Container at PaN facilities

- Why? Who? How?
- Requirements? Missing items & issues?
- Sharing efforts?
- Not: introduction to containers

Disclaimer:

Very far from being an expert!

Why we started to use container @ DESY

Reoccurring question/request:

- Group/User needs to preserve old pieces of scientific software
  - Won’t compile/run on actual operating systems
- Deploy highly complex software framework
  - Beyond users capabilities

Why we started to use container @ DESY

Reoccurring questions/requests:

- Group/User needs to preserve old pieces of scientific software
  - Won’t compile/run on actual operating systems
- Deploy highly complex software framework
  - Beyond users capabilities

Implicit assumption:

- Build it once and it will run “forever” (aka 5-10yrs)

Why we started to use container @ DESY

Reoccurring question/request:

- Group/User needs to preserve old pieces of scientific software
  - Won't compile/run on actual operating systems
- Deploy highly complex software framework
  - Beyond users capabilities

Implicit assumption:

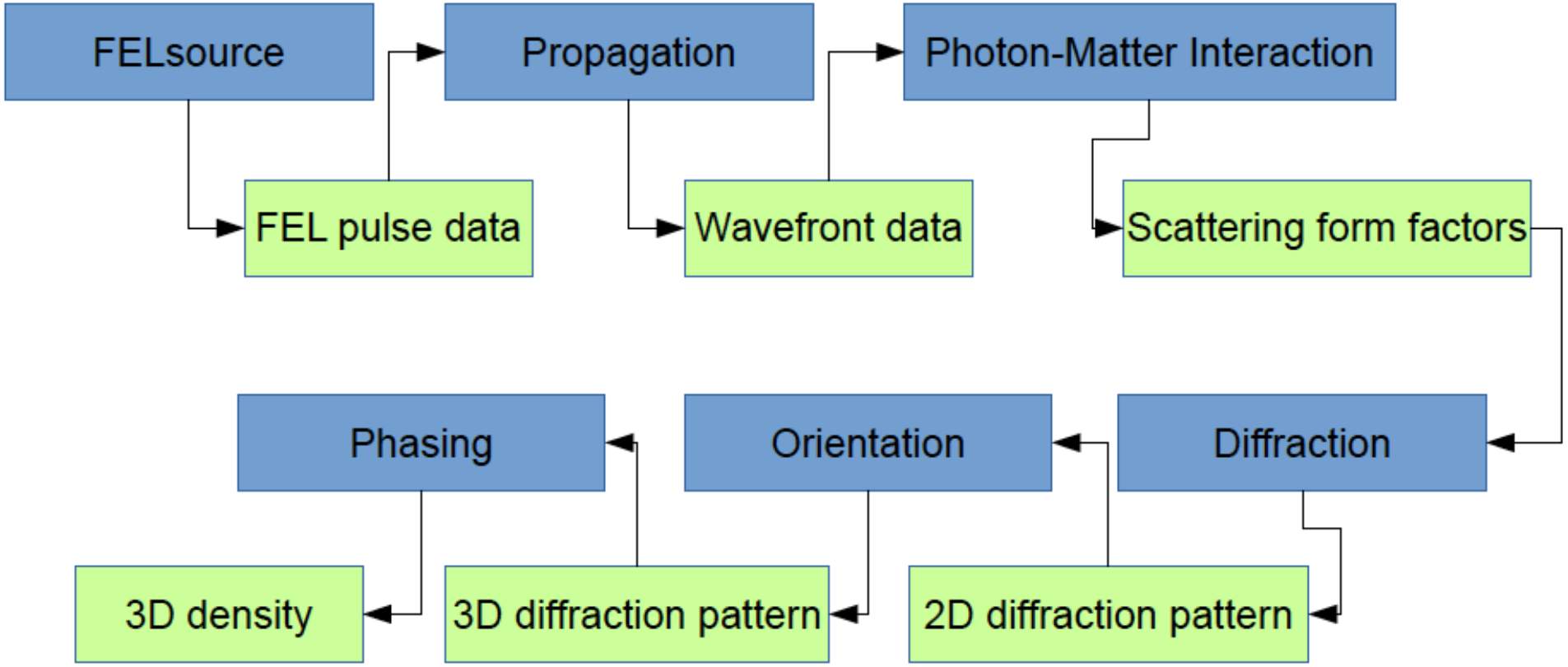
- Build it once and it will run "forever" (aka 5-10yrs)

Not sure it's a valid assumption:

- Lack of detailed, interpretable meta-data & provenance?
- Lack of detailed version control?
- OS vulnerabilities?
- Evolution of underlying engine?
- Interpretation of output (evolution of formats)
- Communication with outside world (evolution of APIs, protocols)

Deploy highly complex HPC software framework

SIMEX: FEL end-to-end simulation framework (https://github.com/eucall-software)



… not entirely HPC, but lacking mechanisms for heterogeneous deployment

Deploy highly complex HPC software framework

- Many modules independently developed over last decades
- Variety of different build instructions (make, cmake, build.sh, f77 –o …)
- Lots of different code basis’
  - Python
  - Fortran (variants)
  - C/C++ (variants)
  - Cuda/OpenCL, openmp, mpi
- A true dependency hell
- Various deployment schemes utilizing
  - HPC
  - Cloud
  - Batch
- Utilizing different schedulers
  - SLURM
  - Htcondor
  - SoGE

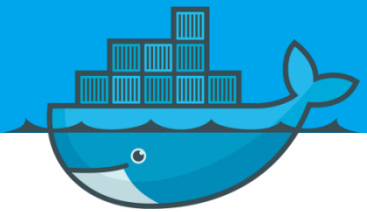
Python madness

- Anaconda nice framework
  - Fairly complete set of modules for typical P&N applications
  - Widely used by our communities
- Spoils reproducibility
  - Result depends on time of assembly (even for identical versions)
  - For example mkl underneath numpy
  - For some simulations we see small deviations
  - For some applications we see segfaults
- Need an image which always produces identical results
  - Maybe shouldn’t have used python in the first place …

Experiences with Docker

- Container seemed the obvious choice to solve most of the issues
- Looked at Docker & Shifter in 2015
  - Shifter at that time not openly available
  - Docker security issues too severe
- Took another start in 2016
  - User namespaces major improvement
    - Unfortunately with some unexpected bugs (BeeGFS)
    - Meanwhile major issues solved
  - Embedded into slurm pro-/epilog
    - Yet another bug making prologs unusable
    - Meanwhile in production on HPC farm
    - Some issues remaining …
    - … and some issues to tackle together

Experiences with Docker



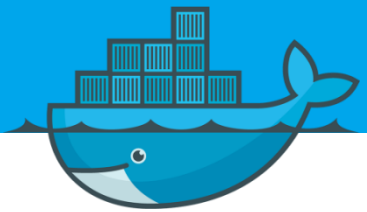
- Overall very positive
- Not too intrusive
- Efforts to implement and maintain very reasonable
  - complaints about limited GPU support
  - complaints about steep learning curve
  - complaints about Windows/Mac beta

- Benchmarks on our HPC platform look great
  - No impact on HPC applications whatsoever
  - Recently had some issues with “multi-threaded” python code (simex)
    - Turned out be the stack size, but hard to figure out

	Cores	HPL (TFLops)	HPCG (TFLops)
Maxwell	64	1.56	0.033
Maxwell+Docker	64	1.559	0.033
Maxwell	368	9.04	0.192
Maxwell+Docker	368	9.0	0.191

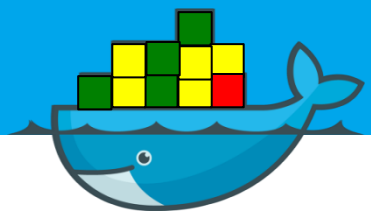
- Benchmarks on ib bandwidth, latency
- filesystem throughput, …
- MPI benchmarks
- HP Linpack
- LNBL mpi-tests slightly worse

Classes of applications



- **User namespaces**
  - User has root inside container → map user (unroot)
  - User does not have root inside container → do not map user
- **Separate images into two classes**
  - Images we trust (staff supplied)
  - images we don't trust (user supplied)
- **Three classes of applications**
  - Applications openly shareable
  - Applications shareable under certain conditions (license, location, group,…)
  - Non-shareable applications

Classes of applications



- We intend to provide an exhaustive list of applications or rather frameworks (aaS) …
  - Electron Microscopy, Tomography, Macromolecular Crystallography, MD, CFD, QC, …
  - Usually there are multiple frameworks sometimes with 100s of applications
- Almost everything can easily be dockerized
  - not everything distributable due to licenses
- … but would very much like to share the efforts

