

Splunk Data Compliance Pipelines for FSI Workshop

Lab Guide

Overview

This lab guide contains the hands-on exercises for the **Splunk Data Compliance Pipelines for FSI** workshop.

Please follow the steps documented here to complete the exercises.

Table of Contents

| | |
|--|----|
| Overview | 1 |
| Table of Contents..... | 1 |
| Exercise 1 – Access Your Lab Environment..... | 5 |
| Description | 5 |
| Log in to Splunk Show | 6 |
| Access Your Splunk Cloud Instance | 7 |
| Access Edge Processor Service | 9 |
| Access Your EC2 Instance (EP - Edge Node)..... | 10 |
| Exercise 2 – Create an Edge Processor | 12 |
| Description | 12 |
| Steps..... | 12 |
| Add a New Edge Processor | 12 |
| Install a New Edge Processor Instance..... | 14 |
| Restart Your Splunk Heavy Forwarder (Manual Step for Today's Environment)..... | 16 |
| Verify Edge Processor is Receiving Data..... | 16 |
| Exercise 3 – Configure an Amazon S3 Destination | 17 |
| Description | 17 |
| Steps..... | 17 |
| Add a new Amazon S3 destination | 17 |
| Exercise 4 – Buttercup Enterprises: Global Regulations (KYC)..... | 19 |

| | |
|--|----|
| Description | 19 |
| KYC Pipeline Guide: Global Regulations (Know Your Customer) | 19 |
| Steps..... | 19 |
| Create a New Pipeline | 20 |
| Define Archival Routing | 23 |
| Define PII Sanitization (Masking, Hashing, Redaction) | 25 |
| Preview Your Pipeline..... | 27 |
| Filtering of Non-Actionable Data..... | 27 |
| Identifying our Data..... | 28 |
| Save Your Pipeline | 29 |
| Apply the Pipeline to Your Edge Processor | 30 |
| Check Your Data in Splunk Cloud | 32 |
| Check Your Data in Amazon S3 Destination | 32 |
| Exercise 5 - Buttercup Enterprises: AMERICAS Regulations (PCI DSS) | 33 |
| Description | 33 |
| PCI Pipeline Guide: AMERICAS Regulations (PCI DSS)..... | 33 |
| Steps..... | 33 |
| Create a New Pipeline | 33 |
| Route Your Data to the 'pci_events Index..... | 36 |
| Define PII Sanitization (Masking, Hashing, Redaction) | 37 |
| Reviewing Fields..... | 37 |
| Masking PAN and Account Number..... | 38 |
| Filtering of Non-Storable Fields | 38 |
| PCI SPL Pipeline: | 38 |
| Preview Your Pipeline..... | 39 |
| Identifying our Data..... | 39 |
| Save Your Pipeline | 40 |
| Apply the Pipeline to Your Edge Processor | 41 |
| Confirm PCI Pipeline is applied | 41 |
| Check Your Data in Splunk Cloud | 41 |
| Exercise 6 – Buttercup Enterprises: EMEA Regulations (DORA) | 43 |
| Description | 43 |
| DORA Pipeline Guide: EMEA Regulations (Digital Operational Resilience Act) | 43 |
| Steps..... | 44 |
| Create a New Pipeline | 44 |
| Route Your Data to the dora_events Index..... | 46 |

| | |
|---|-----------|
| Define Conditional Raw Data Archival | 47 |
| Define Pseudonymization/Anonymization | 48 |
| Define Index Destination based on Severity Level of Event | 49 |
| Define Data Minimization by Field Removal | 52 |
| Add a Custom Indexed Field to Your Data..... | 53 |
| Preview Your Pipeline..... | 53 |
| DORA Pipeline | 53 |
| Save Your Pipeline | 54 |
| Apply the Pipeline to Your Edge Processor | 55 |
| Confirm DORA Pipeline is applied..... | 55 |
| Check Your Data in Splunk Cloud | 56 |
| Exercise 7 – Buttercup Enterprises: APJC Regulations (RMiT) | 57 |
| Description | 57 |
| RMiT Pipeline Guide: APJC Regulations (Risk Management in Technology) | 57 |
| Steps..... | 58 |
| Create a New Pipeline | 58 |
| Route Your Data to the rmit_events Index | 60 |
| Configure Lookup for Events | 61 |
| Lookup Enrichment Pattern for Nested JSON Events | 64 |
| Extract Fields from _raw | 64 |
| Build the Lookup Key and Perform the Lookup..... | 65 |
| Inject Lookup Results Back into _raw | 65 |
| Why This Pattern Matters | 65 |
| Enrich with Asset and Operational Context..... | 66 |
| Mask Sensitive Operational Details..... | 66 |
| Filter Non-Actionable Logs | 67 |
| Route Processed Data Based on RMiT Relevance | 68 |
| Filter Low-Value or Non-Actionable Logs..... | 70 |
| Add a Custom Indexed Field to Your Data..... | 70 |
| Configure Splunk Destination | 70 |
| Preview Your Pipeline..... | 73 |
| RMiT Pipeline..... | 73 |
| Save Your Pipeline | 76 |
| Apply the Pipeline to Your Edge Processor | 76 |
| Confirm RMiT Pipeline is applied | 77 |
| Check Your Data in Splunk Cloud | 77 |

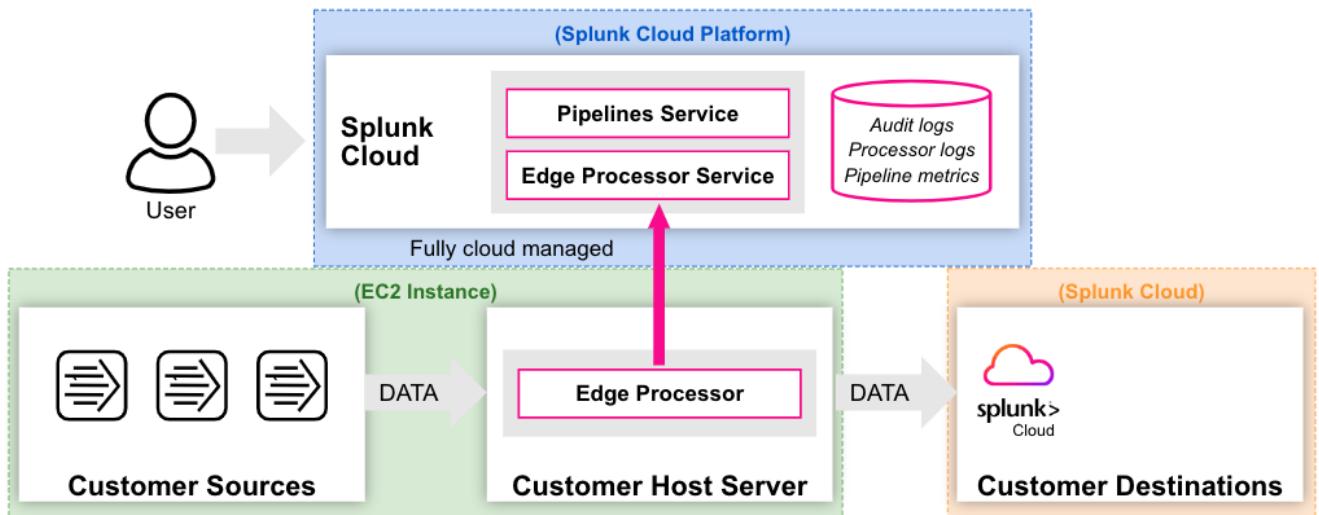
| | |
|---|-----|
| Exercise 8 – Buttercup Enterprises: ANZ Regulations (CPS-230) | 79 |
| Description | 79 |
| CPS 230 Pipeline Guide: A/NZ Regulations (Cross-industry Prudential Standard) | 79 |
| Steps | 80 |
| Create a New Pipeline | 80 |
| Route Your Data to the cps_events Index | 83 |
| Configure Lookup for Events | 84 |
| Lookup Enrichment Pattern for Nested JSON Events | 87 |
| Extract Lookup Key from _raw | 87 |
| Perform the Lookup | 87 |
| Inject Enrichment back into _raw | 87 |
| Clean Up Temporary Fields | 87 |
| Why This Pattern Matters | 87 |
| Define Enrichment with Critical Operation (CO) and Tolerance Context | 88 |
| Define Filtering for Significance and Actionability | 89 |
| Define Data Masking and Hashing | 89 |
| Define Archival Routing | 89 |
| Route Processed Data Based on CPS 230 APRA Notification | 90 |
| Add a Custom Indexed Field to Your Data | 93 |
| Preview Your Pipeline | 94 |
| CPS Pipeline | 95 |
| Save Your Pipeline | 96 |
| Apply the Pipeline to Your Edge Processor | 97 |
| Confirm CPS Pipeline is applied | 98 |
| Check Your Data in Splunk Cloud | 98 |
| Get Started with Splunk Data Management today | 100 |
| Request access to Edge or Ingest Processor | 100 |
| Join the Slack channel | 100 |
| Splunk Adoption Hub | 100 |

Exercise 1 – Access Your Lab Environment

Description

Before you can get hands-on with Edge Processor Service, first you'll need your own environment. Today's lab environment consists of the following:

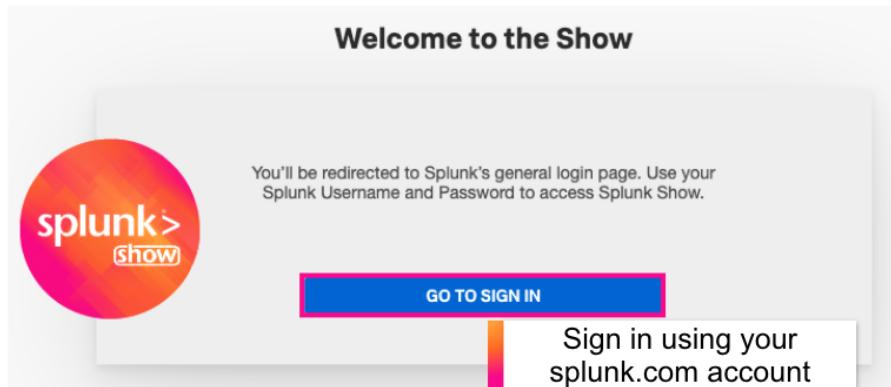
- **Splunk Cloud** - your data will be indexed here once it has been processed by Edge Processor.
- **Edge Processor service** - hosted in Splunk Cloud Platform, this is the cloud-based console through which you will setup, configure and manage Edge Processors, including creating pipelines for data processing (more on this later!)
- **EC2 instance (with demo data)** - this is where you will install a local Edge Processor node, which we will configure via the cloud-based Edge Processor service. Processed data will be forwarded to Splunk Cloud. In the lab environment this is referred to as '*EP - Edge Node*'.





Log in to Splunk Show

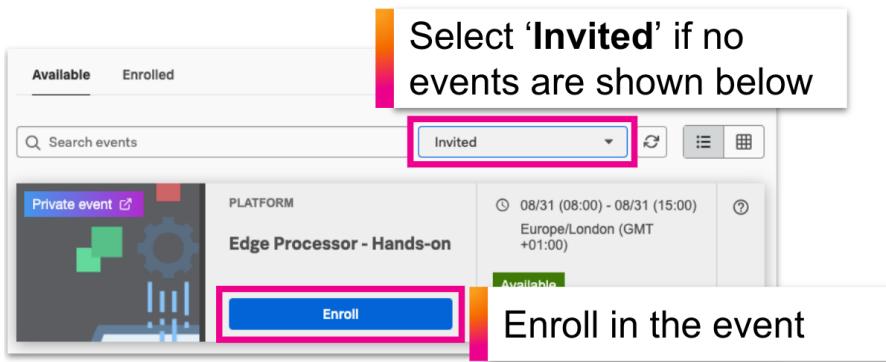
1. Browse to <https://show.splunk.com> and log in using your **Splunk.com account**.



🔑 Don't have a Splunk.com Account?

To access our hands-on workshop events you will need a Splunk.com account. If you don't already have a Splunk.com account, don't worry - it only takes a few minutes to create one! Please create one [here](#).

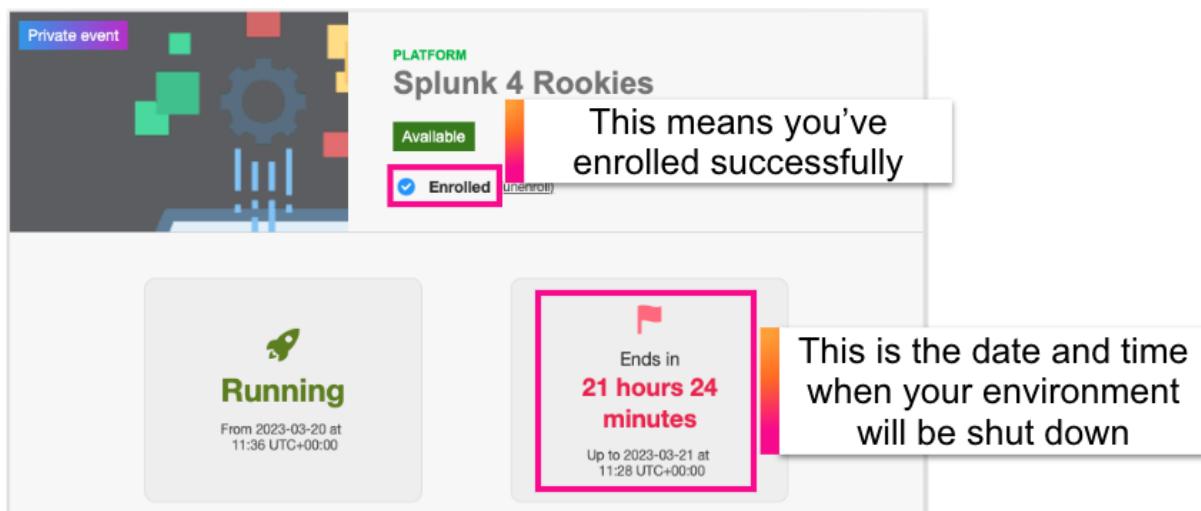
2. Once logged in to Splunk Show you will see the event page for the event that you have been invited to. If no events are listed, try selecting '**Invited**' from the dropdown list. Click on **Enroll** to join the event.



The page will refresh and the event will now display 'Enrolled'.

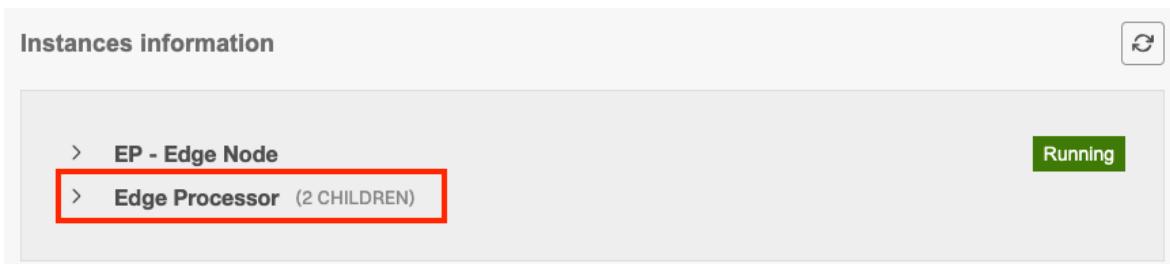
⌚ Lab environment expiration

All Splunk environments that are part of this workshop event will automatically be shut down at the date and time specified on this screen so feel free to continue to play around with your lab environment until then!



Access Your Splunk Cloud Instance

3. Browse to the Splunk Show event page and scroll down to the **Instances information** section. Expand out the '**Edge Processor**' section.



The screenshot shows the 'Instances information' section. It lists two items: 'EP - Edge Node' and 'Edge Processor (2 CHILDREN)'. The 'Edge Processor' item is highlighted with a red border. To the right of the list is a 'Running' status indicator with a refresh icon.

Here you will find credentials for:

- Your **Edge Processor Cloud Stack**, i.e. your Splunk Cloud stack; and
- Your **Edge Processor SCS Tenant**, i.e. the Edge Processor environment



- Locate the 'Edge Processor Cloud Stack' section and click on the 'Stack URL' link to open your Splunk Cloud stack.

Edge Processor Cloud Stack Running

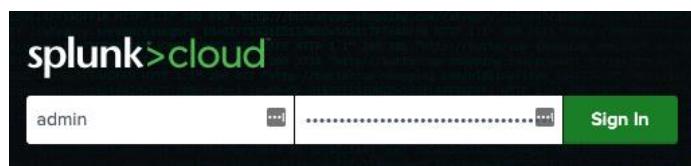
<https://eps-shw-63da8768c54802.stg.splunkcloud.com>

| Instance ID | Termination Date | User ID |
|--------------------------|-------------------------|---------|
| 64f02ea5b353ca8ca64b17e7 | 4 hours 45 minutes left | 6 |

Connection Information

| | |
|----------------|---|
| Admin Username | admin |
| Admin Password | |
| Username | sc_admin |
| Password | |
| Stack URL | https://eps-shw-63da8768c54802.stg.splunkcloud.com |

Use the admin credentials provided here to login to your Splunk account.



- Accept the Terms of Service and click on OK.

splunk>cloud

Terms of Service

SPLUNK GENERAL TERMS

Last updated: August 12, 2021

These Splunk General Terms ("General Terms") between Splunk Inc., a Delaware corporation, with its principal place of business at 270 Brannan Street, San Francisco, California 94107, U.S.A. ("Splunk" or "we" or "us" or "our") and you ("Customer" or "you" or "your") apply to the purchase of licenses and subscriptions for Splunk's Offerings. By clicking on the appropriate button, or by downloading, installing, accessing or using the Offerings, you agree to these General Terms. If you are entering into these General Terms on behalf of Customer, you represent that you have the authority to bind Customer. If you do not agree to these General Terms, or if you are not authorized to accept the General Terms on behalf of the Customer, do not download, install, access, or use any of the Offerings.

See the Definitions Exhibit section for definitions of capitalized terms not otherwise defined herein.

1. License Rights

A. General Rights. You have the nonexclusive, worldwide, nontransferable and nonsublicensable right, subject to payment of applicable Fees and compliance with the terms of these General Terms, to use your Purchased Offerings for your Internal Business Purposes during the Term and up to the Capacity purchased.

I accept these terms **Ok**



Access Edge Processor Service

6. On the Splunk Show event page, under the **Instances information** section, locate the ‘Edge Processor SCS Tenant’ connection information and click on the link provided.

Edge Processor SCS Tenant Running

<https://console.staging.scs.splunk.com/eps-shw-63da8768c54802>

| Instance ID | Termination Date | User ID |
|--------------------------|-------------------------|---------|
| 64f02ea5b353ca9c8f4b17e8 | 4 hours 45 minutes left | 6 |

Connection Information

| | |
|-------------|---|
| Console URL | https://console.staging.scs.splunk.com/eps-shw-63da8768c54802 |
|-------------|---|

Single Sign-On (SSO)

Single Sign-on (SSO) is configured between the Edge Processor service ('SCS Tenant') and Splunk Cloud environments, so if you already logged in to your Splunk Cloud stack you should automatically be logged in to Edge Processor service. If you are prompted for credentials, use the credentials provided in the Edge Processor on Splunk Show event (listed under the 'Edge Processor Cloud Stack' section.)

7. You will now be taken to the Edge Processor service environment. You can manage your Edge Processors via the menu on the left side of the page - we will create and configure these later in the workshop.

splunk>

Data Management

Get started with the Edge Processor solution

To process your data at the edge of your network, configure and install an Edge Processor. Then, create a pipeline and apply it to that Edge Processor.

[Add an Edge Processor](#) [Create Edge Processor pipeline](#)

What's new [View release notes](#)

My workspace **Shared workspaces** **Datasets** **Search** PREVIEW **Data management** **Pipelines** **Edge Processors** **Source types** **Shared settings** **Destinations**



Access Your EC2 Instance (EP - Edge Node)

8. Browse to the Splunk Show event page and scroll down to the **Instances information** section. Expand out the '**EP - Edge Node**' section.

Instances information

> EP - Edge Node Running

> Edge Processor (2 CHILDREN)

9. Locate the IP address and SSH details for your EC2 instance

EP - Edge Node Running

<https://i-0c76d1facb7ed24a1.splunk.show>

| Instance ID | Termination Date | User ID |
|--------------------------|-------------------------|---------|
| 64f02e96282158001dac53b6 | 4 hours 35 minutes left | - |

Connection Information

| | |
|----------------|---|
| Admin Username | admin |
| Admin Password | (redacted) |
| URL | https://i-0c76d1facb7ed24a1.splunk.show |
| SSH Password | (redacted) |
| SSH Command | ssh -p 2222 splunk@3.236.238.177 |

10. Connect to your EC2 instance by clicking on the copy button next to the SSH command.

SSH Command

ssh -p 2222 splunk@44.201.96.115 Copy

11. Paste it into your terminal app of choice, such as Terminal (Mac) or PowerShell (Windows).

```
~ % ssh -p 2222 splunk@44.201.96.115
```

When prompted, type '**yes**' and hit enter to confirm the new SSH connection.

```
The authenticity of host '[44.201.96.115]:2222 ([44.201.96.115]:2222)' can't be established.  
ED25519 key fingerprint is SHA256:WZSo8dQ+fXr6RUMSv+gv+nrTgZwUaKwurK01HFhwZhY.  
This key is not known by any other names  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

12. Copy the SSH password from the Splunk Show event by clicking on the copy button and pasting it into your terminal.

SSH Password

(redacted) Copy

You should now have a green command prompt, meaning you are successfully connected to your EC2 instance.

```
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.15.0-1028-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 System information as of Tue Aug 22 15:11:44 UTC 2023

 System load:  0.0          Processes:           205
 Usage of /:   3.7% of 193.81GB  Users logged in:    0
 Memory usage: 6%           IPv4 address for ens5: 172.31.79.144
 Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

 * Introducing Expanded Security Maintenance for Applications.
   Receive updates to over 25,000 software packages with your
   Ubuntu Pro subscription. Free for personal use.

   https://ubuntu.com/aws/pro

126 updates can be applied immediately.
84 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Wed Feb  8 19:24:19 2023 from 162.249.116.2
splunk@Domane-Demo-i-0e5ccec73c3e9c16b:~$ █
```

Exercise 2 – Create an Edge Processor

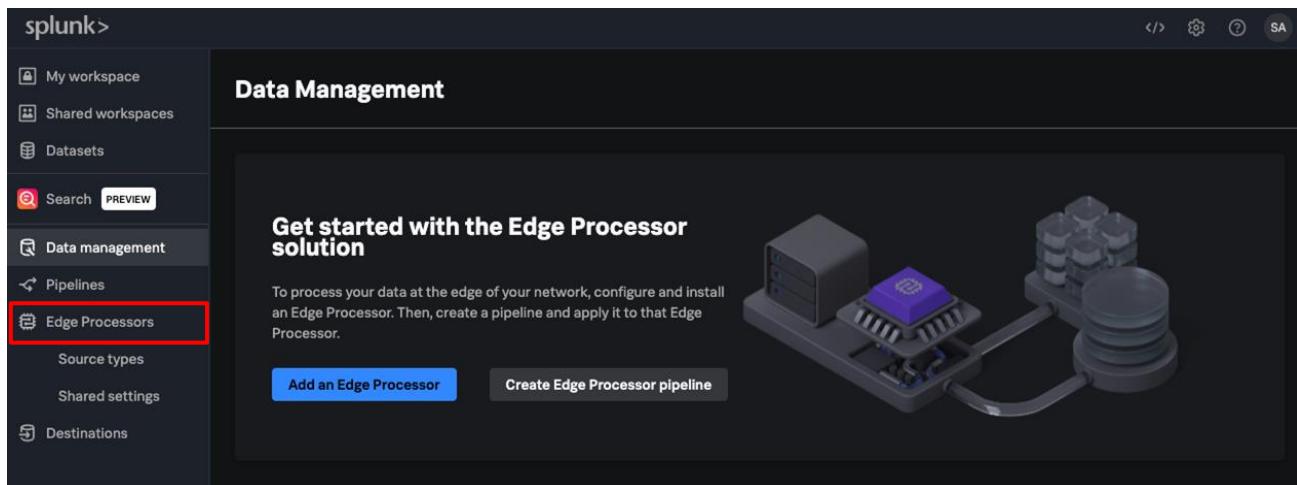
Description

In this exercise you will create a new Edge Processor via the Edge Processor service user interface and then install a new Edge Processor instance on your EC2 server via the command line. This will allow us to create pipelines (to process data) later in the workshop.

Steps

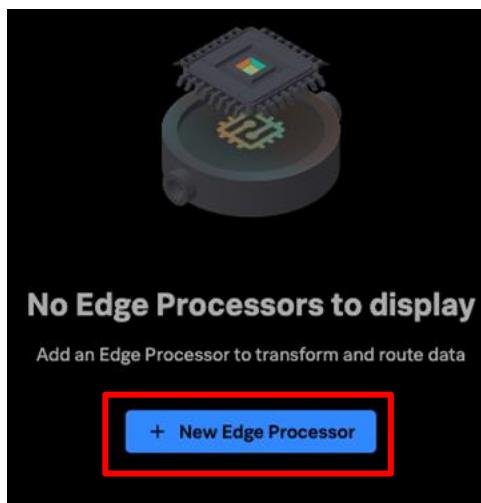
Add a New Edge Processor

1. Log in to the Edge Processor service (see [Access Edge Processor Service](#)) and from the menu on the left of the page click on **Edge Processors**.



The screenshot shows the Splunk Edge Processor service interface. The top navigation bar includes icons for code, gear, help, and user. The left sidebar has a dark background with white text and icons for My workspace, Shared workspaces, Datasets, Search (with PREVIEW button), Data management (highlighted with a red box), Pipelines, Edge Processors (highlighted with a red box), Source types, Shared settings, and Destinations. The main content area is titled 'Data Management' and features a section titled 'Get started with the Edge Processor solution'. It contains text about processing data at the edge of the network, configuring and installing an Edge Processor, and creating a pipeline. Below this are two buttons: 'Add an Edge Processor' (blue) and 'Create Edge Processor pipeline'. To the right is a 3D illustration of an Edge Processor unit connected to a network and databases.

2. In the middle of the **Edge Processors** page click on **+ New Edge Processor**.





3. The **Add an Edge Processor** window will appear. Give your Edge Processor a name. Since each environment is shared among multiple people, use the name '**edge_**' followed by your own name to help identify this as your Edge Processor. For example: **edge_jsmith**

For Default destination select '**default_splunk_cloud_destination**'.

To help simplify our setup for this workshop, disable TLS for Splunk forwarders by checking the **No secure connection protocol option** (this will remove the checks against the options below). Uncheck the HTTP Event Collector and Syslog options.

Click **Save** to create your Edge Processor.

Add an Edge Processor
Define an Edge Processor configuration.

* indicates a required field

Name *

Enter a unique name. Names can contain only letters, numbers, underscores, or hyphens. Maximum 80 characters (13/80)

Description

Maximum 250 characters (0/250)

Default destination
Specify whether the Edge Processor drops unprocessed logs or creates an "Unprocessed data" pipeline that routes it to a default destination. [Data pathway documentation](#)

Default destination *

Receiver settings
Configure how the Edge Processor receives data. Use mutually authenticated TLS (mTLS) to secure communication between this Edge Processor and your data sources. [Data source mTLS documentation](#)

Receive data from these inputs

Splunk forwarders ⓘ
 mTLS
 No secure connection protocol

HTTP Event Collector ⓘ
 mTLS
 TLS
 No secure connection protocol

Syslog
 mTLS

4. A new window will appear. Under “**Instances**” information, you will see a clipboard containing the installation commands, which we can use to install an Edge Processor instance onto a Linux server. Copy this command by clicking on the icon on the bottom, you will see it says “**Copy to Clipboard**”.

