

1783 –

# Day 2 Operations for IBM StreamSets

Hands-on lab guide

Mark Brooks – Solution Architect, IBM StreamSets

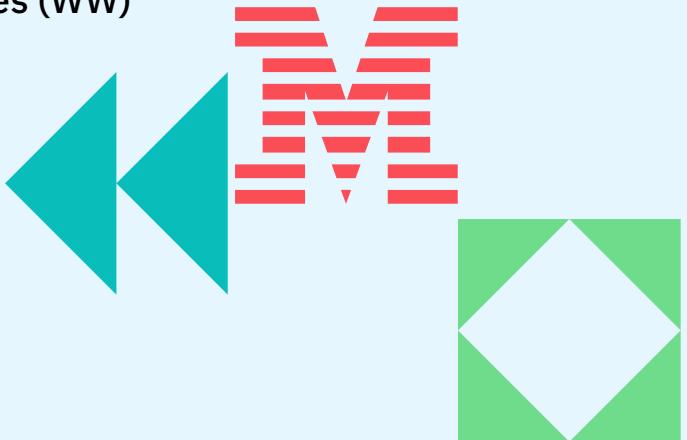
[mark.brooks@ibm.com](mailto:mark.brooks@ibm.com)

Deepak Rangarao – Distinguished Engineer & WW CTO  
Technical Sales - Cloud Paks Global Sales (WW)

[drangar@us.ibm.com](mailto:drangar@us.ibm.com)

Daniel Hancock – Principal WW Data & AI Technical  
Specialist - Global Sales (WW)

[daniel.hancock@us.ibm.com](mailto:daniel.hancock@us.ibm.com)



# Table of Contents

<b>1 Introduction</b>	<b>6</b>
1.1 About this hands-on lab	7
1.1.1 Why Day 2 Operations?	7
1.1.2 What We Will Cover	7
1.1.3 Objectives	8
<b>2 Hands on lab setup</b>	<b>8</b>
2.1 Create user account with IBM StreamSets	8
2.2 Register data plane with control plane	9
2.3 Clone a StreamSets Deployment	9
2.4 Set Permissions on your StreamSets Deployment	12
2.5 Start the Deployment	13
2.6 Generate an Engine Deployment Script	14
2.7 Run the Engine Deployment Script on the Data Plane VM	15
2.8 Confirm the engine is healthy	17
<b>3 Designing pipelines for reuse</b>	<b>17</b>
3.1 Reusable connections	17
3.2 Reusable fragments	18
3.3 Parameterization	26
<b>4 Operationalizing pipelines</b>	<b>28</b>
4.1 Jobs	28
4.2 Job templates	34
4.3 Monitoring	43
4.4 High availability/Disaster recovery	49

# IBM TechXchange

<b>5 Workflow and Orchestration</b>	<b>52</b>
5.1 Scheduling	52
5.2 Sequences	52
5.3 Orchestration Stages	53
5.4 Programmatic Interfaces	53
<b>6 IBM StreamSets SDK for Python</b>	<b>54</b>
6.1 Overview	54
6.2 Code Example	54
<b>7 Getting help and troubleshooting</b>	<b>54</b>
7.1 Common troubleshooting tips	54
7.2 Getting help	54

## Notices and disclaimers

© 2024 International Business Machines Corporation. No part of this document may be reproduced or transmitted in any form without written permission from IBM.

**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.

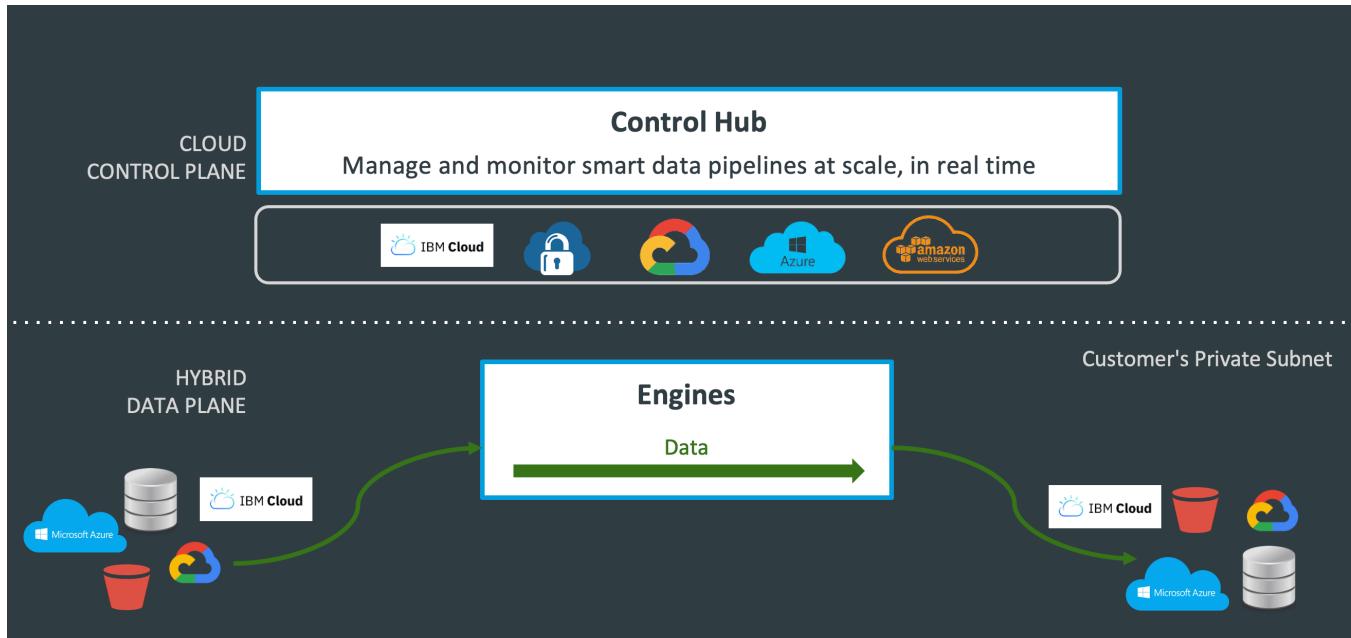
IBM, the IBM logo, and ibm.com are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark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

## IBM TechXchange

[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.

### 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

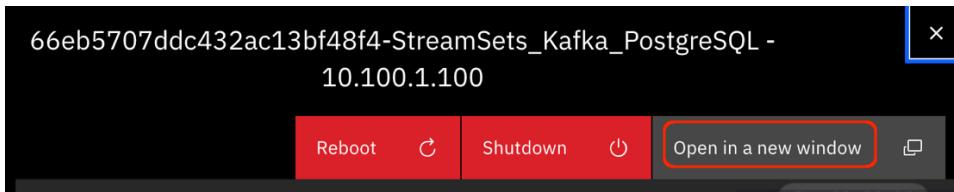
You will be provided with access to a VM; for the purpose of this hands on lab we will use the VM as the host for running the IBM StreamSets “data plane” (aka Engine). The steps below show to install a StreamSets engine and to register the data plane with the control plane.

## 2.3 Clone a StreamSets Deployment

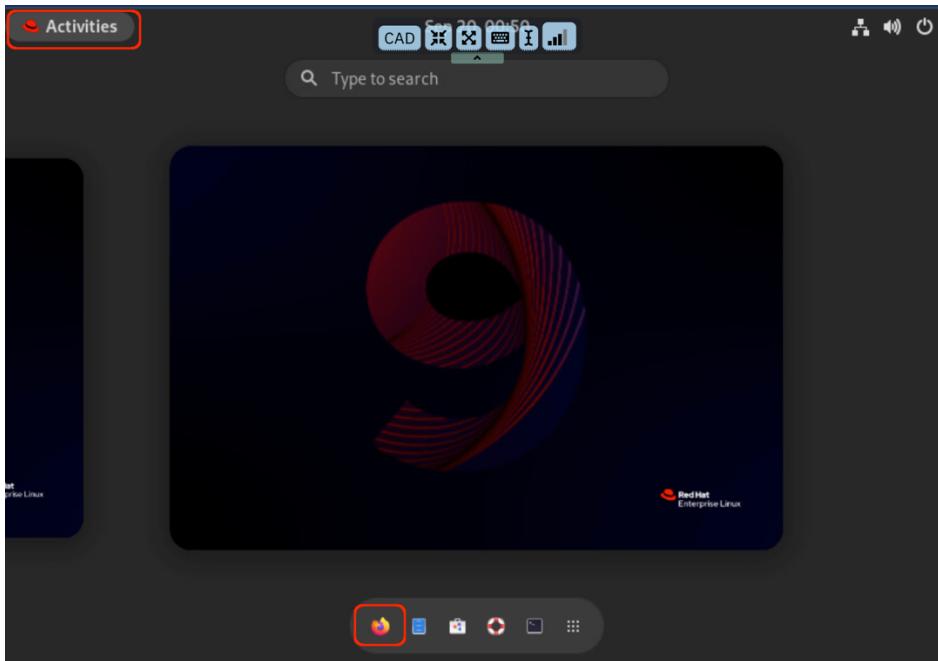
- Click the Console button in your TechZone Reservation page:

Name	OS	IP	Status
66eb5707ddc432ac13bf48f4-StreamSets_Kafka_PostgreSQL	Red Hat Enterprise Linux 9 (64-bit)	10.100.1.100	Running

- 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:

A screenshot of the IBM StreamSets Control Plane interface. The left sidebar has a navigation menu with items: Welcome, Set Up, Environments, Deployments (which is highlighted with a red box), Engines, and Connections. The main content area is titled "Deployments (1 with current filters)". It shows a table with one row:

Deployment Name	Deployment Type	Tags	Last Modified On	State
Hands On Lab	Self-Managed	-	10 minutes ago	Active

The "Hands On Lab" row is also highlighted with a red box. To the right of the table, there are several buttons: Edit, Clone (which is highlighted with a red box), Delete, and Share. Above the table, there are filter options for Deployment Name, Deployment Type, Tags, Last Modified On, and State. The "Deployment Name" filter is set to "Hands On Lab".

