Synadia hands-on platform workshop

Table of Contents

Introduction 1
Prerequisites 2
- Install NATS 2
- Install Go (optional) 2
- Install NATS CLI 2
- Create Synadia Cloud account. 3

Hands-on workshop 3
- Create Account 4
- Create user 4
- Download user creds. 5
- Create JetStream 6
- Create Consumer 7
- Test out publishing messages 8
- Flow control (cross account access) demo 10
- LeafNodesDemo 12

Appendix A: Enable Connectors and Workloads on the Account 14
Appendix B: Connector demo 14

Introduction

In this workshop we will be looking at different aspects of the Synadia Control Plane.

We will explore:

1. Creating Accounts and users
2. Creating streams (JetStream)
3. Publishing and consuming events to/from a stream
4. Observing some of the metrics and stats
5. Connectors (demo)
6. Optional
    1. do cross-account data movement

Prerequisites

Install NATS

https://nats.io/download/

Install Go (optional)

The components are all binaries so setting up GO is not a requirement however, it is a good to have it installed.

https://go.dev/dl/

https://go.dev/doc/install

Install NATS CLI

There are many ways to install the NATS CLI:

Generic

curl -sf https://binaries.nats.dev/nats-io/natscli/nats@latest | sh sudo mv nats /usr/local/bin/

Go-specific

go install github.com/nats-io/natscli/nats@latest

Mac

brew tap nats-io/nats-tools

brew install nats-io/nats-tools/nats

`Windows powershell`

scoop bucket add extras
scoop install extras/natscli

Post installation, you can check to see if thh CLI was installed:

nats --version

Create Synadia Cloud account.

To create a new Synadia Cloud account, one needs to do the following:

1. Open your web browser and go to the Synadia Cloud Portal.
2. Choose your preferred signup method. You can register using your existing **Google** or **GitHub** accounts, or by using a standard **Email and password**.
3. Verify your email if prompted.
4. Once you log in, you will be directed to the main dashboard, which automatically provides access to NGS (Synadia's global NATS system).

There is a short video that you can watch that gives you an overview of the Control Plane.

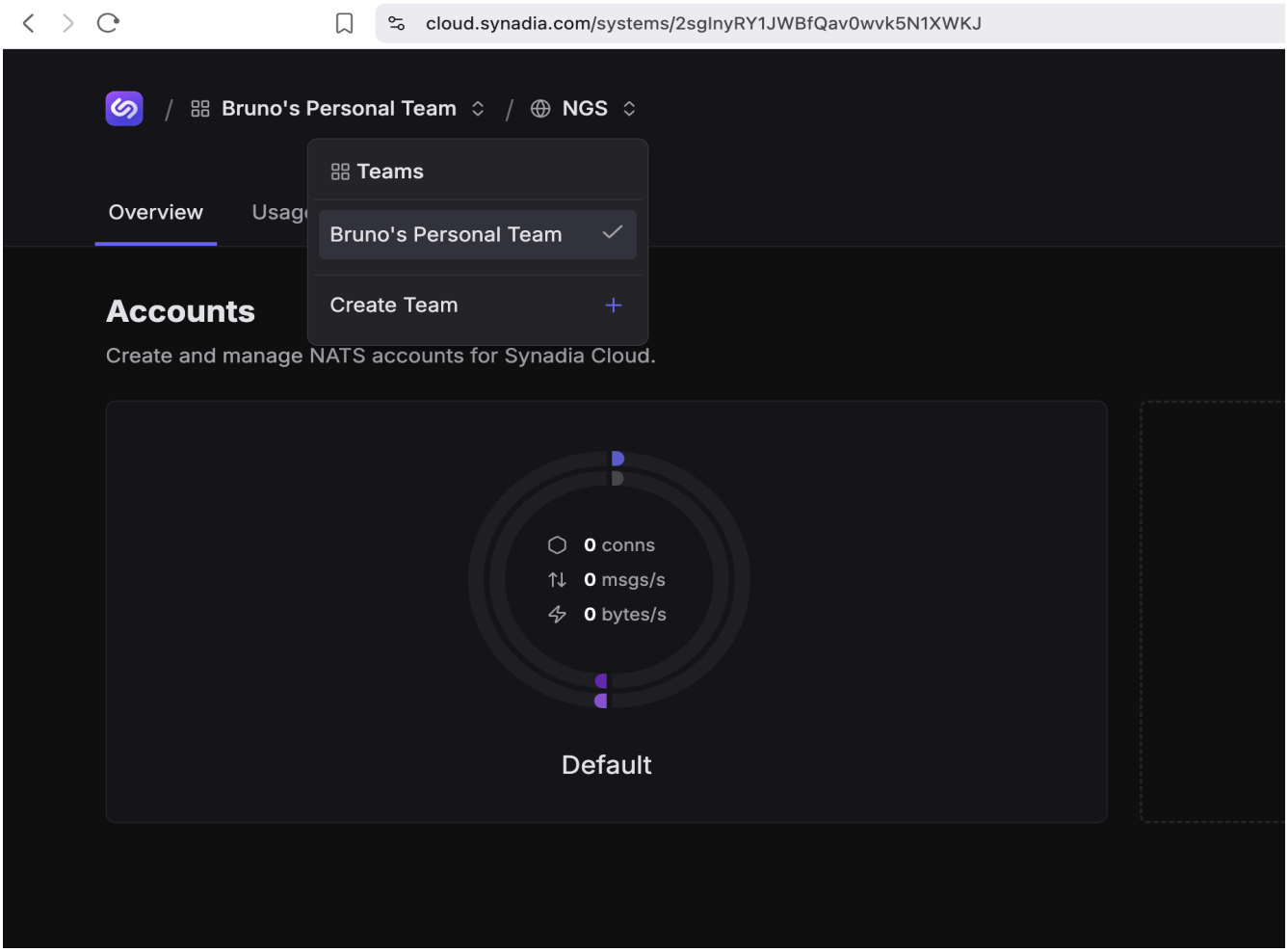
https://www.youtube.com/watch?v=4nLMAXvIsjE

Hands-on workshop

As you log in to the platform you can see the default team that it created for your user.

