

Lecture #1

WSMT 2022-2023

Introduction to Web Services

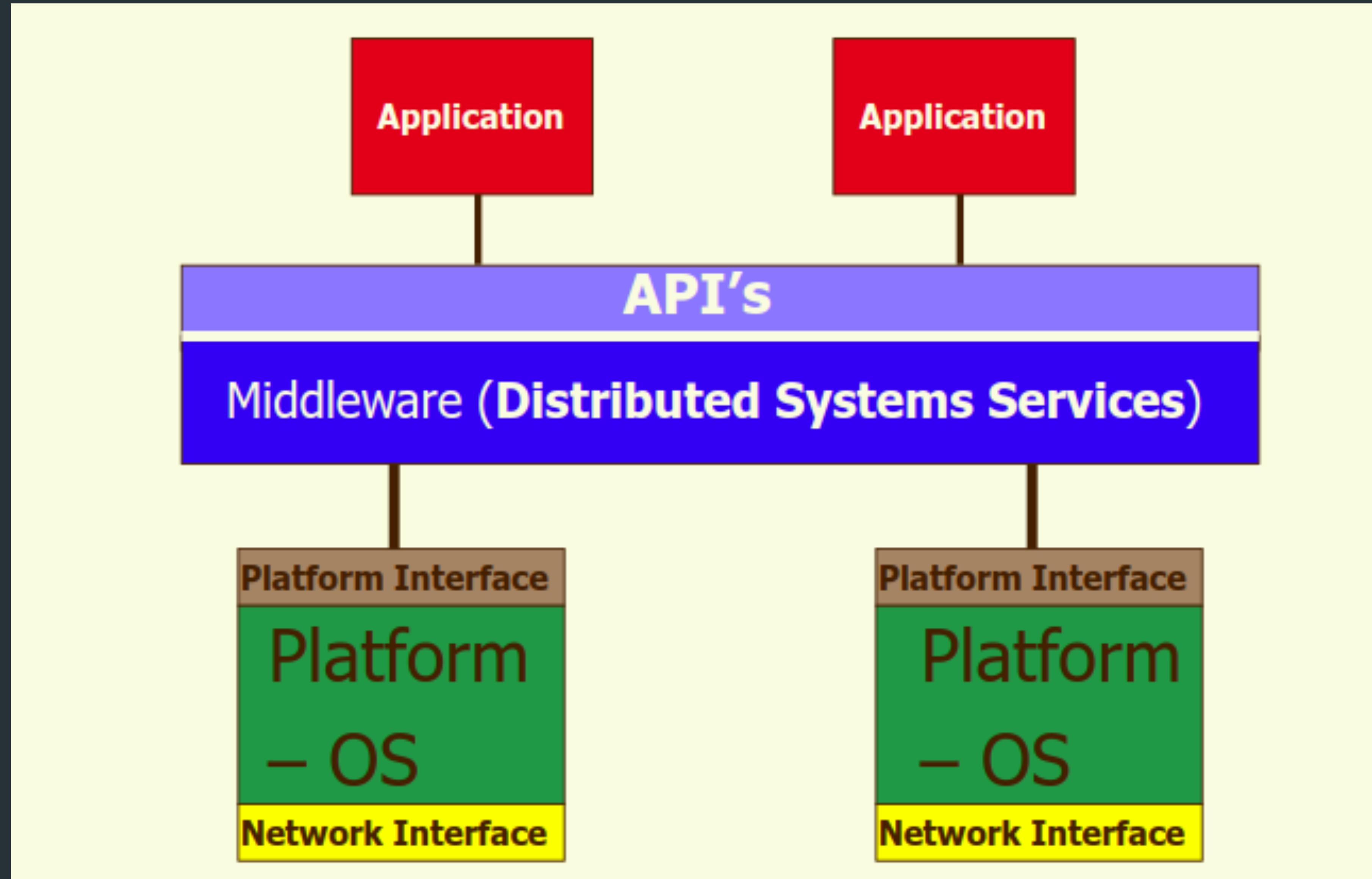
What are Web Services?

What is Web Services?

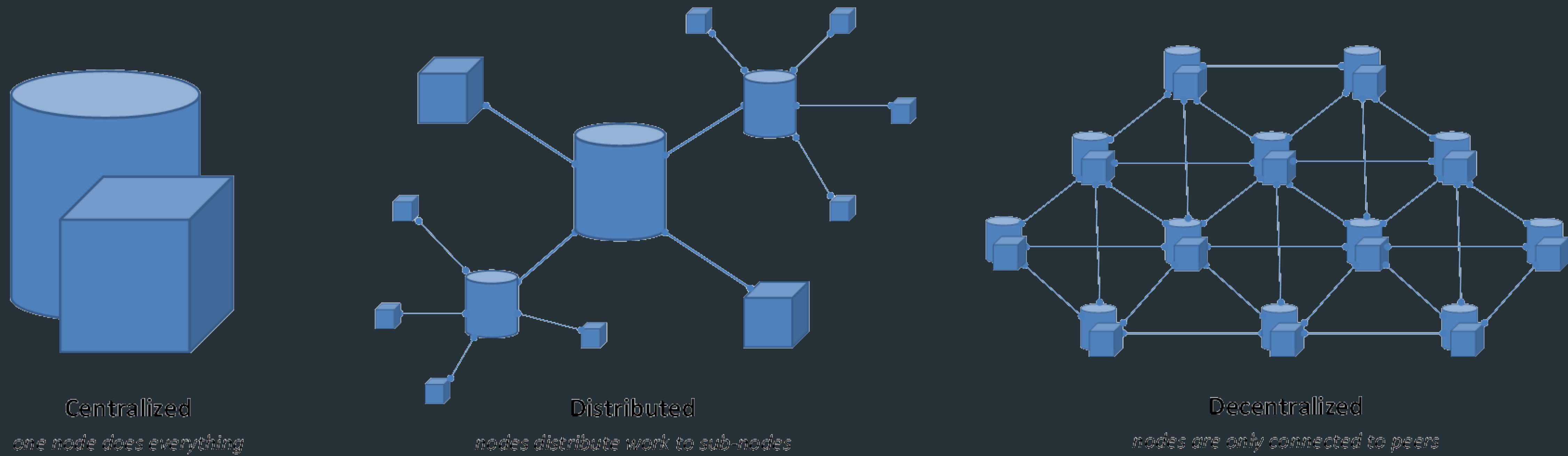


Introduction to Middleware Technologies

What is Middleware?



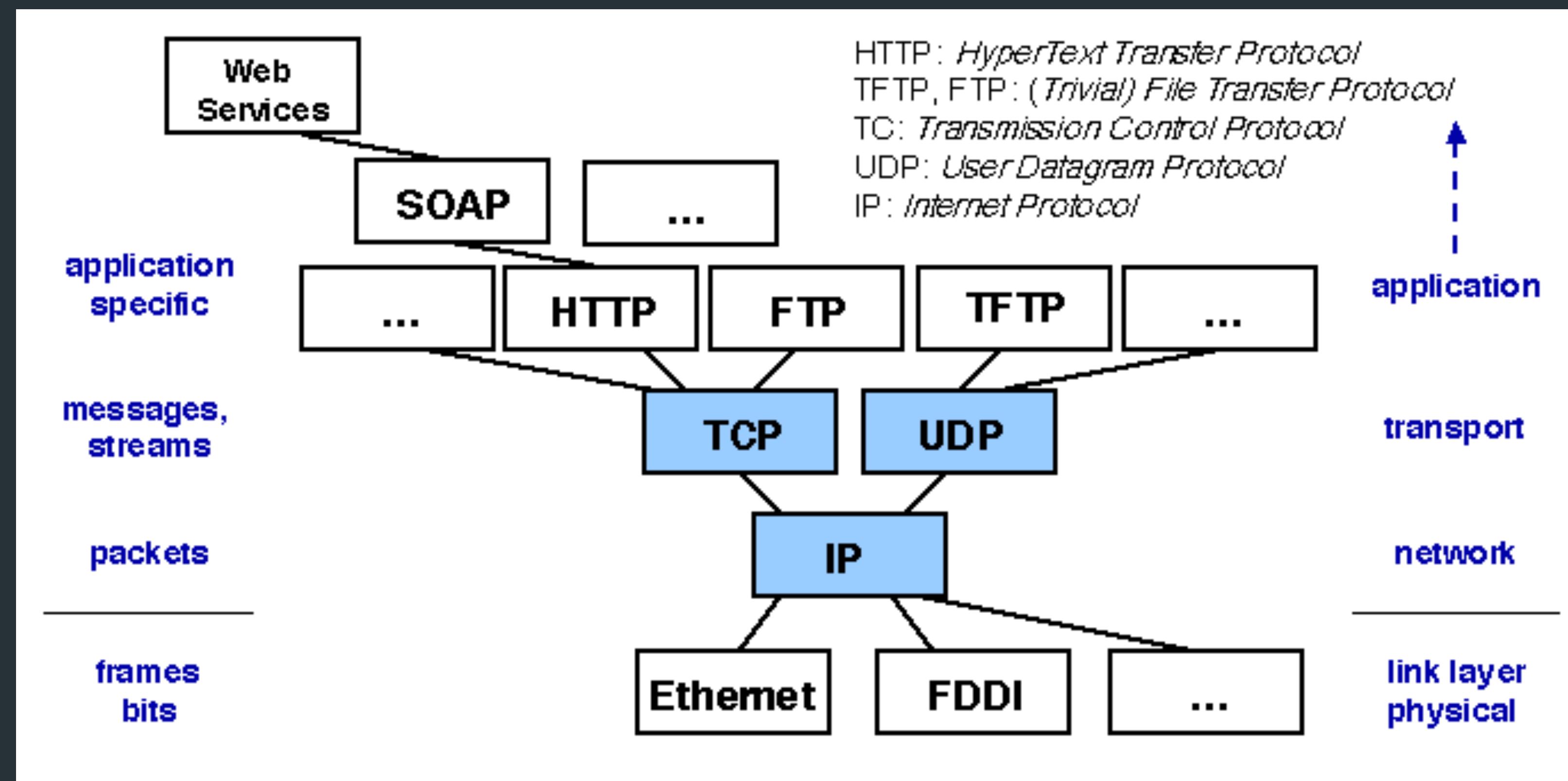
Distributed Systems



Distributed Systems

Understanding Middleware for Distributed Systems

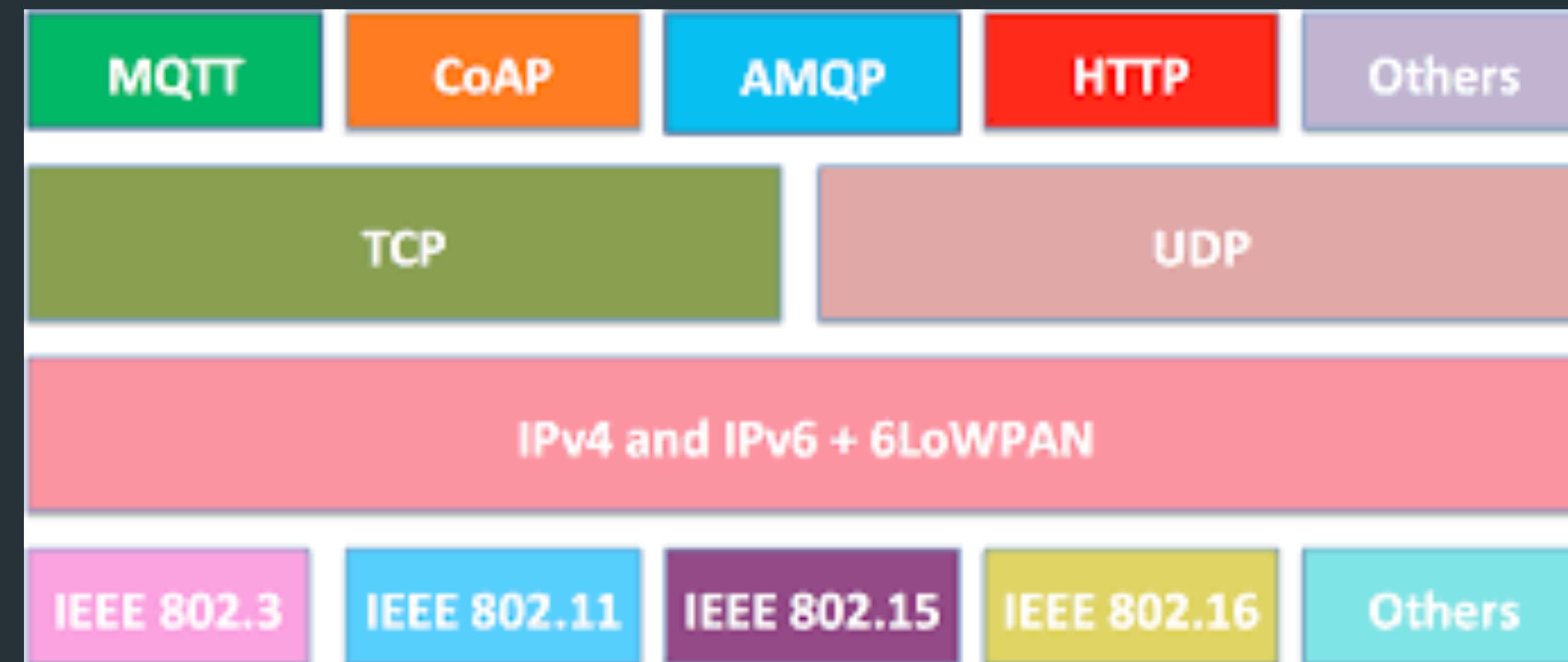
Communication Protocols Used in Middleware



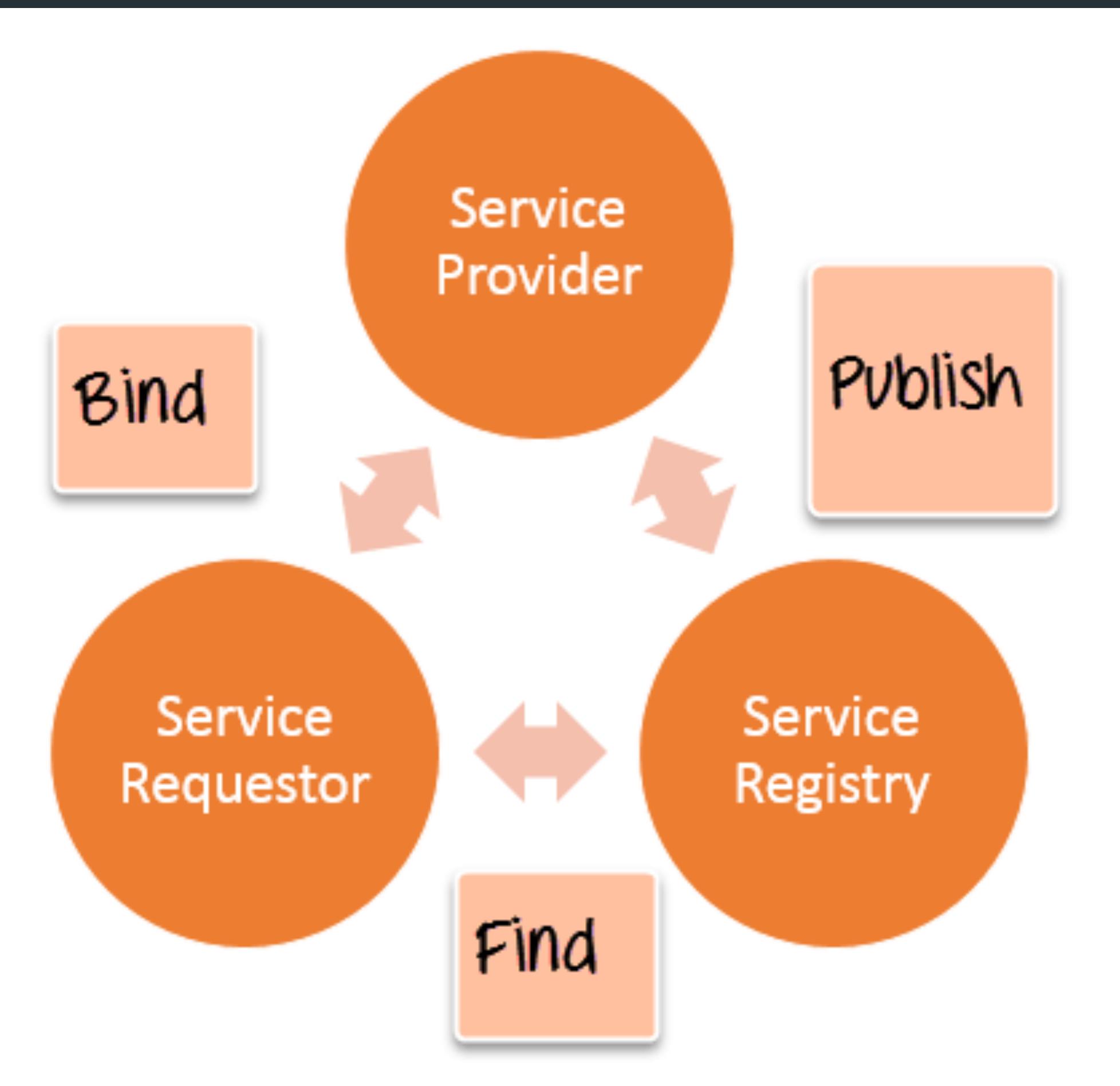
Commonly Used Communication Protocols in Middleware

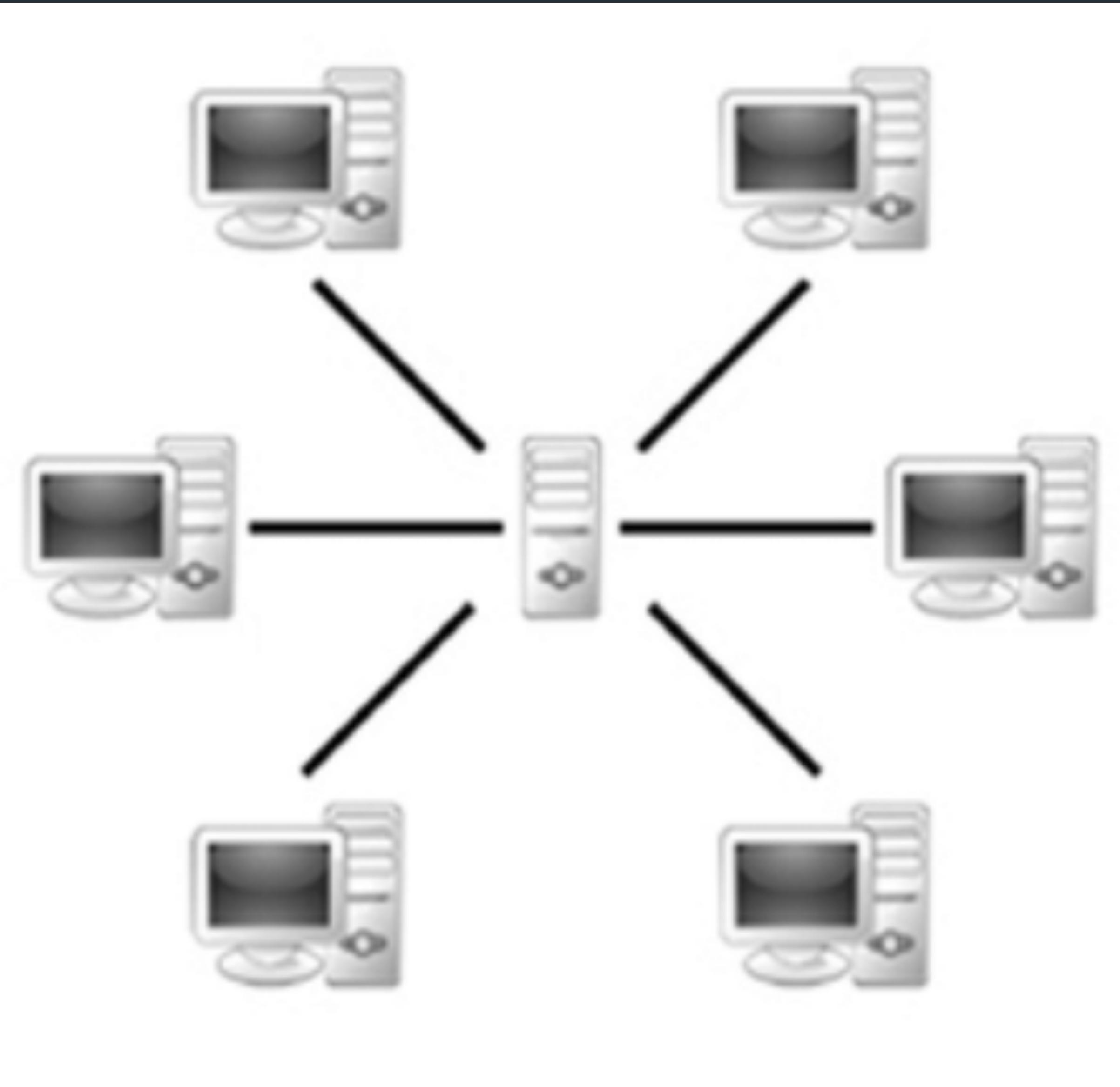
Commonly Used Communication Protocols in Middleware

