



Hello !



- I'm Dan !
- Senior Field Engineer at Heptio VMware
- Ex:
 - Heptio
 - Docker
 - Hewlett-Packard Enterprise
 - SkyBet
 - European Space Agency
 - ...
- Still a maintainer and contributor to various Docker and Moby projects (mainly GO, but I sometimes write C (if need be))
- @thebsdbox – for random nonsense



Where did Docker come from?

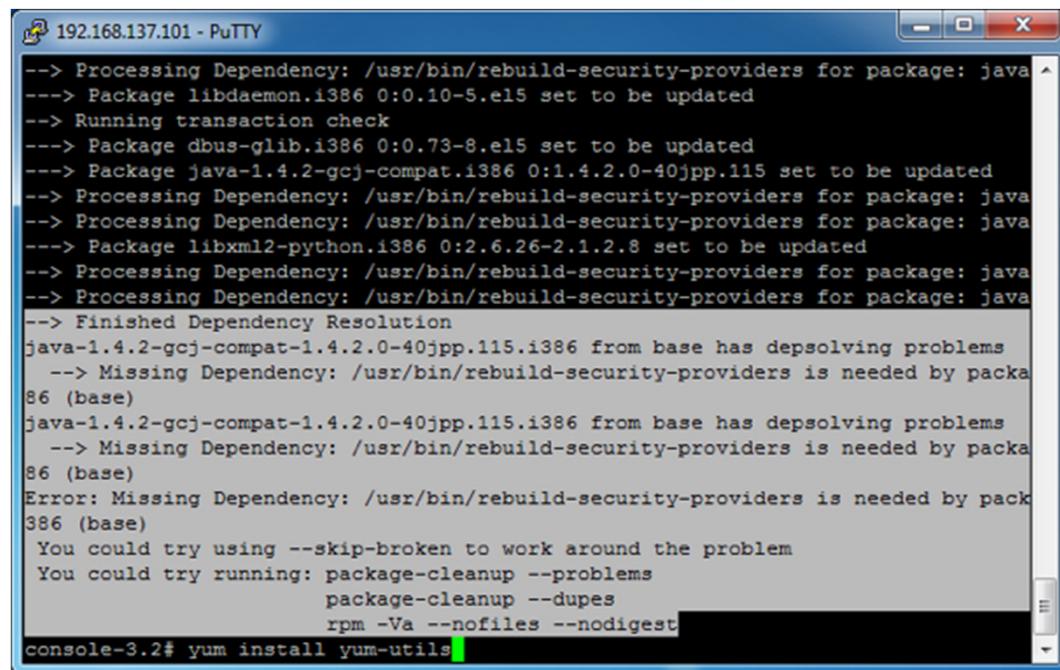


In the beginning ...



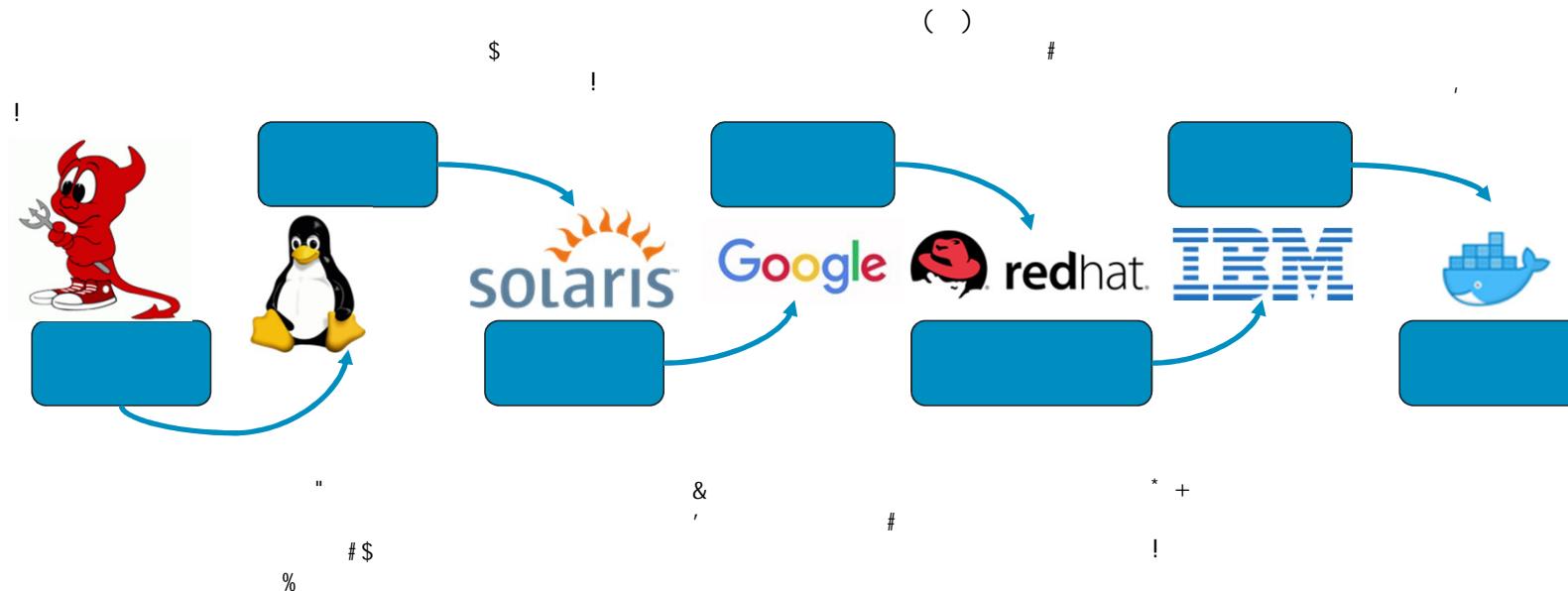
Where did Docker come from?

Things broke a lot . . .

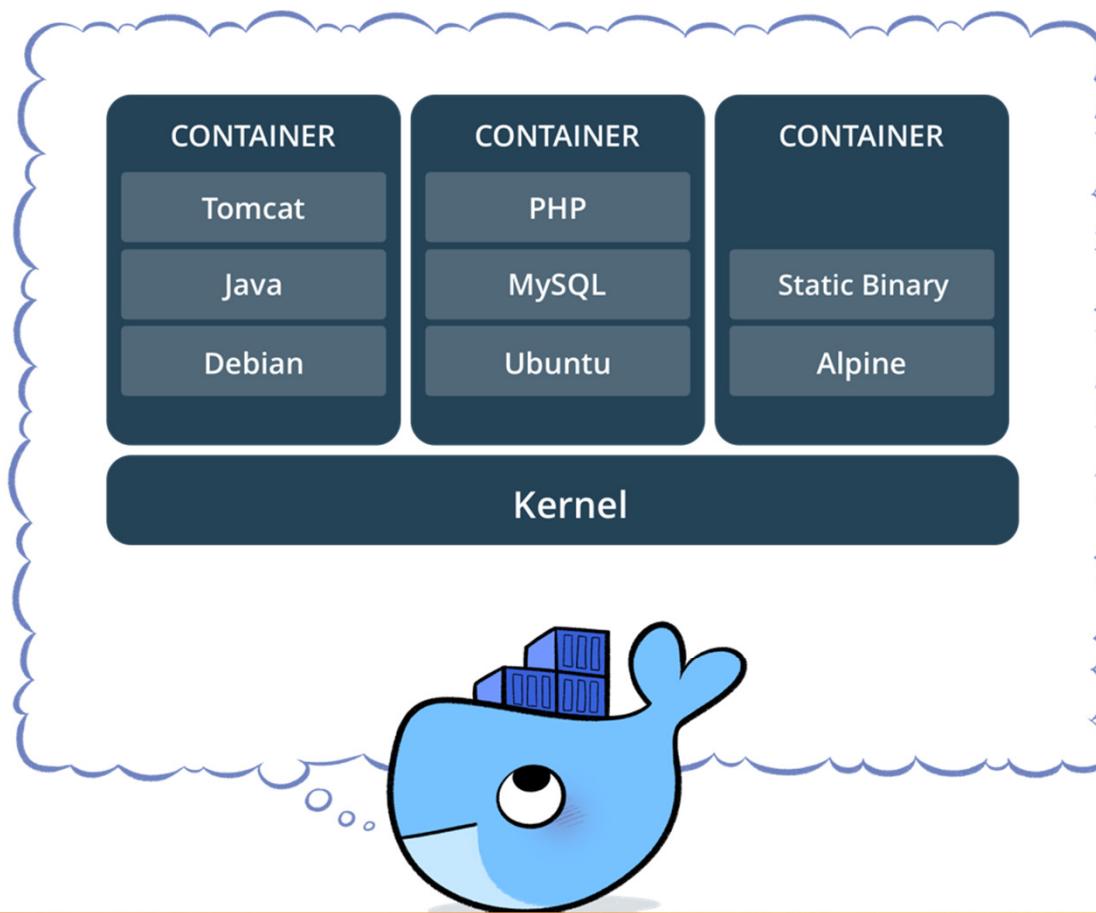


```
192.168.137.101 - PuTTY
--> Processing Dependency: /usr/bin/rebuild-security-providers for package: java
---> Package libdaemon.i386 0:0.10-5.el5 set to be updated
--> Running transaction check
---> Package dbus-glib.i386 0:0.73-8.el5 set to be updated
---> Package java-1.4.2-gcj-compat.i386 0:1.4.2.0-40jpp.115 set to be updated
--> Processing Dependency: /usr/bin/rebuild-security-providers for package: java
--> Processing Dependency: /usr/bin/rebuild-security-providers for package: java
---> Package libxml2-python.i386 0:2.6.26-2.1.2.8 set to be updated
---> Processing Dependency: /usr/bin/rebuild-security-providers for package: java
--> Processing Dependency: /usr/bin/rebuild-security-providers for package: java
--> Finished Dependency Resolution
java-1.4.2-gcj-compat-1.4.2.0-40jpp.115.i386 from base has depsolving problems
  --> Missing Dependency: /usr/bin/rebuild-security-providers is needed by package
     86 (base)
java-1.4.2-gcj-compat-1.4.2.0-40jpp.115.i386 from base has depsolving problems
  --> Missing Dependency: /usr/bin/rebuild-security-providers is needed by package
     86 (base)
Error: Missing Dependency: /usr/bin/rebuild-security-providers is needed by package
     386 (base)
  You could try using --skip-broken to work around the problem
  You could try running: package-cleanup --problems
                        package-cleanup --dupes
                        rpm -Va --nofiles --nodigest
console-3.2# yum install yum-utils
```

Where did Docker come from?



