

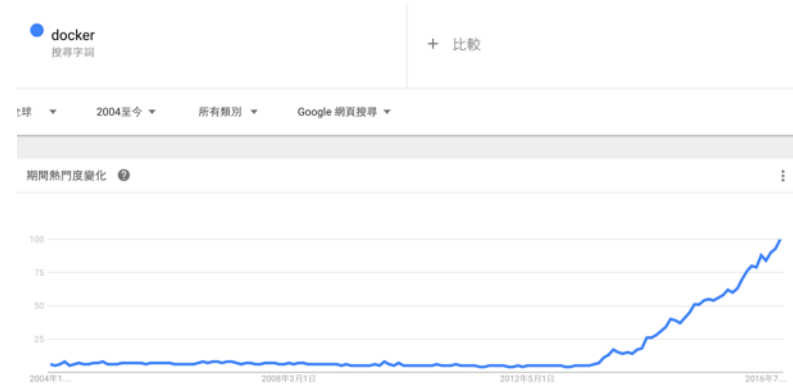
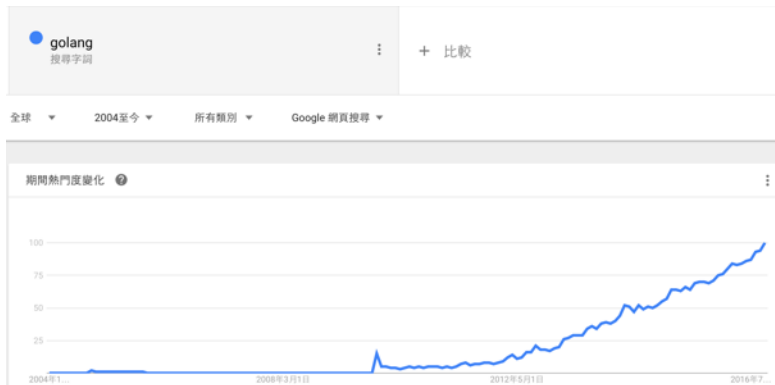
Golang x Docker

Building small Golang Docker image

KEVINGO

故事是這樣開始的

Golang X Docker



Simple Go HTTP Server

```
package main

import (
    "fmt"
    "net/http"
)

func handler(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "Hi there, I love %s!", r.URL.Path[1:])
}

func main() {
    http.HandleFunc("/", handler)
    http.ListenAndServe(":8080", nil)
}
```

Docker + Golang



Start a Go instance in your app

The most straightforward way to use this image is to use a Go container as both the build and runtime environment. In your `Dockerfile`, writing something along the lines of the following will compile and run your project:

```
FROM golang:1.6-onbuild
```

This image includes multiple `ONBUILD` triggers which should cover most applications. The build will `COPY . /go/src/app`, `RUN go get -d -v`, and `RUN go install -v`.

This image also includes the `CMD ["app"]` instruction which is the default command when running the image without arguments.

You can then build and run the Docker image:

```
$ docker build -t my-golang-app .  
$ docker run -it --rm --name my-running-app my-golang-app
```

Note: the default command in `golang:onbuild` is actually `go-wrapper run`, which includes `set -x` so the binary name is printed to stderr on application startup. If this behavior is undesirable, then adding `CMD ["app"]` (or `CMD ["myapp"]` if a [Go custom import path](#) is in use) will silence it by running the built binary directly.

