PipelineJobs System Documentation

Release 0.2.0

TACC Open Source

Feb 04, 2019

Components

1 Pipelines Manager 1

2 PipelineJobs Manager 9

3 PipelineJobs Indexer 13

4 PipelineJobs Agave Proxy 17

5 HTTP Authentication 25

6 Send a Bearer Token 27

7 Use a Nonce 29

8 Authorization Tokens 31

9 Reactor+App with PipelineJobs Integration 33

10 Indices and tables 35

CHAPTER 1

Pipelines Manager

This Reactor enables creation and management of Data Catalog **Pipelines**.

1.1 Pipelines

Pipelines are defined using document written in the **Pipeline** JSON schema. Here is a brief, slightly silly example:

```
{
    "name": "Tacobot 9001",
    "description": "Creates even better breakfast tacos out of silicon, ozone, and␣
↪shredded GPUs",
    "components": [
        {
            "id": "e1LqaqMyqq4xB",
            "opts": {
                "cheese": true,
                "salsa": true,
                "eggs": false
            }
        },
        {
            "id": "bbq-brisket-0.5.0",
            "parameters": {
                "smoke": true,
                "quantity": 4
            }
        }
    ],
    "processing_levels": [
        "1"
    ],
    "accepts": [
        "CSV"
```

(continues on next page)

(continued from previous page)

```
    ],
    "produces": [
        "PNG"
    ]
}
```

A Pipeline is a human-readable name and description, a globally-unique string identifier, and a list of components that defines some number of Abaco Actors, Agave Apps, Deployed Containers, and Web Services. Additionally, one or more data "processing levels" are provided as well as the list of file types accepted and emitted. Of these fields, only `components` is used to create a Pipeline UUID that connects each Pipeline to its compute jobs.

To make this very tangible: In the example above, if *anything*n in `components` changes, the result will have to be a new UUID. If the name, file types, description, or description change, the UUID remains the same.

1.2 Messages

All Pipeline management actions are accomplished by sending JSON-formatted messages to the **Pipelines Manager** Reactor.

1.3 Create a New Pipeline

A new pipeline can be registered by sending its JSON document to the **Pipelines Manager** as a message like so:

```
$ abaco run -m "$(jq -c . my_pipeline.json)" G1p783PxpalBB

gOvQRGRVPPOzZ

# Wait a few seconds

$ abaco logs G1p783PxpalBB gOvQRGRVPPOzZ

EGe6NKeo8Oy5 DEBUG Action selected: create
EGe6NKeo8Oy5 INFO Created pipeline 1064aaf1-459c-5e42-820d-b822aa4b3990 with update␣
↪token 0df45d5e9e0f31e2
```

Note the pipeline's UUID (`1064aaf1-459c-5e42-820d-b822aa4b3990`). This is needed to configure **PipelineJobs** that reference this **Pipeline**. Also note the update token (`0df45d5e9e0f31e2`). This is needed to update the Pipeline at a later date.

1.4 Update a Pipeline

A pipeline's *components* cannot be updated, but its human-readable name and description can be. To accomplish this, edit the pipeline JSON document with new or amended values and send it to the **Pipelines Manager** along with a valid update token.

```
$ abaco run -m "$(jq -c . my_pipeline.json)" -q token=0df45d5e9e0f31e2 G1p783PxpalBB

e5QKEW8L0BeZ4
```

(continues on next page)

(continued from previous page)

```
# Wait a few seconds

$ abaco logs G1p783PxpalBB e5QKEW8L0BeZ4

EkBWRvV1gKG1a DEBUG Action selected: update
EkBWRvV1gKG1a INFO Updated pipeline 1064aaf1-459c-5e42-820d-b822aa4b3990 with update␣
↪token 0df45d5e9e0f31e2
```

1.5 Retire a Pipeline

A pipeline cannot be deleted, but it can be retired from active service.

Coming soon...

1.6 JSON Schemas

Listing 1: pipeline_manager_create

```
{
        "$schema": "http://json-schema.org/draft-07/schema#",
        "$id": "https://schema.catalog.sd2e.org/schemas/pipeline_manager_create.json",
        "title": "PipelinesDefinition",
        "description": "A Pipeline record. POST to PipelinesManager to create.",
        "definitions": {
                "container_repo": {
                        "type": "string",
                        "description": "a Linux container image repository and tag"
                },
                "agave_app": {
                        "description": "Agave application",
                        "type": "object",
                        "properties": {
                                "id": {
                                        "type": "string",
                                        "description": "the distinct 'app.id' for the␣
↪Agave app"
                                },
                                "inputs": {
                                        "type": "object",
                                        "description": "predefined inputs for jobs␣
↪spawned by the app"
                                },
                                "parameters": {
                                        "type": "object",
                                        "description": "predefined parameters for the␣
↪jobs spawned by the app",
                                        "properties": {
                                                "CONTAINER_IMAGE": {
                                                        "$ref": "#/definitions/
↪container_repo",
                                                        "description": "Linux␣
↪container image repository and tag for the deployed app"
```

