
2015-12-08

MESOS CODE FEST



MESOSPHERE

WHO?

Matthias Veit: Tech Lead Marathon
Jan Repnak: Product Support Engineer

AGENDA

1. Theory (90 Min)
 1. Goals & teams
 2. Container 101
 3. Apache Mesos
 4. Marathon
 5. Datacenter Operating System (DCOS)
 6. Learning Resources
2. Hands-on Session

GOALS & TEAMS

GOALS

- Understand container basics incl. Docker
- Understand Mesos and Marathon
- Understand orchestration and scheduling of containers
- Able to implement an application using Docker and Apache Mesos+Marathon

YOUR TEAM ...

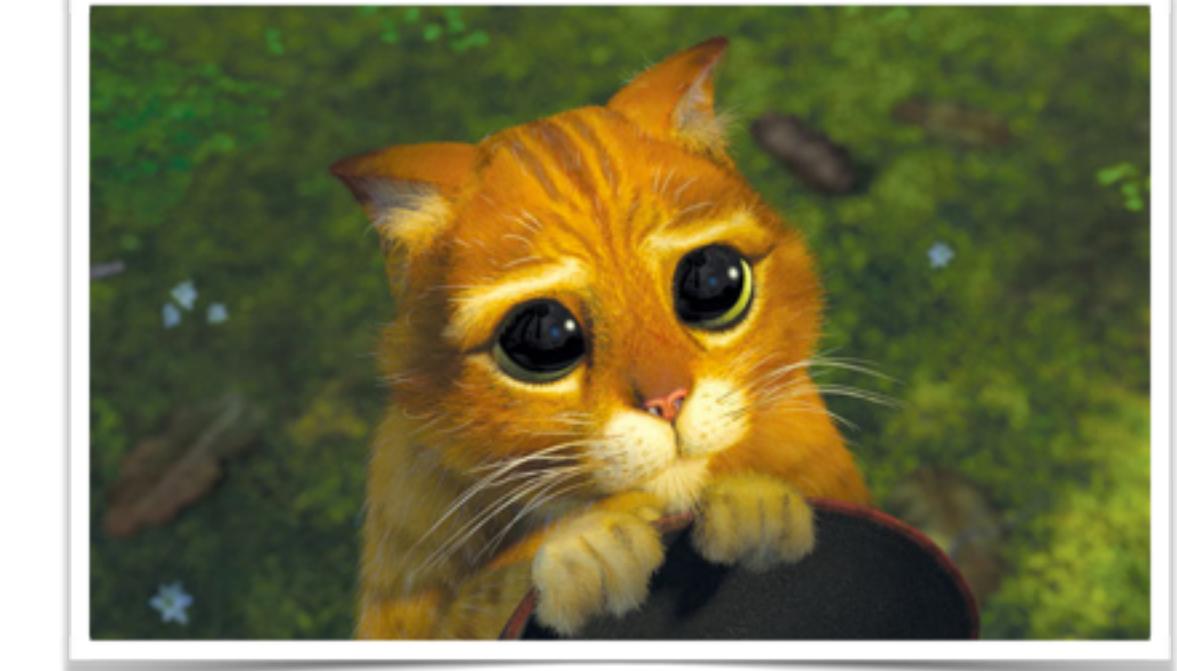
- We have DCOS clusters prepared → team up in groups of three
- Find a place where you can work together
- One person should drive, others help/comment/plan
- Reach out to one of the Mesosphere team members to announce team
- Check out: <http://bit.ly/mesoscodefest>

CONTAINER 101

Pets vs Cattle

WHAT IS THIS ALL ABOUT?

Pets: treat machines as individuals that you give names and when they get ill you nurse them back to health.



Cattle: anonymous, identical to each other; you assign numbers and when they get ill you get rid of it.



http://www.theregister.co.uk/2013/03/18/servers_pets_or_cattle_cern/

CONSEQUENCES OF GOING ALL-IN WITH CATTLE APPROACH

- scale out on commodity hardware
- elasticity
- 'cheap' & 'simple'
- R U on pager duty? Just sleep through!



- social >> technology challenge
- new technical challenges such as service discovery



http://www.theregister.co.uk/2013/03/18/servers_pets_or_cattle_cern/

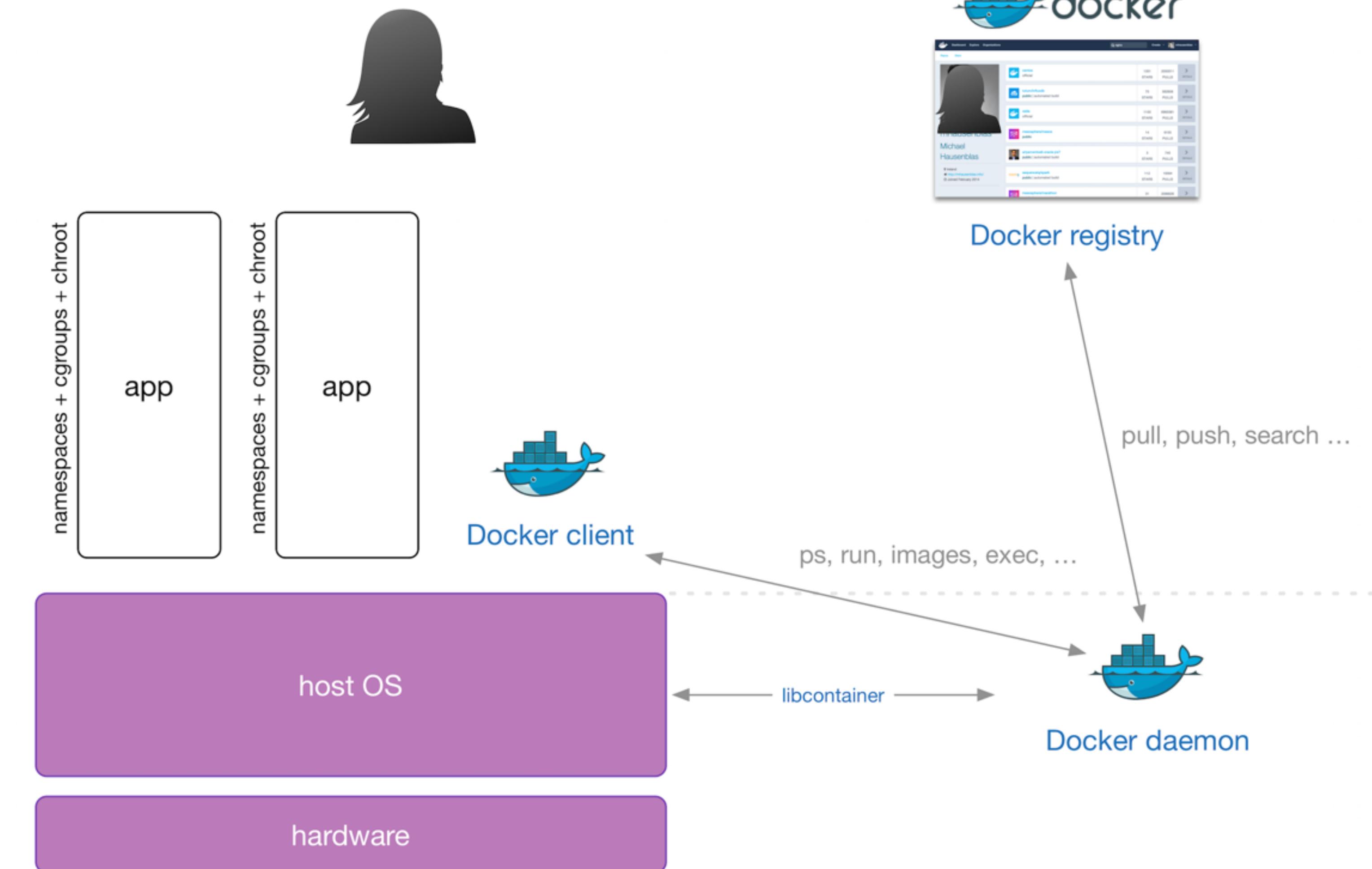
LINUX CONTAINERS

HOW TO ISOLATE PROCESSES

- namespaces
 - Isolate PIDs between processes
 - Isolate network resources (stacks, devices, etc.)
 - Isolate hostname/NIS (UTS)
 - Isolate filesystem mount (chroot)
 - Isolate inter process communication (IPC)
- cgroups

<https://sysadmincasts.com/episodes/14-introduction-to-linux-control-groups-cgroups>

DOCKER



DOCKER

- Package an application with all dependencies (e.g. python 2.7, python 3)
- Create new docker containers by extending existing ones (e.g. ubuntu:latest, java:7,)
- Always run the same way (always have the exact same environment)
- Run docker containers is easy
 - Test locally
 - Run in production

