



DESTROY Long Build Times

using Docker, Go, Java, Bazel & Codefresh

Thanks!



DAN GARFIELD



GUY SALTON



KOSTIS
KAPELONIS

Dan Garfield

Chief Technology Evangelist

 **codefresh**



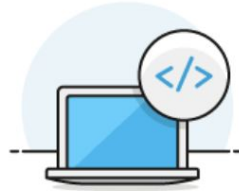
Why speed up builds?

100 Engineers



X

5 Builds/Day



X

Saving 5
Min/Build



Making Average
\$150k/yr

Annual Savings

\$781k

Intangible Benefits



Happy Devs



Creativity Flourishes



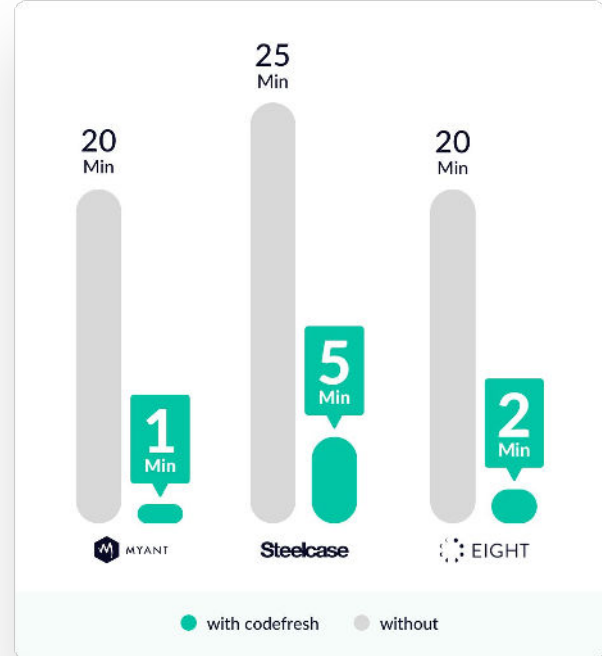
Faster Iteration

What you'll learn to day

How to strategically use distributed caching

Tips for optimizing Docker builds

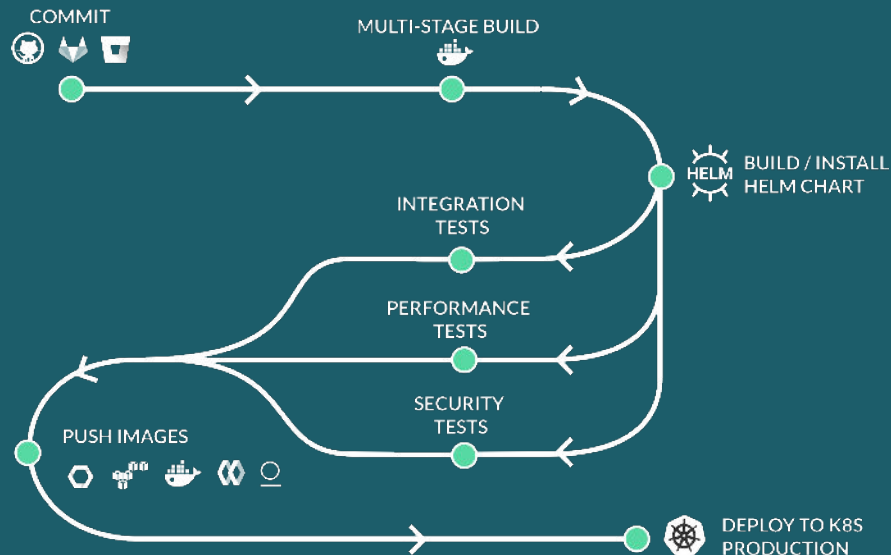
Why you need multi-stage builds



