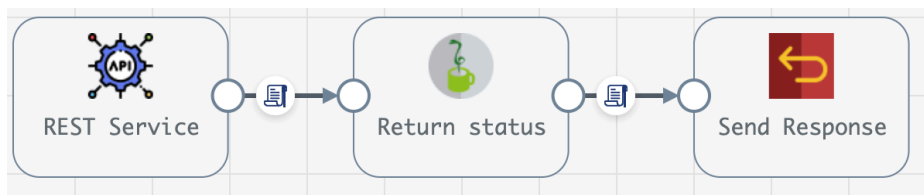


Polling a REST API Until Complete

The pipelines for this example can be downloaded from [here](#)..

Consider a Service named that returns a response `{"status": "still working"}` the first four times it is called, and the response `{"status": "complete"}` the fifth time it is called.

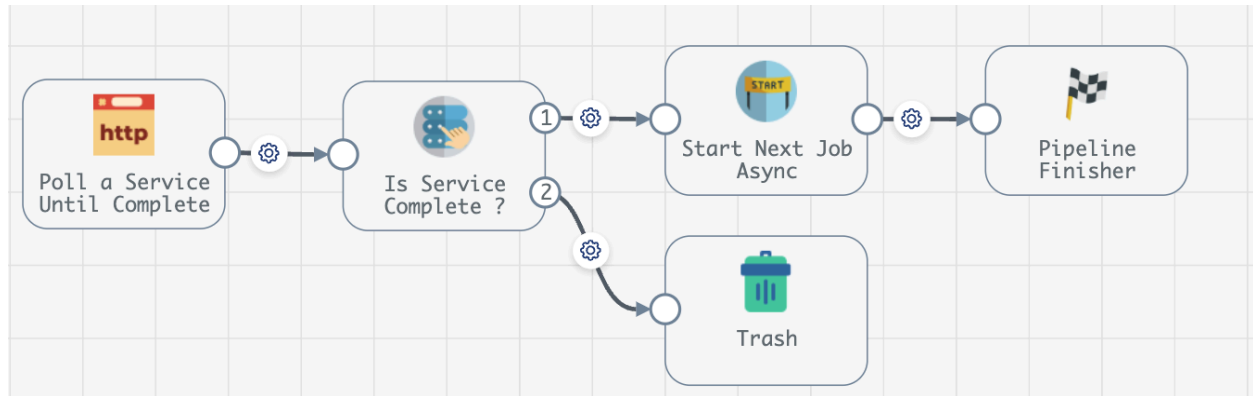
Here is a Data Collector Microservice Pipeline named `ServiceThatNeedsToBeCalledFiveTimes` that implements this behavior. (See the `Return Status` stage's script for the details of saving state across calls):



Here is a series of `curl` calls that confirms the behavior of the service:

```
mark@SAG-FV43FHGCVG ~ % curl -X GET http://portland.onefoursix.com:18630/public-rest/v1/gateway/call_many_times
-H "Content-Type:application/json"
{"statusCode":200,"data":{"status":"still working","counter":1},"error":{}}
mark@SAG-FV43FHGCVG ~ % curl -X GET http://portland.onefoursix.com:18630/public-rest/v1/gateway/call_many_times
-H "Content-Type:application/json"
{"statusCode":200,"data":{"status":"still working","counter":2},"error":{}}
mark@SAG-FV43FHGCVG ~ % curl -X GET http://portland.onefoursix.com:18630/public-rest/v1/gateway/call_many_times
-H "Content-Type:application/json"
{"statusCode":200,"data":{"status":"still working","counter":3},"error":{}}
mark@SAG-FV43FHGCVG ~ % curl -X GET http://portland.onefoursix.com:18630/public-rest/v1/gateway/call_many_times
-H "Content-Type:application/json"
{"statusCode":200,"data":{"status":"still working","counter":4},"error":{}}
mark@SAG-FV43FHGCVG ~ % curl -X GET http://portland.onefoursix.com:18630/public-rest/v1/gateway/call_many_times
-H "Content-Type:application/json"
{"statusCode":200,"data":{"status":"complete","counter":0},"error":{}}
mark@SAG-FV43FHGCVG ~ %
```

Here is an example pipeline named `PollServiceUntilComplete` that polls the `ServiceThatNeedsToBeCalledFiveTimes` REST API, with a user-configurable sleep time between calls, until it receives the response `{"status": "complete"}`, whereupon it starts a downstream Job and then exits.



The `Poll a Service Until Complete` stage is an HTTP Client origin. Here are the two key properties set in that stage:

< General **HTTP** Pagination Credentials OAuth 2 Proxy TLS

Resource URL ⓘ `http://portland.onefoursix.com:18630/public-rest/v1/gateway/call_many_times`

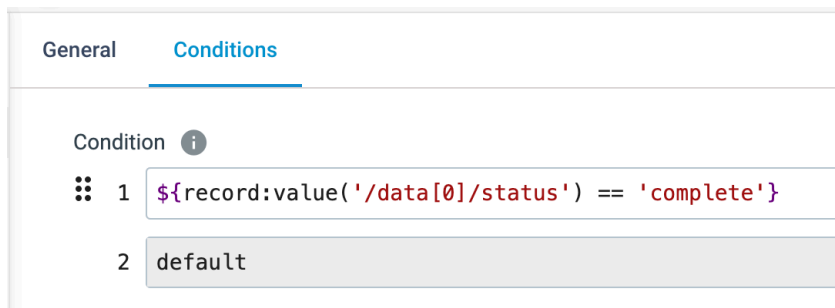
Mode Polling <>

Polling Interval (ms) 5000

HTTP Method ⓘ GET <>

Those settings will poll the target service every five seconds until the pipeline is stopped.

