# EP22: Latency numbers you should know. Also...

ALEX XU
SEP 3, 2022

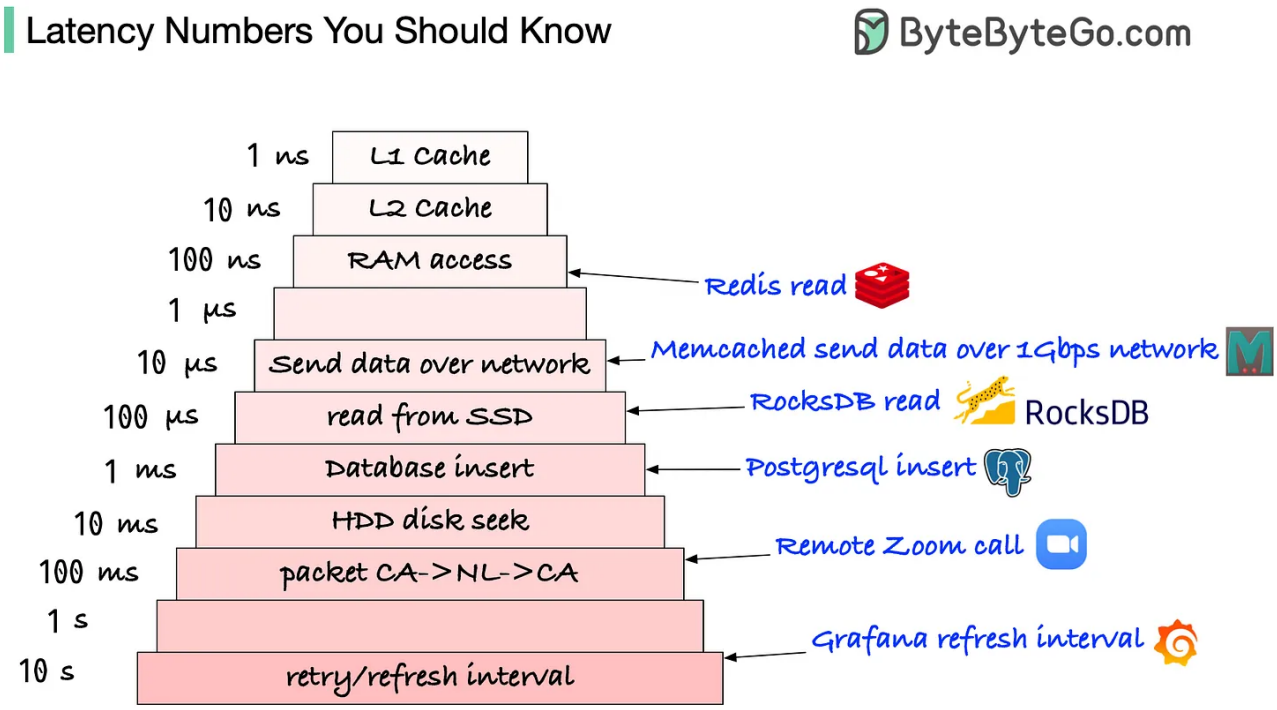


In this newsletter, we'll cover the following topics:

- Latency numbers you should know
- Microservice architecture
- Handling hotspot accounts
- E-commerce workflow
- The Secret Sauce Behind NoSQL: LSM Tree
- Sponsor ByteByteGo Newsletter

## Which latency numbers should you know?

Please note those are not accurate numbers. They are based on some online benchmarks (Jeff Dean's latency numbers + some other sources).




- **L1 and L2 caches: 1 ns, 10 ns**
  E.g.: They are usually built onto the microprocessor chip. Unless you work with hardware directly, you probably don't need to worry about them.
- **RAM access: 100 ns**
  E.g.: It takes around 100 ns to read data from memory. Redis is an in-memory data store, so it takes about 100 ns to read data from Redis.
- **Send 1K bytes over 1 Gbps network: 10 us**
  E.g.: It takes around 10 us to send 1KB of data from Memcached through the network.
- **Read from SSD: 100 us**
  E.g.: RocksDB is a disk-based K/V store, so the read latency is around 100 us on SSD.
- **Database insert operation: 1 ms.**
  E.g.: Postgresql commit might take 1ms. The database needs to store the data, create the index, and flush logs. All these actions take time.
- **Send packet CA->Netherlands->CA: 100 ms**
  E.g.: If we have a long-distance Zoom call, the latency might be around 100 ms.
- **Retry/refresh internal: 1-10s**
  E.g: In a monitoring system, the refresh interval is usually set to 5~10 seconds (default value on Grafana).

