Using the autopilot pattern to deploy container resources at a WLCG Tier–2

Gareth Roy*, David Crooks, Gordon Stewart, Samuel Skipsey, David Britton

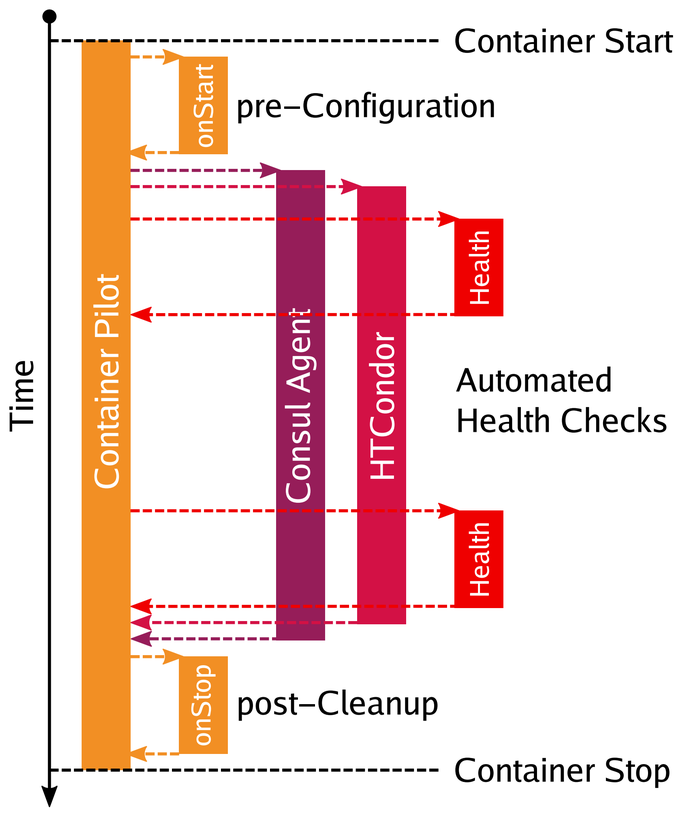
School of Physics and Astronomy, University of Glasgow, G12 8QQ, Scotland
*gareth.roy@glasgow.ac.uk

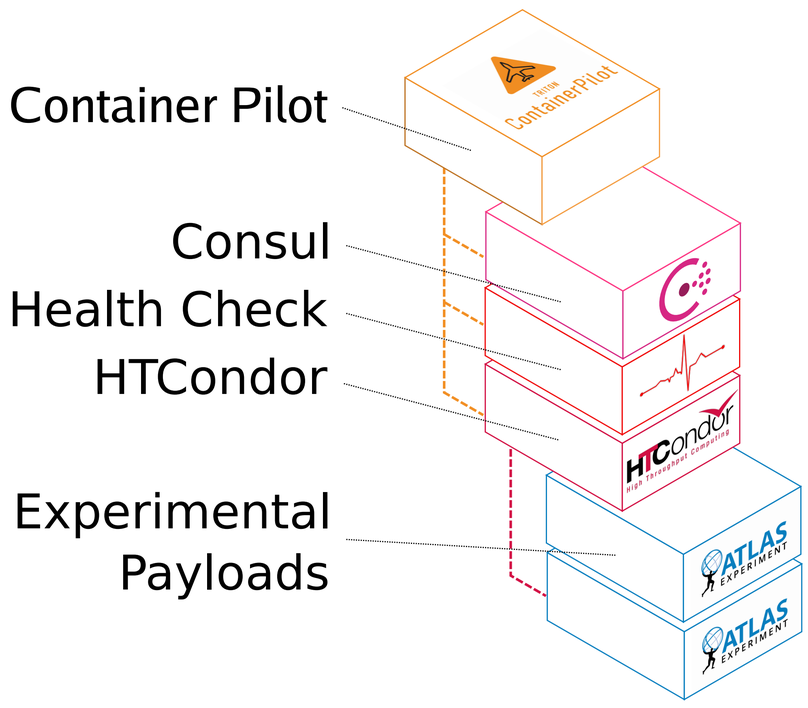
Containers are becoming ubiquitous within the WLCG with CMS announcing a requirement for Singularity at supporting sites in 2018. The ubiquity of containers means it is now possible to reify configuration along with applications as a single easy–to–deploy unit rather than via a myriad configuration management tools such as Puppet, Ansible or Salt. This allows more use of industry devops techniques such as Continuous Integration (CI) and Continuous Deployment (CD) within the operations domain, leading to faster upgrades and more secure systems.

