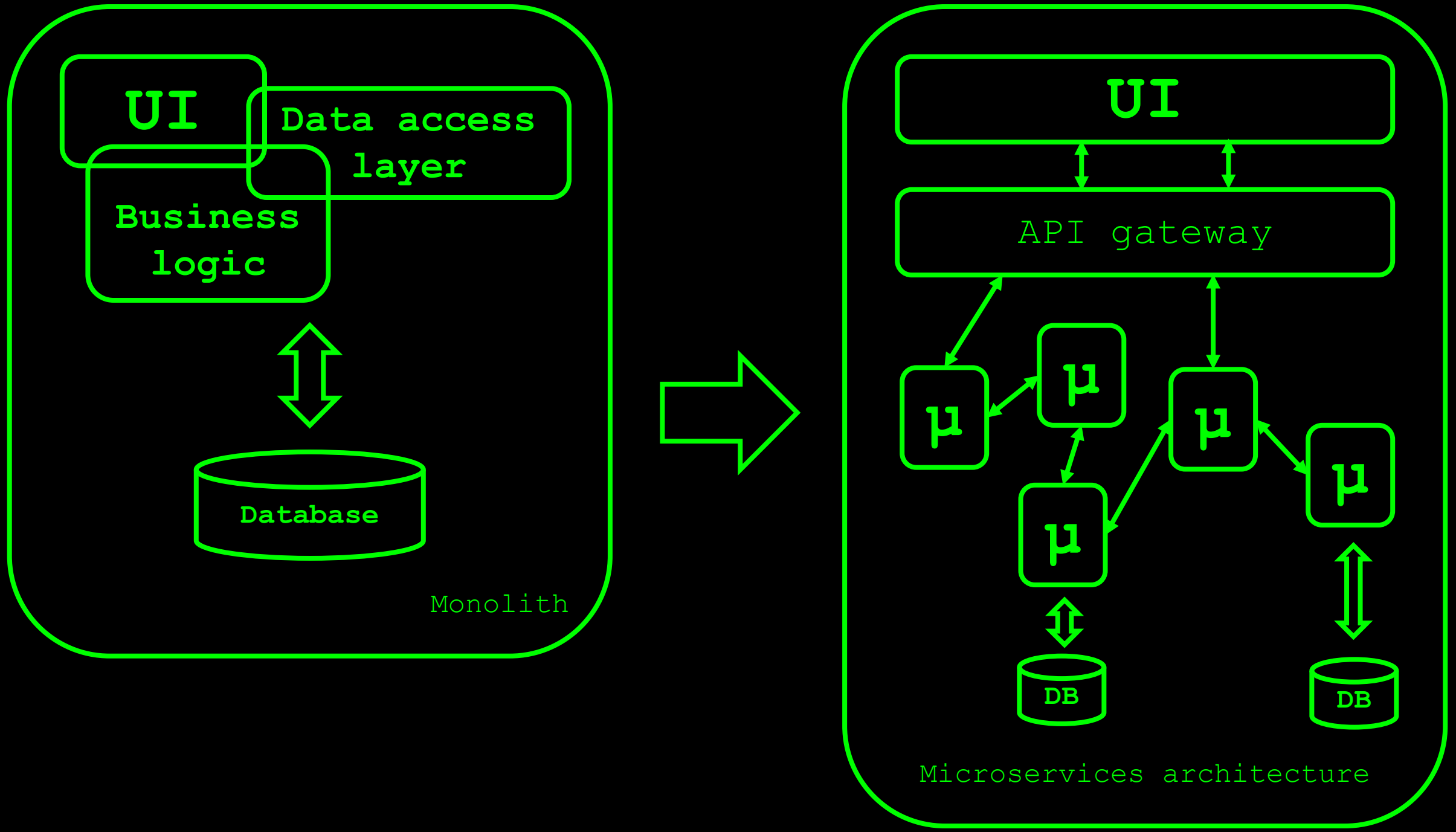
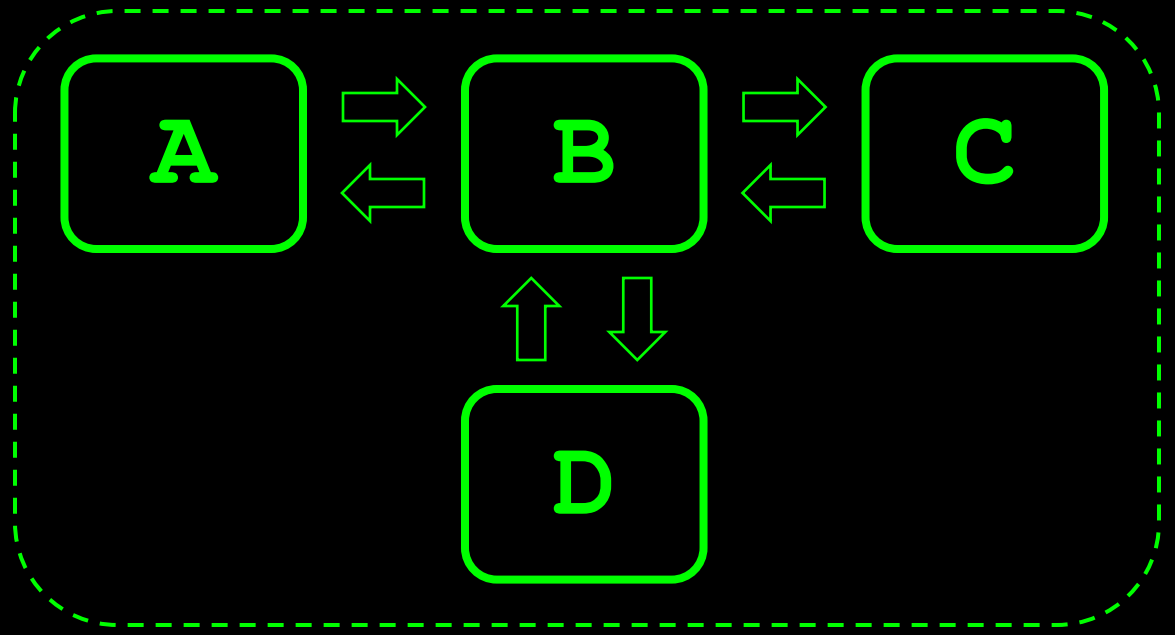
Improving your integration testing efforts with consumer-driven contract testing

