Docker 101

دوكر 101

احمد سمير الغامدي

ahmed-sameer.com

linkedin.com/in/ahmed-alghamdi0

مقدمة في دوكر

ما هو Docker ؟

Docker هو برنامج يساعد المطورين في تشغيل تطبيقاتهم في بيئات معزولة تسمى "**حاويات**" (**Containers**), وميزتها تضمن أن التطبيق **يعمل بنفس الطريقة على أي جهاز أو سيرفر**, بغض النظر عن الاختلافات في الإعدادات أو الأنظمة.

ليش نستخدم Docker؟

- **توحيد بيئة التشغيل**: بدل ما نواجه مشاكل "يشتغل عندي وما يشتغل عندك", دوكر يضمن أن التطبيق يشتغل بنفس الشكل في كل مكان.
- **إدارة الإصدارات بسهولة**: تقدر تشغل نسخ مختلفة من نفس البرنامج بدون تعارض. مثلاً, لو عندك مشروع يحتاج MySQL 5.7 ومشروع ثاني يحتاج MySQL 8, تقدر تشغل النسختين في نفس الوقت بدون مشاكل.
- **توفير الموارد**: الحاويات أخف من الأجهزة الافتراضية (Virtual Machines), فممكن تشغل عدد كبير منها على نفس الجهاز بدون استهلاك كبير للموارد.

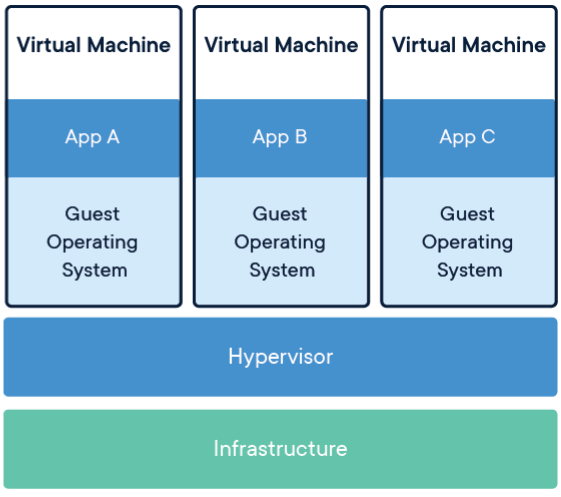
اساسيات Docker

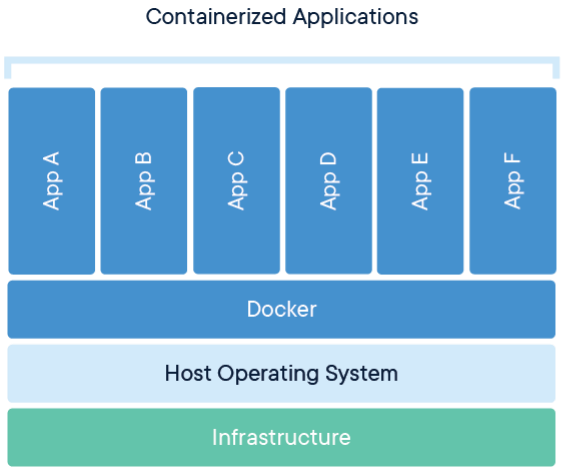
إذا اردت الاستفادة من Docker لأقصى حد, لازم تفهم كيف يشتغل بشكل أساسي.

الحاويات (Containerization)

الحاويات هي جوهر Docker, لأنها تعزل التطبيقات بحيث كل تطبيق يكون مستقل بملفاته ومتطلباته, ولا يتأثر بالبرامج الثانية على نفس الجهاز.

مقارنتا بالـ VMs الحاويات **أخف وأسرع**, لأنها لا تحتاج نظام تشغيل كامل, بس تستخدم نفس نواة (Kernel) في النظام الأساسي.





