Splunk Search Mockup

Introduction

This example describes a two-pipeline design pattern for a simulated Splunk Search.

The first pipeline calls a mocked Splunk Search API, passing in a `search_keyword`, and receives an immediate synchronous response that contains a `sid` (a search ID) , although the search will take some time to complete asynchronously.

A second pipeline polls a different mocked Splunk Search API, waiting for the search identified by the `sid` to complete. Once that particular search has completed, this pipeline calls a third mocked Splunk Search API to retrieve the results of the search and writes the results to a file in an S3 bucket, partitioned by `search_keyword` and `sid`.

Job Templates are used so end users can easily launch multiple different searches concurrently, while shielding them from the pipelines' details.

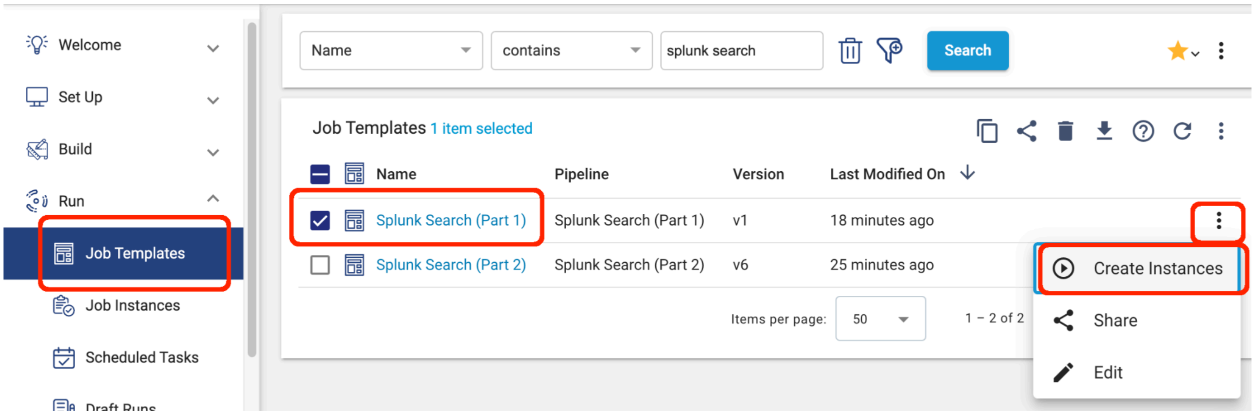
Download the Example Pipelines

You can download the pipelines for this example from here. Installation and configuration steps follow below.

High-Level User Walkthrough of the Example

Here is a high-level user walkthrough of the example

- Navigate to `Run > Job Templates` and select `Create Instances` for the template named `Splunk Search (Part 1):`



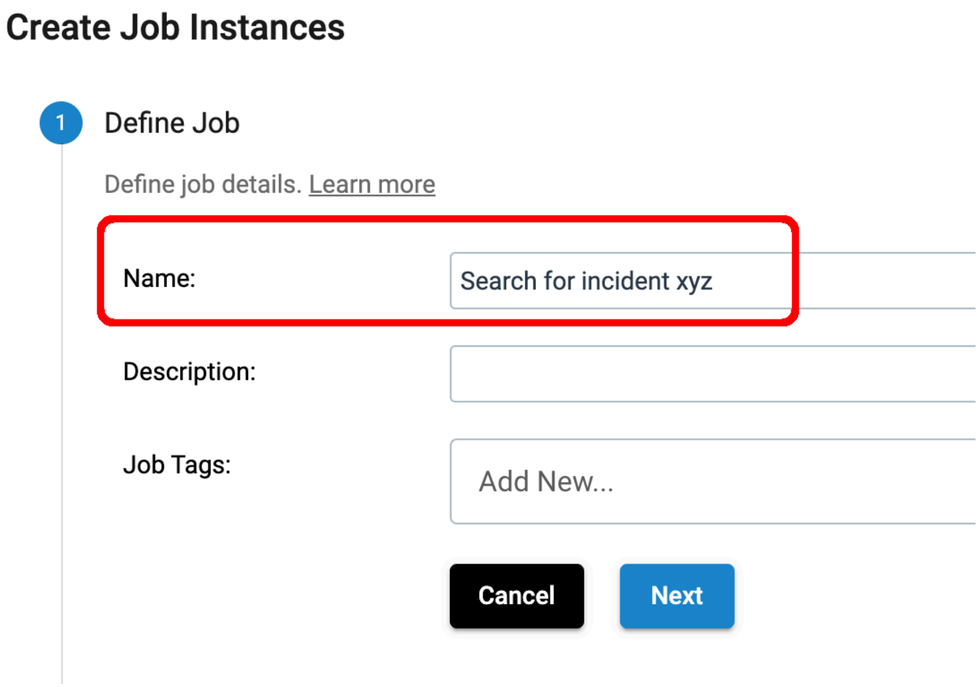
Name | contains | splunk search

Job Templates 1 item selected

Name	Pipeline	Version	Last Modified On
☑ Splunk Search (Part 1)	Splunk Search (Part 1)	v1	18 minutes ago
□ Splunk Search (Part 2)	Splunk Search (Part 2)	v6	25 minutes ago

Items per page: 50   1 – 2 of 2

- Provide a name for the instance; I'll use `Search for incident xyz`:



Create Job Instances

1 Define Job

Define job details. Learn more

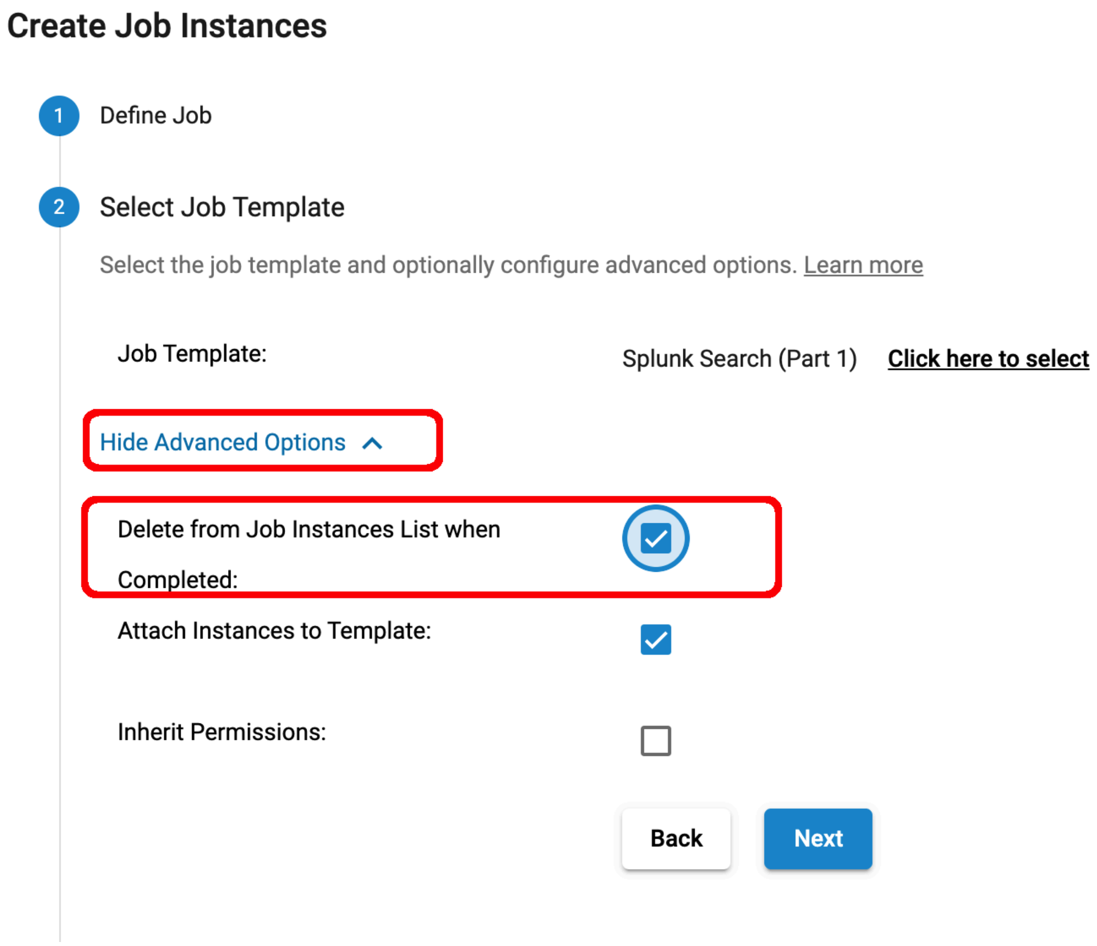
Name: Search for incident xyz

Description:

Job Tags: Add New...

