

Building APIs with Microservices

Things I wish I'd known

James Gough

@Jim__Gough

O'REILLY®

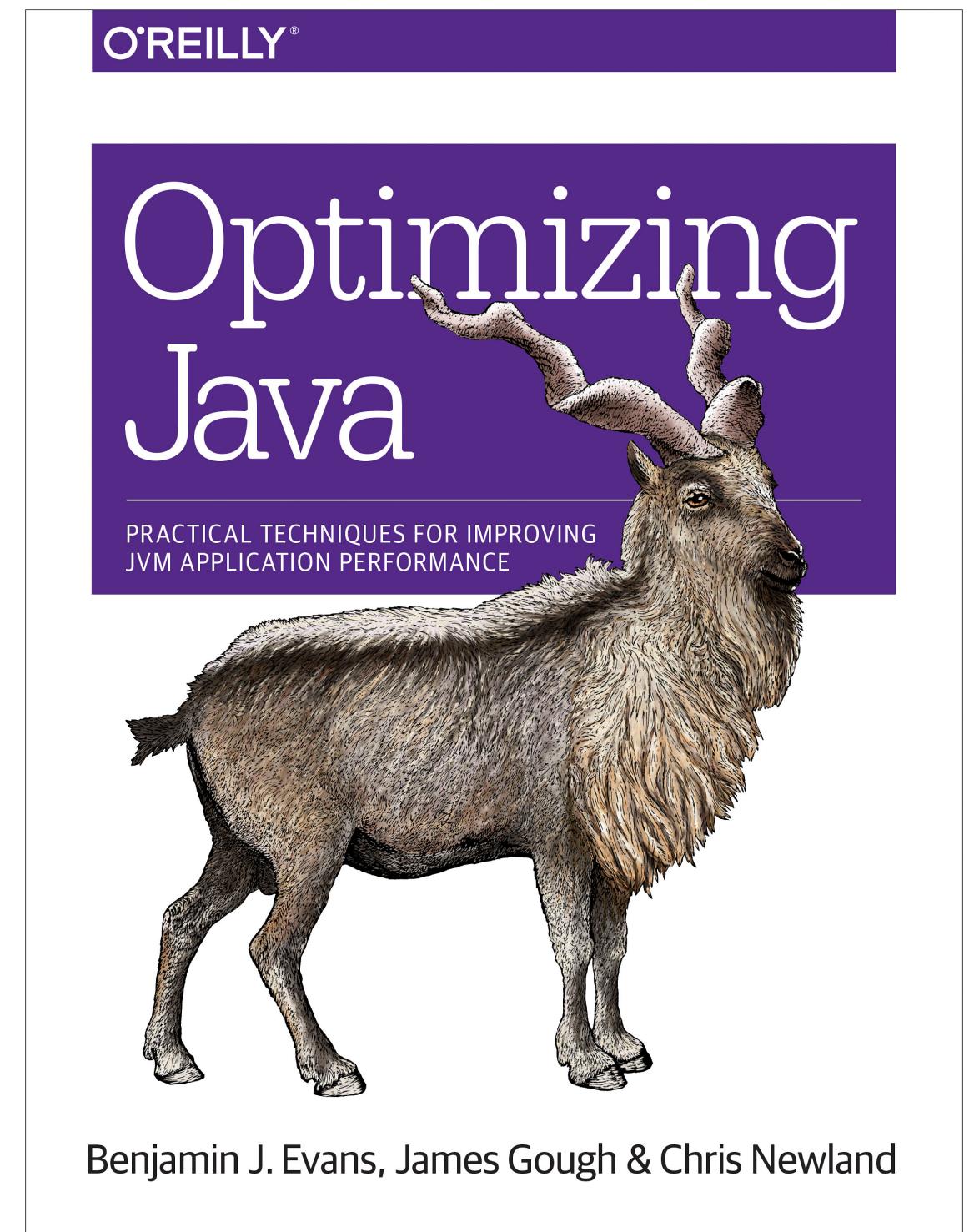
Software
Architecture

Agenda

- What is an API?
- Problems faced when building APIs
- Designing and testing restful APIs
- API versioning concerns
- Developing APIs in a distributed team
- Microservices and Java
- Using Docker
- Scheduling containers
- Kubernetes
- Introducing Istio
- API Management

James (Jim) Gough

- Previous: Java developer and contributor
- Previous: Technical Trainer at Morgan Stanley and Bloomberg
- Current: Java Developer/Architect and API Technical Lead at Morgan Stanley
- Co-Author of Optimizing Java



What is an API?

“API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other. Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you’re using an API”

MuleSoft

What is an API?

“An application-programming interface (API) is a set of programming instructions and standards for accessing a Web-based software application or Web tool.”

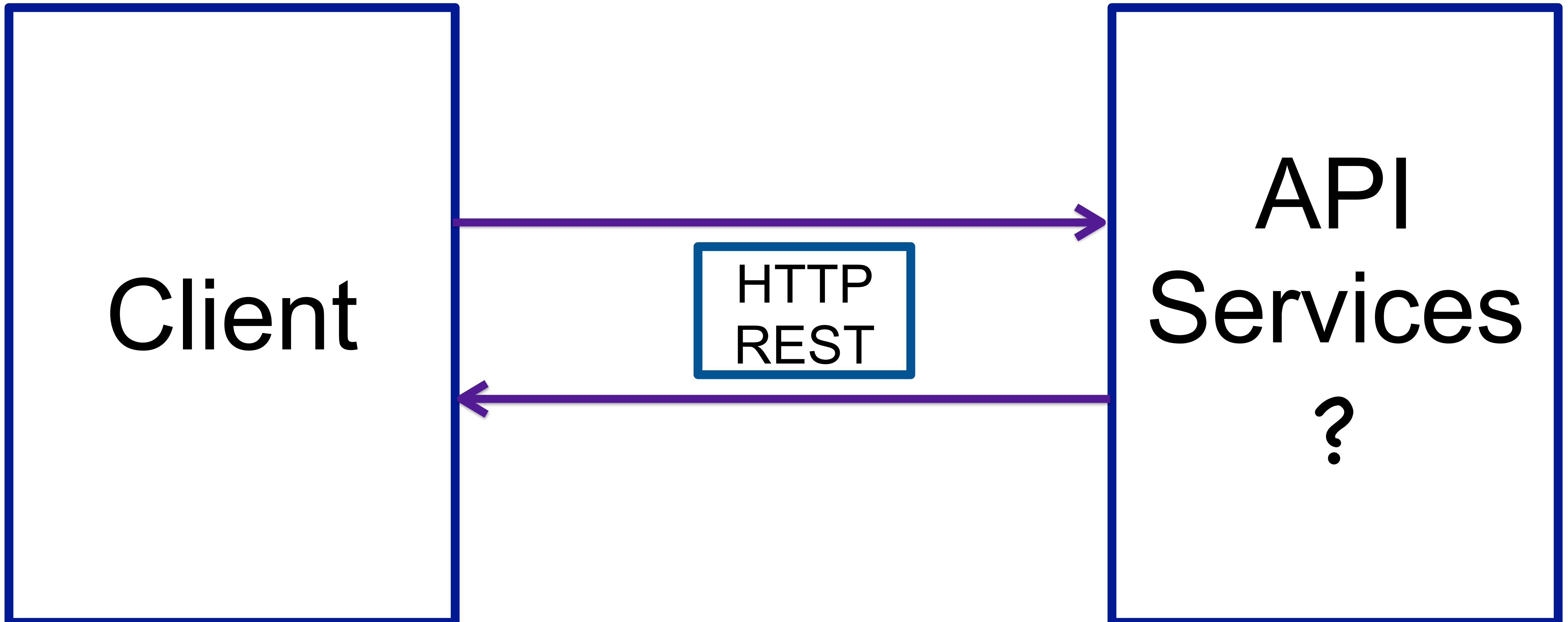
How Stuff Works

What is an API?

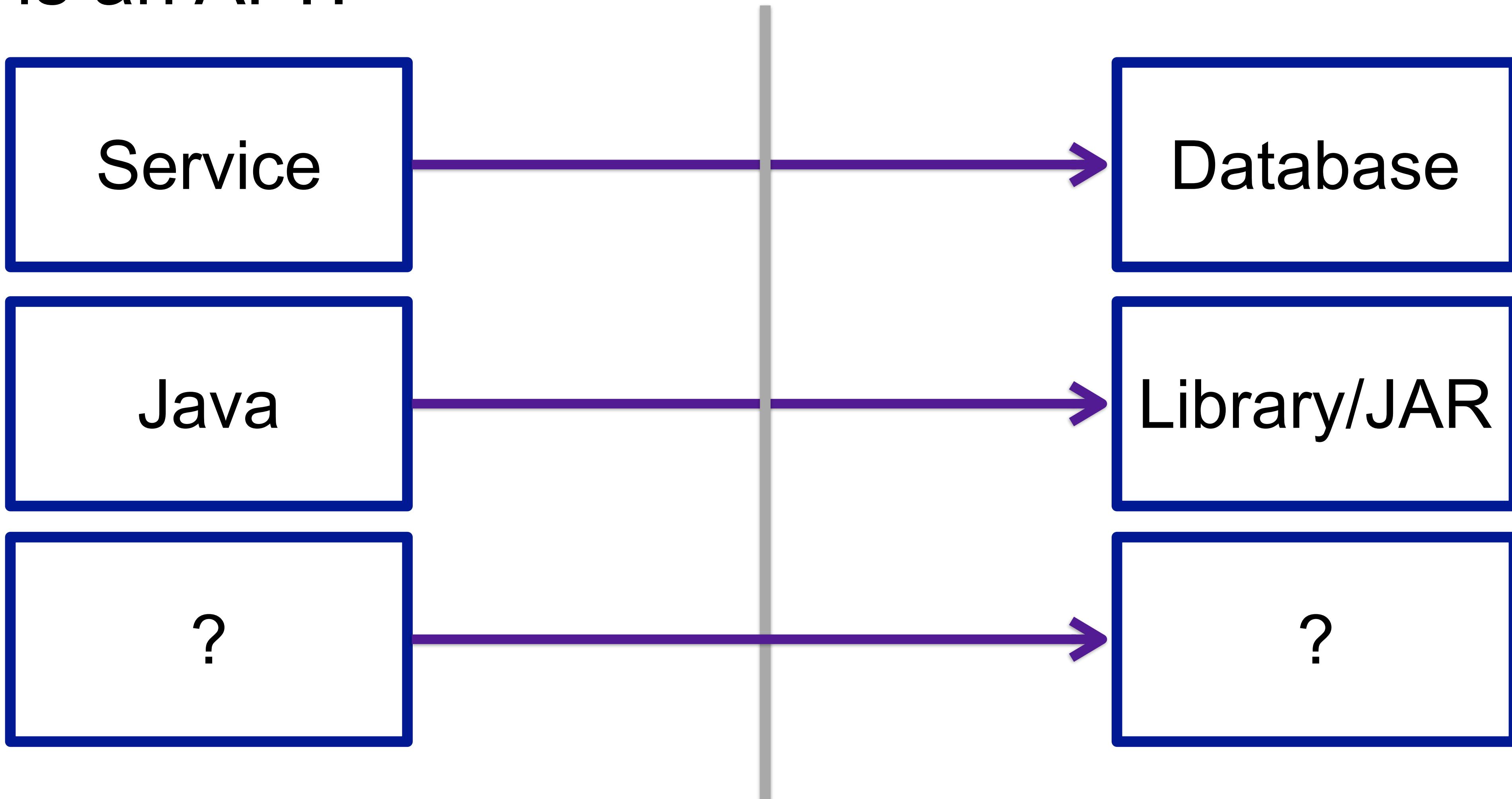
“Application programming interfaces hide complexity from developers, extend systems to partners, organise code, and make components reusable”

InfoWorld

What is an API?



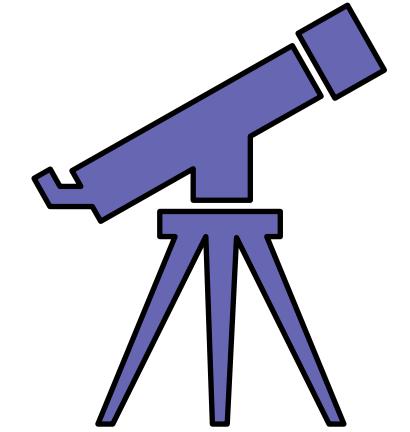
What is an API?



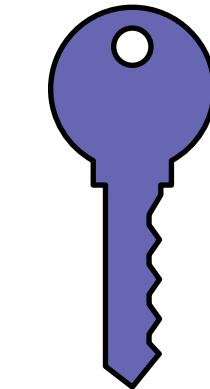
Problems Faced with all APIs

v 1.0
v 1.1

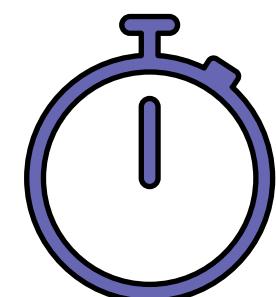
Versioning



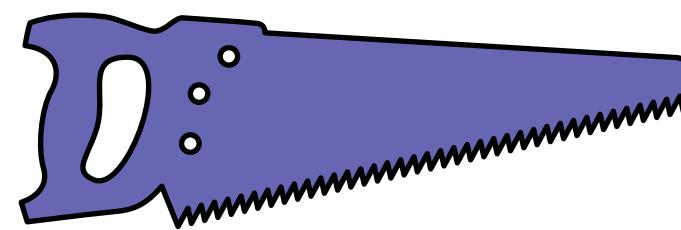
Discovery



Security



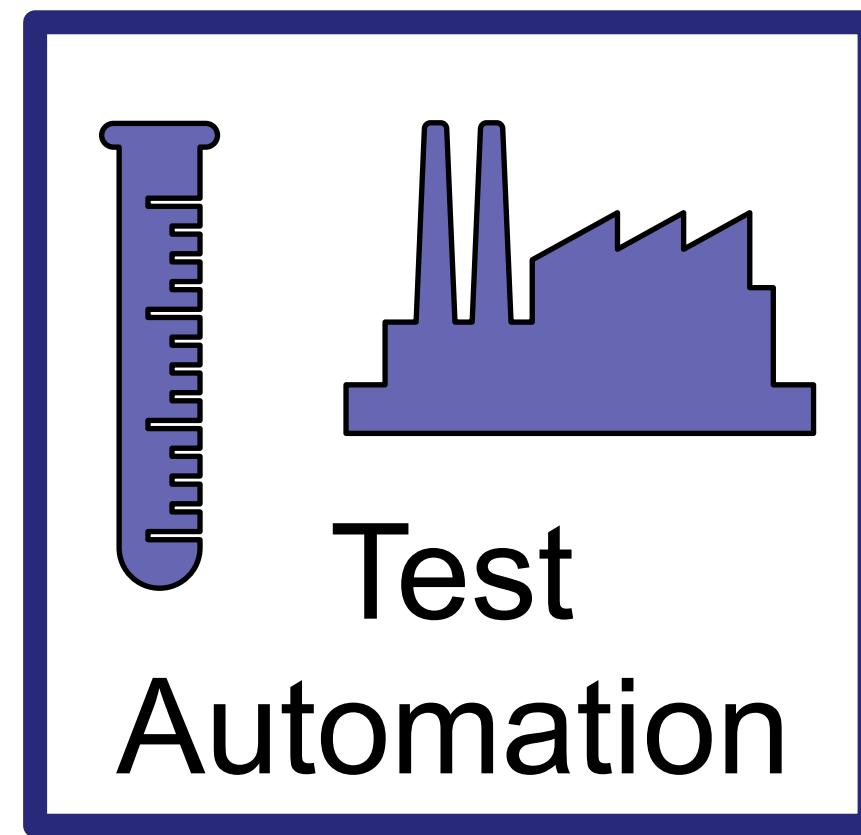
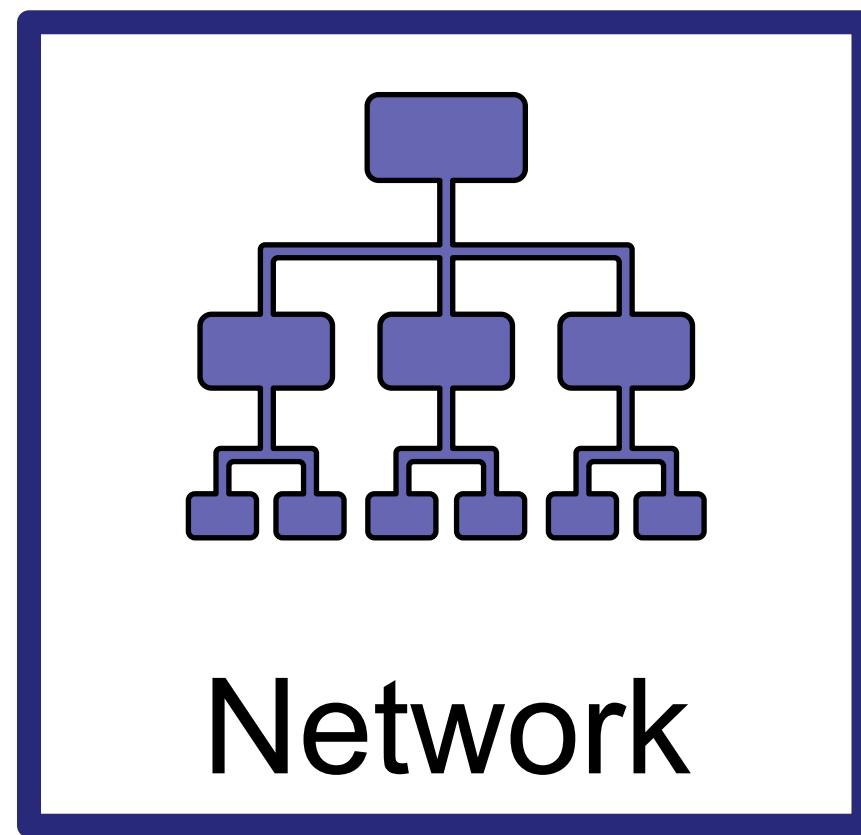
Performance



