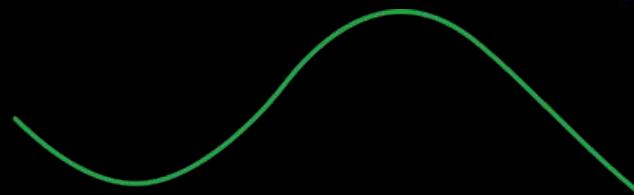
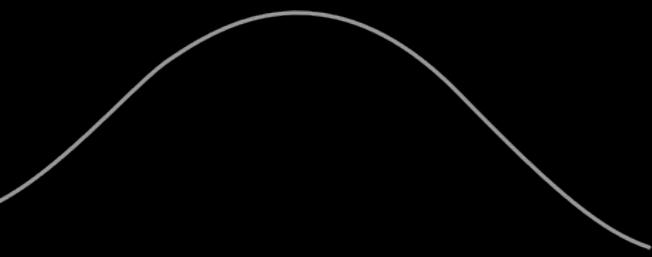


PACTFLOW

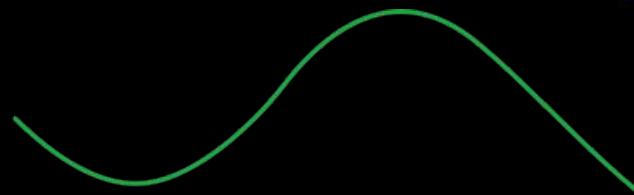
Brought to you by DIUS





PACTFLOW

Brought to you by DIUS



Integration testing is *hard...*

Our **vision** is to **transform** the way teams test and release  
**distributed systems.**

# The numbers

Four key indicators of high performing organisations<sup>1</sup>



Need < 1 day **lead time** for changes = **106x** faster time from commit -> deploy



Are able to **deploy** on demand = **208x** more deployments



Have **change failures** rates < 15% = **7x** lower change failure rates



Can **restore services** within 1 hour = **2604x** faster MTTR

# The numbers

## Challenges facing the market

Only **20%** of companies are “elite” performers<sup>1</sup>

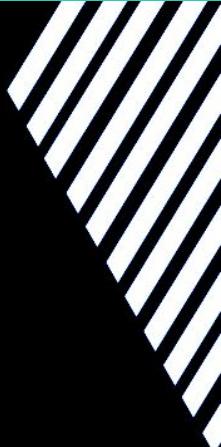
**81%** of teams spend a third of their time or more on **fixing environments**<sup>2</sup>

**36%** of teams are impacted by wait times and cost of **test environments**<sup>2</sup>

**76%** spent one third of their time or more managing **test data**<sup>2</sup>

<sup>1</sup> Data from the DORA 2019 State of DevOps [report](#)

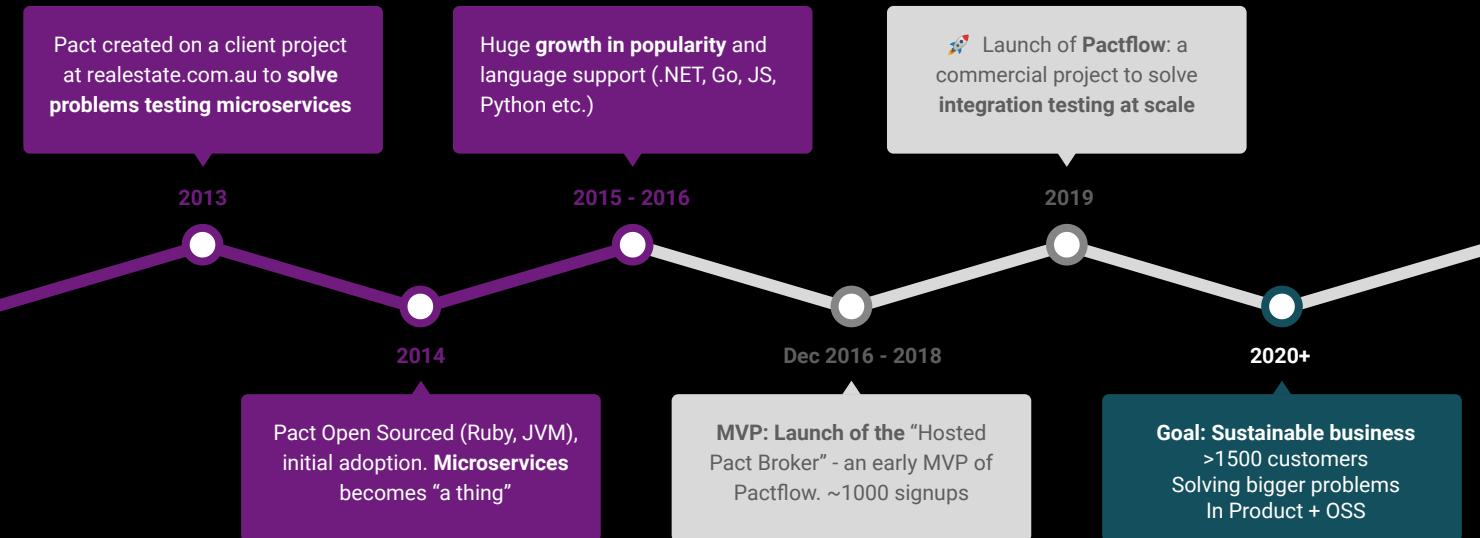
<sup>2</sup> Data from a Capgemini [report](#) on continuous testing in March 2019



In **2013** we created **Pact**, an Open Source tool to solve this problem. In 2019, we launched **Pactflow** to enable organisations to do this *at scale*

# Journey

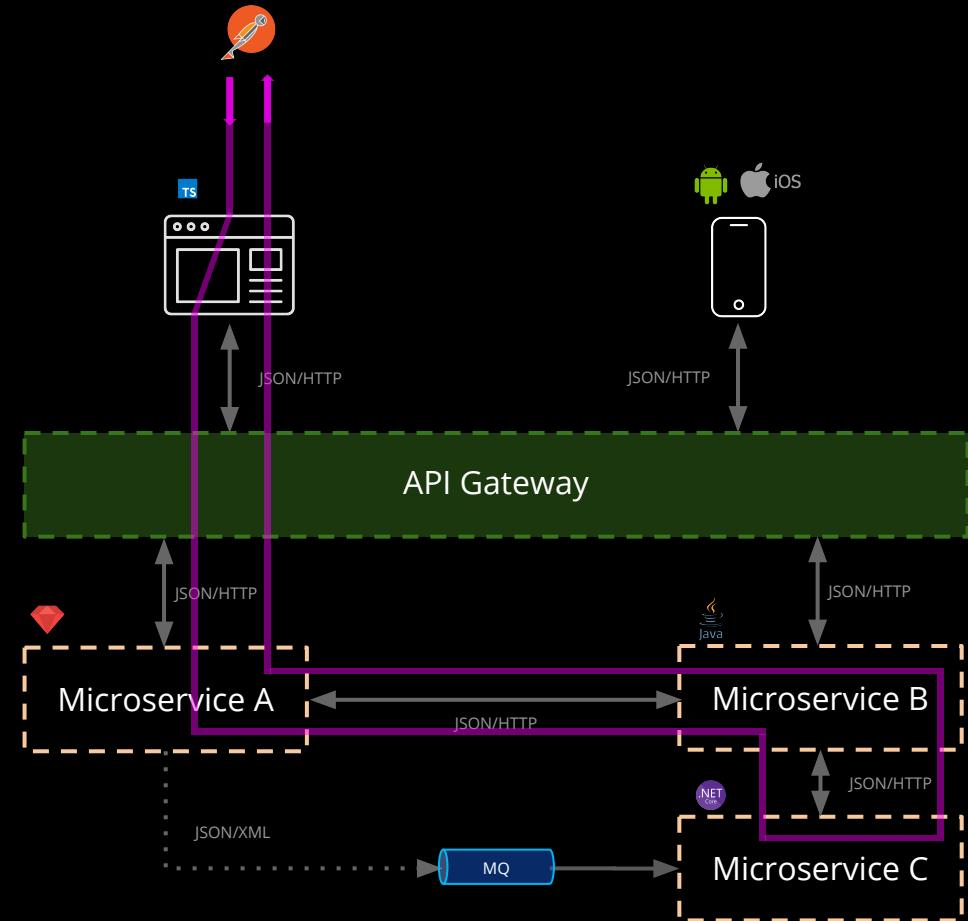
## How we got here



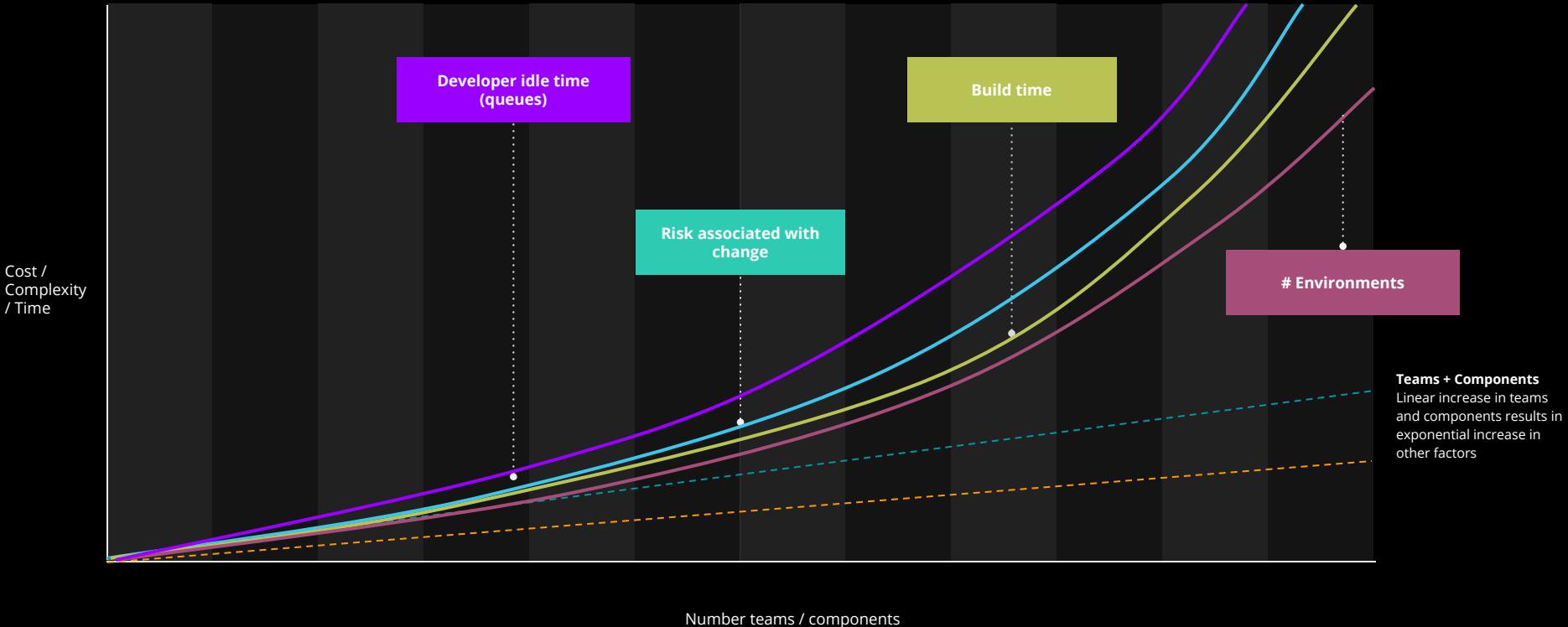
# The old way...

