Splunk Federated Data Management Workshop

Lab Guide

Overview

This workshop will help you understand the power of Ingest Actions (IA), Edge Processor (EP), and Ingest Processor (IP) and how they help manage your data quickly as it streams into Splunk. You'll learn how to use these tools from a web browser and how to filter, mask, transform, enrich, and route data to multiple destinations. You'll also see that by using these tools, you can get faster feedback that your changes are working.

In this hands-on workshop, you will work with all three tools in a Splunk Cloud environment. This practical setting allows you to see where these components reside, when you should use them, and how they each process data. The interactive approach will ensure that you leave the workshop with a confident understanding of these tools and their applications.

Throughout the labs in this workshop, you will work with Cisco ASA logs. Using Ingest Actions, you will complete the base use case and build on it with Edge Processor and, finally, Ingest Processor.

Table of Contents

Base Use Case

The base use case you will use for all three labs follows. Edge Processor and Ingest Processor build on this. Those details are in their respective lab sections.

Ingest Cisco ASA Logs
The environment already has the Cisco ASA logs set up for you. You will need to work with them to achieve the lab goals.

Remove Events with Lower-Level Information
You will send information you don't need daily to S3 instead of a Splunk index. As part of your configuration, removing events with lower-level details is important to streamline your data and focus on the most relevant logs. You still want to keep these logs in a lower-cost object store for compliance. Send logs with the following information to S3 instead of your Splunk index:

1. Built inbound UDP connection.
2. Built inbound TCP connection.
3. Teardown UDP connection.
4. Teardown TCP connection.
5. Disallowing new connections.

Mask the User Name Field
Before indexing the Cisco ASA logs in Splunk, you want to mask the user name, which we regard as sensitive.

Lab Connection Info

Your workshop instance URL and credentials will be provided on a handout. Please refer to that on how to access the workshop.

Sample Data

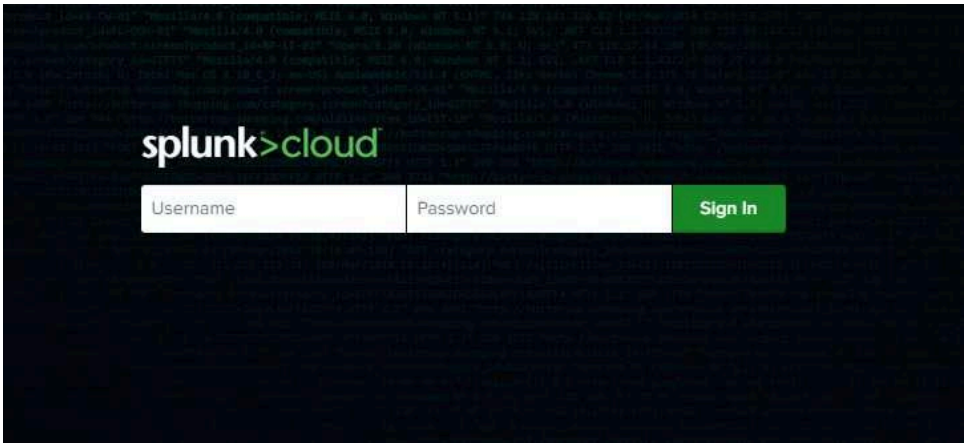
Sample data for this workshop can be downloaded below. Both links point to the same sample file and are provided for ease of access:

Direct Link: **cisco_logs_small.sample**

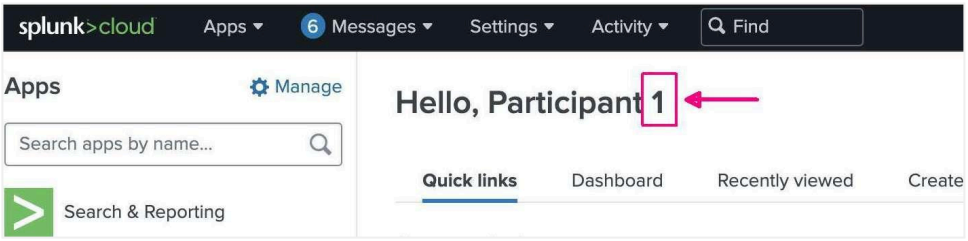
Accessing Your Environment

1. Connect to the lab environment as described in the section above. Labs with unique components may provide specific access instructions for those components.

2. Log into your assigned Splunk Cloud instance via the URL provided. After logging in, you may see a Terms of Service screen. If so, click the “Accept” checkbox as appropriate and continue.



3. Once logged into your Splunk Cloud instance, you’ll see this screen.



   a. The highlighted number corresponds to the sourcetypes you will use throughout the labs. In the example image above, your number would be `1`, so you would use the following sourcetypes through the lab:
      i. Ingest Actions lab uses the `cisco:asa:ia:#` sourcetype.
      ii. Edge Processor lab uses the `cisco:asa:ep:#` sourcetype.
      iii. Ingest Processor lab uses the `cisco:asa:ip:#` sourcetype.

4. The lab is set up this way so lab participants don’t interfere with each other.

5. **Write your participant number down, as you will need it for all labs.**

Lab Exercise 1: Ingest Actions

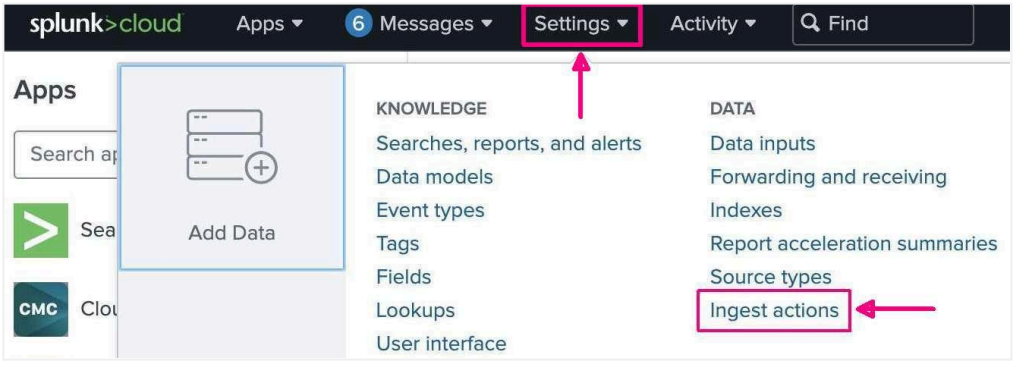
Description

In this lab, you will filter, mask, and route Cisco ASA logs using Ingest Actions ingest-time transformation and routing rules. You will implement the Base Use Case described at the beginning of this guide.

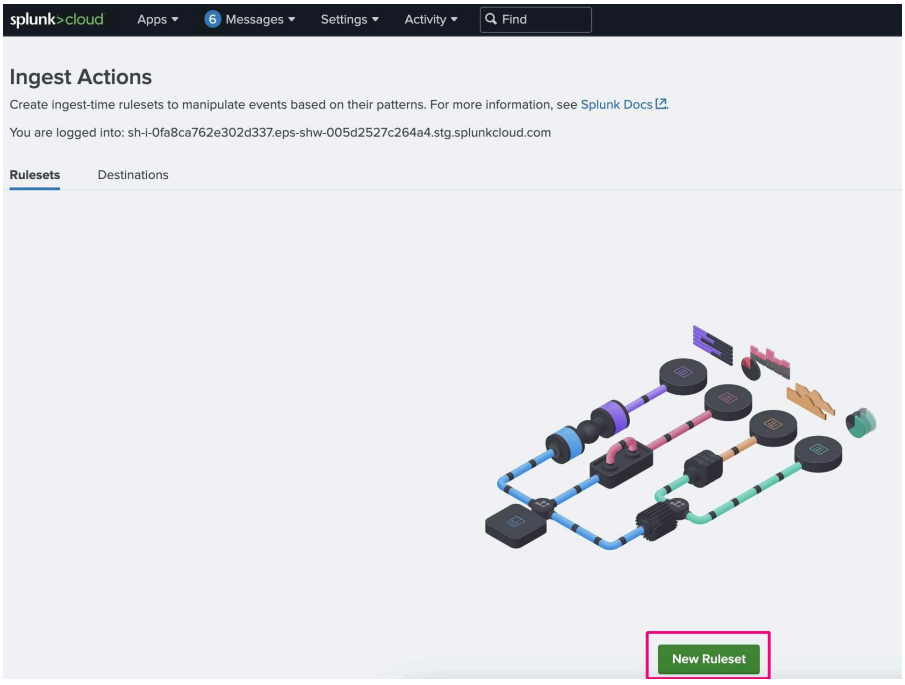
Accessing Ingest Actions

1. Be sure you've read and followed instructions from the Lab Connection Info and Accessing Your Environment above.

   Log in to your Splunk Cloud environment. Browse to the **`Settings`** menu at the top of your screen and choose **`Ingest Actions`** in the upper right area of the menu, as shown in the image below.



2. This will take you to the main Ingest Actions configuration page. Click on the **`New Ruleset`** button near the bottom of your screen.

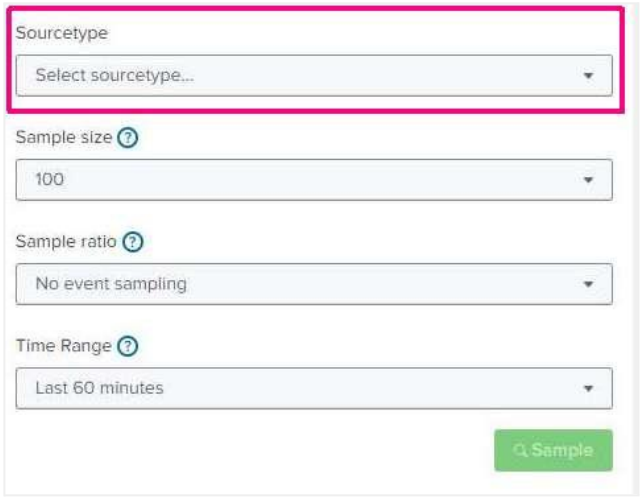


Creating a Ruleset

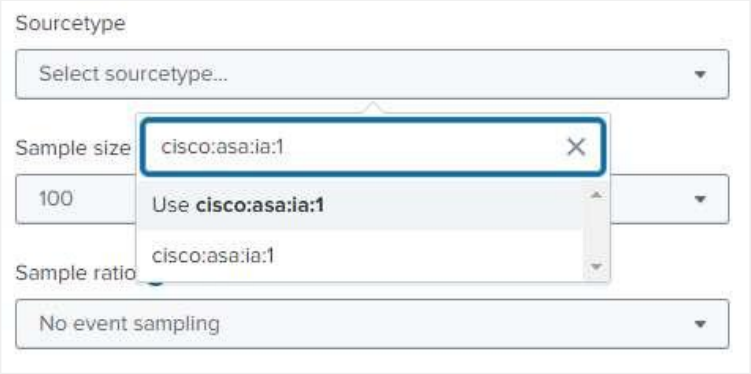
3. You're now on the *Create New Ruleset* configuration screen. The first thing to do is name your ruleset where you see **`Enter Ruleset Name`**. Since multiple people are using the same host as you, simply use **`Participant #`**, **REPLACING # WITH YOUR NUMBER FROM THE Accessing Your Environment SECTION.**

Select Sample Data

4. Looking down this screen, find the **`Sourcetype`** pull-down menu, as shown in the image below.



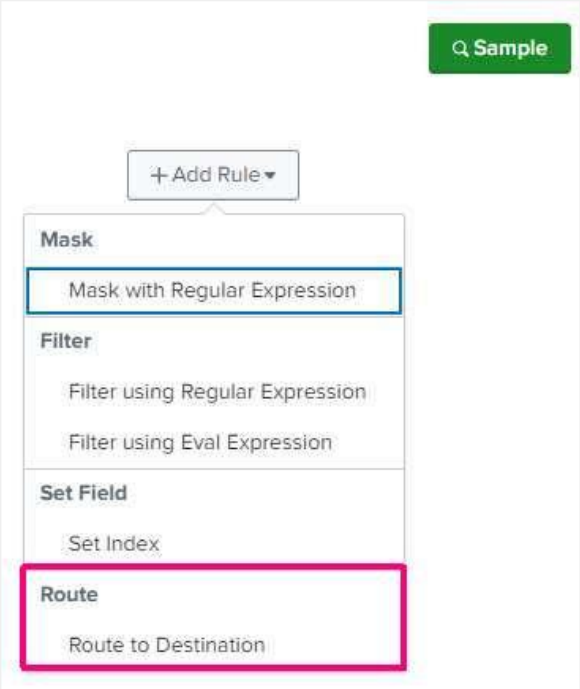
5. Expand the **`Select sourcetype`** menu by clicking on it. You will see several sourcetype categories. In the filter text box at the top, type in **`cisco:asa:ia:#`** , **REPLACING # WITH YOUR NUMBER FROM THE Accessing Your Environment SECTION**. Finally, click on **`Use cisco:asa:ia:#`** in the pull-down to select it for use. See the image below for an example.



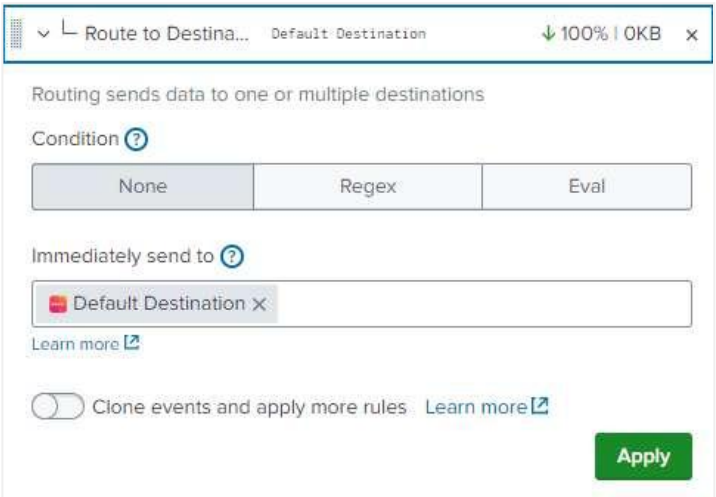
6. Now, set the **`Sample size`** to *5000* and the **`Time Range`** to *Last 7 days* so you get a good sample of data. Finally, click the **`Sample`** button to bring in data to work with. You should see data populated on the right side of the screen.

Route Data to S3

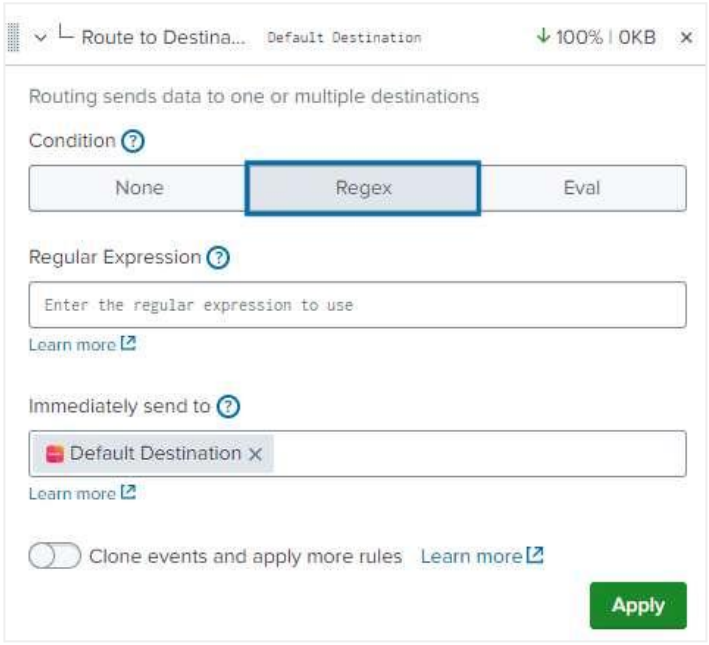
7. Now, we can add rules to our ruleset. You'll see several choices when you click the **`Add Rule`** dropdown, as shown below. Remember, the first thing you want to do is send low-level information events to S3. This means you want to **`Route to Destination`**.



8. Selecting **`Route to Destination`** will add a new rule to your ruleset, as shown in the image below.



9. Since we want to filter out events with various text, let's use a Regex to do that. Click the Regex button to show a textbox where you can enter a regular expression to target specific events

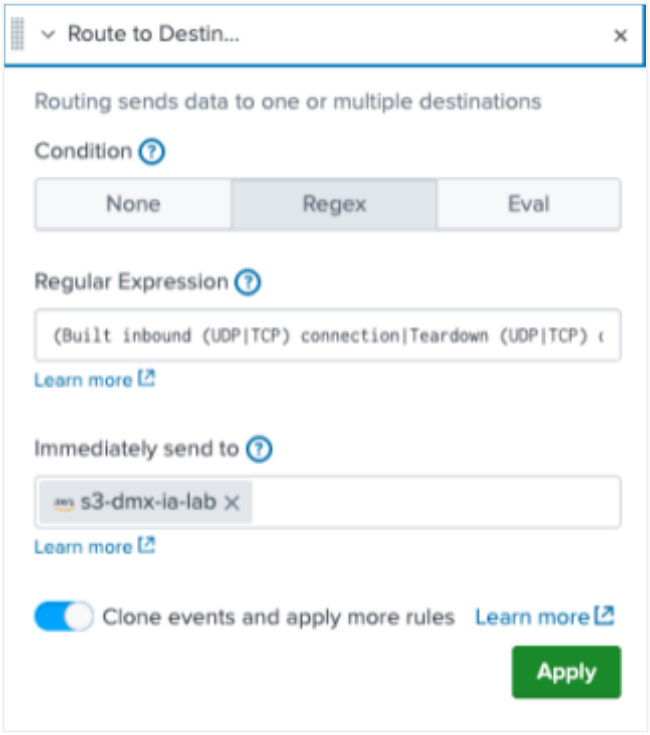


10. Refer back to the Base Use Case in the "*Remove events with lower-level information*" section on what events you need to target. Create a regular expression to match all of them and put it in the **`Regular Expression`** text field.

NOTE: If you need help creating a regular expression, you can use this one:

```
(Built inbound (UDP|TCP) connection|Teardown (UDP|TCP)
connection|Disallowing new connections)
```

11. Next, you'll choose the destination to send events to and make sure to "clone" events so you can apply more rules.
    a. For the destination, you will choose the S3 location called `s3-dmx-ia-lab` and ensure you toggle the `Clone events and apply more rule`s. If you don't toggle cloning, any rules after this point won't apply.
    b. Also, remove the `Default Destination` so events don't get sent to a Splunk index.
    c. See the image below for how your configuration should look.

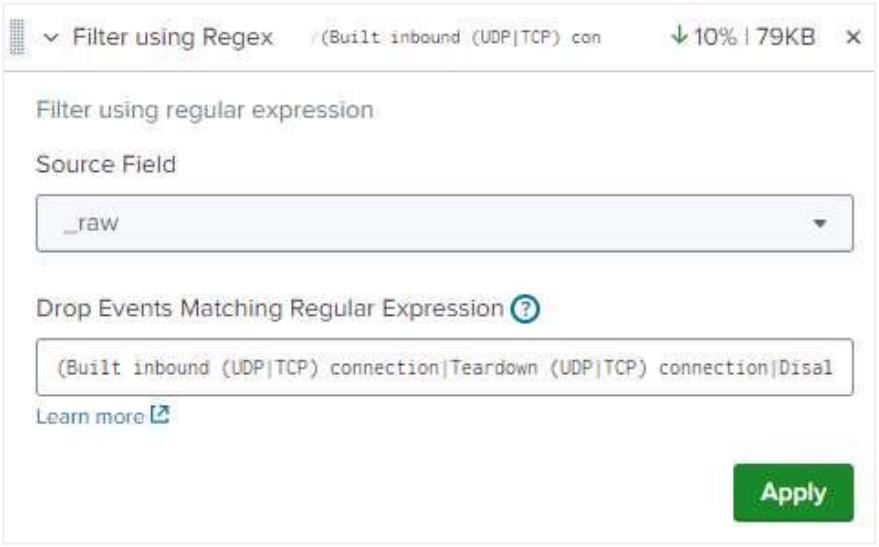


NOTE: Make sure you click the "Apply" button after configuring each rule. Otherwise, Ingest Actions will not save the configuration when you move on to the next rule.

12. Once you click the "Apply" button, look to the upper right of the events view on the right side. You will see three buttons: *All Events*, *Affected Events*, and *Unaffected Events*. These can help you ensure your rules are working properly.
    a. **All Events** shows all events being sampled.
    b. **Affected Events** shows which events are affected by the current rule.
    c. **Unaffected Events** shows which events are not affected by the current rule.
    d. The numbers you see next to these will likely not match the screenshot below, which is OK. In the image, we can see that out of 500 events, the S3 destination routing regex affected 44 events and did not affect 456 events. This means we'll get those 44 events in our S3 bucket.

Filter Data

13. Now that events are properly routed to S3, let's add the next rule. Since you only routed the events to S3 and didn't filter them, they will still pass through the event cloning and end up in a Splunk index, which you don't want. This means you need to filter them out.
    a. Click **`Add Rule`** and choose **`Filter using Regular Expression`** to prevent them from being indexed in Splunk. The great part is that you already created the regular expression you need in the routing rule.
    b. Put the same regular expression in the **`Drop Events Matching Regular Expression`** text box and click the *Apply* button.
    c. You should see the same number of Affected and Unaffected events as before since you're filtering the same events you routed to S3.
    d. In the upper-right corner of the below image, you can also see the 10% reduction in sample data after we filter events.



NOTE: Use the same regular expression as you did in the route to S3 rule:

```
(Built inbound (UDP|TCP) connection|Teardown (UDP|TCP)
connection|Disallowing new connections)
```

Mask Data

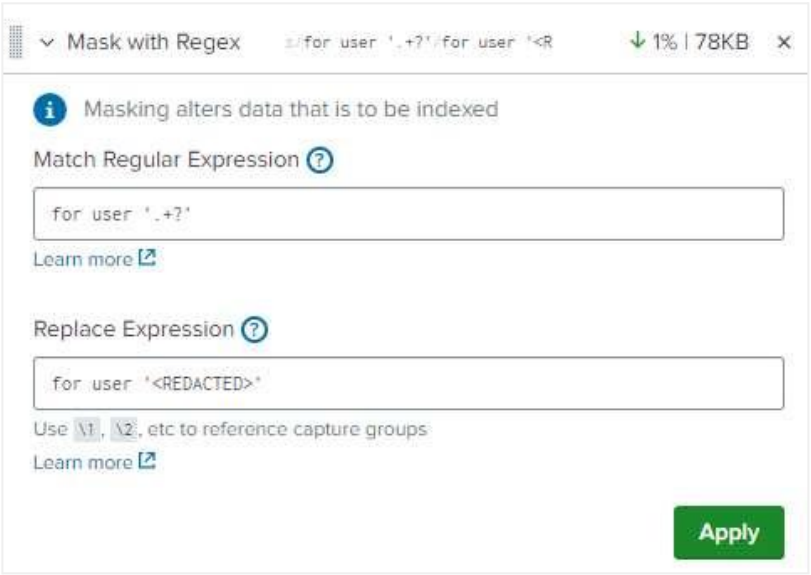
14. Now that you've sent the low-value data to S3 **and** filtered those events out of the data stream, you need to mask the username value in the events before indexing the data to Splunk. There are several places where a username value exists, so you will use three rules to target them. For this, we will replace the user name with a value of <REDACTED>.
15. Add another rule, this time choosing **`Mask with Regular Expression`**. You will see this rule has two inputs - one for the match and one for the replacement. Create a regular expression to match the username. Then, enter in the replacement. Remember, you want to replace the username value with **`<REDACTED>`**.
16. Remember to click the "Apply" button after configuring the rule.

NOTE: These events look like this:

```
Authorization denied (acl=acmetechinbound) for user 'admin'
from
```

NOTE: If you need help creating a regular expression, you can use this one:

Match Regular Expression: `for user '.+?'`
Replace Expression: `for user '<REDACTED>'`



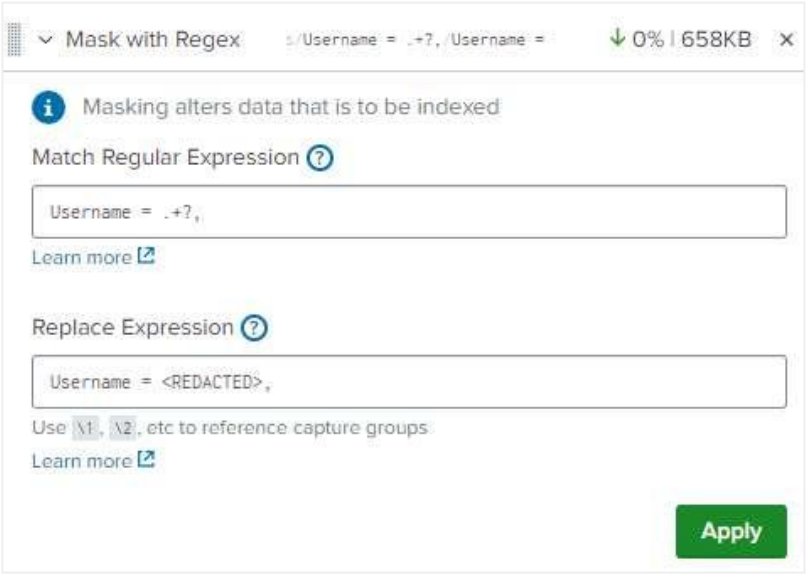
17. Add another **`Mask with Regular Expression`** rule to target the next username value events.