```

edge_hrobalin

Default destination ⓘ default_splunk_cloud_destination

Receivers
Splunk forwarders TLS off

Instances
To install an Edge Processor instance, run the following commands.
Once installed, the software will be automatically updated
periodically and will send service-related logs, metrics, and usage
data to The Splunk Platform.

# Download the Edge Processor installation package
curl -O "https://beam.staging.scs.splunk.com/splunk-edge/"

# Verify the package's integrity before proceeding with i
export SPLUNK_EDGE_PACKAGE_CHECKSUM=$(sha512sum splunk-ed

if [[ "$SPLUNK_EDGE_PACKAGE_CHECKSUM" == "e0e0db4fc4c13bf

then \
# Extract the installation package's contents into a dire
tar -xvzf splunk-edge.tar.gz

# Create a configuration file for the Edge Processor inst
echo "env: staging" > ./splunk-edge/etc/config.yaml
echo "tenant: dmx-shw-b3e6ea656462a" >> ./splunk-edge/et
echo "groupId: 53798c3f-9a1e-4bc4-be1e-d1f5312f76b8" >> .
echo "platform: 1" >> ./splunk-edge/etc/config.yaml
echo "url: https://dmx-shw-b3e6ea656462a.api.staging.scs

# Create an authentication token file used to connect the
echo "eyJhbGciOiJSUzI1NiisImtpZCI6Inl5VW1ET1BmVTRaR1huRm4

# Install and run the Edge Processor instance
mkdir -p ./splunk-edge/var/log
nohup ./splunk-edge/bin/splunk-edge run >> ./splunk-edge/
```

Copy to clipboard

Install a New Edge Processor Instance

5. With your installation command already copied to your clipboard, open your SSH connection to your EC2 instance (see [Access Your EC2 Instance](#)) and paste the command. The command will run automatically.

Due to the way the command is formatted, you will need to hit enter a second time after the initial command has run. You should now see a green prompt like before.

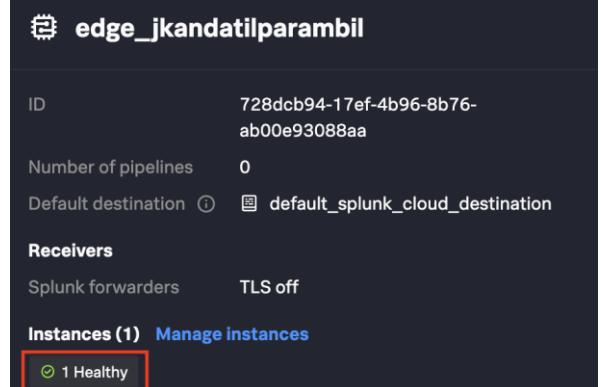
```

> tar -xvzf splunk-edge.tar.gz
>
> # Create a configuration file that determines which Splunk Cloud Services (SCS) environment, tenant, and Edge Processor cluster the instance gets associated with
> echo "groupId: c208c0c3-e19a-40cb-9e4a-ffdf5494b85e" > ./splunk-edge/etc/config.yaml
> echo "tenant: eps-shw-3c23343dca48c3" >> ./splunk-edge/etc/config.yaml
> echo "env: staging" >> ./splunk-edge/etc/config.yaml
>
> # Create a token file containing the authentication token that allows the Edge Processor instance to connect to SCS
> echo "eyJhbGciOiJSUzI1NiIsvtpZCI6In15VW1ET1BmVTRaR1huRm4rQ3JIzitEWjlld1RDQVlwQXI2dTVGcUt3bGsiLCJ0eXAiOjKV1QifQ.eyJhdWQiOlsiYXBp0i8vc2NzLWRlZmF1bHQixSwiY2lkIjoiMG9hOGcybWtxbndpNG5ka3gzNTciLCJlYyI6eyJzaWQiOjI2OTYwN2n1MTliNWR1M2RkY2BmNDkwMDNmMDA5YjRhMyIsInN1Yi6InNjx2FkbWluIn0sImV4cCI6MTY5Mjc2MDE4NSwiaWF0IjoxNjkyNzE20Tg1LCJpZHAiOjJlcHNfc2h3XzNjMjMzNDNkY2E00GMzLmVjLmkC5pZCIsIm1zcyI6Imh0dBz0i8vYXV0aC5zdGFnaW5nLnNjcy5zcGx1bmsuY29tIiwianRpIjoiY2ppY3Z1Y2trbjBxDOrpMWs5bDAiLCJzdWIoiJzY19hZG1pbkBAZxBzLXNody0zYzIzMzQzzGNhNDhjMyIsInR1bmFudCI6ImVwcy1zaHctM2MyMz0M2RjYTQ4YzMiFQ.R-DwmZqPq4efw7UaijbJTeduv20hdJJxVc3d3Yxy3UCw6sq2gK6chW-sekB0s8X8MZructcqWEoumAsI80q0L1bjssKNOAPWX2t9txj2xpGDtow0zt1GwpZ15K99wP70ZmYJYsSN-1S6HzDjtDs_NEjbStIDnqNUdcg90p73IUKPv6ms0tjzuUwgQW7tUroQlpkunck51FeQMK2T-dnLGJ1MYZILQ9833uoYDrf1_rkCfBMGFeGi17uI6Jau4_9Vn58JjlFr91DmBcfwCGfpzcB_vgYd1Cv0NXZtaIIfvf8hsbsUaACJ8zBKPdlUaej_W0Fg7EnjJYP0ciCZU8w" > splunk-edge/var/token
>
> # Install the Edge Processor instance
> mkdir -p ./splunk-edge/var/log
> nohup ./splunk-edge/bin/splunk-edge run >> ./splunk-edge/var/log/install-splunk-edge.out
2>&1 </dev/null &
>
> fi
splunk-edge/
splunk-edge/bin/
splunk-edge/bin/splunk-edge
splunk-edge/var/
splunk-edge/etc/
splunk-edge/etc/splunk-edge.service
[1] 75427
splunk@Domane-Demo-i-0e5cc73c3e9c16b:~$ 

```

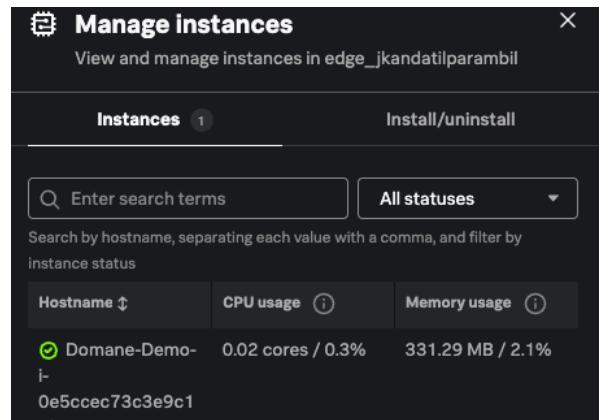
- After a brief moment (it could take a couple of minutes in today's shared environment) you should see the number of instances associated with your Edge Processor turn to **(1)** with the status of **1 Pending**. This will eventually change to **1 Healthy**.

You may need to refresh your browser page to see the status update.



The screenshot shows a Splunk Cloud interface for managing edge instances. At the top, it displays the instance ID: **edge_jkandatilparambil**. Below that, key details are listed: **ID: 728dc94-17ef-4b96-8b76-ab00e93088aa**, **Number of pipelines: 0**, and **Default destination: default_splunk_cloud_destination**. Under the **Receivers** section, it shows **Splunk forwarders: TLS off**. The **Instances (1)** section is highlighted with a red box, showing **1 Healthy**.

If you click on the blue **Manage instances** link you will see a list of instances, including the one you just installed. A green tick means the instance is healthy.



The screenshot shows a modal dialog titled **Manage instances** with the sub-instruction **View and manage instances in edge_jkandatilparambil**. It has two tabs: **Instances** (1) and **Install/uninstall**. The **Instances** tab is selected. At the top, there are search fields for **Enter search terms** and **All statuses**. Below is a table with columns for **Hostname**, **CPU usage**, and **Memory usage**. One instance is listed: **Domane-Demo-i-0e5cc73c3e9c16b**, which is marked as **1 Healthy**.



i Source types and Destinations

In a real world environment you would also need to configure source types and destinations (i.e. destinations where your data can be sent to) for your Edge Processor but to simplify today's workshop these have been pre-configured for you.

Restart Your Splunk Heavy Forwarder (Manual Step for Today's Environment)

7. For today's lab you will need to manually restart the Splunk Heavy Forwarder that is running on your EC2 instance. Note that you would not need to do this in a production environment.
Run the following command in the SSH session on your EC2 instance:

```
sudo /opt/splunk/bin/splunk restart
```

Confirm that Splunk restarts successfully:

```
If you get stuck, we're here to help.  
Look for answers here: http://docs.splunk.com  
  
The Splunk web interface is at https://Domane-Demo-i-03397b9fa7444e973  
  
splunk@Domane-Demo-i-03397b9fa7444e973:~$
```

Verify Edge Processor is Receiving Data

8. Browse to the Edge Processor page and double-click on your Edge Processor.

The screenshot shows the Splunk Edge Processor page. On the left, there is a sidebar with various navigation options: My workspace, Shared workspaces, Datasets, Search (PREVIEW), Data management, Source types, Edge Processors (which is highlighted with a red box), Pipelines, and Destinations. The main area is titled "Edge Processors". It displays five status metrics: All Instances (1), Instances with Error (0), Instances with Warning (0), Instances with Disconnected (0), and Instances with Healthy (1). Below these metrics is a search bar labeled "Search by Edge Processor name. Separate each value with a comma." A table follows, showing a single row for the edge processor: Name (edge_jkandatilparambil), Number of instances (1), Instance health (green bar indicating healthy), and Number of pipelines (0).

9. You have now successfully added the Edge Processor Node. We can now begin on configuring our destination locations.

Exercise 3 – Configure an Amazon S3 Destination

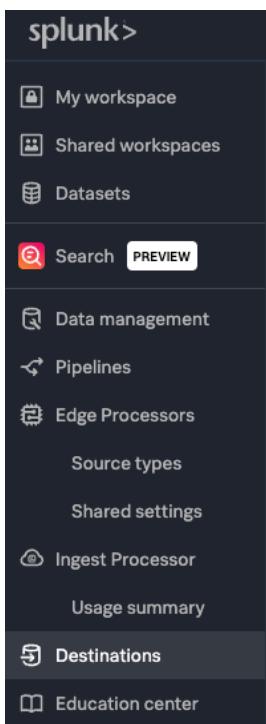
Description

In this exercise you will add new destination to an Amazon S3 bucket. An Amazon S3 bucket can be used as an archival destination.

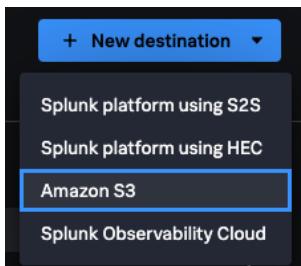
Steps

Add a new Amazon S3 destination

1. On the side menu on the left of the page click on **Destinations**



2. On the top right side, select on **+ New destination**, selecting **Amazon S3** from the dropdown.



3. A new panel will open where you will need to fill out your information. Once complete click **Add**.

Add an Amazon S3 destination dataset

Create a destination for connecting to an Amazon S3 bucket

* indicates a required field

Destination information

* Name Name should start with a letter and contain only lowercase letters, numbers, underscores, or the at (@) character. Maximum 80 characters (0/80)

Description Maximum 250 characters (0/250)

Amazon S3 object key name settings

The object key for your data is constructed using auto-generated values from the processor, and the values you specify in the following fields. [Send data to Amazon S3](#)

* Bucket name

Folder name [\(i\)](#)

File prefix [\(i\)](#)

* Output data format [\(i\)](#)

* Compression type [\(i\)](#)

Amazon S3 general settings

* Region [\(i\)](#)

Add **Cancel**

4. You should now see that your new Amazon S3 destination has been added as a Destination for Edge Processor.

| Destinations | |
|--|---------------------|
| All 4 | Splunk 3 |
| <input type="text"/> Filter by name <input type="button" value="All kinds"/> | |
| Name ↑ | Kind ↓ |
|  default_splunk_cloud_destination | Splunk platform S2S |
|  edge_s3_hrobalin | Amazon S3 |
|  splunk_indexer | Splunk platform |
|  _splunk_telemetry | Splunk platform |

Exercise 4 – Buttercup Enterprises: Global Regulations (KYC)

Description

With your new Edge Processor installed and ready for configuration, it's time to create data pipelines to enable efficient processing at the edge.

The Buttercup Enterprises compliance team has requested that all data from Know Your Customer (KYC) applications be filtered to ensure regulatory compliance. This involves both archiving the data and making it available for analytics.

To meet these requirements, you will create a pipeline that filters incoming KYC data and routes it appropriately for secure storage and analysis.

⚠️ Downstream Impacts of Data Processing

Applying masking, filtering and transforming rules to your data can impact downstream processing and analytics that rely on specific fields - such as IP addresses or usernames - to provide insights and intelligence. For example, Splunk Enterprise Security (ES) relies on IP addresses and usernames as part of its risk-based alerting (RBA) functionality, meaning that masking either of those fields would limit Splunk's ability to detect risk within your security environment.
Always check the potential impact of a pipeline before applying it!

KYC Pipeline Guide: Global Regulations (Know Your Customer)

Objective: The Know Your Customer (KYC) pipeline is designed to process incoming KYC application data. Its primary goals are to:

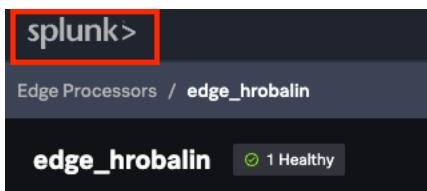
1. **Archive Raw Data:** Send an unmodified copy of all raw KYC data to a secure Amazon S3 bucket for long-term storage.
2. **Sanitize PII:** Redact, mask, or hash Personally Identifiable Information (PII) such as full names, dates of birth, emails, phone numbers, street addresses, and document IDs, before the data reaches Splunk Cloud for analysis.
3. **Filter Relevant Events:** Keep only KYC events that have been "approved" (i.e., `event_type == "kyc_approved"` and `event_status == "decision_approved"`).
4. **Route to Specific Index:** Send the processed and sanitized data to the `kyc_events` index in Splunk Cloud.
5. **Identify Your Data:** Add a custom field to easily identify your events in a shared Splunk Cloud environment.

Steps

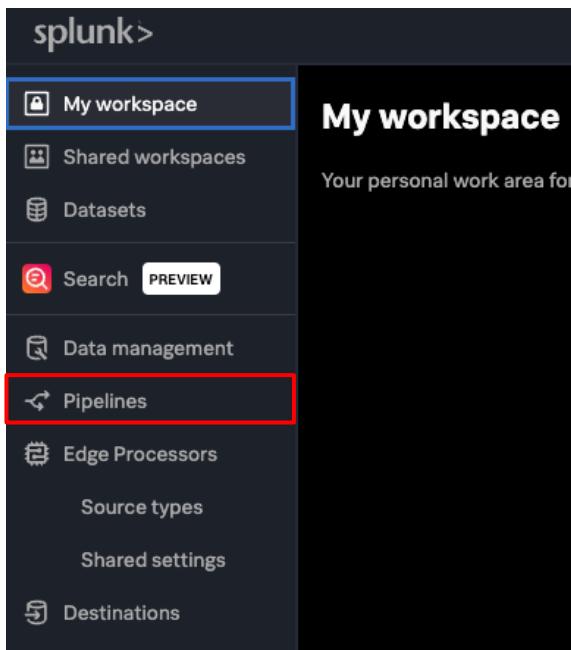


Create a New Pipeline

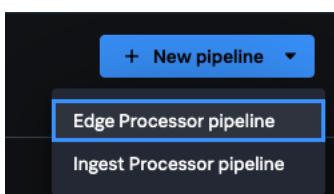
1. Log in to the Edge Processor service (see [Access Edge Processor Service](#)). If you're still on the page for your Edge Processor, you can navigate to the main page by clicking on the Splunk logo in the top left corner of the screen.



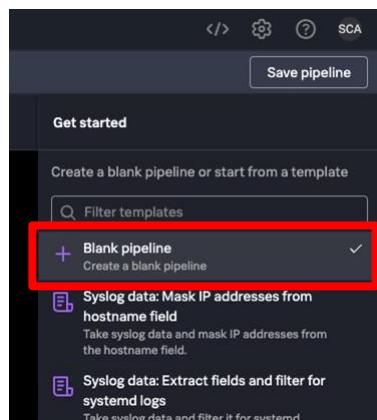
2. On the side menu on the left of the page click on **Pipelines**.



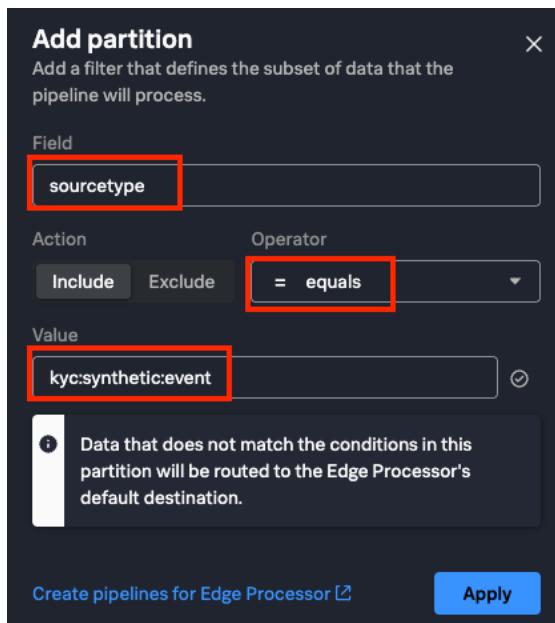
3. In the top right corner of the **Pipelines** page click on **+ New pipeline**. Select **Edge Processor pipeline**



4. On the right-hand side of the **New pipeline** page under the **Get started** pane select on **Blank pipeline** and click on **Next**.



5. On the 'Define your pipeline's partition' pane, click the add button (+) next to the 'Partition' field. Configure the new partition rule by setting 'Field' to sourcetype, 'Action' to 'Include', 'Operator' to '= equals', and set **kyc:synthetic:event** as the 'Value', then click 'Apply'.



Note: Make sure you don't have an extra space.

And then Click on **Next** button at the bottom right.

6. An **Add sample data** menu will appear on the right side.

Download the data-fsi-samples.zip file from <https://splk.it/Data-FSI-Samples> and extract the files to your computer. We will upload each of these samples during various exercises in this workshop.

7. Upload the **kyc_sample_events.jsonl** file that you extracted from the zip file earlier using the "**upload file...**" option at the bottom.



Add sample data

Enter or upload sample data

```
"kyc_rejected", "event_status": "decision_rejected"
"profile_update_requested", "event_status": "updated"
"account_flagged", "event_status": "account_flagged"
"kyc_review_required", "event_status": "escalated"
"profile_update_requested", "event_status": "updated"
"kyc_approved", "event_status": "decision_approved"
"kyc_approved", "event_status": "decision_approved"
"background_check_initiated", "event_status": "initiated"
"profile_update_requested", "event_status": "updated"
"kyc_rejected", "event_status": "decision_rejected"
"background_check_initiated", "event_status": "initiated"
"profile_update_requested", "event_status": "updated"
"kyc_review_required", "event_status": "escalated"
"kyc_rejected", "event_status": "decision_rejected"
"document_uploaded", "event_status": "uploaded"
"address_verification_pending", "event_status": "pending"
"kyc_review_required", "event_status": "escalated"
"kyc_review_required", "event_status": "escalated"
```

23,025 / 30,000 characters

Drop your file anywhere or [upload file...](#)

Sample data format

Raw text CSV

For the **Sample data format** select **Raw text** and click on **Next**.

8. A “**Select a data destination**” menu will appear. Select the **default_splunk_cloud_destination** as your data destination.

Select a data destination

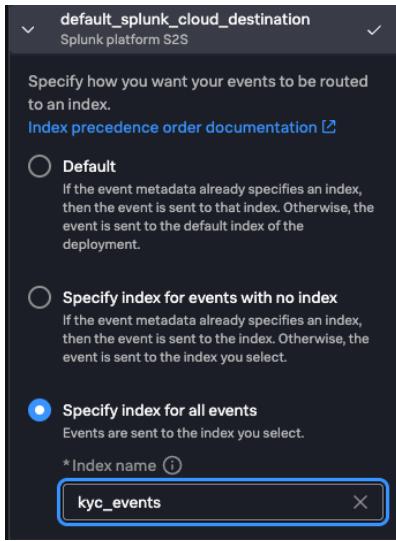
Filter datasets

All kinds

> **default_splunk_cloud_destination**
Splunk platform S2S

edge_s3_hrobalin
Amazon S3

9. A pane below will open, there you will select **Specify index for all events** and set the index to the **kyc_events** from the drop down, as requested, and click the “**Done**” button in the bottom.



10. A screen like below will appear:

| View: Table |
|--|
| <pre></> _raw 1 [{"event_id": "d58ff927-110c-4b87-885c-ea7ea0c50844", "timestamp": "2025-10-30T14:50:36.339772Z", "user_id": "user_26738", "event_type": "kyc_rejected", "event_status": "decision_rejected", "source_ip": "147.264.119.96", "device_id": "6107218c958393721baa404063387", "session_id": "1ba272f1-0cef-4691-9"}, {"event_id": "97ad5eb-c379-4b0b-967b-cfe5e5c73cd4", "timestamp": "2025-10-30T14:50:36.339772Z", "user_id": "user_53239", "event_type": "profile_update_requested", "event_status": "update_request_logged", "source_ip": "118.160.113.137", "device_id": "55c8022240492fde7d6d31a5e5e24cb7", "session_id": "461d"}, {"event_id": "255999df7-f1b8-4c53-a545-a7862ac9ffcd", "timestamp": "2025-10-30T14:34:33.261998Z", "user_id": "user_72792", "event_type": "account_flagged", "event_status": "account_status_flagged", "source_ip": "119.108.245.113", "device_id": "64de001c16695885051eb377595a49c", "session_id": "9f9e1827-3"}]</pre> |
| As index As sourcetype kyc_events kyc:synthetic:event kyc_events kyc:synthetic:event kyc_events kyc:synthetic:event |

As you integrate additional actions into your pipeline, please insert them between the existing pipeline stages and the final index. It is crucial to maintain the order specified in this guide. Refer to the example below:

```
$pipeline = | from $source
. . . ADD NEW ACTIONS HERE
| eval index = "kyc_events"
| into $destination;
```

Define Archival Routing

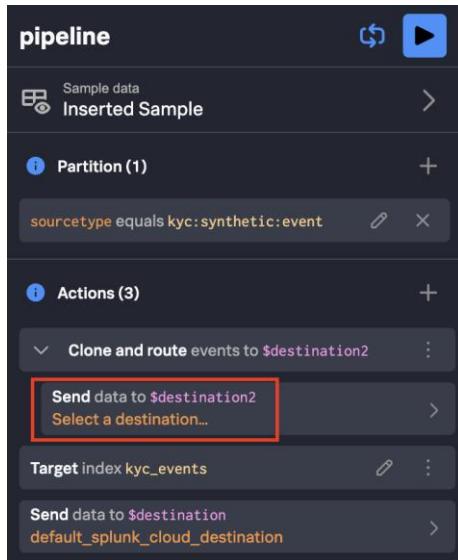
We will paste the following SPL2 statement. This will define Splunk as \$destination, and Amazon S3 Bucket for Archiving as \$destination2:

```
| thru [ | into $destination2 ]
```

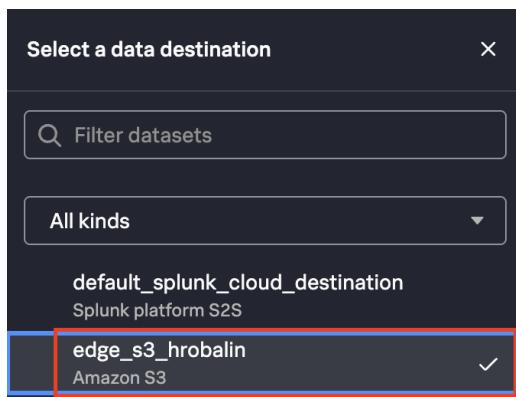
11. Your SPL2 Pipeline should now look like this:

```
$pipeline = | from $source
             | thru [ | into $destination2 ]
             | eval index = "kyc_events"
             | into $destination;
```

12. Now let's define in the pipeline setting which Amazon S3 Bucket will be \$destination2. Click Select a destination



13. Select the Amazon S3 Endpoint for our use case: edge_s3_username.



14. Click **apply** at the bottom right.

15. **Actions** should now look like this, where \$destination2 is S3 Destination:



The screenshot shows the Splunk Actions interface with three actions listed:

- Clone and route events to edge_s3_hrobalin
- Send data to \$destination2 edge_s3_hrobalin
- Target index kyc_events (with a pencil icon)

Define PII Sanitization (Masking, Hashing, Redaction)

16. Now we need to sanitize any of the data going into our Splunk Index. In our case since our data is a CSV (i.e. structured), we will use the [replace function](#) to replace data in our PII field:

First, we will redact the Full Name using the `eval` command.

