Genomic Workflows and Cloud Computing

Jason Walker

BFX Workshop

March 17th, 2025

Stone Soup

Bioinformatics Fast-Food

Bioinformatics Farmer's Market

Bioinformatics Supermarket

Image by ElasticComputeFarm from Pixabay

Ingredients are Data

Utensils are Software

Recipes are Workflows

Appliances are Automation

Compute is our Kitchen

https://pixabay.com/

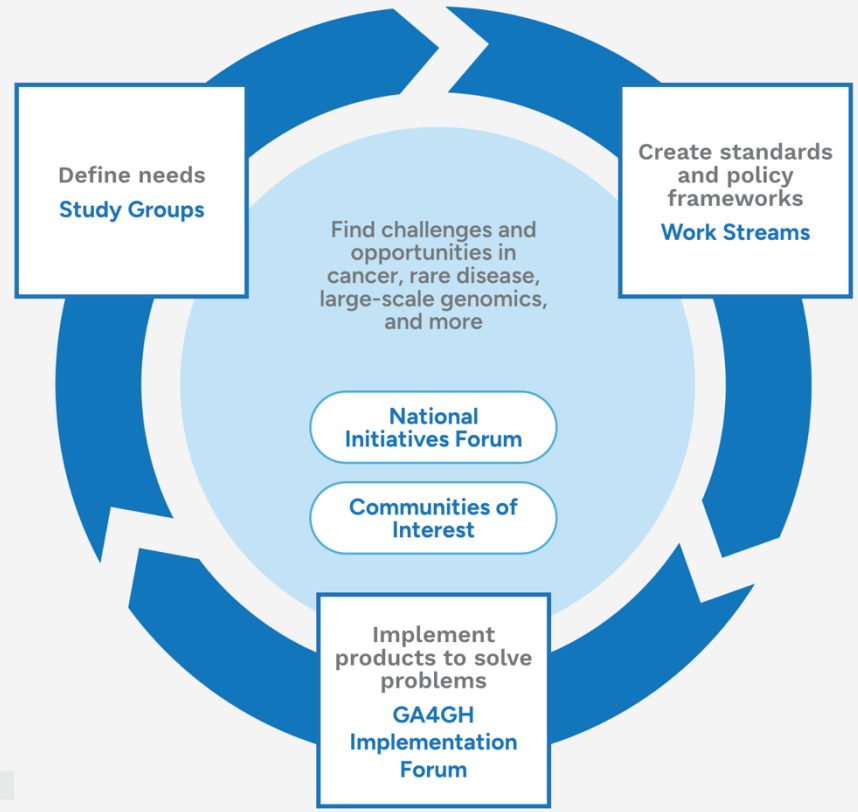
You are the Chef

How we work

Here's our community...

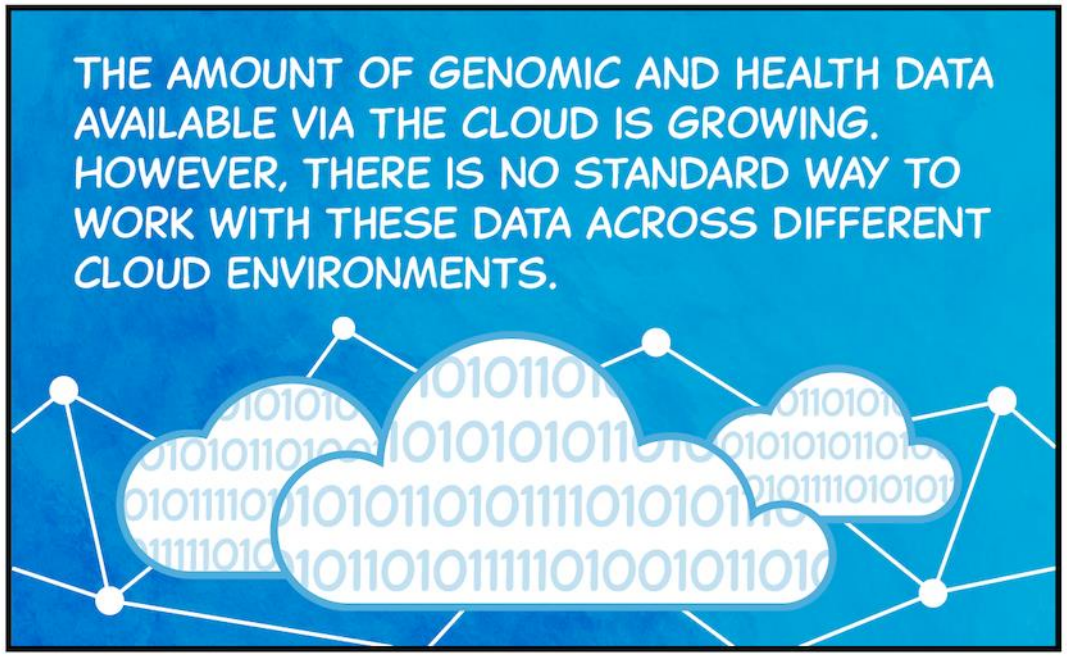
- Strategic Partners
- Driver Projects
- Organisational Members
- Assigned Experts
- Individual Contributors

