Go and Docker

Dev&Ops illustrated

Daniel CHAFFIOL

Softeam Cadextan

(1999)

Amundi ASSET MANAGEMENT	Amundi Asset Management
HSBC	HSBC
SOCIETE GENERALE	Société Générale
BNP PARIBAS	BNP-Paribas

VonC

Stack Overflow

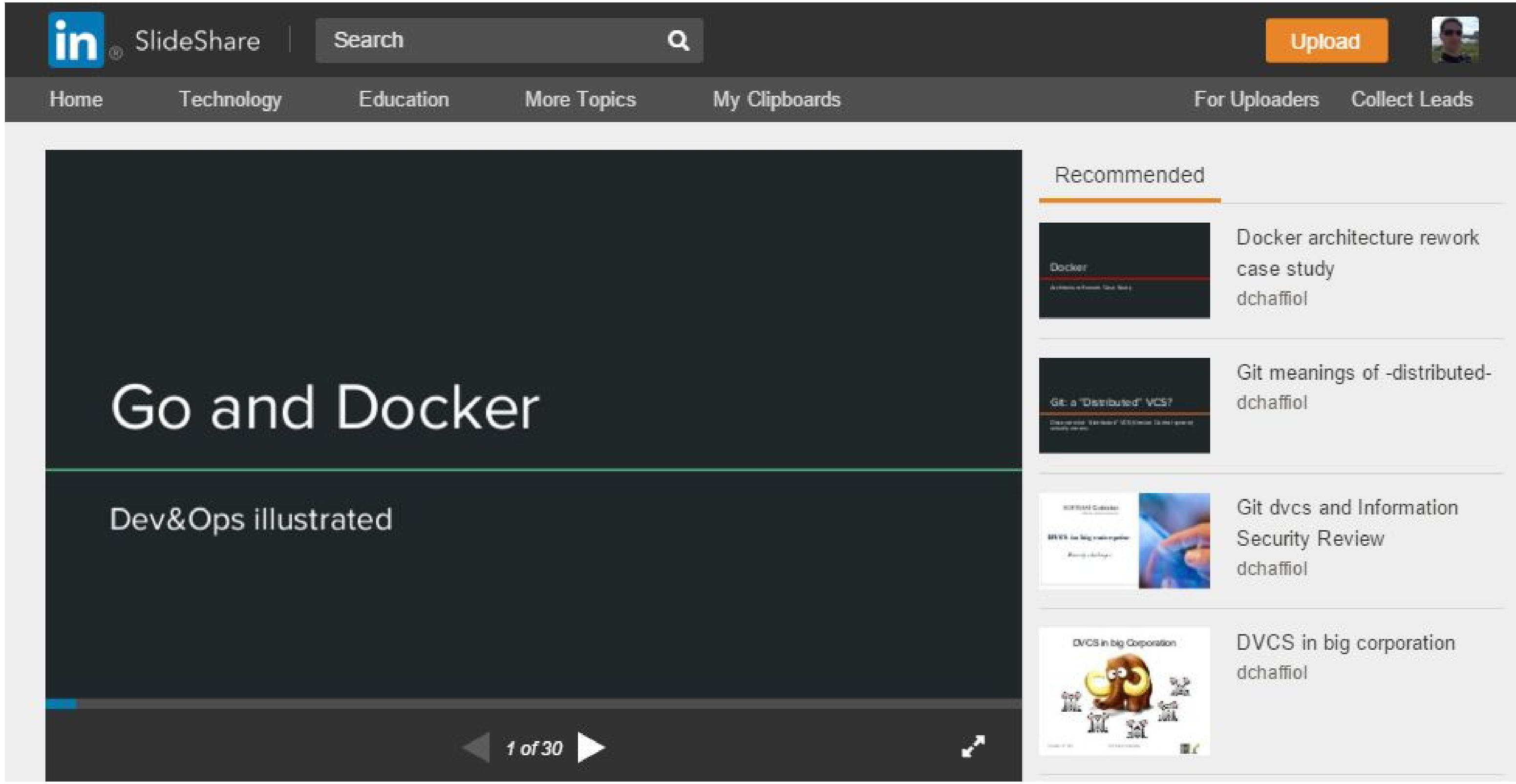
(2008)

Git - Hg – SVN - ClearCase

Golang - Docker

Eclipse - Jenkins

slideshare



Go and Docker

Dev&Ops illustrated

Recommended

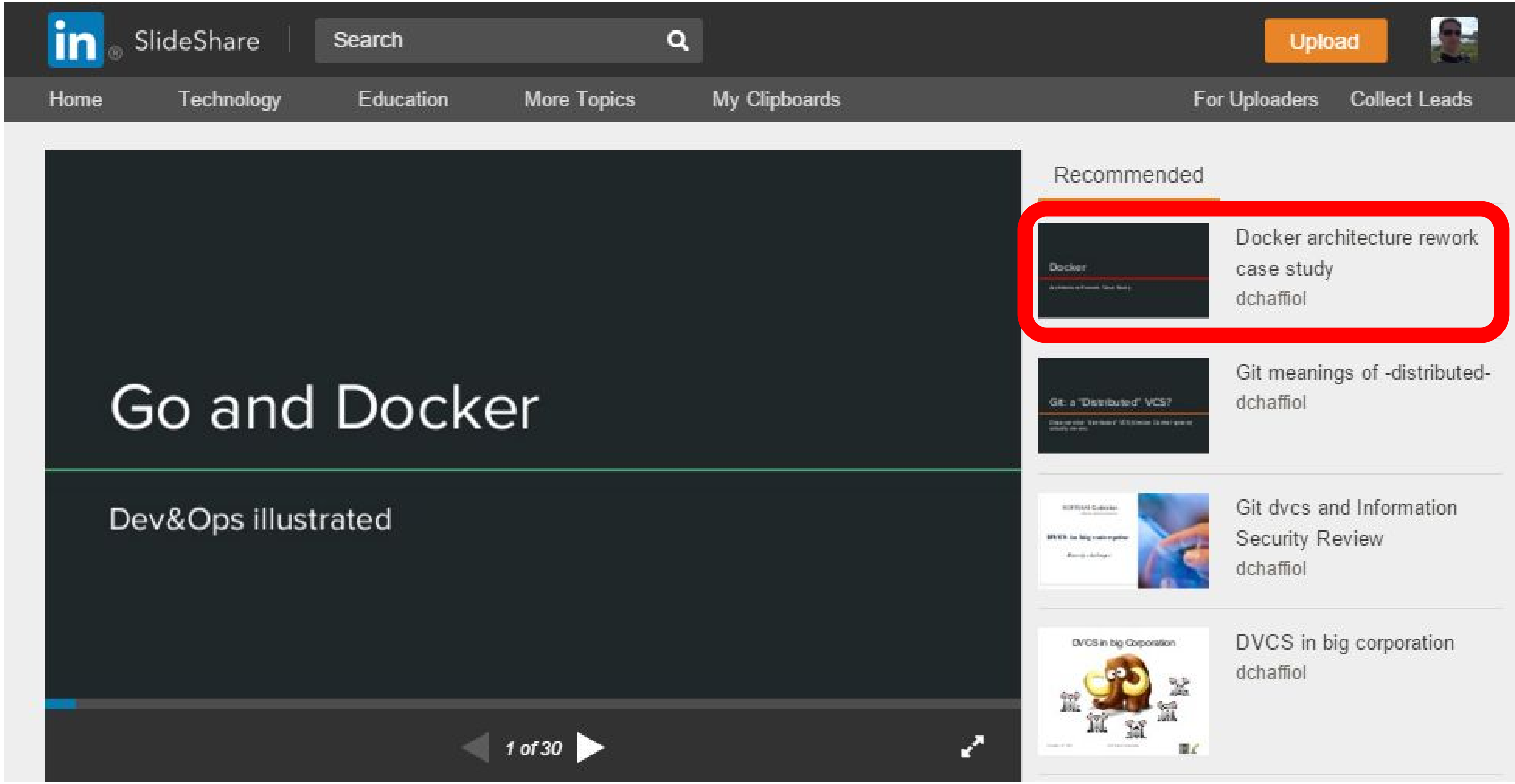
Docker architecture rework case study
dchaffiol

Git meanings of -distributed-
dchaffiol

Git dvcs and Information Security Review
dchaffiol

DVCS in big corporation
dchaffiol

slideshare



Go and Docker

Dev&Ops illustrated

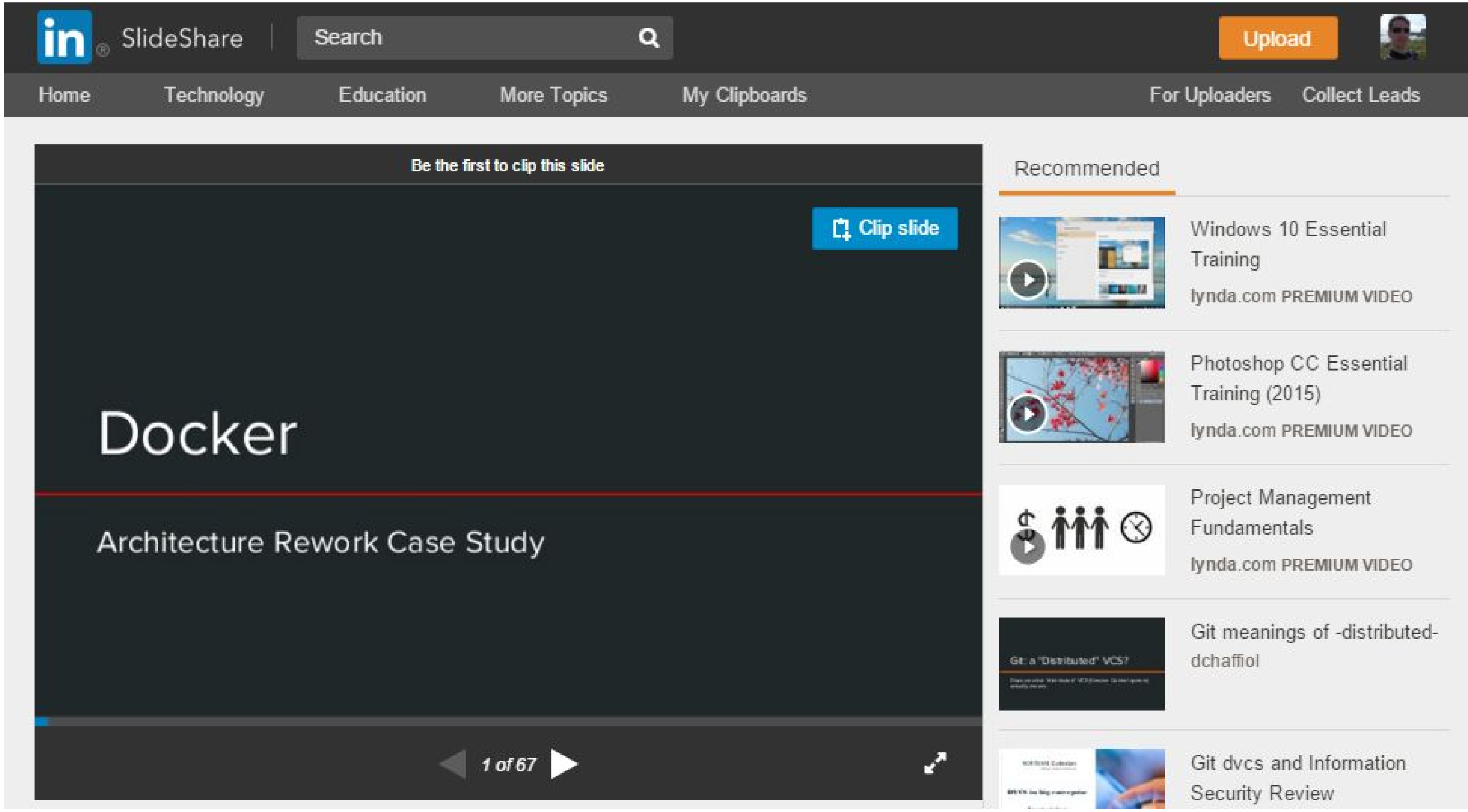
Docker architecture rework case study
dchaffiol

Git meanings of -distributed-
dchaffiol

Git dvcs and Information Security Review
dchaffiol

DVCS in big corporation
dchaffiol

slideshare



Docker

Architecture Rework Case Study

Windows 10 Essential Training
lynda.com PREMIUM VIDEO

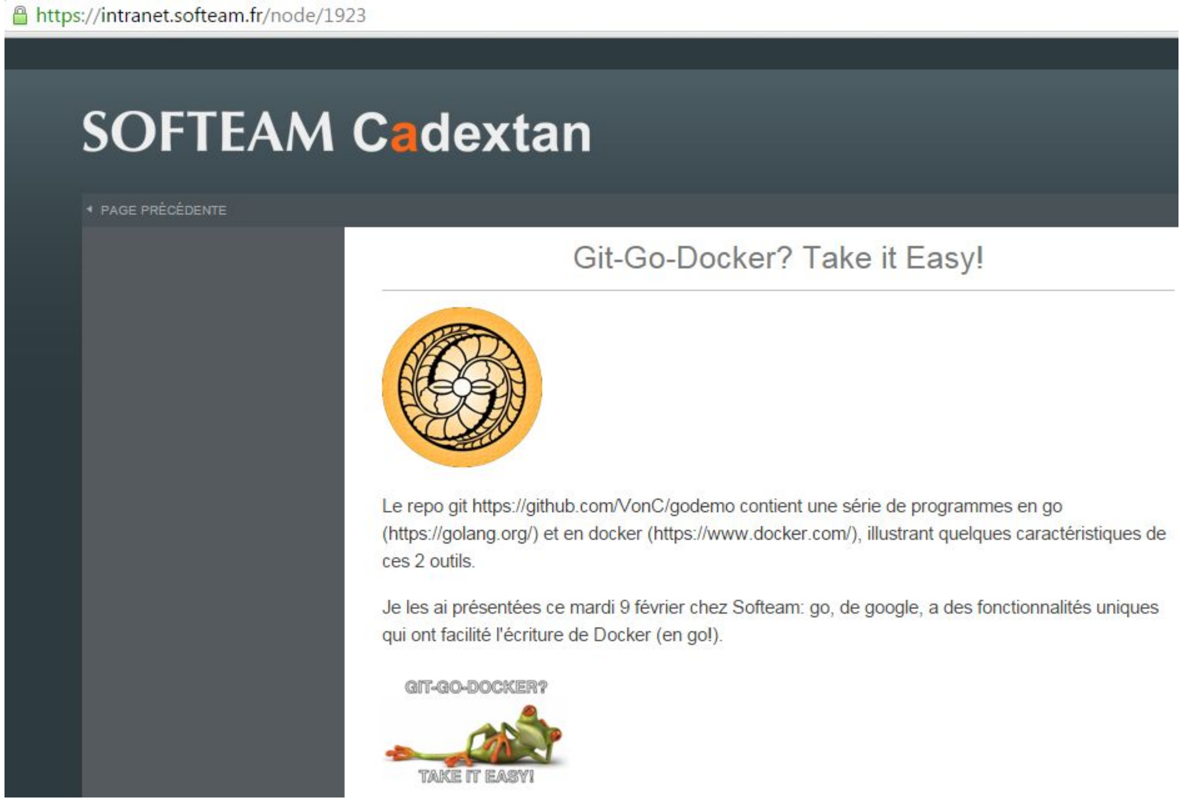
Photoshop CC Essential Training (2015)
lynda.com PREMIUM VIDEO

Project Management Fundamentals
lynda.com PREMIUM VIDEO

Git meanings of -distributed-
dchaffiol

Git dvcs and Information Security Review

Intranet



https://intranet.softeam.fr/node/1923

SOFTEAM Cadextan

PAGE PRÉCÉDENTE

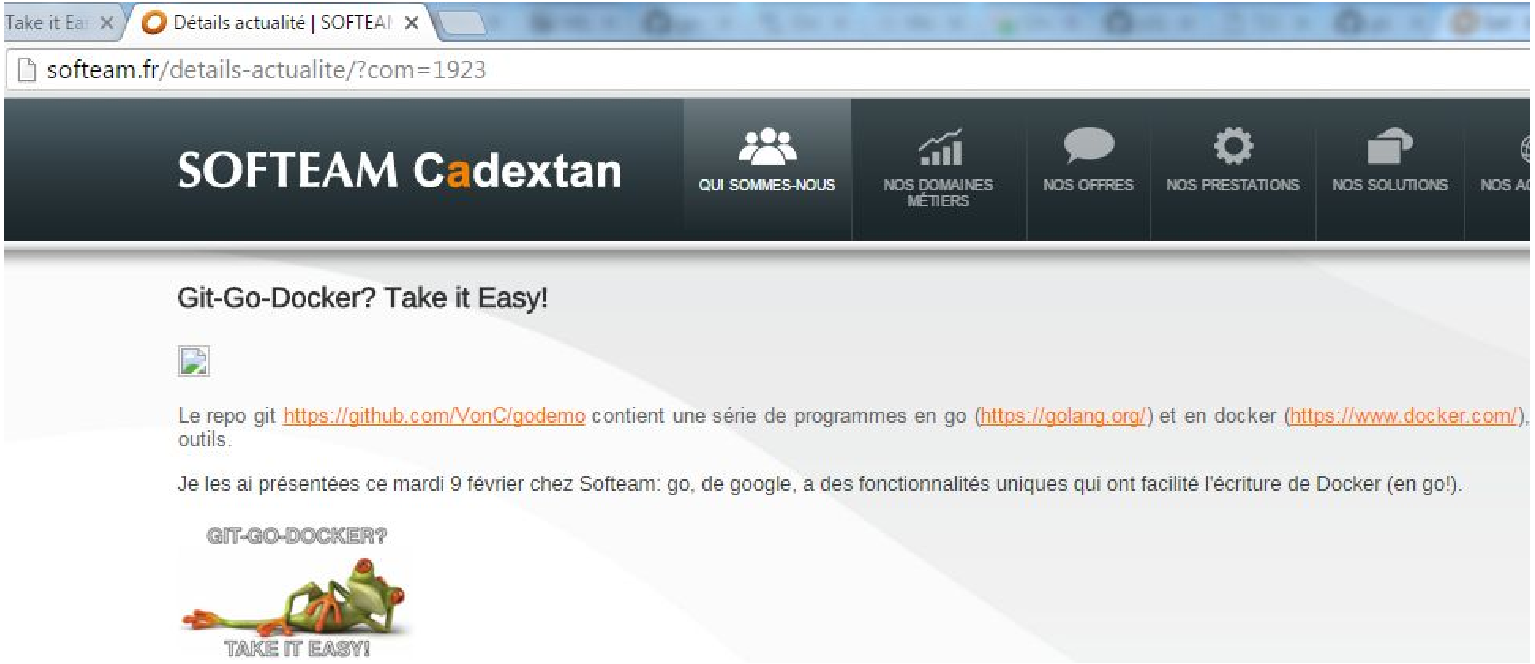
Git-Go-Docker? Take it Easy!

Le repo git https://github.com/VonC/godemo contient une série de programmes en go (https://golang.org/) et en docker (https://www.docker.com/), illustrant quelques caractéristiques de ces 2 outils.