DOCKER

Registries

- Docker Hub

<https://hub.docker.com/>

- Google Cloud

<https://cloud.google.com/tools/container-registry/>

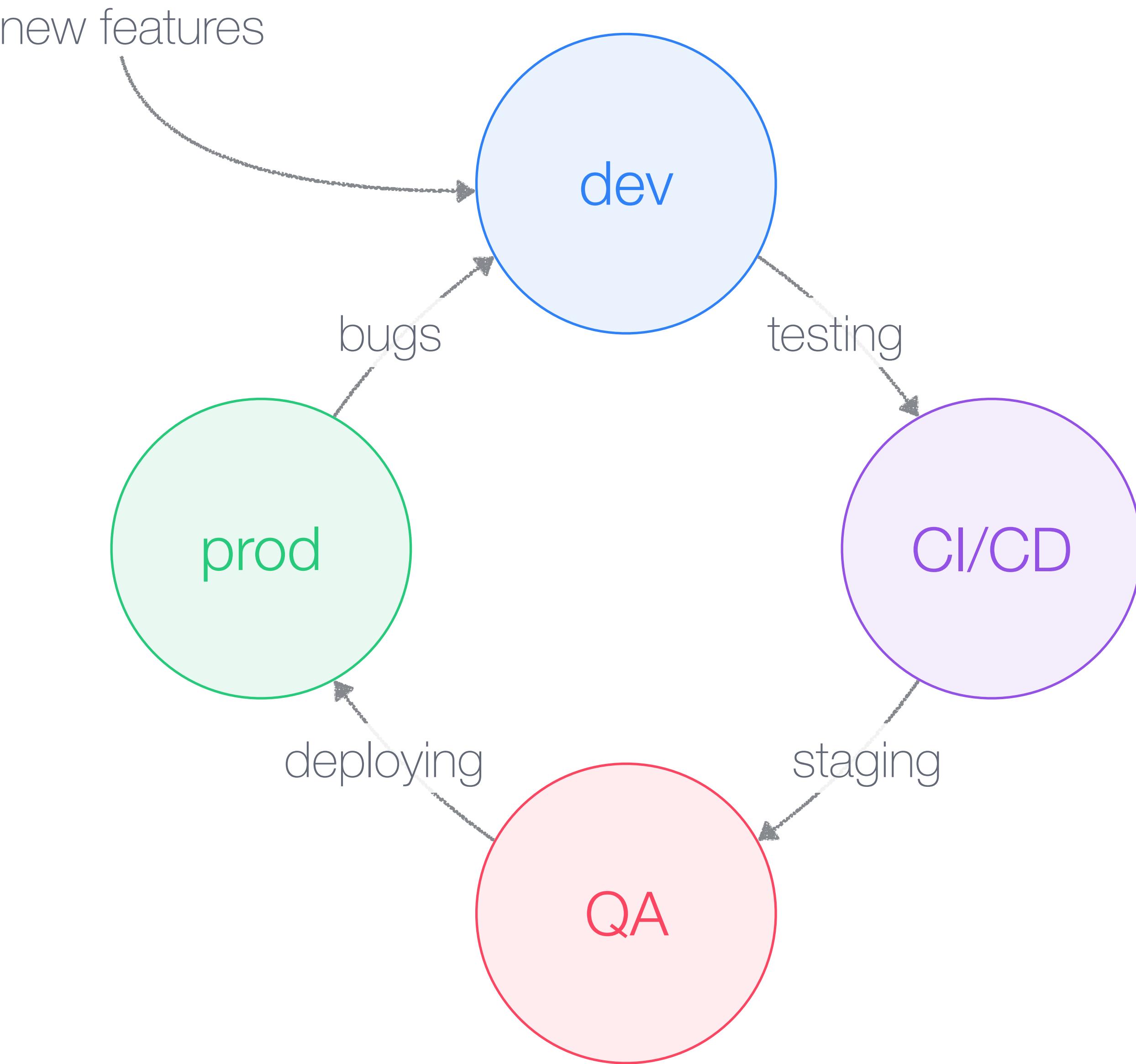
- AWS

<https://aws.amazon.com/ecr/>

- Run your own

<https://docs.docker.com/registry/deploying/>

CONTAINER LIFE CYCLE



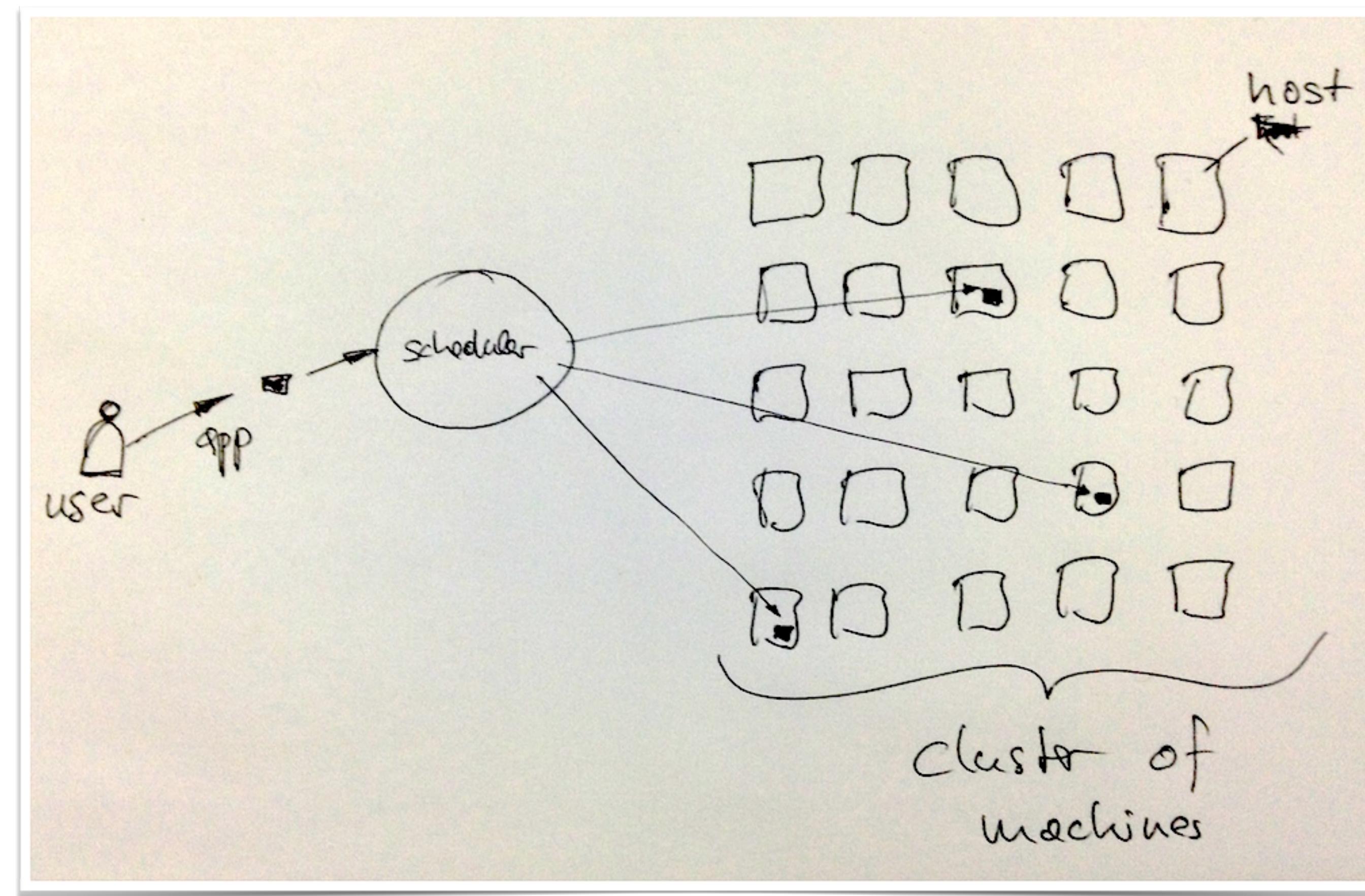
APACHE MESOS

Apache Mesos

- A top-level ASF project
- A cluster resource negotiator
- Scalable to 10,000s of nodes but also useful for a handful of nodes
- Fault-tolerant, battle-tested
- An SDK for distributed apps
- Native Docker support

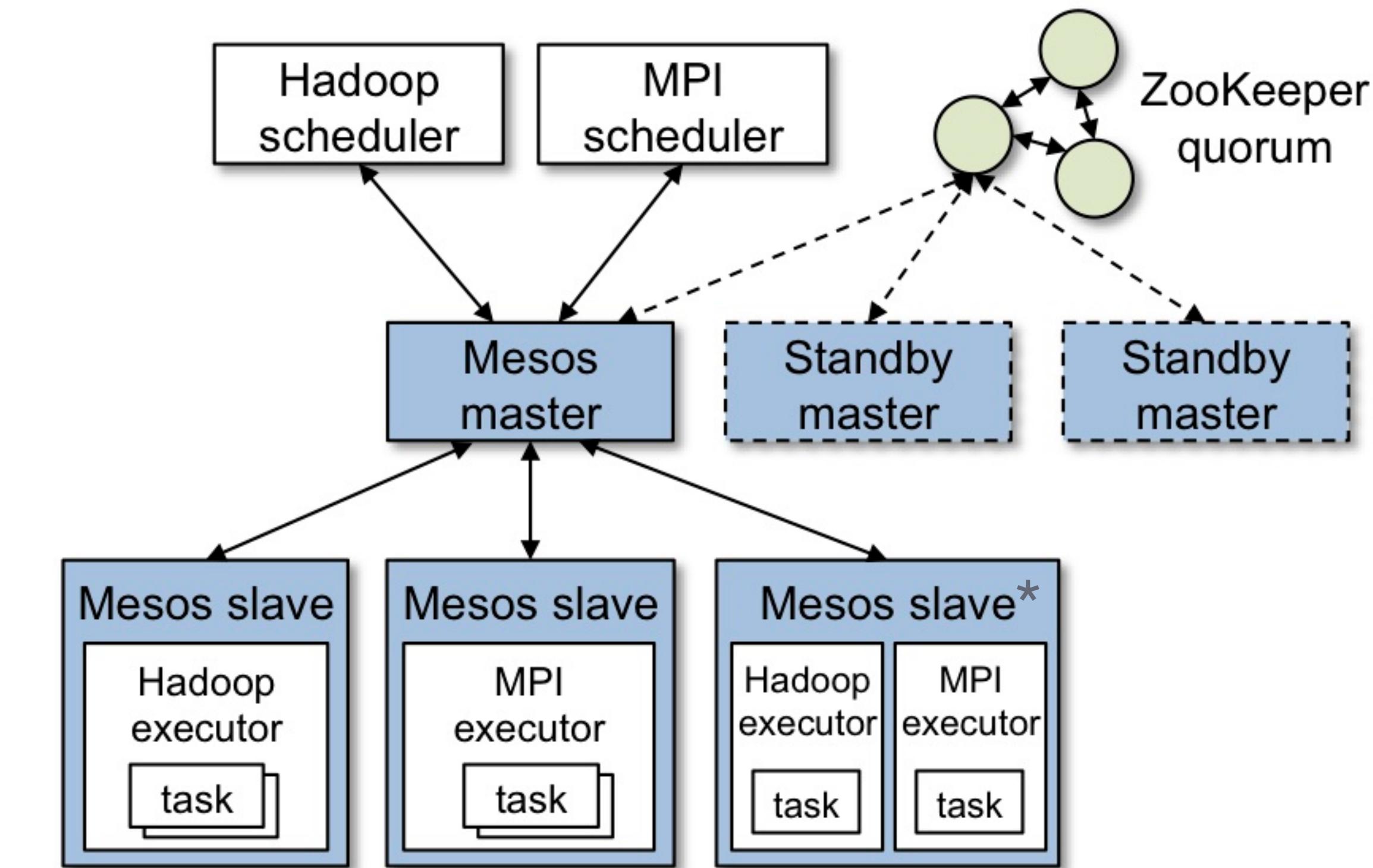


WHAT'S THE GOAL?



Apache Mesos ARCHITECTURE

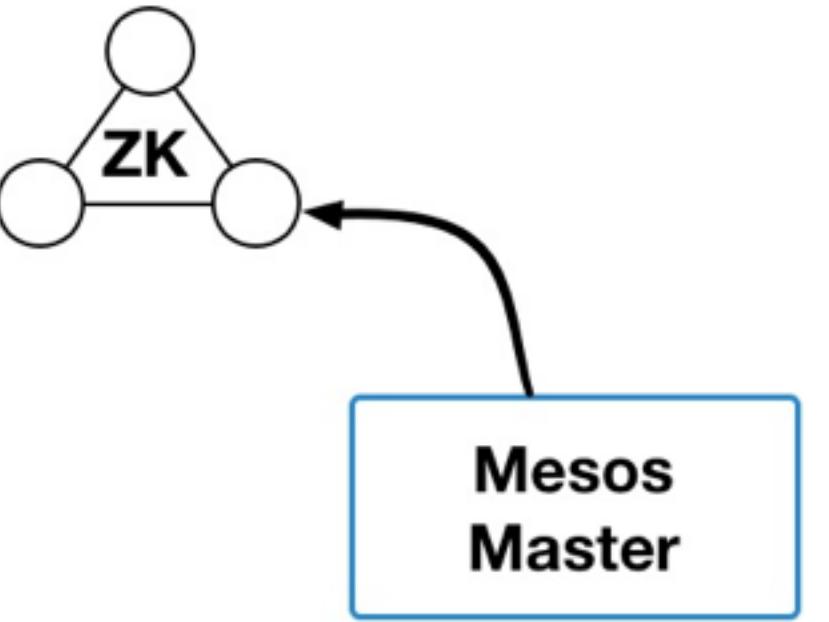
*) now: agent

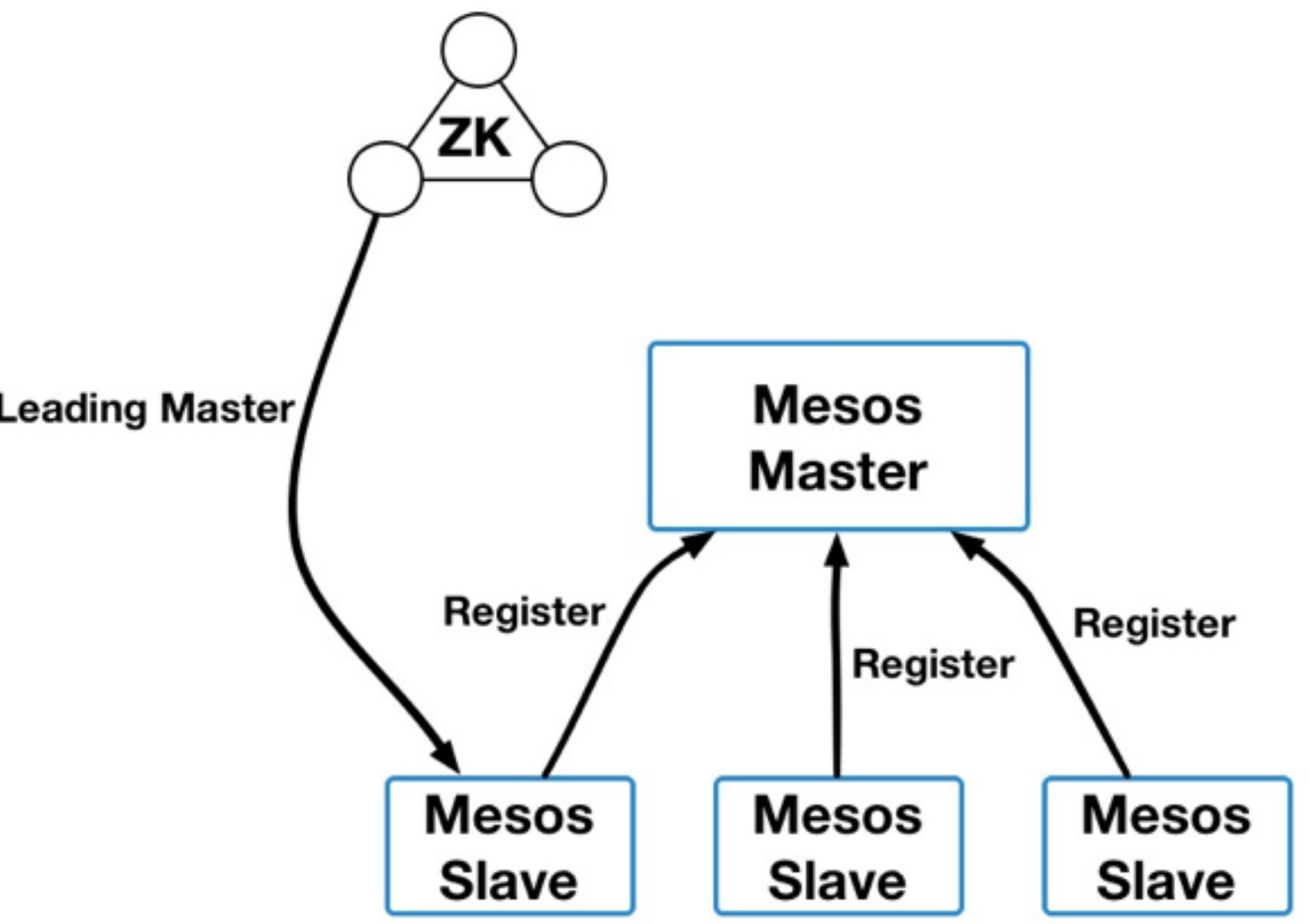


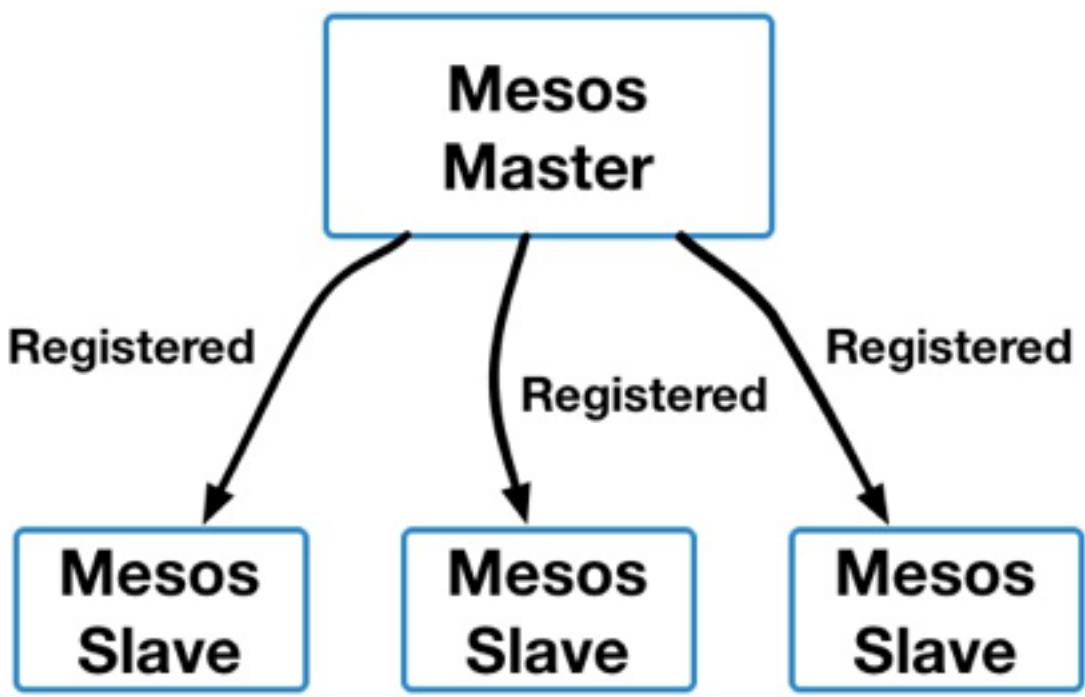
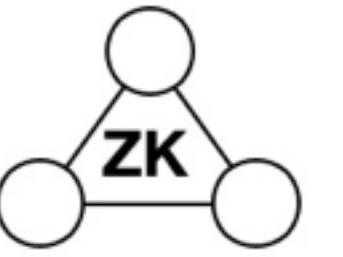
http://mesos.berkeley.edu/mesos_tech_report.pdf

