- شرح ال **Docker** بالكامل لكورس الباشمهندس **Ahmed Mohey** الموجود علي قناه **Codographia**

- رابط الكورس علي ال **YouTube**

https://www.youtube.com/watch?v=DFyPl2cZM2g&list=PLX1bW_GeBRhDkTf_jbdvBbkHs2LCWVeXZ

- ال Presentation المستخدمه في شرح الكورس

https://drive.google.com/drive/folders/1J2Nfe6nf0dt18JR0t9WhXP-sKzNDJqd5?usp=sharing

- شرح ال Kubernetes

https://www.linkedin.com/posts/mohamedelbitawy_%D8%B4%D8%B1%D8%AD-kubernetes-activity-7228526547815526400-RG5?utm_source=share&utm_medium=member_desktop

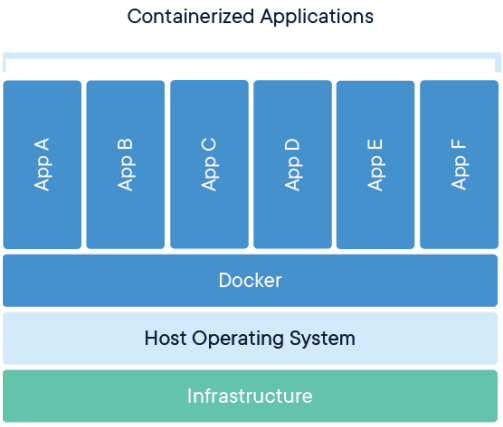
BY: Mohamed Atef Elbitawy

https://www.linkedin.com/in/mohamedelbitawy/

Basic Concepts

What is Docker?

- ال **Docker** هو عباره عن open source standalone application بمعني ان ال docker بيكون ابلكيشن مفتوح المصدر وبتنزله ل واحده وبيستخدم كمحرك engine علشان تشغل بيه Applications باستخدام ال Containers . ف انت تقدر تنزله علي اي Operating System سواء انت شغال علي windows او Linux او MacOS



- لو احنا بصينا علي الصوره دي هنلاقيه جايبلك ال Infrastructure بتاعتك ف ممكن يكون ال Infrastructure عباره عن Server او Laptop او Desktop .ف الجهاز اللي انت شغاله عليه بيكون عليه Operating System ف لو انت شغال علي windows مثلا تقدر تنزل عليه ال Application اللي اسمه Docker ومن خلال ال Docker ده هتقدر ت Manage بيه Containers وجوه كل Container منهم Application زي مانت شايف ان ممكن يكون عندك App A و App B و App C وهكذا

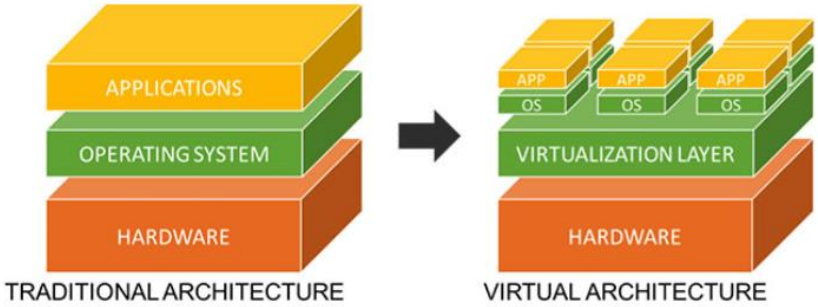
- ال Application اللي هيرن جوه ال Container هيكون معزول عن باقي ال Applications الموجوده في ال Containers التانيه والمسئول انه يعمل ده هو ال Docker engine . وممكن اكتر من Container يشتغل علي Operating System واحد
- تقدر ت run ال Applications دي علي Laptop او علي Server او علي ال Cloud بدون اي مشكله
- نقدر نعرف ال Docker علي انه عباره عن اداه بنقدر من خلالها ان احنا نتحكم في ال Containers

What is a Docker Container?

- ال **Container** هو جزء من ال Software بيكون موجود فيه Packages و Dependencies وملفات اللي الابلكيشن اللي الي هيكون محتاجها علشان تعمله run.يعني مثلا انت محتاج تعمل Container هيكون عليه Application SQL Server ف هيبدأ ال Docker ينزلك ال Files وال Packages اللي ال SQL Server محتاجها فقط .
- وال Container بتغلف ال Applications اللي انت عايزها ك image وال Image دي بتكون جواها كل حاجه انت محتاجها من code و runtime environment و Libraries و Configuration وبتكون خفيفه جدا وقائمه بذاتها

Virtualization

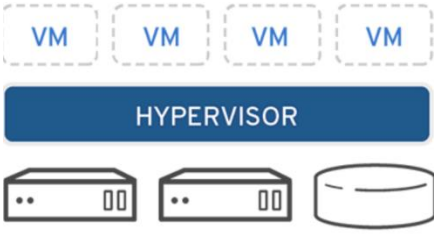
- ال Technoloy الخاصه بال Virtualization بتخليك ان انت تستغل الخدمات وال resources المتاحه ليك بتاعت ال IT اللي كانت مرتبطه بالهارد وير .يعني اقدر اخد Server مثلا عليه 100 جيجا رام واقدر اقسم ال 100 جيجا رام ل 20 جيجا رام في خمسه Servers وابدء اتعامل مع كل واحد علي حدي زي ماهنشوف



- ال **Traditional Architecture** اللي هي المعماريه التقليديه العاديه ان بيكون عندي Hardware وعليه Operating System وبيتحط عليه ال Application اللي انا عايزه .
- اما ال **Virtualization** بتقدر تعمل اللي موجود في ال Traditional Archiecture بشكل افضل ان انت بيكون عندك Hardware عادي وبيكون فيه Layer جديد اسمه Virtualization Layer ومن خلاله بقدر اقسم ال Hardware الواحد اللي عندي ل اكتر من Operating System وكل Operating System اقدر انزل عليه Application مختلف او Service مختلفه وهتتعامل ك server منعزل تماما
- في الصوره دي انا ممكن يكون عندي مثلا Mail Server و Web Server و LEGACY APPS وزي ماحنا شايفين هنلاقي ان ال MAIL مستخدم 30% من ال resources بتاعت الServer او الجهاز بتاعك ف ساعات مبيبقاش مستغل كل ال resources ف هتلاقي جزي من ال resource مفقود . ف ممكن اننا ناخد مثلا ال Legact apps مع ال Mail نحطهم في Server واحد باستخدام technology زي ال Virtualization ف كده انا وفرت Server ممكن استخدمه في حاجه تانيه

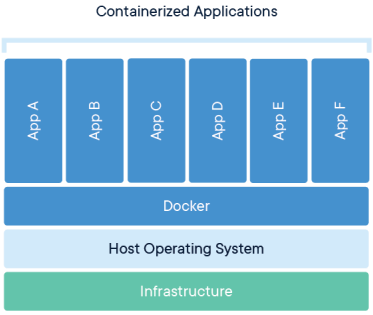
How Does Virtualization Work?

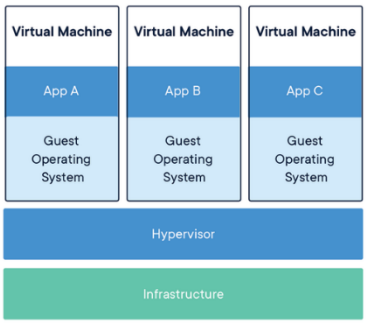
- ال Virtualization بتشتغل ازاي عن طريق ان بيكون فيه Application اسمه hypervisors او Virtual Machine Monitor(VMM) هو ده المسئول انه يفصل ال Physical Resources من HardDisk و Ram و Processor ل Virtual environments او حاجات افتراضيه او Server افتراضي . ف ال hypervisors بيكون نازل علي ال Operating System ومن خلاله يبدء يوزع ال resources



