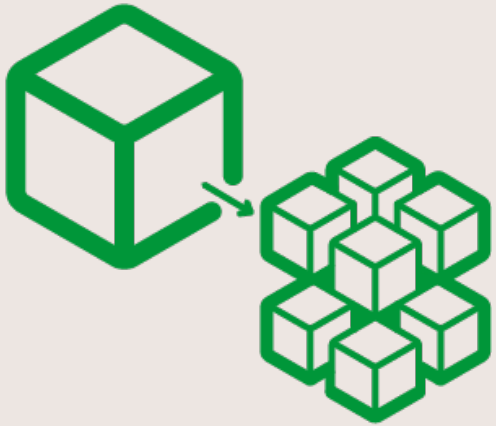
Digital Factory

Microservices with peace of mind

What are microservices?

Microservices are an architectural and organizational approach to software development where software is composed of small independent services that communicate over well-defined APIs. These services are owned by small, self-contained teams.

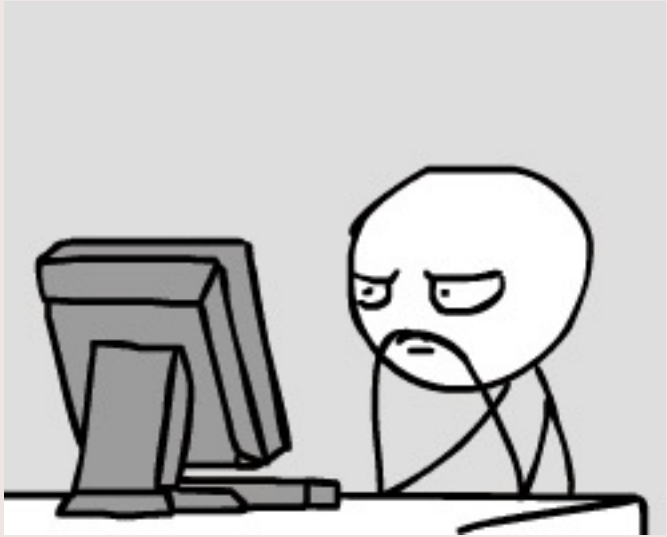


What is of interest to us today about Microservices?

- Communicate between each other (APIs or Buses)
- Built and deployed independently, but they belong to an ecosystem of services

Common challenges

- Does my software adhere to the specifications shared with other teams?
- Will this deployment break “contracts” with other services in the ecosystem?
- Mocking external services



Consumer driven contract testing

Testing microservices shouldn't involve setting up complex end-to-end test environments, creating lengthy integration suites and managing test data. Stop wasting time and start releasing.