寫 Dockerfile

```
FROM golang:onbuild
```

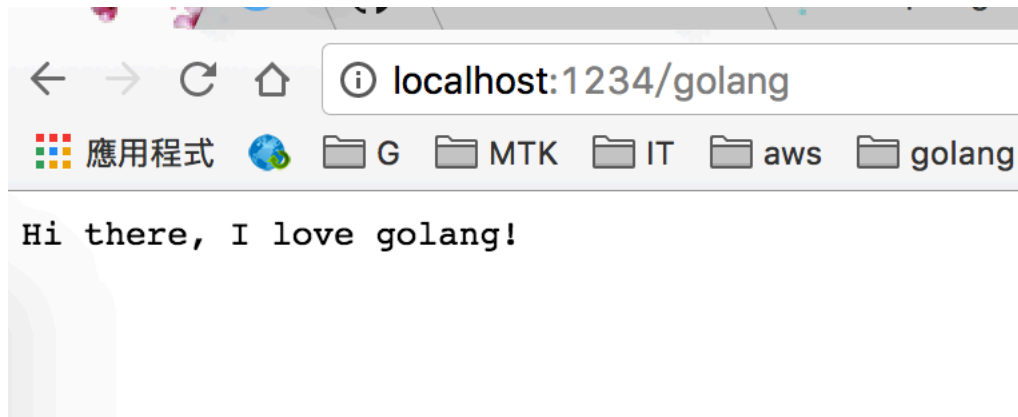
```
EXPOSE 8080
```

Build

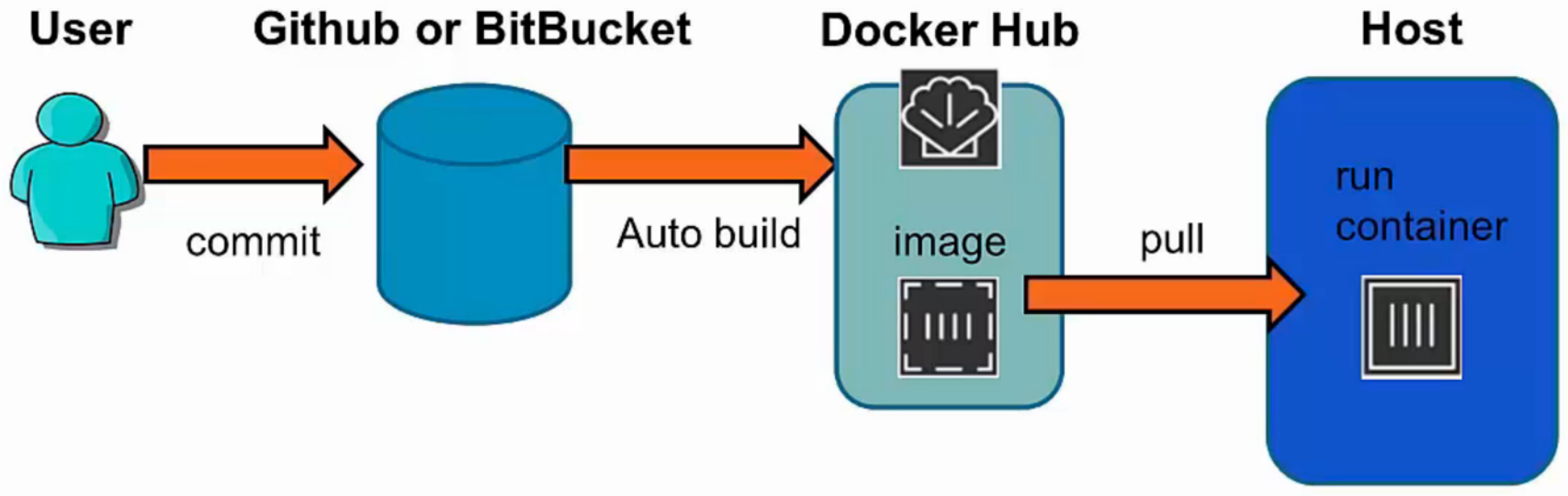
```
$ docker build -t http.onbuild -f Dockerfile.onbuild .
Sending build context to Docker daemon 4.529 MB
Step 1 : FROM golang:onbuild
onbuild: Pulling from library/golang
6a5a5368e0c2: Already exists
7b9457ec39de: Already exists
ff18e19c2db4: Already exists
00075397a1ec: Downloading [=====>] 25.41 MB/59.65 MB
c93c4c92d7de: Downloading [=====>] 27.03 MB/81.63 MB
017212650f08: Download complete
7dab2bd67056: Download complete
54c9af7c7651: Download complete
```


Everything works fine

```
$ docker run --publish 1234:8080 http.onbuild  
+ exec app
```



