

[Open in app](#)

★ Member-only story

# Difference between Horizontal Scalability vs Vertical Scaling in System Design and Microservices

[Soma](#) · [Follow](#)

Published in Javarevisited

8 min read · Feb 26



Listen

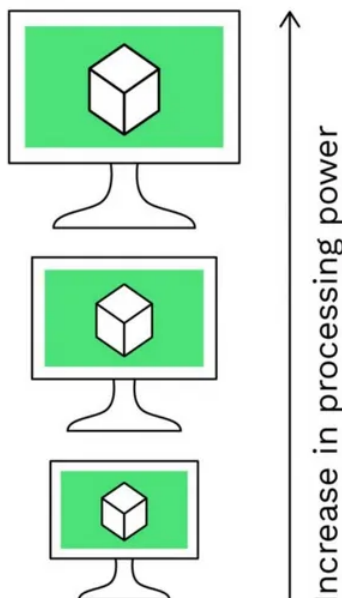


Share

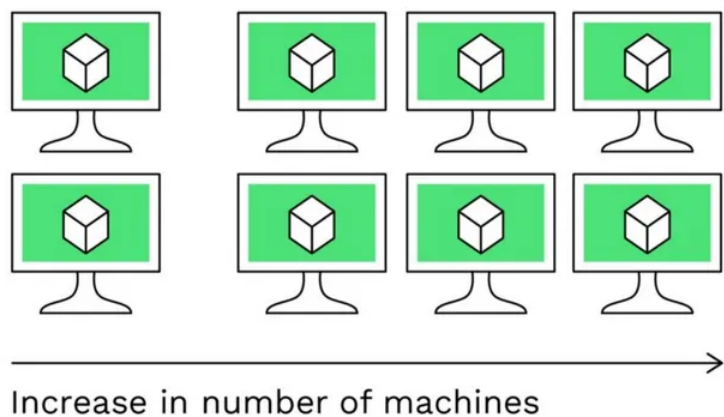
... More

## Scalability

### Vertical scaling



### Horizontal scaling



Hello folks , one question which I have been asked repeatedly on many System design and Java Developer interviews are *how do you scale your system?* In particular **how do you scale your Microservices?** This is a great question to ask any

experienced developers because its open ended and its gives an opportunity to candidate to express themselves and also interviewers to know more about their knowledge, experience, and thought process.

Thanks to my habit of reading blogs and articles, I feel I am little bit more familiar with new concepts and terms compared to my competition so when I was first asked this question on senior Java developer role and I mentioned that there are two ways to scale your application either horizontally or vertically then the interviewer was really impressed because he himself have never heard about the term, even though he may be familiar with the concept and process.

Now, you can say that I got lucky which is true but I was able to make an impression because I read blogs and articles and I can use buzz words and demonstrate that I understand them.

In this article, I will tell you the exact difference between horizontal and vertical scaling and when to use one over other so that you can impress your interviews.

Also, In past few articles I have been sharing my experience on Microservices like [50 Microservices Interview questions](#), [10 Microservice design principles](#), and [10 Microservice Patterns](#) which I shared earlier as well my article about [SAGA Design Pattern](#) and [Monolith vs Microservices architecture](#).

### Top 10 Microservice Design Patterns for Experienced Developers

10 essential Microservices design pattern to create robust, scalable, and reliable Microservices

medium.com

Now, coming back to the topic, **Horizontal scalability and vertical scalability are two approaches to scale any system including Microservices to handle increased load or to meet performance requirements.**

The main difference between them lies in how **they achieve scaling.**

**Horizontal scalability**, also known as **scale-out**, scale your system by **adding more nodes or servers** to distribute the load across them.

*Horizontal scaling is achieved by adding more identical machines to the existing system, which allows for better distribution of load and improved redundancy.*

This was tough say 5 to 10 years ago when you have to manually acquire server and install all the software, create new configuration and deploy your application but now in this age of Cloud Computing and tools like Terraform, Ansible, and Puppet, you can easily add more machine for your application without much fuss.

On the other hand, **vertical scalability**, also known as **scale-up**, involves adding more resources (such as CPU, RAM, or storage) to a single node or server to handle increased load.

This approach allows for an increase in the capacity of an existing node, which can handle more requests.

So, that was the basic difference, I will show you more difference in a while but if you are preparing for Microservice interview, this is one concept I like you to learn and learn better, but before that, I suggest you to **[join Medium](#)** to read your favorite stories without interruption and learn from great authors and developers like Google Engineers and Tech experts from FAANG companies. You can **[join Medium here](#)**

#### Join Medium with my referral link - Soma

Read this and every story from thousands of great writers on Medium). Read more from great writers on Medium like...

[medium.com](https://medium.com)

## Difference between Horizontal and Vertical Scaling in Microservices

Here are some additional points that differentiate horizontal scalability from vertical scalability:

Earlier, I have also shared [25 Advanced Java questions](#), [25 Spring Framework questions](#), [20 SQL queries from Interviews](#), [50 Microservices questions](#), [60 Tree Data Structure Questions](#), [15 System Design Questions](#), and [35 Core Java Questions](#) for Java developer interview and if you are preparing for interview you can see them.