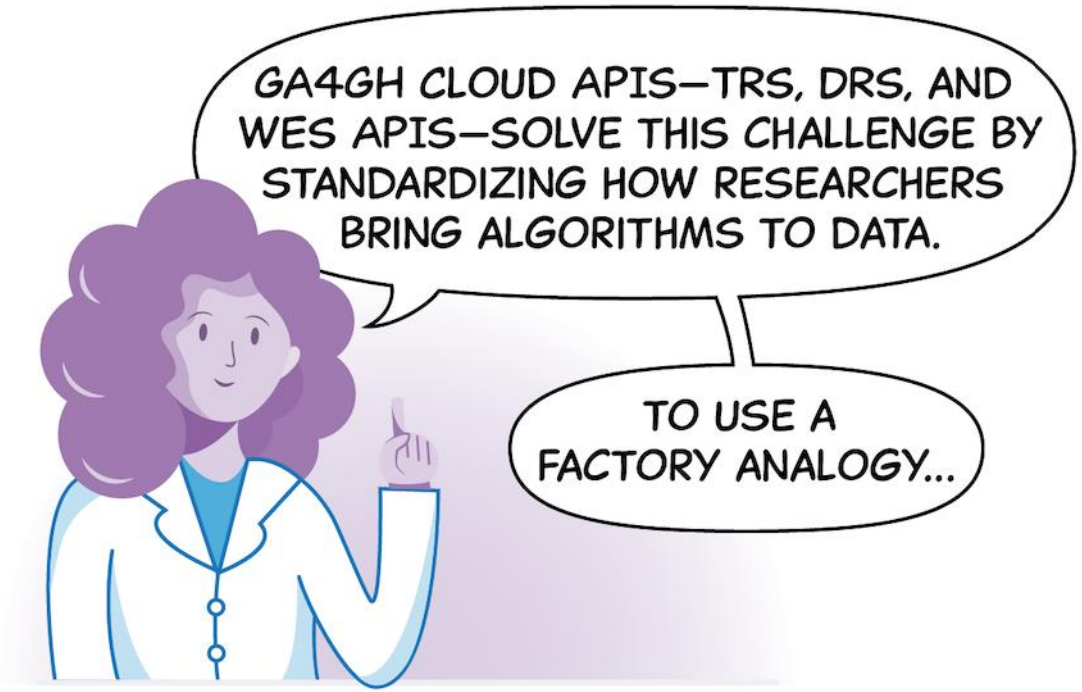
...and here's what we do.



