Distriubted Operation System and Use Cases

Yanqi Ma in Bamberg, 2017

Agenda

- Some ambiguous concepts
- Traditional distributed systems
- Pros and cons of traditional distributed systems
- Distributed operation system concept
- Applications of distributed operation system
- My experience with distributed operation system

Some ambiguous concepts

Concurrency VS. Parallelism

A condition that exists when at least two threads are making progress

Usually software oriented, like asynchronous / reactive programming

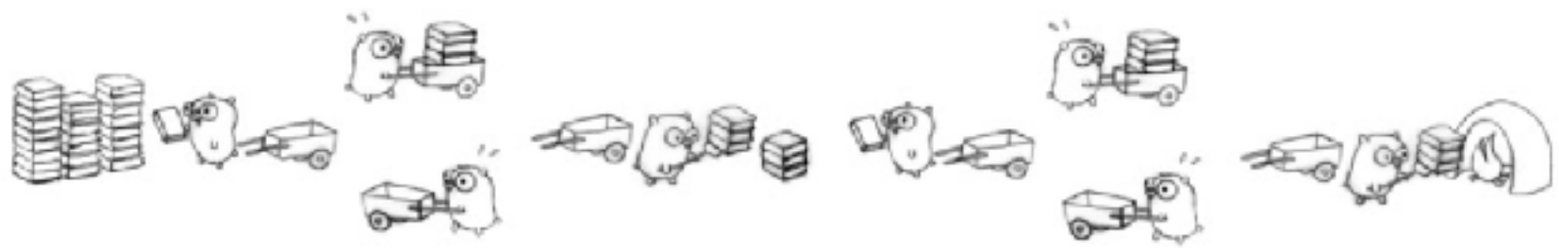
A condition that arises when at least two threads are exciting simultaneously

Usually hardware oriented, like multi-core processing or distributed computing

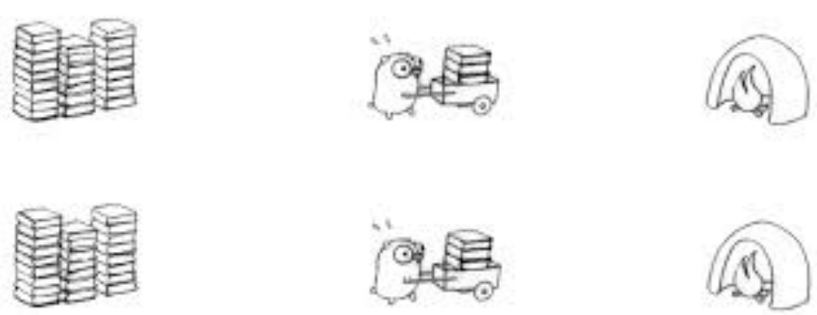
Concurrency VS. Parallelism

A Task

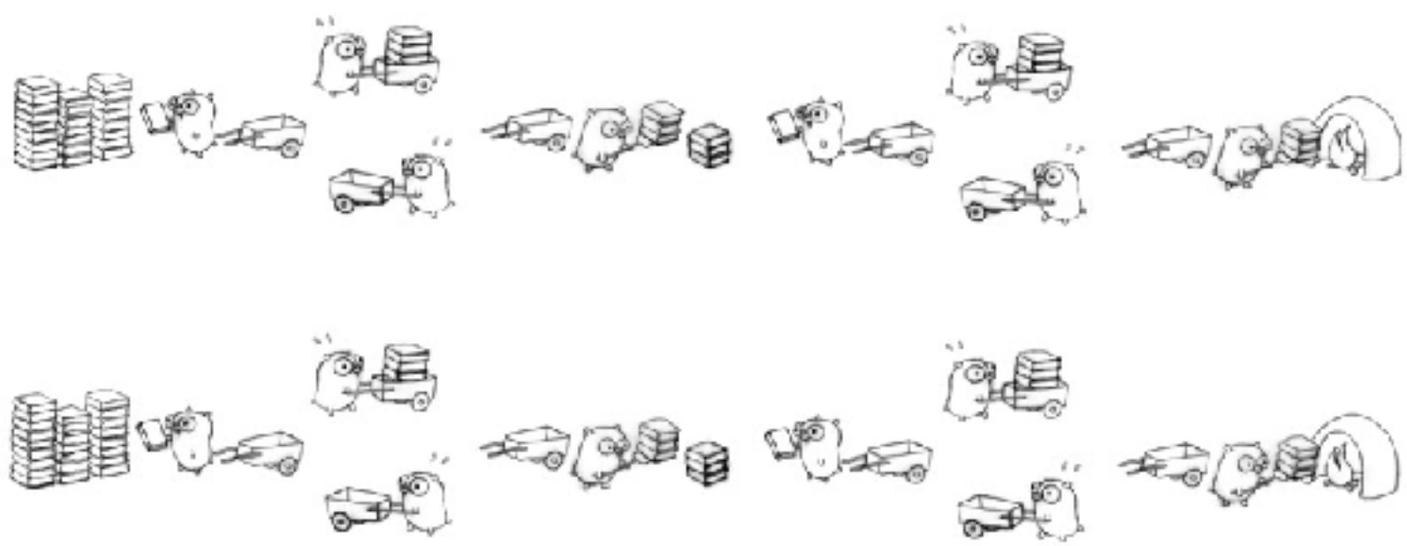
Concurrency



Parallelism



Combined



Some ambiguous concepts

- Multi-process / Multi-thread

  -> usually means single node

- Parallel computing

  -> usually means shared resources or intensive communication between the nodes, like a MPI system.

- Distributed computing

  -> usually means no shared resources or intensive communication, accepts asynchrony and partial failure, like a Hadoop system.

Traditional distributed systems

hadoop

APACHE Spark™

APACHE STORM™

samza

distributed stream processing

@humbertostreb

Flink

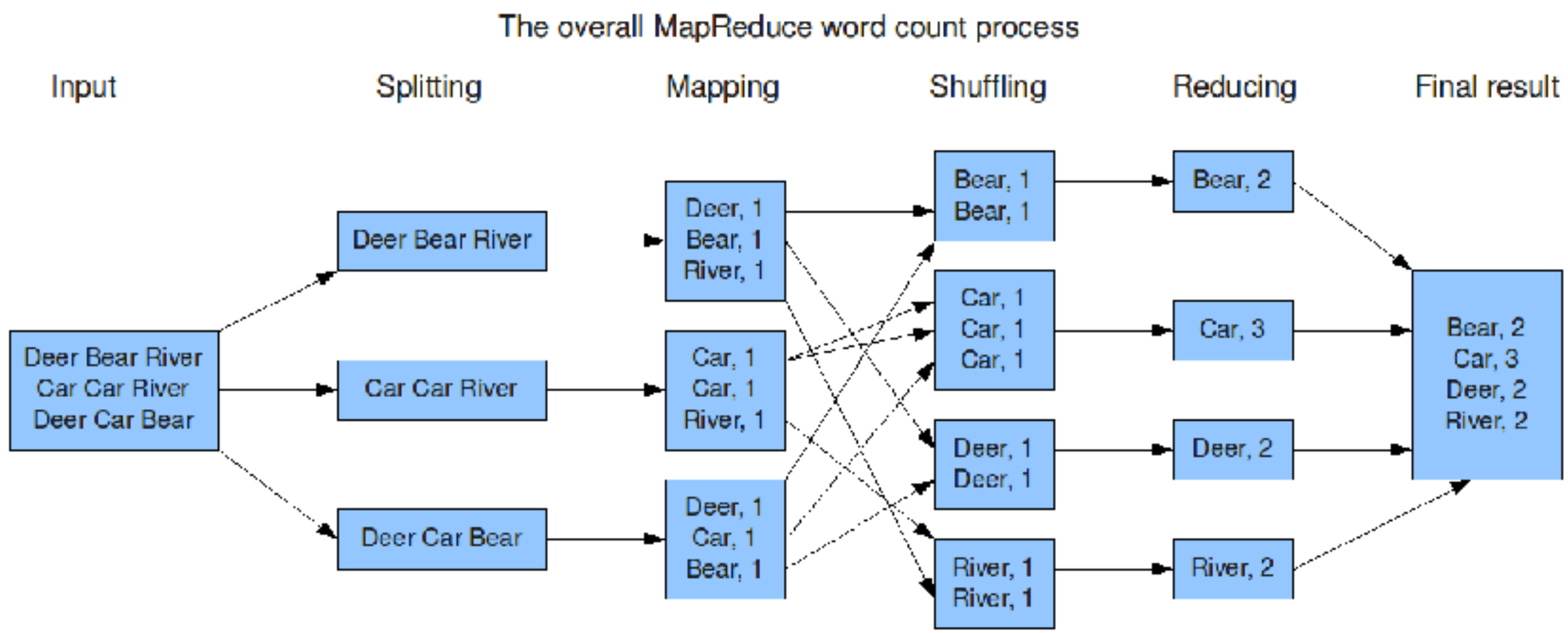
cassandra

ArangoDB

influxdb

Traditional distributed systems

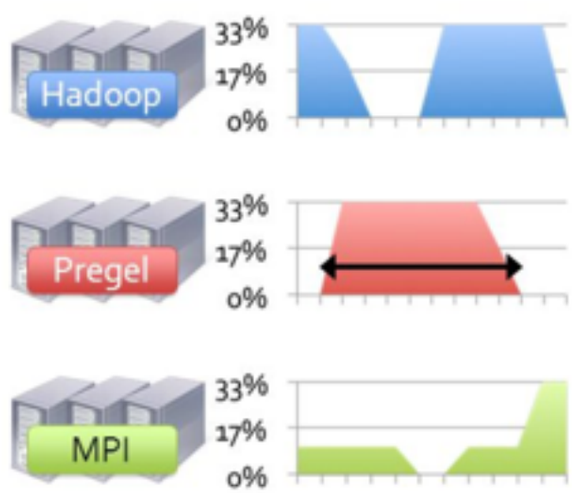
Divide and Conquer a computing task (Input data)



Big Data frameworks use resources to handle storage and computing, but who takes care of the resources ?

