

mesosphere

Deploying Docker to Prod Scale

Ken Sipe, Cloud Solution Architect
ken@mesosphere.io

@Mesosphere

@kensipe

ken@mesosphere.io



Cloud Solution Architect

Developer: Embedded, C++, Java, Groovy, Grails, C#, Objective C

Cloud R&D Researcher

Speaker: JavaOne 2009 Rock Star, NFJS, JAX

Workshop

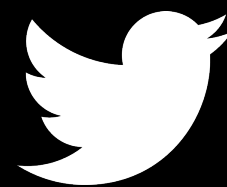
- Labs: Docker
- Labs: Mesos
 - GCE / DO

Expectations

- Docker Installed
- Mesos
 - digitalocean.mesosphere.com
 - google.mesosphere.com
 - USB

Agenda

- Legacy Datacenter
- Datacenter Trends
- Datacenter Goals
- Mesosphere
- Demos



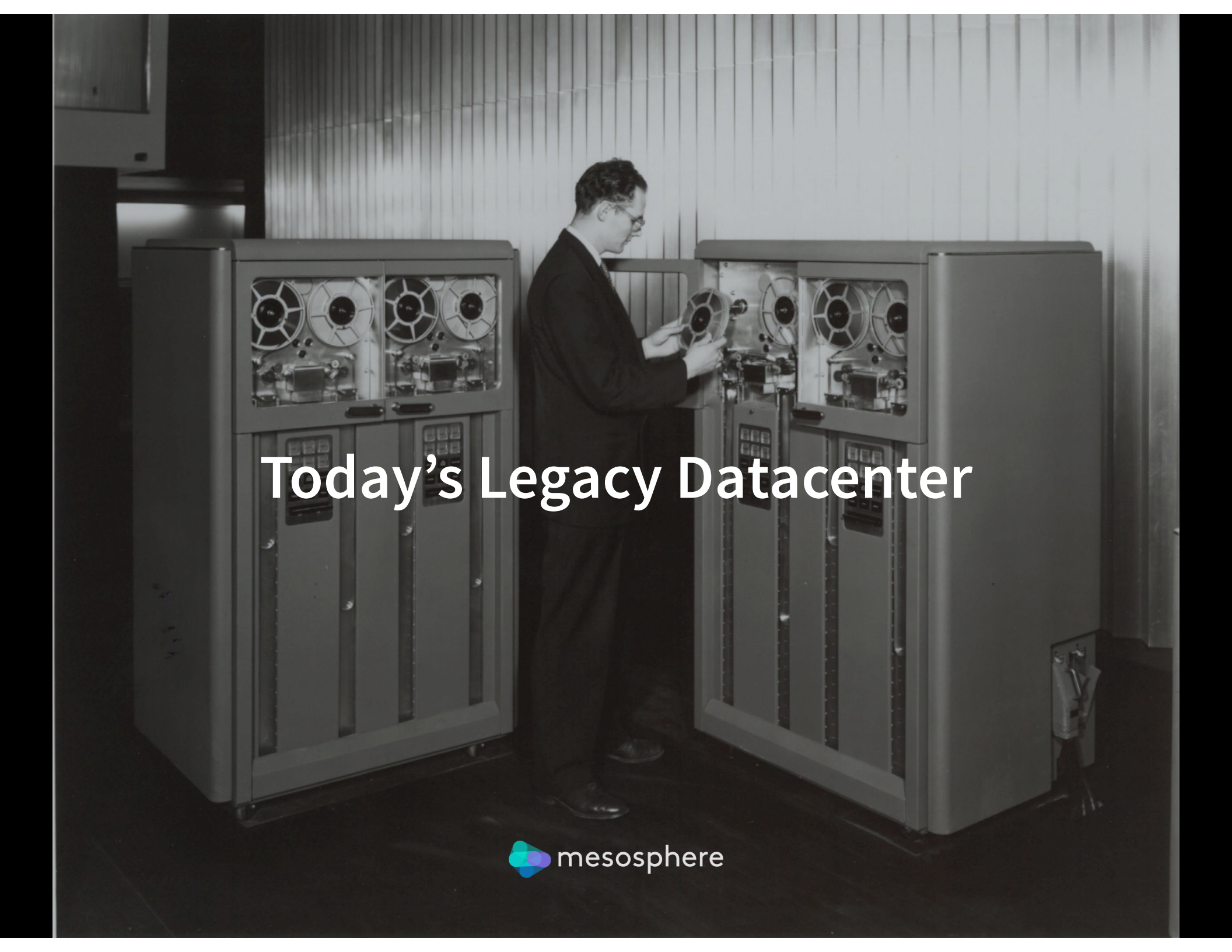
Chris Aniszczyk:

“When is the last time you’ve seen the fail whale on twitter?”

Datacenters

Perfect Storm

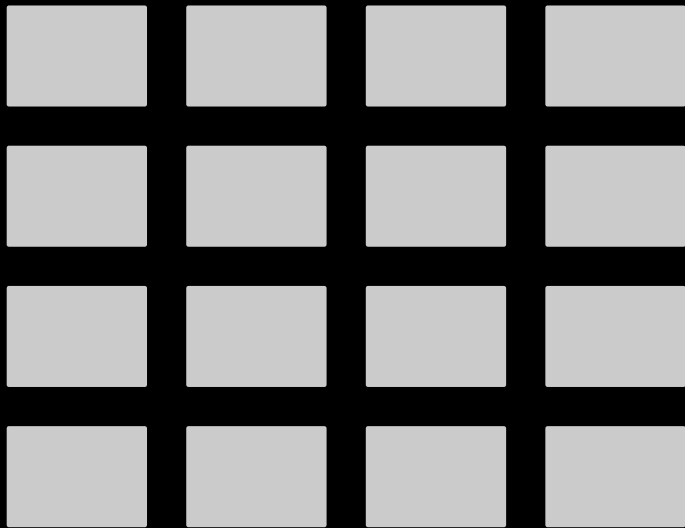




Today's Legacy Datacenter

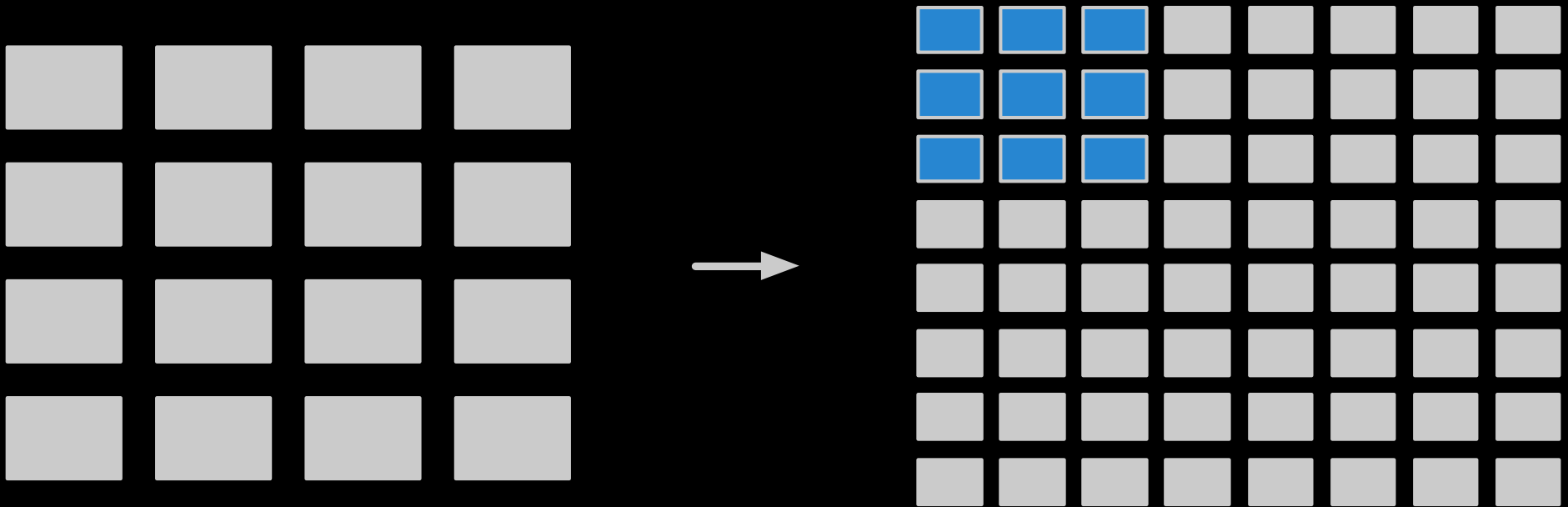
Challenge: Static Partitioning

Today's Legacy Datacenter



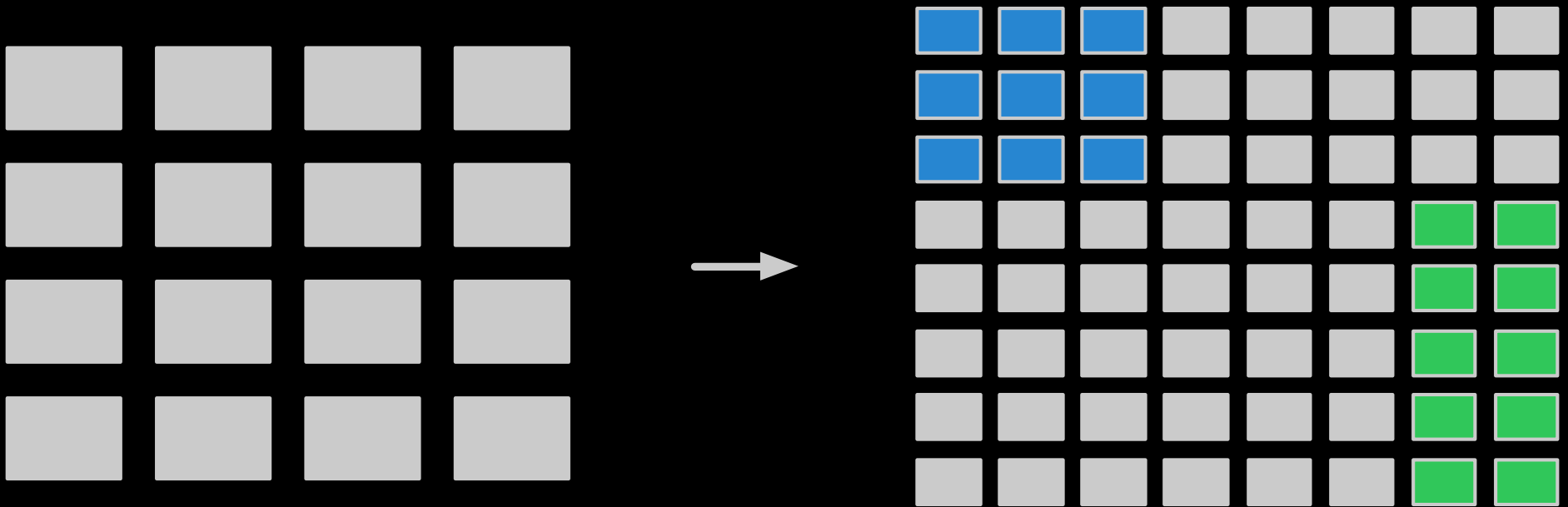
Provision VMs in the cloud or on physical servers

Installing an Application with Static Partitioning



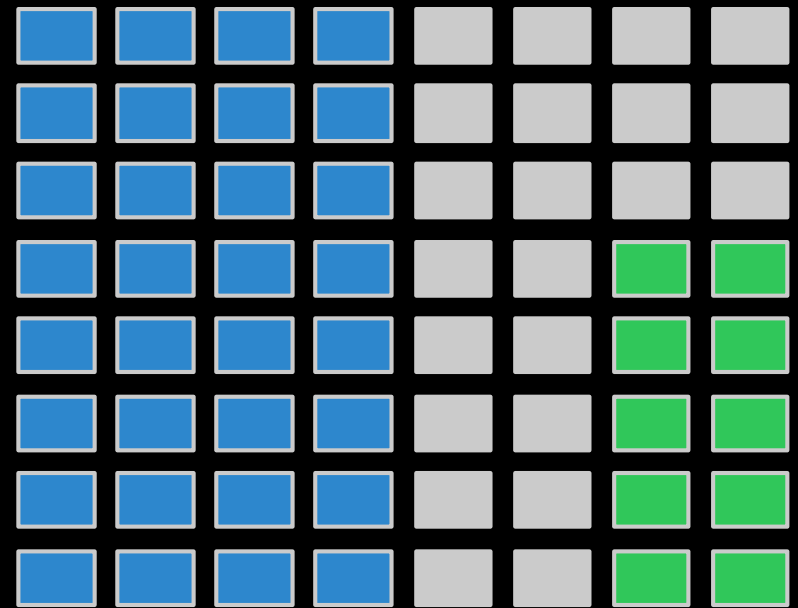
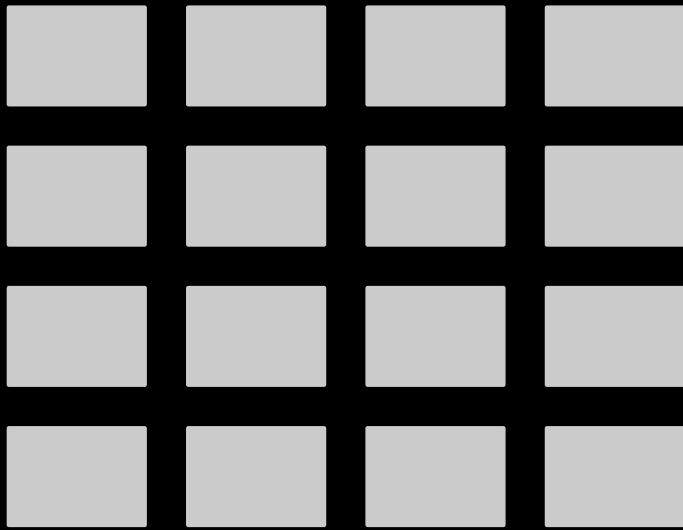
Install Hadoop on a static set of machines

Installing an Application with Static Partitioning



Install Web Server on a static set of machines

Resizing an Application with Static Partitioning

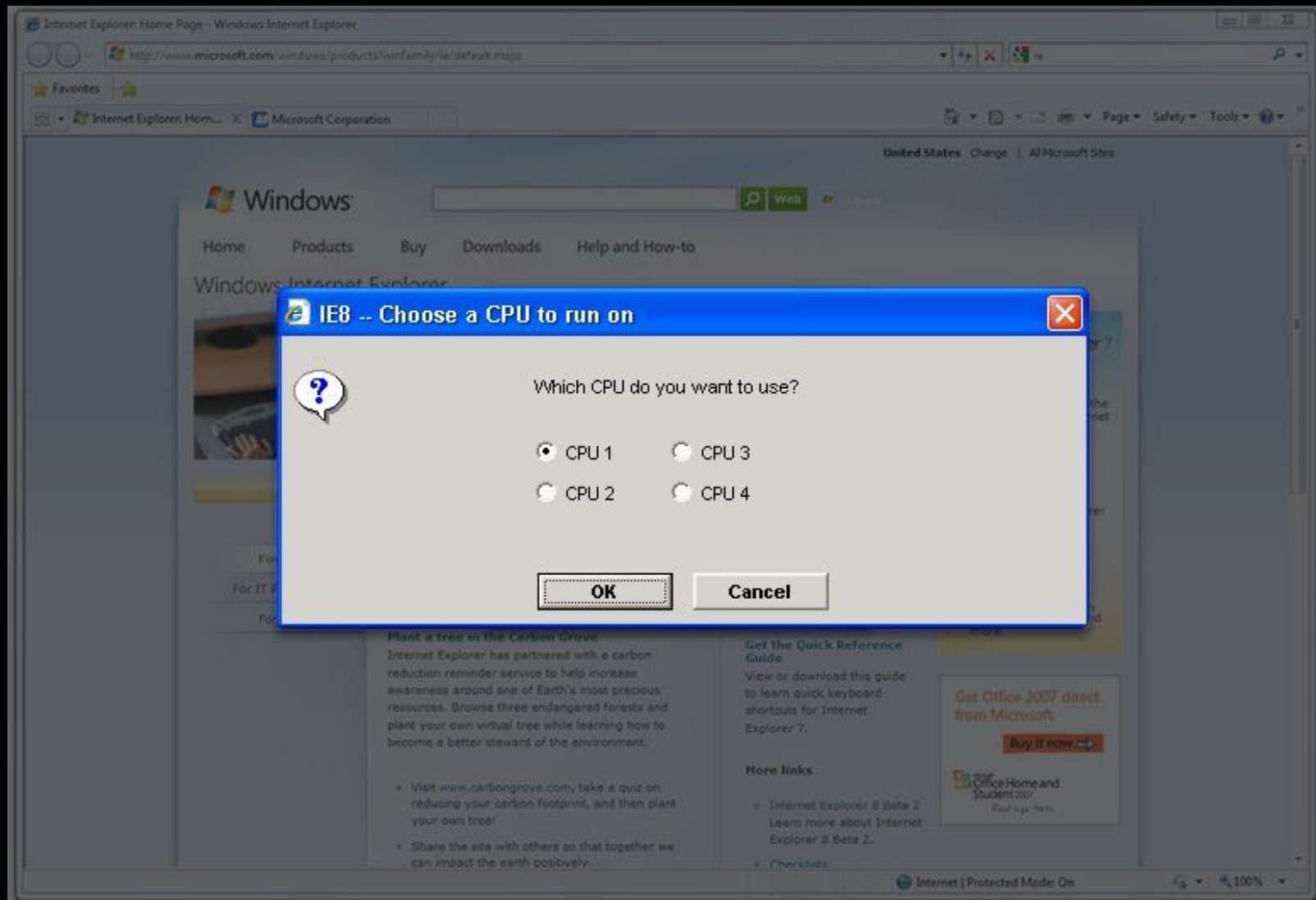


Scale up Hadoop manually

Challenge: Humans Involved!