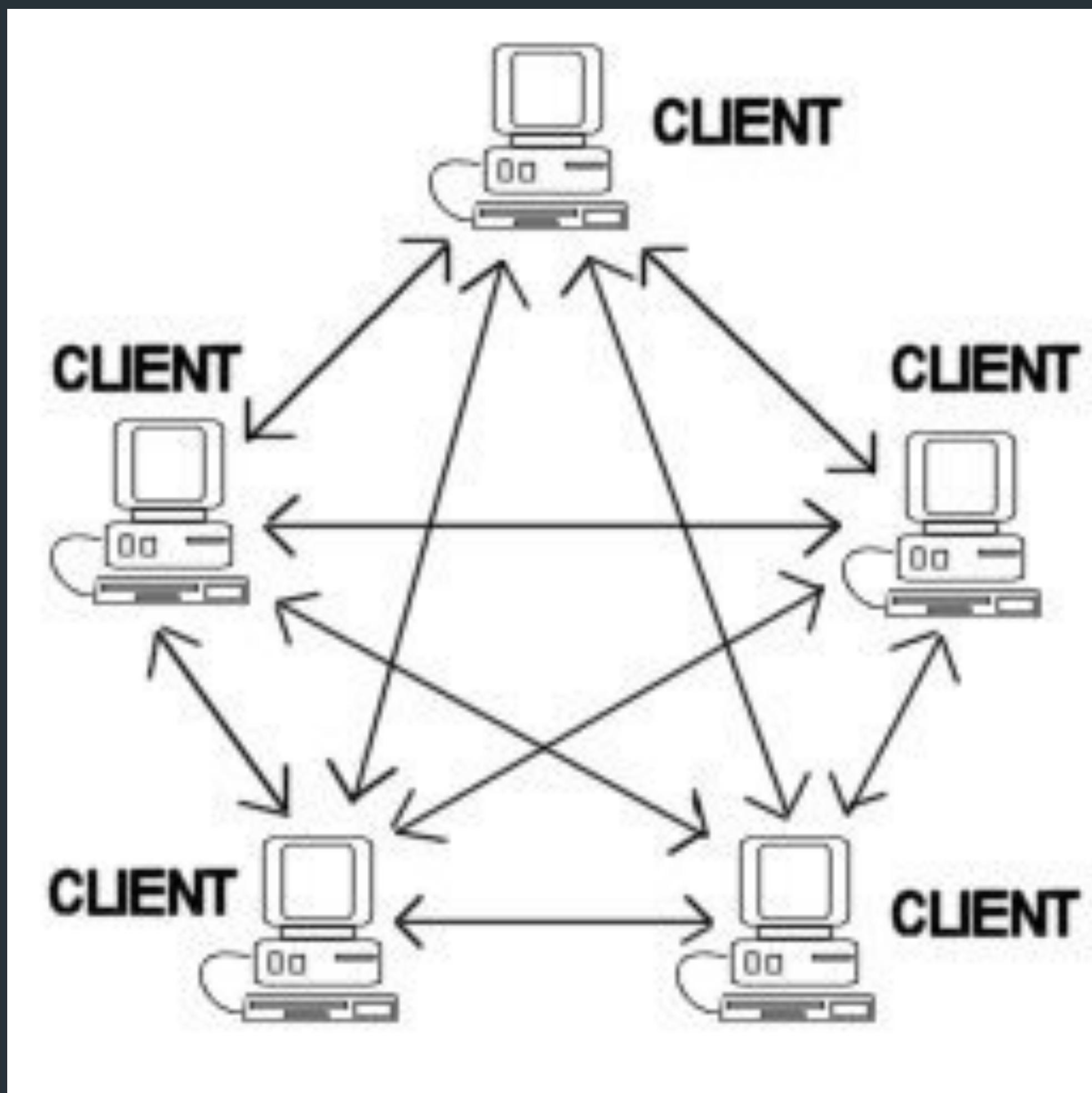
- HTTP
- TCP
- AMQP
- MQTT

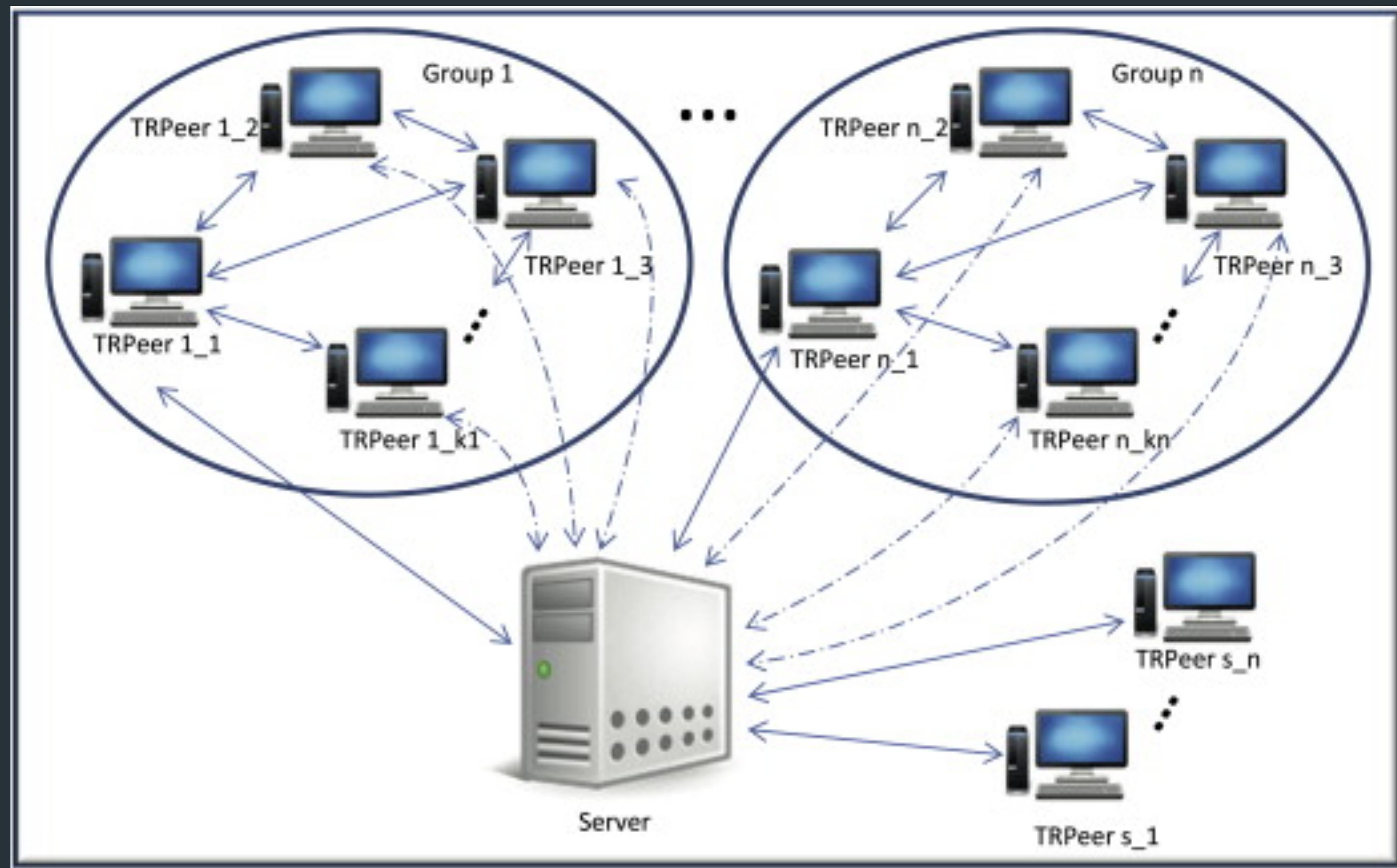


Web Service Architectures









Types of Web Services

Types of Web Services

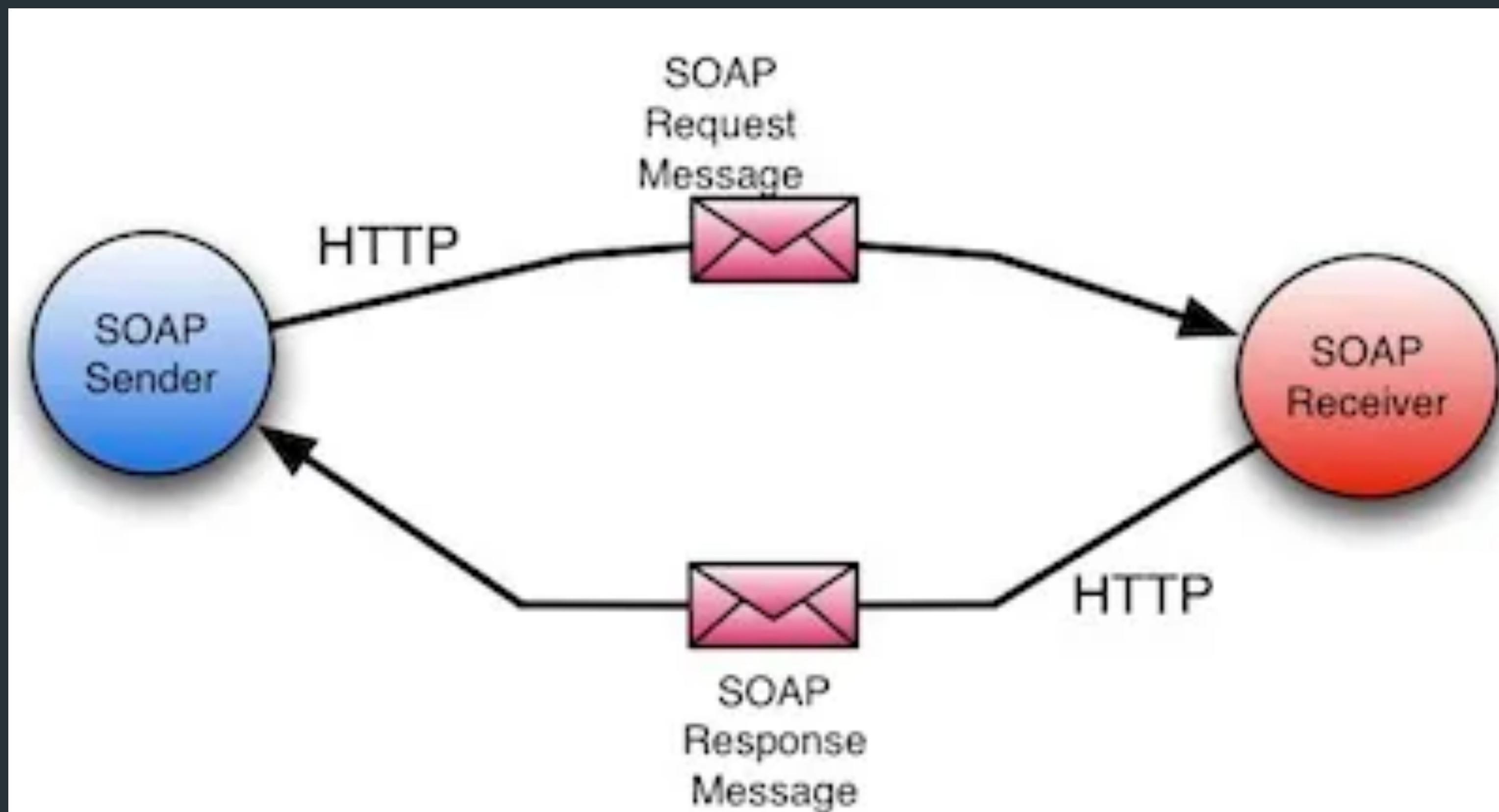
Protocol Comparison			
	First released	Formatting type	Key strength
SOAP	Late 1990s	XML	Widely used and established
REST	2000	JSON, XML, and others	Flexible data formatting
JSON-RPC	mid-2000s	JSON	Simplicity of implementation
gRPC	2015	Protocol buffers by default; can be used with JSON & others also	Ability to define any type of function
GraphQL	2015	JSON	Flexible data structuring
Thrift	2007	JSON or Binary	Adaptable to many use cases

SOAP (Simple Object Access Protocol)

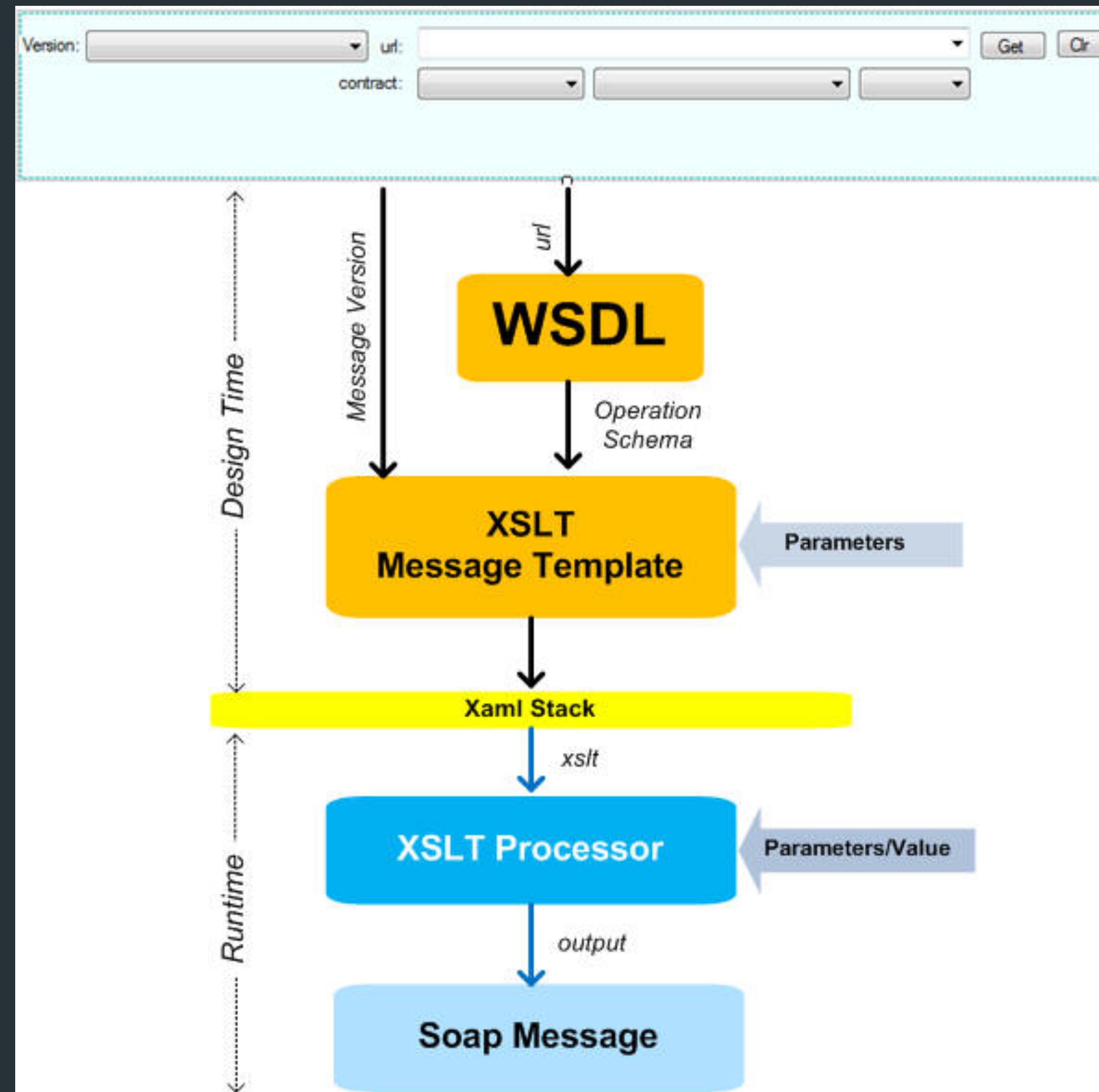
SOAP (Simple Object Access Protocol)



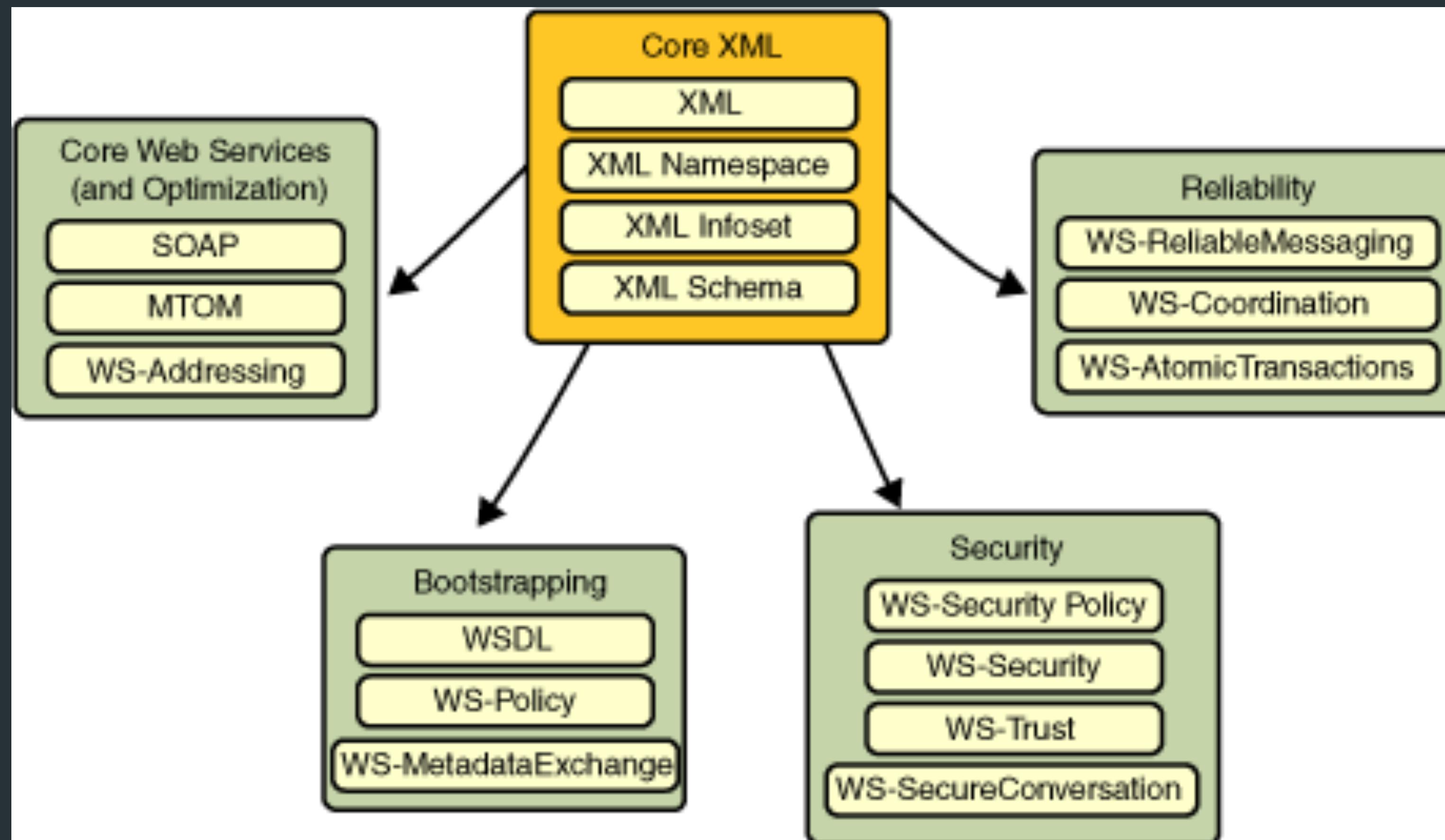
SOAP (Simple Object Access Protocol)



SOAP (Simple Object Access Protocol)



