

<epam>

Building Integration Test Environment Using Docker Containers

Aliaksei Harshkalep





ALIAKSEI HARSHKALEP

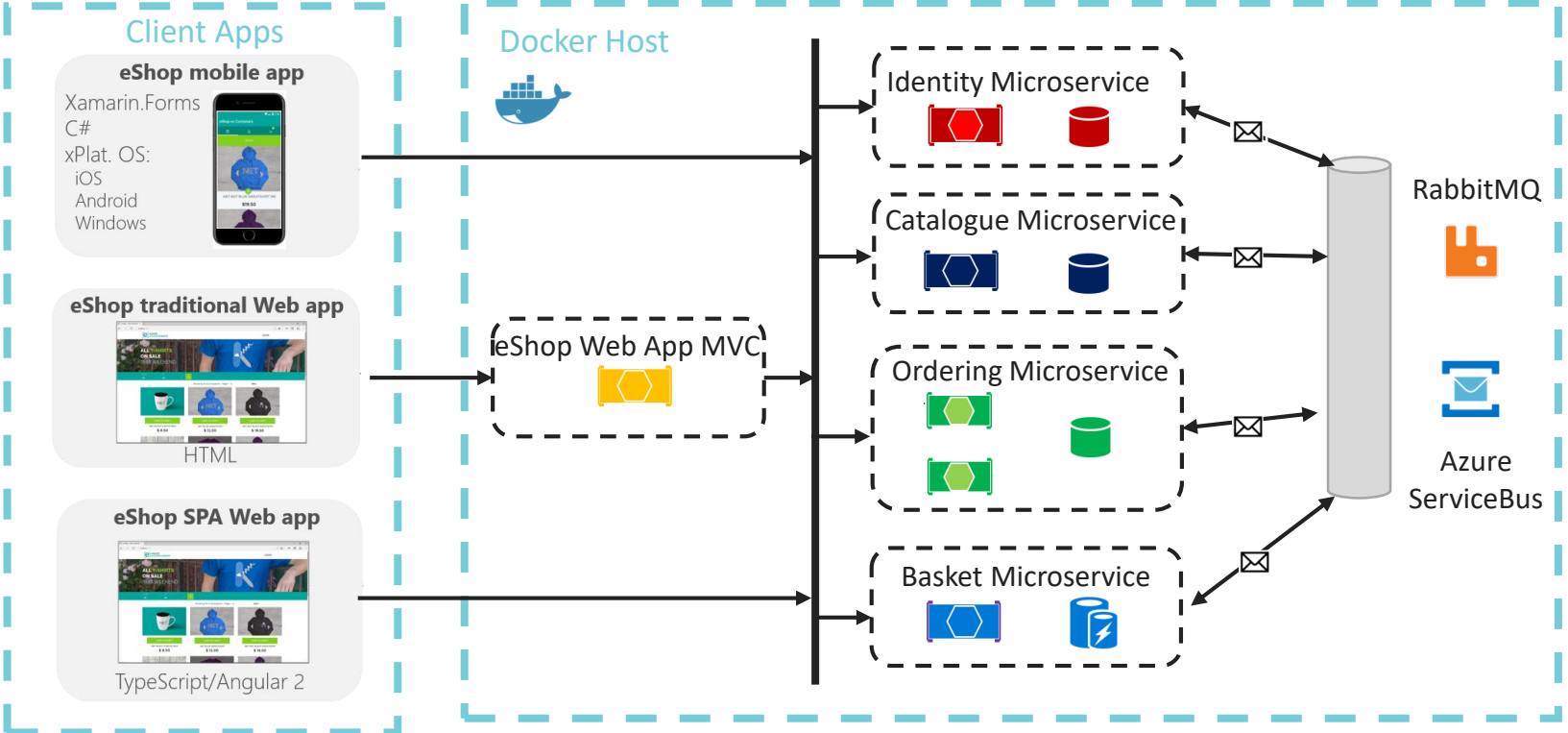
Lead Software Developer, EPAM Gomel

- 5+ years in Software Engineering
- .NET, .NET Core and ASP.NET Core
- Microservice-Oriented Architecture