Je les ai présentées ce mardi 9 février chez Softeam: go, de google, a des fonctionnalités uniques qui ont facilité l'écriture de Docker (en go!).

https://intranet.softeam.fr/node/1923

softeam.fr



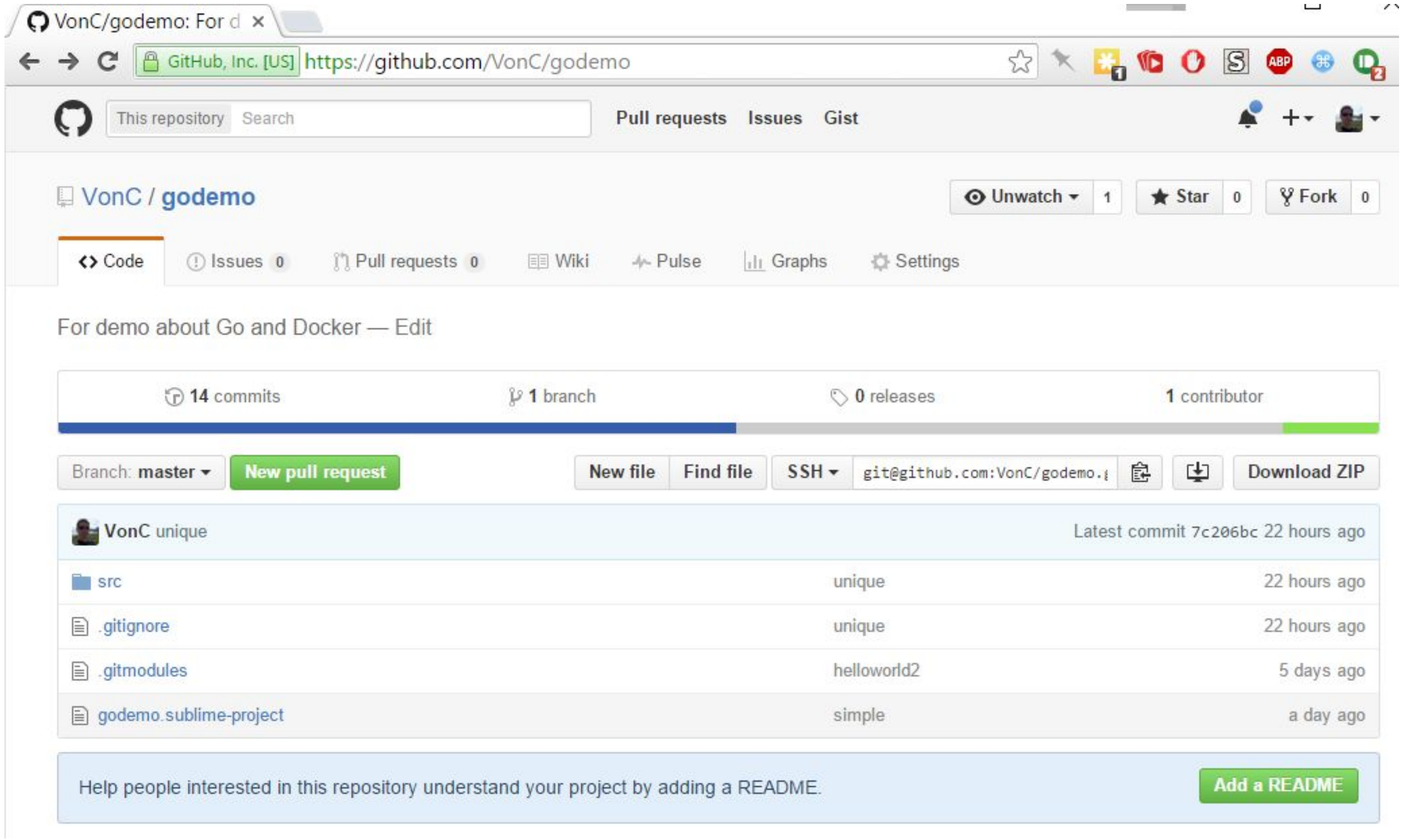
Git-Go-Docker? Take it Easy!

Le repo git https://github.com/VonC/godemo contient une série de programmes en go (https://golang.org/) et en docker (https://www.docker.com/), outils.

Je les ai présentées ce mardi 9 février chez Softeam: go, de google, a des fonctionnalités uniques qui ont facilité l'écriture de Docker (en go!).

http://softeam.fr/details-actualite/?com=1923

GitHub



VonC/godemo: For d
GitHub, Inc. [US] https://github.com/VonC/godemo
This repository Search
Pull requests Issues Gist
Unwatch 1 Star 0 Fork 0
Code Issues 0 Pull requests 0 Wiki Pulse Graphs Settings
Branch: master
New pull request
New file Find file SSH
git@github.com:VonC/godemo.g
Download ZIP
Add a README

VonC / godemo

For demo about Go and Docker — Edit

14 commits | 1 branch | 0 releases | 1 contributor

VonC unique — Latest commit 7c206bc 22 hours ago

src	unique	22 hours ago
.gitignore	unique	22 hours ago
.gitmodules	helloworld2	5 days ago
godemo.sublime-project	simple	a day ago

Help people interested in this repository understand your project by adding a README.

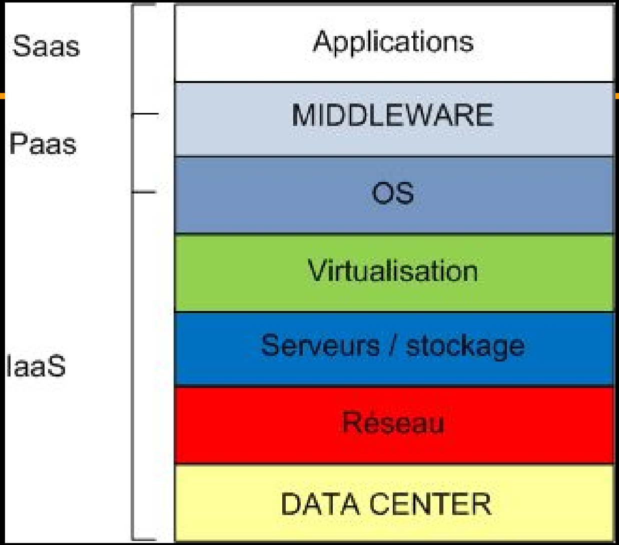
Go:

- Go 1.5-1.6
- Released in 2009 (1.0 since 2012)
- Rob Pike - Google

Docker:

- Docker 1.10
- Released in 2013 (1.0 since 6/2014)
- Solomon Hykes (from DotCloud)

Docker:



- Docker 1.10
- Released in 2013 (1.0 since 6/2014)
- Solomon Hykes (from DotCloud)

Go & Docker:

- Simple
- Unique
- DevOps

Simple

Less is more

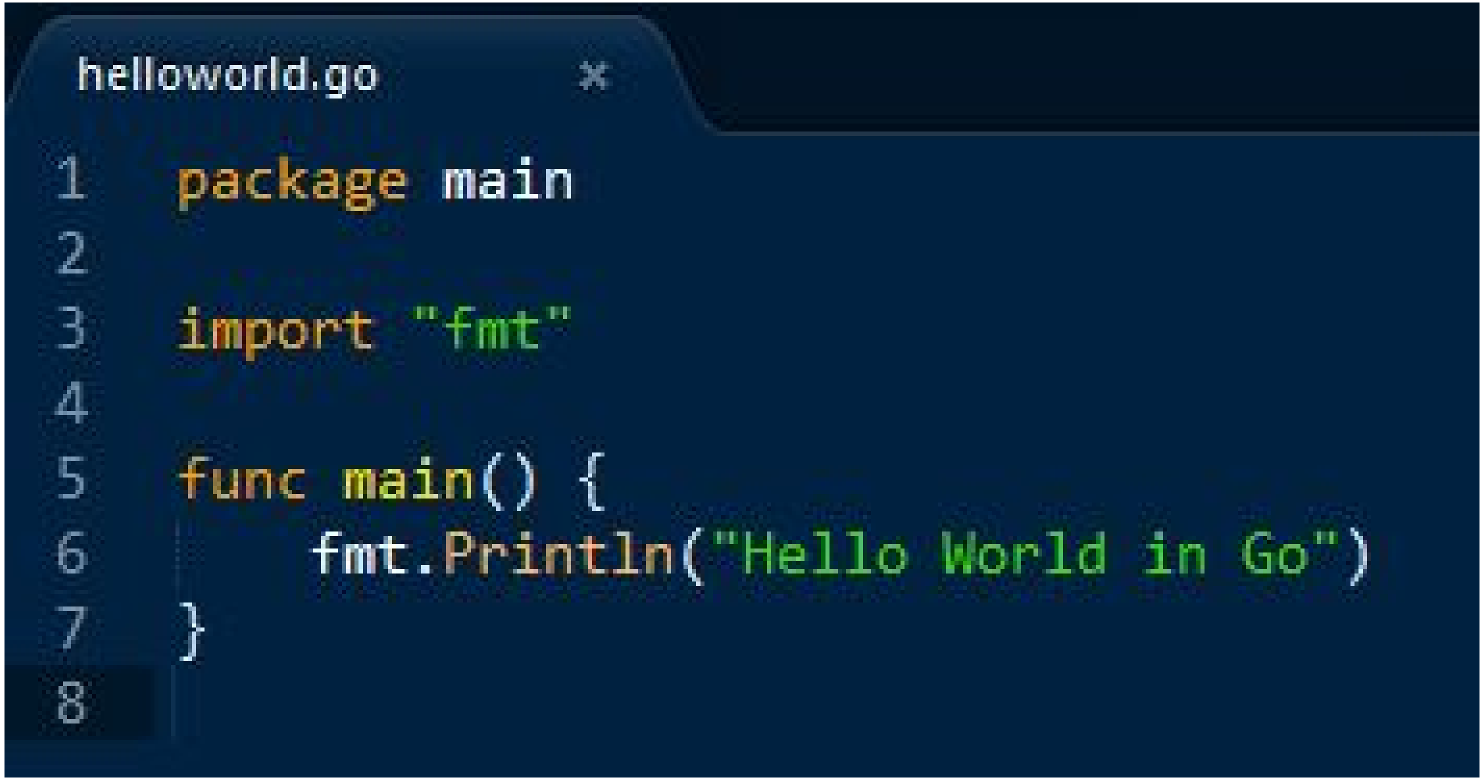
Simple

Less is more

Hello World in Go:

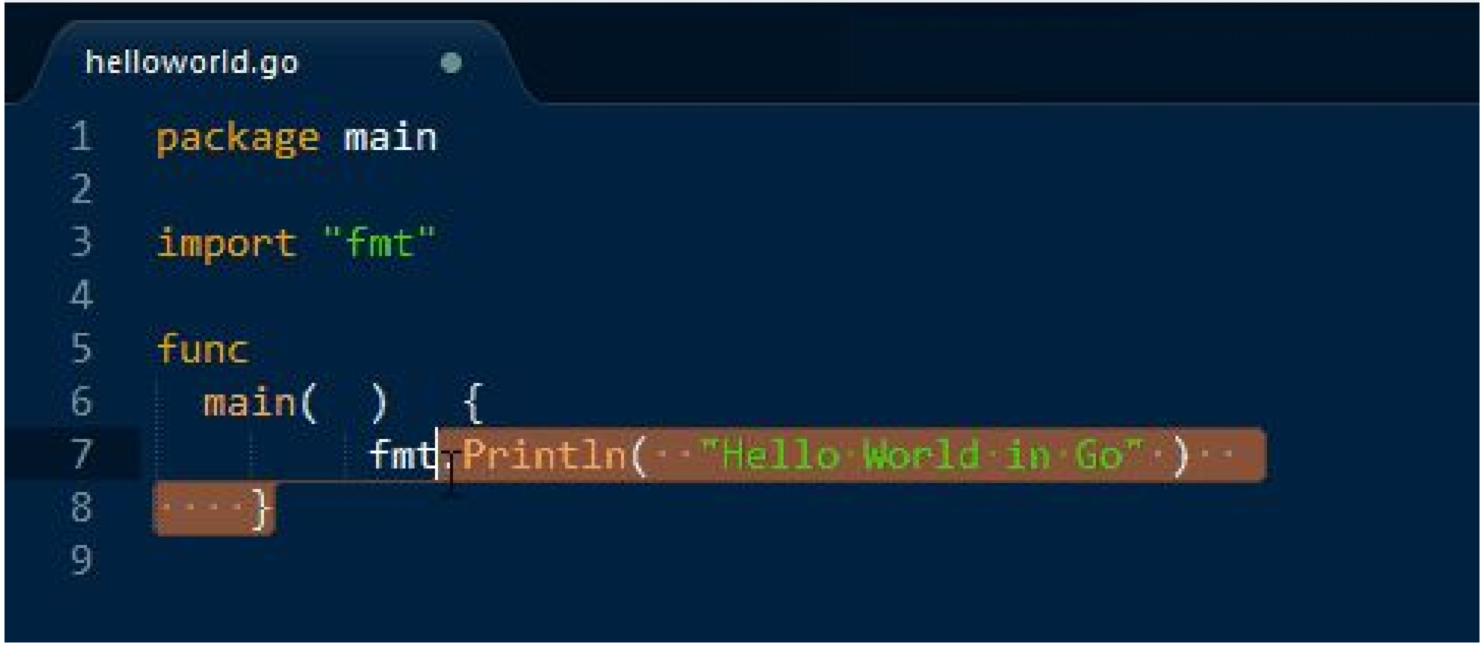
- gofmt
- godoc / go test
- go get

Gofmt



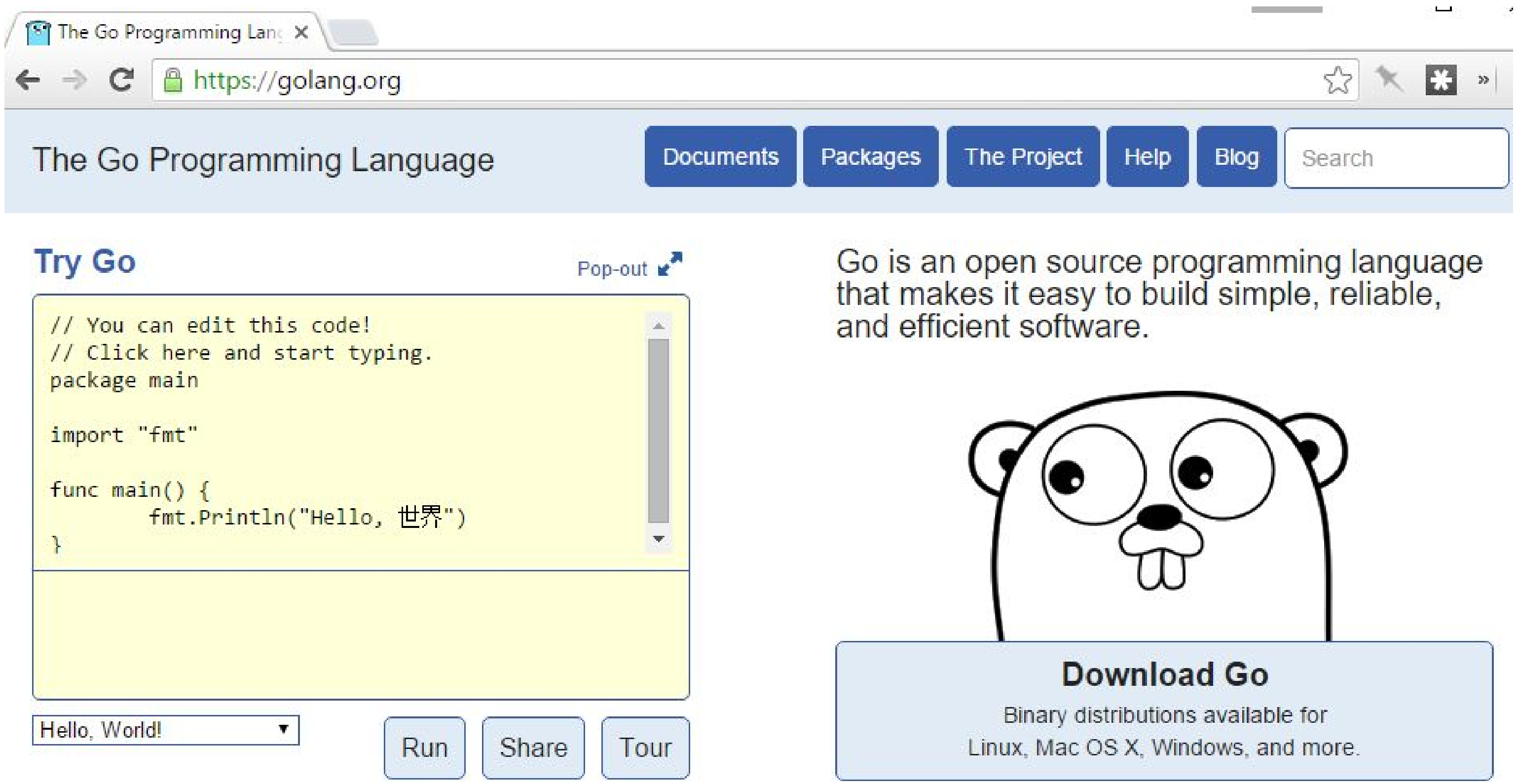
```go
package main

import "fmt"

func main() {
	fmt.Println("Hello World in Go")
}

```

Gofmt


helloworld.go

```
package main

import "fmt"

func
   main(  )   {
            fmt.Println(  "Hello World in Go" )  
    }

```

Godoc



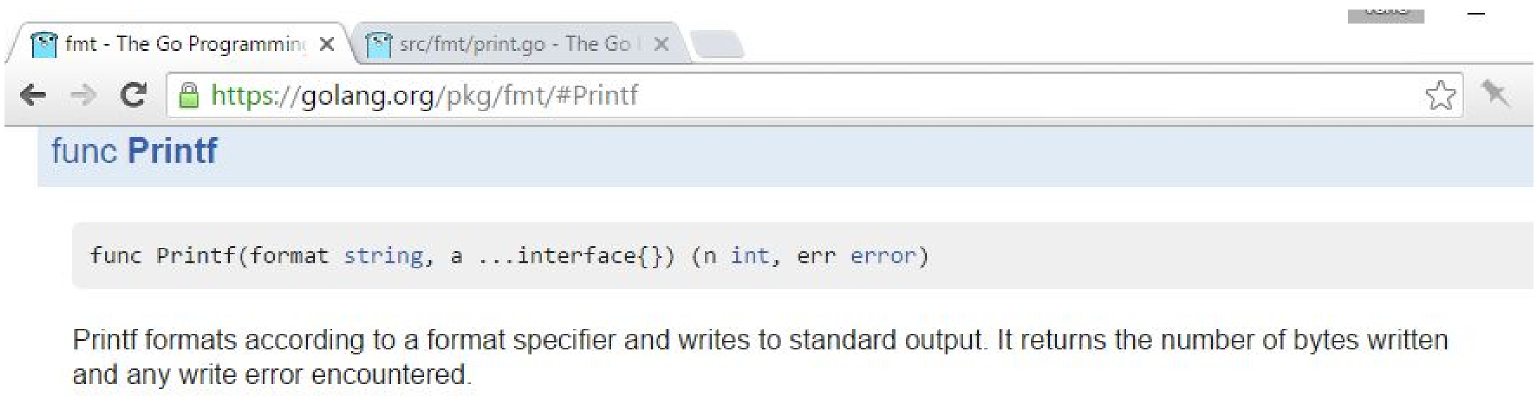
```
// You can edit this code!
// Click here and start typing.
package main

import "fmt"

func main() {
	fmt.Println("Hello, 世界")
}
```

Go is an open source programming language that makes it easy to build simple, reliable, and efficient software.

Download Go

Binary distributions available for Linux, Mac OS X, Windows, and more.

Godoc

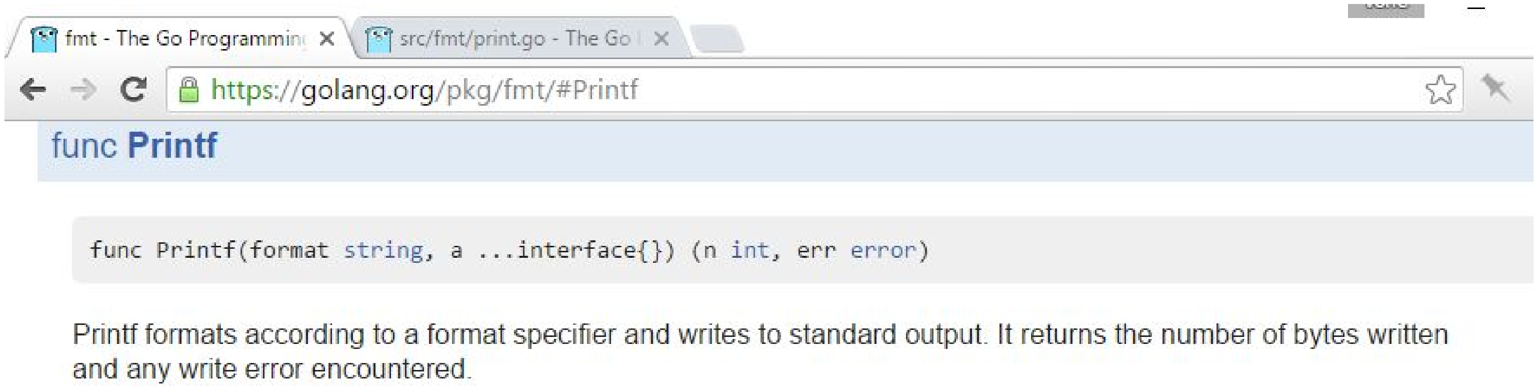


func **Printf**

```
func Printf(format string, a ...interface{}) (n int, err error)
```

Printf formats according to a format specifier and writes to standard output. It returns the number of bytes written and any write error encountered.