- 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:

## Clone Deployment

### 1 Configure Deployment

Configure the name, tags, engine labels, and engine version of the cloned deployment. The cloned deployment retains all other property values from the original deployment. You can check and edit the configuration after saving. [Learn more](#)

Deployment Name ⓘ  
mark\_brooks\_deployment

Deployment Tags ⓘ  
Add New...

Engine Labels ⓘ  
mark\_brooks × Add New...

Engine Version: ⓘ  
5.11.0

Save

→ Click Save

The next dialog looks like this:

## Clone Deployment

### 1 Configure Deployment

### 2 Review & Launch

Successfully created the deployment 'mark\_brooks\_deployment.' Start the deployment to generate an install script to launch Data Collector.



→ 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:

The screenshot shows the IBM StreamSets Platform Control Hub interface. On the left, there's a sidebar with options like Welcome, Set Up, Environments, Deployments (which is highlighted with a red box), Engines, and Connections. The main area is titled 'Deployments 1 item selected' and lists three items: 'StreamSets Free Trial Deploy...' (Kubernetes, Active), 'mark\_brooks\_deployment' (Self-Managed, Deactivated), and 'Hands On Lab' (Self-Managed, Active). The 'mark\_brooks\_deployment' row is selected and highlighted with a red box. In the top right of the list view, there's a sharing icon (a person icon with a share arrow) which is also highlighted with a red box.

- 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:

The screenshot shows the 'Sharing Settings' dialog box. At the top, it says 'Sharing Settings'. Below that is a search bar labeled 'Select Users and Groups' with an 'Add' button next to it. The main area is a table with columns: 'User / Group', 'Read', 'Write', and 'Execute'. There is one entry: 'mark.brooks@ibm.com (Owner)' with all three checkboxes checked. At the bottom are 'Cancel' and 'Save' buttons.

User / Group	Read	Write	Execute
mark.brooks@ibm.com (Owner)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

→ Click Save.

## 2.5 Start the Deployment

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

The screenshot shows the 'Deployments' section of the IBM TechXchange interface. On the left is a navigation sidebar with options like Welcome, Set Up, Environments, Deployments (which is selected and highlighted in blue), Engines, Connections, Build, Run, and Monitor. The main area displays a table titled 'Deployments 1 item selected'. The table has columns: Deployment Name, Deployment Type, Tags, Last Modified On, and State. There are three items listed: 'StreamSets Free Trial Deploy...' (Kubernetes, Active), 'mark\_brooks\_deployment' (Self-Managed, Deactivated), and 'Hands On Lab' (Self-Managed, Active). The 'mark\_brooks\_deployment' row is selected, indicated by a checked checkbox and a red box around the row. To the right of the table is a toolbar with icons for Edit, Clone, Delete, Share, and Start. The 'Start' icon is highlighted with a red box.

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

This screenshot shows the same 'Deployments' list after the deployment has been started. The 'mark\_brooks\_deployment' row is now fully visible, showing it was started 27 minutes ago and is currently in an 'Active' state, indicated by a green button. The rest of the table and the toolbar remain the same as in the previous screenshot.

## 2.6 Generate an Engine Deployment Script

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

The screenshot shows the IBM StreamSets interface. On the left, there's a sidebar with various navigation options like Welcome, Set Up, Environments, Deployments (which is selected and highlighted with a red box), Engines, Connections, Build, Run, Monitor, and Manage. The main area is titled 'Deployments 1 item selected' and lists three items: 'StreamSets Free Trial Deploym...', 'mark\_brooks\_deployment' (which is checked and highlighted with a red box), and 'Hands On Lab'. To the right of the list is a context menu with options: Edit, Clone, Delete, Share, Start, Stop, and 'Get Install Script' (which is highlighted with a red box). Below the list, there are filters for 'Deployment Name', 'Deployment Type', 'Tags', 'Last Modified On', and 'State', along with a 'Items per page' dropdown set to 10.

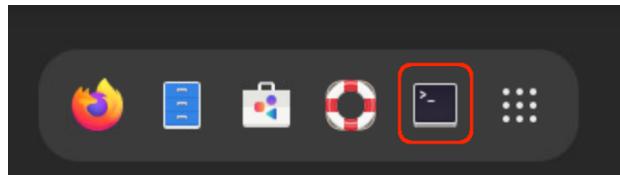
- In the dialog that opens, select Run engine in background, Download and Install from Script, and then click the Copy icon:

The screenshot shows the 'Install Engine Script' dialog. It has two radio button options: 'Run engine in foreground' and 'Run engine in background' (the latter is selected and highlighted with a red box). Below the radio buttons are two buttons: 'Run Docker Image' and 'Download and Install from Script' (the latter is highlighted with a red box). A large text area contains a bash command. At the top right of the text area is a copy icon (highlighted with a red box). At the bottom right is a 'Close' button.

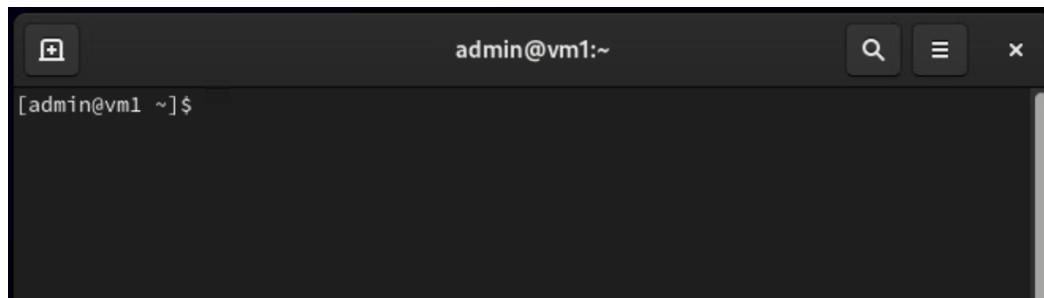
```
bash -c "$(curl -fsSL https://na01.hub.streamsets.com/streamsets-engine-install.sh)" -- --
deployment-id=384d3ecf-6672-4536-af31-46157679b3d7:08e9a095-3f35-11ef-9573-37da5dacf3af --
deployment-
token=eyJ0eXAiOiJKV1QiLCJhbGciOiJub25lIn0.eyJzIjoiYWZkNmQ5YTU1YmZhZmUyMWI3ZjM0NjJ1YmI4ZGMzNDJj
NTczODExMDFlZmE50dg1jh0WV10dh1y1zM3MDY1ZWewMWVknGNhMTFLY2MxY2RmMmIyNzBnNmFmNGNLYTU2ZGjhN2E5YT
k1MDY40TE4Yzc20DQ5MDViMzRhMDA5NTYiLCj2IjoxLCJpc3Mi0iJuYTAXIiwanRpIjoiMTBjZmE0Y2MtMDQyNC00MjA0
LWEwNDMthjhLOGw0Dg4NTUxIiwiIbyI6IjA4ZTlhMDk1LTNmMzUtMTFlZi05NTczLTM3ZGE1ZGFjZjNhZj9. --sch-
url=https://na01.hub.streamsets.com --background
```

