**Saga Orchestration Example DEMO**

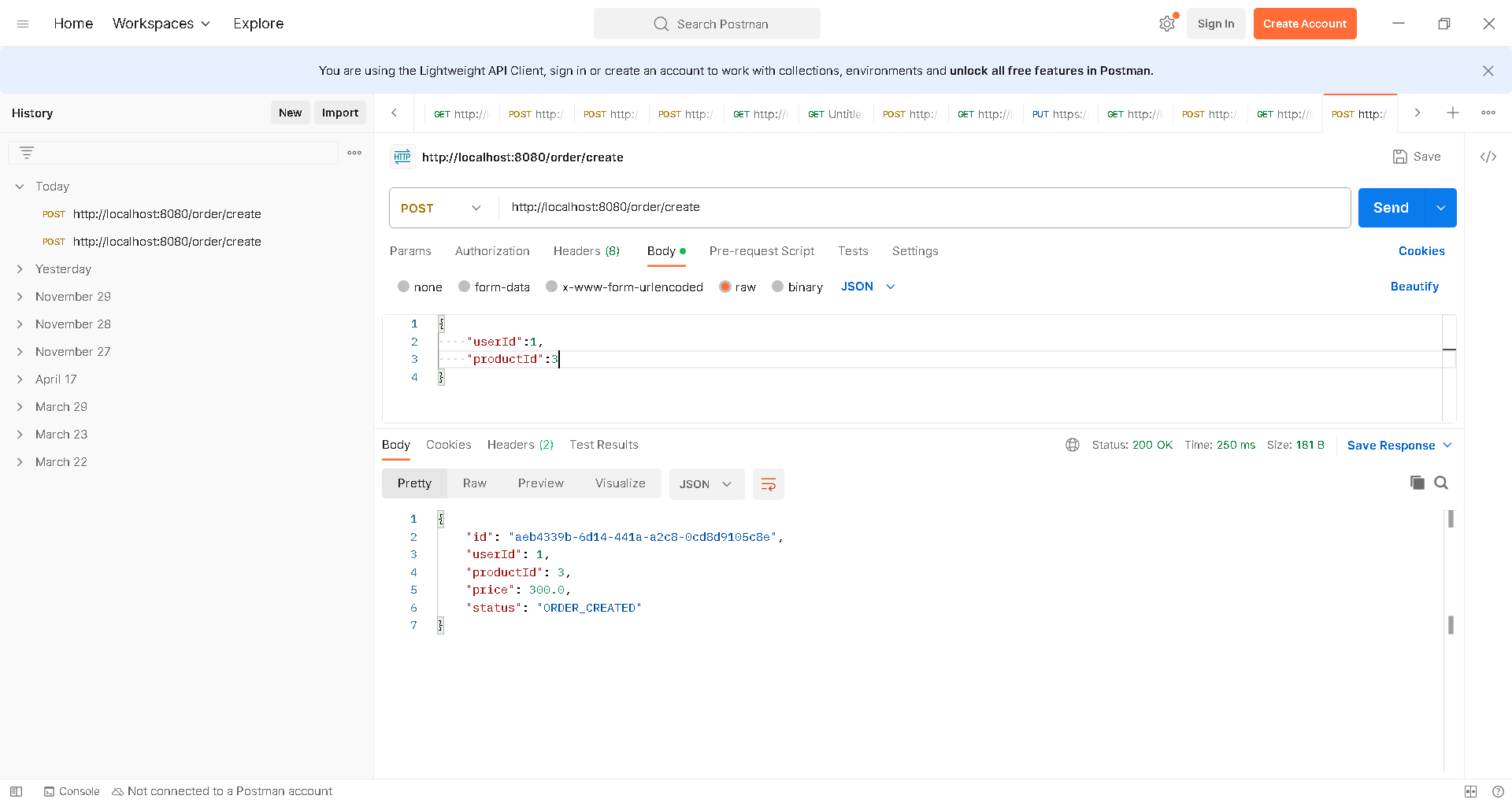
Run below command

docker compose up