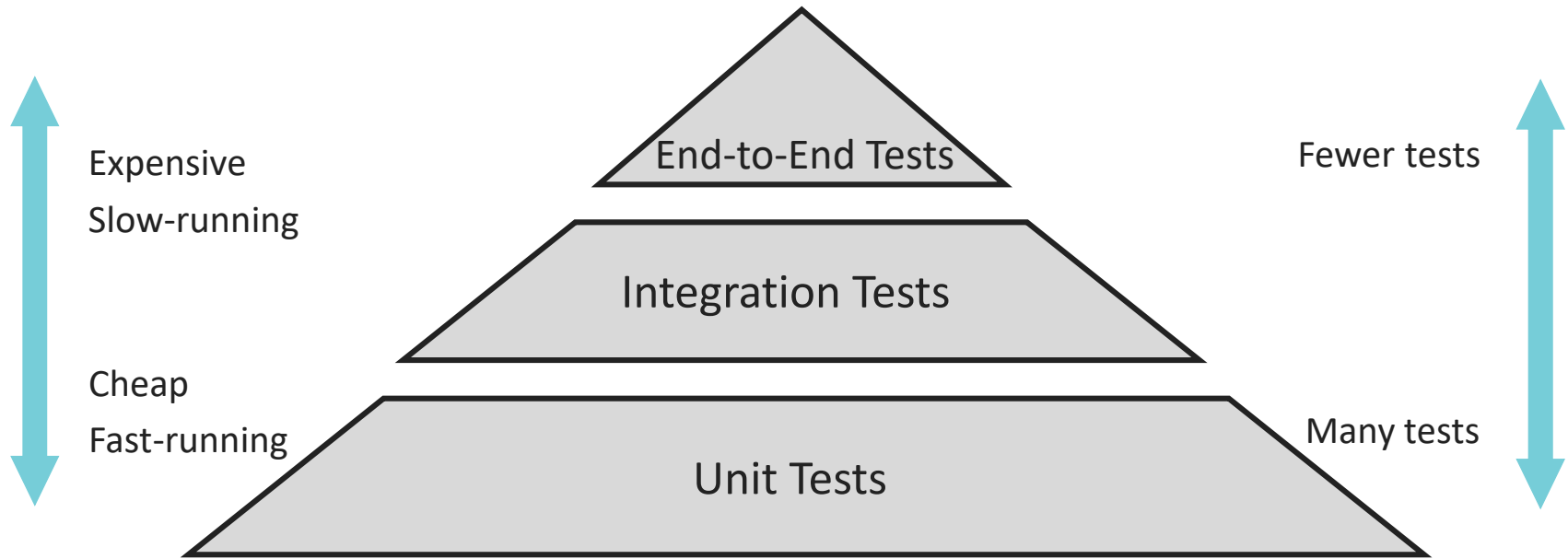
Agenda

- Testing Approaches
- Integration Testing Complexity
- Docker as a Testing Environment
- Demos

Typical Microservice Application



Testing Pyramid



Unit Tests or End-to-End Tests?

UNIT TESTS

- Too Much Time on Data Mocks
- Integration Bugs
- *“We need 100% Unit Test coverage!”*
- Green Tests != Working Application

END-TO-END TESTS

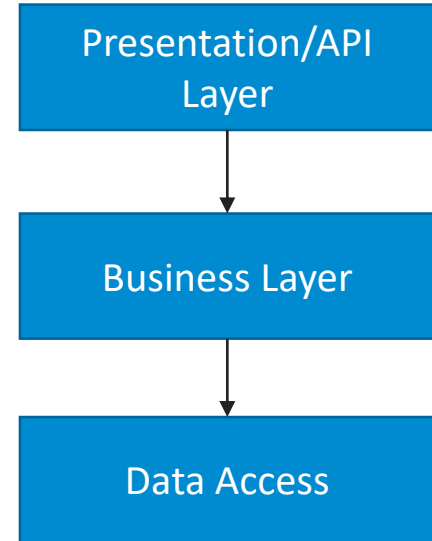
- Too Slow
- Extra efforts to run locally and debug
- Granularity Problem
- Developer or QA?



Integration Testing: Benefits

YOU CAN TEST WITH DIFFERENT GRANULARITY

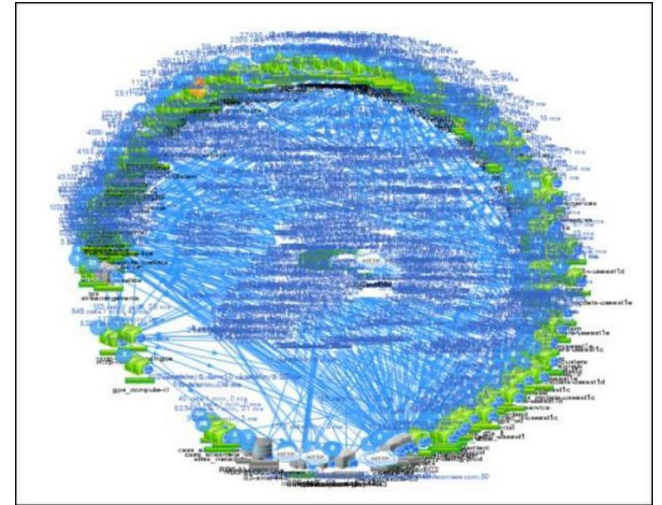
- Data Access Functionality
- Single Component or Business Service
- Multiple Components and their Interaction
- API with all Infrastructure