Bas Dijkstra

bdijkstra@inspiredtesting.com
www.inspiredtesting.com

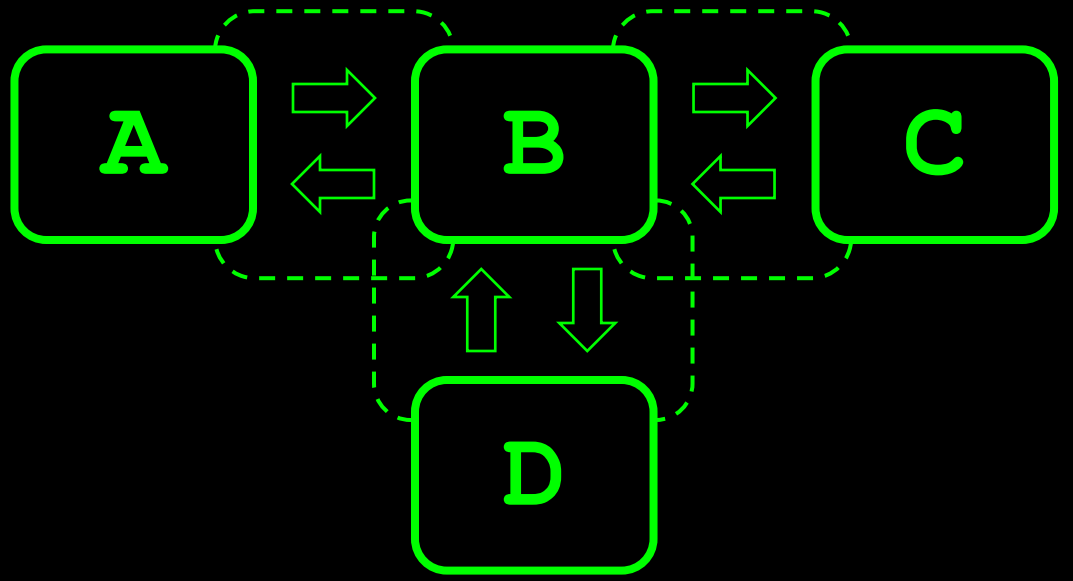

UI
Data access layer
Business logic
Database
Monolith
UI
API gateway
μ
μ
μ
μ
μ
DB
DB
Microservices architecture

Are all individual components and services able to communicate with one another?



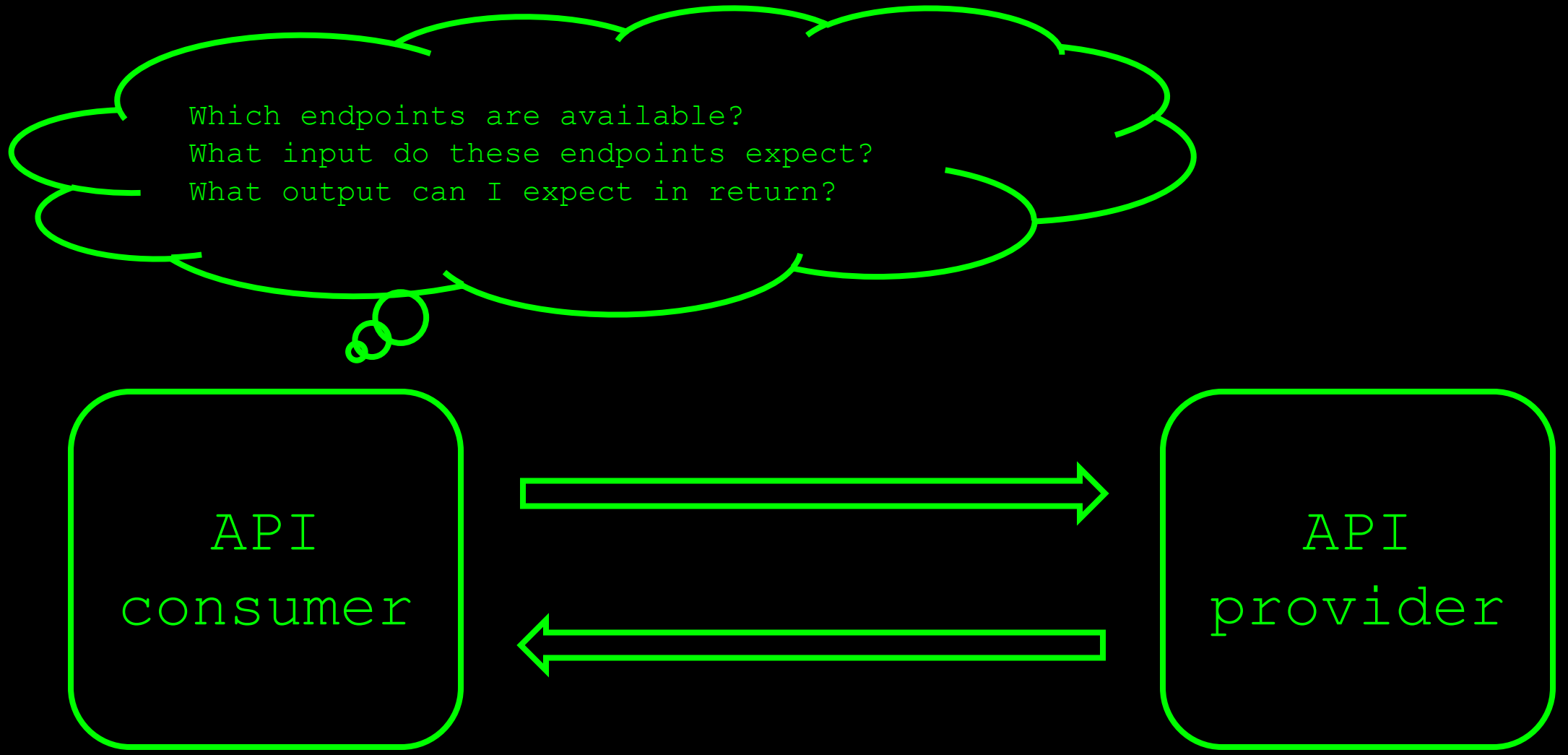
Traditional integration / E2E testing focuses on the integration of ‘everything at once’

