

Go and Docker

Dev&Ops illustrated



Daniel CHAFFIOL

Softteam Cadextan
(1999)



Amundi Asset Management



HSBC



Société Générale



BNP-Paribas



VonC

Stack Overflow

(2008)



Git - Hg – SVN - ClearCase



Golang - Docker



Eclipse - Jenkins

slideshare

The screenshot shows the SlideShare website interface. At the top, there is a navigation bar with the SlideShare logo, a search bar, and an 'Upload' button. Below the navigation bar, there are tabs for 'Home', 'Technology', 'Education', 'More Topics', and 'My Clipboards'. On the right side of the navigation bar, there are links for 'For Uploaders' and 'Collect Leads'. The main content area displays a presentation titled 'Go and Docker' with the subtitle 'Dev&Ops illustrated'. The presentation is currently on slide 1 of 30. To the right of the main content area, there is a 'Recommended' section with four recommendations, each featuring a thumbnail image and a title. The recommendations are: 'Docker architecture rework case study', 'Git meanings of -distributed-', 'Git dvcs and Information Security Review', and 'DVCS in big corporation'.

in SlideShare | Search

Home Technology Education More Topics My Clipboards For Uploaders Collect Leads

Upload

Go and Docker

Dev&Ops illustrated

1 of 30

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

The screenshot shows a SlideShare interface. The main presentation area displays a dark slide with the title "Go and Docker" in large white font, and "Dev&Ops illustrated" in a smaller white font below it. A progress bar at the bottom indicates "1 of 30" slides. The right sidebar features a "Recommended" section with four items, each with a thumbnail and a title. The first item, "Docker architecture rework case study" by dchaffiol, is highlighted with a red rectangular border. The other recommended items are "Git: a 'Distributed' VCS?", "Git dvcs and Information Security Review" by dchaffiol, and "DVCS in big corporation" by dchaffiol. The top navigation bar includes the SlideShare logo, a search bar, and links for Home, Technology, Education, More Topics, My Clipboards, For Uploaders, and Collect Leads. An "Upload" button and a user profile picture are also visible in the top right.

in SlideShare | Search

Home Technology Education More Topics My Clipboards For Uploaders Collect Leads

Upload

Recommended

Docker
Architecture rework case study
dchaffiol

Git: a "Distributed" VCS?
dchaffiol

Git dvcs and Information Security Review
dchaffiol


DVCS in big Corporation
dchaffiol

Go and Docker
Dev&Ops illustrated

1 of 30

slideshare

The screenshot shows the SlideShare interface. At the top is a dark navigation bar with the LinkedIn logo, the text 'SlideShare', a search bar, an 'Upload' button, and a user profile picture. Below this is a lighter navigation bar with links for 'Home', 'Technology', 'Education', 'More Topics', 'My Clipboards', 'For Uploaders', and 'Collect Leads'. The main content area features a large slide with a dark background. The slide text reads 'Docker' in large white font, followed by 'Architecture Rework Case Study' in smaller white font. A red horizontal line separates the two lines of text. In the top right corner of the slide, there is a blue button that says 'Clip slide'. Above the slide, a small text prompt says 'Be the first to clip this slide'. At the bottom of the slide, there is a navigation bar with a left arrow, the text '1 of 67', a right arrow, and a share icon. To the right of the main slide is a 'Recommended' section with a list of video thumbnails and titles: 'Windows 10 Essential Training' by lynda.com, 'Photoshop CC Essential Training (2015)' by lynda.com, 'Project Management Fundamentals' by lynda.com, 'Git: a "Distributed" VCS?' by dchaffiol, and 'Git dvcs and Information Security Review'.

in SlideShare | Search 


Home Technology Education More Topics My Clipboards For Uploaders Collect Leads

Be the first to clip this slide






Clip slide

Docker

Architecture Rework Case Study

1 of 67 

Recommended

-  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: a "Distributed" VCS?
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 gol).

GIT-GO-DOCKER?



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

softeam.fr



The screenshot shows a web browser window with the address bar displaying `softeam.fr/details-actualite/?com=1923`. The website header features the logo "SOFTEAM Cadextan" and a navigation menu with icons and labels: "QUI SOMMES-NOUS", "NOS DOMAINES MÉTIERS", "NOS OFFRES", "NOS PRESTATIONS", "NOS SOLUTIONS", and "NOS AD". The main content area has a title "Git-Go-Docker? Take it Easy!" followed by a small image of a frog. The text below reads: "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!).

Below the text, there is a graphic with the text "GIT-GO-DOCKER?" and an illustration of a green frog sitting on a log, with the text "TAKE IT EASY!" below it.

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

GitHub

The screenshot shows the GitHub web interface for the repository 'VonC/godemo'. The browser address bar shows the URL 'https://github.com/VonC/godemo'. The repository page includes a search bar, navigation links for 'Pull requests', 'Issues', and 'Gist', and a header with repository statistics: 'Unwatch' (1), 'Star' (0), and 'Fork' (0). Below the header, there are tabs for 'Code', 'Issues' (0), 'Pull requests' (0), 'Wiki', 'Pulse', 'Graphs', and 'Settings'. The main content area shows the repository description 'For demo about Go and Docker — Edit', followed by a progress bar indicating '14 commits', '1 branch', '0 releases', and '1 contributor'. Below the progress bar, there are buttons for 'New pull request', 'New file', 'Find file', 'SSH' (with a dropdown), and 'Download ZIP'. A table lists the repository's files and their commit history:

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

At the bottom, there is a blue box with the text 'Help people interested in this repository understand your project by adding a README.' and a green button labeled 'Add a README'.

Go:

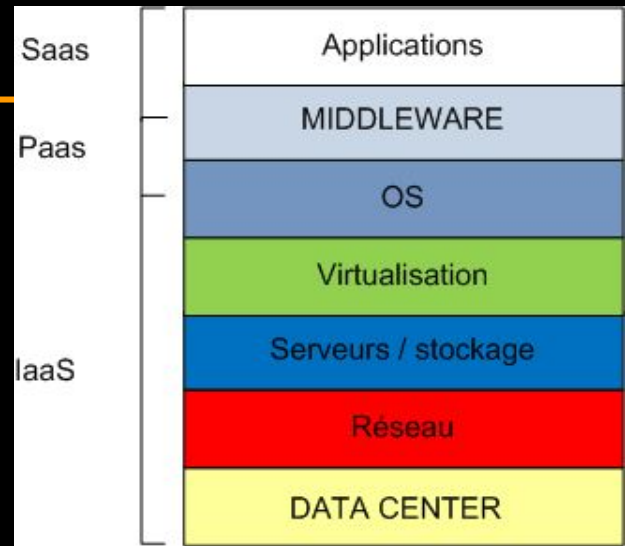
- Go 1.5-1.6
- Released in 2009 (1.0 since 2012)
- Rob Pike - Google