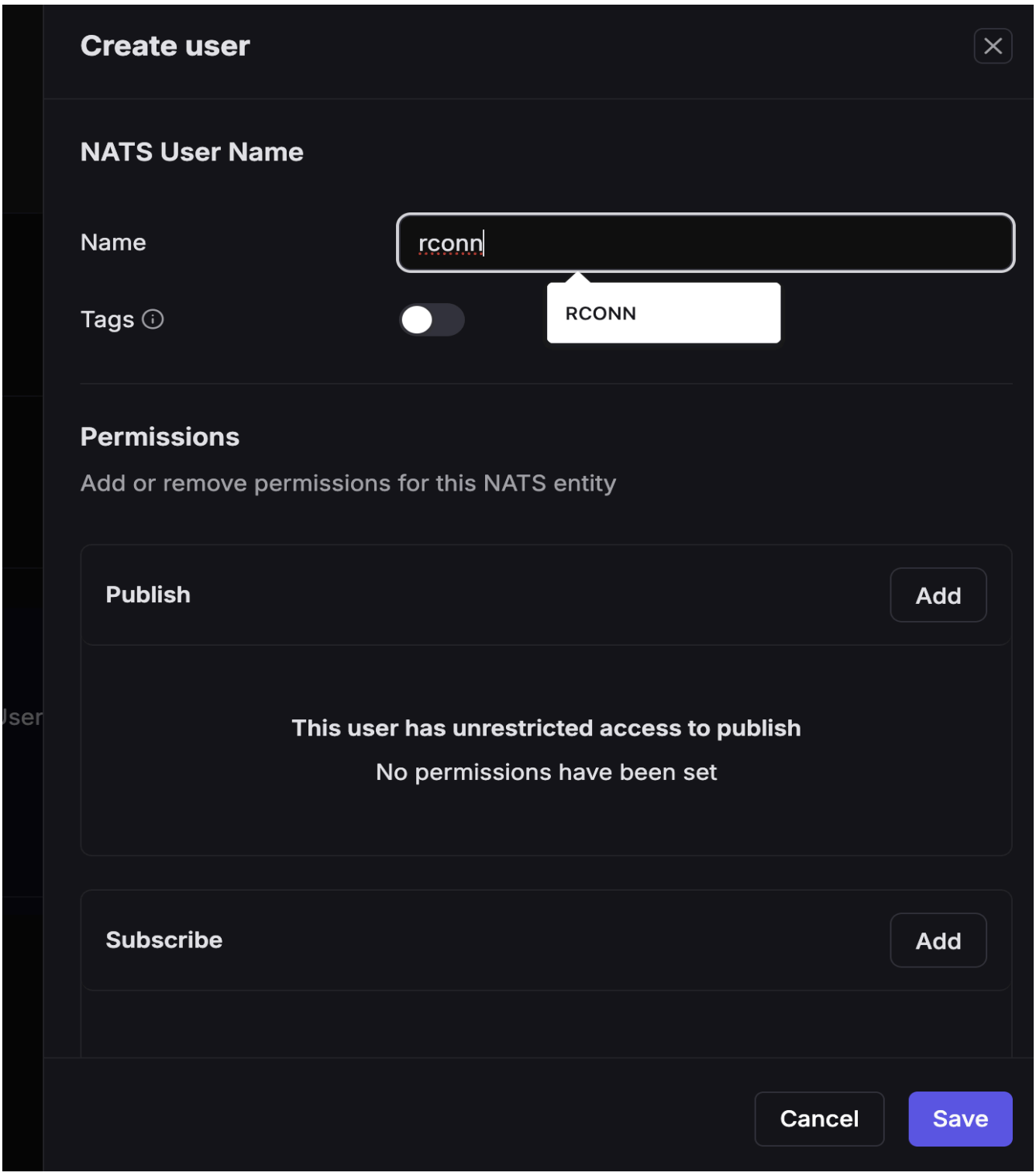
You can of course create other teams as well, if you so desire.

You can now create a new account: for Rethink CONN (RCONN).



Create Account

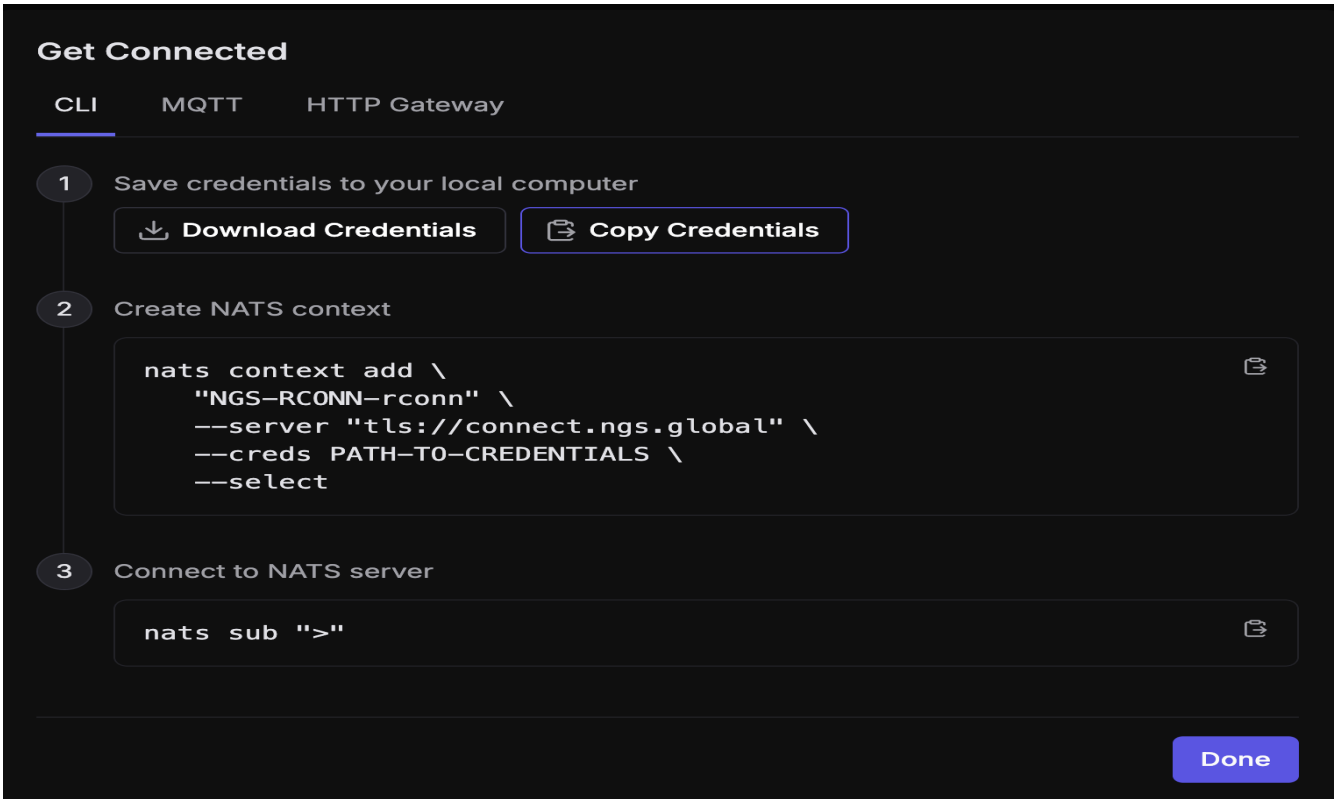
Create user



Download user creds.