## 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:

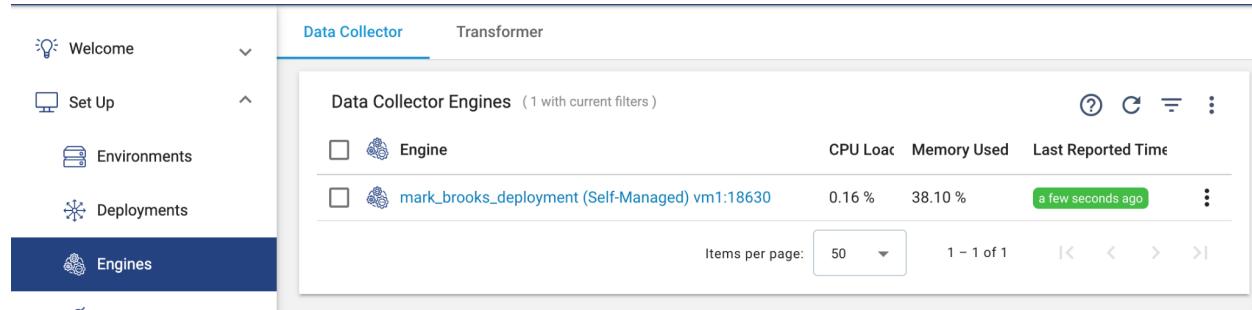
```
[admin@vm1 streamsets]$ bash -c "$(curl -fsSL https://na01.hub.streamsets.com/streamsets-engine-install.sh)" -- --deployment-id=384d3ecf-6672-4536-af31-46157679b3d7:08e9a095-3f35-11ef-9573-37da5acf3af --deployment-token=eyJ0eXAiOiJKV1QiLCJhbGciOiJub25lIn0.eyJzIjoiYWZkNmQ5YTU1YmZhZmUyMWI3ZjM0NjJlYmI4ZGMzNDJiNTczODExMDFlZmE50Dg1Yjh0WVi0DhiYzM3MDY1ZWEmMWVkJGNhMTFLY2MxY2RmMmIyNzBmNmFmNGNlYTU2ZGJhN2E5YTk1MDY40TE4Yzc20DQ5MDViMzRhMDA5NTYiLCJ2IjoxLCJpc3Mi0iJuYTAXIiwianRpIjoiMTBjZmE0Y2MtMDQyNC00MjA0LWEwNDMtNjh1OGQw0Dg4NTUxIiwibyI6IjA4ZTlhMDk1LTNmMzUtMTFlZi05NTczLTM3ZGE1ZGFjZjNhZiJ9. --sch-url=https://na01.hub.streamsets.com --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 pipelines
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:



The screenshot shows the StreamSets Platform interface. On the left, there's a sidebar with icons for Welcome, Set Up, Environments, Deployments, and Engines. The Engines icon is highlighted with a blue bar. The main area is titled "Data Collector" and "Transformer". Under "Data Collector", it says "Data Collector Engines (1 with current filters)". There's a table with one row:

Engine	CPU Load	Memory Used	Last Reported Time
mark_brooks_deployment (Self-Managed) vm1:18630	0.16 %	38.10 %	a few seconds ago

At the bottom, there are buttons for "Items per page: 50", "1 - 1 of 1", and navigation arrows.

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

## 3 Designing pipelines for reuse

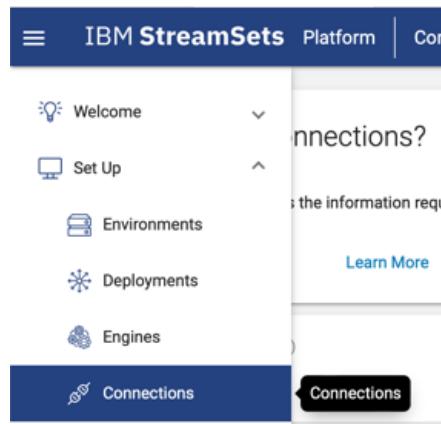
### 3.1 Reusable connections

[*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



The screenshot shows the StreamSets Platform interface. On the left, there's a sidebar with icons for Welcome, Set Up, Environments, Deployments, and Engines. The Connections icon is highlighted with a blue bar. The main area has a heading "What are connections?" and a "Learn More" button.

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

The screenshot shows the IBM StreamSets interface with a sidebar on the left containing 'Welcome', 'Set Up', 'Environments', 'Deployments', 'Engines', and 'Connections'. The 'Connections' item is selected and highlighted with a dark blue background. The main area displays a table titled 'Connections (4)' with four entries:

<input type="checkbox"/>	Name	Type	Tags
<input type="checkbox"/>	Risk Scoring Microservice	WebClient	-
<input type="checkbox"/>	Kafka	Kafka	-
<input type="checkbox"/>	SingleStore	JDBC	-
<input type="checkbox"/>	Postgres	JDBC	-

Items per page: 10 1 - 4 of 4

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

Sharing Settings			
User / Group	Read	Write	
admins	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
hol	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
mark.brooks@ibm.com (Owner)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

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

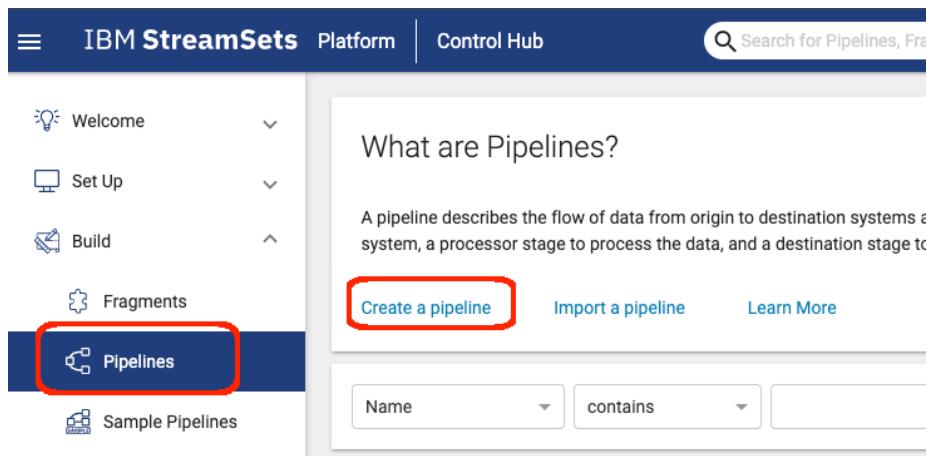
A fragment is a stage or set of connected stages that you can reuse in pipelines. Use fragments to easily add the same processing logic to multiple pipelines and to ensure that the logic is used as designed. You can create fragments from scratch or by selecting a subset of the stages in an existing data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

## New Pipeline

### 1 Define Pipeline

Define the pipeline name, the type of engine for the pipeline, and whether to start with a blank canvas or with a sample pipeline. [Learn more](#)

Name:

Employee Report

?

Description:

?

Engine Type:

- Data Collector** - Runs data ingestion pipelines that perform record-based data transformations in streaming, CDC, or batch modes
