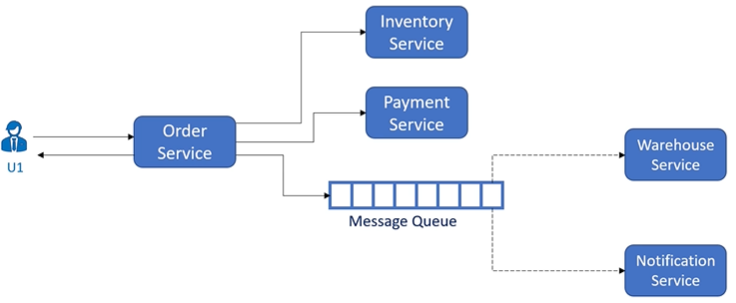
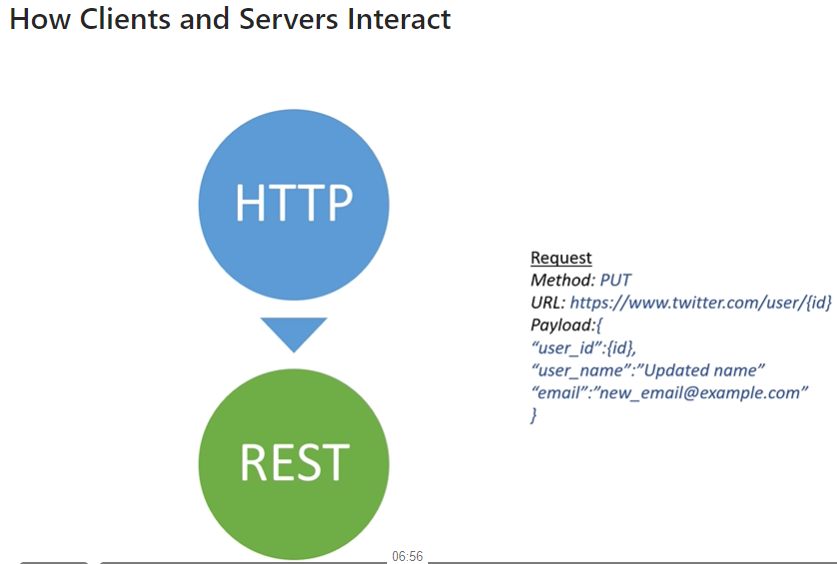
**Kafka** is a message queue system. Formally it is a stream processing system used for messaging, website activity tracking, metrics collection and monitoring, logging, event sourcing, commit logs, and real-time analytics. It’s a good fit for large scale message processing applications since it is more robust, reliable, and fault-tolerant compared to traditional message queues.