The `Is Service Complete ?` is a Stream Selector with these conditions:



General		Conditions
Condition ⓘ		
⋮	1	<code>\${record:value('/data[0]/status')} == 'complete'}</code>
	2	default

On the first four calls to the target service, the response is `{"status": "still working"}` and the default condition of the Stream Selector routes the record to the trash.

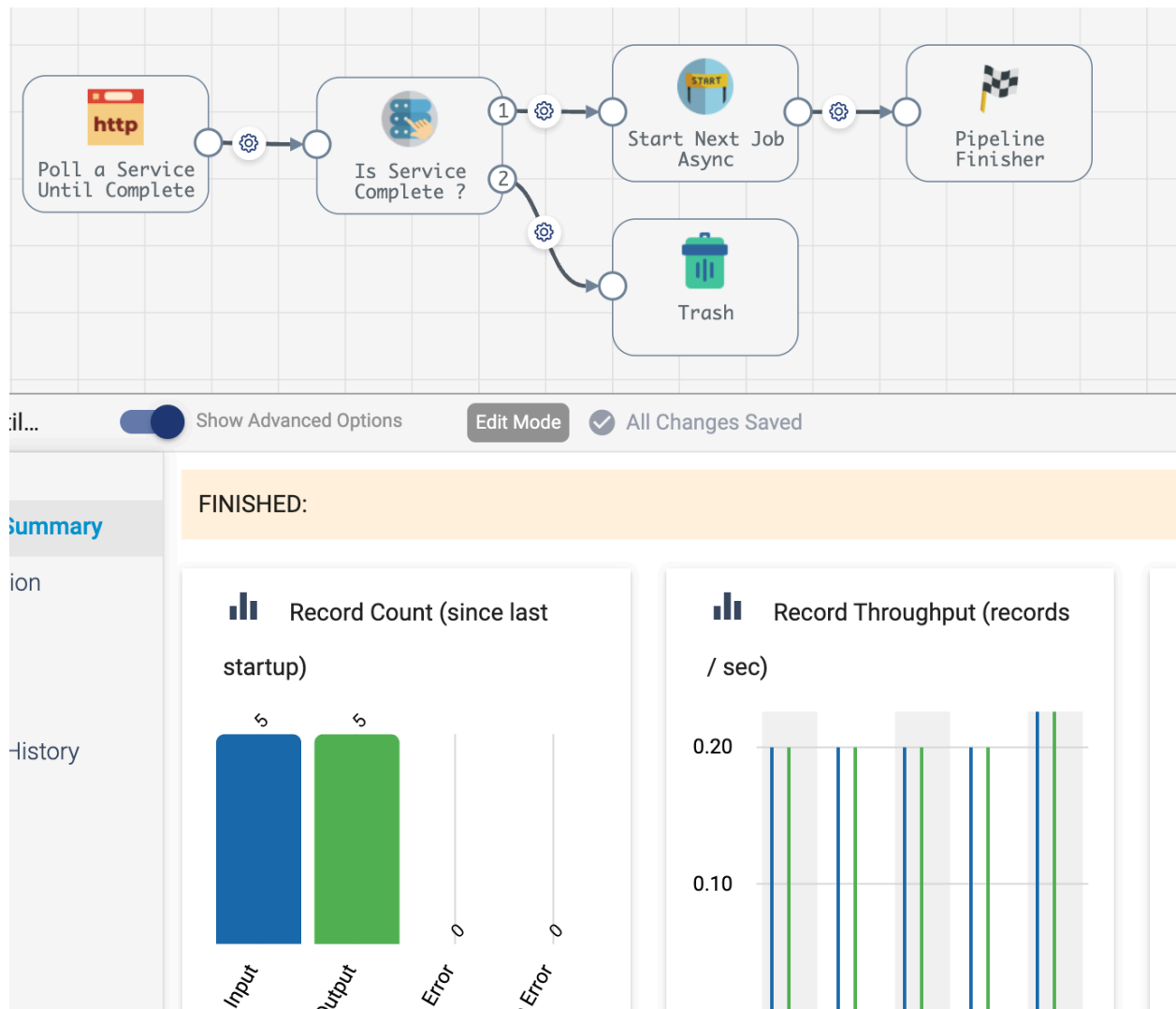
On the fifth call to the target service, the first condition of the Stream Selector evaluates to `true` and routes the record to a Start Jobs Processor to launch the next step in the process and then executes a Pipeline Finisher to stop the pipeline.

Note that the Start Jobs Processor is configured to start its Job in the background (async), so this pipeline will complete right away. This is the setting on the Start Job's `Job` tab:

Run in Background ⓘ ☒

Make sure to set the Job ID for the Start Jobs Processor to one of your own Jobs, or delete that stage altogether if you just want to test that this pipeline will stop on its fifth call.

We can run the pipeline to confirm it terminates after calling the target service 5 times:



And we can see that 4 records were ignored, and 1 record was passed to the Start Jobs stage:

