LABORATÓRIO DE INSTRUMENTAÇÃO
E FÍSICA EXPERIMENTAL DE PARTÍCULAS
partículas e tecnologia

AI and Containers

Joao Pina (jpina@lip.pt)
On behalf of LIP’s Distributed Computing Group

Outline

1. **Containers in Scientific Applications**
2. Containers for batch processing
3. AI applications
4. Machine Learning operations (MLOps)

Why using containers for applications

Running applications across infrastructures may require considerable effort

- **Computers:**
  - Several computing systems
  - Laptops, Desktops, Farms, Cloud, HPC
- **OSes:**
  - Several operating systems
  - Linux flavors, Distribution versions

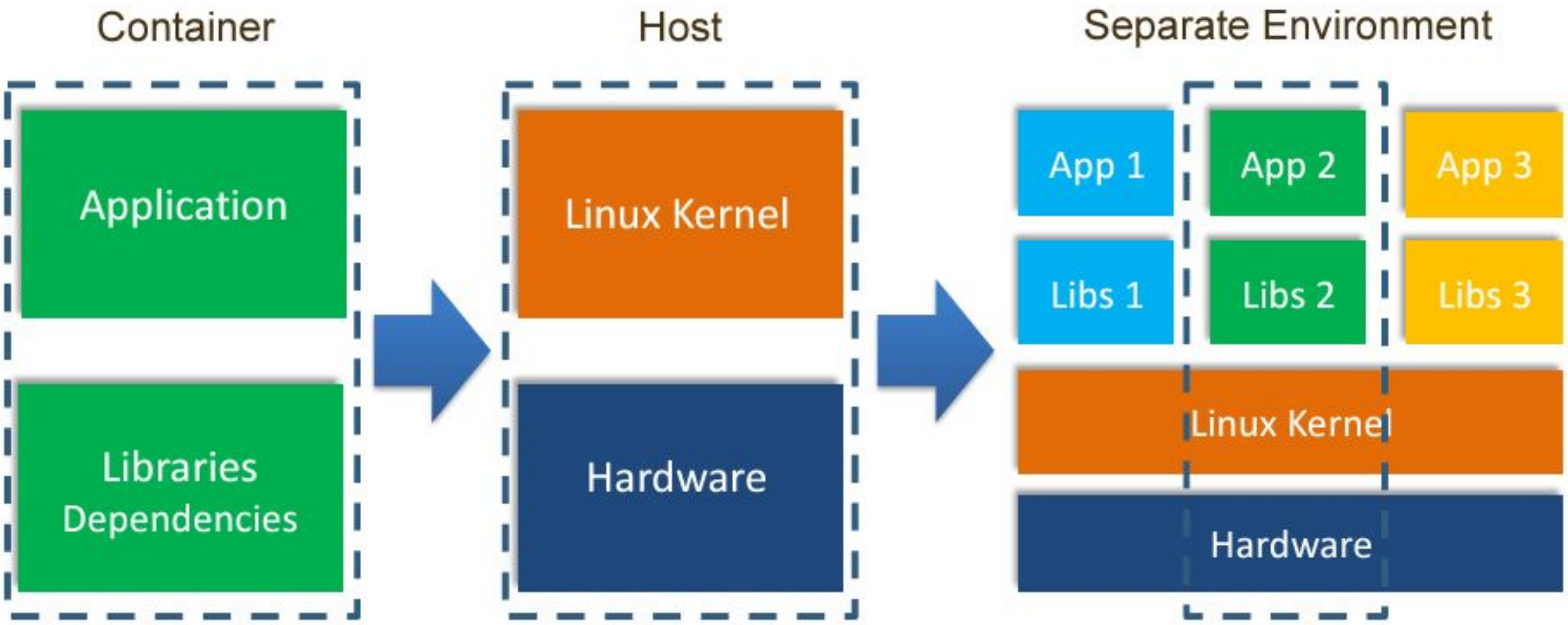
Containerization

- **Lightweight operating system level virtualization method**
  - Relying on isolation instead of virtualization or emulation
  - Isolation of processes from the host operating system with very low overhead
  - Execution across different software environments
- **Enables self contained encapsulation of a given application or service**
  - Including configurations
  - Including software dependencies e.g. libraries and executables
- **Limitations**
  - Hardware architecture must be the same
  - Operating system kernel must have the same binary interface

Why using containers for applications

- **High efficiency**
  - One single operating system kernel shared by many applications
  - Avoids duplication of system processes
  - Performance and resource consumption similar to direct execution in the host
  - Can take advantage of newer more optimized libraries and compilers
- **Better maintainability**
  - Easier application maintenance, distribution and deployment
  - Instead of adapting the user sw to the host, it brings the user environment to the host
- **Easier reproducibility and preservation**
  - Having whole application or service plus its run-time environment in an image
  - Container images can be easily stored for later replay, reuse and preservation

Why using containers for applications



Container Image types

- **Docker and Open Container Initiative (OCI) images**
  - Widely used and supported formats, OCI is a standard
    - docker, **udocker**, Kubernetes, podman, Singularity, Apptainer …
- **Singularity images**
  - Specific format of Singularity
    - Singularity, Apptainer …
- **Others**
  - App Container (AppC) Image Format and Discovery
  - Cloud Native Application Bundle (CNAB)
  - etc