Go into the rconn user, and click the “Get Connected” button.

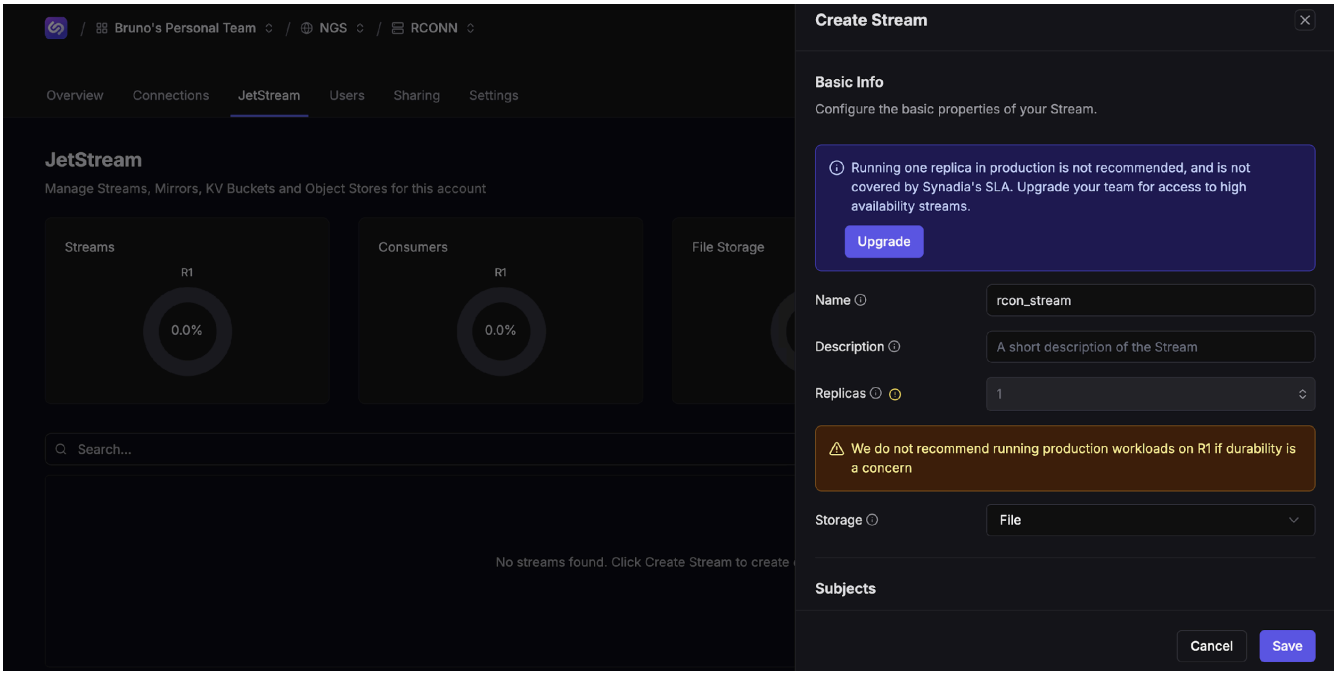
Then select the Download credentials option.



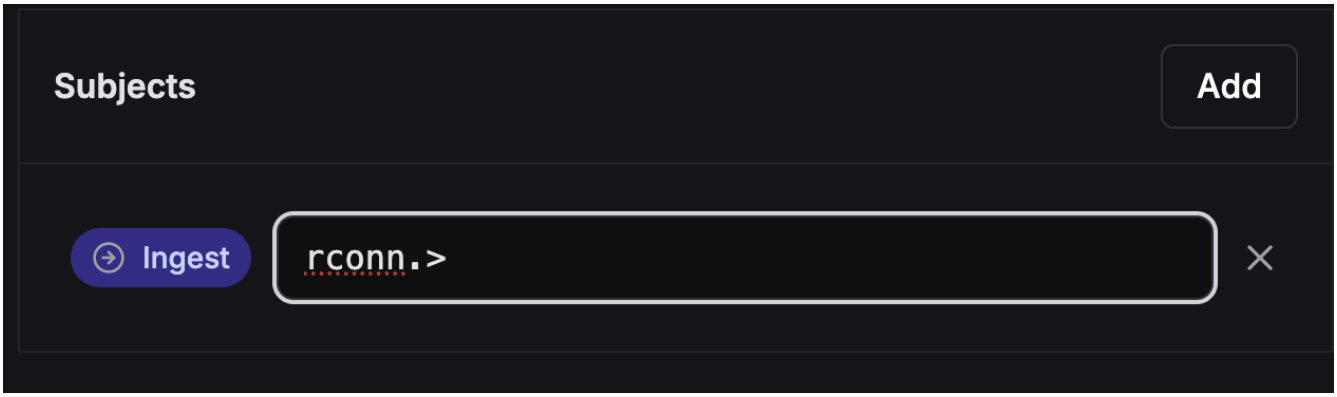
This should save a file (NGS-RCONN-rconn.creds) to a folder of your choosing.

