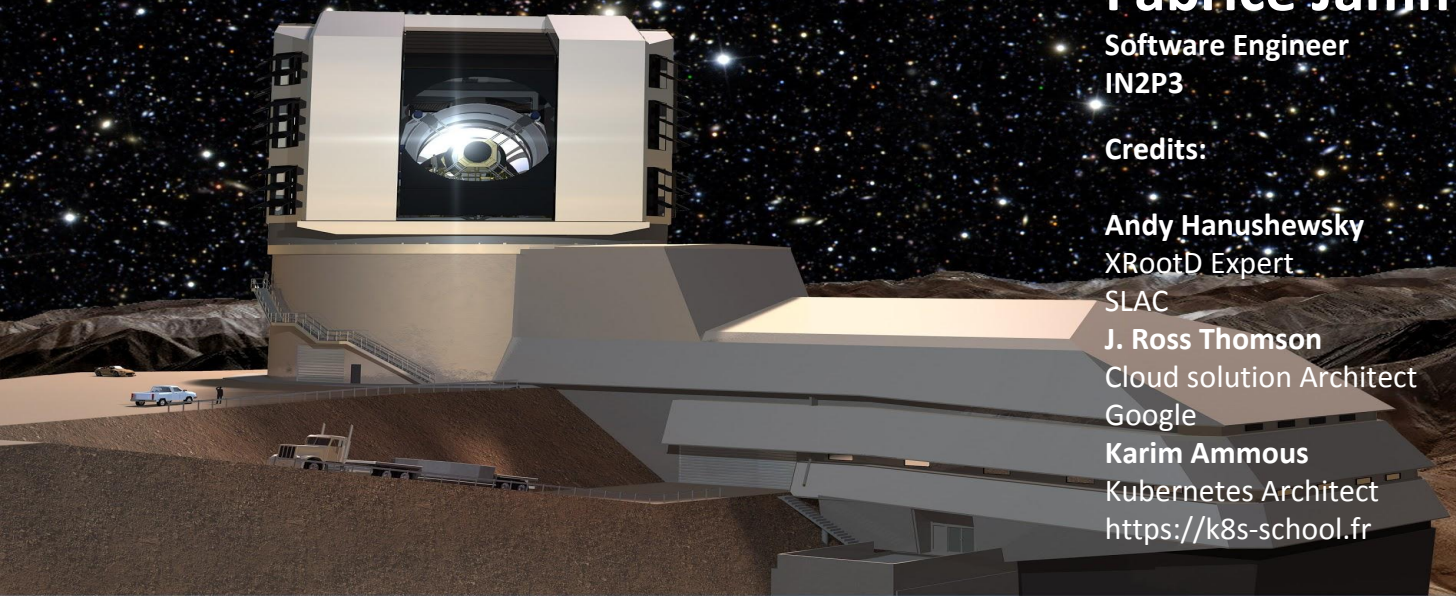
Embedding XRootD inside Qserv the cloud-native petascale database



Fabrice Jammes
Software Engineer
IN2P3

Credits:

Andy Hanushewsky
XRootD Expert
SLAC
J. Ross Thomson
Cloud solution Architect
Google
Karim Ammous
Kubernetes Architect
https://k8s-school.fr

Agenda

1. Large Synoptic Survey Telescope
2. Qserv: LSST Petascale database
3. Kubernetes
4. Benefits of Cloud-Native
5. Implementation details
6. Kubernetes: tracks for HPC

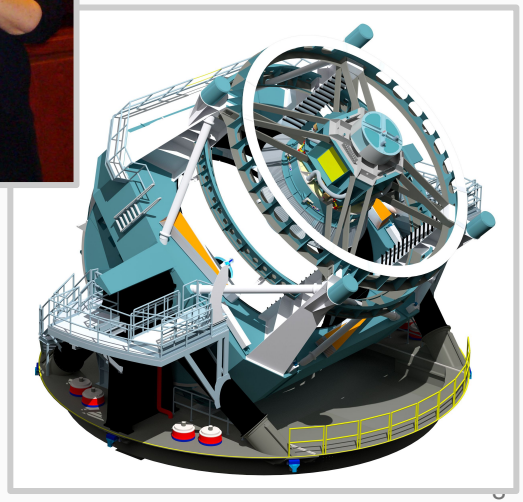
LSST in short

Large Synoptic Survey Telescope

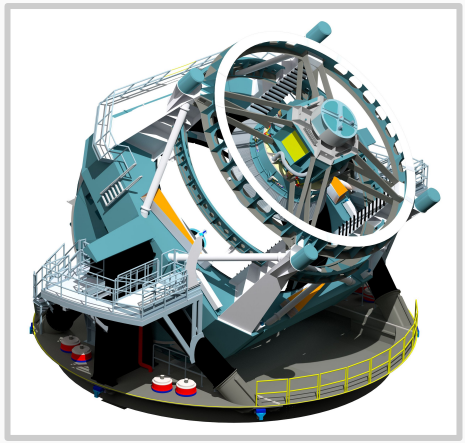
Large aperture, wide-field, ground-based survey telescope
The largest imager ever built for astronomy

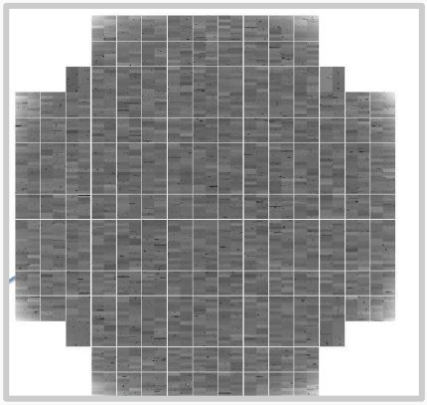
Characteristics

- ★ All visible sky in 6 bands
- ★ ~20000□
- ★ 15 seconds exposures, 1 visit/3 days
- ★ During 10 years!
- ★ **60 PB of raw data**

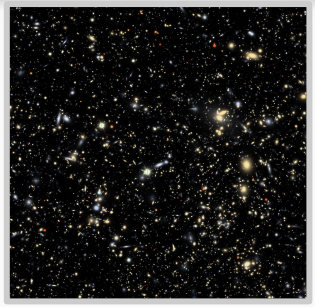


80+ PB of astronomical catalog

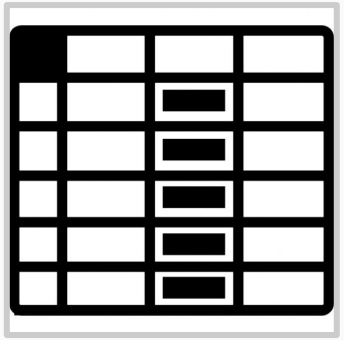




Raw data



Processed image

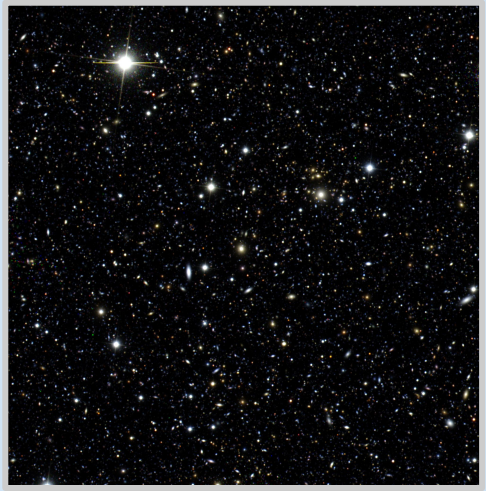


Catalog (stars, galaxies, objects, sources,transients, exposures, etc.)