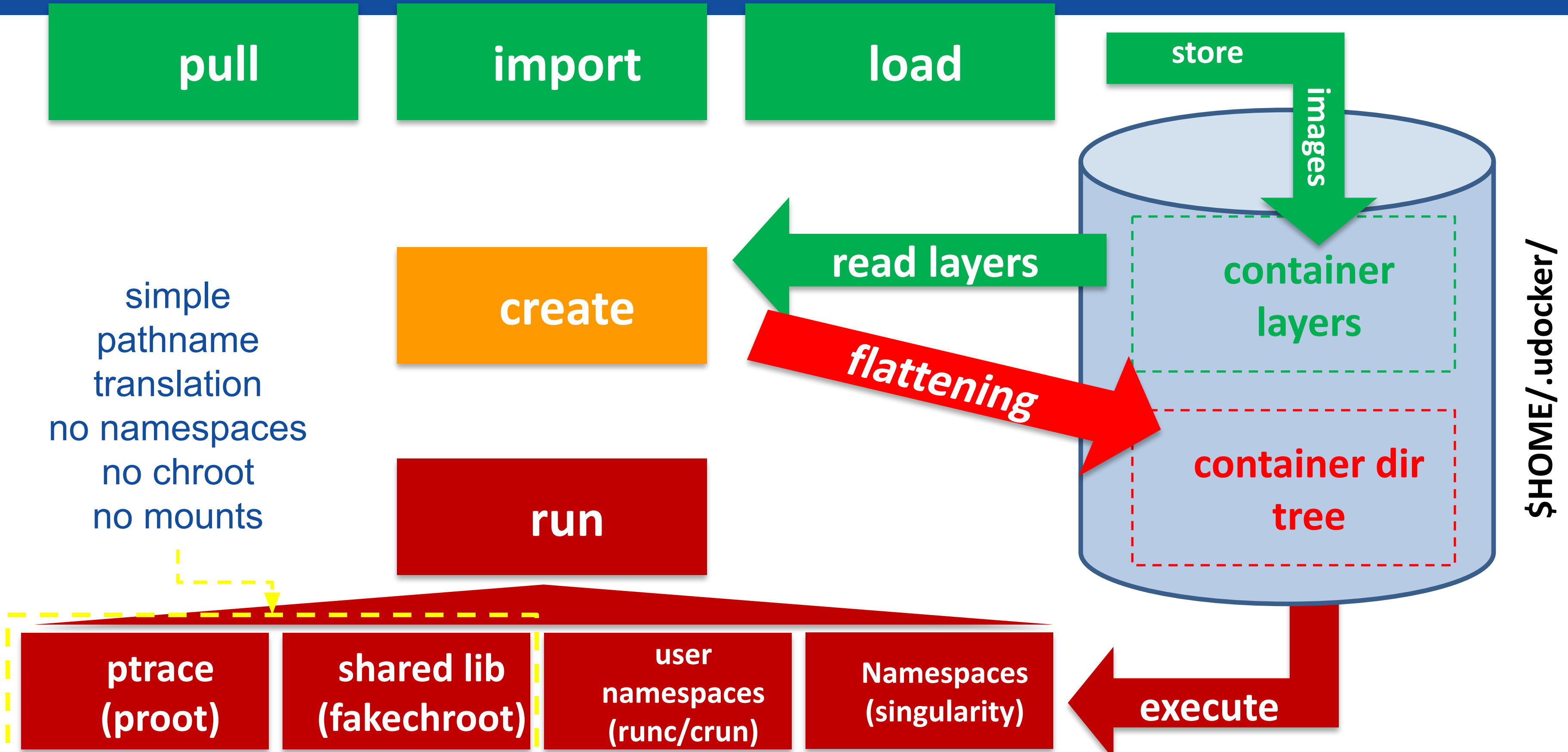
Udocker tool

- **Open Source**
- **Run applications encapsulated in docker containers:**
  - without using docker
  - without using (root) privileges
  - without system administrators intervention without additional system software
  - does not require Linux namespaces
- **Run:**
  - as a normal user
  - with the normal process controls and accounting in interactive or batch systems
- **Tailored to build applications on linux cluster**
  - does not require compilation: Uses Python plus some binaries.
  - Has a minimal dependencies.
  - Required executables are provided statically compiled.

Udocker tool

- **deployment:**
  - Just copy and untar into the user home directory.
  - Ideal to execute containers across different sites.
- **Execution**
  - Allows execution with several approaches/engines. Allows execution with and without Linux namespaces.
  - udocker can be submitted with the batch job:
  - Just fetch or ship the udocker tarball with the job.
- **user interface:**
  - Commands and logic similar to docker.
  - udocker empowers users to use containers:
  - Ideal for heterogeneous computing environments.

udocker is an integration tool



pull

import

load

store

images

container layers

read layers

create

flattening

container dir tree

$HOME/.udocker/

simple pathname translation
no namespaces
no chroot
no mounts

run

execute

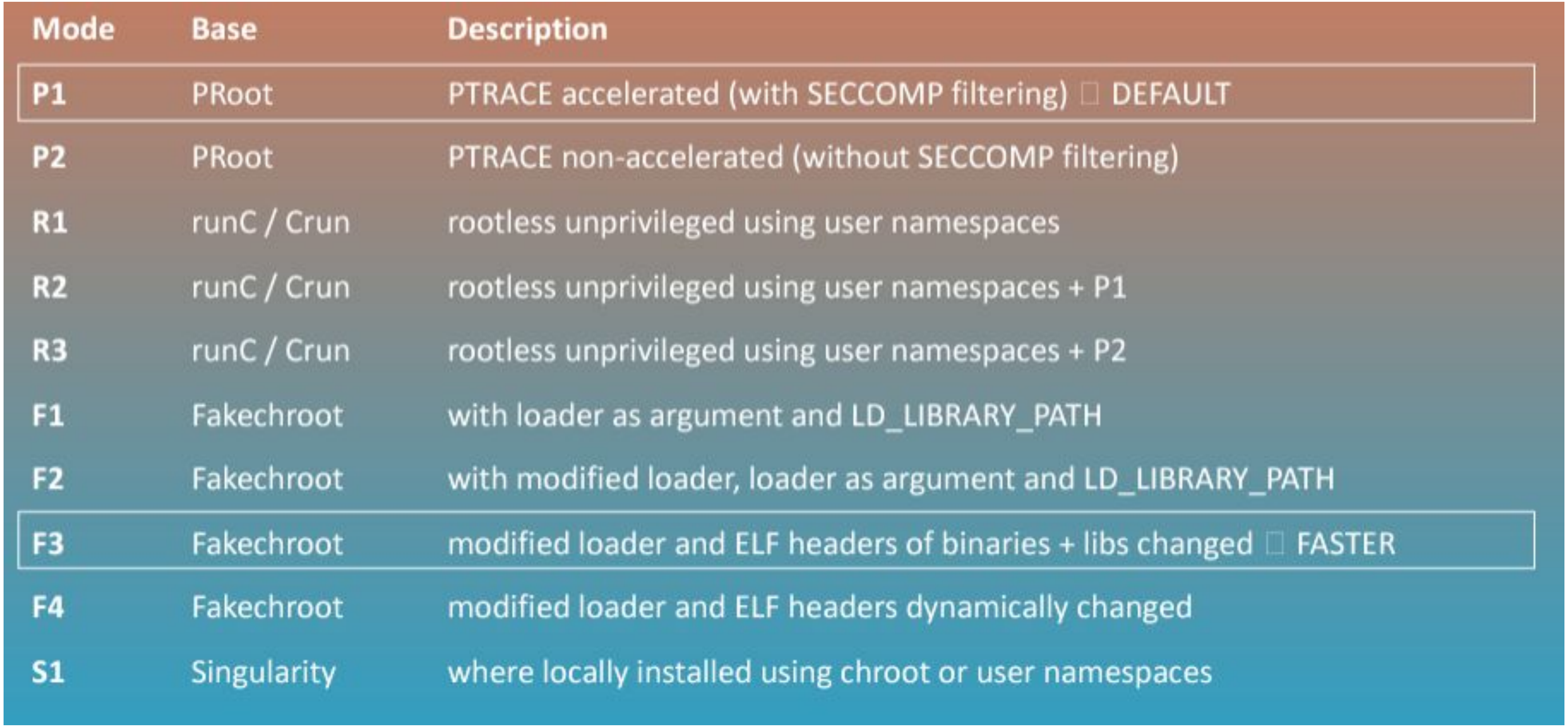
ptrace (proot)

