

[Open in app](#)

★ Member-only story

# Difference between Forward Proxy and Reverse Proxy in System Design

Understanding the Distinctions between Forward and Reverse Proxies in System Design and when to use them

Soma · [Follow](#)

Published in Javarevisited

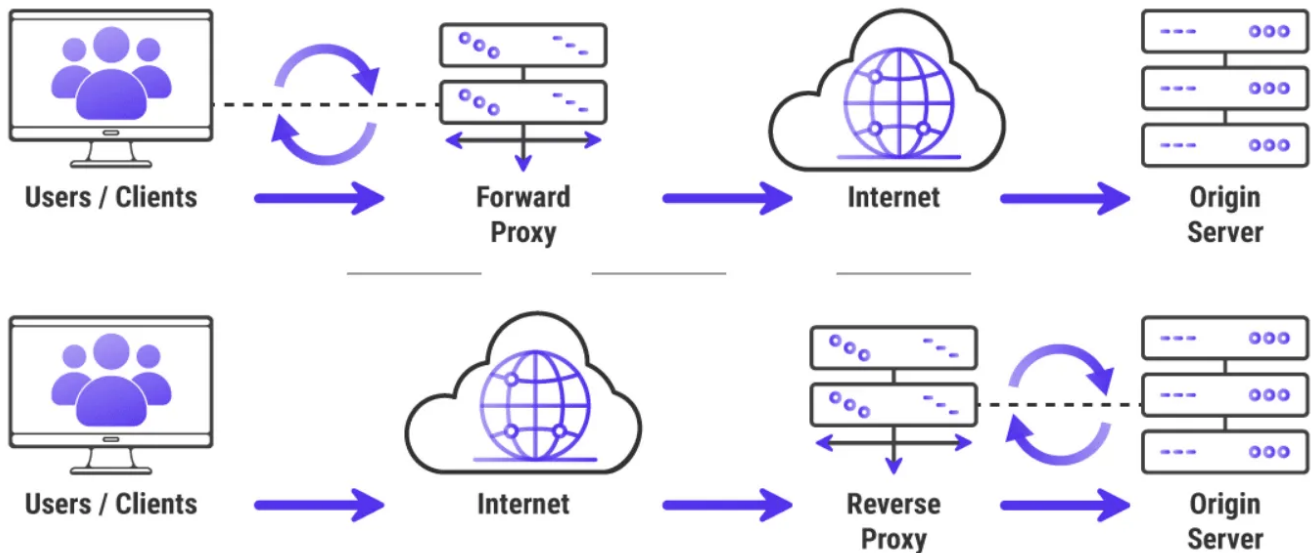
9 min read · Apr 20

Listen

Share

More

## Forward Proxy vs Reverse Proxy



image\_credit — <https://kinsta.com/wp-content/uploads/2020/08/Forward-Proxy-vs-Reverse-Proxy-Servers.png>

Hello folks, if you are preparing for System design interview then knowing the difference between forward proxy and reverse proxy is very important, its one of the most frequently asked question on System Design, along with **difference between API Gateway and Load Balancer**, which we have seen earlier.

When designing complex systems, it's common to use proxy servers to improve performance, security, and reliability. Proxy servers sit between clients and servers and help manage traffic between them.

Two types of proxies that are often used are forward proxies and reverse proxies. While both are designed to **improve the performance and security of a system**, they work in different ways and are used in different contexts. In this article, we'll explore the differences between forward proxies and reverse proxies in system design.

By the way, if you are preparing for senior developer interviews then along with System Design you should also familiar with different architectures like Microservices and various Microservice design patterns like Event Sourcing, CQRS, SAGA, Database Per Microservices, API Gateway, Circuit-Breaker they will help you immensely during interview as they are often used to gauge your seniority level.