Supported by

Java

.Net

Javascript

Scala

PHP

Python

Go

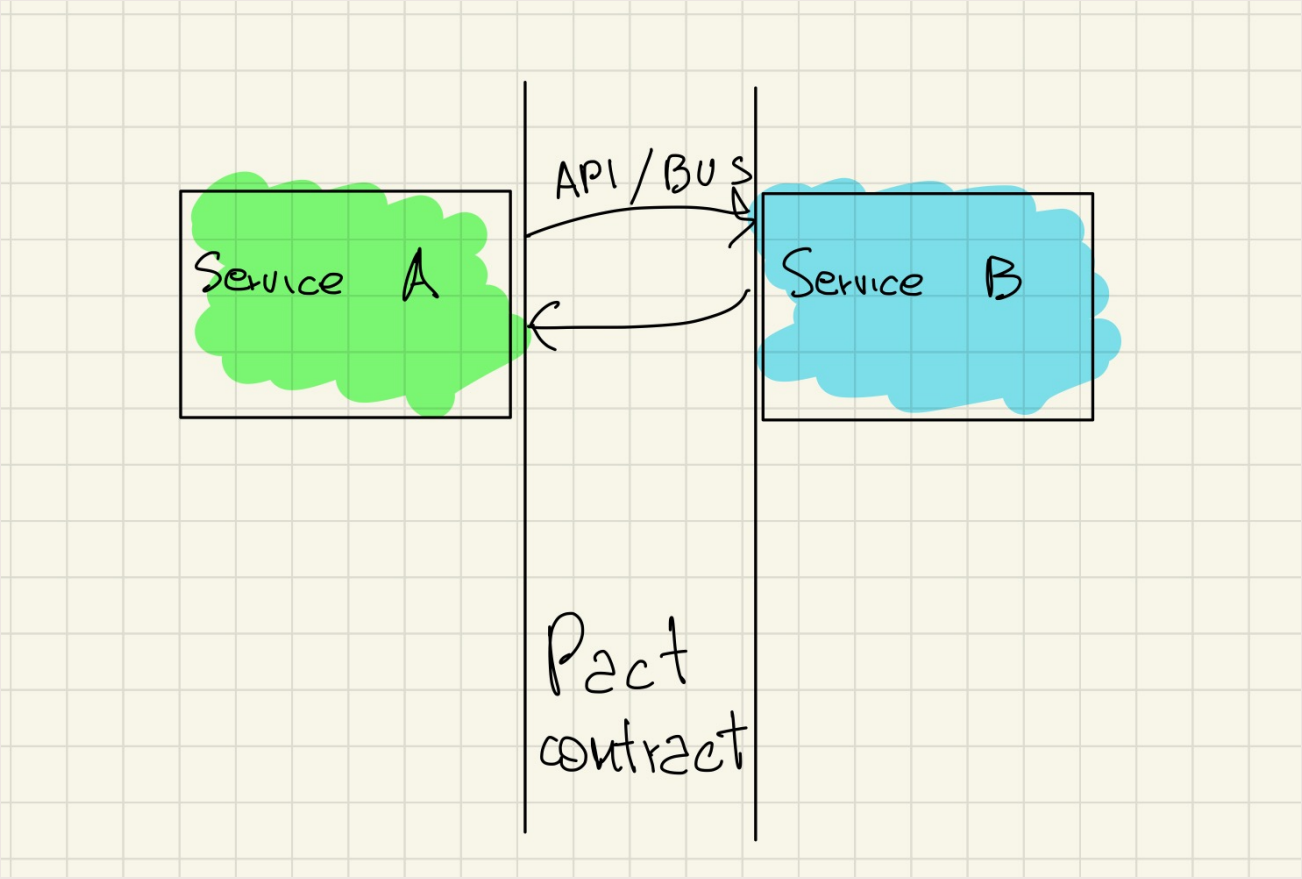
Swift

Rust

Ruby

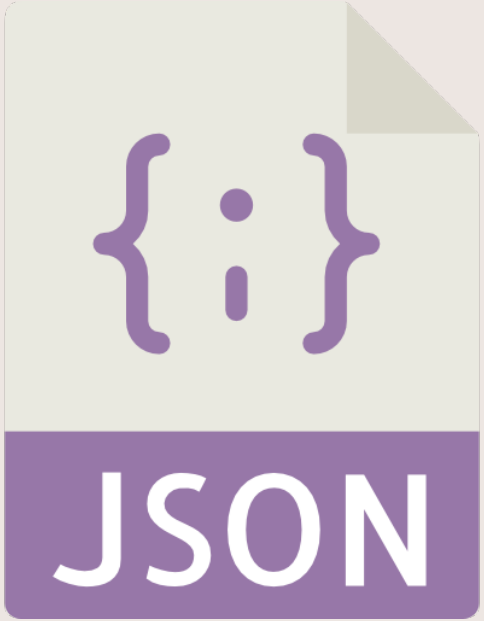
C++

How does Pact work?



A Pact contract

- Is automatically generated when running Unit/Component tests on the consumer
- Mentions what service consumes the data and which one produces it (Consumer / Provider or Client / Server)
- Contains all the details of data format (no side-effect)
- Is stored and distributed from a Pact broker (SaaS Pactflow)



The Consumer’s perspective

```
"use strict";

import { pactWith } from "jest-pact";
import { Matchers } from "@pact-foundation/pact";
import { getAccountInfo } from "./accountService";
const { term } = Matchers;

pactWith(
  {
    consumer: "v1",
    provider: "account-service",
    port: 5000,
    host: "127.0.0.1",
  },
  (provider) => {
    describe("account API", () => {
      const accountResponse = {
        status: 200,
        headers: {
          "Content-Type": Matchers.term({
            generate: "application/json",
            matcher: "application/json",
          }),
        },
        body: Matchers.like({
          account: {
            first_name: "Josefa",
            last_name: "Ortiz",
            email: "Ezekiel32@gmail.com",
            balance: 500,
          },
        }),
      };

      const accountRequest = {
        uponReceiving: "a request for an account data",
        withRequest: {
          method: "GET",
          path: term({
            generate: "/v1/accounts/1",
            matcher: "/v1/accounts/[0-9]+",
          }),
          headers: {
            Accept: "application/json, text/plain, */*",
          },
        },
      };

      beforeEach(() => {
        const interaction = {
          state: "i have an account",
          ...accountRequest,
```

