Contract Driven APIs

An Introduction by Kevin Denver

June 2022

What is Contract Testing?

Contract testing is a technique for testing software application interfaces and integrations.

Contracts are typically either "provider driven" (where the team responsible for the provider application maintains and shares the contract with its consumers) or "consumer driven" (where the teams responsible for the consumers maintain and share the contracts).

https://pactflow.io/what-is-contract-testing-page/

What is consumer driven contract testing?

Consumer driven contract testing is a type of contract testing that ensures that a provider is compatible with the expectations that the consumer has of it.

Explicit Contracts

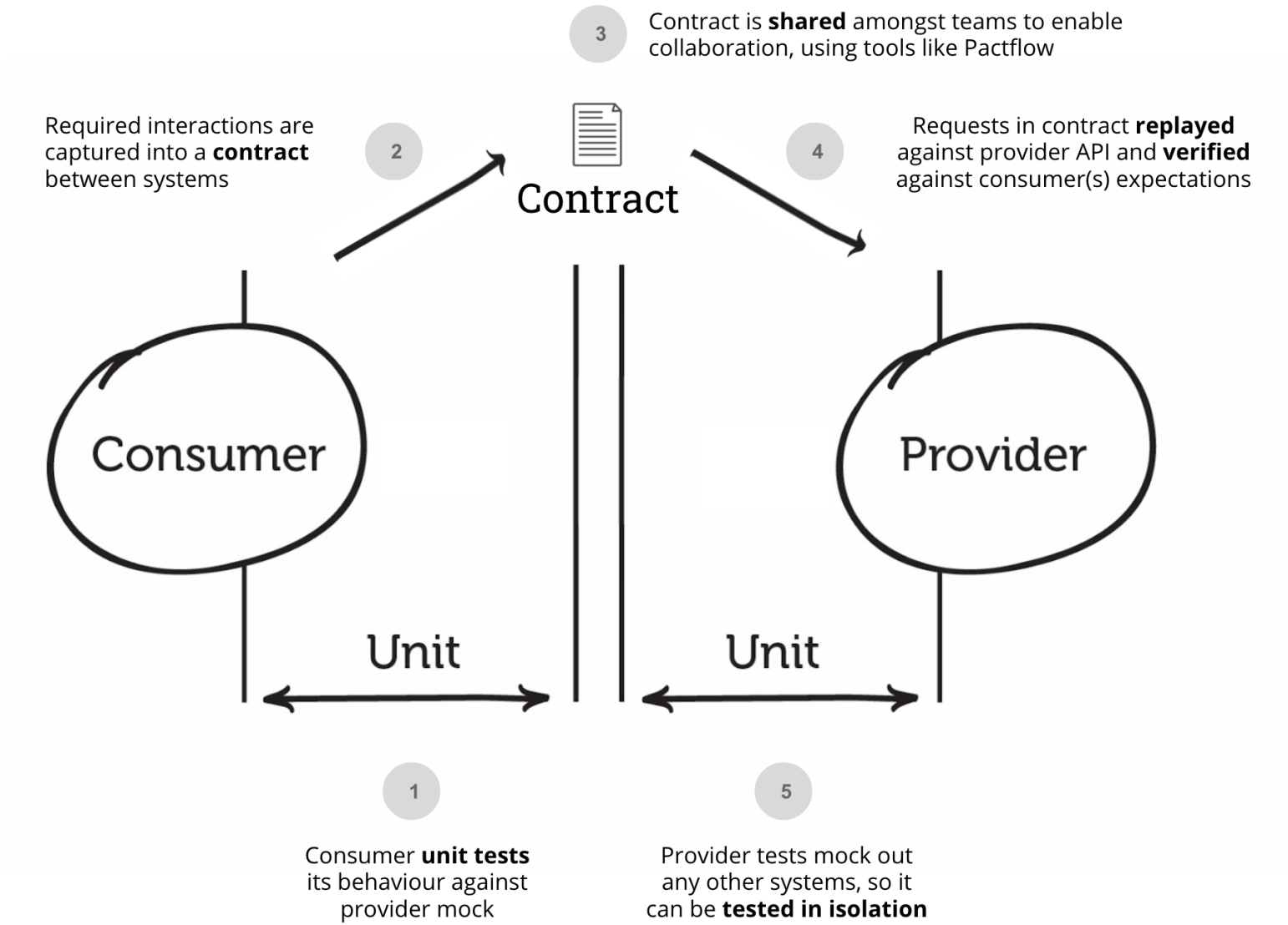
For synchronous protocols such as HTTP, a contract would typically contain a collection of request/response pairs. To verify the contract, each request would be sent to the provider, and the response compared to the expected one. If the actual responses match the expected responses, then the contract has been successfully verified. Any mismatches highlight an incompatibility between the consumer and provider.

