Containerization and its applications for CERN Accelerator Controls

Rémi Voirin

01/12/2021 – Academic Training Lecture – Controlling the CERN Accelerator Complex (3/4)
Overview

Key concepts

- UNIX process isolation: jails, namespaces, containers
- Container orchestration
- Microservices vs monoliths
- Containers at CERN and in Controls

Prerequisites

- UNIX systems basics (kernel, process, init)
- Network basics (DNS, IP address)
- Overall structure of CERN (in particular the A&T Sector, BE and IT Departments)
Outline

Isolation mechanisms in UNIX systems

Containers in industry and at CERN

Status and use cases for Accelerator Controls
Table of Contents

Isolation mechanisms in UNIX systems

Containers in industry and at CERN

Status and use cases for Accelerator Controls
Purposes of isolation

/opt/somestate
Purposes of isolation

/usr/lib/library.so

# yum update
# apt upgrade
# pacman -Syu
Purposes of isolation
Purposes of isolation

Attack
Technology example: FreeBSD jails

Released in 2000 in version 4.0

- Complete system sandboxing
- Main goal: enhanced security
  - Separate file hierarchy (visible filesystem)
  - Separate visible processes
  - Separate users
  - Different installed packages
  - Different IP addresses
Technology example: FreeBSD jails

DEMO TIME!

• More info on jails:
  • FreeBSD Jails Quickstart
    • https://zenlot.medium.com/freebsd-jails-quick-start-64235c6a4361
  • Official FreeBSD documentation
    • https://docs.freebsd.org/en/books/handbook/jails/
Technology example: Linux kernel namespaces

First namespace introduced in 2002 in version 2.4.19

- Eight system calls for eight isolation areas:
  - Mount (CLONE_NEWNS)
  - InterProcess Communication (CLONE_NEWIPC)
  - Time (CLONE_NEWTIME)
  - Time-sharing (CLONE_NEWUTS)
  - User (CLONE_NEWUSER)
  - PID (CLONE_NEWPID)
  - network (CLONE_NEWNET)
  - cgroups (CLONE_NEWCGROUP)

For more info:
$ man 7 namespaces
The cgroup namespace

Linux Control Groups

• Underlying concepts:
  • Slices
  • Resources

• Control Groups can be applied to resources.

This is a Key lime pie.
The cgroup namespace

DEMO TIME!

• Learn more at:
  • $ man 7 cgroups
  • Inside cgroups
    • https://0xax.gitbooks.io/linux-insides/content/Cgroups/linux-cgroups-1.html
  • Series of detailed articles
    • https://www.redhat.com/sysadmin/cgroups-part-one
Industrializing isolation mechanisms

Docker (2013)

• Provided an ecosystem around Linux namespaces
  • Public registry to store “container images”: the Docker Hub
  • One command to deploy a containerized application

The Open Container Initiative (2015)

• Goal: “design standards for operating-system level virtualization” (i.e. containerization)
• Released specifications on storing container images and running containers
• “OCI-compliance” in Docker, Podman, Singularity,...
Containerization vs virtualization

### Container
- Application and dependencies
- Container engine
- System layer or PID 1
- Operating system
- Hardware

### Virtual machine
- Application
- System layer or PID 1
- Operating system
- Hypervisor
- System layer or PID 1
- Operating system
- Hardware
**Containerization: new paradigm**

**Vocabulary**

- Package / stored application (rpm, deb, binary,…) → Image
- Repository → Registry
- Deploy and start an application → Run (a container)
- Download → Pull
- Upload → Push
Table of Contents

Isolation mechanisms in UNIX systems

Containers in industry and at CERN

Status and use cases for Accelerator Controls
Plain containerization: general use cases

Containers are becoming the norm in the industry for:

- Idempotent execution between development and operational environments
- Streamlining the use of DevOps tools
- Managing application dependencies
- Encapsulating legacy solutions
Container orchestration

tcp://myapp.containers.cern.ch:8090

Load balancer

containers.cern.ch

Server

myapp container 1
otherapp container 1

Server

myapp container 2
otherapp container 2

Server

myapp container 3
otherapp container 3
Container orchestration

What you get

• Scalability at hand, can be fully automated
• Load balancing “for free” and specific deployment strategies
• Zero downtime if you run multiple replicas