Click Next

- Expand the Advanced Options and set the checkbox `Delete from Job Instances List when Completed`:



Create Job Instances

1 Define Job

2 Select Job Template

Select the job template and optionally configure advanced options. Learn more

Job Template: Splunk Search (Part 1) Click here to select

Hide Advanced Options

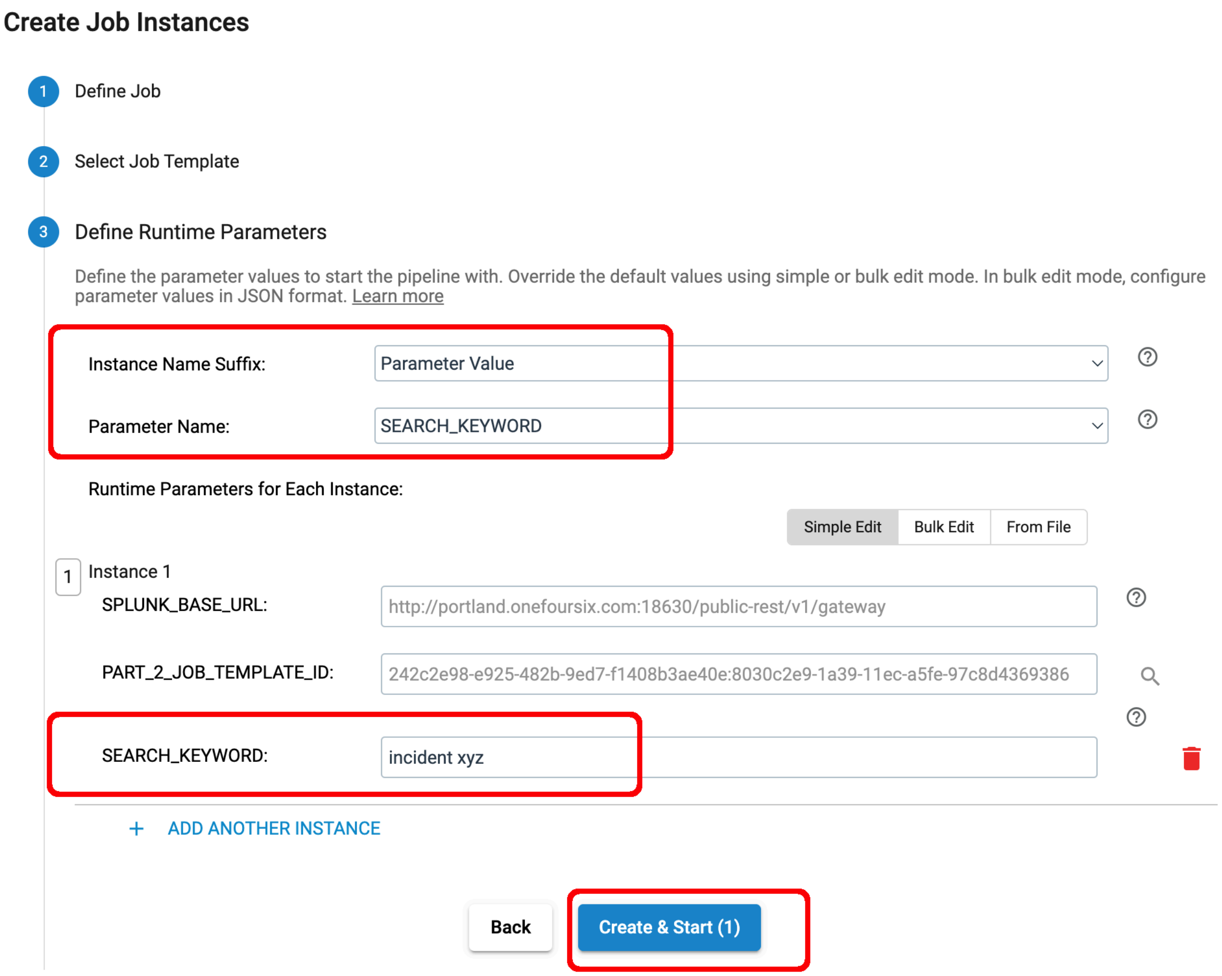
Delete from Job Instances List when Completed: ☑

Attach Instances to Template: ☑

Inherit Permissions: □

Click Next

- Set a keyword for the search like `incident xyz`, and specify that the keyword should be used as a suffix for the template instance name:



Create Job Instances

1 Define Job

2 Select Job Template

3 Define Runtime Parameters

Define the parameter values to start the pipeline with. Override the default values using simple or bulk edit mode. In bulk edit mode, configure parameter values in JSON format. Learn more

Instance Name Suffix: Parameter Value

Parameter Name: SEARCH_KEYWORD

Runtime Parameters for Each Instance:

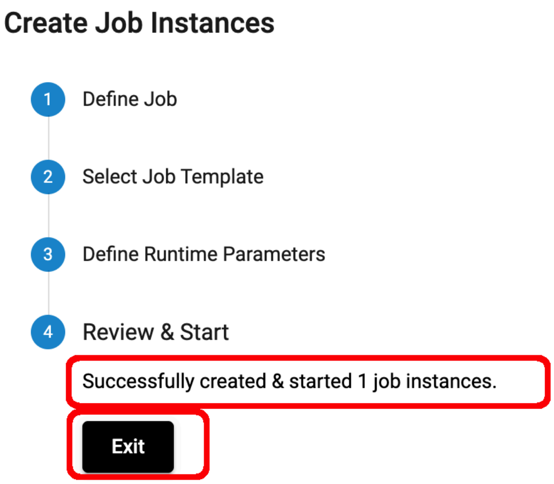
Instance 1

SPLUNK_BASE_URL: http://portland.onefoursix.com:18630/public-rest/v1/gateway

PART_2_JOB_TEMPLATE_ID: 242c2e98-e925-482b-9ed7-f1408b3ae40e:8030c2e9-1a39-11ec-a5fe-97c8d4369386

SEARCH_KEYWORD: incident xyz

Click Create & Start



Create Job Instances

1 Define Job

2 Select Job Template

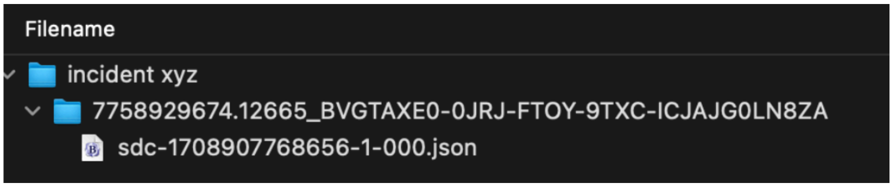
3 Define Runtime Parameters

4 Review & Start

Successfully created & started 1 job instances.

Click Exit

- Within 30 seconds or so (the search is simulated to take 25 seconds) , a file should be written to the designated S3 bucket, with the path `<search_keyword>/<sid>`, like this:



Filename
incident xyz
7758929674.12665_BVGTAXE0-0JRJ-FTOY-9TXC-ICJAJG0LN8ZA
sdc-1708907768656-1-000.json

- The search results written to that file are:.