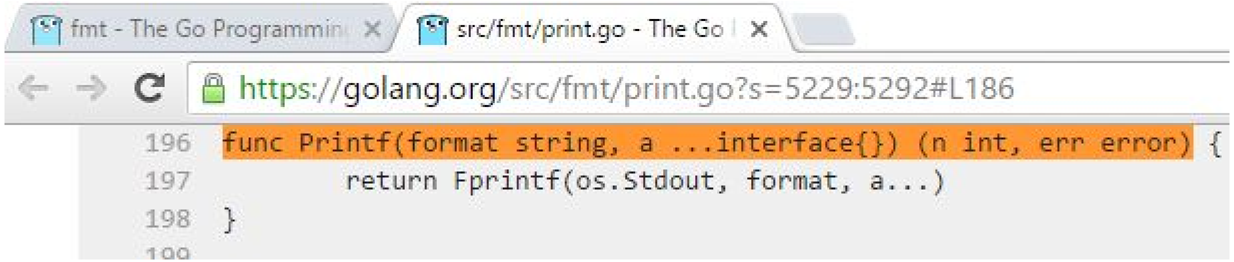
Godoc



func **Printf**

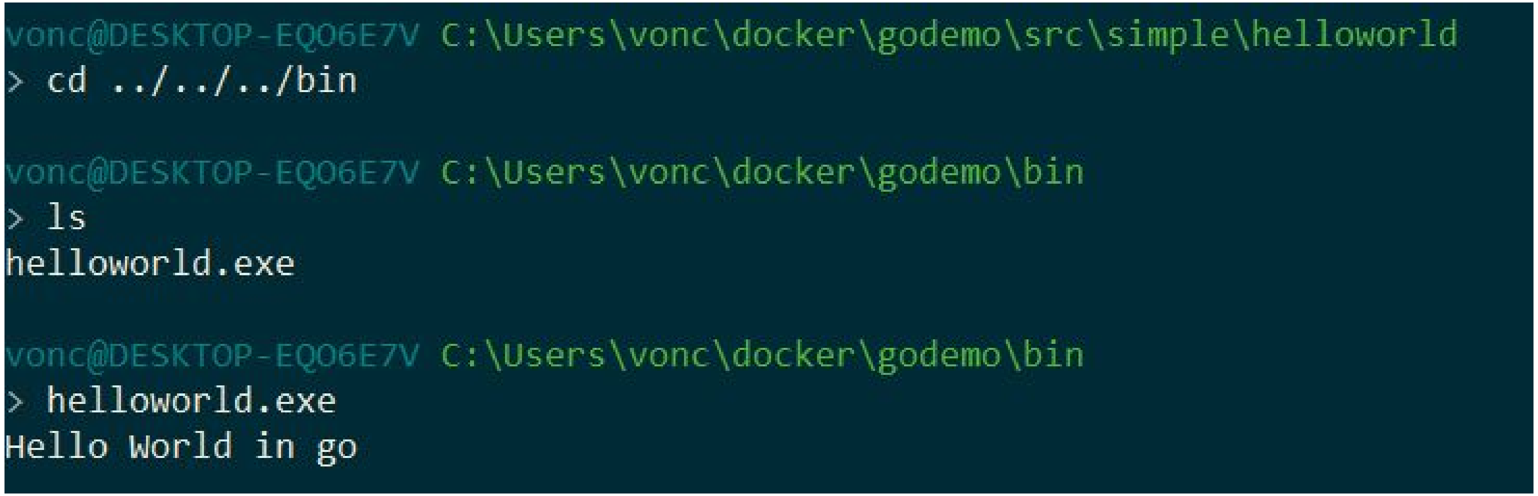
```
func Printf(format string, a ...interface{}) (n int, err error)
```

Printf formats according to a format specifier and writes to standard output. It returns the number of bytes written and any write error encountered.


fmt - The Go Programmin
src/fmt/print.go - The Go
https://golang.org/src/fmt/print.go?s=5229:5292#L186

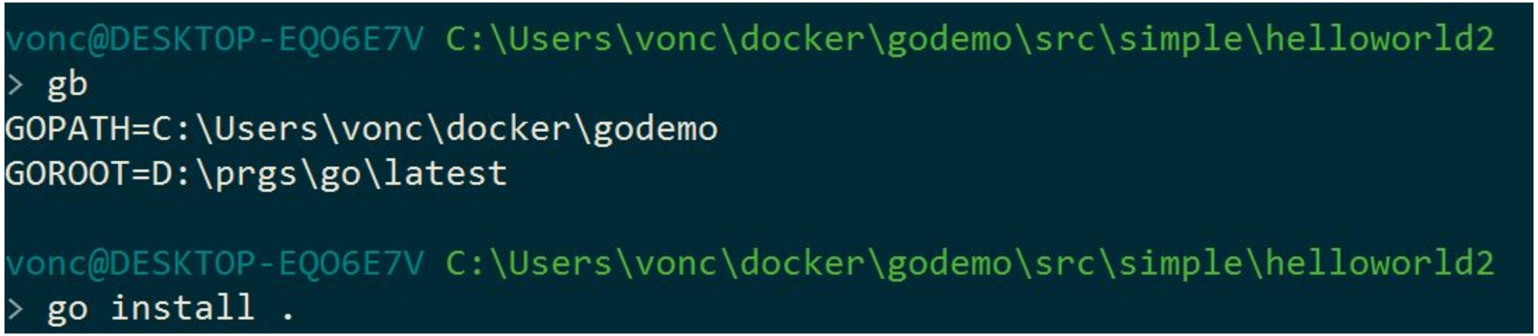
```
func Printf(format string, a ...interface{}) (n int, err error) {
        return Fprintf(os.Stdout, format, a...)
}
```

Goinstall



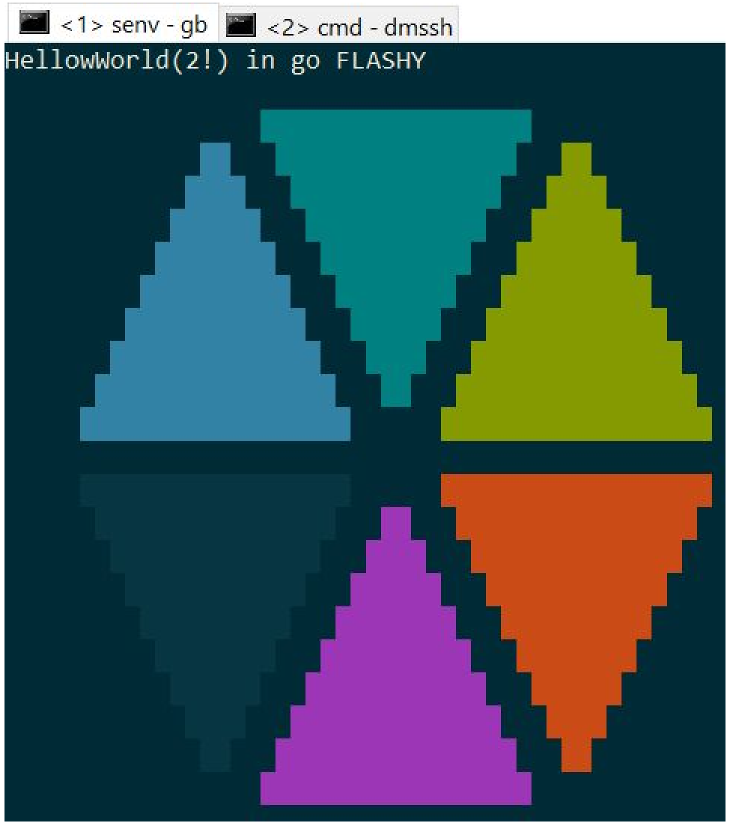
```
vonc@DESKTOP-EQO6E7V C:\Users\vonc\docker\godemo\src\simple\helloworld
> cd ../../../bin

vonc@DESKTOP-EQO6E7V C:\Users\vonc\docker\godemo\bin
> ls
helloworld.exe

vonc@DESKTOP-EQO6E7V C:\Users\vonc\docker\godemo\bin
> helloworld.exe
Hello World in go
```

go get



```
vonc@DESKTOP-EQ06E7V C:\Users\vonc\docker\godemo\src\simple\helloworld2
> gb
GOPATH=C:\Users\vonc\docker\godemo
GOROOT=D:\prgs\go\latest

vonc@DESKTOP-EQ06E7V C:\Users\vonc\docker\godemo\src\simple\helloworld2
> go install .
```

go get



```
HellowWorld(2!) in go FLASHY
```

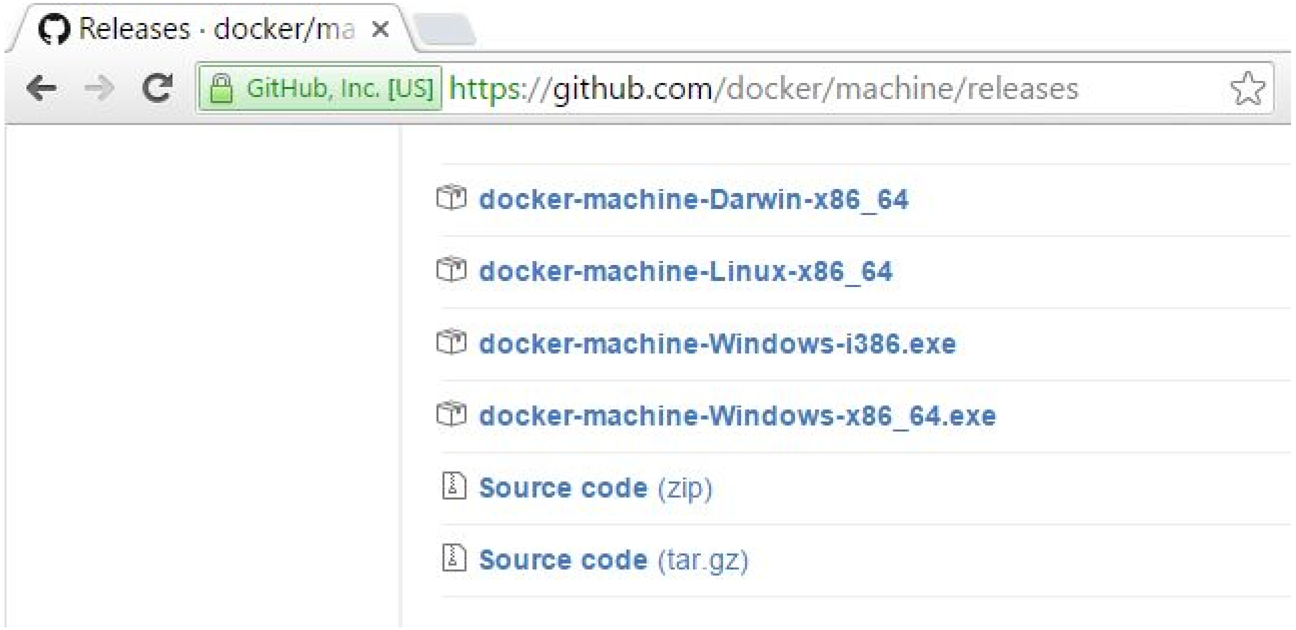
Simple

Less is more

Hello World in Docker:

- docker-machine
- SCRATCH
- docker run

docker-machine: simple



Releases · docker/ma

GitHub, Inc. [US] https://github.com/docker/machine/releases

docker-machine-Darwin-x86_64

docker-machine-Linux-x86_64

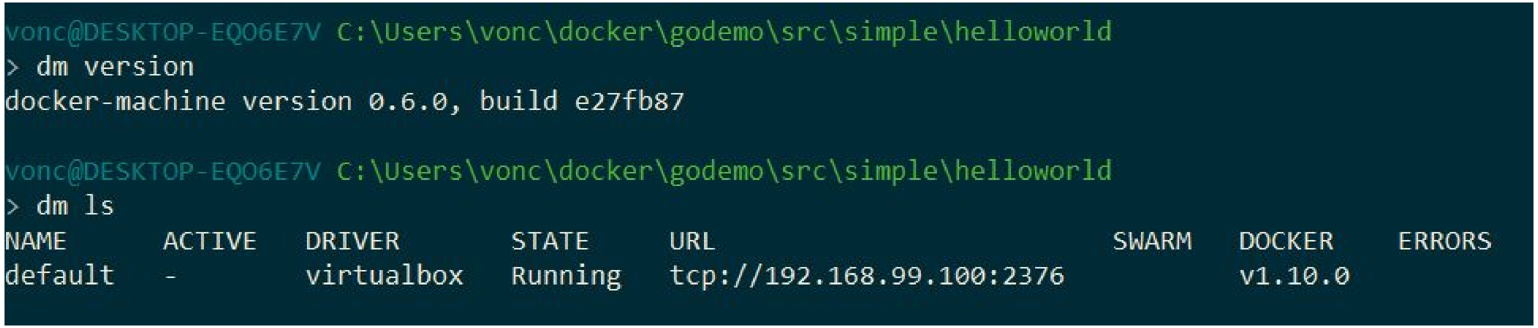
docker-machine-Windows-i386.exe

docker-machine-Windows-x86_64.exe

Source code (zip)

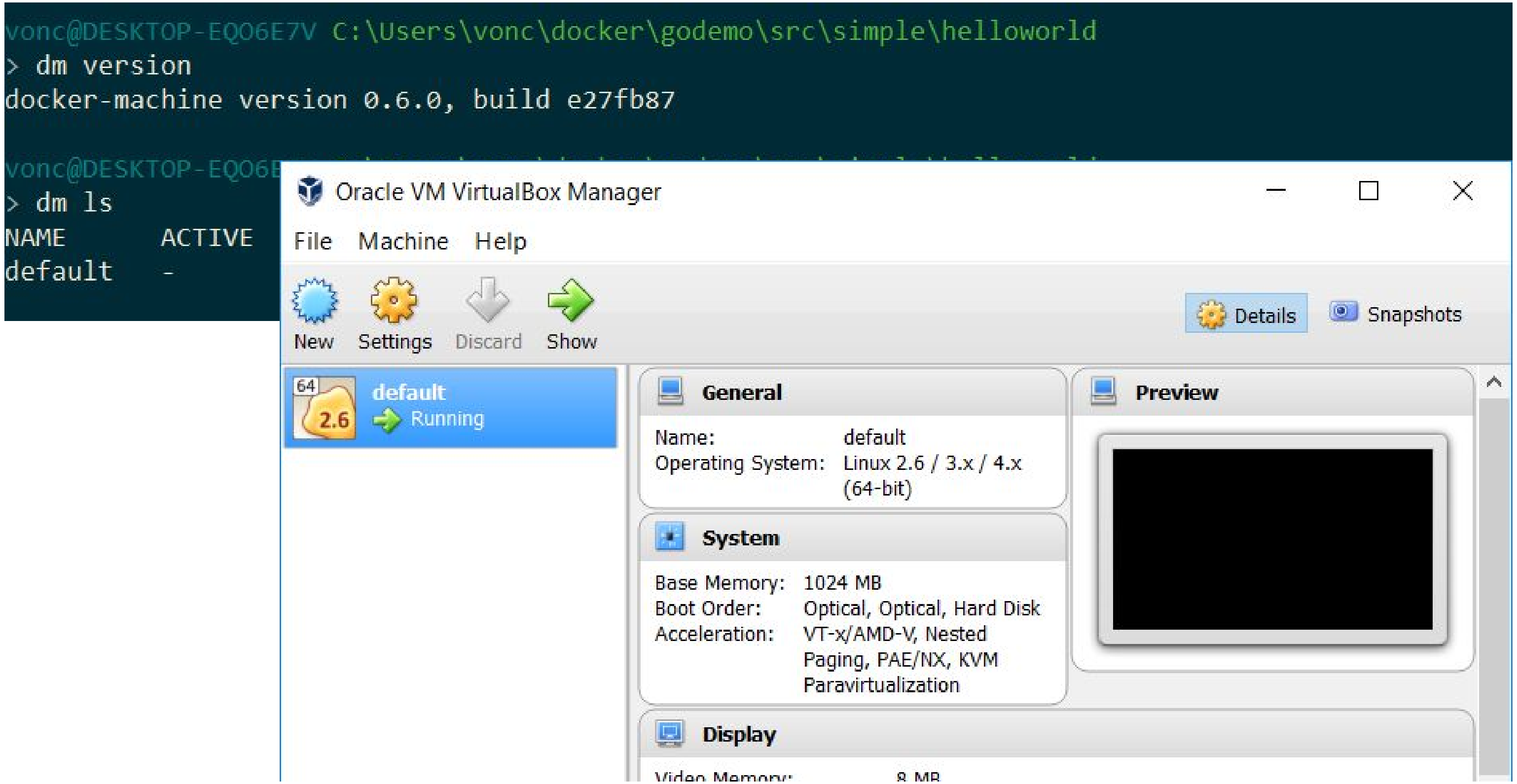
Source code (tar.gz)

docker-machine: simple



```
vonc@DESKTOP-EQO6E7V C:\Users\vonc\docker\godemo\src\simple\helloworld
> dm version
docker-machine version 0.6.0, build e27fb87

vonc@DESKTOP-EQO6E7V C:\Users\vonc\docker\godemo\src\simple\helloworld
> dm ls
NAME      ACTIVE   DRIVER       STATE     URL                         SWARM   DOCKER    ERRORS
default   -        virtualbox   Running   tcp://192.168.99.100:2376           v1.10.0
```

docker-machine: simple



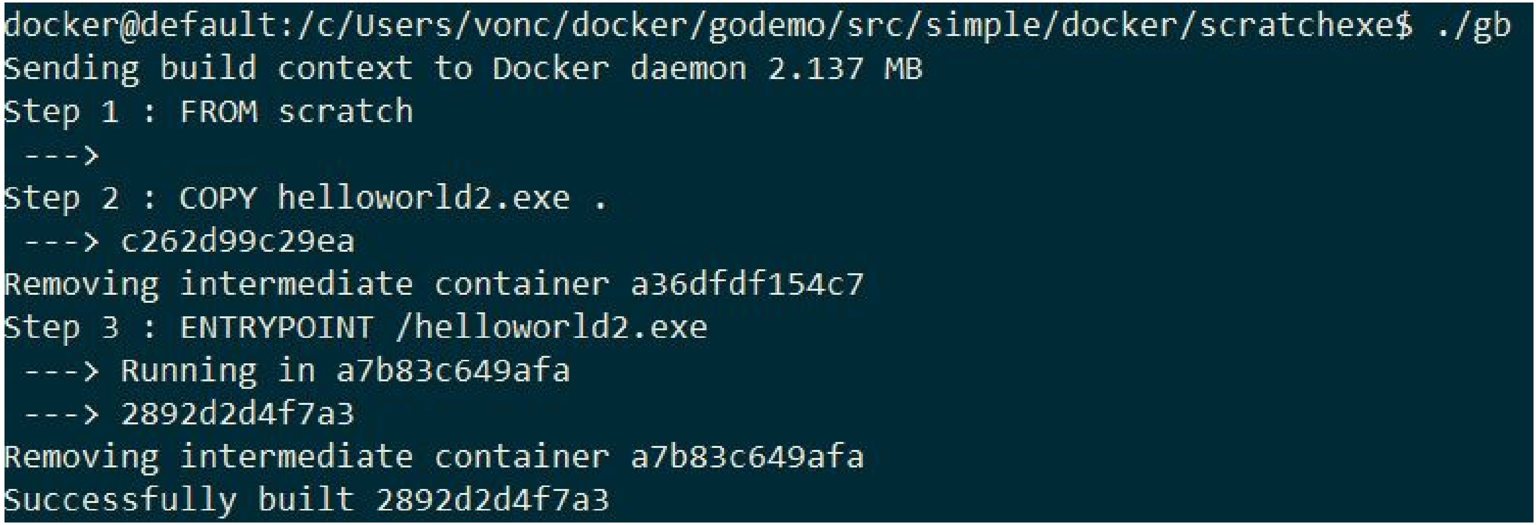
```
vonc@DESKTOP-EQO6E7V C:\Users\vonc\docker\godemo\src\simple\helloworld
> dm version
docker-machine version 0.6.0, build e27fb87

vonc@DESKTOP-EQO6E
> dm ls
NAME         ACTIVE
default      -
```