Create JetStream

As we are creating this on the trial account you can ignore the warnings

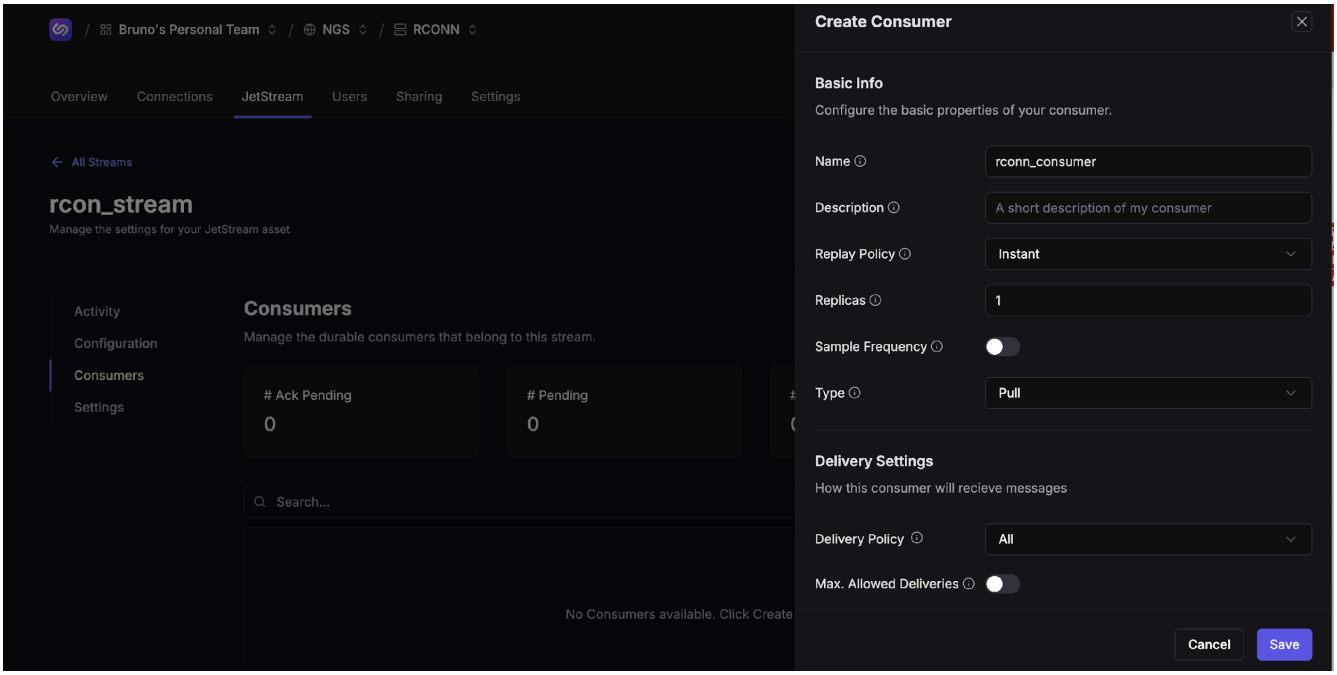


Select a subject that your stream will be ingesting from.



You can keep the rest of the parameters as default.

Create Consumer



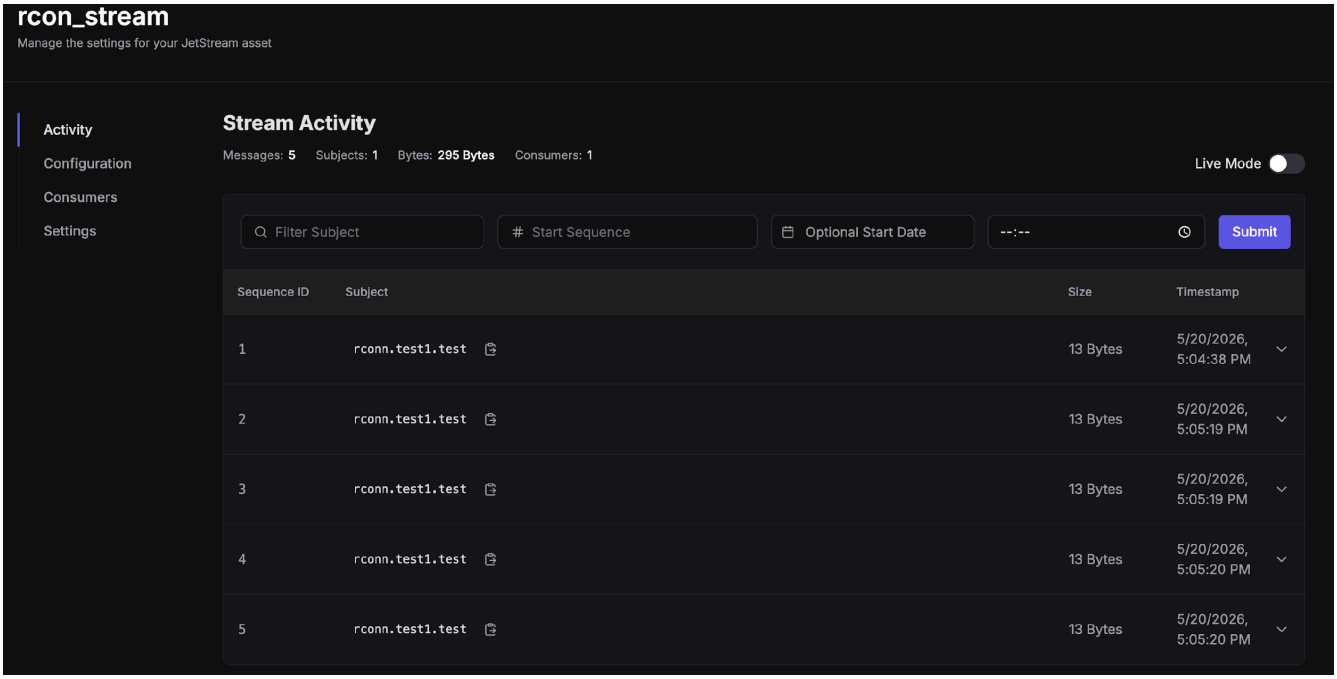
Test out publishing messages

Publish some messages to the srteam:

nats pub rconn.test1.test "test1 message" --server nats://connect.ngs.global –creds creds/NGS-RCONN-rconn.creds --no-context

Publish additional 4 messages:

nats pub rconn.test1.test "test1 message" --server nats://connect.ngs.global --creds creds/NGS-RCONN-rconn.creds --no-context --count 4
17:05:19 Published 13 bytes to "rconn.test1.test"
17:05:19 Published 13 bytes to "rconn.test1.test"
17:05:20 Published 13 bytes to "rconn.test1.test"
17:05:20 Published 13 bytes to "rconn.test1.test"



Create a consumer listener on the newly created consumer, and consume a few messages one by one

nats consumer next rconn_stream rconn_consumer --creds creds/NGS-RCONN-rconn.creds --server nats://connect.ngs.global --no-context
? Select a Consumer rconn_consumer
[17:07:55] subj: rconn.test1.test / tries: 1 / cons seq: 1 / str seq: 1 / pending: 4

test1 message

Acknowledged message

Do 2 more message consumption but this time consume a group of them at the same time:

nats consumer next rcon_stream rconn_consumer_all --creds creds/NGS-RCONN-rconn.creds --server nats://connect.ngs.global --count 2 --no-context
[17:28:02] subj: rconn.test1.test / tries: 1 / cons seq: 1 / str seq: 1 / pending: 4