## Why this is hard

- Slow
- Fragile
- Test data management
- Test environment management
- Coverage?
- All-at-once painful deployments
- Teams wait on build queues



# Scaling





Where do we go **wrong?**

# Where we went wrong

## Cause and effect

- Release coupling
- Environment management vs developer idle time
- Slower pipeline leading to batching
- Batching increases failure and defect rates
- Increased Integration tests failures
- Increased deployment risk
- Separate testing teams introduced
- “Release Manager” introduced to govern the entire release process
- Tech debt accumulation

We used the same tools and approaches that worked for monoliths and applied it to distributed systems

“Integration tests are a **scam**”

- JB Rainsberger

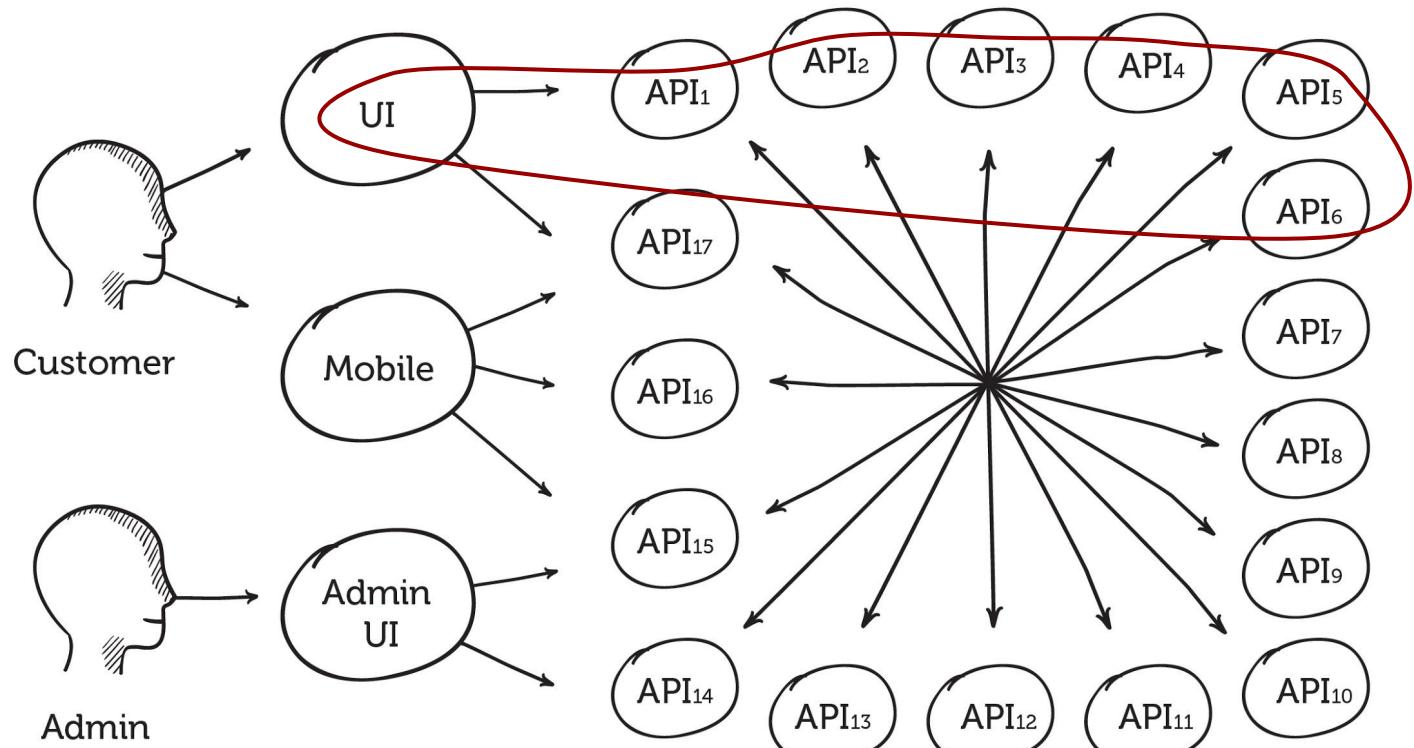
## Scam, you say? Justify!

Integrated tests are:

- Slow
- Fragile
- Hard to manage

When they fail, you can't point to the problem!

## HOW WE TEST MICROSERVICES NOW



## **Branches per box vs test cases required**

2 code branches = 128 tests

5 code branches = 78,125 tests

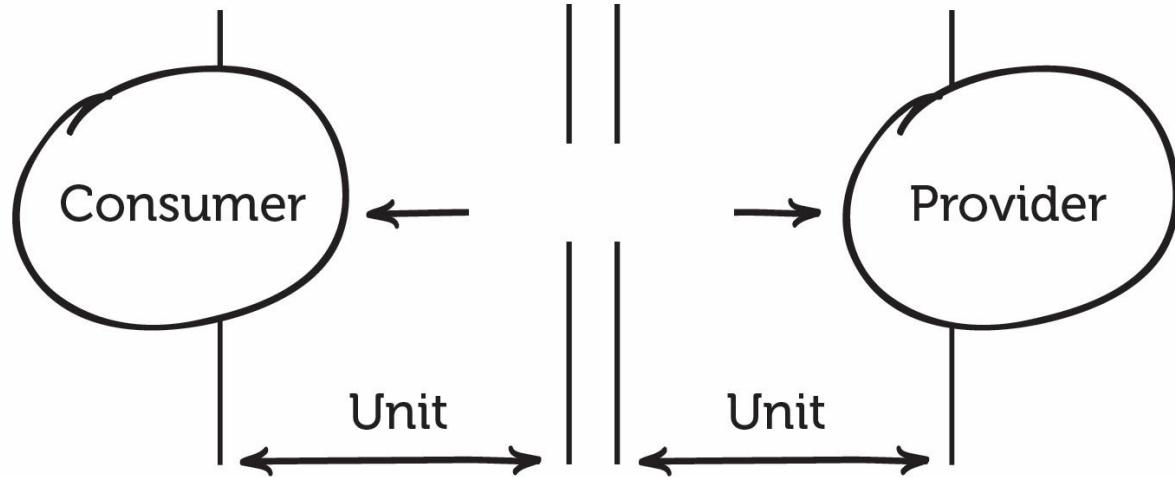
10 code branches = 10M tests

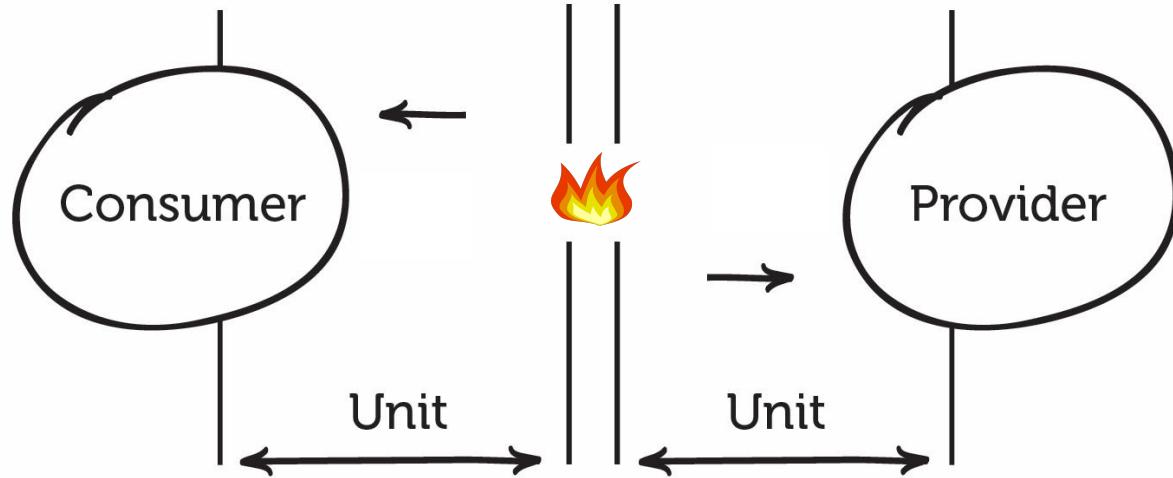
## HOW WE TEST MICROSERVICES NOW



Good tests have the **exact opposite** properties

Mocks to the rescue?





