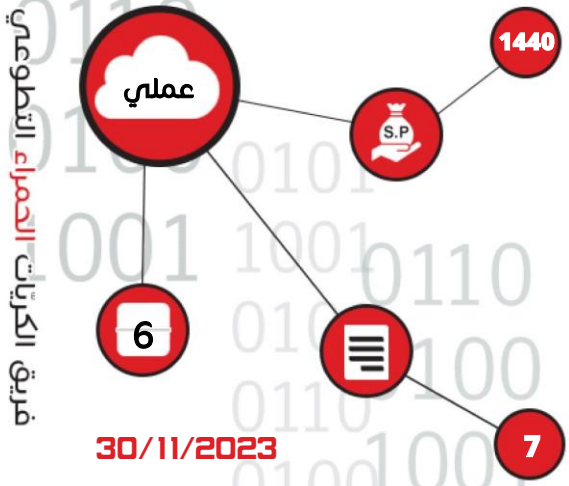


Docker

عملي مشترك

30/11/2023

RB Informatics;

نظم تشغيل (1)

Virtualization

- سمي الserver بالخادم لأنه يخدم الموارد لأجهزة الكمبيوتر عبر الشبكة، وهناك أنواع عديدة له تسمى باسم ما يفعله كخادم التطبيقات وخادم البريد الالكتروني.

- إذا ما هو ال physical server؟

- هو piece of hardware آلة يمكن التعرف عليها مع وحدة المعالجة المركزية والذاكرة واللوحة الأم وما شابه ولا تحتوي الخوادم المادية على فجوة بين الأجهزة الحاسوبية ونظام التشغيل، وقد يقوم الخادم الفعلي بتشغيل Linux او Windows لكنه سيعمل على تشغيل نظام تشغيل واحد فقط في حالة واحدة.

لنعود للوراء قليلا كيف بدأوا مسبقا؟!

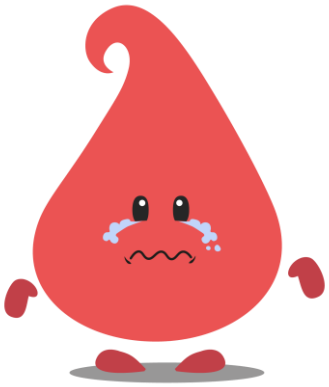
يشترون Server ويكون عليه Application يحتاج مثلا 8RAM ويكون مكلف جدا ولا يأخذ الا مساحة قليلة جدا.

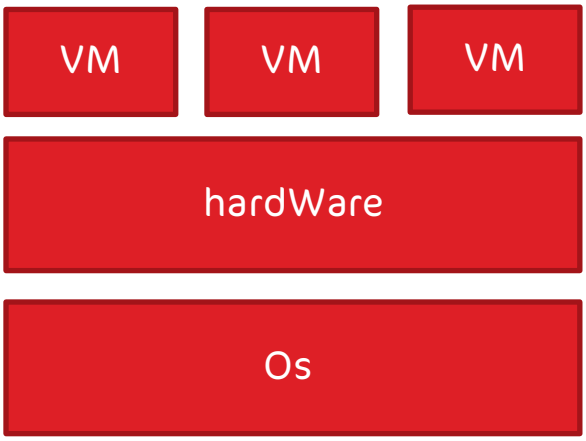
ولو نفس الشركة أرادت انشاء App جديد تشتري Server آخر وترفع عليه التطبيق بالتالي أصبح لدينا هدر وتكاليف كبيرة جداً فما الحل؟!!

