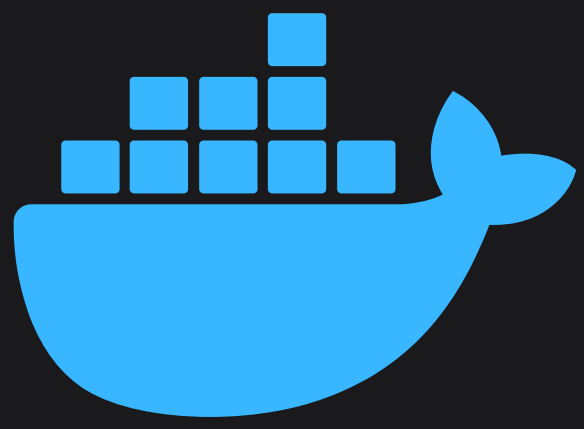
بهینه کردن اندازه

docker image

برای پروژه های TS



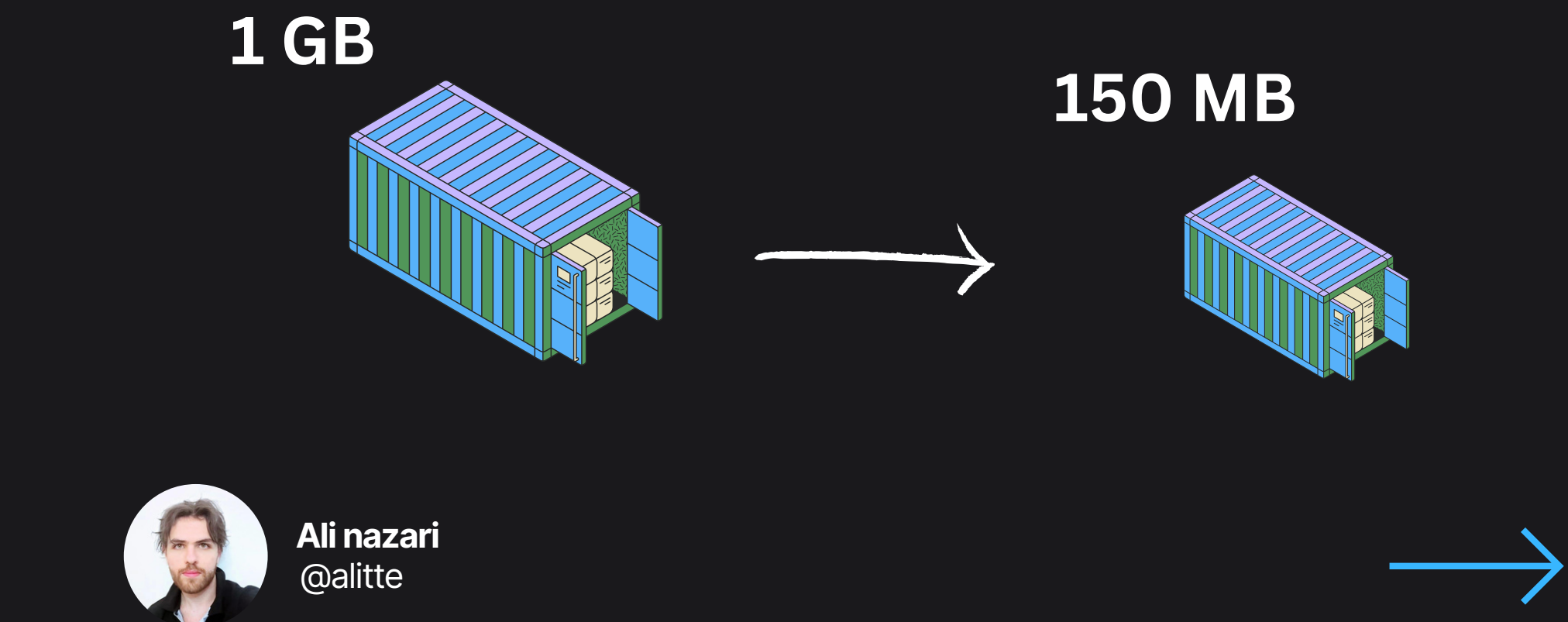
Ali nazari
@alitte

اگه تجربه داکرایز کردن یه پروژه بزرگ رو که با تایپ اسکریپت نوشته شده باشه رو داشته باشین