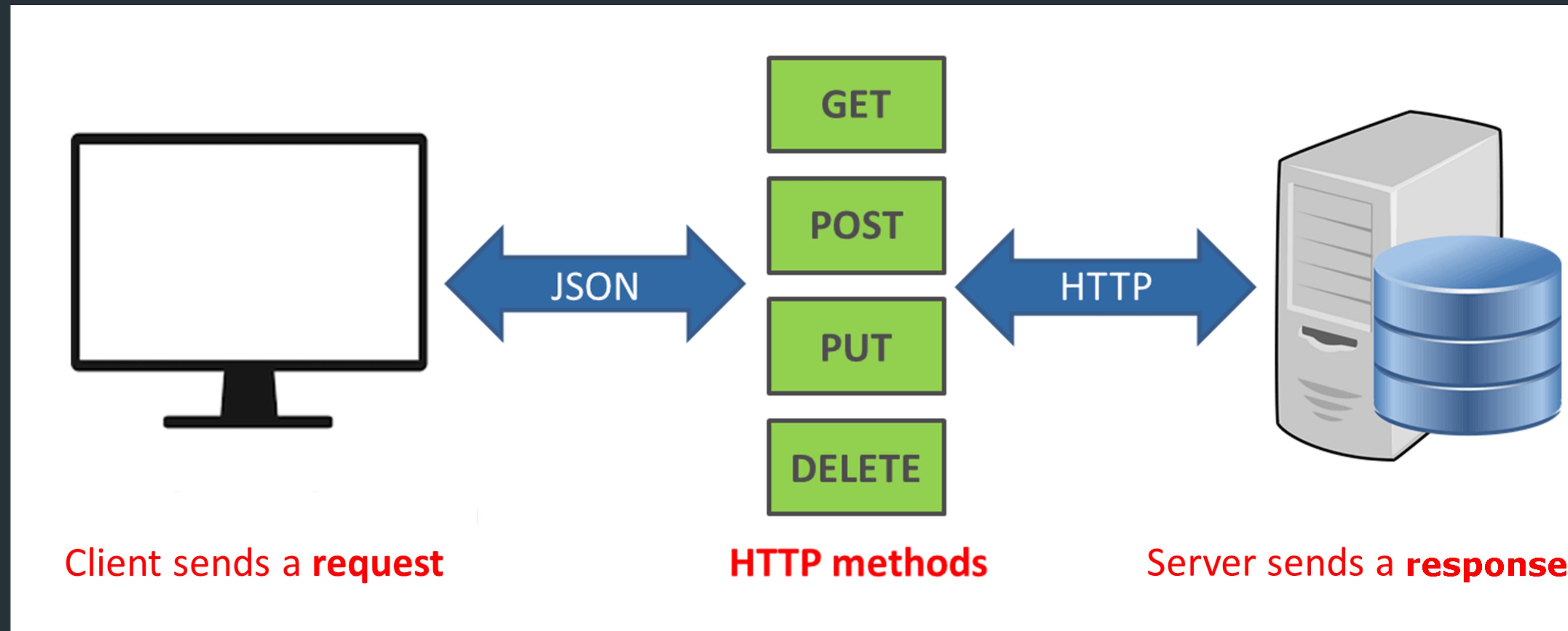
SOAP (Simple Object Access Protocol)



RESTful Web Services

REST (Representational State Transfer)

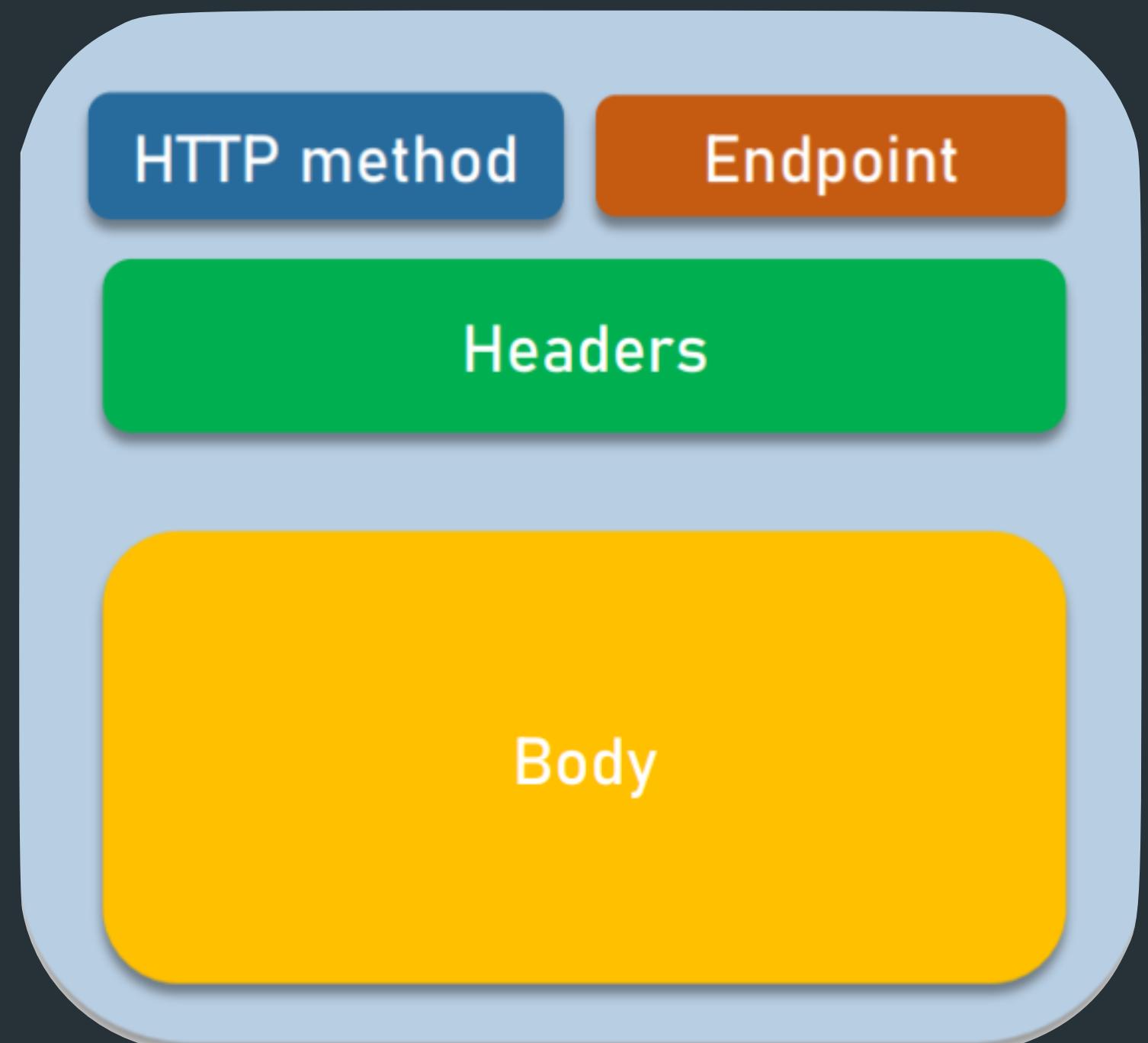
REST (Representational State Transfer)



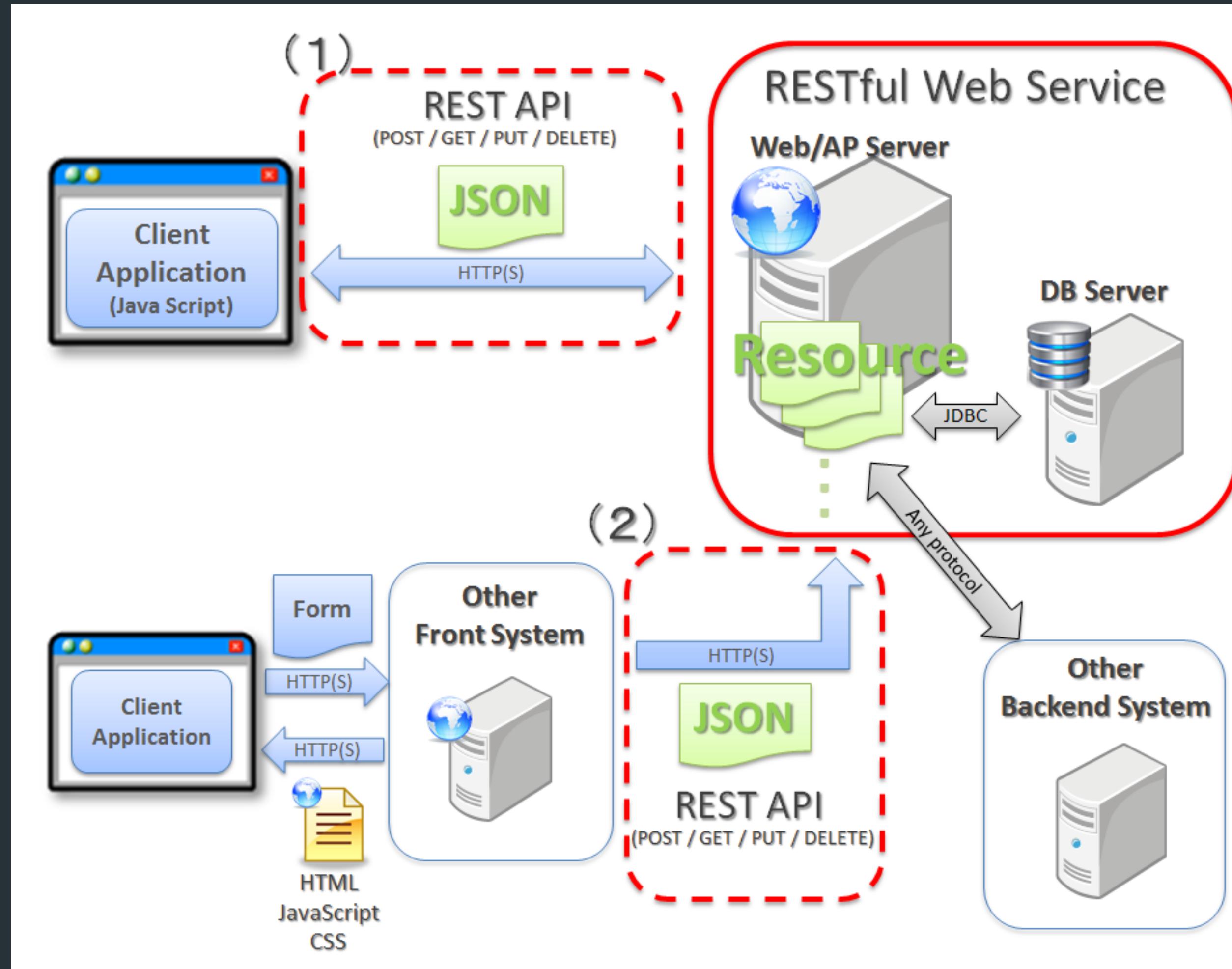
REST (Representational State Transfer)

```
POST /api/2.2/sites/9a8b7c6d-5e4f-3a2b-1c0d-9e8f7a6b5c4d/users HTTP/1.1
HOST: my-server
X-Tableau-Auth: 12ab34cd56ef78ab90cd12ef34ab56cd
Content-Type: application/json

{
  "user": {
    "name": "NewUser1",
    "siteRole": "Publisher"
  }
}
```



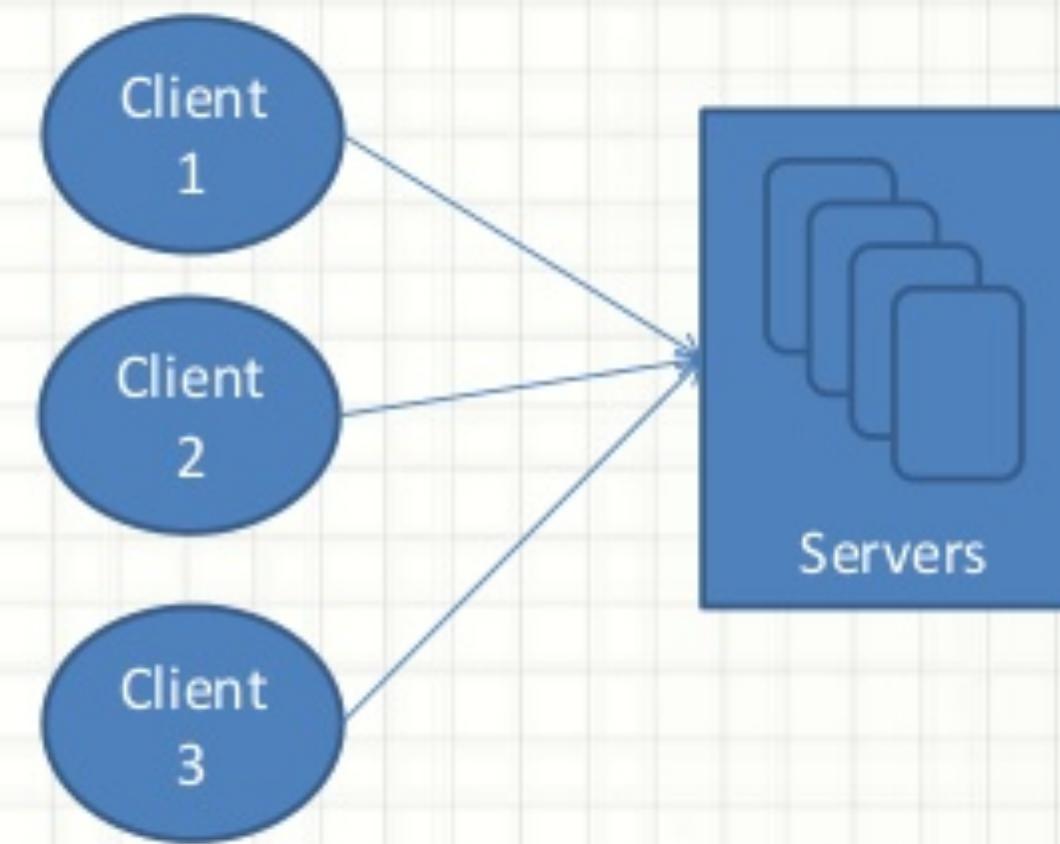
REST (Representational State Transfer)



REST (Representational State Transfer)

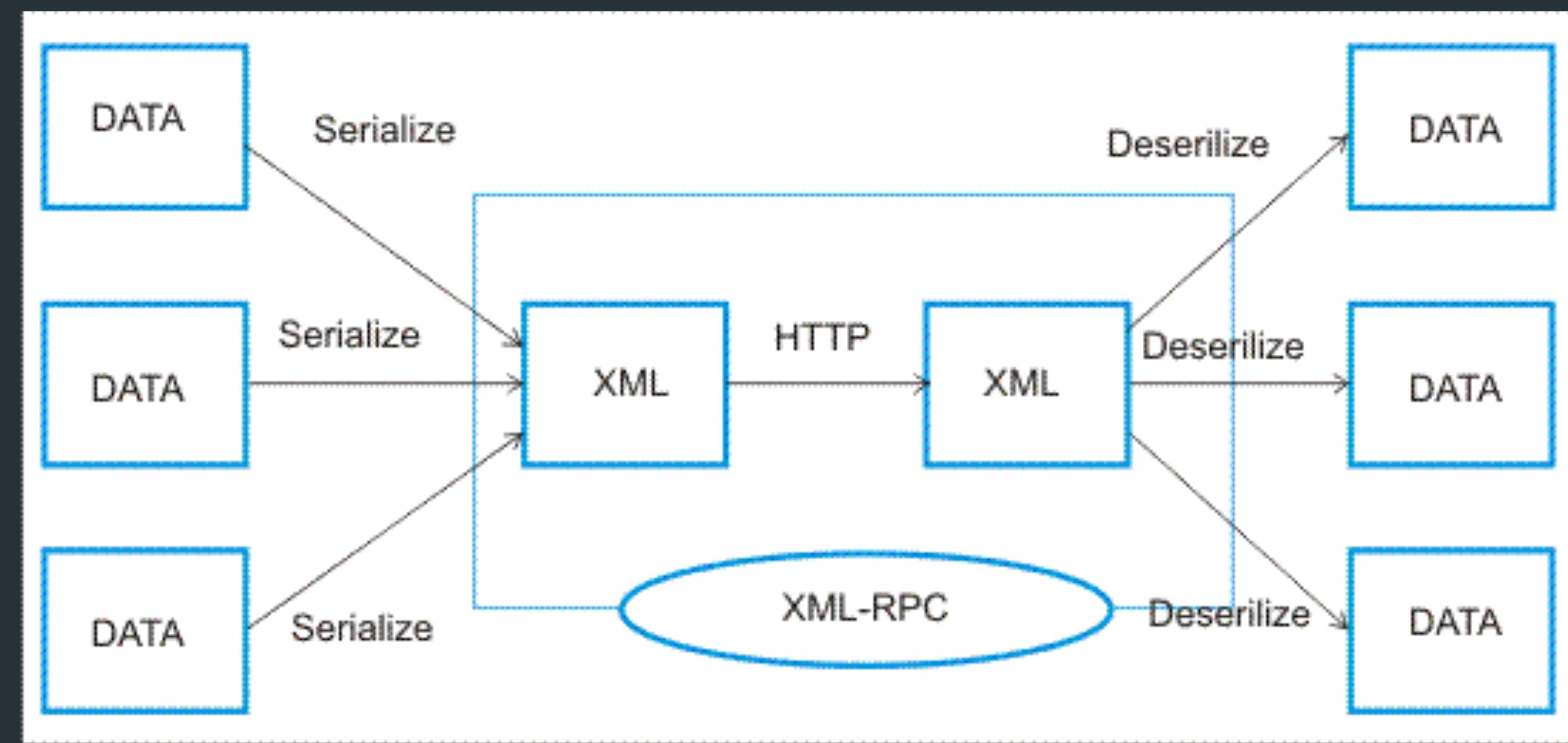
Stateless

- No client state at server.
- Any State is maintained at Client side.
- Each request has all the information to process the request.

