Testing

Subjective descriptions of how to apply it

devies

Why talk about it?

- Many well known, vaguely defined concepts
  - Unit test
  - Automation
  - E2E test
  - etc.
- Many preferred styles of tests
- Opinions grow on you while applying testing
- Opinions change while applying testing

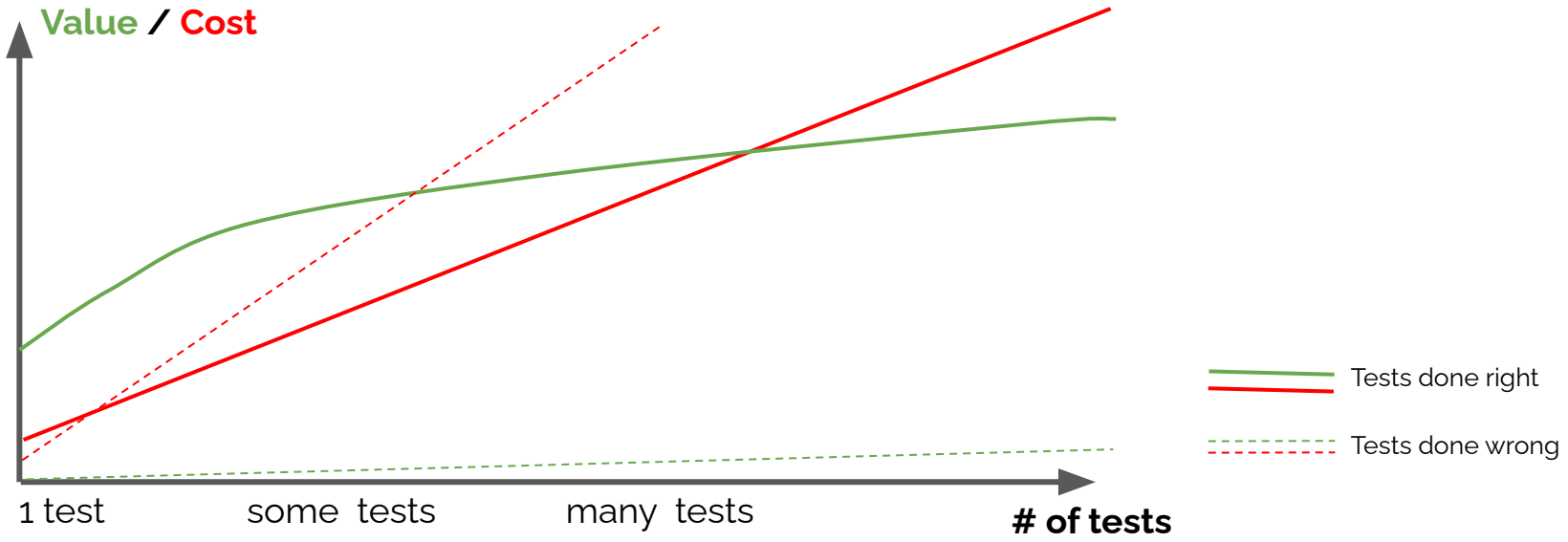
What value does tests provide?

- Only valuable when executed
  - Automatic execution is critical
- Only valuable when acted upon
  - Flaky or constant red tests prevent all value

- Catch issues early
  - Small bug backlog
- Catch issues before customers
- Lower deployment pain
- First test is most valuable

Value VS Cost

- Tests must provide more value than they cost



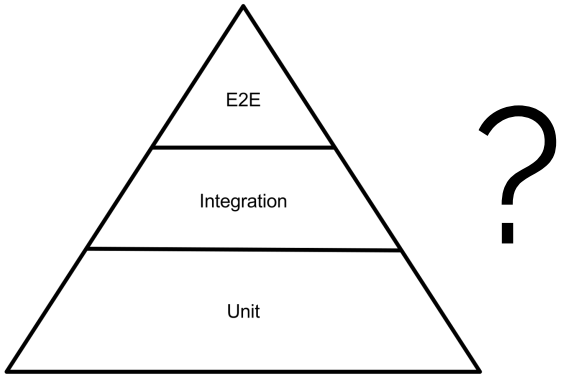
devies

How to control maintenance cost

- Test well defined interfaces
  - They don't change rapidly
  - Small surface between tests and code
  - Tests don't break (as much) when refactoring
- Know what you aim to verify with each test
  - Verify one logical thing (Multiple asserts are OK)
  - Put it in the test name
- Don't tolerate flaky tests
- Small improvements all the time
  - Tests are a living code base, just like production code
- Limit amount of mocking
  - A mock is duplicated functionality
- Naive tests - Allow verbose crash on unexpected state

Types of test

- Is the test pyramid valid?
  - Not really
  - No consensus on what the sections are named or mean
  - Differs based on system complexity
- Focus on what faults you intend to cover with the test

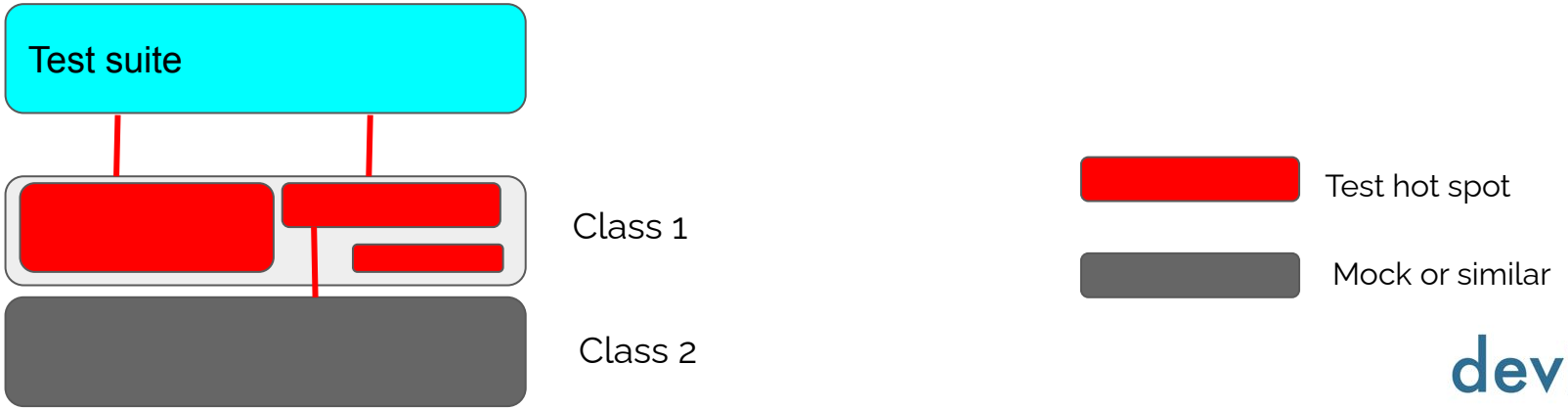


?

devies

Unit tests (ie. testing a software unit)

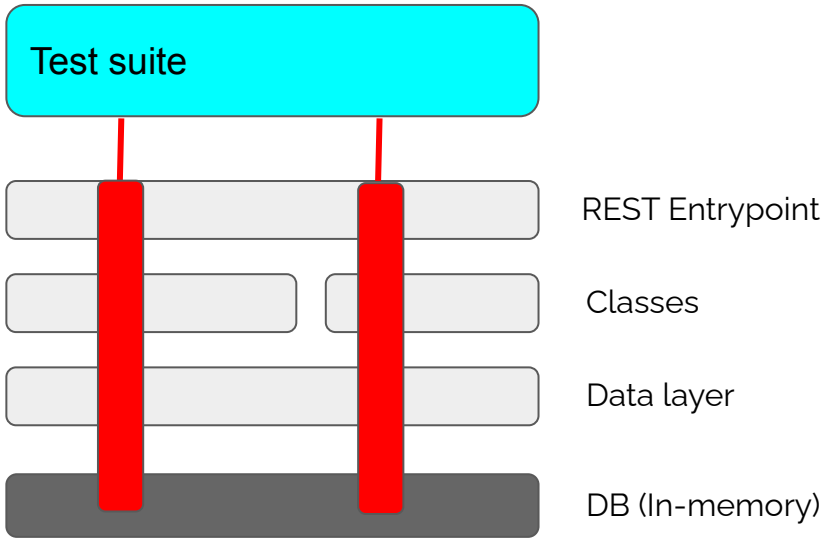
- Do NOT confuse with test frameworks for Unit tests
  - Test frameworks are often labled Unit test frameworks
  - Test frameworks often cover all layers of tests
- Covers detailed domain logic
- Covers most branches
- Useful for slightly trickier or more important code

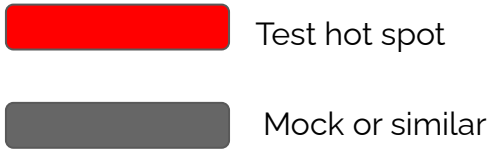


devies

Component test

- Full flow of a single software component (Project)
- Covers primary flows - aims to touch much of the code base
- Will not cover all branches
- Best value / cost

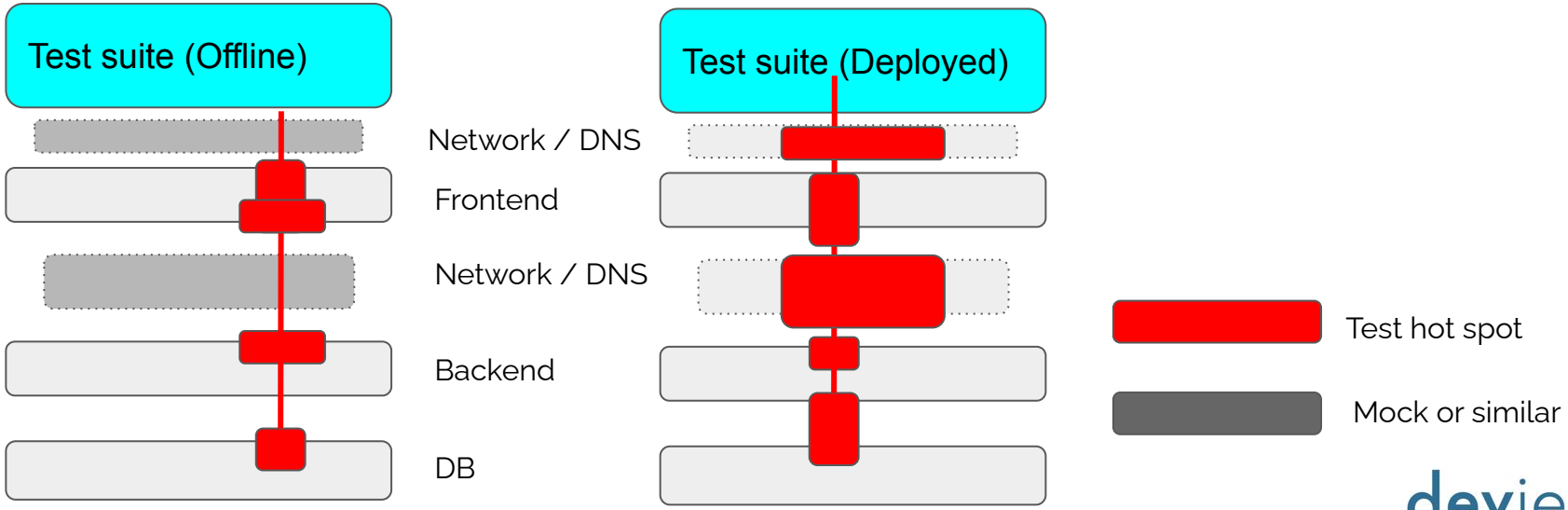




devies

End to End (E2E) test

- Two possible modes : Fake offline environment or toward real deploy
- Covers few primary flows
- Closely resembles real environment
- Focus is on integration between multiple components / networking



devies

Takeaways

- Write your first few tests
- Test stable interfaces
- Start with broad test coverage
- Run the tests automatically

Anything I missed?