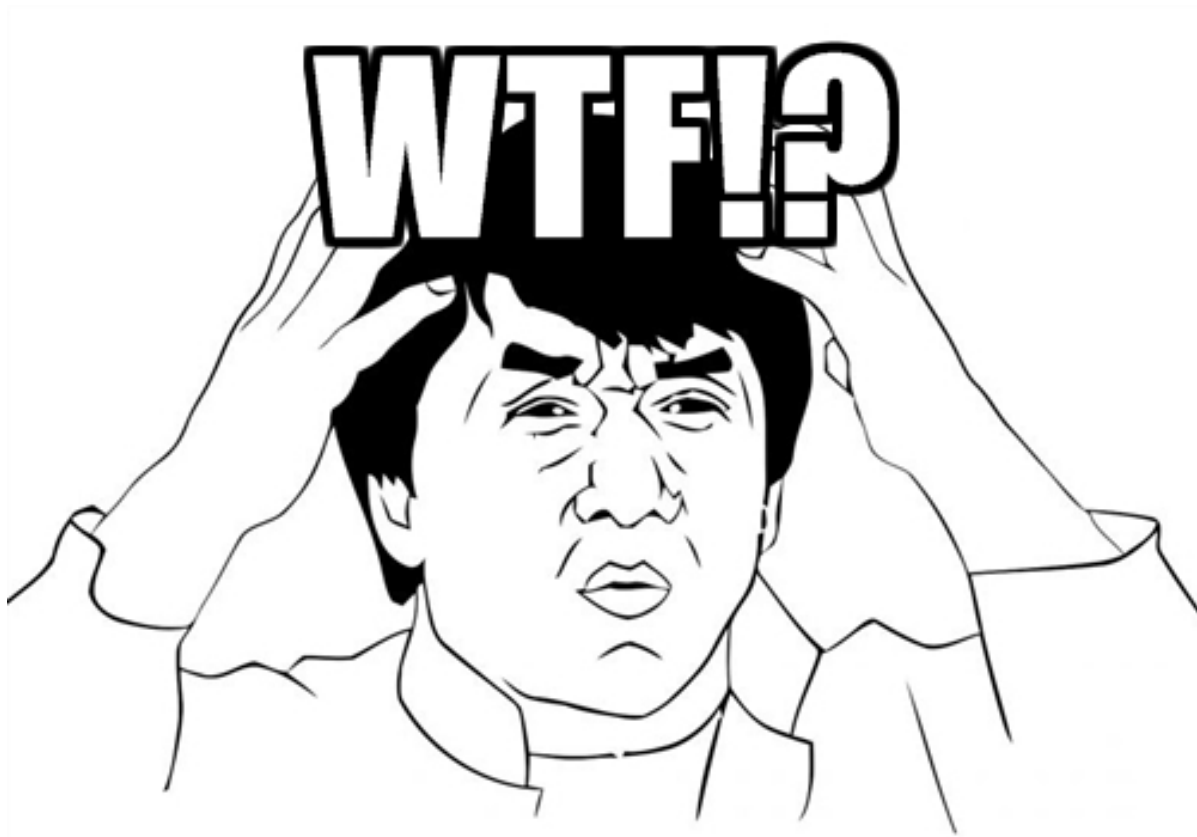
Deploy



Docker image size

```
nick11018@dc-1b15080085: /$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
kevingo/http-onbuild latest             647d6be37263       17 hours ago       682.5 MB
```

吃頻寬、吃 storage ...