Contract-based integration / E2E testing focuses on the integration of individual consumer-provider pairs



‘Traditional’
integration and E2E
testing is **synchronous**

Integration and E2E testing using contracts is **asynchronous**


Which endpoints are available?
What input do these endpoints expect?
What output can I expect in return?
API consumer
API provider

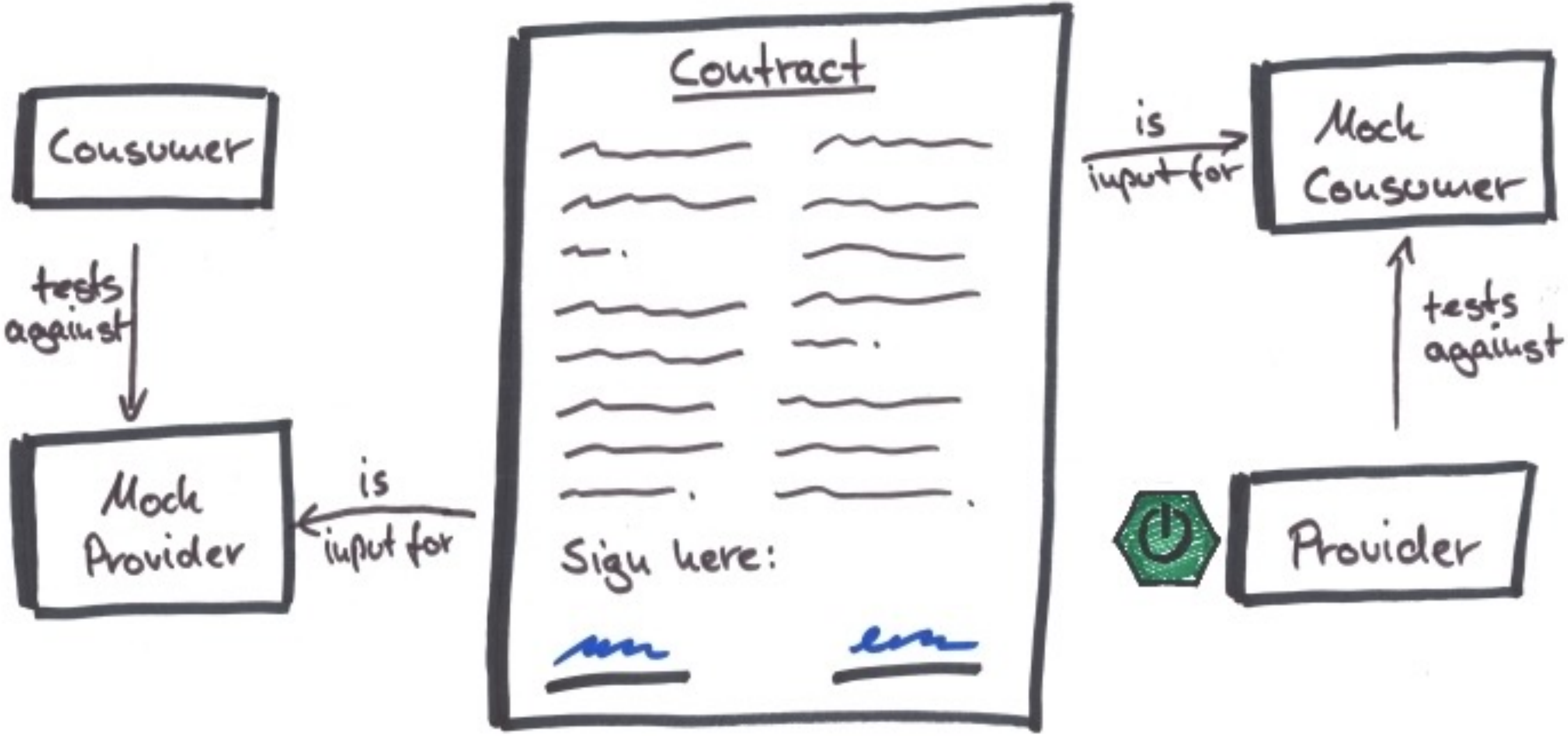
Formalises these expectations in contracts