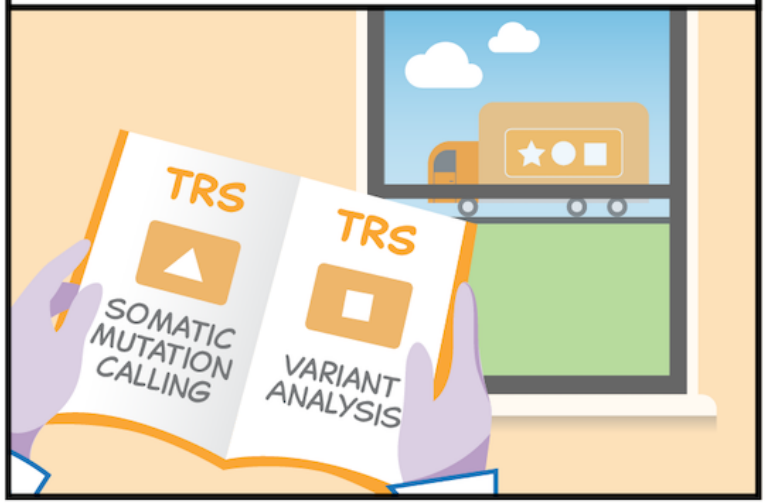
DISCOVER WHAT WE DO

GA4GH CLOUD APIS:
BRINGING ANALYSES TO DATA

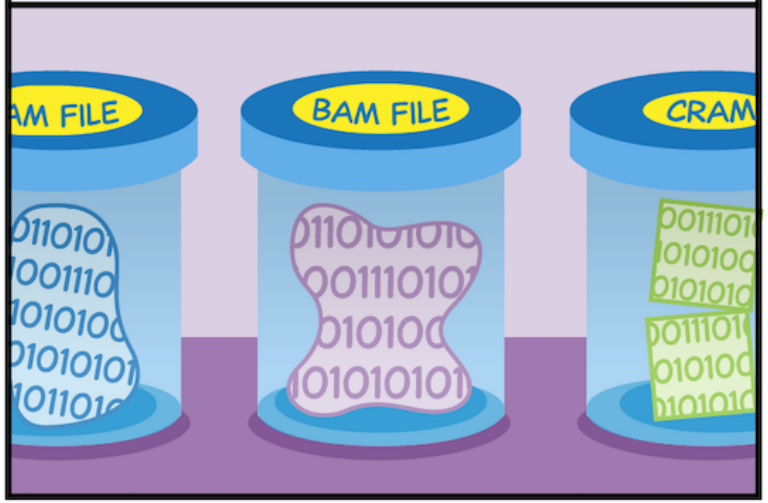




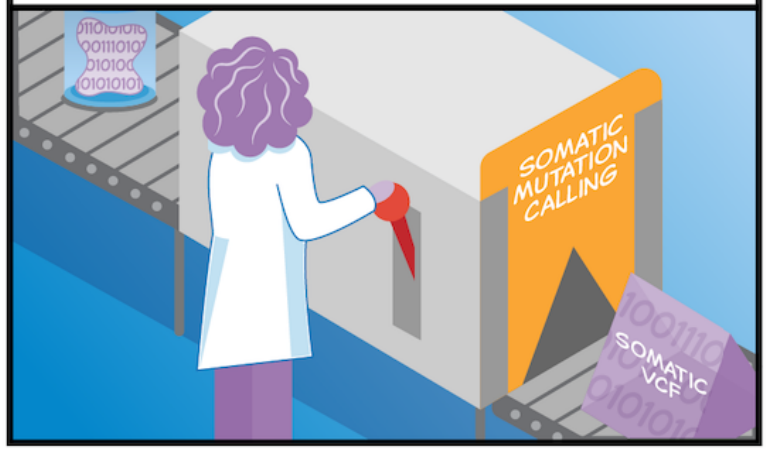
WITH TRS (TOOL REGISTRY SERVICE) API, RESEARCHERS CAN SHARE AND DISCOVER TOOLS AND WORKFLOWS FOR WORKING WITH DATA.



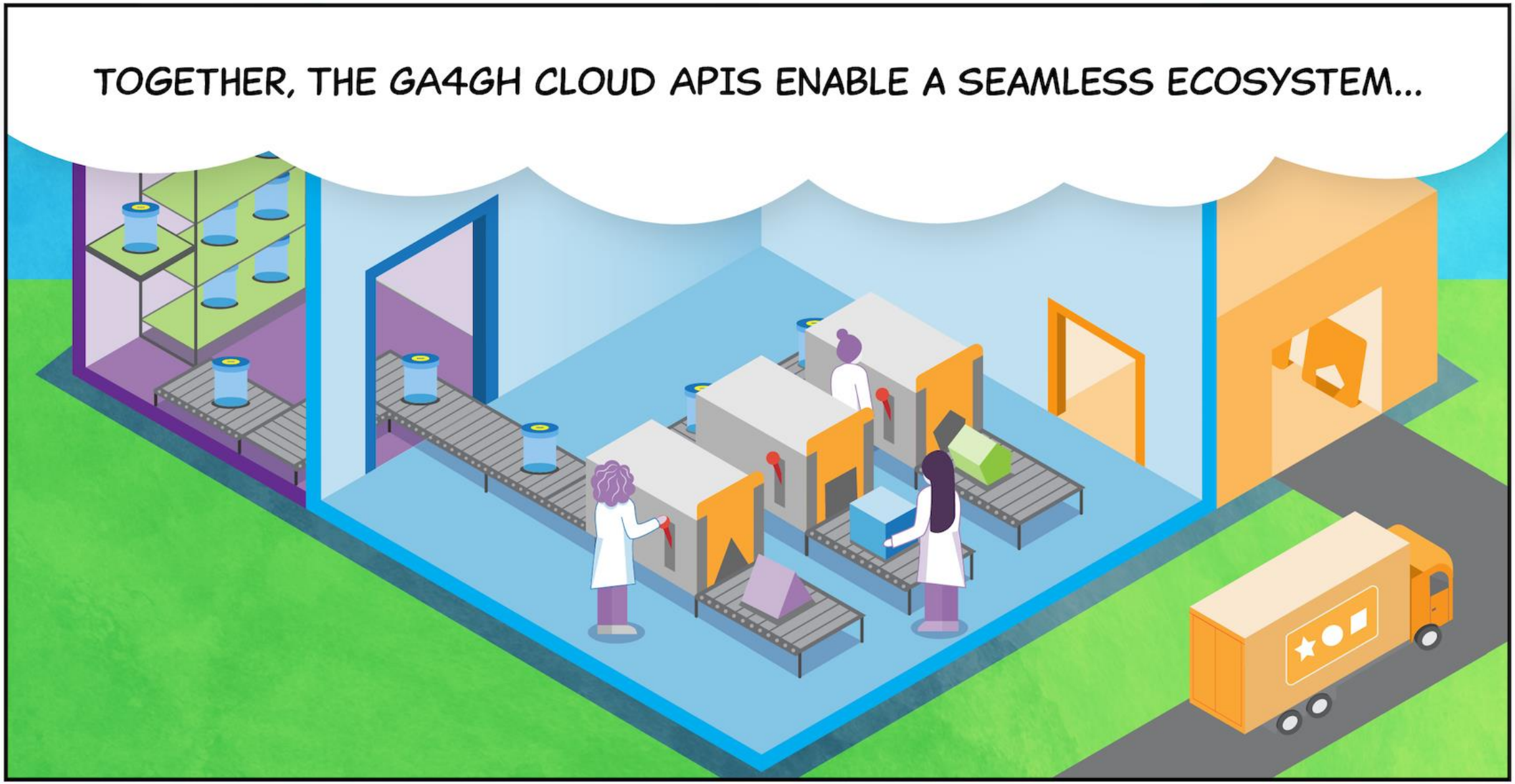
DRS (DATA REPOSITORY SERVICE) API PACKAGES DATA IN A UNIVERSAL WAY, DESPITE HOW THE DATA WAS STORED OR MANAGED ORIGINALLY.