NOTE: These events look like this:

```
Group = Acme_Techcde, Username = practical_lovelace, IP =
10.11.36.23,
```

NOTE: If you need help creating a regular expression, you can use this one:

Match Regular Expression: `Username = .+?,`
Replace Expression: `Username = <REDACTED>,`



18. Add another **`Mask with Regular Expression`** rule to target the next username value events.

NOTE: These events look like this:

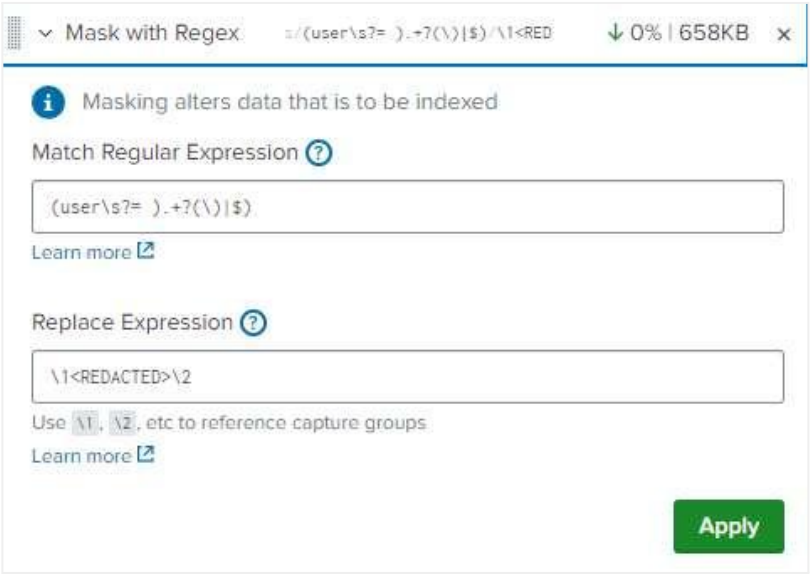
```
local database : user = focused_euclid
```

And this:

```
between 10.11.36.18 and 10.11.36.13 (user=
intelligent_haslett) has been created.
```

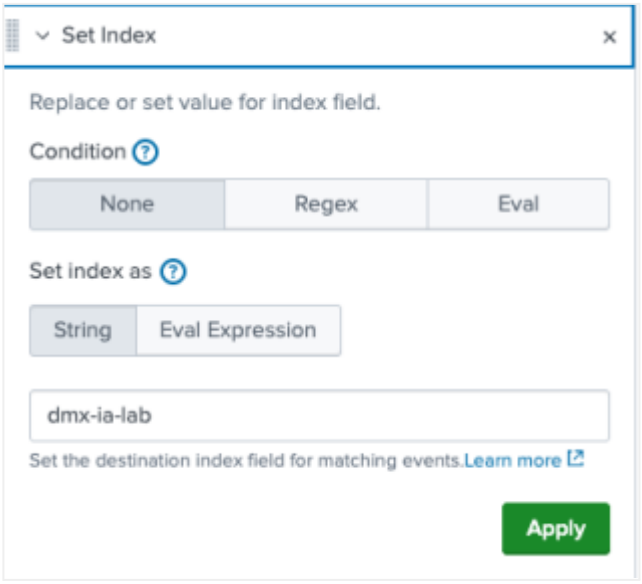
NOTE: If you need help creating a regular expression, you can use this one:

Match Regular Expression: `(user\s?= ).+?(\)|$)`
Replace Expression: `\1<REDACTED>\2`

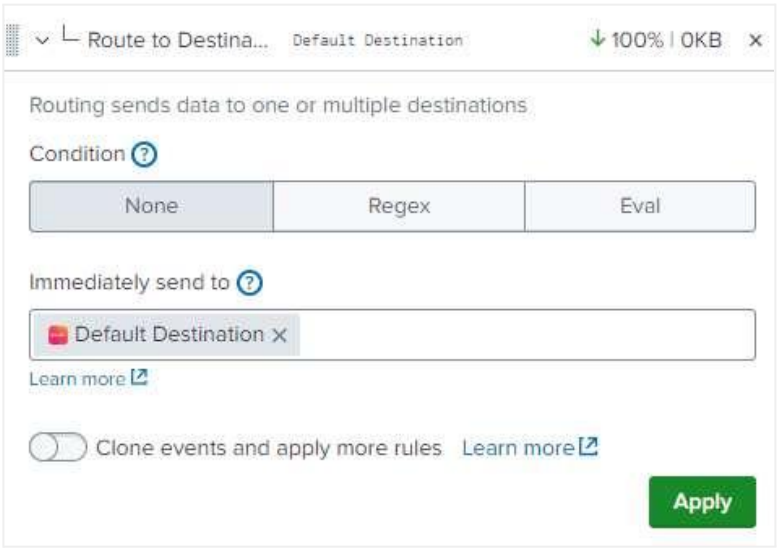


Send Data to a Splunk Index

19. Now, you can send the filtered and masked data to a Splunk index. For this, set the index to **`dmx-ia-lab`** and route the data there. You do this by adding a **`Set Index`** rule to the ruleset.
    a. Since you are setting the index name to a static value, you can leave the rule configurations to default settings and put the index name in the text box.
    b. Since you didn't do a *Regex* or *Eval*, this affects 100% of the remaining events.



20. Almost there - you just need to route the remaining data to Splunk. This is done similar to how you routed data to S3. Add a new rule and choose **`Route to Destination`**. You'll see the destination as **`Default Destination`**. This means that the events will get re-inserted into Splunk's normal pipeline at the indexing stage. Since you set the index to **`dmx-ia-lab`**, that's where the data will get indexed.

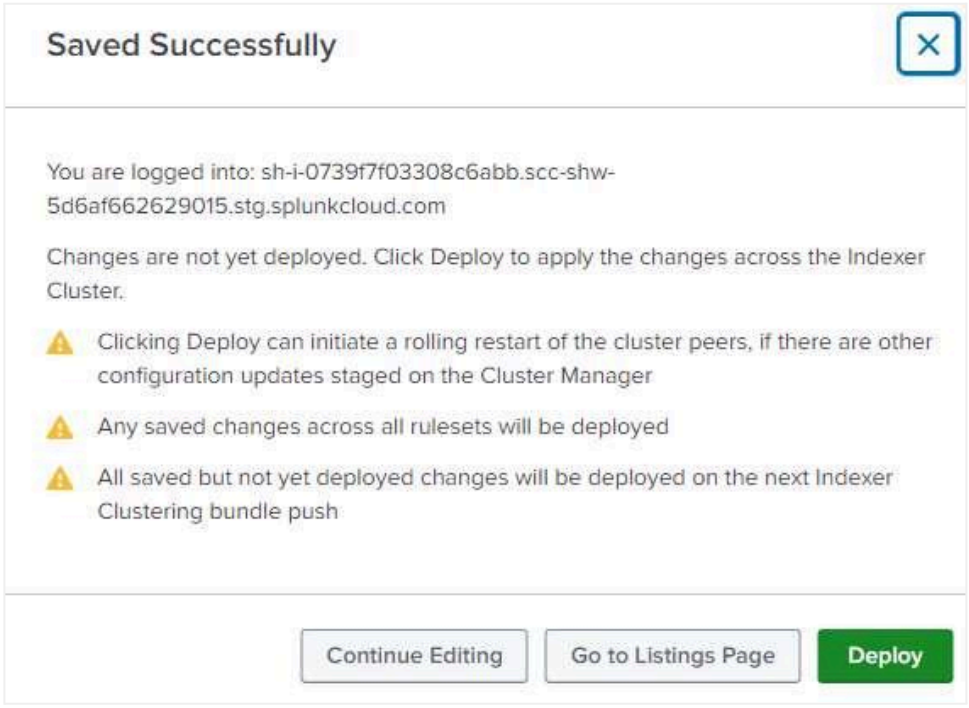


NOTE: Make sure you click the "Apply" button.

Save and Check Your Work

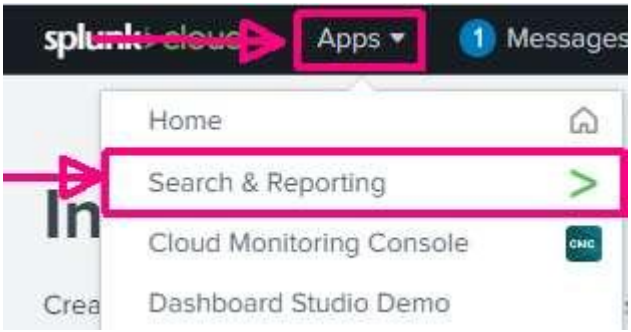
21. Now that the ruleset is complete, in the upper-right corner of the screen, click the **`Save`** button.

22. A modal window may open, as shown below. If it does, click the green **`Deploy`** button to send your changes to the indexers. Once set, you can check your work.



NOTE: Depending on the Splunk Cloud experience and version, the modal window may not appear, and rulesets will be deployed upon saving.

23. In the upper-left corner of the screen, select **`Apps`** menu and then select the **`Search & Reporting`** app. This will take you to a search bar where you can ensure the correct data ended up in the correct index.



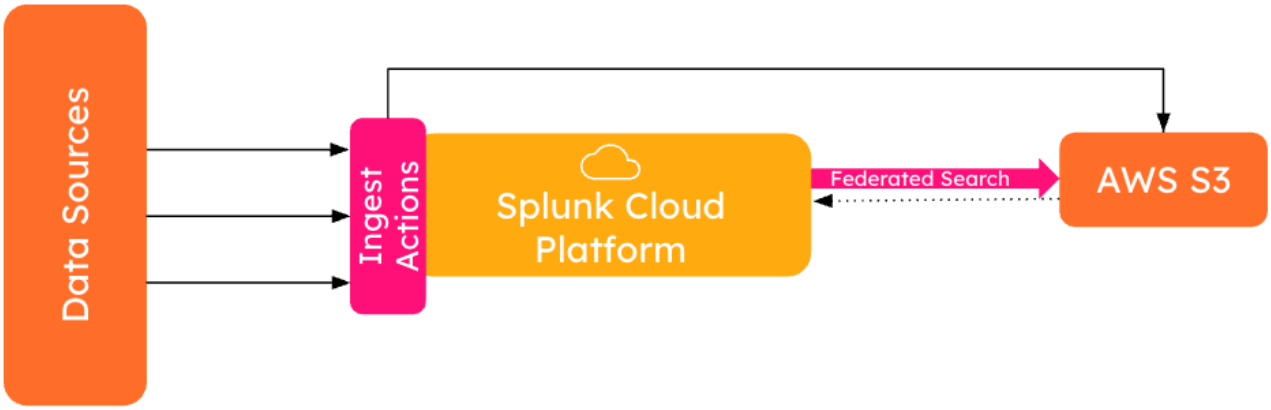
> **NOTE**: You may see a modal window pop up to take a tour of Splunk. In the lower-right corner of the pop-up, click the “Skip tour” link if this happens.

24. Use the search below, replacing the sourcetype **#** portion highlighted in yellow **WITH YOUR ASSIGNED NUMBER**. You will see all **`user`** field values are set to **`<REDACTED>`**.

```
index=dmx-ia-lab sourcetype=cisco:asa:ia:# earliest=-7d@d
```

Splunk Federated Search for Amazon S3

Federated search for Amazon S3 lets you search data in your Amazon S3 buckets from your Splunk Cloud Platform deployment.



25. Use the S3 federated search below, replacing the sourcetype **#** portion highlighted in yellow **WITH YOUR ASSIGNED NUMBER**, to see the data you sent to the S3 bucket.

```
| sdselect * FROM federated:s3-ia-lab-index WHERE
sourcetype="cisco:asa:ia:#"
```

Lab Exercise 2: Edge Processor

Description

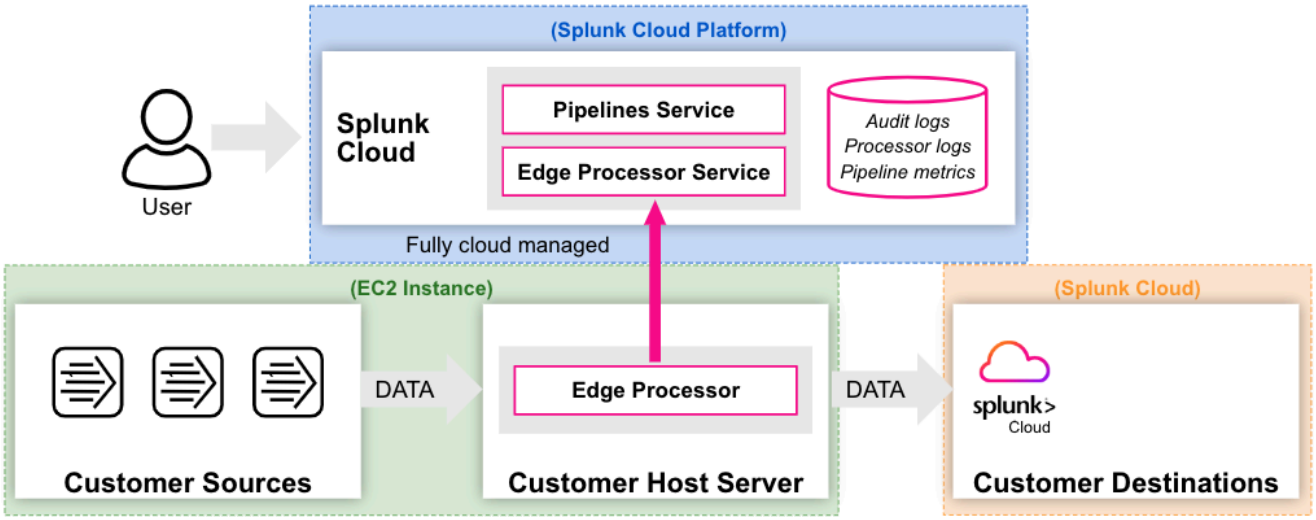
In this lab, you will filter, mask, and route Cisco ASA logs using Edge Processor. You will implement the Base Use Case described at the beginning of this guide. Additionally, the lab guide will demonstrate the ability to enrich log events using lookup commands.

An environment and an Edge Processor have been set up, and you have been provided with the credentials. In case you are interested in architectural background, 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 set up, configure, and manage Edge Processors, including creating pipelines for data processing (more on this later)!

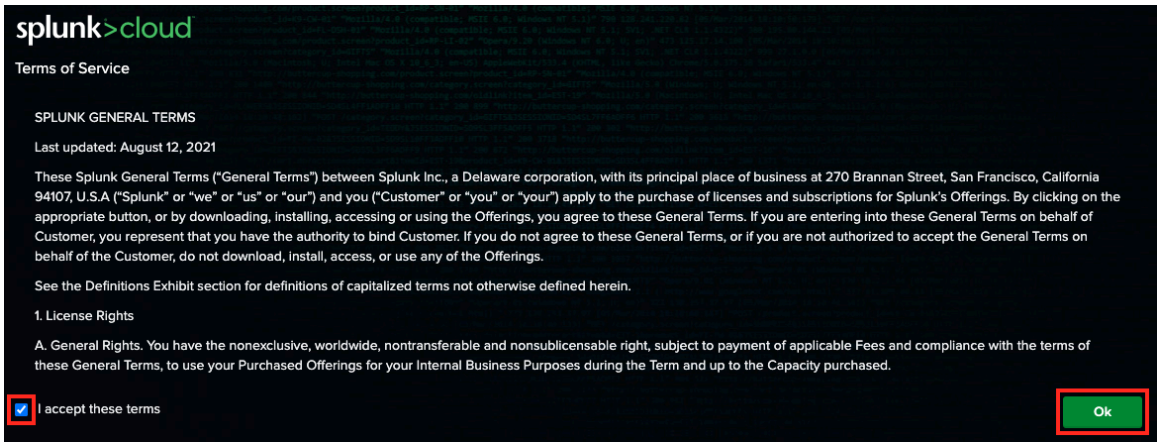
EC2 Instance (with demo data) - An Edge Processor node has been installed for you prior to this exercise in an EC2 instance. The edge processor is configurable via the cloud-based Edge Processor service. Processed data will be forwarded to Splunk Cloud. In the lab environment, this is referred to as 'EP - Edge Node'.



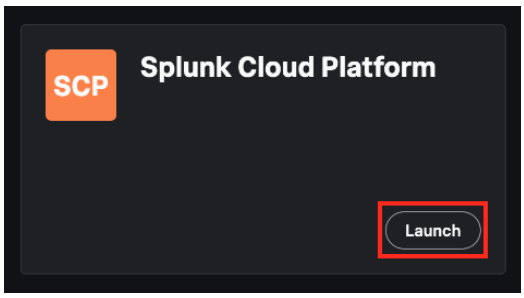
Access Your Splunk Cloud Instance

1. As part of the exercise, You will be given credentials to the EP that has been set up. Use the admin credentials provided here to log in to your account.

2. Accept the Terms of Service and click on **OK**.



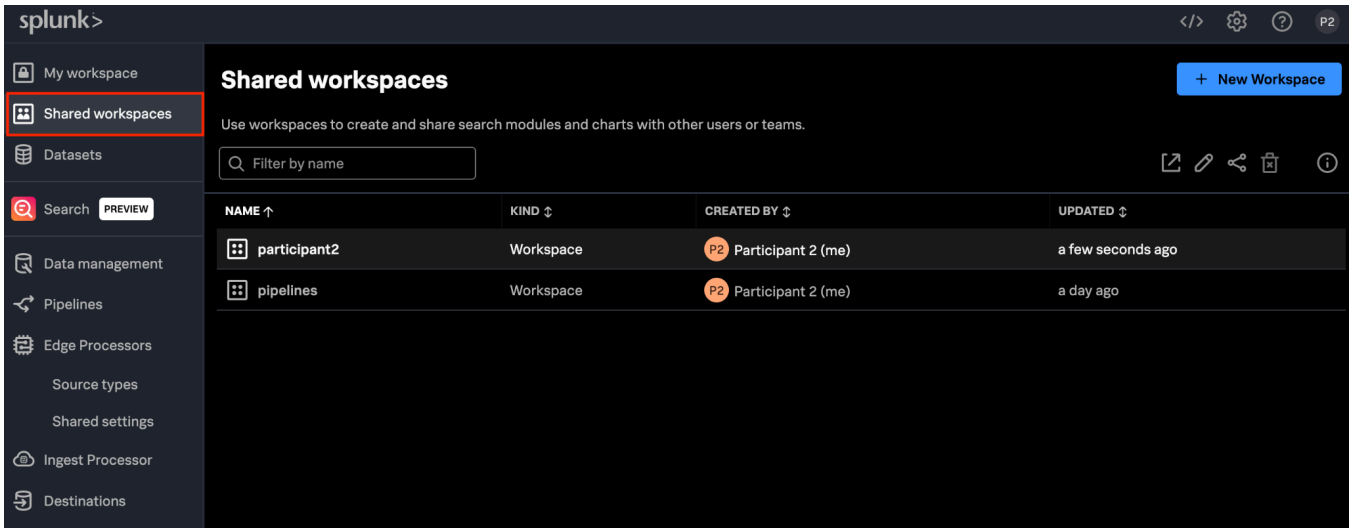
3. On the welcome page, click on **Launch** under **Splunk Cloud Platform**.



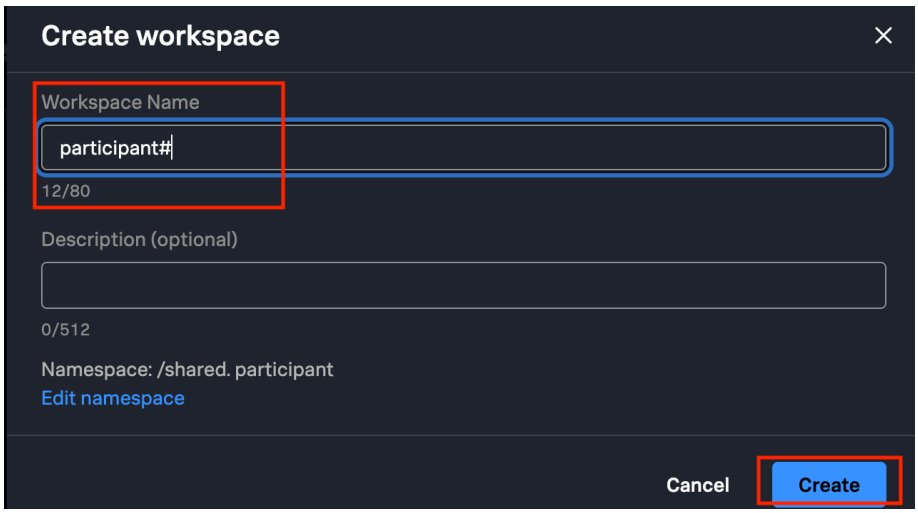
You will now be taken to the data management environment.

Create a Workspace

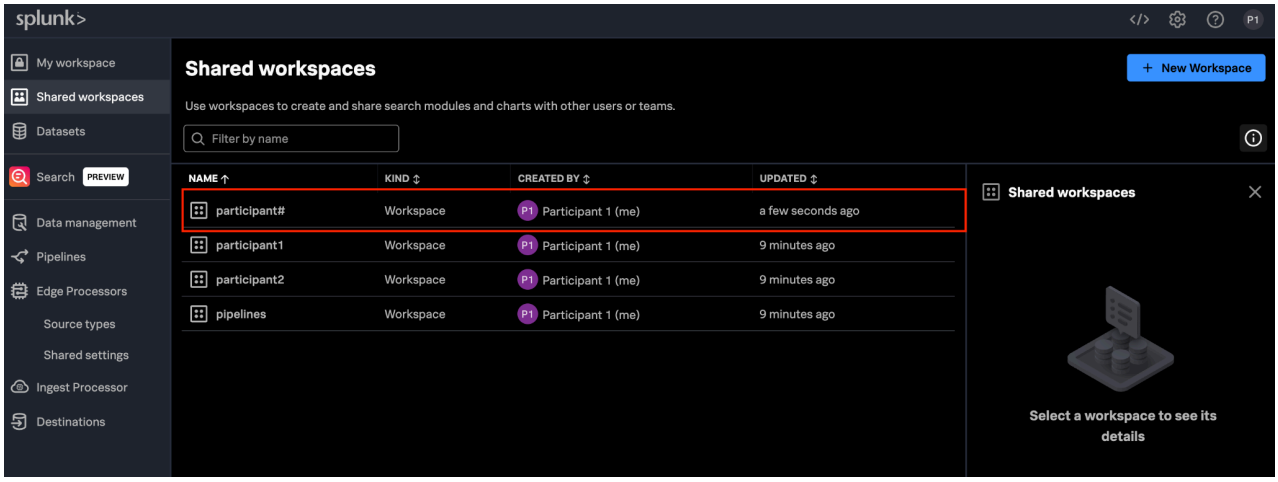
4. Enter the shared workspace by clicking on the Shared Workspaces button on the left side.



5. Create a new workspace for your participant by pressing the **New Workspace** button in the upper right. Name the workspace **participant# (REPLACE # WITH YOUR NUMBER)**. Select the **Create** button. You can skip this step if you have already created this workspace in the past.

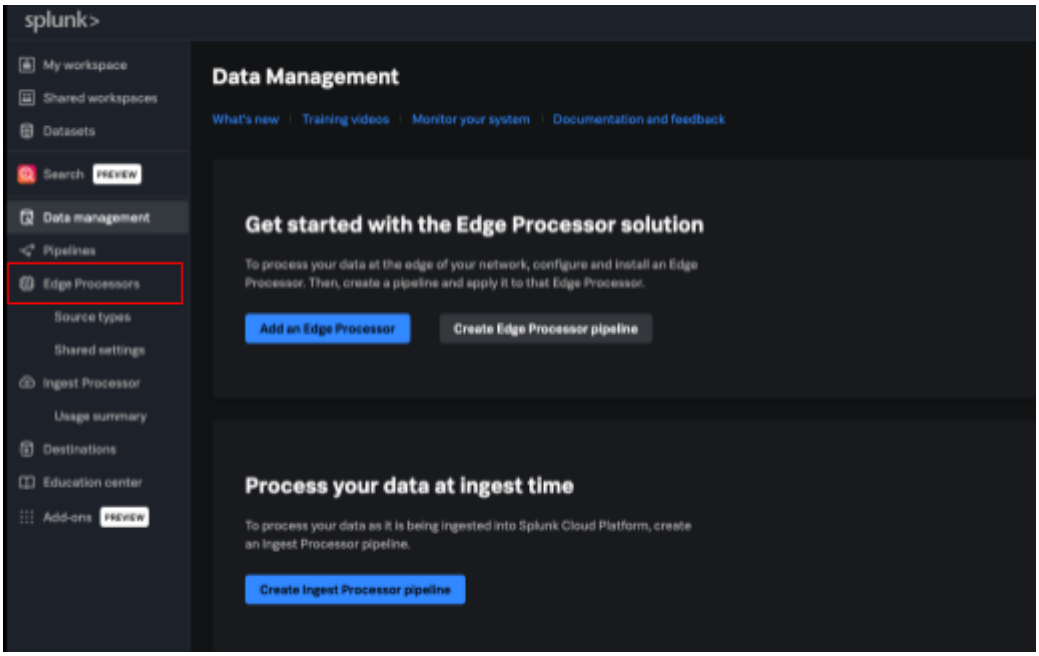


6. Double-click the workspace you created to enter into the workspace.



Add a new Edge Processor

7. In your home page, from the menu on the left of the page click on **Edge Processors**.



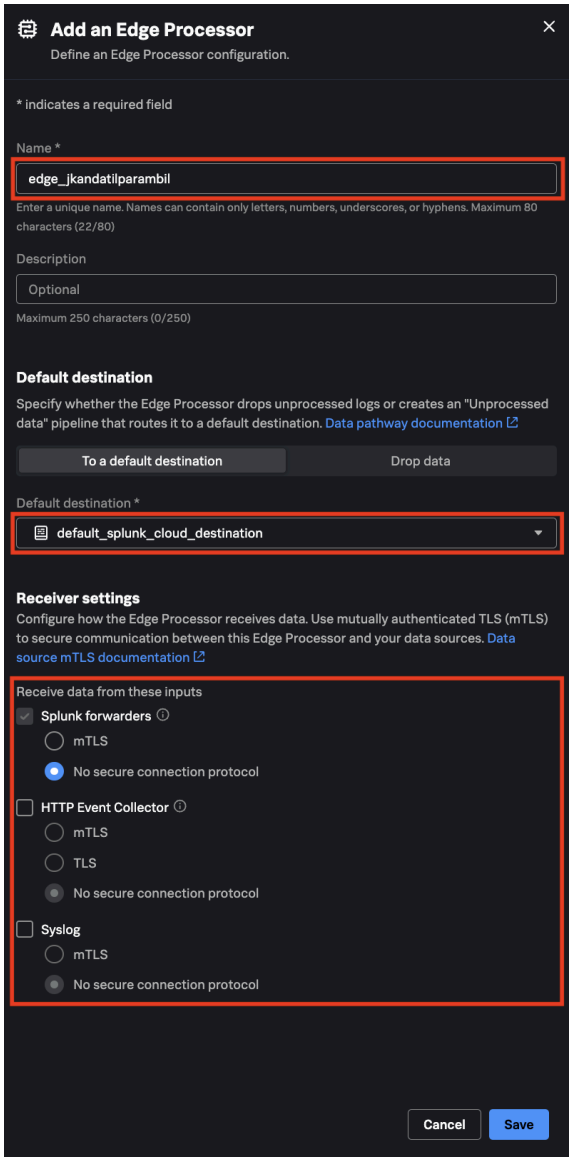
8. In the middle of the **Edge Processors** page click on **+ New Edge Processor**.

9. 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*

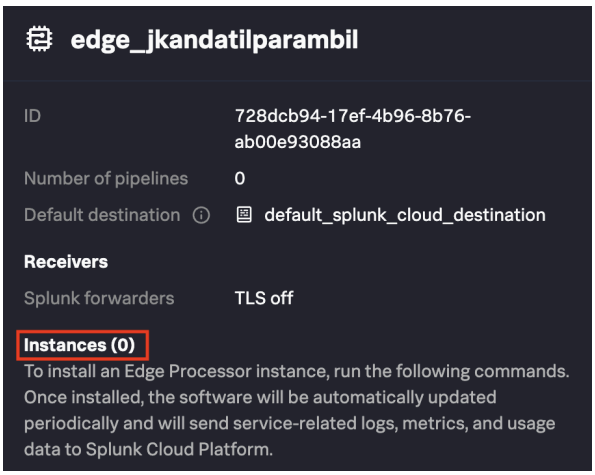
   For Default destination select '**default_splunk_cloud_destination**'.

   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.



10. You will now see a pane detailing information about your Edge Processor on the right hand side of the page. We also see configuration information about our Edge Processor and can see "**Instances (0)**", meaning that there are currently no Edge Processor instances installed.



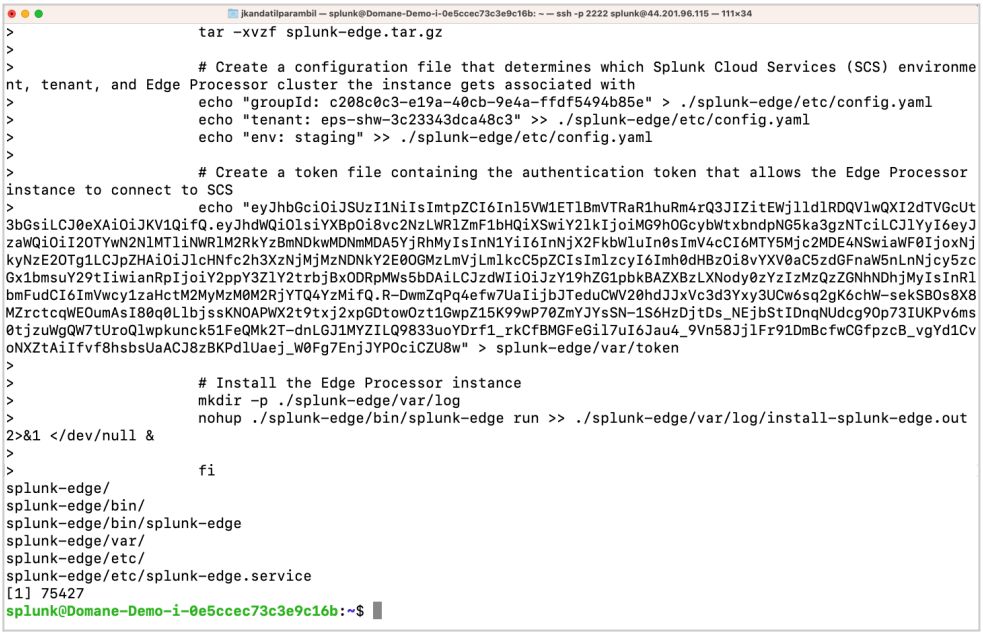
11. Under "**Instances**" information, you will see a clipboard containing the installation commands, which we can use to install an Edge Processor instance onto a Linux server.