Name: default
Operating System: Linux 2.6 / 3.x / 4.x (64-bit)

Base Memory: 1024 MB
Boot Order: Optical, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, PAE/NX, KVM Paravirtualization

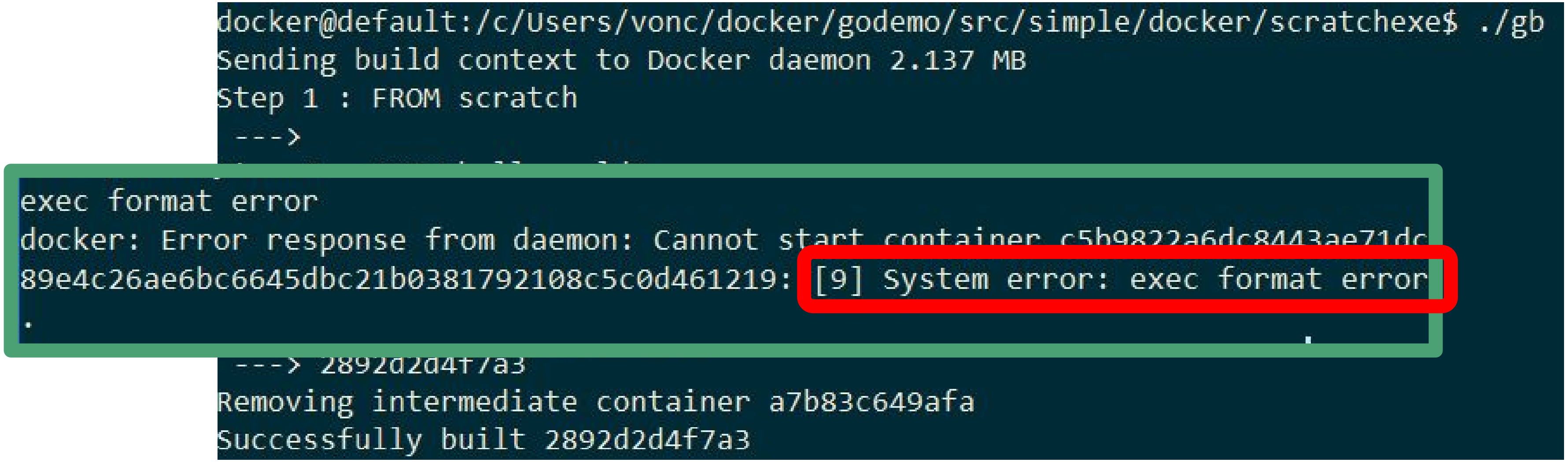
Video Memory: 8 MB

SCRATCH: first try



```
docker@default:/c/Users/vonc/docker/godemo/src/simple/docker/scratchexe$ ./gb
Sending build context to Docker daemon 2.137 MB
Step 1 : FROM scratch
 --->
Step 2 : COPY helloworld2.exe .
 ---> c262d99c29ea
Removing intermediate container a36dfdf154c7
Step 3 : ENTRYPOINT /helloworld2.exe
 ---> Running in a7b83c649afa
 ---> 2892d2d4f7a3
Removing intermediate container a7b83c649afa
Successfully built 2892d2d4f7a3
```

SCRATCH: first try

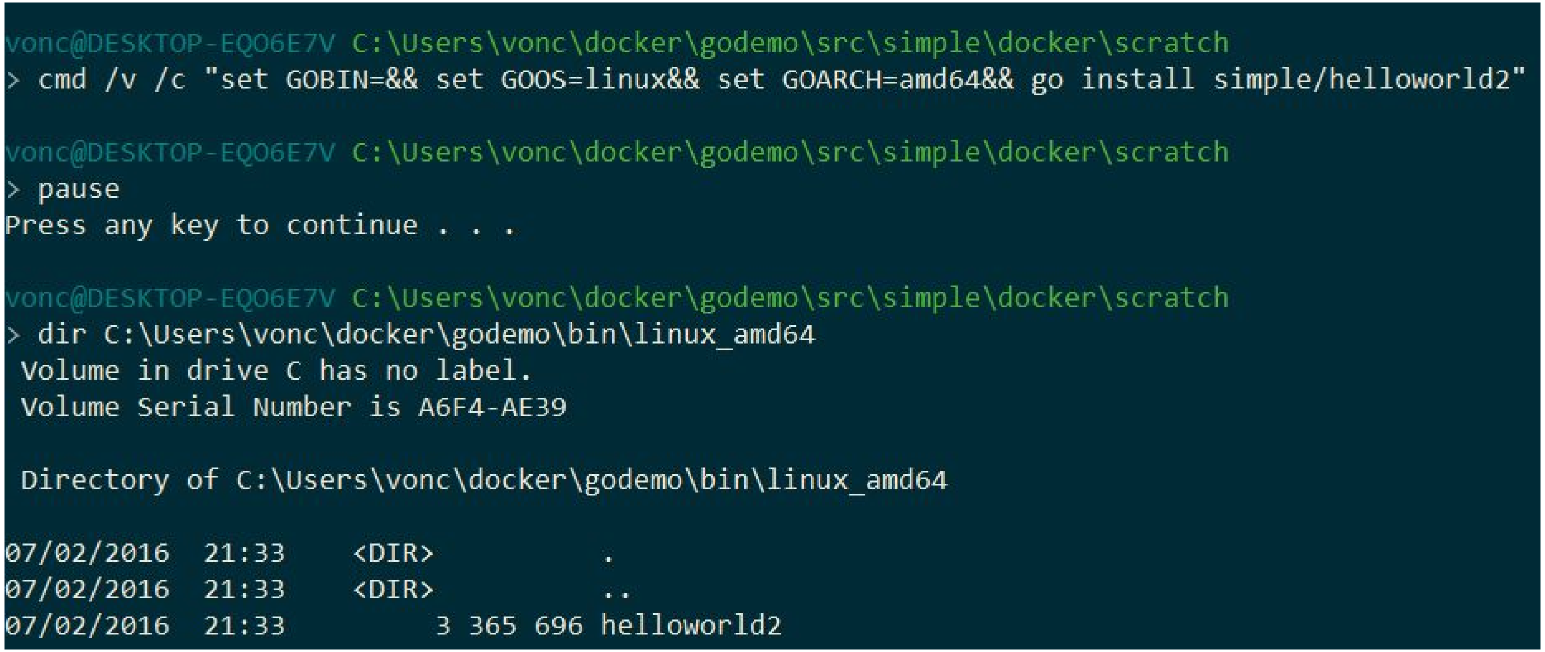


```
docker@default:/c/Users/vonc/docker/godemo/src/simple/docker/scratchexe$ ./gb
Sending build context to Docker daemon 2.137 MB
Step 1 : FROM scratch
 --->
```

```
exec format error
docker: Error response from daemon: Cannot start container c5b9822a6dc8443ae71dc
89e4c26ae6bc6645dbc21b0381792108c5c0d461219: [9] System error: exec format error
.
```

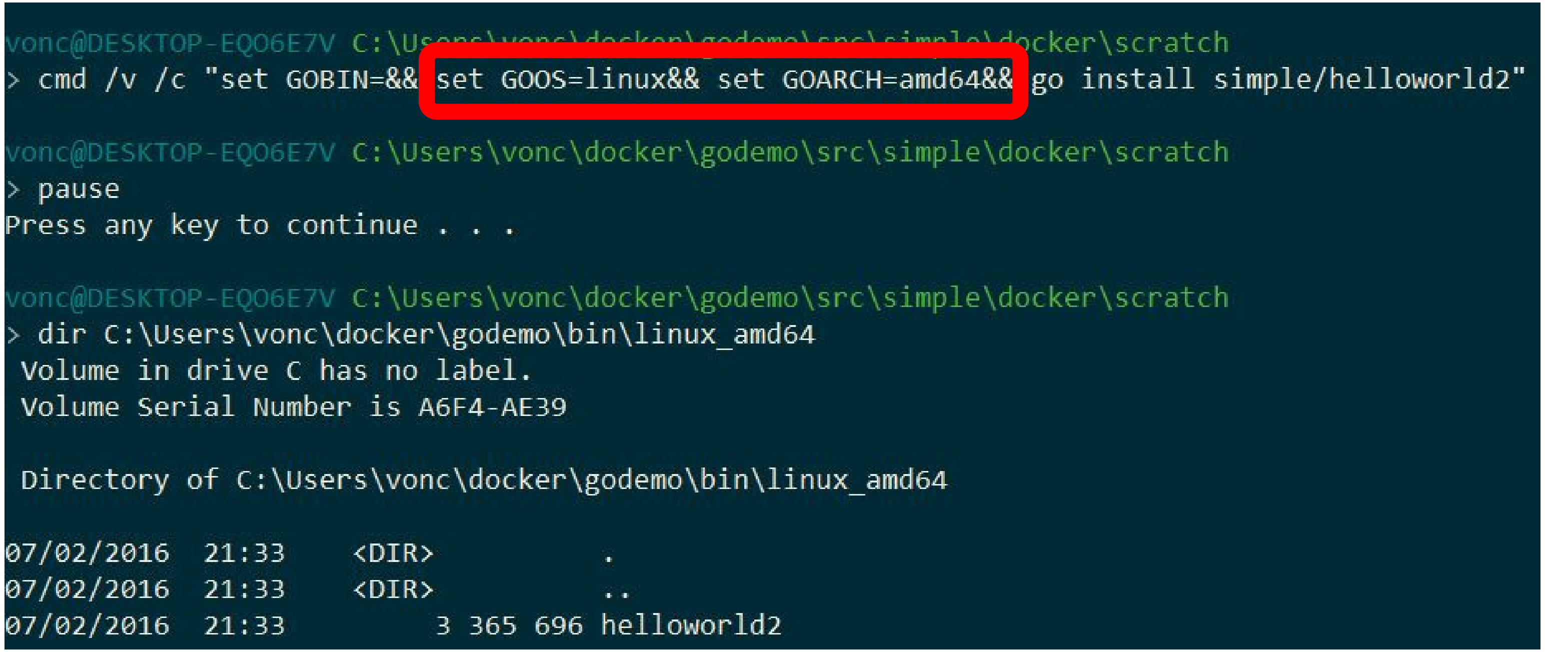
```
 ---> 2892d2d4f7a3
Removing intermediate container a7b83c649afa
Successfully built 2892d2d4f7a3
```

SCRATCH: Linux exe



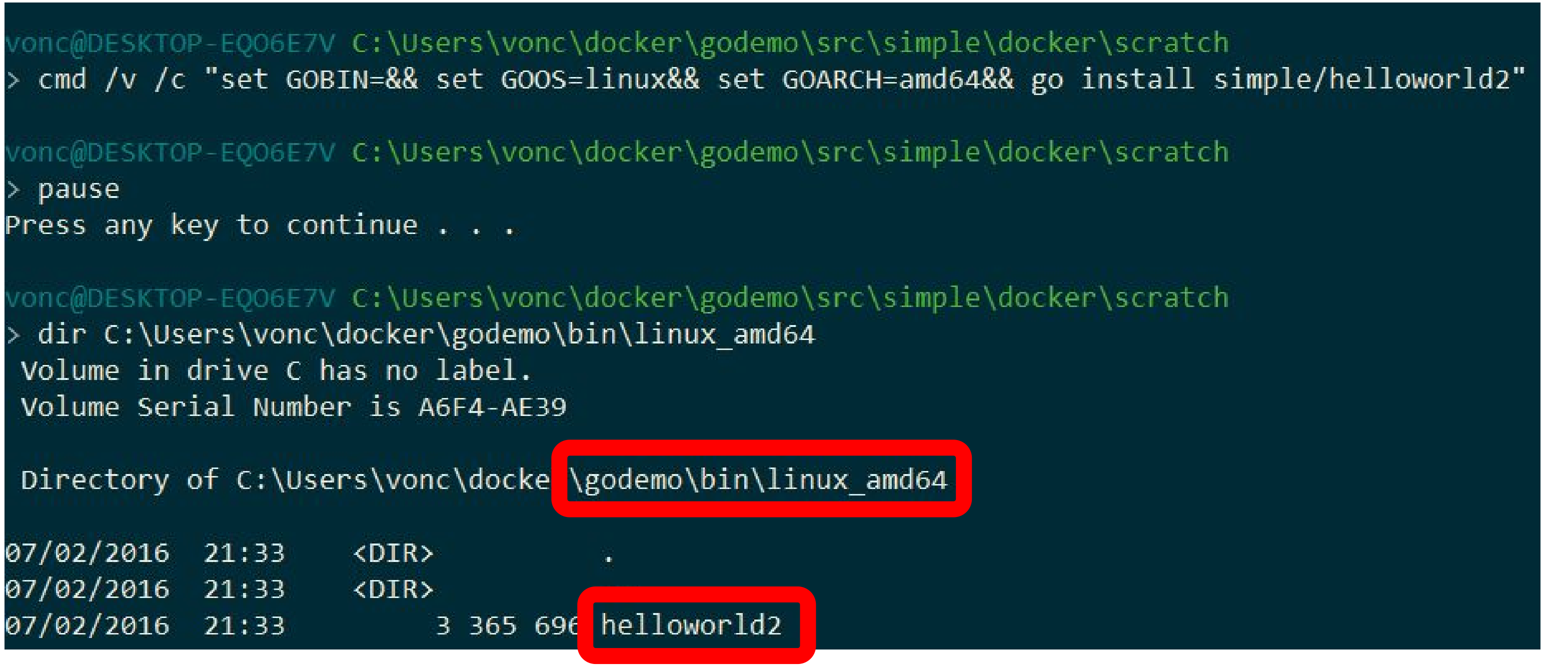
```
vonc@DESKTOP-EQO6E7V C:\Users\vonc\docker\godemo\src\simple\docker\scratch
> cmd /v /c "set GOBIN=&& set GOOS=linux&& set GOARCH=amd64&& go install simple/helloworld2"

vonc@DESKTOP-EQO6E7V C:\Users\vonc\docker\godemo\src\simple\docker\scratch
> pause
Press any key to continue . . .

vonc@DESKTOP-EQO6E7V C:\Users\vonc\docker\godemo\src\simple\docker\scratch
> dir C:\Users\vonc\docker\godemo\bin\linux_amd64
 Volume in drive C has no label.
 Volume Serial Number is A6F4-AE39

 Directory of C:\Users\vonc\docker\godemo\bin\linux_amd64

07/02/2016  21:33    <DIR>          .
07/02/2016  21:33    <DIR>          ..
07/02/2016  21:33         3 365 696 helloworld2
```

SCRATCH: Linux exe



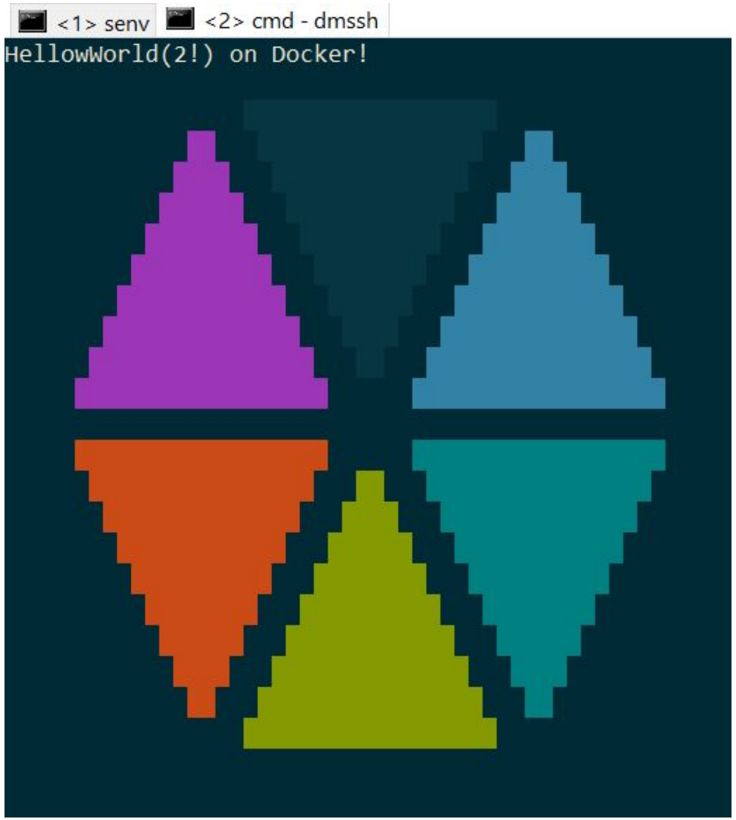
```
vonc@DESKTOP-EQO6E7V C:\Users\vonc\docker\godemo\src\simple\docker\scratch
> cmd /v /c "set GOBIN=&& set GOOS=linux&& set GOARCH=amd64&& go install simple/helloworld2"

vonc@DESKTOP-EQO6E7V C:\Users\vonc\docker\godemo\src\simple\docker\scratch
> pause
Press any key to continue . . .

vonc@DESKTOP-EQO6E7V C:\Users\vonc\docker\godemo\src\simple\docker\scratch
> dir C:\Users\vonc\docker\godemo\bin\linux_amd64
 Volume in drive C has no label.
 Volume Serial Number is A6F4-AE39

 Directory of C:\Users\vonc\docker\godemo\bin\linux_amd64

07/02/2016  21:33    <DIR>          .
07/02/2016  21:33    <DIR>          ..
07/02/2016  21:33         3 365 696 helloworld2
```

SCRATCH: Linux exe



```
vonc@DESKTOP-EQ06E7V C:\Users\vonc\docker\godemo\src\simple\docker\scratch
> cmd /v /c "set GOBIN=&& set GOOS=linux&& set GOARCH=amd64&& go install simple/helloworld2"

vonc@DESKTOP-EQ06E7V C:\Users\vonc\docker\godemo\src\simple\docker\scratch
> pause
Press any key to continue . . .

vonc@DESKTOP-EQ06E7V C:\Users\vonc\docker\godemo\src\simple\docker\scratch
> dir C:\Users\vonc\docker\godemo\bin\linux_amd64
 Volume in drive C has no label.
 Volume Serial Number is A6F4-AE39

 Directory of C:\Users\vonc\docke \godemo\bin\linux_amd64

07/02/2016  21:33    <DIR>          .
07/02/2016  21:33    <DIR>
07/02/2016  21:33         3 365 69  helloworld2
```

SCRATCH: Linux exe



```
HellowWorld(2!) on Docker!
```

Unique

composition

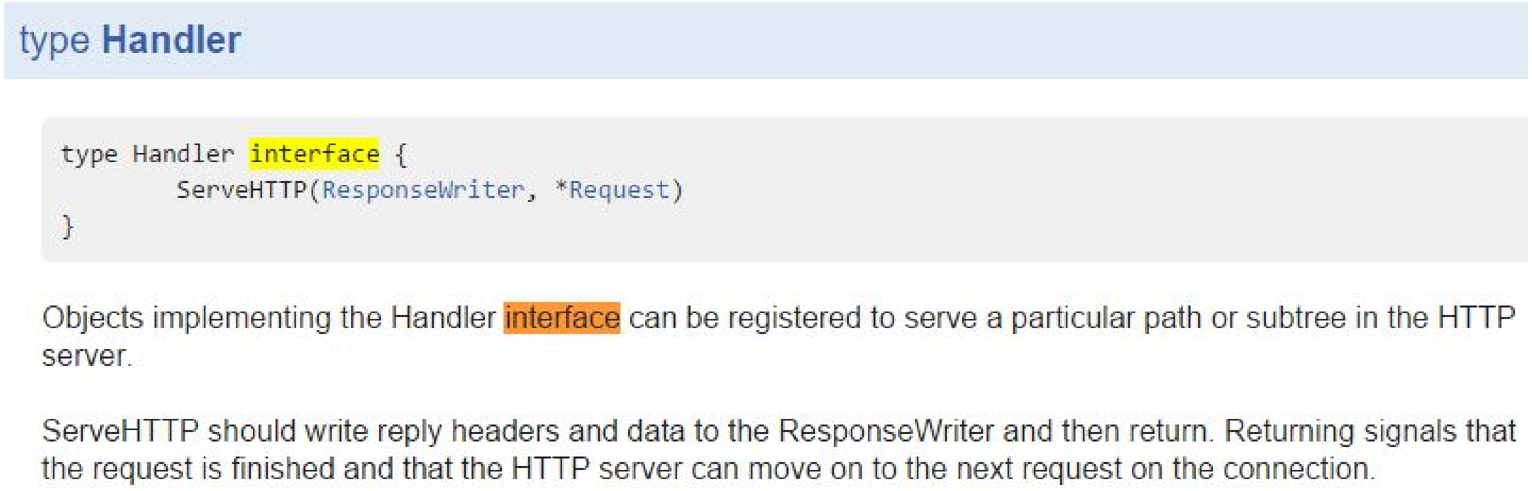
Unique

composition

Hello World in Go (web):