Quality

<https://smartbear.com/SmartBear/media/ebooks/State-of-API-Report-2016.pdf>

Decoupled API Challenges



Designing Restful APIs

HATEOAS

Well Defined
Purposed
Endpoints

Versioning

Agree
Standards

Use Beta
Phase

Developer
Friendly Error
Structure

<https://github.com/paypal/api-standards>

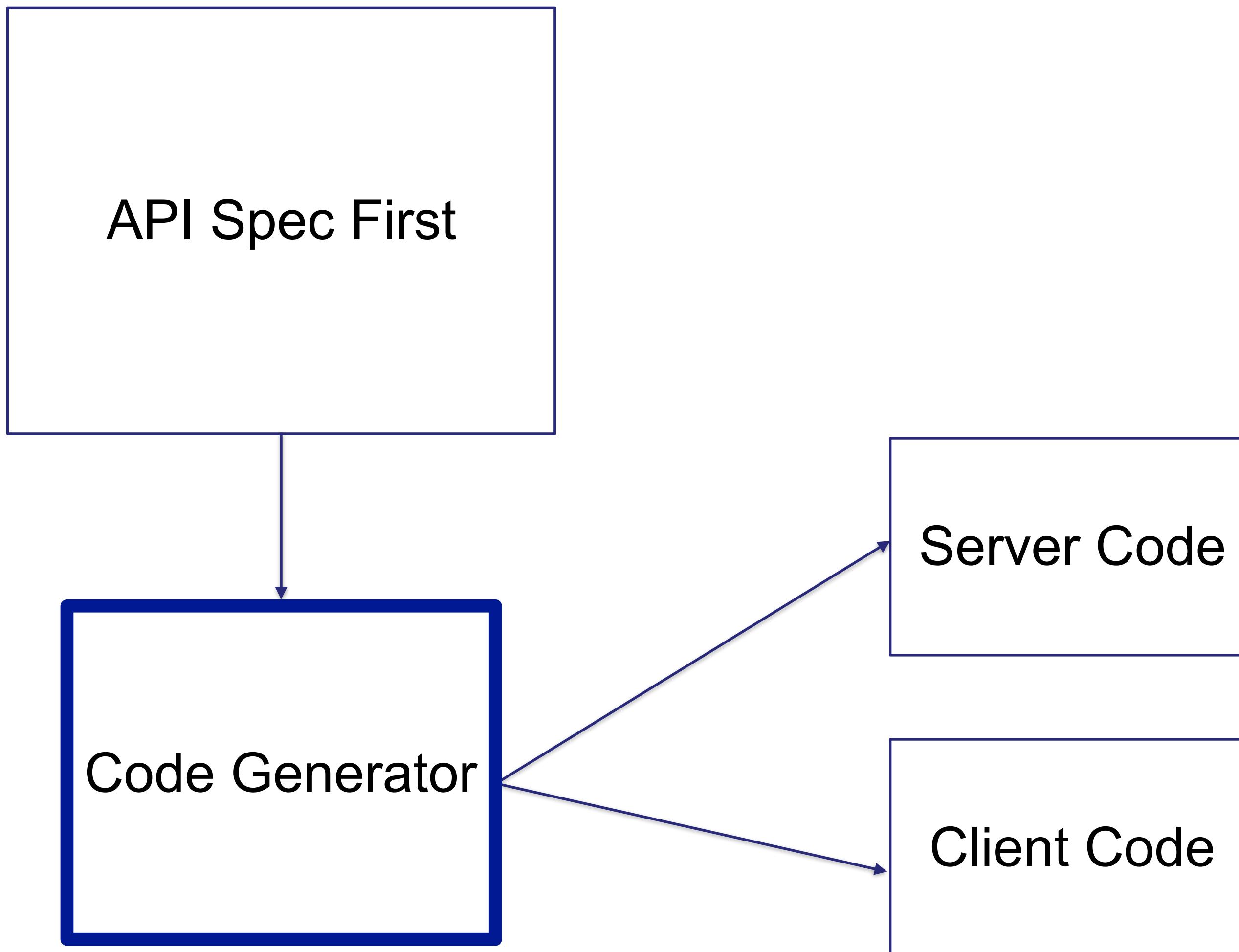
Contract Driven Development

```
org.springframework.cloud.contract.spec.Contract.make {  
    request {  
        method 'POST'  
        body([  
            "date": "2018-09-29"  
        ])  
        url '/day'  
        headers {  
            header('Content-Type', 'application/json')  
        }  
    }  
    response {  
        status 200  
        body([  
            "dayOfWeek": "SATURDAY"  
        ])  
    }  
}
```

Tests

Stub
Server

API Specification First



The screenshot shows the Swagger Editor interface for the Petstore API. On the left, the Swagger JSON code is displayed:

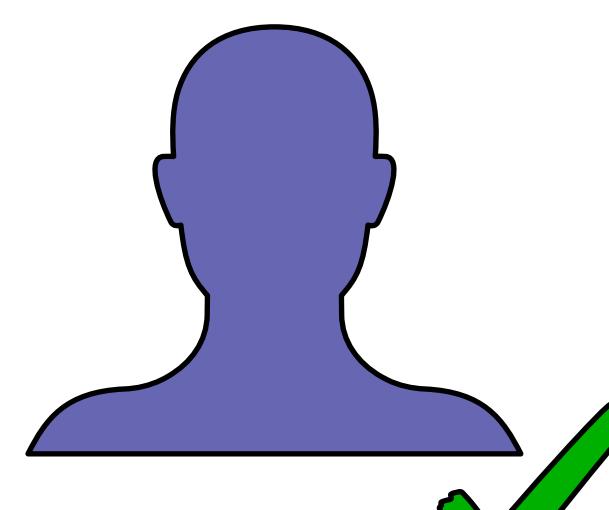
```
swagger: "2.0"
info:
  description: "This is a sample server Petstore server. You can find out more about Swagger at [http://swagger.io](http://swagger.io) or on [irc.freenode.net, #swagger](http://swagger.io/irc/). For this sample, you can use the api key `special-key` to test the authorization filters."
  version: "1.0.0"
  title: "Swagger Petstore"
  termsOfService: "http://swagger.io/terms/"
  contact:
    email: "apiteam@swagger.io"
  license:
    name: "Apache 2.0"
    url: "http://www.apache.org/licenses/LICENSE-2.0.html"
  host: "petstore.swagger.io"
  basePath: "/v2"
  tags:
    - name: "pet"
      description: "Everything about your Pets"
      externalDocs:
        description: "Find out more"
        url: "http://swagger.io"
    - name: "store"
      description: "Access to Petstore orders"
    - name: "user"
      description: "Operations about user"
      externalDocs:
        description: "Find out more about our store"
        url: "http://swagger.io"
  schemes:
    - "https"
    - "http"
paths:
  /pet:
    post:
      tags:
```

On the right, the generated API interface is shown under the heading "Swagger Petstore 1.0.0". It includes sections for "Terms of service", "Contact the developer", and "Apache 2.0". Below this, there are three API endpoints listed:

- pet** Everything about your Pets
Find out more: <http://swagger.io>
 - POST** /pet Add a new pet to the store
 - PUT** /pet Update an existing pet
 - GET** /pet/findByStatus Finds Pets by status

<https://editor.swagger.io>

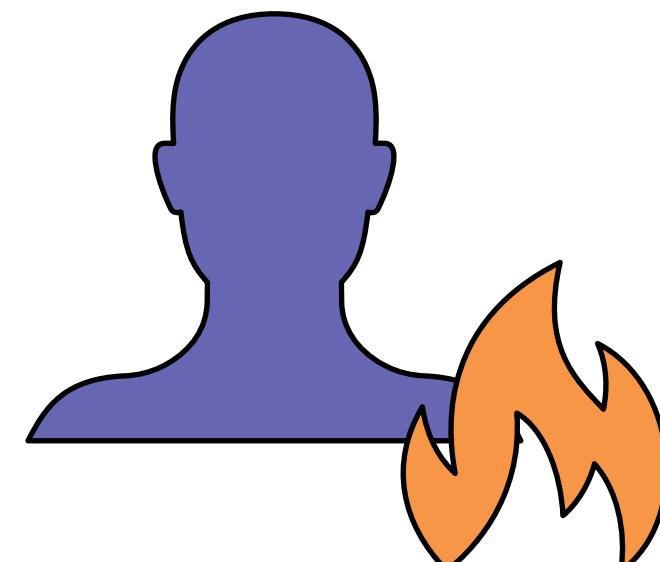
Versioning APIs



App

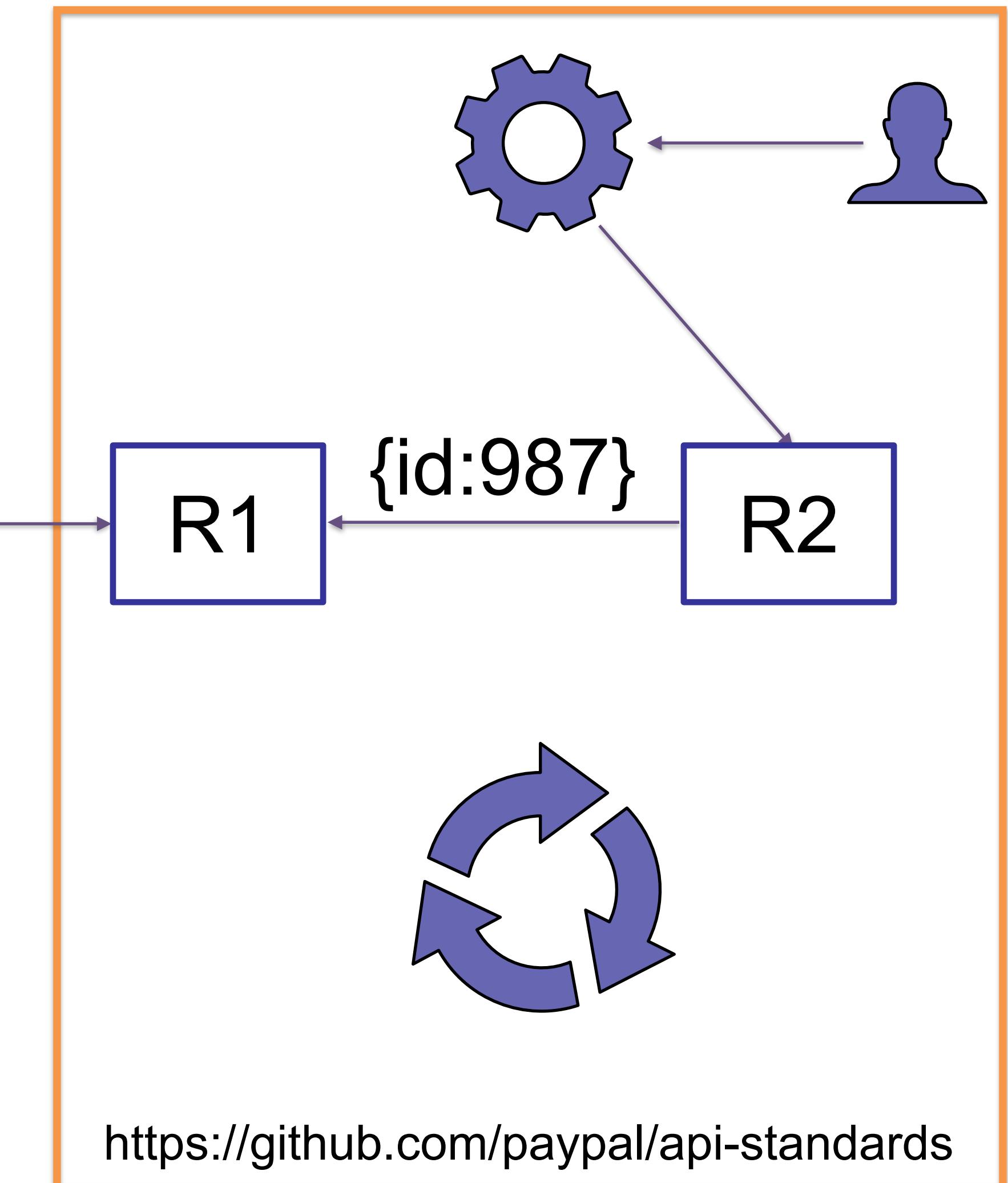
{loanid:987}
{id:987}

/loan

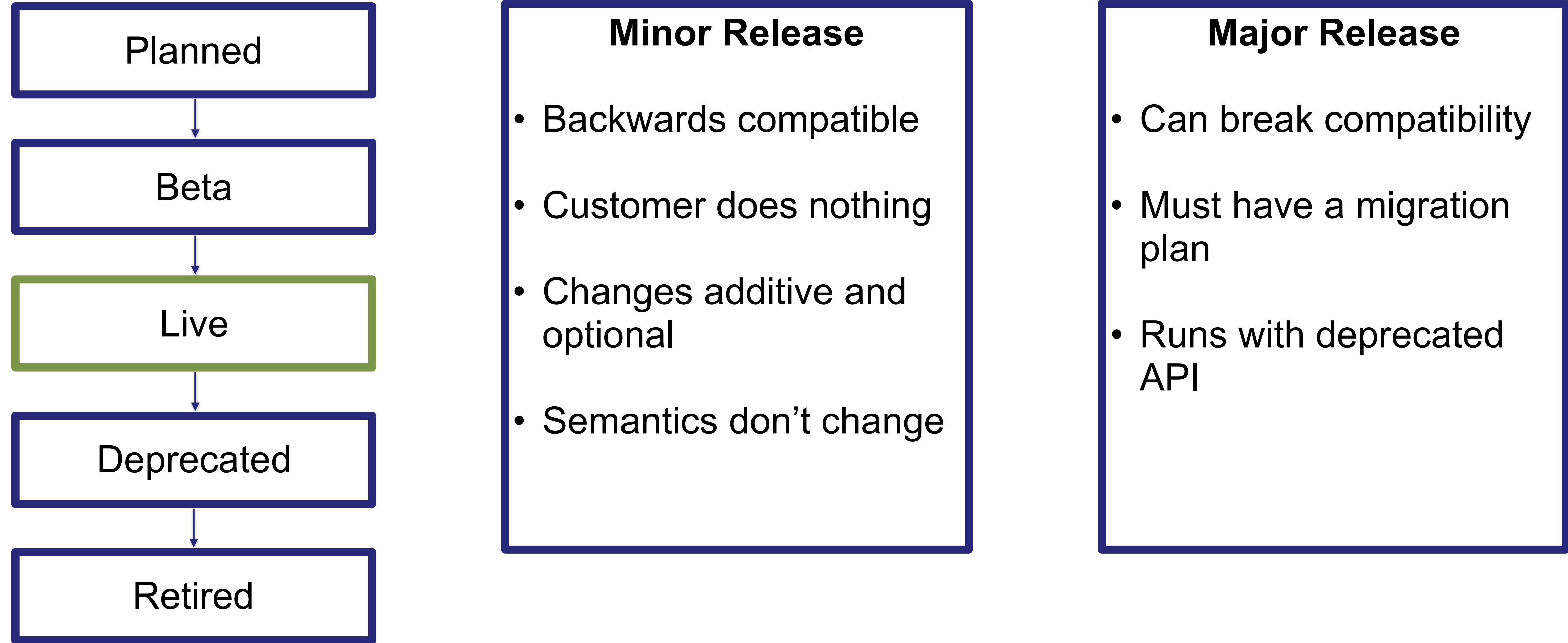


App

{id:621}
{loanid:621}



API Lifecycle



<https://github.com/paypal/api-standards>

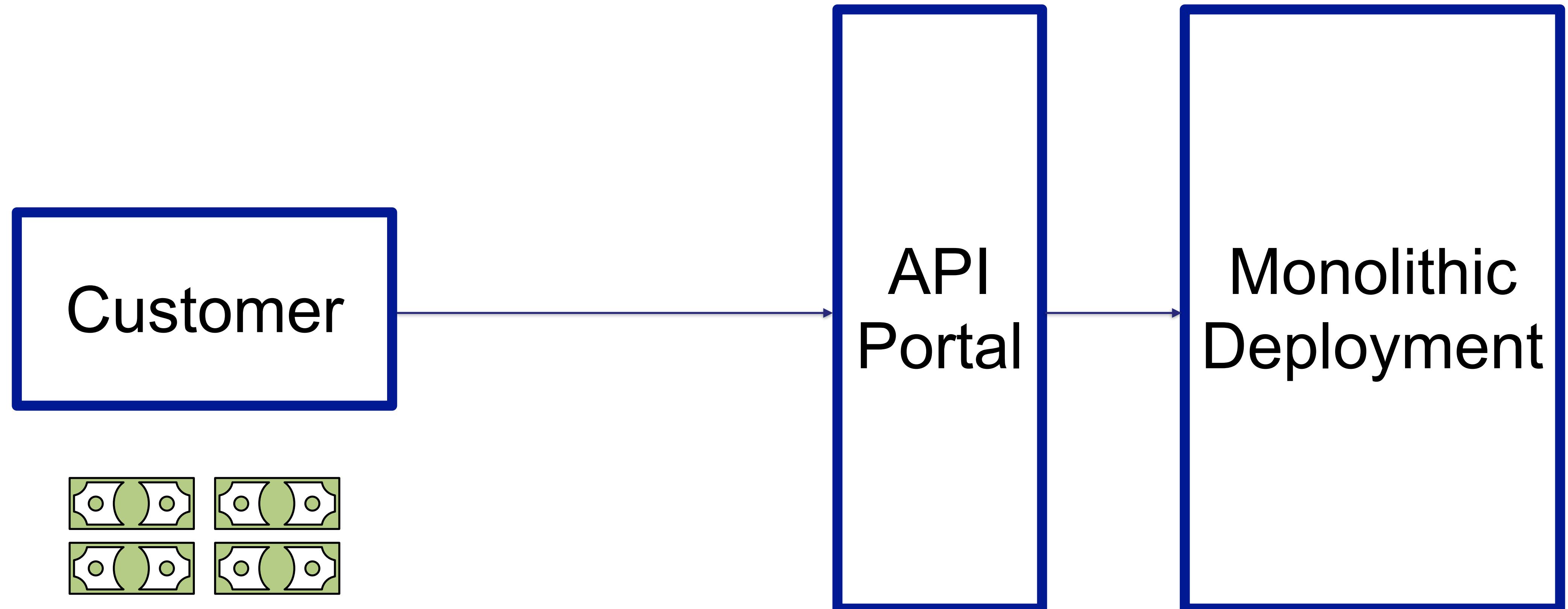
Developing APIs in a Distributed Team

Monolith
v2018.09.31

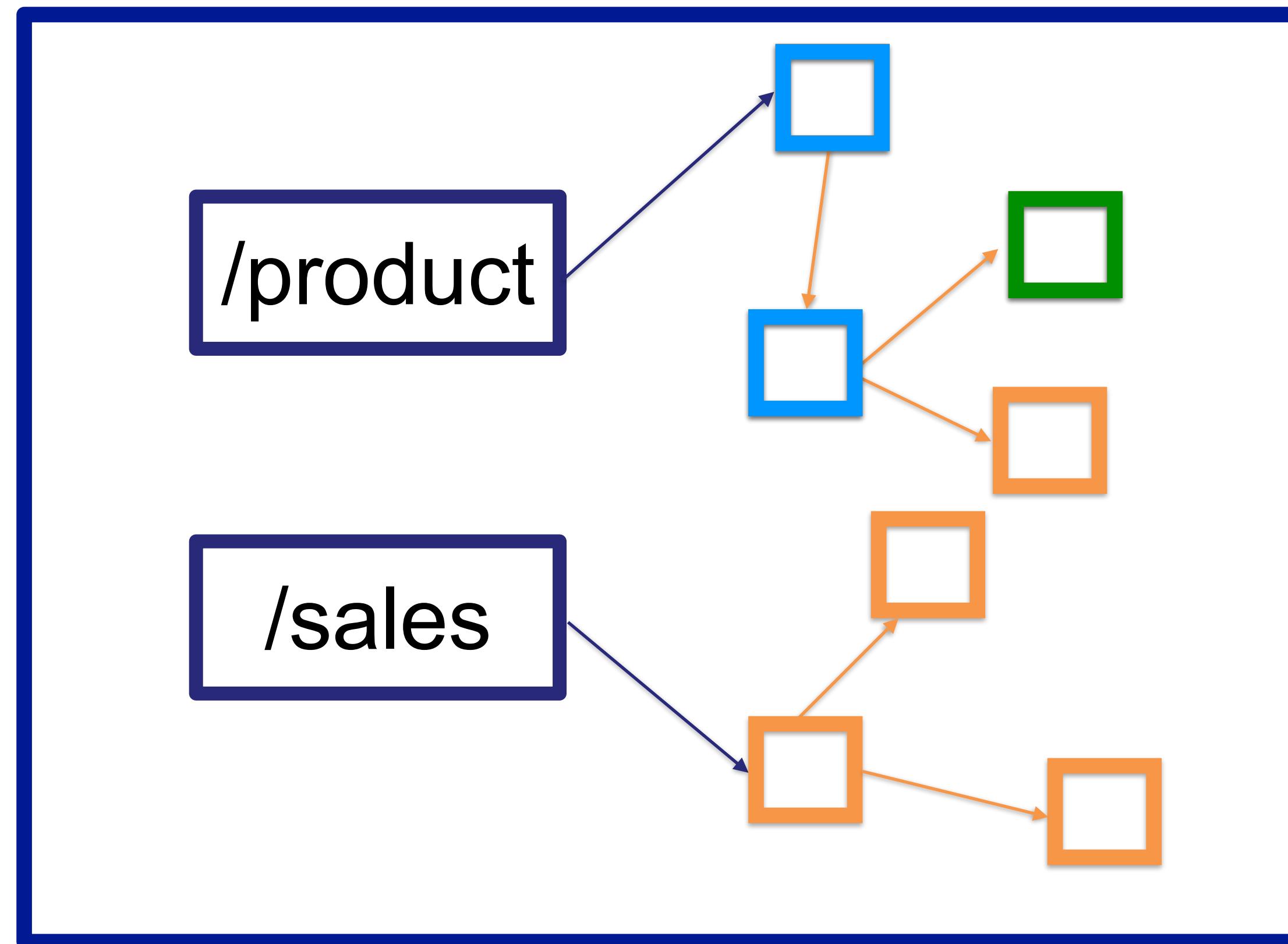
Monolith
v2018.10.31

- ✓ Binary compatibility
- ✓ Build time consistency
- ✓ Easy to reason about
- ✓ Simple to deploy
- ✗ Does not scale.. at all