- Transformer** - Runs data processing pipelines on Spark that perform set-based transformations such as joins, aggregates, and sorts on the entire data set
- Transformer for Snowflake** - Runs data processing pipelines on Snowflake

Start with:

- Blank Pipeline**
- Sample Pipeline**

Cancel

Next

- 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:

## New Pipeline

### 1 Define Pipeline

### 2 Configure Pipeline

If starting with a sample pipeline, select the sample to use. Select the authoring engine to use for pipeline design. The engine with the most recent reported time is selected by default. [Learn more](#)

Authoring Engine:

mark\_brooks\_deployment (Self-Managed) - marks-mbp-2.usca.ibm.com:18630

?

[Click here to select](#)

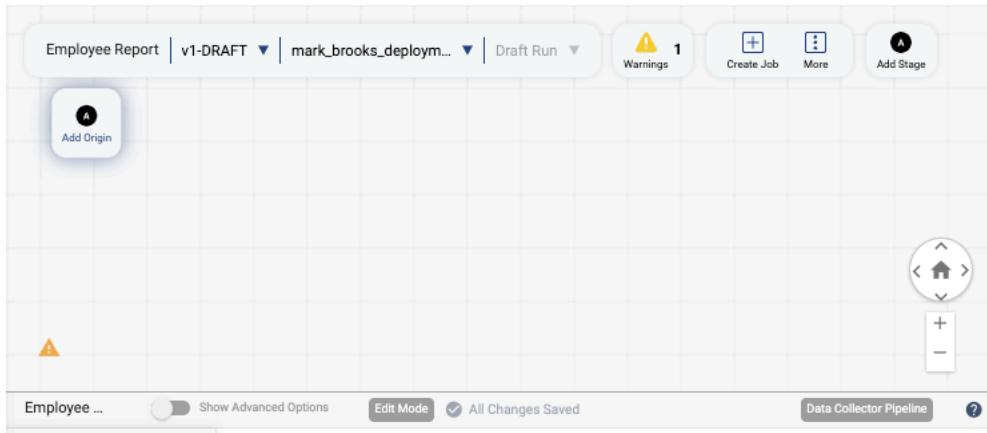
Back

Save & Next

Save & Open in Canvas

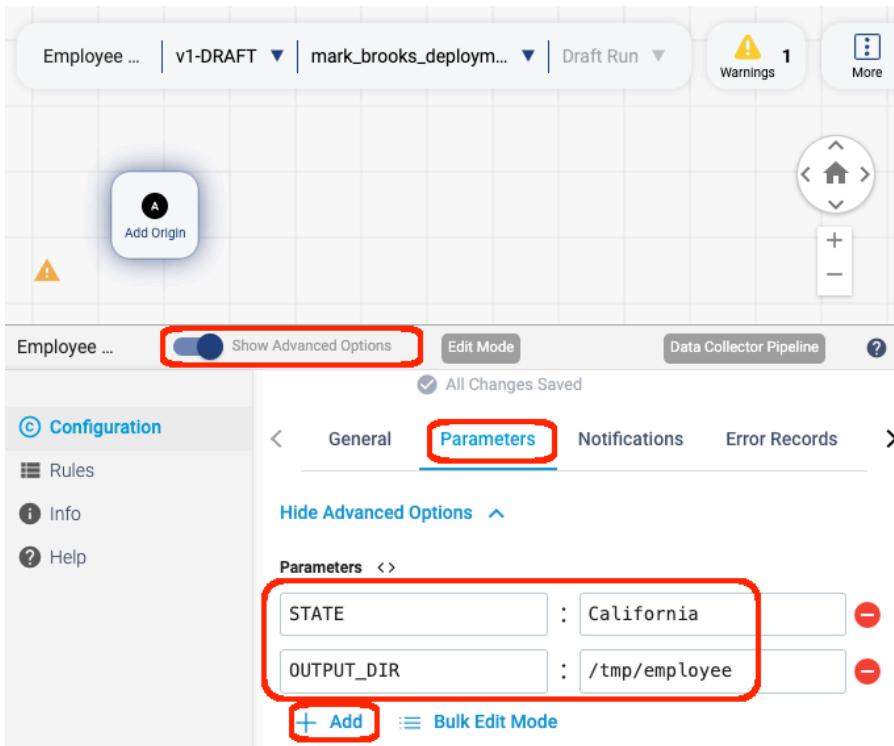
### 3 Share Pipeline

- 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:

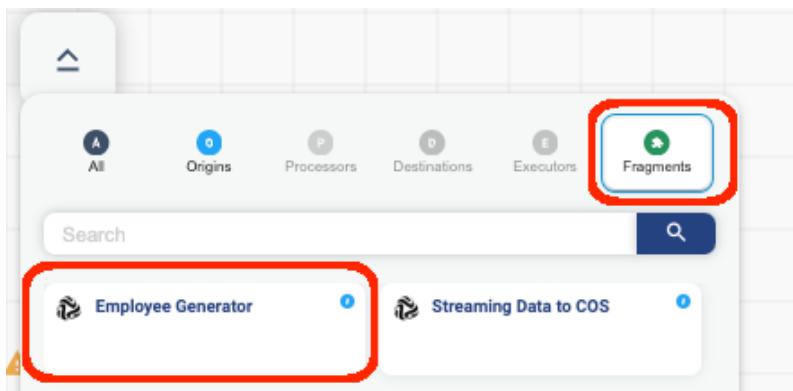


Note that the parameter values can be overridden at runtime by the user.

- Add an Employee Data Generator

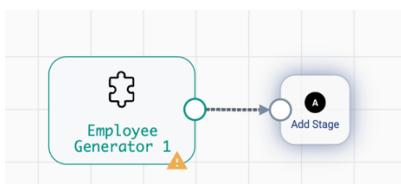
You will use a preconfigured fragment that will generate employee data to simulate reading from a data source.

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:



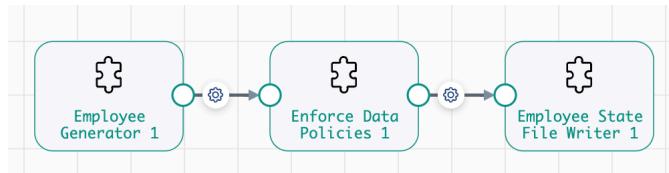
- Add a fragment to enforce data policies. All enterprises have a set of standard data policies such as timestamping records and masking PII. You will add a fragment that is preconfigured to perform those actions.

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:

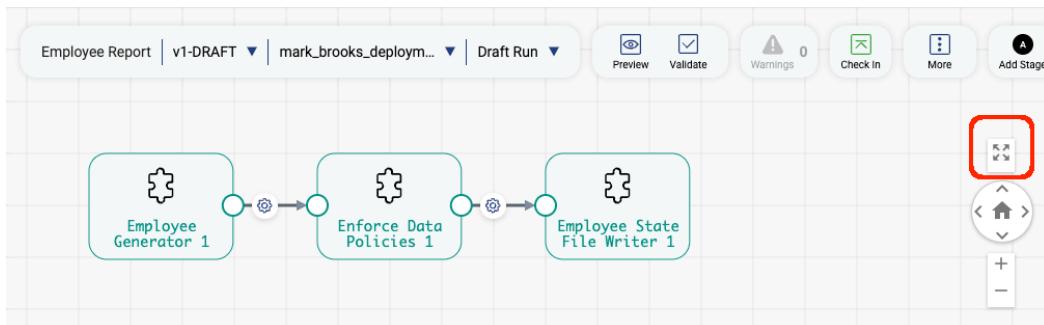


- Finally, you will add a fragment to filter employee records by state and to write the records to a user-defined location.

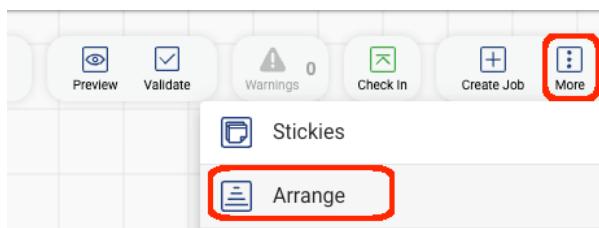
Click the Add Stage widget, add the fragment named "Employee State File Writer" and click Done when prompted for a Fragment Parameter. Your pipeline should now look like this:



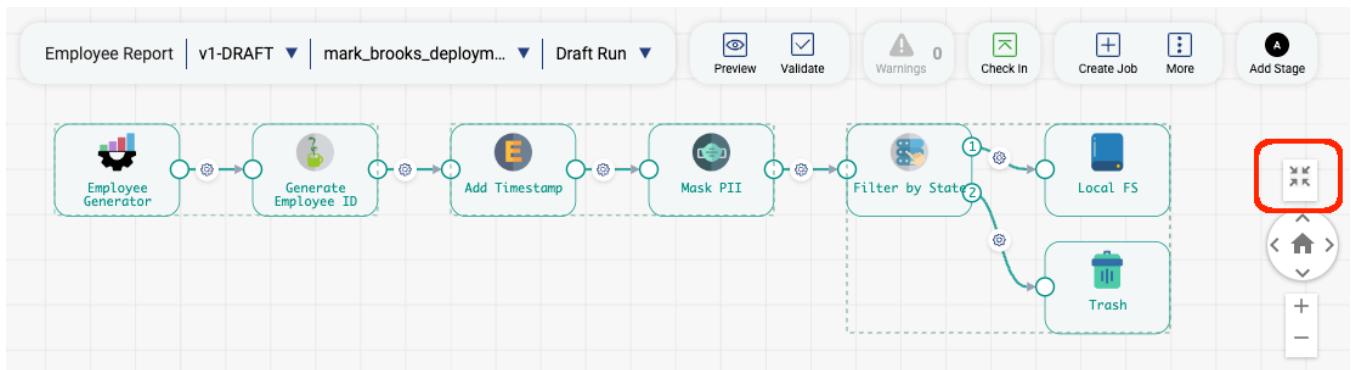
- To understand what's going on within each fragment, click the pipeline's "Expand" button:



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:

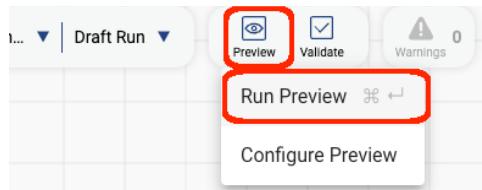


This will reveal the stage libraries contained within each fragment:

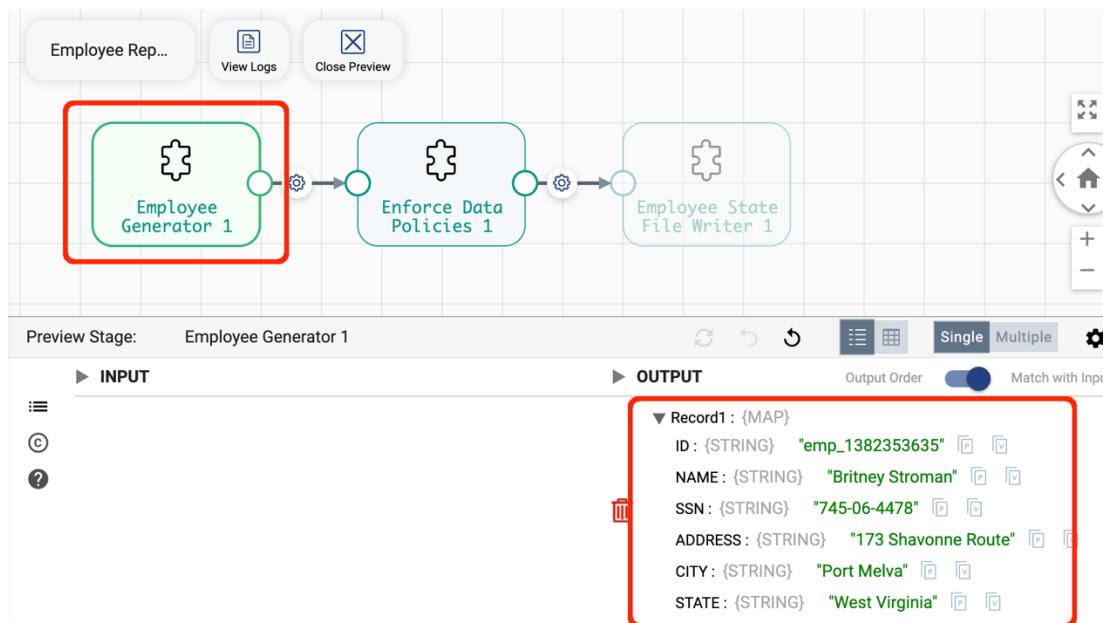


Click the collapse button (circled above) to restore the original view of the pipeline.

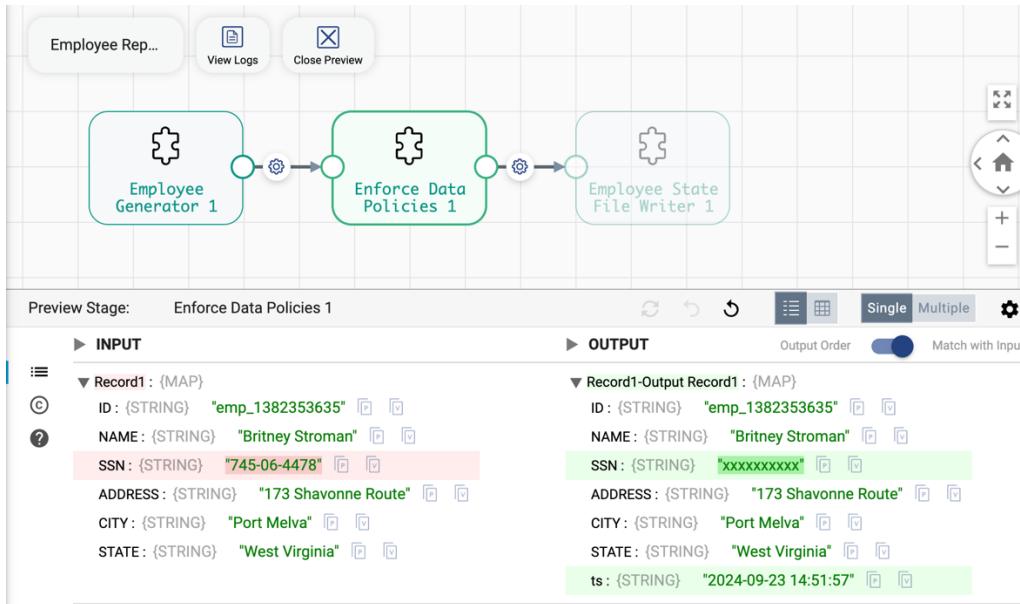
- Preview the pipeline! Pipeline preview allows pipeline developers to inspect how data moves through a pipeline at design time and by default will not write any data to downstream targets. Click the Preview > Run Preview item:



You will see sample records generated by the Employee Generator:

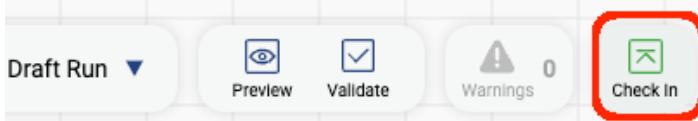


Click on the Enforce Data Policies stage to see the new timestamp field and the masked SSN field:



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:

## Check In

1 Publish Pipeline

Pipeline Name	Employee Report
Commit Message	New Pipeline
<input type="button" value="Cancel"/> <input type="button" value="Publish and Next"/> <input type="button" value="Publish and Close"/>	

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

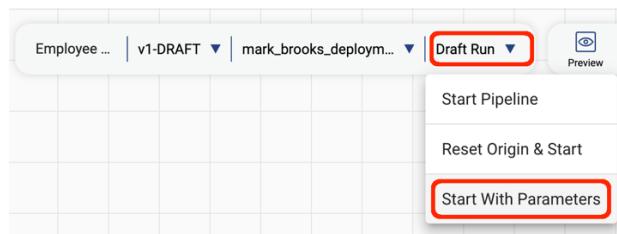
## 3.3 Parameterization

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:

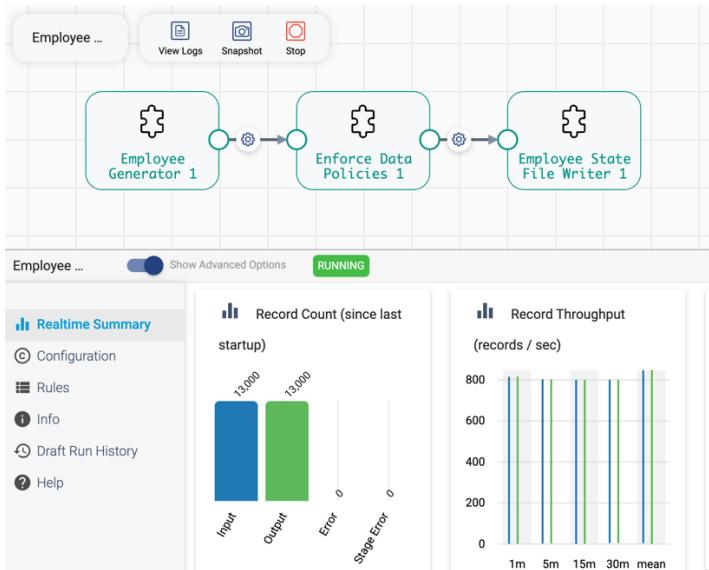


- Specify a different state, like "Texas" (without quotes) and keep the default output directory value:

The dialog box has two input fields: 'STATE' containing 'Texas' and 'OUTPUT\_DIR' containing '/tmp/employee'. At the bottom right are 'Cancel' and 'Start' buttons, with 'Start' highlighted by a red box.

- 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):

#### Published services