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



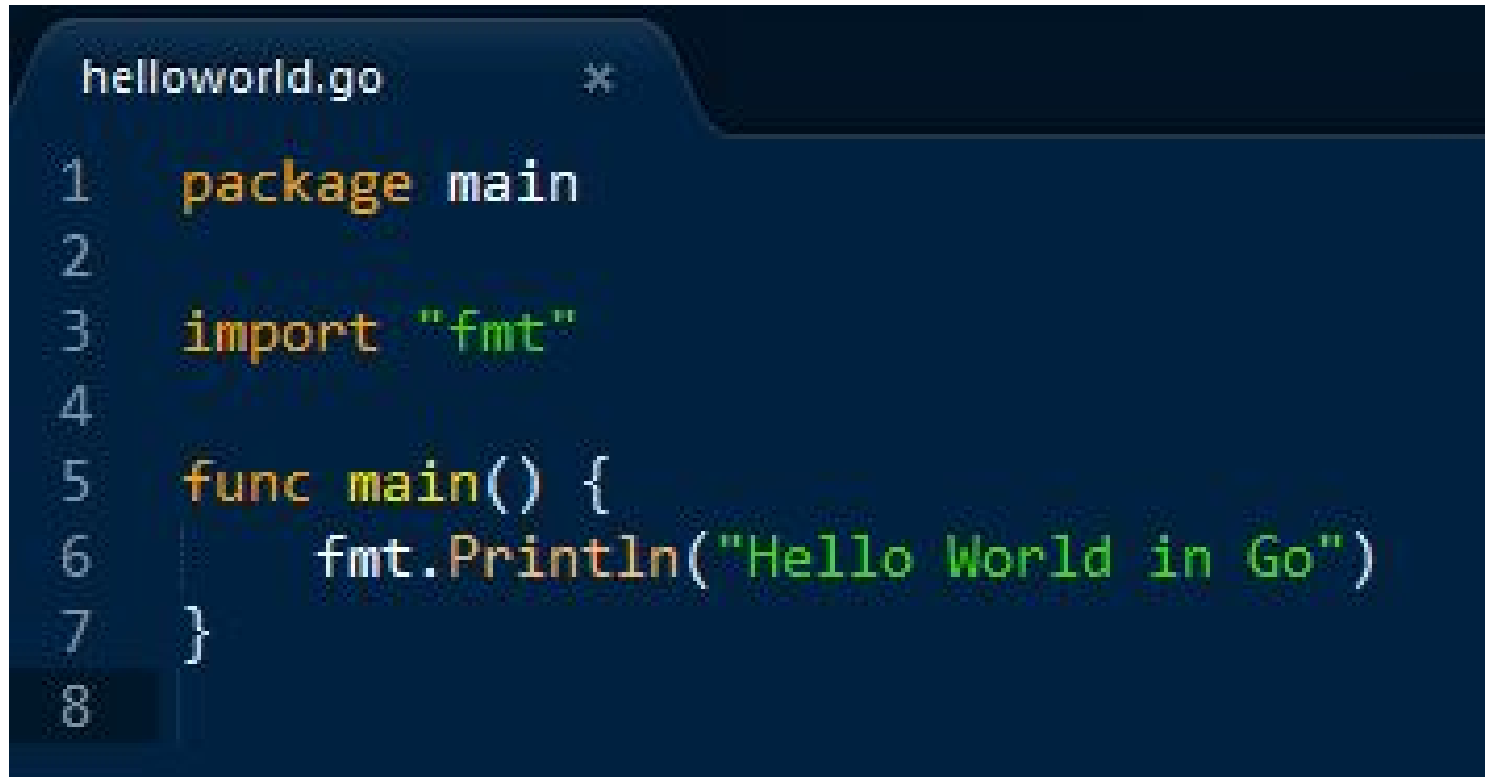
Simple

Less is more

Hello World in Go:

- `gofmt`
- `godoc / go test`
- `go get`

Gofmt



```
helloworld.go *  
1  package main  
2  
3  import "fmt"  
4  
5  func main() {  
6      fmt.Println("Hello World in Go")  
7  }  
8
```


Gofmt

```
helloworld.go
1  package main
2
3  import "fmt"
4
5  func
6      main( ) {
7      fmt.Println("Hello World in Go")
8      }
9
```

Godoc

The Go Programming Language X

← → ↻

https://golang.org

☆ ⚡ * »

The Go Programming Language

DocumentsPackagesThe ProjectHelpBlogSearch

Try Go

Pop-out ↗

```
// You can edit this code!  
// Click here and start typing.  
package main  
  
import "fmt"  
  
func main() {  
    fmt.Println("Hello, 世界")  
}
```

Hello, World! ▾

RunShareTour

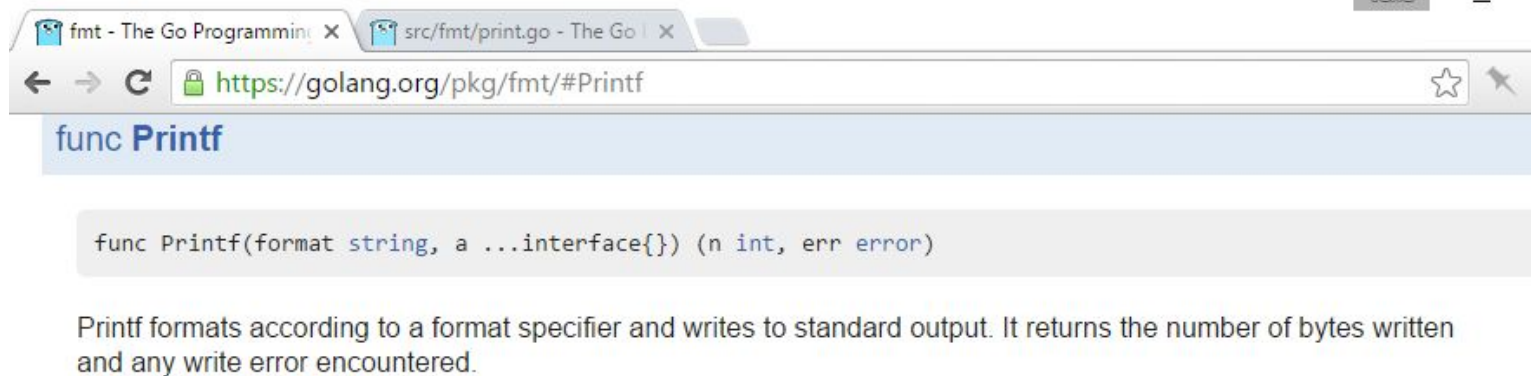
Go is an open source programming language that makes it easy to build simple, reliable, and efficient software.



