<epam>

Building Integration Test Environment Using Docker Containers

Aliaksei Harshkalep

ALIAKSEI HARSHKALEP

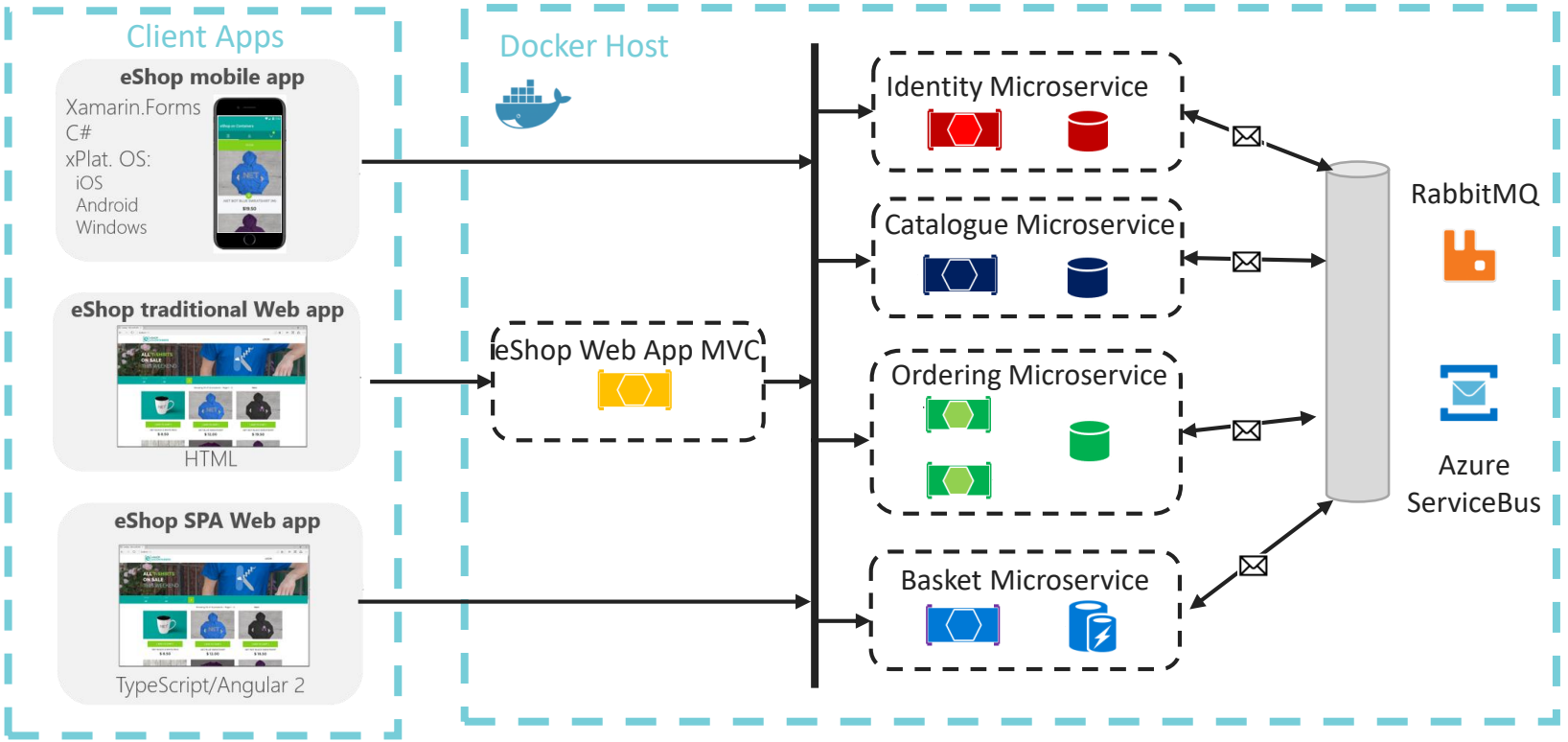
Lead Software Developer, EPAM Gomel

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

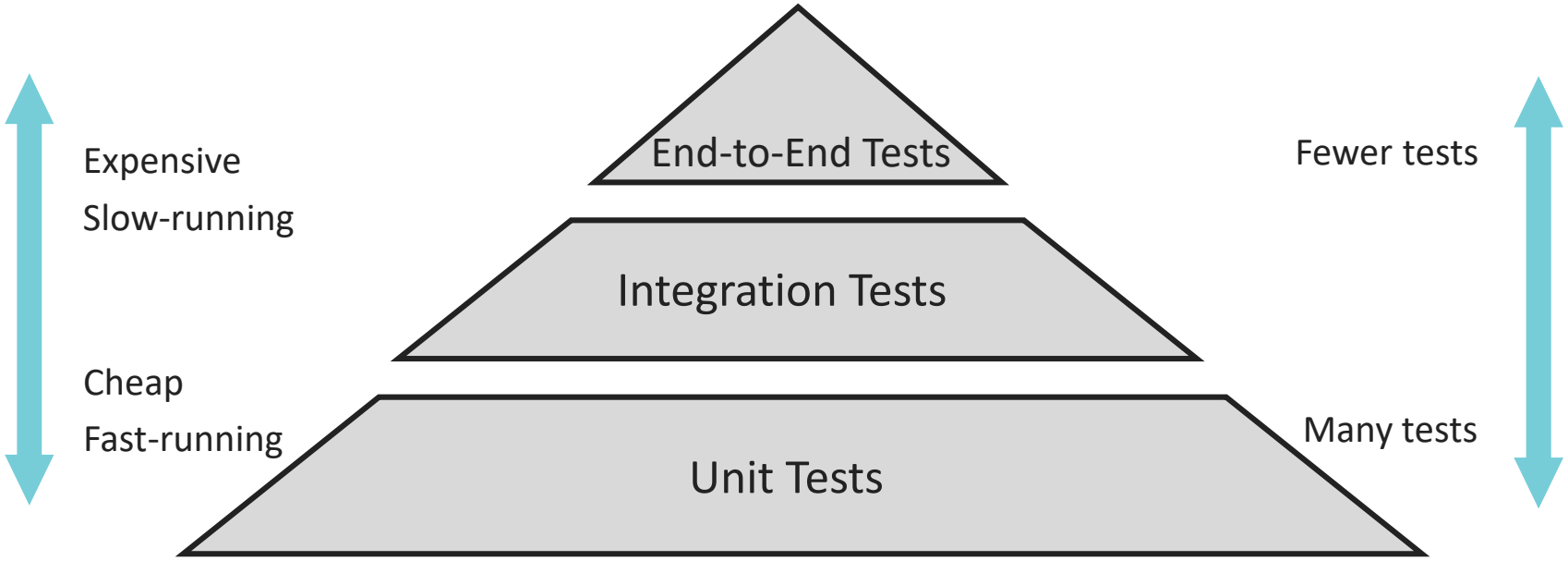
Agenda

- 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