Alpine

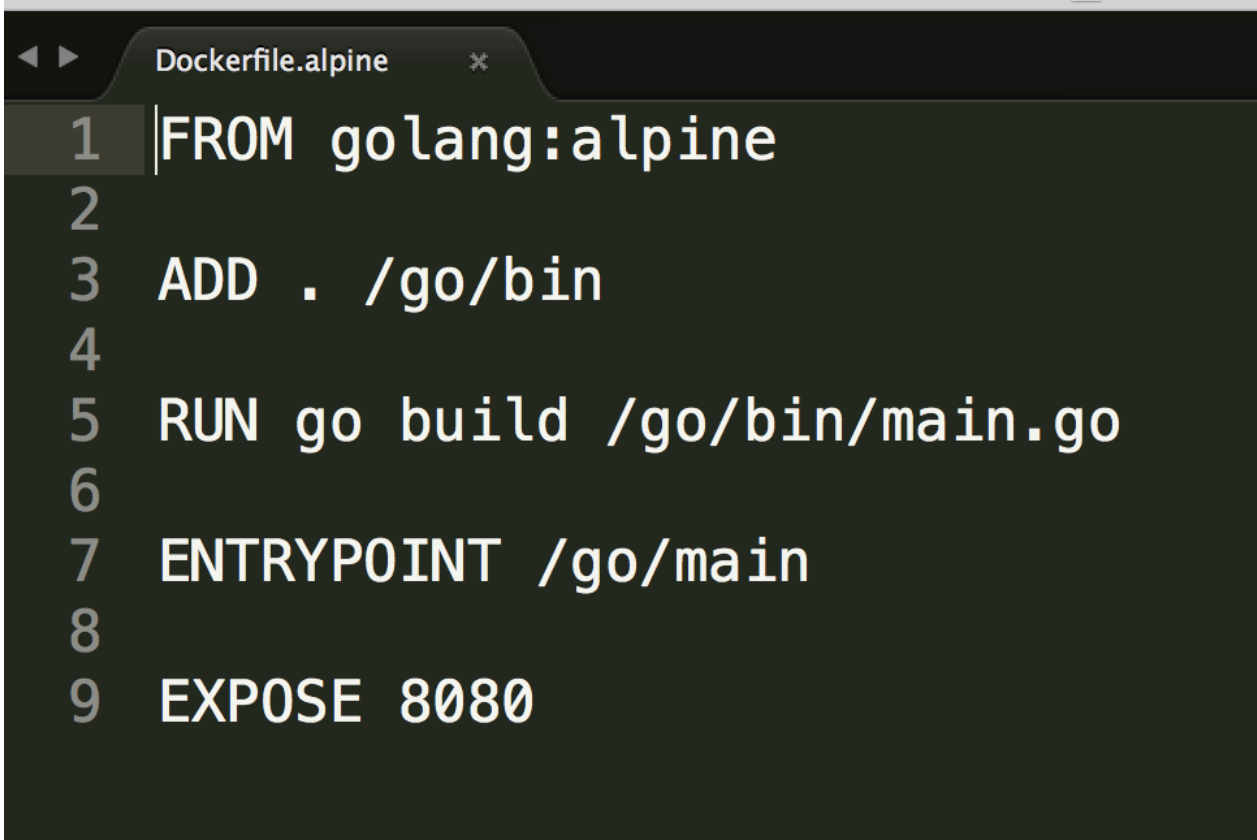
golang:alpine

This image is based on the popular [Alpine Linux project](#), available in [the alpine official image](#). Alpine Linux is much smaller than most distribution base images (~5MB), and thus leads to much slimmer images in general.

This variant is highly recommended when final image size being as small as possible is desired. The main caveat to note is that it does use [musl libc](#) instead of [glibc and friends](#), so certain software might run into issues depending on the depth of their libc requirements. However, most software doesn't have an issue with this, so this variant is usually a very safe choice. See [this Hacker News comment thread](#) for more discussion of the issues that might arise and some pro/con comparisons of using Alpine-based images.

To minimize image size, it's uncommon for additional related tools (such as `git` or `bash`) to be included in Alpine-based images. Using this image as a base, add the things you need in your own Dockerfile (see the [alpine image description](#) for examples of how to install packages if you are unfamiliar).

Using golang alpine image



```
1 FROM golang:alpine
2
3 ADD . /go/bin
4
5 RUN go build /go/bin/main.go
6
7 ENTRYPOINT /go/main
8
9 EXPOSE 8080
```

The image shows a code editor window titled "Dockerfile.alpine". The editor contains a Dockerfile with the following instructions: 1. FROM golang:alpine, 2. (empty line), 3. ADD . /go/bin, 4. (empty line), 5. RUN go build /go/bin/main.go, 6. (empty line), 7. ENTRYPOINT /go/main, 8. (empty line), 9. EXPOSE 8080. The first line is highlighted with a mouse cursor.