```
// Redact full_name within the 'pii' string
| eval _raw=json_set(_raw, "pii.full_name", "REDACTED_NAME")
```

17. Your SPL2 statement, which should now look like this:

```
$pipeline = | from $source

// Send raw data to secure archive (e.g., Amazon S3 Bucket)
| thru [ | into $destination2 ]

// Redact full_name within the 'pii' string
| eval _raw=json_set(_raw, "pii.full_name", "REDACTED_NAME")

// Set index for KYC events
| eval index = "kyc_events"

| into $destination;
```

Regular Expression Syntax in Edge Processor

Edge Processor supports Perl Compatible Regular Expressions (PCRE) syntax. When you use a command or function that uses a regular expression, such as the `rex` command or the `replace` evaluation function, you must write the regular expressions in PCRE syntax (see [Supported regular expression syntax](#) for more information). If you use a regex tester such as [regex101](#) you will need to select '**PCRE2 (PHP >=7.3)**' from the FLAVOR menu on the left of the page in order to ensure your regular expressions will work with Edge Processor!



18. To redact the remaining PII fields we will copy, and paste the code sample below:

```
$pipeline = | from $source

    // Send raw data to secure archive (e.g., Amazon S3 Bucket)
    | thru [ | into $destination2 ]

    // Redact full_name within the 'pii' string
    | eval _raw=json_set(_raw, "pii.full_name", "REDACTED_NAME")

    // Redact date_of_birth within the 'pii' string
    | eval _raw = json_set(_raw, "pii.date_of_birth", "REDACTED_DOB")

    | eval masked_email = replace(_raw.pii.email, /^(.)(.*)(@.*$)/, "$1****$3")
    | eval _raw=json_set(_raw, "pii.email", masked_email)

    // Mask phone_number within the 'pii' string
    | eval masked_phone = replace(_raw.pii.phone_number,
/^(\.{3})(\.{4}[xX\d]*$)/, "$1***$3")
    | eval _raw=json_set(_raw, "pii.phone_number", masked_phone)

    // Redact street_address within the 'address' string
    | eval _raw=json_set(_raw, "address.street_address", "REDACTED_ADDRESS")

    // Extract document_id and replace with hash for all documents
    | eval document_id_0 = json_extract(_raw, "documents{0}.document_id")
    | eval hash_doc_id_0 = md5(document_id_0)
    | eval _raw = json_set(_raw, "documents{0}.document_id", hash_doc_id_0)

    | eval document_id_1 = json_extract(_raw, "documents{1}.document_id")
    | eval hash_doc_id_1 = md5(document_id_1)
    | eval _raw = if(isnotnull(document_id_1), json_set(_raw,
"documents{1}.document_id", hash_doc_id_1), _raw)
```

```
// // Drop the temp field
| fields - masked_phone, masked_email, hash_doc_id, document_id_0,
hash_doc_id_0, document_id_1, hash_doc_id_1

// Set index for KYC events
| eval index = "kyc_events"

| into $destination;
```

19. Below is an example of how we can extract a value, create a hash from it, and write it back into the event:

```
// Extract document_id and replace with hash for all documents
| eval document_id = json_extract(_raw, "documents{0}.document_id")
| eval hash_doc_id = md5(document_id)
| eval _raw = json_set(_raw, "documents{0}.document_id", hash_doc_id)
```

Preview Your Pipeline

20. Test your masking rule by clicking on the blue Preview pipeline button (in the top right corner of the screen. If your pipeline filter is correct you should see some sample events containing “REDACTED” and Hash values inside of address and documents respectively.

| | _raw |
|---|--|
| 1 | {"pii": {"full_name": "REDACTED_NAME", "nationality": "CO", "_internal_nationality_code": "CO", "date_of_birth": "REDACTED_DOB", "phone_number": "361***0406", "email": "D****@EXAMPLE.COM"}, "address": {"country_code": "UY", "street_address": "REDACTED_ADDRESS", "city": "Rhodesville", "state_province": "IA", "postal_code": "869..."}} |
| 2 | {"pii": {"full_name": "REDACTED_NAME", "nationality": "BO", "_internal_nationality_code": "BO", "date_of_birth": "REDACTED_DOB", "phone_number": "001***5350", "email": "F****@EXAMPLE.COM"}, "address": {"country_code": "GB", "street_address": "REDACTED_ADDRESS", "city": "Sheilamouth", "state_province": "MH", "postal_code": "709..."}} |

Sample Data

The results shown in the pipeline preview pane are from sample data that is uploaded when you configure your source types within the Edge Processor Service UI. They are not live events from your data stream.

Filtering of Non-Actionable Data

21. We can filter out data that is not required for our analytics, helping reduce data volume. An example of how to do this here is to filter for event_status where the decision is approved. This would look like the following in SPL2:

```
// Filter to only keep successful KYC event status and event type
| where json_extract(_raw, "event_type") == "kyc_approved" AND json_extract(_raw, "event_status") == "decision_approved"
```

22. Our final SPL2 query to filter our KYC events should look like the following:

```
$pipeline = | from $source
    // Send raw data to secure archive (e.g., Amazon S3 Bucket)
    | thru [ | into $destination2 ]

    // Redact full_name within the 'pii' string
    | eval _raw=json_set(_raw, "pii.full_name", "REDACTED_NAME")

    // Redact date_of_birth within the 'pii' string
    | eval _raw = json_set(_raw, "pii.date_of_birth", "REDACTED_DOB")

    | eval masked_email = replace(_raw.pii.email, /^(.)(.*)(@.*$)/, "$1****$3")
    | eval _raw=json_set(_raw, "pii.email", masked_email)

    // Mask phone_number within the 'pii' string
    | eval masked_phone = replace(_raw.pii.phone_number,
/^.{3})(.{4}[xX\d]*$/, "$1***$3")
    | eval _raw=json_set(_raw, "pii.phone_number", masked_phone)

    // Redact street_address within the 'address' string
    | eval _raw=json_set(_raw, "address.street_address", "REDACTED_ADDRESS")

    // Extract document_id and replace with hash for all documents
    | eval document_id_0 = json_extract(_raw, "documents{0}.document_id")
    | eval hash_doc_id_0 = md5(document_id_0)
    | eval _raw = json_set(_raw, "documents{0}.document_id", hash_doc_id_0)
    | eval document_id_1 = json_extract(_raw, "documents{1}.document_id")
    | eval hash_doc_id_1 = md5(document_id_1)
    | eval _raw = if(isnotnull(document_id_1), json_set(_raw,
"documents{1}.document_id", hash_doc_id_1), _raw)

    // // Drop the temp field
    | fields - masked_phone, masked_email, hash_doc_id, document_id_0,
hash_doc_id_0, document_id_1, hash_doc_id_1

    // Filter to only keep successful KYC event status and event type
    | where json_extract(_raw, "event_type") == "kyc_approved" AND
json_extract(_raw, "event_status") == "decision_approved"

    // Set index for KYC events
    | eval index = "kyc_events"
    | into $destination;
```

Identifying our Data

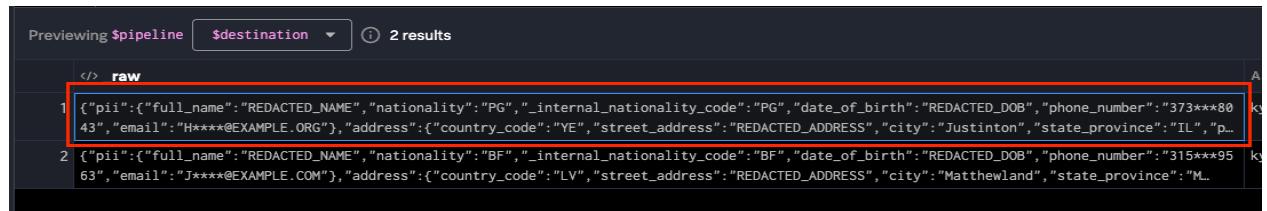
23. Finally, to help you identify data coming from your own pipeline when looking in your shared Splunk Cloud instance, add a custom indexed field to your events by adding an `eval` command to

your pipeline to create a field called ‘participant’ with the value set to your own name. Place this field right before we eval our index as kyc_events as seen below:

```
$pipeline =
| from $source
...
| eval _raw = json_set(_raw, "participant", "hrobalin") ← Update with your own name
| eval index = "kyc_events"
| into $destination;
```

To verify that the new field will be created, click on the Preview button (▶) again. Since we’re defining a new field=value pair as part of our pipeline, the new field will display in the Row preview table.

Select a field from the **_raw** column

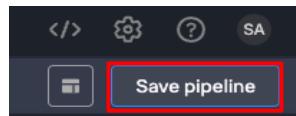


A modal will appear on the right side where the **_raw** event message will be expanded, if you scroll to the bottom, you will see the newly added participant field

```
user_id: "user_70642",
session_id:
"2a1c2709-e9f9-45fa-a142-1148b5979249",
participant: "hrobalin",
event_status: "decision_approved",
source_ip: "10.79.125.235",
timestamp: "2025-10-30T14:34:33.261998Z"
}
```

Save Your Pipeline

- Save your pipeline by clicking on **Save pipeline** in the top right corner of the screen. Give your pipeline a suitable name, such as **kyc_events_send_to_index_<yourName>**. Be sure to include your name somewhere in the pipeline name to avoid confusion among those who are sharing the same environment as you!





Save pipeline

Name *

 (33/80)

Description (optional)

 (0/512)

Cancel Save

25. Click **Save** to save your pipeline.

Apply the Pipeline to Your Edge Processor

26. Now an “Apply Pipeline” dialog box will appear, click on “**Yes, apply**”.

Apply pipeline

You can now apply kyc_events_send_to_index_hrobalin to processors to start manipulating data as defined in the pipeline. Do you want to apply it now?

No Yes, apply

27. A pane called “**Apply or remove <pipeline-name> to Edge Processors**” will appear, Select the edge processor you created earlier (see [Add a New Edge Processor](#)) and click on Save in the bottom.

Apply or remove
"kyc_events_send_to_index_hrobalin" to Edge Processors

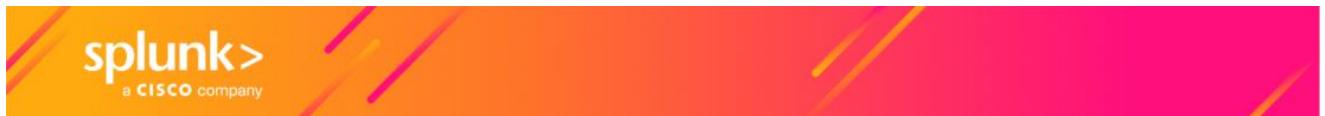
Specify which Edge Processors you want this pipeline to be applied to

| | NAME ↑ | |
|---|---------------|-------------------------------------|
| > | edge_hrobalin | <input checked="" type="checkbox"/> |
| > | | <input type="checkbox"/> |

You will see a brief message stating that your changes are being saved. Note that it may take a few minutes before you see the effects of your pipeline in Splunk Cloud.

Saving your changes

i It might take several minutes for pipelines to be applied to or removed from Edge Processors. You can check the progress on the Pipelines page.



28. To check the status of your edge processor node as the pipeline is being applied, click on the **splunk>** icon on the top left and then click on “**Pipelines**” on the home page and click on the pipeline created.

29. A panel will open up on the right of the page, showing the status of the Edge Processor instance that this pipeline has been applied to.

Click the arrow (**>**) next to the instance name to view more information.

kyc_events_send_to_index_hrobalin

Edit Apply/remove :

Processor type Edge Processor

Usage 1 Edge Processor

ID 21e4aa92-4bae-4119-b74b-89cad3e9b33d

SPL2 definition Expand

Partition sourcetype equals kyc:synthetic:event

Destinations edge_s3_hrobalin, default_splunk_cloud_destination

Pipeline location /shared.pipelines/kyc_events_send_to_index_hrobalin

Created 9 Oct 2025, 2:25PM

[View more](#)

Edge Processors that this pipeline is applied on (1)

| Edge Processor ID | Instance status | Number of pipelines |
|--------------------------------------|-----------------|---------------------|
| fc05caaa-65ac-4513-a69c-658ec62898b8 | 1 | 1 |

30. Once your pipeline has successfully applied the status of your instance should change to

1 Healthy

To verify that the instance is indeed healthy, click on **Edge Processors** in the menu on the left of the screen and refresh your web browser until your Edge Processor displays as Healthy.

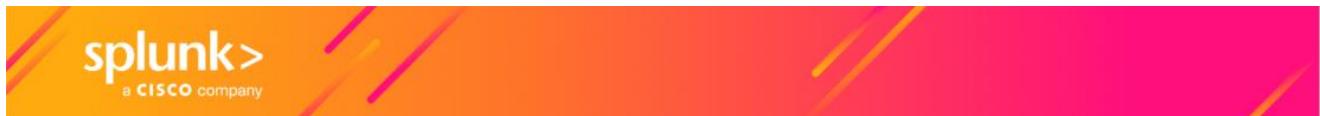
Edge Processors

| All instances | Instances with Error | Instances with Warning | Instances with Disconnected | Instances with Healthy |
|---------------|----------------------|------------------------|-----------------------------|------------------------|
| 1 | 0 | 0 | 0 | 1 |

edge_hrobalin

| Name | # of instances | Inbound data | Outbound data | Instance health | # of pipelines |
|---------------|----------------|-------------------------|------------------------|---|----------------|
| edge_hrobalin | 1 | 163.17 MB / 824K events | 94.85 MB / 824K events | <div style="width: 100%; background-color: green;"></div> 1 | 1 |

Note that you may see the status change back to **1 Pending** and even to **1 Error**. However, your pipeline should still apply and eventually show as **1 Healthy**.



Check Your Data in Splunk Cloud

31. Log in to your Splunk Cloud instance (see [Access Your Splunk Cloud Instance](#)) and open the **Search & Reporting** app.

Run the following search over the **last 15 minutes** and verify that you now see events containing '**root**' in the '**admin**' index:

```
index="kyc_events" participant=hrobalin ← Remember to update the name!
```

| i | Time | Event |
|---|----------------------------|---|
| > | 10/30/25 7:33:01.702 PM | { [-] address: { [+] } } device_id: b436489dd2c2719002716cd836b3db96 documents: [[+]] } event_id: 86430fa3-20a6-45ed-aef5-40d0105225ad event_status: decision_approved event_type: kyc_approved pii: { [+] } } session_id: 2b25d587-a0c3-4318-89d9-eb05143b2fb2 source_ip: 136.4.114.246 timestamp: 2025-10-30T19:33:01.702290Z user_id: user_41836 } Show as raw text host = I-0862d8abf4e45fbf1.splunk.show:8088 participant = hrobalin source = kyc_simulator_v2 sourcetype = kyc:synthetic:event |

Check Your Data in Amazon S3 Destination

32. To access your Edge Processor, click "Splunk>" in the top-left corner. From the main screen, select "Edge Processors" from the left navigation pane.

Double-click your desired Edge Processor to open its dashboard, which displays pipelines running on it. Locate your specific pipeline at the bottom of the screen and expand its details by clicking the '>' icon.

Within the expanded pipeline view, you will see details regarding your Inbound Data, including its total size and the number of events. To the right, a breakdown illustrates how these events were processed and sent to various outbound destinations.

Observe that the Amazon S3 destination's 'Size/NumofEvents' matches the inbound data. This is expected, as the S3 destination is configured to archive a full copy of all events.

Conversely, the 'Size/NumofEvents' for the Splunk Cloud Destination is significantly lower. This reduction is intentional and results from pre-processing steps, such as filtering, designed to optimize the data sent to your Splunk Index.

| 1 pipeline | | | |
|--|-----------------------------------|-------------------------|--|
| View information about pipelines applied to the Edge Processor and the data that this Edge Processor sent in the last 15 minutes | | | |
| <input type="text"/> Enter search terms | | | |
| > | Name ↑ | Inbound data | Outbound data |
| | kyc_events_send_to_index_hrobalin | 25.41 MB / 23.7K events | 27.84 MB / 26.1K events 2.43 MB / 2.39K events 25.41 MB / 23.7K events |
| Destinations | | | |
| 2 destinations | | | |
| <input checked="" type="checkbox"/> default_splunk_cloud_destination | | | |
| <input checked="" type="checkbox"/> edge_s3_hrobalin | | | |

Exercise 5 - Buttercup Enterprises: AMERICAS Regulations (PCI DSS)

Description

As Buttercup Enterprises operates internationally, it must comply with additional regulations governing data security. The next requirement is adherence to the Payment Card Industry Data Security Standard (PCI DSS), which focuses on protecting payment card information.

To ensure PCI DSS compliance, you will configure the Edge Processor to identify, filter, and securely route payment card data. This process will involve setting up a dedicated pipeline to isolate sensitive cardholder information, archive it securely, and prepare it for audit or analytics while maintaining strict security controls.

PCI Pipeline Guide: AMERICAS Regulations (PCI DSS)

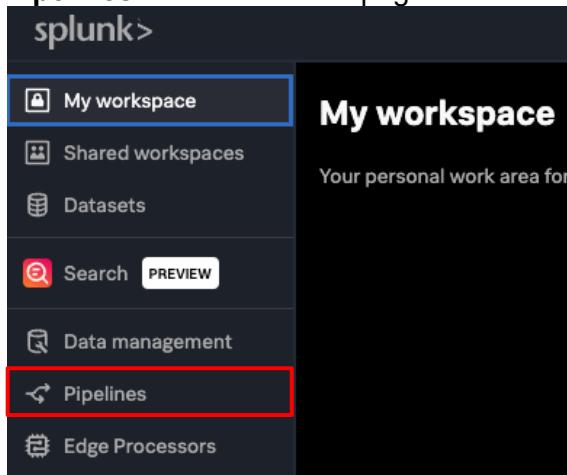
Objective: The Payment Card Industry Data Security Standard (PCI DSS) pipeline is designed to ensure compliance by securely handling payment card information. Its key objectives are to:

1. **Sanitize Sensitive Fields:** Redact cardholder names, and mask Primary Account Numbers (PANS) and account numbers according to PCI DSS requirements.
2. **Drop Non-Storable Fields:** Remove fields that are explicitly forbidden from being stored (e.g., CVV, track data, PIN).
3. **Route to Specific Index:** Send the processed and sanitized data to the `pci_events` index in Splunk Cloud.
4. **Identify Your Data:** Add a custom field to easily identify your events in a shared Splunk Cloud environment.

Steps

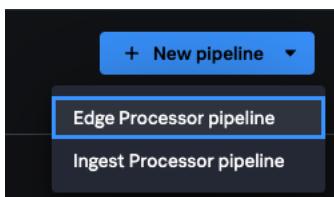
Create a New Pipeline

1. While logged in to the Edge Processor service (see [Access Edge Processor Service](#)) click on **Pipelines** on the left of the page.

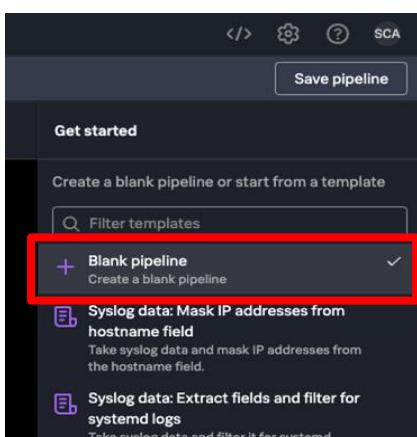




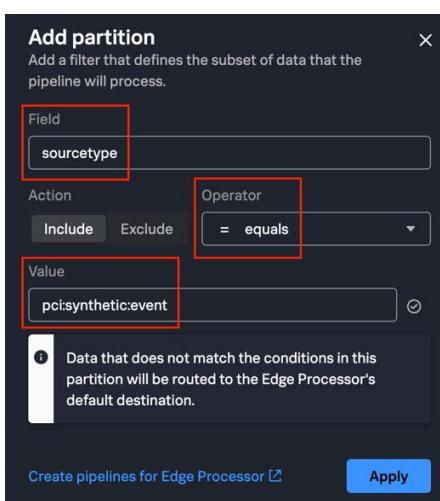
2. In the top right corner of the **Pipelines** page click on **+ New pipeline**. Select **Edge Processor pipeline**



3. On the right hand side of the **New pipeline** page under the **Get started** pane select on **Blank pipeline** and click on **Next**.



4. On the 'Define your pipeline's partition' pane, click the add button (+) next to the 'Partition' field. Configure the new partition rule by setting 'Field' to sourcetype, 'Action' to 'Keep', 'Operator' to '= equals', and set **pcisynthetic:event** as the 'Value'."



Note: Make sure you don't have an extra space at the end.

Click **Apply**, then click on **Next** at the bottom.



5. An **Add sample data** menu will appear on the right side. Upload the **pci_sample_events.jsonl** file that you extracted from the zip file earlier using the “**upload file...**” option at the bottom.

The screenshot shows the 'Add sample data' interface. At the top, there's a title bar with the text 'Add sample data'. Below it is a text input field with placeholder text 'Enter or upload sample data'. A large block of JSON sample data is pasted into this field. Below the input field, it says '17,978 / 30,000 characters'. Underneath the input field is a dashed box containing the text 'Drop your file anywhere or [upload file...](#)'. Below this is a section titled 'Sample data format' with two buttons: 'Raw text' (which has a red box around it) and 'CSV'. At the bottom are two buttons: 'Back' and 'Next' (which has a red box around it).

For the **Sample data format** select **Raw text** and click on **Next**.

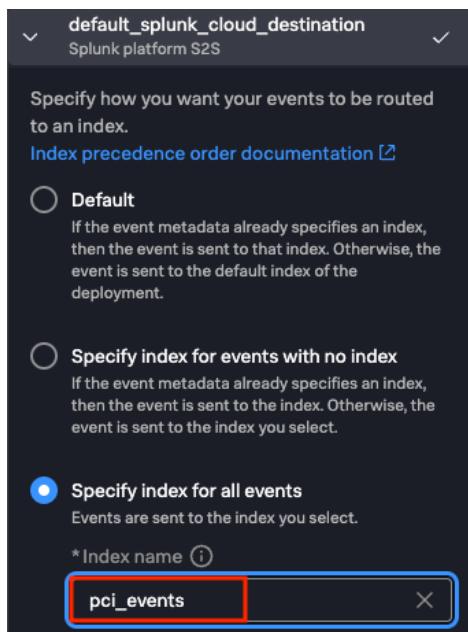
6. A “**Select a data destination**” menu will appear. Select the **default_splunk_cloud_destination** as your data destination and click on **Next**.

The screenshot shows the 'Select a data destination' interface. At the top, there's a title bar with the text 'Select a data destination'. Below it is a search bar with the placeholder text 'Filter datasets'. Underneath the search bar is a dropdown menu set to 'All kinds'. The main list contains two items: 'default_splunk_cloud_destination' (which has a red box around it) and 'edge_s3_hrobalin'. Both items have their descriptions ('Splunk platform S2S' and 'Amazon S3') visible below them.



Route Your Data to the 'pci_events' Index

7. A pane below will open, there you will select **Specify index for all events** and set the index to the **pci_events** from the drop down, as requested by the Buttercup security team, and click on **Done** at the bottom.



8. A screen like below will appear:

| Aa country | Aa city | Aa channel | # account_number | Aa src_ip |
|-----------------|---------------|------------|------------------|-----------------|
| 1 United States | Hendersonport | POS | 345641879709413 | 156.213.158.126 |
| 2 United States | Stephenland | POS | 6011915867210125 | 178.112.215.219 |
| 3 United States | Josefort | POS | 376744115964487 | 171.186.159.229 |
| 4 United States | Lake Justin | online | 4671179128311232 | 195.86.228.71 |
| 5 United States | Samuelland | ATM | 2288667889940245 | 115.229.144.7 |
| 6 United States | Dvianhaven | mobile | 2299640574813795 | 211.118.86.118 |

As you integrate additional actions into your pipeline, please insert them between the existing pipeline stages and the final index. It is crucial to maintain the order specified in this guide. Refer to the example below:

```
$pipeline = | from $source  
. . . ADD NEW ACTIONS HERE  
| eval index = "pci_events"  
| into $destination;
```

Define PII Sanitization (Masking, Hashing, Redaction)

9. Now we need to sanitize any of the data going into our Splunk Index. In our case since our data is a CSV (i.e. structured), we will use the [replace function](#) to replace data in our PII field:

First, we will redact the Full Name using the `eval` command.