- اخترع ما يسمى Virtualization
- وهذا يعني تقسيم الـ Server لـ computers أي تشغيل أنظمة تشغيل متعددة على نظام كمبيوتر في وقت واحد، و كل كمبيوتر له الخدمات الخاصة فيه، لكن جميعهم يعودون لنفس المرجع.
- توفر المحاكاة الافتراضية أيضا القدرة على تشغيل أنظمة تشغيل مختلفة، والفكرة الأهم انها توفر طريقة لتقسيم نظام كبير الى العديد من الأجزاء مما يسمح باستخدام الخادم بشكل أكثر كفاءة من قبل عدد من المستخدمين او التطبيقات المختلفة ذات الاحتياجات المختلفة كما يسمح بالعزل و الحفاظ على تشغيل البرامج داخل جهاز افتراضي في مأمن من العمليات التي تجري في جهاز افتراضي آخر على نفس المضيف

- إذا كيف تتم عملية التشغيل؟؟

الـ Hypervisor هو المسؤول أو المشرف وهو software يتيح لي عملية التقسيم والمعروف باسم VMM (virtual machine monitor) وهو يقوم بإنشاء وتشغيل الأجهزة الافتراضية وهو يسمح لـ one host computer لدعم عدة guest VMs المتشاركة في المصدر





- لو أدرت تشغيل نظام ثاني فوق نظامي الأساسي هنا تأتي فكرة الـ hypervisor وبالتالي هو صلة تخاطب بين الvm وال hardware
- في الرسمة التي في الأعلى يأخذ نظام os كل شيء يخص الـ hardware لكن بعد تنزيل ال vm ستأخذ وتستعير مساحة من الـ os عن طريق ال hypervisor

من يجعل الـ VM تصل لل Source الخاص بها؟

الـ Hypervisor

ويقوم بعمل عزل isolation بين ال VM بمعنى أنه إذا تعطلت أحدهم لا تتأذى الأخرى.

Hypervisor types

1. النوع الأول فهو الأكثر شيوعا يحل محل host's operating system و يقع أعلى الجهاز مباشرة و لهذا السبب تسمى (برامج مراقبة الأجهزة المضمنة) embedded hypervisors

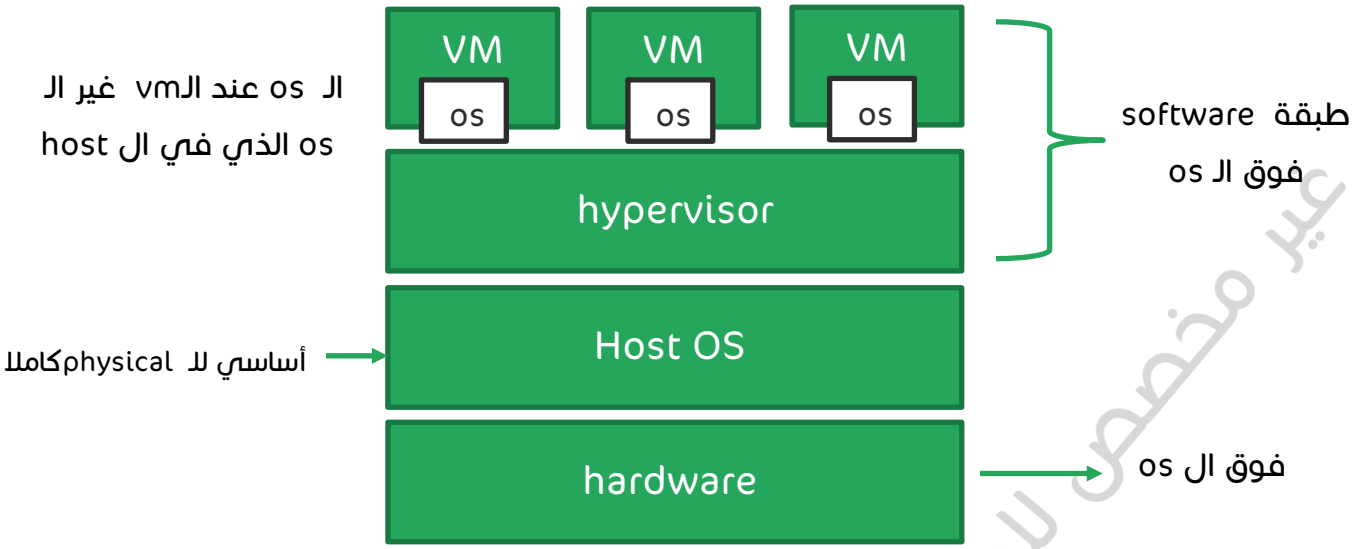
Type 1 Hypervisor Examples:
- VMware hypervisors like vSphere, ESXi and ESX
- Microsoft Hyper-V ,Oracle VM Server ,Citrix Hypervisor

