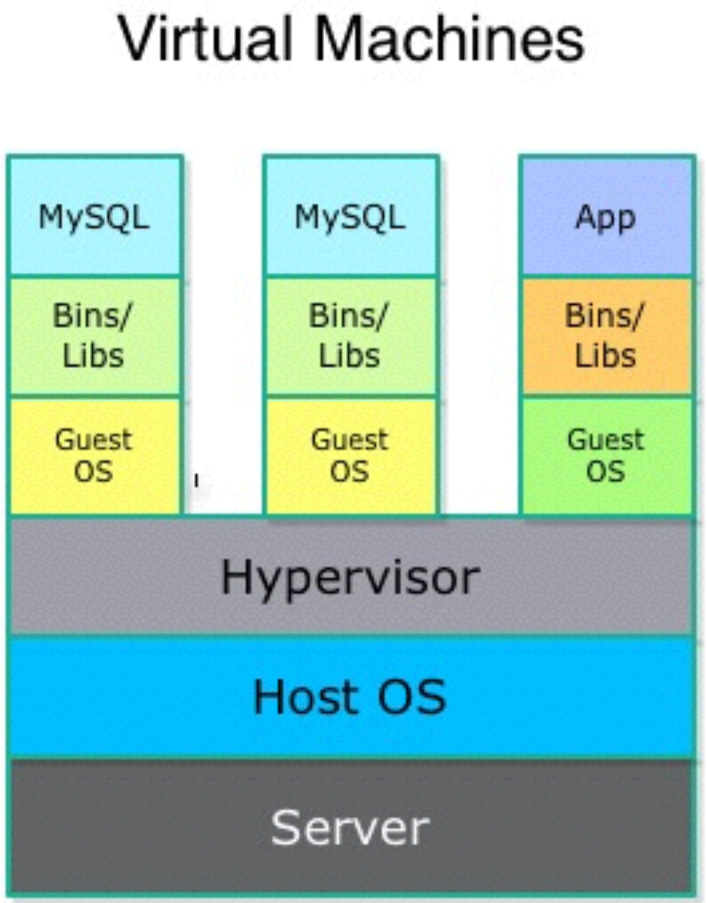
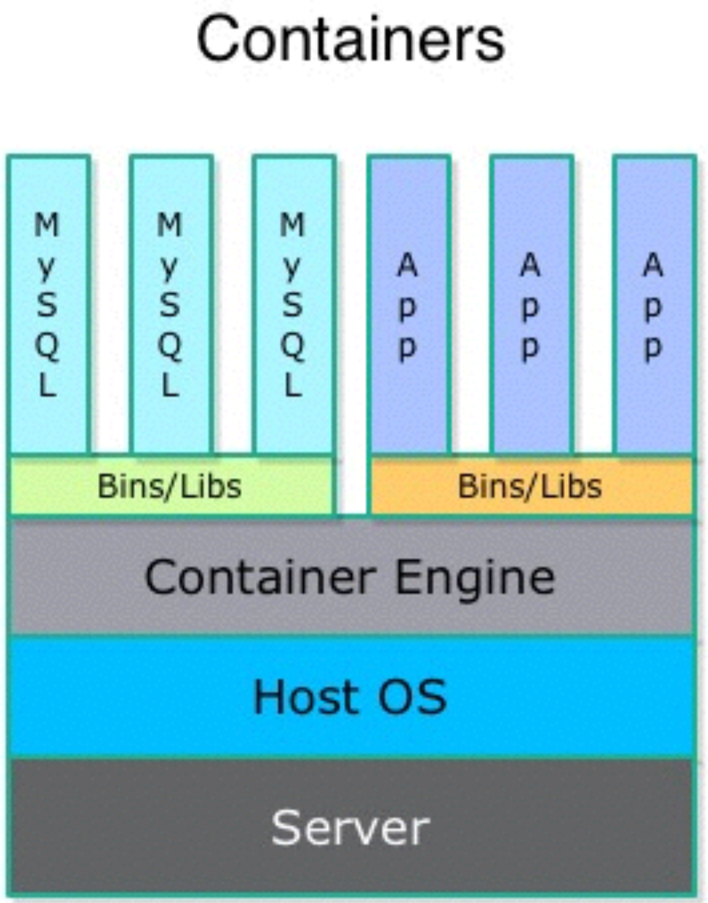
Docker, Mesos & other adventures in Wonderland