WES (WORKFLOW EXECUTION SERVICE) API ALLOWS RESEARCHERS TO EXECUTE ANALYSES ACROSS DIFFERENT ENVIRONMENTS AND ACHIEVE CONSISTENT RESULTS.



TOGETHER, THE GA4GH CLOUD APIS ENABLE A SEAMLESS ECOSYSTEM...




...ALLOWING RESEARCHERS TO ACCESS, ANALYZE, AND WORK WITH DATASETS IN THE CLOUD.

FAIR4RS - Findable

F: Software, and its associated metadata, is easy for both humans and machines to find.
F1. Software is assigned a globally unique and persistent identifier.
F1.1. Components of the software representing levels of granularity are assigned distinct identifiers.
F1.2. Different versions of the software are assigned distinct identifiers.
F2. Software is described with rich metadata.
F3. Metadata clearly and explicitly include the identifier of the software they describe.
F4. Metadata are FAIR, searchable and indexable.

Barker, M., Chue Hong, N.P., Katz, D.S. *et al.* Introducing the FAIR Principles for research software. *Sci Data* **9**, 622 (2022). https://doi.org/10.1038/s41597-022-01710-x

FAIR4RS - Accessible

A: Software, and its metadata, is retrievable via standardised protocols.
A1. Software is retrievable by its identifier using a standardised communications protocol.
A1.1. The protocol is open, free, and universally implementable.
A1.2. The protocol allows for an authentication and authorization procedure, where necessary.
A2. Metadata are accessible, even when the software is no longer available.

Barker, M., Chue Hong, N.P., Katz, D.S. *et al.* Introducing the FAIR Principles for research software. *Sci Data* **9**, 622 (2022). https://doi.org/10.1038/s41597-022-01710-x

FAIR4RS - Interoperable

I: Software interoperates with other software by exchanging data and/or metadata, and/or through interaction via application programming interfaces (APIs), described through standards.
I1. Software reads, writes and exchanges data in a way that meets domain-relevant community standards.
I2. Software includes qualified references to other objects.

Barker, M., Chue Hong, N.P., Katz, D.S. *et al.* Introducing the FAIR Principles for research software. *Sci Data* **9**, 622 (2022). https://doi.org/10.1038/s41597-022-01710-x

FAIR4RS - Reuseable

R: Software is both usable (can be executed) and reusable (can be understood, modified, built upon, or incorporated into other software).
R1. Software is described with a plurality of accurate and relevant attributes.
R1.1. Software is given a clear and accessible license.
R1.2. Software is associated with detailed provenance.
R2. Software includes qualified references to other software.
R3. Software meets domain-relevant community standards.

Barker, M., Chue Hong, N.P., Katz, D.S. *et al.* Introducing the FAIR Principles for research software. *Sci Data* **9**, 622 (2022). https://doi.org/10.1038/s41597-022-01710-x

Workflow Languages

- Defines a process, including the software, settings, and configuration.
- A portable and sharable representation of a processes structure, dependencies, and execution requirements.
- A standard way to represent inputs, outputs and tools.
- CWL, WDL, Nextflow, Galaxy, and Snakemake are the most *common* ways to represent workflow definitions in Genomics.
- Combined with containerization, e.g. Docker or Singularity, these definitions are portable to many High Performance Computing (HPC) environments including public Cloud, e.g., AWS, GCP, Azure.
- The definitions themselves adhere to FAIR Principles for Research Software (FAIR4RS), making the workflows digital data objects as well as the results.

https://www.commonwl.org/

https://openwdl.org/

https://www.nextflow.io/

https://snakemake.github.io/

https://galaxyproject.github.io/

Workflow Language Characteristics

Aspect	Nextflow	Snakemake	WDL	CWL	Galaxy
Parent Language	Ruby and Groovy	Python-based DSL	JSON/YAML-like	YAML-based	Web-based UI
Compilation	Interpreted	Interpreted	Compiled	Compiled	Managed internally
GUIs/Platforms	nf-core, VSCode, Seqera	Snakemake Workflow Catalog, Sequanix	Pipeline Builder, VSCode, Terra, Dockstore	Rabix Composer, Terra, Arvados, Dockstore	Web-based graphical workflow editor, ToolShed
DSL Features	Complete, extensible in Groovy & Java	Python-based, rule-driven	Limited standard library	Limited standard library, extensible via JavaScript	No DSL (drag-and-drop interface)

