Bipin Kumar

Shael.dhn88@gmail.com

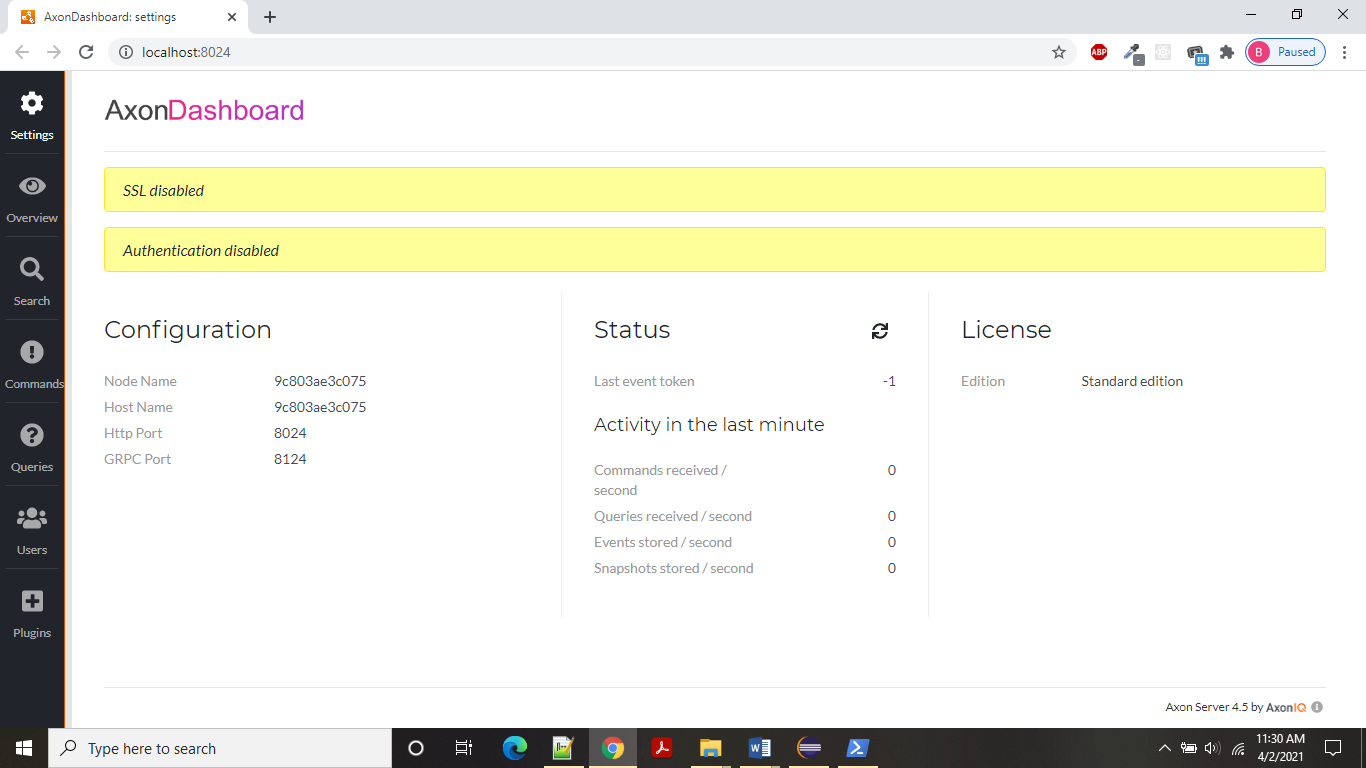
Saga

SAGA Doc

**Docker image for Axon server**:

docker run -d --name axonserver -p 8024:8024 -p 8124:8124 axoniq/axonserver

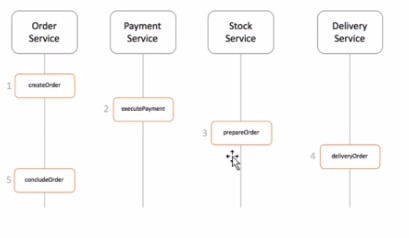
It will start axon server on <http://localhost:8024>



In case of Monolith application, suppose we have service S1, S2, S3 and S4. To complete a transaction, we call S1 and S1 calls other services, transaction get completed when S1 get response from S4 or if any issue occurred in middle of execution we will roll back the transaction. In this case we are using same DB.

