Emerging Devops Tech

July 24, 2014 for the iNovia CTO Summit
Peter Kieltyka / peter@pressly.com

PRESSLY

The devops wish

An easy to configure distributed infrastructure that is easy to maintain, resilient, loosely coupled, and for an application developer to never have to worry about how its all orchestrated

…and doesn’t cost a fortune

I wish to have…

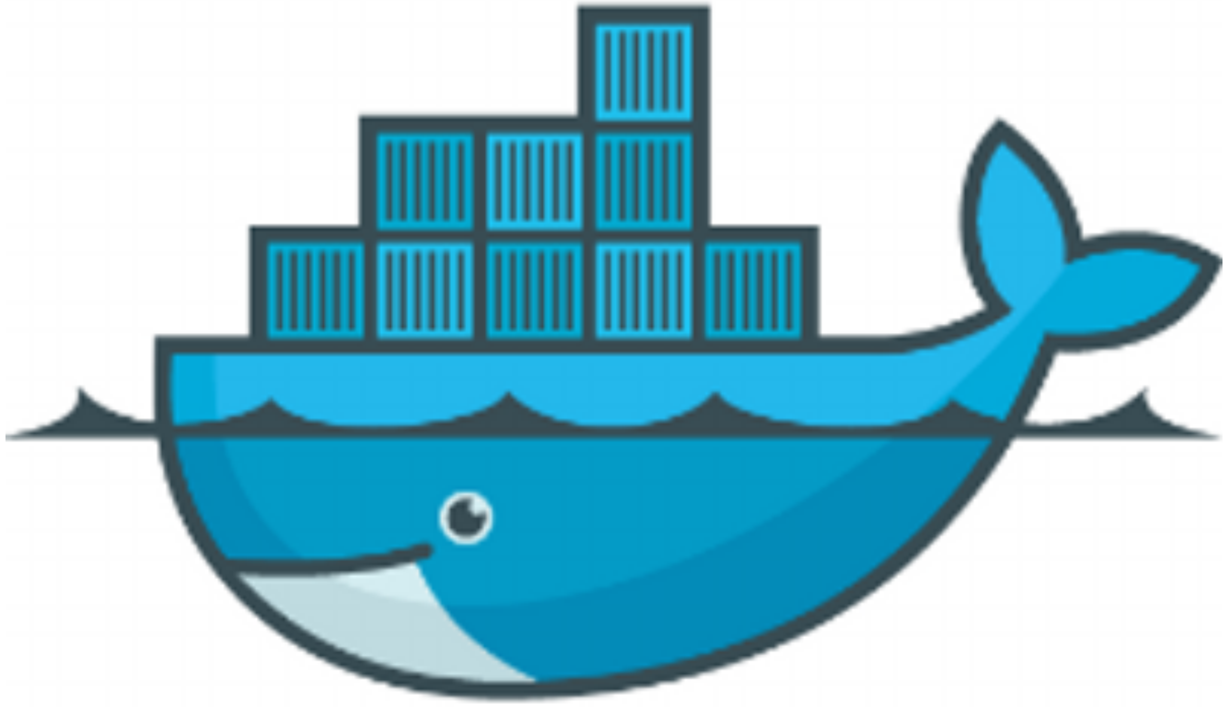
- A single app start/stop interface, regardless of my app's programming language
- Zero down-time application updates
- Easily deploy the stack to any data center: ie. dedicated hardware or some cloud provider (EC2 / Digital Ocean / GCE) — no, vendor lock-in
- Sysadmin friendly
- Easy staging / testing stack setups
- Developer friendly (deploy, and it just connects and works!)
- Easy scaling of a cluster
- A nice Web UI to dashboard for app / net state, logs and metrics — using the same Linux tools you can run in the console
- + More.. extensible; made up of many little components working together

From app to network

1. Machine provisioning
2. Application provisioning *****
3. Deployment (staging/production) *****
4. Service discovery & routing *****
5. Monitoring & metrics
6. Administration
7. Delivery