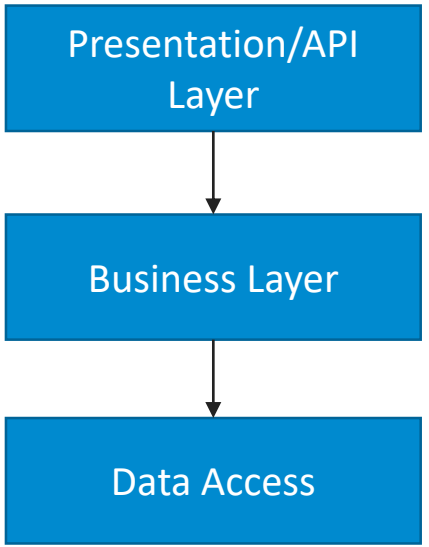
END-TO-END TESTS

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

Integration Testing: Benefits

YOU CAN TEST WITH DIFFERENT GRANULARITY

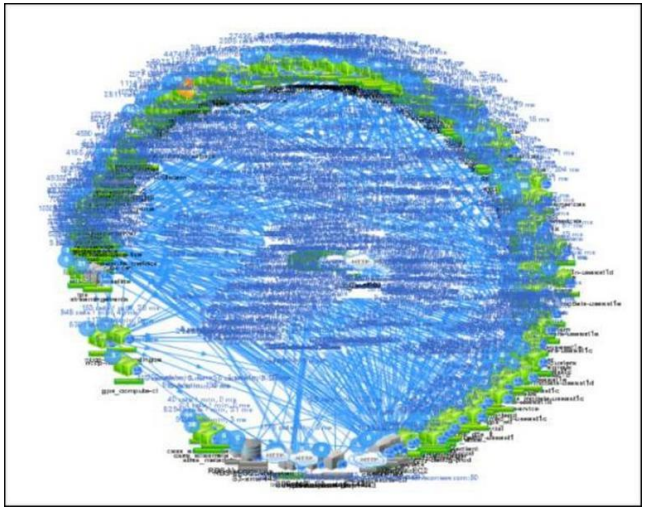
- Data Access Functionality
- Single Component or Business Service
- Multiple Components and their Interaction
- API with all Infrastructure



