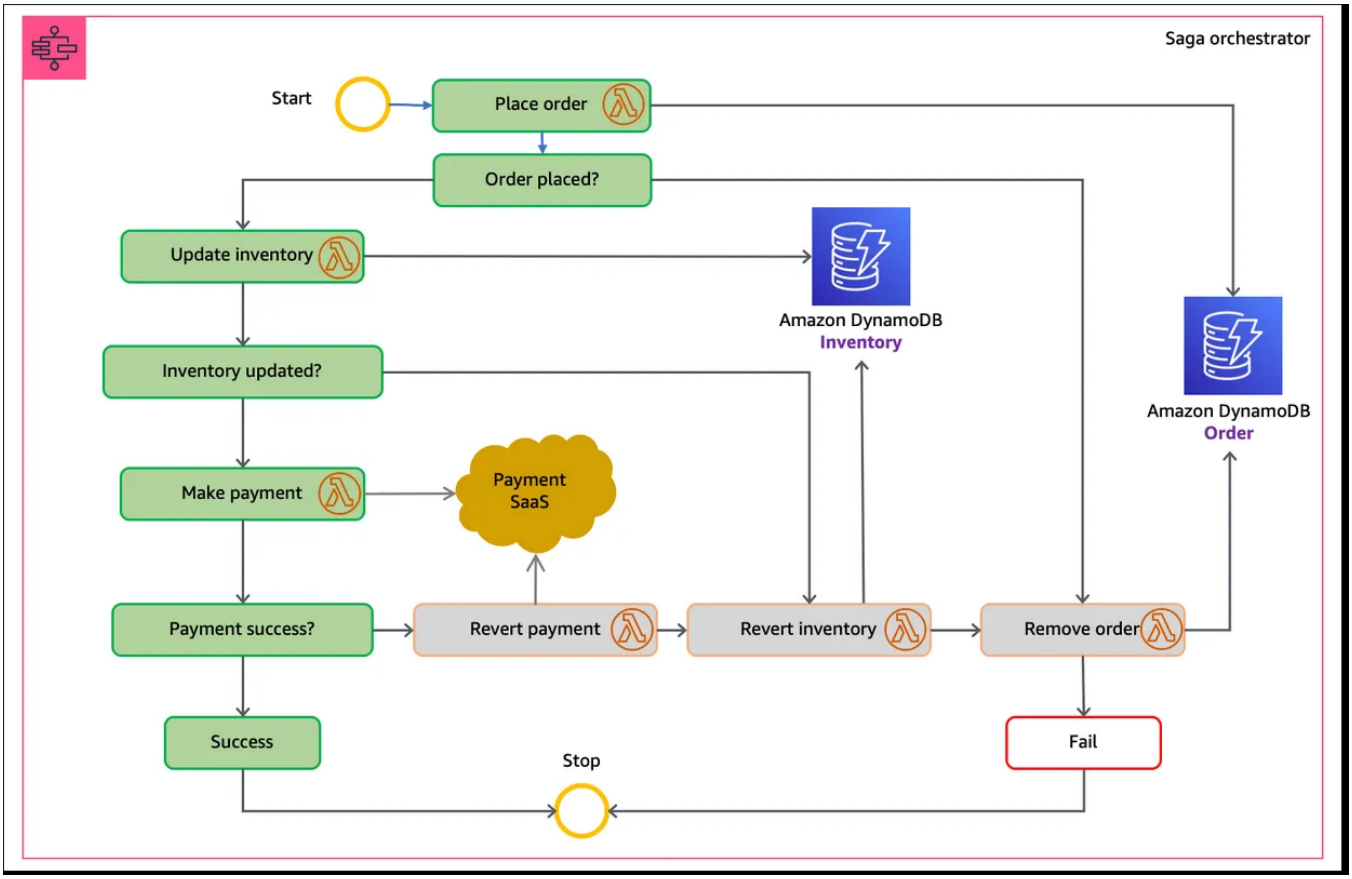
Difference between SAGA Pattern and 2 Phase Commit in Microservices?