Virtualization VS Containerization

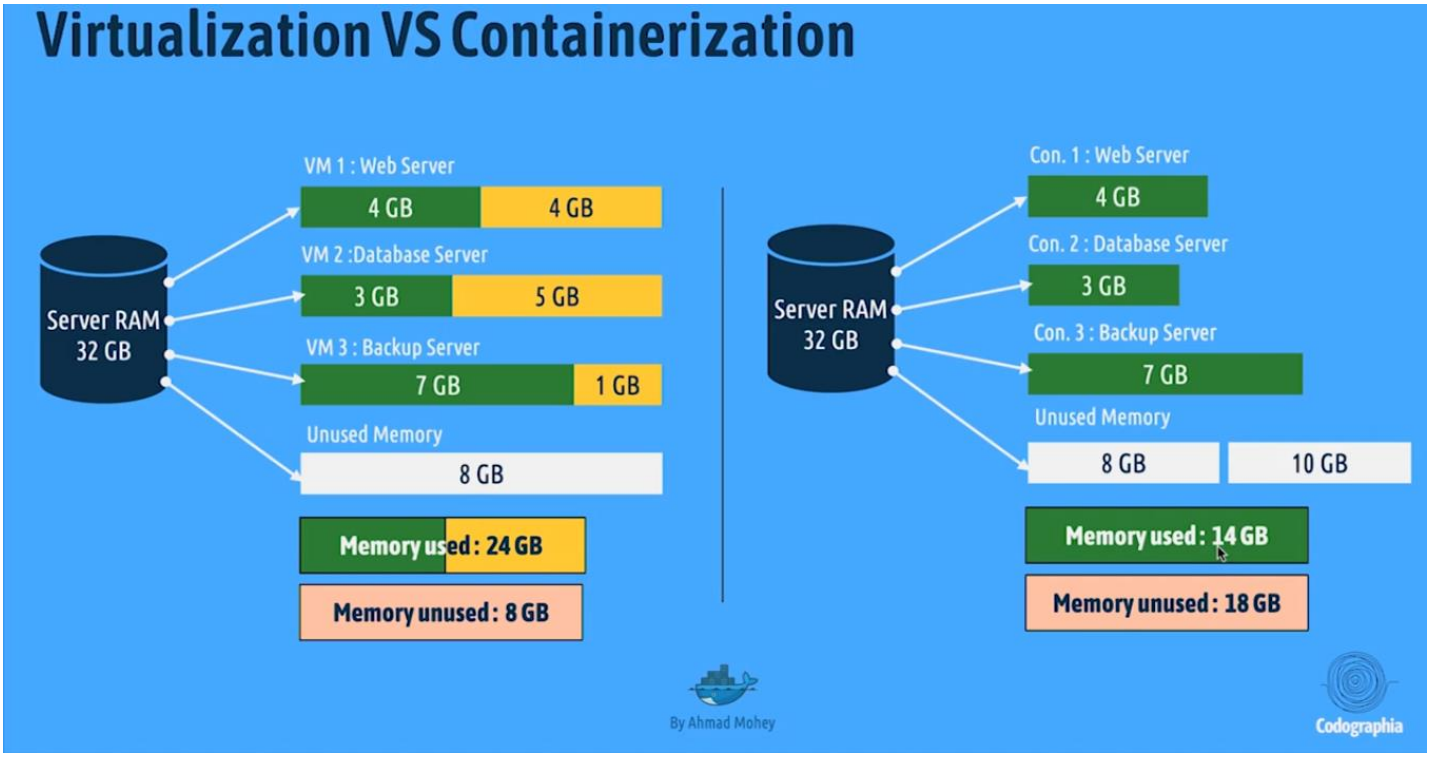
- ايه الفرق مابين ال Virtualization وال Containerization





- لو بصينا علي ال Two Modules دول طريقه عمل كل واحده منهم هنلاقي في ال Virtualization بيكون عندنا Infrastructure عندنا server مثلا وبنزل عليه Hypervisor ونبدء من خلال ال Hypervisor ننزل Operating System كامل او اكتر من OS ونبدء نتعامل معاه كانه جهاز منفصل تماما
- اما في ال Containerized هنلاقي ان فيه حاجه مفقوده وهي فكره ال Operating System بمعني اللي بيحصل في ال Containers ان انت هتنزل ال Docker علي ال Operating System ومش محتاج ان انت تنزل Operating System علشان تشغل ال Applications . وال Docker هو اللي هيعمل manage ل واحده . ف احنا هنا قدرنا احنا نستغني عن حاجه مهمه جدا وهي ال Operating System لانها مكلفه جدا

	Containerization	Virtualization
Portability	قابلية النقل أعلى وأسهل بين الأنظمة المتوافقة	النقل معقد ويتطلب توافق بين الـ Hypervisors
Lightweight	مساحه ال Container بتكون خفيفه لانه لا يحتوي علي نظام تشغيل بالكامل	مساحه ال Virtualization كبيره جدااا نظرا لانه يحتوي علي نظام التشغيل بالكامل
Native Performance	أداء أقرب للأداء الأصلي بدون تأخير تقريبًا	الأداء أبطأ بسبب الـ Hypervisor
Startup Time	يبدأ في ميلي ثانية	يبدأ في غضون ثوانٍ إلى دقائق
Multiple Containers	يمكن تشغيل عدد كبير من الحاويات بكفاءة عالية	يمكن تشغيل عدة VMs لكن مع استهلاك كبير للموارد
Simple and Fast Deployment	نشر سريع وسهل باستخدام أدوات مثل Docker	عملية نشر معقدة وتحتاج لإعداد نظام تشغيل كامل

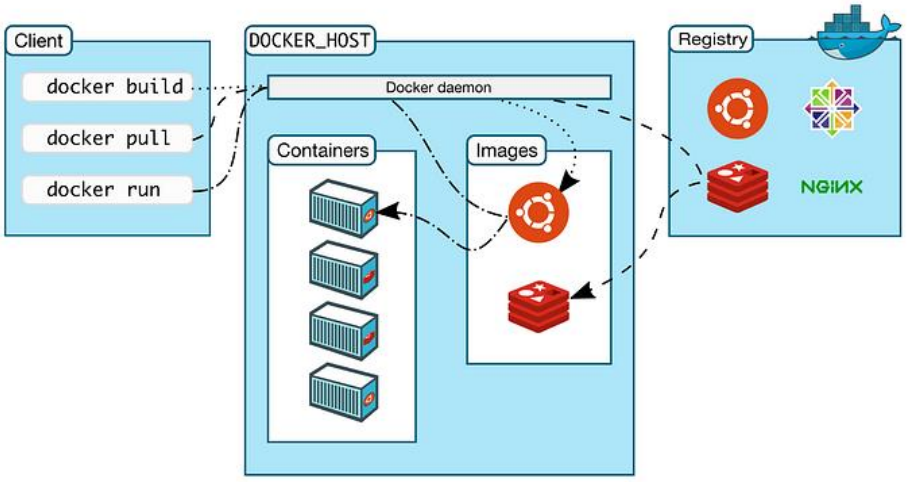


- **في المثال ده هنلاحظ الفرق مابين ال Virtualization وال Containerization**

- لو بصينا علي المثال ده هنلاحظ ان في ال Virtualization عندي Server ال RAM بتاعته بتكون 32GB وموجود عليه تلاته VM واحده Web Server والتانيه Database Server والتالته Backup Server وكل واحده منهم واخده 8GB من ال RAM ف الاخضر هو المستخدم والاصفر اللي مش مستخدم ف هتلاحظ ان انا عندي في ال Web Server ان المستخدم 4GB واللي مش مستخدم 4GB وهنلاقي كمان عندي في ال Database عندي 5GB مش مستخدمين وفي ال Backup عندي 1GB مش مستخدم ف هنلاحظ ان بقي عندي 10 جيجا مش مستخدمين بالاضافه لل 8 جيجا اللي انا كنت سايبهم مستخدمتهمش ف هنلاقي في ال Virtualization ان الميموري المستخدمه 24GB والميموري اللي مش مستخدمه

- اما في ال Container هنلاحظ ان في ال web Server انا انا حاجز 4GB من الرام فقط وسايب الباقي وهكذا في ال Database حاجز 3GB فقط وهكذا في ال Backup حاجز 7GB فقط فهنلاقي اللي متبقي من ال 32GB ال 8GB بتوع ال Server وال 10GB الباقي من كل Server ف هنلاقي في ال Containerization الميموري المستخدمه 14GB والميموري اللي مش مستخدمه 18GB

Docker Architecture



- ال **Image** عباره عن read-only template وبيحتوي علي Instructions تعليمات ازاي ت create من ال Image دي Docker Container
- ال **Container** هو ال Image بس عمول ليها running . بمجرد ان انت عملت Download لل Image اين كان نوع ال Image دي ايه وعملت ليها run كده بقي اسمها Container .وتقدر تعمل create و stop و move و delete ال container باستخدام ال Docker API او ال CLI
- **Registry** ده عباره عن مخزن موجود فيه كل ال Images اللي انا هستخدمها وال Docker Hub هو ده ال Default اللي ال Docker بيستخدمه علشان يعمل Download لل Image وال Docker Hub بيكون Public 0 Registry
- ال **Client** هو ده الوسيله الاساسيه ان ال Users تقدر تتعامل مع ال Docker ف علشان تنفذ command معين وليكن run او Stop ف بتتعامل من خلال ال Client وال Client ده ممكن يكون CLI او GUI . وال Docker بيستخدم ال Docker API علشان ينفذ ال Commands
- ال **Docker Daemon** هو زي ال Service بالظبط بي Listens منتظر ايه ال Command اللي هييجي ليه علشان يبدأ ينفذه وعلي حسب كل Command هييجي ليه هيبدأ يتصرف
- **Namespace:** ال Docker بيستخدم Technology اسمها Namespace هي اللي بتمد ال Docker ب Isolated Workspace ومفيش Namespace بيقدر يشوف ال Files الخاصه بال Workspace التانيه بيكونوا منعزلين عن بعض

➢ لو بصينا علي الصوره اللي فوق هنلاقي ان عندي ال Client اللي من خلاله بكتب ال Commands زي docker build و docker pull و docker run وهكذا وعندي في ال Docker Host عندي ال Docker Daemon اللي احنا قولنا انه زي ال Service بالظبط بي Listens منتظر ايه ال Command اللي هييجي ليه علشان يبدأ ينفذه وعلي حسب كل Command هييجي ليه هيبدأ يتصرف ويقدر يروح لو ال Image مش موجوده عندي علي الجهاز بيروح علي ال Registry علي ال Docker Hub ويطلب منه ال Image دي وليكن عايز Image اسمها NGINX او Ubuntu ف هيبدأ انه هيجيب ال Image دي وهيعملها Download وهيعملها run علشان تبقي Container ومن خلال ال Image دي بقدر من خلالها اني اعمل create ل اكتر من Container