# Mocks

## Solved problems

- Fast feedback
- Few dependencies
- No dedicated environment
- Reliable
- Easy to debug

## New problems

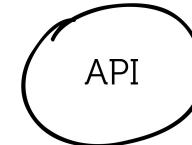
- Hard to keep both sides in sync

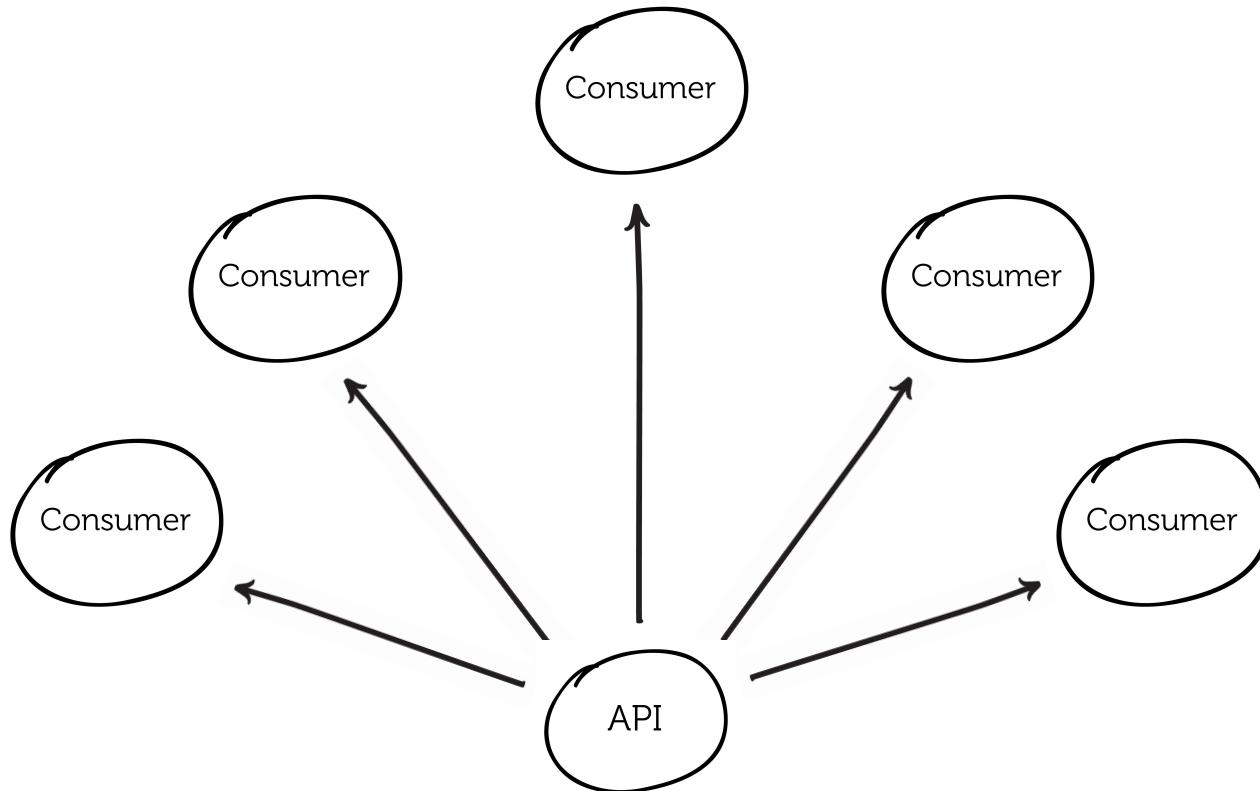
## Dictator Driven Contracts



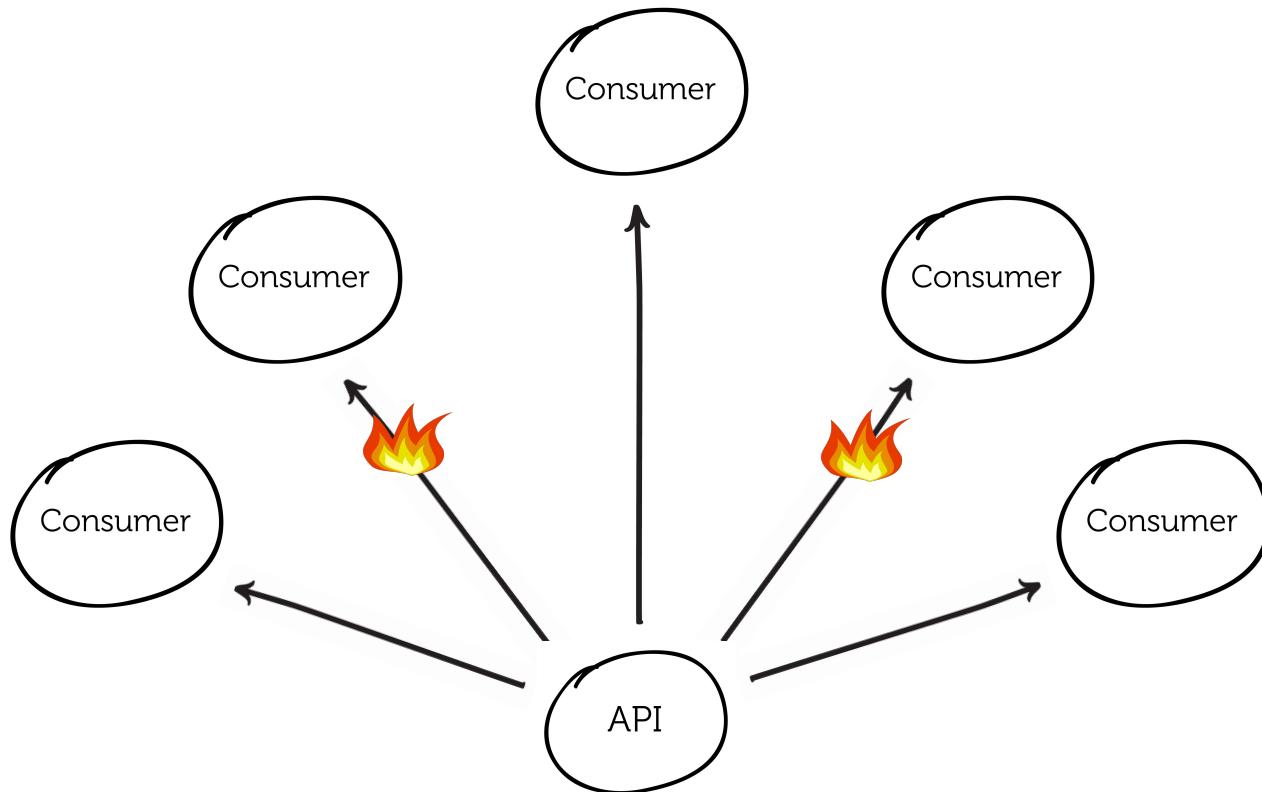
## How to: dictator driven contracts

1. Sit in ivory tower and postulate
2. Document perfect API (Swagger/OAS etc.)
3. Create said API
4. Publish said document to consumers
5. Repeat steps 1-4





## A BETTER WAY TO TEST MICROSERVICES



# Specification first design

## Solved problems

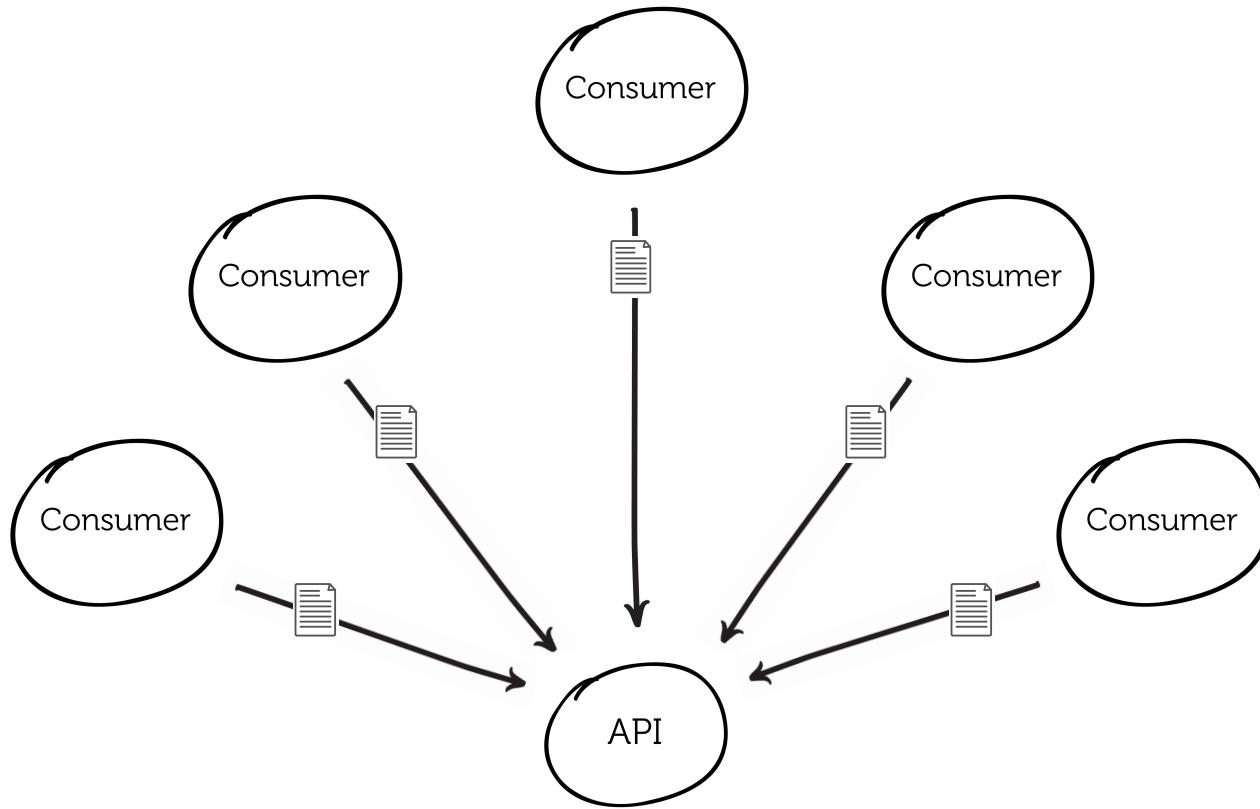
- Good documentation
- Aides discoverability and communication between teams/organisations
- Clearer expectations on API

## New problems

- Who is using my API?
- Requires diligence to ensure backwards compatibility
- Developers hate versioning
- Limited by expressiveness of specification (vague)
- = Hard to get 100% coverage