Download Go

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

Godoc

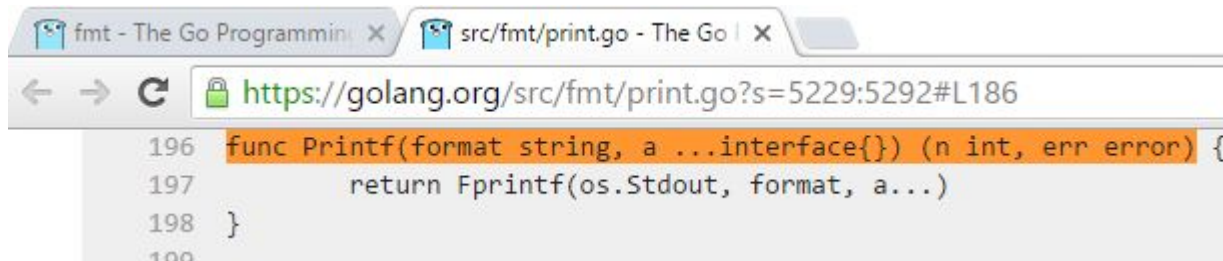


Godoc



```
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.



Goinstall

```
vonc@DESKTOP-EQ06E7V C:\Users\vonc\docker\godemo\src\simple\helloworld  
> cd ../../../../bin  
  
vonc@DESKTOP-EQ06E7V C:\Users\vonc\docker\godemo\bin  
> ls  
helloworld.exe  
  
vonc@DESKTOP-EQ06E7V 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



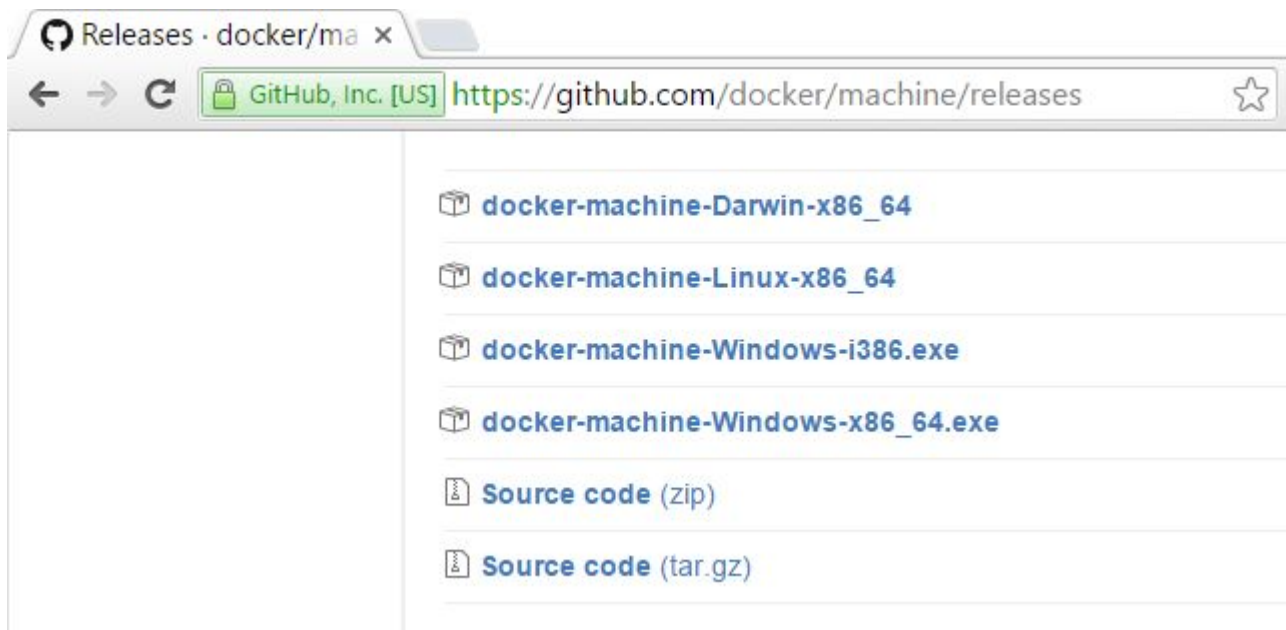
Simple

Less is more

Hello World in Docker:

- docker-machine
 - SCRATCH
 - docker run
-

docker-machine: simple



docker-machine: simple

```
vonc@DESKTOP-EQ06E7V C:\Users\vonc\docker\godemo\src\simple\helloworld
```

```
> dm version
```

```
docker-machine version 0.6.0, build e27fb87
```

```
vonc@DESKTOP-EQ06E7V C:\Users\vonc\docker\godemo\src\simple\helloworld
```

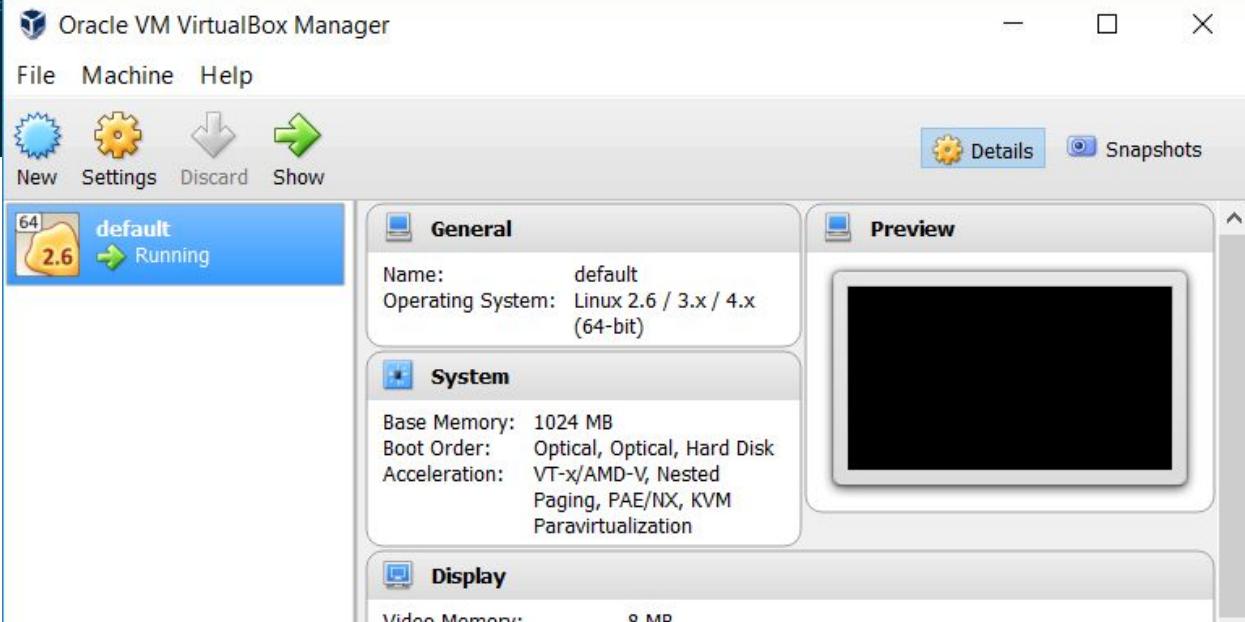
```
> dm ls
```

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-EQ06E7V C:\Users\vonc\docker\godemo\src\simple\helloworld  
> dm version  
docker-machine version 0.6.0, build e27fb87
```