Automatically check that expectations are met

CDCT

Providers can develop and refactor without fear

Consumers can trust providers to keep working


Consumer
tests against
Mock Provider
is input for
Contract
Sign here:
is input for
Mock Consumer
tests against
Provider

Test the internal logic of a service

(only whether consumer and provider meet the contract)

CDCT doesn't…

Test an entire application end-to-end

(only the communication between 1 consumer and 1 provider)

Pact (Ruby, Java, JavaScript, C#, Go, PHP, Python, …)

https://docs.pact.io/

CDCT tools

Spring Cloud Contract (Java)

https://spring.io/projects/spring-cloud-contract

Consumer driven > consumer generates the contract

Through running unit tests

How does Pact work?

1) Unit test checks that expected response can be processed internally

2) Unit test generates contract from defined expectations

Contracts generated by consumer are then distributed

Provider picks these up and verifies whether it can fulfill the contract

How does Pact work?

Verification results are uploaded / communicated

Life goes on… Or a discussion is started

Time to look at the code!

(finally…)

Now it's your turn!

- Run the tests for both consumers using *mvn clean test*
- Copy the contracts (.json) for both consumers to */src/test/pacts* (overwrite)
- Run the tests for the provider using *mvn clean test*
- Check that the tests pass (provider meets the contract for both consumers)

A change request…

Now it's your turn!

- Change the *customer-consumer* tests so that they expect an **HTTP 200** (instead of a 204) when an address is successfully deleted

- Run the tests for both consumers using *mvn clean test*

- Check that the new expectation is written to the contract

- Copy the contracts (.json) for both consumers to */src/test/pacts* (overwrite)

- Run the tests for the provider using *mvn clean test*

- Check that one test fails and inspect the feedback

Another change request…

Now it's your turn!

- First, undo your change from the previous exercise
- Change the *customer-consumer* tests so that the value for the *state* field can only be **Oklahoma** or **California**
  - The regex you're looking for is *(Oklahoma|California)*
- Run the tests for both consumers using *mvn clean test*
- Check that the new expectation is written to the contract
- Copy the contracts (.json) for both consumers to */src/test/pacts* (overwrite)
- Run the tests for the provider using *mvn clean test*
- Check that one test fails and inspect the feedback

Copying the contracts is manual labor

(and therefore not ideal…)