Wait, why caring resources ? I have nodes, I run whatever I like on them.

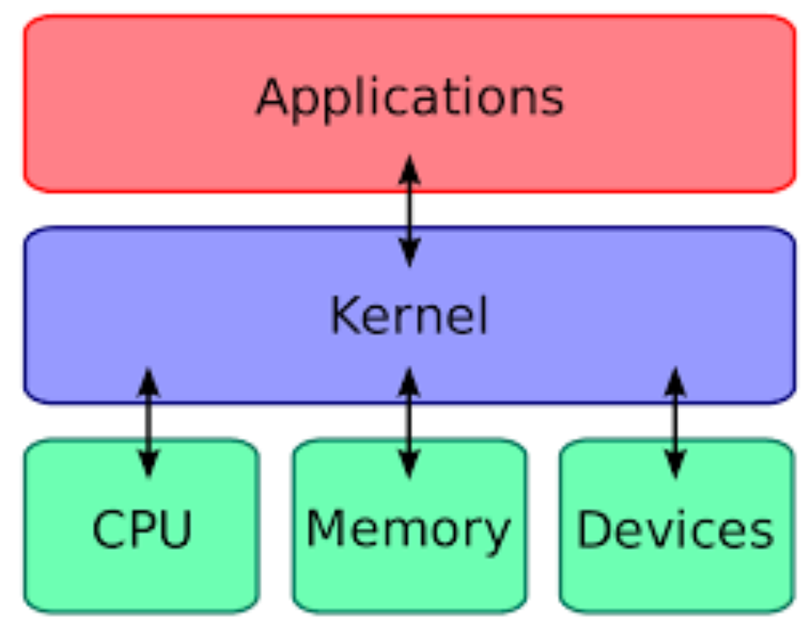
But this leads to resource waste and unexpected resource starvage. Let me illustrate.

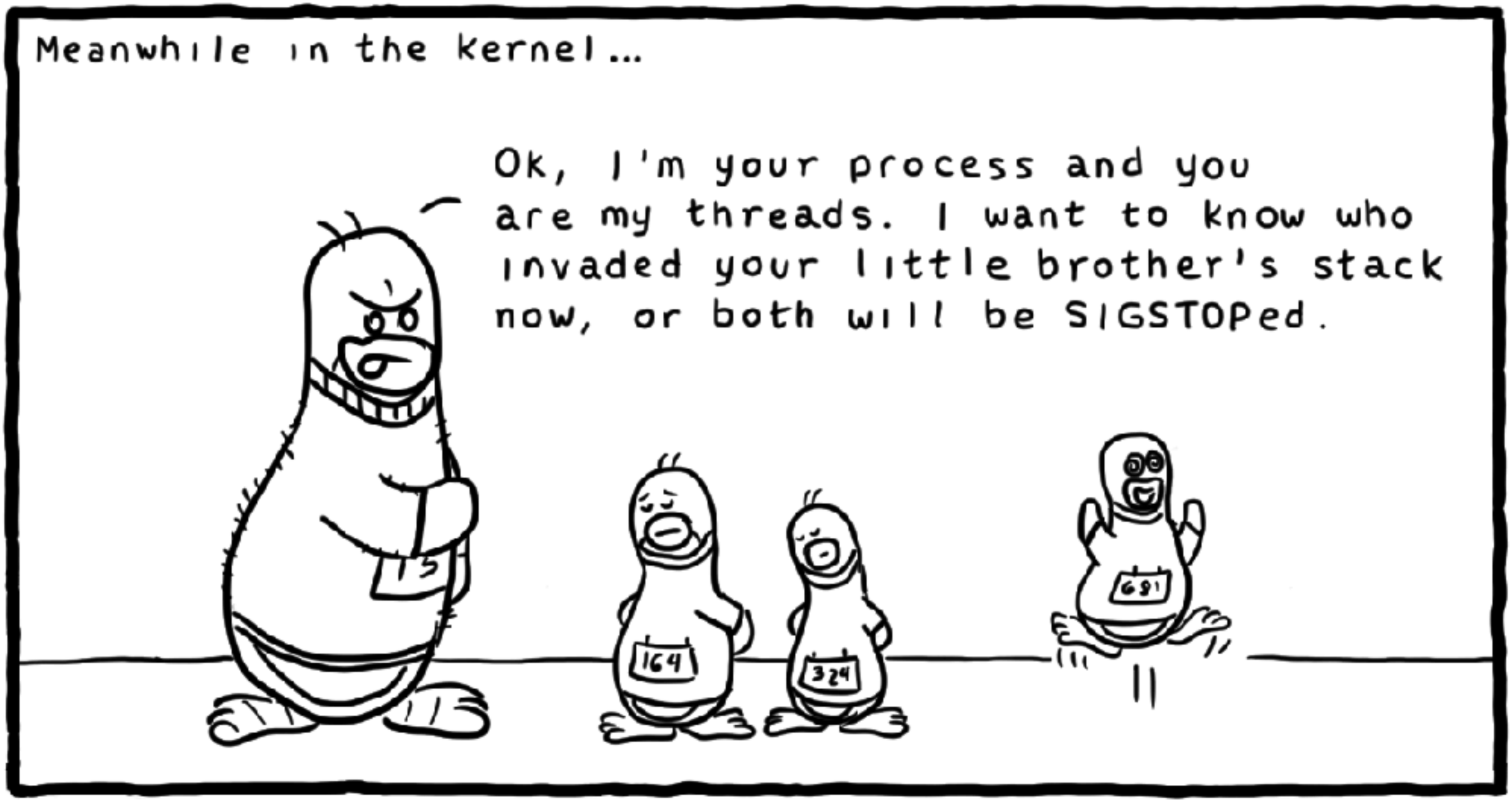


Ok what should I do ?

You need a resource manager, a kernel, such as Mesos.

A resource manager is something that works like a kernel of your operation system. It sits between hardware and applications, monitors the state of hardware and manages accesses from application to hardware. Most important, it tries to be fair !



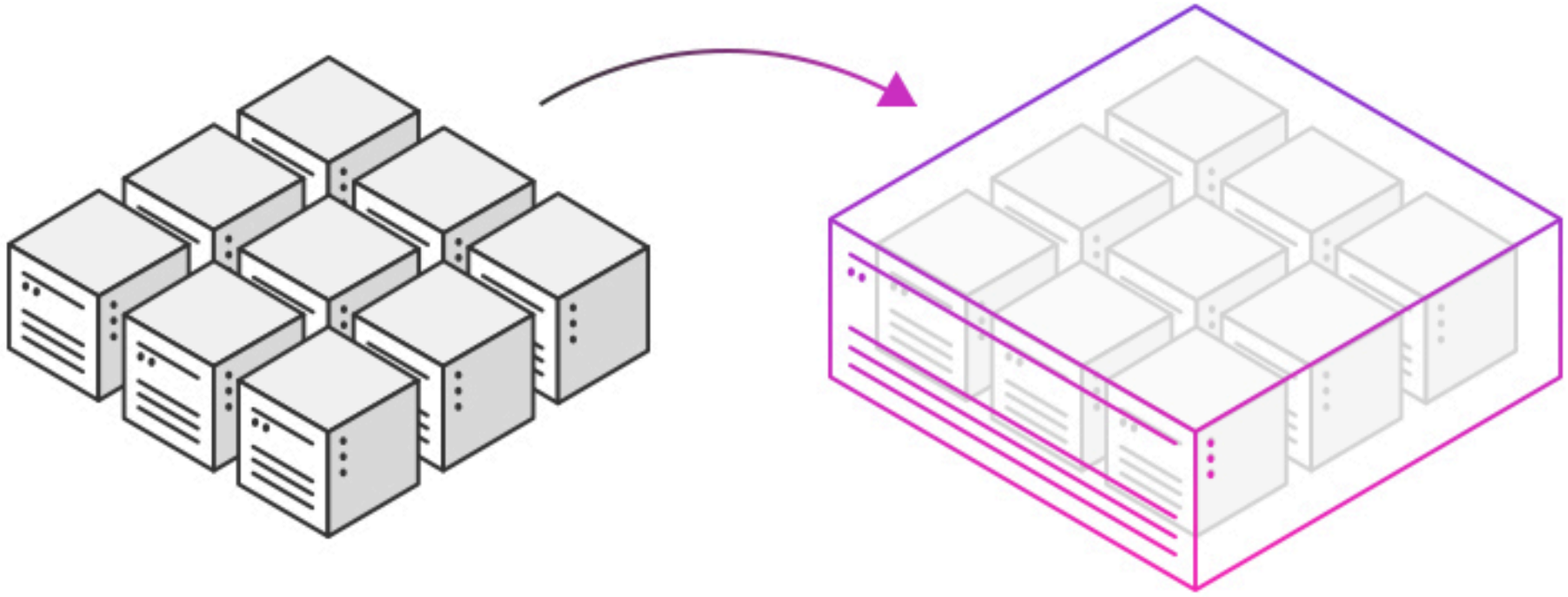


Daniel Stori {turnoff.us}

resource management framework

- abstract hardware resources to a central pool via container
- fairly allocate resources to applications.
- course grained monitoring and reaction to failure.

