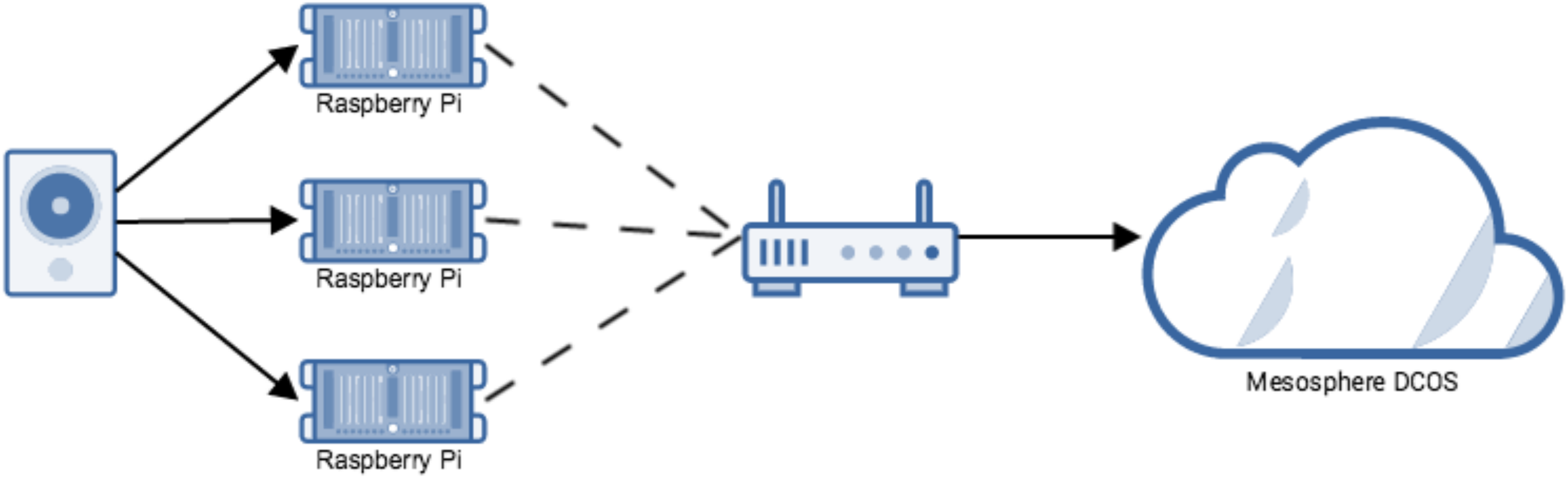
Using Distributed Systems to Ingest Data from the