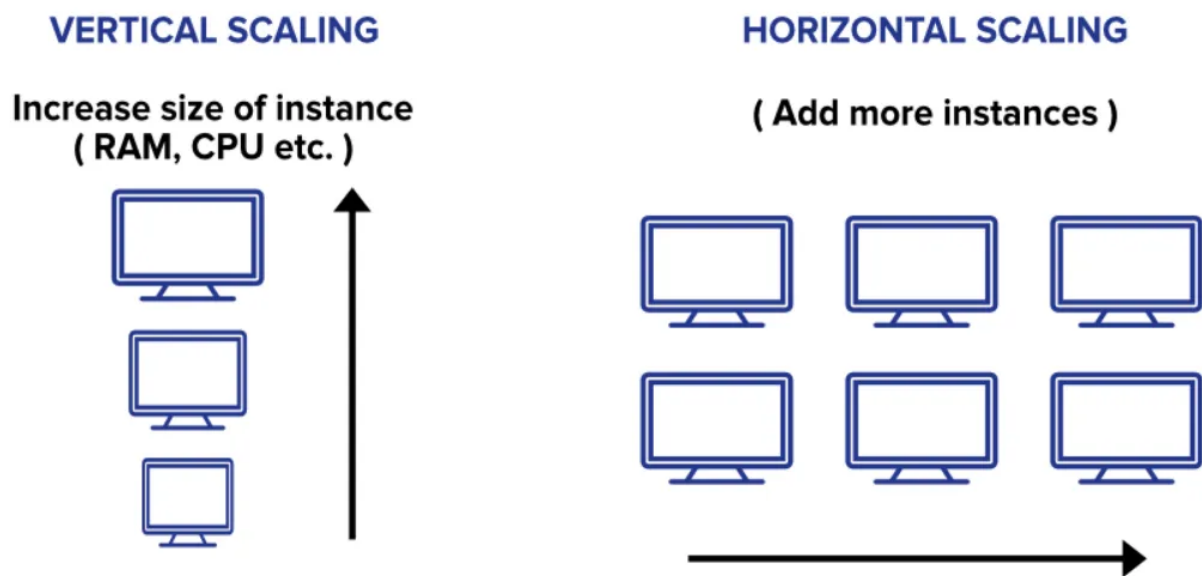
## 21 Lambda and Stream Interview Questions for Experienced Java Programmers

My favorite Java interview questions from Lambda expression and Stream API for experienced Java developers

medium.com

### 1. Scale up vs Scale out

Vertical scaling means adding more resources to single server while horizontal scaling means adding more instances that's why vertical scaling is now as scale up and horizontal scaling is known as scale out.



### 2. Load Balancer

Horizontal scalability requires the **use of load balancers to distribute incoming requests across multiple servers**, while vertical scalability does not, which is often a deciding factor. If your architecture doesn't permit load balancing then you only have to scale vertically.

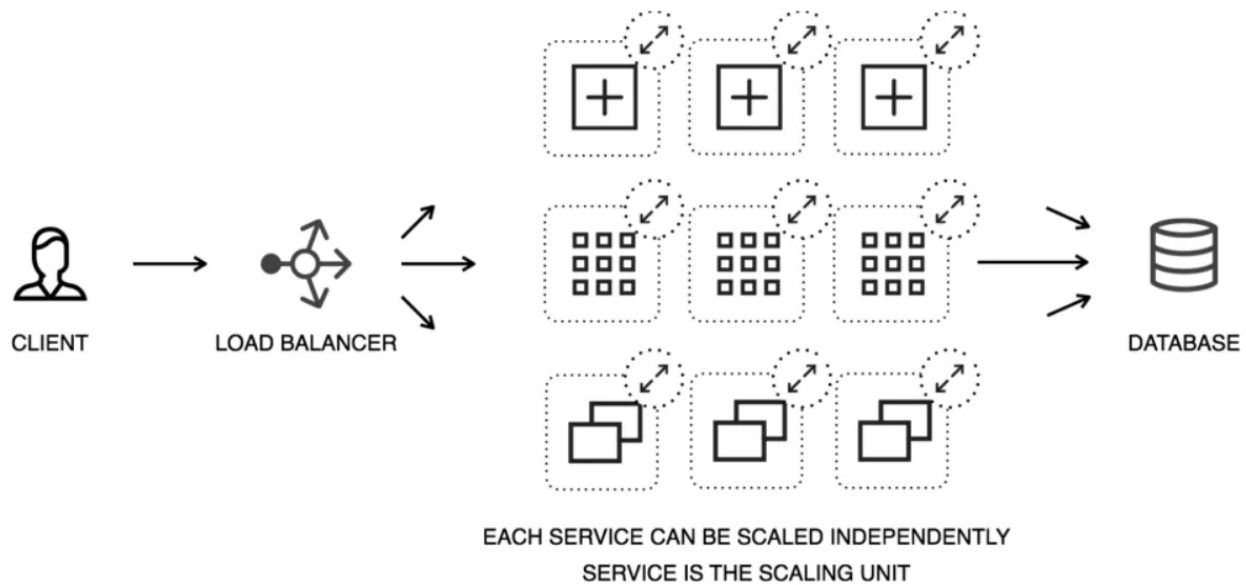
### 3. Higher Scalability

Horizontal scalability can achieve higher levels of scalability as the number of nodes can be increased almost indefinitely, while vertical scalability has limitations due to the maximum capacity of a single node or server.

