

# Harbor Registry at CERN

## Status and Enhancements

CVMFS Workshop, Amsterdam

<https://indico.cern.ch/event/107949>

Ricardo Rocha, CERN

# Quick Overview

<https://registry.cern.ch>

<https://kubernetes.docs.cern.ch/docs/registry/quickstart/>

Harbor is a **CNCF Graduated Project**

Backed by Distribution (the Docker Registry) but with a lot of added things

Project Quotas

OCI Artifact Support (Images, Helm Charts, ML Models, ...)

Vulnerability Scanning (**Trivy**, Clair, Sysdig, ...)

Artifact signing (Notary, sigstore)

Proxy Caches, Automated Replication

Non Blocking Garbage Collection

Tag Immutability, Retention Policies

Projects		Repositories		Storage used
Private	52	Private	126	2.14 TiB
Public	107	Public	1053	
Total	159	Total	1179	

Doubling every 2 months...

## Graduation Stage

To graduate from sandbox or incubating status, or for a new project to join as a graduated project, a project must meet the incubating stage criteria plus:

- Have committers from at least two organizations.
- Have achieved and maintained a Core Infrastructure Initiative [Best Practices Badge](#).
- Have completed an independent and third party security audit with results published of similar scope and quality as the following example (including critical vulnerabilities addressed): <https://github.com/envoyproxy/envoy#security-audit> and all critical vulnerabilities need to be addressed before graduation.
- Explicitly define a project governance and committer process. The committer process should cover the full committer lifecycle including onboarding and offboarding or emeritus criteria. This preferably is laid out in a GOVERNANCE.md file and references an OWNERS.md file showing the current and emeritus committers.
- Explicitly define the criteria, process and offboarding or emeritus conditions for project maintainers; or those who may interact with the CNCF on behalf of the project. The list of maintainers should be preferably be stored in a MAINTAINERS.md file and audited at a minimum of an annual cadence.
- Have a public list of project adopters for at least the primary repo (e.g., ADOPTERS.md or logos on the project website). For a specification, have a list of adopters for the implementation(s) of the spec.
- Receive a supermajority vote from the TOC to move to graduation stage. Projects can attempt to move directly from sandbox to graduation, if they can demonstrate sufficient maturity. Projects can remain in an incubating state indefinitely, but they are normally expected to graduate within two years.

# Singularity

<https://kubernetes.docs.cern.ch/docs/registry/quickstart/#singularity--apptainer>

Support for storing and retrieving singularity images

Version > 3.8.4 which is now ~1 year old

```
$ singularity pull docker://alpine
```

```
$ singularity remote login --username dsouthwi --password <harbor token> oras://registry.cern.ch
```

```
INFO: Token stored in ~/.singularity/remote.yaml
```

```
$ singularity push alpine_latest.sif oras://registry.cern.ch/yourproject/alpine:latest
```

```
INFO: Upload Complete
```

# Multi-Architecture

<https://kubernetes.docs.cern.ch/docs/registry/quickstart/#multi-arch>

Built-in registry support for multiple architectures

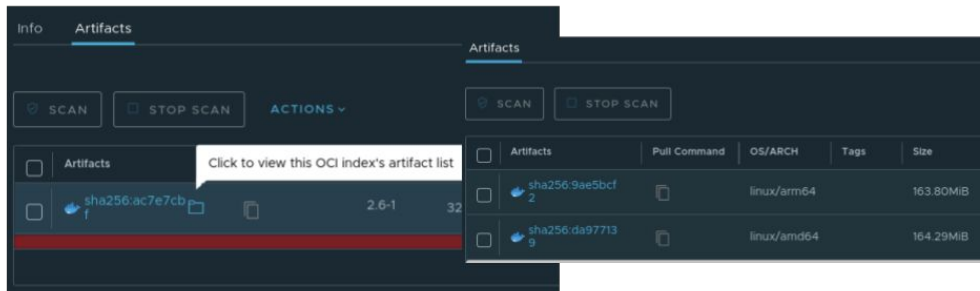
Client support in docker buildx

```
docker buildx build
```

```
--push
```

```
--platform linux/arm64/v8,linux/amd64
```

```
--tag registry.cern.ch/MYREPO/MYIMAGE:MYTAG
```