2. النوع الثاني يعمل كبرنامج على نظام التشغيل/النظام O/S و الذي بدوره يعمل على physical hardware يستخدم عادة لتشغيل أنظمة متعددة على جهاز كمبيوتر واحد مثل تمكين المستخدم من التمهيد الى Linux او Windows

Type 2 Hypervisor Examples:
- VMware Workstation
- VMware Fusion
- Oracle VirtualBox
- Oracle Solaris Zones
- Oracle VM Server for x86

"Plans are only good intentions unless they immediately degenerate into hard work."

Type 2:



What is the virtual machine?

- هو compute resource يستخدم الـ software بدلا من الكمبيوتر الفعلي لتشغيل البرامج والتطبيقات
- يعمل جهاز 'guest' افتراضي واحد أو أكثر على جهاز host فعلي
- يقوم كل جهاز افتراضي بتشغيل نظام التشغيل الخاص به ويعمل بشكل منفصل عن الأجهزة الافتراضية الأخرى، حتى عندما تعمل جميعها على نفس المضيف. هذا يعني أنه، على سبيل المثال، يمكن تشغيل جهاز افتراضي MacOS افتراضي على جهاز كمبيوتر فعلي.

يمكن للأجهزة الافتراضية تشغيل بيئات أنظمة تشغيل متعددة على جهاز كمبيوتر فعلي واحد، مما يوفر المساحة الفعلية والوقت وتكاليف الإدارة

الفوائد الخاصة بالـ VM:

1. Faster provision
2. Resource efficient
3. Downtime

CONTAINERIZATION

الحاويات هي alternative خفيف الوزن للمحاكاة الافتراضية. يتضمن ذلك تغليف تطبيق في حاوية ببيئة تشغيل خاصة به. وبالتالي، بدلا من تثبيت نظام تشغيل لكل جهاز افتراضي، تستخدم الحاويات نظام التشغيل المضيف.

- كل container عبارة عن حزمة برامج قابلة للتنفيذ تعمل فوق نظام التشغيل المضيف. يمكن للمضيف دعم العديد من container بشكل متزامن في وقت واحد.
- هذه container محمولة ويمكن استخدامها على أي بنية تحتية في أي بيئة تدعم تقنية الحاوية، مثل Docker وKubernetes
- فصلت الـ container عن الـ host os عن نظام التشغيل كامل
- بال container لا وجود لـ hypervisor

الخلاصة:

- **Physical Server**: لدي ال application في حالة عمل و آخذ مساحة صغيرة و الباقي مهمل
- أما ال **hypervisor**: استخدمنا المساحة و أشعلنا أكثر من تطبيق لكن كل vim لها تطبيق واحد، كل vim لها نظام
- أما بال **container**: أصبح vim الواحدة او ال computer الواحد يشغل أكثر من application و حجمه خفيف