Build

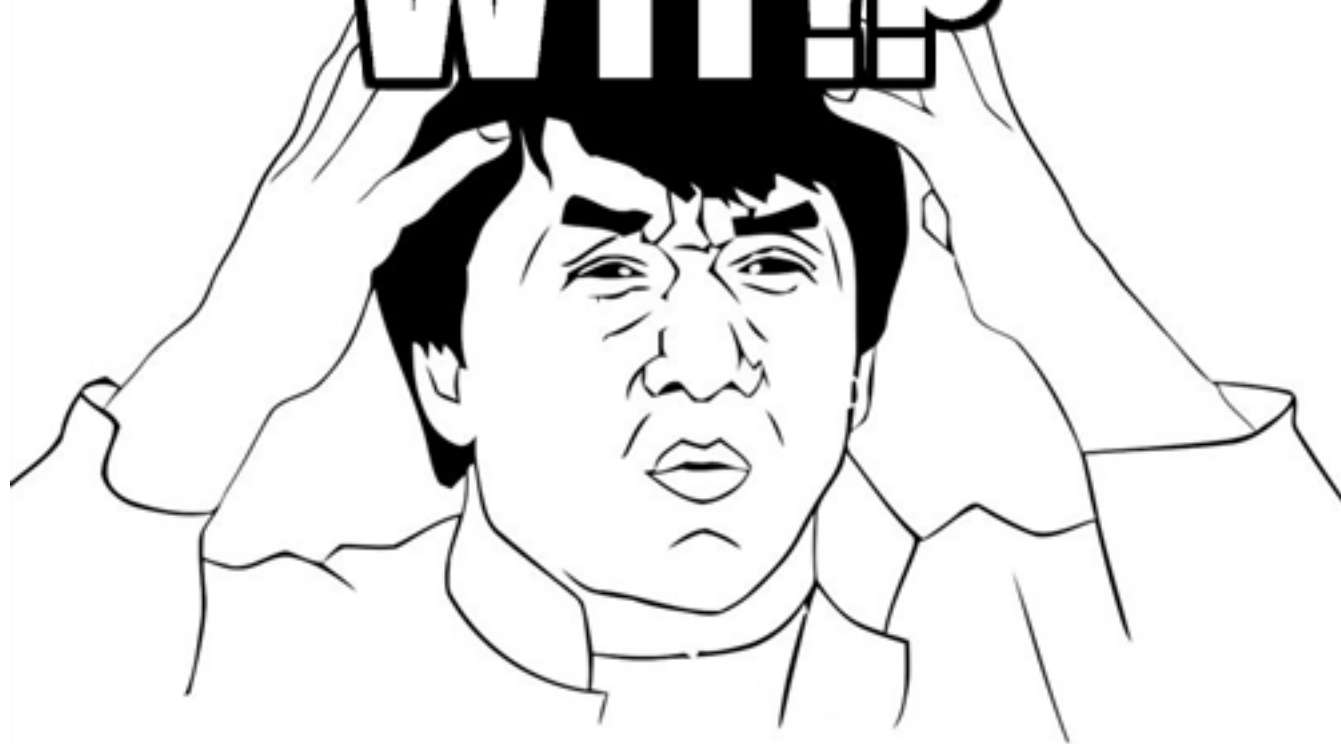
```
$ docker build -t kevingo/http-alpine -f Dockerfile.alpine .
Sending build context to Docker daemon 10.19 MB
Step 1 : FROM golang:alpine
---> 4e874ba52406
Step 2 : ADD . /go/bin
---> 8fa9bf59fc59
Removing intermediate container f2b847ac3f11
Step 3 : RUN go build /go/bin/main.go
---> Running in 9b15e7c54e42
---> 073fdd94c2cc
Removing intermediate container 9b15e7c54e42
Step 4 : ENTRYPOINT /go/main
---> Running in b99a04d3824b
---> aebac9c7a4e4
Removing intermediate container b99a04d3824b
Step 5 : EXPOSE 8080
---> Running in cc0d6647e4c6
---> 5660feb65fde
Removing intermediate container cc0d6647e4c6
Successfully built 5660feb65fde
```

Docker image size

```
mck11018@NB15080089: ~/Documents/go_workspace/src/github.com/kevingo/go-small-docker-images
$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
http.alpine	latest	f4d0de079a38	7 seconds ago	251.2 MB
kevingo/http-onbuild	latest	647d6be37263	18 hours ago	682.5 MB

WTF!?



HOME ABOUT ARCHIVES SEARCH



Joe Shaw

software engineer

Smaller Docker containers for Go apps

<https://joeshaw.org/smaller-docker-containers-for-go-apps/>

Using two containers

Builder Container

Runner Container

大

小

Builder Dockerfile

Build binary	<code>FROM golang:onbuild</code>
Copy runner Dockerfile	<code>COPY Dockerfile.runner /go/bin/Dockerfile</code>
	<code>WORKDIR /go/bin</code>
tar output stream	<code>CMD tar -cf - .</code>

Runner Dockerfile