The next wave of server stacks will look like a shipment yard

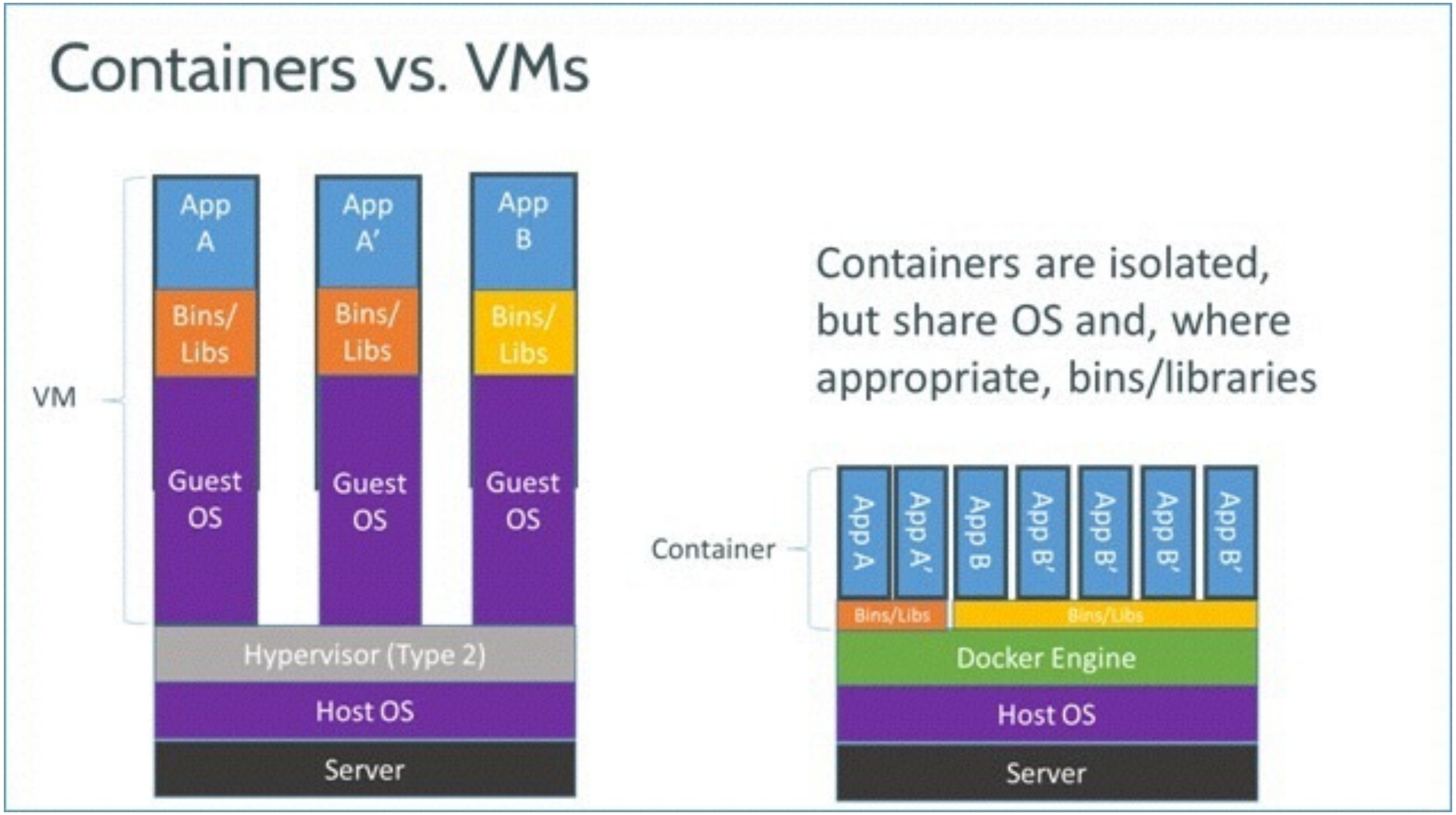
Docker : a simple and robust interface to Linux Containers (LxC)



Docker

- Created by Solomon Hykes, and initially released March 2013
- Docker makes it easy to bundle an application and all its dependencies into a single functioning unit running on an OS
- Container can be shipped over to a completely different Linux OS running docker within a few seconds
- Optimized for deployment of applications, as opposed to machines
- Versioning; includes git-like capabilities for tracking successive versions of a container
- Component reuse; ie. a “base image”
- Sharing; public/private registry of images (like github for app containers)

OS-level virtualization



*source: http://tiewei.github.io/cloud/Docker-Getting-Start/

An example, a story..