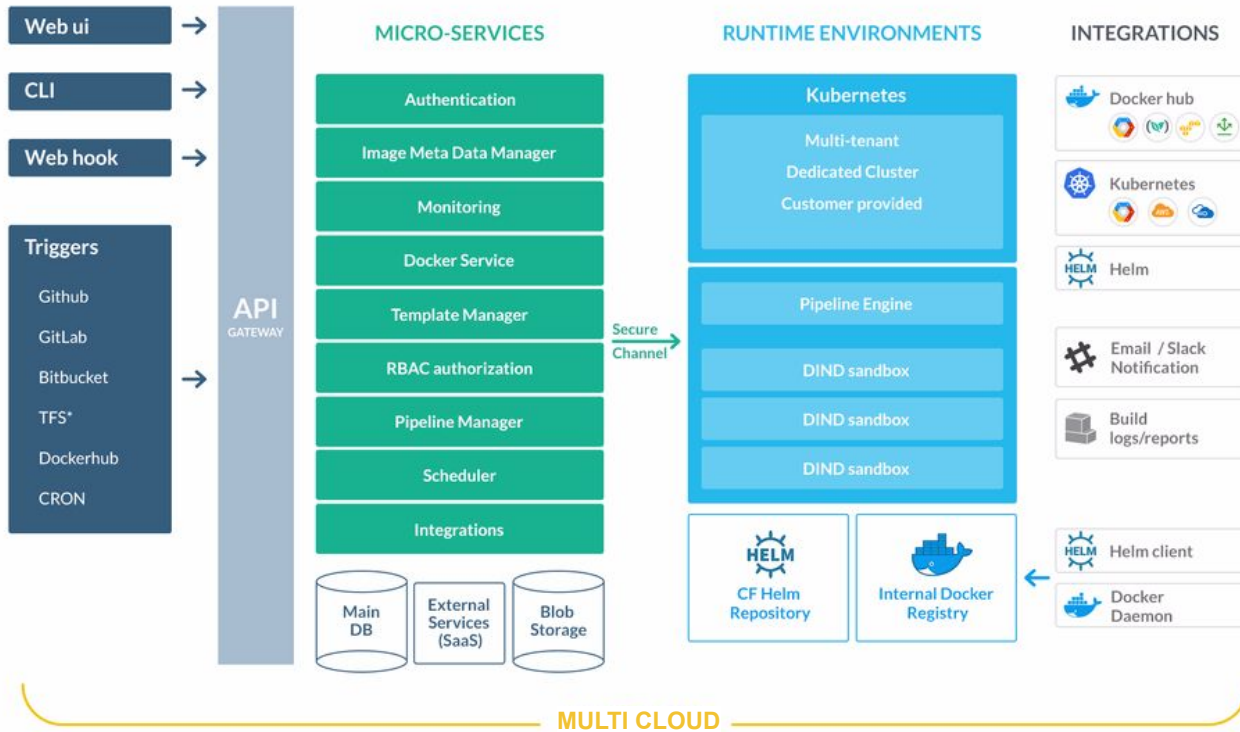
Codefresh

The 1st container-native
CI/CD Platform for
Microservices

-  **Cloud-native**
-  **Intuitive & Robust**
-  **Enterprise Ready**
-  **Flexible Delivery**



CODEFRESH ARCHITECTURE DIAGRAM



Why do CI systems usually struggle with speed?

Jest much slower in [redacted] CI than locally #7647

Open milesj opened this issue on Jan 17 · 22 comments



milesj commented on Jan 17 • edited ↕



Bug Report

Why do my tests take longer to run on [redacted] CI than locally?



Kyle Tryon
April 20, 2018 11:20

Local Build Isn't the Answer

Blocks Dev from working

Not secure or traceable

Not reliable

Why do CI systems usually struggle with speed?