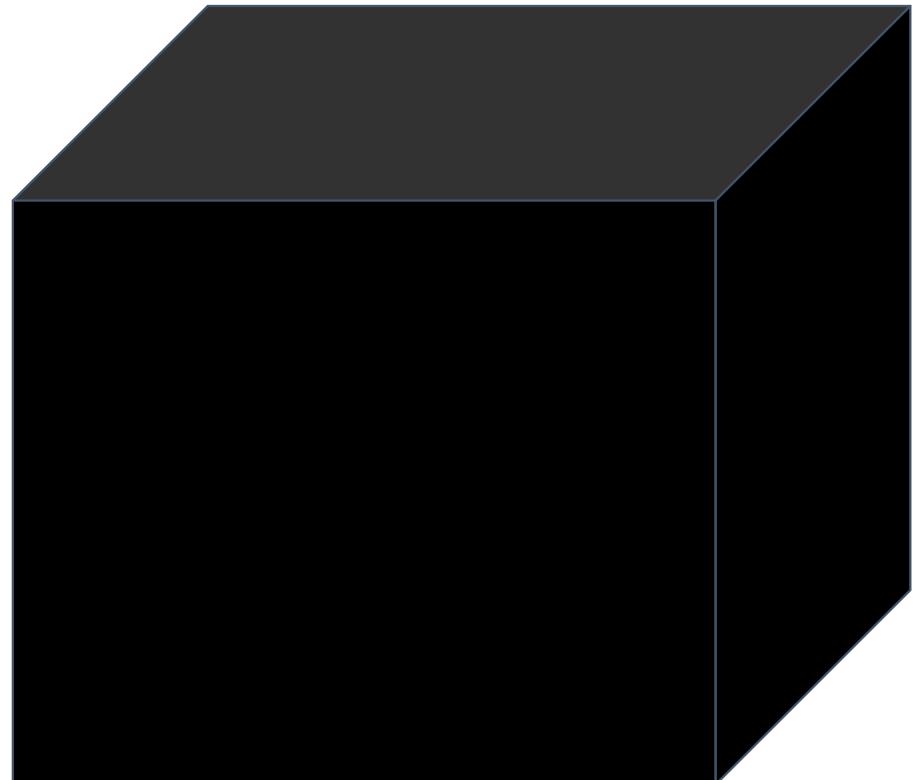
Docker under the hood



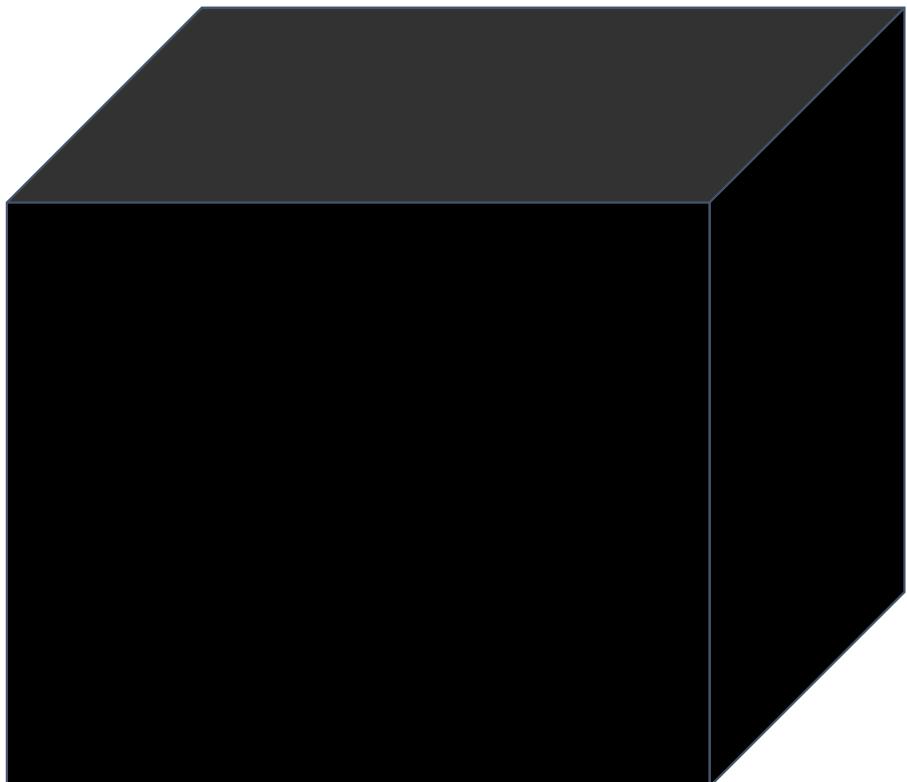
Docker Installation



- ! "
- apt-get install -y docker
- yum install -y docker
- #
-
- \$% & ' %\$(& ' %\$)



Docker Components



Docker CLI

dockerd

containerd

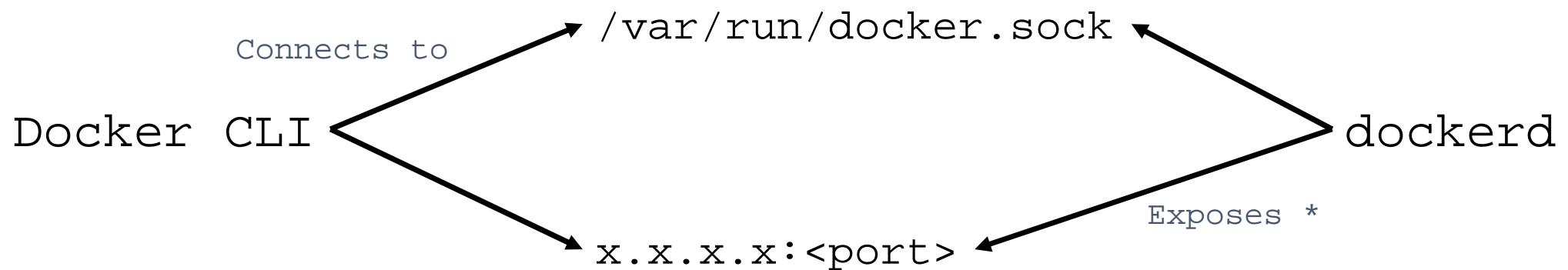
runc



Docker Command Line Interface

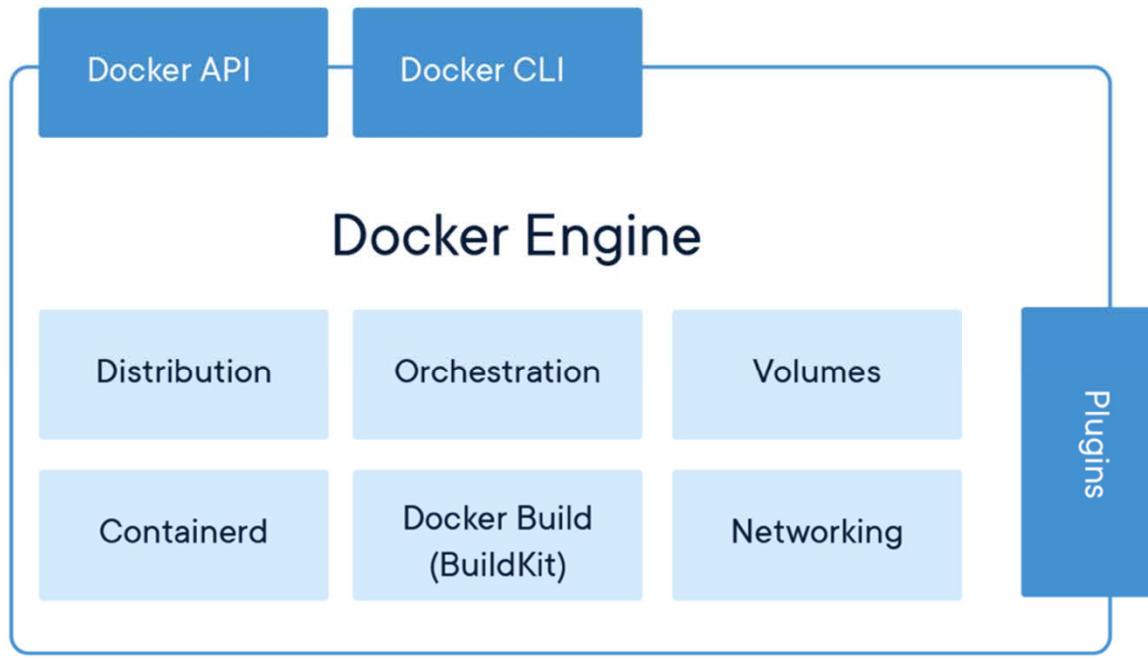
- docker build \$!
- docker run
- docker logs - !
- docker ps # -a
- docker images
- docker rm . docker rmi /
- docker tag
- docker login 0
- docker push/pull 1! (
- docker inspect ! 1 \$ \$2