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](#)**

#### 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)

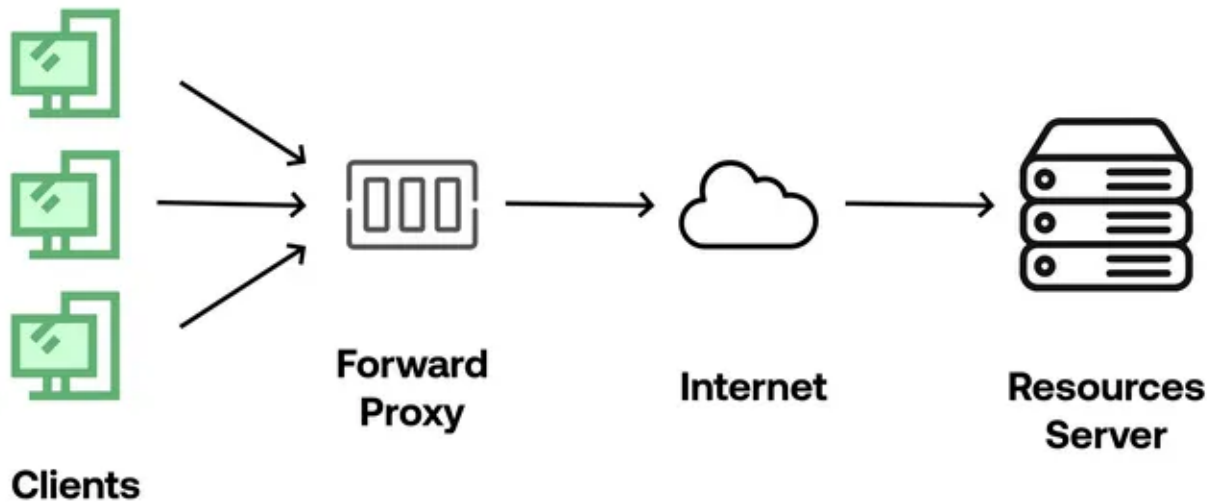
Now, let's come back to the topic and find out more about what is forward and reverse proxies, how to use them, what are their pros and cons and most importantly difference between forward and reverse proxies.

### What is Forward Proxy? when to use it?

A forward proxy is a proxy server that sits between a client and the internet. The client requests a resource or service from the internet through the forward proxy, which acts as an intermediary, forwarding the request on to the internet and then returning the response to the client.

Forward proxies are typically used to **control access to the internet**, **filter content**, or **provide anonymity for the client**. They can also be used to speed up access to resources by caching frequently requested content.

Here is an example of forward proxy:



You can see that clients connect to forward proxy and then it routes request to internet.

### What is Pros and Cons of Forward Proxy?

Now that you are familiar with the location of proxy in forward proxy, you can easily find out the pros and cons of it. Here are some pros and cons of using a forward proxy:

#### Pros:

1. **Enhanced security:** Forward proxy can provide an additional layer of security by hiding the original IP address of the clients accessing the internet.
2. **Improved speed and performance:** Caching frequently requested resources can improve response times for clients accessing the internet.
3. **Access control:** By restricting access to certain resources, organizations can use forward proxies to prevent unauthorized access to sensitive data.
4. **Anonymity:** Users can remain anonymous while browsing the internet, as their IP addresses are hidden.

#### Cons:

1. **Complex configuration:** Forward proxies require a more complex configuration as they have to be set up on individual devices to be effective.

2. **Single point of failure:** If the forward proxy fails, all the devices that rely on it will also fail to access the internet.
3. **Increased latency:** Forward proxies can increase latency and slow down the overall performance of internet access.
4. **Limited control:** Forward proxies can limit users' ability to access certain resources, leading to frustration and reduced productivity.

Apart from the limitation the access control and security benefit it provides is the main reason for using forward proxy on various architecture. Now, let's see what is Reverse proxy and how does it work.

## What is Reverse Proxy? when to use it?

A reverse proxy is a server that sits between the client and the origin server, receiving requests from clients and forwarding them to the appropriate server. The response from the server is then returned to the proxy and forwarded to the client. In essence, it helps to protect the origin server from direct access by clients.