    Copy this command by clicking on the  icon.

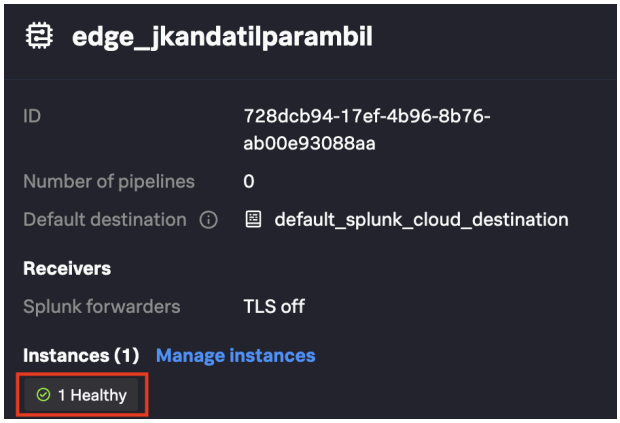
Install a New Edge Processor Instance

12. 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.

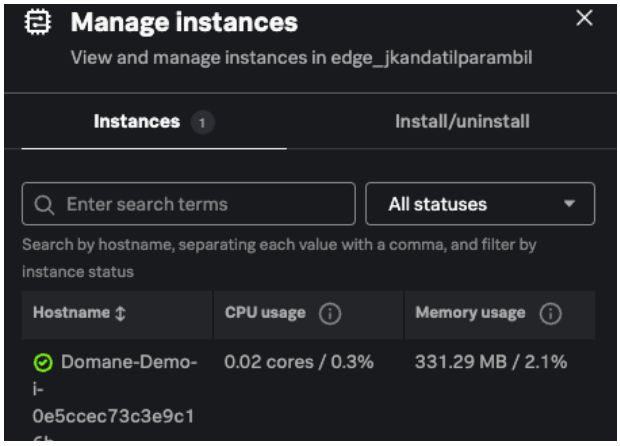
    Due to the way the command is formatted, you will need to hit enter a second time after the initial command has run. You should now see a green prompt like before.



13. 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.



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.



ℹ️ 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)

14. 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:

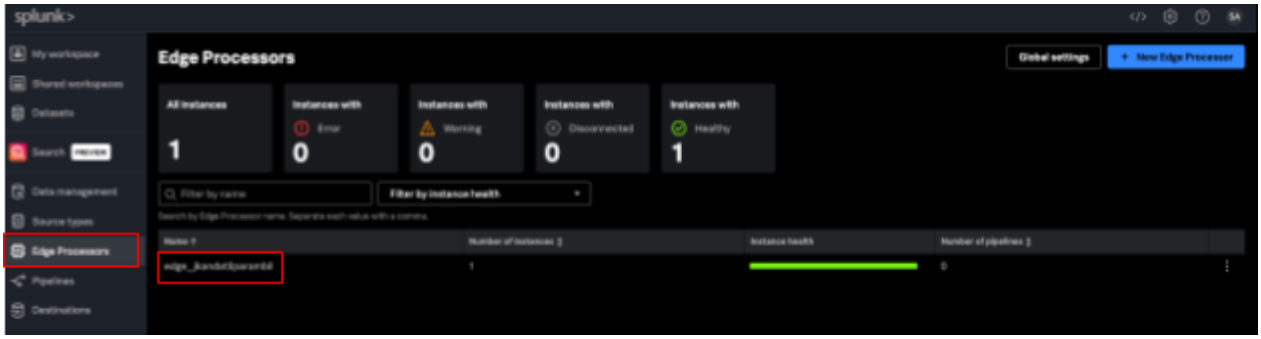
```
sudo /opt/splunk/bin/splunk restart
```

    Confirm that Splunk restarts successfully:

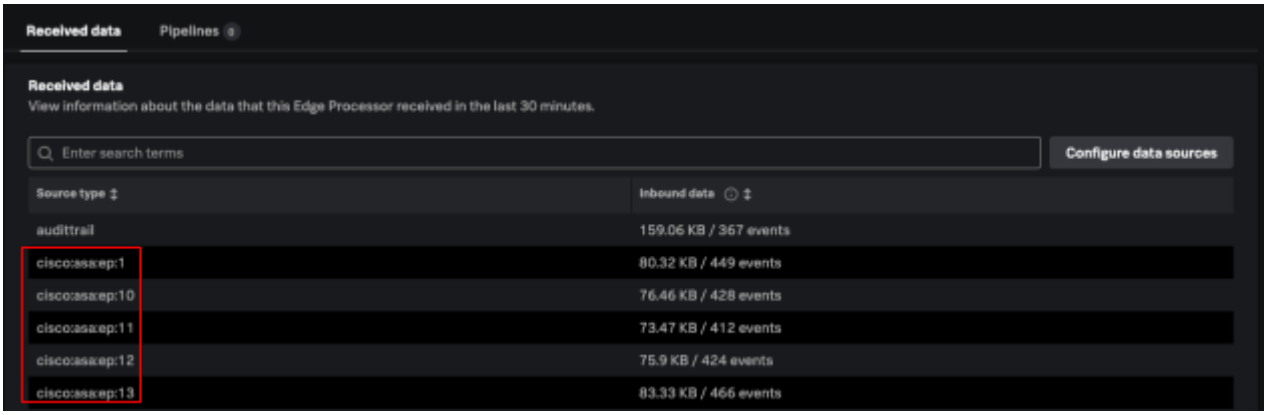
```
If you get stuck, we're here to help.
Look for answers here: http://docs.splunk.com

The Splunk web interface is at https://Domane-Demo-i-03397b9fa7444e973

splunk@Domane-Demo-i-03397b9fa7444e973:~$
```

Verify Edge Processor is Receiving Data

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

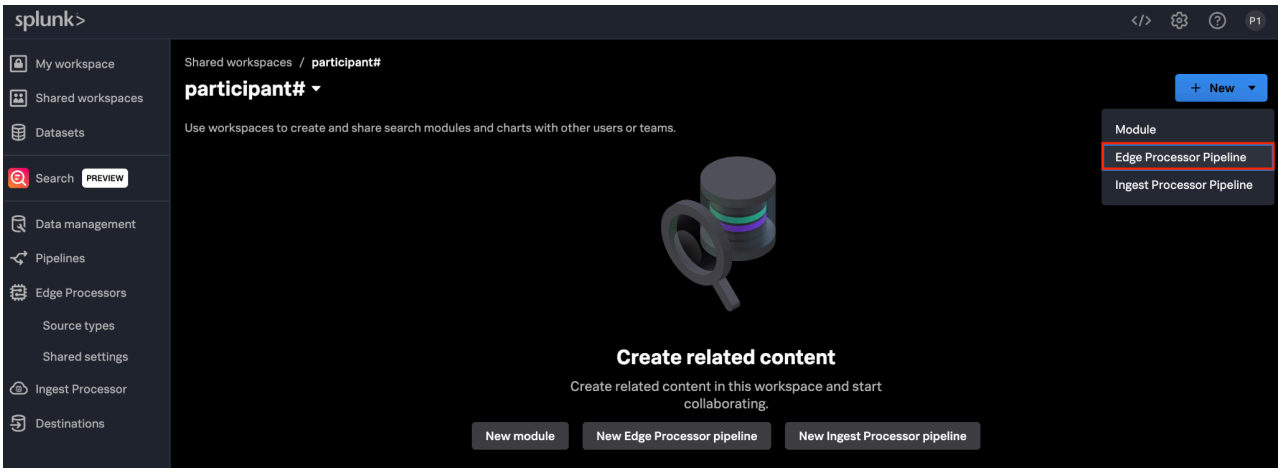


16. Under the **Data sources** pane on the left side of the page, click on **Received data** and verify that your Edge Processor is receiving inbound **cisco:asa:ep:#** source type data.

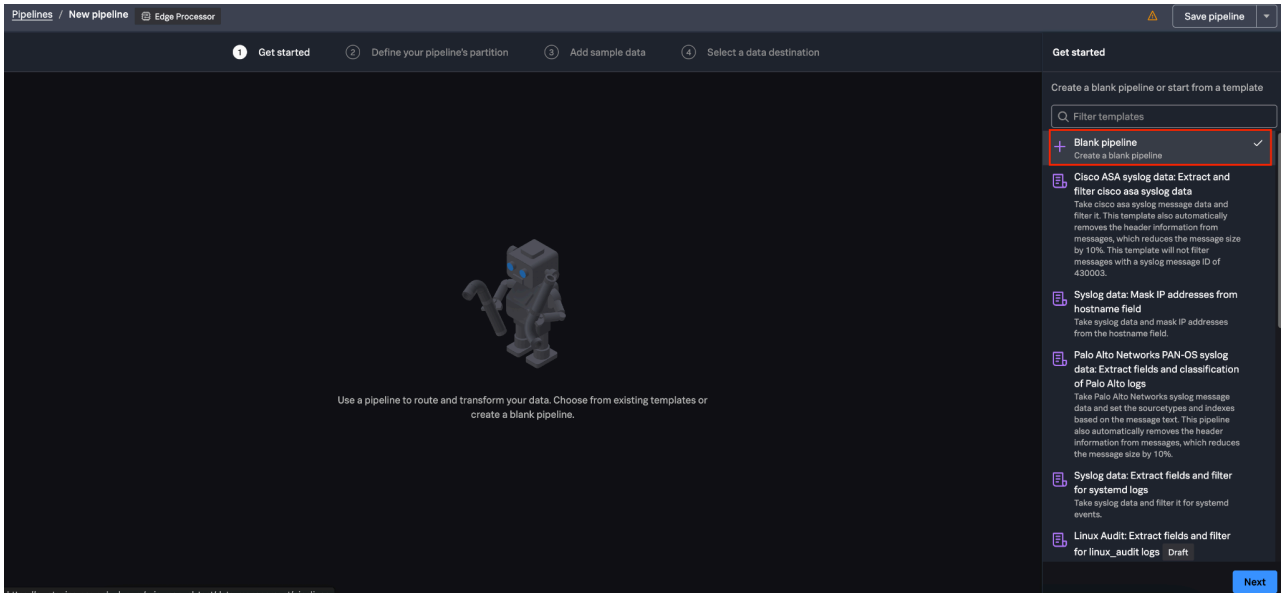


Create a Pipeline

17. In your participant workspace, create a new EP pipeline by selecting **New** → **Edge Processor Pipeline** in the upper right.

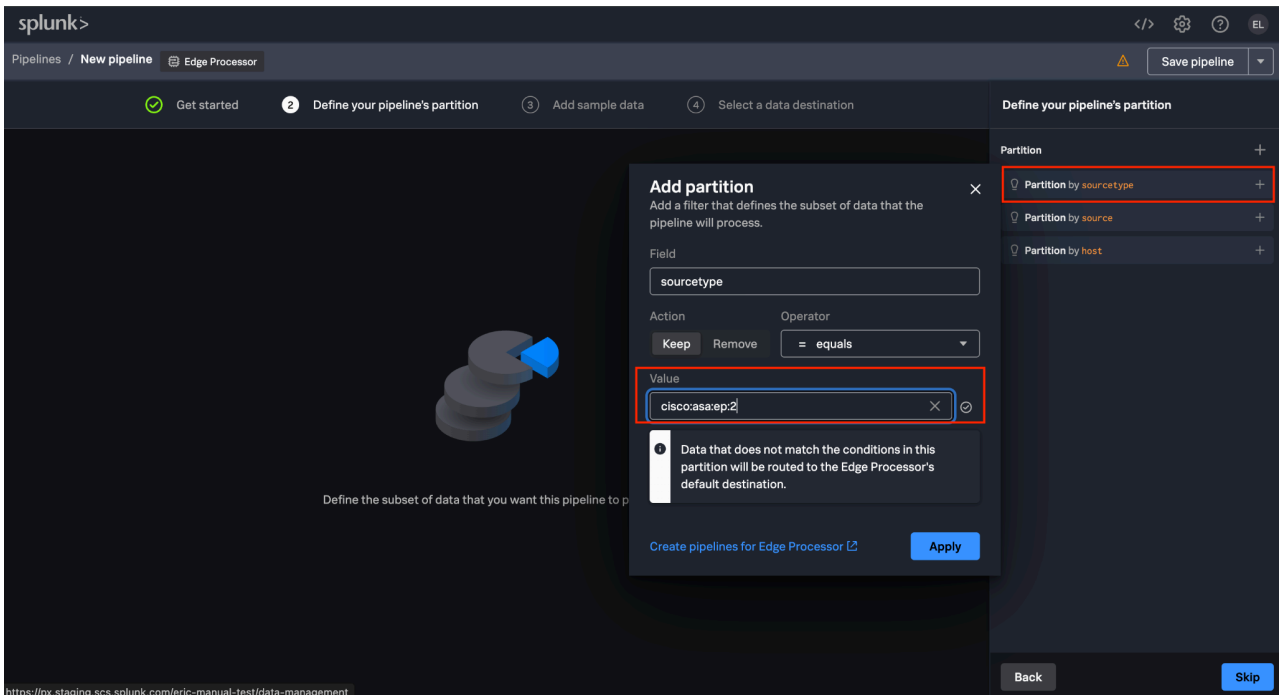


18. On the Get Started screen, you can choose from several templates, including Cisco ASA. For the purposes of this workshop, we will select the **Blank pipeline** and choose **Next**.

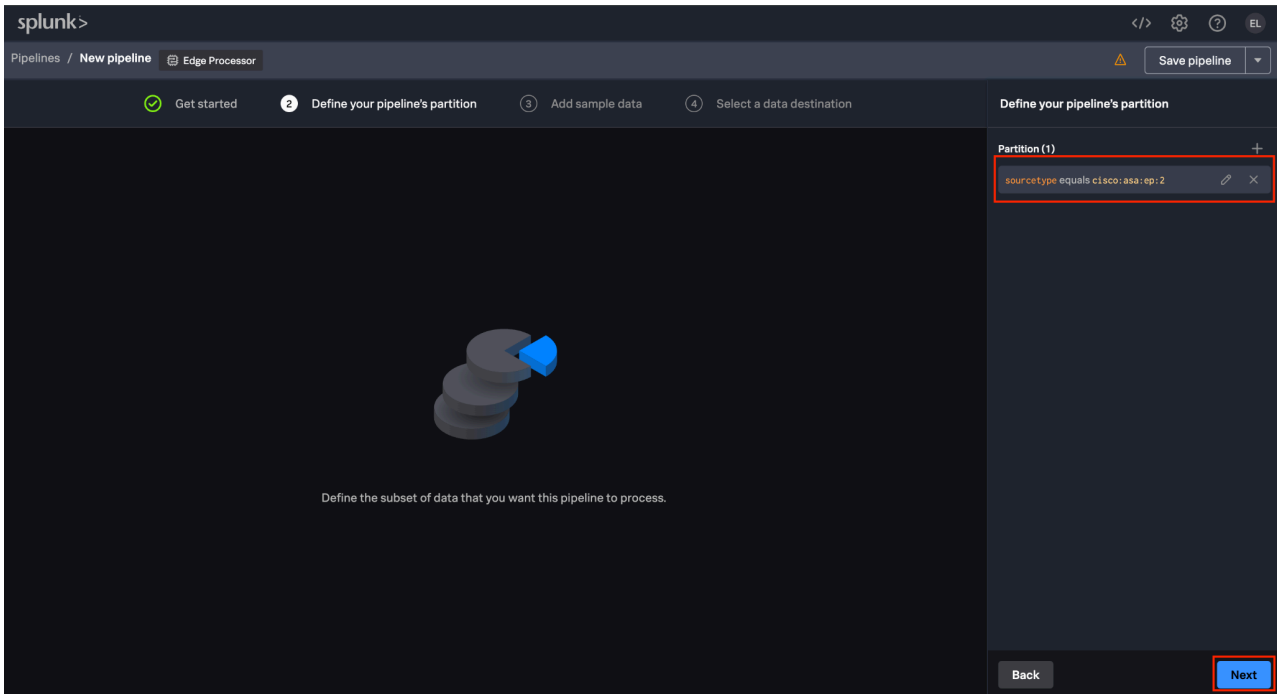


19. The pipeline's partition allows us to select which sourcetype is being ingested into Edge Processor that we want to process. Select **Partition by sourcetype** and enter the Cisco ASA sourcetype corresponding to your participant number. The format is `cisco:asa:ep:#`. Click **Apply**.

Reminder: Your sourcetype should be your participant number. For example, if your participant number is 2, your sourcetype is **cisco:asa:ep:2**

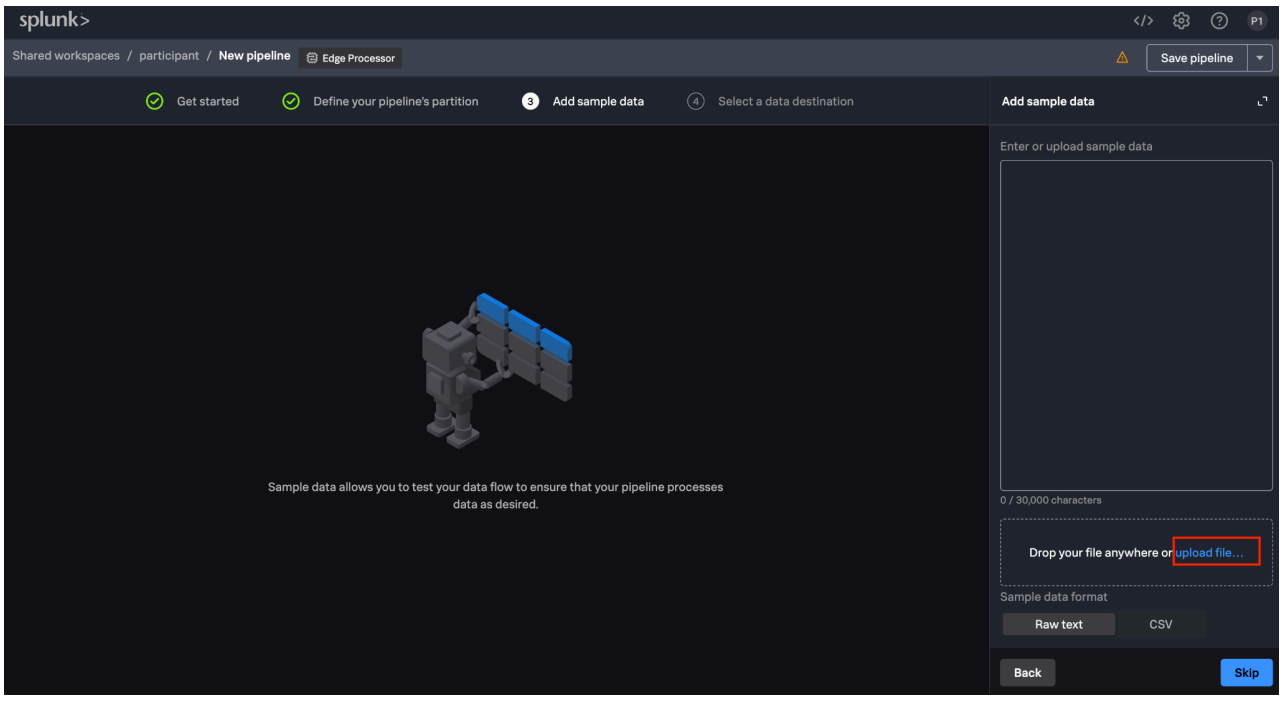


20. Note the partition information now shows the entered sourcetype. Choose **Next**.

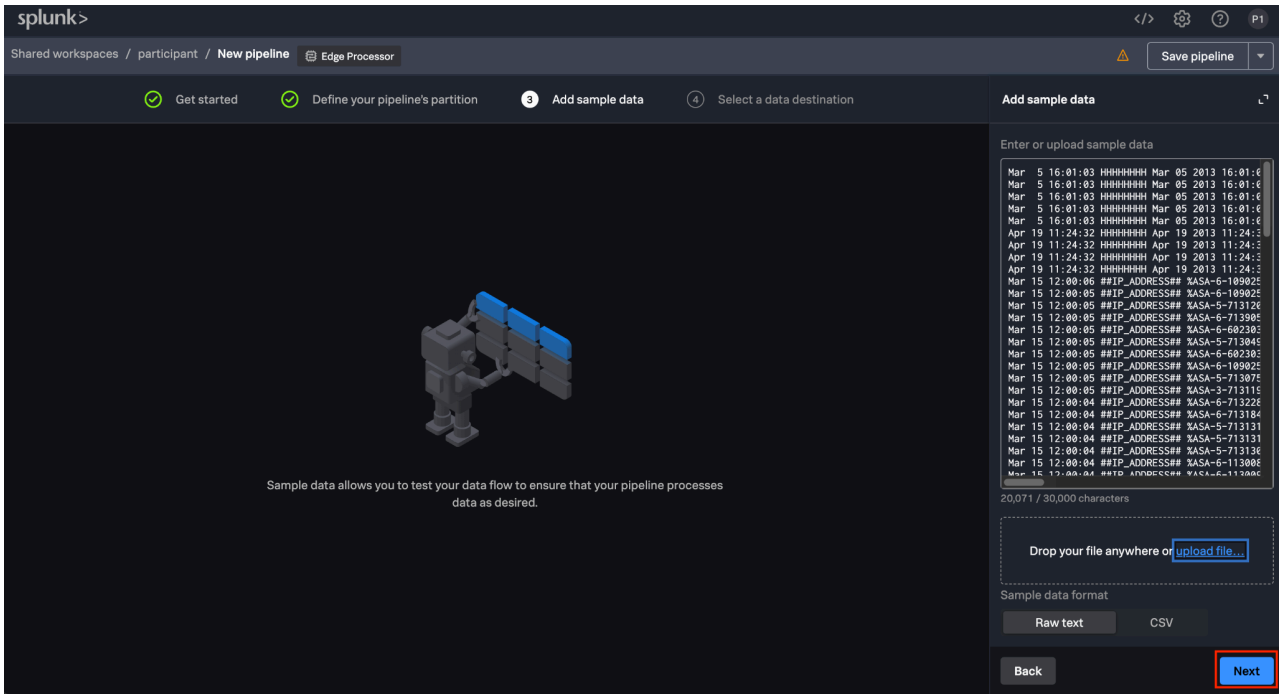


21. Choose to **Add sample data**. Download and use the data found in the following sample file:

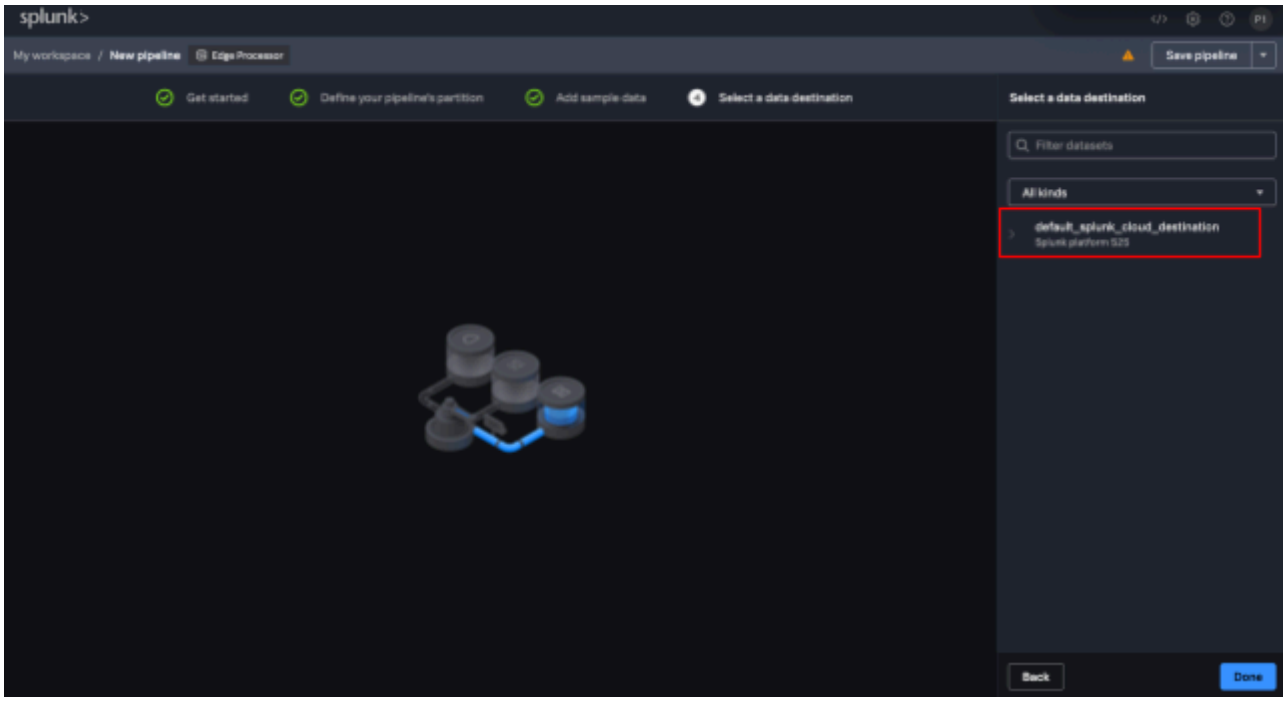
    Direct Link: [cisco_logs_small.sample]



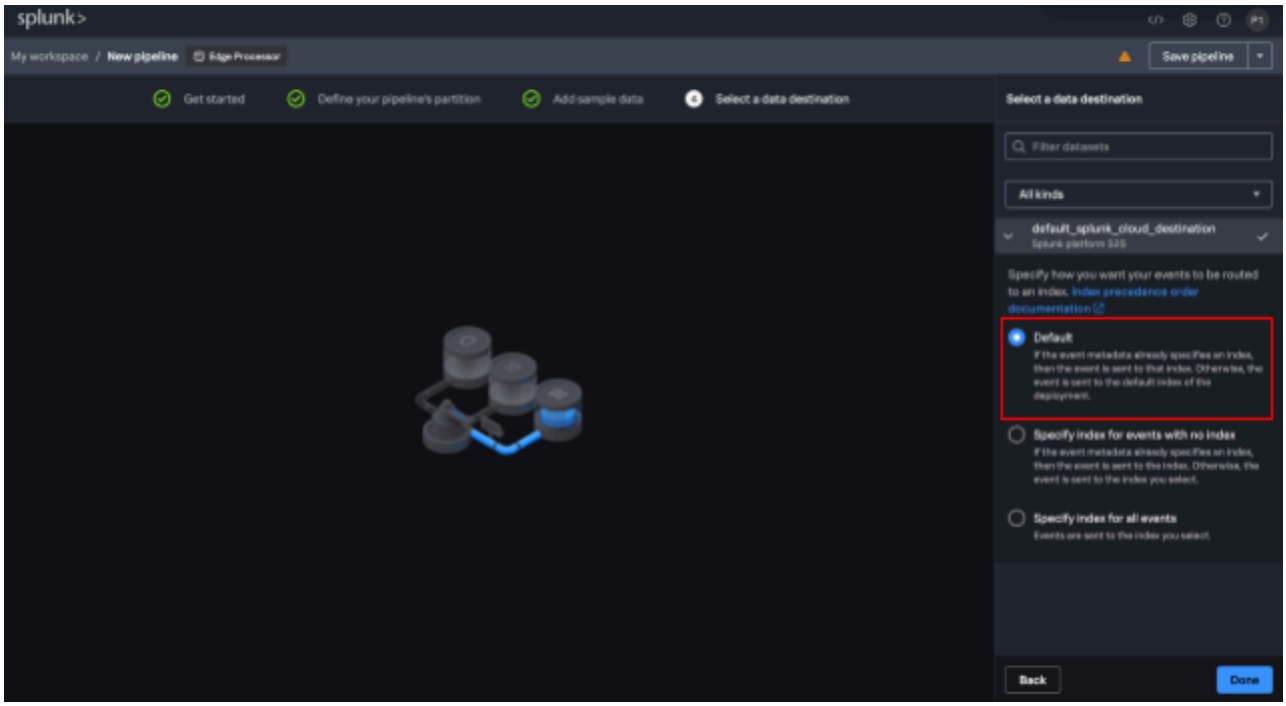
22. After uploading the sample data, press next