One interesting technique is the AutoPilot pattern [1] which provides mechanisms for application lifecycle management from within the container itself. Using modern service discovery techniques each container manages its own configuration, monitors its own health and adapts to changing requirements through the use of event triggers.

In this work, we use Consul [2] as a mechanism for distributed service discovery allowing containers to register and retrieve configuration information, and Container Copilot [3] to manage the processes and services running within each container.

Container Lifecycle



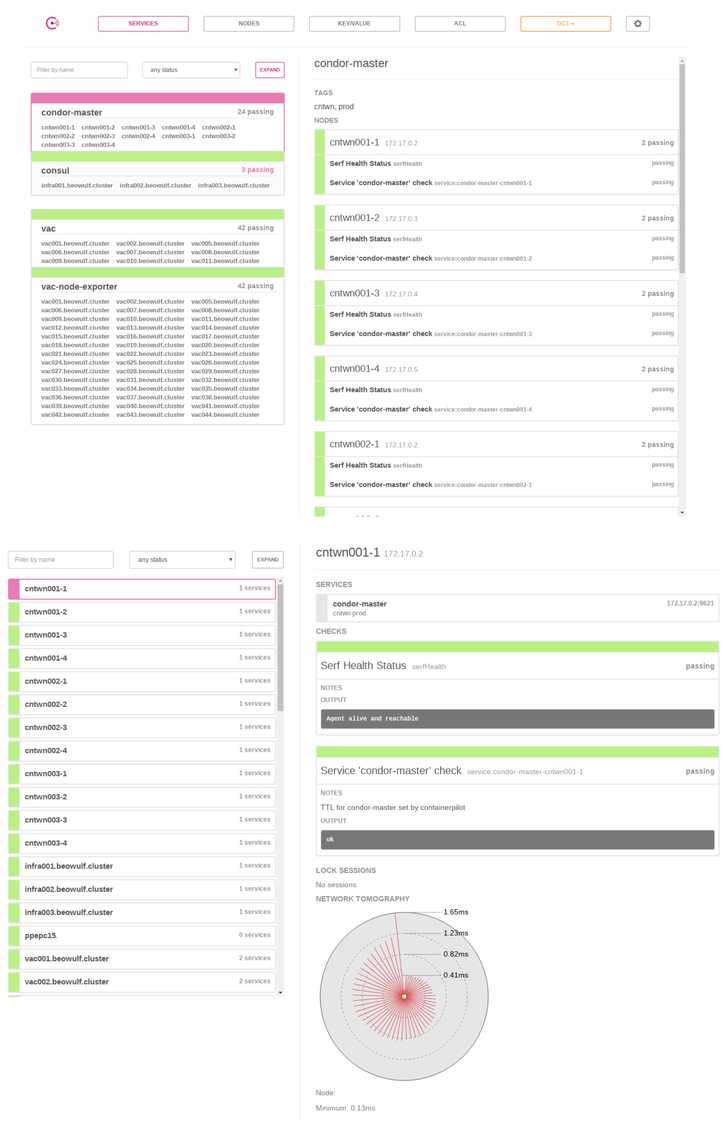


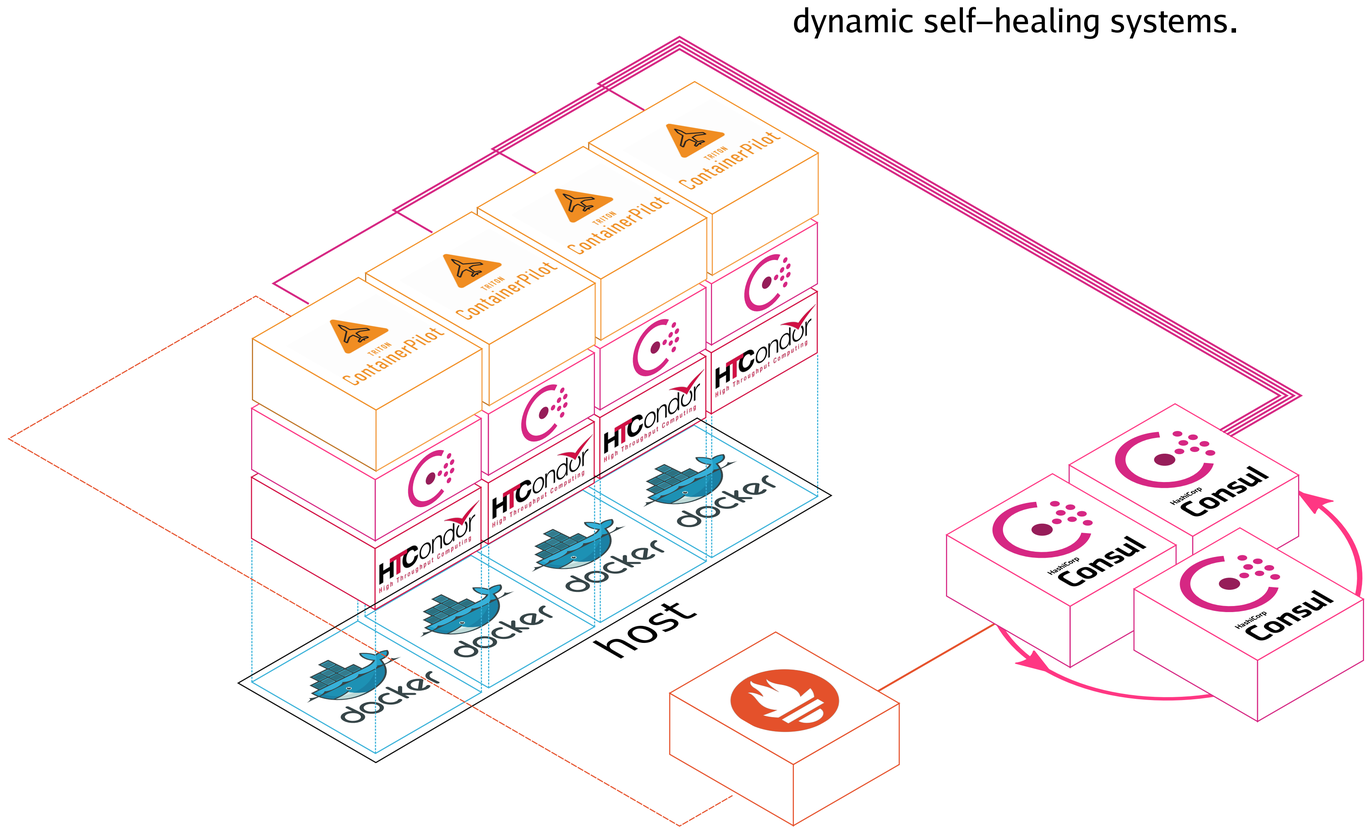
Container Copilot acts as an init system (similar to systemd) within the container itself. When a container is instantiated, its entrypoint is set to run the Container Copilot binary. This binary carries out a set of pre–configuration tasks and then runs and manages any jobs given to it. In our system, we use it to start both a Consul agent along with the HTCondor master daemon.

Each instantiated container registers its existence with the main Consul cluster allowing automated discovery of all the HTCondor endpoints; this allows instantiated containers to be added to site monitoring tools such as Prometheus as well as to match container IDs to servers for auditing purposes.

Container Copilot requires the specification of Healthchecks for all jobs run by it. This allows local periodic monitoring which can carry out repair actions (like restarting stuck daemons) as well as registering state to the Consul service discovery layer. This leads to the construction of dynamic self–healing systems.

Service Discovery via Consul





Conclusion

Using the AutoPilot pattern we have been able to create containers capable of running WLCG payloads that manage their own configuration and lifecycle. These containers use modern service discovery techniques to register their existence which allows automated monitoring and logging to take place.

Additionally, built–in health checking allows each container to monitor its current state, report on its health and take appropriate action to self–heal. It is hoped that applying these techniques across a WLCG Tier–2 site will lead to reduced downtime for resources and an overall reduction in the amount of manpower required to run such a site.

[1] http://autopilotpattern.io
[2] https://consul.io
[3] https://github.com/joyent/containerpilot

University of Glasgow
VIA VERITAS VITA

GridPP
UK Computing for Particle Physics