Deployment of containers on the diverse ATLAS infrastructure
Containers integration

• ATLAS can run containers in multiple ways depending on the site configuration and the user workflow
  • The pilot runs the containers getting the software from CVMFS
  • The pilot runs a standalone container with all the software in it
  • Nested containers i.e. the pilot runs a container in the batch system generic container
    • Sites running non RHEL OS
  • Site runs ATLAS containers as part of the batch system
    • HPC
• Aim is to keep it as uniform and flexible as possible
Containers integration

- The first step towards containerization was to add a thin container layer between the pilot and the transform.
  - Workflow for production and users is identical in this case.
  - Transparent for users don't even realise their payloads run in containers.
  - Single and Multi-core payloads.
Containers integration

- To allow users to use their containers as they are we rearranged things and created a container transform rather than a payload transform
  - Payload has all it needs in the container
  - Containers get downloaded from the registries
    - Image distribution problem (more later)
Images

- All ATLAS images are docker images
- They are mostly built as a hierarchy of docker layers
  - ML on top of docker library images
Type of images

- Images in CVMFS used by all standard jobs
  - Base images
  - Middleware images
- User standalone images in docker/gitlab
  - Analysis and ML software
    - Usually ~ 1 GB compressed
- HPC fat images
  - Used to be multi-release but these are difficult to distribute to multiple HPC sites
  - On the user images model single release + application
    - ~7 GB compressed
    - Poster: https://indico.cern.ch/event/773049/contributions/3473850
Grid deployment

- Deployment of ALRB containers
  - Software from CVMFS and jobs configured by ALRB
- All CentOS7 queues
- Production and analysis

ATLAS grafana job plots
Sites with ≠ Linux

- SLE, CLE, ubuntu, EL8 on some sites soon....
- Run their own containers as part of the batch system to run the standard jobs
  - Docker could run nested singularity for a long time
- Used to be a problem at sites using singularity for payload isolation until this summer
  - Now moved all sites to nested singularity
- Running images from /cvmfs we are in control of both the image the batch system uses and that of the payload
  - This works well also at some HPC sites like SuperMuc and CSCS
HPC

- NeRSC cori – sw distribution problem
  - Import experiment images on a shared file system
  - Get the batch system to run them
  - Multi-release images installed once every several months
- Multi-release images ~200GB difficult to distribute
  - Number of HPC increasing need to improve sw dist agility
  - ALCF, Tokyo, Mare Nostrum, DESY HPC, MPPMU, SuperMuc...
Users standalone containers

- Important for flexibility and reproducibility (analysis preservation)
- Can run with minimal interaction with external environment
- Can run these containers by using the same command line as standard jobs
- System can take both images from the registries and from /cvmfs
  - Average time to download an analysis release image and build a sandbox is 2 minutes
    - Sandbox can be reused by following payloads
- First non expert (ML) user run successfully ~24k jobs
Registries usage

- Docker hub so far hasn't complained but...
- Still... need to mitigate transfers if more users go for standalone containers
- 3 solutions
  - Use a Frontier like system of squid caches
  - Add images to /cvmfs
    - /cvmfs/unpacked.cern.ch
  - Get the runtimes to combine layers from multiple sources when they build the rootfs file system [https://github.com/google/crfs](https://github.com/google/crfs)
Isolation policy

- Mount namespace: VO pilots MUST isolate files of other user payloads and their own files from the user payload by building a custom mount namespace which only exposes parts of the file system used by the current user payload and nothing else.
- Process ID (pid) namespace: VO pilots MUST isolate processes of other user payloads and their own processes by creating a new pid namespace dedicated to a single user payload.
- Interprocess Communication (ipc) namespace: When possible, VO pilots SHOULD create a new ipc namespace dedicated to a single user payload in order to isolate communications of other user payloads and their own internal ones.

- Policy only for user containers not for production
- ATLAS container deployment is almost compliant
  - Batch systems containers not a problem anymore: nested containers
  - Direct I/O pilot uses pilot proxy to access the storage
    - It doesn't interact with users personal proxies
    - Unprivileged user robot proxy: under implementation
Runtime deployment

- User namespaces
  - Unprivileged
  - Rootless
  - demonless
- Exec from CVMFS
  - Uniformity

- System gives flexibility to develop appropriate plugins
  - Description is queue dependent

```
container_type: "singularity:pilot"
container_options: "-B /mnt/lustre_2 --nv"
```
Conclusions

- ATLAS is now using containers on all its payloads
- The system is flexible enough to accommodate a combination of different sites and user requirements
- The system is also flexible enough to introduce different runtimes from singularity
- ATLAS is working on solutions minimizing access to the registries for the distribution of standalone containers
- ATLAS is also compliant with the WLCG isolation policies.
Backup
Containers integration

- **Standard job**
  - Payload
  - Transform
  - CVMFS/ALRB

- **Standard job in batch system container**
  - Payload
  - Transform
  - ALRB/CVMFS

- **Standalone container payload nested in batch system container**
  - Container
  - Transform
  - Pilot2

- **Containerized payload**
  - Payload
  - Transform
  - OS container
  - CVMFS/ALRB

- **Containerized payload nested in batch system container**
  - Payload
  - Transform
  - OS container
  - ALRB/CVMFS

- **Standalone container payload nested in batch system container**
  - Payload
  - Transform
  - Container
  - Pilot2
  - Batch system container