Requirements

• Hardware and geographical redundancy
• To get all the goodness: microservice architecture
• Describe your application to deploy it
Example: Machine-Learning Platform

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: simple-ann-subpackage--secondmodel-simple-ann-default-deployment
  labels:
    app: simple-ann-subpackage--secondmodel-simple-ann-default
spec:
  replicas: 3
  selector:
    matchLabels:
      app: standalone-model
      modelType: simple_ann-subpackage--secondmodel
      modelParameters: simple_ann_default
  template:
    metadata:
      labels:
        app: standalone-model
        modelType: simple_ann-subpackage--secondmodel
        modelParameters: simple_ann_default
    spec:
      containers:
      - name: simple-ann-subpackage--secondmodel-simple-ann-default
        image: registry.cern.ch/acc/simple-ann-subpackage--secondmodel
```
Example: Machine-Learning Platform

Curtesy of Jean-Baptiste de Martel

```yaml
apiVersion: v1
kind: Service
metadata:
  name: simple-ann-subpackage--secondmodel-simple-ann-default
spec:
  selector:
    app: standalone-model
    modelType: simple_ann-subpackage--secondmodel
    modelParameters: simple_ann_default
  ports:
    - protocol: TCP
      port: 5000
      targetPort: 5000
```

Kubernetes PoC for accelerators

Deployment

Service

Ingress
Example: Machine-Learning Platform

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: simple-ann-subpackage-secondmodel-simple-ann-default
  namespace: default
annotations:
  kubernetes.io/ingress.class: traefik
  traefik.ingress.kubernetes.io/rule-type: PathPrefixStrip
  # traefik.ingress.kubernetes.io/rewrite-target: /predict
  traefik.frontend.entryPoints: "http"
spec:
  rules:
  - host: .cern.ch
    http:
      paths:
      - path: /simple_ann-subpackage-secondmodel-simple_ann-default
        backend:
          serviceName: simple-ann-subpackage-secondmodel-simple-ann-default
          servicePort: 5000
```

Curtesy of Jean-Baptiste de Martel
Orchestration use case: scaling out

Infrastructure costs: $
Orchestration use case: scaling out

Infrastructure costs: $$$
On-premise or cloud provider?

… and many more!
Containers at CERN (outside of Controls)

• Plain containers: use your existing system and GitLab, IT images

• Container orchestration: two main choices

  - Kubernetes from the Cloud Container Service
    • Available for all orchestrated deployments
    • More powerful but steeper learning curve
    • Fine tunable
    • GPN (supported) and for accelerators (PoC)

  - OPENSHIFT
    • Made for hosting web applications at CERN
    • User-friendly web interface
    • Many built-in features e.g. TLS certificates
    • Operates from the GPN only

Doc: https://clouddocs.web.cern.ch/containers/quickstart.html
Doc: https://cern.service-now.com/service-portal/?id=kb_article&n=KB0004358

Info on training in extra slides
Table of Contents

Isolation mechanisms in UNIX systems

Containers in industry and at CERN

Status and use cases for Accelerator Controls
Can run containers

Can be orchestrated

CERN's general purpose Ethernet network (GPN)

CERN's Gigabit Ethernet technical network (TN)

Front-end computers (FEC)

PLCs

VME crates

PICMG1.3 platforms

Analogue & digital interfacing

Accelerator Equipment

Fieldbuses
Leveraging container orchestration in Controls

Improved usage of the middle-tier bare metal infrastructure + data center redundancy

Data center 1 (today)
CPU/RAM usage: ~6%

Data center 1
Usage: up to 30%

Data center 2
Usage: up to 30%
Leveraging container orchestration in Controls

Quick failover of all applications

Data center 1

1

4

5

9

Generic

Storage

Synchronous

Data center 2

1 2 3 4 5

6 7 8 9 10

Generic

Storage
Leveraging container orchestration in Controls

Shorter Controls sysadmin days (1/3)

<table>
<thead>
<tr>
<th></th>
<th>46</th>
<th>47</th>
<th>48</th>
<th>49</th>
<th>50</th>
<th>51</th>
<th>52</th>
<th>Week 1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINAC4</td>
<td>Stop on 26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LINAC3</td>
<td>Stop on 19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ISOLDE stand alone**