GitLab runners for both buildx *platform* and native ARM builds

```
build_container_arm:
```

```
...
```

```
tags:
```

```
- docker-arm
```

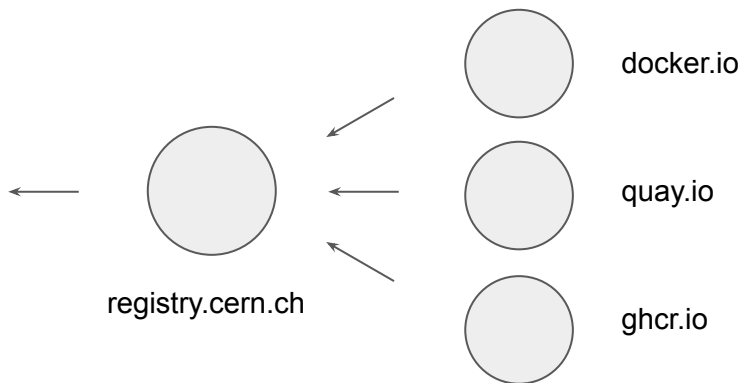
# Proxy Caches

Pull-through cache for other registries

Already enabled a docker.io pull-through cache to cover for recent API restrictions

<https://kubernetes.docs.cern.ch/docs/registry/quickstart/#pull-through-caches>

Optimized access, also helps enforcing CVE/Vulnerability checks



# Accelerated Images, Lazy Pulling (estargz)

( see Kohei's talk )

Different options for image build automation: server side, **client side**

Server side possible with <https://github.com/goharbor/acceleration-service>

Initial attempt: WebHook config required, Overriding Tags?, Load concerns

Currently **delegating task to clients** (see later for CI/CD integration)

```
docker buildx build -t registry.cern.ch/rbritoda/hello:esgz  
-o type=registry,oci-mediatypes=true,compression=estargz,forcecompression=true .
```

**(e)stargz support available** in CERN Kubernetes clusters

# Accelerated Images, Lazy Pulling (estargz)

( see Kohei's talk )

Different options for image build automation: server side, **client side**

Server side possible with <https://github.com/estargz/estargz>

Initial attempt: WebHook conf

Currently **delegating task to client**

```
docker buildx build -t registry.c
```

```
-o type=registry,oci-mediaty
```

**(e)stargz support available in CE**

## Runtime statistics



Virtual  
North America 2020

Exec time, RAM, Network ingress

atlas/athena:21.0.15\_100.0.2

17.2GB / 5.43GB

strigazi/athena:21.0.15\_100.0.2-esgz-bash-version

17.2GB 5.56GB



mode	pulling time	RAM Containerd/ stanpshotter	Ingress on node	execution time workload
native	3m37s	257MB	5.84GB	7m15s
esgz	16s	1360MB	0.84GB	8m14s

- Fast startup time
- low network traffic (workload dependent)
- Memory consumption to be investigated
- 45m to convert to esgz



# Accelerated Images, Lazy Pulling (estargz)

## Container SW Distribution

16 May 2019, 12:00 → 17 May 2019, 23:10 Europe/Zurich

513/R-070 - Openlab Space (CERN)

Ricardo Brito Da Rocha (CERN)

Registration You are registered for this event. [Check details](#)

THURSDAY, 16 MAY

14:00 → 15:00 SM18 Visit 1h 513/R-070 - Openlab Space

FRIDAY, 17 MAY

09:00 → 09:25 CVMFS and Software Distribution on the Grid 25m 513/R-070 - Openlab Space

Speaker: Jakob Blomer (CERN)

cvmfs-containers.pdf

09:25 → 09:50 WLCG and the case for userspace containers 25m 513/R-070 - Openlab Space

Speakers: Alessandra Forti (University of Manchester (GB)), Lukas Alexander Heinrich (CERN)

ContainerSWDistr.pdf

09:50 → 10:15 Overview of Containerd 25m 513/R-070 - Openlab Space

Speaker: Phil Estes (IBM)

FOSDEM 2019\_con...