- interface
- goroutine
- channel

Helloworldweb: interface



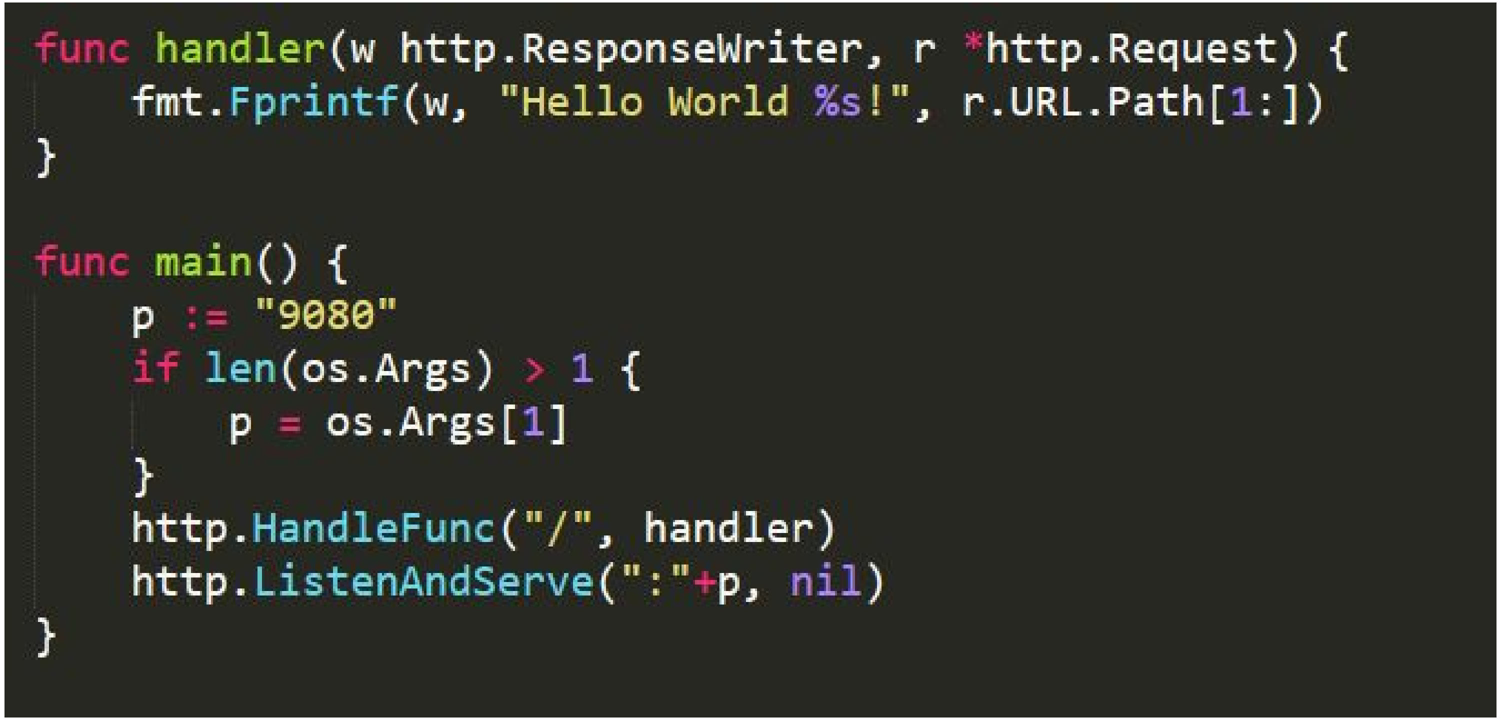
type **Handler**

```
type Handler interface {
        ServeHTTP(ResponseWriter, *Request)
}
```

Objects implementing the Handler interface can be registered to serve a particular path or subtree in the HTTP server.

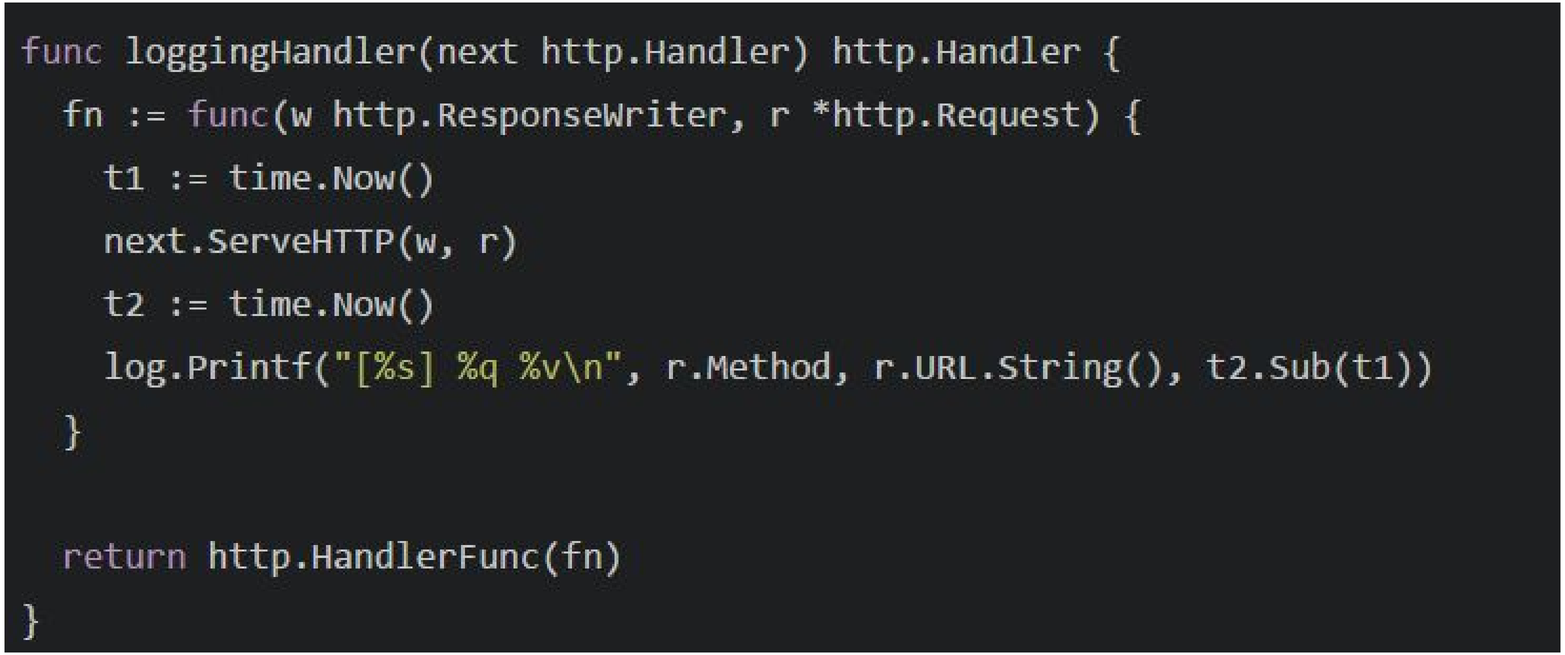
ServeHTTP should write reply headers and data to the ResponseWriter and then return. Returning signals that the request is finished and that the HTTP server can move on to the next request on the connection.

Helloworldweb: interface



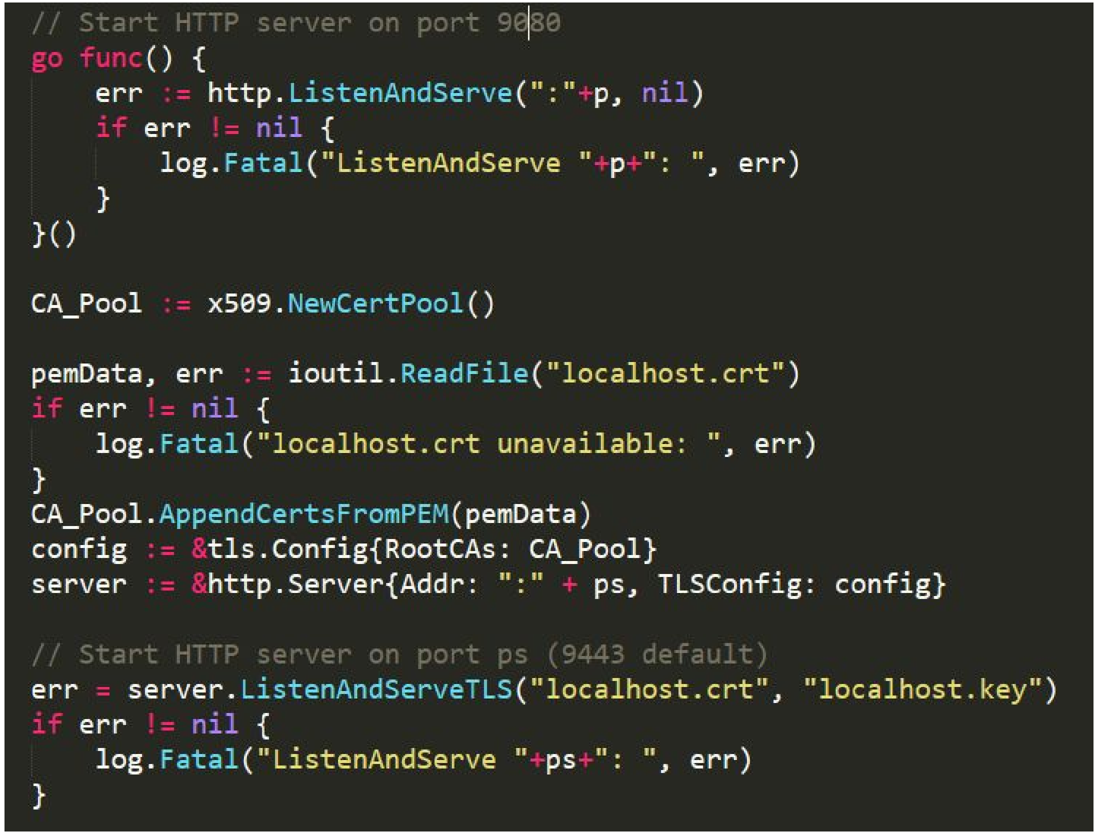
```go
func handler(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "Hello World %s!", r.URL.Path[1:])
}

func main() {
    p := "9080"
    if len(os.Args) > 1 {
        p = os.Args[1]
    }
    http.HandleFunc("/", handler)
    http.ListenAndServe(":"+p, nil)
}
```

Helloworldweb: Middleware



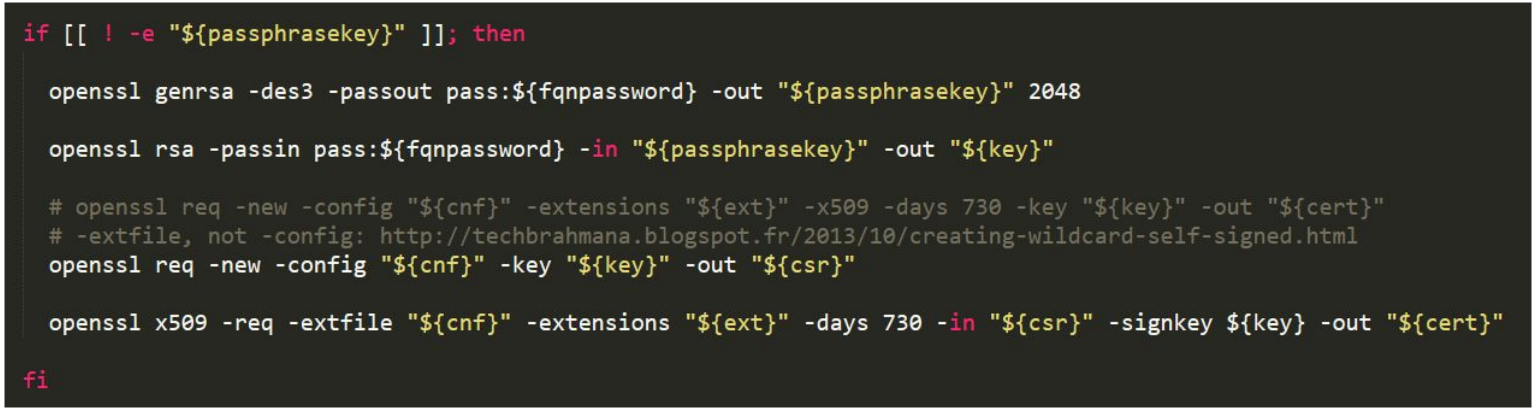
```go
func loggingHandler(next http.Handler) http.Handler {
  fn := func(w http.ResponseWriter, r *http.Request) {
    t1 := time.Now()
    next.ServeHTTP(w, r)
    t2 := time.Now()
    log.Printf("[%s] %q %v\n", r.Method, r.URL.String(), t2.Sub(t1))
  }

  return http.HandlerFunc(fn)
}
```

Helloworldweb https: goroutine



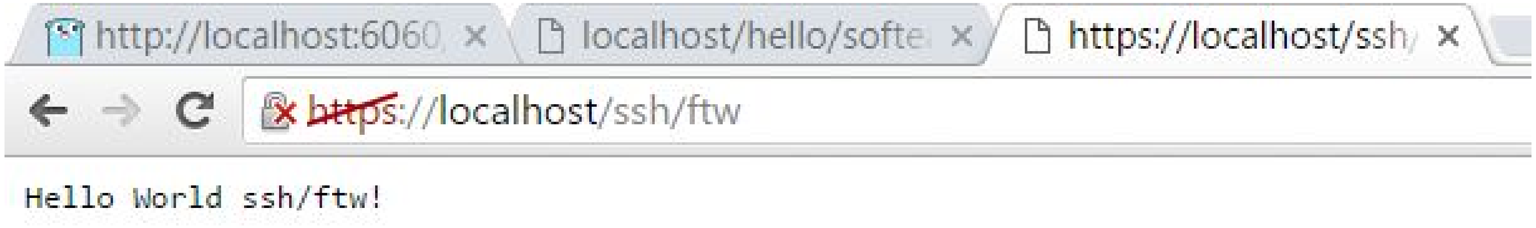
```go
// Start HTTP server on port 9080
go func() {
    err := http.ListenAndServe(":"+p, nil)
    if err != nil {
        log.Fatal("ListenAndServe "+p+": ", err)
    }
}()

CA_Pool := x509.NewCertPool()

pemData, err := ioutil.ReadFile("localhost.crt")
if err != nil {
    log.Fatal("localhost.crt unavailable: ", err)
}
CA_Pool.AppendCertsFromPEM(pemData)
config := &tls.Config{RootCAs: CA_Pool}
server := &http.Server{Addr: ":" + ps, TLSConfig: config}

// Start HTTP server on port ps (9443 default)
err = server.ListenAndServeTLS("localhost.crt", "localhost.key")
if err != nil {
    log.Fatal("ListenAndServe "+ps+": ", err)
}
```

Helloworldweb https: goroutine



```
if [[ ! -e "${passphrasekey}" ]]; then

  openssl genrsa -des3 -passout pass:${fqnpassword} -out "${passphrasekey}" 2048

  openssl rsa -passin pass:${fqnpassword} -in "${passphrasekey}" -out "${key}"

  # openssl req -new -config "${cnf}" -extensions "${ext}" -x509 -days 730 -key "${key}" -out "${cert}"
  # -extfile, not -config: http://techbrahmana.blogspot.fr/2013/10/creating-wildcard-self-signed.html
  openssl req -new -config "${cnf}" -key "${key}" -out "${csr}"

  openssl x509 -req -extfile "${cnf}" -extensions "${ext}" -days 730 -in "${csr}" -signkey ${key} -out "${cert}"

fi
```

Helloworldweb https

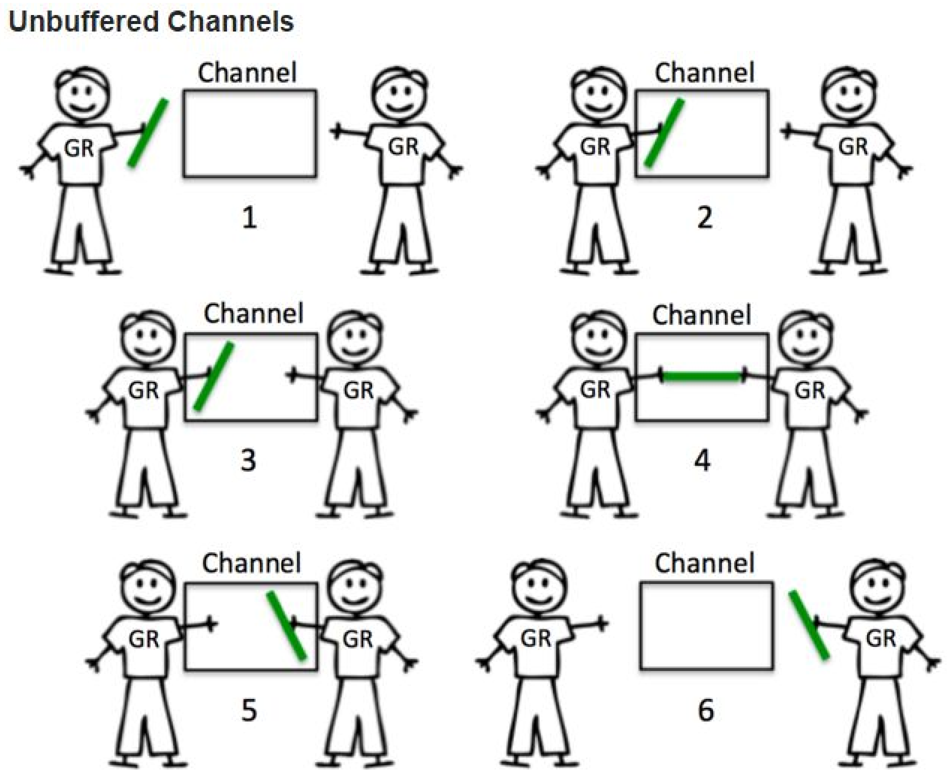


```
Hello World ssh/ftw!
```

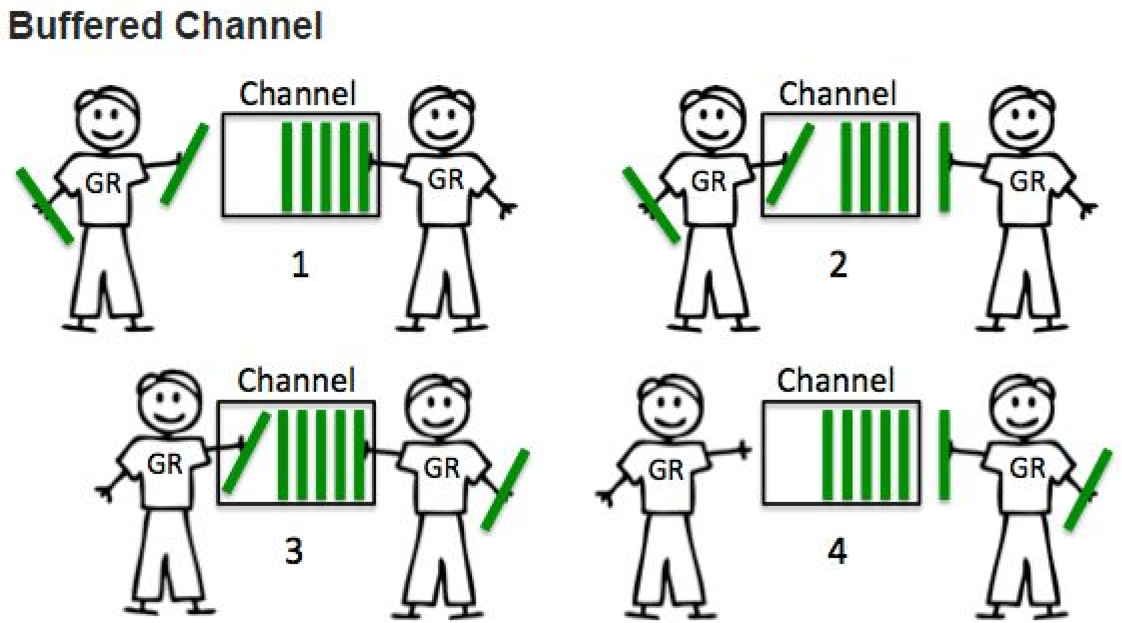
goroutine & channel

Do not communicate by sharing memory;
instead,
share memory by communicating.

