسن الخصائص بشكل مستمر، فوجود أتمته كاملة لنشر البرمجيات ورفعها أمر مهم وسوف يحفظ الكثير من الوقت الضائع في عملية رفع التحديثات يدوياً الى السيرفر (أو السيرفرات في حالة كان البرنامج يعمل على أكثر من سيرفر).

## ما الفرق بين ال Containers وال Virtual Machines؟

وهذا يعد من أكثر من التساؤلات في دوكر وال Containers عموماً، حيث كما ذكرنا سابقاً فأن ال Containers توفر عزل من ناحية نظام الملفات، الشبكات، قائمة العمليات وبداخل كل Container سوف تجد كأنك تعمل على جهاز منفصل، وهي نفس فكرة ال Virtual Machine والتي أيضاً توفر عزل كامل بين ال VM المختلفة وكأنها تعمل على ذلك الجهاز فقط.

لكن هناك فرق بينهم، حيث في ال VM يجب أن يكون هناك نظام تشغيل لكل VM وبعدها يتم تنصيب البرمجيات على ذلك النظام، بينما في ال Containers فسوف نجد أنه بالإمكان أن يعمل مباشرة على ال Host، وهذا أوفر للموارد وأسرع في العمل بكل تأكيد بالإضافة الى أنه يحفظ وقت مدير النظام بدلاً من إدارة 5 برامج تعمل على خمسة أنظمة تشغيل في ال 5 VMs ويقوم بتحديث الأنظمة باستمرار، فالآن سوف يعمل على نظام تشغيل واحد يحدثه باستمرار وكافة البرمجيات تعمل في ال containers ومعزولة عن الأخرى.

لكن بأي حال ال Containers هو ليس بديلاً عن ال VMs وقد تجد في كثيراً من الأحيان أن تستأجر VMs وتقوم بتشغيل عليه مجموعة من ال Containers بدلاً من شراء سيرفر مخصص Physical Servers وتحفظ الموارد وتقلل التكلفة. ومن ثم تستخدم ال Docker فيها لكي تستفيد من ال Containers.




## ما هو دوكر Docker؟

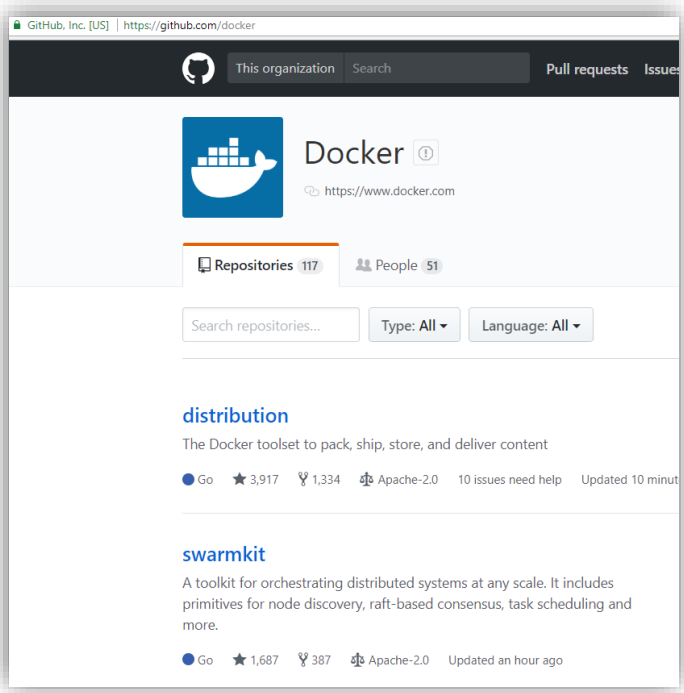
شركة دوكر .Docker Inc من سان فرانسيسكو بدئت في السابق بتطوير Platform as Service PaaS اسمها dotCloud لتقديم خدمات للمطورين على ال Amazon web services، وكانت تستخدم داخلياً مفهوم ال Containers للتشغيل بالإضافة الى نشر Deployment المشروع، وبالرغم من أن المشروع الأساسي هو لم يكن دوكر، ولكن تم تطوير هذه الجزئية واعطيت الاسم Docker ونشرت كمشروع مستقل في 2013.

وبهذا القرار الصائب، أصبحت شركة دوكر الأولى في تقديم ال Containers وحصلت على استثمارات عالية للمضي للأمام، وبيع مشروع ال dotCloud وركزت الشركة على ال Containers وخلال مسيرتها استحوذت على عدة مشاريع containers لشركات ناشئة أخرى، وهذا ما جعلها تتفرد في الساحة.

وشركة دوكر .Docker Inc هي ليست المالكة لمشروع دوكر Docker Project الآن، فهو مشروع مفتوح المصدر برخصة Apache وهذا يعني أي شخص بإمكانه تحميل الكود المصدر وتعديله واستخدامه طالما لم يخرق شروط هذه الرخصة. وهناك الكثير من الاسهامات بواسطة كبرى الشركات مثل IBM, RedHat وغيرها.

ويهدف مشروع الدوكر Docker Project لتقديم كافة الأدوات اللازمة لبناء Develop ونشر Ship/Deploy البرمجيات بشكل أفضل عما كان عليه في السابق. وهذا يعني أن هناك العديد من الأدوات والبرامج بداخل مشروع الدوكر وكل منها يختص بجزئية معينة، وأهمها هو ال Docker Engine وهو الذي يقوم بتشغيل ال Containers وبقية الأدوات تكون مساعدة لل Engine في عمله.

وسوف نجد أن مشروع الدوكر يتكون من العديد من الأدوات مثل ال Docker Engine, Docker Compose, Docker Swarm وغيرها من الأدوات.




اخيراً لأن مفهوم ال Images يعتبر أحد العوامل من نجاح ال Containers Technology فقد اجتمعت العديد من الشركات المهتمة لكي تضع معيار ومقياس standards لبنية هذه ال Image وال Runtime Environment وتم نشر ال Specification لهم. وظهرت حركة Open Container Initiative لوضع المعايير لل Containers بشكل عام، سواءً في شكل هذه ال containers وتشغيلها:

https://www.opencontainers.org

وبهذا المقياس فيمكن أن تقوم بتشغيل أي Image متوافقة مع ال OCI في أي Runtime Engine متوافق مع ال OCI وهذا ما يجعل مثلاً Docker Image تشتغل على Rkt Engine على سبيل المثال..

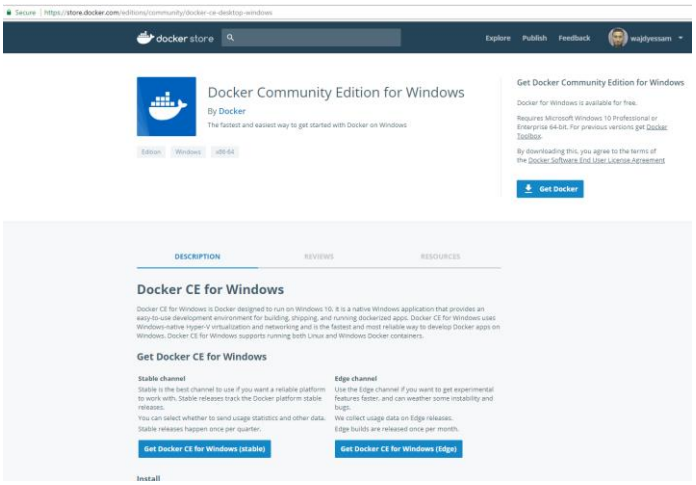
الفصل التالي سوف تتحدث عن تنصيب دوكر، سواءً في ويندوز ولينوكس.

# الفصل الثاني: البداية مع دوكر Docker

سوف نتحدث في هذا الفصل عن تنزيل دوكر في ويندوز، ولينوكس، ومن ثم نقوم بتشغيل بعض البرمجيات containers عليه مثل Nginx و IIS وسوف نرى كيف أصبحت عملية تنزيل البرمجيات سهلة للغاية وبدون أي اعدادات تذكر، وسوف نوضح الفرق بين ال Linux containers و windows containers وكيف يمكن تحويل دوكر ليعمل على أي منهم. وأخيراً سوف نرى العزل Isolation التي توفره ال Containers.

## تحميل دوكر على نظام تشغيل ويندوز

في نظام التشغيل ويندوز، قم بتحميل الدوكر من خلال الرابط Docker Community Edition for Windows المتواجد على Docker Store:




ودوكر للويندوز يحتاج أن تعمل على نظام التشغيل ويندوز 10 اما Pro, Enterprise, Education أو على Windows Server 2016 وغير ذلك فلن يعمل، بالإضافة الى تحديث نظام التشغيل بأخر التحديثات، وهذه من متطلبات دوكر للويندوز، فاذا لم يكن لديك هذه المتطلبات فيمكن تنزيل Docker Toolbox وهي تعتبر الآن قديمة Legacy ولكنها للأجهزة التي لا تتوافق مع متطلبات دوكر للويندوز، سواءً Windows 7 or Windows 8 ولكن يجب أن يكون 64 بت، ويمكن تنزيلها من هنا Docker Toolbox:

ولتحميل دوكر على ويندوز سيرفر 2016 فلن نحتاج اليه وانما فقط يكفي ال Docker Engine.

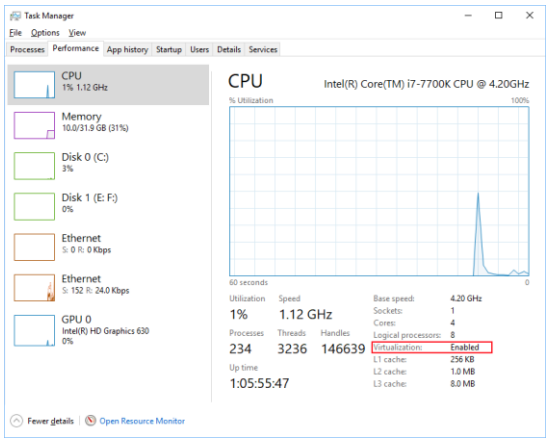
بعد تحميلك Docker for Windows.exe وقم بتنزيله بالخطوات الافتراضية التالي-التالي الى أن ينتهي، وبعد الانتهاء يمكنك تشغيله من خلال قائمة البرامج وفتح Docker for Windows.

وهناك بعض النقاط والمتطلبات المهمة قبل تنصيب الدوكر، وهي:

عند تنصيب الدوكر سوف يقوم بتفعيل ال Microsoft Hyper-V في حال لم يكن مفعلاً، ويجب عليك حينها إعادة تشغيل الجهاز. ولن يعمل أي VM اخر مثلاً Virtual Box حيث لا يمكن تشغيل اثنين في أن واحد، ولن تعمل بجانب ال Docker for Windows.

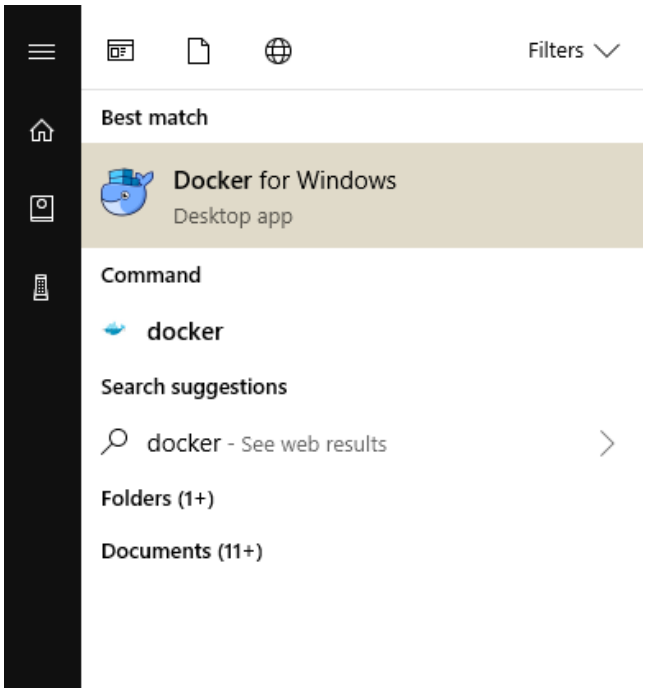
إذا كنت تطور في نظام الأندرويد فلن تستطيع تشغيل ال Android Studio Emulator لأنه يستخدم ال Hardware Accelerated Execution Manager (واختصاراً ال Intel HAXM) وهذا يتعارض مع ال Hyper-V ولا يمكن تشغيلهم في أن واحد، ويجب عليك إعادة التشغيل في كل مرة أردت إيقاف أحدهم والتبديل للبرنامج الاخر.

يجب تفعيل خاصية ال Virtualization من ال BIOS وذلك حتى يعمل دوكر ويستطيع تشغيل ال Linux Containers بداخل الويندوز باستخدام ال Hyper-V، وغالباً ما تكون مفعلة في الوضع الافتراضي، يمكنك التحقق من ذلك من ال Task Manger والذهاب لل CPU ومشاهدة ال Virtualization Enabled:

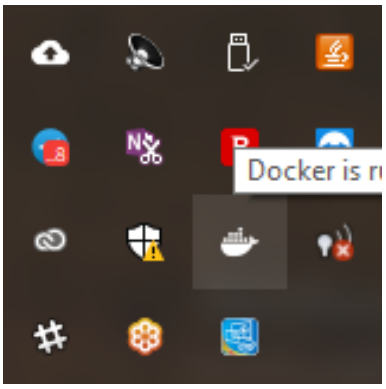



أثناء التنصيب سوف تجد الرسالة، وستجد أنه يمكن تفعيل ال Windows Container مباشرة بدلاً من الافتراضي Linux Container، ولكن على أي حالة يمكن تغييرها فيما بعد من خصائص دوكر. لذلك اتركها كما هي:

بعد التثبيت يمكنك فتحه من قائمة البرامج:

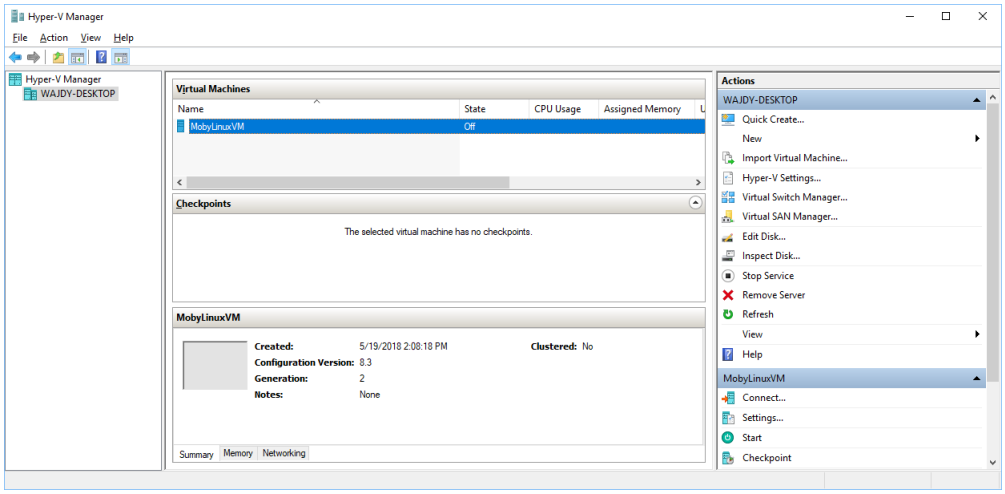



سوف تجد أيقونة دوكر بجانب الساعة ويمكنك فتح الخصائص والاعدادات والتحويل من ال Windows Container الى Linux Containers وغيرها من المهام، وذلك بالضغط عليها بالزر الأيمن، وسوف نتحدث عن ذلك فيما بعد. حالياً عند تثبيت الدوكر سوف يعمل على ال Linux Container mode.

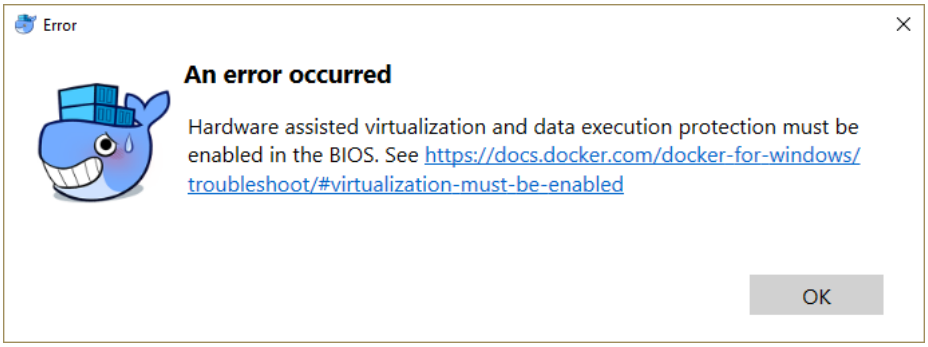



بعد تثبيت ال Docker for Windows سوف ينزل معه العديد من الأدوات مثل Docker Engine و Docker CLI Client و Docker Compose و Docker Machine.

ولأن ال Linux Containers يعني أنه يمكن تشغيل برمجيات لينوكس على ويندوز، وهذا سيتم عن طريق Virtual Machine يتم إنشائها مباشرة ويتم تشغيل كل ال Linux Container بها، فقم بفتح ال Hyper-V Manager من قائمة البرامج وسوف تلاحظ وجود VM من نظام MobyLinux تعمل وهي التي يستخدمها دوكر للتعامل مع ال Linux Containers.

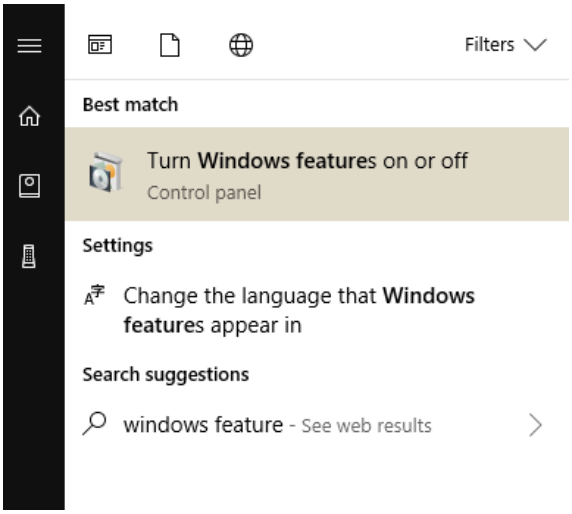



إذا ظهرت لك هذه الرسالة:

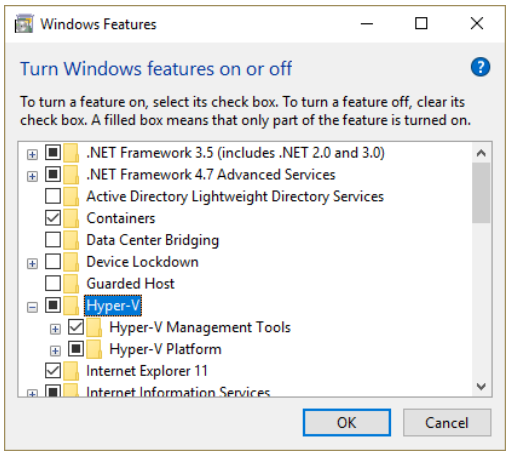



فعليك بالتأكد من دعم ال BIOS وتفعيله لل Virtualization، كما تطرقنا سابقاً، وإذا لم تكن مفعلة فيجب تفعيلها من اعدادت ال BIOS.

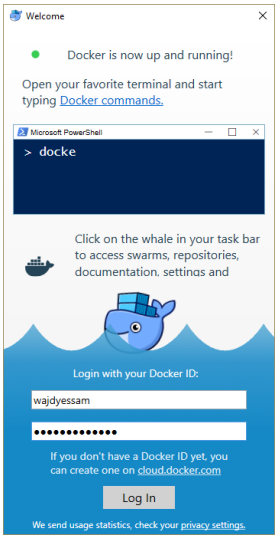
إذا كانت مفعلة فهذه المشكلة تحتاج الى أن تقوم بتعطيل ال Hyper-V يدوياً وإعادة التشغيل ومن ثم إعادة تفعيله مجدداً وإعادة التشغيل ومن ثم سوف يعمل دوكر معك، ولا أدرى لماذا هذه الخطوات ولكنها قد تحدث أحياناً. فقم بفتح خصائص الويندوز:



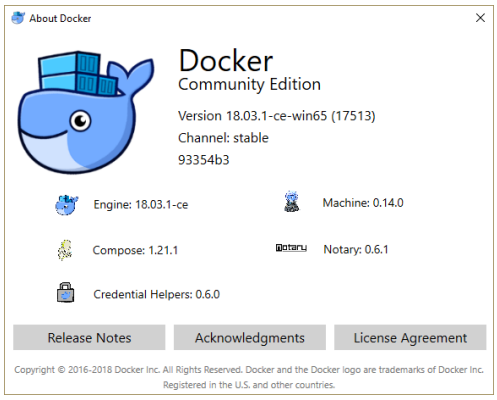
ومن ثم فعل ال Hyper-V:




أو قم بتعطيلها وأعد التشغيل، ومن ثم قم بتفعيلها وأعد التشغيل مجدداً، وبعد إعادة التشغيل سوف يعمل معك بالشكل الصحيح. بعد أن يعمل الدوكر لأول مرة سوف تجد هذه النافذة التالية، وقم بالتسجيل واخذ Docker ID لكي تقوم بتسجيل الدخول فيه:




ولمعرفة النسخة التي تعمل عليها يمكن الضغط على أيقونة دوكر بالأسفل بالزر الأيمن والضغط على about وسوف تجد معلومات النسخة (وقد تتغير من النسخة هنا لأن دوكر تحدث البرنامج كل شهرين تقريباً)، لكن تظل المفاهيم واحدة:



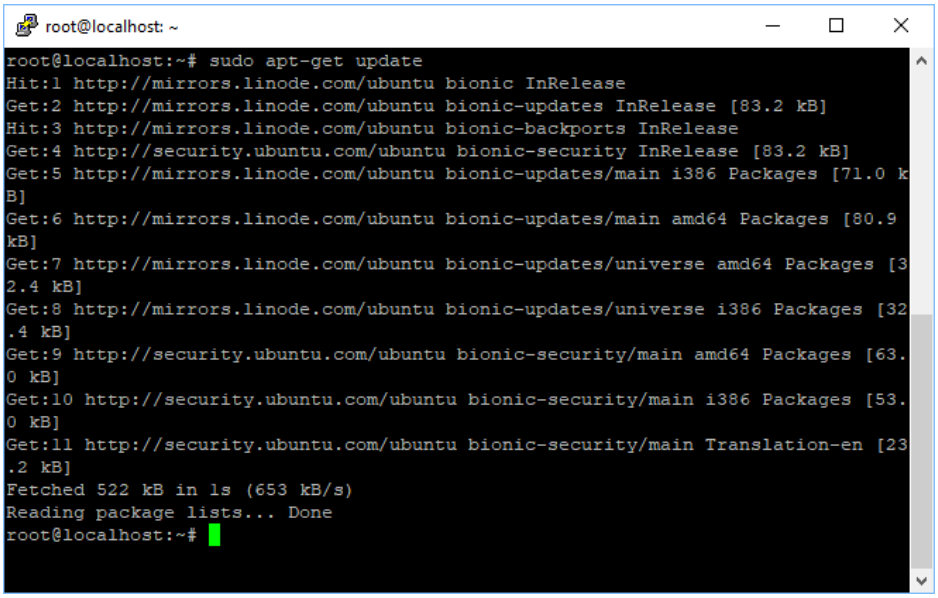
بهذا الشكل نكون قد أنهينا من تنزيل دوكر على الويندوز.

ملاحظة: إذا كنت تعمل على نظام تشغيل Mac فيمكن أيضاً تحميل Docker for Mac ويمكن تطبيق كافة الجزئيات السابقة ما عدا جزئية ال Windows Container. وبالنسبة لل Windows Server 2016 فلن تحتاج لتحميل ال Docker for Windows وانما مباشرة تحتاج ال Engine فقط، وهناك نسخة مخصصة لل Windows Server 2016 وهي ال Enterprise Edition ويمكنك تحميلها بسهولة أيضاً في النظام.

## تنصيب دوكر على لينوكس

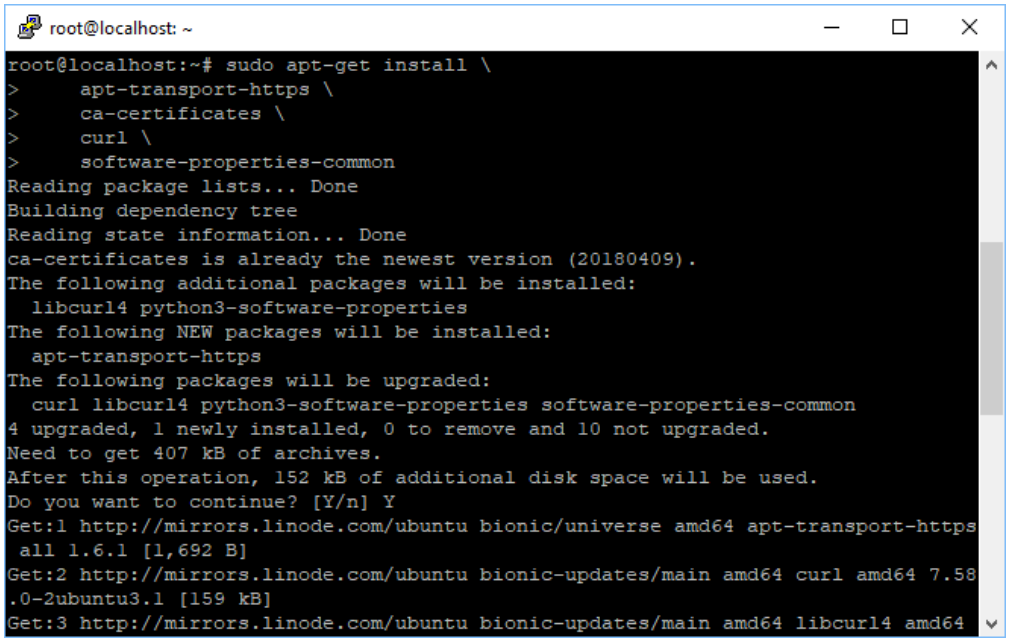
سيتم تنصيبه على نظام Ubuntu 18.4 LTS وطريقة التنصيب في لينوكس بسيطة، فنقوم أولاً بتحديث ال apt من خلال الأمر:

```
1. $ sudo apt-get update
```




بعد ذلك سنقوم بتحميل بعضاً من ال packages حتى يتم التحميل عبر https بواسطة apt:

```
1. sudo apt-get install \
2.     apt-transport-https \
3.     ca-certificates \
4.     curl \
5.     software-properties-common
```



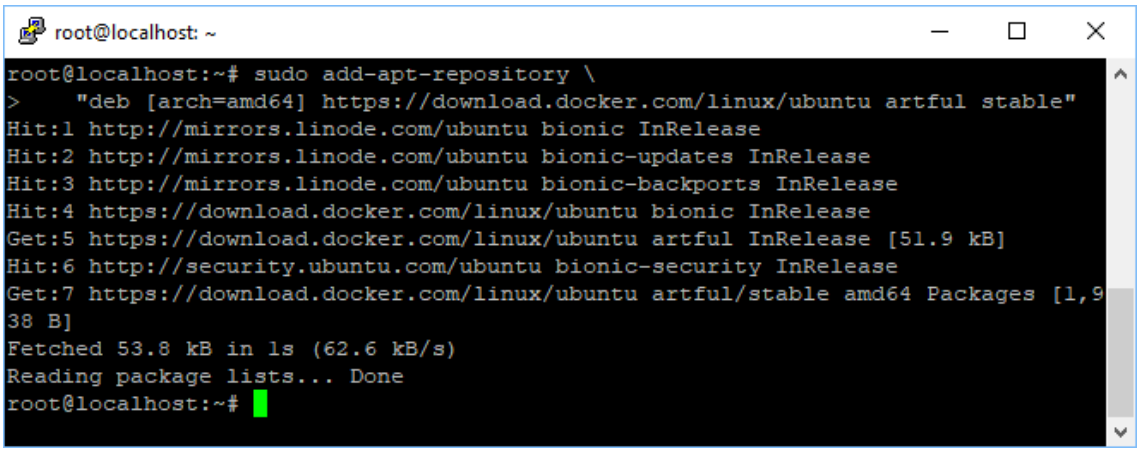

ستقوم بتحميل الدوكر GPG Key:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

```
root@localhost:~# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo
 apt-key add -
OK
```

سنقوم الآن بتحديد ال Stable Repository لتحميل دوكر:

```
$ sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu artful stable"
```

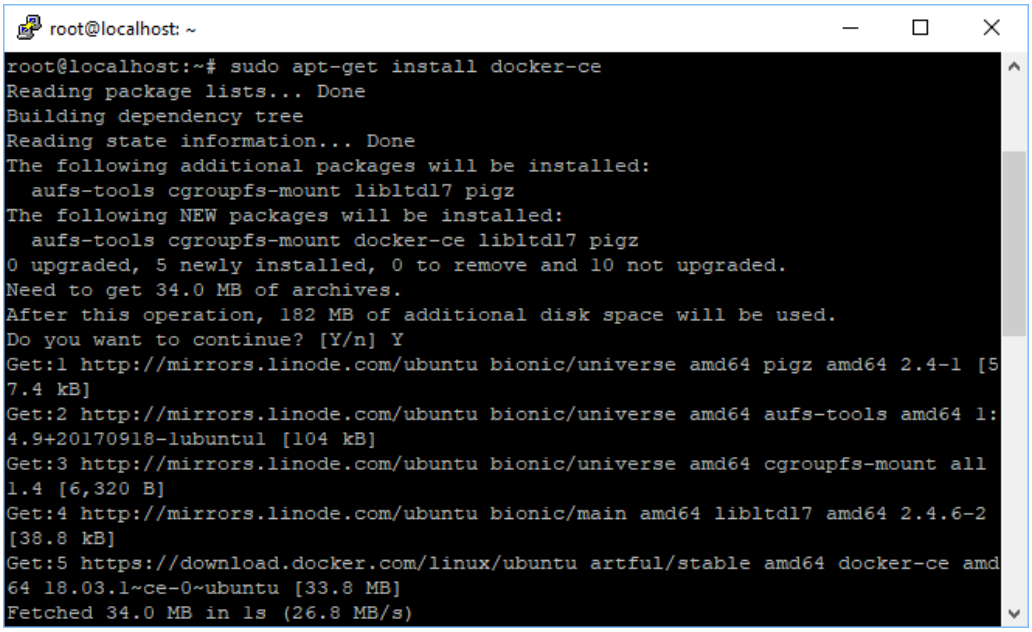



نقوم بتحديث ال apt مجدداً:

```
sudo apt-get update
```

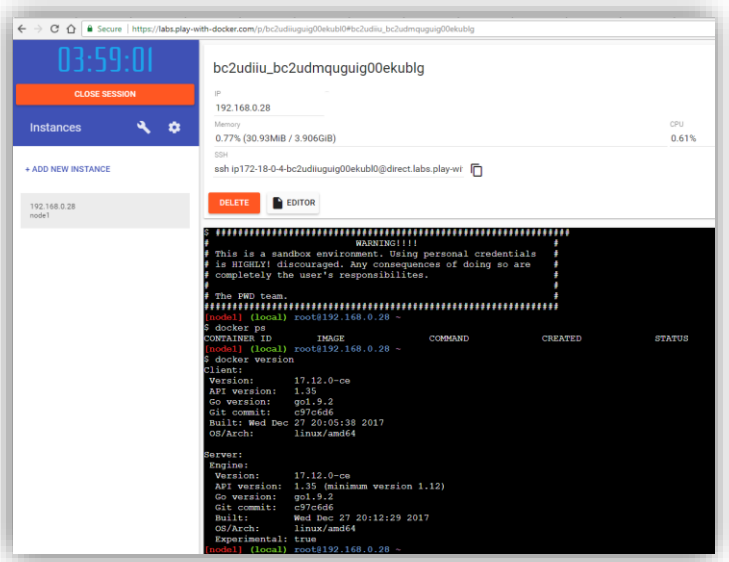
الآن نقوم بتحميل دوكر:

```
sudo apt-get install docker-ce
```




## تجربة دوكر بدون تثبيت أي برامج

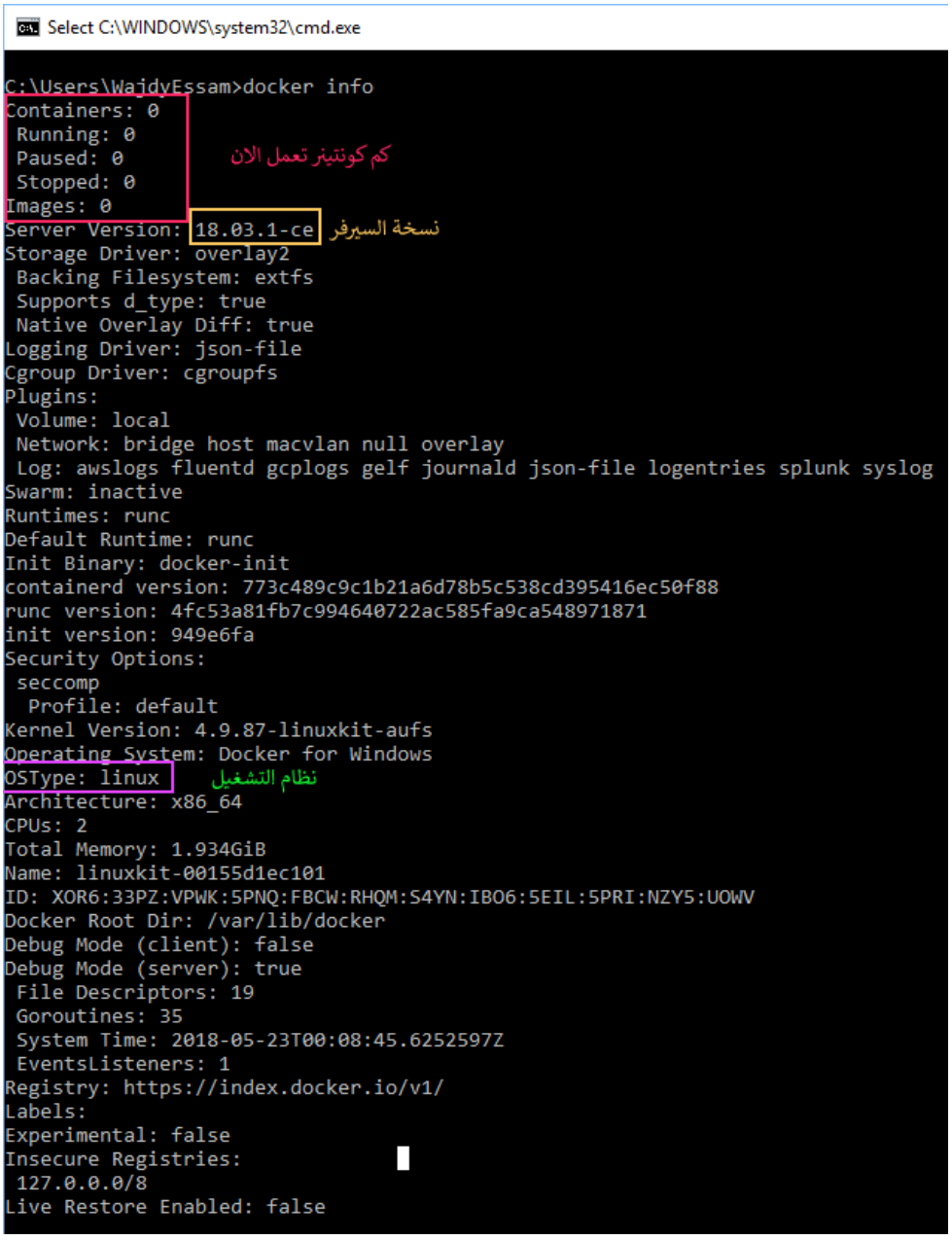
في حال لم تريد تنصيب دوكر، فيمكنك استخدام الموقع Play with Docker وتستطيع اضافة جهاز أو أكثر من جهاز ومن ثم يمكنك تجربة الأوامر التي سوف نأخذها في الفقرات التالية. وسوف يتم اعطائك Session لمدة 4 ساعات للتجربة في جهاز تخيلي.