Delivering a Simple API

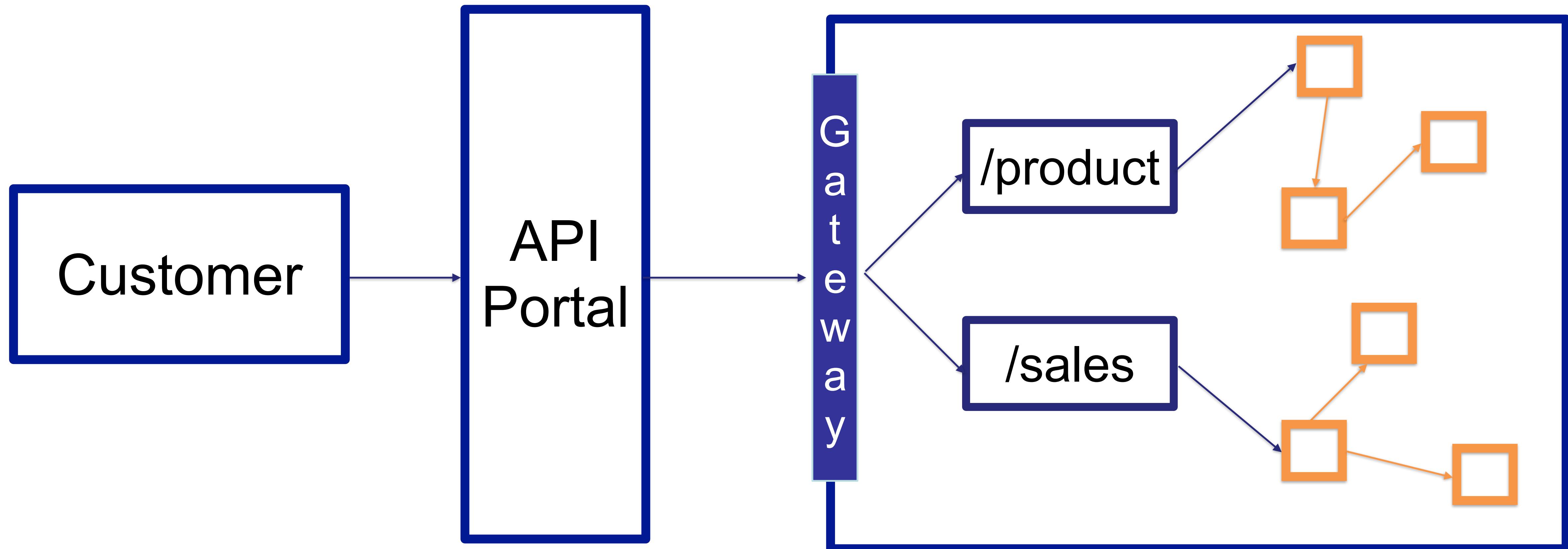


Developing APIs in a Distributed Team

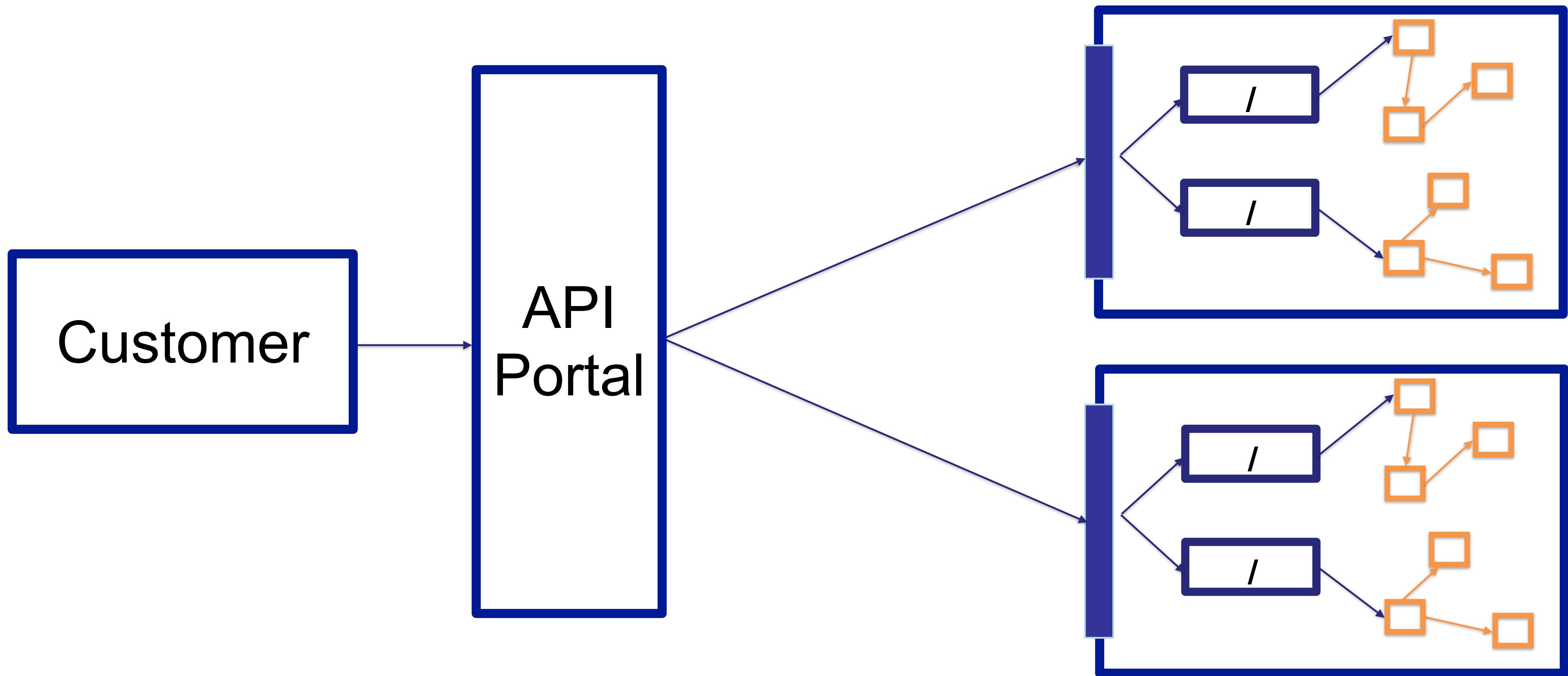


- All API deployments must move freely
- Full automated ephemeral testing available
- Replicated in a UAT environment for testing
- Expect many deployments per day
- Easy roll back
- Enhance test cases if issues found in Prod

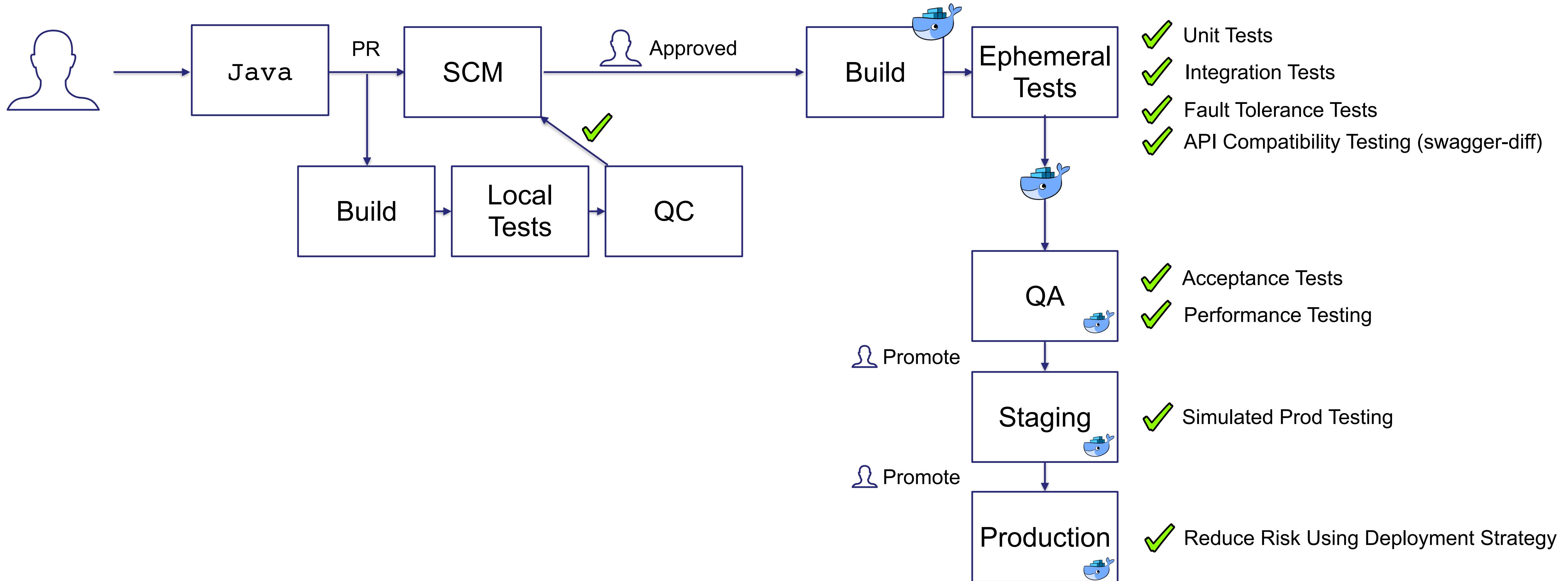
Delivering an API Program



Delivering an API Program



Continuous API Delivery



Daniel Bryant - Continuous Delivery in Java

#OReillySACon

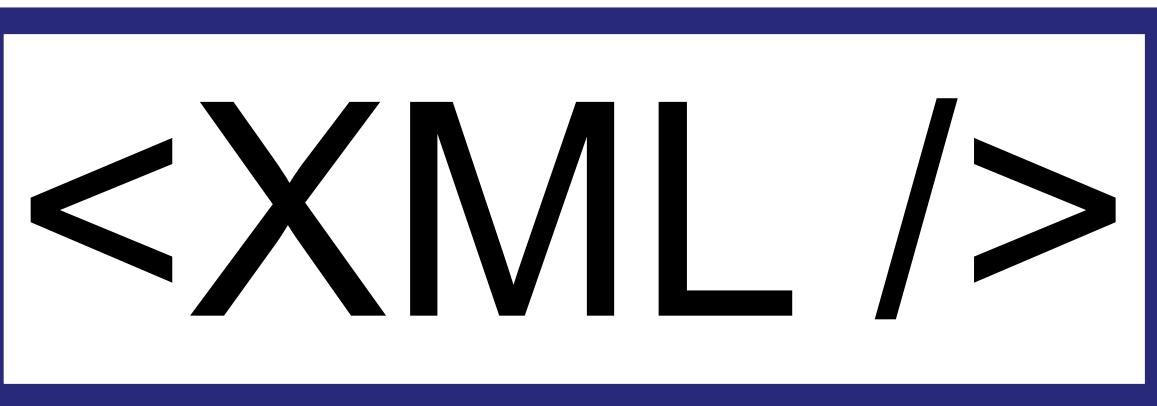
O'REILLY®
Software Architecture

Let's Go! Full Microservices?

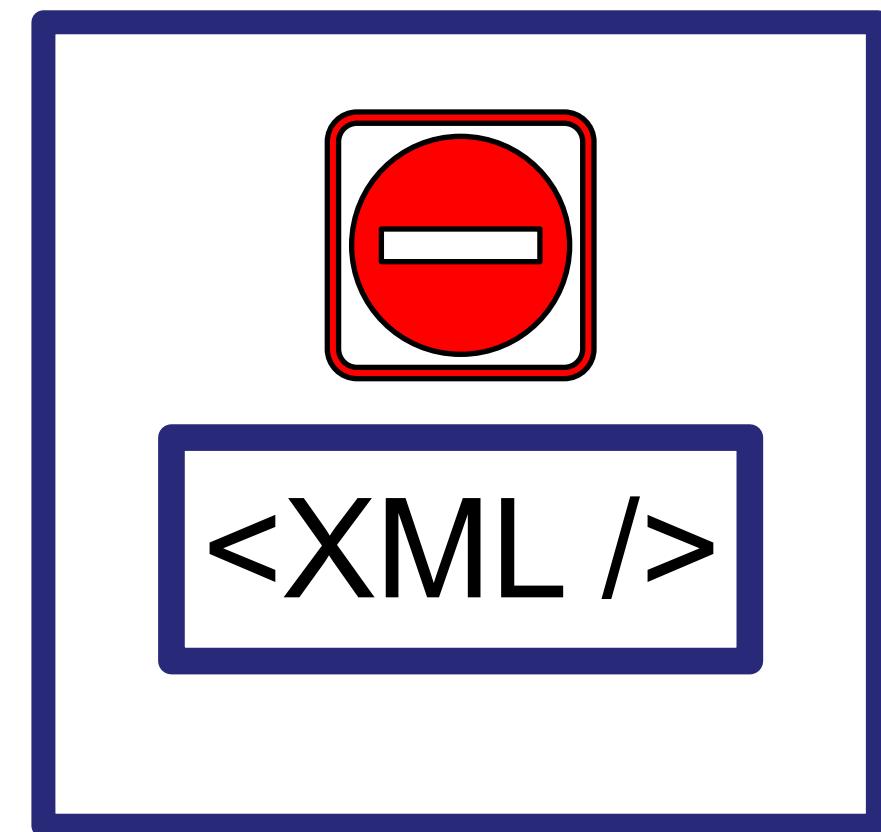
	Level 0 Traditional	Level 1 Basic	Level 2 Intermediate	Level 3 Advanced
Application	Monolithic	Service Oriented Integrations	Service Oriented Applications	API Centric
Database	One Size Fits All Enterprise DB	Enterprise DBs + No SQL	Polyglot, DBaaS	Matured Data Lake
Infrastructure	Physical Machines	Virtualization	Cloud	Containers
Monitoring	Infrastructure	App and Infra Monitoring	APMs	APM and Central Log Management
Process	Waterfall	Agile and CI	CI & CD	DevOps