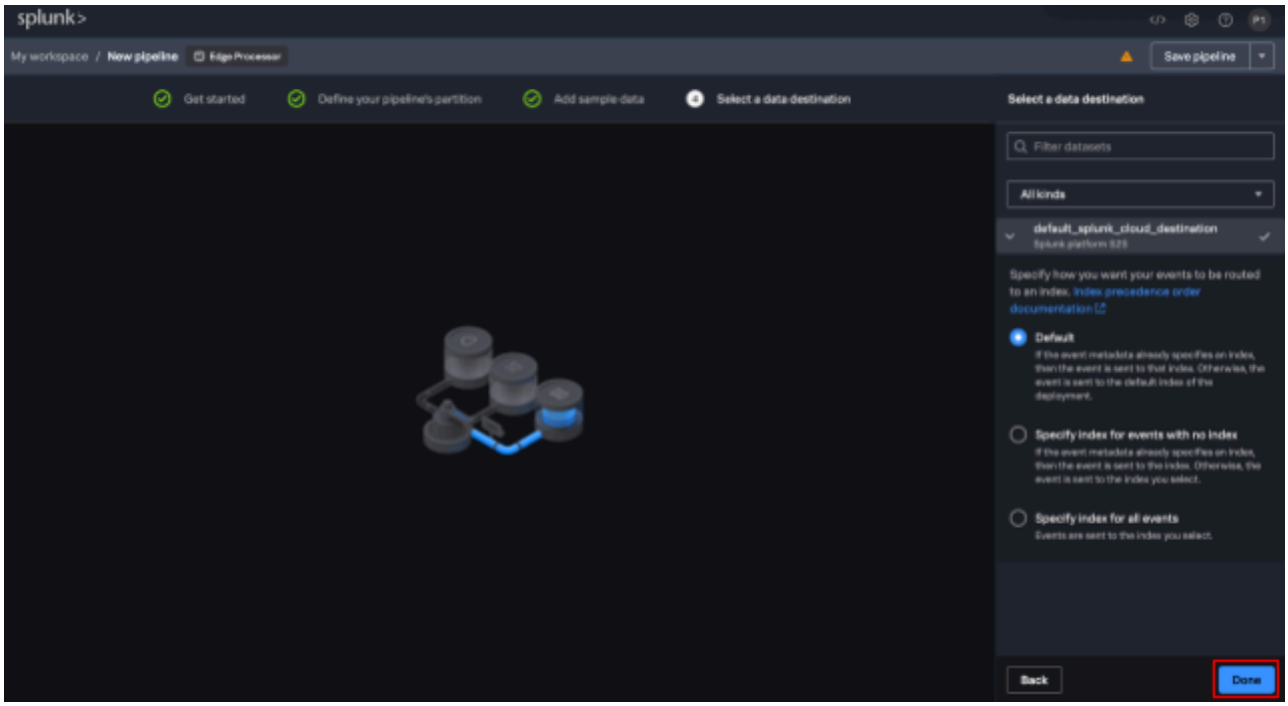
23. Select a data destination. For this exercise, let's use the pre-configured destination "**default_splunk_cloud_destination**"



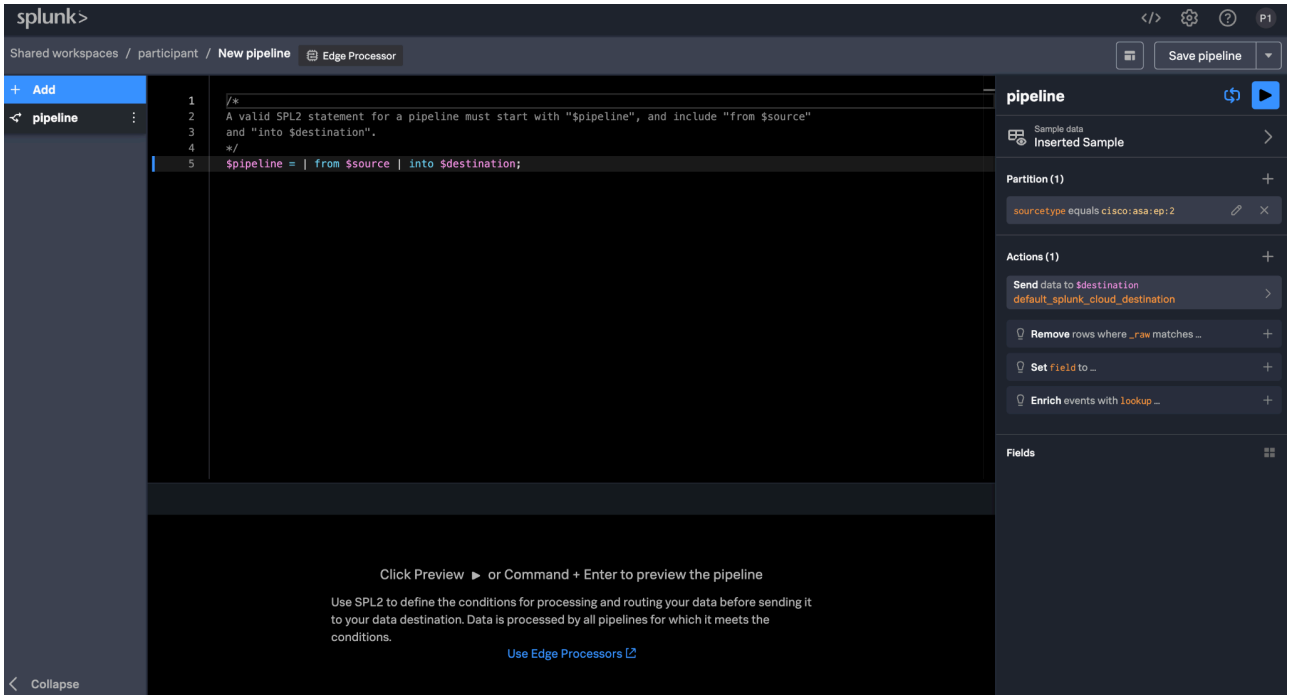
24. After selecting **default_splunk_cloud_destination,** press **Next**.



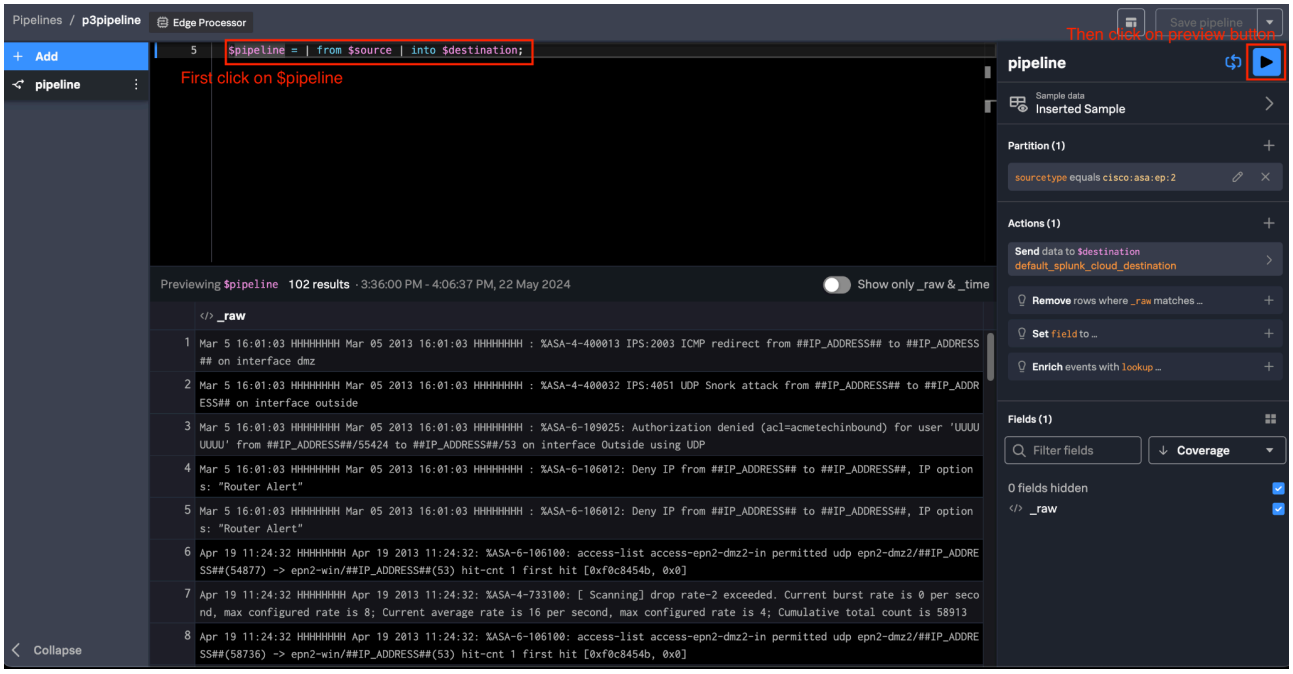
25. For this exercise, we will not select a target index. Press **Done**



26. A screen similar to the one below will appear:

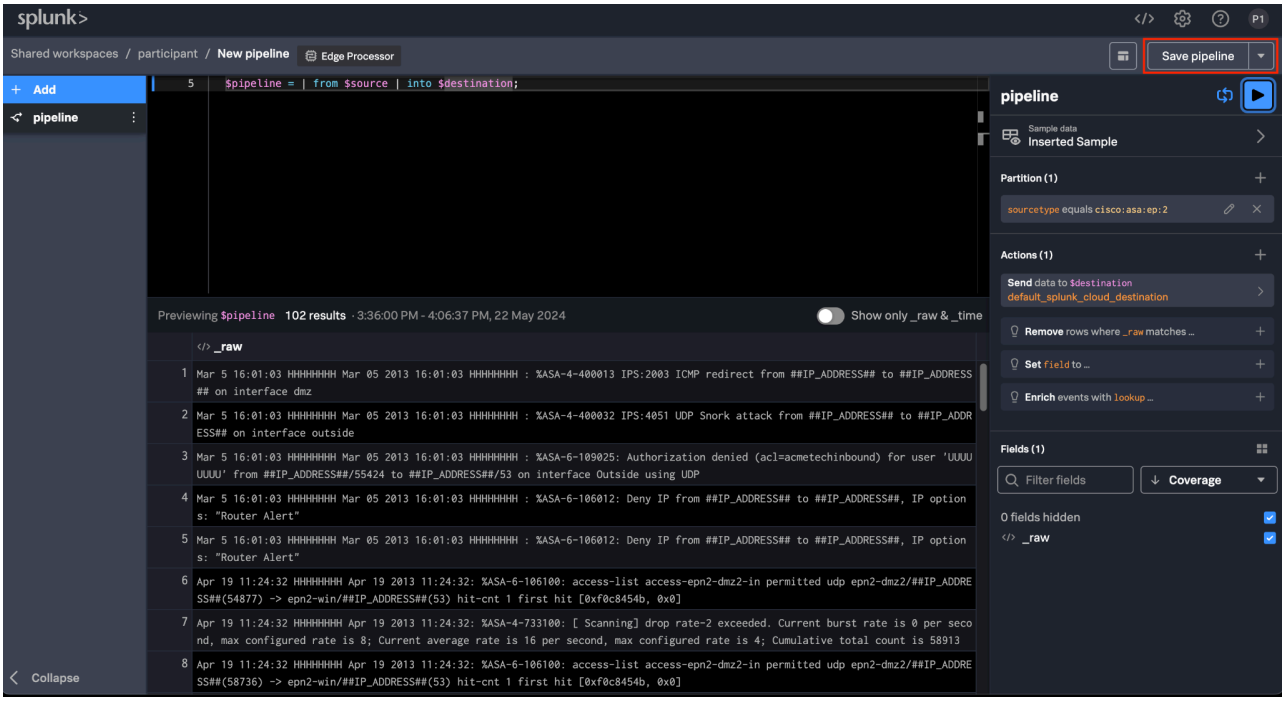


27. Test your pipeline by first clicking on the $pipeline SPL. The blue Preview pipeline button ( ) should appear in the top right corner of the screen. When you click it, you should see a preview of the sample events you uploaded earlier.

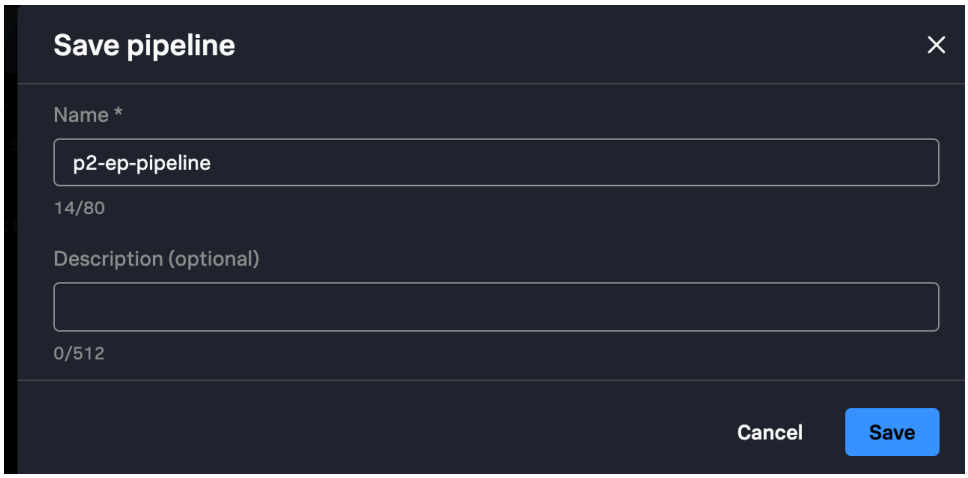


Save Your Pipeline Progress

28. We can inspect the data in the table to understand the fields and sample values that were captured. To save our progress thus far before we start editing the SPL2, choose the **Save Pipeline** button on the top right.

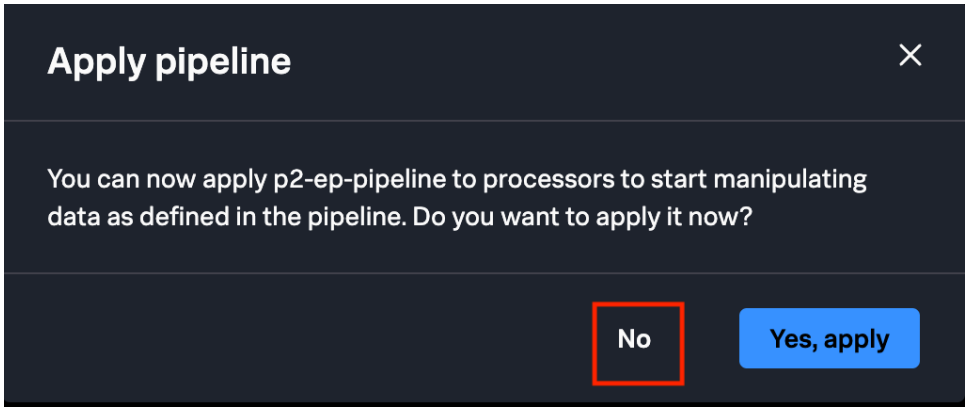


29. In the Save Pipeline modal that appears, name the pipeline `p#-ep-pipeline` **REPLACING # WITH YOUR NUMBER FROM PARTICIPANT NUMBER** and select **Save**.
    a. For example, if your **participant number is 2**, name your pipeline **p2-ep-pipeline**



30. After saving, we will be prompted in another window to Apply the pipeline. The Apply step will push out the pipeline configuration and begin transforming the data according to the SPL2 we wrote. Since we haven't written any SPL2 yet, we will select **No**.

Note: If you accidentally selected “Yes”, don’t worry. We will overwrite the pipeline when we apply our Cisco ASA SPL2 later.



Author Your Pipeline

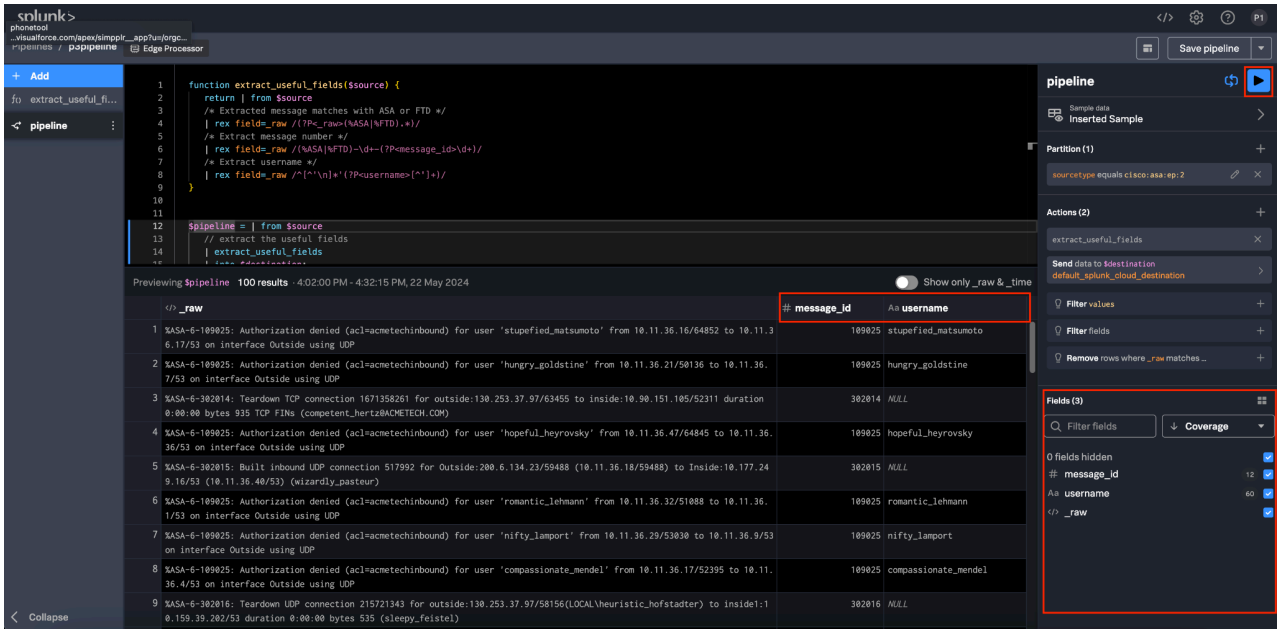
31. Now that we have our pipeline setup, we can start writing some SPL2! The first task will be to extract useful fields out of _raw. To do this, we will create an SPL2 function to extract the fields we care about. Copy/paste the following text into the SPL2 editor, replacing what was there before:

```
function extract_useful_fields($source) {
   return | from $source
   /* Extracted message matches with ASA or FTD */
   | rex field=_raw /(?P<_raw>(%ASA|%FTD).*)/
   /* Extract message number */
   | rex field=_raw /(%ASA|%FTD)-\d+-(?P<message_id>\d+)/
   /* Extract username */
   | rex field=_raw /^[^'\n]*'(?P<username>[^']+)/
}

$pipeline = | from $source
   // extract the useful fields
   | extract_useful_fields
   | into $destination;
```

The `rex` functions inside of `extract_useful_fields` will do the following:

- Ensure that all `_raw` fields start with either ASA or FTD
- Extract the message ID and store it in the message_id field
- Extract the username (if present) and store it in the username field

Pressing the preview button ( ) again will show that the fields have been extracted. Note that sometimes the field list can get quite long, so we have unchecked several fields on the bottom right to reduce the size of the preview results table. You can also rearrange the columns however you want.



Filtering

- Next up, we are going to filter certain message IDs related to security that are noisy. Add the `drop_security_noise` SPL2 function to our editor, then pipe the parsed logs into our new function.

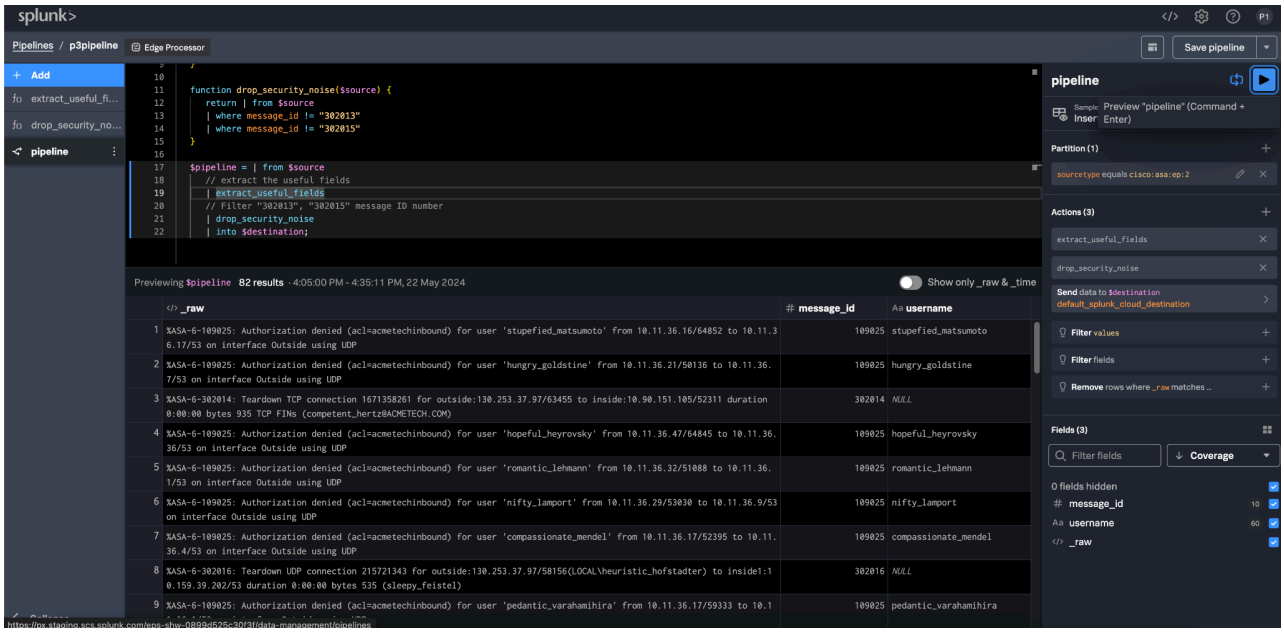
  The new function to add in is:

```
function drop_security_noise($source) {
   return | from $source
   | where message_id != "302013"
   | where message_id != "302015"
}
```

  Update your previous pipeline SPL to:

```
$pipeline = | from $source
   // extract the useful fields
   | extract_useful_fields
   // Filter "302013", "302015" message ID number
   | drop_security_noise
   | into $destination;
```

- Preview again and notice the 302013 and 302015 message IDs have been filtered.



Masking

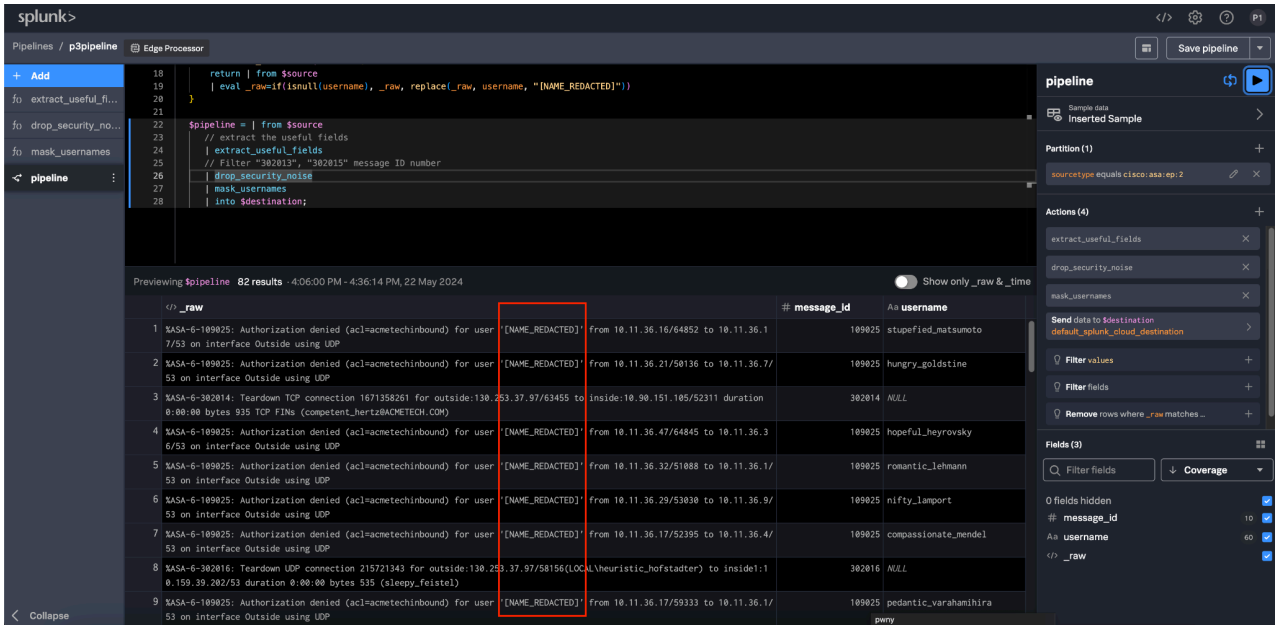
- When we send this data to the index, we've decided to mask the usernames to maintain compliance. Here's the mask_usernames function to replace usernames found in _raw:

```
function mask_usernames($source) {
    return | from $source
    | eval _raw=if(isnull(username), _raw, replace(_raw, username, "[NAME_REDACTED]"))
}
```

Don't forget to invoke `mask_usernames` in your pipeline:

```
$pipeline = | from $source
   // extract the useful fields
   | extract_useful_fields
   // Filter "302013", "302015" message ID number
   | drop_security_noise
   // Mask usernames to protect PII
   | mask_usernames
   | into $destination;
```

When previewing your results, you will see the username has been redacted in _raw. Note that the username is still present in the username field.



Enrichment Via Lookup

- Next, we can enrich our log events by looking up the message ID with some explanations via a pre-uploaded lookup file.