Reverse proxies are commonly used as **load balancer to balance the load across multiple servers**, improve security by hiding the details of the server infrastructure, and provide other value-added services such as caching and SSL termination.

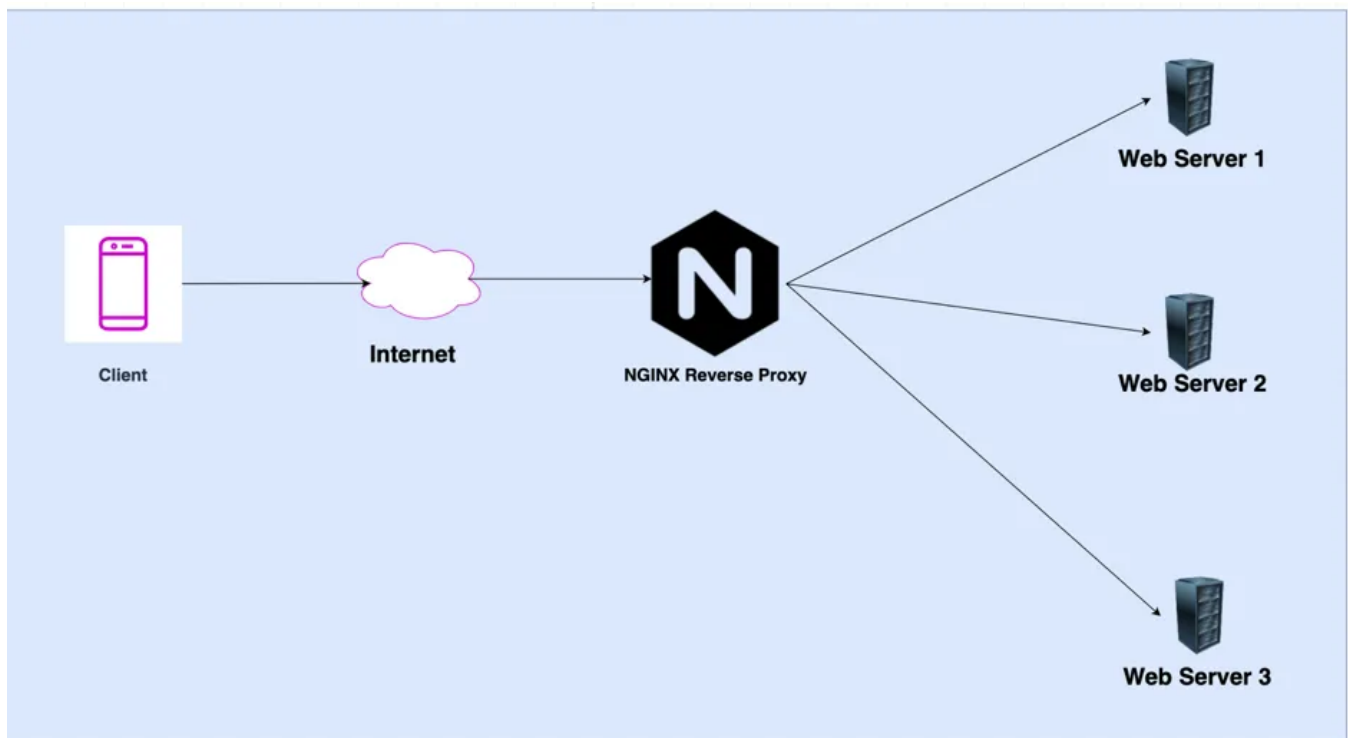
Reverse proxies are often used in the following scenarios:

- **Load balancing:** distributing incoming traffic across multiple servers to improve performance and availability.
- **Security:** protecting the backend servers from direct exposure to the internet and preventing unauthorized access.
- **Scalability:** allowing horizontal scaling of the server infrastructure without affecting the clients.

Reverse proxies **provide a single entry point for clients**, making it easier to manage and monitor the traffic to the backend servers. They also provide a level of

abstraction between the clients and servers, allowing the server infrastructure to be modified or upgraded without affecting the clients.