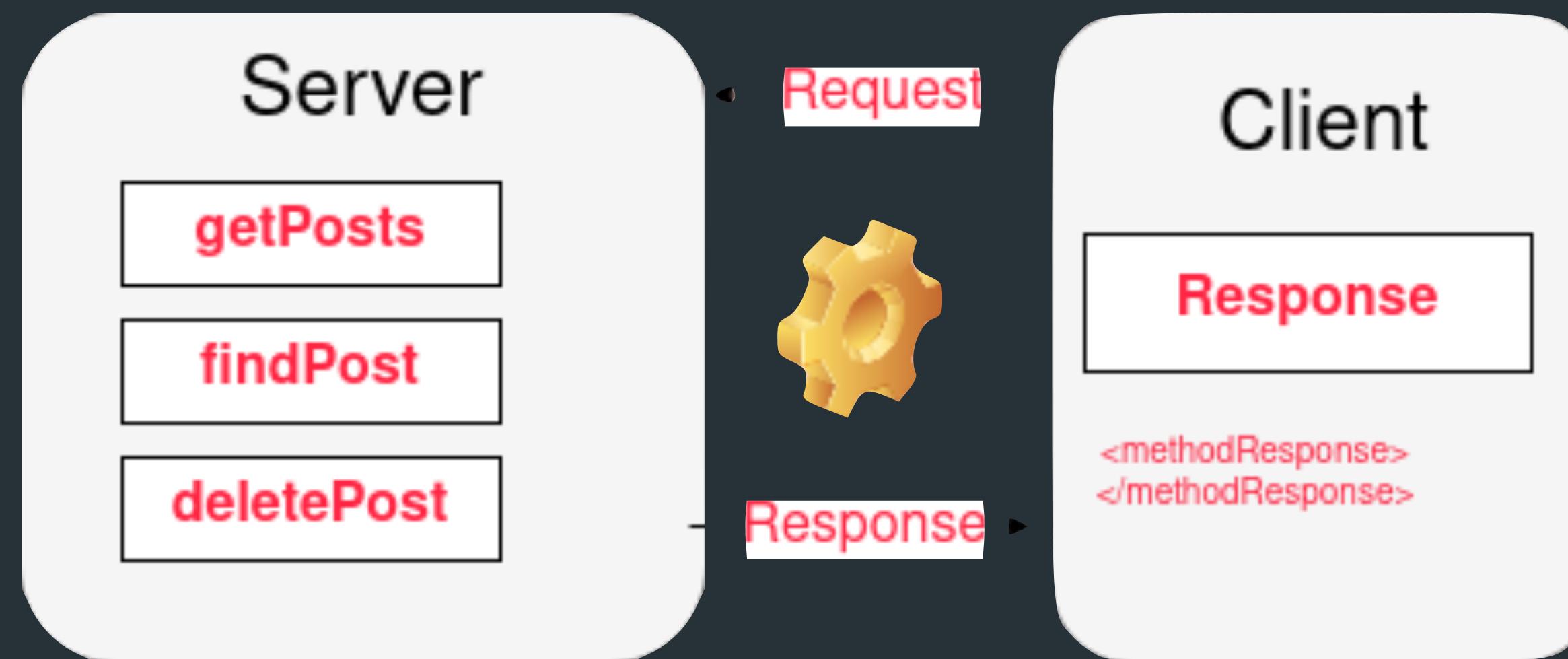


XML-RPC Web Services

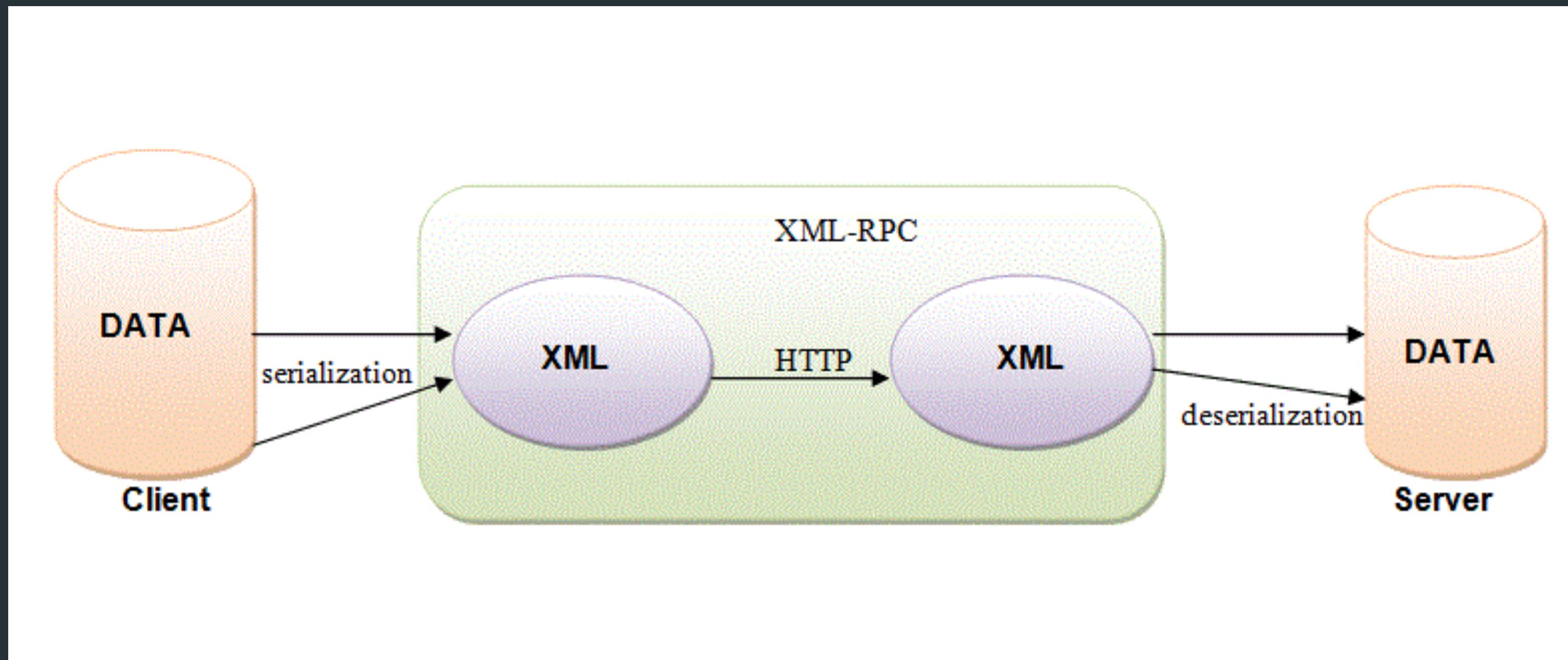
XML-RPC Web Services



XML-RPC Web Services



XML-RPC Web Services



XML-RPC Web Services

SOAP vs XML-RPC

- **Similarities**

- Use XML for messaging
- Messages are usually embedded into HTTP header
- Use request/response mechanism
- Mainly used in remote procedure call
- Platform independent
- Language independent

- **Differences**

- SOAP messages are more complicated than XML-RPC
- Make use of XML namespaces and XML Schemas
- Hence provide a standard way for data encoding and RPC
- Thus allow automatic method invocation on the Web

JSON-RPC Web Services

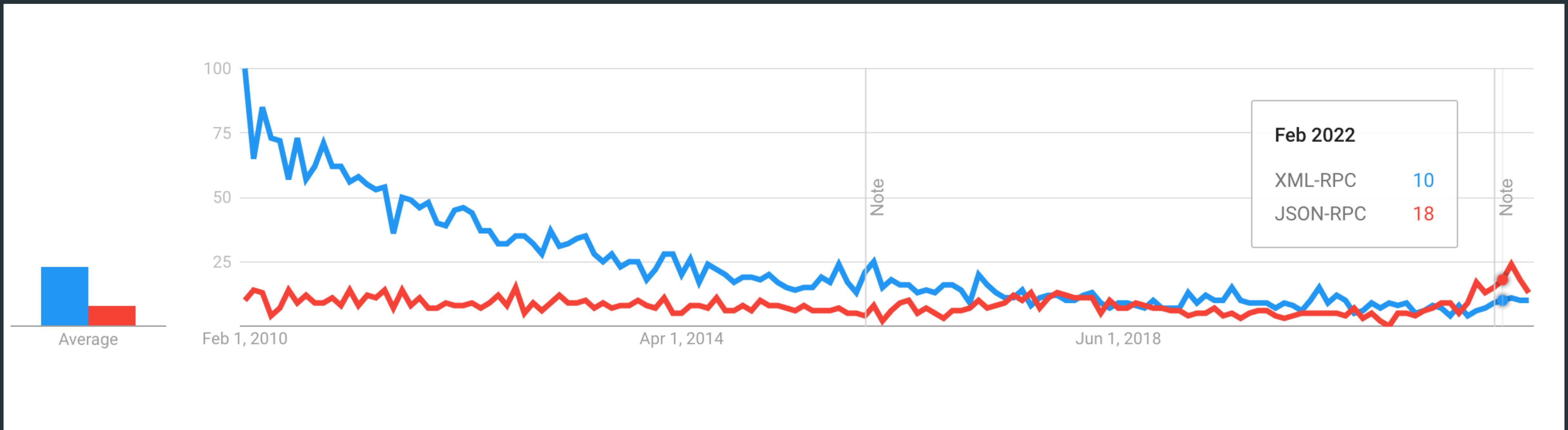
JSON-RPC Web Services



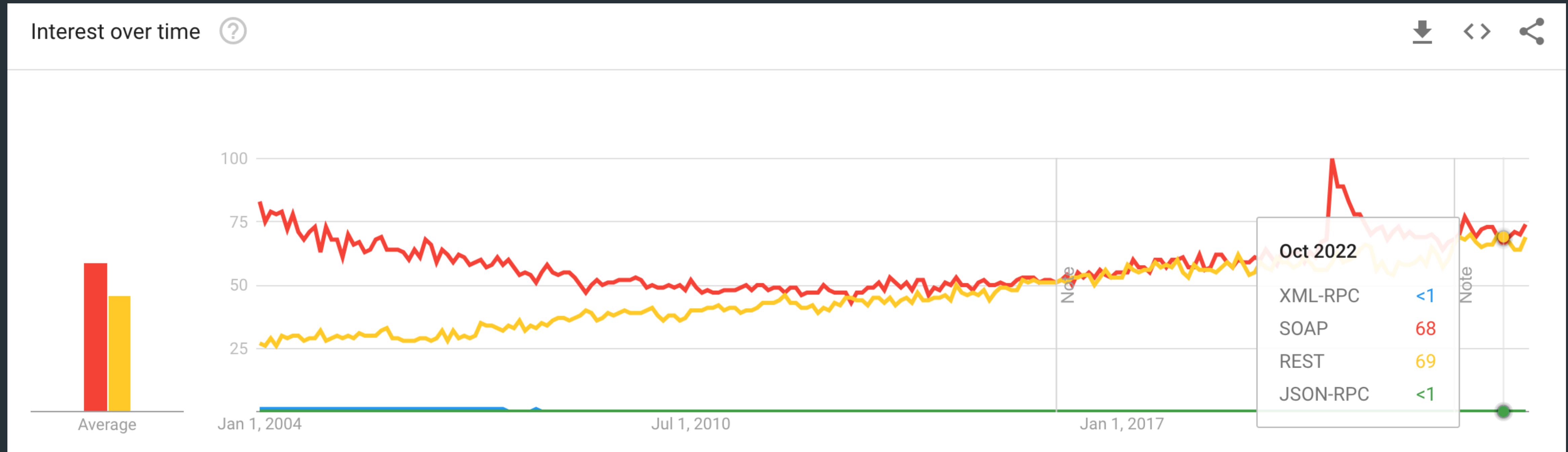
JSON-RPC Web Services

```
{  
    "jsonrpc": "2.0",  
    "method": "user.get",  
    "params": {  
        "output": "extend"  
    },  
    "auth": "038e1d7b1735c6a5436ee9eae095879e",  
    "id": 1  
}
```

JSON-RPC Web Services

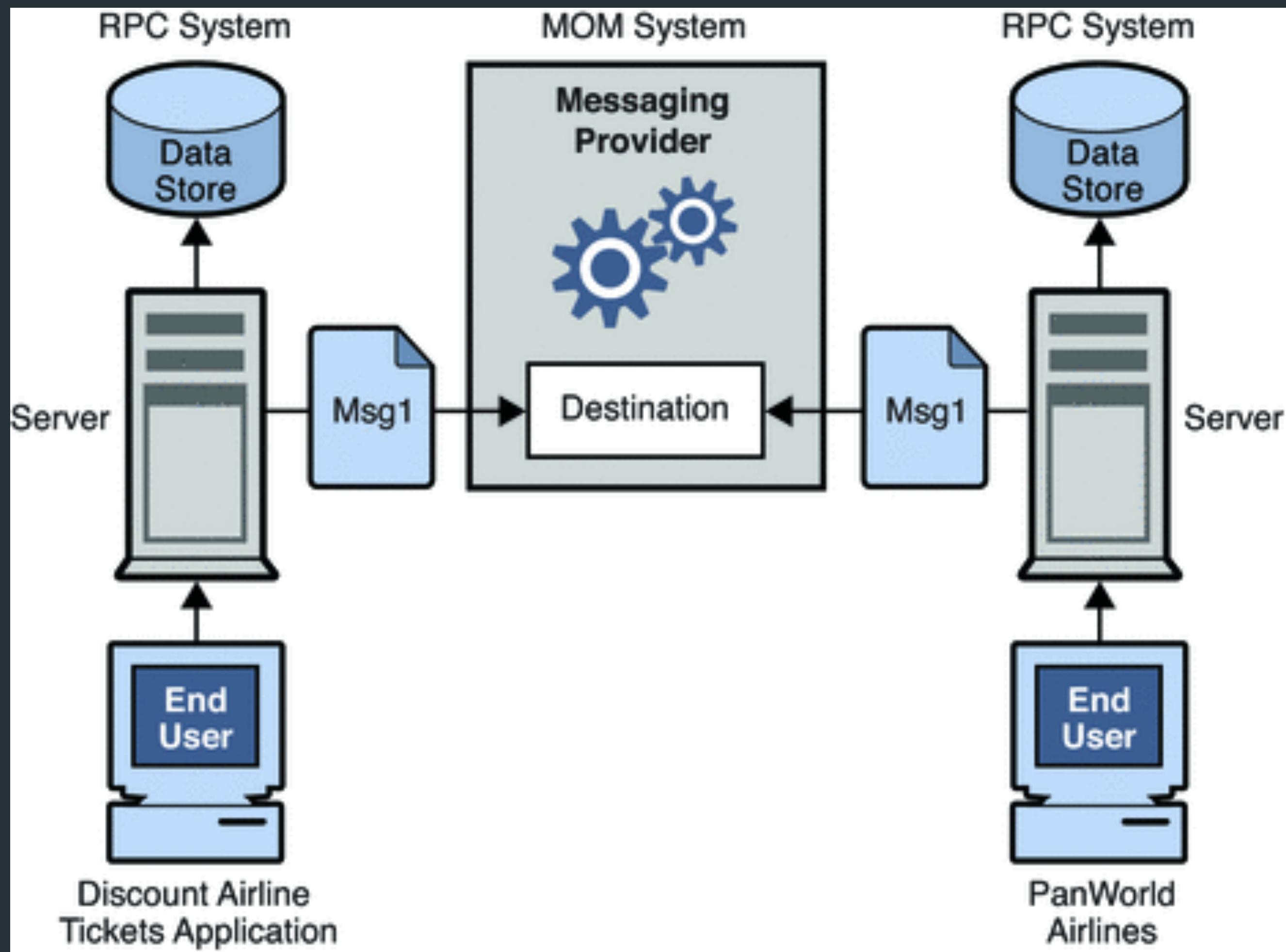


Interest over Time



Message-oriented Middleware (MOM)

Message-oriented Middleware (MOM)



Message-oriented Middleware (MOM)

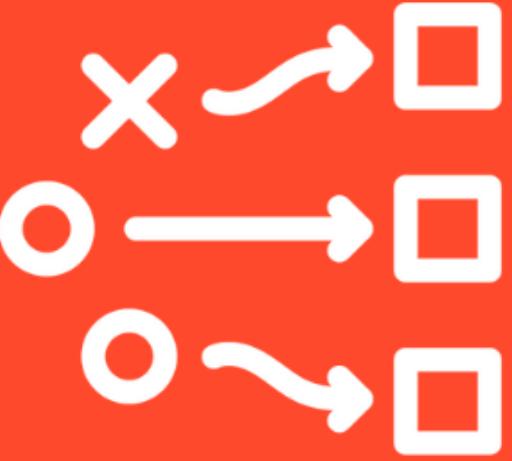
G2.com

MOM



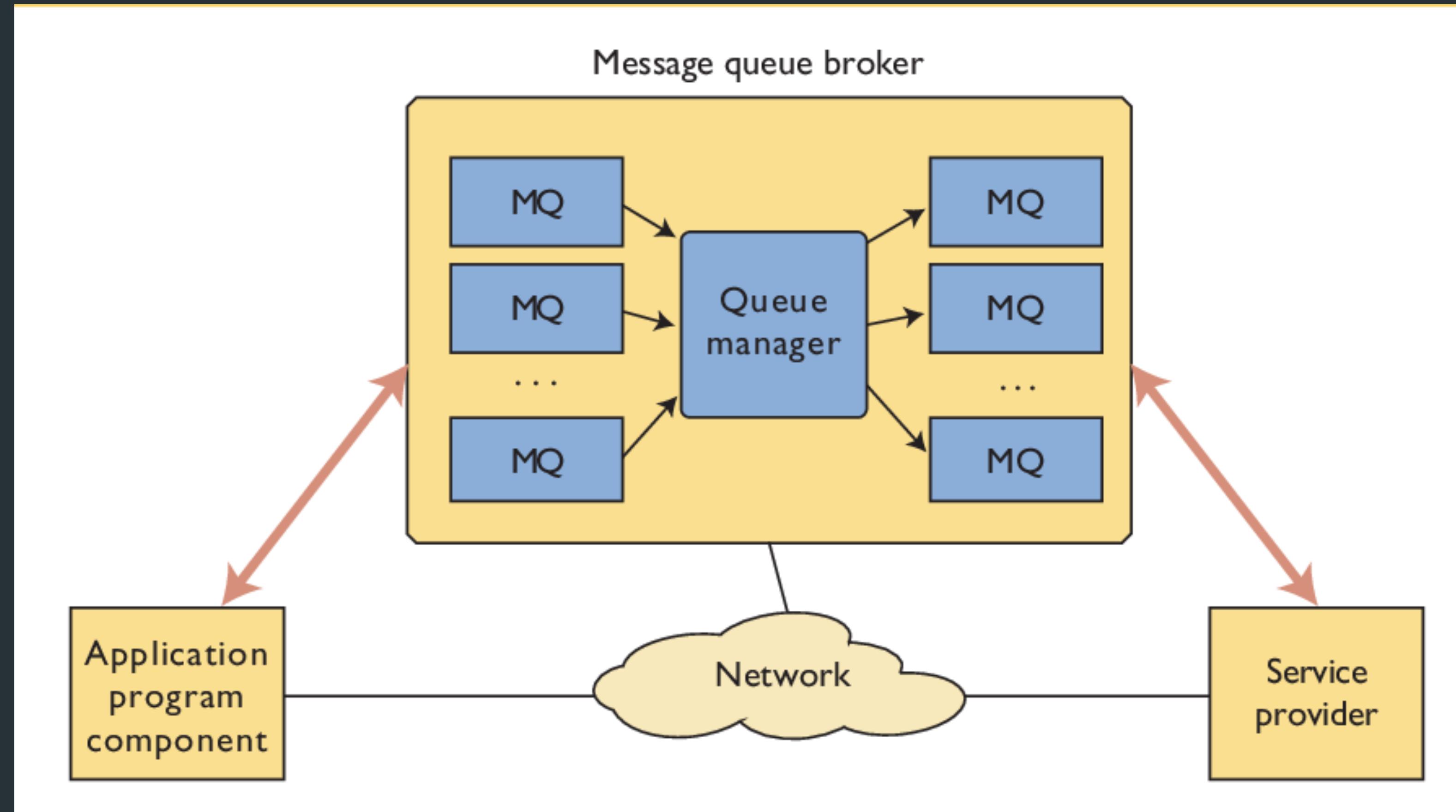
MOM architecture provides **asynchronous communication** between systems.

RPC

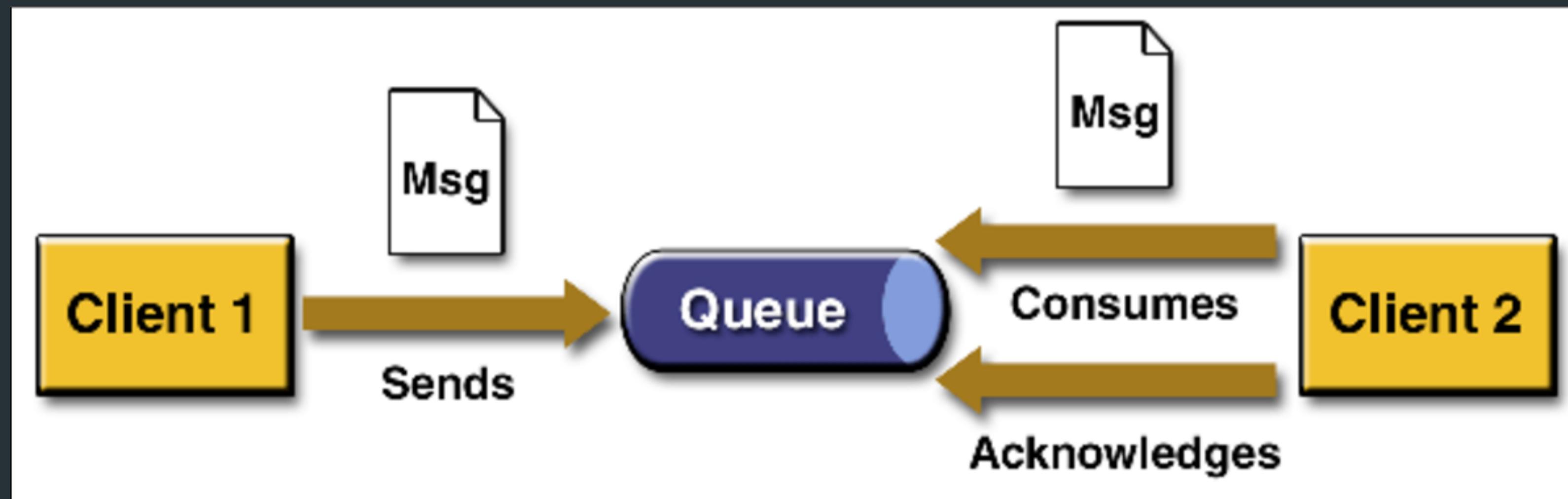


RPC provides **synchronous communication** between systems.