```
// REDACT cardholder_name completely  
| eval _raw = json_set(_raw, "cardholder_name", "REDACTED_NAME")
```

Your SPL2 statement, should now look like this:

```
$pipeline =  
| from $source  
  
// REDACT cardholder_name completely  
| eval _raw = json_set(_raw, "cardholder_name", "REDACTED_NAME")  
  
// Set index for PCI events  
| eval index = "pci_events"  
| into $destination;
```

Reviewing Fields

10. Let's review the remaining fields to see if we need to mask, redact, or hash. We will use the PCI Standard stated [here](#) to guide us:

- **Primary Account Number (PAN)** – If storing, store only the first six and last four digits.
- **Account Number** – If storing, mask except for last four digits (This is not the PAN).
- **CCV** – Cannot be stored, must be dropped.
- **Track1 Data & Track2 Data** – Cannot be stored, must be dropped.
- **Pin, Pin Block** – Cannot be stored, must be dropped.

Masking PAN and Account Number

11. Let's first mask the PAN:

```
// The PAN according to PCI DSS Requirement 3.4 (e.g., first 6, last 4 digits visible)
| eval masked_pan = replace(_raw.pan, /(\d{6})(\d+)(\d{4})/, "$1*****$3")
| eval _raw=json_set(_raw, "pan", masked_pan)
```

Now let's mask the Account Number:

```
// Mask account_number except last 4 digits, this is not the same as PAN
| eval masked_acct_number = replace(_raw.account_number, /(\d{8})(\d{4})/, "XXXXXXXXXX$2")
| eval _raw=json_set(_raw, "account_number", masked_acct_number)
```

We have now both masked the PAN and Account Number.

12. We now need to drop our temporary fields created for the masking. We will do so below:

```
| fields - masked_pan, masked_acct_number
```

Filtering of Non-Storable Fields

13. Part of PCI DSS Requirement 3.4 is not storing specific fields; we will do so below with the following:

```
// Drop sensitive fields that cannot be stored
| eval _raw=json_delete(_raw, "cvv", "track1_data", "track2_data", "pin", "pin_block")
```

PCI SPL Pipeline:

14. As you develop your SPL code, understand that the order of transformations is critical. While our current requirements specify a particular sequence, in other contexts, this order will be driven by downstream data consumption or conditional routing. Keep this in mind to ensure optimal filtering, processing, and enrichment across all your data pipelines.

15. The following is an example of how your Edge Processor Pipeline should look:

```
$pipeline =
| from $source

// REDACT cardholder_name completely
| eval _raw = json_set(_raw, "cardholder_name", "REDACTED_NAME")

// Mask the PAN according to PCI DSS Requirement 3.4 (e.g., first 6, last 4 digits visible)
```

```

| eval masked_pan = replace(_raw.pan, /(\d{6})(\d+)(\d{4})/, "$1*****$3")
| eval _raw=json_set(_raw, "pan", masked_pan)

// Mask account_number except last 4 digits, this is not the same as PAN
| eval masked_acct_number = replace(_raw.account_number, /(\d{8})(\d{4})/, "XXXXXXXXXX$2")
| eval _raw=json_set(_raw, "account_number", masked_acct_number)
| fields - masked_pan, masked_acct_number

// Drop sensitive fields that cannot be stored
| eval _raw=json_delete(_raw, "cvv", "track1_data", "track2_data", "pin",
"pin_block")

// Set index for PCI events
| eval index = "pci_events"
| into $destination;

```

Preview Your Pipeline

16. Test your masking rule by clicking on the blue Preview pipeline button (in the top right corner of the screen. If your pipeline filter is correct, you should see some sample events containing “REDACTED_NAME”, mask values inside of pan and account_number respectively. We will also notice the lack of the fields that were dropped.

| Previewing \$pipeline 25 results | |
|----------------------------------|---|
| | </> _raw |
| 1 | {"transaction_id": "5183f92d-bb7a-45d5-bb8d-507ee048ac24", "account_number": "XXXXXXX2653", "country": "United States", "amount": 554.98, "city": "North Davi d", "expiry_date": "03/26", "channel": "ATM", "card_type": "visa", "cardholder_name": "REDACTED_NAME", "src_ip": "215.249.175.206", "device_info": "desktop_browser", "device_type": "PC", "os": "Windows", "browser": "Chrome", "version": "90.0.4480.89", "language": "en-US", "latency": 100, "latency_ms": 0.1, "latency_us": 100000}, {"transaction_id": "add19f6-a39b-4416-96ce-4bcc69f608dd", "account_number": "XXXXXXX4843", "country": "United States", "amount": 1455.88, "city": "Feliciaside", "expiry_date": "05/28", "channel": "POS", "card_type": "discover", "cardholder_name": "REDACTED_NAME", "src_ip": "222.111.6.126", "device_info": "mobile_android", "device_type": "Smartphone", "os": "Android", "browser": "Unknown", "version": "9.0", "language": "en-US", "latency": 100, "latency_ms": 0.1, "latency_us": 100000}, {"transaction_id": "1b04bfe4-bd40-48fd-84ae-bbb107cf1440", "account_number": "XXXXXXX1037", "country": "United States", "amount": 2044.28, "city": "Marcusport", "expiry_date": "11/30", "channel": "mobile", "card_type": "discover", "cardholder_name": "REDACTED_NAME", "src_ip": "49.58.75.229", "device_info": "mobile_ios", "device_type": "Smartphone", "os": "iOS", "browser": "Unknown", "version": "14.0", "language": "en-US", "latency": 100, "latency_ms": 0.1, "latency_us": 100000}, {"transaction_id": "d0904e7d-9994-448f-ab9e-e8090f46cb7d", "account_number": "XXXXXXX6081", "country": "United States", "amount": 3636.89, "city": "West Heatherville", "expiry_date": "01/26", "channel": "online", "card_type": "discover", "cardholder_name": "REDACTED_NAME", "src_ip": "212.241.244.154", "device_info": "mobile_browser", "device_type": "Mobile", "os": "Android", "browser": "Unknown", "version": "9.0", "language": "en-US", "latency": 100, "latency_ms": 0.1, "latency_us": 100000}], [{"text": "REDACTED_NAME"}] |
| 2 | {"transaction_id": "add19f6-a39b-4416-96ce-4bcc69f608dd", "account_number": "XXXXXXX4843", "country": "United States", "amount": 1455.88, "city": "Feliciaside", "expiry_date": "05/28", "channel": "POS", "card_type": "discover", "cardholder_name": "REDACTED_NAME", "src_ip": "222.111.6.126", "device_info": "mobile_android", "device_type": "Smartphone", "os": "Android", "browser": "Unknown", "version": "9.0", "language": "en-US", "latency": 100, "latency_ms": 0.1, "latency_us": 100000}, {"transaction_id": "1b04bfe4-bd40-48fd-84ae-bbb107cf1440", "account_number": "XXXXXXX1037", "country": "United States", "amount": 2044.28, "city": "Marcusport", "expiry_date": "11/30", "channel": "mobile", "card_type": "discover", "cardholder_name": "REDACTED_NAME", "src_ip": "49.58.75.229", "device_info": "mobile_ios", "device_type": "Smartphone", "os": "iOS", "browser": "Unknown", "version": "14.0", "language": "en-US", "latency": 100, "latency_ms": 0.1, "latency_us": 100000}, {"transaction_id": "d0904e7d-9994-448f-ab9e-e8090f46cb7d", "account_number": "XXXXXXX6081", "country": "United States", "amount": 3636.89, "city": "West Heatherville", "expiry_date": "01/26", "channel": "online", "card_type": "discover", "cardholder_name": "REDACTED_NAME", "src_ip": "212.241.244.154", "device_info": "mobile_browser", "device_type": "Mobile", "os": "Android", "browser": "Unknown", "version": "9.0", "language": "en-US", "latency": 100, "latency_ms": 0.1, "latency_us": 100000}], [{"text": "REDACTED_NAME"}] |
| 3 | {"transaction_id": "add19f6-a39b-4416-96ce-4bcc69f608dd", "account_number": "XXXXXXX4843", "country": "United States", "amount": 1455.88, "city": "Feliciaside", "expiry_date": "05/28", "channel": "POS", "card_type": "discover", "cardholder_name": "REDACTED_NAME", "src_ip": "222.111.6.126", "device_info": "mobile_android", "device_type": "Smartphone", "os": "Android", "browser": "Unknown", "version": "9.0", "language": "en-US", "latency": 100, "latency_ms": 0.1, "latency_us": 100000}, {"transaction_id": "1b04bfe4-bd40-48fd-84ae-bbb107cf1440", "account_number": "XXXXXXX1037", "country": "United States", "amount": 2044.28, "city": "Marcusport", "expiry_date": "11/30", "channel": "mobile", "card_type": "discover", "cardholder_name": "REDACTED_NAME", "src_ip": "49.58.75.229", "device_info": "mobile_ios", "device_type": "Smartphone", "os": "iOS", "browser": "Unknown", "version": "14.0", "language": "en-US", "latency": 100, "latency_ms": 0.1, "latency_us": 100000}, {"transaction_id": "d0904e7d-9994-448f-ab9e-e8090f46cb7d", "account_number": "XXXXXXX6081", "country": "United States", "amount": 3636.89, "city": "West Heatherville", "expiry_date": "01/26", "channel": "online", "card_type": "discover", "cardholder_name": "REDACTED_NAME", "src_ip": "212.241.244.154", "device_info": "mobile_browser", "device_type": "Mobile", "os": "Android", "browser": "Unknown", "version": "9.0", "language": "en-US", "latency": 100, "latency_ms": 0.1, "latency_us": 100000}], [{"text": "REDACTED_NAME"}] |
| 4 | {"transaction_id": "add19f6-a39b-4416-96ce-4bcc69f608dd", "account_number": "XXXXXXX4843", "country": "United States", "amount": 1455.88, "city": "Feliciaside", "expiry_date": "05/28", "channel": "POS", "card_type": "discover", "cardholder_name": "REDACTED_NAME", "src_ip": "222.111.6.126", "device_info": "mobile_android", "device_type": "Smartphone", "os": "Android", "browser": "Unknown", "version": "9.0", "language": "en-US", "latency": 100, "latency_ms": 0.1, "latency_us": 100000}, {"transaction_id": "1b04bfe4-bd40-48fd-84ae-bbb107cf1440", "account_number": "XXXXXXX1037", "country": "United States", "amount": 2044.28, "city": "Marcusport", "expiry_date": "11/30", "channel": "mobile", "card_type": "discover", "cardholder_name": "REDACTED_NAME", "src_ip": "49.58.75.229", "device_info": "mobile_ios", "device_type": "Smartphone", "os": "iOS", "browser": "Unknown", "version": "14.0", "language": "en-US", "latency": 100, "latency_ms": 0.1, "latency_us": 100000}, {"transaction_id": "d0904e7d-9994-448f-ab9e-e8090f46cb7d", "account_number": "XXXXXXX6081", "country": "United States", "amount": 3636.89, "city": "West Heatherville", "expiry_date": "01/26", "channel": "online", "card_type": "discover", "cardholder_name": "REDACTED_NAME", "src_ip": "212.241.244.154", "device_info": "mobile_browser", "device_type": "Mobile", "os": "Android", "browser": "Unknown", "version": "9.0", "language": "en-US", "latency": 100, "latency_ms": 0.1, "latency_us": 100000}], [{"text": "REDACTED_NAME"}] |



Sample Data

The results shown in the pipeline preview pane are from sample data that is uploaded when you configure your source types within the Edge Processor Service UI. They are not live events from your data stream.

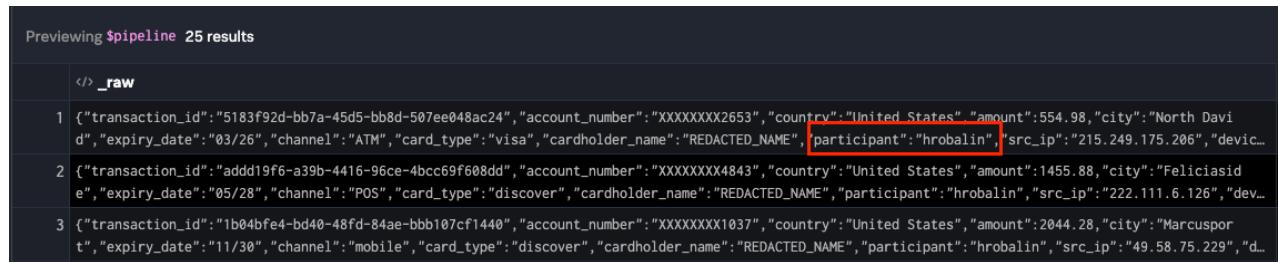
Identifying our Data

17. Finally, to help you identify data coming from your own pipeline when looking in your shared Splunk Cloud instance, add a custom indexed field to your events by adding an eval command to your pipeline to create a field called ‘participant’ with the value set to your own name:

```
$pipeline =
| from $source
...
| eval _raw = json_set(_raw, "participant", "hrobalin") ← Update with your own
name
| eval index = "pci_events"
| into $destination;
```

To verify that the new field will be created, click on the Preview button (▶) again. Since we're defining a new field=value pair as part of our pipeline, the new field will display in the Row preview table.

In our `_raw` field we can see our new value:

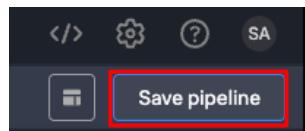
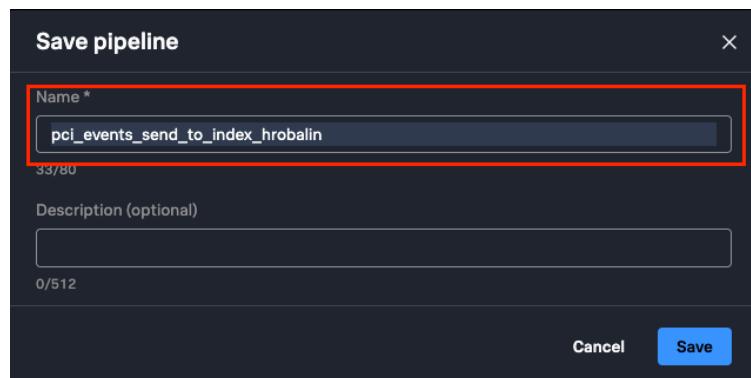


Previewing \$pipeline 25 results

| | <code></> _raw</code> |
|---|---|
| 1 | { "transaction_id": "5183f92d-bb7a-45d5-bb8d-507ee048ac24", "account_number": "XXXXXXXX2653", "country": "United States", "amount": 554.98, "city": "North Davi d", "expiry_date": "03/26", "channel": "ATM", "card_type": "visa", "cardholder_name": "REDACTED_NAME", participant": "hrobalin" , "src_ip": "215.249.175.206", "devic... |
| 2 | { "transaction_id": "add119f6-a39b-4416-96ce-4bcc69f608dd", "account_number": "XXXXXXXXX4843", "country": "United States", "amount": 1455.88, "city": "Feliciasid e", "expiry_date": "05/28", "channel": "POS", "card_type": "discover", "cardholder_name": "REDACTED_NAME", "participant": "hrobalin", "src_ip": "222.111.6.126", "dev... |
| 3 | { "transaction_id": "1b04bfe4-bd40-48fd-84ae-bbb107cf1440", "account_number": "XXXXXXXX1037", "country": "United States", "amount": 2044.28, "city": "Marcuspor t", "expiry_date": "11/30", "channel": "mobile", "card_type": "discover", "cardholder_name": "REDACTED_NAME", "participant": "hrobalin", "src_ip": "49.58.75.229", "d... |

Save Your Pipeline

- Save your pipeline by clicking on **Save pipeline** in the top right corner of the screen. Give your pipeline a suitable name, such as `pci_events_send_to_index_<your_name>` Be sure to include your name somewhere in the pipeline name to avoid confusion among those who are sharing the same environment as you!

Save pipeline

Name*

Description (optional)

0/512

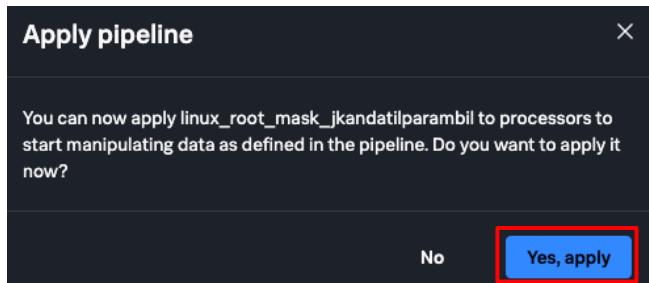
Cancel **Save**

- Click **Save** to save your pipeline.

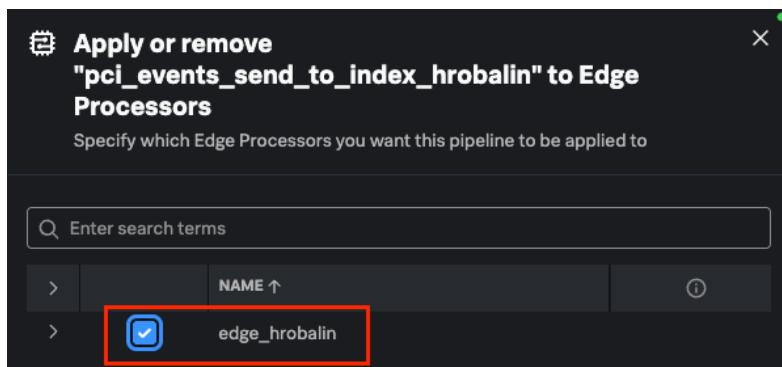


Apply the Pipeline to Your Edge Processor

19. Now an “Apply Pipeline” dialog box will appear, click on “Yes, apply”.



20. A pane called “**Apply or remove <pipeline-name> to Edge Processors**” will appear, Select the edge processor you created earlier (see [Add a New Edge Processor](#)) and click on **Save** at the bottom..



Confirm PCI Pipeline is applied

21. Make sure the PCI Events Pipeline is applied. Navigate to your Pipelines, you should see your pipeline name, and a status if Applied or Unapplied. See image below:

Check Your Data in Splunk Cloud

22. Log in to Splunk Cloud and open up the **Search & Reporting** app. Run the following search over the **last 4 hours** and verify that you now see the redacted events:

```
index="pci_events" participant="hrobalin" ← Remember to update the name!
```

| Time | Event |
|----------------------------|--|
| 10/31/25 8:12:35.744 PM | { [-] account_number: XXXXXXXX9657 amount: 2098.16 card_type: visa cardholder_name: REDACTED_NAME category: Retail channel: mobile city: Port Colleen country: United States currency: USD device_info: desktop_browser event_type: transaction_log_raw expiry_date: 11/26 pan: 401832*****2594 participant: hrobalin service_code: 206 src_ip: 182.52.249.223 status: pending timestamp: 2025-10-31T20:12:35.744738+00:00 transaction_class: domestic transaction_id: 9524a1ea-5f9f-4955-a88f-1db23df5d0a6 vendor: Stevens, Johnson and Rodgers } |

Exercise 6 – Buttercup Enterprises: EMEA Regulations (DORA)

Description

Operating in the EMEA region, Buttercup Enterprises must also comply with the Digital Operational Resilience Act (DORA). This regulation is designed to ensure the operational resilience of financial institutions by strengthening requirements for managing and monitoring ICT (Information and Communications Technology) risks.

To support DORA compliance, you will configure the Edge Processor to monitor and manage data related to operational resilience. This involves creating a pipeline that can filter, archive, and route relevant ICT risk and incident data, enabling secure storage and timely analysis in accordance with DORA requirements.

DORA Pipeline Guide: EMEA Regulations (Digital Operational Resilience Act)

Objective: The Digital Operational Resilience Act (DORA) pipeline aims to ensure compliance by monitoring and managing ICT risks. Its key objectives are to:

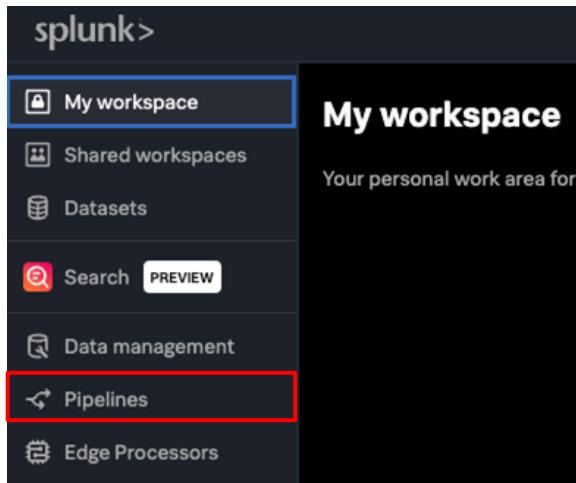
1. **Archive Raw Data Conditionally:** Send an unmodified copy of the raw data to Amazon S3 for archival.
2. **Optimize Event Size:** Drop the `_raw` field after archival since its contents are represented by other extracted fields.
3. **Pseudonymize/Anonymize Data:** Mask IP addresses and pseudonymize User IDs using a salted hash for enhanced privacy while allowing correlation if needed.
4. **Route Based on Severity:** Direct events with "ERROR" or "CRITICAL" severity to a specific index (`dora_errors_events`) and "WARNING" severity events to another (`dora_warning_events`).
5. **Minimize Data:** Remove unnecessary fields (`incident_description`, `description`, `transaction_id`, `session_id`) to reduce data volume for the default stream.
6. **Route to Default Index:** Send remaining events to the `dora_events` index.
7. **Identify Your Data:** Add a custom field to easily identify your events in a shared Splunk Cloud environment.



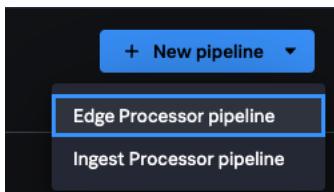
Steps

Create a New Pipeline

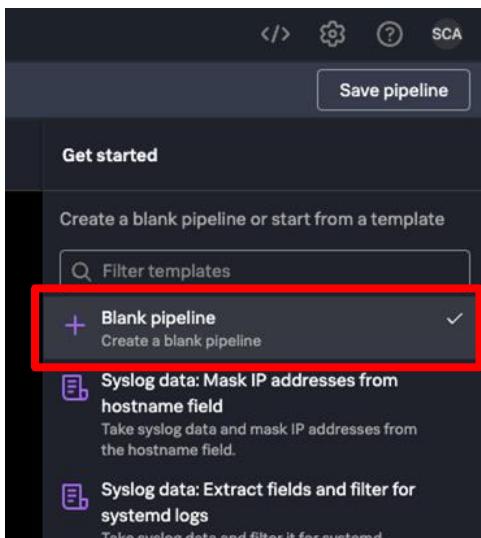
1. While logged in to the Edge Processor service (see [Access Edge Processor Service](#)) click on **Pipelines** on the left of the page.



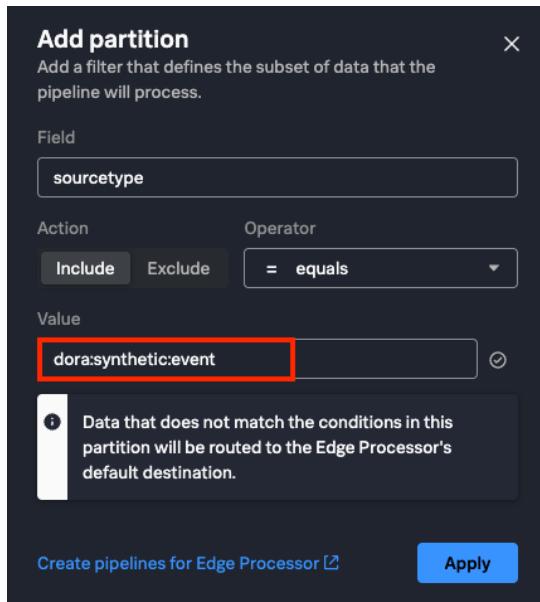
2. In the top right corner of the **Pipelines** page click on **+ New pipeline**. Select **Edge Processor pipeline**



3. On the right-hand side of the **New pipeline** page under the **Get started** pane select on **Blank pipeline** and click on **Next**.

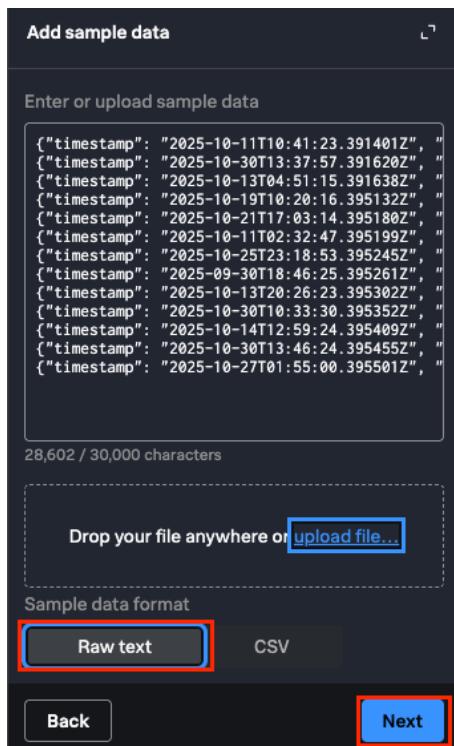


4. On the **Define your pipeline's partition** pane click on the (+) button next to the Partition field and select the value **sourcetype** for Field, select “Keep” for Action, “= equals” for Operator and **dora:synthetic:event** for value on the dropdown list click on and click **Apply**.



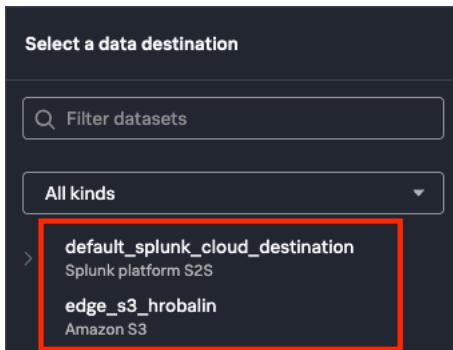
Click on **Next** at the bottom.

5. An **Add sample data** menu will appear on the right side. Upload the [dora_sample_events.jsonl](#) file that you extracted from the zip file earlier using the “**upload file...**” option at the bottom.



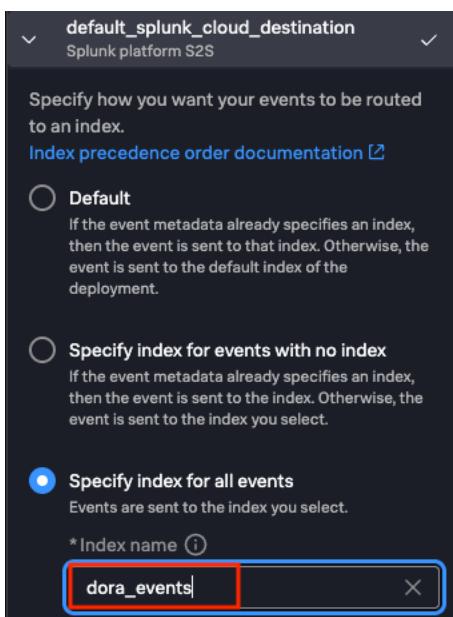
For the **Sample data format** select **Raw text** and click on **Next**.

6. A “**Select a data destination**” menu will appear. Select the **default_splunk_cloud_destination** as your data destination.



Route Your Data to the **dora_events** Index

7. A pane below will open, there you will select **Specify index for all events** and set the index to the **dora_events** from the drop down, as requested by the Buttercup security team, and click on **Done** at the bottom.





8. A screen like below will appear:

| _raw | transaction_id | # num_clients_a... | impacted_s... |
|---|----------------|--------------------|------------------------|
| 1 {"timestamp": "2025-08-20T13:31:14.970387Z", "event_timestamp": "2025-08-20T13:31:14.970387Z", "log_level": "INFO", "severity_level": "INFO", "index": "dora_events", "source": "aws-cloudwatch-logs", "type": "aws-cloudwatch-logs", "version": "1"}, {"timestamp": "2025-08-27T15:31:56.970594Z", "event_timestamp": "2025-08-27T15:31:56.970594Z", "log_level": "INFO", "severity_level": "INFO", "index": "dora_events", "source": "aws-cloudwatch-logs", "type": "aws-cloudwatch-logs", "version": "1"}, {"timestamp": "2025-08-17T23:06:47.974147Z", "event_timestamp": "2025-08-17T23:06:47.974147Z", "log_level": "INFO", "severity_level": "INFO", "index": "dora_events", "source": "aws-cloudwatch-logs", "type": "aws-cloudwatch-logs", "version": "1"}, {"timestamp": "2025-08-30T08:23:48.974219Z", "event_timestamp": "2025-08-30T08:23:48.974219Z", "log_level": "INFO", "severity_level": "INFO", "index": "dora_events", "source": "aws-cloudwatch-logs", "type": "aws-cloudwatch-logs", "version": "1"}] | | | |
| | | | 0 fields hidden |
| | | | transaction_id |
| | | | # num_clients_affected |
| | | | impacted_services |
| | | | log_level |
| | | | index |
| | | | incident_description |

As you integrate additional actions into your pipeline, please insert them between the existing pipeline stages and the final index. It is crucial to maintain the order specified in this guide. Refer to the example below:

```
$pipeline = | from $source
. . . ADD NEW ACTIONS HERE
| eval index = "dora_events"
| into $destination;
```

Define Conditional Raw Data Archival

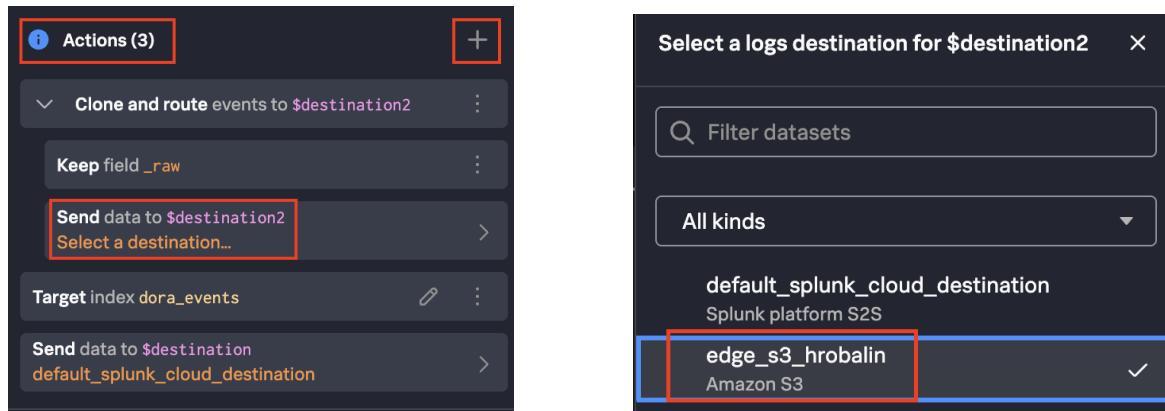
9. Certain DORA requirements require that we archive the data before any processing. This is when we can leverage Edge Processor Pipelines to route the data to an archival location such as Amazon S3, and one to be processed before landing on Splunk.

The `_raw` field in our data provides a comprehensive, unadulterated representation of each event, incorporating all other extracted fields. This makes it a perfect candidate for efficient routing to our long-term archive.

This routing can be configured either from the Actions menu or directly within an editor. For this demonstration, we will utilize the pipeline editor. Meaning we will write the SPL to define that we will have another route for our data. This is done with the following SPL below:

```
// Initial Archive Routing
| thru
[
  | fields _raw
  | into $destination2
]
```

Now that we have defined in SPL that we want to route data, we must use the Actions Menu on the right to configure the route to the proper destination: Amazon S3



The image shows two screenshots from the Splunk interface. On the left, the 'Actions (3)' menu is open, showing three options: 'Clone and route events to \$destination2', 'Keep field _raw', and 'Send data to \$destination2'. The 'Send data to \$destination2' option is highlighted with a red box. On the right, a modal window titled 'Select a logs destination for \$destination2' is displayed. It includes a search bar 'Filter datasets', a dropdown 'All kinds', and a list of destinations. The 'edge_s3_hrobalin' destination is selected and highlighted with a red box.

Now the archival routing now configured, we can optimize event size by dropping the `_raw` field, as its contents are already represented by other extracted fields. This can be achieved using the following SPL:

```
// Drop _raw field to reduce data size after initial archive routing
| eval _raw=json_delete(_raw, "_raw")
```

Make sure to add the above SPL as well.

Define Pseudonymization/Anonymization

10. DORA regulations necessitate the anonymization of potentially identifiable data. Our review identified the following fields as candidates for anonymization:

- IP Address
- User ID

For IP Addresses, we can either mask an octet or hash the complete address. The selection between these methods will depend on specific data usage, display, and storage requirements. Both options will be presented to demonstrate the flexibility and power available for implementing diverse anonymization strategies.

11. Masking IP Address SPL:

```
// Anonymize IP address (e.g., mask last octet or hash)
| eval masked_ip_address = replace(_raw.ip_address, /\.\d+$/, ".XXX")
```

```
| eval _raw=json_set(_raw, "ip_address", masked_ip_address)
```

12. Next, we will address the User ID field. We will anonymize it by hashing, incorporating a unique and consistent salt to enable correlation across related events when necessary.

13. User ID Pseudonymization via Hashing

```
// Pseudonymize user ID (e.g., using a salted hash if correlation is ever
needed under strict controls)
| eval hashed_user_id = if(isnotnull(_raw.user_id), sha256(_raw.user_id +
"dora_secret_salt"), null)
| eval _raw=json_set(_raw, "user_id", hashed_user_id)
```

14. Lastly, we need to drop the temporary fields created.