**LHC Powering Campaign**

**CLEAR**

**SM18**

**INDIVIDUAL SYSTEM TESTS with remote controls LN4 PSB CPS SPS**

| LIN4 only |

**SPS beam Dump tests (SBDS) (SY-ABT)**

| Tests on 22Nov only |

**LHC CRYOGENICS**

| tbc |

**CERN Xmas holidays**

**CONTROLS SYS ADMIN DAYS [4-7 Jan]**

**Legend:**

- **operate**
- **shutdown**

Curtesy of Marine Gourber-Pace
Leveraging container orchestration in Controls

Shorter Controls sysadmin days (2/3)

Please deploy your software on the new server

Controls server 1

Controls server 2

Please deploy your software on the new server

Controls server 1new

Controls server 2new
Leveraging container orchestration in Controls

Shorter Controls sysadmin days (3/3)

Hardware to decommission  Pre-provisioned servers
Moving away from the “Linux shell” interface

Container orchestration changes the way the middle-tier is exposed to developers

• Give access to APIs instead
• More security by design
• Easier for developers
• Some use cases would be easy to translate
  • REST APIs (cf Bartek Urbaniec’s presentation tomorrow)
  • UCAP
  • Other scalable components (e.g. Grafana)
Containerization and its applications for CERN Accelerator Controls

- Virtual Device
  - Data processing algorithm
  - Event builder
  - Data updates

- UCAP Node

- Device Server

- FESA Server

- UCAP Node

- UCAP Node

- Event builder
  - Data processing algorithm
  - Data updates

- Device Server

- Virtual Device

- Server

- Server

- Server

- Server

- UCAP Node

- UCAP Node

- UCAP Node
Container orchestration requirements

Rethink network and storage

In the meantime, we address the need for plain containerization.
Plain containerization: overview

Purpose: offer the ability to run containers on servers and technical consoles

What we need:

• **Base images** that our developers can rely on (*details in extra slides*)
• **An image registry** to store container images
• **A container engine** to run containers on hosts
Plain containerization: registry approval mechanism

```
“\textit{I need Grafana on my Controls server}"
```

Git merge request
```
“Add grafana:8.1.5 to the list”
```

List of allowed images on the Controls registry

GitLab repository

CI pipeline

External images
CERN non-Controls images

Controls registry
Plain containerization: registry

<table>
<thead>
<tr>
<th>Name</th>
<th>Orphans</th>
<th>Artifacts</th>
<th>Pulls</th>
<th>Last Modified Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>acc/simple_ann_default</td>
<td>1</td>
<td>2</td>
<td></td>
<td>7/27/21, 155 PM</td>
</tr>
<tr>
<td>acc/ucap-build</td>
<td>2</td>
<td>5</td>
<td></td>
<td>8/17/21, 4:38 PM</td>
</tr>
<tr>
<td>acc/metrics-scraper</td>
<td>1</td>
<td>16</td>
<td></td>
<td>9/6/21, 2:51 PM</td>
</tr>
<tr>
<td>acc/dashboard</td>
<td>1</td>
<td>16</td>
<td></td>
<td>9/6/21</td>
</tr>
<tr>
<td>acc/debug</td>
<td>1</td>
<td>16</td>
<td></td>
<td>9/6/21</td>
</tr>
<tr>
<td>acc/nginx</td>
<td>1</td>
<td>106</td>
<td></td>
<td>9/6/21</td>
</tr>
<tr>
<td>acc/mlp-ci</td>
<td>1</td>
<td>16</td>
<td></td>
<td>9/6/21</td>
</tr>
<tr>
<td>acc/es</td>
<td>1</td>
<td>16</td>
<td></td>
<td>9/6/21</td>
</tr>
</tbody>
</table>
Plain containerization: engine

Podman is our container engine of choice:

- Containers run as ordinary Controls services

1. Declare the containerized service
2. Deploy and run in one command
Plain containerization: engine

Podman is our container engine of choice:

- Containers run as ordinary Controls services
- Rootless for security and practicality
- Daemonless architecture
- Community-driven project
Plain containerization: pros and cons

Drawbacks

- **More complex debugging** and diagnostics on the sysadmin side
- Risk of discouraging investment in getting rid of **legacy software**
- Need to manage **obsolescence** of containerized systems
Plain containerization: pros and cons