**Notes**

1 ns = 10^-9 seconds

1 us = 10^-6 seconds = 1,000 ns

1 ms = 10^-3 seconds = 1,000 us = 1,000,000 ns

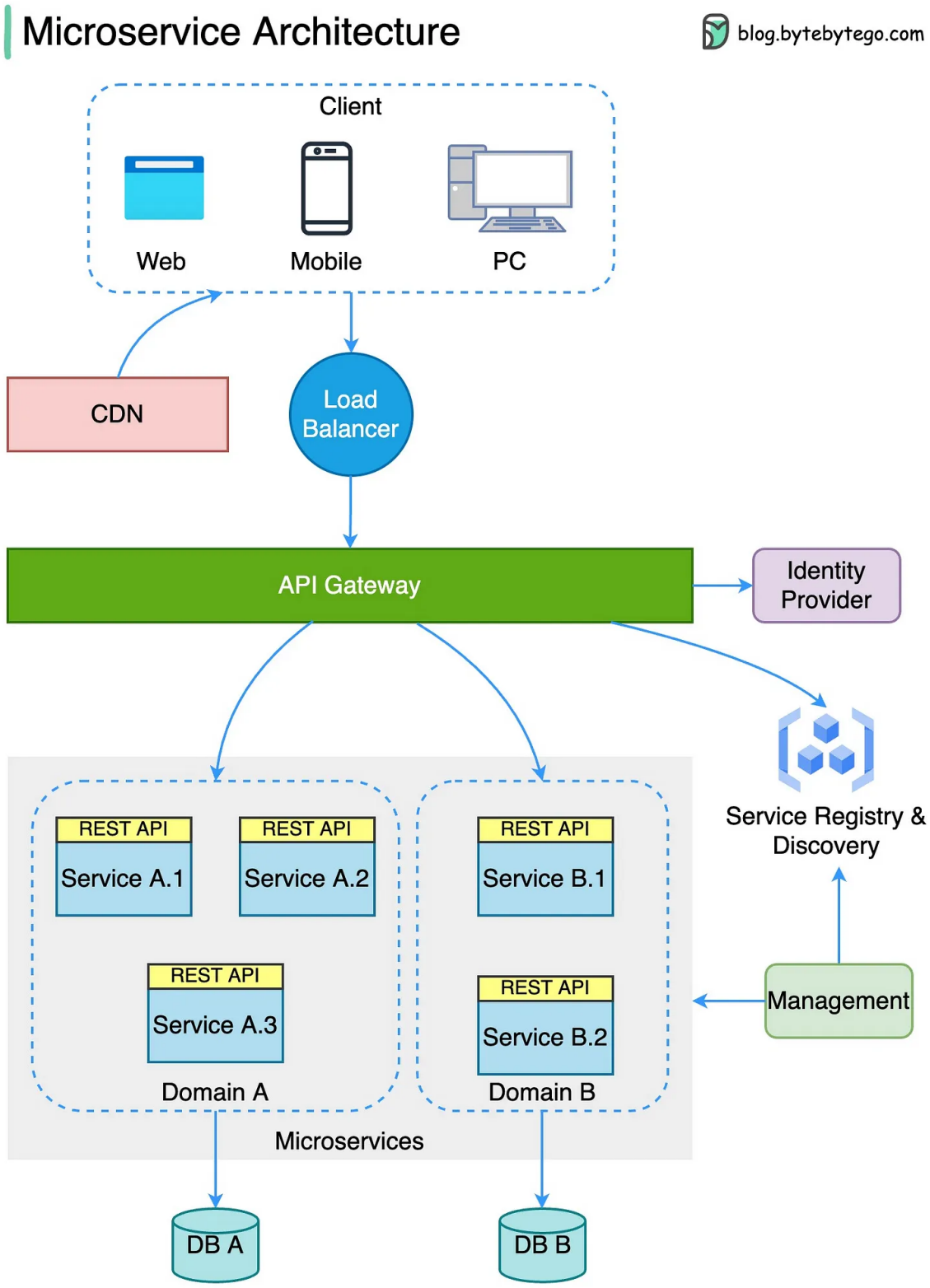
Quick quiz:

1). Do you know all?