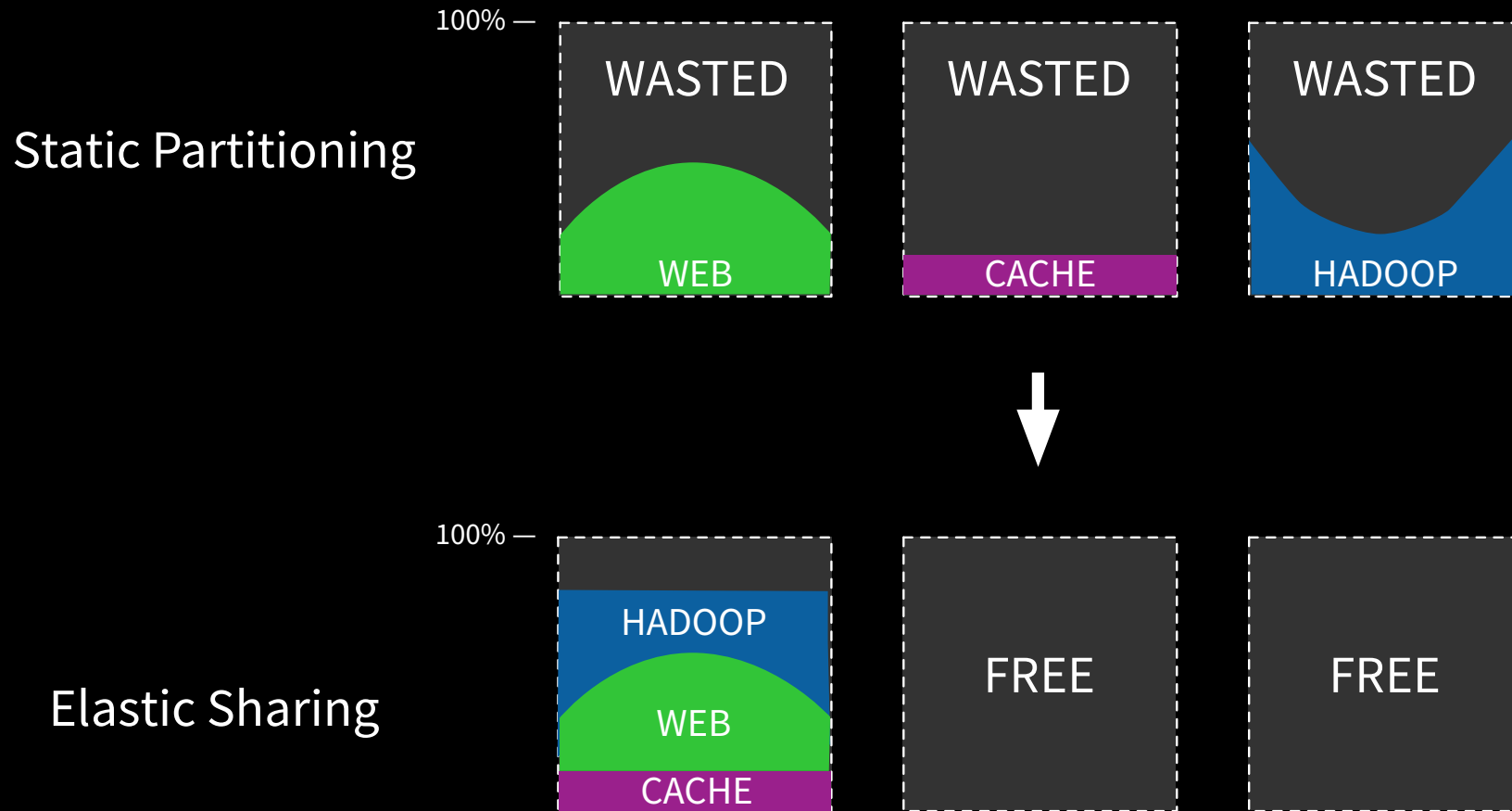
Challenge: Known IP and Port for Resource

What if your Laptop was operated like your Data Center?



Challenge: Resource Utilization

From Static Partitioning to Elastic Sharing



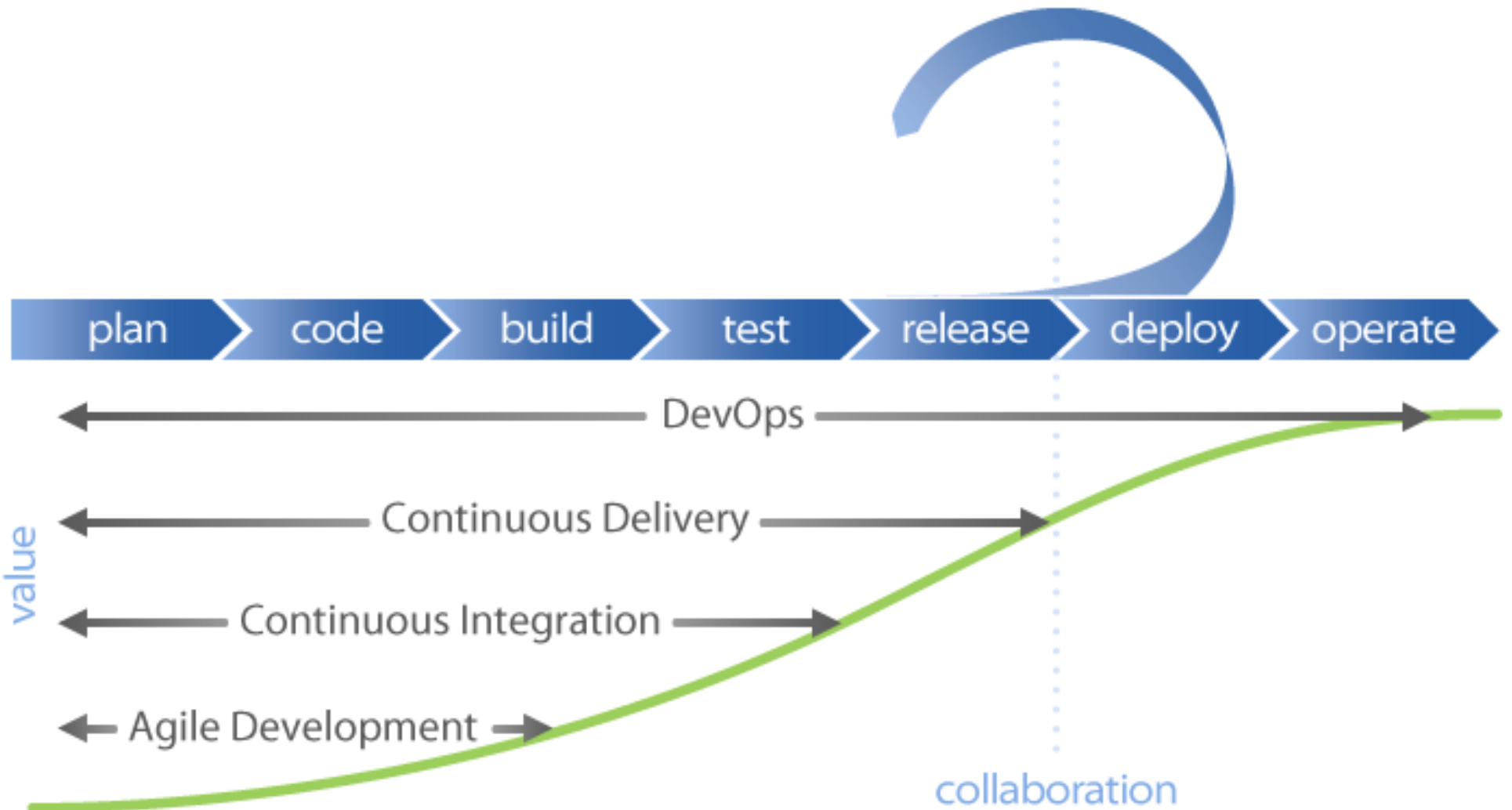
Google Borg



<http://www.wired.com/wiredenterprise/2013/03/google-borg-twitter-mesos/all/>

Challenge: Time to Production

Continuous Delivery / DevOps



Challenge: Continuous Delivery

- Virtual Machine / Stage Provisioning

Challenge: Virtual Machines

Challenge: Virtual Machines

- Large / Heavy Solution
- No Meta-Data
- Re-provisioned For Each Environment

Issues with Statically Partitioned Data Centers

Complex

Machine sprawl, manual resize/scale

Fragile

No software failure handling, “black box”

Inefficient

Static partitioning, overhead

Not Developer-Friendly

Long time to roll out software, development starts at the machine level

Perfect Storm



Trends in Datacenters

New Class of Applications

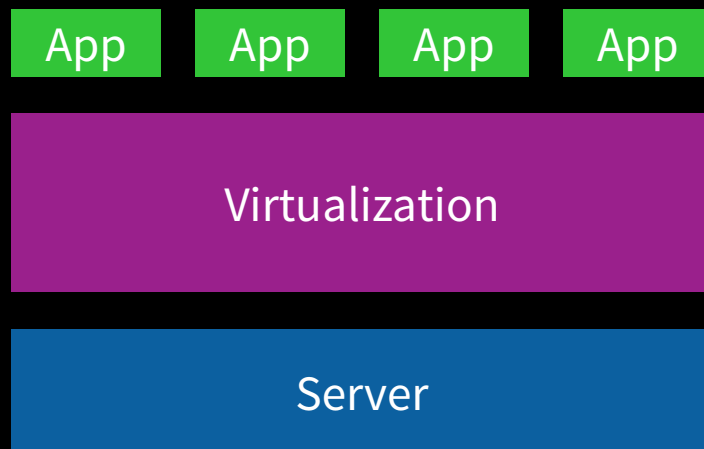
Elastic Partitioning

Containers

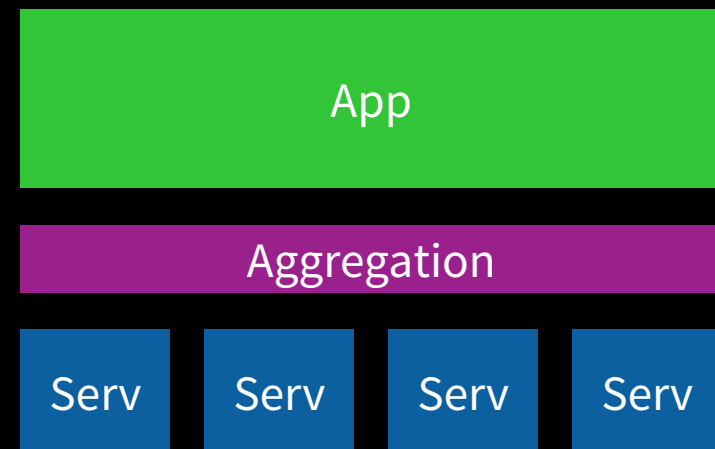
Micro-Services Architecture



Applications in the Cloud Era

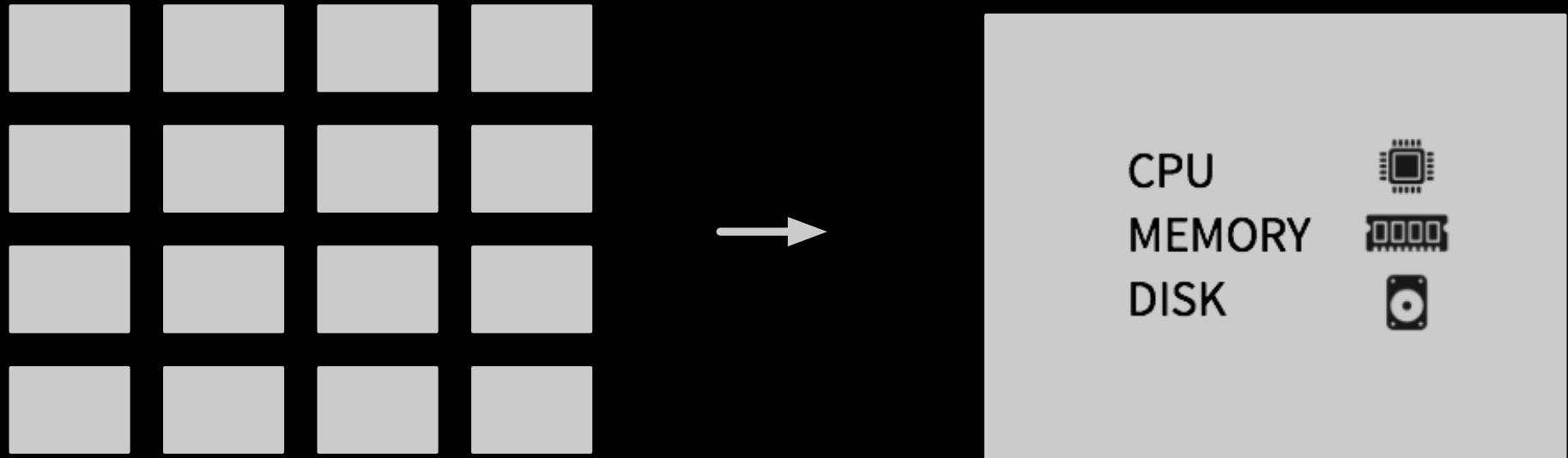


Client-Server Era:
Small apps, big servers



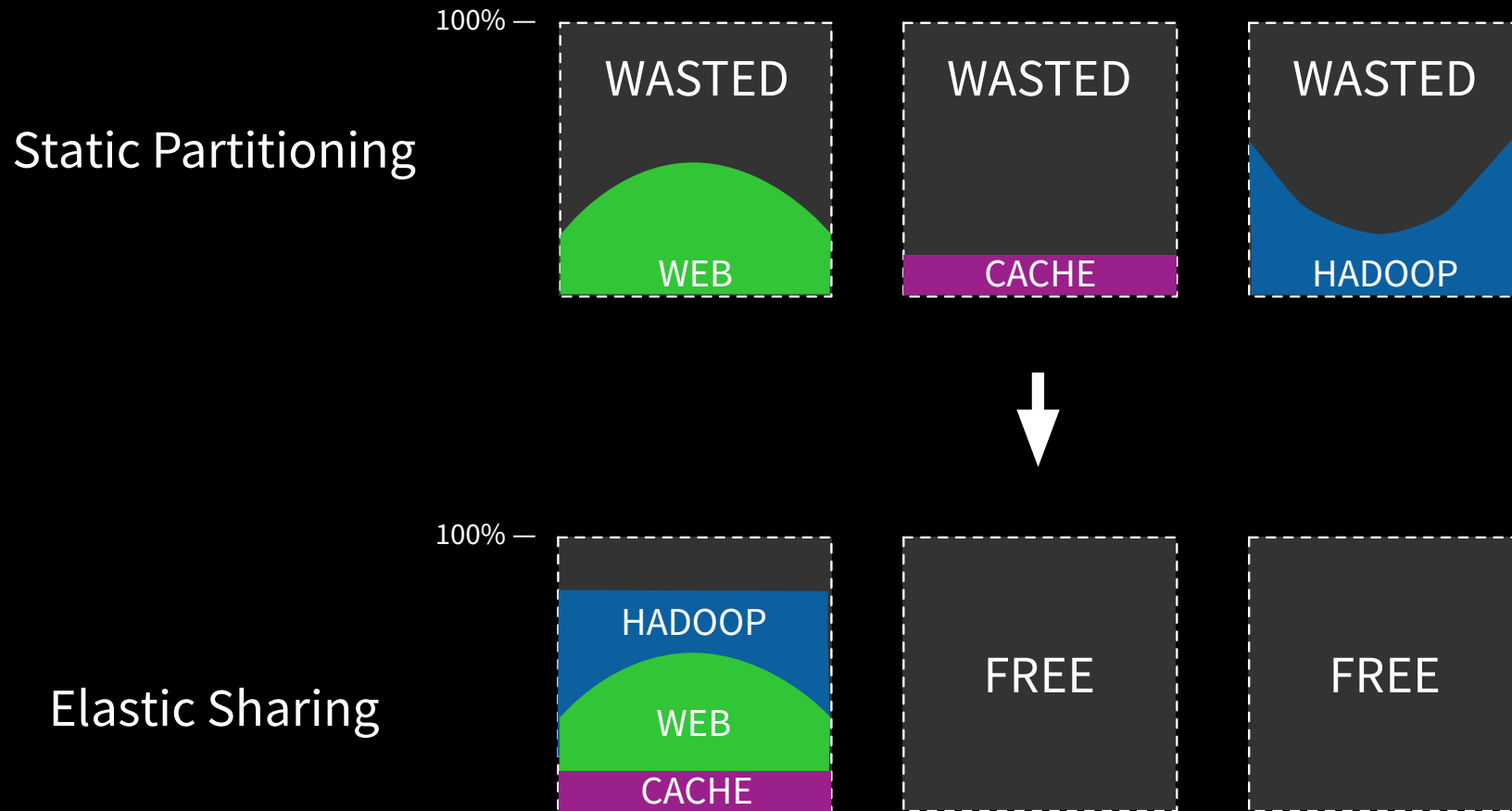
Cloud Era:
Big apps, small servers

Aggregation



Mesosphere aggregates resources, makes a data center look like one big computer
Mesosphere runs on top of a VM or on bare metal

From Static Partitioning to Elastic Sharing



Containers



docker

- Immutable Containers

How to scale Docker containers in production



▲
51
▼
☆
71

So I recently discovered this awesome tool, and it says

Docker is an open-source project to easily create lightweight, portable, self-sufficient containers from any application. The same container that a developer builds and tests on a laptop can run at scale, in production, on VMs, bare metal, OpenStack clusters, public clouds and more.

Let's say I have a docker image which runs Nginx and a website connects to external database. How do I scale the container in production?

scale production docker

share | edit | flag

add a comment

start a bounty

edited May 2 at 12:04
Tshepang
2,747 ● 8 ● 40 ● 79

asked Aug 17 '13 at 4:47
James Lin
2,800 ● 2 ● 19 ● 44

6 Answers

active oldest votes

asked 1 year ago
viewed 16177 times
active 7 days ago

Featured on Meta

- Impose a 24 hour voting freeze on questions being discussed on Meta
- Can we make this meta site work for mentoring?