Understanding the Pros and Cons of SAGA Pattern and 2 Phase Commit for Distributed Transactions in Microservices

Soma · Follow
Published in Javarevisited
6 min read · Apr 29



Hello folks, in my last interview for a senior Java developer position, I was asked this question, *"what is difference between SAGA and 2 Phase Commit? Which one will you prefer and why?"*, while I have **fair idea of both SAGA Pattern and 2 Phase commit** as I have not only used them but also written about them, but I think interview

pressure got me and I couldn't explain the difference as well I could, at least that's what I felt.

I haven't heard from them after that so I think, I probably not going to receive any offer and it was any way very senior position of managing their suite of Microservices and architecture new strategic solution, which I felt will require a lot of communication with stack-holders and very less coding, something not desirable at this moment, I would like to remain hands-on with coding.

Anyway, let's come back to the questions so that you can answer this question better when you see in your next Java Interview.

By the way, if you are not a Medium member then I highly recommend you to join Medium and read great stories from great authors from real field. You can **join Medium here**

Difference in SAGA Pattern vs 2 Phase Commit

While both the Saga pattern and Two-Phase Commit (2PC) are used to maintain data consistency in distributed systems, but they differ in their approaches.

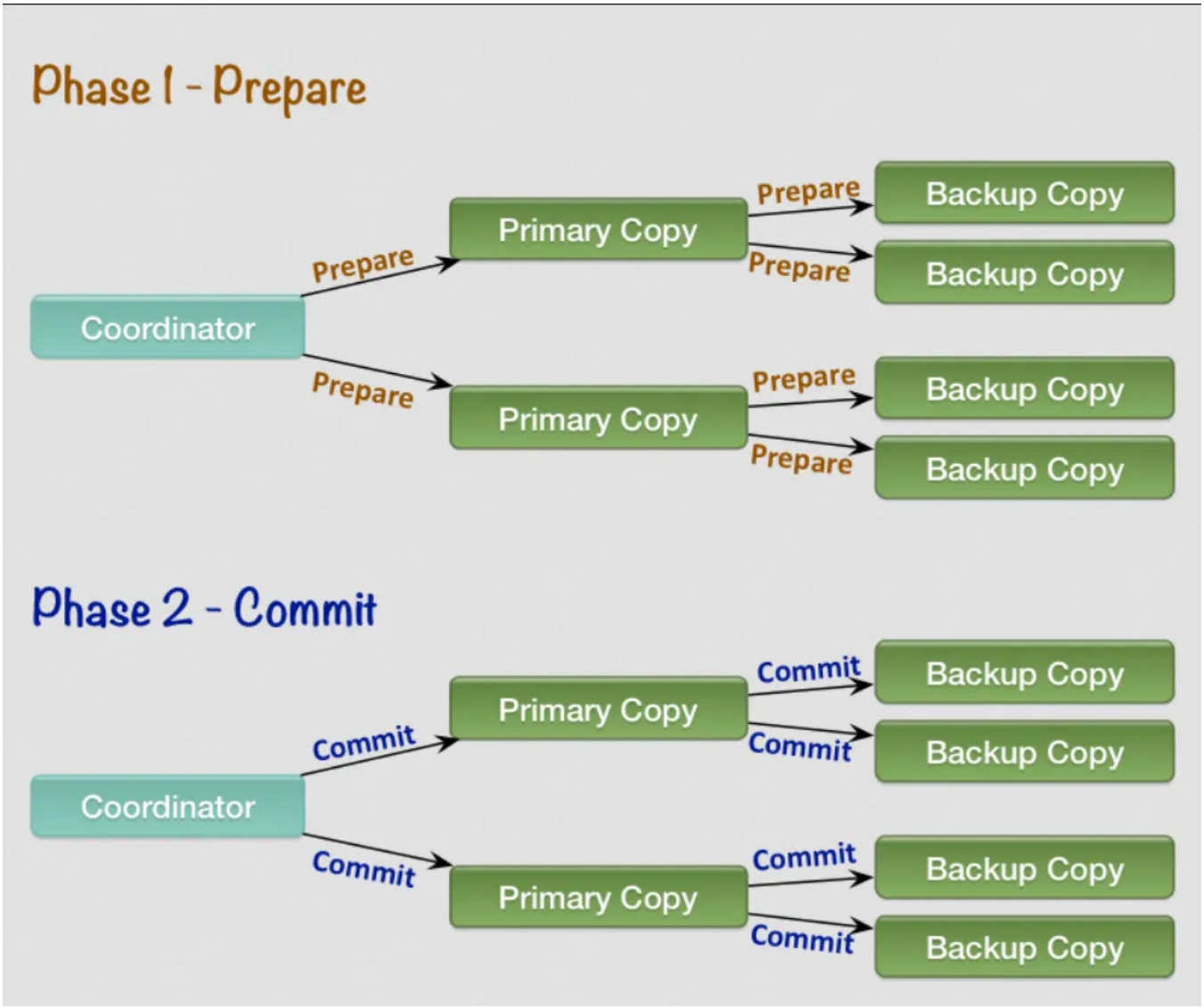
The Two-Phase Commit (2PC) protocol is a distributed algorithm used to ensure that all nodes in a distributed **system agree to commit or abort a transaction.**