Installing Docker Engine on Ubuntu

خطوات تنزيل ال Docker Engine علي ال Ubuntu

```
sudo apt update
```

```
sudo apt install -y ca-certificates curl gnupg lsb-release
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
sudo apt-get update
```

```
sudo apt install docker-ce docker-ce-cli containerd.io -y
```

```
sudo usermod -aG docker $USER
```

```
newgrp docker
```

Basic Commands in Docker

- اول command عندي وهو ال **run** ال run ده بيعمل حاجتين بيروح يعمل run لل image اللى انت مديهاله لو موجودة هايعملها run عالطول لو مش موجودة هايروح ينزلها من ال registry و باعدين يعملها run

```
mohamed@MohamedAtef:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:1408fec50309afee38f3535383f5b09419e6dc0925bc69891e79d84cc4cdcec6
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

هنلاقيه هنا بيقولك انه مش لاقي ال Image اللي اسمه hello-world مش لاقيها locally وبيقولك انه هيروح يسحبلك نسخه من علي ال Registry الي اسمه Docker Hub وبقا Pull Complete

وهنا طلعلنا ال message اللي هي Hello from Docker

هنا بيقولك ان عمليه ال installation بتاعت ال Docker شغاله بدون اي مشكله

- لو انا عايز اشوف ال Containers اللي عندي هستخدم **docker ps -all** ال command ده لما بنستخدمه بيعرضلي كل ال Containers اللي عندي بس لو فيه command عملتله run اكتر من مره بيجبلي اخر واحد اتعمله run .ولو نفذنا ال command ده هنلاقي حاجه اسمها CONTAINER ID وده بيكون عباره عن رقم مميز او فريد هنقدر نتعامل بيه مع كل ال Containers

```
mohamed@MohamedAtef:~$ docker ps -all
CONTAINER ID   IMAGE         COMMAND    CREATED          STATUS                      PORTS     NAMES
35c61af389b6   hello-world   "/hello"   24 minutes ago   Exited (0) 24 minutes ago             keen_sammet
```

- ممكن استخدم ال Command ده كمان لو انا عايز اعمل List او اشوف كل ال Containers اللي عندي بس ال command ده مختلف عن ال command الل قبليه في انه بيعرضلي كل ال containers اللى عندى بس لو فى command عملتله run اكتر من مرة هو بيجبهم كلهم

```
mohamed@MohamedAtef:~$ docker container ls -a
CONTAINER ID   IMAGE         COMMAND    CREATED          STATUS                      PORTS     NAMES
3835229d9f43   hello-world   "/hello"   5 seconds ago    Exited (0) 3 seconds ago              silly_varahamihira
35c61af389b6   hello-world   "/hello"   40 minutes ago   Exited (0) 40 minutes ago             keen_sammet
```

- ممكن يكون عندي Image واحده ويكون ليها اكتر من Containers وكل ال Containers هيكونوا منعزلين عن بعض
- هنعمل download ل image اسمها fedora وال fedora ده عباره عن Linux Operating system ف هنستخدم نفس ال command اللي استخدمناه وهو **run** او ممكن نستخدم **pull** بدل run ف لو لاحظنا هنلاقيه بيعمل Download لل fedora وحجمها صغير جدااا 80.12MB لانه مش منزل اي حاجه تانيه مفيش GUI مفيش ICON ومفيش Applications هو عامل download لاقل حاجه ممكن ال OS اللي اسمه fedora يشتغل بيها

```
mohamed@MohamedAtef:~$ Docker container run fedora
Unable to find image 'fedora:latest' locally
latest: Pulling from library/fedora
d4df0db66c89: Downloading [==================>                ]  30.76MB/80.12MB
```

- لو عملنا List لل Containers هنلاقيه عمل container تاني اسمه fedora ومعمو ليه create من 30 ثانيه ولو اخدنا بالنا في ال Status هنلاقيه معمول ليه Exited هو خرج من ال Container او قفله علشان انت عندك بالفعل Operating System بس مفيهوش اي Servers شغاله جواه وبمجرد ان بقي فيه Service شغاله ال OS هيفضل شغال لحد اما انت اللي تقفل ال Container

```
mohamed@MohamedAtef:~$ docker container ls -a
CONTAINER ID   IMAGE         COMMAND      CREATED          STATUS                      PORTS     NAMES
0f288a20e852   fedora        "/bin/bash"  30 seconds ago   Exited (0) 5 minutes ago              kind_albattani
3835229d9f43   hello-world   "/hello"     25 minutes ago   Exited (0) 22 minutes ago             silly_varahamihira
35c61af389b6   hello-world   "/hello"     About an hour ago  Exited (0) About an hour ago        keen_sammet
```

- لو انا عايز اعمل Remove ل Container هستخدم **docker rm** وبعدين ال Container ID وانت بتعمل remove لازم يرد عليك بنفس ال container id علشان تتاكد انه عمل remove بالفعل ولا لا . ممكن نحذف ال container ده اللي اسمه hello-world وال Container ID بتاعه بيكون d9f433835229

```
mohamed@MohamedAtef:~$ docker rm 3835229d9f43
3835229d9f43
```

- لو انا عايز اشوف ال Image اللي عندي هستخدم docker images هنلاقي عندنا في ال Repository هنلاقي Two Images

```
mohamed@MohamedAtef:~$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
fedora       latest    5e22da79803c   3 months ago  222MB
hello-world  latest    d2c94e258dcb   15 months ago 13.3kB
```

- هل لو انا عملت remove لكل ال Containers اللي عندي هل كده مبقاش عندي Images علي الجهاز لا ال Images هتكون موجوده لما ت run تاني هيجبلك من النسخه الموجوده في ال repository هنا
- لو انا عايز اعمل remove ل Image معينه هستخدم **docker Image rm** وبعدين اسم ال Image ولازم اتاكد ان مفيش اي containers عندي شغالين من نفس ال Images ولو فيه Containers من نفس ال Images شغالين ف لازم اعمل الاول remove لل containers d]وبعدين اعمل remove لل image

```
mohamed@MohamedAtef:~$ docker image rm fedora
Untagged: fedora:latest
Untagged: fedora@sha256:5ce8497aeea599bf6b54ab3979133923d82aaa4f6ca5ced1812611b197c79eb0
Deleted: sha256:5e22da79803c567fceb0e255f1168977259525a4279cb518016a60df025412fb
Deleted: sha256:505397f2c295cb36210061d24939c4bcbd449e57c8a53618feb114bfcdc70cc8
```

- هنشتغل علي Image اسمها Redis ودي من اشهر ال Databases وبيكون حجمها صغير جدا وسريعه جدا
- هنعمل download لل image دي باستخدام **docker containers run redis** لما تستخدم ال command هتلاقيه وهو بيعمل download لل image هتلاقي ان انت مش عارف تستخدم ال Terminal او ومش عارف تنفذ اي command بسبب انه واخد منك ال cruiser

```
mohamed@MohamedAtef:~$ docker container run redis
Unable to find image 'redis:latest' locally
latest: Pulling from library/redis
efc2b5ad9eec: Pull complete
82797145fff6: Pull complete
405e1ffae71e: Pull complete
0beb16fe974a: Pull complete
```

- لو فتحت Terminal تانيه وعملت list لل Container هنلاقي ان ال Container ده ال Status بتاعته Running وبمجرد ان انا قفلت ال terminal اللي فيها ال image ال container هيقفل

```
mohamed@MohamedAtef:~$ docker container ls
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS      NAMES
d4f1896470ba   redis     "docker-entrypoint.s…"   13 minutes ago  Up 13 minutes  6379/tcp   affectionate_morse
```

- ف الحل ان احنا هنشغلها في ال background ف هنستخدم مع ال command اوبشن **-d** وال **-d** دي معناها detached ان ال container هيتعمله run في ال background

```
mohamed@MohamedAtef:~$ docker container run -d redis
f59c8c59d6e2e9e6881c0f6e41e7789c1c641cf881f2990b309a6cfa7d97f0a4
mohamed@MohamedAtef:~$
```

- لو انا عايز اشوف معلومات عن ال Image من خلال ال command اللي اسمه **docker inspect**

```
mohamed@MohamedAtef:~$ docker inspect redis
[
    {
        "Id": "sha256:509b2fc82da65579aa63481c5b4909d1d777040ea574bf4be7aa1d6d48bf4b5f",
        "RepoTags": [
            "redis:latest"
        ],
        "RepoDigests": [
            "redis@sha256:79676a8f74e4aed85b6d6a2f4e4e3e55d8a229baa7168362e592bbfdc67b0c9b"
        ],
        "Parent": "",
        "Comment": "buildkit.dockerfile.v0",
        "Created": "2024-07-29T07:59:06Z",
        "ContainerConfig": {
            "Hostname": "",
```

- لو انا عايز اعرف ال Logs الخاصه بال container هستخدم **docker logs** وبعدين ال container id

```
mohamed@MohamedAtef:~$ docker logs f59c8c59d6e2
1:C 30 Jul 2024 21:12:42.859 * oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 30 Jul 2024 21:12:42.859 * Redis version=7.4.0, bits=64, commit=00000000, modified=0, pid=1, just started
1:C 30 Jul 2024 21:12:42.859 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/redis.conf
1:M 30 Jul 2024 21:12:42.860 * monotonic clock: POSIX clock_gettime
1:M 30 Jul 2024 21:12:42.861 * Running mode=standalone, port=6379.
1:M 30 Jul 2024 21:12:42.862 * Server initialized
1:M 30 Jul 2024 21:12:42.862 * Ready to accept connections tcp
```

- ممكن استخدم command اسمه **docker stats** ده بيجبلي stats عن ال Container واخد قد ايه من ال resources بتاعة الجهاز او ال Server بتاعي