### 4. Architecture Preference

Horizontal scalability is commonly associated with distributed systems, while vertical scalability is usually associated with monolithic systems.

# The Microservices Architecture



## 5. Handling Load

Horizontal scalability is more suitable for handling unpredictable or varying loads, while vertical scalability is more suitable for handling predictable loads.

## 6. Complexity

Horizontal scalability involves adding more nodes or servers to a system in order to handle increased load or traffic. However, horizontal scalability can be more complex to implement than vertical scalability, as it often requires more sophisticated load balancing and coordination mechanisms between nodes.

Additionally, horizontal scalability may not always result in a linear increase in performance, since additional nodes may introduce more overhead or communication delays.

On the other hand, Vertical scalability involves increasing the resources (such as CPU, RAM, or storage) available to a single server or instance in order to handle increased load or traffic.

The main advantage of vertical scalability is that it is often easier and less complex to implement than horizontal scalability, since it involves simply adding more resources to an existing system.

However, vertical scalability may be limited by the physical capacity of the hardware running the system, and may not be able to scale indefinitely.

Additionally, vertical scalability can be more expensive than horizontal scalability, since it often involves upgrading to more powerful or specialized hardware.

## 7. Example

An example of horizontal scalability is a web application that utilizes a load balancer to distribute incoming requests across multiple web servers. As traffic to the application increases, additional servers can be added to the cluster to handle the additional load.

Similarly, one example of vertical scalability is a database system that uses **sharding** to partition data across multiple disks or storage devices within a single server. As the size of the database grows, additional storage devices can be added to the server to accommodate the additional data.

## Which one is better for Microservices in Cloud?

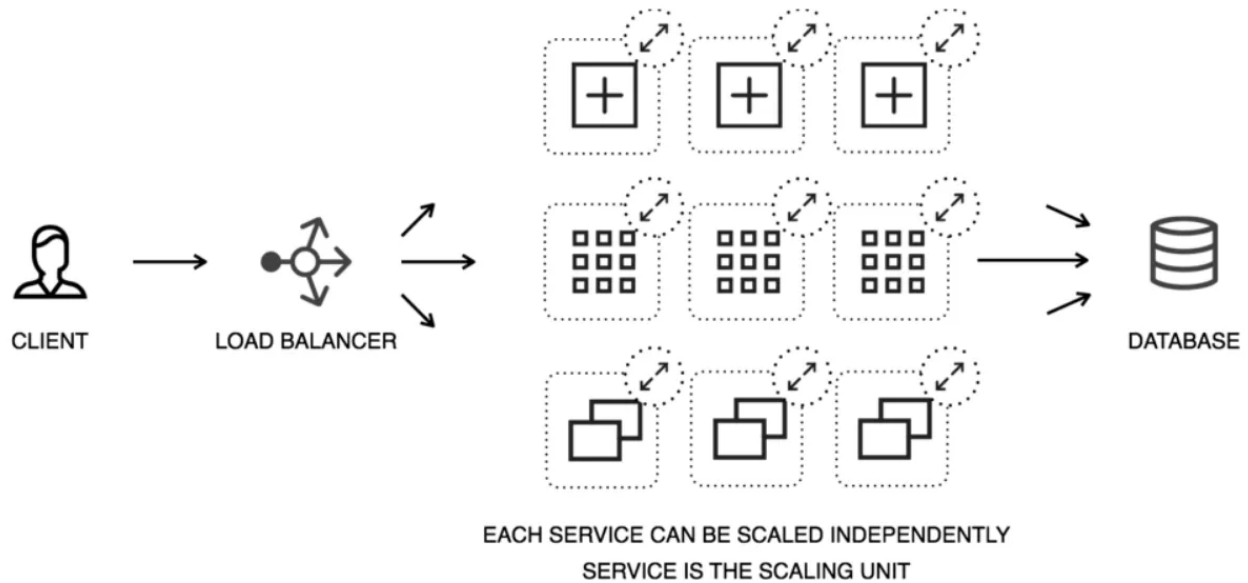
In general, horizontal scaling is considered better for microservices in the cloud because it allows you to add more instances of the service to handle increased load. With horizontal scaling, you can distribute the workload across multiple instances, which helps to ensure that the service remains available even if one instance goes down.

Additionally, horizontal scaling is more cost-effective because it allows you to add resources on an as-needed basis, rather than requiring you to invest in more powerful hardware upfront.

Vertical scaling, on the other hand, can be more expensive and less flexible because it requires upgrading the resources on a single instance to handle increased load.