متوجه شدین که image که در نهایت ساخته میشه فوق العاده **حجیمه** 📦📦

ایمیج های کوچک تر منجر به **سریع تر شدن پروسه build** و **دیپلوی سریع تر** میشن

همچنین این موضوع برای CI/CD pipline هایی که تعریف میشه هم خیلی میتونه سودمند باشه



Ali nazari
@alitte

روش هایی که میشه اندازه ایمیج رو بهینه کرد

- **پاک کردن npm cache**

npm برای این که هر بار پکیج های تکراری و پر استفاده شما رو از npm registry دانلود نکنه میاد داخل یه دایرکتوری پکیج هارو کش میکنه

و در صورت وجود پکیج شما رو از همون دایرکتوری لود میکنه

آدرس دایرکتوری

~/.npm (**Linux**)

%AppData%/npm-cache (**windows**)

```
RUN npm cache clean --force
```

Ali nazari
@alitte

- **استفاده از &&**

به جای نوشتن چند تا دستور RUN میشه همه دستورات رو با یه دستور RUN تعریف کرد

کم کردن تعداد دستورات RUN میتونه به کاهش image شما منجر بشه چون هر کدوم از این دستورات یه لایه به image شما اضافه میکنن

پس از این :

```
RUN apt-get update
RUN apt-get install vim
```

به این میرسیم :

```
RUN apt-get update && apt-get install vim
```

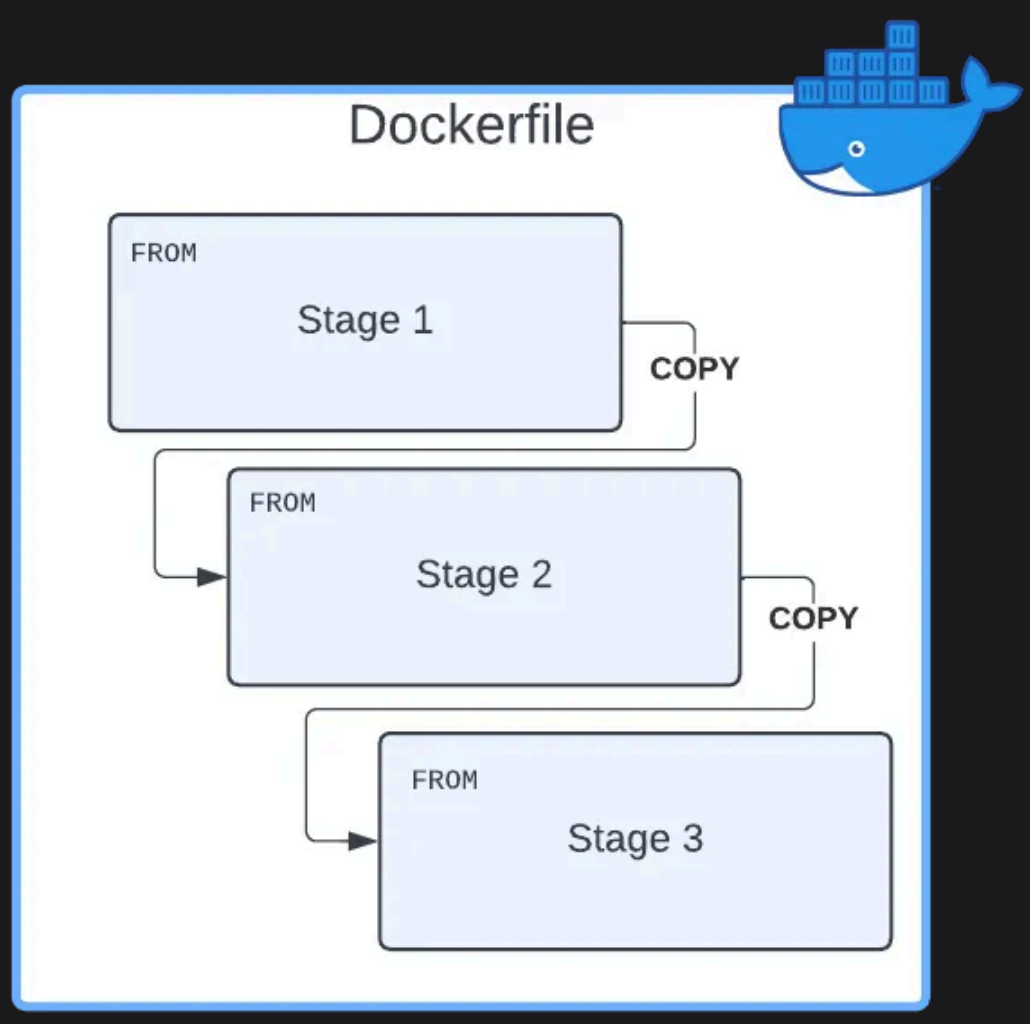
Ali nazari
@alitte

- **استفاده از Multistage docker builds**

docker multistage چیه؟

یه قابلیت هستش که به شما اجازه میده داخل Dockerfile از دو تا دستور FROM استفاده کنین

پس ما یه stage برای **ساختن** نرم افزار خودمون تعریف میکنیم و بعد در مرحله بعد بخش های حیاتی که نرم افزار فقط نیاز داره رو میبریم به stage بعدی (runner)



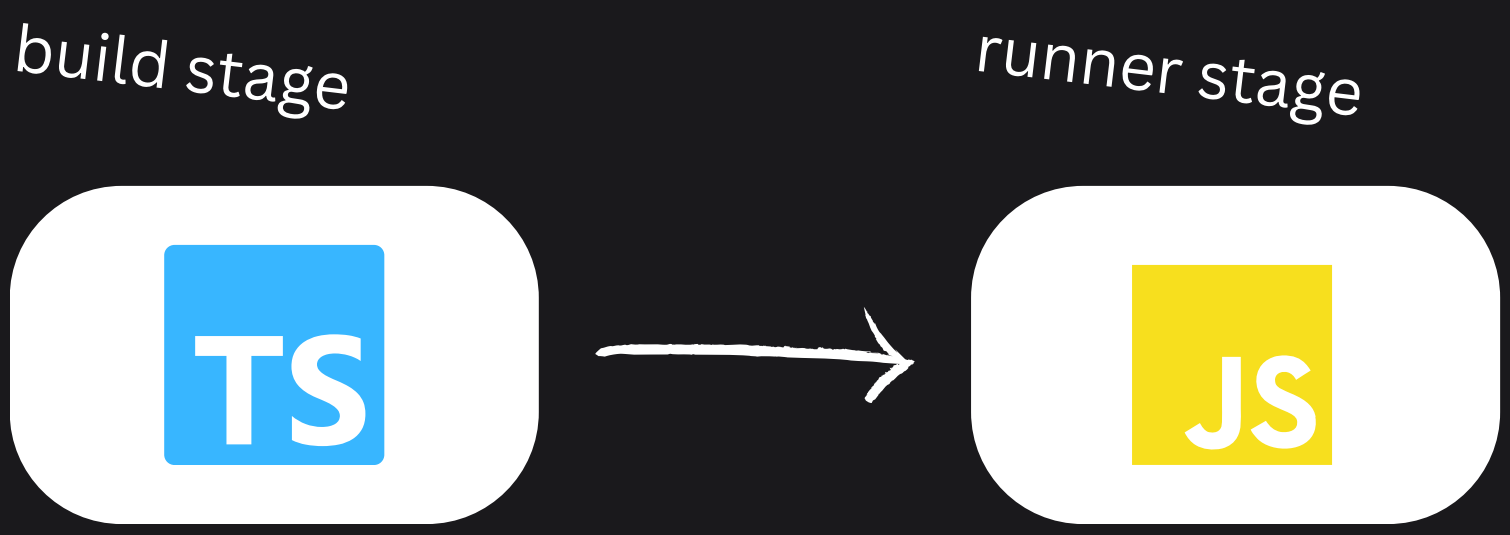
Ali nazari
@alitte

داخل پروژه های تایپ اسکریپتی ما نیاز به تایپ اسکریپت و ابزار های توسعه مربوط بهش رو نداریم

یا کلا نیازی به هیچکدوم از پکیج هایی که تو حالت dev-dependency نصب شدن نیازی نداریم !