```
mohamed@MohamedAtef:~$ docker stats f59c8c59d6e2
ONTAINER ID   NAME         CPU %   MEM USAGE / LIMIT   MEM %   NET I/O       BLOCK I/O   PIDS
f59c8c59d6e2   quirky_pare   1.39%   8.527MiB / 1.827GiB   0.46%   1.15kB / 0B   0B / 0B     6
```

- ممكن استخدم ال command ده **docker info** لو انا عايز اشوف معلومات اكتر وبالتفصيل عن ال Containers

```
mohamed@MohamedAtef:~$ docker info
Server:
 Containers: 3
  Running: 2
  Paused: 0
  Stopped: 1
 Images: 2
 Server Version: 26.1.1
 Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Using metacopy: false
  Native Overlay Diff: true
  userxattr: false
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Cgroup Version: 2
```

- لو انا عايز اعرف ال IP Address الخاص بال Container هستخدم ال Command ده

docker inspect --format='{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' container-id

```
mohamed@MohamedAtef:~$ docker inspect --format='{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' f59c8c59d6e2
172.17.0.3
```

مثال :- اللي هنعمله هنجيب Image هنعمل منها Container ل حاجه اسمها NGINX وال NGINX ده عباره عن Web Server ف احنا هنستخدم ال NGINX علشان نعرف ازاي نستخدم ال Commands

- ف احنا هنقوله docker run container وهنحط Two Options اول حاجه هنحطها وهي d- او ممكن نحطها --detach هنستخدم الاوبشن ده علشان يعمل run في ال background تاني اوبشن هنستخدمه وهو --publish وبعدين هحددله ال Port اللي هيعمل للابلكيشن ده Publish او expose عليه علشان اقدر استخدمه واحنا هنستخدم 80:80 وبعدين هستخدم اوبشن --name واحديله الاسم اللي انا عايزه وليكن n1 وبعد كده بديله اسم ال Image وهي nginx

```
mohamed@MohamedAtef:~$ docker container run --detach --publish 80:80 --name n1 nginx
4a04457b45c65c67372d108dc5589a01d8323d653a3e80de1b7dd0650cdf36ae
```

- لو احنا عايزين نتاكد ان اللي احنا عملنا صح ولا لا هنروح علي اي Browser وفي ال URL ونكتب localhost هيفتحلك صفحه موجوده بالفعل في ال Image اللي اسمها nginx وبيقولك Welcome to nginx

- علشان اعمل Stop لل Container هتسخدم docker container stop وبعدين اسم ال container اللي احنا حددناه واحنا بنعمل download لل nginx ممكن استخدم ال name او استخدم ال Container-Id

```
mohamed@MohamedAtef:~$ docker container stop n1
n1
```

- لو انا عايز اعمل Start لل Container هستخدم

```
mohamed@MohamedAtef:~$ docker container start n1
n1
```

- لو انا عايز ابدء ادي Commands لل Nginx ذات نفسه بيكون فيه جوه ال Nginx بيكون فيه Terminal او حاجه شبه ال Power shell اسمها Bash ف علشان اشغل ال Bash علشان ابدء انفذ ال Commands علي ال Nginx هستخدم **Docker exec -it container-name command** وال **-it** اختصار ل **-i** ودي معناها interactive و **-t** ودي معناها tty او pseudo-TTY . ف لما ننفذ ال Commands هنلاقيه مديك root ومديك ال ID الخاص بال Container

```
mohamed@MohamedAtef:~$ docker exec -it n1 bash
root@4a04457b45c6:/#
```

- احنا كده جوه ال nginx ف نقدر ننفذ اي command احنا محتاجينه وليكن انا عايز اعرف ال Versions الخاص بال Nginx هقوله service nginx

```
root@4a04457b45c6:/# service nginx -V
service ver. 1.65.2
```

- وممكن كمان اعرف ال Status بتاعت ال Nginx

```
root@4a04457b45c6:/# service nginx status
nginx is running.
```

Docker Tags, Image Layers and Dockerfiles

Docker Tags

- ال **Docker Tags** هي عباره عن reference ل Docker Images بمعني ان هي عنوان لل Docker Image يعني بدل اما انا بجيب ال Image اللي عندي بال ID بتاعها اللي غالبا بيكون رقم ف انا بدي لل Image دي Tags علشان اقدر اعمل Commands بكل سهوله بعمل Tag عشان اعمل صورة طبق الاصل من ال image بس باسم مختلف
- واكتر من Tags يقدر يشاور علي نفس ال Image يعني انا اقدر يكون عندي Tags اسمه Latest و Tag تاني اسمه 2022 ك version او حتي يكون عندي Tag ك Custom Image كل ال Tags دي ممكن يكونوا بيشاوروا علي نفس ال Image ID بدون اي مشكله
- كلمه Latest في ال Tags مش معناهاا اخر نسخه موجوده لل Docker Image تقدر تقربها ان هي ال Default . انت لما تيجي ت Build image لو انت محددتش Tag ال Docker ذات نفسه بيحط ال Latest ف هو ب Act ان هي ال Default
- ال Image ID او ال Image بتتكون من Image Layers

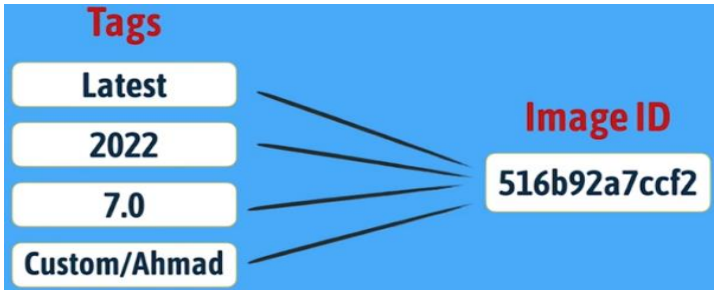
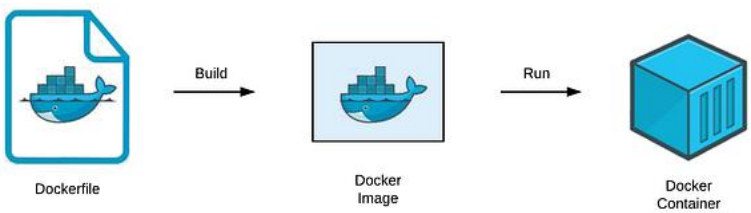


Image Layers

- ال **Image Layers** في ال Docker ان اي تغير هتعمل علي ال Image بيعتبر Layer منفصل يعني مثلا عملت Command او عملت Run او Copy او نزلت Applications او Service فكل تغييره من دول بتعتبر Layer وال Layer بتكون Image قائمة بذتها .يعني مثلا انا هعمل Image ل Ubuntu ف ده كده بقي Layer قائمه بذاتها وكمان علي ال Layer او ال Image دي هحط فوقيه Apache ف ده بقي Layer قائم بذاته واجي في الاخر هحط عليهم ال App ف كل واحده من ال Images دي بتكون قائمه بذاتها
- مجموع ال Layers دي او ال Commands اللي نت بتكتبها بتتحط في حاجه اسمها Docker File

Dockerfile

- ال **Dockerfile** بيكون عباره عن file بيكون موجود فيه مجموعه من ال Instructions او ال Commands وبيروح ال Docker Client يقراها ال File ده وبعدين يعملك build لل Image

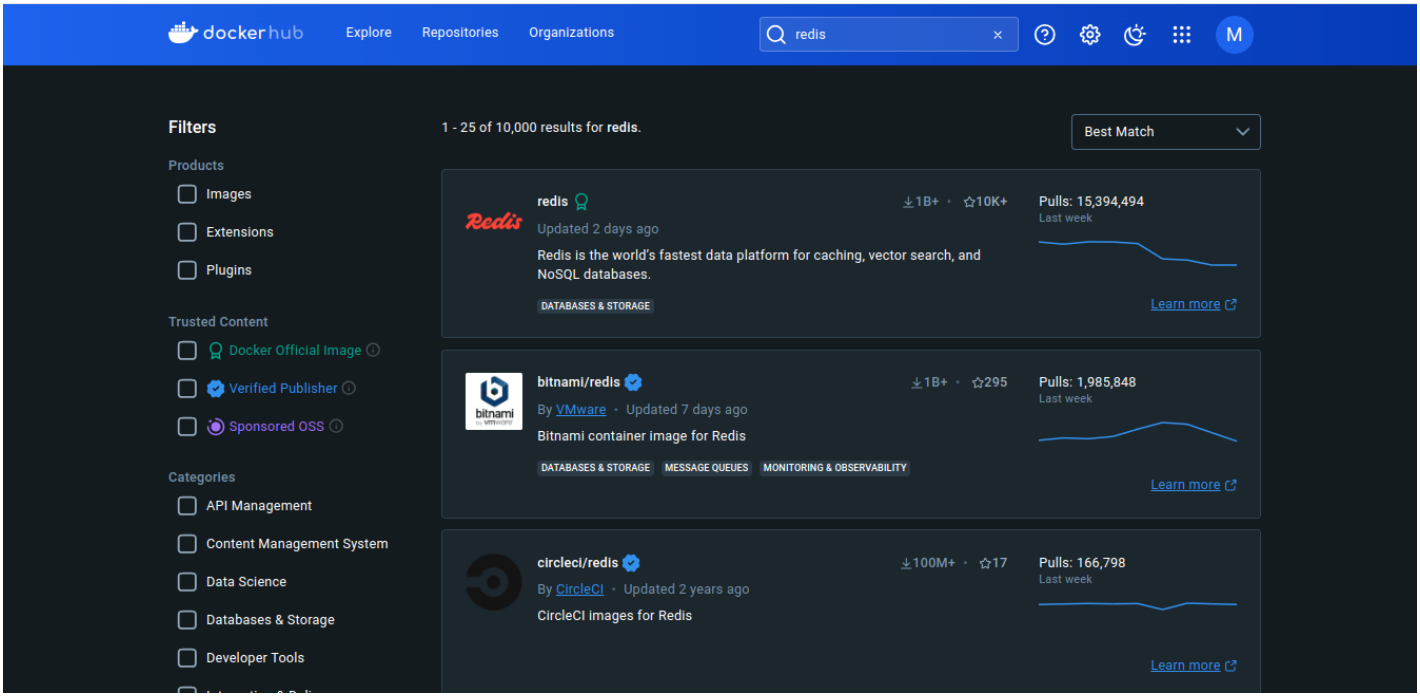


Create New Tags and Push to Docker Hub

- لو نفذنا ال command اللي اسمه docker tag هنلاقيه بيقولك ان ال Tag بياخد Two Arguments بياخد ال Source_Image وبياخد ال Target_Image وممكن تحط Tag لكل واحد منهم وبيكون اوبشن وال Tag بيروح يعمل Create ل Target Image نفس ال Source اللي انا هدهوله يعني لو انا قولتله redis هيروح يعمل create ل صوره طبق الاصل من ال Image اللي اسمها Redis وانا هسميها اي اسم تاني

