# Cloudless Testing of Serverless Applications


## Dr. Jan Mayer

jan.mayer@valtech-mobility.com

janmayer | dr-jan-mayer

> The hardest single part of building a software system is deciding precisely what to build.
>
> - Frederick Brooks, 1987

> The hardest single part of building a software system is deciding precisely what to build.
>
> - Frederick Brooks, 1987

> The second hardest part is keeping the documentation in sync with reality.
>
> - me, now

# Behavior-Driven Development

You are already doing it.

# Behavior?

- **Who** & **Why**: User Experience
- **How**: Contracts (Boundaries, Interfaces)
- **What**: Features (Acceptance Criteria)

# New requirement:
# Car Descriptions

> We need a detailed description for individual cars
> to display on the frontend and in our PDFs for vendors.
> The cars are highly customizable, so
> every description will be different.

# Formalize Features ...

```
# features/generate_vehicle_descriptions.feature (Gherkin)

Feature: Generate Vehicle Descriptions

    As a vehicle description service user
    I want to generate descriptions for vehicles
    So that I can provide comprehensive information to customers.

    Scenario: Generate description for known vehicle
        Given details about a specific vehicle are available
        And the AI description generator is functional
        When the user requests a description for this vehicle
        Then a description for this vehicle is provided
```

# … and automate

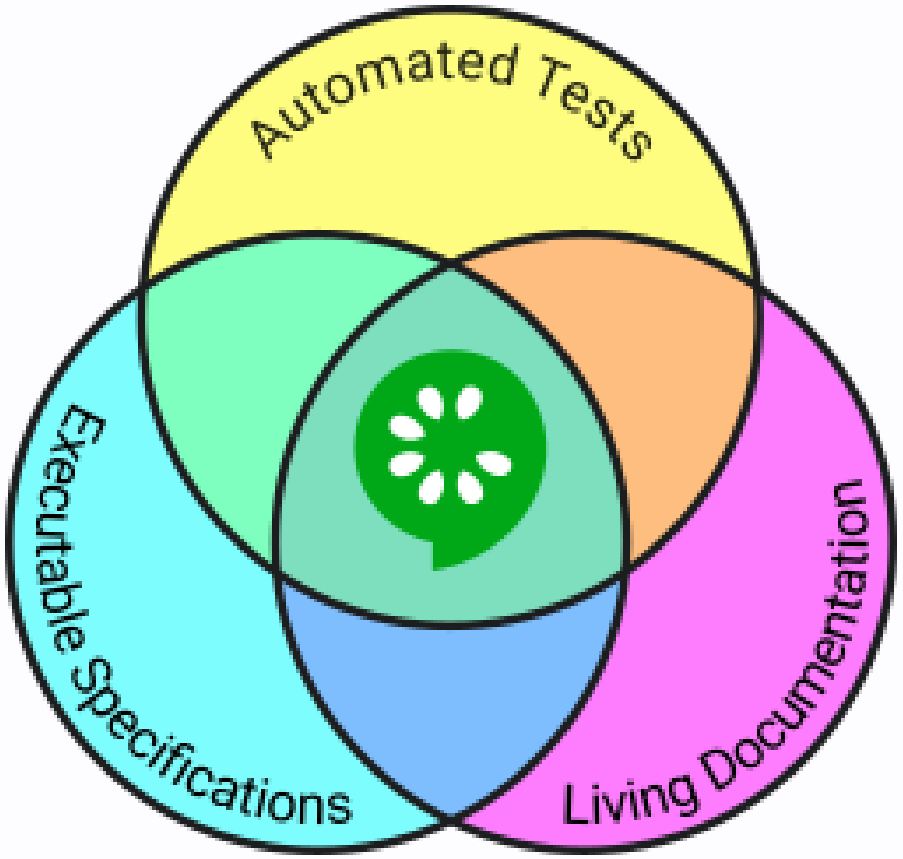
```python
# features/steps/steps.py (Python)

@when("the user requests a description for this vehicle")
def request_description(context):
    context.response = requests.get(
        f"http://localhost:8080/vehicles/{context.vin}/description"
    )

@then("a description for this vehicle is provided")
def ensure_description(context):
    assert context.response.status_code == 200
    assert context.response.json()["vin"] == context.vin
    assert context.response.json()["description"]
```