5 Rootless containers 513/R-070 - Openlab Space

Speaker: Akihiro Suda (NTT)

[CERN] Rootless Co...

6 Containerd Remote Snapshotter 513/R-070 - Openlab Space

7 Discussion: Containerd Remote Snapshotter 513/R-070 - Openlab Space

CONTAINERS & KUBERNETES

## Introducing GKE image streaming for fast application startup and autoscaling



William Dennis  
Product Manager, GKE  
Autopilot, Google Kubernetes  
Engine

November 3, 2021

We're excited to announce the general availability of a new feature in [Google Kubernetes Engine \(GKE\)](#): image streaming. This revolutionary GKE feature has the potential to drastically improve your application scale-up time, allowing you to respond to increased user demand more rapidly, and save money by provisioning less spare capacity. We achieve this by reducing the image pull-time for your container from several minutes (in the case of large images) to a couple of seconds (irrespective of container size), and allowing your application to start booting immediately while GKE

# Security Features

Image signing, vulnerability scans, preventing vulnerable pulls, CVE allowlists

Deployment security

☒ Cosign  
Allow only verified images to be deployed.

☒ Prevent vulnerable images from running.  
Prevent images with vulnerability severity of Critical ▾ and above from being deployed.

Vulnerability scanning

☒ Automatically scan images on push  
Automatically scan images when they are pushed to the project registry.

CVE allowlist

Project allowlist allows vulnerabilities in this list to be ignored in this project when pushing and pulling images.

You can either use the default allowlist configured at the system level or click on 'Project allowlist' to create a new allowlist

Add individual CVE IDs before clicking 'COPY FROM SYSTEM' to add system allowlist as well.

☒ System allowlist ☐ Project allowlist

# Security Features

**Image signing**, vulnerability scans, preventing vulnerable pulls, CVE allowlists

Based on *cosign*, part of the sigstore project <https://www.sigstore.dev/>

Support available since Harbor v2.5

```
$ docker push registry.cern.ch/rocha/myimage:mytag
```

```
$ cosign sign --key mycosign.key registry.cern.ch/rocha/myimage:mytag
```

```
Enter password for private key:
```

```
Pushing signature to: registry.cern.ch/rocha/myimage:mytag
```

# Security Features

Image signing, **vulnerability scans**, preventing vulnerable pulls, **CVE allowlists**