```
mohamed@MohamedAtef:~$ docker tag
"docker tag" requires exactly 2 arguments.
See 'docker tag --help'.
Usage:  docker tag SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG]
Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
```

- لازم تعمل Account علي ال Docker Hub ف لو احنا فتحنا ال Docker Hub هنلاحظ ان اول image وهي ال Redis دي هي ال Official Image ف لما بعمل download لل Image دي بستخدم **docker pull redis** بس لو انا استخدمت ال Image التانيه اللي اسمها bitnami/redis ال image دي مش ال official Image لان ال bitnami دي شركه تانيه عاملة نسخه من ال redis ف لازم تكتب قبليها اسم ال username وبعدين /slash وبعدين اسم الImage ف لما بعمل download لل image دي بقوله **docker pull bitnami/redis**
- ولو انا هرفع Image علي ال Docker Hub محتاج ان انا هقولك ال username الخاص بال account بتاعي زي مثلا mohamedelbitawy وبعدين /slash وبعدين اسم ال Image زي مثلا mohamedelbitawy/redis



- علشان اعمل Image طبق الاصل من ال redis بس باسم مختلف هسنخدم

```
mohamed@MohamedAtef:~$ docker tag redis mohamedelbitawy/redis
```

- لو عملنا List لل Image اللي عندي هنلاقيه عمل create ل Image باسم mohamedelbitawy/redis وواخده Tag latest لان احنا محددنلهوش Tag

```
mohamed@MohamedAtef:~$ docker image ls
REPOSITORY                TAG       IMAGE ID       CREATED        SIZE
redis                     latest    509b2fc82da6   2 days ago     117MB
mohamedelbitawy/redis     latest    509b2fc82da6   2 days ago     117MB
nginx                     latest    a72860cb95fd   5 weeks ago    188MB
hello-world               latest    d2c94e258dcb   15 months ago  13.3kB
```

- لو انا عايز احدد Tag غير ال Latest وليكن هسميه dev باستخدام

```
mohamed@MohamedAtef:~$ docker tag redis mohamedelbitawy/redis:dev
```

- لو عملنا List تاني لل Image هنلاقي Image اللي كنا عملينهاا في الاول واخده tag latest والتانيه واخده dev وهنلاقيهم الاتنين بيشاورو علي نفس ال Image Id

```
mohamed@MohamedAtef:~$ docker image ls
REPOSITORY                TAG       IMAGE ID       CREATED        SIZE
redis                     latest    509b2fc82da6   3 days ago     117MB
mohamedelbitawy/redis     dev       509b2fc82da6   3 days ago     117MB
mohamedelbitawy/redis     latest    509b2fc82da6   3 days ago     117MB
nginx                     latest    a72860cb95fd   5 weeks ago    188MB
hello-world               latest    d2c94e258dcb   15 months ago  13.3kB
```

- لو انا عندي Image وعايز ارفعها علي ال Docker Hub لازم اول حاجه اعمل Login علي ال Docker Hub من خلال ال CLI عن طريق اني استخدم docker login

```
mohamed@MohamedAtef:~$ docker login
Log in with your Docker ID or email address to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com/ to create one.
You can log in with your password or a Personal Access Token (PAT). Using a limited-scope PAT grants better security and is required for organizations using SSO. Learn more at https://docs.docker.com/go/access-tokens/

Username: mohamedelbitawy1
Password:
Login Succeeded
```

- بعد كده هرفع ال Image علي ال Docker Hub

```
mohamed@MohamedAtef:~$ docker push mohamedelbitawy1/redis
Using default tag: latest
The push refers to repository [docker.io/mohamedelbitawy1/redis]
a40c28010fa8: Pushed
5f70bf18a086: Layer already exists
d006ca741dee: Pushed
c1f1787a9884: Pushed
76d5ae2cbb38: Pushed
a3c1988f2d6e: Pushed
3bae4306a5ad: Pushed
e0781bc8667f: Pushed
latest: digest: sha256:1805cbcbf3f5da4c7f8b676e947547059347e3195257e0ad516a319f1782afc6 size: 1986
```

Building Docker Image from Docker file

- هنشوف ازاي هنعمل Build ل Image من الصفر وعلشان نعمل كده هنستخدم ال Docker file وال Docker file بيكون فيه Script و ال Docker Client بيروح يقرئه علشان ينشئ ال Image اللي انت بتبتدي بعد كده تسحبها وتستخدمها علشان تبقي Container
- ف احنا هنعمل Folder باسم dockerfile وهنعمل file باسم Dockerfile بنفس ال Syntax ومن غير اي format او extension وهنفتح ال File دي بأي Editor موجود وليكن Visual Studio Code
- ال Script اللي بيبني ال Docker File بيكون موجود فيه Commands كتيره جدا زي مثلا
  - ال **FROM** بستخدمها لو انا عايز Operating System لل Image الجديده عن طريق اني بكتب بعد FROM بكتب ال Operating System اللي انا عايزه وليكن alpine ودي تعتبر اصغر نسخه لينكس موجوده
  - ال **CMD** بستخدمه لو انا عايز اطلع message معينه و بكتب جواها ال action اللى هانفذه فى اول argument بكتب ال command و الargument التانى بكتب ال argument بتاعة ال command نفسه
  - كل اللي هيعمله هيعمل Create ل Linux OS نوعه Alpine وهيعمل run لل Command وهيطلعلي message اسمها Hello Mohamed, I am Your Custom Image!

```
FROM alpine
CMD [ "echo", "Hello Mohamed, I am Your Custom Image!" ]
```

- علشان اعمل Build لل Dockerfile لازم ادخل علي نفس ال Path اللي موجود فيه ال Dockerfile وبعدين اعمل Build باستخدام Docker build وبعدين احددله ال Path اللي موجود فيه ال Dockerfile ولو انا واقف في نفس المسار هستخدم دوت . هو هيفهم لما احط دوت ان ال Dockerfile ده موجود local

```
mohamed@MohamedAtef: ~/Desktop/dockerfile$ docker build .
[+] Building 11.2s (6/6) FINISHED                                docker:desktop-linux
 => [internal] load build definition from Dockerfile                              0.4s
 => => transferring dockerfile: 105B                                              0.2s
 => [internal] load metadata for docker.io/library/alpine:latest                  4.2s
 => [auth] library/alpine:pull token for registry-1.docker.io                     0.0s
 => [internal] load .dockerignore                                                 0.2s
```

- علشان اتاكد انها اتعملها Build هنعمل List علي ال Image ف هنلاقي اتعملها create وهنلاقيها باسم none لان احنا محددناش ليها Repository ولا Tag

```
mohamed@MohamedAtef: ~/Desktop/dockerfile$ docker image ls
REPOSITORY          TAG       IMAGE ID       CREATED        SIZE
<none>              <none>    6787dc91493f   9 days ago     7.8MB
```

- وعلشان اتاكد فعلا ان هي دي ال Image اللي اتعملها build هنروح نعملها run باستخدام ال Image id ف المفروض لما نعمل Run هيطلعلنا ال Message اللي كنا كاتبينها في الCMD في ال Dockerfile

```
mohamed@MohamedAtef: ~/Desktop/dockerfile$ docker container run 6787dc91493f
Hello Mohamed, I am Your Custom Image!
```

- علشان اغير اسم ال Imageوانا بعمل build هتسخدم tag-- وبعدين اسم ال Image وبعدين احددله مسار ال Dockerfile

