Experience Migrating a Pipeline for the C-MĀIKI gateway from Tapis v2 to Tapis v3

Yick Ching Wong
wongy@hawaii.edu
University of Hawai'i at Mānoa
Honolulu, Hawaii, USA

Sean B. Cleveland
University of Hawai'i -System
Honolulu, Hawaii, USA
seanbc@hawaii.edu

Gwen A. Jacobs
University of Hawa'i - System
Honolulu, HI, USA
gwenjh@hawaii.edu

ABSTRACT

The C-MĀIKI gateway is a science gateway that leverages a computational workload management API called Tapis to support modern, interoperable, and scalable microbiome data analysis. This project is focused on migrating an existing C-MĀIKI gateway pipeline from Tapis v2 to Tapis v3 so that it can take advantage of the new robust Tapis v3 features and stay modern. This requires three major steps: 1) Containerization of each existing microbiome workflow. 2) Create a new app definition for each of the workflows. 3) Enabling the ability to submit jobs to a SLURM scheduler inside of a singularity container to support the Nextflow workflow manager. This work presents the experience and challenges in upgrading the pipeline.

CCS CONCEPTS

• **Information systems → Computing platforms**.

KEYWORDS

Tapis Framework, C-MĀIKI, Application migration

ACM Reference Format:
Yick Ching Wong, Sean B. Cleveland, and Gwen A. Jacobs. 2022. Experience Migrating a Pipeline for the C-MĀIKI gateway from Tapis v2 to Tapis v3. In *Practice and Experience in Advanced Research Computing (PEARC '22), July 10–14, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3491418.3535139

1 INTRODUCTION

The C-MĀIKI science gateway allows researchers to run microbial workflows on a computer cluster with just a click of a button. This is possible because of the Tapis framework developed at the Texas Advanced Computing Center (TACC). Currently, the C-MĀIKI gateway uses the v2 version of Tapis, which has been refactored into a new v3 version that is more robust and has added capabilities such as support for containerized apps, a new Streaming Data API, and multi-site security kernel. This project aims to keep the C-MĀIKI gateway up-to-date and modern by migrating the gateway from the pre-existing Tapis v2 framework to the new Tapis v3 framework starting with one of the pipelines applications as a pilot.

PEARC '22, July 10–14, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9161-0/22/07…$15.00
https://doi.org/10.1145/3491418.3535139

2 BACKGROUND

2.1 Tapis

The Tapis Framework is an open-source, NSF-funded Application Program Interface (API) platform for distributed computation [5]. It provides production-grade capabilities to enable researchers to:

(1) Securely execute workflows that span geographically distributed providers
(2) Store and retrieve streaming/sensor data for real-time and batch job processing with support for temporal and spatial indexes and queries
(3) Leverage containerized codes to enable portability, and reduce the overall time-to-solution by utilizing data locality and other "smart scheduling" techniques
(4) Improve repeatability and reproducibility of computations with history and provenance tracking built into the API
(5) Manage access to data and results through a fine-grained permissions model, to enable secure sharing of digital assets with colleagues or the community at large

Researchers and applications are able to interact with Tapis by making authenticated HTTP requests to Tapis's public endpoints. In response to requests, Tapis's network of microservices interacts with a vast array of physical resources on behalf of users including high performance and high throughput computing clusters file servers and other storage systems, databases, bare metal, and virtual servers. Tapis aims to be the underlying cyberinfrastructure for a diverse set of research projects: from large-scale science gateways built to serve entire communities, to smaller projects and individual labs wanting to automate one or more components of their process.

Tapis can be accessed as either a centrally hosted solution, self-hosted or hybrid multi-site hosted. Tapis is multi-tenant, meaning that there can be a number of organizations (i.e. a grouping of users, such as an institution, lab, or project) using the same set of Tapis API services but persisting data in a logically separate, secure, namespace. The centrally hosted instance is currently hosted by the Texas Advanced Computing Center (TACC) at the University of Texas at Austin. Other institutions, such as the University of Hawaii (UH), have elected on a hybrid deployment. This allows the selection of subsets of Tapis API services deployed locally while leveraging others hosted at TACC.

