Containerized, Cloud-Native Operations for Big Data Analytics

SoCal DevOps
July 19, 2017
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
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
- Container
- Virtual network
- Application
- ...

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

- Application
- Container
- Host
Cloud-native with DC/OS

Universal Container Runtime (UCR), Docker

DC/OS, Apache Mesos
MODERN APPLICATION -> FAST DATA BUILT-IN

Use Cases:
- Anomaly detection
- Personalization
- IoT Applications
- Predictive Analytics
- Machine Learning
OK, got it!
Now integrate it with the rest of your technology stack
Unification of tooling

- Integrates into your existing, familiar infrastructure
- Reduces resource consumption (avoids multiple monitoring, logging agents, etc)
- Simplifies troubleshooting (tracing a problem through the stack)
- Consolidates view for all parties (from operations to app developers)
Anyone can write a deployment tool.

What’s next?
Metrics and Monitoring

- Collecting metrics
- Downstream processing
  - Alerting
  - Dashboards
  - Storage (long-term retention)

Logging

- Scopes
- Local vs. centralized
- Security considerations
DAY 2 OPERATIONS

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

Troubleshooting
- Debugging
  - Services
  - System
- Tracing
- Chaos engineering
METRICS & MONITORING
METRICS CONCEPTS

node

- service
- container
- host
- collectd

event router

- storage
- dashboard
- alerting
METRICS TOOLCHAIN

- local scraping:
  - a. collectd
  - b. cAdvisor

- event router:
  - a. fluentd
  - b. Flume
  - c. Kafka
  - d. logstash
  - e. Riemann
● storage:
   a. Elasticsearch
   b. Graphite
   c. InfluxDB
   d. KairosDB/Cassandra
   e. OpenTSDB/HBase
   f. others such a local filesystem, Ceph FS, HDFS, etc.
METRICS TOOLCHAIN

- dashboard:
  - a. D3
  - b. Grafana
  - c. signal fx

- alerting:
  - a. BigPanda
  - b. PagerDuty
  - c. signal fx
  - d. VictorOps
INTEGRATED METRICS TOOLCHAIN

- Amazon CloudWatch
- AppDynamics
- Azure Monitor
- Circonus
- DataDog
- dcos/metrics
- Ganglia
- Google Stackdriver
- Hawkular
- Icinga
- Librato
- Nagios
- New Relic
- OpsGenie
- Pingdom
- Prometheus
- Ruxit Dynatrace
- Sensu
- Sysdig
- Zabbix
LOGGING
LOGGING
SCOPES

Application

Container

Host
LOGGING TOOLING EXAMPLES (PRIMITIVES)

- DC/OS logging overview
- Docker logging drivers
- systemd's journalctl
Centralized app logging with fluentd

DC/OS

ELK stack log shipping

Splunk

Graylog

Loggly

Papertrail

Sumo Logic
TROUBLESHOOTING

Incl. examples with DC/OS
Effective troubleshooting

A high level view to discover where the error or failure has occurred (preferably a unified view)

Tooling for tracing an error through the stack (systems, networks, etc)

Team communication and tooling for delegating solutions responsibility
• **Services**: typically specific to service, use logging (for example, `dcos task log`) and `dcos node ssh` or `dcos task exec` for per-node investigations

• **System**:
  ○ Simple [diagnostics](#) via `dcos node diagnostics`
  ○ Comprehensive dump via [clump](#)
  ○ Services deployment troubleshooting dashboard
Recent Resource Offers (2)

When you attempt to deploy a service, DC/OS waits for offers to match the resources your service requires. If the offer does not satisfy the requirement, it is declined and DC/OS retries. Learn more.

**Summary**

![Graph showing resource offers](image)

- Role: 50% matched, 50% declined
- Constraints: 100% matched
- CPU: 100% matched
- Memory: 0% matched
- Disk: 0% matched
- Ports: 0% matched

**Matched** green, **Declined** gray

**Details**

<table>
<thead>
<tr>
<th>HOST</th>
<th>RLE</th>
<th>CSTR</th>
<th>CPU/MEM/DSK</th>
<th>PRT</th>
<th>RECEIVED</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.193</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>10.0.4.126</td>
<td>✗</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
</tr>
</tbody>
</table>
● Tracing
  ○ Idea: identify latency issues and perform root-cause analysis in a distributed setup
  ○ OpenTracing

● Chaos Engineering
  ○ Idea: proactively break (parts of) the system to understand how it reacts
  ○ Chaos Monkey
  ○ DRAX
MAINTENANCE & BEYOND
Overview

- How to install a new version of X?
- When to scale what (service-level vs. nodes)
- Who gets to access/install which services in what way?

Upgrades

Sizing

User and package management

- Is everything getting where it needs to be? Does some traffic need priority?
- What services can talk to each other and in which way?
- Who accessed what, when and how?
- How is the continuous operation of the cluster and the services accomplished?
  What happens when cluster (or critical infra components like ZK) go down?

Networking

Auditing

Disaster Recovery
Planning

Things will go wrong.

These things can’t be an afterthought.

You must build time into your deployment and maintenance plans.
Cloud-Native Infrastructure “Must Haves”

- Metrics collection
- Centralized logging
- Debugging tools that cover:
  - Host
  - Container
  - Application
- Upgrade strategy
- Backups
- Disaster recovery
Questions? Feedback?

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

@dcos
chat.dcos.io
users@dcos.io
/dcos
/dcos/examples
/dcos/demos
Let’s process some data!

https://github.com/dcos/demos/tree/master/fastdata-iot/
The SMACK Stack

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