Containers and Orchestration in the CERN Cloud

Ricardo Rocha, Mathieu Velten, Bertrand Noel

April 8th 2016, IT Technical Forum
Outline

- Goals
- Containers and Orchestration (Swarm, Kubernetes)
- OpenStack Magnum
- Use Cases
- Demos
- Status and Future Plans
Goals

- Integrate containers with OpenStack at CERN
  - Common Identity, Resource allocation, Networking, Data Access, ...
- Container orchestration agnostic
  - Support for Docker Swarm, Kubernetes, Mesos, ...
- Fast and easy to use
  - Quick launch, easy scaling of clusters
Container Overview
Container Overview

- Process isolation - kernel namespaces & cgroups
- Same kernel, improved performance
- Microservices
- Images repository
  - dockerhub
  - private repos: docker.cern.ch pilot from Linux team
Container Orchestration

- Core OS
- Apache Mesos
- Docker
- rkt
- Consul
- etcd
- Project Atomic

And a looooot more...
Docker Swarm / Compose

- Clustering of Docker nodes
- Native Docker API
- Docker Compose for orchestration

- If it works with Docker, it works with Swarm!
Docker Swarm: uses the docker client

```bash
docker info
Containers: 3

Nodes: 2

docker run -d nginx
7100f1bc8619580a8b9e70bb5c12e60a4bd7e543f189e26018ef1924fd641a0a

docker ps
7100f1bc8619 nginx "nginx -g 'daemon off" 18 seconds ago Up 9 seconds 80/tcp, 443/tcp

7160f045dcf0 /bin/sh
[root@1836f9c42754 /]
```

Docker Swarm: advanced features

- **Scheduler filters**
  - Node filters - where should my container run (SSDs or not? Storage driver?)
  - Container filters - affinities, dependencies, ports, etc

- **Labels**
  - Tag a node with a label (docker daemon config)
  - Tag a container with a label
  - Use labels later for scheduling
  - Examples: production vs dev, fast vs slow storage

```
docker run ... -e affinity:container==frontend -e constraint:storage==ssd
```
Docker Compose

- Container orchestration
- Easy way to define a full application stack in one file
  - Networks
  - Containers
  - Affinities
  - Exposed ports
  - ...
- Additional features to scale applications, access logs, ...
Docker Compose

lb:
  image: docker.io/tutum/haproxy
  ports:
    - 80:80
  links:
    - web
web:
  image: docker.io/rochaporto/python-redis
  expose:
    - 5000
  links:
    - redis
redis:
  image: docker.io/redis

docker-compose up
docker-compose scale web=3
Kubernetes

- Container orchestration with some more advanced concepts
- Pod
  - Unit of deployment, one or more containers
  - Sharing network, filesystem areas
- Service
  - Entrypoint to a service (tcp, http)
- Replication Controller
  - Manages number of Pod instances
  - Scaling
- Auto scaling policies
apiVersion: v1
kind: ReplicationController
metadata:
    name: redis-controller
spec:
    replicas: 1
    selector:
        app: redis
    template:
        metadata:
            labels:
                app: redis
        spec:
            imagePullPolicy: Always
            containers:
                - name: redis
                  image: redis
                  ports:
                      - containerPort: 6379

apiVersion: v1
kind: ReplicationController
metadata:
    name: web-controller
spec:
    replicas: 1
    selector:
        app: web
    template:
        metadata:
            labels:
                app: web
        spec:
            imagePullPolicy: Always
            containers:
                - name: web
                  image: docker.io/rochaporto/python-redis
                  env:
                      - name: REDIS_HOSTNAME
                        value: 10.254.13.13
                  ports:
                      - containerPort: 5000
Kubernetes

`kubectl create -f stack.yaml`

`kubectl get pods`

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>redis-controller-u0b3m</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>1m</td>
</tr>
<tr>
<td>web-controller-r2d8z</td>
<td>1/1</td>
<td>Pending</td>
<td>0</td>
<td>1m</td>
</tr>
</tbody>
</table>

`kubectl scale --replicas=3 rc web-controller`
scaled

`kubectl get pods`

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>redis-controller-u0b3m</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>10m</td>
</tr>
<tr>
<td>web-controller-0ya1w</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>10s</td>
</tr>
<tr>
<td>web-controller-4qhzj</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>10s</td>
</tr>
<tr>
<td>web-controller-r2d8z</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>10m</td>
</tr>
</tbody>
</table>
## Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>Docker Swarm / Compose</th>
<th>Kubernetes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Docker API</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Expose Port</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Load Balancing</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Failover</td>
<td>Experimental</td>
<td>Yes</td>
</tr>
<tr>
<td>Node Scaling</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Container Scaling</td>
<td>Manual</td>
<td>Auto</td>
</tr>
<tr>
<td>Cluster Network</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
OpenStack Magnum

- Container orchestration (COE) as first class resources in OpenStack
- Easy orchestration of container clusters
- Support multiple container engines
  - Swarm, Kubernetes, Mesos
- Native COE API access
  - Hard to abstract 100% functionality
- Higher level abstractions when possible
  - Like container-create, unclear if it will stay
## OpenStack Magnum Concepts