Let's say a developer on our team has just written a Sinatra web service that they'd like to deploy..

server.rb:

```
require ‘sinatra'

get '/hi' do
  "Hello World!"
end
```

Include a Dockerfile to build the application image

Dockerfile:

```
FROM litaio/ruby:2.1.2
RUN gem install sinatra
ADD . /app/
EXPOSE 4567
CMD ["ruby","/app/server.rb","-o","0.0.0.0"]
```

Build the application image

```
$ sudo docker build -t "pressly/sinatra-fun" .
```

```
Sending build context to Docker daemon 3.584 kB
Sending build context to Docker daemon
Step 0 : FROM litaio/ruby:2.1.2
 ---> cf2c24f3ceb2
Step 1 : MAINTAINER Peter Kieltyka <peter@pressly.com>
 ---> Using cache
 ---> 1659df108388
Step 2 : RUN gem install sinatra
 ---> Using cache
 ---> 98db28d9cbfa
Step 3 : ADD . /app/
 ---> c227d807a888
Removing intermediate container ddd6d7040252
Step 4 : EXPOSE 4567
 ---> Running in 5bfdb339861e
 ---> 2372fea64334
Removing intermediate container 5bfdb339861e
Step 5 : CMD ["ruby","/app/server.rb","-o","0.0.0.0"]
 ---> Running in ca58a7491a81
 ---> d0b933286767
Removing intermediate container ca58a7491a81
Successfully built d0b933286767
```

..we just created an image called pressly/sinatra-fun and tagged it 'latest'

```
$ sudo docker images
```

```
REPOSITORY             TAG            IMAGE ID          CREATED             VIRTUAL SIZE
pressly/sinatra-fun     latest         d0b933286767        37 seconds ago      577.6 MB
```

..the images available on our host OS

```
$ sudo docker run -d -p 80:4567 --name webapp pressly/sinatra-fun
```

```
d7e818a8097d3ee23424e4a7c43390c6893c2cf14fb00c88312aa71ca0356a28
```

..we just started a docker container running the image “pressly/sinatra-fun” with ‘docker run’, which just returned to us our **container ID** let’s make sure its running

```
$ sudo docker ps
```

```
CONTAINER ID        IMAGE                          COMMAND               CREATED             STATUS              PORTS                    NAMES
d7e818a8097d        pressly/sinatra-fun:latest   ruby /app/server.rb    6 seconds ago       Up 5 seconds        0.0.0.0:80->4567/tcp   webapp
```

let’s tail the container’s logs

```
$ sudo docker logs -f webapp
```

```
[2014-07-24 12:53:25] INFO  WEBrick 1.3.1
[2014-07-24 12:53:25] INFO  ruby 2.1.2 (2014-05-08) [x86_64-linux]
== Sinatra/1.4.5 has taken the stage on 4567 for development with backup from WEBrick
[2014-07-24 12:53:25] INFO  WEBrick::HTTPServer#start: pid=1 port=4567
```

Tada!

```
$ curl http://<host-ip>/hi (ie: 54.237.200.128)
```

```
Hello World!
```

Push a release

- Sign up docker.io for public / private
- Quay.io is another registry for private repos

```
$ sudo docker login
<...>
```

```
$ sudo docker push pressly/sinatra-fun
The push refers to a repository [pressly/sinatra-fun] (len: 1)
Sending image list
Pushing repository pressly/sinatra-fun (1 tags)
511136ea3c5a: Image already pushed, skipping
e465fff03bce: Image already pushed, skipping
23f361102fae: Image already pushed, skipping
9db365ecbcbb: Image already pushed, skipping
ad892dd21d60: Image already pushed, skipping
aed4716443df: Image already pushed, skipping
cf2c24f3ceb2: Image already pushed, skipping
1659df108388: Image successfully pushed
98db28d9cbfa: Image successfully pushed
c227d807a888: Image successfully pushed
2372fea64334: Image successfully pushed
d0b933286767: Image successfully pushed
Pushing tag for rev [d0b933286767] on {https://registry-1.docker.io/v1/repositories/pressly/sinatra-fun/
tags/latest}
```

Anyone can run it

From any machine running Docker, the *pressly/sinatra-fun* application can be deployed in seconds! ie. can be a Vagrant dev machine, AWS, GCE, hardware, etc. Try it!