Integration Testing: Benefits

INTEGRATION TESTS FOR API/MICROSERVICE

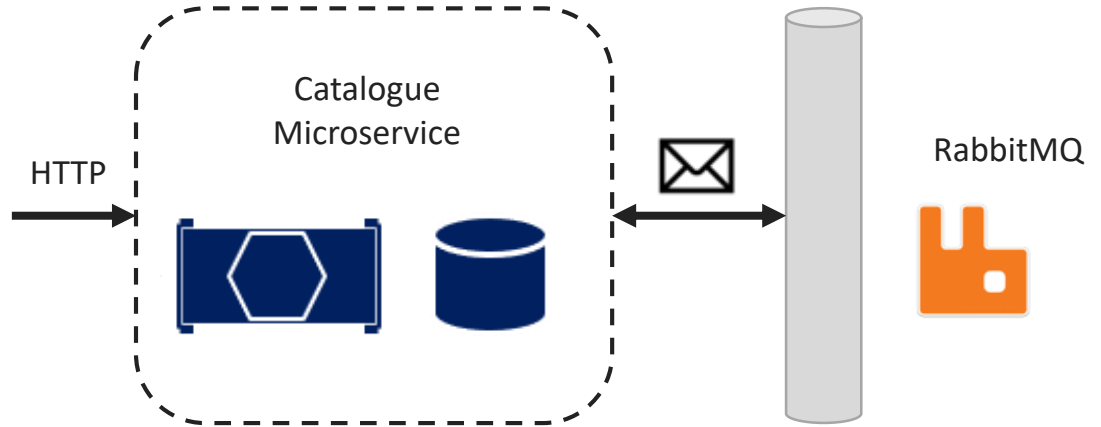
- Given/When/Then acceptance criteria
- Safe refactoring
- Integration bugs detection (DI, Serialization)
- Data Contract and Versioning Testing (crucial for Microservice Architecture)



Microservice Testing

MICROSERVICE TESTING CHALLENGES

- Host the service on HTTP Server
- Setup repeatable environment – DB, Message Broker, Caching and etc.



In-Memory Test Server

- [Microsoft.AspNetCore.Mvc.Testing](#)
- Fast – it does not start a real server
- Reliable – no needs to reserve ports or clean up resources after it runs
- Easier than external test driver

```
[Fact]
0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
public async Task CanGetHomePage()
{
    // Arrange
    var webHostBuilder = WebHost.CreateDefaultBuilder()
        .UseStartup<Startup>();
    var server = new TestServer(webHostBuilder);
    var client = server.CreateClient();

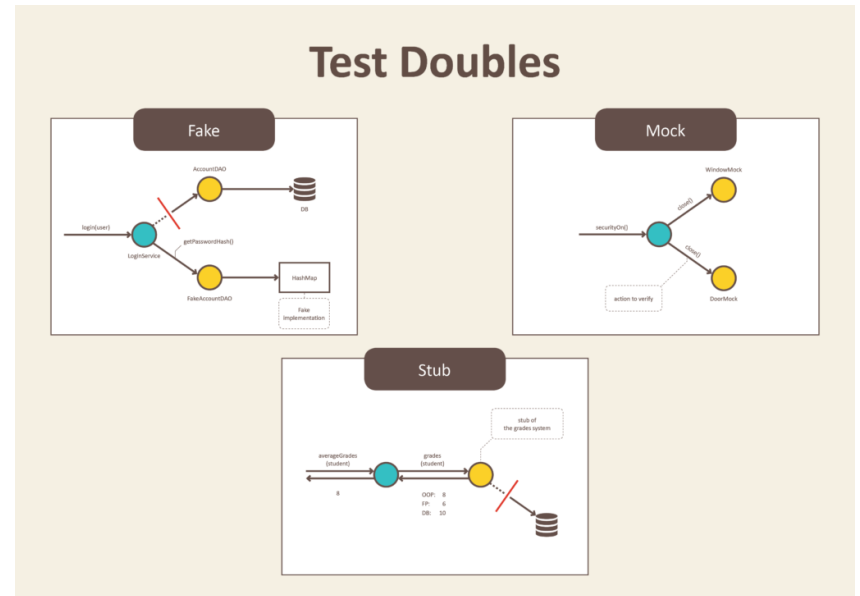
    // Act
    var response = await client.GetAsync("/");

    // Assert
    Assert.Equal(HttpStatusCode.OK, response.StatusCode);
}
```

Test Dependencies

WAYS TO SOLVE TEST DEPENDENCIES

- Test Doubles – Mocks, Stubs, Spies and etc.
- Software Provisioning on Testing Machine
- Containerization



What is Docker?

Docker Containers

Docker improves the deployment of applications with portable, self-sufficient containers, Linux or Windows, that can run on any cloud or on-premises.

No more:

"It works in my dev machine!...

Why not in production?"



