Container Orchestration, Cloud, and Petabytes of Data: The Rubin Observatory Example

Bastien Gounon
Fabio Hernandez
Sabine Elles
Dominique Boutigny
Fabrice Jammes
1. Large Synoptic Survey Telescope
2. The largest astronomical catalog
3. Cloud-Native: Kubernetes
4. Cloud-Native: Gitops & CI
5. Cloud-Native: Kubernetes Operators
6. Cloud-Native: Storage management
7. Cloud-Native: Workflows
A project that makes you dream

A revolutionary telescope
The largest digital camera in the world
The largest celestial catalogs ever made

Funding
~$1 billion, 20% dedicated to data management
Key role of CNRS/IN2P3

Objective:
Define the nature of dark energy
The largest astronomical catalog

LSST will produce a catalog of **40 billion galaxies and stars** and their associated physical properties, i.e. **500 PB** of data
Qserv The Petascale database
International context

7 engineers

- Stanford
- LPC
- Data Access and Database (DAX)

7 engineers

- Devops
- 6 engineers
- Google Cloud
- Infrastructure
- 3 engineers

- Validation/Tests
- 2 engineers
- 1 senior researcher

- Infrastructure
- 4 engineers
Qserv design

Relational database, 100% open source
Spatially-sharded with overlaps
Map/reduce-like processing, highly distributed

~1000 workers, 20 chunks/5TB per workers
Highly automated deployment

Targets:

In France
CC-IN2P3 will analyze 50% of the data stream and provide access to the entire catalog

In the US
Google hosts the Interim Data Facility

~1000 machines per database instance

Coordination of Rubin Observatory, IN2P3 and Google
Kubernetes accepted by the project and validated for 20% of the target
Cloud-Native
Kubernetes
All you really care about
Workload portability

Portability

Build your apps on-prem, lift-and-shift into cloud when you are ready

Before Kubernetes
~3 months to deploy Qserv inside a new cluster

With Kubernetes
5 minutes to 1 day
Cloud-Native
Gitops & CI
Automated deployment: Cloud Native

Qserv repository

Build/Validate/Push Qserv images

IDF repository

Create/Update Cloud Datacenter

Legend:
CI Workflow
Docker

Cloud Infrastructure:
Google Kubernetes Engine

Storage:
~ 35TB Catalogs
Google Persistent Disk
CI in practice: Qserv integration tests
CI in practice: Qserv image scanning

Vulnerability Scanning & Policy-Compliance for Containers
Gitops: CI + IaC

Delegate access to infrastructure management
Track who does what on infrastructure
Recreate infrastructure from scratch
Ease Kubernetes maintenance/upgrade

Kubernetes is fully managed by Google Cloud / GKE
In practice

Add five nodes to the GKE cluster. Kubernetes will then allow to easily scale Qserv.
Cloud-Native Kubernetes operators
How does an operator works?

Allow to deploy a complex application with only a few lines of yaml

Software Developer
Kubernetes user

K8s API

Kubernetes operator

Native Kubernetes resources

Custom resource definition

Custom Kubernetes controller

Watch Event

Reconcile

Custom resource

```yaml
apiVersion: qserv.lsst.org/v1alpha1
kind: Qserv
metadata:
  name: qserv
  namespace: database
spec:
  czar:
    image: qserv/lite-qserv:2021.10.1-rc1
    replicas: 1
    storage: 1Ti
  worker:
    image: qserv/lite-qserv:2021.10.1-rc1
    replicas: 10
...
```

Deployments
StatefulSets
Autoscalers
Secrets
Config maps
Qserv is available on operatorHub

https://operatorhub.io/operator/qserv-operator

Seamless upgrade is a work in progress
Cloud-Native Storage management
Storage management

GKE: Dynamic storage provisionning

User deploy Qserv instance

Create PVClaims

Google Storage creates automatically PersistentVolume+Google Disks (ssd+hdd)

On-premise:

Storage is manually declared to Kubernetes (via PV) and created

Easier on GKE, but better performance on-premise
Cloud-Native Workflows
A powerful data ingest workflow

Qserv has a powerful distributed ingest algorithm
Flexible but require orchestrating tasks (DAG)

Argo Workflow project help us a log

Case study 2021: **Implementation of a large-scale data loading algorithm**
*Ingestion of 15000 files and 15TB in 1h30*
### Argo: screenshots

#### Command Output

```
fjames@clinfoport18 ~ ➤ argo get @latest | tail -n 15

<table>
<thead>
<tr>
<th>STEP</th>
<th>TEMPLATE</th>
<th>PODNAME</th>
<th>DURATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>qserv-ingest-mdh4</td>
<td>main</td>
<td>qserv-ingest-mdh4-2075476264</td>
<td>3s</td>
</tr>
<tr>
<td></td>
<td>queue</td>
<td>qserv-ingest-mdh4-964548720</td>
<td>4s</td>
</tr>
<tr>
<td></td>
<td>register</td>
<td>qserv-ingest-mdh4-1041421248</td>
<td>14s</td>
</tr>
<tr>
<td></td>
<td>transactions</td>
<td>qserv-ingest-mdh4-3195504171</td>
<td>2s</td>
</tr>
<tr>
<td></td>
<td>publish</td>
<td>qserv-ingest-mdh4-4256901816</td>
<td>12s</td>
</tr>
<tr>
<td></td>
<td>index-tables</td>
<td>qserv-ingest-mdh4-1866502525</td>
<td>2s</td>
</tr>
<tr>
<td></td>
<td>validate</td>
<td>qserv-ingest-mdh4-493206715</td>
<td>2s</td>
</tr>
<tr>
<td></td>
<td>benchmark</td>
<td>qserv-ingest-mdh4-1797710727</td>
<td>5s</td>
</tr>
</tbody>
</table>
```

#### Workflow Diagram

![Workflow Diagram](image-url)
Public cloud: pros and cons

Pros
★ Flexibility (access, provisioning)
★ Excellent support
★ Low maintenance
★ Cool proprietary features

Cost-effective over time if organizations learn to use and operate it more efficiently

Cons
★ Cost difficult to understand
★ Vendor lock-in
★ Hide Kubernetes internals (black box)
★ Run slower than bare-metal (~25%)

The higher the average server load, the less attractive the cloud is financially
Qserv is going on

1. Container orchestration helps a lot
2. Commercial cloud is worth considering

Conclusion