Finally, Easy Integration Testing with Testcontainers

Featuring MicroProfile, MicroShed Testing, …

Rudy De Busscher

payara®

Testcontainers

- Testing, really?
- Testcontainers features and how it works
- Demos
  - Basic
  - With MicroShed Testing
  - Very advanced interactions

payara®

Rudy De Busscher

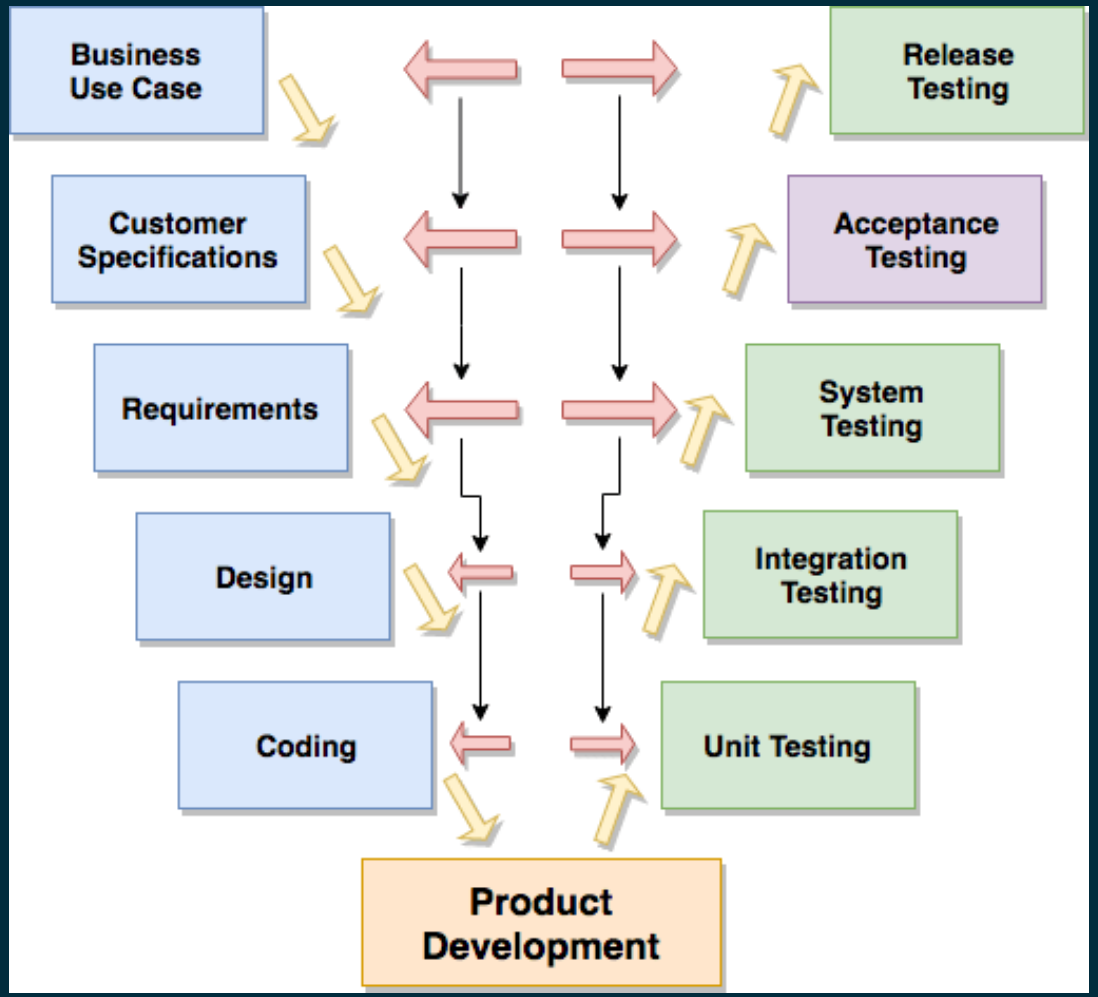
- Payara
  - Service team
- Involved in
  - Committer of MicroProfile
  - Committer in Eclipse EE4J groups
  - Java EE Security API Expert group member

@rdebusscher

https://blog.payara.fish/
https://www.atbash.be

payara®

Types of Testing - Classic



- **Release**: Flawless and works as expected
- **Acceptance**: tested for acceptability
- **System** : Complete and *fully integrated software* product.
- **Integration** : individual units are combined and *tested as a group*.

