Plancton: an opportunistic distributed computing project based on Docker containers

Matteo Concas, Università di Torino
mconcas@cern.ch

Thursday, 26 May 2016 -Torino Cloud Users Micro-Workshop

Outline

- What are Linux containers?
- Docker: Linux containers for everybody
- What is Plancton for
- A disposable pilot approach
- What Plancton is
- Use case example: A volunteer batch facility for ALICE
  - Volunteer container with HTCondor and ALICE software
  - Results
- Conclusions
- Outlook

What are Linux containers

- «Linux Containers is an operating-system-level virtualisation method for running multiple isolated Linux systems on a single control host» [wiki.archlinux.org]
- **Containers are not lightweight VMs**: they are **chroot on steroids**
  - They are usually employed to sandbox applications or whole services in optimal and consistent environments
  - In some scenarios they could behave in a similar way to Virtual Machines, sometimes they can even offer new approaches
- **Less features than VMs:**
  - No custom kernel nor virtual hardware are virtualised in a container
  - Containers on the same host share the same Linux kernel

Docker: Linux containers for everybody

- Docker is a powerful and versatile tool to build, deploy and manage Linux containers. To start an interactive shell, just type:

```
$ docker run –rm -it ubuntu /bin/bash
```

- We can wrap applications in a consistent environment built *ad hoc,* without affecting the underlying host space ➙ Docker provides isolation inter-processing and with the rest of the host
- We can add volumes, capabilities, and software inside each container or even build one from scratch through Dockerfiles.

Plancton: Docker containers for scientific computing?

- Providing consistent environments for job execution
- Ensuring isolation between jobs

} Linux containers provide such features

Docker makes Linux containers trivial!

- Container deployment for free
- Easy interface for cooking new images
- exposes powerful REST API

A disposable pilot approach

- Thanks to their **low deployment times** and their **low overhead**, it is possible to adopt an approach based on pilot containers
  - Container are spawned with a limited time to leave (TTL)
  - They patiently wait for a job; if anything comes it is processed; container exits and it is disposed of at the end ➙ **one job - one container**
  - If no job comes the container exits after a short while without doing anything
- With Virtual Machines this approach is too onerous
  - Plenty of overhead and non-negligible deployment time ➙ disposable approach not convenient

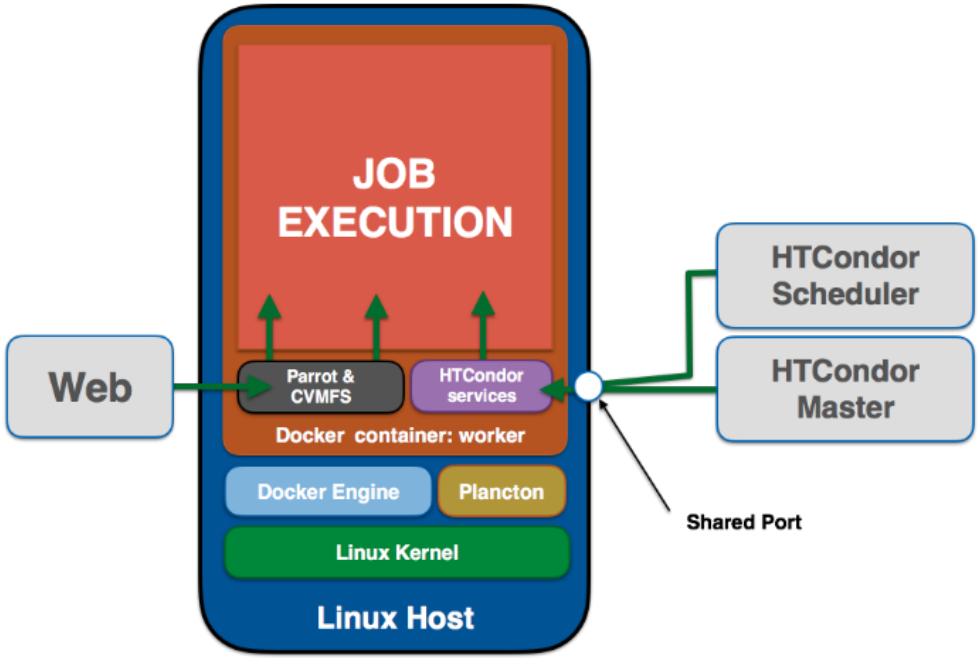
What Plancton is

- It is a simple automation: while(1) with whistles and bells
  - Periodically repeat a simple routine
  - Case-specific requirements are specified through configuration files
- Workflow
  - Check **available resources** (CPU measurement, …)
  - If enough resources available ➙ **start new container** (multiple Docker images possible)
  - Other basic checks and operations (**clean up exited** containers, query status, etc…)
  - Check **overdue** containers (job taking too long, stopped ones…) to handle misbehaviours

Use case example - Volunteer batch facility for ALICE

- **Head Node** (dedicated login machine)
  - HTCondor schedd, collector daemons ➙ needed for job submission
- **Host Nodes** (Plancton nodes)
  - Plancton spawns Docker containers designed for volunteer computing
  - Physical nodes require only Plancton and Docker - works in a firewalled environment (no incoming connections)
  - Each Plancton daemon may push metrics for monitoring/accounting purposes (work in progress with Elasticsearch)

Volunteer container with HTCondor and ALICE software (2)



Plancton Host

- Summary
  - Applications fully "docked"
  - Extra container config not fitting in Docker image is injected at deploy time (e.g. secrets...)
  - Containers for isolation and known environment chosen with an eye on volunteer/ opportunistic computing: **can we use same design on a dedicated facility?**

Use case example - Results

- Built a small farm on few hosts for test&debug
  - Successfully added virtual nodes from Centro di Calcolo di Torino (the Connection Broker made it possible) ➙ larger test environment for development purposes
  - Plancton works reliably and does not interfere with user's activities

Conclusions

- Docker containers as pilot worker nodes are a viable way to deploy consistent environments for running jobs (*e.g.* SLC6 containers running on Ubuntu 16.04…)
- Resource capping and isolation come natural with Docker

Outlook

- To test the generality of the tool we are working on supporting other use cases
- Most of this work is close to some activities within the **INDIGO-DataCloud** project; we are in touch with the Torino people working in the project to explore possible collaborations