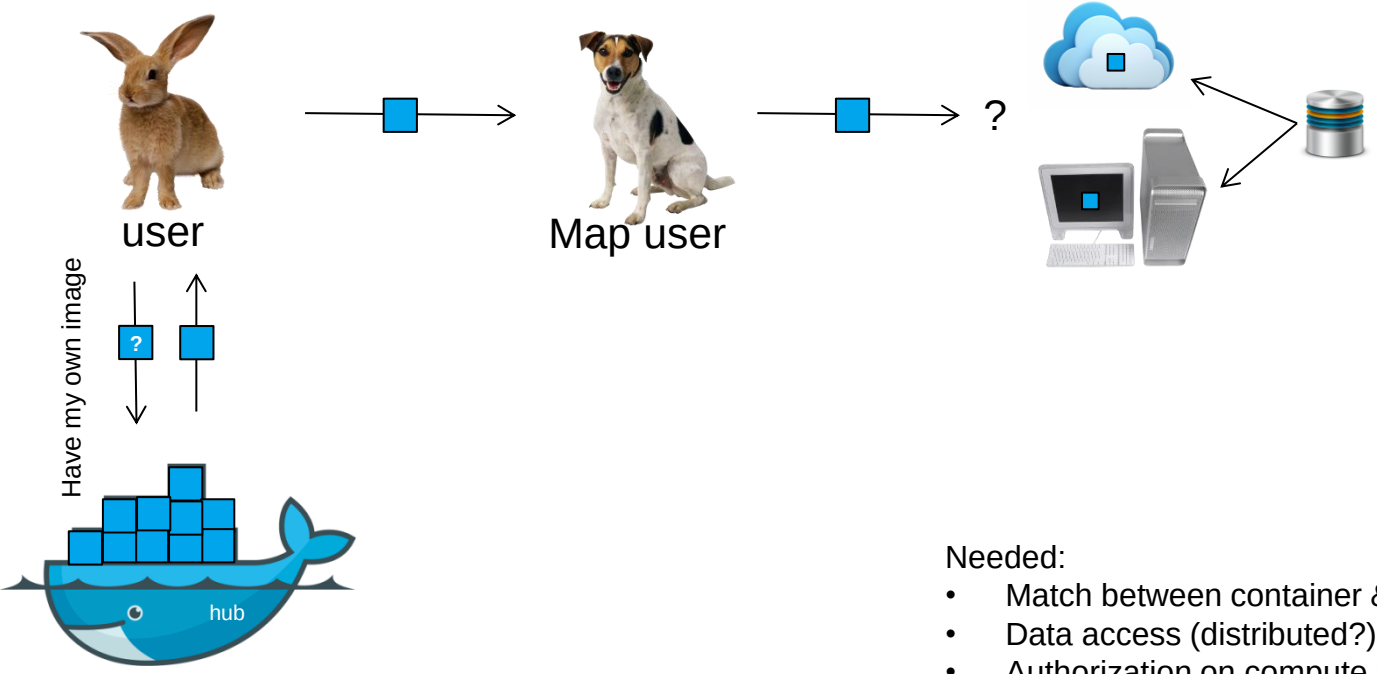
Trusted Openly shareable
- Signed or Certified Container
- Provenance (who & how)
- Versioning
- Upload controlled

Untrusted Openly shareable
- Provenance (who & how)
- Versioning

Trusted restricted shareable
- Signed or Certified Container
- Provenance (who & how)
- Versioning
- Access controlled

Not shareable
- Signed or unsigned
- Provenance (who & how)
- Versioning
- Exec control