~~Dictator~~ Consumer Driven Contracts

## A BETTER WAY TO TEST MICROSERVICES



# Consumer Driven Contracts

## Benefits

You know when you **break a consumer**

You get a form of **documentation**

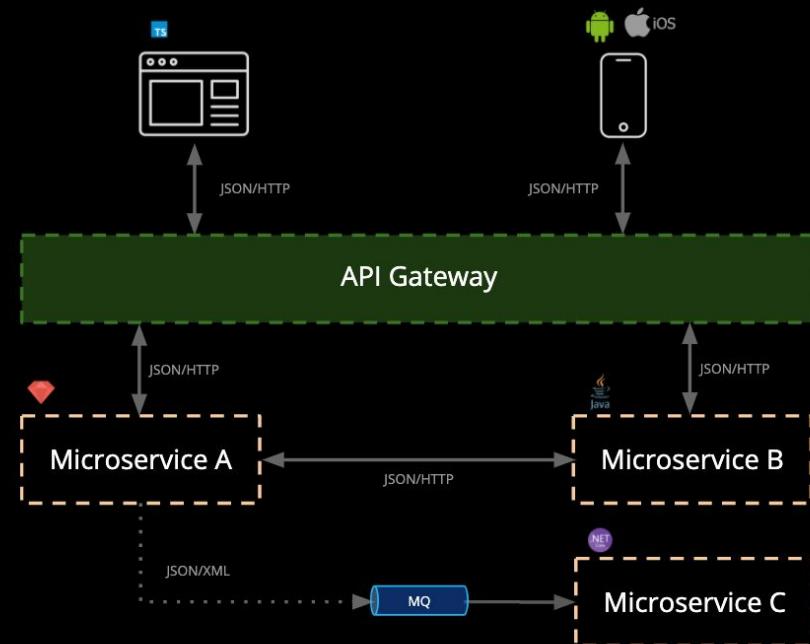
You can test things **independently**

# What is Pact?

Microservice testing made easy

Benefits:

- **Focus** on testing a single integration at a time - without having to deploy
- No **dedicated test environments**
- Get **fast**, reliable feedback
- Tests that scale **linearly**
- **Deploy** services independently



# What is Pact?

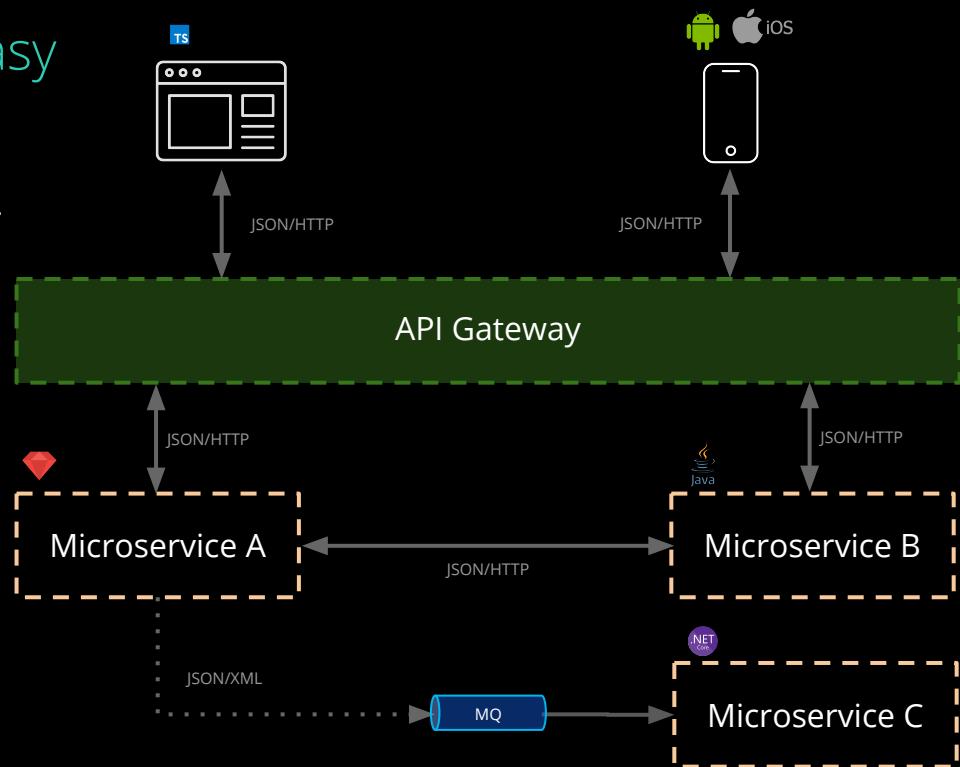
## Microservice testing made easy

Pact is an Open Source tool that makes it easy to test microservices quickly, independently and release safely.

Pact is already used by thousands of companies worldwide.

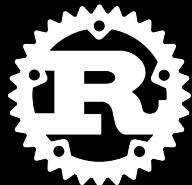
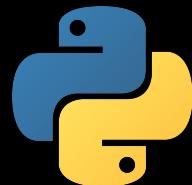
Use cases:

- Javascript web applications (e.g. React)
- Native mobile applications
- RESTful microservices with JSON and XML
- Asynchronous messaging (e.g. MQ)
- Removing end-to-end integrated tests
- Reducing reliance on complex test environments

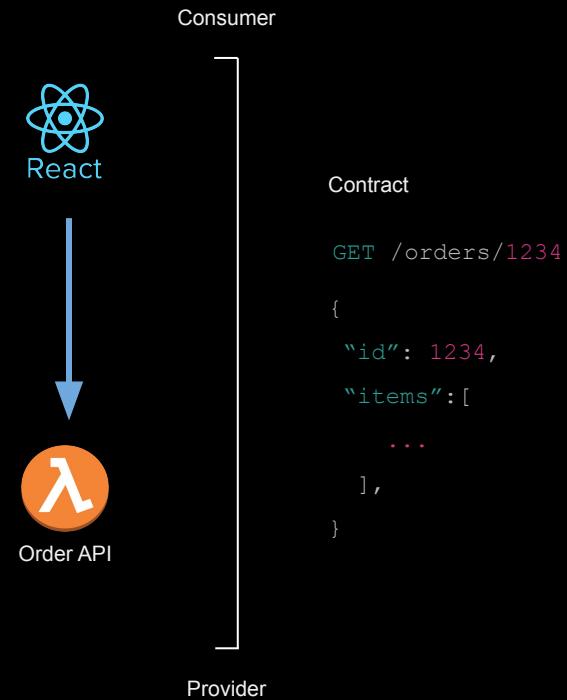


# Open Source

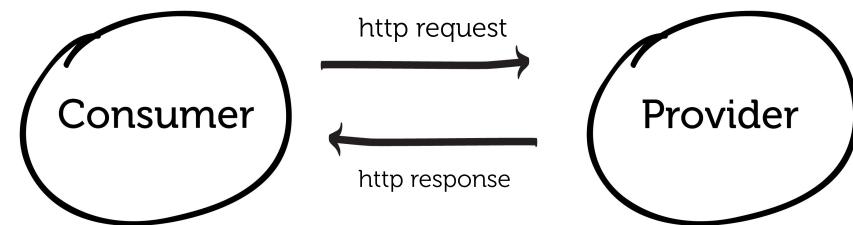
...and in your preferred language



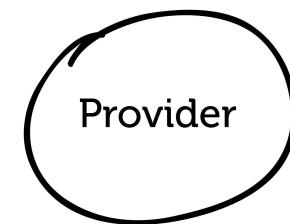
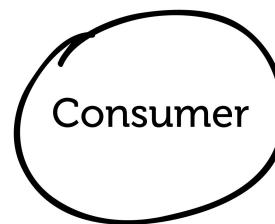
## HOW PACT WORKS