Build Node

Build Node

Build Node

Build Node

Build Node

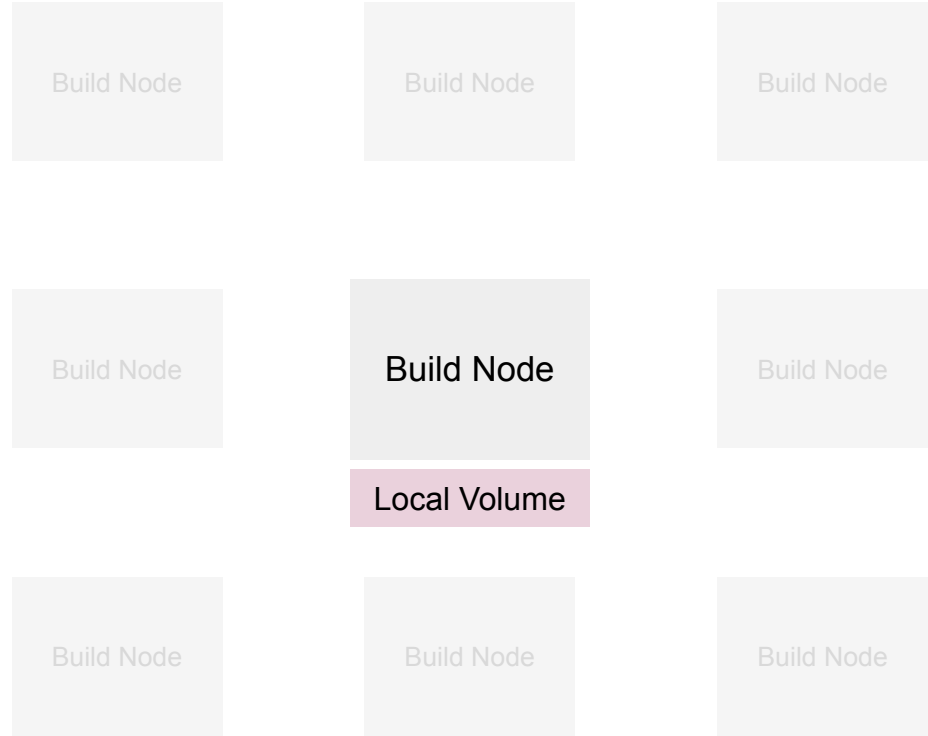
Build Node

Build Node

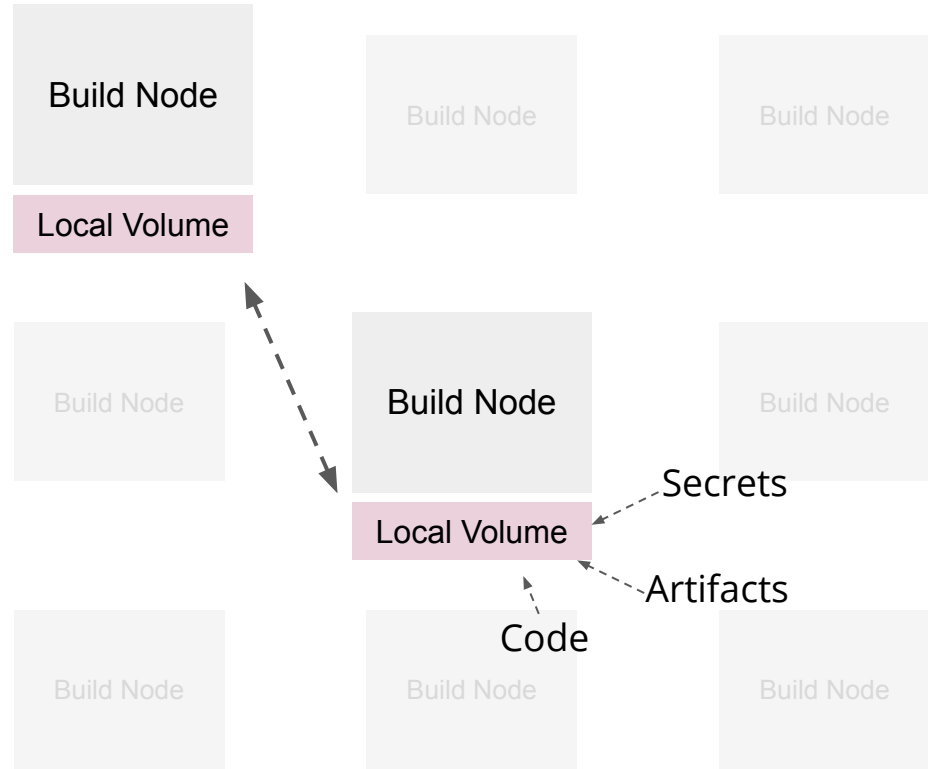
Build Node

Build Node

Why do CI systems usually struggle with speed?