Relying on Trivy as vulnerability scanner, input from multiple CVE databases

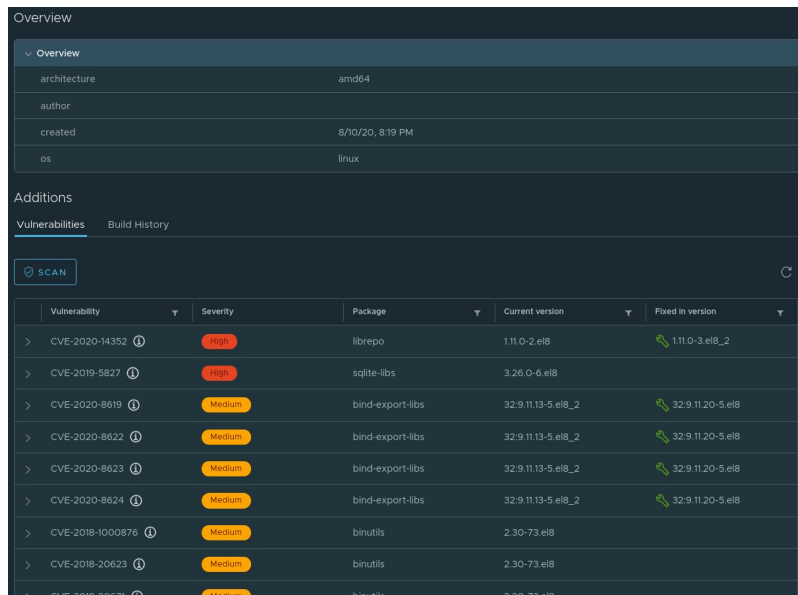
Scan on push, *rescan weekly*

Curated CVE allowlists managed centrally

*Global allowlist* applied to all projects

*Per project allowlist* managed via MRs

Review, Approval shared with security

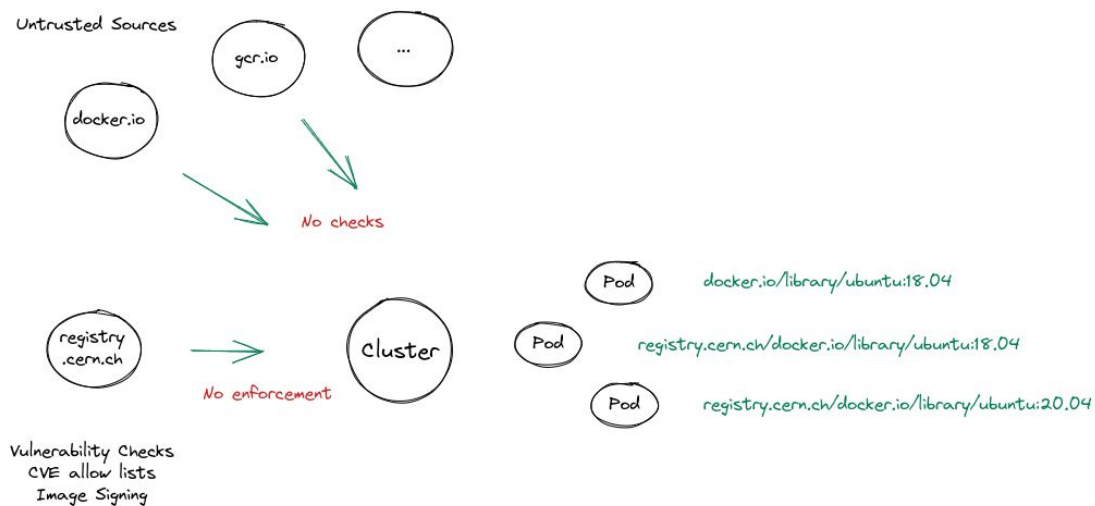


The screenshot shows a web interface for vulnerability management. It has a dark theme. At the top, there's a tab labeled 'Overview'. Below it, a table shows metadata: architecture (amd64), author, created (8/10/20, 8:19 PM), and os (linux). Below that is a section 'Additions' with two tabs: 'Vulnerabilities' (selected) and 'Build History'. Under 'Vulnerabilities', there's a 'SCAN' button. The main part of the interface is a table of vulnerabilities.

Vulnerability	Severity	Package	Current version	Fixed in version
> CVE-2020-14352 ⓘ	High	librepo	1.11.0-2.el8	1.11.0-3.el8_2
> CVE-2019-5827 ⓘ	High	sqlite-libs	3.26.0-6.el8	
> CVE-2020-8619 ⓘ	Medium	bind-export-libs	32.9.11.13-5.el8_2	32.9.11.20-5.el8
> CVE-2020-8622 ⓘ	Medium	bind-export-libs	32.9.11.13-5.el8_2	32.9.11.20-5.el8
> CVE-2020-8623 ⓘ	Medium	bind-export-libs	32.9.11.13-5.el8_2	32.9.11.20-5.el8
> CVE-2020-8624 ⓘ	Medium	bind-export-libs	32.9.11.13-5.el8_2	32.9.11.20-5.el8
> CVE-2018-1000876 ⓘ	Medium	binutils	2.30-73.el8	
> CVE-2018-20623 ⓘ	Medium	binutils	2.30-73.el8	
> CVE-2018-20631 ⓘ	Medium	binutils	2.30-73.el8	

