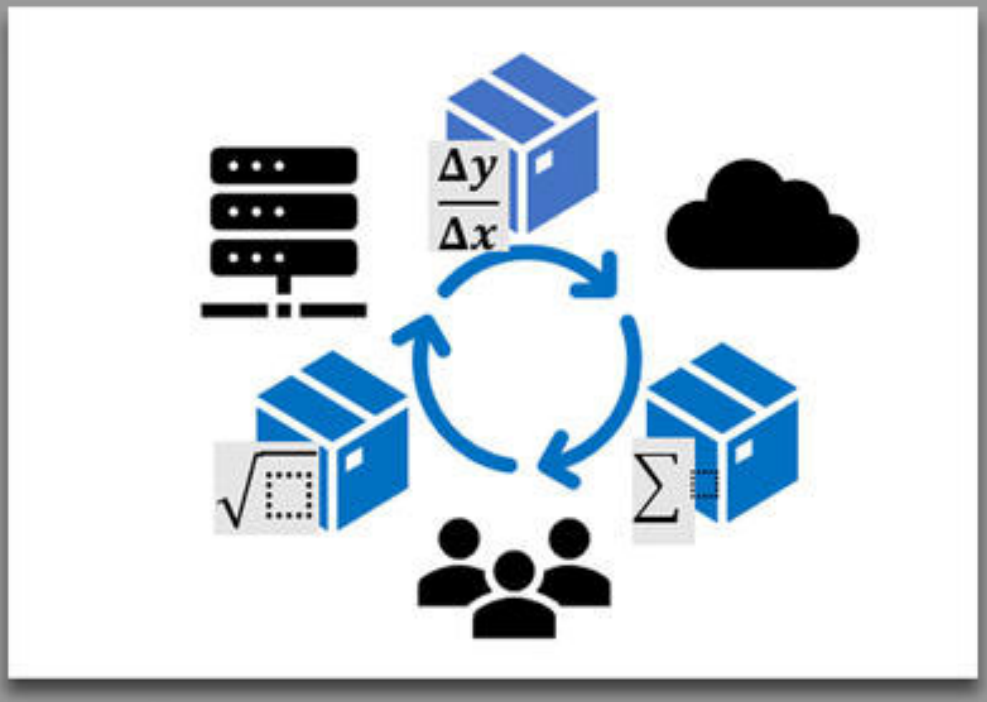
# 2nd International Workshop on FAIR Containerized Computational Software

***December 5-7, 2023***




NIST | NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
U.S. DEPARTMENT OF COMMERCE



# Welcome to the Day Three of the 2nd International Workshop on FAIR Containerized Computational Software

*Co-organized by National Institute of Standards and Technology (NIST) and National Center for Advancing Translational Sciences (NCATS) at National Institutes of Health (NIH)*