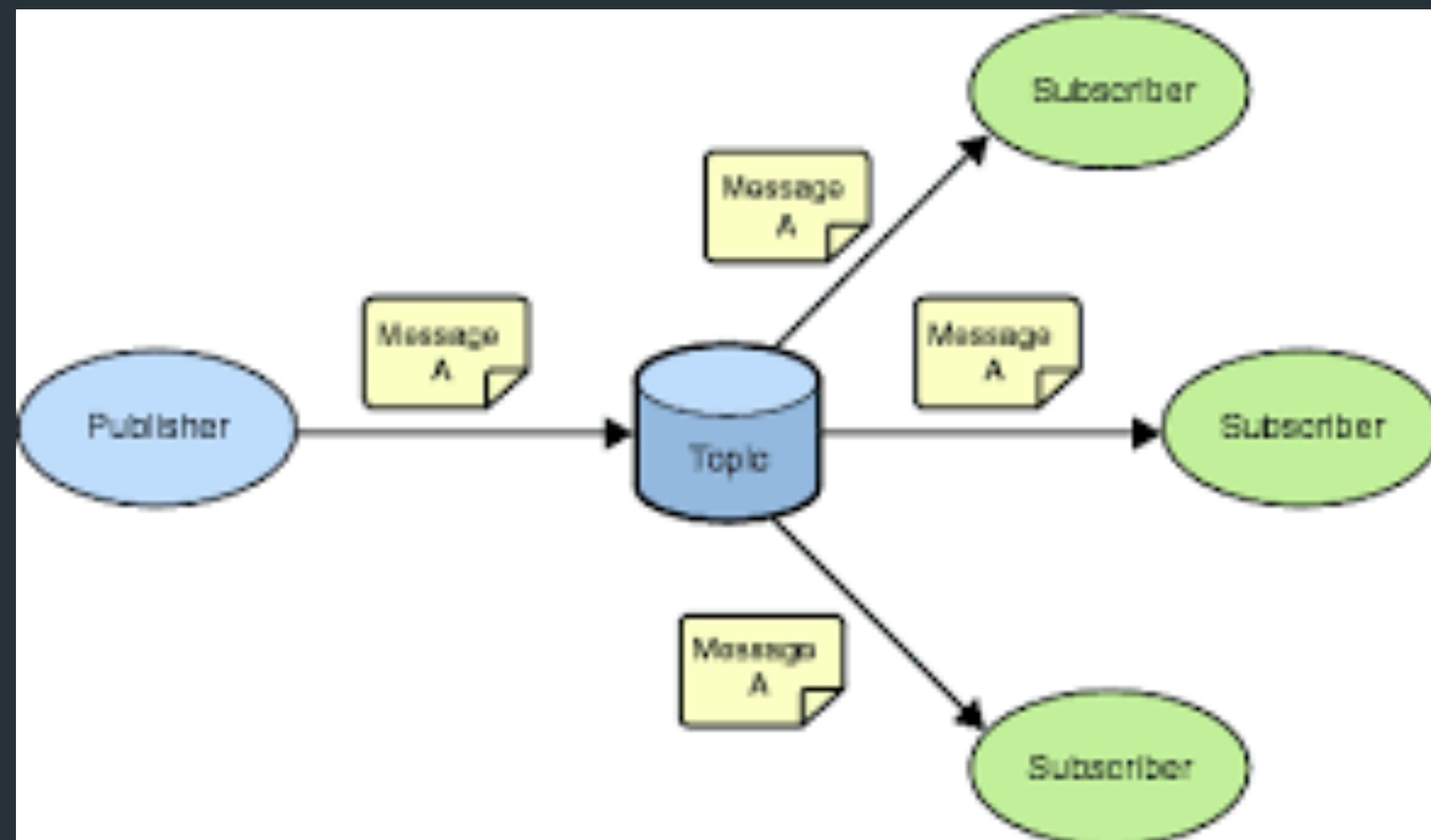
Message-oriented Middleware (MOM)



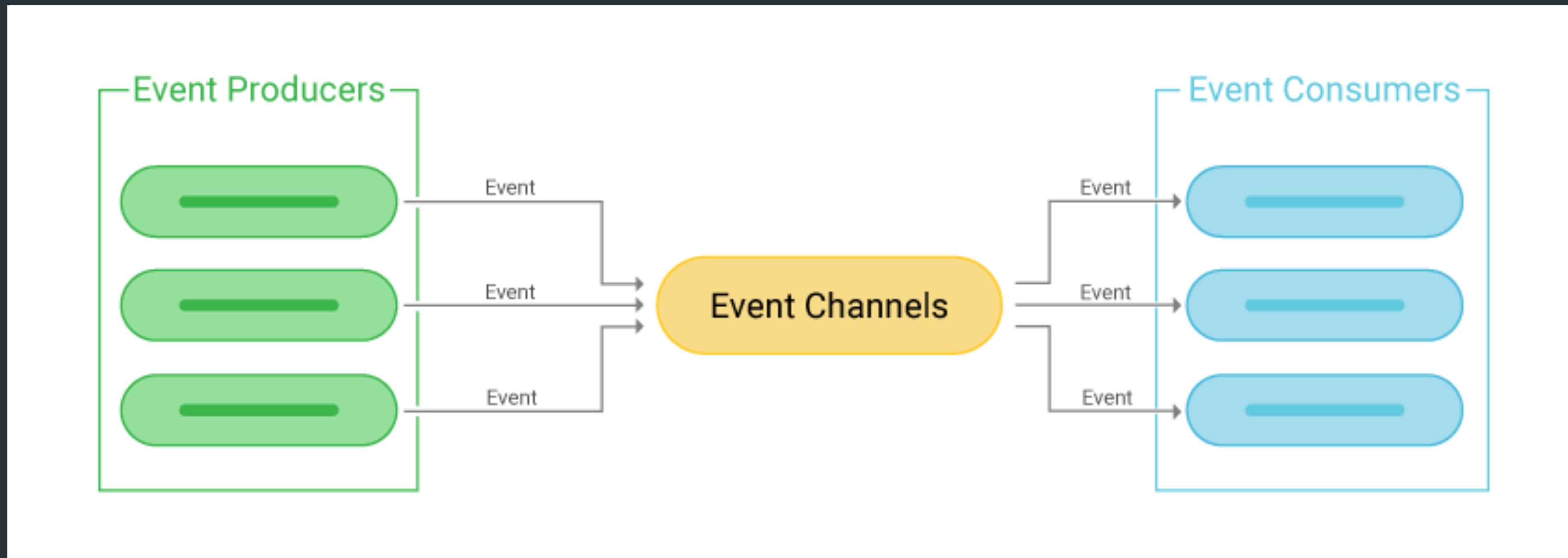
Message-oriented Middleware (MOM)



Message-oriented Middleware (MOM)



Message-oriented Middleware (MOM)



Service Oriented Architecture (SOA)

Service Oriented Architecture (SOA)



Service Oriented Architecture (SOA)

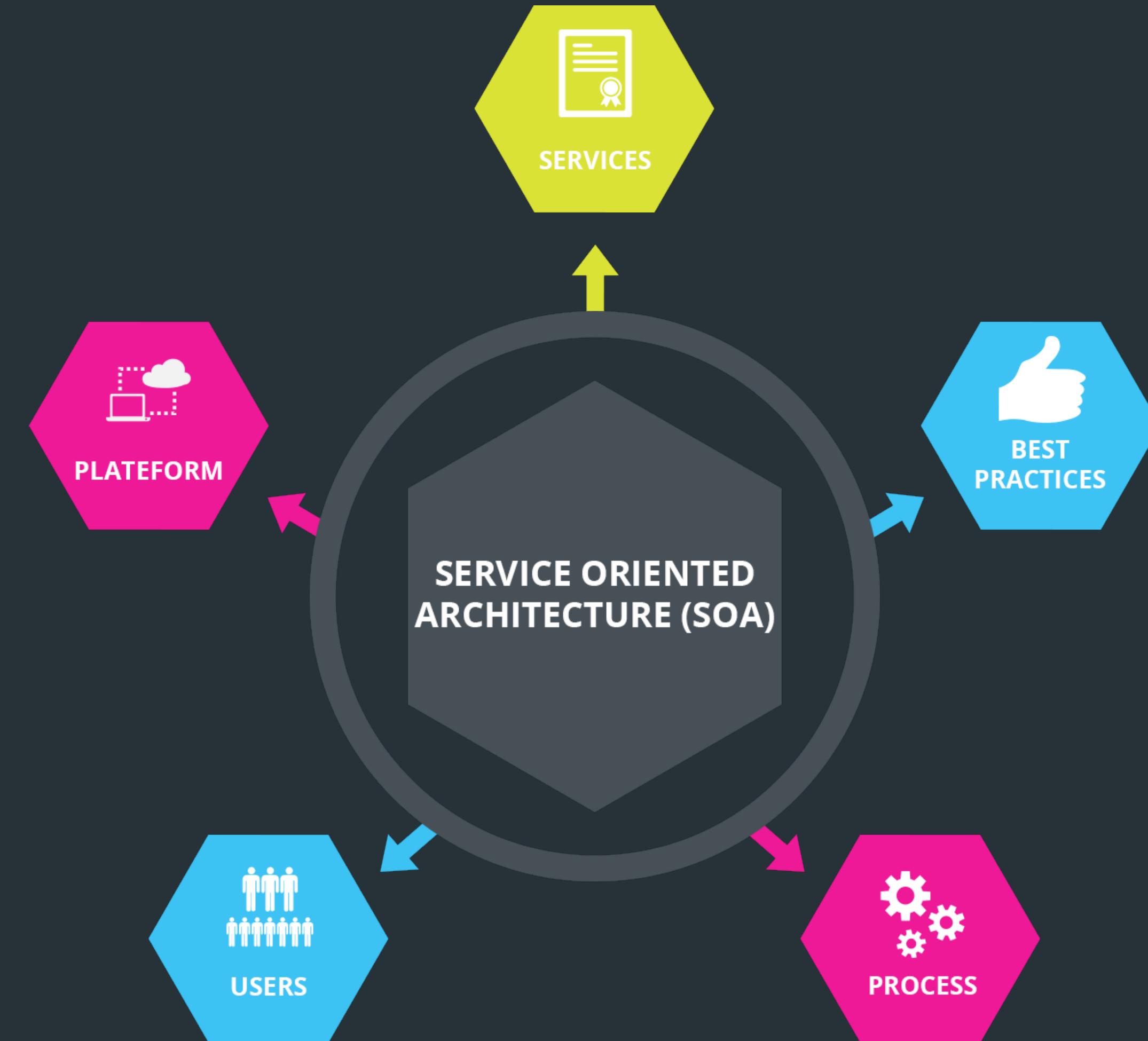


Service Oriented Architecture (SOA)



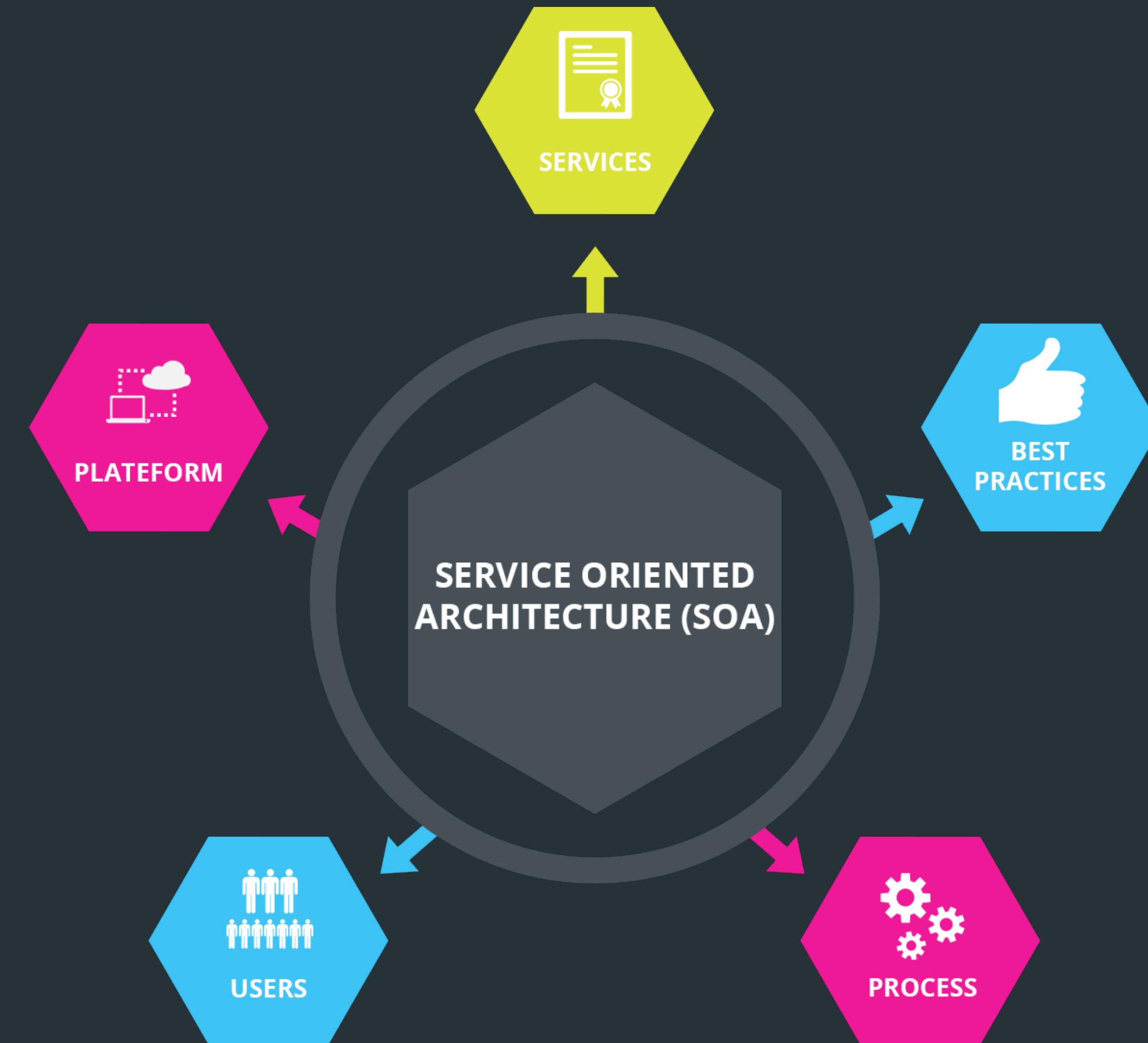
SOA Key Features

- Service contract
- Service description
- Service discovery
- Service composition
- Service security



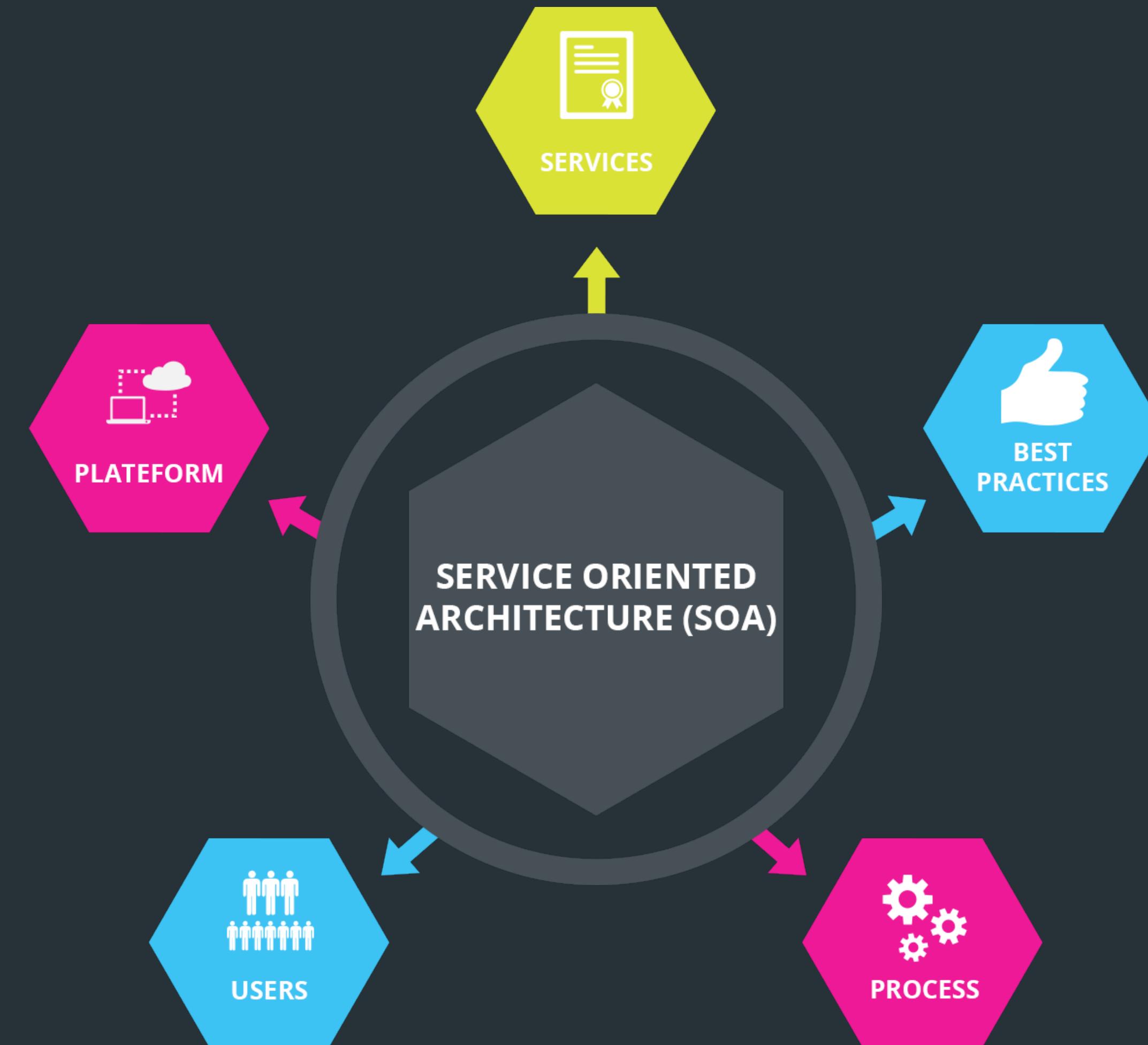
SOA Key Features

- **Service contract**
- Service description
- Service discovery
- Service composition
- Service security



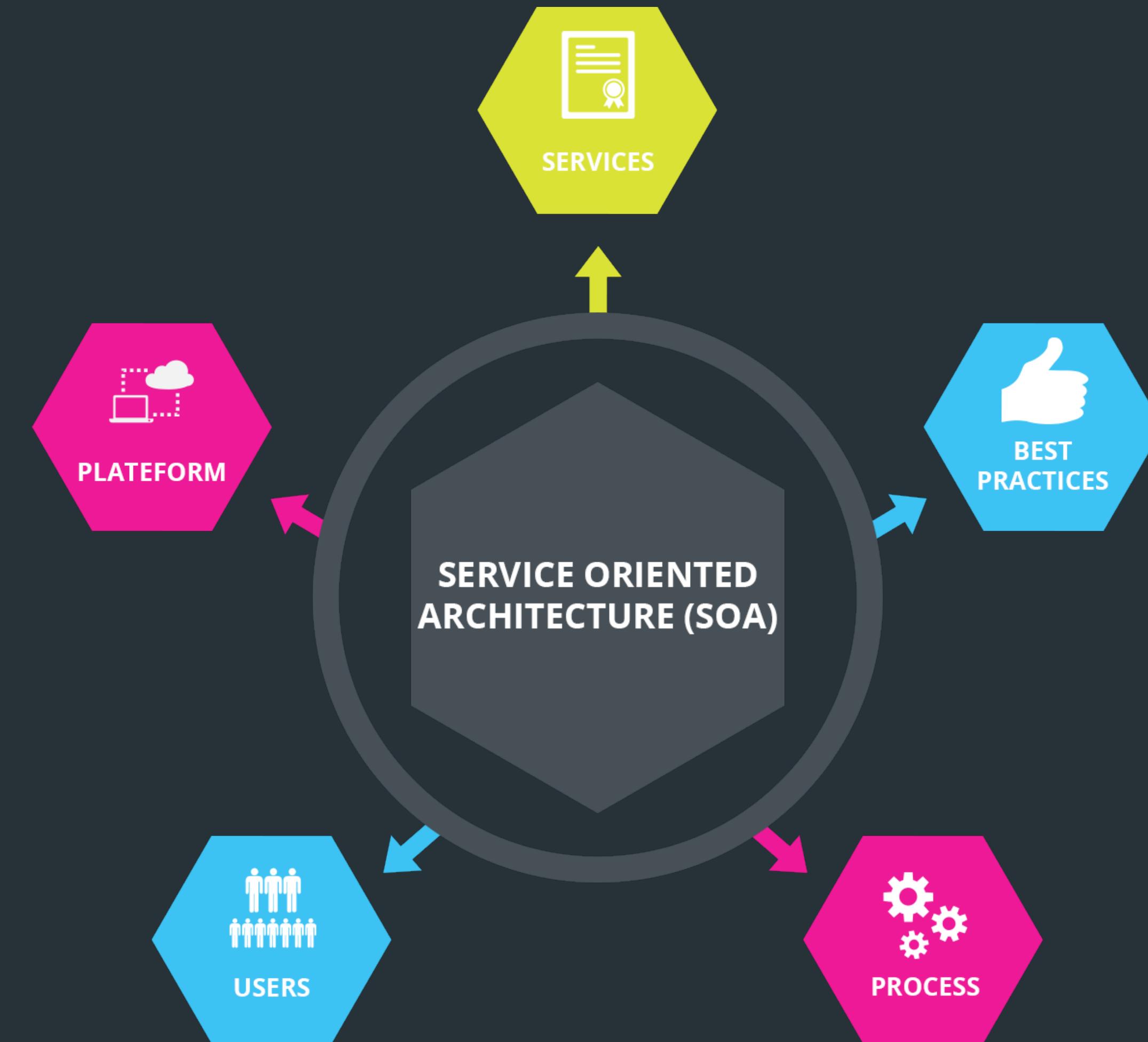
SOA Key Features

- Service contract
- **Service description**
- Service discovery
- Service composition
- Service security