```
mohamed@MohamedAtef: ~/Desktop/dockerfile$ docker build --tag hellovs .
[+] Building 11.2s (6/6) FINISHED                                docker:desktop-linux
 => [internal] load build definition from Dockerfile                              0.4s
 => => transferring dockerfile: 105B                                              0.2s
 => [internal] load metadata for docker.io/library/alpine:latest                  4.2s
 => [auth] library/alpine:pull token for registry-1.docker.io                     0.0s
 => [internal] load .dockerignore                                                 0.2s
```

- لو عملنا تاني Image ls هنلاقي اسم ال Image بقي hellovs وواخد tag latest

```
mohamed@MohamedAtef: ~/Desktop/dockerfile$ docker image ls
REPOSITORY          TAG       IMAGE ID       CREATED        SIZE
hellovs             latest    6787dc91493f   9 days ago     7.8MB
```

- لو عملنا run تاني لل Image

```
mohamed@MohamedAtef: ~/Desktop/dockerfile$ docker container run hellovs
Hello Mohamed, I am Your Custom Image!
```

- فيه Command بنستخدمه في ال Docker file اسمه **RUN** وال Command ده بستخدمه لو انا عايز اعمل Run ل Commands معينه زي مثلا ان انا عايز اعمل Update او ان انا اعمل Install ل Package معينه جوه ال Image وهكذا وال apk معناها ان دي Package Manger الخاصه بال Alpine وهو هنا بيعمل Update لل redis

```
FROM alpine
RUN apk add --update redis
CMD [ "redis-server" ]
```

- لو عملنا Build لل Dockerfile

```
mohamed@MohamedAtef: ~/Desktop/dockerfile$ docker build --tag ourredis .
[+] Building 12.0s (7/7) FINISHED                                docker:desktop-linux
 => [internal] load build definition from Dockerfile                              0.3s
 => => transferring dockerfile: 98B                                               0.0s
 => [internal] load metadata for docker.io/library/alpine:latest                  2.6s
 => [auth] library/alpine:pull token for registry-1.docker.io                     0.2s
 View a summary of image vulnerabilities and recommendations → docker scout quickview
```

- لو عملنا run لل container

```
mohamed@MohamedAtef: ~/Desktop/dockerfile$ docker container run ourredis
1:C 01 Aug 2024 13:08:58.265 * oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 01 Aug 2024 13:08:58.265 * Redis version=7.2.5, bits=64, commit=00000000, modified=0, pid=1, just started
1:C 01 Aug 2024 13:08:58.265 # Warning: no config file specified, using the default config. In order to specify a config file use
redis-server /path/to/redis.conf
1:M 01 Aug 2024 13:08:58.268 * monotonic clock: POSIX clock_gettime
1:M 01 Aug 2024 13:08:58.270 * Running mode=standalone, port=6379.
1:M 01 Aug 2024 13:08:58.271 * Server initialized
1:M 01 Aug 2024 13:08:58.271 * Ready to accept connections tcp
```

Data Management in Docker

- هنتكلم عن ال **Data Management**

- لو اخدت بالك ان بمجرد ان انت عملت Delete لل Container حتي لو ال Image موجوده لما بتيجي تعمل Containerجديد منها بتلاقي ان اي حاجه موجوده جوه ال Container بتتمسح ومتبقاش موجوده ف معني كده ان انت بتخسر الداتا الموجوده اول اما بت Delete ال Container ف دي مشكله لازم يكون ليها حل

- فيه concept اسمه **Separation of Concerns** ان احنا نفصل الاختصاصات عن بعض ان انا لو مسحت ال Application ال Data تفضل زي ماهي ان هما عملوا جزء خاص بال Application وجزء خاص بال Data ال Data نفسها اللي ال Application هيعملها manipulation او داتا بيز مثلا بيتحفظ فيها Data معينه مفصوله علي جنب تقدر من ال Docker ان انت تفصلها عن بعض ونشتغل بيها عادي جدا من خلال Two Options موجودين عندنا اول Option اسمه Volumes وتاني Options اسمه Bind Mounting

Volumes

- النوع اللي اسمه Volumes بيتحفظ ك جزء من ال Host filesystem ال Host ده هو الجهاز اللي انت منزل عليه ال Docker عندك وبيتعمله Manage بال Docker ولو انت شغال علي ال Linux ال Path بتاعه بيكون /var/lib/volumes/
- ومن الخصائص بتاعته ان اي Process مش تبع ال Docker المفروض متعدلش في ال Filesystem
- و Docker بت recommend لو انت عندك داتا عايز تحتفظ بيها يكون ال Volumes هو اختيارك
- وتقدر كمان من خلال ال Volumes ان انت ب Volume واحد بس تقدر تعمله Mount ان انت توصله بأكتر من Containers في نفس الوقت يعني انا لو عندي File Share اقدر اربطه ب اكتر من Container في نفس الوقت
- وتقدر تستنتج لما تعمل Delete ل Container من الكلام اللي شرحناه ان ال Volumes اللي Attached لل Container مبيتعملهوش Remove بيفضل موجود وتقدر ت attached ال Volume ده علي Container تاني ولو انت عايز تعمله Delete بتعمله Delete ل واحده

Bind Mounting

- ال Bind Mounting بتقدر تعملها Store علي اي مكان علي ال Host Systems عندك
- علي عكس ال Volumes ان اي Process مش تبع ال Docker موجوده علي ال Docker Host او علي ال Docker Container نفسه تقدر تعدل فيهم في اي وقت بدون اي مشكله
- ولو انت محتاج File او Folder وهو مش موجود بيتعمله Create on demand في ساعتها
- وعلشان تستخدم خاصيه ال Bind Mound مع ال Host مع ال Container لازم تديله ال Full Path علي ال Host Machine

Using Docker Volumes and Bind Mounting

- هنعمل Download ل Image اسمها MariaDB ودي عباره عن Database زي ال MySQL ودي هنستخدمها في الشرح والامثله
- هنعمل Pull لل image اللي اسمها MariaDB علشان اعملها Download

```
mohamed@MohamedAtef:~$ Docker pull mariadb
Using default tag: latest
latest: Pulling from library/mariadb
9c704ecd0c69: Pull complete
f8f7f3c9a741: Pull complete
97c034108521: Pull complete
```

- لو عملنا docker inspect علشان اعرف معلومات عن ال Image دي هنلاقي ان فيه section عندي اسمه Volumes وبيقولك ان ال Path بتاعه هو /var/lib/mysql

```
mohamed@MohamedAtef:~$ docker inspect mariadb
"Image": "",
        "Volumes": {
            "/var/lib/mysql": {}
        },
```

- هنعمل run لل mariadb وهندلها اسم mymariadb وبعدين هنحط للداتا بيز دي باسورد

```
mohamed@MohamedAtef:~$ docker run -d --name mymariadb -e MARIADB_ROOT_PASSWORD=1234 mariadb
a06a270b80425cdc93809cd926025201848a0d95bfc183eeef2fd1037e20365e
```

- لو عملنا container ls هنلاقي ال mariadb معمول ليها running وواخده اسم mymarisdb

```
mohamed@MohamedAtef:~$ docker container ls
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS      NAMES
a06a270b8042   mariadb   "docker-entrypoint.s…"   7 seconds ago   Up 4 seconds   3306/tcp   mymariadb
```

- لو انا عايز اعرف inspect معلومات عن ال Container ذات نفسه هستخدم docker inspect mymariadb ولو روحنا علي الجزء اللي فيه ال Mounts هنلاقيه بيقولك ان ال Type اللي احنا مستخدمينه هو volume وجايبلك كمان ال Name بتاع ال Volume وجايبلك ال source وال Destination وال Source ده عندي انا علي ال Host موجود فين وال Destination هو علي ال Container موجود فين

```
mohamed@MohamedAtef:~$ docker inspect mymariadb
"Mounts": [
        {
            "Type": "volume",
            "Name": "cb4c6e17ce9cdab34a2e9bf6aa1a0a7cef86177405088e0fd2e207a925ba28cc",
            "Source":"/var/lib/docker/volumes/cb4c6e17ce9cdab34a2e9bf6aa1a0a7cef86177405088e0fd2e207a925ba28cc/_data",
            "Destination": "/var/lib/mysql",
            "Driver": "local",
            "Mode": "",
            "RW": true,
            "Propagation": ""
        }
```

- اقدر اعمل check علي ال Name اللي ظاهرلي ده

cb4c6e17ce9cdab34a2e9bf6aa1a0a7cef86177405088e0fd2e207a925ba28cc

```
mohamed@MohamedAtef:~$ docker inspect cb4c6e17ce9cdab34a2e9bf6aa1a0a7cef86177405088e0fd2e207a925ba28cc
[
    {
        "CreatedAt": "2024-08-01T17:54:09Z",
        "Driver": "local",
        "Labels": {
            "com.docker.volume.anonymous": ""
        },
        "Mountpoint": "/var/lib/docker/volumes/cb4c6e17ce9cdab34a2e9bf6aa1a0a7cef86177405088e0fd2e207a925ba28cc/_data",
        "Name": "cb4c6e17ce9cdab34a2e9bf6aa1a0a7cef86177405088e0fd2e207a925ba28cc",
        "Options": null,
```

- لو انا عايز اعمل List لكل ال Volumes اللي عندي هستخدم docker volume ls