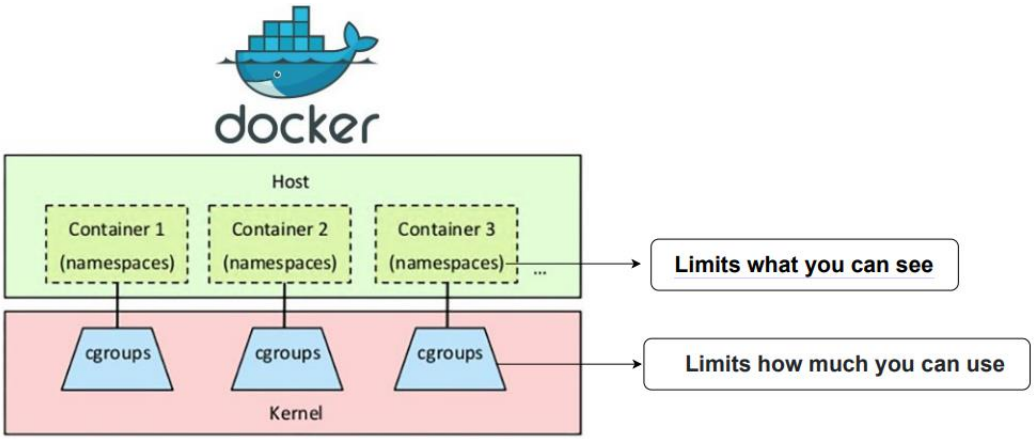
اعداد\ احمد سمير الغامدي

كيف يستغل Docker؟

Docker يستخدم ميزتين رئيسيتين في نظام لينكس:

- **Namespaces:** تعزل كل حاوية بحيث يكون لها نظام ملفات خاص, ومعرفات عمليات (PIDs), وواجهة شبكة خاصة. يعني كل حاوية تكون منفصلة تمامًا عن الثانية.
- **cgroups:** تتحكم في الموارد اللي تستخدمها كل حاوية, مثل الذاكرة والمعالج, حتى لا تستهلك حاوية وحدة كل الموارد وتسبب بطء في النظام.

بكل بساطة Namespace تحدد ايش تقدر توصله وCgroups تحدد قد ايش موارد تقدر تستهلك



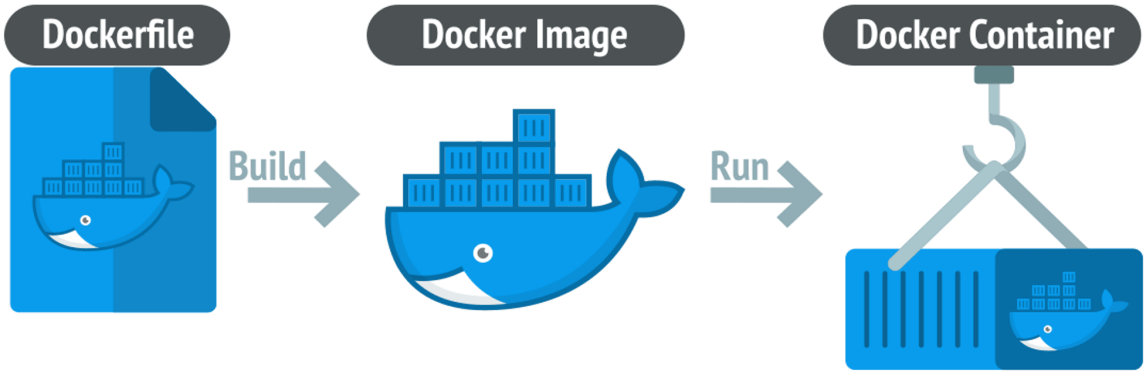
بناء الصور (Images) في Docker

Docker يعمل على مفهوم **الصور (Images)**, وهي القوالب التي منها يتم تشغيل الحاويات.

كل صورة تحتوي على التطبيق مع ملفاته ومتطلباته, وتُبنى باستخدام **Dockerfile**, وهو ملف يحتوي على خطوات تجهيز البيئة للتطبيق.

ميزة Dockerfile؟

- يضمن إن كل مرة تبني التطبيق يكون بنفس المواصفات.
- يقلل وقت البناء, لأن Docker يخزن كل خطوة كطبقة (Layer) ويعيد استخدامها لو ما تغيرت.