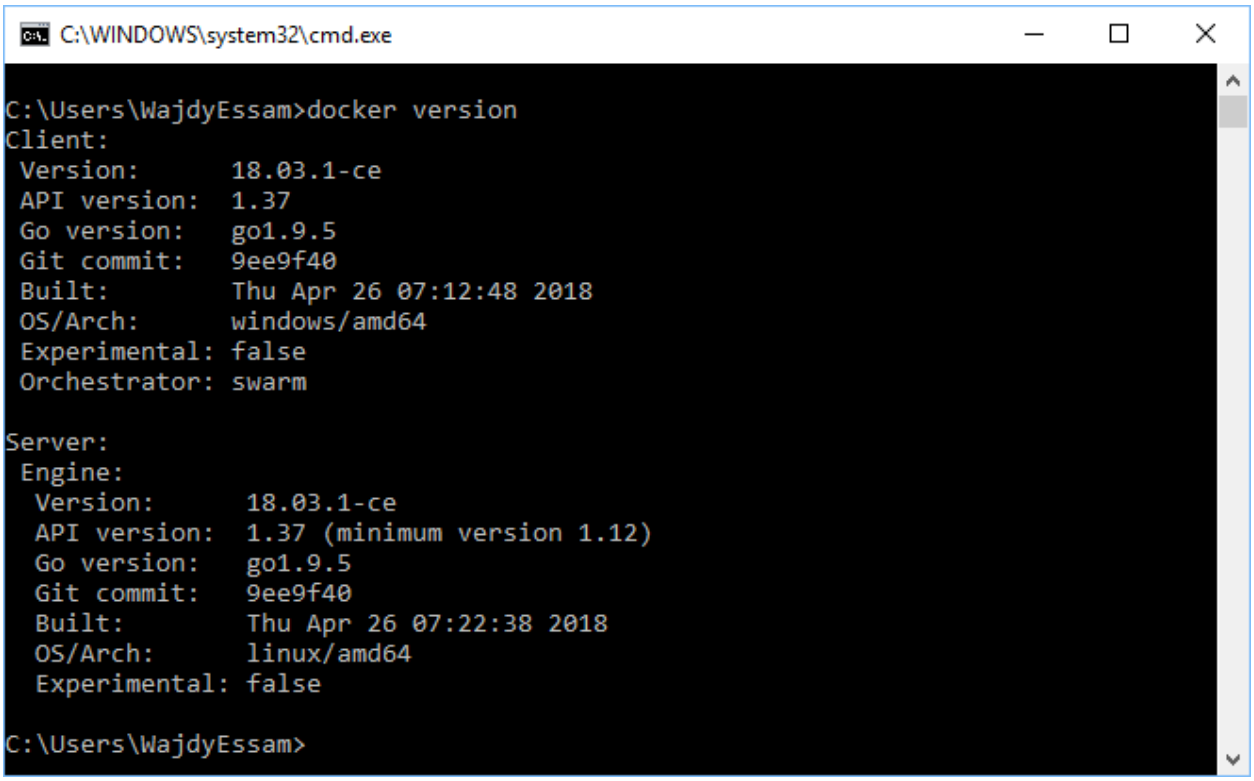
## البداية في أوامر Docker

سوف ننتقل الآن الى التعامل مع دوكر، ولحسن الحظ كافة الأوامر التي سوف نتعامل معها سوف تعمل على دوكر بغض النظر عن نظام التشغيل المستخدم، سواءً ويندوز أو ماك أو لينوكس.

أول أمر هو لمعرفة معلومات عن الدوكر وهو docker info وسوف يخرج معلومات عن ال docker engine:

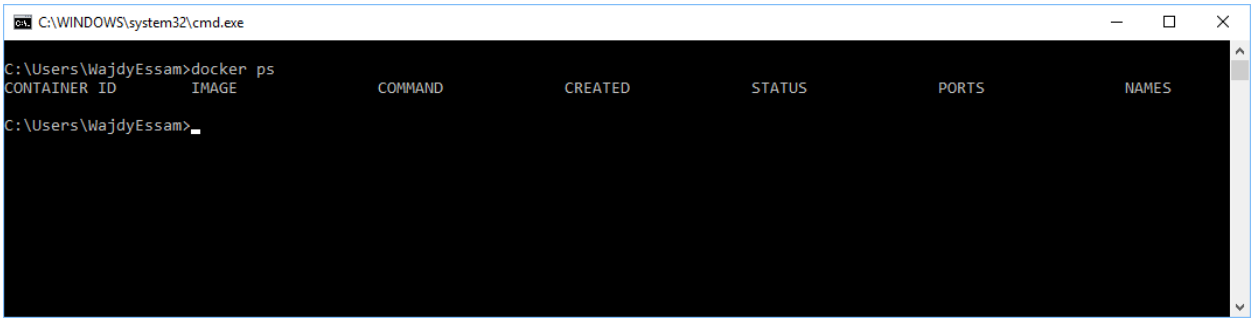



أيضاً يمكن استخدام docker version لمعرفة معلومات السيرفر والكلاينت كما يلي:




سوف تلاحظ أن الكلاينت يعمل على ويندوز وذلك لأن نظام الشغيل هو ويندوز، بينما السيرفر يعمل على ال Linux (باستخدام VM)، وكما ذكرنا يمكن تغييره الى ويندوز كما سيأتي فيما بعد لتشغيل Windows Containers مباشرة على النظام.

لننتقل الى أولى الأوامر: وهو docker ps لمعرفة حالة العمليات process status، وسوف يخرج كل ال containers التي تعمل الآن:

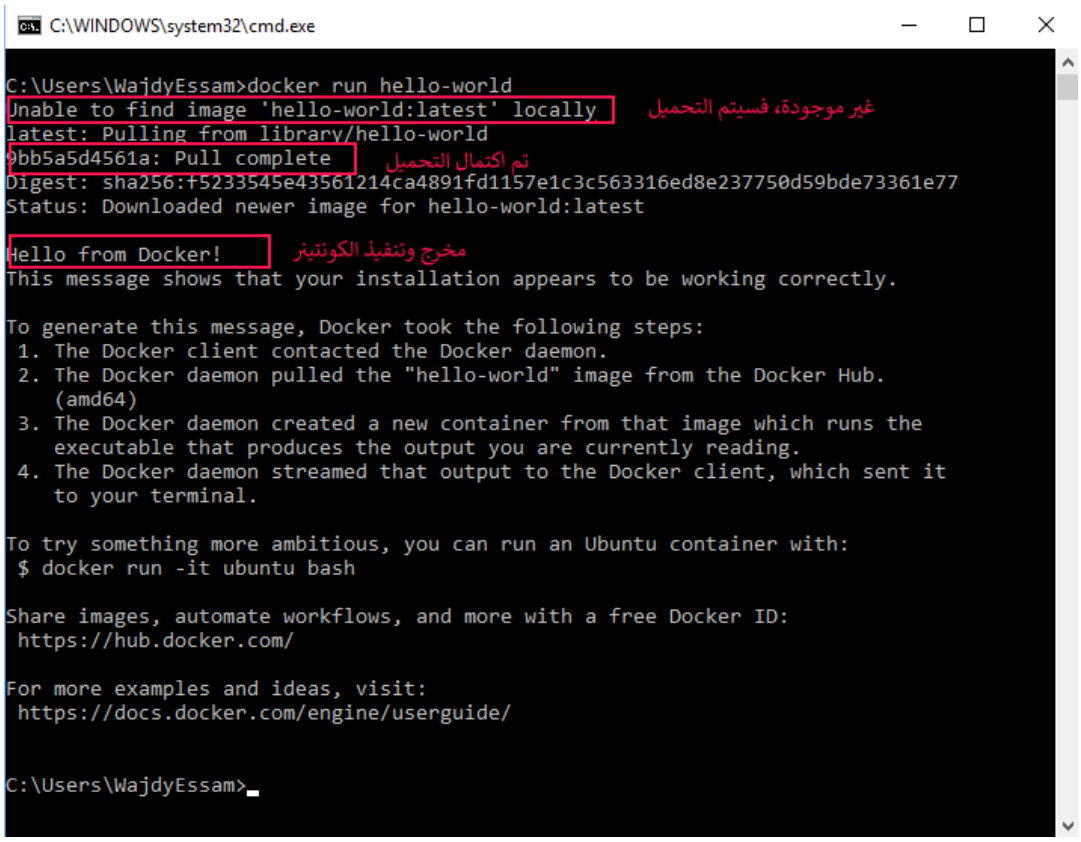



سوف تلاحظ أنه لا يوجد أي Container يعمل الآن، والقائمة فارغة.

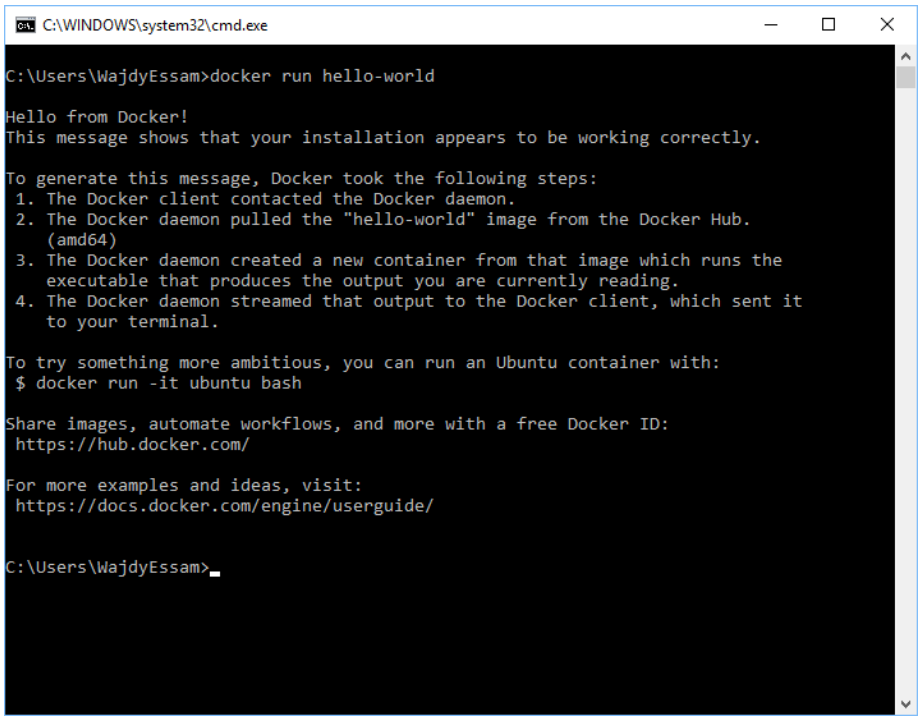
سوف نقوم بتشغيل أول container وهو يعتبر Hello-World وأسمه كذلك أيضاً، من خلال الأمر:

```
docker run hello-world
```

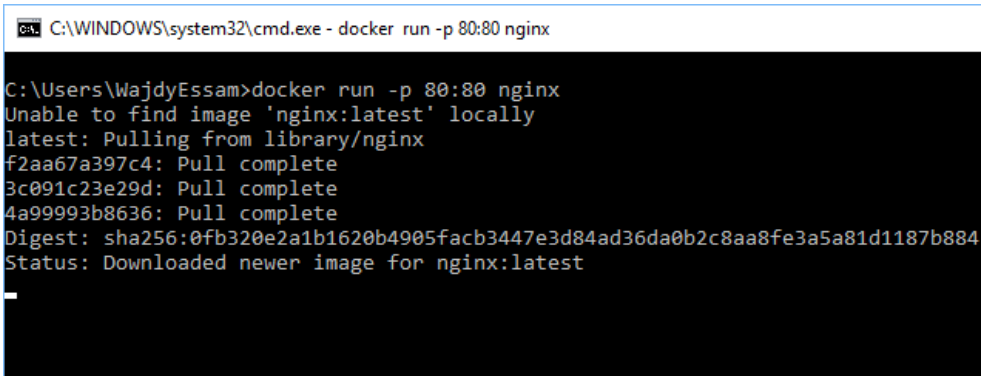
وكما ذكرنا سابقاً أن الأمر run سوف يقوم بعمل pull وبعدها يقوم بتشغيل ال container في أن واحد:



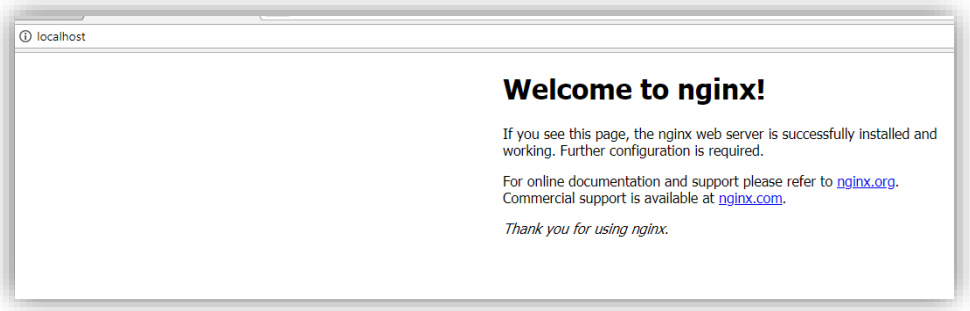

بعد أن عملت ال container سوف تنتهي مباشرة، ويمكن أن نشغلها مرة ثانية بنفس الأمر، وسوف نلاحظ أنه لن يتم تحميلها مجدداً، والسبب أن Image أصبحت موجودة في الجهاز وسيتم تشغيل ال container منها مباشرة:



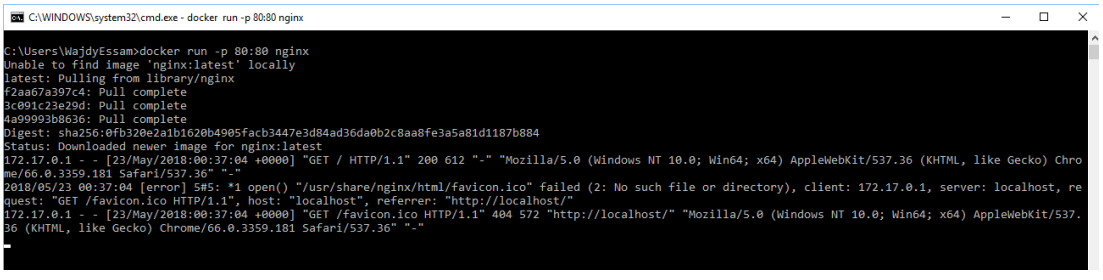
سوف نقوم بتشغيل مثال أخر وليكن ويب سيرفر مثلاً Nginx من خلال الأمر docker run –p 80:80 nginx، وسوف نلاحظ الآن استخدام المعامل p– وهو لتحديد البورت فأول بورت 80 هو الخاص بالجهاز الحالي وهذا يعني أريد أي طلب بالبورت 80 من ال host ينتقل لهذه ال container، وبداخل ال container سوف يكون ال nginx يستمع للبورت 80 داخل ال container يتم التقاط الطلب بواسطتها، وهذا يعني كأنه يتم تشغيل ال nginx على بورت 80 في الجهاز مباشرة، ويمكن أن نغير البورت الأول مثلاً 8080 وبالتالي سوف يتم طلب 8080 ويتم تمريره لل nginx الذي يستمع عبر بورت 80.




سوف نلاحظ أن هناك 3 أشياء تم تحميلها وهي الطبقات Layers لأنه غالباً ال Image يكون بها أكثر من طبقة وسوف نتحدث عن ذلك لاحقاً، وبعد أن يعمل ال nginx سوف تجد أنه لا مخرج منه لكنه يعمل الآن، ويمكن فتح المتصفح مباشرة localhost وسوف تجد رسالة من الويب سيرفر:



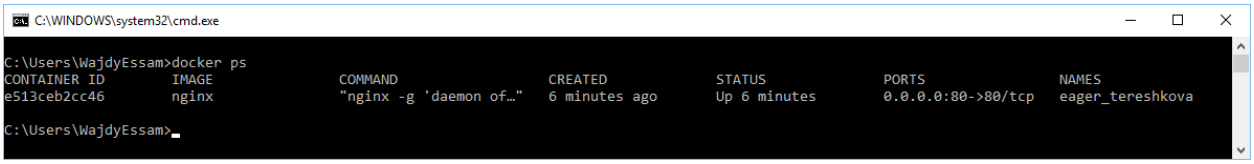

هكذا بكل بساطة عمل ال nginx من خلال أمر واحد فقط، وستجد بعد فتحك الصفحة أن هناك مخرجات على الكونسول التي يعمل بها ال nginx وهي عبارة عن logs لأي طلب يأتي للويب سيرفر:




فكرة ال containers مع ال Docker غيرت من مفاهيم تعلم الأدوات، فلن تحتاج الى أن تتعلم كيف يمكن تنصيب البرنامج قبل أن تستخدمه، فيمكنك الآن أن تستخدمه مباشرة، وبعدها قرر هل تريد أن تتعلم طريقة التنصيب وكيف يتم استخدامه.

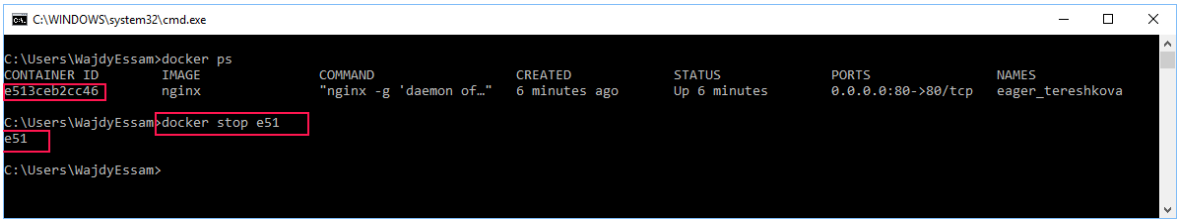
الآن لكي اخرج من ال container التي تعمل عليك بعمل detach وذلك اما من خلال CTRL+C أو CTRL+PQ لكي تخرج منها، ولكن سوف تظل ال container قيد العمل، وانما هو خروج من مخرجات تلك ال container التي تعمل.

ولكي تتأكد من ذلك قم بمشاهدة ال containers التي تعمل بالأمر docker ps، وسوف تجد أن ال nginx مازال يعمل:

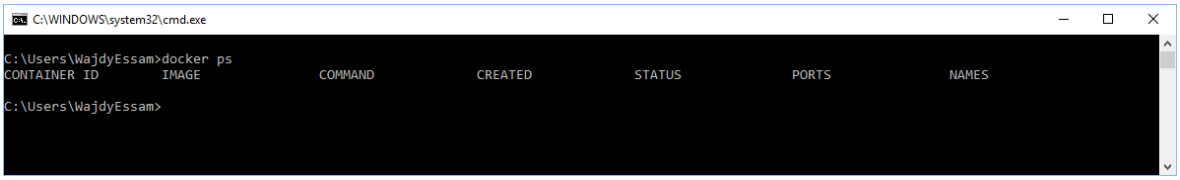



سوف تجد ال Container ID وهو رقم ال container ويمكنك التعامل معه من خلال الرقم (أو جزء من الرقم فقط)، اسم ال Image، الأمر الذي تم تشغيله وهو يأتي مع ال image، وقت الإنشاء، حالة ال container، البورت المفتوح، واسم عشوائي يعطي لل container (في حال لم تتسد اسم لها) ويمكنك التعامل معها أيضاً من خلال الاسم. ويمكنك أيضاً فتح ال localhost وسوف تجد الموقع ما زال يعمل.

الآن سوف نستخدم أمر إيقاف ال container وذلك من خلال docker stop Identifier ويمكن استخدام معرف ال container سواء كان ال CONTAINER_ID في الصورة السابقة أو الاسم name، وبالتالي يمكن أن يتم استخدام docker stop e51 أو docker stop eager_tereshkova وكلهم سوف يعطوا نفس النتيجة، ولاحظ لم نقم بكتابة كامل المرقم وانما أول جزء منه فقط، ويمكن وضع أي عدد من الأرقام، لكن في حال هناك أكثر من container كانوا لهم نفس البداية فسوف يحصل conflict وتخرج لك رسالة خطأ بأنه هناك أكثر من container بنفس البادئة وعليك بوضع عدد أطول من الأرقام حتى يتم معرفة ال container الذي تقصده.

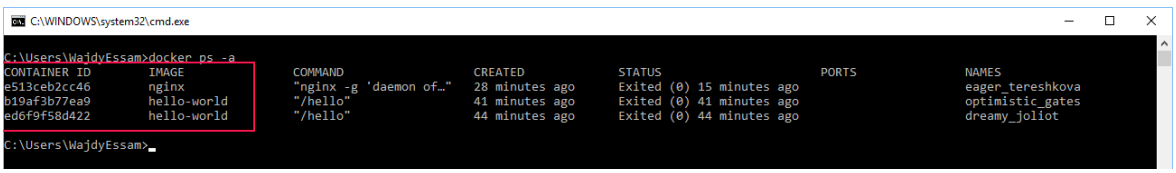



إذا تم الإيقاف بشكل صحيح سوف تحصل على نفس الجزء من المعرف الذي قمت بإدخاله، ويمكنك التأكد من أنه تم الإيقاف عن طريق docker ps ولن تجد ال container تعمل بعدها:




وأيضاً إذا ذهبت للموقع localhost فلن تجده يعمل لأن الويب سيرفر لا يعمل.

سوف نقوم الآن بتشغيل نفس container السابقة، وذلك من خلال الأمر docker start identifier، وقبل أن نقوم بتشغيلها، سوف نعرض الأمر docker ps –a والذي يظهر كل ال containers سواء التي تعمل أو التي تم ايقافها، وسوف تشاهد المخرج التالي:



سوف تلاحظ 3 containers موجودة، أول واحدة هي ال nginx التي تم ايقافها قبل قليل، والأخريات هم ال hello-world حيث قمنا بتشغيلها مرتين docker run hello-world وهي تنتهي مباشرة بعد العمل، لذلك توجد نسختان هنا.

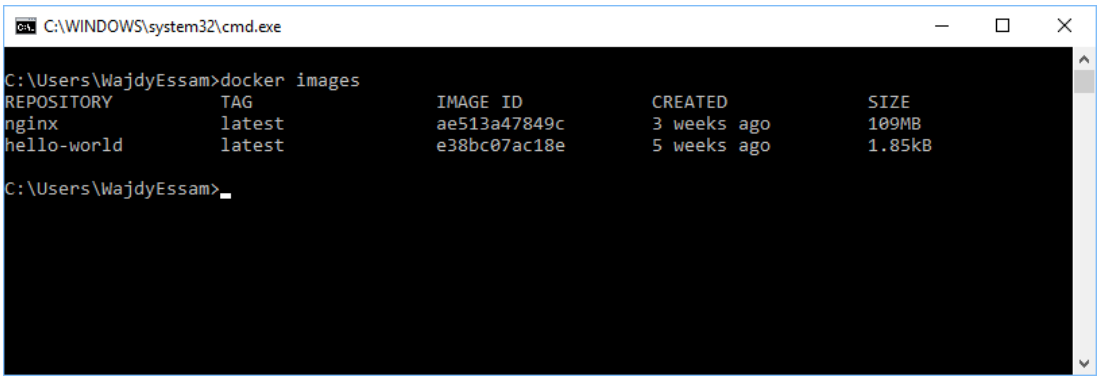
يمكن تشبيه الفكرة بالطريقة التقليدية عندما تقوم بتشغيل برنامج ما وايقافه عن العمل فهذا لا يعني أنه تم مسحه، وانما يكون موجوداً، ونفس الفكرة في ال containers فبعد ايقافها فهي تكون موجودة ويمكن حذفها أو إعادة تشغيلها مجدداً.

سنقوم الآن بتشغيل ال nginx container باستخدام الامر start من خلال معرف ال container أو اسمه، وسوف يكون المخرج الرقم نفسه لدلالة على اكتمال العملية:

```
C:\Users\WajdyEssam>docker start e5
e5
```

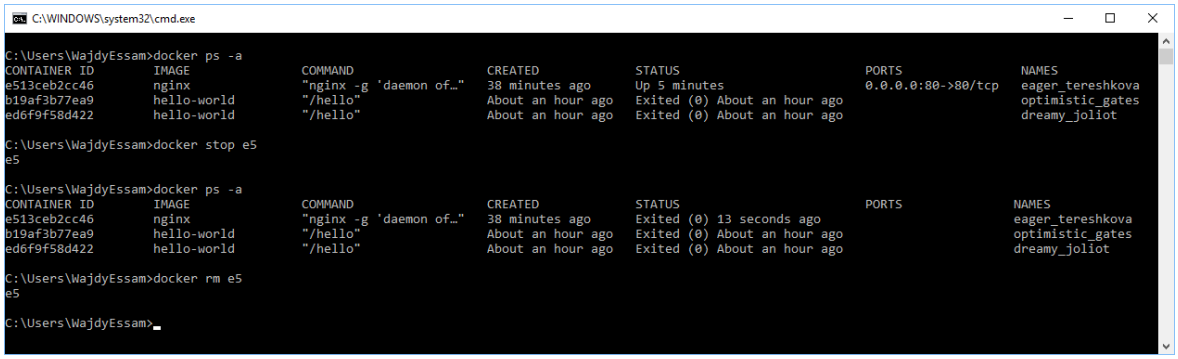
يمكن الآن فتح الموقع مجدداً localhost وستجده يعمل بدون أي مشاكل. ويمكنك مشاهدة ال container يعمل عند تنفيذ الأمر docker ps.

من الأوامر الشائعة أيضاً هو الأمر docker images وسوف يعرض كل ال Images التي تحميلها ومتواجدة على الجهاز الآن كما يلي:

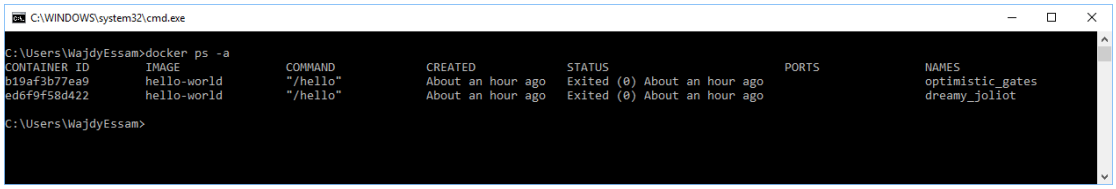



وهذه ال images تعتبر هي القالب التي يتم منها إنشاء ال container.

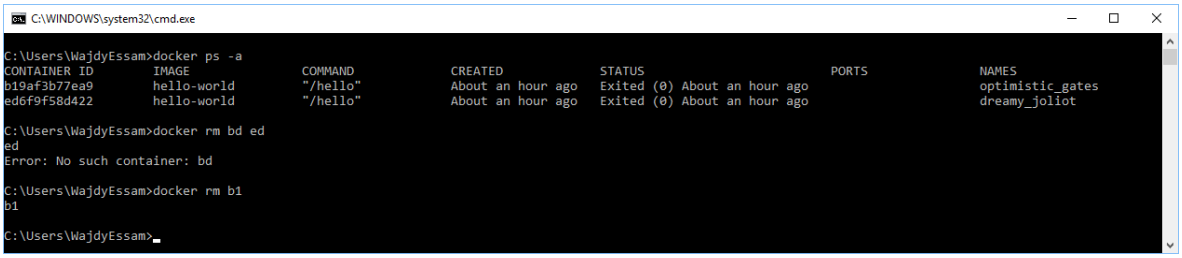
الآن لحذف ال container سوف نستخدم الأمر: docker rm identifier، وقبل أن نقوم بحذفه يجب أن يكون متوقفاً عن العمل، وسوف نقوم الآن بحذف ال nginx container كما يلي:



سوف تجد أعلاه قمنا أولاً بعرض كافة العمليات docker ps – a وسوف تلاحظ أن ال nginx تعمل، وقمنا بعدها بإيقافها من خلال docker stop e5، وبعد ذلك عرضنا العمليات مجدداً والآن أصبحت متوقفة عن العمل، واخيراً قمنا بحذفها docker rm e5، الآن في حال قمنا بعرض العمليات مجدداً فلن تكون موجودة:

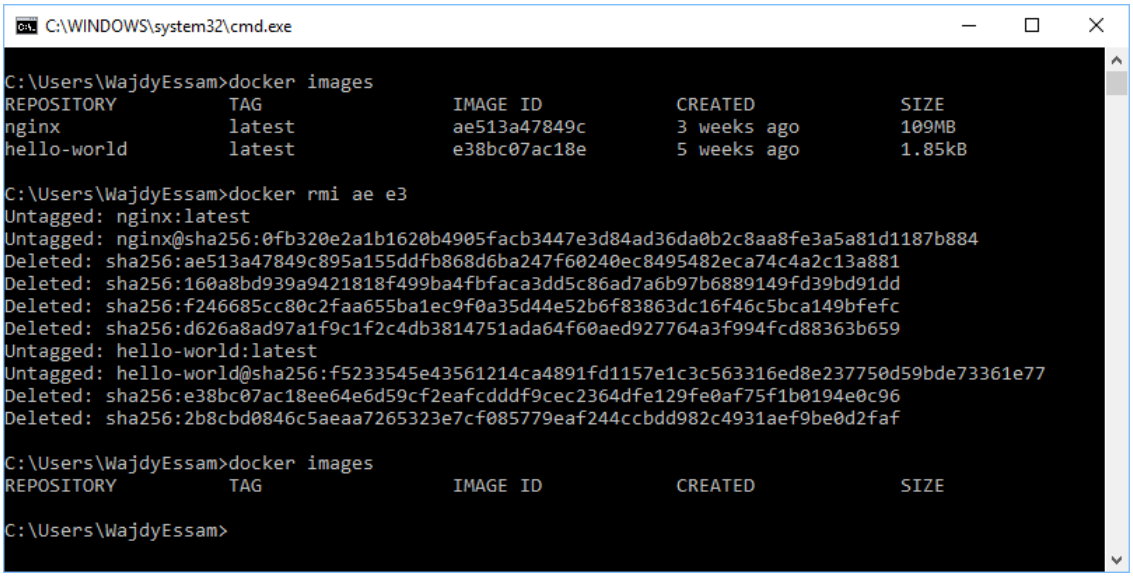



سوف نقوم الآن بحذف بقية ال containers وطالما هي متوقفة فيمكن اصدار امر الحذف مباشرة، ويمكن حذفهم في آن واحد من خلال كتابة المعرفات وبينهم مسافة، كما يلي:




سوف تجد أنه تم كتابة اسم ال container الأول خطأ، وتم حذف الثاني، ومن ثم قمنا بكتابته صحيحاً وتم الحذف في الأمر الذي يليه. الآن عند كتابة docker ps –a فلن تجد أي containers تعمل أو متوقفة.