Now it is:

"If it works in Docker, it works in production"

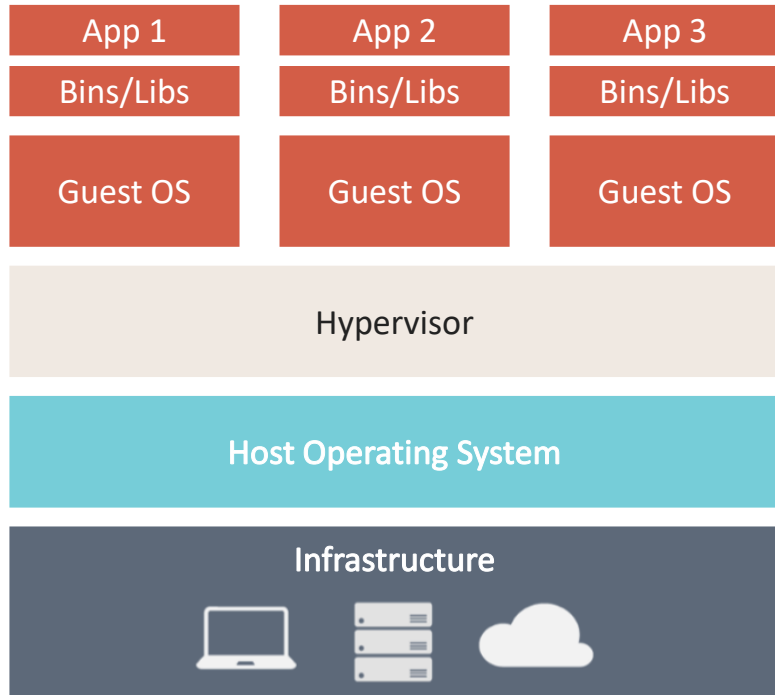
Keywords about WHY Docker?

