



PACTFLOW

Brought to you by **DIUS** 

Agenda

Plan for today

1. Problem
2. How Pact works
3. Demo
4. Q & A

Integration testing is *hard*...

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

PACTFLOW

Brought to you by DIUS 

The numbers

Four key indicators of high performing organisations¹



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

¹ Data from the DORA 2019 State of DevOps [report](#)

The numbers

Challenges facing the market

Only **20%** of companies are “elite” performers¹

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

36% of teams are impacted by wait times and cost of **test environments**²

76% spent one third of their time or more managing **test data**²

¹ Data from the DORA 2019 State of DevOps [report](#)

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

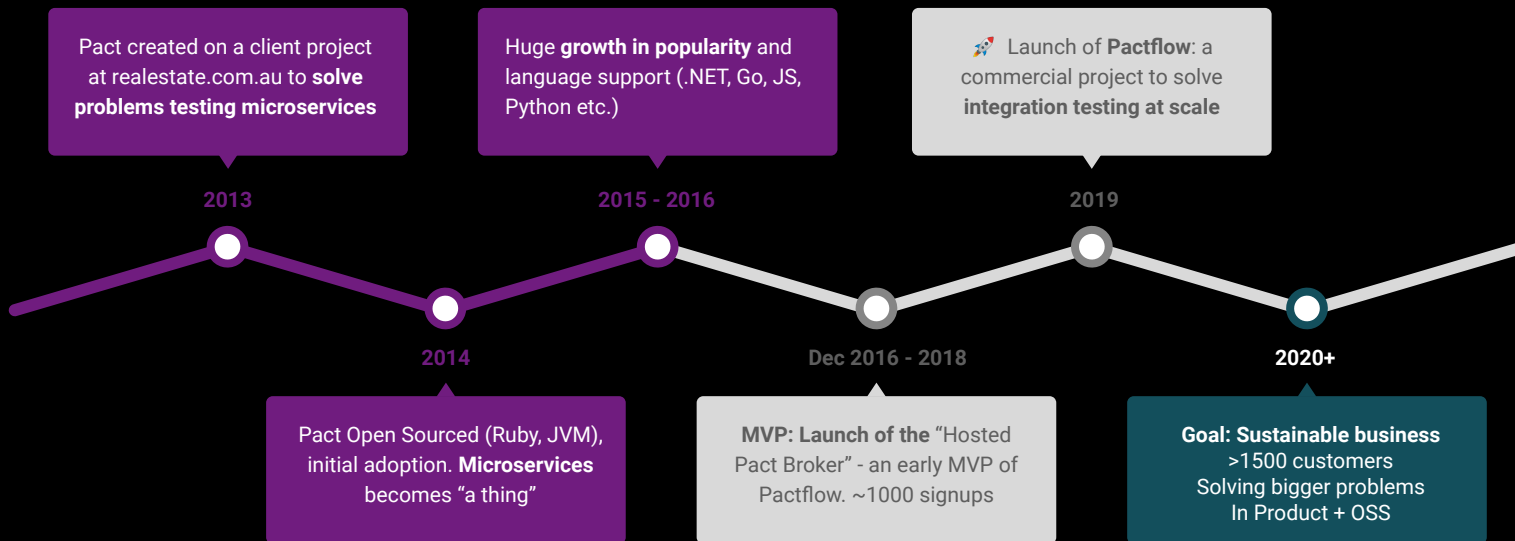
In 2013 **Pact** (an Open Source tool) was created to solve this problem. In 2019, we launched **Pactflow** to enable organisations to do this *at scale*.