Docker Command Line Interface



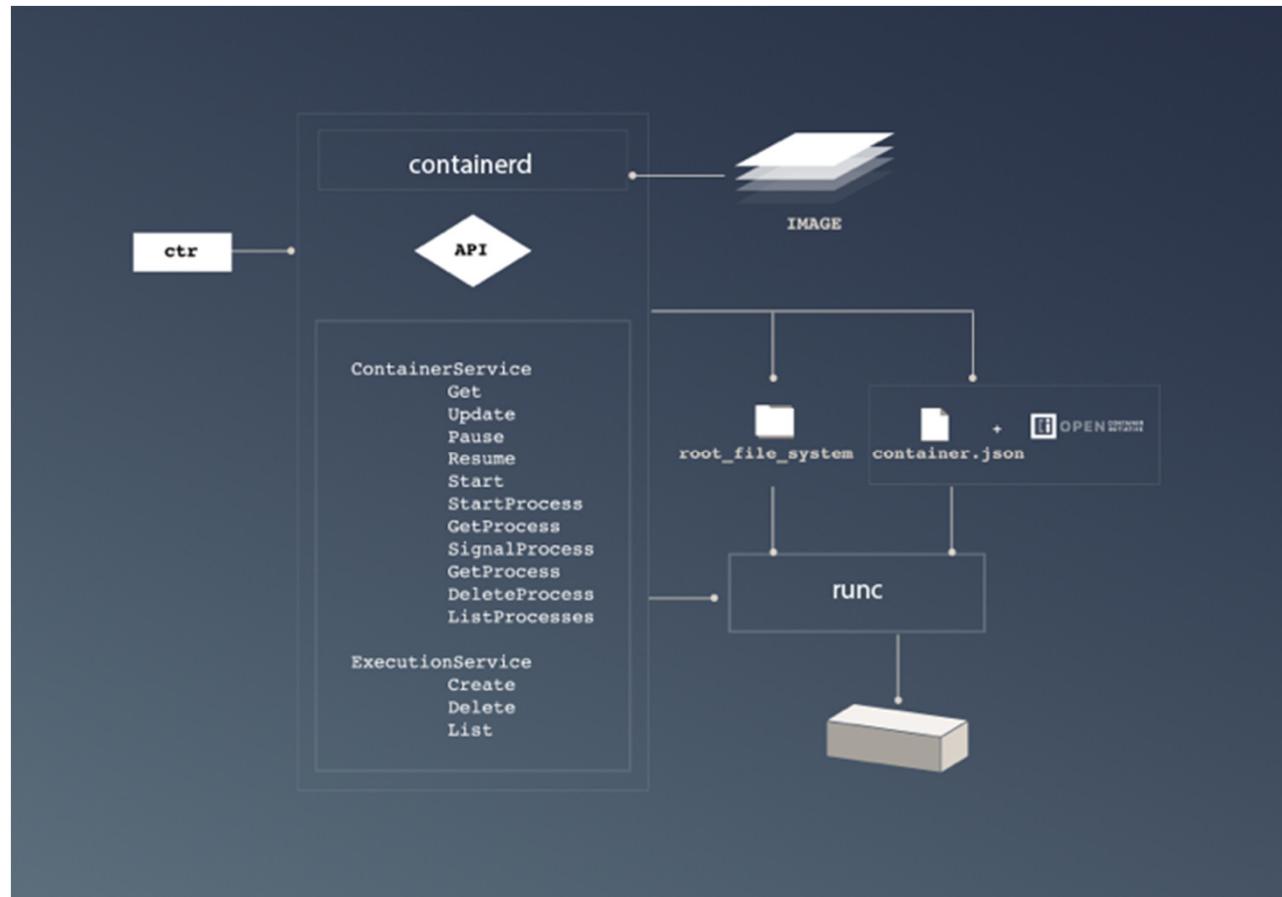
* Configured in `/etc/docker/json`

Docker Daemon



containerd

- + ! | 0 /
-
- * ! 1
- , ! 5 *
- -
-

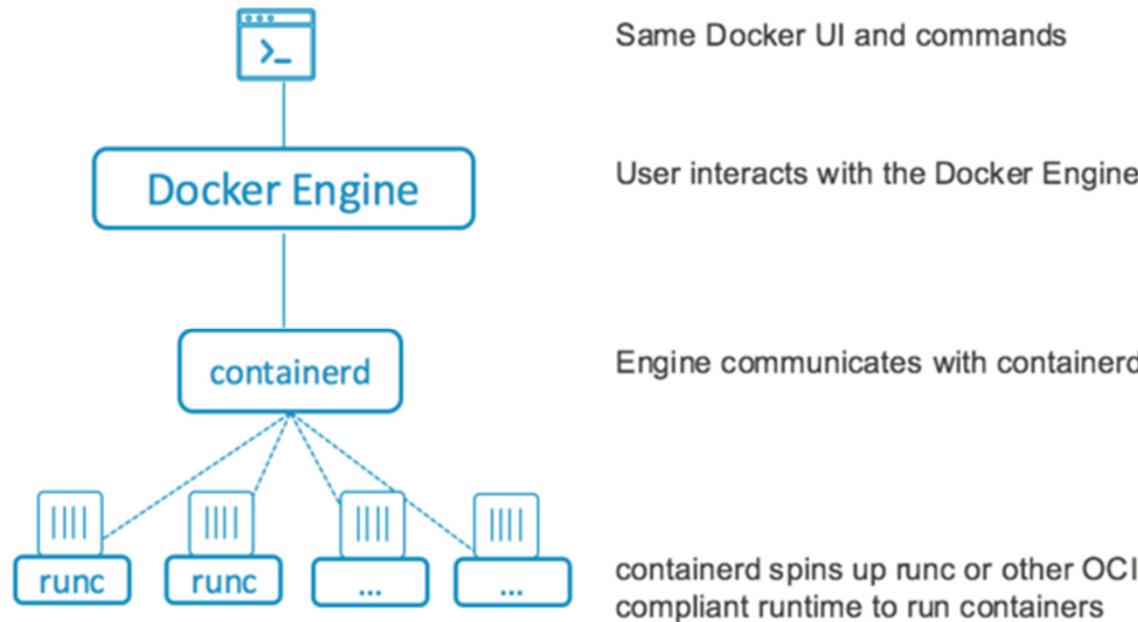


runc (or any OCI specific runtime)

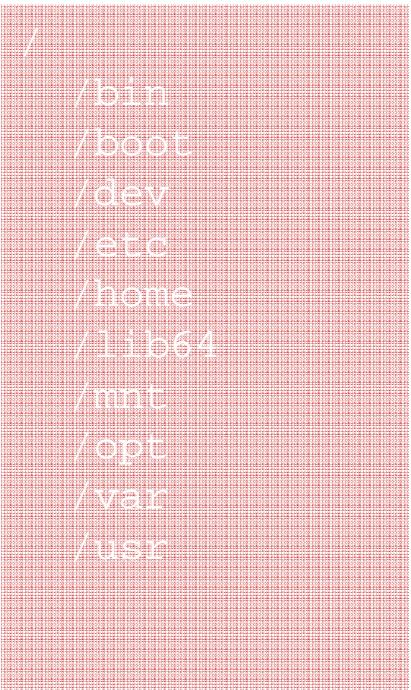
- -
- * 5 *
- \$!
- * 6 ! 3
-) #
- !