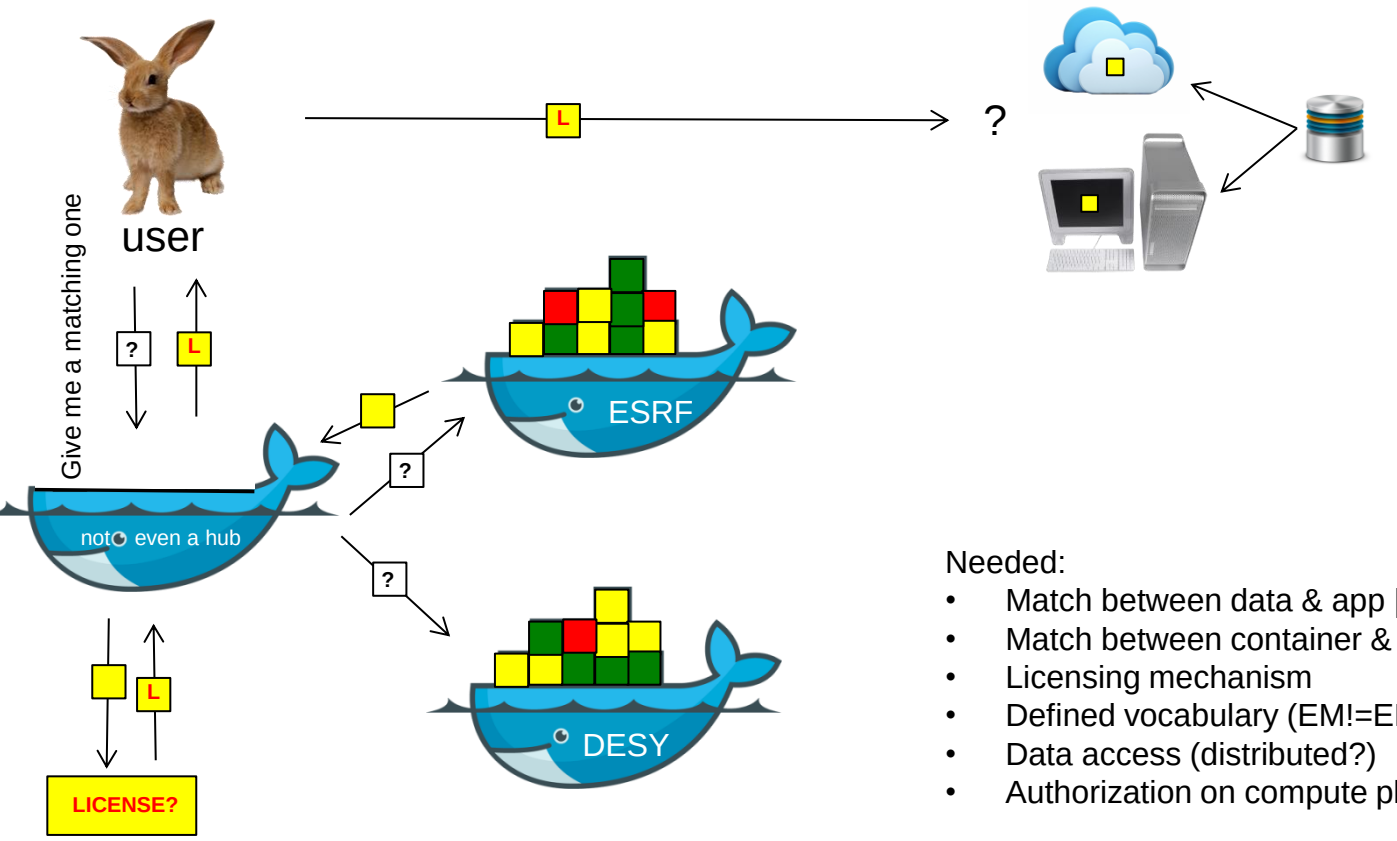
Scenarios I



Needed:

- Match between container & compute resource (cloud? HPC?)
- Data access (distributed?)
- Authorization on compute platform