PACTFLOW

Brought to you by DIUS 

Journey

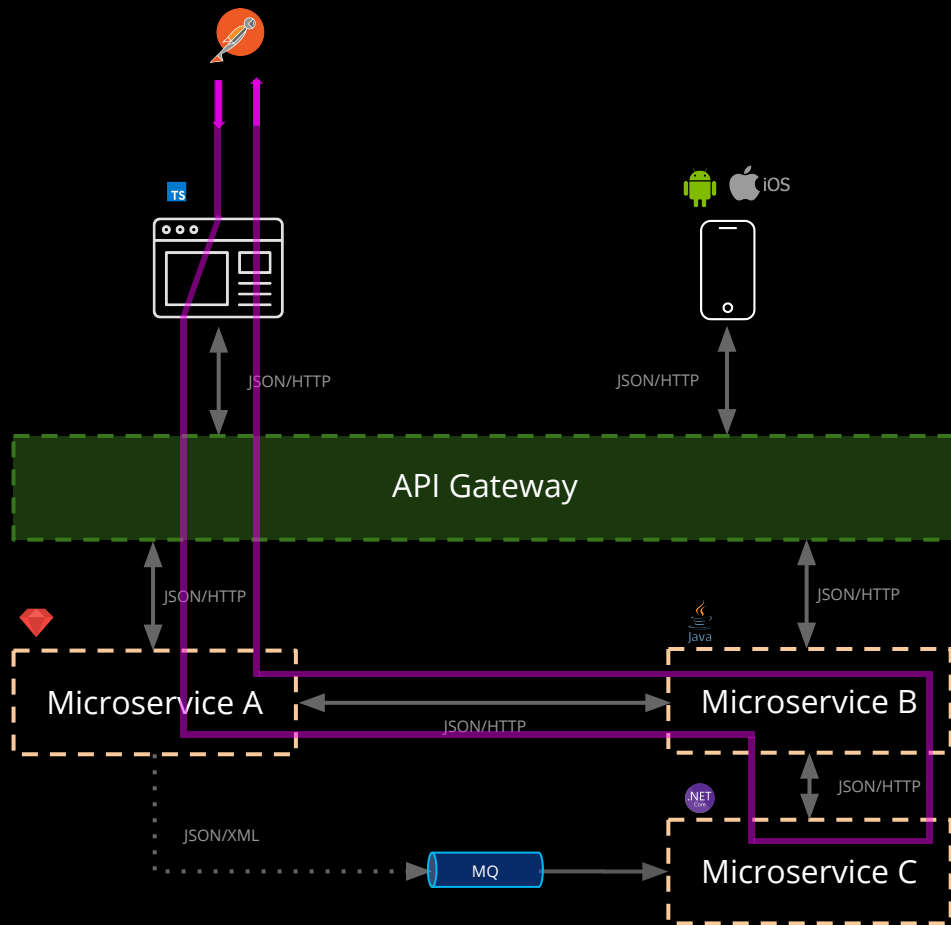
How we got here



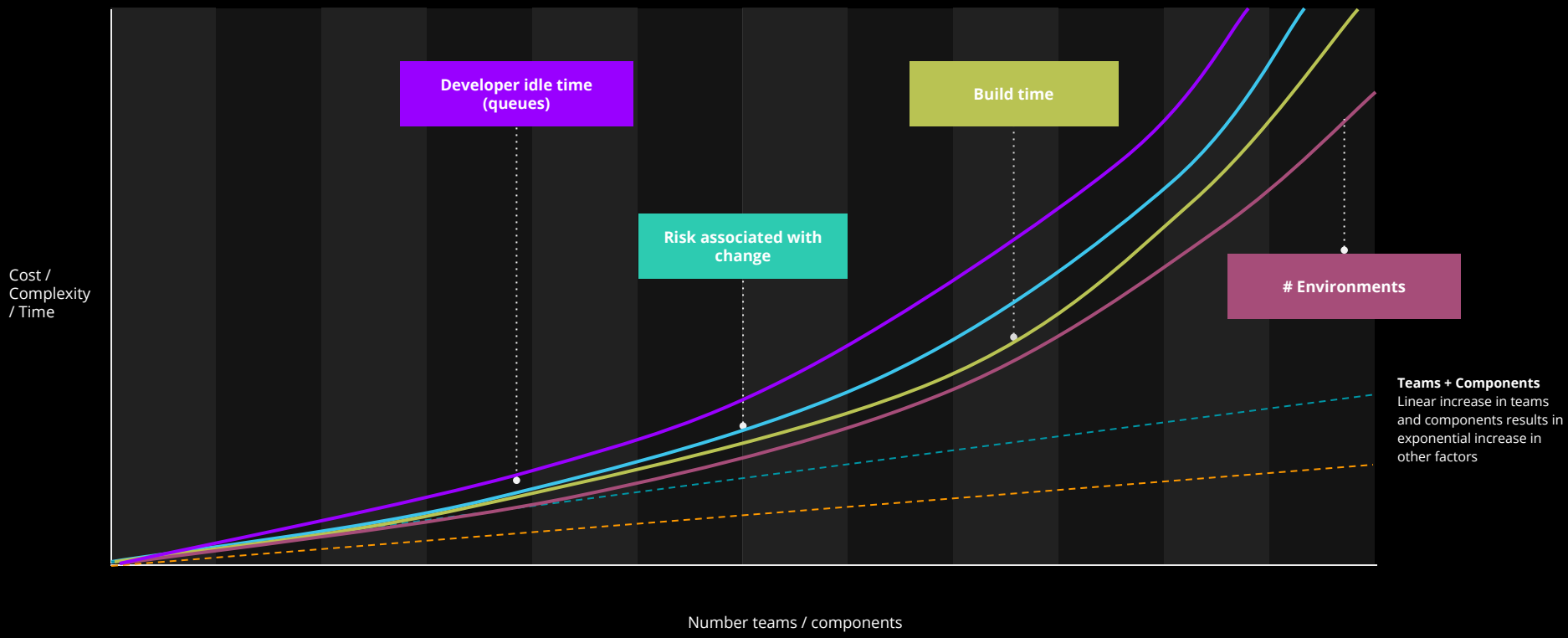
The old way...

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 we went wrong

Cause and effect

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

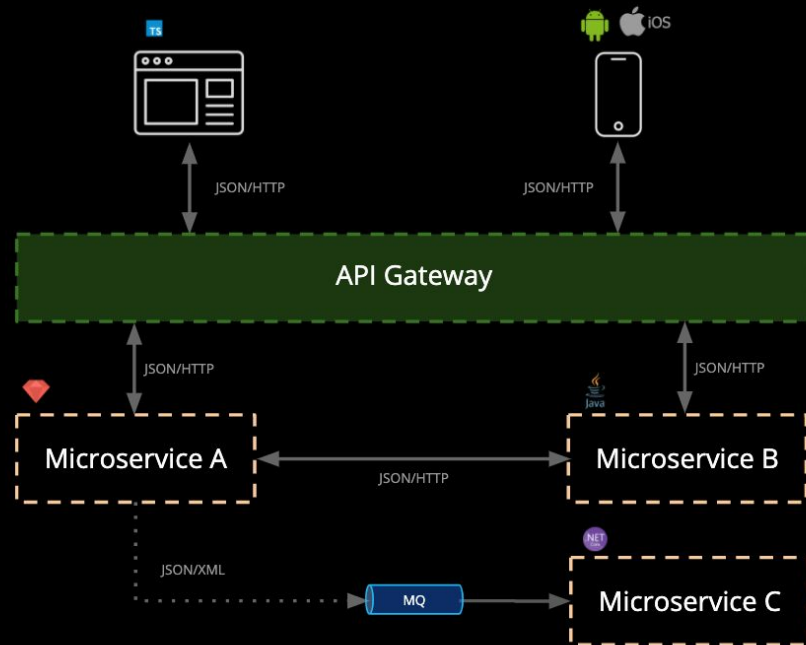
Approach for monoliths does not work for distributed systems