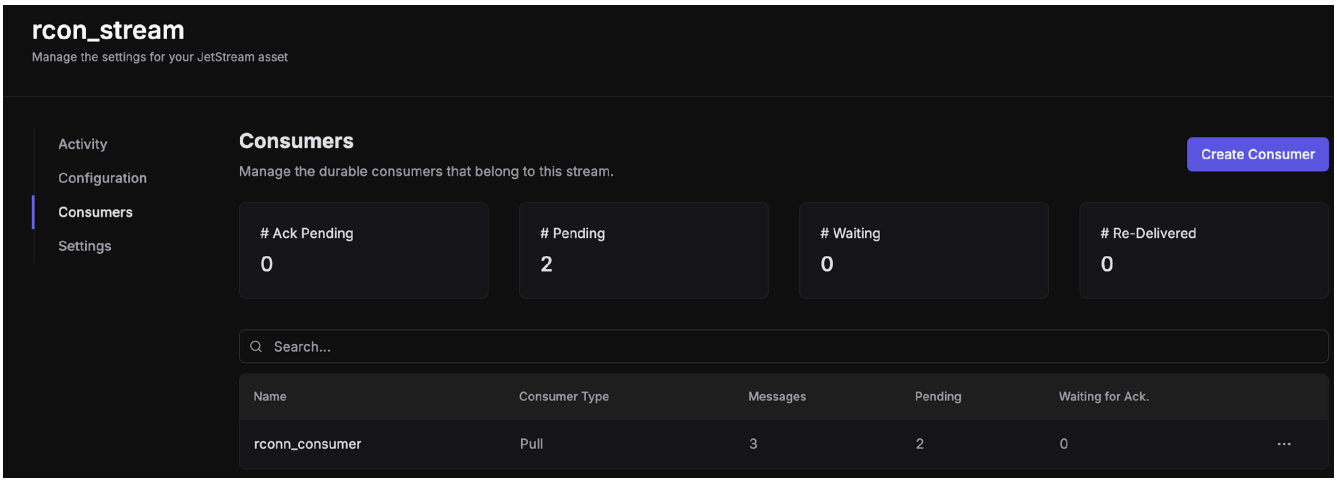
test1 message

Acknowledged message

[17:28:03] subj: rconn.test1.test / tries: 1 / cons seq: 2 / str seq: 2 / pending: 3

test1 message

Acknowledged message



Flow control (cross account access) demo

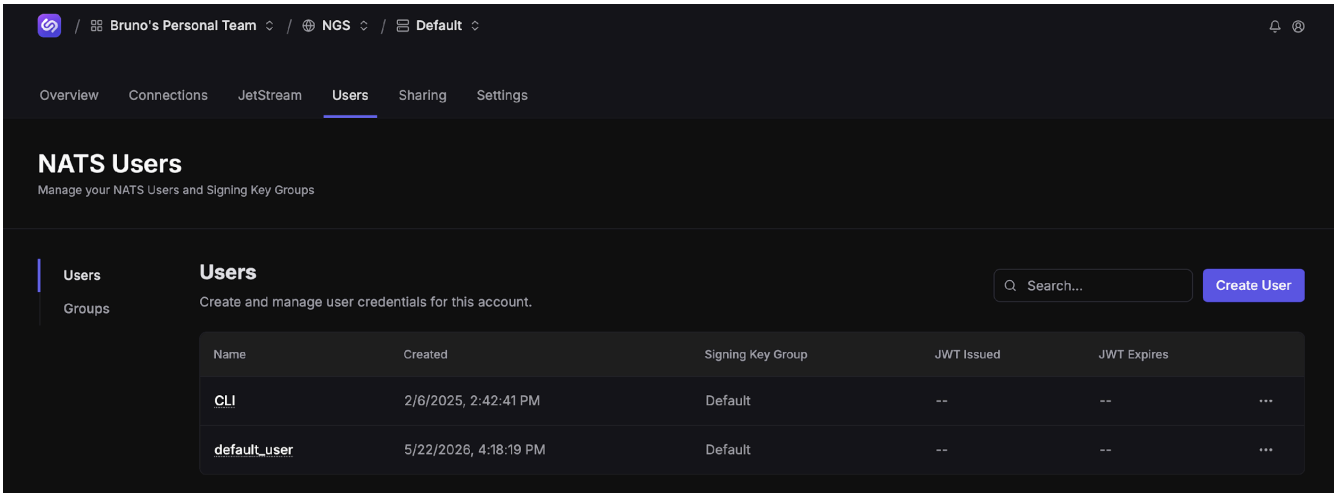
In this we will test out how data can be shared across Account boundaries.

As a preamble let look at some terms: services vs. streams.

If we define a service in Account A that means that user in account A can only publish to that service. Conversely if we import that service into account B, users in account B can only publish to that service. The same applies to stream however in reverse: you can publish to a feed in Account A but you can only consume from a feed in Account B.

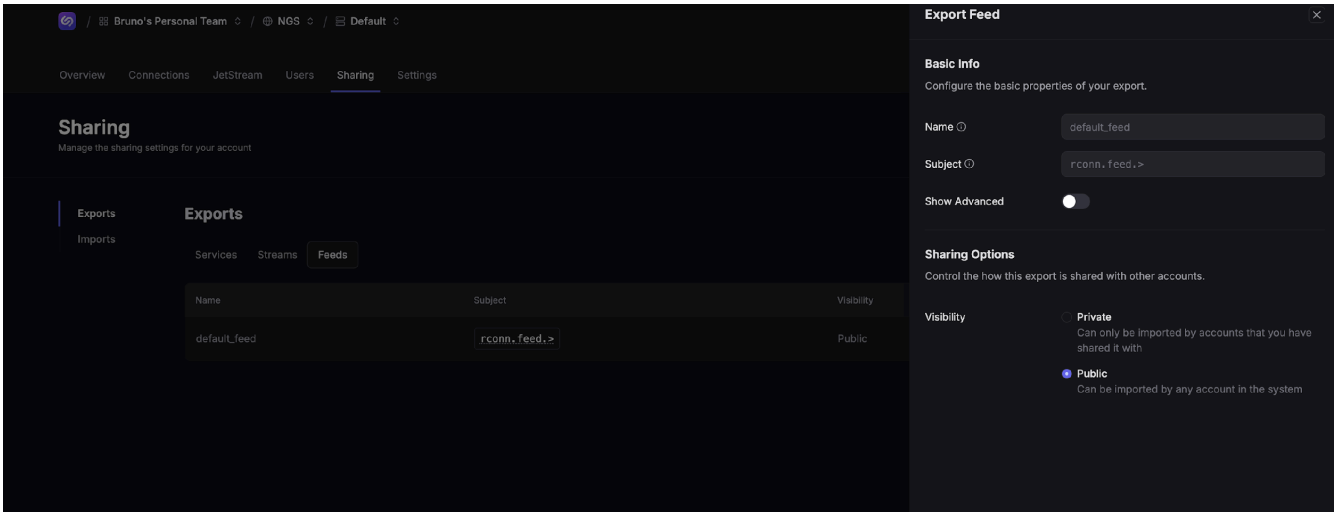


We start by creating another account and user (RCONN2/rconn2), or if you cannot, as you maybe only be able to create one account on a trial version, then just create a user in the Default/default_user account. Use all the defaults.



