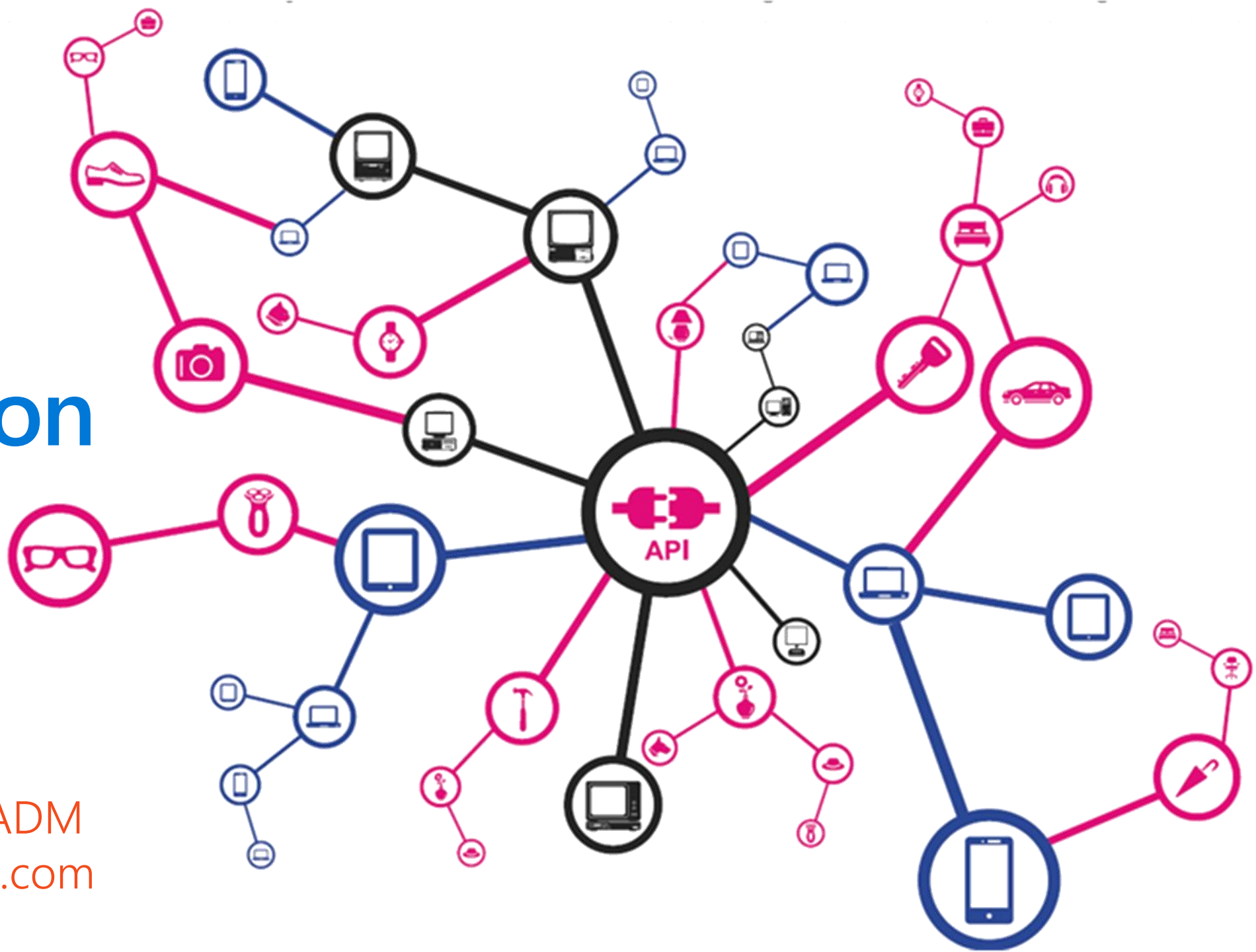


Franck Théolade - ADM  
frtheola@Microsoft.com



# Agenda

Brainstorming

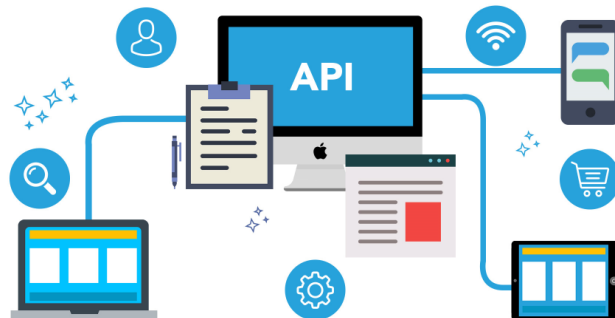
What is an API?

Comfort Break  
~ 5-10 minutes

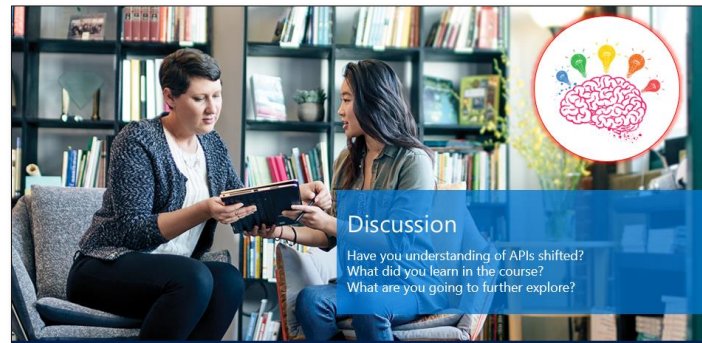
Why/When To Use an

{ API }

Comfort Break  
~ 5-10 minutes



API Testing



Discussion

Have your understanding of APIs shifted?  
What did you learn in the course?  
What are you going to further explore?

Discussion | Your feedback and next actions...

API Back to Basics | Wrap Up

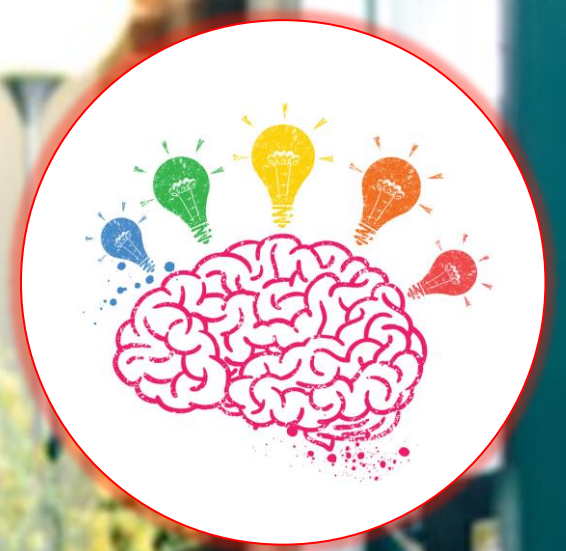
Before we wrap up...  
Please take a moment to give us your feedback...



<https://aka.ms/APIAutomationTesting>

# Brainstorming





What is your understanding of APIs?