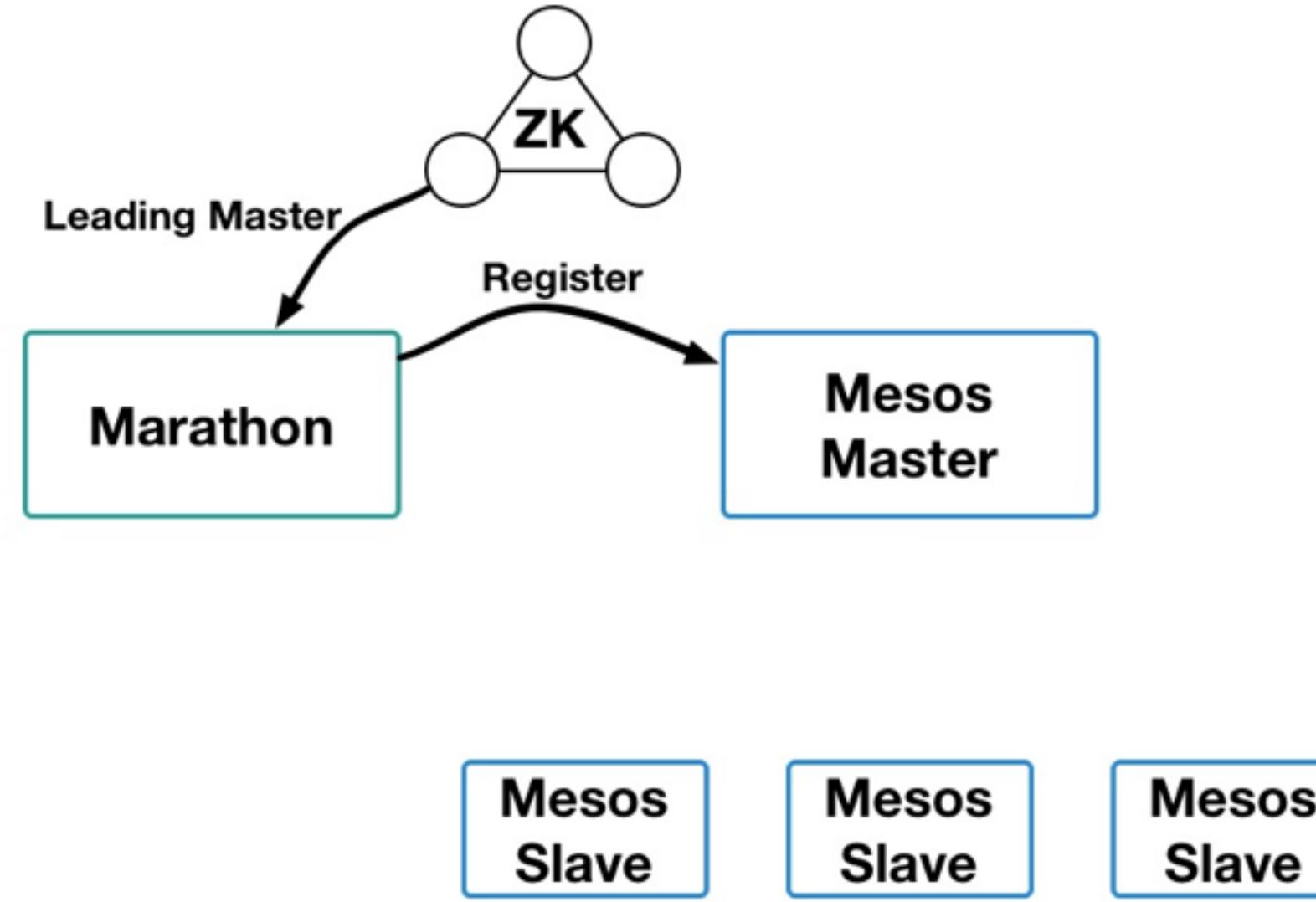
RESOURCES

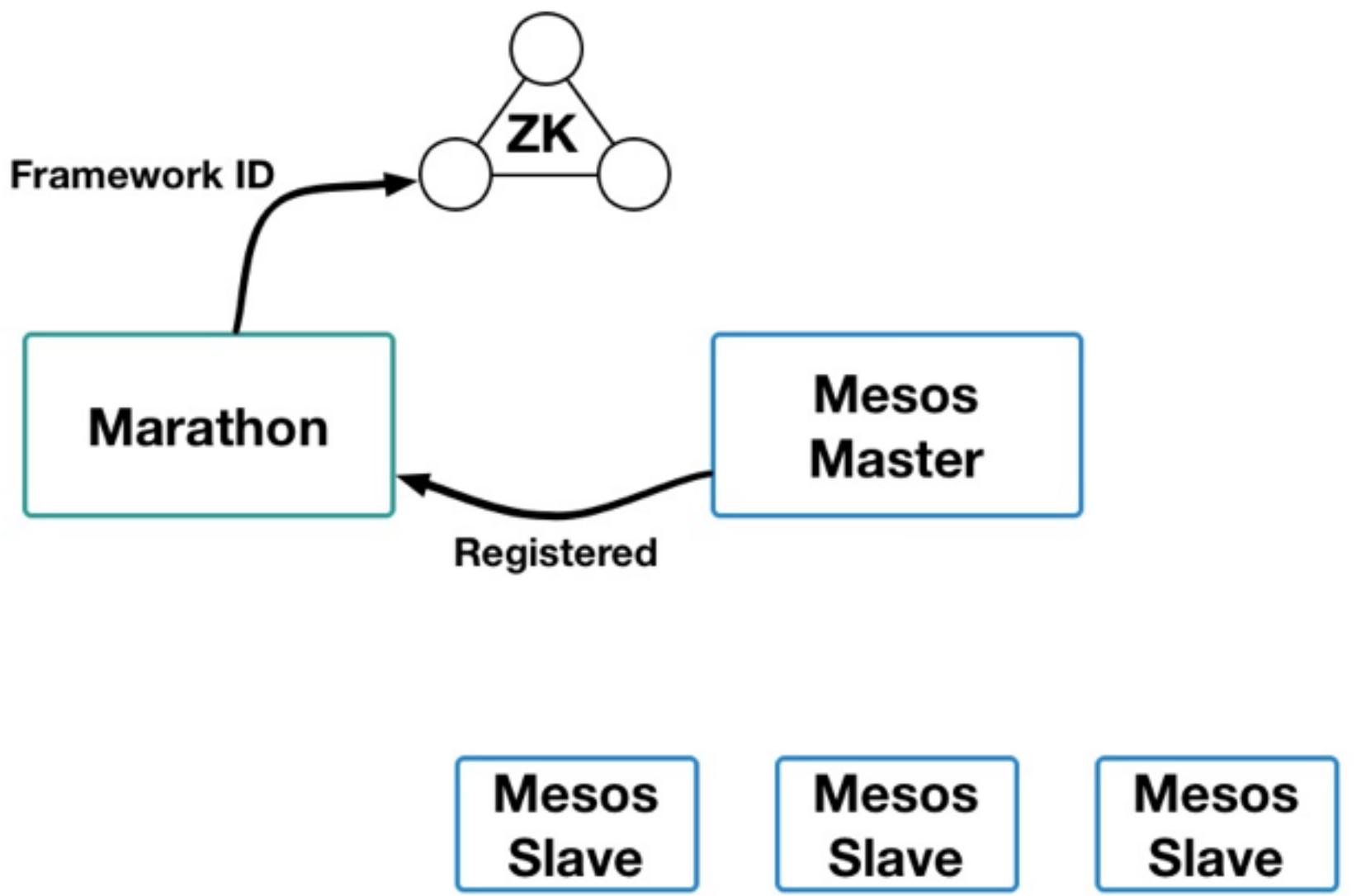
- **resource** == anything a task/executor consumes in order to do their work
- standard resources: cpu, mem, disk, ports
- at its core: DRF (Dominant Resource Fairness) algorithm, for fair sharing across-resource types