```
vonc@DESKTOP-EQ06E7V  
> dm ls  
NAME      ACTIVE  
default   -
```



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-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\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\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"

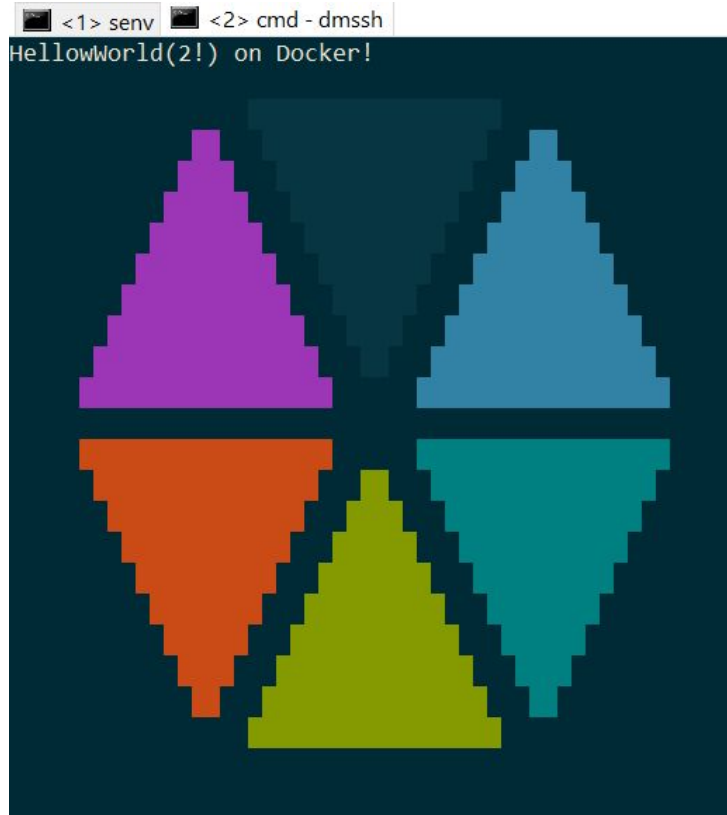
vonc@DESKTOP-EQ06E7V C:\Users\vonc\docker\godemo\src\simple\docker\scratch
> pause
Press any key to continue . . .

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\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



Unique

composition



Unique

composition

Hello World in Go (web):

- interface
 - goroutine
 - channel
-

Hello worldweb: 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.

Hello worldweb: interface

```
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

```
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

```
// 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)
}
```

Hello world web https: goroutine

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

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

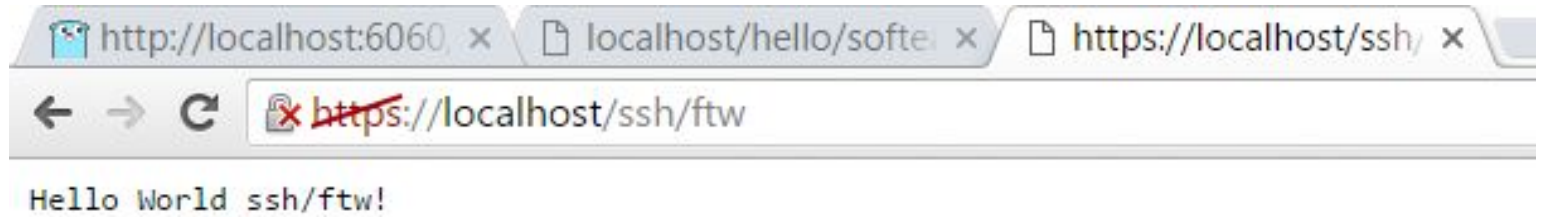
    openssl rsa -passin pass:${fqnpassword} -in "${passphrasekey}" -out "${key}"

    # 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
```


Hello world web https

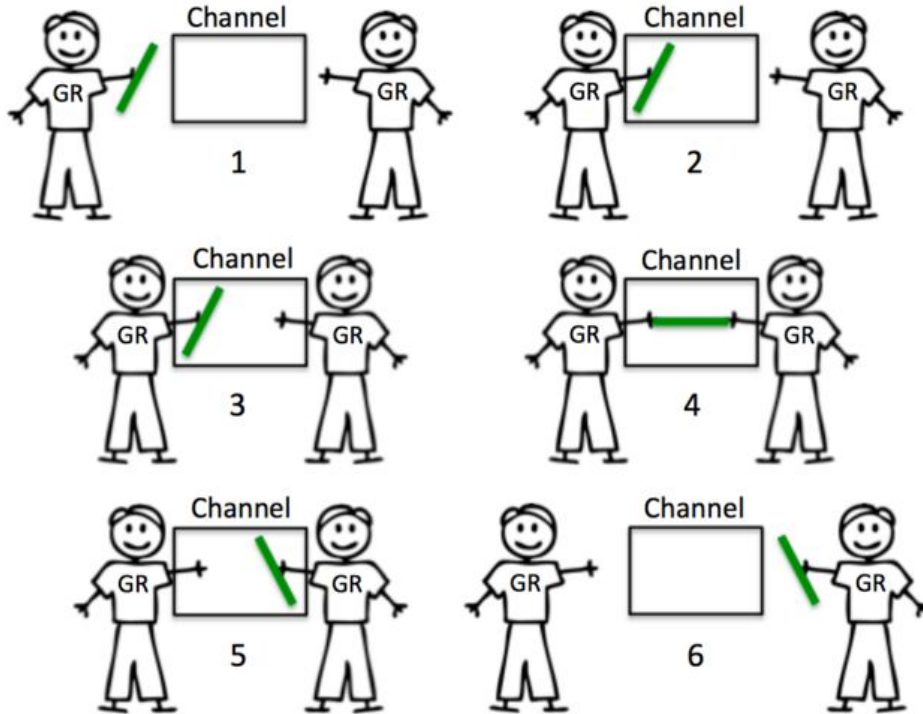


goroutine & channel

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

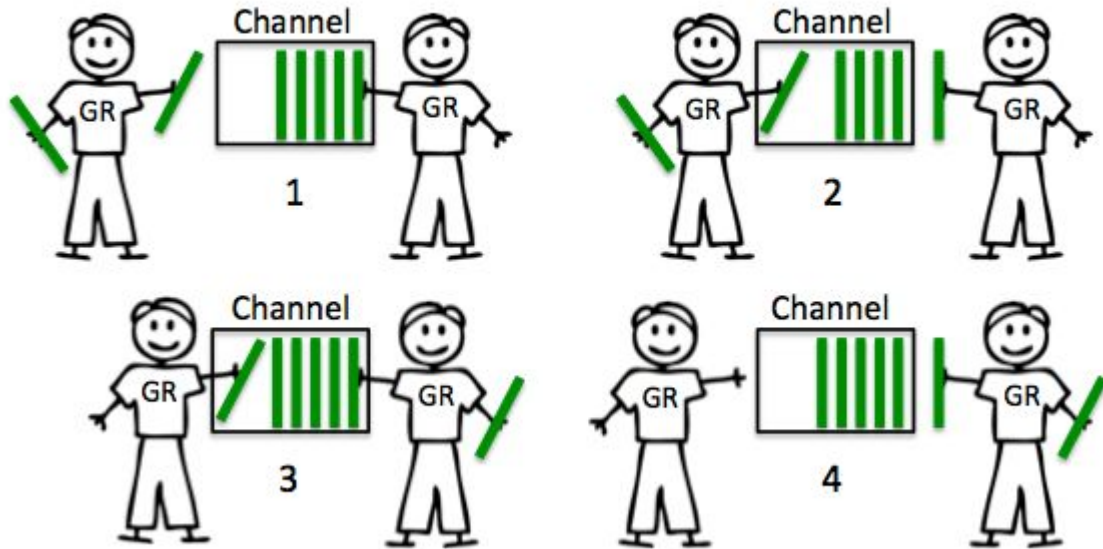
goroutine & channel

Unbuffered Channels



goroutine & channel

Buffered 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 .
```

```
docker@default:~$ curl http://localhost
```

```
curl: (7) Failed connect to localhost:80; Connection refused
```

```
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"]
```

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 .
```