لكن ال Images ما زالت متواجدة، فعند كتابة docker images فسوف ترى ال images ولحذفها عليك بكتابة الأمر docker rmi identifier وكتابة معرف ال image، وفيما يلي سيتم حذفهم في آن واحد:

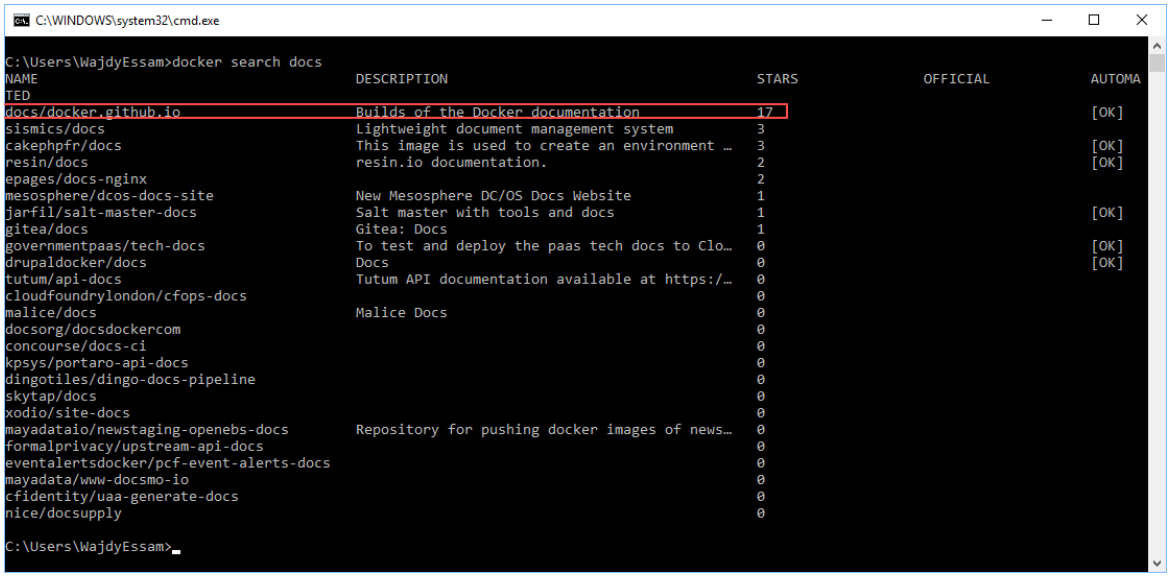



وسيتم حذف كافة ال Layers في ال images وبعدها عند كتابة docker images فلن تجد أي image متواجدة.

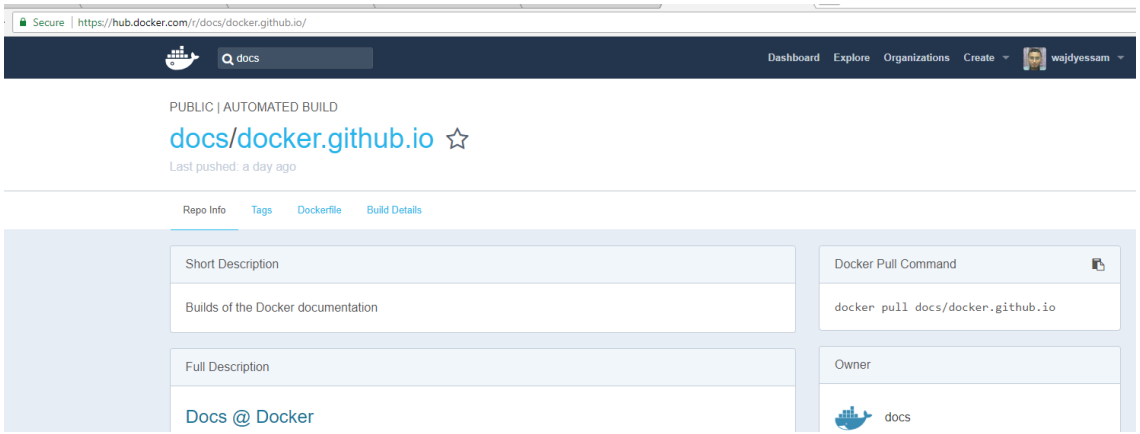
## مثال ثاني

سوف نأخذ مثال أخر، وليكن هذه المرة توثيق ال Docker حيث يمكن فتح التوثيق بأكمله أونلاين من خلال هذا الرابط Docker Documentation.

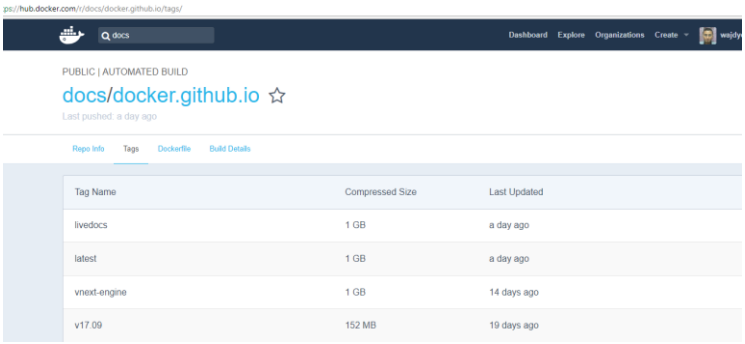
ولكن يمكنك أيضاً تحميله offline ك docker images وتشغله محلياً على ويب سيرفر، وسوف نقوم بذلك، ويمكن استخدام الأمر docker search docs لكي ترسل أي كلمة بحث وارسلنا الآن docs لكي تخرج كل ال images المتواجد على ال docker hub بكامل الاسم لها، كما يلي:



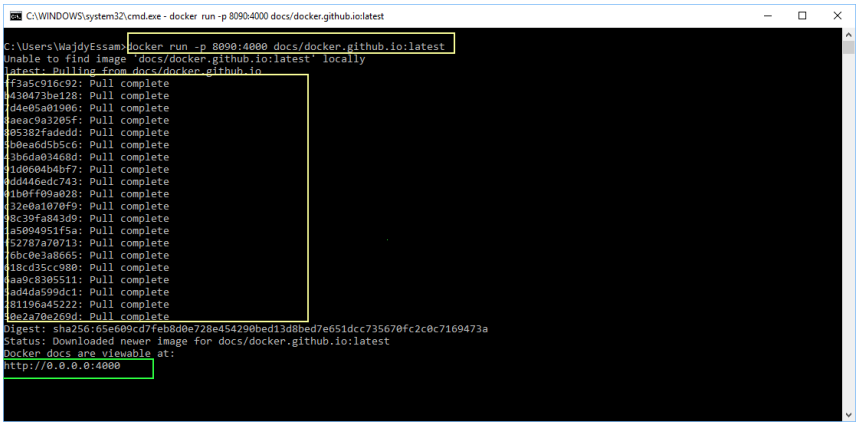

أو يمكنك فتح ال docker hub ومن ثم البحث عن docs وسوف تجد ال docker docs وتستطيع مشاهدة تفاصيلها كما يلي:




وقبل التحميل سوف نلقى نظرة عن ال tags وهي مشابه لل git tags فهي نسخ أو مراحل ما من هذا البرنامج، ويمكنك تحميل أي منها من خلال ال tag، وإذا لم يتم تحديد فسوف يتم تحميل ال latest (التاق الافتراضي).



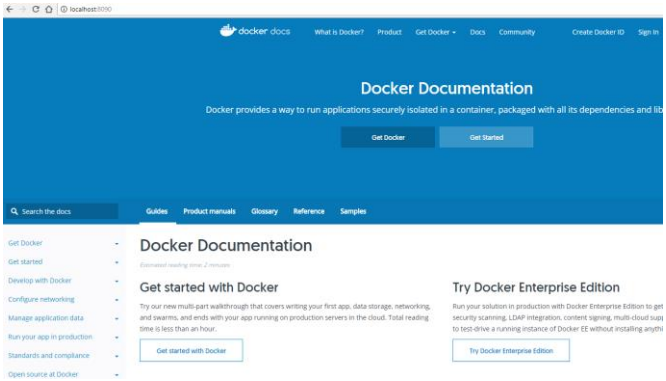

سوف تلاحظ أن حجم ال image هي 1 قيقا، ولذلك قد تأخذ وقتاً للتحميل بحسب سرعة الاتصال لديك، وسوف نقوم بتحميل وتشغيل ال image بالأمر run كما يلي:




لاحظ أننا قمنا أولاً بتشغيل ال container بالبورت 8090 على ال host، أما البرنامج من داخل container فسوف يعمل على البورت 4000، وقمنا أيضاً بكتابة ال tag الذي نريده، ويمكنك تحديد أي نسخة تريد، وطالما حددنا latest فسوف يعطي اخر نسخة، وهذا الذي يحدث في حال لم يتم تحديد أي نسخة وهو تحميل ال latest.

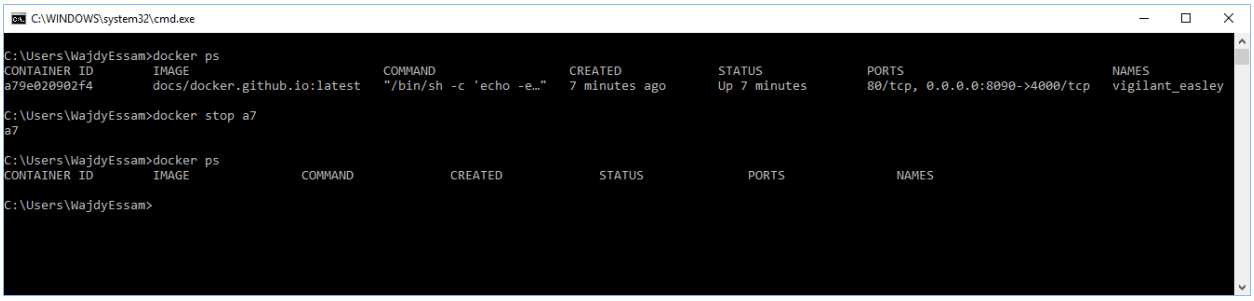
لاحظ أيضاً رابط ال image يبدأ بال docs/docker.github.io حيث ال docs هذه تعتبر اسم الشركة، وبداخل هذه الشركة سوف تجد العديد من ال images، نفس فكرة الحساب في github فبداخله العديد من ال repositories.

أيضاً لاحظ أن هناك العديد من ال Layers عند تحميل هذه ال Image فهي كبيرة من 1 قيقا بايت. واخيراً بعد أنتهى التحميل اشتغلت مباشرة وظهر المخرج، ونستطيع فتح المتصفح الآن localhost:8090 وسوف تجد كامل التوثيق متواجد لديك offline.

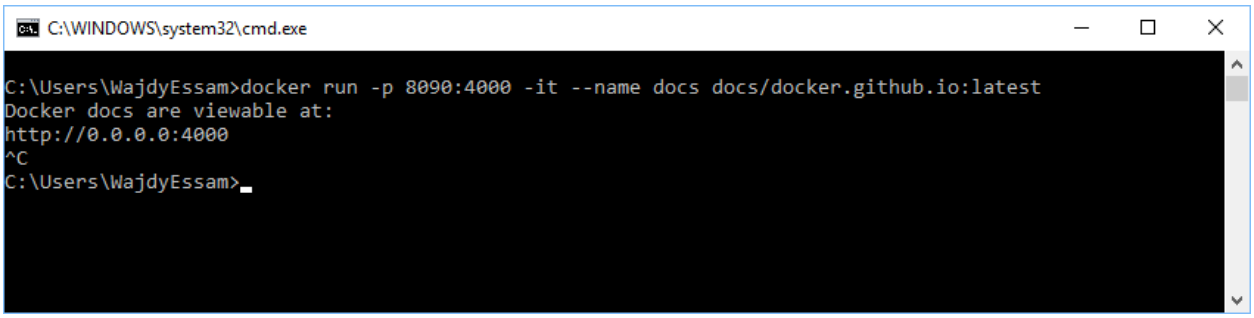


بعد فتحك للموقع وتصفحه سوف تجد في الكونسول العديد من المخرجات وهي logs من الويب سيرفر المضمن بداخل هذه ال container.

الآن لكي تخرج من هذه ال container وتعود الى سطر الأوامر يمكنك الضغط على CTLR+C، وبالطبع كما ذكرنا سابقاً فهذا لن ينهي ال container وانما ستظل تعمل. وسنقوم الآن بإيقاف هذه ال container من خلال الأمر stop كما ذكرنا سابقاً.




الآن سوف نأخذ معاملات جديدة لأي أمر وهو تشغيل ال container في وضع ال interactive mode بحيث عند الضغط على CTLR+C فسوف يتم ايقافها من العمل وذلك بتمرير المعامل it– للأمر، بالإضافة الى تسمية ال container بدلاً من الاسم العشوائي الذي يظهر تلقائياً من خلال المعامل name– كما يلي:



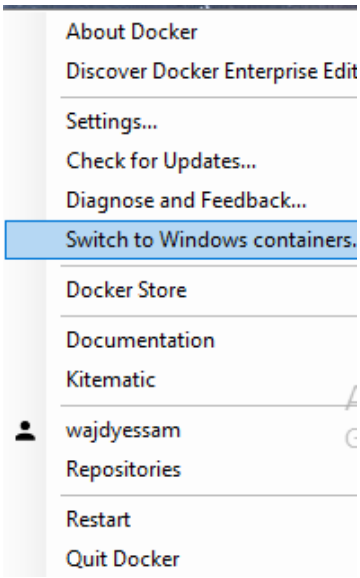

سوف تلاحظ أنه تسمية ال container بالاسم docs (ويمكنك كتابة الأمر docker ps أو docker ps –a وسوف تجد الاسم لهذا ال container)، بالإضافة الى تشغيلها في وضع ال it بحيث تستجيب لأي مدخل، وكما ترى في الصورة السابقة بعد تشغيل ال container والتأكد من عمل الموقع، قمنا بإدخال CTLR+C لإيقافها (وهي سوف توقف البرنامج الرئيسي الذي يعمل على ال container والذي يعني إيقاف ال container من العمل) ومباشرة توقفت عن العمل ولن تظهر في قائمة docker ps.

إذا قمت بتشغيل الأمر السابق مجدداً فسوف تحصل على رسالة خطأ فلن تستطيع إنشاء container اخر بنفس الاسم وعليك اما بحذف القديم، أو تسمية الجديد باسم اخر.

ولكي تخرج من ال interactive mode container يجب عليك أن تقوم بإدخال CTRL+PQ حتى تخرج بشكل صحيح بدون أن توقف ال container التي تعمل.

## تجربة ال Windows Containers

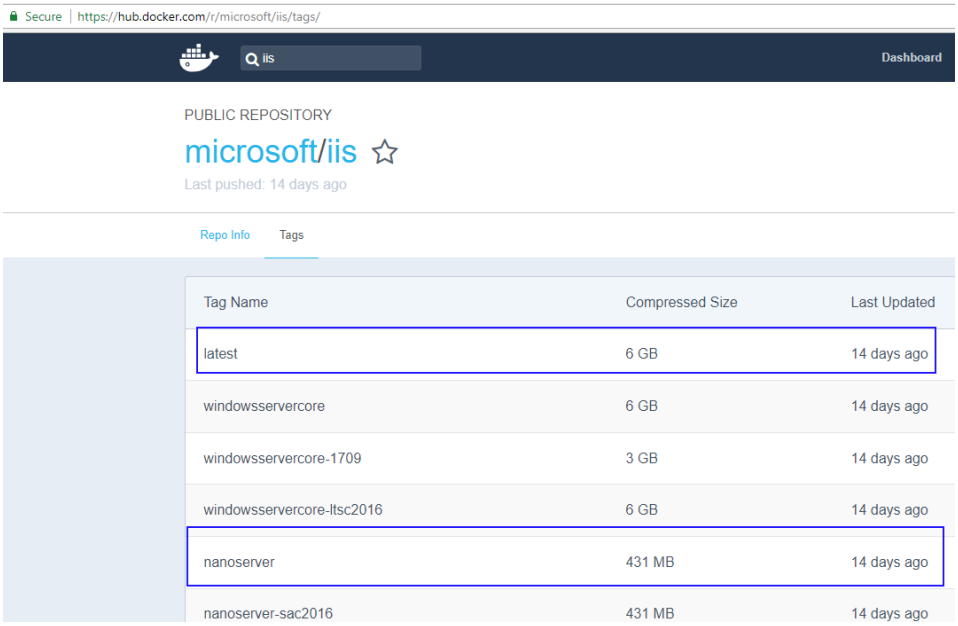
سوف نلقي نظرة الآن عن تشغيل ال windows container، ولكي نقوم بالتحويل من ال Linux Container الى Windows Container فيجب الضغط في الأسفل على زر دوكر بالزر الأيمن واختيار Switch to Windows Container (وسوف يتم تفعيل ال Windows Container Feature) المتواجد على ويندوز 10.



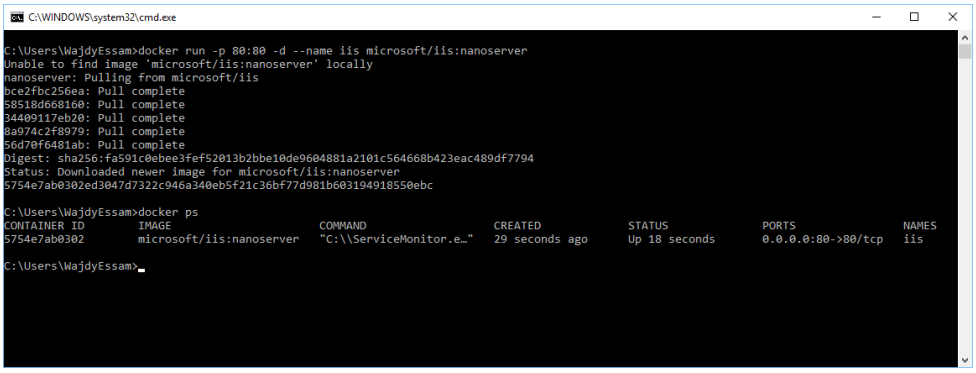

بعد الانتقال عند الضغط مجدداً على نفس القائمة سوف تجدها الآن أصبحت Switch to Linux Container وهذا يعني أنك تعمل على ال Windows Container ويمكنك تجربة الأوامر info, version الخاصة بدوكر لمعرفة تفاصيل ال client & server كما تطرقنا له في البدء.

عند كتابتك docker ps أو docker images (أو حتى docker ps –a) فلن تجد ال containers أو ال images التي كانت في ال Linux container حيث تم الانتقال ال windows container وهو مختلف عن ال Linux Containers.

وسوف نقوم بتشغيل IIS وهو الويب سيرفر المشهور في أنظمة الويندوز:



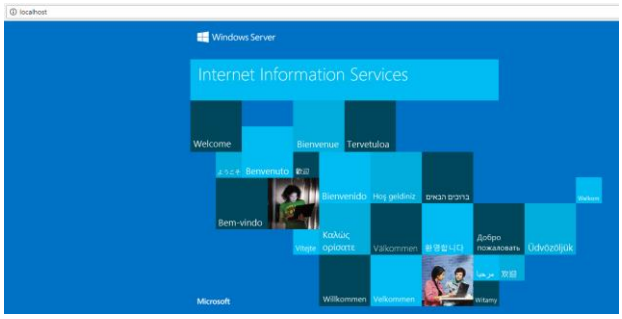

وسوف نجد هناك نسختين الأولى تعتمد على ال windowsservercore والثانية على ال nanoserver وفرق الحجم كبير، لذلك سوف نستخدم ال nanoserver وسوف نوضح الفرق بينهم فيما بعد.




لاحظ الأمر أعلاه، سوف تجد أنه تم تحديد البورت 80 على ال host machine والبورت 80 داخل ال container لأنه هو البورت الذي سيعمل على ال iis، وأيضاً ممرنا المعامل d– وهو اختصار لل detach بمعنى سيعمل ال container مباشرة ويرجع سطر الأوامر يستقبل الأوامر (كأننا نقوم بتشغيله بالطريقة العادية ومن ثم CTLR+C لكي نرجع لسطر الأوامر وفي نفس الوقت يكون ال container يعمل)، بمعنى كأنه يعمل في الخلفية الآن.

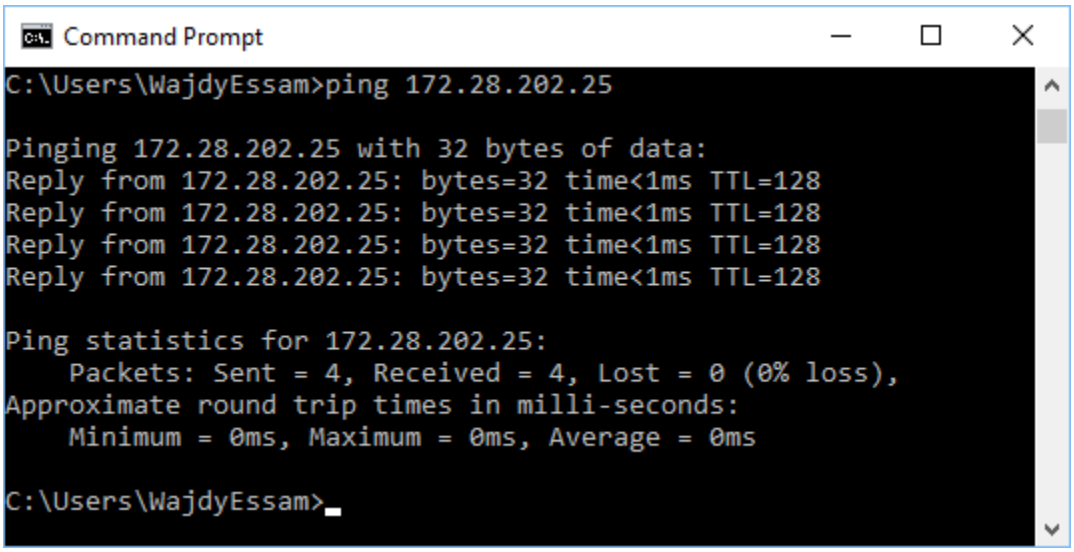
أيضاً قمنا بتحديد اسم له iis ومن ثم حددنا Tag حتى نقوم بتحميل ال nanoserver وليس latest. وبعد أن يتم التحميل ويعمل مباشرة سوف تلاحظ أنك الآن في سطر الأوامر ويمكن مشاهدة قائمة العمليات وسوف تجده على القائمة docker ps.

الآن يمكنك فتح المتصفح والدخول الى localhost وستجد ال IIS يعمل ولديك الصفحة الافتراضية فيه:

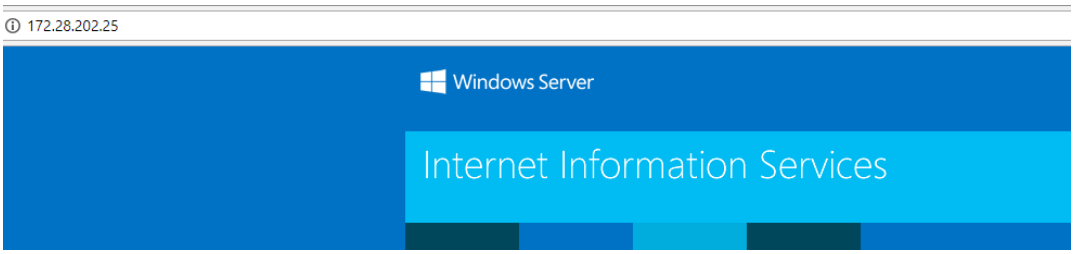



هناك طريقة أخرى لم نتحدث عنها وهي كيف يمكن فتح الويب سيرفر بدون استخدام ال port mapping وانما باستخدام عنوان ال container واستخدام ال port للويب سيرفر بداخله، ولكن عليك بمعرفة عنوان ال IP لل container، ويمكن من خلال الأمر docker inspect containerId حيث تمرر ال containerId وسوف تجد معلومات حول ال container التي تعمل وستجد ال IP لها، كما يلي:

يمكنك عمل ping لهذا ال IP وسوف يبدوا كأنه جهاز اخر وانما حقيقة هو Virtual Adapter وكل البرامج بداخل ال container سوف تتصل بهذا ال network adapter وهذا ال IP Address.




ويمكنك استخدام هذا ال IP مباشرة بدلاً من ال Port Mapping كما يلي:




وهذا يعتبر جزء من ال Isolation التي توفره الدوكر وهو ال Network Isolation وسوف نتحدث عنها ونرى أنواعها وبشكل مبسط فقط، وبعد أن ننهي من الأساسيات كاملة. سوف نرى ماذا يحدث خلف الكواليس.

## أنواع العزل Isolation في دوكر

البرامج التي تعمل في ال containers تكون معزولة وكأنها تعمل على جهاز لوحدها، وليس لديها علم عن البرامج ونظام الملفات المتواجد على النظام ال host (ونعنى بال host نظام التشغيل الأساسي الذي يعمل عليه الدوكر)، وهناك أنواع من ال isolation كما يلي:

- فصل وحجب قائمة العمليات Process
- فصل وحجب نظام الملفات File System
- فصل وحجب مسجل النظام Registry
- فصل وحجب الشبكات Networking
- فصل وحجب المستخدمين Users & Groups

وسوف نتأكد من كل هذه الأنواع ونرى كيف أن البرامج بداخل ال containers لن تستطيع مشاهدة هذه التفاصيل وانما لها بيانات أخرى، فيما يلي، وسوف نبدأ بمثال أخير نقوم بتحميله ومن ثم مشاهدة تفاصيل ال Image ونبذة عنها وبعدها ندخل في ال isolation

سوف نقوم بتحميل Microsoft/dotnet:nanoserver وهو ال DotNet Runtime ونقوم بتشغيلها مباشرة من خلال الأمر run كما يلي:

```
C:\Users\WajdyEssam>docker run microsoft/dotnet:nanoserver
Unable to find image 'microsoft/dotnet:nanoserver' locally
nanoserver: Pulling from microsoft/dotnet
bce2fbc256ea: Already exists
58518d668160: Already exists
871dd90799b2: Pull complete
c60cfd8021ee: Pull complete
0400f3f2f26a: Pull complete
edad1b2b7da1: Pull complete
021722d9d5b4: Pull complete
29b86cb579ec: Pull complete
627b5e4f721c: Pull complete
Digest: sha256:18cac41e449fb09ae65f0fe363e45d92c8f8847f9f4dc670cfad40ca72b7c3ef
Status: Downloaded newer image for microsoft/dotnet:nanoserver
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\>
```

بعد أن تحميل وتشغيل ال dotnet فمباشرة ظهر C:\ ومن ثم توقفت عن العمل، ولو نظرت لقائمة العمليات docker ps فلن تجدها، وسوف تجدها متوقفة في ال docker ps –a.

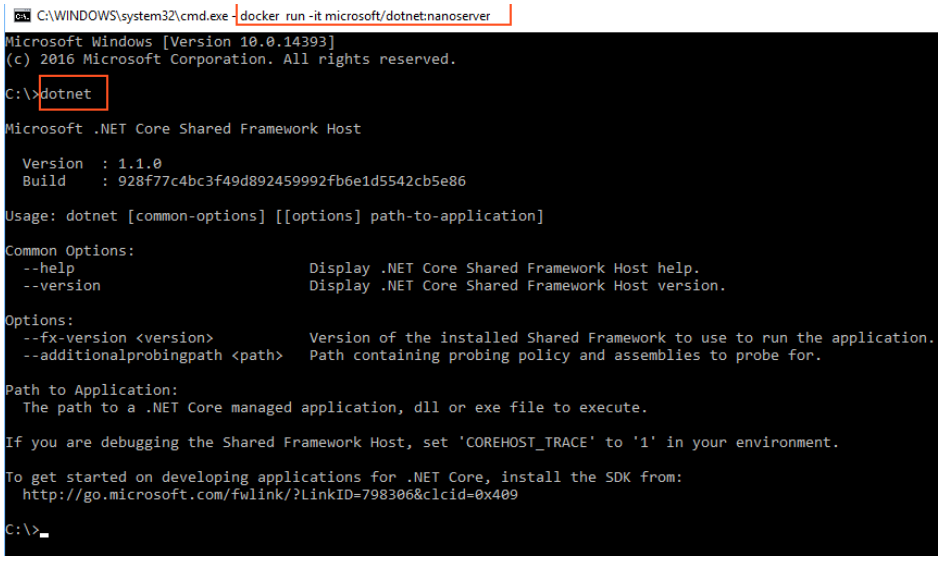
```
C:\Users\WajdyEssam>docker ps
CONTAINER ID        IMAGE                     COMMAND                  CREATED             STATUS              PORTS               NAMES
5754e7ab0302        microsoft/iis:nanoserver  "C:\\ServiceMonitor.e…"  13 hours ago        Up 13 hours         0.0.0.0:80->80/tcp  iis

C:\Users\WajdyEssam>docker ps -a
CONTAINER ID        IMAGE                     COMMAND                  CREATED             STATUS                      PORTS               NAMES
8493cd778039        microsoft/dotnet:nanoserver  "c:\\windows\\system32…"  5 minutes ago       Exited (0) 5 minutes ago                        pedantic_chandrasekhar
5754e7ab0302        microsoft/iis:nanoserver  "C:\\ServiceMonitor.e…"  13 hours ago        Up 13 hours         0.0.0.0:80->80/tcp  iis
```

ولاحظ أن أمر التشغيل Command على ما يبدوا هو لتشغيل سطر الأوامر ولكن حصل truncate في المخرج، لذلك يمكنك استخدام المعامل no-trunc– حتى يظهر كامل المخرجات بدون اختصار:

```
C:\WINDOWS\system32\cmd.exe
C:\Users\WajdyEssam>docker ps -a --no-trunc
CONTAINER ID                                                       IMAGE                       COMMAND                                  CREATED             STATUS
    PORTS               NAMES
8493cd7780395a96a86f9258cb1b67d24c7b258f1d7cf72d378791b884574023   microsoft/dotnet:nanoserver  "c:\\windows\\system32\\cmd.exe"         7 minutes ago       Exited (0) 7 minutes ag
o                       pedantic_chandrasekhar
5754e7ab0302ed3047d7322c946a340eb5f21c36bf77d981b603194918550ebc   microsoft/iis:nanoserver    "C:\\ServiceMonitor.exe w3svc"           13 hours ago        Up 13 hours
    0.0.0.0:80->80/tcp  iis
```

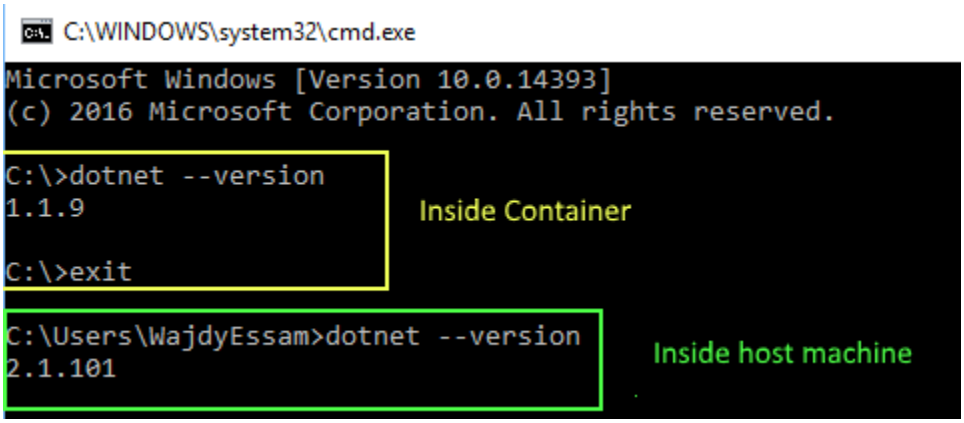
وكما يتبين فإن ال command line قد عملت مباشرة مع ال containers ومن ثم انتهت بعدها، لذلك يمكن أن نقوم بعمل attach لسطر الأوامر من خلال المعامل it– الذي سبق لنا تجربته حتى نستطيع التعامل مع سطر الأوامر ولا يتم انهاء ال container مباشرة. وبعد أن تقوم بكتابة سوف يتم مسح سطر الأوامر وكأنك تنتقل الى سطر أوامر اخر بداخل ال container.

الصورة التالية توضح الأمر، وطالما هذه container بها dotnet فيمكن أن تستعلم عن ال framework بها، كما يلي:




حتى هذه اللحظة ما زلنا بداخل سطر الأوامر بداخل ال container ويمكن العودة للسطر الأوامر الخاص بالهوست عن طريق كتابة أمر الخروج exit وسوف تعود لل host.