*Peter Bajcsy*
*NIST*

*Nathan Hotaling*
*NCATS NIH*

# FAIR Containerized Computational Software

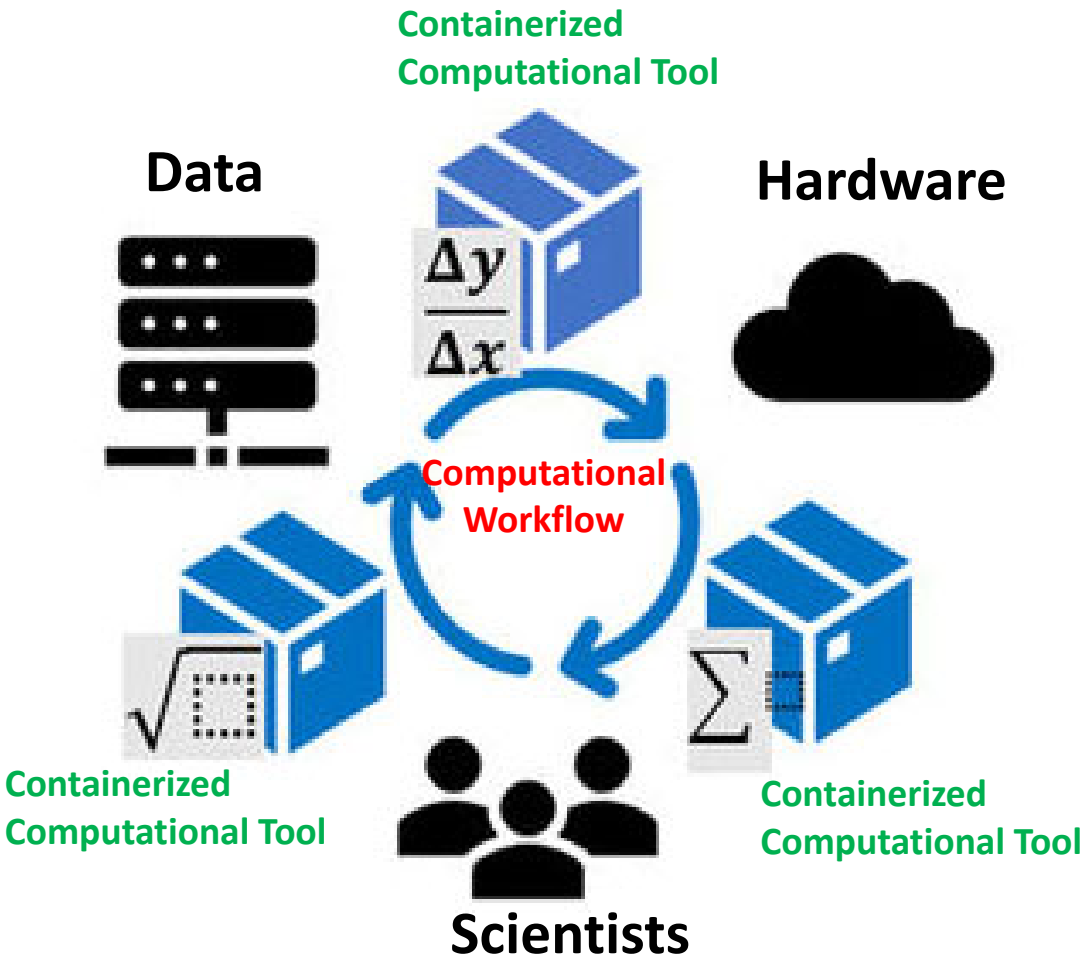
**Digital assets:** computational software in sciences