```
//Remove Temp Fields
| fields - masked_ip_address, hashed_user_id
```

15. Check that your fields are masked and hashed correctly by previewing the pipeline (▶). You should see the columns displayed in the preview pane:

```
Previewing $pipeline $destination ▶ 11 results

</> _raw
1 {"num_clients_affected":0,"impacted_services":["fraud_detection"],"log_level":"INFO","ip_address":"000.000.000.XXX","event_categ
ory":"ICT_Operational_Event","event_timestamp":"2025-10-11T10:41:23.391401Z","severity_level":"INFO","participant":"hrobalin","d
ata_loss_details":"No data loss","is_reportable_dora_incident":false,"component":"fraud_detection","event_type":"operational_eve
nt","user_id":"bdfde7fbb58c7149e84f60d767ef8b3478f45e4066199d0d17fc626e5bf41ec8"} "ict_system_id": "ICT-2424", "dora_incident_class
ification": "Operational Event", "source_hostname": "srv-fraud-detection-1.prod.example.com", "dora_compliance_tag": "DORA_v1.0_lo
g", "timestamp": "2025-10-11T10:41:23.391401Z"}
```

Define Index Destination based on Severity Level of Event

16. We know have archived a raw unmodified copy of our data, masked, and hashed sensitive data. Our team now is asking us to make sure to send specific events messages to specific index. The team would like us to filter based on Severity Levels: **Error**, **Critical**, and **Warning**.

Index: dora_error_events -> ERROR or CRITICAL

Index: dora_warning_events -> WARNING

17. Next, we will add the **Routing** as specified above. (Note: You can add a **Route** via the UI, in our use case it is better to go via editing the pipeline.)

First we will start by adding our Route Import function to the top of our Pipeline, meaning this will be our new line 1:

```
import route from /splunk.ingest.commands
```

18. Now we can begin to configure our **Route** so that our pipeline match both **ERROR** and **CRITICAL** in the severity_level field.

We will also need to make sure that we route these events to a specific index,

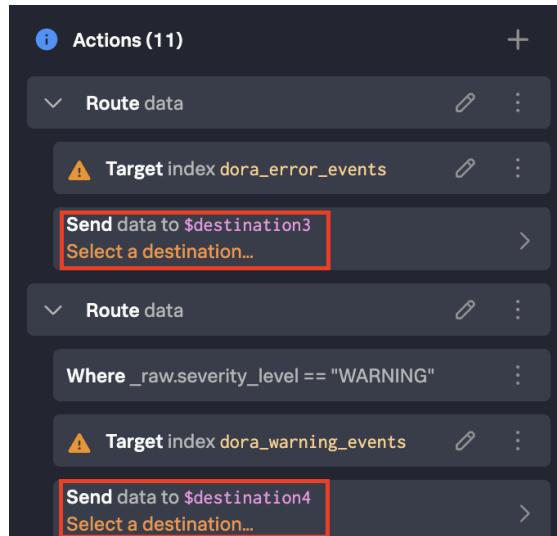
See the following for an example that creates a new route in SPL. We will use this.

```
// Filter for Error and Critical Severity Levels
| route _raw.severity_level == "ERROR" OR _raw.severity_level =
"CRITICAL",
[
    | eval index= "
    | into $destination3
]
```

19. To configure Warning Severity Level, we can use the below SPL:

```
// Filter for Warning Severity Levels
| route _raw.severity_level == "WARNING",
[
    | eval index="dora_warning_events"
    | into $destination4
]
```

20. Before we can preview our changes, we need to define in the pipeline setting which Amazon S3 Bucket will be \$destination3 and \$destination4.



21. Click **Select a destination**

22. Select the Amazon S3 Endpoint for our use case: **edge_s3_username**



Select a data destination

Filter datasets

All kinds

default_splunk_cloud_destination
Splunk platform S2S

edge_s3_hrobalin
Amazon S3

23. Click **Apply** at the bottom right.

24. **Actions** should now look like this, where \$destination3 is an S3 Destination:

Actions (3)

Clone and route events to edge_s3_hrobalin

Send data to \$destination2
edge_s3_hrobalin

Target index kyc_events

Send data to \$destination
default_splunk_cloud_destination

25. Repeat for \$destination4.

26. Let's verify that each destination receives only its intended events. We can do this by previewing the pipeline (▶). Let's filter by destination drop down on the preview panel top left:

If we select \$destination3 which is filtering for ERROR and CRITICAL, we can see our events are indeed Error or Critical.

Previewing \$pipeline \$destination3 ⓘ 2 results

| | <code></> _raw</code> |
|---|---|
| 1 | <pre>{"transaction_id": "", "num_clients_affected": 4742, "impacted_services": ["backup_restore_system", "fraud_detection"], "log_level": "ERROR", "description": "Unauthorized access in backup_restore_system. Investigation ID: 1c8f79be-e1b5-4a7b-9348-6d66fa95c07a", "incident_description": "Unauthorized access in backup_restore_system. Investigation ID: 1c8f79be-e1b5-4a7b-9348-6d66fa95c07a", "session_id": "", "ip_address": "000.000.000.XXX", "event_category": "ICT_Security_Incident", "event_timestamp": "2025-10-21T17:03:14.395180Z", "severity_level": "ERROR", "data_loss_details": "No data loss", "is_reportable_dora_incident": true, "component": "backup_restore_system", "event_type": "incident", "user_id": "bdfde7fbb58c7149e84f60d767ef8b3478f45e4066199d0d17fc626e5bf41ec8", "ict_system_id": "ICT-7227", "dora_incident_classification": "Security Breach", "source_hostname": "srv-backup-restore-system-1.prod.example.com", "dora_compliance_tag": "DORA_v1.0_log", "timestamp": "2025-10-21T17:03:14.395180Z"} {"transaction_id": "", "num_clients_affected": 526, "impacted_services": ["incident_response_platform", "fraud_detection"], "log_level": "CRITICAL", "description": "Unauthorized access in incident_response_platform. Investigation ID: 6992a374-3aef-430b-af9d-bbc4ccf8ded5", "incident_description": "Unauthorized access in incident_response_platform. Investigation ID: 6992a374-3aef-430b-af9d-bbc4ccf8ded5", "session_id": "", "in_address": "000.000.000.XXX", "event_category": "ICT_Security_Incident", "event_timestamp": "2025-10-25T23:18:53.395245Z", "severity_level": "CRITICAL", "data_loss_details": "full data loss", "is_reportable_dora_incident": false, "component": "incident_response_platform", "event_type": "incident", "user_id": "bdfde7fbb58c7149e84f60d767ef8b3478f45e4066199d0d17fc626e5bf41ec8", "ict_system_id": "ICT-2084", "dora_incident_classification": "Third-Party Failure", "source_hostname": "srv-incident-response-platform-3.prod.example.com", "dora_compliance_tag": "DORA_v1.0_log", "timestamp": "2025-10-25T23:18:53.395245Z"}</pre> |

If we select \$destination4 which is filtering for WARNING, we can see our events are indeed Warning.

Previewing \$pipeline \$destination4 ⓘ 1 results

| | <code></> _raw</code> |
|---|---|
| 1 | <pre>{"transaction_id": "", "num_clients_affected": 27, "impacted_services": ["auth_service"], "log_level": "WARNING", "description": "New admin user created on auth_service by user karen.murphy from 129.153.198.180.", "incident_description": "New admin user created on auth_service by user karen.murphy from 129.153.198.180.", "session_id": "83426dd2-682a-4507-ad12-7cb3adb5b0f8", "ip_address": "129.153.198.XX", "event_category": "ICT_Operational_Event", "event_timestamp": "2025-10-13T04:51:15.391638Z", "severity_level": "WARNING", "data_loss_details": "No data loss", "is_reportable_dora_incident": false, "component": "auth_service", "event_type": "security_event", "user_id": "181e056597f980ba30cb75a1f74c4e30ac0e8f3f6664408fafd26cae74e6ac92", "ict_system_id": "ICT-6514", "dora_incident_classification": "Security Event", "source_hostname": "srv-auth-service-3.prod.example.com", "dora_compliance_tag": "DORA_v1.0_log", "timestamp": "2025-10-13T04:51:15.391638Z"}</pre> |

Now that we have verified our pipeline is working as expected, let's proceed to the next step.

Define Data Minimization by Field Removal

27. Now let's minimize the data, we can do this by removing fields not required.

In our case we will remove the following:

- incident_description, description, transaction_id, session_id

We can do this by adding the following SPL:

```
// Remove Unnecessary Fields for Info Events
| eval _raw = json_delete(_raw, "incident_description", "description",
"transaction_id", "session_id")
```

28. Checking the Pipeline preview for \$destination, you will notice that indeed we have dropped these fields.

Add a Custom Indexed Field to Your Data

29. Finally, add a ‘**participant**’ field to your events to help you identify them in Splunk Cloud:

```
$pipeline =
| from $source
...
| eval _raw = json_set(_raw, "participant", "hrobalin") ← Update with your
  own name
| index = "dora_events"
| into $destination;
```

Preview Your Pipeline

30. Test your update by clicking on the blue Preview pipeline button (▶) in the top right corner of the screen. You should see a preview of your events in the center of the screen showing the IP addresses redacted. You should also see your new ‘**participant**’ field.

| Event ID | Type | Participant | Timestamp |
|----------|-----------------------|-------------|-----------------------------|
| 1 | ICT_Operational_Event | hrobalin | 2025-10-11T10:41:23.391401Z |
| 2 | DORA_v1.0_log | hrobalin | 2025-10-11T10:41:23.391401Z |

DORA Pipeline

Your DORA Pipeline SPL should look like the below:

```
import route from /splunk.ingest.commands

$pipeline = | from $source

// Initial Archive Routing
| thru
[
    | eval _raw = json_extract(_raw, "_raw")
    | into $destination2
]

// Drop _raw field to reduce data size after initial archive routing
| eval _raw=json_delete(_raw, "_raw")
```

```

// Anonymize IP address (e.g., mask last octet)
| eval masked_ip_address = replace(_raw.ip_address, /\.\d+$/, ".XXX")
| eval _raw=json_set(_raw, "ip_address", masked_ip_address)

// Pseudonymize user ID (e.g., using a salted hash if correlation is ever
needed under strict controls)
| eval hashed_user_id = if(isnotnull(_raw.user_id), sha256(_raw.user_id +
"dora_secret_salt"), null)
| eval _raw=json_set(_raw, "user_id", hashed_user_id)

| fields - masked_ip_address, hashed_user_id

// Filter for Error and Critical Severity Levels
| route _raw.severity_level == "ERROR" OR _raw.severity_level =
"CRITICAL",
[
    | eval index="dora_errors_events"
    | into $destination3
]

// Filter for Warning Severity Levels
| route _raw.severity_level == "WARNING",
[
    | eval index="dora_warning_events"
    | into $destination4
]

// Remove Unnecessary Fields for Info Events
| eval _raw=json_delete(_raw, "incident_description", "description",
"transaction_id", "session_id")

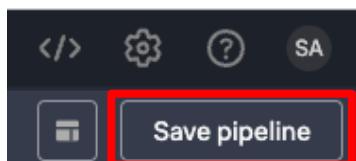
| eval _raw = json_set(_raw, "participant", "hrobalin")

// Finalize and Route Info Events
| eval index = "dora_events"
| into $destination;

```

Save Your Pipeline

31. Save your pipeline by clicking on **Save pipeline** in the top right corner of the screen. Give your pipeline a suitable name, such as **dora_events_send_to_index_<yourName>**. Remember to add your name or initials at the end to help identify it.





Save pipeline

Name *

34/80

Description (optional)

0/512

Cancel Save

Apply the Pipeline to Your Edge Processor

32. Now an “Apply Pipeline” dialog box will appear, click on “**Yes, apply**”.

Apply pipeline

You can now apply dora_events_send_to_index_hrobalin to processors to start manipulating data as defined in the pipeline. Do you want to apply it now?

No Yes, apply

33. A pane called “**Apply or remove <pipeline-name> to Edge Processors**” will appear, Select the edge processor you created earlier (see [Add a New Edge Processor](#)) and click on **Save** at the bottom.

Apply or remove

"dora_events_send_to_index_hrobalin" to Edge Processors

Specify which Edge Processors you want this pipeline to be applied to

Enter search terms

| | NAME ↑ | ⋮ |
|---|---------------|-------------------------------------|
| > | edge_hrobalin | <input checked="" type="checkbox"/> |

Confirm DORA Pipeline is applied

34. Make sure the DORA Events Pipeline is applied. Navigate to your Pipelines, you should see your pipeline name, and a status if Applied or Unapplied.

Check Your Data in Splunk Cloud

35. Log in to Splunk Cloud and open up the **Search & Reporting** app. Run the following search over the **last 15 minutes** and verify that you now see the redacted events.

```
index=dora_events participant="hrobalin" ← Update the name!
```

| i | Time | Event |
|---|------------------------|---|
| > | 11/4/25 8:46:00.756 AM | <pre>{ component: user_management_portal data_loss_details: No data loss dora_compliance_tag: DORA_v1.0_log dora_incident_classification: Operational Event event_category: ICT_Operational_Event event_timestamp: 2025-11-03T23:46:00.756953Z event_type: operational_event ict_system_id: ICT-4297 impacted_services: [{ ip_address: 69.128.63.XXX is_reportable_dora_incident: false log_level: INFO num_clients_affected: 0 participant: hrobalin ← Update the name! severity_level: INFO source_hostname: srv-user-management-portal-5.prod.example.com timestamp: 2025-11-03T23:46:00.756953Z user_id: 4ae67952e0648da9dfd4ede12631c7db8c8158a0e82c47ce681abc1b73115ae }] Show as raw text } host = srv-user-management-portal-5.prod.example.com source = dora_simulator sourcetype = dora:synthetic:event</pre> |

Note that since our search results will only contain those events that match our IP address range you may need to wait up to a couple of minutes for events matching that range to arrive at Splunk. If you don't get any results immediately try re-running the search after a minute or two.

Exercise 7 – Buttercup Enterprises: APJC Regulations (RMiT)

Description

In the APJC region, Buttercup Enterprises must comply with the Risk Management in Technology (RMiT) guidelines, established to enhance the technology risk management practices of financial institutions.

To address RMiT requirements, you will set up the Edge Processor to capture, filter, and securely route technology risk-related data. This includes creating a dedicated pipeline for monitoring, archiving, and analyzing data related to technology incidents and controls, ensuring compliance with RMiT standards.

RMiT Pipeline Guide: APJC Regulations (Risk Management in Technology)

Objective: The Risk Management in Technology (RMiT) pipeline ensures compliance by capturing, filtering, and securely routing technology risk-related data. Its key objectives are to:

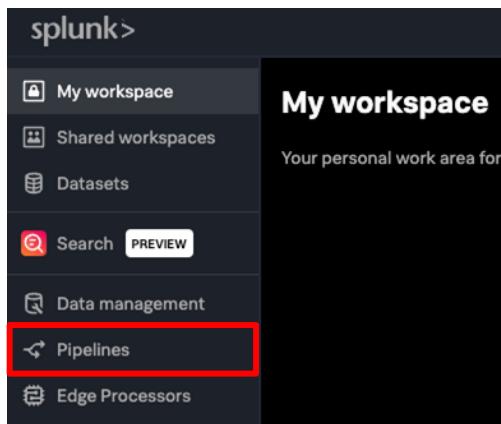
1. **Enrich Data with Lookups:** Enhance events with asset inventory information (criticality, location, service impact) using a lookup file.
2. **Define Risk Levels:** Assign a risk_level (High, Medium, Low) based on the criticality from the lookup.
3. **Mask Sensitive Operational Details:** Mask IP addresses, pseudonymize user_id, mask reporting_user_or_process, and redact event_description if it contains PII.
4. **Filter Non-Actionable Logs:** Retain only relevant events based on status and criticality.
5. **Route Based on Event Type:** Direct events to different Splunk indexes (rmit_cyber_events, rmit_change_events, rmit_tprm_events) based on their event_type.
6. **Minimize Data:** Remove low-value or empty fields from the default stream.
7. **Identify Your Data:** Add a custom field to easily identify your events in a shared Splunk Cloud environment.



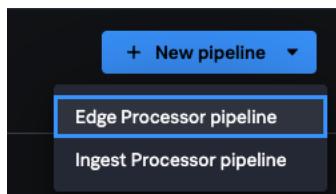
Steps

Create a New Pipeline

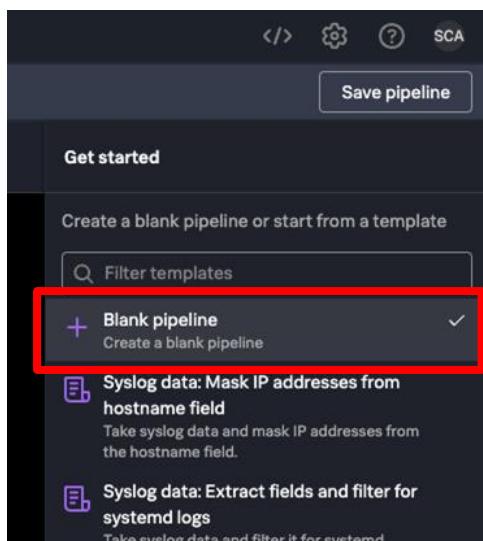
1. While logged in to the Edge Processor service (see [Access Edge Processor Service](#)) click on **Pipelines** on the left of the page.



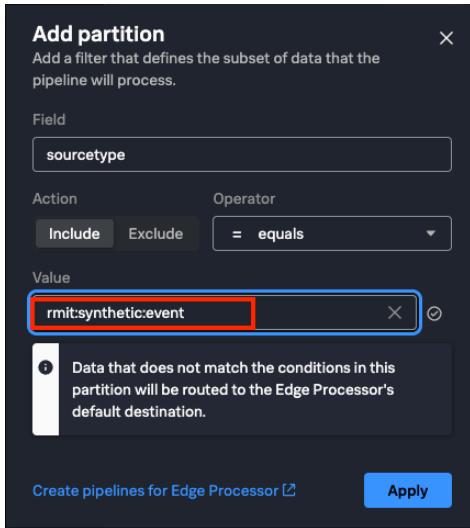
2. In the top right corner of the **Pipelines** page click on **+ New pipeline**. Select **Edge Processor pipeline**



3. On the right hand side of the **New pipeline** page under the **Get started** pane select on **Blank pipeline** and click on **Next**.



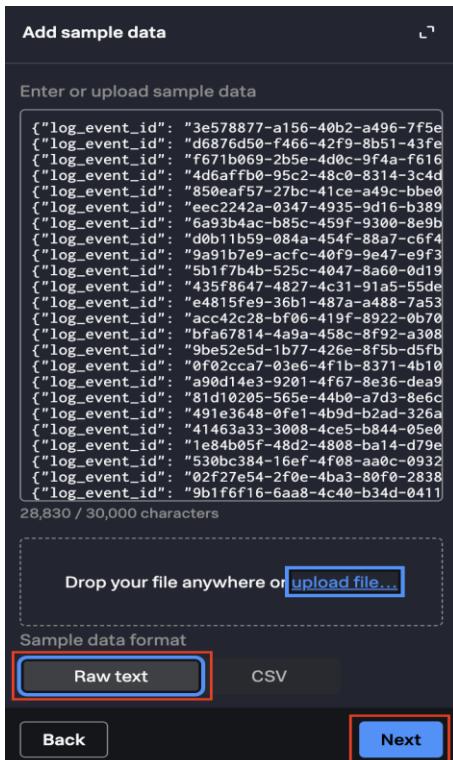
4. On the **Define your pipeline's partition** pane click on the (+) button next to the Partition field and select the value **sourcetype** for Field, select “Keep” for Action, “= equals” for Operator and **rmit:synthetic:event** for value on the dropdown list click on and click **Apply**.



Note: Please make sure to check that there is no extra space at the end.

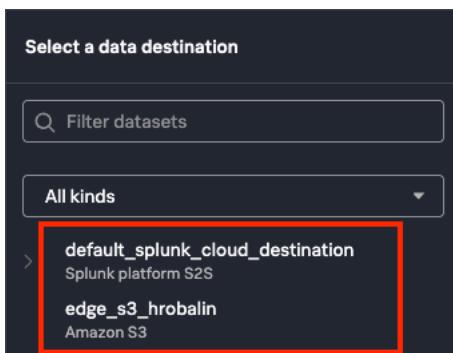
Click on **Next** at the bottom.

5. An **Add sample data** menu will appear on the right side. Upload the **rmit_sample_events.jsonl** file that you extracted from the zip file earlier using the “**upload file...**” option at the bottom.



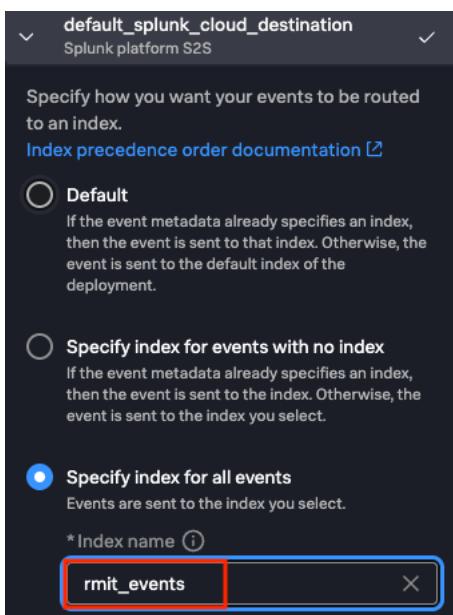
For the **Sample data format** select **Raw text** and click on **Next**.

6. A “Select a data destination” menu will appear. Select the **default_splunk_cloud_destination** as your data destination and click on **Next**.



Route Your Data to the `rmit_events` Index

7. A pane below will open, there you will select **Specify index for all events** and set the index to the **rmit_events** from the drop down, as requested by the Buttercup security team, and click on **Done** at the bottom.



8. A screen like below will appear:



splunk> Pipelines / New pipeline Edge Processor

```
+ 1 /*  
2 A valid SPL2 statement for a pipeline must start with "$pipeline", and include "from $source"  
3 and "into $destination".  
4 */  
5 $pipeline = | from $source  
6 | eval index = "rmit_events" | into $destination;
```

Sample Preview "pipeline" (Command + Enter)

Partition (1)

Actions (2)

Target index rmit_events

Send data to \$destination default_splunk_cloud_destination

Fields (3)

Filter fields

Name

_raw

index

sourcetype

| | Aa _raw | Aa index | Aa sourcetype |
|---|---|----------|---------------|
| 1 | {"log_event_id": "3e578877-a156-40b2-a496-7f5e7ceec1d0", "event_timestamp_utc": "2025-10-31T13:47:46.732034Z", "event_timestamp": "2025-10-31T13:47:46.732034Z", "financial_institution_id": "AmBank", "log_level": "INFO", "log_message": "Success", "log_source": "System", "log_time": "2025-10-31T13:47:46.732034Z", "log_type": "System", "log_version": "1.0", "log_version_minor": "0", "log_version_patch": "0", "log_version_major": "1"}, {"log_event_id": "d6876d50-f466-42f9-8b51-43fe3117cede", "event_timestamp_utc": "2025-10-31T13:47:46.732034Z", "event_timestamp": "2025-10-31T13:47:46.732034Z", "financial_institution_id": "RHB_Bank", "log_level": "INFO", "log_message": "Success", "log_source": "System", "log_time": "2025-10-31T13:47:46.732034Z", "log_type": "System", "log_version": "1.0", "log_version_minor": "0", "log_version_patch": "0", "log_version_major": "1"}, {"log_event_id": "f671b069-2b5e-4d0c-9f4a-f61651587e9b", "event_timestamp_utc": "2025-10-31T13:47:46.732034Z", "event_timestamp": "2025-10-31T13:47:46.732034Z", "financial_institution_id": "Hong_Kong_Bank", "log_level": "INFO", "log_message": "Success", "log_source": "System", "log_time": "2025-10-31T13:47:46.732034Z", "log_type": "System", "log_version": "1.0", "log_version_minor": "0", "log_version_patch": "0", "log_version_major": "1"}, {"log_event_id": "4d6afffb-95c2-48c8-8314-5cd1d317787", "event_timestamp_utc": "2025-10-31T13:47:46.732034Z", "event_timestamp": "2025-10-31T13:47:46.732034Z", "financial_institution_id": "OCBC_Bank", "log_level": "INFO", "log_message": "Success", "log_source": "System", "log_time": "2025-10-31T13:47:46.732034Z", "log_type": "System", "log_version": "1.0", "log_version_minor": "0", "log_version_patch": "0", "log_version_major": "1"}] | | |

As you integrate additional actions into your pipeline, please insert them between the existing pipeline stages and the final index. It is crucial to maintain the order specified in this guide. Refer to the example below:

```
$pipeline = | from $source
. . . ADD NEW ACTIONS HERE
| eval index = "rmit_events"
| into $destination;
```

Configure Lookup for Events

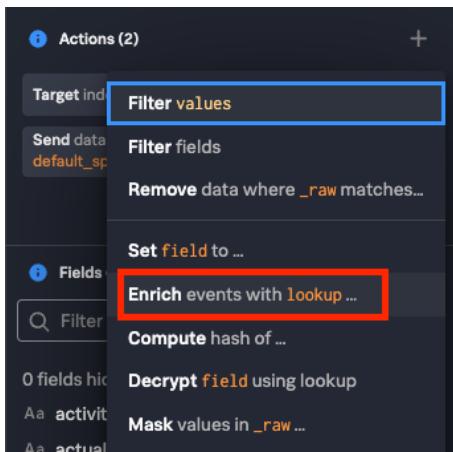
9. We want to first start by enriching this data, it will provide us more context to an event coming in as well, as allow us to do advance routing based on enriched data.

To accomplish this, we will import a lookup using Edge Processing, through the **Actions** menu on the right side. Press the Plus button on the right side.

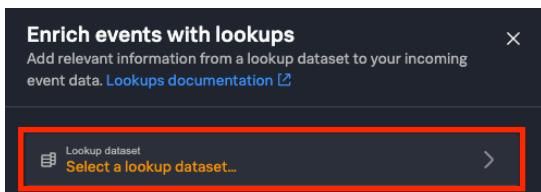
- Actions (2)
 - Target index `rmit_events` 🔗 ⚙️ ⋮
 - Send data to \$destination
`default_splunk_cloud_destination` >



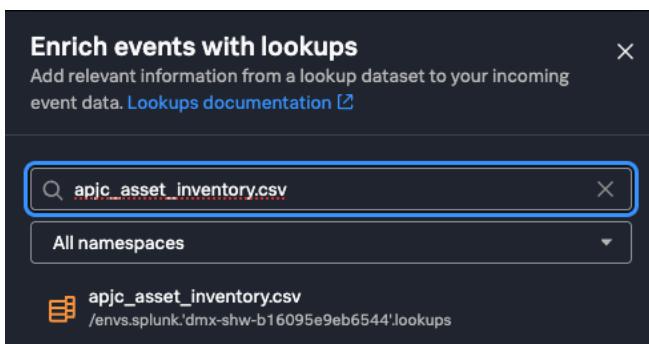
10. We will select: **Enrich events with lookup**



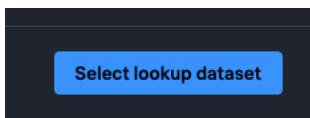
11. Now, we will click on **Select a Lookup dataset**



12. In our example we will search for **apjc_asset_inventory.csv**, and **select** the option that comes up:



13. Make sure to press the button on the bottom right, labeled: **Select lookup dataset**



14. Now we need to configure the Lookup fields to match, and the fields to output from the Lookup.

Enrich events with lookups

Add relevant information from a lookup dataset to your incoming event data. [Lookups documentation](#)

Lookup dataset: apjc_asset_inventory.csv

Match fields

| | |
|---------------|---------------|
| *Lookup field | Event field |
| host | = lookup_host |

+ Add

Output fields

Output all fields

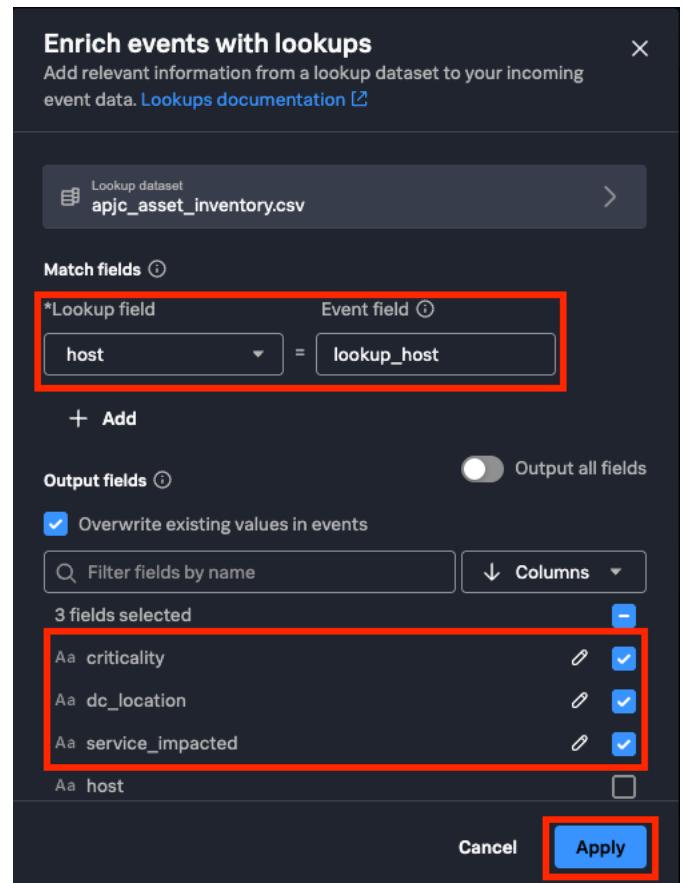
Overwrite existing values in events

Filter fields by name Columns

3 fields selected

| | |
|---------------------|-------------------------------------|
| Aa criticality | <input checked="" type="checkbox"/> |
| Aa dc_location | <input checked="" type="checkbox"/> |
| Aa service_impacted | <input checked="" type="checkbox"/> |
| Aa host | <input type="checkbox"/> |

Cancel



15. For **Match fields** configure the following:

- Lookup field as **host**
- Event field as **lookup_host**

16. For **Output fields** select the following:

- **criticality**
- **dc_location**
- **service_impacted**

17. Confirm your configuration matches the image on the right.

18. Select **Apply**.

19. To ensure the lookup file has been correctly added and is ready for use within our pipeline, let's perform a quick check.

20. You can confirm its successful addition by looking for the following line at the top of your pipeline:

```
import 'apjc_asset_inventory.csv' from /envs.splunk.'dmx-shw-2c53feabe860f8'.lookups
```

21. You'll observe that the IDE has placed the lookup SPL line after | eval index="rmit_events". To ensure that our events are enriched before any potential routing decisions are made, the lookup must occur earlier in the pipeline.

Therefore, we will move this lookup line to immediately follow | from \$source.

The revised pipeline should appear as follows:

```
import 'apjc_asset_inventory.csv' from /envs.splunk.dmx-show-  
2c3742ae24f8.lookups  
  
$pipeline =  
| from $source  
| lookup 'apjc_asset_inventory.csv' host AS lookup_host OUTPUT dc_location,  
criticality, service_impacted  
| eval index = "rmit_events"  
| into $destination;
```

Before we can enrich our event data, we must ensure we have a reliable key to connect it to our external asset inventory.

Below is our lookup file; pay close attention to the header names:

```
host,criticality,service_impacted,dc_location  
Maybank_MY,Tier1,Core Banking System,KUL-DC2  
Maybank_MY,Tier2,RENTAS Gateway,HKG-DC3
```

We've identified that the headers in our lookup table and the field names within our log events are inconsistent. This discrepancy necessitates the creation of a matching key to enable the integration of lookup data with our incoming events.

Lookup Enrichment Pattern for Nested JSON Events

In this section, we demonstrate how to enrich events that arrive wrapped in a _raw JSON field by performing a lookup and injecting the results back into the event.

Extract Fields from _raw

22. Edge Processor lookups operate on top-level fields, not nested JSON. Since our events arrive wrapped in _raw, we first extract the fields we need into temporary top-level fields using

`json_extract()`. We also extract the first element from the `impacted_systems` array using `mvindex()` for later processing.

Build the Lookup Key and Perform the Lookup

23. Extract the bank name (`financial_institution_id` from `_raw`) and use it as the lookup key to match against the host column in our CSV lookup table. The lookup returns three enrichment fields: data center location, criticality tier, and service name.

Inject Lookup Results Back into `_raw`

24. Using `json_set()`, we add each lookup result back into the `_raw` JSON object. Each line includes a null check to ensure we only inject valid values. This ensures the enrichment travels with the event through all subsequent transformations and routing stages.

Why This Pattern Matters

Then we are injecting enrichment back into `_raw`, we preserve the complete enriched event in the archived data. When the event is routed to different destinations based on type, the enrichment (criticality tier, data center location, service name) is already part of the event payload—no need to re-look it up at query time.

25. The above can be implemented in SPL in the following way:

```
// Extract fields from _raw for lookup key building
| eval affected_service = json_extract(_raw, "affected_service")
| eval change_description = json_extract(_raw, "change_description")
| eval impacted_systems_array = json_extract(_raw, "impacted_systems")
| eval first_impacted = if(isnotnull(impacted_systems_array),
mvindex(impacted_systems_array, 0), null)

    // Create lookup key: match on bank name only (lookup CSV has bank as the
host key)
| eval lookup_host = json_extract(_raw, "financial_institution_id")
| lookup 'apjc_asset_inventory.csv' host AS lookup_host OUTPUT
dc_location, criticality, service_impacted

    // Inject lookup results into _raw so enrichment travels with the event
| eval _raw = if(isnotnull(dc_location), json_set(_raw, "dc_location",
dc_location), _raw)
| eval _raw = if(isnotnull(criticality), json_set(_raw, "criticality",
criticality), _raw)
| eval _raw = if(isnotnull(service_impacted), json_set(_raw,
"service_impacted", service_impacted), _raw)
```

26. Remove any temporary fields created for the lookup:

```
// Remove temporary top-level lookup fields to avoid duplication (after
injecting into _raw)
```

```
| fields - affected_service, change_description, impacted_systems_array,
first_impacted, lookup_host, dc_location, criticality, service_impacted
```

Enrich with Asset and Operational Context

27. Having successfully created a composite key and mapped our lookup to the events, the next step is to enable enrichment. We aim to introduce a new field, risk_level, to assign a severity to specific services.

We want to define the new field risk_level in this order:

- Tier 0 -> High
- Tier 1 -> Medium
- All Other Tiers -> Low

SPL Example to do this:

```
// Extract criticality from _raw for enrichment logic
| eval criticality_from_raw = json_extract(_raw, "criticality")
| eval risk_level = case(criticality_from_raw == "Tier0", "High",
criticality_from_raw == "Tier1", "Medium", true, "Low")
| eval _raw = json_set(_raw, "risk_level", risk_level)
```

28. Check that newly created field risk_level which was enriched by the lookup is showing up by previewing the pipeline (▶). You should see the columns displayed in the preview pane:

| | </> _raw | Aa index | AA ip |
|---|---|----------|-----------|
| 1 | 10.2.1.33 - - [29/Apr/2013 18:09:05:132] "GET /category.screen?uid=9299979e-a268-4151-a4cd-2994c6971fce&category=Misc&JSESSIONID=SD3SL1FF7ADFF8 HTTP 1.1" 503 1829 "http://www.buttercupe.. | web | 10.2.1.33 |
| 2 | 10.2.1.33 - - [29/Apr/2013 18:09:05:132] "GET /category.screen?uid=f510fde4-e9aa-4e69-a713-34abc83574e5&category=Misc&JSESSIONID=SD3SL1FF7ADFF8 HTTP 1.1" 503 1829 "http://www.buttercupe.. | web | 10.2.1.33 |
| 3 | 10.2.1.33 - - [29/Apr/2013 18:09:03:144] "GET /product.screen?uid=feb32a24-87be-48b6-b8bd-30efb96a6591&product_id=MCF-3&JSESSIONID=SD1SL7FF2ADFF3 HTTP 1.1" 503 3985 "http://www.buttercu.. | web | 10.2.1.33 |
| 4 | 10.2.1.33 - - [29/Apr/2013 18:09:05:132] "GET /category.screen?uid=fdbcea80-0060-49fa-a0bf-db33fa1bde31&category=Misc&JSESSIONID=SD3SL1FF7ADFF8 HTTP 1.1" 503 1829 "http://www.buttercu.. | web | 10.2.1.33 |

Mask Sensitive Operational Details

29. We have several requirements for Masking Operational Details:

- Mask the last octet of all IP Addresses
- Hash all User IDs using SHA256
- Redact all Reporting User or Process
- Redact Event Descriptions if contains PII

30. Masking IP Addresses (Src & Dest)

```
// Mask IP addresses Source and Destination (last octet)
| eval masked_src_ip_address = replace(_raw.source_ip, /\.\d+$/, ".XXX")
| eval _raw = json_set(_raw, "source_ip", masked_src_ip_address)
```

```

    | eval masked_dest_ip_address = replace(_raw.destination_ip, /\.\d+$/, ".XXX")
    | eval _raw = json_set(_raw, "destination_ip", masked_dest_ip_address)

```

31. Hash all User IDs using SHA256:

```

// Hash user_id with SHA256 and a salt
| eval hashed_user_id = if(isnotnull(_raw.user_id), sha256(_raw.user_id +
"rmit_secret_salt"), null)
| eval _raw = json_set(_raw, "user_id", hashed_user_id)

```

32. Redact all Reporting User or Process:

```

// Redact reporting_user_or_process
| eval masked_reporting = if(isnotnull(json_extract(_raw,
"reporting_user_or_process")), "REDACTED", null)
| eval _raw = json_set(_raw, "reporting_user_or_process",
masked_reporting)

```

33. Redact Event Descriptions if contains PII:

```

// Redact event_description if it contains PII (example: redact if
contains 'user_')
| eval event_desc = json_extract(_raw, "event_description")
| eval redacted_desc = if(isnotnull(event_desc) and match(event_desc,
/user_/, "REDACTED_EVENT_DESC", event_desc)
| eval _raw = json_set(_raw, "event_description", redacted_desc)

```

34. Drop Temporary Fields:

```

// Drop Temporary fields used for processing
| fields - masked_src_ip_address, masked_dest_ip_address, hashed_user_id,
masked_reporting, event_desc, redacted_desc, criticality_from_raw, risk_level

```

Filter Non-Actionable Logs

35. Next, we will drop all events where the Status is 'Closed' or 'Resolved', unless Criticality is "Tier0" or "Tier1":

```

// Filter out events with status = "Closed" or "Resolved" unless
criticality is "Tier0" or "Tier1"
| eval status_from_raw = coalesce(json_extract(_raw, "status_code"),
json_extract(_raw, "status"))
| eval criticality_check = json_extract(_raw, "criticality")
| where (status_from_raw != "Closed" and status_from_raw != "Resolved") or
(criticality_check == "Tier0" or criticality_check == "Tier1")

```

Route Processed Data Based on RMiT Relevance

36. Next, we must first define our routes, and we do this by creating a new field called event_route we will define this field based on what our event_type is:

- cybersecurity -> CYBER_ROUTE
- change_request_log -> CHANGE_ROUTE
- third_party_risk_update -> TPRM_ROUTE
- All Else -> DEFAULT_ROUTE

```
// Route based on event_type - extract from _raw and store in _raw for
routing logic
| eval event_type_from_raw = json_extract(_raw, "event_type")
| eval event_route_value = case(event_type_from_raw ==
"cybersecurity_alert", "CYBER_ROUTE", event_type_from_raw ==
"change_request_log", "CHANGE_ROUTE", event_type_from_raw ==
"third_party_risk_update", "TPRM_ROUTE", true, "DEFAULT_ROUTE")
| eval _raw = json_set(_raw, "event_route", event_route_value)
```

We will also drop the temporary fields we just created:

```
// Drop Temporary fields used for processing
| fields - criticality_check, event_type_from_raw,
status_from_raw, event_route_value
```

37. Next, we will go ahead and start adding our routing. We can do this via the Action Menu on the right side. For our case, we will avoid this, the reason is that adding via the Action Menu will add the route to the wrong location and require extra configuration. To avoid confusion, we will manually add a route, and in a real world this is the way you would do it.

38. Since we are going the manual way for Routes, we need to manually import the route function in the pipeline, we do this by adding the following line to the top of our pipeline, above where the pipeline is importing our lookup.

```
import route from /splunk.ingest.commands
```

39. Now let's define what fields we are routing, to what destination, and their respective indexes:

- Event_Route = CYBER_ROUTE
 - Index = rmit_cyber_events
 - \$destination2
- Event_Route = CHANGE_ROUTE
 - Index = rmit_change_events
 - \$destination3
- Event_Route = TPRM_ROUTE
 - Index = rmit_tprm_events

- \$destination4

Be aware that we in our SPL below we will also drop several fields, the reason for this is that several fields are only relevant for specific routes, so we need to drop fields that are empty for these specific routes.

40. This is our SPL for CYBER_ROUTE:

```
| route _raw.event_route == "CYBER_ROUTE",
[
    // Remove unwanted keys from `_raw` for cyber events so the archived
payload is compact
    | eval _raw = json_delete(_raw, "change_request_id",
"change_description", "change_status", "risk_impact_assessment",
"planned_start_datetime_utc", "actual_end_datetime_utc", "third_party_id",
"vendor_name", "service_type", "risk_assessment_update_reason",
"overall_risk_rating", "key_risks_identified", "due_diligence_status")
    | eval index = "rmit_cyber_events"
    | into $destination2
]
```

41. This is our SPL for CHANGE_ROUTE:

```
| route _raw.event_route == "CHANGE_ROUTE",
[
    // Remove unwanted keys from `_raw` for change events so the archived
payload is compact
    | eval _raw = json_delete(_raw, "cyber_event_id", "threat_type",
"severity", "status", "impacted_systems", "detection_source",
"response_actions_taken", "bnm_notification_sent", "bnm_report_ref_id",
"third_party_id", "vendor_name", "service_type",
"risk_assessment_update_reason", "overall_risk_rating",
"key_risks_identified", "due_diligence_status")
    | eval index = "rmit_change_events"
    | into $destination3
]
```

42. This is our SPL for TPRM_ROUTE:

```
| route _raw.event_route == "TPRM_ROUTE",
[
    // Remove unwanted keys from `_raw` for TPRM events so the archived
payload is compact
    | eval _raw = json_delete(_raw, "cyber_event_id", "threat_type",
"severity", "status", "impacted_systems", "detection_source",
"response_actions_taken", "bnm_notification_sent", "bnm_report_ref_id",
"change_request_id", "change_description", "change_status",
```

```
"risk_impact_assessment", "planned_start_datetime_utc",
"actual_end_datetime_utc")
    | eval index = "rmit_tprm_events"
    | into $destination4
]
```

We will go ahead, and define the index for \$destination2, as rmit_cyber_events

Filter Low-Value or Non-Actionable Logs

43. Next, we will optimize our data by trimming and dropping low-value or non-actionable fields. We've observed that many fields remain empty unless associated with specific routes. By removing these empty fields, we can significantly reduce event size and send only relevant, populated fields to our default index.

This can be achieved using the following SPL:

```
// Drop fields associated with event_routes filtered before as they are empty
| fields - cyber_event_id, threat_type, severity, status,
impacted_systems, detection_source, response_actions_taken,
bnm_notification_sent, bnm_report_ref_id, change_request_id,
change_description, change_status, risk_impact_assessment,
planned_start_datetime_utc, actual_end_datetime_utc, third_party_id,
vendor_name, service_type, risk_assessment_update_reason, overall_risk_rating,
key_risks_identified, due_diligence_status
```

Add a Custom Indexed Field to Your Data

44. Finally, add a '**participant**' field to your events to help you identify them in Splunk Cloud:

```
$pipeline =
| from $source
...
| eval _raw = json_set (_raw, " participant ", " hrobalin ") ← Update with
  your own name
| index = "rmit_events"
| into $destination;
```

Configure Splunk Destination

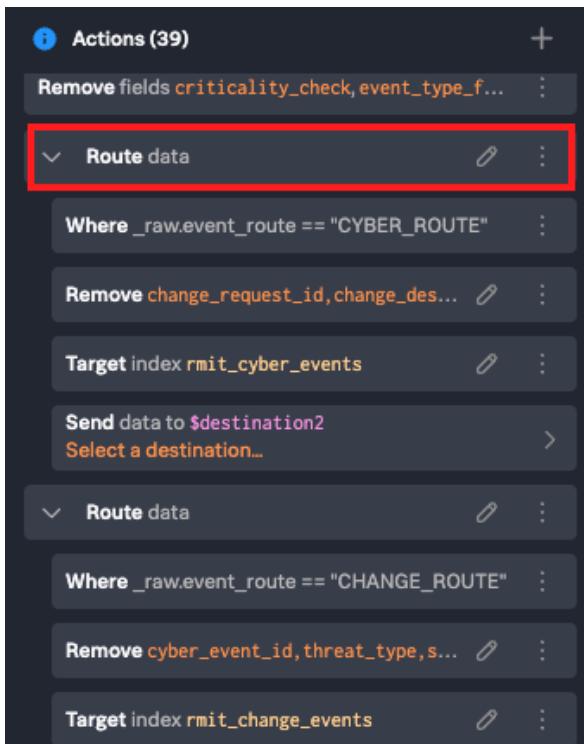
45. Before we can preview the pipeline, we will need to manually configure the destinations we are using for Log Destinations depending on your data routing requirements. We will need to do that for each of the following destinations:

\$destination2 -> default_splunk_cloud_destination

\$destination3 -> default_splunk_cloud_destination

\$destination4 -> default_splunk_cloud_destination

46. On the right side, under Actions, scroll down to the Route Data sections:



The screenshot shows the Splunk Actions interface with a dark header bar. Below it, there's a list of actions. A red box highlights the first 'Route data' section. This section contains a 'Where' clause ('_raw.event_route == "CYBER_ROUTE"'), a 'Remove' clause ('change_request_id, change_des...'), a 'Target' index ('rmit_cyber_events'), and a 'Send data to \$destination2' section with a 'Select a destination...' button. Below this is another 'Route data' section, also highlighted with a red box. It contains a 'Where' clause ('_raw.event_route == "CHANGE_ROUTE"'), a 'Remove' clause ('cyber_event_id, threat_type, s...'), and a 'Target' index ('rmit_change_events'). Each section has edit and more options icons.

47. Select on Send data to \$destination2, and select default_splunk_cloud_destination:



Screenshot of the Splunk interface showing the 'Route data' configuration screen. A red box highlights the 'Send data to \$destination2' step, which has a dropdown menu open. Another red box highlights the 'default_splunk_cloud_destination' option in the dropdown list.

48. Select **Apply** on the bottom right.

49. Select on Send data to \$destination3, and select default_splunk_cloud_destination:

Screenshot of the Splunk interface showing the 'Route data' configuration screen. A red box highlights the 'Send data to \$destination3' step, which has a dropdown menu open. Another red box highlights the 'default_splunk_cloud_destination' option in the dropdown list.

50. Select **Apply** on the bottom right.

51. Select on Send data to \$destination4, and select default_splunk_cloud_destination:

Screenshot of the Splunk interface showing the 'Route data' configuration screen. A red box highlights the 'Send data to \$destination4' step, which has a dropdown menu open. Another red box highlights the 'default_splunk_cloud_destination' option in the dropdown list.



52. Select **Apply** on the bottom right.

Preview Your Pipeline

53. Test your pipeline by clicking on the blue Preview pipeline button (▶) in the top right corner of the screen. You should see a preview of your events in the center of the screen with processed data and an index field added.

| Previewing: \$pipeline | | \$destination | 5 results | View: Table |
|------------------------|--|---------------|----------------------|-------------|
| | </> _raw | | | |
| 1 | {"status_code": "Failed", "dc_location": "JPN-DC4", "event_timestamp_utc": "2025-10-31T13:47:46.732034Z", "ops_event_id": "OPS-F49FD0D6", "criticality": "Tier 0", "event_description": "REDACTED_EVENT_DESC", "event_source_component": "SIEM_Platform", "event_timestamp": "2025-10-31T13:47:46.732034Z", "participant": "hrobalin", "source_ip": "67.174.182.XX", "time": "2025-10-31T13:47:46.732034Z"}, {"status_code": "Completed", "dc_location": "SGP-DC1", "event_timestamp_utc": "2025-10-31T13:47:46.732034Z", "ops_event_id": "OPS-3FAD012C", "criticality": "Tier 1", "event_description": "REDACTED_EVENT_DESC", "event_source_component": "Automated_Monitoring", "event_timestamp": "2025-10-31T13:47:46.732034Z", "participant": "hrobalin", "source_ip": "67.174.182.XX", "time": "2025-10-31T13:47:46.732034Z"}, {"status_code": "Detected", "dc_location": "JPN-DC4", "event_timestamp_utc": "2025-10-31T13:47:46.732034Z", "ops_event_id": "OPS-AE8A4F3A", "criticality": "Tier 3", "event_description": "REDACTED_EVENT_DESC", "event_source_component": "Manual_Log", "event_timestamp": "2025-10-31T13:47:46.732034Z", "participant": "hrobalin", "source_ip": "67.174.182.XX", "time": "2025-10-31T13:47:46.732034Z"}, {"status_code": "Completed", "dc_location": "KUL-DC2", "event_timestamp_utc": "2025-10-31T13:47:46.732034Z", "ops_event_id": "OPS-3FAD012C", "criticality": "Tier 1", "event_description": "REDACTED_EVENT_DESC", "event_source_component": "GRC_Tool", "event_timestamp": "2025-10-31T13:47:46.732034Z", "participant": "hrobalin", "source_ip": "67.174.182.XX", "time": "2025-10-31T13:47:46.732034Z"}, {"status_code": "Completed", "data_sensitivity_level": "Restricted", "dc_location": "JPN-DC4", "event_timestamp_utc": "2025-10-31T13:47:46.732034Z", "ops_event_id": "OPS-3FAD012C", "criticality": "Tier 2", "event_description": "REDACTED_EVENT_DESC", "event_source_component": "Manual_Log", "data_event_type": "Data_Access_Violation_Attempt", "event_time": "2025-10-31T13:47:46.732034Z", "participant": "hrobalin", "source_ip": "67.174.182.XX", "time": "2025-10-31T13:47:46.732034Z"}] | | | |
| | | As index | As sourcetype | |
| | | rmit_events | rmit:synthetic:event | |

RMIT Pipeline

Your RMIT Pipeline SPL should look like the following (Be aware copying below will require updating env name on second line):

```
import route from /splunk.ingest.commands
import 'apjc_asset_inventory.csv' from /envs.splunk.'dmx-shw-9cf6b5b1083388'.lookups

$pipeline =
| from $source

// Extract fields from _raw for lookup key building
| eval affected_service = json_extract(_raw, "affected_service")
| eval change_description = json_extract(_raw, "change_description")
| eval impacted_systems_array = json_extract(_raw, "impacted_systems")
| eval first_impacted = if(isnotnull(impacted_systems_array),
mvindex(impacted_systems_array, 0), null)