- **Dependencies (self-sufficient)**
- **Deployment**

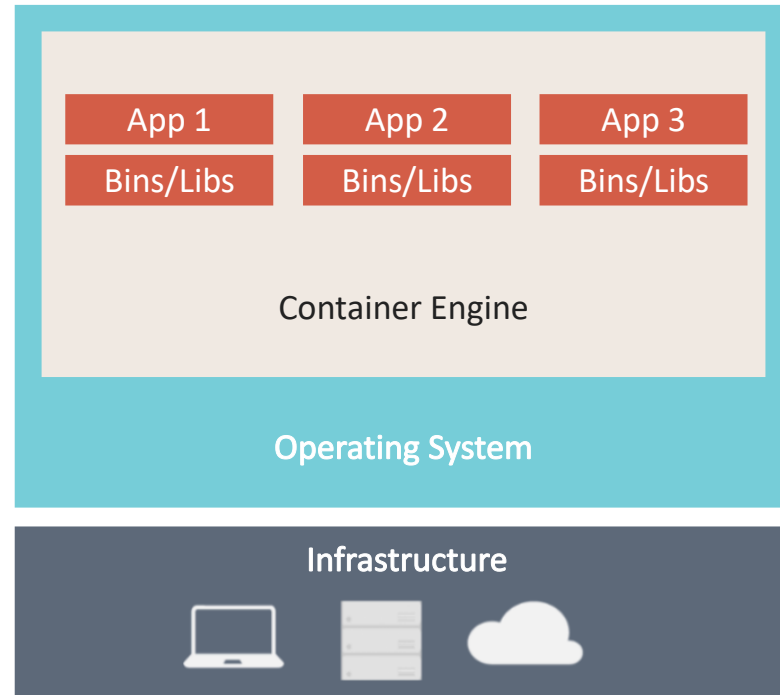


VMs vs Docker Containers

VIRTUAL MACHINES



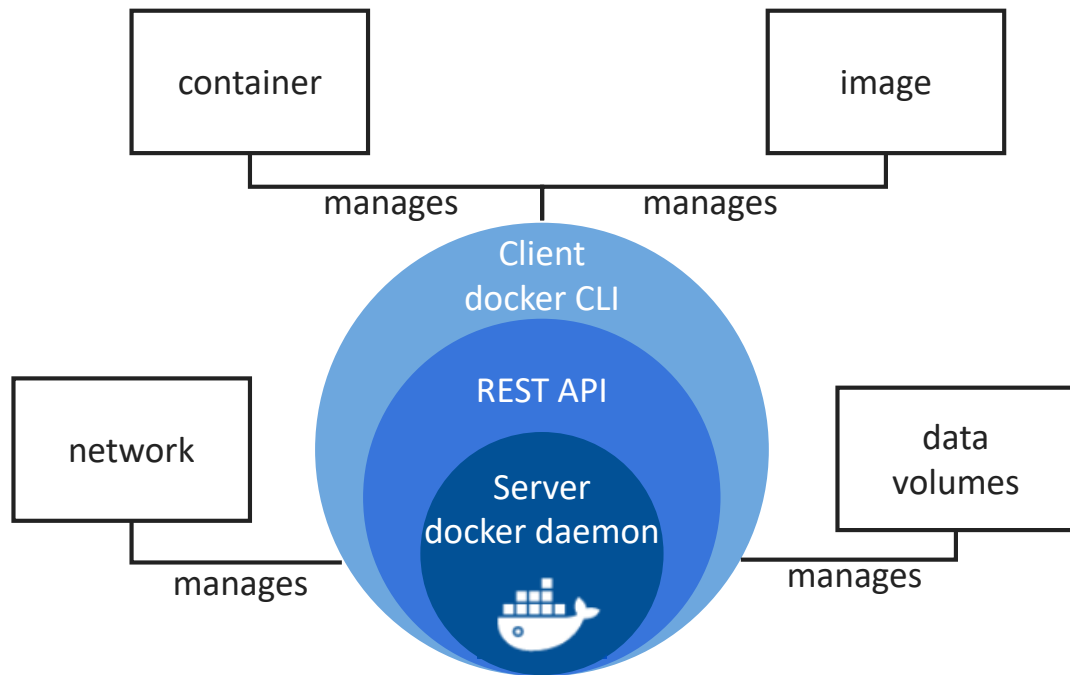
DOCKER CONTAINERS



Docker Architecture

DOCKER MAJOR PARTS

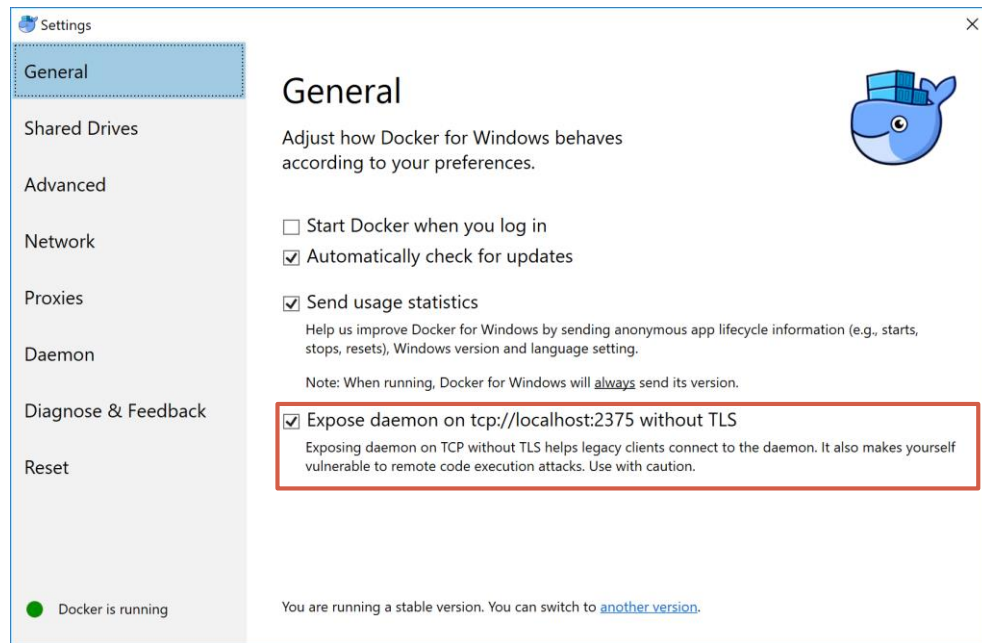
- Daemon Process
- A REST API
- A command line interface (CLI)



Docker API

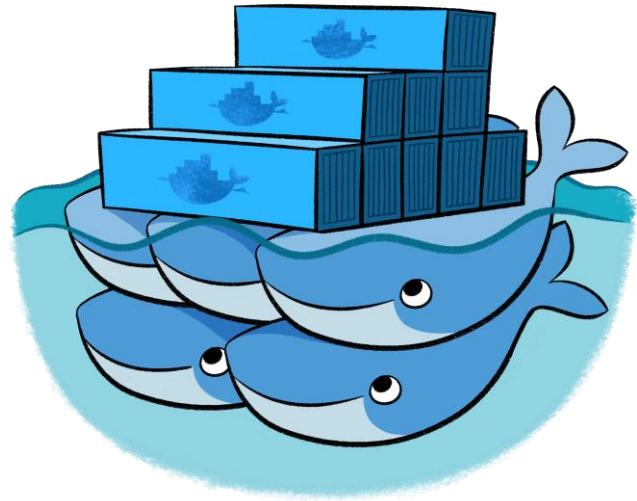
TO CONSUME DOCKER API

- Full [documentation](#) is available
- Default URIs:
<unix:///var/run/docker.sock>
npipe://./pipe/docker_engine
- Use C#, Java, Python or whatever language libraries



How to Use Docker API from C#?