Integration Testing: Benefits

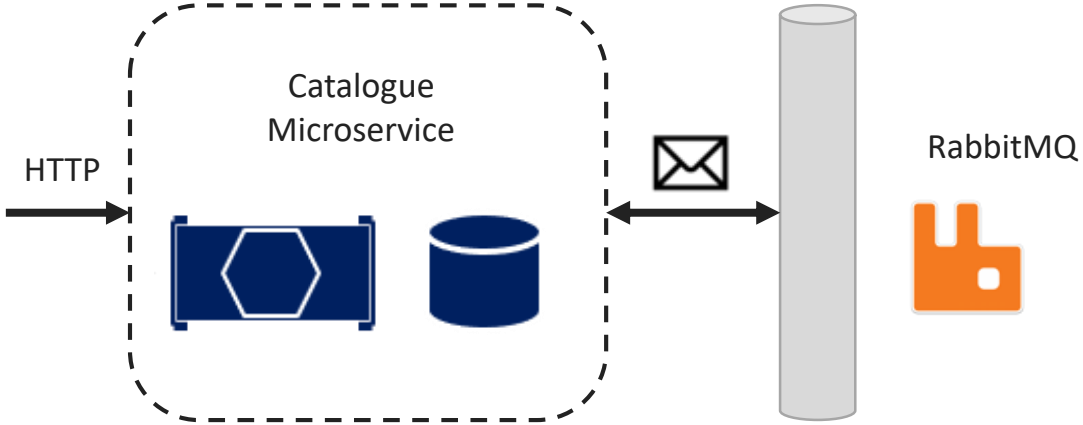
INTEGRATION TESTS FOR API/MICROSERVICE

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



MICROSERVICE TESTING CHALLENGES

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



In-Memory Test Server

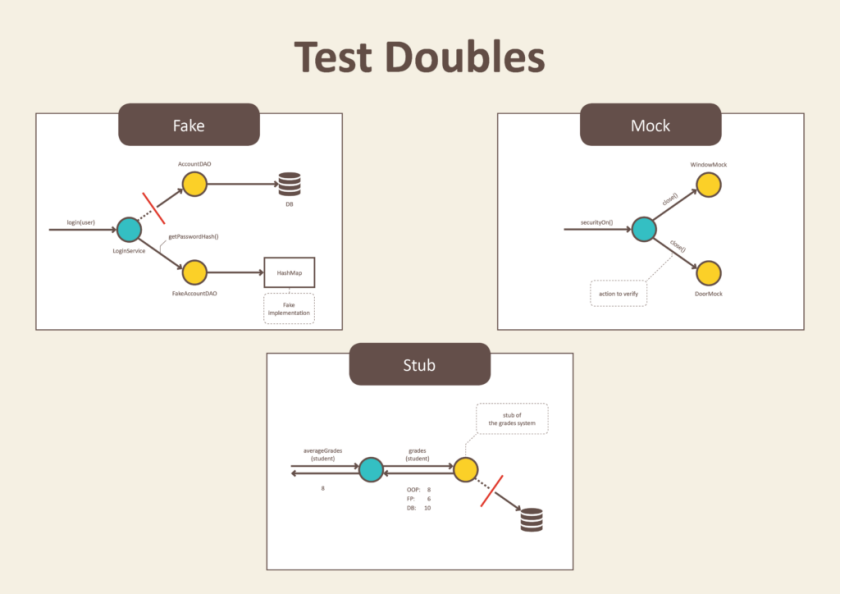
- Microsoft.AspNetCore.Mvc.Testing
- Fast – it does not start a real server
- Reliable – no needs to reserve ports or clean up resources after it runs
- Easier than external test driver