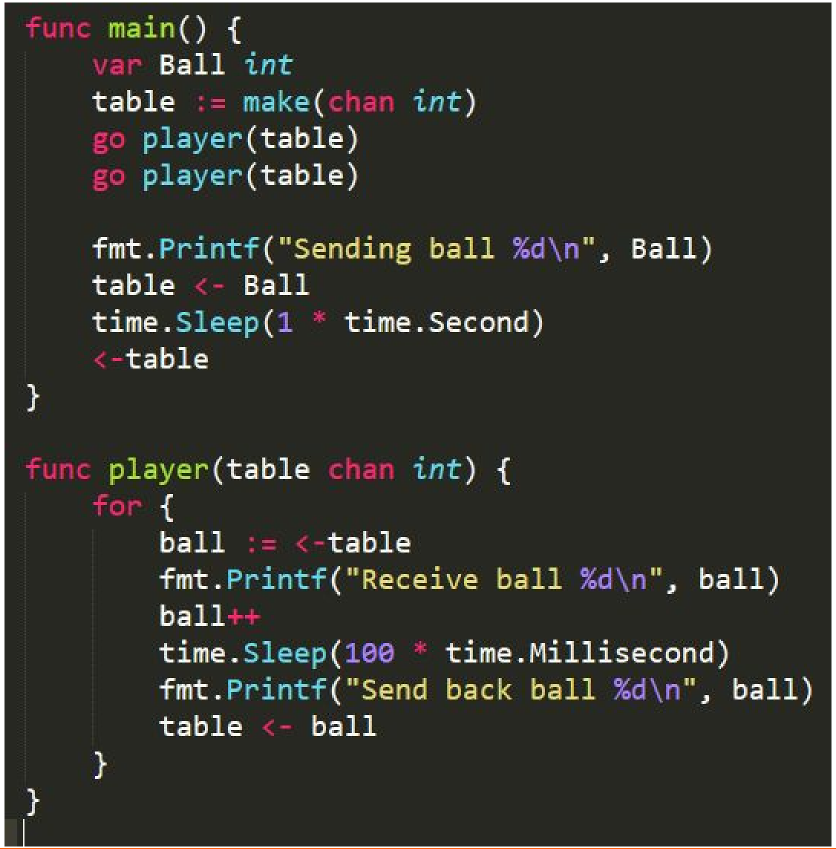
goroutine & channel



goroutine & channel



goroutine & concurrency



```
func main() {
    var Ball int
    table := make(chan int)
    go player(table)
    go player(table)

    fmt.Printf("Sending ball %d\n", Ball)
    table <- Ball
    time.Sleep(1 * time.Second)
    <-table
}

func player(table chan int) {
    for {
        ball := <-table
        fmt.Printf("Receive ball %d\n", ball)
        ball++
        time.Sleep(100 * time.Millisecond)
        fmt.Printf("Send back ball %d\n", ball)
        table <- ball
    }
}
```

goroutine & concurrency

http://divan.github.io/posts/go_concurrency_visualize/

Unique

composition

Hello World in Docker (web):

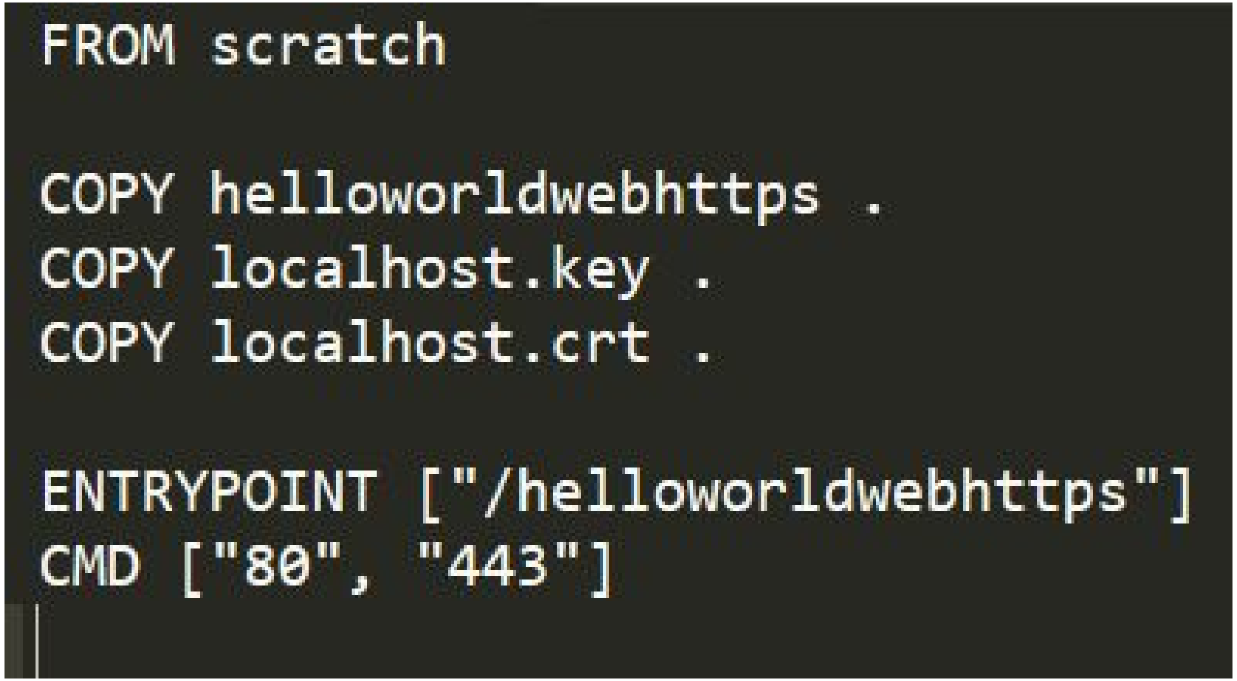
Unique

composition

Hello World in Docker (web):

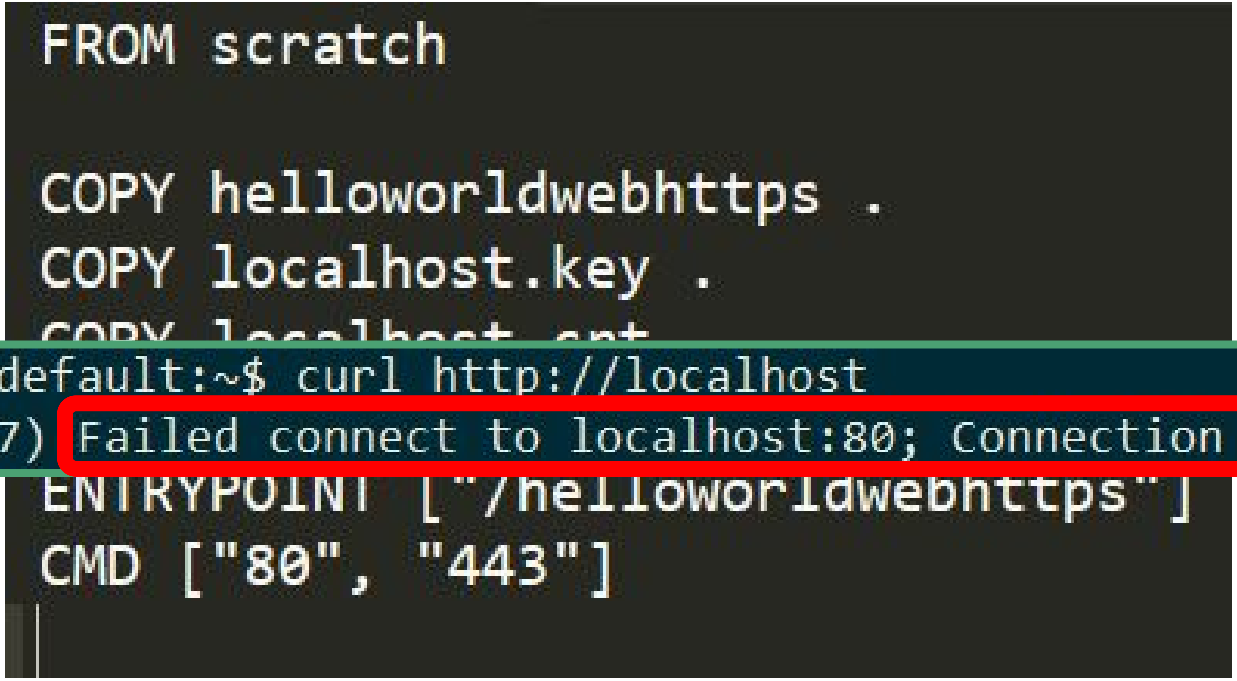
- Dockerfile
- EXPOSE
- port-forward

Dockerfile

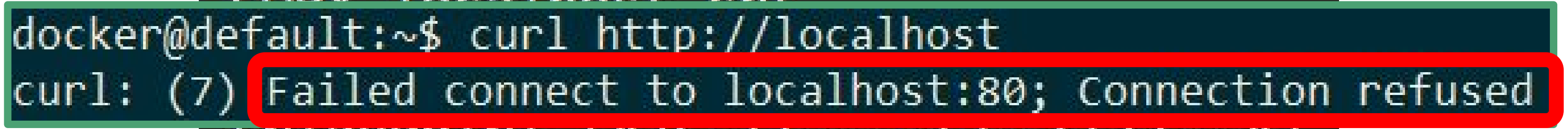


```
FROM scratch

COPY helloworldwebhttps .
COPY localhost.key .
COPY localhost.crt .

ENTRYPOINT ["/helloworldwebhttps"]
CMD ["80", "443"]
```

Dockerfile

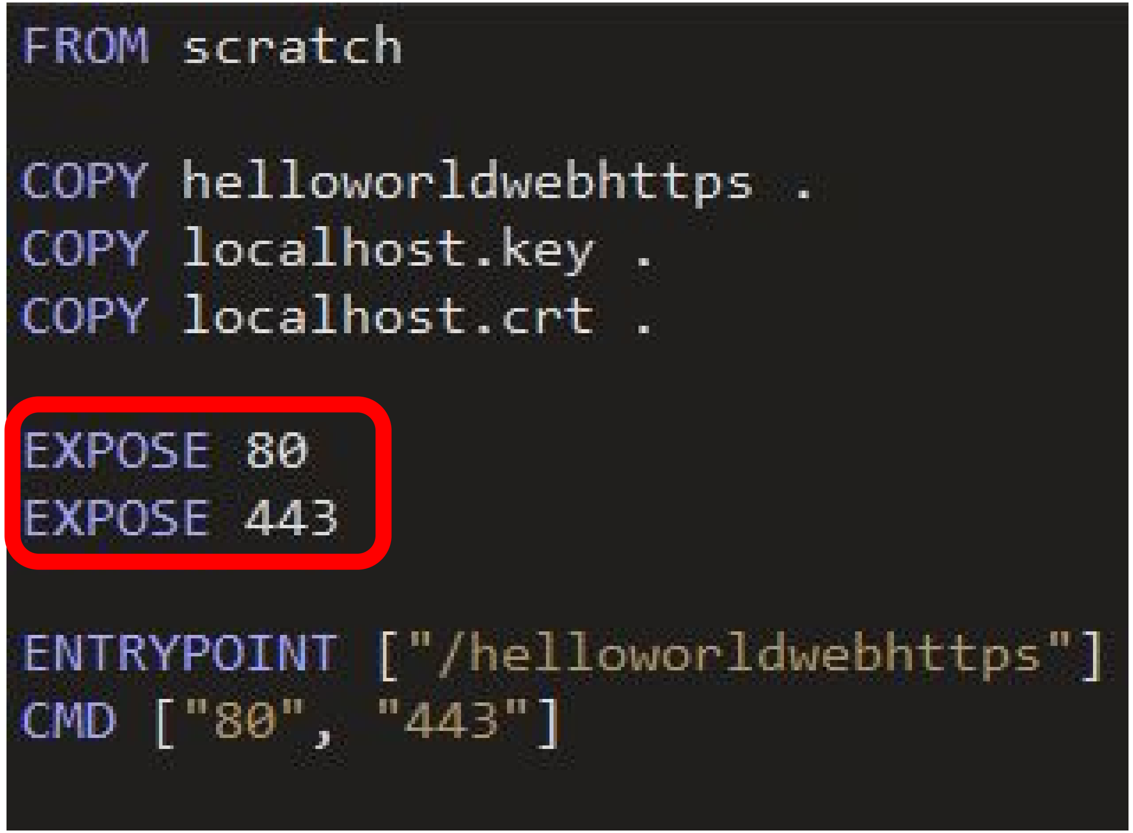


```
FROM scratch

COPY helloworldwebhttps .
COPY localhost.key .
COPY localhost.crt
ENTRYPOINT ["/helloworldwebhttps"]
CMD ["80", "443"]
```



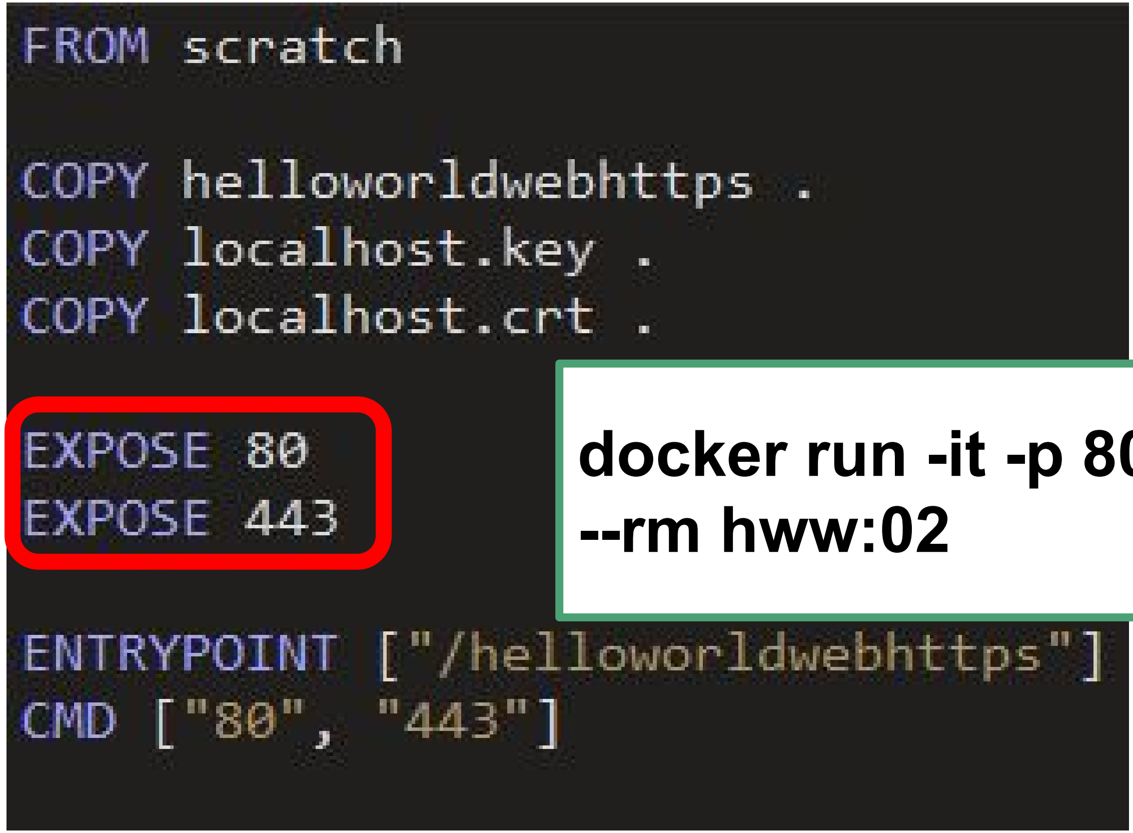
```
docker@default:~$ curl http://localhost
curl: (7) Failed connect to localhost:80; Connection refused
```

EXPOSE



```
FROM scratch

COPY helloworldwebhttps .
COPY localhost.key .
COPY localhost.crt .

EXPOSE 80
EXPOSE 443

ENTRYPOINT ["/helloworldwebhttps"]
CMD ["80", "443"]
```

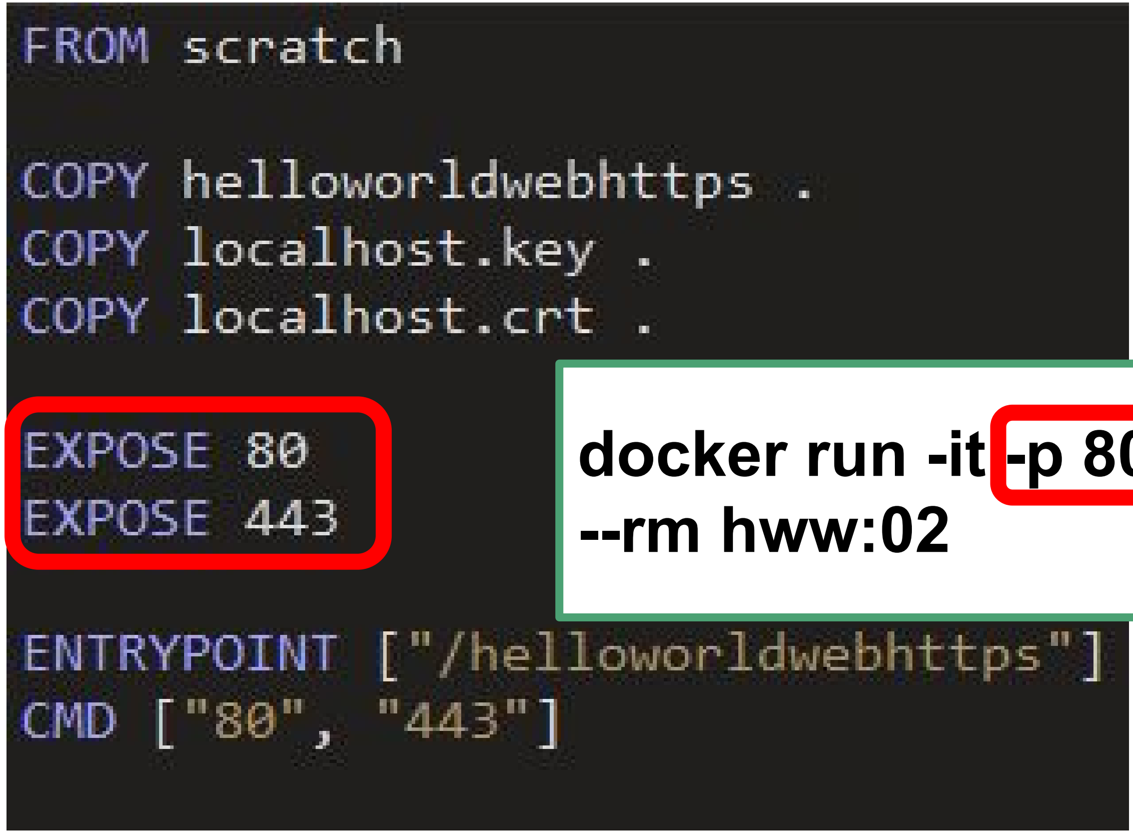
EXPOSE



```
FROM scratch

COPY helloworldwebhttps .
COPY localhost.key .
COPY localhost.crt .

EXPOSE 80
EXPOSE 443

ENTRYPOINT ["/helloworldwebhttps"]
CMD ["80", "443"]
```