Docker

Middle ground between VMs and processes





Mesos

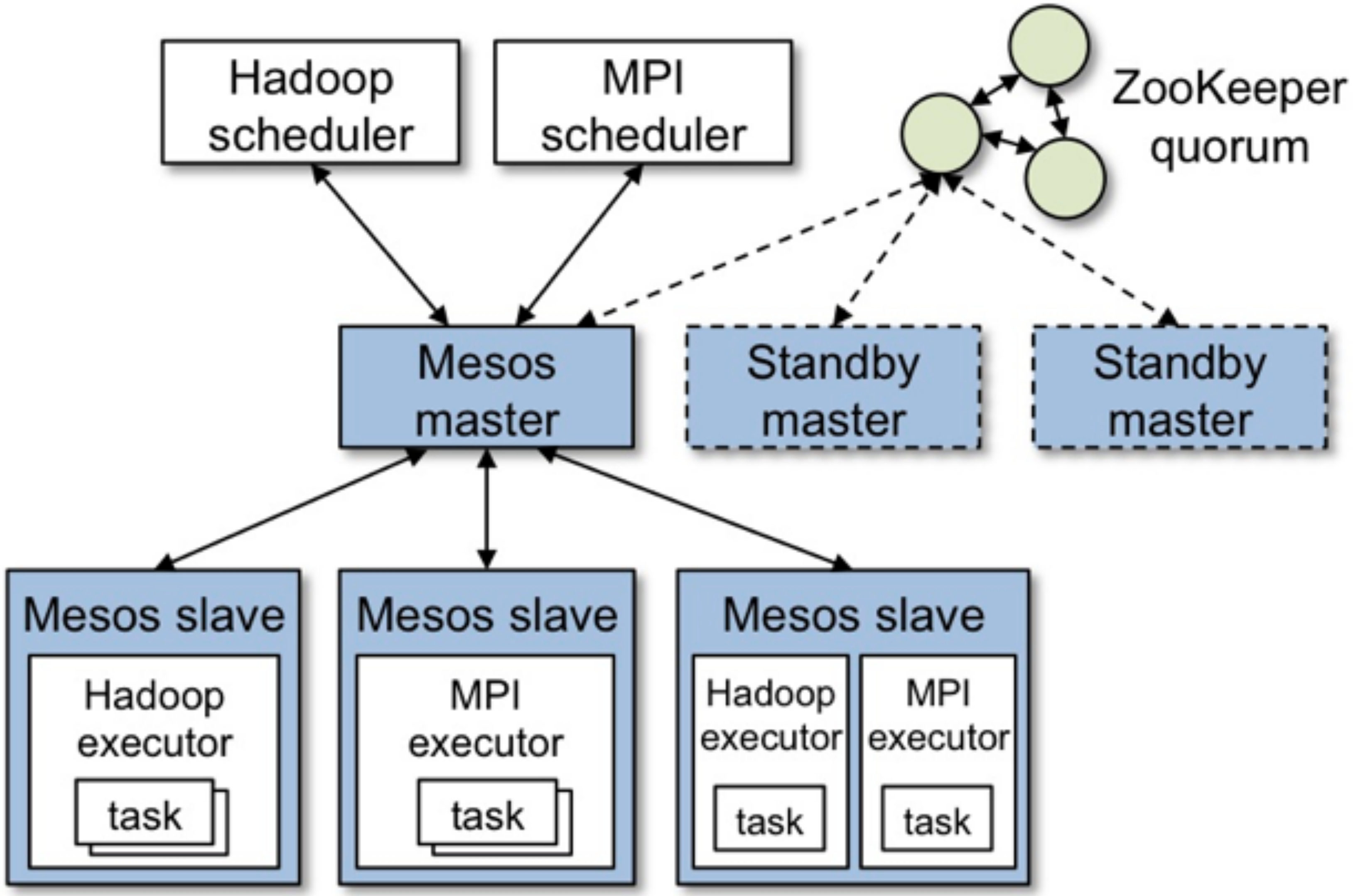
An “Operating System” for the cluster (http://mesos.apache.org)