2). Nowadays, disk and tape are used as data backup. Do you know which one has a higher write speed?

## What does a typical microservice architecture look like?

The diagram below shows a typical microservice architecture.




- **Load Balancer:** This distributes incoming traffic across multiple API gateway instances for high availability.
- **CDN (Content Delivery Network):** CDN is a group of geographically distributed servers that hold static content for faster delivery. The clients look for content in CDN first, then progress to backend services.

- **API Gateway:** This handles incoming requests and routes them to the relevant services. It talks to the identity provider and service discovery.
- **Identity Provider:** This handles authentication and authorization for users.
- **Service Registry & Discovery:** Microservice registration and discovery happen in this component, and the API gateway looks for relevant services in this component to talk to.
- **Management:** This component is responsible for monitoring the services.
- **Microservices:** Microservices are designed and deployed in different domains. Each domain has its own database. The API gateway talks to the microservices via REST API or other protocols, and the microservices within the same domain talk to each other using RPC (Remote Procedure Call).

Benefits of microservices:
- They can be quickly designed, deployed, and horizontally scaled.
- Each domain can be independently maintained by a dedicated team.
- Business requirements can be customized in each domain and better supported, as a result.

Quick questions:
1). What are the drawbacks of the microservice architecture?
2). Have you seen a monolithic system be transformed into microservice architecture? How long does it take?

## Handling Hotspot Accounts

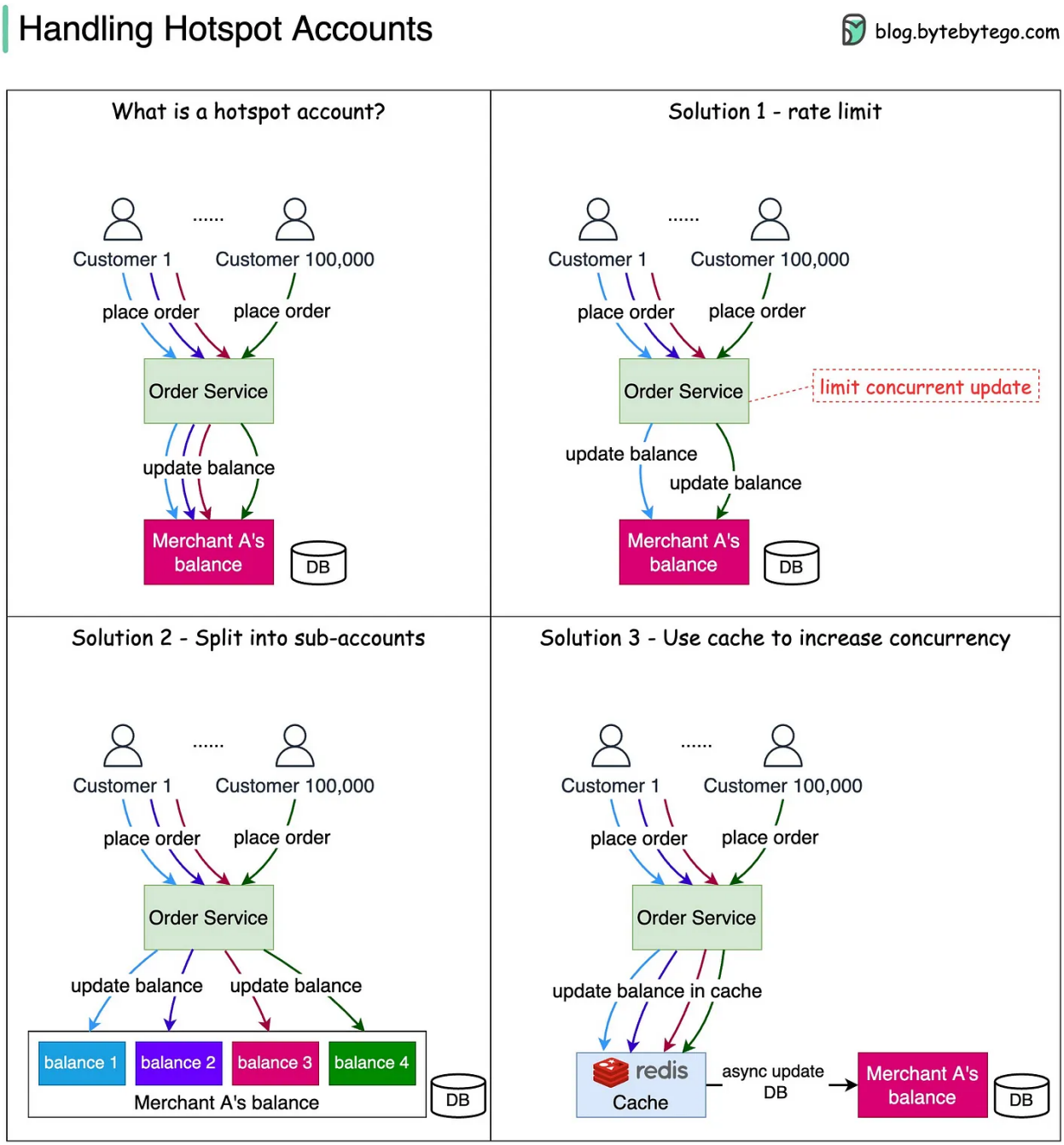
Big accounts, such as Nike, Procter & Gamble & Nintendo, often cause hotspot issues for the payment system.