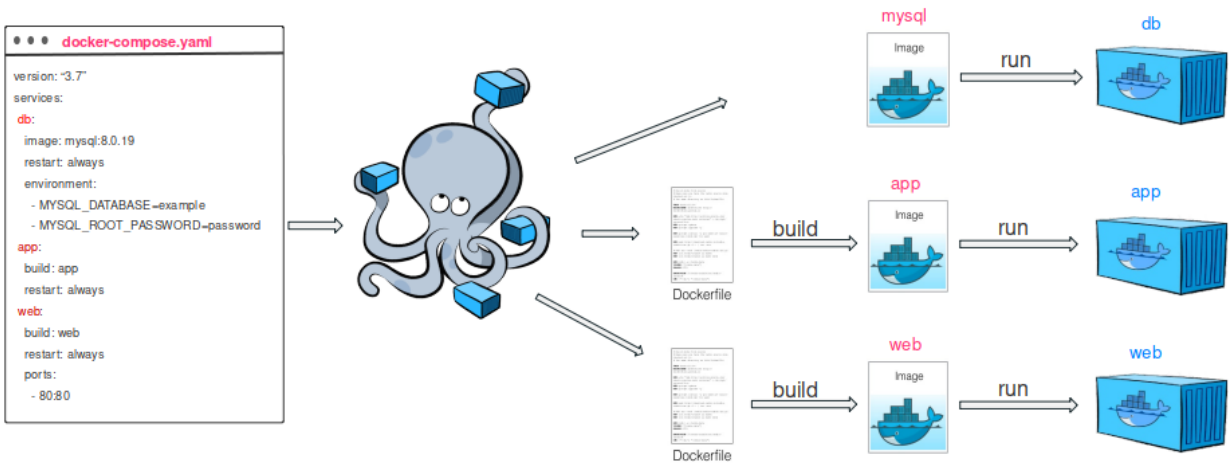


رسم 1- تنشئ ملف Dockerfile داخل مشروعك وتحدد فيه خطوات بناء الصورة (image) وبعد انشاء الصورة تشغلها داخل حاوية (container)

طيب ولو كان عندك قاعدة بيانات وباك-اند وفرونت-اند؟ 🤔

هنا تحتاج تنشئ **Dockerfile** لكل وحده منها (ما عدا قاعدة البيانات, موجود لها صور جاهزة), وحتى تنشئ صورة لكل وحده, هنا **بياخذ وقت طويل انك تغشل جميع الصور وتحدد المنافذ(ports) المطلوبه** خاصتا لو كان مشروعك مكون من 5 صور او اكثر.

هنا يجي دور docker-compose



عن طريق **docker-compose** تقدر تأتمت عملية بناء الصور وتشغيلها بكل سهولة عن طريق ملف **docker-compose.yaml**