Here is an example of NGINX reverse proxy setup:



You can see that client connects to internet directly but servers are behind proxy so client will never know which server their request are processed, hence security your infrastructure from outside.

### What is Pros and Cons of Reverse Proxy Architecture?

Here are some pros and cons of using a reverse proxy architecture:

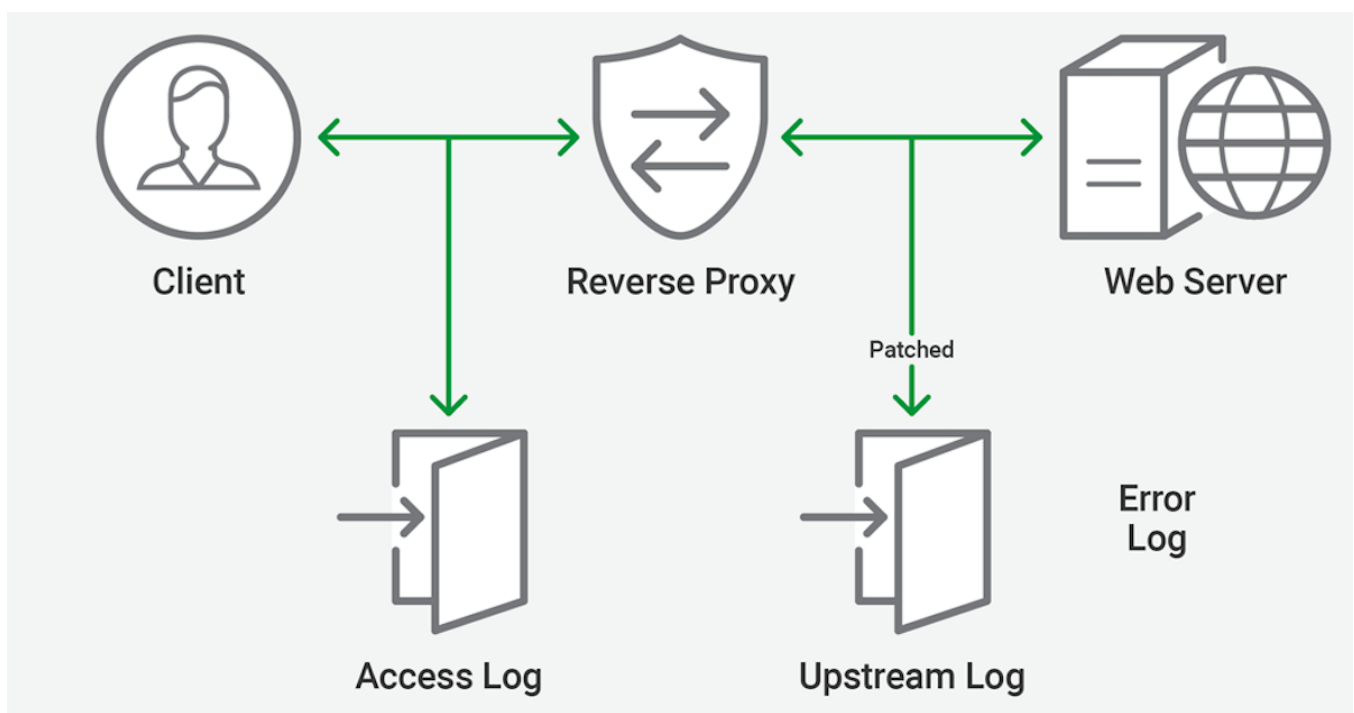
#### Pros:

1. **Increased security:** Reverse proxies can provide an additional layer of security by masking the identity and location of backend servers, preventing direct access to them from external clients.
2. **Better scalability:** Reverse proxies can distribute traffic evenly across multiple backend servers, ensuring that no single server becomes overloaded and causing the application to crash.
3. **Improved performance:** By caching and compressing data, reverse proxies can reduce the amount of data that needs to be transferred between clients and servers, leading to faster response times.