2PC works by coordinating transactions between a coordinator node and multiple participant nodes. The coordinator sends a request to the participants to prepare for the transaction, and once all participants respond with a positive acknowledgement, the coordinator sends a commit message to each participant to **commit the transaction.**

If any participant fails to respond or sends a negative acknowledgement, the coordinator sends an abort message to all participants, and the transaction is rolled back.

2PC guarantees that all nodes will commit or abort a transaction, but it can be slow and vulnerable to failure.

Here is a nice diagram which illustrate **how 2 Phase commit works:**

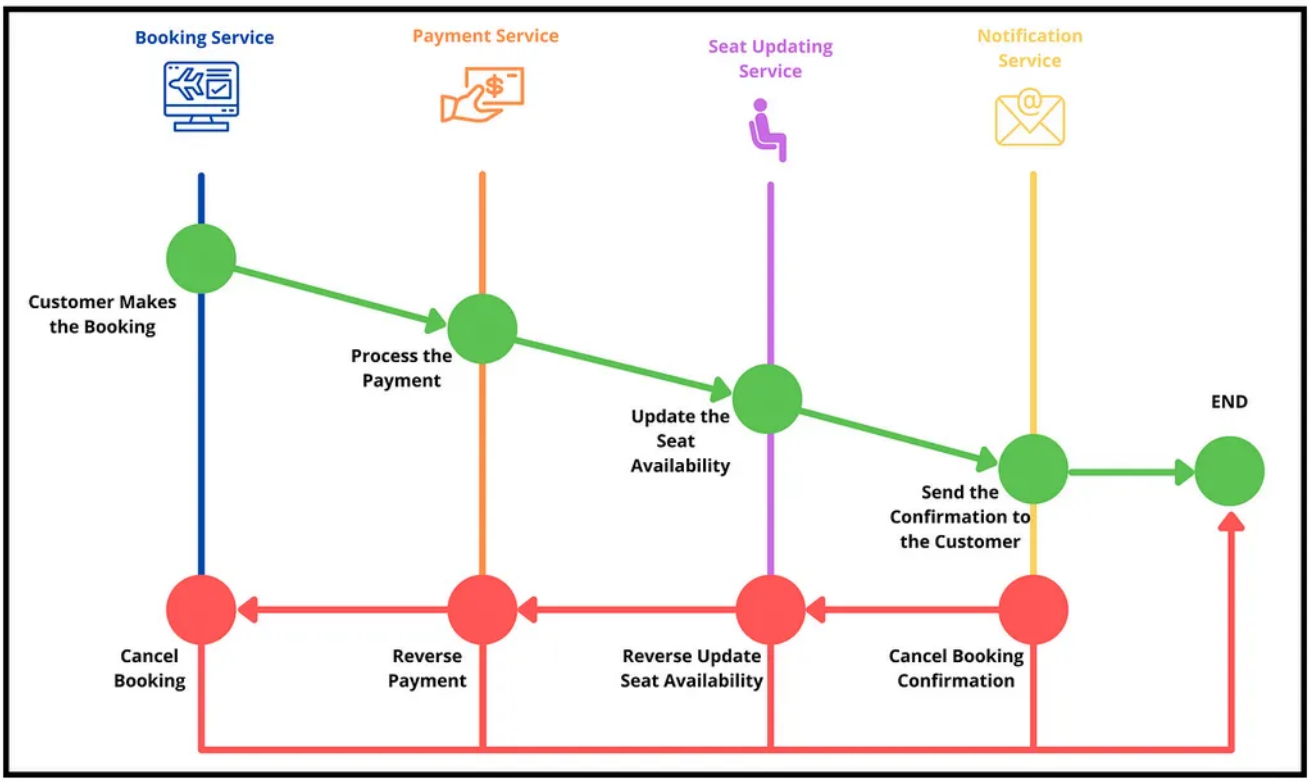


On the other hand, SAGA Pattern is a Microservices design pattern for **managing long-lived transactions in a distributed system.** A saga is a sequence of local transactions, where each local transaction updates the state of a single service, and each service has its own database.

Each local transaction is an atomic operation that either completes successfully or compensates for its effects. If a local transaction fails, the saga executes a compensating transaction that undoes the effects of the failed transaction.

The saga pattern allows for more flexibility than 2PC and can better handle complex, long-lived transactions, but it requires careful design to ensure data consistency.

Here is a nice diagram which shows **how SAGA Pattern works:**



When to use SAGA and 2 PC commit in Design?

This is the second part of the question which slightly hard to answer because we only know that both are used to manage distributed transaction in microservices but there are more to it.

The 2PC protocol is **useful in situations where all participants of the distributed transaction must commit or roll back the transaction together.** It ensures **atomicity** and **consistency** of the transaction but can lead to *blocking and performance issues in highly distributed systems*.