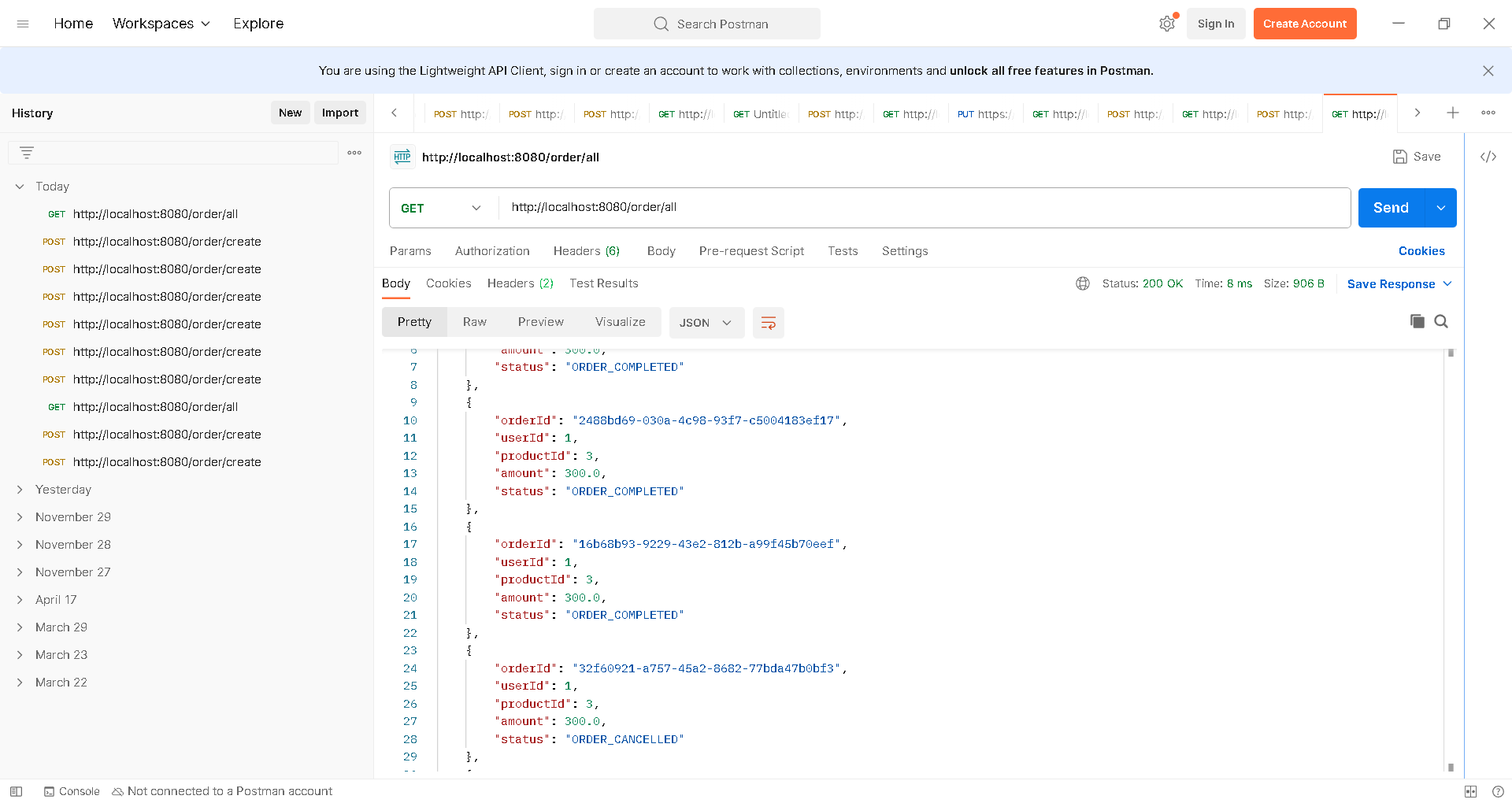
Once all the services are up and running, I send a POST request to create order. I get the order created status.

