Introduction to DC/OS

Elizabeth K. Joseph
@pleia2
Elizabeth K. Joseph, Developer Advocate

- Developer Advocate at Mesosphere
- 15+ years working in open source communities
- 10+ years in Linux systems administration and engineering roles
- Founder of OpenSourceInfra.org
- Author of The Official Ubuntu Book and Common OpenStack Deployments
What are Cloud-Native Systems?

You no longer have a single server with everything running on it.

You have a multi-tier system with various layers and owners down the stack:

- Hardware
- Network
- Resource abstraction
- Scheduler
- Containers
- Virtual network
- Application
- ...

© 2017 Mesosphere, Inc. All Rights Reserved.
Cloud-native scopes

- Application
- Container
- Host
Cloud-Native with Containers
Apache Mesos: The datacenter kernel

http://mesos.apache.org/
Building block of the modern internet

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

http://mesos.apache.org/documentation/latest/powered-by-mesos/
THE BIRTH OF MESOS

CS262B
Ben Hindman, Andy Konwinski and Matei Zaharia create “Nexus” as their CS262B class project.

Twitter Tech Talk
The grad students working on Mesos give a tech talk at Twitter.

Apache Incubation
Mesos enters the Apache Incubator.

Mesos Published
Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center is published as a technical report.

DC/OS

Spring 2009
March 2010
September 2010
December 2010
April 2016
MULTIPLEXING OF DATA, SERVICES, USERS, ENVIRONMENTS

Typical Datacenter
siloed, over-provisioned servers,
low utilization

Apache Mesos
automated schedulers, workload multiplexing onto the
same machines
Active Tasks

<table>
<thead>
<tr>
<th>Framework ID</th>
<th>Task ID</th>
<th>Task Name</th>
<th>Role</th>
<th>State</th>
<th>Started</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>62df48e-df44-4309-9f40-73d5e94ab01e0001</td>
<td>bus-demo_dashboard.37943816-8677-11e7-b432-425fcb45b8</td>
<td>dashboard.busdemo</td>
<td>slave_public</td>
<td>RUNNING</td>
<td>a minute ago</td>
<td>10.0.5.101</td>
</tr>
<tr>
<td>62df48e-df44-4309-9f40-73d5e94ab01e0001</td>
<td>bus-demo_ingest.099da65-8676-11e7-b432-425fcb45b8</td>
<td>ingest.busdemo</td>
<td>slave_public</td>
<td>RUNNING</td>
<td>9 minutes ago</td>
<td>10.0.1.204</td>
</tr>
<tr>
<td>62df48e-df44-4309-9f40-73d5e94ab01e0004</td>
<td>broker-2__581647a0-6953-4cfe-af96-356d04535c38</td>
<td>broker-2</td>
<td>kafka-role</td>
<td>RUNNING</td>
<td>12 minutes ago</td>
<td>10.0.3.240</td>
</tr>
<tr>
<td>62df48e-df44-4309-9f40-73d5e94ab01e0004</td>
<td>broker-1__24b1885-8600-4ae9-8f6b-502fcded5fe</td>
<td>broker-1</td>
<td>kafka-role</td>
<td>RUNNING</td>
<td>13 minutes ago</td>
<td>10.0.3.7</td>
</tr>
<tr>
<td>62df48e-df44-4309-9f40-73d5e94ab01e0004</td>
<td>broker-0__eb077c0-f416-4918-9c8d-1f5b1ea8c10d</td>
<td>broker-0</td>
<td>kafka-role</td>
<td>RUNNING</td>
<td>13 minutes ago</td>
<td>10.0.1.204</td>
</tr>
<tr>
<td>62df48e-df44-4309-9f40-73d5e94ab01e0001</td>
<td>kafka.8a668774-8675-11e7-b432-425fcb45b8</td>
<td>kafka</td>
<td>slave_public</td>
<td>RUNNING</td>
<td>13 minutes ago</td>
<td>10.0.0.68</td>
</tr>
<tr>
<td>62df48e-df44-4309-9f40-73d5e94ab01e0003</td>
<td>node-2__a9c29921-d7c1-4a32-8eb5-4f373b25665d</td>
<td>node-2</td>
<td>cassandra-role</td>
<td>RUNNING</td>
<td>14 minutes ago</td>
<td>10.0.3.7</td>
</tr>
</tbody>
</table>
Mesos can’t run applications on its own.

A Mesos framework is a distributed system that has a scheduler.

Schedulers like Marathon start and keep your applications running. A bit like a distributed init system.