Therefore, it's typically used in *scenarios with a small number of participants,* where the latency and scalability limitations of 2PC can be managed.

In contrast, **SAGA pattern is useful in situations where the transaction is too large to be managed by a single 2PC protocol.** SAGA breaks the transaction down into smaller, local transactions that can be independently managed by each microservice.

This allows for greater scalability, fault tolerance, and performance in highly distributed systems. SAGA is typically used in complex, long-running business processes that involve multiple microservices, where a transaction rollback would be expensive or not feasible.

What is SAGA Pattern in Microservice Architecture? Which Problem does it solve?

SAGA is an essential Micro service Pattern which solves the problem of maintaining data consistency in distributed...

medium.com

Java and Spring Interview Preparation Material

Before any Java and Spring Developer interview, I always use to read the below 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 SAGA Pattern and 2 Phase commit in Microservice architecture.** In summary, 2PC is a distributed protocol that coordinates transaction commits, while the Saga pattern is a design pattern for

managing long-lived transactions by breaking them down into a sequence of local transactions that can be compensated for if necessary.

2 Phase commit is more appropriate for *scenarios with a small number of participants,* where the latency and scalability limitations of 2PC can be managed, while **SAGA pattern is useful in situations where the transaction is too large to be managed by a single 2PC protocol**

And, if you are preparing for java interviews, you can also check my previous articles where I have shared more than 300+ questions with answers like ***21 Software Design Pattern questions***, 10 *Microservice Scenario based questions*, *20 SQL queries from Interviews*, ***50 Microservices questions***, *60 Tree Data Structure Questions*, 15 System Design Questions, and ***35 Core Java Questions*** and *21 Lambda and Stream questions* etc.

