CONTAINERIZED CONTROL STRUCTURE FOR ACCELERATORS

I. Arredondo (inigo.arredondo@ehu.eus), J. Jugo, University of the Basque Country, Leioa, Spain

Nowadays, modern accelerators are starting to use virtualization to implement their control systems. Following this idea, one of the possibilities is to use containers. **Containers are highly scalable, easy to produce/reproduce, easy to share, resilient, elastic and low cost in terms of computational resources. All of those are characteristics that fit with the necessities of a well defined and versatile control system**. In this paper, a control structure based on this paradigm is discussed. Firstly, the technologies available for this task are briefly compared, starting from containerizing tools and following with the container orchestration technologies. As a result, Kubernetes and Docker are selected. Then, the basis of Kubernetes/Docker and how it fits into the control of an accelerator is stated. Following the control applications suitable to be containerized are analyzed: electronic log systems, archiving engines, middleware servers,... Finally, a particular structure for an accelerator based on EPICS as middleware is sketched.

Control Systems and Virtualization

Control System Characteristics:

- *Maintainable*:
  - Easy to describe and deploy automatically.
  - Based on few standards.
  - Based on reliable technologies:
    - + Proven/Tested.
    - + Supported and maintained (vendor, community). Easy to share.
    - + Being developed to fit new challenges.
    - + Better if open source: Particular necessities.
- *Robust*: Early error detection and fast recovery.
- *Easy to scale*: To accomplish upgrades.
- *Efficient*: Use the resources efficiently and maintain low energy consumption.

Virtualization fits very well with the most of the control system needs

As reported by J-Parc, HZB, DELTA, NIF, CERN,... the main uses of virtualization are:

1.- Add High Availability.
2.- Resources optimization.
3.- Improve maintainability: Easy backup and upgrades/changes.
3.- Software standardization and maintaining.

They state also two main **drawbacks**:

1.- A host crash implies several services malfunctioning.
2.- Large system recovery time.

Containers Orchestration

Orchestrators offer:

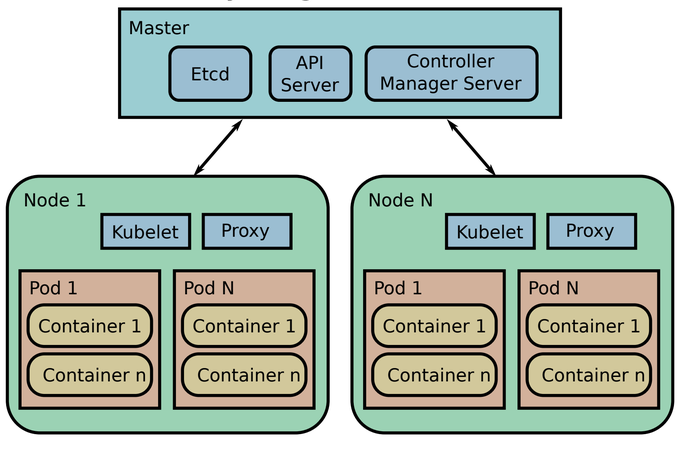
- *Provisioning*: Provide and launch containers efficiently.
- *Configuration-as-text*: For easy edition, versioning and sharing.
- *Monitoring*: Check the health of the containers in the cluster.
- *Rolling Upgrades and Rollback*: Incremental upgrading.
- *Policies for Placement, Scalability etc.*: Load balancing, HA.
- *Service Discovery*: Container placement agnosticism.
- *Easy Administration*: Integration with the IT infrastructure.

Orchestration tools solve the main drawbacks of the virtualization in control systems

Orchestration tool selection

- *Kubernetes*:
  - Tested tool for medium-large clusters.
  - Large and active community.
  - Managed by the Cloud Native Computing Foundation (CNCF): Amazon, CoreOS, Mesosphere, Samsung, Microsoft, Red Hat, IBM, Intel, Oracle, Docker, Cisco, Google,...
- *Swarm*: The best Docker compatibility and it is easy to use. It is preferred for no very large clusters.
- *Mesos*: Well proven tool (Twitter, eBay, Netflix,...). Designed and tested for very large clusters.

Selected: **Kubernetes**



Containerization

Nowadays, there are two alternatives for virtualization: Virtual Machines (VMs) and Containers.

Containers vs Virtual Machines

- Benefits of containers:
  - Fast application delivery: Maintenance and update.
  - Better scaling: In case of more resources need. About 22 times better than VMs.
  - More rapid spawning and termination.
  - Better resource utilization (lightweight).
    - + Higher workloads with grater density.
    - + Containers are able to run multiple isolated processes in a host without the overhead caused by the hypervisor layers introduced by VMs.
- Benefits of VMs:
  - Better compatibility and isolation.
  - More robust that containers.

Containerization fits better than VMs for a control system services

Container selection

Main container Alternatives on Linux: *LXC, rkt, Docker*.
- Based on their capabilities they are similar.
- rkt and Docker are the ones with the largest inertia.

Focused on non critical services the election is based on:

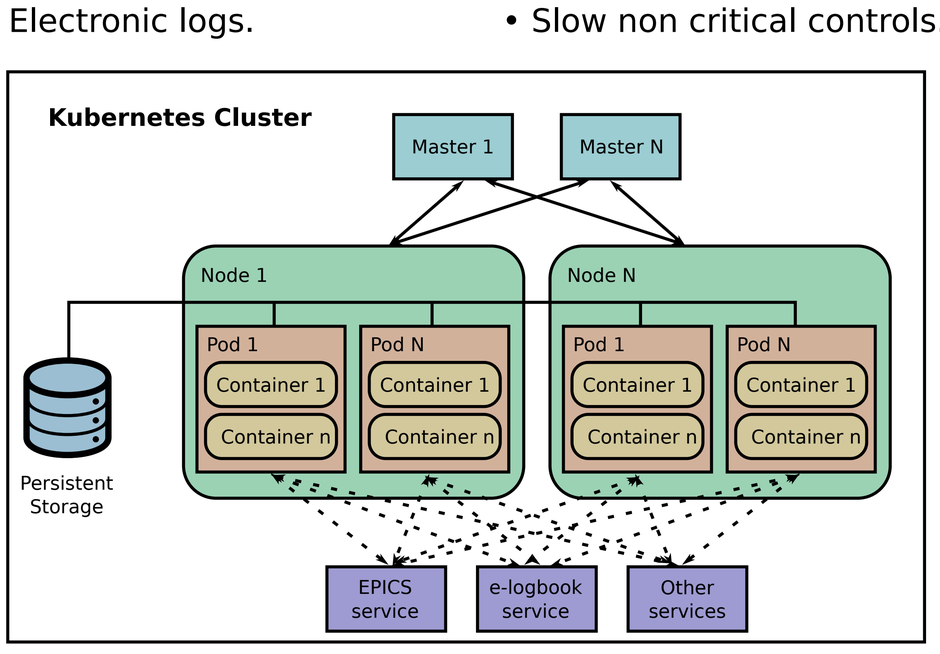
- Ease of utilization
- Support
- Active development
- Compatibility with orchestration tools, etc.

Selected: **Docker**

Container based control architecture

Manage non critical services of the control system.

- Alarm handlers.
- Data archivers.
- Electronic logs.
- LDAP services.
- User interface managers.
- Slow non critical controls.



EPICS based control system:

- e-logbook
- EPICS Archiver Appliance
- EPICS base
- EPICS virtual IOCs: soft IOCs + network devices.
- open LDAP
- BEAST alarm handler

Measured less than 10 seconds from an EPICS virtual IOC crash to its recover