Hot Meta Posts

- 11 Need for a P4VS tag?
- 3 Tooltip for comment flags displays the wrong reason



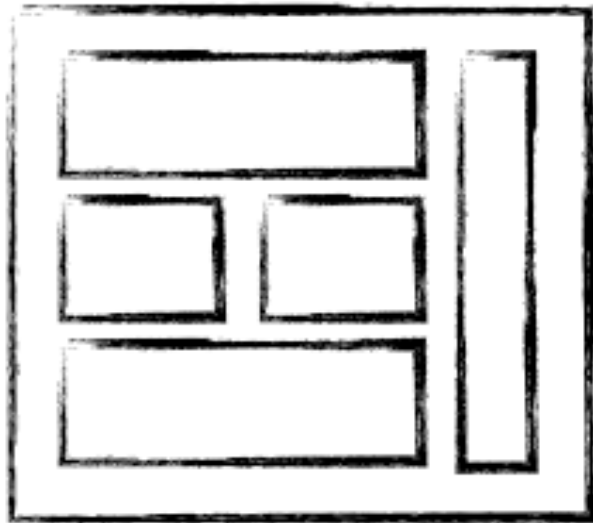
Kubernetes

<https://github.com/GoogleCloudPlatform/kubernetes>

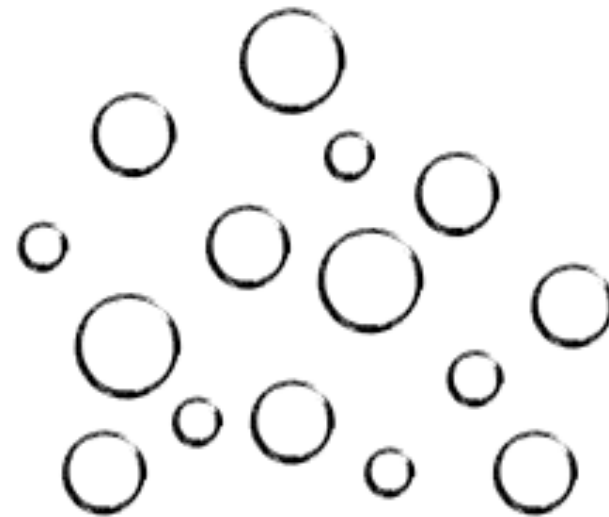
<https://github.com/mesosphere/kubernetes-mesos>