Additionally, you can also prepare Microservices Questions like *difference between API Gateway and Load Balancer*, *SAGA Pattern*, *how to manage transactions in Microservices*, and *difference between SAGA and CQRS Pattern*, they are quite popular on interviews.

And, if you are not a Medium member then I highly recommend you to join Medium and read great stories from great authors from real field. You can **join Medium here**

Other **Java Microservices articles** you may like:

50+ Spring Boot Interview Questions and Answers for Java Programmers

These are 50+ Spring Boot Interview Questions and answers to Crack your Next Java Developer interview

medium.com

Top 10 Microservice Architecture Design Patterns Every Developer Should Learn

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

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

Microservices | Programming | Development | Software Engineering | Software Development

Follow

Written by Soma

4.1K Followers · Editor for Javarevisited

Java and React developer, Join Medium (my favorite Java subscription) using my link 👉
https://medium.com/@somasharma_81597/membership

More from Soma and Javarevisited

Implementation	implementation	an implementation	features
Persistence API	Hibernate provides its own API	Defines a set of standard APIs for ORM	Builds on JPA APIs, adds more functionality
Database Support	Supports various databases through dialects	Depends on the JPA implementation used	Depends on the JPA implementation used
Transaction Management	Provides its own transaction management	Depends on the JPA implementation used	Depends on the JPA implementation used
Query Language	Hibernate Query Language (HQL)	JPQL (Java Persistence Query Language)	JPQL (Java Persistence Query Language)
Caching	Provides first-level and second-level caching	Depends on the JPA implementation used	Depends on the JPA implementation used
Configuration	XML, annotations, or Java-based configuration	XML, annotations, or Java-based configuration	XML, annotations, or Java-based configuration
Integration	Can be used independently or with JPA	Works with any JPA-compliant implementation	Works with any JPA-compliant implementation

Soma in Javarevisited

Difference between Hibernate, JPA, and Spring Data JPA?

Hello folks, if you are preparing for Java Developer interviews then part from preparing Core Java, Spring Boot, and Microservices, you...