All tied together



Docker Image

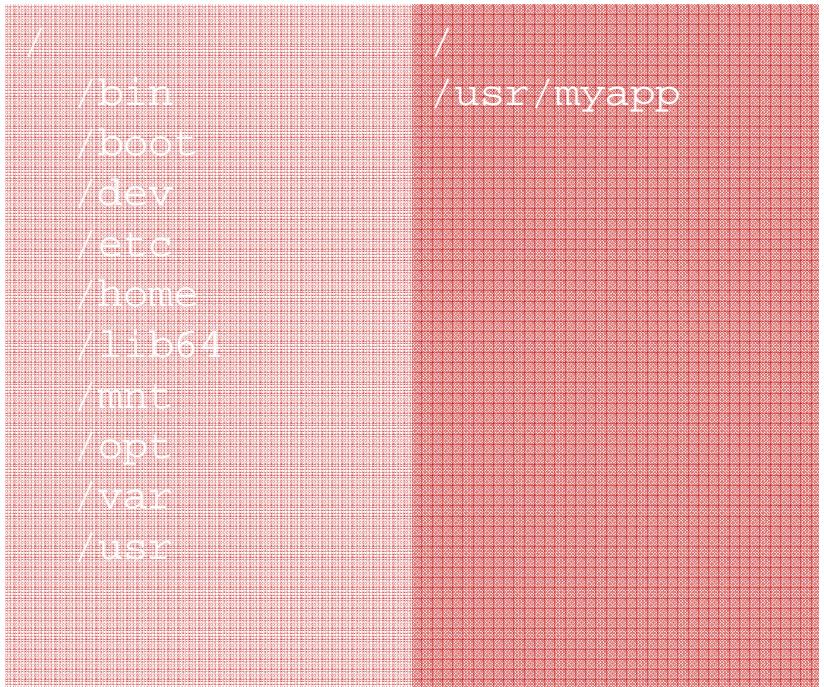


aabbcccddee



Manifest.json

Docker Image



aabbcccddee

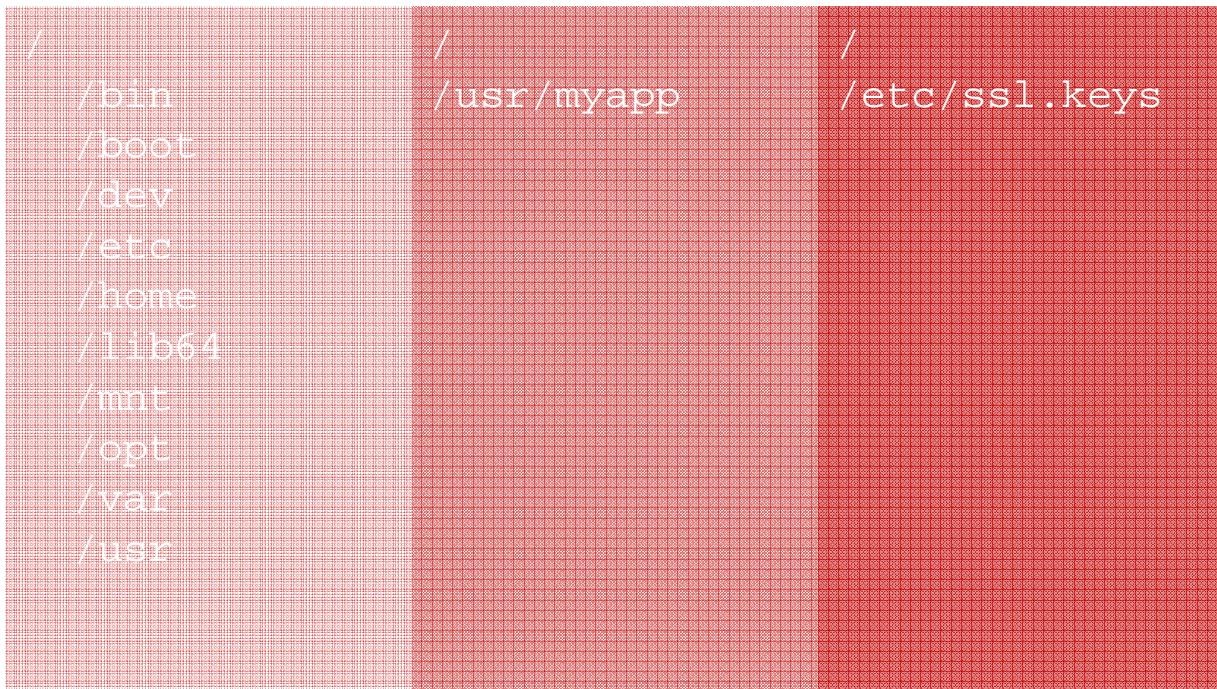
aabbcccddee



Manifest.json



Docker Image



aabbccdddee

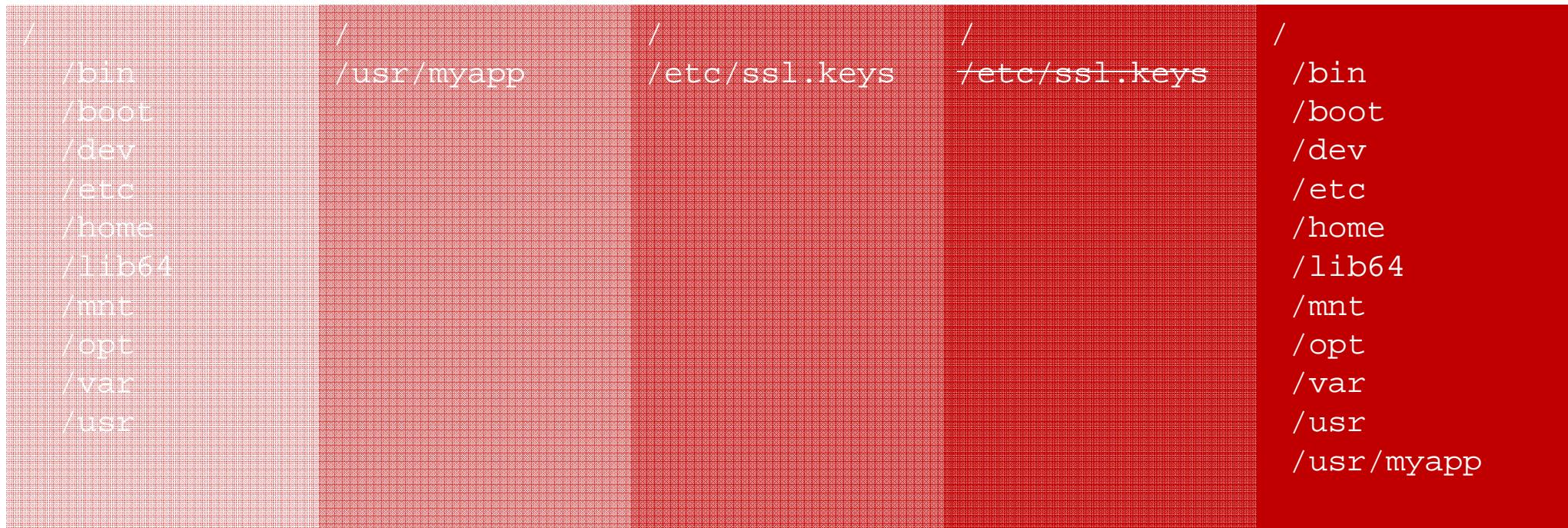
aabbccdddee

aabbccdddee

Manifest.json

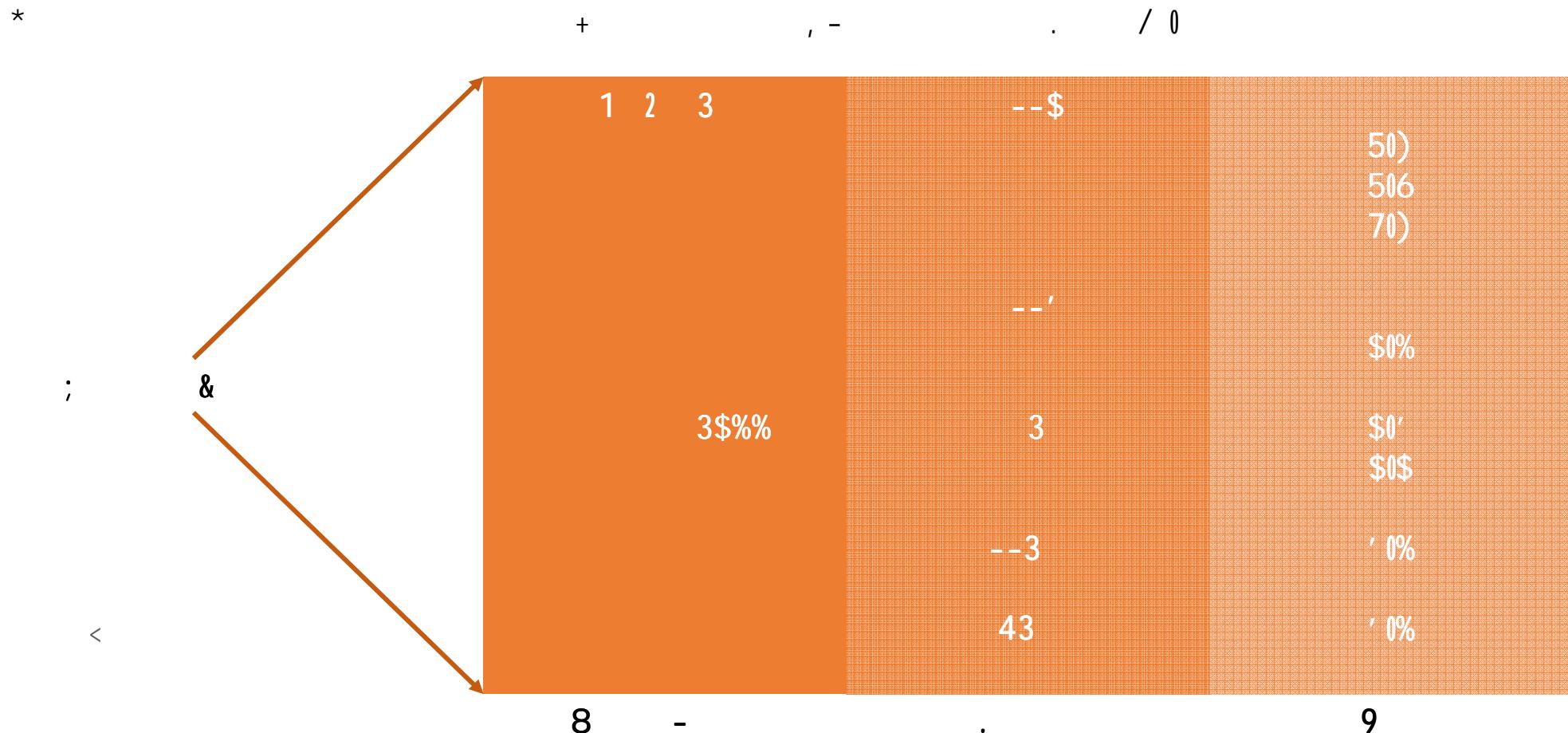


Docker Image



Manifest.json

Container Registry



Pulling an image from a Registry



```
*   / - - - 2 ! - " < - !/ "
```

The terminal window shows a series of characters and symbols: an asterisk (*), a forward slash (/), a minus sign (-), a dash (-), a number 2, an exclamation mark (!), another dash (-), a double quote ("), a less than symbol (<), another dash (-), an exclamation mark (!), a forward slash (/), and a double quote (").

What happens if no tag is specified?

What is a container?

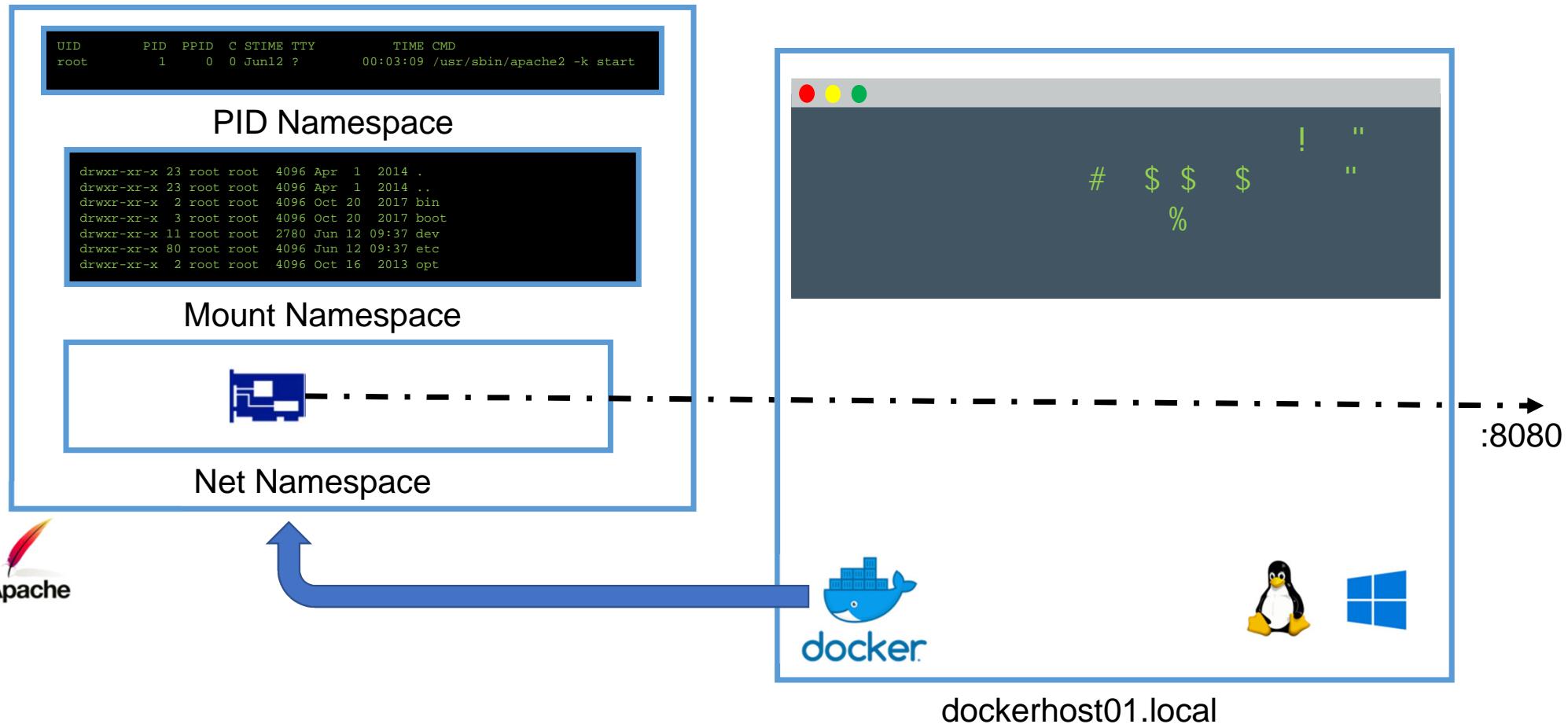
```
drwxr-xr-x 23 root root 4096 Apr  1 2014 .
drwxr-xr-x 23 root root 4096 Apr  1 2014 ..
drwxr-xr-x  2 root root 4096 Oct 20 2017 bin
drwxr-xr-x  3 root root 4096 Oct 20 2017 boot
drwxr-xr-x 11 root root 2780 Jun 12 09:37 dev
drwxr-xr-x 80 root root 4096 Jun 12 09:37 etc
drwxr-xr-x  4 root root 4096 Apr 21 2014 home
drwxr-xr-x 13 root root 4096 Apr 10 2014 lib
drwxr-xr-x  2 root root 4096 Oct 20 2017 lib64
drwx----- 2 root root 16384 Apr  1 2014 lost+found
drwxr-xr-x  2 root root 4096 Oct 16 2013 media
drwxr-xr-x  2 root root 4096 Sep 22 2013 mnt
drwxr-xr-x  2 root root 4096 Oct 16 2013 opt
```