**Containerization:** packaging of software applications so that the software can run in any computational environment

**FAIR:** Findable, Accessible, Interoperable, and Reusable

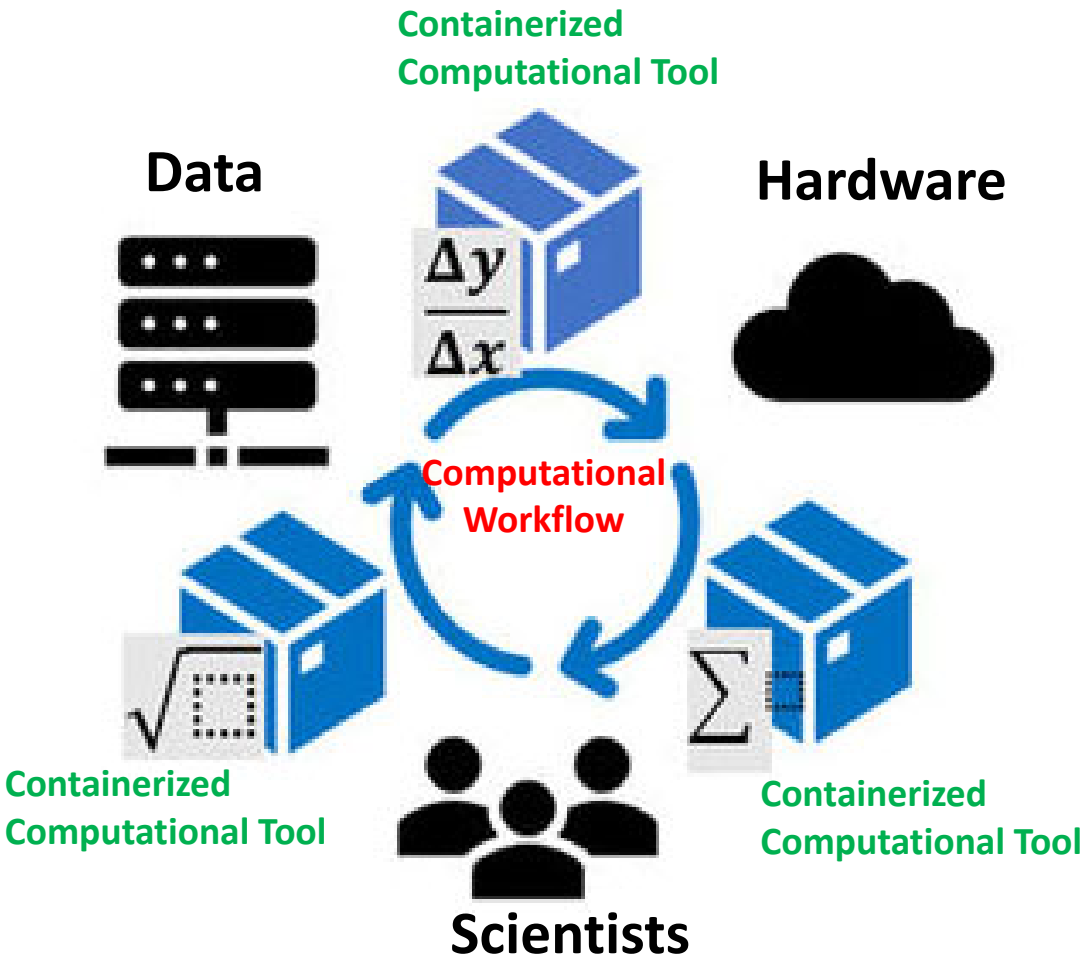
# Main goal

*The main goal* for the workshop is to establish a community consensus on creating interoperable containerized computational tools that can be chained into scientific workflows/pipelines and executed over large image collections regardless of the cloud infrastructure components.