✨ · 10 min read · May 26

253 1

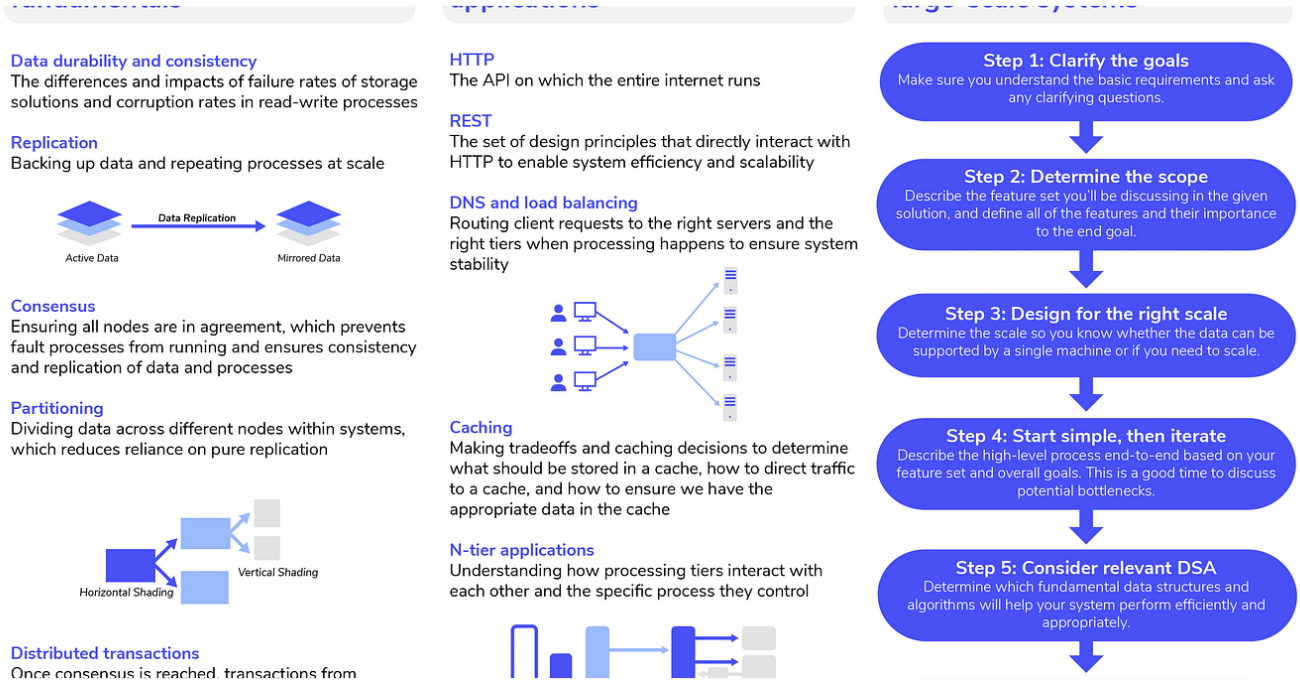
javinpaul in Javarevisited

Top 10 Websites to Learn Python Programming for FREE [2023]

Hello guys, if you are here then let me first congratulate you for making the right decision to learn Python programming language, the...

13 min read · Apr 22, 2020

328 6



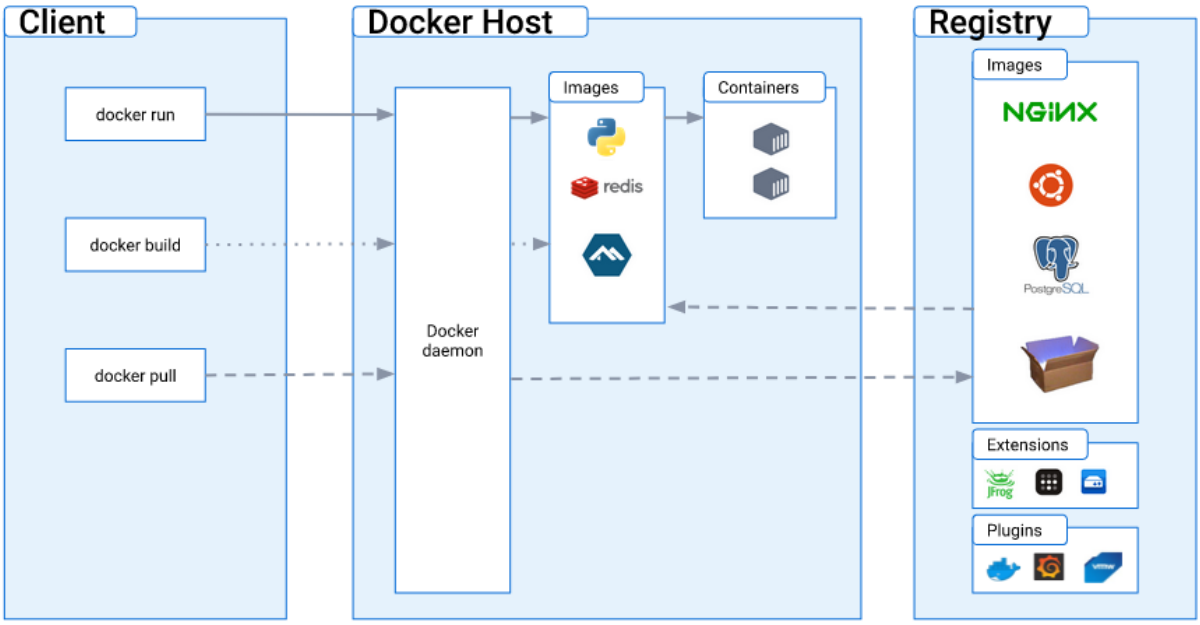
javinpaul in Javarevisited

Top 3 System Design Cheat Sheets for Developers

3 System Design Cheat Sheet you can print and put on your desktop to revise before Tech interviews

6 min read · Jul 19

204 1



Soma in Javarevisited

How Docker works internally? Magic Behind Containerization

Exploring the Inner Workings of Docker: Unveiling the Magic Behind Containerization

