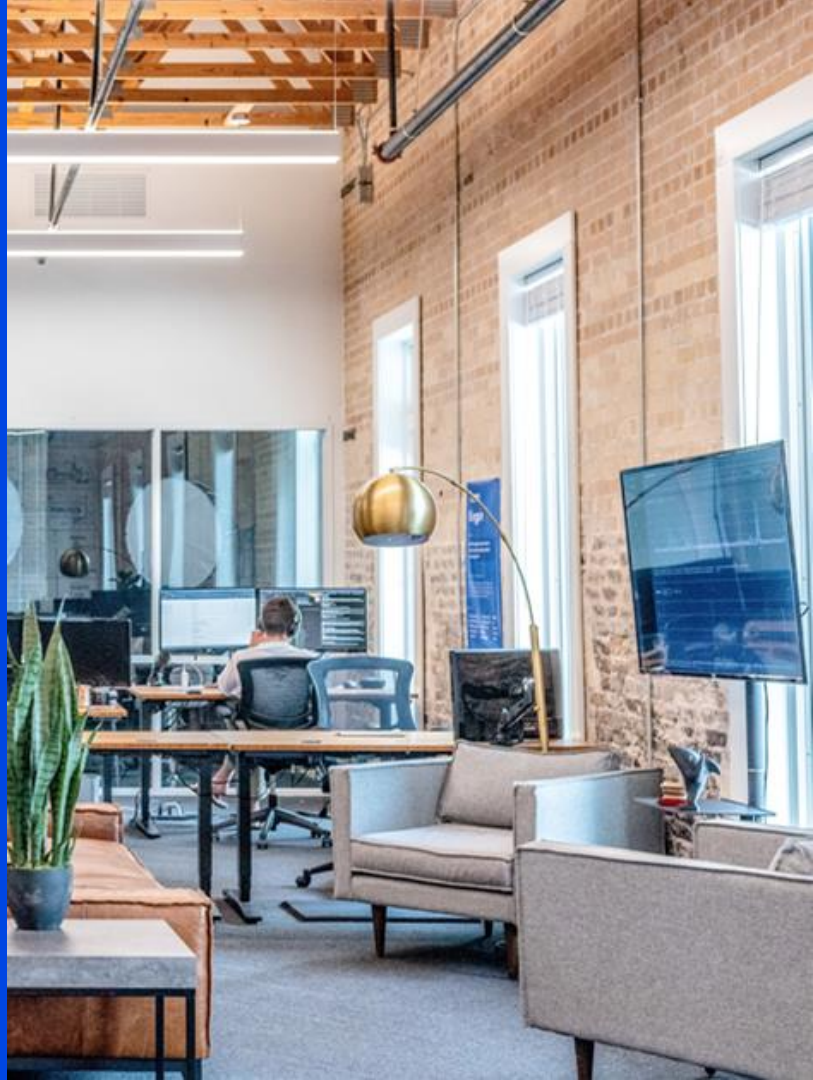
How to test apps in a distributed environment

Piotr Litwinski

zartis



Topics

#

The Problem

#

Types of tests

Build time tests

- Unit tests
- Contract tests
- Service/Component tests (e.g., in-memory integration tests)
- Classic integration tests

Environmental tests

- Deployed
- E2E
- Explaratory

System tests

- Performance
- Resilience / Availability

#

The tools

Developer tools

- In-process tools
  - Stubs & mocks
  - Coarse grained “unit” tests
  - Well defined contracts
  - In-memory testing frameworks (e.g., in-memory databases, test server)
- Resilience / Availability
  - Contract verification
  - Webdriver + (headless) browser
  - Local counterparts (e.g., local sql, json server, etc.)
  - Easily distributable counterparts (e.g. MySql hosted on docker)

Developer tools - Examples

- Coarse grain unit tests
- In-memory database and test server
- WebDriver + HttpServer + Json Server + headless Firefox

Devops tools

- Tools to use after successful deployment
  - Pipeline runs deployed tests on the environment
  - Service runs its own diagnostics on startup

Ops tools

- Pipelines
- Monitoring
- Dashboards

#

How to write a test

- Choose strategy – what is the actual goal?
- Choose methodology – test function/class/component vs scenario testing
- Plan – think before you write anything!
- Modularity – try to use building blocks

How to write a test - Examples

- Scenario testing
- Modularity

#

How to test a test?

- Planning
- Code reviews
- Metrics – keep track of bugs not found / caused by wrong tests
- Test coverage
- Mutation tests

#

Summary

Difficulties & Benefits

Difficulties

- Standardization and automation on a company level
- More complex changes can be more … complex
- Cost (short term) – learning curve, trainings, etc.

Benefits

- Huge savings on regression
- Catching up bugs much earlier
- More confidence while refactoring
- Standardization across company

Questions?

</>

Thank you all

</>

zartis