**Monolithic Architecture**

**Service-Oriented Architecture (SOA)**

* Uses reusable, loosely coupled services, often with an Enterprise Service Bus (ESB).
* Heavier than microservices; used in enterprise environments.

**Microservices Architecture**

* Application split into small, independently deployable services communicating over the network.
* Typically uses REST, gRPC, or messaging systems.

**Event-Driven Architecture (EDA)**

* Components communicate via events through message brokers (Kafka, RabbitMQ, NATS).
* Enables asynchronous, decoupled systems.

**Peer-to-Peer (P2P) Architecture**

* Each node can act as both client and server.
* Used in torrenting, decentralized systems.

Tool used for that:

Quick reminder about socket

REST

GraphQL

RPC

gRPC

**What is Remote Procedure Call (RPC)?**

**The RPC protocol enables users to work with remote procedures as if the procedures were local.**

Remote Procedure Call (RPC) is a type of technology used in computing to enable a program to request a service from software located on another computer in a network without needing to understand the network's details. RPC abstracts the complexities of the network by allowing the developer to think in terms of function calls rather than network details, facilitating the process of making a piece of software distributed across different systems.

RPC works by allowing one program (a client) to directly call procedures (functions) on another machine (the server). The client makes a procedure call that appears to be local but is run on a remote machine. When an RPC is made, the calling arguments are packaged and transmitted across the network to the server. The server unpacks the arguments, performs the desired procedure, and sends the results back to the client

1. A client invokes a client stub procedure, passing parameters in the usual way. The client stub resides within the client's own address space.

2. The client stub marshalls(pack) the parameters into a message. Marshalling includes converting the representation of the parameters into a standard format, and copying each parameter into the message.

3. The client stub passes the message to the transport layer, which sends it to the remote server machine.  On the server, the transport layer passes the message to a server stub, which **demarshalls(unpack)** the parameters and calls the desired server routine using the regular procedure call mechanism.

4. When the server procedure completes, it returns to the server stub **(e.g., via a normal procedure call return)**, which marshalls the return values into a message.

5. The server stub then hands the message to the transport layer. The transport layer sends the result message back to the client transport layer, which hands the message back to the client stub.

6. The client stub demarshalls the return parameters and execution returns to the caller.

A GraphQL API is analogous to publishing the back-end’s database schema and offering clients an SQL console: SELECT \* FROM table1 JOIN table2 JOIN ...