· 6 min read · Jun 30

183

See all from Soma

See all from Javarevisited

Recommended from Medium

Anto Semeraro in Level Up Coding

Microservices Orchestration Best Practices and Tools

Optimizing microservices communication and coordination through orchestration pattern with an example in C#

✨ · 10 min read · Mar 27

67

Amit Himani

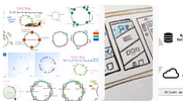
Distributed Transaction in Spring Boot Microservices

Distributed transactions in microservices refer to transactions that involve multiple microservices, each handling a part of the...

✦ · 5 min read · Feb 17

49 3

Lists

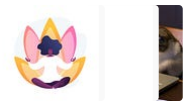


General Coding Knowledge

20 stories · 193 saves

It's never too late or early to start something

13 stories · 67 saves



Stories to Help You Grow as a Software Developer

19 stories · 263 saves

Coding & Development

11 stories · 100 saves

arctic_fox in Stackademic

Leaderboard using Spring Boot and Relational Database

Design for scalable and configurable leaderboard in spring boot and relational database

6 min read · Jul 31

2

Sowmya.L.R

Docker—Container era (PART-I)

In earlier days people rarely used web apps. But in this digital era, every single task is done by any particular app. Ex grocery buying...

4 min read · 2 days ago

10

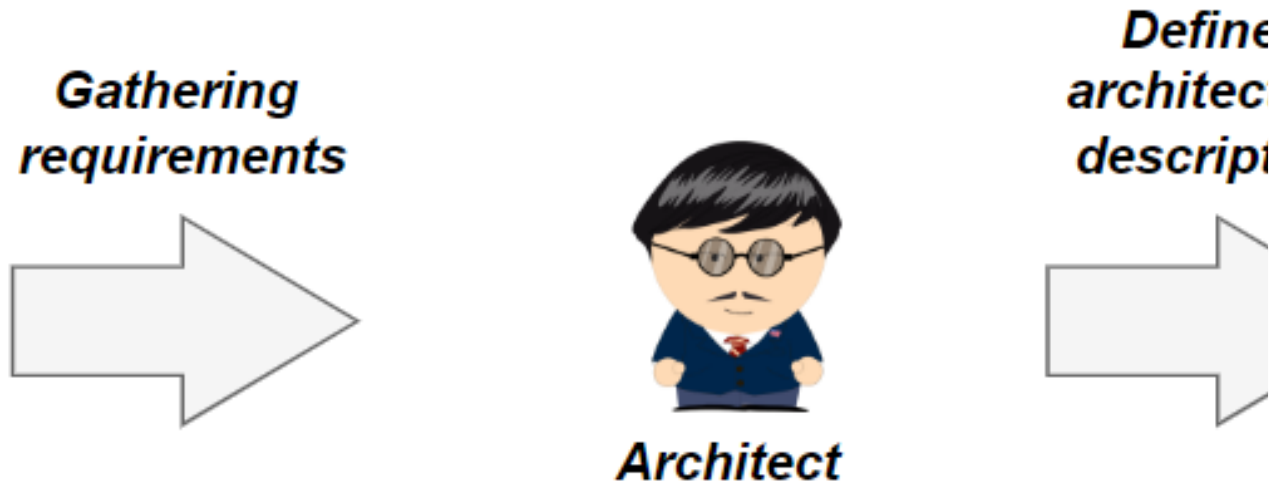
Keshav Gupta in Walmart Global Tech Blog

Event Sourcing Design Pattern

This blog will discuss the Event Sourcing design pattern and how this can be used in designing large-scale distributed systems.

5 min read · Mar 7

55



Svosh

Who is the architect?

In a few words, a solution architect is a professional who specializes in designing and developing solutions to address the specific needs...

5 min read · Jul 18

 1