IBM **TechXchange**

1783 – Day 2 Operations for IBM StreamSets

Hands-on lab guide

Mark Brooks – Solution Architect, IBM StreamSets
mark.brooks@ibm.com

Deepak Rangarao – Distinguished Engineer & WW CTO Technical Sales - Cloud Paks Global Sales (WW)
drangar@us.ibm.com

Daniel Hancock – Principal WW Data & AI Technical Specialist - Global Sales (WW)
daniel.hancock@us.ibm.com

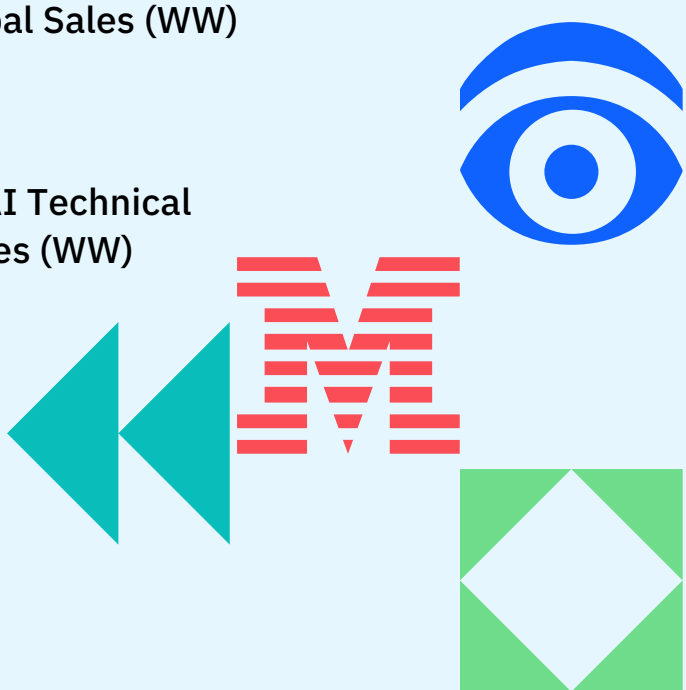


Table of Contents

Notices and disclaimers

U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

This document is current as of the initial date of publication and may be changed by IBM at any time. Not all offerings are available in every country in which IBM operates.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information.

This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity. IBM products and services are warranted per the terms and conditions of the agreements under which they are provided. The performance data and client

examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

IBM products are manufactured from new parts or new and used parts.
In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in a controlled,
isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

Notices and disclaimers (Continued)

It is the customer's responsibility to ensure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**

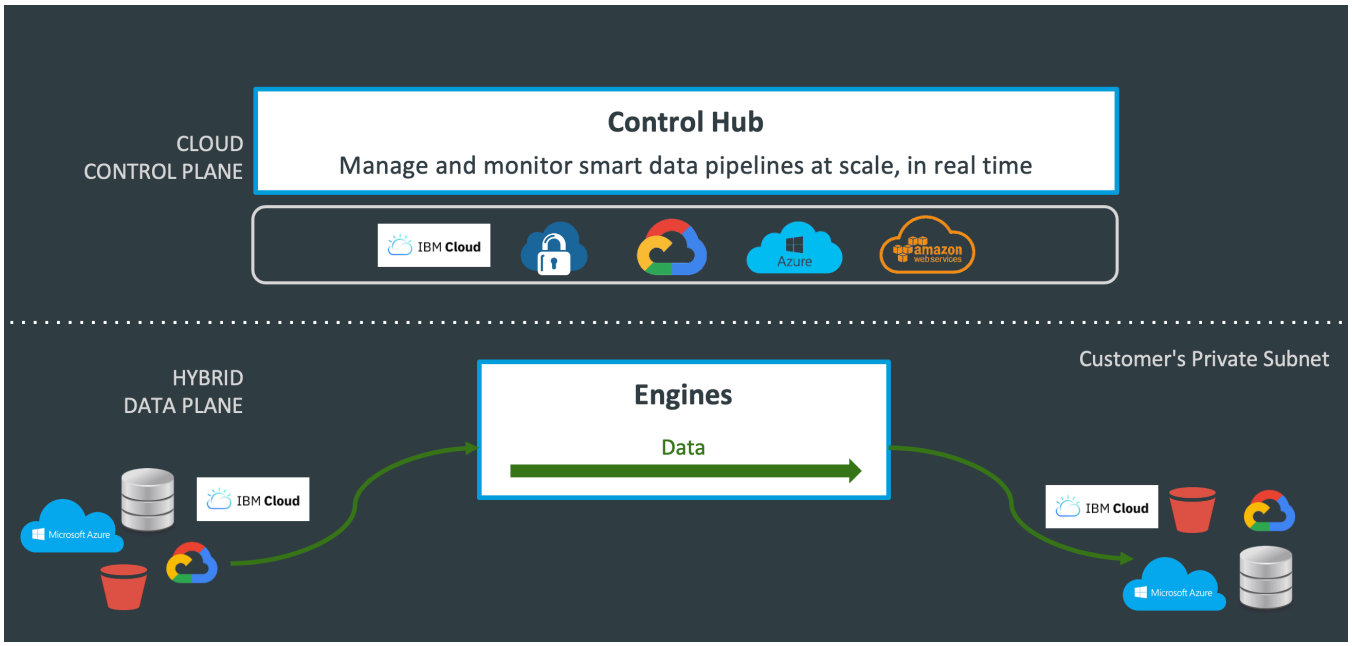
The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, and ibm.com are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at

Learn more →

1 Introduction

IBM StreamSets is a powerful and flexible data integration platform designed to simplify the process of building, monitoring, and managing near real-time data pipelines. In today's fast-paced, data-centric world, organizations need to move and process vast amounts of streaming data across diverse environments. IBM StreamSets provides the tools to do this efficiently and reliably, allowing users to handle complex data flows with ease.



Key Features of IBM StreamSets include:

1. **No-Code Data Integration:**
    - IBM StreamSets allows data integrations to be implemented using a rich graphical environment, with drag and drop connectors and processors; no coding is required.
2. **Schema Flexibility:**
    - IBM StreamSets is designed to handle dynamic and evolving data schemas, making it ideal for environments where data structures are constantly changing. It automatically adapts to schema changes, ensuring that data pipelines remain operational even when data formats shift.
3. **Modular and Reusable Pipelines:**
    - IBM StreamSets promotes modularity through reusable pipeline fragments, parameterization, and Job templates, allowing users to create and maintain complex pipelines more efficiently. This approach reduces redundancy and simplifies pipeline management.

4. **Custom Data Transformations:**
    - With support for Groovy, Jython, and other scripting languages, IBM StreamSets allows users to embed custom logic within their pipelines. This flexibility ensures that specific business requirements can be met with the flexibility of custom scripting/coding.
5. **Real-Time Monitoring and Debugging:**
    - IBM StreamSets provides robust monitoring and debugging tools, including comprehensive runtime metrics, snapshots, and previews, which help users identify and resolve issues quickly. This real-time visibility into pipeline performance ensures that data flows are optimized and reliable.
6. **Broad Connectivity:**
    - IBM StreamSets supports a wide range of connectors for popular data sources and targets, including databases, cloud storage, messaging systems, and more. This extensive connectivity makes it easy to integrate data across different platforms and technologies snologies.

1.1 About this hands-on lab

Welcome to this hands-on lab session, designed for architects, administrators, and data engineers who are looking to deepen their understanding of Day 2 operations with IBM StreamSets. In today's session, we'll focus on the best practices and advanced techniques necessary to efficiently manage, maintain, and scale your IBM StreamSets pipelines in a production environment:

1.1.1 Why Day 2 Operations?

Day 2 operations are crucial in ensuring that your data pipelines remain robust, resilient, and adaptable as they move from development into production. It's not just about building pipelines; it's about designing them for reuse, operationalizing them effectively, and ensuring they can handle the demands of a dynamic, real-world environment.

1.1.2 What We Will Cover

1. **Designing Pipelines for Reuse**
    - **Connections:** Learn how to manage and reuse connections to different data sources and destinations, reducing redundancy and simplifying maintenance.
    - **Fragments:** Explore how pipeline fragments can help you modularize and reuse pipeline components across multiple pipelines, enhancing consistency and reducing development time.
    - **Parameterization:** Discover how to parameterize pipelines to make them more flexible and adaptable to different environments and use cases.

2. **Operationalizing Pipelines**
    - **Jobs:** Understand how to run and manage pipeline jobs, and how they play a critical role in operationalizing your data pipelines.
    - **Job Templates:** Dive into job templates to streamline job creation and ensure consistency across your deployments.
    - **Monitoring:** Learn about the tools and techniques for monitoring your pipelines to ensure they are running smoothly and efficiently.
    - **HA/DR (High Availability/Disaster Recovery):** Get insights into how to configure your IBM StreamSets environment for high availability and disaster recovery to ensure business continuity.
3. **Workflow and Orchestration**
    - Understand how to orchestrate complex workflows that involve multiple pipelines and external processes, ensuring that your data processing tasks are executed in the correct sequence and with proper coordination.
4. **Automation Using IBM StreamSets Python SDK**
    - Learn how to leverage the IBM StreamSets Python SDK to automate various aspects of pipeline management, from creation and deployment to monitoring and maintenance, enabling you to scale your operations efficiently.

1.1.3 Objectives

By the end of this session, you will have hands-on experience with advanced IBM StreamSets features that are essential for Day 2 operations. You'll be equipped with the knowledge to design pipelines that are not only effective but also easy to manage and scale. Additionally, you'll gain practical insights into how to automate and orchestrate your data workflows to maximize efficiency and reliability.

Let's get started and dive into the world of IBM StreamSets Day 2 operations!

2 Hands on lab setup

2.1 Create user account with IBM StreamSets

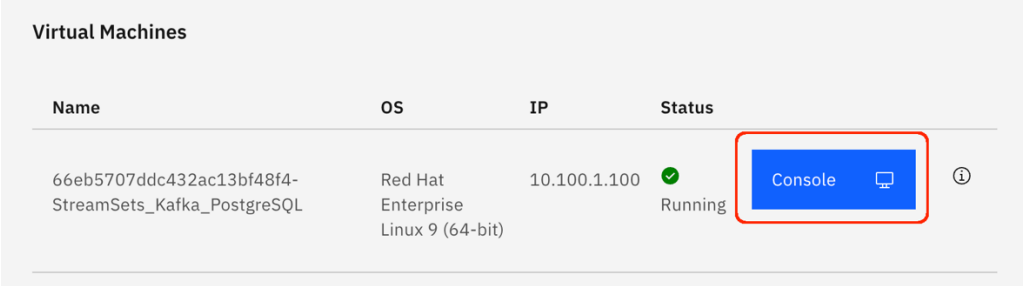
Provide the lab lead with your email ID and they will invite you to the IBM StreamSets SaaS platform. You will need to check your email (and perhaps your spam folder) for the invite to the IBM StreamSets control plane and accept the invitation to join the IBM StreamSets organization and to set your StreamSets password.

2.2 Register data plane with control plane

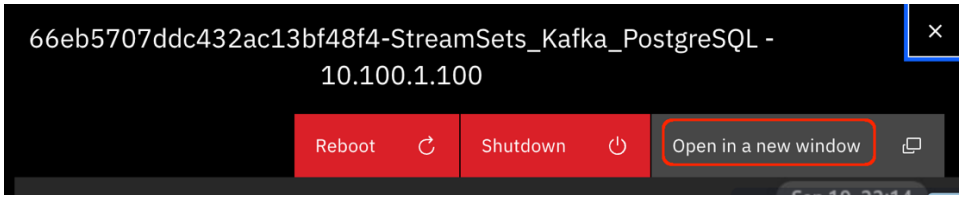
You will be provided with access to a VM; for the purpose of this hands on lab we will use the VM as the host for running the IBM StreamSets “data plane” (aka Engine). The steps below show to install a StreamSets engine and to register the data plane with the control plane.

2.3 Clone a StreamSets Deployment

- Click the Console button in your TechZone Reservation page:

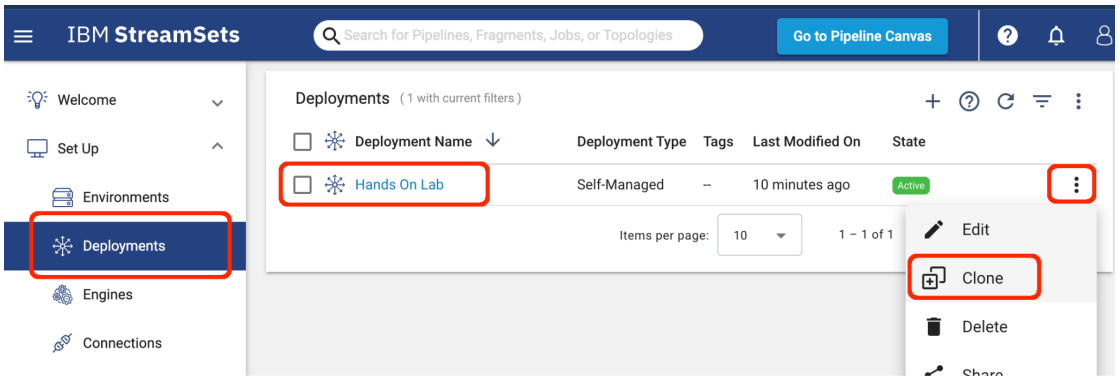


- In the page that opens, click the Open in a new window button:



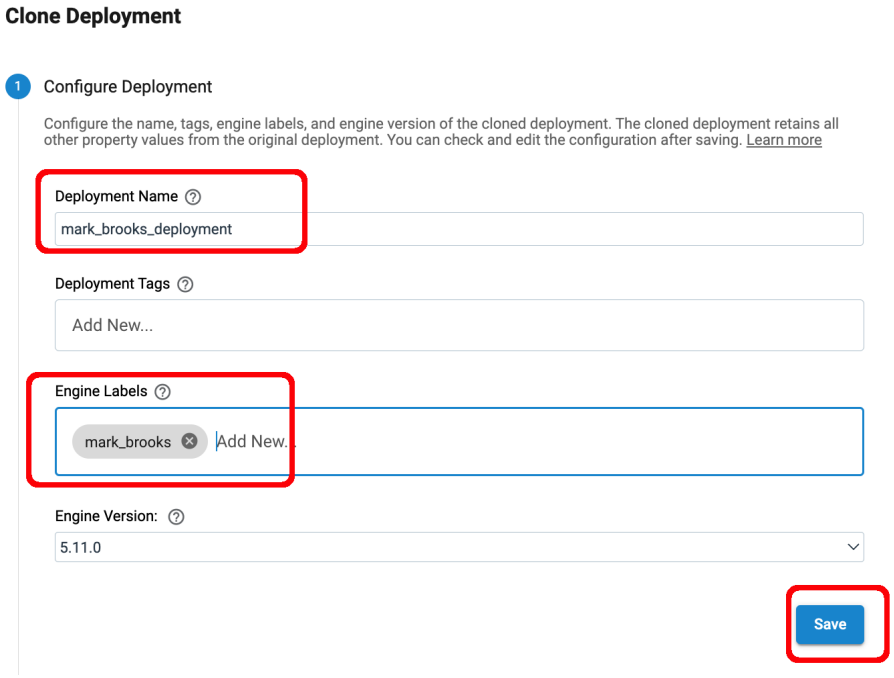
- Log in as the `admin` using the password `IBMDem0s`
- Open a browser window inside the VM by clicking on the “Activities” button on the top left and then on the Firefox icon on the button bar:

- Using Firefox, open the IBM StreamSets control plane login page at https://cloud.login.streamsets.com/login . Login with your credentials (your email address and the password that you received as part of the registration step above).
- Navigate to `Setup > Deployments` and clone the `Hands On Lab` Deployment:



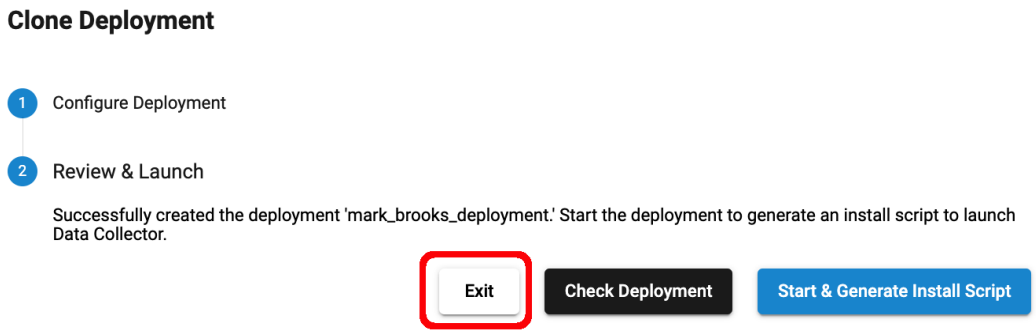
- Edit the Deployment Name using the naming convention `<first_name>_<last_name>_deployment`. For example, I'll name mine `mark_brooks_deployment`.
- Replace the Engine Label with a label of the form `<first_name>_<last_name>`, in my case `mark_brooks`
- Leave the Engine Version as 5.11.0

  For example, here are how things look in my environment:



→ Click `Save`

The next dialog looks like this:

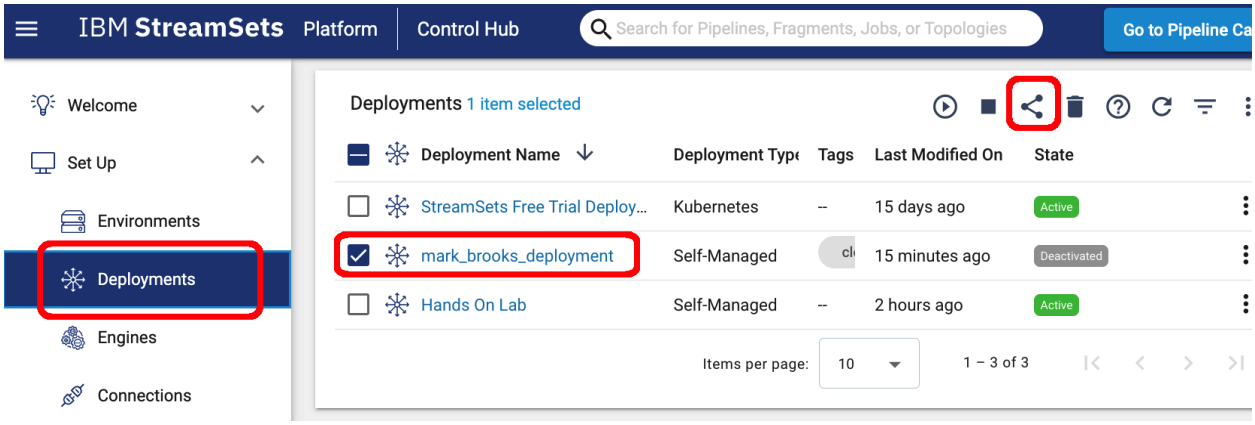


→ Click `Exit`

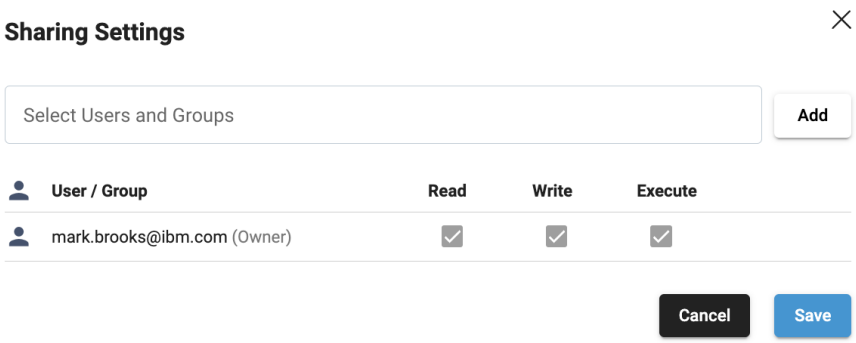
2.4 Set Permissions on your StreamSets Deployment

Important Note: Make sure to complete this step to ensure that no one else will be able to run Jobs on your engine!