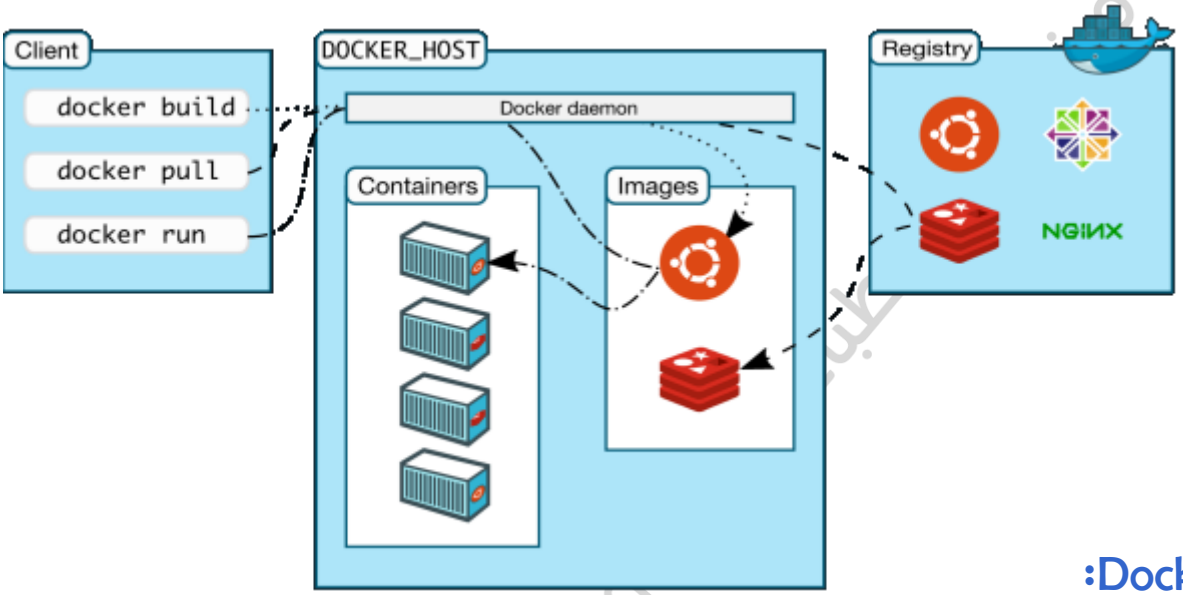
VIRTUALIZATION VS CONTAINERIZATION

	Virtualization	Containerization
Isolation	يوفر عزلا تاما عن نظام التشغيل المضيف و other VMs	يوفر عادة عزلا خفيفا من المضيف والحاويات الأخرى، ولكنه لا يوفر حدودا أمنية قوية مثل VM
Operating System	يعمل على نظام تشغيل كامل بما في ذلك kernel، وبالتالي يتطلب المزيد من موارد النظام مثل وحدة المعالجة المركزية والذاكرة والتخزين	يقوم بتشغيل جزء وضع المستخدم من نظام التشغيل، ويمكن تخصيصه ليحتوي فقط على الخدمات المطلوبة لتطبيقك باستخدام موارد نظام أقل
Guest Compatibility	يعمل على أي نظام تشغيل داخل VM	يعمل على نفس إصدار نظام التشغيل مثل host
Networking	يستخدم محولات الشبكة الافتراضية	يستخدم طريقة عرض معزولة لمحول شبكة افتراضية. وبالتالي، يوفر محاكاة افتراضية أقل قليلا
Deployment	نشر VM الفردية باستخدام برنامج Hypervisor	نشر حاويات فردية باستخدام Docker أو نشر حاويات متعددة باستخدام منسق مثل Kubernetes
Persistent Storage	استخدام قرص ثابت افتراضي (VHD) للتخزين المحلي لجهاز افتراضي واحد أو مشاركة ملف Server Message Block (SMB) للتخزين المشترك بواسطة خوادم متعددة	استخدام الأقراص المحلية للتخزين المحلي لعقدة واحدة أو SMB للتخزين المشترك بين عقد أو خوادم متعددة
Load Balancing	تتم موازنة تحميل الجهاز الافتراضي عن طريق تشغيل الأجهزة الافتراضية في خوادم أخرى في مجموعة تجاوز الفشل	يمكن للمنسق بدء تشغيل الحاويات أو إيقافها تلقائيا على عقد نظام المجموعة لإدارة التغييرات في التحميل والتوافر

Docker

container tool لتشغيل الـ container الخاصة بك

- Docker عبارة عن نظام أساسي برمجي يسمح لك ببناء التطبيقات واختبارها وتعبئة البرامج في وحدات قياسية تسمى container. التي تحتوي على كل ما يحتاجه البرنامج للتشغيل بما في ذلك المكتبات وأدوات النظام والتعليمات البرمجية ووقت التشغيل



بنية Docker:

- Docker client: (عامل الإرساء) هو الطريقة الأساسية التي يتفاعل بها العديد من مستخدمي Docker مع Docker. عند استخدام أوامر مثل docker run، يرسل العميل هذه الأوامر إلى dockerd، الذي ينفذها. يستخدم أمر Docker واجهة برمجة تطبيقات Docker (Docker API). ويمكن لعميل Docker التواصل مع أكثر من برنامج خفي واحد.
- Docker host: يوفر مضيف Docker بيئة كاملة لتنفيذ التطبيقات وتشغيلها. وهو يتألف من برنامج Docker الخفي والصور والحاويات والشبكات والتخزين. كما ذكرنا سابقا، فإن البرنامج الخفي مسؤول عن جميع الإجراءات المتعلقة بالحاوية ويتلقى الأوامر عبر CLI أو REST API. كما يمكنه التواصل مع other daemons لإدارة خدماته
- Docker daemon: يستمع برنامج Docker الخفي (dockerd) إلى طلبات Docker API ويدير objects Docker مثل الصور والحاويات والشبكات ووحدات التخزين. يمكن للبرنامج الخفي (A daemon) أيضًا التواصل مع برامج خفيّة أخرى (other daemons) لإدارة خدمات Docker.
- Docker registry: يقوم سجل Docker بتخزين Docker images.
- Docker Hub: هو سجل عام يمكن لأي شخص استخدامه، ويتم تكوين Docker للبحث عن الصور على Docker Hub افتراضيا. يمكنك حتى تشغيل السجل الخاص بك.

عند استخدام أوامر سحب docker pull أو تشغيل docker run، يتم سحب الصور المطلوبة من السجل الذي تم تكوينه. عند استخدام الأمر docker push، يتم دفع صورتك إلى السجل الذي تم تكوينه.

DOCKER OBJECTS

- Image: الصورة عبارة عن قالب للقراءة فقط يحتوي على إرشادات لإنشاء حاوية Docker. في كثير من الأحيان، تعتمد الصورة إلى صورة أخرى، مع بعض التخصيصات الإضافية.

على سبيل المثال، يمكنك إنشاء صورة تعتمد على صورة ubuntu، ولكن تقوم بتثبيت خادم الويب Apache والتطبيق الخاص بك، بالإضافة إلى تفاصيل التكوين اللازمة لتشغيل التطبيق الخاص بك.

- يمكنك إنشاء صورك الخاصة أو يمكنك فقط استخدام الصور التي أنشأها الآخرون والمنشورة في السجل. لإنشاء صورتك الخاصة، يمكنك إنشاء Dockerfile بصيغة بسيطة لتحديد الخطوات اللازمة لإنشاء الصورة وتشغيلها. كل تعليمة في Dockerfile تنشئ طبقة في الصورة. عندما تقوم بتغيير Dockerfile وإعادة إنشاء الصورة، يتم إعادة إنشاء تلك الطبقات التي تم تغييرها فقط. هذا جزء مما يجعل الصور خفيفة الوزن وصغيرة وسريعة، عند مقارنتها بتقنيات المحاكاة الافتراضية الأخرى.

CONTAINER

- container: هي نسخة قابلة للتشغيل من الصورة. يمكنك إنشاء حاوية أو تشغيلها أو إيقافها أو نقلها أو حذفها باستخدام Docker API أو CLI. يمكنك توصيل container بشبكة واحدة أو أكثر، أو إرفاق وحدة تخزين بها، أو حتى إنشاء صورة جديدة بناءً على وضعها الحالي.
- في الوضع الافتراضي، تكون الحاوية معزولة بشكل جيد نسبيًا عن الحاويات الأخرى والجهاز المضيف لها. يمكنك التحكم في مدى عزل (شبكة الحاوية أو وحدة تخزينها أو غيرها من الأنظمة الفرعية المضمنة) عن الحاويات الأخرى أو عن الجهاز المضيف.
- يتم تعريف الحاوية من خلال صورتها بالإضافة إلى أي خيارات تكوين تقوم بتوفيرها لها عندما تنشئها أو تبدأ بتشغيلها. عند إزالة حاوية، تختفي أي تغييرات في حالتها اذا كانت غير مخزنة في وحدة التخزين الدائمة.

NETWORKING

Docker networking هي ممر تتواصل من خلاله جميع الحاويات المعزولة.