Mesos



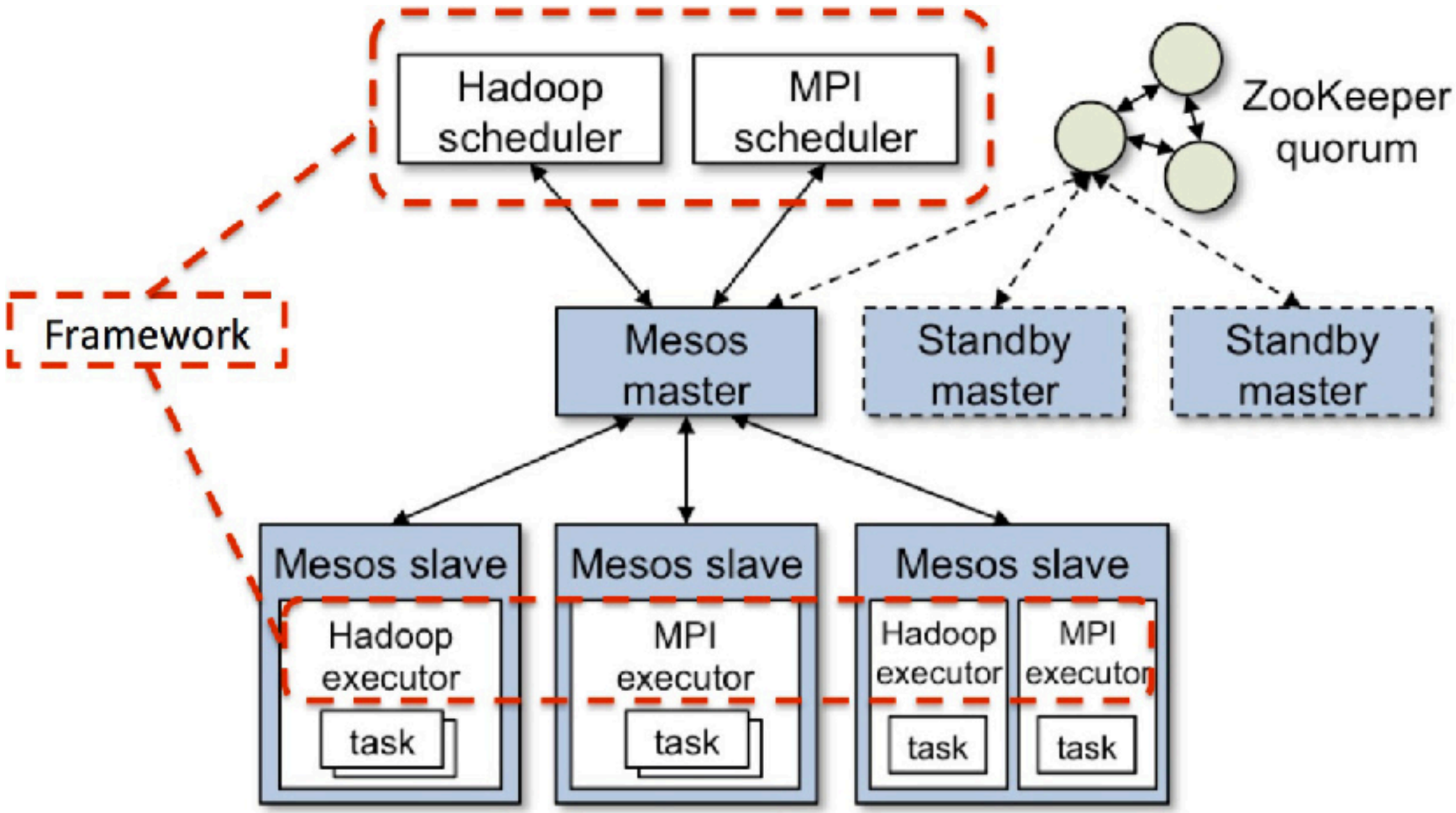
Datacenter or Cloud

Gone are the days where writing and deploying new applications means managing individual machines and static partitions.

With Mesosphere

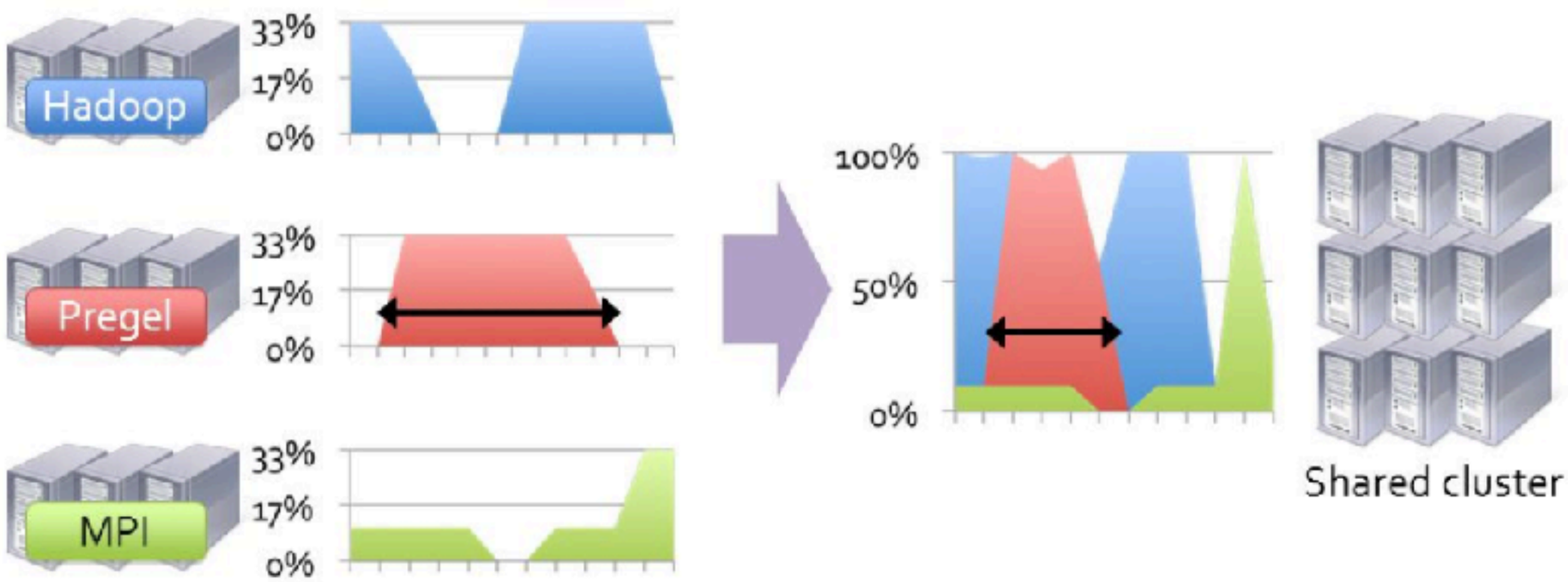
Pool your datacenter and cloud resources, so all your apps run together on the same machines —reducing complexity and waste.

How Mesos work ?



Mesos

The result of Mesos



Big Data framework + A resource manager are all I need right ? I got a kernel, I got applications.

Still not enough, just like in Linux, apart from kernel, there is also drivers, networking, file system, user interface and desktop. To make a data center run intelligently and efficiently, we need those helpers as well.

Therefore, it is time to introduce distributed operation system.

distributed operation system

- A collection of software over multiple machines, providing similar functions like a operating system — resource management, process management, fault recovery, networking and etc.
- Providing additional features like Scalability & Availability.
- Traditional distributed operation system is based on completely individual architecture and usually only stays in the Lab, such as Plan 9.

distributed operation system

DCOS is a distributed operation system for common cluster, based on Mesos and naturally support for big data frameworks.