(continues on next page)

(continued from previous page)

```
                                                }
                                        }
                                }
                        },
                        "required": ["id", "inputs", "parameters"]
                },
                "container": {
                        "description": "Deployed Linux container",
                        "type": "object",
                        "properties": {
                                "repo": {
                                        "$ref": "#/definitions/container_repo",
                                        "description": "Linux container image repository and tag for the service"
                                },
                                "hash": {
                                        "type": "string",
                                        "description": "Linux container image hash"
                                },
                                "options": {
                                        "type": "object",
                                        "description": "Deployment options"
                                }
                        },
                        "required": ["repo"]
                },
                "service": {
                        "description": "External networked resource",
                        "type": "object",
                        "properties": {
                                "identifier": {
                                        "type": "string",
                                        "description": "An identifier for a specific instance of a service"
                                },
                                "uri": {
                                        "type": "string",
                                        "description": "Canonical URI for the resource",
                                        "format": "uri"
                                },
                                "options": {
                                        "type": "object",
                                        "description": "Additional descriptive attributes"
                                }
                        },
                        "required": ["uri"]
                },
                "abaco_actor": {
                        "description": "Abaco Reactor",
                        "type": "object",
                        "properties": {
                                "id": {
                                        "type": "string",
                                        "description": "Distinct 'actor.id' for the Reactor"
```

(continues on next page)

(continued from previous page)

```
                                },
                                "repo": {
                                        "$ref": "#/definitions/container_repo",
                                        "description": "Linux container image repository and tag for the deployed actor"
                                },
                                "options": {
                                        "type": "object",
                                        "description": "Predefined runtime options for the Reactor"
                                }
                        },
                        "required": ["id", "repo"]
                },
                "pipeline_type": {
                        "description": "the general class of action done by the pipeline",
                        "type": "string",
                        "enum": ["generic-process", "data-transfer", "metadata-management", "primary-etl", "secondary-etl"],
                        "default": "primary-etl"
                },
                "processing_level": {
                        "description": "a data processing level",
                        "type": "string",
                        "enum": ["0", "1", "2", "3"]
                },
                "collections_level": {
                        "description": "a data processing level",
                        "type": "string",
                        "enum": ["reference", "user_file", "challenge_problem", "experiment", "sample", "measurement", "file", "pipeline", "job", "product"]
                }
        },
        "type": "object",
        "properties": {
                "name": {
                        "type": "string"
                },
                "description": {
                        "type": "string"
                },
                "components": {
                        "description": "an unordered array of apps and actors in the pipeline (required)",
                        "type": "array",
                        "items": {
                                "anyOf": [{
                                        "$ref": "#/definitions/agave_app"
                                },
                                {
                                        "$ref": "#/definitions/abaco_actor"
                                },
                                {
                                        "$ref": "#/definitions/service"
                                },
                                {
```

(continues on next page)

(continued from previous page)

```
                                                "$ref": "#/definitions/container"
                                        }
                                ]
                        }
                },
                "collections_levels": {
                        "type": "array",
                        "items": {
                                "$ref": "#/definitions/collections_level"
                        },
                        "description": "level(s) of data input that the pipeline acts upon",
                        "default": []
                },
                "processing_levels": {
                        "type": "array",
                        "description": "level(s) of data product produced by the pipeline",
                        "items": {
                                "$ref": "#/definitions/processing_level"
                        },
                        "default": []
                },
                "pipeline_type": {
                        "items": {
                                "$ref": "#/definitions/pipeline_type"
                        }
                },
                "accepts": {
                        "type": "array",
                        "description": "file types accepted by the pipeline",
                        "items": {
                                "type": "string"
                        },
                        "default": ["*"]
                },
                "produces": {
                        "type": "array",
                        "description": "file types produced by the pipeline",
                        "items": {
                                "type": "string"
                        },
                        "default": ["*"]
                },
                "__options": {
                        "description": "private object for passing runtime options to a pipeline (optional)",
                        "type": "object"
                }
        },
        "required": ["name", "components"],
        "additionalProperties": false
}
```

