Managing Docker containers and Kubernetes clusters in the EGI Cloud

Enol Fernández
Cloud Solutions Manager – EGI Foundation
Outline

• Containers & container orchestration
  ▪ Docker
  ▪ Kubernetes

• Running containerised applications in EGI
  ▪ udocker
  ▪ Docker VM
  ▪ Kubernetes

• Q & A
Containers

OS – level virtualisation
Containers

• OS-level virtualisation
  ▪ Isolated runtime built on Linux namespaces and cgroups, without VMM overhead
Why Containers?

• Container ship all application code and dependency on a single package
  ▪ Run the same thing everywhere from devel to production

• Containers share the machine’s OS system kernel
  ▪ Little to none overhead, server consolidation

• Containers run isolated from other applications
  ▪ OS provides a sandboxed view of the system where all resources are virtualised
Docker

• “Open-platform for building, shipping and running distributed applications”

• Docker commoditizes containers
  ▪ Hides and automates container management process
  ▪ One-command-line deployment of applications
  ▪ Easy to move from development to production
  ▪ Provides ecosystem to create and share images
Container images

• Images are the read-only template from which containers are started from
• Layers of filesystems built with a Dockerfile

Sample from https://github.com/docker/labs/blob/master/beginner/chapters/webapps.md
DockerHub / container registries

• Container registry service for sharing applications

• DockerHub is the main registry available
  ▪ Public and private repositories
  ▪ Official repositories
  ▪ Automated builds
### DockerHub

**thank you for attending DockerCon 2020! watch the recordings.**

<table>
<thead>
<tr>
<th>Repository Name</th>
<th>Stars</th>
<th>Forks</th>
<th>Visibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>egfeciocloud</td>
<td>0</td>
<td>0</td>
<td>PUBLIC</td>
</tr>
<tr>
<td>egfeciocloud / egfeciocloud</td>
<td>0</td>
<td>0</td>
<td>PUBLIC</td>
</tr>
<tr>
<td>egfeciocloud / case</td>
<td>205</td>
<td>0</td>
<td>PUBLIC</td>
</tr>
<tr>
<td>egfeciocloud / cloudboids</td>
<td>330</td>
<td>0</td>
<td>PUBLIC</td>
</tr>
<tr>
<td>egfeciocloud / siteboids</td>
<td>226</td>
<td>0</td>
<td>PUBLIC</td>
</tr>
<tr>
<td>egfeciocloud / boids</td>
<td>271</td>
<td>0</td>
<td>PUBLIC</td>
</tr>
<tr>
<td>egfeciocloud / cloudkeeper-os</td>
<td>275</td>
<td>0</td>
<td>PUBLIC</td>
</tr>
<tr>
<td>egfeciocloud / cloudkeeper</td>
<td>794</td>
<td>0</td>
<td>PUBLIC</td>
</tr>
<tr>
<td>egfeciocloud / rps-cloud-info</td>
<td>90</td>
<td>0</td>
<td>PUBLIC</td>
</tr>
</tbody>
</table>
• Just use docker run!
  ▪ Usual practice: 1 container = 1 process

• What if I have to run more than one process/container? → Docker Compose
  ▪ Define multi-container applications
    o What containers to run? (services)
    o How to communicate between them? (network)
    o How to interact with storage? (volumes)
  ▪ Manage the application from a single tool (docker-compose)

• What if 1 machine is not enough? → Container orchestration
version: '2.0'
services:
  web:
    build: .
    ports:
    - "5000:5000"
  volumes:
    - .:/code
    - logvolume01:/var/log
  links:
    - redis
  redis:
    image: redis
  volumes:
    logvolume01: {}
Container orchestration

- Schedule containers to physical or virtual machines
- Restart containers if they stop
- Provide private container network
- Scale up and down
- Service discovery

App A

App B

Container Orcheduler

Infrastructure
Kubernetes

- Kubernetes is an **open-source platform for automating deployment, scaling, and operations of application containers across clusters of hosts, providing container-centric infrastructure.**

- Kubernetes:
  - Places containers on nodes
  - Recovers automatically from failure
  - Basic monitoring, logging, health checking
  - Enables containers to find each other
Kubernetes principles

• Declarative > imperative
  ▪ API driven application management
  ▪ Agents monitor endpoints for state changes (real-time)
  ▪ Controllers enforce desired state, self-healing

