📦 تقليل حجم الـ Docker Image باستخدام Multi-Stage Builds  
في عالم الـ Docker، الهدف الرئيسي هو توفير الموارد سواء من حيث المساحة أو الوقت. لما عملت أول Dockerfile لتشغيل التطبيق، كان شغال تمام، لكن اكتشفت إن حجم الصورة (image) كان كبير جدًا - حوالى 213 ميجابايت! 🙄  
  
عشان كده قررت أستخدم multi-stage builds، واللي فكرتها ببساطة هي تقسيم عملية بناء الصورة إلى مراحل. الفكرة هنا إنك تقوم بعمل كل الخطوات اللازمة في مرحلة البناء (build)، وبعدها تأخذ الحاجات الضرورية فقط لتشغيل التطبيق في مرحلة الإنتاج (production). النتيجة؟ صورة أصغر وأكفأ - حجمها بعد التعديل بقى 145 ميجابايت!  
  
الـ Dockerfile قبل التعديل (بدون Multi-Stage)  
  
FROM alpine   
RUN apk add --update nodejs npm   
COPY . /src   
WORKDIR /src   
RUN npm install   
EXPOSE 8080   
ENTRYPOINT ["node", "./app.js"]  
المشكلة:  
الصورة كانت كبيرة لأن كل الأدوات والحزم اللي استخدمتها أثناء عملية البناء موجودة في الصورة، حتى لو مش هتحتاجها لتشغيل التطبيق  
  
الـ Dockerfile بعد استخدام Multi-Stage Builds  
# Build stage  
  
FROM alpine AS build  
RUN apk add --no-cache nodejs npm curl  
COPY . /src  
WORKDIR /src  
RUN npm install  
  
# Prod stage  
  
FROM alpine AS prod  
RUN apk add --no-cache nodejs curl  
COPY --from=build /src /src  
WORKDIR /src  
EXPOSE 8080  
ENTRYPOINT ["node", "./app.js"]  
  
الفكرة في Multi-Stage Builds  
في مرحلة البناء (build):  
بتثبت كل الأدوات والحزم اللي محتاجها للبناء.  
بتعمل npm install لتثبيت الحزم.  
في مرحلة الإنتاج (prod):  
بتنسخ الملفات النهائية (مع الحزم المثبتة) من مرحلة البناء للصورة النهائية.  
النتيجة؟ الصورة النهائية بتحتوي على الحاجات الضرورية بس لتشغيل التطبيق، وده بيقلل حجمها بشكل كبير.