Kubernetes + Google

5 lessons learned from 8+ years of containers

Victor Marmol (vmarmol@google.com)
Tim Hockin (thockin@google.com)
Kubernetes

κυβερνήτης: Greek for “pilot” or “helmsman of a ship”
the open source cluster workload manager from Google

- Manage applications, not machines
- Inspired by Google’s internal systems
  - >10 years experience running containerized apps
  - >2B containers / week
- Extensible & portable
- Could run anywhere
- Apache 2.0 licensed
- Written in Go
Cluster

Kubernetes Master/Scheduler

Kubernetes Agent

Machine Host
Lesson 1: Pods

Kubernetes Agent
Machine Host

Kubernetes Agent
Machine Host

Kubernetes Agent
Machine Host

Kubernetes Agent
Machine Host

Kubernetes Agent
Machine Host

Kubernetes Agent
Machine Host

Kubernetes Agent
Machine Host

Kubernetes Agent
Machine Host

Kubernetes Master/Scheduler
Lesson 1: Pods

- Easy (and thorough) usage accounting and limiting
- Enables resource sharing
Lesson 2: Labels & Services

Kubernetes - Master/Scheduler

- Kubernetes Agent
- Machine Host
- Kubernetes Agent
- Machine Host
- Kubernetes Agent
- Machine Host
- Kubernetes Agent
- Machine Host
- Kubernetes Agent
- Machine Host
- Kubernetes Agent
- Machine Host

labels:
  type: frontend
Lesson 2: Labels & Services

Web-Service

id: web-service
port: 9000
selector:
  tier: frontend
  env: production

Kubernetes - Master/Scheduler
Lesson 3: IP per Pod and IP per Service

- Makes addressing any entity easy (no need for host, port location tuple)
- Users should never share a namespace they don't control
- Port brokering and scheduling is very difficult
Lesson 4: *Everything* is in a Container

- If something is not containerized, nothing is
- Everything has a limit
- Detailed accountability of resource usage
- We use subcontainers extensively for accounting

<table>
<thead>
<tr>
<th>Pod 1</th>
<th>Pod 2</th>
<th>Pod 3</th>
<th>System Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1 CPU, 1GB)</td>
<td>(1.5 CPUs, 1GB)</td>
<td>(1 CPU, 1.5GB)</td>
<td>(0.5 CPUs, 0.5GB)</td>
</tr>
</tbody>
</table>

**Machine** (4 CPUs, 4GB)
### Lesson 5: Resource Tiers & Priority

<table>
<thead>
<tr>
<th>Component</th>
<th>CPU</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frontend (1 CPU, 1GB)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frontend (1.5 CPUs, 1GB)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frontend (1 CPU, 1.5GB)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Services (0.5 CPUs, 0.5GB)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Batch (1 CPU, 1GB)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Machine (4 CPUs, 4GB)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Enables selling “tiers” of resources with varying guarantees
- Increases utilization
- Allows handling out of resource conditions (what pod do I kill?)
- Must handle out of resource conditions well
- Intent-based at higher layers
Questions?

Tim Hockin: thockin@google.com
Victor Marmol: vmarmol@google.com

Kubernetes
github.com/googlecloudplatform/kubernetes

Mailing List: google-containers@googlegroups.com
IRC: #google-containers on Freenode
Backup
Application stack on single host

CMS (drupal php-fpm)
webserver (nginx)
memcache
DB (mysql)

init  sshd  logger  monitor
OS
Application stack in single container

CMS (drupal php-fpm)
webserver (nginx)
memcache
DB (mysql)
supervisor
sshd

Container

init sshd logger monitor

OS
Application stack in separate containers

CMS (drupal php-fpm)

webserver (nginx)

memcache

DB (mysql)

init    sshd    logger    monitor

OS
Replicated application stack
Distributed application stack

- CMS (drupal)
- webserver (nginx)
- memcache
- DB (mysql)

init
sshd
OS
Liberated application stack

- CMS (drupal)
- memcache
- webserver (nginx)
- memcache
- DB (mysql)
- DB (mysql)

init

sshd

OS
Master components
Colocated, or spread across machines, as dictated by cluster size.