مقدمه ای بر داکر

نظرسنجی سال 2022 استک اور فلو از هفتاد هزار توسعه‌دهنده نرم‌افزار نشان می‌دهد که امروز داکر در کنار ابزارهای اساسی دیگر، مانند گیت، به یکی از ابزارهای اصلی و عصای دست برنامه‌نویسان تبدیل شده است. به همین خاطر است که این روزها کمتر آگهی استخدامی در حوزه برنامه‌نویسی می‌بینید که اشاره‌ای به آشنایی یا تسلط به داکر نداشته باشد.

۱. چطور داکر به دوره «ولی روی سیستم من کار می‌کند!» پایان داد؟

دوران طلایی؛ ورود داکر

در سال 2010 سالامون هایکس و سباستین پال، بنیانگذاران شرکت dotCloud، کار ساخت داکر را آغاز کردند و در سال 2011 داکر رسماً در پروژه‌های این شرکت مورداستفاده قرار گرفت.

به جهان متن‌باز خوش آمدید!

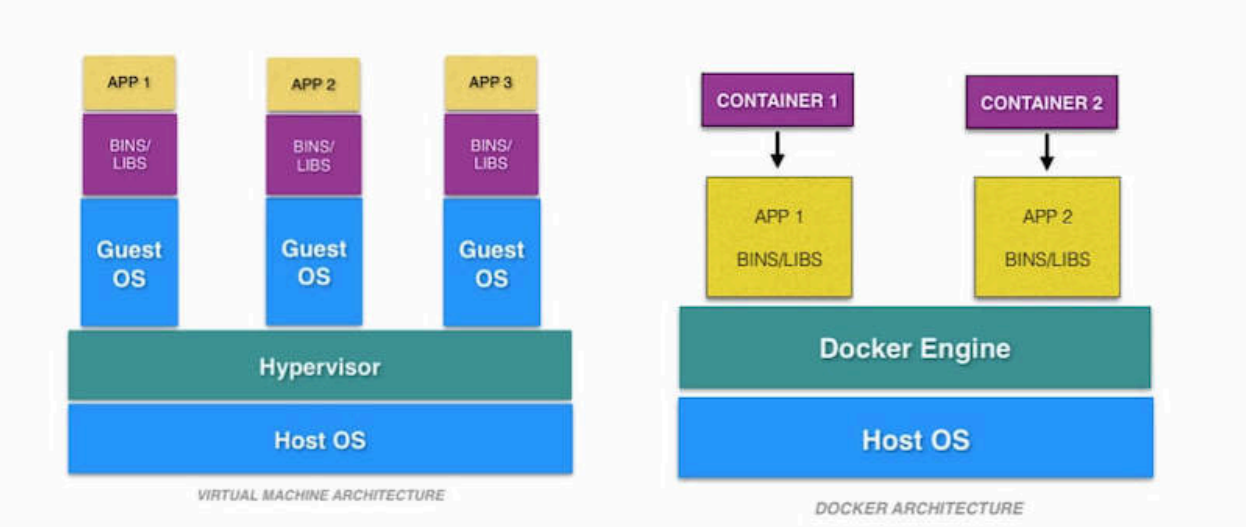
بیایید به مارس 2013 برگردیم. زمانی که سالامون هایکس اعلام کرد که docker از این پس به‌صورت متن‌باز در دسترس خواهد بود. شش ماه بعد از انتشار متن‌باز داکر، این ابزار توانست در گیت‌هاب به 6700 استار برسد و 175 نفر در آن به‌صورت داوطلبانه مشارکت کنند.

2. داکر چه کاربردهایی دارد؟

«سریع‌تر توسعه بده. هر جا که خواستی اجرا کن». این شعار دقیقاً خلاصه‌ای است از امکاناتی که docker برای‌تان فراهم می‌کند.