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?

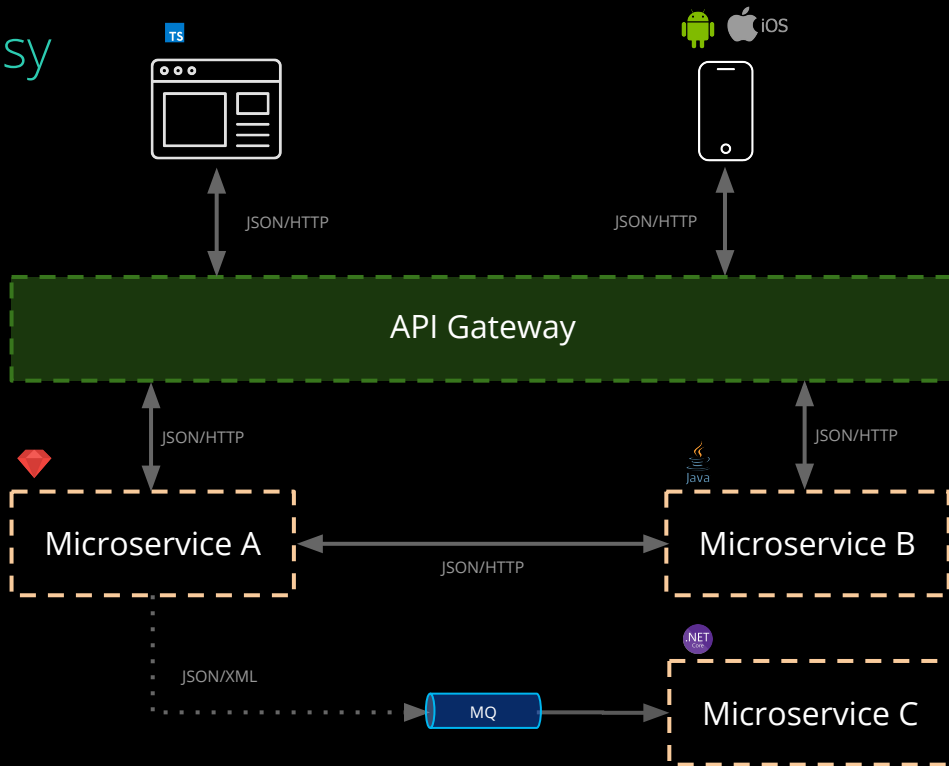
Microservice testing made easy

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)

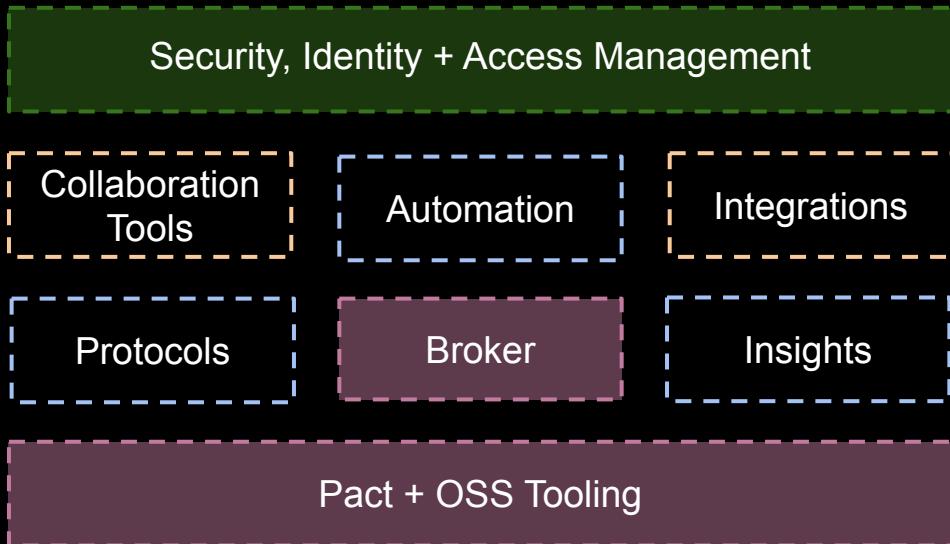


Pactflow

Contract-testing at *scale*

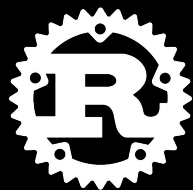
Pactflow's microservices continuous delivery platform