LSST will build a catalog of 20 billion galaxies and 17 billion stars and their associated physical properties

Data

Images
Persisted: **~38 PB**
Temporary: **~½ EB**



★ ~3 million “visits”
★ ~47 billion“objects”
★ ~9 trillion “detections”

★ Largest table: **~5 PB**
★ Tallest table: ~50 trillion rows
★ Total (all data releases, compressed): **~83 PB**

Ad-hoc user-generated data
Rich provenance

Qserv The LSST Petascale database

Who we are

Database and Data access team

- ★ 10 engineers at Stanford University + 1 IN2P3
  - *Software development*

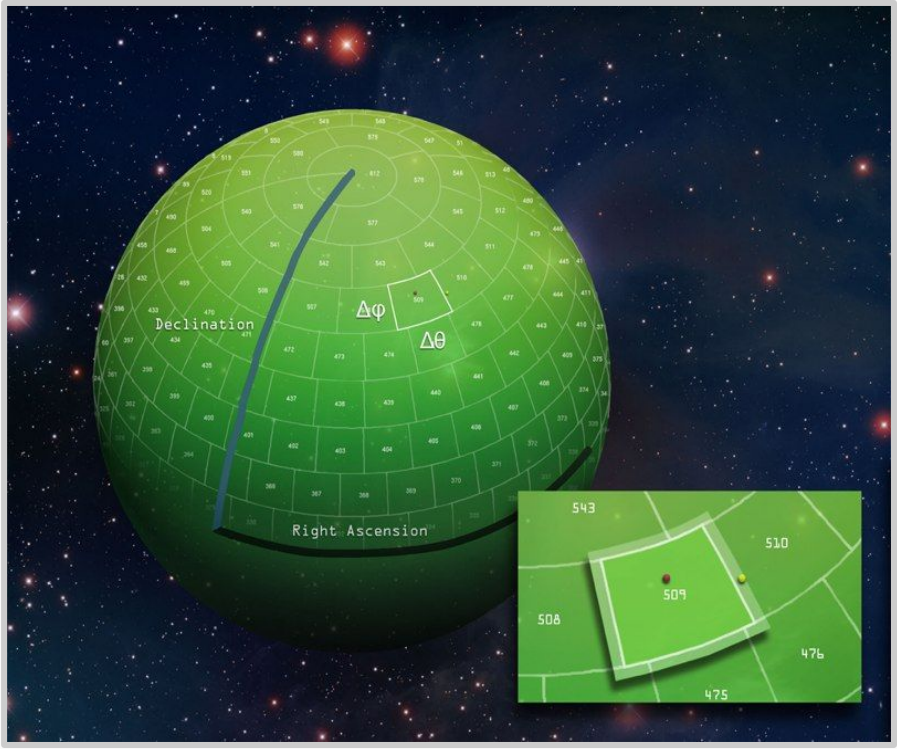
Operations teams

- ★ 5 sysadmins at NCSA/IN2P3
  - *Large Scale development platform*
  - *Prototype Data Access Center*
  - *Cloud Native / Kubernetes*
  - *System administration, Monitoring*

What we do

Data Access and **Database**

- ★ Data and metadata
- ★ Images and databases
- ★ Persisting and querying
- ★ For pipelines and users
- ★ Real time Alert Production and annual Data Release Production
- ★ For Archive Center and all Data Access Centers
- ★ For USA, France and international partners
- ★ Persisted and virtual data
- ★ **Estimating, designing, prototyping, building, and productizing**

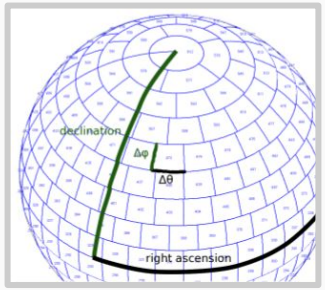


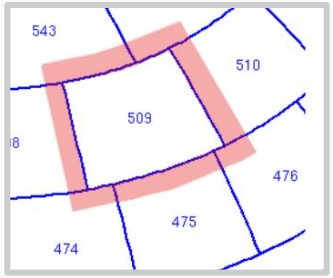
Qserv design

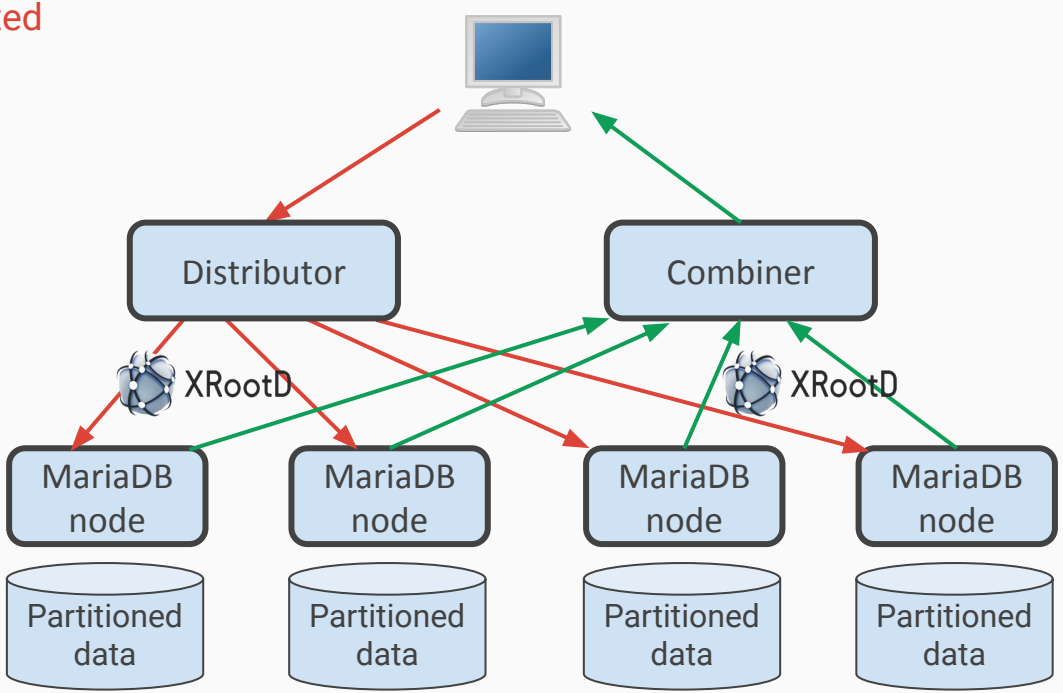
Relational database, 100% open source

Spatially-sharded with overlaps

Map/reduce-like processing, highly distributed







From Cloud-Native to Bare-Metal

Target for production

~500 nodes clusters in 2 international Academic data-centers

Running now

Development platform (CC-IN2P3)

400 cores, 800 GB memory

500 TB storage (upgrade in progress)

=> +250 TB of synthesized data

Prototype Data Access Center (NCSA)

500 cores, 4 TB memory

700 TB storage,

=> WISE catalog ("real" dataset)

EVERYONE'S EXCITED ABOUT

KUBERNETES

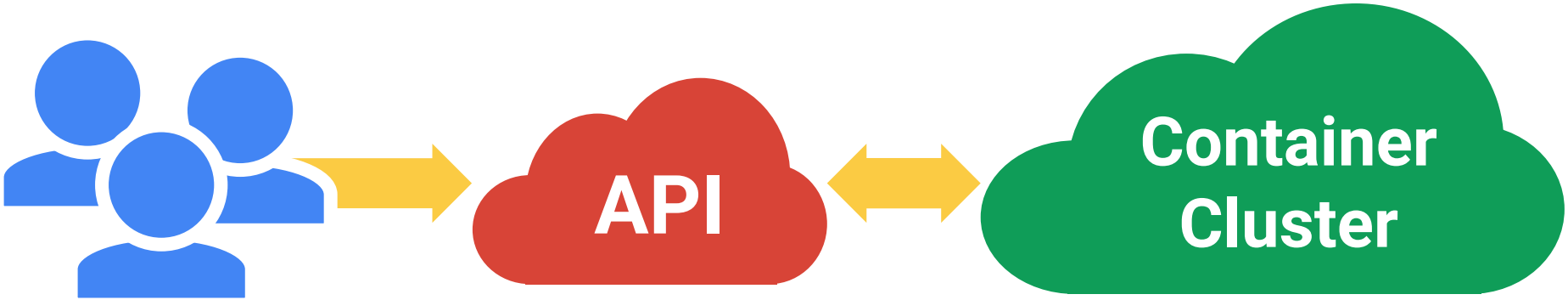
Kubernetes: a modern version of LHC grid

Greek for *“Helmsman”*; also the root of the words *“governor”* and *“cybernetic”*

- Manages container clusters
- Inspired and informed by Google’s experiences and internal systems
- Supports multiple cloud and bare-metal environments
- Supports multiple container runtimes
- **100% Open source**, written in Go