This approach can result in downtime if the instance needs to be taken offline for maintenance or upgrades, and it may not be sufficient to handle sudden spikes in traffic. However, vertical scaling can be appropriate in certain situations where the workload is predictable and consistent, and it may be more straightforward to manage a single, more powerful instance than multiple smaller instances.

# The Microservices Architecture



## Java and Spring Interview Preparation Material

Before any Java and Spring Developer interview, I always read the below two resources

### Grokking the Java Interview

[Grokking the Java Interview: click here](#)

I have personally bought these books to speed up my preparation.

You can get your sample copy [here](#), check the content of it and go for it

**Grokking the Java Interview [Free Sample Copy]: [click here](#)**



*If you want to prepare for the Spring Boot interview you follow this consolidated ebook, it also contains microservice questions from spring boot interviews.*

Grokking the Spring Boot Interview

You can get your copy here — [Grokking the Spring Boot Interview](#)





That's all about **difference between horizontal and vertical scalability in context of System design and Microservices**. Overall, both horizontal and vertical scalability have their own advantages and disadvantages, and choosing the right one depends on the specific requirements and constraints of the system being developed. It is often a good idea to start with horizontal scaling and add more instances as needed, while keeping an eye on performance and costs.

And, if you like this article I also suggest you to **join Medium** to read your favorite stories without interruption and learn from great authors and developers like Google Engineers and Tech experts around the world. You can **join Medium [here](#)**

### **Join Medium with my referral link - Soma**

Read this and every story from thousands of great writers on Medium). Read more from great writers on Medium like...

[medium.com](https://medium.com)

Other Microservices articles yo may like to explore:

### **Top 10 Microservices Design Principles and Best Practices for Experienced Developers**

Designing Microservices for your organization? Follow these design principle to create a robust and scalable...

[medium.com](https://medium.com)

### **Difference between Microservices and Monolithic Architecture for Java Interviews**

What are differences between Monolith and Microservices architecture? Pros and cons, and when to use each of them

[medium.com](https://medium.com)

### **50 Microservices Design and Architecture Interview Questions for Experienced Java Programmers**

Preparing for Senior Java developer role where Microservices skill is required? Here are 50 questions which you should...

medium.com

## Top 10 Microservice Design Patterns for Experienced Developers

10 essential Microservices design pattern to create robust, scalable, and reliable Microservices

medium.com

Java

Programming

Microservices

Software Development

System Design Interview



Follow

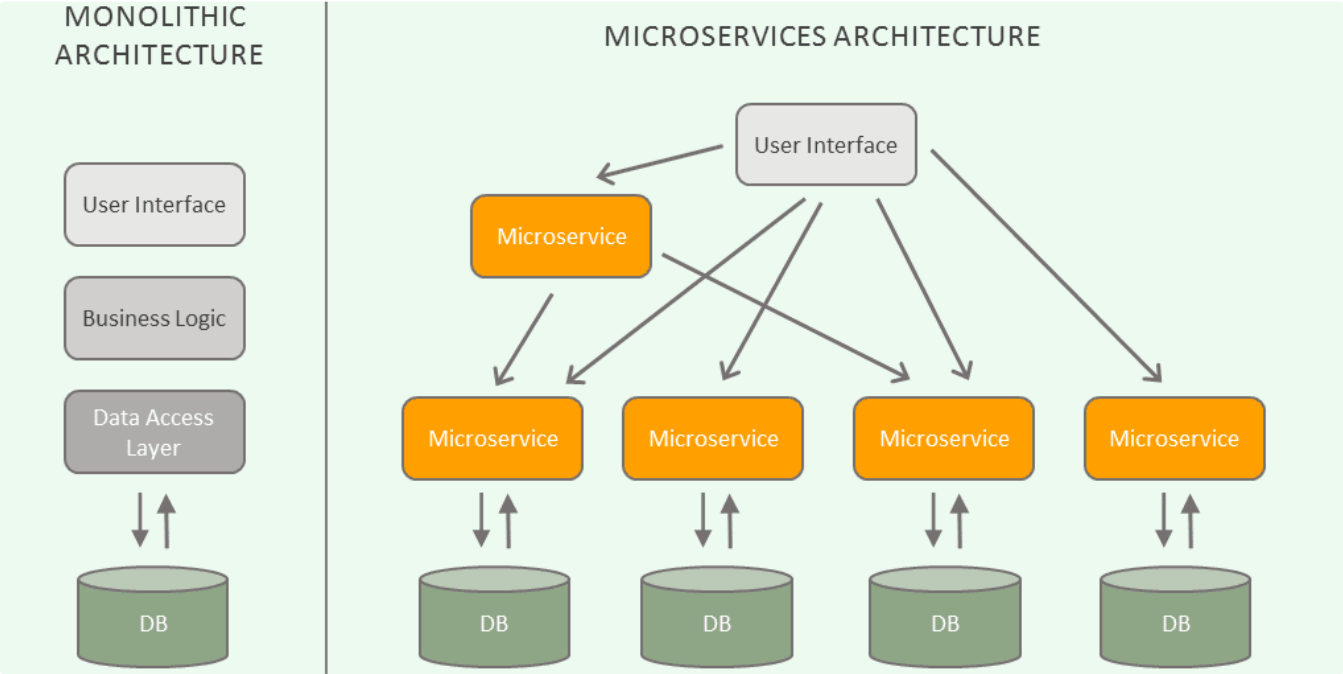
## Written by Soma

4.2K Followers · Editor for Javarevisited

Java and React developer, Join Medium (my favorite Java subscription) using my link 🙌