- Select the deployment you just created and select the `Sharing` icon:



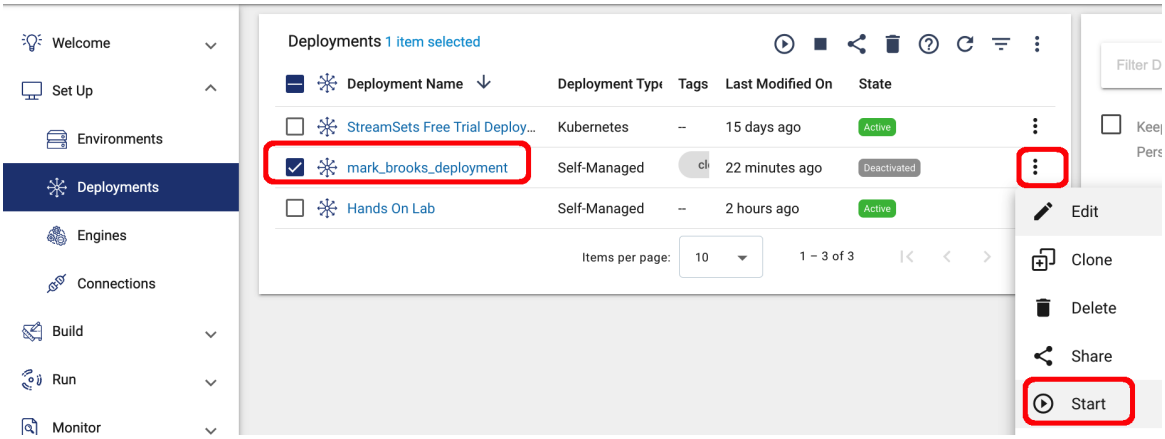
- Edit the ACLs so that only you have read, write, and execute permissions (i.e. delete all entries except your own). The edited ACLs in my example look like this:



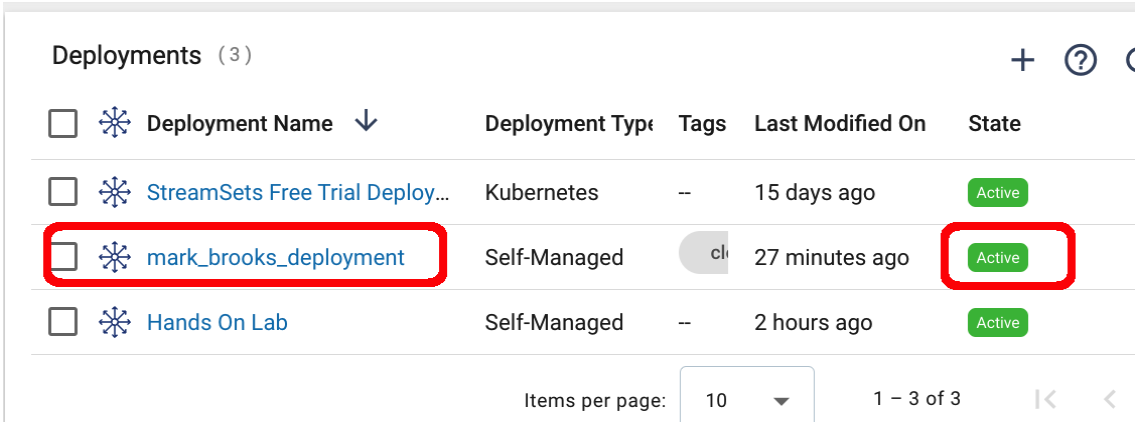
→ Click `Save`.

2.5 Start the Deployment

- Select the just-created Deployment and click the Start icon:

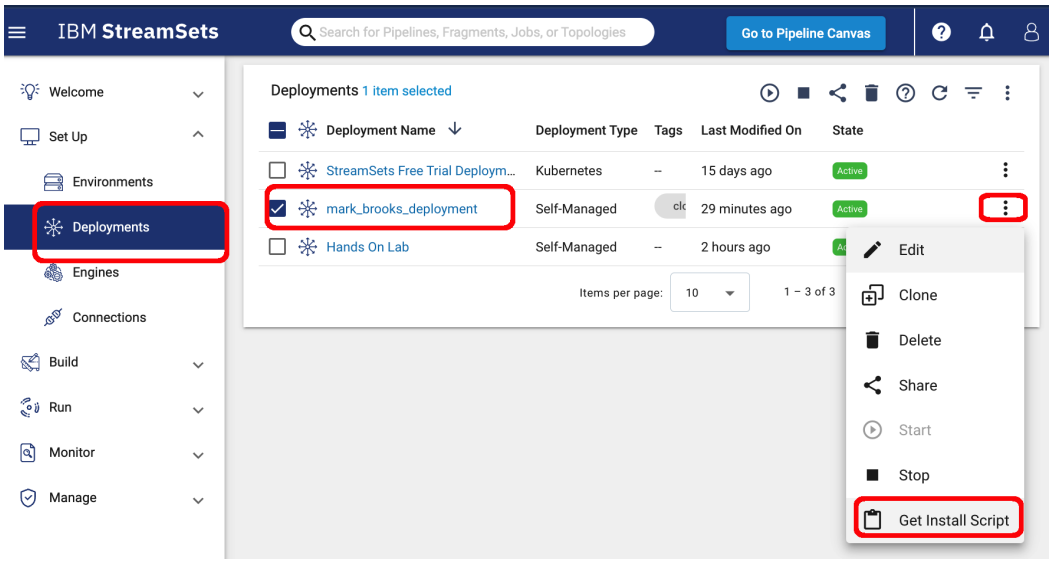


- The Deployment state should transition to Active and turn green:

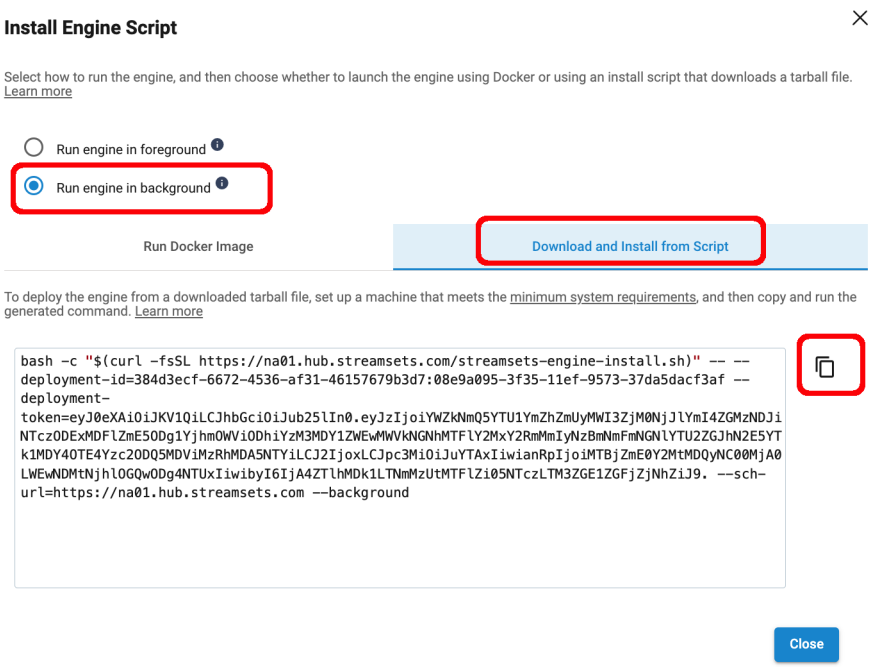


2.6 Generate an Engine Deployment Script

- Select the just-created Deployment and click the `Get Install Script` item:

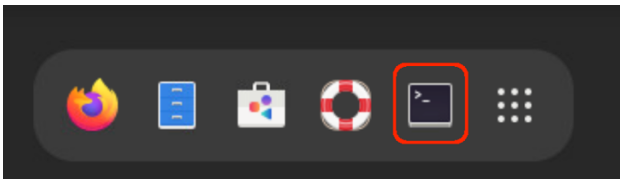


- In the dialog that opens, select `Run engine in background`, `Download and Install from Script`, and then click the `Copy` icon:

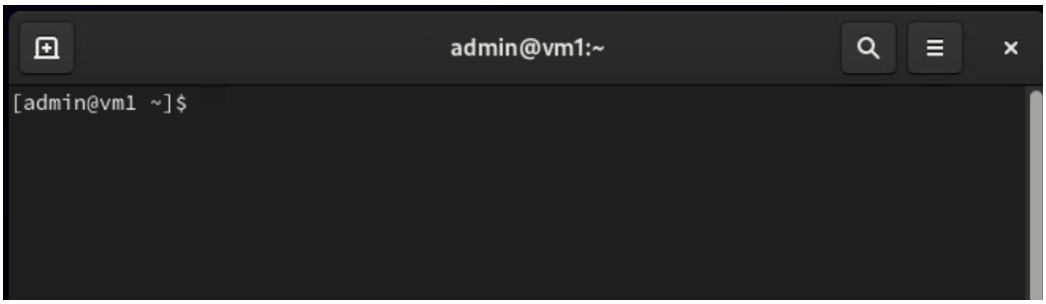


2.7 Run the Engine Deployment Script on the Data Plane VM

- Run the copied install script in a terminal on the data plane VM using the following steps:
    - Click on the terminal session icon within the VM's GUI.



    - You should see a terminal session window open like this:



    - Execute the following command in the terminal session:

```
$ sudo su - admin
```

    - Switch to the `/streamsets` directory

```
$ cd /streamsets
```

Paste in the script copied from the previous step.

When prompted for a download directory, enter the value `sdc-download` and click `enter` and then click `enter` again to confirm.

When prompted for an install directory, enter the value `sdc` and click `enter`, and then click `enter` again to confirm.

Here is an example session with my user input highlighted:

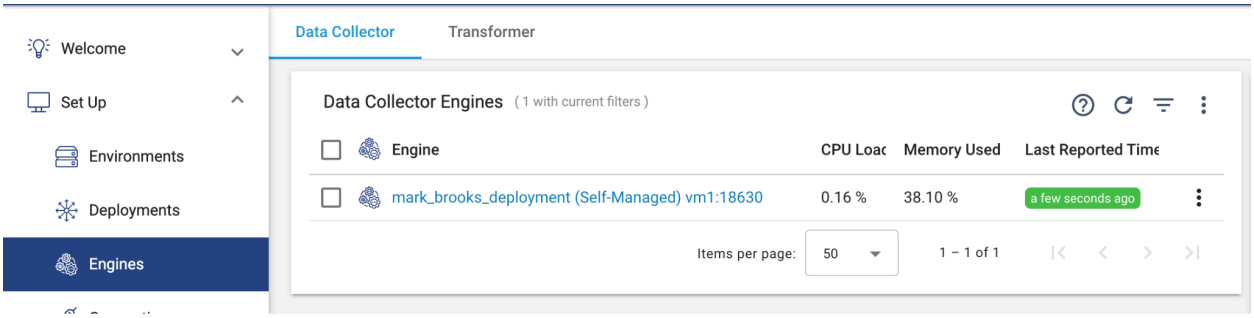
```
[admin@vm1 streamsets]$ bash -c "$(curl -fsSL https://na01.hub.streamsets.com/streamsets-
engine-install.sh)" -- --deployment-id=384d3ecf-6672-4536-af31-46157679b3d7:08e9a095-3f35
-11ef-9573-37da5dacf3af --deployment-token=eyJ0eXAiOiJKV1QiLCJhbGciOiJub25lIn0.eyJzIjoiYW
ZkNmQ5YTU1YmZhZmUyMWI3ZjM0NjJlYmI4ZGMzNDJiNTczODExMDFlZmE5ODg1YjhmOWViODhiYzM3MDY1ZWEwMWV
kNGNhMTFlY2MxY2RmMmIyNzBmNmFmNGNlYTU2ZGJhN2E5YTk1MDY4OTE4Yzc2ODQ5MDViMzRhMDA5NTYiLCJ2Ijox
LCJpc3MiOiJuYTAxIiwianRpIjoiMTBjZmE0Y2MtMDQyNC00MjA0LWEwNDMtNjhlOGQwODg4NTUxIiwibyI6IjA4Z
TlhMDk1LTNmMzUtMTFlZi05NTczLTM3ZGE1ZGFjZjNhZiJ9. --sch-url=https://na01.hub.streamsets.co
m --background
curl --fail -s -S -X GET https://na01.hub.streamsets.com/deployment-tools.sh
INFO - Preparing to download and install engine
Engine process will run in background.
Enter a download directory. (Default: /home/admin/.streamsets/download/dc) sdc-download
Directory sdc-download does not exist. Create It? [Y/N] (Default: Y):
Enter an installation directory. (Default: /home/admin/.streamsets/install/dc) sdc
Directory sdc does not exist. Create It? [Y/N] (Default: Y):
INFO - Starting download of https://archives.streamsets.com/datacollector/5.11.0/tarball/
streamsets-datacollector-core-5.11.0.tgz
Extracting tarball
```

If all goes well, after a couple of minutes, you should see an "`Engine started successfully`" message like this:

```
Running on URI : 'http://vm1:18630'
Successfully connected to the Control Hub Tunneling application and WebSocket session is
active for Control Hub UI to engine communication. Use the Control Hub UI to design pipel
ines and to complete administrative tasks on this engine: https://na01.hub.streamsets.com
/



Engine started successfully.
```

2.8 Confirm the engine is healthy

In the StreamSets Platform UI, make sure the engine has a green state and `Last Reported Time` within the past minute:



→ You have now deployed the data plane engine for IBM StreamSets.

3 Designing pipelines for reuse

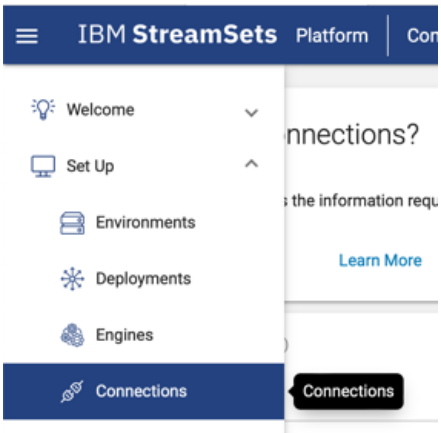
3.1 Reusable connections

[*There are no user exercises for this section, it is only for your information*].

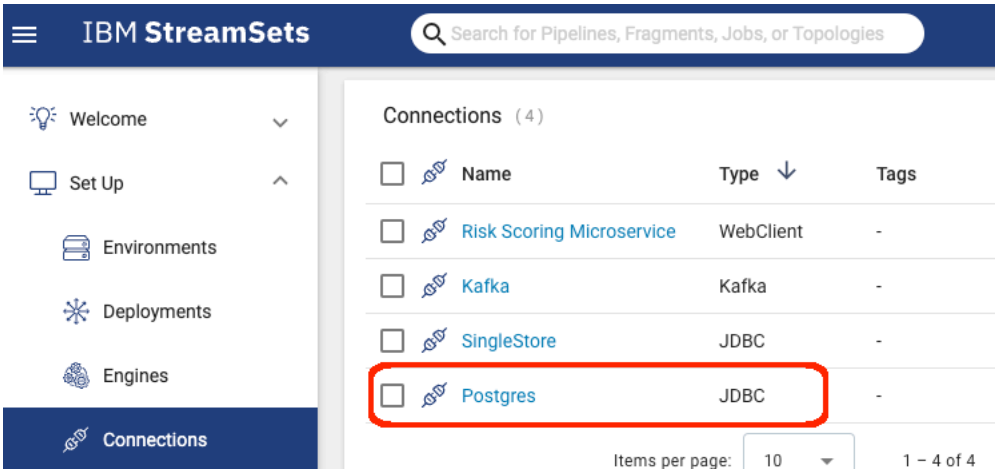
IBM StreamSets enables administrators to define global connections to a variety of data sources and to share the connections with individual users or groups of users. A key aspect of this capability is that users may be granted access to use connections shared with them, but they cannot see the connection details such as URLs and credentials. This provides a high level of security for connection credentials.

Note: This is just a review of the capabilities; you will not be creating connections or sharing them with other attendees.

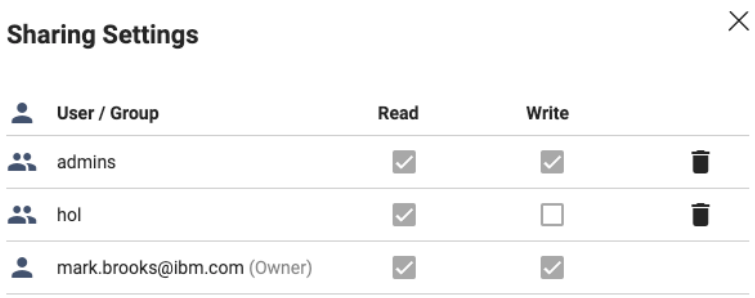
- Navigate to the Setup->Connections screen



- Review the List of Connections. We have created a JDBC connection for the local instance of PostgreSQL running on the data plane/engine VM.



- Here are the access control (sharing) settings for the Postgres connection:



  The permissions shown above allow members of the admins group and the connection's owner to have read/write access to the connection, which lets them view and set all details of the connection including credentials. However, members of the hol (Hands On Lab) group only have read permissions, which means they can use the connection in pipelines, but they are not able to access any of the connection's settings, for example, they can't see the connection's credentials.

- In addition to the security benefits of connections, centralized connections provide a single place to update multiple pipelines. For example, if there are 100 pipelines that read data from an Oracle database, and the Oracle password is changed, all one needs to do is to update the password in the connection shared across all of those pipelines.

3.2 Reusable fragments

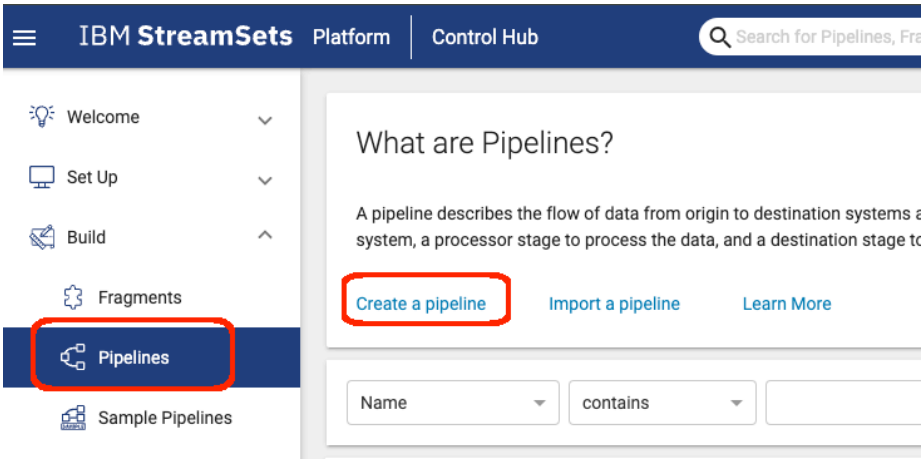
A fragment is a stage or set of connected stages that you can reuse in pipelines. Use fragments to easily add the same processing logic to multiple pipelines and to ensure that the logic is used as designed. You can create fragments from scratch or by selecting a subset of the stages in an existing data pipeline.