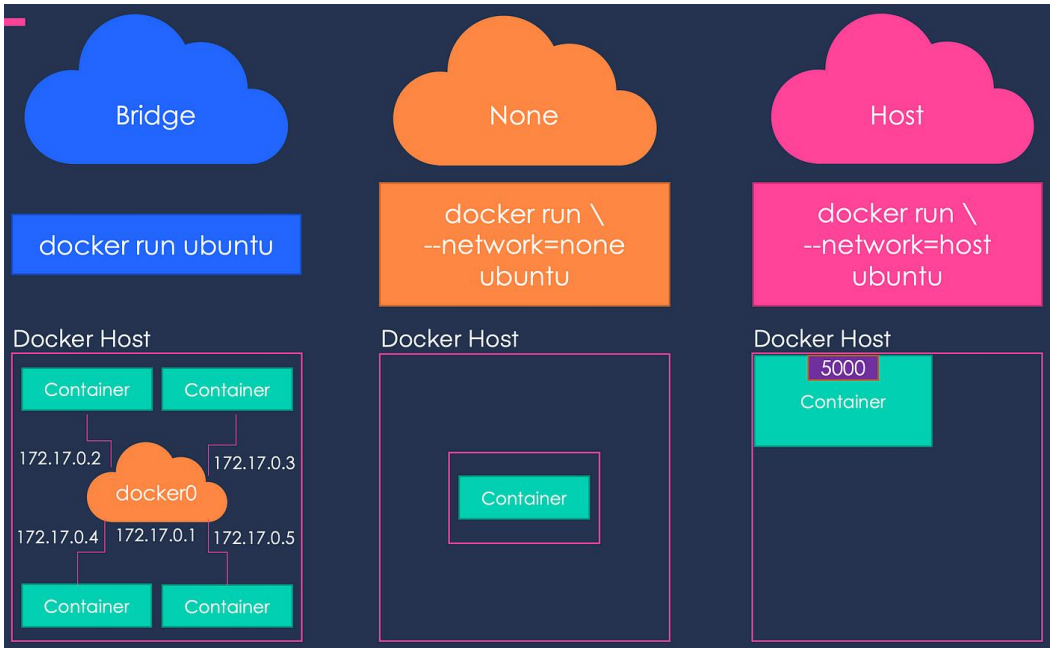
كيف يشتغل docker-compose؟

- يتيح لك تعريف **الخدمات (Services)** بدلاً من تشغيل الصور يدويًا.
- يتيح لك **ربط الحاويات ببعضها تلقائيا**, مثل ربط الباك-اند مع قاعدة البيانات بدون الحاجة لتعريف الشبكات يدويًا.
- يسهل إدارة المشاريع المعقدة المكونه من عدة حاويات

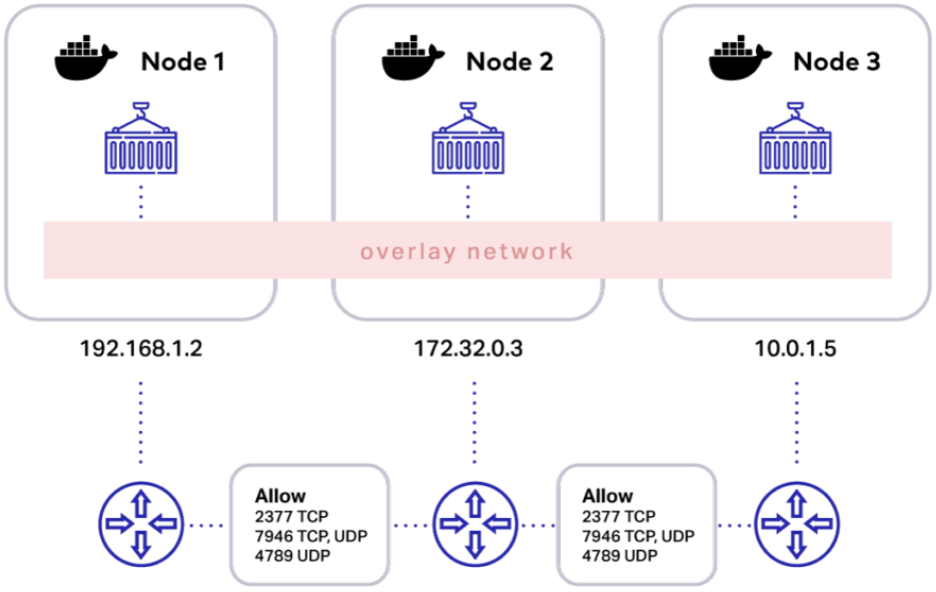
الشبكات (Networking) في Docker

الحاويات تحتاج تتواصل مع بعضها, وهنا يأتي دور الشبكات في Docker, وعندنا 3 أنواع رئيسية:

1. **Bridge (الجسر):** الافتراضي, يربط الحاويات ببعض على نفس الجهاز باستخدام عناوين IP.
2. **Host (المضيف):** يجعل الحاوية تستخدم نفس شبكة النظام الأساسي بدون فصل.
3. **Overlay (المتراكب):** يربط الحاويات على أجهزة مختلفة (مفيد إذا كنت تستخدم Kubernetes)