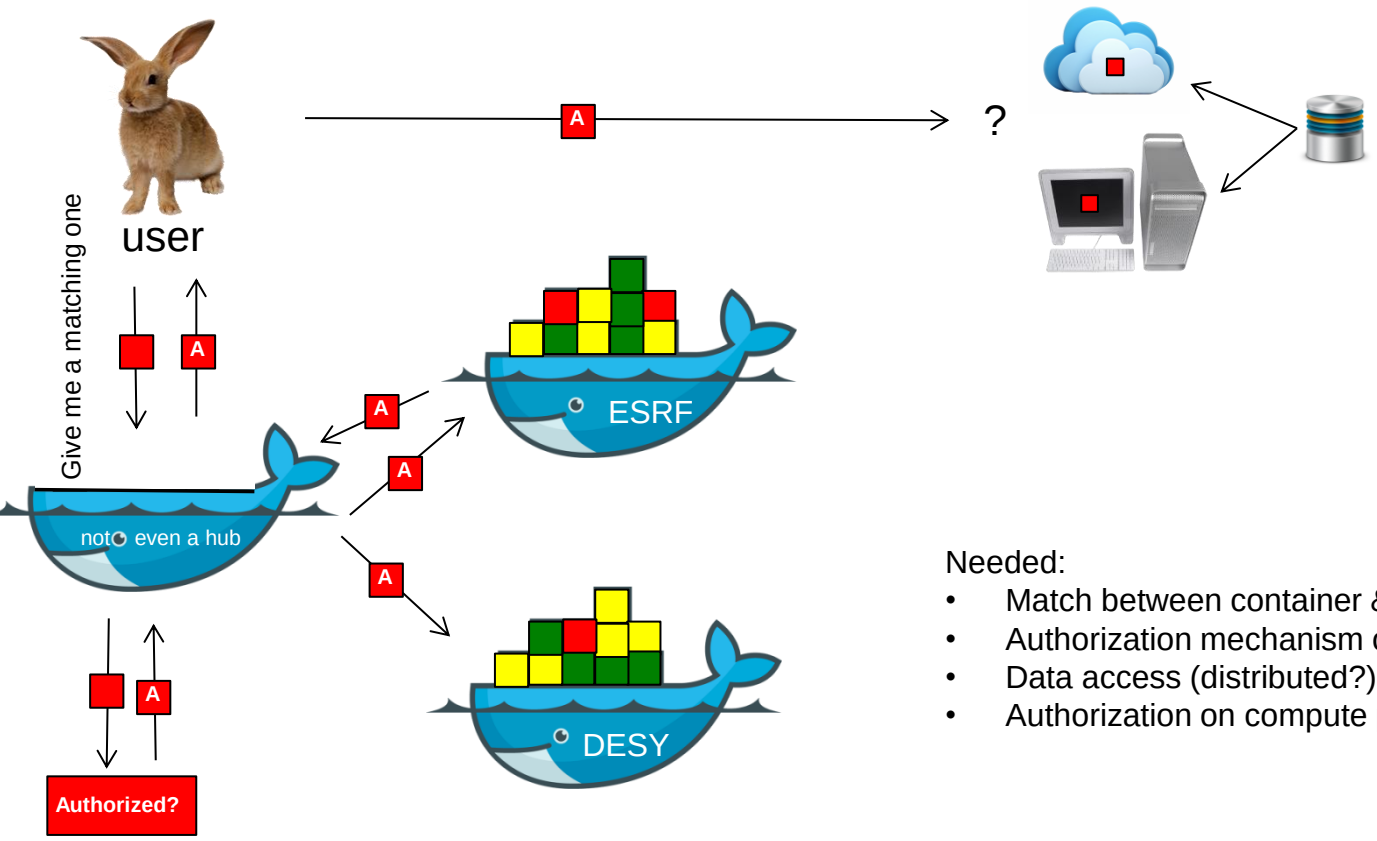
Scenarios II



Needed:

- Match between data & app || framework || capabilities
- Match between container & compute resource
- Licensing mechanism
- Defined vocabulary (EM!=EM)
- Data access (distributed?)
- Authorization on compute platform

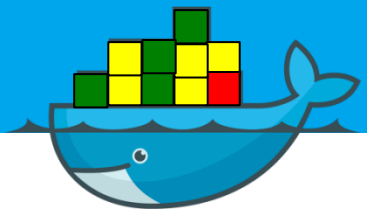
Scenarios III



Needed:

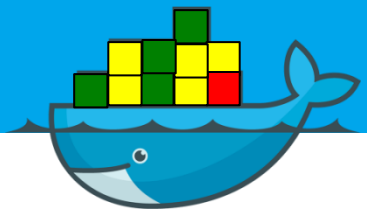
- Match between container & compute resource
- Authorization mechanism on repositories
- Data access (distributed?)
- Authorization on compute platform

Some basic requirements



- Trusted repositories (certified?)
- Trusted images (signed. Who? What? How?) with unique ID
  - Not sure signed tags are sufficient
- Deep versioning
- Authorization mechanisms
- Licensing mechanism
- Data with a defined experimental context and possibly some constraints
  - Needs a defined vocabulary
- Frameworks with defined capabilities: needs a matching vocabulary
- Capability discovery across repositories constrained by licenses & authorization
- Orchestration
- User defined &| application defined compute requirements
- Data access. Secure credential PAGs
- Users choices (only from ESRF! Only frameworks with relion!)

Embedding information & constraints



- Some information and light-weight data exchange …
  - … between service, container, user, platforms, etc
  - … some frameworks might require mixed compute resources
- Some information must be embedded inside images/container
  - Docker allows custom tags, everything which is a “string”
  - Licensing etc as custom tags?
  - Complex information and constraints, parameters encoded as json
  - Frictionless DP to inject complex tag structures & exchange between instances?