Why do CI systems usually struggle with speed?



CODEFRESH BUILD OPTIMIZATION

Every step a container

Codefresh attaches build volumes

Step 1

Step 2

Step 3

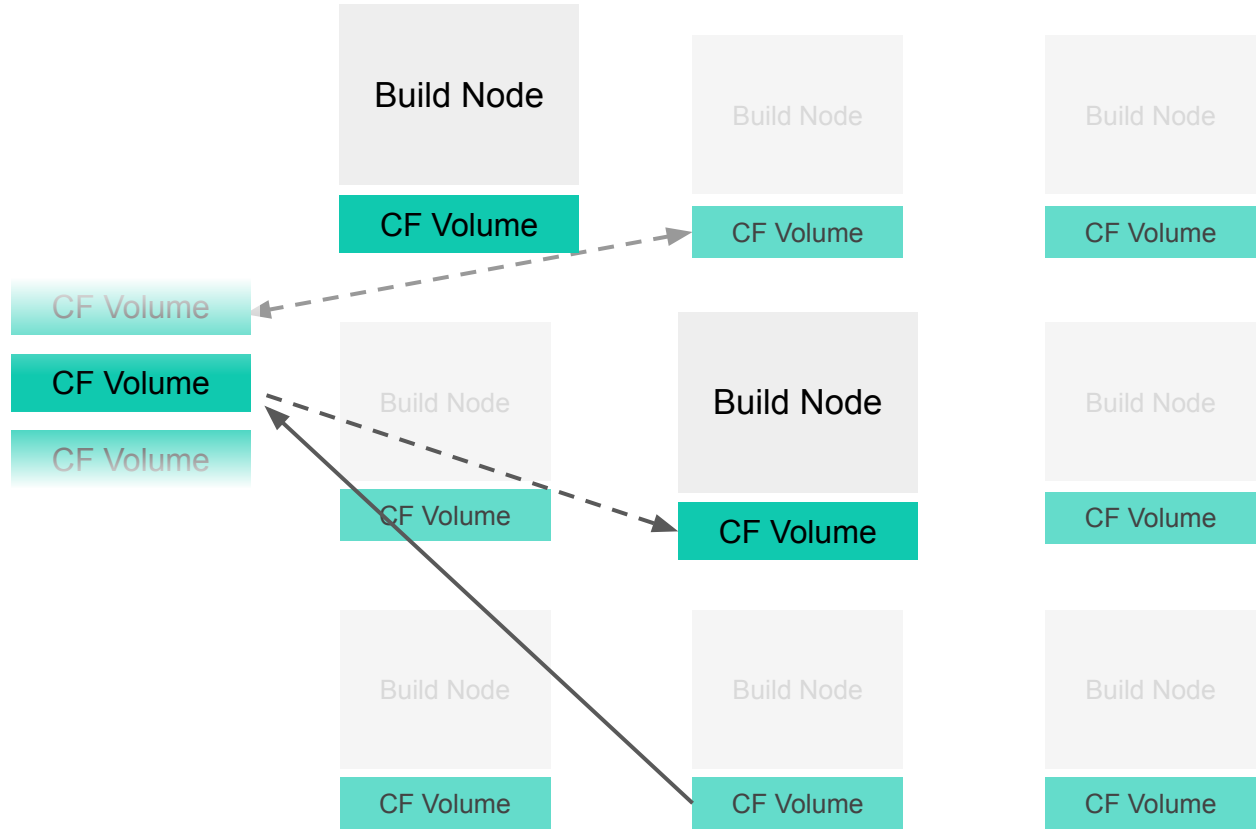
Step 4



Build Volume



Codefresh Distributed Caching



Demo 1: Using Distributed Cache

<https://q.codefresh.io/build/5d831fed577e2f81e4569104>

Demo 2: Running Locally

```
$ codefresh run 'Go Speed/Hugo Build' --local
```

My code change



Rebuilt
dependencies



80-90%
Build Time
is Wasted!