docker run -it -p 80:80 -p 443:443 --rm hww:02

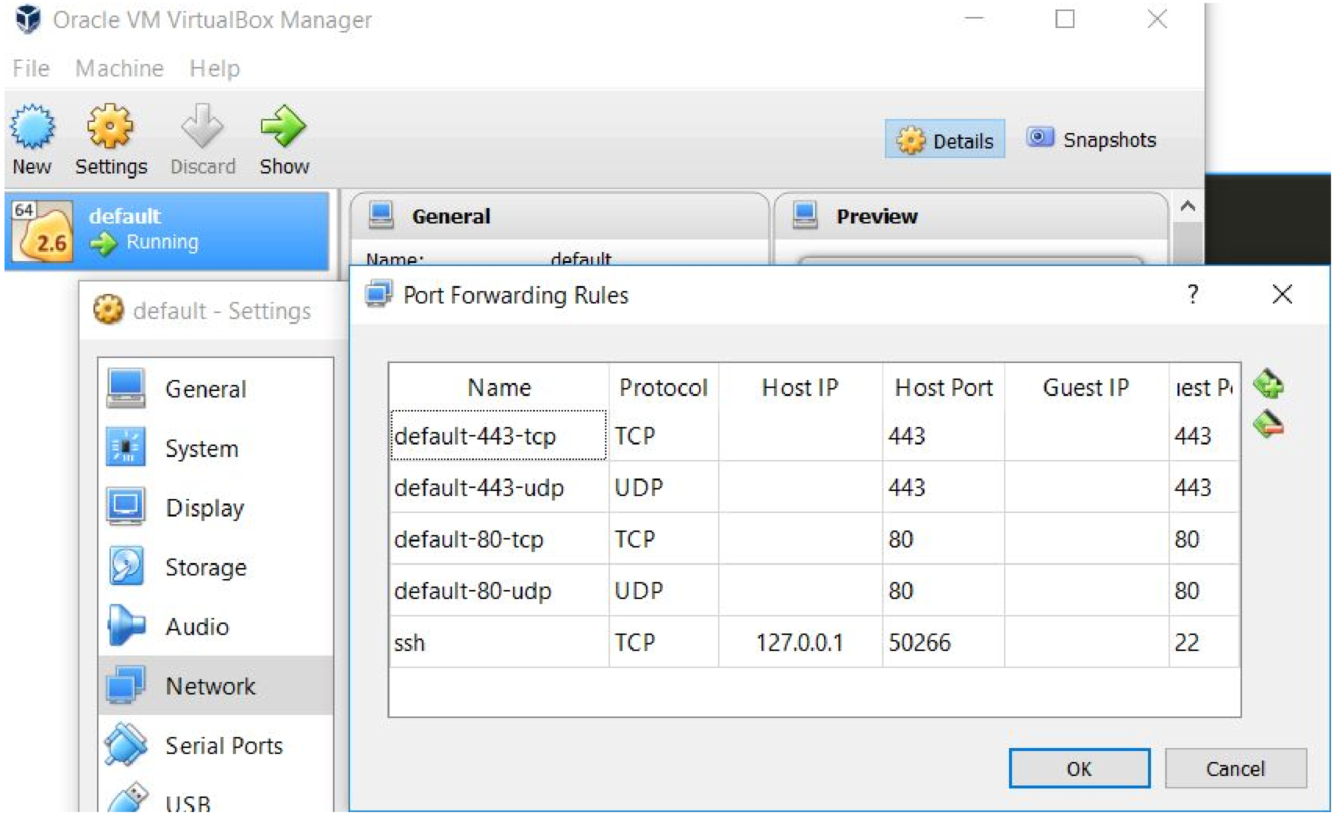
EXPOSE



```
FROM scratch

COPY helloworldwebhttps .
COPY localhost.key .
COPY localhost.crt .

EXPOSE 80
EXPOSE 443

ENTRYPOINT ["/helloworldwebhttps"]
CMD ["80", "443"]
```

docker run -it -p 80:80 -p 443:443 --rm hww:02

Port-Forward



Name	Protocol	Host IP	Host Port	Guest IP	iest P
default-443-tcp	TCP		443		443
default-443-udp	UDP		443		443
default-80-tcp	TCP		80		80
default-80-udp	UDP		80		80
ssh	TCP	127.0.0.1	50266		22

DevOps

isolation

DevOps

isolation

Hello World!

- Dev 2 Ops
- Ops 2 Dev
- Conclusion

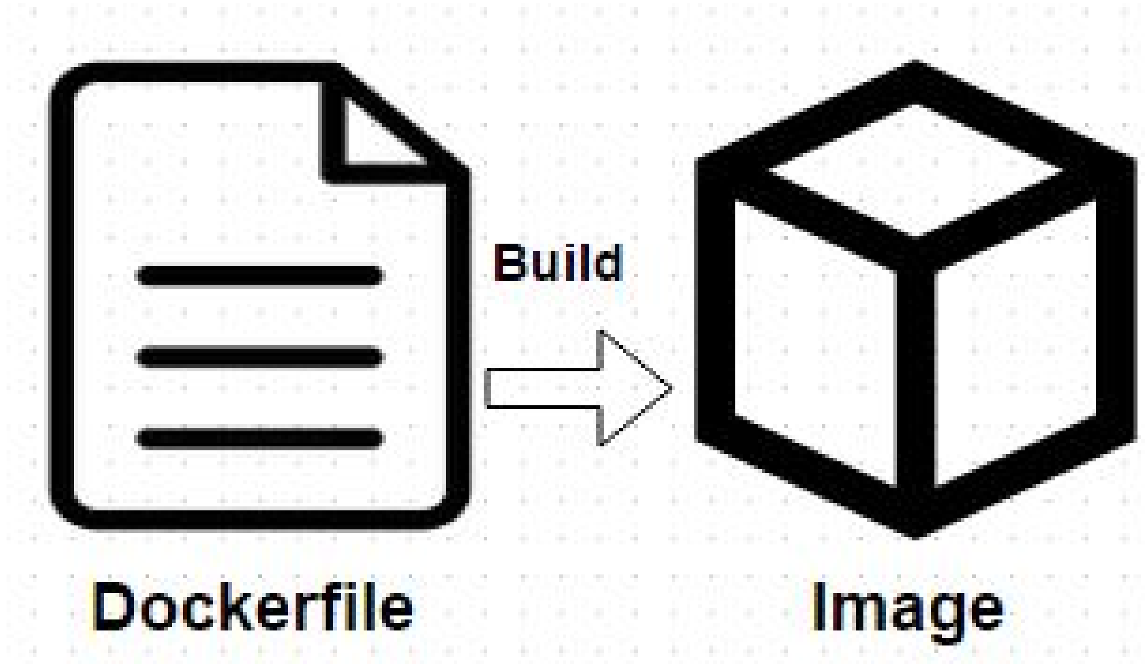
DevOps

isolation

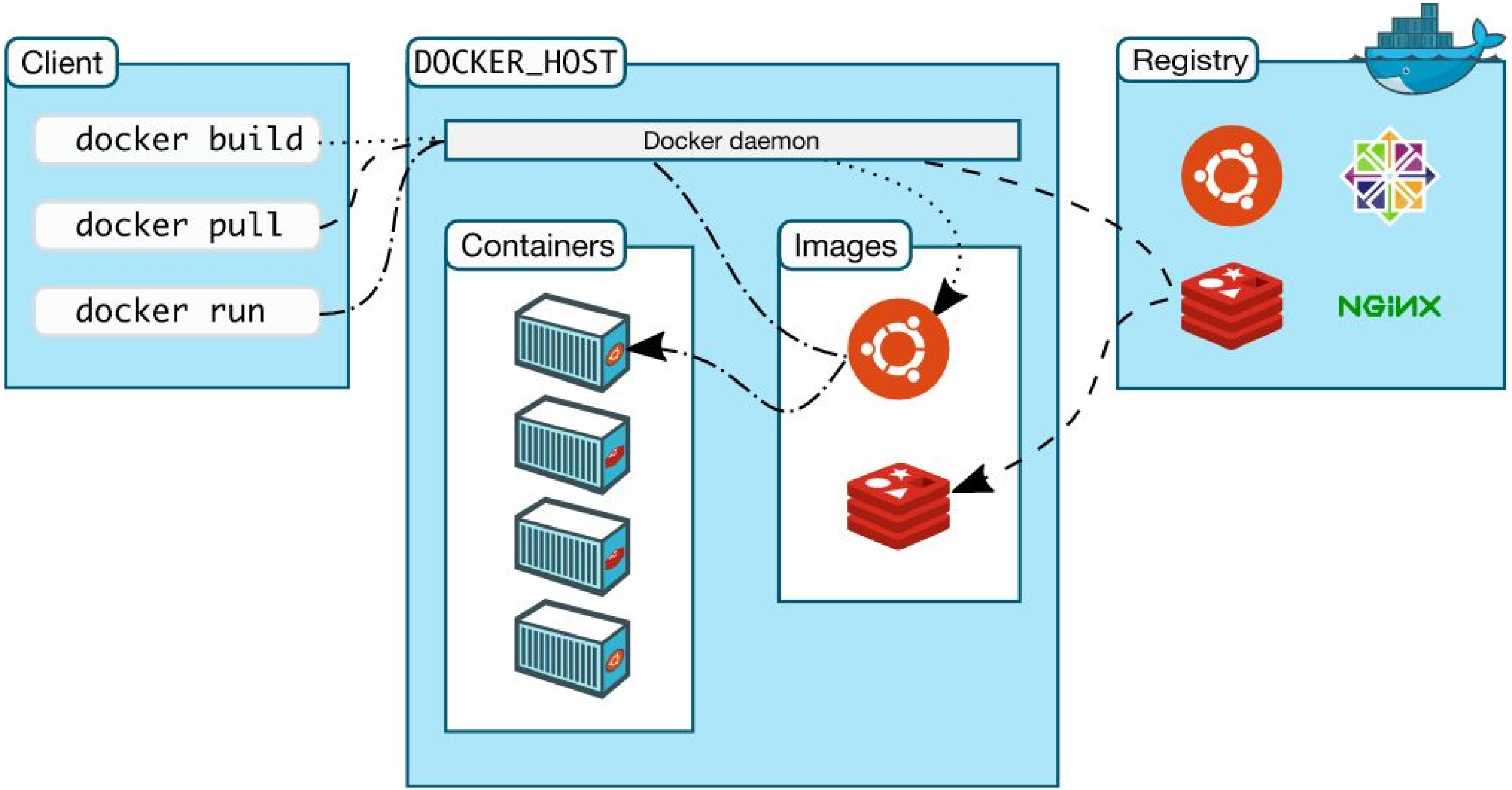
Hello World!

Dev2Ops

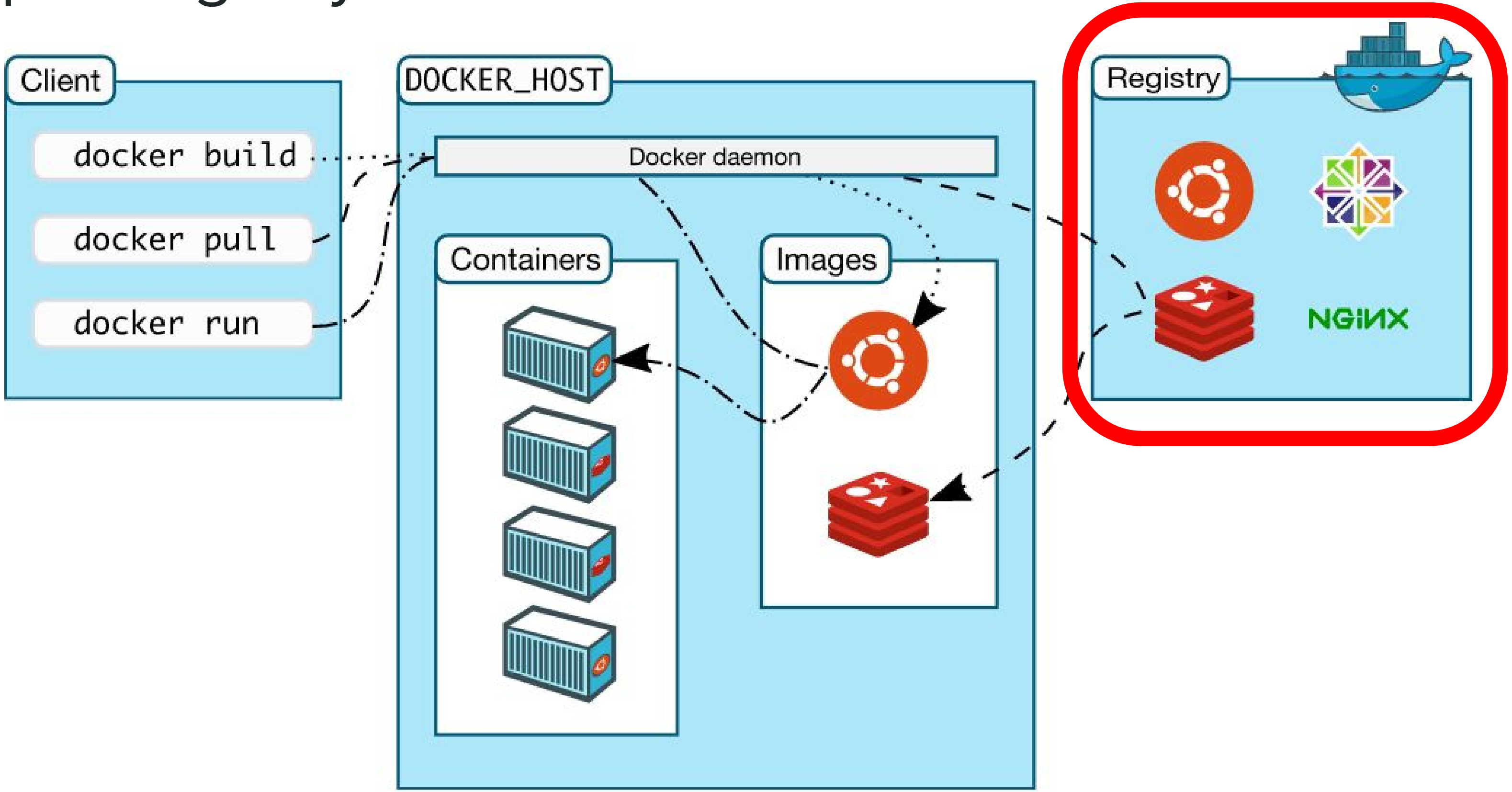
Dev2Ops: simple as build?



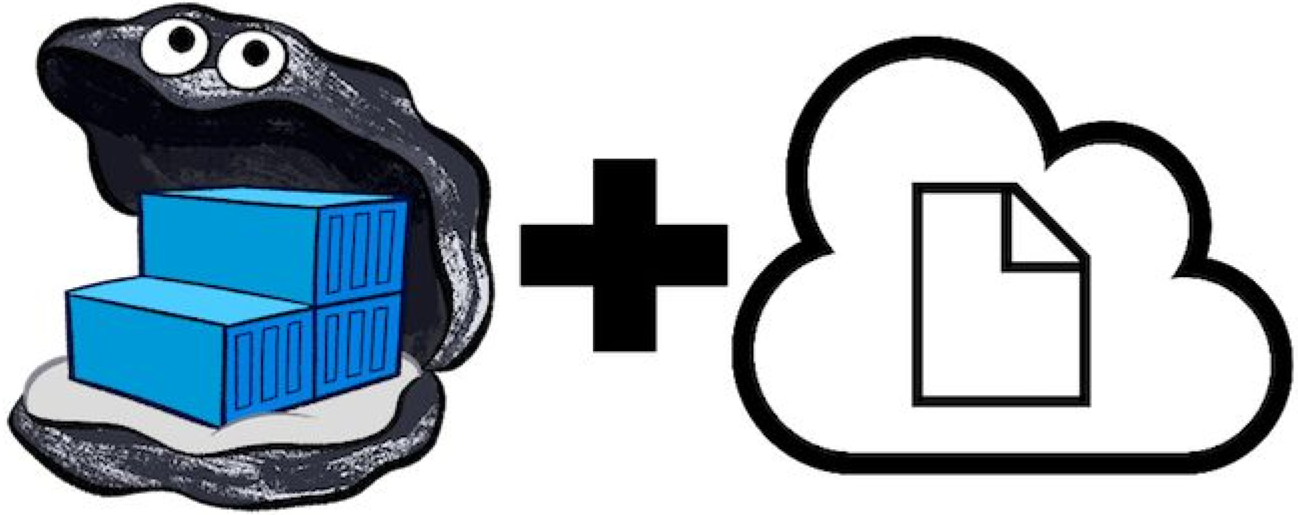
Dev2Ops: Registry...



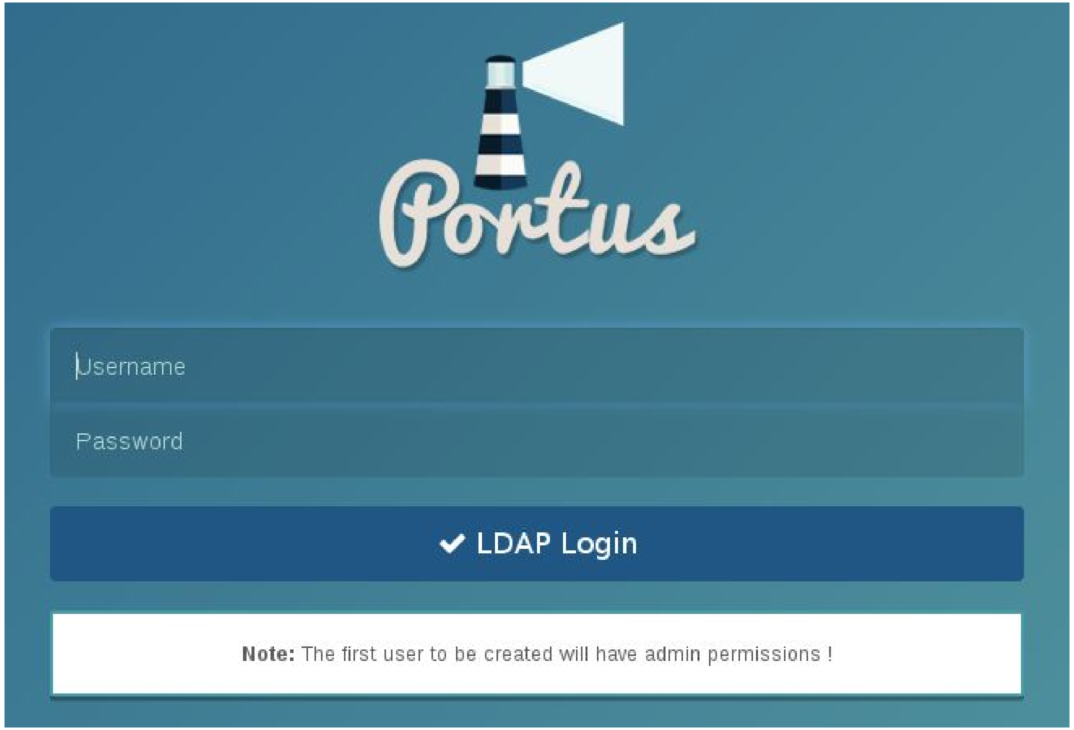
Dev2Ops: Registry...



Dev2Ops: Registry...



Dev2Ops: Portus!?



Note: The first user to be created will have admin permissions !

http://port.us.org/

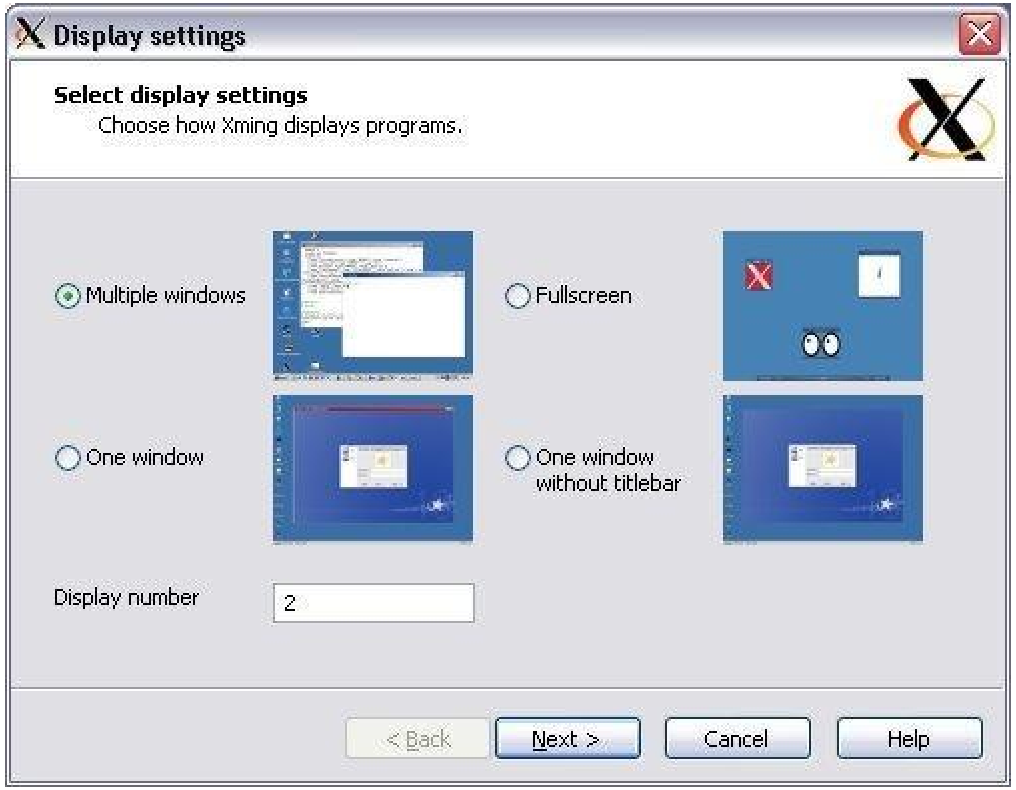
DevOps

isolation

Hello World!