2.2 C-MĀIKI gateway

The Center for Microbiome Analysis through Island Knowledge and Investigations (C-MĀIKI), the Hawaii EPSCoR Ike Wai project, and the Hawaii Data Science Institute developed a science gateway to automate and run MetaFlow|mics[1], a collection of three pipelines for processing microbiome samples. The C-MĀIKI gateway provides

a web-based interface for accessing advanced high-performance computing resources and storage to support and accelerate microbiome research. By leveraging the Tapis framework, this gateway has made microbial sequence analysis workflows and data easier to access, launch, track, manage and reproduce through a user-friendly web-accessible interface. To date the C-MĀIKI gateway has enabled 40+ researchers, graduates, and undergraduates to run over 1,200 pipeline jobs which break out into more than 750,000 parallel sub-jobs. This has enabled users to access more than 150,000 CPU hours and process more than 6 TB of data which would have taken close to 16 years sequentially to process.

3 MIGRATION

3.1 Compute Systems

For the initial setup of an environment for development and testing a Jetstream[4] development, VM was employed with Singularity 3.8 installed. This VM was then registered as an execution system with Tapis v3 using the tapipy Python library, allowing it to be used as a computational resource for job submissions. An example of a system definition used to set up the VM on Tapis v3 is shown in Listing 1. One goal of the project is to have the C-MĀIKI applications run on distributed systems, therefore the UH Manoa HPC system, MANA, was also registered as an execution system. The system definition for an HPC system is very similar to that seen on Listing 1, with the only difference being the host IP and the default authentication method (defaultAuthnMethod) which have been set to use public key authentication (PKI) keys for security.

3.2 Applications

One major difference between Tapis v2 and v3 is how applications are handled and registered. Specifically, applications in Tapis v3 must be containerized. The C-MĀIKI gateway currently supports 3 application pipelines created by the MetaFlow|mic project [1]. The 3 pipelines utilize Nextflow[2] to manage some of the parallelizations of steps in the pipelines, where the computation for each step is run inside of a Singularity[3] container. This leads to the first interesting challenge of the project which is that we must containerize an application that in itself is also containerized. To enable this "container within a container" the installation of SingularityCE 3.9.0 into our parent container definition was required and was implemented using a Dockerfile. This project leverage the ability of Singularity to convert a docker image to a singularity image for generating the Singularity containers used to run the pipelines as the UH Mana HPC system uses Singularity instead of Docker to mitigate permission escalations, as do many shared computational clusters.

These applications can be run either on a single machine or be run on an HPC system with a SLURM[6] scheduler. This requires an understanding of the location of where the SLURM files are installed on the compute resource and binding those files inside of the container in order to enable Nextflow to submit a batch script to run each step of the pipeline. A listing of the SLURM paths that need to be bound inside the container is shown on Listing 2. This functionality is still currently under development, due to bind mount challenges that arise from how Tapis stages the files for a job on the execution system.

```
1  {
2    "id": system_id_vm,
3    "description": "System for testing jobs on a VM",
4    "systemType": "LINUX",
5    "host": host,
6    "defaultAuthnMethod": "PASSWORD",
7    "port": 22,
8    "rootDir": "/home/"+user_id,
9    "canExec": True,
10   "jobRuntimes": [ { "runtimeType": "SINGULARITY" } ],
11   "jobWorkingDir": "workdir",
12   "canRunBatch": False
13 }
```

Listing 1: Example of a Tapis v3 system definition to set up a VM for development. Line 2 define the id of the system we are registering. Line 3 define a short description of the system. Line 4 represents what type of system it is, currently it supports Linux, Amazon Simple Cloud Storage/S3, Globus, and irods. Line 5 define the hostname/ip of the system. Line 6 define the default Authentication Method for the system, it currently supports Authentication methods such as passwords, public key infrastructure, and an access key. Line 7 define the port on the system to connect to. Line 8 define the root directory that Tapis will access. Line 9 define if the system can be used to execute jobs, if set to false, then it is just a system for storing and retrieving files. Line 10 define the types of runtime environment available on the system, the available inputs are Docker and Singularity. Line 11 define the name of the working directory where Tapis will work in, it will be created inside of the root directory. Line 12 define if the system is able to run batch jobs.

```
1  slurm_cmds="/bin/srun,/bin/sinfo,/bin/squeue,/bin/sbatch"
2  slurm_conf="/etc/slurm,/etc/slurm/plugins,/mnt/config/etc/sl
   ↪ urm/plugins/"
3  slurm_libs="/usr/lib64/slurm/"
4  munge_libs="/usr/lib64/libmunge.so.2"
5  munge_sock="/var/run/munge"
```