A hotspot payment account is an account that has a large number of concurrent operations on it.

For example, when merchant A starts a promotion on Amazon Prime day, it receives many concurrent purchasing orders. In this case, the merchant's account in the database becomes a hotspot account due to frequent updates.

In normal operations, we put a row lock on the merchant's balance when it gets updated. However, this locking mechanism leads to low throughput and becomes a system bottleneck.

The diagram below shows several optimizations.



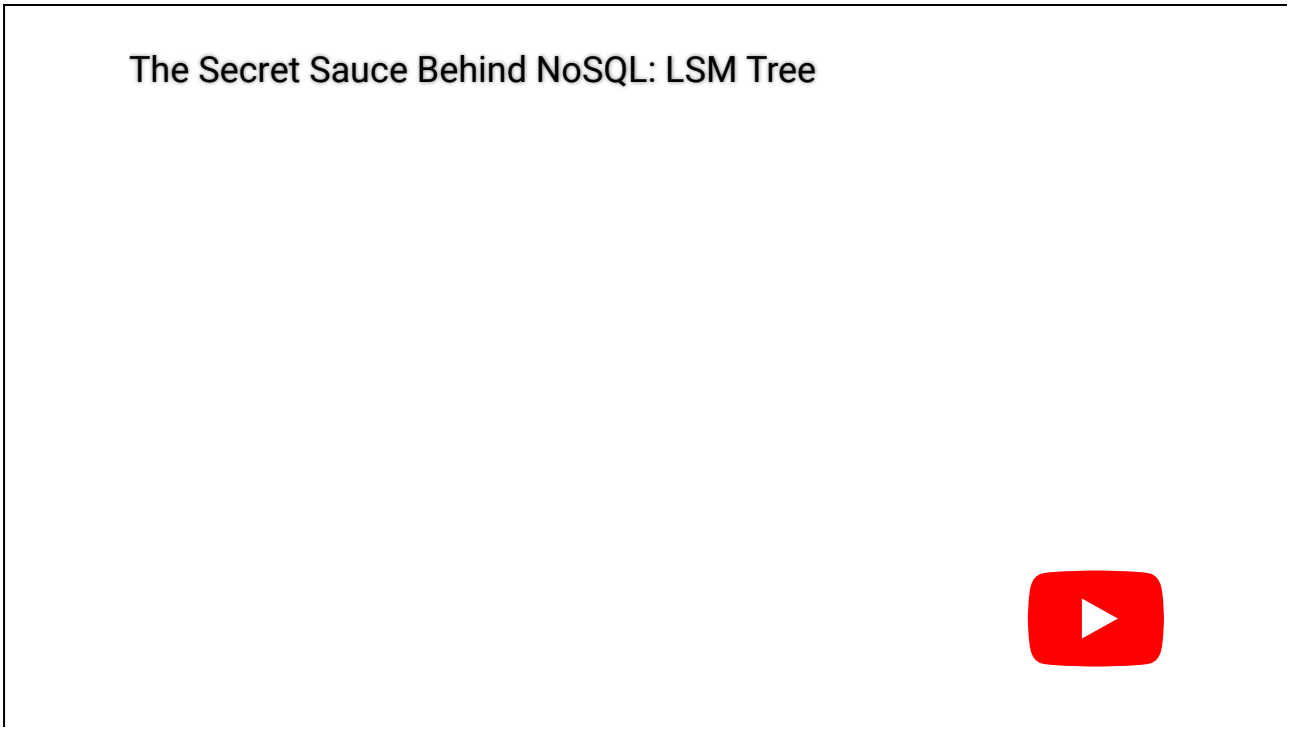

- **Rate limit**
  We can limit the number of requests within a certain period. The remaining requests will be rejected or retried at a later time. It is a simple way to increase

the system's responsiveness for some users, but this can lead to a bad user experience.

- **Split the balance account into sub-accounts**
  We can set up sub-accounts for the merchant's account. In this way, one update request only locks one sub-account, and the rest sub-accounts are still available.
- **Use cache to update balance first**
  We can set up a caching layer to update the merchant's balance. The detailed statements and balances are updated in the database later asynchronously. The in-memory cache can deal with a much higher throughput than the database.

Quick question: We can also put the requests into a message queue so the requests can be processed at the service's own pace. Can you think of the limitations of this approach?