Manage applications, not machines

All you really care about



A highest velocity open-source project

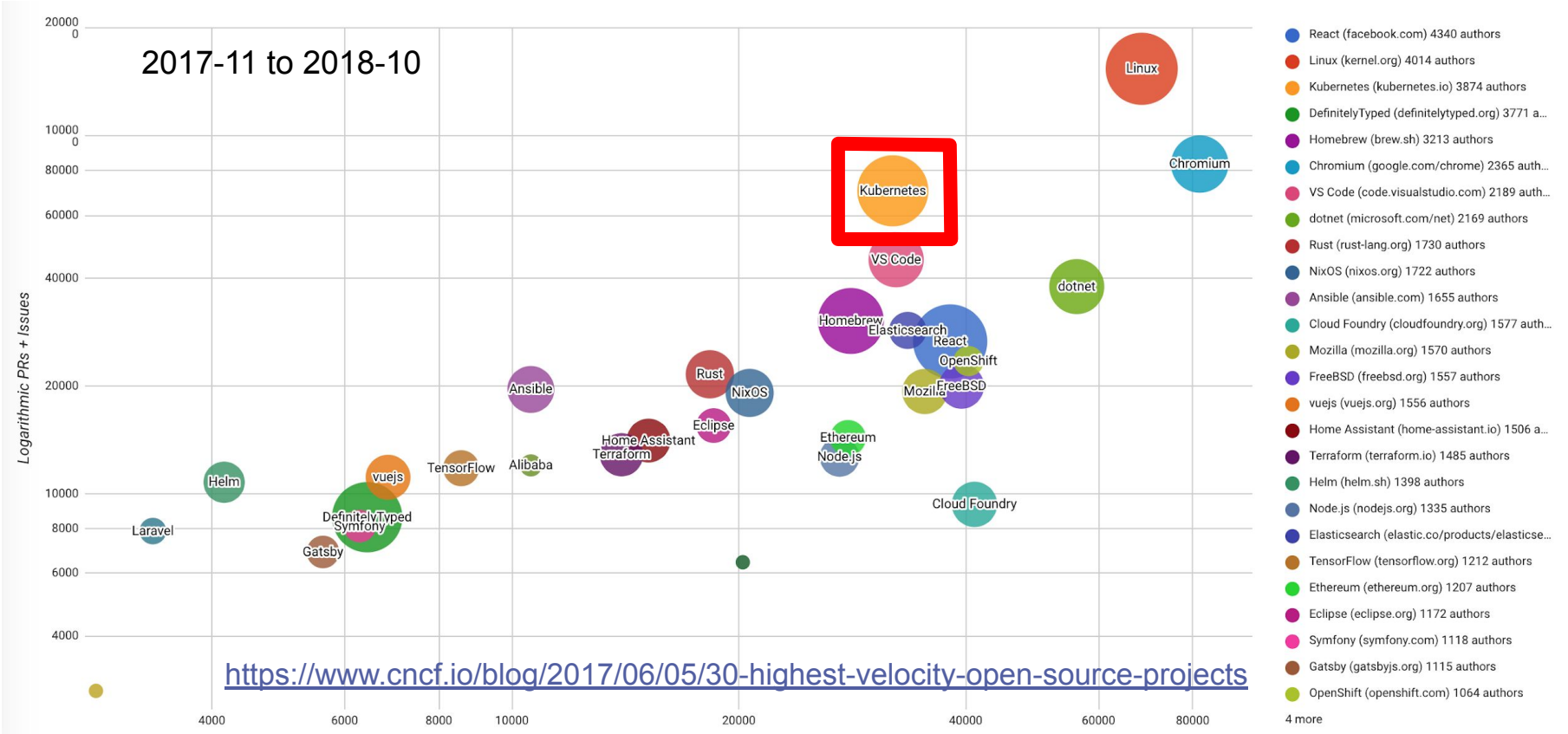
- Kubernetes 1.0 launched in **July 2015**
- New minor version every **3 months**. Currently on 1.10.
- Official **CNCF** Project

32 000+
pull requests
the latest year

60,000+
commits
the latest year

~23 PRs
merges/day
in the core repo

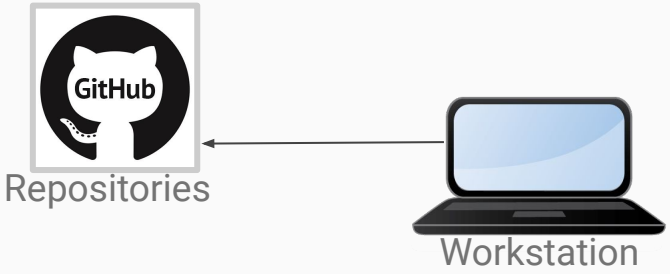
30 Highest Velocity Open Source Projects



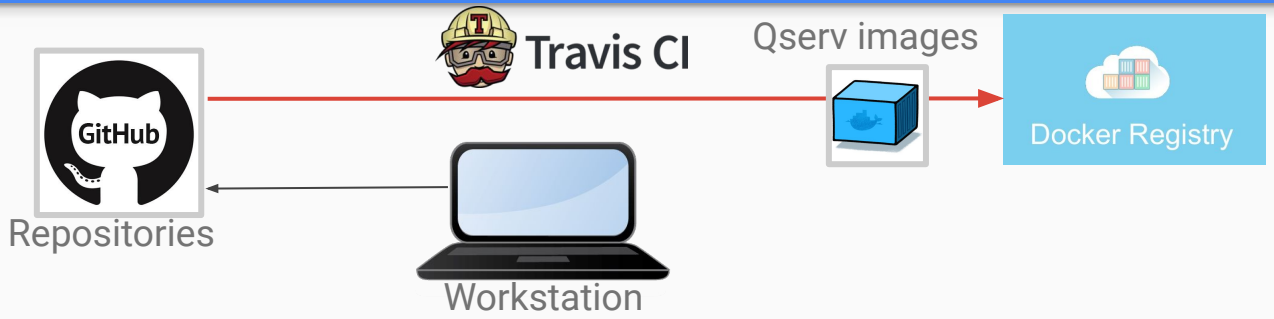
© 2018 Cloud Native Computing Foundation