```
[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?

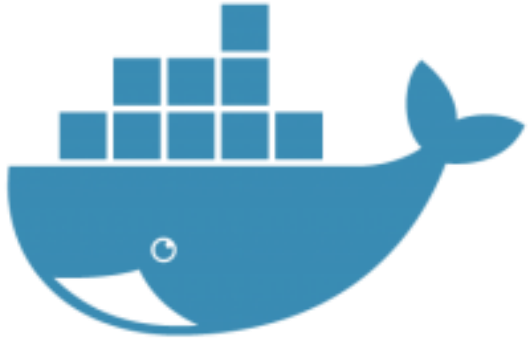
Docker Containers

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

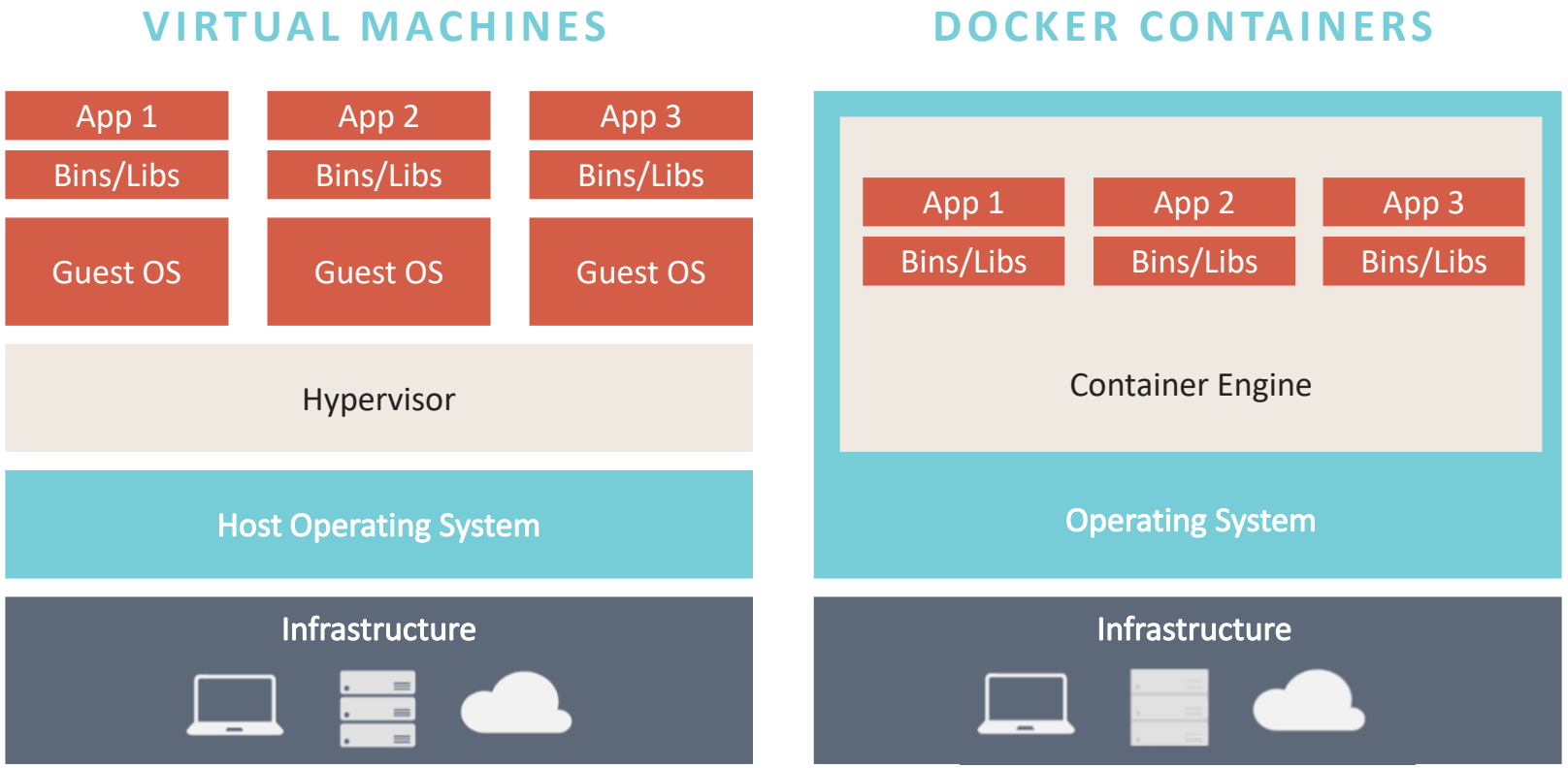
No more:
"It works in my dev machine!...
Why not in production?"

Now it is:
"If it works in Docker, it works in production"

Keywords about WHY Docker?
- ***Dependencies (self-sufficient)***
- ***Deployment***



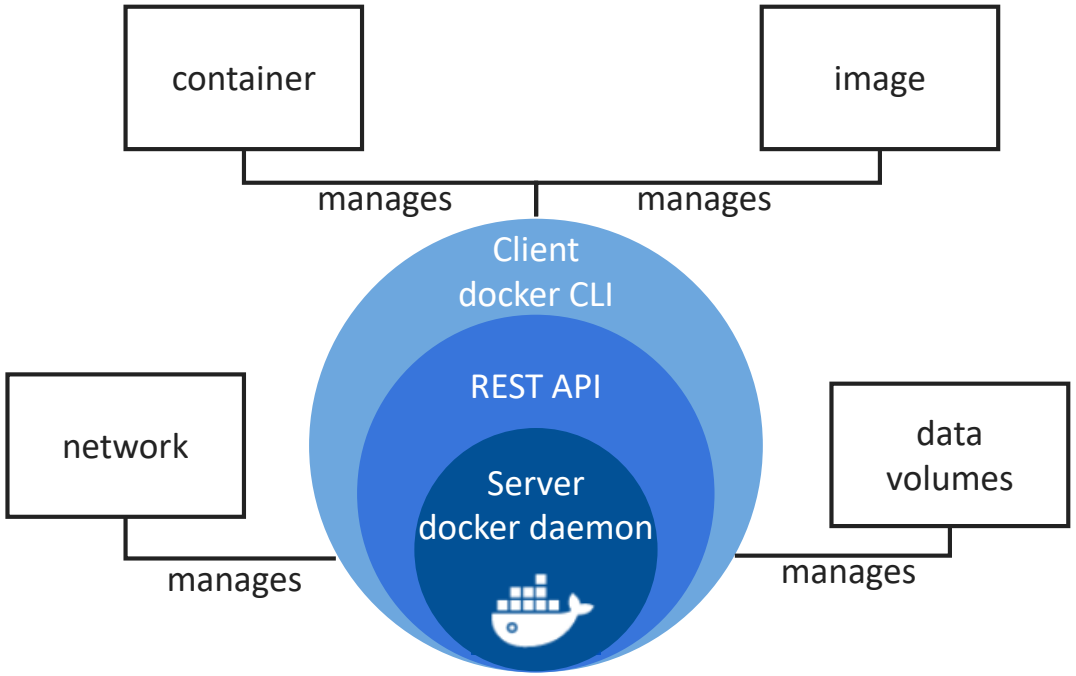
