Tapis Pipelines Documentation

Joe Stubbs

Aug 26, 2021

INTRODUCTION:

CHAPTER

ONE

TAPIS PIPELINES

Welcome to the Tapis Pipelines documentation.

1.1 What are Tapis Pipelines?

Tapis Pipelines automate data analysis workflows on TACC Cloud and HPC resources in a secure, robust, and fault-tolerant way. The Tapis Pipelines software integrates directly with the Tapis v3 API Framework to allow users to define workflows that incorporate Tapis services such as the Tapis Metadata, Actors (functions-as-a-service), Systems, Apps, Jobs and other services.

Tapis Pipelines have been architected to fit within broader data analysis processing workflows where only a part of the computation actually runs at TACC. This work grew out of collaborations with NASA JPL and partners to support the processing of data (exposures) captured by the NEID telescope instrument at Kitt Peak National Observatory. Partner institutions include Caltech, IPAC (Infrared Processing & Analysis Center), JPL, Kitt Peak, Penn State, and the University of Arizona.

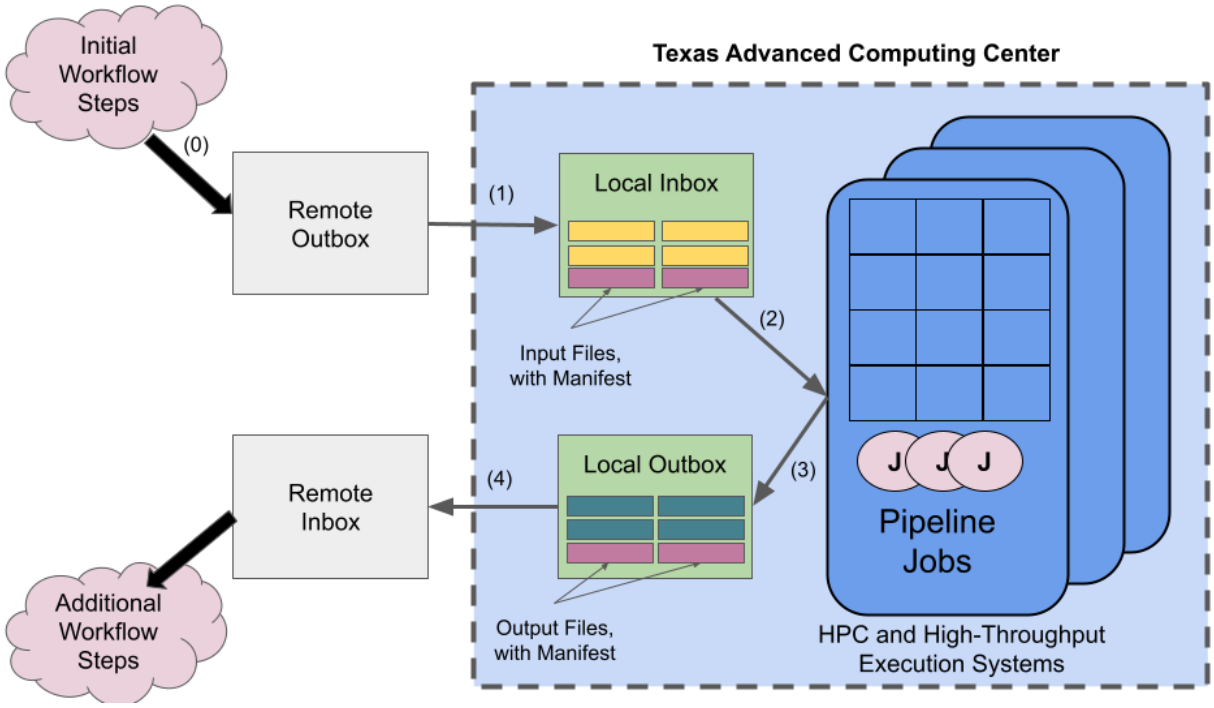
There are two ways to build Tapis pipelines. First, the Tapis Pipelines software is available as a Python package that can be installed and configured directly on a Linux-compliant machine. This method is available today and provides the ability to customize the Pipeline software itself, offering the most flexibility. The second method, currently in development, is to utilize the Tapis Pipelines Service, a hosted version of the software developed and maintained by TACC staff.

