Integrating Containers in the CERN Private Cloud

Ricardo Rocha
(on behalf of the CERN Cloud team)
Why containers?

- *Isolation*, via kernel namespaces and cgroups
- *Performance*, same kernel
- *Improved resource utilization*
- *Ease of use*, microservices, deployment units, image repositories
Goals and Timeline

- Integrate containers in the CERN cloud
  - Shared identity, networking integration, storage access, ...
- Agnostic to container orchestration engines
  - Docker Swarm, Kubernetes, Mesos
- Fast, easy to use

CERN / HEP Service Integration, Networking, CVMFS, EOS

Container Investigations 11 / 2015
Magnum Tests 02 / 2016
Upstream Development

Pilot Service Deployed

Mesos Support 10 / 2016
Production Service
OpenStack Magnum

- OpenStack container project
- Orchestrate deployment of container clusters
- Key features
  - Swarm, Kubernetes, Mesos support
  - Client access using native clients / APIs
  - Cluster scaling
  - Lifecycle operations
Example Usage

- Clusters are described by *cluster templates*
- Shared/public templates for most common setups, customizable by users

```
$ magnum cluster-template-list
+------------------------+---------------------------+
| uuid | name                  |
|......|-----------------------|
| .... | swarm                 |
| .... | swarm-ha              |
| .... | kubernetes            |
| .... | kubernetes-ha         |
| .... | mesos                 |
| .... | mesos-ha              |
+------------------------+---------------------------+

$ magnum cluster-template-show swarm
...
| coe         | swarm          |
| master_flavor_id | m1.small      |
| flavor_id   | m1.small       |
| server_type | vm            |
| image_id    | fedora-23-atomic|
| labels      | {}            |
| network_driver | docker       |
```
Example Usage

- Create a cluster in a single command (no matter what size)

```bash
$ magnum cluster-create --name myswarmcluster --cluster-template swarm --node-count 100

$ magnum cluster-list
+-----------------+-----------------+-----------------+-----------------+
<table>
<thead>
<tr>
<th>uuid</th>
<th>name</th>
<th>node_count</th>
<th>master_count</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>myswarmcluster</td>
<td>100</td>
<td>1</td>
<td>CREATE_COMPLETE</td>
</tr>
<tr>
<td>------</td>
<td>----------------</td>
<td>------------</td>
<td>--------------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>

$ $(magnum cluster-config myswarmcluster --dir magnum/myswarmcluster)

$ docker info / ps / ...
$ docker run --volume-driver cvmfs -v atlas.cern.ch:/cvmfs/atlas -it centos /bin/bash
[root@32f4cf39128d /]#
```
Example Usage

- Scale your cluster (up)

  ```
  $ magnum cluster-update myswarmcluster replace node_count=200
  ```

- Scale your cluster (and down…)

  ```
  $ magnum cluster-update myswarmcluster replace node_count=5
  ```

- Support for built-in orchestration tools
  - Docker Compose, Kubernetes, Marathon/DCOS
Use Cases

- Example: Spark on Mesos

```bash
$ magnum cluster-create --name myspark --cluster-template mesos --node-count 20

$ magnum cluster-show myspark | grep api_address
   | api_address | 137.138.7.77 |

$ spark-shell --master mesos://zk://137.138.7.77:2181/mesos
scala> val NUM_SAMPLES = 1000
   val count = sc.parallelize(1 to NUM_SAMPLES).map{i =>
       val x = Math.random()
       val y = Math.random()
       if (x*x + y*y < 1) 1 else 0
   }.reduce(_ + _)
println("Pi is roughly " + 4.0 * count / NUM_SAMPLES)
Pi is roughly 3.142532
```
Use Cases

- Example: File Transfer Service

```bash
$ magnum cluster-create --name fts --cluster-template kubernetes --node-count 20
$ $(magnum cluster-config fts --dir magnum/fts)
$ kubectl create -f fts-server.yaml
$ docker-compose fts-server.yaml
```

[https://indico.cern.ch/event/505613/contributions/227329/](https://indico.cern.ch/event/505613/contributions/227329/)
Use Cases

- And many others used to collect requirements…
  - Continuous Integration / Deployment
  - Swan / MyBinder / Jupyter Notebooks
  - ML / TensorFlow
  - …

- These have triggered a lot of the work we’ve done in the last months
  - Collaboration with Rackspace via CERN OpenLab
  - Indigo DataCloud project
    - https://indico.cern.ch/event/505613/contributions/2227435/
Performance

- How fast is cluster deployment?
- How does it scale with cluster size?
- How does a cluster itself scale?
- First try (May 16):
  - Kubernetes 1 million reqs/sec
  - Suboptimal latency
  - But we got to 2 million reqs/sec
Performance

- Second try (Aug 2016)
  - Much **better latency**
  - Managed **7 million requests / sec**
- And an analysis of cluster deployments

<table>
<thead>
<tr>
<th>Cluster Size (Nodes)</th>
<th>Deployment Time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2.5</td>
</tr>
<tr>
<td>32</td>
<td>4</td>
</tr>
<tr>
<td>128</td>
<td>5.5</td>
</tr>
<tr>
<td>512</td>
<td>14</td>
</tr>
<tr>
<td>1000</td>
<td>23</td>
</tr>
</tbody>
</table>
Conclusion

- Production end of October 2016
- Swarm, Kubernetes, Mesos one click away
- Integration with common HEP services
  - CVMFS, EOS
- Extensive scalability tests
  - Clusters of 1000 nodes, millions of requests / sec

- Ongoing Work
  - Node Groups
  - Lifecycle Operations

http://clouddocs.web.cern.ch/clouddocs/containers/index.html