**Rust multi module microservices Part 8 — The conclusion**

[[](https://medium.com/@omprakashsridharan?source=post_page-----b670f7b71308--------------------------------)](https://medium.com/@omprakashsridharan?source=post_page-----b670f7b71308--------------------------------)

[Omprakash Sridharan](https://medium.com/@omprakashsridharan?source=post_page-----b670f7b71308--------------------------------)

·

Follow

3 min read

·

Jun 27

9

The entire code that we wrote until now is available for reference [here](https://github.com/omprakashsridharan/rust-superapp). Let’s run our code !

Terminal 1

docker compose up

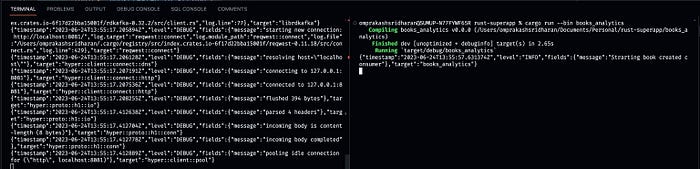
Please wait for the containers to start ( ~20 seconds ) and then run the below commands.

Terminal 2

cargo run --bin books\_api

Terminal 3

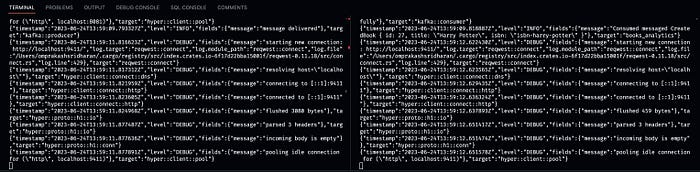
cargo run --bin books\_analytics



You should see both the services starting with some logs. Now let’s hit the api with request to create a book using curl.

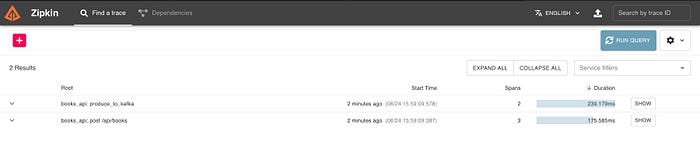
curl --location --request POST 'localhost:8080/api/books' \ --header 'Content-Type: application/json' \ --data-raw '{ "title":"Harry Potter", "isbn": "isbn-harry-potter" }'

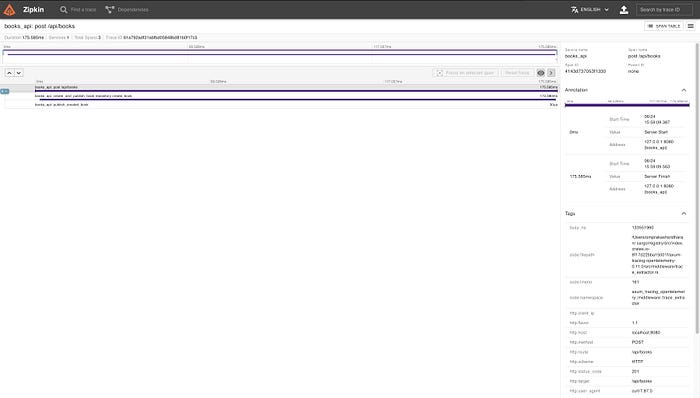


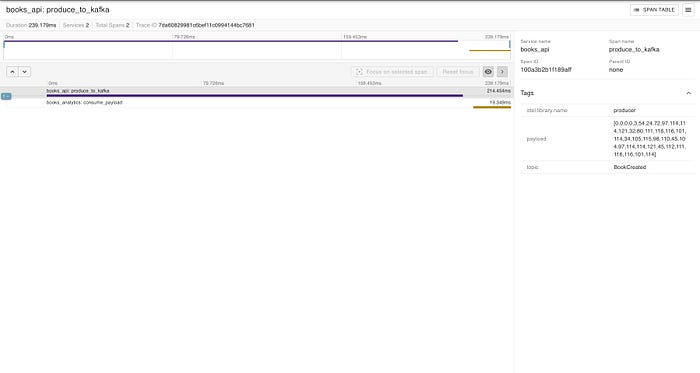


As you can see in the above screenshot right terminal, where the books\_analytics has consumed the message and printed it and now let’s look at the tracing.

In your browser go to [localhost:9411](http://localhost:9411/) and you should see the Zipkin UI. Click on run query to refresh and see two new traces. One for the HTTP API Call and another is to trace the Kafka producer and consumer.







You can also examine the Kafka cluster using the Kafka UI service in our Docker Compose, allowing you to inspect schemas, topics, and much more. With this, we now have a working setup of a Rust multi-module Cargo workspace project, which includes reusable modules for creating microservices and shared libraries. This project, in its current state, is not entirely production-ready, but with a bit of cleanup, such as config abstraction, improved logging, and file structuring, it should become highly efficient and performant, thanks to Rust ❤️

If you made it this far, really appreciate your time & patience. Please feel free to contribute to the repository and happy to hear any feedback/suggestions/improvements. Enjoy your Rust learning journey and hope you had fun as I did doing and writing this article. If you like this series and wanted to support me, you can do so by buying me a ☕️ [**here**](https://www.buymeacoffee.com/omprakashop). Have a nice day and a year ahead 👋