payara®

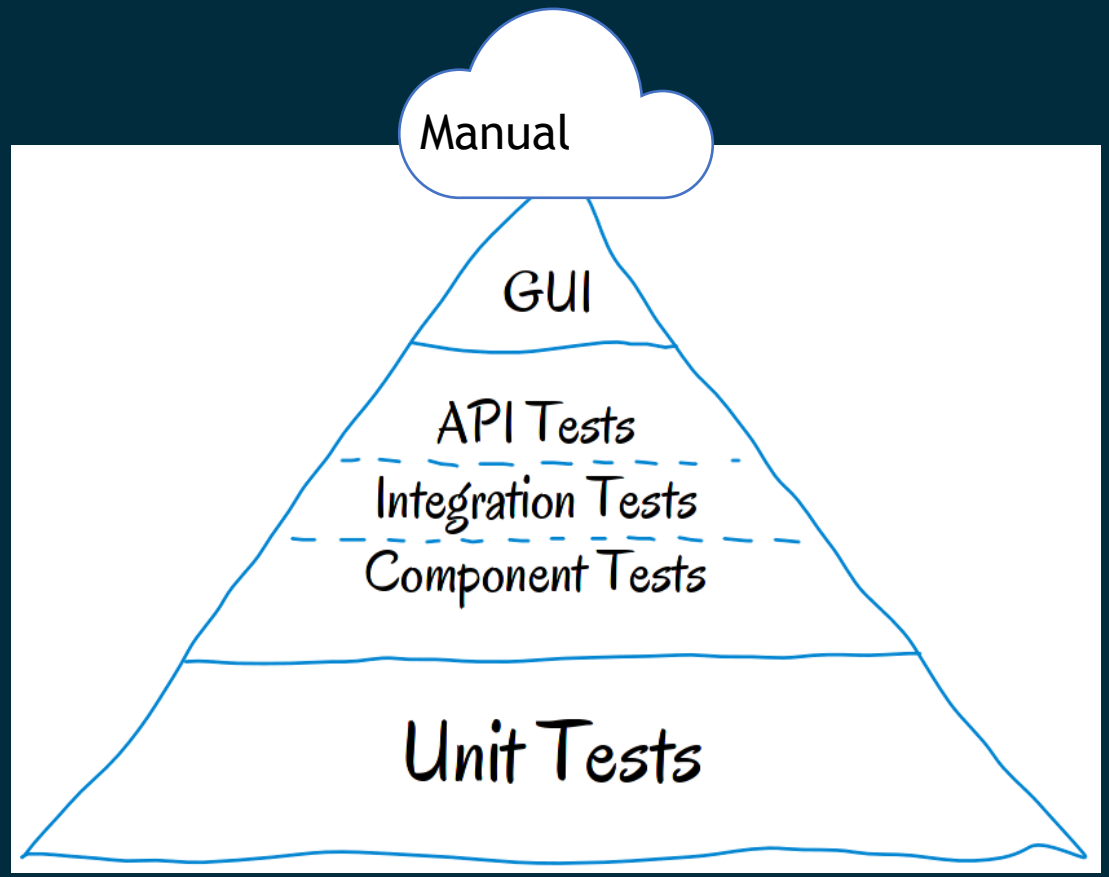
Unit vs Integration Testing

- **Unit Testing**: Individual methods
  -  • Fast, Easy
  -  • Business relation
- **Integration testing** : Multiple components, systems, …
  -  • Realistic
  -  • Slow, Brittle, …

payara®

Testing

- 10%
- 20%
- 70%



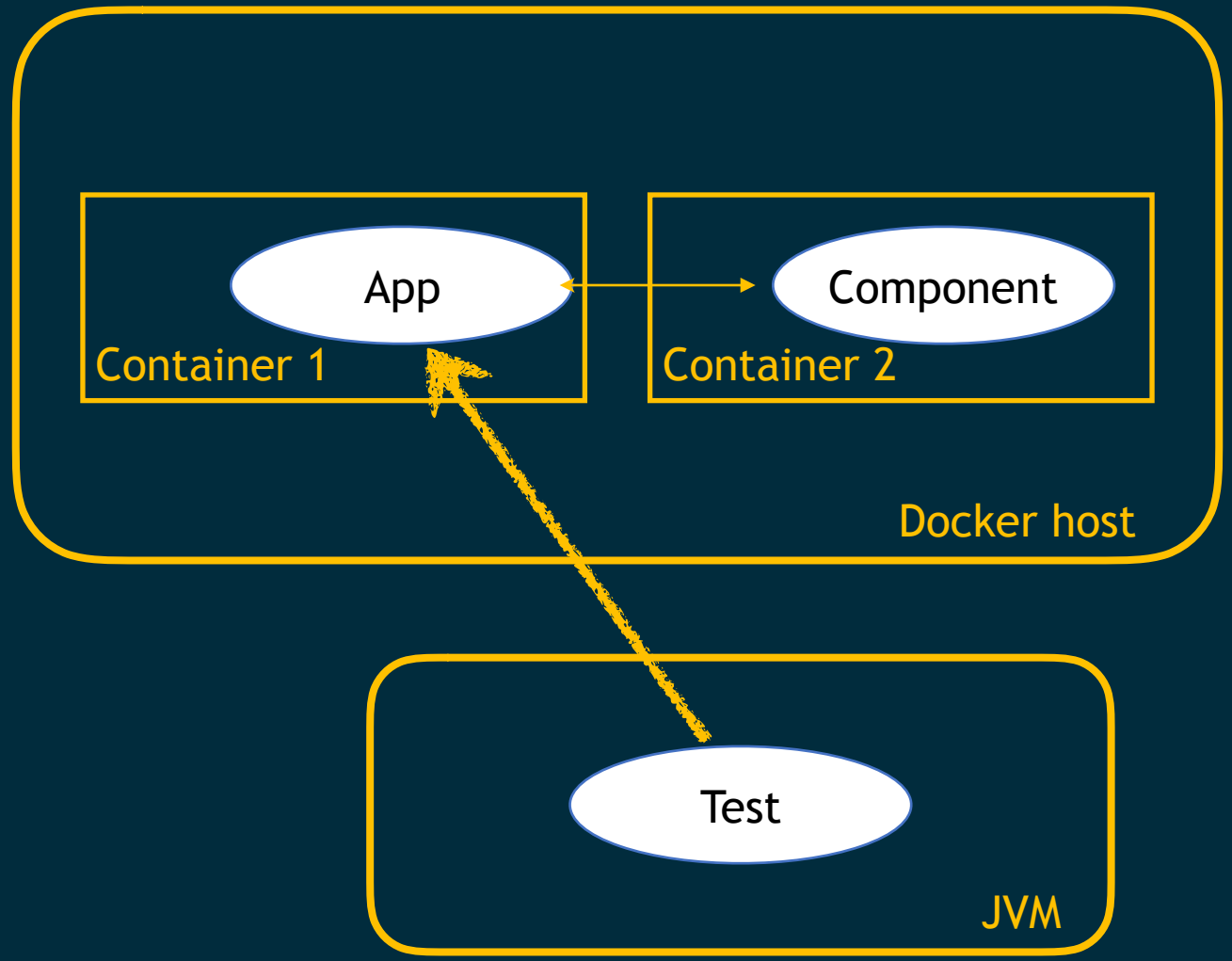
Integration Testing issues

Testcontainers

- Components run in Docker Container.
- Easy Integration of components.
- No limitations on external dependencies.
- Complete control over components in test.

payara®

Testcontainers - Overview



Testcontainers - Usages

- Repeatable tests using real systems
  - Database, monitoring, …
- Selenium to verify frontend of your application
- Simulating cluster on single computer
- Simulating network issues
- Manual testing / debugging realistic scenarios

payara®

Testcontainers - Example

```
@Testcontainers
public class TypicalTest {

    @Container
    private static PostgreSQLContainer postgresqlContainer = new PostgreSQLContainer()
            .withDatabaseName("foo")
            .withUsername("foo")
            .withPassword("secret");

    @Container
    private static GenericContainer<?> applicationContainer = new GenericContainer<>( dockerImageName: "my-app:v1")
            .waitingFor(Wait.forHttp( path: "/health"));

    @Test
    public void endpoint() throws IOException {
        String url = String.format("http://%s:%s/endpoint"
                , applicationContainer.getContainerIpAddress()
                , applicationContainer.getMappedPort( originalPort: 8080));
        String data;
        try (InputStream in = new URL(url).openStream()) {
            data = new Scanner(in,  charsetName: "UTF-8").useDelimiter("\\A").next();
        }
        assertThat(data).
    }
```

Testcontainers - Features

- Starts / configure Containers
- Wait strategy (http status, Log entry, …)
- Predefined containers (Database, Kafka, Selenium, …)
- Containers destroyed after test / JVM exit
- Highly extensible (JUnit 5)

Demo

- Plain Basic Testcontainer example
- MicroProfile - MicroShed testing
- Multiple services
- Advanced
  - Debugging
  - Clustering

payara®

Q & A

Thank You

Download the open source software:
https://payara.fish/downloads

Need support for the Payara Platform?
https://payara.fish/support

payara®