4. **Simplified architecture:** Reverse proxies can be used to consolidate multiple backend servers into a single endpoint, simplifying the overall architecture of the application.

**Cons:**

1. **Single point of failure:** If the reverse proxy fails, the entire application may become unavailable. This is its biggest drawback.
2. **Increased complexity:** Implementing and maintaining a reverse proxy can be more complex than a simple client-server architecture.
3. **Limited customization:** Reverse proxies may not offer the same level of customization as a direct connection between clients and servers, which could limit the functionality of the application.
4. **Additional cost:** Implementing a reverse proxy can require additional hardware and software, which could increase the cost of the overall system.

**What is difference between Forward Proxy and Reverse Proxy?**

Now that you have basic idea of what is forward and reverse proxy, their location as well their function, now is the time to look into differences to understand them better:

## **1. Direction**

The main difference between a forward proxy and a reverse proxy is the direction of traffic flow. Forward proxy is used to forward traffic from a client to the internet, while a reverse proxy is used to forward traffic from the internet to a web server.

## **2. Client Access**

With a forward proxy, clients have to be explicitly configured to use the proxy server. In contrast, a reverse proxy is transparent to the client, and clients can access the web server directly without needing to know the proxy server's address.

## **3. Load Balancing**

Reverse proxies can distribute incoming requests across multiple servers to balance the load, while forward proxies cannot.

## **4. Caching**

Forward proxies can cache frequently accessed resources to reduce the load on the web server and speed up the response time for subsequent requests. Reverse proxies can also cache resources, but the caching is typically done closer to the client to improve performance.

## **5. Security**

A forward proxy can be used to protect a client's identity by hiding their IP address from the internet. A reverse proxy can be used to protect a server by hiding its identity and exposing a single IP address to the internet.

## **6. SSL/TLS Termination**

A reverse proxy can terminate SSL/TLS connections on behalf of a web server to reduce the load on the server and simplify certificate management. Forward proxies typically do not terminate SSL/TLS connections.

## **7. Content Filtering**

Forward proxies can be used to filter content, block access to specific websites, and enforce access policies. Reverse proxies can also perform content filtering, but this is typically done closer to the client to reduce the load on the web server.

## **8. Routing**

Forward proxies can be used to route traffic to different servers based on predefined rules. Reverse proxies can also perform routing, but this is typically done based on the requested URL and other criteria.