In this step you will create a new Pipeline built entirely using Fragments that will perform these steps:

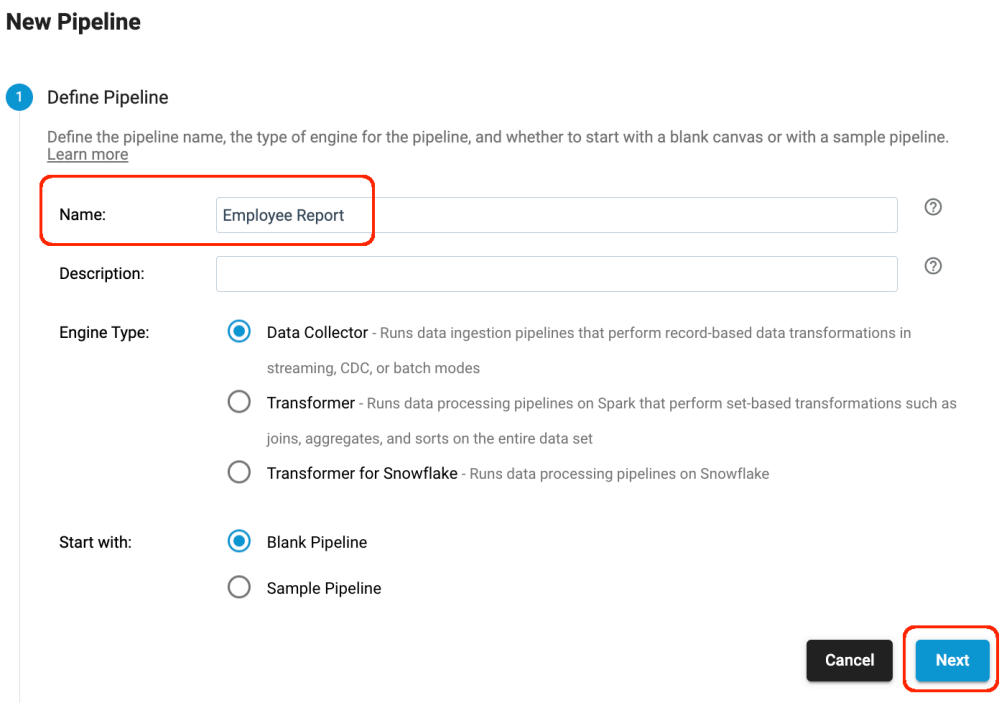
- Generate a stream of Employee data including address information.
- Timestamp each record as it is processed
- Mask any PII information in the records
- Filter employees by a user-specified State value (like "California")
- Write the timestamped, masked, and filtered data to a user-defined directory.
-

Here are the steps to create the pipeline

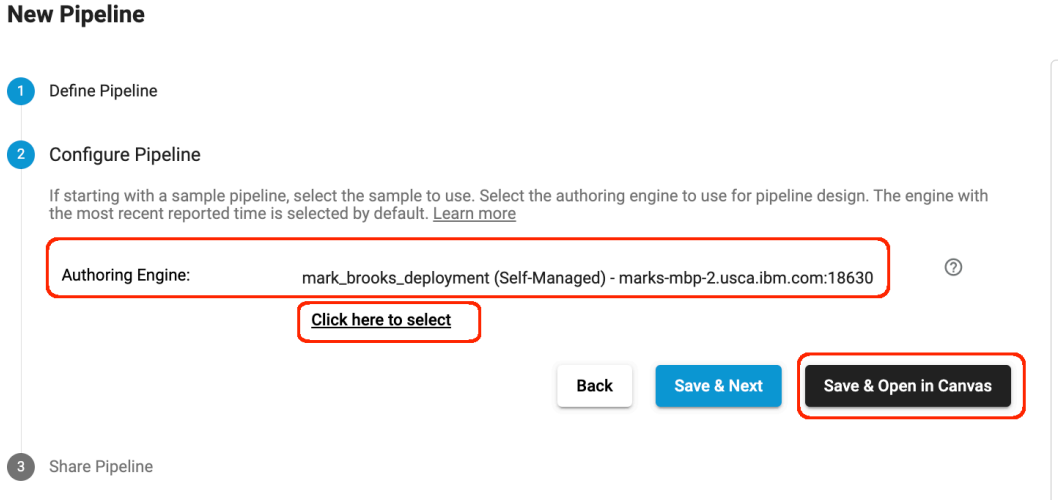
- Create a new Pipeline:



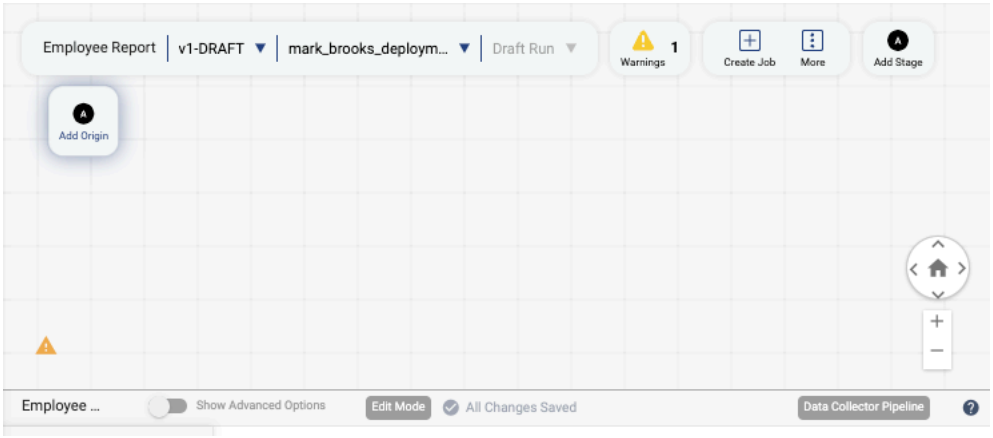
- Enter a name for the pipeline and click the Next button:



- Make sure your engine is selected as the "Authoring Engine"; if not, click the "Click here to select" link and select your engine. Click the Save & Open in Canvas button:

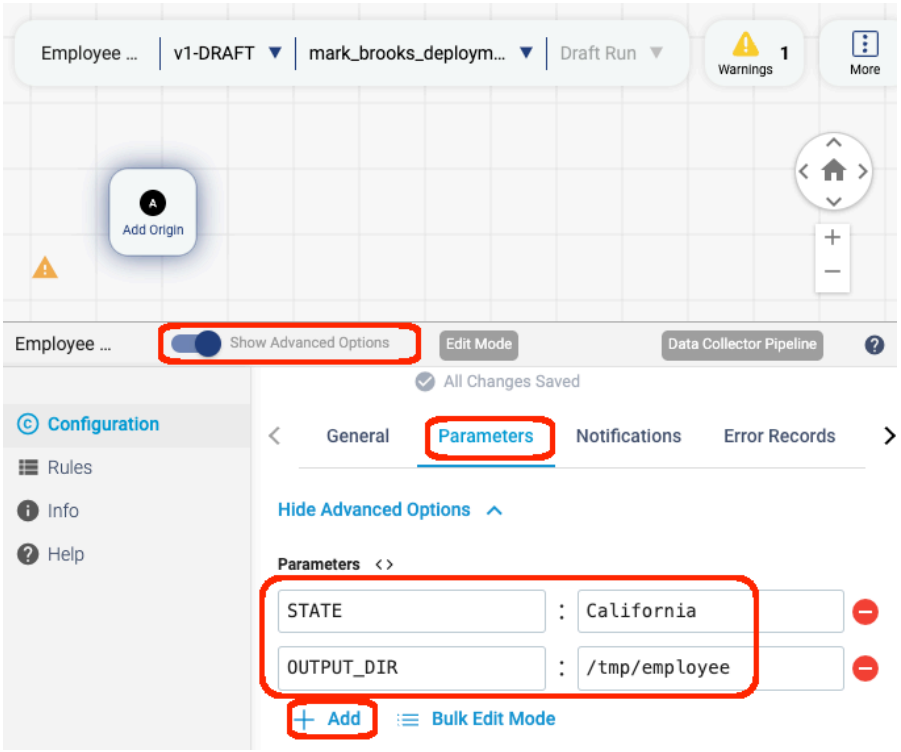


- You will see an empty pipeline canvas:



- You will add pipeline parameters to let users specify a state to filter employees by and an output directory for the pipeline to write to. For now, you'll just add the parameters to the pipeline; we'll explore pipeline parameters in depth in sections 3.3 and 4 below.

  To add the two pipeline parameters you need, enable the "Show Advanced Options" toggle, click on the Parameters tab, click the "Add Parameter" button twice, and add a parameter named `STATE` (in upper case) with a default value of "California" (without quotes) and a parameter named `OUTPUT_DIR` (in upper case) with a default value of "/tmp/employee" (without quotes) . The completed configuration should look like this:

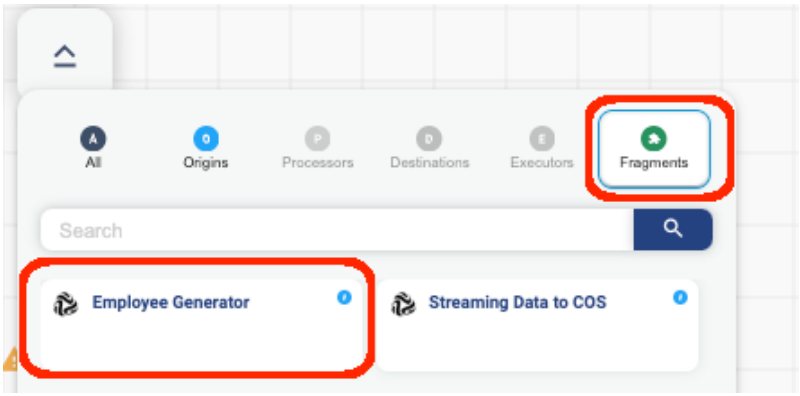


Note that the parameter values can be overridden at runtime by the user.

- Add an Employee Data Generator

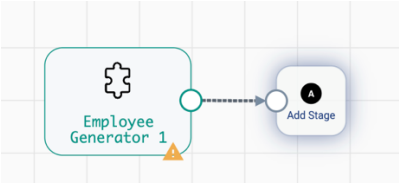
  You will use a preconfigured fragment that will generate employee data to simulate reading from a data source.

  Click the Add Origin widget on the pipeline canvas, filter by Fragments and click on the "Employee Generator" fragment:



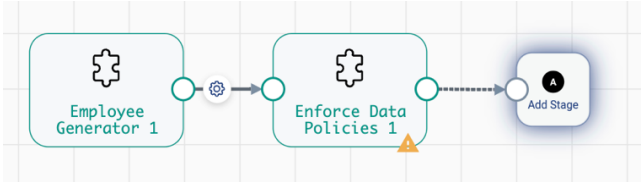
  When prompted for a Fragment Parameter, just click the Done button.

  Your pipeline should look like this:



- Add a fragment to enforce data policies. All enterprises have a set of standard data policies such as timestamping records and masking PII. You will add a fragment that is preconfigured to perform those actions.

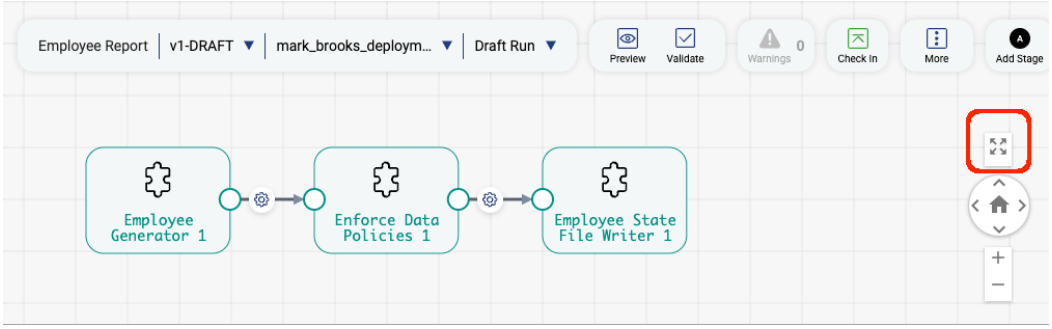
  Similar to the previous step, click the Add Stage widget, add the fragment named "Enforce Data Policies" and click Done when prompted for a Fragment Parameter. Your pipeline should now look like this:



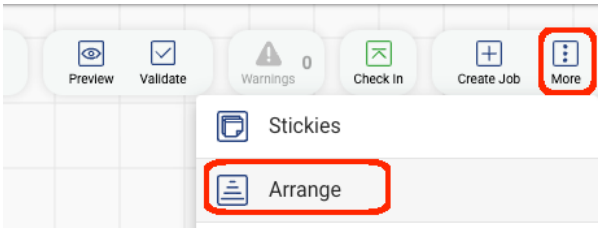
- Finally, you will add a fragment to filter employee records by state and to write the records to a user-defined location.

  Click the Add Stage widget, add the fragment named "Employee State File Writer" and click Done when prompted for a Fragment Parameter. Your pipeline should now look like this:

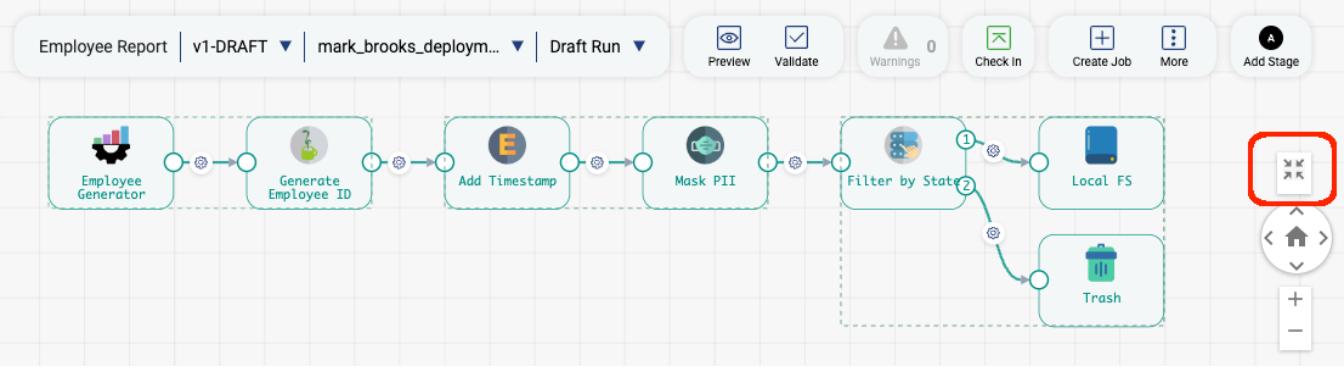
- To understand what's going on within each fragment, click the pipeline's "Expand" button:



  And then click the "Arrange" button which will either be in the button bar at the top of the pipeline canvas, or a menu item in the "More" menu like this:

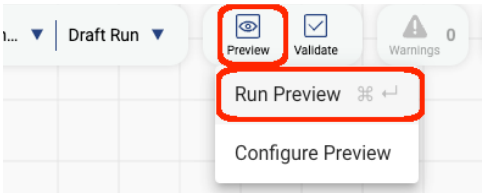


  This will reveal the stage libraries contained within each fragment:

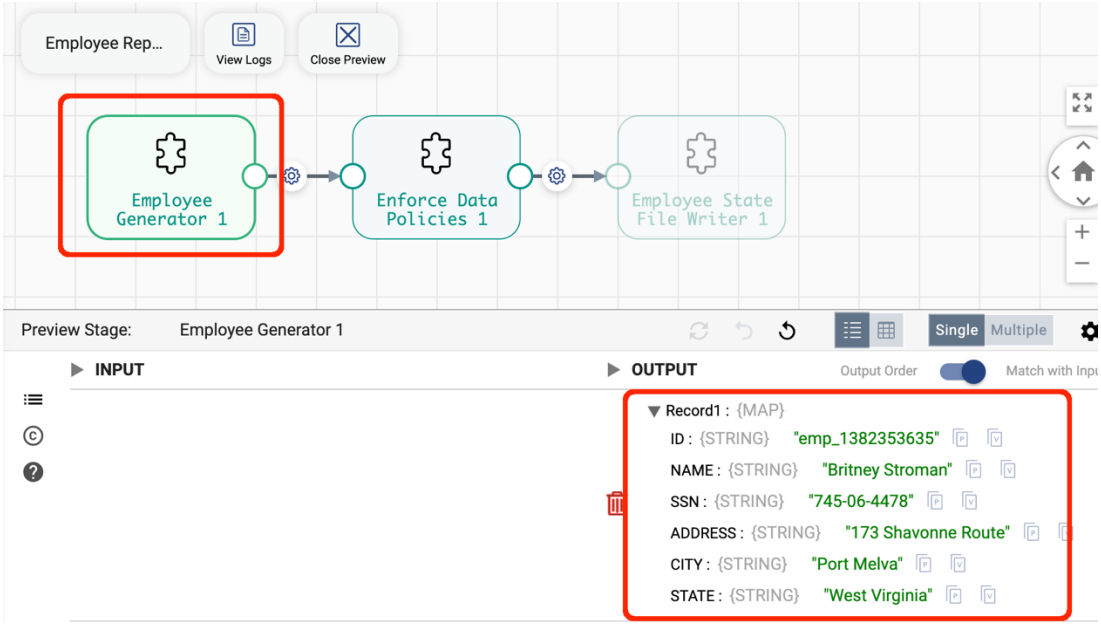


  Click the collapse button (circled above) to restore the original view of the pipeline.

