Test idea

What we want to measure

Latency of the requests - Test how long it takes to get a response from the same MS using diff SMTs.
Resource Usage - Test how many resources these SMTs consume.

What cases we want to measure

Single user - sensibly spaced requests - We can test how well the SMTs handle a single user sending appropriately timed requests (Most relaxed usage)
Single user - Lots of continuous requests - We test how well the SMTs handle a single user spamming the services (WRK2 won't send next request until receiving response to this one but we can measure accumulated latency)
Multiple users - sensibly spaced requests - We can test how well the SMTs handle multiple users sending appropriately timed requests (Stress Test a bit)
Multiple users - Lots of continuous requests - We test how well the SMTs handle multiple users spamming the services (Stress Test a lot)

Experiment Setup

Deploy a number X of Count Microservices to Cluster.
MicroService URL are set as follows:
"http://MicroCounterService-{ServiceNumber}/count"
ServiceNumber ranges from 0..X-1

The Count Microservices take a request which includes the total number of microservices in the cluster (X), a current counter value (C) and an objective counter value (Y).

When the Count Microservice receives Request(X, C, Y):

```
// It increases the counter value by 1
C += 1
//Checks if we've reached the desired counter value
if C == Y:
    // If reached it sends a confirmation
    return "Counter Reached"
else:
    // We calculate what the next microservice would be (Assume C always started at 0)
    ServiceNumber = C
    while ServiceNumber > X:
        ServiceNumber = ServiceNumber - X
    MicroUrl = "http://MicroCounterService-{ServiceNumber}/count"
    // We send a request with the updated Counter to the next MicroService, and return its response
    return SendRequest(MicroUrl, R(X, C, Y))
```

When sending a request to this service, e.g. R(3, 0, 20), a request will loop between 3 Counter MicroServices, increasing once each time it reaches a MS until its value is 20. Then a response will follow the same path backwards until it reaches the original sender.

There will be a pod that sends the original request e.g. R(5, 0, 30).
This pod will time the amount of time that it takes to get a request back.
The code that sends these requests inside the pods can be called from multiple threads.
This way we can also stress test the network with multiple users (Z).

Why Use this Structure

We can test with an unlimited number of microservices
We can stress test the network with multiple users
By selecting a large counter value, we will chain a lot of requests, making differences more noticeable

How to measure the desired data

Latency:

- WRK2 is a project that submits requests and times how long it takes to get them back
- Create my own timer sender process (Preferably not, but it is an option)

Resource Usage:

- Prometheus and Grafana can be used to view resource consumption
- Not sure yet how to record it, but there should be a way to do it