Consumer’s Unit tests

API mock

Contract definition

PACTS

Mocks the Provider’s responses

Clear Channel

The Provider’s perspective

PACTS

Mocks the Provider’s responses

Acts as our consumer

```
accountService.test.js
"use strict";

import { pactWith } from "jest-pact";
import { Matchers } from "@pact-foundation/pact";
import { getAccountInfo } from "./accountService";
const { term } = Matchers;

pactWith(
  {
    consumer: "ui",
    provider: "account-service",
    port: 5000,
    host: "127.0.0.1",
  },
  (provider) => {
    describe("account API", () => {
      const accountResponse = {
        status: 200,
        headers: {
          "Content-Type": Matchers.term({
            generate: "application/json",
            matcher: "application/json",
          }),
        },
        body: Matchers.like({
          account: {
            first_name: "Josefa",
            last_name: "Ortiz",
            email: "Ezekiel32@gmail.com",
            balance: 300,
          },
        }),
      };

      const accountRequest = {
        uponReceiving: "a request for an account data",
        withRequest: {
          method: "GET",
          path: term({
            generate: "/v1/accounts/1",
            matcher: "/v1/accounts/[0-9]+",
          }),
          headers: {
            Accept: "application/json, text/plain, */*",
          },
        },
      };

      beforeEach( fn: () => {
        const interaction = {
          state: "i have an account",
          ...accountRequest,
```

Provider’s Unit tests

Pact allows us to verify expectations from distributed environments without the need to set up locally or in the cloud ”complex” integration environments.

And I am not done yet…

The Matrix

The Matrix

Pacticipant name zoo-app — All versions

Pacticipant name animal-service — All versions

- (•) **Show all results**
- ( ) **Show latest result for each consumer version/provider version**
- ( ) **Show latest result for each consumer version/provider**