يمكنك أن تجرب وتقوم بكتابة نفس الأمر dotnet على سطر الأوامر في الهوست، وإذا لم تكن لديك على جهازك الأساسي فسوف تحصل على رسالة خطأ، وهذا يعني أن البرنامج dotnet فقط يعمل بداخل ال container، أما إذا كانت ال dotnet متواجدة في ال host فسوف تعمل أيضاً. لكنها مفصولة تماماً عن التي متواجدة بداخل ال container وقد تكون النسخ مختلفة أيضاً، الصورة التالية تبين ذلك:



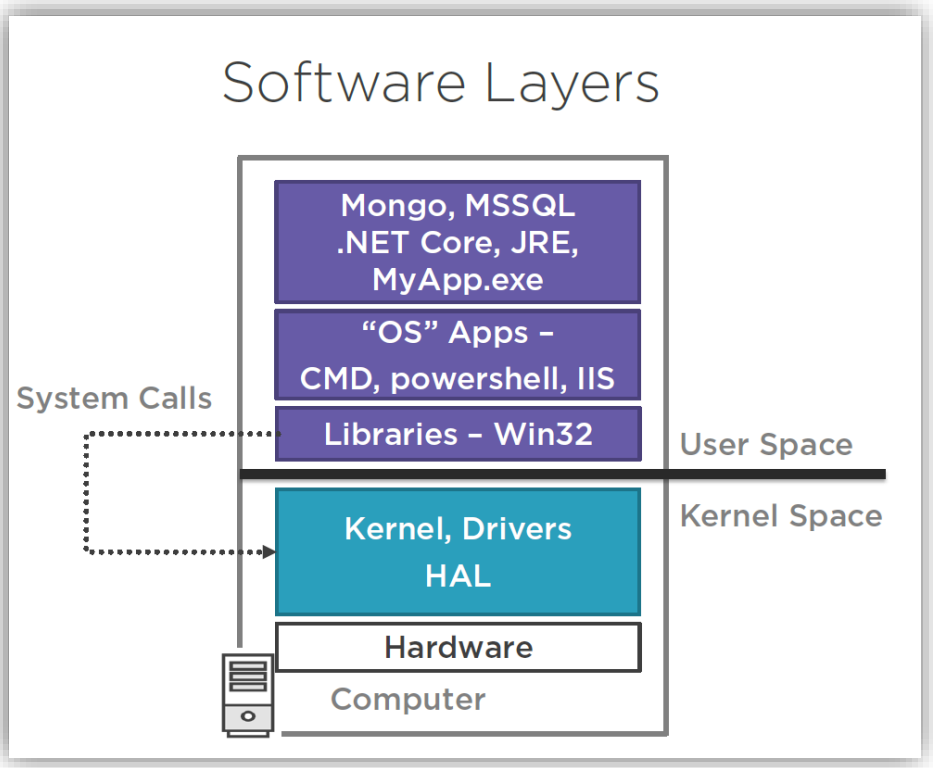

وبهذا الشكل نستطيع تشغيل أي برنامج بداخل ال container سواءً كان البرنامج المحدد أن يعمل مباشرة مع ال container أو أي برنامج اخر نريده يمكن أن نقوم بتشغيله، كما في الفصل الأول في المقدمة، حيث قمنا بفتح ال mongo client المتواجد داخل نفس ال container من خلال الأمر exec.

ولكي نعرف كيف حصل هذا، يجب علينا أن نلقى نظرة الآن حول هذه ال image وبنيتها.

## مكونات ال Image

ذكرنا سابقاً أن ال Image هي القالب للبرنامج، والذي يتم تشغيله في Container، وهي التي تحتوي على ملفات البرنامج والاعتماديات الخاصة بها، ولكنها تحتوي أيضاً على شيء أضافي وهو مكتبات نظام التشغيل.

الصورة التالية تعرض كيف تعمل البرمجيات في نظام التشغيل بشكل عام، حيث بداخل جهاز الحاسب، لدينا القطع المادية مثل المعالج، الذاكرة، القرص الصلب وغيرها، وبعد أن يعمل نظام التشغيل سوف يتم تحميل الكيرنل Kernel والذي يتعامل مع ال Hardware بشكل مباشر، بغض النظر عن نظام التشغيل ويندوز أو لينوكس فهناك كيرنل يعمل بنفس المفهوم، ومن فوق الكيرنل هناك API مثل Win32 أو POSIX والتي يوفرها النظام لكل البرامج فيه، ومن فوق طبقة ال API فهناك طبقة برمجيات النظام التشغيل التي يأتي بها، مثلاً cmd وغيرها، واخيراً هناك طبقة برمجيات المستخدم التي يقوم بتنزيلها مثلاً جافا أو بايثون أي دوت نت.




سوف تجد أن الطبقات مقسمة الى قسمين الأول ما يعرف بال Kernel Space والثانية هي ال User Space وهي تضم كافة البرمجيات ومكتبات نظام التشغيل الضرورية والتي تتعامل مع الكيرنل، وهذا التصميم شائع في أنظمة التشغيل حيث يوفر حماية لل Hardware من الوصول المباشرة بواسطة ال User Space ويجب أن يمر الطلب عبر الكيرنل والذي يحدد ويضبط العملية من خلال ال System Calls.

وهذا يعني مثلاً برنامج الجافا سوف يطلب مكتبات نظام التشغيل Win32 والتي تقوم بعمل System Call الى الكيرنل لتنفيذ المهمة.

الذي يهم في شرح هذه العملية هو أنه في السابق عندما نقوم بنشر البرمجيات فنركز على نشر الطبقة العليا "برمجيات المستخدم" فقط مثلاً جافا أو Mongo وغيرها، ونقوم بتحميل ال Installer وهذا ال Installer يعتمد على أشياء يجب أن تكون موجودة مثلاً مكتبات نظام التشغيل والا فلن يعمل بشكل صحيح.

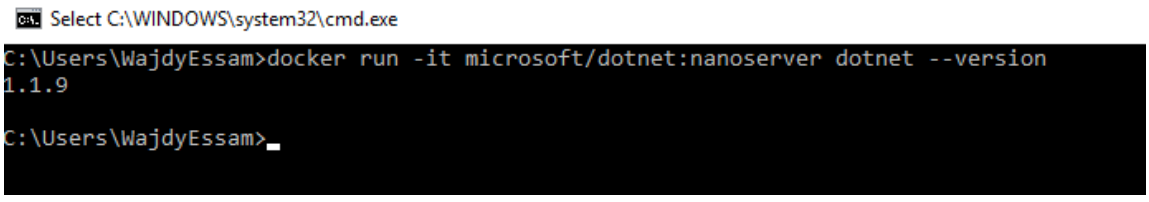
ولكي يتم تفادي مشكلة أن المكتبات غير موجودة فال containers & images تقوم بتغير الطريقة التقليدية، فبدلاً من نشر MonogoDB لوحده فقط، فسيتم نشر كل ما يتواجد في ال User Space وبالتالي يعني البرمجيات + برامج النظام + مكتبات النظام كلها تكون متواجدة في ال image.


وهكذا تكون ال Image مقسمها لعدة طبقات، منها طبقة برمجيات ومكتبات النظام، وبعدها طبقة البرمجيات والاعتماديات الخاصة بها، ويتم رفع ال image على ال Docker Hub.

وهنا يجب الإشارة الى بعض النقاط المهمة:

- بما أن ال Image بها برامج النظام، فيمكن أن يتم تشغيل البرنامج الأساسي في ال Image عند تشغيل ال Container، ويمكن أن يتم تشغيل أي من برامج النظام المتواجدة في ال Image أيضاً.
- إذا تم تحميل مثلاً java Image لل windows فسوف تكون المكتبات والبرمجيات في الطبقة الأساسية Base Layer هي للويندوز، فعند تحميل ال image سوف تلاحظ أنها مكونة من طبقات كما شاهدنا في الأمثلة السابقة، الآن بعد أن ينتهي التحميل، واردنا تحميل image أخرى تستخدم نفس ال base layer فلن يتم تحميلها مجدداً لأنها متواجدة وسيتم تحميل فقط طبقة البرمجيات ويتم استخدام الطبقة المتواجدة، وهذا لأن كل الطبقات التي يتم تحميلها لا تتغير فهي للقراءة فقط Read only. وبالتالي لن تحتاج الى تحميل كل ال User Space في كل مرة كاملة في حال كان يتواجد لديك Layers محملة من قبل في Images تستخدم نفس ال layers.
- عند تشغيل أكثر من Containers فلكل منهم كافة ال User Space ولكنهم في النهاية يشتركوا في نفس الكيرنل. وهذا ما يجعل ال Container أكثر كفاءة من ال VM حيث ال VM تستخدم كيرنل لكل VM يعمل بينما هنا كيرنل واحد لكل ال Containers.

الآن بعد أن تعرفنا على أنه يمكن تشغيل أي من البرامج المتواجدة في ال image فسوف نستخدم نفس ال dotnet image ونقوم بتشغيل ال dotnet مباشرة بدلاً من ال cmd:



طبعاً بعد أن ينتهي البرنامج سوف تتوقف ال Container عن العمل. وبنفس المفهوم يمكن تمرير PowerShell بدلاً من ال dotnet للدخول على ال PowerShell بداخل ال container.

الآن حان الوقت لكي نجرب فكرة ال Isolation ونرى هل هناك عزل أم لا.

## عزل العمليات Process Isolation

ويمكن اختبارها بسهولة، حيث يمكن الدخول لأي Container وتنفيذ دالة استخراج كل العمليات التي تعمل، وأيضاً تطبق نفس الأمر على ال host وسوف ترى الفرق في العمليات وأن لكل منهم عملياته الخاصة كأنه يعمل في جهاز منفصل.

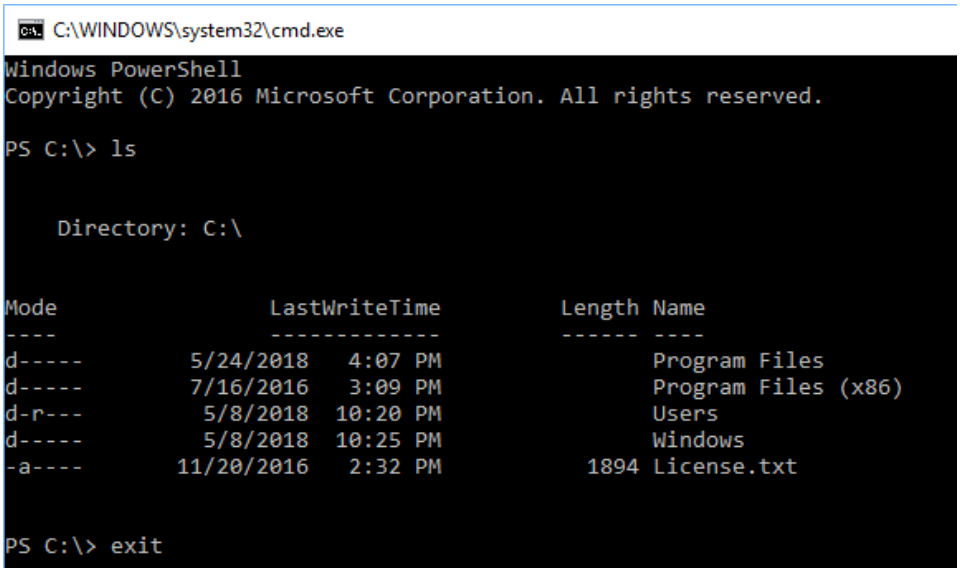
```
C:\Users\WajdyEssam>docker run -it microsoft/dotnet:nanoserver powershell


Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\> Get-Process

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)      Id  SI ProcessName
-------  ------    -----      -----     ------      --  -- -----------
      0       5     1032       4476       0.03    1264   1 CExecSvc
      0       5      692       1900       0.17     928   1 csrss
      0       0        0          4                  0   0 Idle
      0      15     2696       8976       0.06    1004   1 lsass
      0      37    43144      66532       1.78    1456   1 powershell
      0      10     1968       5276       0.05     980   1 services
      0       3      324       1136       0.08     908   0 smss
      0      14     5444       9868       0.08       8   1 svchost
      0       8     1880       5840       0.00     736   1 svchost
      0      13     1648       6008       0.02     784   1 svchost
      0      11     2800       9068       0.02    1056   1 svchost
      0       7     1388       4588       0.02    1096   1 svchost
      0      36     5456      12944       0.20    1124   1 svchost
      0      11     3048       9236       0.08    1240   1 svchost
      0       0      128        136       0.70       4   0 System
      0       8     1248       4160       0.03     956   1 wininit


PS C:\> exit
```

الآن إذا قمت بالدخول على ال PowerShell في ال host وطبقت نفس الأمر Get-Process سوف تجد قائمة كبيرة مختلفة من العمليات، وبالطبع يمكن أن تجرب نفس الامر في لينوكس من خلال الامر ps وشاهد الفرق بين العمليات، ولكن الآن سنركز على الويندوز لمشاهدة طرق العزل، ولكن نفس الفكرة يمكن تطبيقها بأوامر اللينوكس بسهولة.

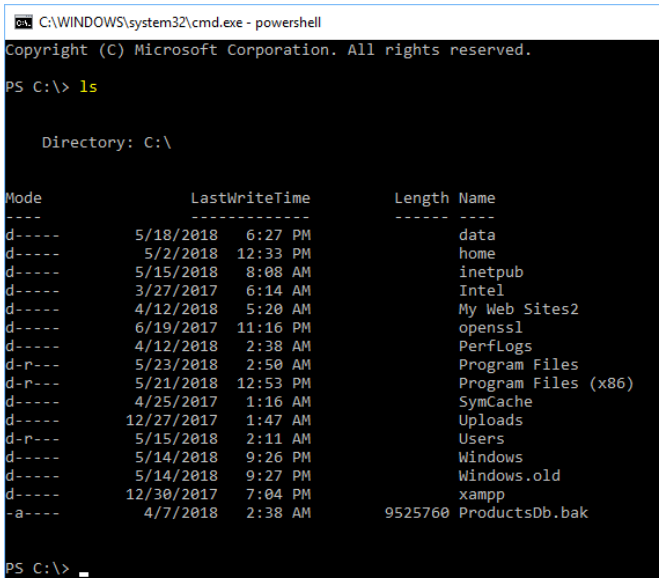
```
C:\WINDOWS\system32\cmd.exe - powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\WajdyEssam> Get-Process

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)      Id  SI ProcessName
-------  ------    -----      -----     ------      --  -- -----------
    426      20     9340      23328       0.19   24892   1 AcroRd32
    511      52    97536     107112       2.75   25988   1 AcroRd32
    458      40    60380      30852      18.58   11248   1 Adobe CEF Helper
    451      34    28736      33828     658.52   14580   1 Adobe CEF Helper
    777      62   112344      49768     121.97   11020   1 Adobe Desktop Servic
    248      17     4508      10308      83.80   13812   1 AdobeIPCBroker
    203      14     2128       7048               3904   0 AdobeUpdateService
    303      17     4176      13320               3836   0 AGSService
    619      36    32884      26692       1.25   16336   1 ApplicationFrameHost
    315      16     3092      13460               3720   0 armsvc
    162      10     1952       7712       0.03    7936   1 browser_broker
    502      26    15972      36348       0.33   19520   1 Calculator
     50       4      528       1976       0.00   14404   1 CCXProcess
    323      45   109632      95320       8.53     992   1 chrome
    330      47   115000      46020       5.14    3312   1 chrome
    333      47   119580      47784      12.17    4296   1 chrome
    312      26    36852      43184      21.22    5352   1 chrome
    336      46   105136      44392       5.05    7356   1 chrome
    284      21    21864      18016       0.97    7492   1 chrome
    298      25    34076      46332       0.25    9088   1 chrome
    285      22    23812      21384       1.16   10636   1 chrome
    362      50   131676     112644      13.88   10832   1 chrome
    292     102   444932     217044     300.83   10932   1 chrome
```

## عزل نظام الملفات File System Isolation

سوف نقوم الآن بعرض نظام الملفات بداخل ال Container ومشاهدة نظام الملفات على ال host وسوف ترى الفرق، وأن لكل منهم tree مختلفة من الملفات. الصورة التالية تعرض الملفات بداخل على القرص سي بداخل ال Container:

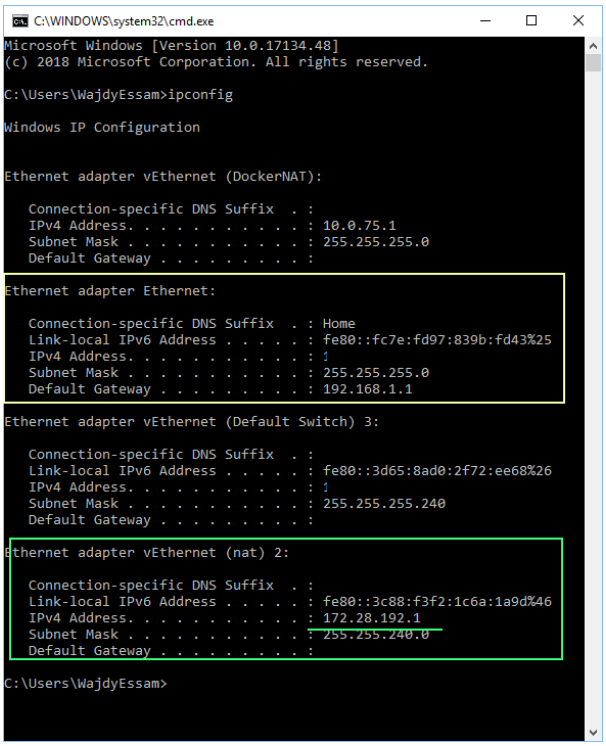



بينما على ال host فالملفات مختلفة للغاية:

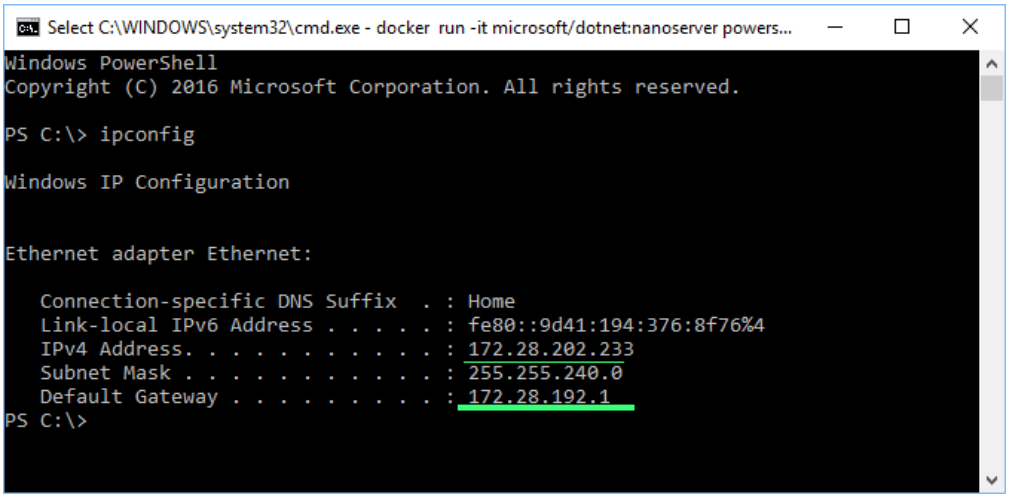



## عزل الشبكات Network Isolation

بنفس الفكرة فال Network Adapter سوف تكون معزولة هي الأخرى، وسوف نقوم الآن باستخراج معلومات ال Adapters المتواجد باستخدام الأمر ipconfig




وسنقوم بتطبيق الامر بداخل ال Container




وبالتالي داخل ال container سوف تكون على شبكة أخرى كأنه بالضبط يعمل على جهاز آخر.

وكما تلاحظ سوف تجد أن هناك Virtual Adapter على الهوست تكون هي ال Getaway المستخدمة في ال Container حتى يتم التخاطب بينهم من خلال هذه الشبكة.

وهذا يعني أستطيع تشغيل ويب سيرفر على ال container في البورت 80، حتى لو كان البورت 80 مستخدم في ال host حيث لكل منهم شبكة منفصلة عن الأخرى.

وهكذا سوف نجد أن العزل مطبق على كل هذه الأمور (العمليات، نظام الملفات، الشبكات) وأكثر من ذلك حيث يتم تطبيقه على متغيرات النظام Environment Variable وعلى مسجل النظام Registry وعلى المستخدمين في النظام أيضاً، وتستطيع التأكد منهم بنفس الطرق أعلاه، بعرض النتيجة في ال host وال container ومشاهدة الفرق بينهم.

أخر نقطة بخصوص ال Windows Container فهي تطبق داخلياً أما بال Windows Server Container والتي تستخدم للويندوز سيرفر وسيتم التشغيل مباشرة على النظام، والطريقة الثانية وهي ال Hyper-V Containers وهي بالاعتماد على Hyper-V (وهذا ما يتم في ويندوز 10) لتشغيل ال containers. وستجد أن الثانية بالرغم من أنها تستخدم VM (لكنه أسرع بكثير من ال VM العادي) الا أنها أيضاً قد توفر حماية أفضل بسبب ال VM والذي يتم العزل كلياً.

## خلاصة

وصلنا لنهاية الفصل الثاني، وكما نشاهد أن docker & container تساعدك على تشغيل البرمجيات وجعل حياتك أسهل عما كانت عليه سابقاً، الجدول التالي يعرض كيف كنا نعمل بالطريقة التقليدية سابقاً، وما هو طريقة دوكر لها، وما هو الأمر في دوكر لتطبيق هذا الأمر.

| Traditional Software | Docker Equivalent | Docker Command |
|---|---|---|
| Find software | Docker Hub | |
| Download software, i.e. a zip file or MSI | Pull an image | docker pull |
| Install software | Create a container from an image | docker create |
| Start software | Run the container | docker start |
| Stop software | Stop the container | docker stop |
| Uninstall software | Remove the container | docker rm |
| Not Possible | Do all of this with one command! | docker run |

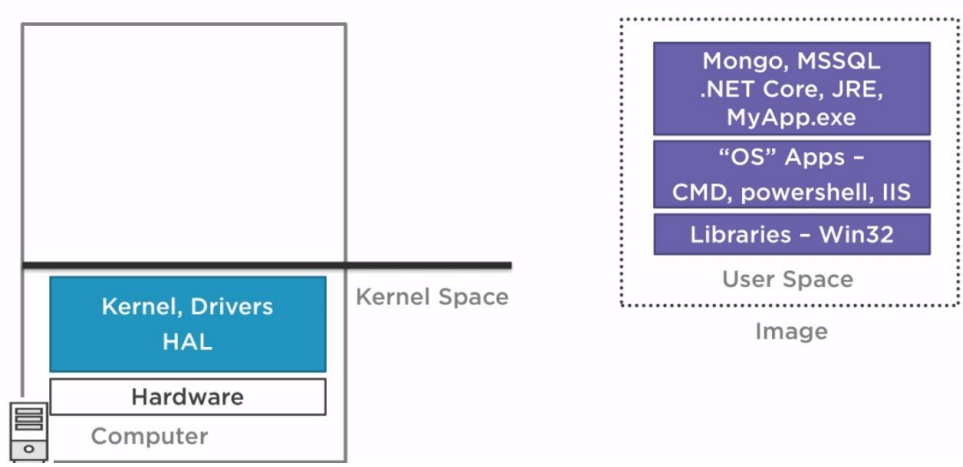
وسوف تجد أن الأمر الخاص بتحميل وتشغيل ال container في أن واحد لا يوجد له بديل بالطرق التقليدية وهذا ما يميز دوكر ويجعلها المستقبل للمطورين لتشغيل البرمجيات وتعلمها أيضاً بسهولة.

# الفصل الثالث: مشاركة المجلدات مع ال Containers

أحد المميزات في دوكر Docker هي سهولة تشغيل البرامج بدون خطوات وتعقيدات كثيرة في التنزيل، وهذا الأمر سوف نستخدمه كثيراً لتحميل البرامج التي نحتاجها لاستضافة مواقعنا مثل nginx أو IIS وأيضاً لتشغيل أي Command Line Application أخر.

في هذا الفصل سوف نقوم بتشغيل عدة برامج من سطر الأوامر Command Line ومنها نتعرف على بعض الأمور في دوكر وال Containers، خصوصاً ما يجري بداخل ال Container Image، وكيف يمكن مشاركة مجلدات أو ملفات بين نظام التشغيل الأساسي مع ال Container ولماذا نحتاج لذلك.

ذكرنا في المواضيع السابقة أن كل ال User Space تكون بداخل ال Container Image سواءً كان نظام الملفات أو المكتبات الخاصة بنظام التشغيل وأيضاً بعض البرامج الأخرى، كلها تكون بداخل وعاء ال Image، كما في الصورة التالية:




سوف نقوم الآن بالتأكد من ذلك وسحب image ومشاهدة محتواها، وسوف نقوم أولاً بمعرفة ال Images التي لدينا على الجهاز، وذلك من خلال الأمر docker images:

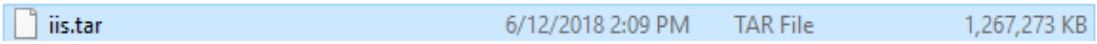
```
C:\WINDOWS\system32\cmd.exe

C:\Users\WajdyEssam>docker images
REPOSITORY                      TAG                 IMAGE ID            CREATED             SIZE
microsoft/dotnet                nanoserver          8f13c75d6f1d        4 weeks ago         1.84GB
microsoft/iis                   nanoserver          e49ce207d81d        4 weeks ago         1.24GB
microsoft/windowsservercore     latest              4dba31379dad        4 weeks ago         10.6GB
```

وسوف نعمل على ال iis image فهي الأقل حجماً، وإذا لم تكن لديك فيمكن تحميلها من خلال الأمر docker pull microsoft/iis:nanoserver وسيتم تحميلها.

الآن سوف نستخدم أمر تصدير ال image الى ملف tar. مضغوط حتى نستطيع فكه ومشاهدة محتوياته، وذلك من خلال الأمر docker save image كما يلي:

```
C:\WINDOWS\system32\cmd.exe

C:\Users\WajdyEssam>docker save microsoft/iis:nanoserver -o iis.tar

C:\Users\WajdyEssam>_
```

بعد الانتهاء من التصدير سوف تجد الملف المضغوط الذي قمت بتصديره في المسار المحدد، كما يلي:



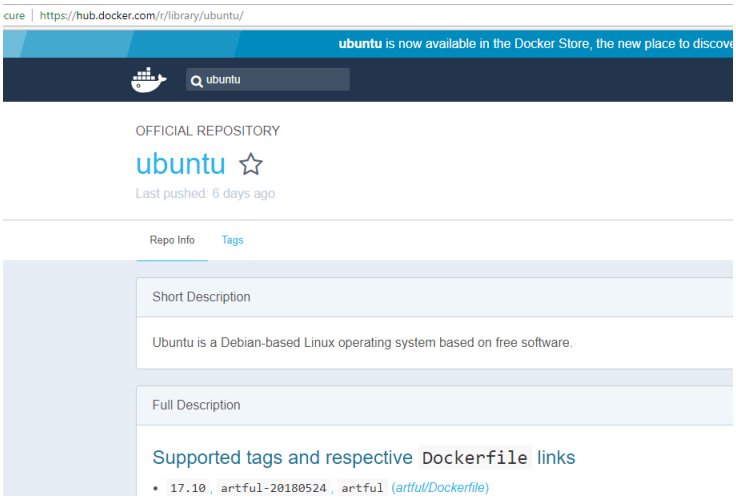

سوف نحتاج الى فك الضغط من ذلك الملف، وسوف تحتاج لبرنامج لفك الضغط، ولأن نظام الويندوز لا يدعم ذلك الملف، فيمكنك تحميل أي برنامج للقيام بفك الملف مثلاً WinRAR أو 7zip أو أن تستفيد من دوكر وتقوم بتشغيل Linux Container وظيفته فك الملف من الضغط، وهذا ما سنفعله حيث أن فكرة تشغيل برامج نظام تشغيل آخر لكي تقوم ببعض العمليات مثل فك الضغط من خلال أمر واحد وكان البرنامج يعمل على جهازك الحالي هي من الأفكار الجميلة في دوكر.

كل أنظمة تشغيل لينوكس تأتي ب tar command line لضغط وفك الملفات وسوف نستخدم هذا البرنامج بداخل ال Linux container للقيام بهذا الأمر.

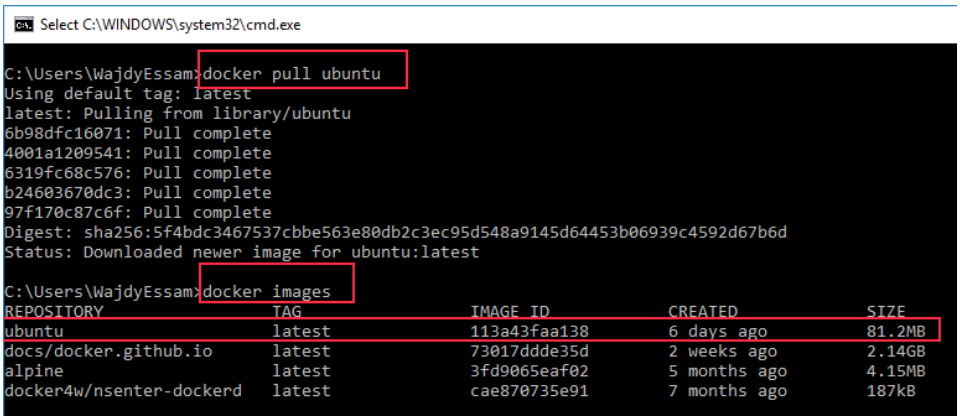
لذلك سنقوم أولاً بالتحويل الى ال Linux Containers وذلك بالضغط على ايقونة دوكر بالأسفل ومن ثم اختيار Switch to Linux Containers، وبعد التحويل تستطيع تجربة مشاهدة ال docker images وسترى لديك ال images الخاصة بنظام اللينوكس.

```
C:\WINDOWS\system32\cmd.exe

C:\Users\WajdyEssam>docker images
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
docs/docker.github.io   latest              73017ddde35d        2 weeks ago         2.14GB

C:\Users\WajdyEssam>_
```

سوف نحتاج الى image بها برامج لينوكس ويمكن تحميل أي من Linux images وسوف نختار Ubuntu وسوف نقوم بتحميل ال image من المصدر الأصلي لها official حتى نضمن أنها من المصدر الأصلي ولا يوجد بها أي تعديلات.



سنقوم بتحميلها من خلال الأمر docker pull ubuntu ومن ثم التأكد من أن ال image أصبحت موجودة لدينا:




الآن سنقوم بتشغيل ال container من هذه ال image وسوف نقوم بفتح برنامج ال shell وهو سطر الأوامر في لينوكس، وسوف نقوم بكتابة امر فك الضغط من الملف حتى تخرج الملفات التي نريد فكها.

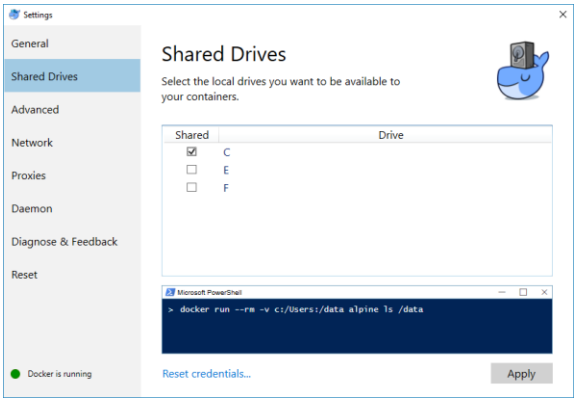
ولكن كما ذكرنا سابقاً أن ال container يعطي انعزال isolation للعمليات ولنظام الملفات File System، فهذا يعني أنه من داخل ال container فلن تستطيع الوصول لأي ملف على ال File System لأنها معزولة عنها، ولذلك يبقى السؤال كيف يمكن الوصول للملف الذي تريد فك ضغطه اذاً؟

## استخدام ال Volumes لمشاركة جزء من نظام الملفات

لحل هذه المشكلة سوف نقوم بعمل mount لجزء من نظام الملفات النظام الأساسي، سواءً كان مجلدا كاملاً بكل محتواه، أو حتى ملف واحد فقط، وهذا الجزء الذي تم عمل mount له سيكون متاح لل container للقراءة والكتابة فيه، يمكنك تخيلها كأنه أدخلت USB للجهاز فسوف تجد قرص جديد على جهازك، فبالنسبة لل container الملف الذي تم عمل mount له يكون بمثابة قرص أو جزء جديد فيه. هذه العملية تسمى في دوكر بال Volumes.

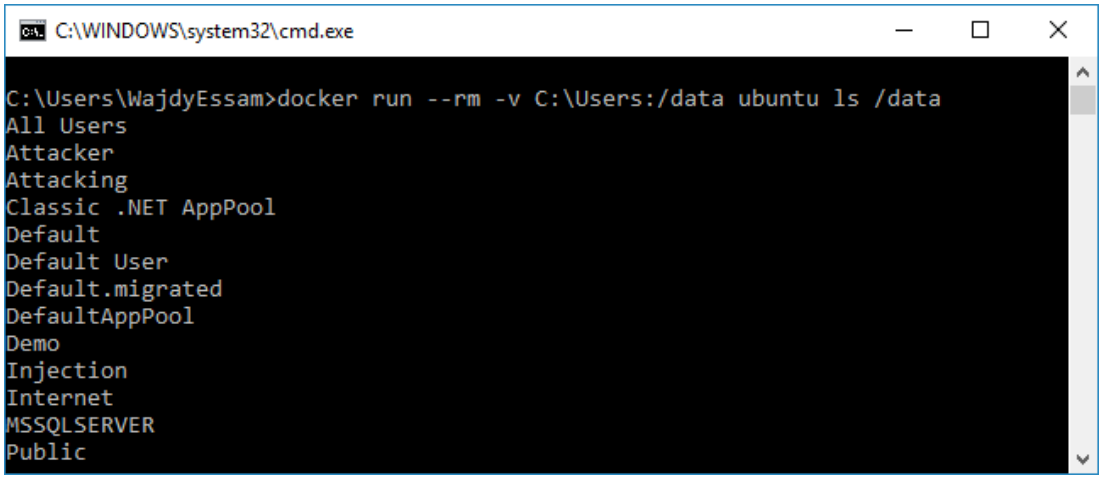
ولأننا الآن في Linux Containers وهذا يعني كل ال Containers سوف تعمل على ال Moby VM بداخل ال Hyper-V، فيجب أن نقوم بعمل مشاركة من نظام ويندوز الى لينوكس أولاً، حتى بعد ذلك نستطيع تحديد الأجزاء التي نريدها أن نشاركها لل Containers.