```
mohamed@MohamedAtef:~$ docker volume ls
DRIVER    VOLUME NAME
local     becc083475fe527be1f2b868047fe8106432457f58c78e5cbce5afde37701996
local     cb4c6e17ce9cdab34a2e9bf6aa1a0a7cef86177405088e0fd2e207a925ba28cc
local     maria-vol
local     mariavolume
```

- بيبقي صعب جدا ان انت تعرف ال Volume بالشكل الصعب ده ف اكيد لما تحب تستخدم ال Volume مش هتكتب كل الارقام والحروف بالشكل ده ف اللي احنا هنعمله هنغير اسم ال Volume ل اسم سهل انت تقدر تتذكره وتقدر تتعامل معاه

```
mohamed@MohamedAtef:~$ docker run -d --name mariadbnew -e MARIADB_ROOT_PASSWORD=1234 -v mariavolume:/var/lib/mysql mariadb
```

- ف علشان ادي ال Volume اسم بدل الاسم الكبير هستخدم نفس ال command اللي استخدمناه واحنا بنعمل run لل Container بس هزود اوبشن v- علشان بحددله ان انا بديله اسم لل Volume وبعدين هسمي ال Volume باي اسم وليكن mariavolume وبعدين column : وبعدين ال Path الخاص بال Volume وبعدين اسم ال image

```
mohamed@MohamedAtef:~$ docker volume ls
DRIVER    VOLUME NAME
local     becc083475fe527be1f2b868047fe8106432457f58c78e5cbce5afde37701996
local     cb4c6e17ce9cdab34a2e9bf6aa1a0a7cef86177405088e0fd2e207a925ba28cc
local     maria-vol
local     mariavolume
```

- لو عملنا inspect لل container اللي اسمه mariadbnew هنلاحظ ان ال Name استغير اسمه بدل الارقام الكتيره بقي mariavolume

```
mohamed@MohamedAtef:~$ docker inspect mariadbnew
"Mounts": [
        {
            "Type": "volume",
            "Name": "mariavolume",
            "Source": "/var/lib/docker/volumes/mariavolume/_data",
            "Destination": "/var/lib/mysql",
            "Driver": "local",
            "Mode": "z",
            "RW": true,
            "Propagation": ""
```

- ولو عملنا List علي ال Volumes هنلاحظ ان اسم ال Volume اصبح mariavolume

```
mohamed@MohamedAtef:~$ docker volume ls
DRIVER    VOLUME NAME
local     4d4325a5e2969be4d32a694e3d5d47782e5c6b59d3f84a819d27eb1e9652ab8c
local     cb4c6e17ce9cdab34a2e9bf6aa1a0a7cef86177405088e0fd2e207a925ba28cc
local     maria-vol
local     mariavolume
```

- ممكن اعمل Create ل Volume جديد عن طريق اني استخدم **Docker volume create volume-name**

```
mohamed@MohamedAtef:~$ docker volume create maria-vol
maria-vol
```

- لو عملنا List تاني لل Volume هنلاقيه عمل create ل volume بنفس الاسم اللي احنا حددناه

Using Bind Mounting in Docker

- الفكره اللي قائم عليها ال Bind Mounting ان هي عامله زي ال Share Folder في ال Windows كانك بالظبط بتعمل Share ل Folder واكتر من شخص يقدر يعمل Access عليه هي نفس القصه احنا هنشوف مكان علي ال Host اذا كان Folder او مجموعه من ال Folders وال Files ونعملها Share بطريقه معينه جوه ال Container بحيث ان ال Container يبقي شايفها

- ف احنا اول حاجه هنعملها هنعمل Create ل Folder ونعمل جواه file علشان ال Folder ده اللي هنعمله Share علي ال Container ف هنعمل create ل folder باسم data وهنعمل جوه ال folder ده ملف باسم data01.txt

```
mohamed@MohamedAtef:~$ mkdir data
mohamed@MohamedAtef:~$ cd data
mohamed@MohamedAtef:~/data$ touch data01.txt
mohamed@MohamedAtef:~/data$ ls
data01.txt
```

- كل اللي احنا هنعمله هنعمل نسخه بسيطه من ال Linux alpine وهعمل Share مابين ال Host ومابين ال Container هعمل share ل ال Folder اللي احنا عملنا ليه create اللي اسمه data ف علشان اعمل كده هستخدم docker container run -it وبعدين --name علشان احددله اسم ال Container وبعدين اوبشن -v وبعدين ال Path وبعدين اسم ال Folder اللي هيكون علي ال Container وبعدين اسم ال Image . لما انفذ ال Command ده هيدخلي عال container واقدر انفذ ال Commands اللي انا عايزها

```
mohamed@MohamedAtef:~/data$ docker container run -it --name alpine -v $(pwd):/mounted_folder alpine
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
Digest: sha256:0a4eaa0eecf5f8c050e5bba433f58c052be7587ee8af3e8b3910ef9ab5fbe9f5
Status: Downloaded newer image for alpine:latest
/ #
```

- ف لو عملنا ls جوه ال Container هنلاقي ال Folder اللي اللي احنا عملنا ليه create واحنا بنعمل ال Container وهو ال mounted_folder ولو دخلنا علي ال Folder ده هنلاقي ال File اللي اجنا عملنا ليه Create في ال Folder اللي اسمه data

```
/ # ls
bin         lib         opt         sbin        usr             dev         media       proc        srv         var
etc         mnt         root        sys         home        mounted_folder  run         tmp
```

Docker Networking

- المقصود بال **Docker Networking** ببساطه ان احنا ببساطه نعمل Connections بطريقه معينه بين ال Containers وبعضها وبين ال Containers وال Host اللي ال Daemon بيكون شغال عليه وال Docker هو اللي ب handle الموضوع ده من خلال انه بيعمل Create لحاجه اسمها Bridge Network
- عندي 3 انواع من ال Defaults Network وهوا ال Bridge وال Host وال None

1- وال Bridge Network ببساطه هو اللي بيخلي ال Containers اللي علي نفس ال Bridge تشتغل وتاخد IP وتشوف بعض بدون اي مشكله وده بيكون ال default settings اول اما انت بت Create ال Containers ال docker هو اللى بيعمله و بيوصل ال containers كلها ببعضها من خلال انه بيعمل شبكة داخلية توصلهم كلهم ببعض و ممكن اقسم الشبكة دى لاكتر من شبكة بس كل شبكة بتبقى منفصلة عن التانية

2- ال Host ده وظيفته انه بيلغي فكره ال Isolation هو بيعمل كده لان ال type اللي اسمه Host بيبتدي يستخدم نفس ال Network Interface الموجوده علي ال Host

3- ال None بفصل ال Network علي ال Container ان ال Network Driver مش بي Attach ال Container لأي Network ده بيخلي ال Container مش accessible لل Containers اللي معاه ولا بي accessible الاجهزه من بره يعني كأنك قفلت ال Network علي ال Container

➢ لو انا عايز اعرض كل ال Network اللي عندي هستخدم **docker network ls**

```
mohamed@MohamedAtef:~$ docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
b5931a82a306   bridge    bridge    local
3487d9a6cb0b   host      host      local
8caeb3c24c06   none      null      local
```

➢ لو انا عايز اعمل Create ل Network جديده **Docker network create –driver bridge bridge-name**

```
mohamed@MohamedAtef:~$ docker network create --driver bridge n1
47d241808aaa9b56c624a64b214aa71a745c113bc12a3997444d934545f2d138
```

➢ لو عملنا ls لل network هنلاقي ان بقي عندي Network اسمه n1 ونوعه bridge زي ماحددنا

```
mohamed@MohamedAtef:~$ docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
b5931a82a306   bridge    bridge    local
3487d9a6cb0b   host      host      local
47d241808aaa   n1        bridge    local
8caeb3c24c06   none      null      local
```

- لو انا عايز اعرف ال IP بتاعت ال Network اللي عملنا ليها Create هنعمل Inspect علي ال Network اللي اسمها n1 ف هنلاقي بيقولك انه واخد IP 172.18.0.1 وال subnet بتاعه بيكون 172.18.0.0/16

```
mohamed@MohamedAtef:~$ docker network inspect n1
"Config": [
            {
                "Subnet": "172.18.0.0/16",
                "Gateway": "172.18.0.1"
            }
        ]
```

- لو انا عايز اعمل Container واعمله attached لل Network اللي اسمها n1

```
mohamed@MohamedAtef:~$ docker run --network=n1 -d -it ubuntu
9a9fc18c4625d01e30b3e3de0205ee6964b5a1d02f601d28d24efe96c0bf257c
```

- لو انا عايز اعمل Network جديده بس انا اللي هحدد ال subnet

```
mohamed@MohamedAtef:~$ docker network create --driver bridge --subnet 172.25.0.0/16 n2
767dfd9217a1307085f337e37773956ab29066783b9e01d12cde05ebf6127043
```

- لو عملنا inspect لل network علشان اتاكد انه اخد نفس ال ip اللي انا مديهوله ولالا

```
mohamed@MohamedAtef:~$ docker network inspect n2
 "Config": [
            {
                "Subnet": "172.25.0.0/16"
            }
        ]
    },
```

- لو انا عايز اعمل disconnect لل Bridge Network من علي ال Container ان افصل ال Network عن طريق ان استخدم **Docker network disconnect bridge-name container-id**

```
mohamed@MohamedAtef:~$ docker network disconnect n1 9a9fc18c4625
```

- لو انا عايز اعمل Connect علي Network تانيه هستخدم