5 mins

Team Exercise

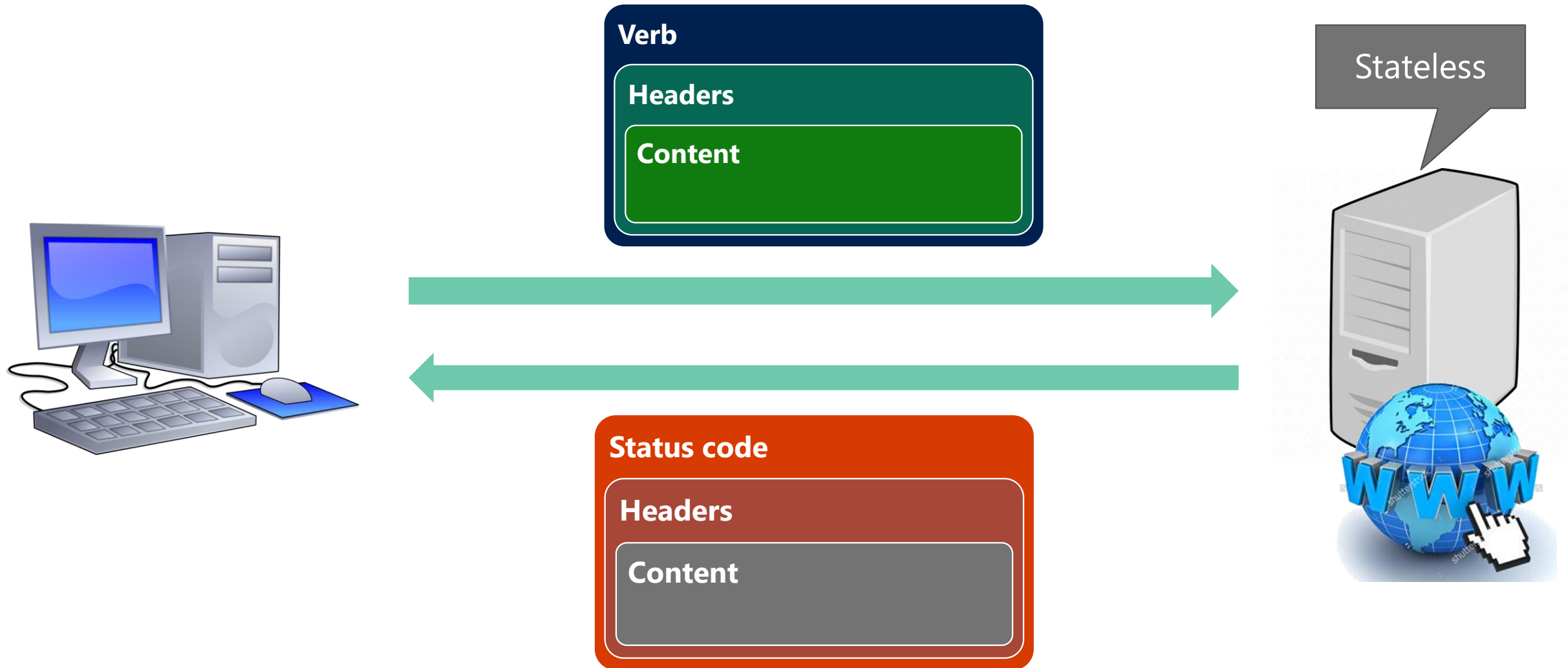
API Back to Basics | Introductions & Logistics



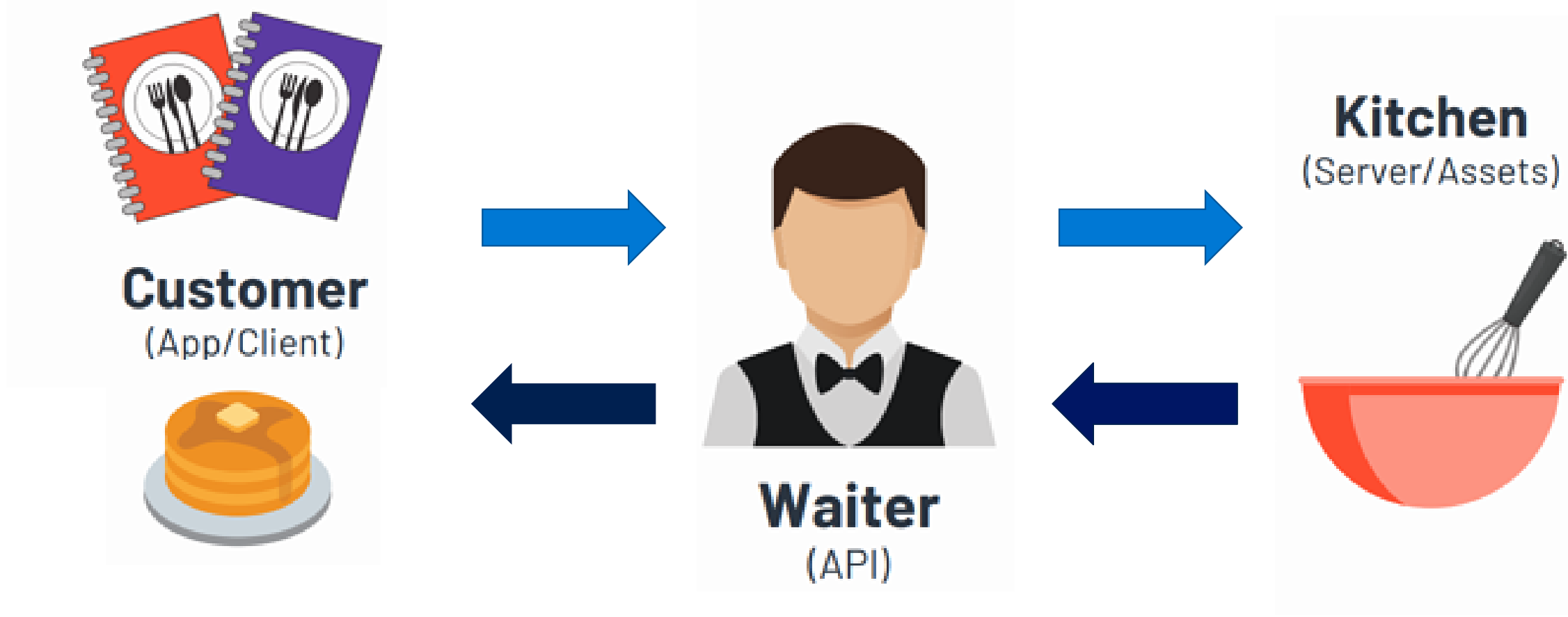
What is an API?