VMs vs Docker Containers



Docker Architecture

DOCKER MAJOR PARTS

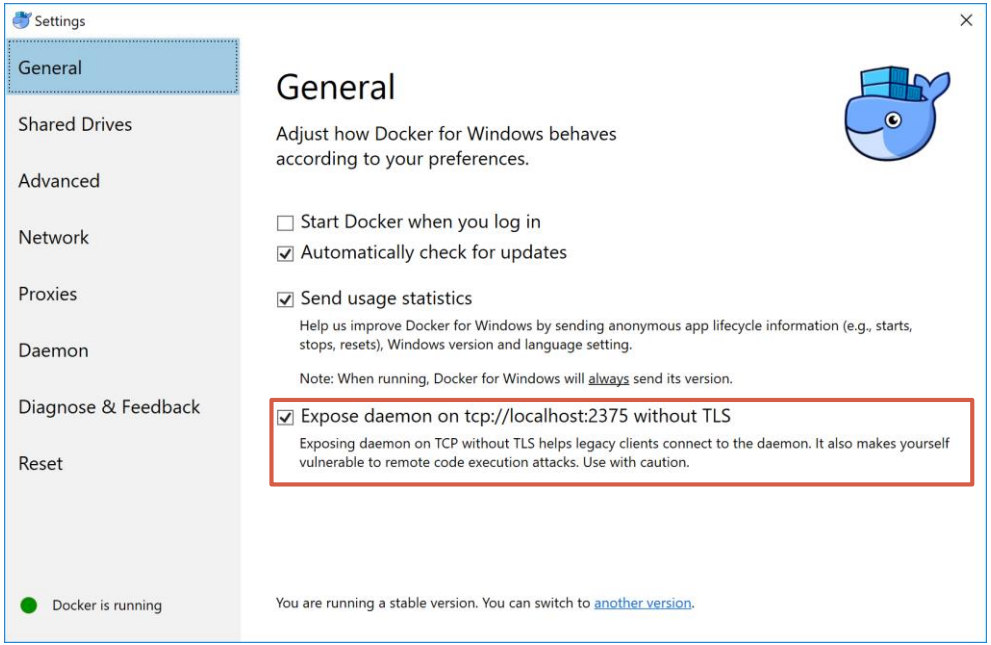
- Daemon Process
- A REST API
- A command line interface (CLI)



Docker API

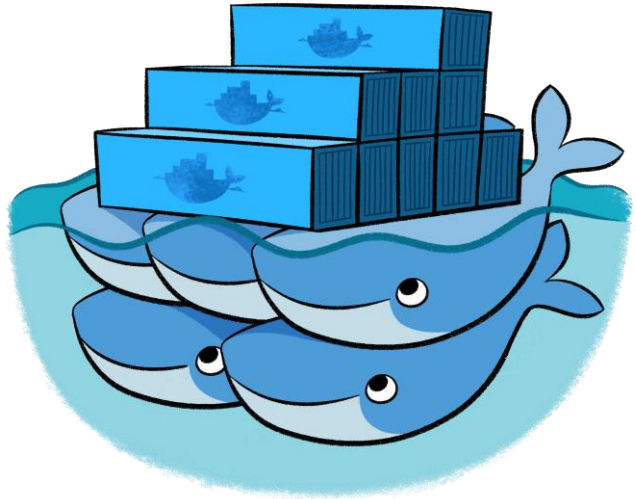
TO CONSUME DOCKER API

- 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#?

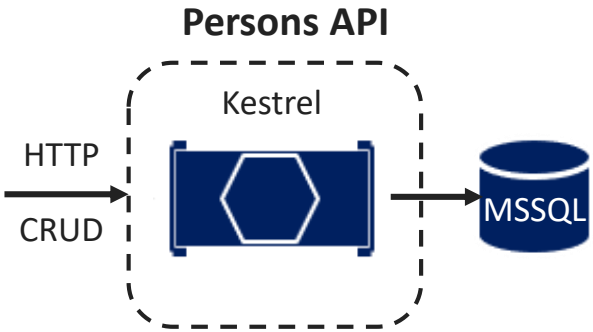
- Install Docker.DotNet
- Get Docker URL
- Use DockerClient Methods to:
  - *List Containers*
  - *Pull Images*
  - *Create Containers*