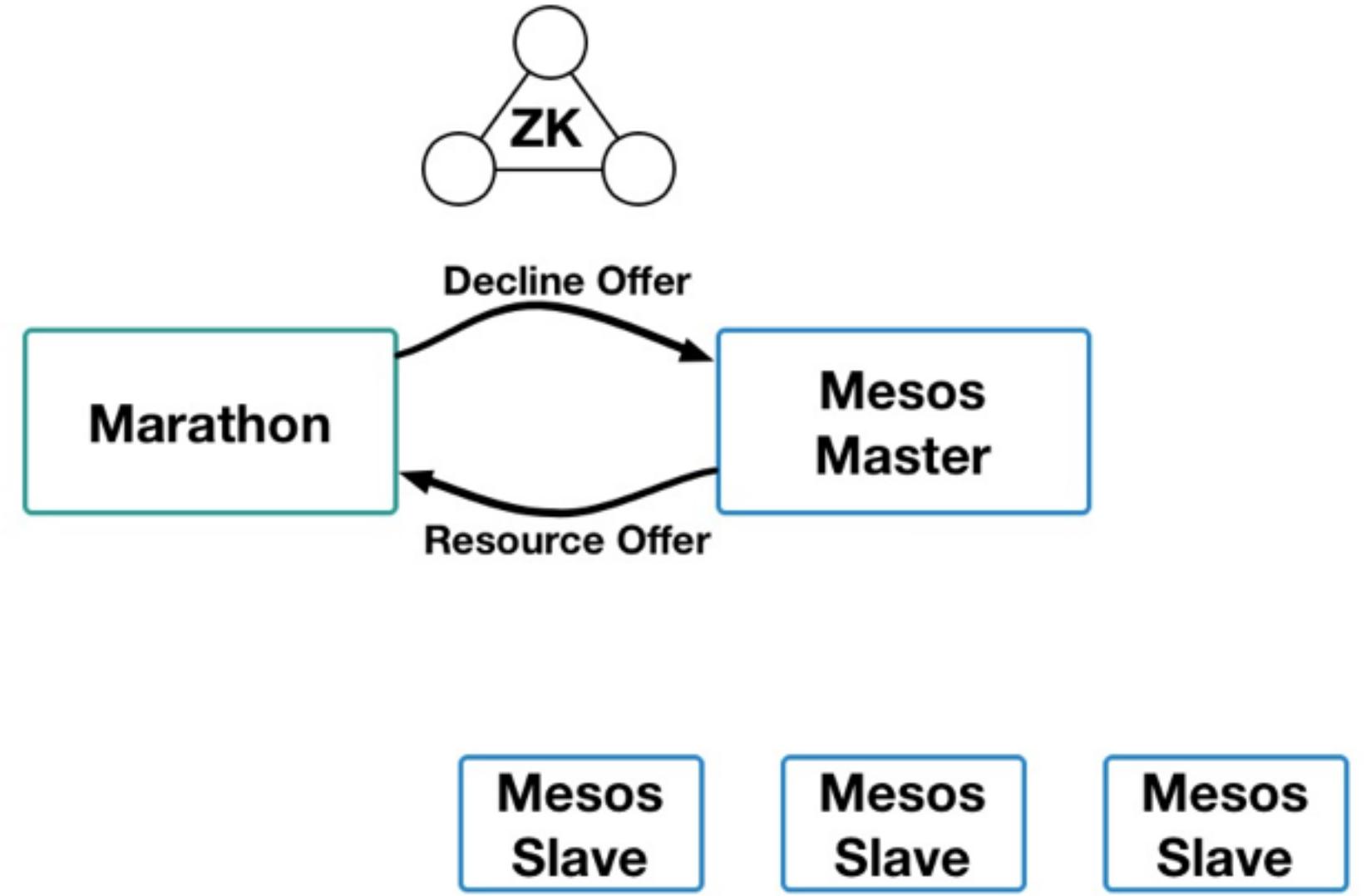


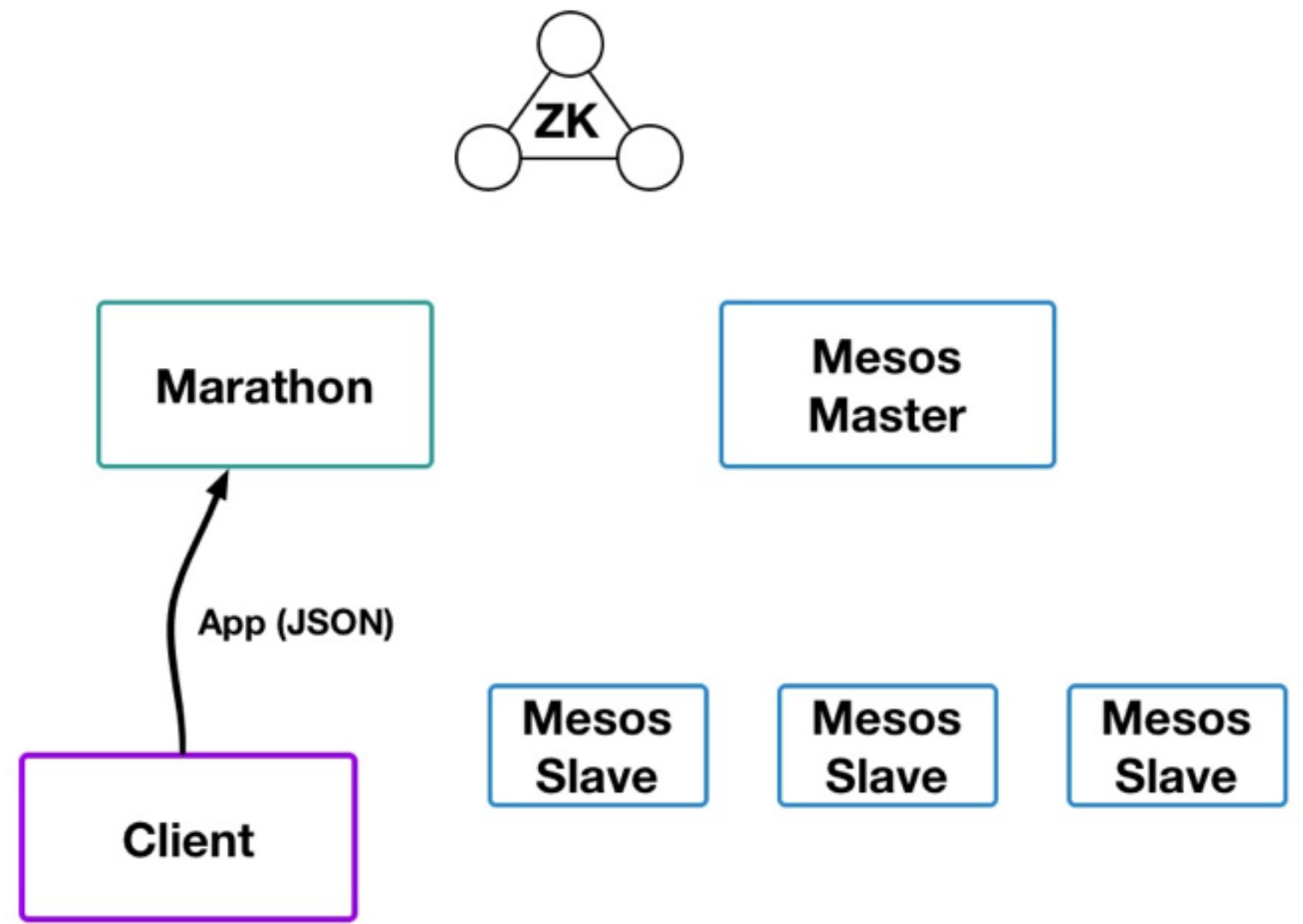


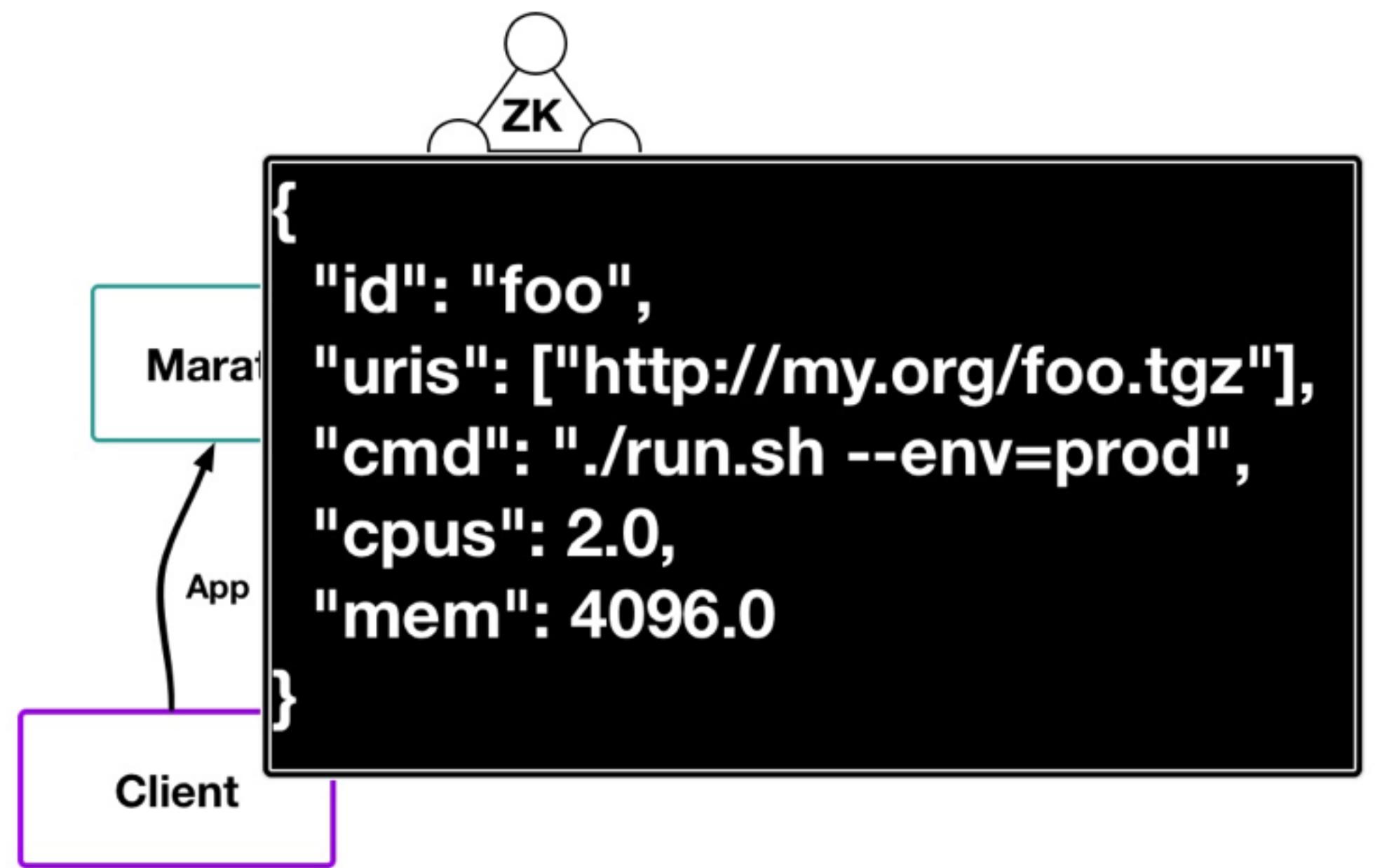


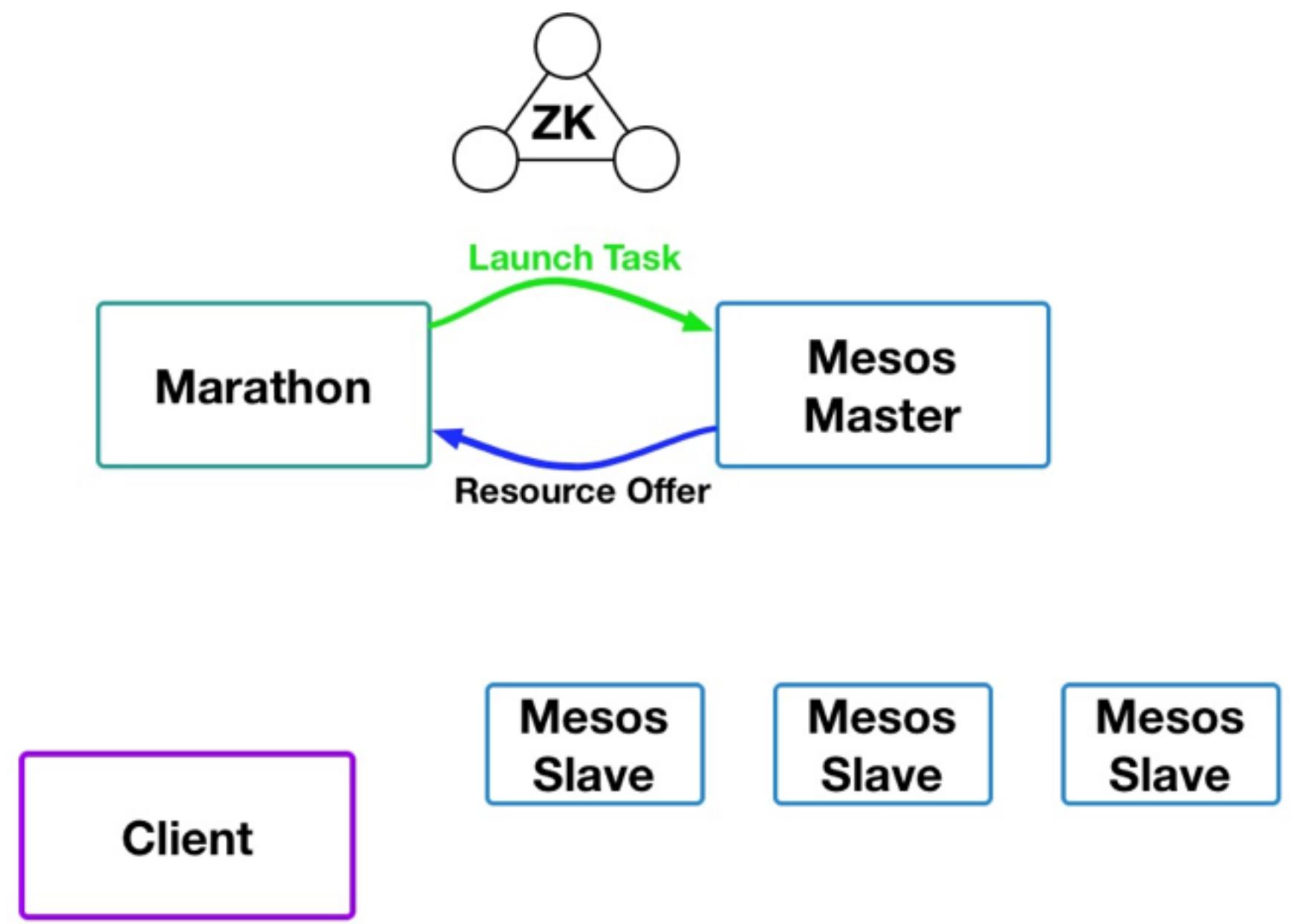


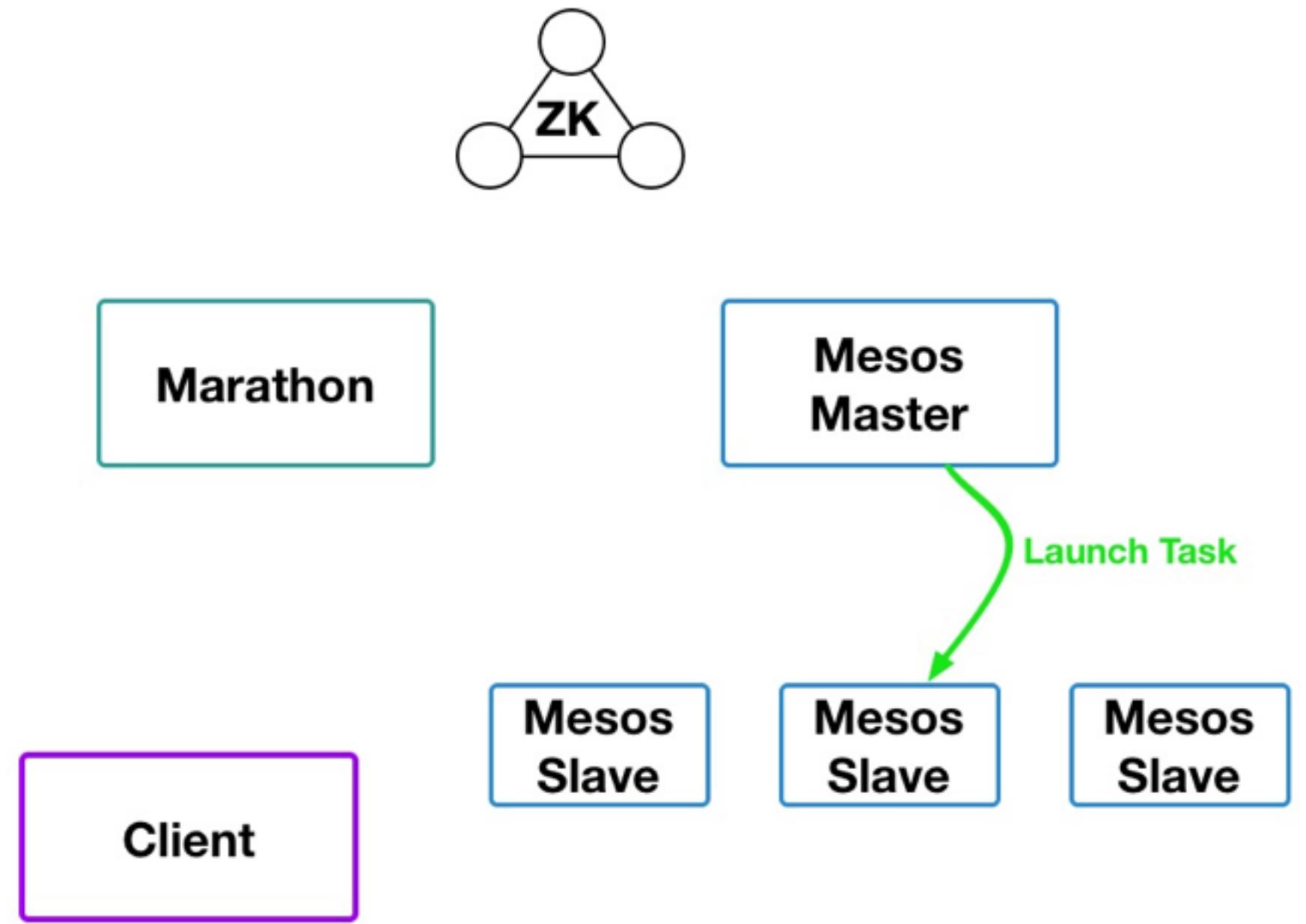


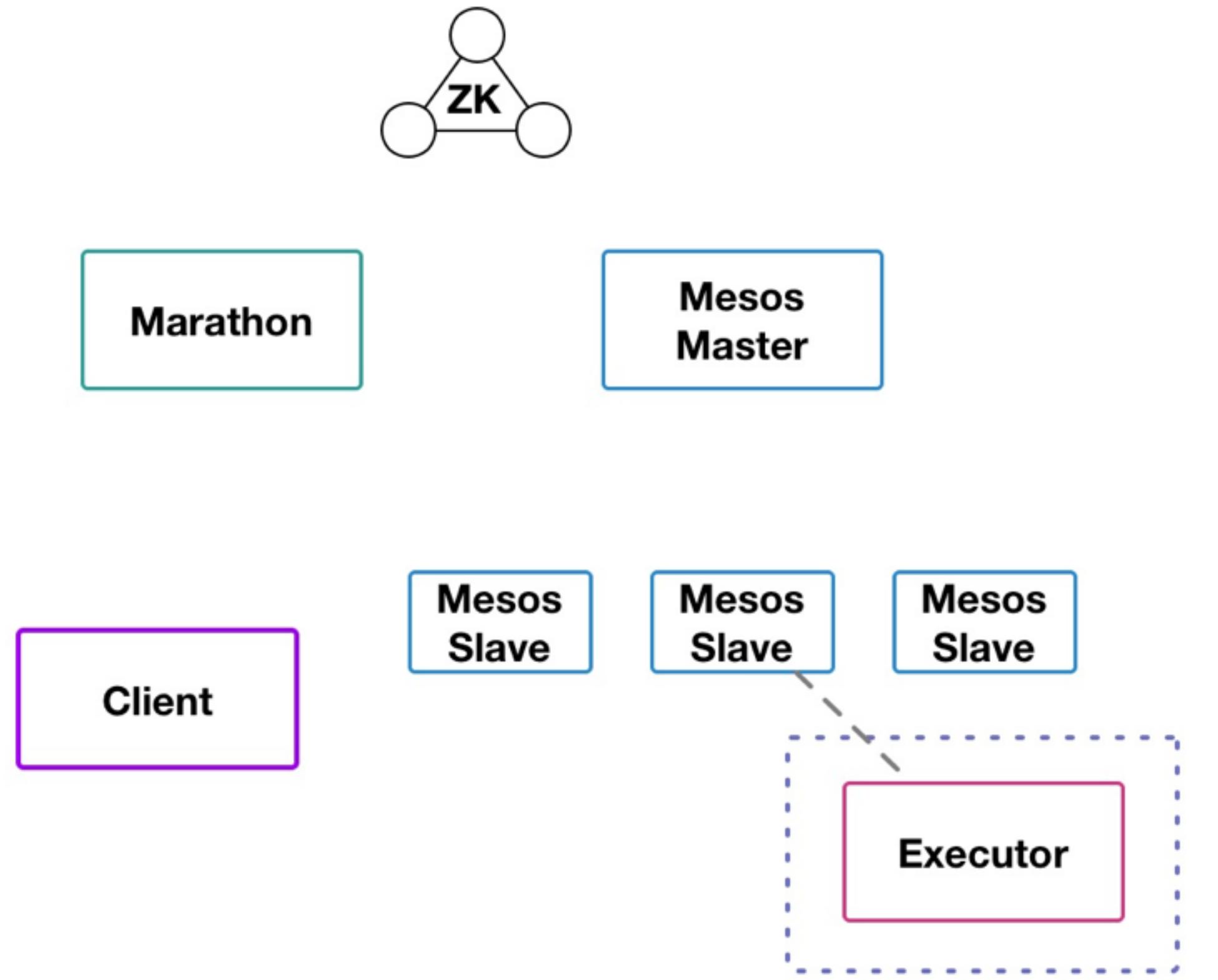


