Enhancing The Tapis Streams API - Keeping it Modern, Secure and Accessible

1st Sean Cleveland
University of Hawaii - System, Honolulu, HI, USA
seanbc@hawaii.edu

2nd Anagha Jamthe
Texas Advanced Computing Center, Austin, TX, USA
ajamthe@tacc.utexas.edu

3rd Jared McLean
University of Hawaii at Manoa, HI, USA
mcleanj@hawaii.edu

4th Smruti Padhy
Texas Advanced Computing Center, Austin, TX, USA
spadhy@tacc.utexas.edu

5th Maytal Dahan
Texas Advanced Computing Center, Austin, TX, USA
maytal@tacc.utexas.edu

6th Joe Stubbs
Texas Advanced Computing Center, Austin, TX, USA
jstubbs@tacc.utexas.edu

7th Gwen A. Jacobs
University of Hawaii - System, Honolulu, HI, USA
gwenh@hawaii.edu

***Abstract*—The Tapis Streams API is a production grade quality service that provides REST APIs for storing, processing and analyzing real-time streaming data. This paper focuses on improvements made to Tapis 1.0 Streams API for making it up-to-date and easily accessible. The newer version, Tapis 1.2 Streams API adopts the latest version of InfluxDB, InfluxDB 2.X, which has built-in security features and supports next generation data analytics and processing with a data processing language Flux. This paper also discusses the measures implemented in the Tapis 1.2 Streams API to mitigate potential security risks involved in unauthorized data stream access by users who do not own it. Additionally, new data Channel Actions supporting 3rd Party notification and web-hooks has been released. Lastly a tool, Tapis UI, which is a self contained server less application to access Tapis Services via rest calls is discussed in the paper. Tapis UI is a lightweight browser only client application which allows interactive access to Streams resources and real-time streaming data.**

***Index Terms*—CHORDS, Tapis, InfluxDB, Flux, UI**

I. Introduction

The recent rise of inexpensive devices in the Internet of Things (IoT) as well as the number of instruments and sensors to observe and measure everything has led to a deluge of data and demand for support related to storing, processing, and analyzing time-series data. Many science use cases based on monitoring require ongoing data processing or special processing/modeling and notification of anomalous or special events that can be identified by processing and computing the data as it arrives. Systems that have been designed for sensors are often aimed at industry, and either are complex to deploy and maintain or, if hosted, are expensive. This leaves a gap for hosted academic streaming data solutions capable of supporting data event-driven computational workflows. To help address this cyberinfrastructure need to support streaming data for science, the Texas Advanced Computing Center (TACC) and the University of Hawaii (UH) have developed an open-source unified middleware API infrastructure platform, Tapis, with collaborative features to fill gaps in the existing streaming time-series data landscape.

A sustainable cyberinfrastructure that supports real-time streaming data is essential for the success of data-driven scientific use-cases and workflows, which demand continuous streaming-data processing. Very few systems designed today support academic uses-cases and provide an end-to-end solution for integrating streaming-data workflows, storage, analysis, and retrieval of the time and location-sensitive data. Streams API was developed as a part of the NSF-funded Tapis Project [1], a collaborative grant between the Texas Advanced Computing Center and the University of Hawaii, to support real-time sensor data collection and analysis. The goals of the Streams API [2] have been focused on creating an interface that eliminates the complexities of gathering, storing, and processing sensor data, enabling researchers and domain scientists to create scientific gateways and workflows using simple HTTP requests. Early adopters such as climatologists working on the precipitation mapping and real-time water quality monitoring provided requirements for developing Streams API.

In this paper, we present the updates to the architecture of the Tapis Streams service along with influxDB migration implementation from 1.x to 2.x. We also provide design

Presented at Gateways 2022, San Diego, USA, October 18–20, 2022. https://zenodo.org/communities/gateways2022/

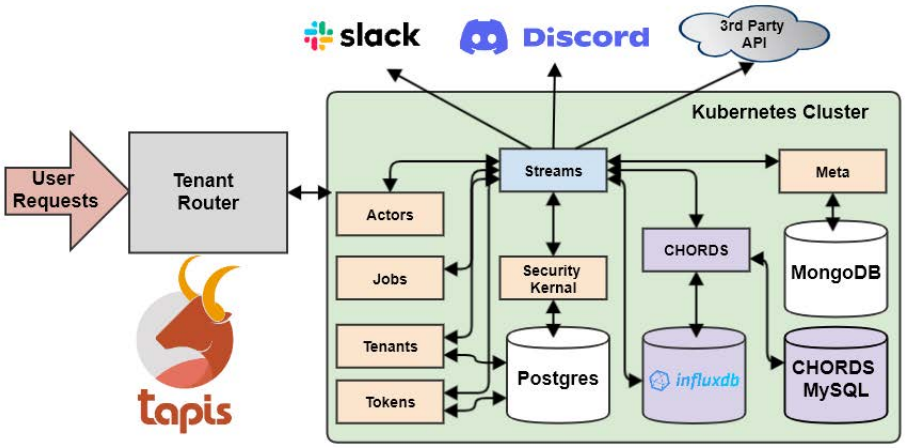

Fig. 1. Tapis Streams API Architecture and New Channels Action Integrations (Slack, Discord and 3rd Party Web-hooks)