- Collaborate with Pact and Spring Cloud Contract across many teams
- Fully managed and hardened for scale
- Gain insights into API integration issues quickly
- Integrates with your tools + workflow
- Orchestrate complex build, test and deployment pipelines
- ...and more



Open Source

...and in your preferred language



The numbers, revisited

How Pactflow supports high performing organisations



Need < 1 day **lead time** for changes



Pact's "consumer driven" contracts enable APIs to evolve quickly and safely



Are able to **deploy** on demand



Deploy services independently and confidently using Pact's `can-i-deploy` tool



Have **change failures** rates < 15%



Pact tests are focussed, scale linearly and cover more of your API in less time



Can **restore services** within 1 hour



Roll forward, not back - Pact tests run fast and don't need E2E environments

PACTFLOW

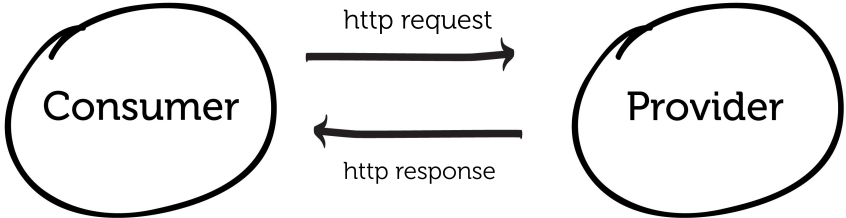
Brought to you by DIUS 



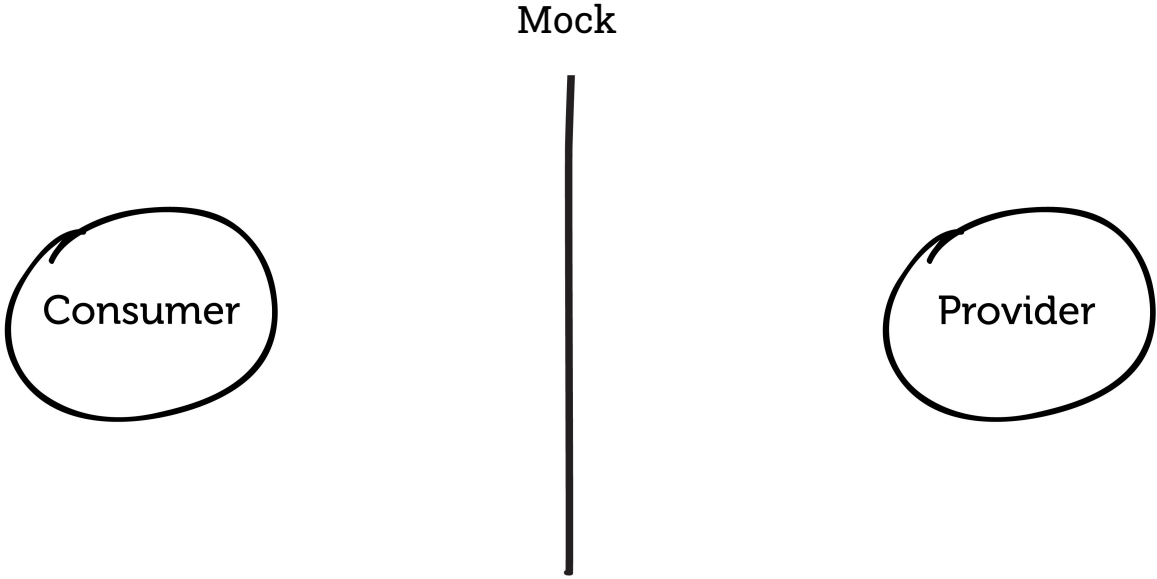
HOW PACT WORKS

PACTFLOW

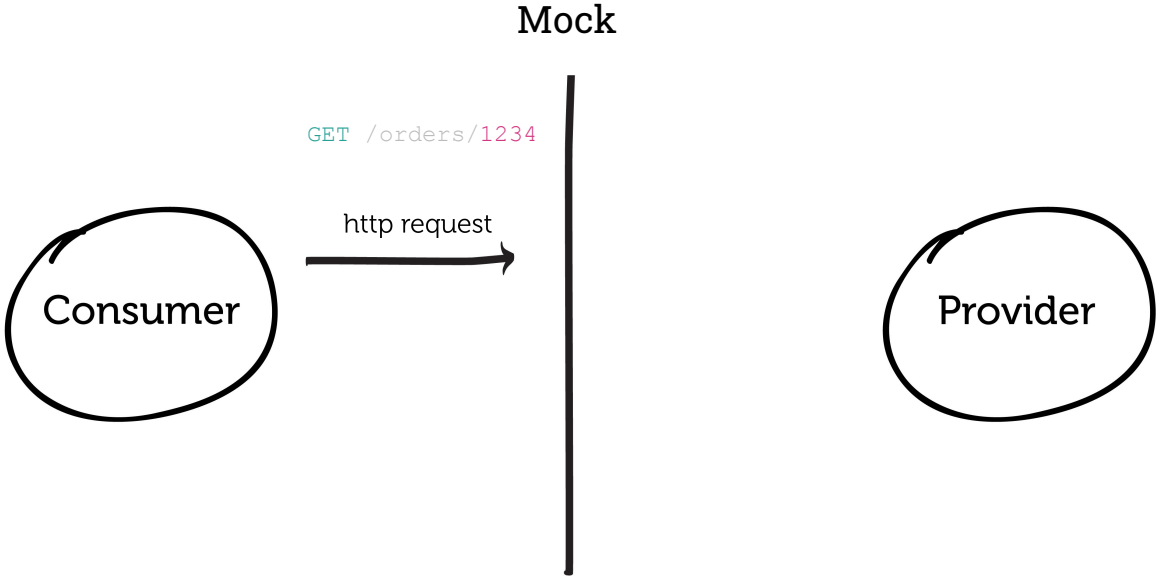
Brought to you by **DIUS** 



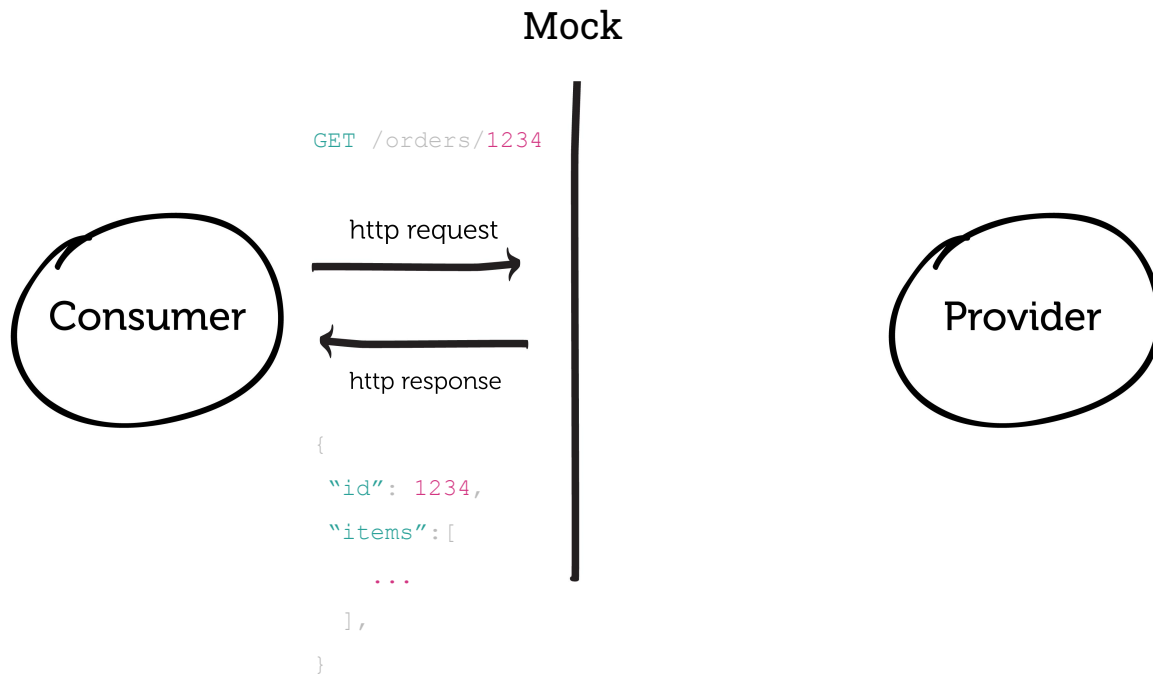
Step 1: test the consumer



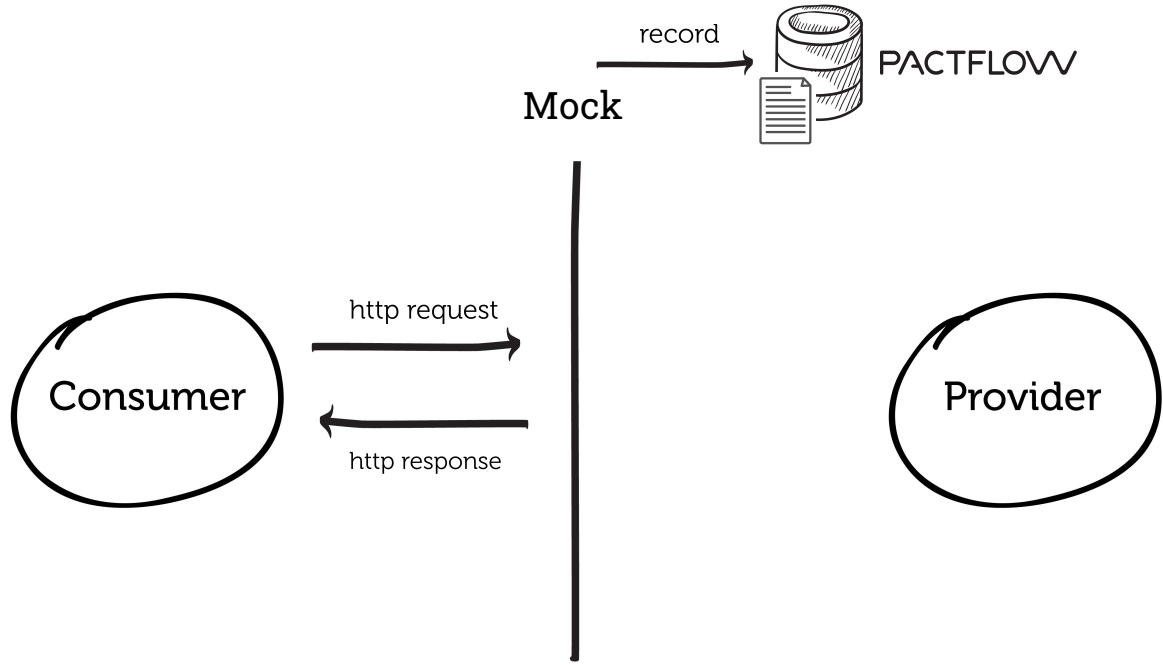
Step 1: test the consumer



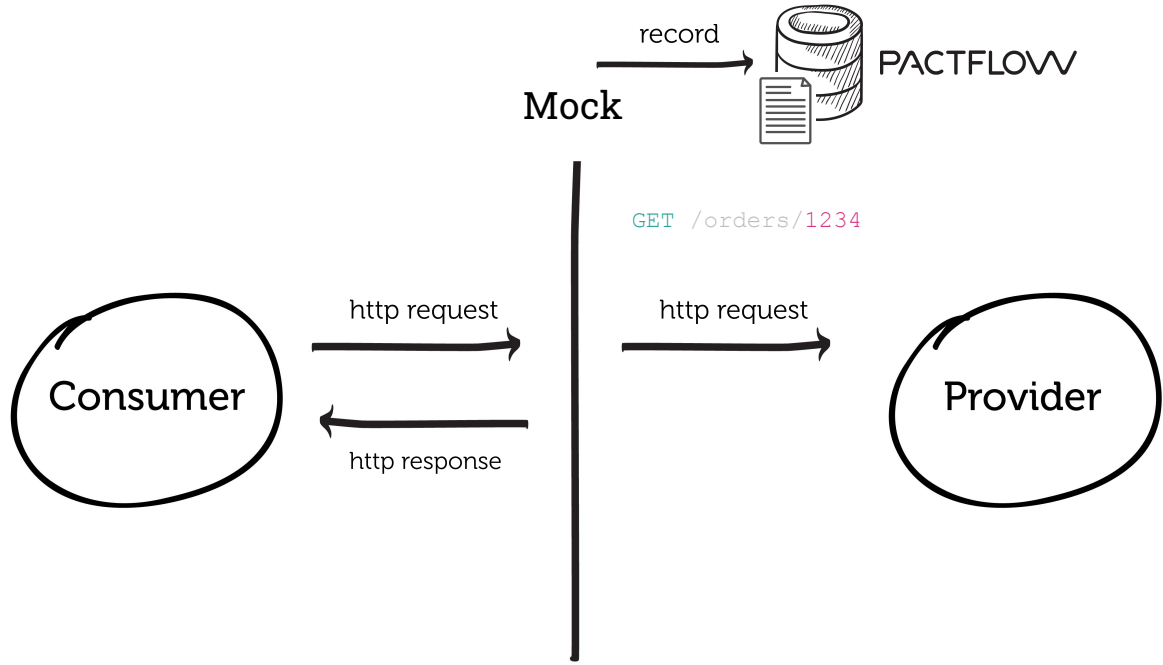
Step 1: test the consumer



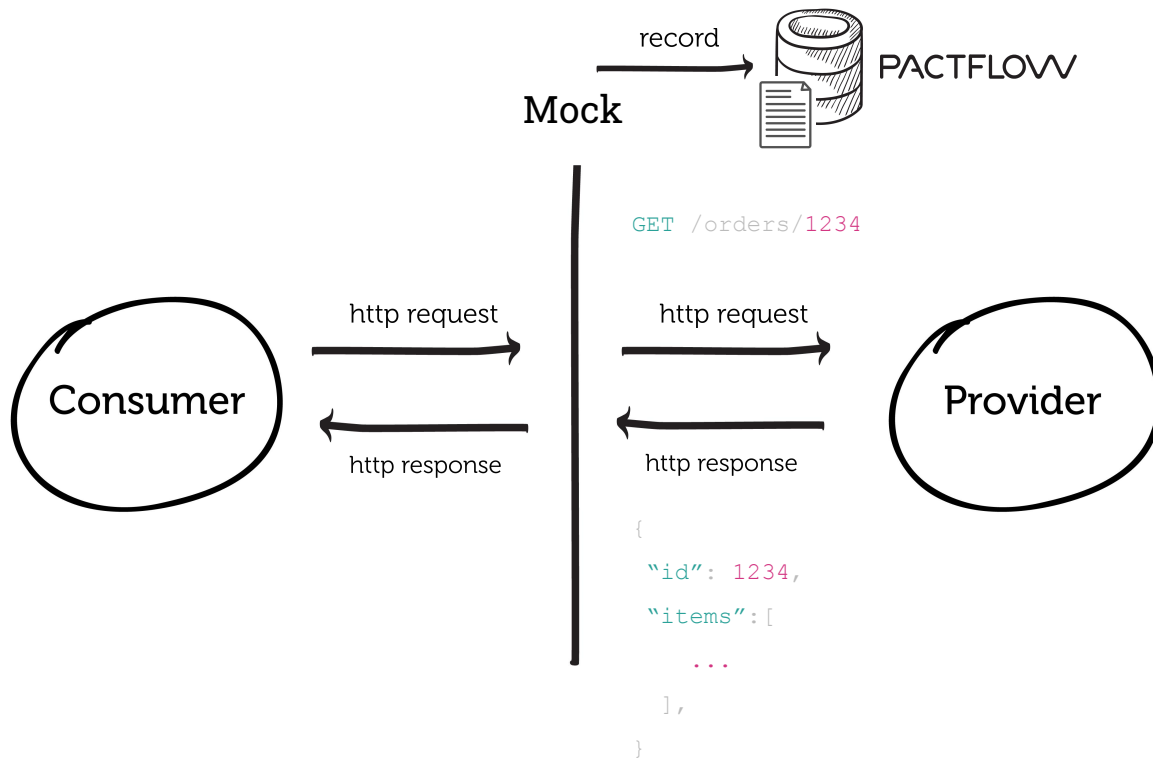
Step 2: share the contract with the Pactflow