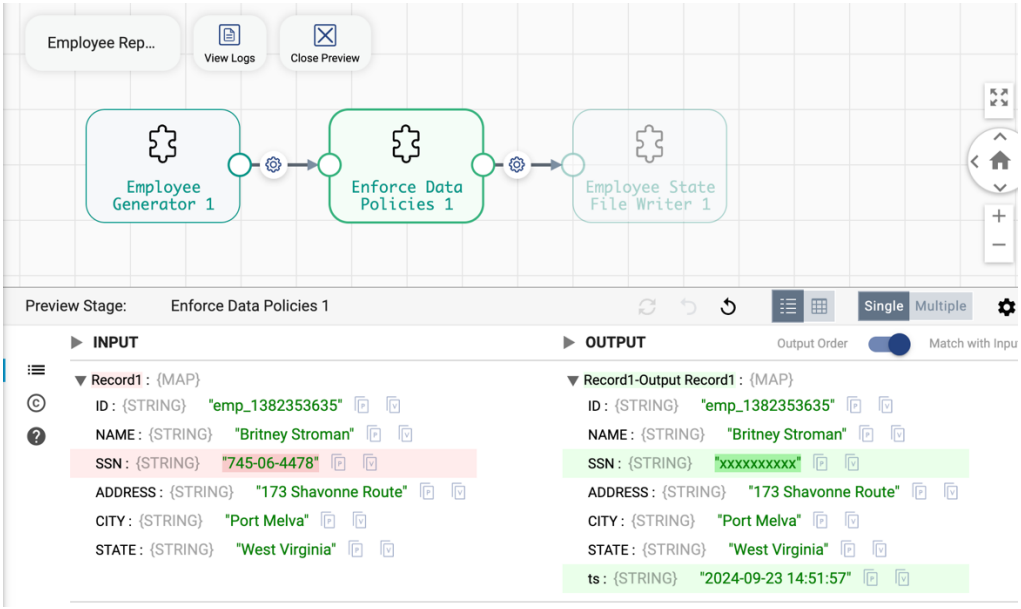
- Preview the pipeline! Pipeline preview allows pipeline developers to inspect how data moves through a pipeline at design time and by default will not write any data to downstream targets. Click the Preview > Run Preview item:



You will see sample records generated by the Employee Generator:



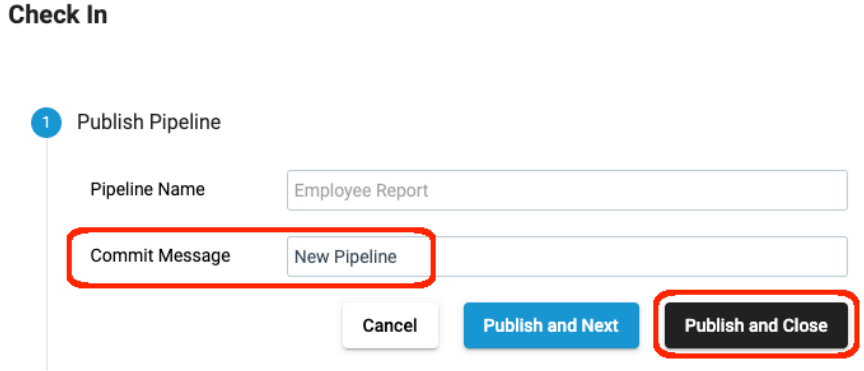
Click on the Enforce Data Policies stage to see the new timestamp field and the masked SSN filed:



Click the "Close Preview" button to exit the preview.

- Save this version of the pipeline by checking it into the StreamSets pipeline repository. Click the "Check In" button:

Provide a commit comment and click Publish and Close:

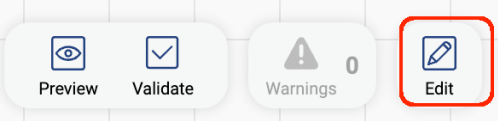


- You have completed the development of a pipeline using a set of pre-configured reusable fragments.

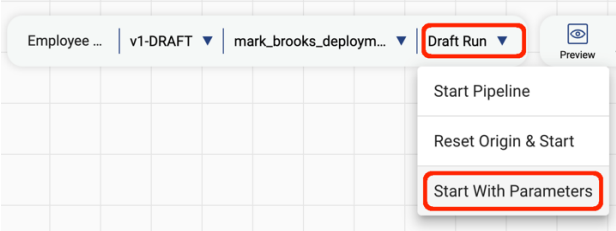
3.3 Parameterization

Pipeline Parameters allow pipelines to be configured at runtime by the user. For example, you can run the pipeline developed in the previous steps with a parameter value to select employees from Texas rather than California. Here is how to do that:

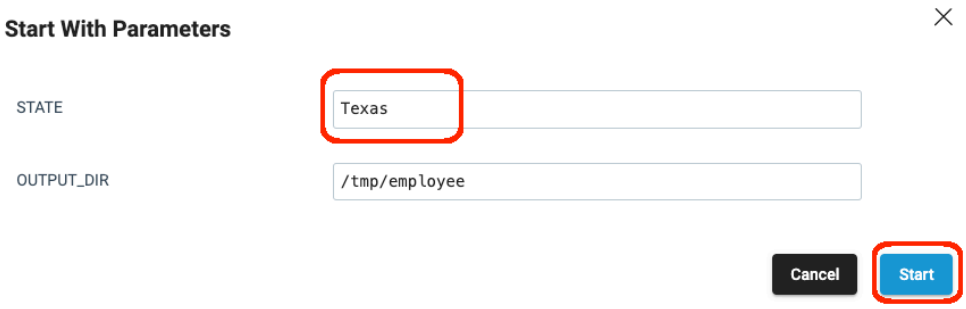
- Click the Edit button to open a new draft version of the pipeline:



- In the pipeline editor, select the menu item Draft Run > Start With Parameters:

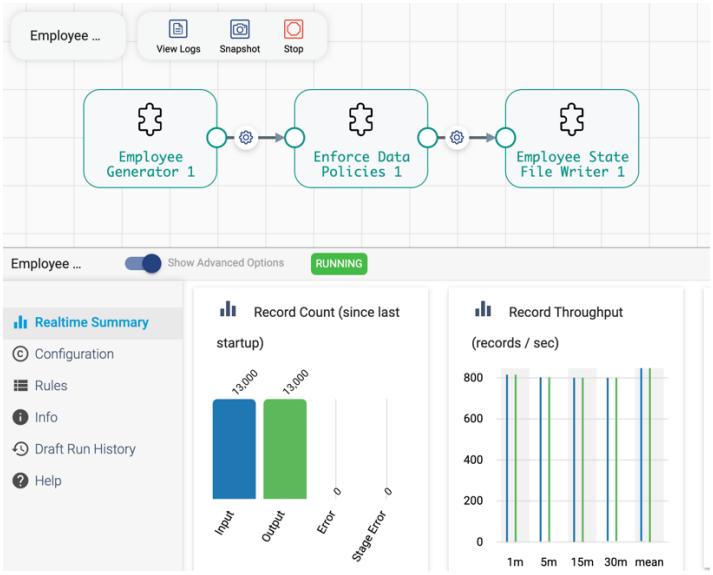


- Specify a different state, like "Texas" (without quotes) and keep the default output directory value:



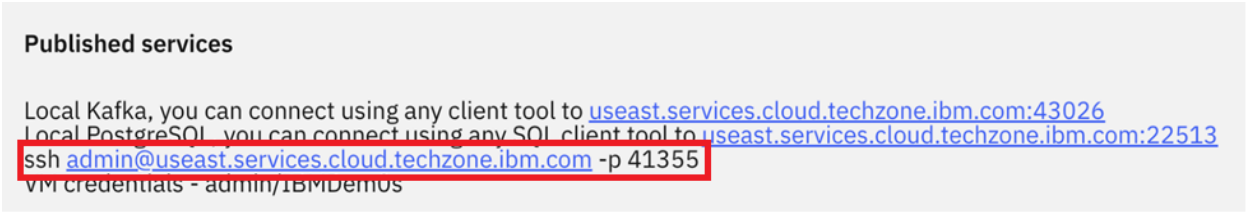
- Click the Start button.

  As the pipeline runs, note the runtime metrics:



After the pipeline has run for a minute or so, stop the pipeline.

- Inspect the data written by the pipeline by either opening a terminal session on the Dataplane VM as shown in section 2.3 above, or by ssh'ing to the Dataplane VM using the ssh command shown in your TechZone Reservation's Published services page like this (your port number will be different):



- Once your terminal or ssh session is open, execute an `ls` command to confirm a directory named "Texas" was created within the specified output directory with one or more files in it:

```
$ ls -l /tmp/employee/Texas/

-rw-r--r--  1 mark  wheel  39069 Sep 23 15:42 018ad0de-2456-4aef-9d44-623a8aa2d7ea.json
```

- Execute a `cat` command to inspect the data within the output file(s) to confirm the employee records are from Texas, the SSN field is masked, and records are timestamped. For example, in my environment:

```
$ cat /tmp/employee/Texas/018ad0de-2456-4aef-9d44-623a8aa2d7ea.json | head

{"ID":"emp_1287356724","NAME":"Greg Stokes","SSN":"xxxxxxxxxx","ADDRESS":"036 DuBuque
Field","CITY":"Ashlibury","STATE":"Texas","ts":"2024-09-23 15:42:37"}

{"ID":"emp_1497265364","NAME":"Jonathon Gutmann","SSN":"xxxxxxxxxx","ADDRESS":"16372
Cremin Square","CITY":"Tarshaville","STATE":"Texas","ts":"2024-09-23 15:42:37"}

{"ID":"emp_1736121568","NAME":"Virgil Zemlak","SSN":"xxxxxxxxxx","ADDRESS":"99010
Graham Plaza","CITY":"West Shannanview","STATE":"Texas","ts":"2024-09-23 15:42:37"}
```

- This confirms you are able to run pipelines using different parameter values.

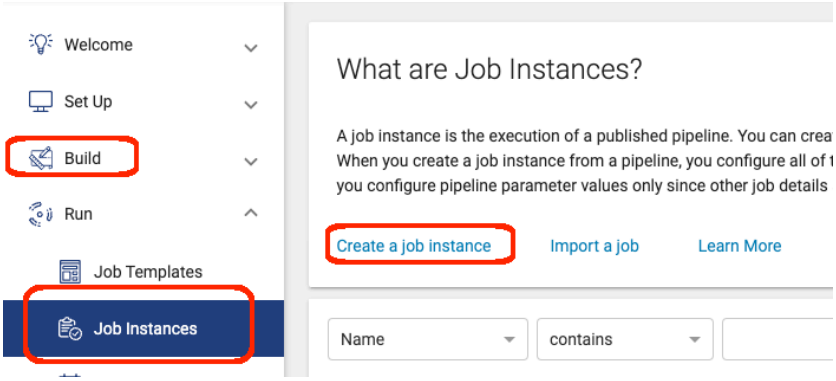
4 Operationalizing pipelines

4.1 Jobs

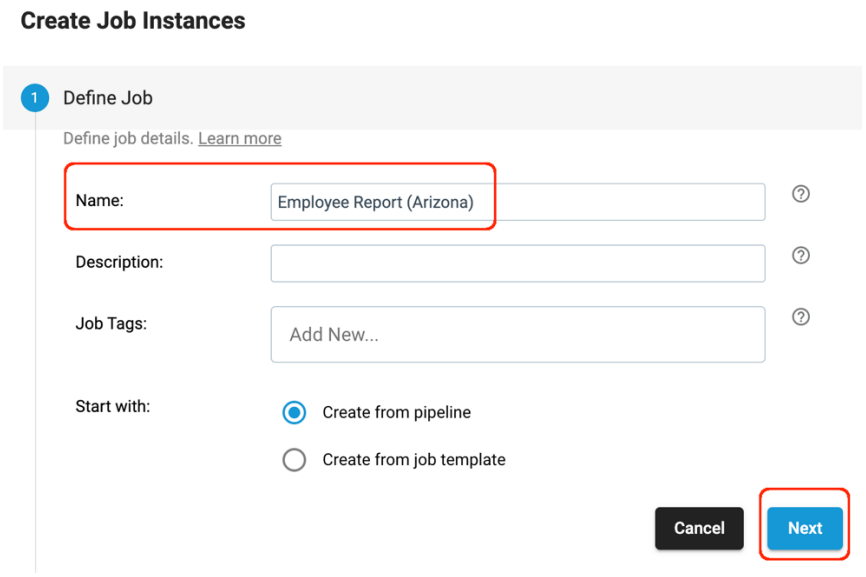
In the previous section, a pipeline was developed and run within the Pipeline Editor. To run a pipeline in Production, one would create a StreamSets Job. A StreamSets Job is configured with three main settings:

  - A specific version of a pipeline,
  - A specific set of parameter values.
  - One or more "Engine Labels" which determine which StreamSets engine the Job's pipeline will run on.

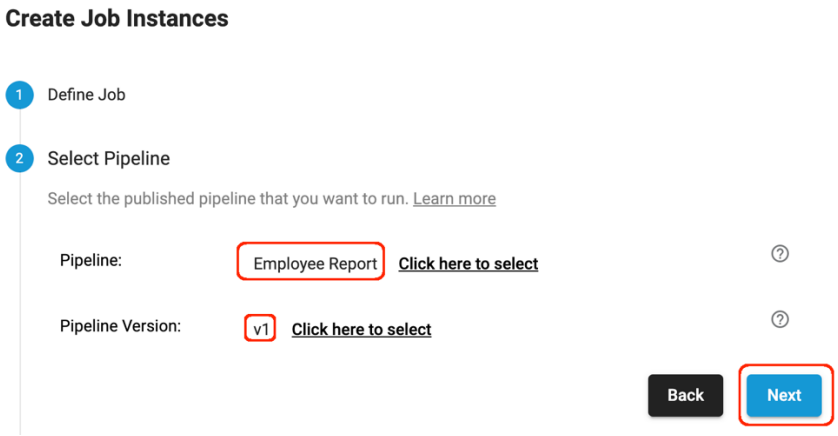
- Create a Job by navigating to Run > Job Instances and clicking the "Create a Job Instance" link:



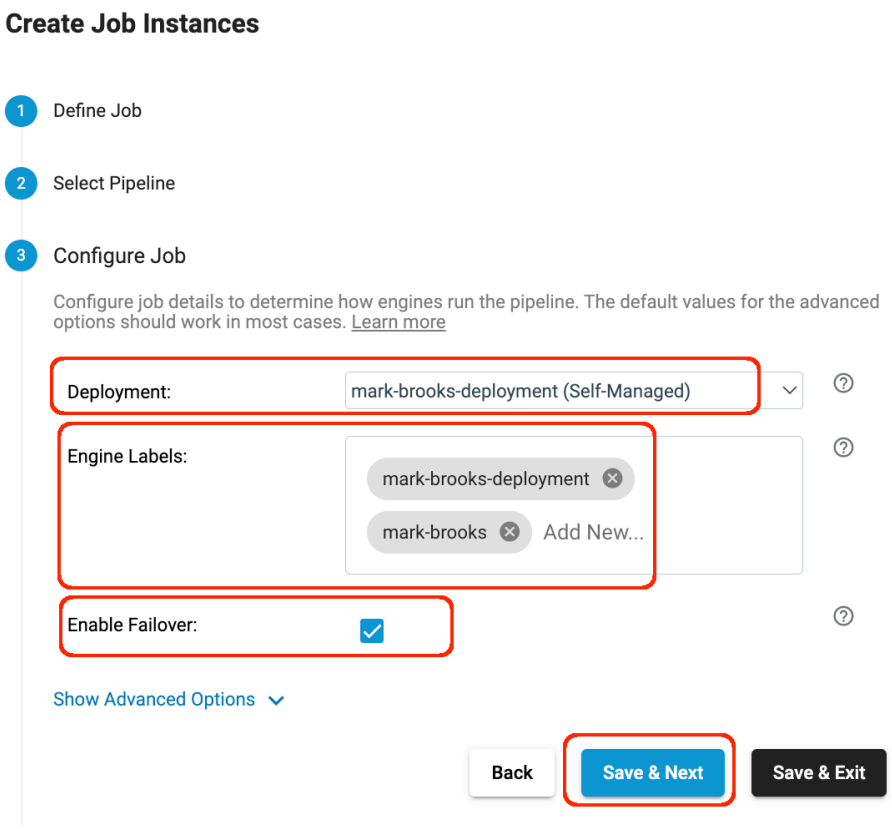
- Name the Job "Employee Report (Arizona) and click the Next button



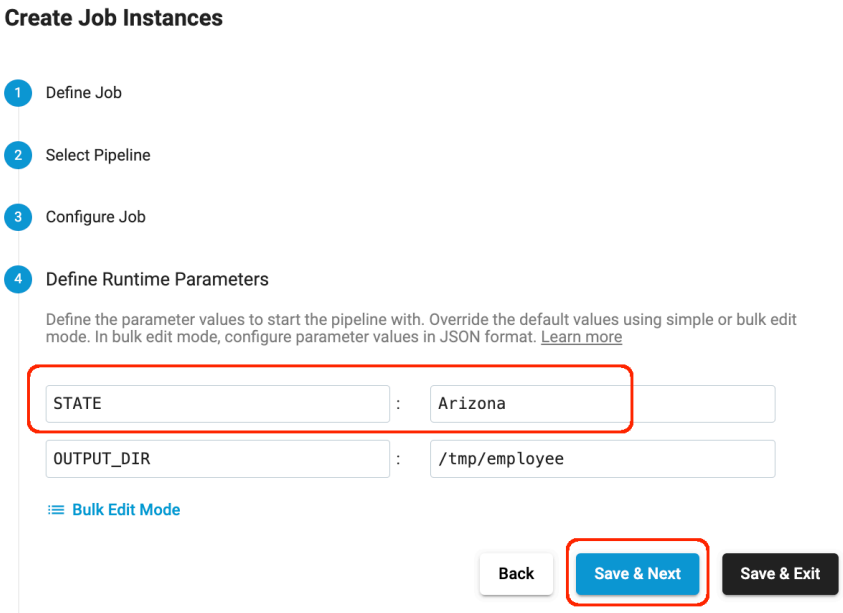
- Click the "Click here to select" link and select version 1 of the Employee Report pipeline. The dialog should look like this after you have selected the pipeline. Click the Next button:



- Select your deployment, which should set the engine labels that match your engine. Set the "Enable Failover" checkbox (you'll learn about failover in Section 4.4 below), and click Save & Next.