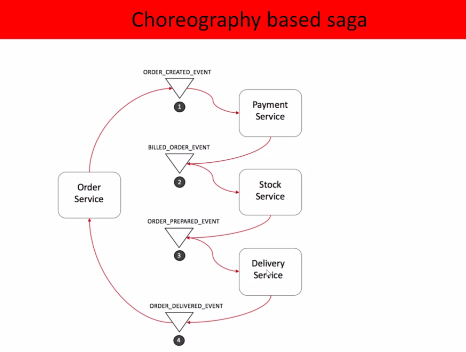
But in case of microservice we have different microservices which execute the task and we can’t open a transaction from one microservice and it will go to other microservice because all service has its own DB. So, in case of microservice to achieve transaction we can use a pattern called SAGA pattern.

* A Saga is sequence of local transaction.
* Each local transaction updates the DB and publishes a message or event to trigger the next local transaction in the saga.
* If a local transaction falls because it violates a business rule then the saga executes a series of compensating transactions that undo the change that were made by the preceding local transactions.

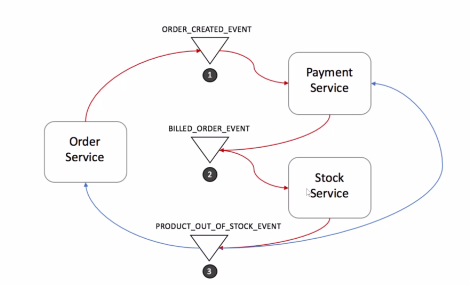


There are two type of Saga:

* Choreography: Each local transaction publishes domain events that trigger local transactions in other services.

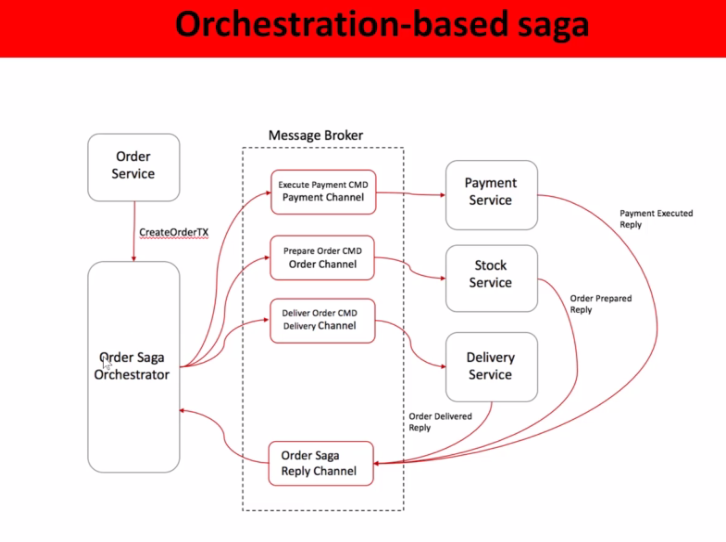
****

Failure scenario:

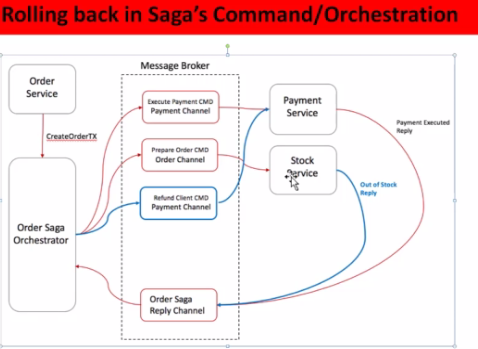


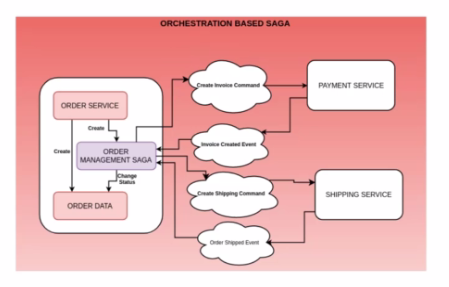
It is suggested to used in case of small microservices and there is less no. of events is there. But in this case we can implement new microservice without much efforts or can say without changing other services.

* Orchestration: An orchestration (Object) tells the participants what local transaction to execute.



Failure case:





In above diagram we have orchestration based saga. Here when order service gets a request to create it call order management saga, I will create createordercommand and then it will call other service using order management saga which will call payment service to generate invoice, It will create invoice and create invoice created event and saga will listen that event and then it will crate shipping command and send it to shipping service. After successful shipment it will generate order shipped event and send it to saga where it checked the status of shipment and based on it complete the transaction.