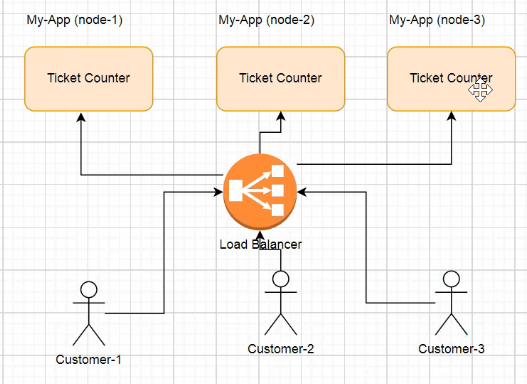
Load balancer

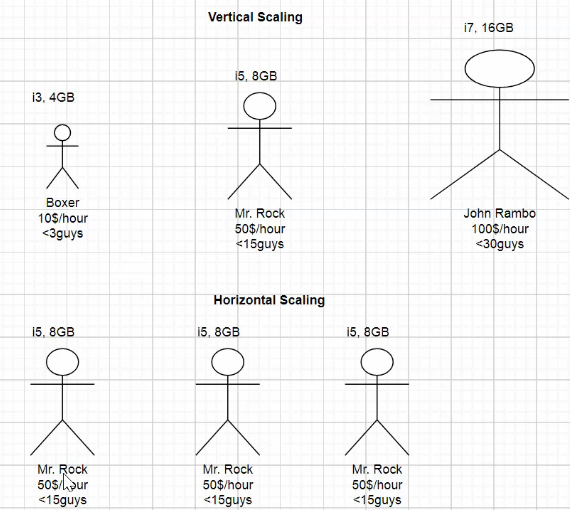


If u go via load balancer, when any request comes it will 1st goto load balancerand it will see whether node-3 is down or not,

If it is down then requests will not be routed to the svc /instance which is down

So always have a load balancer

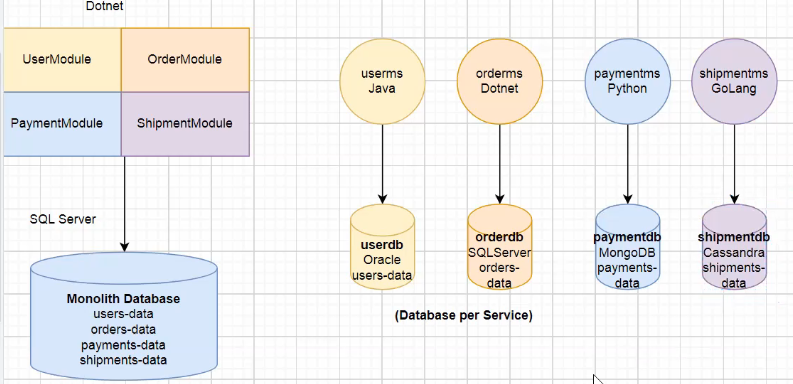
Horizantal vs vertical scaling



Horizontal scaling means , using same 3..n machines like hiring many guys for each technology

Vertical scaling means increasing ram on same computer, like 1 guy becoming full stack developer

## When to go for micro services



If we have independent functionality then only we can keep it as separate,

Ex:- loans app, money transfer both are independent so if these are in same app, then if we have any change in any apart, entire app must be down

But only if incase of independent functionality, we have to separate, we should follow DDD – domain driven design

Case 1:- uber

Uber has many modules like –users, Bill, payment, notifications, Driver, GPS,

All rest api’s must contact via api gateway

## Features of micro services

1. Each service has specific design
2. All svcs should interact with each other using REST, Message based communication

### Advantages of micro services

1. Faster development time

Agile means ability to move quickly ,fastly

Agile means reacting to the change, adaptive

Agile == faster, If u follow monolithic architecture, u cant be agile , u cant be faster

1. Scaling Horizontally or vertically a small ms component is easy because it requires less ram

If it is monolithic, to horizontally scale that it needs a separate computer with huge RAM

1. Ex:- loans, credit card issuing both are independent functionality so split

In case of independent functionality if we split into 2 apps, because of change in 1 place, we don’t need to shutdown entire application, just shutdown 1 small ms alone

1. If u split u can code each appn in any kind of language, java ,.net, python

### Cons of micro services

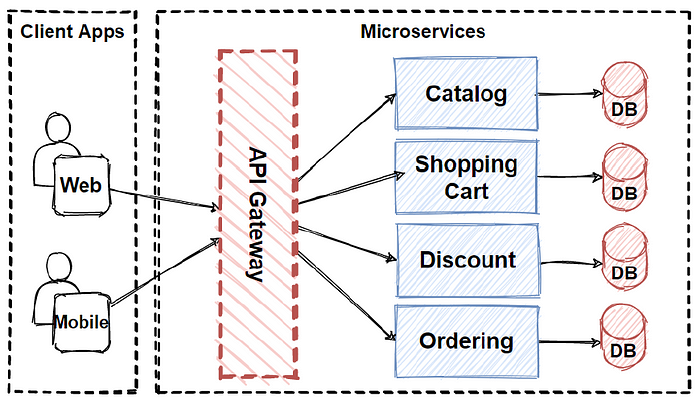
1. Inter service communication-(latency, failures)when app are decomposed, If 1 app wants to talk to another application, we should use http REST calls, there is a chance another app can be down and bec of this latency will be there, incase of monolithic we just need to create object of that class and call that method
2. Transaction management – apply global transaction management on 2-diff dbs is difficult
3. Taking 1 db per ms is not at all recommended –

## Design patterns in micro services

### **Aggregator design pattern**

In microservices the Aggregator Design Pattern is a service that receives a request, then that service makes requests of multiple services, combines the results and responds to the initiating request.

### API gateway pattern



It is similar to the **facade pattern**of Object-Oriented Design, so it provides a **single entry point**to the APIs with encapsulating the underlying system architecture.

Here the api gateway is in between client and internal microservices, It routes the requests ,

It provides authentication, SSL termination, cache..

Chained pattern provides a single output, which is a combination of multiple chained outputs.

Event sourcing pattern – helps in tracking changes

Branch pattern – this pattern can simultaneously process the requests from 2 or more services