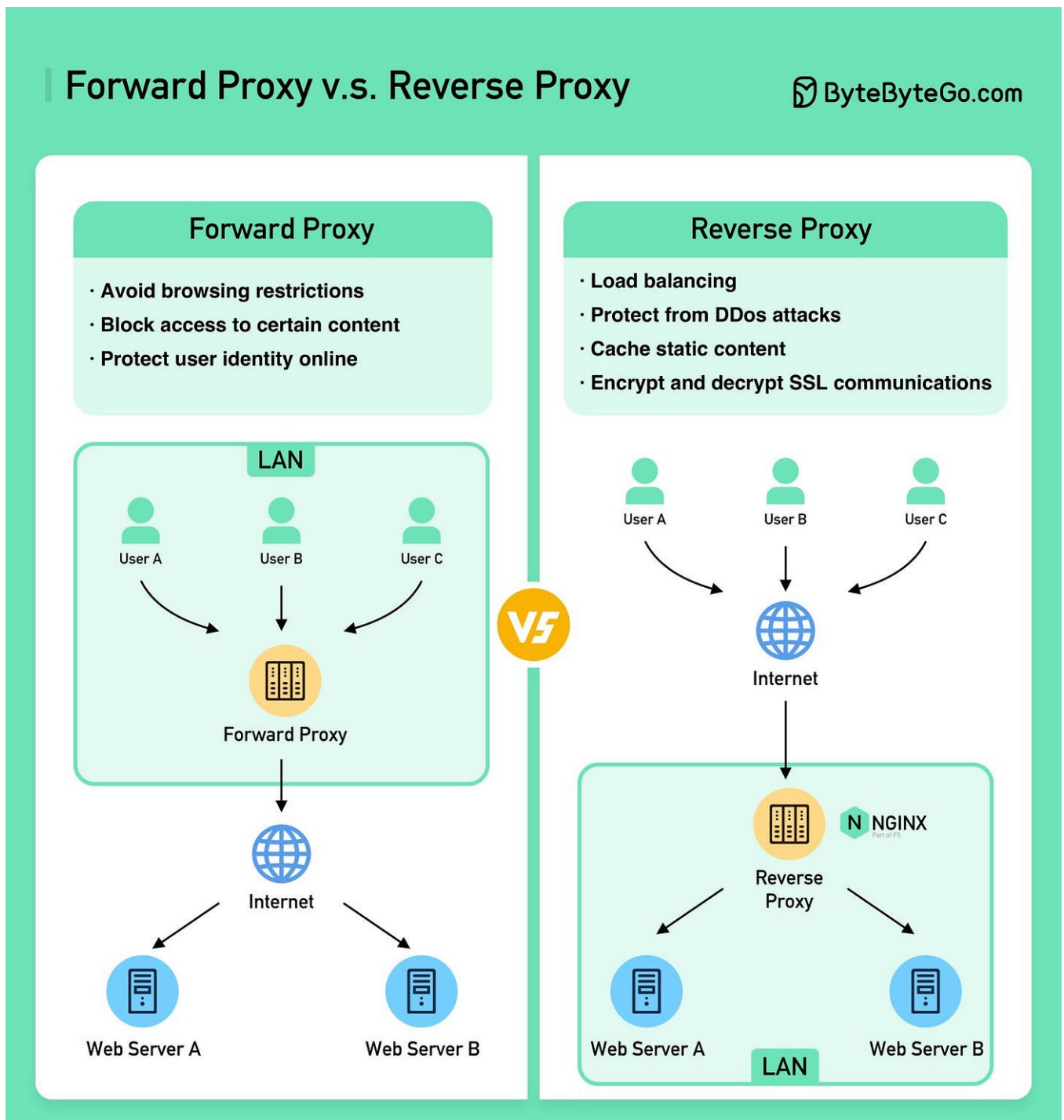
## **9. Scalability**

Reverse proxies can be used to scale web applications horizontally by distributing traffic across multiple servers. Forward proxies do not provide this scalability feature.

## 10. Network Complexity

Forward proxies are relatively simple to set up and manage, while reverse proxies can be more complex due to their load balancing, SSL/TLS termination, and caching features.

Here is a nice diagram I found online which also highlight the difference between Forward and Reverse Proxy





ByteByteGo is a great place to learn System Design, they also run a newsletter and have a YouTube channel, if you are preparing for System Design interview, you can also check that as a resource. Also, if you want to learn more about reverse proxy and forward proxy you can watch this YouTube video from their channel

## **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](#)



## Conclusion

In conclusion, **understanding the differences between forward proxy and reverse proxy is essential for designing and building scalable and secure systems.** While both proxies play important roles in facilitating communication between clients and servers, they differ in their functions, use cases, and pros and cons.

**Forward proxies are primarily used to hide a client's IP address** and to access restricted content, whereas **reverse proxies are used to protect and secure servers** and to distribute traffic across multiple servers. Ultimately, the choice between a forward proxy and reverse proxy will depend on the specific needs of the system and the desired outcome.

More importantly, this one is an important question for System design interviews and you must prepare for it. 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](#)**

### 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 System Design Articles you may like to read

#### Top 10 System Design Concepts Every Programmer should learn

Essential System Design Concepts for Programmers: 10 Key Principles for Building Scalable, Reliable, and...