# Security Features

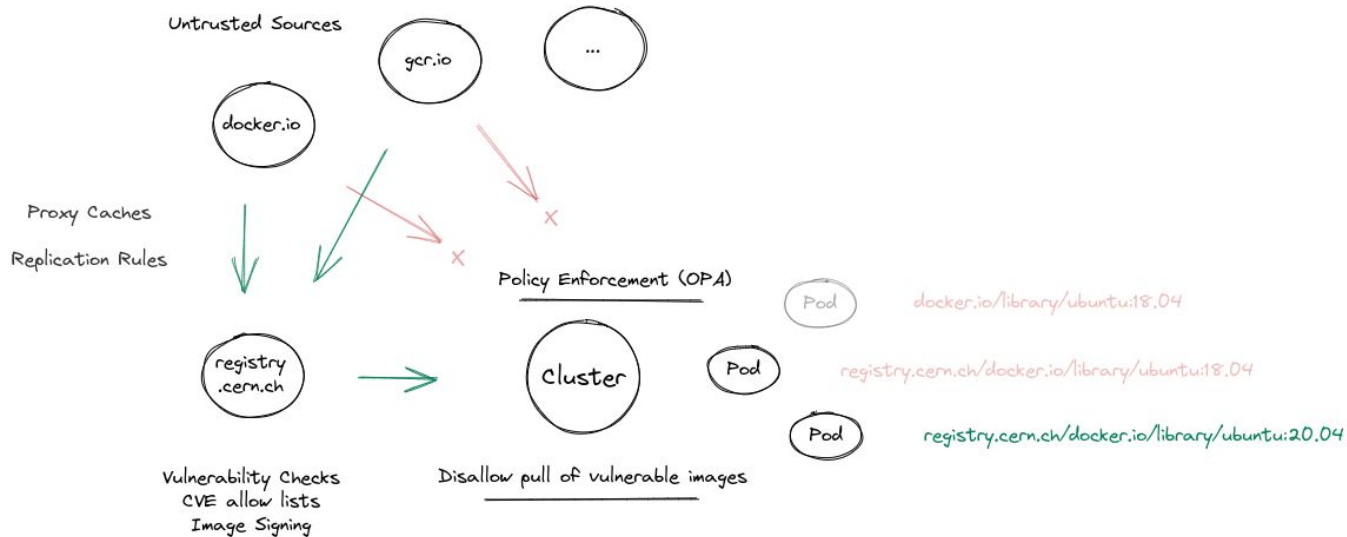
## OPA Policies, Falco Runtime Checks



# Security Features

## OPA Policies, Falco Runtime Checks

Enforce pulls from central registry.cern.ch



# Security Features

## OPA Policies, Falco Runtime Checks

```
- macro: container
  condition: container.id != host
- macro: spawned_process
  condition: evt.type = execve and evt.dir=<
- rule: run_shell_in_container
  desc: a shell was spawned by a non-shell program in a container. Container entrypoints
are excluded.

  condition: container and proc.name = bash and spawned_process and proc.pname exists and
not proc.pname in (bash, docker)

  output: "Shell spawned in a container other than entrypoint (user=%user.name
container_id=%container.id container_name=%container.name shell=%proc.name
parent=%proc.pname cmdline=%proc.cmdline)"

  priority: WARNING
```

# Integration Points: GitLab CI/CD

Built-in jobs to build **images**, helm charts

<https://kubernetes.docs.cern.ch/docs/registry/gitlab/>

## kaniko

```
include:
- project : 'ci-tools/container-image-ci-templates'
  file : 'kaniko-image.gitlab-ci.yml'
  ref: master
```

```
include:
- project : 'ci-tools/container-image-ci-templates'
  file : 'docker-image.gitlab-ci.yml'
  ref: master
```

## docker-in-docker

- **REGISTRY\_IMAGE\_PATH: Mandatory field.** The path/to/image:tag to use. ex: "registry.cern.ch/MYPROJECT/MYIMAGE:MYTAG".
- **CONTEXT\_DIR:** Default value is set to an empty string "". The build context is set to the base folder of the repository. It represents a directory containing a Dockerfile which kaniko will use to build the image. If the build context for kaniko is `folder_name/subfolder_name/`, then set the variable `CONTEXT_DIR` as `folder_name/subfolder_name`.
- **DOCKER\_FILE\_NAME:** Default value is set to `Dockerfile`. It stores the name of the Dockerfile to be built. `DOCKER_FILE_NAME` variable is relative to `CONTEXT DIR` variable. (i.e. if the `CONTEXT_DIR` variable is `folder_name` and the `DOCKER_FILE_NAME` variable is `dockerfile_name`, then kaniko will build the image using the docker file `folder_name/dockerfile_name` in the repository)
- **PUSH\_IMAGE:** Default value is `"false"`. Select if the gitlab step should push the built image.
- **ACCELERATED\_IMAGE:** Create estargz accelerated image. Defaults to `"false"`.
- **PLATFORMS:** Only available for docker builder. The platforms to which create the target container. See [buildx reference](#) for usage information.