// Create lookup key: match on bank name only (lookup CSV has bank as the host key)
| eval lookup_host = json_extract(_raw, "financial_institution_id")
| lookup 'apjc_asset_inventory.csv' host AS lookup_host OUTPUT dc_location,
criticality, service_impacted

// Inject lookup results into _raw so enrichment travels with the event
| eval _raw = if(isnotnull(dc_location), json_set(_raw, "dc_location", dc_location),
_raw)
| eval _raw = if(isnotnull(criticality), json_set(_raw, "criticality", criticality),
_raw)
| eval _raw = if(isnotnull(service_impacted), json_set(_raw, "service_impacted",
service_impacted), _raw)
```

```

    // Remove temporary top-level lookup fields to avoid duplication (after injecting into
    _raw)
    | fields - affected_service, change_description, impacted_systems_array,
first_impacted, lookup_host, dc_location, criticality, service_impacted

    // Extract criticality from _raw for enrichment logic
    | eval criticality_from_raw = json_extract(_raw, "criticality")
    | eval risk_level = case(criticality_from_raw == "Tier0", "High", criticality_from_raw
== "Tier1", "Medium", true, "Low")
    | eval _raw = json_set(_raw, "risk_level", risk_level)

    // Mask IP addresses Source and Destination (last octet)
    | eval masked_src_ip_address = replace(_raw.source_ip, /\.\d+$/, ".XXX")
    | eval _raw = json_set(_raw, "source_ip", masked_src_ip_address)

    | eval masked_dest_ip_address = replace(_raw.destination_ip, /\.\d+$/, ".XXX")
    | eval _raw = json_set(_raw, "destination_ip", masked_dest_ip_address)

    // Hash user_id with SHA256 and a salt
    | eval hashed_user_id = if(isnotnull(_raw.user_id), sha256(_raw.user_id +
"rmit_secret_salt"), null)
    | eval _raw = json_set(_raw, "user_id", hashed_user_id)

    // Redact reporting_user_or_process
    | eval masked_reporting = if(isnotnull(json_extract(_raw,
"reporting_user_or_process")), "REDACTED", null)
    | eval _raw = json_set(_raw, "reporting_user_or_process", masked_reporting)

    // Redact event_description if it contains PII (example: redact if contains 'user_')
    | eval event_desc = json_extract(_raw, "event_description")
    | eval redacted_desc = if(isnotnull(event_desc) and match(event_desc, /user_/),
"REDACTED_EVENT_DESC", event_desc)
    | eval _raw = json_set(_raw, "event_description", redacted_desc)

    // Drop Temporary fields used for processing
    | fields - masked_src_ip_address, masked_dest_ip_address, hashed_user_id,
masked_reporting, event_desc, redacted_desc, criticality_from_raw, risk_level

    // Filter out events with status = "Closed" or "Resolved" unless criticality is
    "Tier0" or "Tier1"
    | eval status_from_raw = coalesce(json_extract(_raw, "status_code"),
json_extract(_raw, "status"))
    | eval criticality_check = json_extract(_raw, "criticality")
    | where (status_from_raw != "Closed" and status_from_raw != "Resolved") or
(criticality_check == "Tier0" or criticality_check == "Tier1")

```

```

// Route based on event_type - extract from _raw and store in _raw for routing logic
| eval event_type_from_raw = json_extract(_raw, "event_type")
| eval event_route_value = case(event_type_from_raw == "cybersecurity_alert",
"CYBER_ROUTE", event_type_from_raw == "change_request_log", "CHANGE_ROUTE",
event_type_from_raw == "third_party_risk_update", "TPRM_ROUTE", true, "DEFAULT_ROUTE")
| eval _raw = json_set(_raw, "event_route", event_route_value)

// Drop Temporary fields used for processing
| fields - criticality_check, event_type_from_raw, status_from_raw, event_route_value

// Route to different indexes based on event_type
| route _raw.event_route == "CYBER_ROUTE",
[
    // Remove unwanted keys from `_raw` for cyber events so the archived payload is
compact
    | eval _raw = json_delete(_raw, "change_request_id", "change_description",
"change_status", "risk_impact_assessment", "planned_start_datetime_utc",
"actual_end_datetime_utc", "third_party_id", "vendor_name", "service_type",
"risk_assessment_update_reason", "overall_risk_rating", "key_risks_identified",
"due_diligence_status")
        | eval index = "rmit_cyber_events"
        | into $destination2
]
| route _raw.event_route == "CHANGE_ROUTE",
[
    // Remove unwanted keys from `_raw` for change events so the archived payload is
compact
    | eval _raw = json_delete(_raw, "cyber_event_id", "threat_type", "severity",
"status", "impacted_systems", "detection_source", "response_actions_taken",
"bnm_notification_sent", "bnm_report_ref_id", "third_party_id", "vendor_name",
"service_type", "risk_assessment_update_reason", "overall_risk_rating",
"key_risks_identified", "due_diligence_status")
        | eval index = "rmit_change_events"
        | into $destination3
]
| route _raw.event_route == "TPRM_ROUTE",
[
    // Remove unwanted keys from `_raw` for TPRM events so the archived payload is
compact
    | eval _raw = json_delete(_raw, "cyber_event_id", "threat_type", "severity",
"status", "impacted_systems", "detection_source", "response_actions_taken",
"bnm_notification_sent", "bnm_report_ref_id", "change_request_id", "change_description",
"change_status", "risk_impact_assessment", "planned_start_datetime_utc",
"actual_end_datetime_utc")
        | eval index = "rmit_tprm_events"
        | into $destination4
]

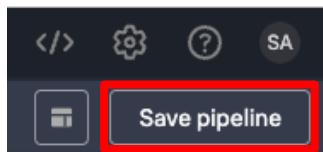
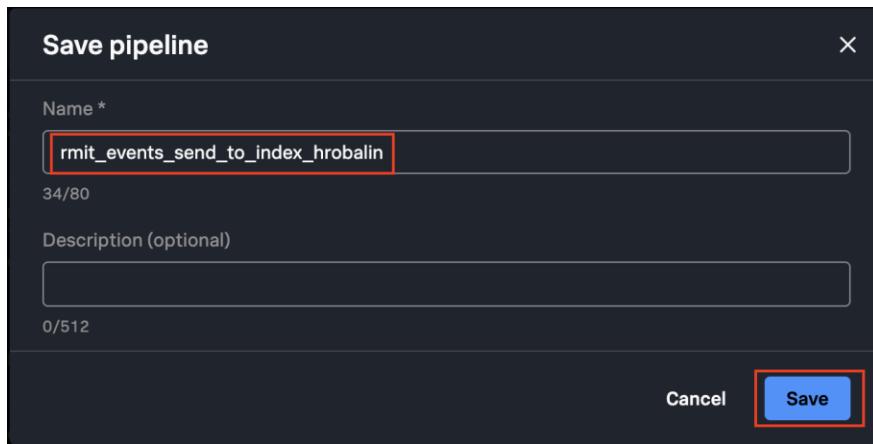
```

```
// Drop fields associated with event_routes filtered before as they are empty
| eval _raw = json_delete(_raw, "cyber_event_id", "threat_type", "severity", "status",
"impacted_systems", "detection_source", "response_actions_taken", "bnm_notification_sent",
"bnm_report_ref_id", "change_request_id", "change_description", "change_status",
"risk_impact_assessment", "planned_start_datetime_utc", "actual_end_datetime_utc",
"third_party_id", "vendor_name", "service_type", "risk_assessment_update_reason",
"overall_risk_rating", "key_risks_identified", "due_diligence_status")

| eval index = "rmit_events"
| into $destination;
```

Save Your Pipeline

54. Save your pipeline by clicking on Save pipeline in the top right corner of the screen. Give your pipeline a suitable name, such as **rmit_events_send_to_index_yourName** or something similar. As before, be sure to include your name or initials somewhere in the pipeline name to avoid confusion among those who are sharing the same environment as you!

Save pipeline

Name *

34/80

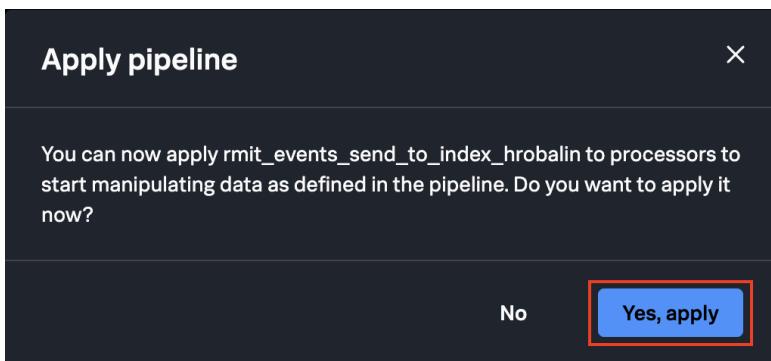
Description (optional)

0/512

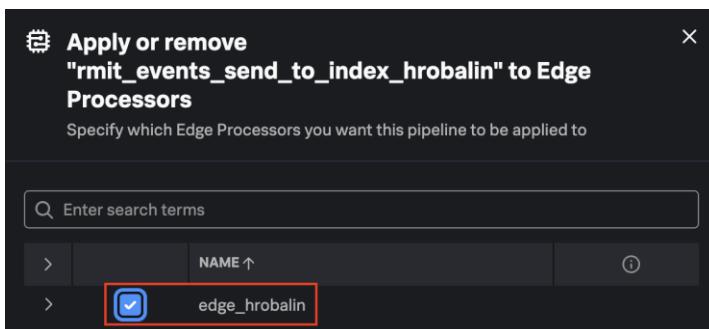
Cancel Save

Apply the Pipeline to Your Edge Processor

55. Now an “Apply Pipeline” dialog box will appear, click on “**Yes, apply**”.

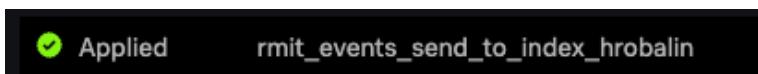


56. Check the box next to the name of the Edge Processor you created earlier and click on Save.



Confirm RMiT Pipeline is applied

57. Make sure the RMiT Events Pipeline is applied. Navigate to your Pipelines, you should see your pipeline name, and a status if Applied or Unapplied. See image below:



Check Your Data in Splunk Cloud

58. Log in to Splunk Cloud and open up the **Search & Reporting** app. Run the following search over the **last 15 minutes** and verify that you now see the redacted events.

```
index=rmit_events participant="hrobalin" ← Update the name!
```

| i | Time | Event |
|---|---|--|
| > | <u>11/5/25</u> <u>5:45:57.971 AM</u> | <pre>{ [{ affected_service: Mobile_Banking_App criticality: [] }] dc_location: [] destination_ip: 216.185.131.XXX duration_minutes: null event_description: REDACTED_EVENT_DESC event_route: DEFAULT_ROUTE event_source_component: GRC_Tool event_timestamp: 2025-11-04T20:45:57.971562Z event_timestamp_utc: 2025-11-04T20:45:57.971562Z event_type: it_operations_event financial_institution_id: UOB_Malaysia_MY log_event_id: fa83939f-9196-4866-9789-7b083276029b operation_event_type: Capacity_Threshold_Breached ops_event_id: OPS-5E46AEA6 participant: hrobalin reporting_user_or_process: REDACTED resolution_details: System rebooted and services restored successfully. risk_level: Low rmit_controlViolation: true rmit_event_category: it_operations_event service_impacted: [] } source_ip: 70.116.9.XXX status_code: Completed system_criticality: Critical system_id: SYSTEM-2469 technology_domain: Server_And_Storage user_id: 9caab987b9ad3b18cdd4be153f96f5dd548907be91156a00f763e5b62d74338d</pre> |

Note that all the preprocessing, and enrichment that was done on this event. From redacting fields, to masking, hashing fields, and dropping fields. We can even verify that our events are routed properly.

Exercise 8 – Buttercup Enterprises: ANZ Regulations (CPS-230)

Description

In the ANZ region, Buttercup Enterprises must adhere to the CPS 230 standard, which focuses on operational risk management and resilience within the financial sector.

To comply with CPS 230, you will configure the Edge Processor to identify, filter, and securely route data related to operational risk events and controls. This involves creating a specialized pipeline for capturing, archiving, and analyzing relevant operational risk data, ensuring alignment with CPS 230 requirements for ongoing oversight and reporting.

CPS 230 Pipeline Guide: A/NZ Regulations (Cross-industry Prudential Standard)

Objective: The Cross-industry Prudential Standard (CPS) 230 pipeline ensures compliance by identifying, filtering, and securely routing data related to operational risk events and controls.

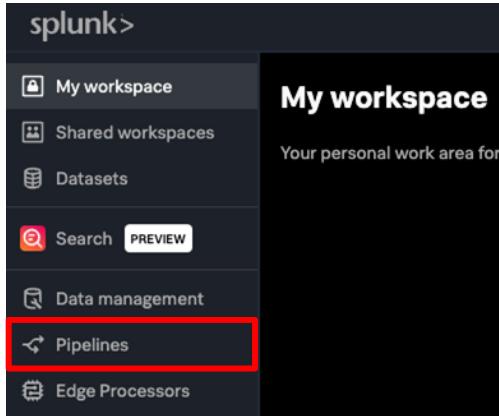
1. **Enrich Data with Lookups:** Enhance events with asset inventory information (tolerance level, critical operation name, and business unit) using a lookup file.
2. **Define Risk Levels:** Assign a risk_level (High, Medium, Low) based on the criticality from the lookup.
3. **Mask Sensitive Operational Details:** Mask IP addresses, pseudonymize user IDs, mask reporting users/processes, and redact event descriptions if they contain PII.
4. **Filter Non-Actionable Logs:** Retain only relevant events based on status and criticality.
5. **Route Based on APRA Notification Candidate Status:** Direct events to different Splunk indexes (cps_apra_notify_events, cps_significant_events) based on whether they are APRA notification candidates.
6. **Minimize Data:** Remove low-value or empty fields from the default stream.
7. **Identify Your Data:** Add a custom field to easily identify your events in a shared Splunk Cloud environment.



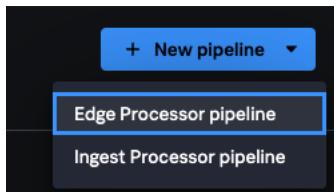
Steps

Create a New Pipeline

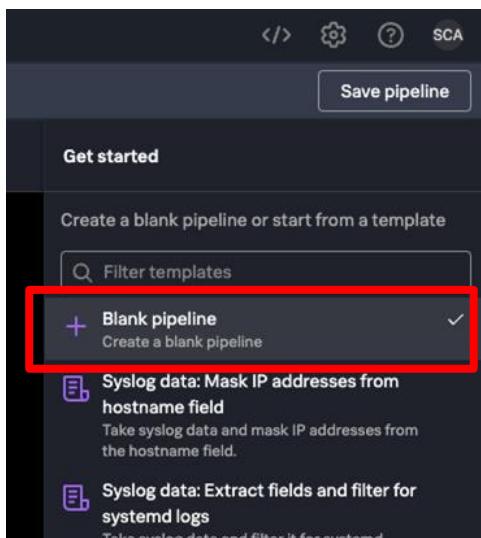
1. While logged in to the Edge Processor service (see [Access Edge Processor Service](#)) click on **Pipelines** on the left of the page.



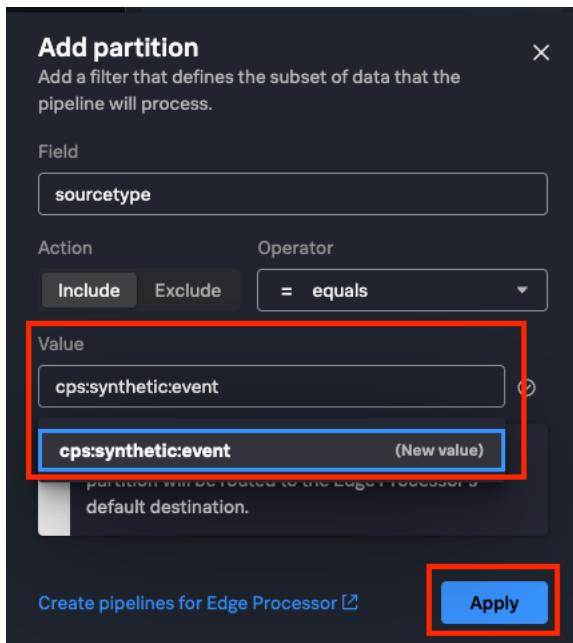
2. In the top right corner of the **Pipelines** page click on **+ New pipeline**. Select **Edge Processor pipeline**



3. On the right hand side of the **New pipeline** page under the **Get started** pane select on **Blank pipeline** and click on **Next**.



4. On the **Define your pipeline's partition** pane click on the (+) button next to the Partition field and select the value **sourcetype** for **Field**, select “Keep” for Action, “= equals” for Operator and write **cps:synthetic:event** for the **value (this will be a new value)** and click **Apply**.



And then click on **Next** button at the bottom right.

5. An **Add sample data** menu will appear on the right side. Upload the **cps_sample_events.jsonl** file using the “**upload file...**” option at the bottom.

Add sample data

Enter or upload sample data

```
{"event_id": "d196f85d-1957-4cb6-a330-2da140d8
{"event_id": "a72a97fc-69fe-4e2a-8e26-c7076658
{"event_id": "5c7faaf5-71ee-4535-911d-89c561d3
{"event_id": "e7e35d84-17df-4df2-b53c-c4b31e19
{"event_id": "3cb75bc2-d1f5-4edd-be6d-ce634dee
{"event_id": "474b426d-ba18-4070-9813-09c1b2f9
{"event_id": "48f7dbdf-9405-493e-be1a-4cac9f7b
{"event_id": "d9e12787-da91-4b12-b9a7-d1464b40
{"event_id": "42fe2376-1562-472e-b1ab-254ff2eb
{"event_id": "231a5708-3aab-4564-a89a-fa0402be
{"event_id": "497191f5-f3df-4648-809f-ce044596
{"event_id": "77e84b43-c3fa-4c5f-b9b6-a8f920f5
{"event_id": "dedc7e06-a8ad-4921-9cc6-c311cb44
{"event_id": "1e3dd20f-948f-4fb9-86c7-5e5b0d3b
{"event_id": "45a72900-fee4-4fd9-abe4-20cf1fbb
{"event_id": "5267d123-d78e-40de-8d0d-1bb84cda
{"event_id": "4a4d1b21-6666-4ade-a102-258c2c5
28,963 / 30,000 characters
```

Drop your file anywhere or [upload file...](#)

Sample data format

[Raw text](#) [CSV](#)

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

For the **Sample data format** select **Raw text** and click on **Next**.

6. A “Select a data destination” menu will appear. Select the **default_splunk_cloud_destination** as your data destination and click on **Next**.

Select a data destination

Filter datasets

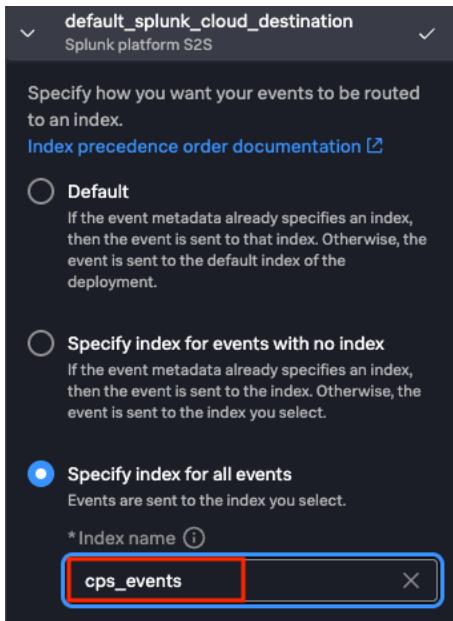
All kinds

- default_splunk_cloud_destination** Splunk platform S2S
- edge_s3_hrobalin Amazon S3



Route Your Data to the cps_events Index

7. A pane below will open, there you will select **Specify index for all events** and set the index to the **cps_events** from the drop down, as requested by the Buttercup security team, and click on **Done** at the bottom.



8. A screen like below will appear:

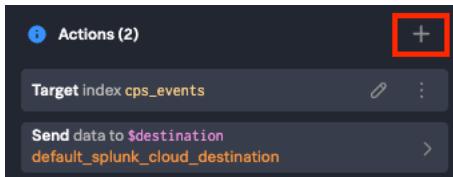
| _raw | _index | _sourcetype |
|--|------------|---------------------|
| 1 {"event_id": "d196f85d-1957-4cb6-a330-2da14bd8ff82", "event_timestamp": "2025-10-31T13:46:08.999576Z", "timestamp": "2025-10-31T13:46:08.999576Z", "entity_y_id": "ENTITY_961C", "critical_operation_id": "CRITOP-4299", "event_type": "control_assessment", "impact_description": "Impact to Superannuation Fund A."} | cps_events | cps:synthetic:event |
| 2 {"event_id": "a7a9afc-69fe-4e2a-8e20-c7076658dd4d", "event_timestamp": "2025-10-31T13:46:08.999576Z", "timestamp": "2025-10-31T13:46:08.999576Z", "entity_y_id": "ENTITY_961C", "critical_operation_id": "CRITOP-5636", "event_type": "control_assessment", "impact_description": "Impact to Trade Settlements department."} | cps_events | cps:synthetic:event |
| 3 {"event_id": "5cf7aa5-71ee-4559-91d-895c61d1fbef", "event_timestamp": "2025-10-31T13:46:08.999576Z", "timestamp": "2025-10-31T13:46:08.999576Z", "entity_y_id": "ENTITY_FA49", "critical_operation_id": "CRITOP-5889", "event_type": "third_party_risk_event", "impact_description": "Impact to Liquidity Management system."} | cps_events | cps:synthetic:event |

As you integrate additional actions into your pipeline, please insert them between the existing pipeline stages and the final index. It is crucial to maintain the order specified in this guide. Refer to the example below:

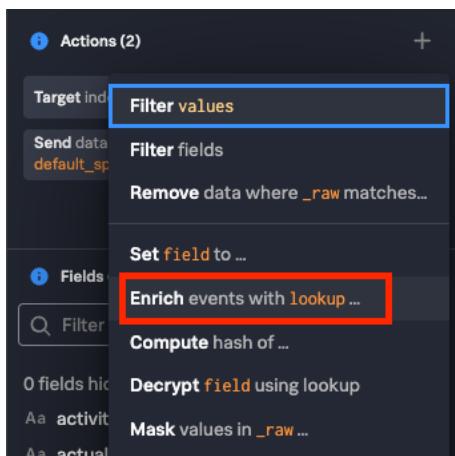
```
$pipeline = | from $source
. . . ADD NEW ACTIONS HERE
| eval index = "cps_events"
| into $destination;
```

Configure Lookup for Events

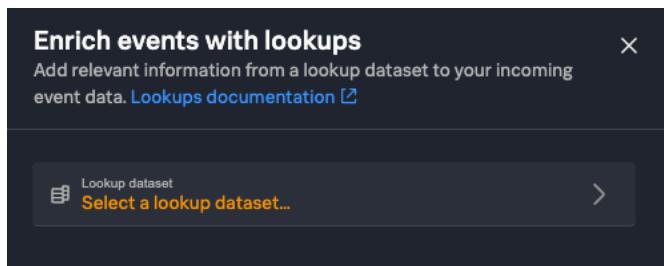
9. We want to first start by enriching this data, it will provide us more context to an event coming in as well, as allow us to do advance routing based on enriched data. To accomplish this, we will import a lookup using Edge Processing, through the **Actions** menu on the right side. Press the Plus button on the right side.



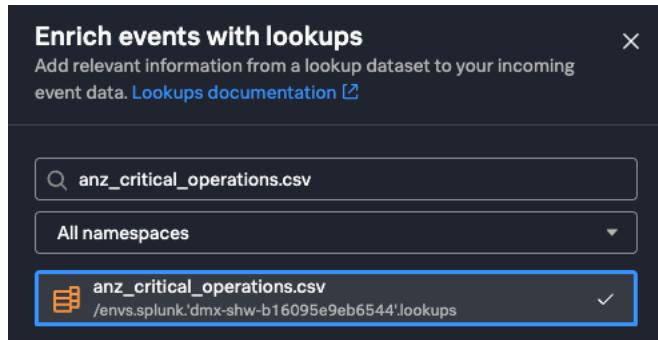
10. We will select: **Enrich events with lookup**



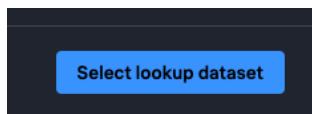
11. Now we need to select a Lookup dataset.



12. In our example we will search for **anz_critical_operations.csv**, and select the option that comes up.



13. Make sure to press the button on the bottom right, labeled: **Select lookup dataset**



14. Now we need to configure the Lookup fields to match, and the fields to output from the Lookup.

15. For **Match fields** configure the following:

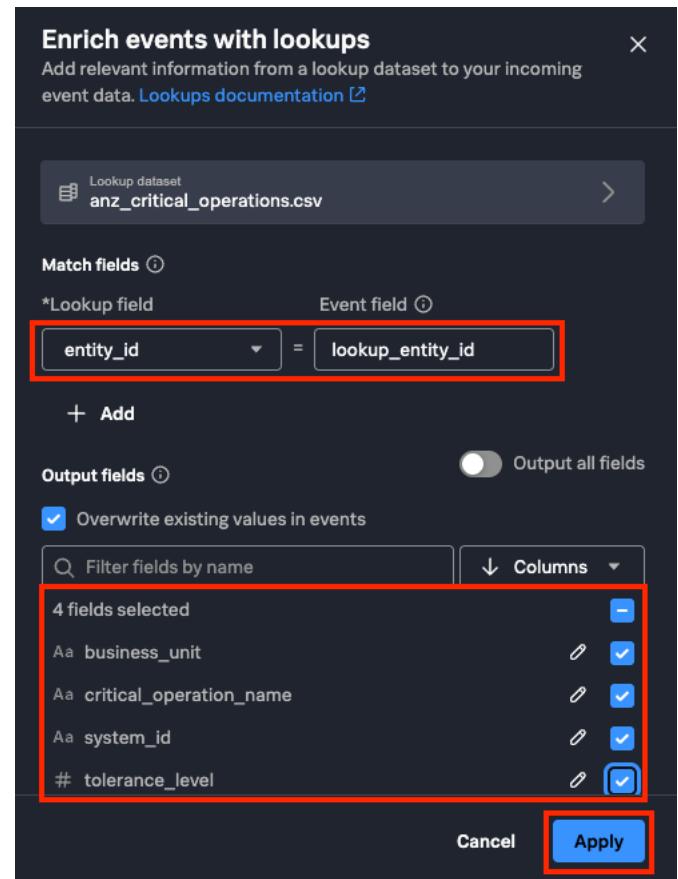
- Lookup field as **entity_id**
- Event field as **lookup_entity_id**

16. For **Output fields** select the following

- **business_unit**
- **critical_operation_name**
- **system_id**
- **tolerance_level**

17. Confirm your configuration matches the image on the right.

18. Select **Apply**.



19. Confirm lookup file is added to pipeline, and ready to use.

To confirm that you have added your lookup file correctly, you should see the following line at the top of your pipeline (be aware the dmx instance id may be different for your environment) :