Internet of Pis

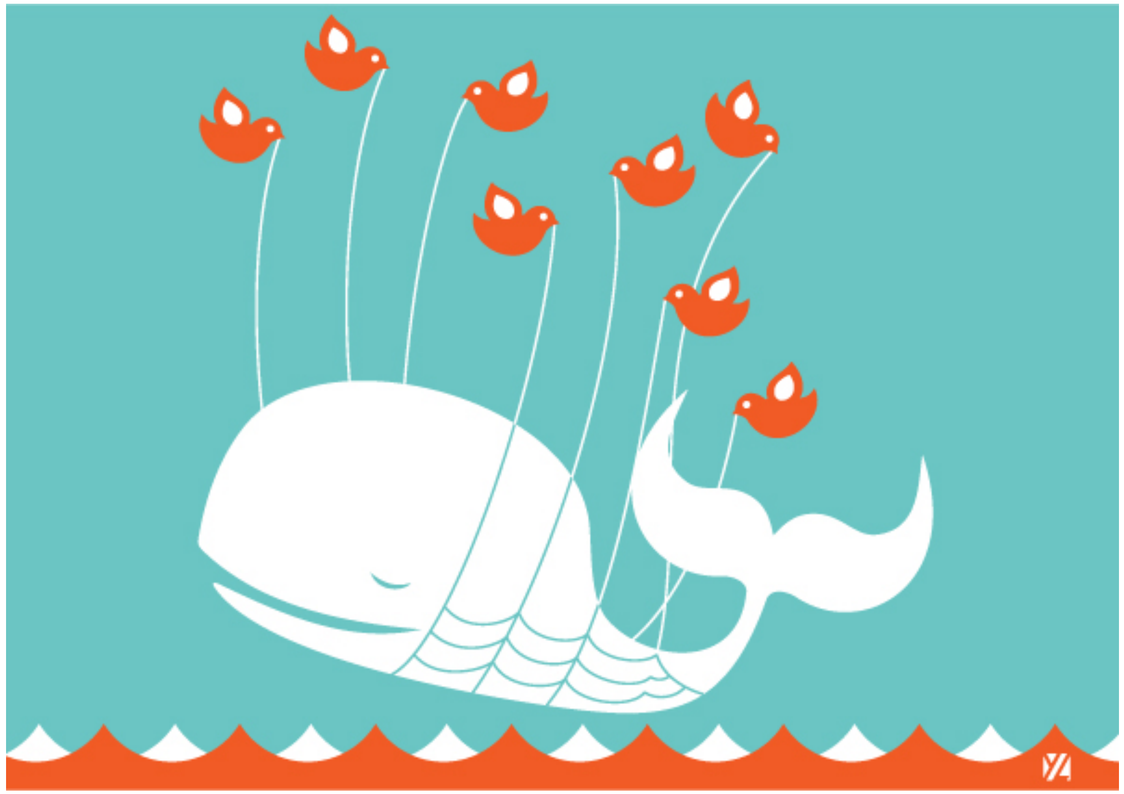
MESOSPHERE

What’s the point?

Internet of Pis

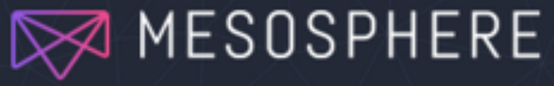


Scaling manually is hard!



- Mesos is credited with “slaying the fail whale”.
- Twitter’s user growth outstripped their infrastructure growth.