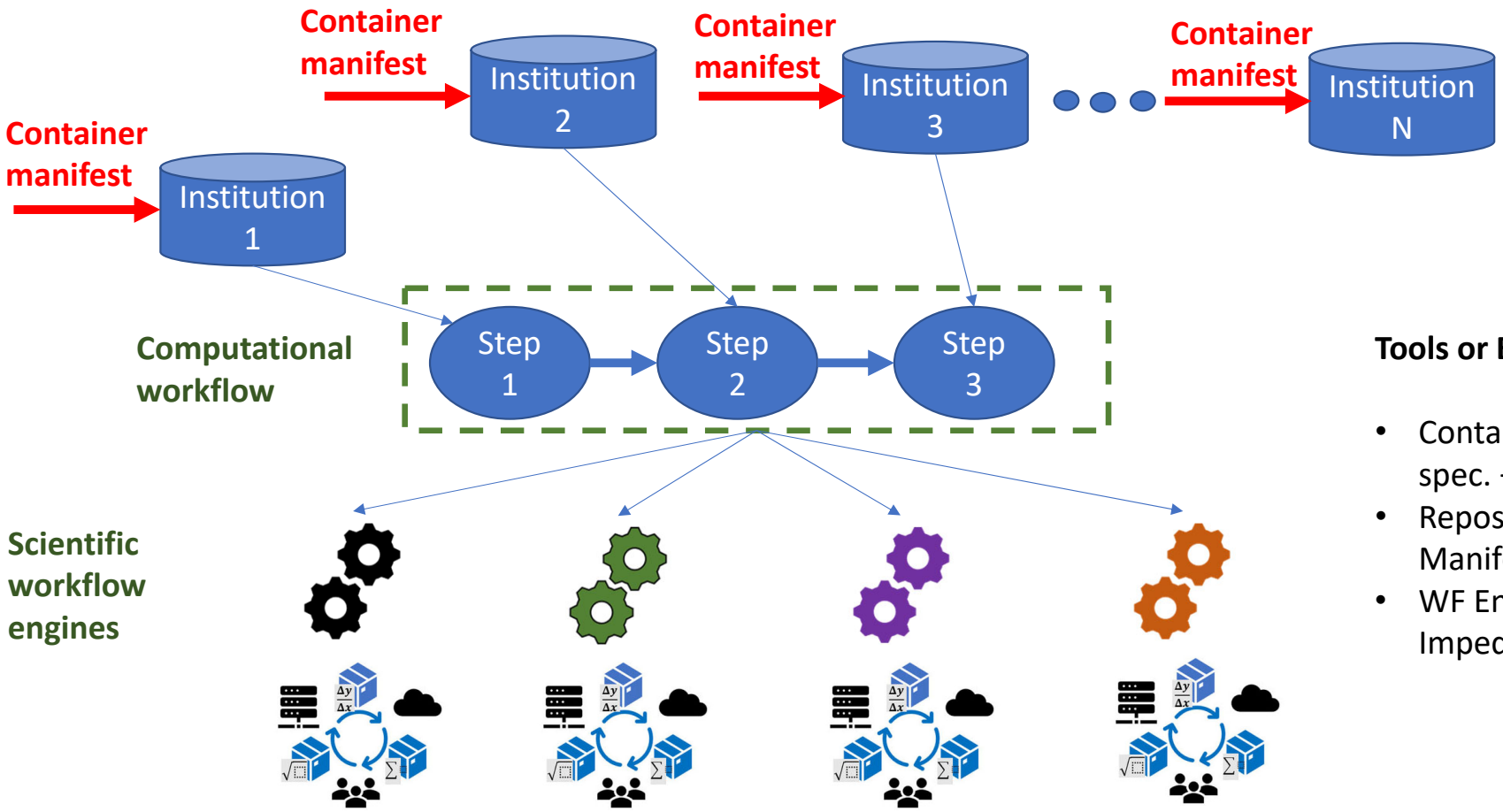


# Approach to Forming Computational Workflows

- Create a manifest file accompanying each containerized software tool
- Specify fields in the manifest file for
  - **Inputs/Outputs**
  - **Security**
  - **GUI**
  - **Hardware Requirements**



# A Light at the End of the Tunnel




**Tools or Eco-system:**

- Container manifest spec. + Tools
- Repository with Manifest & WF + API
- WF Engines + SW/HW Impedance Matching

# Structure of the 2nd International Workshop on FAIR Containerized Computational Software

# Workshop Structure

## Themes for each day:

December 5 (Day 1): Inputs/Outputs and Security for FAIR containerized computational software

December 6 (Day 2): Graphical User Interfaces for FAIR containerized computational software

December 7 (Day 3): Hardware Requirements for FAIR containerized computational software

## Top level program outline (Each day):

Session 1: General session consisting of opening remarks and background introduction (one hour)

Session 2: Five breakout sessions consisting of about 15-30 people discussing specified topics (two hours)

Session 3: General session consisting of summaries from breakout sessions and closing remarks (one hour)

# The Workshop Flow

- Session 1: Main Zoom room
  - After the introduction to the theme of the day, NIST conference facility staff will move participants to breakout
- Session 2: Breakout Zoom room
  - Moderators and scribes will go over a set of questions/topics to be discussed. The topics map to the those posted at Federal Registry. The set of questions/topics is the same for all breakouts.
- Session 3: Main Zoom room
  - The moderators and scribes report a set of unique answers/solutions for each question/topic and the results of polls