Do note that user 1 tries to order product id 3 which costs $300

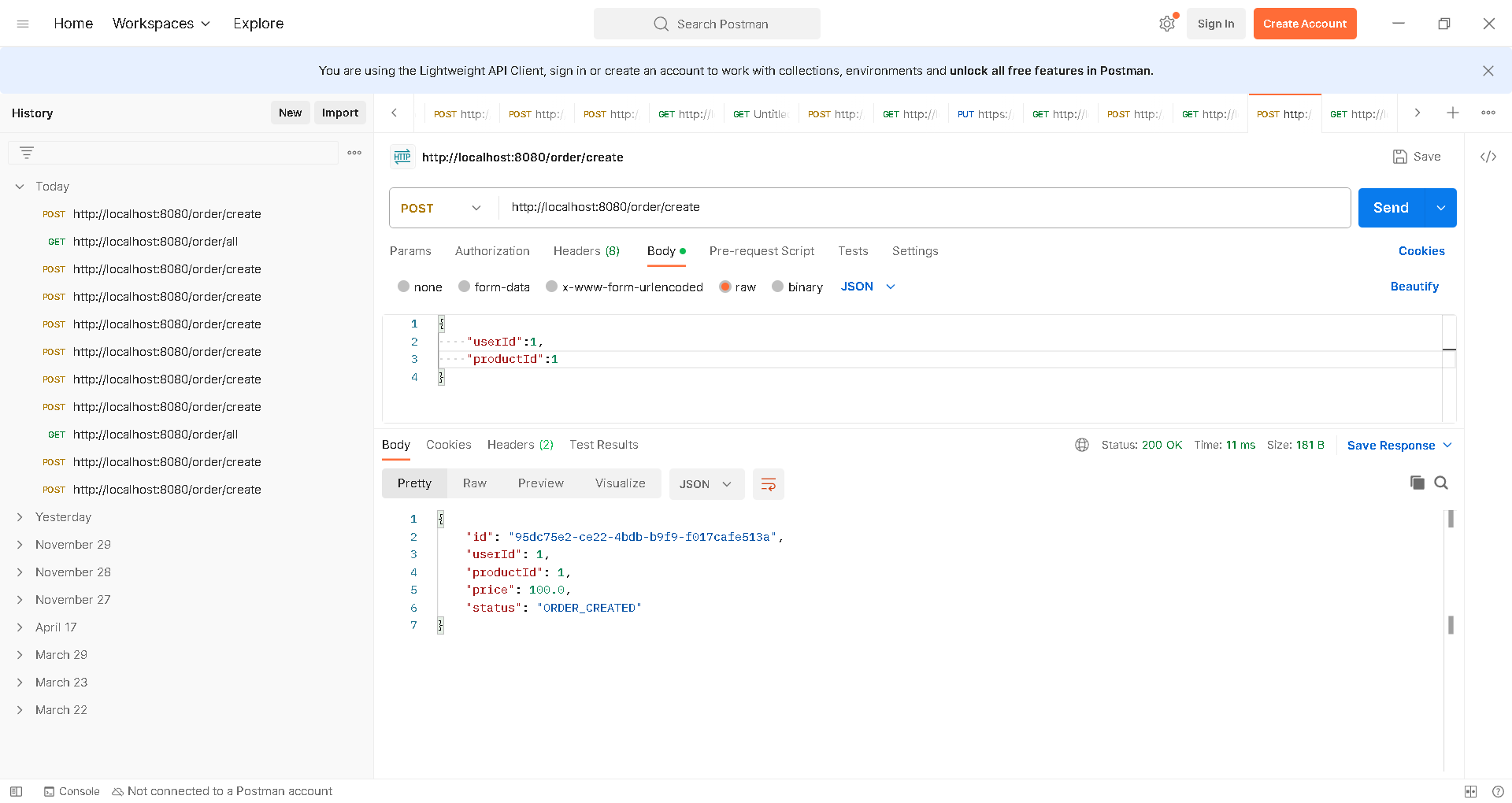
The user’s credit limit is $1000

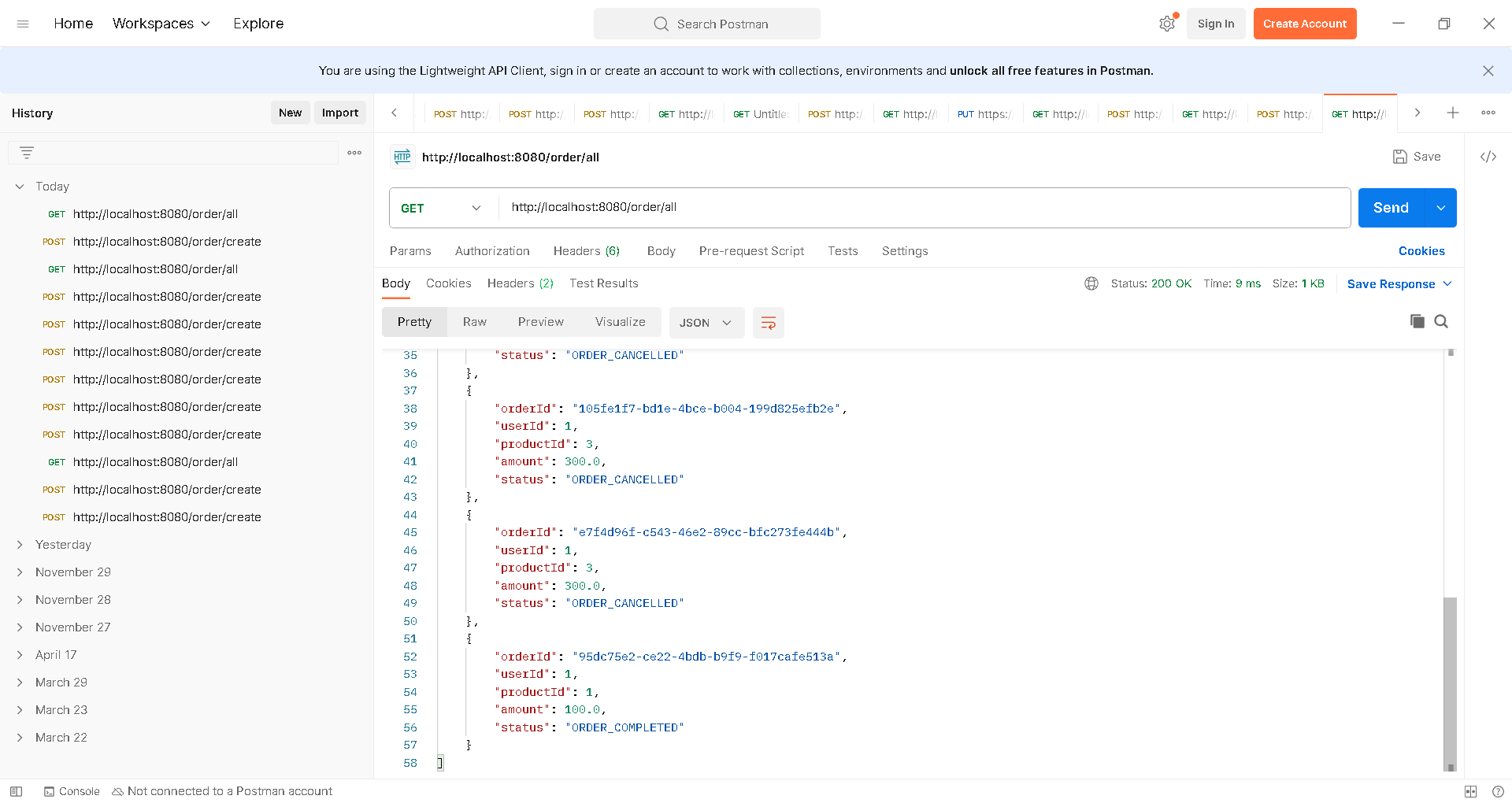


I sent 4 requests. So 3 requests were fulfilled. Not the 4th one as the user would have only $100 left and we can not fulfill the 4th order. So the payment service would have declined.



The user 1 with this available balance $100, he can buy product id 1 as it costs only $100.