Spring 5.0 Microservices - Second Edition by Rajesh R V

Building Microservices Using Java



Introducing Spring Boot



Demo Spring Boot

Greeter

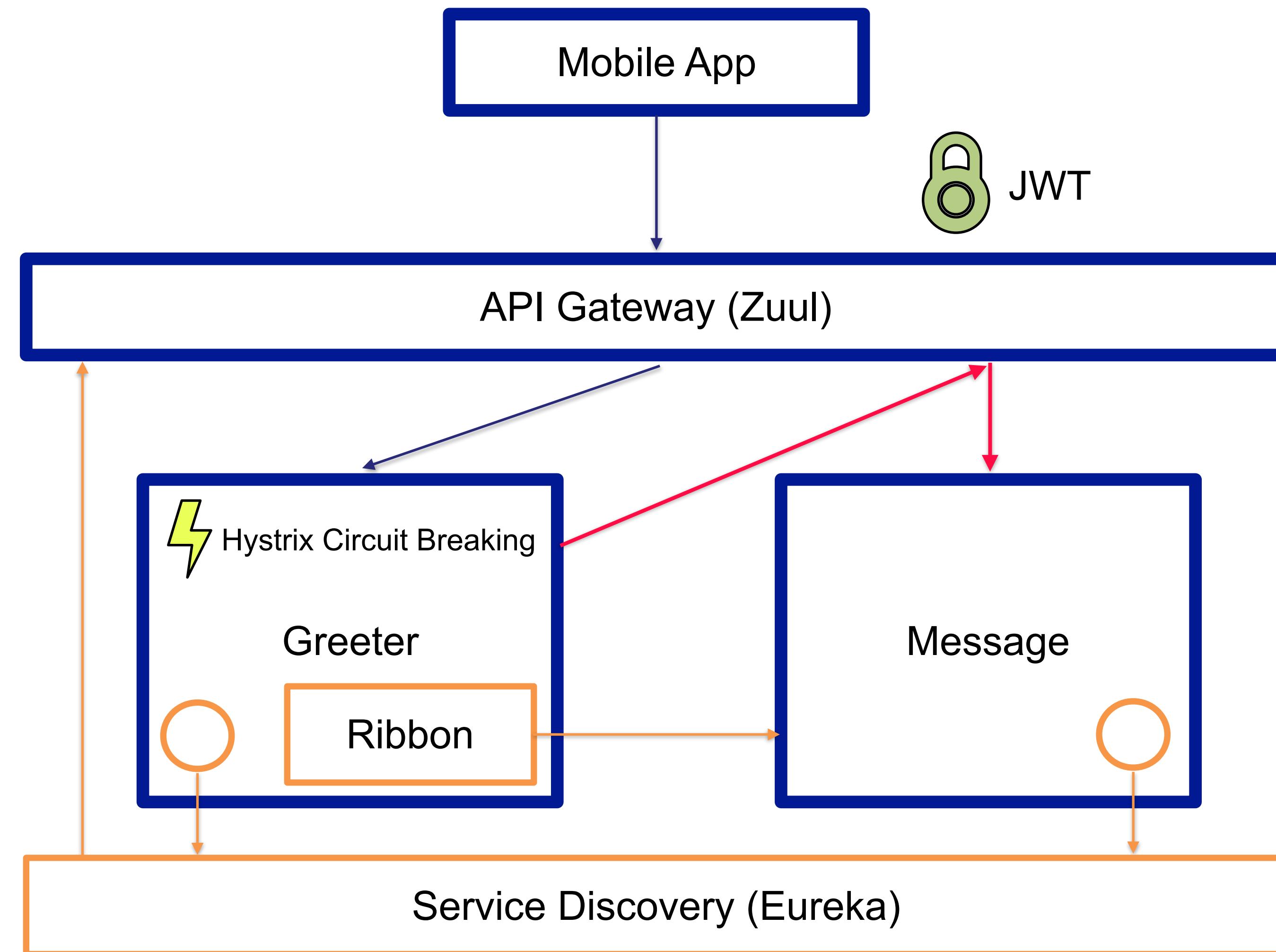
Message

Java is All We Need

Spring Boot
App

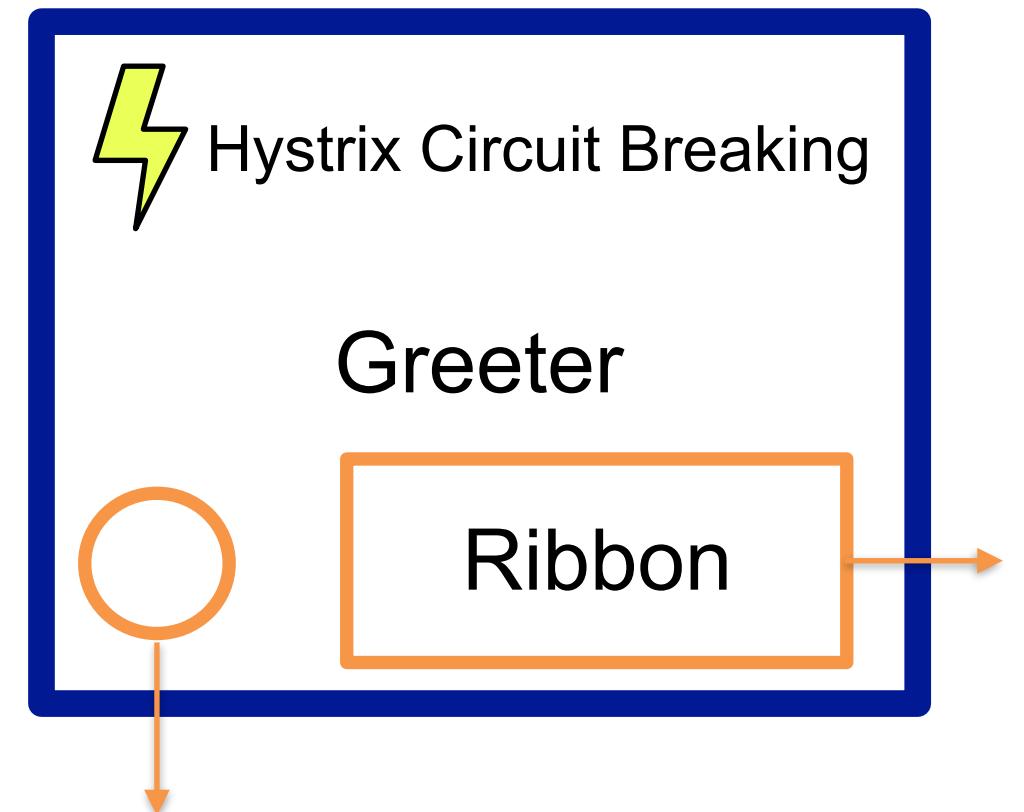
- Doesn't work on my machine!
 - Different version of Java installed
 - Different file system supported
- Consistency between environments
 - Can't be tested locally
- Coordinating Microservices is not just about Java
 - Use the right language for the job
- Coordinating Java is the wrong abstraction
 - Lots of scripts and Java specifics

The Netflix Stack



The Netflix Stack Today

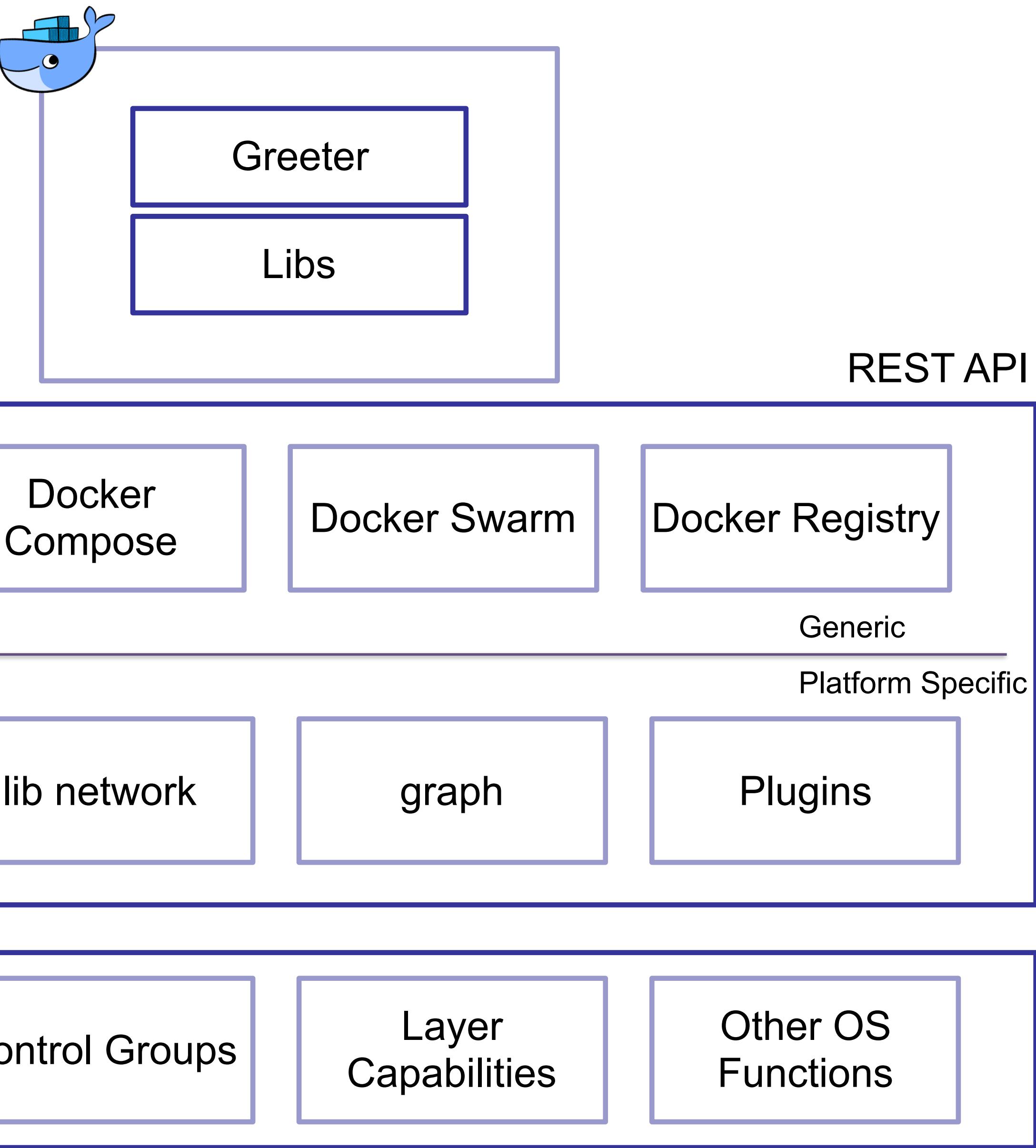
- Provided a set of libraries to solve most microservices problems
- Are these services still micro?
- Doesn't solve the problem of where to run services and deployment strategies
- Absolutely key in driving forward this complex technical space
- Environments can be challenging - a good place to start
- Can we do better? Is Java the wrong building block?



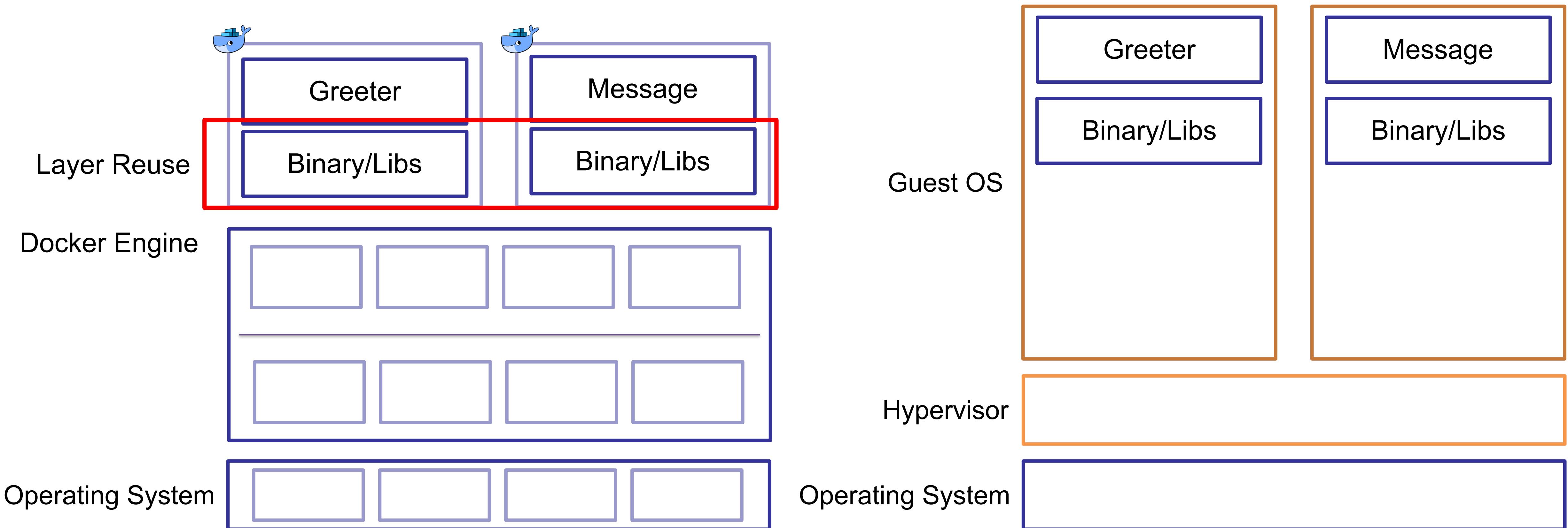
What is Docker?

Docker Engine

Operating System



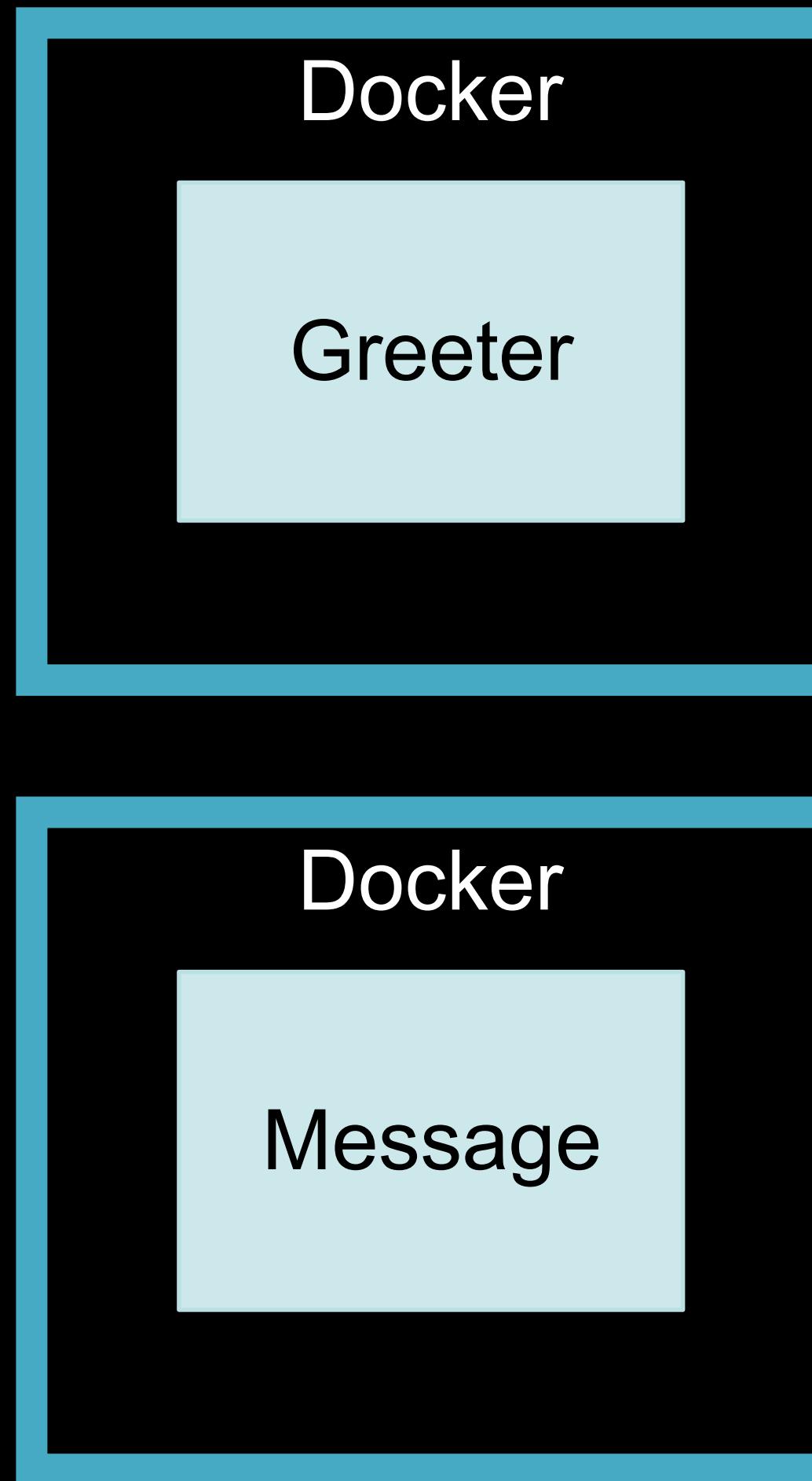
Difference between a VM and Docker



Advantages of using Containers

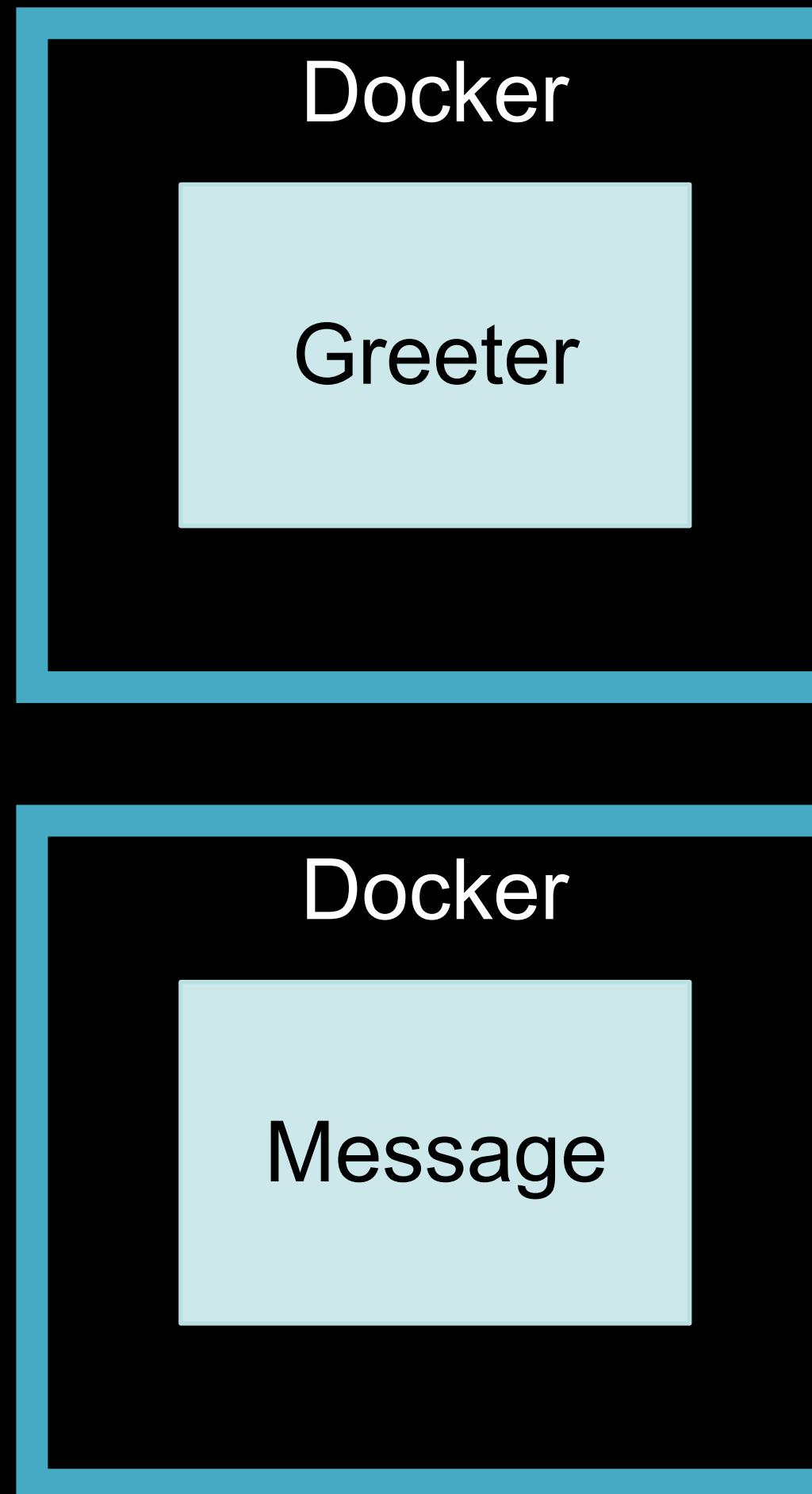
- Immutability
- Works across all environments and deployments
 - Easier to test
- Fundamental building block to simplify deployment
- Leave the specifics of the language behind

Demo Docker



```
$ cat Dockerfile  
FROM openjdk:8-jdk-alpine  
VOLUME /tmp  
ADD build/libs/greeter-0.0.1-SNAPSHOT.jar app.jar  
EXPOSE 8080  
ENV JAVA_OPTS=""  
ENTRYPOINT [ "sh", "-c", "java $JAVA_OPTS -Djava.security.egd=file:/dev/.urandom -jar
```