shared lib (fakechroot)

user namespaces (runc/crun)

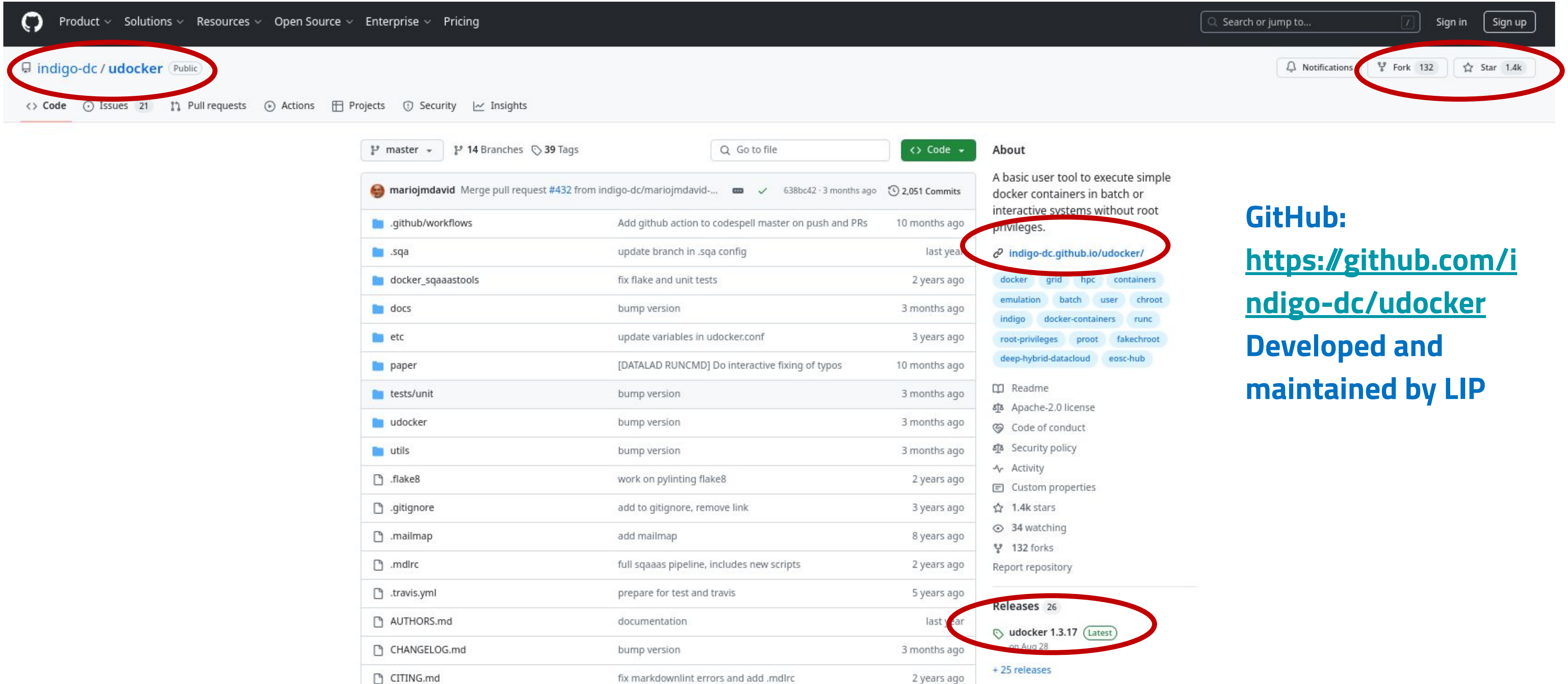
Namespaces (singularity)

Udocker execution modes



Mode	Base	Description
P1	PRoot	PTRACE accelerated (with SECCOMP filtering) □ DEFAULT
P2	PRoot	PTRACE non-accelerated (without SECCOMP filtering)
R1	runC / Crun	rootless unprivileged using user namespaces
R2	runC / Crun	rootless unprivileged using user namespaces + P1
R3	runC / Crun	rootless unprivileged using user namespaces + P2
F1	Fakechroot	with loader as argument and LD_LIBRARY_PATH
F2	Fakechroot	with modified loader, loader as argument and LD_LIBRARY_PATH
F3	Fakechroot	modified loader and ELF headers of binaries + libs changed □ FASTER
F4	Fakechroot	modified loader and ELF headers dynamically changed
S1	Singularity	where locally installed using chroot or user namespaces

Udocker tool



mariojmdavid Merge pull request #432 from indigo-dc/mariojmdavid-... ✓ 638bc42 · 3 months ago 2,051 Commits

.github/workflows	Add github action to codespell master on push and PRs	10 months ago
.sqa	update branch in .sqa config	last yea
docker_sqaaastools	fix flake and unit tests	2 years ago
docs	bump version	3 months ago
etc	update variables in udocker.conf	3 years ago
paper	[DATALAD RUNCMD] Do interactive fixing of typos	10 months ago
tests/unit	bump version	3 months ago
udocker	bump version	3 months ago
utils	bump version	3 months ago
.flake8	work on pylinting flake8	2 years ago
.gitignore	add to gitignore, remove link	3 years ago
.mailmap	add mailmap	8 years ago
.mdlrc	full sqaaas pipeline, includes new scripts	2 years ago
.travis.yml	prepare for test and travis	5 years ago
AUTHORS.md	documentation	last year
CHANGELOG.md	bump version	3 months ago
CITING.md	fix markdownlint errors and add .mdlrc	2 years ago