Ops2Dev

Ops2Dev: X11 server



Display settings

Select display settings

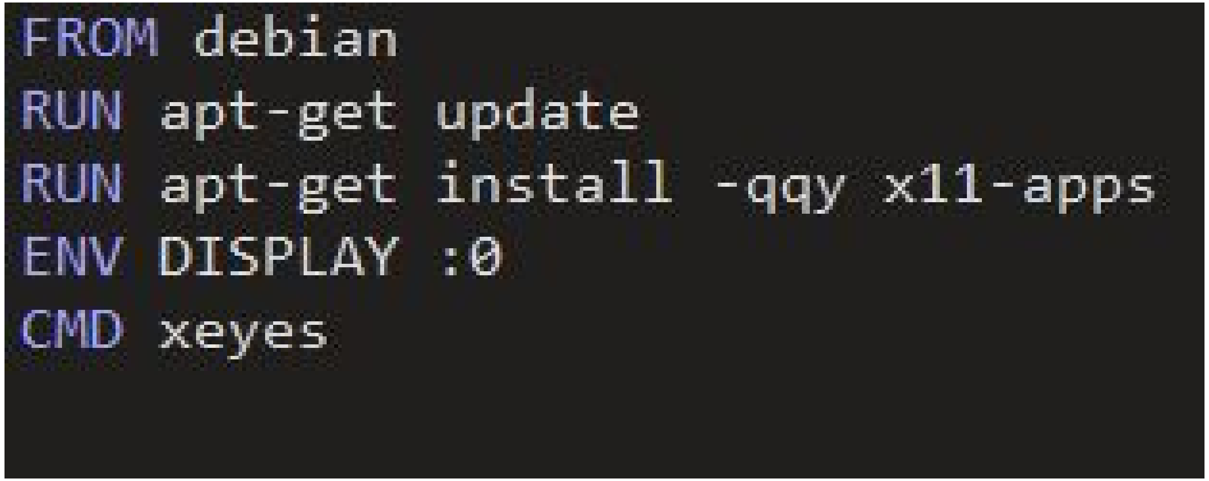
Choose how Xming displays programs.

[✓ Multiple windows] Fullscreen

One window One window without titlebar

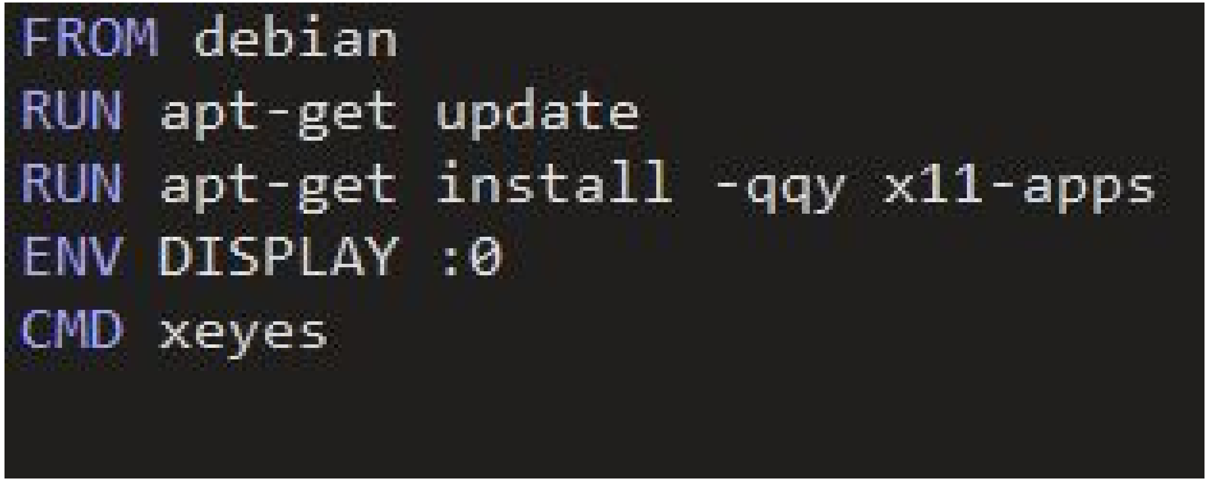
Display number: 2

Ops2Dev: X11 server



```
FROM debian
RUN apt-get update
RUN apt-get install -qqy x11-apps
ENV DISPLAY :0
CMD xeyes
```

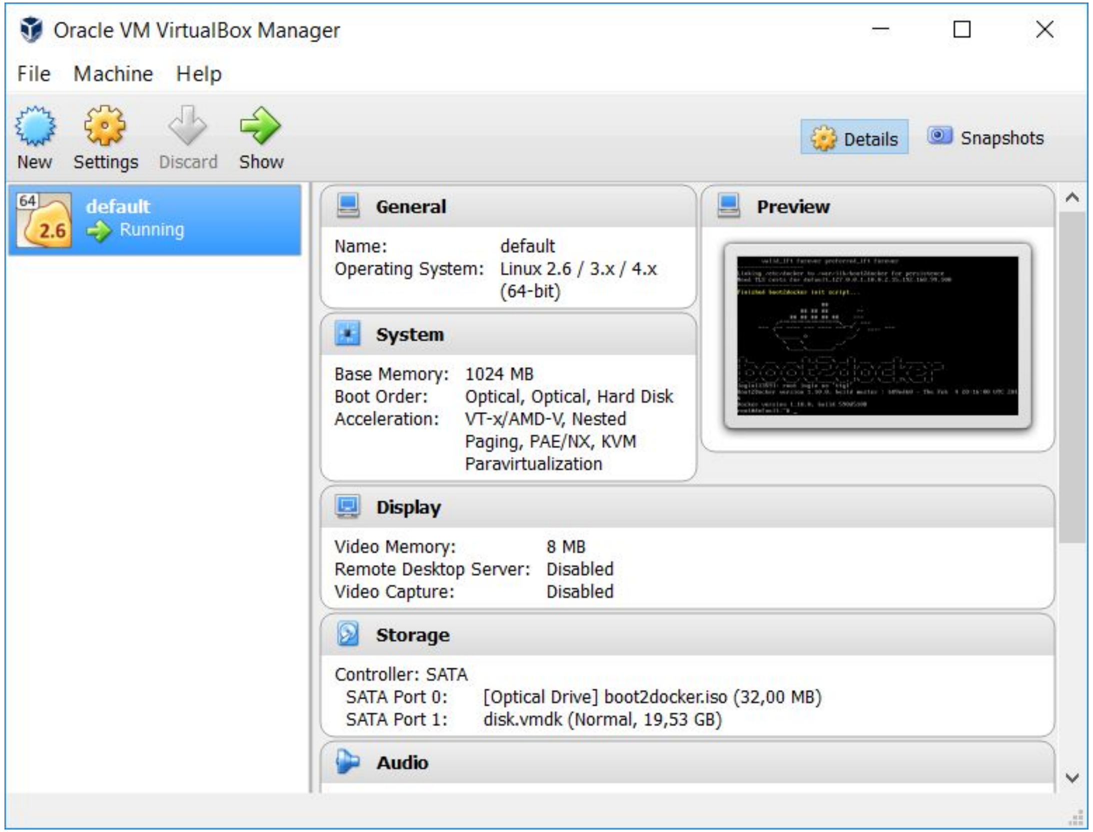
Ops2Dev: X11 server



```
FROM debian
RUN apt-get update
RUN apt-get install -qqy x11-apps
ENV DISPLAY :0
CMD xeyes
```

docker run -it -e DISPLAY=<ping -4 $(hostname)>:10 --rm xeye

Ops2Dev: VirtualBox...



Name: default
Operating System: Linux 2.6 / 3.x / 4.x (64-bit)

Base Memory: 1024 MB
Boot Order: Optical, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, PAE/NX, KVM Paravirtualization

Video Memory: 8 MB
Remote Desktop Server: Disabled
Video Capture: Disabled

Controller: SATA
SATA Port 0: [Optical Drive] boot2docker.iso (32,00 MB)
SATA Port 1: disk.vmdk (Normal, 19,53 GB)

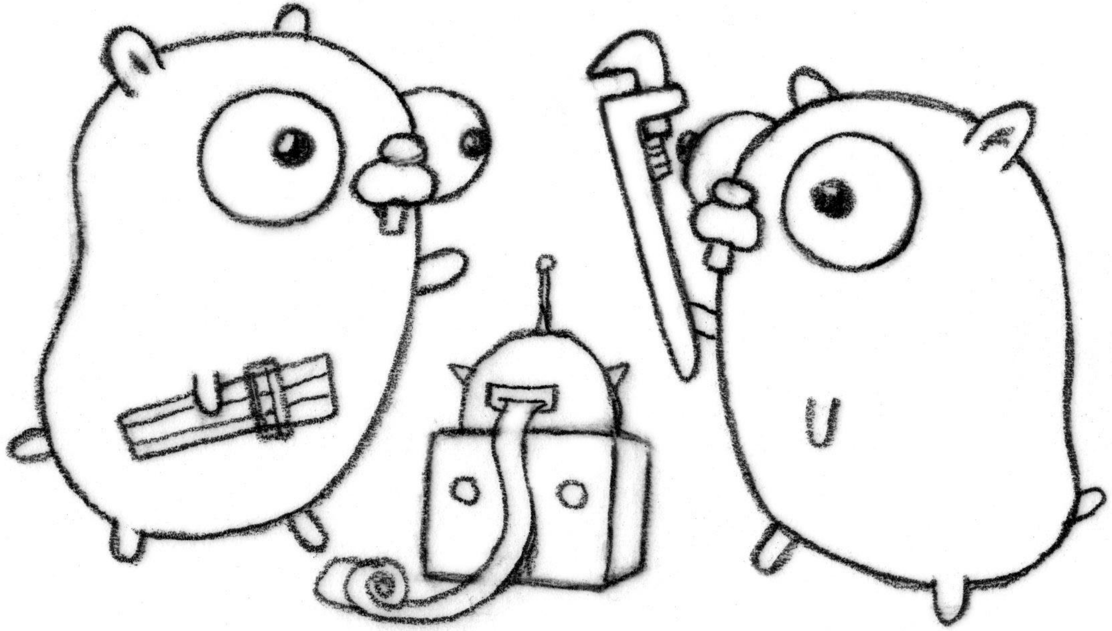
DevOps

isolation

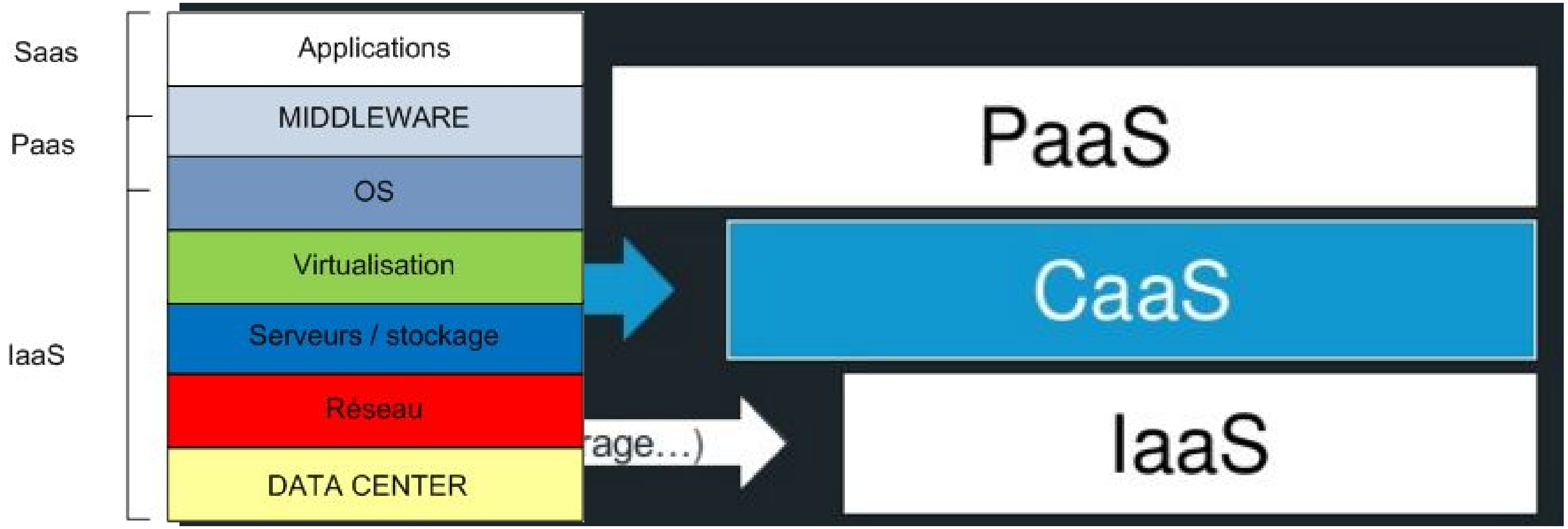
Conclusion

Brace yourself: go & docker are coming!

Conclusion: go (go doc, go test, go get)

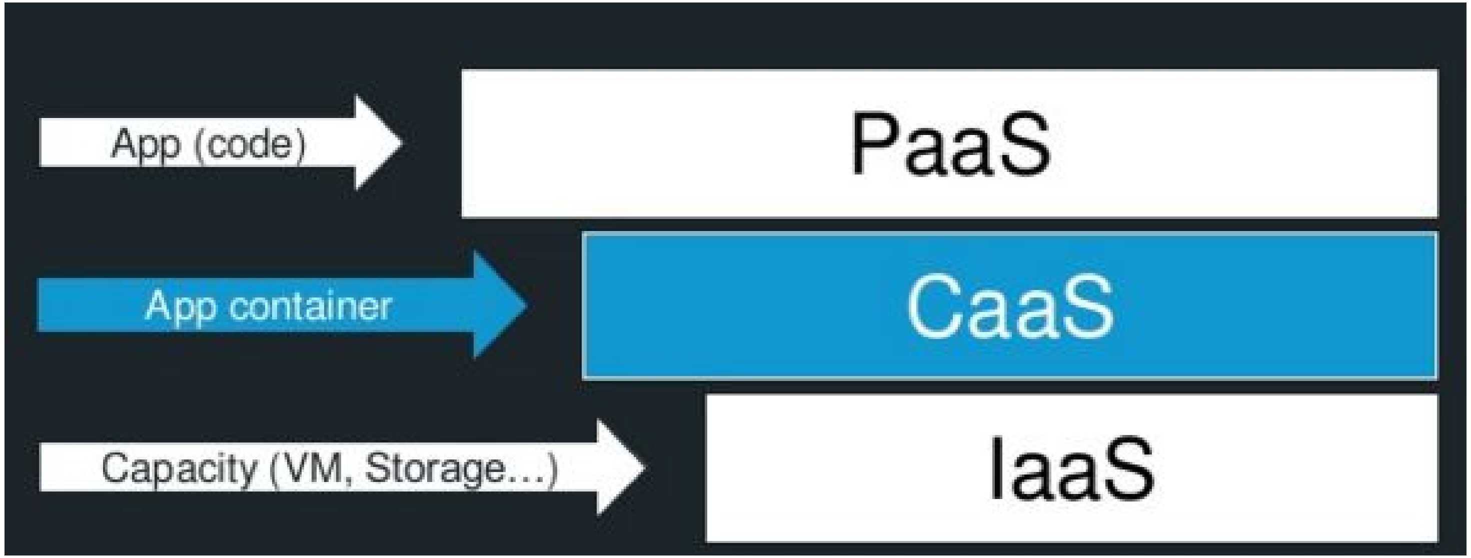


Conclusion: docker (CaaS)



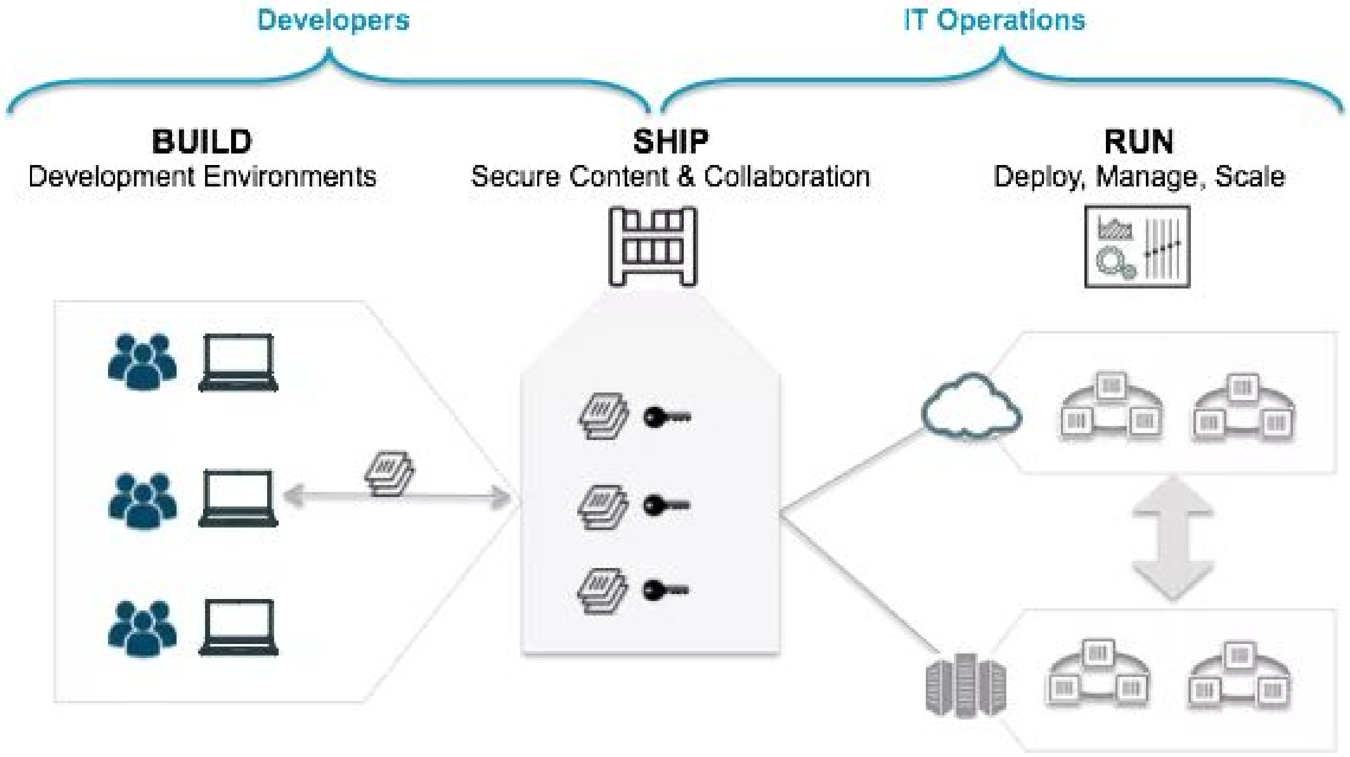
http://www.slideshare.net/Docker/continuous-delivery-leveraging-on-docker-caas-by-adrien-blind

Conclusion: docker (CaaS)



http://www.slideshare.net/Docker/continuous-delivery-leveraging-on-docker-caas-by-adrien-blind

Conclusion: docker (CaaS)


Developers
IT Operations
BUILD
Development Environments
SHIP
Secure Content & Collaboration
RUN
Deploy, Manage, Scale