NRC·CNRC

Image Metadata in Container-based Science Platforms

IVOA Interop, Malta, Nov 2024

Brian Major[1], Saurabh Mookherjee[2], Rajesh Tamhane[2], Dustin Jenkins[1], Shiny Brar[1], Sharon Goliath[1], Sebastien Fabbro[1], Chris Willott[1], Toby Brown[1]

[1] Canadian Astronomy Data Centre, NRC
[2] Thoughtworks India

Agenda

> Background and Objectives
> Image Discovery
> Interactive Container Execution
> Summary and Questions

Container-based Science Platforms

- Definition: Containers vs Images vs Software?
- Definition of Container-based Science Platforms:
  - OCI (eg Docker) containers are the 'code' in 'code-to-data'
  - Containers are pulled to the compute environment and executed
  - I/O to target data is optimized in the compute environment
- Examples: CANFAR, Rubin Science Platform, ESA DataLabs, SciServer, Rosetta, and others…

Why strive for Interoperable Platforms?

- Healthy Implementation Diversity
  - Unrealistic to expect same platform technology and implementations
  - Standards and API based approach allows for individual operational and infrastructure management and optimizations
- Focus on Science
  - Sets the "platform" for astronomy software development: containers.
  - These containers can be shared amongst the entire astronomy community
  - New projects have a head start – just deploy a standard platform.

CANFAR

- Executes containers in Kubernetes - interactive and batch modes
- Handles all Authentication, Authorization, Identity
  - Single OpenID Connect login to portal
  - POSIX uids / gids mapped to containers and storage at runtime
- Astronomy/research specializations:
  - GPU scheduling, CARTA, Firefly (soon)
  - Applications: CASA versions 3.4.0 - 6.3.3, TOPCAT, Aladin, DS9, Visivo, etc..
- Workflow support coming
- Helm chart installation aims to allow for infrastructure variance

Use Cases for Interoperability

- Science Reproducibility
  - Want to share and use the exact same software (the container images) on data in different locations (different instances of science platforms)
- Enable Container (and associated software) discovery
  - Should be able to find and query for containers based on certain criteria (image metadata)
- Consistent User Experience, and support Distributed Programmatic Execution
  - Don't require container or execution modifications for specific platforms
  - Allow containers to be hosted at different image registries (eg dockerhub, gitlab, harbor)

Aspects of Platform Interoperability

- Authentication and Authorization
- Specifying container resource requirements (RAM, GPU, etc…)
- Expectations of data location and access
- **Allow for the discovery and selection of containers/images across registries**
    - Alternative: CVMFS – not discussed here
- **Compatible container execution across platforms**
    - Batch Execution - easy: command and arguments built in, or provided
    - **Interactive - more difficult…**