```
FROM flynn/busybox
```

```
COPY app /bin/app
```

```
EXPOSE 8001
```

```
CMD ["/bin/app"]
```

Just one line

```
$ docker build -t builder -f Dockerfile.builder . && docker run builder | docker build -t runner -
```

Build builder image

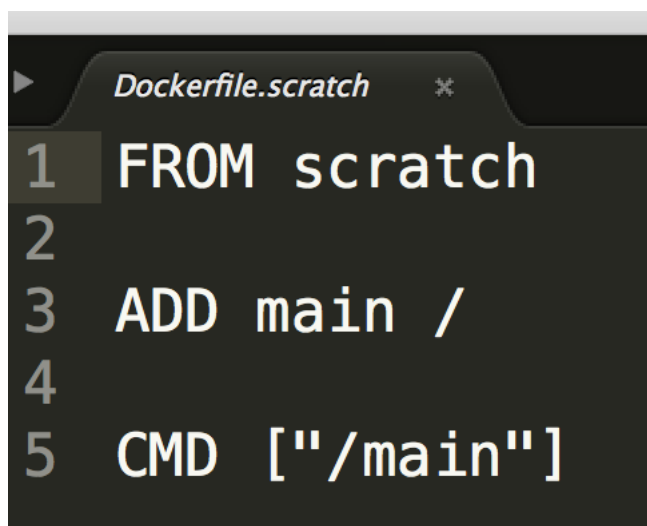
Build runner image
(using builder image output)

Docker image size

kevingo/http-runner	latest	1aa183397374	24 hours ago	10.19 MB
kevingo/http-builder	latest	4be323614fea	24 hours ago	688.2 MB
kevingo/http-alpine	latest	5660feb65fde	2 days ago	256.9 MB
kevingo/http-onbuild	latest	647d6be37263	4 days ago	682.5 MB

同場加映 - Static Build + Scratch Docker image

```
nick12010@dc-1b23666665: /Documents/go_not_kspace/src/golang.com/revengo/go_smart$ CGO_ENABLED=0 GOOS=linux GOARCH=amd64 go build -a -installsuffix cgo -o main .
```



```
Dockerfile.scratch
1 FROM scratch
2
3 ADD main /
4
5 CMD ["/main"]
```


Build