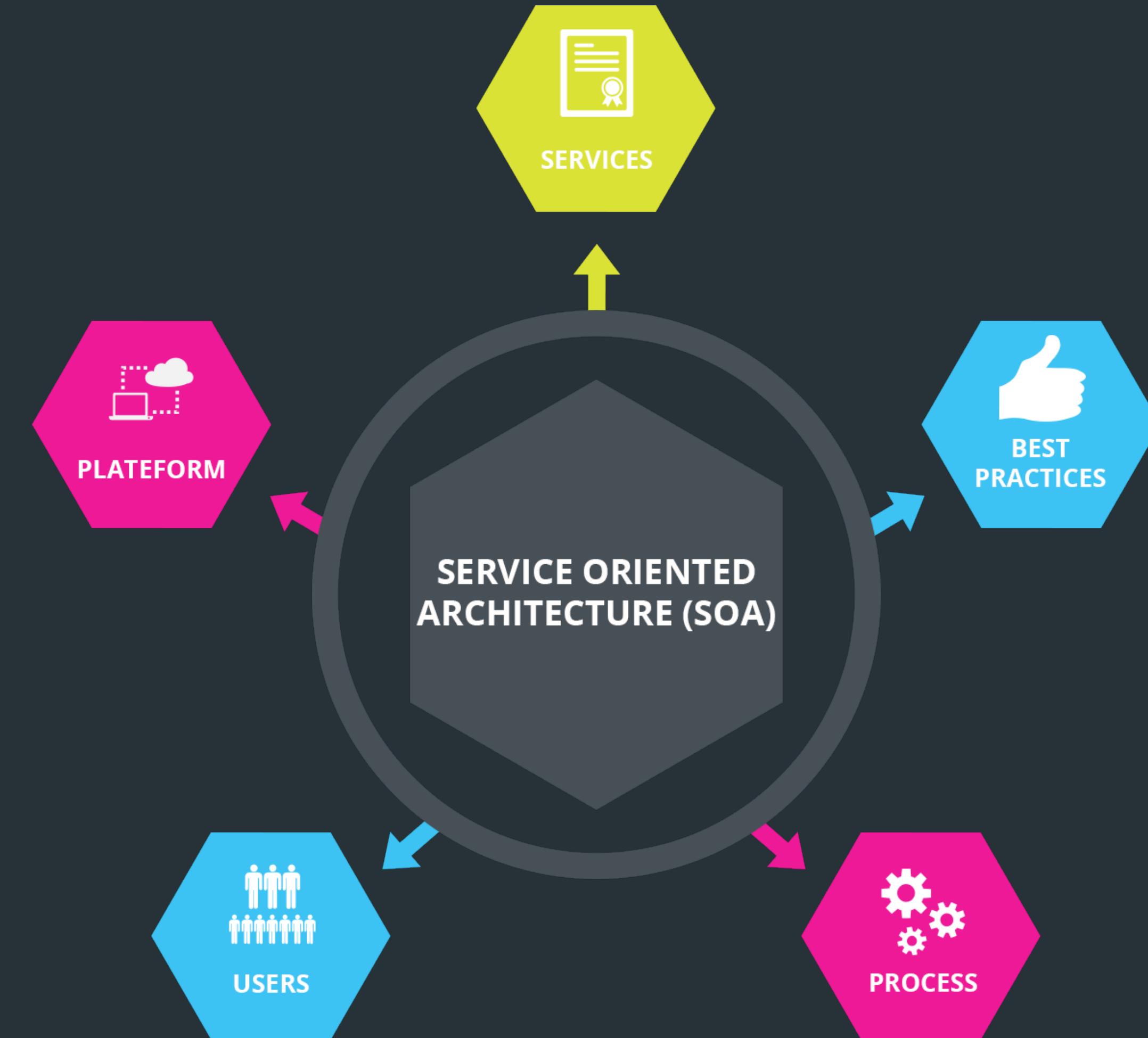
SOA Key Features

- Service contract
- Service description
- **Service discovery**
- Service composition
- Service security



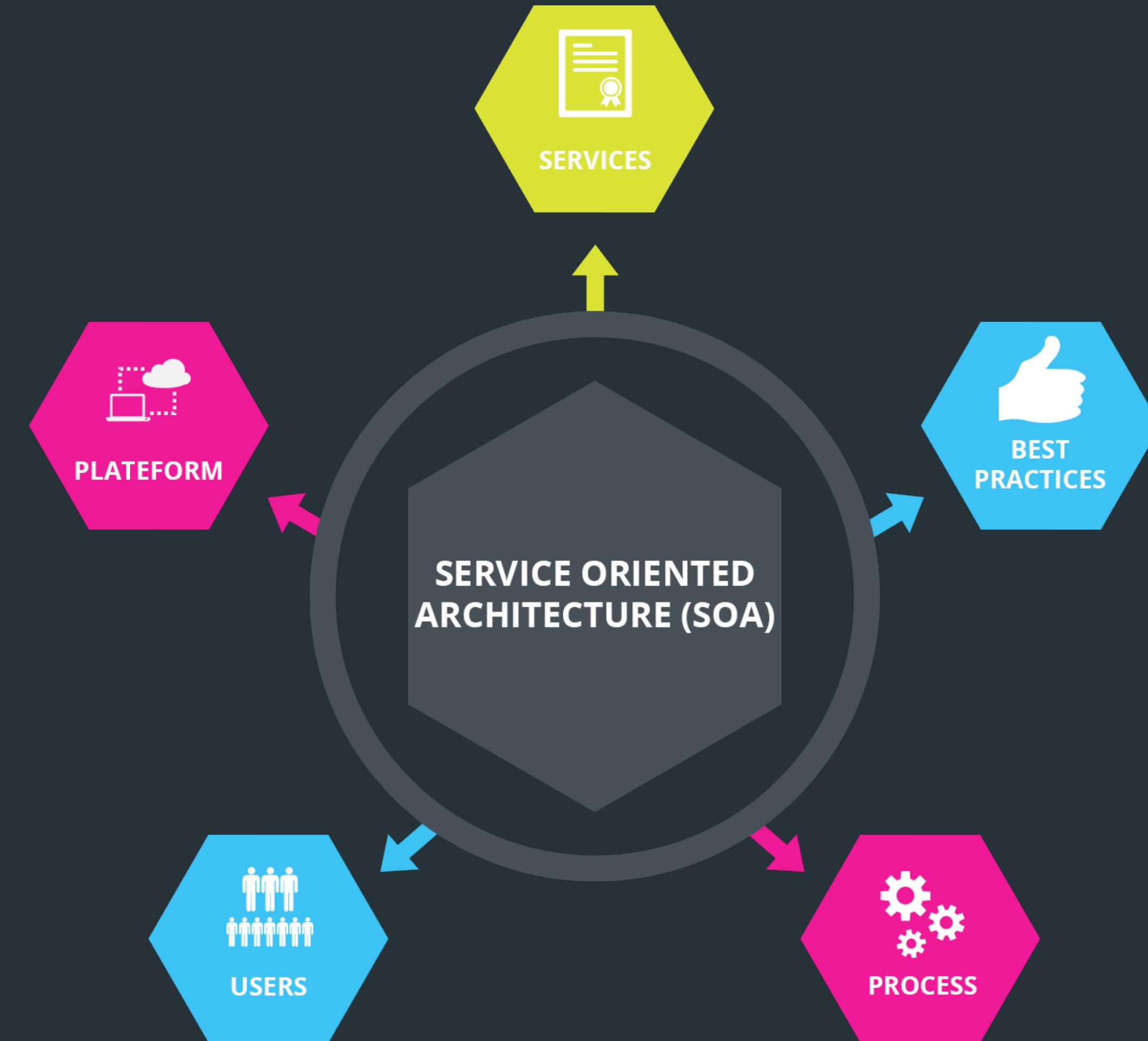
SOA Key Features

- Service contract
- Service description
- Service discovery
- **Service composition**
- Service security



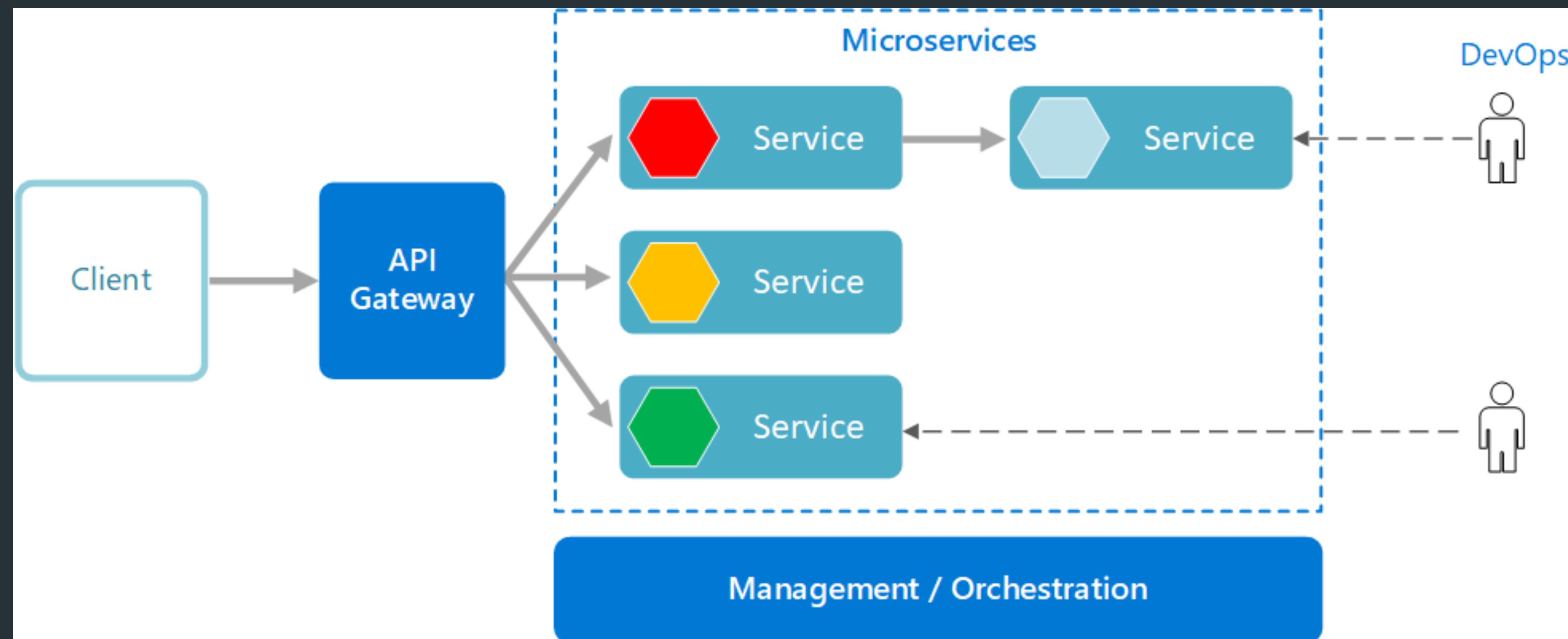
SOA Key Features

- Service contract
- Service description
- Service discovery
- Service composition
- **Service security**

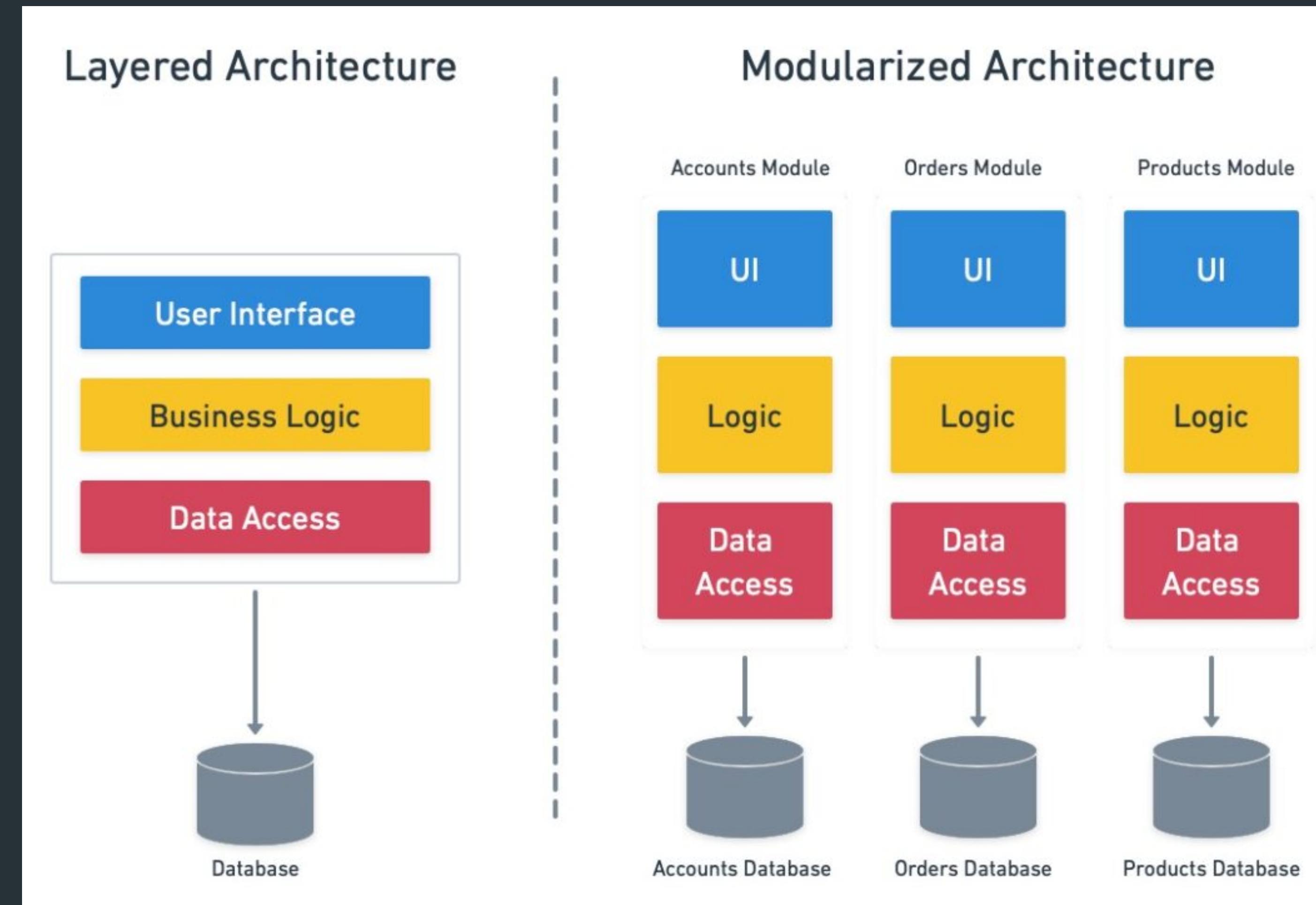


Microservices Architecture

Microservices Architecture



Microservices Architecture

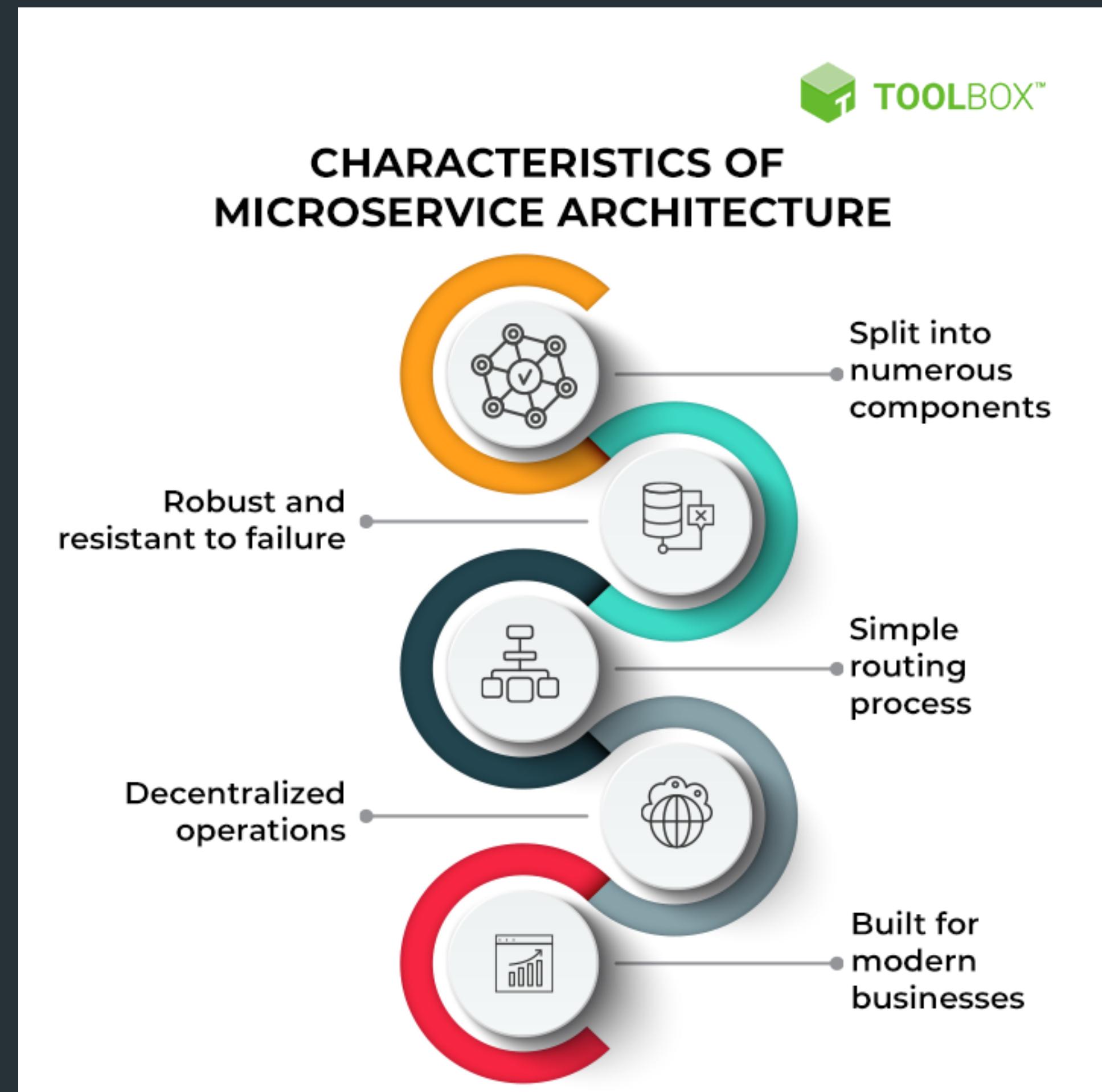


Microservices Architecture



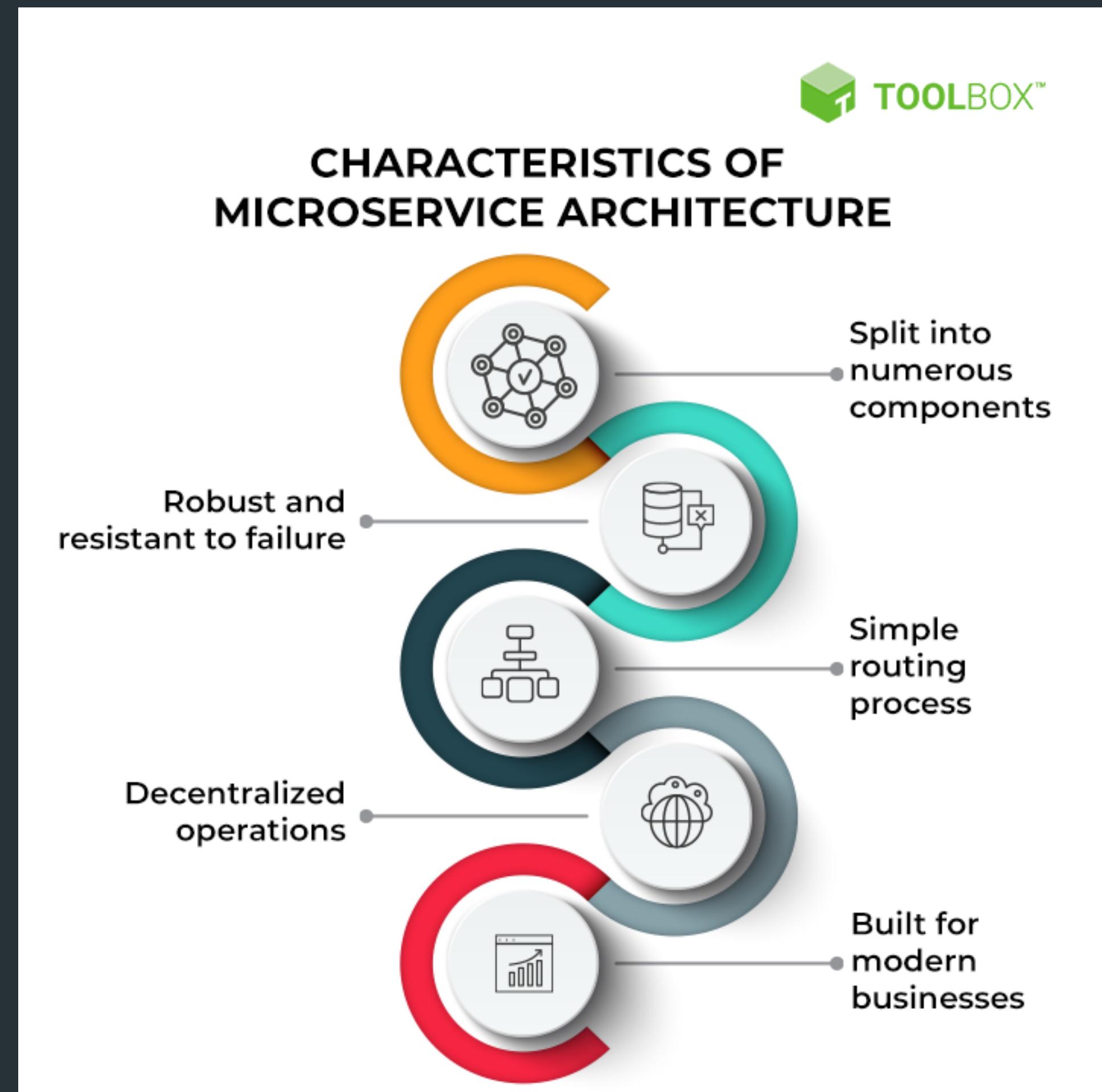
Microservices Key Concepts

- Service boundaries
- API-first design
- Decentralized governance
- Continuous integration and deployment
- Resilience and fault-tolerance