[https://medium.com/@somasharma\\_81597/membership](https://medium.com/@somasharma_81597/membership)

## More from Soma and Javarevisited



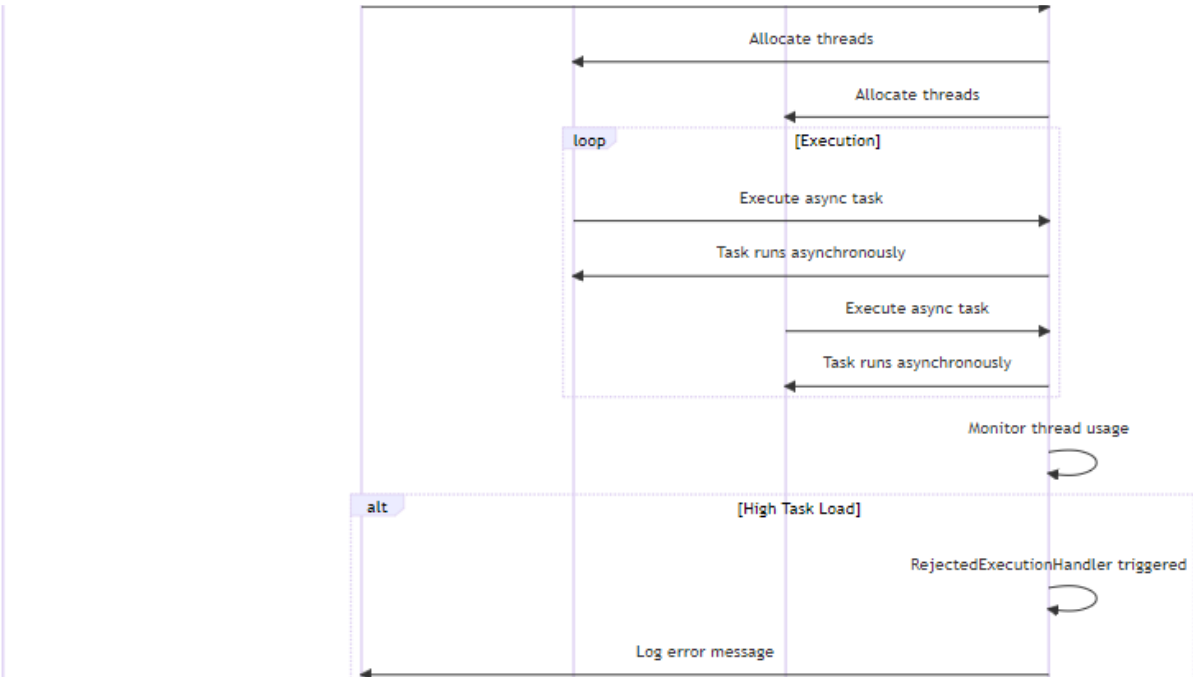
Soma in Javarevisited

## 18 Advanced Microservices Interview Questions for Experienced Developers: Challenges and Insights

From Microservice Architect to Scalability: 18 Interview Questions to Test the Microservices Skills of Developers with 5 to 10 Years of...

🌟 · 13 min read · 3 days ago

114



Srikanth Dannarapu in Javarevisited

## Using Async Schedulers in Spring Boot

Problem Statement: Efficiently Managing Multiple Asynchronous Schedulers in a Spring Boot Application

5 min read · Aug 7



109



1



javinpaul in Javarevisited

## 10 Projects You Can Build to Learn Golang in 2023

My favorite Golang Projects with courses for beginners and experienced developers

11 min read · Aug 7



98



1





 Soma in Javarevisited

## Difference between @RequestMapping and @GetMapping Annotations in Spring MVC Web Framework

Hello friends , if you are preparing for Java Developer and Spring Developer interviews then you must prepare for questions like...