Download the default_user credentials.

Create a **feed** in Default and export it.

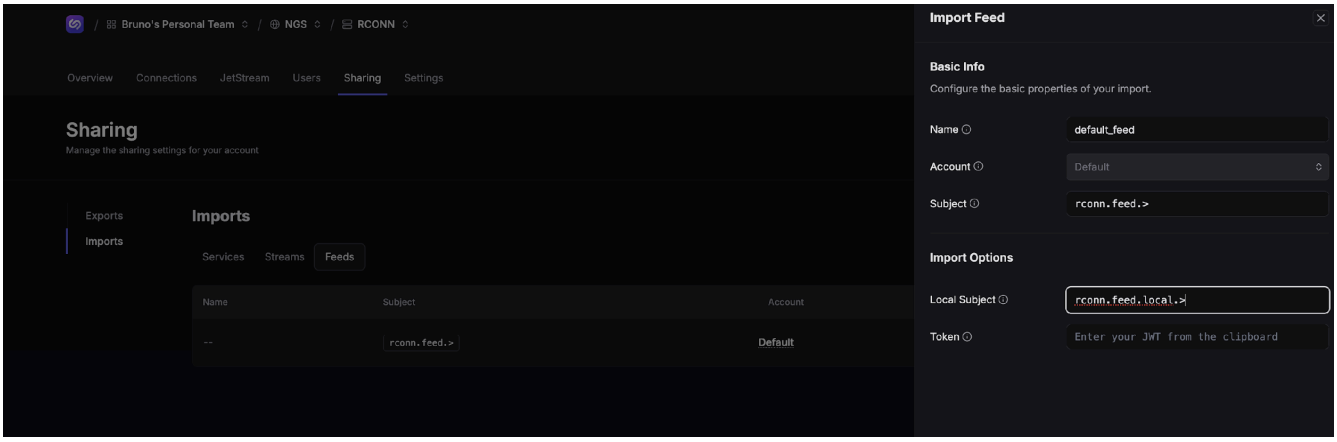


Now you can publish an event on the feed subject in the Default account and try to listen on it on the same subject in the RCONN account but you will se nothing.

```
nats pub rconn.feed.test "test feed  message" --server nats://connect.ngs.global  --creds creds/NGS-Default-default_user.creds --no-context
```

```
nats sub "rconn.feed.>" --creds creds/NGS-RCONN-rconn.creds --server nats://connect.ngs.global --no-context
```

Now Import the feed into RCONN.



NOTE: *here we are mapping the rconn.feed.> exported from the Default acount to a local subject: rconn.feed.local.>*

Let's try again now.

NOTE: if we listened on the rcon.feed.> subject we should get the message, however in our case we need to try out the local subject mapping so we will listen on rcon.feed.local.>

```
nats sub "rconn.feed.local.>" --creds creds/NGS-RCONN-rconn.creds --server nats://connect.ngs.global --no-context
17:00:43 Subscribing on rconn.feed.local.>
[#1] Received on "rconn.feed.local.test"
test feed  message
```

```
nats pub rconn.feed.test "test feed  message" --server nats://connect.ngs.global  --creds creds/NGS-Default-default_user.creds –no-context
```

LeafNodesDemo

Use the NATS server config for in the ngs_leaf_node folder.

Note: make sure you download the NGS/Synadia Cloud credentials file, and update the NATS server config file to reflect the location of the creds file:

```
  credentials: "creds/NGS-RCONN-rconn.creds"
```

The first use case we will just publish an event on a leaf node, and we will see it consumed on the Cloud. This illustrates how interest propagation works.

Start the leaf node:

```
nats-server -c ngs_leaf_node/ngs_leaf_node.conf
```

Start a listener on the Synadia Cloud:

```
nats sub rconn.leaf.* -s nats://connect.ngs.global  --creds creds/NGS-RCONN-rconn.creds
--no-context
11:33:29 Subscribing on rconn.leaf.*
```

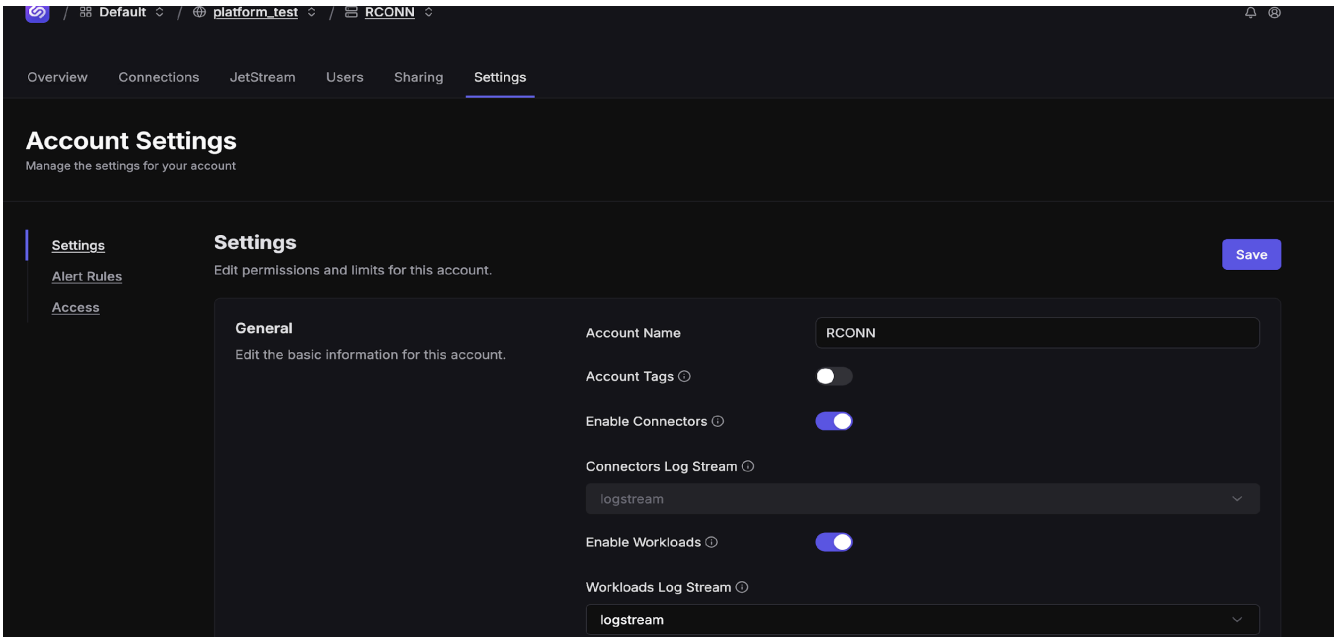
Publish a message on the Leafnode:

```
nats pub rconn.leaf.test "testing leaf node" --user leaf_user --password leaf_user -s
nats://localhost:4747 --no-context
```