Linux Namespace



dockerhost01.local

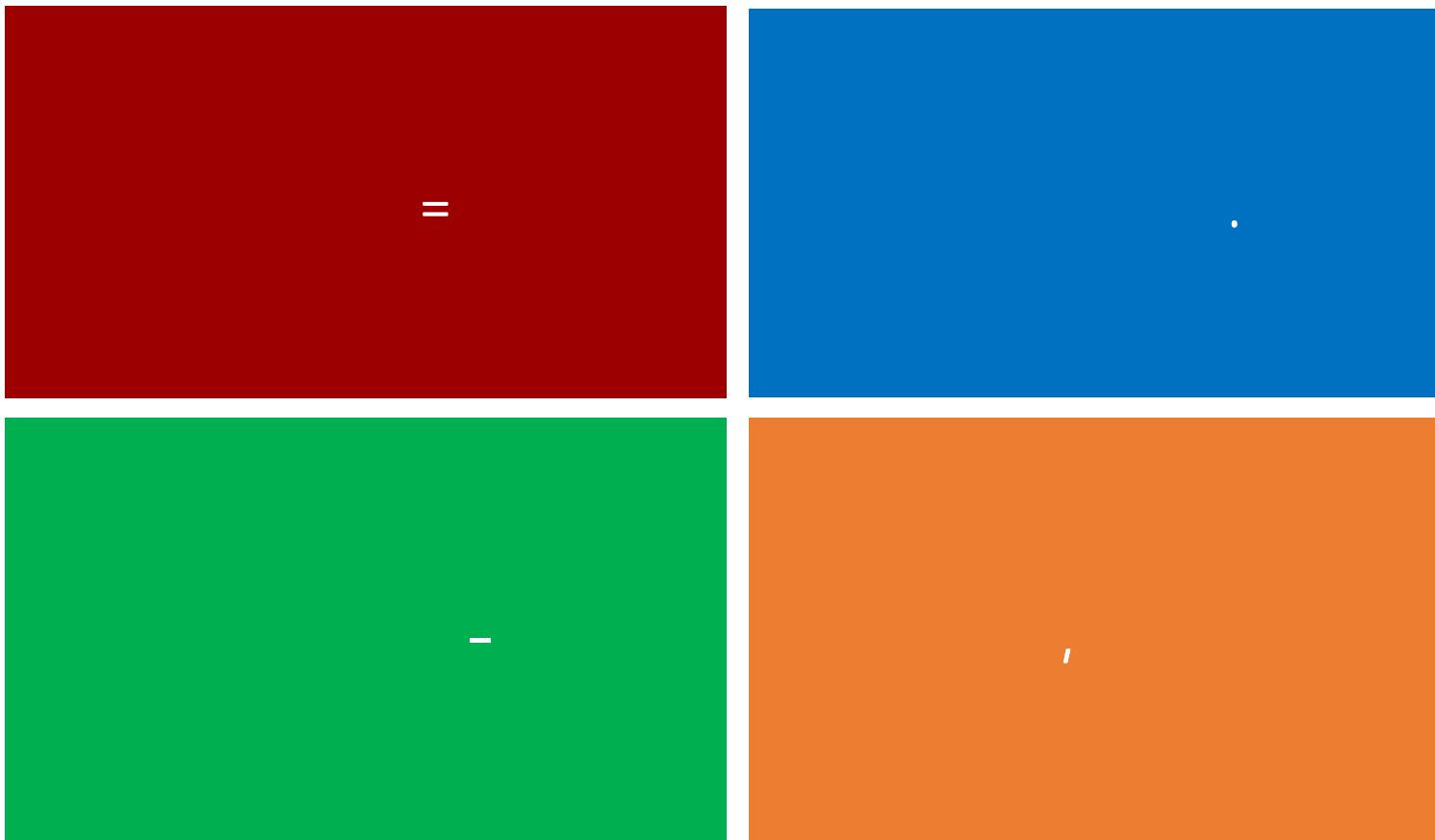
What is a container?



Using Docker



Using Docker



Dockerfile



=: , #

- > ;
0

9

!

0 "

A screenshot of a terminal window displaying a Dockerfile. The code is color-coded: red for operators like '&', '(', ')', '+', '#', and '1'; yellow for arithmetic operators like '*', '%', and '/'; green for numbers like '234', '5/6', '78', '4/6', '9', and '9#%'; and blue for comments like '/*' and '*/'. The terminal has a dark background with a light gray header bar containing three colored dots (red, yellow, green). The command 'docker build -t test .' is visible at the bottom of the terminal window.

```
&' () *      *      +      #      , -  
' . / *      *      %      %      0  
1(' 234' *      *      %      %      0  
' . / *      #  
5/6' 78(4/6  9      9  
:)3  9#%  9
```

Dockerfile



```
: 8
- ; ;
-
; < ; ;
;
; 0
- go get -
-
; <
```

A screenshot of a terminal window displaying a Dockerfile. The code is color-coded: red for operators like '&', yellow for symbols like '(', ')', and green for numbers like '8'. The background of the terminal is dark grey, and the text is white. The terminal window has three colored tabs at the top: red, yellow, and green.

```
&' () *      *      +      #      ,      -
' . / *      *      %      %      0
1(' 234' *      *      %      %      0
' . / *      #
5/6' 78(4/6 9      9
:)3 9##% 9
```

Dockerfile

A screenshot of a terminal window displaying a Dockerfile. The code is color-coded: red for punctuation like '&', '(', ')', and '#'; yellow for operators like '+', '*', '%', and '/'; green for numbers like '1', '2', '3', '4', '5', '6', '7', '8', '9', and '0'; and blue for reserved words like 'FROM'. The background of the terminal is dark grey, and the status bar at the top shows three colored dots (red, yellow, green).

```
& () * * + # , -  
. / * * % % 0  
1( 234 * * % % 0  
. / * #  
5/6 78(4/6 9 9  
:)3 9##% 9
```

Dockerfile



```
?89: @1, .89 -
```

A screenshot of a terminal window displaying a Dockerfile. The code is color-coded: red for punctuation like '&', yellow for operators like '+', and green for symbols like '%'. The terminal has a dark background with a light gray header bar containing three colored dots (red, yellow, green).

```
&' () *      *      +    #      , -
' . / *      *      %      %      0
1(' 234' *      *      %      %      0
' . / *      #
5/6' 78(4/6  9      9
:)3  9##%  9
```

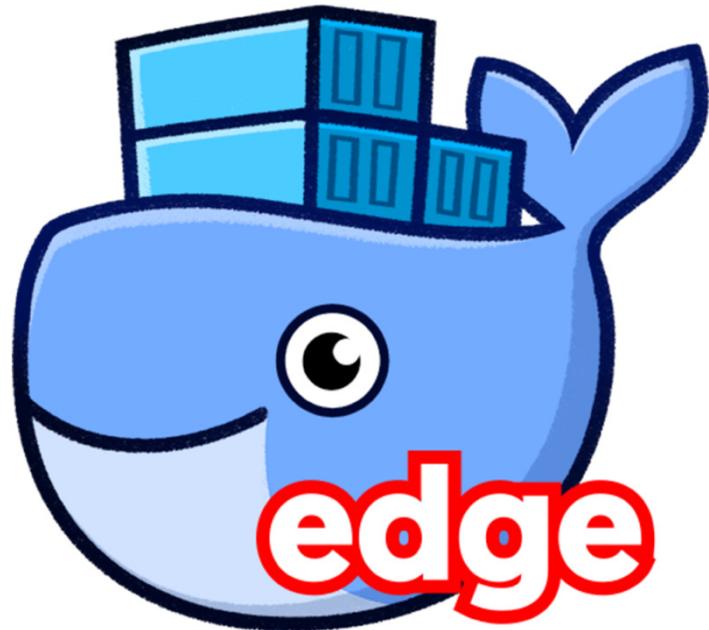
Efficient Dockerfile(s)



```
● ● ●  
& () * * + # , - ; <  
' ./ * * % % 0  
1('234' * * % % 0  
' ./ * #  
  
= 6% * *  
& () %  
:(87 ##> ?  
  
5/6'78(4/6 9  
:)3 9##% 9
```

```
# 1 ;  
- 0  
- & -  
-- <  
; / 0
```

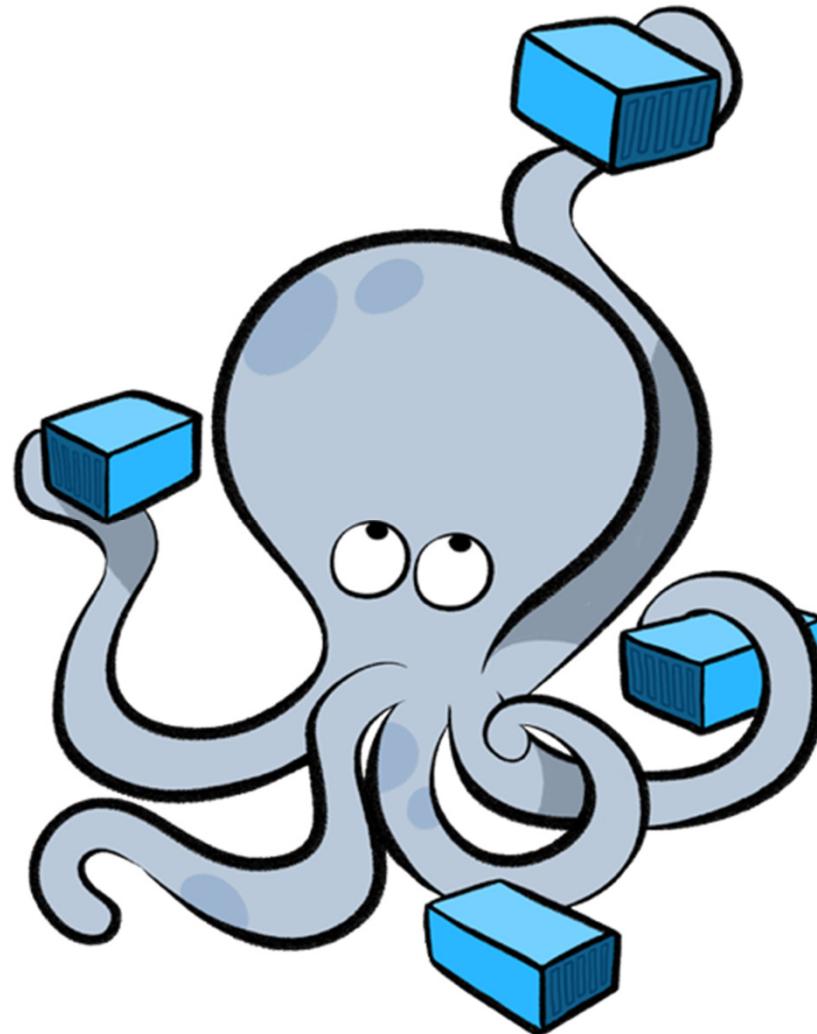
Docker CLI



9 . - < A
0
B / . < ; -- !
/ - ; ; "

Docker Compose

```
- - - - /  
- ; <  
- ; -- 0  
- - /yaml  
- - -- <  
/ 0
```



Docker Compose



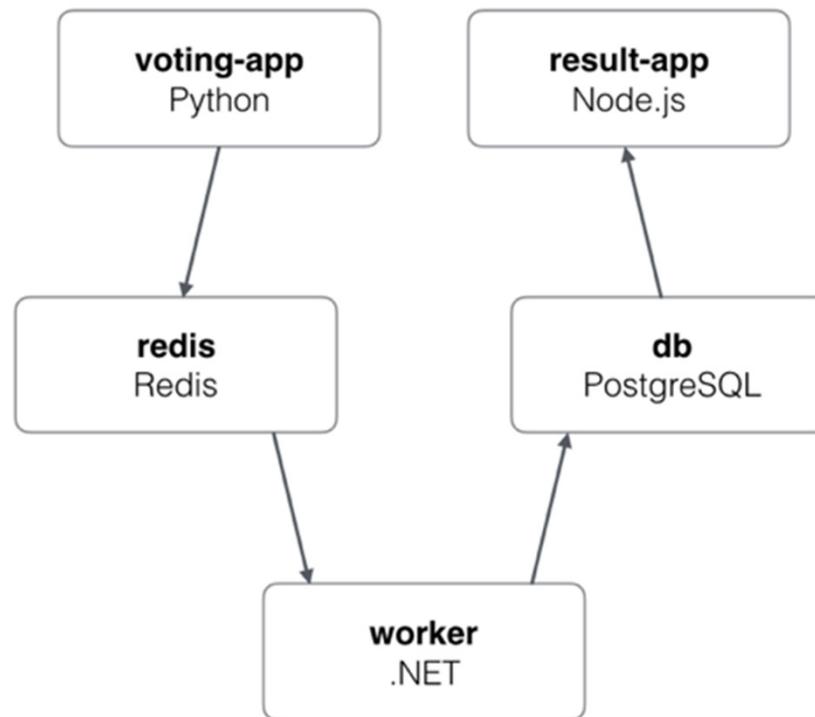
A screenshot of a terminal window with a dark background and light-colored text. The window has three tabs at the top, with the second tab active. The text in the terminal is as follows:

```
@      9, 9
@      A
@      #
# 9B    $ 9
# 9B$B$ B$B$9

*
9-, C+9

*      *      + D
@      #      @      *
      9
```