🌟 · 6 min read · Aug 19

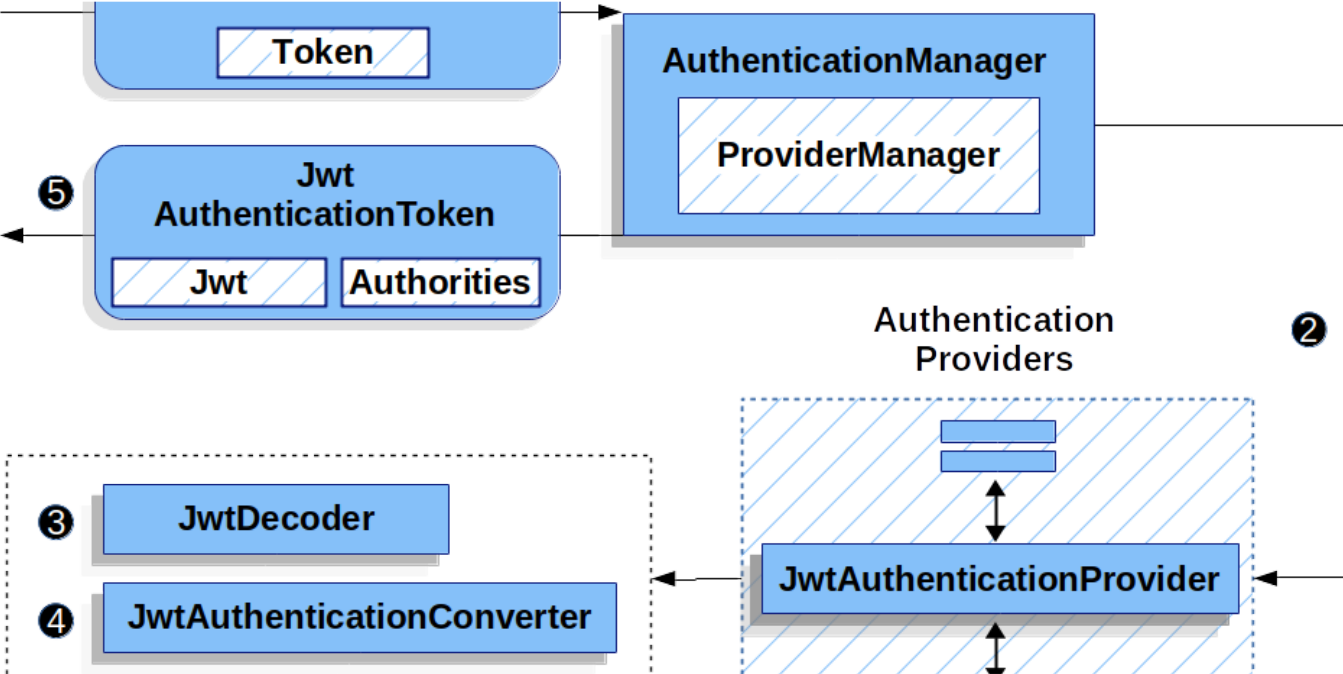
 49 

See all from Soma

See all from Javarevisited

### Recommended from Medium



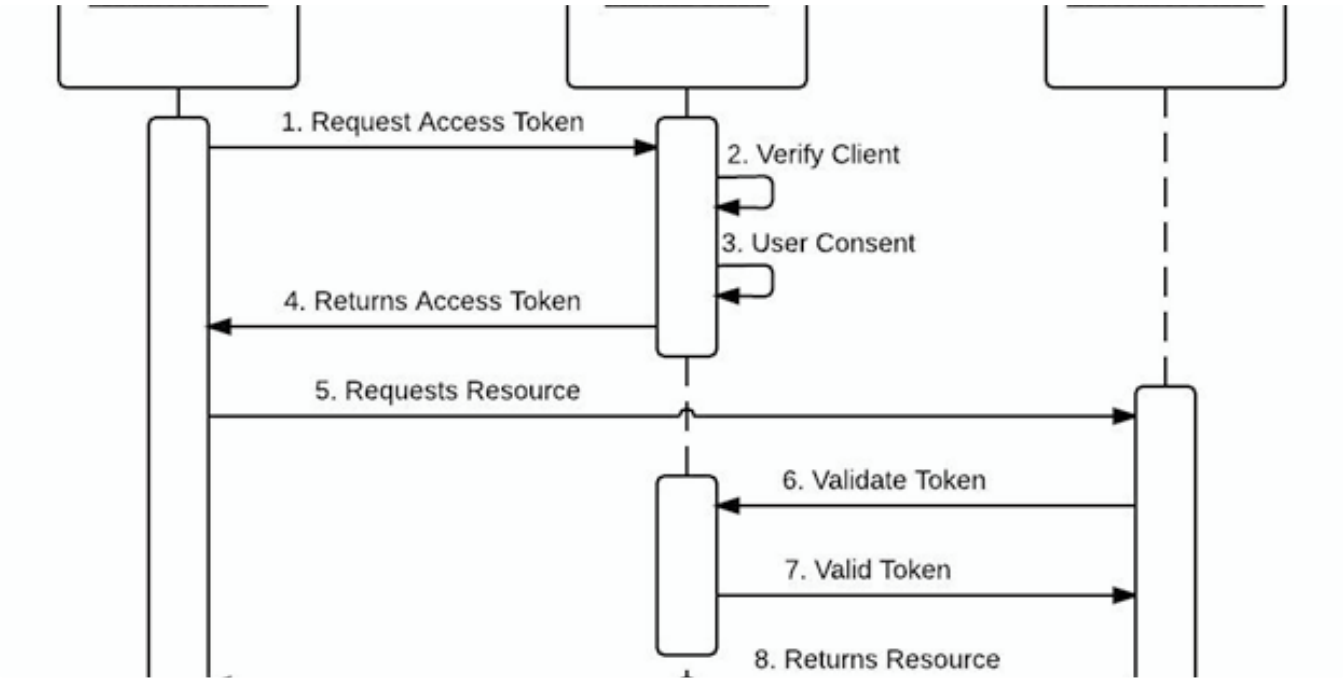
Alper Kurtoglu

## Spring OAuth2 with OIDC: Moving from ‘Scope’ to ‘Roles’

In this tutorial, I will discuss how to use authorities instead of scopes as roles in Spring OAuth2.