1.1.1 10,000 Foot View

At a high level, each Tapis Pipeline consists of the following steps:

0. Files to be processed are copied to the pipeline's configured *Remote Outbox*. (This step is not performed by the Tapis Pipelines software).
1. As new files arrive in the Remote Outbox, they are staged to the pipeline's configured *Local Inbox* at TACC where basic validation and pre-processing can occur.
2. The pipeline's configured *entrypoint* is launched to "process" files in the Local Inbox. Each execution of the entrypoint is referred to as a *pipeline job*.
3. As the pipeline job runs, output files are written to the pipeline's *Local Outbox* at TACC.
4. On successful completion of a pipeline job, files in the Local Outbox are transferred to the pipeline's configured *Remote Inbox*. From there, additional workflow steps can occur outside of TACC.

For each step above, there are several configurable aspects, and the software includes mechanisms for ensuring robustness and integrity of the pipeline processing. Some of the most salient aspects include:



1. The Remote Inbox can be configured to be a Globus endpoint or a path on a Tapis system. Tapis supports defining systems via a number of protocols and transfer methods, including SSH (e.g., an SFTP server) as well cloud object store (e.g., an AWS S3 bucket) among others.
2. Files staged to the Remote Inbox include *manifest files* which describe a set of files that should be processed together as a single job. Manifest files must conform to a schema specific to Tapis Pipelines software.
3. The Local Inbox should be chosen in conjunction with the execution system(s) where pipeline jobs will run. For example, the Local Inbox could be a directory on TACC's global file system, Stockyard, a directory on a scratch file system on one of the HPC systems, or a directory on TACC's Corral storage resource, mounted on a number of cloud and high-throughput execution systems.
4. Pipeline jobs are configured with one or more execution targets, a *primary* and optionally, one or more *secondary* systems at TACC where pipeline jobs will actually run. These can be traditional HPC supercomputers or cloud/high-throughput systems. When a primary system is offline – for example, due to a planned maintenance – jobs will be automatically routed to a secondary system. (Note: the Local Inbox must be available on each execution target).
5. Each pipeline configuration includes a *run schedule* which dictates how new pipeline jobs are scheduled. The options include a fixed schedule, similar to cron (e.g., run every day at 2 AM), or *streaming*, which runs jobs as soon as new manifest files arrive in the Remote Outbox.

These topics are covered in more detail in the *User Guide*. Additionally, The *Developer Guide* includes design and implementation details about the Tapis Pipelines software itself, useful for customizing and extending the software with new features.

CHAPTER

TWO

USER GUIDE

This User guide provides a complete reference to using the Tapis Pipeline software.

2.1 Overview and Prerequisites

The Tapis Pipeline software is designed to assist you in running recurring computational jobs against multiple data sets. The design is based on the idea of ETL. It has been designed with TACC resources in mind, but should be broadly applicable to other resources as well. The general idea is:

- Download input data from remote source (the Remote Outbox) to TACC (the Local Inbox).
- Compute data products from the input using whatever software you like (the pipeline job software).
- Send resulting output from TACC (the Local Outbox) back to a remote storage site (the Remote Inobx).

Before you begin building your first Tapis Pipeline, there are a few decisions to make and some prerequisites your project should meet. We collect these here. Think of it as checklist.

2.1.1 TACC Storage and Computing Allocation

Your project will need a storage allocation on some TACC storage resource (for example, Corral, Stockyard, or one of our Cloud-based storage resources) to serve as the project's Local Inbox and Outbox. The size of the required allocation greatly depends on the size of the files that will be processed in the pipeline.

Your project will also need one or more allocations on a computing system at TACC, such as Frontera, Stampede2, Lonestar5, or one of cloud computing systems. The allocation will be used to run pipeline jobs.

2.1.2 Packaging and Installation of Analysis (Pipeline Job) Software

The ultimate goal of a pipeline is to process data via one or more programs, defined by the project. This software, referred to as the *pipeline job* software, must be packaged and accessible to the Tapis Pipelines software.

There are a few packaging options available:

1. Create a container image with the job software. We recommend using Singularity containers for running jobs on HPC systems and Docker, or a container runtime that supports the k8s Container Runtime Interface (CRI), for running jobs on cloud resources, such as the Kubernetes Freetail system.
2. Package the software using conventional methods, such as RPMs, Python packages, Ruby gems, git repositories, etc.

These lead to the following installation options:

1. If the job software is packaged as a container as in option 1, the software can be registered as a Tapis App (Singularity or Docker) or Tapis Actor/function (Docker). This is the preferred approach, as it does not require maintaining a separate installation of the job software on each execution system to be used. It also simplifies permissions on the underlying files that must be maintained so that the Tapis Pipelines software can execute it.
2. Install the job software on the head node (login node) of a TACC execution system. Choose the system matching the allocation for the project. If there are multiple systems, the software must be installed on each one. This method is **not recommended**.