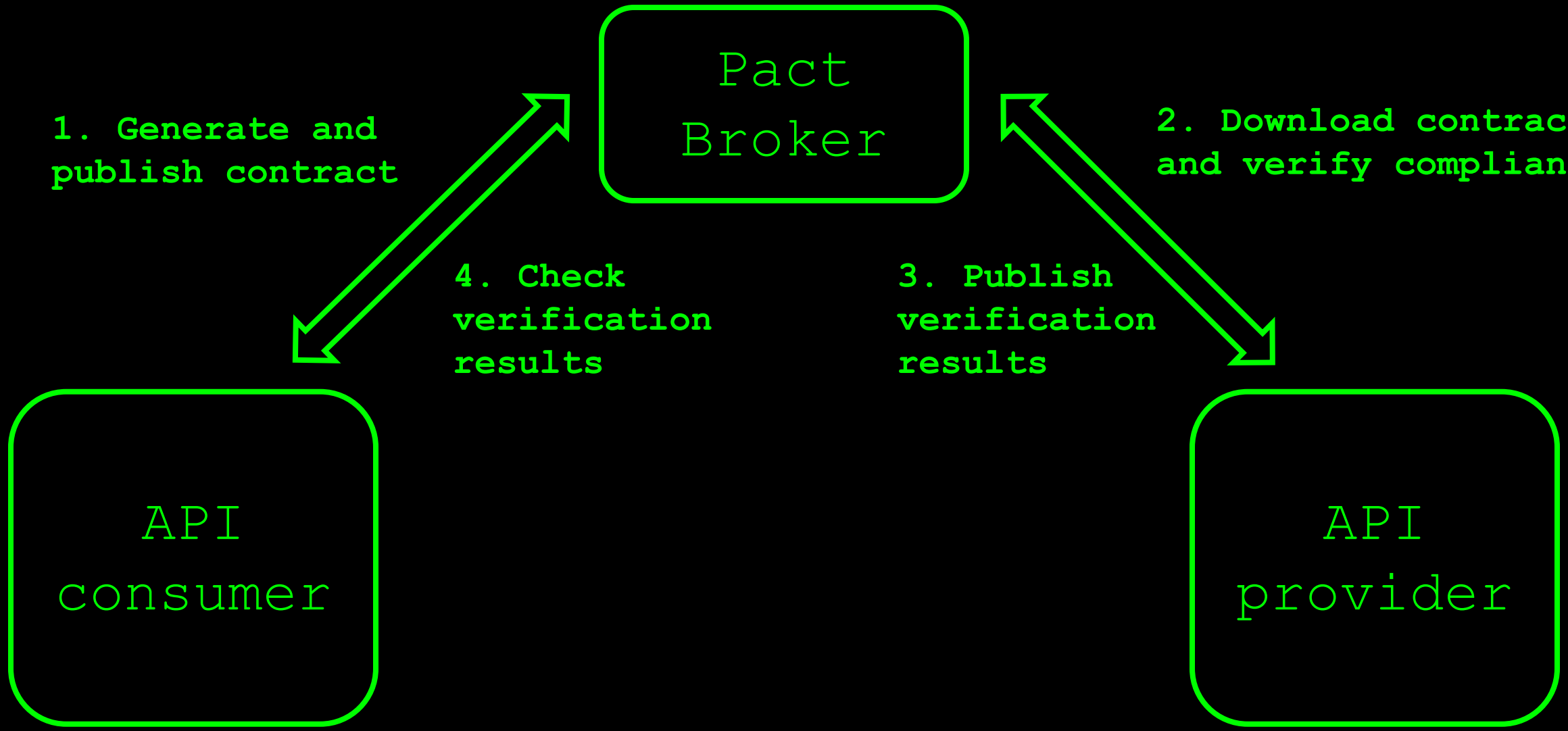
Manual distribution of contracts is not efficient

