**Docker چیست؟**

**Docker** یک پلتفرم متن‌باز است که برای توسعه، ارسال (deploy)، و اجرای برنامه‌ها در محیط‌های ایزوله‌شده به نام **کانتینر (Container)** استفاده می‌شود. کانتینرها محیط‌هایی سبک‌وزن و قابل‌حمل هستند که کد و وابستگی‌های آن را بسته‌بندی می‌کنند و اطمینان می‌دهند که برنامه در هر جایی که اجرا شود، به‌درستی کار کند.

**چرا Docker استفاده می‌شود؟**

1. **پرتابل بودن:**  
   کانتینرها شامل تمام چیزهایی هستند که یک برنامه برای اجرا نیاز دارد (کد، کتابخانه‌ها، تنظیمات). بنابراین می‌توانید یک برنامه را بدون تغییر در هر محیطی اجرا کنید.
2. **سازگاری:**  
   با Docker می‌توانید اطمینان داشته باشید که برنامه شما در توسعه‌دهنده‌ها، محیط آزمایشی و محیط تولید دقیقاً یکسان اجرا می‌شود.
3. **بهره‌وری بالا:**  
   برخلاف ماشین‌های مجازی (VMs)، کانتینرها سبک‌تر هستند و منابع کمتری مصرف می‌کنند.
4. **سرعت:**  
   کانتینرها بسیار سریع‌تر از VM‌ها راه‌اندازی می‌شوند.
5. **مدیریت بهتر:**  
   می‌توانید از Docker برای اجرای چندین سرویس در یک سرور استفاده کنید، به‌ویژه در معماری‌های مبتنی بر **میکروسرویس**.

**مفاهیم اصلی در Docker**

1. **Image (ایمیج):**  
   یک فایل تنها و ایستا که شامل تمام نیازمندی‌های نرم‌افزار (کد، ابزارها، کتابخانه‌ها و تنظیمات) است.  
   مثال: ایمیج یک وب‌سرور مانند Nginx.
2. **Container (کانتینر):**  
   نسخه‌ای از ایمیج که اجرا می‌شود. هر کانتینر یک محیط ایزوله‌شده برای اجرای برنامه است.
3. **Dockerfile:**  
   فایلی متنی که دستورات لازم برای ساخت یک ایمیج Docker را تعریف می‌کند.  
   مثال:

dockerfile

Copy code

FROM ubuntu:20.04

RUN apt-get update && apt-get install -y nginx

CMD ["nginx", "-g", "daemon off;"]

1. **Docker Hub:**  
   یک مخزن آنلاین برای ذخیره و اشتراک‌گذاری ایمیج‌های Docker.
2. **Volume (حجم):**  
   برای ذخیره داده‌ها به صورت پایدار و به اشتراک‌گذاری آن بین کانتینرها استفاده می‌شود.

**مزایای Docker**

* اجرای چندین سرویس در یک سرور.
* ایزوله‌سازی بین برنامه‌ها.
* قابلیت توسعه و آزمایش در یک محیط یکنواخت.
* راه‌اندازی سریع‌تر نسبت به VMها.
* امکان اجرای هر نوع برنامه (جاوا، پایتون، دات‌نت، و غیره).

**کاربرد Docker در پروژه‌ها**

1. **توسعه و آزمایش:**
   * ایجاد محیطی که شبیه محیط تولید باشد.
   * اشتراک‌گذاری ایمیج‌ها بین تیم‌های توسعه و عملیات.
2. **استقرار برنامه:**
   * اجرای سرویس‌های مختلف در یک سرور.
   * کاهش پیچیدگی استقرار برنامه‌ها.
3. **مدیریت میکروسرویس‌ها:**
   * جدا کردن هر سرویس در یک کانتینر مجزا.

**دستورات مهم در Docker**

1. **ساخت ایمیج:**

bash

Copy code

docker build -t my-image .

* + -t: نام ایمیج.

1. **اجرای کانتینر:**

bash

Copy code

docker run -d -p 8080:80 my-image

* + -d: اجرای کانتینر در پس‌زمینه.
  + -p: نگاشت پورت میزبان به کانتینر.

1. **مشاهده کانتینرهای فعال:**

bash

Copy code

docker ps

1. **مشاهده تمام کانتینرها (فعال و غیرفعال):**

bash

Copy code

docker ps -a

1. **توقف کانتینر:**

bash

Copy code

docker stop container\_id

1. **حذف کانتینر:**

bash

Copy code

docker rm container\_id

1. **حذف ایمیج:**

bash

Copy code

docker rmi image\_id

**کاربرد Docker در ASP.NET Core**

**ساخت Dockerfile برای پروژه ASP.NET Core:**

1. فایل Dockerfile را در ریشه پروژه خود ایجاد کنید:

dockerfile

Copy code

FROM mcr.microsoft.com/dotnet/aspnet:6.0 AS base

WORKDIR /app

EXPOSE 80

EXPOSE 443

FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build

WORKDIR /src

COPY ["MyApp/MyApp.csproj", "MyApp/"]

RUN dotnet restore "MyApp/MyApp.csproj"

COPY . .

WORKDIR "/src/MyApp"

RUN dotnet build "MyApp.csproj" -c Release -o /app/build

FROM build AS publish

RUN dotnet publish "MyApp.csproj" -c Release -o /app/publish

FROM base AS final

WORKDIR /app

COPY --from=publish /app/publish .

ENTRYPOINT ["dotnet", "MyApp.dll"]

1. **ساخت ایمیج Docker:**

bash

Copy code

docker build -t my-aspnetcore-app .

1. **اجرای کانتینر:**

bash

Copy code

docker run -d -p 8080:80 my-aspnetcore-app

1. **مشاهده برنامه در مرورگر:**  
   باز کردن http://localhost:8080

**جمع‌بندی**

Docker یک ابزار قدرتمند برای توسعه، آزمایش و استقرار برنامه‌ها است. در پروژه‌های ASP.NET Core، استفاده از Docker باعث ایجاد محیط‌های قابل‌حمل، ساده‌تر شدن استقرار، و مدیریت سرویس‌های مختلف می‌شود. اگر سوال یا مشکلی در پیاده‌سازی دارید، خوشحال می‌شوم کمک کنم