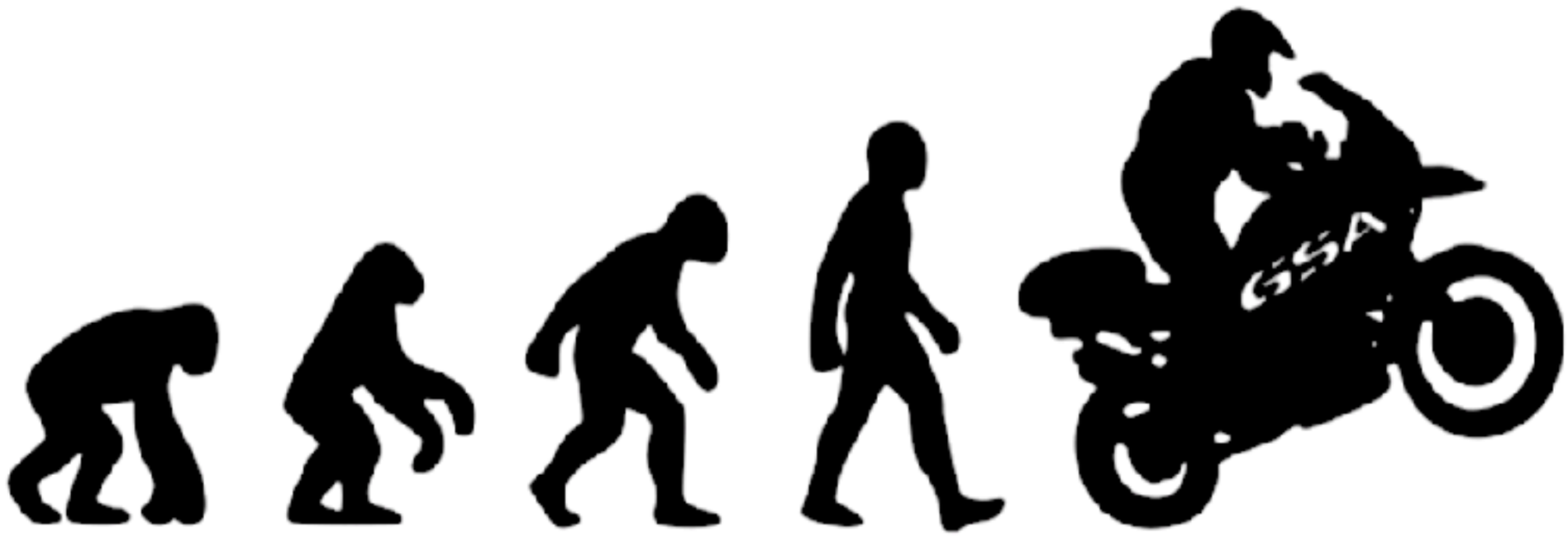
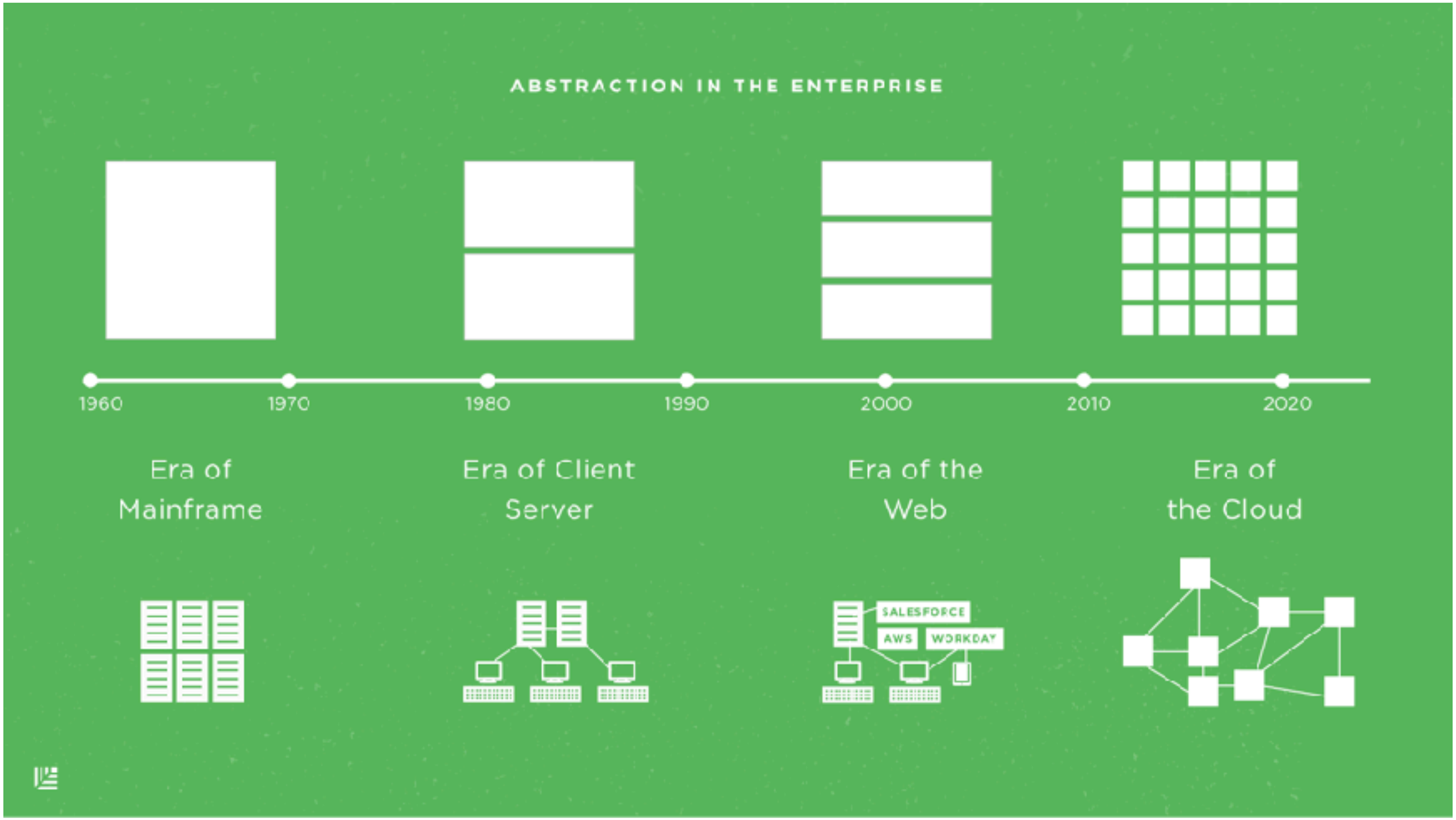
Evolution of Architecture



@somkiat

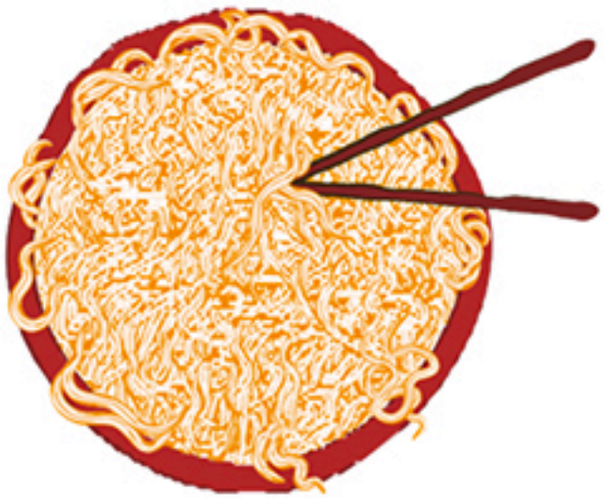
Evolution of Architecture



Developer’s perspective

1990s and earlier

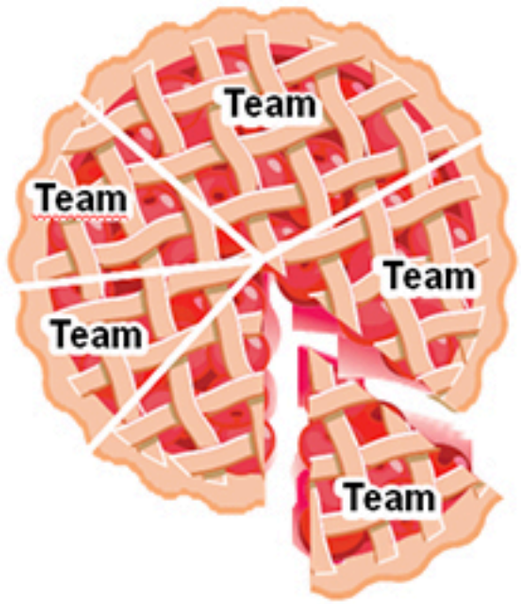
Pre-SOA (monolithic)
Tight coupling



For a monolith to change, all must agree on each change. Each change has unanticipated effects requiring careful testing beforehand.

2000s

Traditional SOA
Looser coupling



Elements in SOA are developed more autonomously but must be coordinated with others to fit into the overall design.

2010s

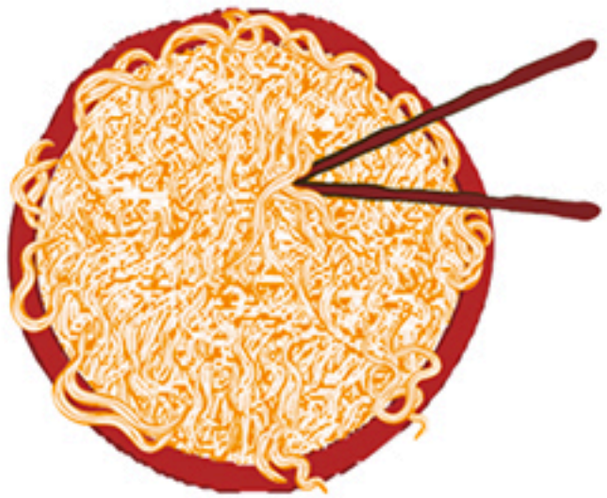
Microservices
Decoupled

Developers can create and activate new microservices without prior coordination with others. Their adherence to MSA principles makes continuous delivery of new or modified services possible.