2.1.3 Remote Outbox and Inbox

Each pipeline must configure a Remote Outbox and a Remote Inbox where files requiring processing (respectively, output files resulting from processing) will be stored. Conceptually, the Remote Outbox and Inbox are storage resources independent of TACC, but they must provide programmatic access. Options for the Remote Outbox and Inbox include:

1. A path on a Tapis System, including POSIX (SSH/SFTP) and Object storage (S3-compatible).
2. A Globus endpoint.

With Option 1, the Tapis Pipelines software will be able to utilize Tapis transfers to move data to/from the Remote Outbox and Inbox to any TACC resource. This is the recommended option.

With Option 2, the Tapis Pipelines software utilizes Globus Personal Connect to move data to/from the Remote Outbox and Inbox to the Local Outbox and Inbox. From there, Tapis transfers will be utilized, as needed.

2.1.4 Manifest Files

A key to the Tapis Pipeline architecture is the *manifest file*.

Pipeline jobs process files that get transferred to the Remote Outbox. Each job will process any number of files, and the number of files processed by a single job is determined by the manifest file. The manifest file is a simple JSON file that describes one or more files to be processed by a job. It can include basic validation information (such as an MD5 checksum of the file), project generated identifies for the job, and some limited support for overriding the default job runner behavior (for example, to specify the job can run at a lower priority than other jobs).

Critically, **the presence of a manifest file in the Remote Outbox instructs the Tapis Pipelines software that the files referenced within it are ready to be processed as a job**. In particular, all data transfer to the Remote Outbox has completed. No files in the Remote Outbox will be processed until they are included in some manifest file.

The manifest file must adhere to a required format described by a JSON Schema. Invalid manifest files are never processed.

The following describes the format of the mainfest file.

http://github.com/tapis-project/tapis-pipelines/core/manifest_schema.json	
A schema describing a valid Tapis Pipelines manifest file.	
type	*object*
properties	
• **files**	*files_list*
• job_config	*job_config*

files_list

List of files to be processed by this job.	
type	*array*
items	*file*

file

A file to be processed as part of a job.		
type	*object*	
properties		
• **file_path**	Path to the file on the remote inbox.	
	type	*string*
• md5_checksum	The md5 checksum of the file. Used for validation purposes.	
	type	*string*

job_config

Special configuration overrides to apply to this specific job.		
type	*object*	
properties		
• priority	Specify a different priority for this job.	
	type	*string*

2.2 Installing Tapis Pipelines Software

The Tapis Pipelines software is available as a Python package. To install it, simply type:

```
$ pip install tapis-pipelines
```

Installing Tapis Pipelines in a virtualenv is recommended.

Alternatively, you can install Tapis Pipelines from source by checking out the repository from GitHub.

2.3 Configuration of Tapis Pipelines

An instance of the Tapis Pipelines software must be configured for a specific pipeline. The configuration is provided as a JSON file that conforms to the Tapis Pipeline config JSON Schema definition.

http://github.com/tapis-project/tapis-pipelines/core/configschema.json	
type	*object*
properties	
• **remote_inbox**	*remote_box_definition*
• **remote_outbox**	*remote_box_definition*
• **pipeline_job**	*pipeline_job_definition*
• tapis_config	*tapis_config_definition*

2.3.1 remote_box_definition

Configuration of remote inbox or outbox		
type	*object*	
properties		
• **kind**	The type of Remote Box being configured.	
	type	*string*
	enum	tapis, globus
• **box_definition**	*box_definition*	

2.3.2 box_definition

oneOf	*tapis_box_definition*
	globus_box_definition

2.3.3 tapis_box_definition

A pipeline box defined using a Tapis system and path.		
type	*object*	
properties		
• **system_id**	The id of the Tapis system to use for the box definition.	
	type	*string*
• **path**	Path on the Tapis system to use for the box definition.	
	type	*string*

2.3.4 globus_box_definition

A pipeline box defined using a Globus endpoint.		
type	*object*	
properties		
• **client_id**	The id of the Globus client to use when issuing transfers.	
	type	*string*
• **endpoint_name**	The name of the Globus endpoint.	
	type	*string*
• **directory**	The directory within the Globus endpoint to use for the box definition.	
	type	*string*

2.3.5 pipeline_job_definition

Description of the pipeline job to run on new input files.	
oneOf	*tapis_app_job*
	tapis_actor_job