and implementation experiences such as maintaining security, and updating to influxDB 2.x. We also present new Channel action methods and the Tapis UI, a self-contained, serverless application to access Tapis Services via rest calls. Tapis UI is a lightweight browser-only client with built-in typescript and react that enables accessing Streams resources and visualizing the time-series data.

Further, we will discuss the background in Section II, implementation in Section III, Tapis UI in Section IV, future work in Section V, and opportunities related to the Tapis Streams API in Section VI.

II. Background

This section describes the key concepts involved in the implementation of Tapis Streams API and the API itself.

A. *InfluxDB*

InfluxDB is a time series database designed to handle high write and query loads. InfluxDB is used in many applications involving large amounts of timestamped data, including DevOps monitoring, application metrics, IoT sensor data, and real-time analytics. The Tapis Streams project has leveraged the InfluxDB and it's ecosystem of services, like Kapaictor a real-time streams processing engine, as the time series infrastructure for processing and storing time-series data. The InfluxDB 1.X version is what was leveraged to implement the original Tapis 1.0 Streams APIs. The latest release of the Tapis Streams API has migrated to InfluxDB 2.0 this has resulted in some significant implementation changes for the better. Three major changes include moving the Kapacitor processing engine into InfluxDB, adopting Flux as the supported scripting language for queries and task and making the Influx database into a bucket. Addressing these changes is discussed in more detail in the Implementation Section.

B. *Tapis v3 Streams*

Tapis provides a enriched set of open source, hosted Application Program Interface (API) platform for distributed computation that enables researchers to manage data and execute codes on a wide range of remote systems, from high-speed storage and high-performance computing systems to commodity servers.

Tapis V3 Streams API has been developed to support real-time streaming data workflows with storage, retrieval and analysis of the temporal sensor data. The API is built using Python Flask web framework and interacts with other Tapis Services such as Actors, Jobs, Security Kernel, Meta, Tenants, and Tokens services as shown in 1. Streams resources are hierarchical and based on the Cloud HOsted Real-time Data Services(CHORDS) [3] resource model, for example, Project is at the top level in the hierarchy which contains important information such as project description, principal investigator, owner, metadata about the project. Next in the hierarchy is a Site. A site is a geographical location with spatial co-ordinates such as latitude, longitude and elevation, where the physical hardware for remote sensing is located. A project can have multiple sites and the geo-spatial coordinates associated with the sites can be used to search data related to that site. The physical hardware where multiple sensing devices are embedded is next in the hierarchy and is known as an instrument. Each site can host multiple instruments and they can be identified with their unique ids. Individual sensors are referred to as variables, which sense physical parameters such as temperature, humidity, rainfall, etc. and these measurements are stored in the InfluxDB time-series database. All the resources can be accessed only by authorized users and individual user roles.

Streams API is deployed on a on-premise Kubernetes cluster (Fig 1.) hosted at TACC along with other production grade Tapis services. The Streams API deployment consists of four primary components: the Python API, the CHORDS server, the and two databases: InfluxDB for time-series measurements and MySQL which CHORDS uses. CHORDS and InfluxDB services are also deployed as individual services in the same cluster. Kubernetes configmaps and secrets objects are used to

configure the deployments. A Tapis deployer tool, developed at TACC is used to automate the creation of configmaps and secrets, persistent volume claims and to start the entire Streams API stack. Every user request to access Streams resources first goes through the Tapis Security Kernel for authorization and authentication check to ensure that the user has the necessary role to perform requested action on the specified Streams resource. Tokens service provides a signed service JWT, which lets the Security Kernel and Metadata service to know that request is coming from an authentic source, i.e., Streams service. Metadata service provides a backend MongoDB for Streams API, which stores all the metadata associated with the Streams resources.