Listing 2: pipeline_manager_update

```
{
        "$schema": "http://json-schema.org/draft-07/schema#",
        "$id": "https://schema.catalog.sd2e.org/schemas/pipeline_manager_update.json",
        "title": "PipelinesDefinitionUpdate",
        "description": "Update a Pipeline record",
        "type": "object",
        "properties": {
                "uuid": {
                        "description": "The job UUID",
                        "type": "string"
                },
                "body": {"type": "object"},
                "action": {
                        "type": "string",
                        "enum": ["update"]
                },
                "token": {
                        "description": "an authorization token issued when the pipeline was created",
                        "type": "string",
                        "minLength": 16,
                        "maxLength": 17
                },
                "__options": {
                        "type": "object",
                        "description": "an object used to pass runtime options to a reactor (private, optional)"
                }
        },
        "required": ["uuid", "action", "body", "token"],
        "additionalProperties": false
}
```

CHAPTER 2

PipelineJobs Manager

This Reactor manages updates to **PipelineJobs** once they are created by other processes using the **ManagedPipelineJob** and **ReactorManagedPipelineJob** classes.

2.1 Update Job State

PipelineJobs Manager can update a job's state via three mechanisms:

1. Receipt of a JSON-formatted **pipelinejob_manager_event**
2. Receipt of URL parameters sufficient to form a **pipelinejob_manager_event**
3. Receipt of a **agave_job_callback** POST combined with `uuid`, `status` and `token` URL parameters

These named document formats are documented below in *JSONSchemas*.

2.1.1 JSON Event Messages

The default method for updating a PipelineJob's state is to send a JSON message to the Manager actor. Here is an example of a **finish** event for job `1073f4ff-c2b9-5190-bd9a-e6a406d9796a`, which will indicate the job has completed primary computation and archiving steps.

```
{
  "uuid": "1073f4ff-c2b9-5190-bd9a-e6a406d9796a",
  "name": "finish",
  "token": "0dc73dc3ff39b49a"
}
```

This can be sent directly over HTTP like so:

```
curl -XPOST -H "Authorization: Bearer 969d11396c43b0b810387e4da840cb37" \
    --data '{"uuid": "1073f4ff-c2b9-5190-bd9a-e6a406d9796a", \
    "token": "0dc73dc3ff39b49a",\
```

(continues on next page)

(continued from previous page)

```
    "name": "finish"}' \
    https://api.tacc.cloud/actors/v2/<actorId>/messages
```

It can also be sent from within a Recator like so:

```
rx = Reactor()
manager_id = '<actorId>'
finish_mes = { 'uuid': '1073f4ff-c2b9-5190-bd9a-e6a406d9796a',
               'name': 'finish',
               'token': '0dc73dc3ff39b49a'}
rx.send_message(manager_id, finish_mes)
```

2.1.2 URL Parameters Event

The `uuid`, and `event`, `token` fields can be sent as URL parameters in an HTTP POST. The contents of the POST body will be attached to the event if one is present. Our **finish** event expressed as URL parameters looks like:

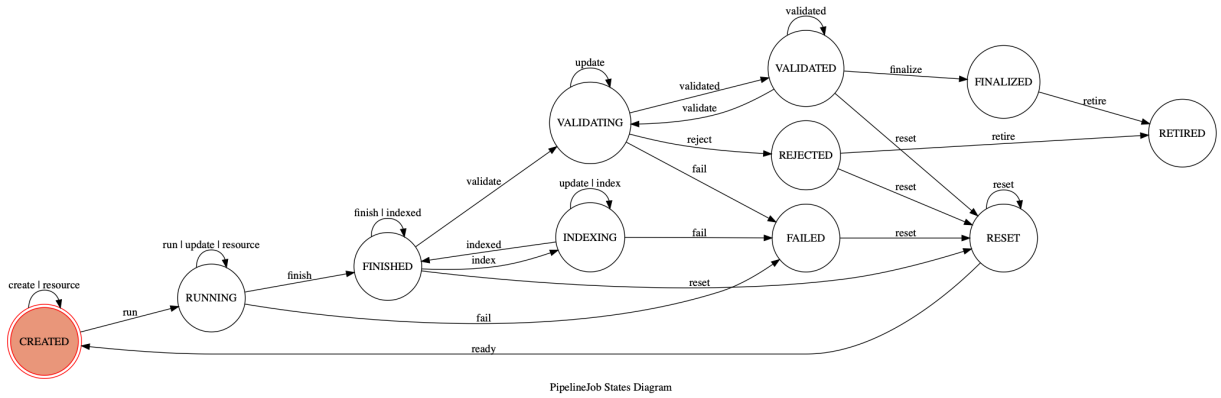
```
curl -XPOST --data '{"arbitrary": "key value data"}' \
    https://api.tacc.cloud/actors/v2/<actorId>/messages?uuid=1073f4ff-c2b9-5190-bd9a-
↪e6a406d9796a&\
    event=finish&token=0dc73dc3ff39b49a
```

2.1.3 Agave Jobs Notification

HTTP POST body and URL parameters are combined to link the Agave Jobs system with **PipelineJobs**, which is quite handy as Agave jobs often are enlisted to do the computational heavy lifting in analysis workflows. This approach is demonstrated in the **demo-jobs-reactor-app** repository.