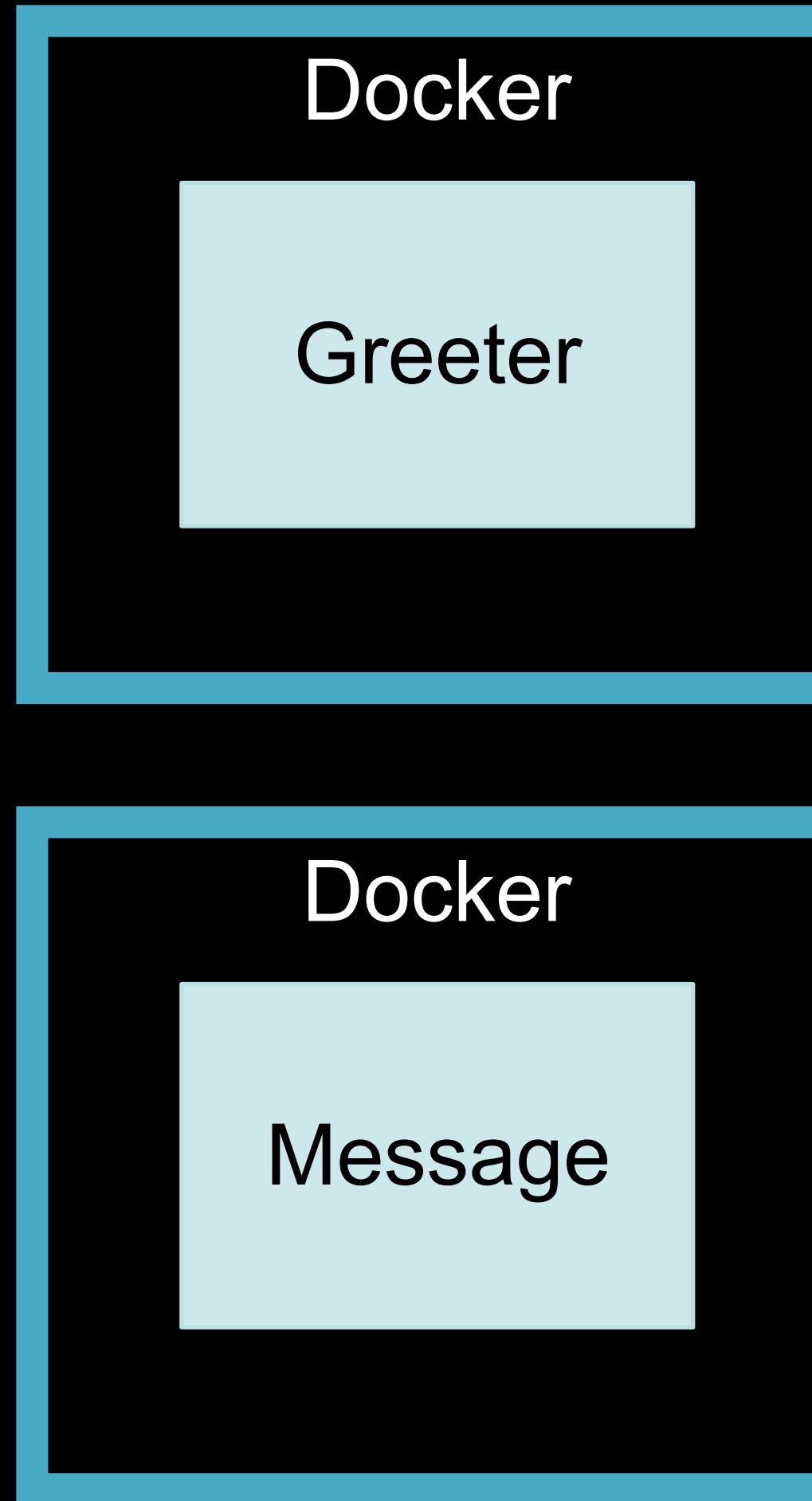
Demo Docker



```
$ docker build -t greeter:conf .

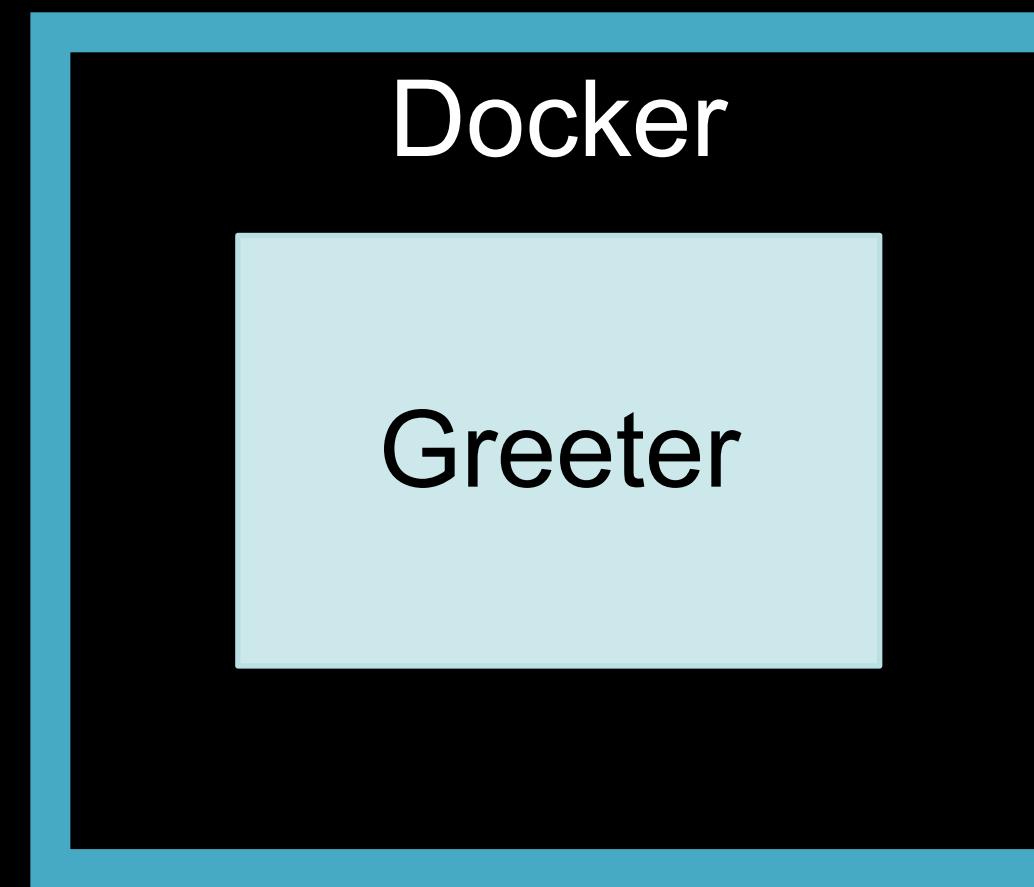
Sending build context to Docker daemon 24.29MB
Step 1/6 : FROM openjdk:8-jdk-alpine
--> 54ae553cb104
Step 2/6 : VOLUME /tmp
--> Using cache
--> be3f9b66169a
Step 3/6 : ADD build/libs/greeter-0.0.1-SNAPSHOT.jar app.jar
--> Using cache
--> 9ebc7b30c927
Step 4/6 : EXPOSE 8080
--> Using cache
--> 13e6d1568396
Step 5/6 : ENV JAVA_OPTS=""
--> Using cache
--> 9659c9657c0b
Step 6/6 : ENTRYPOINT [ "sh", "-c", "java $JAVA_OPTS -Djava.security.egd=file:/dev/.urandom -jar /app.jar" ]
--> Using cache
--> 7622a23bac14
Successfully built 7622a23bac14
Successfully tagged greeter:conf
```

Demo Docker



```
$ docker run -p 8081:8081 -t  
message:conf
```

Demo Docker



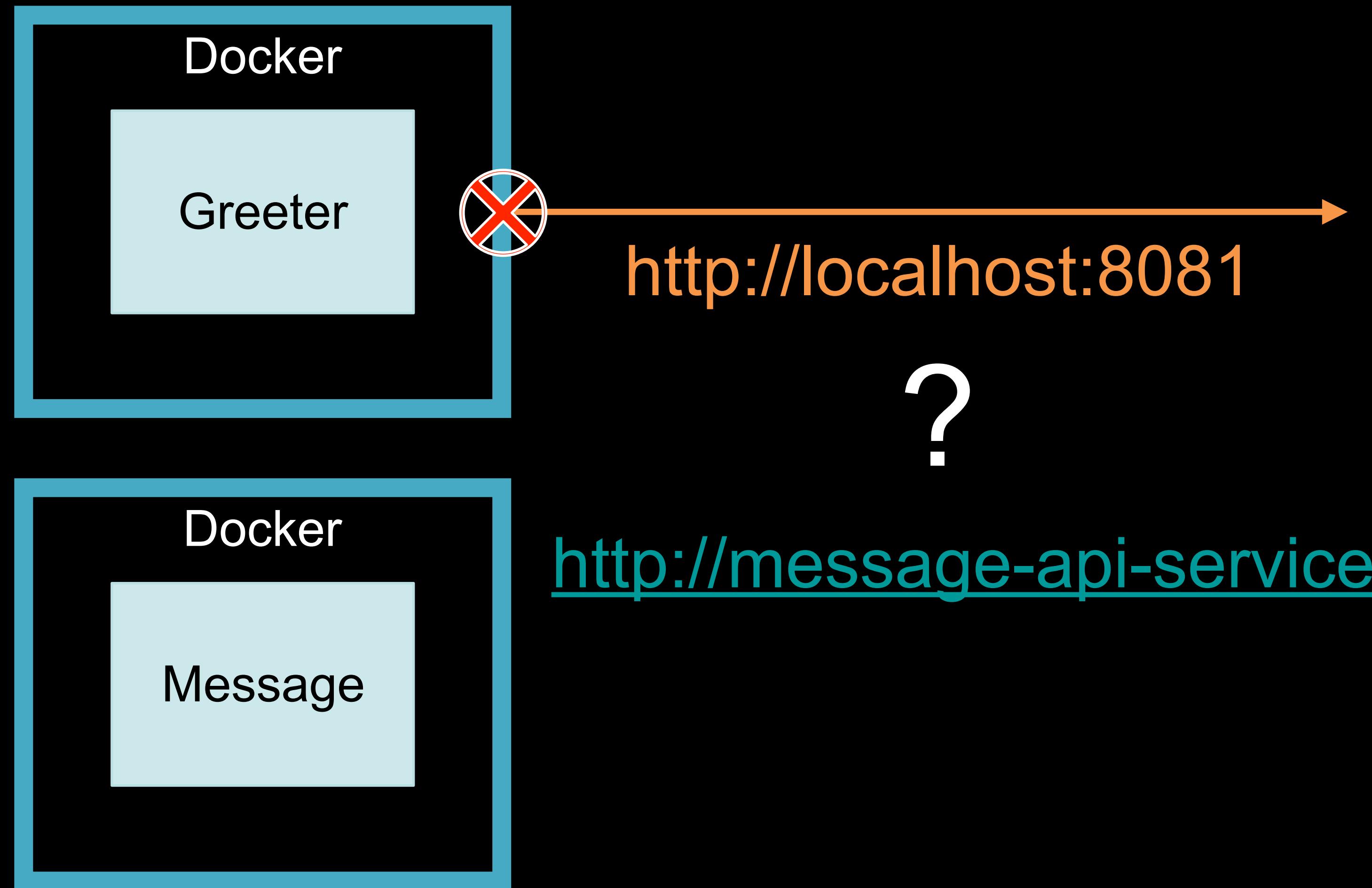
```
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND
CREATED	STATUS	PORTS
NAMES		
224ab1a98b3f	message:conf	"sh -c 'java
\$JAVA_0..."	About a minute ago	Up About a minute
0.0.0.0:8081->8081/tcp	naughty_leakey	
912c04bda714	greeter:conf	"sh -c 'java
\$JAVA_0..."	2 minutes ago	Up 2 minutes
0.0.0.0:8080->8080/tcp	boring_hermann	

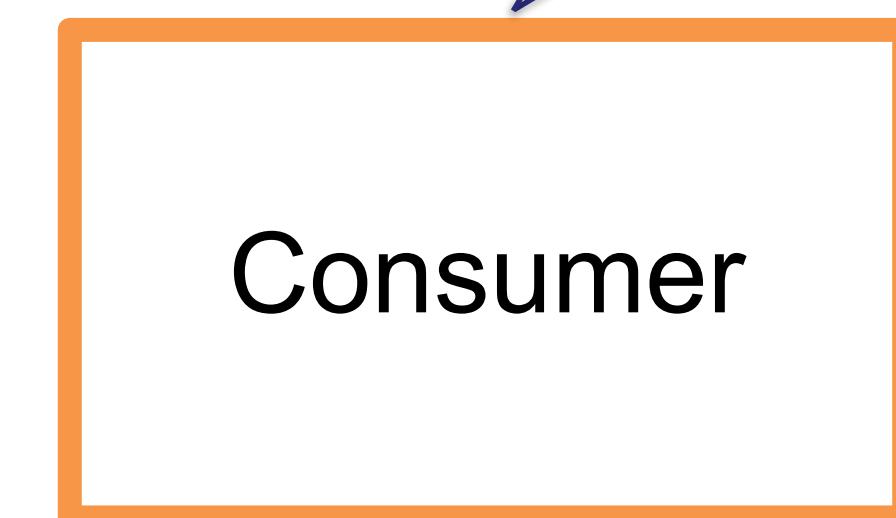
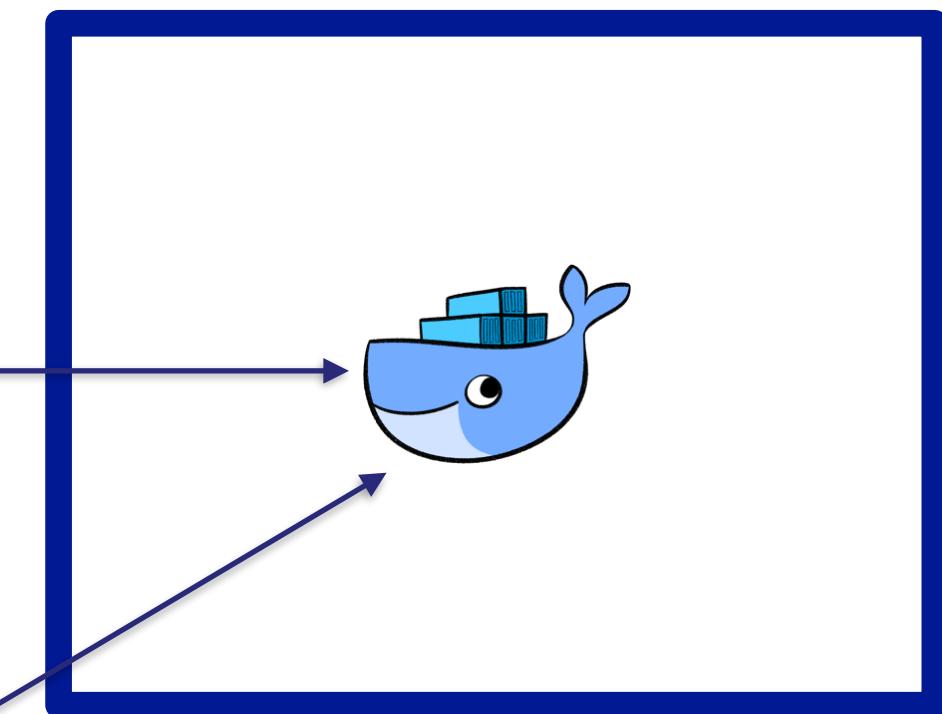
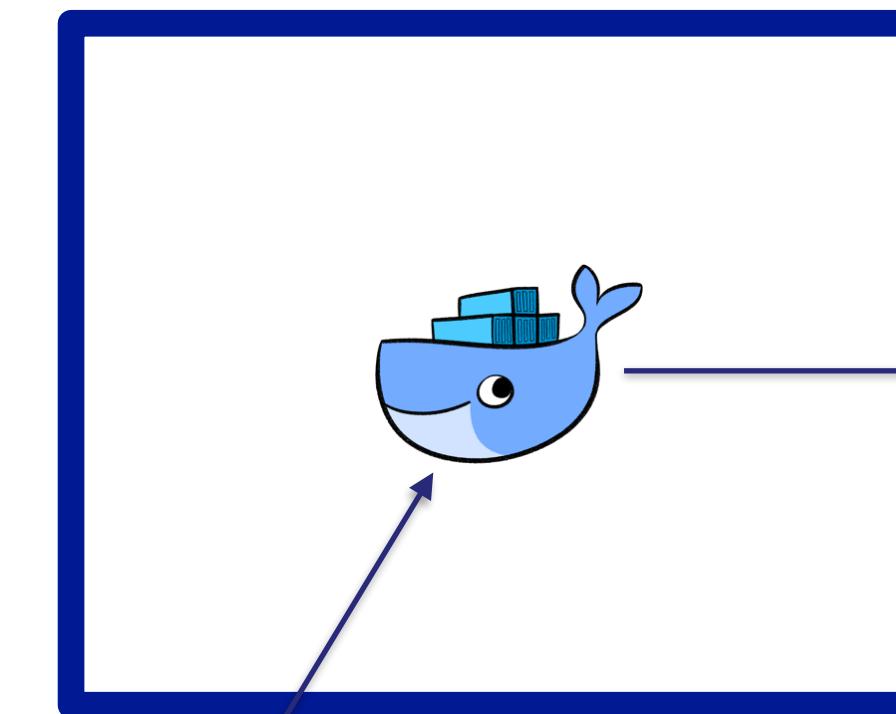
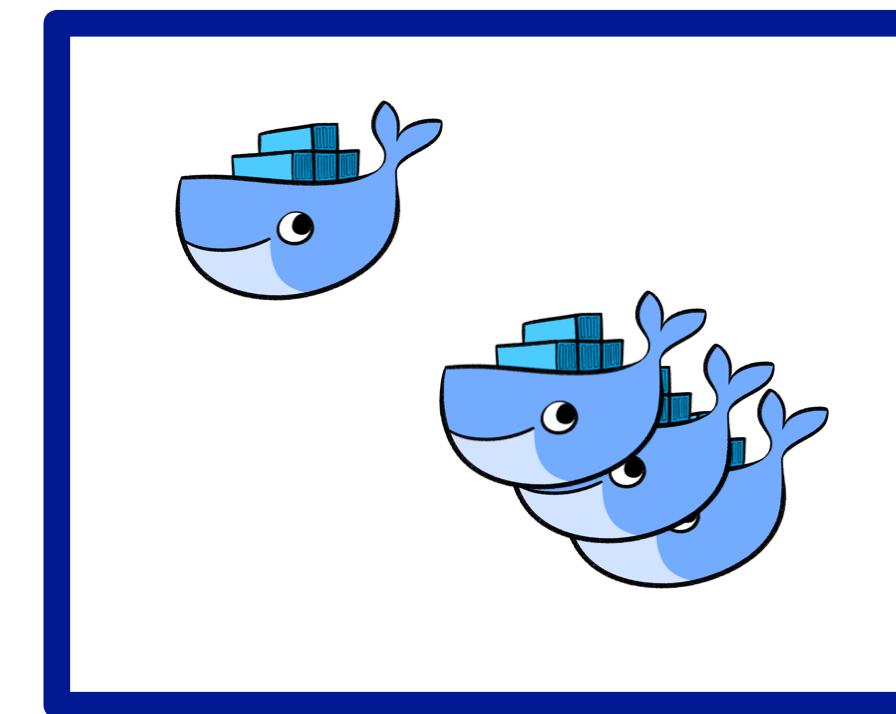
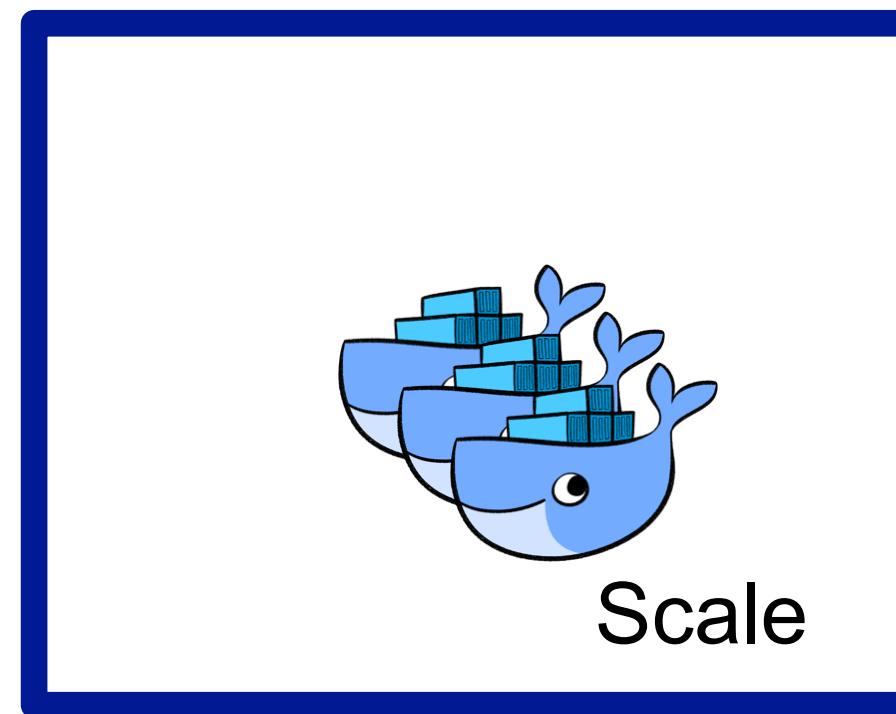
```
$ curl http://localhost:8080/simple
{"message": "Hello Conference"}
```