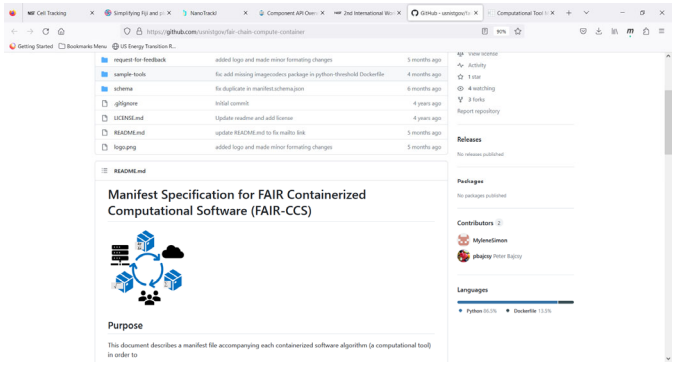
# Time and Information Resources

## Everyday Meeting Times:

US East Coast (11am-3pm),
US West Coast (8am-noon),
UK (4pm-8pm),
Germany (5pm-9pm),
Korea/Japan (1am-5am)


**Comments on the topics outside of the workshop:**
https://www.federalregister.gov/documents/2023/08/24/2023-18263/request-for-information-regarding-file-specification-for-findable-accessible-interoperable-and

**Workshop event URL**: https://www.nist.gov/news-events/events/2023/12/2nd-international-workshop-fair-containerized-computational-software




**The GitHub repository with the current specification:**
https://github.com/usnistgov/fair-chain-compute-container

**Registries of manifests adhering to the current specification:**
https://wipp-plugins.nist.gov/
https://wipp-registry.ci.ncats.io/

# The Workshop Information

- During the workshop:
  - The workshop is not recorded.
  - The summary notes and chats from breakout sessions will be used for preparing the workshop report.
- Additional Topics
  - Use cases
  - Ecosystem of tools to support seamless integration to scientific research
  - Educational materials
  - Benchmarking information
  - Governance

# Quick Summary from December 6, 2023 (Day 2)

# Theme: Graphical User Interfaces (GUI) for FAIR containerized computational software

# Quick Summary: Q1

1. What is your experience for file type or service for containerized computational workflows?

- **Requirements:**
  - Open frameworks and APIs to help with adoption
  - Needs to work offline for HPC
- **Format considerations:**
  - The specific format is not as important as the tooling around it
  - Flat files can take more time to validate, want to confirm variables are correct before submission
  - The specification is more important than the format itself
  - Dated formats need to be supported because legacy tools still use it

# Quick Summary: Q2

2. What are the basic types of algorithmic parameters?

- **GUI in manifest?** UI considerations should not appear in the specification Versus UI for parameters, outputs, and intermediates
  - Human in-the-loop is the boundary between workflows
- **GUI Complexity:** Non-trivial params passed via raw files since we cannot process every possible input
- **GUI Representation:** Use a markup language to define the supported basic types and show a preview of how they look like in a given GUI
- **GUI Ontologies:** The UI should use ontologies to better describe the content of each step.
  - A simple Graphic Tool that allows the user to run a workflow online.
  - Need for validation following a standard ontology (RO-Crate, EDAM, Bioschemas, GA4GH WES)

# Quick Summary: Q3

- 3. What is your experience with the languages and libraries used for on-the-fly (on-demand, dynamic) GUI creation?
- **Languages/Libraries:** Swagger Docs, FastAPI micro services in Python, JavaScript server in Node.js, TypeScript/Django, PyQT, JSX, Vega-Light; ReactJS
  - swagger does not support conditional parameter input logic yet
  - OpenAPI: https://github.com/OAI/OpenAPI-Specification/issues/256
- **From library-based code to Web:** Galaxy interprets the XML; From JSON to UI; (JS, Angular, Node.js) -> Uses a NGX Schema to Web form; R to DashApp to HTML; Wanted: Python Notebooks to Typescript app
- **On-the-fly execution:** Specify library-based code location in manifest and how to launch it