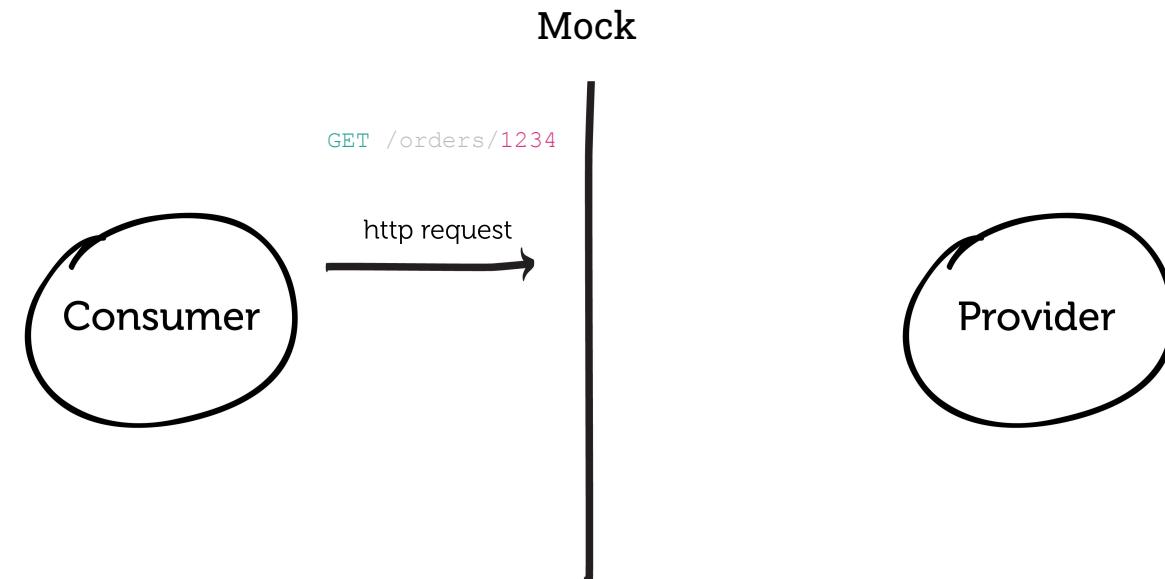


## HOW IT WORKS



Mock

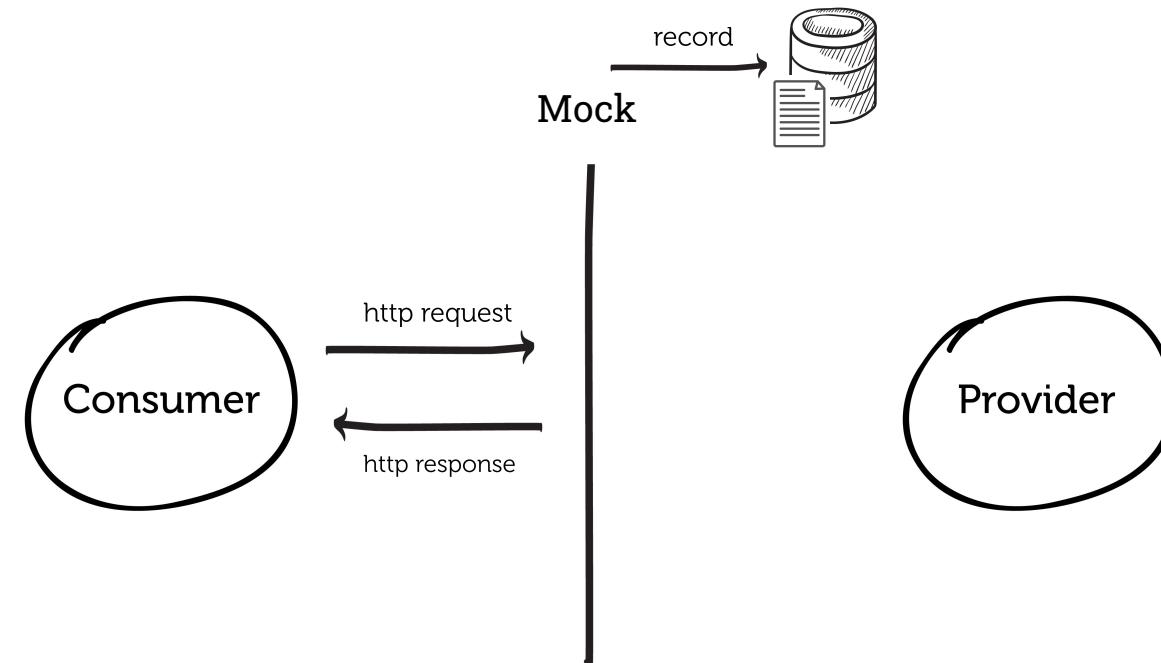




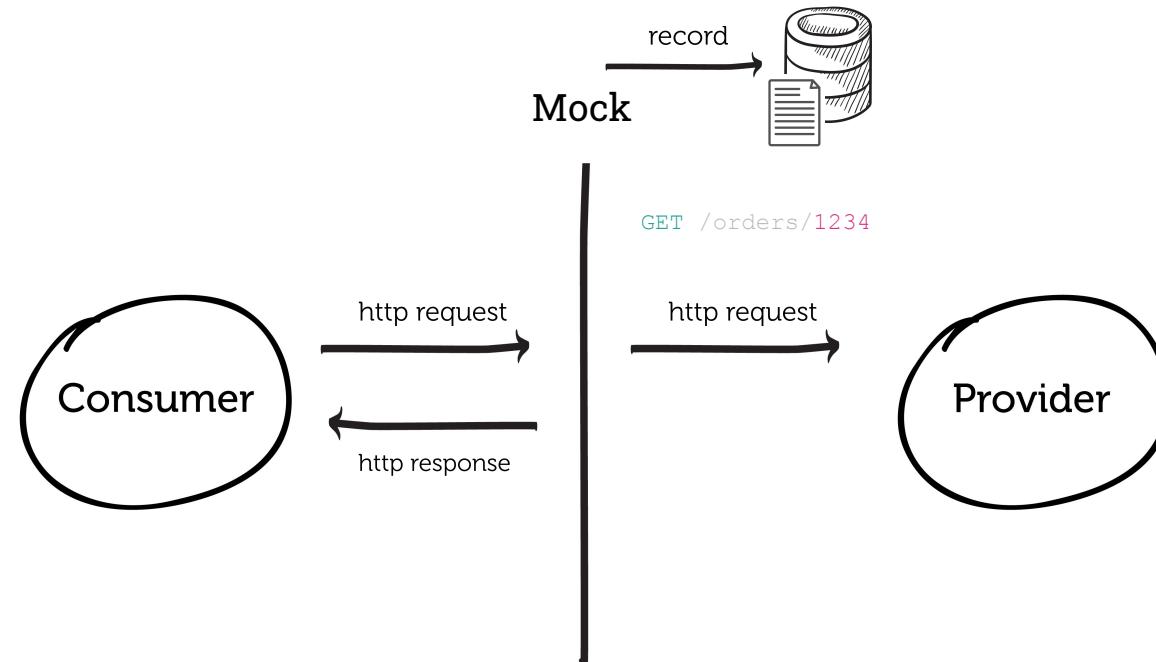
## Mock



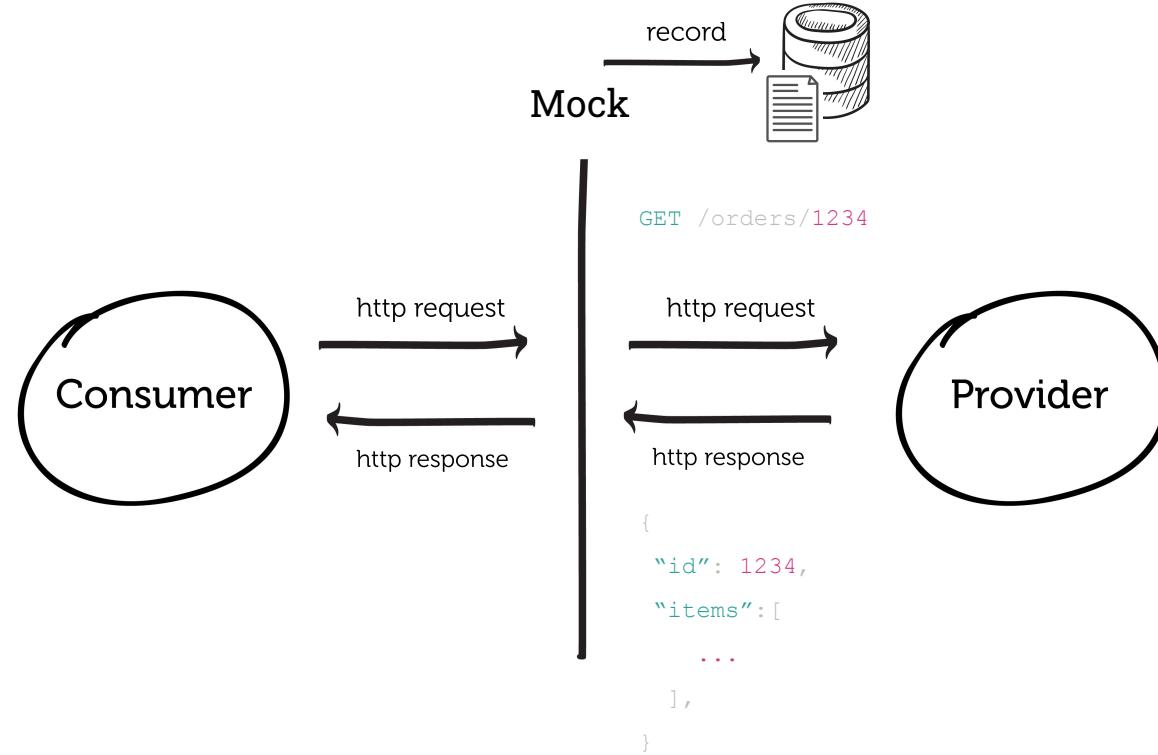
## HOW IT WORKS

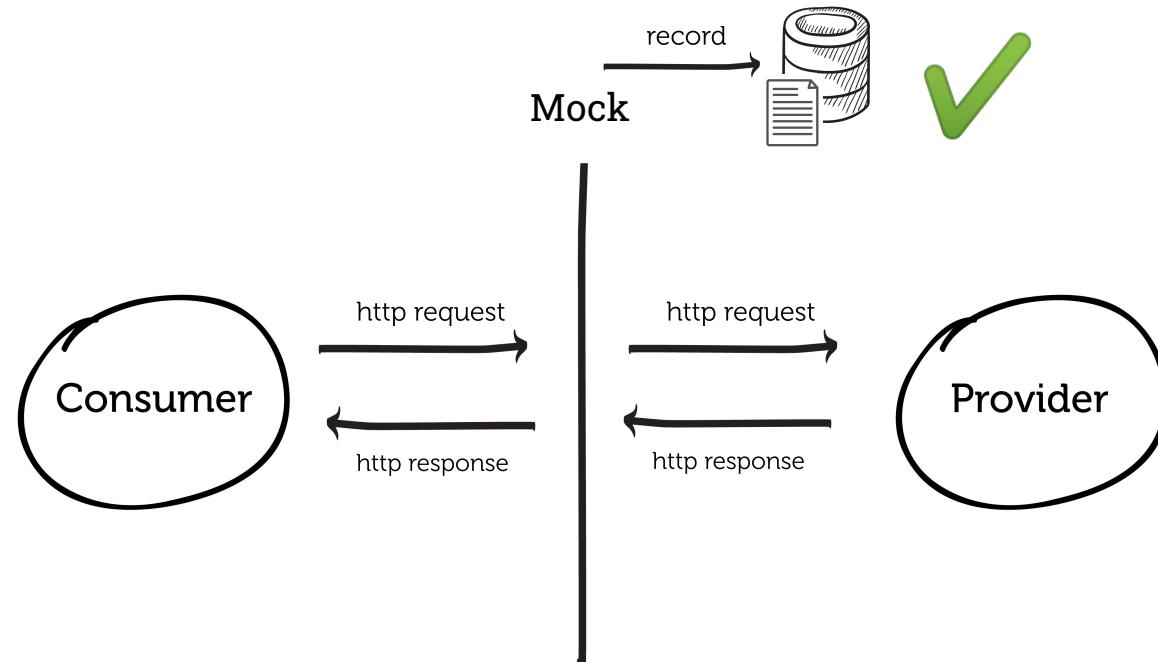


## HOW IT WORKS

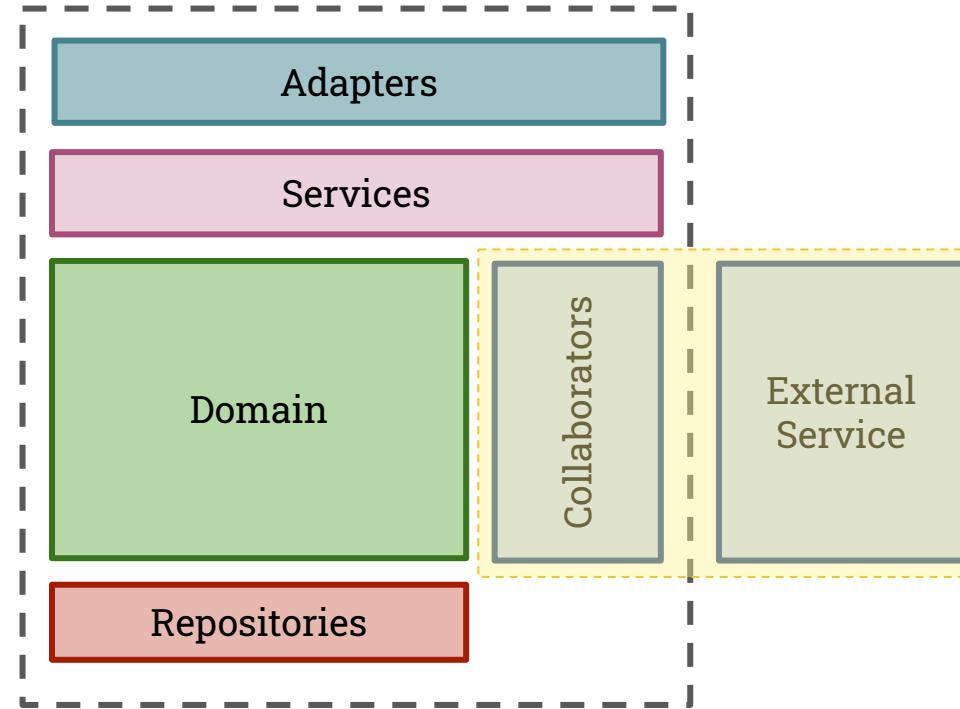


## HOW IT WORKS

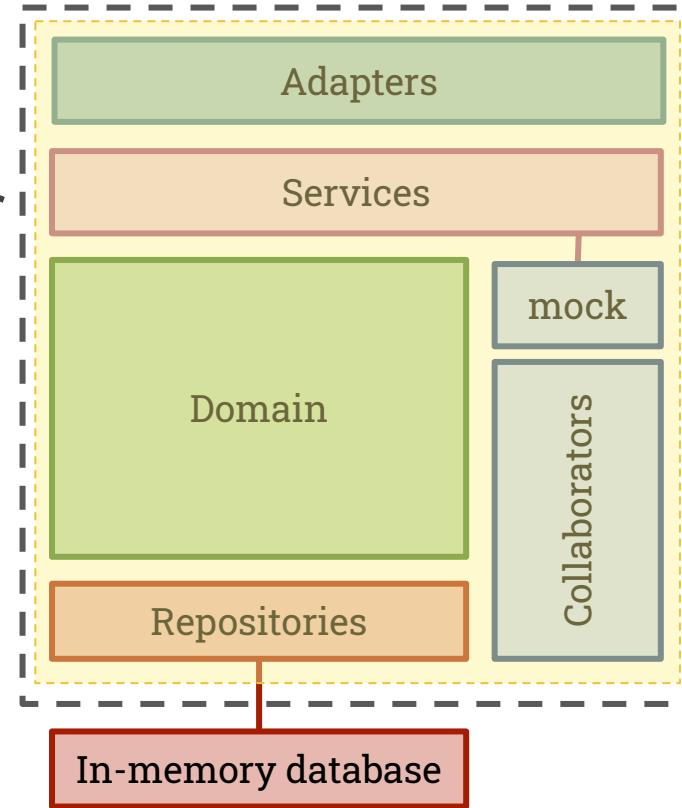




## Scope of consumer test



# Scope of Provider Test



DEMO

DEMO

# Introducing Pactflow

## Pact

1. **Sharing** and **collaborating** on pacts between teams
2. **Lifecycle management**: managing contracts across code branches and environments
3. Orchestrating builds to know when it is **safe to deploy**
4. Integrating into your **processes** and **tooling**

# Pactflow

Contract testing at *scale*

## Additional Capabilities

- Fully managed platform + hardened for scale
- Better user experience
- Secure access management
- Collaboration and insights
- Expansion to other integration technologies\*



Security, Identity + Access Management

Collaboration Tools

Intelligence

Integrations

Protocols

Broker

Insights

Pact + OSS Tooling

Integrated collaboration tool to and share and manage pacts\* between teams.