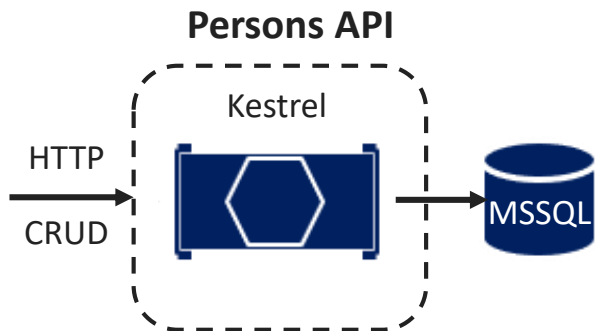
- Install [Docker.DotNet](#)
- Get Docker URL
- Use DockerClient Methods to:
 - *List Containers*
 - *Pull Images*
 - *Create Containers*
- Connect to newly created container



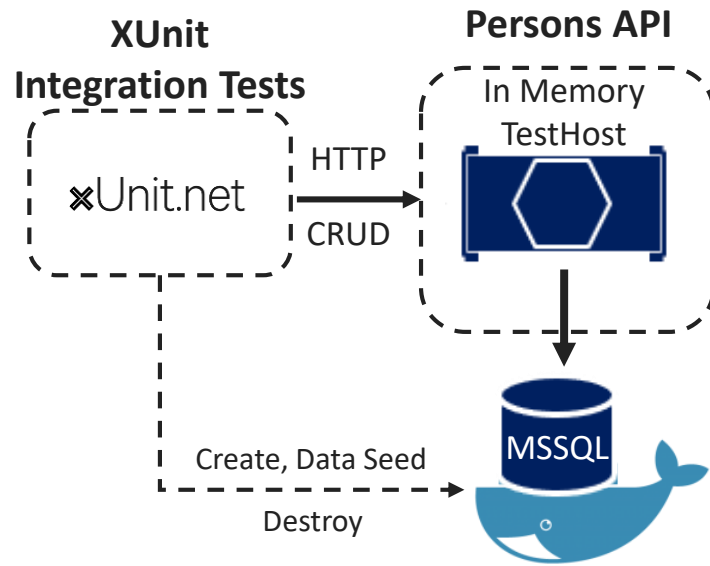
**DEMO 1:
RUN MSSQL CONTAINER WITH
DOCKER.NET**

Microservice Testing with Docker

PERSONS MICROSERVICE TO TEST



TESTING WITH XUNIT AND DOCKER

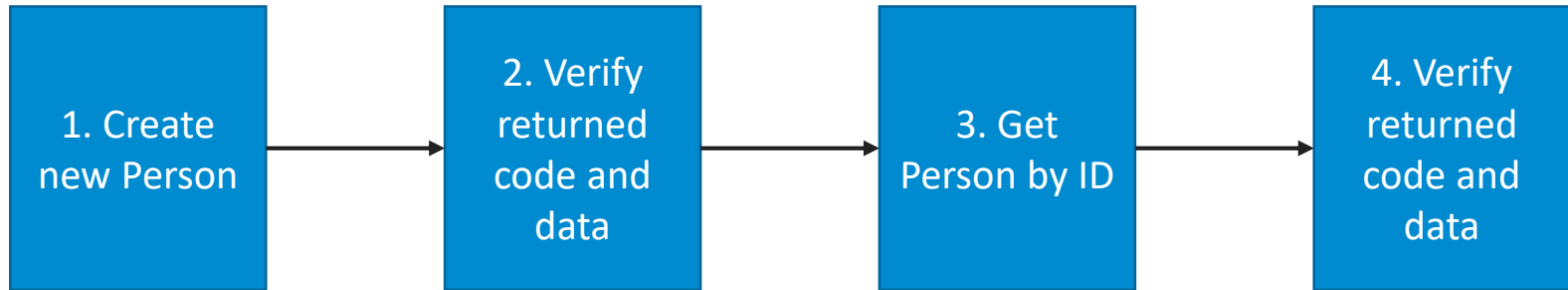


Technology List?

- **ASP.NET Core + MVC** – for Web API
- **EF Core + MSSQL Server** – for storing the state
- **Swagger UI** – for API exploring
- **XUnit** – for testing
- **Microsoft.AspNetCore.Mvc.Testing** – in memory Test Host
- **Docker.DotNet** – for setting up testing environment

Sample API Testing Scenario

CREATEPERSON_WHENPOSTVALIDDATA_SHOULDRETURNCREATED



DEMO 2: MICROSERVICE TESTING

TestEnvironment.Docker: Motivation

- Reusability
- Enrich with features
- Readiness problems
- Make it CI/CD Friendly (GitLab)
- Share with Community



TestEnvironment.Docker

<https://ci.appveyor.com/project/Deffiss/testenvironment-docker>

Deffiss / testenvironment-docker

No description, website, or topics provided.

Manage topics

54 commits 2 branches 0 releases 2 contributors

Branch: master	New pull request	Create new file	Upload files	Find file	Clone or download
Deffiss Update README.md					Latest commit 24713e 15 days ago
src					Move MSSQL and Elasticsearch specific containers to separate projects. 17 days ago
test/TestEnvironment.Docker.Tests					Move MSSQL and Elasticsearch specific containers to separate projects. 17 days ago
.gitignore					Initial commit. 4 months ago
README.md					Update README.md. 15 days ago
TestEnvironment.sln					Move MSSQL and Elasticsearch specific containers to separate projects. 17 days ago
appveyor.yml					Fix appveyor error. 17 days ago

Test environment with Docker containers

Pre requirements

You need docker to be installed on your machine. Tested both on Windows and Linux.