Benefits of Cloud-Native

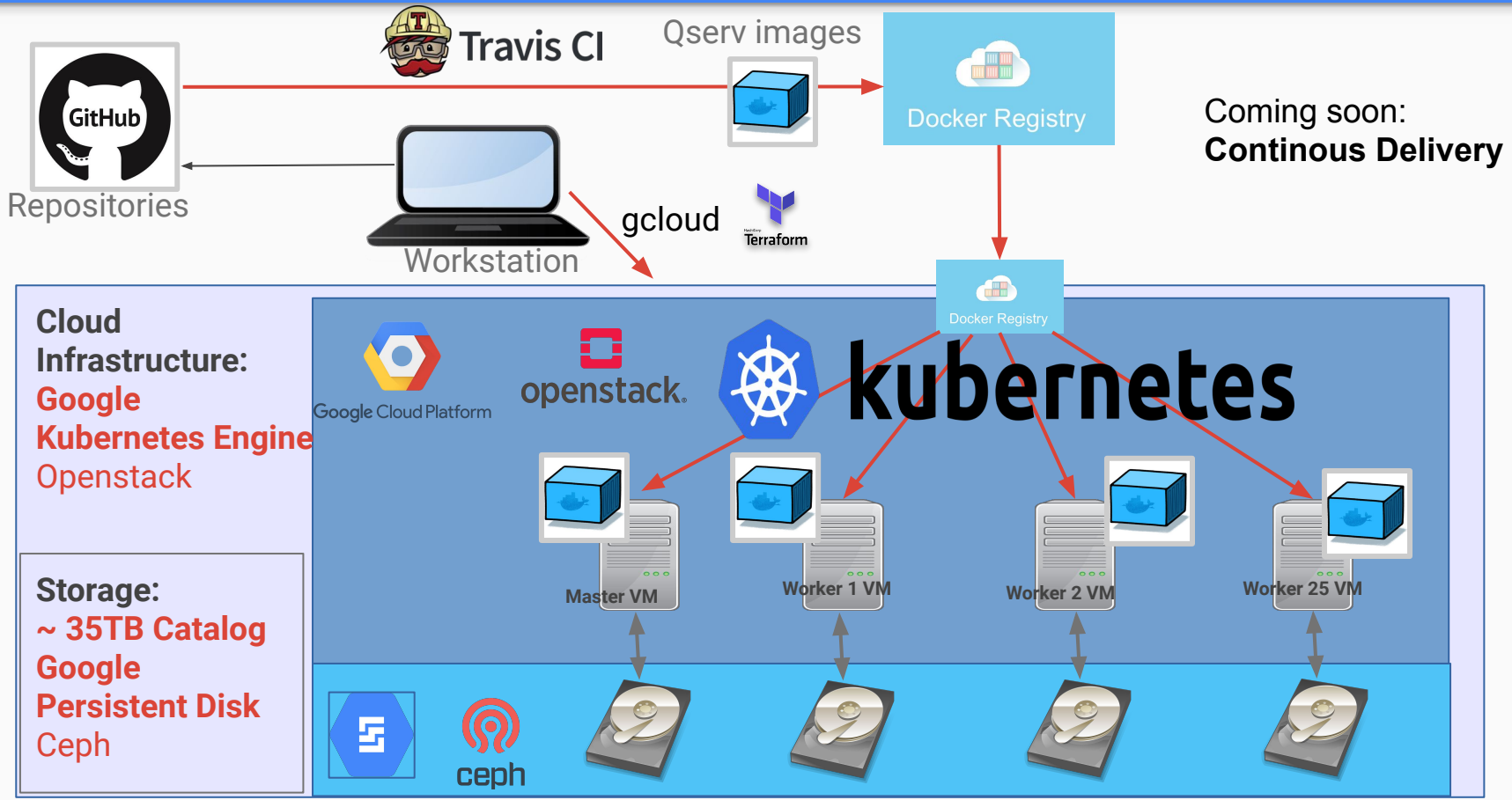
Automated Qserv deployment



Automated Qserv deployment

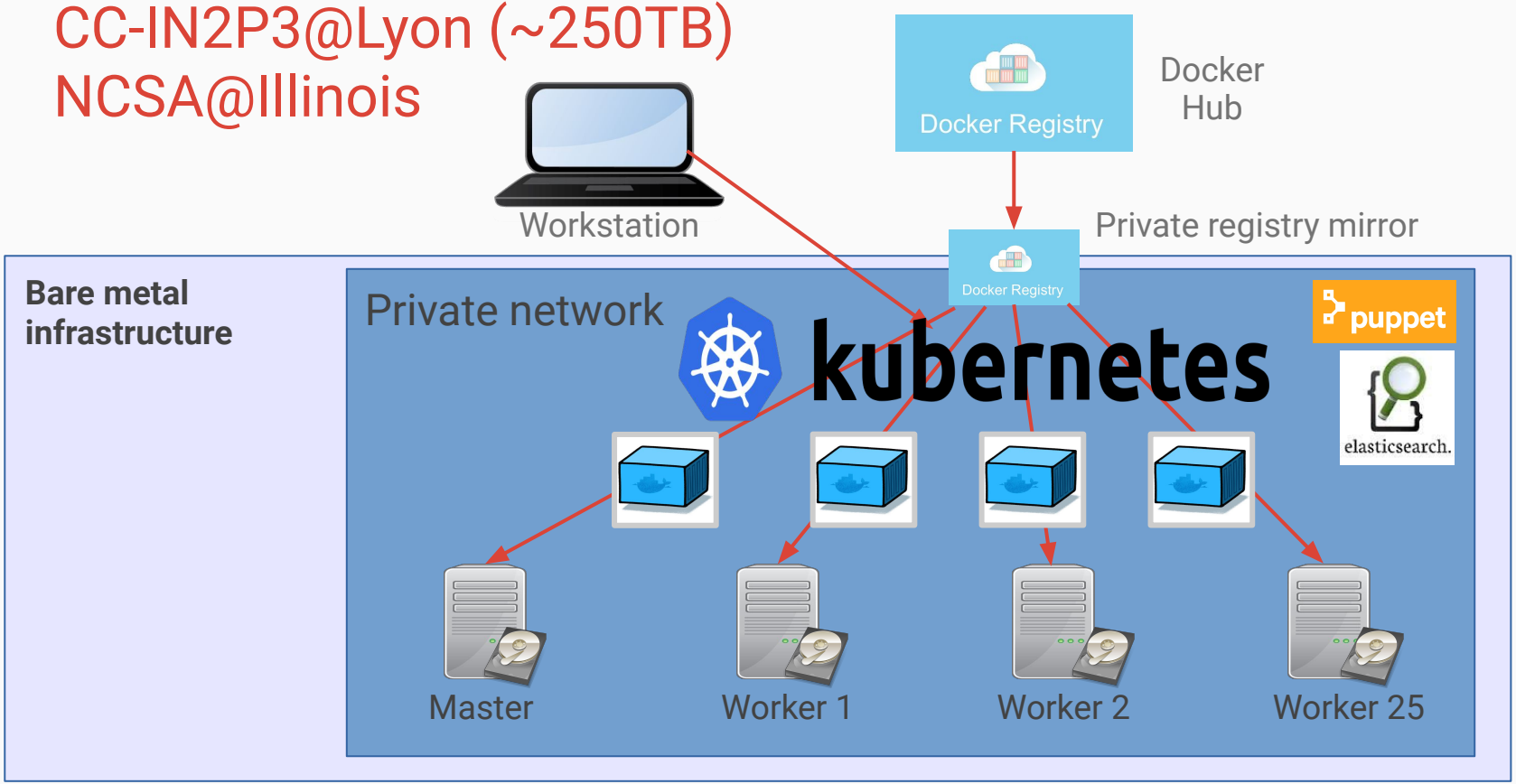


Automated deployment: Cloud Native

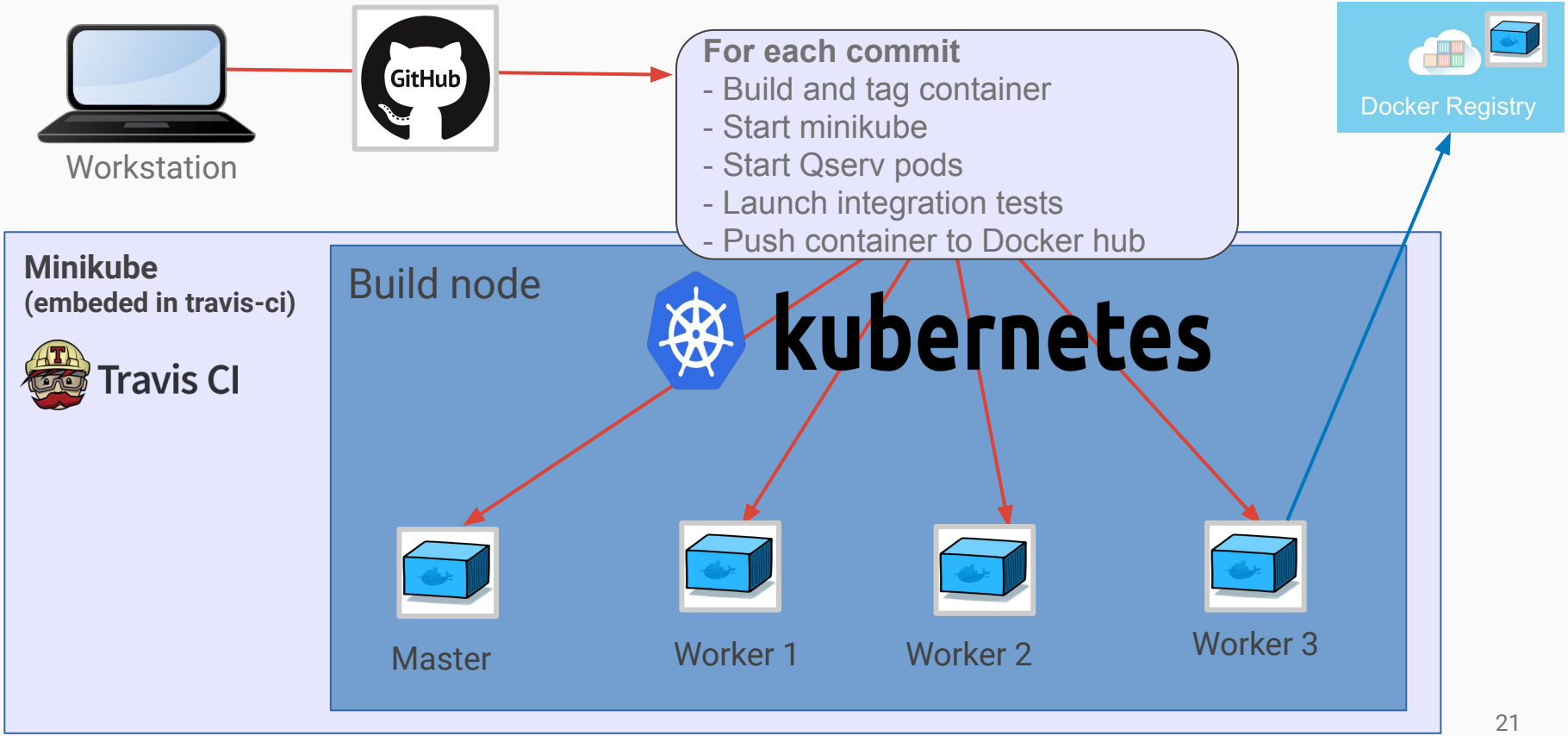


Automated deployment: bare-metal

CC-IN2P3@Lyon (~250TB)
NCSA@Illinois

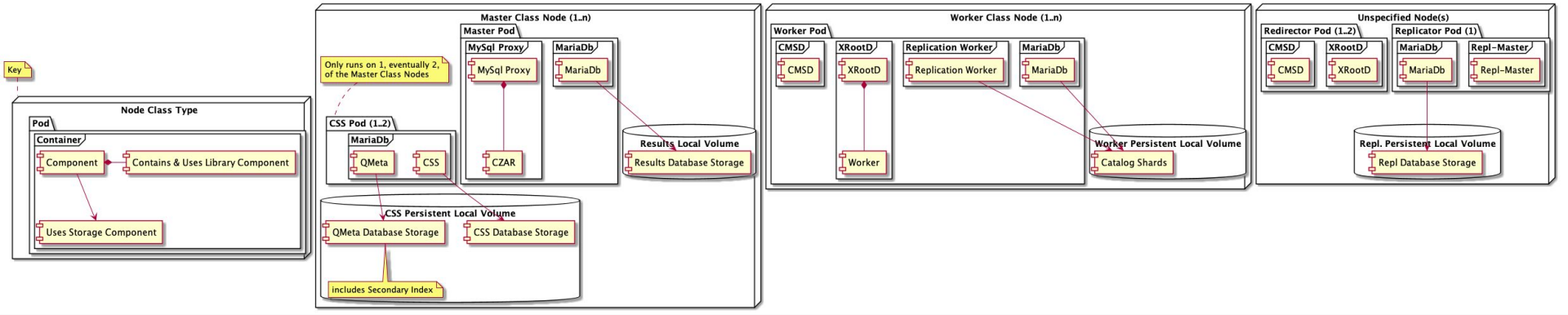


Automated deployment: CI



K8s + Microservice features

- ★ Automated scaling