We need a better mechanism

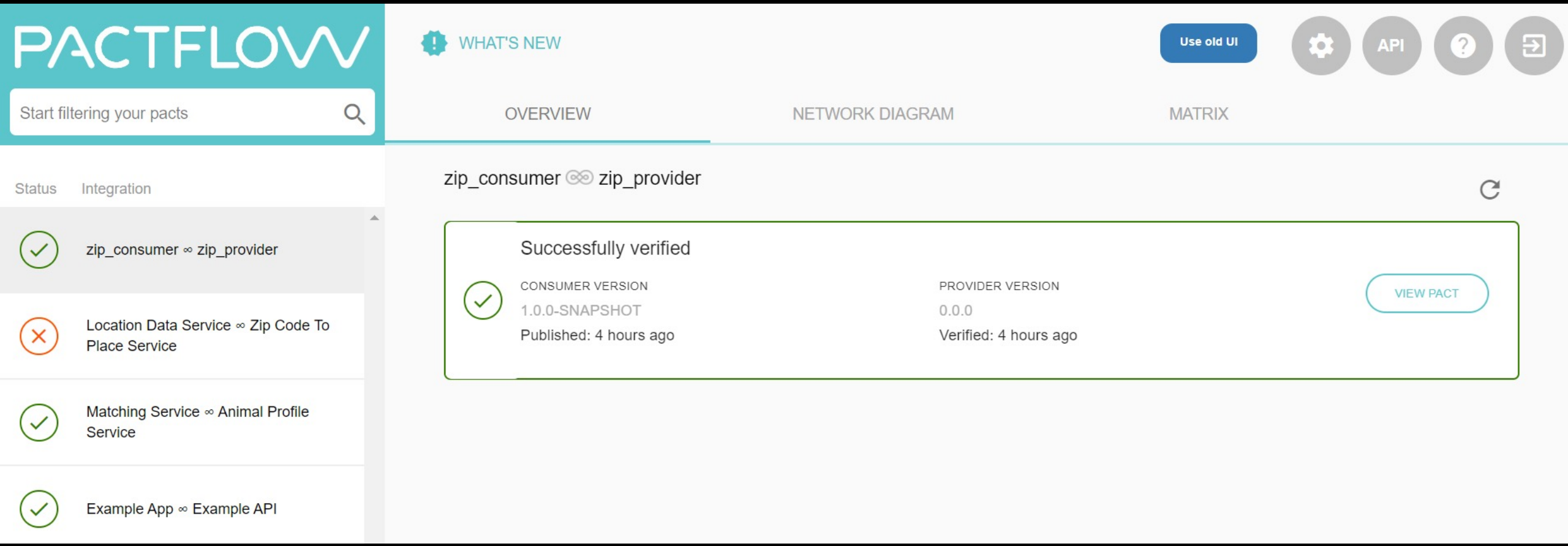
Pact Broker

Automatic distribution and versioning of contracts

Store contract verification results - do providers meet their consumer's contracts?


Pact Broker
1. Generate and publish contract
2. Download contract and verify compliance
4. Check verification results
3. Publish verification results
API consumer
API provider

Pactflow.io


PACTFLOW
Start filtering your pacts
WHAT'S NEW
Use old UI
API
OVERVIEW
NETWORK DIAGRAM
MATRIX
Status
Integration
zip_consumer ∞ zip_provider
Location Data Service ∞ Zip Code To Place Service
Matching Service ∞ Animal Profile Service
Example App ∞ Example API
zip_consumer zip_provider
Successfully verified
CONSUMER VERSION
1.0.0-SNAPSHOT
Published: 4 hours ago
PROVIDER VERSION
0.0.0
Verified: 4 hours ago
VIEW PACT

Testing implementation details of a provider

Testing public APIs

What CDCT does **not** do (well)

https://docs.pact.io
/getting_started
/what_is_pact_good_for

Some useful resources

https://docs.pact.io

https://www.youtube.com/watch?v=U05q0zJsKsU
(the problem with E2E integrated tests)

https://www.youtube.com/watch?v=IetyhDr48RI
(contract testing and how Pact works)

CDCT video series

https://www.youtube.com
/watch?v=6Qd-kq1AzZI
(Demo - contract testing with PactJS)

https://www.youtube.com
/watch?v=3T8J8Pwu3I4
(how do I remove E2E tests?)

https://www.ontestautomation.com/an-introduction-to-contract-testing-part-1-meet-the-players/

https://github.com/basdijkstra/introduction-to-contract-testing

Contact

_Email: bdijkstra@inspiredtesting.com

_Website: https://www.inspiredtesting.com
https://www.ontestautomation.com

_LinkedIn: https://www.linkedin.com/in/basdijkstra