بالزر الأيمن من دوكر بالأسفل اختر اعدادات Settings ومن ثم Shared Drives واختر القرص الذي تريد مشاركته مع نظام التشغيل لينوكس واضغط على Apply وسوف يطلب منك إدخال كلمة مرور النظام للتأكيد.



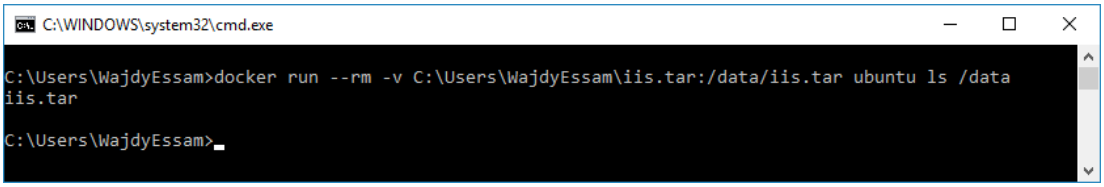
الآن أصبح القرص C متاحاً لنظام التشغيل لينوكس، ولكن ما زالت ال Containers لن تستطيع الوصول له، ويجب أن نستخدم ال volume الآن لتحديد الجزء من الملفات التي نرغب بالوصول لها.

سوف نقوم بتجربة عدة أوامر لكي نفهم تلك الفكرة، ونبدأ بالأمر التالي وهو أمر تشغيل عادي docker run، ولكن ارسلنا له معامل –rm وهي تعني قم بحذف ال container بعد أن تنتهي وليس لها علاقة بال volumes، بعد ذلك يأتي المعامل v– وبها معاملين الأول هو جزء نظام الملفات الذي نريد مشاركته مع ال container، وبعدها : ومن ثم المسار الذي ستظهر فيه الملفات في ال containers، ولأن ال Container هي لينوكس فيجب أن تتعامل بمسارات اللينوكس حيث تبدأ ب / وهي الروت، بعكس الويندوز التي تبدأ بالقرص أولاً. بعد ذلك نحدد اسم ال image التي سنشغل ال container منها، وبعدها البرنامج بداخلها وهو ls الذي يعرض الملفات، وهو يستقبل معامل أيضاً وتم تمرير /data.




الآن ستعمل ال containers وتم مشاركة ال C:/Users لها على المسار /data وبالتالي تستطيع مشاهدتها، وبالفعل تم تشغيل برنامج ls بذلك المسار وتم طباعة كل محتوى ذلك المجلد. وهو نفس الملفات إذا قمت بفتحها بال Windows Explorer أو من خلال ال dir وسوف تجد نفس الملفات بالضبط، وهذا يعني أن ال mounting للقرص يعمل والآن ال container تتشارك بالملفات.

الآن بدلاً من عمل mount لكل الملفات هذه، سوف نقوم بعملها للملف الواحد الذي نريده، ونقوم بتحديده وبالتالي ال container لن ترى الا هذا الملف فقط، كما يلي:




وهذا أفضل من ناحية الأمان، حيث نعطي ال Containers البيانات التي تحتاجها فقط في أداء عملها لا أكثر من ذلك.

أصبحنا جاهزين لتشغيل ال sh بنفس الطريقة السابقة بدلاً من العرض، وسوف نستخدم وضع ال it– حتى نستطيع ادخال الأوامر، وأول امر هو tar –tf وهو لمشاهدة محتوى الملف المضغوط بدون فك:

```
C:\WINDOWS\system32\cmd.exe - docker  run --rm -it -v C:\Users\WajdyEssam\iis.tar:/data/iis.tar ubuntu sh

C:\Users\WajdyEssam>docker run --rm -it -v C:\Users\WajdyEssam\iis.tar:/data/iis.tar ubuntu sh
# tar -tf /data/iis.tar
23fc664f6b78b0b6652bbb83e52a7286d9d33c7765e6f13a2d251ce7c343ea09/
23fc664f6b78b0b6652bbb83e52a7286d9d33c7765e6f13a2d251ce7c343ea09/VERSION
23fc664f6b78b0b6652bbb83e52a7286d9d33c7765e6f13a2d251ce7c343ea09/json
23fc664f6b78b0b6652bbb83e52a7286d9d33c7765e6f13a2d251ce7c343ea09/layer.tar
491dd052ce3c5236841a1d870bb557d1f5d5217966a853e9aca7995506f9338e/
491dd052ce3c5236841a1d870bb557d1f5d5217966a853e9aca7995506f9338e/VERSION
491dd052ce3c5236841a1d870bb557d1f5d5217966a853e9aca7995506f9338e/json
491dd052ce3c5236841a1d870bb557d1f5d5217966a853e9aca7995506f9338e/layer.tar
5862d78dbf431ae1056a44441f706bbb98d3d315fa1807c1cf2b499b0b185609/
5862d78dbf431ae1056a44441f706bbb98d3d315fa1807c1cf2b499b0b185609/VERSION
5862d78dbf431ae1056a44441f706bbb98d3d315fa1807c1cf2b499b0b185609/json
5862d78dbf431ae1056a44441f706bbb98d3d315fa1807c1cf2b499b0b185609/layer.tar
5c8322e82d15118d455379e19589a17066496ab6c1c5a4cd0dbbdd775c563831/
5c8322e82d15118d455379e19589a17066496ab6c1c5a4cd0dbbdd775c563831/VERSION
5c8322e82d15118d455379e19589a17066496ab6c1c5a4cd0dbbdd775c563831/json
5c8322e82d15118d455379e19589a17066496ab6c1c5a4cd0dbbdd775c563831/layer.tar
d2c0a78744dc8dbe7d13b8dac1e2f738e69ab00de79e6d937ec38e0bca2cd0bf/
d2c0a78744dc8dbe7d13b8dac1e2f738e69ab00de79e6d937ec38e0bca2cd0bf/VERSION
d2c0a78744dc8dbe7d13b8dac1e2f738e69ab00de79e6d937ec38e0bca2cd0bf/json
d2c0a78744dc8dbe7d13b8dac1e2f738e69ab00de79e6d937ec38e0bca2cd0bf/layer.tar
e49ce207d81d0565ab894b1a9ff6d194305484acd360997e1a4d422ab1063aef.json
manifest.json
repositories
#
```

لاحظ المحتوى هو مجموعة من الطبقات المكونة منها هذه ال image.

سنقوم الآن بفك المحتوى من نفس الشل وذلك من خلال الأمر tar –xf كما يلي:

```
/ # tar -xf /data/iis.tar -C /data/extract
tar: can't change directory to '/data/extract': No such file or directory
/ # mkdir /data/extract
/ # tar -xf /data/iis.tar -C /data/extract
/ # cd /data/extract/
/data/extract # ls
23fc664f6b78b0b6652bbb83e52a7286d9d33c7765e6f13a2d251ce7c343ea09      d2c0a78744dc8dbe7d13b8dac1e2f738e69ab00de79e6d937ec38e0bca2cd0bf
491dd052ce3c5236841a1d870bb557d1f5d5217966a853e9aca7995506f9338e      e49ce207d81d0565ab894b1a9ff6d194305484acd360997e1a4d422ab1063aef.json
5862d78dbf431ae1056a44441f706bbb98d3d315fa1807c1cf2b499b0b185609      manifest.json
5c8322e82d15118d455379e19589a17066496ab6c1c5a4cd0dbbdd775c563831      repositories
```

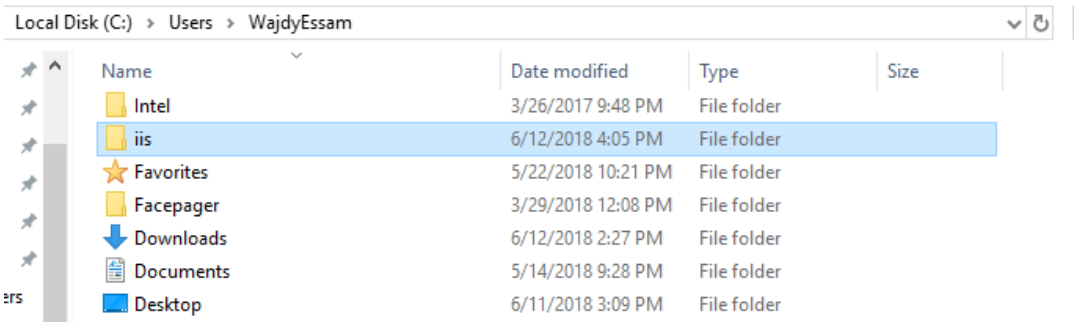
بعد فك الملف لمجلد جديد تم تحديده بالمعامل C– ظهرت رسالة بأن المجلد غير موجود، وبعدها قمنا بإنشائه ومن ثم تم فك الضغط مجدداً، وبعد ذلك قمنا بعرض محتوى المجلد الجديد وظهرت البيانات فيه.

بهذا الشكل قمنا بعمل فك للملف المضغوط في ويندوز بداخل ال container، ولكن ماذا إذا اردت أن يتم الفك أيضاً الى ويندوز؟ فالتعديل بسيط، فبدلاً من أن نقوم بعمل mount ملف من ويندوز الى مجلد بداخل لينوكس، سوف نقوم بعمل mount مجلد من ويندوز الى مجلد داخل لينوكس، وبعدها نقوم بفك الملفات في ذلك المجلد أو مجلد جديد بداخله، وسوف نرى ذلك التغيير في ويندوز.

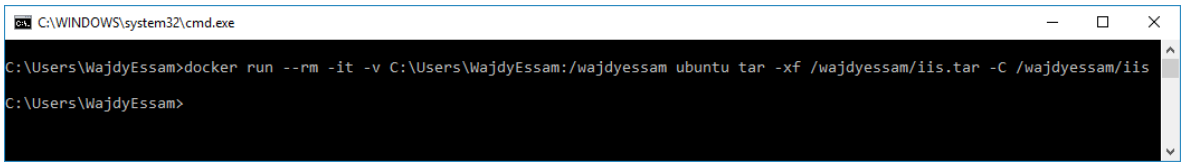
سوف نحتاج أن ننشئ مجلد جديد في الويندوز في داخل المجلد ال mounted حتى نضع الملفات فيه، ويمكن فعله من خلال الويندوز وإنشاء مجلد جديد بالزر الأيمن وتسميته مثلاً iis، أو حتى سوف نطبق نفس الفكرة وهي تشغيل امر لينوكس لإنشاء مجلد في ال mounted folder كما يلي:

```
C:\Users\WajdyEssam>docker run --rm -it -v C:\Users\WajdyEssam:/wajdyessam ubuntu mkdir /wajdyessam/iis
```

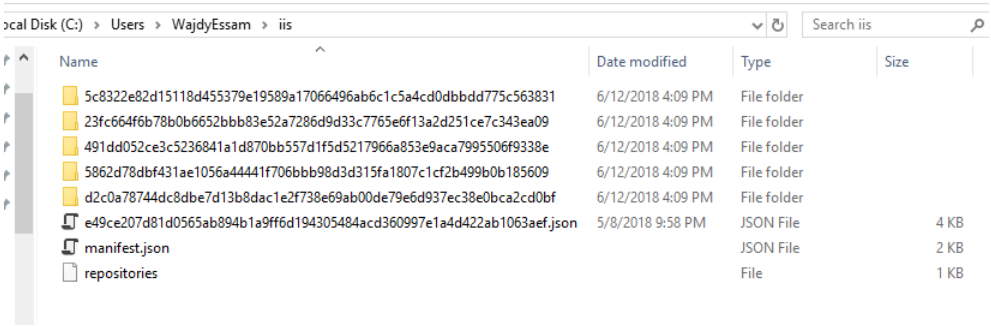
الآن قم بفتح المسار في الويندوز وستجد أن المجلد الجديد متواجد الآن:




الآن سنقوم بكتابة أمر فك الضغط الى ذلك المجلد، وبدلاً من فتح الشل ومن ثم كتابة أوامر الضغط يمكن الاختصار وتشغيلها مباشرة كما في الصورة التالية:

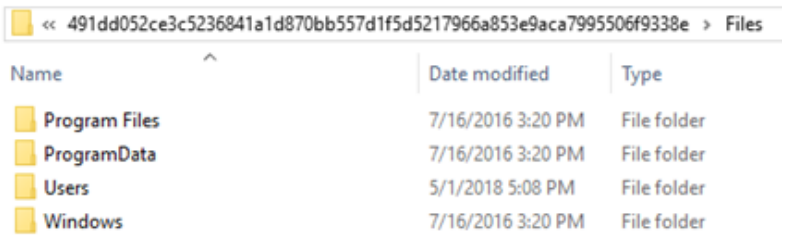



بهذا الأمر الآن سوف نجد أنه تم فك الملف المضغوط في ويندوز، الى مجلد بداخل الويندوز وذلك من خلال برنامج يعمل بداخل نظام التشغيل لينوكس، أمر رائع!




طبعاً مفهوم ال Volumes أتاح مشاركة بين مجلدات أو ملفات في النظام الأساسي وال container وبالتالي أي تغيير يتم عمله بداخل ال container في ذلك المجلد سوف يظهر في النظام الأساسي، وهذا ما تم تطبيقه في الأعلى.

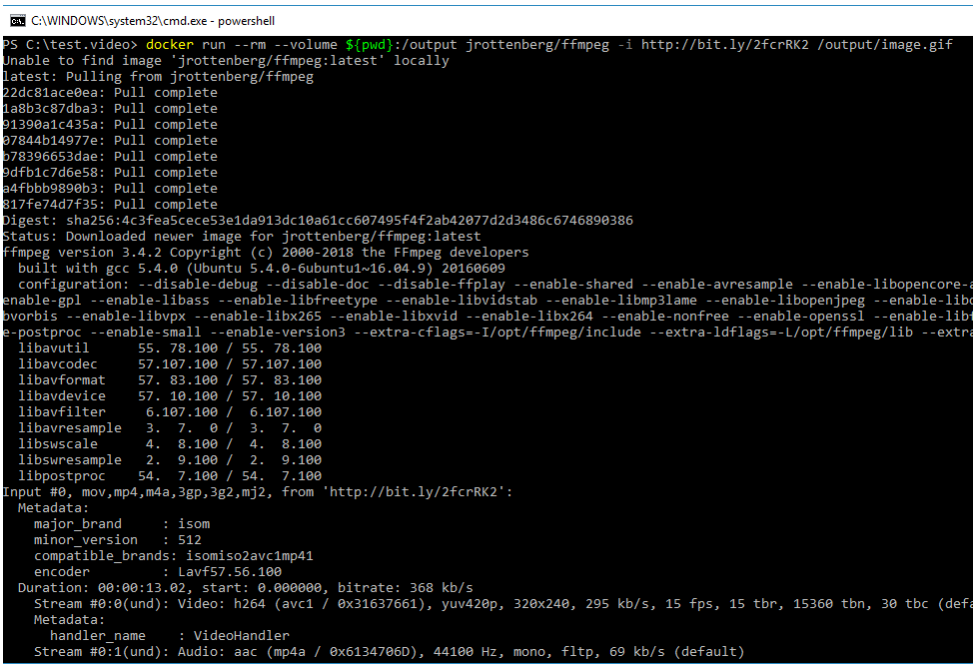
بدخل أي من المجلدات السابقة سوف نرى أيضاً ملف الطبقة مضغوط، وسوف نختار المجلد الذي يبدأ ب 49 ونقوم بفك الملف المضغوط بداخله، سوف نكرر نفس الخطوات، وهذه المرة سوف ندخل على الشل، ونقوم بفكها في مجلد نقوم بإنشائه أيضاً، وذلك من خلال أمر إنشاء المجلد mkdir باي اسم ومن ثم فك ملف الطبقة المضغوط لهذا الملف بنفس الأمر السابق، وبعد الفك سوف نقوم بفتحه من داخل الويندوز الآن، وهو يبدوا كما يلي:



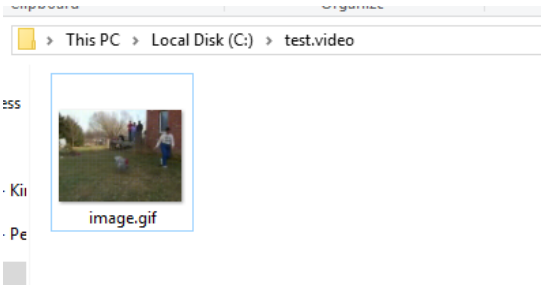
هل تلاحظ ما هو محتوى هذه الطبقة؟ هو نظام الملفات لل image، فكما ذكرنا أن ال image تحمل ال File System الخاصة بها لأنها معزولة عن الخاصة بنظام التشغيل.

## تشغيل برنامج ffmpeg للتعامل مع ملفات الفيديو

لنفرض اردت تحويل مقطع فيديو صغير الى صورة git، فسوف تحتاج الى تنزيل برنامج وتتأكد من أنه يخلوا من المشاكل وغيرها، وأحد أشهر البرامج هو ffmpeg ويعمل على عدة أنظمة تشغيل، وسوف نقوم بتشغيله بنفس الفكرة ونستفيد من ال Linux Container لكي يحول لنا الفيديو الى صورة ويضعها لنا على نظام الملفات في مجلد نحدده، مثلاً الفيديو ملف mp4 المتواجد في هذا الرابط سوف نقوم الآن بتشغيل برنامج ffmpeg وإعطاء أمر التحويل بداخل volume حتى نستطيع الوصول لها بسهولة، مثلاً أريد كتابته على الفولدر C:\test.video سوف نقوم بإنشائه ومن ثم ندخل على ال PowerShell من سطر الأوامر وننفذ أمر التشغيل:




تم فتح ال PowerShell وذلك حتى نستفيد من خاصية المتغير pwd وهي تعني المسار المحلي، لأنه كما يتضح في الصورة التنفيذ سوف يتم من داخل المجلد الذي يتواجد عليه ال cmd وهو test.video ويمكن اذا أردت تشغيله من ال cmd بشكل عادي أن تقوم بكتابة المسار الذي تريد عمل mount له. أيضاً لاحظ استخدمنا volume– بدلاً من الاختصار v– وكلاهما في دوكر لنفس الأمر وهو استخدام ال volume لكي تحدد مجلد تقوم بعمل mounting له في مجلد بداخل ال container. بعد الانتهاء سوف تجد أن الصورة الآن لديك على المجلد:

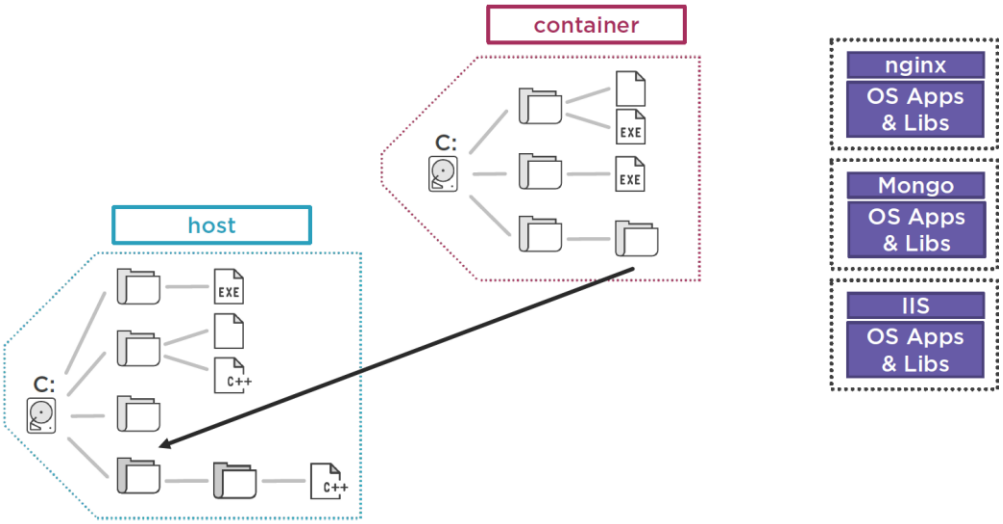


## خلاصة

فكرة ال Volume وال Mounting مشابه لفكرة تركيبك ل USB في نظام التشغيل، حيث ستجد قرص جديد لديك.

أولاً يكون النظام الأساسي Host به نظام الملفات والأقراص مثلاً في نظام الويندوز، وبعد أن يتم تشغيل Container سوف يتم تشغيلها من Image موجودة، وكما رأينا سابقاً ال Image يكون بها نظام الملفات الخاص بها وبالتالي لديها File System Tree بداخلها معزولة عن نظام التشغيل الأساسي وكل ذلك يتم بمجرد تشغيل ال Container.

الآن بعد إضافة ال Volume وعمل Mounting لمجلد من النظام الأساسي الى مجلد في ال Container فسوف تكون كأنها عبارة اختصار للوصول الى المجلد الأصلي في النظام الأساسي وأي عملية تجري سيتم مشاهدتها مباشرة من النظام الأساسي.




وسوف نرى لاحقاً أن لهذا الأمر فوائد عديدة مثلاً قمت بتطوير الموقع، فيتم وضعه على النظام الأساسي وبعدها يتم عمل mounting للمجلد الخاص بالموقع ويتم تشغيل ال nginx مثلاً، وبالتالي يتم قراءة الملفات والكتابة أيضاً بها، ولل Volumes استخدامات أخرى وأهمها الحفاظ على البيانات بعد أن يتم إيقاف أو حذف ال container وسوف يتم التطرق لها عند الحديث عن تشغيل قواعد البيانات بداخل ال Containers في مواضيع قادمة.

بهذا الشكل نكون قد تعرفنا على فكرة ال Sharing Host File الى ال Containers وكيف يتم كتابة الملفات من والى النظام الأساسي، وأيضاً وضحنا فوائد وسهولة تشغيل البرامج بداخل ال containers للقيام بأي عمليات مطلوبة وذلك من خلال أوامر دوكر البسيطة، الفصل القادم سوف يكون حول بناء Image خاصة بنا.

# الفصل الرابع: بناء ال Docker Images لاستضافة المواقع

في هذا الفصل سوف نتحدث عن كيفية بناء ال Docker Image وكيف نقوم بوضع موقعنا ك Image ومن ثم نقوم بتشغيلها، وسوف نقوم بالتعامل مع Static Website مكون من ملفات جافا سكربت و HTML ونشغله ك docker container ومن ثم تصفحه بواسطة المتصفح. وأخيراً سوف نتحدث عن ال Dockerfile وكيف يسهل بناء ال Images بسهولة.

ولكي نقوم بتشغيل الموقع في دوكر، يجب أن يتم تطوير الموقع أولاً، وفي حالتنا هذه فالموقع جاهز وهو Angular Web Site، بعد ذلك هناك العديد من الخيارات:

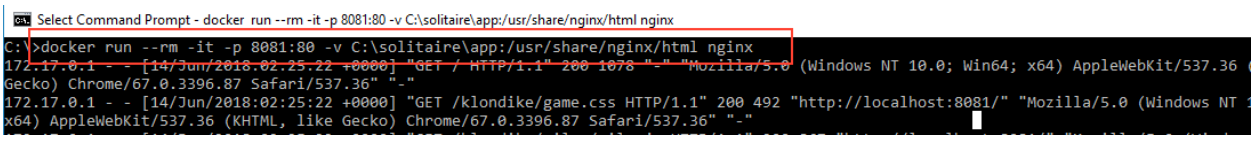
- تنزيل nginx image ومن ثم وضع ملفات الموقع في مجلد ما وعمل mount لهذا المجلد ومن ثم تشغيل ال container بهذا الملفات.
- تنزيل nginx image ومن ثم نسخ ملفات الموقع الى داخل ال Container File system.
- بناء Image بها الملفات الخاصة بالموقع ومن ثم تشغيل ال Container منها ولن تحتاج لنسخ أو أي ملفات لأن كل شيء متواجد بداخلها

الصورة التالية تبين الخطوات الثلاثة التي يمكن القيام بها لتشغيل الموقع بداخل دوكر:


قبل أن نبدأ في شرح الطرق الثلاثة هذه، قم بتحميل الموقع من هذا الرابط وبعد التحميل قم بفك الضغط وأنسخ المجلد solitaire الى أي مسار تريده، وليكن ال C:\ ، وبالتالي سوف يكون المجلد متواجد على القرص السي مباشرة.

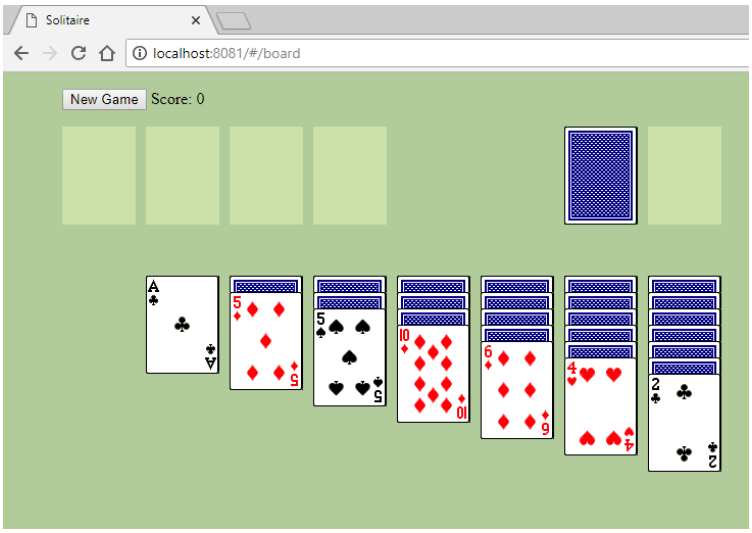
## تشغيل الموقع عن طريق ال Mounting

لقد تناولنا هذه الطريقة في الفصل السابق وهي بسيطة، حيث يتم تشغيل البرنامج المطلوب وهو في حالتنا أي ويب سيرفر مثل nginx ويتم تحديد ال v– بمسار الملفات في النظام الأساسي ومن ثم : ومن ثم المسار بداخل ال container ويكون بصيغة ذلك النظام الذي يعمل عليه. الصورة التالية تبين الامر لتشغيل ال container وتم استخدام ال p– لكي يتم تحديد البورت في الجهاز الأساسي والبورت الذي يعمل به الموقع في ال container.




لاحظ المسار بداخل ال container وغالباً لكل ويب سيرفر مسار خاص به، ويجب الرجوع للتوثيق الخاص بال nginx في ال docker hub لكي تعرف المسار المناسب لكي يعمل الموقع.

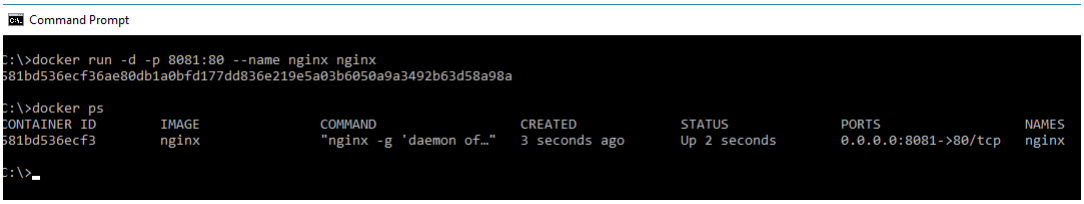
الآن قم بفتح ال localhost:8081 وسوف تجد أن اللعبة تعمل لديك:



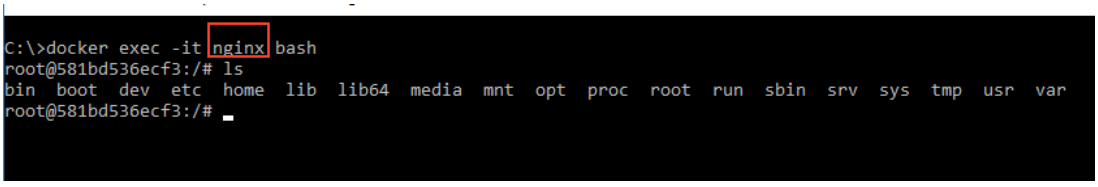

## تشغيل الموقع عن طريق نسخ الملفات لل Container

كما ذكرنا أن الدوكر يعطي Isolation لنظام الملفات، فلن تستطيع ال Container الوصول لنظام ملفات النظام الأساسي، الا في حال ال Mounting/Volume فقط ويكون بحسب ما تم تحديده بها، والطريقة الثانية هي بنسخ الملفات من النظام الأساسي الى ال Container كما هو الحال عندما يتم نقل الملفات من جهاز لجهاز أخر.

سوف نقوم أولاً بتشغيل ال nginx container في الخلفية d– وقمنا بوضع اسم لها name–، وبعدها قمنا بطباعة docker ps لكي تتأكد من أنها تعمل ولها الاسم الذي تم إعطائه لها، يمكنك الآن الذهاب للمتصفح وسوف تجد أن الموقع الرئيسية بصفحة ال nginx الافتراضية تعمل.

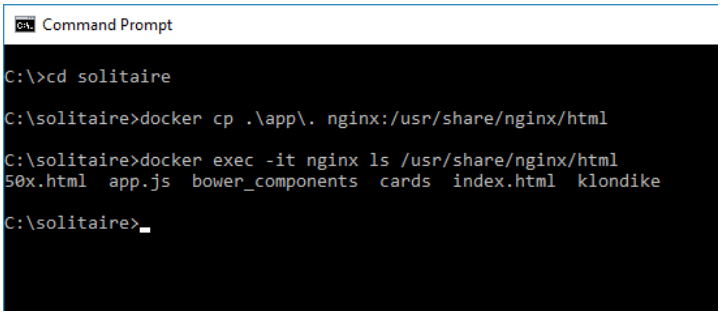



سوف نقوم الآن بالدخول لهذه ال container التي تعمل، وسوف نستخدم الأمر docker exec الذي تناولناه سابقاً وهو لتشغيل برنامج اخر بداخل نفس ال container، وهذا الأمر يأخذ اسم ال container أو ال ID ولذلك قمنا بتسمية ال container باي اسم (واخترنا nginx) وسوف نحدده الآن عند الأمر exec.




بعد الدخول على الشل قمنا بعرض الملفات وكما هو واضح فنظام الملفات مختلف تماماً عن النظام الأساسي، ويمكن الدخول على المسار /usr/share/nginx/html وسوف تجد المسار فارغاً ونحن نريد تعبئته بالملفات من النظام الأساسي، بعد ذلك يمكنك الخروج من الشل

باستخدام الأمر exit. اذاً هذه الخطوة فقط للمشاهدة والتأكد من أن المسار فارغاً، والخطوة الضرورية هي النسخ وذلك من خلال الأمر docker cp وهذا الأمر يأخذ معاملين الأول هو المسار من النظام الأساسي، ومن ثم اسم أو معرف ال container التي تعمل ومن ثم : ومن ثم المسار الذي تريد النسخ عليه، الصورة التالية تبين كيفية النسخ بهذا الأمر:

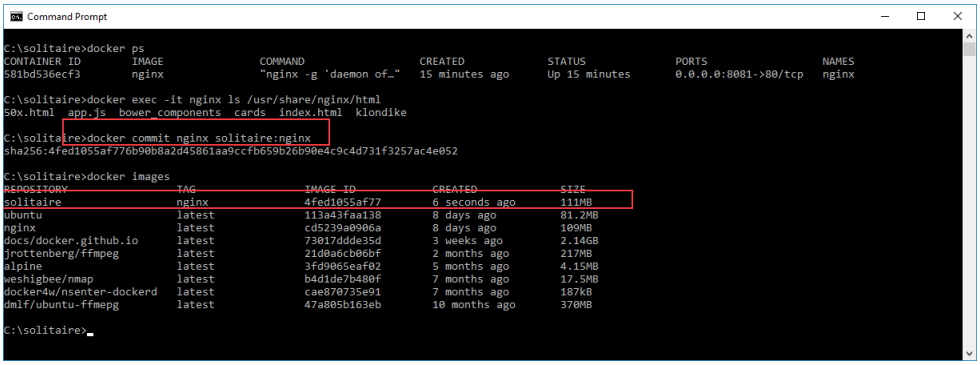



الآن يمكنك الدخول بالشل مجدداً كما في الخطوة قبل السابقة والتأكد من وجود الملفات، أو يمكنك الذهاب للمتصفح والدخول ومشاهدة الموقع الآن.