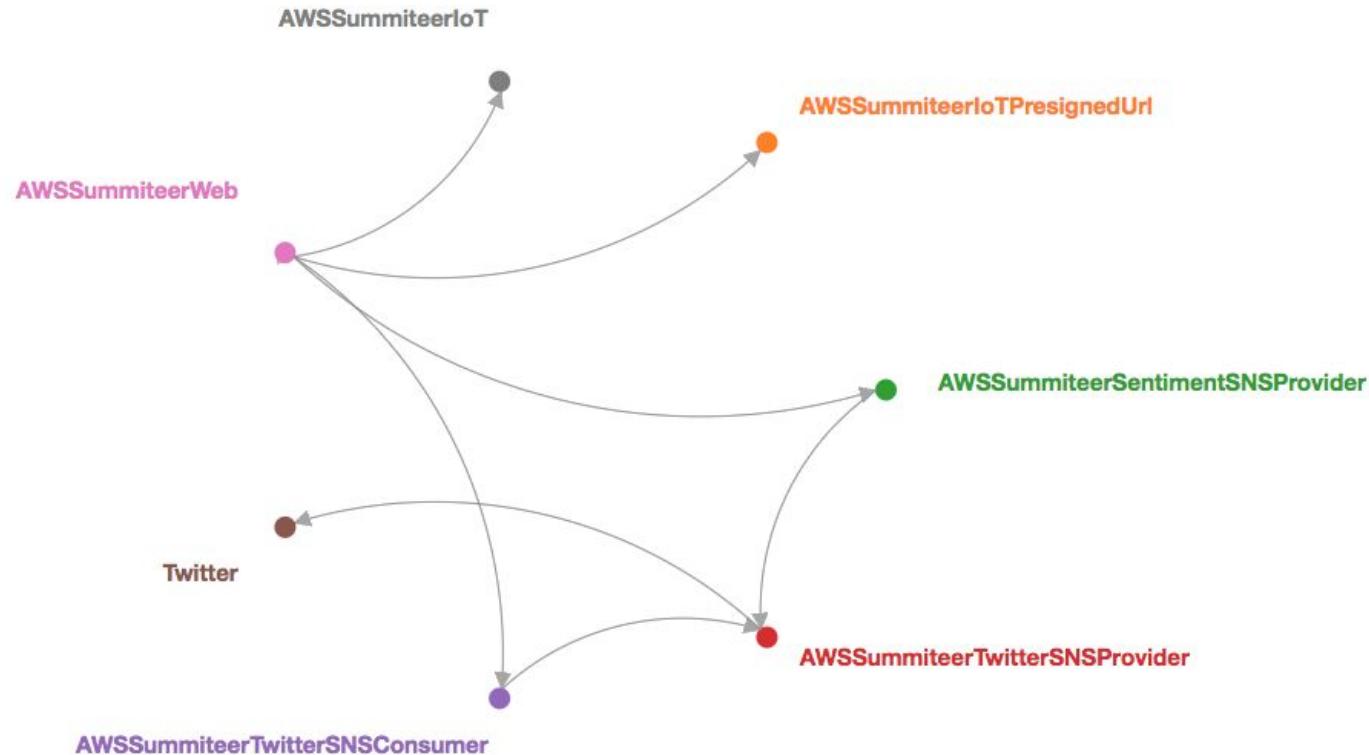
- **API documentation** that is guaranteed to be up-to date
- **Visualisations** of the relationships between your services
- **Dashboards** containing contract verification status
- Pact **tagging** and **versioning**
- **Webhooks** for integration and communication
- View pact **diffs**
- ...and more!

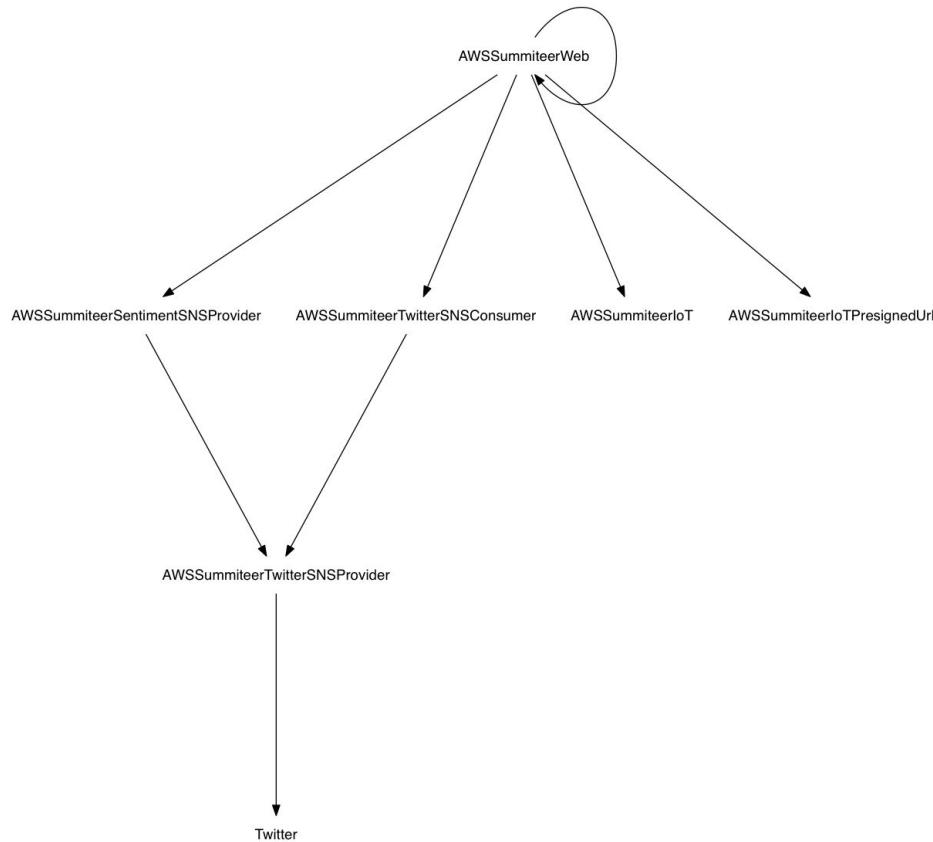
All powered by a RESTful **API**

# Did I mention README badges and integrations?

dummyservice/thedummyprofileservice pact verified

<b>chore: modify pact to test e2e flow</b>  bethesque committed 18 minutes ago ✓	<b>All checks have passed</b> 2 successful checks    Animal Service — Pact tests    continuous-integration/travis-ci/push — The Travi... <span style="float: right;">Details</span>
<b>chore: create webhook for changed pac</b>  bethesque committed an hour ago	
<b>feat: updated gemfile</b>  bethesque committed 2 hours ago ✓	





- Verify that an application is safe to release
- Enables “matrix” style testing

	<b>Consumer Head (1.0.1)</b>	<b>Consumer Prod (1.0.0)</b>
<b>Provider Head (2.0.0)</b>	?	?
<b>Provider Prod (1.1.13)</b>	?	Already tested