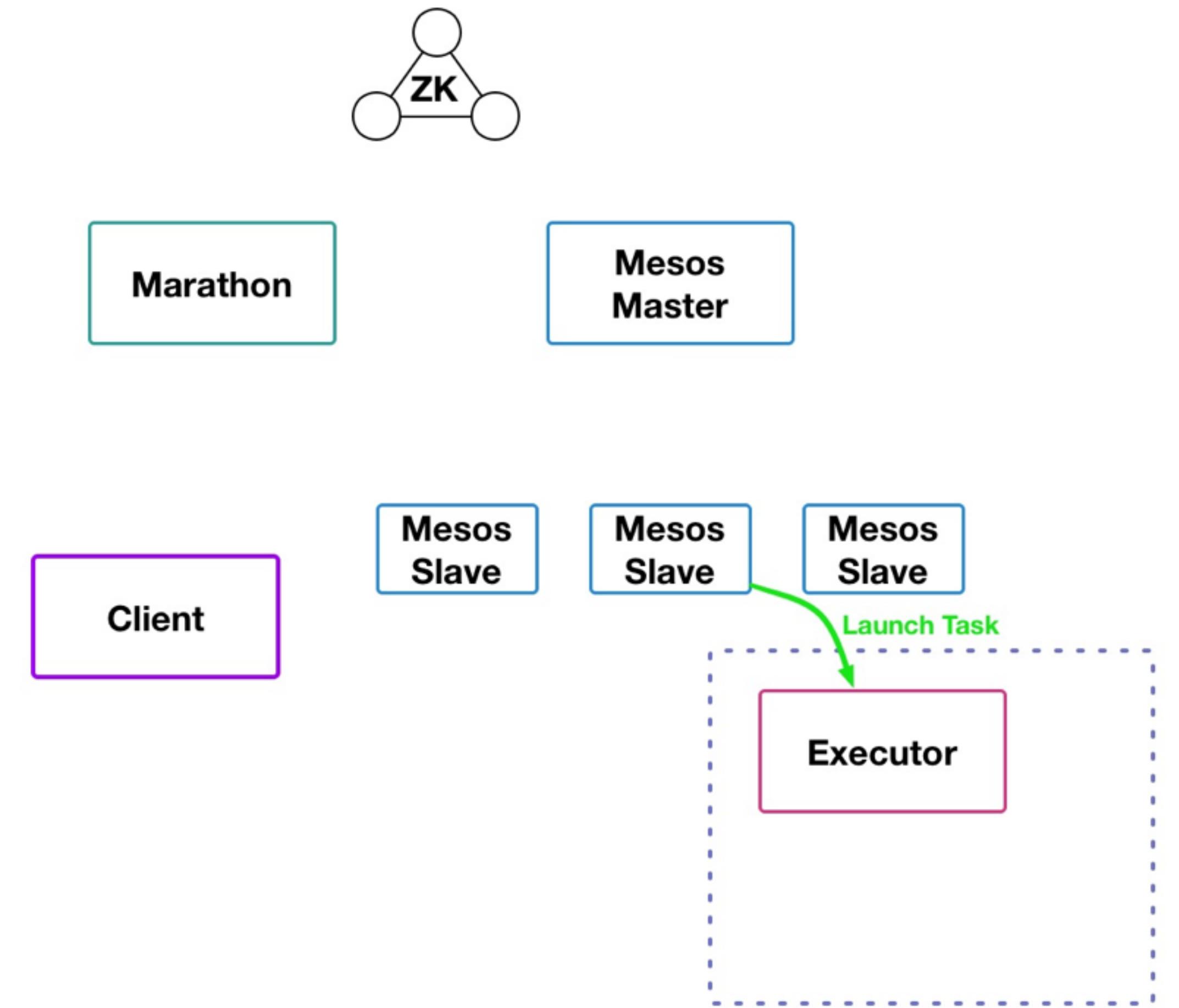


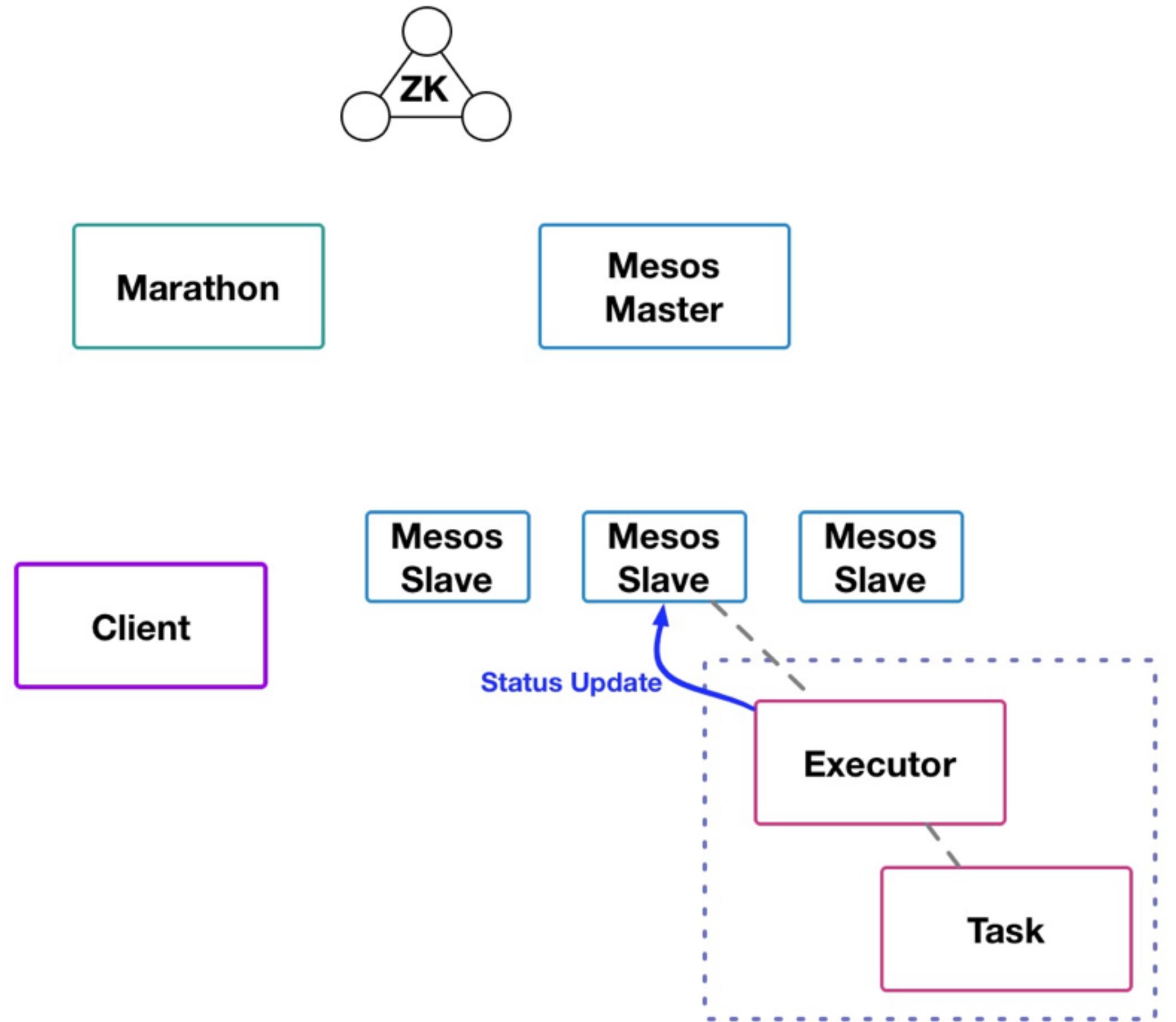


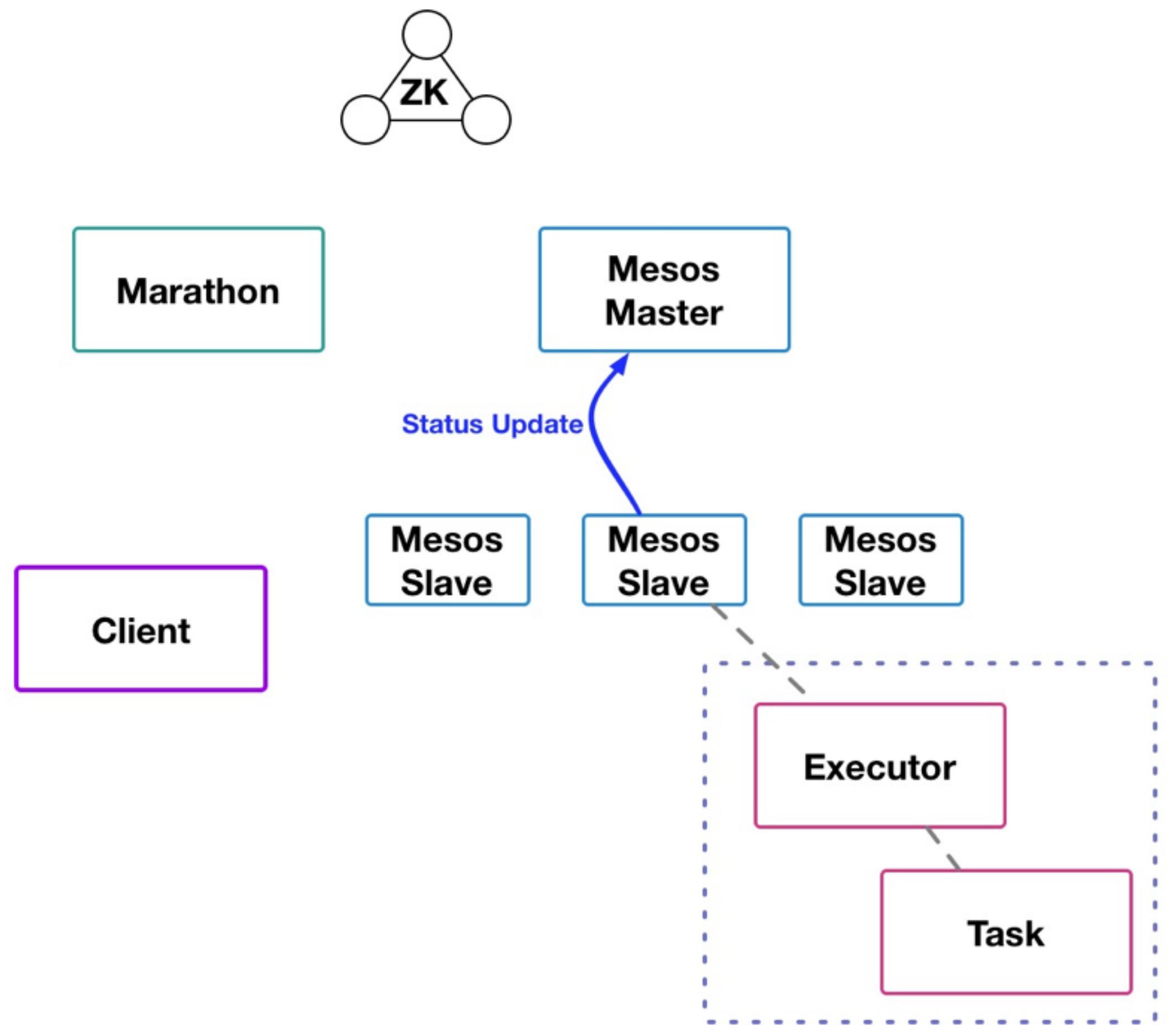


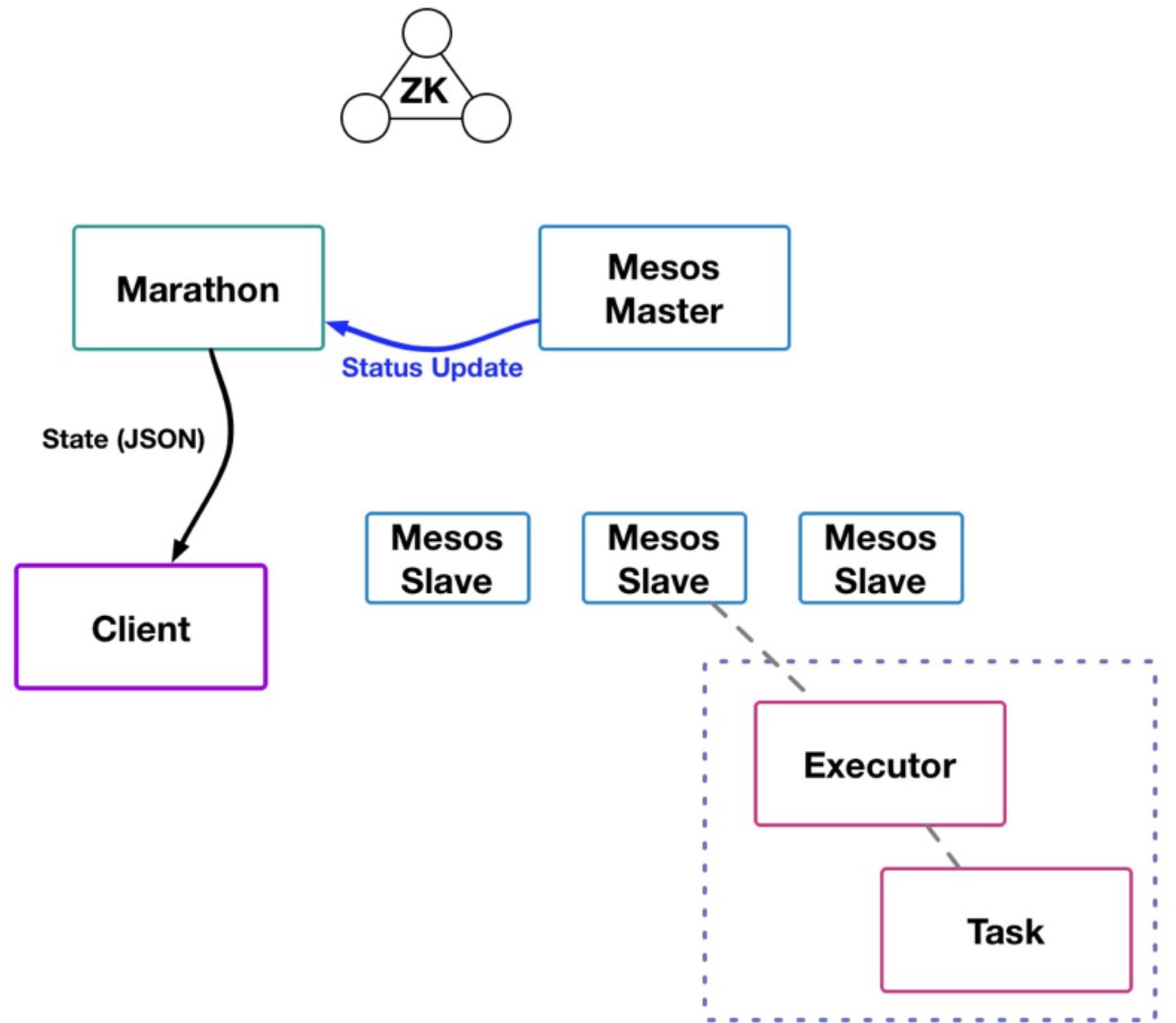


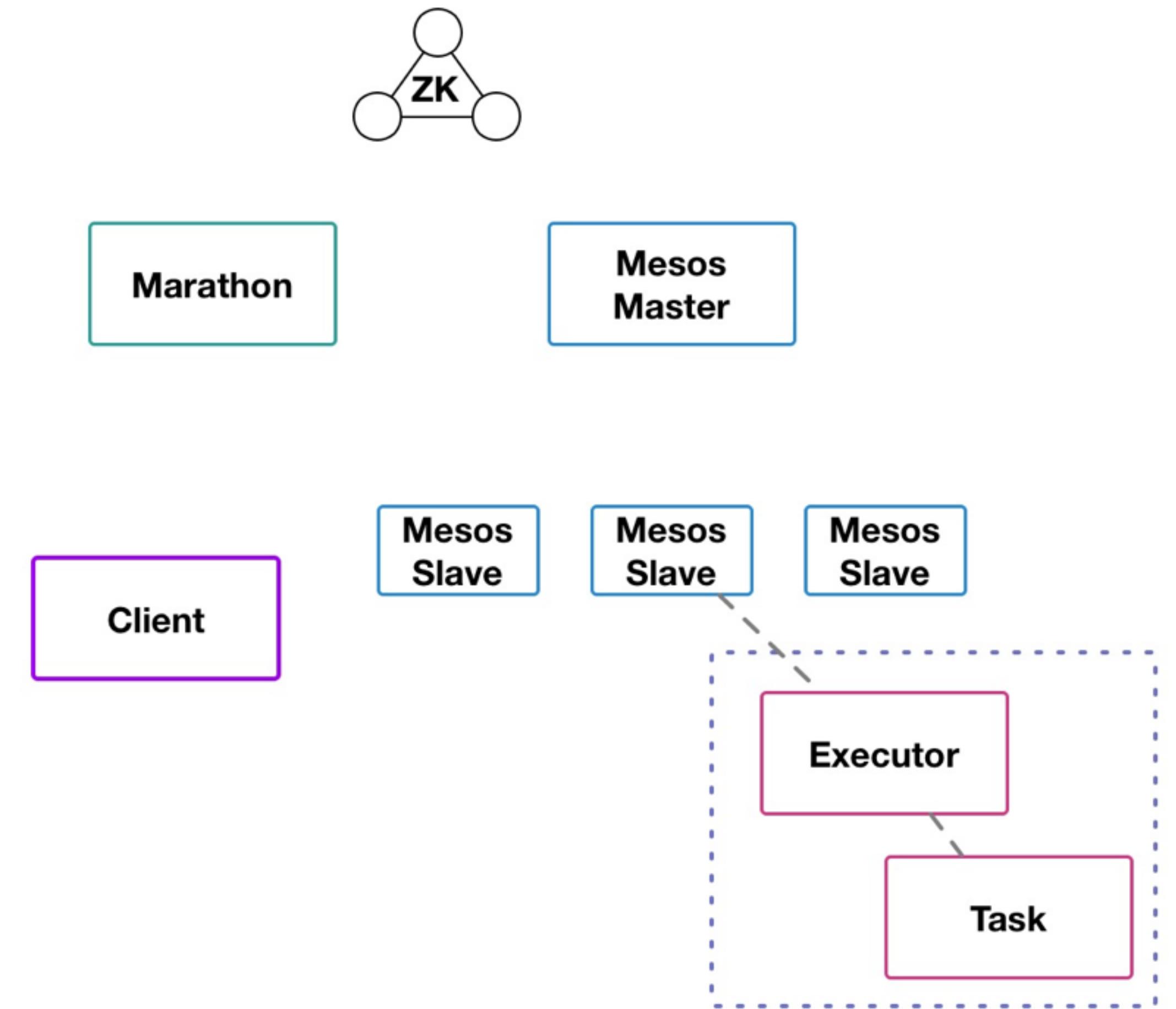










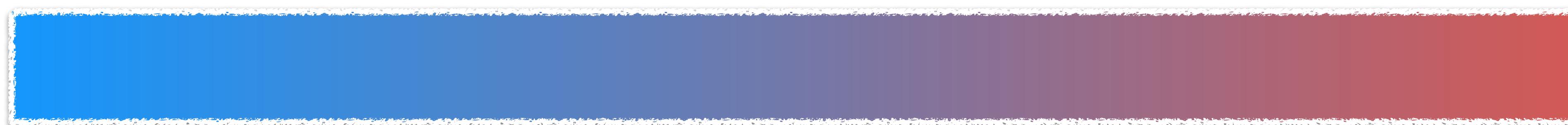


WORKLOADS ...