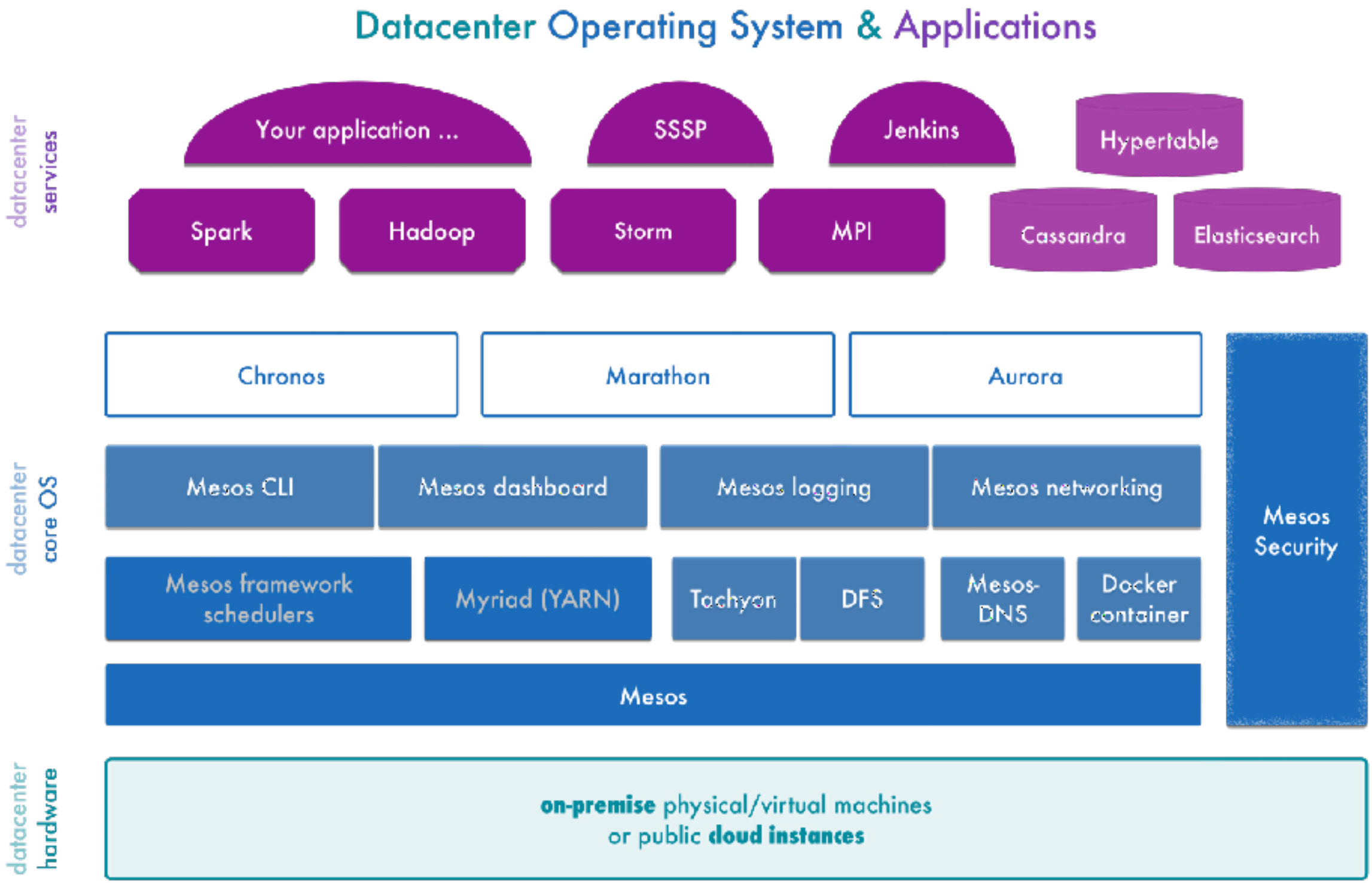
DC/OS

DCOS

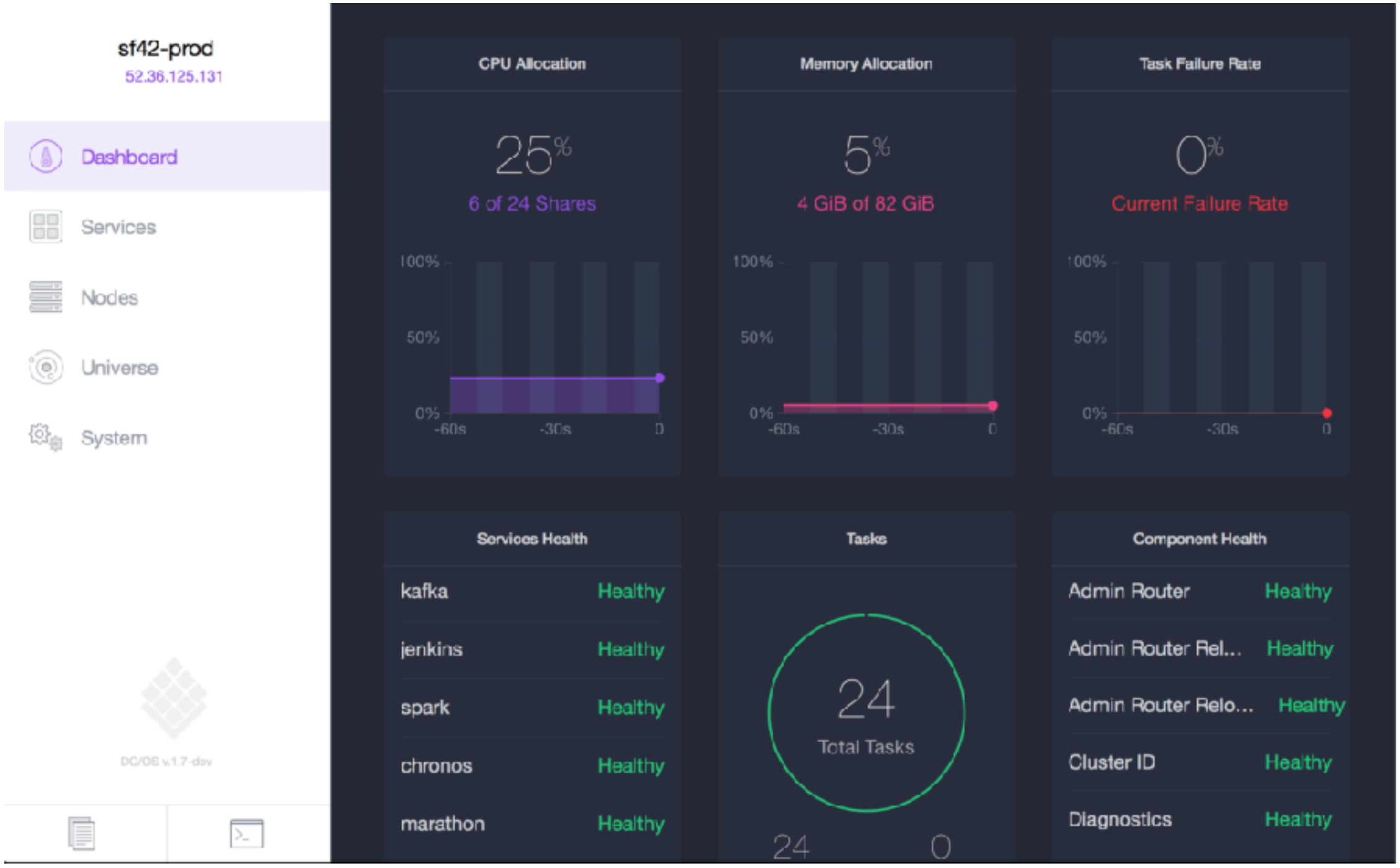
DCOS is

- **a distributed system**
  master & slave
  quorum & election
- **a cluster manager**
  scheduler coordinates resource and tasks on master
  executor implements tasks on slave
- **a container platform**
  schduler Marathon/Metronome + Container Docker & Mesos
  application specific logic
  stateful service
  container repository
- **a operating system**
  abstract hardware & software
  package management
  networking
  logging
  metrics
  storage and volumes
  identity management

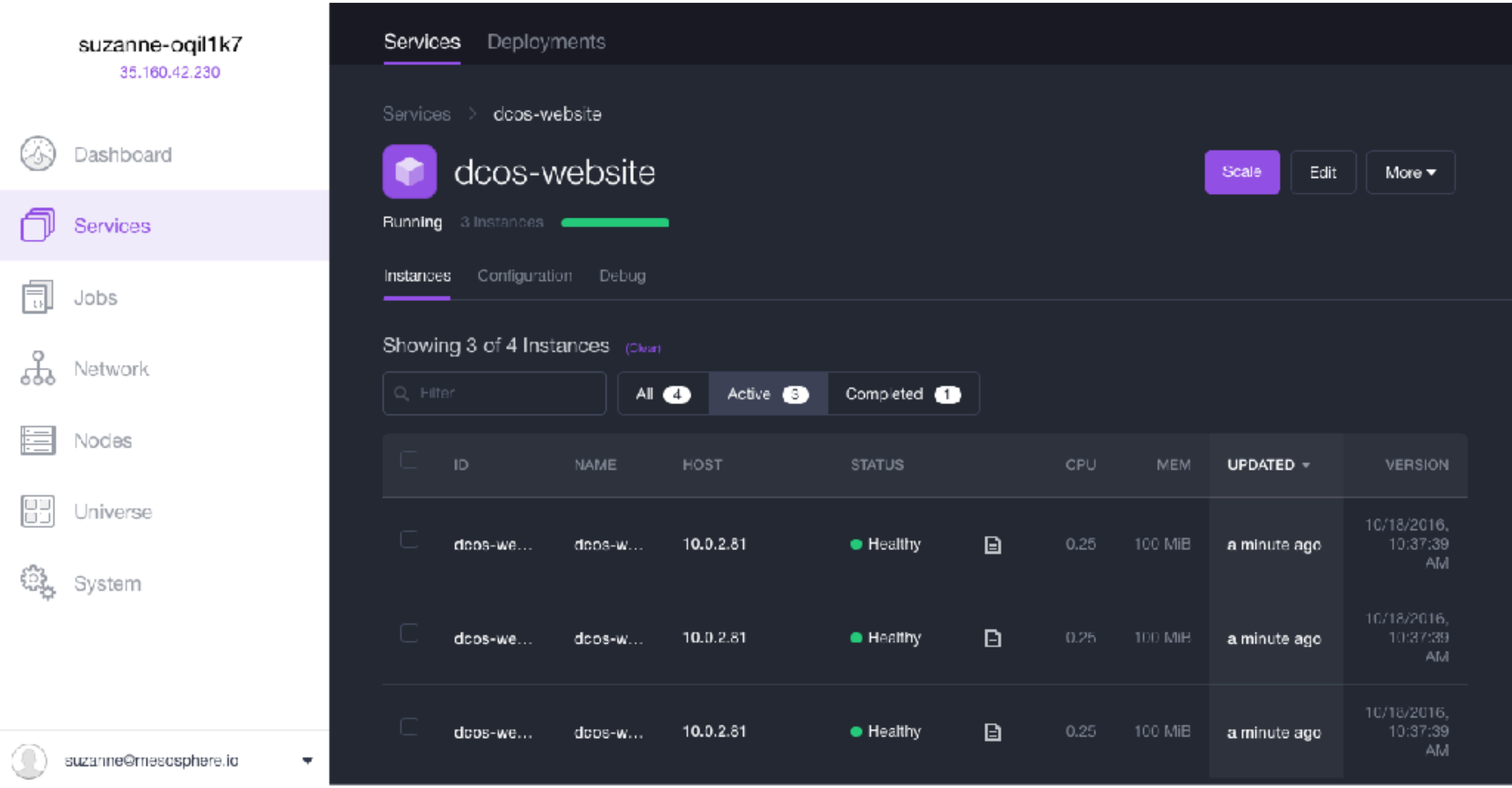
DCOS


Datacenter Operating System & Applications
datacenter services
Your application ...
SSSP
Jenkins
Hypertable
Spark
Hadoop
Storm
MPI
Cassandra
Elasticsearch
Chronos
Marathon
Aurora
datacenter core OS
Mesos CLI
Mesos dashboard
Mesos logging
Mesos networking
Mesos Security
Mesos framework schedulers
Myriad (YARN)
Tachyon
DFS
Mesos-DNS
Docker container
Mesos
datacenter hardware
on-premise physical/virtual machines
or public cloud instances