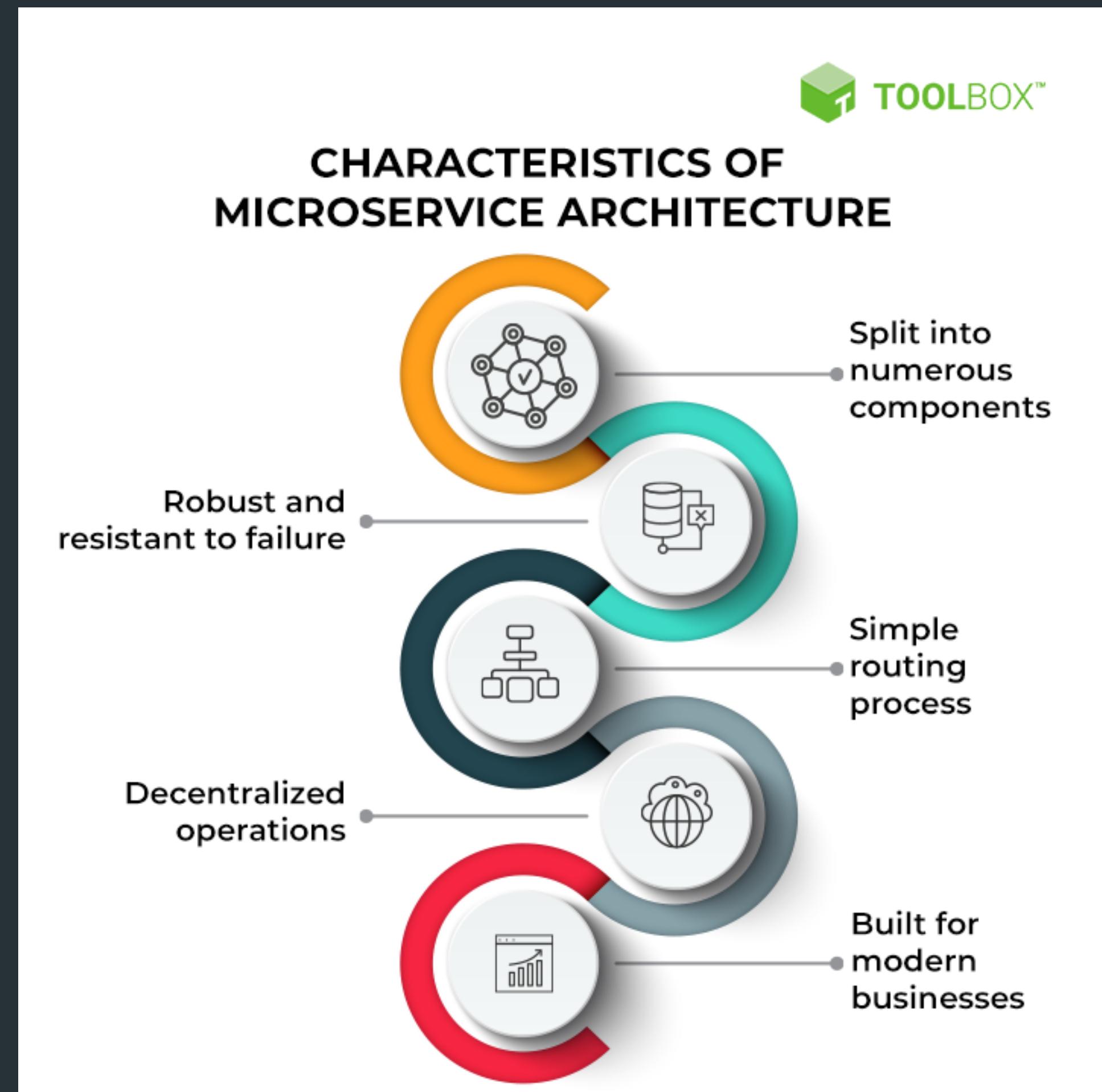
Microservices Key Concepts

- **Service boundaries**
- API-first design
- Decentralized governance
- Continuous integration and deployment
- Resilience and fault-tolerance



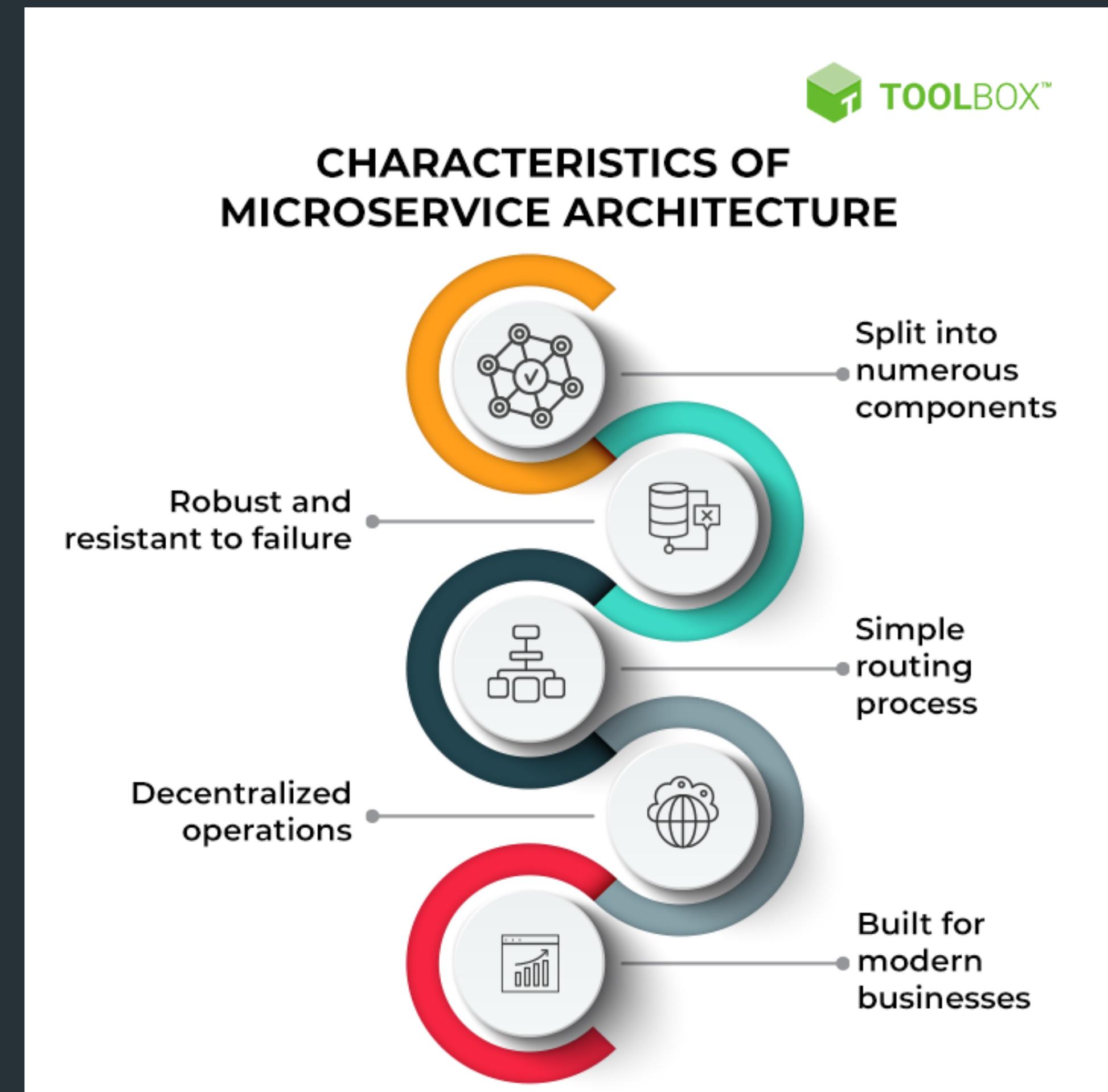
Microservices Key Concepts

- Service boundaries
- **API-first design**
- Decentralized governance
- Continuous integration and deployment
- Resilience and fault-tolerance



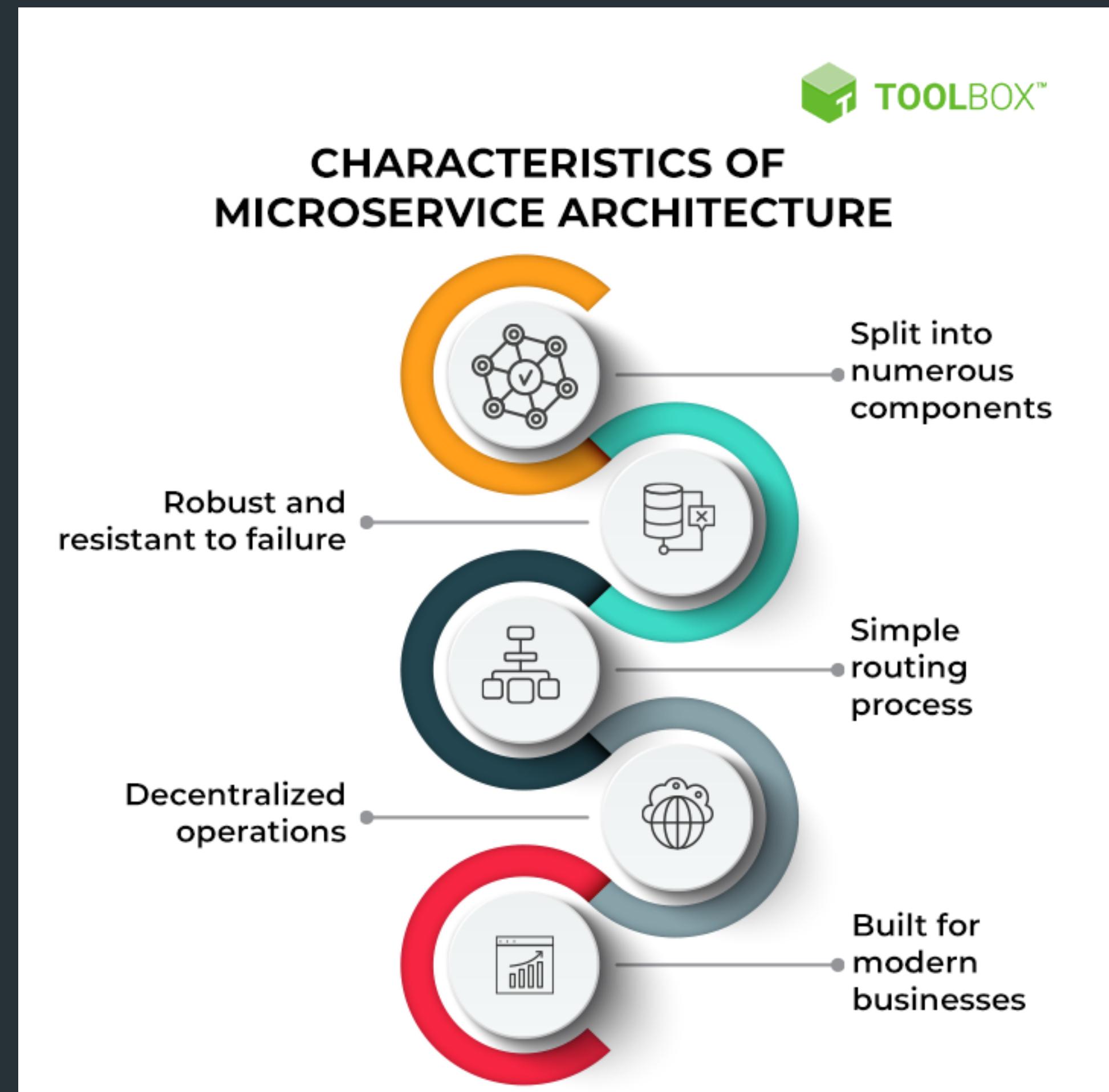
Microservices Key Concepts

- Service boundaries
- API-first design
- **Decentralized governance**
- Continuous integration and deployment
- Resilience and fault-tolerance



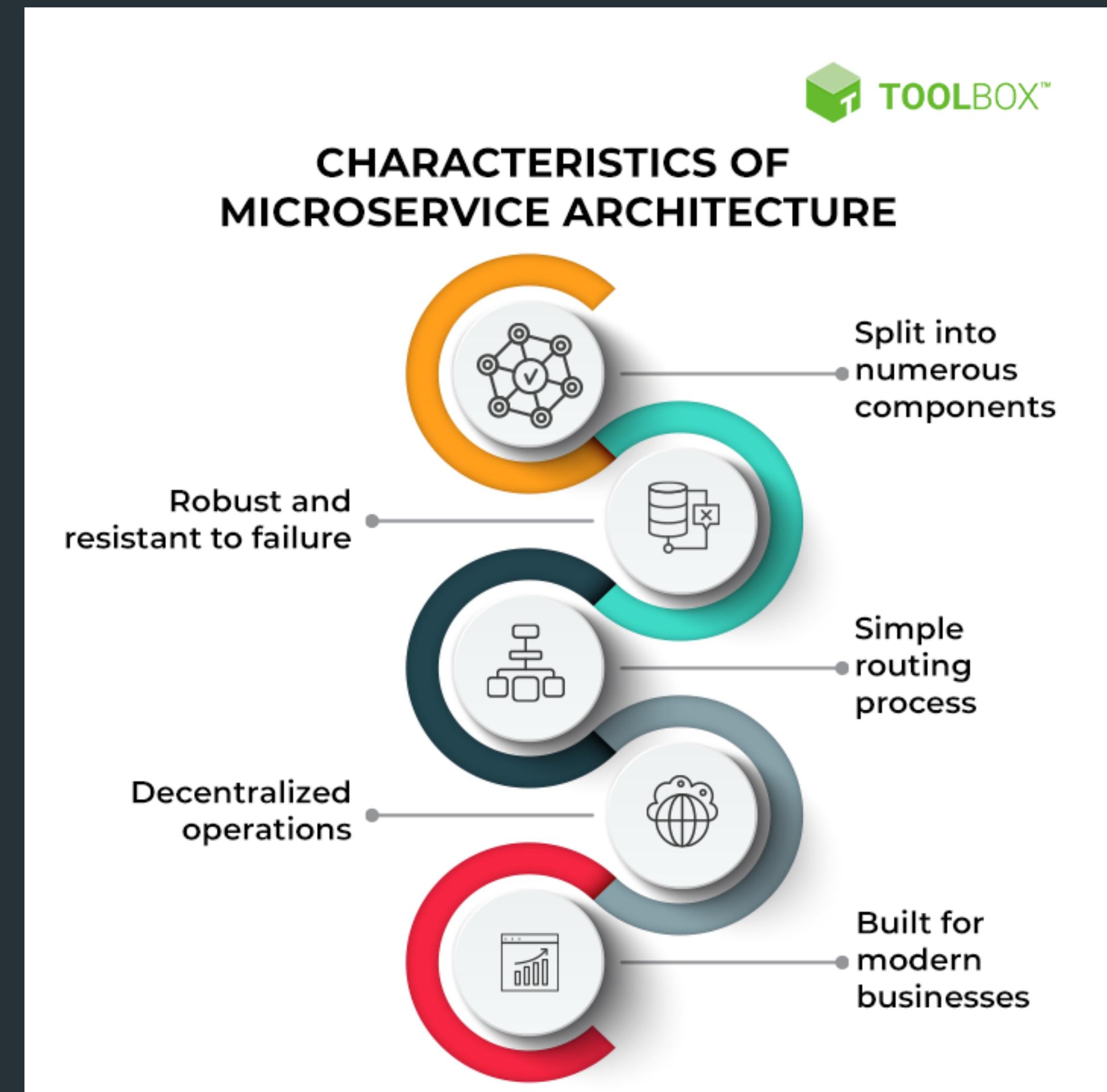
Microservices Key Concepts

- Service boundaries
- API-first design
- Decentralized governance
- **Continuous integration and deployment**
- Resilience and fault-tolerance



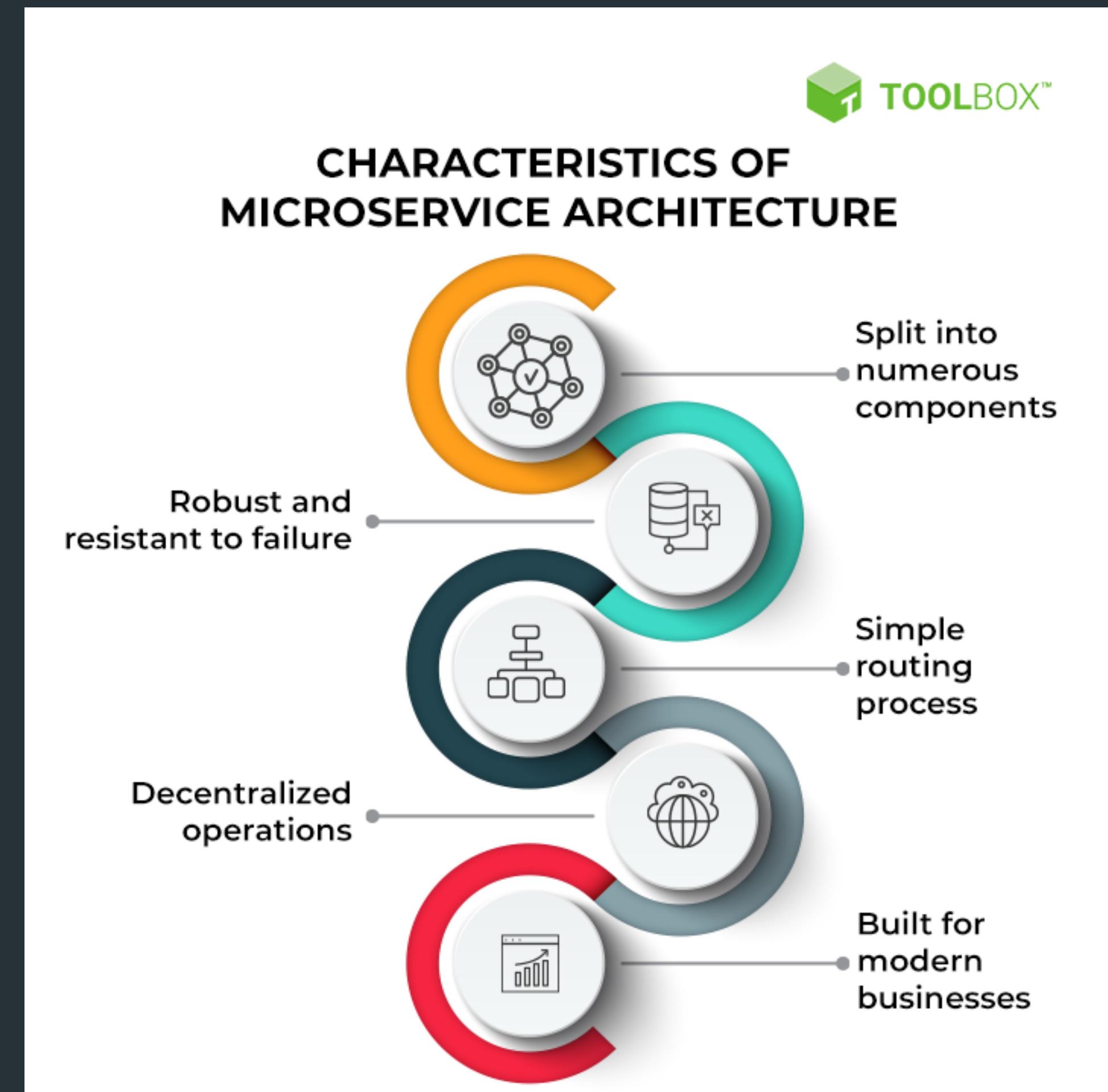
Microservices Key Concepts

- Service boundaries
- API-first design
- Decentralized governance
- Continuous integration and deployment
- **Resilience and fault-tolerance**