- با استفاده از docker می‌توانید اپلیکیشن‌های خود را به محیط تست ببرید و تست‌های دستی و خودکار را روی آن اجرا کنید.
- وقتی به‌عنوان دولوپر در کارتان به باگ می‌خورید، می‌توانید آن را در محیط توسعه برطرف کنید و فایلتان را برای تست و سنجش اعتبار دوباره دیپلوی کنید.

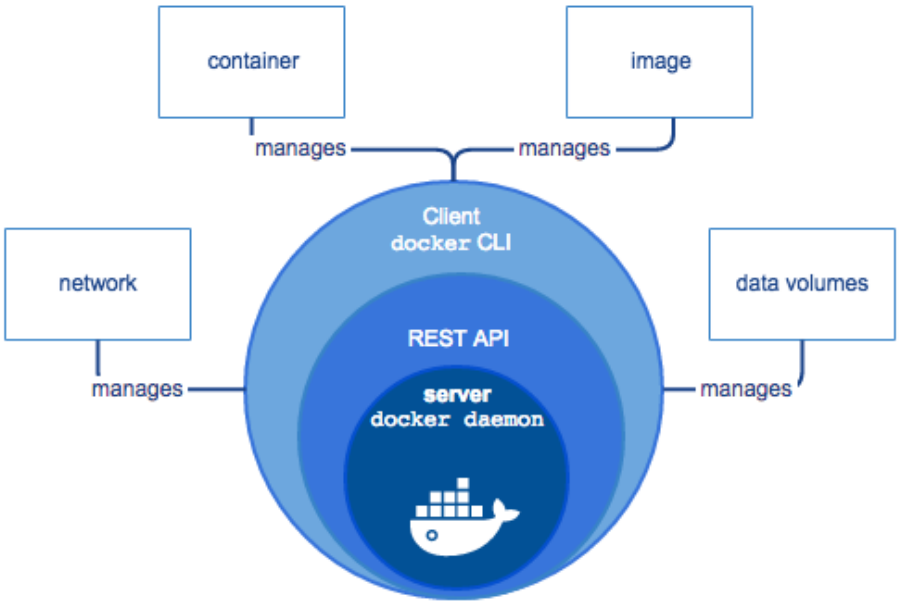
تفاوت اصلی در معماری آنها نهفته است که در زیر نشان داده شده است.تفاوت اصلی در معماری آنها نهفته است که در زیر نشان داده شده است.



3. داکر، ابزار و بستری را برای مدیریت چرخه‌ی عمر کانتینرها فراهم می کند

وقتی برنامه ی شما به عنوان یک کانتینر آماده ی ارائه بود و خواستید روی محیط اجرای نهایی آن را قرار دهید تا در دسترس مشتری هایتان باشد، دیگر فرقی ندارد که محیط اجرای تان یک هاستینگ است یا یک ارائه دهنده ی سرویس های ابری و یا حتی ترکیبی از هر دوی آنها، مراحل کاری شما دقیقا مشابه خواهد بود.

Continuous Integration یا به اختصار CI به طور خلاصه به فرآیندی اشاره دارد که از آن طریق فیچرهای جدید به صورت خودکار با ریپازیتوری اصلی ادغام می‌شوند
اما CD هم مخفف واژگان Continuous Delivery است و هم به Continuous Deployment اشاره دارد به طوری که اصطلاح اول به فرآیندی اشاره می‌کند که از آن طریق نرم‌افزار دائماً آماده دیپلوی است اما اصطلاح دوم سازوکاری است که به صورت خودکار کدهای آماده را روی سرور یا سرورهای اصلی منتشر می‌کند.



- **سرور، که یک برنامه long-running است و به آن پردازه ی Daemon گفته می شود**
- **REST API، که رابط‌هایی را مشخص می کند تا برنامه ها بتوانند از آنها برای صحبت کردن با Daemon استفاده کنند و به آن بگویند که چه کاری را باید انجام دهد.**
- **یک برنامه‌ی رابط خط فرمان (Command Line Interface)، که همان دستور Docker است.**

همانطور که در تصویر بالا مشاهده می کنید، CLI از REST API برای کنترل و ارتباط با daemon، از طریق اسکریپت و یا دستورات مستقیم CLI استفاده می کند. برنامه های مختلفی که برای مدیریت داکر مشاهده می کنید هم دقیقا از همین زیرساخت یعنی API و CLI استفاده می کنند تا خدماتی را به شما بدهند.