# Integration Points: GitLab CI/CD

Built-in jobs to build images, **helm charts**

<https://kubernetes.docs.cern.ch/docs/registry/gitlab/>

```
include:
- project : 'ci-tools/container-image-ci-templates'
  file : 'helm.gitlab-ci.yml'
  ref: master
```

- **REGISTRY\_PATH:** The harbor chartrepo project name. ex: "registry.cern.ch/chartrepo/MYPROJECT".
- **REGISTRY\_SIGNKEY:** (optional) The key with which to sign the helm chart blobs.
- **PUSH\_CHART:** Default value is "false". Select if the gitlab step should push the packaged chart.

# Integration Points: WebHooks

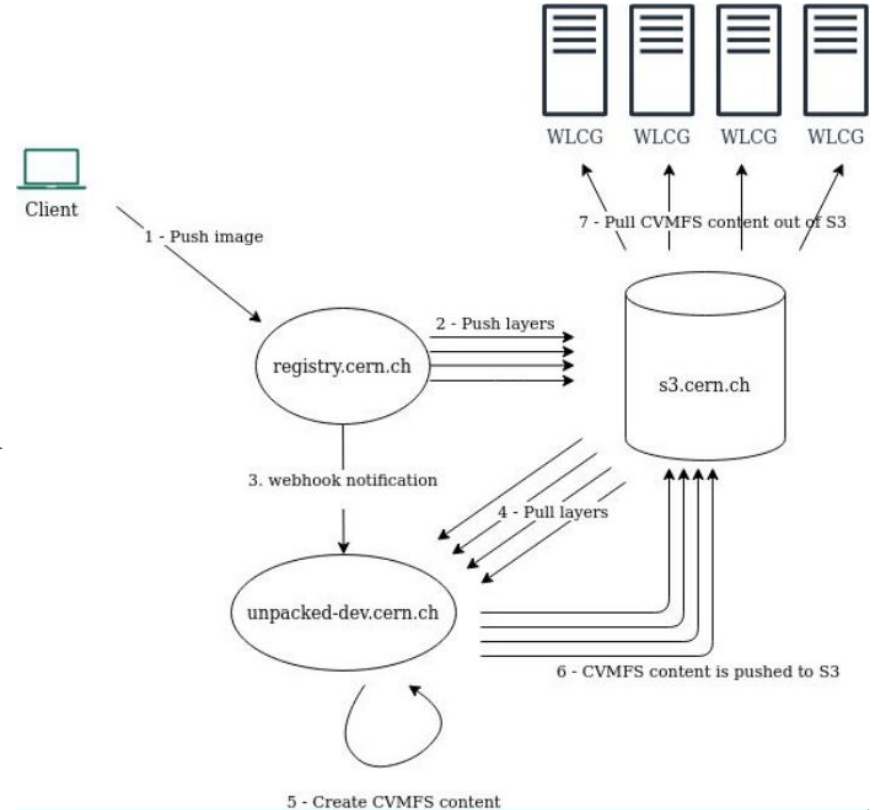