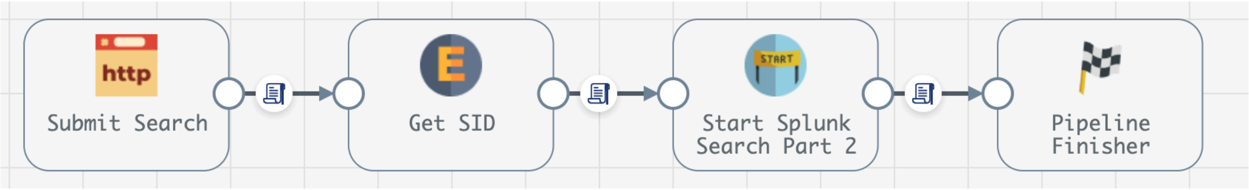
```
{
  "error": [],
  "search_details": {
    "search_keyword": "incident xyz",
    "sid": "2789752210.05484_X2NHKKDM-4WU5-KNMD-PMAA-OKZGKUYVONKF",
    "status": "complete"
  },
  "search_results": [
    {
      "content": "AJS29NBPKVHIA8YN77AF",
      "name": "Q9T4KP",
      "id": 0
    },
    {
      "content": "SLYAGF7T59WXGSVWEFO0",
      "name": "WXWSZK",
      "id": 1
    },
    {
      "content": "LL7FQN25HXPFFH79YRHJ",
      "name": "88UVH3",
      "id": 2
    },
    {
      "content": "0AUVHFJFQXXV9UTZ5NTQ",
      "name": "NUBCWG",
      "id": 3
    },
    {
      "content": "SK7FHZZSPKYQBQUGRRZE",
      "name": "6NCVFA",
      "id": 4
    }
  ]
}
```

Splunk Search Pipelines

These are the pipelines used in this example:

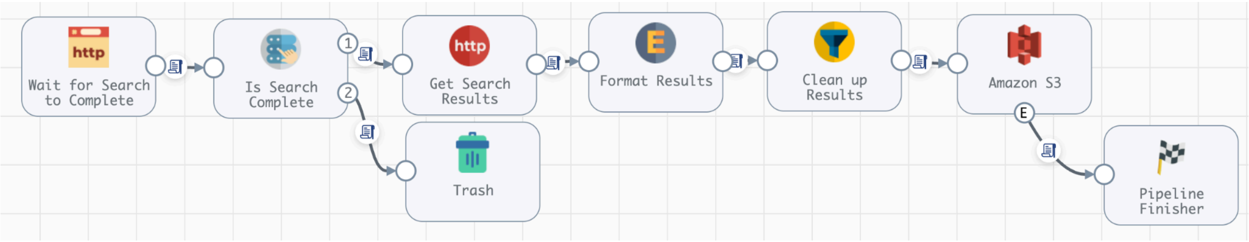
Splunk Search (Part 1)

This pipeline calls the `/splunk_submit_search` mock endpoint, passing in a `search_keyword` and getting back a `sid`. The `sid` is extracted from the response and used to launch the `Splunk Search (Part 2)` pipeline asynchronously, and then exits. Here is the pipeline:



Splunk Search (Part 2)

This pipeline polls the `/splunk_poll_complete` mock endpoint every five seconds for a particular sid. The search's status is extracted from the response once the status is "complete", the pipeline calls the `/splunk_download_report_data` mock endpoint to get the search results. The results are then formatted and cleaned up, and written to a file in S3, partitioned by <search keyword> and <sid>. Here is the pipeline:

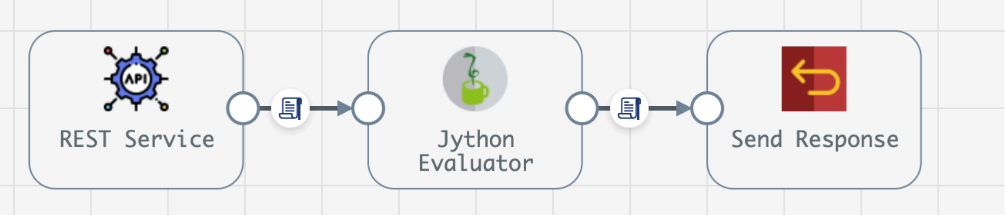


Splunk Mock APIs (Microservice Pipelines)

There are three Splunk Mock API microservice pipelines:

Splunk_Mock_API_Submit_Search

This microservice is at the endpoint `/splunk_submit_search`. It takes a `search_keyword` and returns a random `sid.` Here is the pipeline:



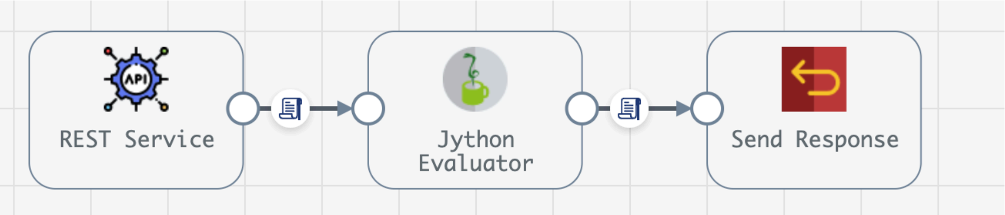
Here is an example `curl` call and response with an XML response format

```
$curl http://portland.onefoursix.com:18630/public-rest/v1/gateway/splunk_submit_search
  -H "Content-Type:application/xml" -H "X-Requested-By:curl" \
  -d '{"search_keyword": "incident_xyz"}'

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<response>
  <httpStatusCode>200</httpStatusCode>
  <data>
    <search_keyword>incident_xyz</search_keyword>
    <response>
      <sid>1413441957.34400_6YZRTWEG-IVFM-FNVJ-WMYR-BPYIBEUTFYTX</sid>
    </response>
  </data>
</response>
```

Splunk_Mock_API_Poll_Complete

This microservice is at the endpoint `/splunk_poll_complete`. It takes a `sid` and the first four times it is called for a given `sid` it returns the status "still working" and the fifth time it is called for that same `sid` it returns the status "complete". Here is the pipeline:



Here is an example `curl` call and response for a new `sid`:

```
$ curl
http://portland.onefoursix.com:18630/public-rest/v1/gateway/splunk_poll_complete \
 -H "Content-Type:application/xml" -H "X-Requested-By:curl" \
 -d '{"sid": "1707494499.314426_A3BD8382-D141-4939-B445-35378A4DA4BD"}'

{"httpStatusCode":200,"data":[{"sid":"1707494499.314426_A3BD8382-D141-4939-B445-35378A
4DA4BD","status":"still working","counter":1}],"error":[]}
```

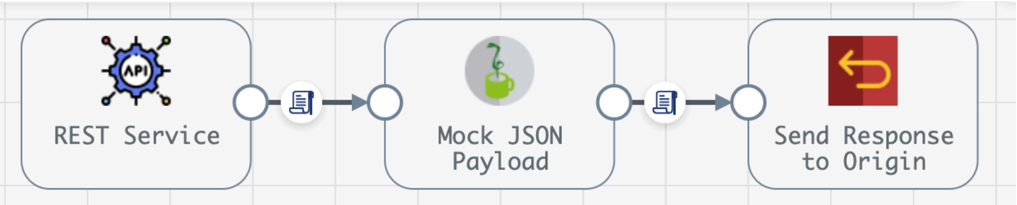
And here is the response on the fifth call for the same `sid`

```
$ curl
http://portland.onefoursix.com:18630/public-rest/v1/gateway/splunk_poll_complete \
  -H "Content-Type:application/xml" -H "X-Requested-By:curl" \
  -d '{"sid": "1707494499.314426_A3BD8382-D141-4939-B445-35378A4DA4BD"}'

{"httpStatusCode":200,"data":[{"sid":"1707494499.314426_A3BD8382-D141-4939-B445-35378A
4DA4BD","status":"complete","counter":0}],"error":[]}
```

Splunk_Mock_API_Download_Report_Data

This microservice is at the endpoint `/splunk_download_report_data`. It takes a `sid` and returns a few random generated strings to serve as mocked search results for the `sid`. Here is the pipeline:



Here is an example `curl` call and response for a `sid`:

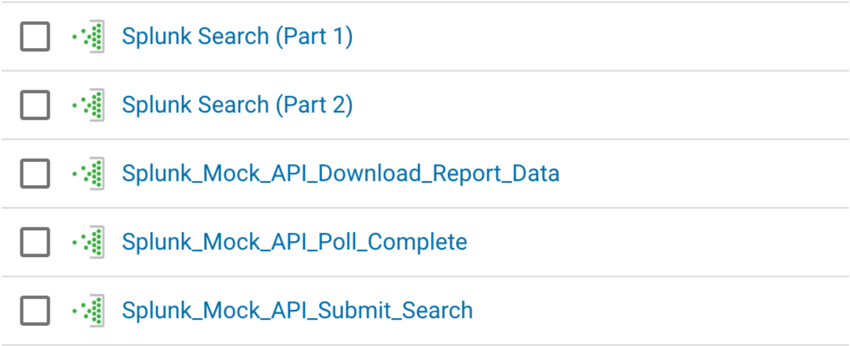
```
$ curl
http://portland.onefoursix.com:18630/public-rest/v1/gateway/splunk_download_report_data  \
 -H "Content-Type:application/xml" -H "X-Requested-By:curl" \
 -d '{"sid": "1707494499.314426_A3BD8382-D141-4939-B445-35378E4DA4BD"}'
```

```
{"httpStatusCode":200,"data":[{"results":[{"content":"OY3EPFM3GIAJXN9LBUBX","na
me":"QHC3WR","id":0},{"content":"JQUSYT3W0FXWHEOUTJ2D","name":"DE1JDE","id":1},
{"content":"QSNNL6QPVWJIOV2YI2DZ","name":"DF6GVK","id":2},{"content":"ZLZNYO2GN
OEUHGEZ5PZK","name":"BFN4DY","id":3},{"content":"ZVBFXPMNKAVU8ISXY6JI","name":"
DYHKWV","id":4}]}],"error":[]}
```