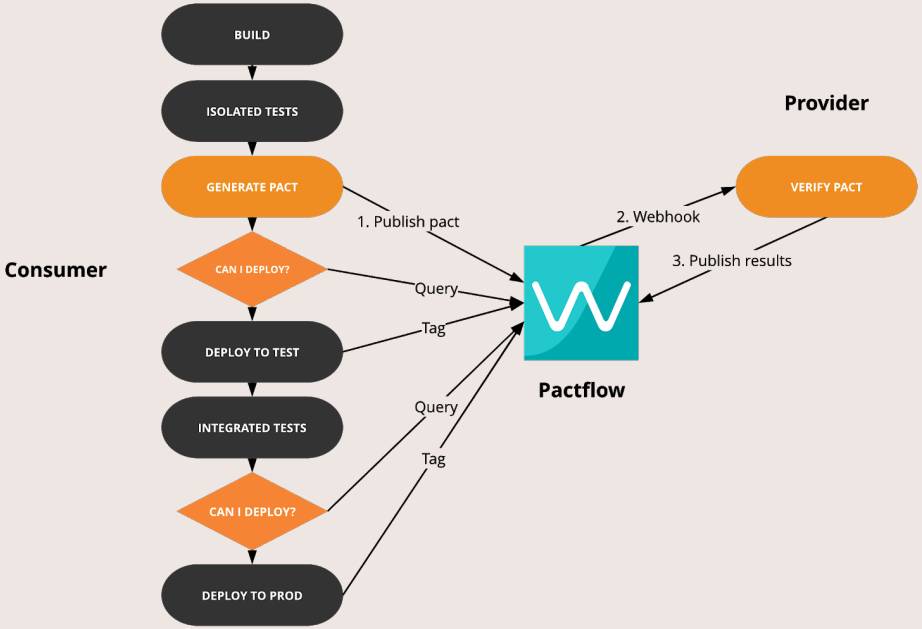
Limit* 100

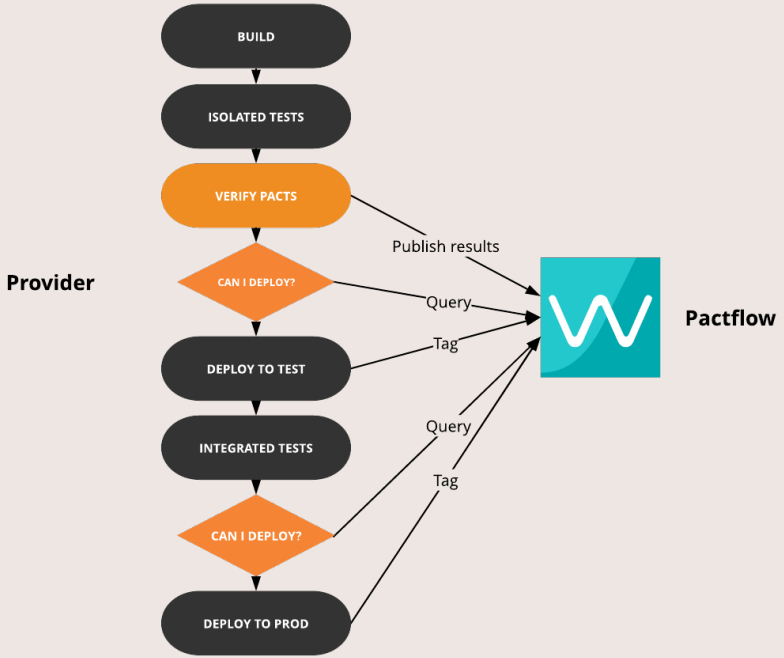
Submit

3 rows

Consumer	Consumer Version	Pact Published	Provider	Provider Version	Pact verified
zoo-app	0724701 master	about 12 hours ago (revision 1)	animal-service		
zoo-app	955b9ea master	2 days ago (revision 1)	animal-service	82b59ef master	1 day ago (number 2)
zoo-app	9632729 prod master	4 days ago (revision 1)	animal-service	a519731 prod master	3 days ago (number 1)

How could pipelines potentially look like?



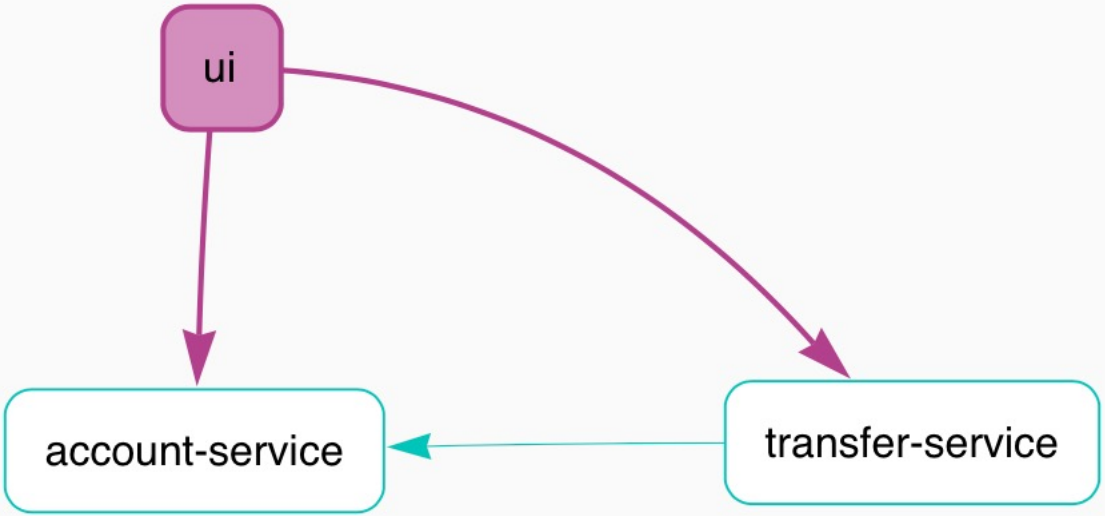


Clear Channel

Questions?

Demo

Money transfer between accounts using Pact


ui
account-service
transfer-service

Source | Destination | Amount | SUBMIT

first name	last name	email	balance
Alessio Eros	Ferri	alessio.ferri@clearchannel.co.uk	€350.00
Diana	Boyer	Genesis92@hotmail.com	€750.00
Andreane	Klocko	Helena16@yahoo.com	€300.00

Questions?