2.2 PipelineJobs Events

The state of every PipelineJob proceeds through a defined lifecycle, where transitions occur in response to receipt of named events. This is illustrated in the following image:



PipelineJob States Diagram

PipelineJobs Manager accepts any of the events (lower case words attached to the graph edges) save for **create**, which is reserved for other agents.

2.3 Authentication

POSTs to a **PipelineJobs Manager** must be authenticated by one of two means:

1. Send a valid TACC.cloud Oauth2 Bearer token with the request
2. Include a special URL parameter called a **nonce** with the HTTP request

2.4 JSON Schemas

Listing 1: agave_job_callback

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "$id": "https://schema.catalog.sd2e.org/schemas/agave_job_callback.json",
    "title": "AgaveJobsPost",
        "description": "A job status POST from the Agave API jobs service",
    "type": "object",
    "properties": {
        "owner": {
            "type": "string"
        },
        "appId": {
            "type": "string"
        },
        "executionSystem": {
            "type": "string"
        },
        "archivePath": {
            "type": "string"
        },
        "archiveSystem": {
            "type": "string"
        },
        "status": {
            "type": "string"
        },
        "inputs": {},
        "parameters": {},
        "_links": {}
    },
    "required": [
        "appId",
        "executionSystem",
        "owner",
        "archivePath",
        "archiveSystem",
        "status",
        "inputs",
        "parameters",
        "_links"
    ]
}
```

Listing 2: pipelinejob_manager_event

```
{
        "$schema": "http://json-schema.org/draft-07/schema#",
        "$id": "https://schema.catalog.sd2e.org/schemas/pipelinejob_manager_event.json
↪",
        "title": "PipelineJobManagerEvent",
        "description": "A state-change event for a PipelineJob",
        "type": "object",
        "properties": {
                "uuid": {
                        "$ref": "pipelinejob_uuid.json"
                },
                "name": {
                        "$ref": "pipelinejob_eventname.json"
                },
                "data": {
                        "description": "Additional information to attach to the event␣
↪(optional)",
                        "type": "object"
                },
                "token": {
                        "$ref": "update_token.json"
                }
        },
        "required": ["uuid", "name"],
        "additionalProperties": false
}
```

CHAPTER 3

PipelineJobs Indexer

This Reactor indexes the contents of **ManagedPipelineJob** archive paths. It implements two actions: **index** and **indexed**. It is normally run automatically via **PipelineJobs Manager** when a job enters the **FINISHED** state, but can also be activated on its own.

3.1 Index a Job

PipelineJobs Indexer can receive an **index** request via:

1. A JSON-formatted **pipelinejob_index** document
2. URL parameters that replicate a **pipelinejob_index** document

Here are the critical fields to request indexing:

1. `uuid` ID for job to be indexed (must validate as a known job)
2. `name` This is always **index**
3. `token` The job's update token (optional for now)
4. `level` The processing level for output files (default: **1**)
5. `filters` List of **url-encoded** Python regex that select a subset of `archive_path`

3.1.1 Index Request as JSON

This message will index outputs of job `1079f67e-0ef6-52fe-b4e9-d77875573860` as level "2" products, sub-selecting only files matching `sample\.uw_biofab\.141715` and `sample-uw_biofab-141715`.

```
{
    "uuid": "1079f67e-0ef6-52fe-b4e9-d77875573860",
    "name": "index",
    "filters": [
```

(continues on next page)

(continued from previous page)

```
        "sample%5C.uw_biofab%5C.141715",
        "sample-uw_biofab-141715"
    ],
    "level": "2",
    "token": "0dc73dc3ff39b49a"
}
```

3.1.2 Index Request as URL Params

```
curl -XPOST \
    https://<tenantUrl>/actors/v2/<actorId>/messages?uuid=1073f4ff-c2b9-5190-bd9a-
↪e6a406d9796a&\
    level=2&token=0dc73dc3ff39b49a&name=index --data '{"filters": ["sample.uw_biofab.
↪141715", "sample-uw_biofab-141715"]}'
```

Note: Remember that `filters` cannot currently be passed as URL parameters.

3.2 Mark as Indexed

PipelineJobs Indexer can receive an **index** request via JSON message or URL parameters. Here is an example.

