Hands-on activity: consumer groups

Introduction

Description

In this exercise, we'll take a look at some example Python code that demonstrates ways of working with consumer groups. We'll use the natural numbers stream example as before, keeping the same producer code. This time though, we'll organize the consumers into consumer groups so that they can work together.

Code

The code is all contained in a single Python file **consumer_group.py**. It creates the following sub-processes:

- A producer, which pushes increasing natural numbers into a stream at semi-random intervals.
- A consumer group containing three consumers, each of which reads messages from the stream and calculates whether or not the number in the message is prime.
- A chaos process which periodically randomly terminates one of the consumers and restarts it, forcing it to recover and resume its former workload as it restarts.

Before each run, the code will reset the stream. This example code uses the Python Redis Client

to connect to Redis and issue commands.

This exercise assumes that you have cloned the course GitHub repository and followed the setup instructions. Note that you will need to activate your Python virtual environment and set any required environment variables for each new terminal session that you use here.

← Previous Next →

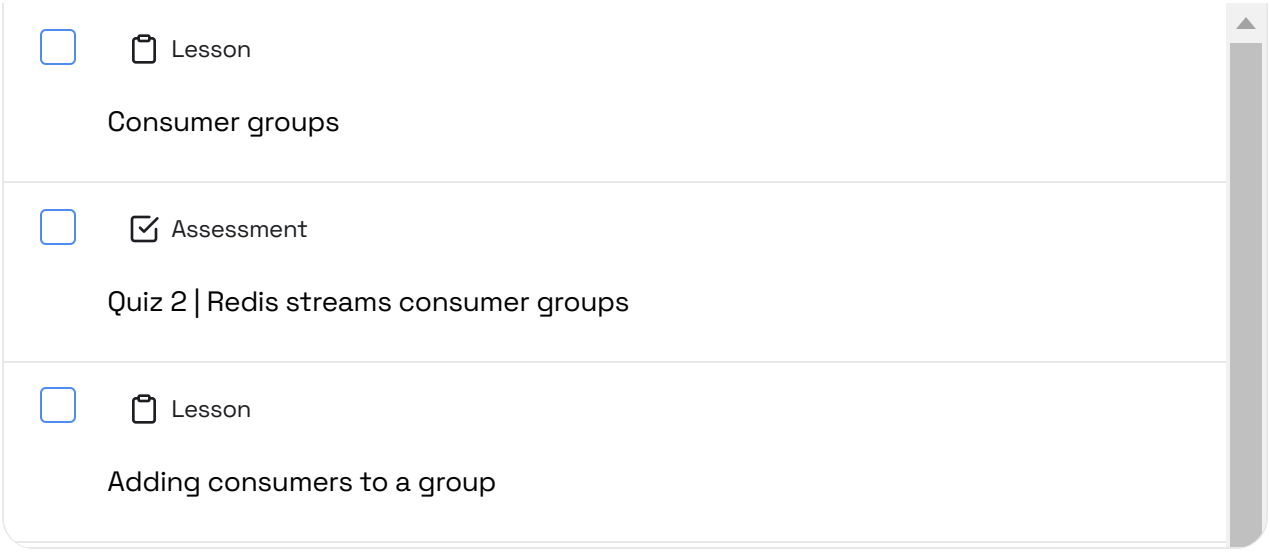
Modules

Course overview

- Lesson: Course overview
- Lesson: Environment setup

Consumer groups

- Lesson: The problem with slow consumers
- Assessment: Quiz 1 | Redis streams consumer groups


Lesson
Consumer groups
Assessment
Quiz 2 | Redis streams consumer groups
Lesson
Adding consumers to a group