DCOS


sf42-prod
52.36.125.131
Dashboard
Services
Nodes
Universe
System
DC/OS v.1.7-dev
CPU Allocation
25%
6 of 24 Shares
100%
50%
0%
-60s
-30s
0
Memory Allocation
5%
4 GiB of 82 GiB
Task Failure Rate
0%
Current Failure Rate
Services Health
kafka Healthy
jenkins Healthy
spark Healthy
chronos Healthy
marathon Healthy
Tasks
24
Total Tasks
24
0
Component Health
Admin Router Healthy
Admin Router Rel... Healthy
Admin Router Relo... Healthy
Cluster ID Healthy
Diagnostics Healthy

DCOS


suzanne-oqil1k7
35.160.42.230
Dashboard
Services
Jobs
Network
Nodes
Universe
System
suzanne@mesosphere.io
Services
Deployments
Services > dcos-website
dcos-website
Running
3 Instances
Scale
Edit
More
Instances
Configuration
Debug
Showing 3 of 4 Instances
(Clear)
Filter
All 4
Active 3
Completed 1
ID
NAME
HOST
STATUS
CPU
MEM
UPDATED
VERSION
dcos-we...
dcos-w...
10.0.2.81
Healthy
0.25
100 MiB
a minute ago
10/18/2016, 10:37:39 AM

DCOS

BACK

Run a Service

JSON EDITOR

REVIEW & RUN

Services

- nginx
- memcached

Networking

Volumes

Health Checks

Environment

Networking

Configure the networking for your service.

NETWORK TYPE

Virtual Network: dcos

Service Endpoints

DC/OS can automatically generate a Service Address to connect to each of your load balanced endpoints.

nginx

CONTAINER PORT: 8080

SERVICE ENDPOINT NAME: web

HOST PORT: $PORT0

ASSIGN AUTOMATICALLY

PROTOCOL: UDP TCP

LOAD BALANCED SERVICE ADDRESS

Load balances the service internally (layer 4), and creates a service address. For external (layer 7) load balancing, create an external load balancer and attach this service.

ENABLED app.marathon.l4lb.thisdcos.directory:8080

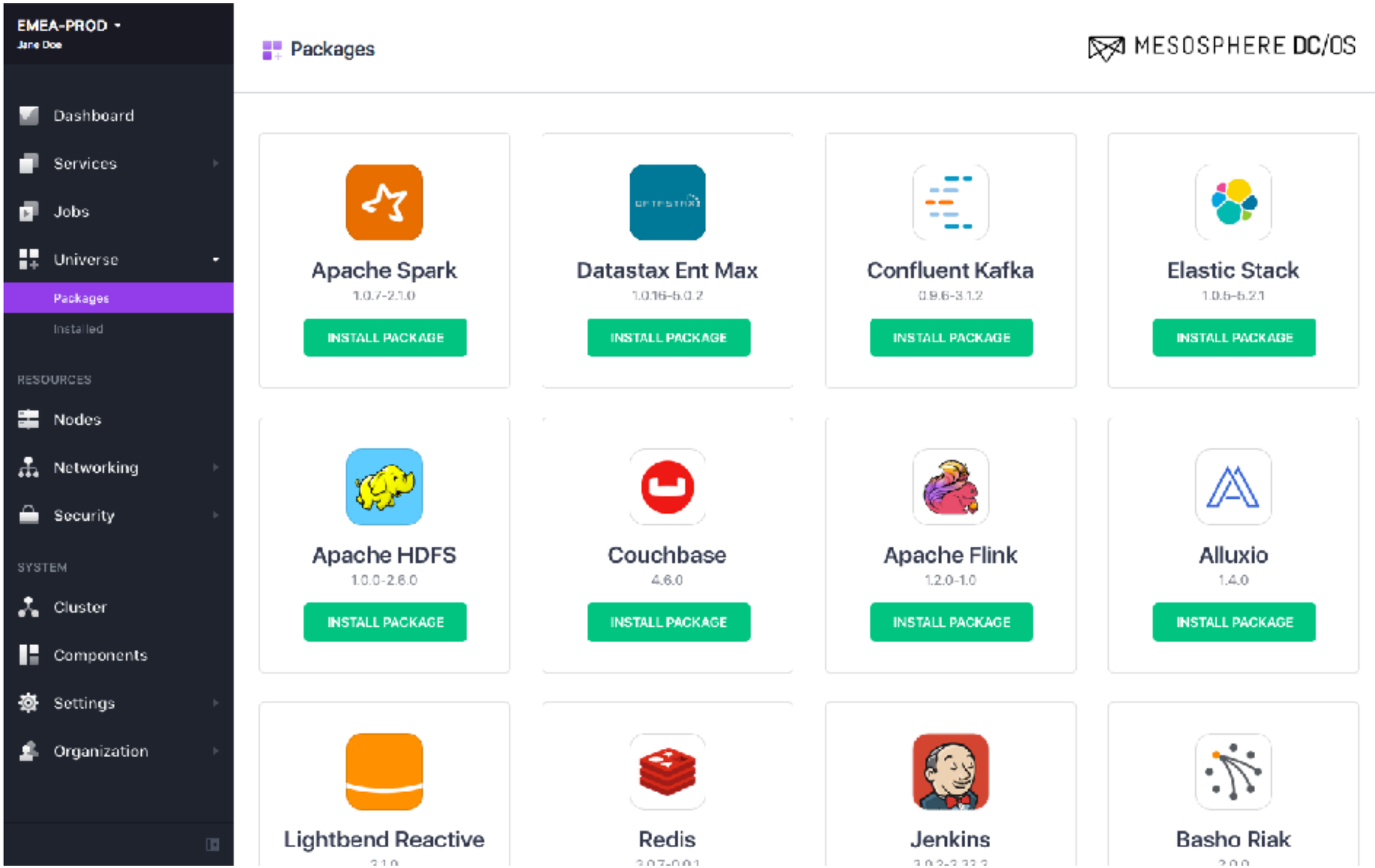
+ ADD SERVICE ENDPOINT

memcached

+ ADD SERVICE ENDPOINT

```
1  {
2    "id": "/app",
3    "containers": [
4      {
5        "name": "nginx",
6        "resources": {
7          "cpus": 0.1,
8          "mem": 128
9        },
10       "endpoints": [
11         {
12           "name": "web",
13           "containerPort": 8080,
14           "hostPort": 0,
15           "protocol": [
16             "tcp"
17           ],
18           "labels": {
19             "VIP_0": "/app:8080"
20           }
21         }
22       ],
23       "image": {
24         "id": "nginx",
25         "kind": "DOCKER"
26       }
27     },
28     {
29       "name": "memcached",
30       "resources": {
31         "cpus": 0.1,
32         "mem": 128
33       }
34     }
35   ],
36   "scaling": {
37     "kind": "fixed",
38     "instances": 1
39   },
40   "networks": [
41     {
42       "name": "dcos",
43       "mode": "container"
44     }
45   ]
46 }
```

DCOS


EMEA-PROD
Jane Doe
Dashboard
Services
Jobs
Universe
Packages
Installed
RESOURCES
Nodes
Networking
Security
SYSTEM
Cluster
Components
Settings
Organization
Packages
MESOSPHERE DC/OS
Apache Spark
1.0.7-2.1.0
INSTALL PACKAGE
Datastax Ent Max
1.0.16-5.0.2
INSTALL PACKAGE
Confluent Kafka
0.9.6-3.1.2
INSTALL PACKAGE
Elastic Stack
1.0.5-5.2.1
INSTALL PACKAGE
Apache HDFS
1.0.0-2.6.0
INSTALL PACKAGE
Couchbase
4.6.0
INSTALL PACKAGE
Apache Flink
1.2.0-1.0
INSTALL PACKAGE
Alluxio
1.4.0
INSTALL PACKAGE
Lightbend Reactive
Redis
Jenkins
Basho Riak