My code change



Only Rebuild the Part that changes

- Cached Docker Layers
- Gradle Cache
- Go Cache
- Bazel



Caching

Go has GOPATH

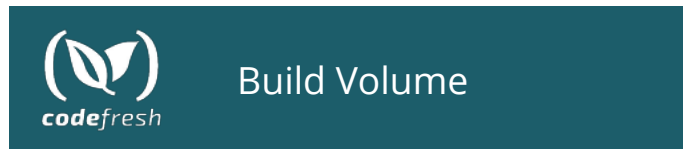
Using `CF_VOLUME_PATH` for your go path with cache the modules.

Java has Gradle Cache

Using `CF_VOLUME_PATH` for your gradle cache.

Bazel has Cache

Using `CF_VOLUME_PATH` for your gradle cache.



Caching

Go has GOPATH

Using `${CF_VOLUME_PATH}` for your go path with cache the modules.

Java has Gradle Cache

Using `${CF_VOLUME_PATH}` for your gradle cache.



Bazel has Cache

Using `${CF_VOLUME_PATH}` for your gradle cache.

<https://github.com/todaywasawesome/bazelbuild-examples>

Demo 3: Using App Cache

<https://g.codefresh.io/build/5d82fb46c23d4552518b95e3>

Demo 4: Optimizing Layers

<https://g.codefresh.io/build/5d832c75353ca93b16374775>

Docker multi-stage build

Multi-stage build is available starting from Docker 17.05 (released in 2017!) - so why now?

↑ 1 ↓	Generate next closest timestamp based on WeekDay (ex. Sunday), Hour and Minute Posted by u/sh4rk1z 22 hours ago	5
↑ 3 ↓	Casting between slices of interfaces Posted by u/Pockensuppe 1 day ago	9
↑ 33 ↓	Algorithm examples in Go and in other languages Posted by u/nwss00 1 day ago	7
↑ 0 ↓	Golang high cpu usage for websockets? Posted by u/tlewis334 1 day ago	7
↑ 120 ↓	Docker & Golang - reducing container size using multi-stage builds Posted by u/johnmidd 2 days ago	59
↑ 5 ↓	Joined a project, team keeps all their enums in one package, causes confusion Posted by u/cuteTiger 1 day ago	6
↑ 0 ↓	Help with my code Posted by u/Mineges 1 day ago	3
↑ 0 ↓	Help needed with: missing ';' before newline in composite literal Posted by u/ludikoff 23 hours ago	12

https://www.reddit.com/r/golang/comments/crkibq/docker_golang_reducing_container_size_using/

- “oh man! thank you! I've been fighting with build times on an image stack for weeks”
- “Wow, this is great for deploying tonnes of microservices!”
- “I've been meaning to use multistage builds, thanks for the walkthrough!”

Dockerfile and Docker build

- **Dockerfile** - imperative DSL that defines build commands
- Each **Docker build** command generates ONE image layer
- Complete **Docker build** execution generates ONE Docker image

Dockerfile and Docker build

```
1 FROM golang:1.7.1
2
3 # Copy everything from the src directory to /go/src directory inside the container
4 COPY src /go/src
5
6 # Build the Go app
7 RUN CGO_ENABLED=0 GOOS=linux go build -o bin/sample src/sample/trivial-web-server.go
8
9 # This container exposes port 8080 to the outside world
10 EXPOSE 8080
11
12 # Run the binary program
13 CMD ["/bin/sample"]
```

Demo 5: Docker build on GO app

<https://github.com/codefresh-contrib/helm-sample-app>

The Problem with Docker build

Image we want

runtime
configuration
application

X (4..10)