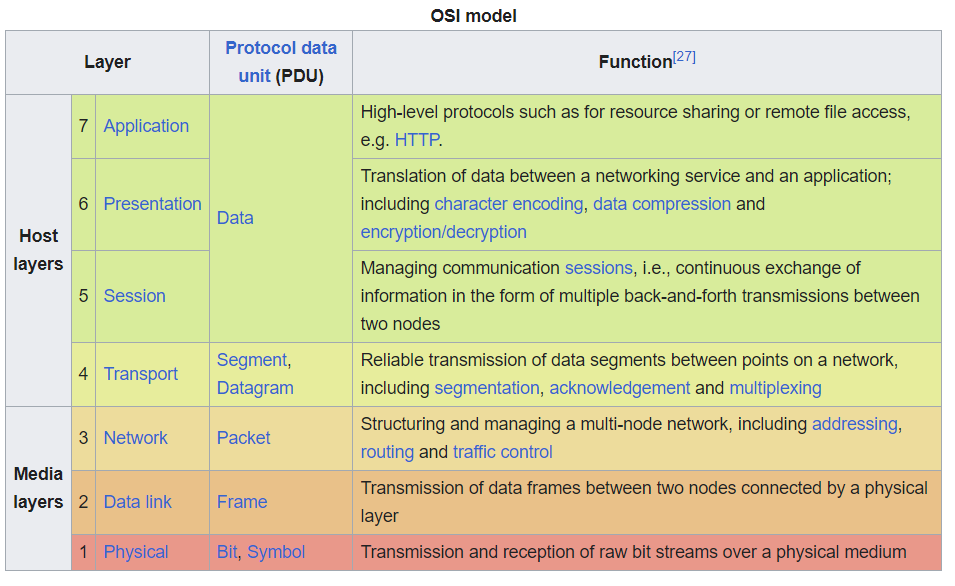


When Inventory Service, Payment Service are finished, Order Service publishes a notification in the queue. Both Warehouse and Notification Service keep reading the message queue. If they find something that demands their actions, they act, and mark/update/delete the messages.

HTTPS and REST APIs:



REST APIs look like a dictionary and is probably implemented as a hashmap. HTTPS is used for one directional client-to-server request, e.g. visiting the websites of Google or Amazon. It can be used for bi-directional, open and persistent connection, e.g. chat service. For this, we need web sockets.



WebSockets is built upon TCP which is low-level, bi-directional, full-duplex and guaranteed order transport layer. Unlike TCP, which is a streaming transport, WebSockets is a message-based transport: messages are delimited on the wire and are re-assembled in-full before delivery to the application.

Consistent hashing: consider an array of machine IDs of size N. Replicate this array M (e.g. 100) times and concatenate the replicas. Shuffle the large array at random. Denote the array by A. Assume x is the hash value of a data blob. Then the data should be stored on the machine with ID A[x % (N \* M)].

If a new machine is added, select N slots in A[] at random, and change their values to the new machine’s ID. So when a new data blob comes in, it has some chance to be stored on the new machine.

Before deletion of a machine, data blobs on the machine needs to be redistributed onto other machines. We identify all the slots in A[] with the to-be-deleted machine’s ID, and assign IDs of other machines at random. Then, we distribute the data blobs to those machines following some strategy, and change the key-value store.

Caches: Redis. Whatever system you design, some caching solution is probably required. Cache: key-value store.

**Database**:

Cache: redis.

Blobs storage: e.g. image and videos: Amazon S3.

CDN: content delivery network when using blobs storage like S3. CDN is generally used to distribute the same data blob geographically in a lot of locations. For example, you open an online bookstore and store all book images on Amazon S3, then you would need the CDN to deliver the image to your own servers around the globe to reduce the time for customers’ queries. Note that Amazon S3 are in only a handful of data centers around the globe. Amazon S3 + CDN is quite final for solutions to blob storage.

Text search : Elastic search/solar, which are able to perform fuzzy search. A search engine is not a database and the data fed to them won’t be preserved.

Time series database: OpenTSDB (open time series database) use it when sequential updates dominate. Regular relational database allows you to update random records, but time series database is optimized for read and write blocks of sequential data.

Datawarehouse : Hadoop, offline analytics over data generated/received from different parts of the whole company. Provide all kinds of querying capabilities for serving many kinds of reports. Not used for transactional system.

RDBMS: for structured data, for transactional applications, and for the need of ACID. If you don’t need ACID, RDBMS won’t hurt since synchronization cost disappears automatically.

Document database: e.g., Mongo DB, for nonstructured data where items can have different attributes, and for complex queries against the DBs. This type of database is optimized for queries on, e.g., JSON records. Elastic search is a special case of document DB.

Column-oriented database: for ever-increasing data, or for a large amount of data but simple queries such. While a relational database is optimized for storing rows of data, typically for transactional applications, a columnar database is optimized for fast retrieval of columns of data, typically in analytical applications. Column-oriented storage for database tables is an important factor in analytic query performance because it drastically reduces the overall disk I/O requirements and reduces the amount of data you need to load from disk.

**Mix them up**: for designing Amazon, you can use RDBMS to store the orders that have been placed but not yet been delivered to the customer. Once it’s delivered, you can move the record into Cassandra/Hbase (based on hadoop) for data warehousing.

Difference between Hadoop and Cassandra: the former has a master-based architecture while the latter has a masterless one, so HBase comes with a single failure point, while Cassandra does not.

Rate-limiting : leaky bucket, fixed window, sliding window (accurate but costly).