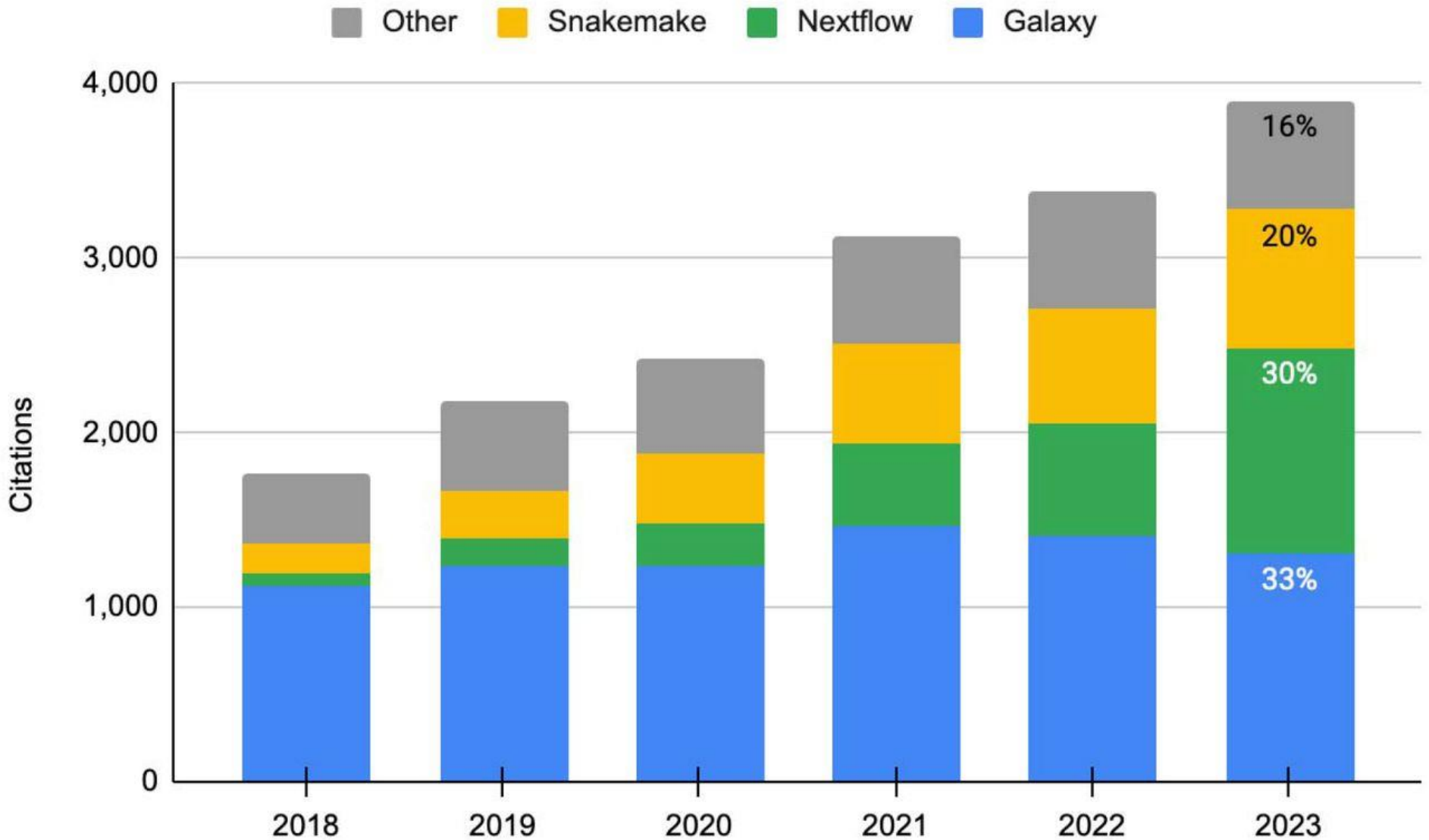
Workflow Language Programming Features

Aspect	Nextflow	Snakemake	WDL	CWL	Galaxy
Variables	Qualified, unique within scope	Named variables in rules	Typed, fully qualified names	Typed, unique identifiers	Managed through UI
Loops	Parallel queue channels	Iterators in rules	Parallel scatter	Parallel scatter via ScatterFeatureRequirement	Implicit (batch processing via UI)
Conditionals	when declaration within a process	Conditional rules using Python logic	If blocks producing optional output types	when and pickValue fields (CWL v1.2)	Implicit via UI options
Parallelization	Implicit (dataflow model)	Explicit (rules & directives)	Explicit (task dependencies)	Explicit (task dependencies)	Managed internally