<table>
<thead>
<tr>
<th><strong>NODE</strong></th>
<th><strong>BAY MODEL</strong></th>
<th><strong>BAY</strong></th>
<th><strong>CONTAINER</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>A physical node or a virtual machine</td>
<td>A description of a container cluster</td>
<td>A container cluster</td>
<td>Runs in a specific bay</td>
</tr>
<tr>
<td>Runs one or more containers</td>
<td>Which node flavor, image, network to use</td>
<td>Based on a bay model</td>
<td>Not directly launched by Magnum today, use the native API</td>
</tr>
<tr>
<td></td>
<td>Which container orchestrator engine (COE)</td>
<td>Additional properties like name, number of nodes, number of masters</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Can be scaled (number of nodes)</td>
<td></td>
</tr>
</tbody>
</table>
OpenStack Magnum Usage

```
magnum baymodel-create --name rocha-swarm-model \
    --flavor-id m2.medium \
    --image-id fedora-atomic-23 \
    --keypair-id rocha-cern \
    --external-network-id CERN_NETWORK \
    --dns-nameserver 137.138.17.5 \
    --coe swarm

magnum bay-create --baymodel rocha-swarm-model --node-count 1 --name rocha-swarm-bay01

magnum ca-sign --bay rocha-swarm-bay01 --csr cert.csr > cert.pem
magnum ca-show --bay rocha-swarm-bay01 > ca.pem

magnum bay-show rocha-swarm-bay01 | grep api_address
| api_address | https://137.138.6.99:2376 |

vim env.sh
export DOCKER_CERT_PATH="/home/rocha/bays/rocha-swarm-bay01"
export DOCKER_HOST="tcp://137.138.6.99:2376"
export DOCKER_TLS_VERIFY="true"
```
OpenStack Magnum Usage

```
docker info

Containers: 3
Images: 15
Role: primary
Strategy: spread
Filters: health, port, dependency, affinity, constraint
Nodes: 2
gi-r-swarm-f23-cl2rjehq46cn-swarm-master-batso7sig26i.novalocal: 137.138.6.99:2375
  Status: Healthy
  Containers: 3
  Reserved CPUs: 0 / 2
  Reserved Memory: 0 B / 4.053 GiB
  Labels: executiondriver=native-0.2, kernelversion=4.3.3-301.fc23.x86_64, operatingsystem=Fedora 23 (Twenty Three), storagedriver=devicemapper
  Error: (none)
  UpdatedAt: 2016-02-24T14:04:08Z
...
```
OpenStack Magnum Usage

docker run -d nginx
7100f1bc8619580a8b9e70bb5c12e60a4bd7e543f189e26018ef1924fd641a0a


docker ps
CONTAINER ID    IMAGE                  COMMAND                              CREATED                 STATUS                    PORTS                  NAMES
7100f1bc8619    nginx                    "nginx -g 'daemon off"              18 seconds ago           Up 9 seconds              80/tcp, 443/tcp          gi-pmahou2a6o7-0-du6blcrn5x6h-swarm-node-4x51hufqcy7.novalocal/sad_visvesvaraya

magnum bay-update replace node_count=5
Container Use Cases
Easy scaling with Swarm: GitLab CI

- Continuous integration in GitLab (for the cloud team)
- gitlab-ci-multi-runner, using the docker executor
- Specific requirements: none
Infrastructure Services - FTS Example

- Currently scaling at node level
  - Frontend
  - Transfer Agent
  - Staging Agent (bringOnline)
  - Monitoring

- Scale instead at component level
- Think app component, forget about node
- Specific requirements: AZ awareness
Jupyter Notebooks - End User Analysis

- Everware, Binder, ROOTaaS/Swan, Recast
- Analysis environment at a click of a link
- Jupyter notebooks, web frontend
- Most work with Docker, simply point to Swarm

- Specific requirements: access to the usual storage
  - CVMFS, EOS
Ongoing Work
Integration with CVMFS

- Implemented as a docker volume plugin
- Manages the CVMFS mounts on request (shared between containers)
- Integrated into Magnum @ CERN

```
magnum baymodel-create --name rocha-bay-model --labels cvmfs=true
```

- Usable with any docker deployment

```
docker volume create -d cvmfs --name atlas.cern.ch

docker run -it --volume-driver cvmfs -v atlas.cern.ch:/atlas centos:7 /bin/bash
```
Persistent storage (via OpenStack Cinder)

- "Attach X GB of persistent storage to Container Z"
  - And reattach later to another, ...
- At CERN this means getting a Ceph volume attached to my container
- Code is ready in upstream OpenStack Magnum
  - We’re testing it
- Kubernetes has built-in support, leverage on it

```
magnum baymodel-create --name rocha-bay-model --coe kubernetes --volume-driver cinder ...
```

- Swarm uses the REX-Ray docker volume plugin, which supports Cinder

```
magnum baymodel-create --name rocha-bay-model --coe swarm --volume-driver rexray ...
```
Upstream Contributions

- Everything we can is pushed upstream (as we do for all OpenStack projects)
- Puppet module contributions (puppet-magnum)
- Installation guide for Magnum
- Docker storage driver selection
  - x5 performance improvement using overlayfs instead of devicemapper (Fedora default)
- OpenStack Rally integration (monitoring of service in production)
- Availability Zones (AZ) awareness
- And other smaller patches..

- CERN OpenLab / Rackspace fellow (Spyros Trigazis)
- Also work in the context of the Indigo DataCloud project
Demos

- Deployment of a Kubernetes cluster
- Everware / Jupyter notebooks
- Scaling distributed processing
Summary & Plans

● Pilot service deployed, used by ~10 projects
  ○ Covering common use cases in our environment
  ○ Using production resources, but enabled only for a subset of projects
● Kubernetes, Docker Swarm/Compose fully supported
● With CVMFS and persistent storage support, first set of requirements fulfilled

● And coming next...
● Investigate access to EOS (credential handling is tricky)
● Integration with LBaaS (load balancing)
● Container Monitoring (cAdvisor, Heapster)
● Bay auto scaling (unclear on policies to use)
When can i use it?

- If you have a nice use case, you can use the pilot service today
  - Drop us an email to get access
- No big changes foreseen, service is quite stable
- Couple of bug fixes left to deal with

- Aiming for production Q3 2016
Questions?

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