About

A basic user tool to execute simple docker containers in batch or interactive systems without root privileges.

indigo-dc.github.io/udocker/

docker grid hpc containers emulation batch user chroot indigo docker-containers runc root-privileges proot fakechroot deep-hybrid-datacloud eosc-hub

Readme
Apache-2.0 license
Code of conduct
Security policy
Activity
Custom properties
1.4k stars
34 watching
132 forks
Report repository

Releases 26

udocker 1.3.17 Latest
on Aug 28

+ 25 releases

GitHub:
https://github.com/indigo-dc/udocker
Developed and maintained by LIP

Container and Infrastructure environments

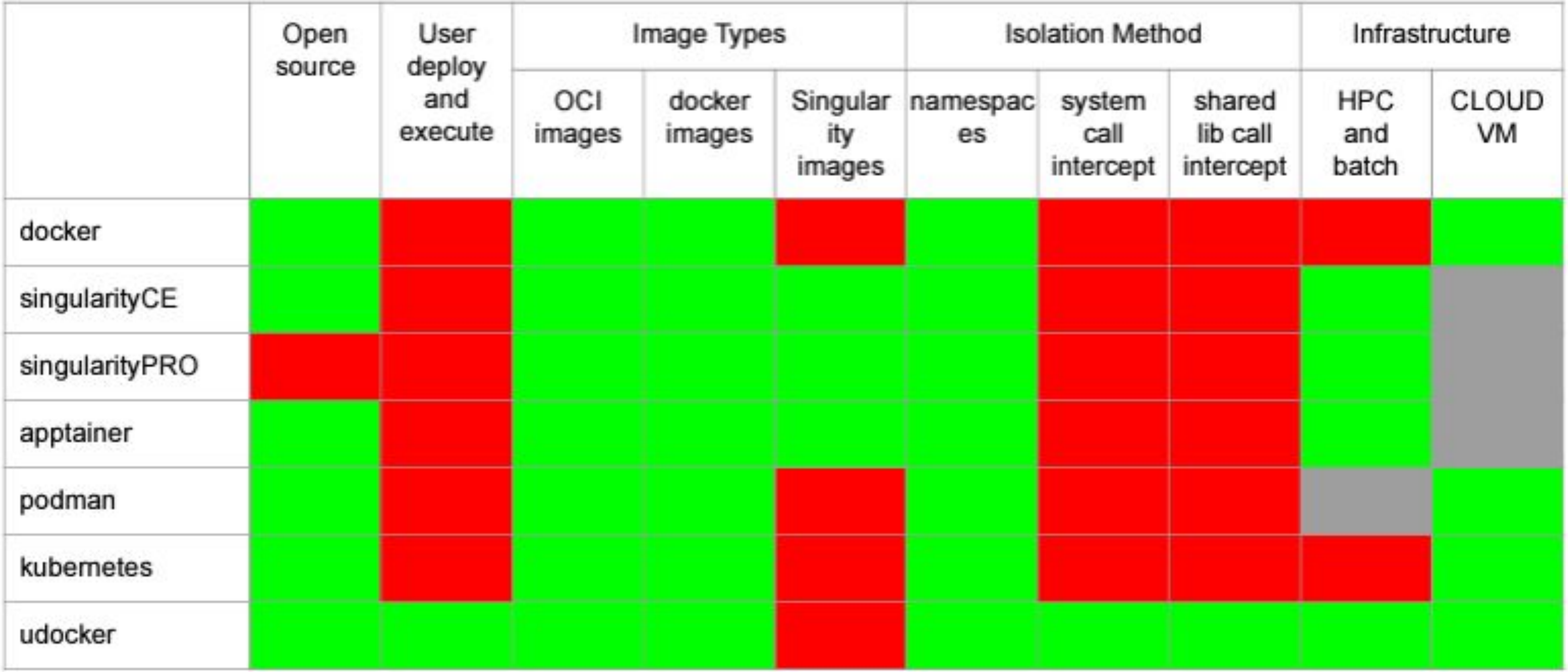
- **Cloud**
  - docker: simple container execution or execution via workflow managers.
  - Kubernetes: execution of containerised services with scalability and HA
- **HPC**

○ udocker: execution everywhere, execution across heterogeneous hosts, execution without namespaces, privileges or other dependencies

○ Singularity or Apptainer: execution in HPC environments, singularity image format may yield faster file access within the container

- **Limitations**
  - Hardware architecture must be the same
  - Operating system kernel must have the same binary interface

Container and Infrastructure environments



	Open source	User deploy and execute	Image Types			Isolation Method			Infrastructure	
			OCI images	docker images	Singularity images	namespaces	system call intercept	shared lib call intercept	HPC and batch	CLOUD VM
docker	🟩	🟥	🟩	🟩	🟥	🟩	🟥	🟥	🟥	🟩
singularityCE	🟩	🟥	🟩	🟩	🟩	🟩	🟥	🟥	🟩	▨
singularityPRO	🟥	🟥	🟩	🟩	🟩	🟩	🟥	🟥	🟩	▨
apptainer	🟩	🟥	🟩	🟩	🟩	🟩	🟥	🟥	🟩	▨
podman	🟩	🟥	🟩	🟩	🟥	🟩	🟥	🟥	▨	🟩
kubernetes	🟩	🟥	🟩	🟩	🟥	🟩	🟥	🟥	🟥	🟩
udocker	🟩	🟩	🟩	🟩	🟥	🟩	🟩	🟩	🟩	🟩

Outline

1. Containers in Scientific Applications

2. **Containers for batch processing**

3. AI applications

4. Machine Learning operations (MLOps)

Containers for Batch processing

- **Encapsulation:**
  - Applications, dependencies, configurations everything packed together. Portability across heterogeneous Linux systems.
  - Makes easier the distribution and sharing of ready to use software.

- **Efficiency:**
  - One single kernel shared by many applications.
  - Performance and resource consumption similar to host execution. Take advantage of newer more optimized libraries and compilers.

Containers for Batch processing

- **Challenges of batch systems?**
  - Integrate it with the batch system (how to start/stop etc) ? Respect batch system policies (such as quotas/limits) ?
  - Respect batch system actions (job delete/kill) ?
  - Collect accounting ?
  - execute in a more basic way?
  - Can we download container images?
  - Can we run without a layered filesystem? Can we run them as normal user?
  - Can we still enforce container metadata?