- Set the STATE parameter to Arizona and then click Save & Next:



- Click the Start & Monitor Job button and you should see the Job running:

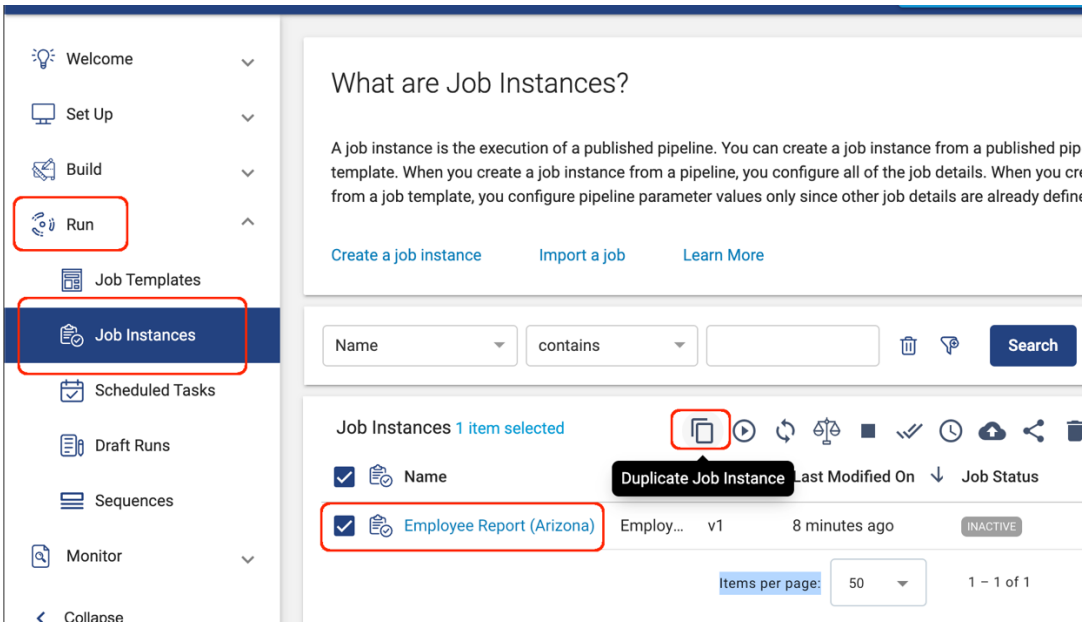
- Click the Stop button to stop the pipeline
- In your terminal session or ssh session, execute an `ls` command to confirm a directory named "Arizona" was created within the specified output directory with one or more files in it:

```
$ ls -l /tmp/employee/Arizona

-rw-r--r--  1 mark  wheel  514671 Sep 24 12:20 7e99ab34-2d49-432c-9cb6-5b780e5644fe.json
```

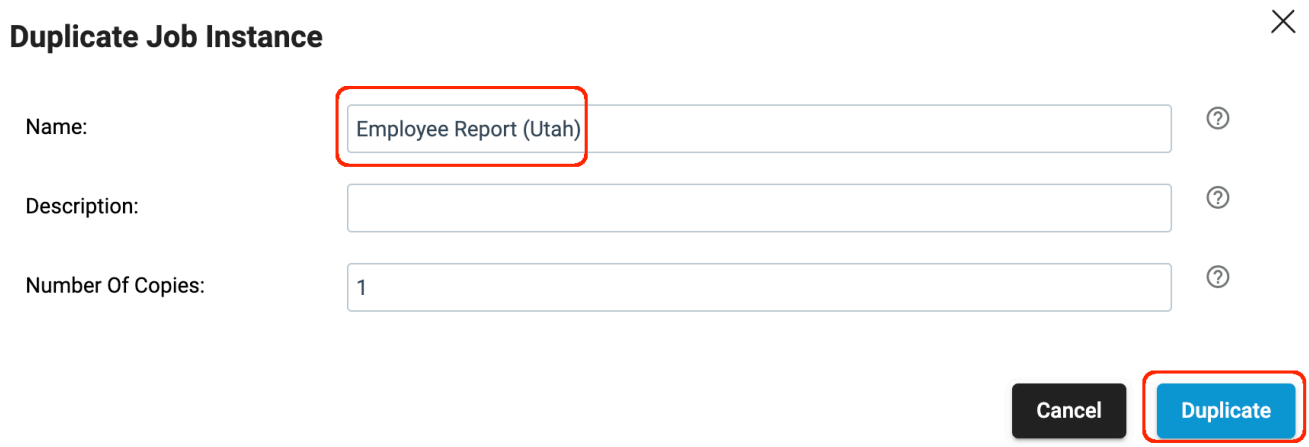
- Execute a `cat` command to inspect the data within the output file(s) to confirm the employee records are from Arizona. For example, in my environment:

```
$ cat /tmp/employee/Arizona/7e99ab34-2d49-432c-9cb6-5b780e5644fe.json | head

{"ID":"emp_887266273","NAME":"Jesus Wolf","SSN":"xxxxxxxxxx","ADDRESS":"5800 Davis
Route","CITY":"Pricemouth","STATE":"Arizona","ts":"2024-09-24 12:16:55"}

{"ID":"emp_1098627699","NAME":"Adan Hane II","SSN":"xxxxxxxxxx","ADDRESS":"145 Mariella
Mountains","CITY":"West Monteberg","STATE":"Arizona","ts":"2024-09-24 12:16:55"}

{"ID":"emp_507653370","NAME":"Christopher O'Kon","SSN":"xxxxxxxxxx","ADDRESS":"14157
Stacy Estates","CITY":"East Florentina","STATE":"Arizona","ts":"2024-09-24 12:16:55"}
```

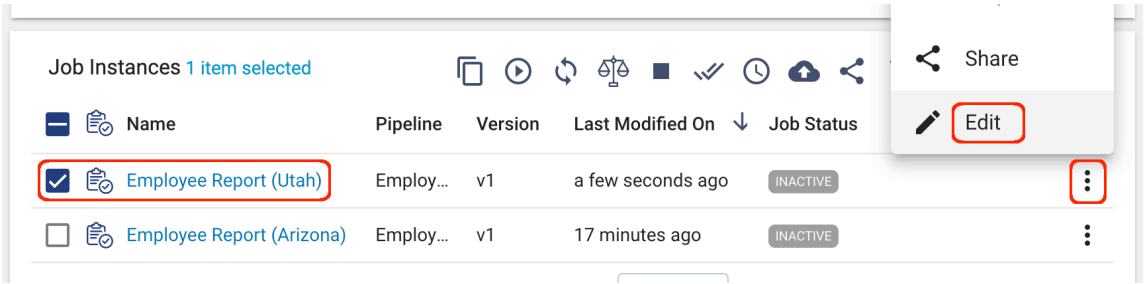
- Next, you will duplicate the Job and set a different value for the STATE parameter. Select the Job from the Job Instances list and click the Duplicate button:



- Change the Job's name to "Employee Report (Utah)" and click the Duplicate Button:

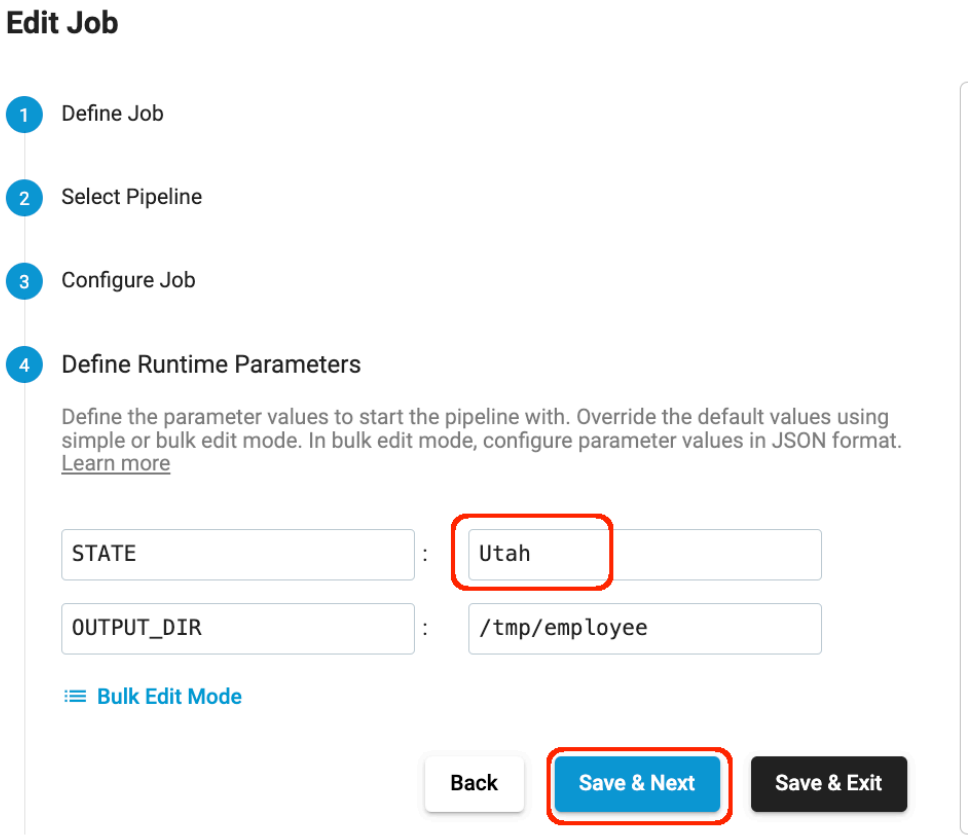


- Select the new Job from the Job Instances list and click its Edit item:



- Click the Next button in each of the edit screens until you reach the Define Runtime Parameters screen.

- Change the STATE to Utah and click Save & Next:



- Click Start and Monitor Job and confirm the Job starts and then click the Stop button:

- Once the pipeline has stopped, confirm you have a Utah directory and data:

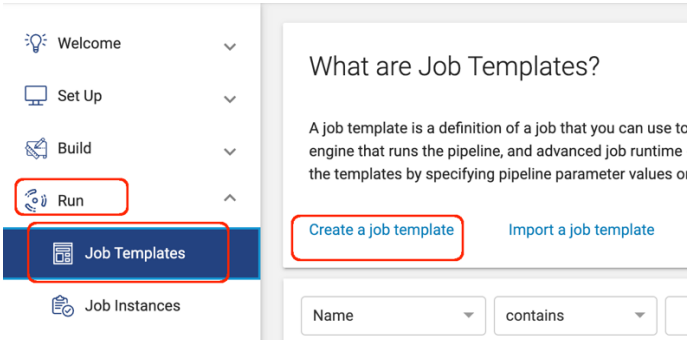
```
$ ls -l /tmp/employee/Utah

-rw-r--r--  1 mark  wheel  478356 Sep 24 12:42 860a40d5-e36e-4f04-908e-c0f1d7072aee.json

$ cat /tmp/employee/Utah/860a40d5-e36e-4f04-908e-c0f1d7072aee.json | head

{"ID":"emp_491023667","NAME":"Lynda Pfeffer","SSN":"xxxxxxxxxx","ADDRESS":"5375
Sheron Square","CITY":"Rosechester","STATE":"Utah","ts":"2024-09-24 12:39:20"}

{"ID":"emp_726593617","NAME":"Philomena
Powlowski","SSN":"xxxxxxxxxx","ADDRESS":"08449 Mann Mall","CITY":"South
Tamekia","STATE":"Utah","ts":"2024-09-24 12:39:20"}

{"ID":"emp_96434058","NAME":"Julietta Beahan","SSN":"xxxxxxxxxx","ADDRESS":"8449
Crooks Curve","CITY":"Lake Edwin","STATE":"Utah","ts":"2024-09-24 12:39:20"}
```

- This confirms you are able to configure and run parameterized Jobs without needing to edit the Job's underlying pipeline.

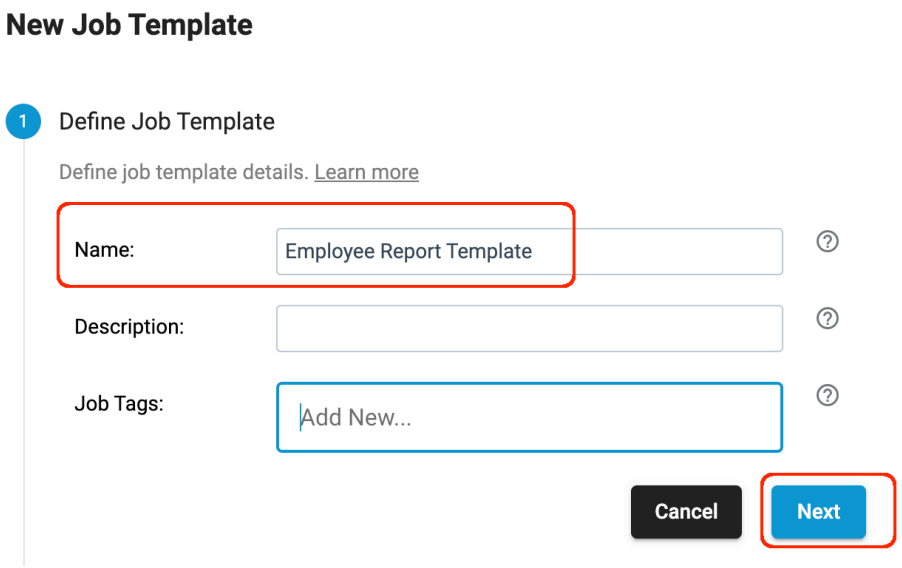
4.2 Job templates

Job Templates allow multiple Job instances to be created, parameterized, and launched using a template. In this section you will create a Job Template for the Employee Report pipeline, and then use the Job Template to launch three instances of the pipeline, each for a different state.

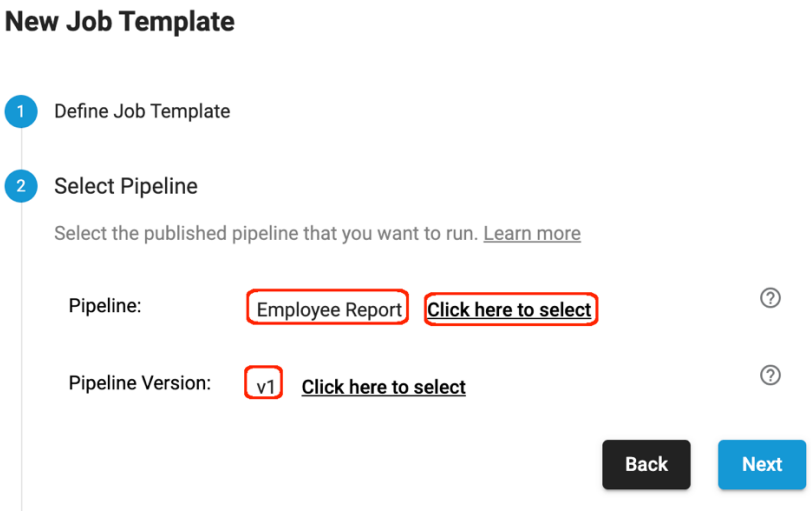
- Start by creating a new Job Template:



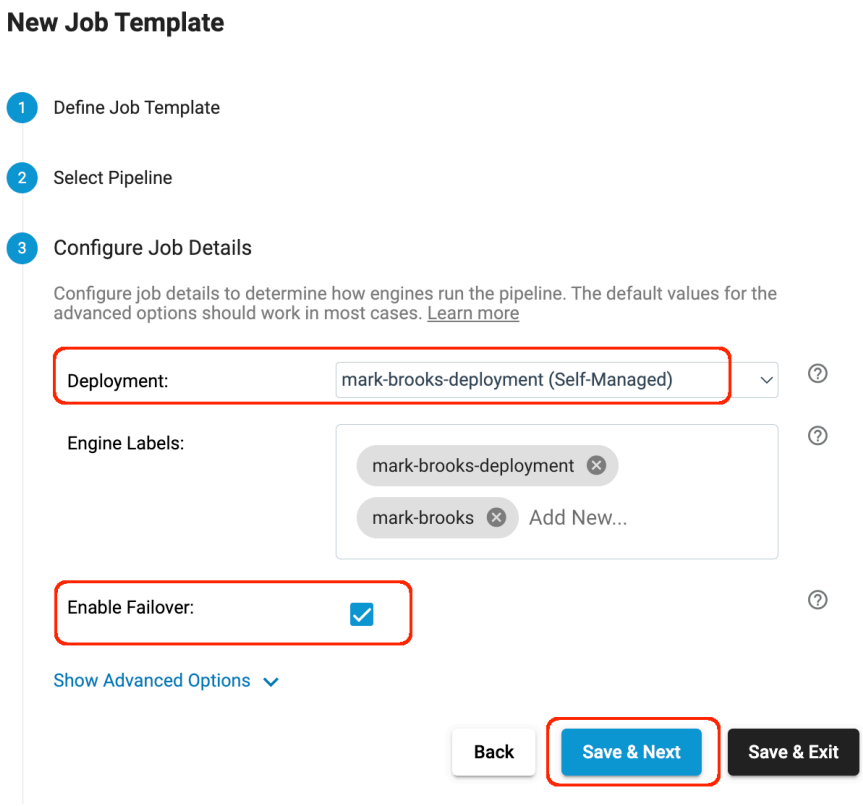
- Name the template "Employee Report Template" and click Next:



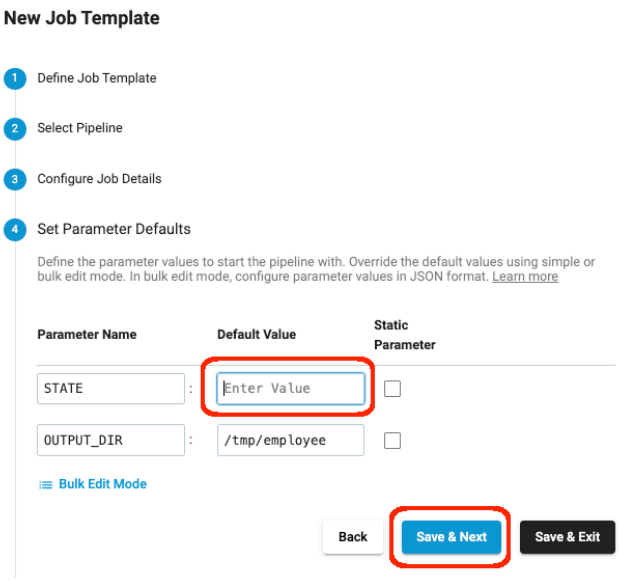
- Click the "Click here to select" link and select version 1 of the Employee Report pipeline. Click Next:



- Select your deployment to get the engine labels for your engine and set the Enable Failover checkbox. Click Save & Next:

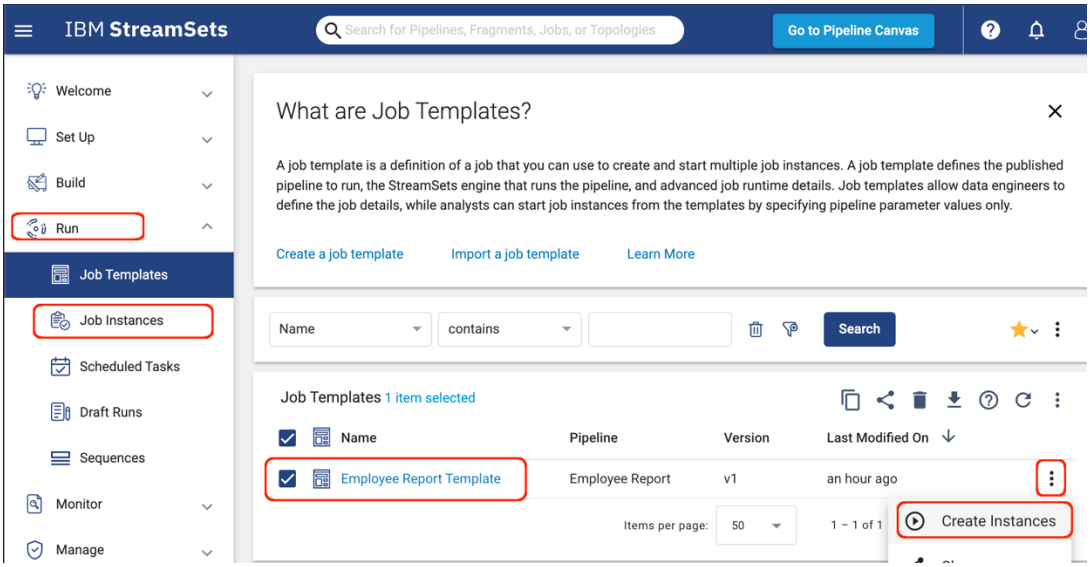


- Delete the default value for the STATE parameter, so users will be forced to specify a state when using the template and leave the default value for the OUTPUT_DIR parameter. Click Save & Next:

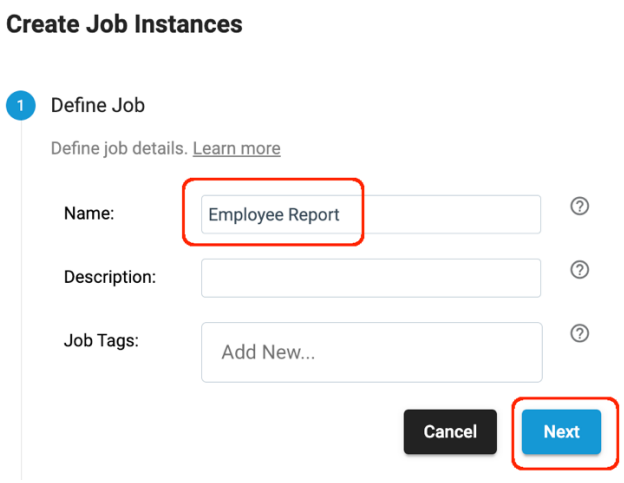


- Click Exit:

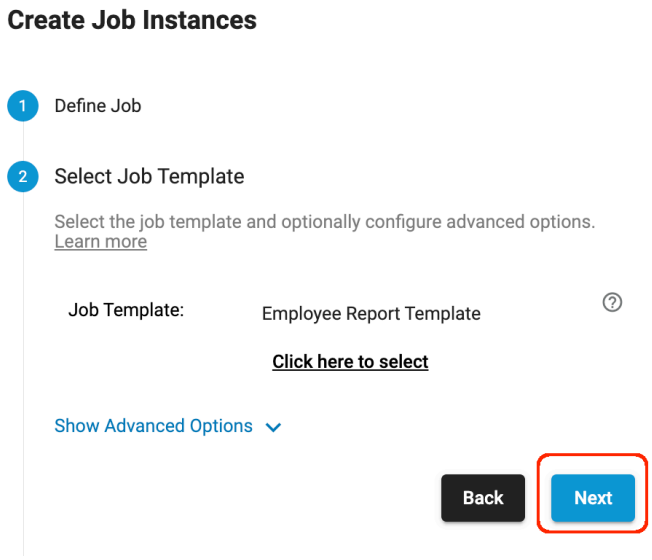
- You have successfully created a Job Template. Note that no Jobs have been launched yet; you just created a template. Now you will use the template to launch three separate Employee Reports Job instances.
- Open the Job Templates list, select your template, and click the three dots menu and select the Create Instances item:



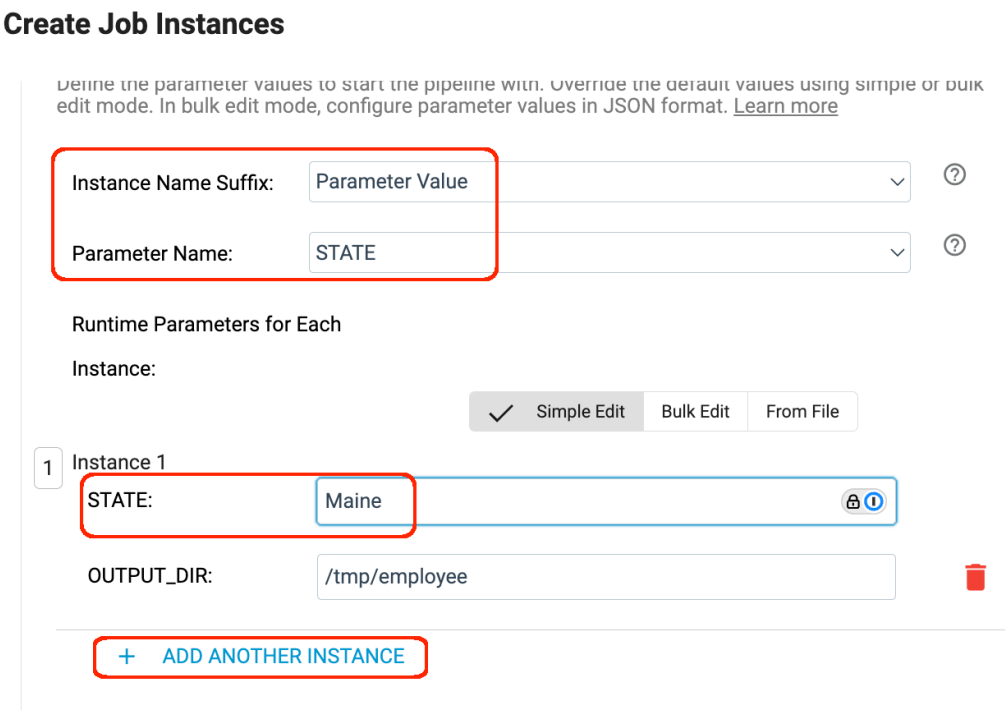
- Enter the name "Employee Report" and click Next:



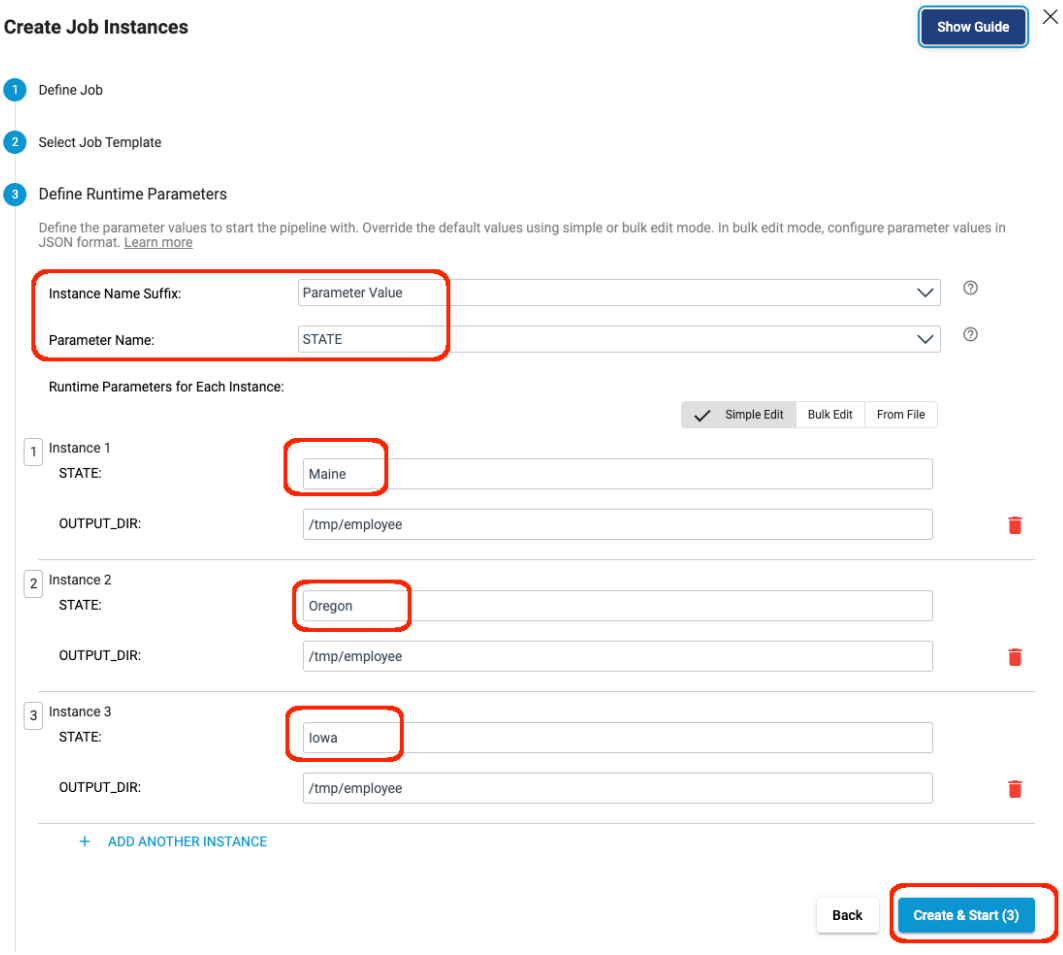
- Accept the default settings in this dialog and click Next:



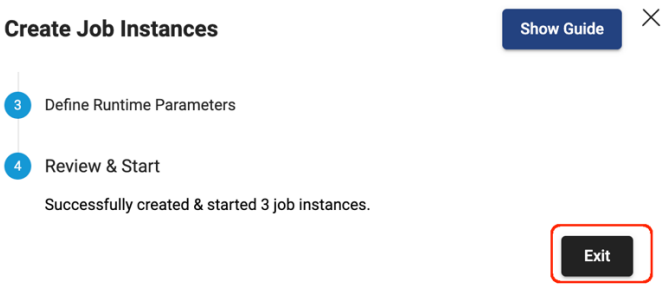
- In the dialog that opens, set the "Instance Name Suffix" property to "Parameter Value" and set the "Parameter Name" to "State". These settings will make it easy for you to identify which state each Job instance is running a report for. Enter a value of "Maine" for the STATE parameter, then click the ADD ANOTHER INSTANCE link:



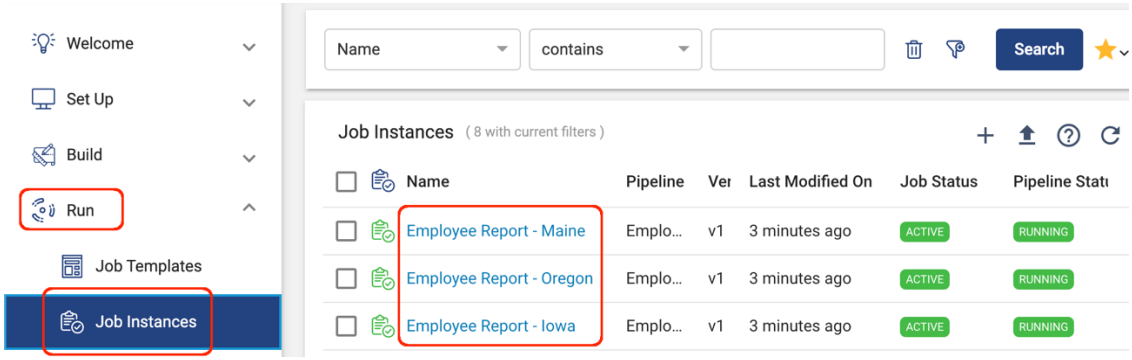
- Set the STATE for the second instance to Oregon and click the ADD ANOTHER INSTANCE link again and enter set the STATE for the third instance to Iowa. The completed dialog should look like this:



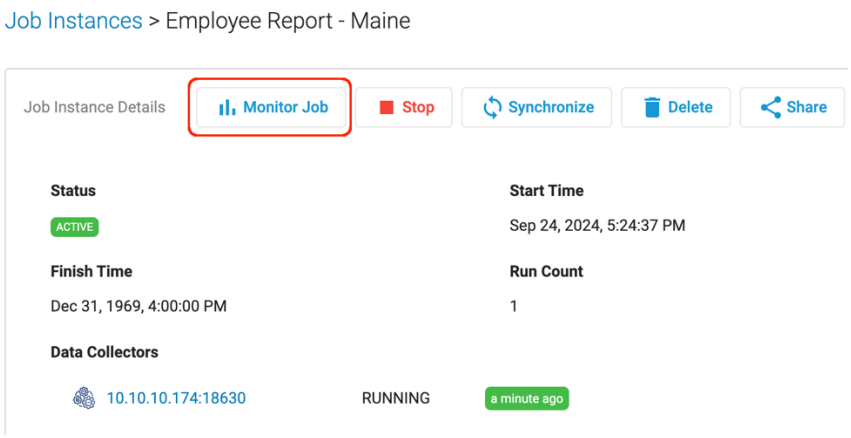
- Click the Create & Start (3) button to launch three Job instances, each for a different state.
- You should see a message confirming the three Job instances were launched. Click the Exit button:



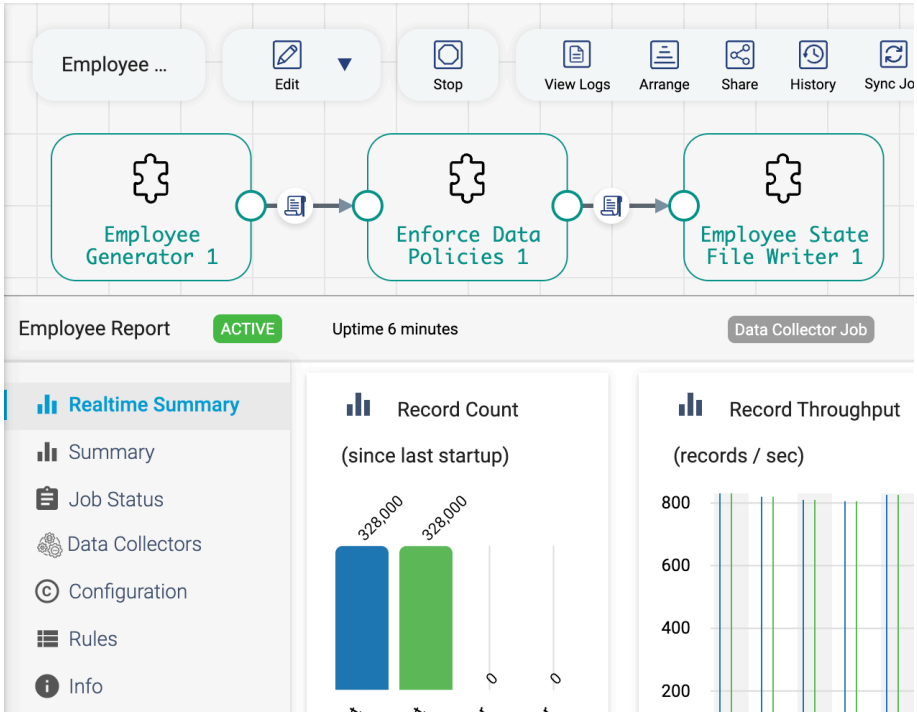
- Navigate to the Job Instances list and you should see three jobs are running. Note that each Job instance name includes the STATE parameter value set for it:



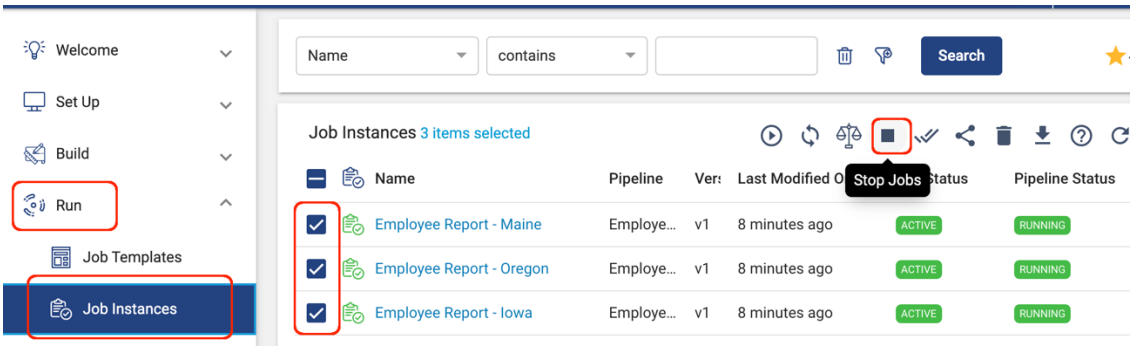
- Click on the Job instance for Maine to see e Job instance detail pane and then click on the Monitor Job button:



- You will see metrics for the running Job instance:



- Return to the Job Instances list, select all three running Job instances, and click the Stop Jobs button:



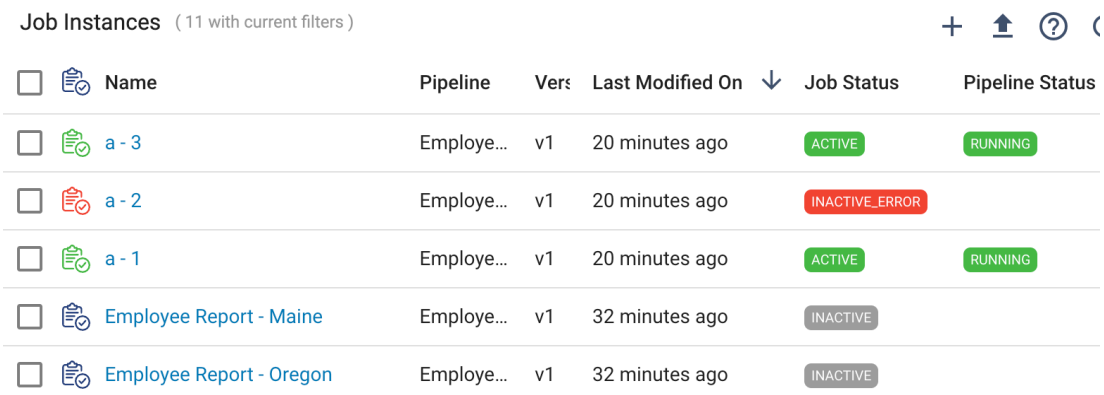
- Confirm you now have data directories written for the three new states:

```
$ ls -l /tmp/employee
drwxr-xr-x  3 mark  wheel   96 Sep 24 17:36 Arizona
drwxr-xr-x  4 mark  wheel  128 Sep 24 17:35 Iowa
drwxr-xr-x  4 mark  wheel  128 Sep 24 17:35 Maine
drwxr-xr-x  4 mark  wheel  128 Sep 24 17:35 Oregon
drwxr-xr-x  3 mark  wheel   96 Sep 24 17:36 Texas
drwxr-xr-x  3 mark  wheel   96 Sep 24 17:36 Utah
```

4.3 Monitoring

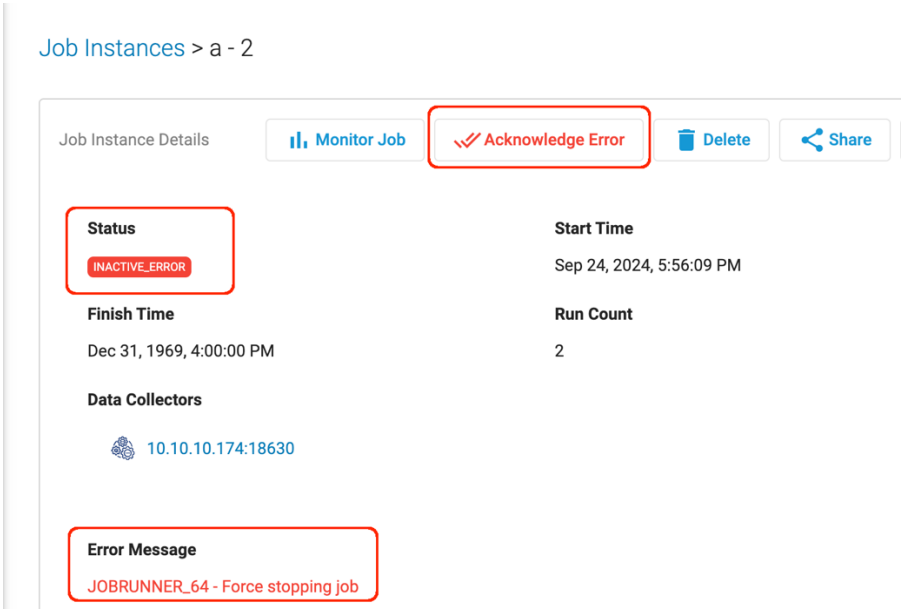
IBM StreamSets provides a number of mechanisms for monitoring Job execution and engine health. You will look at Job status, Job metrics, Job History, and the Operations Dashboard. The discussion and screenshots below also cover Alerts and Subscriptions.