- Connect to newly created container



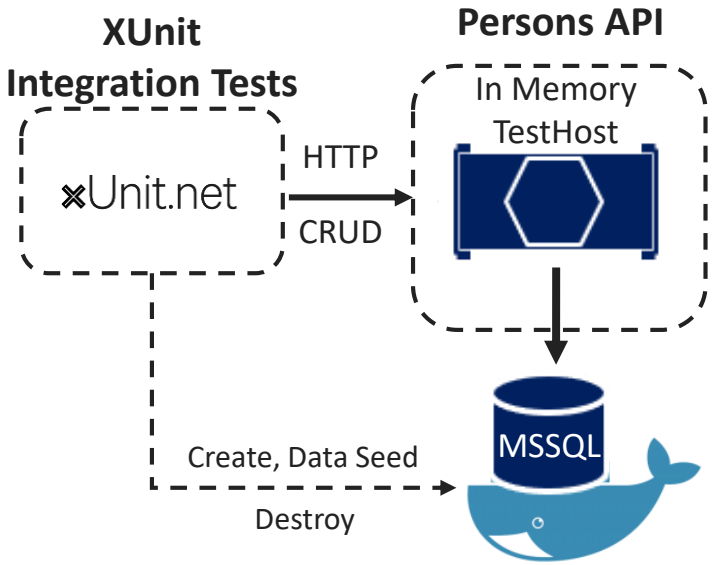
DEMO 1:
RUN MSSQL CONTAINER WITH DOCKER.NET

Microservice Testing with Docker

PERSONS MICROSERVICE TO TEST



TESTING WITH XUNIT AND DOCKER

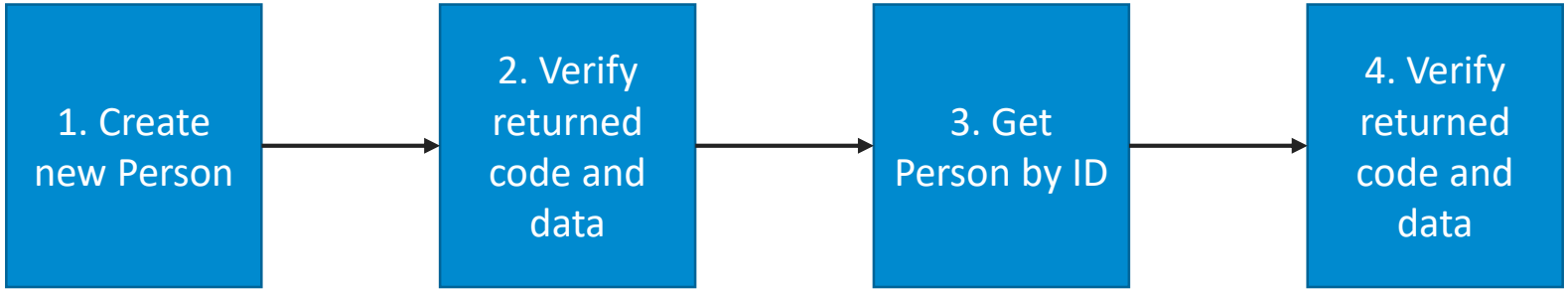


Technology List?

- **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



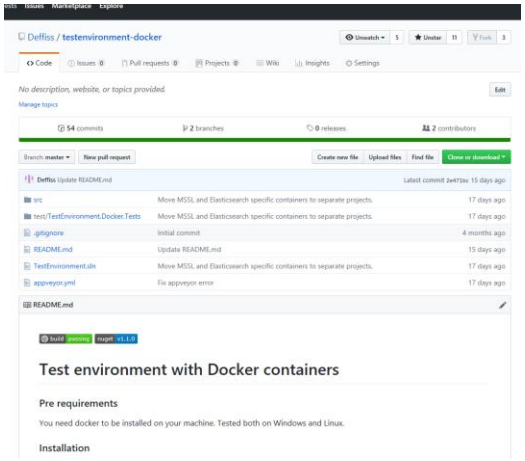
DEMO 2:
MICROSERVICE TESTING

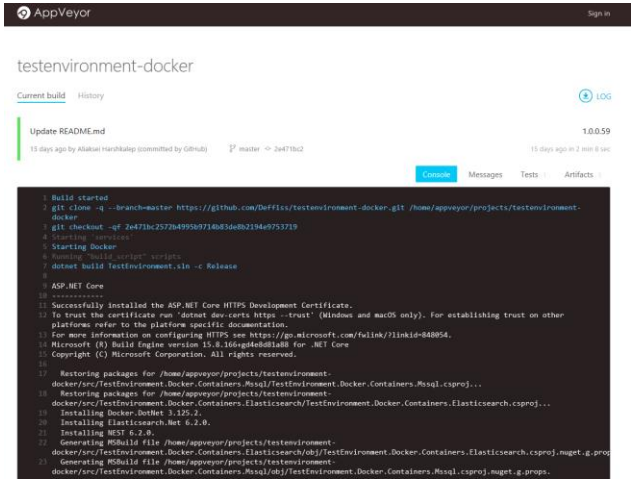
TestEnvironment.Docker: Motivation

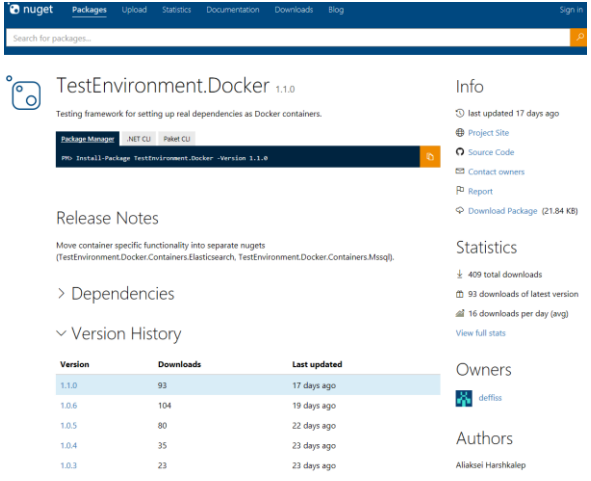
- Reusability
- Enrich with features
- Readiness problems
- Make it CI/CD Friendly (GitLab)
- Share with Community

TestEnvironment.Docker

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





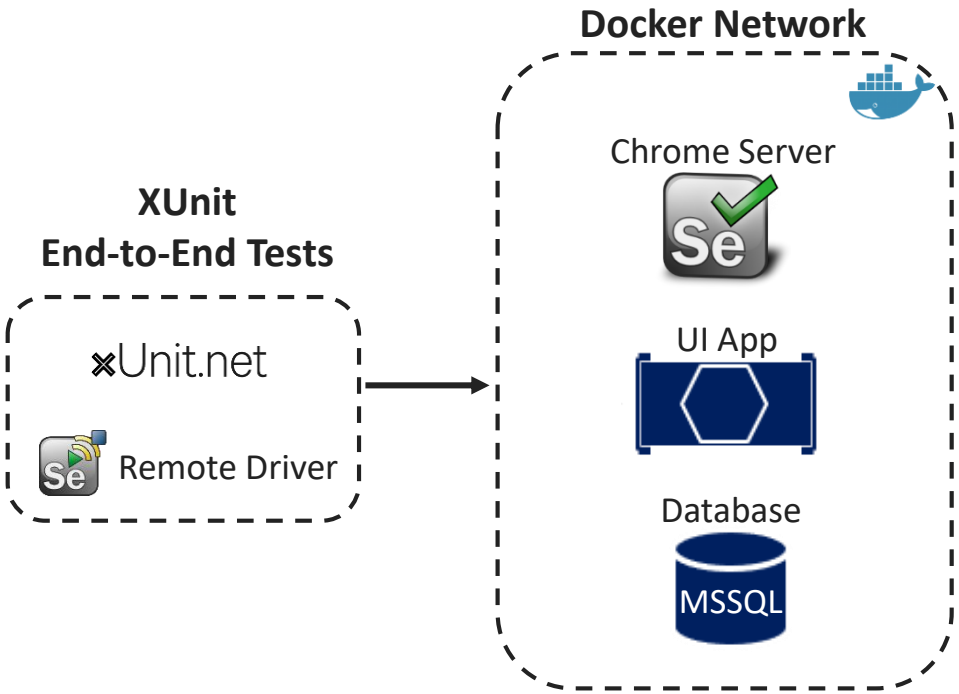


https://github.com/Deffiss/testenvironment-docker

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

DEMO 3: TESTENVIRONMENT.DOCKER

End-to-End-Testing with Docker

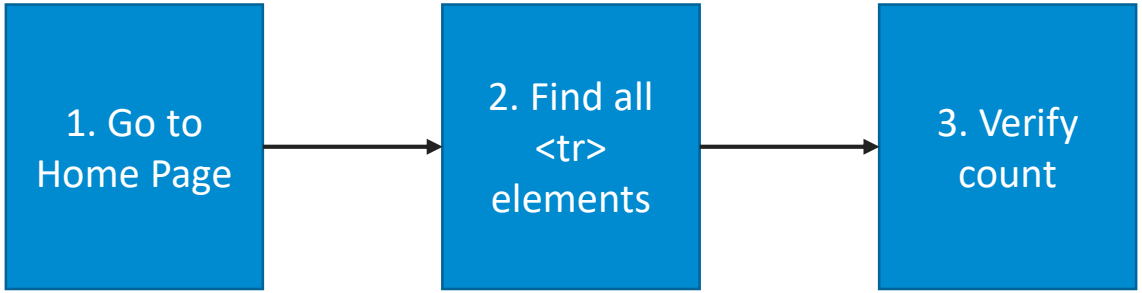


Technology List?