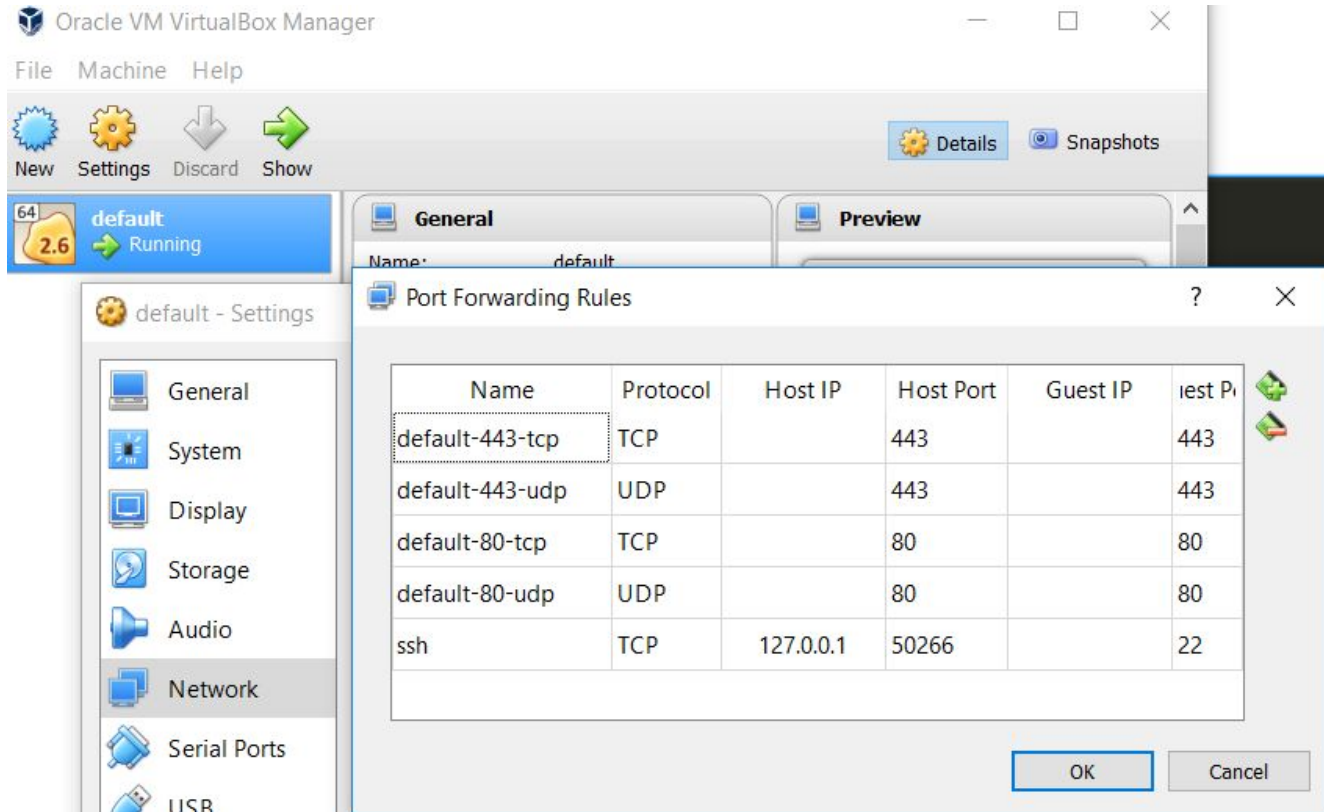
```
EXPOSE 80  
EXPOSE 443
```

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

```
ENTRYPOINT ["/helloworldwebhttps"]
```

```
CMD ["80", "443"]
```

Port-Forward



DevOps

isolation



DevOps

isolation

Hello World!

- Dev 2 Ops
- Ops 2 Dev
- Conclusion

DevOps

isolation

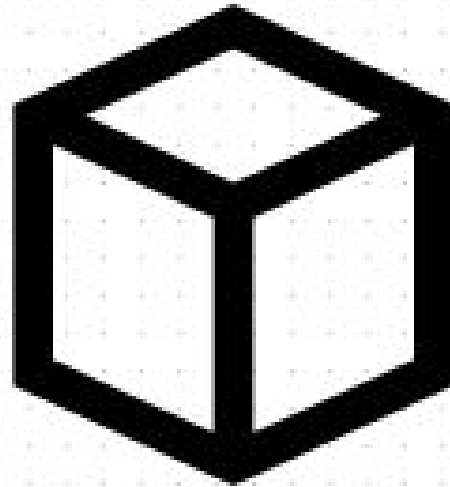
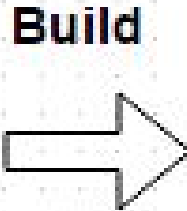
Hello World!

Dev2Ops

Dev2Ops: simple as build?