Monorail => Micro Services

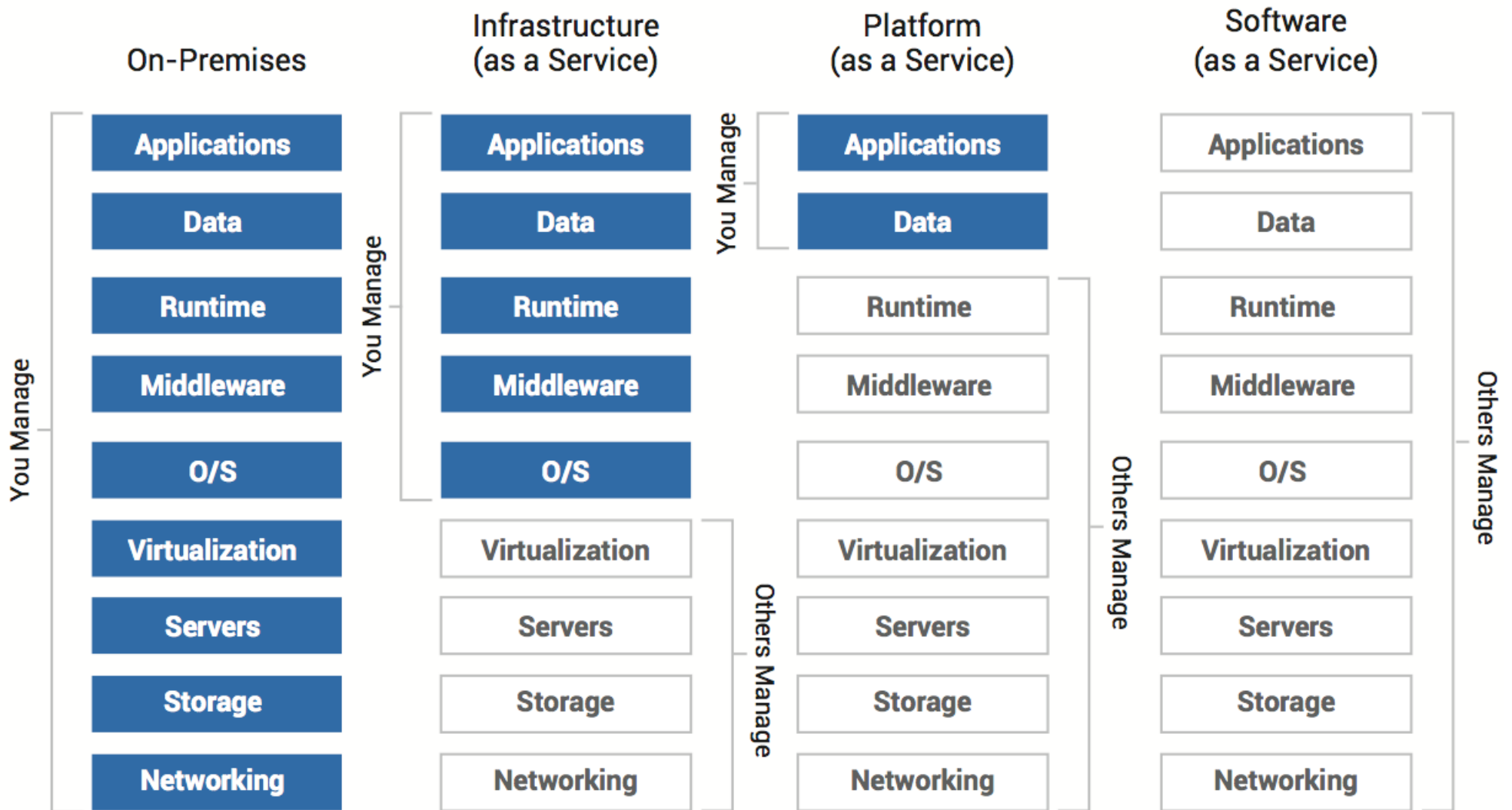


MONOLITHIC/LAYERED

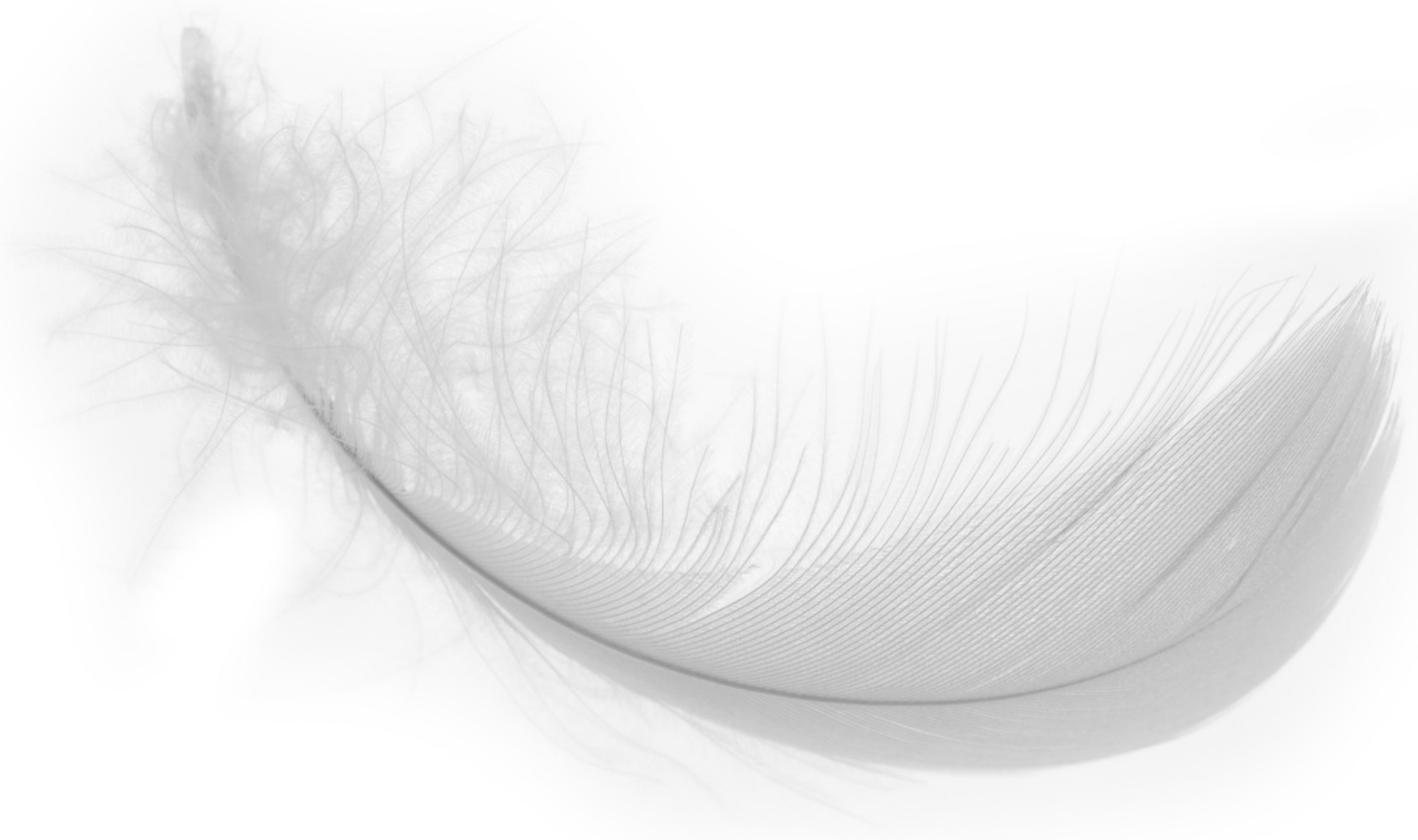


MICRO SERVICES







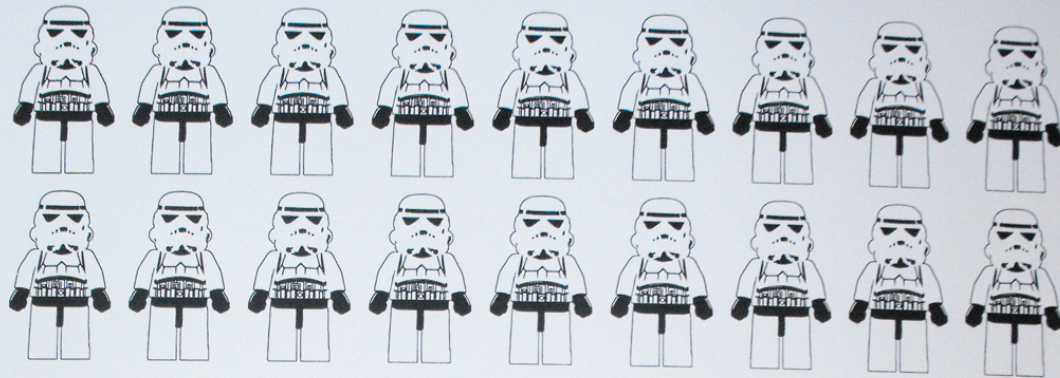




© Disney/Pixar

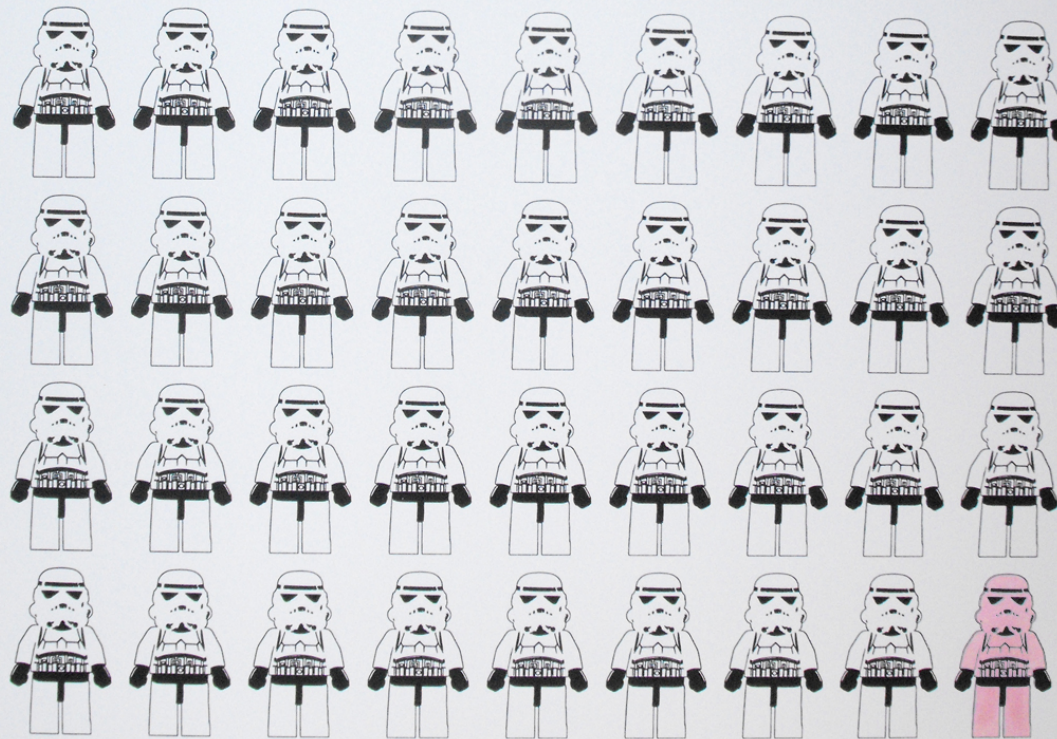






UNIFORMITY

THE BUILDING BLOCK OF AN EMPIRE



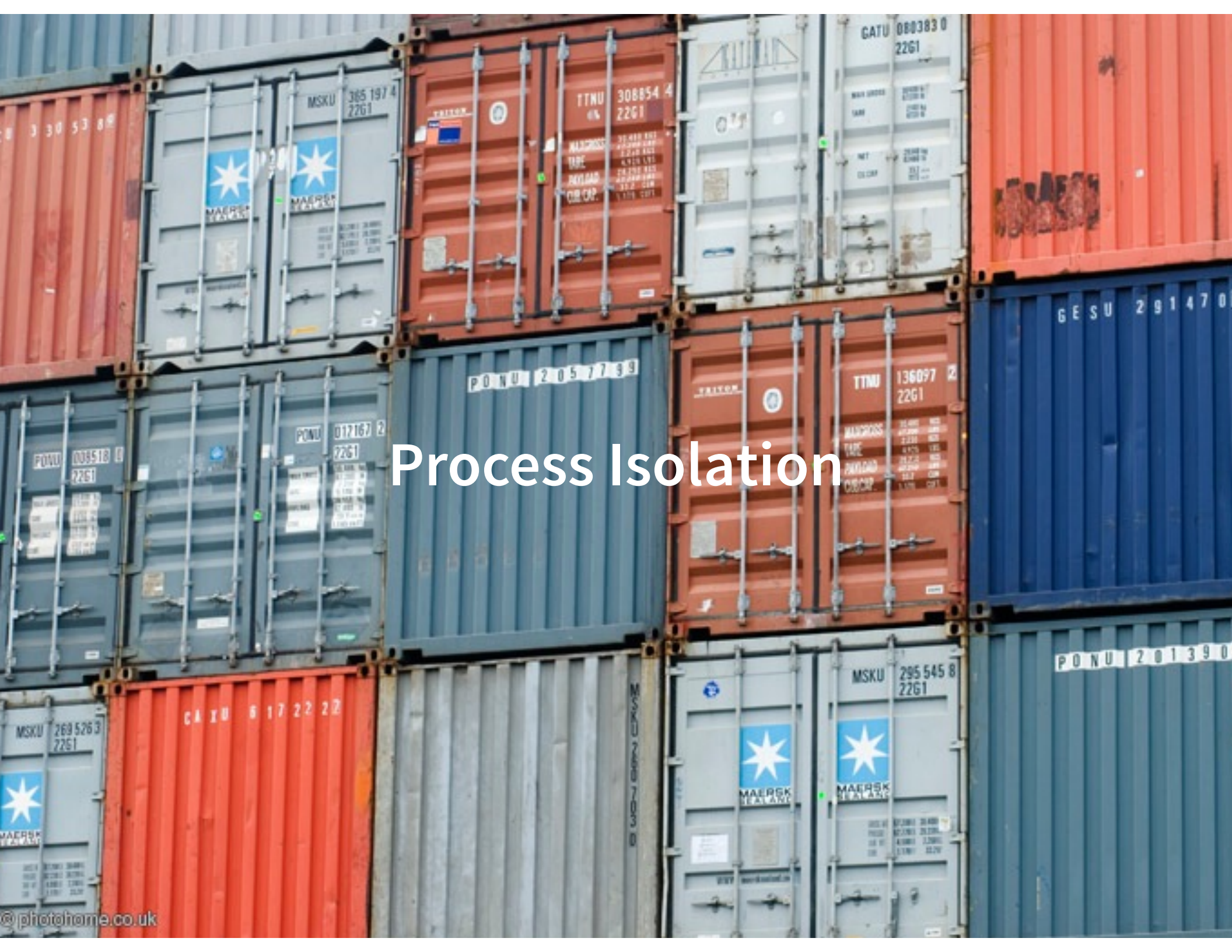
1/

Handwritten signature

- Peak Hours
 - Web Traffic 75%
 - Analytics 25%

- Off Peak Hours
 - Web Traffic 25%
 - Analytics 75%





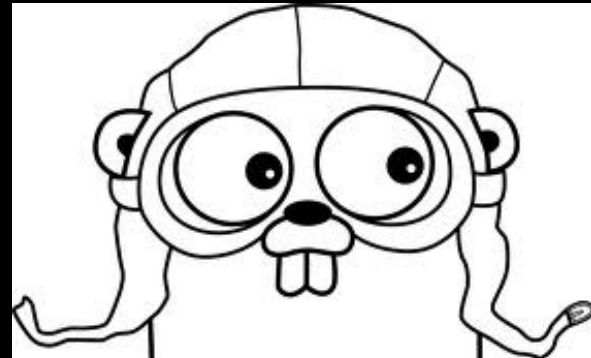
Process Isolation



docker

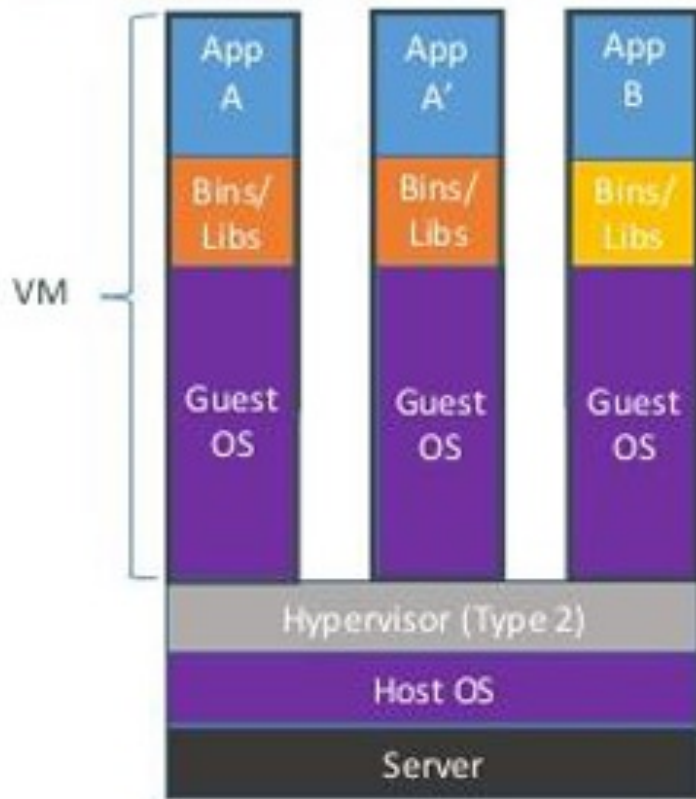


docker

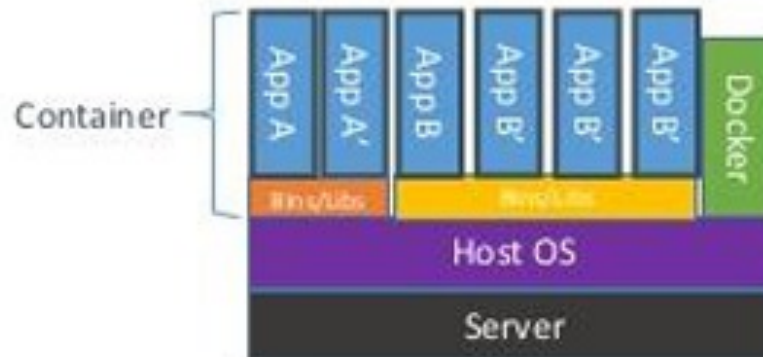


Containers

Containers vs. VMs



Containers are isolated, but share OS and, where appropriate, bins/libraries



Platforms:

Ubuntu



Arch Linux



Gentoo



Fedora



openSUSE



FrugalWare



From Binaries

```
64 00 75 00
00 00 00 00
72 00 6f 00
72 00 73 00
30 74 5f 63
```

Provisioning instructions:

MacOS



Windows



Amazon EC2



Rackspace



Google Cloud



Containers

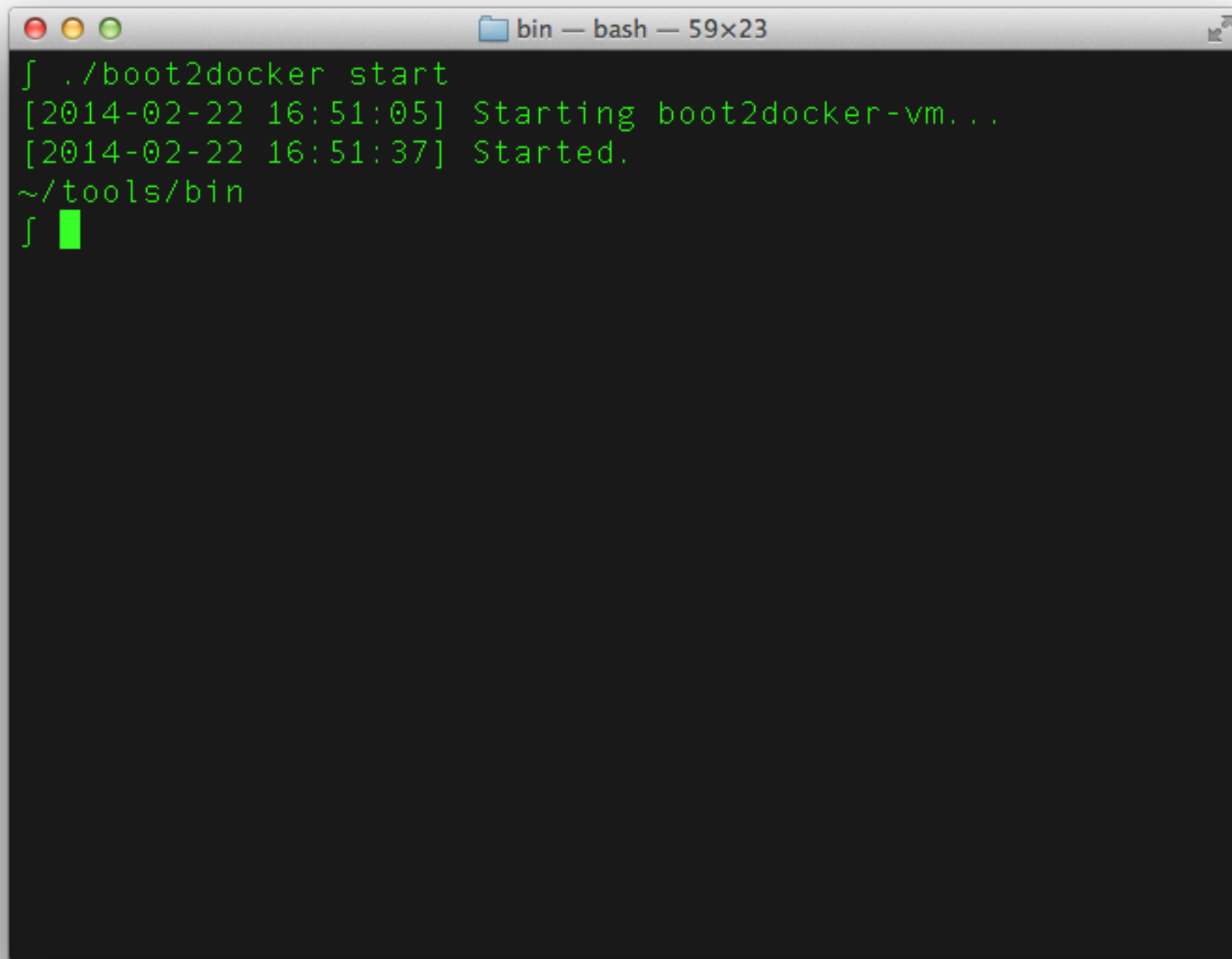
VirtualBox

boot2docker

docker OS X client



boot2docker



```
bin — bash — 59x23
└─$ ./boot2docker start
[2014-02-22 16:51:05] Starting boot2docker-vm...
[2014-02-22 16:51:37] Started.
~/tools/bin
└─$ █
```

Docker Version

```
bin — bash — 59x23
└─ docker version
Client version: 0.8.1
Go version (client): go1.2
Git commit (client): a1598d1
Server version: 0.8.1
Git commit (server): a1598d1
Go version (server): go1.2
~/tools/bin
└─ █
```

Docker Containers

Container Lifecycle

- docker run
- docker stop
- docker start
- docker restart
- docker kill
- docker attach

Container Info

- `docker ps`
- `docker inspect`
- `docker logs`
- `docker port`
- `docker top`
- `docker diff`

Docker Run (port map)

```
bin — bash — 59x23
└─┬─ docker run -d -p 5000 stackbrew/registry
  14dd17cc3233010fd13be208681057a271e6155659e3e1869e2be78e5c0
  97d3e
  ~/tools/bin
└─┬─ docker ps
  CONTAINER ID          IMAGE                COMMAND
  CREATED              STATUS              PORTS
  NAMES
  14dd17cc3233         stackbrew/registry:0.6.1  /bin/sh -c c
  d /docke    13 seconds ago      Up 11 seconds      0.0.0.0:
  49153->5000/tcp    silly_newton
  ~/tools/bin
└─┬─
```

Docker Demo: Running

Docker Images

Image Lifecycle

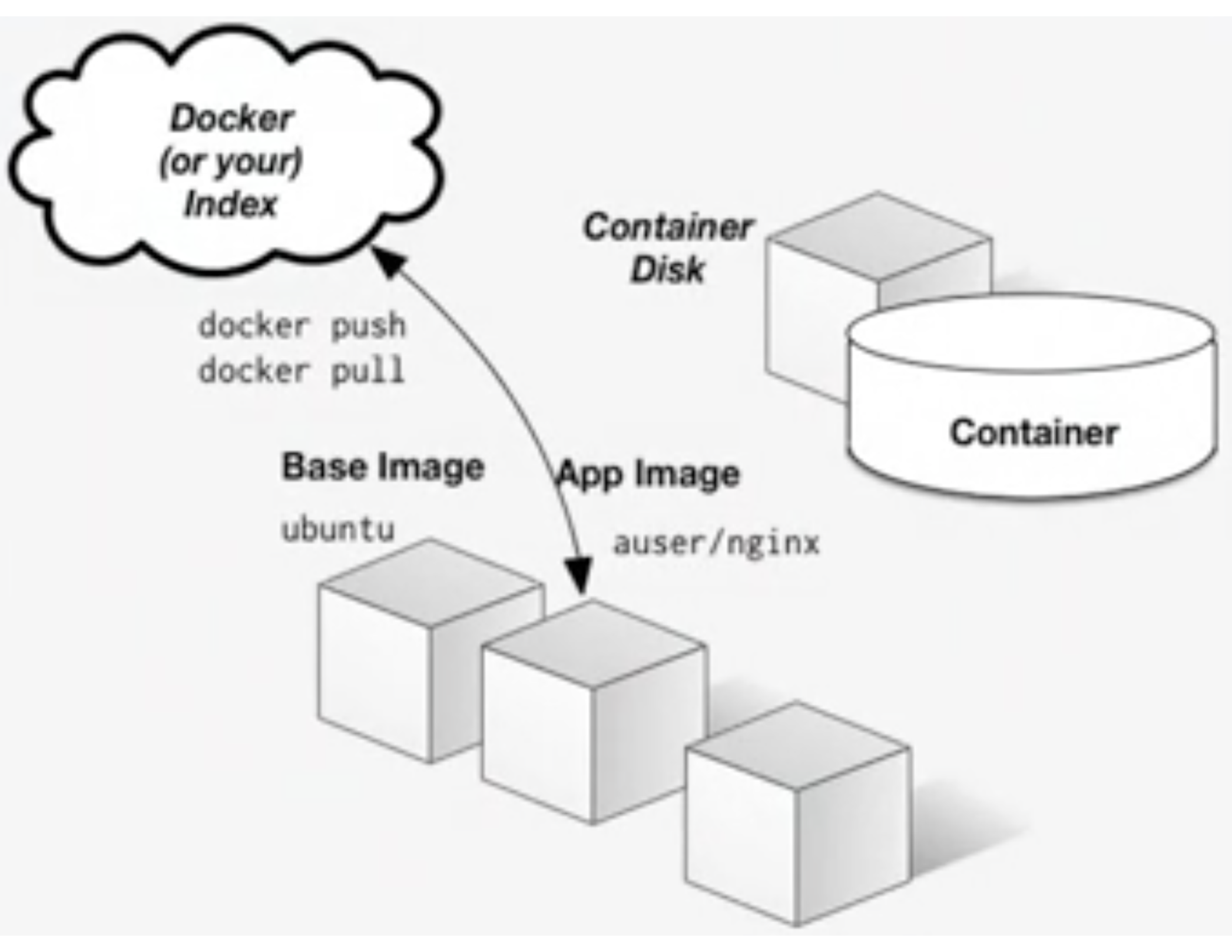
- docker images
- docker build
- docker commit
- docker rmi
- docker tag

Docker Registry

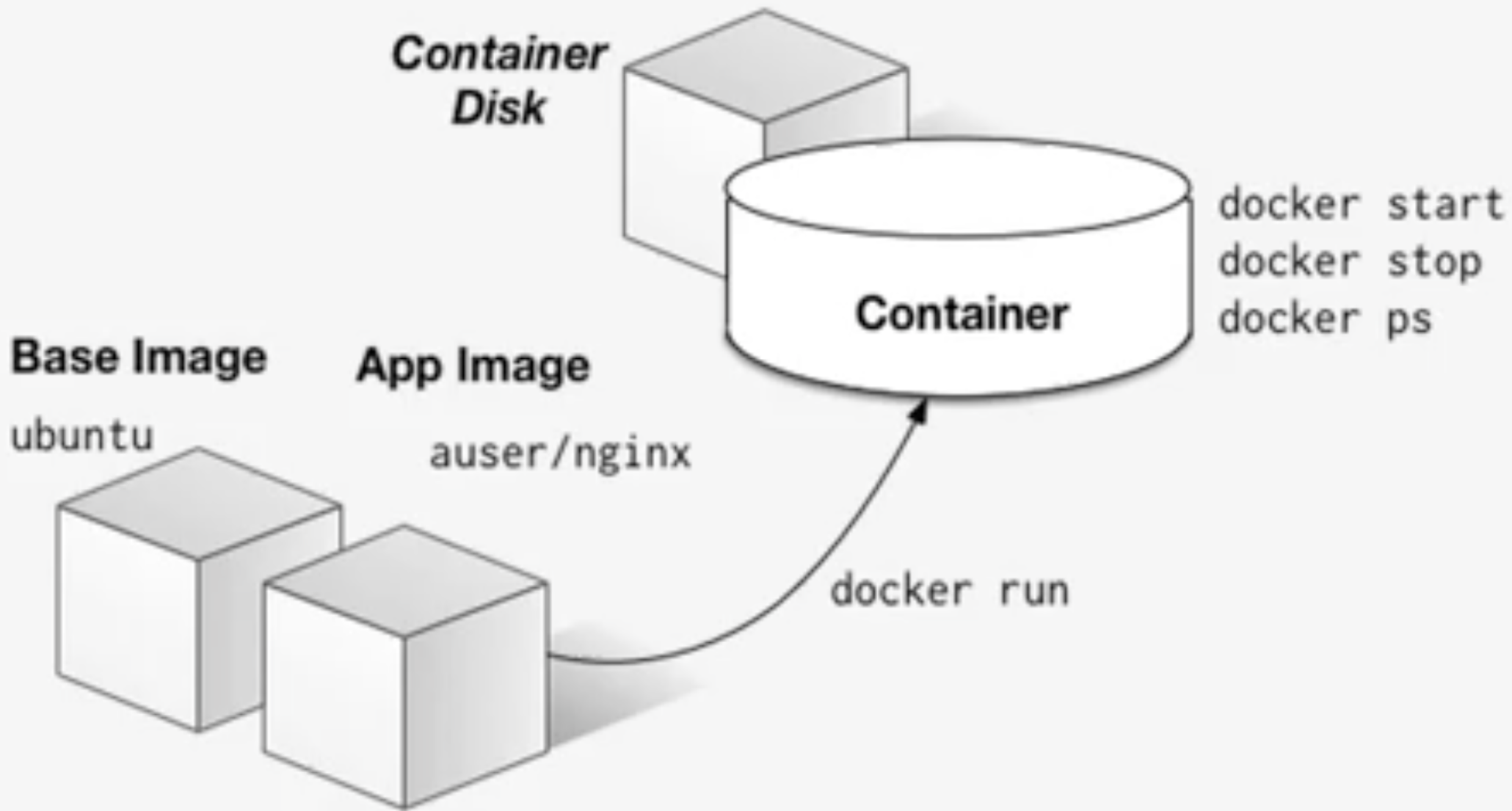
- docker login
- docker search
- docker pull
- docker push

Docker Pull

```
bin — docker — 59x23
] docker pull stackbrew/registry
Pulling repository stackbrew/registry
3321c2caa0cf: Pulling image (latest) from stackbrew/registr
3321c2caa0cf: Pulling image (latest) from stackbrew/registr
y, endpoint: https://cdn-registry-1.docker.io/v1/
ddba540fb44c: Pulling image (0.5.9) from stackbrew/registry
ddba540fb44c: Pulling image (0.5.9) from stackbrew/registry
, endpoint: https://cdn-registry-1.docker.io/v1/
3321c2caa0cf: Pulling dependent layers
ddba540fb44c: Pulling dependent layers
982e72a4385c: Pulling dependent layers
8dbd9e392a96: Download complete
5eaf3a8f1e4d: Downloading 10.91 MB/45.24 MB 16s
511136ea3c5a: Download complete
adfd622eb223: Download complete
9a776d8a79dd: Download complete
4f73ac94d99d: Downloading 6.382 MB/33.47 MB 23s
█
```



