Docker Building Go

> `go mod download` considered unnecessary

Nicholas Capo

Senior Infrastructure Engineer

Axios, Inc

Overview

1. Conventional Wisdom
2. But Go
3. A Better Way
4. Results

Assumptions

- We are using Docker to build our Go applications
- The docker build is from source, meaning we aren't copying the binary into the image (e.g. `ko`)

Conventional Wisdom

Node / Python / Ruby / Erlang / etc

```
FROM node:24
WORKDIR /app

COPY package.json package-lock.json ./
RUN npm install

COPY . .
RUN npm run build
```

The Go Version

```
FROM golang:1.24 AS builder
WORKDIR /app

COPY go.mod go.sum ./
RUN go mod download

COPY . .
RUN go build -o main .

# TODO: use distroless
FROM scratch AS runtime
COPY --from=builder /app/main ./
```

Why?

- Docker layer cache (avoid invalidation)
- `npm` is a tool we use for package management in the `node` ecosystem

But Go

- Go doesn't have a split between the runtime/compiler and the package manager

	Dependencies	Testing	Build/Compile	Runtime
Node	`npm`	`jest`	`npm` / `node`	`node`
Go	`go`	`go`	`go`	[binary]

A Better Way

```
FROM golang:1.24 AS builder
WORKDIR /app

- COPY go.mod go.sum ./
- RUN go mod download

COPY . .
RUN go build -o main .

# TODO: use distroless
FROM scratch AS runtime
COPY --from=builder /app/main ./
```

Advantages

- Faster to build (!)
  - Avoid downloading unused dependencies
  - Avoid building unused dependencies
  - Download and Build in parallel(?)
- Fewer layers (the same invalidation)

In the real world

- Monorepo (workspace)
- Multiple services
- Multiple languages
- Shared common libraries (Go and other)

Build with `lib`

```
FROM golang:1.24 AS builder

WORKDIR /lib
COPY lib/go.mod lib/go.sum /lib/
RUN go mod download

COPY lib /lib/

WORKDIR /app
COPY app/go.mod app/go.sum /app/
RUN go mod download

COPY app /app/
RUN go build -o main .

# TODO: use distroless
FROM scratch AS runtime
COPY --from=builder /app/main ./
```

Change

```
FROM golang:1.24 AS builder

- WORKDIR /lib
- COPY lib/go.mod lib/go.sum /lib/
- RUN go mod download

COPY lib /lib/

WORKDIR /app
- COPY app/go.mod app/go.sum /app/
- RUN go mod download

COPY app /app/
RUN go build -o main .

# TODO: use distroless
FROM scratch AS runtime
COPY --from=builder /app/main ./
```

Results

Service	Before	After	Change
`core`	116.0s	89.2s	-23%
`queue`	80.8s	38.3s	-52%
`runner`	120.6s	49.9s	-58%

Service: `core`

Steps	34	→	29
Total	116.0s	→	89.2s
go mod download	16.2s		
go mod download	26.3s		
go build ./app	54.8s	→	68.9s
go build ./app/cron-1	4.2s	→	2.0s
go build ./app/cron-2	1.9s	→	1.9s
go build ./app/cron-3	2.0s	→	2.3s
go build ./app/cron-4	1.8s	→	2.0s
go build ./app/cron-5	2.2s	→	2.8s

Service: `queue`

Steps	24	→	18
Total	80.8s	→	38.3s
go mod download	16.9s		
go mod download	20.6s		
go build ./app	23.9s	→	33.0s
go build ./app/cmd-1	13.9s	→	2.0s

Service: `runner`

Steps	27	→	20
Total	120.6s	→	49.9s
go mod download	21.3s		
go mod download	23.9s		
go build ./app	39.9s	→	44.9s
go build ./app/cmd-1	15.8s	→	1.6s

Go forth and `go build`

https://github.com/nicholascapo/talk-docker-building-go