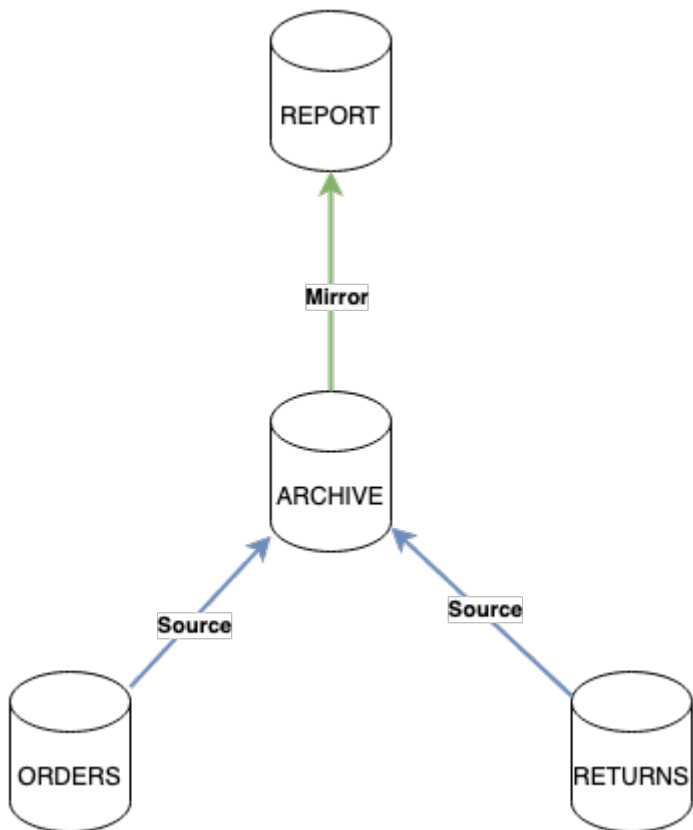


Data Replication



- [Mirrors](#)
 - [Sources](#)
 - [Configuration](#)
-

Replication allows you to move data between streams in either a 1:1 mirror style or by multiplexing multiple source streams into a new stream. In future builds this will allow data to be replicated between accounts as well, ideal for sending data from a Leafnode into a central store.



Here we have 2 main streams - *ORDERS* and *RETURNS* - these streams are clustered across 3 nodes. These Streams have short retention periods and are memory based.

We create a *ARCHIVE* stream that has 2 *sources* set, the *ARCHIVE* will pull data from the sources into itself. This stream has a very long retention period and is file based and replicated across 3 nodes. Additional messages can be added to the ARCHIVE by sending to it directly.

Finally, we create a *REPORT* stream mirrored from *ARCHIVE* that is not clustered and retains data for a month. The *REPORT* Stream does not listen for any incoming messages, it can only consume data from *ARCHIVE*.

Mirrors

A *mirror* copies data from 1 other stream, as far as possible IDs and ordering will match exactly the source. A *mirror* does not listen on a subject for any data to be added. A *mirror* can filter by subject and the Start Sequence and Start Time can be set. A stream can only have 1 *mirror* and if it is a mirror it cannot also have any *source*.

Sources

A *source* is a stream where data is copied from, one stream can have multiple sources and will read data in from them all. The stream will also listen for messages on it's own subject. We can therefore not maintain absolute ordering, but data from 1 single source will be in the correct order but mixed in with other streams. You might also find the timestamps of streams can be older and newer mixed in together as a result.

A Stream with sources may also listen on subjects, but could have no listening subject. When using the `nats` CLI to create sourced streams use `--subjects` to supply subjects to listen on.

A source can have Start Time or Start Sequence and can filter by a subject.

Configuration

The ORDERS and RETURNS streams as normal, I will not show how to create them.

```
nats s report
```

Obtaining Stream stats

Stream	Storage	Consumers	Messages	Bytes	Lost	Deleted	Cluster
ORDERS	Memory	0	0	0 B	0	0	n1-c2, n2-c2*, n3-c2
RETURNS	Memory	0	0	0 B	0	0	n1-c2*, n2-c2, n3-c2

We now add the ARCHIVE:

```
nats s add ARCHIVE --source ORDERS --source RETURNS
```

```
? Storage backend file
? Retention Policy Limits
? Discard Policy Old
? Stream Messages Limit -1
? Message size limit -1
? Maximum message age limit -1
? Maximum individual message size -1
? Duplicate tracking time window 2m0s
? Allow message Roll-ups No
? Allow message deletion Yes
? Allow purging subjects or the entire stream Yes
? Replicas 1
? Adjust source "ORDERS" start Yes
? ORDERS Source Start Sequence 0
? ORDERS Source UTC Time Stamp (YYYY:MM:DD HH:MM:SS)
? ORDERS Source Filter source by subject
? Import "ORDERS" from a different JetStream domain No
? Import "ORDERS" from a different account No
? Adjust source "RETURNS" start No
? Import "RETURNS" from a different JetStream domain No
? Import "RETURNS" from a different account No
Stream ARCHIVE was created
```

Information for Stream ARCHIVE created 2022-01-21T11:49:52-08:00

Configuration:

```
Acknowledgements: true
  Retention: File - Limits
  Replicas: 1
  Discard Policy: Old
  Duplicate Window: 2m0s
Allows Msg Delete: true
  Allows Purge: true
  Allows Rollups: false
Maximum Messages: unlimited
```

```
Maximum Bytes: unlimited
Maximum Age: unlimited
Maximum Message Size: unlimited
Maximum Consumers: unlimited
Sources: ORDERS
        RETURNS
```

