Day 2
Operations of Cloud-Native Systems

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

- 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*
Day 2 Operations

Anyone can write a deployment tool.

What’s next?
Cloud-Native Systems

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

It’s now a multi-tier system with various owners down the stack:

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

© 2017 Mesosphere, Inc. All Rights Reserved.
This gets out of hand very quickly

Unification of operations and tracking becomes important

- Reduces resource consumption (multiple monitoring & logging agents, etc)
- Simplifies troubleshooting (tracing a problem through the stack)
- Consolidates view for all parties (from operations to app developers)
DAY 2 OPERATIONS

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
• local scraping:
  a. collectd
  b. cAdvisor

• event router:
  a. fluentd
  b. Flume
  c. Kafka
  d. logstash
  e. Riemann
METRICS TOOLCHAIN

● storage:
  a. Elasticsearch
  b. Graphite
  c. InfluxDB
  d. KairosDB/Cassandra
  e. OpenTSDB/HBase
  f. others such as 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

SCOPES

- service (app/business)
- container
- host & intra-host
LOGGING TOOLING EXAMPLES (PRIMITIVES)

- DC/OS logging overview
- Docker logging drivers
- systemd's journalctl
LOGGING TOOLING EXAMPLES (INTEGRATED)

- Centralized app logging with fluentd
- DC/OS
  a. ELK stack log shipping
  b. 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

<table>
<thead>
<tr>
<th>Role</th>
<th>Constraints</th>
<th>CPU</th>
<th>Memory</th>
<th>Disk</th>
<th>Ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>50%</td>
<td>100%</td>
<td>100%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

Matched ▶️ Declined

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

Upgrades
Sizing
User and package management

- 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?

Networking
Auditing
Disaster Recovery

- 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?
These things can’t be an afterthought when something goes wrong.

Build time into deployment and maintenance plan.
Checklist

Cloud-Native Infrastructure “Must Haves”

- Metrics collection
- Centralized logging
- Debugging tools that cover:
  - Host
  - Container
  - Application
- Upgrade strategy
- Backups
- Disaster recovery
To conclude

Properly managing cloud-native systems is complicated!

- Ask the right questions
- Unify and simplify as much as you can
- Have a checklist of considerations
- Plan in time to complete everything