Image Discovery

Image Discovery from the CANFAR Portal

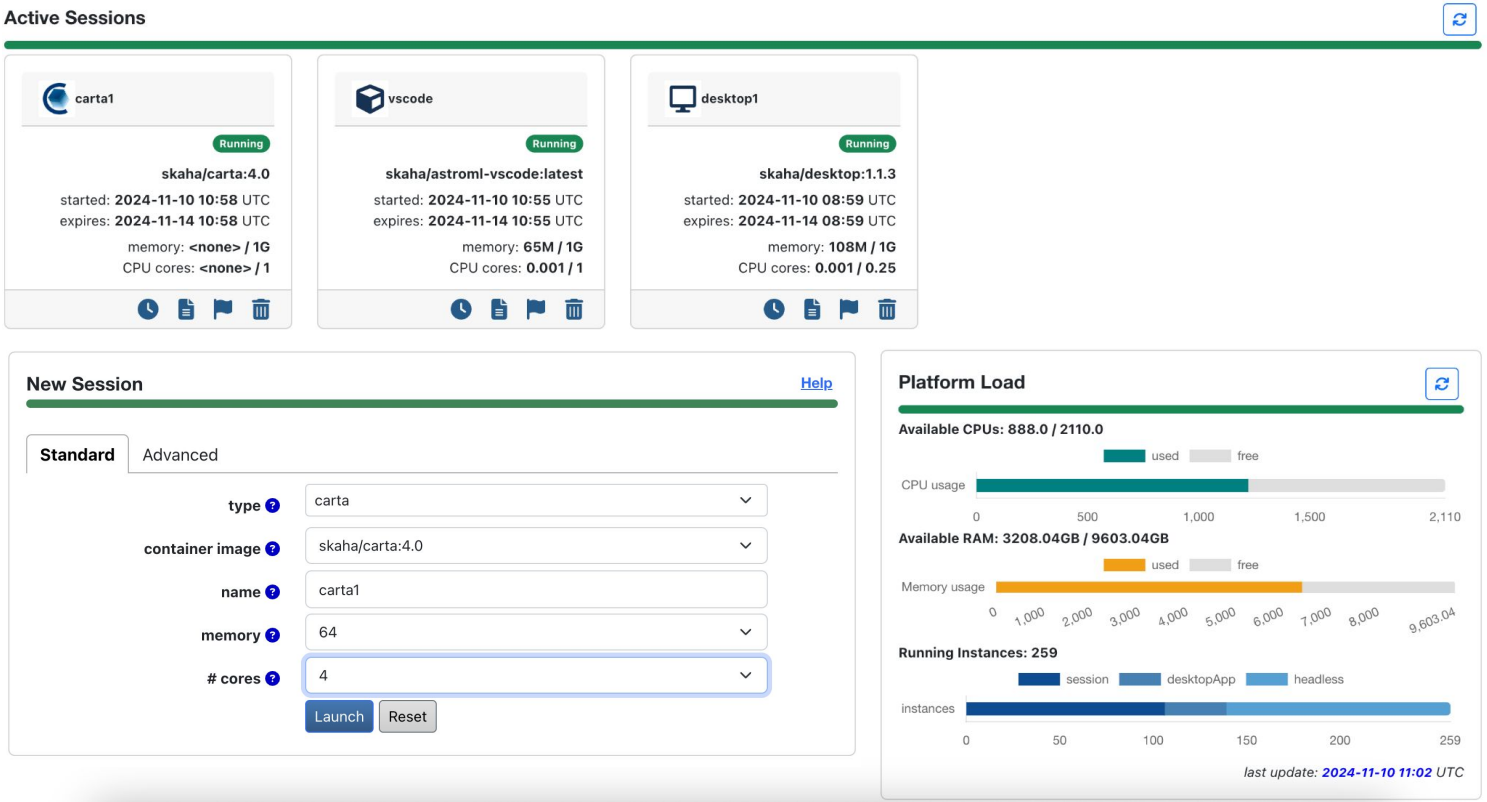


Image Discovery from the CANFAR Portal

Standard image uri format contains some info:

`host/[namespace]/repository:tag`

But not enough, also need:

- description
- author
- software
- use and data format
- etc…

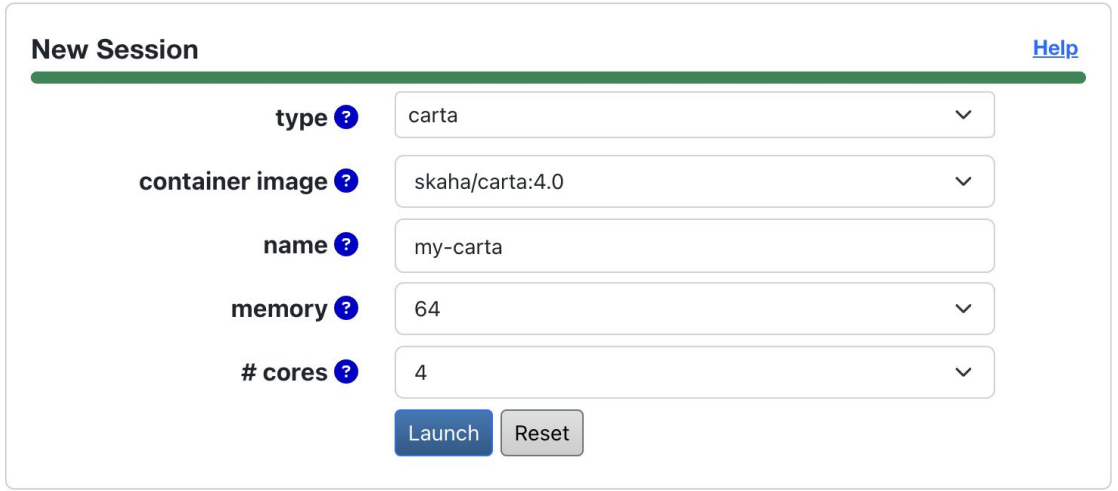


Image Listing through Registries

- Looking for something like the search functionality on PyPi, but for containers.
- Open Container Initiative (OCI) Distribution Specification (REST API and command line):
  - Optional API Extension supports listing of images in 'repository'
- Image Listing supported (HTTP REST APIs) by:
  - harbor
  - quay.io
  - docker registry v2 API - an optional extension to the API.
- But all slightly different… projects vs catalogs vs repositories
  - Limited metadata returned
  - Not queryable

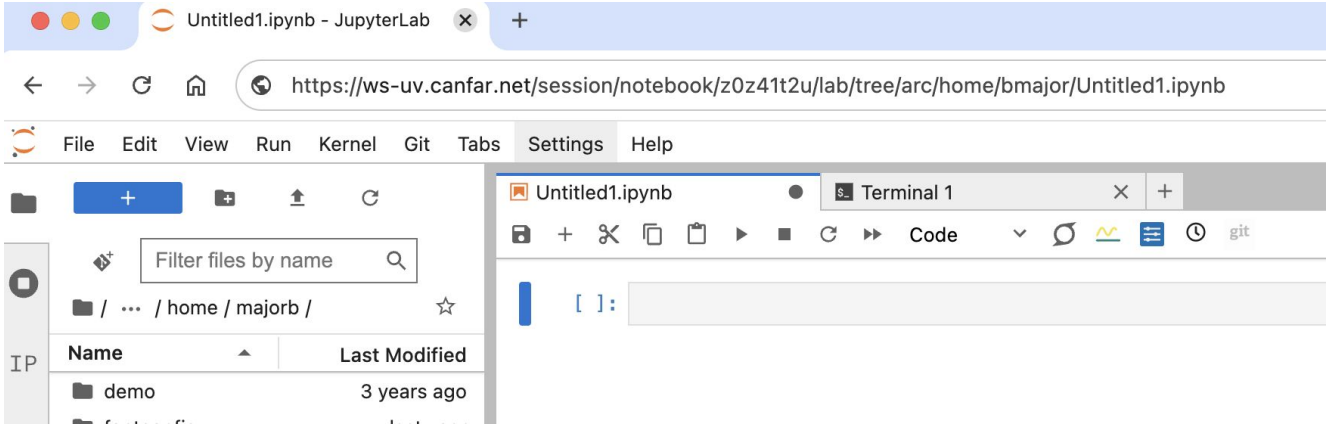
(Single) Image Metadata through Registries

If you know the image URI…

- Standards and tools around querying metadata for a single image
  - Two standards: OCI Image Registry, Docker v2
  - Tool **skopeo**, can inspect metadata from images hosted in all repositories
    - (but can't list images…)
- Metadata can be added to images in different ways: ENV vars, image labels, manifest annotations
- Existing images can be modified to include metadata
  - Build time with metadata in Dockerfile (ENV vars, labels)
    - Build time (manifest annotations)

Interactive Container Execution

User Experience from the CANFAR Portal



https://ws-uv.canfar.net/session/notebook/z0z41t2u/lab/...

Interactive Container Execution

- Docker comparison
  - '`docker run -it`' – but web access, not shell access
  - or '`docker run`', and access URL just shown in log output
- (CANFAR) Science Platforms need:
  - Access URL and path
    - some web tools can figure this out dynamically
    - some web tools need to be told where they will be running
    - some need to run without a path! (cannot support)
  - Port(s)
    - to discover the port(s) being used; or
    - to specify the port being used (doesn't work when multiple)
- **Sometimes this info can only specified on container startup**

'Type' identifies container characteristics

- Not needed for batch job execution - use Dockerfile CMD or API param
- Characteristics defined by 'Type':
  - The startup script – often controls the path and port
  - The ingress rules – external to internal URL rewrites
- "Contributed" type
  - Attempt to standardize expectations of port and path of interactive containers
  - Works out-of-the-box for some (vscode, pluto, …)
  - Does not work for others (JupyterLab, NoVNC, …)