Consumer ↓↑	Version ↓↑	Pact Published ↓↑	Provider ↓↑	Version ↓↑	Pact verified ↓↑
A	1	1 day ago	B	1	1 day ago
A	1	1 day ago	B	2	1 day ago
A	1	1 day ago	B	4	1 day ago

```
$ pact-broker can-i-deploy  
  --app A --version 1  
  --app B --version 4
```

✓ 🎉 Can I deploy? npm run can-i-deploy

⌚ Ran in 17s ⌚ Waited 5s 📁 buildkite-l-0954ce6607393c4bc-1

☰ Log Artifacts Agent Environment

+ Expand groups - Collapse groups

>Delete Download Follow

```
1 ▶ Running global environment hook
4 ▶ ↳ Setting up elastic stack environment (v2.3.5)
24 ▶ Downloading secrets from 📁 buildkite-managedsecretsbucket-1eu0tov5vnt6q
55 ▶ Setting up plugins
57 ▶ Preparing build directory
76 ▶ Running global pre-command hook
81 ▶ Running local pre-command hook
93 ▶ Running plugin github.com/buildkite-plugins/docker-compose-buildkite-plugin#v2.0.0 command hook
97 ▶ 🎉 Found a pre-built image for wayfinder
98 ▶ 🎉 Creating docker-compose override file for prebuilt services
103 ▶ 🎉 Pulling services wayfinder
109 ▶ 🎉 Running command in Docker Compose service: wayfinder
111 $ docker-compose -f docker-compose.yml -p buildkite3f89dbc5046b4b72aee0a7480a879d6a -f docker-compose.buildkite-375-override.yml run --name
buildkite3f89dbc5046b4b72aee0a7480a879d6a_wayfinder_build_375 wayfinder npm run can-i-deploy
112 Creating network "buildkite3f89dbc5046b4b72aee0a7480a879d6a_default" with the default driver
113
114 > cd-test-endpoints@1.0.0 can-i-deploy /app
115 > ./scripts/can-i-deploy.sh
116
117
118 -----> Checking if we're safe to deploy!
119     have version: 1.0.0-ab976cc9ccf3a85ec3bec4d808f3cc311306a3a
120     running can-i-deploy
121 Computer says yes \o/
122
123 CONSUMER | C.VERSION | PROVIDER | P.VERSION | SUCCESS?
124 -----
125 CentreDiscoveryWeb | 0.13.0-ebe9050a61d9f49ad16b... | CentreDiscoveryApi | 1.0.0-ab976cc9ccf3a85ec3be... | true
126
127 All verification results are published and successful
128
129
130
131
132 Update available 5.6.0 + 6.2.0
133 Run npm i -g npm to update
134
135
136
137 🎉 Container exited normally
138 🎉 Checking linked containers
```

```
114 > cd-test-endpoints@1.0.0 can-i-deploy /app
115 > ./scripts/can-i-deploy.sh
116
117
118 -----> Checking if we're safe to deploy!
119     have version: 1.0.0-ab976cccd9ccf3a85ec3bec4d808f3cc311306a3a
120     running can-i-deploy
121 Computer says yes \o/
122
123 CONSUMER | C.VERSION | PROVIDER | P.VERSION | SUCCESS?
124 -----
125 CentreDiscoveryWeb | 0.13.0-ebe9050a61d9f49ad16b... | CentreDiscoveryApi | 1.0.0-ab976cccd9ccf3a85ec3be... | true
126
127 All verification results are published and successful
128
129
130
131
132 Update available 5.6.0 → 6.2.0
133 Run npm i -g npm to update
134
135
136
137 ▶ Container exited normally
138 ▶ Checking linked containers
141 ▶ Uploading container logs as artifacts
143 ▶ Cleaning up after docker-compose
151 ▶ Running global pre-exit hook
154 ▶ Stopping ssh-agent 7164
160 ▶ Running local pre-exit hook
```

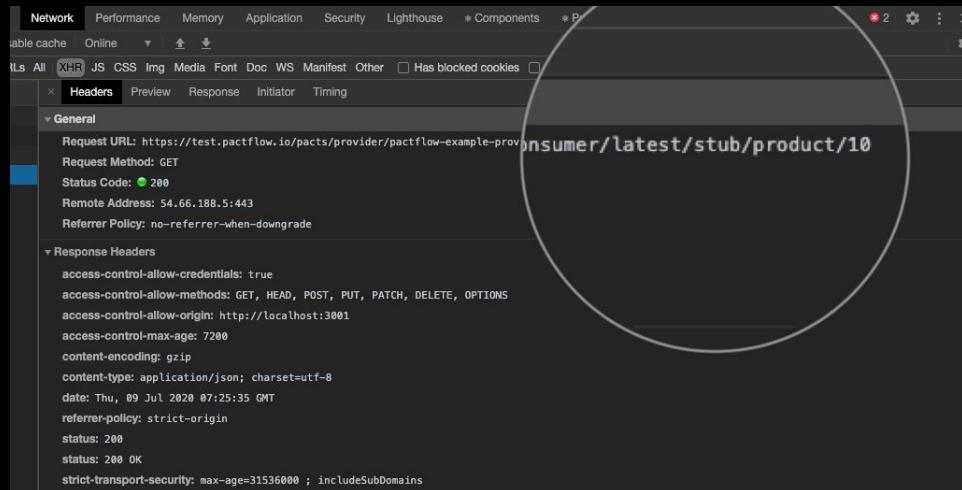
# Features

## Instant API Stubs

Replace fragile test environments with lightning fast + reliable hosted stubs

- **Instant** API backend for all contracts
- Reliable environment for **UI e2e tests**
- **Simplify** local development against multiple backends
- **Discover** and explore other APIs

Find out more at [pactflow.io/blog/hosted-stubs/](https://pactflow.io/blog/hosted-stubs/)



**EDIT WEBHOOK**

All webhook requests are made with HTTP POST.

Description  
POST master.ci.my.domain

Consumer Provider

ALL angular-testing-pyramid

Events \*

- Contract published with changed content or tags
- Contract published
- Verification results published

URL \*  
<https://postman-echo.com/post>

Headers

```
Accept: application/json
```

Body

```
{
  "message": "Triggered by changed pact for ${pactbroker.consumerName} version
${pactbroker.consumerVersionNumber} and ${pactbroker.providerName}",
  "pactUrl": "${pactbroker.pactUrl}"
}
```

Basic auth username

Basic auth password

Enabled

**UPDATE** **TEST**

# Features

## Webhooks

Orchestrate complex build, test and deployment pipelines.

- Trigger a **build** on your CI (such as Travis or Bamboo)
- Publish your commit status to **GitHub**
- Notify your teams via **Slack** of a change to a contract

Find out more at [pactflow.io/blog/webhooks/](https://pactflow.io/blog/webhooks/)

# Features

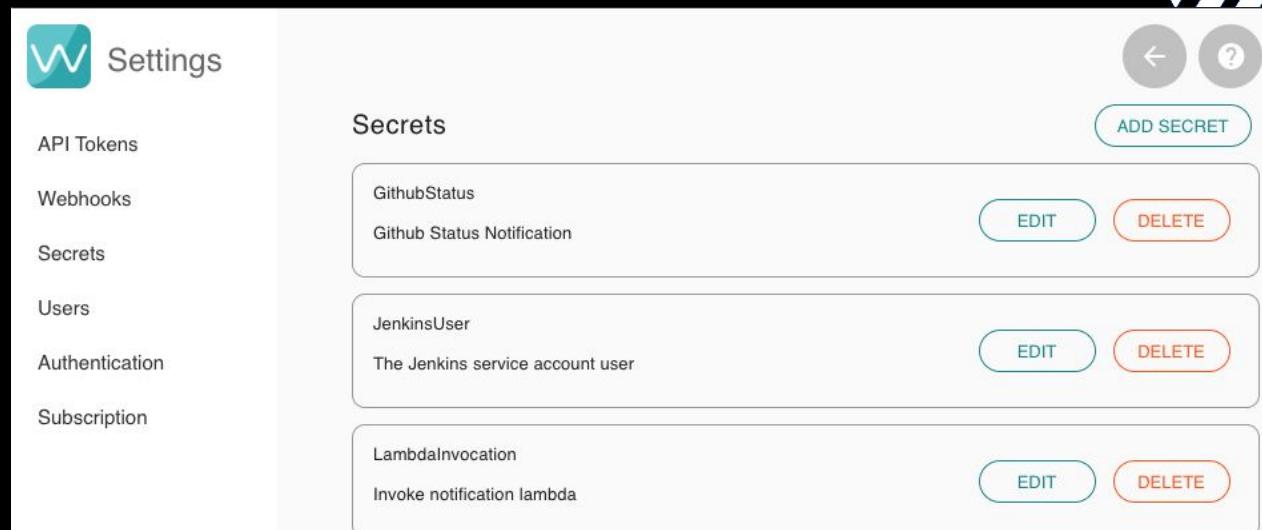
## Secrets

Manage sensitive information with our Secrets Management

All secrets are:

- **Encrypted** with customer specific keys
- **Redacted** in all log files
- Hidden from all users in the UI

*Find out more at*  
[pactflow.io/blog/secrets/](https://pactflow.io/blog/secrets/)



The screenshot shows the Pactflow Settings interface. On the left, a sidebar lists 'API Tokens', 'Webhooks', 'Secrets' (which is selected and highlighted in blue), 'Users', 'Authentication', and 'Subscription'. The main content area is titled 'Secrets' and displays three entries in a table-like format:

Secret Name	Description	Actions
GithubStatus	Github Status Notification	<button>EDIT</button> <button>DELETE</button>
JenkinsUser	The Jenkins service account user	<button>EDIT</button> <button>DELETE</button>
LambdaInvocation	Invoke notification lambda	<button>EDIT</button> <button>DELETE</button>

At the top right of the main content area are three circular icons: a back arrow, a question mark, and a refresh symbol. At the bottom right is a green button labeled 'ADD SECRET'.