Developer’s perspective

1990s and earlier

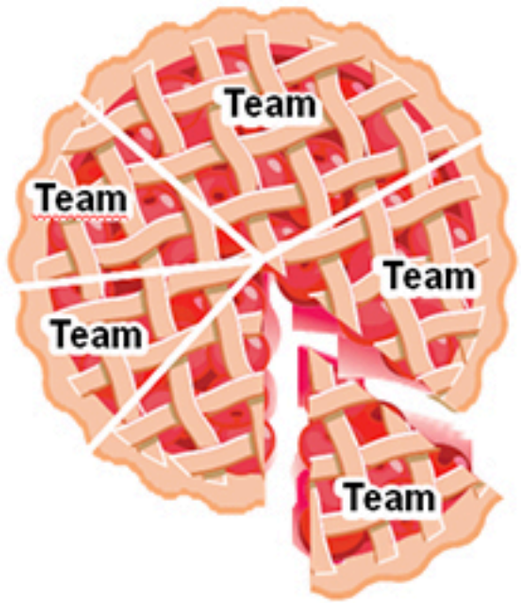
Pre-SOA (monolithic)
Tight coupling



For a monolith to change, all must agree on each change. Each change has unanticipated effects requiring careful testing beforehand.

2000s

Traditional SOA
Looser coupling



Elements in SOA are developed more autonomously but must be coordinated with others to fit into the overall design.

2010s

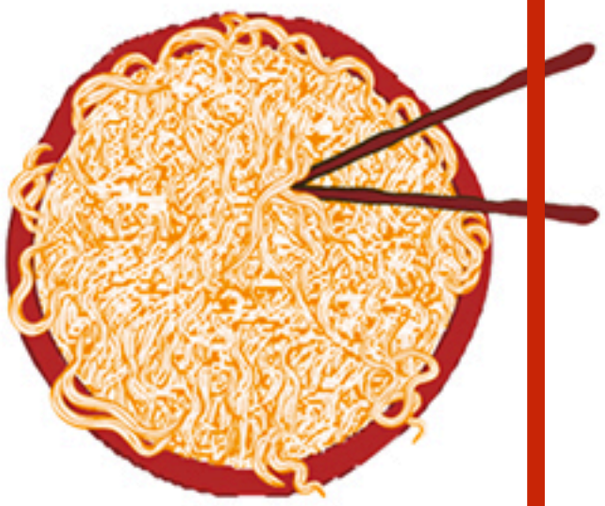
Microservices
Decoupled

Developers can create and activate new microservices without prior coordination with others. Their adherence to MSA principles makes continuous delivery of new or modified services possible.

Developer's perspective

1990s and earlier

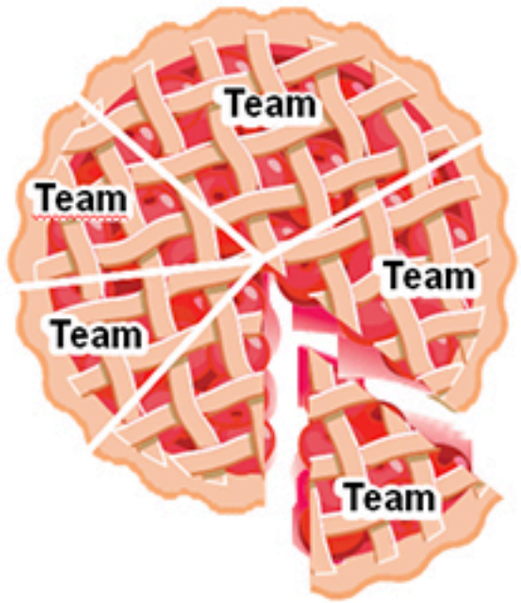
Pre-SOA (monolithic)
Tight coupling



For a monolith to change, all must agree on each change. Each chang has unanticipated effects requiring careful testing beforehand.

2000s

Traditional SOA
Looser coupling



Elements in SOA are developed more autonomously but must be coordinated with others to fit into the overall design.

2010s

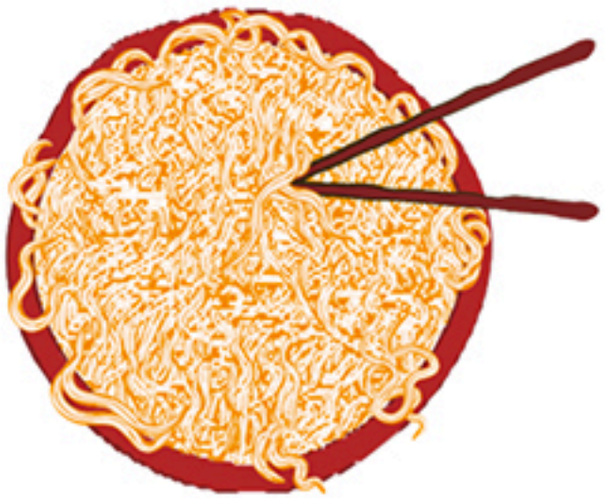
Microservices
Decoupled

Developers can create and activate new microservices without prior coordination with others. Their adherence to MSA principles makes continuous delivery of new or modified services possible.

Developer's perspective

1990s and earlier

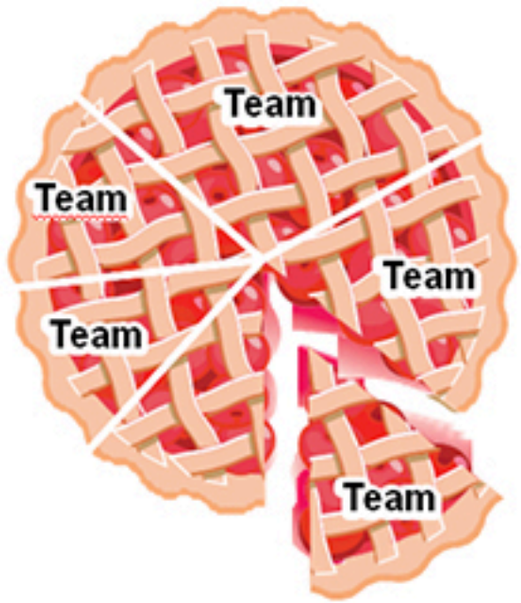
Pre-SOA (monolithic)
Tight coupling



For a monolith to change, all must agree on each change. Each change has unanticipated effects requiring careful testing beforehand.

2000s

Traditional SOA
Looser coupling



Elements in SOA are developed more autonomously but must be coordinated with others to fit into the overall design.

2010s

Microservices
Decoupled

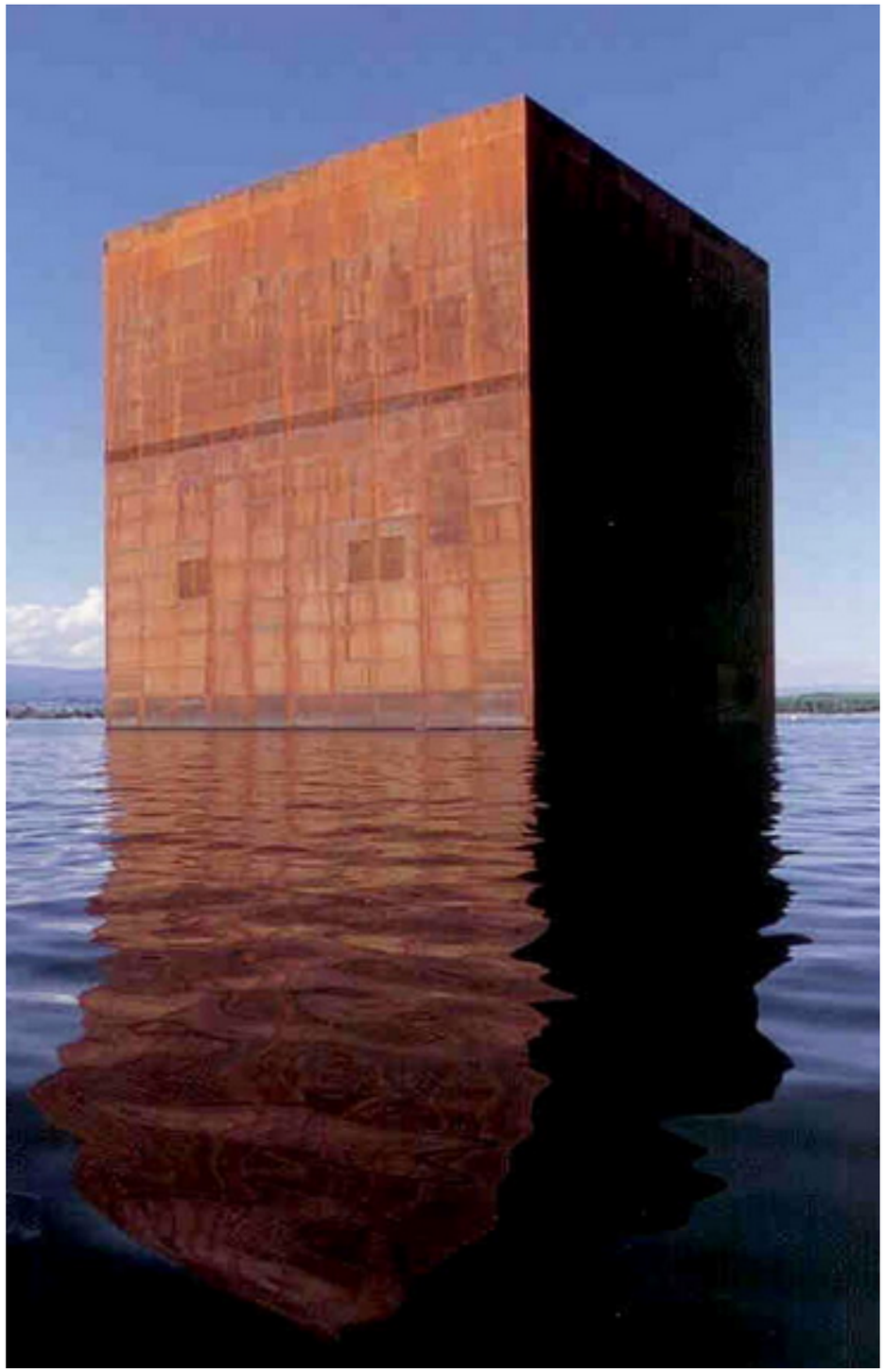
Developers can create and activate new microservices without prior coordination with others. Their adherence to MSA principles makes continuous delivery of new or modified services possible.

Evolution of services

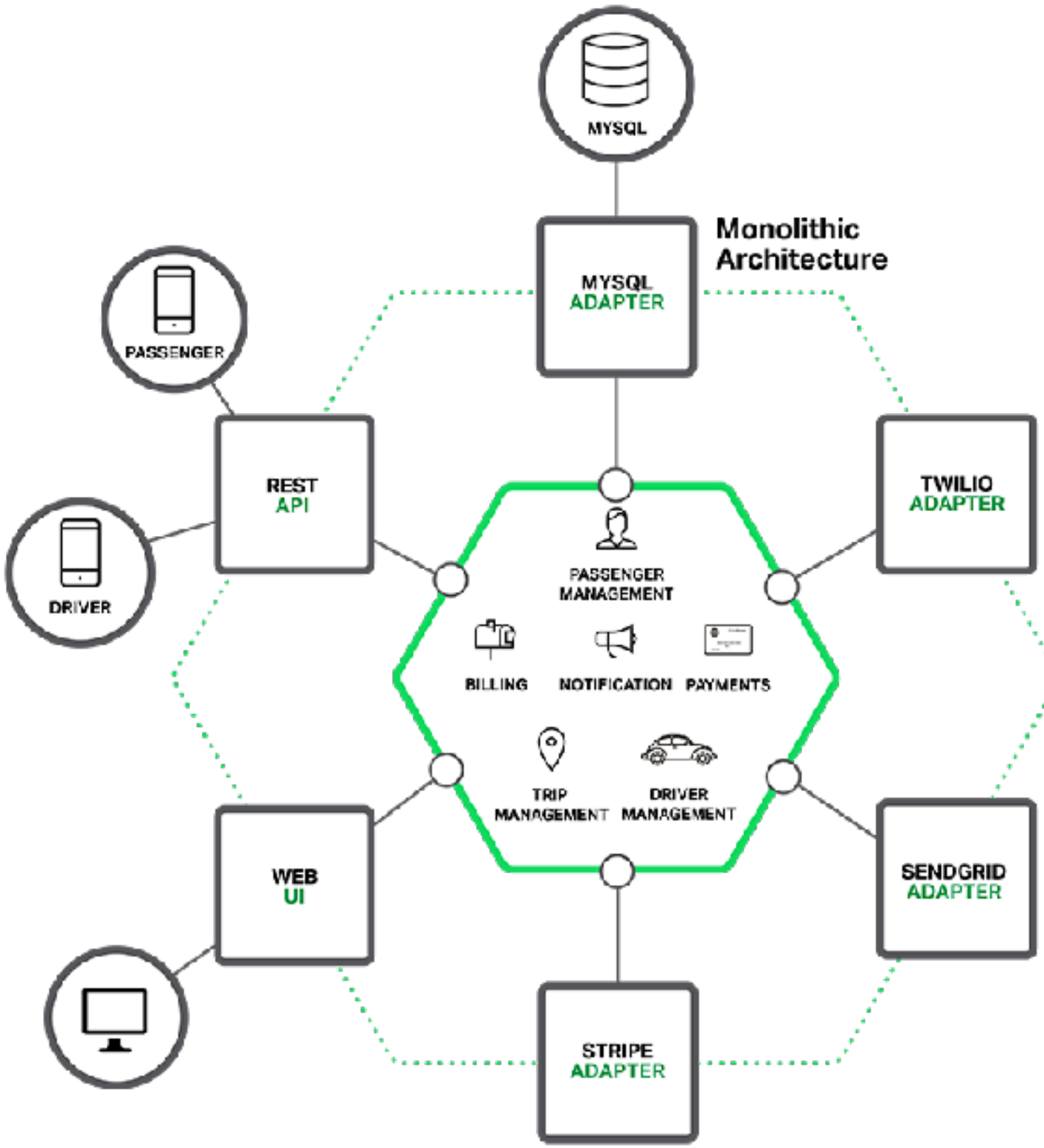
1990s and earlier	2000s	2010s
Coupling		
Pre-SOA (monolithic)	**Traditional SOA**	**Microservices**
Tight coupling	Looser coupling	Decoupled

Monolith

Monolith



Monolith


MYSQL
Monolithic
Architecture
MYSQL
ADAPTER
PASSENGER
REST
API
TWILIO
ADAPTER
DRIVER
PASSENGER
MANAGEMENT
BILLING
NOTIFICATION
PAYMENTS
TRIP
MANAGEMENT
DRIVER
MANAGEMENT
WEB
UI
SENDGRID
ADAPTER
STRIPE
ADAPTER

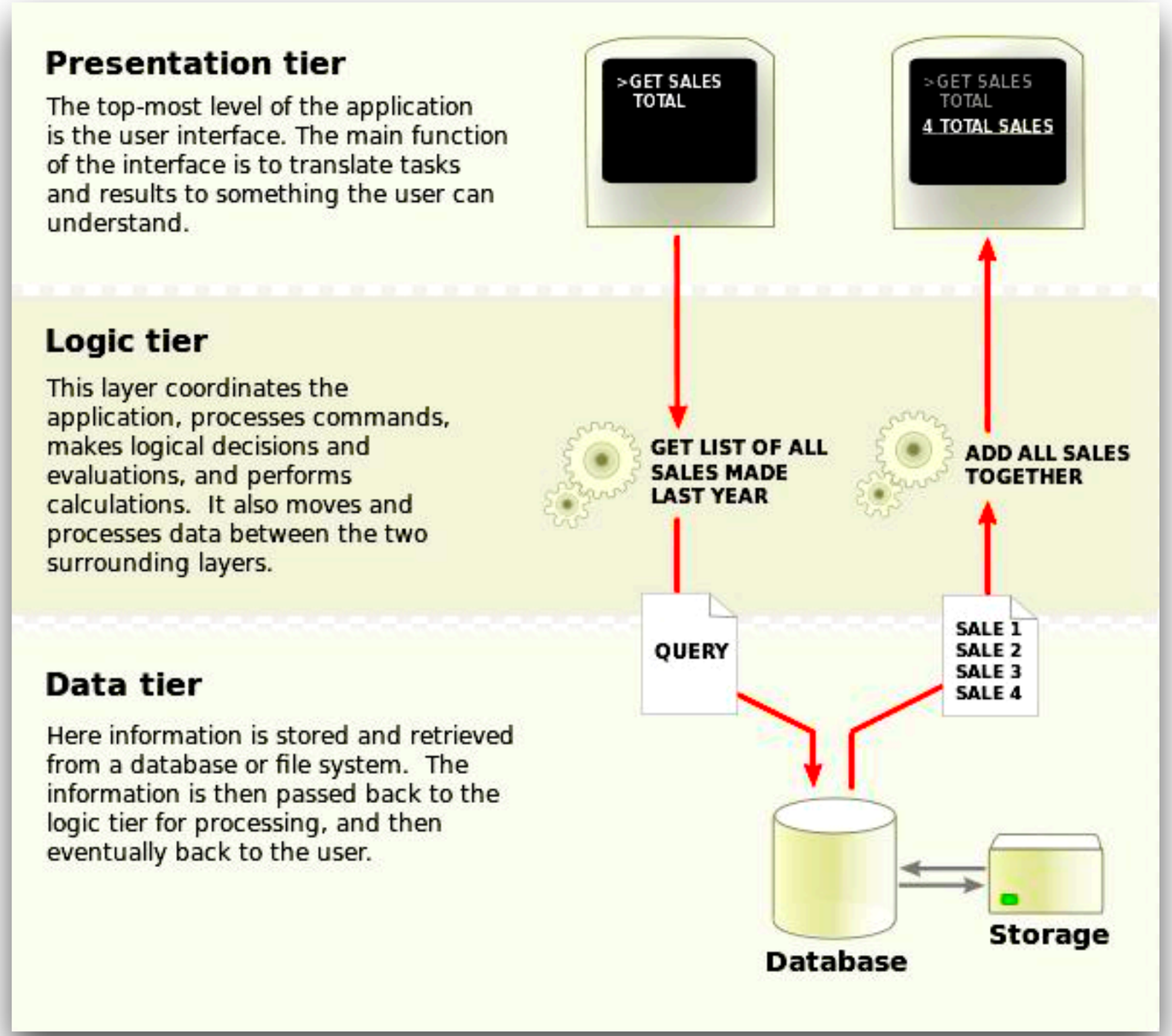
Pros & Cons

?

N-tiers

N-tiers

N-tiers


Presentation tier
The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.
>GET SALES TOTAL
>GET SALES TOTAL
4 TOTAL SALES
Logic tier
This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.
GET LIST OF ALL SALES MADE LAST YEAR
ADD ALL SALES TOGETHER
Data tier
Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.
QUERY
SALE 1
SALE 2
SALE 3
SALE 4
Database
Storage

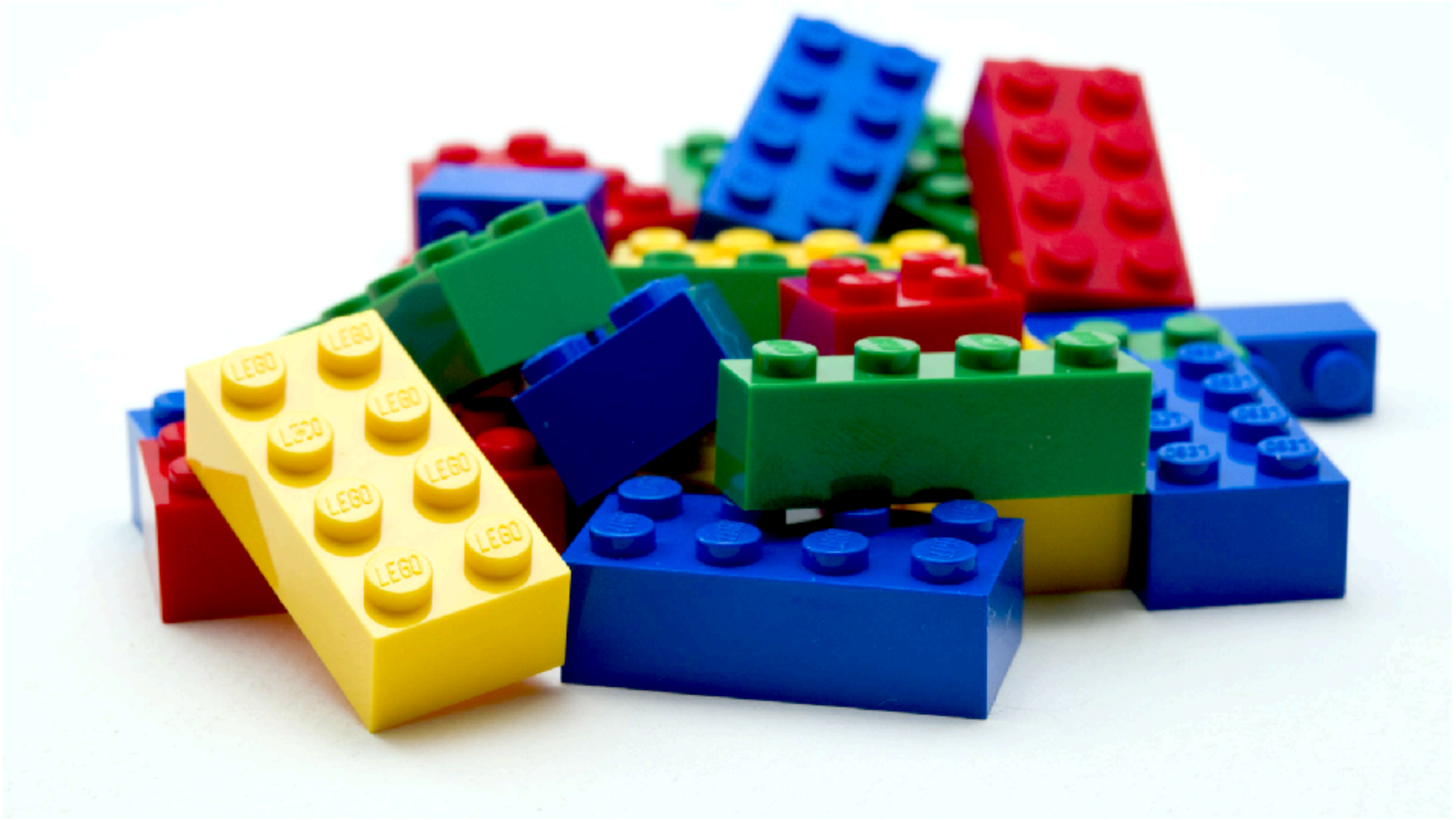
Pros & Cons

?

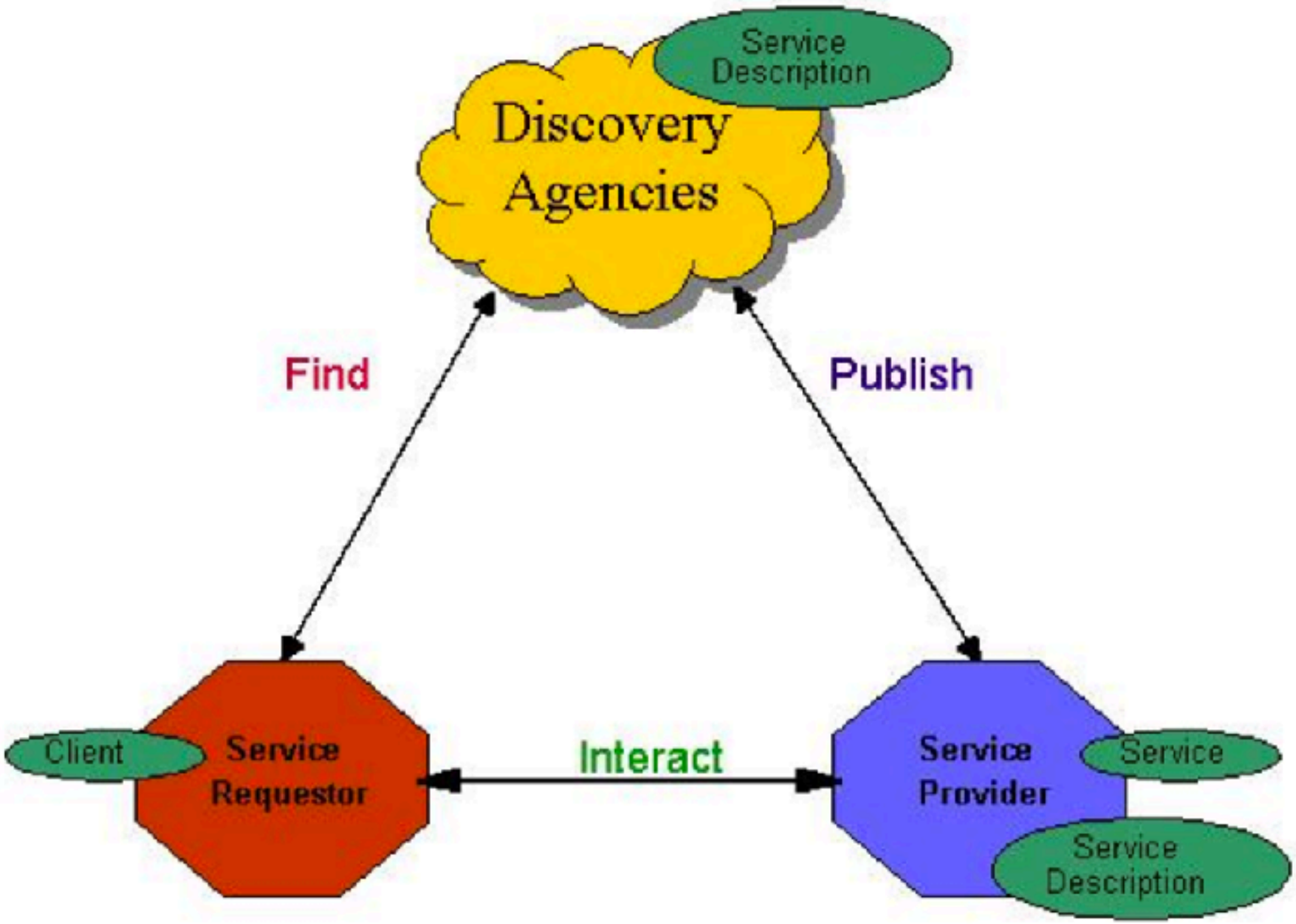
SOA

Service **O**riented **A**rchitecture

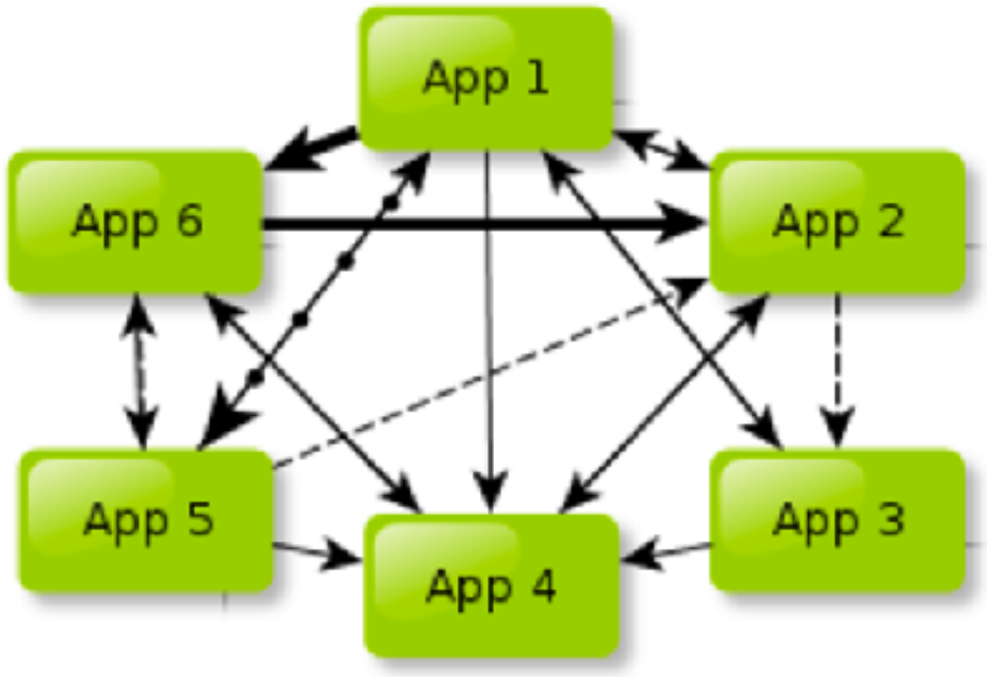
SOA



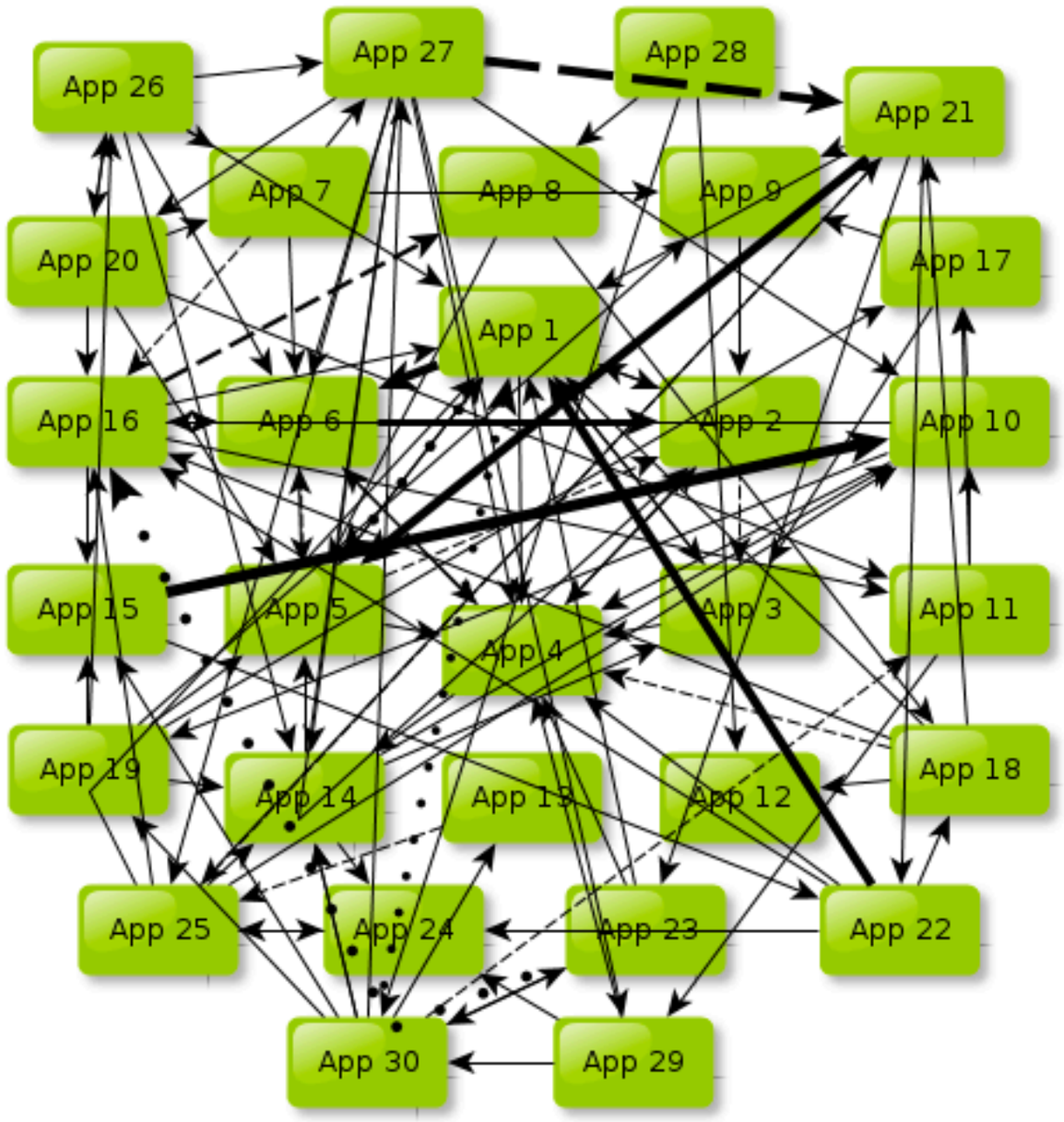
SOA


Service Description
Discovery Agencies
Find
Publish
Client
Service Requestor
Interact
Service Provider
Service
Service Description

SOA ?

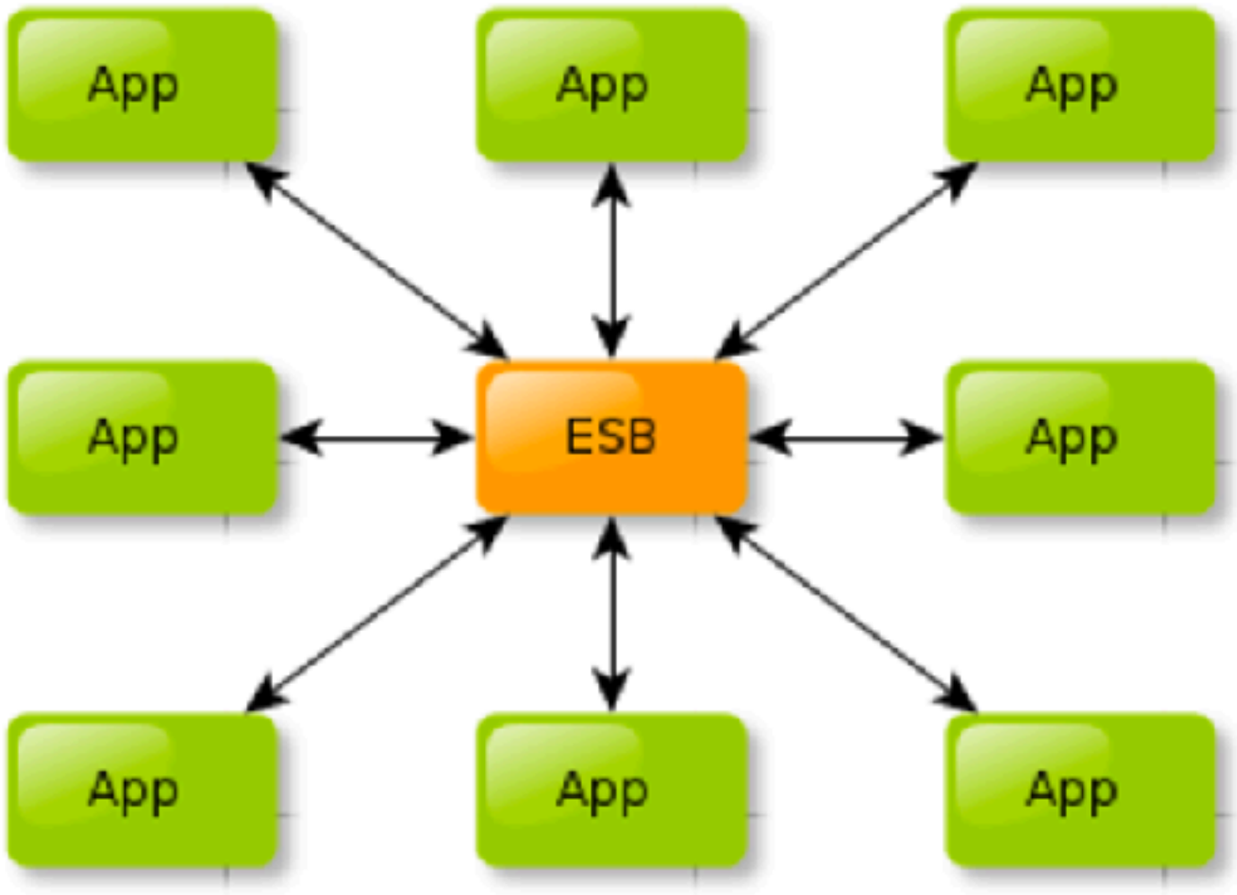


SOA ?

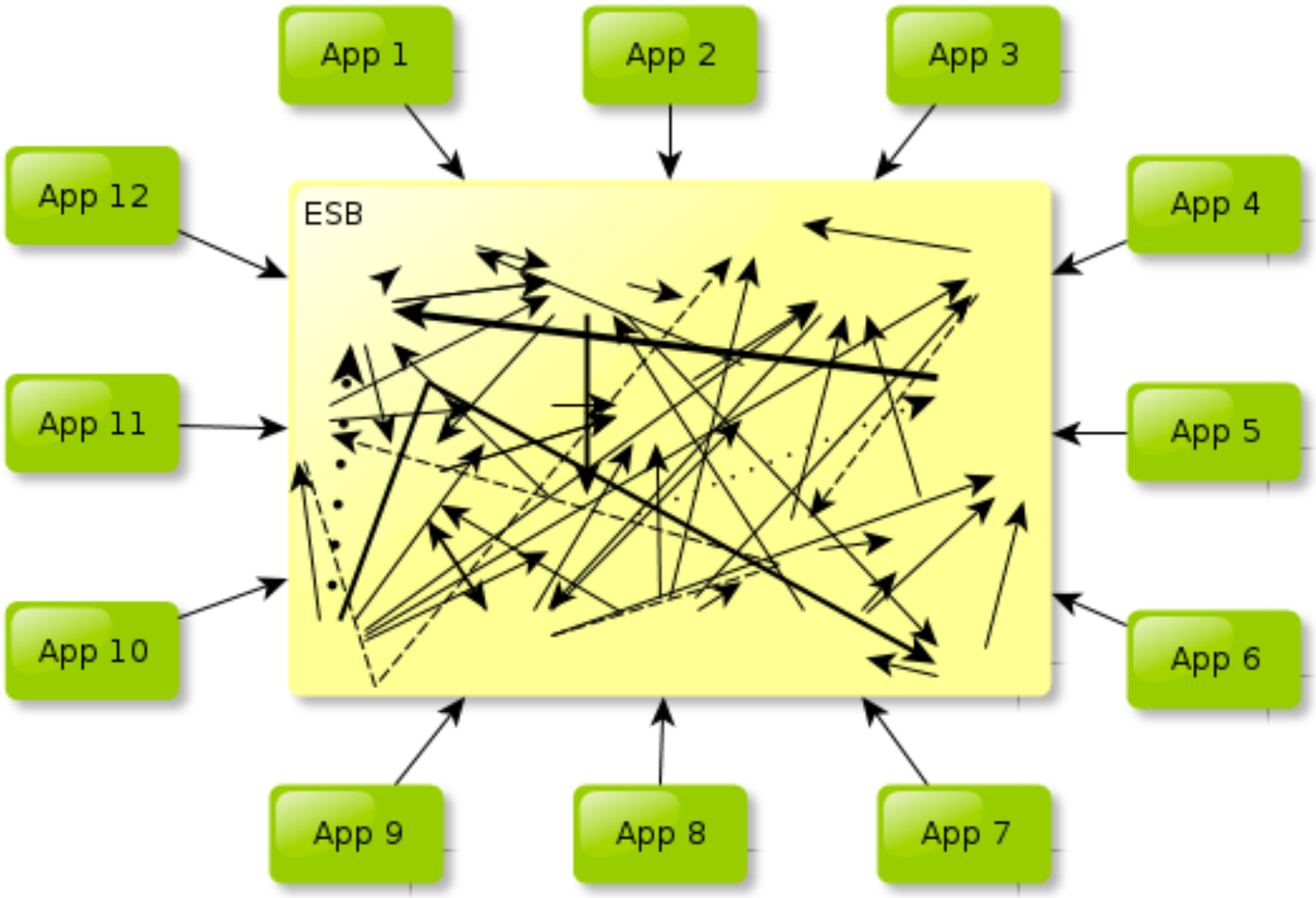

App 26
App 27
App 28
App 21
App 7
App 8
App 9
App 20
App 17
App 1
App 16
App 6
App 2
App 10
App 15
App 5
App 3
App 11
App 4
App 19
App 14
App 13
App 12
App 18
App 25
App 24
App 23
App 22
App 30
App 29

Solution ?

SOA with ESB

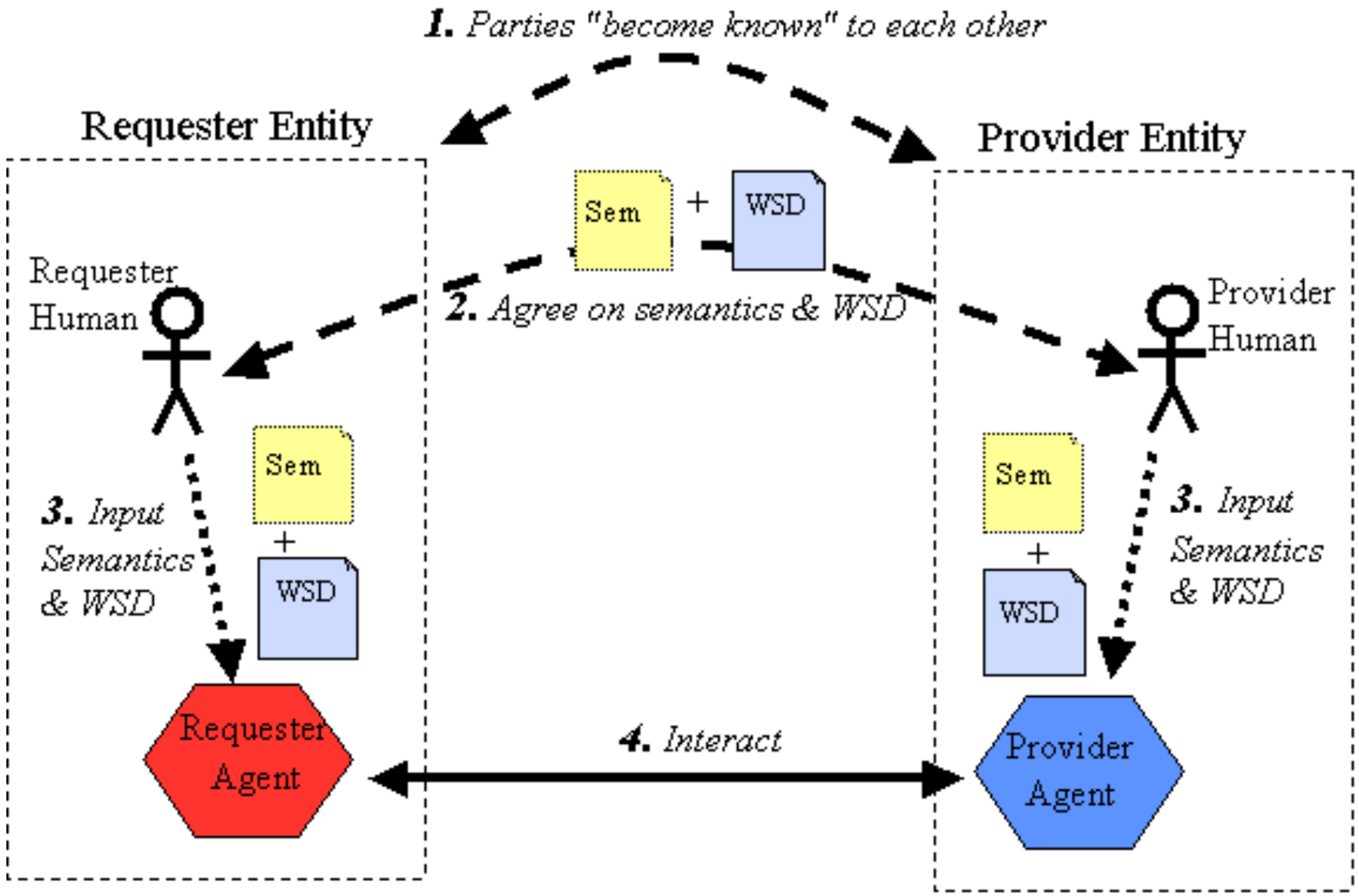


SOA with ESB


App 1
App 2
App 3
App 12
ESB
App 4
App 11
App 5
App 10
App 6
App 9
App 8
App 7

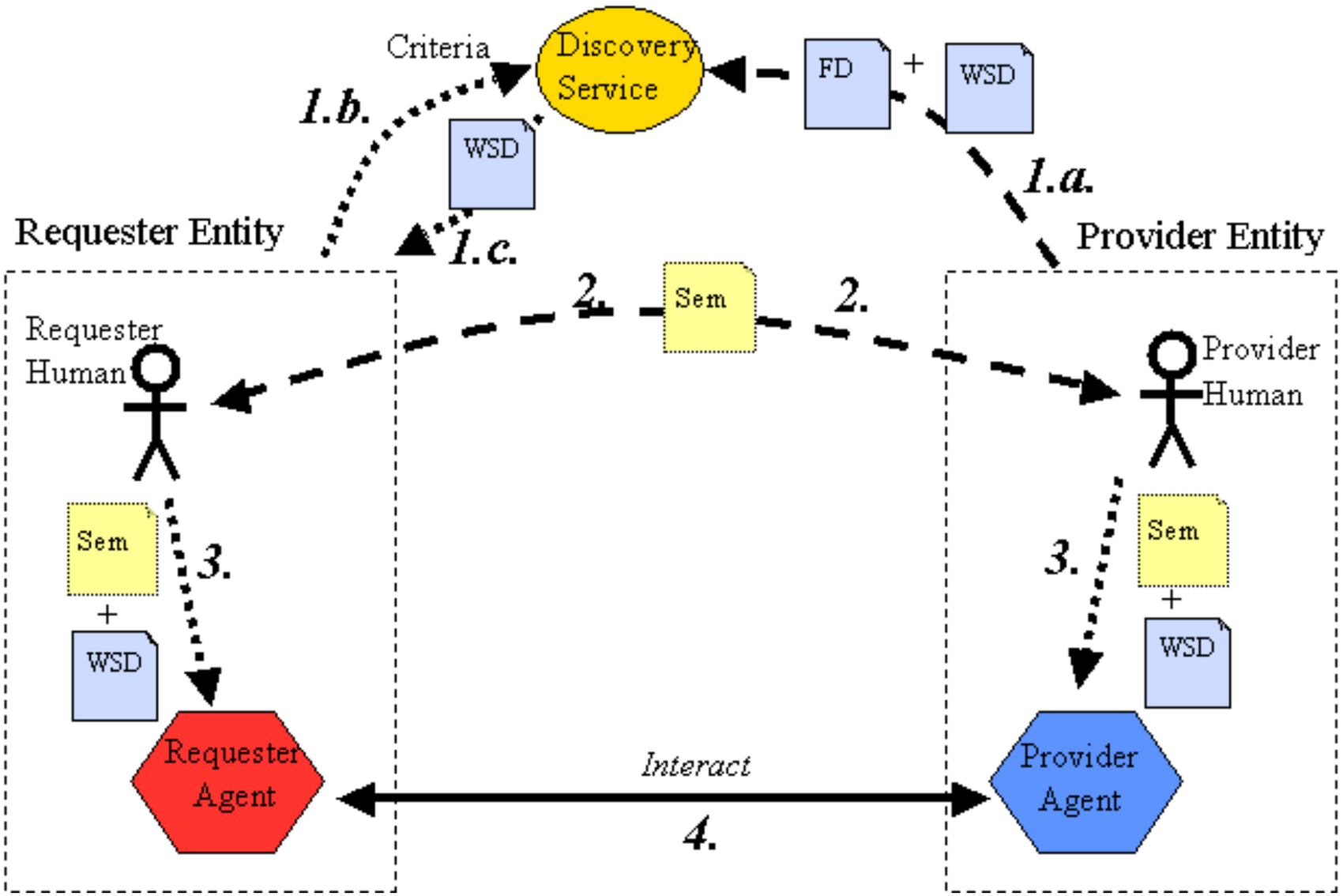
WebService

WebService



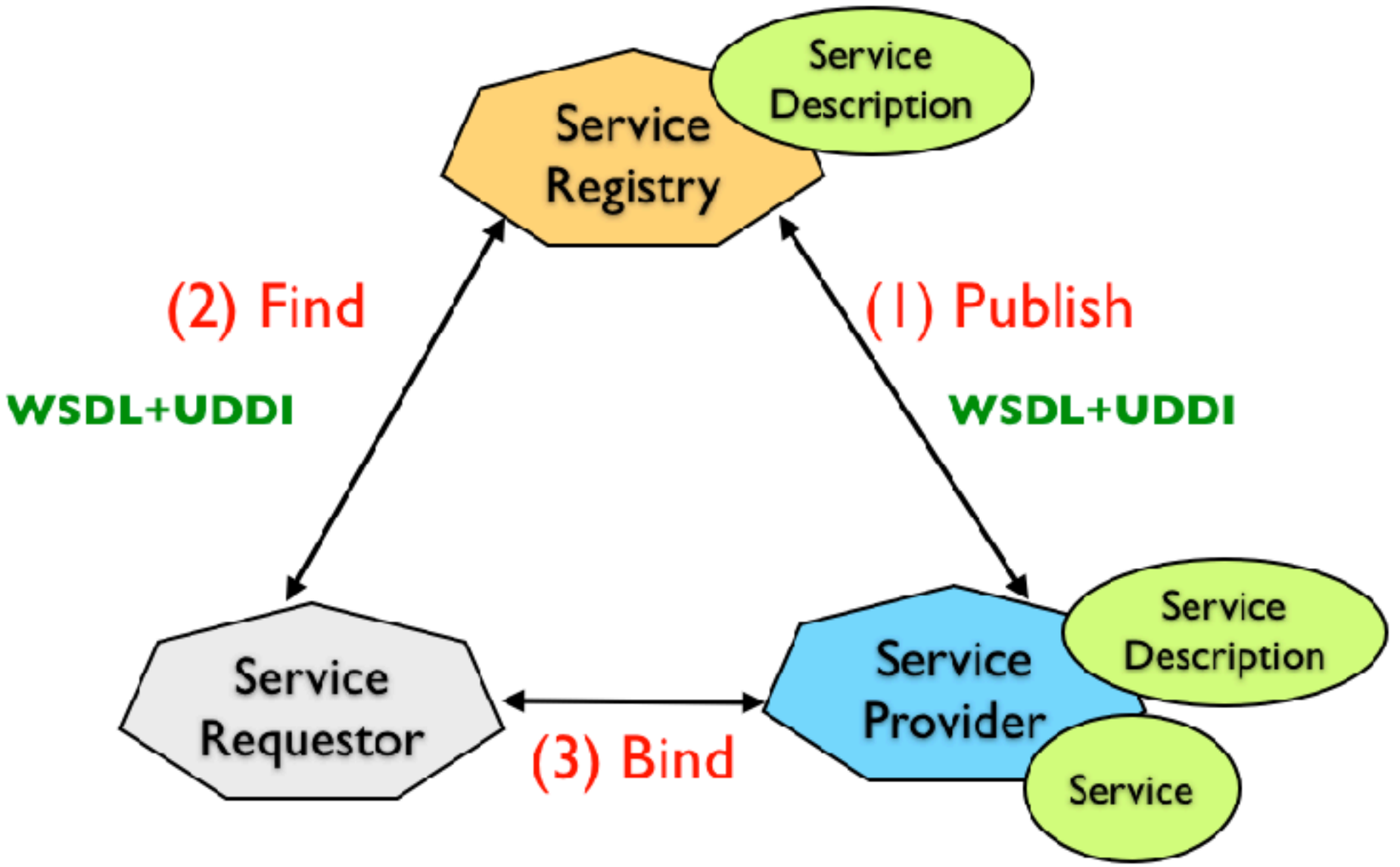
https://www.w3.org/TR/ws-arch

Service Discovery



https://www.w3.org/TR/ws-arch

WebService


Service Description
Service Registry
(2) Find
WSDL+UDDI
(1) Publish
WSDL+UDDI
Service Requestor
(3) Bind
Service Provider
Service Description
Service

WebService

WSDL

UDDI

SOAP

Pros & Cons

?