- Mesos, a cluster resource manager, let them scale.


Siri: Today
Third Generation
powered by



PUT YOUR DATACENTER ON AUTOPILOT.

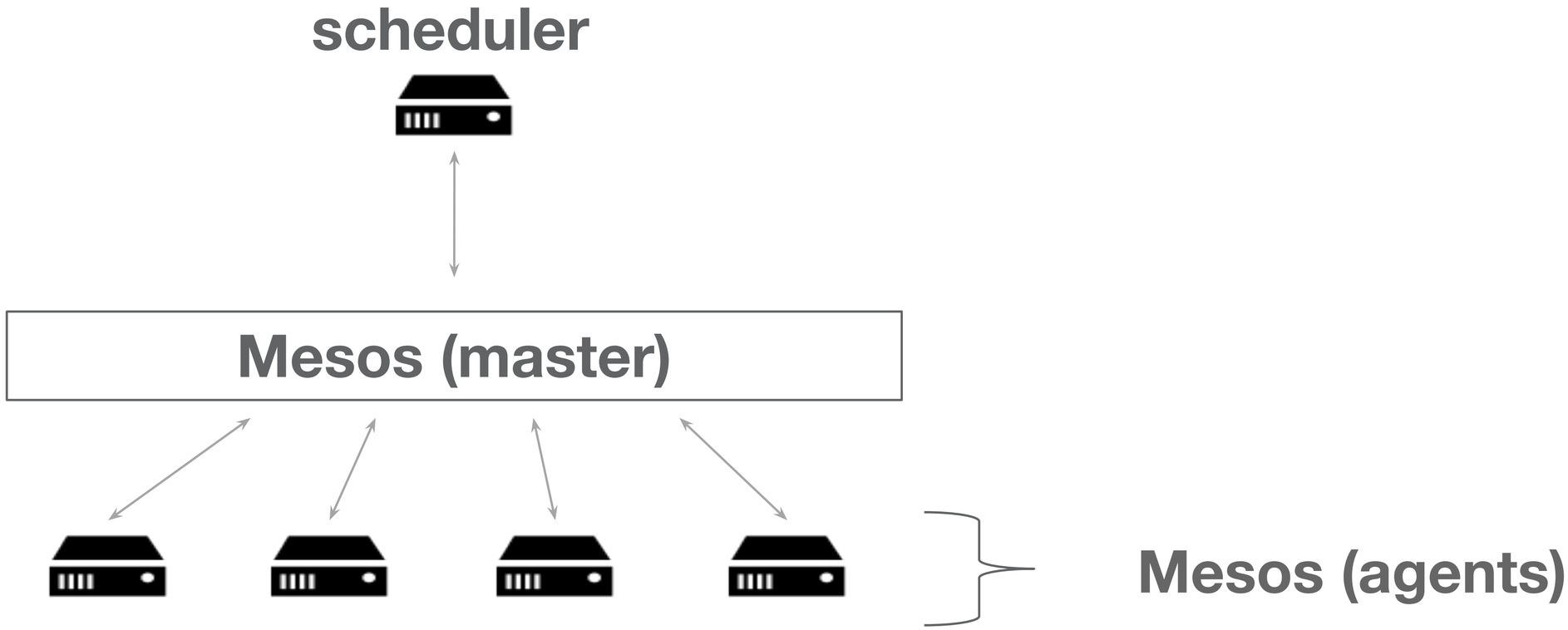
Deploy apps faster. Scale massively. Automate effortlessly.