- ★ Container scheduling
- ★ Auto-healing
- ★ Continuous deployment
- ★ Volume management (storage)
- ★ Easy monitoring
- ★ Healthcheck
- ★ Security

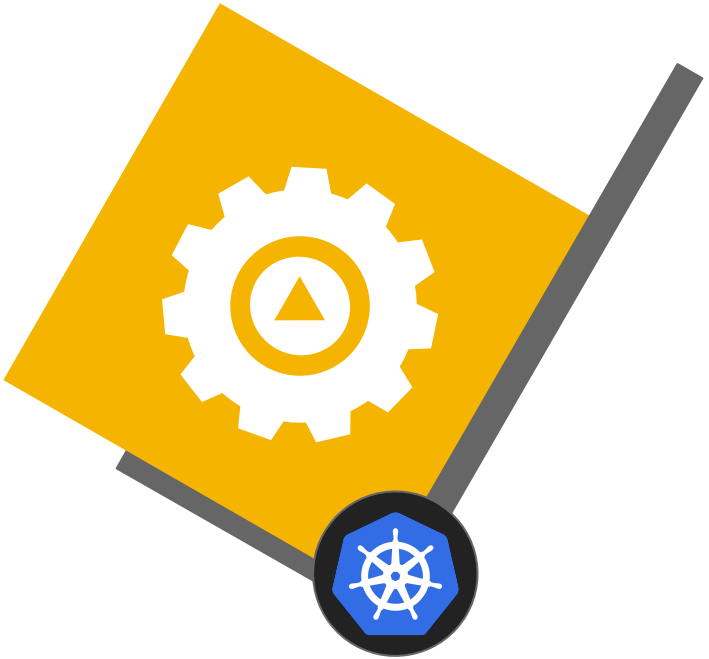


The killer feature: workload portability

Result: Portability

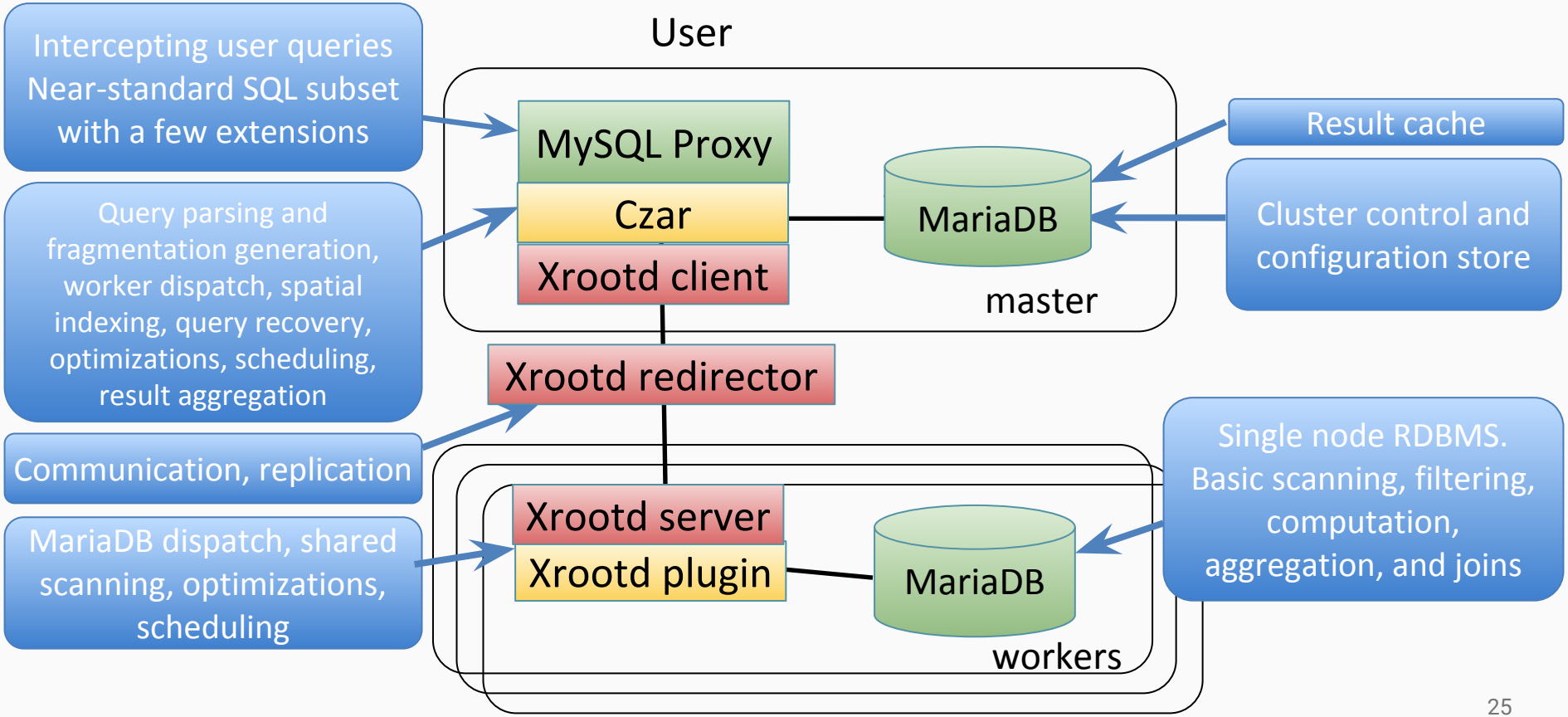
Put your app on wheels and move it whenever and wherever you need

Easily move your distributed application anywhere Kubernetes is supported, in seconds.