C. Tapis UI

To facilitate the usage of the Tapis APIs python and TypeScript packages were developed. These packages provide wrapper functions for submitting requests to Tapis. Additionally, an online portal for interacting with Tapis via a user interface, Tapis UI, was developed. This interface was created using React, an open-source JavaScript framework for creating web applications, and leverages the Tapis TypeScript library for dispatching user actions to the Tapis APIs. A component for interacting with the Tapis Streams API was included in this portal. More details on Tapis UI is discussed in the later sections.

III. Implementation

A. Updating to InfluxDB2

To keep the Tapis Streams API up to date migration from InfluxDB 1.X was required. The latest version of InfluxDB 2 was a full rewrite by the Influx developers to enhance performance and add additional features. This updated version of Influx streamlined some of the services, where with the 1.X version the Kapacitor service was an add on separate process and in the latest Influx it's features were incorporated directly into the main Influx service. This meant that the Tapis Streams API had to be refactored on the backend to implement templates and channels against a new set of Influx APIs. Further, the Python SDK for InfluxDB 2 is a different library with different methods for writing/reading data in addition to Tasks.

Updating the influxDB2 deployment for the Tapis Streams API required changing the InfluxDB container version and tag to influxdb:2.1.1-alpine. In development environments a docker-compose file is used for deploying all the local dependencies (chords application, mysql, influxdb etc.) so the container and tag were updated there. In production environments Tapis is deployed on a Kubernetes cluster so this required updating the config-map files. In addition to the container other configuration variables had to be added to support InfluxDB2. These included the default bucket name (this replaced the previous influx database name), InfluxDB admin username, password and token. With those

B. Maintaining Security

Due to the ability for users to submit Flux scripts as part of the Streams Channels there was the potential for a bad actor to generate a channel that could subscribe to a data stream they did not own. To mitigate this the latest Streams implementation takes advantage of having databases as buckets to segregates each projects data into it's own bucket and strips out references to hard-coded buckets in the Flux codes. The Tapis Streams service inserts its own project bucket reference at the time of Channel creation that corresponds to the project definition related to the specific instruments and variable parameters. This ensures new Channel definitions can only access the project bucket the user has authorized access defined.

C. Expanding Channel Actions

The first Streams API release included the ability for registering an instrument's variable value for evaluation during ingestion to trigger a Tapis Actor execution. With the migration to InfluxDB2 the lastest Steams API added some additional actor methods. In addition to "ACTOR" as a method there is now "DISCORD", "SLACK" and "WEBHOOK". These new methods support passing a message to Slack, Discord or a 3rd part API HTTP POST endpoint (Listing 1). The "SLACCK" and "DISCORD" actions were implemented to construct POST bodies appropriate to those respective APIs while the "WEBHOOK" action has additional fields to define the appropriate body for a generic HTTP POST API JSON body. These new actions allow for lightweight integration for notifications and integrations to other services.

```
client.streams.create_channels(channel_id="training
    .discord.demo.tapis.channel",
        channel_name='discord-demo.tapis.channel',
        template_id="default_threshold",
        triggers_with_actions=[
        {"inst_ids":[inst_id],
         "condition":{"key":inst_id+".rainfall",
         "operator":">",
         "val":150},
         "action":{
           "method":"DISCORD",
           "webhook_url":"https://discordapp.com/api
    /webhooks/XXXXXXXXX/XXXXX",
           "message":"My Instrument exceeded
    Rainfall threshold val ${ r.value}"
         }}])
```

Listing 1. Example Streams Channel definition with Discord notification action using the tapipy Tapis Python SDK

IV. Tapis UI for the Streams API

The Tapis UI Streams interface allows users to view data stored by the Streams API. Projects, Sites, and Instruments are listed in a hierarchical interface. Users are initially presented with a list of projects. Clicking on a project will list the sites associated with that project, and clicking on a site will list the instruments associated with that site. Once a site is selected the measurements for each variable tracked by that instrument are displayed.

The displayed measurement data is grouped by variable name. Each group provides a listing of the measurement values