Workflow Language Usage & Performance

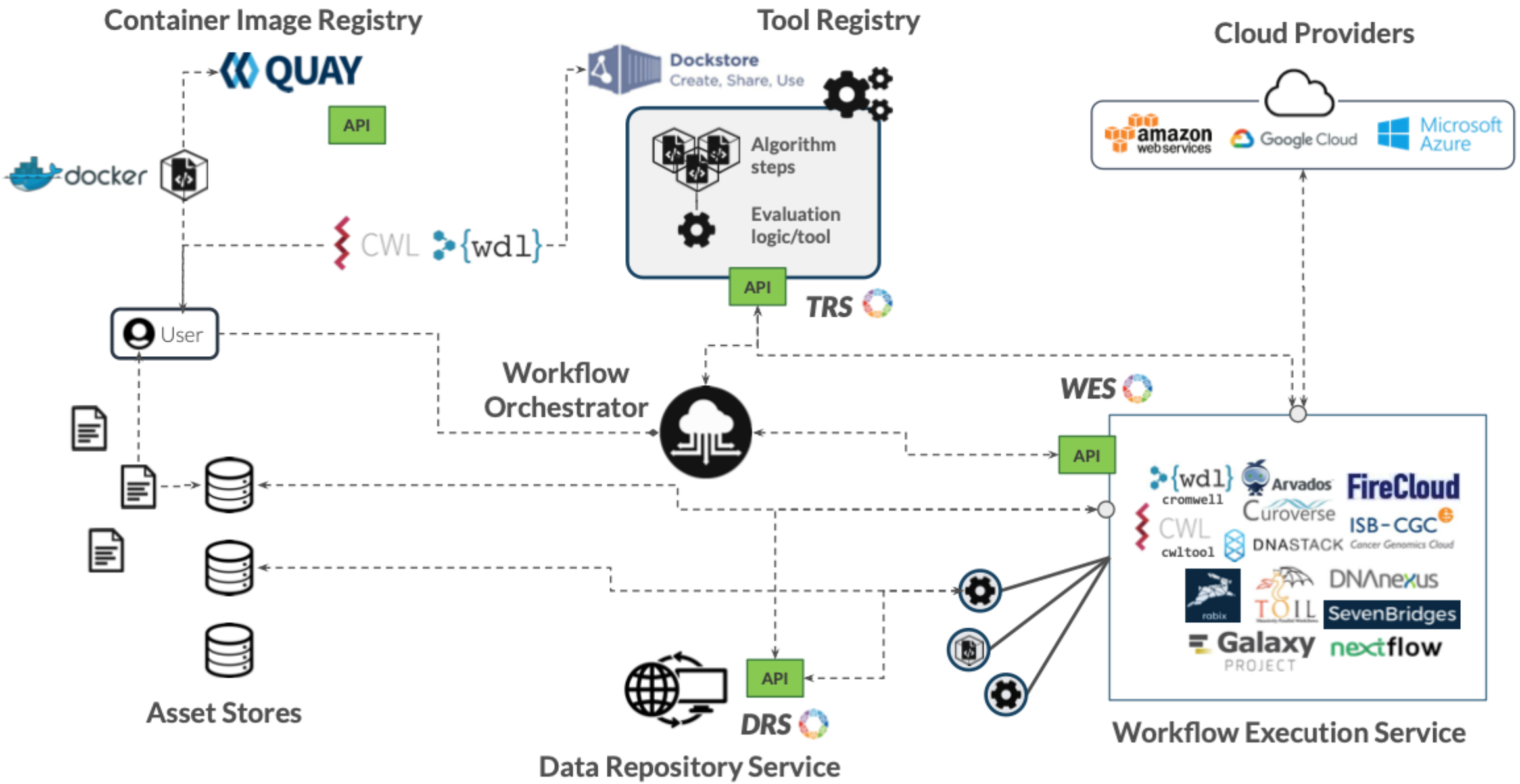
Aspect	Nextflow	Snakemake	WDL	CWL	Galaxy
Portability	Highly portable (Docker, Singularity, AWS Batch, Google Cloud)	Portable, but best suited for local/cluster execution	Portable, requires execution engine (Cromwell, MiniWDL)	Standardized portability via CWL spec	Limited outside Galaxy ecosystem
Ease of Use	Moderate (DSL learning curve)	Easier for Python users	Readable but requires execution engine	Complex due to YAML structure	Easiest (web-based UI, no coding required)
Best for	Scalable, cloud-native workflows	Local and HPC-based workflows	Cloud/hybrid execution needing readability	Highly reproducible workflows across platforms	Non-programmers, community-curated workflows
Runtime Efficiency	Fast, optimized for parallel execution and cloud scalability	Efficient, but less scalable than Nextflow	Moderate, requires Cromwell/MiniWDL overhead	Moderate, strict execution rules can introduce overhead	Varies, depends on server load and tool implementations
Community Support	Strong, nf-core community, industry adoption (AWS, Google Cloud)	Active, strong in academia	Moderate, growing WDL community, strong Broad Institute backing	Strong, backed by an open-source standards community	Very Strong, Galaxy Project, global user base, extensive training