Listing 2: Example of paths that must be bound into the container so that it is able to submit SLURM batch script. It should be noted that these paths could be different between systems. Line 1 define a list of absolute paths to where the SLURM executables are store, in this example it includes srun, sinfo, squeue and sbatch. Line 2 define a list of absolute paths to where the SLURM configuration files are. Line 3 define a list of absolute paths that define where the slurm library files are. Line 4 defines the path where the munge library is, since munge is a needed authentication component for SLURM to work.Lastly Line 5 define the path to the library for a munge socket to run.

In order to register an application into Tapis, an app definition in the form of a JSON document must be created. This leads to another major difference between v2 and v3 which is how the app definition is structured. Input arguments were originally exported

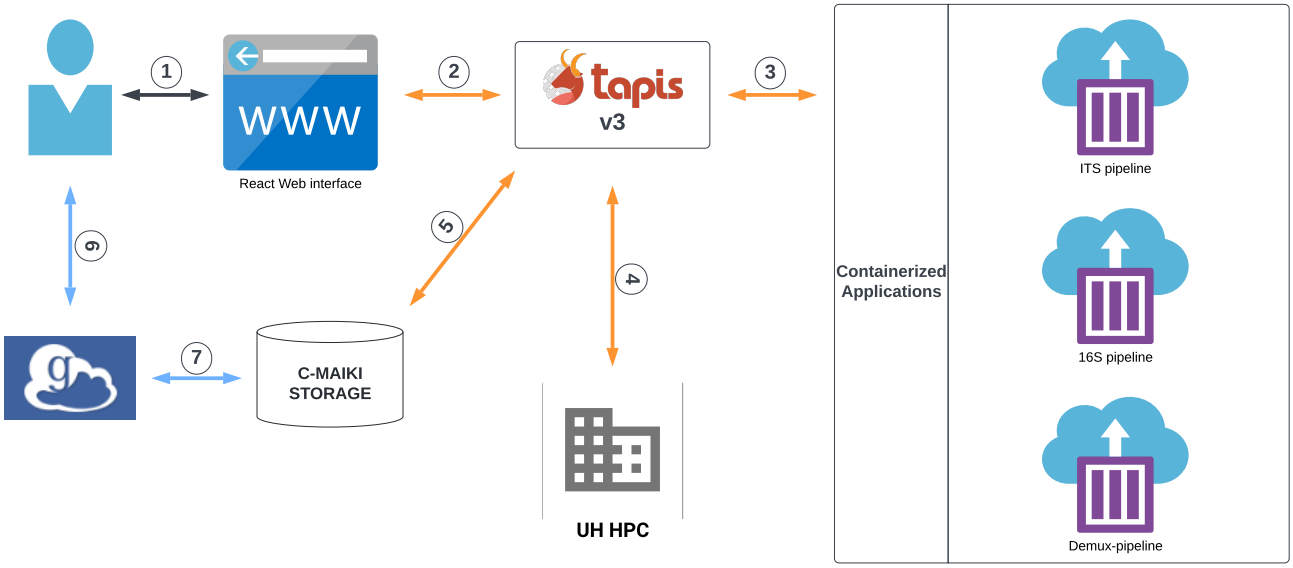

Figure 1: Diagram of how the C-MĀIKI gateway interacts with the Tapis framework (Orange arrows) to provide easy workflow execution on computing resources. Globus (blue arrows) also provide a second interface to access the storage system. Arrow 1 shows the initial connection between the researcher/user with the react web interface that will allow for easy job execution. Arrow 2 shows the second step where the web interface will then make calls to the Tapis API. Arrow 3 then shows the Tapis API gathering information on what type of containerized application is available, in this case, there is 3. Arrow 4 then shows the Tapis API gathering information on the type of systems available. All the information on the available system(s) and containerized application(s) will then be reported back to the web interface for the user to decide which job to execute and the system to execute the job on. Finally, Arrow 5 shows the last step where the output data will be stored on some type of storage system that the user can interact with via Globus shown in Arrow 6 and 7.

into the execution system as environment variable which is then read by a wrapper script that starts the entire pipeline. In v3, those input arguments are passed into the container itself as environment variables. The wrapper script is run inside the container, therefore adding a level of abstraction between the execution system and the application. Other application definition changes include renaming some of the keys in the JSON documents.

