# Microservices

**Definition**: - Microservices architecture (often shortened to microservices) refers to an architectural style for developing applications. Microservices allow a large application to be separated into smaller independent parts, with each part having its own realm of responsibility.

**Communication in a microservice architecture:**

Client and services can communicate through many different types of communication, each one targeting a different scenario and goals. Initially, those types of communications can be classified in two axes.

The first axis defines if the protocol is synchronous or asynchronous:

* Synchronous protocol. HTTP is a synchronous protocol. The client sends a request and waits for a response from the service. That's independent of the client code execution that could be synchronous (thread is blocked) or asynchronous (thread isn't blocked, and the response will reach a callback eventually). The important point here is that the protocol (HTTP/HTTPS) is synchronous and the client code can only continue its task when it receives the HTTP server response.
* Asynchronous protocol. Other protocols like AMQP (a protocol supported by many operating systems and cloud environments) use asynchronous messages. The client code or message sender usually doesn't wait for a response. It just sends the message as when sending a message to a RabbitMQ queue or any other message broker.

The second axis defines if the communication has a single receiver or multiple receivers:

* Single receiver. Each request must be processed by exactly one receiver or service. An example of this communication is the [Command pattern](https://en.wikipedia.org/wiki/Command_pattern).
* Multiple receivers. Each request can be processed by zero to multiple receivers. This type of communication must be asynchronous. An example is the [publish/subscribe](https://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern) mechanism used in patterns like [Event-driven architecture](https://microservices.io/patterns/data/event-driven-architecture.html). This is based on an event-bus interface or message broker when propagating data updates between multiple microservices through events; it's usually implemented through a service bus or similar artifact like [Azure Service Bus](https://azure.microsoft.com/services/service-bus/) by using [topics and subscriptions](https://learn.microsoft.com/en-us/azure/service-bus-messaging/service-bus-dotnet-how-to-use-topics-subscriptions).

A microservice-based application will often use a combination of these communication styles. The most common type is single-receiver communication with a synchronous protocol like HTTP/HTTPS when invoking a regular Web API HTTP service. Microservices also typically use messaging protocols for asynchronous communication between microservices.