Installation

AppVeyor

testenvironment-docker

Current build History

Update README.md 1:00:59

15 days ago by Alkesh Harshikalp (submitted by GitHub) master · 24713e2

Build started

```
1 git clone -q --branch=master https://github.com/Deffiss/testenvironment-docker.git /home/appveyor/projects/testenvironment-docker
2 git checkout -f 24713e25704995097140856d0219469753719
3 Running 'build' scripts
4 Starting Docker
5 dotnet build TestEnvironment.sln -c:Release
6
7 ASP.NET Core
8
9 Successfully installed the ASP.NET Core HTTPS Development Certificate.
10 To trust the certificate run 'dotnet dev-certs https --trust' (Windows and macOS only). For establishing trust on other platforms refer to the platform specific documentation.
11 For more information on configuring HTTPS see https://go.microsoft.com/fwlink/?linkid=848954.
12 Microsoft (R) build engine version 15.8.16646e46d1808 for .NET Core
13 Copyright (C) Microsoft Corporation. All rights reserved.
14
15 Restoring packages for /home/appveyor/projects/testenvironment-docker/src/TestEnvironment.Docker.Containers.Mssql.csproj...
16 Restoring packages for /home/appveyor/projects/testenvironment-docker/src/TestEnvironment.Docker.Containers.Elasticsearch/TestEnvironment.Docker.Containers.Elasticsearch.csproj...
17 Installing Docker.DotNet 3.125.2.
18 Installing Elasticsearch.Net 6.2.0.
19 Installing NEST 6.2.0.
20 Generating MSBuild file /home/appveyor/projects/testenvironment-docker/src/TestEnvironment.Docker.Containers.Elasticsearch/TestEnvironment.Docker.Containers.Elasticsearch.csproj.magnet.g.props.
21 Generating MSBuild file /home/appveyor/projects/testenvironment-docker/src/TestEnvironment.Docker.Containers.Mssql/TestEnvironment.Docker.Containers.Mssql.csproj.magnet.g.props.
```

nuget Packages Upload Statistics Documentation Downloads Blog

Search for packages...

TestEnvironment.Docker 1.1.0

Testing framework for setting up real dependencies as Docker containers.

Package Manager .NET CLI Paket CLI

PM: Install-Package TestEnvironment.Docker -version 1.1.0

Info

- last updated 17 days ago
- Project Site
- Source Code
- Contact owners
- Report
- Download Package (21.84 KB)

Statistics

- 409 total downloads
- 93 downloads of latest version
- 16 downloads per day (avg)
- View full stats

Owners

- deffiss
- Alkesh Harshikalp

Authors

Alkesh Harshikalp

Release Notes

Move container specific functionality into separate nugets (TestEnvironment.Docker.Containers.Elasticsearch, TestEnvironment.Docker.Containers.Mssql).

> Dependencies

∨ Version History

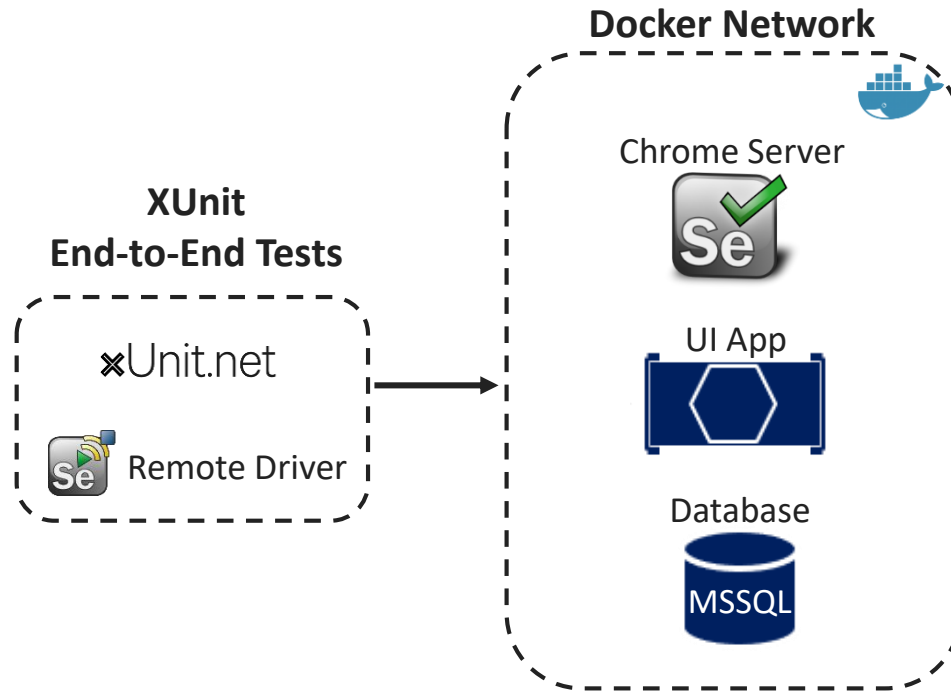
Version	Downloads	Last updated
1.1.0	93	17 days ago
1.0.6	104	19 days ago
1.0.5	80	22 days ago
1.0.4	35	23 days ago
1.0.3	23	23 days ago