**Saga Choreography Example DEMO**

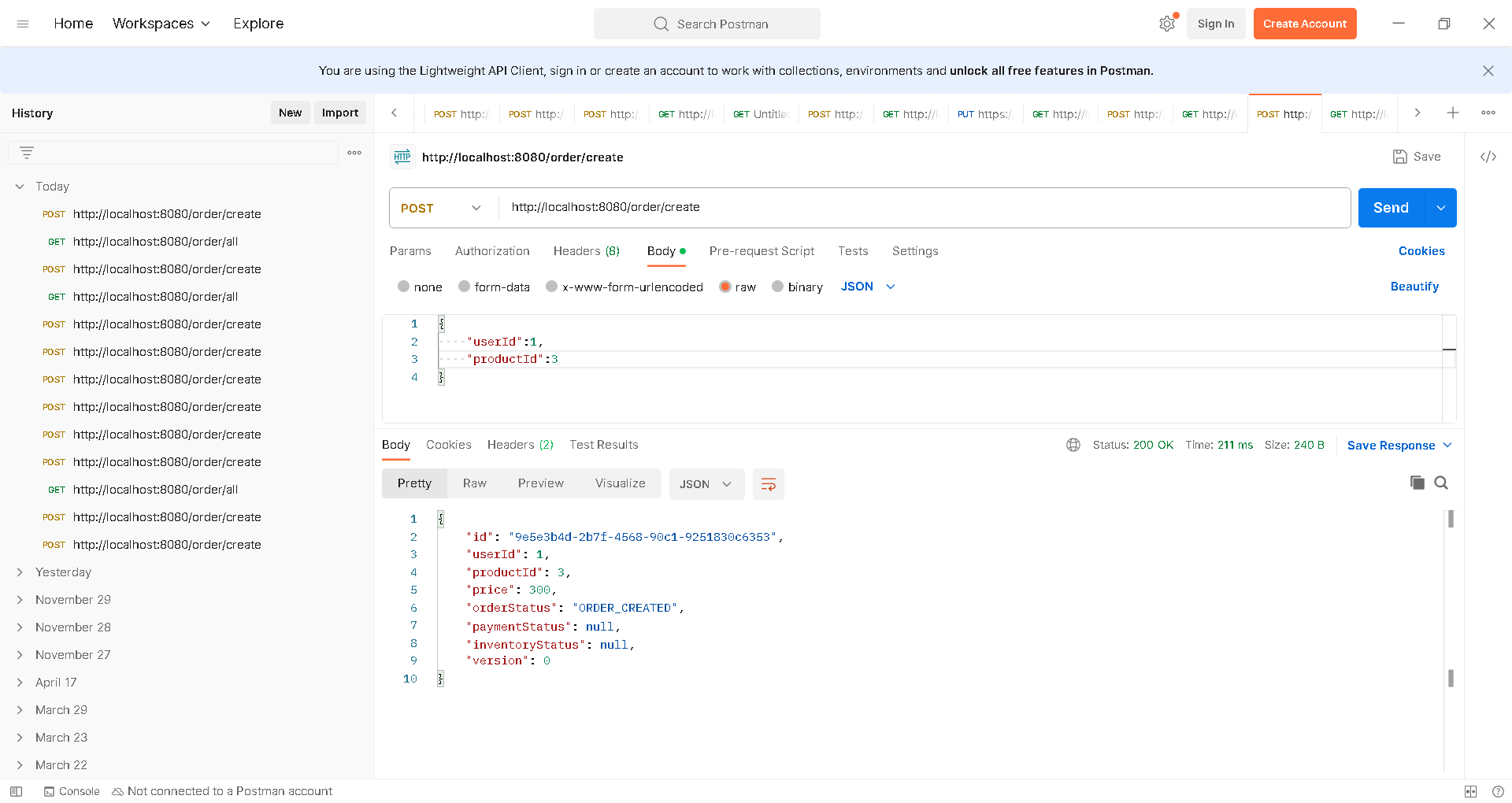
Run below command

docker compose up

Once I start my application, I send an order request. The productId=3 costs $300 and user’s credit limit is $1000.

As soon as I send a request, I get the immediate response saying the order\_created / order\_pending.

I send 4 requests.



If I send /order/all requests to see all the orders, I see that 3 orders in fulfilled and the 4th order in cancelled status as the user does not have enough balance to fulfill the 4th order.

We can add additional services in the same way. For ex: once order-service fulfills the order and raises an another event. A shipping service listening to the event and takes care of packing and shipping to the given user address. Order-service might again listen to those events updates its DB to the order\_shipped status.

As we had mentioned earlier, committing/rolling back a transaction which spans multiple microservices is very challenging. Each service should have the event-handlers, for committing / rolling back the transaction to maintain the data consistency, for every event it is listening to!