You should now see the event having been received by the Synadia Cloud client:

```
nats sub rconn.leaf.* -s nats://connect.ngs.global  --creds creds/NGS-RCONN-rconn.creds
--no-context
11:33:29 Subscribing on rconn.leaf.*
[#1] Received on "rconn.leaf.test"
testing leaf node
```

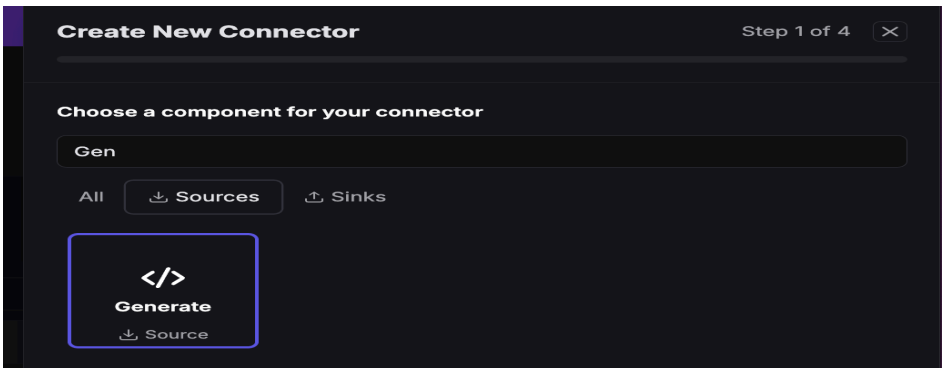
Appendix A: Enable Connectors and Workloads on the Account