Advantages (1/2)

• **Unify building/packaging process with CI/CD templates**
  • Auto versioning and easy rollback

• **Relocatability, portability**
  • “Certified to run anywhere” when transferring to other labs
Plain containerization: pros and cons

Advantages (2/2)

- Decoupling between the operating system and software
  - Avoid constraints with new hardware
  - WinCC OA upgrades vs CentOS/RHEL upgrades
- Encourage good development practices
  - e.g. send logs to the tracing infrastructure
  - Implement Software Bill of Materials when available
Summary: containerized and orchestrated applications

<table>
<thead>
<tr>
<th></th>
<th>Non-containerized</th>
<th>Containerized</th>
<th>Orchestrated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unified packaging</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Portability</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decoupling OS/software</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ensure good dev practices</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deploy with API</td>
<td></td>
<td>Manual</td>
<td>Integrated feature</td>
</tr>
<tr>
<td>Replication &amp; scale-out</td>
<td>Manual</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zero-downtime strategy</td>
<td>Manual</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Load-balancing</td>
<td>Manual</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Unless tied to specific hardware (e.g. timing cards), all applications can benefit from containerization.

A list of active use cases is in extra slides, now let’s focus on one: the LHC Orbit Feedback.
Example use case: LHC Orbit Feedback testing

New orbit as BPM UDP Packets

Testing framework
\[ \Delta \text{Position} = \text{RM} \cdot \Delta K \]

RefOrbit

Optic

OP API (Java)

BFC (FESA)

Deflections (\( \Delta K \))

Curtesy of Andrea Calia
Example use case: LHC Orbit Feedback testing

- Goal: not having to rely on physical servers with timing cards
- Run on commit and daily
  - Based on Gitlab CI Pipelines
  - Currently ~100 unit tests
- Detailed history of issues
  - “This has happened before”
- Better development cycle
  - “It worked on my machine!”

More information and full presentation on the Evian Workshop 2021 page: https://indico.cern.ch/event/1077835/timetable/

Note: some of these components are not officially supported (e.g. Fesa-base)

Curtesy of Andrea Calia
Summary

Current containerization and orchestration come from twenty years of development

• From Linux kernel namespaces to Kubernetes

Containerization technologies are broadly used in industry

• Use cases: CI/CD pipelines, portability, scalability, redundancy,…

We’re brought plain containerization while working on necessary Controls network & storage improvements

• First step for developers: take a look at the Containerisation FAQ at https://wikis.cern.ch/display/CONT/Containerisation+Guide+and+FAQ

• Help is available on the Mattermost channel be-dep #Containers

• The support line is acc-adm-support@cern.ch for now
Thanks for your attention!
Kubernetes training

- Currently offered at CERN
- Provided by the Linux Foundation
- Multiple choices: from Application Development to Security
- For developers: CKAD is recommended
  https://www.cncf.io/certification/ckad/
- LMS https://lms.cern.ch/ekp/servlet/ekp?PX=N&TEACHREVIEW=N&CID=EKP000043693&TX=FORMAT1&LANGUAGE_TAG=en&DECORATEPAGE=N
  or look for “Kubernetes” in Catalog > Search
Plain containerization: base images

- acc_cc7
- + Oracle JDK 8
  - acc_cc7_jdk8
- + OpenJDK 11
  - acc_cc7_openjdk11

Built using

A
Container project: base images

Install base CentOS packages in subdirectory

Apply custom configuration files

Create an archive of the work directory

Convert to OCI image

- dnf -y --installroot=${WORKDIR}/rootfs --nodoc install ${PACKAGES}
- tar -C ${WORKDIR}/rootfs -cf ${WORKDIR}/rootfs.tar
- podman import - acc_cc7:latest < rootfs.tar
Plain containerization: base images
Plain containerization: active use cases

• Currently:
  • Standard way to deploy software (e.g. SourceGraph, Nexus)
  • Unified deployment and operational environments (e.g. LHC Injector Chain Timing Sequence Manager)
  • Replicate production in local environments (e.g. LHC Orbit Feedback)

• Foreseen:
  • Decoupling software upgrades from operating system upgrades (e.g. WinCC OA 3.16 on EL8 consoles)
  • Unless tied to specific hardware (e.g. timing cards), all applications can benefit from containerization