

FAQ

- [Why am I seeing "java.io.IOException: Broken pipe" exceptions in my logs?](#)
- [I see "Unable to load realm info from SCDynamicStore" in my logs. Why?](#)
- [Why is my job processing old messages?](#)
- [How should Samza be run on AWS?](#)
- [How should I use/configure RocksDB stores in my Samza application?](#)

Why am I seeing "java.io.IOException: Broken pipe" exceptions in my logs?

If you are using a Kafka system, this exception can be caused by a number of issues.

1. Using a VIP/load balancer, they occasionally close idle connections.
2. You have systems.<your system>.producer.request.required.acks undefined, or set to 0.

In the second case, not defining your acks setting, Kafka defaults to an acknowledgment level of zero. If the Kafka broker has an issue (such as `kafka.common.MessageSizeTooLargeException`), the only way that the broker can communicate with your producer is to close the connection, since your producer is not listening to acknowledgements from the broker. If you set your producer's acks setting to be non-zero, you'll then be able to see the real issue that's causing the Kafka broker to close your connection.

For more details, have a look at Kafka's [producer configuration page](#).

I see "Unable to load realm info from SCDynamicStore" in my logs. Why?

This is a Java 6 bug that has no real impact to Samza. It does appear in the logs in certain circumstances. More details are available on [HADOOP-7489](#).

Why is my job processing old messages?

There are several ways in which this can occur:

1. Your job has no checkpoint, and is configured to start reading from the beginning of a topic.
2. Your job has a checkpoint, but is configured to disregard it and start reading from the beginning of a topic.
3. The Kafka consumer for your job gets an `OffsetNotFound` exception, and begins reading from the beginning of a topic.

There are several important configurations:

```
systems.<system>.streams.<your stream>.samza.reset.offset=true
systems.<system>.streams.<your stream>.samza.offset.default=oldest
systems.<system>.consumer.auto.offset.reset=smallest
```

The `samza.reset.offset` configuration tells Samza whether to pay attention to checkpoint messages, which store the last offset you read from before the container stopped. When your container starts up again, it normally picks up where it left off in the stream. This setting tells the container not to pick up where it left off.

The `samza.offset.default` setting tells the container what to do when there's no checkpoint available (or it's been ignored because of `samza.reset.offset`). If you say "oldest", Samza will start reading from the OLDEST message in the topic. If you say "upcoming", Samza will start reading from the newest message in the topic.

The `consumer.auto.offset` configuration tells the Kafka consumer what to do in cases where the consumer has fallen off the edge of the topic: it has an offset that's either too old, or too new for the topic it's trying to read from. If set to smallest, the Kafka consumer will start reading from the oldest message. If set to largest, it'll start reading from the newest message in the topic.

Note that the first two configurations also have system-level settings (i.e. `systems.<your system>.samza.reset.offset` and `systems.<your system>.samza.offset.default`).

How should Samza be run on AWS?

From Gian Merlino:

We've been using Samza in production on AWS for a little over a month. We're just using the YARN runner on a mostly stock hadoop 2.4.0 cluster (not EMR). Our experience is that c3s work well for the YARN instances and i2s work well for the Kafka instances. Things have been pretty solid with that setup.

For scaling up and scaling down YARN, we just terminate instances or add instances, and this works pretty well. It can take a few minutes for the cluster to realize a node has gone and respawn containers elsewhere.

We have a separate Kafka cluster just for Samza's use, different from our main Kafka cluster. The main reason is that we wanted to isolate off the disk and network load of state compactions and restores (we don't use compacted topics in our main Kafka cluster, but we do use them with Samza, and the extra load on Kafka can be substantial).

How should I use/configure RocksDB stores in my Samza application?

Please follow the how-to guide [here](#).