2.3.6 tapis_app_job

A Pipeline job described using a Tapis app		
type	*object*	
properties		
• **app_id**	The app id to use when submitting the job,	
	type	*string*
• **manifest_input_name**	The name of the input on the Tapis app for the manifest file.	
	type	*string*
	default	manifest_file
• raw_files_input_name	The name of the input on the Tapis app to be used for sending the raw input files. If empty, no input name will be specified.	
	type	*string*
	default	

2.3.7 tapis_actor_job

A Pipeline job described using a Tapis actor		
type	*object*	
properties		
• **actor_id**	The id of the actor. The Tapis Pipelines software will send a JSON message to the actor with details about the job (see documentation).	
	type	*string*

2.3.8 local_script_job

A Pipeline job described using a local script	
type	*object*

2.3.9 tapis_config_definition

General configuration for Tapis usage.		
type	*object*	
properties		
• **base_url**	The base URL for the Tapis tenant to interact with.	
	type	*string*
• **username**	The Tapis username to use when accessing Tapis services.	
	type	*string*

2.4 Testing A Pipeline

A number of measures can be take to validate that a pipeline will run correctly before using real cycles.

2.4.1 Validate Configuration

The Tapis Pipelines software includes a config validator that can be run to ensure that all required configurations are present and valid. The validator does basic type checking of all fields. Run the config validator first before moving on to subsequent validation.

2.4.2 Package tests

The Tapis Pipelines software includes a package of tests that can be run once the software is configured. These tests exercise some of the primary functions of the software, such as interacting with the Tapis APIs using the configured authentication. If any of these functions fails, some installation or configuration step is likely missing or incorrect and the pipeline jobs are unlikely to run correctly.

2.4.3 Test Pipeline Runs

In some cases, it can be possible to issue end-to-end test runs of a pipeline using sample data.

To do, more on this coming soon…

2.5 Production Pipelines and Dashboard

The Pipelines software makes use of Tapis Metadata service to track the status of jobs as they progress. We include a simple dashboard for displaying the information. The dashboard code can be deployed relatively quickly to most modern web servers.

2.6 Troubleshooting and FAQ

Coming soon...

CHAPTER

THREE

DEVELOPER GUIDE

The Developer Guide provides details about the architecture and design of the Tapis Pipelines software. This information is intended for developers wanting to customizing and/or extend the software with new features, or who just want to understand how Tapis Pipelines work at a deeper level.

3.1 Steps in Pipeline

These steps are designed to be somewhat idempotent, so that each step may be run multiple times and it will not harm existing or previously finished jobs.

This helps to facilitate the automation of the jobs.

Each job is associated with an input data set and each is tracked separately using Tapis Metadata service.

3.1.1 1. Get Remote

This downloads input files from the remote source. It scans the remote source (Remote Outbox) for files which have not already been downloaded and initiates transfer.

Once transferred, it sets the metadata to transfer_to_local_done.

If required by the app, it also unpacks the data (e.g. if it is inside a .tar file.) It then sets the metadata to unpack_on_local_done.

The implementation of this step is captured in the 'get-remote-via-globus-personal.py' script.

3.1.2 2. Compute Job

This submits the job to the computing resource, determines that the compute job has finished, and optionally runs post-compute checks. These checks can determine if the job finished correctly or should be resubmitted.

In the case of submitting a Slurm job to a queue, it creates the sbatch file from a template. Since this template is application specific, it is provided by the project. A simple example of such a template is provided in the repository (sbatch_template.j2).

This step can optionally have several components or sub-steps, as required by the project. Complex decision trees are possible within this step.

Once processing is finished, metadata is set to processing_data_done.

The implementation of this step is captured in the 'compute-local-jobs-auto.py' script.

3.1.3 3. Send Results

This packs up the job output and sends it back to the Remote Inbox. It optionally packs the data into an archive file (e.g. tar).

Once transfer is finished, metadata is set to transfer_to_remote_done.

The implementation of this step is captured in the 'send-results-via-globus-personal.py' script.

3.1.4 4. Finished Pipeline

Once the output is successfully transferred to Remote storage, the pipeline for this job is considered complete and metadata is set to pipeline_done.

3.2 Automating the Steps

In a future release, these components may be launched via more advanced Tapis API processes (e.g. Actors, Jobs, etc.)

We include examples of scripts to run the various steps periodically from cron. Each runs separately so that they do not need to wait for each other to finish. They record the output of the scripts in dated files.

```
$ crontab -l
SHELL=/bin/bash
1 * * * * . /etc/profile; $HOME/pipeline/01runget
2 * * * * . /etc/profile; $HOME/pipeline/02runcompute
3 * * * * . /etc/profile; $HOME/pipeline/03runtarup
4 * * * * . /etc/profile; $HOME/pipeline/04runsend
```