batch

streaming

PaaS



CHRONOS

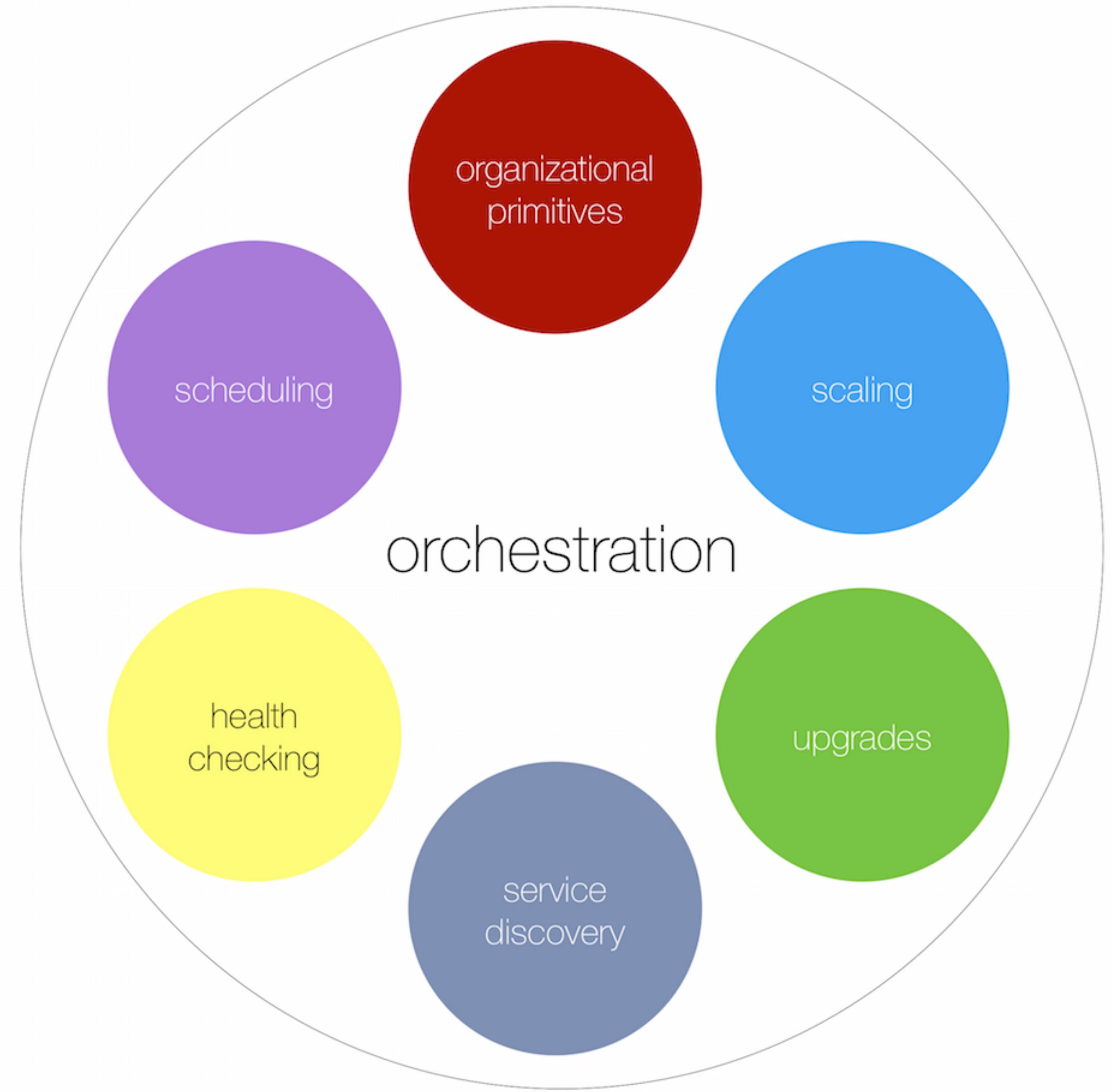


MapReduce



samza





MESOS FUTURE TOPICS

- Dynamic Reservations (mesos 0.25)
- Persistent Volumes (mesos 0.23)
- IP per Container (mesos 0.26)
- Quota (??)
- Maintenance Primitives (??)