ما میتونیم بگیم در **build stage** تایپ اسکریپت کامپایل بشه و فایل های js رو درست کنه

و تو stage نهایی که بهش runner stage میگن بیایم فقط جاوااسکریپت رو اجرا کنیم



Ali nazari
@alitte

```dockerfile
#Build stage
FROM node AS build

WORKDIR /app

COPY package*.json .

RUN npm install

COPY . .

RUN npm run build

#Production stage
FROM node AS production

WORKDIR /app

COPY package*.json .

RUN npm ci --only=production

COPY --from=build /app/dist ./dist

CMD ["node", "dist/index.js"]
```

بهتره از ci به جای npm install استفاده کنین. برای این که با زدن این دستور همون ورژنی از وابستگی ها که در محیط توسعه نصب شدن نصب میشن

Ali nazari
@alitte

و اگر یکی از وابستگی ها ارتقا پیدا کرده بود ورژن جدیدش براتون نصب نمیشه !🚫

- **استفاده از slim version**

ما در اینجا از base image به اسم node استفاده کردیم که داره از ابونتو استفاده میکنه که خیلی سنگینه

اگر از نسخه های سبک تر مثل node-alpine استفاده کنیم کاهش حجم خیلی زیادی اتفاق میفته

```
raunak@hp:/mnt/d/workspace/coding/web/deployment/Docker/multi-stage-build$ docker image ls | grep -i ts-docker
ts-docker/alpine                latest    e63392d57d8d   8 minutes ago       147MB
ts-docker/multistage            latest    05aaaf915610   20 minutes ago      1.1GB
ts-docker/single                latest    b28b5423b73d   About an hour ago   1.14GB
```

Ali nazari
@alitte

- **حذف گیت**

در اخر اگر مطمئن هستین که نمیخواین داخل کانتینر از گیت استفاده کنین حذف کردنش میتونه حجم قابل توجهی از ایمیج رو کم کنه

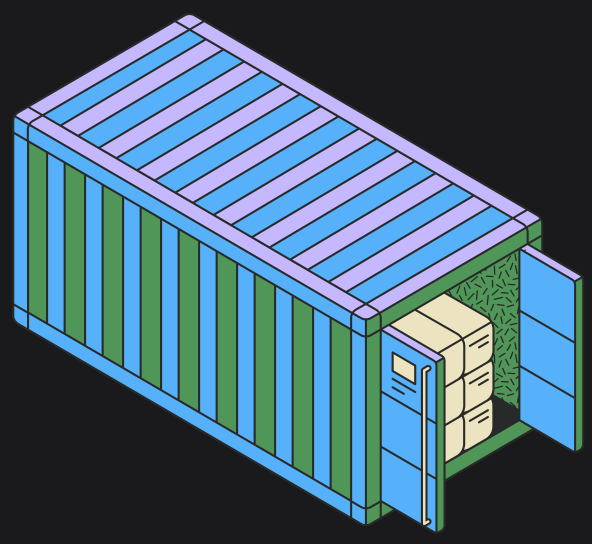
```
# Remove Git
RUN apk del git
```

به چه چیز هایی رسیدیم؟💡📦

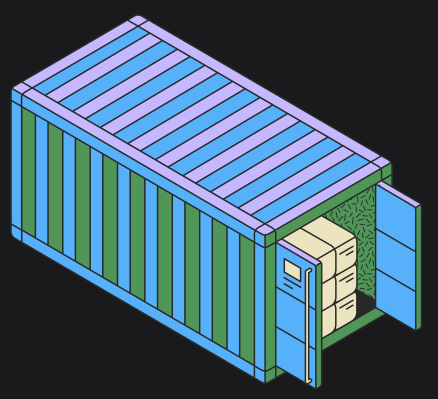
- حجم کم تر image
- اعمال best practice ها و سادگی در کانفیگ داکر
- امنیت (alpine علاوه بر سبکی تمرکز ویژه ای بر روی امنیت داره)

Ali nazari
@alitte

1 GB



150 MB



TS

از **لایک های دل گرم کننده** ای که قراره بزارین و یا **کامنت های دل سرد کننده** شما ممنونم

Ali nazari
@alitte