# Container Registry 2.0

## Implementation of new CI/CD framework for container images maintenance

Prateek Gupta

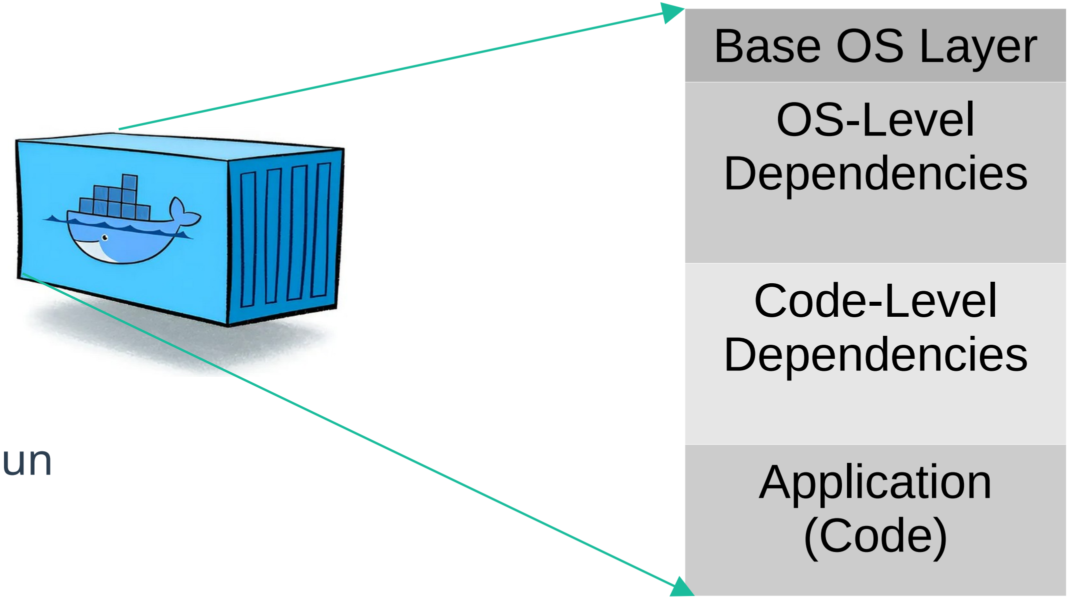
Thüringer Landessternwarte (TLS), Tautenburg

# What is container image ?

**Container images are snapshot of an environment which encapsulate;**

- system libraries
- system tools
- platform settings
- files and other components

that required for an application to run



Base OS Layer

OS-Level Dependencies

Code-Level Dependencies

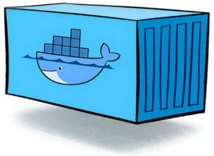
Application (Code)

- **Offers lightweight compute environment in comparison to Virtual machines**

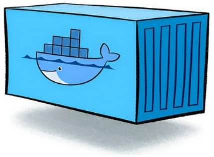
# Significance of container images in PUNCH

- **Users are from variety of research areas : particle, astro-, astroparticle, hadron and nuclear physics**
  - different data-set and different tools and techniques
  - installing supporting libraries and tools on each remotely accessible compute resources !!
    --- **not feasible**

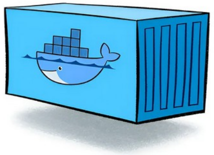
- **Maintain different containers for different use-cases/projects**



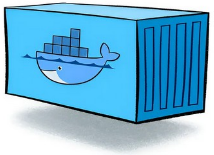
**LINC**



**Simulation**



**LQCD**



**h4lepton**

# What next ?

## Once container images created;

- need to save somewhere !!!
- mechanism for sharing and accessing them !!!

# What next ? → → → Container registry

## Once container images created;

- need to save somewhere !!!
- mechanism for sharing and accessing them !!!