Image we build

Compilers, debuggers, ...
Linters, tests, profilers, ...
code, build and test logs, ...
runtime
configuration
application

The Problem with Docker build

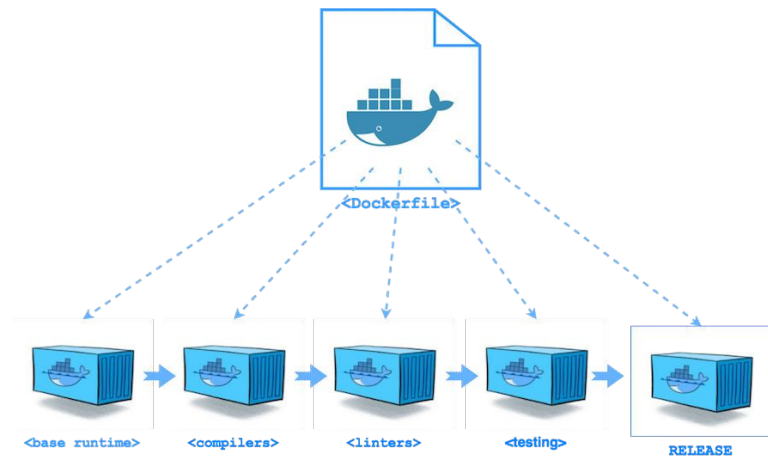
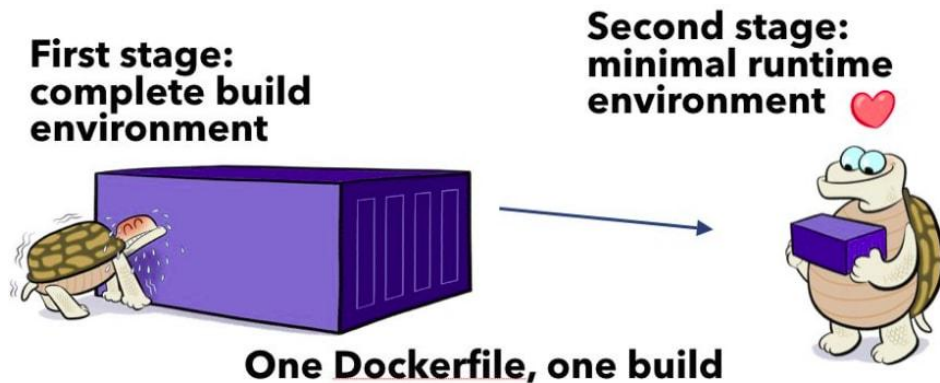
- **2 Dockerfiles**
 - 1st for build tools
 - 2nd for runtime

- **Drawbacks**
 - 2+ Dockerfiles
 - Orchestration needed: Bash, make, YAML, ...