Get Started Today

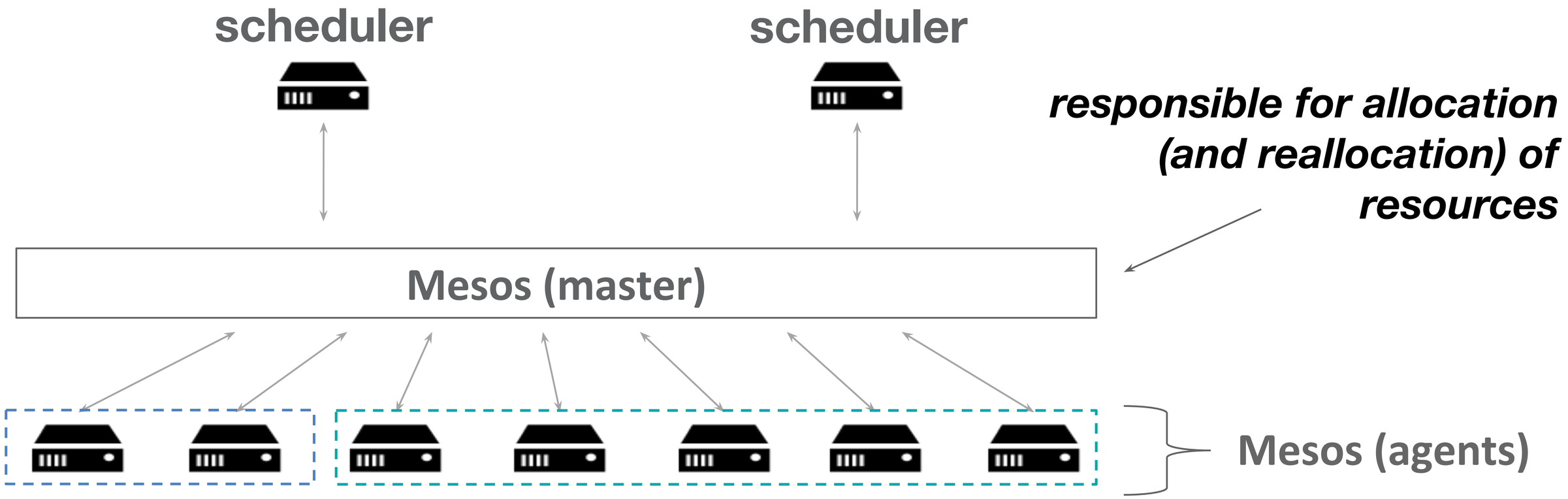
Learn More

Mesoswho

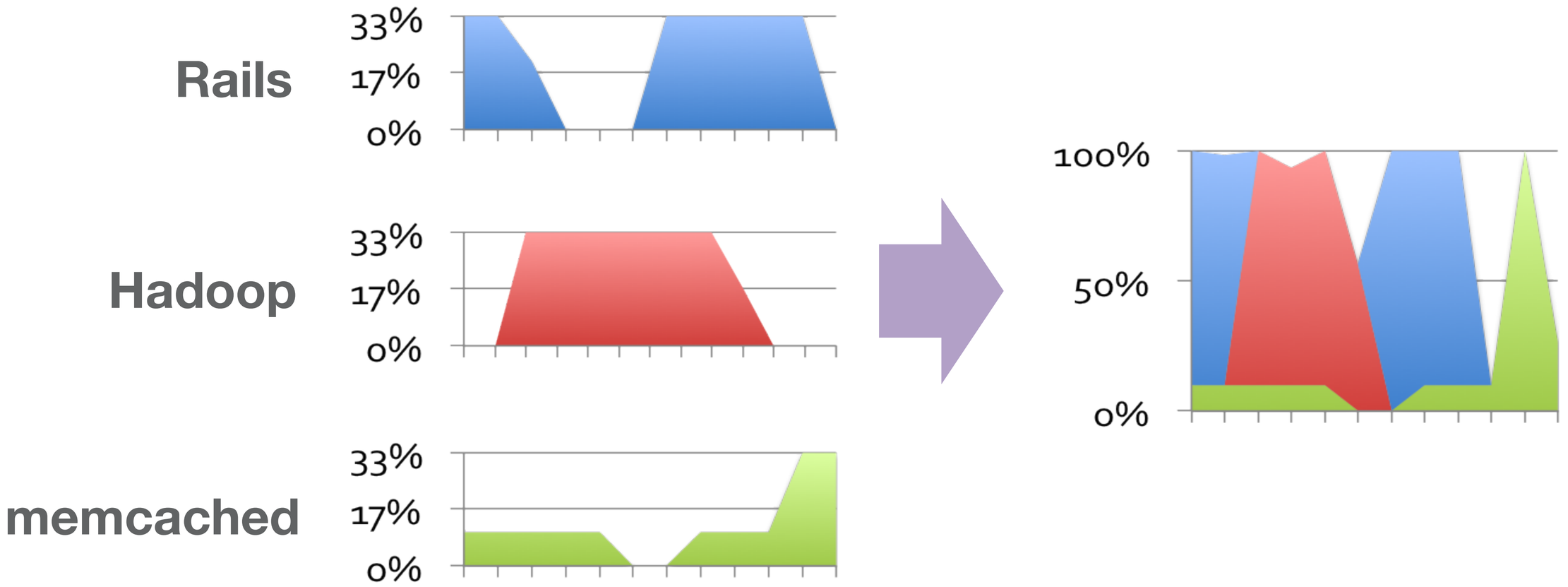
Mesos: Level of Indirection



Mesos: Level of Indirection



Mesos: Improves Utilization



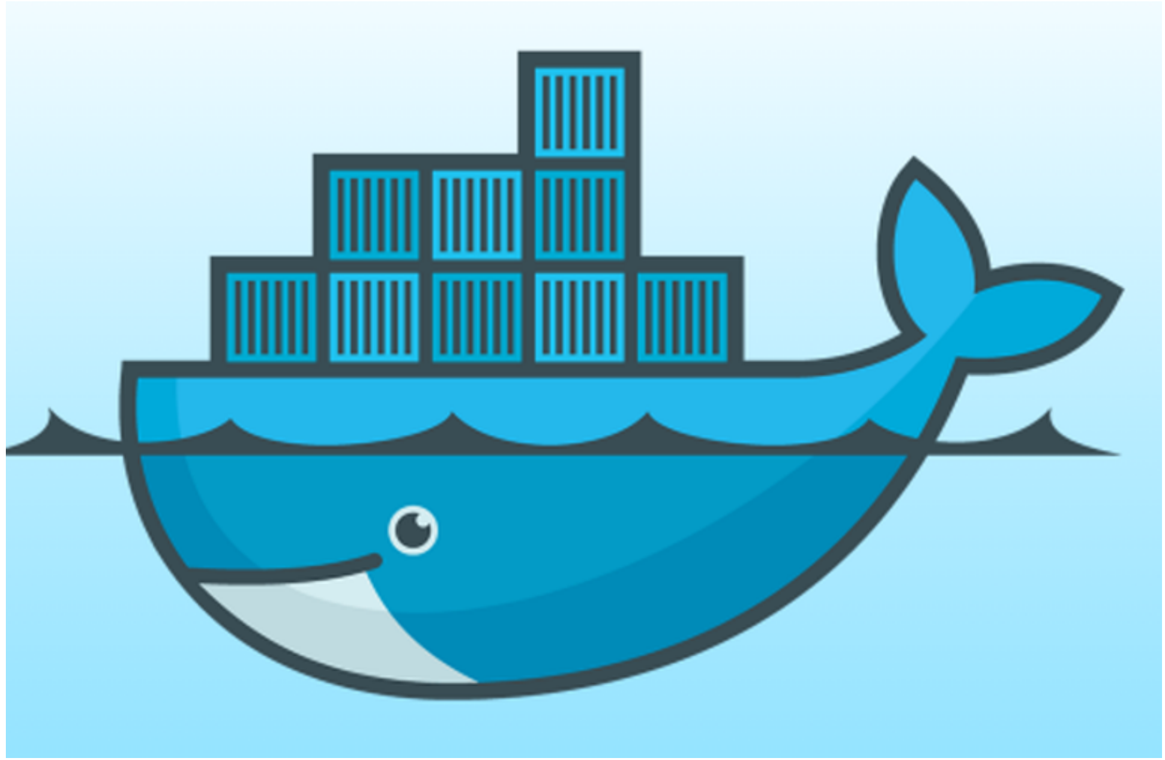
Mesos: Battle Tested

- A top-level Apache project
- A cluster resource negotiator
- Scalable to 10,000s of nodes
- Fault-tolerant, battle-tested
- An SDK for distributed apps

Containerwhat

What are containers?

- Shippable unit of execution.
- Take an operating system, bundle together libraries and application files and ship it around.
- Allows deployment environments (i.e. servers) to be homogenous, “Cattle, not pets”



Workshop!

http://hackers-at-berkeley.mesosphere.io

Tyler's

Philosophy of the Distributed Champion

1. Contain

2. Deploy

3. Discover

First Steps

Set up Docker

Docker is a format and tool that allows you to build and run containers.

We'll use it to package our applications for deployment on the cluster.

1. Install Docker on your local machine: https://docs.docker.com/installation/
2. Sign up for a Docker Hub account: https://hub.docker.com

Set up the DCOS CLI

DCOS is Mesosphere's Mesos distribution.

We will use the DCOS CLI, a command line client, to easily manage and deploy applications on a DCOS cluster.

1. Install the DCOS CLI: https://docs.mesosphere.com/install/cli,

   using the hostname `http://dcos.hackers-at-berkeley.mesosphere.io`

Get the example files

1. Clone the workshop Github repository to get our example files: https://github.com/mesosphere/hackers-at-berkeley

(for git help, try https://help.github.com/articles/cloning-a-repository/)