  For lookup import, everyone would have a different path. Your import path is **/envs.splunk.'[insert your demo tenant]'.lookups**

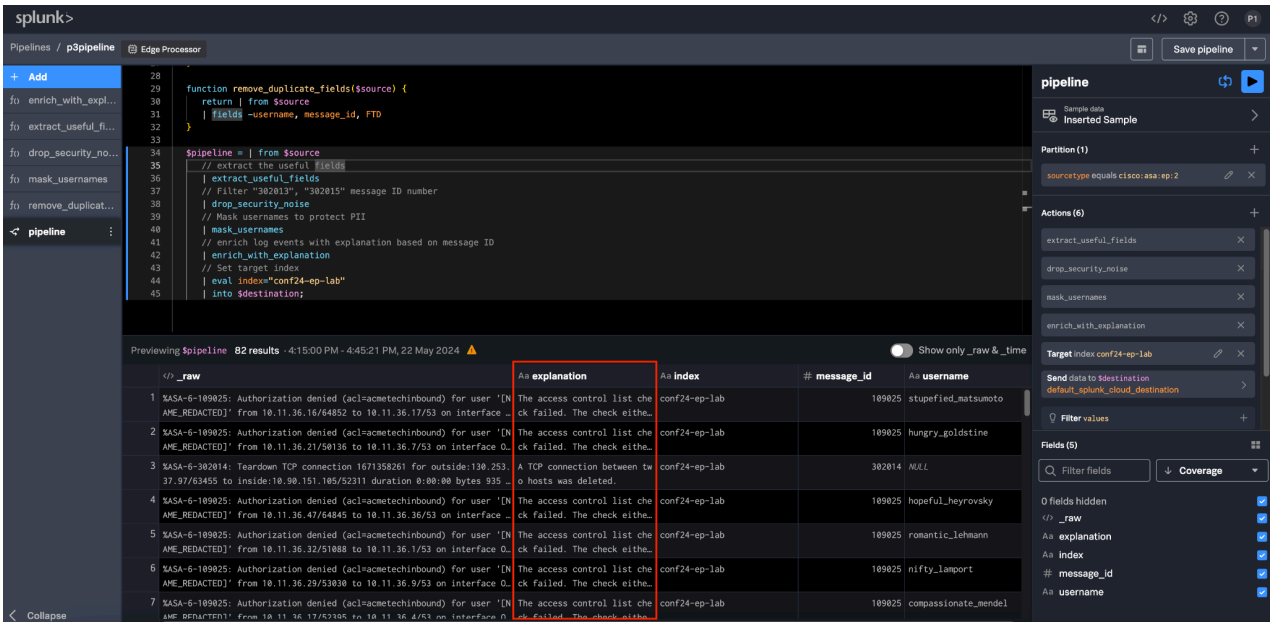
  Tenant information can be found at your URL. For example, if your URL starts with https://px.staging.scs.splunk.com/eps-shw-0899d525c30f3f/, your tenant is **eps-shw-0899d525c30f3f**. Then your import path would be **/envs.splunk.'eps-shw-0899d525c30f3f'.lookups**

```
// import statement should be on the top of the file
import 'cisco_msg_id.csv' from /envs.splunk.'eps-shw-0899d525c30f3f'.lookups

function enrich_with_explanation($source) {
   return | from $source |
   lookup 'cisco_msg_id.csv' message_id AS message_id OUTPUT explanation AS explanation
}
```

  Then, we can apply this function in our pipeline:

```
$pipeline = | from $source
  // extract the useful fields
  | extract_useful_fields
  // Filter "302013", "302015" message ID number
  | drop_security_noise
  // Mask usernames to protect PII
  | mask_usernames
  // enrich log events with explanations based on message ID
  | enrich_with_explanation
  | into $destination;
```

You will now see a new column with an explanation for each log event:



Drop Fields & Set Index

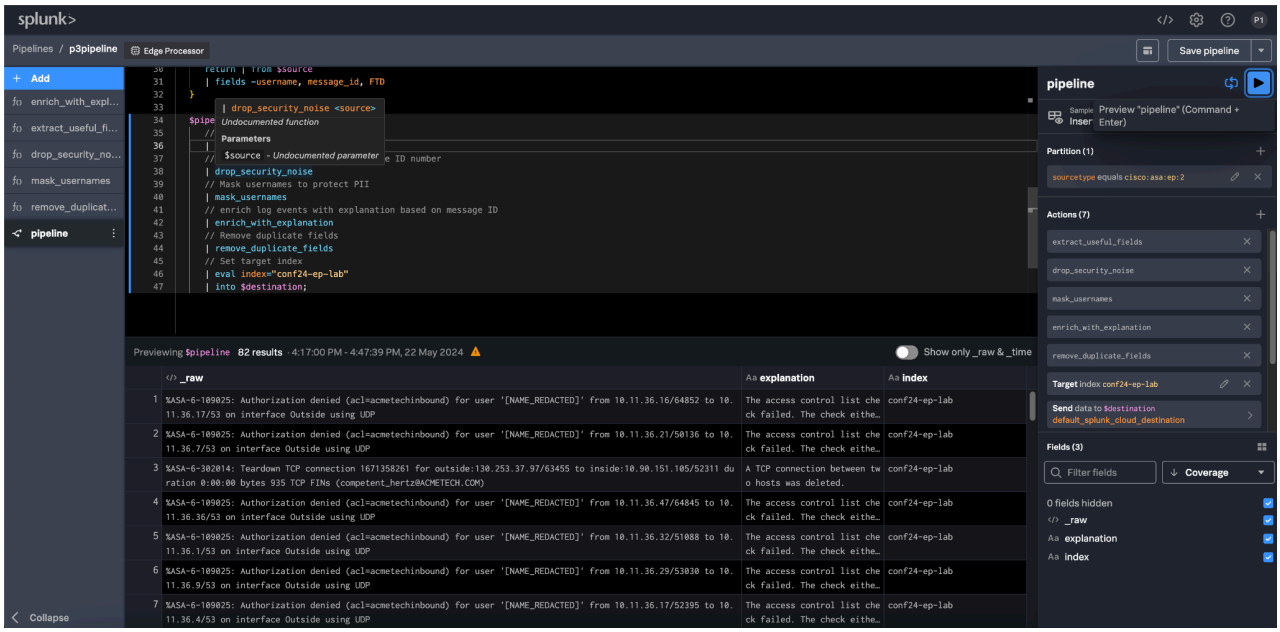
- Finally, we can remove the parsed fields as they are no longer needed, and we don't want them sent to the indexer. Create the remove_duplicate_fields function:

```
function remove_duplicate_fields($source) {
   return | from $source
   | fields -username, message_id, FTD
}
```

Add the `remove_duplicate_fields` function to your pipeline definition and set the index to `dmx-ep-lab`.

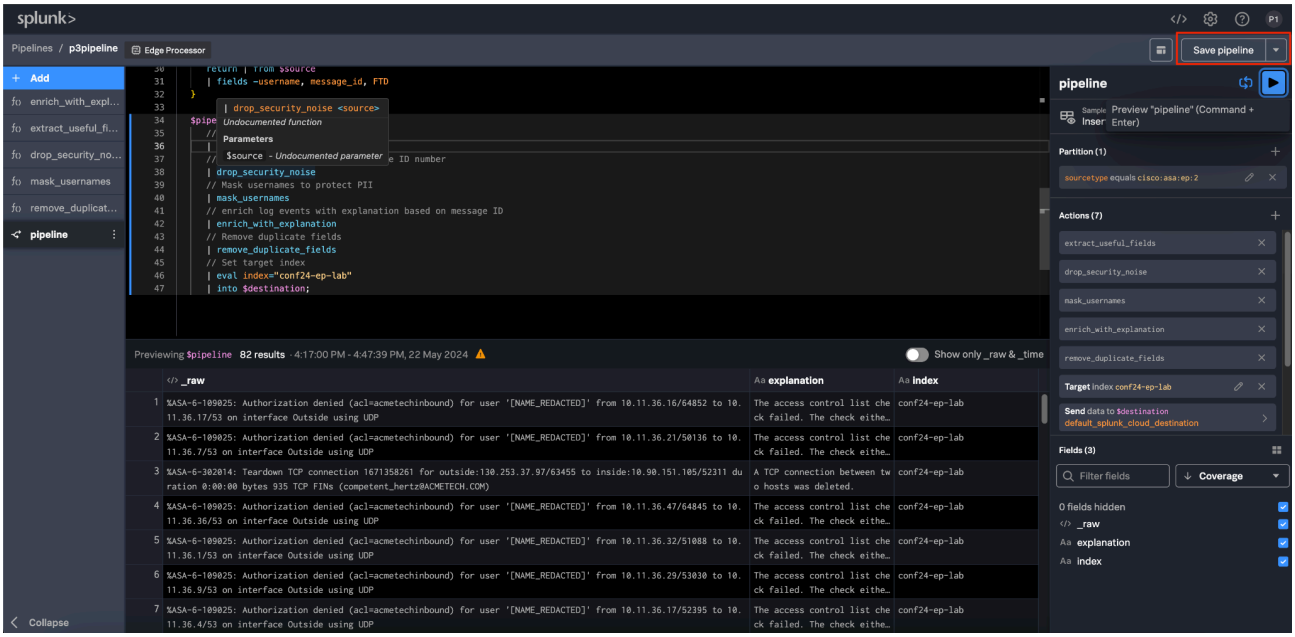
```
$pipeline = | from $source
  // extract the useful fields
  | extract_useful_fields
  // Filter "302013", "302015" message ID number
  | drop_security_noise
  // Mask usernames to protect PII
  | mask_usernames
  // enrich log events with explanations based on message ID
  | enrich_with_explanation
  // Remove duplicate fields
  | remove_duplicate_fields
  // Set target index
  | eval index="dmx-ep-lab"
  | into $destination;
```

Preview now shows the results without these fields along with the index field set:

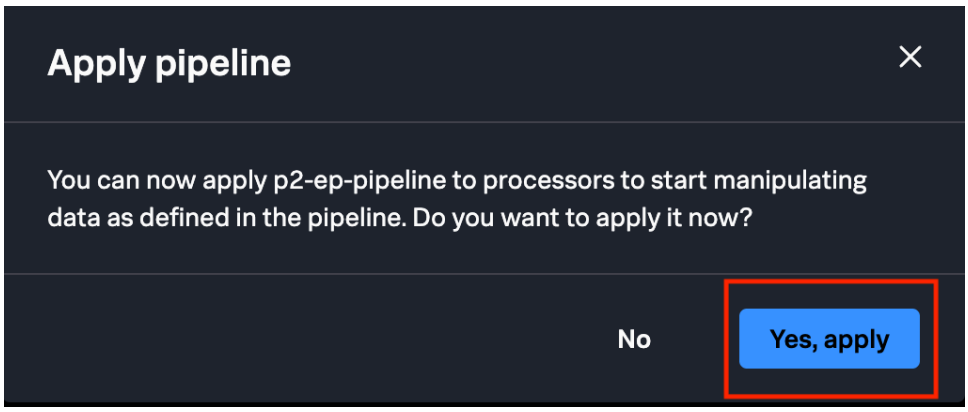


Apply the Pipeline to your Edge Processor

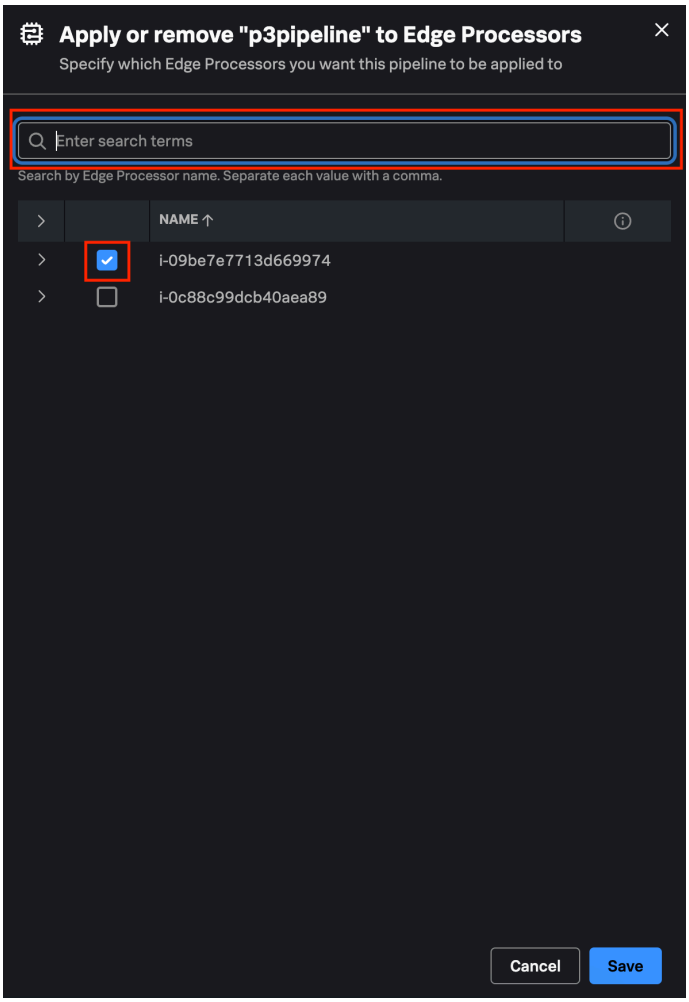
- Being satisfied with the results of our pipeline, we will now apply the pipeline. As we did previously, select the **Save pipeline** button on the top right.



If prompted, select **Yes, apply** to apply the pipeline. If you are not prompted, the pipeline will be automatically applied on save.

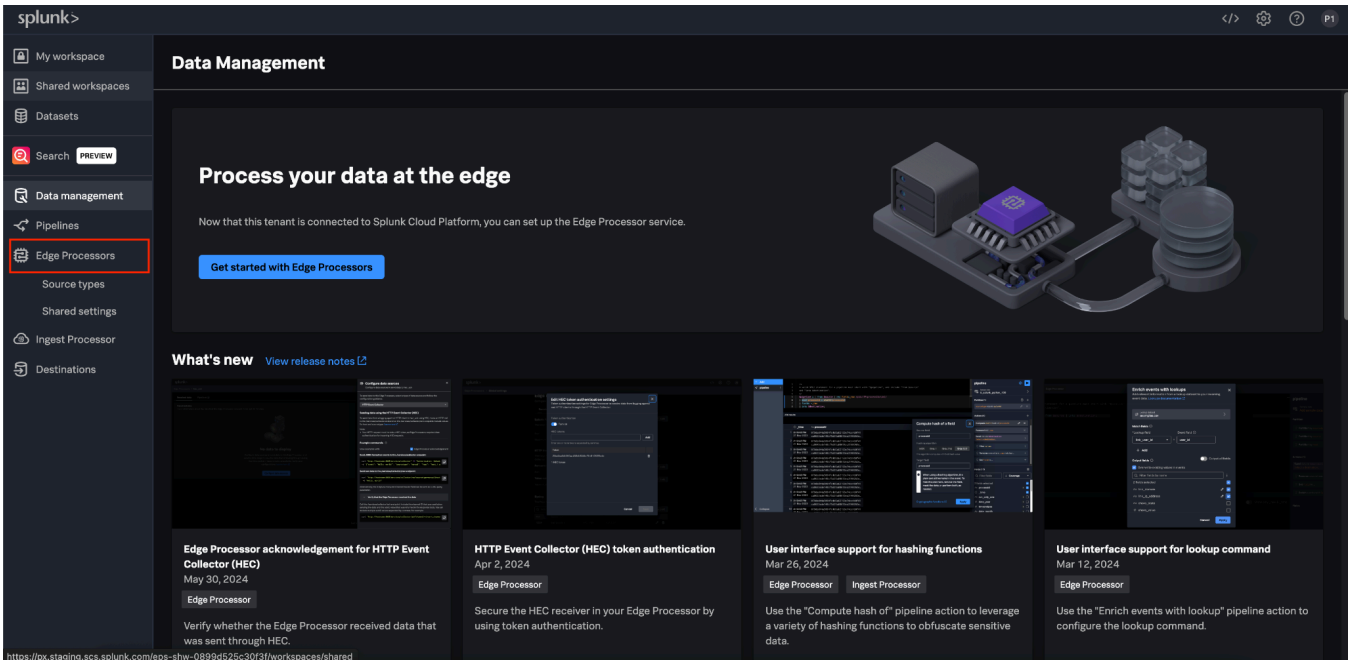


Search and select the edge processor instance that is assigned to you. Then press **Save**

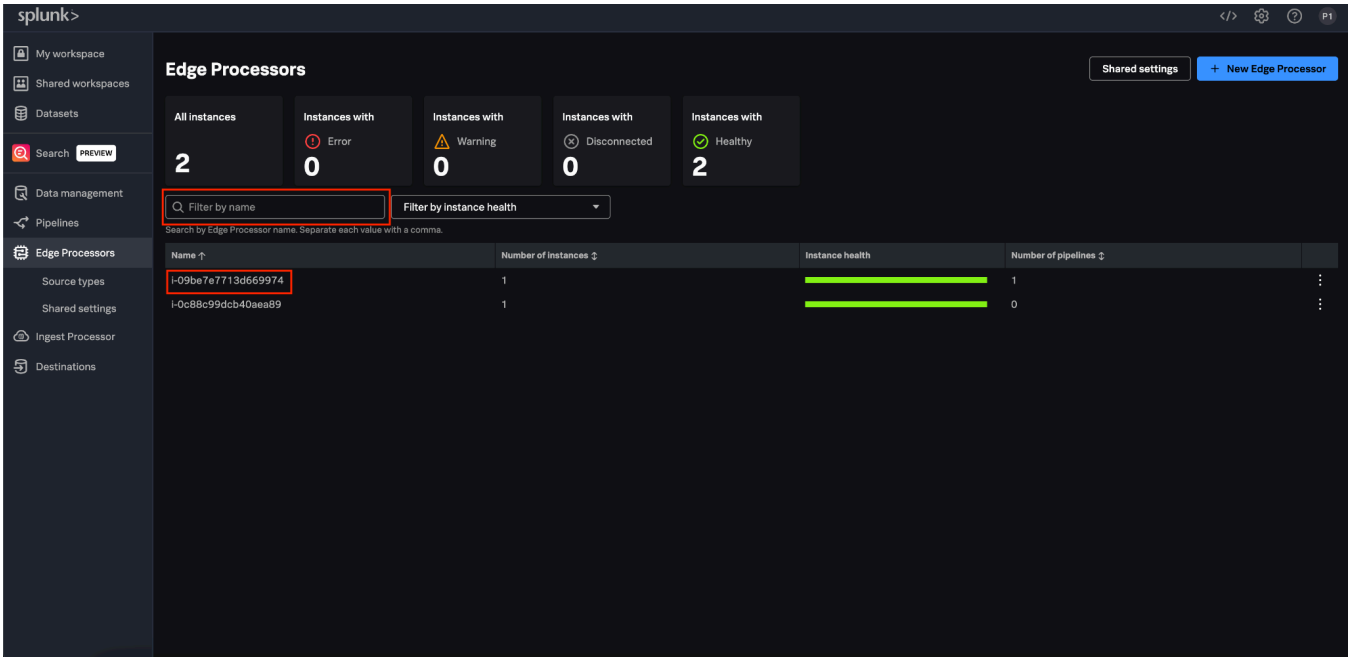


Verify Edge Processor is Receiving Data

- After applying your pipeline successfully, click the Splunk logo on the top left to navigate back to the home page. Then click **Edge Processors** on the left panel

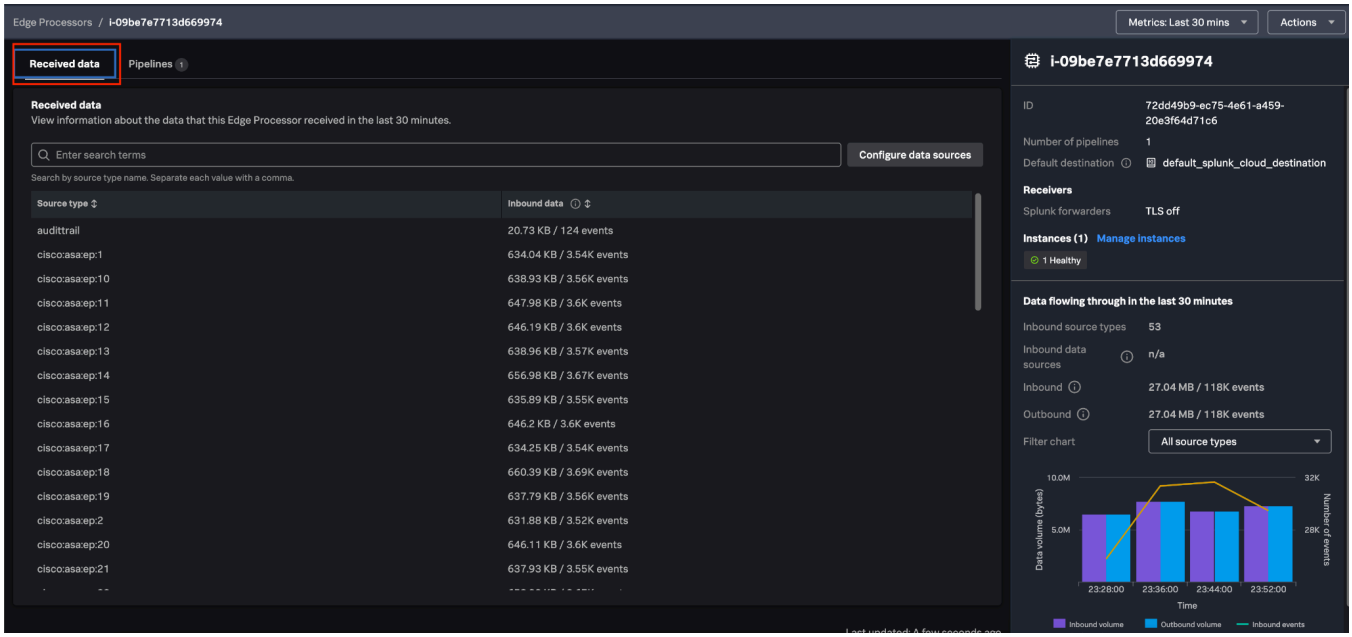


- Search for the instance that is assigned to you and double-click on it

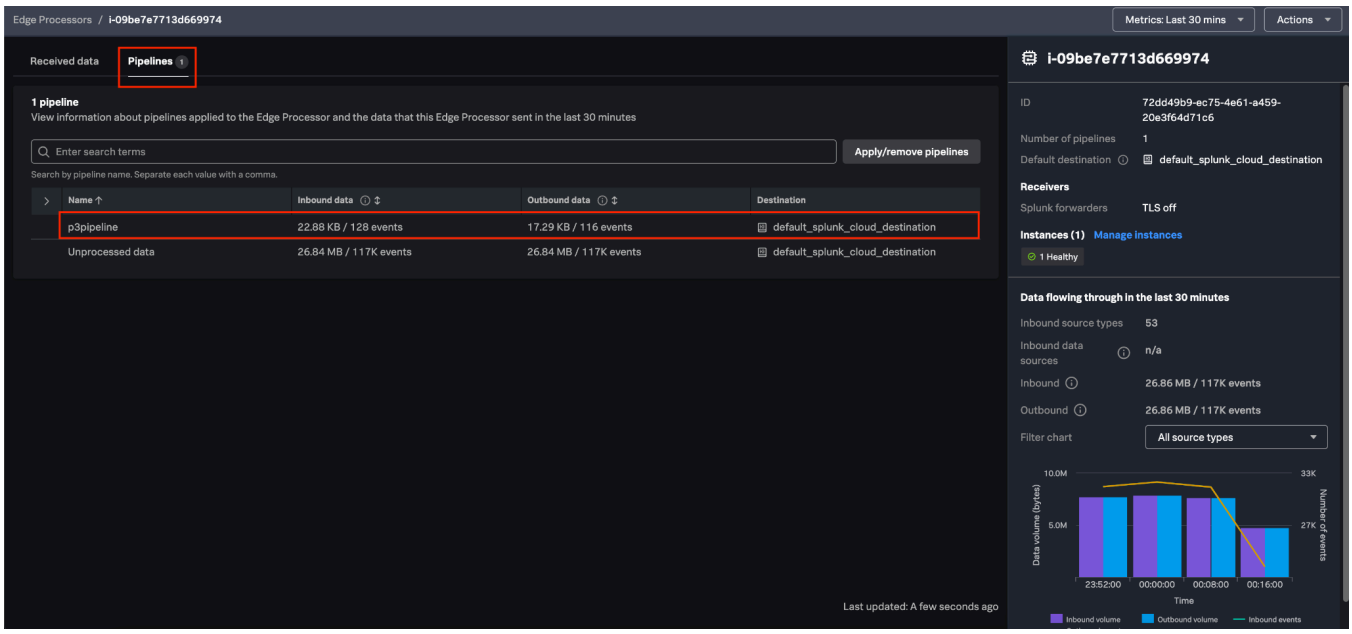


- Under the **Data sources** pane on the left side of the page, click on **Received data** and verify that your Edge Processor is receiving inbound **cisco:asa:ep:#** sourcetype data. You will notice multiple sourcetypes with a different suffix number. This is to allow each participant to experiment with their data without interfering with other participants.

  Reminder: if your **participant number** is **2**, please ensure your pipeline is set up to process data with sourcetype **cisco:asa:ep:2** in step 9.



- In the pipelines panel, you should be able to see the pipeline you just applied to your Edge Processor.



Check Your Data in Splunk Cloud

- 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:

  Reminder: Your sourcetype should be your participant number. For example, if your participant number is 2, your sourcetype is **cisco:asa:ep:2**

  ```
  index=dmx-ep-lab sourcetype=cisco:asa:ep:#
  ```

Lab Exercise 3: Ingest Processor

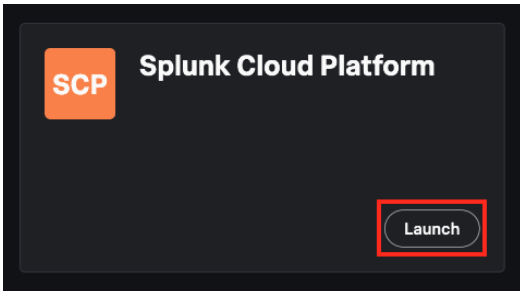
Description

In this lab, you will filter, mask, and route Cisco ASA logs using Ingest Processor. The transformation pipelines you write will execute immediately prior to indexing. You will implement the Base Use Case described at the beginning of this guide. You will also implement a basic Logs To Metrics conversion and send metrics to Splunk Observability (https://www.splunk.com/en_us/products/observability.html) (aka O11y).

Access Your Ingest Processor Console

1. Log into the Splunk Cloud Ingest Processor UI. Refer to Lab Connection Info to find your login URL and credentials.

2. On the welcome page, click on **Launch** under **Splunk Cloud Platform**.

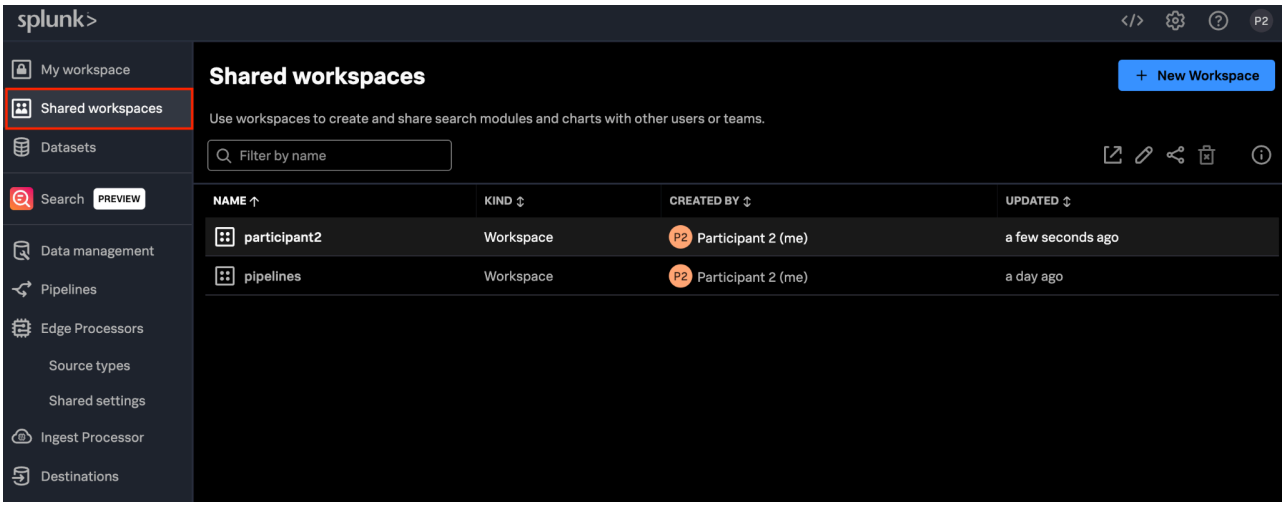


You will now be taken to the Ingest Processor management environment.