```
Local Kafka, you can connect using any client tool to useast.services.cloud.techzone.ibm.com:43026
Local PostgreSQL, you can connect using any SQL client tool to useast.services.cloud.techzone.ibm.com:22513
ssh admin@useast.services.cloud.techzone.ibm.com -p 41355
VM credentials - admin/IBMDemos
```

- 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": "xxxxxxxxxxxx", "ADDRESS": "036 DuBuque Field", "CITY": "Ashlibury", "STATE": "Texas", "ts": "2024-09-23 15:42:37"}
{"ID": "emp_1497265364", "NAME": "Jonathon Gutmann", "SSN": "xxxxxxxxxxxx", "ADDRESS": "16372 Cremin Square", "CITY": "Tarshaville", "STATE": "Texas", "ts": "2024-09-23 15:42:37"}
{"ID": "emp_1736121568", "NAME": "Virgil Zemlak", "SSN": "xxxxxxxxxxxx", "ADDRESS": "99010 Graham Plaza", "CITY": "West Shannanview", "STATE": "Texas", "ts": "2024-09-23 15:42:37"}
```

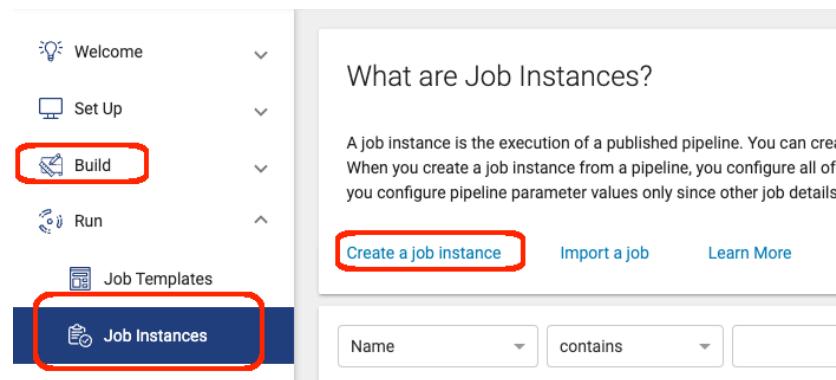
- This confirms you are able to run pipelines using different parameter values.

## 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

## Create Job Instances

1 Define Job

Define job details. [Learn more](#)

Name:

Description:

Job Tags:

Start with:

Create from pipeline

Create from job template

[Cancel](#) [Next](#)

- 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:

## Create Job Instances

1 Define Job

2 Select Pipeline

Select the published pipeline that you want to run. [Learn more](#)

Pipeline:  [Click here to select](#)

Pipeline Version:  [Click here to select](#)

[Back](#) [Next](#)

- 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.

## Create Job Instances

- 1 Define Job
- 2 Select Pipeline
- 3 Configure Job

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: mark-brooks-deployment (Self-Managed)

Engine Labels:

- mark-brooks-deployment
- mark-brooks

Enable Failover:

Show Advanced Options ▾

Back Save & Next Save & Exit

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

## Create Job Instances

- 1 Define Job
- 2 Select Pipeline
- 3 Configure Job
- 4 Define Runtime Parameters

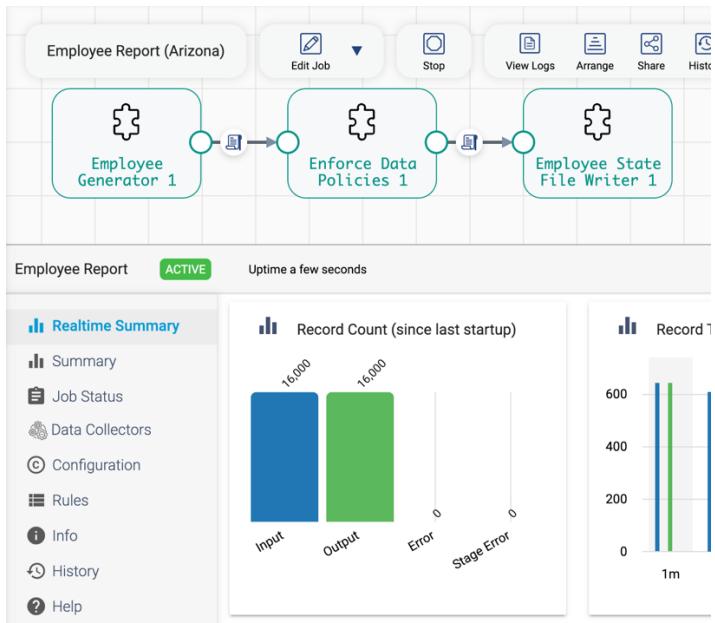
Define the parameter values to start the pipeline with. Override the default values using simple or bulk edit mode. In bulk edit mode, configure parameter values in JSON format. [Learn more](#)

STATE	:	Arizona
OUTPUT_DIR	:	/tmp/employee

[Bulk Edit Mode](#)

Back Save & Next Save & Exit

- 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:

What are Job Instances?

A job instance is the execution of a published pipeline. You can create a job instance from a published pipeline template. When you create a job instance from a pipeline, you configure all of the job details. When you create a job instance from a job template, you configure pipeline parameter values only since other job details are already defined.

Create a job instance Import a job Learn More

Name	contains	Search

Job Instances 1 item selected

Name	Pipeline	Version	Last Modified On	Job Status
<input checked="" type="checkbox"/> Employee Report (Arizona)	Employ...	v1	8 minutes ago	INACTIVE

Items per page: 50 1 - 1 of 1

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

### Duplicate Job Instance

Name:	<input type="text" value="Employee Report (Utah)"/>	(?)
Description:	<input type="text"/>	(?)
Number Of Copies:	<input type="text" value="1"/>	(?)

