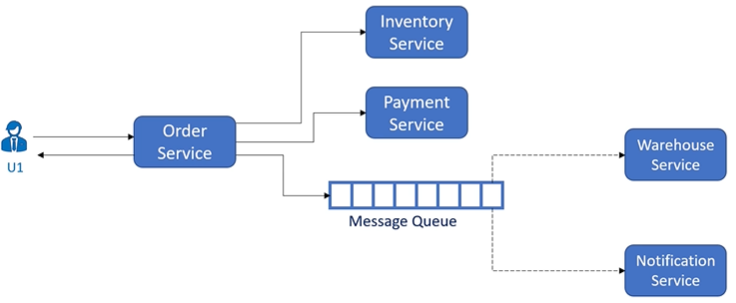
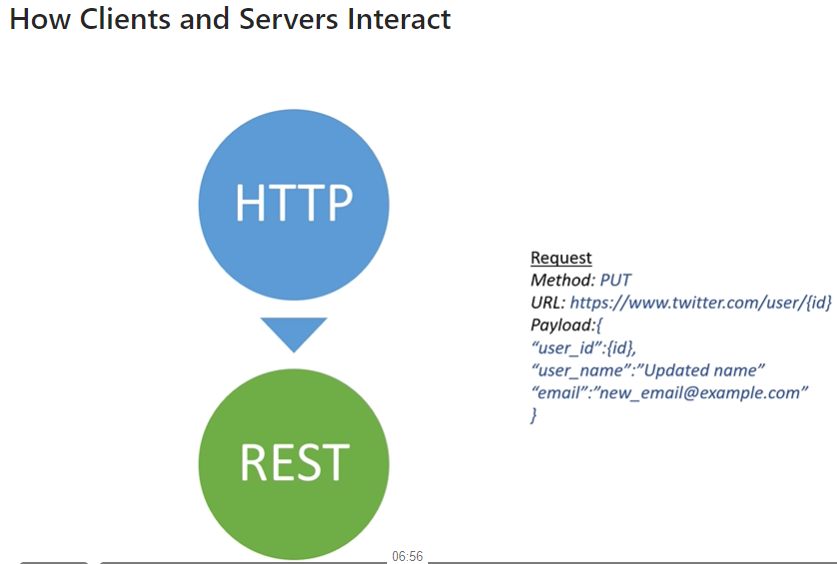
**Kafka** is a message queue system. Formally it is a stream processing system used for messaging, website activity tracking, metrics collection and monitoring, logging, event sourcing, commit logs, and real-time analytics. It’s a good fit for large scale message processing applications since it is more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

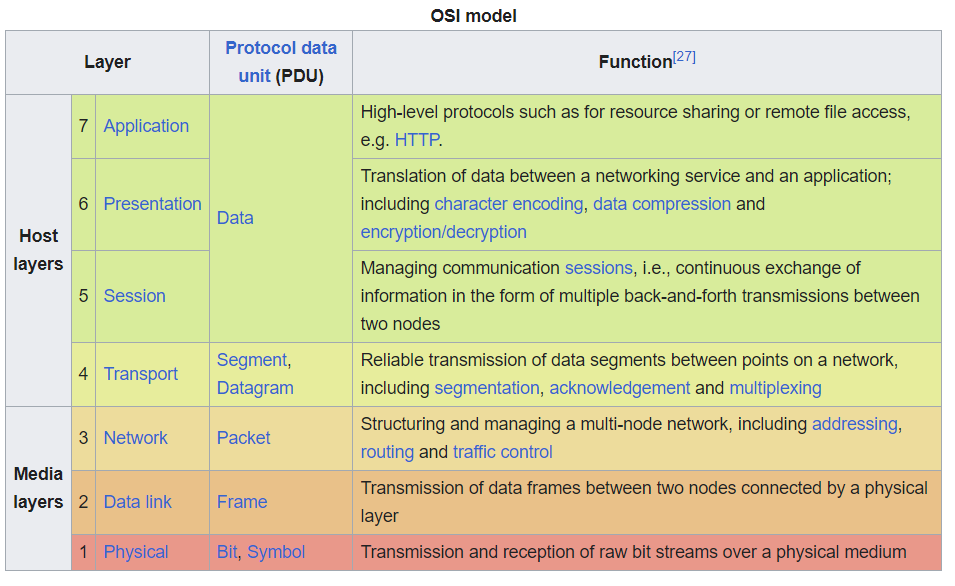


When Inventory Service, Payment Service are finished, Order Service publishes a notification in the queue. Both Warehouse and Notification Service keep reading the message queue. If they find something that demands their actions, they act, and mark/update/delete the messages.

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.