• No hidden APIs
  ▪ Immutable
  ▪ Modularity

• Workload portability
  ▪ Decouple application development from cluster implementation
  ▪ Cattle > pets
Kubernetes Cluster

Taken from https://github.com/jbeda/slides-kubernetes-101
Kubernetes: Pod, ReplicaSets

Taken from https://github.com/jbeda/slides-kubernetes-101
Kubernetes Services

Taken from https://github.com/jbeda/slides-kubernetes-101
Kubernetes: sample

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
# modify replicas according to your case
  replicas: 3
selector:
  matchLabels:
    tier: frontend
template:
  metadata:
    labels:
      tier: frontend
  spec:
    containers:
    - name: php-redis
      image: gcr.io/google_samples/gb-frontend:v3
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

Examples taken from kubernetes docs: [https://kubernetes.io/docs](https://kubernetes.io/docs)
• Helm is a package manager for Kubernetes
• Allows to easily install applications/services easily
• Creates from templates all the k8s objects needed to run the application
• Wide catalogue of charts with (normally) sane defaults
Containers on EGI
Containers as jobs

Using HTC for running containers

• Docker requires root access to run your container
  ▪ Need alternative runtimes for running jobs

• Singularity:
  ▪ Container image and runtime specifically designed to run on multi-tenant computational resources
  ▪ Conversion of images from docker (automatic)
  ▪ Requires system administrator to configure

• udocker:
  ▪ run applications encapsulated in docker containers without using docker, without using privileges, without system administrators intervention and without additional system software
  ▪ Run as a normal user, with the normal process controls and accounting, in interactive or batch systems
udocker

- Can be installed on-the-fly as regular user as part of job preamble
  - Requirement: Python
- Provides a docker-like CLI
- Pulls images from Docker Hub
- Modular engine to execute images
- Support for GPGPUs and MPI applications

https://github.com/indigo-dc/udocker
Containers on EGI Cloud Compute

Run docker on VMs

- Pre-built image with Docker, Docker-compose and Kubernetes installed
- Instantiate at one provider, ssh and:
  
  ```bash
  $ docker run ...
  ```
Cloud Container Compute gives you the ability to deploy and scale Docker containers on-demand using Kubernetes technology. The service provides with easy provision of Kubernetes clusters on EGI Cloud Compute resources that can be scaled and upgraded without the overhead of installing, managing and operating the nodes.
Cloud Container Compute [BETA]

1. Provision cluster VMs

2. Deploy Kubernetes cluster on provisioned VMs

3. Use native Kubernetes tooling

EGI Cloud Container Compute

IaaS Federated Access Tools

IaaS API

Cloud Management Framework

EGI Cloud Compute Service
Deploy with EC3

• EC3: Elastic Cloud Computing Cluster

• Deploys and configures Kubernetes
  ▪ Elasticity – can grow/shrink depending on your load
  ▪ GUI and CLI access
  ▪ Integrated with Check-in and AppDB

• Builds on ansible and kubeadm
Kubernetes in EGI

• Provides Kubernetes 1.16+

• LoadBalancer ServiceType:
  ▪ A NGINX ingress configured by default ready to be used offering similar functionality
  ▪ Expandable with auto-configuration of Let’s Encrypt certificates

• Volumes
  ▪ NFS driver for flexible and shared volumes
  ▪ Cinder (OpenStack volume) driver also available

• Check-in integration
  ▪ Use Check-in tokens for AuthN & AuthZ in the Kubernetes API!
  ▪ https://kubernetes.io/docs/reference/access-authn-authz/authentication/#openid-connect-tokens
EGI Notebooks

Running on top of Kubernetes

- JupyterHub hosted in the EGI Cloud
  - Offers Jupyter notebooks ‘as Service’
  - One-click solution: login and start using
Kubernetes and JupyterHub

- ingress
  - proxy service
    - proxy deployment
      - proxy pod
        - spawns: create pods and volumes
          - redirects
            - hub deployment
              - Hub pod
                -Hub volume
                  - single user pod
                    - User volume
                      - redirects
                        - Configurable HTTP Proxy
                          - /hub/
                            - /user/[name]/
                              - /api/auth
                                - Notebook
                                  - Authenticator
                                    - User Database
                                      - Spawner
                                        - Hub
                                          - Browser
                                            - www.egi.eu
                                              - 25/06/2020
• User documentation:
Thank you for your attention.

Questions?