# Understanding the World Wide Web's Data Protocol

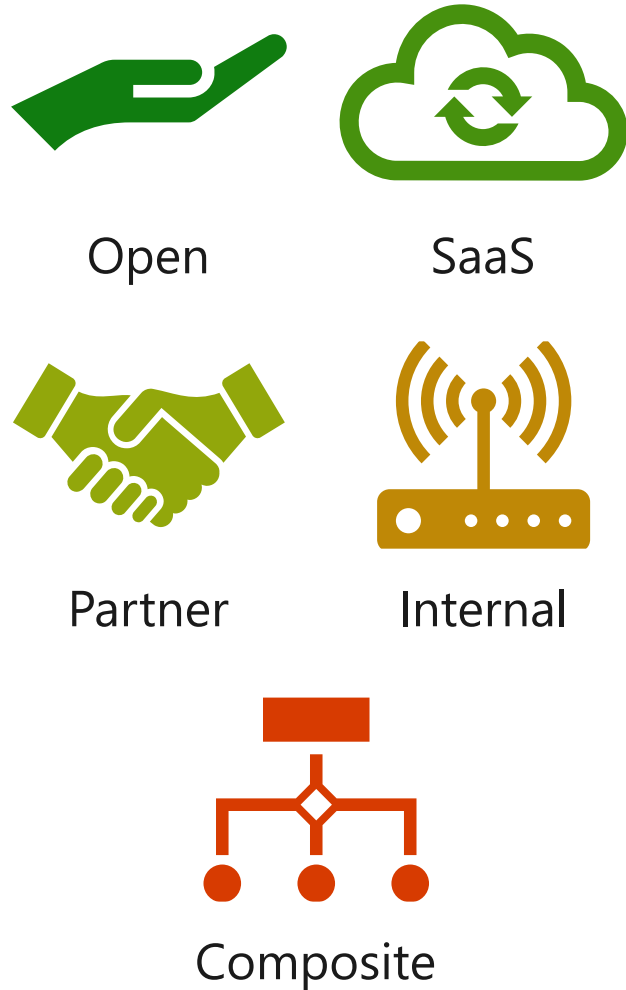


# What is an Application Programming Interface?

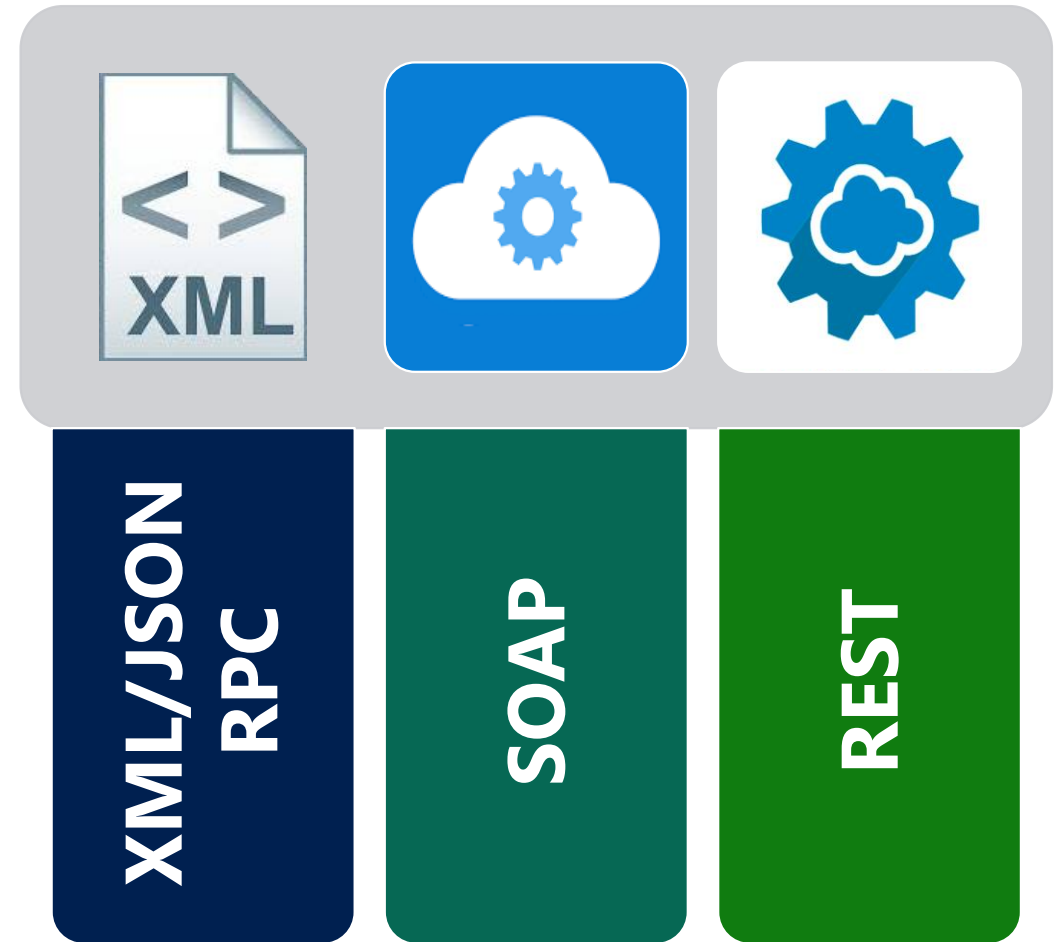


*"A set of procedures or protocols that allow the data and functionality of an application or service to be accessed and integrated into another application or service."*

# Different Types of APIs



# Web Services Flavours





# Functional APIs

REST defines  
URLs as  
Resources

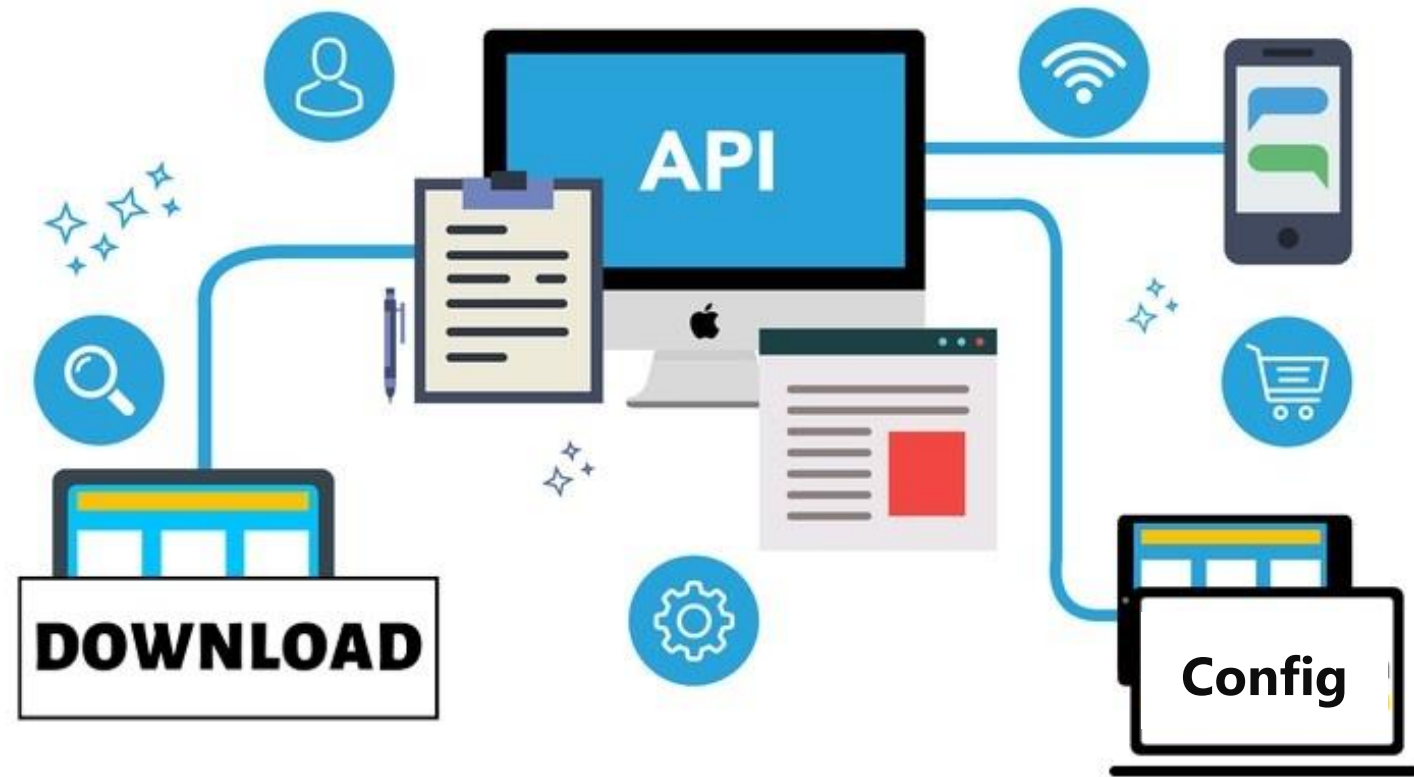
What to do  
when can't be  
a resource?