Cancel
Duplicate

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

Name	Pipeline	Version	Last Modified On	Job Status
<input checked="" type="checkbox"/> Employee Report (Utah)	Employ...	v1	a few seconds ago	INACTIVE
<input type="checkbox"/> Employee Report (Arizona)	Employ...	v1	17 minutes ago	INACTIVE

- 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:

## Edit Job

- 1 Define Job
- 2 Select Pipeline
- 3 Configure Job
- 4 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](#)

STATE	:	Utah
OUTPUT_DIR	:	/tmp/employee

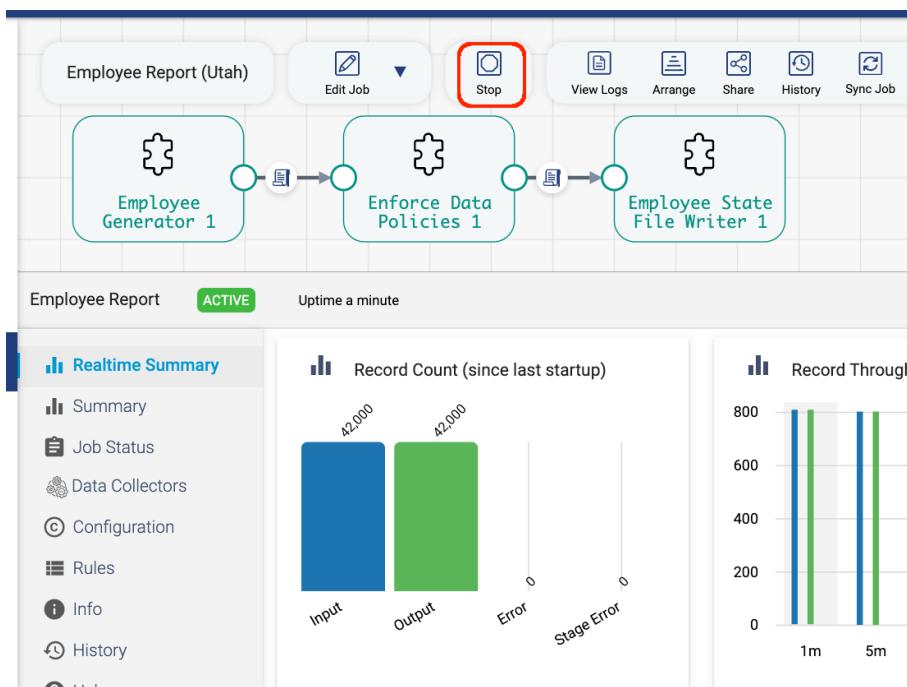
[Bulk Edit Mode](#)

Back

**Save & Next**

Save & Exit

- 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:

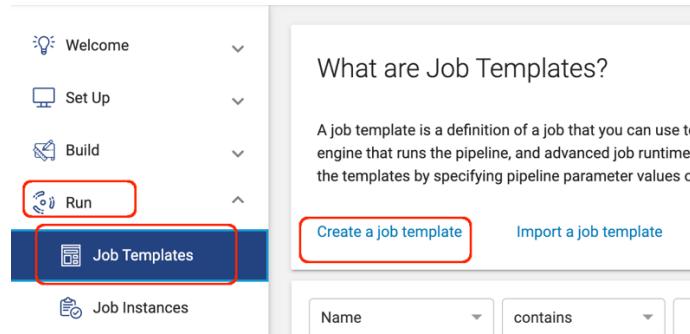
```
$ ls -l /tmp/employee/Utah  
-rw-r--r-- 1 mark wheel 478356 Sep 24 12:42 860a40d5-e36e-4f04-908e-c0f1d7072aee.json  
  
$ cat /tmp/employee/Utah/860a40d5-e36e-4f04-908e-c0f1d7072aee.json | head  
  
{ "ID": "emp_491023667", "NAME": "Lynda Pfeffer", "SSN": "xxxxxxxxxx", "ADDRESS": "5375  
Sheron Square", "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:

## New Job Template

### 1 Define Job Template

Define job template details. [Learn more](#)

Name:

Employee Report Template

?

Description:

?

Job Tags:

Add New...

?

Cancel

Next

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

## New Job Template

### 1 Define Job Template

### 2 Select Pipeline

Select the published pipeline that you want to run. [Learn more](#)

Pipeline:

Employee Report

[Click here to select](#)

?

Pipeline Version:

v1

[Click here to select](#)

?

Back

Next

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

## New Job Template

1 Define Job Template

2 Select Pipeline

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: mark-brooks-deployment (Self-Managed) ?

Engine Labels:

- mark-brooks-deployment x
- mark-brooks x Add New...

Enable Failover:  ?

Show Advanced Options v

Back Save & Next Save & Exit

- 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:

## New Job Template

1 Define Job Template

2 Select Pipeline

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 edit mode. In bulk edit mode, configure parameter values in JSON format. [Learn more](#)

Parameter Name	Default Value	Static Parameter
STATE	<input type="text" value="Enter Value"/>	<input type="checkbox"/>
OUTPUT_DIR	/tmp/employee	<input type="checkbox"/>
<a href="#">Bulk Edit Mode</a>		

Back Save & Next Save & Exit

- Click Exit:

## New Job Template

- 1 Define Job Template
- 2 Select Pipeline
- 3 Configure Job Details
- 4 Set Parameter Defaults
- 5 Review

Successfully created the job template 'Employee Report Template'.

**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:

The screenshot shows the IBM StreamSets interface. On the left, there's a sidebar with various options like Welcome, Set Up, Build, Run, Job Instances (which is selected and highlighted with a red box), Scheduled Tasks, Draft Runs, Sequences, Monitor, and Manage. The main area has a title 'What are Job Templates?' with a description. Below it, there are buttons for 'Create a job template', 'Import a job template', and 'Learn More'. A search bar is present. The main content area shows a table titled 'Job Templates' with one item selected: 'Employee Report Template'. The table columns are Name, Pipeline, Version, and Last Modified On. The 'Employee Report Template' row is highlighted with a red box. At the bottom right of the table, there's a 'Create Instances' button, which is also highlighted with a red box.

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

## Create Job Instances

1 Define Job

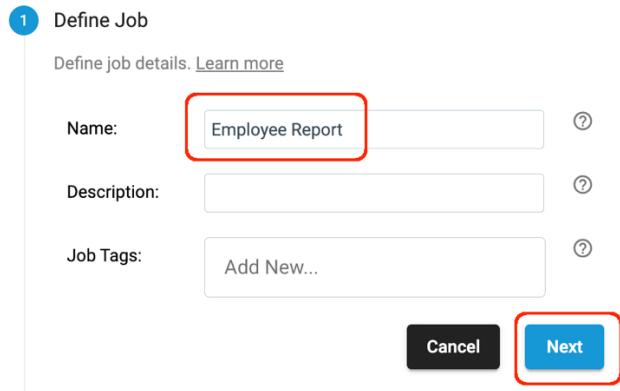
Define job details. [Learn more](#)

Name:  ?

Description:

Job Tags:  ?

Cancel Next



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

## Create Job Instances

1 Define Job

2 Select Job Template

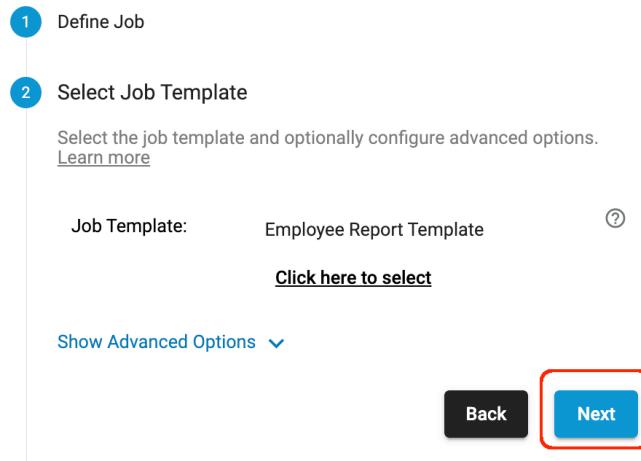
Select the job template and optionally configure advanced options. [Learn more](#)

Job Template: Employee Report Template ?

[Click here to select](#)

[Show Advanced Options](#) ▾

Back 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:

## Create Job Instances

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:

STATE

Runtime Parameters for Each

Instance:



Simple Edit

Bulk Edit

From File

1 Instance 1

STATE:

Maine



OUTPUT\_DIR:

/tmp/employee



+ ADD ANOTHER INSTANCE

- 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:

**Create Job Instances**

Show Guide X

- 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:	STATE	?

Runtime Parameters for Each Instance:

✓ Simple Edit Bulk Edit From File

1 Instance 1	STATE: Maine	DELETE
2 Instance 2	STATE: Oregon	DELETE
3 Instance 3	STATE: Iowa	DELETE

+ ADD ANOTHER INSTANCE

Back Create & Start (3)

- 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:

**Create Job Instances**

Show Guide X

- 3 Define Runtime Parameters
- 4 Review & Start

Successfully created & started 3 job instances.

Exit

- 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:

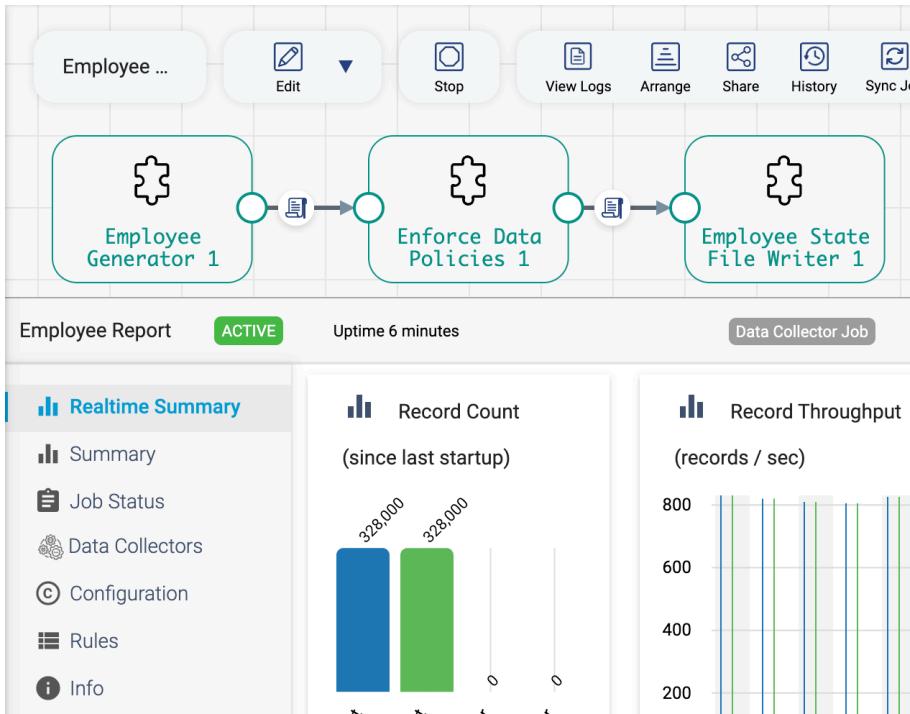
Name	Pipeline	Ver	Last Modified On	Job Status	Pipeline Stat
Employee Report - Maine	Emplo...	v1	3 minutes ago	ACTIVE	RUNNING
Employee Report - Oregon	Emplo...	v1	3 minutes ago	ACTIVE	RUNNING
Employee Report - Iowa	Emplo...	v1	3 minutes ago	ACTIVE	RUNNING

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

[Job Instances](#) > Employee Report - Maine

Status	Start Time
ACTIVE	Sep 24, 2024, 5:24:37 PM
Finish Time	Run Count
Dec 31, 1969, 4:00:00 PM	1
Data Collectors	
10.10.10.174:18630	RUNNING
	a minute ago

- 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:

The screenshot shows the "Job Instances" list. On the left, there is a navigation sidebar with "Run" and "Job Instances" highlighted with red boxes. The main area shows a table of "Job Instances 3 items selected". The table columns are Name, Pipeline, Ver, Last Modified On, Stop Jobs (button), Status, and Pipeline Status. Three rows are listed, each with a checkmark in the "Selected" column and a green "ACTIVE" status. The "Stop Jobs" button in the header is also highlighted with a red box.

- 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:

Job Instances ( 11 with current filters )							+
<input type="checkbox"/>		Name	Pipeline	Vers	Last Modified On	Job Status	Pipeline Status
<input type="checkbox"/>		a - 3	Employee...	v1	20 minutes ago	<span>ACTIVE</span>	<span>RUNNING</span>
<input type="checkbox"/>		a - 2	Employee...	v1	20 minutes ago	<span>INACTIVE_ERROR</span>	
<input type="checkbox"/>		a - 1	Employee...	v1	20 minutes ago	<span>ACTIVE</span>	<span>RUNNING</span>
<input type="checkbox"/>		Employee Report - Maine	Employee...	v1	32 minutes ago	<span>INACTIVE</span>	
<input type="checkbox"/>		Employee Report - Oregon	Employee...	v1	32 minutes ago	<span>INACTIVE</span>	

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.

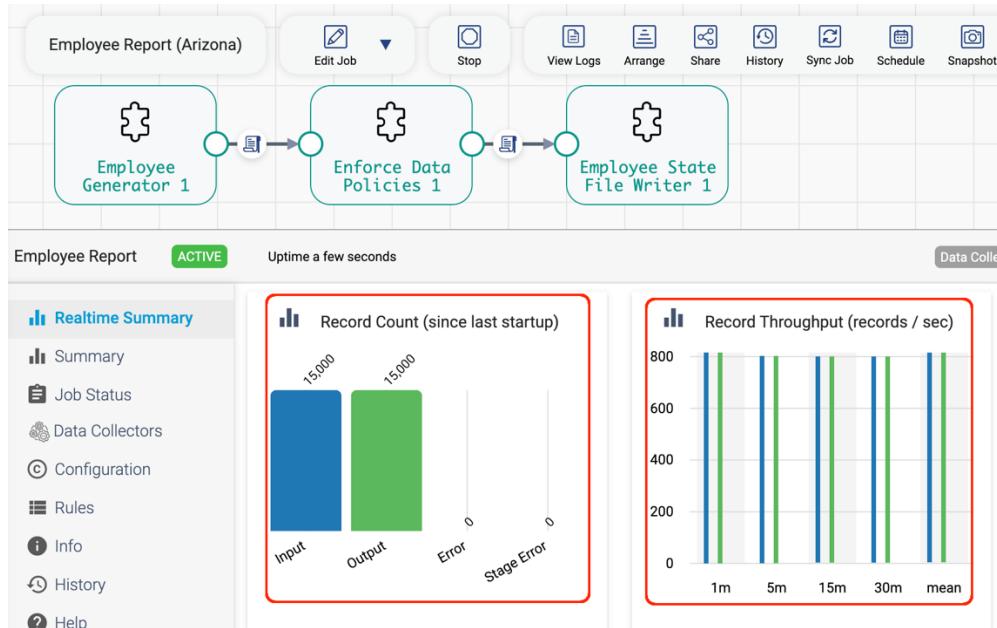
- To view Job instance details, click on the Job instance name. For example, the detail for the unhealthy Job shown above looks like this:

Job Instances > a - 2

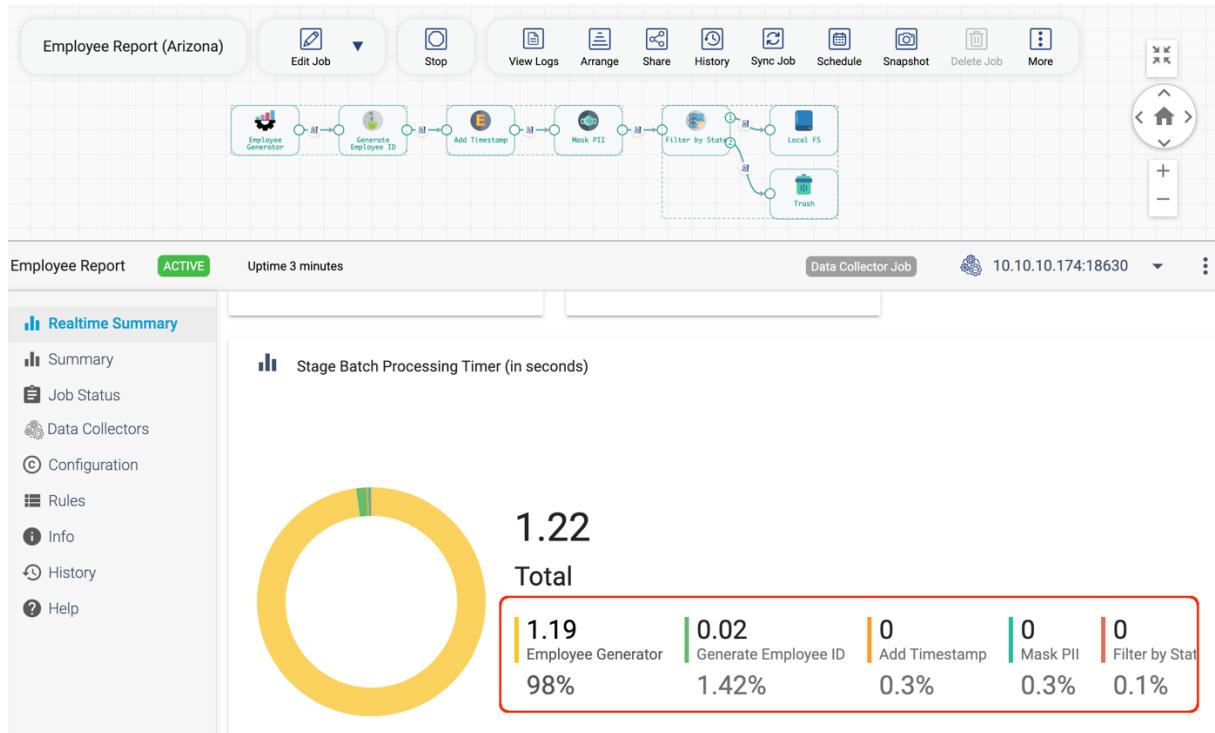
Job Instance Details		<span style="color: red;">! Monitor Job</span>	<span style="color: red;">✓ Acknowledge Error</span>	<span style="color: blue;">Delete</span>	<span style="color: blue;">Share</span>
<b>Status</b>			<b>Start Time</b>	Sep 24, 2024, 5:56:09 PM	
<span style="background-color: red; color: white; padding: 2px 5px;">INACTIVE_ERROR</span>			<b>Finish Time</b>	Dec 31, 1969, 4:00:00 PM	
			<b>Run Count</b>	2	
<b>Data Collectors</b>					
<span style="color: green;">10.10.10.174:18630</span>					
<b>Error Message</b>					
JOBRUNNER_64 - Force stopping job					

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:

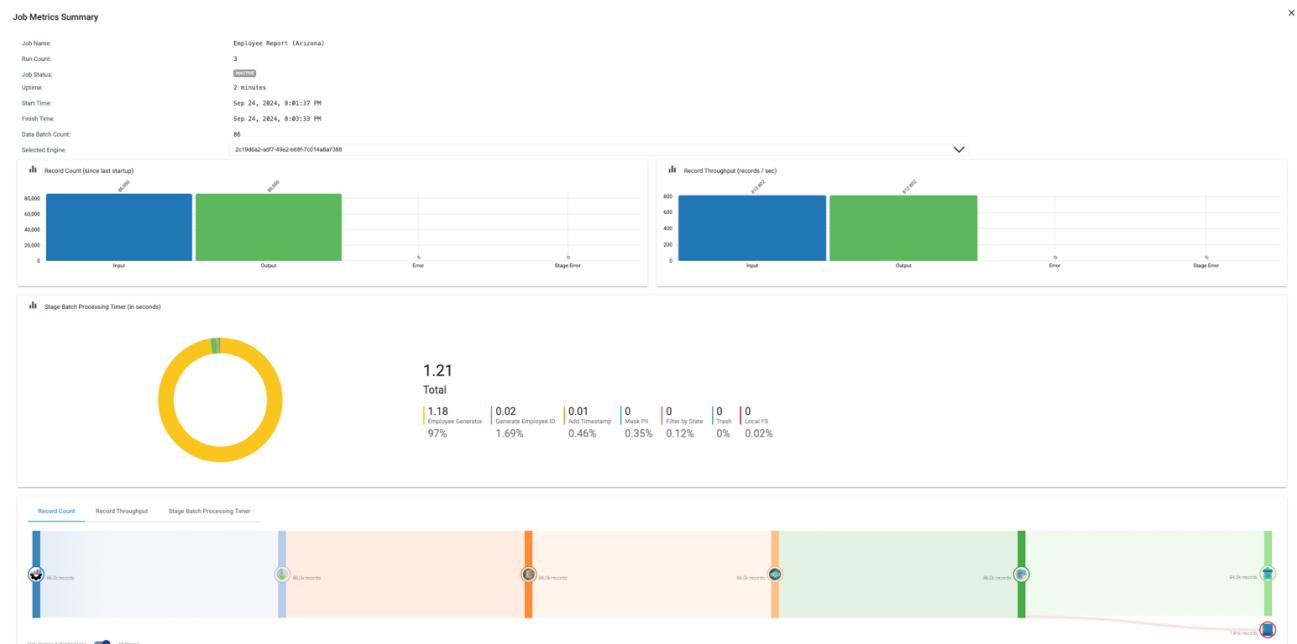


Note that 98% of the pipeline's time is spent in the Employee Generator stage, generating an ID takes 1.42% of the time, and the remaining processing steps take less than 1% each. This view allows you to identify if a poorly implemented processor is a bottleneck, or if a processor is slowed down by a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