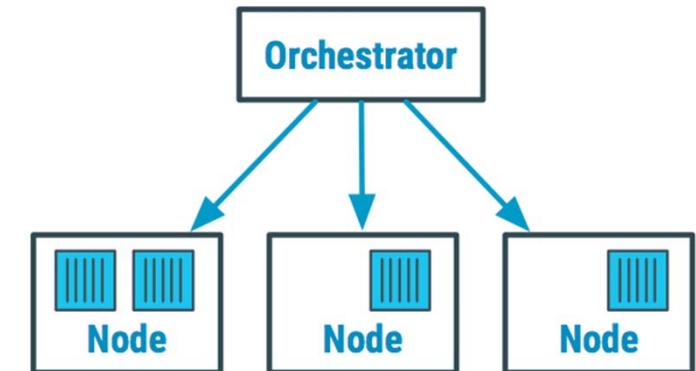
Docker Compose example



/11 \$, 1 1 " "

Container Orchestration

```
0      - /      >    ! 0 0      /  
--  "      ;  
- ;      -  
! ;"  
• 1 /  
• .  
• :  
• ? -  
• B
```

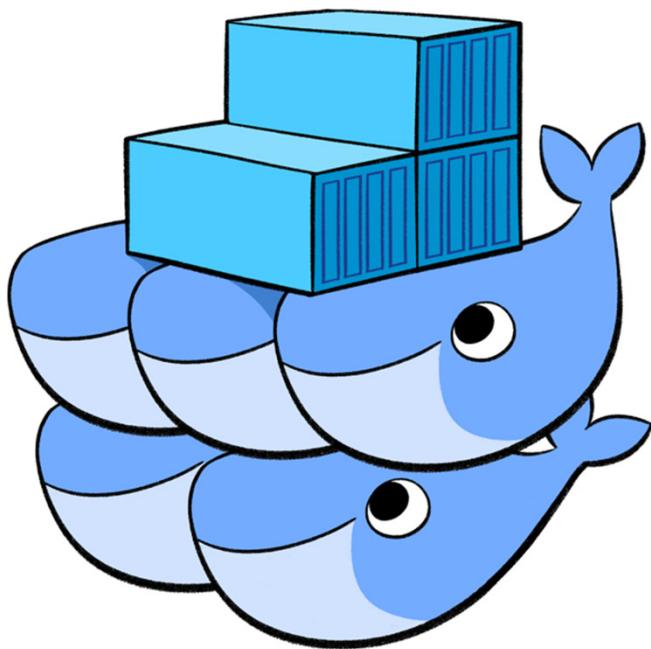


Container Orchestrators



kubernetes

Docker Swarm



```
B       C 1           < /          ;          -  
      . -           0  
      1.0           ;  
      0  
      #           C  
  
docker swarm init  
...  
      docker swarm join --token aabbcc 192.168.0.1:2377  
...  
      C  
  
docker swarm join --token aabbcc 192.168.0.1:2377
```



Kubernetes

; E
;
•
• ,
• /
• :
• :
• :
F- G# < - 0 / 0
-





Questions?

•

•

, 2

•

8

•

- • 1 / ' < --
- # ; --
 - G
 - 8
 - B
 - 000
- • 1 / H H --
 - | ! 6 & "C J A E
- |





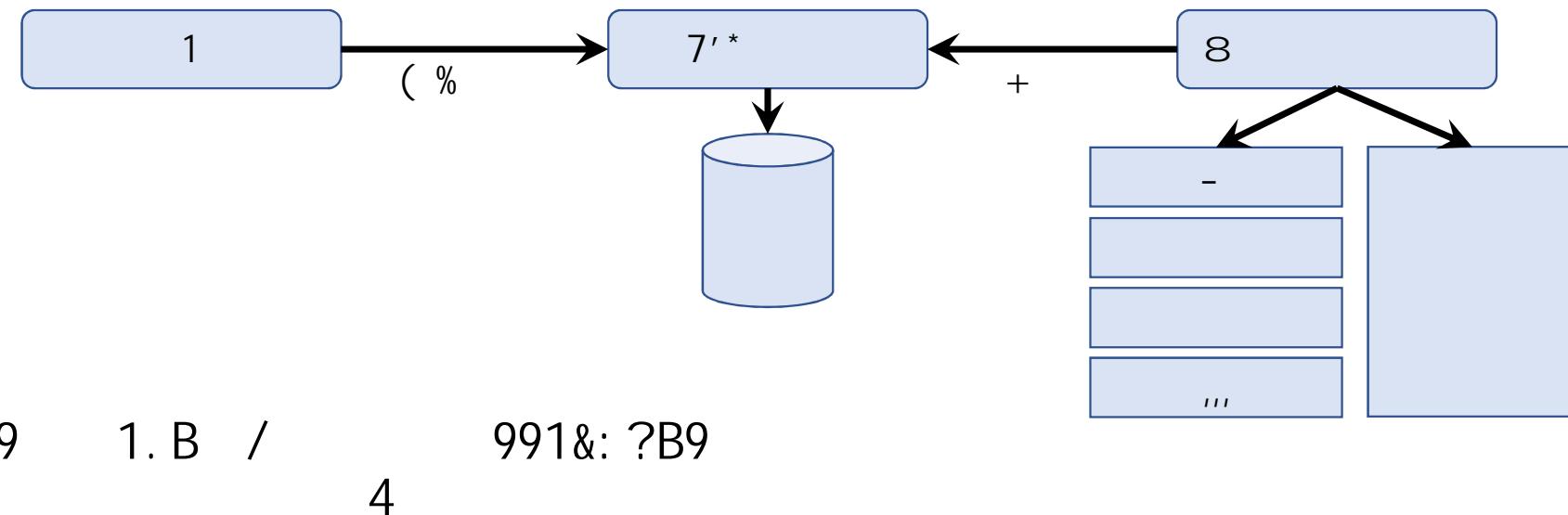
- 4 E A 4 ,
- , - E /
• 8 / - =
- - ; ? -
• : <# <G# <.4#< +



< ! "0

• 9 4 - | 40

• - ;



!

000"



- 9

A ; C < ; ; -- ; < ; / < 0 - <

- , ; A/ ;

< 0 !B C 0 "



#

:

/ ; - -

- # -
- B
- - ;
- ?/
- ? -
- .
- K
- 8
- 8 -
- 1
- 1 G
- : - B
- B
- B /
- B /
- B B <

- ;
- / - 0
- 9 <
- 1 C F
 - B
 - B / ; -
 - : - B < - ; <B - B <00
- B /
 - - ; -- 8B ;
 - ; - -- ;

000

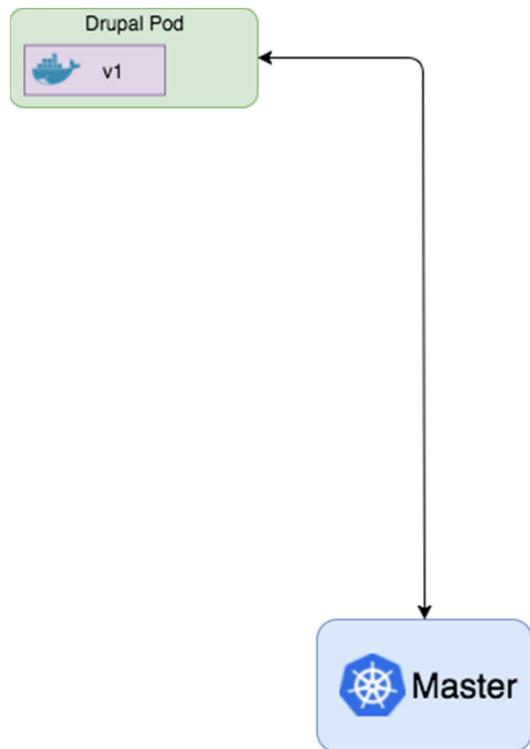
, 2



9 C

- 1
- B /
- G
- 8 -

, 2 C -



- 9 ' ! : \$
- ! ;
- 4 ! ! #
- ! !

```
spec:  
template:  
  spec:  
    containers:  
      - name: drupal  
        image: cr.io/repo/mydrupal:v1
```