Microservices Key Concepts

- Service boundaries
- API-first design
- Decentralized governance
- Continuous integration and deployment
- **Resilience and fault-tolerance**

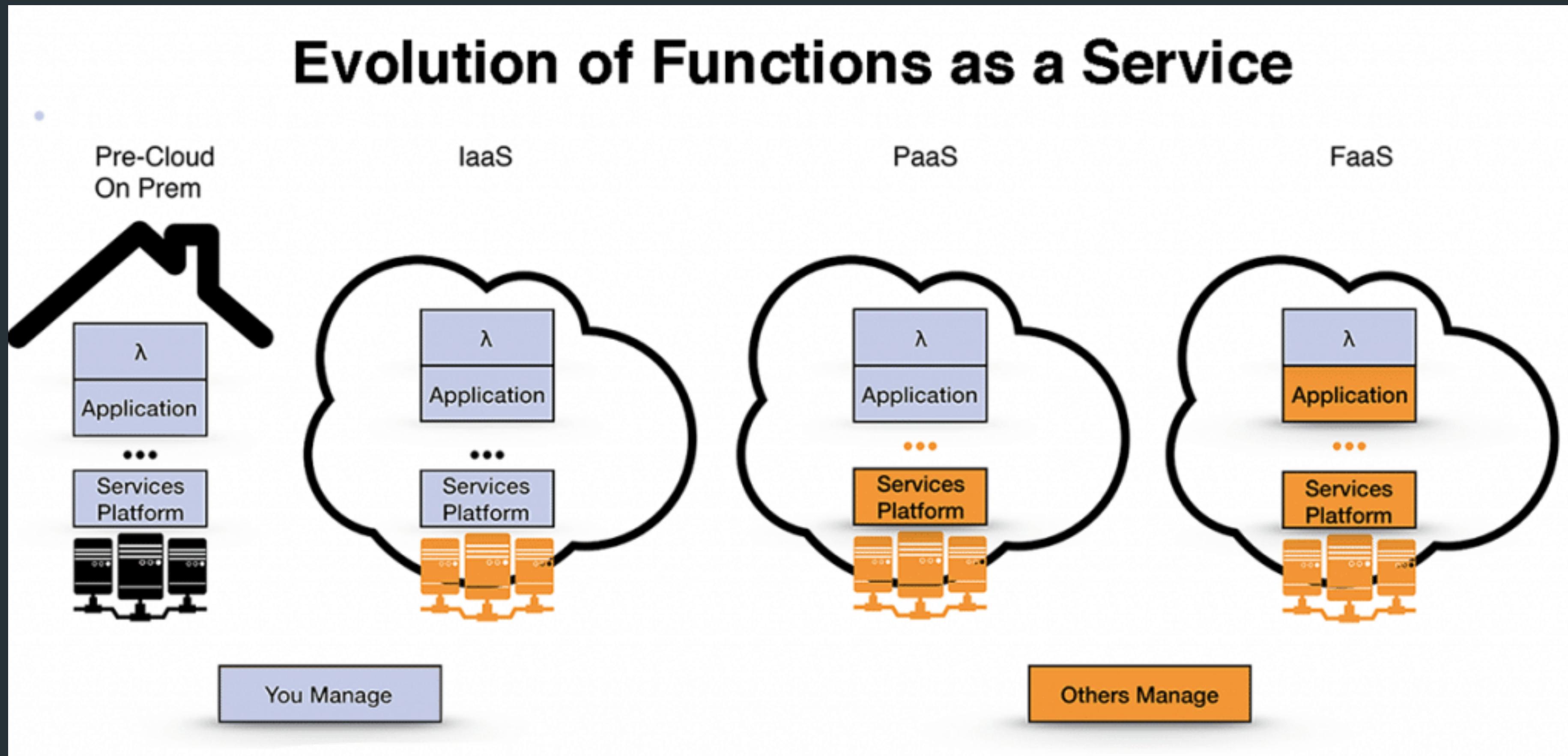


Serverless Computing

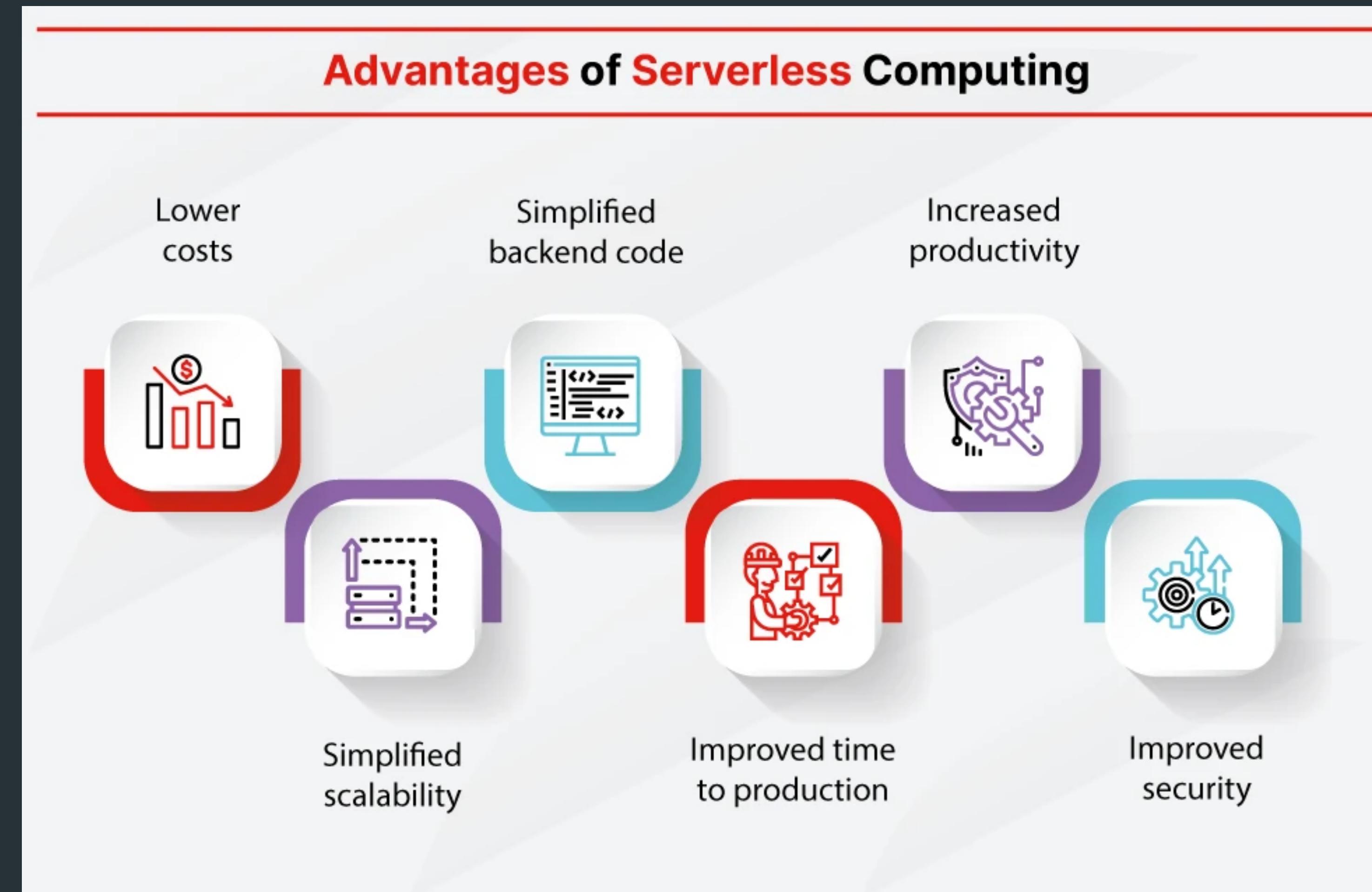
Serverless Computing



Serverless Computing



Serverless Computing



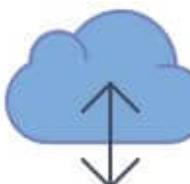
Serverless Key Concepts

- Event-driven architecture
- Stateless functions
- Pay-per-use billing
- Third-party services
- Cold start latency

Main benefits of serverless for business owners



Shorter time to market



Quick deployment



Event-driven project scaling



Pay only for what you use



Reduced costs of running a server

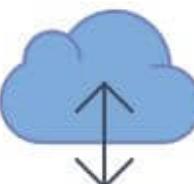
Serverless Key Concepts

- **Event-driven architecture**
- Stateless functions
- Pay-per-use billing
- Third-party services
- Cold start latency

Main benefits of serverless for business owners



Shorter time to market



Quick deployment



Event-driven project scaling



Pay only for what you use



Reduced costs of running a server

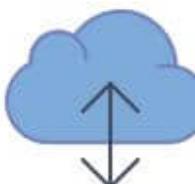
Serverless Key Concepts

- Event-driven architecture
- **Stateless functions**
- Pay-per-use billing
- Third-party services
- Cold start latency

Main benefits of serverless for business owners



Shorter time to market



Quick deployment



Event-driven project scaling



Pay only for what you use



Reduced costs of running a server

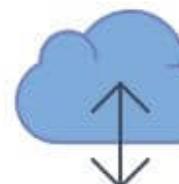
Serverless Key Concepts

- Event-driven architecture
- Stateless functions
- **Pay-per-use billing**
- Third-party services
- Cold start latency

Main benefits of serverless for business owners



Shorter time to market



Quick deployment



Event-driven project scaling



Pay only for what you use



Reduced costs of running a server

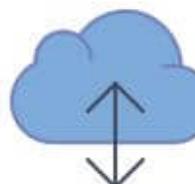
Serverless Key Concepts

- Event-driven architecture
- Stateless functions
- Pay-per-use billing
- **Third-party services**
- Cold start latency

Main benefits of serverless for business owners



Shorter time to market



Quick deployment



Event-driven project scaling



Pay only for what you use



Reduced costs of running a server

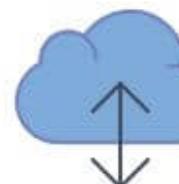
Serverless Key Concepts

- Event-driven architecture
- Stateless functions
- Pay-per-use billing
- Third-party services
- **Cold start latency**

Main benefits of serverless for business owners



Shorter time to market



Quick deployment



Event-driven project scaling



Pay only for what you use



Reduced costs of running a server

Plan

Homework Plan

- Short essay (> 500 words) explaining the basic concepts of web services.
- Simple RESTful web service.
- Design and implement a message-oriented middleware system.

Homework Plan

- Short essay (> 500 words)
**explaining the basic concepts
of web services.**
- Simple RESTful web service.
- Design and implement a
message-oriented middleware
system.

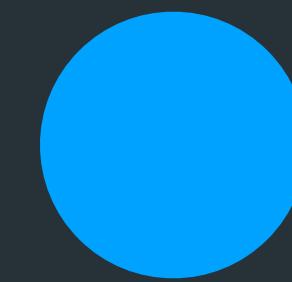
Homework Plan

- Short essay (> 500 words) explaining the basic concepts of web services.
- **Simple RESTful web service.**
- Design and implement a message-oriented middleware system.

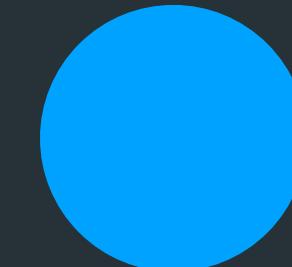
Homework Plan

- Short essay (> 500 words) explaining the basic concepts of web services.
- Simple RESTful web service.
- **Design and implement a message-oriented middleware system.**

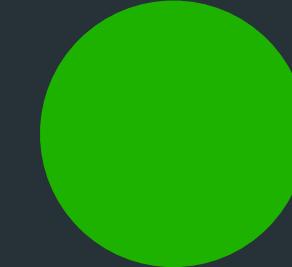
Homework Plan



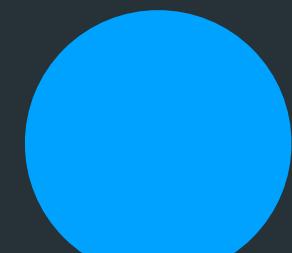
Short essay (> 500 words)
explaining the basic concepts
of web services.



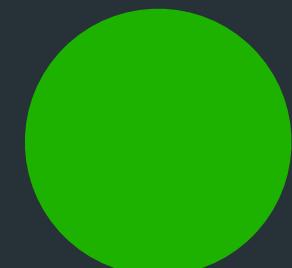
Simple RESTful web service.



Design and implement a
message-oriented middleware
system.



Individual

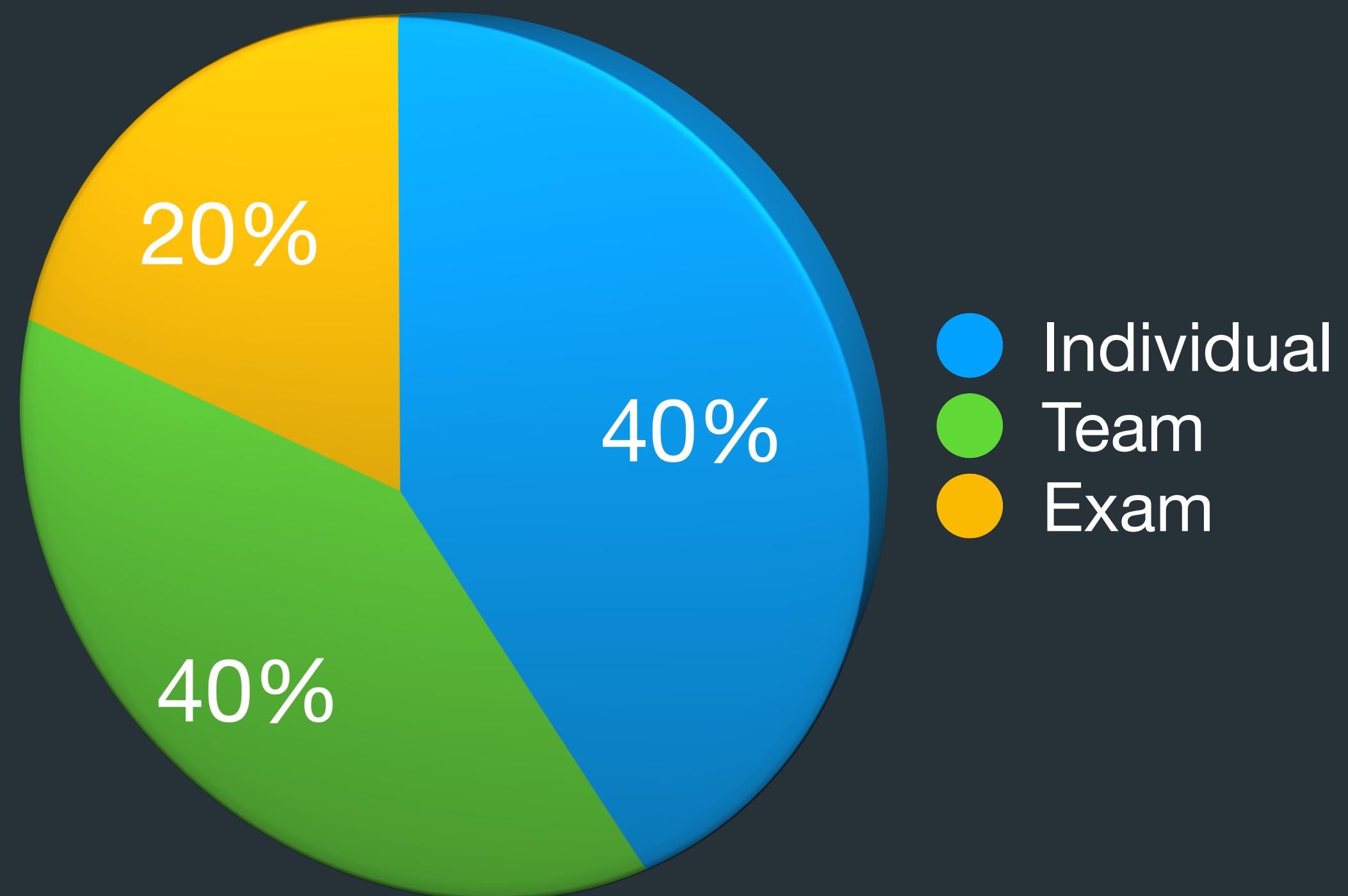


Team of Two

Homework Plan

- Everything should be submitted on GitHub Classroom repository.
- To receive a grade the homework should be presented in class.

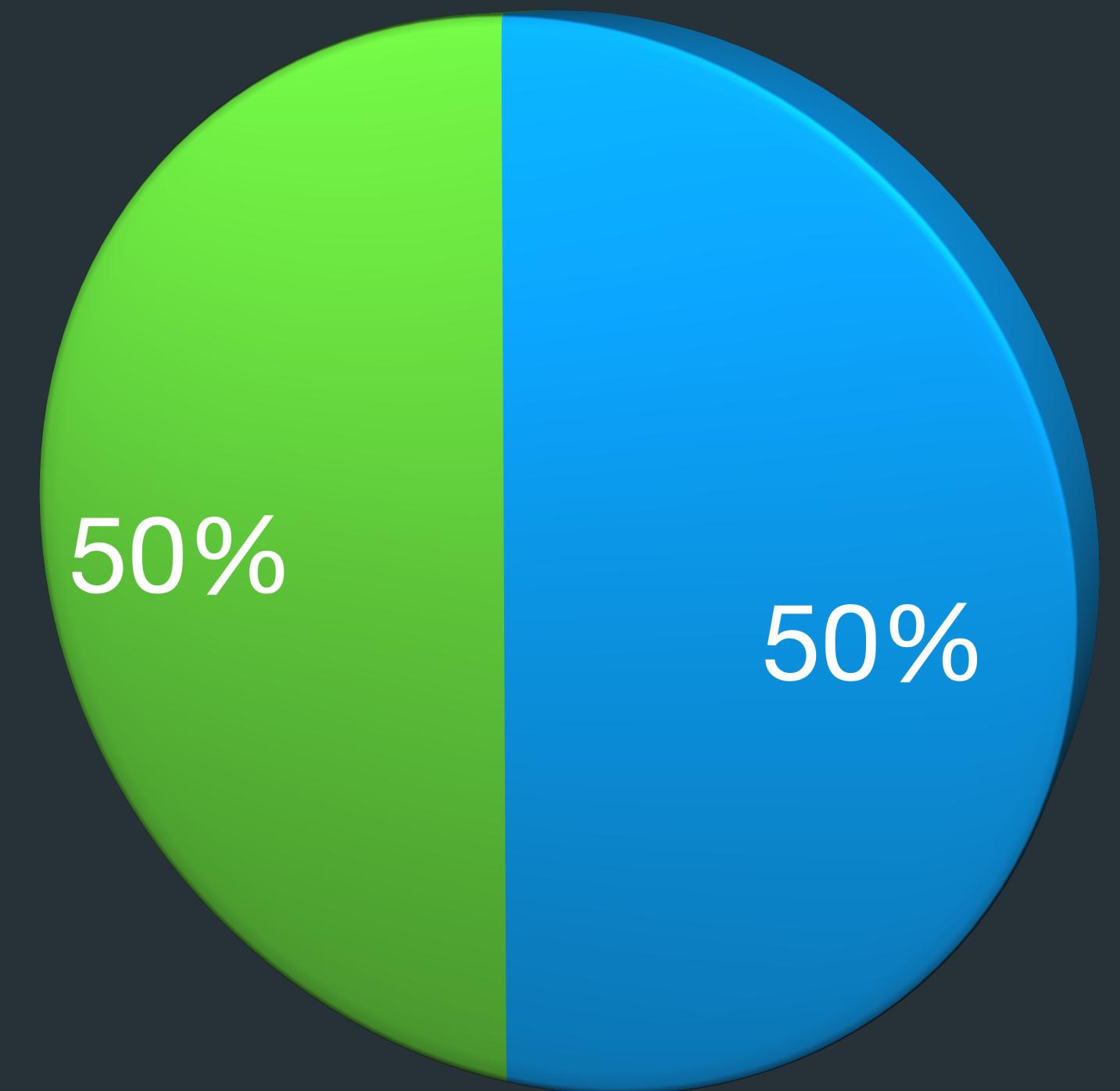
Grades



Grades

Done
before
May 25th

Done
before
Easter



- Individual
- Team
- Exam

Lecture outcomes

- Web services & middleware - essential for distributed systems
- Various architectures & technologies - different benefits & trade-offs
- SOA, Microservices, Serverless - different architectures for web services
- MOM - important for asynchronous communication
- Plan - the course plan.