Implementation details

Software components



Pods

Small group of containers & volumes

Tightly coupled

The **atom** of scheduling & placement

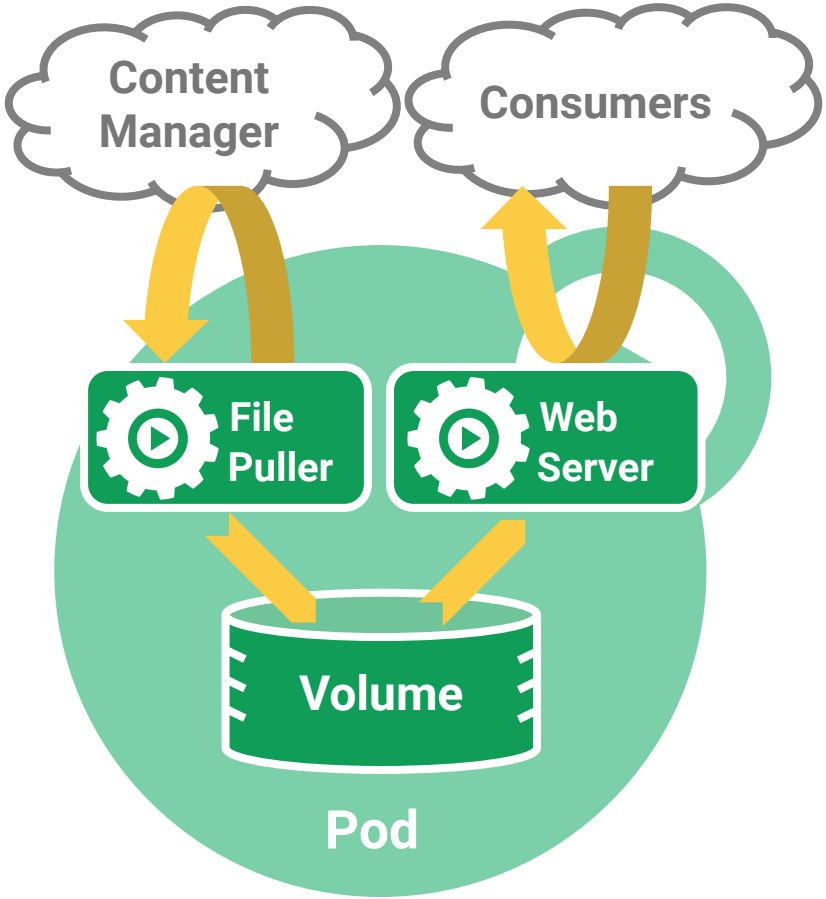
Shared namespace

- share IP address & localhost
- share IPC, etc.