رسم 2- الفرق بين bridge وhost, الـhost يوصل الحاوية بشكل مباشر عبر منفذ 5000 الخاص بالجهاز والـbridge يوصل الحاويات داخل محرك دوكر عن طريق شبكة افتراضية



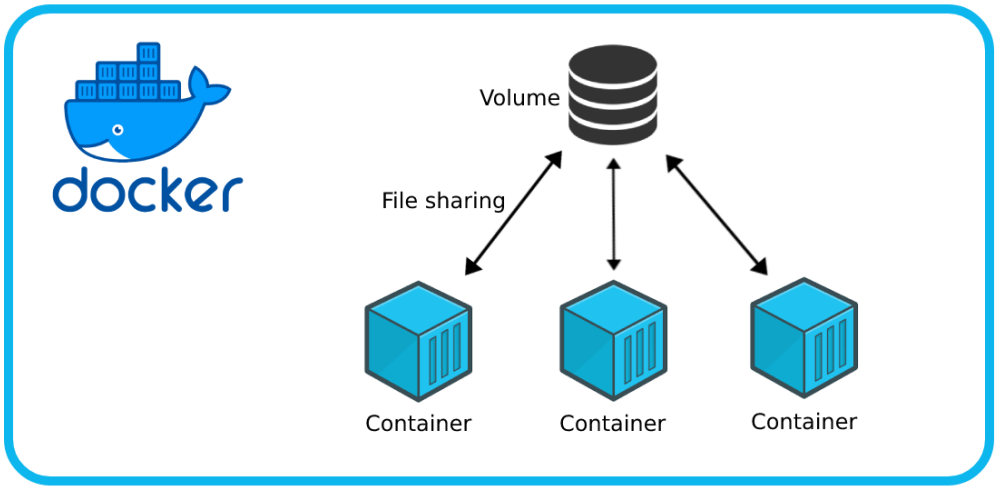
رسم 3- الـoverlay يوصل الحاويات من جهاز لجهاز وتستخدم لربط الحاويات في الشبكة المحلية مع الحاويات في السحابة

تخزين البيانات (Volumes)

بما إن الحاويات مؤقتة, أي بيانات تخزن بداخلها تحذف إذا حذفت الحاوية. الحل؟ **استخدام Volumes.**

الفائدة؟

- تخزين البيانات حتى لو تغيرت الحاوية.
- مشاركة البيانات بين أكثر من حاوية.
- أداء أفضل مقارنة بالتخزين داخل الحاوية نفسها.



التوسعة (Scalability) مع Docker

Docker يسمحلك تضيف أو تحذف حاويات بسهولة حسب الحاجة, وخصوصا مع **Kubernetes** ووظيفته يدير تشغيل الحاويات تلقائيا.

مثال: موقع متجر الكتروني وقت التخفيضات, يقدر يزيد عدد الحاويات بسرعة لاستيعاب العدد الكبير من المستخدمين, ثم يرجع للوضع الطبيعي بعد انتهاء الضغط.

Docker في الـ CI/CD

Docker يستخدم بشكل كبير في **خطوات التكامل المستمر (CI/CD)**, لأنه يضمن ان نفس الكود اللي يتم اختباره هو نفسه في الإنتاج.

الفائدة؟

- نشر أسرع وأكثر استقرارًا.
- نفس البيئة من التطوير حتى الإنتاج.

تحويل الأنظمة القديمة إلى Docker

إذا عندك تطبيق قديم, تقدر تحوله إلى Docker بدون اعادة كتابته بالكامل, وهذا يساعدك في:

- تقليل التكاليف
- تحسين الأمان
- تحديث النظام بسهولة

مزايا وعيوب Docker

العيوب	المزايا
يحتاج وقت لتعلمه	النقل بسهولة بين الأجهزة
إدارة البيانات داخل الحاويات معقدة	استخدام فعال للموارد
الشبكات داخل Docker معقدة أحيانًا	التوسع بسهولة
بعض الأدوات ما تدعمه بواجهة رسومية	بيئة موحدة لكل المراحل
مخاطر أمنية لو ما تم ضبط الإعدادات	عزل التطبيقات عن بعض

كيف تحل العيوب؟

- استعمل **Volumes** لحفظ البيانات.
- استخدم أدوات مثل Kubernetes لإدارة الشبكات.
- خذ وقتك في تعلم الأساسيات مع شروحات ودورات.
- حدث الصور باستمرار للحصول على آخر التحديثات الأمنية.