Exercise 1

Build and run a Python webservice

Exercise 2

Deploy a Python webservice

Exercise 3

Create and deploy a Python webservice that talks to Cassandra

Build & Deploy

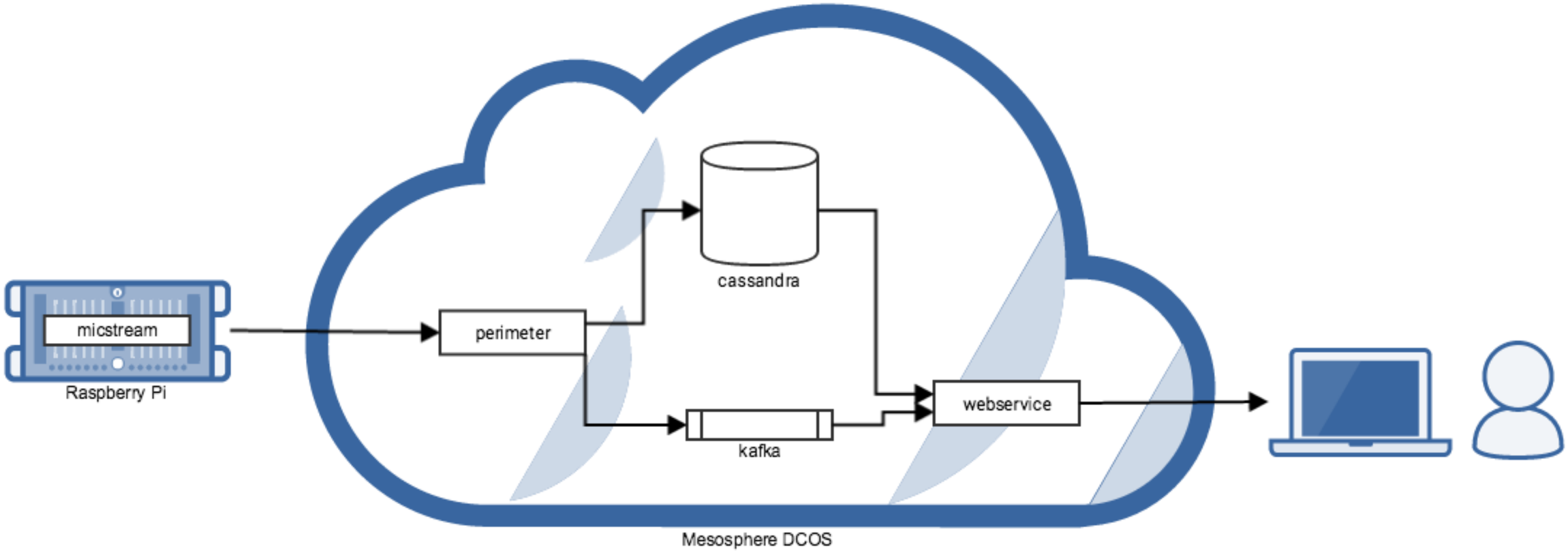
1. Build your Docker web service:

```
# from the Github repository root directory:
cd webservice

# now edit marathon.json, placing your
# username in the “id” field. then, do:
docker build -t awesomeuser/my-webapp .
```

2. Push the image to the Docker Hub:

```
docker build -t awesomeuser/my-webapp-name
```



Thanks!

(come and work for us)

ACT 1 - make something

1. make a WSGI app! (Flask, Django, etc…)
    - examples/snippets/http_app.py
    - examples/frontend_service/front_end.py
2. containerize it!
    - examples/frontend_service/Dockerfile (note the port)
    - `docker build -t <your_dockerhub_name>/<app_name>`
    - `docker push <your_dockerhub_name>/<app_name>`

ACT 2 - Deploy it

- Marathon is a scheduler that runs processes on Mesos based on your specifications (host types, ports, memory/disk usage, etc…)
- We can tell it to run cli commands, arbitrary executables, Docker containers, etc… using an http POST.
  - examples/frontend_service/marathon.json
  - ```
    curl -X POST <host>:8080/v2/apps \
    -d @<your_file>.json \
    -H "Content-type: application/json"
    ```
    ```

How to see it:

1. In the marathon UI, check out your app (find it by the name you set in the JSON you POST'd to marathon)
2. You'll see something like "ip-10-0-4-67.us-west-2.compute.internal"
3. find its public IP in our "magic table" :)
    - or find-hack.mesosphere.io/<app_name>

ACT 3 - Service Discovery

- Our apps will be scheduled to ports that are available on the machine, which means they cannot be predicted ahead of time.
- We can use SRV DNS records to find both an (internal) IP and a port for a name
  - examples/snippets/marathon_SRV_discovery.py
  - `srvlookup.lookup('myapp', 'tcp', 'marathon.mesos')`