3.3 Job/workflow submission

In terms of job/workflow submission, the only major difference that had to be changed was how the user's arguments are passed to the app. The arguments are now environment variables inside the container instead of the execution system. Currently, an application is only ran using the test data that exists inside of the container, the ability for a user to provide a path to the data to be analyzed is still currently under development.

4 FUTURE WORK

The future work for this effort is to continue migrating the remaining pipelines to Tapis v3 by containerizing them and migrating their application definitions as has been done with the ITS pipeline. There is also a plan for a new user interface that to leverage the tapis-ui developed by the tapis-project group as a template for the new C-MĀIKI gateway for a very modern React-based front-end application that can be hosted on GitHub pages and make it easier for other groups to leverage some of the C-MĀIKI gateway capabilities in other Tapis tenants. A diagram showcasing how the C-ĀIKI gateway should interact with Tapis and the user interface is shown in Figure 1.

5 CONCLUSION

While there is still development to be done to migrate the entire C-MĀIKI gateway the initial pilot has shown that it is possible to run these Nextflow pipelines as a "container within a container" for parallelization. And though the migration from Tapis v2 to v3 for the C-MAIKI pipelines is challenging the ability to leverage a single application definition in Tapis and easily point the same app at a different execution system makes it worthwhile for the portability gained. This work will be leveraged to inform future work on migrating this and other Tapis-based science gateways.

6 SOFTWARE AVAILABILITY

The source code for the C-MAIKI gateway is available on GitHub at https://github.com/UH-CI/cmaiki-gateway and code for the MetaFlow|mics pipeline is available on GitHub at https://github.com/hawaiidatascience/metaflowmics with the documentation compiled and deposited on ReadTheDocs at https://metagenomics-pipelines.readthedocs.io/en/latest/. The

containerization of the pipeline is available on GitHub at https://github.com/UH-CI/c-maiki-tapis-ui

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation Office of Advanced CyberInfrastructure - Tapis Framework #1931439 and #1931575, RII Track 1: ʻIke Wai Securing Hawaiʻi's Water Future NSF OIA #1557349 and Division of Ocean Sciences #1636402.

REFERENCES

[1] Cedric Arisdakessian, Sean B Cleveland, and Mahdi Belcaid. 2020. MetaFlow|mics: Scalable and Reproducible Nextflow Pipelines for the Analysis of Microbiome Marker Data. In *PEARC20: Proceedings of the Practice and Experience of Advanced Research Computing*. PEARC.

[2] Paolo Di Tommaso, Maria Chatzou, Evan W Floden, Pablo Prieto Barja, Emilio Palumbo, and Cedric Notredame. 2017. Nextflow enables reproducible computational workflows. *Nature Biotechnology* 35, 4 (April 2017), 316–319. https://doi.org/10.1038/nbt.3820

[3] Gregory M Kurtzer, Vanessa Sochat, and Michael W Bauer. 2017. Singularity: Scientific containers for mobility of compute. *PloS one* 12, 5 (2017), e0177459.

[4] Craig A. Stewart, George Turner, Matthew Vaughn, Niall I. Gaffney, Timothy M. Cockerill, Ian Foster, David Hancock, Nirav Merchant, Edwin Skidmore, Daniel Stanzione, James Taylor, and Steven Tuecke. 2015. Jetstream. In *Proceedings of the 2015 XSEDE Conference on Scientific Advancements Enabled by Enhanced Cyberinfrastructure - XSEDE '15*. ACM Press. https://doi.org/10.1145/2792745.2792774

[5] Joe Stubbs, Richard Cardone, Mike Packard, Anagha Jamthe, Smruti Padhy, Steve Terry, Julia Looney, Joseph Meiring, Steve Black, Maytal Dahan, Sean Cleveland, and Gwen Jacobs. 2020. Tapis: An API Platform for Reproducible, Distributed Computational Research. (2020). submitted.

[6] Andy B Yoo, Morris A Jette, and Mark Grondona. 2003. Slurm: Simple linux utility for resource management. In *Workshop on job scheduling strategies for parallel processing*. Springer, 44–60.