```
import anz_critical_operations.csv /from /envs.splunk.'dmx-shw-2c53feabe860f8'.lookups
```

20. You'll observe that the IDE has placed the lookup SPL line after **| eval index="cps_events"**. To ensure that our events are enriched before any potential routing decisions are made based on that enrichment, the lookup must occur earlier in the pipeline. Therefore, we will move this lookup line to immediately follow **| from \$source**.

The revised pipeline should appear as follows:

```
import 'anz_critical_operations.csv' from /envs.splunk.buttercup.lookups

$pipeline =
| from $source
    // Lookup enrichment
    | lookup 'anz_critical_operations.csv' entity_id OUTPUT system_id,
critical_operation_name, tolerance_level, business_unit
    | eval index = "cps_events"
    | into $destination;
```

21. Before we can enrich our event data, we must ensure we have a reliable key to connect it to our external asset inventory.

22. Below is our lookup file; pay close attention to the header names:

```
system_id,entity_id,critical_operation_name,tolerance_level,business_unit  
sys_001,ENTITY_6885,Payments Processing,60,Operations  
sys_002,ENTITY_961C,Customer Account Management,180,Operations
```

Lookup Enrichment Pattern for Nested JSON Events

In this section, we demonstrate how to enrich events that arrive wrapped in a `_raw` JSON field by performing a lookup and injecting the results back into the event.

Extract Lookup Key from `_raw`

23. Edge Processor lookups require top-level fields to match against CSV data. Since our events arrive wrapped in `_raw`, we first extract the `entity_id` (which identifies the critical operation) into a temporary top-level field.

Perform the Lookup

24. We match the extracted `entity_id` against the CSV lookup table's `entity_id` column. The lookup returns four enrichment fields:

- **system_id**: System identifier for the critical operation
- **critical_operation_name**: Human-readable operation name (e.g., "Payments Processing")
- **tolerance_level**: Maximum downtime tolerance in minutes
- **business_unit**: Department that owns the operation

Inject Enrichment back into `_raw`

25. Using `json_set()`, we add each lookup result back into the `_raw` JSON object. Each line includes a null check to ensure we only inject valid values. This ensures the enrichment travels with the event through all subsequent transformations and routing stages, so the archived record contains complete context about which critical operation was affected.

Clean Up Temporary Fields

26. We remove the temporary top-level fields after injection. This prevents duplication—the enrichment now lives in `_raw` where it belongs, and the top-level fields are no longer needed.

Why This Pattern Matters

27. By injecting enrichment into `_raw`, we preserve the complete enriched event in the archived data. When analysts or compliance officers query these events later, they can immediately see which critical operation was affected, its tolerance level, and business unit—without needing to perform a separate lookup at query time. This is especially important for compliance audits where event

context must be frozen and immutable.

28. The above can be implemented in the following SPL:

```
// Extract entity_id from _raw for lookup key building
| eval lookup_entity_id = json_extract(_raw, "entity_id")

// Lookup enrichment: match on entity_id from events to entity_id in CSV
| lookup 'anz_critical_operations.csv' entity_id AS lookup_entity_id
OUTPUT system_id, critical_operation_name, tolerance_level, business_unit

// Inject lookup results into _raw so enrichment travels with the event
| eval _raw = if(isnotnull(system_id), json_set(_raw, "system_id",
system_id), _raw)
| eval _raw = if(isnotnull(critical_operation_name), json_set(_raw,
"critical_operation_name", critical_operation_name), _raw)
| eval _raw = if(isnotnull(tolerance_level), json_set(_raw,
"tolerance_level", tolerance_level), _raw)
| eval _raw = if(isnotnull(business_unit), json_set(_raw, "business_unit",
business_unit), _raw)

// Remove temporary top-level lookup fields to avoid duplication (after
injecting into _raw)
| fields - lookup_entity_id, system_id, critical_operation_name,
tolerance_level, business_unit
```

29. To implement the above, copy, and replace the current lookup line added by the editor.

Define Enrichment with Critical Operation (CO) and Tolerance Context

30. To further enrich our data, we will assign a severity_numeric to each event, based on the incident's severity. We will achieve this by extracting the relevant information from the _raw field and using a case statement to convert it into a numerical value. This numerical assignment is particularly useful for optimizing computational processes.

Insert the following SPL into your pipeline, immediately after the previous addition.

```
// Extract severity from _raw and convert to numeric for filtering
| eval cps230_incident_severity = json_extract(_raw,
"cps230_incident_severity")
| eval severity_numeric = case(
    cps230_incident_severity == "Critical", 4,
    cps230_incident_severity == "High", 3,
    cps230_incident_severity == "Medium", 2,
    cps230_incident_severity == "Low", 1,
    true, 0
)
```

Define Filtering for Significance and Actionability

31. Next in our step we need to filter events based on the following:

- APRA Notifications Candidate is True
- Severity Level: Critical & High

32. To accomplish this, we can use the SPL below:

```
// Filtering: keep events that are APRA notification candidates OR have high  
severity  
  
| where json_extract(_raw, "apra_notification_candidate") == true OR  
severity_numeric >= 3
```

Define Data Masking and Hashing

33. We have filtered the events we want to keep. Before we can proceed and route these events, we will need to mask, and hash

```
// Pseudonymize reporter_user_id  
| eval reporter_user_id = json_extract(_raw, "reporter_user_id")  
| eval hashed_reporter_id = if(isnotnull(reporter_user_id),  
sha256(reporter_user_id + "cps230_salt"), null)  
| eval _raw = json_set(_raw, "reporter_user_id", hashed_reporter_id)  
  
// Mask vendor_name  
| eval vendor_name = json_extract(_raw, "vendor_name")  
| eval masked_vendor = if(isnotnull(vendor_name), "REDACTED", null)  
| eval _raw = json_set(_raw, "vendor_name", masked_vendor)  
  
// Drop temporary fields used for processing  
| fields - lookup_entity_id, system_id, critical_operation_name,  
tolerance_level, business_unit, reporter_user_id, hashed_reporter_id,  
vendor_name, masked_vendor, cps230_incident_severity, severity_numeric
```

Define Archival Routing

34. Now that we have filtered, hashed, and masked our data, we must archive it before sending it to Splunk.

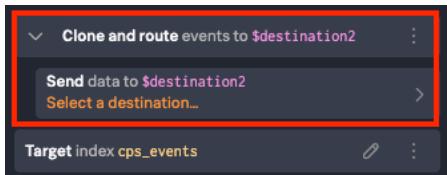
This can be done in two ways, via the UI, or via the pipeline itself. In our case we will manually do it. The reason for this is that if we do via the UI, the UI will put the archive query in the wrong location.

35. Similar as we have done before let's paste the following SPL2 statement after the above SPL.

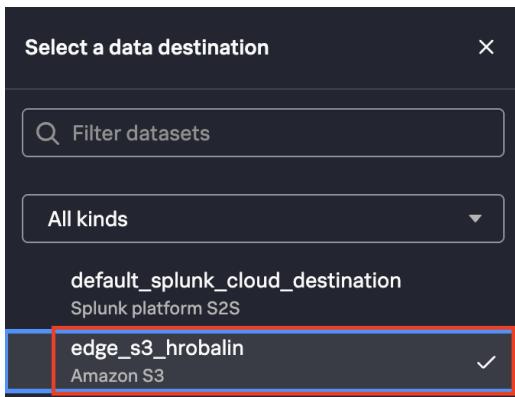
This will define Splunk as \$destination, and Amazon S3 Bucket for Archiving as \$destination2.

```
// Send raw data to secure archive (e.g., Amazon S3 Bucket)
| thru [ | into $destination2 ]
```

36. Now let's define in the pipeline setting which Amazon S3 Bucket will be \$destination2. Click Select a destination

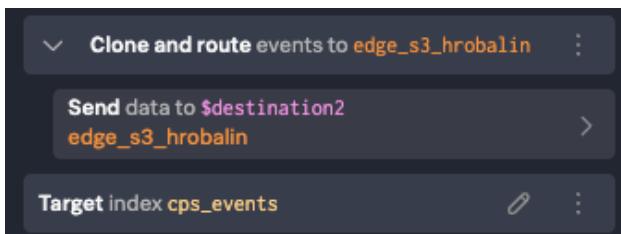


37. Select the Amazon S3 Endpoint for our use case: edge_s3_username.



38. Click **Apply** at the bottom right.

39. Actions should now look like this, where \$destination2 is an S3 Destination:



Route Processed Data Based on CPS 230 APRA Notification

40. Following the archival of all events, our focus shifts to filtering for 'APRA notification candidates,' whose criteria are outlined below:

- apra_notification_candidate -> true
- apra_notification_candidate -> false

41. Next, we will go ahead and start adding our routing. We can do this via the Action Menu on the right side. For our case, we will avoid this, the reason is that adding via the Action Menu will add

the route to the wrong location and require extra configuration. To avoid confusion, we will manually add a route, and in a real world this is the way you would do it.

42. Since we are going the manual way for Routes, we need to manually import the route function in the pipeline, we do this by adding the following line to the top of our pipeline, above where the pipeline is importing our lookup.

```
import route from /splunk.ingest.commands
```

43. Now let's define what fields we are routing, to what destination, and their respective indexes:

- `apra_notification_candidate = true`
 - `Index = cps_apra_notify_events`
 - `$destination3`
- `apra_notification_candidate = false`
 - `Index = cps_significant_events`
 - `$destination4`

44. This is our SPL for true:

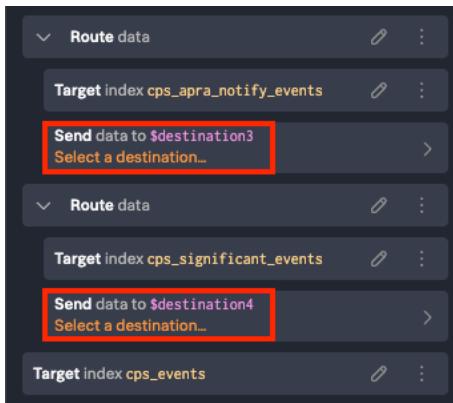
```
// Route APRA notification candidate events to APRA notifications index
| route _raw.apra_notification_candidate == true,
[
    | eval index = "cps_apra_notify_events"
    | into $destination3
]
```

45. This is our SPL for false:

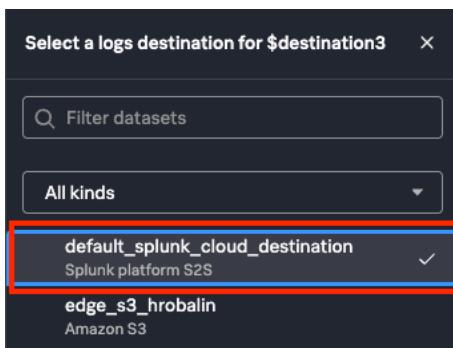
```
// Route non-APRA events to significant events index
| route _raw.apra_notification_candidate == false,
[
    | eval index = "cps_significant_events"
    | into $destination4
]
```

46. Navigate to the 'Route Section' located under 'Actions' on the right-hand side (Make sure to scroll down).

You'll notice that we need to **Select a destination**

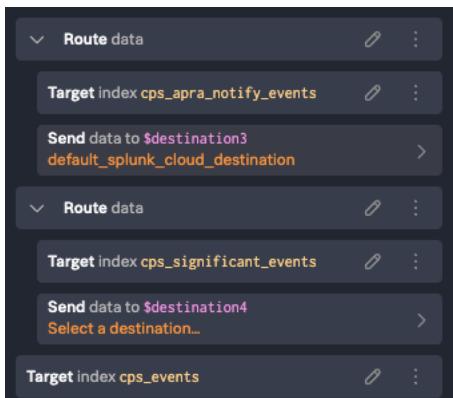


47. Click on Select a destination for \$destination3, a new pane will appear, select default_splunk_cloud_destination.

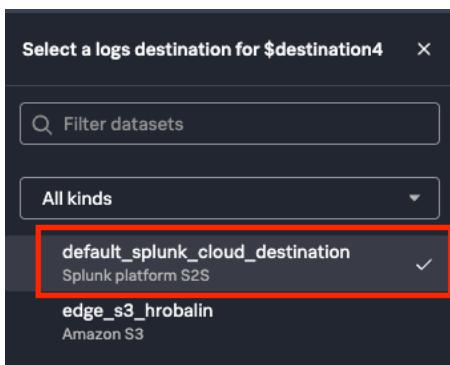


Click on **Apply** at the bottom right.

48. Let's confirm our selection is updated:



49. Click on Select a destination for \$destination4, a new pane will appear, select default_splunk_cloud_destination.

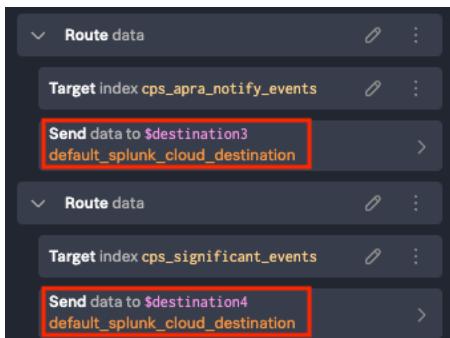


Click on **Apply** at the bottom right.

50. Let's confirm that we have configured both correctly.

Navigate to the 'Route Section' located under 'Actions' on the right-hand side (Make sure to scroll down).

We should see something like the following:



51. We have now configured two distinct destinations for specific events, enabling routing to indexes monitored by different teams and supporting use cases that demand rapid notification.

Add a Custom Indexed Field to Your Data

52. Finally, add a '**participant**' field to your events to help you identify them in Splunk Cloud.

In this case make sure to add the line to **before** the Route to specific indexes.

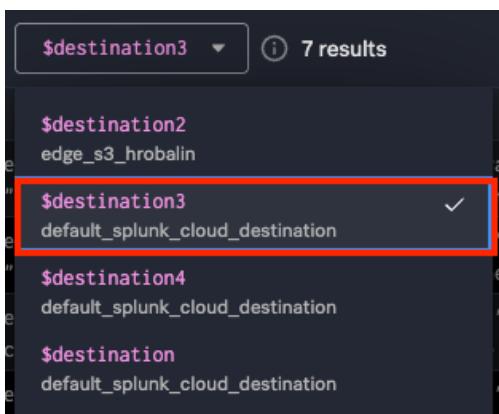
```
$pipeline =
| from $source
...
| eval _raw = json_set(_raw, "participant", "hrobalin") ← Update with your
  own name
//Route to different Indexes based on APRA Notifications
```

```
...
| index = "cps_events"
| into $destination;
```

Preview Your Pipeline

53. Test your pipeline by clicking on the blue Preview pipeline button (▶) in the top right corner of the screen. You should see that the pipeline window has a dropdown, this dropdown allows us to select the preview for all the different destinations configured.

We will select \$destination3



Now we should see something like the below:

| Previewing \$pipeline | |
|-----------------------|--|
| | \$destination3 |
| </> | _raw |
| 1 | {"critical_operation_name": "Superannuation Fund Administration", "impacted_operation": "Superannuation Fund Administration", "resolution_time_hours": "", "detection_method": "Internal Audit", "description": "Low incident impacting Superannuation Fund Administration: data loss detected. Immediate action required."} |
| 2 | {"critical_operation_name": "Customer Account Management", "impacted_operation": "Online Banking Services", "resolution_time_hours": "", "detection_method": "User Reported", "description": "High incident impacting Online Banking Services: security breach detected. Immediate action required.", "event_timestamp": "2025-10-31T14:00:00Z"} |
| 3 | {"critical_operation_name": "Customer Account Management", "impacted_operation": "Trade Settlement", "resolution_time_hours": "", "detection_method": "Monitoring Alert", "description": "Critical incident impacting Trade Settlement: system outage detected. Immediate action required.", "event_timestamp": "2025-10-31T14:00:00Z"} |

If we inspect the other destination such as \$destination and \$destination4, you will notice that the preview is empty. This is not a mistake.

The reason for this is that the CPS pipeline demonstrates selective event routing based on compliance criticality. The 'where' clause filters to only high-impact events: those flagged as APRA notification candidates (regulatory requirement) OR events with High/Critical severity. This significantly reduces data volume while preserving compliance-critical events. Most low-severity events with null severity ratings are filtered out before reaching the routing stage, which is why \$destination3 (APRA notifications) and \$destination2 (archive) see more traffic than \$destination4 (general significant events).

CPS Pipeline

54. Your CPS Pipeline SPL should look like the following (Be aware copying below will require updating env name on second line):

```

import route from /splunk.ingest.commands
import 'anz_critical_operations.csv' from /envs.splunk.'dmx-shw-a602e5f186a868'.lookups

$pipeline =
| from $source

    // Extract entity_id from _raw for lookup key building
    | eval lookup_entity_id = json_extract(_raw, "entity_id")

    // Lookup enrichment: match on entity_id from events to entity_id in CSV
    | lookup 'anz_critical_operations.csv' entity_id AS lookup_entity_id
OUTPUT system_id, critical_operation_name, tolerance_level, business_unit

    // Inject lookup results into _raw so enrichment travels with the event
    | eval _raw = if(isnotnull(system_id), json_set(_raw, "system_id",
system_id), _raw)
        | eval _raw = if(isnotnull(critical_operation_name), json_set(_raw,
"critical_operation_name", critical_operation_name), _raw)
            | eval _raw = if(isnotnull(tolerance_level), json_set(_raw,
"tolerance_level", tolerance_level), _raw)
                | eval _raw = if(isnotnull(business_unit), json_set(_raw, "business_unit",
business_unit), _raw)

    // Remove temporary top-level lookup fields to avoid duplication (after
injecting into _raw)
    | fields - lookup_entity_id, system_id, critical_operation_name,
tolerance_level, business_unit

    // Extract severity from _raw and convert to numeric for filtering
    | eval cps230_incident_severity = json_extract(_raw,
"cps230_incident_severity")
    | eval severity_numeric = case(
        cps230_incident_severity == "Critical", 4,
        cps230_incident_severity == "High", 3,
        cps230_incident_severity == "Medium", 2,
        cps230_incident_severity == "Low", 1,
        true, 0
    )

    // Filtering: keep events that are APRA notification candidates OR have
high severity

```

```

| where json_extract(_raw, "apra_notification_candidate") == true OR
severity_numeric >= 3

// Pseudonymize reporter_user_id
| eval reporter_user_id = json_extract(_raw, "reporter_user_id")
| eval hashed_reporter_id = if(isnotnull(reporter_user_id),
sha256(reporter_user_id + "cps230_salt"), null)
| eval _raw = json_set(_raw, "reporter_user_id", hashed_reporter_id)

// Mask vendor_name
| eval vendor_name = json_extract(_raw, "vendor_name")
| eval masked_vendor = if(isnotnull(vendor_name), "REDACTED", null)
| eval _raw = json_set(_raw, "vendor_name", masked_vendor)

// Drop temporary fields used for processing
| fields - lookup_entity_id, system_id, critical_operation_name,
tolerance_level, business_unit, reporter_user_id, hashed_reporter_id,
vendor_name, masked_vendor, cps230_incident_severity, severity_numeric

// Send raw data to secure archive (e.g., Amazon S3 Bucket)
| thru [ | into $destination2 ]

| eval _raw = json_set(_raw, "participant", "hrobalin")

// Route to different indexes based on APRA notification status
// Route APRA notification candidate events to APRA notifications index
| route _raw.apra_notification_candidate == true,
[
    | eval index = "cps_apra_notify_events"
    | into $destination3
]

// Route non-APRA events to significant events index
| route _raw.apra_notification_candidate == false,
[
    | eval index = "cps_significant_events"
    | into $destination4
]

| eval index = "cps_events"
| into $destination;

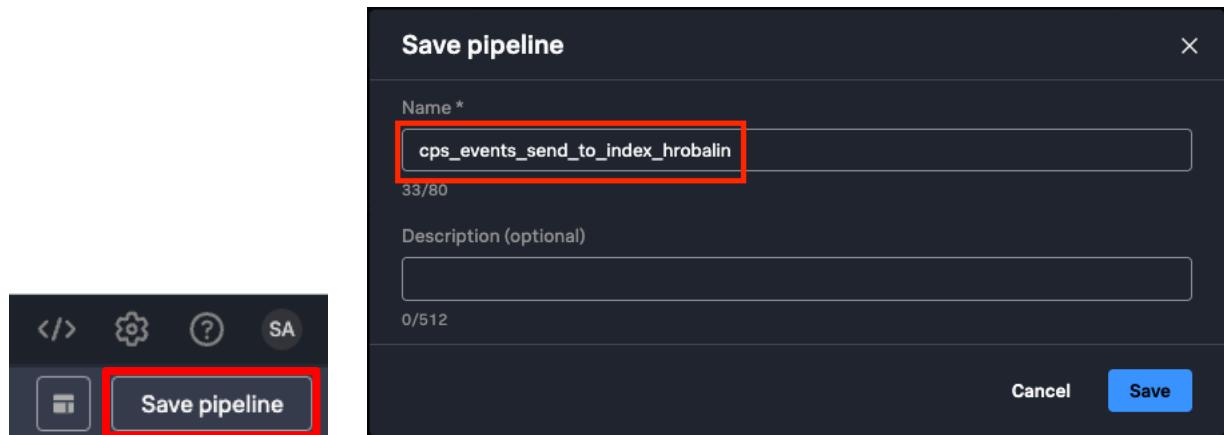
```

Save Your Pipeline

55. Save your pipeline by clicking on Save pipeline in the top right corner of the screen. Give your pipeline a suitable name, such as **cps_events_send_to_index_<yourName>** or something similar. As before, be sure to include your name or initials somewhere in the pipeline name to

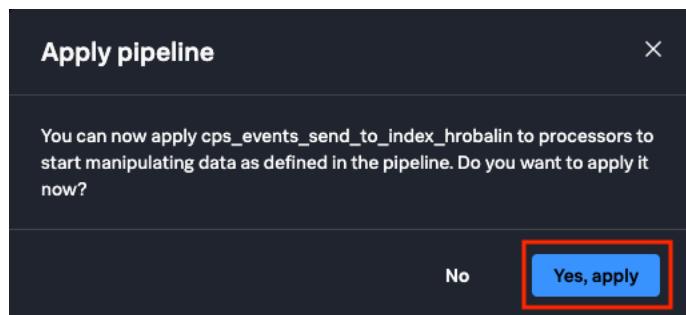


avoid confusion among those who are sharing the same environment as you!

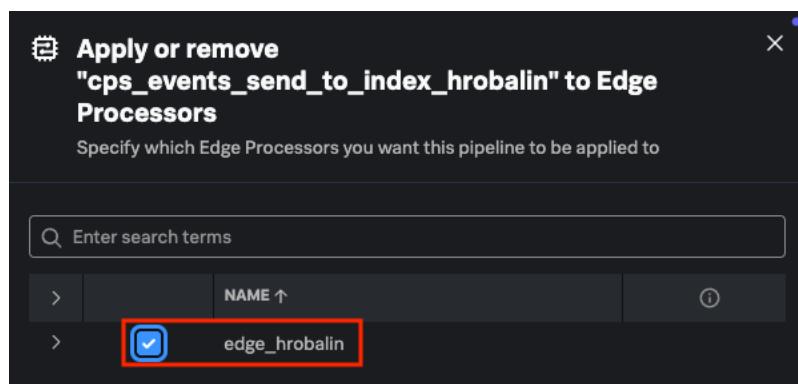


Apply the Pipeline to Your Edge Processor

56. Now an “Apply Pipeline” dialog box will appear, click on “Yes, apply”.



57. Check the box next to the name of the Edge Processor you created earlier and click on **Save**.



Confirm CPS Pipeline is applied

58. Make sure the CPS Events Pipeline is applied. Navigate to your Pipelines, you should see your pipeline name, and a status if Applied or Unapplied. See image below:

| Status | Name ↑ |
|---------|-----------------------------------|
| Applied | cps_events_send_to_index_hrobalin |

Check Your Data in Splunk Cloud

59. Log in to Splunk Cloud and open up the **Search & Reporting** app. Run the following search over the **last 15 minutes** and verify that you now see the redacted events.

```
index=cps_apra_notify_events participant=hrobalin ← Update the name!
```

| i | Time | Event |
|---|----------------------------|--|
| > | 11/8/25 12:15:25.137 AM | <pre>{ [-] apra_notification_candidate: true business_unit: [[+]] control_id_failed: null cps230_incident_severity: Critical critical_operation_id: CRITOP-6105 critical_operation_name: [[+]] description: Critical incident impacting Superannuation Fund Administration: data loss detected. Immediate action required. detection_method: User Reported entity_id: ENTITY_6885 estimated_downtime_minutes: 348 event_id: d55a0e41-66d5-489a-b910-0f981d16ddaf event_timestamp: 2025-11-07T15:15:25.137025Z event_type: operational_incident impact_description: Impact to Liquidity Management: performance degradation observed. impacted_operation: Superannuation Fund Administration incident_id: INC-B55085C3 participant: hrobalin reporter_user_id: bf291e80007f575a7b0b53515c6749925a5115b2be1efb975fe00b55bb2985ff resolution_time_hours: null root_cause_category: null service_provider_id: null severity: Critical source_system: MonitoringTool status: Investigating system_id: [[+]] timestamp: 2025-11-07T15:15:25.137025Z tolerance_level: [[+]] vendor_name: null }</pre> |

Note that since our search results will only contain those events that match our IP address range you may need to wait up to a couple of minutes for events matching that range to arrive at Splunk. If you don't get any results immediately try re-running the search after a minute or two.

60. You will notice that if you do a search for index=cps_events you will see no results for last 15 mins. If we expand the range to an hour the answer becomes obvious. We do get events, until we enabled the Edge Processor Pipeline for CPS. This is when our filtering becomes effective, and we can see a way that some events are not routed for storage, others are enriched before ingestion.



See image below for example.

A screenshot of the Splunk search interface titled "New Search". The search bar contains the query "index='cps_events'". The results section shows "4,690 events" from "1/29/26 3:55:00.000 PM to 1/29/26 4:55:10.000 PM" with "No Event Sampling". The timeline format is set to "1 minute per column". The time range is set to "Last 60 minutes". A red box highlights the search bar and the timeline format area. Another red box highlights the time range dropdown.

61. You will notice that if you do a search for index=cps_significant_events you will see no results. Even if you increase the time range. The reason is that so far the pipeline has not found an event that meets all the requirements in the ingestion data to allow to get ingested into Splunk.

You've finished!



Get Started with Splunk Data Management today

Request access to Edge or Ingest Processor

Fill out this [form](#) to request access to Data Management Pipeline Builders. If you already have access, you can directly navigate to it using the instructions on this form.

Join the Slack channel

Get direct access to the Data Management team, ask questions, get help, and collaborate with the Community. Request access [here](#).

Splunk Adoption Hub

Access curated [Splunk resources](#) to get started and maximize value—all in one place. Tailored to your needs and regularly updated by Splunk experts.