Functional APIs

Operational  
requirements

No returned  
data

Avoid RPC



# Simple Object Access Protocol API

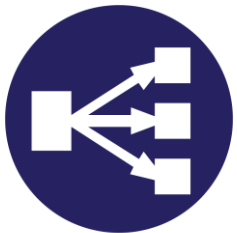
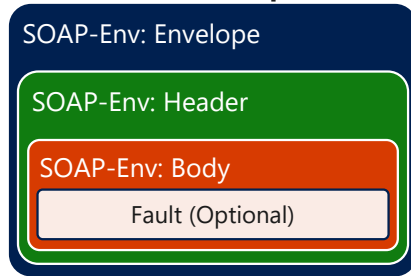


## Messaging Protocol

*Transport over HTTP, SMTP, MQ*  
*Extensibility, neutrality & independence*

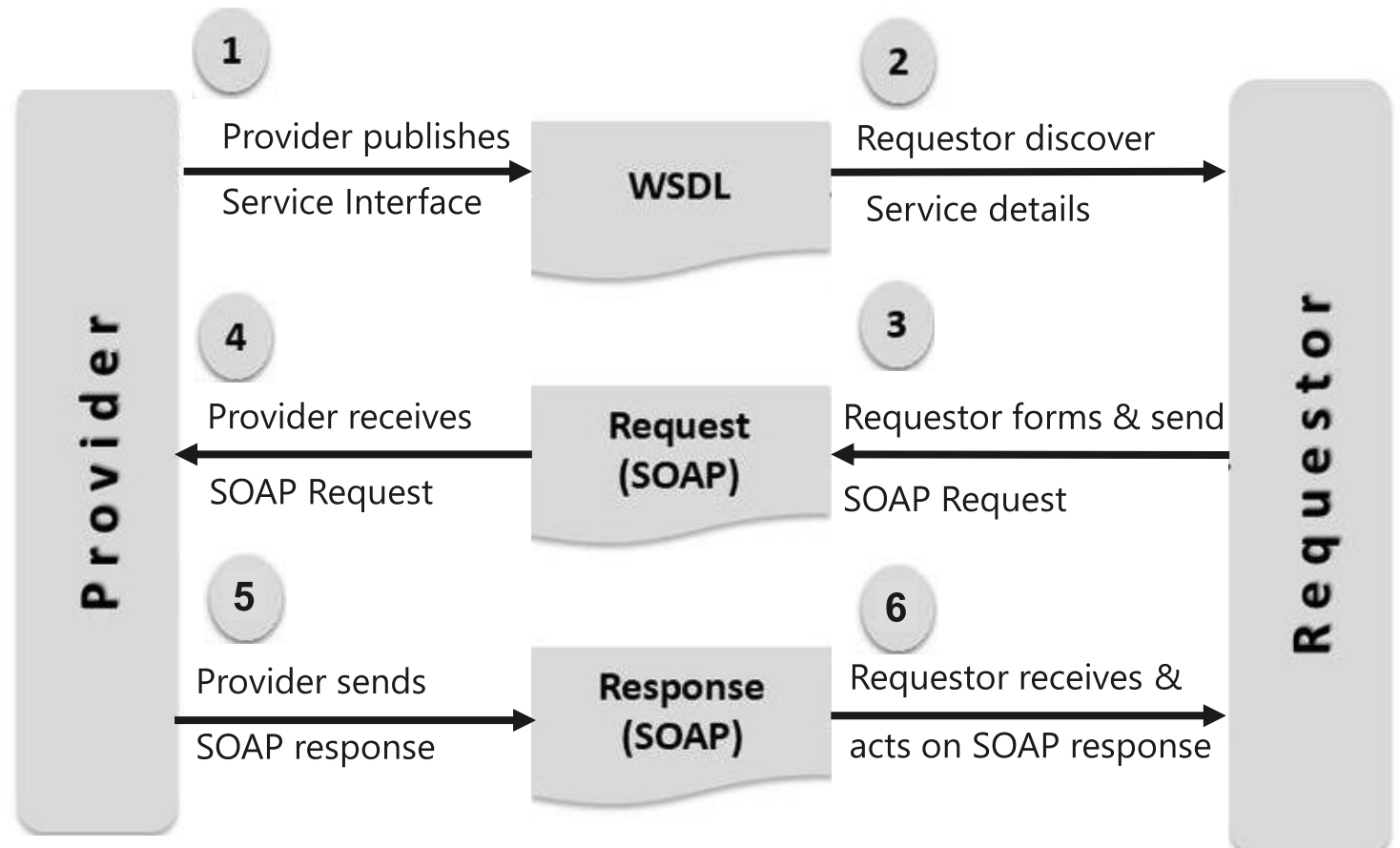


## XML based protocol



## WDSL

*Web Service Interface Discovery*  
*Supports only POST as v1..0*  
*Supports all HTTP verbs as v2.0*

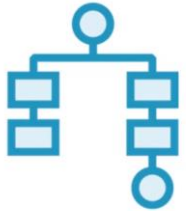


# REpresentational State Transfer API



## Architectural principles

*Separation of Client and Server*  
*Server Requests are Stateless*  
*Cacheable Requests*  
*Uniform Interface*



## What are Resources?

*Noun-based delimited set of data*



## Basic Http Verbs

*Retrieve, add, update or delete resources*

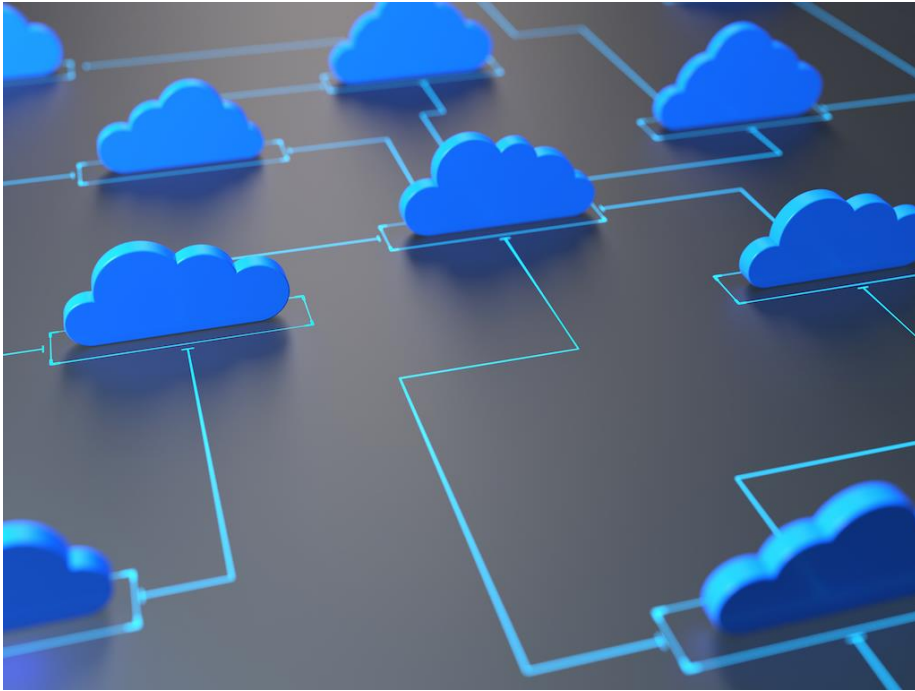


## What is an URIs?

*<https://api.yourserver.com/invoices>*



# Versioning your API



## What happens once your API is published?

- Consumers rely on it
- Requirements will change

## Many API versioning schemes

- Product versioning is not API versioning!
- Find one that works and stick to it!
- Your serving clients not yourselves