<https://github.com/Deffiss/testenvironment-docker>

<https://www.nuget.org/packages/TestEnvironment.Docker>

**DEMO 3:
TESTENVIRONMENT.DOCKER**

End-to-End-Testing with Docker

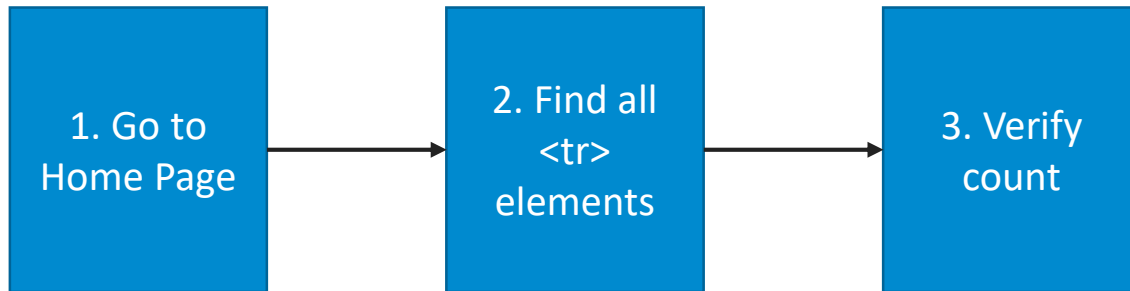


Technology List?

- ASP.NET Core + Razor Pages – for Web UI
- EF Core + MSSQL Server – for storing the state
- XUnit – as test engine
- Selenium.WebDriver + Selenium Server (Chrome) – for interacting with Browser
- TestEnvironmentet.Docker – for setting up testing environment and application

Sample UI Testing Scenario

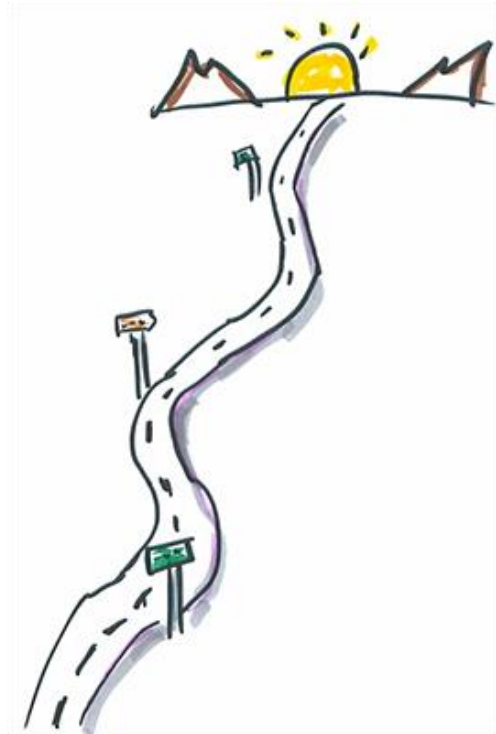
HOMEPAGE_WHENGETHOMEPAGE_SHOULDRETURTABLE



DEMO 4: END-TO-END TESTS

Roadmap

- Configurability (.yaml)
- Dependencies Between Containers
- Containers from Dockerfile
- More Built-in Containers Support (MYSQL, MongoDB, Redis, RabbitMQ)
- ~~• Container Cleanup and Work with Volumes~~
- Test Frameworks Integration (XUnit, MSTest)
- End-to-End Testing



Alternatives?

TESTENVIRONMENT.DOCKER

GitHub Repo:

<https://github.com/Deffiss/testenvironment-docker>

Nuget Package:

<https://www.nuget.org/packages/TestEnvironment.Docker>

- Cross Platform & CI/CD Friendly
- Container Specific Functionality
- Readiness Check
- Single Point of Control

TESTCONTAINERS (.NET)



GitHub Repo:

<https://github.com/testcontainers/testcontainers-dotnet>

Nuget Package:

<https://www.nuget.org/packages/TestContainers>

- Port of Popular Java Package
- Cross Platform & CI/CD
- Readiness Check
- More Built-in Containers
- Community

Summary

- Real Environment
- -> 0 Manual Efforts
- Run in 1 click
- Fast (40-45 secs on my project)
- CI/CD Friendly

THANK YOU!
QUESTIONS?