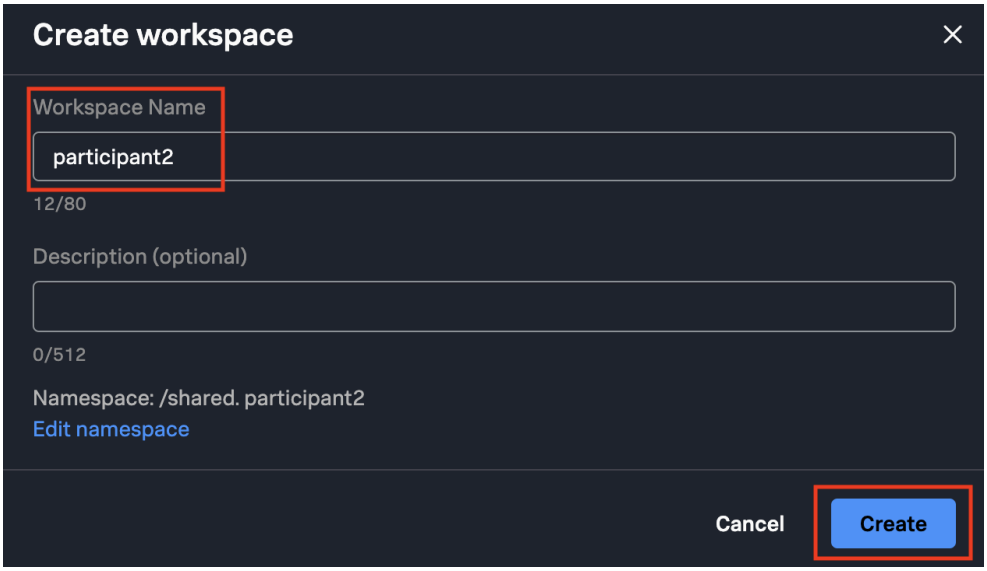
Create a Workspace

Note: You can skip this step if you have already created this workspace in the past.

3. Enter the shared workspace by clicking on the Shared Workspaces button on the left side.

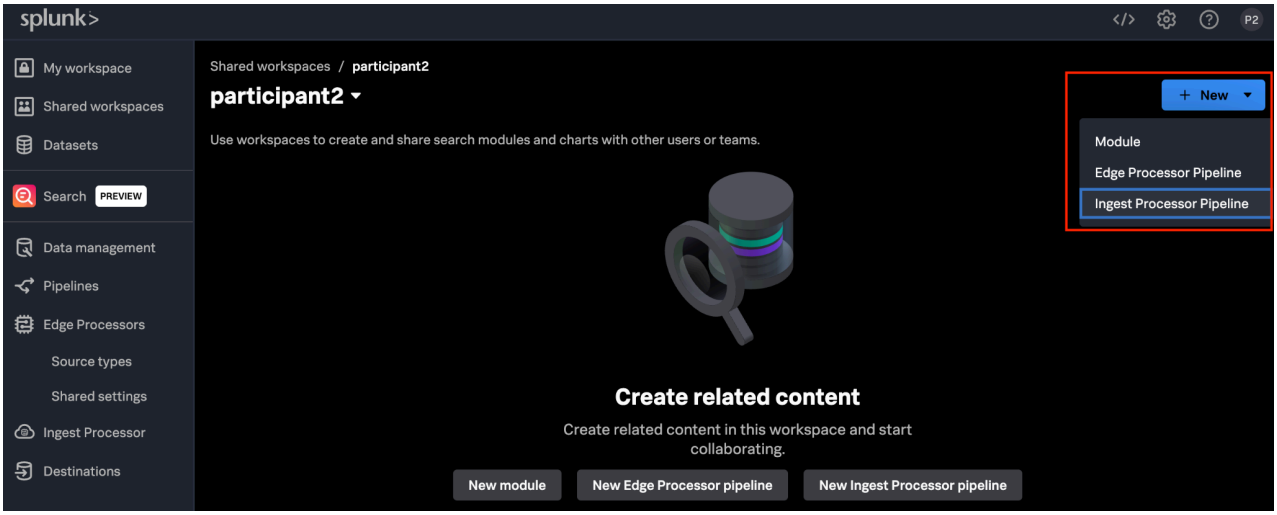


4. Create a new workspace for your participant. Name the workspace **participant# (REPLACE # WITH YOUR PARTICIPANT NUMBER)**. Select the **Create** button.

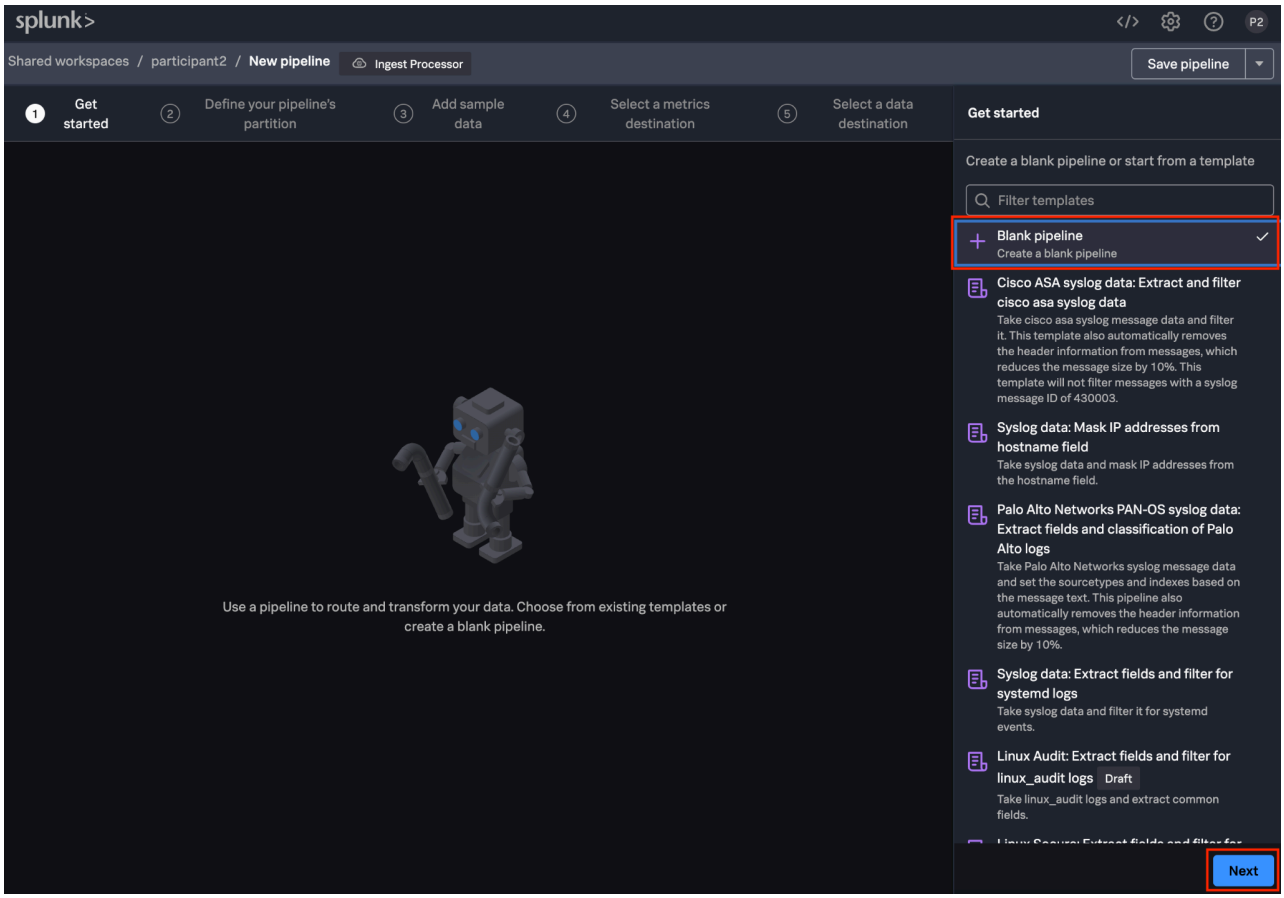


Create a Pipeline

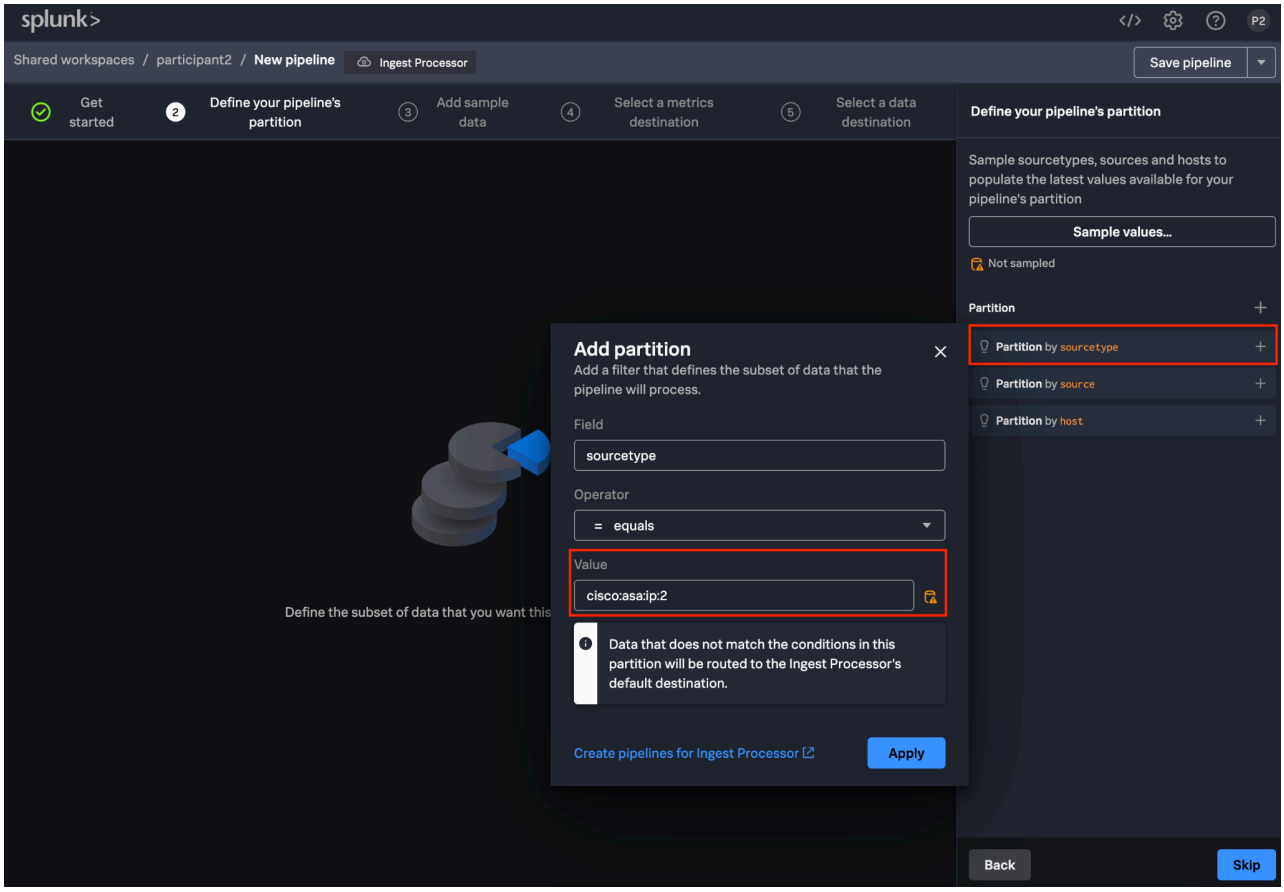
5. In your participant workspace, create a new IP pipeline by selecting **New** → **Ingest Processor Pipeline** in the upper right.



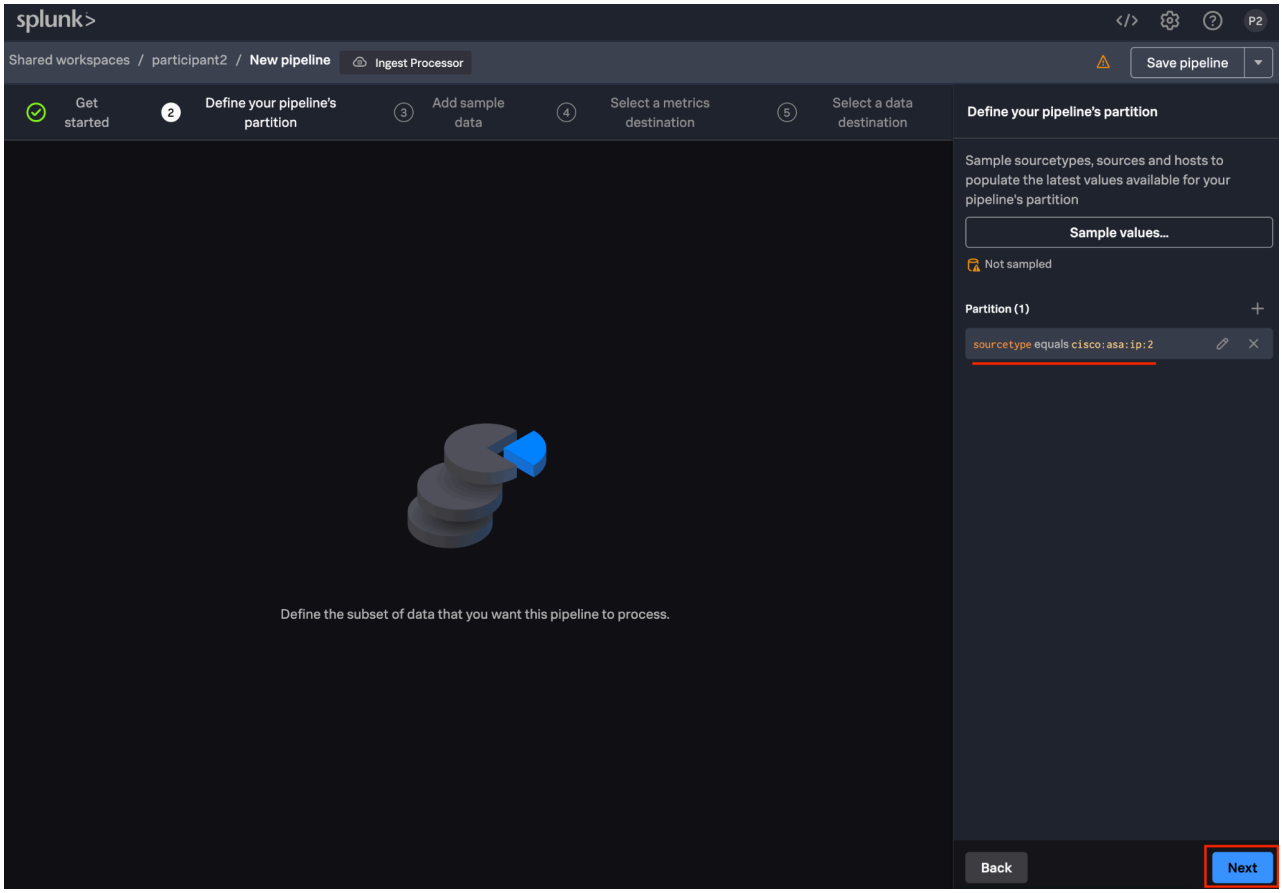
6. On the Get Started screen, you can select from several templates, including Cisco ASA. For the purposes of this workshop, we will select the **Blank Pipeline** and choose **Next**.



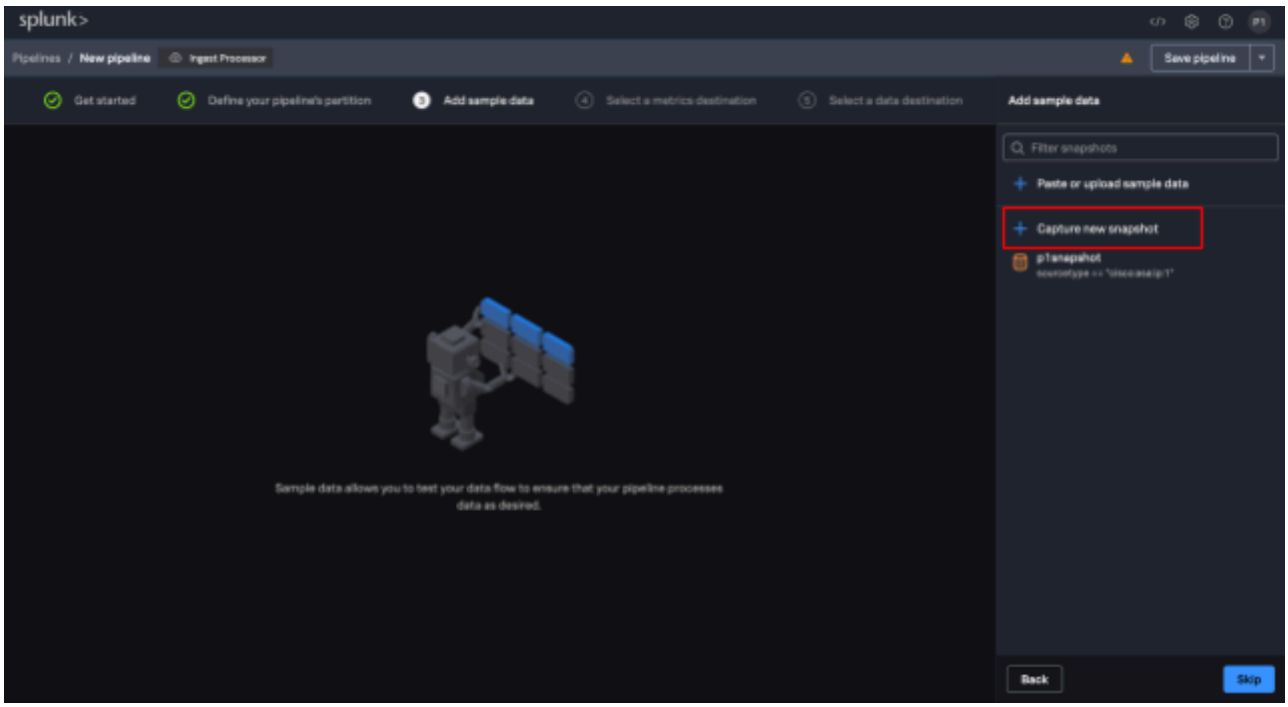
7. The pipeline's partition allows us to select which sourcetype is being ingested into Splunk Cloud that we want to process prior to indexing. Select **Partition by sourcetype** and enter the Cisco ASA sourcetype corresponding to your participant number. The format is `cisco:asa:ip:#`. **REPLACE # WITH YOUR PARTICIPANT NUMBER.** Click **Apply**.



8. Note the partition information now shows the entered sourcetype. Choose **Next**.

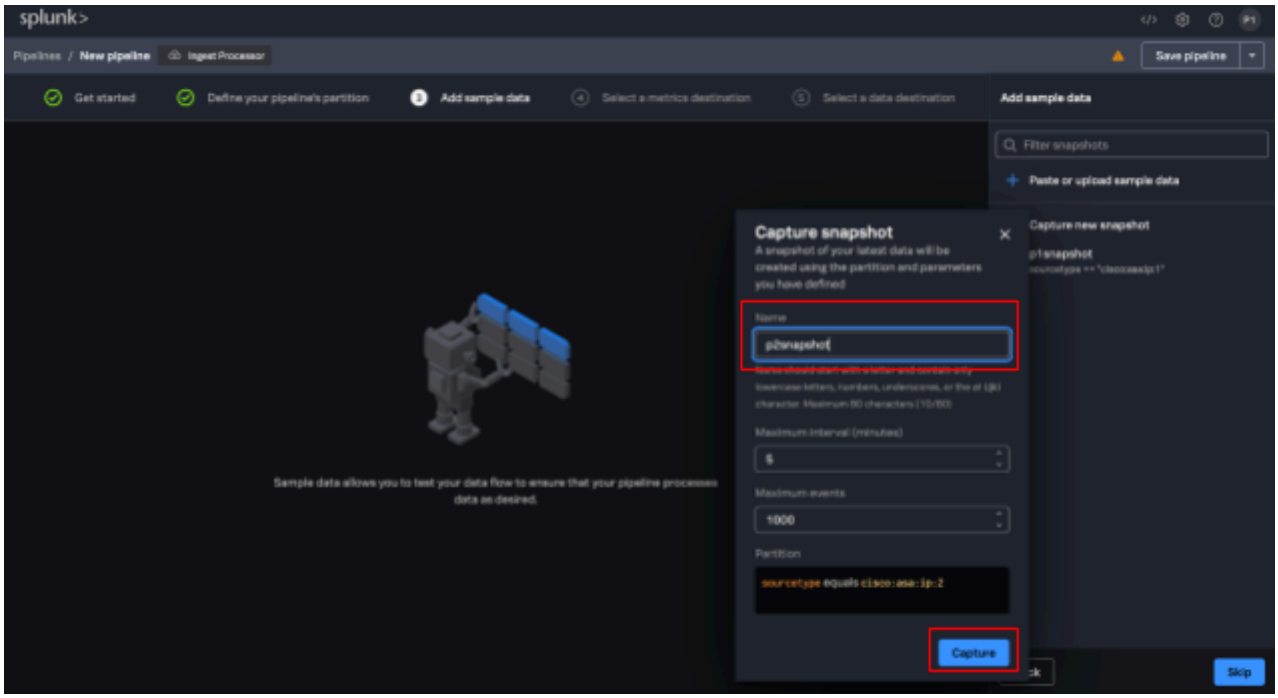


9. Choose to add sample data **From snapshot**.

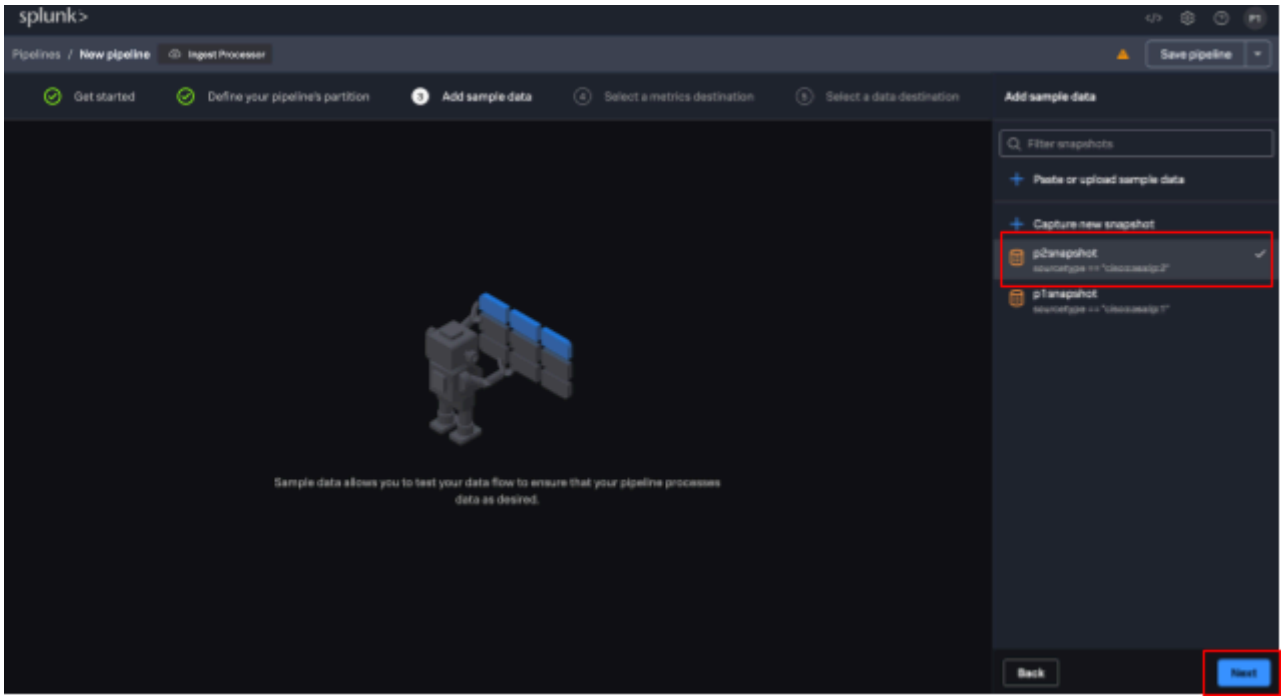


Create a Preview Snapshot

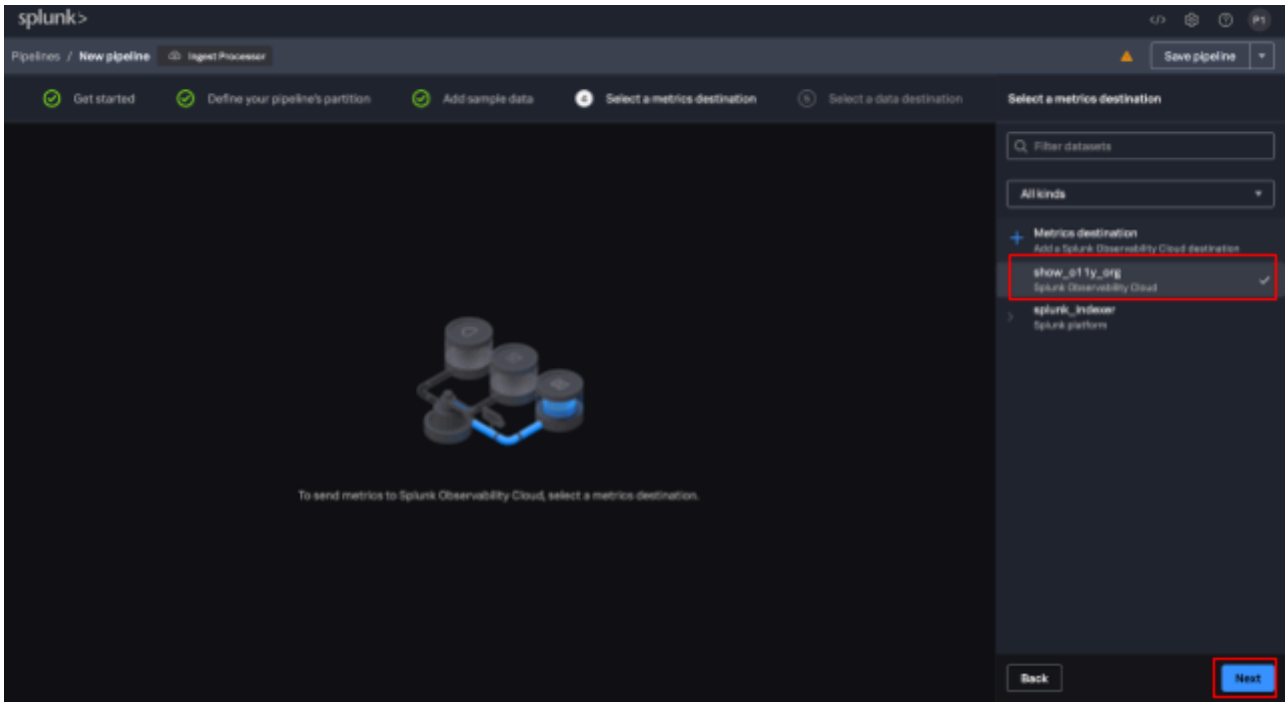
10. We will create a new snapshot from the data that is being ingested at this moment. Choose **Create new snapshot**, enter the snapshot name (such as `p#snapshot` **REPLACING # WITH YOUR PARTICIPANT NUMBER**), and then initiate the snapshot process by selecting **Capture**. This will begin capturing up to 1000 events over the next 5 minutes. Note that our pipeline sourcetype is automatically populated.