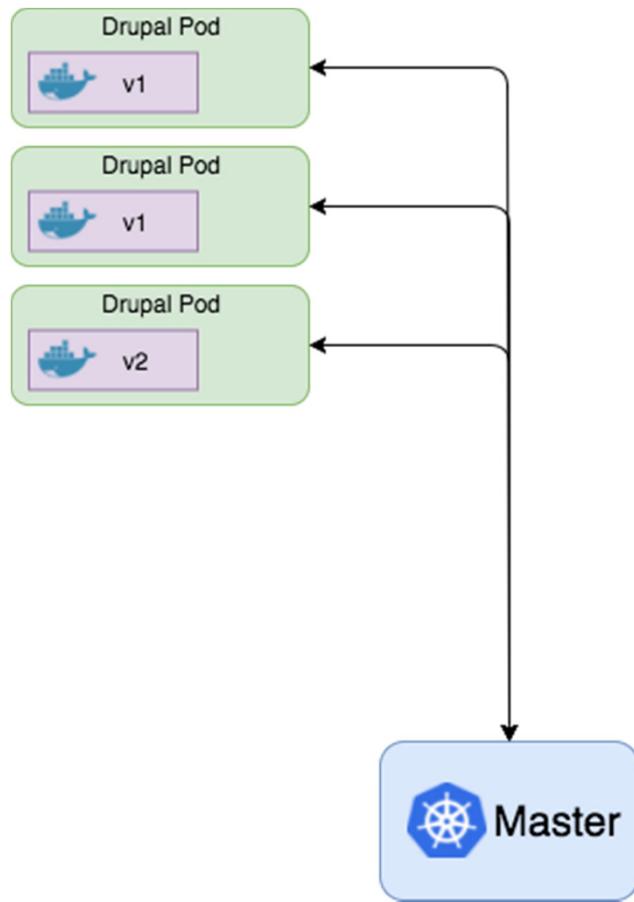


, 2 C -

• - - ; 0
• < -
• ; - | ; F - "
• < 6 --

• ? - C
• > .1
• / / - - -

, 2 C -

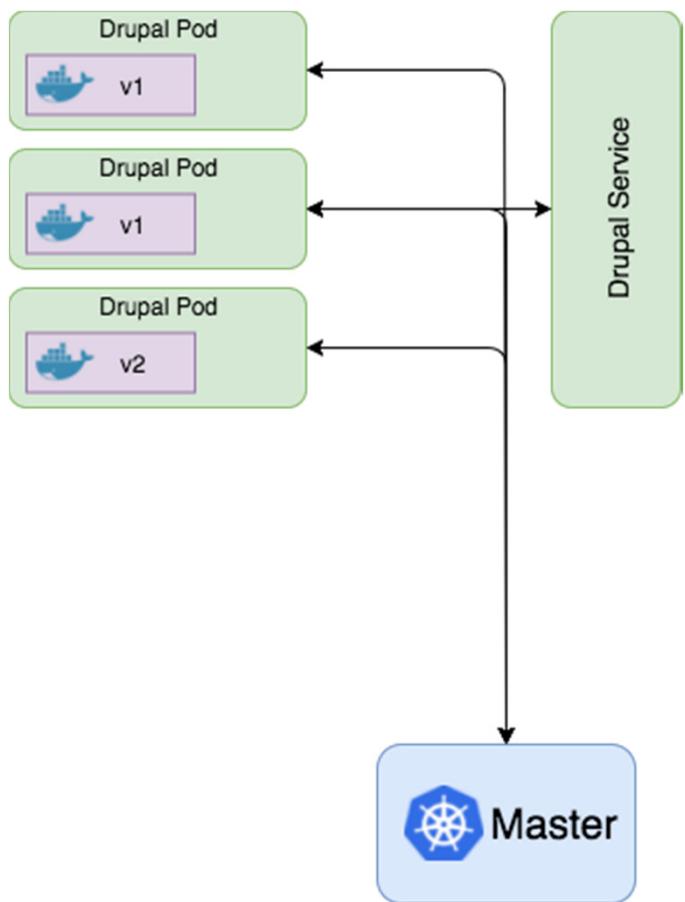


- ,
- B; 1
- B - &
- B; #
 - 1
 - ; ; ;
- - ;
 - - ; /
 - ; ; ;
- 4 &E - ;
- : - ;



, 2 C /

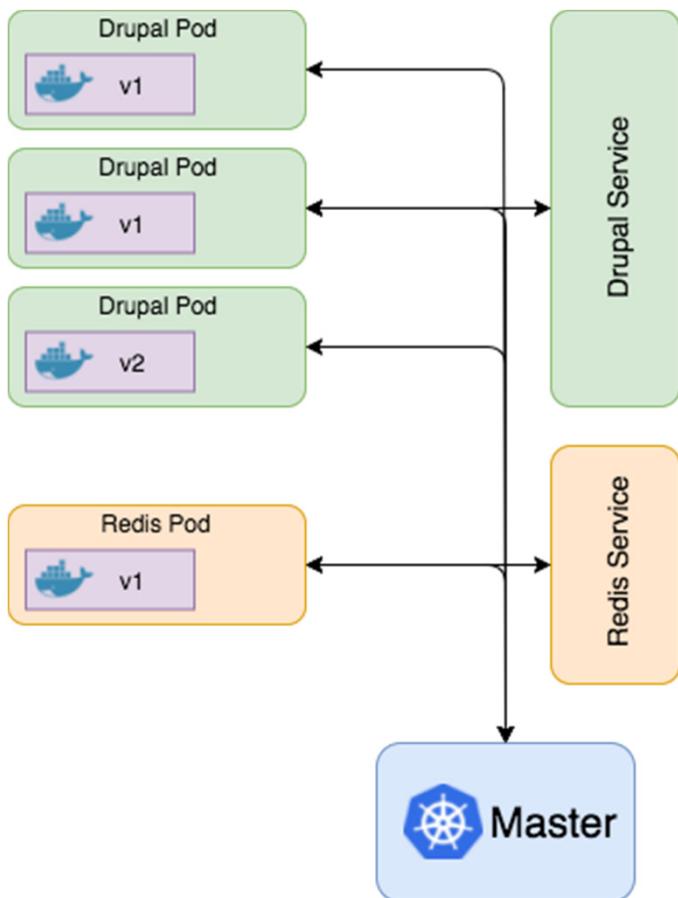
• - -
• / !
• - - ! " "
• B / C
• / .1 ! ; !
; ; "
• - / - <
• / < / /



, 2 C /
• B /
• A
• ;
/

```
apiVersion: v1
kind: Service
metadata:
  name: drupal
spec:
  selector:
    app: drupal
  ports:
    - name: http-port
      port: 80
  type: LoadBalancer
```

, 2 C /



- ; B / / ;
 - B / / ;
 -
 - B
- B / / ; - 2 /
- B / / - -
 - B
 - B - - /
 - ? B /
- 9 B / ; / ; ; ; -



, 2 C /

• ;< ; < - - ; / -

• / < - / F - L



, 2 C -

• - / - ;
• - / - C - >
• /
• ; A - 2 /
C

- - ;
• : - B
• <
• / ;
• / - !
- B B
• - ; - <
• ; < - <
- B
• - ; -
• < \emptyset
- K
• - <

- .
- 1 - ; /
• - C&& 0 & & & F
- 9 ; - / 2 &;

```
$ kubectl (create|get|apply|delete) -f myResource.yaml
```

8

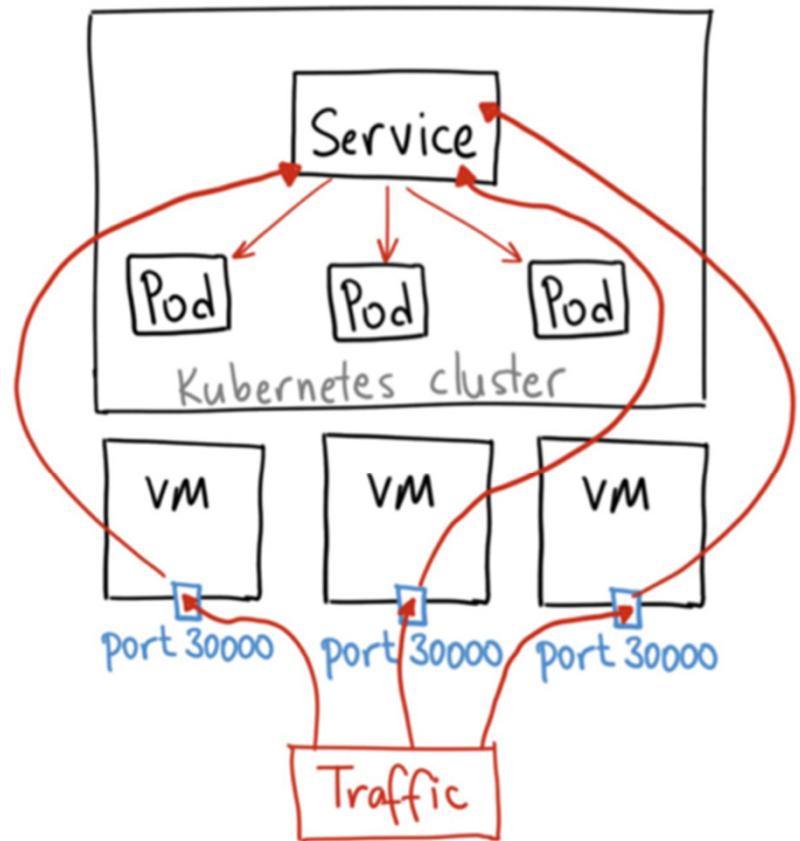
| 9; - &= | ' | M | , / ; | |
|---------|----------|---------|-----------|-----------|
| B ; | 1 ; ' | 1 / ; | 1 - / ; ; | 1 / |
| 9 ; ; | ' : 1< 4 | : - 4E1 | GN 8 | |
| ? - | 8 | 8 | , / ; | 8 |
| ? - | | | = < / | E B< B< B |

8

| = | 8 1 | 4 | . |
|-----|---|-----------------|-------------|
| B ; | B / - / - ! C M' %%%FM' O(0" | 9; - ; - ; - | 9; - ; - - |
| .1 | 8 .1 | ? / / .1 | # ; / .1< - |
| | 9 | M B / | 0 B / |
| ? - | | E ?8 | 8 < |

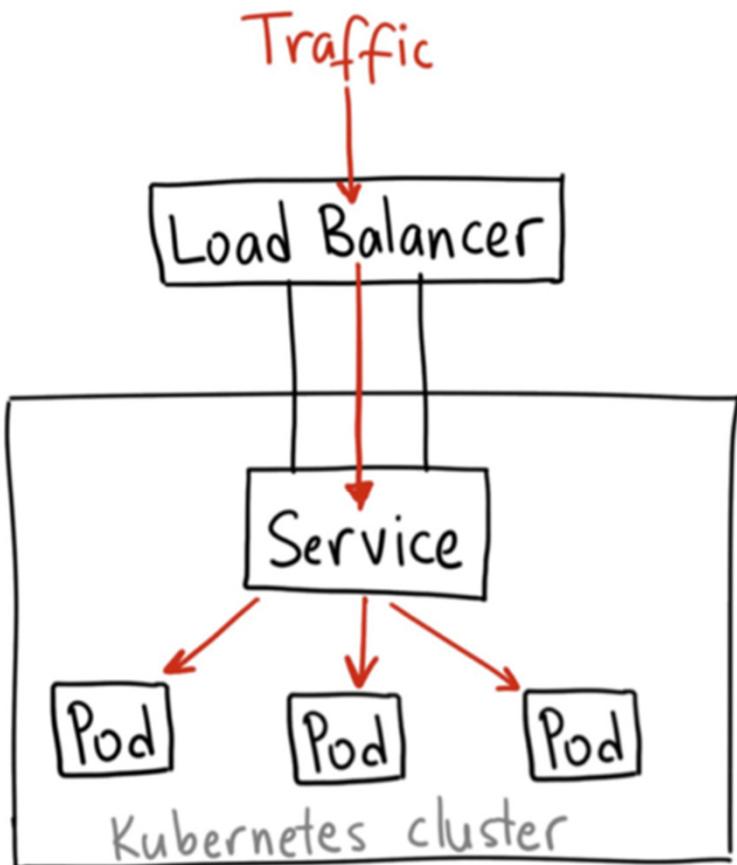
8

1



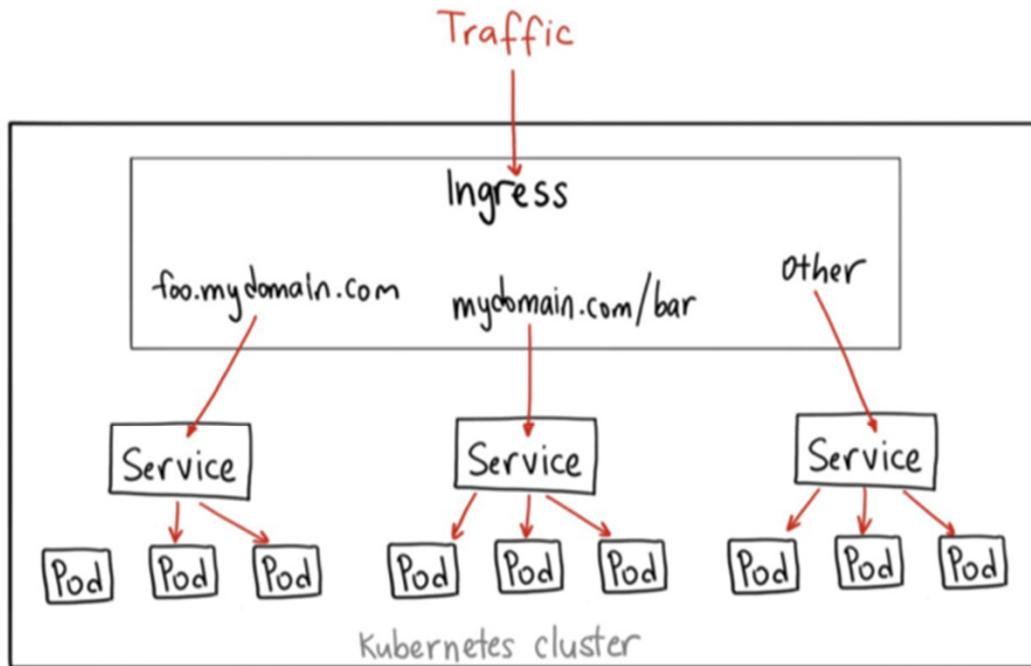
```
apiVersion: v1
kind: Service
metadata:
  name: productpage
  labels:
    app: productpage
spec:
  type: NodePort
  ports:
  - port: 30000
    targetPort: 9080
  selector:
    app: productpage
```