Summary and Questions

Summary

- Discovery
  - Listing images - **no standard** is registry APIs
    - Most implementations support it over HTTP
    - Complex, one-time querying not possible anywhere
  - For **single** images, a number of OCI options exist for **adding** metadata
- Execution
  - A **custom startup** script is sometimes required
  - A **custom ingress** is sometimes required

Questions - Image Discovery

- How can we list and query images metadata across registries?
- Should we use metadata in the OCI specification?
  - Enough to support an Image metadata model?
- Or should metadata be decoupled and made available through TAP?
  - If so how do you reference the images, and prevent them from being disconnected?
  - Like VO data/metadata approach, but more volatile and less curated?

Questions - Interactive Execution

Where to put the details of startup and web access?

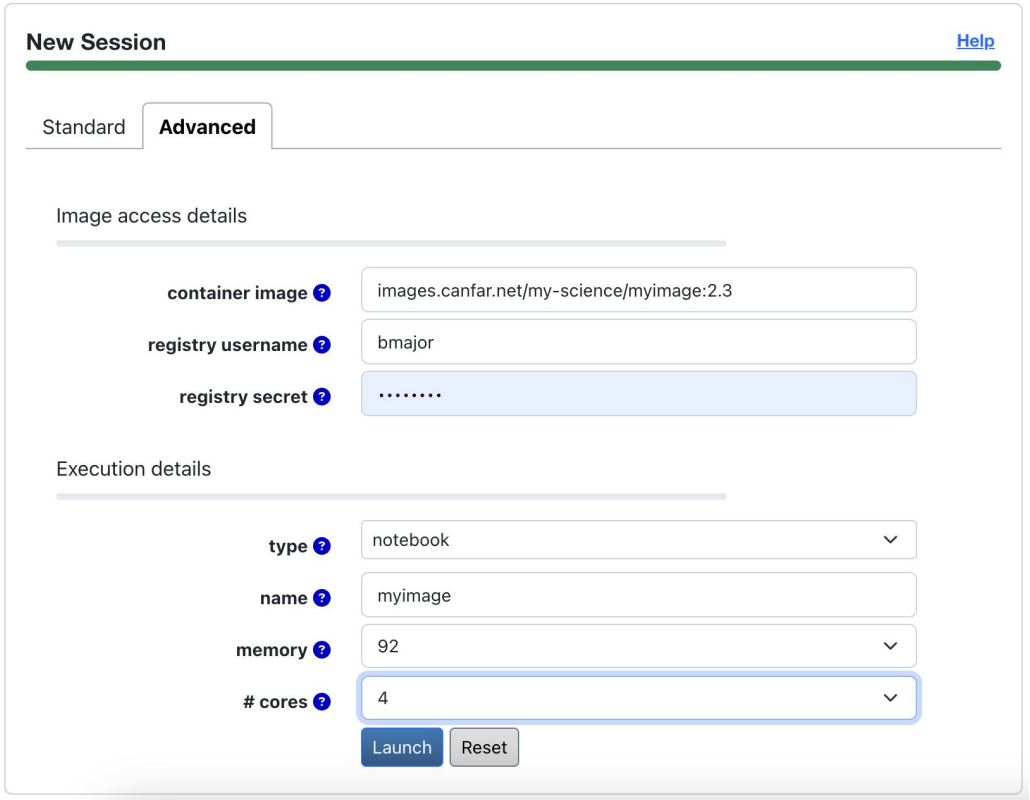
Options:

1. Well-known set of **types**, param provided by users (current CANFAR model)
2. Startup script and access **information provided by users/clients**?
3. **Rules** on how interactive containers are built?
    a. "Contributed type" - Only works for some, cannot influence container builds from other research communities (eg Jupyter)
4. **Metadata attached** to containers specify startup and access rules?

Prototyping General Image Access in CANFAR

For now…

- No listing from repo
- Metadata coming
- Supports all registries
- Supports proprietary image access
- 'type' remains and is supplied by user



Thank you

brian.major@nrc-cnrc.gc.ca