Containers for Batch processing: limitations

- **Kernel namespaces:** isolate system resources from process perspective
  - Mount namespaces: isolate mount points
  - UTS namespaces: hostname and domain isolation
  - IPC namespaces: inter process communications isolation
  - PID namespaces: isolate and remap process identifiers
  - Network namespaces: isolate network resources
  - User namespaces: isolate and remap user/group identifiers
- **Cgroup namespaces**: isolate Cgroup directories
- **Seccomp:** system call filtering
- **Cgroups:** process grouping and resource consumption limits
- **POSIX capabilities:** split/enable/disable root privileges
- **chroot and pivot_root:** isolated directory trees
- **AppArmor** and **SELinux**: kernel access control

Outline

1. Containers in Scientific Applications
2. **Containers for batch processing**
3. **AI Challenges**
4. Machine Learning operations (MLOps)

AI models challenges

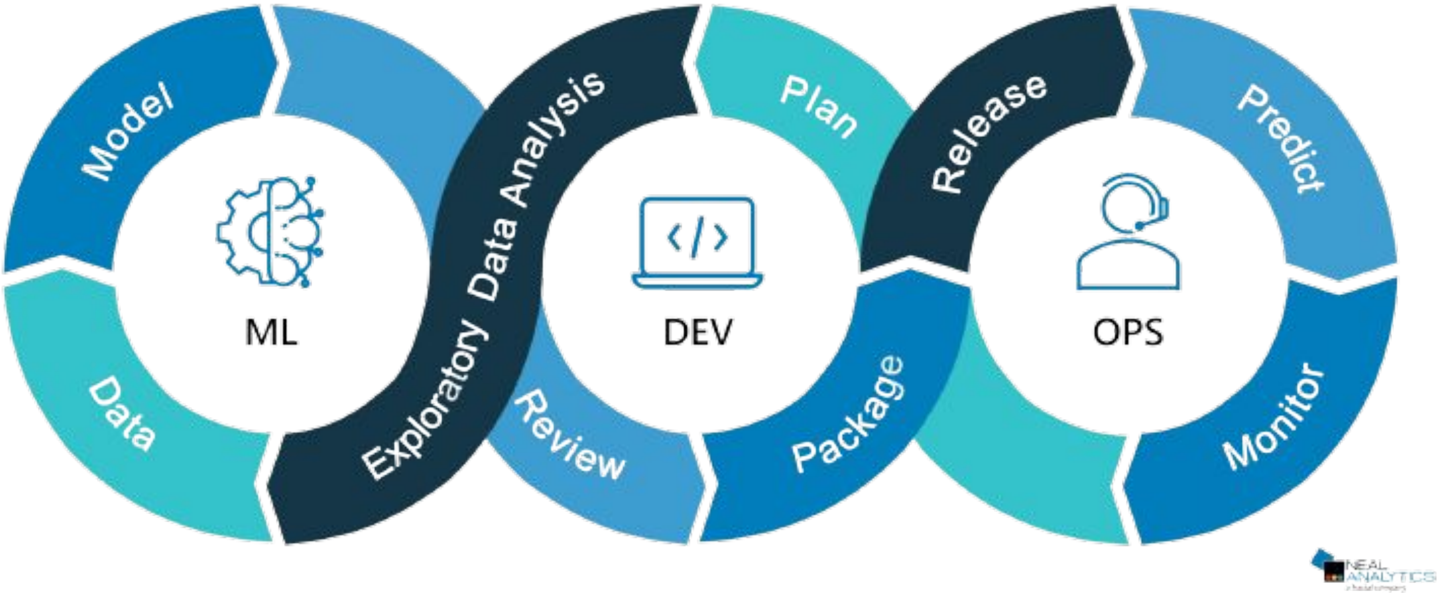
- **Train ML/AI** models is computationally intensive and time-consuming
  - Requires optimization of the of the training process with the right amount of resources (computing + storage)
  - Tailored resources
  - Many interactions on the process (development)
  - Different AI techniques: Composite AI, Federate AI,LLM's, etc.
- Requires interactive reproducible development
  - version tracking (GitHub, GitLab) and Workflow (many solution Pycompss, Node-Red, etc)
  - Containerization of the applications
- Improve efficiency
  - Requires monitoring of resources usage and consumption

AI models challenges

- Enable Secure Access
- Organize and and track all training
  - Requires use of external tools to keep tracking of parameters, changes in workflows (critical on teams working)
- Provide metadata and training dataset
- Deployment
  - Requires precise deployment of the correct version and workflow
  - Promote CI/CD approaches

AI MLOps

ML/DL LifeCycle: ML + Dev + OPS



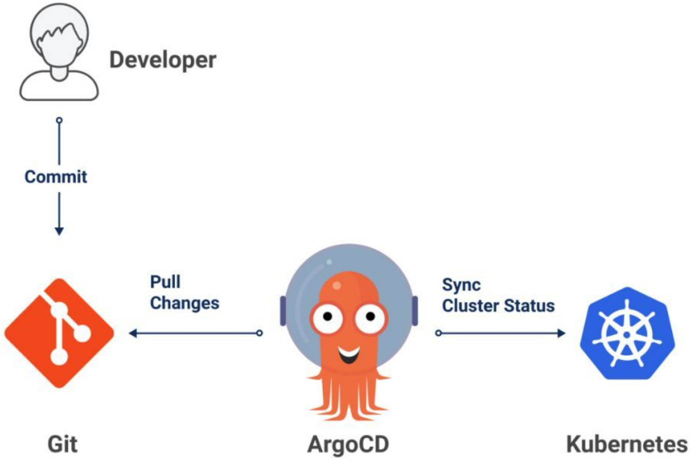
Kubernetes for AI

- Kubernetes (K8s) is the industry-leading container orchestration technology. A powerful platform for automating deployment, scaling and management of containerized applications.

- Also, by adopting a declarative paradigm, K8s simplifies the management of multiple and complex environments.