Docker Repository

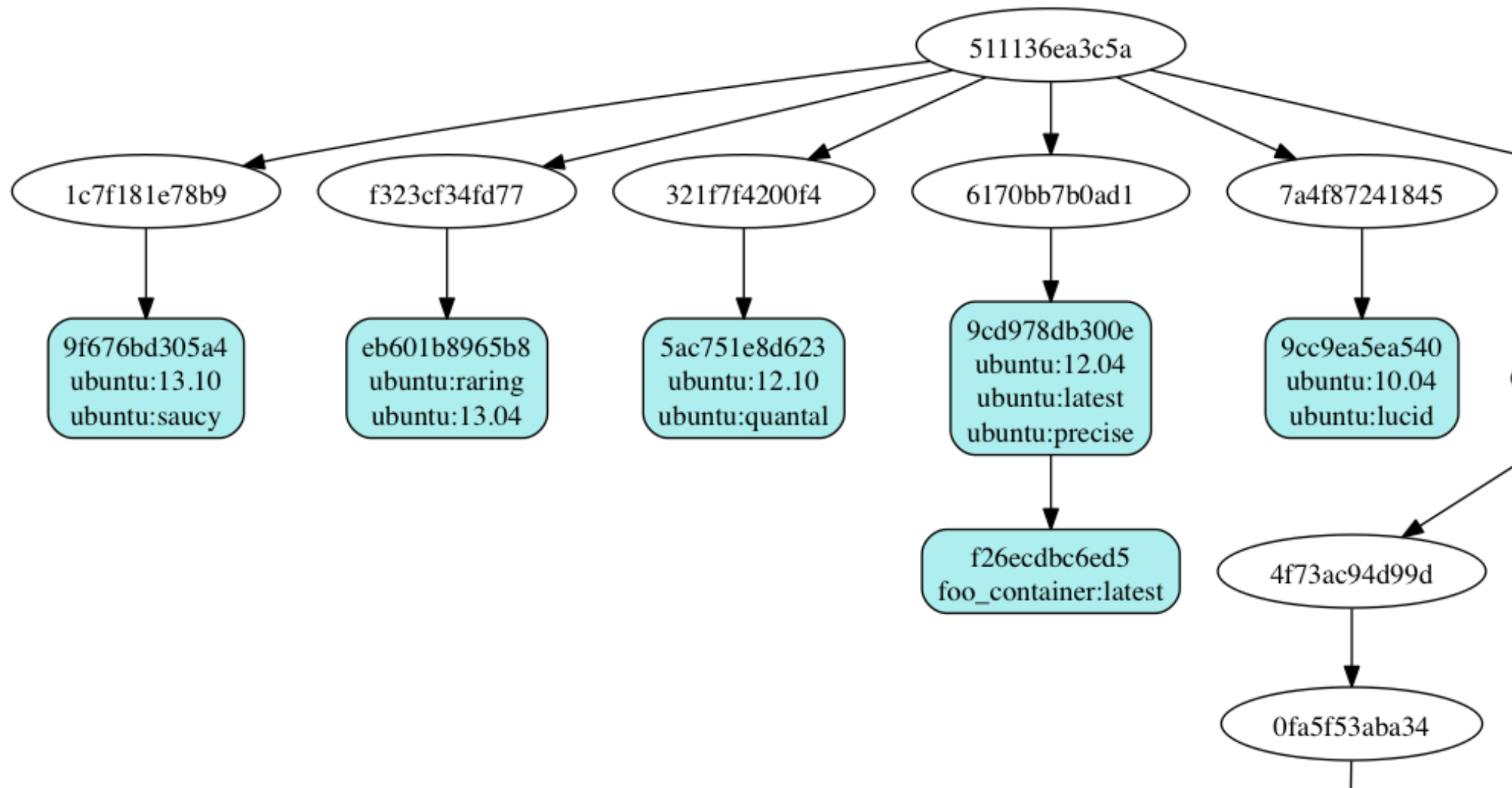


Docker Pull (run couchDB)

```
bin — bash — 59x23
└─┬─ docker pull shykes/couchdb
   │ Pulling repository shykes/couchdb
   │ 018c41c359c3: Download complete
   │ 27cf78414709: Download complete
   │ b750fe79269d: Download complete
   │ 90b456ed4a74: Download complete
   │ c10e80f808d9: Download complete
   │ a5a8870436cc: Download complete
   │ a01acde268b6: Download complete
   └─ ~/tools/bin
     └─ █
```


Docker Image DAGs

```
bin — bash — 53x29
└─ docker images --viz=true | dot -Tpng -o graph.png
~/tools/bin
└─ █
```



Common Docker Exes

Command Prompt in a Docker

```
docker run -it ubuntu /bin/bash
```

```
docker run -it ubuntu
```

Run in Background

```
docker run -d -P redis
```

Common What's Running Commands

- Running
 - `docker ps`
- Just SHA
 - `docker ps -q`
- Last Run
 - `docker ps -l`
- Last Run SHA
 - `docker ps -l -q`

Killing Containers

- With SHA
 - `docker kill <sha>`
- All Running
 - `docker kill $(docker ps -q)`
- Last Run
 - `docker kill $(docker ps -la)`

Demo + Labs

Dockerfiles

```
FROM ubuntu:12.04
```

```
MAINTAINER Quinten Krijger <qkrijger [at] gmail {dot} com>
```

```
# make sure the package repository is up to date
```

```
RUN echo "deb http://archive.ubuntu.com/ubuntu precise main universe" > /etc/apt/sources.list
```

```
RUN apt-get update && apt-get -y install python-software-properties
```

```
RUN add-apt-repository ppa:webupd8team/java
```

```
RUN apt-get update && apt-get -y upgrade
```

```
# automatically accept oracle license
```

```
RUN echo oracle-java7-installer shared/accepted-oracle-license-v1-1 select true | /usr/bin/debconf-set-selections
```

```
# and install java 7 oracle jdk
```

```
RUN apt-get -y install oracle-java7-installer && apt-get clean
```

```
RUN update-alternatives --display java
```

```
RUN echo "JAVA_HOME=/usr/lib/jvm/java-7-oracle" >> /etc/environment
```

1. MAINTAINER

2. RUN

3. ADD

4. CMD

5. EXPOSE

6. ENTRYPOINT:

7. WORKDIR

8. ENV

9. USER

10. VOLUME

MAINTAINER <author name>

RUN <command>

ADD <src> <destination>

CMD ["executable", "param 1", "param2"]

EXPOSE <port>;

ENTRYPOINT ['executable', 'param 1', 'param2']

WORKDIR /path/to/workdir

ENV <key> <value>

USER <uid>

VOLUME ['/data']

```
FROM ubuntu:12.04
```

```
MAINTAINER Quinten Krijger <qkrijger [at] gmail {dot} com>
```

```
# make sure the package repository is up to date
```

```
RUN echo "deb http://archive.ubuntu.com/ubuntu precise main universe" > /etc/apt/sources.list
```

```
RUN apt-get update && apt-get -y install python-software-properties
```

```
RUN add-apt-repository ppa:webupd8team/java
```

```
RUN apt-get update && apt-get -y upgrade
```

```
# automatically accept oracle license
```

```
RUN echo oracle-java7-installer shared/accepted-oracle-license-v1-1 select true | /usr/bin/debconf-set-selections
```

```
# and install java 7 oracle jdk
```

```
RUN apt-get -y install oracle-java7-installer && apt-get clean
```

```
RUN update-alternatives --display java
```

```
RUN echo "JAVA_HOME=/usr/lib/jvm/java-7-oracle" >> /etc/environment
```

Dockerfile

```
1 FROM csobuild/java-1.7
2
3 MAINTAINER Steven Borrelli <steve@borrelli.org>
4
5 RUN echo "deb http://security.ubuntu.com/ubuntu precise-security main universe" >>
6 /etc/apt/sources.list
7
8 RUN apt-get update
9
10 RUN apt-get -y install tomcat7
11
12 RUN echo "JAVA_HOME=/usr/lib/jvm/java-7-oracle" >> /etc/default/tomcat7
13
14 #Remove the default container
15 RUN rm -rf /var/lib/tomcat7/webapps/ROOT /var/lib/tomcat7/webapps/ROOT.war
16
17 EXPOSE 8080
18
19 CMD service tomcat7 start && tail -f /var/lib/tomcat7/logs/catalina.out
```

Line: 13 Column: 19

Plain Text

Tab Size: 4

Docker Build File Labs

Mesosphere



What is Mesosphere?

Mesosphere



MESOS



MARATHON

- Mesosphere Chronos
- HAProxy
- Docker Support

What is Mesos?



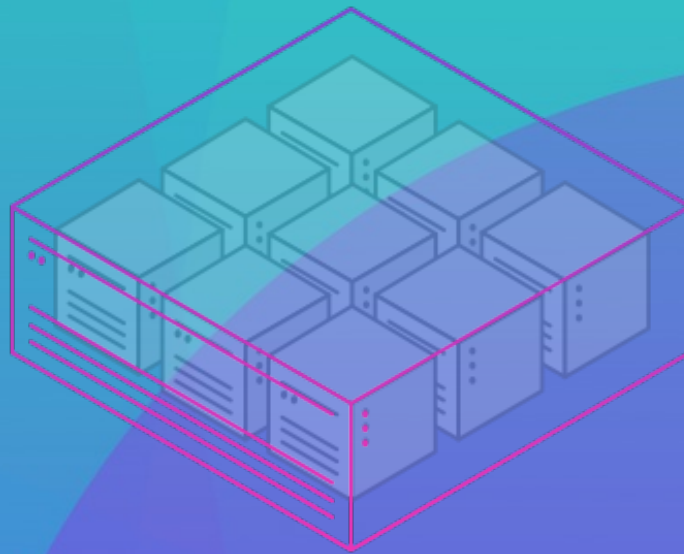
Mesos is...

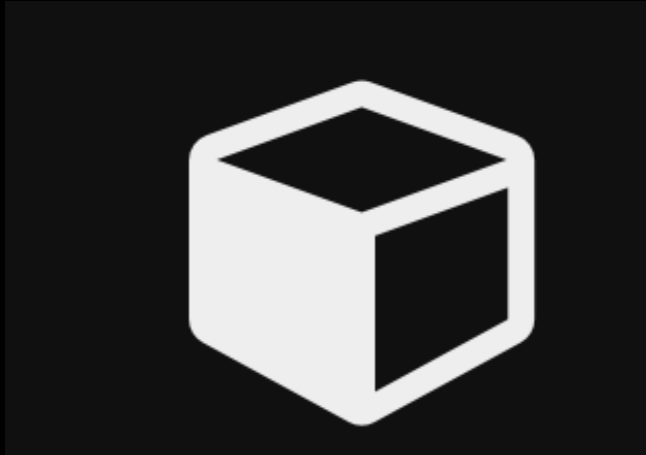
- Open Source Apache project
- Cluster Resource Manager
- Scalable to 10,000 of nodes
- Fault Tolerant

Mesos lets you treat a cluster of nodes...

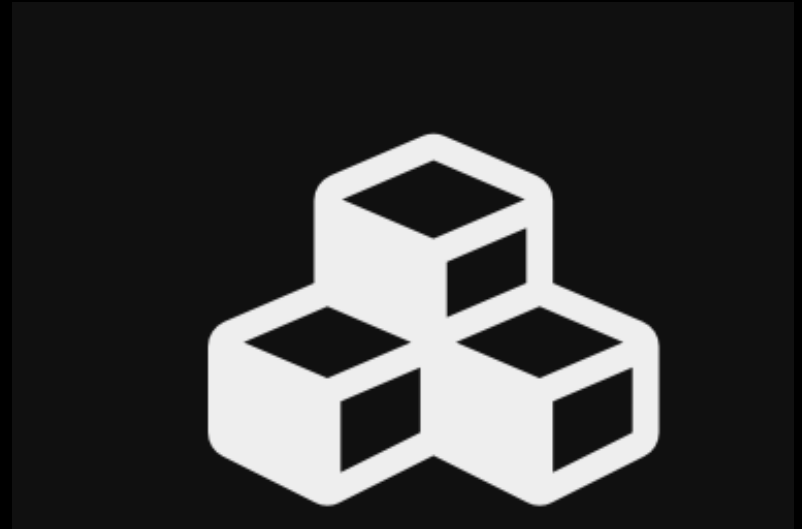


As one big computer





**Not as individual
machines**



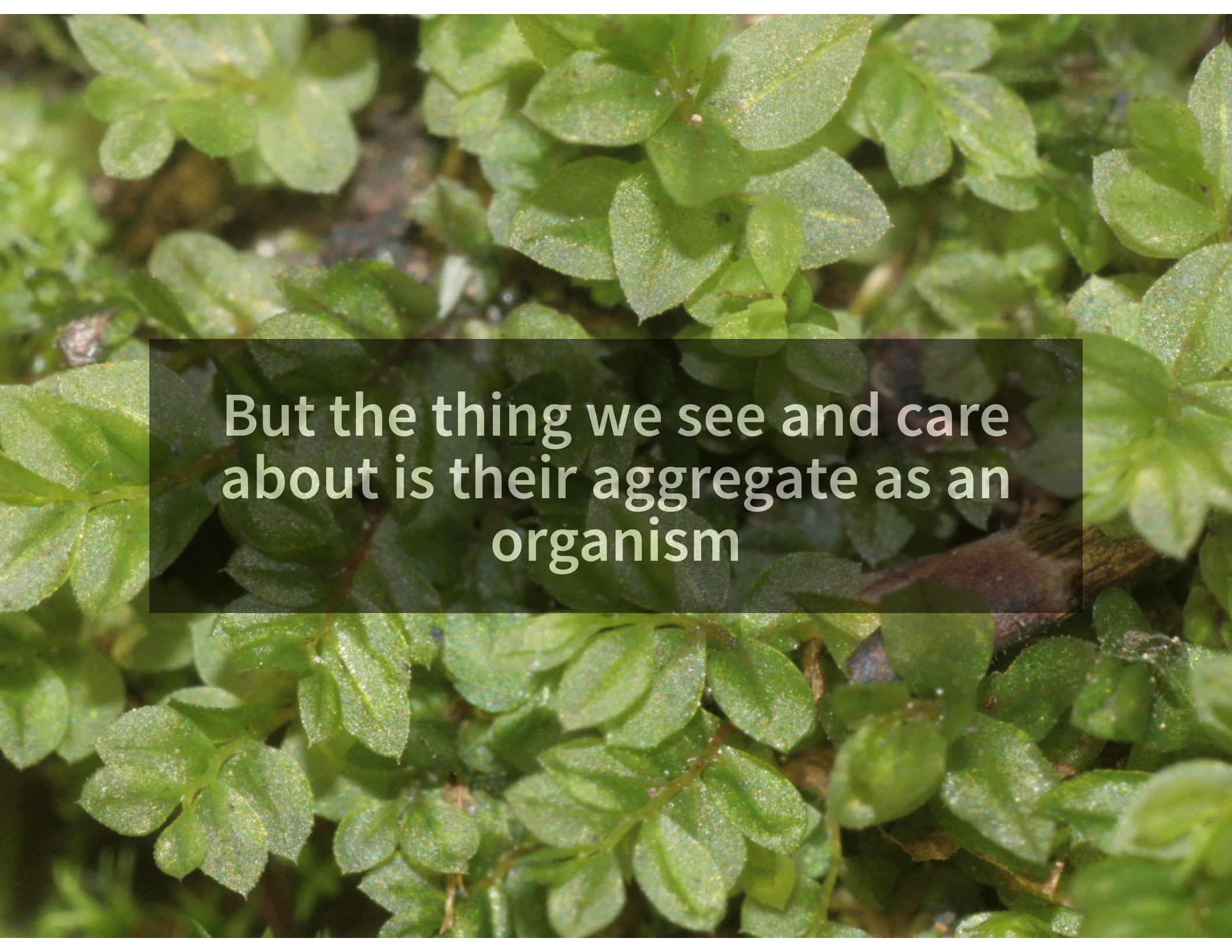
Not as VMs



But as computational resources
like cores, memory, disks, etc.

A microscopic view of plant tissue, likely a leaf cross-section, showing a dense arrangement of green, polygonal cells. The cells are organized into a regular, grid-like pattern, with distinct cell walls visible between them. The overall color is a vibrant green, with some lighter areas where the cells are more densely packed or where the light reflects off the surface. The image is used as a background for a text overlay.

**Just like a cell is a crucial building
block of a larger system**

A close-up photograph of a dense carpet of green moss. The individual plants are small and have several rounded, overlapping leaves. The moss is growing on a dark, textured surface, possibly soil or a rock. The lighting is bright, highlighting the vibrant green color and the fine details of the leaves.

But the thing we see and care
about is their aggregate as an
organism

A close-up photograph of a dense carpet of green moss. The moss consists of numerous small, rounded, leaf-like structures that are tightly packed together, creating a textured, vibrant green surface. The lighting is soft, highlighting the individual clusters and their intricate details. A semi-transparent dark grey rectangular box is overlaid in the center of the image, containing white text.

This is what Mesosphere lets us do
with clusters

“We wanted people to be able to program for the datacenter just like they program for their laptop”

**— Benjamin Hindman,
Apache Mesos PMC Chair**

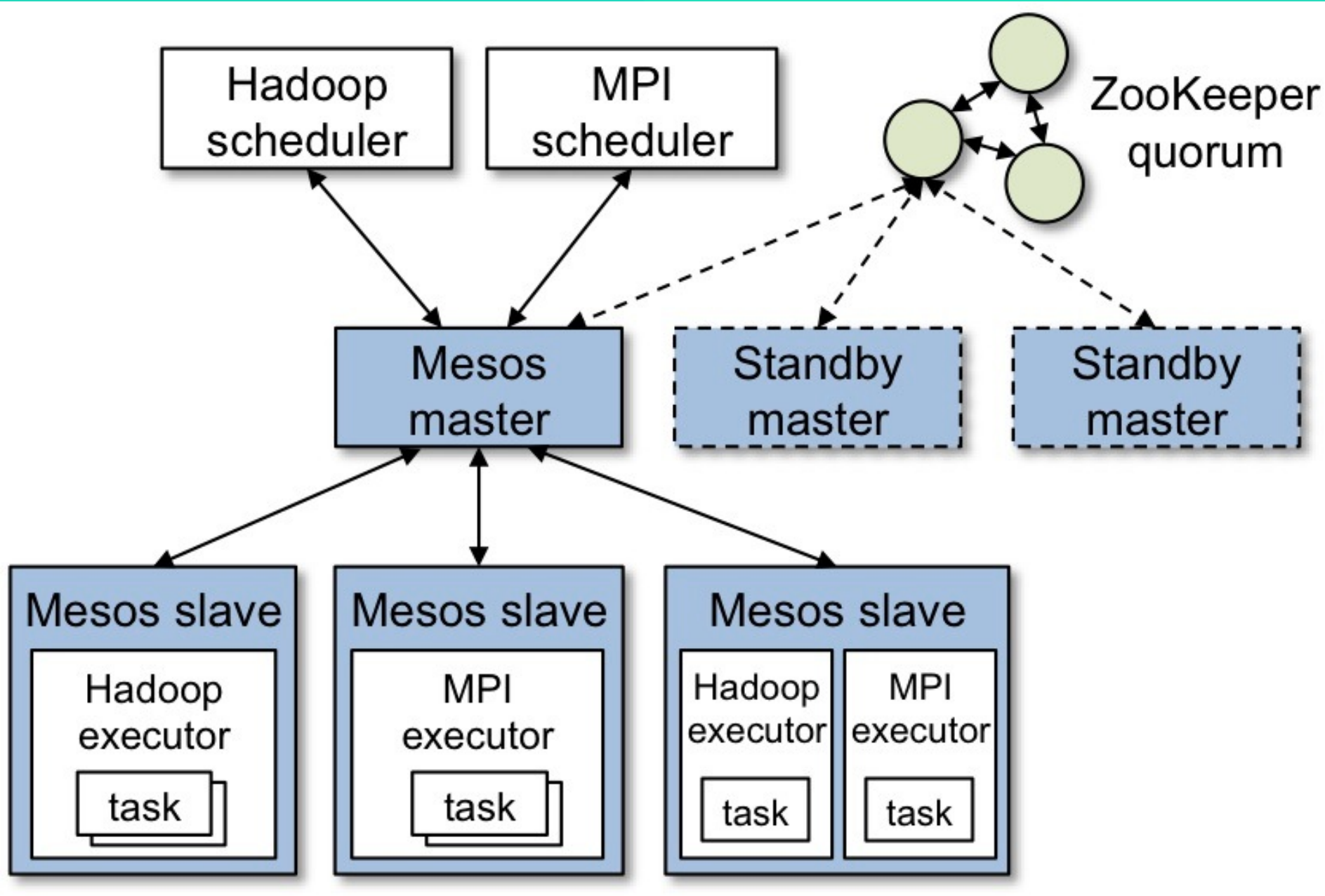


What is Mesos?