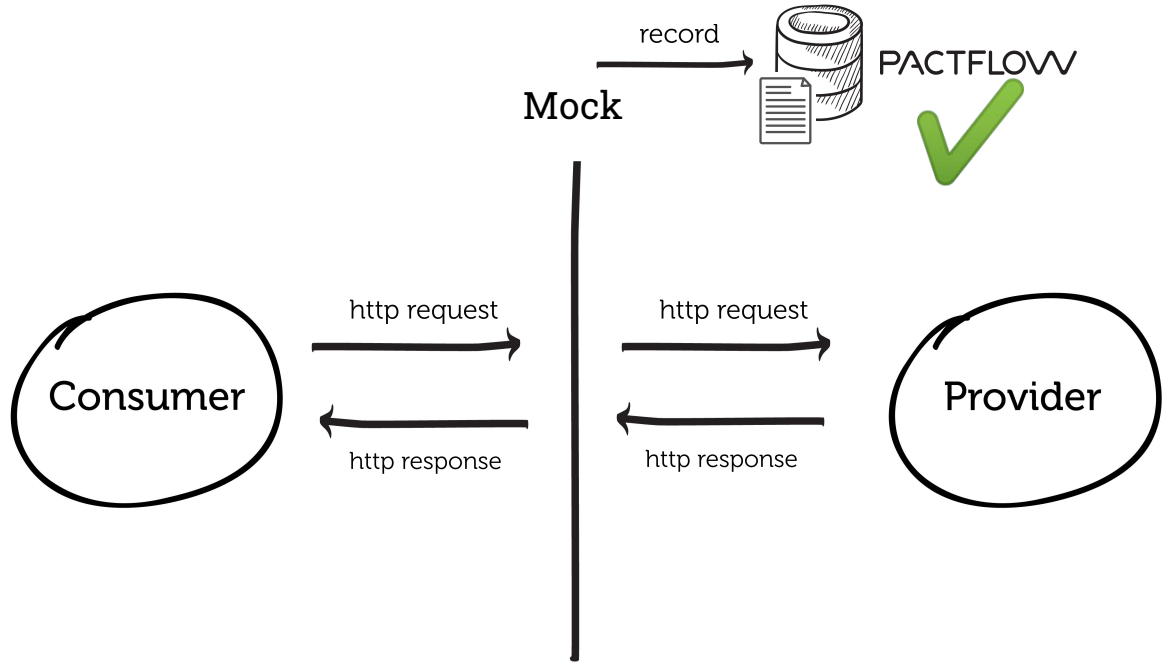
Step 3: test the provider



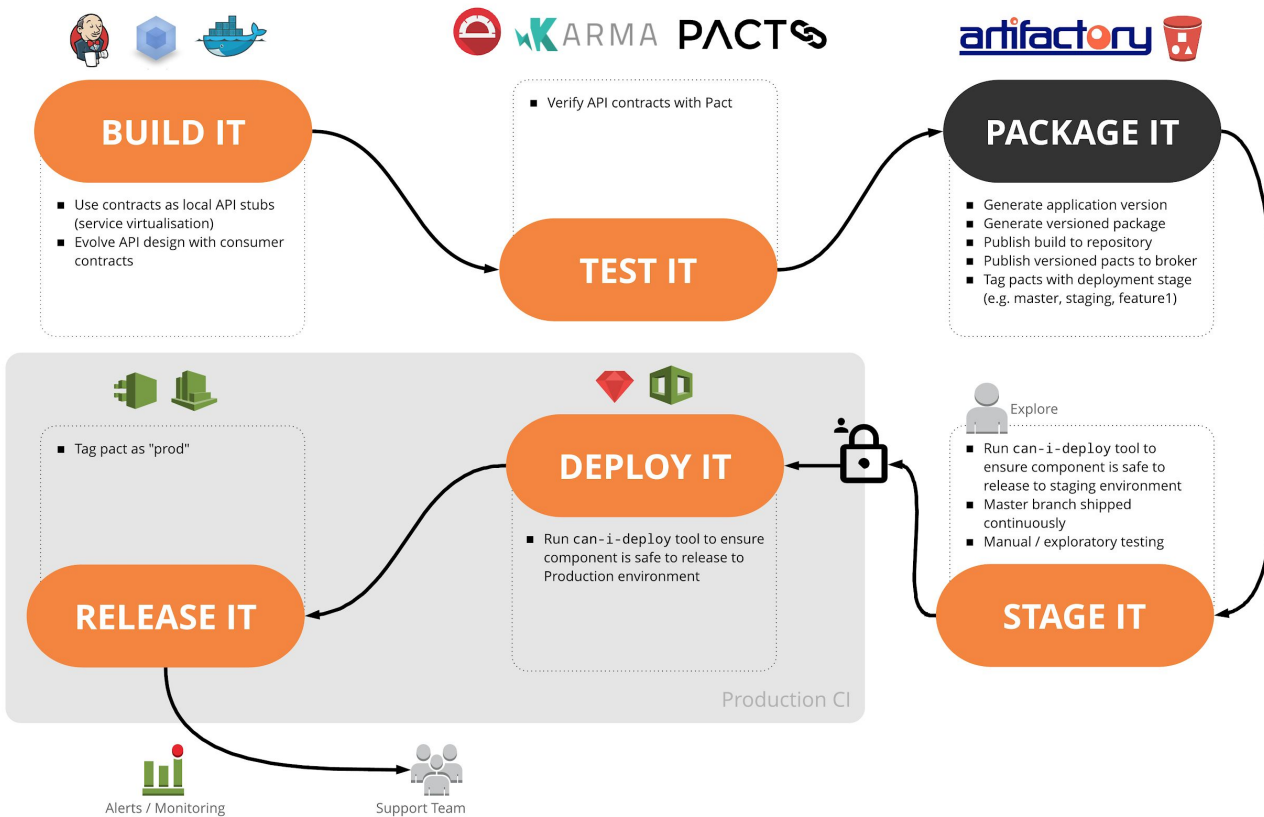
Step 3: test the provider



Step 3: test the provider



Step 4: hook into your CI and CD



Step 4: hook into your CI and CD

`dummyservice/thedummyprofiles-service pact` **verified**

chore: modify pact to test e2e flow

 bethesque committed 18 minutes ago ✓

chore: create webhook for changed pact

 bethesque committed an hour ago

feat: updated gemfile

 bethesque committed 2 hours ago ✓

All checks have passed

2 successful checks

✓  **Animal Service** — Pact tests

✓  **continuous-integration/travis-ci/push** — The Travi... [Details](#)

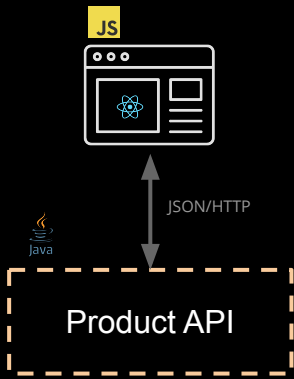
Step 4: hook into your CI and CD

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

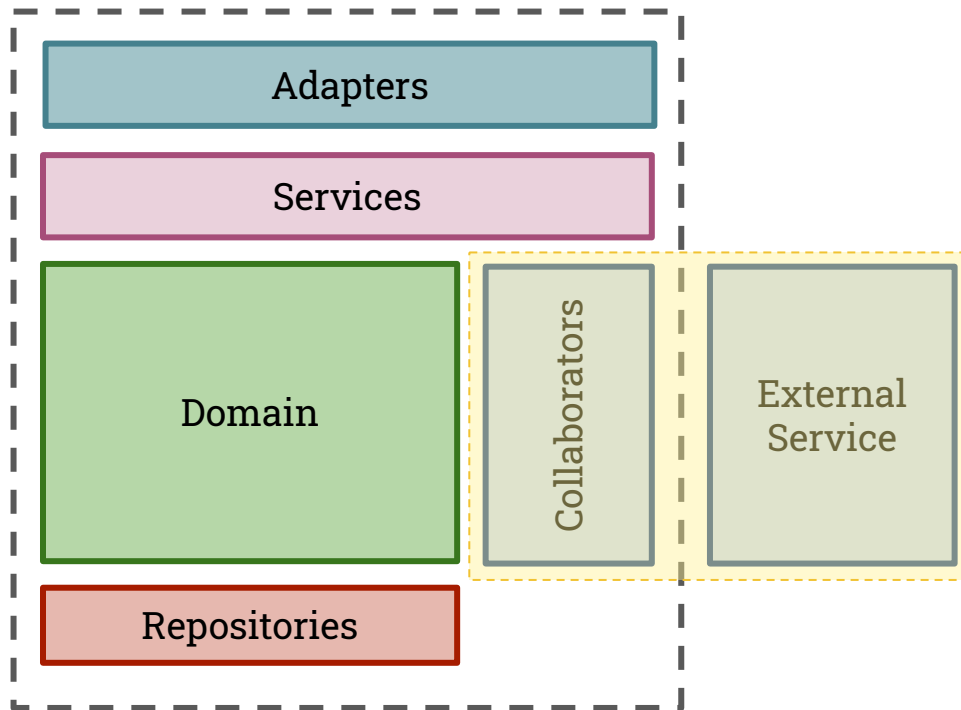
```
$ pact-broker can-i-deploy  
  --app A --version 1  
  --to prod
```

DEMO

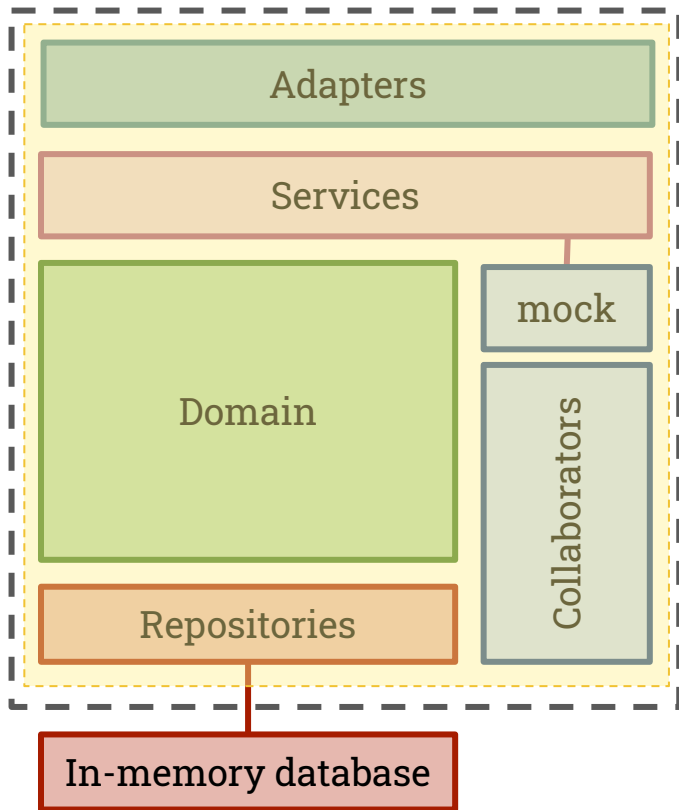
1. Use case: React Product Catalog
2. Consumer pact workflow
 - a. Update `/product/:id` endpoint test
 - b. Show how consumers can work independently
 - c. Publish
 - d. Verification Results
3. Provider workflow