```
$ curl http://localhost:8080/custom
{"timestamp": "2018-10-27T12:29:45.679+0000", "status": 500, "error": "Internal Server Error"}
```

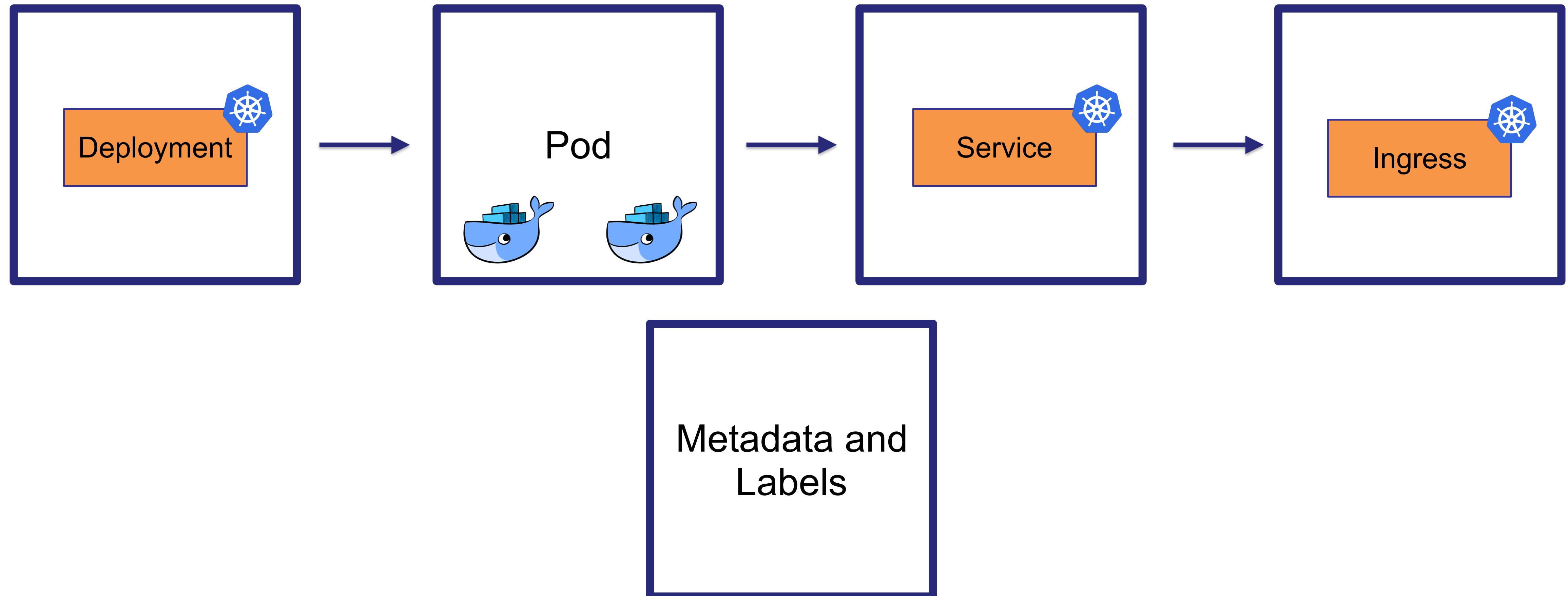
Demo Docker



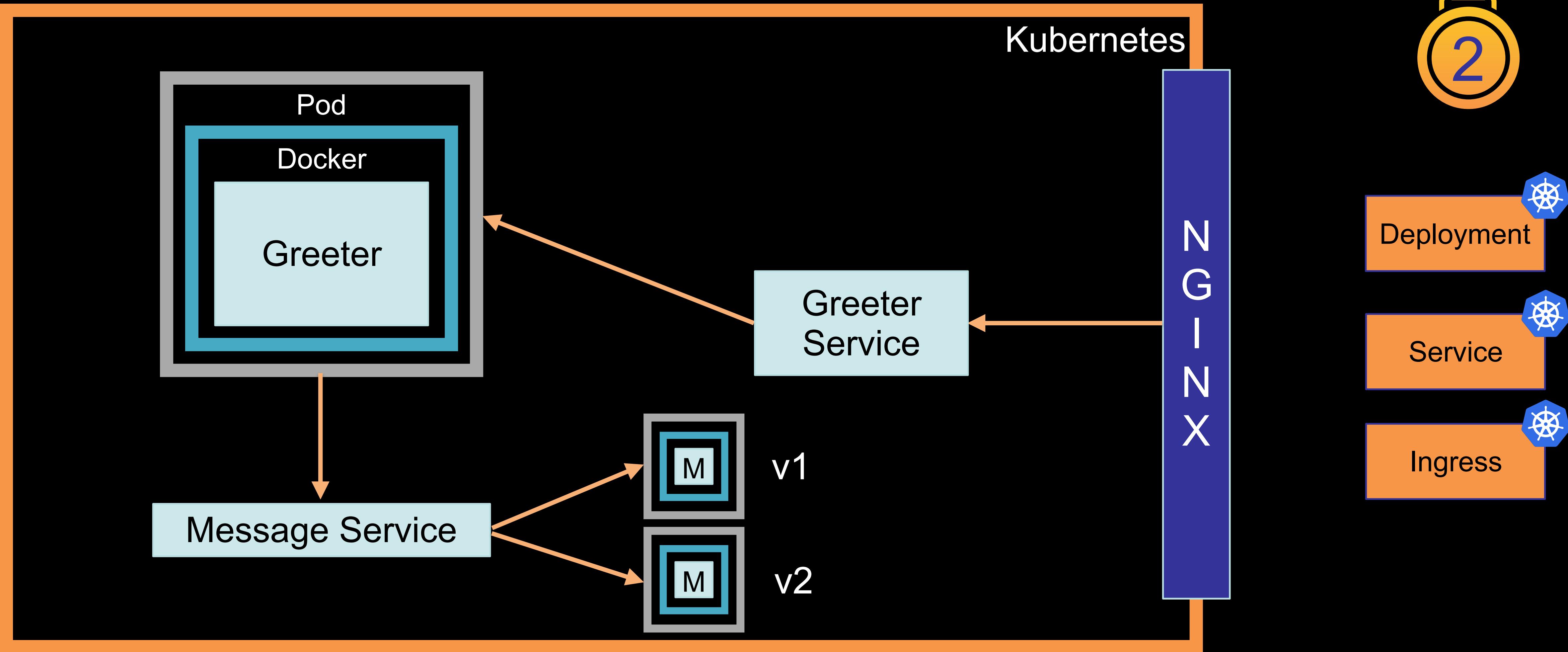
Scheduling and Running Containers



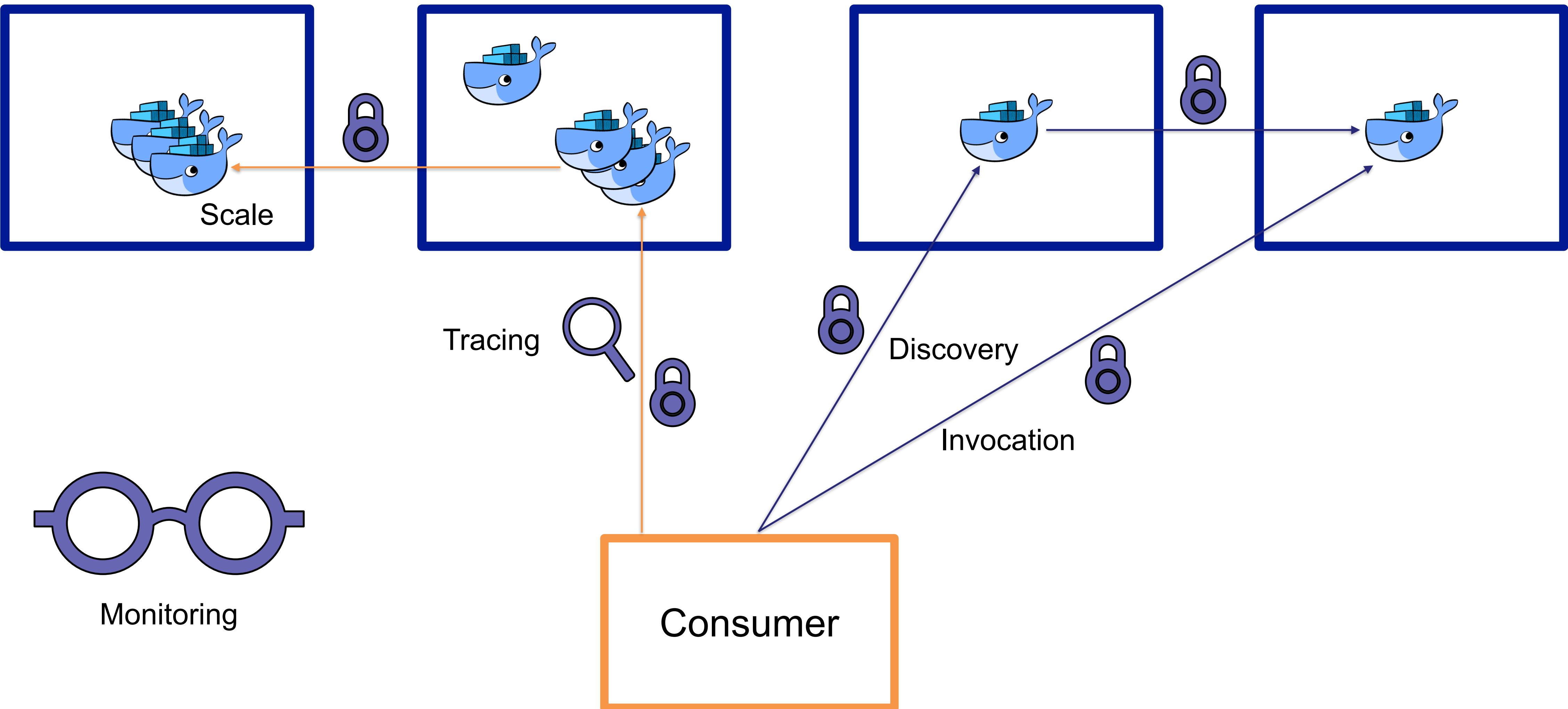
Components of Kubernetes



Demo Kubernetes

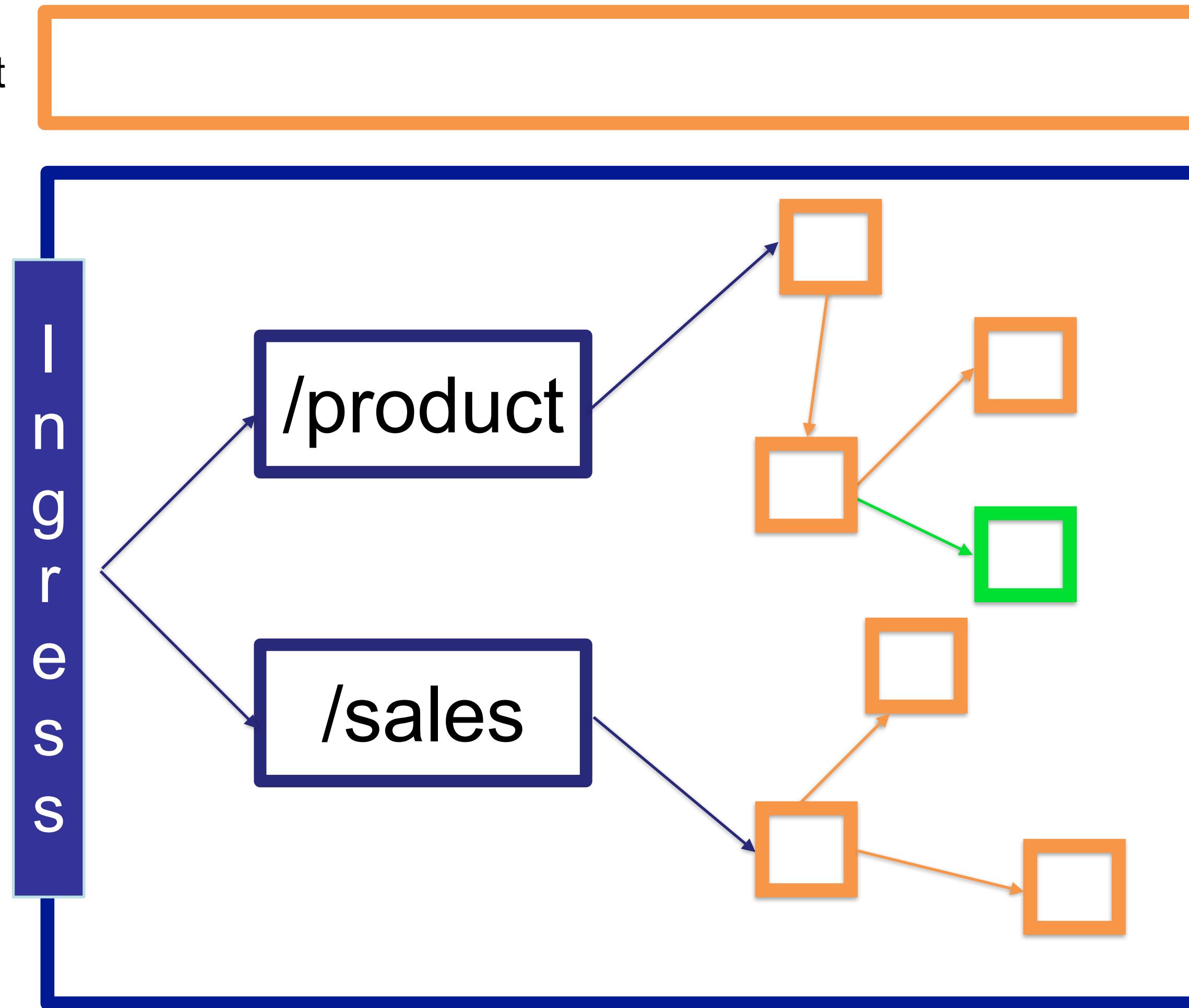


Scheduling and Running Containers Revisited



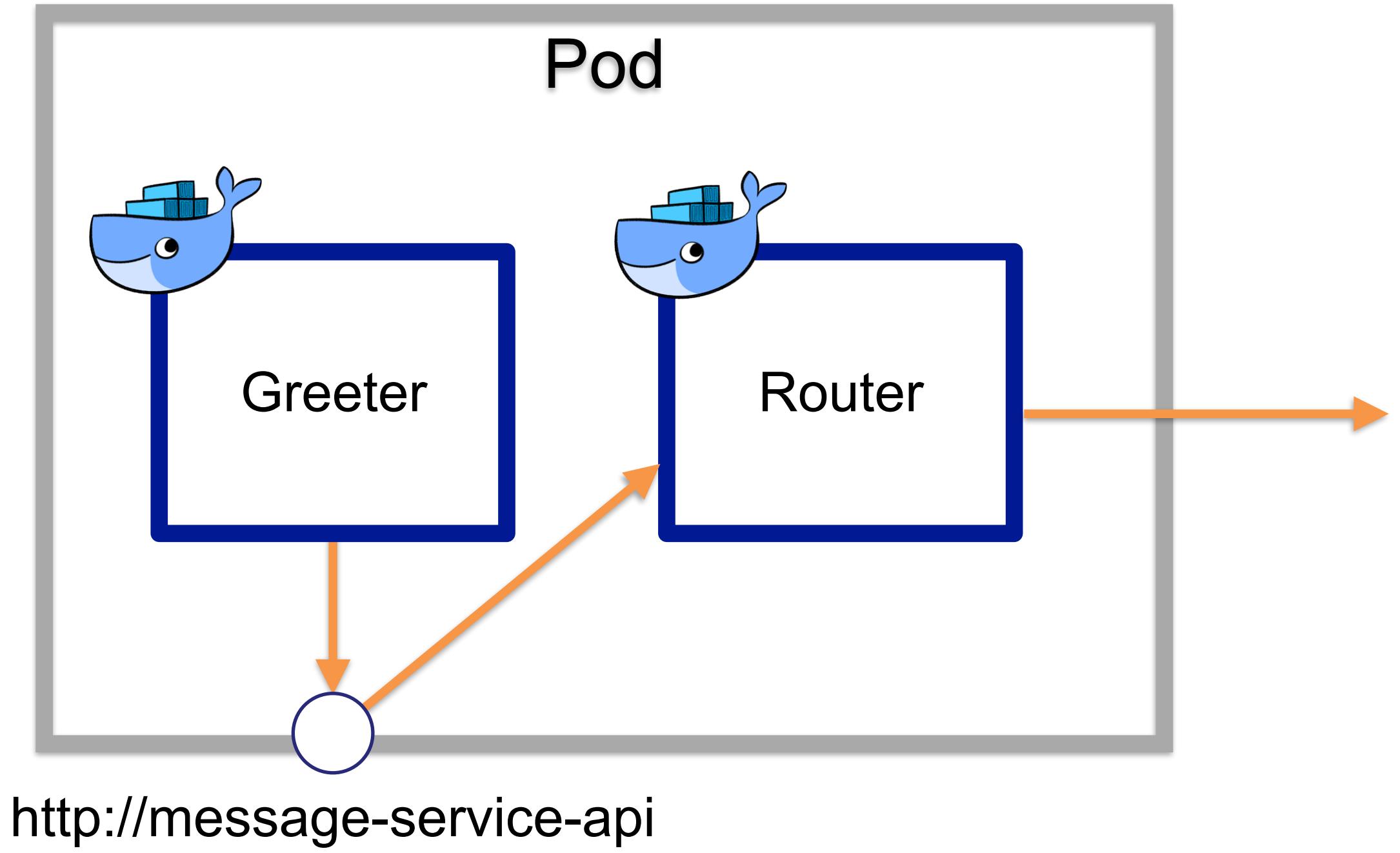
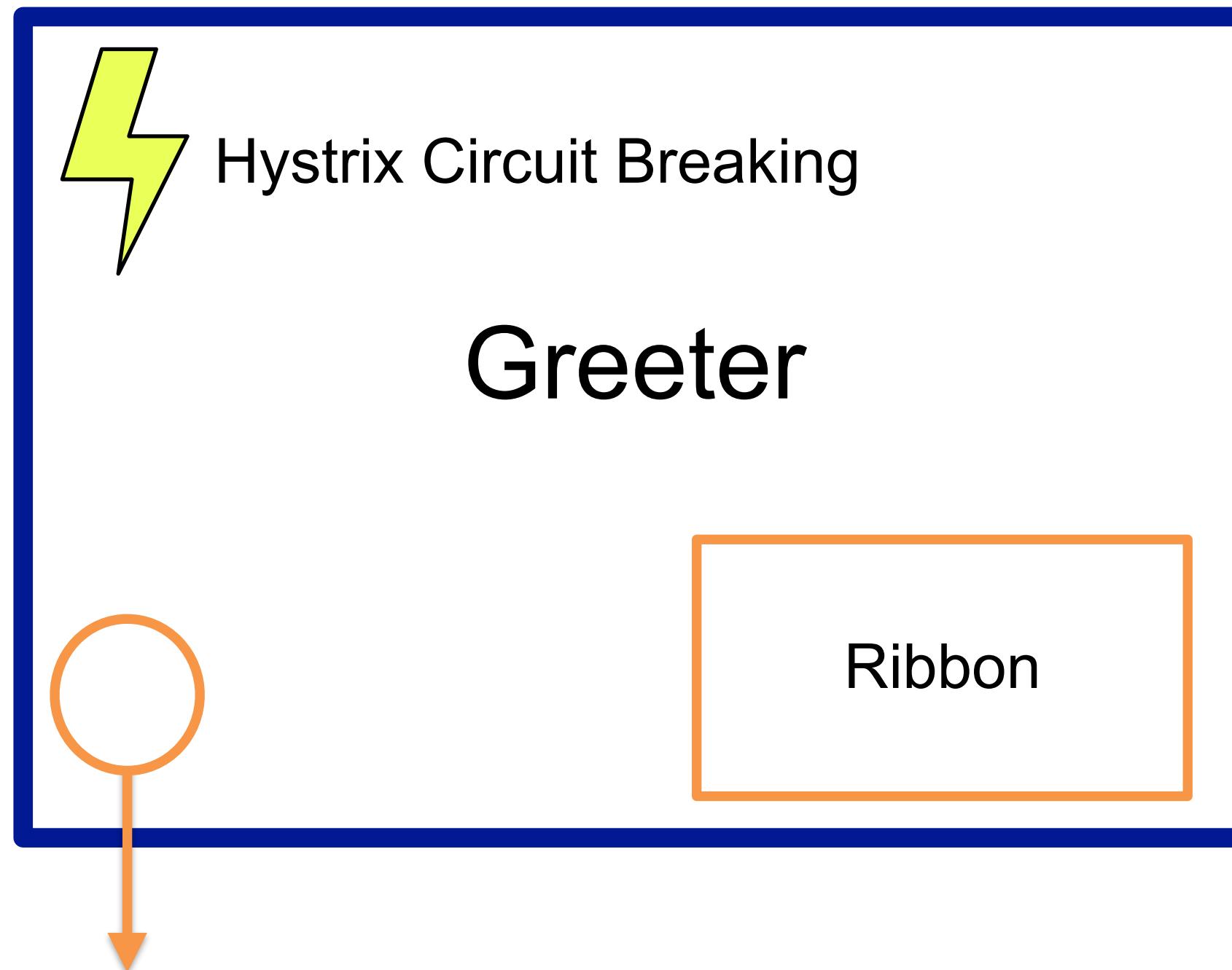
Introducing Service Mesh

Control and Management



- Security
- Tracing
- Dealing with Failures

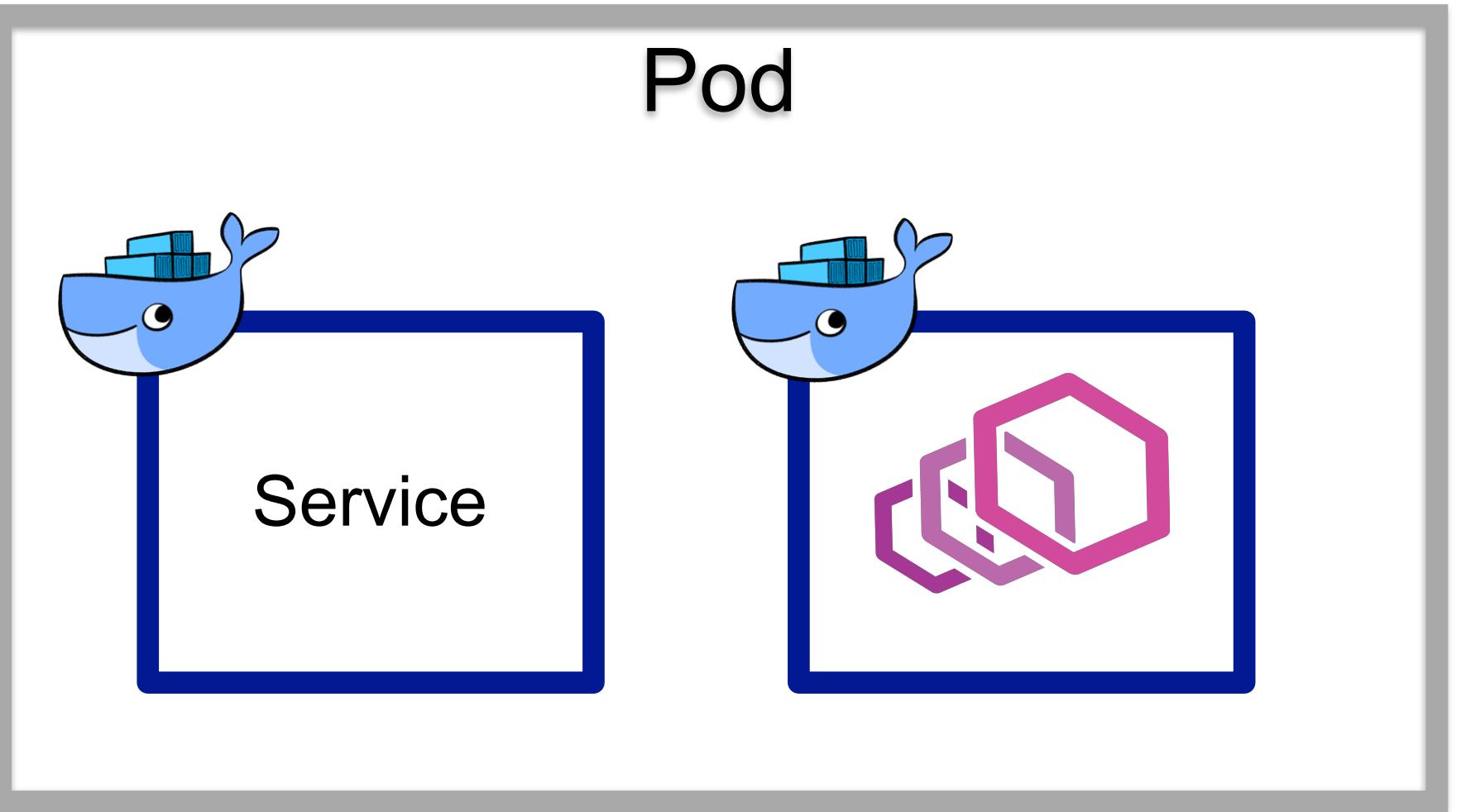
Decoupled APIs Revisited



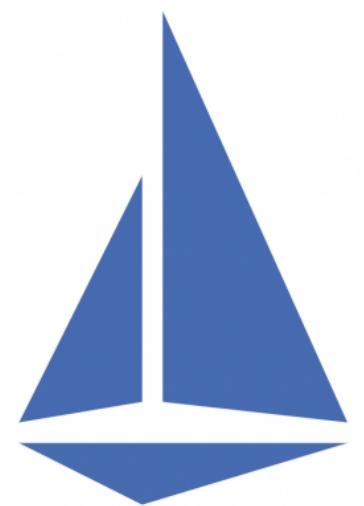
Istio - Building on the Envoy Sidecar



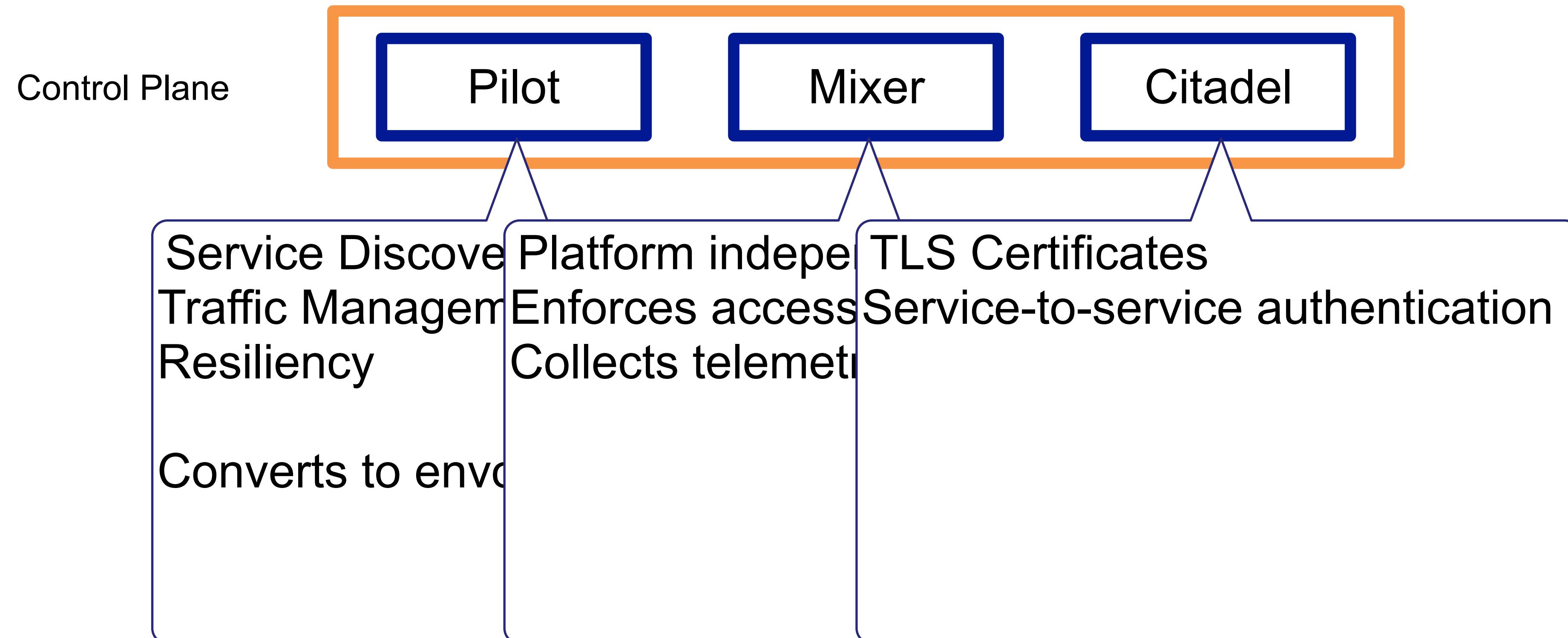
- Dynamic service discovery
- Load balancing
- TLS termination
- HTTP/2 and gRPC proxies
- Circuit breakers
- Health checks
- Staged rollouts with %-based traffic split
- Fault injection
- Rich metrics



<https://istio.io/docs/concepts/what-is-istio/>

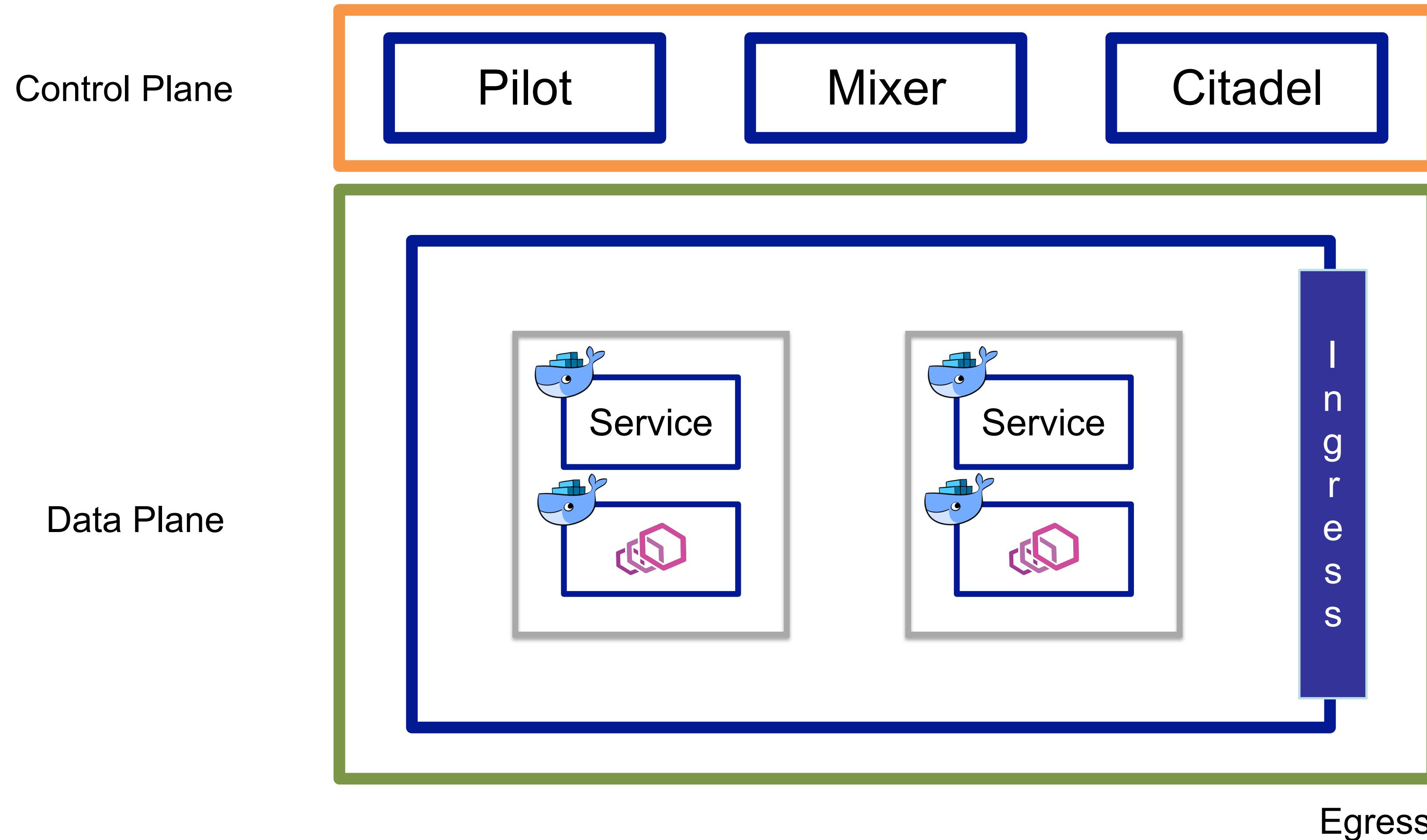


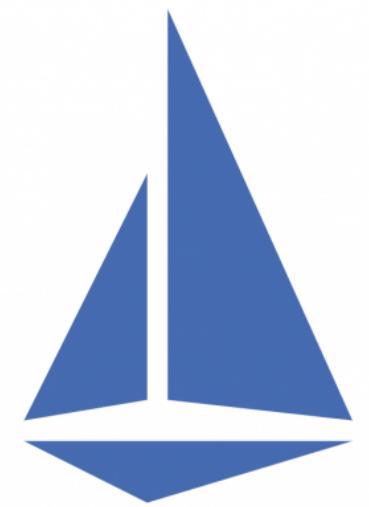
Istio Main Components



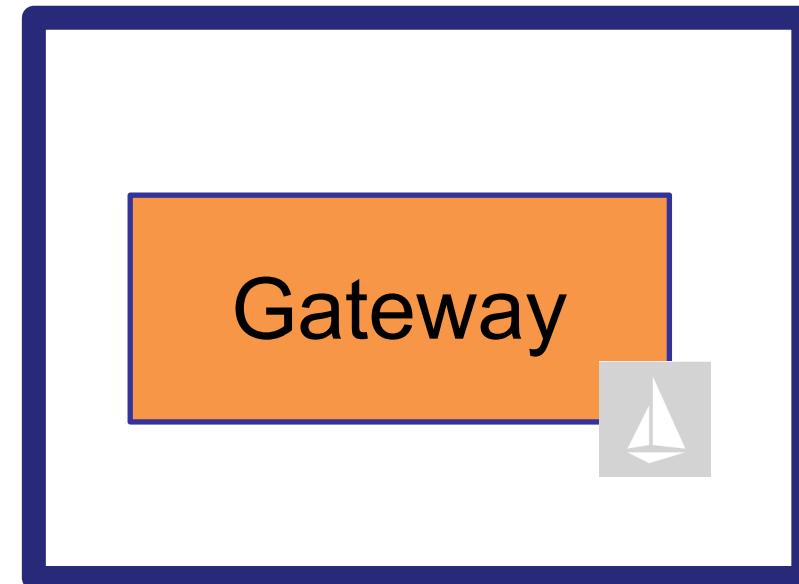


Istio Main Components

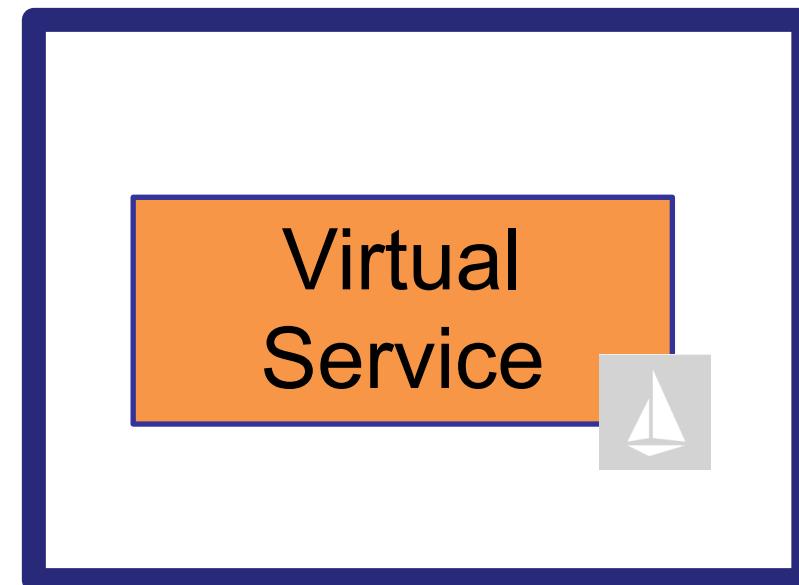




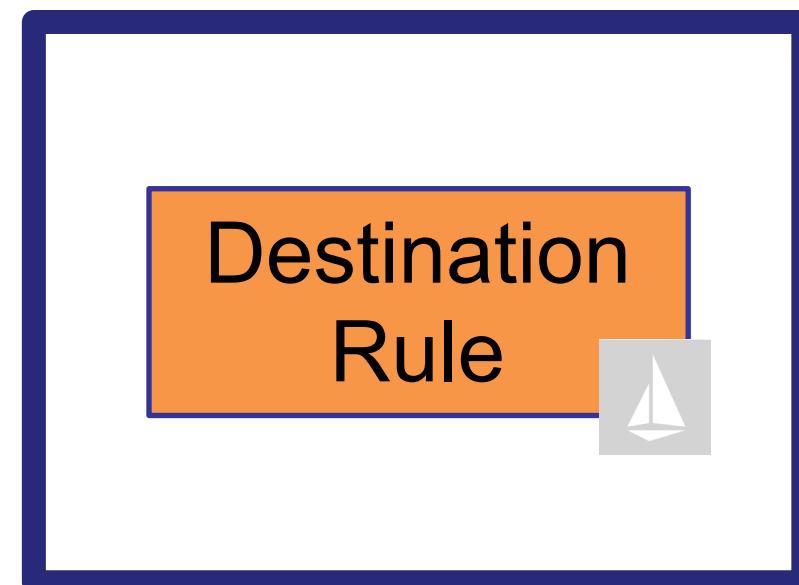
Istio Networking Configuration



Configure a load balancer for incoming HTTP Traffic
Attach against Ingress

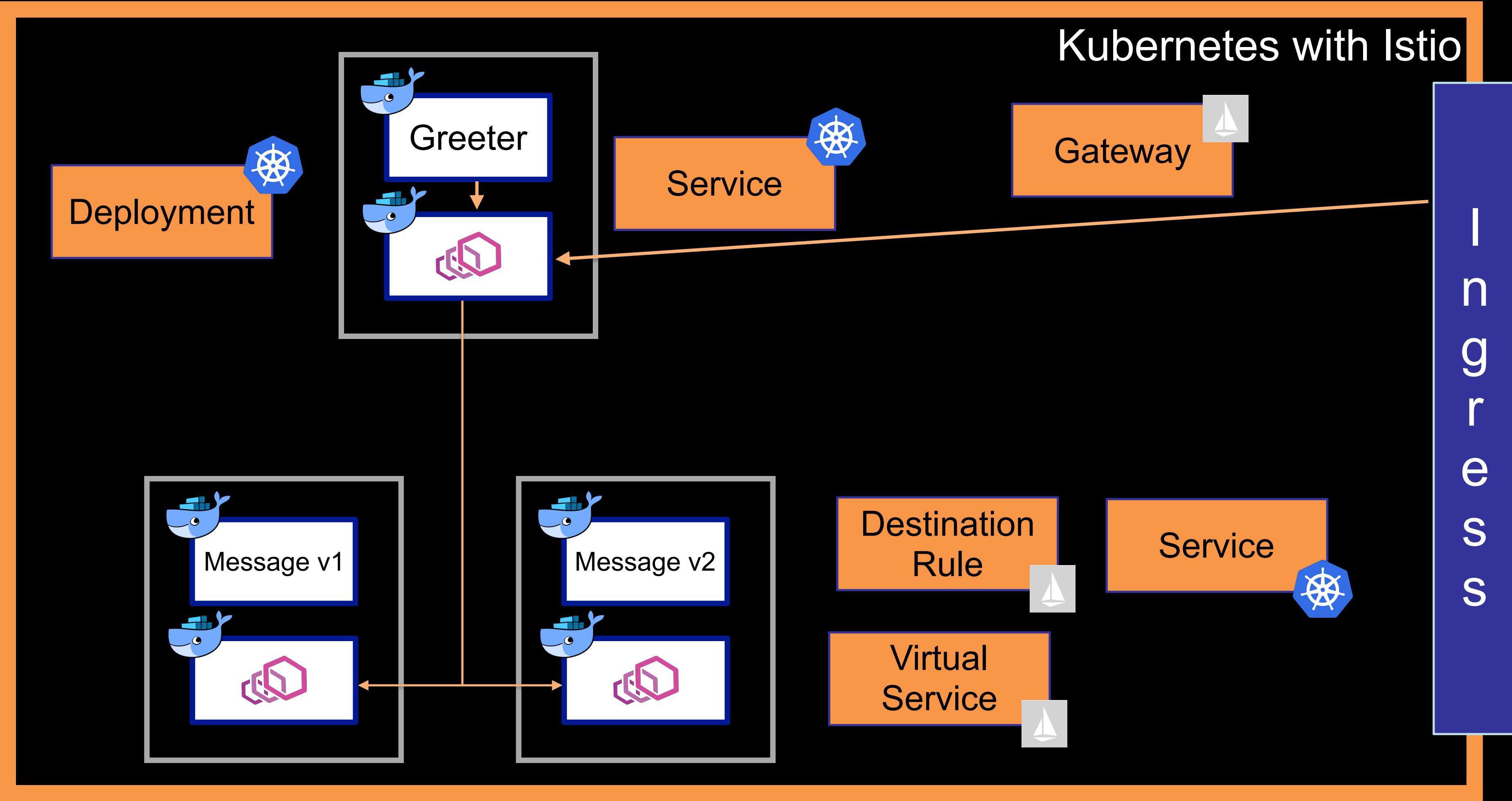


Configure how requests are routed within the Istio service mesh



Configure traffic after VirtualService routing, e.g Least Connected

Demo Istio on Google Cloud



API Management - Customer View

The screenshot displays the API Marketplace interface, specifically the customer view. The left sidebar contains a navigation menu with categories such as Tools, Education, Devices, Finance, Advertising, Commerce, Other, Location, Business, Social, Communication, Entertainment, Media, Medical, Sports, Reward, Data, and Travel. The main content area shows eight API listings arranged in two rows of four. Each listing includes the API name, developer, logo, brief description, price tier, and user statistics.

Name	Developer	Description	Price	Reviews	Popularity
RANDOM FAMOUS QUOTES	ANDRUXNET	Get a random quote in JSON format. Current categories are: - famous - movies	FREE	16448 reviews (★ 16205)	100% up
FACEPLUSPLUS FACE DETECTION	FACEPLUSPLUS	Detect the information of the given photo(e.g. face location, age, race, gender etc.)	FREE	7666 reviews (★ 7168)	100% up
SENTIMENT	VIVEKN	This tool works by examining individual words and short sequences of words (n-grams) and comparing them with a	FREE	7243 reviews (★ 6744)	100% up
HEARTHSTONE	OMGVAMP	This API provides up to date Hearthstone data pulled directly from the game.	FREE	7175 reviews (★ 7215)	100% up
GEOCODE LOCATION LOOKUP	MONTANAFLYNN	Geocode and reverse lookup location information by IP address, address, or latitude / longitude coordinates.	FREE	5658 reviews (★ 5335)	100% up
EMAIL VALIDATOR	POZZAD	Checks for fake DNS as well as uses regex functions to check the email for the right length and accepted characters. For	FREE	5472 reviews (★ 4987)	100% up