Learn more at https://mesosphere.github.io/marathon/
<table>
<thead>
<tr>
<th>Name</th>
<th>CPU</th>
<th>Memory</th>
<th>Status</th>
<th>Running Instances</th>
<th>Health</th>
</tr>
</thead>
<tbody>
<tr>
<td>bus-demo</td>
<td>0.2</td>
<td>4 GiB</td>
<td></td>
<td>2 of 2</td>
<td></td>
</tr>
<tr>
<td>cassandra</td>
<td>0.5</td>
<td>2 GiB</td>
<td>Running</td>
<td>1 of 1</td>
<td></td>
</tr>
<tr>
<td>kaika</td>
<td>1.0</td>
<td>1 GiB</td>
<td>Running</td>
<td>1 of 1</td>
<td></td>
</tr>
<tr>
<td>spark</td>
<td>1.0</td>
<td>1 GiB</td>
<td>Running</td>
<td>1 of 1</td>
<td></td>
</tr>
</tbody>
</table>
Containers

- Rapid deployment
- Some service isolation
- Dependency handling
- Container image repositories
CONTAINER ORCHESTRATION

CONTAINER SCHEDULING

RESOURCE MANAGEMENT

SERVICE MANAGEMENT
DC/OS brings it all together

- Resource management
- Task scheduling
- Container orchestration
- Logging and metrics
- Network management
- “Universe” catalog of pre-configured apps (including Apache Spark, Apache Kafka…), browse at http://universe.dcos.io/
- And much more https://dcos.io/
DC/OS is ...

- 100% open source (ASL2.0)
- An umbrella for ~30 OSS projects
  + Roadmap and designs
  + Documentation and tutorials
- Not limited in any way
- Familiar, with more features
  + Networking, Security, CLI, UI, Service Discovery, Load Balancing, Packages, ...
DC/OS
Architecture Overview

Services & Containers

- HDFS
- Jenkins
- Marathon
- Cassandra
- Flink
- Spark
- Docker
- Kafka
- MongoDB
- +30 more...

DC/OS

- Container Orchestration
- Security & Governance
- Monitoring & Operations
- User Interface & Command Line

ANY INFRASTRUCTURE

- Physical Servers
- Virtual Servers
- Private Cloud
- Public Cloud Providers (Google, AWS, Azure)
Interact with DC/OS (1/2)

Web-based GUI

https://dcos.io/docs/latest/usage/webinterface/
Interact with DC/OS (2/2)

**CLI tool**

https://dcos.io/docs/latest/usage/cli/

**API**

https://dcos.io/docs/latest/api/
Catalog of Applications (Universe)
Install an Application
"service": {
  "name": "gitlab",
  "cpus": 1,
  "mem": 2048,
  "role": "",
  "host-volume": "/srv/gitlab",
  "host-shared-volume": "/srv/gitlab-data"
},
"routing": {
  "https-redirect": false,
  "ssh-port": 22222,
  "registry-port": 50000
},
"email": {
  "enabled": false,
  "port": 25,
  "authentication": "login",
  "enable-starttls-auto": true,
  "openssl-verify-mode": "peer",
  "tls": false
},
"high-availability": {
  "enabled": false,
  "postgres": {},
  "redis": {};
},
"single-node": {
  "local-volumes": {},
  "external-volumes": {
    "enabled": false
  }
},
"enterprise": {
  "enterprise-edition": false
}
Building a Real-World Pipeline
MODERN APPLICATION -> FAST DATA BUILT-IN

Use Cases:
- Anomaly detection
- Personalization
- IoT Applications
- Predictive Analytics
- Machine Learning
The SMACK Stack

Data Ingestion

Request/Response

Microservices

Distributed Storage

Message Queue/Bus

Analytics (Streaming)

Use Cases:
- Anomaly detection
- Personalization
- IoT Applications
- Predictive Analytics
- Machine Learning

Devices

Sensors

Client
Keeping things running: Day 2 Operations

Metrics & Monitoring
- Collecting metrics
- Routing events
- Downstream processing
  - Alerting
  - Dashboards
  - Storage (long-term retention)

Logging
- Scopes
- Local vs. Central
- Security considerations
Day 2 Operations con’t

Maintenance
- Cluster Upgrades
- Cluster Resizing
- Capacity Planning
- User & Package Management
- Networking Policies
- Auditing
- Backups & Disaster Recovery

Troubleshooting
- Debugging
  - Services
  - System
  - Access
- Tracing
- Chaos Engineering
Demo

https://dcos.io/demos/
Questions?
Feedback?

Elizabeth K. Joseph
Twitter: @pleia2
Email: lyz@princessleia.com