Kubernetes for AI

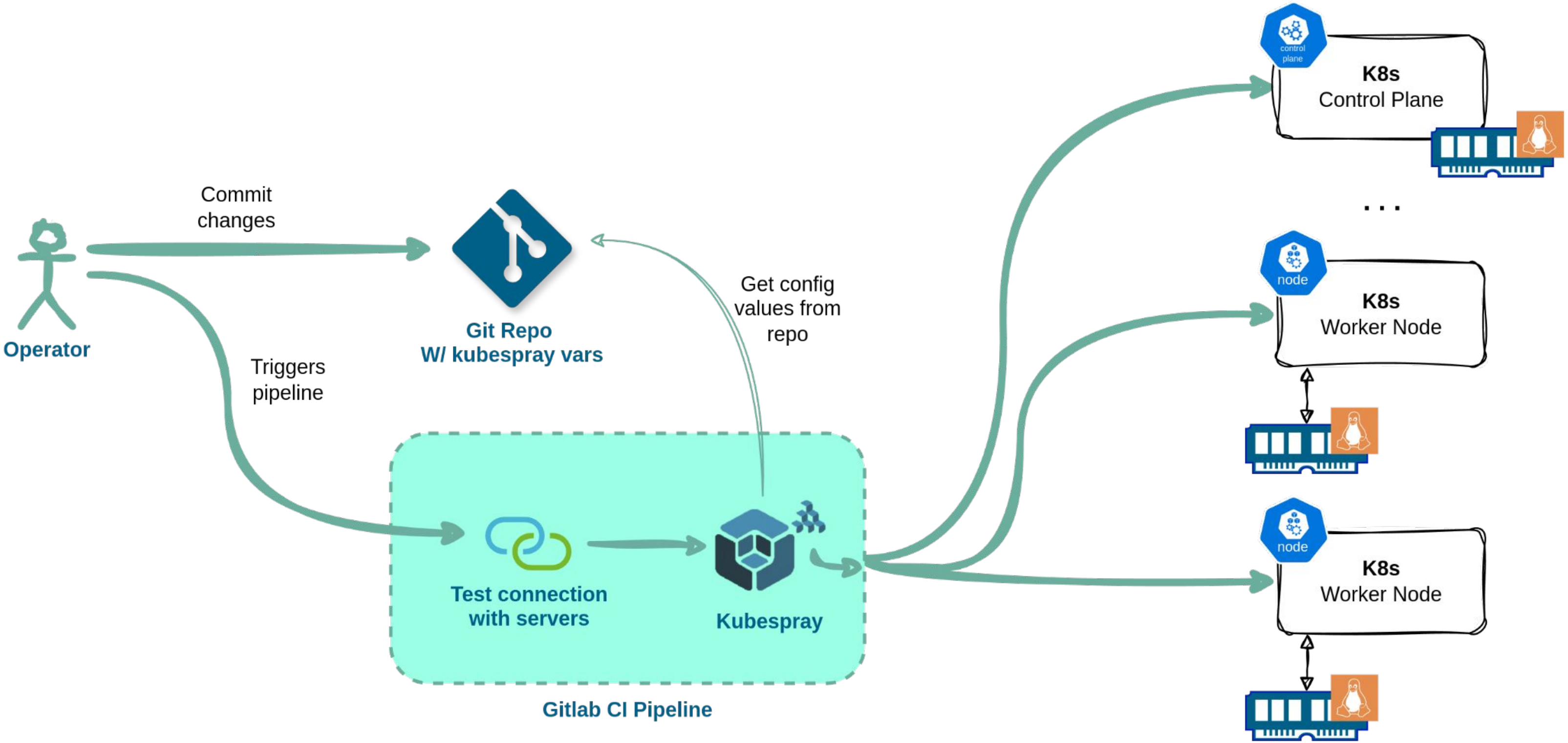
- Allows integration with tools such as ArgoCD and Gitlab CI pipelines, which makes it easier fo organizations to implement the GitOps methodology.



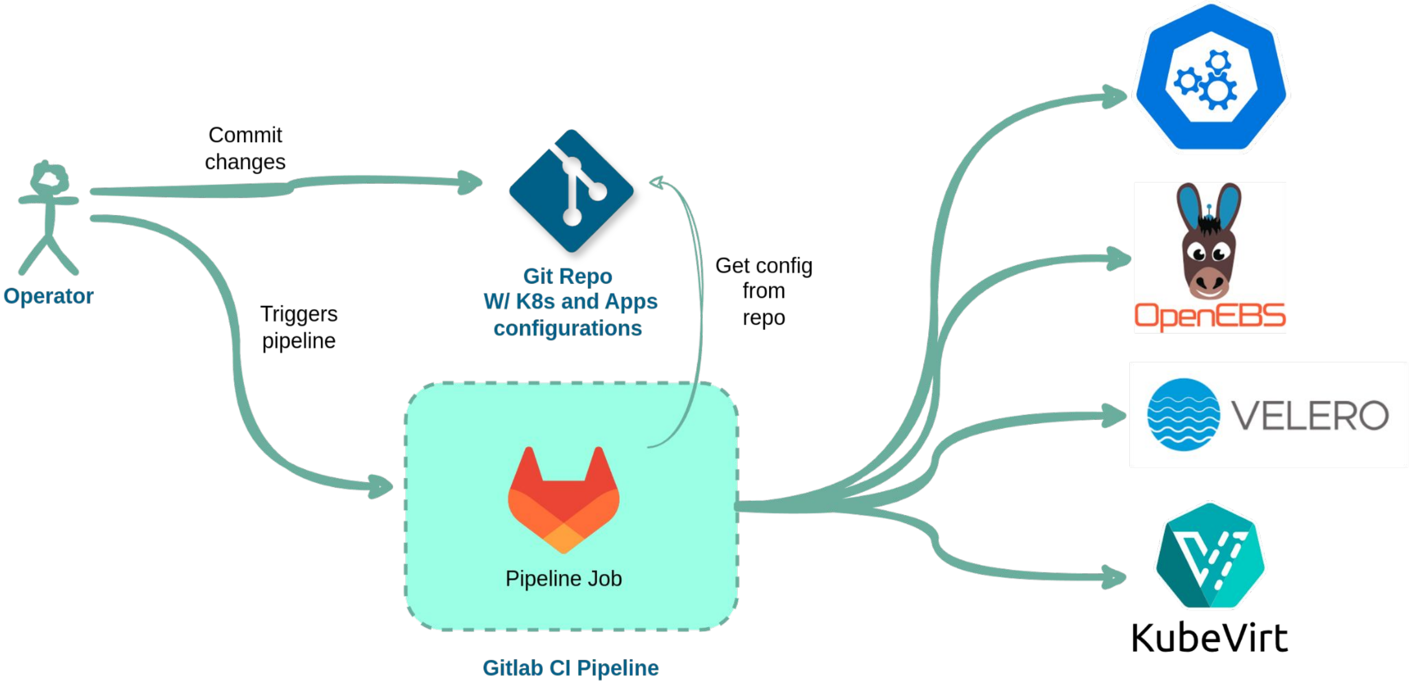
Why Kubernetes for AI?