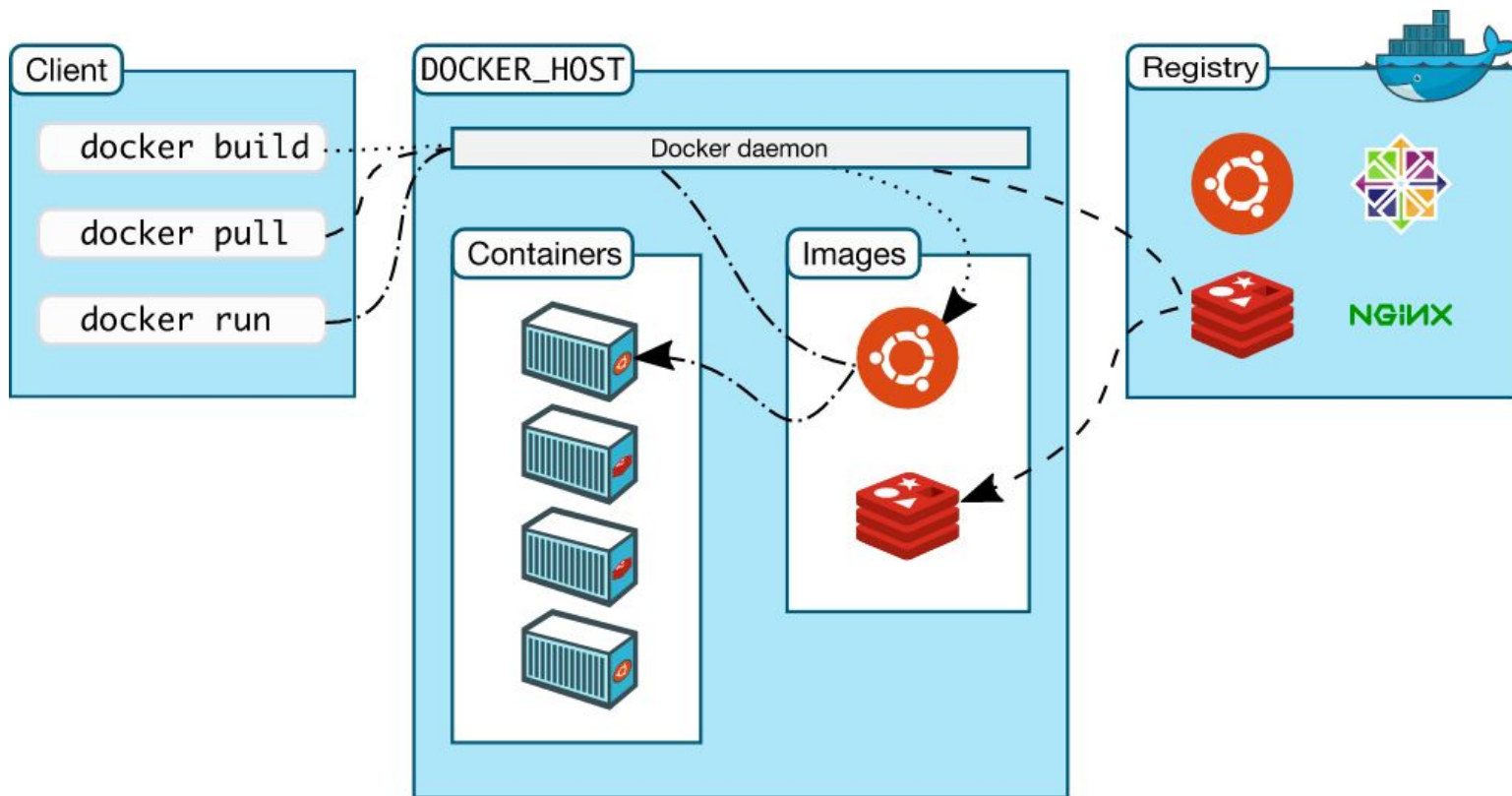


Dockerfile

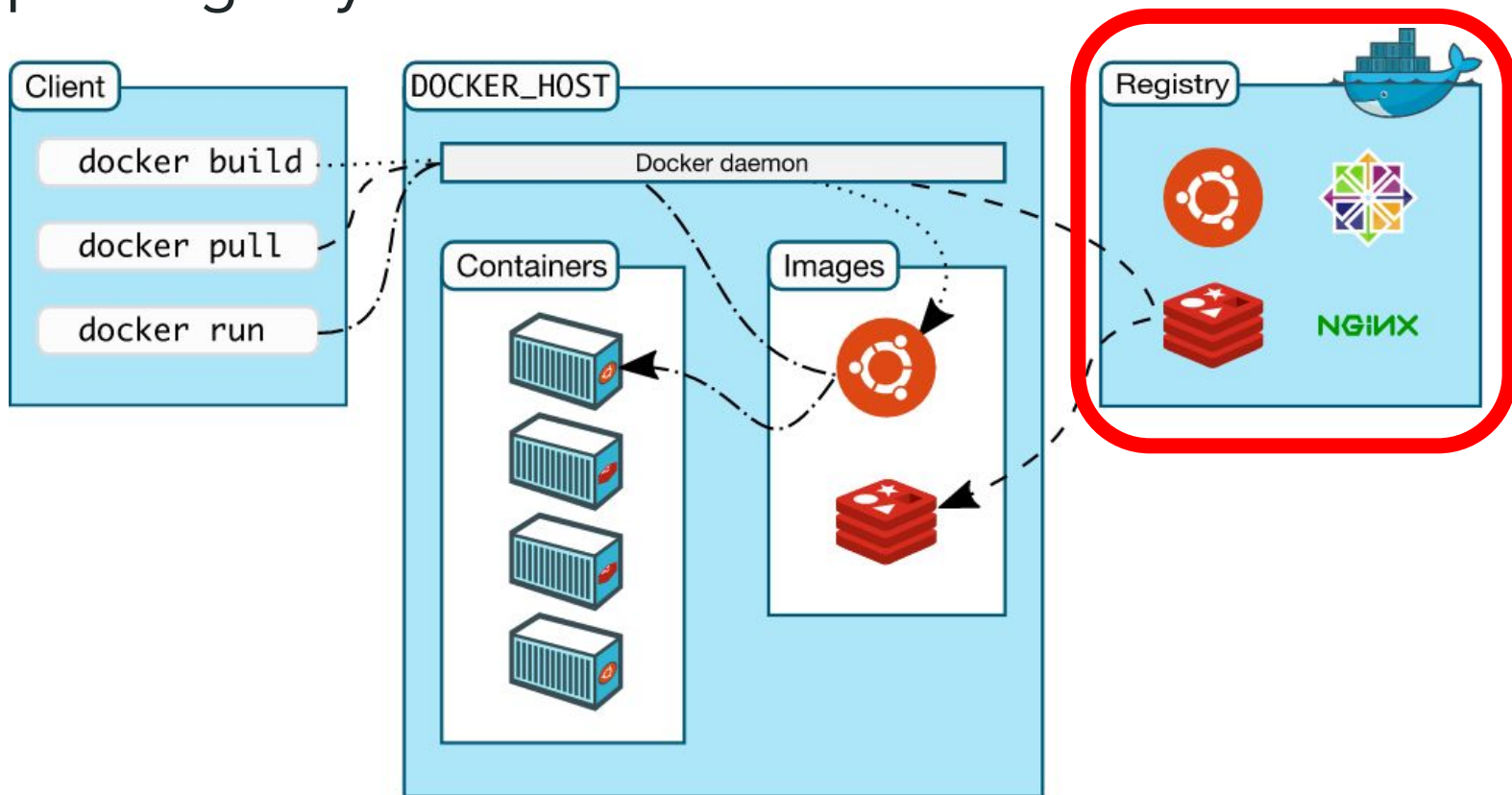


Image

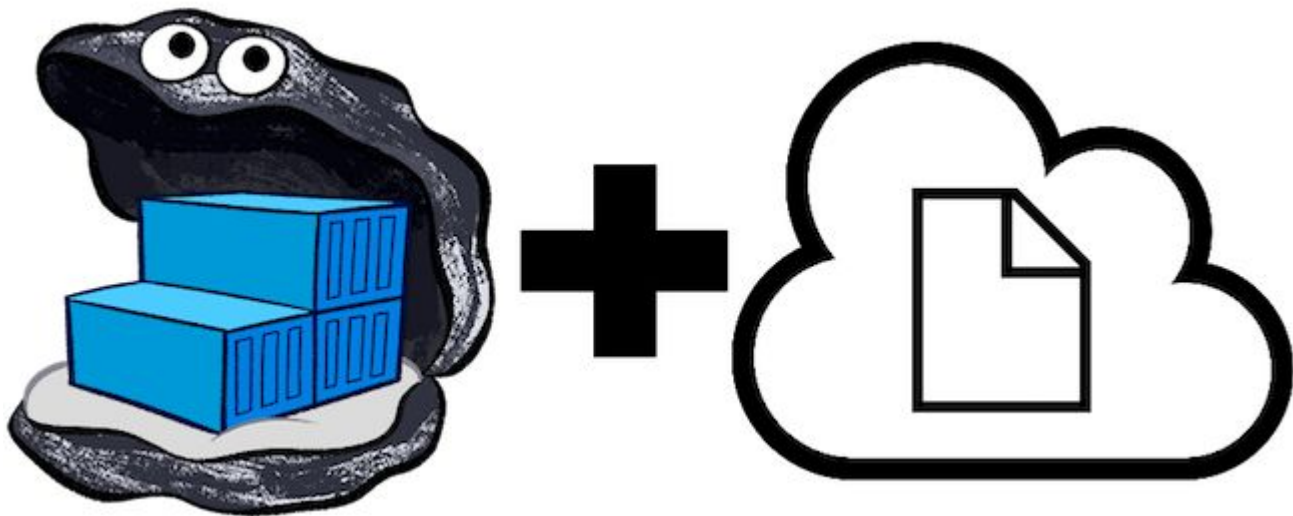
Dev2Ops: Registry...