- 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
- TestEnvironmenet.Docker – for setting up testing environment and application

Sample UI Testing Scenario

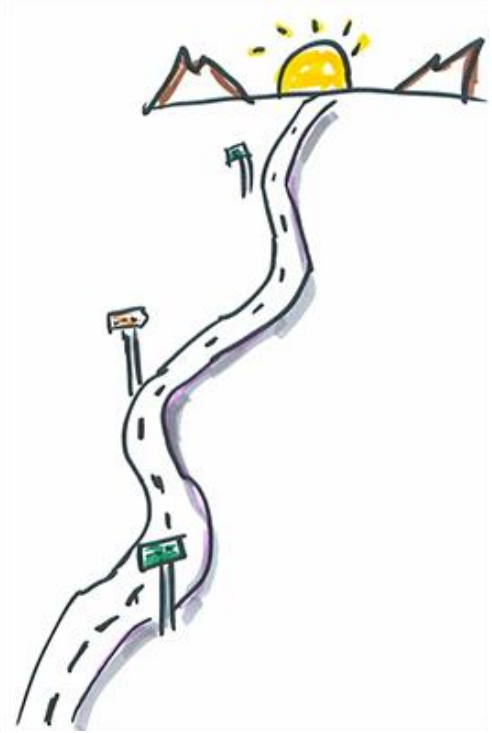
HOMEPAGE_WHENGETHOMEPAGE_SHOULDRETURTABLE



DEMO 4:
END-TO-END TESTS

Roadmap

- Configurability (.yaml)
- Dependencies Between Containers
- Containers from Dockerfile
- More Built-in Containers Support (MYSQL, MongoDB, Reddis, 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?