- **Job Status**: Job instances have a status and status color that is one of the following:
    - INACTIVE (grey)
    - INACTIVE (red)
    - INACTIVE_ERROR (red)
    - ACTIVATING (green)
    - ACTIVE (green)
    - ACTIVE (red)
    - DEACTIVATING (green)
    - DEACTIVATING (red)
- For example, the Job instances list view might look like this:



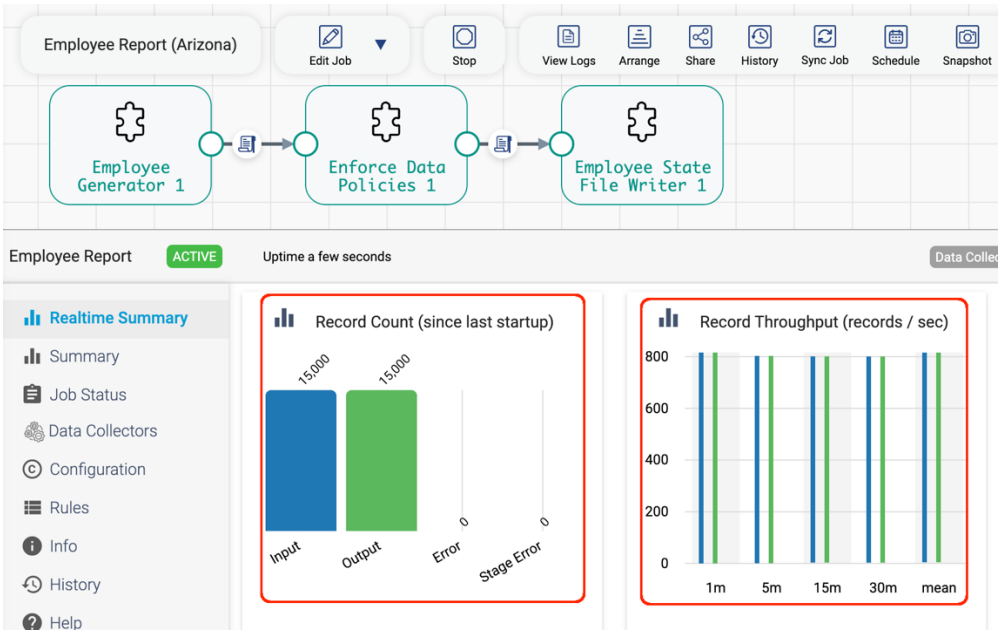
Note that two Job instances are running in a healthy (green) state, two Job instances completed without errors – they are INACTIVE (grey)) – and one Job instance is unhealthy – it is in an INACTIVE_ERROR (red) state.

- To view Job instance details, click on the Job instance name. For example, the detail for the unhealthy Job shown above looks like this:

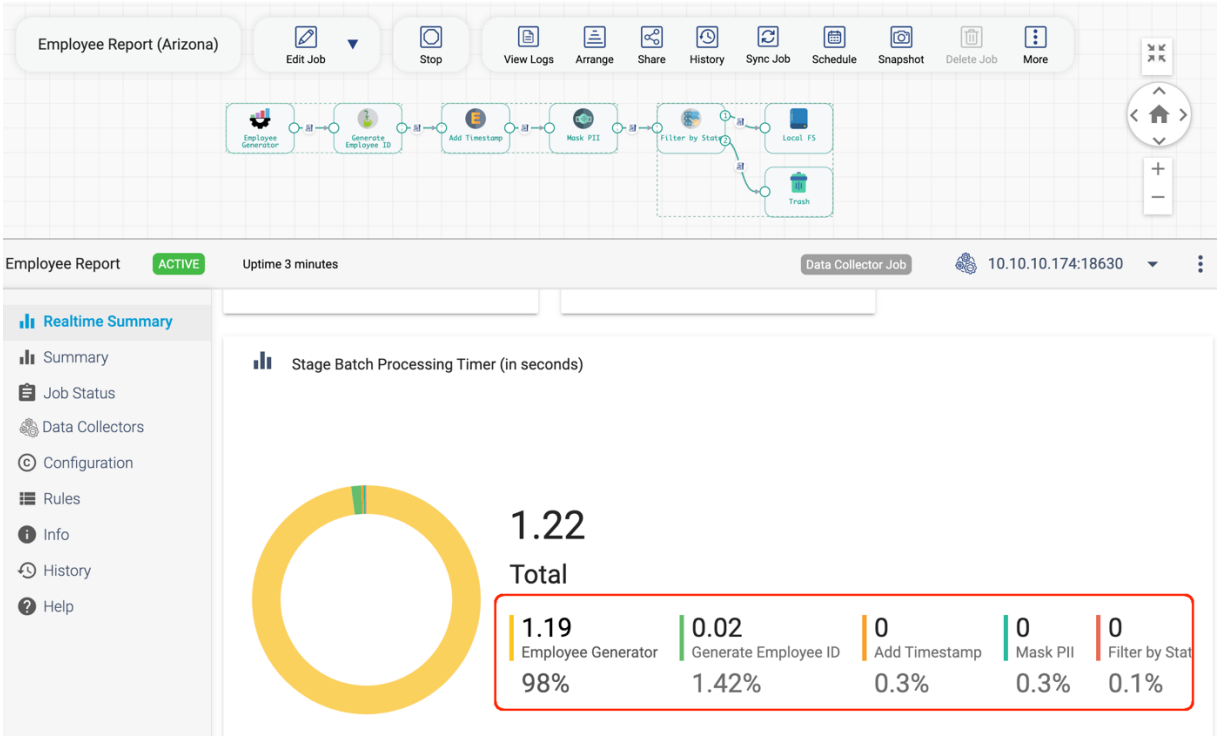


The error message shows the Job was "force stopped" and the operator will need to click the Acknowledge Error button once the underlying condition has been resolved and the Job can be run again.

- Job Metrics: Job metrics are displayed in a "Realtime Summary" for running Jobs. For example, start an Employee report Job, go to its monitoring page and you will see metrics like this:

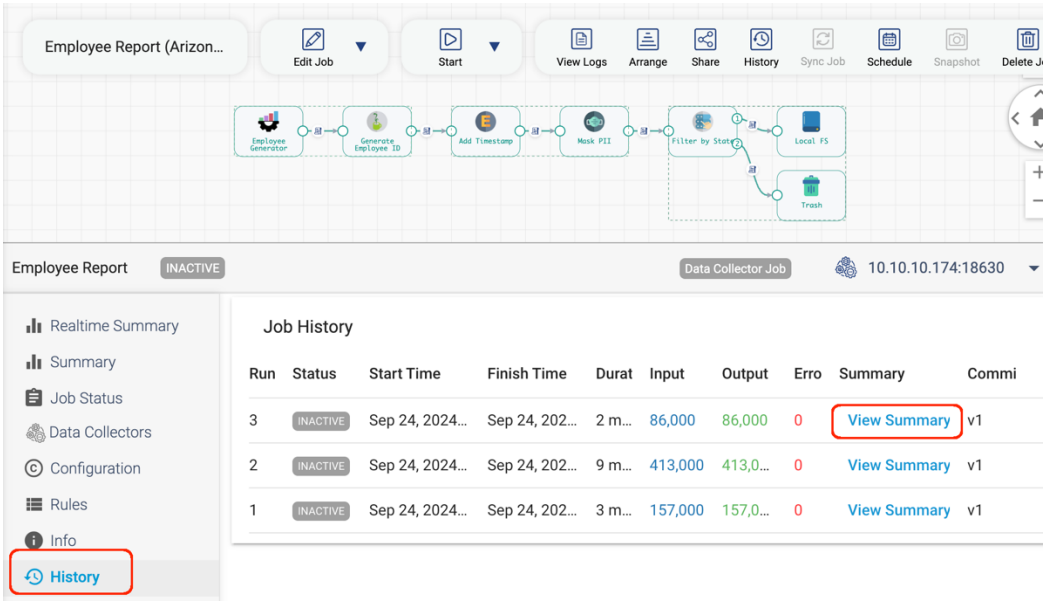


- Expand and arrange the pipeline fragments and scroll down to see where the pipeline is spending its time:

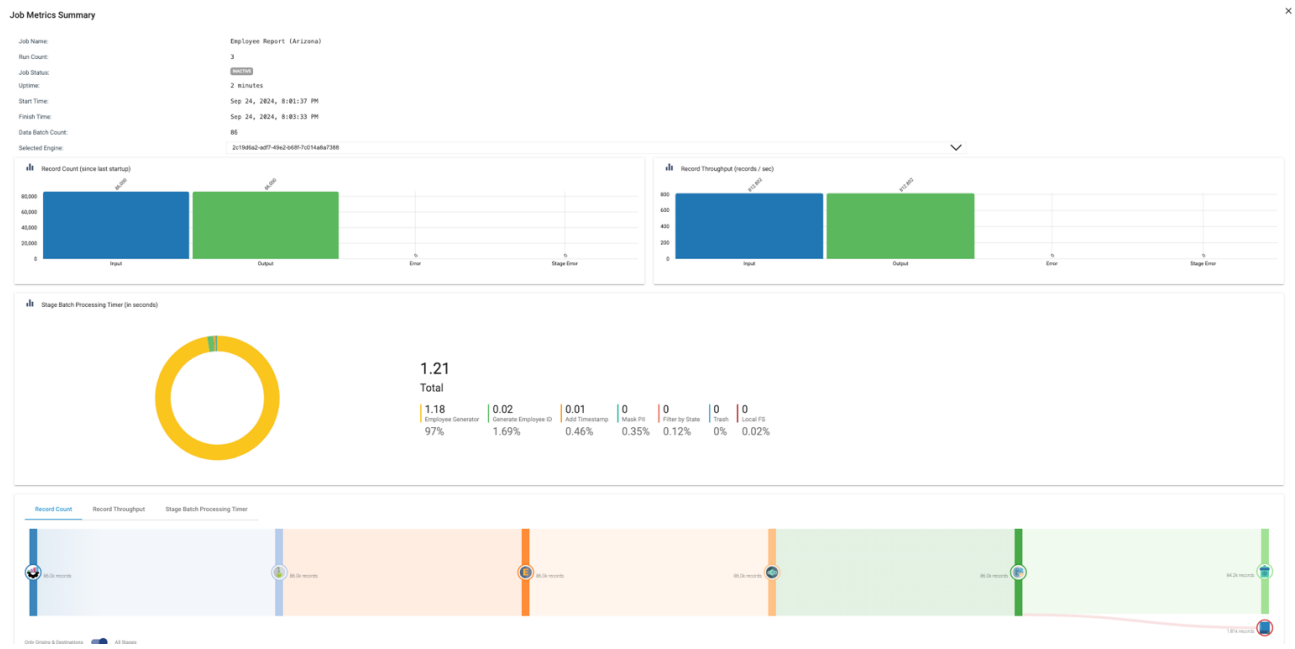


Note that 98% of the pipeline's time is spent in the Employee Generator stage, generating an ID takes 1.42% of the time, and the remaining processing steps take less than 1% each. This view allows you to identify if a poorly implemented processor is a bottleneck, or if a processor is slowed down by a high latency remote call.

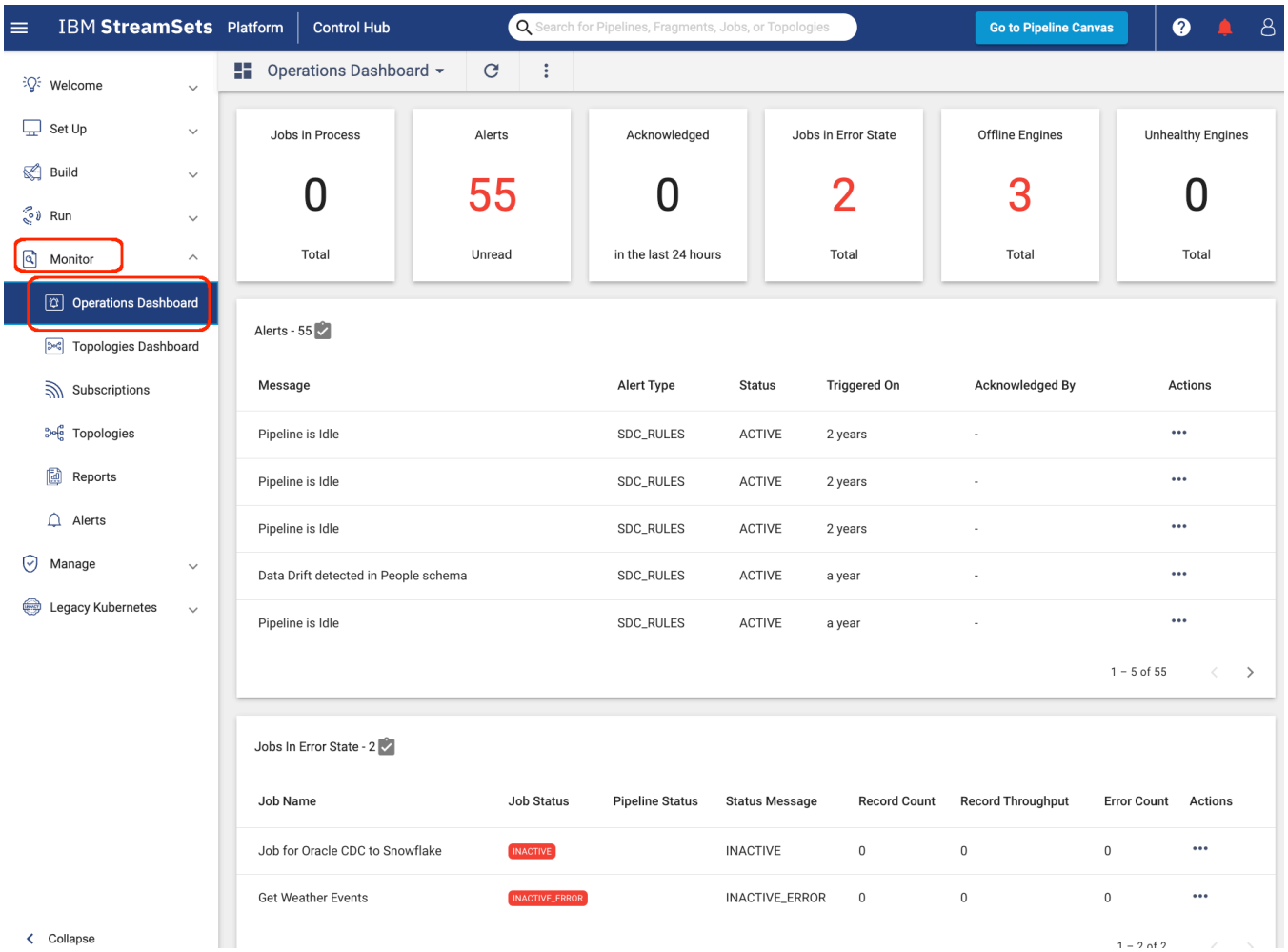
- Job History: Click on the History link on the left side of the monitoring pane. You will see metrics and statistics for each run:



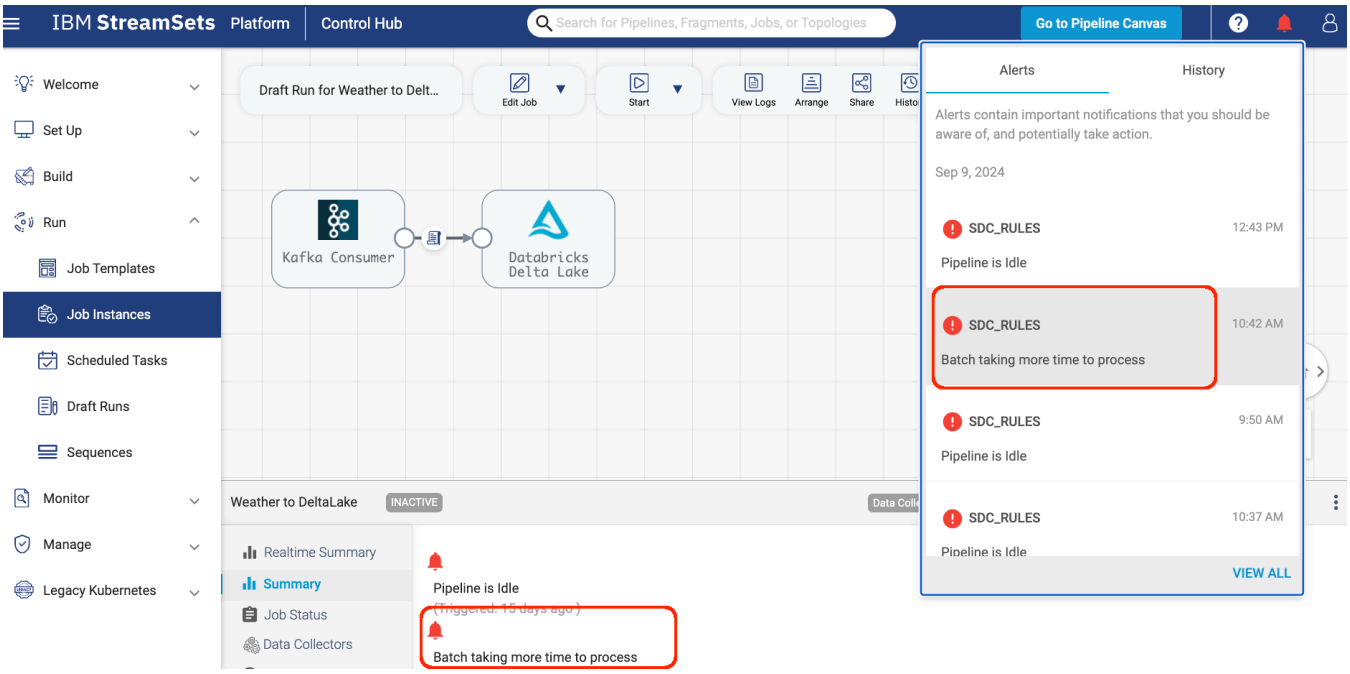
- Click the View Summary link for each entry in the history to see a wealth of additional information:



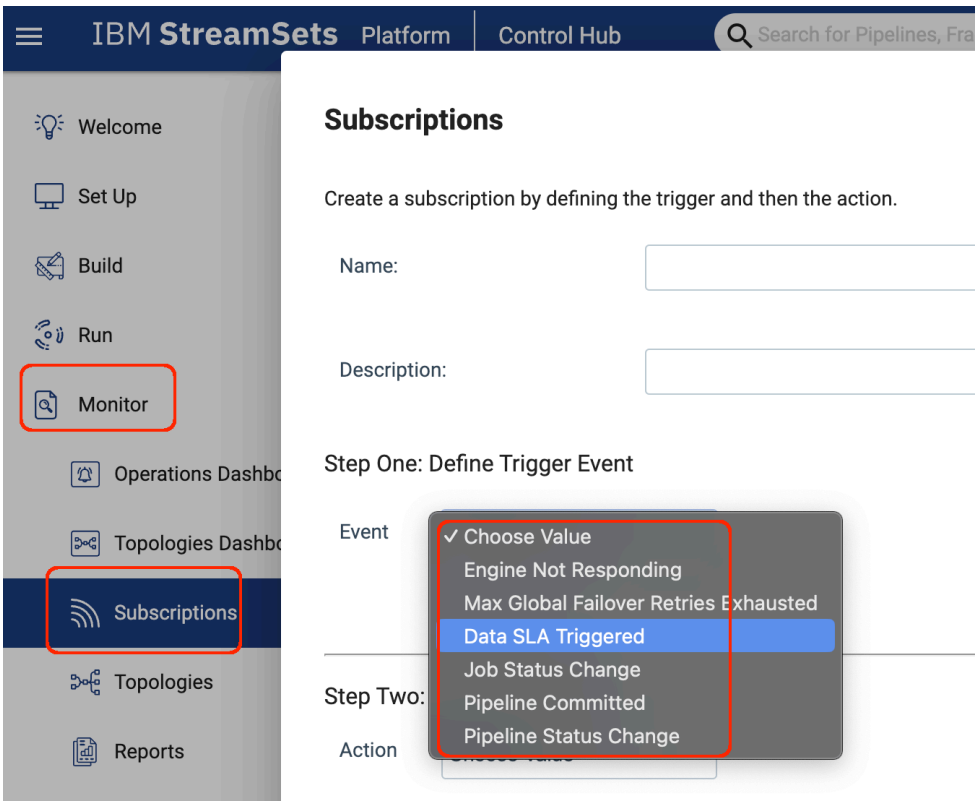
- **Operations Dashboard**: The Operations Dashboard provides essential information to operators regarding Jobs in process, alerts, Jobs in an error state, offline or unhealthy engines, etc...