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

#### 7 System Design Problems to Crack Software Engineering Interviews in 2023

Preparing for System design? Here are my favorite System Design Problems from interviews which you can use to practice

medium.com

### Top 10 Microservices Problem Solving Questions for 5 to 10 years experienced Developers

From Service Discovery to Security: 10 Problem-Solving Questions to Test the Microservices Skills of Developers with 5...

medium.com

System Design Interview

Microservices

Java

Programming

Software Engineering



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](https://medium.com/@somasharma_81597/membership)

## More from Soma and Javarevisited


Implementation	Provides its own implementation	Just a specification, requires an implementation	Builds on top, provides additional 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



### Data durability and consistency

The differences and impacts of failure rates of storage solutions and corruption rates in read-write processes

### Replication

Backing up data and repeating processes at scale

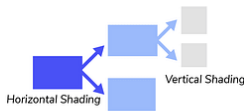


### Consensus

Ensuring all nodes are in agreement, which prevents fault processes from running and ensures consistency and replication of data and processes

### Partitioning

Dividing data across different nodes within systems, which reduces reliance on pure replication



### Distributed transactions

Once consensus is reached, transactions from

### HTTP

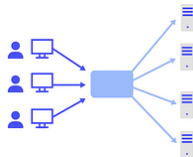
The API on which the entire internet runs

### REST

The set of design principles that directly interact with HTTP to enable system efficiency and scalability

### DNS and load balancing

Routing client requests to the right servers and the right tiers when processing happens to ensure system stability



### Caching

Making tradeoffs and caching decisions to determine what should be stored in a cache, how to direct traffic to a cache, and how to ensure we have the appropriate data in the cache

### N-tier applications

Understanding how processing tiers interact with each other and the specific process they control



### Step 1: Clarify the goals

Make sure you understand the basic requirements and ask any clarifying questions.

### Step 2: Determine the scope

Describe the feature set you'll be discussing in the given solution, and define all of the features and their importance to the end goal.

### Step 3: Design for the right scale

Determine the scale so you know whether the data can be supported by a single machine or if you need to scale.

### Step 4: Start simple, then iterate

Describe the high-level process end-to-end based on your feature set and overall goals. This is a good time to discuss potential bottlenecks.

### Step 5: Consider relevant DSA

Determine which fundamental data structures and algorithms will help your system perform efficiently and appropriately.