Use of an explicit contract is best when the provider is able to incorporate the contract verification step into its own build, providing fast feedback to ensure that no breaking changes are made. This is ideal as it means that breaking changes should not be able to make it into a production release of the application or library.

Implicit contracts

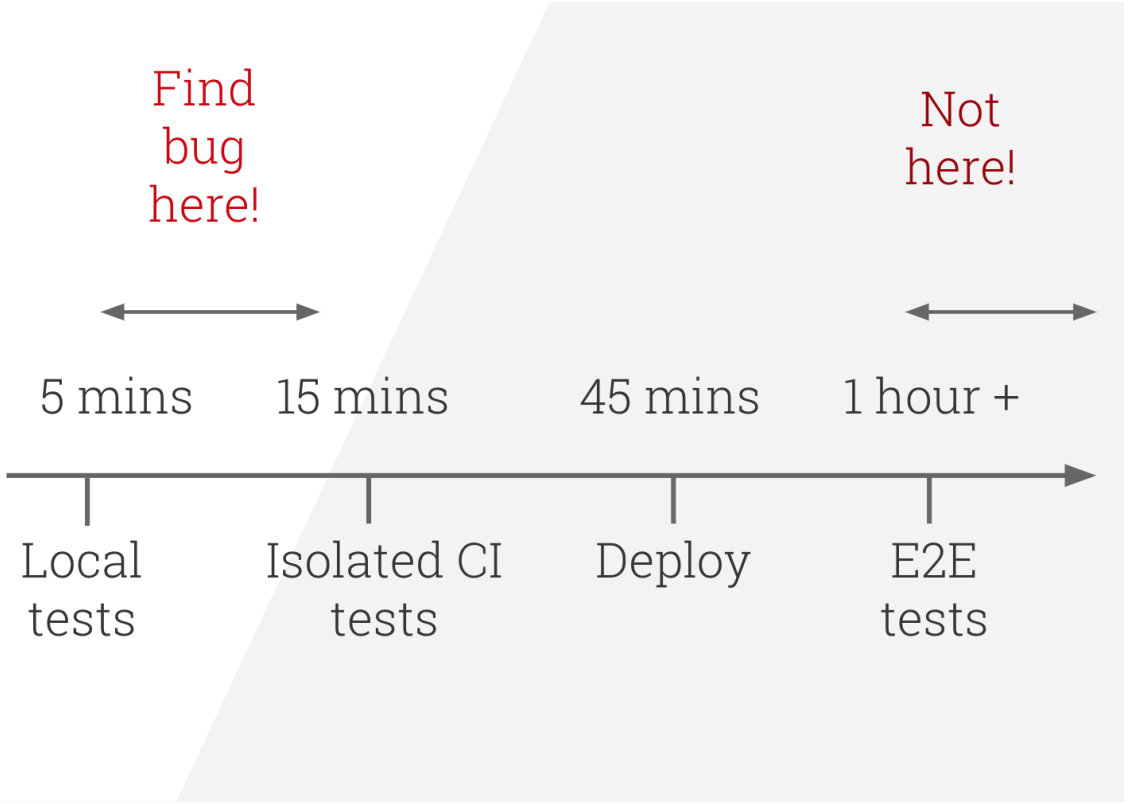
When explicit contracts cannot be used, a similar outcome can be achieved by using a provider test double when running consumer tests, and executing a test harness at regular intervals to ensure that the real provider and the doubled provider behave the same way. The test harness enforces an implicit contract with the provider.

Use of a test harness to enforce an implicit contract is best when the contract verification process cannot be done during the provider's build - for example, for a public API or an OSS code library. While it won't stop breaking changes being released, it will ensure that they are highlighted as soon as possible.


3
Contract is **shared** amongst teams to enable collaboration, using tools like Pactflow
Required interactions are captured into a **contract** between systems
2
Contract
4
Requests in contract **replayed** against provider API and **verified** against consumer(s) expectations
Consumer
Provider
Unit
Unit
1
5
Consumer **unit tests** its behaviour against provider mock
Provider tests mock out any other systems, so it can be **tested in isolation**

Benefits

- The ability to develop the consumer before the API
- The ability to drive out the requirements for your provider first, meaning you implement exactly and only what you need in the provider.
- The ability to immediately see which consumers will be broken if a change is made to the provider API.



https://pactflow.io/what-are-the-benefits-of-contract-testing/

References

- Consumer-Driven Contracts: A Service Evolution Pattern