خطوات تشغيل Docker لأول مرة

تثبيت Docker

حمله من الموقع الرسمي:

- **Windows/Mac/Linux:** تقدر تحمل **Docker Desktop**
- **Linux** تقدر تستخدم أمر:

```
sudo apt-get install docker.io
```

أوامر Docker الأساسية

تشغيل حاوية:

```
docker run hello-world
```

بناء صورة:

```
docker build -t my_image .
```

تحميل صورة جاهزة:

```
docker pull nginx
```

إنشاء Dockerfile

لو تبغى تحط تطبيق Node.js داخل حاوية, استخدم هالملف:

```
# استخدم صورة جاهزة فيها Node.js
FROM node:14

# حدد المجلد الرئيسي
WORKDIR /usr/src/app

# انسخ ملفات المشروع
COPY package*.json ./

# ثبت الحزم المطلوبة
RUN npm install

# انسخ باقي الملفات
```

```
COPY . .

# حدد المنفذ اللي يسمع عليه التطبيق
EXPOSE 3000

# شغل التطبيق
CMD ["node", "server.js"]
```

بناء الصورة من Dockerfile

```
docker build -t my-node-app .
```

تشغيل الحاوية

```
docker run -p 8080:3000 my-node-app
```

الحين, تقدر تفتح المتصفح على:
http://localhost:8080

إدارة الحاويات

عرض الحاويات اللي تشتغل حاليا:

```
docker ps
```

إيقاف حاوية:

```
docker stop [container_id]
```

حذف حاوية:

```
docker rm [container_id]
```

تنظيف كل الحاويات المتوقفة:

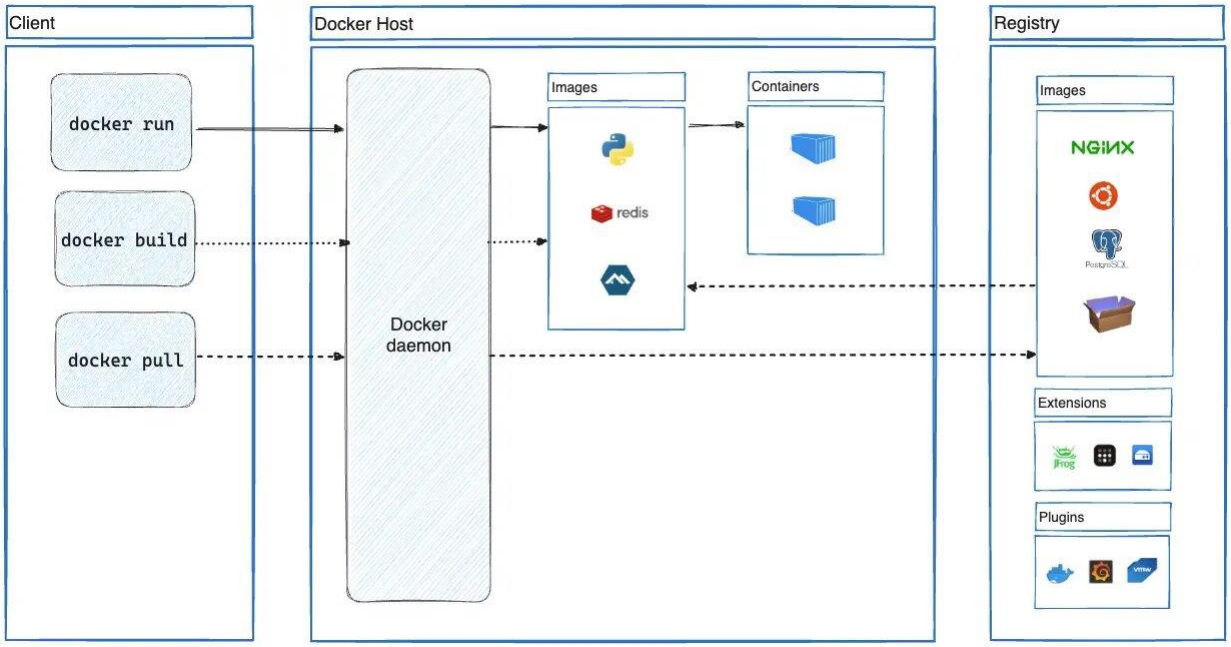
```
docker container prune
```

رفع الصورة لمستودع Docker

لو تبغى ترفع الصورة على Docker Hub عشان تستخدمها بأي مكان:

```
docker tag my-node-app username/my-node-app
docker login
docker push username/my-node-app
```

معمارية Docker



معمارية Docker ببساطة:

Docker يشتغل بنظام **العميل – الخادم**, يعني فيه **العميل (Client)** اللي تتعامل معه مباشرة, و **الخادم (dockerd–Daemon)** اللي يسوي الشغل الثقيل مثل تشغيل الحاويات (Containers) وبنائها وتوزيعها. العميل والخادم ممكن يكونون على نفس الجهاز, أو تقدر تتحكم في خادم Docker عن بعد باستخدام عميل Docker. التواصل بينهم يتم عن طريق **REST API** سواء عبر **UNIX sockets** أو عن طريق الشبكة.

مكونات Docker:

- **خادم (dockerd-Daemon)**: هو المسؤول عن تنفيذ أوامر Docker، يدير الصور (Images)، الحاويات (Containers)، الشبكات (Networks)، والتخزين (Volumes). يقدر يتواصل مع خوادم Docker ثانية لإدارة الخدمات.

- **عميل (Client)**: هي الواجهة اللي تتعامل معها، يعني اذا كتبت أوامر مثل docker run، العميل يرسلها للخادم عشان ينفذها, والعميل يقدر يتواصل مع أكثر من خادم.

- **Docker Desktop**: برنامج سهل التثبيت على **Windows, Mac, Linux** يخليك تشغل Docker بسهولة. فيه كل اللي تحتاجه مثل Docker Daemon, Docker Client, Docker Compose، وحتى Kubernetes.

- **Docker Registries**: هو مكان يخزن فيه **صور Docker**. أشهر **ريجستري** هو **Hub Docker** ويحتوي على آلاف الصور الجاهزة، لكن تقدر تسوي ريجستري خاص فيك لو بغيت. لما تسوي docker pull يسحب الصور من الريجستري، ولما تسوي docker push يرفع الصور إلى الريجستري اللي انت محدده.

الخاتمة

Docker مو مجرد أداة, هو تغيير شامل في طريقة تشغيل البرامج ويضمن لك:

- نشر أسرع.
- بيئة موحدة من التطوير حتى الإنتاج.
- سهولة التوسع والتحديث.

سواء كنت مطور مبتدئ أو محترف,Docker صار ضروري لكل شخص يتعامل مع التقنيات الحديثة, مثل **الميكروسيرفس** و**الخدمات السحابية**.

💡 **نصيحة:** خذ وقتك في التعلم, وابدأ بخطوات بسيطة, وبتشوف كيف Docker بيختصر عليك كثير من المشاكل في نشر المشاريع, ويفضل انك تحاول تطور مهاراتك في استخدام نظام تشغيل لينكس لانك تحتاج تعرف كيف تتعامل مع الحاويات والحاويات غالبا تشتغل بلينيكس.

مصادر:

:Documentation

https://docs.docker.com

مقدمة تعريفية لدوكر

YouTube 5-minutes – دوكر في 5 دقائق

دورة دوكر مجانية

YouTube 2-hours – Docker Course for Beginners