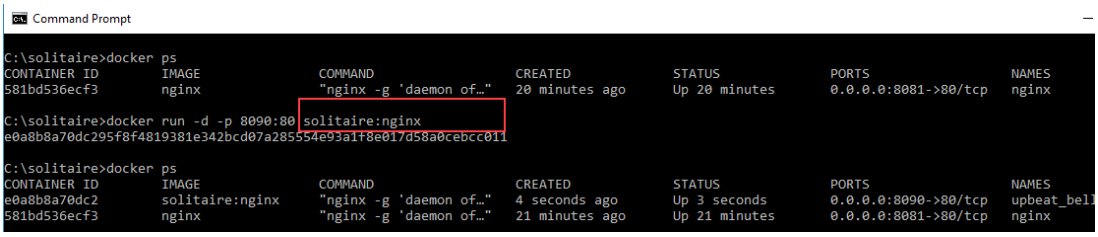
## تشغيل الموقع عن طريق وضع كامل الملفات على شكل Image

والطريقة لذلك هي نفس خطوات الطريقة الثانية، حيث نقوم أولاً بتشغيل ال container ومن ثم بنسخ الملفات بداخلها، وبعد الانتهاء وكل شيء نقوم بأخذ صورة Snapshot للحالة الحالية وهي سوف تعتبر Image ويمكن تشغيل ال Image مباشرة فيما بعد ويكون بداخلها كل الملفات.

اذاً يتم تشغيل ال Container من خلال ال Image، ويتم عمل ال Image من خلال صورة أو commit لل Container التي تعمل، وسوف نستخدم الأمر commit ويتم تحديد اسم أو معرف ال Container ومن ثم ال Image الجديدة، وأيضاً يمكن تحديد ال Tag الذي تريدها.

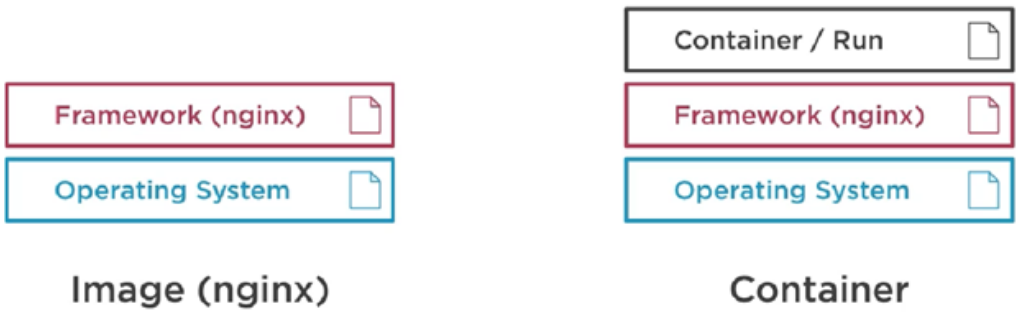



الآن سوف نقوم بتشغيل Container جديدة من ال Image الجديدة والتي بها كل شيء بدون أن نحتاج لنسخ أي ملفات أخرى، كما يلي:

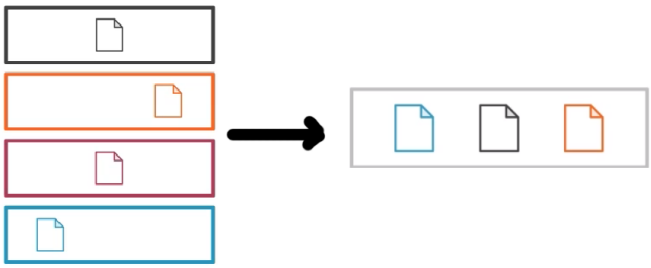


ولاحظ أنه في هذه المرة حددنا اسم ال Image التي قمنا بعملها وليس ال nginx فقط، وتستطيع الآن زيارة الموقع localhost:8090 وسوف تجده يعمل أيضاً.

للتذكير، فقد بدئنا أولاً بأخذ ال nginx Image والتي تحتوي على ملفات نظام التشغيل الأساسية، وبها برنامج nginx نفسه، وبعد ذلك عندما نقوم بتشغيل ال Container فسوف يتم إضافة layer جديدة قابلة للقراءة والكتابة وهي التي نضع بها الملفات الخاصة بنا.




ولأن كل Layer بها ملفات خاصة فسوف يتم عمل دمج بين كل هذه الملفات حتى تكون في نظام الملفات وذلك باستخدام عملية ال Union File System، مثلاً الصورة التالية تعرض ال Container مكون من عدة طبقات واخيراً الطبقة الأولى التي يمكن الكتابة عليها، ومن ثم حصل الدمج Union بين الملفات في تلك الطبقات لنحصل على نظام ملفات واحد.

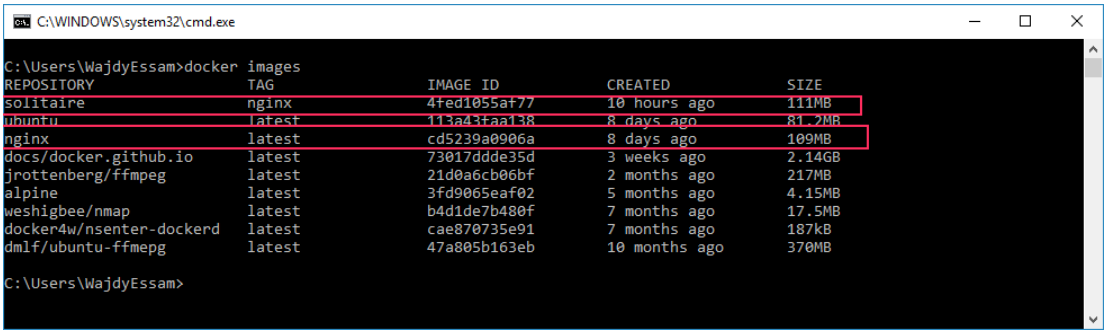


هنا العديد من الأشياء التي تحدث أثناء دمج الملفات:

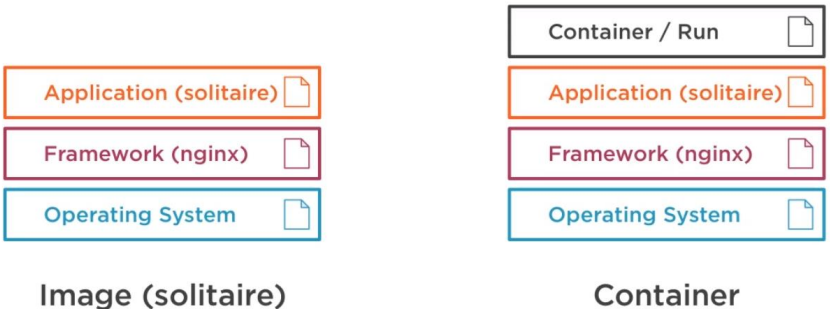
- إذا كان هناك ملف اسمه A.txt في طبقة سفلية، ومن ثم حصل تغيير في ذلك الملف في الطبقة الأعلى فسوف يتم اجراء ما في الطبقة العليا، أو حتى في حال الحذف فسوف يتم الحذف أيضاً، لذلك الطبقة العليا تفوز دائماً.
- الطبقة العليا تحتوي على التغييرات للطبقة السفلى فقط، سواءً إضافة ملفات، تحديث ملفات، أو حتى حذف ملفات. وهذا ما يجعلها صغيرة ولن تحتاج لوضع كل الملفات الموجودة في طبقة أسفل منها.

وهكذا عندما قمنا بنسخ ملفات الموقع في ال container التي بدئت بال nginx image وقمنا بعمل Image جديدة فسوف تكون ال Image الجديدة مكونة من كل الطبقات في ال nginx image بالإضافة الى طبقة واحدة جديدة بها الملفات، وذلك من خلال الأمر docker commit.

الصورة التالية توضح ذلك حيث يوضح حجم ال image الجديدة ذلك التغيير:




وبعد أن قمنا بتشغيل ال container من ال image الجديدة سوف يعمل وتضاف الطبقة الأولى الخاصة بالكتابة والقراءة لذلك ال container حتى يستطيع إضافة أي تعديلات يريدها ويمكن بعد ذلك حفظ image أخرى وهكذا.




يمكنك أن تقوم بعمل export لل image وتقوم بفك ضغطها ومشاهدة الملفات الخاصة بالموقع في أحد المجلدات كما قمنا بذلك في الفصل السابق.

## استخدام ال Dockerfile بدلاً من إعادة كتابة الأوامر

لقد قمنا بالعديد من الخطوات حتى نقوم بعمل Image خاصة بنا:

1- تشغيل container يعتمد على nginx image من خلال الأمر docker run –d –p 8080:80 –name nginx وهذا الأمر سيقوم بتنزيل nginx image وتشغيل container في الخلفية ويعطيه اسم nginx

2- نسخ الملفات من النظام الأساسي الى مجلد بداخل الويب سيرفر في ال nginx container التي تعمل الآن من خلال الأمر docker cp .\app\. nginx:/usr/share/nginx/html

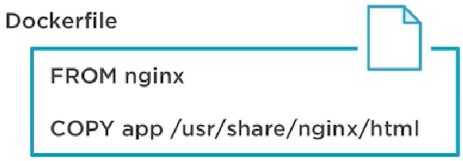
3- أخذ صورة من ال nginx container من خلال الأمر: docker commit nginx solitaire:nginx وتم تسميتها بالاسم solitaire ووضع تاق بها أيضاً nginx.

4- وقد تحتاج لأوامر نسخ أخرى أو حذف لملفات معينة وسوف تكون موجود هنا، مثلاً تريد وضع اعدادات خاصة لل nginx أو حذف الكاش أو غيرها من الأمور، وكلها تكون قبل ال committing لل container حتى تخرج ال image الجديدة

```
docker run -d -p 8080:80 --name nginx nginx

docker cp .\app\. nginx:/usr/share/nginx/html

docker commit nginx solitaire:nginx
```

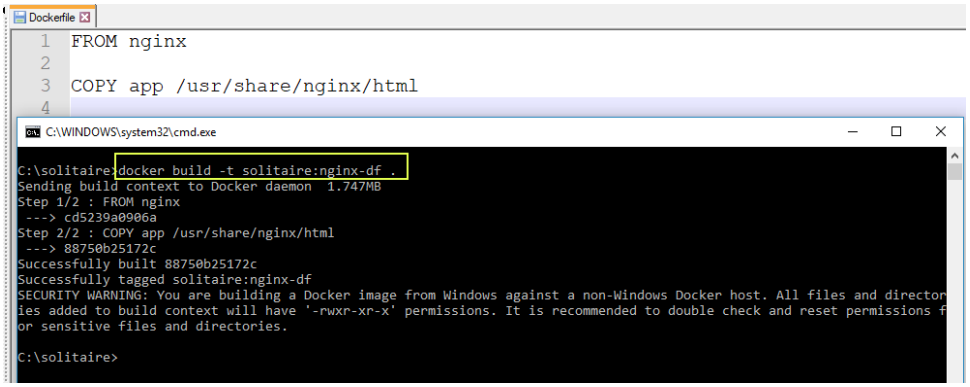
كل هذه الخطوات حتى تنشئ ال image التي تريدها، وبدل من أن تكتبها في كل مرة فيمكنك وضعها في ملف سكربت اسمه Dockerfile وسوف يقوم دوكر بتشغيلها واحداً تلو الأخر، وسوف نقوم الآن بتحويل الأوامر أعلاه الى Dockerfile

1- بدلاً من تشغيل Container ونسخ ملفات فيه، فسوف نقوم بتحديد ال Image وذلك من خلال الأمر FROM nginx
2- نسخ الملفات وذلك من خلال الأمر COPY app /usr/share/nginx/html
3- انتهينا من الملف ونقوم بحفظه، وسوف نقوم ببناء ال image وذلك من خلال الأمر docker build ويتم تمرير ملف ال Dockerfile لها، ويتم تمرير اسم ال image الجديدة لها أيضاً، وذلك من خلال الأمر docker build –f Dockerfile –t solitaire:nginx . ويمكن اختصار الأمر الى docker build –t solitaire:nginx . في حال تم تسمية ال Dockerfile بالاسم الافتراضي. المعامل الأخير هو مسار التي يتواجد عليه الملف وهو . وتعني المسار الحالي.



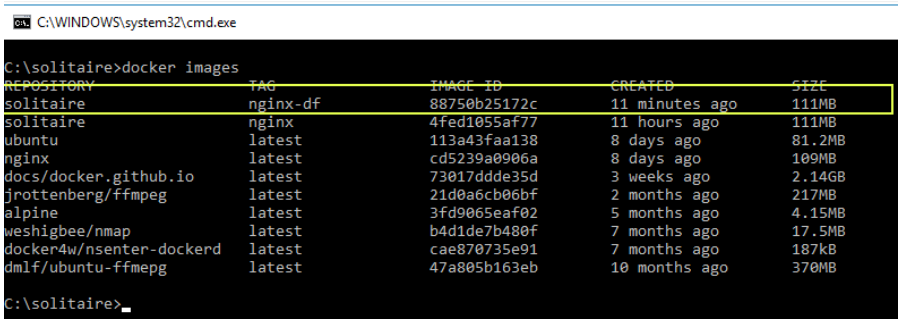

docker build -t solitaire:nginx .

الصورة التالية توضح الخطوات فقمنا بإنشاء الملف في نفس المسار solitaire folder وقمنا بوضع محتوى ال Dockerfile وقمنا ببنائه كما يلي:




لكل أمر سيتم تطبيقه على حدة، وأول أمر FROM nginx سوف يقوم فعلياً بتنزيل ال image إذا لم تكن موجودة، ومن ثم بتشغيلها، وبعد ذلك ينتقل للخطوة التالية وهي نسخ الملفات بداخلها، وبعد ذلك تم حذف ال container التي استخدمت اثناء عملية نقل الملفات، واخيراً تم حفظ ال image الجديدة، وظهر تحذير بأنه تم بناء Linux container في ويندوز ولكنه عادي أثناء التطوير، لكن في ال production فيفضل البناء على لينوكس إذا كنت تستخدم ال Linux Container.

يمكنك الآن أن تستعرض ال images الموجودة وسوف ترى ال image الجديدة:



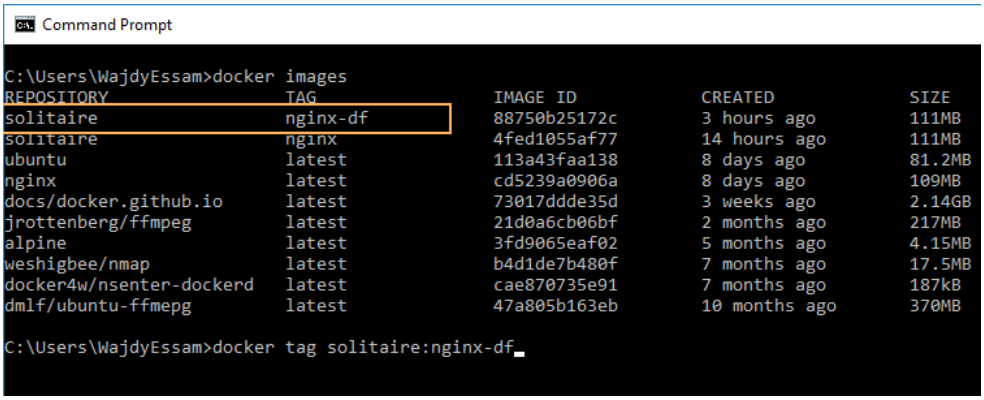

وسوف تكون بنفس الحجم لل image التي قمنا بإنشائها يدوياً، وهذا يعني أن النتيجة هي واحدة سواءً بكتابة كافة الأوامر، أو من خلال الاستفادة من ال Dockerfile وبنائه وهو الأسهل بالتأكيد.

يمكنك كتدريب أن تقوم بالتحويل الى Windows Container ومن ثم استضافة وتشغيل الموقع على ال iis بدلاً من ال nginx وذلك بتغيير ال nginx في Dockerfile الى iis:nanoserver وأيضاً تغيير مسار الملفات على الويب سيرفر الى مسار ال iis وهو C:\inetpub\www وسيعمل الموقع بلا مشاكل على ال Windows Container.

## رفع ال Image على ال Docker hub

بعد أن قمنا بعمل ال Images بعدة طرق فقد تريد مشاركتها سواءً لمطور اخر أو لسيرفر اخر يقوم بتنزيلها وتشغيل النسخة مباشرة، وسوف نقوم برفعها على ال Docker Hub، وللقيام بذلك يجب تسجيل حساب في ال Docker Hub وبعد ذلك سوف ترفعها على اسم المستخدم الخاص بك، ويجب أن نقوم بتسمية ال image بما يتناسب مع الحساب حتى يتم رفعها بشكل صحيحاً.

سوف نظهر الآن جميع ال images الموجودة:




وبعد ذلك نقوم بتسمية ال Image وسوف نستخدم الأمر docker tag ويتم تمرير اسم ال image الموجودة، والمعامل الثاني هو الاسم الجديد التي نريد رفعها باسم الحساب، فمثلاً اسم حسابي هو wajdyessam فسوف يكون اسمها wajdyessam/new_name، كما يلي:

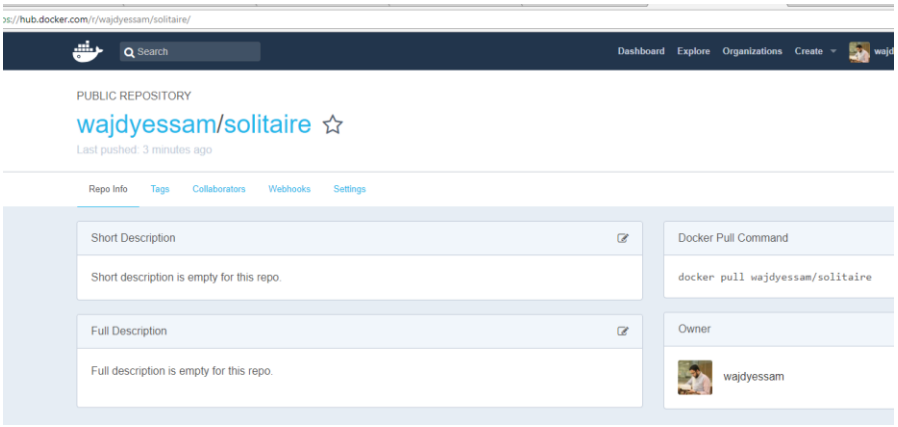
```
Command Prompt

C:\Users\WajdyEssam>docker tag solitaire:nginx-df wajdyessam/solitaire:nginx

C:\Users\WajdyEssam>docker images
REPOSITORY                TAG                 IMAGE ID            CREATED             SIZE
solitaire                 nginx-df            88750b25172c        3 hours ago         111MB
wajdyessam/solitaire      nginx               88750b25172c        3 hours ago         111MB
solitaire                 nginx               4fed1055af77        15 hours ago        111MB
ubuntu                    latest              113a43faa138        8 days ago          81.2MB
nginx                     latest              cd5239a0906a        8 days ago          109MB
docs/docker.github.io     latest              73017ddde35d        3 weeks ago         2.14GB
jrottenberg/ffmpeg        latest              21d0a6cb06bf        2 months ago        217MB
alpine                    latest              3fd9065eaf02        5 months ago        4.15MB
weshigbee/nmap            latest              b4d1de7b480f        7 months ago        17.5MB
docker4w/nsenter-dockerd  latest              cae870735e91        7 months ago        187kB
dmlf/ubuntu-ffmepg        latest              47a805b163eb        10 months ago       370MB

C:\Users\WajdyEssam>
```

بعد ذلك سوف نستخدم الأمر docker push ويتم تمرير اسم ال Image التي نريد رفعها وسوف يتم رفعها مباشرة في الحساب:

```
C:\Users\WajdyEssam>docker push wajdyessam/solitaire:nginx
The push refers to repository [docker.io/wajdyessam/solitaire]
611f0060eefb: Pushed
3ff93588120e: Mounted from library/nginx
24ee0a3fd4b9: Mounted from library/nginx
d626a8ad97a1: Mounted from library/nginx
nginx: digest: sha256:84d026af0b7b5e1d5833d5d0e3066ce1c64bff59287c13f7524fc5990d716a43 size: 1158

C:\Users\WajdyEssam>
```

هذه الخطوة تستوجب أنك قمت من قبل بالدخول على حسابك في الدوكر Hub، فاذا حصلت على رسالة خطأ Authentication Required فيجب عليك الدخول لحسابك أولاً قبل رفعك لل Image وذلك من خلال الأمر docker login ومن ثم سوف يطلب منك اسم المستخدم وكلمة المرور وسوف تحصل على رسالة بأنه تم تسجيل الدخول بنجاح.

الآن تستطيع الدخول لحسابك وسوف تجد ال Image مرفوعة، وكما يمكنك تشغيلها مباشرة بالأمر docker run –d wajdyessam/solitaire من أي جهاز تريده.




وصلنا لنهاية الفصل، وتعرفنا على ال Dockerfile وأنه سهل الكثير من الخطوات لبناء ال Image والتي يتم اخذ صورة من ال Container الذي يعمل حتى يتم بنائها، الفصل القادم سوف نتحدث عن تشغيل قواعد البيانات في ال Containers.

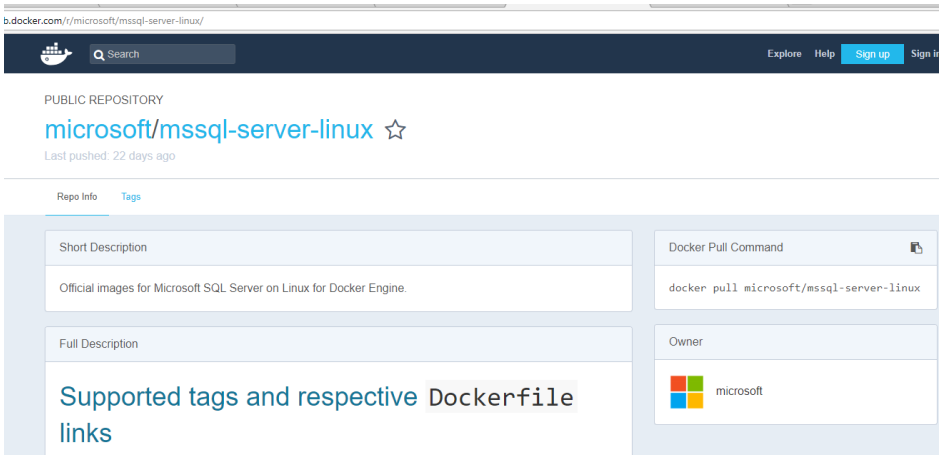
# الفصل الخامس: تشغيل قواعد البيانات في دوكر

عملية تنصيب وتشغيل قواعد البيانات بالطريقة التقليدية ليست سهلة وتحتاج للعديد من الخطوات والاعدادات، في هذا الفصل سوف نقوم بالتعامل مع Microsoft SQL Server وأيضاً MySQL ونقوم بتشغيلهم بداخل ال Containers، وسوف نتحدث عن كيفية الاحتفاظ بالبيانات حتى لو تم حذف ال Container وذلك من خلال ال Volumes. وأخيراً سوف نتحدث عن أوامر حذف ال Images وال Containers وال Volumes.

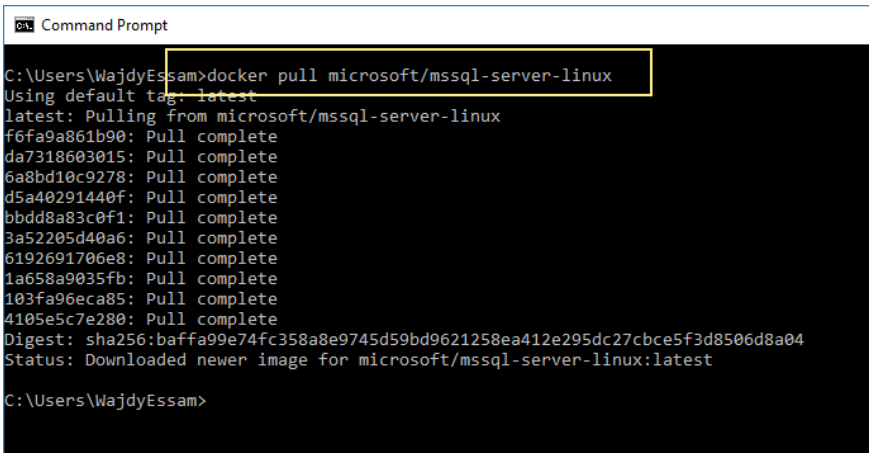
## تشغيل MS-SQL Server بداخل ال Container

حتى وقت قريب كانت كل ال MS-SQL Server تعمل بداخل ويندوز، ولكن بدءاً من الإصدارات الحديثة قامت مايكروسوفت بدعم نظام لينوكس، وأصبح بالإمكان تشغيلها على لينوكس. وفي دوكر سوف تجد هناك بعض ال Images لل SQL Server التي تعمل على ال Linux Container وبعضها على ال Windows Container.

وسوف نستخدم النسخة التي تعمل على ال Linux Container بداخل ويندوز وهي نسخة رسمية من مايكروسوفت:

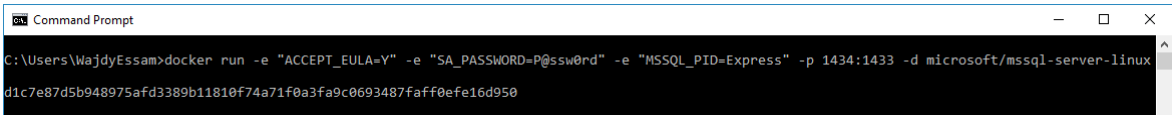



سوف نقوم أولاً بتحميل ال Image المطلوبة:




بعد ذلك سوف نقوم بتشغيل ال SQL Server، وعادة يجب الرجوع للتوثيق في ال Docker Hub لتعرف ما هي القيم المرسلة التي يجب ارسالها سواءً عند التشغيل أو لتغيير الاعدادات الافتراضية، ومثلاً في قواعد البيانات يجب أن تضع كلمة مرور للمستخدم الأساسي، وتحدد

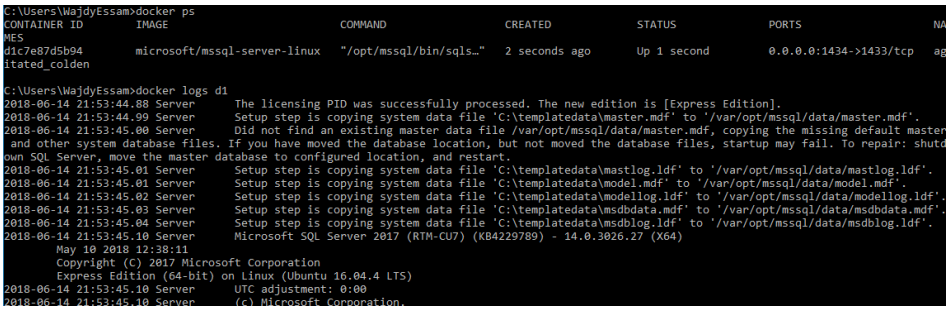
ال Mapping Port في النظام الأساسي والبورت الذي يعمل عليه ال SQL Server بداخل ال Container حتى تستطيع الوصول لها من النظام الأساسي، الصورة التالية تبين تشغيل ال SQL Server في ثواني معدودة وليس كما لو تستخدم الطريقة التقليدية فتثبيته يحتاج وقتاً واعدادات.




في الأمر أعلاه قمنا بتمرير عدة متغيرات نظام Environment Variable وهي الطريقة لكي ترسل أي متغيرات واعدادات للبرامج، حيث وافقنا على الشروط، ووضعنا كلمة مرور ويجب أن تكون معقدة Complex والا سوف تحصل على خطأ، وأيضاً حددنا أنه نريد ال Express Edition وتم تشغيل ال Container وحصلنا على ال ID

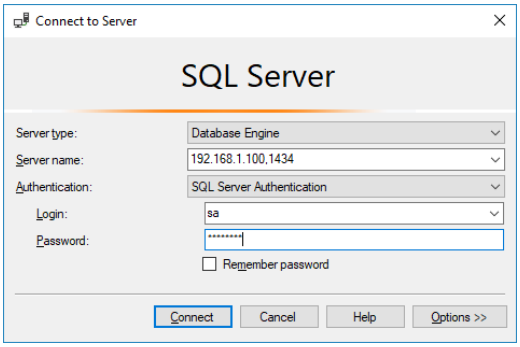
```
docker run -e "ACCEPT_EULA=Y" -e "SA_PASSWORD=P@ssw0rd" -e "MSSQL_PID=Express" -p 1434:1433 -d microsoft/mssql-server-linux
```

سوف نقوم بمشاهدة السجلات logs لل Container وهي قد تحتاجها في حال لم تعمل أي Container فعليك استخدام الأمر docker logs id حيث تشاهد كل السجلات وتعرف الخطأ الذي صدر:



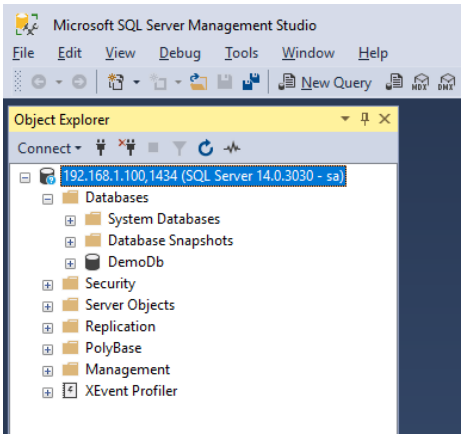

الآن سوف نقوم بالاتصال مع ال SQL Server من خلال ال SQL Server Management Studio وسوف أقوم في النظام الأساسي Host بفتحه والاتصال من خلال اسم المستخدم وعنوان IP الجهاز الحالي لأن ال Container تعمل عليه وقمت بعمل Mapping للبورت وبالتالي أستطيع الوصول له من خلال ذلك البورت.

من المهم أن تستخدم نسخة حديثة من ال SQL Server Management Studio مثلاً 2017 حيث هناك مشاكل تحدث عندما تحاول الوصول للسيرفر من كلاينت يعمل بنسخة قديمة، سواءً كان يعمل بداخل دوكر أو حتى بشكل عادي، لذلك قم بتحميل ال Microsoft SQL Server Management Studio 2017.



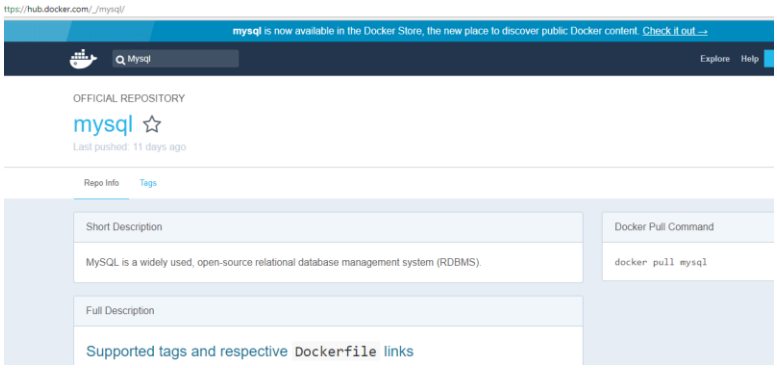
لاحظ طريقة الاتصال وهو برقم ال IP للجهاز ويمكن الحصول عليه من خلال ipconfig ومن ثم فاصلة ورقم البورت الذي قمت بعمل Mapping ولأنه يتواجد لدى في جهازي الأساسي SQL Server يعمل على البورت 1433 الافتراضي فقمت بتغيير البورت لبورت اخر.

هكذا تم الاتصال بين ال SQL Server Client المتواجد على النظام الأساسي في ويندوز، وتستطيع إنشاء أي قاعدة بيانات تريدها بشكل طبيعي.