TEXT TO SPEECH	MONTANAFLYNN	Turns text into an mp3 audio file with a nice female voice similar to Siri.	FREE	5107 reviews (★ 4845)	100% up
RECIPE - FOOD - NUTRITION	SPOONACULAR	The spoonacular Nutrition, Recipe, and Food API allows you to access over 365,000 recipes and 86,000 food products.	FREEMIUM	4779 reviews (★ 4623)	100% up

<https://market.mashape.com>

API Management - Customer View

The screenshot shows the API Marketplace interface for the Hearthstone API. The top navigation bar includes links for marketplace, search APIs, explore APIs, docs, features, add your API, sign up free, and login. Below the navigation is a header with documentation, announcements, API support, and a user count of 18. The main content area is divided into sections: Endpoints, All Cards, URL Parameters, Request Example, and Request Headers.

Endpoints

- GET All Cards**
- GET Card Backs
- GET Card Search
- GET Card Set
- GET Cards by Class
- GET Cards by Faction
- GET Cards by Quality
- GET Cards by Race
- GET Cards by Type
- GET Info
- GET Single Card

All Cards
Returns all available Hearthstone cards including non collectible cards.

URL PARAMETERS

Name	Type	Description
attack	NUMBER	optional Return only cards with a certain attack.
callback	STRING	optional Request data to be returned as a JsonP callback.
collectible	NUMBER	optional Set this to 1 to only return collectible cards.
cost	NUMBER	optional Return only cards of a certain cost.
durability	NUMBER	optional Return only cards with a certain durability.
health	NUMBER	optional Return only cards with a certain health.
locale	STRING	optional What locale to use in the response. Default locale is enUS. Available locales: enUS, enGB, deDE, esES, esMX, frFR, itIT, koKR, ptPT, ruRU, zhCN, zhTW, jaJP, thTH.

REQUEST EXAMPLE

curl --get --include 'https://omgvamp-hearthstone-v1.p.mashape.com/cards' \ -H 'X-Mashape-Key: <required>'

REQUEST HEADERS

X-Mashape-Key **SIGN UP TO CONSUME THIS API**

The Mashape application you want to use for this session.

TEST ENDPOINT

<https://market.mashape.com>

API Management - Development View

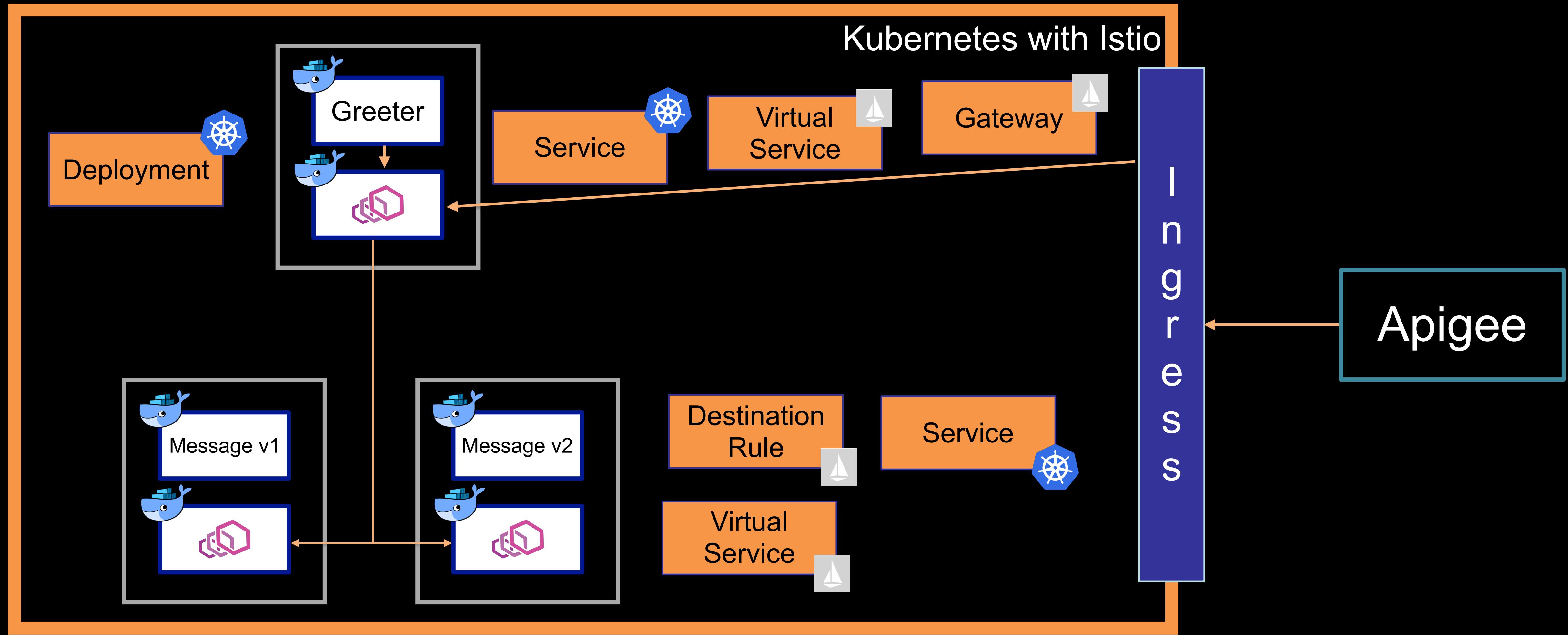
The screenshot shows the API Management development interface. The left sidebar has a user profile (JG, James Gough) and navigation links: DEVELOP (selected), Specs, API Proxies (highlighted in red), Shared Flows, Offline Trace, and API BaaS. The main area shows a flow editor titled "API Proxies > Birthday > Develop > 1". The flow is named "PreFlow" and consists of a REQUEST and a RESPONSE. The REQUEST path starts with "JSON to XML-1" and ends at "default". The RESPONSE path starts from "default" and ends with "PostFlow". The "default" section contains several API endpoints: "getHelloForCliffUsingGET", "getDayOfBirthUsingPOST", "todayUsingGET", and "PostFlow". The "Code" section displays the XML configuration for the "default" proxy endpoint:

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <ProxyEndpoint name="default">
3    <Description/>
4    <FaultRules/>
5    <PreFlow name="PreFlow">
6      <Request/>
7      <Response/>
8    </PreFlow>
9    <PostFlow name="PostFlow">
10      <Request/>
11      <Response/>
12    </PostFlow>
13    <Flows>
14      <Flow name="getHelloForCliffUsingGET">
15        <Description>getHelloForCliff</Description>
```