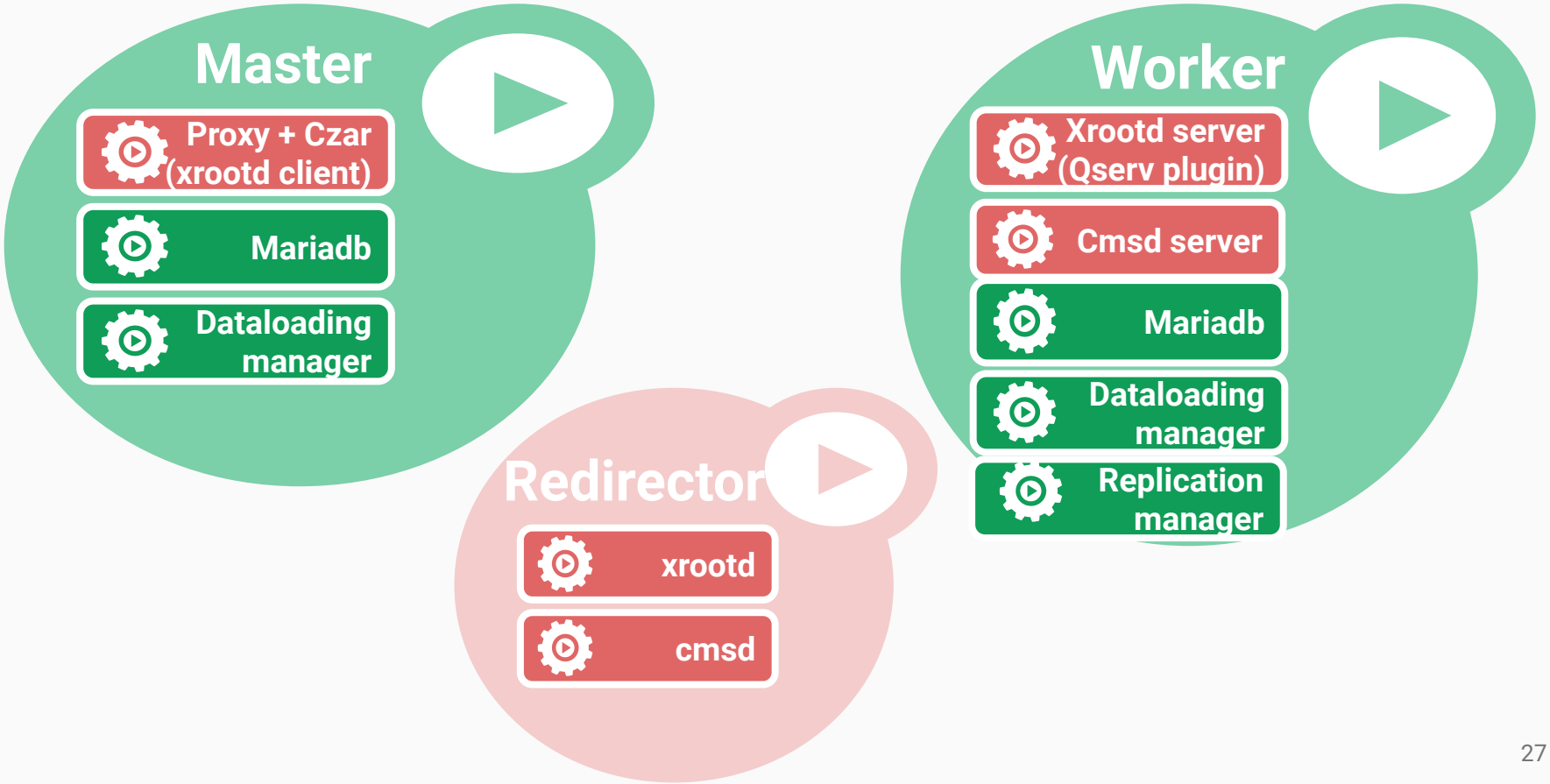
Managed lifecycle

- bound to a node, restart in place
- can die, cannot be reborn with same ID

Example: data puller & web server



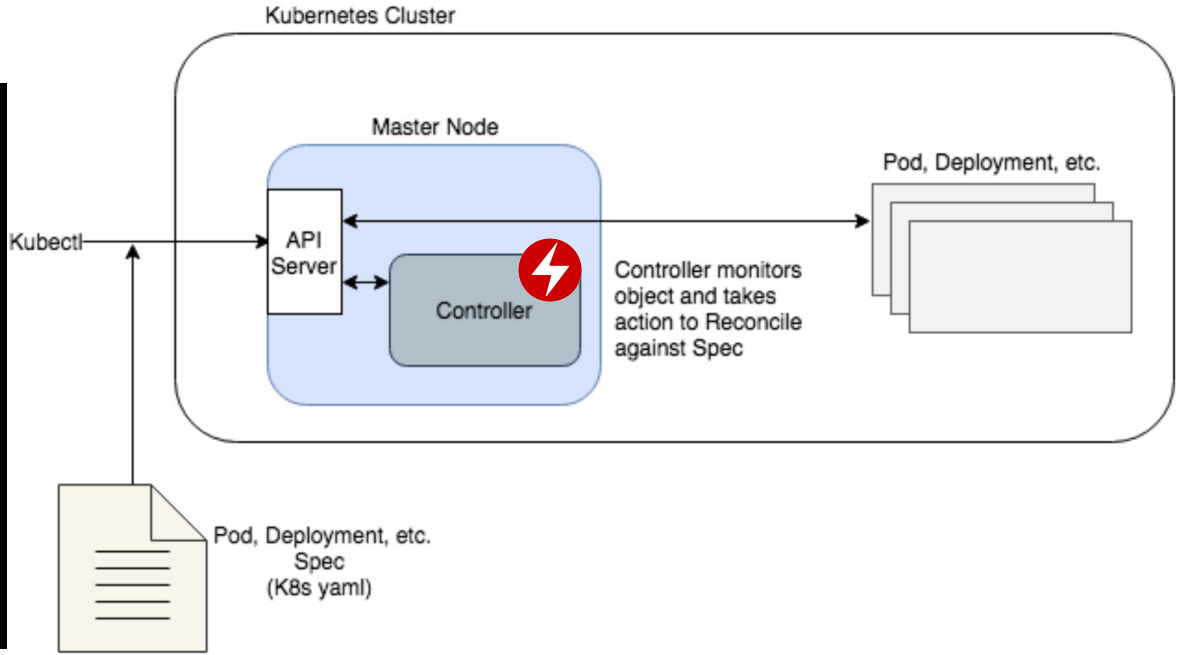
Qserv pods


Master
Proxy + Czar (xrootd client)
Mariadb
Dataloading manager
Worker
Xrootd server (Qserv plugin)
Cmsd server
Mariadb
Dataloading manager
Replication manager
Redirector
xrootd
cmsd

Status

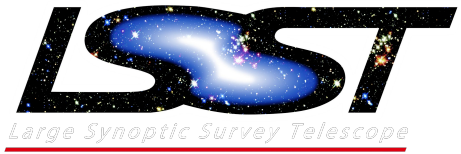
- Support Qserv custom **k8s operator** *"Operators implement and automate common Day-1 (installation, configuration, etc) and Day-2 (re-configuration, update, backup, failover, restore, etc.) activities in a piece of software running inside your Kubernetes cluster, by integrating natively with Kubernetes concepts and APIs. "*

```
apiVersion:
qserv.lsst.org/v1alpha1
kind: Qserv
metadata:
  name: example-qserv
spec:
  worker:
    replicas: 3
    image:
"qserv/qserv:9701693"
  ...
```



Status

- Push-button installation and configuration
- Support multiple xrootd redirectors
- Support Kubernetes virtual networks
- Support Readiness and Liveness probes



=> **Most advanced Kubernetes integration in the worldwide Xrootd community**

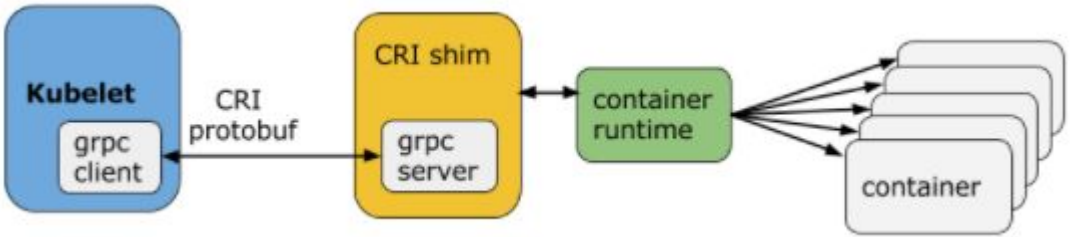
Future

- **Real scale test case in partnership with CC-IN2P3**
- Integration with Prometheus
- Add security (PodSecurityPolicy, NetworkPolicy)
- Simulate failures, w.r.t. data replication
- Setup on-premise Kubernetes for production

Tracks for HPC

Ability to change the container runtime:

- CRI : container runtime interface
- OCI : standardized image format



- HKube: High Performance Computing over Kubernetes http://hkube.io/ ?

  => Looking for a use-case/POC

Running Jobs

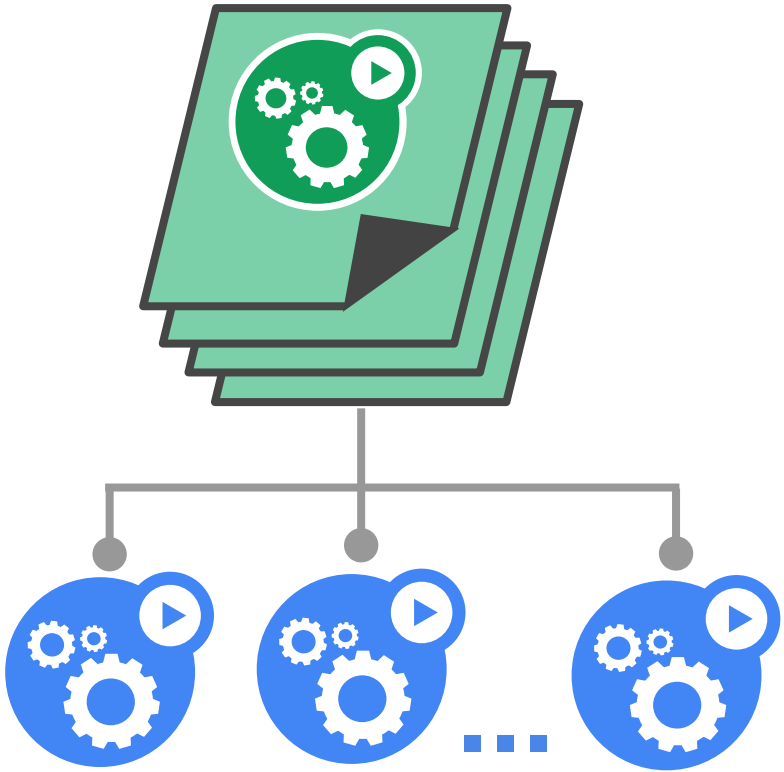
Jobs

Run-to-completion, as opposed to run-forever

- Express parallelism & required completions
- Workflow: restart on failure
- Build/test: don't restart on failure

Aggregates success/failure counts

Built for batch and big-data work



PersistentVolumes

Volumes

Pod-scoped storage

Support many types of volume plugins

- Empty dir (and tmpfs)
- Host path
- Git repository
- GCE Persistent Disk
- AWS Elastic Block Store
- Azure File Storage
- iSCSI
- NFS
- Photon
- Portworx
- Quobyte
- vSphere
- GlusterFS
- Ceph File and RBD
- Cinder
- FibreChannel
- ScaleIO
- StorageOS
- Secret, ConfigMap, DownwardAPI
- Flex (exec a binary)
- ...

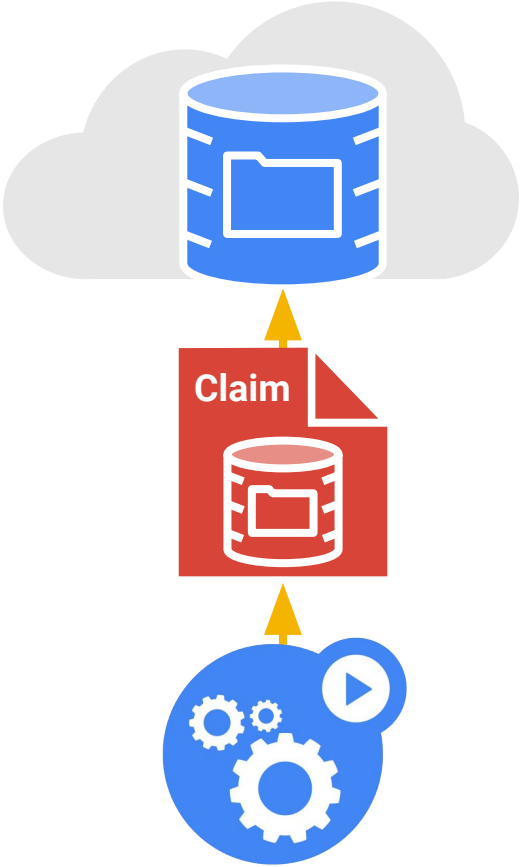
PersistentVolumes

A higher-level storage abstraction
- insulation from any one cloud environment

Admin provisions them, users claim them
- **NEW: auto-provisioning**

Independent lifetime from consumers
- lives until user is done with it
- can be handed-off between pods