# Quick Summary: Q4

- 4. What is your experience with tools to automate generation of manifest sections describing conditional GUIs for encoding complex logic into static web forms for conditional user interfaces?
- Lesson learned:
  - KNIME (server version, local version), build custom GUI from primitives
  - Label studio - annotation library, XML → HTML; https://labelstud.io/playground/
  - Flask with Jinja2 (Web Server Gateway Interface with templates)
  - Pixme from Google Apps (Android)
  - streamlit-jupyter – create interactive GUI inside of Jupyter Notebook (alternative to voila)
  - Users can inject JS in the manifests which present a security issue.
- Recommendations:
  - Translate DAG to JSON (or other format) and starts from the JSON to generate the GUI
  - GUI tools needs to be able to handle conditional logic
  - Manifest should not be too specific and tied to specific GUI

# Quick Summary: Q5

- 5. What command line interfaces (CLI) are of interest for creation of GUIs?
- **Polus:** Workflow builder https://github.com/PolusAI/workflow-builder
- **Napar**i allows the user to create modules that can describe the GUI.
- In **Matlab**, if you define a GUI you can create pop-ups using CLIs.
- **Textualize**: they have tools https://github.com/Textualize
- **TxTk, WxWidgets**: Allow for testing before launching workflows in the production environment.
- **Python**: argparse
- **JSON** schema
- **Nextflow** - If task fails in CLI, inspect the log to triage the issue
- **SnakeMake** - https://snakemake.readthedocs.io/en/stable/ - Python based workflow management system

# Quick Summary: Q6

- 6. What web application programming interfaces (web API) are of interest for creation of GUIs?
- **Use case:** Progress report GUI for large scale simulations
  - Web page would be parsing outputs and with a template it would inform end users about the progress (elastic search like)
- **Creation of GUI by using libraries and services:**
  - Ability to submit through a queue / service
  - https://www.tensorflow.org/tensorboard
  - https://wandb.ai/site
  - There is a tool that will take GraphQL schema and create a frontend for it.
  - Airtable Connected Apps Platform can create apps for non-technical users.
- **Forward looking support:** LLMs can be of use to generate web API

# December 7, 2023 (Day 3)

# Theme: Hardware Requirements for FAIR containerized computational software

# Agenda

- Session 1: Introduction
  - Peter Bajcsy (NIST) –summary of Days 1 & 2 and the goals for Day 3
  - Tim Blattner (NIST) –current hardware specification and the needs at NIST
  - Nick Schaub (NIH) –hardware heterogeneity at NCATS NIH
  - Derek Juba (NIST) –hardware heterogeneity at NIST
- Session 2: Breakouts

| Date | Breakout | Moderator | Scribe |
|---|---|---|---|
| 7-Dec | 1 | Timothy Blattner | Antoine Gerardin |
| 7-Dec | 2 | Derek Juba | Andrew Toler |
| 7-Dec | 3 | Sameeul Samee | Michael Majurski |
| 7-Dec | 4 | Nick Schaub | Timothee Kheyrkhah |
| 7-Dec | 5 | Mylene Simon | Nathan Hotaling |

- Session 3: Summary

# Questions for the Breakout Sessions

1. What is your experience with the benefits of including hardware requirements?
2. What is your experience with ontologies for computer hardware (existence, coverage, access)?
3. What metadata fields describing hardware can be used for requesting hardware on computer nodes in the cloud or on HPC resources by job schedulers (e.g., SLURM, Kubernetes, CWL command line tool)?
4. What is your experience with metadata fields describing hardware that can be automatically detected by using hardware discovery tools in the cloud or on HPC resources by job schedulers (e.g., SLURM, Kubernetes, CWL command line tool)?
5. What metadata fields describing hardware should be included about the hardware on which a container was tested/validated?
6. What metadata fields describing hardware should be included about all hardware types on which a container has been benchmarked?