State:

```
Messages: 0
Bytes: 0 B
FirstSeq: 0
LastSeq: 0
Active Consumers: 0
```

And we add the REPORT:

```
nats s add REPORT --mirror ARCHIVE
```

```
? Storage backend file
? Retention Policy Limits
? Discard Policy Old
? Stream Messages Limit -1
? Message size limit -1
? Maximum message age limit -1
? Maximum individual message size -1
? Allow message Roll-ups No
? Allow message deletion Yes
? Allow purging subjects or the entire stream Yes
? Replicas 1
? Adjust mirror start No
? Import mirror from a different JetStream domain No
? Import mirror from a different account No
Stream REPORT was created
```

Information for Stream REPORT created 2022-01-21T11:50:55-08:00

Configuration:

```
Acknowledgements: true
  Retention: File - Limits
    Replicas: 1
  Discard Policy: Old
Duplicate Window: 2m0s
Allows Msg Delete: true
  Allows Purge: true
  Allows Rollups: false
Maximum Messages: unlimited
  Maximum Bytes: unlimited
    Maximum Age: unlimited
Maximum Message Size: unlimited
Maximum Consumers: unlimited
  Mirror: ARCHIVE
```

State:

```
  Messages: 0
    Bytes: 0 B
  FirstSeq: 0
    LastSeq: 0
Active Consumers: 0
```

When configured we'll see some additional information in a `nats stream info` output:

```
nats stream info ARCHIVE
```

Output extract

```
...
Source Information:

  Stream Name: ORDERS
    Lag: 0
  Last Seen: 2m23s

  Stream Name: RETURNS
    Lag: 0
  Last Seen: 2m15s
```

```

...

$ nats stream info REPORT
...
Mirror Information:

    Stream Name: ARCHIVE
        Lag: 0
    Last Seen: 2m35s
...

```

Here the `Lag` is how far behind we were reported as being last time we saw a message.

We can confirm all our setup using a `nats stream report`:

```
nats s report
```

```

+-----+
+-----+
|                                     Stream
Report                                     |
+-----+-----+-----+-----+-----+-----+-----+
+-----+
| Stream | Storage | Replication | Consumers | Messages | Bytes | Lost | Deleted |
Cluster          |
+-----+-----+-----+-----+-----+-----+-----+
+-----+
| ARCHIVE | File     | Sourced     | 1         | 0         | 0 B    | 0     | 0         | n1-c2*,
n2-c2, n3-c2 |
| ORDERS  | Memory   |              | 1         | 0         | 0 B    | 0     | 0         | n1-c2, n2-
c2*, n3-c2 |
| REPORT  | File     | Mirror      | 0         | 0         | 0 B    | 0     | 0         | n1-
c2*         |
| RETURNS | Memory   |              | 1         | 0         | 0 B    | 0     | 0         | n1-c2, n2-
c2, n3-c2* |
+-----+-----+-----+-----+-----+-----+-----+
+-----+

+-----+
|                                     Replication Report                                     |

```

Stream	Kind	Source Stream	Active	Lag	Error
ARCHIVE	Source	ORDERS	never	0	
ARCHIVE	Source	RETURNS	never	0	
REPORT	Mirror	ARCHIVE	never	0	

We then create some data in both ORDERS and RETURNS:

```
nats req ORDERS.new "ORDER {{Count}}" --count 100
nats req RETURNS.new "RETURN {{Count}}" --count 100
```

We can now see from a Stream Report that the data has been replicated:

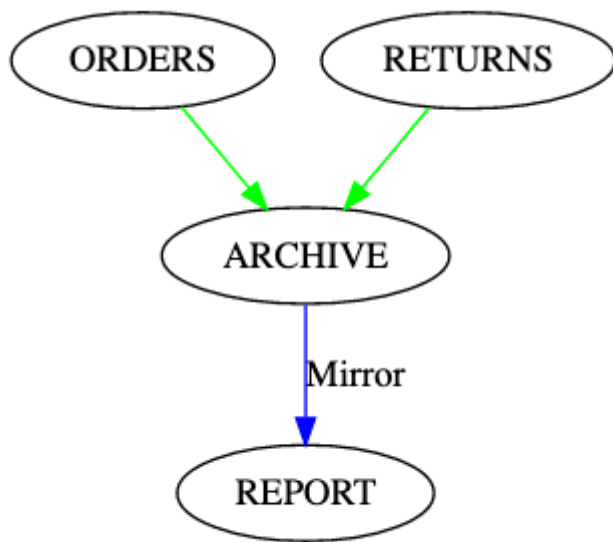
```
nats s report --dot replication.dot
```

Obtaining Stream stats

Stream	Storage	Consumers	Messages	Bytes	Lost	Deleted	Cluster
ORDERS	Memory	1	100	3.3 KiB	0	0	n1-c2, n2-c2*, n3-c2
RETURNS	Memory	1	100	3.5 KiB	0	0	n1-c2*, n2-c2, n3-c2
ARCHIVE	File	1	200	27 KiB	0	0	n1-c2, n2-c2, n3-c2*
REPORT	File	0	200	27 KiB	0	0	n1-c2*

Replication Report						
Stream	Kind	Source Stream	Active	Lag	Error	
ARCHIVE	Source	ORDERS	14.48s	0		
ARCHIVE	Source	RETURNS	9.83s	0		
REPORT	Mirror	ARCHIVE	9.82s	0		

Here we also pass the `--dot replication.dot` argument that writes a GraphViz format map of the replication setup.



Stream Replication Structure

#2

□ cocobook-sync ## 2026-10-28 11:14:40 UTC

□ cocobook-sync ## 2026-10-29 03:58:10 UTC