<u>این Daemon است که اجزای داکر، از جمله network ، container ، image و volume را می سازد و مدیریت می کند.</u>

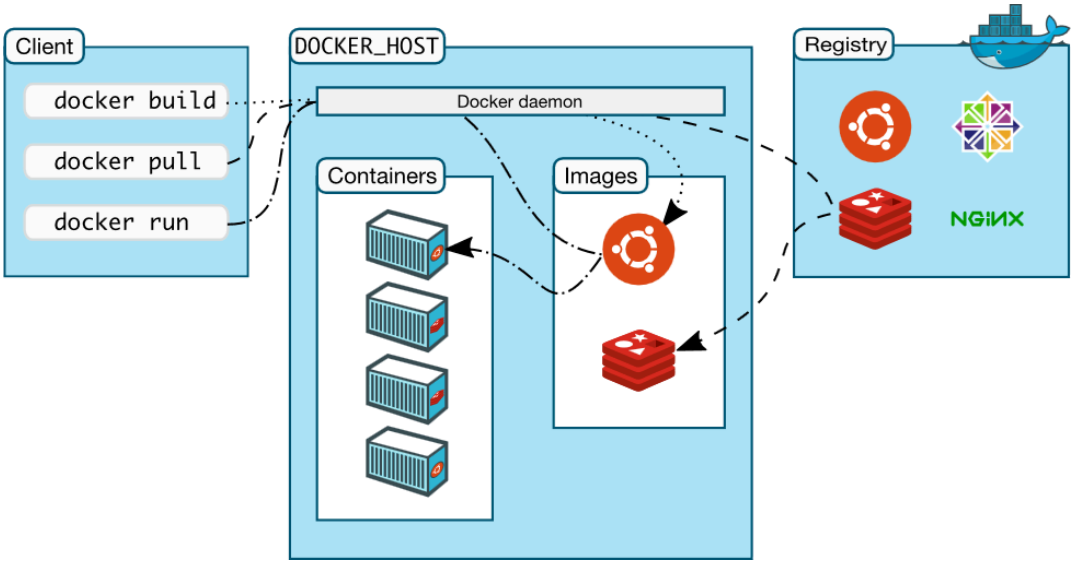


Image ها

یک Image الگوی غیر قابل تغییری است که دستور العمل‌هایی برای ساخت یک کانتینر داکر دارد. به طور عمومی، یک image براساس یک image دیگر به همراه برخی سفارشی سازی‌های اضافه است. برای مثال، شما می‌توانید image ای بر اساس ubuntu بسازید سپس وب سرور Apache و برنامه‌ی خود را روی آن نصب کنید و همچنین تنظیماتی که برای راه اندازی برنامه‌تان به آنها نیاز دارید را روی آن انجام دهید. شما ممکن است image های خود را ایجاد کنید یا از image های دیگران که در یک رجیستری به اشتراک گذاشته شده‌اند استفاده کنید. برای آنکه خودتان بتوانید یک image بسازید باید یک Dockerfile ساخته و در آن، با قواعدی ساده، قدم‌هایی که برای ساخت و اجرای image نیاز است را معرفی کنید. هر دستور العمل داخل Dockerfile یک لایه در image اضافه می‌کند. وقتی که شما Dockerfile را تغییر می‌دهید و مجدد image را می‌سازید فقط لایه‌هایی که در Dockerfile تغییر کرده‌اند، دوباره ساخته می‌شوند.