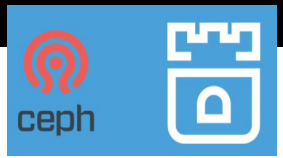
Dynamically “scheduled” and managed, like nodes and pods



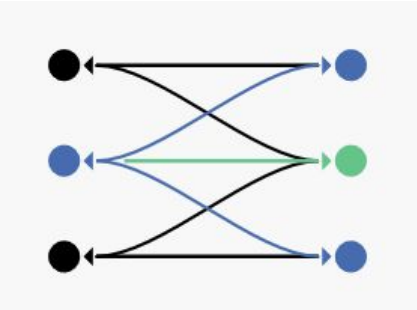
Rook: storage operator

- Cloud-Native Storage Orchestrator
- Extends Kubernetes with custom types and controllers
- Automates deployment, bootstrapping, configuration, provisioning, scaling, upgrading, migration, disaster recovery, monitoring, and resource management
- Framework for many storage providers and solutions
- Open Source (Apache 2.0)
- Hosted by the Cloud-Native Computing Foundation (CNCF)

```
 $ kubectl create -f common.yaml
namespace/rook-ceph created
$ kubectl create -f operator.yaml
deployment.apps/rook-ceph-operator created
$ kubectl create -f cluster.yaml
cephcluster.ceph.rook.io/rook-ceph created
$ kubectl -n rook-ceph get pods
NAME                  READY   STATUS    AGE
rook-ceph-agent       1/1     Running   4m15s
rook-ceph-mgr-a       1/1     Running   61s
rook-ceph-mon-a       1/1     Running   92s
rook-ceph-mon-b       1/1     Running   84s
rook-ceph-mon-c       1/1     Running   71s
rook-ceph-osd-0       1/1     Running   21s
rook-ceph-operator    1/1     Running   4m17s
rook-discover         1/1     Running   4m15s
```



Istio : Service Mesh

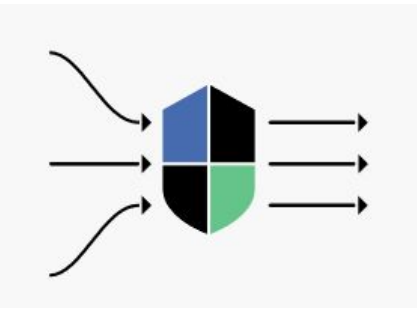


Connect

Istio can intelligently control the flow of traffic between services, conduct a range of tests and upgrade gradually with blue/green deployments.

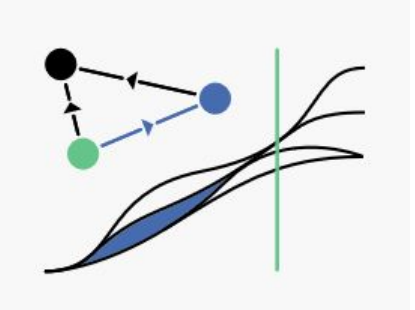
Secure

Automatically secure your services through managed authentication, authorization, and encryption of communication between services.



Control

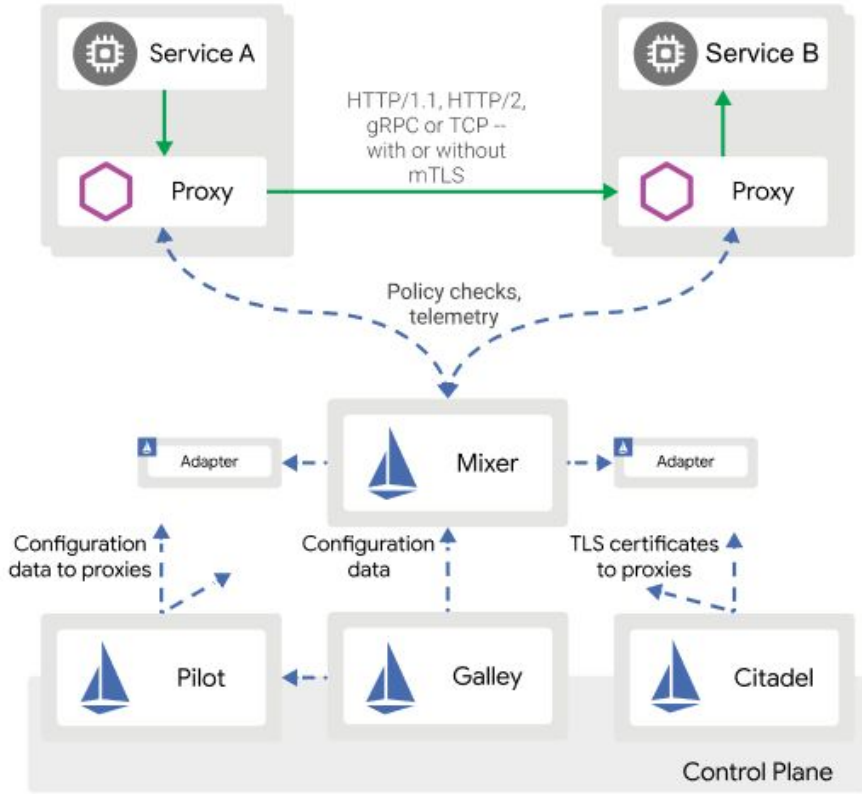
Apply policies and ensure that they are enforced and that resources are fairly distributed among consumers.



Observe

See what's happening with rich automatic tracing, monitoring, logging of all your services.

Istio: Architecture



Envoy: Sidecar Network proxy to intercept communication and apply policies.

Pilot: Control plane to configure and push service communication policies.

Mixer: Provides telemetry collection as well as sophisticated policy checks.

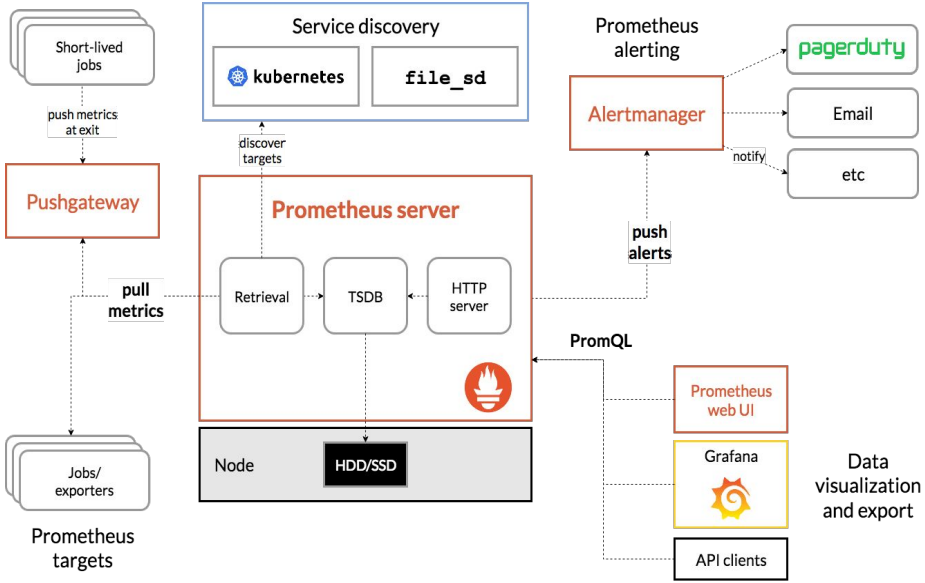
Citadel: Service-to-service auth[n,z] using mutual TLS, with built-in identity and credential management.

Galley: Configuration validation, distribution

Prometheus: monitoring

Prometheus is an open-source systems monitoring and alerting toolkit

- a multi-dimensional data model with time series data identified by metric name and key/value pairs
- PromQL, a flexible query language
- no reliance on distributed storage; single server nodes are autonomous
- time series collection happens via a pull model over HTTP
- targets are discovered via service discovery or static configuration
- multiple modes of graphing and dashboarding support



Thanks!

Contact:

Fabrice JAMMES
LPC
Clermont-Ferrand

fabrice.jammes@in2p3.fr