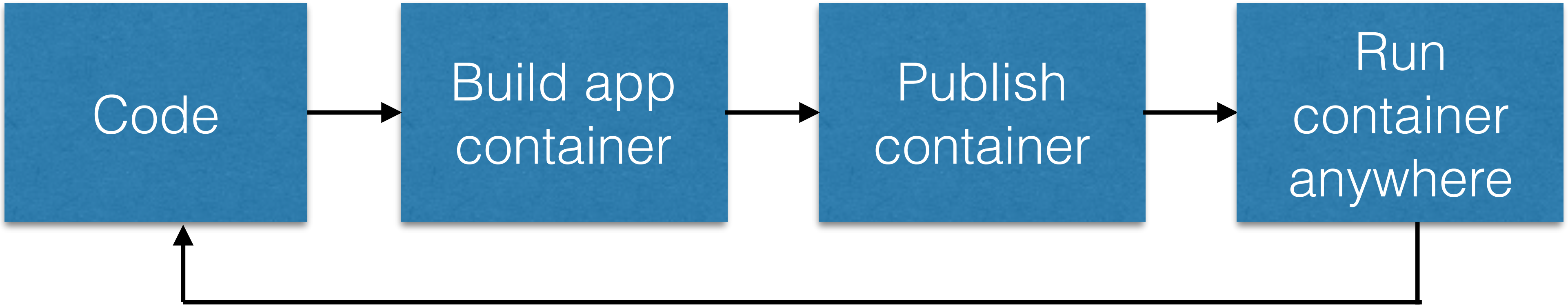
```
$ sudo docker run -d -p 1234:4567 --name sinatra pressly/sinatra-fun
Unable to find image 'pressly/sinatra-fun' locally
Pulling repository pressly/sinatra-fun
d0b933286767: Download complete
511136ea3c5a: Download complete
e465fff03bce: Download complete
23f361102fae: Download complete
9db365ecbcbb: Download complete
ad892dd21d60: Download complete
aed4716443df: Download complete
cf2c24f3ceb2: Download complete
1659df108388: Download complete
98db28d9cbfa: Download complete
c227d807a888: Download complete
2372fea64334: Download complete
8de547edc196cd6c36f022090308bdb8e41e0de7a63282d002f736e6fa7873ff
```

.. the sinatra webapp is now running and listening on port 1234

Btw, this is the *ruby* Dockerfile

```
FROM ubuntu:14.04
MAINTAINER Jimmy Cuadra
RUN apt-get update && \
  DEBIAN_FRONTEND=noninteractive apt-get upgrade -y && \
  DEBIAN_FRONTEND=noninteractive apt-get -y install \
    build-essential \
    curl \
    git-core \
    libcurl4-openssl-dev \
    libreadline-dev \
    libssl-dev \
    libxml2-dev \
    libxslt1-dev \
    libyaml-dev \
    zlib1g-dev && \
  curl -O http://ftp.ruby-lang.org/pub/ruby/2.1/ruby-2.1.2.tar.gz && \
  tar -zxvf ruby-2.1.2.tar.gz && \
  cd ruby-2.1.2 && \
  ./configure --disable-install-doc && \
  make && \
  make install && \
  cd .. && \
  rm -r ruby-2.1.2 ruby-2.1.2.tar.gz && \
  echo 'gem: --no-document' > /usr/local/etc/gemrc
```

Continuous app delivery



The next pieces to Docker

A lot of amazing work is happening right now to thats adds a higher-level orchestration to containers. Some projects:

- **CoreOS** — a Linux flavour designed for running clusters — like the low-level parts of Heroku (more on this)
- **Vagrant** — easier development environments via HW virtualization
- **Packer** — virtual machine builder with script provisioning (no vendor lock in!)
- **Serf** — autonomous node discovery via gossip`ing
- **Etcd** / Consul — service discovery + data persistence
- **Fleet** — distributed init system
- **Deis** / Flynn — PaaS .. more layers, less thinking
- **Dokku** — a mini-heroku in 100 lines of bash
- **Fig** — a simple way to express a docker dev environment, and run it
- ..and expect many more..

Enter CoreOS

- A minimal Linux operating system designed for deploying distributed server stacks, built on Docker
- A read-only boot partition that self auto-updates and will even reboot your system on an update
- Docker pre-installed
- No package manager
- Systemd init system
- Latest Linux kernel (3.15+)
- Created coreos/etcd project
- Created coreos/fleet project

Etcd

- Service discovery & cluster storage for small data
- Essentially, a simple key/value store managed by the Raft consensus protocol
- Raft provides a strongly consistent, distributed log
- Then overlay a key/value store for each node with an HTTP service
- It's like a really simple bitcoin blockchain :P
- Perfect for keeping data in-sync between servers.. for example, like local IP's and hostnames for easy routing
- *another really cool alternative for service discovery is Consul, by the Vagrant guys

Fleet

a distributed init system built on systemd

```
$ cat sinatra.service
[Unit]
Description=Sinatra Fun
After=docker.service
Requires=docker.service

[Service]
ExecStart=/usr/bin/docker run --rm -p 80:4567 --name webapp pressly/sinatra-fun

$ fleetctl start sinatra.service
Job sinatra.service launched on b064a38e…/10.5.170.101

$ curl http://10.5.170.101/hi
Hello World!
```

The End

Docker is stable, has reached v1.0 on June 2014, and today is at v1.1.1. CoreOS has a lot of great ideas and beginnings, but there still lots of work before my devops dreams come true!

.. but, it’s happening right now!