A pluggable two level scheduler for long running services and batch jobs

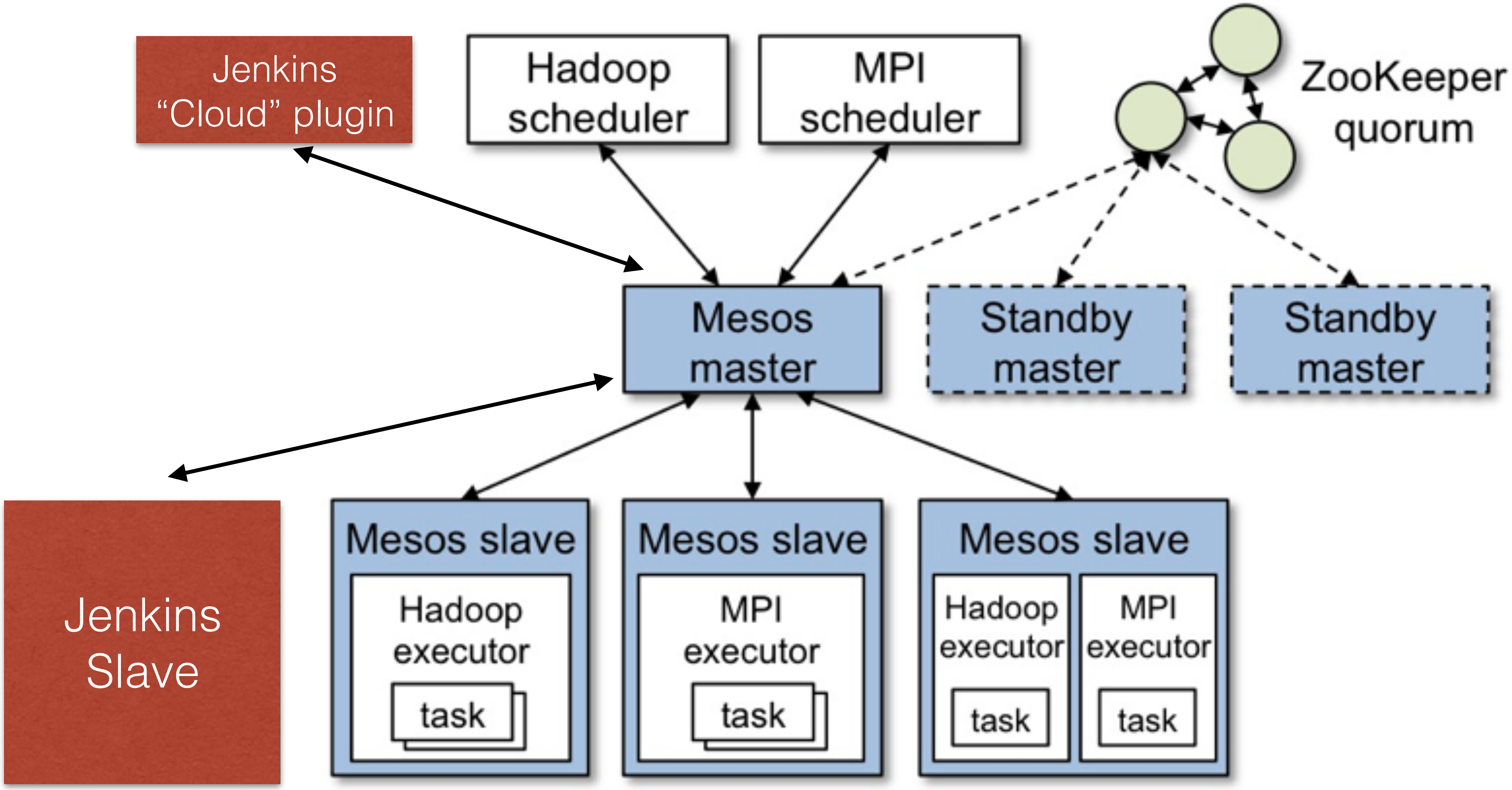
First developed at UC Berkeley, now widely adopted in the industry (Twitter, AirBnB, … Apple)