```
mohamed@MohamedAtef:~$ docker network connect n1 9a9fc18c4625
```

Docker Compose

- ال **Docker Compose** هو ببساطه عباره عن Tool بنقدر ن define ون Run بيها اكتر من Container في نفس الوقت باستخدام language اسمها YAML بنقدر نكتب اسكريبت في ال Configuration file وال Docker Compose بيروح يقرأ ال Configuration File ويبتدي يعمل Create وبيكون اسمه **docker-compose.yml**
- ال YAML ده اختصار ل Yet Another Markup Language ودي مجرد لغه بنقدر نكتب بيها ال Configuration بتاعت اكتر من Tool ومنهم ال Tool اللي اسمها Docker Compose
- لما نيجي نعمل ال Configuration file هنلاقيه مكتوب دايما فيه Service اول اما تشوف كلمه Service تعرف ان هو قصده علي Container
- ف ال Docker Compose بيخليك ت Create ال Development Environment بتاعتك كلها مره واحده ان انت تكتب ال Service اللي انت عايزها وتعملها Run او Stop مره واحده
- **فيه عندي كذا Version في ال Docker Compose**

1- Version: 1

- دي كانت أول إصدار وبتعتبر الأساس ولكن مبنستخدمهاش. الملف بتاعها بيتكتب بصيغة YAML . في ال Version 1 مبنكتبش رقم الإصدار في أول الملف في الوقت ده، لو كنت عايز تربط كونتينرات ببعض، كنت بتستخدم خاصية الـ links. كنت بتعمل زي "وصلة" بين الكونتينرات عشان يقدروا يتواصلوا مع بعض زي مثلا

```
redis:
  image: redis
db:
  image: postgres:9.4
vote:
  image: voting-app
  ports:
    - 5000:80
  links:
    - redis
```

2- Version: 2

- في Version 2 من Docker Compose، لازم تحدد الإصدار في بداية ملف docker-compose.yml
- كل الكونتينرات اللي هتشغلها لازم تتحط تحت القسم services. ده بيسهل تنظيم الكود ويساعد في إدارة التطبيقات اللي بتتكون من أكتر من خدمة
- في Version 2، مش محتاج تستخدم links عشان تربط الكونتينرات ببعض. Docker Compose بيعمل شبكة نوعها bridge بشكل تلقائي بين كل الخدمات الموجودة في الملف. وده معناه إن كل الكونتينرات هتقدر تتواصل مع بعضها بالاسم مباشرةً بدون الحاجة لتعريف روابط (links).
- لو عندك كونتينر معين عايز تتأكد إنه يشتغل قبل كونتينر آخر، بتستخدم depends_on. ده بيضمن إن الـ container اللي بتعتمد عليه الخدمة يشتغل الأول.

- مثال على استخدام depends_on:

```
version: 2
services:
  redis:
    image: redis
  db:
    image: postgres:9.4
  vote:
    image: voting-app
    ports:
      - 5000:80
    depends_on:
      - redis
```

- **الخلاصة:**
- Version 2 بيوفر لك طريقة مرتبة وسهلة لإدارة كونتينراتك. بتحدد الإصدار، تحط كل الخدمات تحت services، مش محتاج تعمل links، وبتستخدم depends_on عشان تتحكم في ترتيب تشغيل الكونتينرات

3- Version: 3

- Version 3 بتشبه Version 2 بشكل كبير في طريقة تعريف الخدمات والشبكات واستخدام depends_on. الفرق الأساسي هو إن Version 3 بتوفر لك بعض الميزات المتقدمة زي deploy اللي بتكون مفيدة لو شغال على Distributed Application أو محتاج تحكم أكتر في الموارد.
- كمان تقدر تتحكم في حجم الـ resources (زي الـ CPU والـ memory) اللي كل كونتينر هيستخدمها، وده بيديك تحكم أكتر في تشغيل التطبيق

```
version: 3
services:
  redis:
    image: redis
  db:
    image: postgres:9.4
  vote:
    image: voting-app
    ports:
      - 5000:80
```

- لو انا عايز اشغل كل ال services الموجوده في ملف docker-compose.yml

```
mohamed@MohamedAtef:~$ docker-compose up
```

- لو انا عايز اشغل كل ال services الموجوده في ال Background هستخدم اوبشن d-

```
mohamed@MohamedAtef:~$ docker-compose up  -d
```

- لو انا عايز اوقف واحذف كل ال Services وال Networks اللي تم إنشاؤها بواسطة docker-compose up.

```
mohamed@MohamedAtef:~$ docker-compose down
```

- لو انا عايز اعرض قائمة بالكونتينرات اللي شغالة حاليًا والمُدارة بواسطة Docker Compose.

```
mohamed@MohamedAtef:~$ docker-compose ps
```

- لو انا عايز اعمل build لل images المحددة في ملف docker-compose.yml بدون ما اشغلها.

```
mohamed@MohamedAtef:~$ docker-compose build
```

- لو انا عايز اعمل stop لل Containers اللي شغالة بدون ما احذفها.

```
mohamed@MohamedAtef:~$ docker-compose stop
```

- لو انا عايز اعمل start لل Containers اللي معمول ليها Stop

```
mohamed@MohamedAtef:~$ docker-compose start
```

- لو انا عايز اعمل اعاده تشغيل لل Containers

```
mohamed@MohamedAtef:~$ docker-compose restart
```

- لو انا عايز اعرض ال Logs الخاصه بكل ال Services

```
mohamed@MohamedAtef:~$ docker-compose logs
```

- لو انا عايز اشغل امر معين داخل Container شغال

```
mohamed@MohamedAtef:~$ docker-compose exec [service_name] [command]
```

- لو انا عايز اتحقق من صحه ملف ال docker-compose.yml

```
mohamed@MohamedAtef:~$ docker-compose config
```

ايه الفرق مابين كل من ال Kubernetes - Docker Swarm - Docker Compose – Docker

1- Docker

- ال Docker هو عباره عن Containerization Platform او منصه للتعامل بتقنيه ال Containerization اللي بسهوله بتخليك ت Create وت Deploy وت Run ال Applications في ال Docker Containers وكل ال Dependencies بتكون موجوده جوه ال Container سواء كان Frameworks او Libraries او files
- ال Container يقدر يتحرك بين اي بيئه عمل وبيئه تانيه بسهوله تامه ويشتغل بدون اي مشاكل تقدر تشغل ال Container علي لينكس او ماك او ويندوز او سيرفر وده بيخليك تضمن الكود نفسه اللي انت جربته بيشتغل علي ال Production Environment

2- Docker Compose

- ال Docker Compose بيستخدم علشان يعمل Configuring ويعمل Start ل Multiple Docker Containers علي نفس ال Host وبيتعاملوا كأنهم Single Service
- ومن المميزات ان انت تقدر تعمل Start لل Containers او ال Services ب Command واحد بس

3- Docker Swarm

- ال Docker Swarm عباره عن Container orchestration tool الخاص بعدد كبير من ال Containers وممكن يشتغلوا علي اكترمن Host
- وال Docker Swarm بيقدر يعمل Task زي ال Scaling او ان هو يشغلك Container لو حصله Crashes وبيقدر يعمل Networking بين ال Containers

4- Kubernetes

- ال Kubernetes هو عباره عن Container orchestration tool زي ال Docker Swarm بس ال Kubernetes علي Scale اكبر بكتير
- ال Kubernetes بيعمل Orchestrate لل Containerized Applications وبتعمل Run لل Applications علي Cluster من ال Hosts سواء كان on-premises infrastructure او Public Cloud Platforms

الفرق بين CMD و ENTRYPOINT

- هنشوف ايه الفرق مابين ال CMD وال ENTRYPOINT في ال Docker file لو روحنا نفذنا ال Command ده ["echo", "Hello World"] في ال CMD او ال ENTRYPOINT زي مواضح كده ف هنلاقي النتيجه بعد اما نعمل Create لل Image ونعمل Run لل Container اللي هيكون مبني علي الكلام ده ان هو هيطلع رساله Hello World في الحالتين بس الفرق ان في ال CMD لو روحت عملتلها Overwrite او لو عندك اكتر من CMD هي دايما هتنفذلك في ال Container اخر CMD عندك. اما في ال ENTRYPOINT ممكن ت execute command وت Append عليها وتعمل Overwrite وتطلعلك Message تانيه باستخدام نفس ال Command

CMD	ENTRYPOINT
`CMD ["echo", "Hello World"]`	`ENTRYPOINT ["echo", "Hello World"]`

Docker Commit

- ال **Docker Commit** عباره عن انه بياخد ال Container بعد اما تعمله Create وبعد اما تعمل عليه كل التعديلات لو انت عدلت في ال Container او ان انت حطيت Files او Application او عملت اي حاجه ف ال Container ف ال Docker Commit بيحول ال Container ل Image جديده ب اسم جديد او Tag جديد تقدر تستخدمها علي طول من ال State اللي انت وصلتلها .بس مش هيشمل معاه اي data متخزنه في ال Volumes هو بيتعامل مع ال Container فقط وفي مرحله ماهو بياخد ال Container ويحوله ل Image بيعمل Pause لل Process اللي جوه ال Container

➢ لو انا عايز اعمل Image جديده ل Container

```
mohamed@MohamedAtef:~$ docker commit <container_id_or_name> <new_image_name>:<tag>
```

او

```
mohamed@MohamedAtef:~$ docker commit abc123def456 my_new_image
```

او

```
mohamed@MohamedAtef:~$ docker commit abc123def456 my_new_image:v1.0
```

BY: Mohamed Atef Elbitawy

https://www.linkedin.com/in/mohamedelbitawy/