Workflow Language Citations



Empowering bioinformatics communities with Nextflow and nf-core bioRxiv 2024.05.10.592912 doi: https://doi.org/10.1101/2024.05.10.592912

WES in the ecosystem of GA4GH APIs



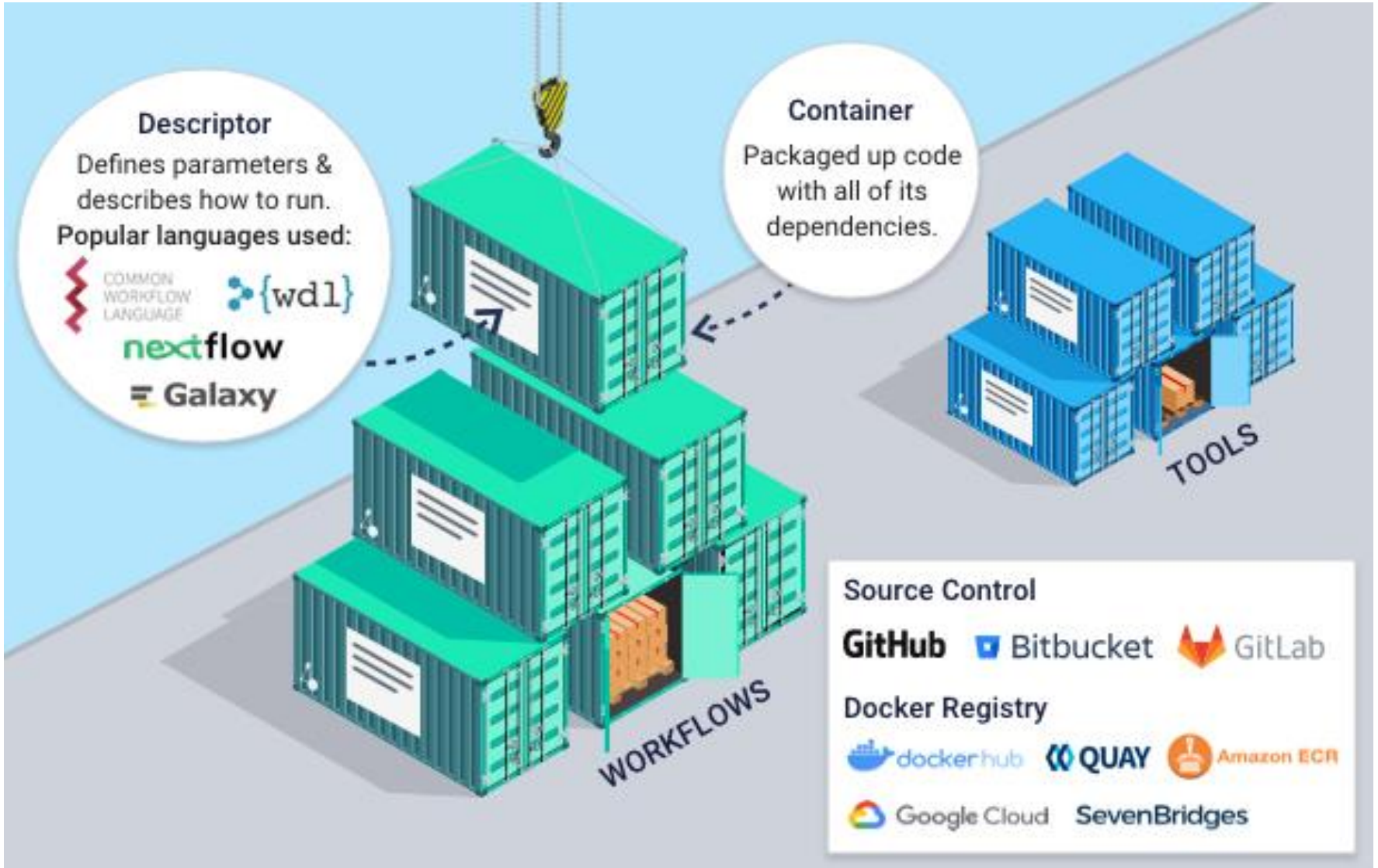
Workflow Executor Services (WES)

Feature	Seqera aka. Nextflow Tower	Snakemake Executor	Cromwell	Terra	Galaxy Platform
Works With	Nextflow	Snakemake	WDL	WDL, CWL	Galaxy workflows
Cloud Support	AWS, GCP, Azure	Manual configuration for cloud execution	AWS, GCP	Google Cloud	Galaxy.eu, Galaxy.org, and private instances
Scheduler Integration	SLURM, LSF, SGE, Kubernetes, AWS Batch	SLURM, LSF, SGE, Kubernetes	SLURM, LSF, AWS Batch, Google Cloud Life Sciences	Google Cloud-native execution	Managed internally
Ease of Use	Intuitive UI, enterprise support	CLI-based, requires manual setup	CLI and API-driven, requires configuration	User-friendly web interface	Extremely user-friendly (drag-and-drop UI)
Best for	Enterprise & large-scale cloud workflows	Researchers familiar with Python	WDL-based pipelines needing scalability	Cloud-based execution for non-experts	Accessible bioinformatics analysis without coding
Runtime Efficiency	High, optimized for batch processing & cloud	Moderate, efficient on HPC but manual cloud setup	Moderate, overhead from Cromwell	Varies, dependent on Google Cloud infrastructure	Varies, depends on Galaxy server load
Community Support	Strong, industry and academic adoption	Active, large user base in bioinformatics	Moderate, supported by Broad Institute	Moderate, growing user base due to cloud adoption	Very strong, Galaxy Project, global collaboration