# Benefits

- Collaboration
- Clarity
- Testability
- Documentation



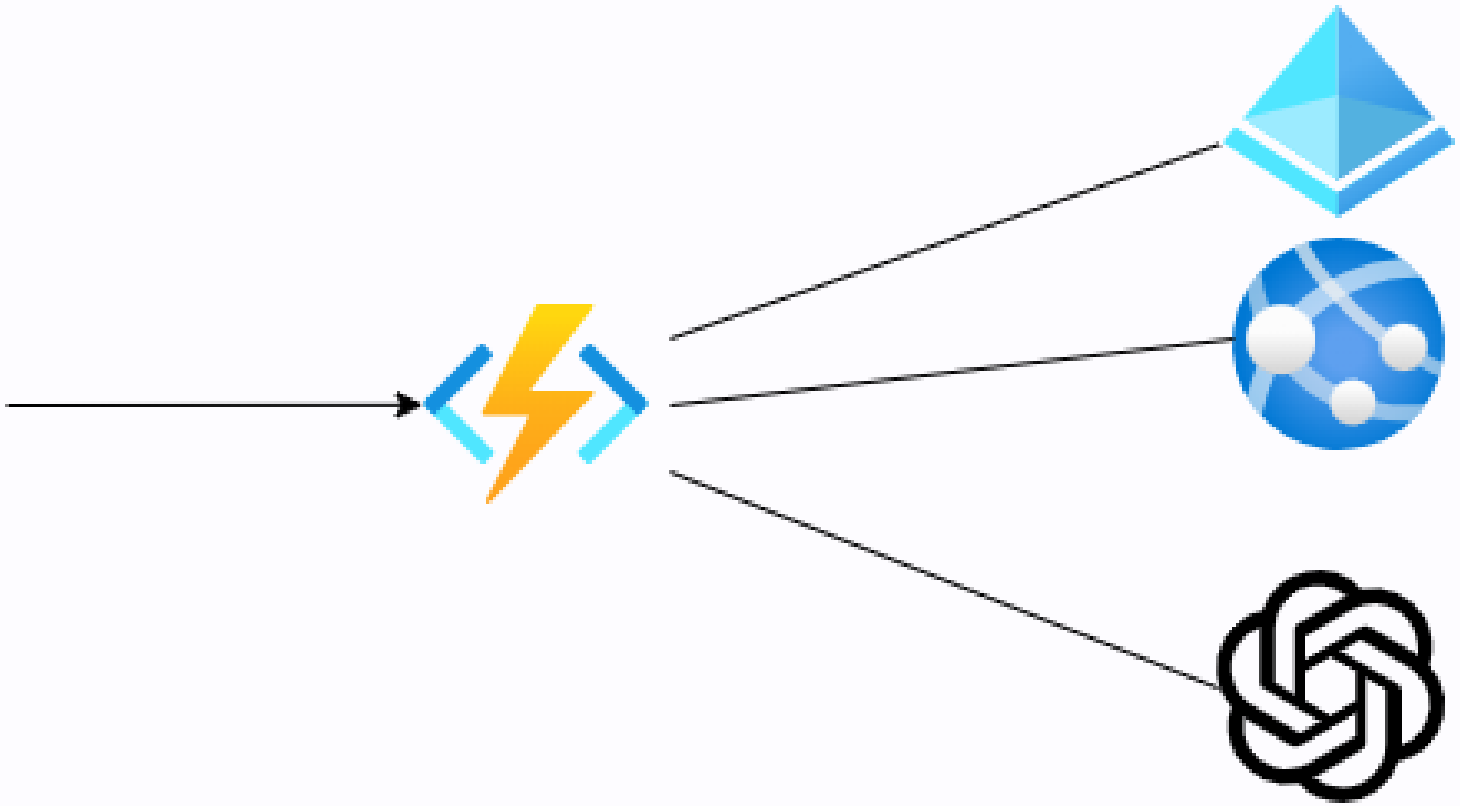

https://cucumber.io/docs/

# Recommendations

- Part of your codebase (can be refactored!)
- Not an exact science
  - Strive for symmetry
  - Hide irrelevant details
  - Consider shortcuts
- If need tests for your tests, you have gone too far

# Serverless

# Implementation on Azure



# Building for Production

- It should run in Production
  - No `if $test`
  - No changes to code to make it "testable"
- Should also run on
  - Several other stages
  - Continuous Integration pipelines
  - Locally for developers

# Problems when Testing

- Developer Experience
  - Needs to be *fast*
- Continuous Integration
  - Might compete for resources
  - Deploy every run to the cloud !?
- External Systems
  - Can't test failure cases with live systems


Cloudless

# Local Development Demo

# Local Replacements

| Azure | Docker |
|---|---|
| Function App | azure-functions (Official) |
| AD, Key Vault, Management, ... | Wiremock |
| Storage Account | Azurite (Official) |
| Cosmos DB | Cosmos DB Emulator (Official), cosmosdb-server |
| Event Grid | azureeventgridsimulator |
| Event Hub | ?, (Kafka) |
| Service Bus | ?, (RabbitMQ) |

# No Silver Bullet

- Infrastructure & Security
  - Deploy to Dev
  - Test Suite for Security
- Integration
  - (E2E) Happy Cases
  - Health Monitoring
- Performance
  - Load Testing

# BDD + Cloudless = ❤️

- You are already doing BDD
  - Language & infrastructure agnostic
  - Ultimate flexibility when refactoring
- Cloudless environments
  - Fast iteration locally
  - Better CI pipelines