4



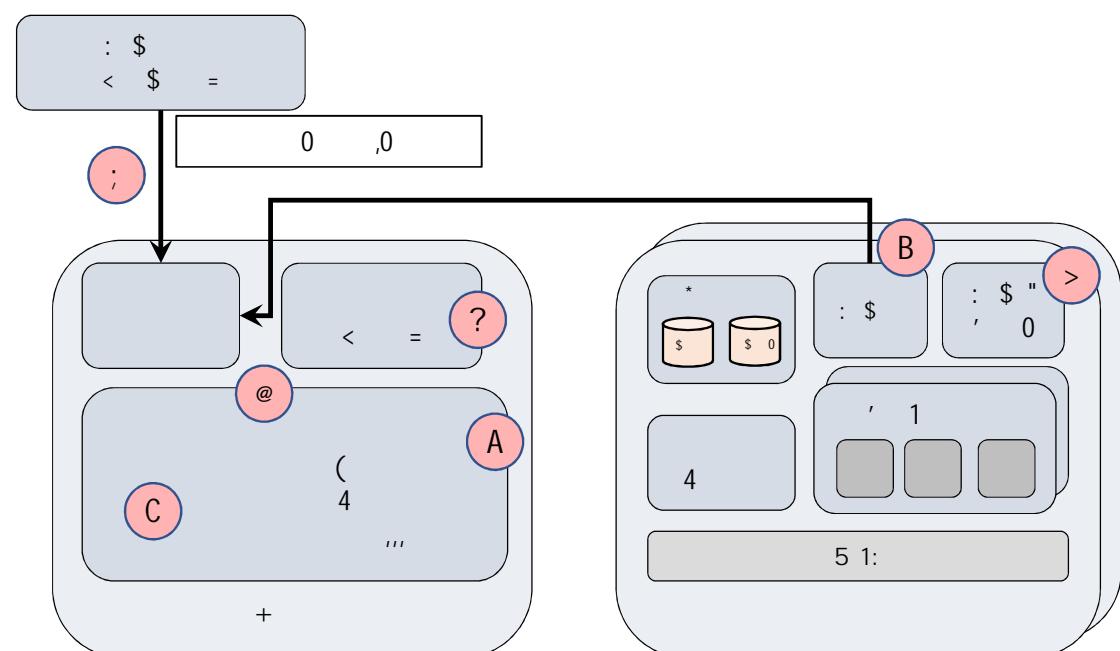
```
apiVersion: v1
kind: Service
metadata:
  name: productpage
  labels:
    app: productpage
spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 9080
  selector:
    app: productpage
```

4



```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: gateway
spec:
  backend:
    serviceName: productpage
    servicePort: 9080
  rules:
  - host: mydomain.com
    http:
      paths:
      - path: /productpage
        backend:
          serviceName: productpage
          servicePort: 9080
```

\$0 / ; --
 ' 0 1. / / >
 M0 & -
 70 : - B & -
 P -
 50 B -
 (0 - ! 0 0 ; "
 00 - ; Q / / ;

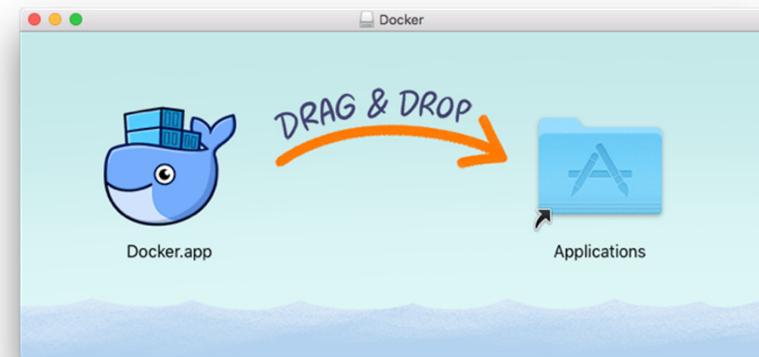
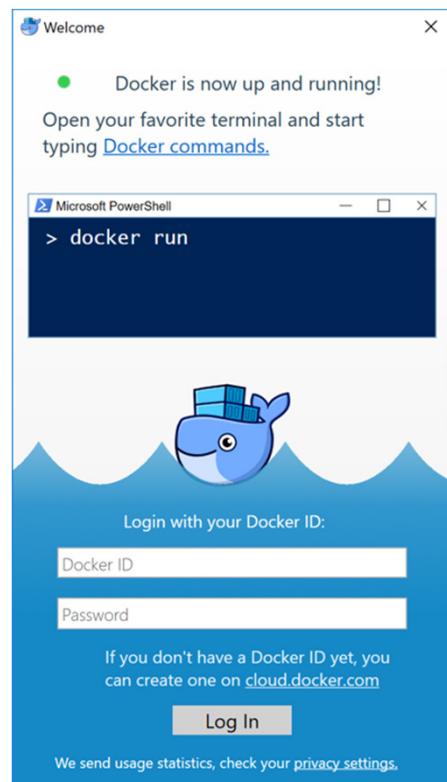




L

- # F - C&& 0
- B F - C&& 0 &
- @ 9 F - C&& 0; 0 & & R' %> 9, #% @ 32 .

E



• +

B /

• + ?

B /

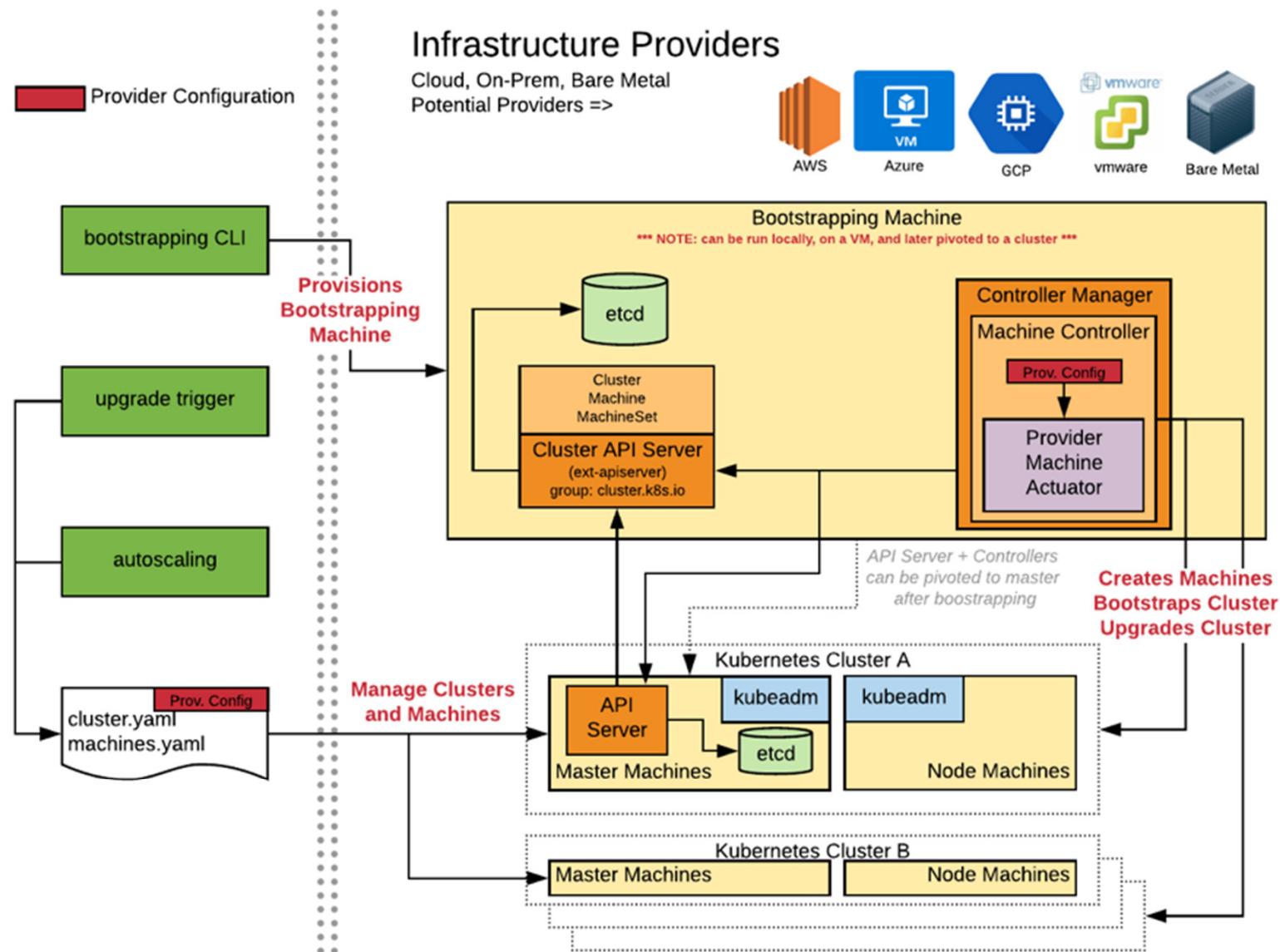
• E

?

• ,

F 1.

- ; \$ ' -
- ; \$" 4 -
 - B / %
- ; ' " #
 - # ;
 - -
 - B
- B + ;
 - # ; < / ; -
- ? - L





Leverage KubeVirt and Kubernetes to manage virtual machines for impractical-to-containerize apps.



Combine existing virtualized workloads with new container workloads on the one platform.



Support development of new microservice applications in containers that interact with existing virtualized applications.

8 =

* 8 / - =