## URI Versioning

### URI Path

*/api/v2/Customers*

### Query String

*/api/Customers?v=2.0*

## Headers

### Attribute

*GET /api/Customers HTTP/1.1*

*HTTP/1.1 Host: localhost:44388*

*Content-Type: application/json*

*X-Version: 2.0*

### Accept

*...*

*Content-Type: application/json*

*Accept: application/json;version=2.0*

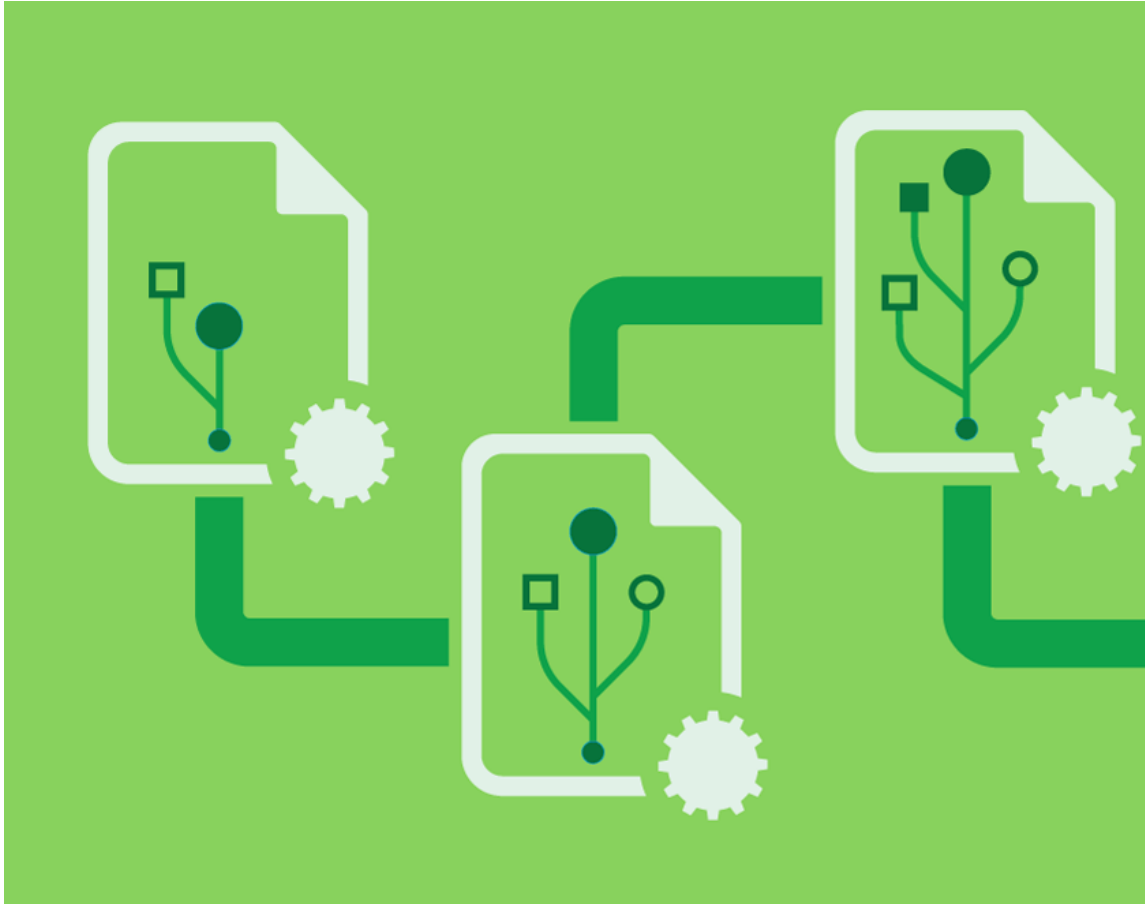
### Application specific Content-Type

*...*

*Content-Type: application/vnd.[yourapp].v1+json*

*Accept: application/vnd.[yourapp].v1+json*

# Versioning your ASP.NET Web API



ASP.NET & ASP.NET Core NuGet Package



Supports URI & Header versioning schemes



Attributes based controllers & actions



Supports namespaces segmentation



Can combine multiple versioning scheme



Supports version scheme conventions

<https://github.com/Microsoft/aspnet-api-versioning/wiki/How-to-Version-Your-Service>



Comfort Break  
~ 5-10 minutes

# Why/When To Use an

A silver laptop is shown from a front-facing perspective, slightly angled. The screen is white and displays the text "{ API }" in a large, bold, black sans-serif font. The background of the entire image is a solid orange color.

**{ API }**

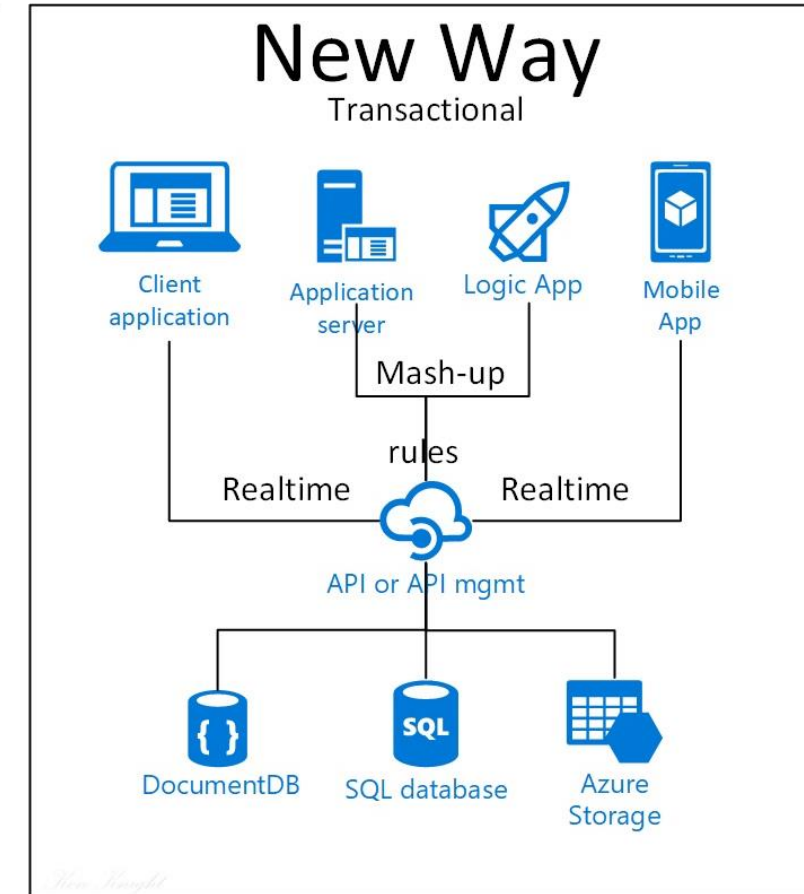
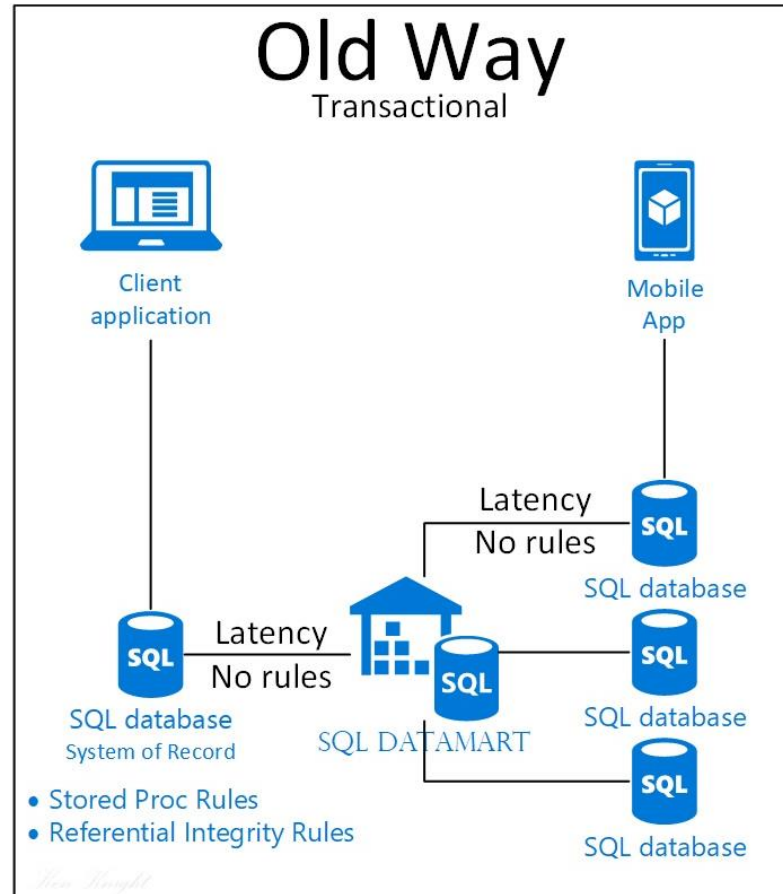
# Why/When to Use APIs?