On Image / Chart PUSH, PULL, DELETE

Image Scan COMPLETED, FAILED

Project Quota EXCEED

Example for **unpacked CVMFS**

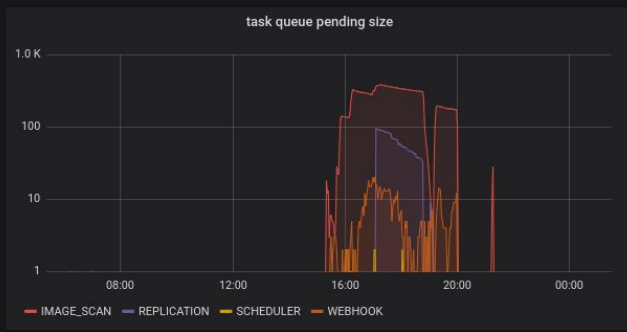
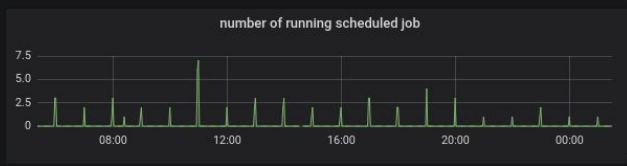
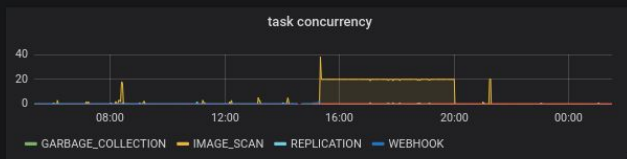
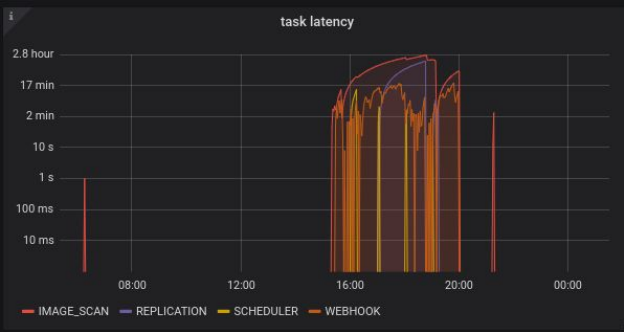
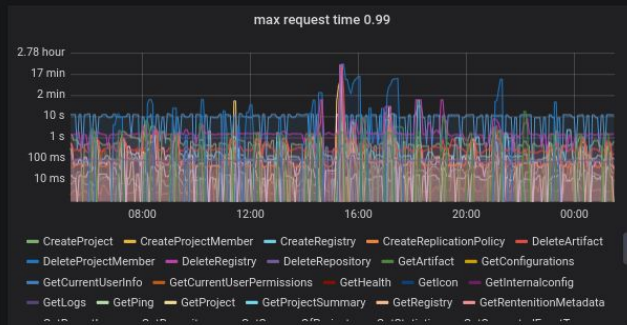
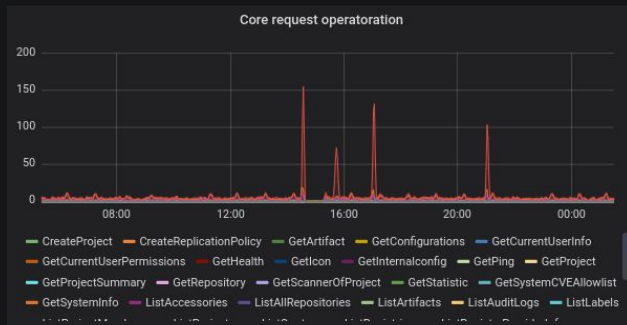
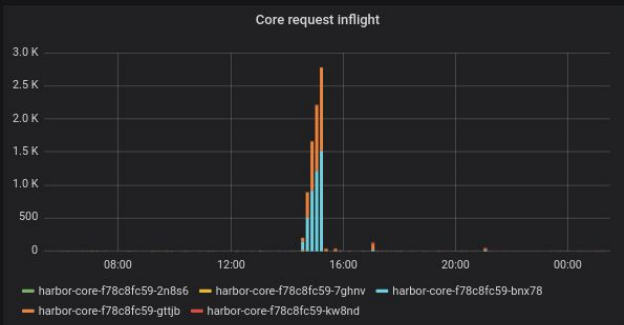
Configuration done at project level



# Integration Points: WebHooks

On Image / Chart PUSH, PULL, DELETE

Image Scan COMPLETED, FAILED



# Use Case: Air-Gapped Environment (TechNet)

Deployed for CERN Accelerator and Controls (BE/ATS)