Container – what labs are looking at

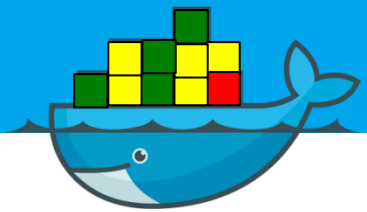
Short survey 09/2016

	Container	Hubs / Repos	CI	Orchestration	Cloud / HPC / GPU
DESY	Docker Shifter	Internal. Separated User & Staff supplied images	Bamboo	Kubernetes et al investigated	+ / + / +
PSI	Shifter Docker	?	?	?	? / + / ?
ISIS	Docker Vagrant	DockerHub GitHub	Travis	Ansible + Docker -	+ / + / +
DLS	?	?	?	?	?
ESRF	Docker	internal	?	?	+ / + / ?
ANKA/KIT	Shifter Docker	?	?	?	? / + / +
ALBA	Docker	DockerHub	Jenkins	Docker-compose	+ / + / ?
ELETTRA	Docker	Internal for sysadmin & selected developers	?	CoreOS & Fleet Testing Swarm	+ / + / +
HZB	Docker	DockerHub	?	?	? / ? / ?

Container – what labs plan

	Supported Applications	Plans	Status	share	Concerns
DESY	SIMEX: FEL simulation framework	Photon Science / EM Apps in public Cloud & HPC	HPC in production In preparation otherwise	+	Security (resolved) Performance (python multi-processing poor)
PSI	Personal development tool	No concrete plans	Concept evaluation	+	
ISIS	EPICS, Plankton Services for ESS	Service & App deployment	Close to production	+	Docker Mac/Win beta Poor w/r Toolbox
DLS	?	Postprocessing. SaaS	?	+	?
ESRF	iSPyB components owncloud	Photon Science apps in Cloud & HPC	In production for services	+	Security Orchestration
ANKA/KIT	Tomography		In preparation	+	Security GPU support
ALBA	Tango/Sardana	CI in the cloud	Proof-of-concept	+	Stability, security Tools standards Config mgmt
ELETTRA	Scientific. HPC mgmt, developmt, prototyping	Sharing, Funding	In production since early docker	+	Steep learning curve
HZB	ICAT (data catalog)	Virtual environment for scientific data analysis	Testing	+	Security Credential handling

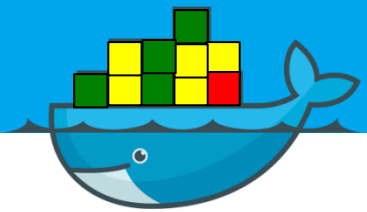
Summary of labs views on Container



- Almost all P&N labs are just starting to explore the Docker/shifter ecosystem
  - Shifter & Docker the only serious candidates
- All P&N have positive experience albeit limited expertise!
- All P&N have security concerns
- Most labs have concrete plans to use container for scientific data analysis
  - Focus are clearly scientific applications
  - Packaging of complex frameworks
  - Preservation & Reproducibility
  - Deployment & orchestration on HPC, Clouds
- All P&N would like to share efforts / collaborate / join forces

Quote from ISIS: *Overall, our experience with Docker has been very positive. Containers help tremendously in managing the mess of dependencies and conflicts that inevitably come with using large systems that have been built over decades with contributions from many different sources. I think the motivation behind Docker, to be able to create predictable and reproducible results, isolated from environment, fits very well with the spirit of science.*

Share efforts?



Potential topics for joint efforts:

- Best practices for software preservation & reproducibility (community specific?)
- Recipes for orchestration of heterogeneous applications on heterogeneous platforms
  - Deployment on clouds like HNSciCloud, EOSC
- Issues with licensing & credentials & authorization
- Provisioning and sharing of images
  - Trust
  - Catalog (app store), maintainer, credits
- Capability discovery & well-defined vocabulary
- data access & exchange (HDF5 Rest API, h5pyd?, Frictionless Data?)

- Most of it might already exist
  - e.g. anchore, fleet, jfrog, swarm, kubernetes, shipyard, dockerui, autodock, compose, flocker, weave, helios, but …
- … exchange of knowledge, expertise & experiences would be greatly beneficial

- Intend to organize a workshop with Docker/shifter experts
- Intend to apply for funding

Fin

- **Good enough to spin off activities??**
- **Probably not really an RDA biz??**
- **Workshop/Collaborations/KE??**
- **Grants??**
- **Best ways to stay in touch and exchange experiences?**
- **Feedback?**
- **Suggestions?**