```
{
    "uuid": "1079f67e-0ef6-52fe-b4e9-d77875573860",
    "name": "indexed",
    "token": "0dc73dc3ff39b49a"
}
```

PipelineJobs Indexer sends itself an **indexed** message after completing an indexing action. Thus, it is not usually necessary to send one manually and in fact, should be avoided. Documentation on the **indexed** event is included here mostly for the sake of completeness.

3.3 Authentication

POSTs to a **PipelineJobs Indexer** must be authenticated by one of two means:

1. Send a valid TACC.cloud Oauth2 Bearer token with the request
2. Include a special URL parameter called a **nonce** with the HTTP request

3.4 JSON Schemas

Listing 1: pipelinejob_index

```
{
        "$schema": "http://json-schema.org/draft-07/schema#",
        "$id": "https://schema.catalog.sd2e.org/schemas/pipelinejob_index.json",
```

(continues on next page)

(continued from previous page)

```
        "title": "PipelineJobIndexEvent",
        "description": "Request indexing of a completed PipelineJob's archive path",
        "type": "object",
        "properties": {
                "uuid": {
                        "$ref": "pipelinejob_uuid.json"
                },
                "name": {
                        "type": "string",
                        "enum": [
                                "index"
                        ]
                },
        "filters": {
            "type": "array",
            "description": "List of Python regular expressions defining which output files to associate with the job. Omit entirely if you do not want to apply filtering.",
            "items": {
                "type": "string"
            }
        },
        "level": {
            "$ref": "processing_level.json"
                },
                "token": {
                        "$ref": "update_token.json"
                }
        },
        "required": ["uuid", "name"],
        "additionalProperties": false
}
```

Listing 2: pipelinejob_indexed

```
{
        "$schema": "http://json-schema.org/draft-07/schema#",
        "$id": "https://schema.catalog.sd2e.org/schemas/pipelinejob_indexed.json",
        "title": "PipelineJobIndexedEvent",
        "description": "Mark PipelineJob indexing as completed.",
        "type": "object",
        "properties": {
                "uuid": {
                        "$ref": "pipelinejob_uuid.json"
                },
                "name": {
                        "type": "string",
                        "enum": [
                                "indexed"
                        ]
                },
                "token": {
                        "$ref": "update_token.json"
                }
        },
        "required": ["uuid", "name"],
```

(continues on next page)

(continued from previous page)

```
        "additionalProperties": false
}
```

CHAPTER 4

PipelineJobs Agave Proxy

This Reactor provides a generalized proxy for running Agave API jobs such that their inputs, parameterization, and outputs are connected to (and thus discoverable from within) the Data Catalog.

4.1 Register Agave App as a Pipeline

Before an Agave App can be run by this proxy, three things must happen:

1. It must be architected to fit the PipelineJobs workflow
2. It must be public or shared with user **sd2eadm**
3. It must be registered as a Data Catalog `Pipeline`

4.1.1 App Architecture

The app must generate filenames that are distinguishable between runs. This is enforced to prevent accidentallly overwriting of files when multiple jobs share an archiving destination. Furthermore, the app definition and any interior runtime logic must use fully-qualified Agave files URLs to define inputs. Finally, the app's `id` must be unique not only in the **Agave Apps Catalog** (this is automatically enforced) but also in the **Data Catalog Pipelines** collection.

4.1.2 Share or Publish the App

Coming soon...

4.1.3 Registering a Pipeline

Coming soon...

4.2 Launching a Managed Agave Job

Construct and send a message including the following components to the **PipelineJobs Agave Proxy** Reactor.

1. An Agave job definition
2. A metdata linkage parameter
3. Optional control parameters

Note: The **agave_pipelinejob** format is documented in *JSONSchemas*.

4.2.1 Agave Job Definition

The Agave job definition must be included as as subdocument in the message. To illustrate this, start with a basic Agave job definition: Here is an example for an imaginary Agave app `tacobot9000-0.1.0u1`.

```
{
  "appId": "tacobot9000-0.1.0u1",
  "name": "TACObot job",
  "inputs": {"file1": "agave://data.tacc.cloud/examples/tacobot/test1.txt"},
  "parameters": {"salsa": true, "avocado": false, "cheese": true},
  "maxRunTime": "01:00:00"
}
```

To launch this via the Agave, this document would be sent directly to the `/apps` endpoint. To send it instead to the proxy, move it to key `job_definition` in a JSON document.