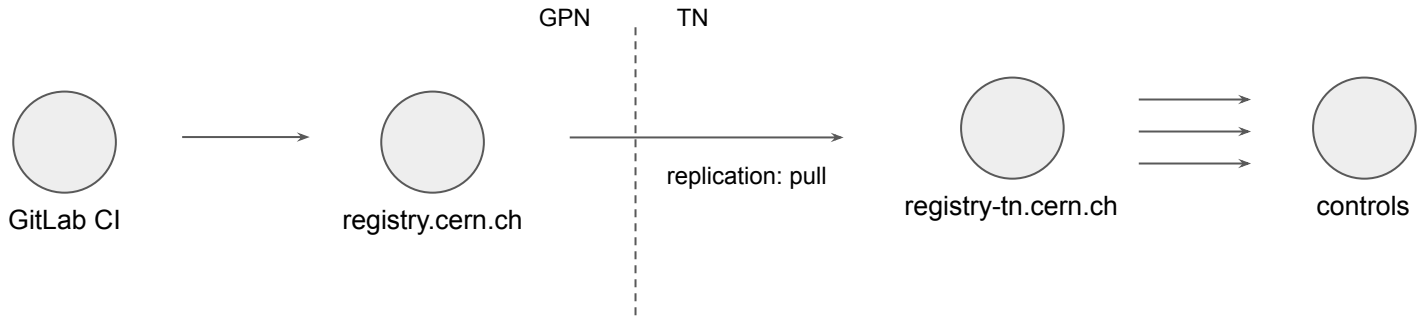
Building on **automated replication** capabilities

Version controlled replication rules

Built-in process for MR submission, approval

Synced with the registry configuration

Vulnerability scans done at first push



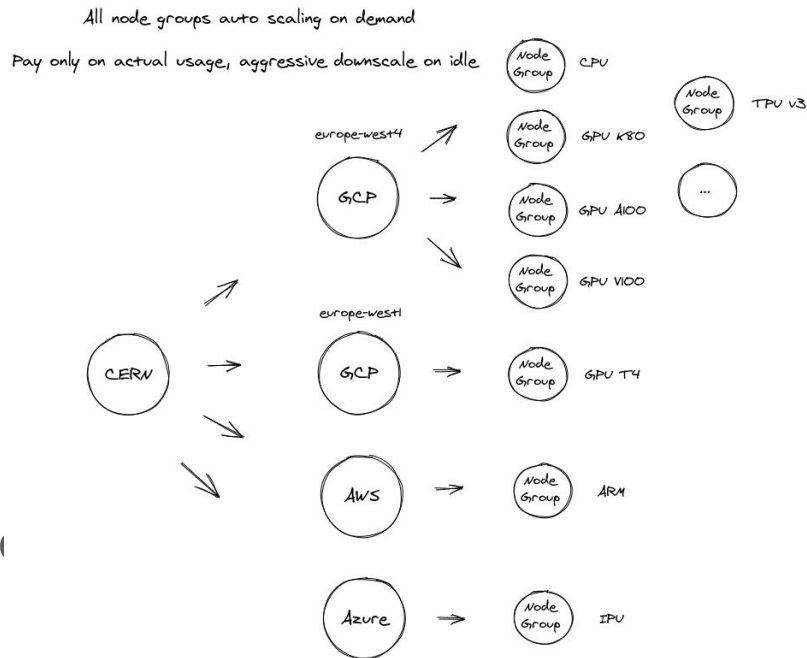
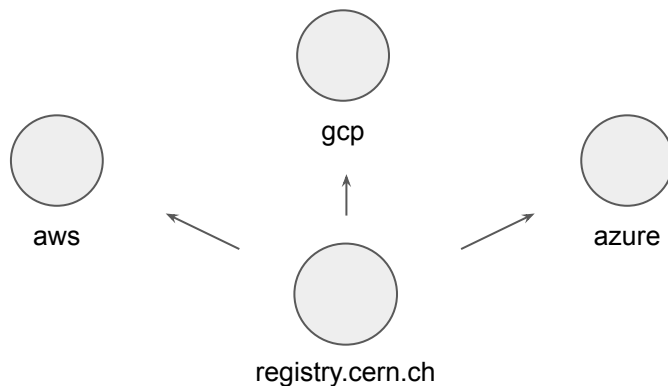
# Use Case: Public Clouds

**Pull-through cache** from central registry

Enforced vulnerability scans, policies

**Automated replication** for *popular images*

Similar MR mechanism possible for end user



Questions?