Natively supports docker containers (among other possible isolation mechanism)

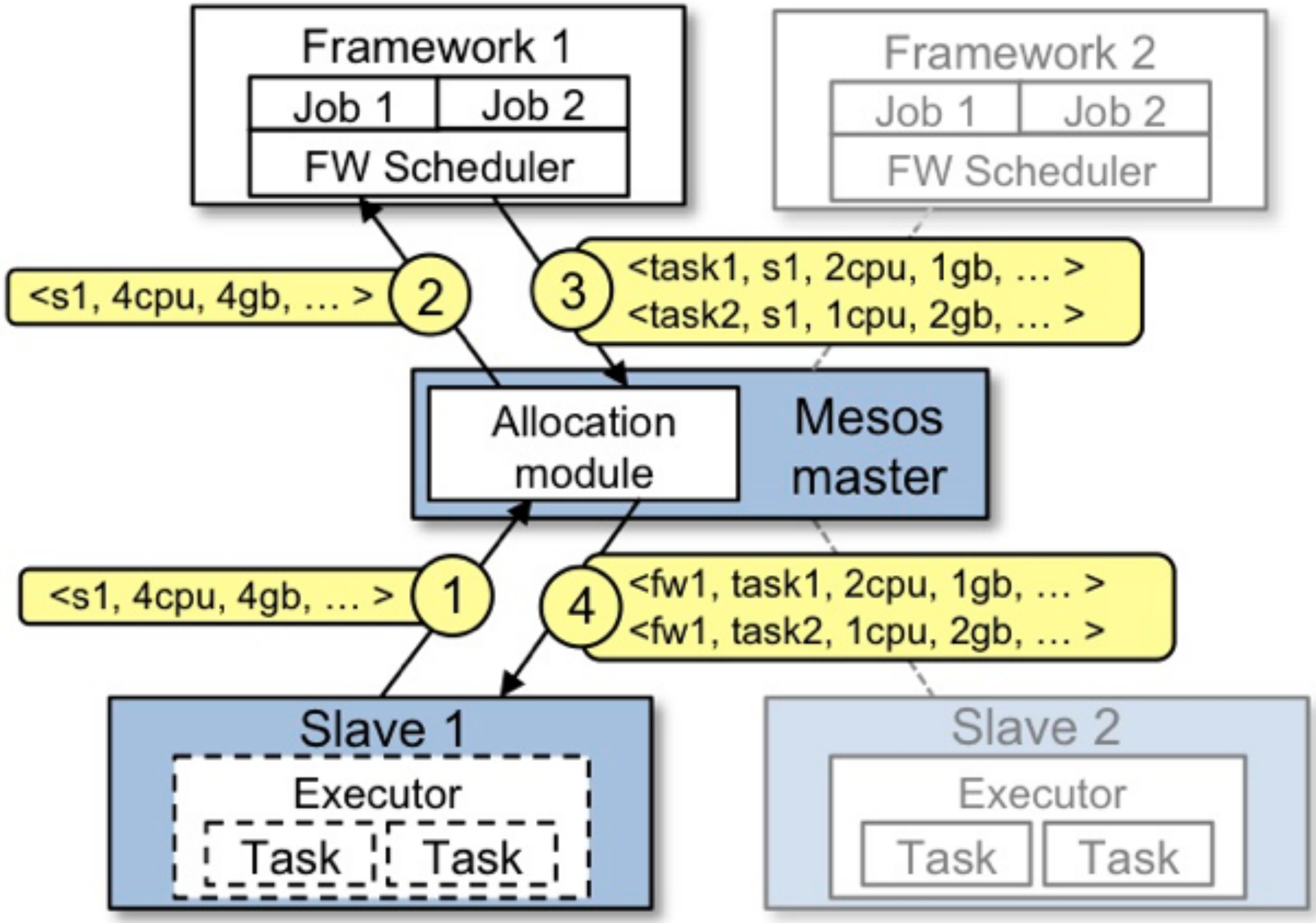
Mesos architecture



Mesos architecture



Mesos resource offers



All this available as a C++ / python / Java API

Marathon

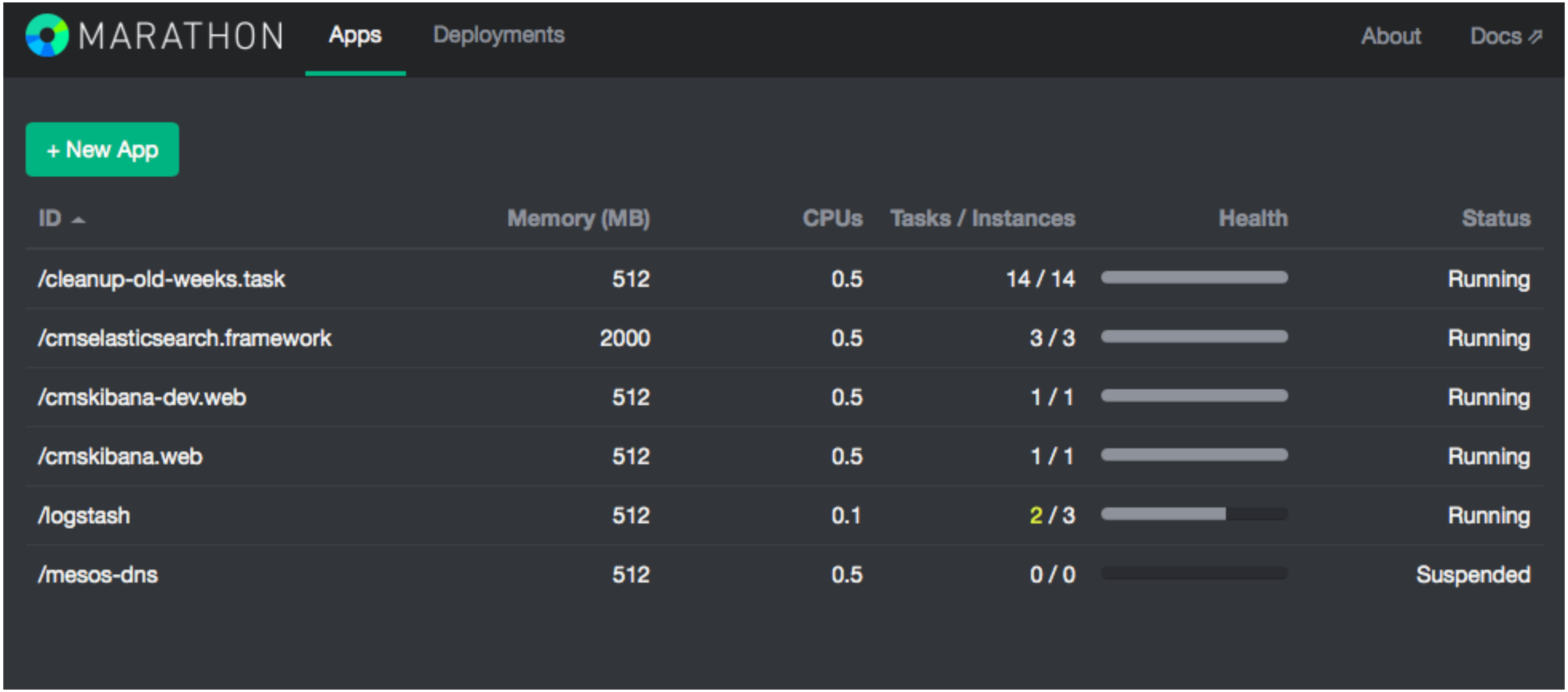
While writing your own “Framework” and “Scheduler” makes sense for specific applications (e.g. the Jenkins plugin, or what Apple did for for Siri), for more coarse grained usage(like deploying services) you want to use something pre-cooked (i.e. you want a PaaS).