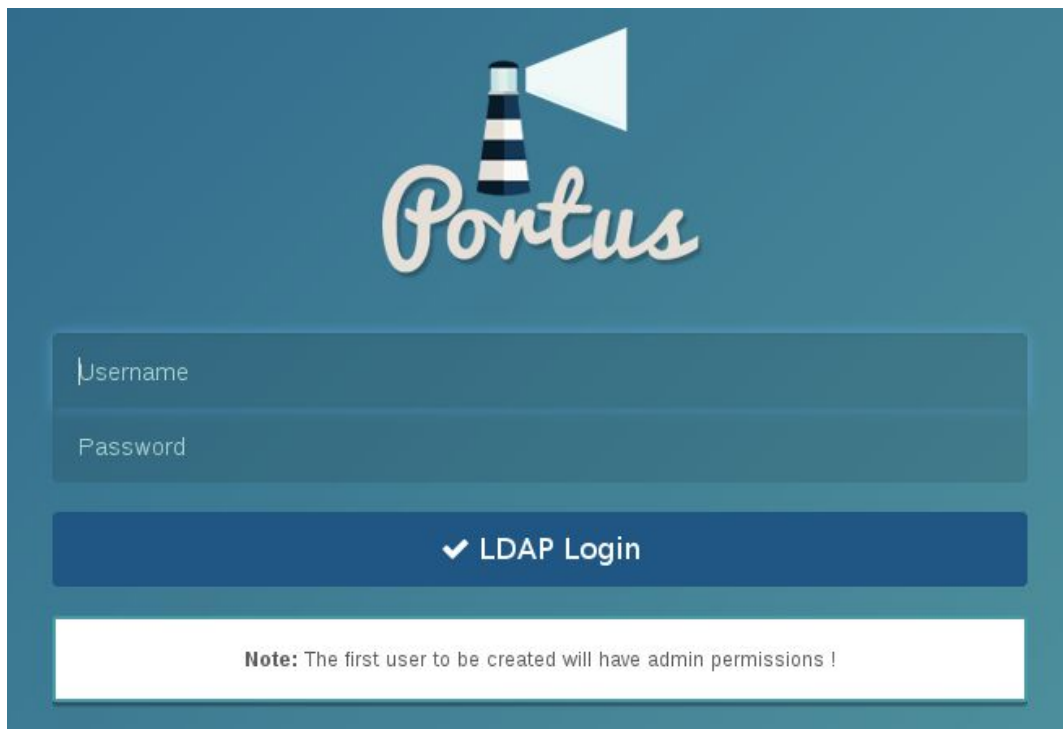
Dev2Ops: Registry...



Dev2Ops: Registry...



Dev2Ops: Portus!?



The image shows the Portus login interface. At the top, there is a logo featuring a lighthouse with a striped tower and a beam of light, with the word "Portus" in a white, cursive font below it. Below the logo, there are two input fields: "Username" and "Password". Below these fields is a blue button with a white checkmark and the text "LDAP Login". At the bottom, there is a white box with a blue border containing a note: "Note: The first user to be created will have admin permissions !".

Portus

Username

Password

✓ LDAP Login

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

<http://port.us.org/>

DevOps

isolation

Hello World!

Ops2Dev

Ops2Dev: X11 server



Ops2Dev: X11 server

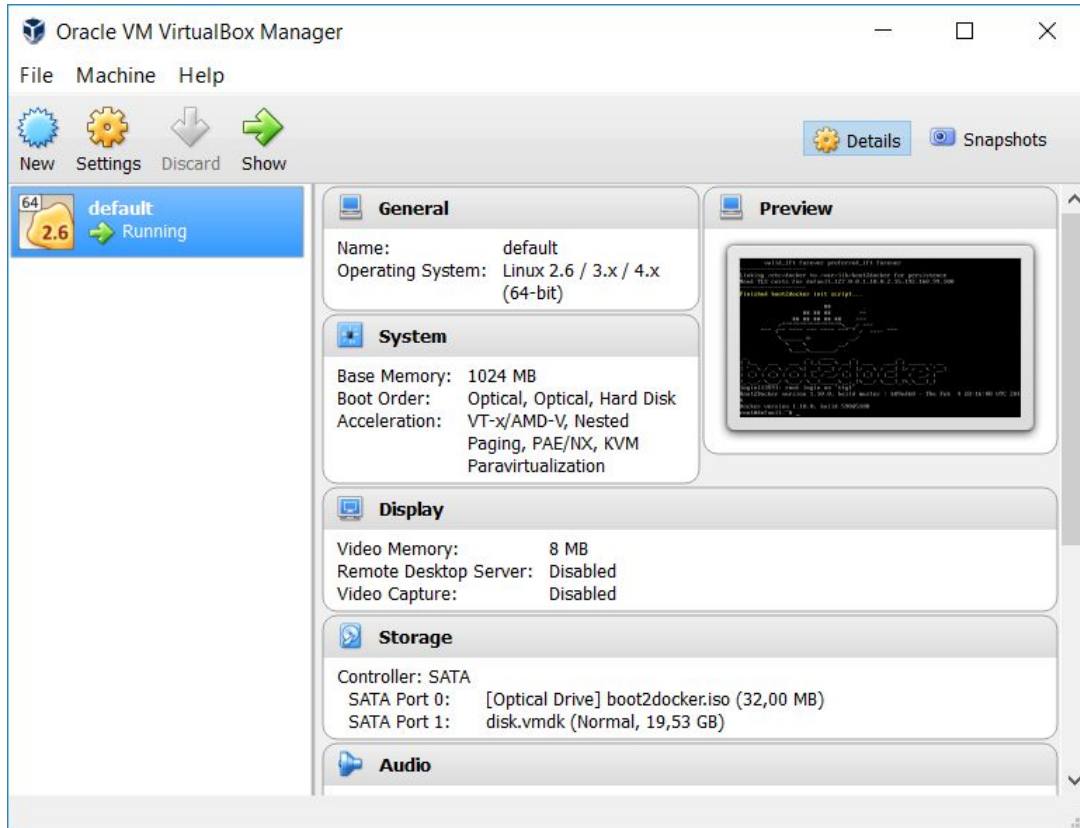
```
FROM debian
RUN apt-get update
RUN apt-get install -qqy x11-apps
ENV DISPLAY :0
CMD xeyes
```

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...