Simplify & Accelerate Development

Centralising domain and services

Control & Protection to Resources

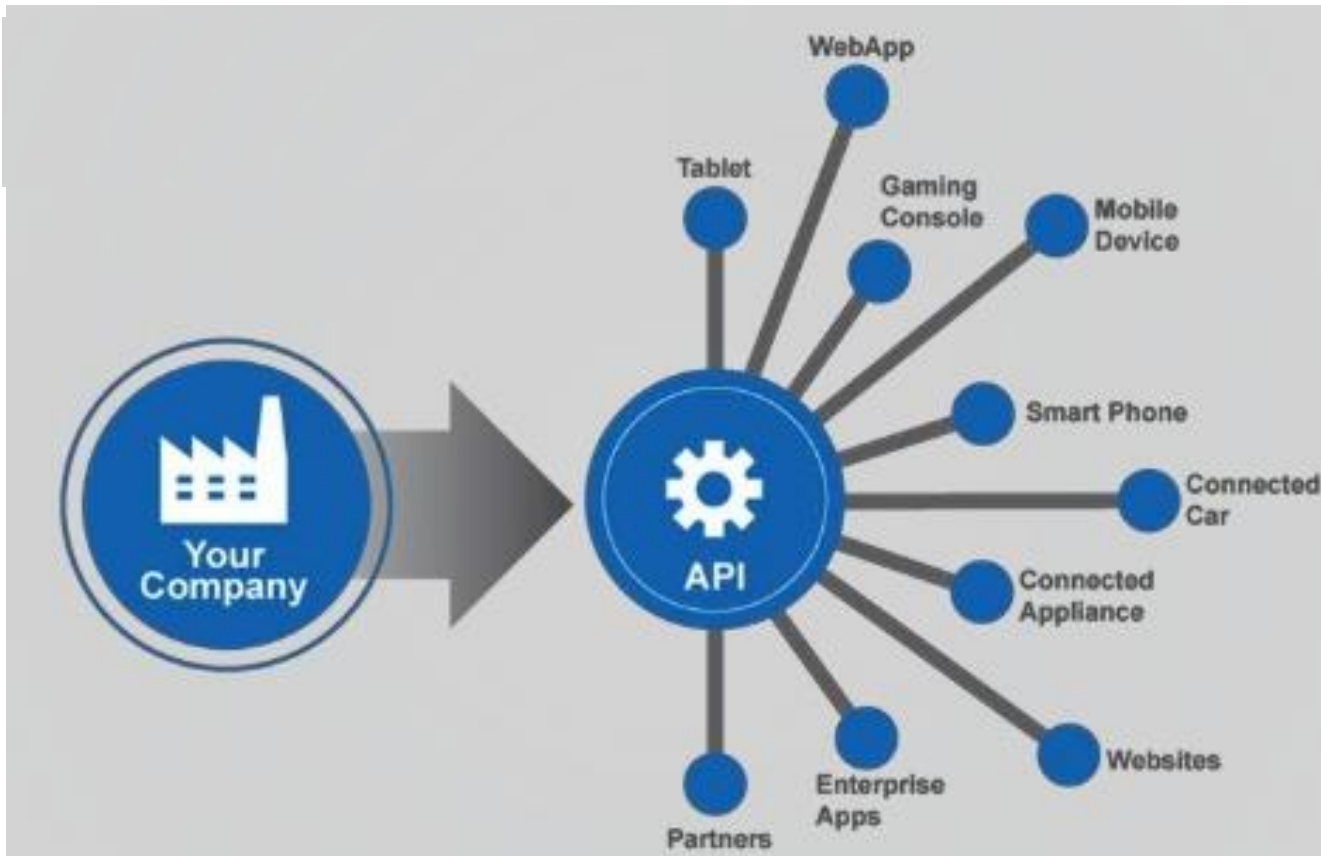
Service to Service communication



# The rise of the API Economy

“The API Economy is an enabler for turning a business or organization into a platform.”

- Kristin R. Moyer, vice president & distinguished analyst at Gartner.