MARATHON

An init System for datacenters

- starts instances of a long-running service somewhere in the cluster, for example, as Docker containers
- restarts the instances if they crash
- provides composition primitives
- supports health checks
- supports rolling upgrades

MARATHON 101

BASICS

- application is a blueprint for mesos tasks
- group is a container for apps and groups

UI

- HTTP Rest Endpoint
- Web UI
- DCOS CLI

TEAM PLAYER

- Integrates nicely into the DCOS ecosystem
- Doesn't try to solve everything itself

MARATHON APPLICATION JSON EXAMPLE

```
{  
  "id": "webserver",  
  "cmd": "python3 -m http.server 8080",  
  "cpus": 0.5,  
  "mem": 32.0,  
  "container": {  
    "type": "DOCKER",  
    "docker": {  
      "image": "python:3",  
      "network": "BRIDGE",  
      "portMappings": [  
        { "containerPort": 8080, "hostPort": 0 }  
      ]  
    }  
  },  
  "acceptedResourceRoles": [  
    "slave_public"  
  ],  
  "constraints": [  
    ["hostname", "UNIQUE"]  
  ]  
}
```

MARATHON APPLICATION WEB-UI EXAMPLE

The screenshot shows the Marathon web UI for editing an application named "/webserver".

General Application Settings:

- ID:** /webserver
- CPUs:** 0.1
- Memory (MB):** 64
- Disk Space (MB):** 0
- Instances:** 1

Command: (Empty text area)

Current Version - 21 Oct: (Text indicating the current version and date)

Configuration Options:

- Tasks
- Configuration (selected)
- Constraints
- Container

Docker Container Settings:

- Image:** nginx
- Network:** Bridged

Privileges:

- Extend runtime privileges to this container

Select to give this container access to all devices on the host

Port Mappings:

Container Port	Host Port	Service Port	Protocol
80	0	10000	Select

Actions:

- Suspend
- Scale
- Start App
- Destroy App
- Refresh
- Edit

MARATHON EXAMPLE APPLICATION

id: "webserver"

Naming

args: [sample, keywords]

env: [... twitter api auth, kafka brokers ...]

container:

type: "DOCKER"

image: "python:3"

Run

cpus: 0.5

mem: 2048

ports: [8080, 8081, 9090]

Resources

instances: 123

Scale

healtChecks:

type: "HTTP",

path: "/health"

Health

dependencies: ["/db/memsql"]

Dependencies

minimumHealthCapacity: 0.5

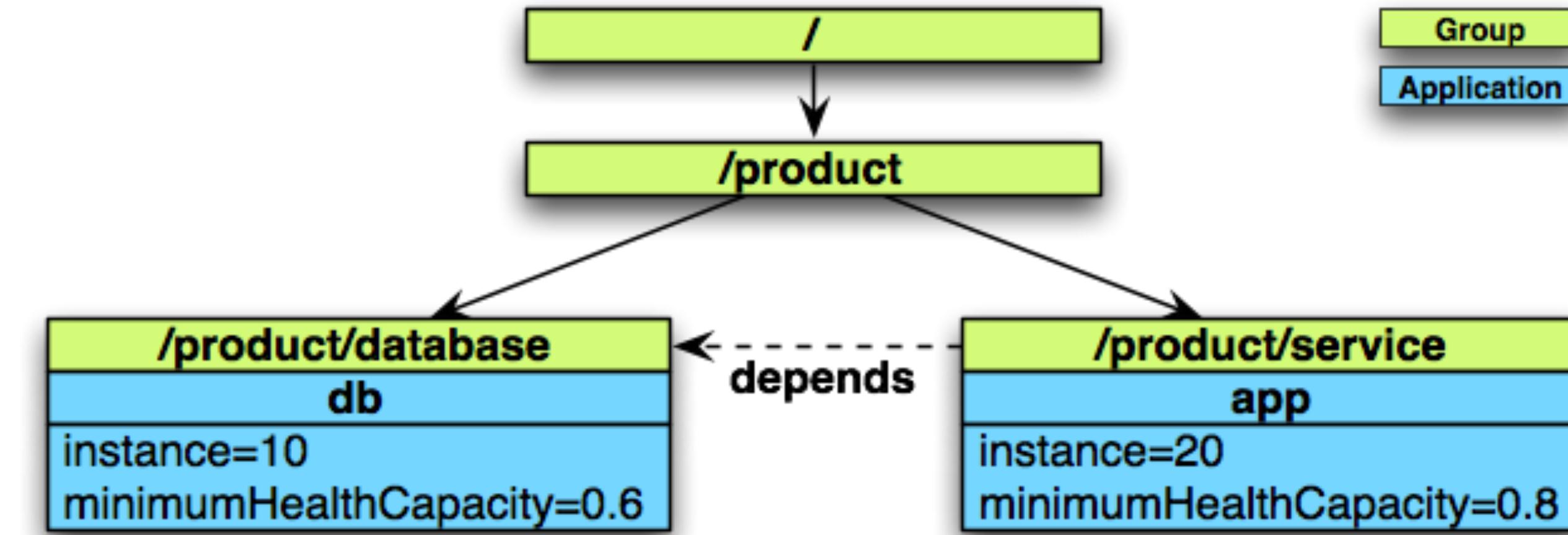
UpgradeStrategy

constraints: [[gpu LIKE "NVD-*"], [rack GROUP_BY]]

Constraints

...

MARATHON GROUP



- Groups can hold zero or more Apps or Groups
- Groups can build a tree
- Dependencies can be expressed on App and Group level
- Deployments can be executed on the level of Groups
- Example Use Cases
 - map application tiers to groups
 - map organization tiers to groups

SERVICE DISCOVERY

- Well known port: marathon_lb, bamboo, ...
- DNS: MesosDNS, Consul, Weave-DNS, ...
- Programmatic: zookeeper, etcd, ...

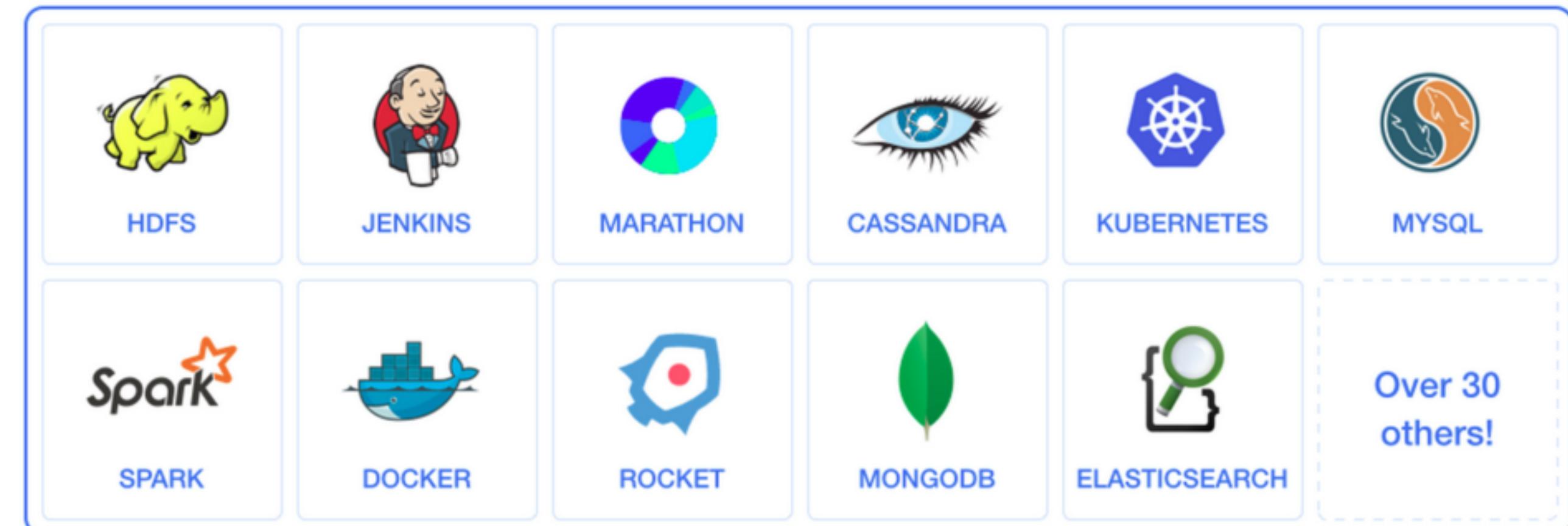
DATA CENTER OPERATING SYSTEM (DCOS)

DCOS IS A DISTRIBUTED OPERATING SYSTEM

- kernel == Apache Mesos, scaling to 10,000 of nodes
- fault-tolerant in all components, rolling upgrades throughout
- containers first class citizens (LXC, Docker)
- local OS per node (container enabled)
- scheduling (long-lived, batch)
- service discovery, monitoring, logging, debugging

Services & Containers

Your favorite services, container formats, and those yet to come.



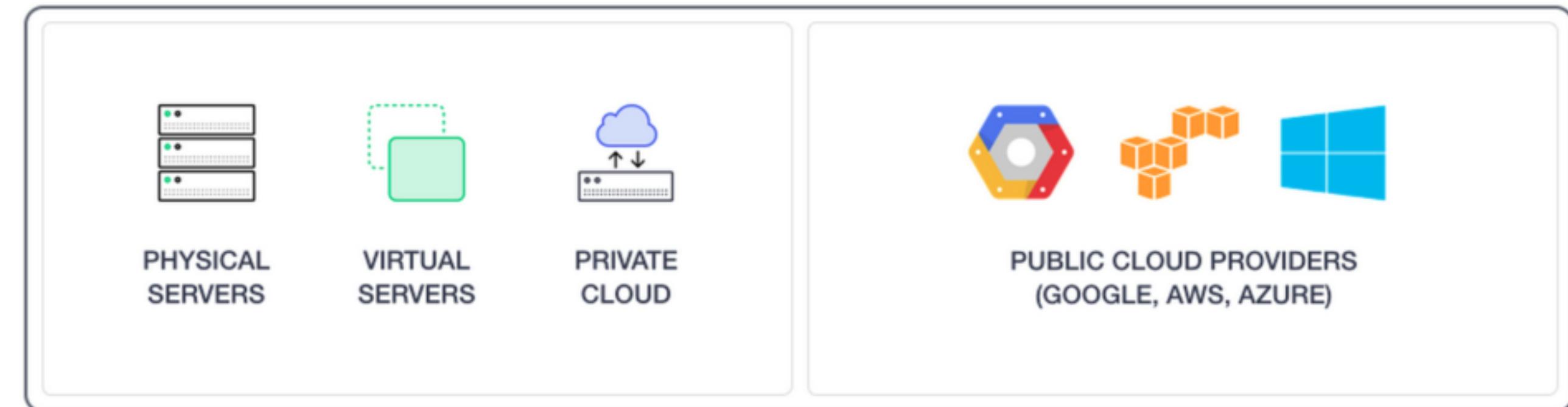
Mesosphere DCOS

Runs distributed apps anywhere as simply as running apps on your laptop.



Any Infrastructure

Build apps once on DCOS, and run it anywhere.