A simple PaaS to deploy processes and containers on a Mesos cluster is Marathon, from Mesosphere (https://mesosphere.com)

GUI for monitoring and spawning processes

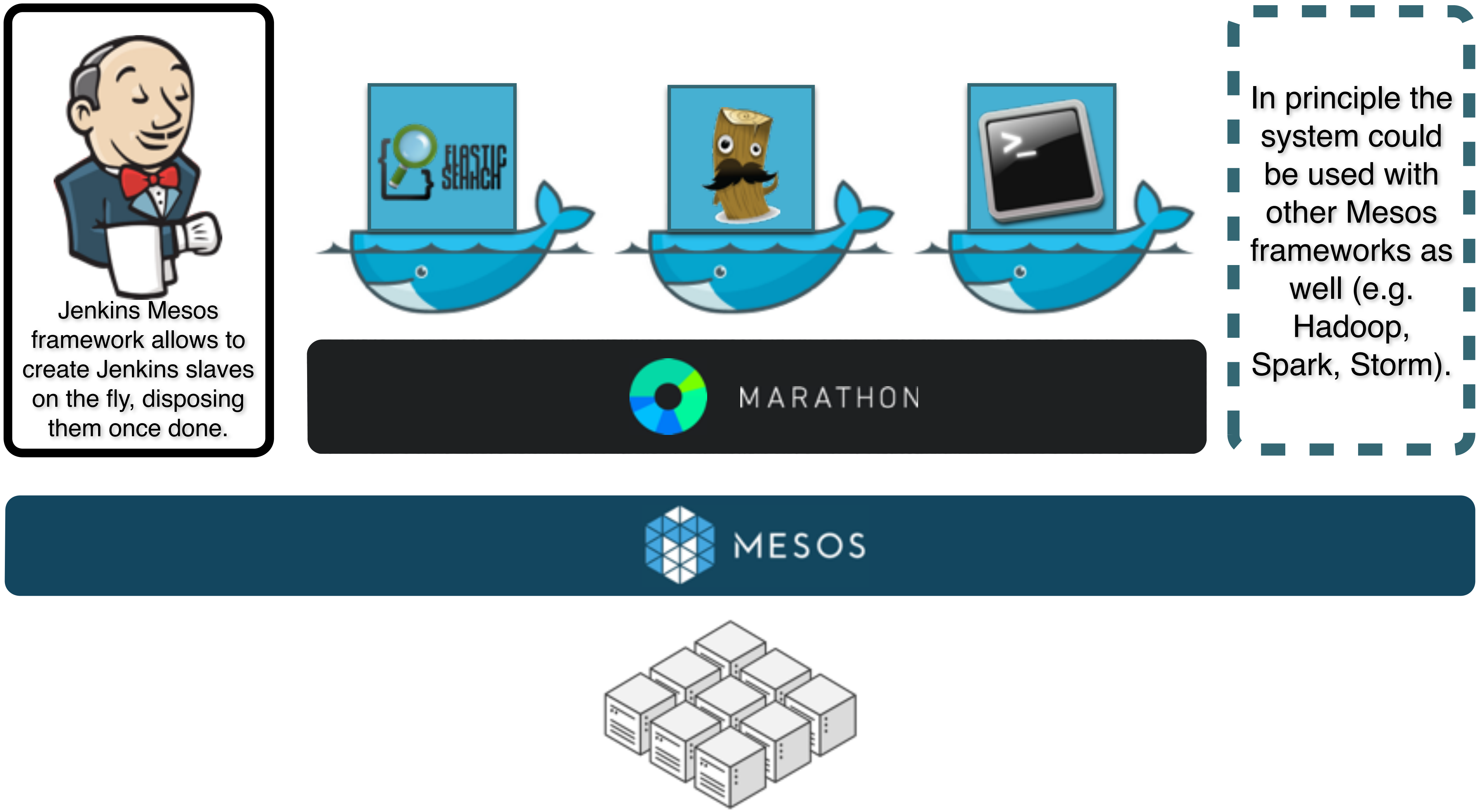
Advanced configuration via a JSON file pushed through the REST API