کانتینر ها

کانتینر یک نمونه‌ی قابل اجرا از image است. شما با استفاده از API داکر یا CLI آن می‌توانید کانتینر ها را ایجاد (Create)، راه اندازی (Start)، متوقف (Stop) ، جابجا (Move) یا حذف (Delete) کنید. می‌توانید یک کانتینر را به یک یا چند شبکه متصل کنید، حافظه (Storage) ای را به آن اضافه کنید یا براساس وضعیت فعلی آن، Image ای را از روی آن بسازید. به صورت پیش فرض ، یک کانتینر به خوبی از کانتینرهای دیگر و ماشین میزبانش جدا (ایزوله) شده است. البته شما می‌توانید چگونگی و حدود جدا شدن یک کانتینر از کانتینرهای دیگر، ماشین میزبانش و سیستم‌های زیر مجموعه‌ی آن را ، یا شبکه ای که به آن متصل است یا حافظه ای که به آن الصاق شده است را مدیریت کنید. یک کانتینر با image ای که از آن ساخته شده است و گزینه های تنظیماتی که در هنگام ایجاد یا شروع به کار آن برایش تعریف کرده اید معرفی می‌شود. وقتی یک کانتینر حذف می‌شود، هر تغییری که در آن اتفاق افتاده باشد ولی در حافظه ی دائمی آن ذخیره نشده باشد از بین می‌رود.

فرایند اجرای دستور docker run

```
docker run -i -t ubuntu /bin/bash (--interactive, --tty)
```

```
docker run -d -p 80:80 docker/getting-started
```

راه اندازی برنامه با داکر

```
git clone https://github.com/G2Tech-co/docker-getting-started.git
```

```
docker build -t getting-started .
```

```
docker run -dp 3000:3000 getting-started
```

```
docker ps -a
```

```
docker stop <the-container-id>
```

```
docker rm <the-container-id>
```

اشتراک گذاری برنامه ی داکری

```
docker login -u YOUR-USER-NAME
```

```
docker tag getting-started YOUR-USER-NAME/getting-started
```

```
docker push YOUR-USER-NAME/getting-started
```

پایدار کردن دیتابیس (Persisting Database) برنامه ی داکری

```
docker volume create todo-db
```

```
docker run -dp 3000:3000 -v todo-db:/etc/todos getting-started
```

```
docker volume inspect todo-db
```

یک برنامه با چند کانتینر (Multi container apps)

```
docker network create todo-app
```

```
docker run -d \
  --network todo-app \
  --network-alias mysql \
  -v todo-mysql-data:/var/lib/mysql \
  -e MYSQL_ROOT_PASSWORD=secret \
  -e MYSQL_DATABASE=todos \
  mysql:5.7
```

```
docker run -it --network todo-app nicolaka/netshoot
```

```
drill mysql
```

راه اندازی یک کانتینر برای توسعه نرم افزار (Dev-Mode Container)

```
docker run -d \
  -p 3001:3000 \
  -w /app \
  -v "$(pwd):/app" \
  --network todo-app \
  -e MYSQL_HOST=mysql \
  -e MYSQL_USER=root \
  -e MYSQL_PASSWORD=secret \
  -e MYSQL_DB=todos \
  node:18-alpine \
  sh -c "yarn install && yarn run dev"
```

(src/static/js/app.js:109)

```
docker logs -f
```

```
docker exec -it <mysql-container-id> mysql -p todos
```

```
select * from todo_items;
```

دیپ دایو

```
docker -H ssh://trip images
```

```
sudo iptables -vnL
```

```
sudo iptables -t nat -vnL
```

```
docker compose up
```

```
docker system prune --all --force
```

گروه تلگرامی :

https://t.me/g2docker

لینک های مفید:

https://labs.play-with-docker.com/

https://www.docker.com/

https://www.docker.com/blog/key-insights-from-stack-overflows-2022-developer-survey/

https://survey.stackoverflow.co/2022/#most-loved-dreaded-and-wanted-tools-tech-love-dread

https://www.toptal.com/developers/gitignore

https://www.atlassian.com/microservices/cloud-computing/containers-vs-vms

https://virgool.io/G2Tech

https://dzone.com/articles/how-do-the-docker-client-and-docker-serves-work