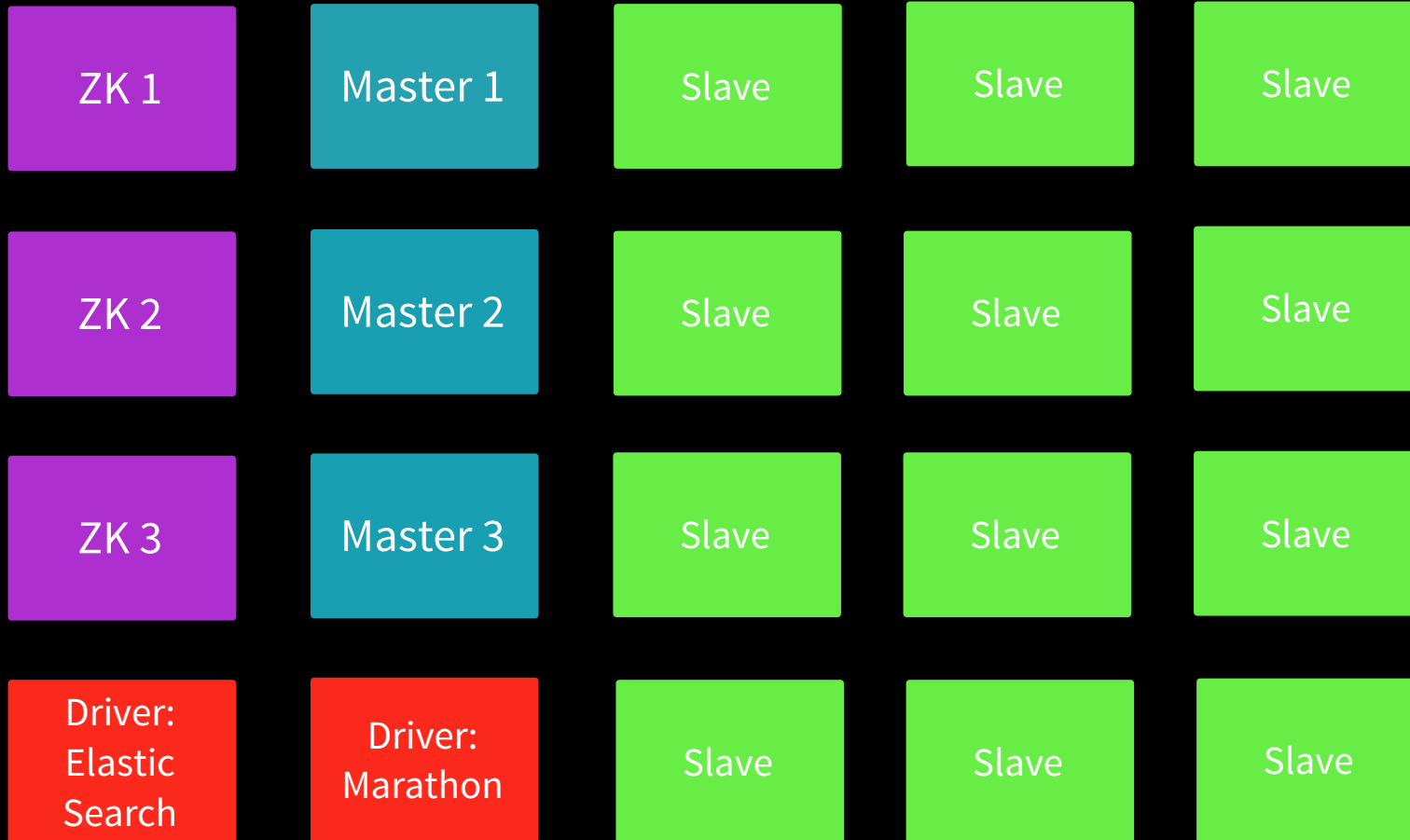


Mesos - Overview

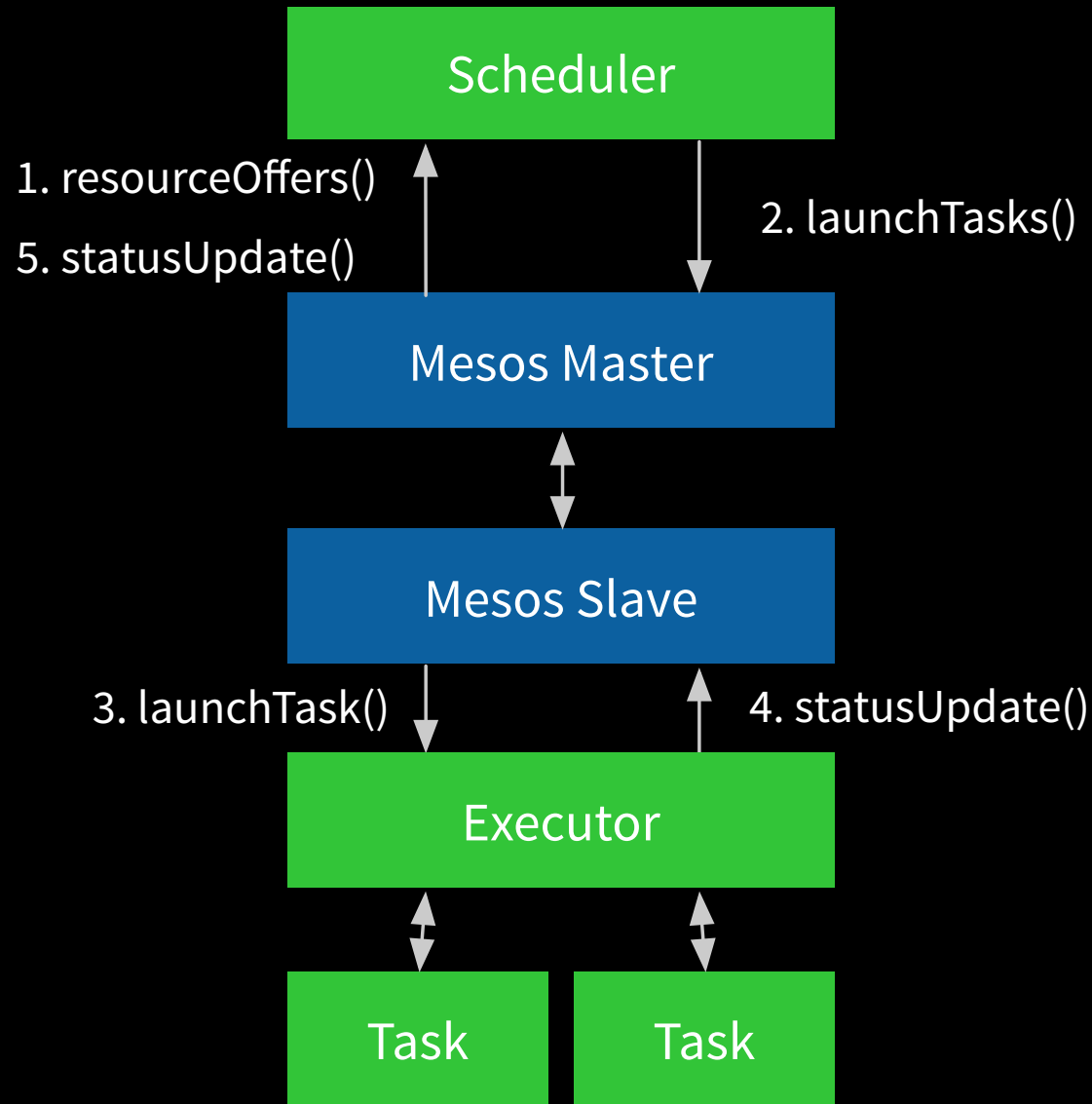
- Mesos Master
- Framework Scheduler (Driver)
- Mesos Slave
 - Isolation, Reporting
 - Framework Executor



Mesos Framework Overview



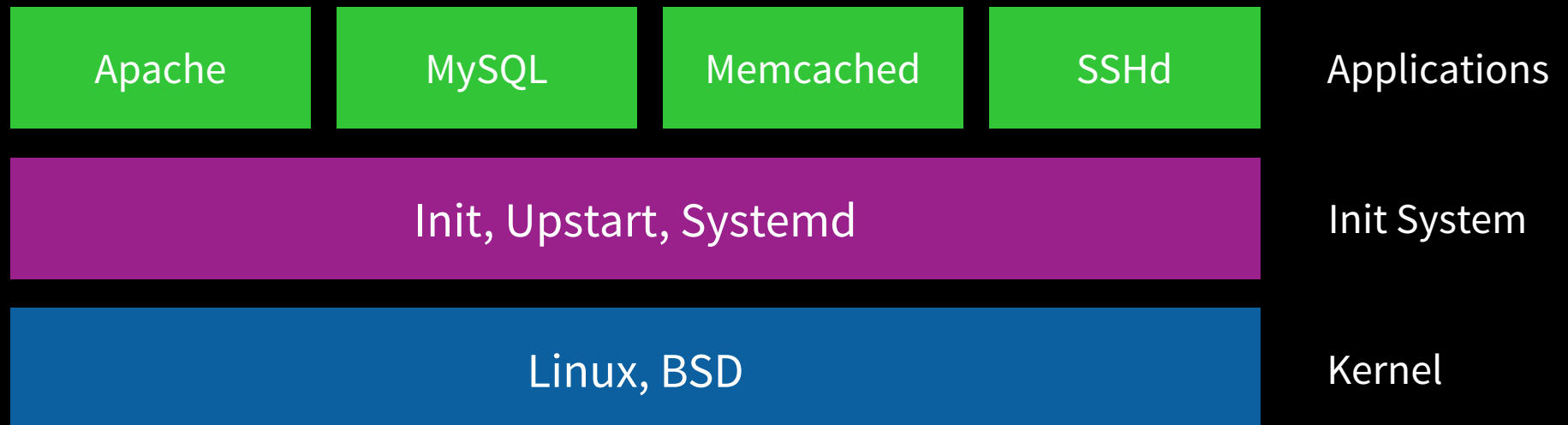
Mesos Framework Components



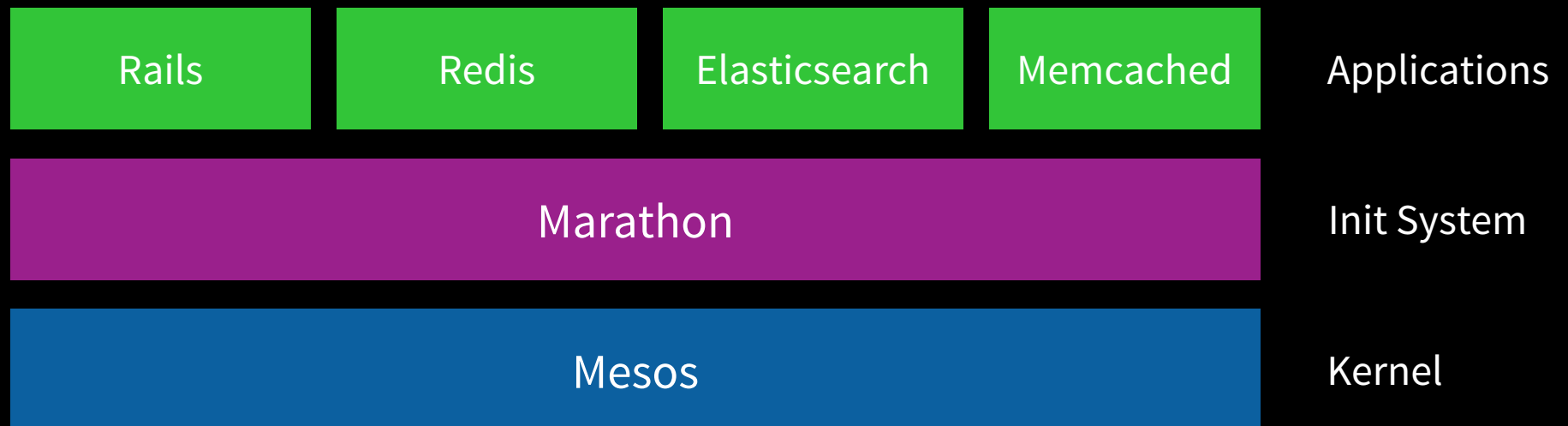
Like an OS kernel
You rarely interact directly with
Mesos ...

Like an OS kernel
You rarely interact directly with
Mesos ...
You interact with Mesos
Frameworks on top of Mesos