```
{
    "job_definition": {
            "appId": "tacobot9000-0.1.0u1",
            "name": "TACObot job",
            "inputs": {
                    "file1": "agave://data.tacc.cloud/examples/tacobot/test1.txt"
            },
            "parameters": {
                    "salsa": true,
                    "avocado": false,
                    "cheese": true
            },
            "maxRunTime": "01:00:00"
    }
}
```

4.2.2 Metadata Linkage Parameter

An explicit linkage to objects in the Data Catalog must be established. This is done via the `parameters` key, which must contain a valid value for one of the following:

- `experiment_id`
- `sample_id`
- `measurement_id`

Either single values or an array of values may be passed, and either the readable text value may be provided or the corresponding UUID.

Which Parameter to Pass

A PipelineJob is always linked to a set of measurements by way of the linkage parameter. The job's archive path is also determined by the linkage parameter. To illustrate:.

If a job's `measurement_id=['measurement.tacc.1234', 'measurement.tacc.2345']`, it will linked to these two measurements and its archive path will end with a hash of the two `measurement_id` values.

Assuming those measurements are children of `sample.tacc.abcde` and the only linkage parameter sent was `sample_id='sample.tacc.abcdef'`, the job will still be linked to all the child measurements of that sample. Its archive path will end with a hash of `sample.tacc.abcde`. Howver, if both measurement_id and sample_id are passed, the linkages are made to the specified measurement(s) while the archive path is a function of the sample_id value(s).

For experiment_id, the specific samples are linked to the job and the archive path is a function of experiment_id value(s).

This design allows files generated by the job to be linked to only one level of the metadata hiearchy, while allowing collection of outputs at higher levels of organization in the file system.

Here is a worked example of the current example job request, as it stands:

```
{
    "parameters": {
            "sample_id": "sample.tacc.abcde"
    },
    "job_definition": {
            "appId": "tacobot9000-0.1.0u1",
            "name": "TACObot job",
            "inputs": {
                    "file1": "agave://data.tacc.cloud/examples/tacobot/test1.txt"
            },
            "parameters": {
                    "salsa": true,
                    "avocado": false,
                    "cheese": true
            },
            "maxRunTime": "01:00:00"
    }
}
```

4.2.3 Additional Control Parameters

Job behavior can be refined with additional control parameters.

instanced

Each PipelineJob has a distinct archive path derived from its Pipeline UUID, the `data` dictionary passed at job `init()` and/or `setup()`, and a function of its linkage parameters to experiments, samples, or measurements. To avoid inadvertent over-writes, the archive path is extended with an *instancing directory* named in the form `adjective-animal-YYYYMMDDTHHmmssZ`. To avoid use of the instancing directory, include `instanced: false` in the job request message.

Example: `"instanced": false`

index_patterns

The default behavior of the PipelineJobs System is to index every file found under a job's archive path to be linked to that specific job. To subselect only specific files, it is possible to include one or more Python regular expressions in `index_patterns`. Only files matching these patterns will be linked to the job.

Example: `"index_patterns": []`

processing_level

The default behavior of the PipelineJobs System is to index files under a job's archive path as processing level "1". To change this, an alternative `processing_level` may be passed in the job request message.

Example: `"processing_level": "2"`

Note: Only one automatic indexing configuration can be active for a given job. Additional indexing actions with other configurations may be initiated by sending a message directly to **PipelineJobs Indexer**

4.3 Job Life Cycle

Here is complete record from the Pipelines system showing how the information from job creation and subsequent events is stored and discoverable. A few key highlights:

- The top-level `data` field holds the original parameterization of the job
- Three events are noted in the `history`: create, run, finish
- The actor and execution for the managing instance of **PipelineJobs Agave Proxy** are available under `agent` and `task`, respectively

```
{
    "agent": "https://api.tacc.cloud/actors/v2/G46vjoAVzGkkz",
    "archive_path": "/products/v2/103f877a7ab857d182807b75af4eab6e/
↪106bd127e2d257acb9be11ed06042e68/eligible-awk-20181127T173243Z",
    "archive_system": "data-sd2e-community",
    "data": {
        "appId": "urrutia-novel_chassis_app-0.1.0",
        "archivePath": "",
        "inputs": {
            "file1": "agave://data.tacc.cloud/examples/tacobot/test1.txt"
        },
        "maxRunTime": "01:00:00",
        "name": "TACObot job",
        "parameters": {
            "avocado": false,
            "cheese": true,
            "salsa": true
        }
    },
    "derived_from": [
        "1022efa3-4480-538f-a581-f1810fb4e0c3"
```

(continues on next page)

(continued from previous page)