```
$ docker build -t kevingo/http-scratch -f Dockerfile.scratch .  
Sending build context to Docker daemon 10.22 MB  
Step 1 : FROM scratch  
--->  
Step 2 : ADD main /  
---> affe51644a3a  
Removing intermediate container 01107054f9c5  
Step 3 : CMD /main  
---> Running in 8778b034a991  
---> 2a0b30982695  
Removing intermediate container 8778b034a991  
Successfully built 2a0b30982695
```

Image size

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
kevingo/http-scratch	latest	98c0ca9613ca	9 hours ago	5.647 MB
kevingo/http-runner	latest	1aa183397374	17 hours ago	10.19 MB
kevingo/http-builder	latest	4be323614fea	17 hours ago	688.2 MB
kevingo/http-alpine	latest	5660feb65fde	41 hours ago	256.9 MB
kevingo/http-onbuild	latest	647d6be37263	3 days ago	682.5 MB

使用前請詳閱公開說明書

> Is there any disadvantages of using the pure Go name resolution library now? I'm interested in limiting the number of threads created for a simple solution.

Some, depending on your operating system. On OSX for example, crypto/tls will not be able to find any root certificates as the integration with the OS X keychain requires cgo. Also, using the pure DNS resolver may cause lookups for 'localhost' to trigger a firewall connection warning if your corporate IT department has set your firewall to paranoid mode. YMMV.

> I tested a server application with 1000 concurrent connections, for each connection the server will call net.Dial and then pass data over the connection.

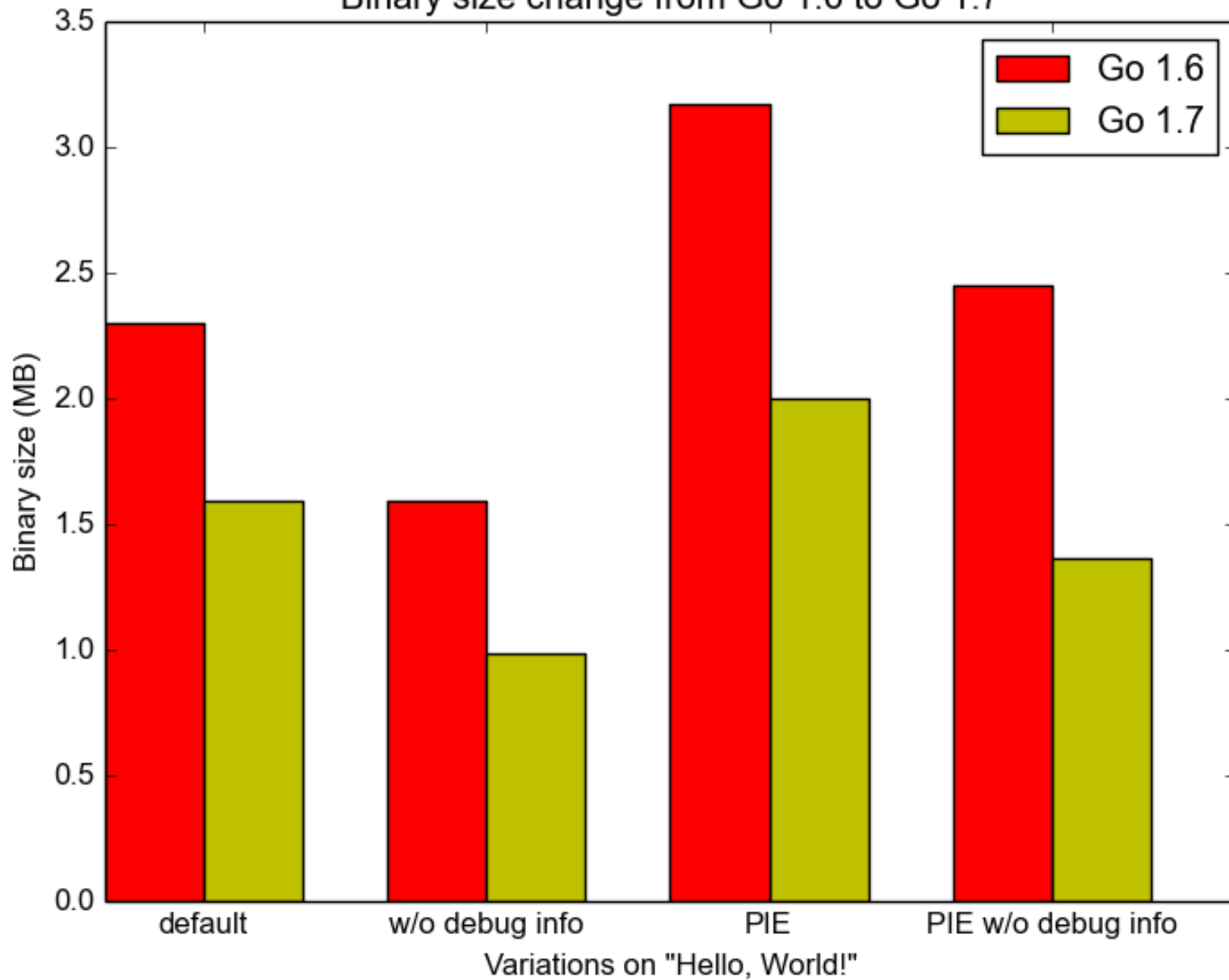
> Using cgo would spawn up to 64 threads in my test, while disabling cgo would spawn up to only 8 threads.

This request comes up quite often, and I believe it is due to a long running limitation in the libc resolver. You should probably log a feature request on the golang issue tracker.

Cheers

Dave

Binary size change from Go 1.6 to Go 1.7





<https://github.com/kevingo/Go-Small-Docker-Image>

Thanks!



@KEVINGO



KEVINGO TSAI