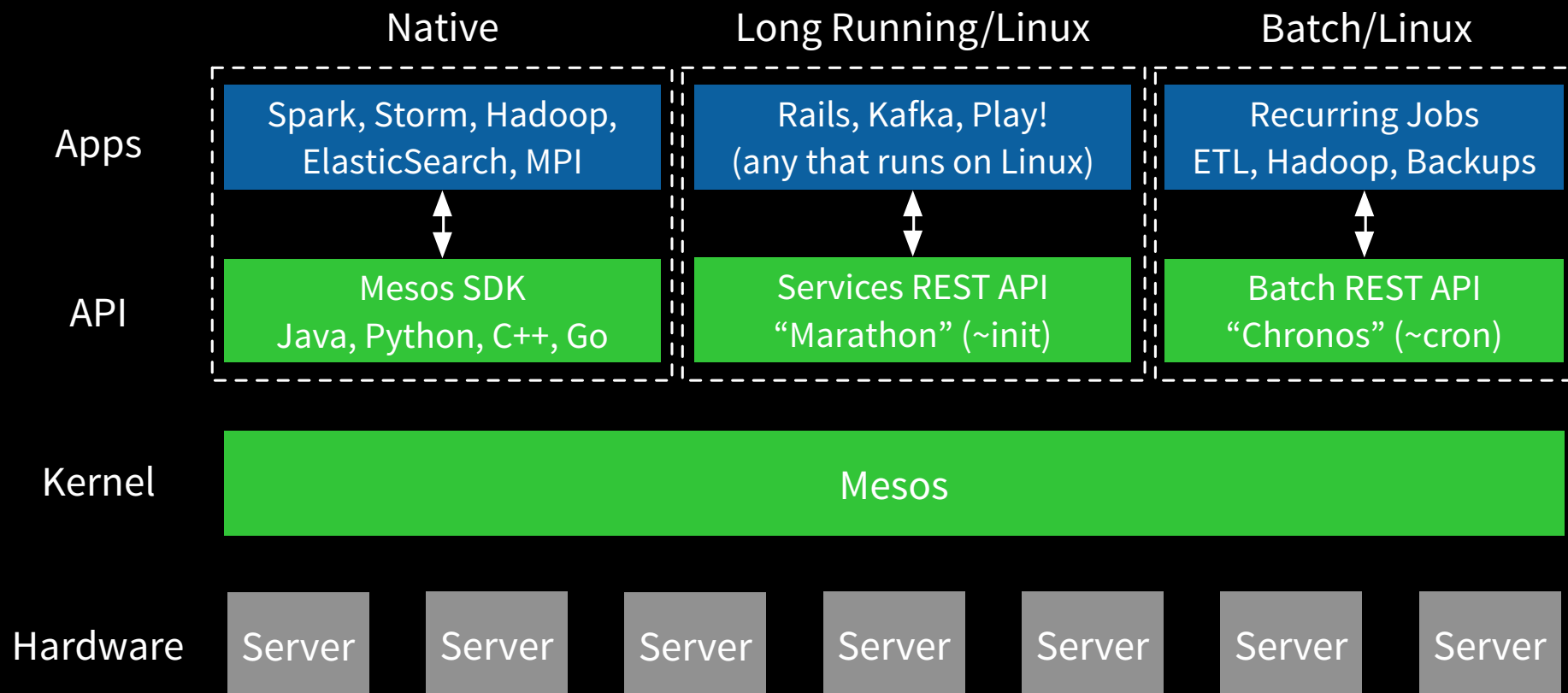
The UNIX Operating System Stack



The Mesosphere Operating System Stack



Mesosphere Stack



Mesosphere Value Propositions

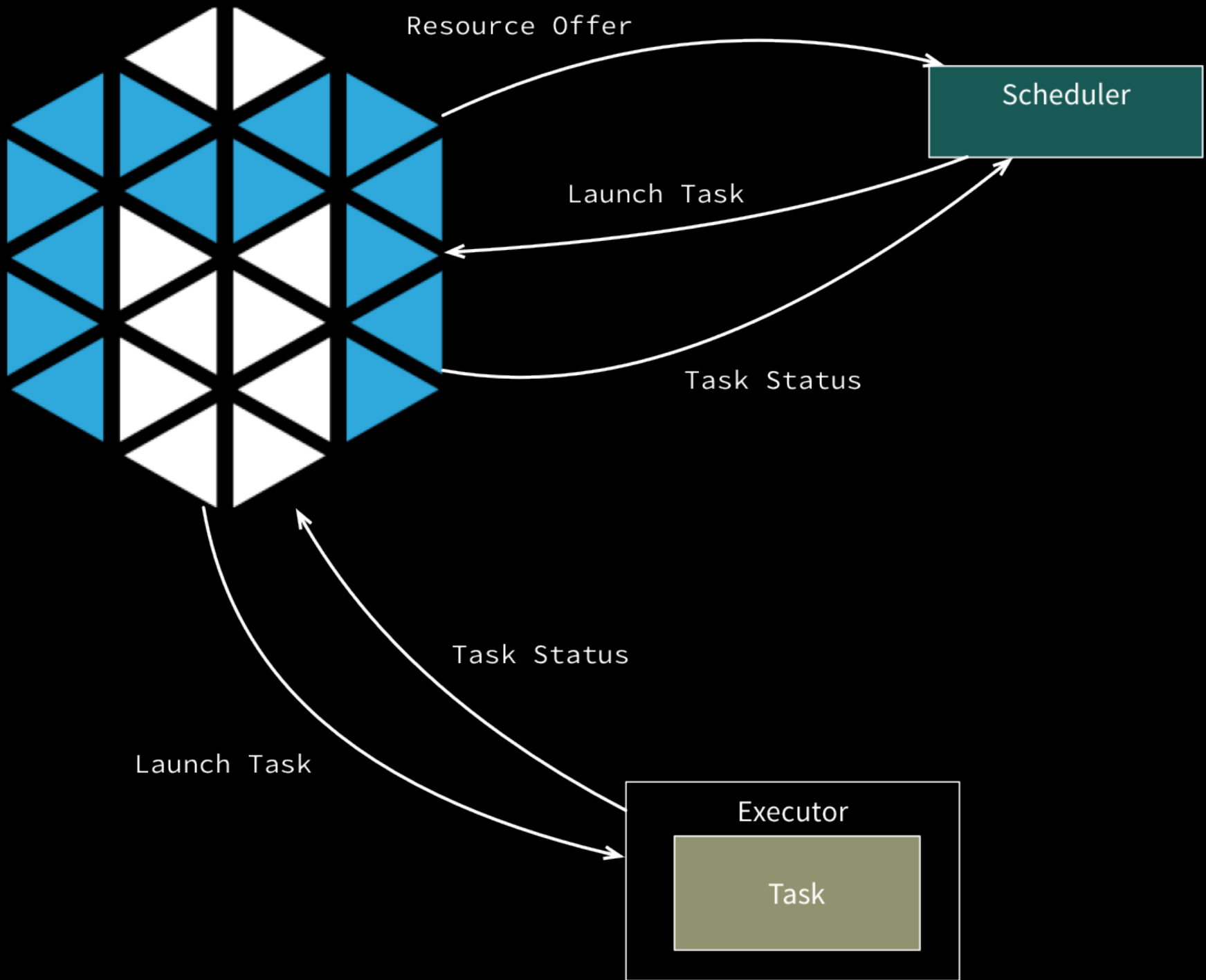
Fault Tolerance through Automatic Software/Hardware Failure Handling



Simplified Operations - Homogenous Layer Supporting Diverse Infrastructure

Up to 65% Hardware Cost Savings

Efficiency and Productivity Gains Enable New Elastic Applications

Mesos Frameworks



Au Aurora	Cc Cray Chapel	Dp Dpark	Ch Chronos	Jk Jenkins	
Ma Marathon	Ex Exelixa	Ha Hadoop	Tq Torque	Ca Cassandra	
Sp SSSP	Mi MPI	Sk Spark	St Storm	Es ElasticSearch	Ht Hypertable

Mesos

Case Study: Airbnb - Chronos



Mesos on 4,000+ cores

Entire Analytics Built on Mesos: Hadoop, Storm, Kafka, Cassandra, ...

Quick Rollout of New Applications

More Automation, Less People to Manage Servers

Apache Spark Framework

- Apache Spark vs. Hadoop
 - 100x faster in memory
 - 10x faster on disk
- Jobs in Java, Scala or Python





MARATHON

What is Marathon?

“Init Daemon” for the data center

- Runs any Linux binary without modification (e.g. Rails, Tomcat, ...)
- Cluster-wide process supervisor

Private PaaS

- Service discovery
- Automated software and hardware failure handling
- Deployment and scaling

ID ^	Command	Memory (MB)	CPUs	Instances
agora	java -jar current.war --httpPort=\$PORT	1024	1	3 / 3
chronos	cd chronos && ./bin/chronos-marathon	385	0.1	1 / 1
liquor-store	./liquorstore-*/bin/liquorstore -Dhttp.port=\$PORT -Dnewrelic.bootstrap_cl...	1024	1	3 / 3
uname	uname -a && sleep 60	16	0.1	3 / 3

ID ^	Command	Memory (MB)	CPUs	Instances
agora	java -jar current.war --httpPort=\$PORT	1024	1	3 / 3
chronos	cd chronos && ./bin/chronos-marathon	385	0.1	1 / 1
liquor-store	./liquorstore-*/bin/liquorstore -Dhttp.port=\$PORT -Dnewrelic.bootstrap_cl...	1024	1	3 / 3
uname	uname -a && sleep 60	16	0.1	3 / 3

Writing your own Framework

- Sophisticated scheduling algorithms and policies
- Sophisticated scale policies
- Advanced task semantics

Case Study: Large Hedge Fund

- Trading algorithms on Mesos
- 20,000 cores
- Better algorithm gets better weighted processing time!

Mesosphere & Marathon in Action

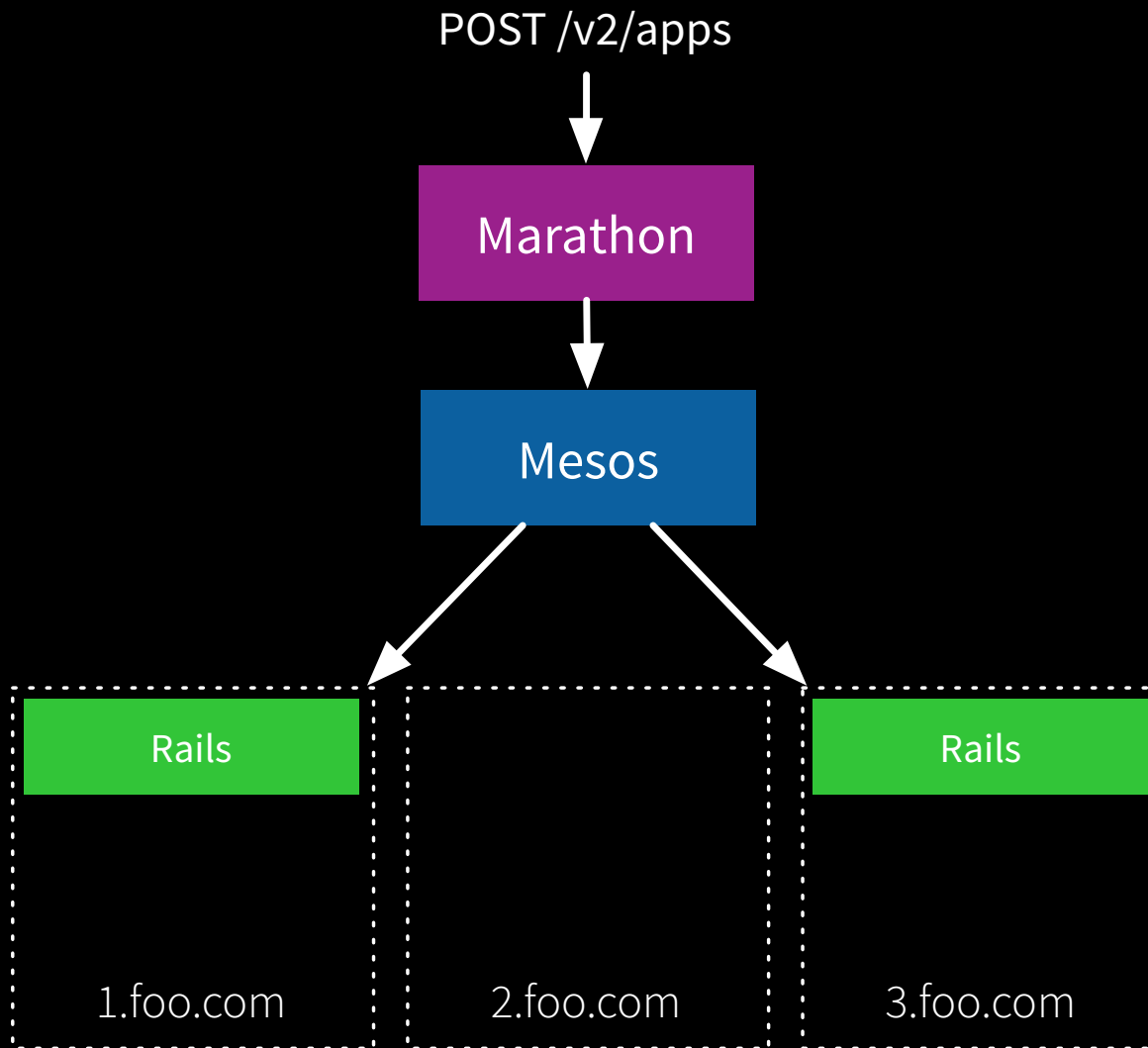
What does Mesos Do for Me?

- Task distribution, launching, monitoring, failure detection, killing and cleanup
- Resource Isolation with containers

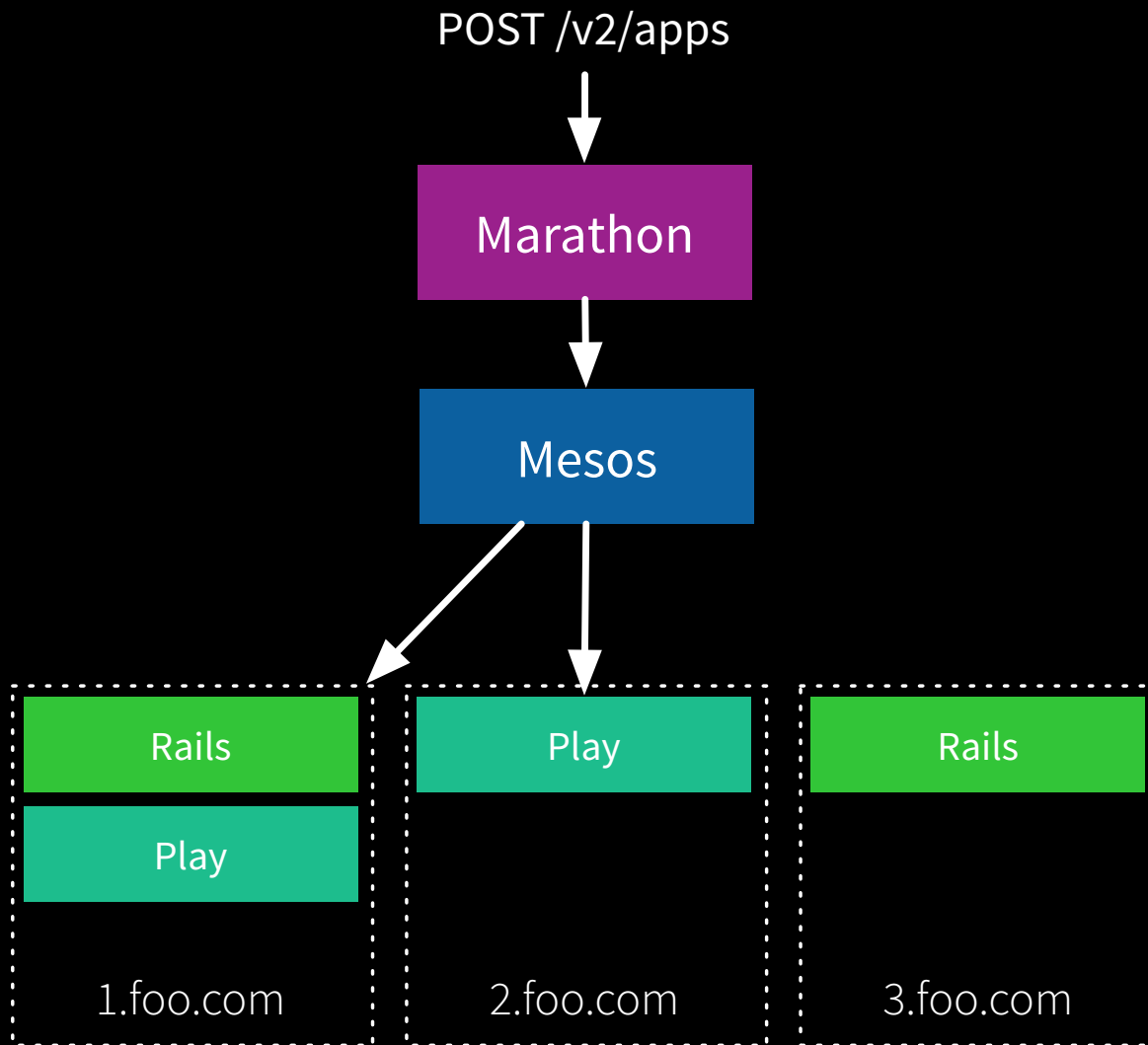


MARATHON

Marathon Workflow



Marathon Workflow



Marathon API - Launching Self-Contained Apps

- Command to start the app
- URL(s) to the app archive/configuration
- Environment variables

POST /v2/apps

```
{  
  "id": "Play",  
  "uris": ["http://downloads.mesosphere.io/tutorials/PlayHello.zip"]  
  "cmd": "./Hello-*/bin/hello -Dhttp.port=$PORT",  
  "env": {"SECRET": "password123"}  
}
```

Marathon API - Launching Dockers

- Starting with Mesos 0.20 containers are 1st class citizens

POST /v2/apps

```
{
  "id": "Cassandra",
  "container": {
    "image": "docker:///mesosphere/cassandra:2.0.6",
    "options": ["-v", "/mnt:/mnt:rw", "-e", "CLUSTER_NAME=prod"]
  }
}
```

Marathon API - Scaling Apps

- Just tell Marathon how many you want!

```
PATCH /v2/apps/Play
```

```
{  
  "instances": 4  
}
```

Marathon Service Discovery Design Goals

Be as simple as connecting to a host and port with TCP

Discovery should happen transparently, don't require special clients

No retry logic required in the client

Registration out-of-band, to support any app without modification

Real-time failover

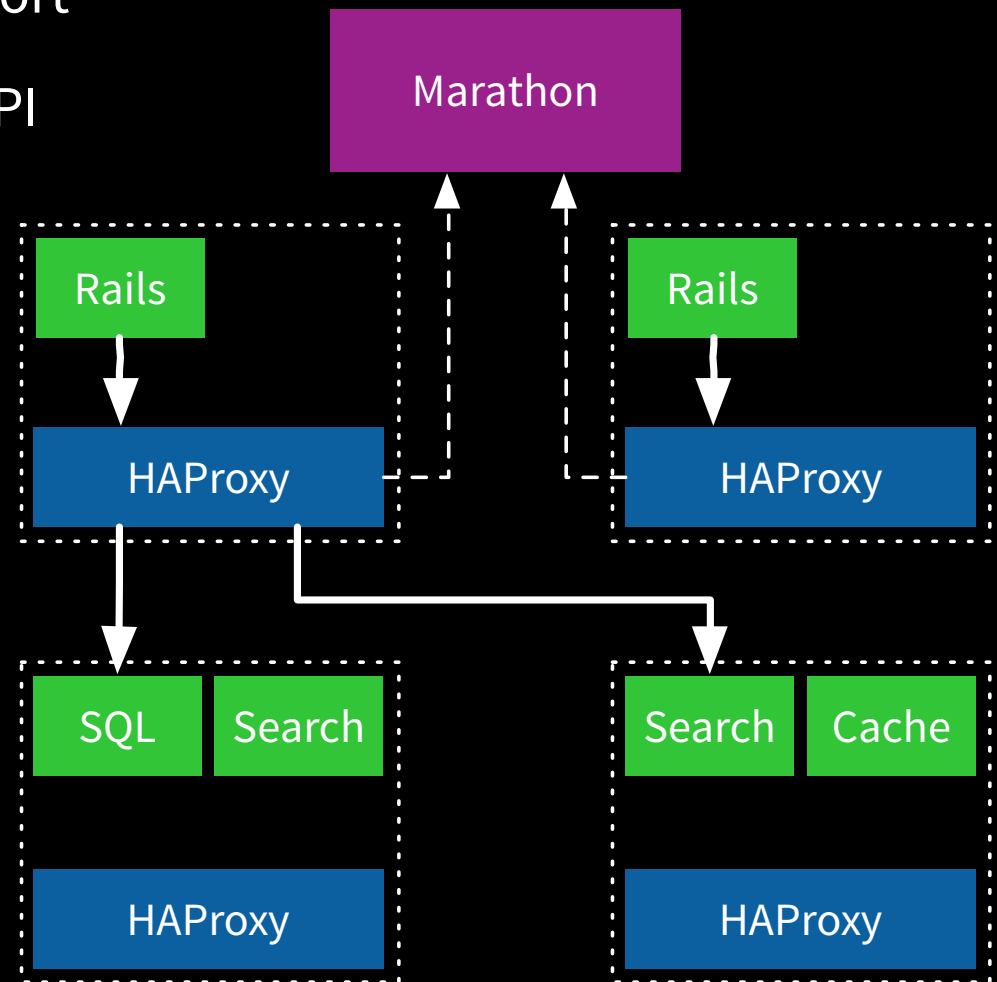
Marathon Service Discovery with HAProxy

Apps available on localhost & known port

HAProxy updates via Marathon REST API

HAProxy runs on every cluster node

Configurable policies



Other Service Discovery Options

Poll the REST API

GET /v2/tasks

This is what HAProxy does

Push via event handlers

Marathon pushes events to any HTTP endpoint

Can be used with hardware load balancers

Marathon Roadmap

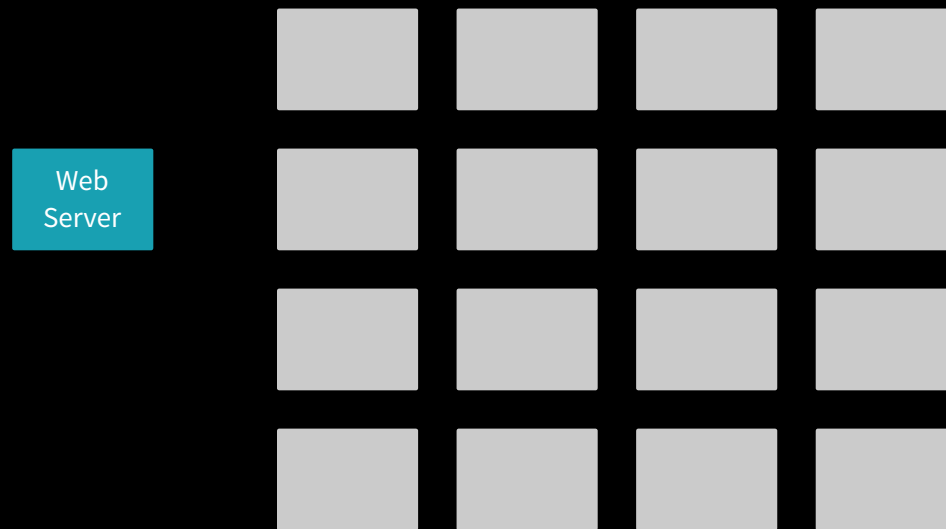
Configurable health checks (TCP, HTTP, HTTPS) [DONE]