```
    ],
    "generated_by": [
        "106bd127-e2d2-57ac-b9be-11ed06042e68"
    ],
    "history": [
        {
            "data": {
                "appId": "tacobot9000-0.1.0u1",
                "inputs": {
                    "file1": "agave://data.tacc.cloud/examples/tacobot/test1.txt"
                },
                "maxRunTime": "01:00:00",
                "name": "TACObot job",
                "parameters": {
                    "avocado": false,
                    "cheese": true,
                    "salsa": true
                }
            },
            "date": "2018-12-08T00:08:32.000+0000",
            "name": "create"
        },
        {
            "data": {
                "appId": "tacobot9000-0.1.0u1",
                "archive": true,
                "archivePath": "/products/v2/103f877a7ab857d182807b75af4eab6e/
↪106bd127e2d257acb9be11ed06042e68/eligible-awk-20181127T173243Z",
                "archiveSystem": "data-tacc-cloud",
                "batchQueue": "normal",
                "created": "2018-12-07T18:08:37.000-06:00",
                "endTime": null,
                "executionSystem": "hpc-tacc-stampede2",
                "id": "7381691026605150696-242ac11b-0001-007",
                "inputs": {
                    "file1": "agave://data.tacc.cloud/examples/tacobot/test1.txt"
                },
                "lastUpdated": "2018-12-07T18:09:40.000-06:00",
                "maxRunTime": "01:00:00",
                "memoryPerNode": 1,
                "name": "TACObot job",
                "nodeCount": 1,
                "outputPath": "tacobot/job-7381691026605150696-242ac11b-0001-007-
↪TACObot-job",
                "owner": "tacobot",
                "parameters": {
                    "avocado": false,
                    "cheese": true,
                    "salsa": true
                },
                "processorsPerNode": 1,
                "startTime": null,
                "status": "RUNNING",
                "submitTime": "2018-12-07T18:09:40.000-06:00"
            },
            "date": "2018-12-08T00:10:12.000+0000",
            "name": "run"
```


(continues on next page)

(continued from previous page)


```
        },
        {
            "data": {
                "appId": "tacobot9000-0.1.0u1",
                "archive": true,
                "archivePath": "/products/v2/103f877a7ab857d182807b75af4eab6e/
↪106bd127e2d257acb9be11ed06042e68/eligible-awk-20181127T173243Z",
                "archiveSystem": "data-tacc-cloud",
                "batchQueue": "normal",
                "created": "2018-12-07T18:08:37.000-06:00",
                "endTime": null,
                "executionSystem": "hpc-tacc-stampede2",
                "id": "7381691026605150696-242ac11b-0001-007",
                "inputs": {
                    "file1": "agave://data.tacc.cloud/examples/tacobot/test1.txt"
                },
                "lastUpdated": "2018-12-07T18:53:20.000-06:00",
                "maxRunTime": "01:00:00",
                "memoryPerNode": 1,
                "name": "TACObot job",
                "nodeCount": 1,
                "outputPath": "tacobot/job-7381691026605150696-242ac11b-0001-007-
↪TACObot-job",
                "owner": "tacobot",
                "parameters": {
                    "avocado": false,
                    "cheese": true,
                    "salsa": true
                },
                "processorsPerNode": 1,
                "startTime": "2018-12-07T18:09:49.000-06:00",
                "status": "FINISHED",
                "submitTime": "2018-12-07T18:09:40.000-06:00"
            },
            "date": "2018-12-08T00:53:45.000+0000",
            "name": "finish"
        }
    ],
    "last_event": "finish",
    "pipeline_uuid": "106bd127-e2d2-57ac-b9be-11ed06042e68",
    "session": "casual-bass",
    "state": "FINISHED",
    "task": "https://api.tacc.cloud/actors/v2/G46vjoAVzGkkz/executions/Myp6wvklV0zgQ",
    "updated": "2018-12-08T00:53:45.000+0000",
    "uuid": "10743f9e-f5ae-5b4c-859e-6774ef4ab08b"
}
```


4.4 JSON Schemas

Listing 1: agave_pipelinejob


```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "$id": "https://schema.catalog.sd2e.org/schemas/agave_pipelinejob.json",
```


(continues on next page)

(continued from previous page)

```
    "title": "AgavePipelineJob",
    "description": "Launch an Agave job as a PipelineJob",
    "type": "object",
    "definitions": {
        "datacatalog_field": {
            "type": "object",
            "anyOf": [
                {
                    "required": [
                        "sample_id"
                    ]
                },
                {
                    "required": [
                        "experiment_design_id"
                    ]
                },
                {
                    "required": [
                        "experiment_id"
                    ]
                },
                {
                    "required": [
                        "measurement_id"
                    ]
                }
            ],
            "properties": {
                "sample_id": {
                    "type": "string",
                    "pattern": "^sample.(uw_biofab|transcriptic|ginkgo|emerald)."
                },
                "experiment_design_id": {
                    "type": "string",
                    "$ref": "experiment_reference.json"
                },
                "experiment_id": {
                    "type": "string",
                    "pattern": "^experiment.(uw_biofab|transcriptic|ginkgo|emerald)."
                },
                "measurement_id": {
                    "type": "string",
                    "pattern": "^measurement.(uw_biofab|transcriptic|ginkgo|emerald)."
                }
            }
        }
    },
    "properties": {
        "parameters": {
            "$ref": "#/definitions/datacatalog_field"
        },
        "job_definition": {
            "description": "An Agave API job definition",
            "type": "object"
        },
        "archive_path": {
```

(continues on next page)

(continued from previous page)