BENEFITS

- Run stateless services such as Web servers, app servers (via Marathon) and stateful services like Crate, Kafka, HDFS, Cassandra, ArangoDB etc. together on one cluster
- Dynamic partitioning of your cluster, depending on your needs (business requirements)
- Increased utilization (10% → 80% and more)

Dashboard

mh9-sandbox
52.25.91.69

Dashboard Services Nodes Mesosphere DCOS v.1.0.1

CPU Allocation: 30% (4.72 of 16 Shares)

Memory Allocation: 14% (8 GiB of 55 GiB)

Task Failure Rate: 0% (Current Failure Rate)

Services Health: Elasticsearch Healthy, Kubernetes Healthy

Tasks: 7

Nodes: 4 Connected Nodes

DCOS CLI

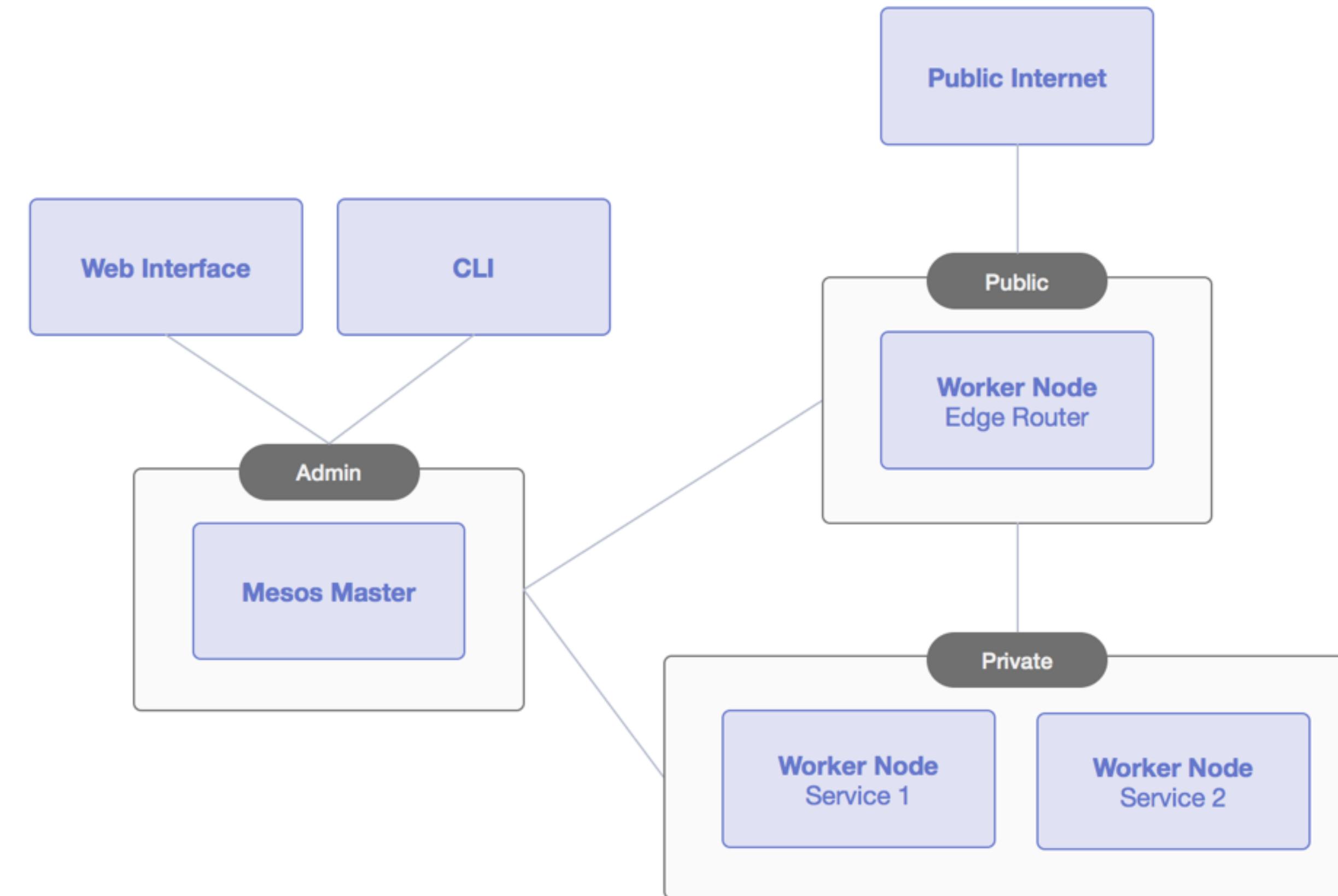
```
~/sandbox/dcos/ccm/dcos $ dcos package list
NAME      VERSION      APP      COMMAND  DESCRIPTION
elasticsearch  0.2.0    /elasticsearch  ---  DCOS implementation of the Mesos-Elasticsearch framework
kubernetes  v1.0.5-v0.6.4-alpha  /kubernetes  ---  Manage a cluster of Linux containers as a single system to accelerate Dev and simplify Ops.
~/sandbox/dcos/ccm/dcos $ dcos help
Command line utility for the Mesosphere Datacenter Operating System (DCOS). The Mesosphere DCOS is a distributed operating system built around Apache Mesos. This utility provides tools for easy management of a DCOS installation.

Available DCOS commands:

  config      Get and set DCOS CLI configuration properties
  help        Display command line usage information
  marathon    Deploy and manage applications on the DCOS
  node        Manage DCOS nodes
  package     Install and manage DCOS packages
  service     Manage DCOS services
  task        Manage DCOS tasks

Get detailed command description with 'dcos <command> --help'.
~/sandbox/dcos/ccm/dcos $ _
```

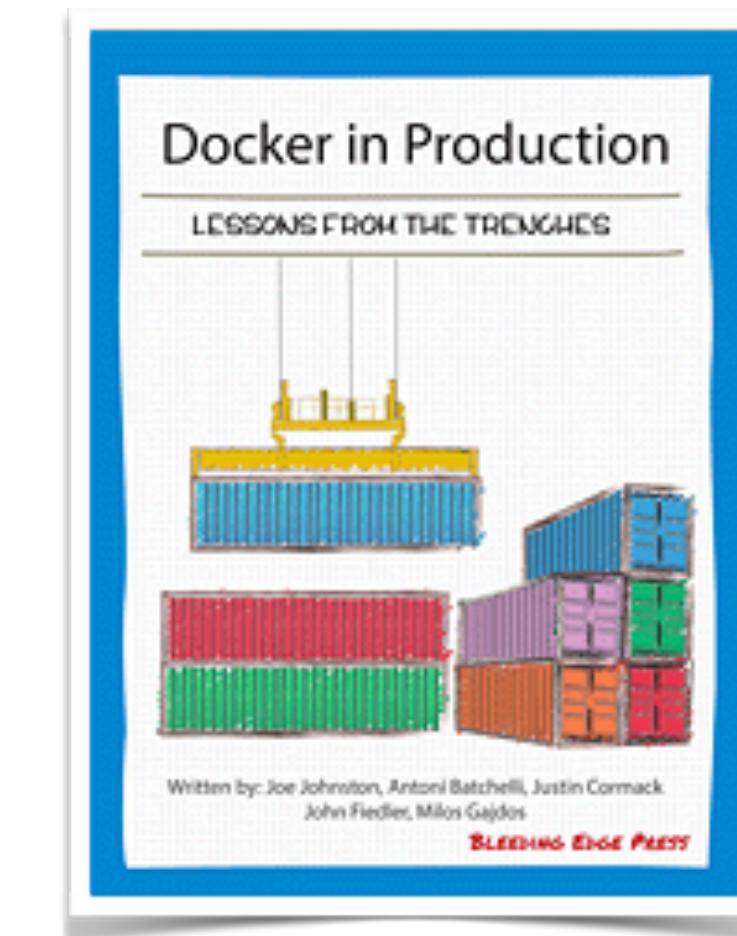
DCOS ARCHITECTURE



LEARNING RESOURCES

WHERE CAN I LEARN MORE?

<http://shop.oreilly.com/product/9781939902184.do>



<http://shop.oreilly.com/product/0636920035671.do>



WHERE CAN I LEARN MORE?

<http://shop.oreilly.com/product/0636920039952.do>



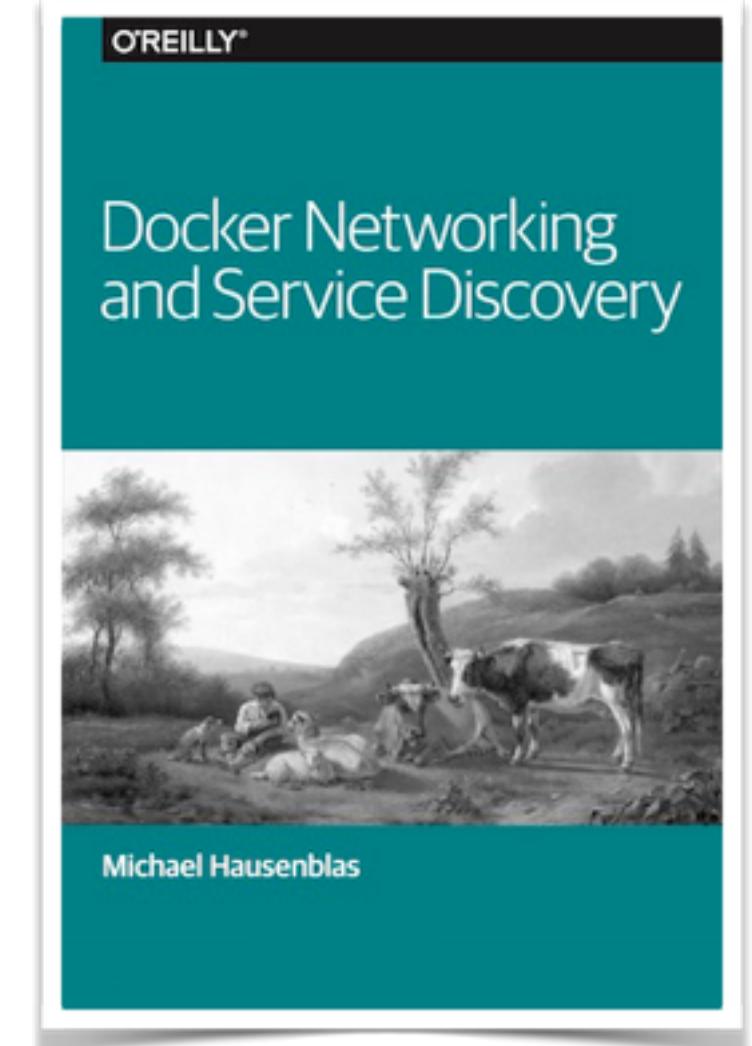
<https://manning.com/books/mesos-in-action>



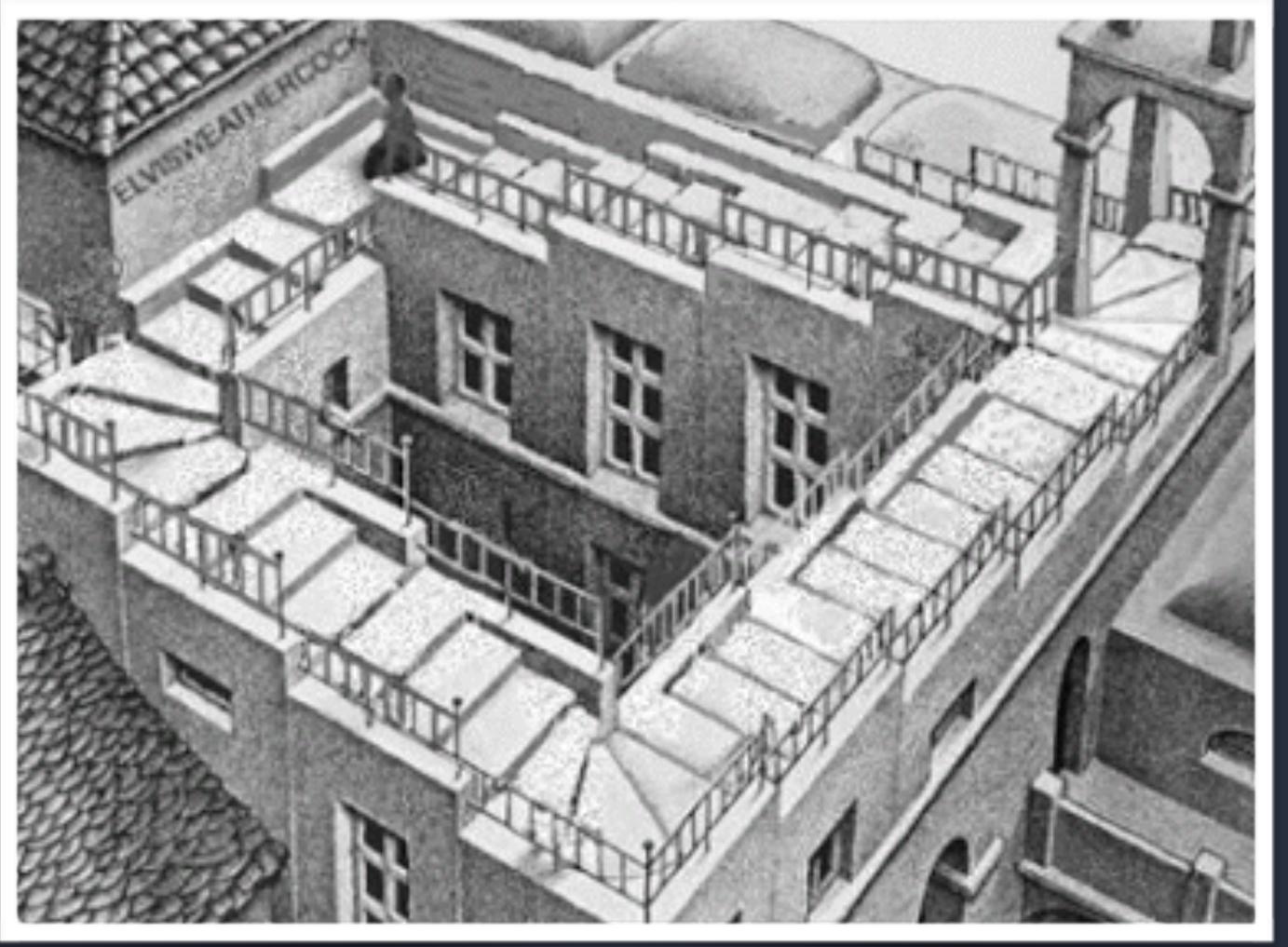
WHERE CAN I LEARN MORE?

<http://p24e.io>

The screenshot shows the homepage of the p24e website. At the top, there's a navigation bar with links for 'Guides', 'Technologies', 'About', and 'Contributors'. Below the navigation is a section titled 'Latest updates' which lists several blog posts with their publication dates and small thumbnail images. To the right of this section is a quote: "'programmable infrastructure (p24e) is the concept of applying methods and tooling established in software development onto the management of IT infrastructure. This includes but is not limited to: automation, versioning, APIs, immutability and agile techniques'" attributed to 'The Elders of the Internet'. Below the update section is a 'Technologies' section divided into four categories: 'Base Images' (represented by a Linux icon), 'Configuration Management' (represented by a code icon), 'Container Runtime' (represented by a gear icon), and 'Host OS' (represented by a server icon). Each category has a brief description and a list of specific technologies or projects.



HANDS-ON



LABS

- Container 101
- Apache Mesos + Marathon