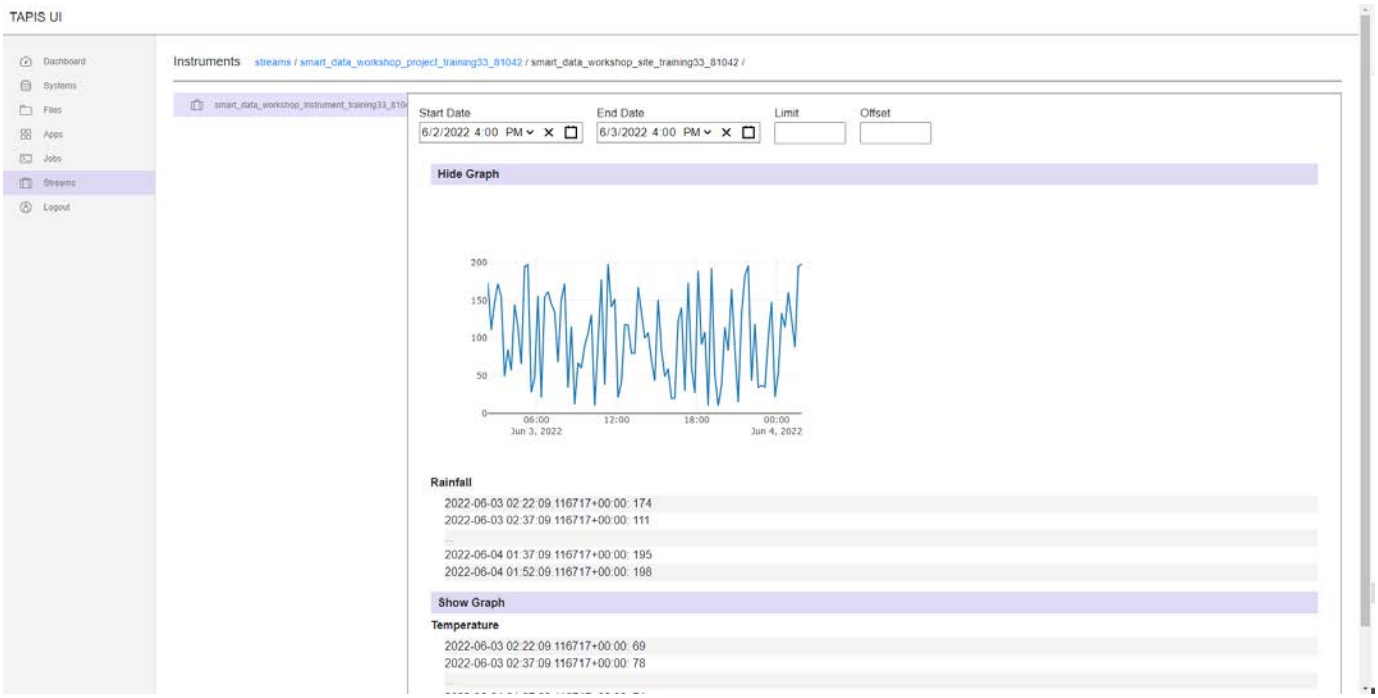

Fig. 2. Tapis UI for the Streams API. A set of measurements for rainfall and temperature over a 24 hour period are displayed.

and a timestamp for when the measurement was taken. Additionally, a graph representing a time series of the measured values is provided (Figure 2). Large numbers of values are collapsed by default, displaying only the first and last two measurements. This can be expanded by clicking on the block of values.

The set of returned measurements can also be limited using a set of filters displayed at the top of the measurements panel. The available parameters are start date, end date, limit, and offset. Setting a start or end date will limit the returned measurements to the specified range based on their timestamp. The limit field specifies a maximum number of values to be returned. The offset field specifies the first value to be returned. For example, an offset of 5 will skip the first four measurements and return values starting at the fifth measurement.

This portal provides a simple pre-developed dashboard for users to view data being pushed into the Streams API. This can limit the need for additional developer overhead for generating basic visuals or monitor and validate data streams.

V. Future Work

The Tapis Streams API will continue to evolve with additional actions to include Tapis jobs integrations, advanced search capabiites, share-able pre-authenticated data links and ontology support for rich metadata.

VI. Conclusion

In conclusion this paper has presented the updates to the Tapis Streams design and implementation that enhanced security, utility and access while maintaining the integrity of the specifications. These new enhancements keep the Tapis Streams API current while allowing it to better serve researchers through enhanced notifications, integrations and basic data access and visualization of Stream's data.

VII. Software Availability

The source code for the Tapis Streams API is available on GitHub at https://github.com/tapis-project/streams-api. Source code for the Tapis-UI with streams can be found https://github.com/tapis-project/tapis-ui

Acknowledgements

This work is supported by the National Science Foundation Office of Advanced CyberInfrastructure - Tapis Framework #1931439 and #1931575.

References

[1] J. Stubbs *et al.*, "Tapis: An api platform for reproducible, distributed computational research," *Future Generation Computer Systems*, 2021, accepted.

[2] S. Cleveland *et al.*, "Tapis-chords integration: Time-series data support in science gateway infrastructure." Proceedings of Science Gateways Conference 2019, 2019.

[3] B. Kerkez *et al.*, "Cloud hosted real-time data services for the geosciences (chords)." Geoscience Data Journal, 2016, pp. 2–4.