What is scaling in Microservices

Description

Introduction

One thing to keep in mind is failures can or will happen no matter you buy the best kit, the most expensive hardware, but you cannot avoid from the fact that things can and will fail. We have to work with the assumption that things will fail. We need to handle the failure of a service gracefully.

Keys factor for Scaling

Before you think about scaling we must consider what are the factors to decide for scaling.

Response time of your application.

We need to always check what load is acceptable for our application and how long does it take for the application to respond i.e. response time. We can also check the number of sessions active or may be the number of concurrent connections/users you will expect your software to handle.

Availability

We need to know how critical our application is and we need to ensure the availability of the application 24/7. But we also need to make sure our app is up and running all the time.

CPU Utilization

We need to check how much CPU is utilized by a particular microservice.

Memory

Memory is also one of the important factors we need to consider. This is one of the important factors

because you know what is the best memory we can allocate, less is always problematic as your app might keep crashing.
Now that we know that we should scale our services if we are worried about failures or performance. Lets see what are the types of scaling

Types of Scaling

Vertical scaling

- You will have to buy a bigger machine. ðŸ™‚,
- The request processing is going to be faster as we have inter process communication.
- Load balancing [hyperlink here] is not required here as there is only a single machine.
- Since we have only one machine and if that fails that we are in trouble. We can also call this a single point of failure.

Horizontal Scaling

- You will have to buy more machines.
- The request processing is going to be through the network, hence a bit slower compared to inter process communication.
- Horizontal scaling is resilient i.e. Since we have more machines, the request can land up to any one of the machines.
- Load balancing would be required as we have more than one machine.
- If one of the machines fails, we have another machine to handle the requests.

Category

1. Design
2. Microservices

Date Created
March 29, 2021
Author
kk-ravi144gmail-com