**That’s where a container registry comes in**

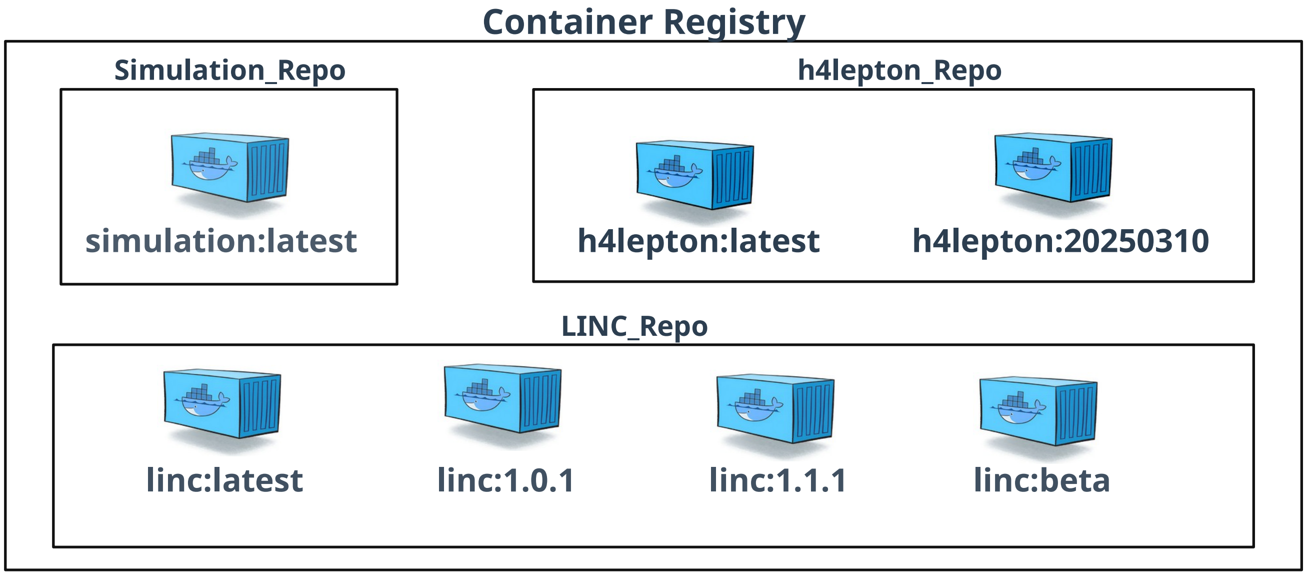
# What is Container Registry ?

**A container registry is a service that host (stores, manages) and distribute application or container images**

- Providing an efficient, centralized resource for container image discovery, distribution, and deployment

# Container Registry architecture

## Container registry is a single or collection of repositories – used to store and access container images



Container Registry

Simulation_Repo

simulation:latest

h4lepton_Repo

h4lepton:latest

h4lepton:20250310

LINC_Repo

linc:latest

linc:1.0.1

linc:1.1.1

linc:beta

Inside an image repository, there can be one or more images;

- with different tags or versioning

# Container Registry and CI/CD pipeline

In practical use-case, with CI/CD pipeline, one can achieve a scalable development of containers or applications on container registry

**CI/CD → continuous integration and continuous development pipelines**

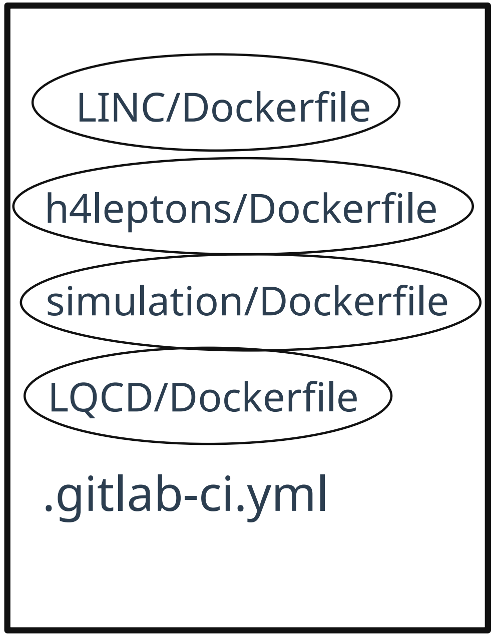
Performing:

- A series of steps in order to deliver a new version of container images
- changes to code(s) are automatically tested and pushed out for delivery and deployment

# Container Registry 1.0

- Using GitLab container registry; a secure and private registry for docker container images
- All containers (and Dockerfiles) in one repository; “**Container Stacks**”
- Any Change(s) to the files in “Container Stacks” triggers the CI/CD pipeline to update the container registry
- Create docker containers for all supporting projects/experiments in one go.