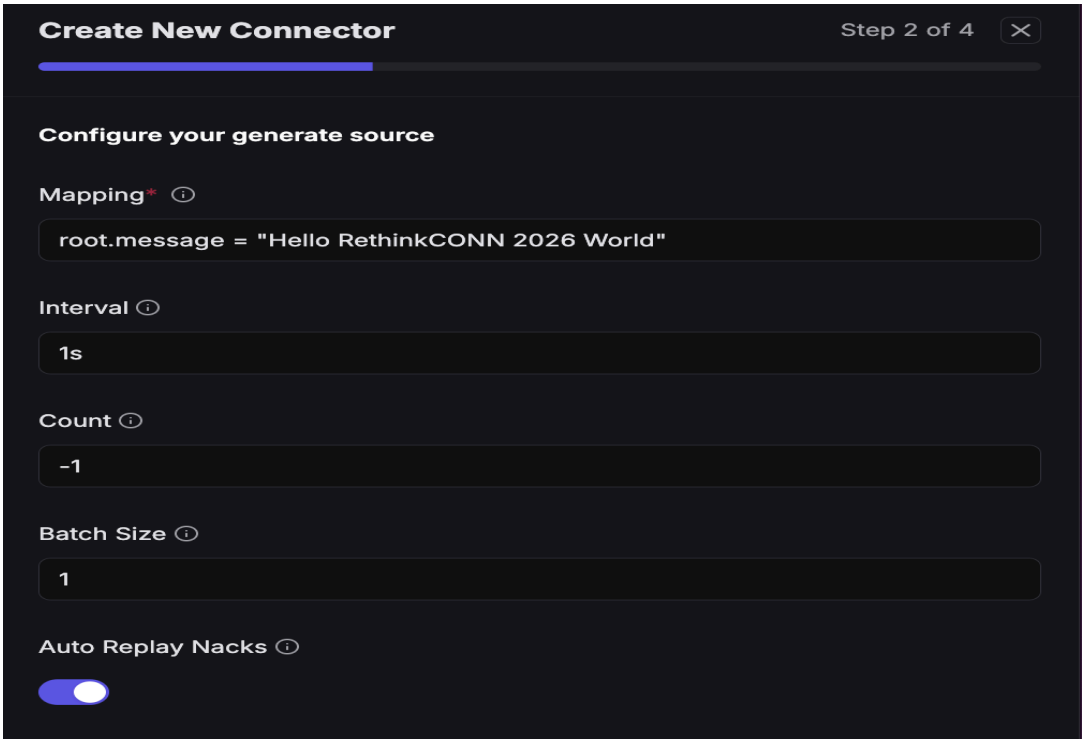


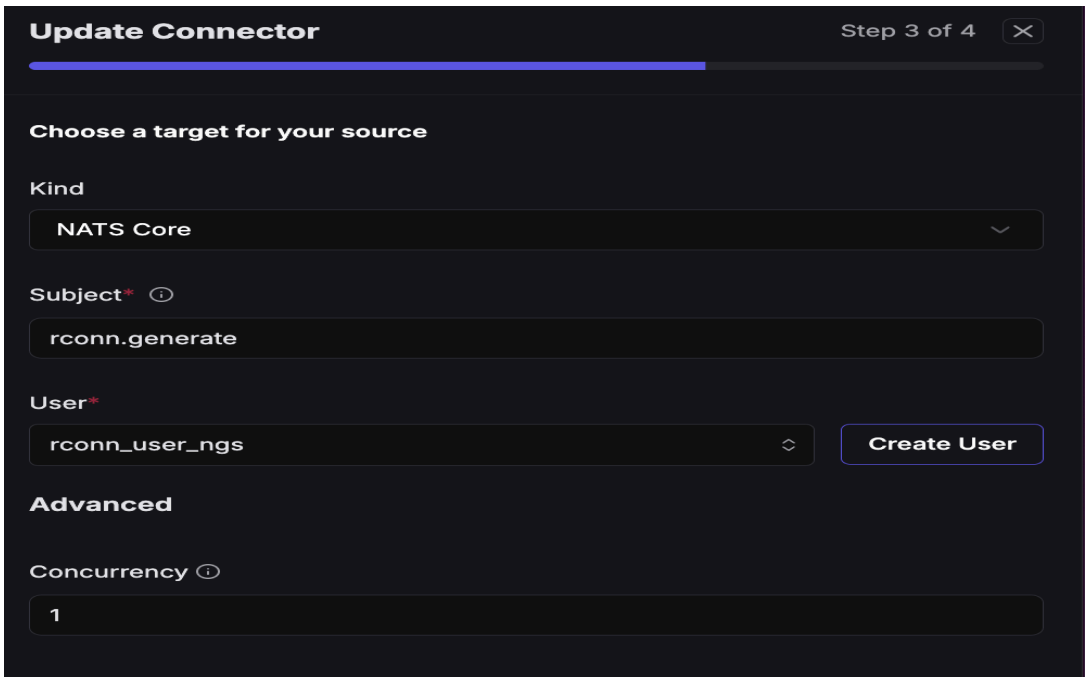
Appendix B: Connector demo

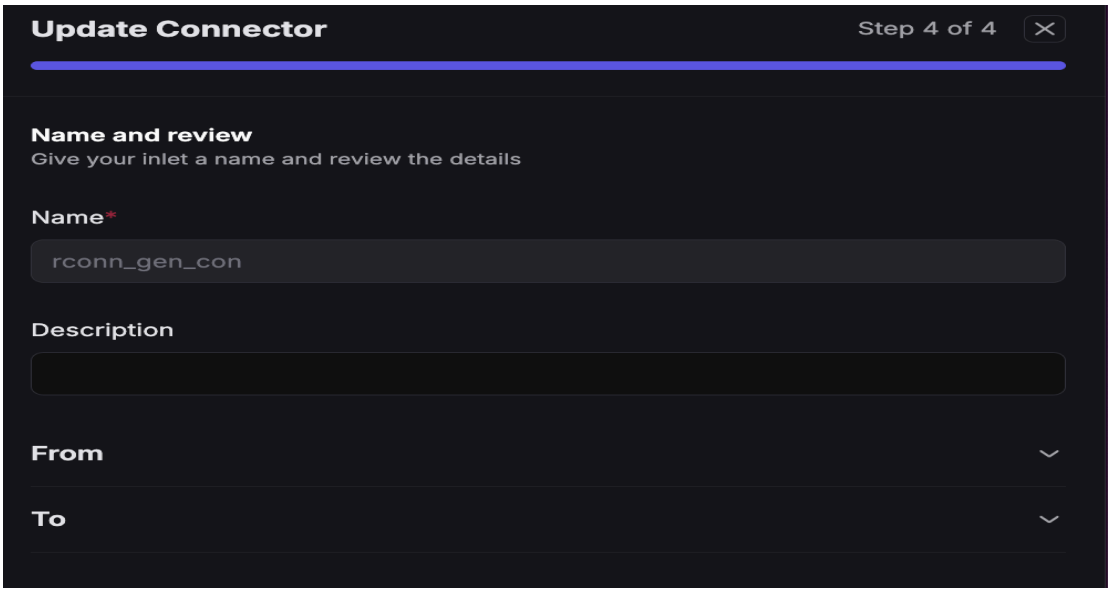
In the current Synadia Cloud platform trial account, Connectors, Workloads and registries aren’t enabled.

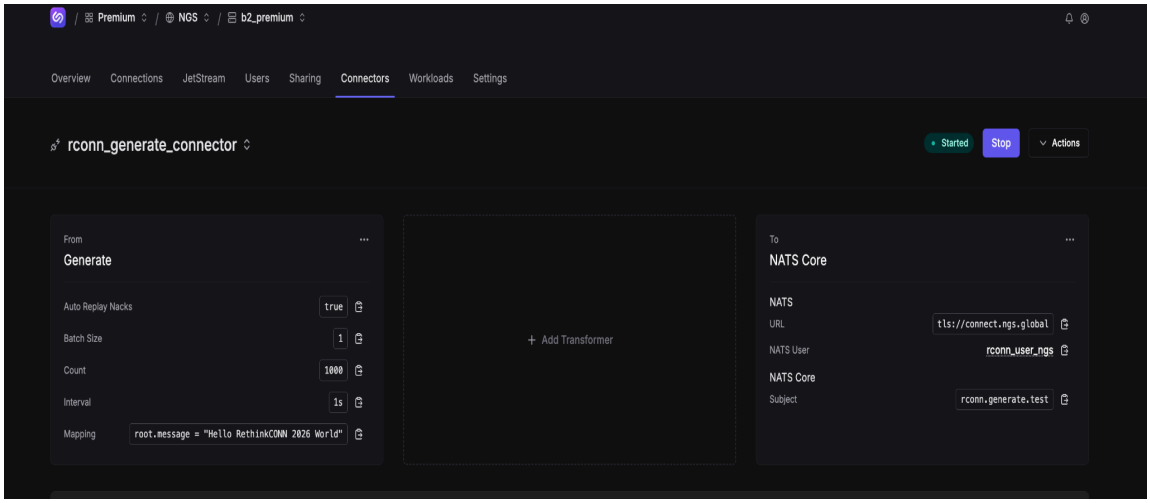
Create a “Generate” connector.




Create New Connector
Step 2 of 4
Configure your generate source
Mapping*
root.message = "Hello RethinkCONN 2026 World"
Interval
1s
Count
-1
Batch Size
1
Auto Replay Nacks


Update Connector
Step 3 of 4
Choose a target for your source
Kind
NATS Core
Subject*
rconn.generate
User*
rconn_user_ngs
Create User
Advanced
Concurrency
1





Now lets test this out, by subscribing to the topic on which the Generate connector emits the messages:

nats sub rconn.generate.* --server nats://connect.ngs.global --creds creds/NGS-b2_premium-rconn_user_ngs.creds --no-context

13:03:23 Subscribing on rconn.generate.*

[#1] Received on "rconn.generate.test"

{"message":"Hello RethinkCONN 2026 World"}

[#2] Received on "rconn.generate.test"

{"message":"Hello RethinkCONN 2026 World"}