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

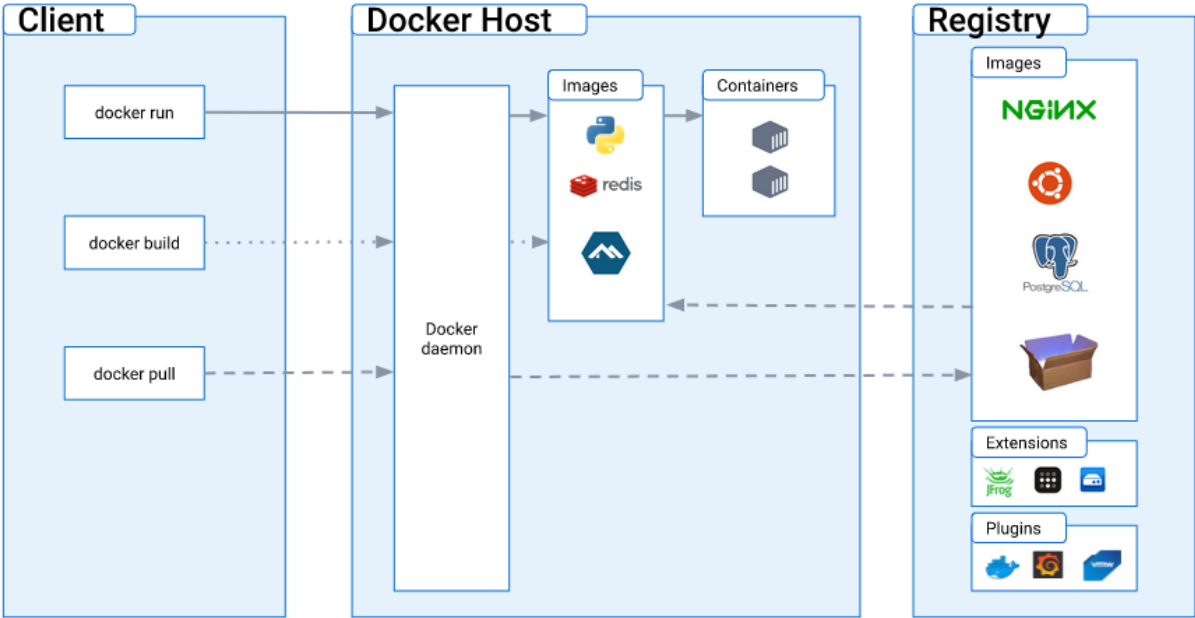


204



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





Rupert Waldron

## Create a non-blocking REST Api using Spring @Async and Polling

Get the great asynchronous, non-blocking experience you deserve with Spring's basic Rest Api, polling and @Async annotation.

12 min read · Feb 26



17



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



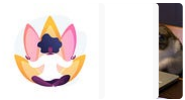
#### General Coding Knowledge

20 stories · 193 saves



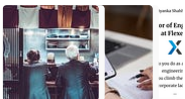
#### It's never too late or early to start something

13 stories · 67 saves



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

19 stories · 263 saves



#### Leadership

35 stories · 91 saves



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



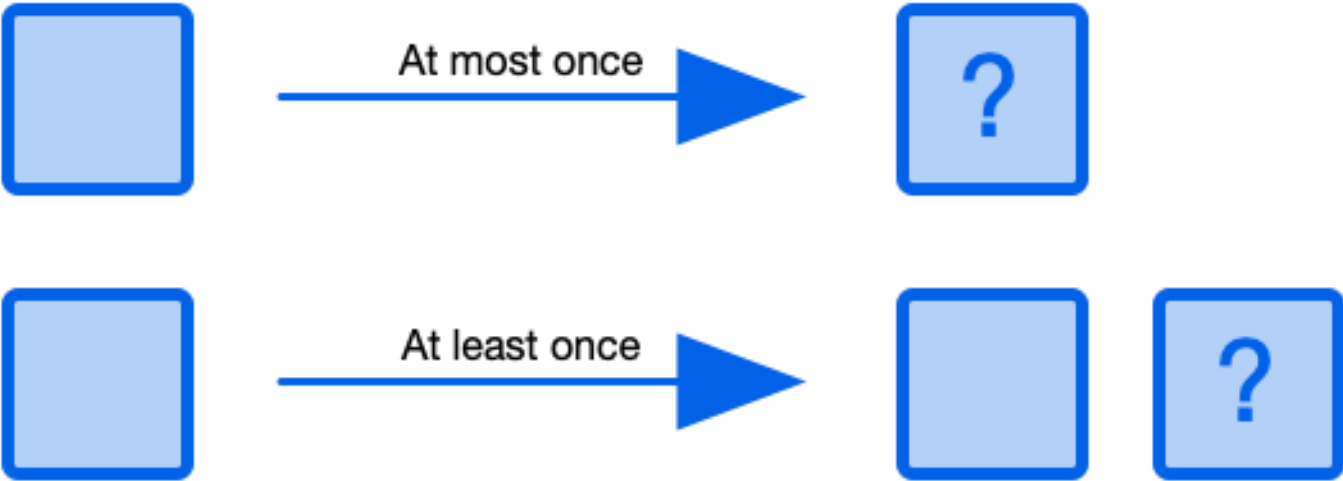
 Aparna Rathore

## Spring Boot with JPA and hibernate

Spring Boot is a popular Java framework that simplifies the development of production-ready applications. When combined with Java...

3 min read · Aug 5





 Andy Bryant

### Processing guarantees in Kafka

Each of the projects I’ve worked on in the last few years has involved a distributed message system such as AWS SQS, AWS Kinesis and more...

21 min read · Nov 16, 2019

 1.91K     5        




### APACHE IGNITE CACHING

WITH SPRING BOOT AND POSTGRESQL



A blog post by Virendra Oswal.



 Virendra Oswal

# Spring Boot Apache Ignite Caching: Boost Performance & Scale Effortless

Banner

10 min read · Jul 21



See more recommendations