The screenshot shows the IBM Data Collector interface. At the top, there's a toolbar with various icons like Edit Job, Start, View Logs, Share, History, Sync Job, Schedule, Snapshot, and Delete Job. Below the toolbar is a job flow diagram for an "Employee Report (Arizona)" job. The flow consists of several stages: Employee Generator, Generate Employee ID, Add Timestamp, Mask PII, Filter by State, Local FS, and a Trash stage. The "Filter by State" stage has a feedback loop pointing back to the "Add Timestamp" stage. The main area below the diagram shows the "Employee Report" job status as "INACTIVE". To the right, it displays the IP address "10.10.10.174:18630". On the left, there's a sidebar with navigation links: Realtime Summary, Summary, Job Status, Data Collectors, Configuration, Rules, Info, and History. The "History" link is highlighted with a red box. The main content area also contains a "Job History" table with three entries, each with a "View Summary" link.

- 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...

The screenshot shows the IBM StreamSets Control Hub interface. On the left, a sidebar lists various navigation options like Welcome, Set Up, Build, Run, Monitor, Operations Dashboard (which is highlighted with a red box), Topologies Dashboard, Subscriptions, Topologies, Reports, Alerts, Manage, and Legacy Kubernetes. Below the sidebar, there's a 'Collapse' button.

The main content area is titled 'Operations Dashboard'. It features a dashboard with six cards: 'Jobs in Process' (0 Total), 'Alerts' (55 Unread), 'Acknowledged' (0 in the last 24 hours), 'Jobs in Error State' (2 Total), 'Offline Engines' (3 Total), and 'Unhealthy Engines' (0 Total).

Below the dashboard, there are two sections: 'Alerts - 55' and 'Jobs In Error State - 2'. The 'Alerts' section has a table with columns: Message, Alert Type, Status, Triggered On, Acknowledged By, and Actions. The 'Jobs In Error State' section also has a table with similar columns: Job Name, Job Status, Pipeline Status, Status Message, Record Count, Record Throughput, Error Count, and Actions.

- **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:

The screenshot shows the IBM StreamSets Control Hub interface. On the left, a sidebar lists various navigation options: Welcome, Set Up, Build, Run, Job Templates, Job Instances (which is selected), Scheduled Tasks, Draft Runs, Sequences, Monitor, Manage, and Legacy Kubernetes. The main area displays a pipeline diagram with a Kafka Consumer node connected to a Databricks Delta Lake node. Below the diagram, a specific pipeline named "Weather to DeltaLake" is shown as INACTIVE. The pipeline status summary indicates it is "Pipeline is Idle" (Triggered: 13 days ago). A red box highlights a notification: "Batch taking more time to process". To the right, a panel titled "Alerts" shows a list of recent alerts. One alert, "SDC\_RULES" from Sep 9, 2024, at 12:43 PM, states "Pipeline is Idle" and has a red box around it. Another alert from 10:42 AM also mentions "Batch taking more time to process". A third alert from 9:50 AM and a fourth from 10:37 AM both state "Pipeline is Idle". A "VIEW ALL" button is at the bottom right of the alerts panel.

- **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:

The screenshot shows the "Subscriptions" configuration page in the IBM StreamSets Control Hub. The left sidebar includes "Welcome", "Set Up", "Build", "Run", "Monitor" (which is selected and highlighted with a red box), "Operations Dashboard", "Topologies Dashboard", "Subscriptions" (which is also highlighted with a red box), "Topologies", and "Reports". The main content area is titled "Subscriptions" and contains instructions: "Create a subscription by defining the trigger and then the action." It has fields for "Name:" and "Description:". Below these, under "Step One: Define Trigger Event", there is a "Event" dropdown menu. The menu is open, showing several options: "Choose Value", "Engine Not Responding", "Max Global Failover Retries Exhausted", "Data SLA Triggered" (which is selected and highlighted with a blue box), "Job Status Change", "Pipeline Committed", and "Pipeline Status Change".

- 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:

Engine	Running Pipelines
SDC-1 (Self-Managed) localhost:1...	0
SDC-3 (Self-Managed) localhost:1...	0
SDC-5 (Self-Managed) localhost:1...	0
SDC-2 (Self-Managed) localhost:1...	0
SDC-4 (Self-Managed) localhost:1...	0

- 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:

The screenshot shows the left sidebar with navigation options like Welcome, Set Up, Build, Run, Job Templates, Job Instances (highlighted with a red box), Scheduled Tasks, Draft Runs, Sequences, Monitor, Manage, and Legacy Kubernetes. The main area displays 'Job Instances > Prod Job for US-West'. It includes tabs for Job Instance Details (Status: INACTIVE, Start Time: Sep 24, 2024, 9:33:54 PM, Finish Time: Dec 31, 1969, 4:00:00 PM, Run Count: 0) and Data Collectors (empty). Below these are sections for Job Instance Name (Prod Job for US-West, Pipeline: Prod Job for US-West (v1)), Enable Failover (true), and Engine Labels (us-west, prod). Buttons for Monitor Job, Start Job, Edit, Export, Delete, Share, and Refresh are also present.

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

### Data Collector Engines (6)

<input type="checkbox"/>	Engine	Running
<input type="checkbox"/>	SDC-1 (Self-Managed) localhost:10...	
<input type="checkbox"/>	prod sdc-1 us-west	0
<input type="checkbox"/>	SDC-3 (Self-Managed) localhost:10...	
<input type="checkbox"/>	prod us-east sdc-3	0
<input type="checkbox"/>	SDC-5 (Self-Managed) localhost:11...	
<input type="checkbox"/>	dev sdc-5 on-prem	0
<input type="checkbox"/>	SDC-2 (Self-Managed) localhost:15...	
<input type="checkbox"/>	prod sdc-2 us-west	0
<input type="checkbox"/>	SDC-4 (Self-Managed) localhost:18...	
<input type="checkbox"/>	prod us-east sdc-4	0

- 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:

Data Collector Engines (6)

<input type="checkbox"/> Engine	Running F
<input type="checkbox"/> SDC-3 (Self-Managed) localhost:10...  <span style="border: 1px solid red; padding: 2px;">prod us-east sdc-3</span>	0
<input type="checkbox"/> SDC-4 (Self-Managed) localhost:18...  <span style="border: 1px solid red; padding: 2px;">prod us-east sdc-4</span>	0
<input type="checkbox"/> SDC-2 (Self-Managed) localhost:15...  <span style="border: 1px solid blue; padding: 2px;">prod sdc-2 us-west</span>	0
<input type="checkbox"/> SDC-1 (Self-Managed) localhost:10...  <span style="border: 1px solid blue; padding: 2px;">prod sdc-1 us-west</span>	0

- 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.

The screenshot shows the StreamSets interface with the 'Scheduled Tasks' page open. On the left, there's a sidebar with icons for Welcome, Set Up, Build, Run, Job Templates, Job Instances, and Scheduled Tasks. The 'Scheduled Tasks' icon is highlighted with a blue bar. The main area displays a table titled 'Scheduled Tasks (3)'. The columns are: Name, Cron Expression, Time Zone, Status, and Next Execution Time. The table contains three rows: 'Get Weather Events' (Cron: 0 0/1 \* ? \*), 'SQLServer to Snowflake' (Cron: 0 0/1 \* ? \*), and 'Weather Raw to Refined' (Cron: 0 0/1 \* ? \*). All three tasks are listed as 'Running' and will execute at '9/25/24, 5:00 PM'. At the bottom of the table, there are pagination controls for 'Items per page: 50' and '1 - 3 of 3'.

Name	Cron Expression	Time Zone	Status	Next Execution Time
Get Weather Events	0 0/1 * ? *	UTC	Running	9/25/24, 5:00 PM
SQLServer to Snowflake	0 0/1 * ? *	UTC	Running	9/25/24, 5:00 PM
Weather Raw to Refined	0 0/1 * ? *	UTC	Running	9/25/24, 5:00 PM

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:

The screenshot shows a user interface for managing a sequence of jobs. At the top, there are buttons for 'Add a Start Condition' (inactive), 'Disable Sequence', and help/cancel icons. Below is a table with columns for Step, Name, and Status (all inactive). Each row has a warning icon and a more options icon.

Step	Name	Status
1.	Get Weather Events	INACTIVE
2.	Weather Raw to Refined	INACTIVE
3a.	Weather to S3	INACTIVE
3b.	Weather to MongoDB	INACTIVE
3c.	Weather to ES	INACTIVE

Each job step has an associated configuration panel below it, containing an 'Auto-Start Next Step When in Error' checkbox.

## 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 parmeters>']

# 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](#)