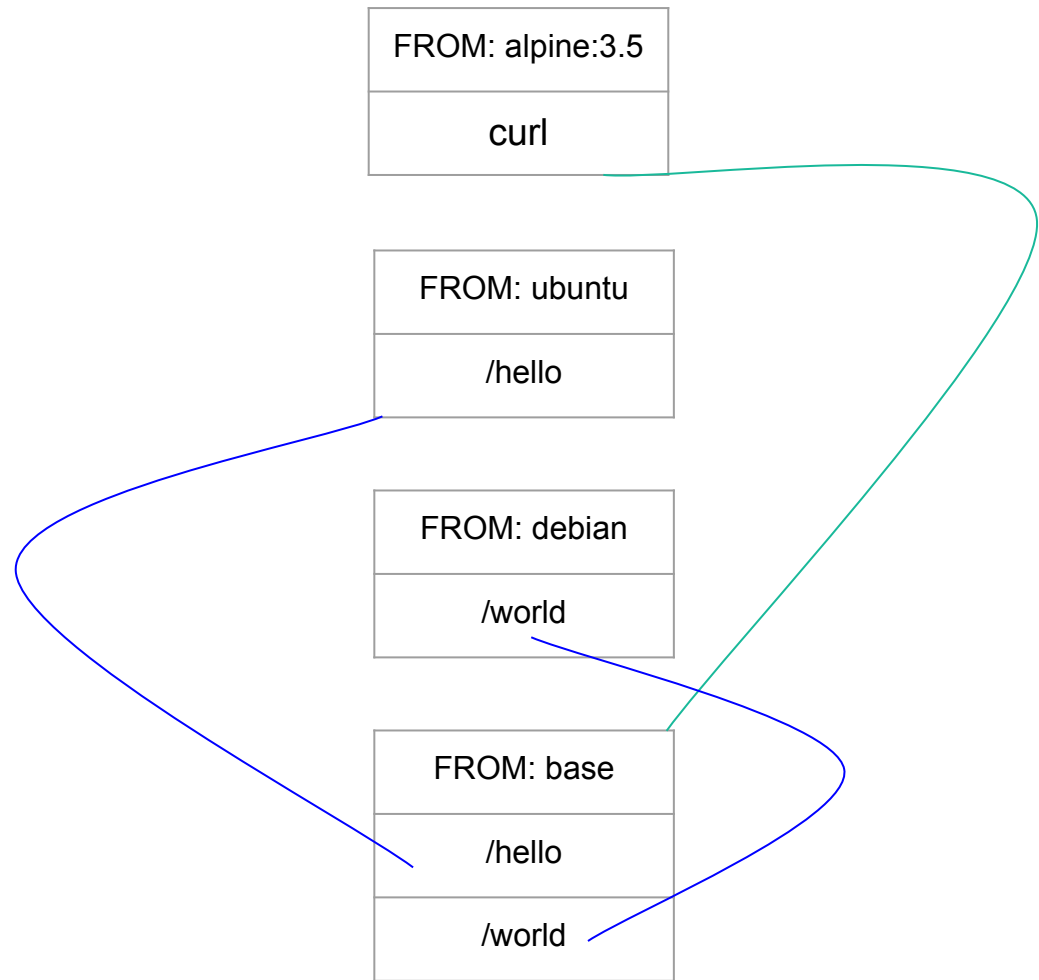
Solution: Docker multi-stage build

- **Benefits**

- One Dockerfile
- One syntax to learn
- Same build
 - Local and CI
- Create multiple stages







```
1 # Base Image
2 FROM alpine:3.5 AS base
3 RUN apk add --no-cache curl
4
5 # Second Image
6 FROM debian AS second
7 RUN echo hello > /hello
8 LABEL image=second
9
10 # Third Image
11 FROM ubuntu AS third
12 RUN echo world > /world
13 LABEL image=third
14
15 # FINAL Image
16 FROM base
17 # Copy files from other images
18 COPY --from=second /hello /hello
19 COPY --from=third /world /world
20 RUN curl --version
```



Demo 6: Docker multi-stage build

Docker multi-stage build

You can enjoy multi-stage build with every programming language (not only GO):

- GO example - <https://codefresh.io/docs/docs/learn-by-example/golang/golang-hello-world/#create-a-multi-stage-docker-image-for-go> 
- JAVA example - <https://codefresh.io/docs/docs/learn-by-example/java/spring-boot-2/#spring-boot-2-and-docker-multi-stage-builds> 
- Node example - <https://codefresh.io/docs/docs/learn-by-example/nodejs/react/#react-and-docker-multi-stage-builds> 
- PHP example - <https://codefresh.io/docs/docs/learn-by-example/php/#the-example-php-project> 

Docker anti-patterns

<https://codefresh.io/containers/docker-anti-patterns/>

Summary

- Saving just 5 min is worth big \$\$
- Distributed Caching is basically free optimization
- Pair with Application Cache



Summary

- Using 1 Docker image for both build and production results in slow deployment and lots of CVE violations
- Multi-stage build to produce lean, secure and production ready Docker image
- On Codefresh, speedier builds thanks to caching across all images and layers



Questions?

Try it free at
codefresh.io



Upcoming Events

[Codefresh.io/events](https://codefresh.io/events)