App versioning [DONE]

Deployment orchestration features

App groups & dependencies

Elasticity and Resource Sharing



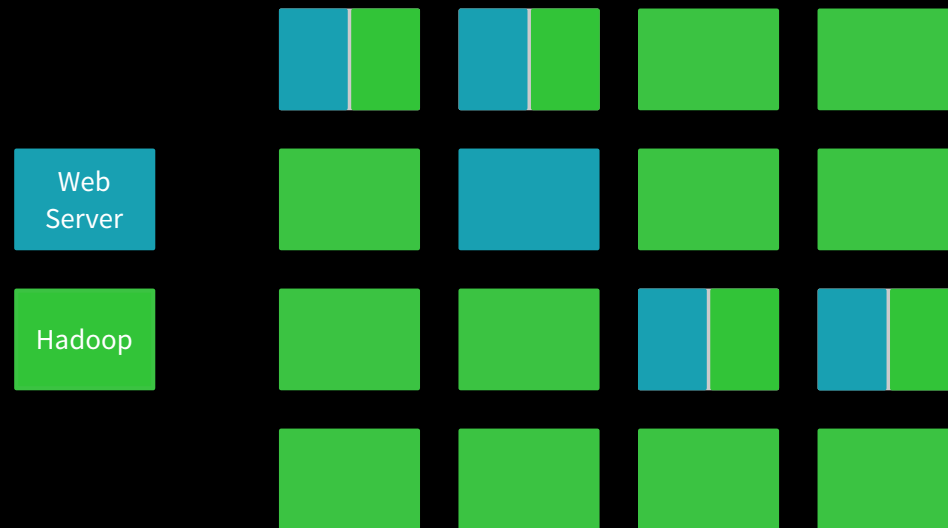
Elasticity and Resource Sharing



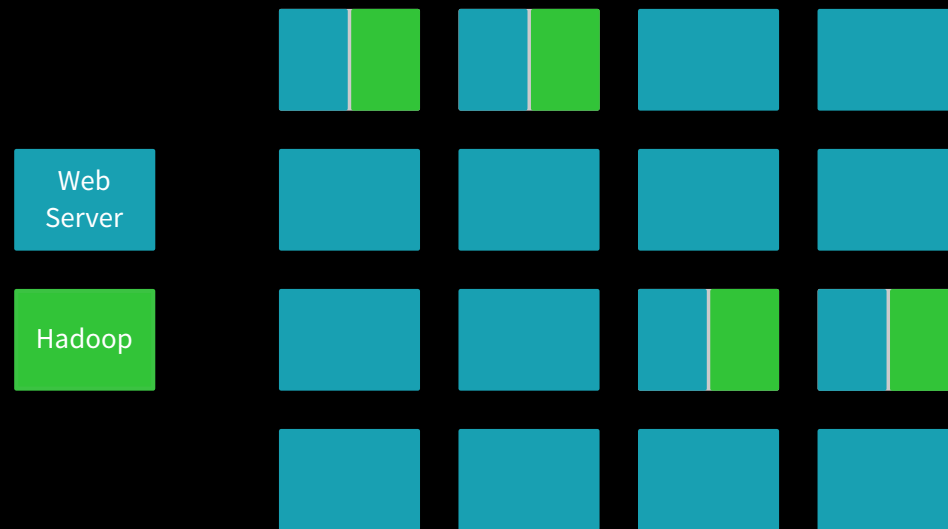
Elasticity and Resource Sharing



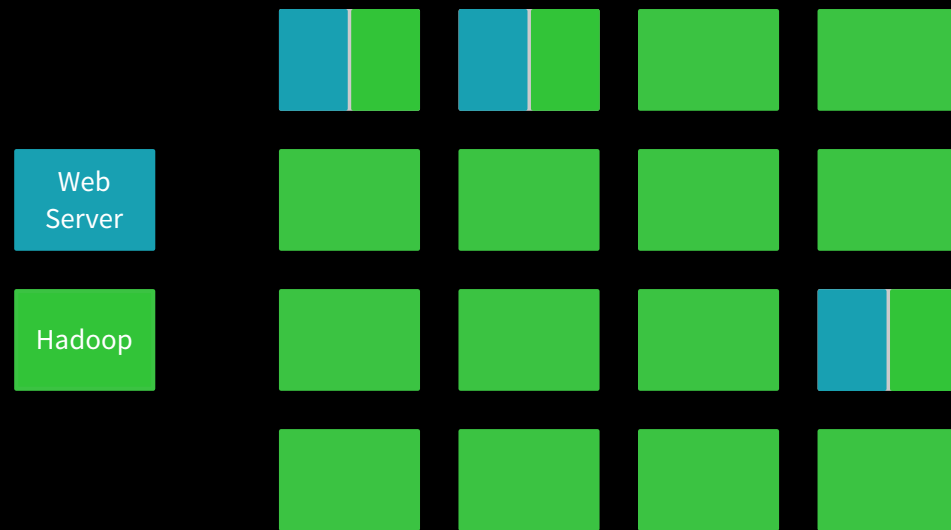
Elasticity and Resource Sharing



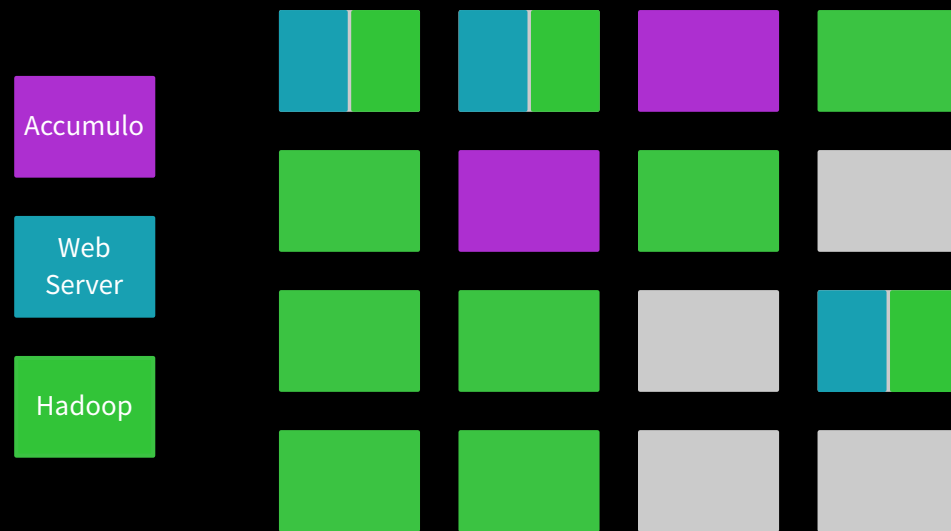
Elasticity and Resource Sharing



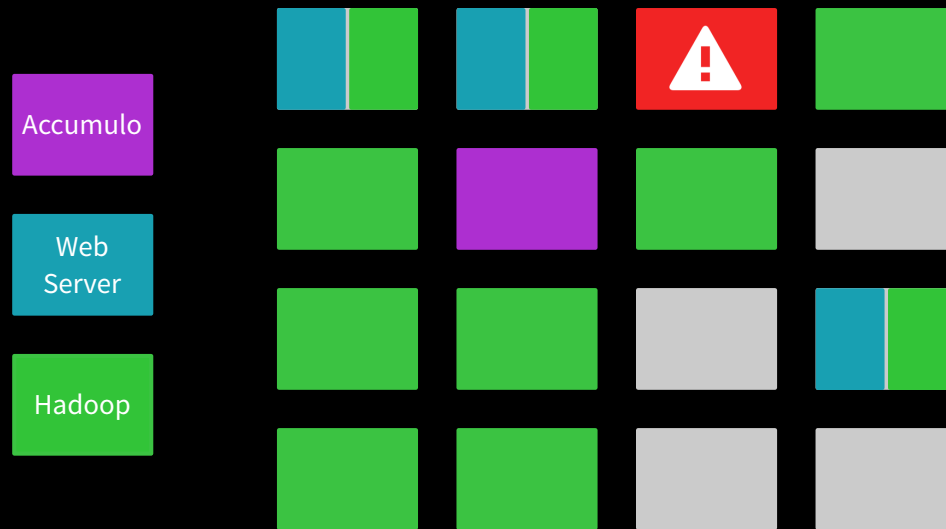
Elasticity and Resource Sharing



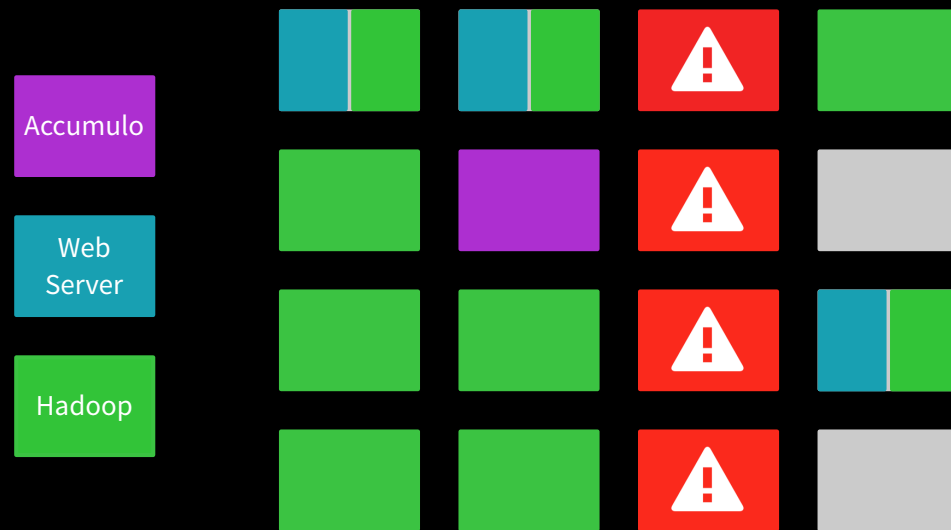
Elasticity and Resource Sharing



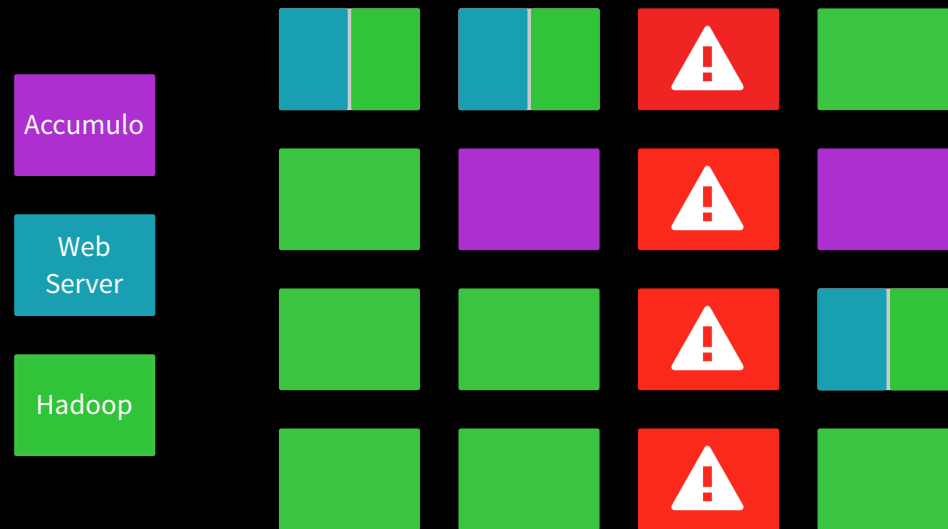
Handling Failure



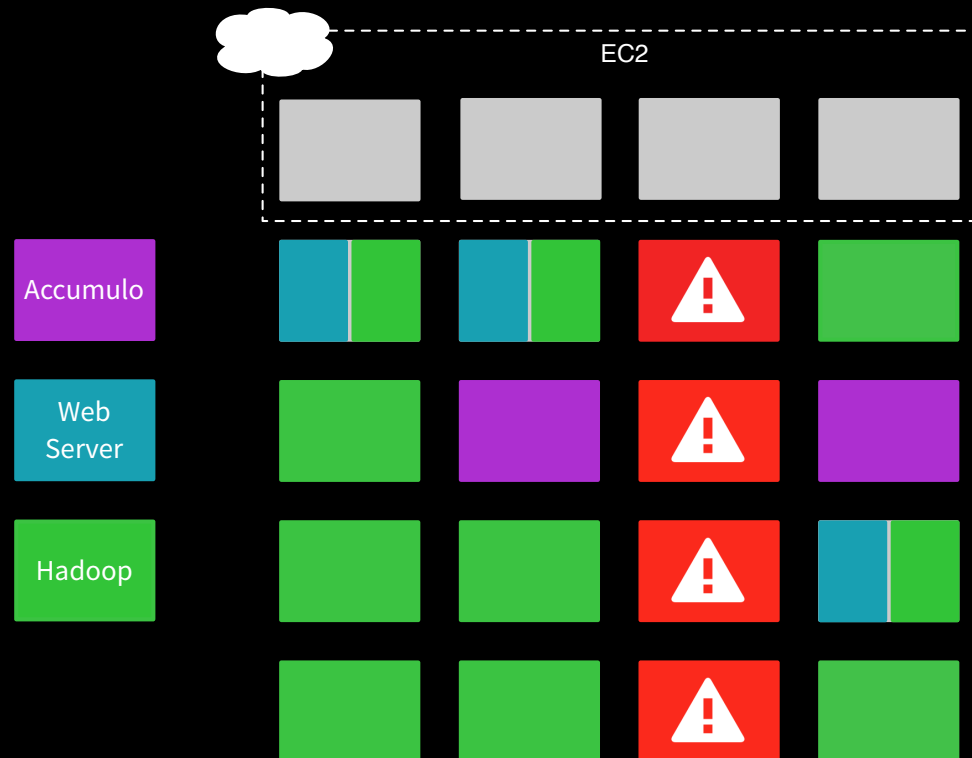
Handling Failure



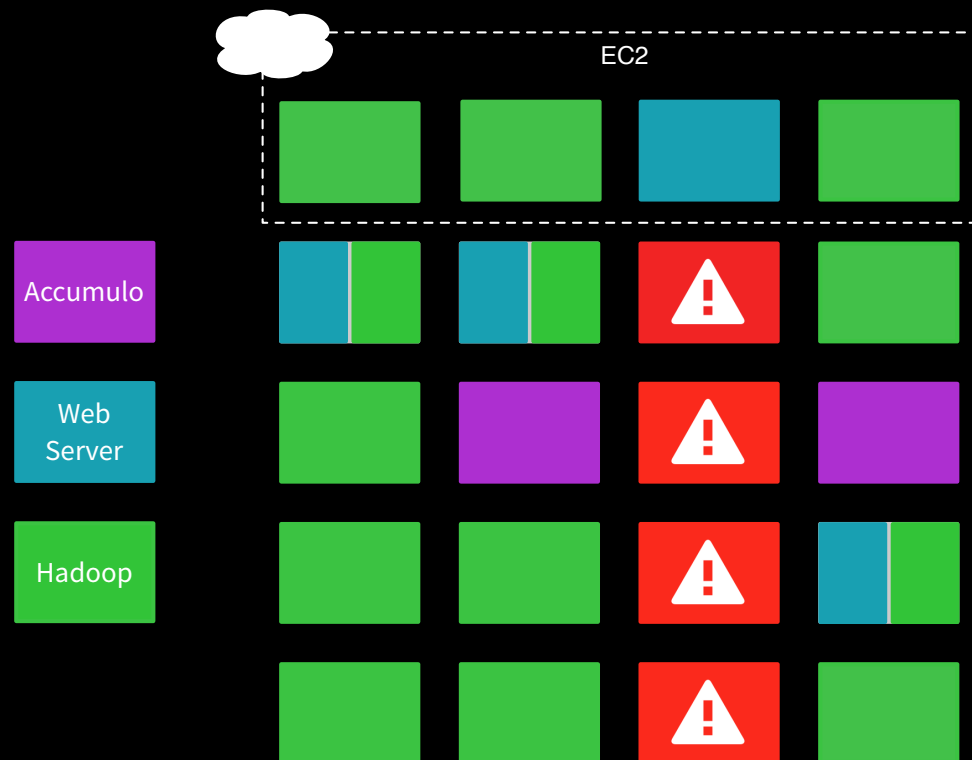
Handling Failure



Adding Capacity (in the Cloud)



Adding Capacity (in the Cloud)



DCOS DEMO



Google Cloud Platform

<http://google.mesosphere.io>



Mesosphere

Thank you.