Installing and Configuring the Example

Import the Pipelines

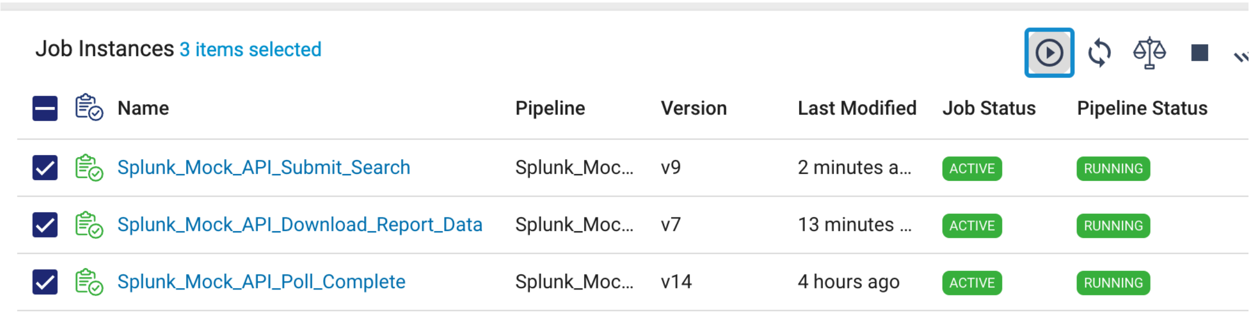
Import the pipelines from the downloaded archive. You should see these five pipelines:



- □ Splunk Search (Part 1)
- □ Splunk Search (Part 2)
- □ Splunk_Mock_API_Download_Report_Data
- □ Splunk_Mock_API_Poll_Complete
- □ Splunk_Mock_API_Submit_Search

Create and Start Job Instances for the Microservices

Create Job Instances for each of the three microservice pipelines, and start the Job Instances. Make sure all three Jobs are green / running:



Job Instances 3 items selected

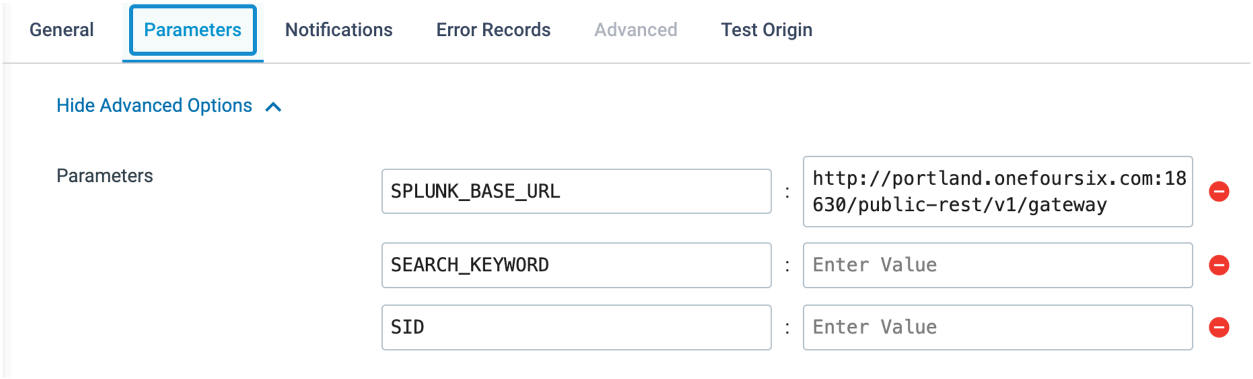
☐	Name	Pipeline	Version	Last Modified	Job Status	Pipeline Status
☑	Splunk_Mock_API_Submit_Search	Splunk_Moc...	v9	2 minutes a...	ACTIVE	RUNNING
☑	Splunk_Mock_API_Download_Report_Data	Splunk_Moc...	v7	13 minutes ...	ACTIVE	RUNNING
☑	Splunk_Mock_API_Poll_Complete	Splunk_Moc...	v14	4 hours ago	ACTIVE	RUNNING

Test each microservice using a curl command as shown in the sections above.

Edit the Splunk Search (Part 2) pipeline

Edit the `Splunk Search (Part 2)` pipeline:

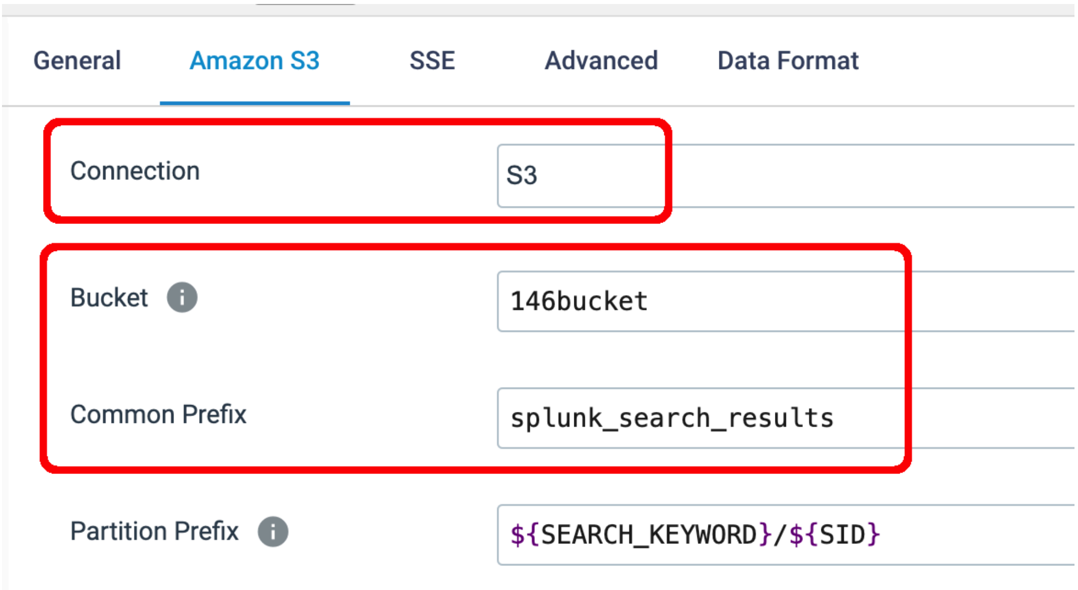
- In the pipeline parameters, set the base URL for your SDC's gateway. You can leave the other two parameters blank



Parameters

SPLUNK_BASE_URL	:	http://portland.onefoursix.com:18630/public-rest/v1/gateway
SEARCH_KEYWORD	:	Enter Value
SID	:	Enter Value

- In the S3 Destination, set your connection, bucket and top level directory for where the search results will be saved:

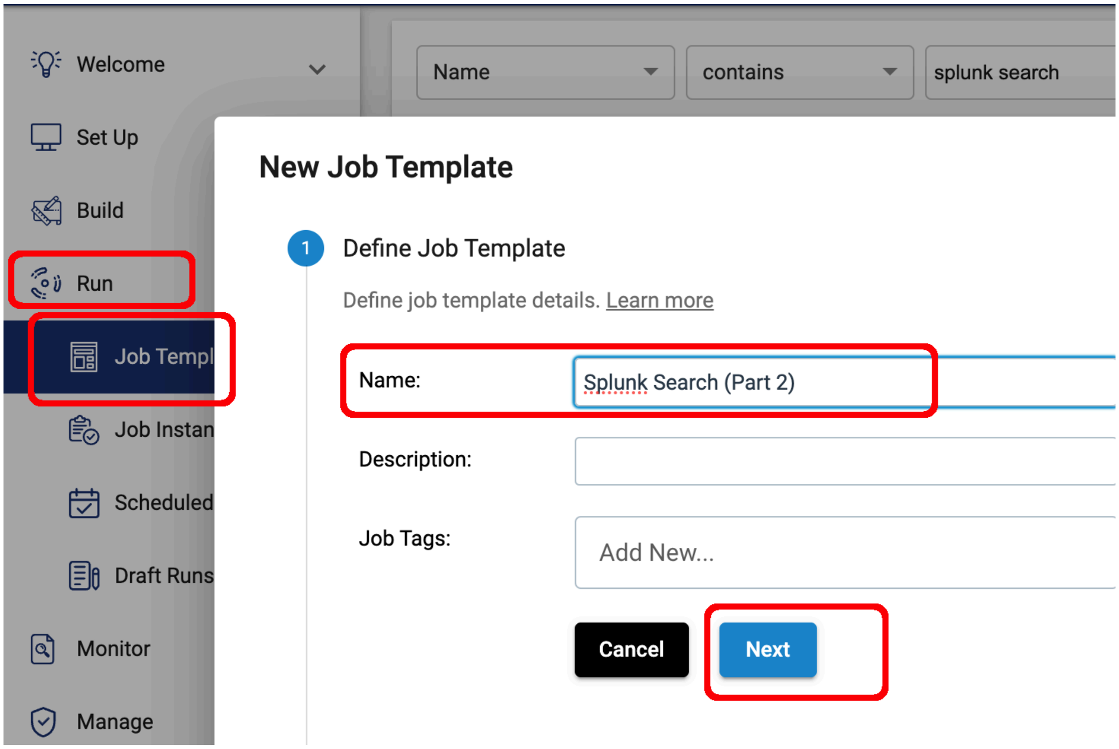


Connection	S3
Bucket	146bucket
Common Prefix	splunk_search_results
Partition Prefix	${SEARCH_KEYWORD}/${SID}

Save the updated pipeline version

Create a Job Template for the Splunk Search (Part 2) pipeline

Create a Job Template for the Splunk Search (Part 2) pipeline:



New Job Template

1 Define Job Template

Define job template details. Learn more

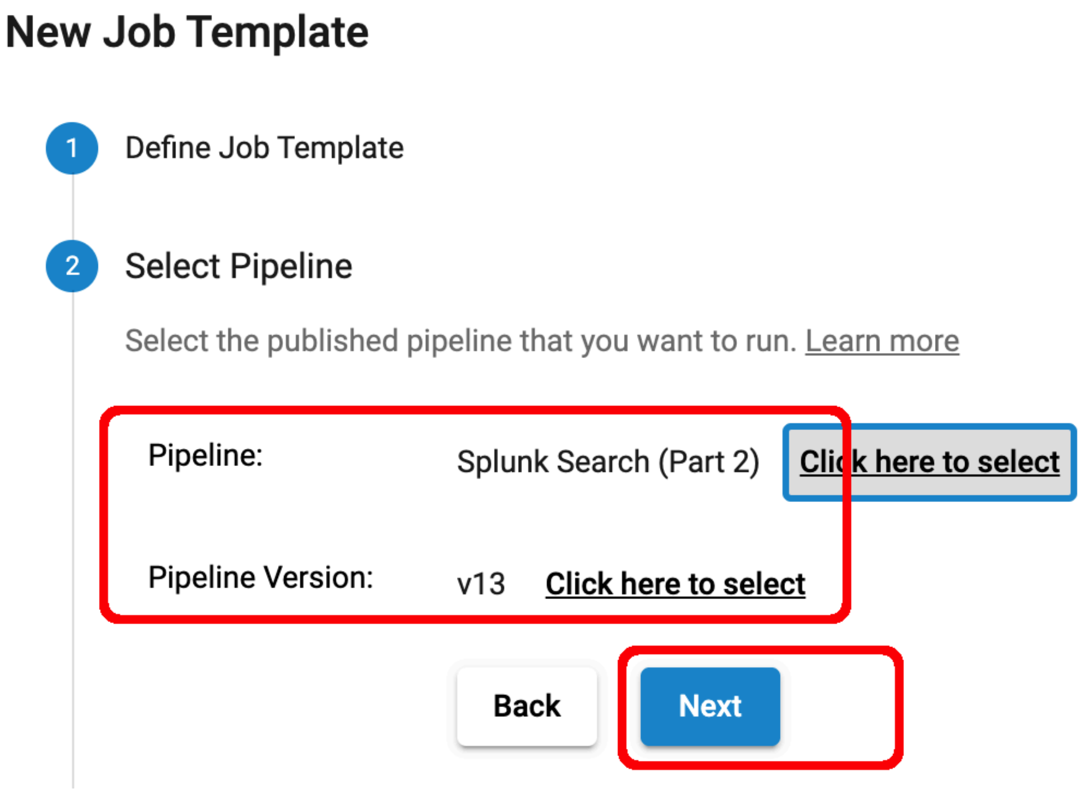
Name: Splunk Search (Part 2)

Description:

Job Tags: Add New...

Give it a name and click next

- Select the latest version of the Splunk Search (Part 2) pipeline:



New Job Template

1 Define Job Template

2 Select Pipeline

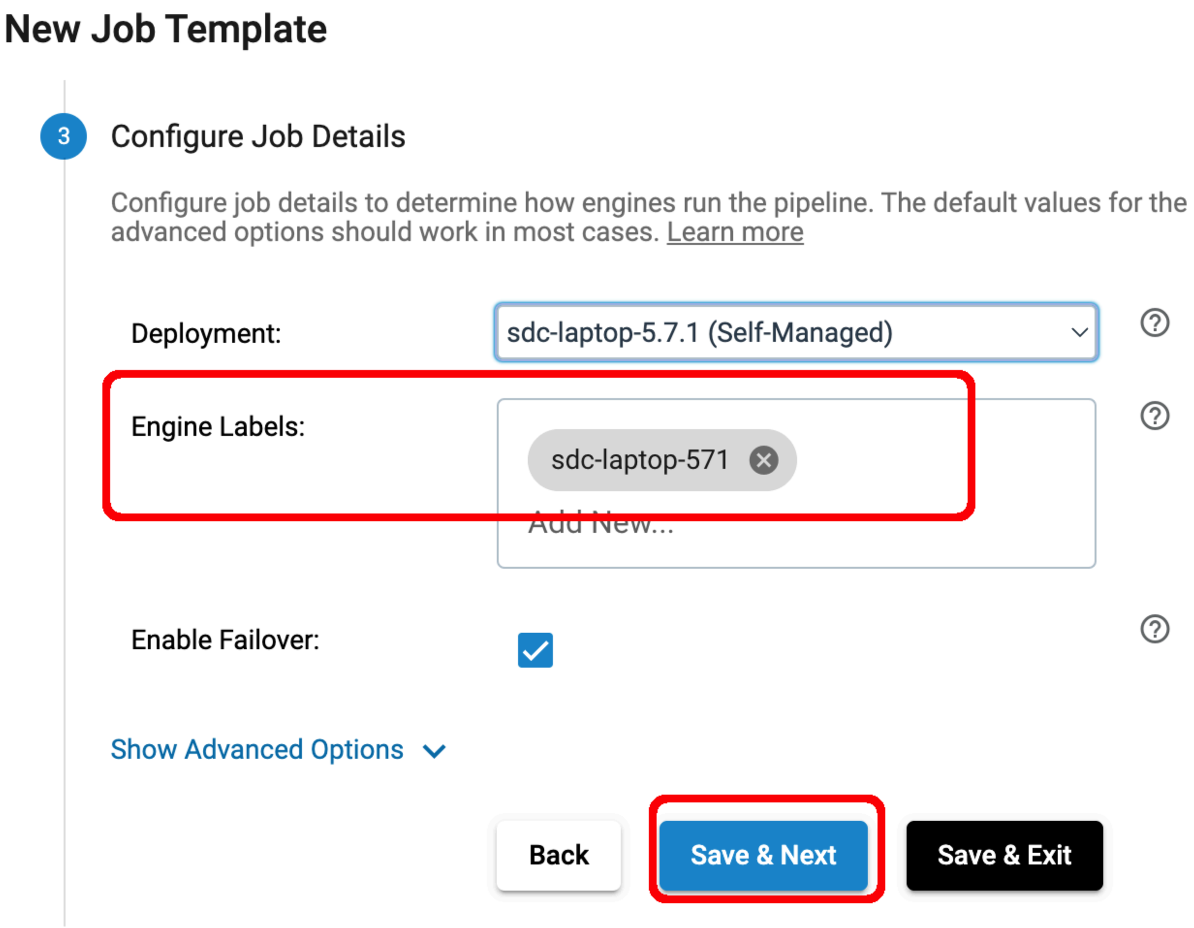
Select the published pipeline that you want to run. Learn more

Pipeline: Splunk Search (Part 2)

Pipeline Version: v13

Click Next

- Set the engine label:



New Job Template

3 Configure Job Details

Configure job details to determine how engines run the pipeline. The default values for the advanced options should work in most cases. Learn more

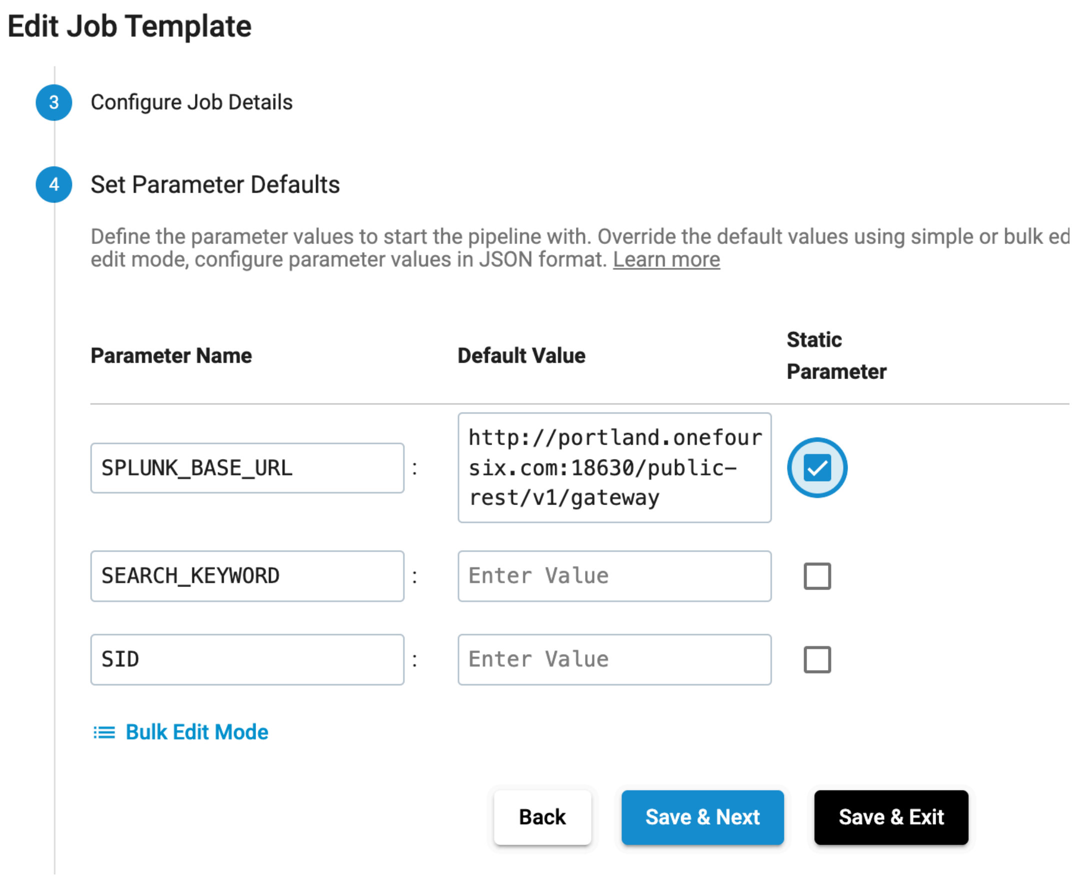
Deployment: sdc-laptop-5.7.1 (Self-Managed)

Engine Labels: sdc-laptop-571

Enable Failover: ☑

Click Save & Next

- Confirm the value for the SPLUNK_BASE_URL parameter, and set it as static:



Edit Job Template

3 Configure Job Details

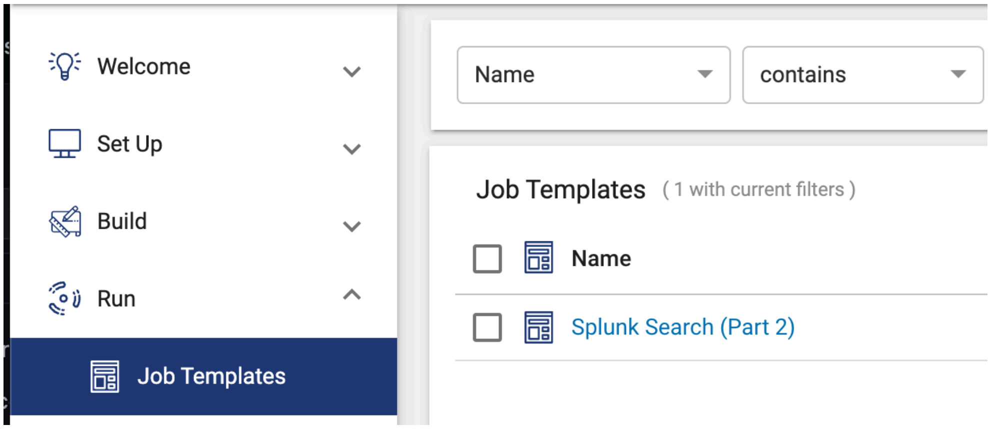
4 Set Parameter Defaults

Define the parameter values to start the pipeline with. Override the default values using simple or bulk ed edit mode, configure parameter values in JSON format. Learn more

Parameter Name		Default Value	Static Parameter
SPLUNK_BASE_URL	:	http://portland.onefoursix.com:18630/public-rest/v1/gateway	☑
SEARCH_KEYWORD	:	Enter Value	□
SID	:	Enter Value	□

Click Save and Next.

Finally, Click Exit and you should have a Job Template for the pipeline Splunk Search (Part 2)

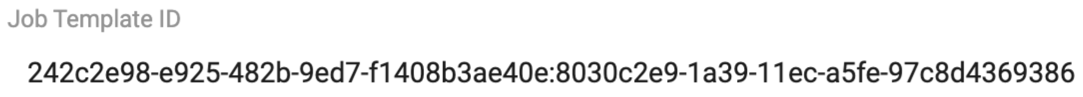


	Name
□	Splunk Search (Part 2)

Save the Job ID for the Splunk Search (Part 2) Job Template

Click on the Splunk Search (Part 2) Job Template and expand the `Show Additional Info` widget:

Copy the Job Template ID and paste it into a safe place (we'll need that value when we configure the Job Template for Splunk Search (Part 1)



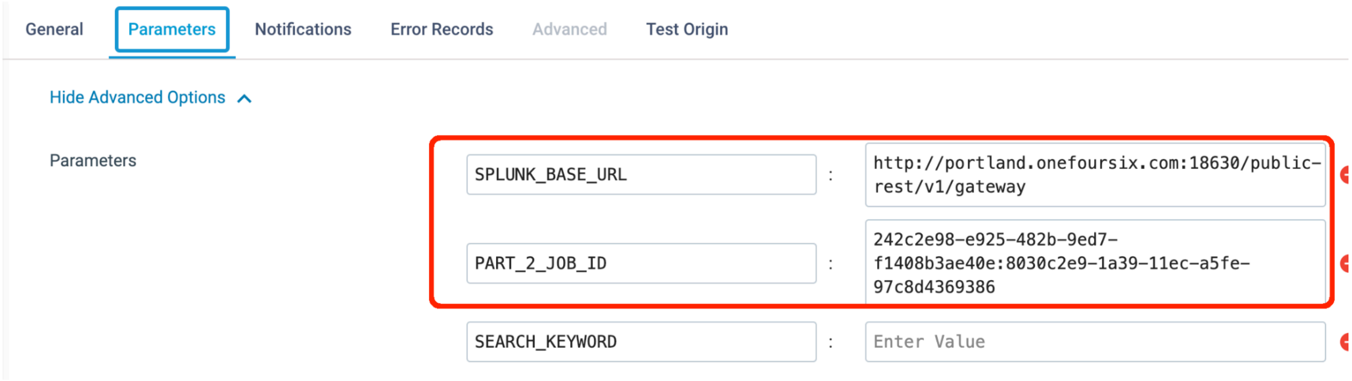
Job Template ID

242c2e98-e925-482b-9ed7-f1408b3ae40e:8030c2e9-1a39-11ec-a5fe-97c8d4369386

Edit the Splunk Search (Part 1) pipeline

Edit the `Splunk Search (Part 1)` pipeline:

- Set these two parameters. Use the Job Template ID for Splunk Search (Part 2) for the `PART_2_JOB_ID` parameter. You can leave the `SEARCH_KEYWORD` parameter blank