Scope of consumer test



Scope of Provider Test



Finance fintech and neo-banks



Insurance b2b and consumer



Retail and consumer



Technology high-tech and startups



Logistics and supply chain



Marketing technology



Education and online learning



Health and medical technology

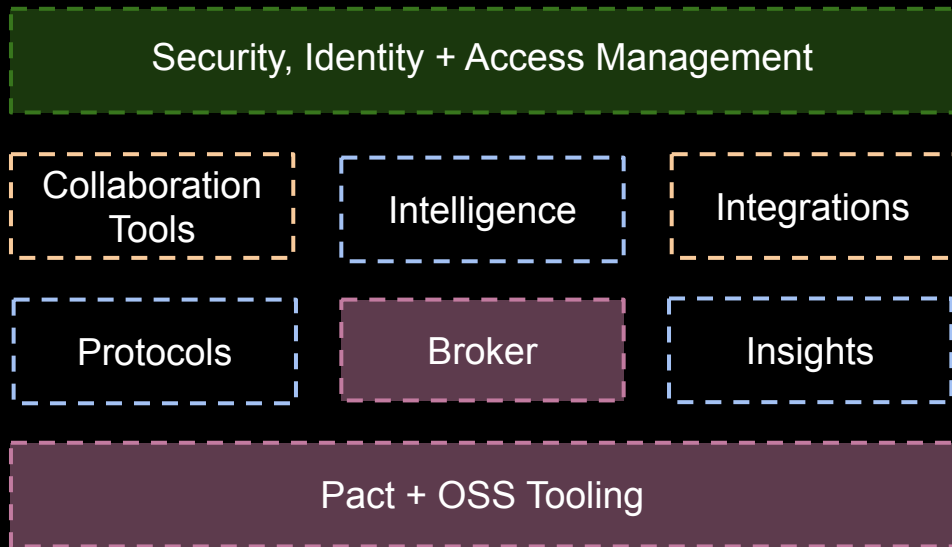


Pactflow

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*



* 2020 Roadmap

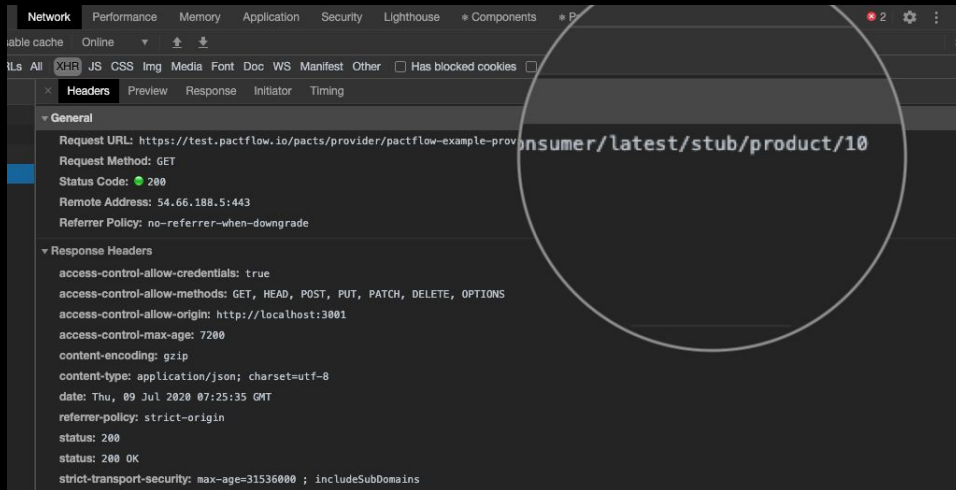
Features

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/



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/

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

PACTFLOW

Brought to you by DIUS 

Features

Secrets

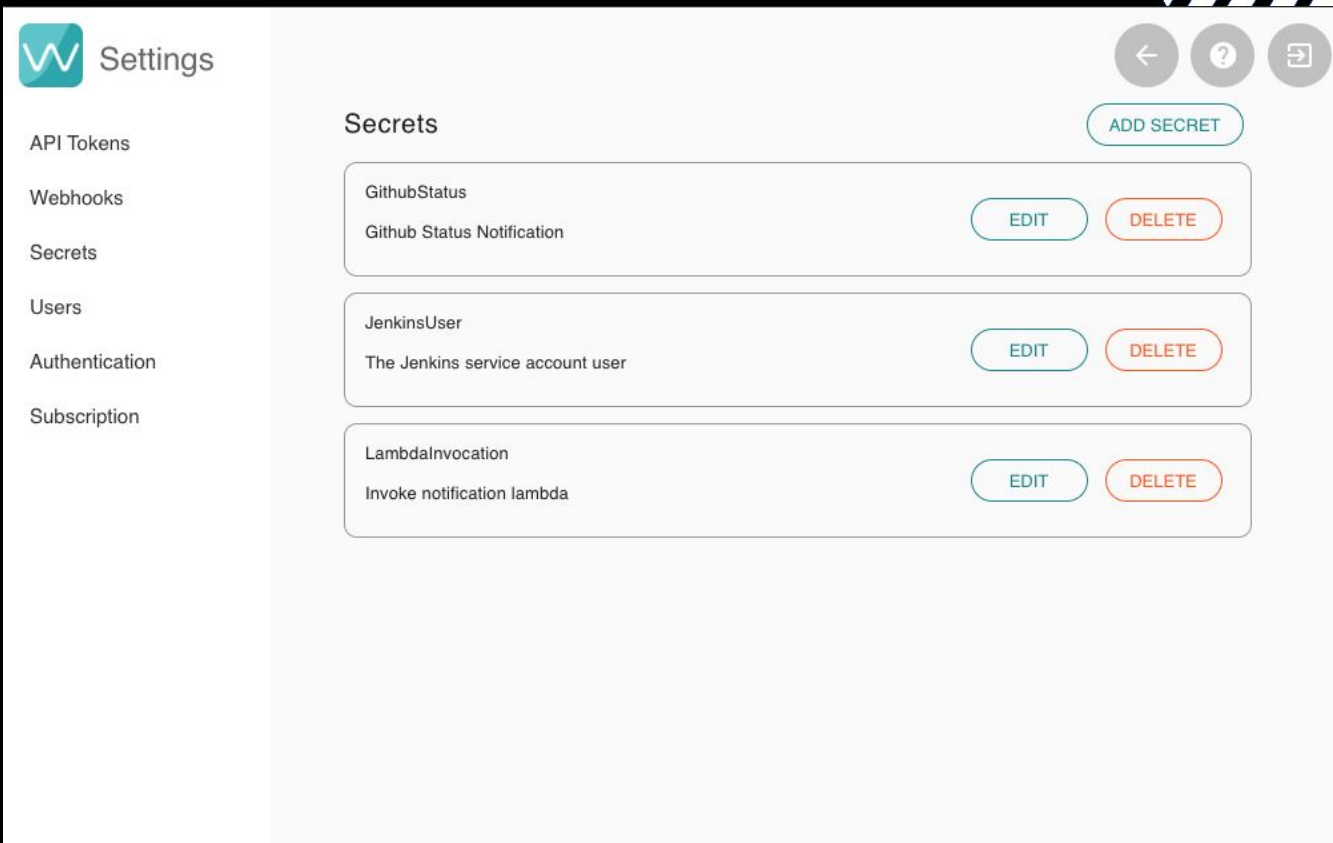
Manage sensitive information with our Secrets Management

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/



The screenshot shows the 'Settings' page in the Pactflow UI. The left sidebar contains a navigation menu with the following items: API Tokens, Webhooks, Secrets, Users, Authentication, and Subscription. The main content area is titled 'Secrets' and features an 'ADD SECRET' button in the top right corner. Below this, there are three secret entries, each with an 'EDIT' button and a 'DELETE' button:

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

Features

Verification Results

The screenshot displays the Pactflow interface for a pact between Matching Service and Animal Profile Service. The main content area shows a list of mismatches, with one highlighted: "A request for all animals given has some animals" with 3 mismatches. The detailed view for this mismatch shows the following error messages:

```

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]
  
```

The Request section shows the following details:

```

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

The Expected response section shows the following details:

```

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

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/

PACTFLOW

Brought to you by DIUS

Features

Infra-as-Code

Automate your Pactflow configuration with Terraform:

- Participants
- Webhooks
- Secrets
- API Tokens

Find out more at


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

```
+ 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.

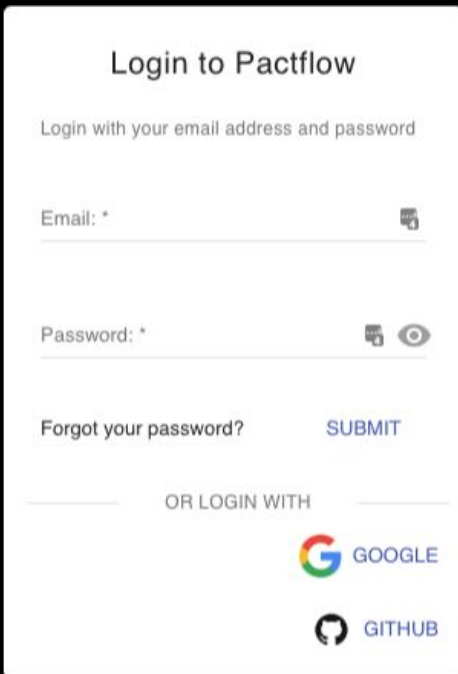


PACTFLOW

Brought to you by DIUS 

Features

Social Login, SSO and SAML 2.0



The screenshot shows a login form titled "Login to Pactflow". Below the title is the instruction "Login with your email address and password". There are two input fields: "Email: *" and "Password: *". The password field has a toggle icon for visibility. Below the password field is a link "Forgot your password?" and a "SUBMIT" button. A horizontal line separates the form from the social login options, which are labeled "OR LOGIN WITH". There are two options: "GOOGLE" with the Google logo and "GITHUB" with the GitHub logo.

Choose how you want to authenticate and manage your users:

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

Find out more at

pactflow.io/blog/saml-and-federated-authentication/

¹ Available on Team or Business plans

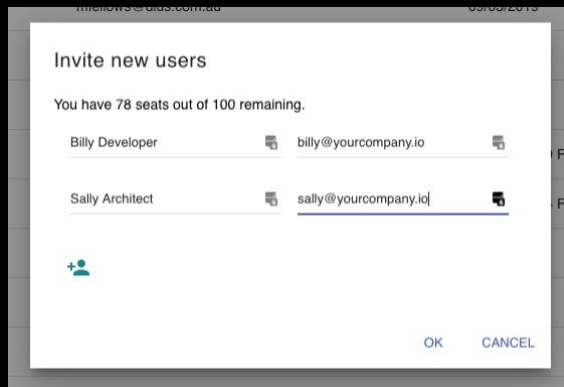
² Available only on Business plans

Features

User Management

Manage your team and access:

- Invite new users
- Review login activity
- Assign roles
- Disable access
-



Name	Email	Role	Created	Updated	Last Login	
Sally Fragins	sally@yourcompany.io	Administrator	09/03/2019	Today at 10:07 PM	06/09/2020	...
Billy Sampson	billy@yourcompany.io	User	09/02/2019	Today at 10:07 PM	06/02/2020	...
Geoffrey Humphrey	geoff@yourcompany.io	User	02/25/2020	Today at 10:07 PM	02/25/2020	...

Find out more at

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

PACTFLOW

Brought to you by DIUS 

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/

PACTFLOW

Brought to you by DIUS 

```
{
  "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/_UjhYvvyEjM9L2SgWd0qsw/regenerate",
        "queryString": "",
        "method": "POST",
        "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_4) Appl",
        "referer": "http://somebroker.pact.dius.com.au/settings/api-tokens",
        "params": {
          "resource_name": "regenerate_token",
          "token_uuid": "_UjhYvvyEjM9L2SgWd0qsw"
        }
      }
    },
    ...
  ],
  "_links": {
    "self": {
      "href": "http://somebroker.pact.dius.com.au/audit?from=zH00xNcjUseyU6DsisadXw"
    },
    "next": {
      "href": "http://somebroker.pact.dius.com.au/audit?from=I1ECUSbMLJL0y8gFbLLuIg"
    }
  }
}
```

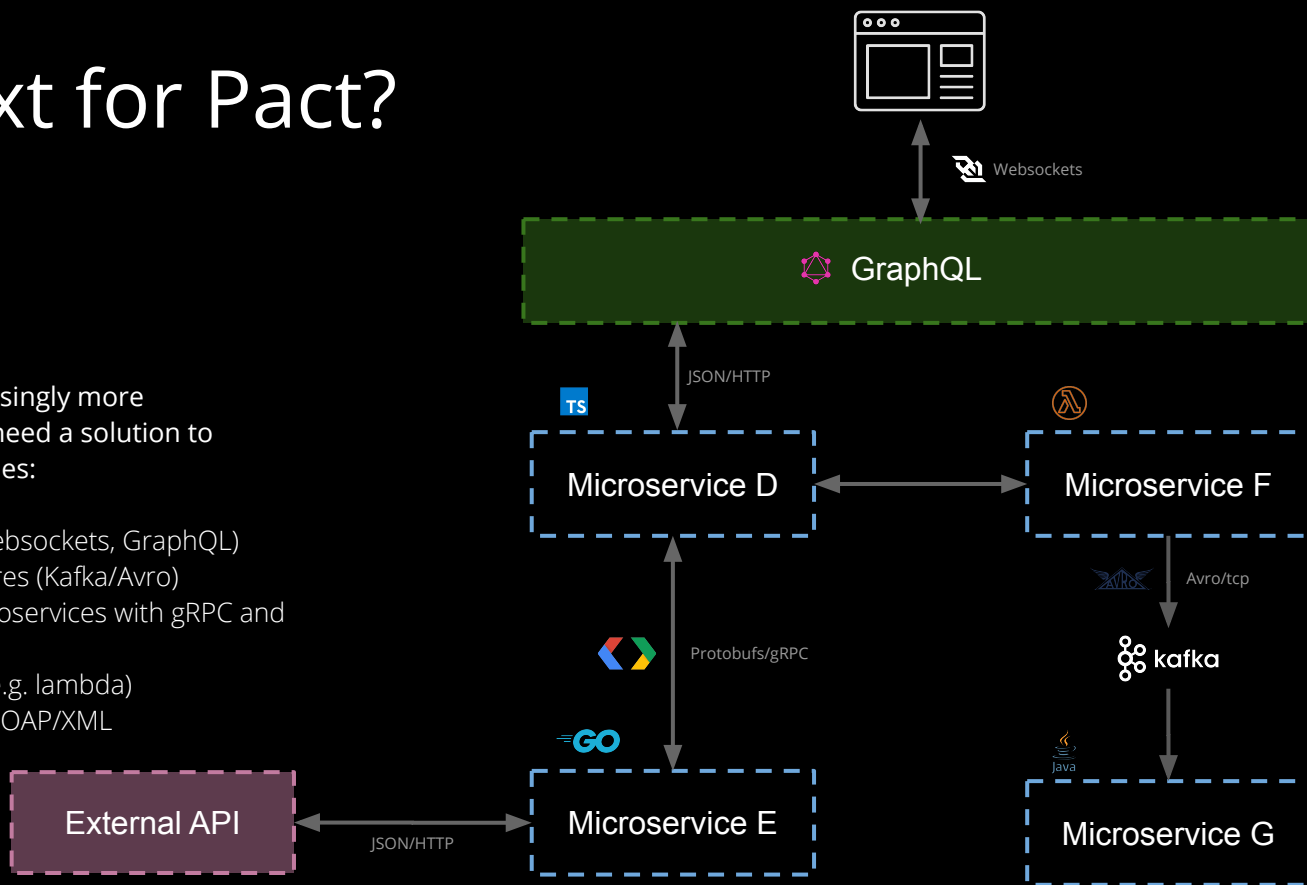
What's next for Pact?

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
- Cypress integration
- External APIs



Team

Email: hello@pactflow.io

Web: 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

PACTFLOW

Brought to you by DIUS 