- **Scalability**: Handles increasing workloads for AI training and inference.
- **Portability**: Runs on any cloud or on-premises infrastructure.
- **Integration**: Easy integration with tools like Gitlab CI/CD pipelines.
- **Serverless AI**: Run AI workloads with managing underlying infrastructure.
- **GitOps**: Use Git as the source of truth for infrastructure and applications, accelerating the delivery of applications.

Kubernetes deployment



Kubernetes deployment



Kubernetes adoption

- Start with docker image creation
- Prepare the deployment using docker compose
- Use available tools to help developers to generate kubernetes ready deployment configurations

Kompose - docker compose to kubernetes

- configuration files
- helm chart

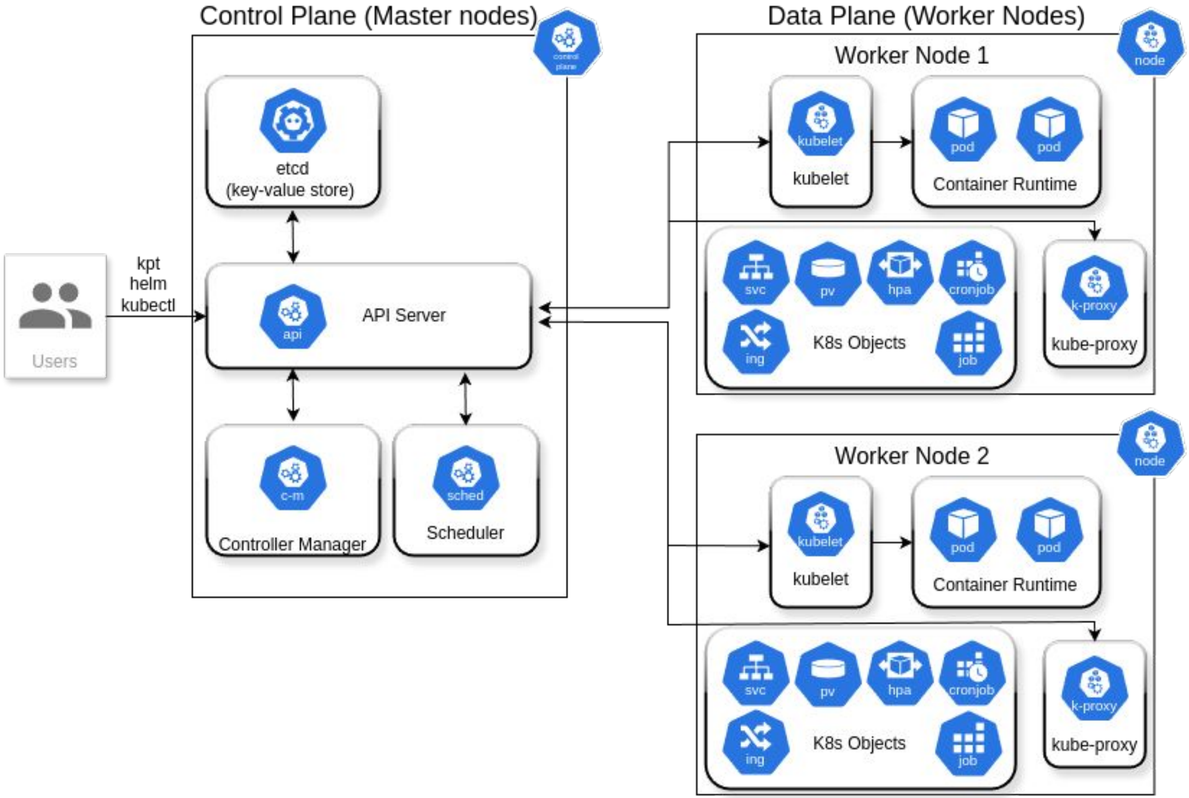
ArgoCD: GitOPS implementation over Kompose generated configurations (CD)

Kubevela: application deployment workflow + **FluxCD**: Kubevela GitOPS application (CD)

Scaffold: build, push, test, deploy, verify (CI/CD)

INCD Public Kubernetes

- Tasks and services in same definition level
- Multiple tools to interact with available API
- Flexibility and easy adoption of new functionality abstracted by K8s objects
- Containers are a first class citizen (supports all available implementations)



Projects Implementations

- **DT-Geo (https://dtgeo.eu/)**
  - Digital Twin of geophysical extremes.
  - Analyse and forecast the impact of geohazards from **earthquakes, volcanoes, tsunamis and anthropogenic seismicity.**
  - Digital Twin of **virtual replica of physical systems that combine real-time data streams and high-fidelity**
  - For integration in **Destination Earth** Initiative.
  - Started in September of 2022.