هناك خمسة برامج تشغيل للشبكة بشكل أساسي في docker:

- Bridge: إنه برنامج تشغيل الشبكة الافتراضي للحاوية. يمكنك استخدام هذه الشبكة عند تشغيل تطبيقك على حاويات مستقلة، أي حاويات متعددة تتواصل مع نفس مضيف docker.
- Host: يقوم برنامج التشغيل هذا بإزالة عزل الشبكة بين حاويات Docker ومضيف Docker. يمكنك استخدامه عندما لا تحتاج إلى أي عزل للشبكة بين المضيف والحاوية.
- Overlay: تمكن هذه الشبكة خدمات السرب swarm services من التواصل مع بعضها البعض. يمكنك استخدامه عندما تريد تشغيل الحاويات على مضيفي Docker مختلفين أو عندما تريد تشكيل خدمات سرب بواسطة تطبيقات متعددة.

- Macvlan: يقوم برنامج التشغيل هذا بتعيين عنوان mac للحاويات لجعلها تبدو وكأنها أجهزة فعلية. يقوم بتوجيه حركة المرور بين الحاويات من خلال عناوين mac الخاصة بهم. يمكنك استخدام هذه الشبكة عندما تريد أن تبدو الحاويات كجهاز فعلي، مثال: أثناء ترحيل إعداد الجهاز الافتراضي (migrating a VM setup).

STORAGE

Storage: يمكنك تخزين البيانات داخل writable layer من الحاوية ولكنها تتطلب برنامج تشغيل تخزين. كونها غير ثابتة، فإنها تهلك عندما لا تعمل الحاوية. إضافة إلى أنه ليس من السهل نقل هذه البيانات.

فيما يتعلق بالتخزين المستمر، يقدم Docker العديد من الخيارات:

- Docker volume: هي التقنية الأكثر استخداما للتخزين الدائم لبيانات الحاوية. تتم إدارة وحدة تخزين Docker بواسطة Docker نفسها ولديها نظام ملفات مخصص على المضيف، ولا يعتمد على بنية نظام الملفات على المضيف. تتم إدارة وحدات تخزين Docker بشكل صريح عبر Docker command line ويمكن إنشاؤها بمفردها أو أثناء تهيئة الحاوية. عند إيقاف حاوية أو حذفها، تظل Docker volume مخزن بشكل دائم.
- Docker bind mount: هو خيار التخزين الدائم الثاني ولكن مع خيارات محدودة أكثر من Docker volume. لا يمكن إدارته عبر Docker CLI ويعتمد كليا على توفر نظام الملفات الخاص بالمضيف. يمكن إنشاء نظام ملفات مضيف عند تشغيل حاوية. Bind mounts هي نوع من مجموعة شاملة من وحدات التخزين (مسماة أو غير مسماة)
- Tmpfs : هو خيار تخزين ثالث غير دائم مثل Docker volume أو bind mount. تتم كتابة البيانات مباشرة على ذاكرة المضيف وحذفها عند إيقاف الحاوية. مفيد جدا عندما يتعلق الأمر ببيانات حساسة لا تريدها أن تكون دائمة. الفرق المهم حقا هو أن الحاويات لا يمكنها مشاركة مساحة tmpfs إلا إذا كانت تعمل على نظام التشغيل Linux. يتم استخدام علامتين عند إنشاء tmpfs volume: tmpfs و mount. يعد Mount flag أحدث ويدعم خيارات متعددة أثناء بدء تشغيل الحاوية. تتم كتابة أنظمة الملفات المؤقتة إلى ذاكرة الوصول العشوائي (أو إلى ملف المبادلة الخاص بك إذا كانت ذاكرة الوصول العشوائي ممتلئة) وليس إلى المضيف أو طبقة نظام الملفات الخاصة بالحاوية في Docker tmpfs: Docker.com.
- Storage Plugins: توفر Storage Plugins القدرة على الاتصال بمنصات التخزين الخارجية. تقوم هذه المكونات الإضافية بتعيين التخزين من المضيف إلى مصدر خارجي مثل storage array or an appliance.

انتهت المحاضرة

وانتهى مقرر العملي

بالتوفيق