Service

This time you will need 2 logically bound microservices. For example, a service that handles customer orders and another that manages and keeps track of stock. In order to create an order for the customer, the stock will have to be checked and updated accordingly.

- Long-running saga transaction (https://developers.redhat.com/blog/2018/10/01/patterns-for-distributed-transactions-within-a-microservices-architecture).
- Database redundancy/replication + failover (https://searchdatamanagement.techtarget.com/definition/database-replication). Make various instances of services connect to different DB replicas.
- Set up ELK stack (https://www.elastic.co/what-is/elk-stack) **OR** Prometheurs (https://prometheus.io/) + Grafana for logging (https://grafana.com)

Gateway

- Service high availability (https://en.wikipedia.org/wiki/High_availability). If a request to a service fails or the service is otherwise unavailable, route the request to a different one. **Bonus:** Circuit breaker should still be tripped should multiple such cases occur.
- **Bonus:** perform a microservice based 2 phase commit including a distributed transaction (https://developers.redhat.com/blog/2018/10/01/patterns-for-distributed-transactions-within-a-microservices-architecture/)
- Cache high availability. Similar to services.
- Set up ELK stack (https://www.elastic.co/what-is/elk-stack) **OR** Prometheurs (https://prometheus.io/) + Grafana for logging (https://grafana.com)

Cache (only if 4 members)

- Set up cache replication. (https://redis.io/topics/replication)
- Make the cache distributed using consistency hashing (https://www.toptal.com/big-data/consistent-hashing, https://blog.baowebdev.com/2019/04/distributing-a-cache/)