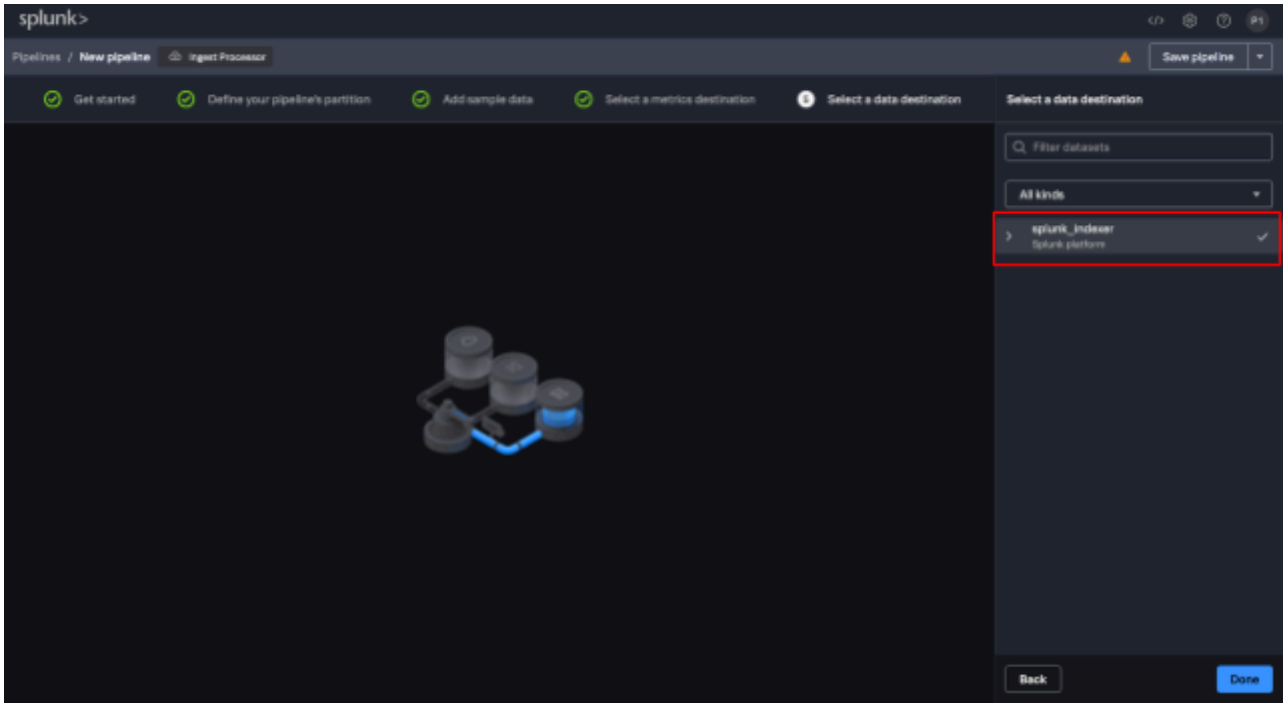
11. Select the newly created snapshot and choose **Next**.



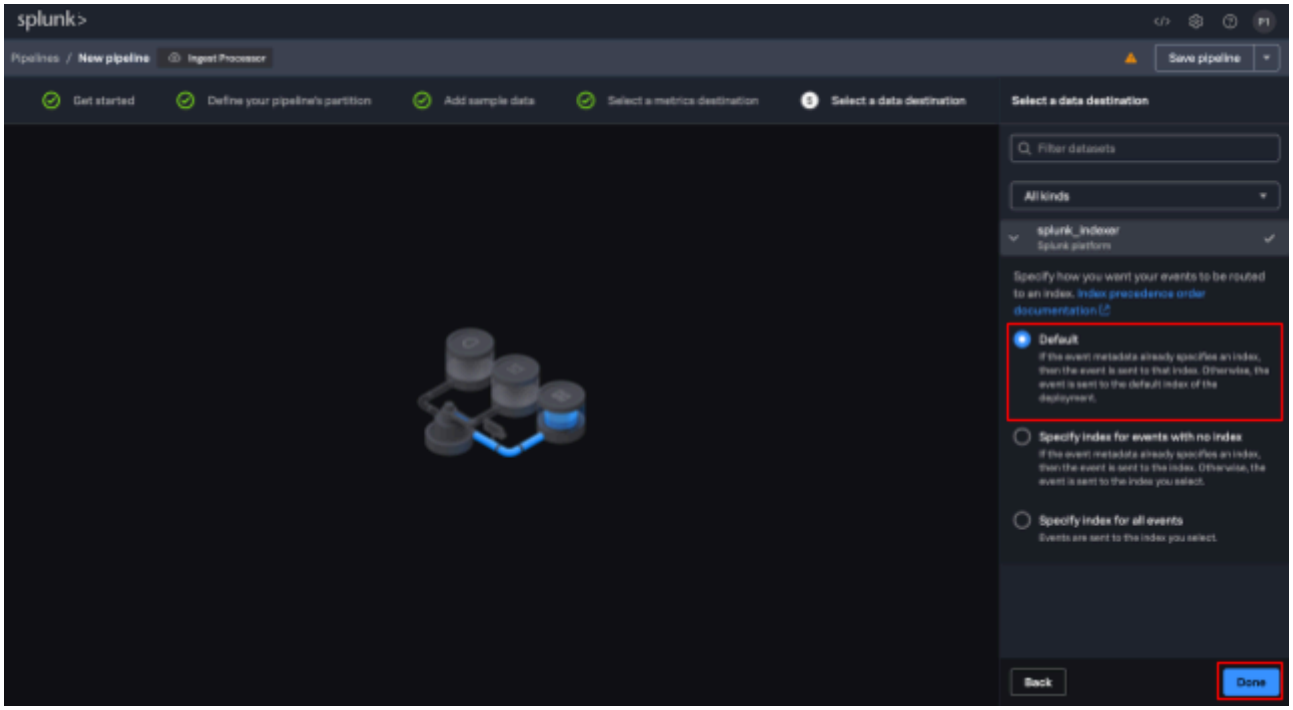
12. In this step, you are prompted for the default metrics destination. Choose the Splunk Show Splunk Observability destination that was automatically created for you, **show_o11y_org**. Then click **Next**.



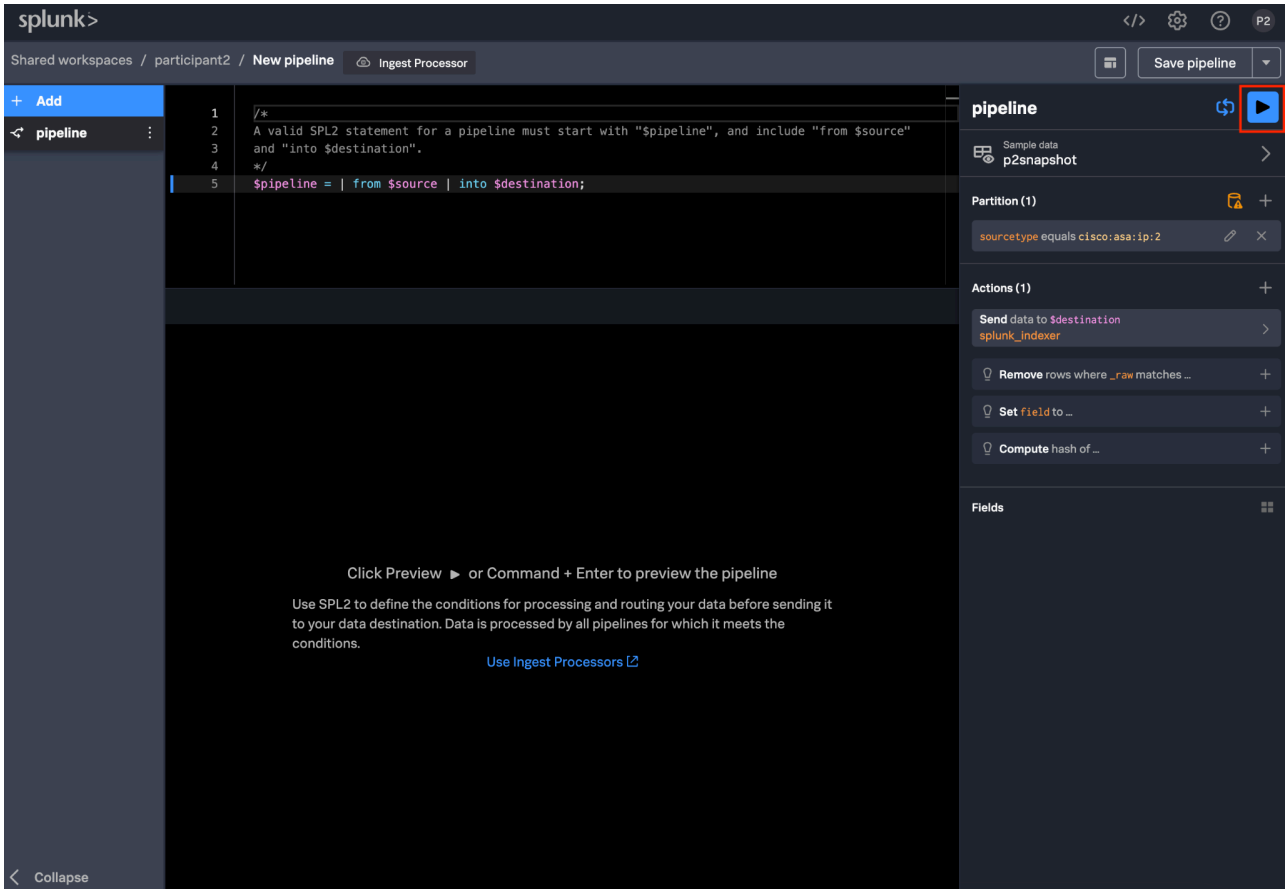
13. In this step, you are prompted to go to the default logs destination. Choose the Splunk indexer destination that was automatically created for you called **splunk_indexer**. Then click **Next**.



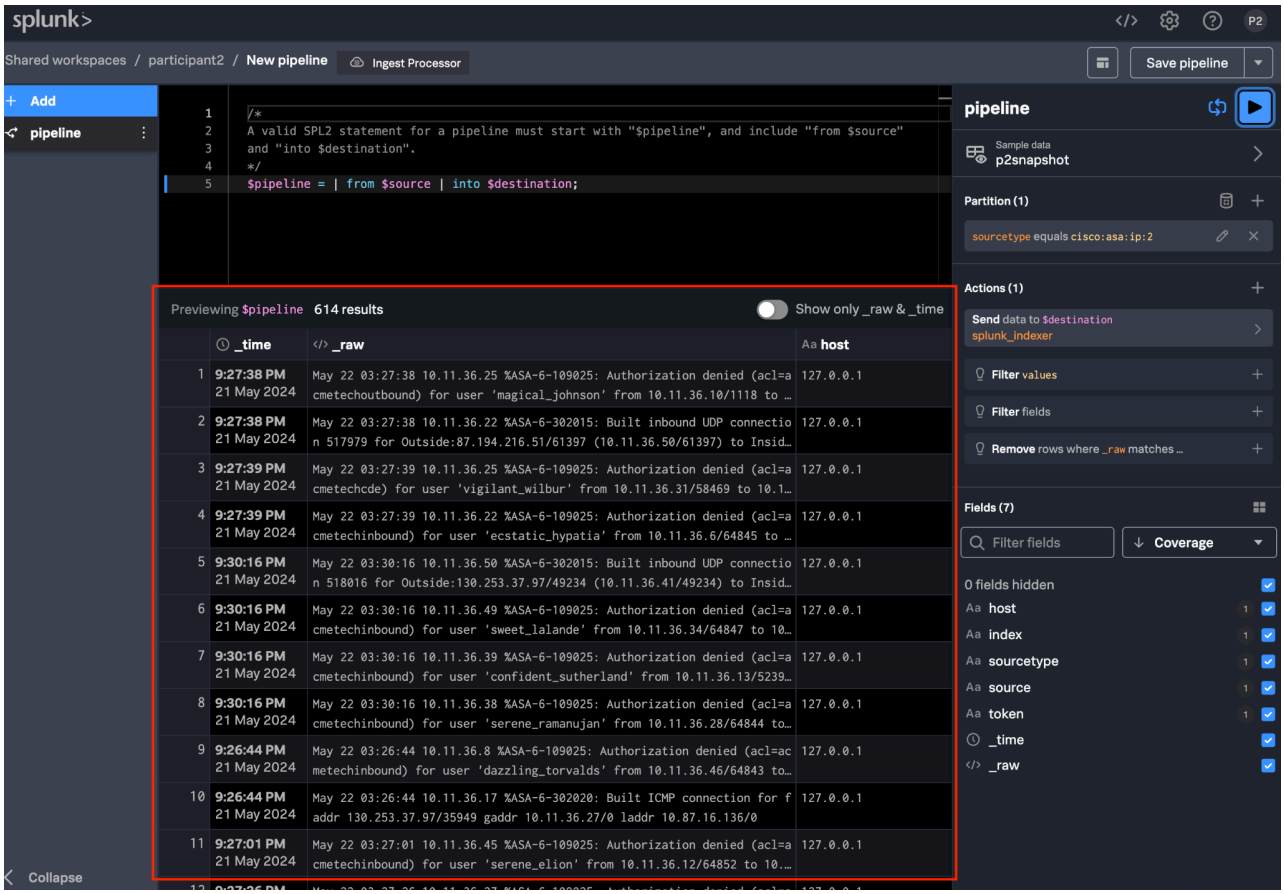
14. Leave the target index blank and select **Done**.



15. At this point, you will see the code editor experience, where you can write, edit, and test SPL2. This page should show a basic passthrough pipeline. Information about the pipeline is available on the right-hand side panel. The **Actions** section provides a UI-driven code editing experience with support for tasks like masking, filtering, selecting destinations, etc. To see the data that we captured in the snapshot in the previous steps, select the **play button** on the upper right.

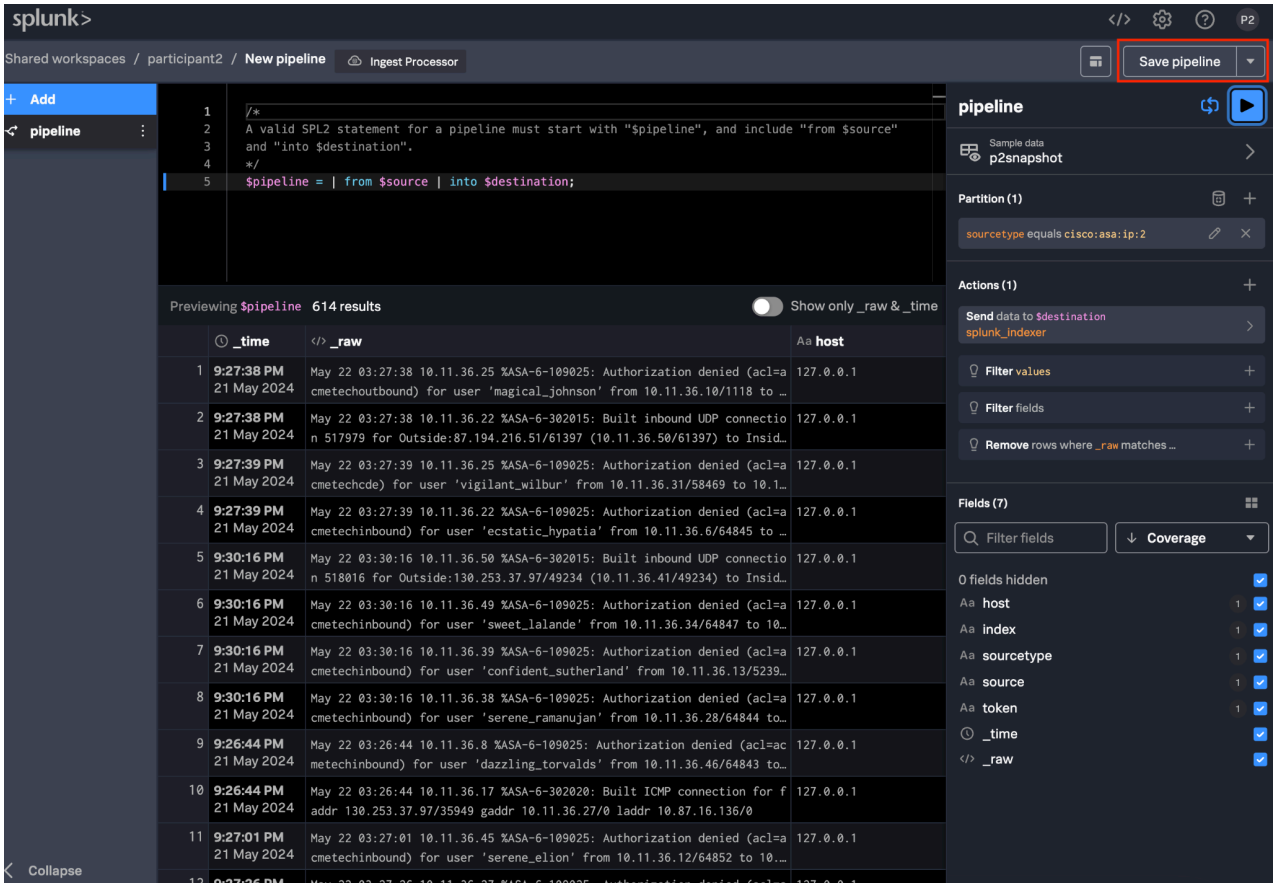


16. The snapshot data that has been captured so far is shown in the preview table below.



Save Your Pipeline Progress

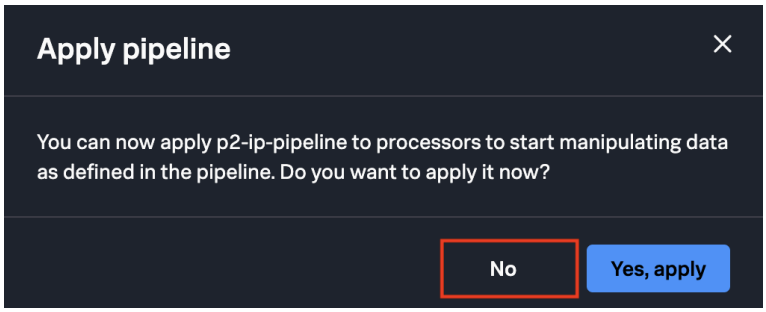
17. We can inspect the data in the table to understand the fields and sample values that were captured during ingestion to Splunk Cloud. Note that these records shown in the preview are raw events that were also sent to the indexer. To save our progress thus far before we start editing the SPL2, choose the **Save Pipeline** button on the top right.



18. In the Save Pipeline modal that appears, name the pipeline `p#-ip-pipeline` **REPLACING # WITH YOUR PARTICIPANT NUMBER** and select **Save**.

19. After saving, we will be prompted in another window to Apply the pipeline. The Apply step will push out the pipeline configuration and begin transforming the data according to the SPL2 we wrote. Since we haven't written any SPL2 yet, we will select **No**.

    Note: If you accidentally selected Yes, don't worry, we will overwrite the pipeline when we Apply our Cisco ASA SPL2 later.



Author Your Pipeline

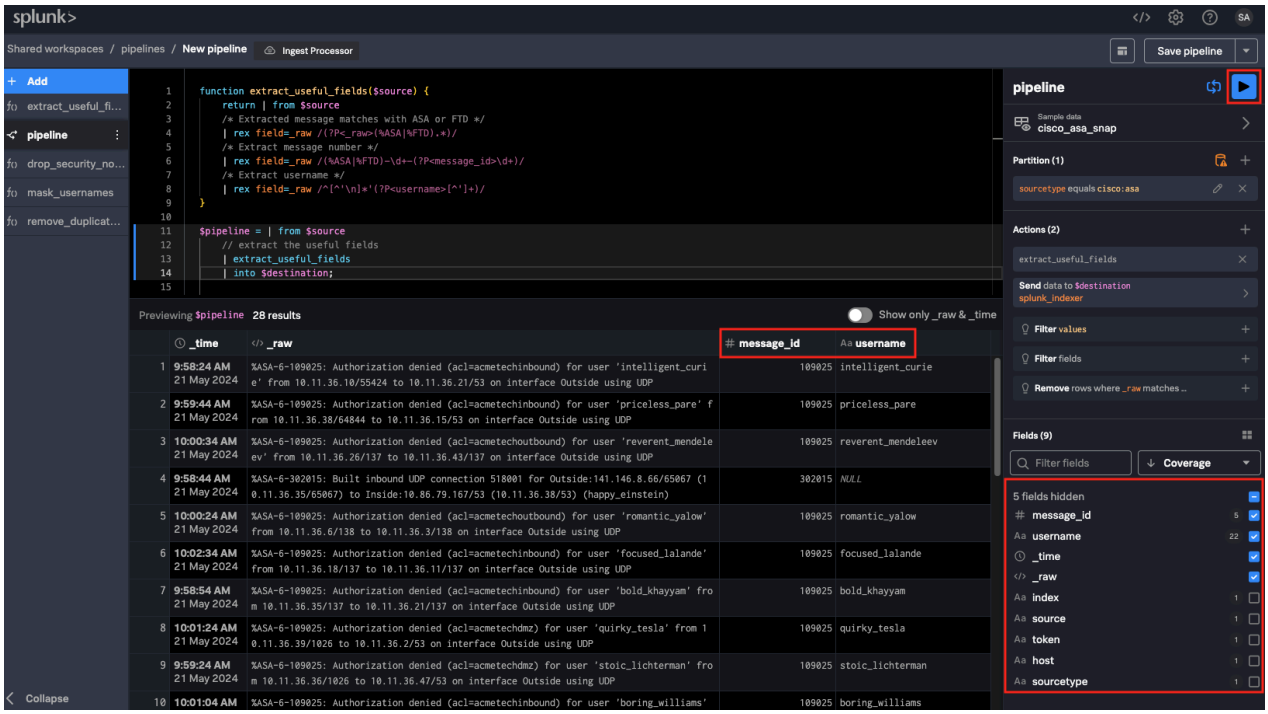
20. Now that we have our pipeline setup, we can start writing some SPL2! The first task will be to extract useful fields out of _raw. To do this, we will create an SPL2 function to extract the fields we care about. Copy/paste the following text into the SPL2 editor, replacing what was there before:

```
function extract_useful_fields($source) {
   return | from $source
   /* Extracted message matches with ASA or FTD */
   | rex field=_raw /(?P<_raw>(%ASA|%FTD).*)/
   /* Extract message number */
   | rex field=_raw /(%ASA|%FTD)-\d+-(?P<message_id>\d+)/
   /* Extract username */
   | rex field=_raw /^[^'\n]*'(?P<username>[^']+)/
}

$pipeline = | from $source
   // extract the useful fields
   | extract_useful_fields
   | into $destination;
```

The `rex` functions inside of `extract_useful_fields` will do the following:

- Ensure that all `_raw` fields start with either ASA or FTD
- Extract the message ID and store it in the message_id field
- Extract the username (if present) and store it in the username field

Pressing the preview “play” button again will show that the fields have been extracted. Note that sometimes the field list can get quite long, so we have unchecked several fields on the bottom right to reduce the size of the preview results table. You can also rearrange the columns however you want.

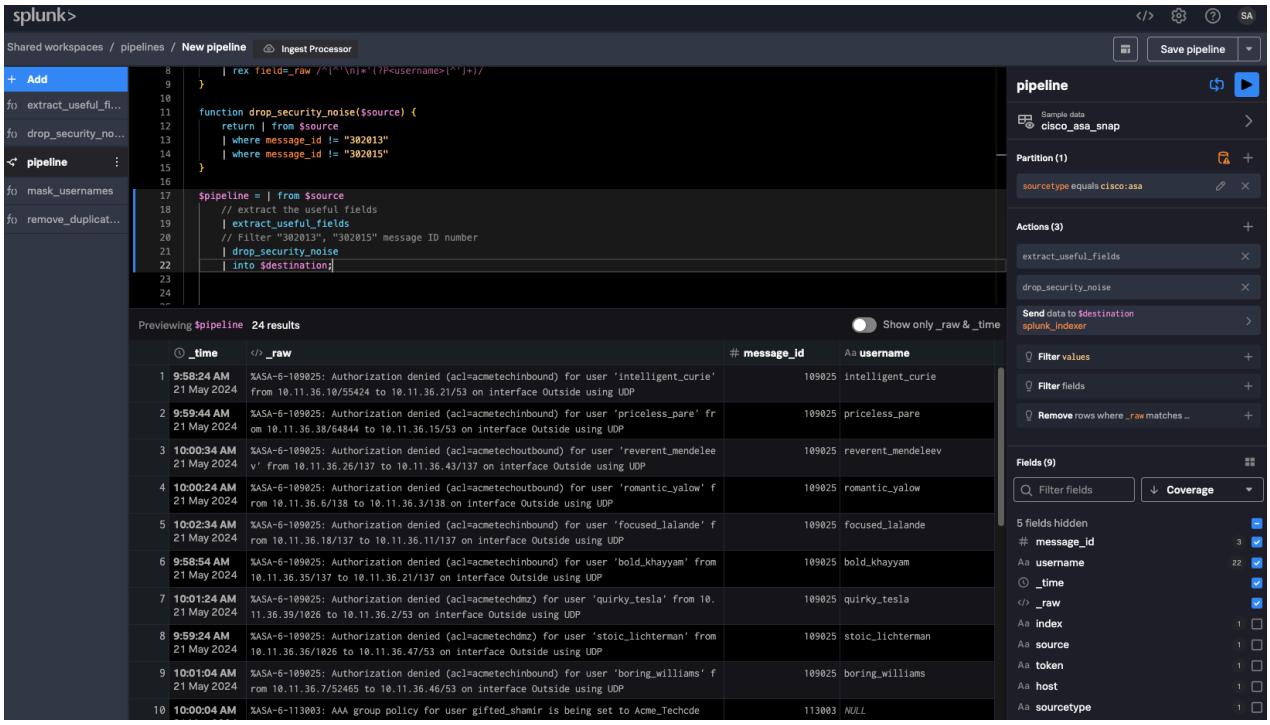


Filtering

21. Next, we are going to filter certain message IDs related to security that are noisy. Add the `drop_security_noise` SPL2 function to our editor, then pipe the parsed logs into our new function. The complete SPL2 now looks like:

```
function extract_useful_fields($source) {
   return | from $source
   /* Extracted message matches with ASA or FTD */
   | rex field=_raw /(?P<_raw>(%ASA|%FTD).*)/
   /* Extract message number */
   | rex field=_raw /(%ASA|%FTD)-\d+-(?P<message_id>\d+)/
   /* Extract username */
   | rex field=_raw /^[^'\n]*'(?P<username>[^']+)/
}

function drop_security_noise($source) {
   return | from $source
   | where message_id != "302013"
   | where message_id != "302015"
}

$pipeline = | from $source
   // extract the useful fields
   | extract_useful_fields
   // Filter "302013", "302015" message ID number
   | drop_security_noise
   | into $destination;
```

Preview again and notice the 302013 and 302015 message IDs have been filtered.