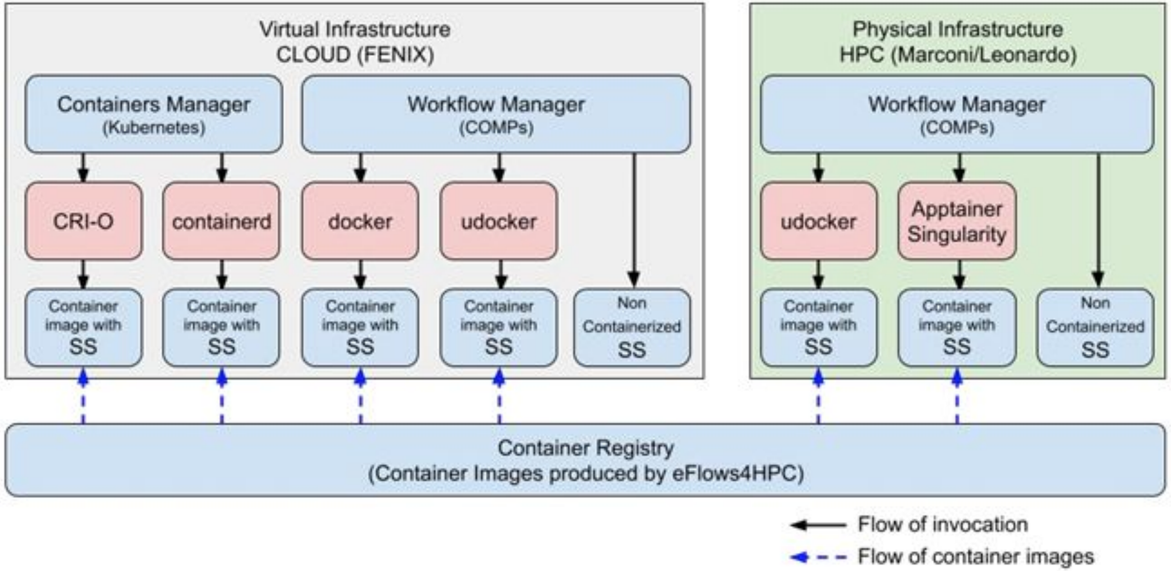
- **InterTwin (https://www.intertwin.eu/)**
  - Develop a common approach to the implementation of Digital Twins (digital twin engine - DTE)
  - Co-design, develop and provide a Digital Twin Engine that simplifies & accelerates the development of complex application-specific DTs that benefits researchers, business and civil society
  - Simplify DT application development with tools to manage AI workflows and the model lifecycle while reinforcing open science practices
  - Liaison with Destination Earth
  - Started in September 2022

- **AI4EOSC (https://ai4eosc.eu/)**
  - AI4EOSC will deliver an **enhanced set of services for the development of Artificial Intelligence (AI), Machine Learning (ML) and Deep Learning (DL) models** and applications for the European Open Science Cloud (EOSC).
  - Started in September 2022

- **iMagine (https://www.imagine-ai.eu/)**
  - Imaging data and services for aquatic science: iMagine provides a portfolio of image datasets, h**igh-performance image analysis tools empowered with Artificial Intelligence (AI),** and Best Practice documents for scientific image analysis.
  - Life Sciencies (aquatic sciences)
  - Started in September of 2022.
  - https://www.imagine-ai.eu/

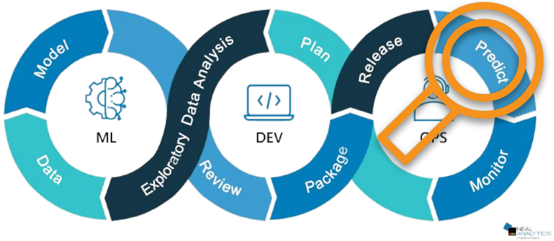
DT-GEO

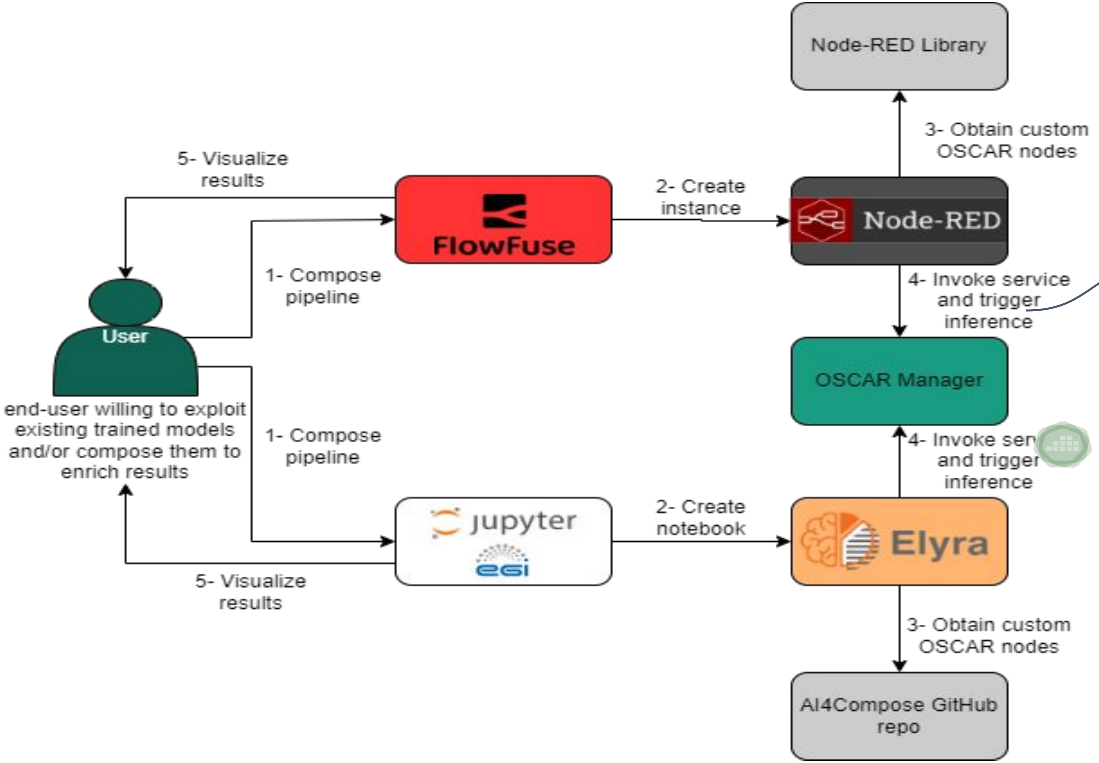
- **Container execution engines**



AI4EOSC

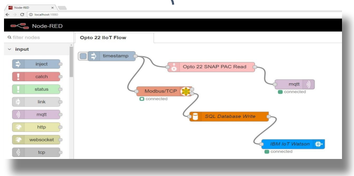
- **AI Model Inference Pipelines:**





Node-RED Library
3- Obtain custom OSCAR nodes
5- Visualize results
2- Create instance
FlowFuse
Node-RED
1- Compose pipeline
4- Invoke service and trigger inference
User
OSCAR Manager
end-user willing to exploit existing trained models and/or compose them to enrich results
1- Compose pipeline
4- Invoke service and trigger inference
2- Create notebook
jupyter
egi
Elyra
5- Visualize results
3- Obtain custom OSCAR nodes
AI4Compose GitHub repo

OSCAR



LIP

LABORATÓRIO DE INSTRUMENTAÇÃO
E FÍSICA EXPERIMENTAL DE PARTÍCULAS
partículas e tecnologia

Thanks!

Discovery through science

Innovation through technology

Sharing with People