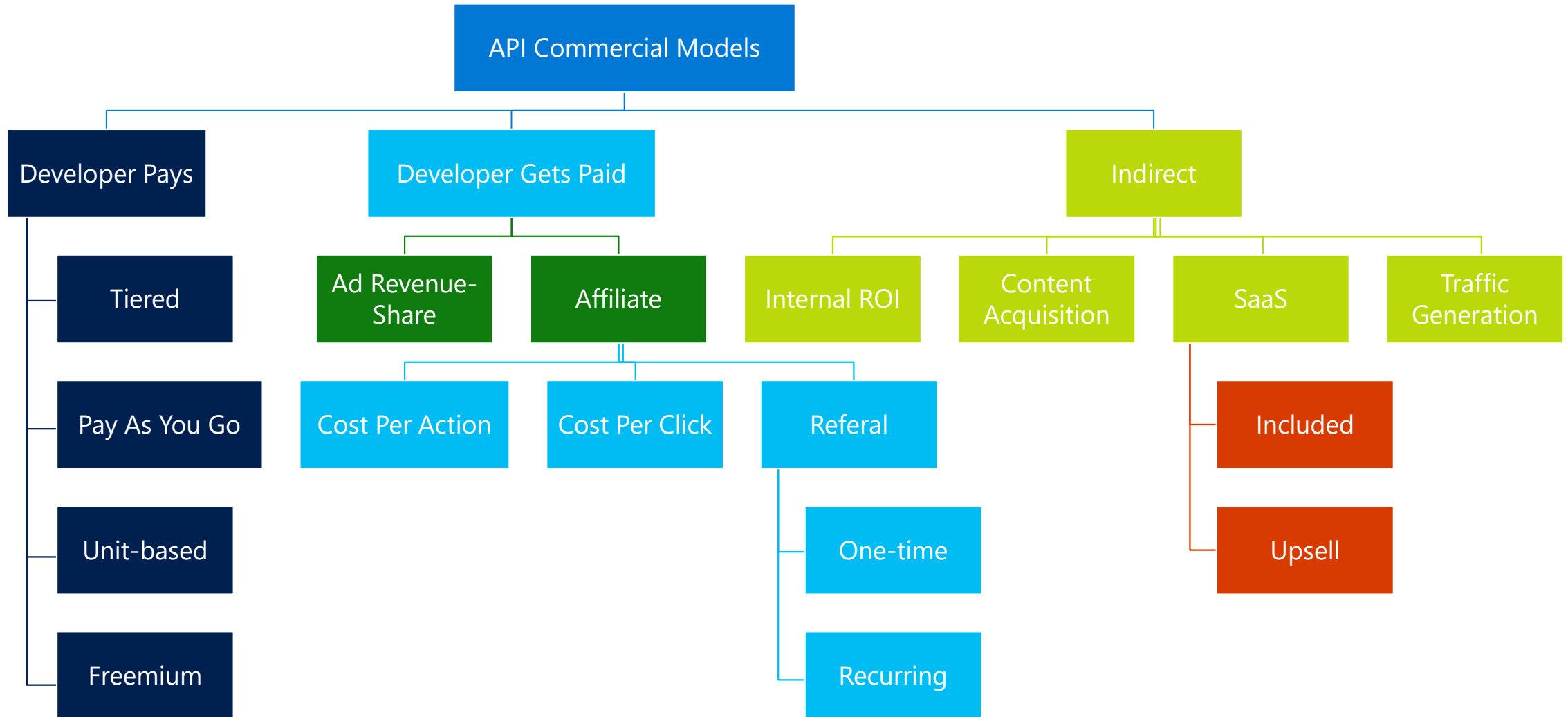
Technology trends

New business strategies

New regulations

Technology standards

# Commercial Models In the API Economy





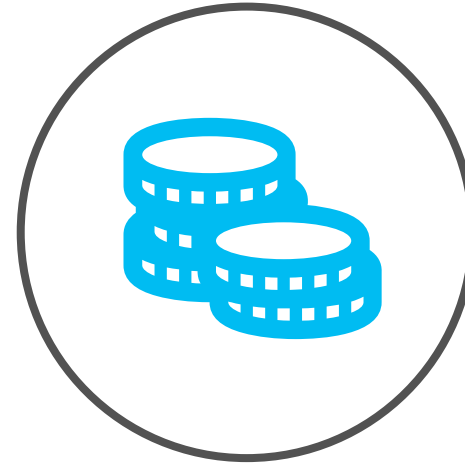
# Benefits of APIs



Expand  
market reach



Foster  
innovation



Lower  
TCO



Highly scalable  
business models

# Risks of Exposing APIs



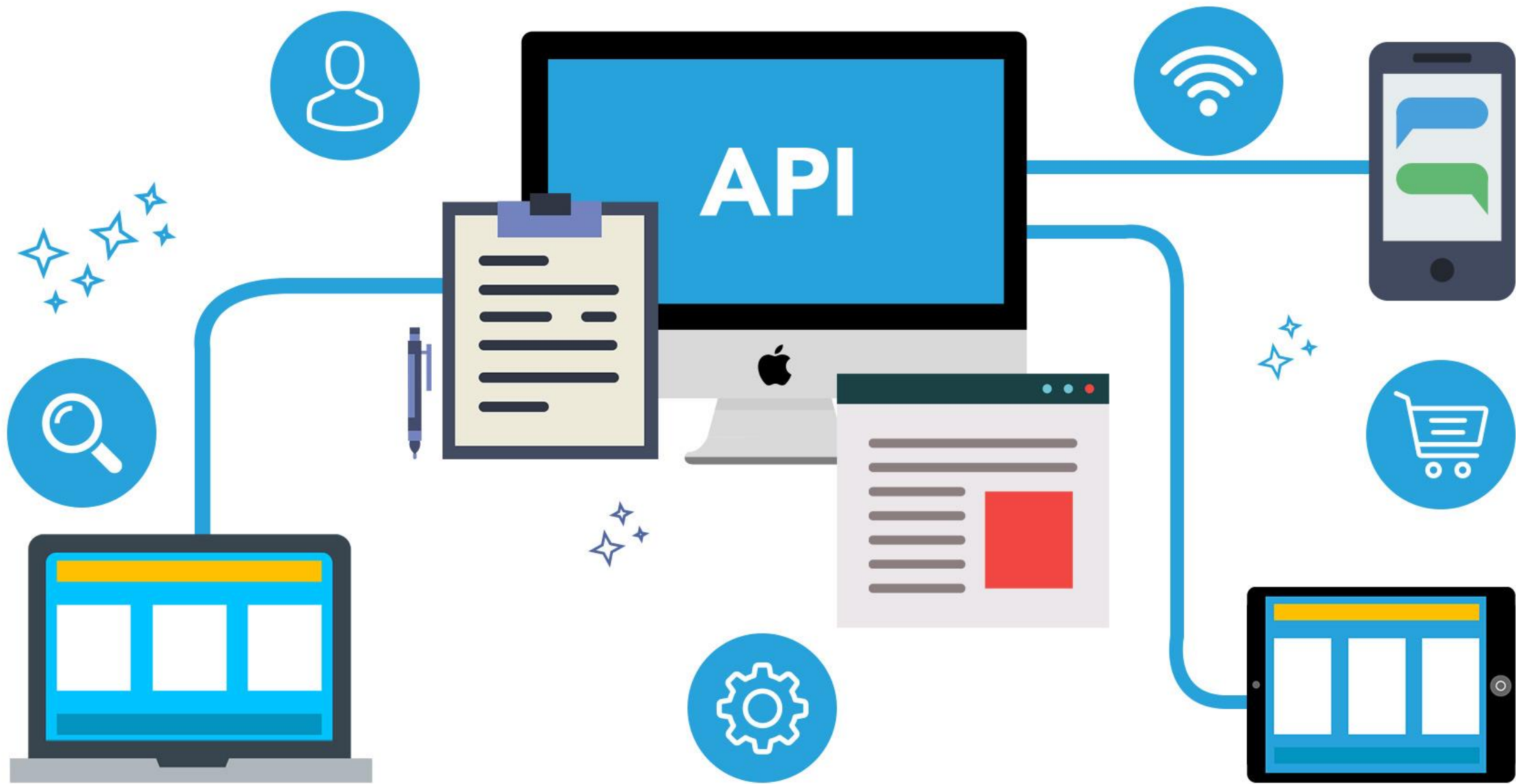
Cyber attacks

Over exposure of assets

Reputation damage

Core business

Comfort Break  
~ 5-10 minutes



# API Testing

# The Importance of API Testing

User Input should not be trusted

Output must be verified

Implementation details should not be exposed




Authentication & Authorization

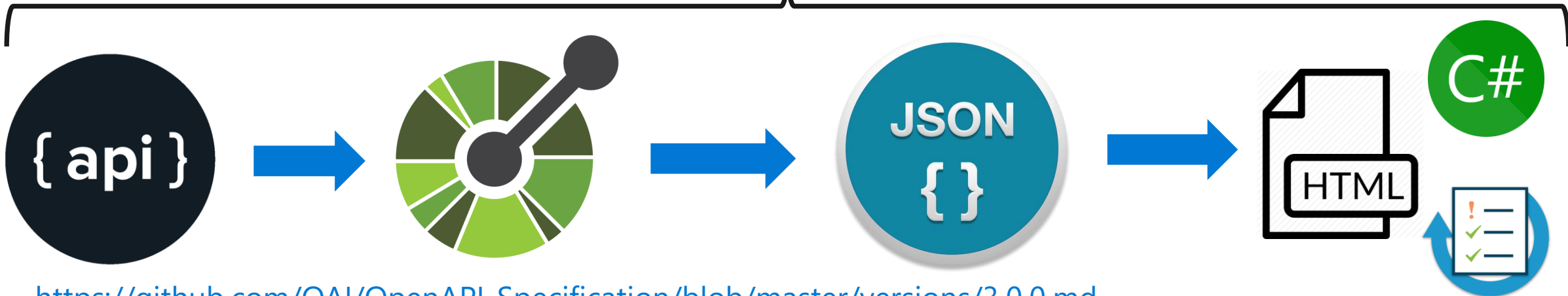




# Why Documenting your API?



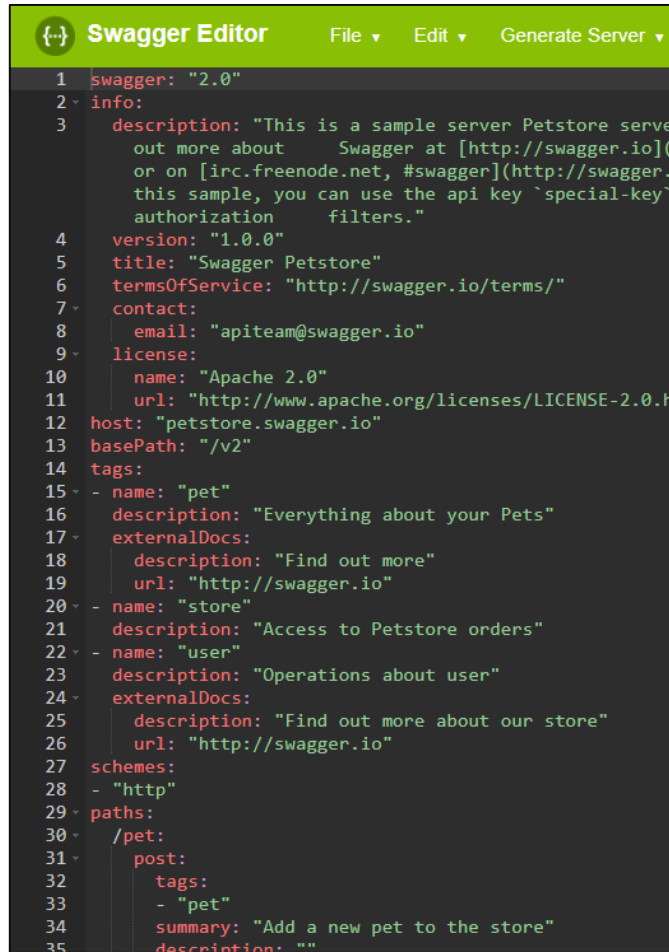
-  Not only public API needs to be documented
-  Great documentation fosters to adoption
-  Saves Time and Money



<https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.0.md>

# Swagger Toolset

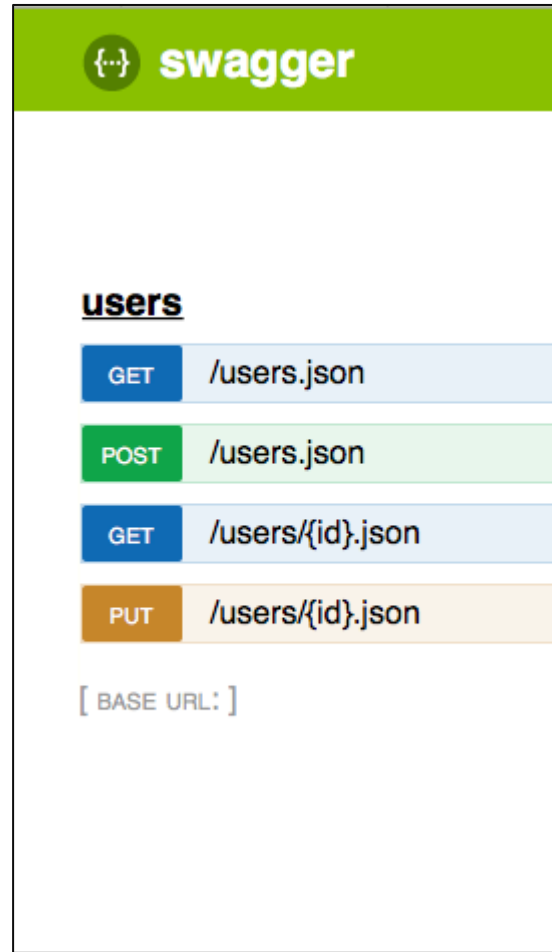
## Swagger Editor



```
1 swagger: "2.0"
2 info:
3   description: "This is a sample server Petstore server.
4     out more about Swagger at [http://swagger.io]()
5     or on [irc.freenode.net, #swagger](http://swagger.io/irc).
6     this sample, you can use the api key `special-key`
7     authorization filters."
8   version: "1.0.0"
9   title: "Swagger Petstore"
10  termsOfService: "http://swagger.io/terms/"
11  contact:
12    email: "apiteam@swagger.io"
13  license:
14    name: "Apache 2.0"
15    url: "http://www.apache.org/licenses/LICENSE-2.0.html"
16 host: "petstore.swagger.io"
17 basePath: "/v2"
18 tags:
19 - name: "pet"
20   description: "Everything about your Pets"
21   externalDocs:
22     description: "Find out more"
23     url: "http://swagger.io"
24 - name: "store"
25   description: "Access to Petstore orders"
26 - name: "user"
27   description: "Operations about user"
28   externalDocs:
29     description: "Find out more about our store"
30     url: "http://swagger.io"
31 schemes:
32 - "http"
33 paths:
34 /pet:
35   post:
36     tags:
37     - "pet"
38     summary: "Add a new pet to the store"
39     description: ""
```

<https://github.com/swagger-api>

## Swagger UI



## Swagger CodeGen

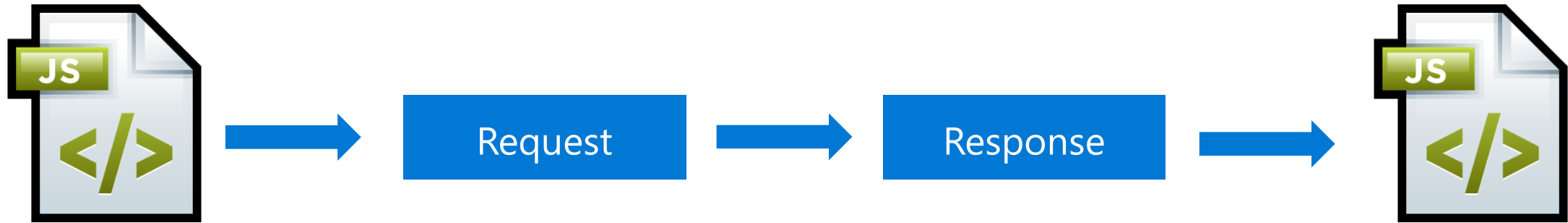


<https://github.com/domaindrivendev/Swashbuckle>

## SwashBuckle



# Generate Automated Test with Postman & Newman



```
var responseTimeTest = () =>
pm.test("Response time must be below 300ms", function () {
  pm.expect(pm.response.responseTime).to.be.below(300);
});
pm.globals.set("responseTimeTest",
  responseTimeTest.toString());
```

```
pm.test("Status code is 200", function () {
  pm.response.to.have.status(200);
});

eval(pm.globals.get("responseTimeTest"))();
```

# Demo







## Discussion

Have your understanding of APIs shifted?  
What did you learn in the course?  
What are you going to further explore?

Discussion | Your feedback and next actions...

API Back to Basics | Wrap Up



**Before we wrap up...**

Please take a moment to give us your feedback...



<https://aka.ms/APIAutomationTesting>

Q&A

You have

Questions

We have

Answers

# Thank You

ευχαριστώ    Salamat Po    متشكراً    شكراً    Grazie  
благодаря    ありがとうございます    Kiitos    Teşekkürler    谢谢  
ໂປບຄຸນຄວັ້ນ    Obrigado    شكریه    Terima Kasih    Dziękuję  
Hvala    Köszönöm    Tak    Dank u wel    ДЯКУЮ    Tack  
Mulțumesc    спасибо    Danke    Cám ơn    Gracias  
多謝晒    Ďakujem    תודה    நன்றி    Děkuji    감사합니다