3 min read · Aug 17

83



Somal Chakraborty in Dev Genius

# Securing a Microservice with OAuth 2.0 (Part 2 : How Token Based Security Works Internally)

Table of content

5 min read · May 1



37



1



## Lists



### General Coding Knowledge

20 stories · 257 saves



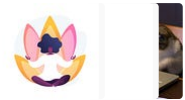
### It's never too late or early to start something

15 stories · 93 saves



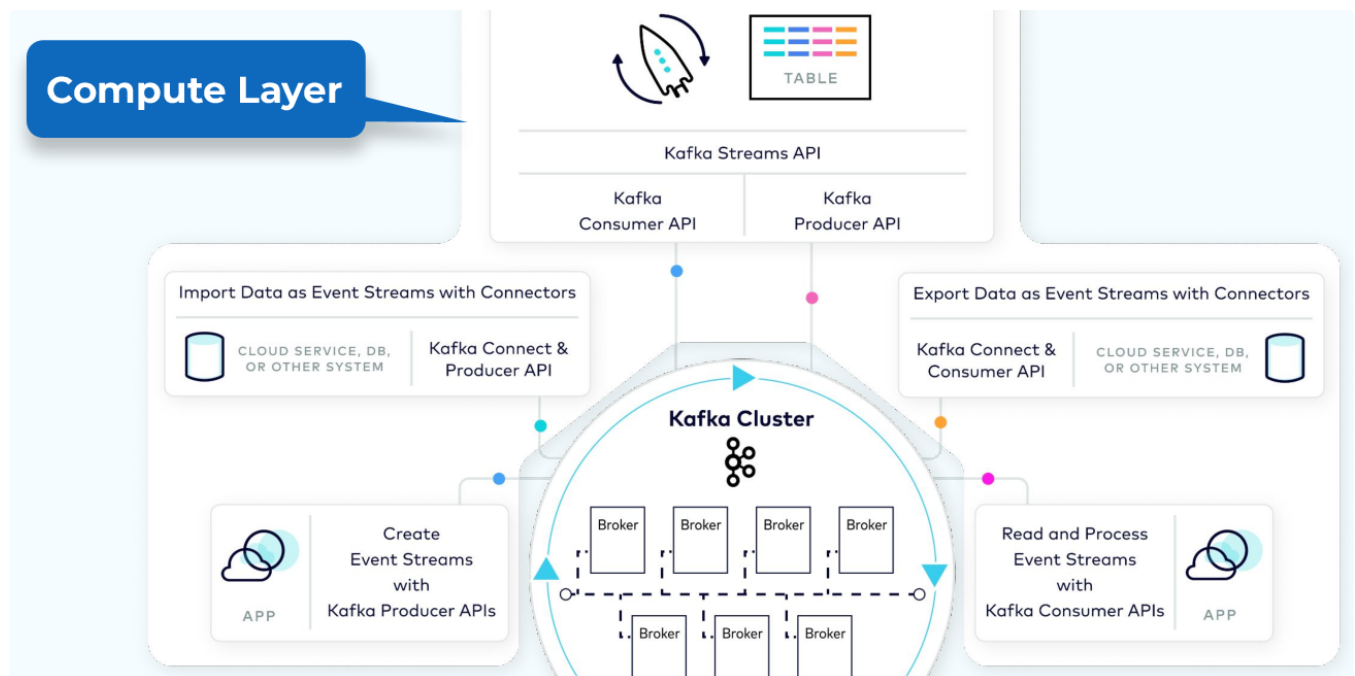
### Coding & Development

11 stories · 129 saves



### Stories to Help You Grow as a Software Developer

19 stories · 309 saves



Mahesh Saini

## Foundational Concepts of Kafka and Its Key Principles

Apache Kafka is a distributed event store and stream-processing platform. The project aims to provide a unified, high-throughput...