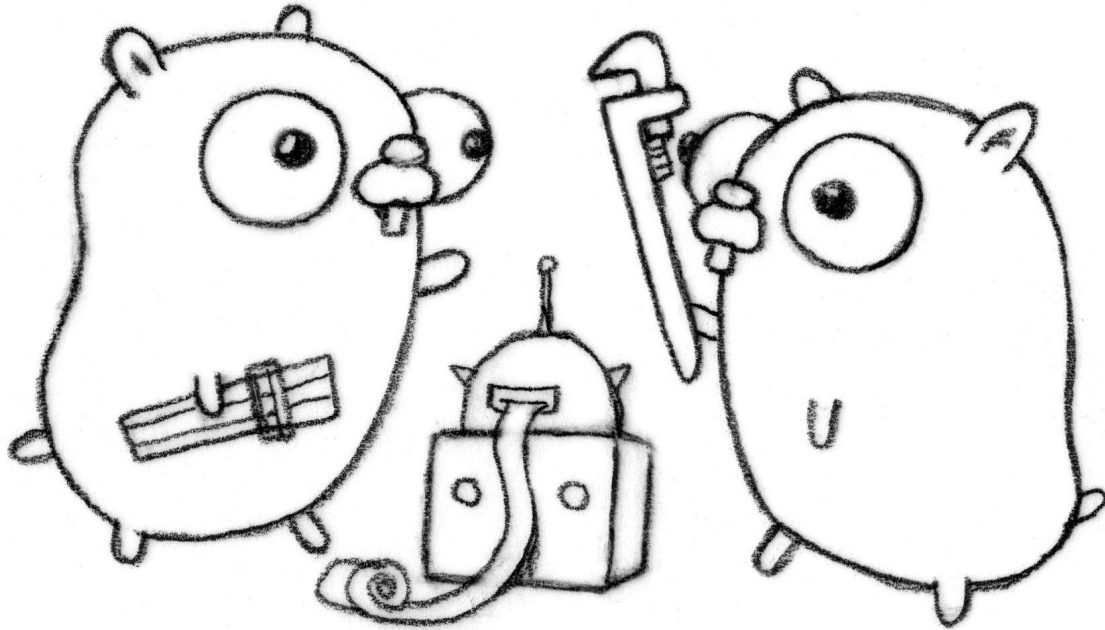
DevOps

isolation

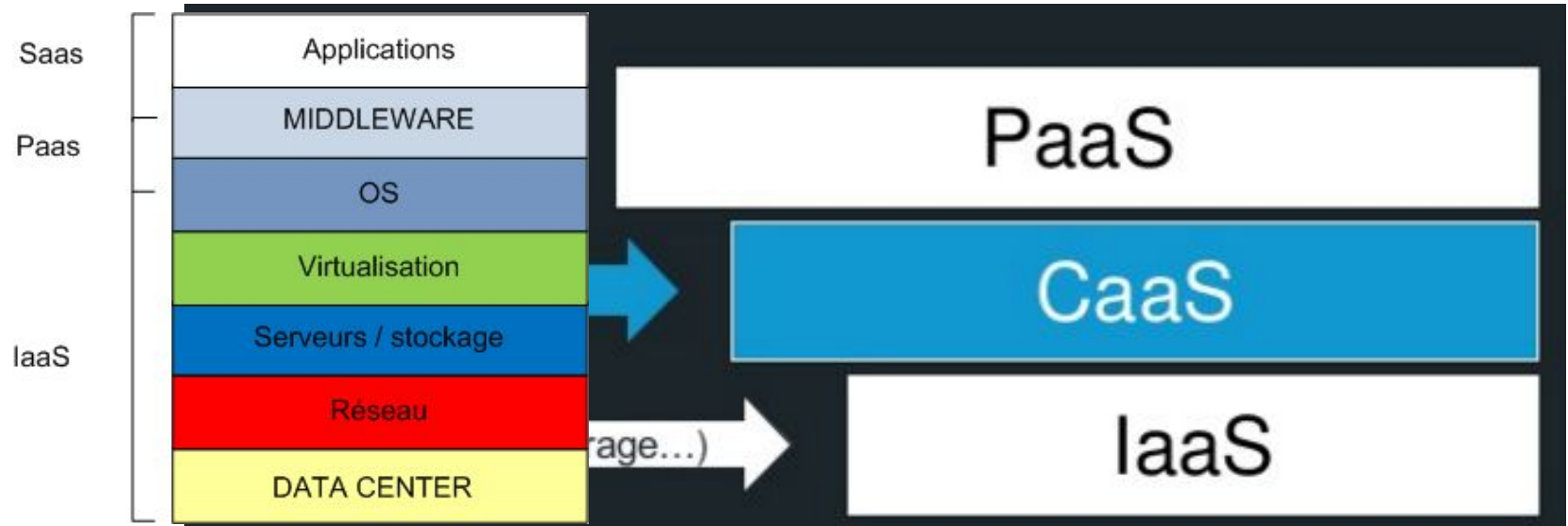
Conclusion

Brace yourself:
go & docker are
coming!

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

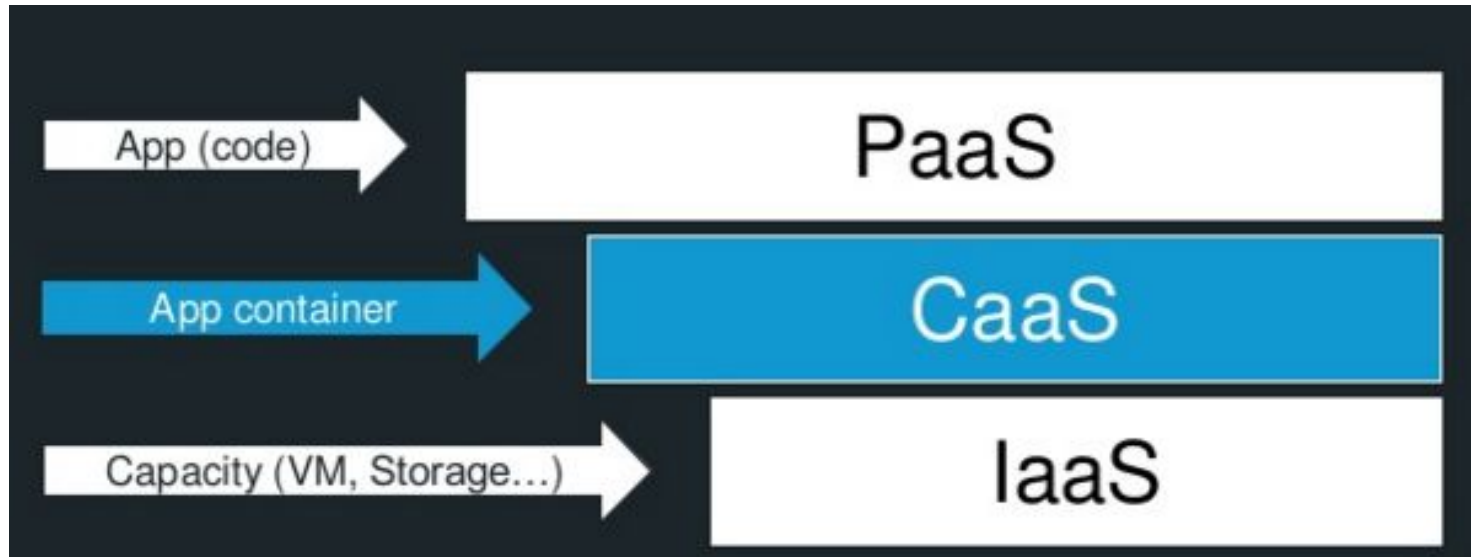


Conclusion: docker (CaaS)



<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)