REST API

REpresentational **S**tate **T**ransfer

REST

Consider "Martin Lawrence" as your data

SOAP

Data + SOAP Standards

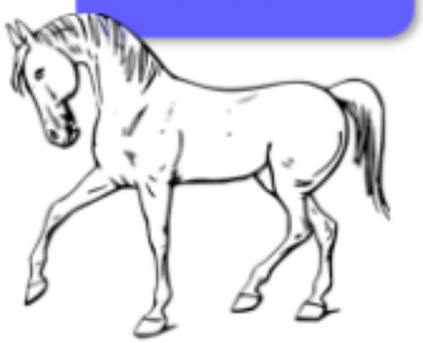
Your data sent to SERVER would become huge as Big mama

REST

Rest is like sending the DATA as such

REST

HTTP



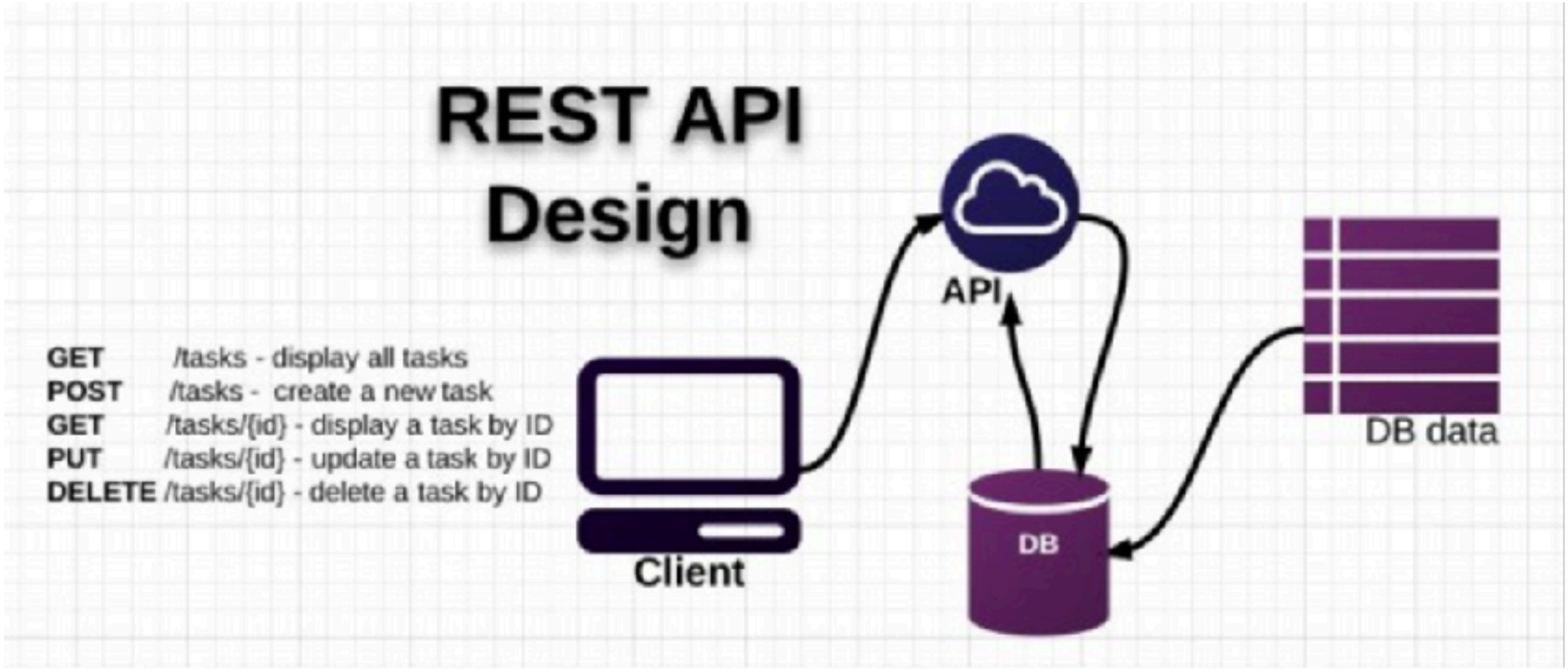
REST

Rides directly on HTTP. Plain and simple. In reality, this is all you need to send data from point A to point B and get the required response. Catch: Until something that represents a service contract is put in to place, it's kinda "anything goes".

SOAP

The coach is your SOAP envelope: it wraps your data. Main strength is the presence of a contract: the WSDL. Gives you the "comfort" of easily generating artifacts. Catch: look at the complexity and added weight.

REST


REST API
Design
GET /tasks - display all tasks
POST /tasks - create a new task
GET /tasks/{id} - display a task by ID
PUT /tasks/{id} - update a task by ID
DELETE /tasks/{id} - delete a task by ID
Client
API
DB
DB data