Workflow Containers and Virtualization

Feature	Docker	Singularity	Kubernetes	Conda
Type	Containerization	Containerization	Orchestration	Environment Management
Use Case in Genomics	Standardized, portable execution of tools	HPC-friendly containerization	Manages/scales workflows	Manages dependencies without full containerization
Interoperability	Cloud, local, hybrid	Best for HPC, adaptable to cloud/local	Ideal for cloud/hybrid	Local/HPC-friendly but not as portable
Container Runtime	Requires Docker daemon (root access)	Runs unprivileged (better for HPC)	Supports Docker and Singularity	No containers, uses Conda virtual environments
HPC Compatibility	Limited due to root requirements	Designed for HPC (rootless execution)	Requires Kubernetes support	Works well on HPC but lacks full isolation
Ease of Use in Genomics Pipelines	Well-integrated with Nextflow, CWL, WDL	Preferred for HPC-based Nextflow, CWL	Used for scaling workflows	Preferred in Snakemake for dependency resolution
Dependency Handling	Uses container images (DockerHub, private registries)	Uses container images (Singularity Hub, DockerHub, custom registries)	Manages containers at scale but not dependencies directly	Resolves package dependencies within a Conda environment
Scalability	Works locally and in cloud; limited multi-node support	Optimized for HPC clusters	Highly scalable (auto-scaling, multi-node execution)	Not designed for large-scale workflows

Tool Registry Service (TRS)



Container & Tool Registry Overview

- **BioContainers** A community-driven registry providing bioinformatics tools as Docker and Singularity containers. Integrates with Conda.
- **Dockstore** A platform for sharing containerized bioinformatics workflows, supporting WDL, CWL, and Nextflow. Integrates with Terra, Galaxy, and other cloud platforms.
- **Bioconda** A repository of Conda packages specifically for bioinformatics, often used in Snakemake workflows.
- **Galaxy ToolShed** A repository of bioinformatics tools for the Galaxy platform, often linking to Docker images or Conda dependencies.
- **Quay.io** A general-purpose container registry (owned by Red Hat) where bioinformatics projects store Docker/Singularity images.
- **DockerHub** The largest general-purpose container registry for storing and distributing Docker images, widely used in bioinformatics workflows with Nextflow, CWL, and Snakemake.

Container & Tool Registry Service (TRS)

Feature	BioContainers	Dockstore	Bioconda	Galaxy ToolShed	DockerHub	Quay.io
Type	Container Registry	Workflow & Container Registry	Package Repository	Bioinformatics Tool Repository	General Container Registry	Container Registry
Primary Use Case	Provides pre-built bioinformatics containers	Hosts and shares workflows and containerized tools	Manages bioinformatics dependencies via Conda	Hosts tools and dependencies for Galaxy workflows	Stores and distributes general-purpose and bioinformatics container images	Stores and distributes container images
Supported Formats	Docker, Singularity	Docker, Singularity, WDL, CWL, Nextflow	Conda Packages	Galaxy Tools (XML-based), Conda	Docker	Docker, OCI Containers
Interoperability	Integrates with Conda, DockerHub, Singularity	Works with cloud platforms (Terra, Galaxy, AWS, etc.)	Primarily used in Snakemake and Conda environments	Integrates into the Galaxy bioinformatics platform	Supports Docker-based workflows, integrates with Kubernetes and HPC via Singularity	Supports public/private container hosting
Dependency Handling	Pre-packaged bioinformatics containers	Links workflows to containerized tools	Provides Conda-based dependency resolution	Manages tool dependencies via Conda/Docker	Requires manually built container images with dependencies	Stores general and bioinformatics containers
Workflow Language Support	Compatible with Nextflow, CWL, WDL	Designed for WDL, CWL, Nextflow	Works well with Snakemake	Native to Galaxy, can link to Conda/Docker	Works with Nextflow, CWL, WDL, Snakemake	Compatible with Nextflow, CWL, WDL
Scalability	Limited to available container images	Highly scalable, supports cloud and hybrid execution	Scalable for local/HPC package management	Scales with Galaxy instances	Scales globally via Kubernetes, HPC (via Singularity)	Scales with Kubernetes/OpenShift

Getting Started:

https://it.wustl.edu/services/cloud-computing/

A Workflow Example:

[TODO: Homework Link]