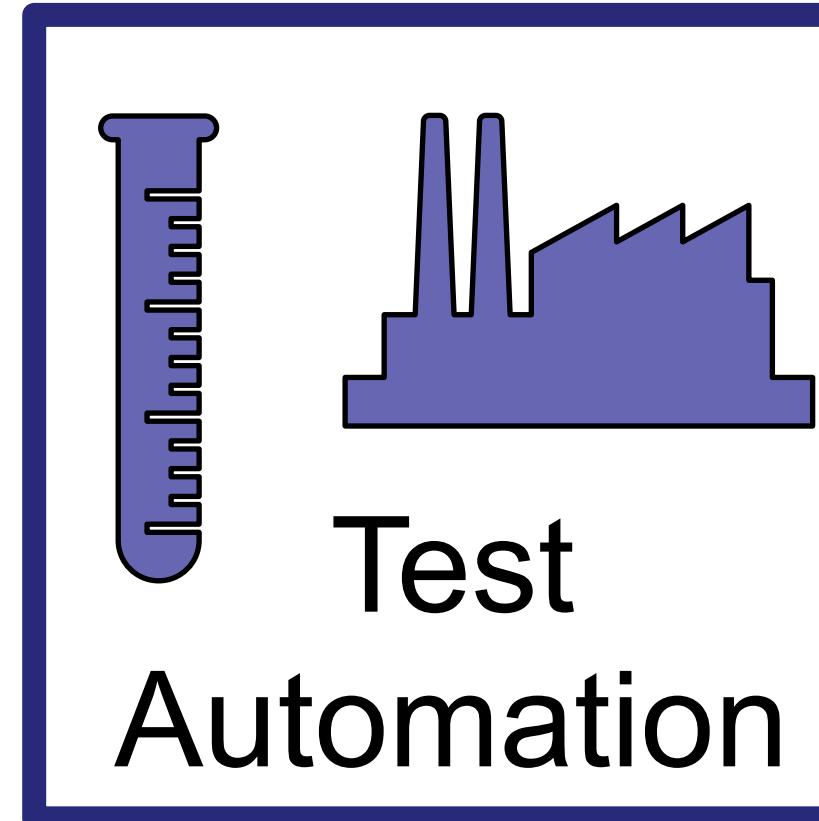
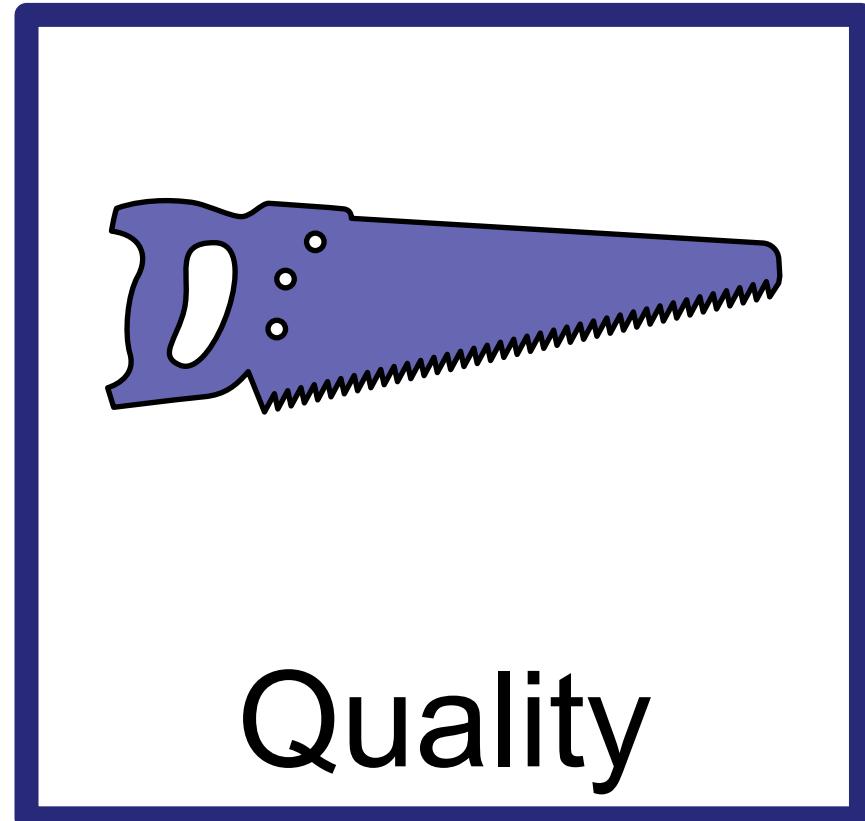
The "Property Inspector" panel on the right shows the properties for the selected "PreFlow" step, including "name": "PreFlow", "Request", and "Response".

<https://apigee.com/>

Demo Istio with Apigee



Summary



@Jim__Gough



	Level 0 Traditional	Level 1 Basic	Level 2 Intermediate	Level 3 Advanced
Application	Monolithic	Service Oriented Integrations	Service Oriented Applications	API Centric
Database	One Size Fits All Enterprise DB	Enterprise DBs + No SQL	Polyglot, DBaaS	Matured Data Lake
Infrastructure	Physical Machines	Virtualization	Cloud	Containers
Monitoring	Infrastructure	App and Infra Monitoring	APMs	APM and Central Log Management
Process	Waterfall	Agile and CI	CI & CD	DevOps