Monolith

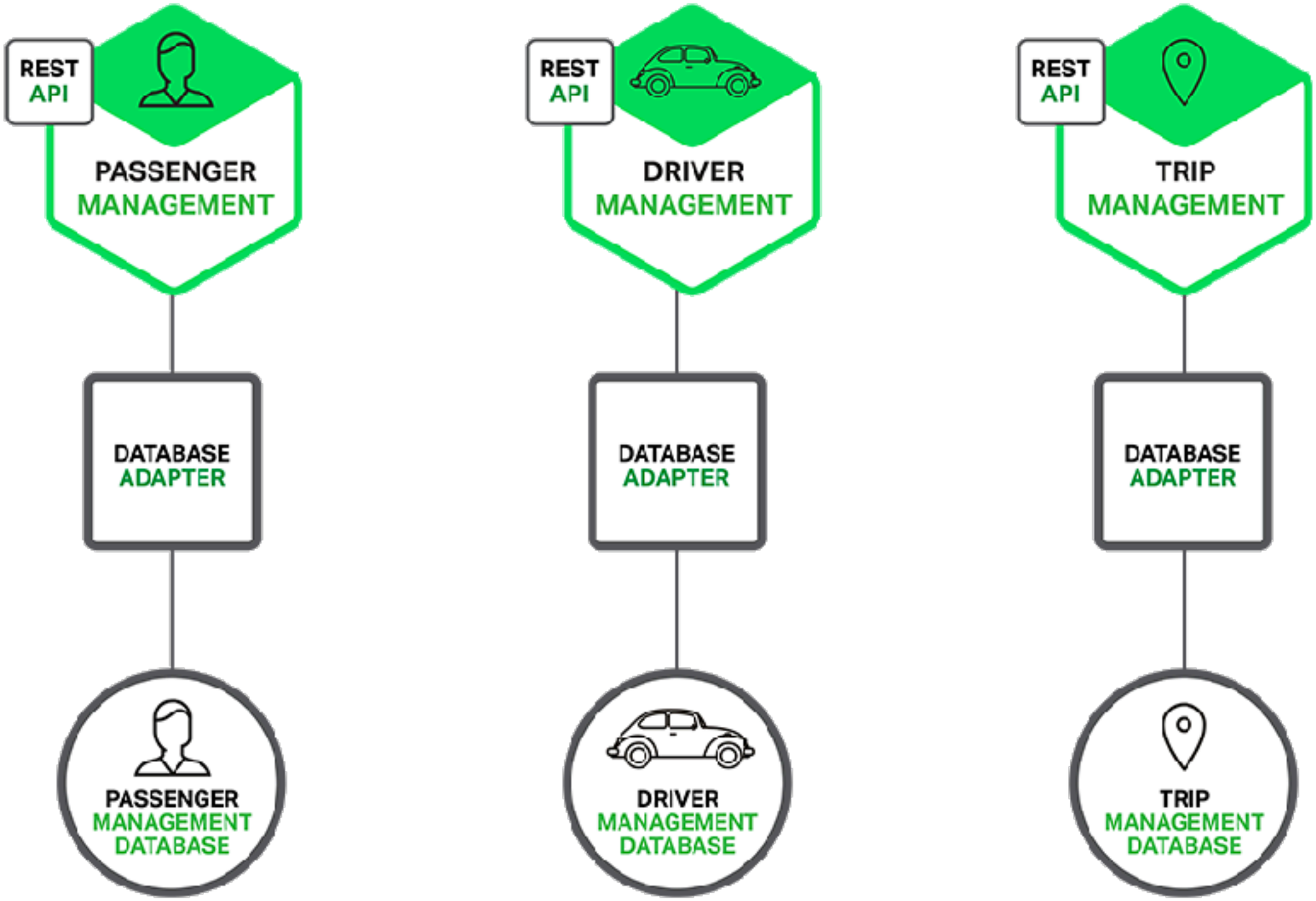
N-tier

SOA/
WebService

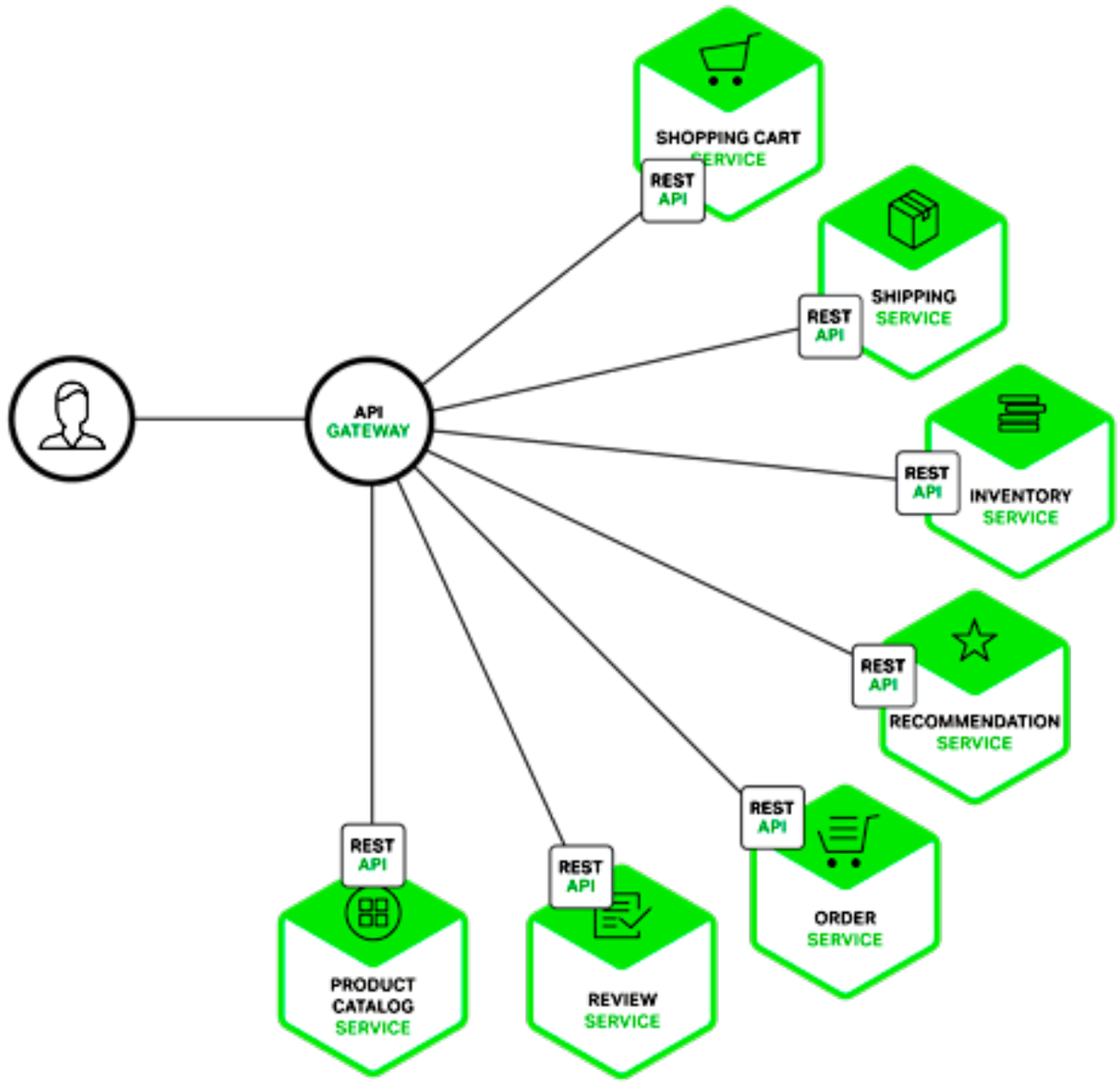
REST API

Microservice

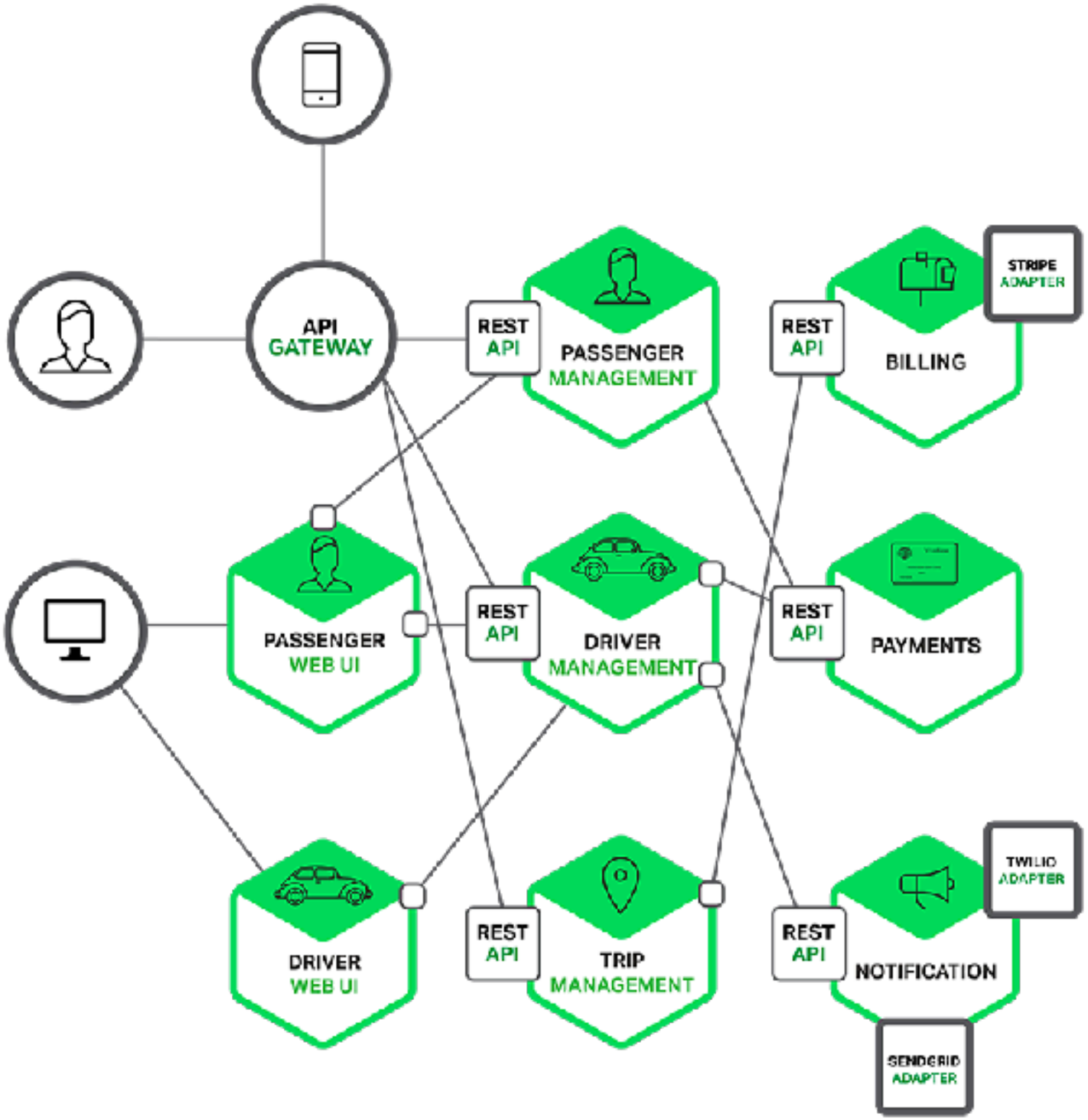
Microservice



Microservice



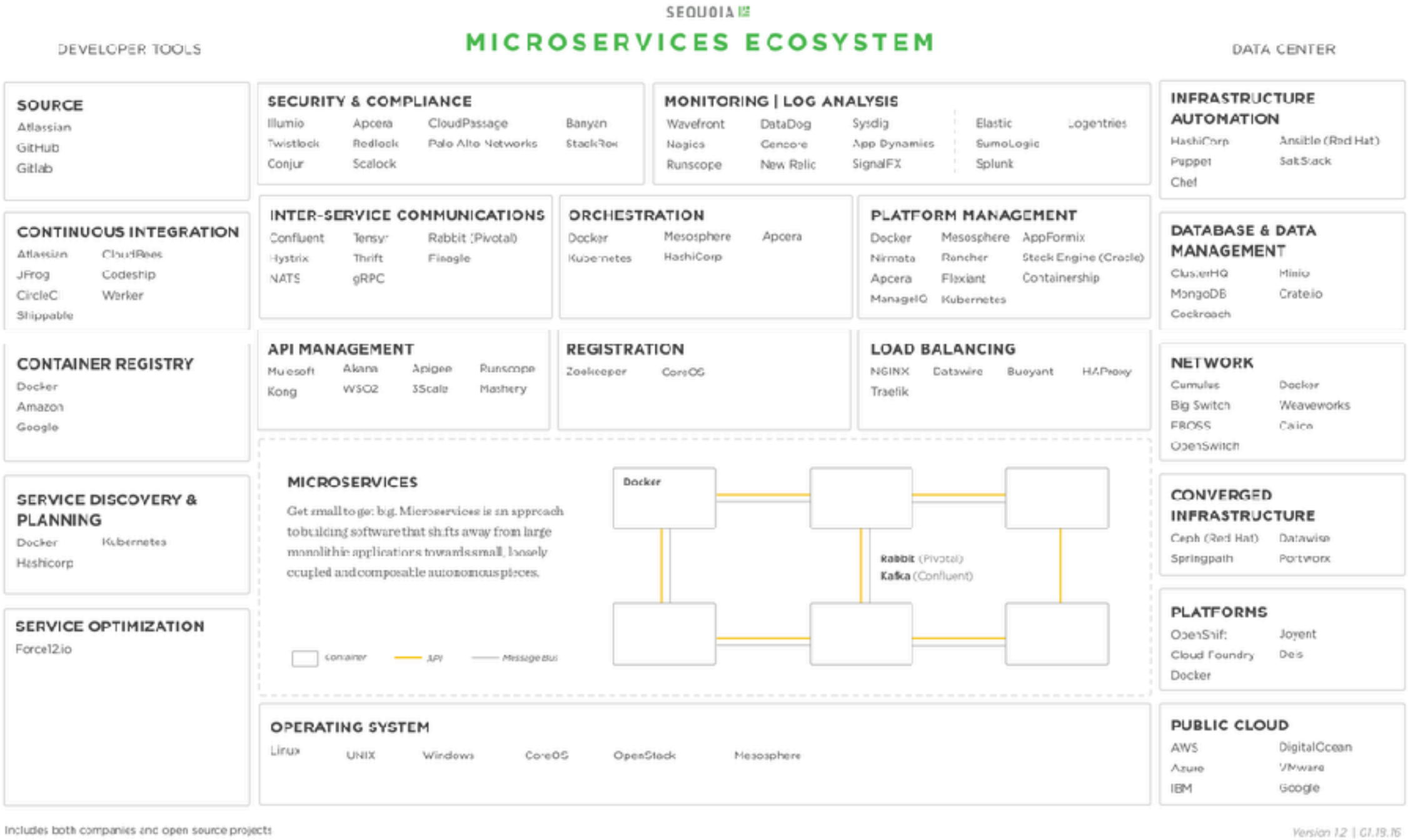
Microservice



Pros & Cons

?

Microservice Ecosystem



https://twitter.com/mcmiller00/status/690894999370633216

Serverless

FaaS

Function as a Service

Let’s workshop