# The Secret Sauce Behind NoSQL: LSM Tree

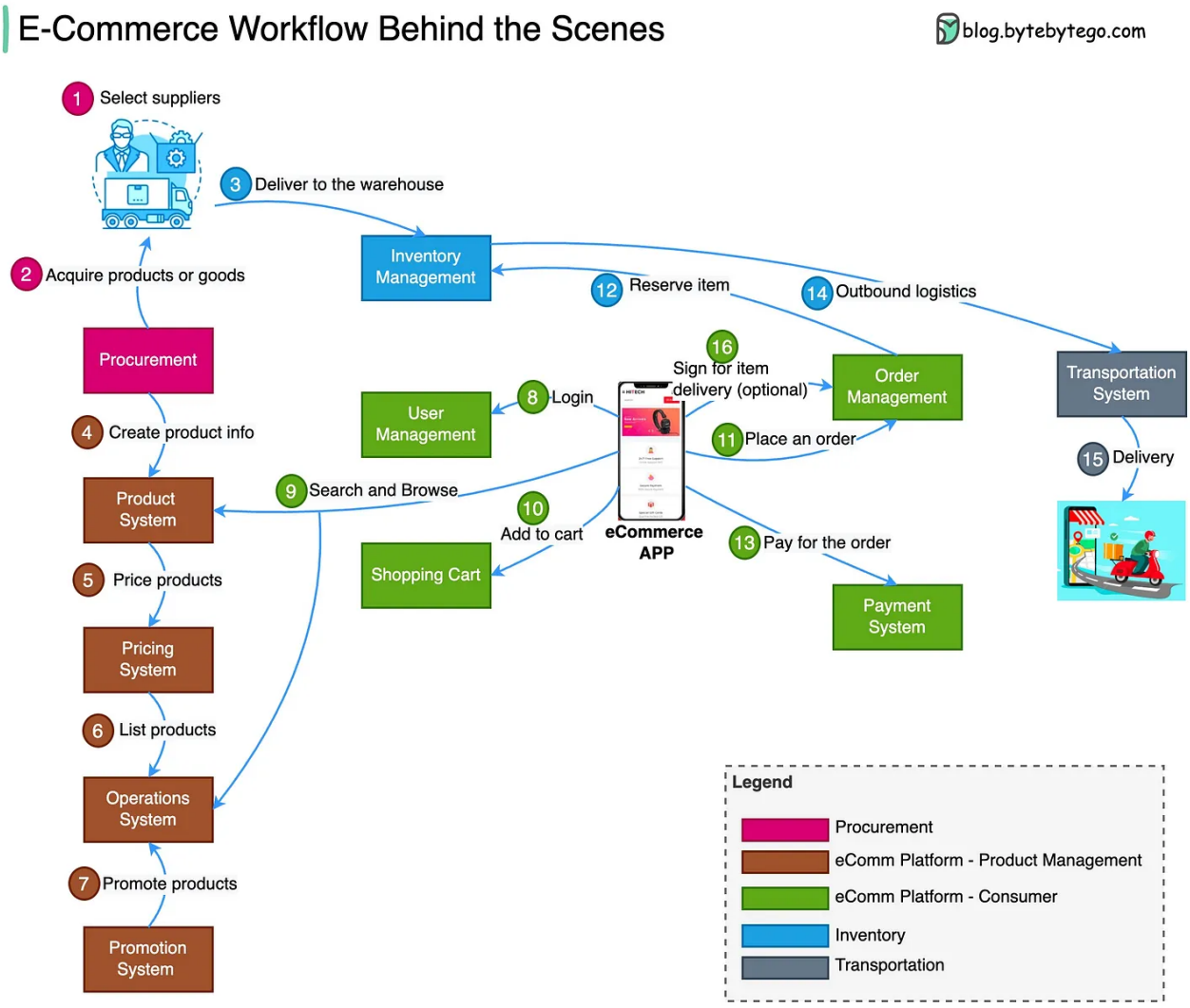



# E-commerce Workflow

What happens behind the scenes when we shop online?

Disclaimer: I have limited knowledge of the eCommerce system. The diagram below is based on my research. Please suggest better names for the components or let me know if you spot an error.

The diagram below shows the 4 key business areas in a typical e-commerce company: procurement, inventory,  eComm platform, and transportation.




## 1 Procurement

Step 1 - The procurement department selects suppliers and manages contracts with them.

Step 2 - The procurement department places orders with suppliers, manages the return of goods, and settles invoices with suppliers.

**2️⃣ Inventory**

Step 3 - The products or goods from suppliers are delivered to a storage facility. All products/goods are managed by inventory management systems.

**3️⃣ eComm platform**

Steps 4-7 - The “eComm platform - Product Management” system creates the product info managed by the product system. The pricing system prices the products. Then the products are ready to be listed for sale. The promotion system defines big sale activities, coupons, etc.

Step 8-11 - Consumers can now purchase products on the e-commerce APP. First, users register or log in to the APP. Next, users browse the product list and details, adding products to the shopping cart. They then place purchasing orders.

Steps 12,13 - The order management system reserves stock in the inventory management system. Then the users pay for the product.

**4️⃣ Transportation**

Steps 14,15 - The inventory system sends the outbound order to the transportation system, which manages the physical delivery of the goods.

Step 16 - Sign for item delivery (optional)

Quick question: If a user buys many products, their big order might be divided into several small orders based on warehouse locations, product types, etc. Where would you place the “order splitting” system in the process outlined below?

## We are looking for sponsors!

To make it a sustainable business, I'd like to explore monetizing options for the newsletter. We will be opening **1 ad slot** per week.

We created a Sponsor Kit for the newsletter (100K+ subscribers). If your business is interested, please contact **hi@bytebytego.com**.

186 Likes

## 13 Comments

Write a comment...

**Michael** Nov 20, 2022

Thanks for sharing. But for microservice architecture, it may not be accurate. The LB is usually behind the gateway, or is coarse LB fleet front of gateway, or one LB for one service behind gateway.

LIKE REPLY SHARE

**Quan Huynh** Oct 26, 2022

Cool

LIKE REPLY SHARE

**11 more comments...**