Marathon



ID	Memory (MB)	CPUs	Tasks / Instances	Health	Status
/cleanup-old-weeks.task	512	0.5	14 / 14		Running
/cmselasticsearch.framework	2000	0.5	3 / 3		Running
/cmskibana-dev.web	512	0.5	1 / 1		Running
/cmskibana.web	512	0.5	1 / 1		Running
/logstash	512	0.1	2 / 3		Running
/mesos-dns	512	0.5	0 / 0		Suspended

Marathon



Jenkins Mesos framework allows to create Jenkins slaves on the fly, disposing them once done.

In principle the system could be used with other Mesos frameworks as well (e.g. Hadoop, Spark, Storm).

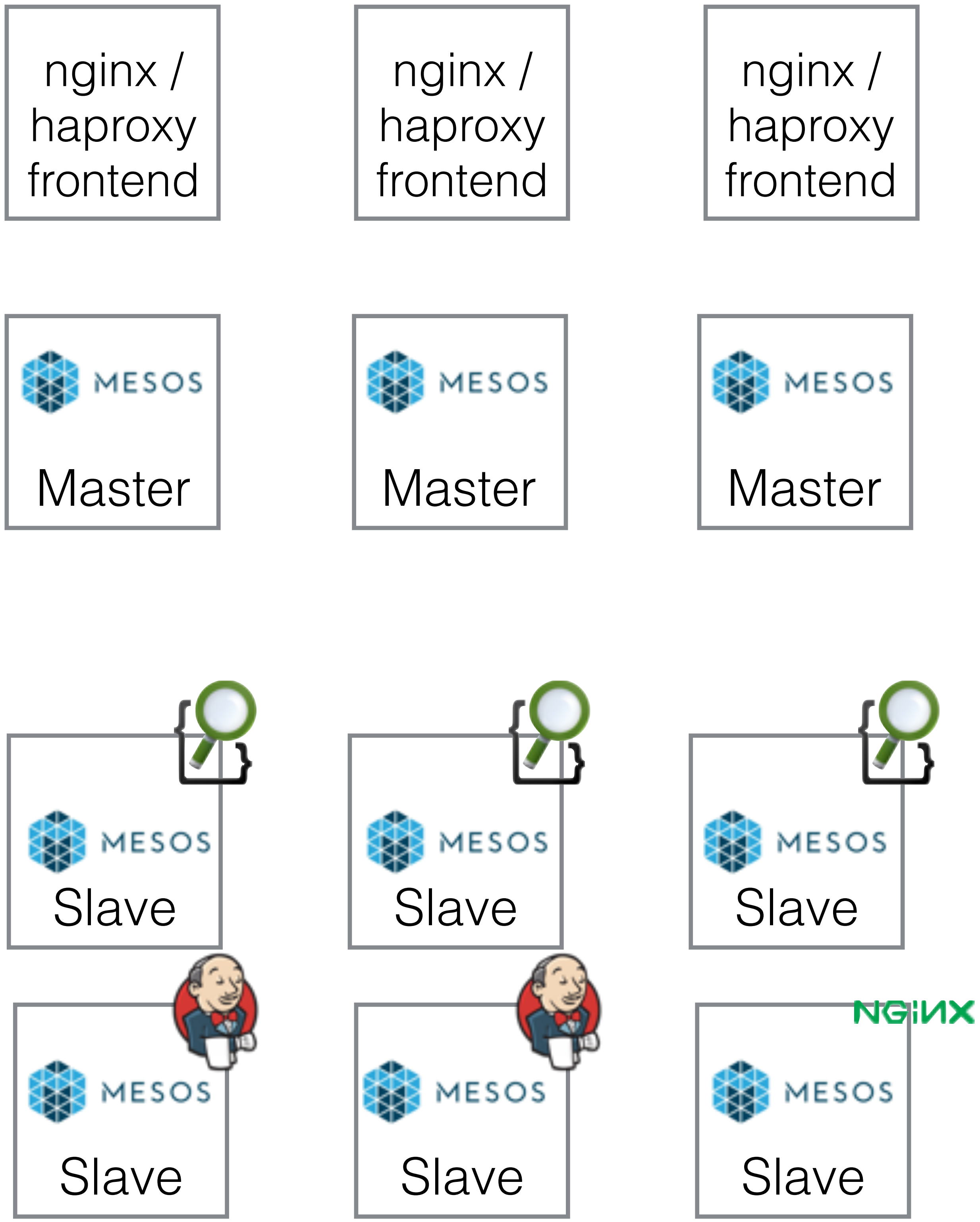
CMS Build Infrastructure

Frontend with HA setup using CERN LB DNS, nginx for SSL termination and authorisation, haproxy for traffic routing and SSO backend

Resource arbitration via 3-way redundant **Apache Mesos** setup, using different OpenStack zones, leader election via Zookeeper (1 dead master resilience)

Services run on undifferentiated CPU boxes, either running on the bare OS or running inside **Docker**

Services which we run varies from **Jenkins** build slave, to **web server backends** or **Elasticsearch**



nginx / haproxy frontend

nginx / haproxy frontend

nginx / haproxy frontend

MESOS Master

MESOS Master

MESOS Master

MESOS Slave

MESOS Slave

MESOS Slave

MESOS Slave

MESOS Slave

MESOS Slave

Service Discovery via DNS, populated with A and SRV records discovery by mesos registry information (ala Consul) or using **Marathon Framework** REST API.

All the services are being restarted automatically by Marathon whenever they die on machines that offer a compatible set of resources. Looking forward dynamic resource allocation (i.e. persistent disk storage on slaves) to simplify setup even further.

Try it at home

Run it on your laptop (using boot2docker):

```
docker run --net=host -it cmssw/zookeeper &
docker run --net=host -it cmssw/mesos-master &
docker run --net=host -it cmssw/marathon &
docker run --net=host \
           -v /var/run/docker.sock:/var/run/docker.sock \
           -v /usr/local/bin/docker:/usr/bin/docker \
           -v /sys:/sys \
           -it \
           cmssw/mesos-slave
```

and connect to localhost:8080 (or `boot2docker ip`:8080)

Pipe dreaming…

- A shared pool of resources all running Mesos, with a central Marathon instance.
- Use Marathon to run “Mesos on top of Mesos” and use it to run a “per user” Marathon instance.
- Use the per user Marathon entry point to launch your own Dockerized services.
- Add some economic model so that people can “buy and sell” (or “loan and rent”) their assigned resources.
- Profit.