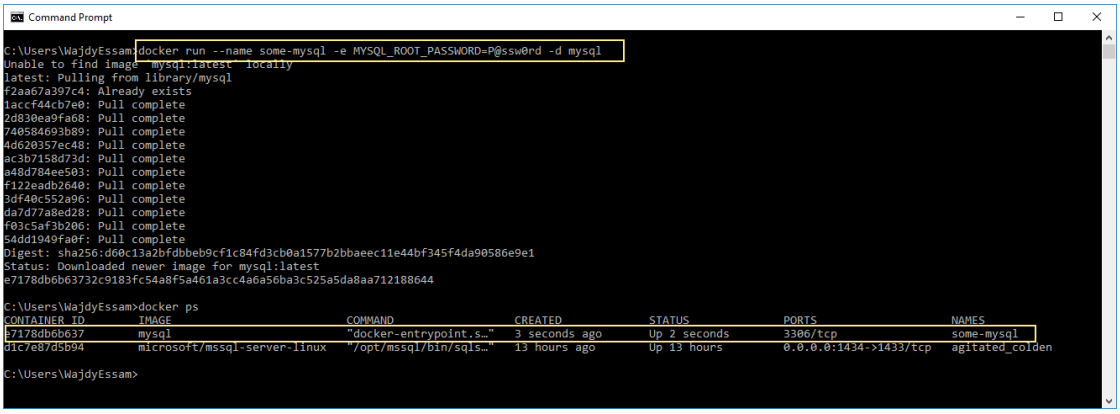



## تشغيل MySQL بداخل ال Container

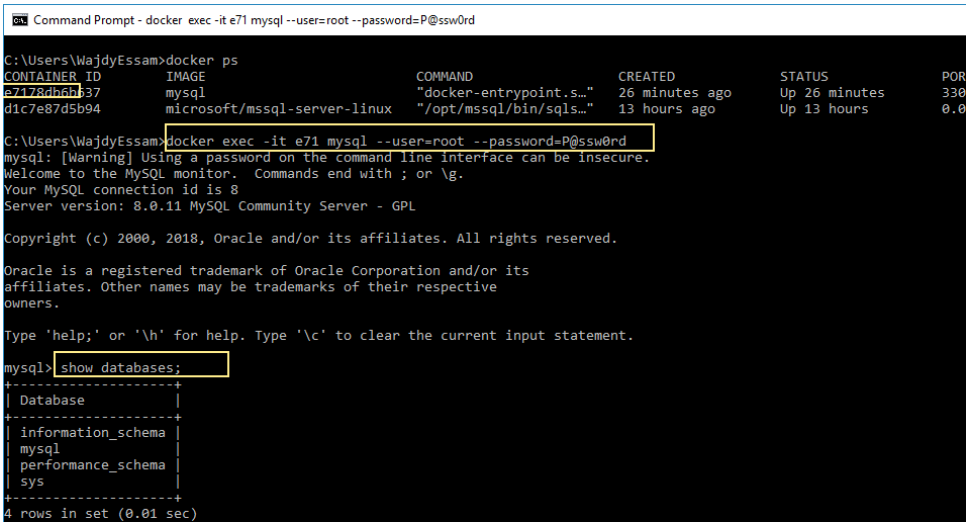
يمكنك تحميل النسخة الرسمية ل MySQL من خلال Docker Hub:




وسوف تجد في هذه الصفحة تفاصيل تشغيل الداتابيس وما هي البورت التي يجب عمل Mapping لها، وسوف نقوم بتشغيلها كالتالي:



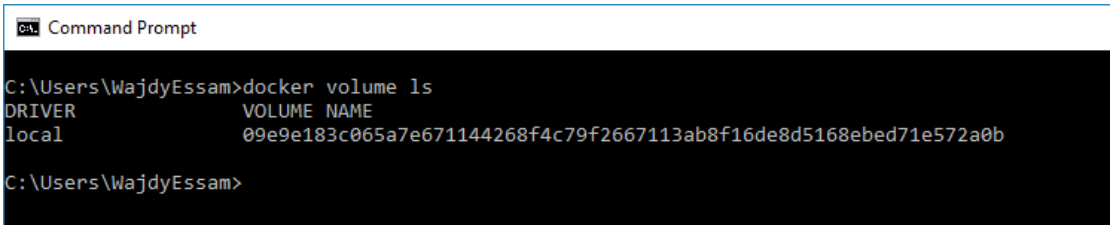
لاحظ أننا لم نقوم بعمل Mapping لل container وبالتالي لن نستطيع الوصول لها من النظام الأساسي، ويمكن القيام بذلك أو يمكن أن نستخدم ال MySQL Client المتواجد في هذه ال Container، لذلك سوف نقوم بالدخول عليها من خلال الأمر docker exec ونقوم بالاتصال مع الكلاينت لكي نستعرض قاعدة البيانات التي تعمل الآن.



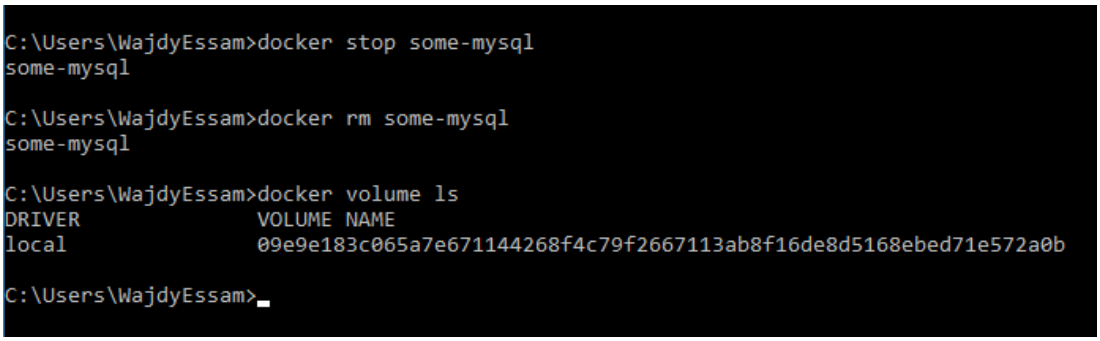

## حفظ البيانات في ال Volumes

تحدثنا سابقاً عن مفهوم ال Volume وأنه يمكن عمل Mounting لمجلد من النظام الأساسي الى مسار بداخل ال Container، ولقد ذكرنا أن هناك استخدام آخر لل Volumes الا وهو في حفظ البيانات حتى لو تم إيقاف أو حذف ال Container سوف تبقى البيانات محفوظة في حال تم استخدام ال Volume.

لنقوم بكتابة الأمر التالي لعرض كل ال Volume الموجودة في الجهاز docker volume ls:




لنجرب الآن إيقاف ال MySQL وذلك من خلال الأمر docker stop some-mysql وقم بتنفيذ الأمر docker volume ls مجدداً وسوف تجد ال volume متواجد أيضاً، وهذه ال Volume الموجودة تم إنشائها عند تشغيل ال MySQL أول مرة.



وهذا الأمر جيد فحتى لو حذفت ال database container فسوف تكون البيانات موجودة، والبيانات سوف تكون موجودة على النظام الأساسي في مجلد باسم غير محدد (لكن بالإمكان تحديده كما سيلي) مع مسار تخزين البيانات بداخل ال container وهذه يتم تحديدها بواسطة ال image ومن قام ببنائها، وبالتالي لن يتم تخزين أي بيانات في ال Container وانما ستكون بنفس فكرة ال Volume التي تحدثنا عنها سابقاً وأنها مشاركة للمجلد المتواجد على النظام الأساسي.

بالتالي في حال قمت بتشغيل new MySQL container فسوف تجد أنه يعمل بنفس البيانات بالضبط، وهذه ميزة جيدة حيث بالإمكان ترقية ال database version وتشغيله على نفس البيانات بالضبط.

سوف نقوم بتشغيل MySQL instance جديدة وسوف نحدد named volume لكي يستخدمها، وبعد الإنشاء كما تلاحظ سوف يتم إنشاء volume بالاسم الذي تم تحديده بدلاً من رقم عشوائي

```
Command Prompt

C:\Users\WajdyEssam>docker run --name some-mysql -e MYSQL_ROOT_PASSWORD=P@ssw0rd -d -v db:/var/lib/mysql mysql
1ca12f40ddf80248a20733c5229f8d526b57442d8b446a5f9bc74b5cd8f84d6f

C:\Users\WajdyEssam>docker volume ls
DRIVER              VOLUME NAME
local               09e9e183c065a7e671144268f4c79f2667113ab8f16de8d5168ebed71e572a0b
local               db

C:\Users\WajdyEssam>
```

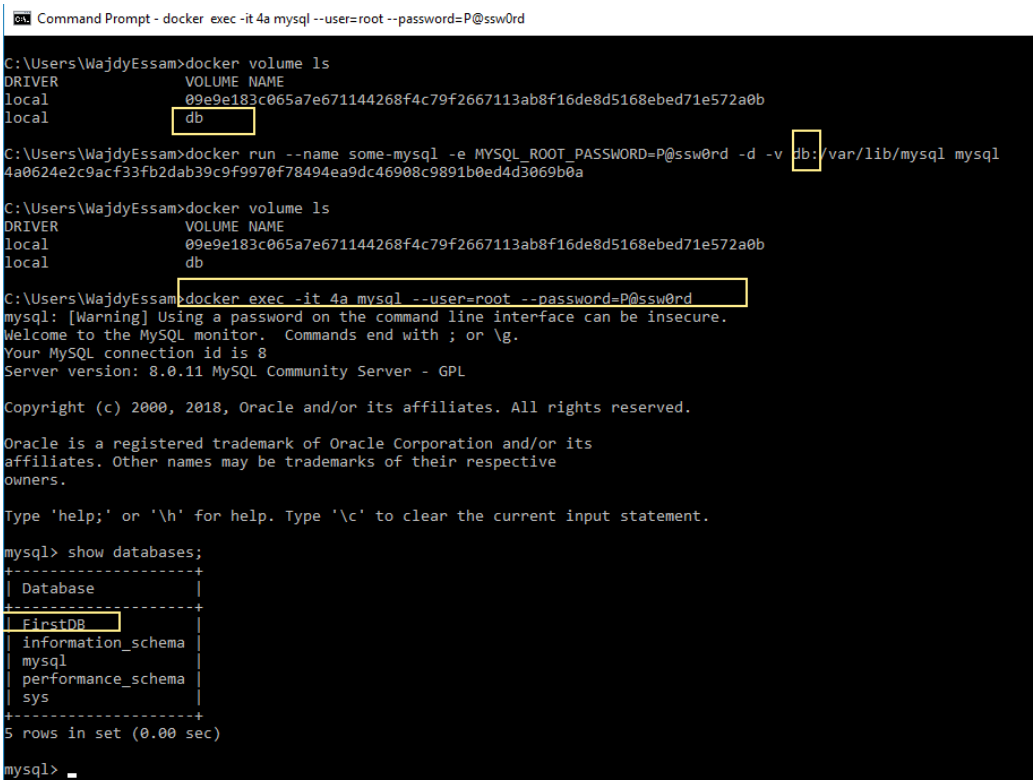
سوف نقوم الآن بالدخول لل MySQL وننشئ قاعدة بيانات جديدة بالاسم firstDb كما يلي:

```
C:\Users\WajdyEssam>docker exec -it 1c mysql --user=root --password=P@ssw0rd
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.11 MySQL Community Server - GPL

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE FirstDB;
Query OK, 1 row affected (0.08 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| FirstDB            |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)

mysql>
```

بعد ذلك سنقوم بحذف ال MySQL container ونتأكد من أنه لا يوجد أي MySQL Container تعمل:

```
Select Command Prompt

C:\Users\WajdyEssam>docker ps
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS              PORTS                    NAMES
1ca12f40ddf8        mysql                          "docker-entrypoint.s…"   5 minutes ago       Up 5 minutes        3306/tcp                 some-mysql
d1c7e87d5b94        microsoft/mssql-server-linux   "/opt/mssql/bin/sqls…"   13 hours ago        Up 13 hours         0.0.0.0:1434->1433/tcp   agitated_colden

C:\Users\WajdyEssam>docker rm -f 1ca
1ca

C:\Users\WajdyEssam>docker ps
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS              PORTS                    NAMES
d1c7e87d5b94        microsoft/mssql-server-linux   "/opt/mssql/bin/sqls…"   13 hours ago        Up 13 hours         0.0.0.0:1434->1433/tcp   agitated_colden

C:\Users\WajdyEssam>
```

بعد حذف ال container فلن يتم حذف ال Volume، وفي حال قم بتشغيل new MySQL container بدون تحديد ال volume فسوف يتم إنشاء volume جديد، ولكن في حال قمت بتحديد volume موجود لديك فسوف يتم استخدامه، الصورة التالية تبين أولاً عرض ال volume ومن ثم قمنا بتشغيل ال MySQL container بال volume المحدد، وبعد ذلك عرضنا ال volume للتأكد من عدم إنشاء volume جديدة، وأخيراً قمنا بالدخول على ال container وعرضنا الداتابيس وفعلاً وجدنا قاعدة البيانات التي أنشأناها مسبقاً.

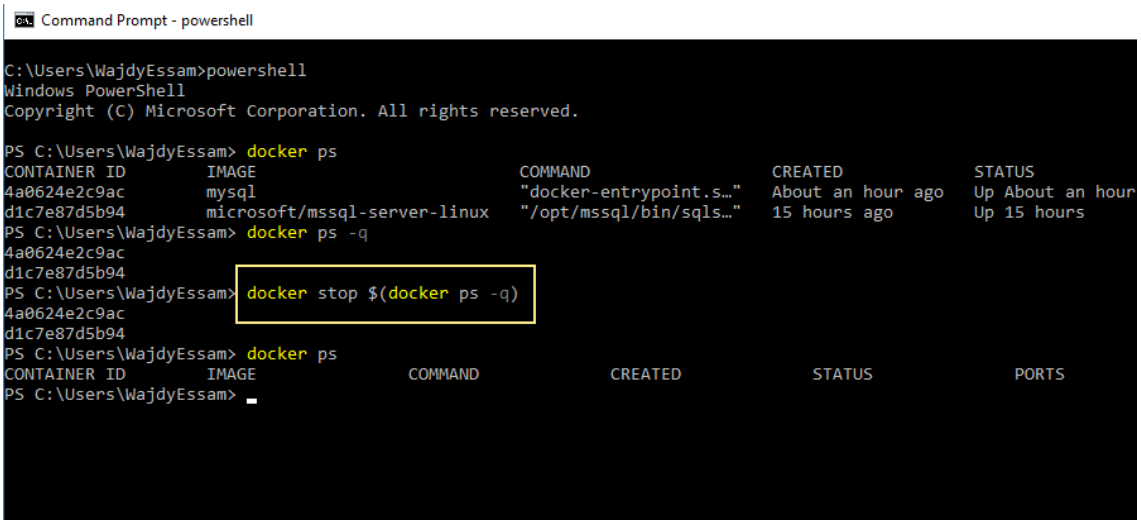



وهكذا تكون قواعد البيانات محفوظة بغض النظر عن ال container الذي قام بإنشائها.

## الحذف - إيقاف ال Containers

عند تعاملك مع ال containers سوف تجد أنك تقوم بتحميل العديد من ال images وقد تكون كبيرة ولذلك قد تود حذف البيانات التي لا تريدها.

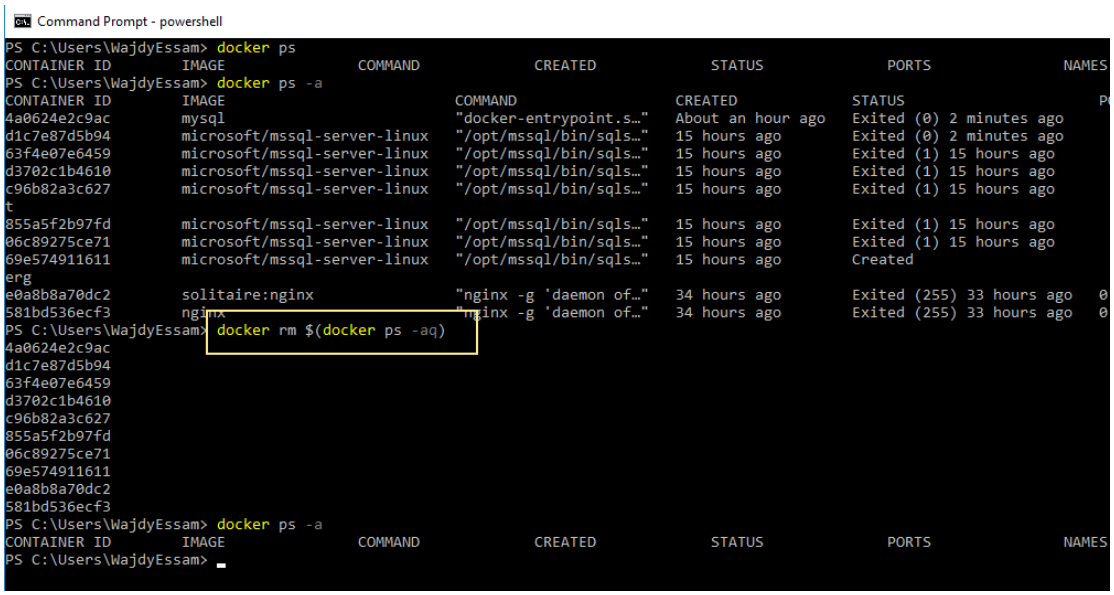
نريد أولاً إيقاف ال Containers التي تعمل، وذلك من خلال الأمر docker stop id ولكن في حال لديك عدة containers تعمل وتريد ايقافهم جميعاً فيمكنك استخدام الأمر docker ps –q والذي يعرض ال Id لل containers، الصورة التالية توضح الأمر docker ps –q حيث ظهر كل ال id containers، والأمر الثاني هو دمج docker stop مع docker ps –q وبالتالي إيقاف كل ال containers التي تعمل.




لاحظ قمنا بالدخول أولاً على ال PowerShell حتى يسمح لنا باستخدام معامل $ وهكذا كما هو واضح تم إيقاف كل ال containers التي تعمل.

## الحذف - حذف ال Containers

عند ايقافك لل containers فسوف تتوقف من العمل ولكنها تكون موجودة ويمكن تشغيلها مجدداً، وإذا اردت حذفها فهذا عن طريق الأمر docker rm id وسوف نقوم الآن بعرض كافة ال containers (التي تعمل والمتوقفة) وهم جميعهم الآن Stopped ونقوم بحذفهم جميعاً وذلك من خلال دمج الأمر docker ps –a مع الأمر docker rm مع إضافة q الى docker ps –aq حتى نحصل على ال ids لهم فقط، كما يلي:




وأيضاً يمكن استخدام الأمر docker container prune والذي سوف يقوم بحذف كل ال containers المتوقفة. إذا لم تكن تريد حذف كل شيء، وذلك لأن الأمر docker ps –a سوف يظهر جميع ال containers التي تعمل والتي لا تعمل.

## الحذف - حذف ال Volumes

كما ذكرنا يتم إنشاء ال Volume بواسطة ال Database Containers غالباً حيث نريد الاحتفاظ بالبيانات حتى بعد حذف ال containers، وفيما يلي سوف نستخدم أمر الحذف docker volume rm ونمرر لها اسم ال volume، أو نقوم بحذفهم جميعاً عن طريق الدمج docker volume ls –q مع أمر الحذف كما يلي:

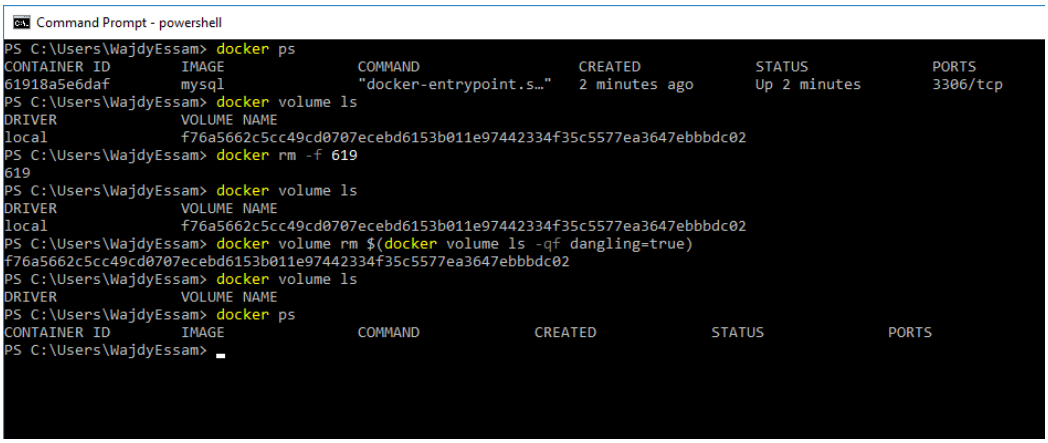
```
Command Prompt - powershell
PS C:\Users\WajdyEssam> docker volume ls
DRIVER              VOLUME NAME
local               09e9e183c065a7e671144268f4c79f2667113ab8f16de8d5168ebed71e572a0b
local               db
PS C:\Users\WajdyEssam> docker volume ls -q
09e9e183c065a7e671144268f4c79f2667113ab8f16de8d5168ebed71e572a0b
db
PS C:\Users\WajdyEssam> docker volume rm $(docker volume ls -q)
09e9e183c065a7e671144268f4c79f2667113ab8f16de8d5168ebed71e572a0b
db
PS C:\Users\WajdyEssam> docker volume ls
DRIVER              VOLUME NAME
PS C:\Users\WajdyEssam>
```

ويمكن أيضاً استخدام الأمر docker volume prune لحذف ال volume غير المستخدمة بواسطة containers. وأيضاً يمكن استخدام الأمر docker rm –fv id عند إيقاف وحذف ال container وتحديد v لحذف ال volume المرتبطة به، ولكن يجب أن تكون بدون اسم، وسيتم حذفها معهم.

عندما يتم حذف ال container المرتبط بال volume سوف تعتبر dangling volume لأنها غير مستخدمة وال container تم حذفها أيضاً، فيمكن استخدام امر لحذف ال dangling volume كما يلي، وسيتم أولاً إنشاء 2 containers وبعدها سنقوم بحذف واحد منهم، وبالتالي أحد ال volume سوف يكون dangling، وسوف نقوم بعرضه أولاً ومن ثم حذفه:

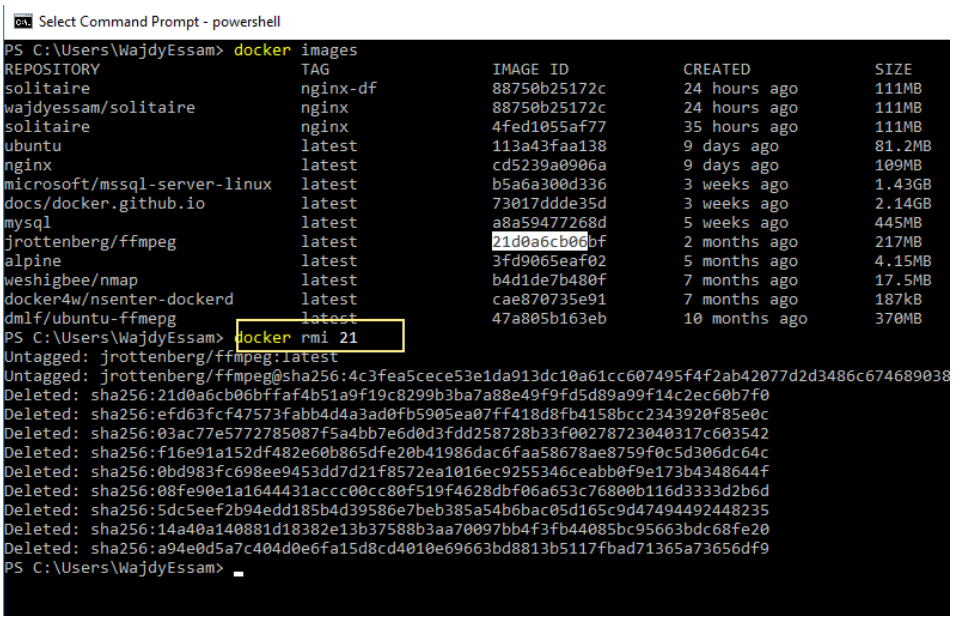
```
Command Prompt - powershell
PS C:\Users\WajdyEssam> docker run -e MYSQL_ROOT_PASSWORD=P@ssword -d mysql
61918a5e6daf5a16e9239576c1581db1399dc2c6a3fdb19caa2087c4468d4ce7
PS C:\Users\WajdyEssam> docker run -e MYSQL_ROOT_PASSWORD=P@ssword -d mysql
65c5f24ddd200f13684ceb00b90eae90f1610c8541ca105eb0b66861e309dd5f
PS C:\Users\WajdyEssam> docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
65c5f24ddd20        mysql               "docker-entrypoint.s…"   3 seconds ago       Up 2 seco
61918a5e6daf        mysql               "docker-entrypoint.s…"   5 seconds ago       Up 4 seco
PS C:\Users\WajdyEssam> docker volume ls
DRIVER              VOLUME NAME
local               216496b6fb9658b8616956b6166794e4e750a95c8da3f0df440ba6a09089afdc
local               f76a5662c5cc49cd0707ecebd6153b011e97442334f35c5577ea3647ebbbdc02
PS C:\Users\WajdyEssam> docker rm -f 65c
65c
PS C:\Users\WajdyEssam> docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
61918a5e6daf        mysql               "docker-entrypoint.s…"   27 seconds ago      Up 26 sec
PS C:\Users\WajdyEssam> docker volume ls
DRIVER              VOLUME NAME
local               216496b6fb9658b8616956b6166794e4e750a95c8da3f0df440ba6a09089afdc
local               f76a5662c5cc49cd0707ecebd6153b011e97442334f35c5577ea3647ebbbdc02
PS C:\Users\WajdyEssam> docker volume ls -f dangling=true
DRIVER              VOLUME NAME
local               216496b6fb9658b8616956b6166794e4e750a95c8da3f0df440ba6a09089afdc
PS C:\Users\WajdyEssam> docker volume rm $(docker volume ls -qf dangling=true)
216496b6fb9658b8616956b6166794e4e750a95c8da3f0df440ba6a09089afdc
PS C:\Users\WajdyEssam> docker volume ls
DRIVER              VOLUME NAME
local               f76a5662c5cc49cd0707ecebd6153b011e97442334f35c5577ea3647ebbbdc02
PS C:\Users\WajdyEssam>
```

الآن ما زالت هناك container تعمل وبها أيضاً volume موجودة، فيمكن حذفهم بالطرق السابقة، كما يلي:




## الحذف - حذف ال Images

لحذف ال image يتم استخدام docker rmi image_name ويتم تمرير اسم أو معرف ال Image الذي تريد حذفها:




ولن تستطيع حذف Image مستخدمة بواسطة container ويجب إيقافه أولاً وحذفه قبل حذف ال image المستخدمة. ولحذف جميع ال images يمكن تطبيق الأمر docker rmi $(docker images –q) وسيتم حذفها جميعاً.

ويمكن استخدام الأمر docker image prune لحذف كل ال dangling image، وهي ال Image التي لا Tag واسم لها، وهي تحدث عندما تقوم ببناء ال Images في جهازك بشكل متكرر بنفس ال tag، فقد تجد النسخ القديمة أصبحت بدون تاق. ويمكن حذفها بهذا الأمر.

```
PS C:\Users\WajdyEssam> docker rmi $(docker images -q)
Untagged: solitaire:nginx
Deleted: sha256:4fed1055af776b90b8a2d45861aa9ccfb659b26b90e4c9c4d731f3257ac4e052
Deleted: sha256:367afbf7d19a10efcdb391ed09672ee5146607ad86a16981c3b8c64a1b121457
Untagged: ubuntu:latest
Untagged: ubuntu@sha256:5f4bdc3467537cbbe563e80db2c3ec95d548a9145d64453b06939c45
Deleted: sha256:113a43faa1382a7404681f1b9af2f0d70b182c569aab71db497e33fa59ed87e6
Deleted: sha256:a9fa410a3f1704cd9061a802b6ca6e50a0df183cb10644a3ec4cac9f6421677a
Deleted: sha256:b21f75f60422609fa79f241bf80044e6e133dd0662851afb12dacd22d199233a
Deleted: sha256:038d2d2aa4fb988c06f04e3af208cc0c1dbd9703aa04905ade206d783e7bc06a
Deleted: sha256:b904d425ea85240d6af5a6c6f145e05d5e0127f547f8eb4f68552962df846e81
Deleted: sha256:db9476e6d963ed2b6042abef1c354223148cdcdbd6c7416c71a019ebcaea0edb
Untagged: microsoft/mssql-server-linux:latest
Untagged: microsoft/mssql-server-linux@sha256:baffa99e74fc358a8e9745d59bd9621258
Deleted: sha256:b5a6a300d336e9f43eadc5413efb29d6d2c18d387c94fb75aa045e6a6d84acc3
Deleted: sha256:533223aa511e2fed63beabff315d6137841eaeb0b5c02c164fc311e8c717577b
Deleted: sha256:b2cc8654324a344375946d95d07f04bd114e1df2deebe72bcb1c87b00c4b9ec5
Deleted: sha256:d82caf71d0d861e0b99f7b88b9bd0e308a6da23f933c875a589154528f8f36b4
Deleted: sha256:0486238b750a922a5c60dd0fbbe97cb31a130cd10b57a25c08bb92d5a91318ca
Deleted: sha256:392244f4181714ae68243038100b36857776e0f4016c6170c58431f6d545e7e6
Deleted: sha256:ec1fd849ff0a8f0aa2fd1acc29ad5dabbc79b89f63b74a4f54e31a7b0a100aa1
Deleted: sha256:e3f6dffa20cf36460d23bfb22e17be6e5339891f8537f32db79887caf832048b
Deleted: sha256:c213ffdc9f7032702de5a8e9045fcce2353b7221ef6bf4509e02005cfc858f58
Deleted: sha256:3fddf55a451aa43707518f2d8788c12ee5eb1f1e3075433f5bcf4d445d5c275c
Deleted: sha256:0f5ff0cf6a1c53f94b15f03536c490040f233bc455f1232f54cc8eb344a3a368
Untagged: docs/docker.github.io:latest
Untagged: docs/docker.github.io@sha256:65e609cd7feb8d0e728e454290bed13d8bed7e651
Deleted: sha256:73017ddde35da53be5273030d8db669e28461de0d703cd1883e7cede0f807844
Deleted: sha256:d3dd07e37350bfd677ad5a3d517ad2c8abf7fef51e2565be62687e2d97dbb985
```

## خلاصة

هناك نوعين من ال Volumes:

- الأول لمشاركة الملفات عن طريق Mounting لملفات من النظام الأساسي لل Container
- النوع الثاني Managed Volume وهي التي يتم إنشائها بواسطة الأمر docker volume سواءً للإنشاء أو الحذف وغيرها، وهي كأنه يتم إنشاء قرص صلب hard disk ومن ثم عمل mount له الى ال container.

ال Managed Volume تكون غير متعلقة بال Container فهي تعتبر قرص صلب إضافي ولكن تم عمل mount له لل Container، فاذا تم حذف تلك ال Container فتكون البيانات متواجدة، ويمكن عمل mount لها مجدداً في Container جديد يتم إنشاءه.

وأيضاً البيانات في ال Volume ليست موجودة في ال Image، فمثلاً إذا قمت بإنشاء container ونسخت بيانات الموقع الى ال volume فعند اخذ صورة commit له فلن تحصل على البيانات لأنها لا تعتبر جزءاً من ال image.

لذلك دائماً ضع البيانات الخاصة بالقراءة heavy read-only data في image، وضع البيانات التي ستكتب في ال Volumes مثل قواعد البيانات وغيرها.

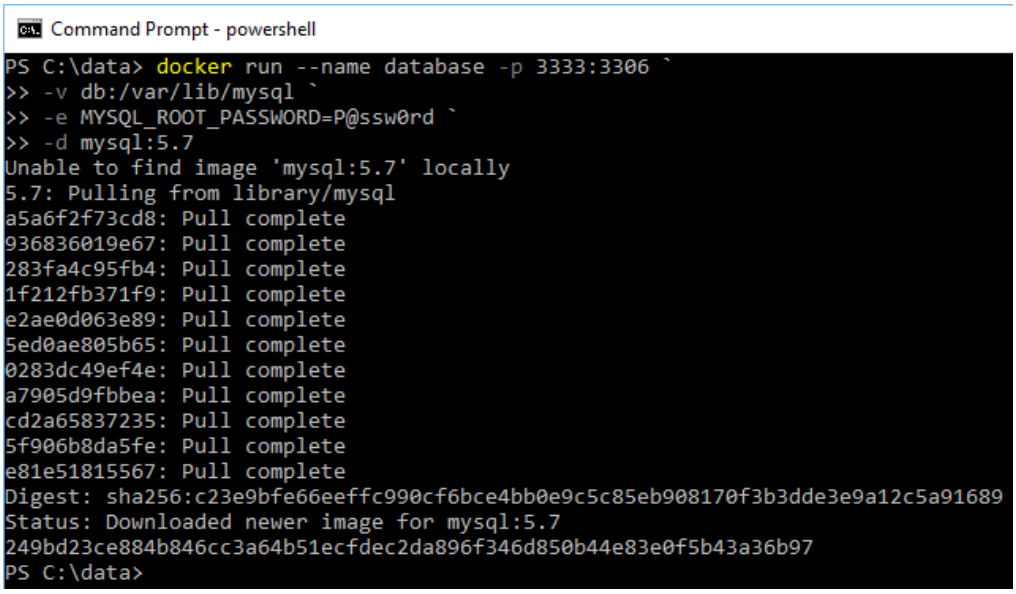
# الفصل السادس: تشغيل البرامج بال Docker Compose

تحدثنا سابقاً عن كيفية تشغيل البرامج وال Containers، وكيف يمكن أن تشغل أحد البرامج بدخل ال Containers التي تعمل، وكيف نقوم بتشغيل ال Web Servers، وأيضاً كيف نتعامل مع قواعد البيانات Databases، في هذا الفصل سوف نقوم بكل هذه الخطوات ونشغل أي خدمة بكامل ملحقاتها موقع أو ويب سيرفس بالإضافة مع قاعدة البيانات وذلك عن طريق ال Docker Compose. بالإضافة الى التعامل مع الشبكات واضافة أي container لأي شبكة تريدها.

## أهمية ال Docker Compose

عندما تبرمج المواقع العادية أو الصغيرة فهي تكون مكونة من قسمين، الموقع نفسه Web Application بالإضافة الى قاعدة البيانات Database، وإذا أردنا تشغيلهم في دوكر فيجب أن نضع كل جزء في Container خاص به، بمعنى Container للموقع، والأخر لقاعدة البيانات، بالطبع يمكن وضعهم بداخل Container واحد ولكن الأفضل تقسيمهم بكل الحالات.