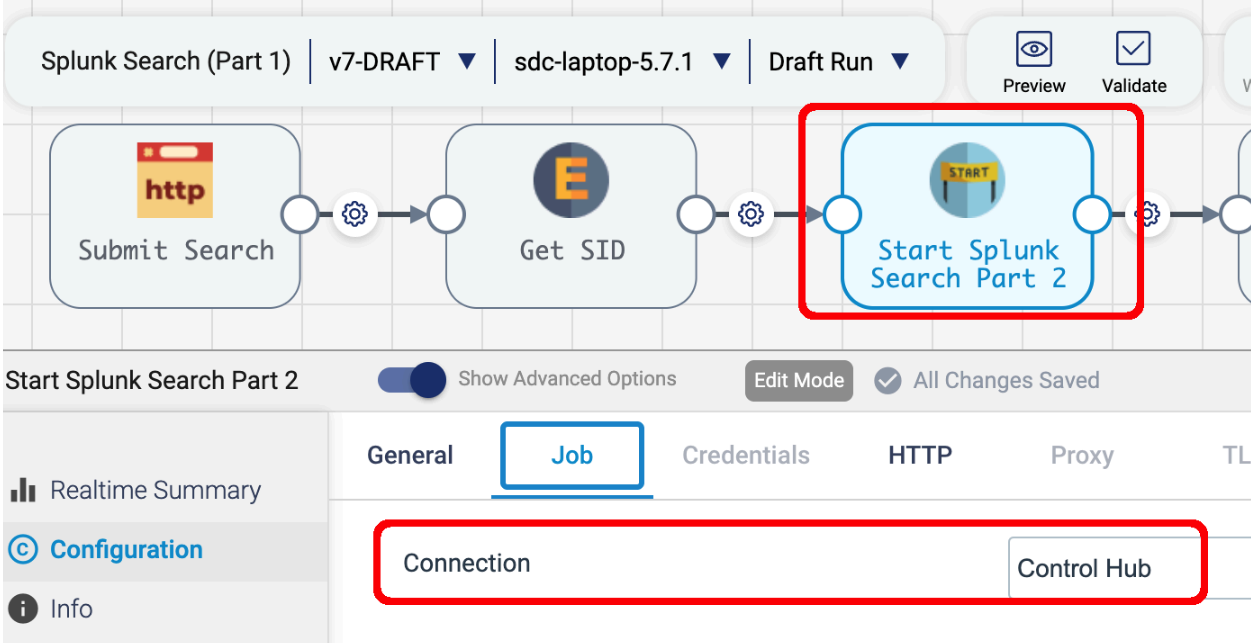
Parameters

SPLUNK_BASE_URL : http://portland.onefoursix.com:18630/public-rest/v1/gateway

PART_2_JOB_ID : 242c2e98-e925-482b-9ed7-f1408b3ae40e:8030c2e9-1a39-11ec-a5fe-97c8d4369386

SEARCH_KEYWORD : Enter Value

Set a Control Hub Connection in the `Start Splunk Search Part 2` stage



Start Splunk Search Part 2

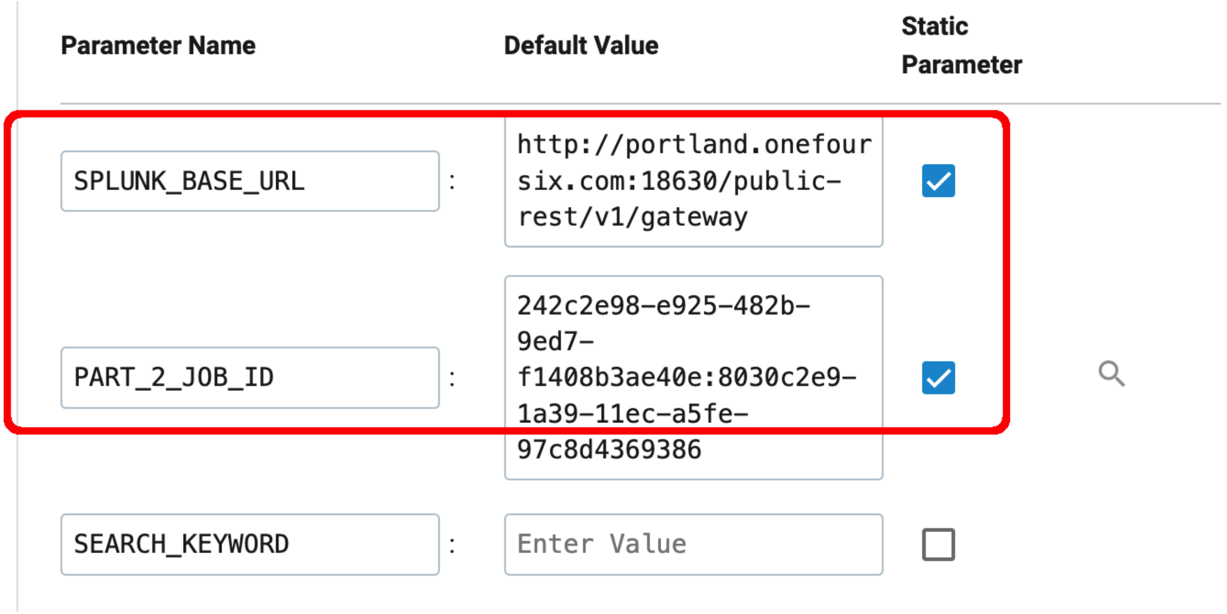
Connection: Control Hub

Save the updated version of the pipeline.

Create a Job Template for the Splunk Search (Part 1) pipeline

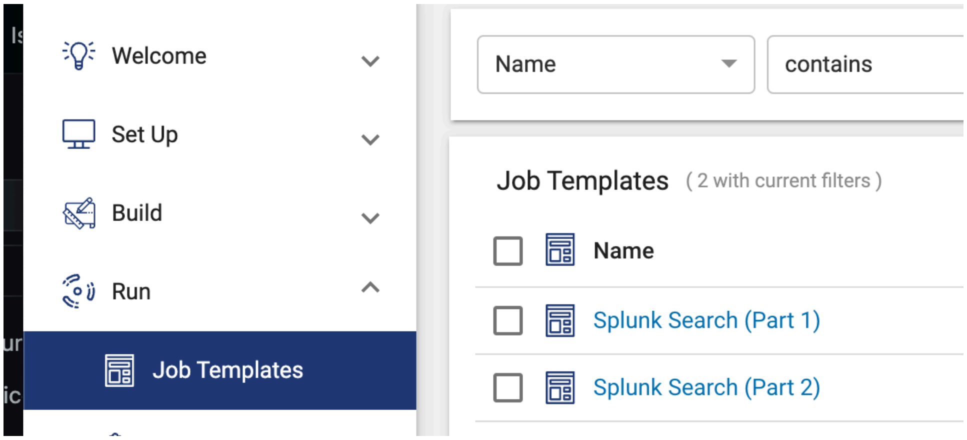
Repeat the previous step to create a Job Template for the `Splunk Search (Part 1)` pipeline.

Set values for these two parameters, and mark them as static.



Parameter Name		Default Value	Static Parameter
SPLUNK_BASE_URL	:	http://portland.onefoursix.com:18630/public-rest/v1/gateway	☑
PART_2_JOB_ID	:	242c2e98-e925-482b-9ed7-f1408b3ae40e:8030c2e9-1a39-11ec-a5fe-97c8d4369386	☑
SEARCH_KEYWORD	:	Enter Value	□

You should now have two Job Templates:



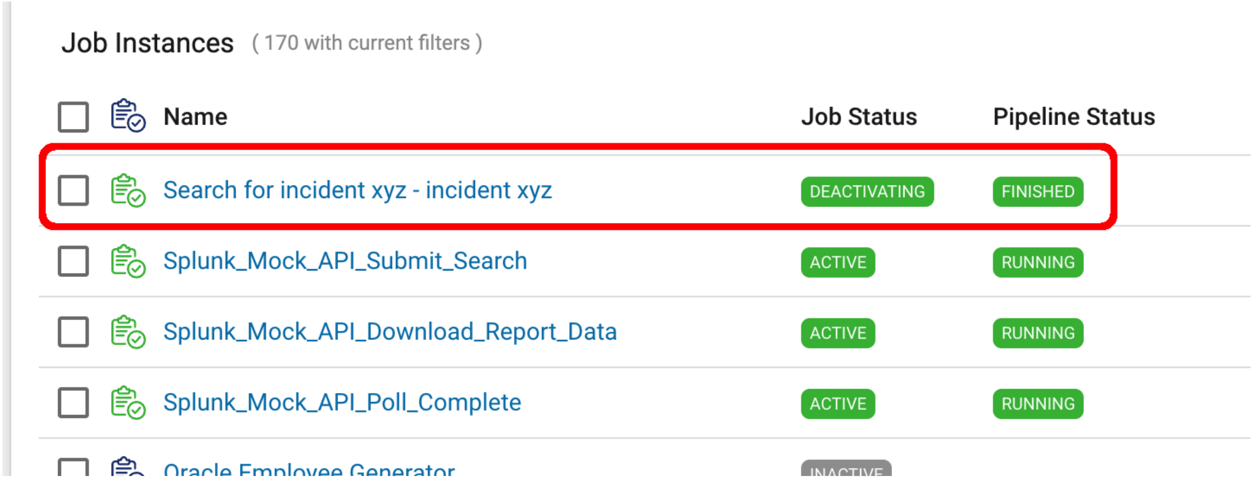
Job Templates (2 with current filters)

□	Name
□	Splunk Search (Part 1)
□	Splunk Search (Part 2)

→ If all goes well you should now be able to run the example as described in the high level walkthrough section above.