7 min read · May 2


 180











Ionut Anghel

## REST Endpoint Best Practices Every Developer Should Know

5 min read · Mar 18

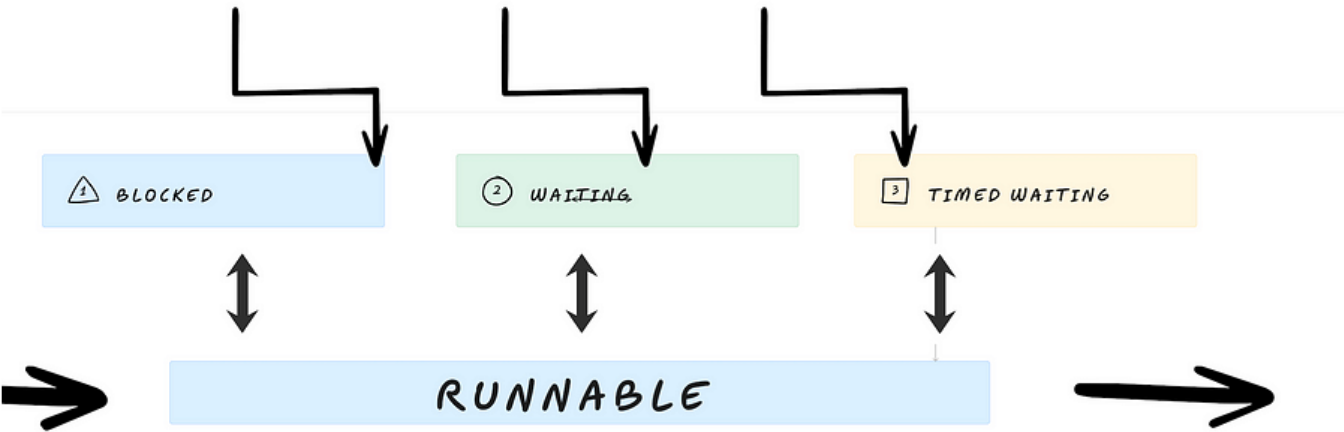
 79


 1










 Vikas Taank

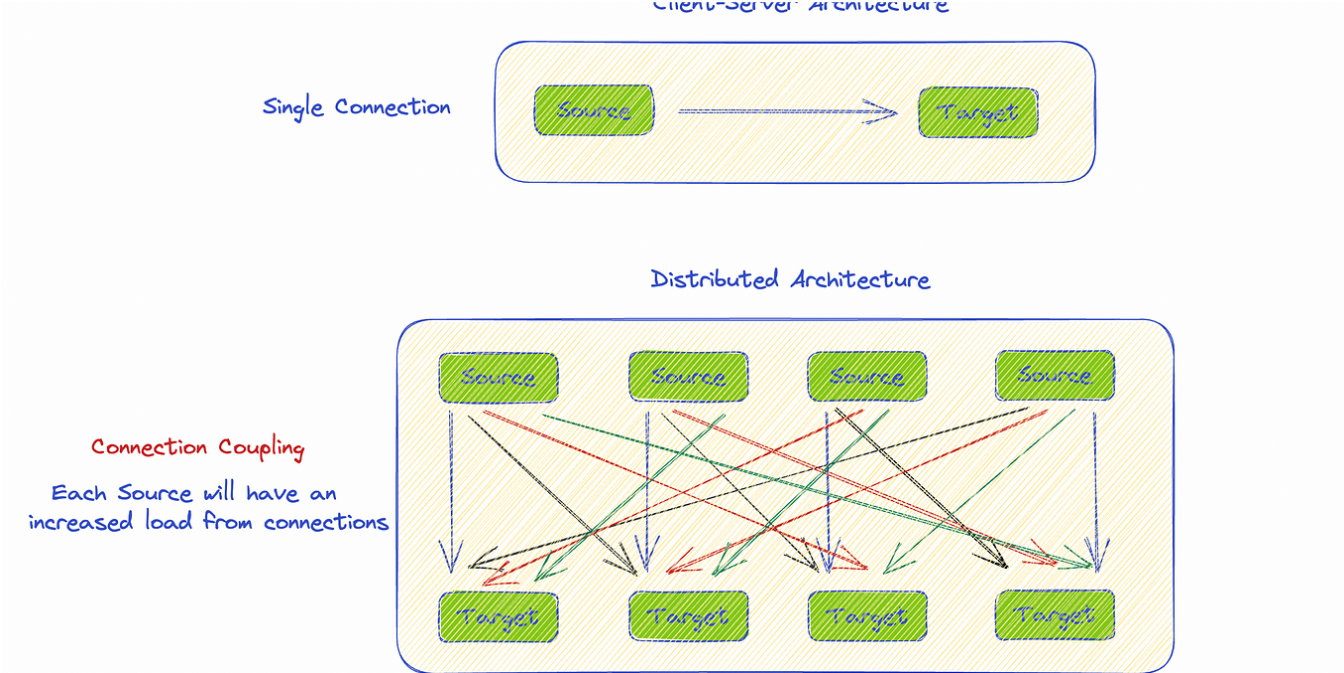
## All about Java Threads


Thread: What is a Thread , thread is a light weight process that denotes and execution in operating system which has a life cycle and that...

6 min read · Jul 14

 18 



 Akshat Vashishtha

# Kafka with Spring Boot using docker-compose

Prior understanding to Kafka, we should understand the problem Kafka try to solve. In simple Client-Server architecture source machine...

4 min read · May 9



39



See more recommendations