```
            "description": "Optional Agave URN defining the job's archive path",
            "$ref": "agave_files_uri.json"
        },
        "instanced": {
            "description": "Whether the generated archive path should be instanced with a randomized session",
            "type": "boolean",
            "value": true
        },
        "data": {
            "description": "Optional dict-like object describing the job's run-time parameterization",
            "type": "object"
        },
        "index_patterns": {
            "type": "array",
            "description": "List of Python regular expressions defining which output files to associate with the job. Omit entirely if you do not want to apply filtering.",
            "items": {
                "type": "string"
            }
        },
        "processing_level": {
            "description": "Defaults to '1' if not provided",
            "$ref": "processing_level.json"
        }
    },
    "required": [
        "job_definition",
        "parameters"
    ],
    "additionalProperties": false
}
```

CHAPTER 5

HTTP Authentication

All POSTs to PipelineJobs components must be authenticated. There are two mechanisms by which this can be accomomplished.

1. Send a valid TACC.cloud Oauth2 Bearer token with the request
2. Include a special URL parameter called a **nonce** with the HTTP request

CHAPTER 6

Send a Bearer Token

Requests authenticated using a Bearer token run as the TACC.cloud identity of the user that issued the token. This is usually your account (or a role account if you have appropriate credentials).

6.1 Usage Example

```
curl -XPOST -H "Authorization: Bearer 969d11396c43b0b810387e4da840cb37" \
    --data '{"uuid": "1073f4ff-c2b9-5190-bd9a-e6a406d9796a", \
    "token": "0dc73dc3ff39b49a",\
    "name": "finish"}' \
    https://<tenantUrl>/actors/v2/<actorId>/messages
```

CHAPTER 7

Use a Nonce

```
curl -XPOST --data '{"arbitrary": "key value data"}' \
    https://<tenantUrl>/actors/v2/<actorId>/messages?uuid=1073f4ff-c2b9-5190-bd9a-
↪e6a406d9796a&\
    name=finish&token=0dc73dc3ff39b49a&\
    x-nonce=TACC_XXXXxxxxYz
```

CHAPTER 8

Authorization Tokens

Update and delete actions for Pipelines and PipelineJobs are authorized via an additional token scoped to the Pipelines system. There are two types:

- Document update token
- Administrative action token

A document update token authorizes management of **one specific** Pipeline or Job. The token for a given document is returned after each update or management action and allows future actions to be taken at any time in the future. An administrative action token authorizes **any action** for **any Pipeline or Job**. They are obtained via an external process, and only by persons who possessing the **Administrator Token API Key**. Because they are powerful, administrative action tokens expire after 30 seconds.

8.1 Sending a Token

A token must be sent with the requested action or event. It can be included in the request message as field (`"token": "<token>"`) or included as a URL parameter `token=<token>`.

CHAPTER 9

Reactor+App with PipelineJobs Integration

This is an example implementation of a Reactor that launches an Agave app, where both Reactor and the App integrate with the PipelineJobs system to update the state and history of a PipelineJob.

9.1 Key Aspects

Note the following as you go through the code and configuration of this project:

1. Use of the `ManagedPipelineJob` class to instantiate the job
2. Addition of Agave job notifications allowing the job to send updates
3. Configuration of this Reactor via the `mongodb` and `pipelines` stanzas in `config.yml`
4. Use of Abaco API keys, or _nonces_, in `secrets.json`
5. Leveraging shared Abaco Reactors

9.2 Deploying your own Reactor

Note: Familiarity with configuration and deployment of Abaco Reactor is assumed.

Obtain the following values from the Infrastructure team for *secrets.json*:

- `_REACTOR_MONGODB_AUTHN`
- `_REACTOR_PIPELINES_JOB_MANAGER_NONCE`

Confirm the following values with the Infrastructure team for *config.yml*:

- `pipelines.job_manager_id`
- `pipelines.job_indexer_id`

Danger: Never include values from **secrets.json** in `config.yml` file. If you do this, they will be committed forever to a publicly-accessible Docker image when the Reactor is deployed.

Once you've set up the configuration and secrets, build then deploy your Reactor.

9.3 Send a Message

```
abaco run -m '{"data":{"key1":"value 1"}, "sample_id": "sample.tacc.20001}' <ACTOR_ID>
```

CHAPTER 10

Indices and tables

- genindex
- modindex
- search