API Contract Testing with Postman/Newman or Pact

API contract testing ensures that the communication between different components of a system (e.g., client and server) follows a defined structure or "contract." This becomes crucial in distributed systems and microservice architectures where breaking a contract can cause downstream failures.

🧪 Why API Contract Testing?

- Prevents breaking changes in APIs
- Improves collaboration between frontend and backend teams
- Fast and lightweight compared to full end-to-end tests
- Helps enable CI/CD by validating integration points

2. Pact (Consumer-Driven Contract Testing)

Pact is a powerful framework based on **consumer-driven contracts**. It allows consumers to define their expectations, which providers must then fulfill.

✅ **Pros:**

- Strong versioning and contract negotiation
- Supports multiple languages (JS, Java, Python, Go)
- CI/CD friendly with Pact Broker for sharing contracts

🔄 **Workflow:**

1. **Consumer** writes a test that generates a pact file (JSON).
2. Pact file is shared with **provider**.
3. **Provider** runs tests to ensure it satisfies the consumer’s expectations.

🔧 **Example (Node.js):**

```
const { Pact } = require('@pact-foundation/pact');
const provider = new Pact({
  port: 1234,
  consumer: 'FrontendApp',
  provider: 'UserService',
});

describe('Contract Test', () => {
  before(() => provider.setup());

  it('should return a user', async () => {
    await provider.addInteraction({
      uponReceiving: 'a request for user data',
      withRequest: {
        method: 'GET',
        path: '/user/1',
      },
      willRespondWith: {
        status: 200,
        body: { id: 1, name: 'Alice' },
      },
    });

    // call your API client here
  });

  after(() => provider.finalize());
});
```

🔄 CI/CD Integration Example

- Run **Newman** collections post-deployment in your Jenkins or GitHub Actions workflow.
- Use **Pact** to verify contract compatibility during pull requests before merging to main.

📌 Conclusion

API contract testing is a critical part of ensuring stable and reliable communication between services. Whether you're using Postman for quick validation or Pact for full-blown consumer-driven contracts, integrating these practices helps catch issues early and streamline development in modern, service-oriented architectures.