- **Alerts**: IBM StreamSets provides a wide variety of built-in and user-configurable alerts that can be used to generate push notifications using email or webhooks to systems like ServiceNow, PagerDuty, Slack, etc...
- Alerts are also displayed with the UI with the ability to drill-down the pipeline that generated the alert. Here is an example:



- **Subscriptions**: [*There are no user exercises for this topic, it is only for your information*]. IBM StreamSets allows operators to "subscribe" to a variety of different events to trigger push notifications:



- Subscription events can be filtered with user-defined conditions, such as to send a webhook notification for any production Job that transitions from a state of ACTIVE (green) to ACTIVE (red), while ignoring issues for non-production Jobs. Here is an example webhook payload for such a Subscription:

```
The Job "Oracle CDC to Kafka"  with Job ID 23349e92-48c6-4d35-980e-c4dbe86139f4:schbrooks changed status from GREEN
ACTIVE to RED ACTIVE triggered on 2021-06-24 18:28:09 with the error message "JOBRUNNER_63 - Pipeline status:
'STARTING_ERROR', message: 'KAFKA_68 - Error getting metadata for topic 'g1' from broker 'sequoia.onefoursix.com:9092' due
to error: org.apache.kafka.common.errors.TimeoutException: Timeout expired while fetching topic metadata'"
```

- In this section you learned how to monitor Job execution and engine health.

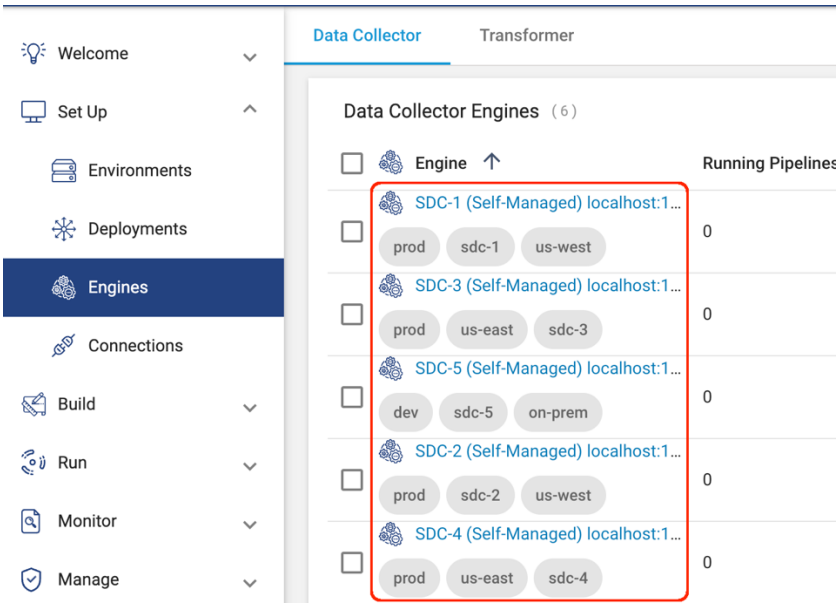
4.4 High availability/Disaster recovery

[*There are no user exercises for this section, it is only for your information*].

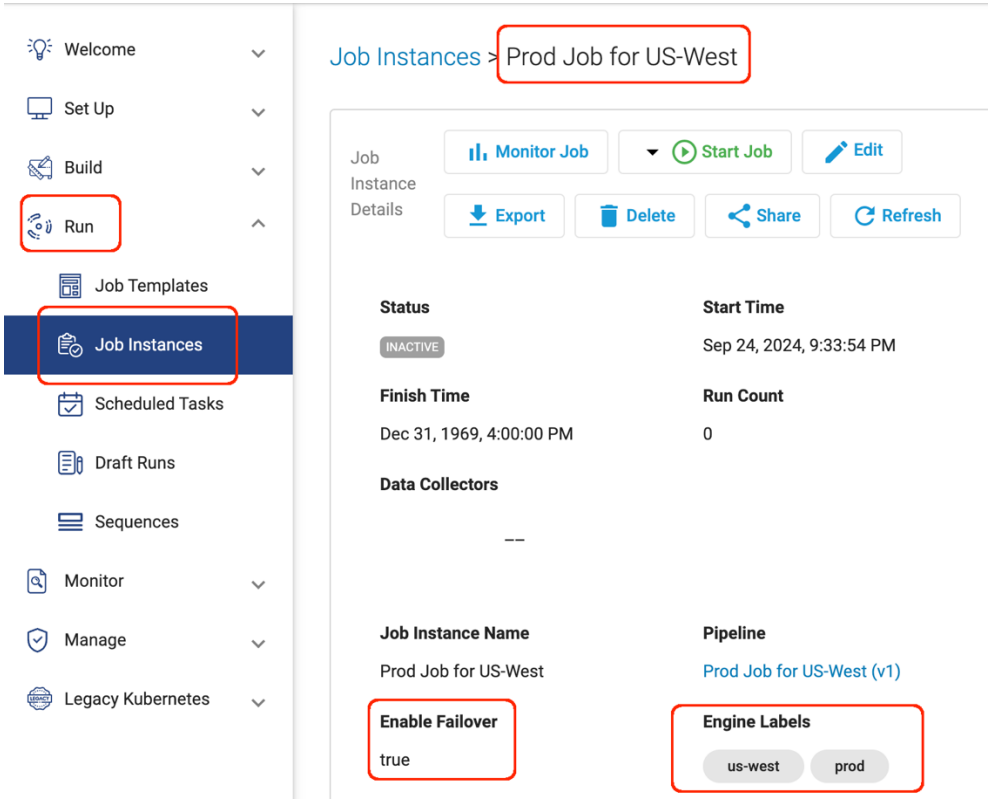
IBM StreamSets provides mechanisms for both High Availability and Disaster Recovery.

- **High Availability (HA):** The basis for IBM StreamSets HA is to have two or more engines deployed within a given region or network with the same capabilities and the same engine labels. The engines are shared-nothing.

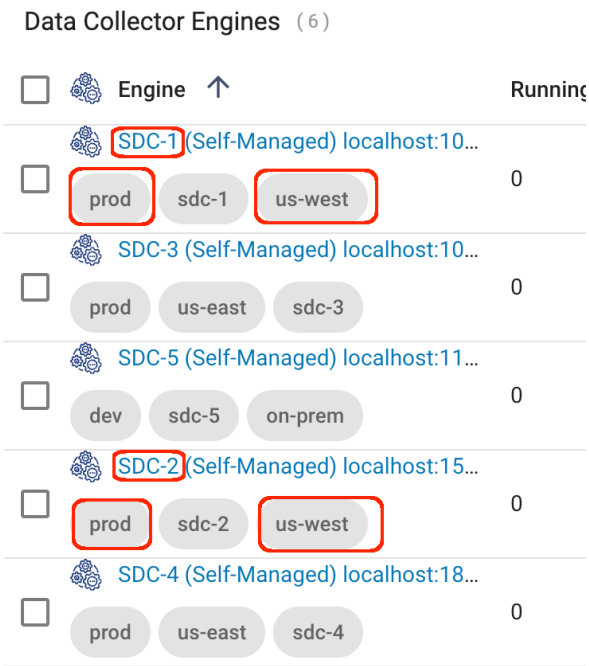
  For example, consider an enterprise that wants IBM StreamSets HA in their us-west region for production jobs. They may have deployed five engines, each one labeled to indicate its environment (prod or dev), its region or location (us-west, us-east, on-prem) and a unique identifier (sdc-1, sdc-2, etc..). Here is what the engines and their associated labels might look like:



- To run a production Job with HA in the us-west region, the Job should be enabled for failover, and given the engine labels `prod` and `us-west`, like this:

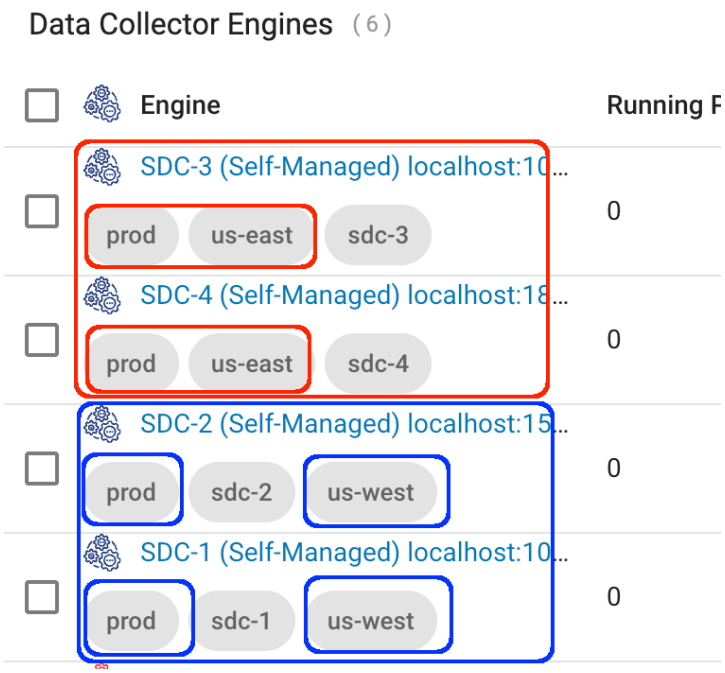


- Note that there are two engines that support both of those labels:



- When the Job is started, Control Hub will select one of the two engines that matches the Job's engine labels and will instruct that engine to start the Job.
- If the Job or the selected engine fails, Control Hub will detect that condition and resume the Job on the other engine that has both of the same engine labels. The job will be resumed from its last committed offset to avoid duplicate processing when it resumes on the new engine.
- In summary, given that there is more than one engine with the labels `prod` and `us-west`, Jobs with the engine labels `prod` and `us-west` are able to take advantage of HA. In the case of a failure, Jobs will failover to another engine.
- Note that in this configuration, Jobs with the labels `prod` and `us-west` will not failover to engines running in the dev environment, nor to engines running in the us-east region.

- Disaster Recovery (DR): In a DR scenario, an enterprise may need to move operations from one datacenter or region to another. There must be sufficient resources to run the workload in either of the datacenters or regions. In the example above, one can see HA pairs of engines deployed in the us-west region and in the us-east region:



- Once again, Jobs that are assigned region labels will failover within their region but not across regions.
- In a DR scenario, Jobs and/or engines can have their labels updated and Control Hub can "synchronize" the Jobs, meaning the Jobs will stop running on engines that no longer have corresponding labels, and will resume their processing on engines with matching labels. This operation can be manual or completely automated using the IBM StreamSets SDK. The result is that operators have the ability to activate a DR datacenter or region, and to have IBM StreamSets coordinate the process of moving Jobs over to the engines in the desired DR datacenter or region.

- In this section you learned how IBM StreamSets provides High Availability and Disaster Recovery capabilities.

5 Workflow and Orchestration

[*There are no user exercises for this section, it is only for your information*].

There are four main capabilities within IBM StreamSets for defining workflows and orchestrating multi-step integration processes:

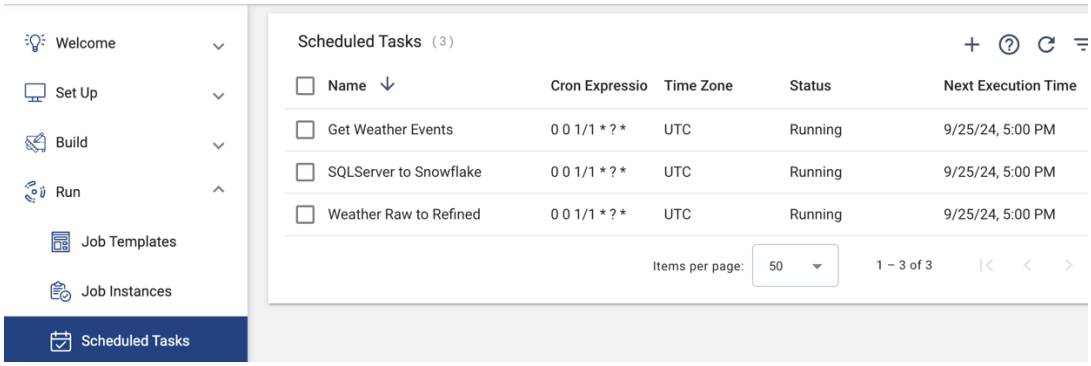
- Scheduling
- Sequences
- Orchestration Stages
- Programmatic Interfaces

Here is a brief overview of these options:

5.1 Scheduling

[*There are no user exercises for this section, it is only for your information*].

Scheduling – IBM StreamSets provides a built in scheduler that allows tasks to be defined, like starting, stopping, or upgrading a Job at regular intervals.



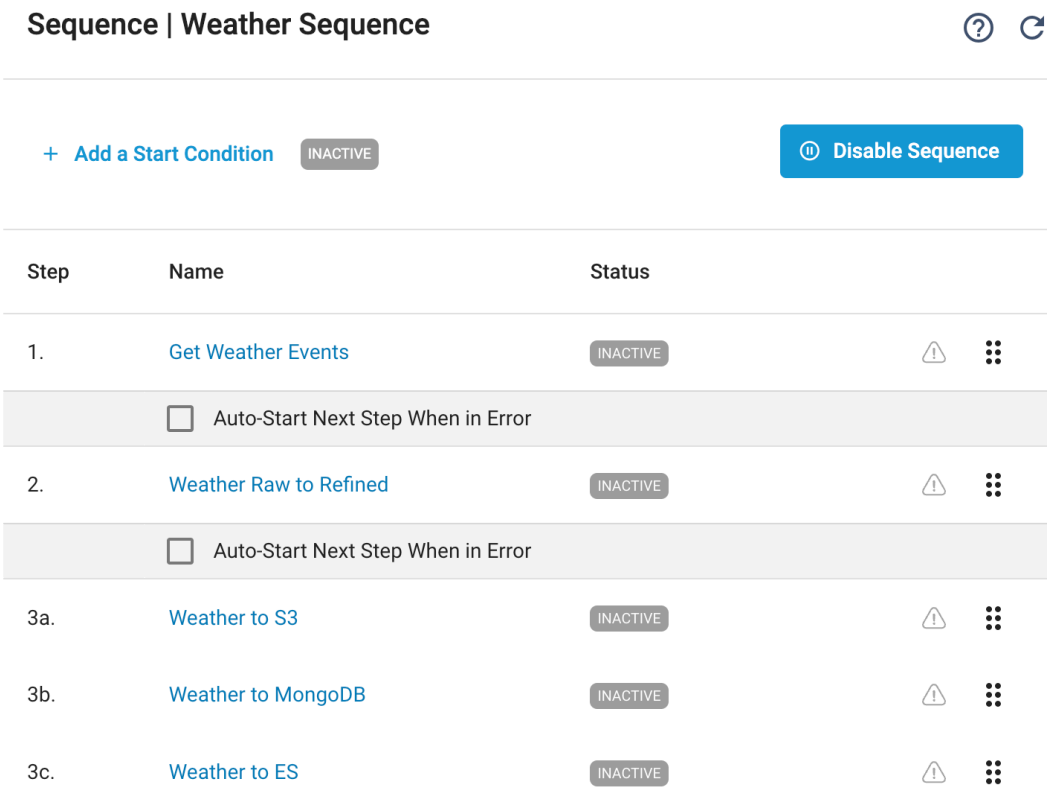
Tasks can be scheduled to repeat at user defined intervals using a GUI or cron-like syntax.

5.2 Sequences

[*There are no user exercises for this section, it is only for your information*].

- **Sequences** – IBM StreamSets provides a Job Sequencer that allows sets of Jobs to be defined with both serial and parallel execution, and with each step gated by the success or failure of previous steps. For

example, here is a three-step sequence that runs the first two Jobs serially, and the last three Jobs in parallel:



5.3 Orchestration Stages

[*There are no user exercises for this section, it is only for your information*].

- Orchestration Stages – Orchestration Stages allow more complex workflow logic to be implemented within "Orchestration" pipelines. An Orchestration pipeline has access to built-in "Start Job" and "Stop Job" processors and can call any Control Hub operation as needed. This allows dynamic behaviors within workflows, for example to start a particular Job based on any aspect of the data received. For example, if a Job that watches a particular directory picks up a JSON file, it might pass the file to a JSON-processing Job, whereas if it picks up a fixed-length file, it might pass the file to a different fixed-length file processing Job.

5.4 Programmatic Interfaces

[*There are no user exercises for this section, it is only for your information*].

- Programmatic Interfaces – IBM StreamSets provides a comprehensive set of REST APIs that allows workflow and orchestration to be controlled by third-party schedulers like Control-M or Tidal. IBM StreamSets also provides a complete Python-based SDK that allows custom Python scripts or tools like Airflow to control every aspect of the platform. See the next section for additional detail on the StreamSets Python SDK.

6 IBM StreamSets SDK for Python

[*There are no user exercises for this section, it is only for your information*].

6.1 Overview

- All of the discussion and instructions above have relied completely on interactions with IBM StreamSets' user interface. What may not be obvious is that every aspect of IBM StreamSets – deploying engines, building pipelines, creating, starting and stopping Jobs, monitoring Jobs, gathering metrics, etc…– can be performed programmatically using a high level Python SDK.
- IBM StreamSets' Python SDK is a native Python3 module you can install with pip. Complete instructions and documentation can be found at https://docs.streamsets.com/platform-sdk/latest/index.html

6.2 Code Example

- Here is a Python snippet that shows how to parameterize and start multiple instances of a Job Template:

```
# Connect to Control Hub
sch = ControlHub(
    credential_id=CRED_ID,
    token=CRED_TOKEN)

# Job Template to start instances for
JOB_ID= '<your-job-template-id>'

# Job Template Instances Runtime Parameters
RUNTIME_PARAMETERS = ['<your parmaters>']

# Get the Job Template
job_template = sch.jobs.get(job_id = JOB_ID)

# Create and start the Job Template Instances
job_template_instances = sch.start_job_template(job_template,
                             instance_name_suffix='PARAM_VALUE',
                             parameter_name='PARAM_1',
                             runtime_parameters=RUNTIME_PARAMETERS,
                             attach_to_template=True,
                             delete_after_completion=False)
```

7 Getting help and troubleshooting

This section provides information about getting help with your lab and some common troubleshooting topics.

7.1 Common troubleshooting tips

- See the docs here for IBM StreamSets troubleshooting tips

7.2 Getting help

Here are links to StreamSets resources:

IBM StreamSets on Seismic

IBM StreamSets Slack Channels:

- #ibmstreamsets_sales
- #ibmstreamsets_users

IBM StreamSets Platform Documentation

IBM StreamSets Source Connector Documentation

IBM StreamSets Target Connector Documentation

IBM StreamSets Processors Documentation