**Container Stacks**



LINC/Dockerfile

h4leptons/Dockerfile

simulation/Dockerfile

LQCD/Dockerfile

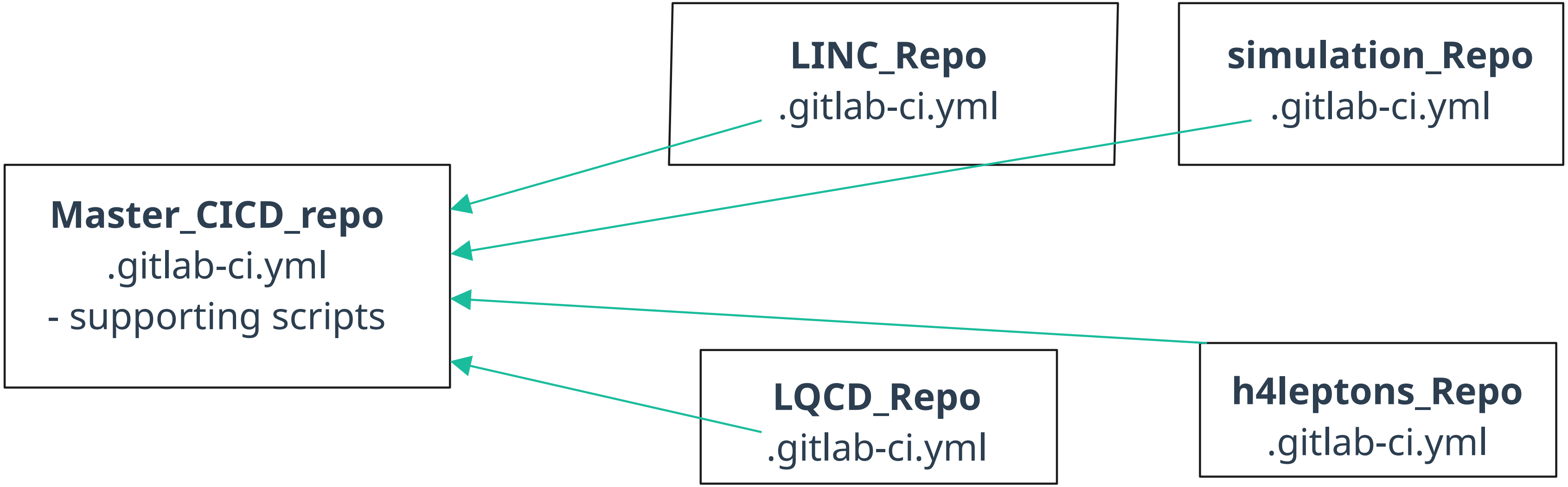
.gitlab-ci.yml

# Container Registry 1.0

- **Limitations:**
  - Even if there is a change updated for a single docker container; it triggers the CI/CD for all
  - Hence, no control on versioning the docker containers
  - Versioning is based on ‘Commit_SHA’ and ‘Date’
  - Create same copies of container images if CI/CD triggered because of other project Dockerfile modification; with new “date” tag
  - Even a change in README file will triggers the CI/CD pipeline

# Container Registry 2.0

- Separate Repository for each project
- Master CI/CD pipeline; again in separate repository
- Master CI/CD linked to individual projects repository



# What Master CI/CD can do?

- **Only the container image corresponds to the modified project repository will be updated; rest will remain unaffected**
- **Update/modification to the README file will not trigger the CI/CD**
- **Enables versioning:**
  - Semantic versioning : MAJOR . Minor . Patch
    - Versioning decision based on commit message: matches the regular expression; accordingly update the version
    - If commit message doesn’t match the key-words, no increment in sementic version
  - Tag them based on ‘date’ as well
  - A version with latest tag always created once the CI/CD tiggered

# Summary

- One CI/CD for all project; preserves previous feature
- Versioning will be easier as each container images reserve there own repository/space
- Versioning style can be flexible: developer's suggestions required
- Limitation of Container Registry 2.0
  - Need a separate repository for each project

# Thank you