The screenshot shows the Pactflow interface with the following details:

- Pactflow Header:** WHAT'S NEW, OVERVIEW, NETWORK DIAGRAM, MATRIX, PACT, Use old UI.
- Status Bar:** Start filtering your pacts, Search icon.
- Left Sidebar (Status Integration):**
  - Matching Service ≈ Animal Profile Service (Status: Failed)
  - Order Web ≈ Order API (Status: Passed)
  - jmarie ≈ loginprovider (Status: Failed)
  - Example App ≈ Example API (Status: Passed)
- Main Content Area:**

### A pact between Matching Service and Animal Profile Service

Matching Service (consumer) version: 1.0.1569369168 prod  
Animal Profile Service (provider) version: 1.0.0 test  
Pact publication date: a month ago  
Verification failed: a month ago  
Pact specification version: 2.0.0

**Mismatches (3):**

  - A request for all animals given has some animals
  - Body:
    - Could not find key "first\_name" (keys present are: last\_name, animal, age, available\_from, gender, location, eligibility, interests, id) at \$[0]
    - Could not find key "first\_name" (keys present are: animal, last\_name, age, available\_from, gender, location, eligibility, interests, id) at \$[1]
    - Could not find key "first\_name" (keys present are: last\_name, animal, age, available\_from, gender, location, eligibility, interests, id) at \$[2]

**Request:**

```
Method:GET
Path:/animals/available
Headers:
{
  "Authorization": "Bearer token"
}
```

**Expected response:**

```
Status:200
Headers:
{
  "Content-Type": "application/json; charset=utf-8"
}
Body:
```

# Features

## Verification Results

Understand build failures with detailed error reporting via Verifications:

- **Give visibility** to consumer teams and **reduce time-to-diagnosis**
- **Breakdown** of successful and failed interactions
- **Understand** which field, header or status code was the cause of the problem

Find out more at  
[pactflow.io/blog/verification-results/](https://pactflow.io/blog/verification-results/)

# Features

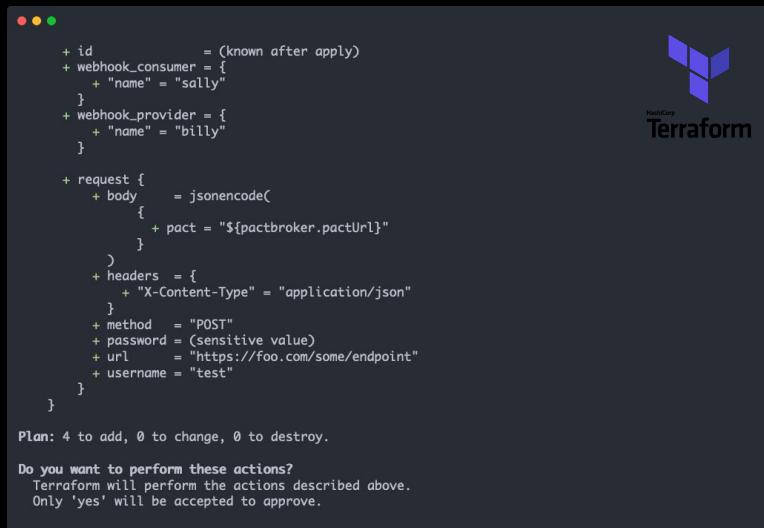
## Infra-as-Code

Automate your Pactflow configuration  
with Terraform:

- Participants
- Webhooks
- Secrets
- API Tokens

*Find out more at*

<https://pactflow.io/blog/terraform/>



A screenshot of a Mac OS X terminal window showing Terraform configuration code. The code defines participants (Sally and Billy), their webhook consumer details, and a request configuration including headers, method, password, URL, and username. A 'Plan' command shows 4 resources to add. A confirmation message asks if the user wants to proceed.

```
+ id          = (known after apply)
+ webhook_consumer = [
+   "name" = "sally"
]
+ webhook_provider = [
+   "name" = "billy"
]

+ request {
+   body      = jsonencode(
+     {
+       pact = "${pactbroker.pactUrl}"
+     }
+   )
+   headers  = [
+     "X-Content-Type" = "application/json"
+   ]
+   method   = "POST"
+   password = (sensitive value)
+   url      = "https://foo.com/some/endpoint"
+   username = "test"
}
}

Plan: 4 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.
```

# Features

Login to Pactflow

Login with your email address and password

Email: \*

Password: \*

Forgot your password? [SUBMIT](#)

OR LOGIN WITH

GOOGLE

GITHUB

## Social Login, SSO and SAML 2.0

Choose how you want to authenticate and manage your users:

- **Pactflow's in built user database**
- **Github authentication**<sup>1</sup>
- **Google OpenID connect**<sup>1</sup>
- **SAML**<sup>2</sup>

*Find out more at  
[pactflow.io/blog/saml-and-federated-authentication/](https://pactflow.io/blog/saml-and-federated-authentication/)*

<sup>1</sup> Available on Team or Business plans

<sup>2</sup> Available only on Business plans

```
{
  "events": [
    {
      "uuid": "16WlpdLpDMzfMYxLTZYXYw",
      "timestamp": "2019-12-10T09:15:24.864+11:00",
      "type": "SaasBroker::Api::Resources::RegenerateApiToken",
      "db_user_id": 2,
      "user_email": "someuser@somecompany.com",
      "payload": {
        "path": "/settings/tokens/_UjhYvyyEjM9L2SgWd0qsw/regenerate",
        "queryString": "",
        "method": "POST",
        "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36",
        "referer": "http://somebroker.pact.di.us.com.au/settings/api-tokens",
        "params": {
          "resource_name": "regenerate_token",
          "token_uuid": "_UjhYvyyEjM9L2SgWd0qsw"
        }
      }
    },
    ...
  ],
  "_links": {
    "self": {
      "href": "http://somebroker.pact.di.us.com.au/audit?from=zH00xNcjUseyU6Dsiaadxw"
    },
    "next": {
      "href": "http://somebroker.pact.di.us.com.au/audit?from=I1ECUSbMLJL0y8gFbLLuIg"
    }
  }
}
```

# Features

## Audit Log

Integrate Pactflow into your SOC:

- **Immutable** audit log available via API
- Full **traceability** of access and system usage, including references to IdP identities

Available to Business Plans

*Find out more at*

[pactflow.io/blog/audit-api/](https://pactflow.io/blog/audit-api/)

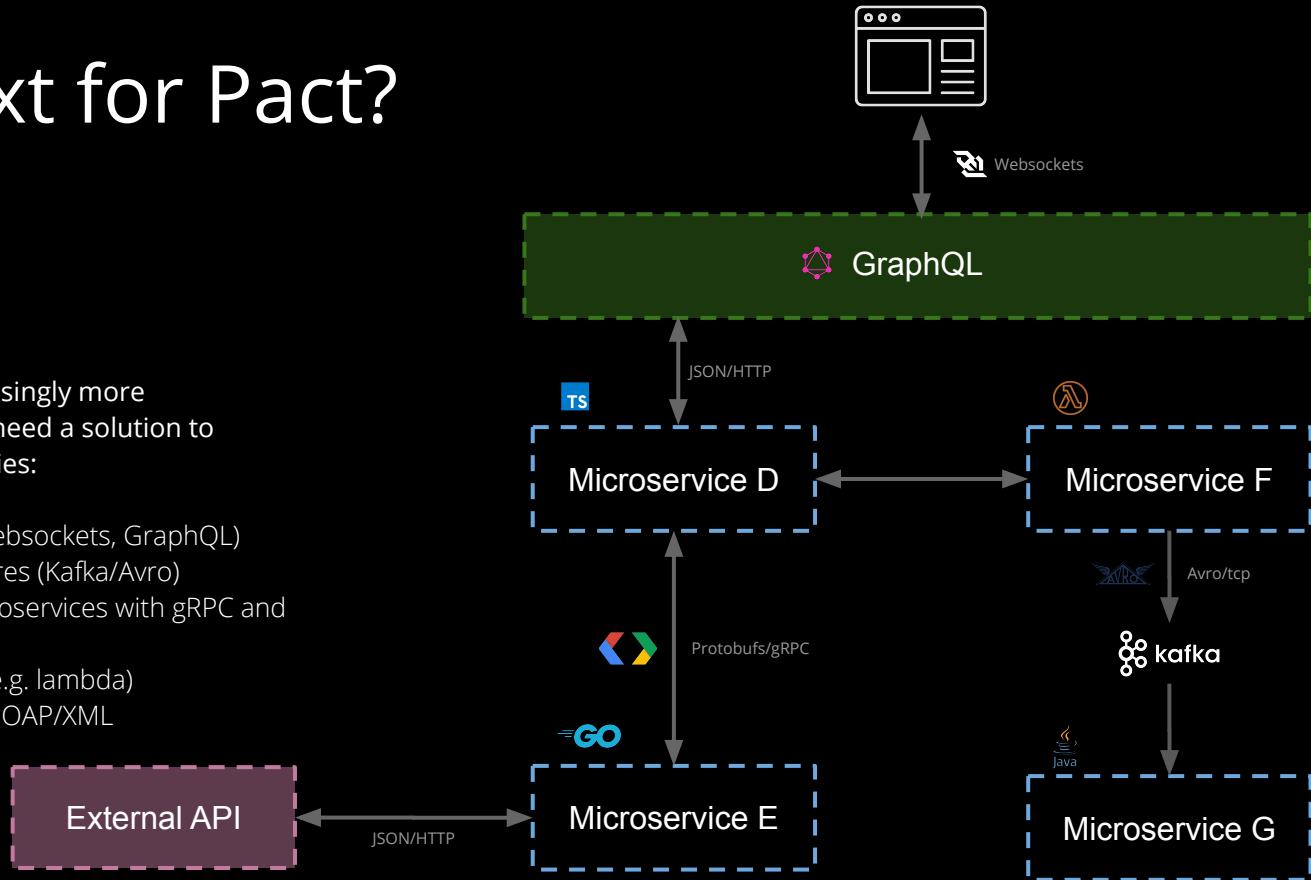
# What's next for Pact?

## Modern APIs

### New Capabilities

Distributed systems are increasingly more complicated, and enterprises need a solution to polyglot integration technologies:

- Real-time web apps (Websockets, GraphQL)
- Event-based architectures (Kafka/Avro)
- High-performance microservices with gRPC and Protobufs
- Function-as-a-Service (e.g. lambda)
- Legacy SOA apps with SOAP/XML
- External APIs



# Team

Email: [hello@pactflow.io](mailto:hello@pactflow.io)

Web: [pactflow.io](https://pactflow.io)

Twitter: [@pactflow](https://twitter.com/pactflow)



**Matt Fellows - Founder + JS/Go/Rust Maintainer**

@matthewfellows

mfellows@dius.com.au



**Beth Skurrie - Founder + Ruby Maintainer/Pact Broker Creator**

@bethesque

bskurrie@dius.com.au



**Ron Holshausen - Founder + Pact JVM/Rust Maintainer**

@uglyog

rholshausen@dius.com.au