DCOS Applications

Application of DCOS

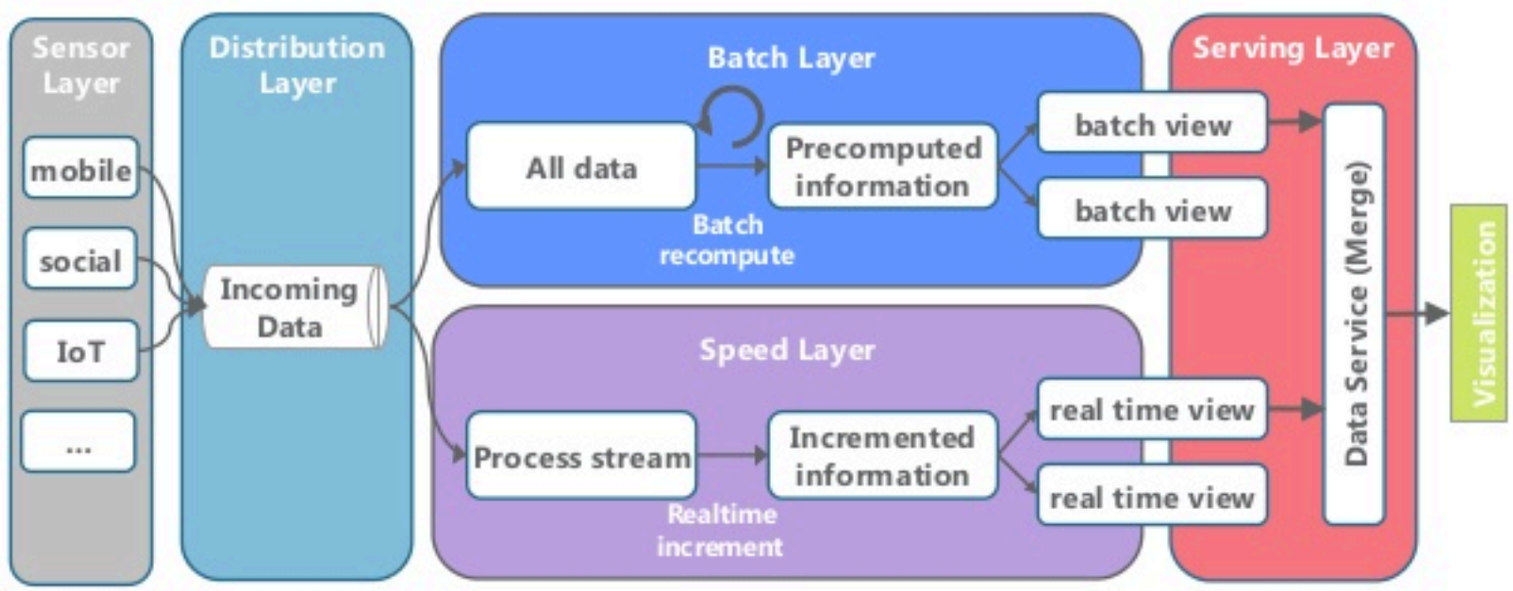
- microservices & containers
- big data & analytics
- data lake
- general infrastructure

DCOS Use Cases

- continuous integration & deployment
- lambda structure + SMACK stack

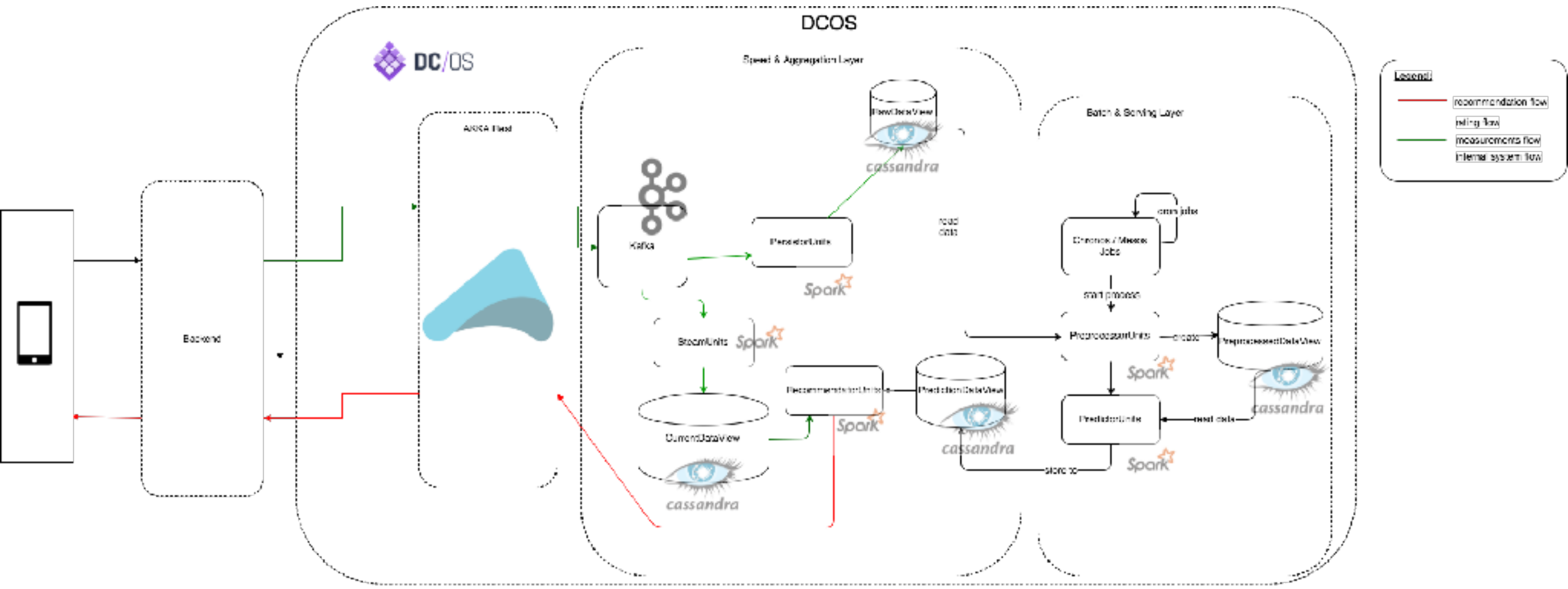
DCOS & SMACK

- Lambda Architecture

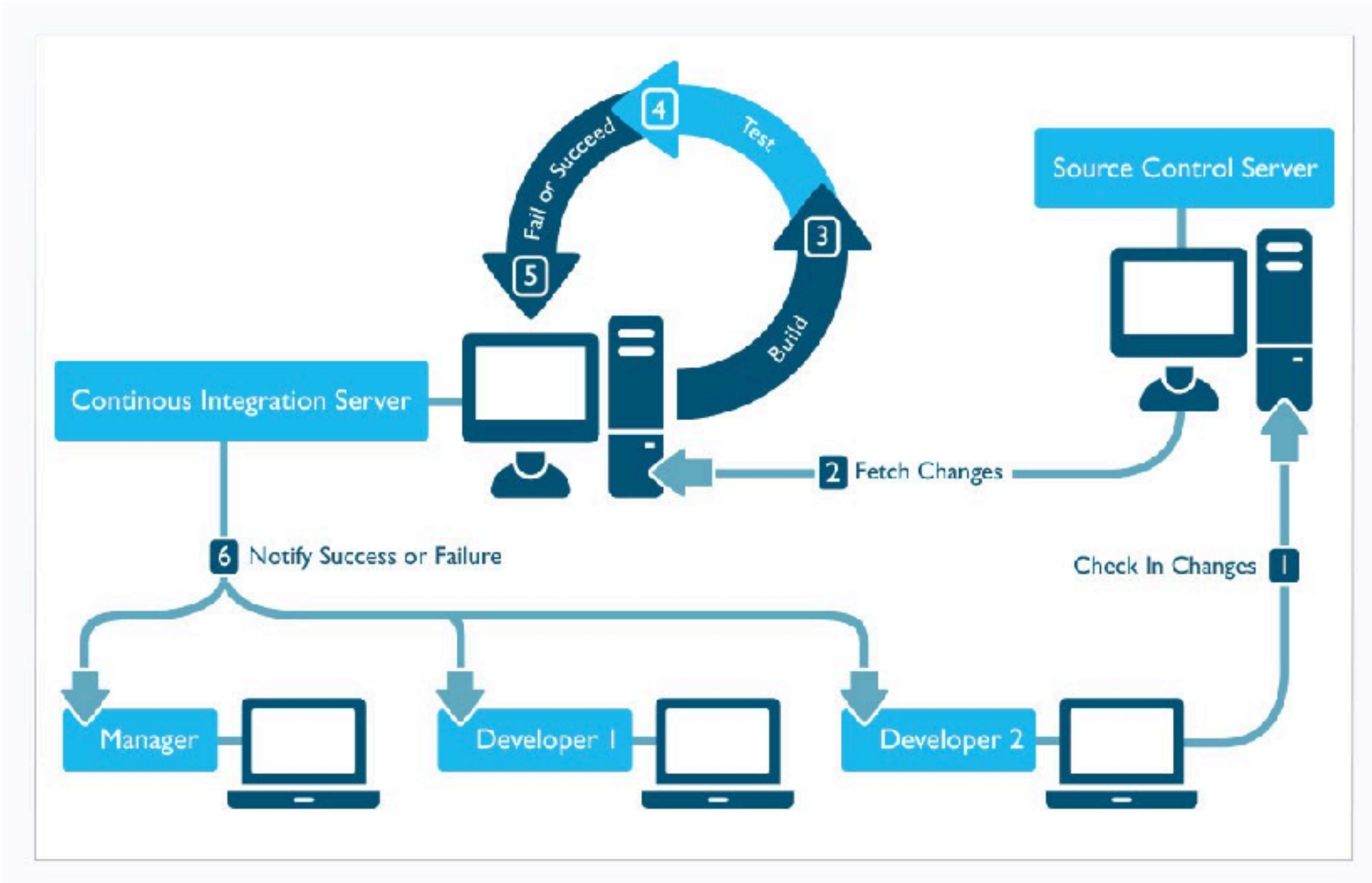


Adapted from: Marz, N. & Warren, J. (2013) Big Data. Manning.