Masking

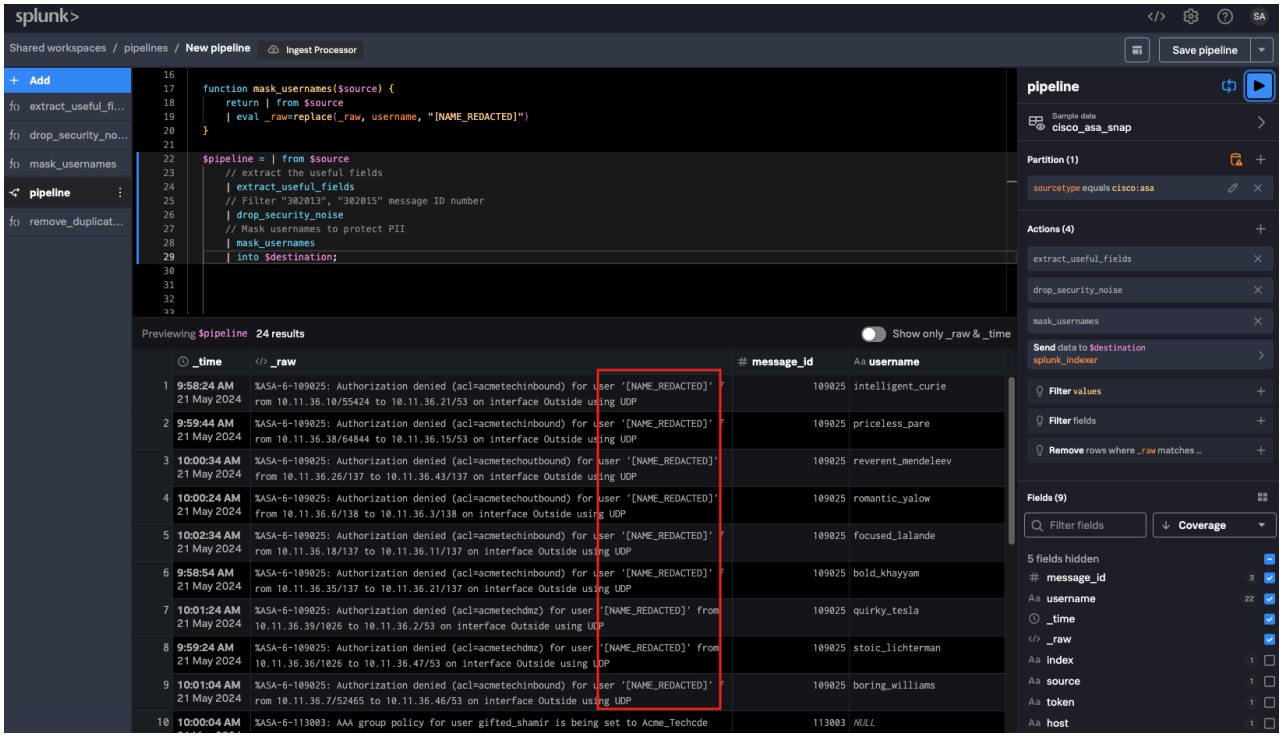
22. When we send this data to the index, we've decided to mask the usernames to maintain compliance. Here's the mask_usernames function to replace usernames found in _raw:

```
function mask_usernames($source) {
   return | from $source
   | eval _raw=replace(_raw, username, "[NAME_REDACTED]")
}
```

Don't forget to invoke `mask_usernames` in your pipeline:

```
$pipeline = | from $source
   // extract the useful fields
   | extract_useful_fields
   // Filter "302013", "302015" message ID number
   | drop_security_noise
   // Mask usernames to protect PII
   | mask_usernames
   | into $destination;
```

When previewing your results, you will see the username has been redacted in _raw. Note that the username is still present in the username field.



Drop Fields & Set Index

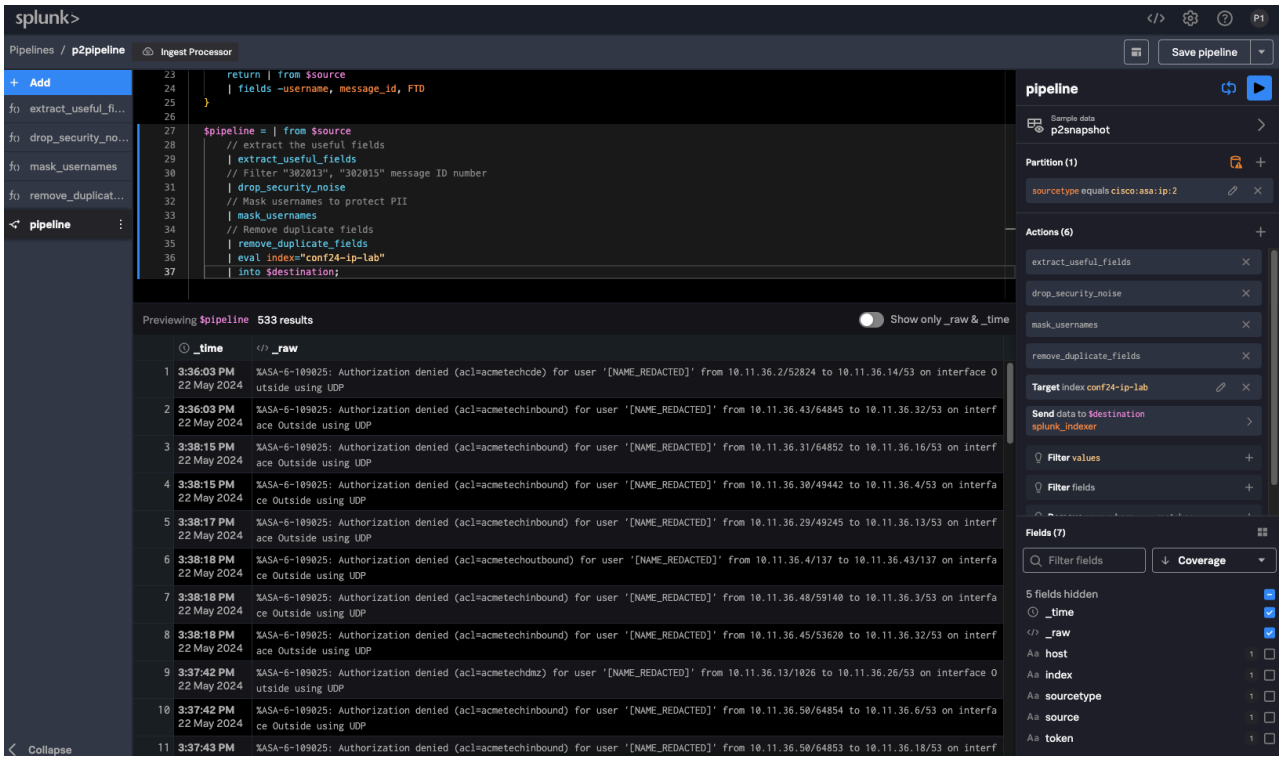
23. Finally, we can remove the parsed fields as they are no longer needed, and we don't want them sent to the indexer. Create the remove_duplicate_fields function:

```
function remove_duplicate_fields($source) {
   return | from $source
   | fields -username, message_id, FTD
}
```

Add the `remove_duplicate_fields` function to your pipeline definition and set the index to `dmx-ip-lab`.

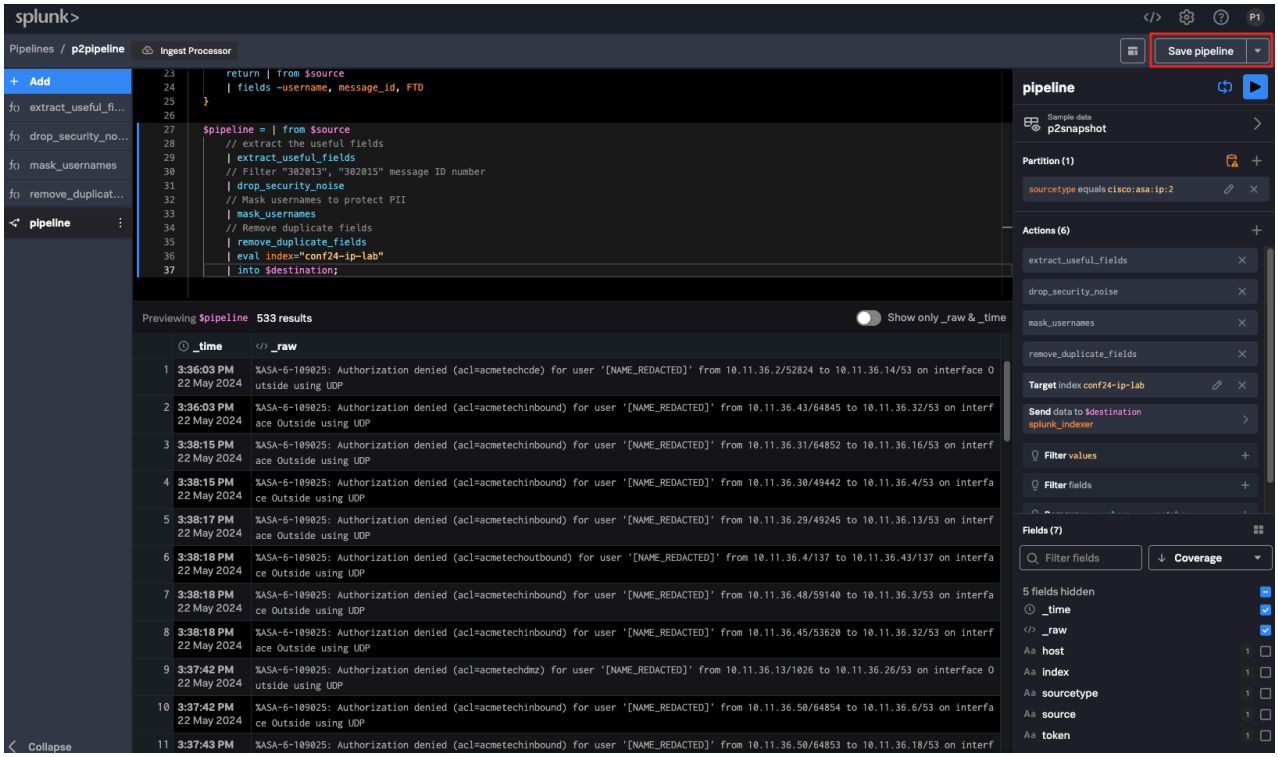
```
$pipeline = | from $source
   // extract the useful fields
   | extract_useful_fields
   // Filter "302013", "302015" message ID number
   | drop_security_noise
   // Mask usernames to protect PII
   | mask_usernames
   // Remove duplicate fields
   | remove_duplicate_fields
   | eval index="dmx-ip-lab"
   | into $destination;
```

Preview now shows the results without these fields:

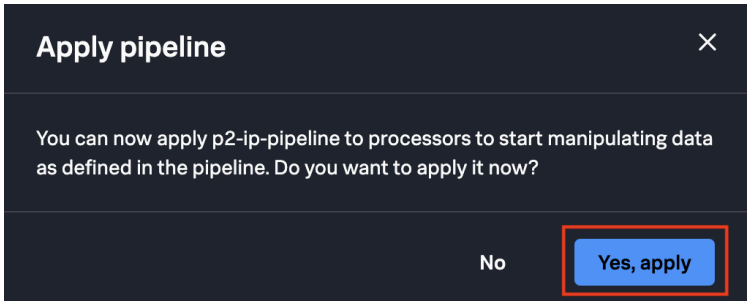


Apply Your Pipeline

24. We are satisfied with the results of our pipeline, so we will now apply it. Select the Save pipeline button at the top right.



If prompted, select **Yes, apply** to apply the pipeline. If you are not prompted, the pipeline will be automatically applied on save.



Check Your Data in Splunk Cloud

25. 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=dmx-ip-lab sourcetype=cisco:asa:ip:#
    ```

    (REPLACING # WITH YOUR PARTICIPANT NUMBER)

Convert Logs To Metrics

We can instrument our incoming Cisco ASA log data using IP's capability to convert logs to metrics and send them to Splunk Observability (o11y). To illustrate this capability, we will create a counter metric that increments by one each time we see a Cisco ASA log. We will also set message_id as a dimension, allowing us to compare message counts by ID. The metrics will be available in the Splunk Observability UI.

26. In the SPL2 code editor, at the top of the file, add the logs_to_metrics import statement:

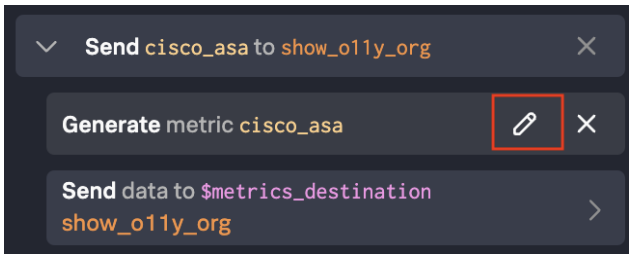
    ```
    import logs_to_metrics from /splunk.ingest.commands
    ```

    Replace the definition of $pipeline with one that contains a `thru` command:

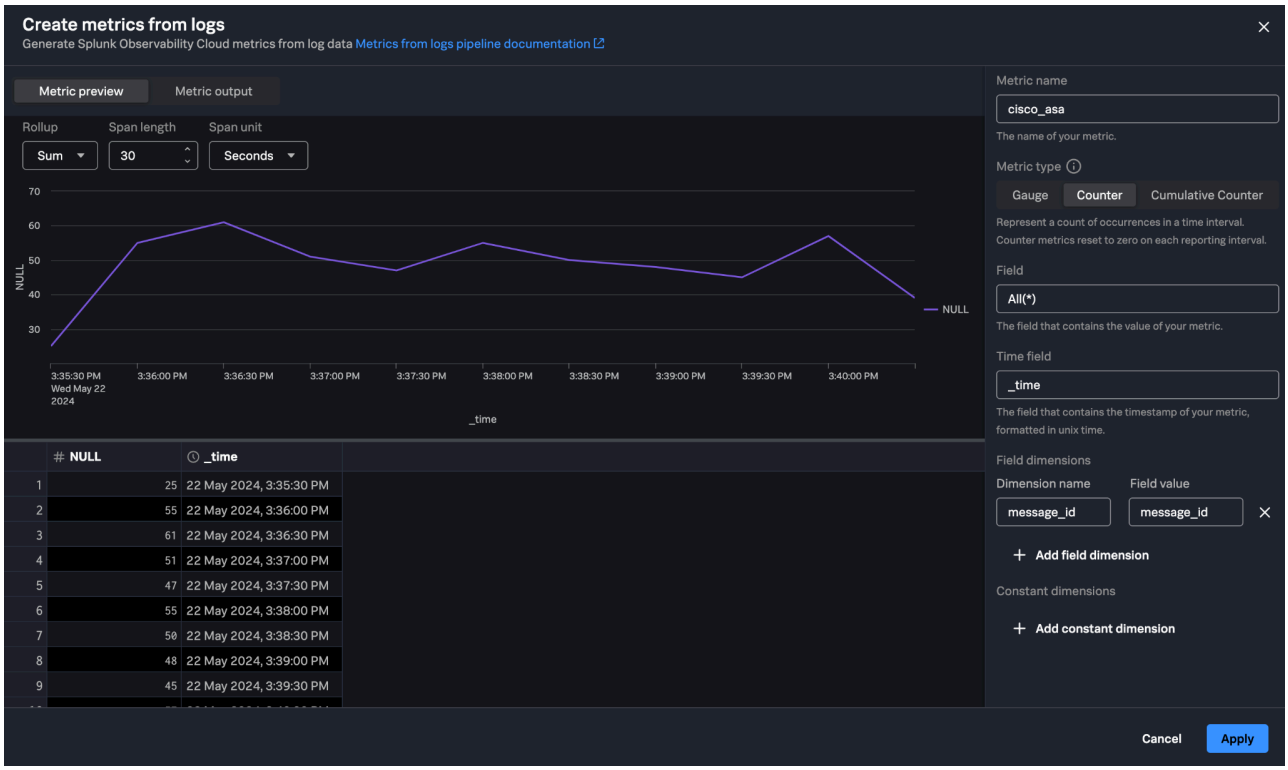
    ```
    $pipeline = | from $source
       // extract the useful fields
       | extract_useful_fields
       // Filter "302013", "302015" message ID number
       | drop_security_noise
       // Mask usernames to protect PII
       | mask_usernames
       | thru [
            | logs_to_metrics time=_time name="cisco_asa" value=1 metrictype="counter"
    dimensions={"message_id": message_id}
            | into $metrics_destination
       ]
       // Remove duplicate fields
       | remove_duplicate_fields
       | eval index="dmx-ip-lab"
       | into $destination;
    ```

    The `thru` command creates a branch where each path gets a copy of the data, similar to the Unix "tee" command. In this way we are able to continue sending data to the Splunk indexer while also sending a copy of all events down the logs_to_metrics path.

27. Several new actions are available in the right-hand panel. Choose the “Edit” icon on the “Generate metrics cisco_asa” to see a UI-driven metricization experience, including a preview of the metrics.

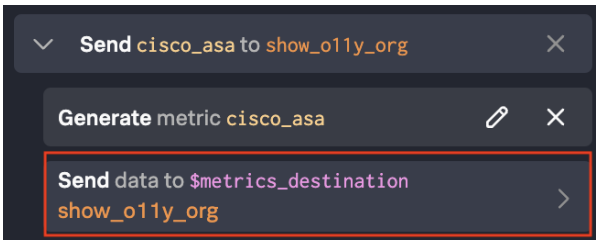


You will see a screen similar to this:



Choose cancel to exit out of this screen.

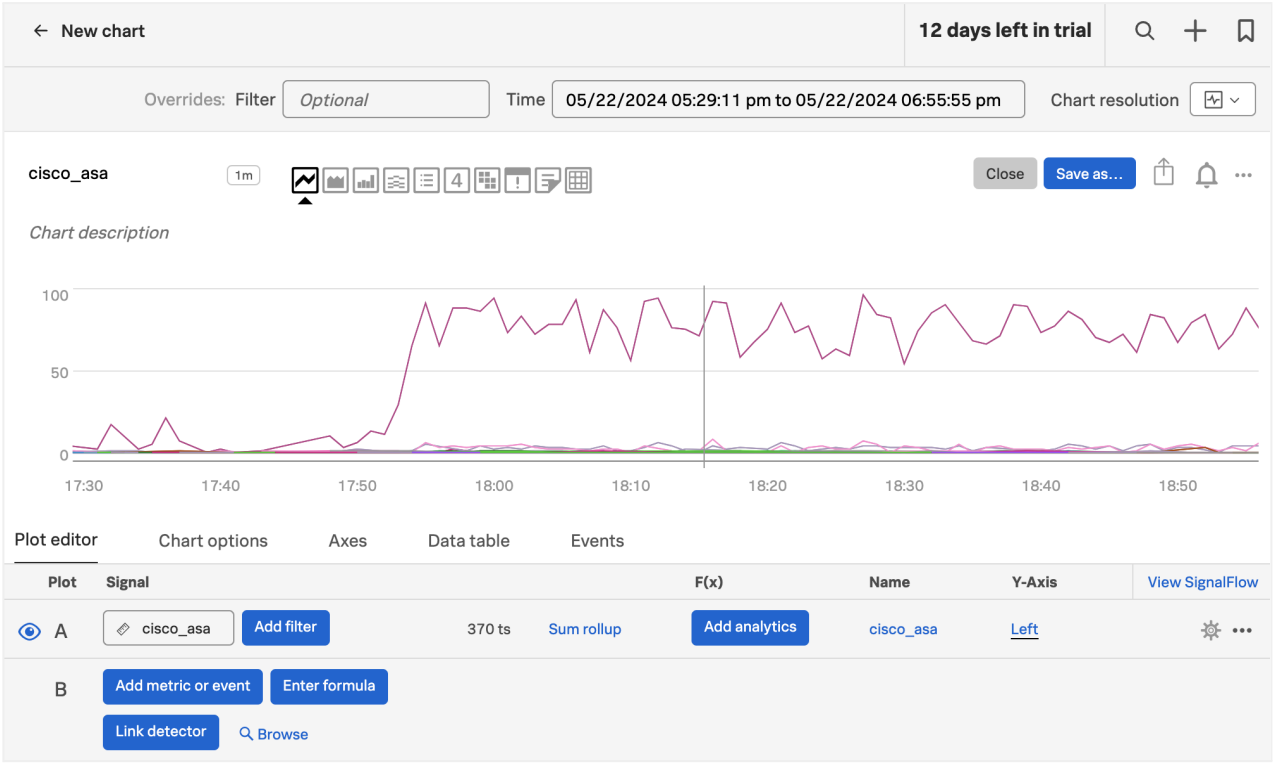
28. Because we selected show_o11y_org as our default metrics destination when we created this pipeline, **it should automatically be selected for you**. However if no metrics destination is selected, you can click on **Send data to $metrics_destination** and choose the o11y destination if necessary.



29. Save and apply your pipeline. IP will now send metrics to Splunk Observability while continuing to send logs to the Splunk Indexer.

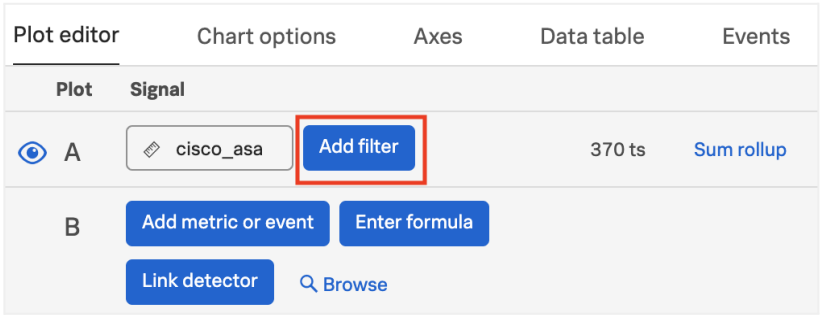
Check Your Data in Splunk Observability

30. Log into Splunk Observability using this link with the username and password provided in Lab Connection Info. If you open the link and only see a white or black screen, please log out of your existing Splunk Observability (SignalFx) account prior to proceeding (choose Settings -> Sign Out).

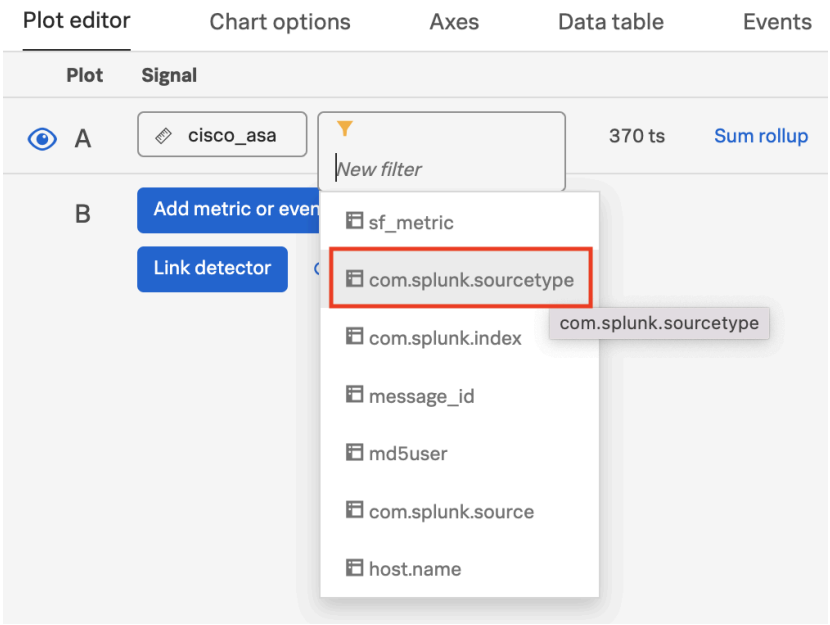


You will be taken immediately to the Metric Finder, which displays the metric values for the cisco_asa metric from the last 15 minutes. Metrics from other participants will also be present in this chart. Use the filtering functionality shown below to filter metrics to those sent by your IP pipeline only.

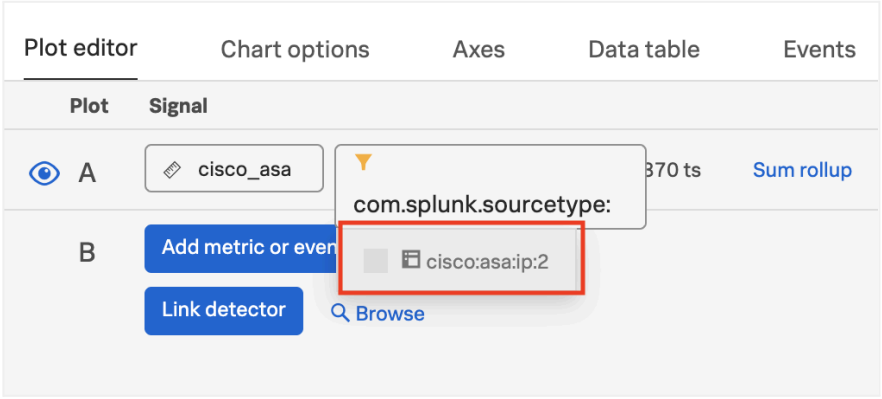
To filter by sourcetype, select Add filter → com.splunk.sourcetype → cisco:asa:ip:#.



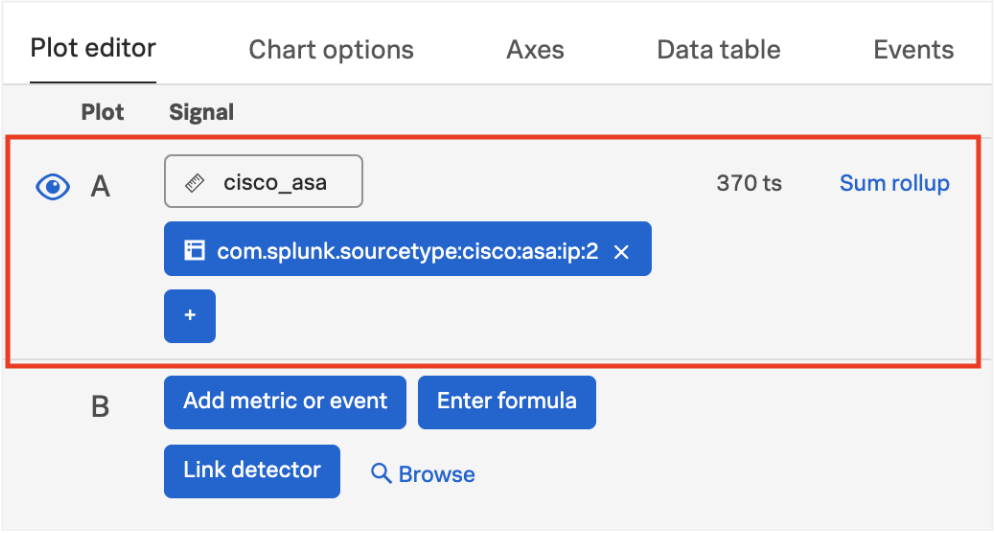
Then choose com.splunk.sourcetype.



And select the sourcetype corresponding to your participant ID.



Once selected, you will see the new filter applied to the metric.



Once your participant sourcetype has been filtered, you can confirm that Splunk o11y is receiving your metrics.

Lab Summary

In this workshop, we explored three flexible and powerful ways to ingest data into the Splunk environment.

Ingest Actions (IA) allow you to quickly author, preview, and deploy transformation rules at ingest time through an intuitive user interface. These rulesets allow you to define one or more rules with a few clicks, masking, truncating, routing, or eliminating data without having to handwrite stanzas in configuration files.

In this workshop, we used Ingest Actions to route certain higher-volume, lower-value data sources to lower-cost S3 storage for future use with Splunk Federated Search for S3.

Edge Processor (EP) goes beyond the built-in “getting data in (GDI)” functionality available in the core platform with Ingest Actions. EP uses dedicated Edge Nodes to route, filter, mask, transform, and convert data where it resides in the customer-managed infrastructure. Edge Nodes are high-performance OTel data processors that reside proximally to the source data.

As we explored today, Edge Nodes are managed centrally by a Splunk-hosted management user interface. In the lab, we created a workspace, uploaded sample data, and built a data pipeline using SPL2. We then enriched the data in the stream via a lookup and validated functionality.

Ingest Processor (IP) is a data processing capability that works within your Splunk Cloud Platform deployment. Use IP to configure data flows, control data format, apply transformation rules prior to indexing, and route to destinations. The Ingest Processor solution is suitable for Splunk Cloud Platform administrators who use forwarders or HTTP Event Collector (HEC) to get data into their deployments. Ingest Processor. While IP/EP management interfaces are quite similar in scope and capability, the IP removes the administrative burden of managing and maintaining edge nodes.

In the lab, we extended the base use case that we built in the IA and EP labs. We created a workspace and a pipeline to process data in the stream. We were able to preview, save, and deploy filtering, masking, and routing use cases. Finally, we converted log data to metric-formatted data and reviewed our work.

The workshop authors hope that you enjoyed these labs and are able to take with you knowledge of both how to manage your data and when to use each Splunk capability to build more effective data pipelines.

Thank you for attending today’s workshop. Happy Splunking!

Get Started with Splunk Data Management Today

Request access to Edge Processor 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.