Monitoring

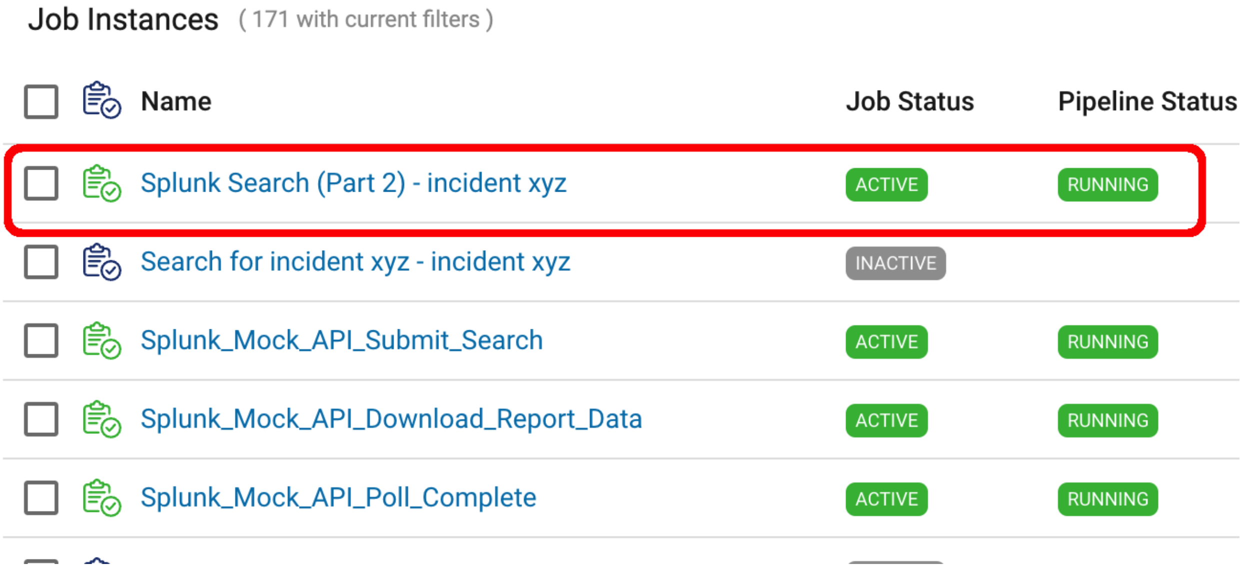
When you first start and run a Job Template Instance for `Splunk Search (Part 1)`, if you immediately switch over the the Job Instance list, you can see the Splunk Search (Part 1) instance started up and then quickly shuts down, as it got the `sid` and launched Splunk Search (Part 2):



Job Instances (170 with current filters)

☐	Name	Job Status	Pipeline Status
☐	Search for incident xyz - incident xyz	DEACTIVATING	FINISHED
☐	Splunk_Mock_API_Submit_Search	ACTIVE	RUNNING
☐	Splunk_Mock_API_Download_Report_Data	ACTIVE	RUNNING
☐	Splunk_Mock_API_Poll_Complete	ACTIVE	RUNNING
☐	Oracle Employee Generator	INACTIVE	

In a moment, the instance for Splunk Search (Part 2) will start up, and will stick around for about 30 seconds before it too shuts down:



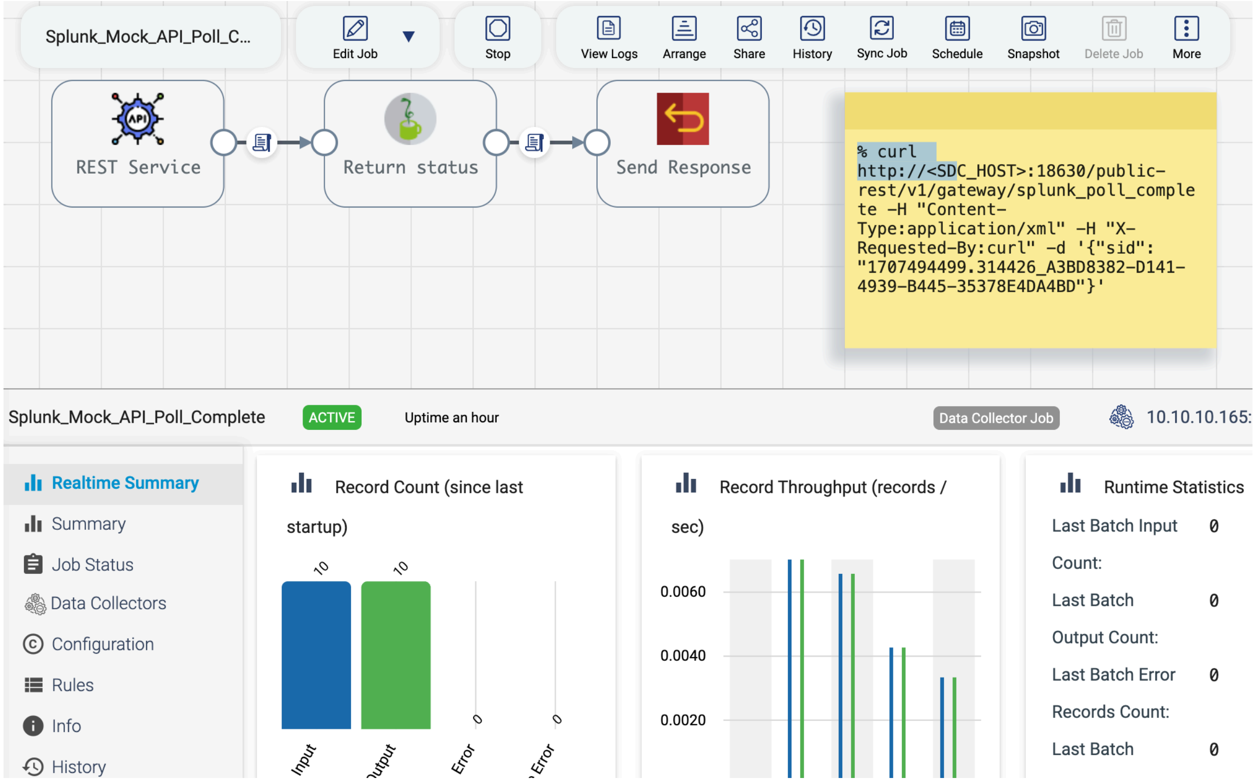
Job Instances (171 with current filters)

☐	Name	Job Status	Pipeline Status
☐	Splunk Search (Part 2) - incident xyz	ACTIVE	RUNNING
☐	Search for incident xyz - incident xyz	INACTIVE	
☐	Splunk_Mock_API_Submit_Search	ACTIVE	RUNNING
☐	Splunk_Mock_API_Download_Report_Data	ACTIVE	RUNNING
☐	Splunk_Mock_API_Poll_Complete	ACTIVE	RUNNING

At this point you should see the search results in your S3 Bucket.

You can also observe the microservices all received and processed calls during the run.

For example, after two runs, I see 10 calls made to the polling microservice:



```
% curl
http://<SDC_HOST>:18630/public-
rest/v1/gateway/splunk_poll_comple
te -H "Content-
Type:application/xml" -H "X-
Requested-By:curl" -d '{"sid":
"1707494499.314426_A3BD8382-D141-
4939-B445-35378E4DA4BD"}'
```

Last Batch Input Count: 0
Last Batch Output Count: 0
Last Batch Error Records Count: 0
Last Batch 0