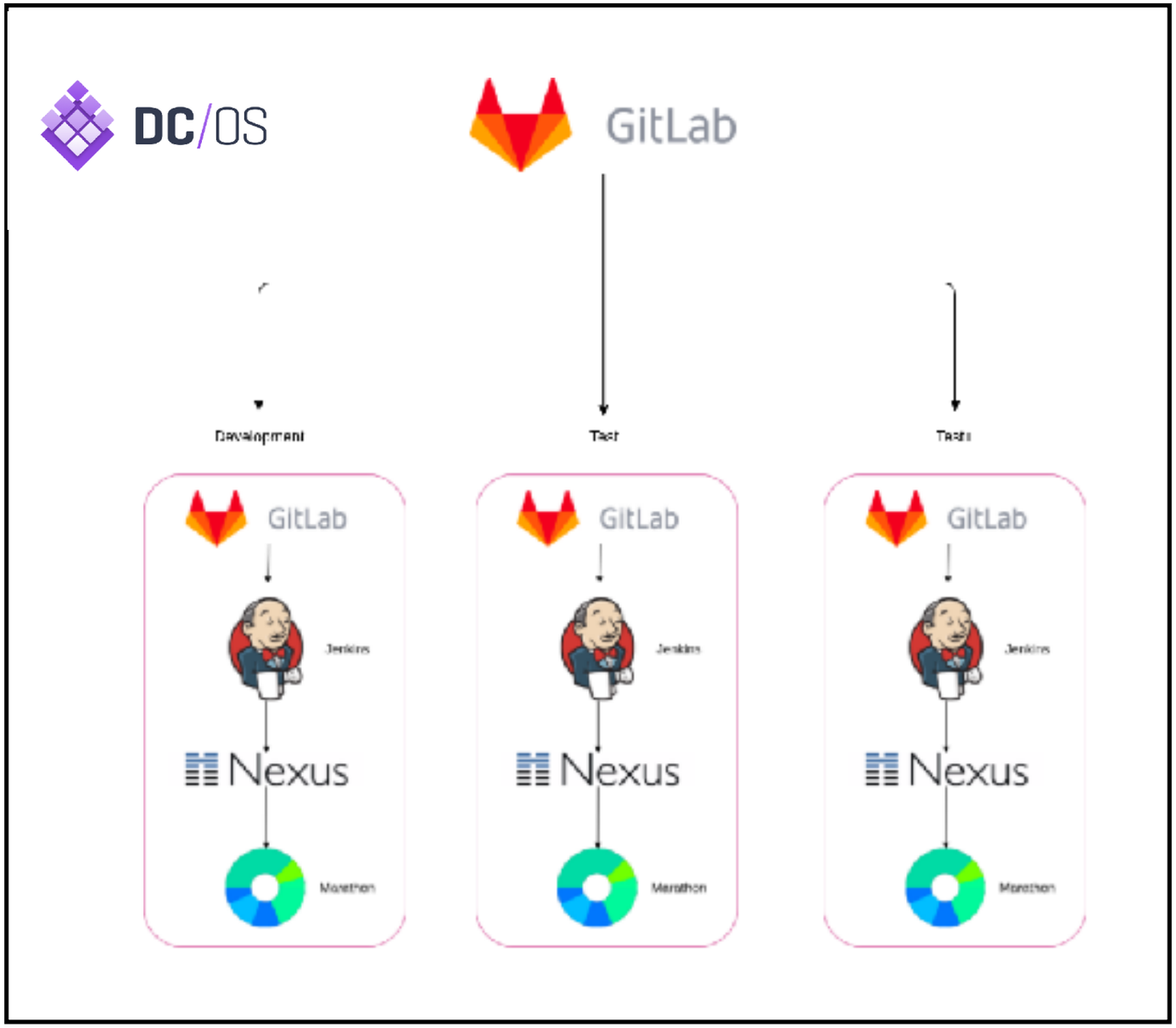
DCOS & SMACK


DCOS
DC/OS
Speed & Aggregation Layer
Batch & Serving Layer
Legend:
recommendation flow
measurements flow
Backend
Kafka
cassandra
Spark
cassandra
Spark
Spark
cassandra
Spark
cassandra
Spark
cassandra
Spark

DCOS CI

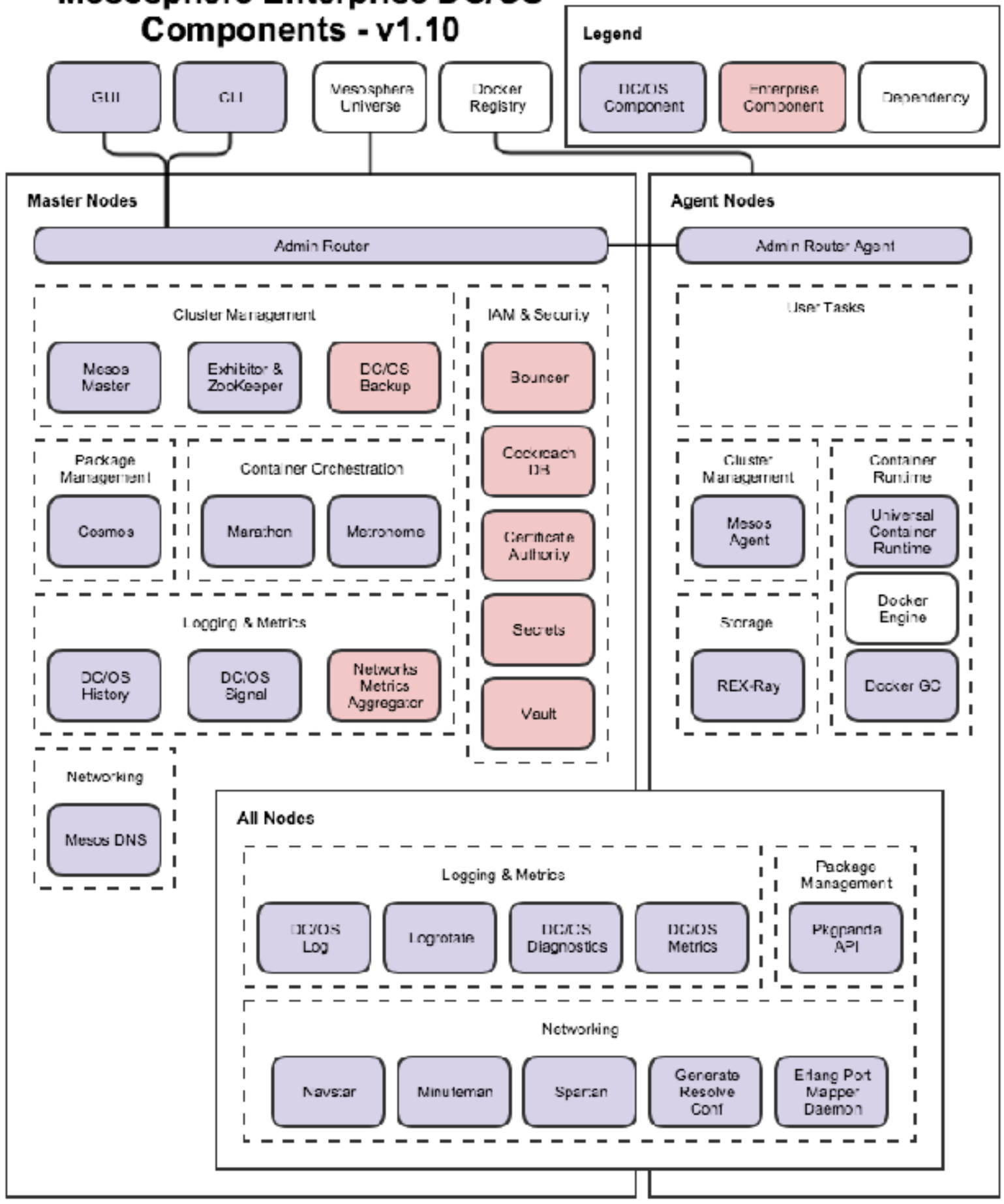

Source Control Server
Continous Integration Server
Test
Build
Fail or Succeed
1 Check In Changes
2 Fetch Changes
3
4
5
6 Notify Success or Failure
Manager
Developer 1
Developer 2

DCOS CI


DC/OS
GitLab
Development
Test
Tests
GitLab
Jenkins
Nexus
Marathon
GitLab
Jenkins
Nexus
Marathon
GitLab
Jenkins
Nexus
Marathon

Questions ?

Mesosphere Enterprise DC/OS Components - v1.10

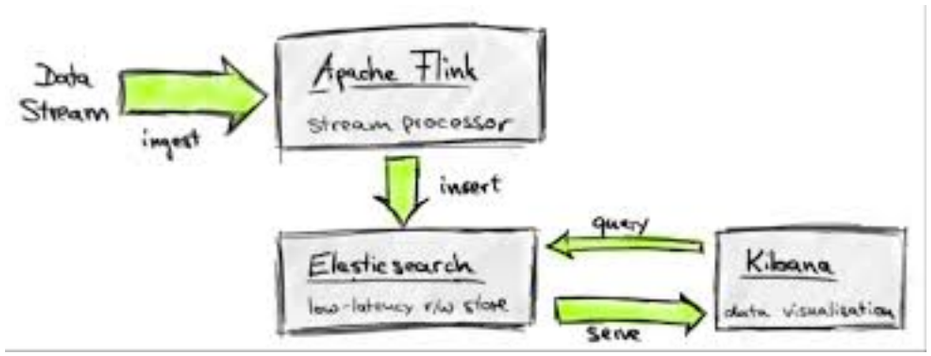


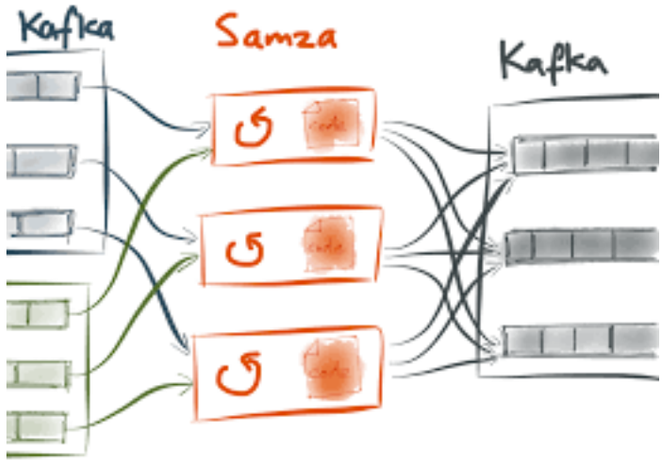
A slide about Mesos and YARN and resource fairness algorithm. Generally it tries to ensure every framework get a fair amount of its main resource.

https://www.slideshare.net/payberah/mesos-43904525

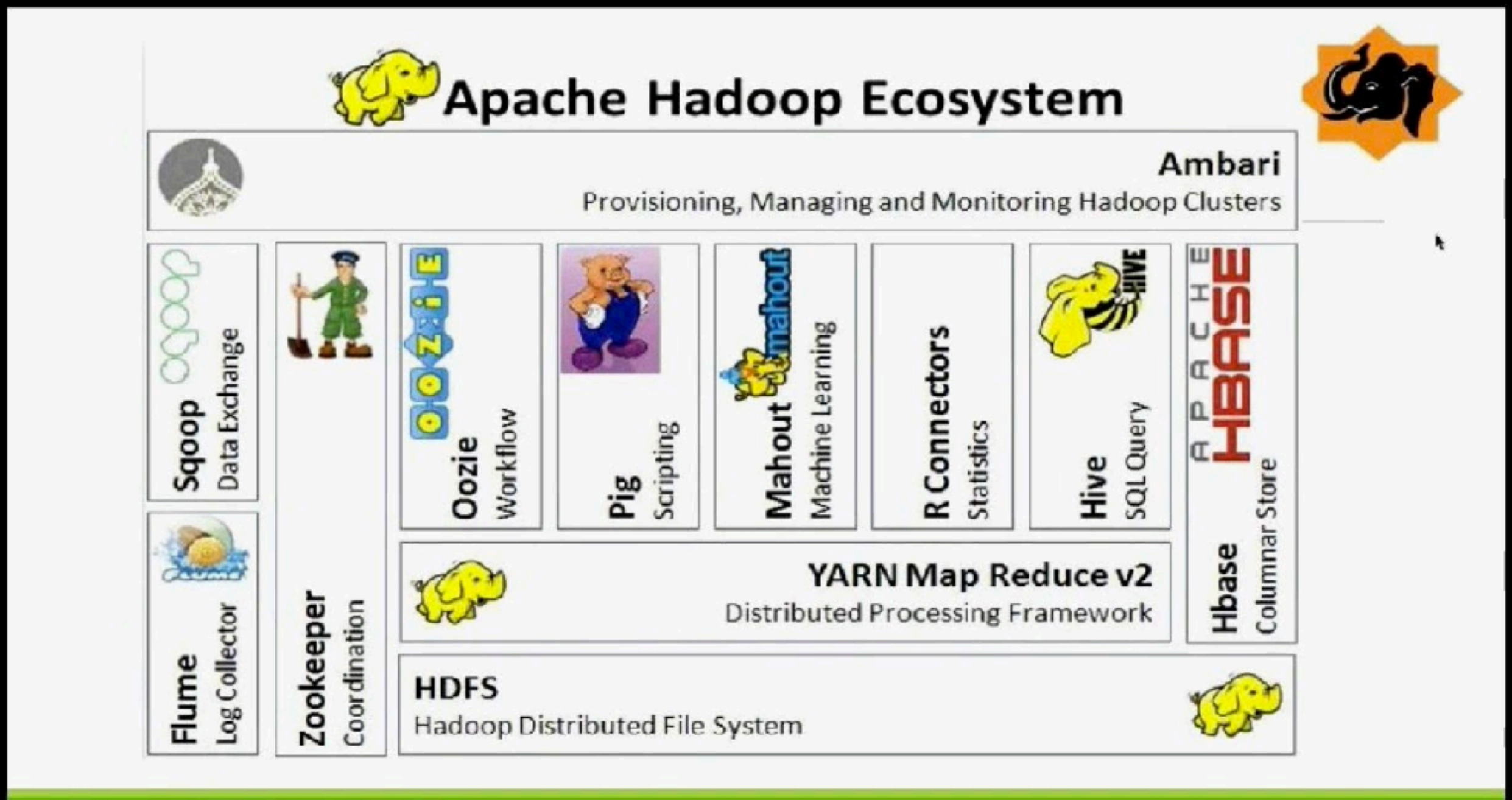
Some words about each framework

- Hadoop -> Batch mode, low abstraction, HDFS and MapReduce
- Spark -> Peudo-stream with micro batch, faster through In-memory process, more actions, fluent API, caching to accelerate iteration.
- Flink -> Real-time streaming, support batch as well, high-performance, support event, exactly-once insurance, snapshots
- Samza -> Kafka based streaming framework.
- Storm -> light-weight and really fast, no batch support.

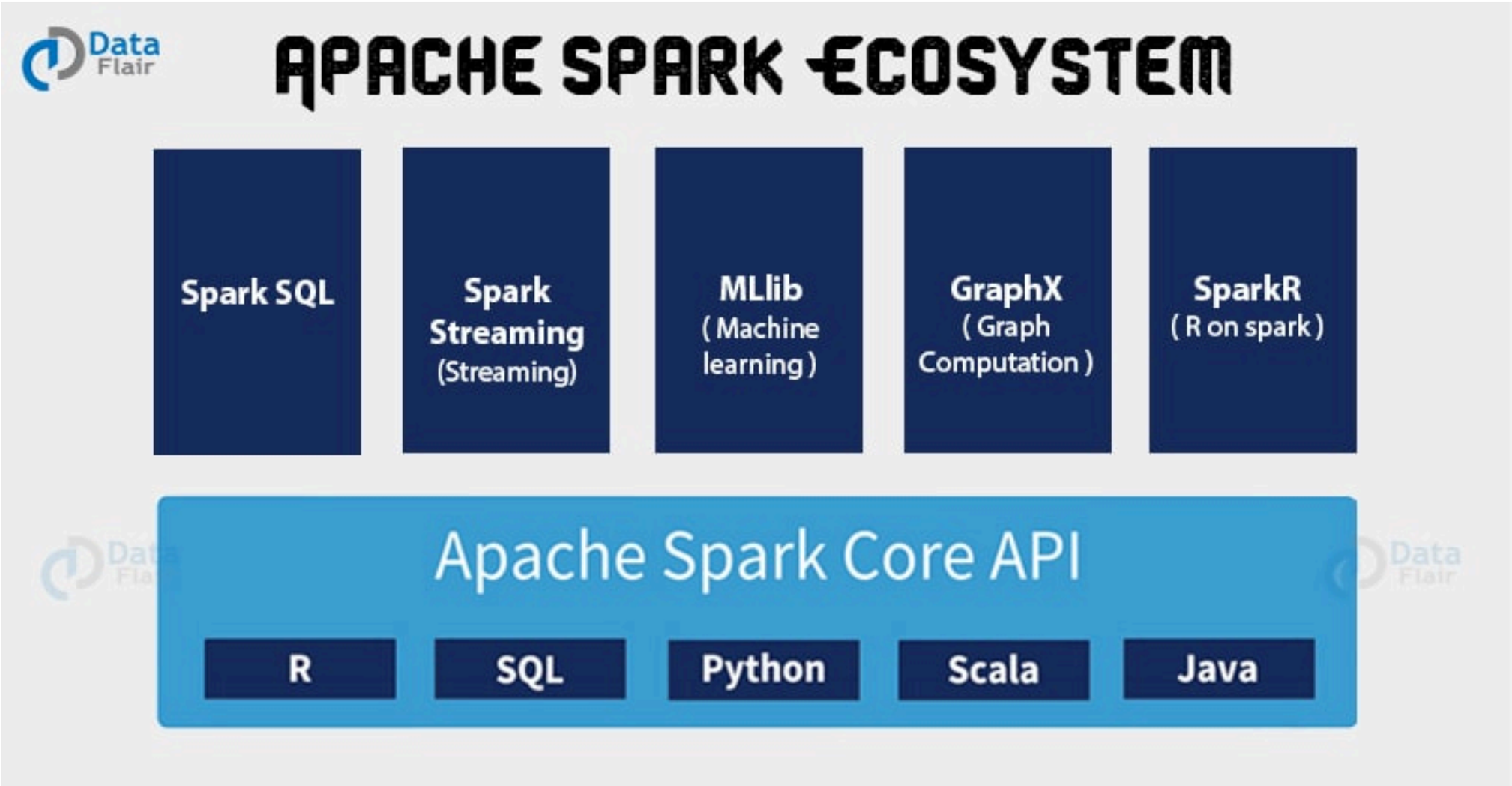




Hadoop


Apache Hadoop Ecosystem
Ambari
Provisioning, Managing and Monitoring Hadoop Clusters
Sqoop
Data Exchange
Flume
Log Collector
Zookeeper
Coordination
Oozie
Workflow
Pig
Scripting
Mahout
Machine Learning
R Connectors
Statistics
Hive
SQL Query
Hbase
Columnar Store
YARN Map Reduce v2
Distributed Processing Framework
HDFS
Hadoop Distributed File System
www.BigDataTrunk.com
4

Spark


Data Flair
APACHE SPARK ECOSYSTEM
Spark SQL
Spark Streaming (Streaming)
MLlib (Machine learning)
GraphX (Graph Computation)
SparkR (R on spark)
Apache Spark Core API
R
SQL
Python
Scala
Java

Big Data

Open Source Ecosystem