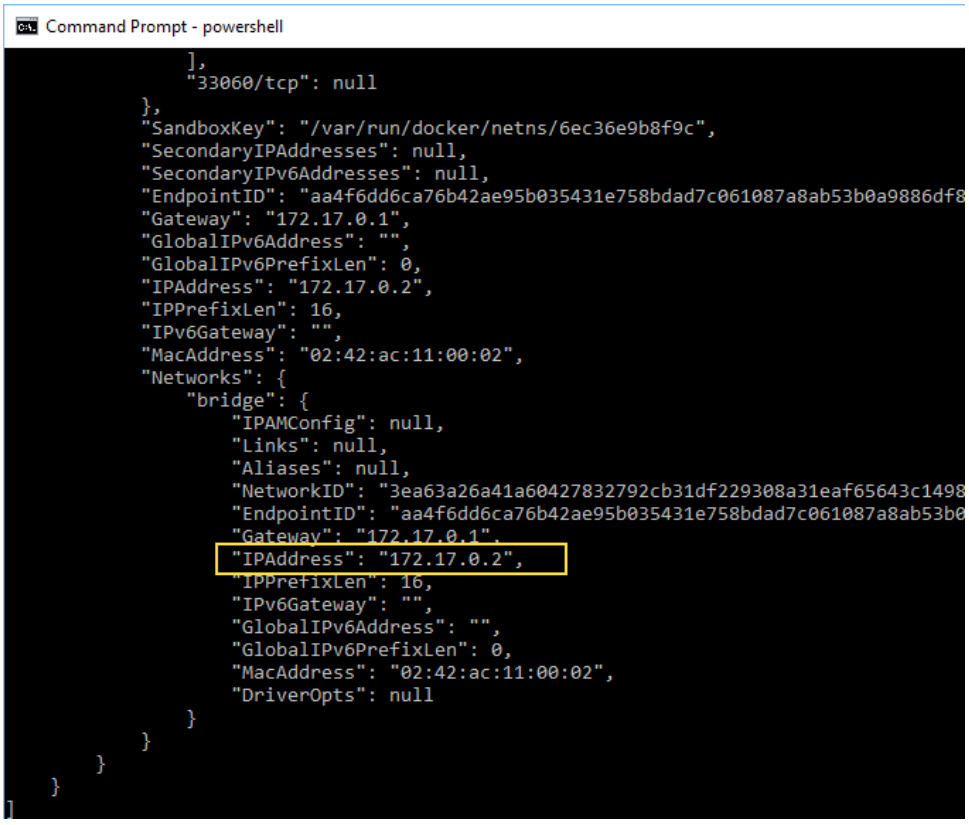
لكي نقوم بهذه الخطوات بالأوامر التي قمنا سابقاً، فيجب أولاً تشغيل قاعدة البيانات ونحدد Volume لها حتى يتم تخزين البيانات، ومن ثم معرفة ال IP الخاص بال Container لأنه نحتاجه لكي نضعه في الموقع ويستطيع التخاطب مع قاعدة البيانات، وبعد ذلك تشغيل الموقع وارسال ال IP الخاص بقاعدة البيانات حتى يستطيع التواصل معها، بالإضافة الى Volume Mapping لكي نشارك ملفات الموقع الذي نريد تشغيله (أو وضعها في ال image مباشرة) الصورة التالية توضح تشغيل قاعدة البيانات:



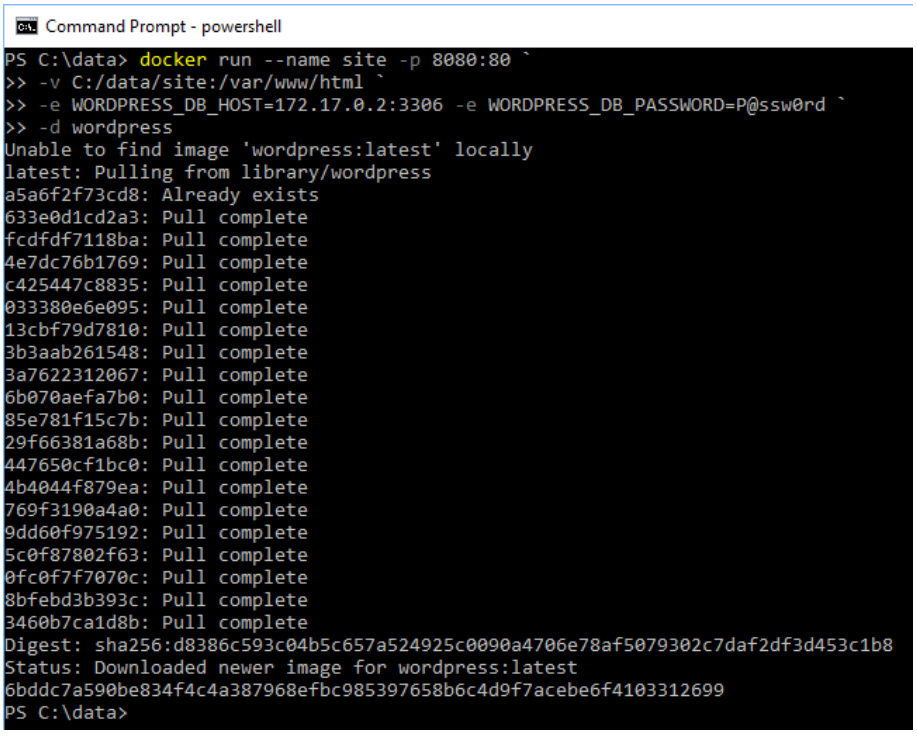

تم تقسيم الأمر من خلال علامة ` في ال PowerShell لعدة اسطر للمقروئية فقط، ويمكن كتابة كامل الأمر في سطر واحد بدون الحاجة للعلامات `:

```
docker run --name database -p 3333:3306 -v db:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=P@ssw0rd -d mysql:5.7
```

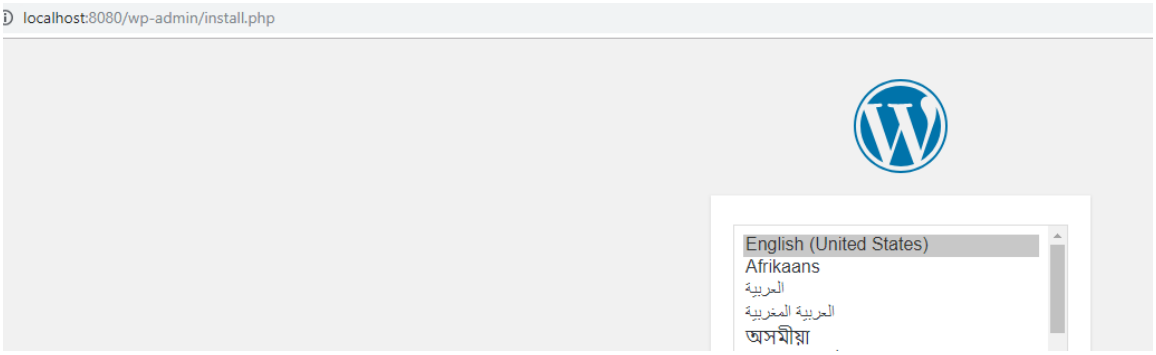
بعد تشغيل قاعدة البيانات قمنا بتشغيل الأمر التالي docker inspect database حتى نحصل على ال IP الخاص بذلك ال Container، ومن ثم تمريره للموقع الذي يستقبل البورت وعنوان قاعدة البيانات حتى يستطيع التواصل معها، ويجب أن تبرمج موقعك بحيث يقرأ القيم المرسلة في ال environment variable ويتصل على أساسها، وإذا لم تكن موجودة فيمكن وضع قيم افتراضية حينها.




بعد ذلك سنقوم بتشغيل الموقع، وسوف نقوم بتشغيل موقع WordPress يتصل بهذه القاعدة، وسوف نحتاج أن نمرر عنوان قاعدة البيانات حتى يستطيع الاتصال بها، كما يلي:

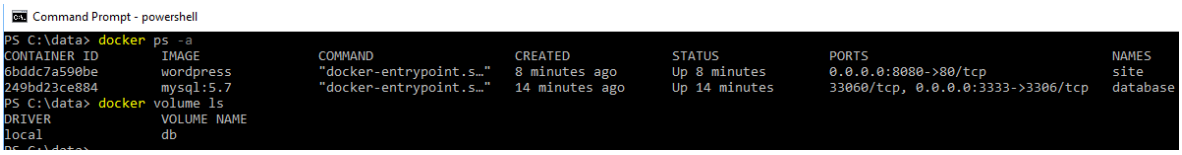


الآن بهذه الخطوات الثلاثة اشتغل الموقع كما يلي:

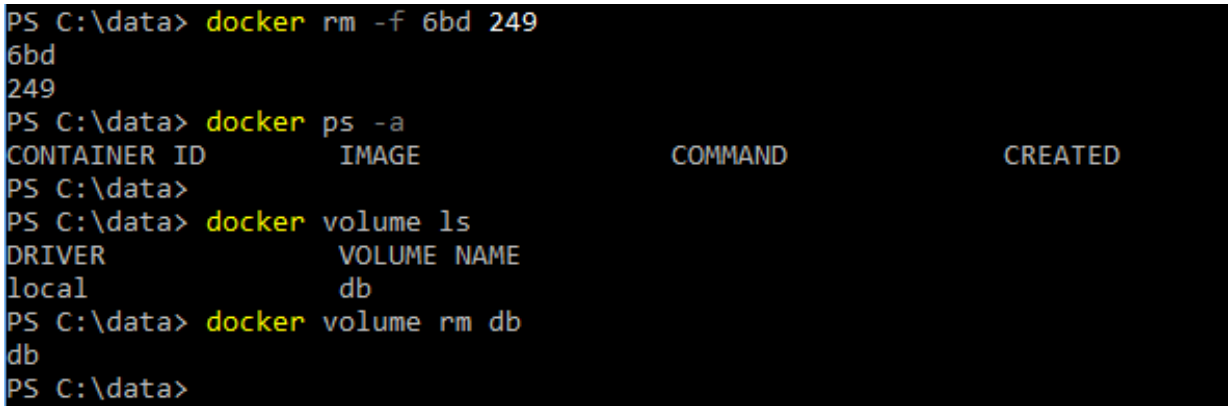



ويمكن أن نضع هذه الأوامر في ملف سكربت مثلاً Bach Script وتشغيله مرة واحدة بدلاً من إعادة هذه الخطوات عند تشغيل الموقع على أي سيرفر، ولكن لا نحتاج لذلك لأن دوكر يوفر Docker-Compose لهذه المهمة وهو تشغيل العديد من ال Containers بدلاً من إعادة كتابتها كل مرة، وال Docker-Compose يتعامل مع ملفات بها الأوامر التي سوف يقوم بتشغيلها بصيغة YML.

سوف نقوم الآن بحذف الموقع وقاعدة البيانات، وأيضاً ستجد في الحذف والايقاف أنك بحاجة لإصدار الأمر لكل منهم على حدة كما في التشغيل:




الصورة التالية تبين إيقاف وحذف البرامج أولاً، ومن ثم حذف قاعدة البيانات:

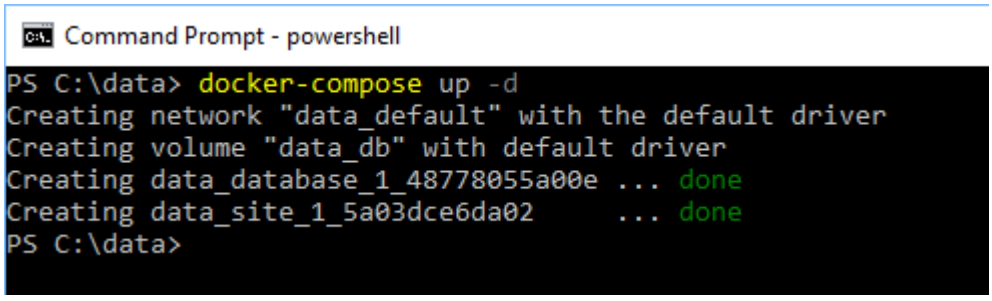



## استخدام ال Docker Compose

بدلاً من القيام بهذه الخطوات الثلاثة لتشغيل الموقع ومعرفة عنوانه الداخلي ومن ثم تشغيل قاعدة البيانات بذلك العنوان، فيمكن أن نستخدم ال Compose File ونضعها في ال yml format وهي نفس البيانات لكن بصيغة Yml، ومن ثم نحفظ الملف docker-compose.yml ونقوم بتشغيل Docker-Compose ونمرر هذا الملف حتى يقوم بتشغيل كل ما فيه.

```yaml
version: "3"

volumes:
    db:

services:
    database:
        image: mysql:5.7
        ports:
            - "3333:3306"
        environment:
            - MYSQL_ROOT_PASSWORD=P@ssw0rd
        volumes:
            - "db:/var/lib/mysql"

    site:
        image: wordpress
        ports:
            - "8080:80"
        environment:
            - WORDPRESS_DB_HOST=database
            - WORDPRESS_DB_PASSWORD=P@ssw0rd
        volumes:
            - "C:/data/site:/var/www/html"
```

ويتم تسمية الملف بالاسم docker-compose.yml ، وبعد ذلك يمكن تشغيل الملف بالذهاب لموقع الملف ومن سطر الأوامر يتم تنفيذ الأمر docker-compose up وسيتم التشغيل لهذين ال Containers.

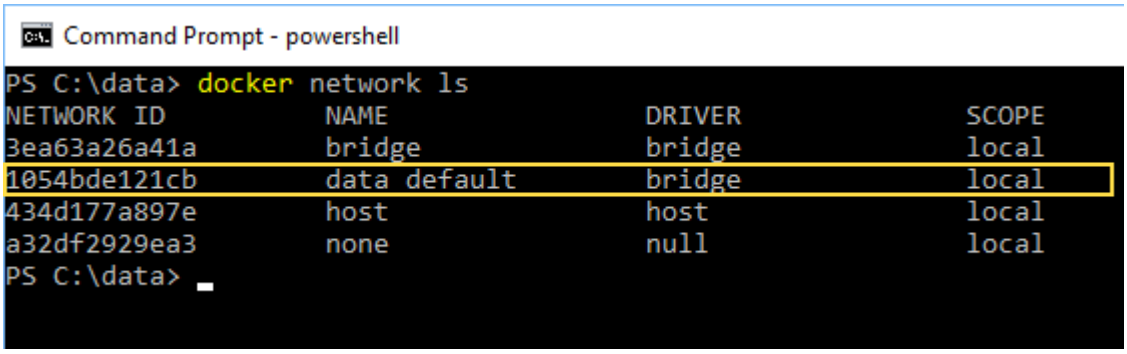



بهذه الطريقة اختصرنا بدل تشغيل ال container الأولى ومن ثم اخذ ال IP لها ، وتشغيل الثانية وارسال ال IP للأولى ، فنقوم بوضع التعليمات داخل ال docker-compose ولاحظ أن ال container الثانية اخذت اسم ال container الأولى وسيتم عمل resolve لها داخلياً لكي يحصل على ال IP. حيث يتم إنشاء شبكة خاصة لهذه البرامج، ويقوم ال DNS Server المدمج بمعرفة ال IP من خلال الاسم، وبالتالي فغالباً عندما تعمل على دوكر وتريد تشغيل أي مشروع مكون من عدة أجزاء فسوف تتعامل مباشرة مع ال docker-compose file ويتم تشغيل هذه البرامج مباشرة.

الشبكة الخاصة التي تم إنشائها هي بالاسم data_default وهي تكون باسم المجلد الذي تعمل به، وبعدها يأتي default_. وهذا الاسم سوف يسبق كل من ال volumes وال containers لكي تدل على أنها بهذا المشروع.

وميزة الشبكة الخاصة أن كل ال containers بها تستطيع التواصل مع الأخرى من خلال الاسم، ولا تستطيع أي containers أخرى من شبكة أخرى التواصل مع هذه ال containers، الا فقط من تكون بداخل الشبكة.

الآن الموقع أصبح يعمل الآن، وسوف نقوم بالتعامل مع هذه الشبكة، وسوف نقوم بطباعة كافة الشبكات المتواجدة:




سوف نقوم بفحص هذه الشبكة من خلال الأمر inspect كما يلي:

وسوف تجد قائمة البرامج بالإضافة الى عنوان كل منهم. وتستطيع في دوكر أن تقوم بإنشاء شبكة، إضافة container لهذه الشبكة، أو إزالة container من هذه الشبكة، أو مسح الشبكة ككل.

من المهم ملاحظة عندما قمنا بتشغيل البرامج يدوياً فهي انضمت تلقائياً للشبكة bridge، بينما في حالة ال docker compose فهو يقوم بإنشاء شبكة خاصة لكل البرامج المتواجدة فيه، وهذا أيضاً من الفروقات المهمة بين التشغيل اليدوي أو استخدام ال compose.

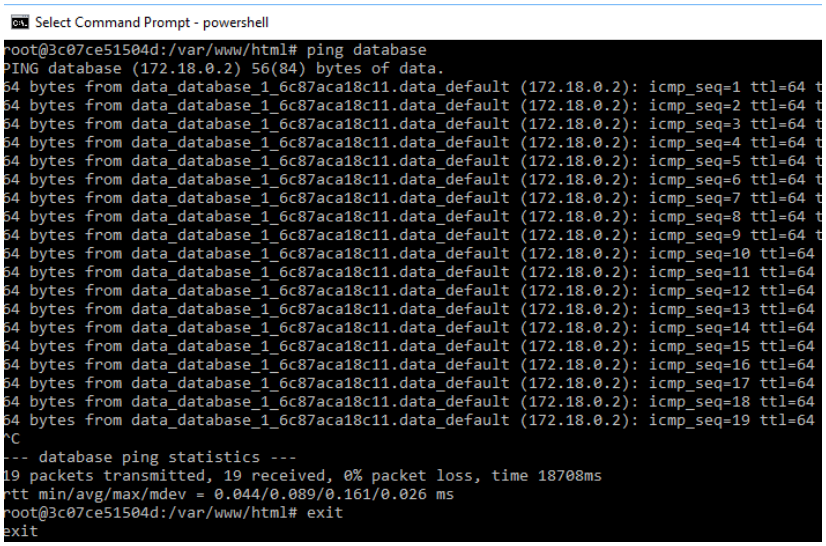
سوف نقوم الآن بالدخول الى أحد ال containers ولاحظ في الصورة التالية استخدمنا الطريقة التقليدية أولاً، وبعدها قمنا بالخروج والدخول بطريقة أخرى باستخدام docker-compose ولن نعد نحتاج الى استخدام ال it– فهي تلقائية في هذه الحالة:

```
Command Prompt - powershell
PS C:\data> docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
3c07ce51504d        wordpress           "docker-entrypoint.s…"   17 minute
443a1c2096fd        mysql:5.7           "docker-entrypoint.s…"   17 minute
c11
PS C:\data> docker exec -it 3c bash
root@3c07ce51504d:/var/www/html# exit
exit
PS C:\data>
PS C:\data> docker-compose exec site bash
```

الآن نريد أن نتأكد هناك اتصال بين ال containers وأنه يمكن أن تقوم أي منهم بعمل ping للأخرى، وسوف نقوم أولاً بتحميل برنامج ال ping لأنه لا يتواجد في نسخ دوكر:

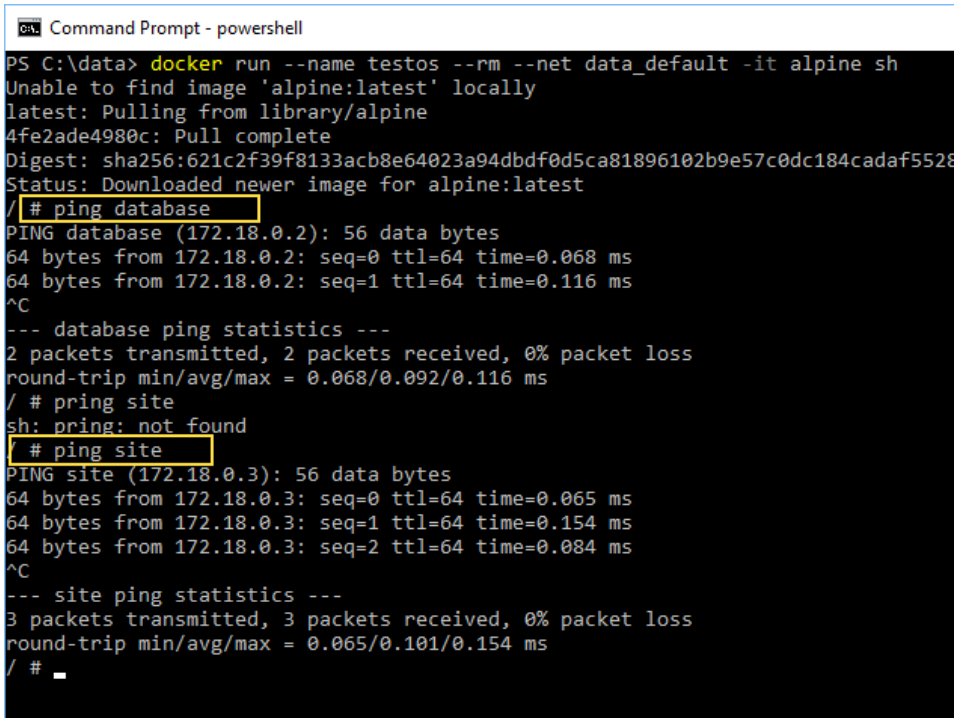
```
Command Prompt - powershell
root@3c07ce51504d:/var/www/html# ping database
bash: ping: command not found
root@3c07ce51504d:/var/www/html# apt-get update
Ign:1 http://cdn-fastly.deb.debian.org/debian stretch InRelease
Get:2 http://cdn-fastly.deb.debian.org/debian stretch-updates InRelease [91.0 kB]
Get:3 http://security-cdn.debian.org/debian-security stretch/updates InRelease [9
Get:4 http://cdn-fastly.deb.debian.org/debian buster InRelease [150 kB]
Get:6 http://security-cdn.debian.org/debian-security buster/updates InRelease [3
Get:5 http://cdn-fastly.deb.debian.org/debian buster-updates InRelease [47.6 kB]
Get:8 http://security-cdn.debian.org/debian-security stretch/updates/main amd64 P
Get:7 http://cdn-fastly.deb.debian.org/debian stretch Release [118 kB]
Get:9 http://cdn-fastly.deb.debian.org/debian stretch-updates/main amd64 Packages
Get:10 http://cdn-fastly.deb.debian.org/debian buster/main amd64 Packages [7731 k
Get:11 http://cdn-fastly.deb.debian.org/debian stretch Release.gpg [2434 B]
Get:12 http://cdn-fastly.deb.debian.org/debian stretch/main amd64 Packages [7089
Fetched 15.8 MB in 4s (3858 kB/s)
Reading package lists... Done
root@3c07ce51504d:/var/www/html# apt-get install iputils-ping
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libcap2 libcap2-bin libpam-cap
The following NEW packages will be installed:
  iputils-ping libcap2 libcap2-bin libpam-cap
0 upgraded, 4 newly installed, 0 to remove and 0 not upgraded.
Need to get 114 kB of archives.
After this operation, 290 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://cdn-fastly.deb.debian.org/debian stretch/main amd64 libcap2 amd64 1
```

بعد ذلك سوف نقوم بعمل ال ping ونتصل بال database container:



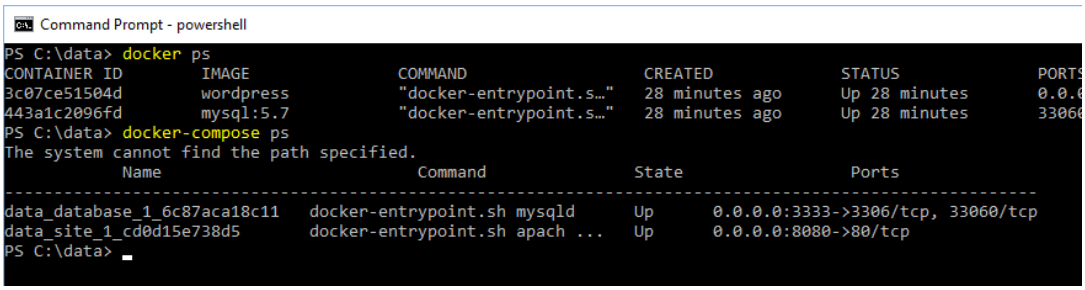

ونفس الأمر لو قمت بذلك من ال container الأخرى سوف تستطيع التخاطب مع الأولى.

سوف نقوم الآن بتشغيل container جديدة منفصلة ومن ثم نربطها مع هذه الشبكة التي تمت إنشاءها مع المشروع، كما يلي:



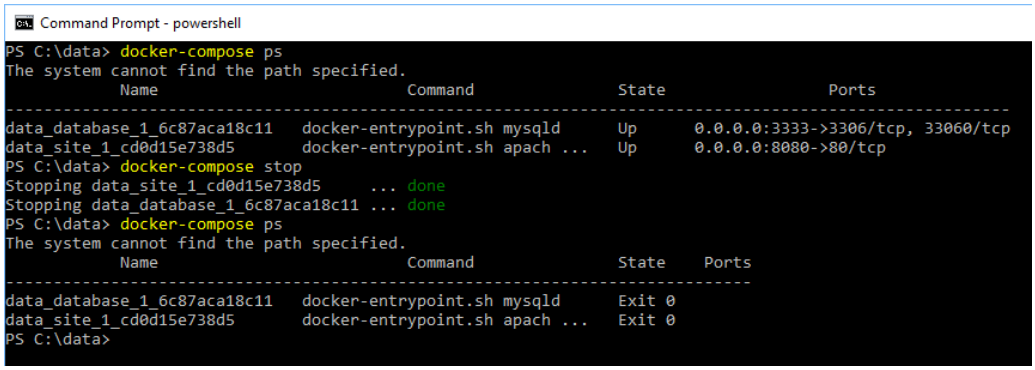

وهكذا كل من ال containers أصبحت بداخل الشبكة (بالرغم من أن الأخيرة لم تكن ضمن بال compose file).

سوف نقوم الآن بعرض كافة ال containers باستخدام docker ps ومن ثم عرض ال containers التي تعمل في هذه ال compose كما يلي:



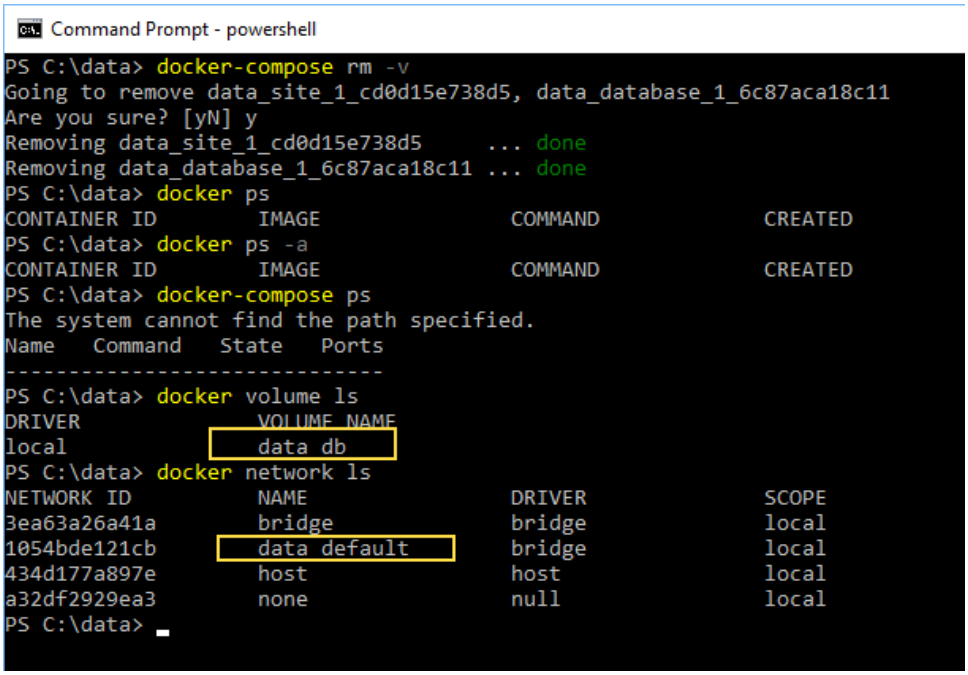

ويمكن أن نقوم بتشغيل أي من ال container بداخل ال compose أو ايقافها باستخدام docker-compose stop name أو docker-compose start name وتحدد ال container الذي تريده إيقافه أو تشغيله.

سنقوم الآن بإيقاف كل ال containers وذلك من خلال docker-compose stop بدون تحديد أي من الأسماء، كما يلي:

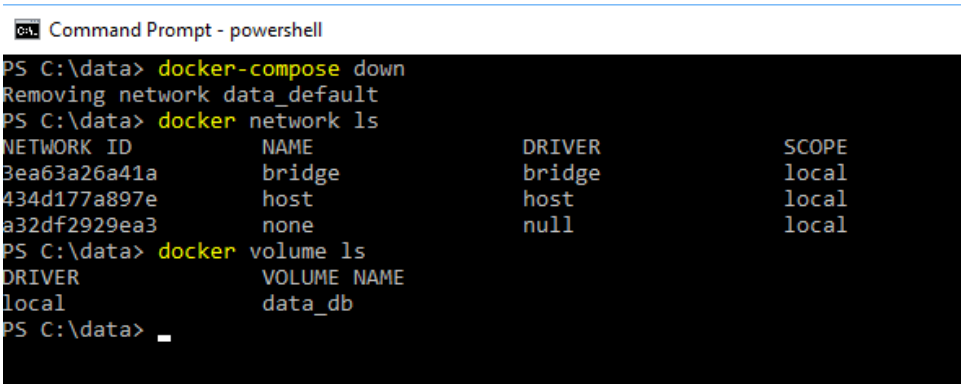



ويمكن تشغيلها مجدداً بالأمر start.

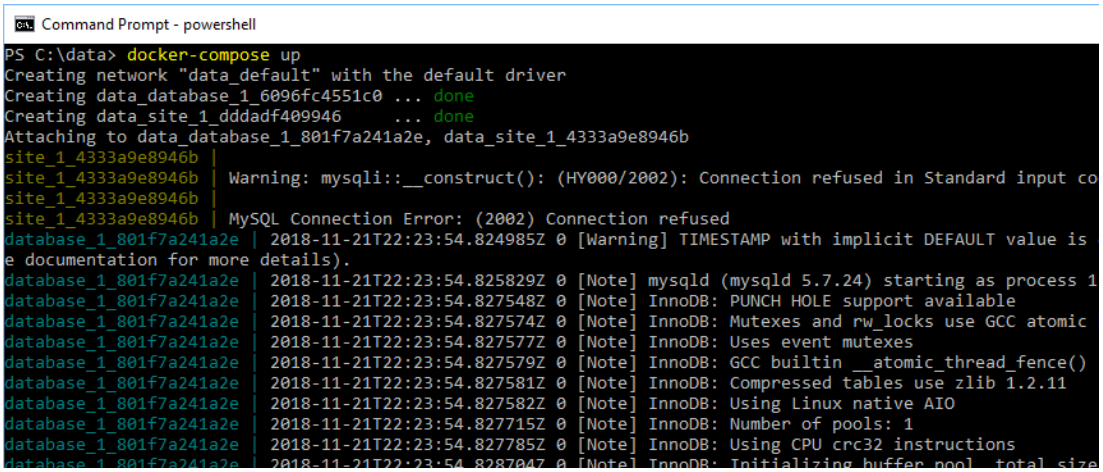
سنقوم الآن بحذف هذه ال containers مع ال volumes التي لم تسمى (لم يتم تحديد أي اسم لها) بها كما يلي:



ويمكن أن نقوم بعمل down وسوف يتم حذف الشبكة كما يلي:




الآن يمكنك حذف ال volume يدوياً أن اردت، وإذا اردت أن تقوم باسترجاع كل شيء مع بياناتك، فسوف تقوم بعمل Up:




لاحظ لم نقم بتشغيله بالوضع d– ولذلك ظهرت المخرجات.

إذا اردت أن تقوم بحذف ال named volumes فيمكنك استخدام docker-compose down –v وسوف يقوم بإيقاف كل شيء وحذف كل ال volumes المرتبطة بها.

## خلاصة

ال Dockerfile يستخدم لتسهيل Automation لبناء ال Images بدلاً من تشغيل Container ومن ثم نسخ الملفات ومن ثم commit، وال Docker-Compose هو تسهيل Automation لإنشاء ال Containers بدلاً من ادخال الأوامر واحداً تلو الأخر.

# خاتمة

وصلنا لنهاية هذا الكتاب، أرجوا أن تكون قد استفدت منه ولو شيئاً يسيراً، واعذرونا على التقصير والأخطاء إن وجدت، وإذا كان لديك أي سؤال او استفسار او مشكلة ما أو تحتاج لاستشارة فيمكنك التواصل معي عبر الحسابات التالية:

Email: wajdyessam@hotmail.com
Twitter: @wajdyessam
Website: https://informatic-ar.com
Questions: https://io.hsoub.com/docker

تابعنا عبر موقع انفورماتيك وصفحاتنا الاجتماعية للمزيد من المواضيع والكتب الجديدة بإذن الله:

https://twitter.com/informatic_arab
https://www.facebook.com/informatics.arab/

# في أمان الله

وجدي عصام عبدالرحيم
**2018-11-24**