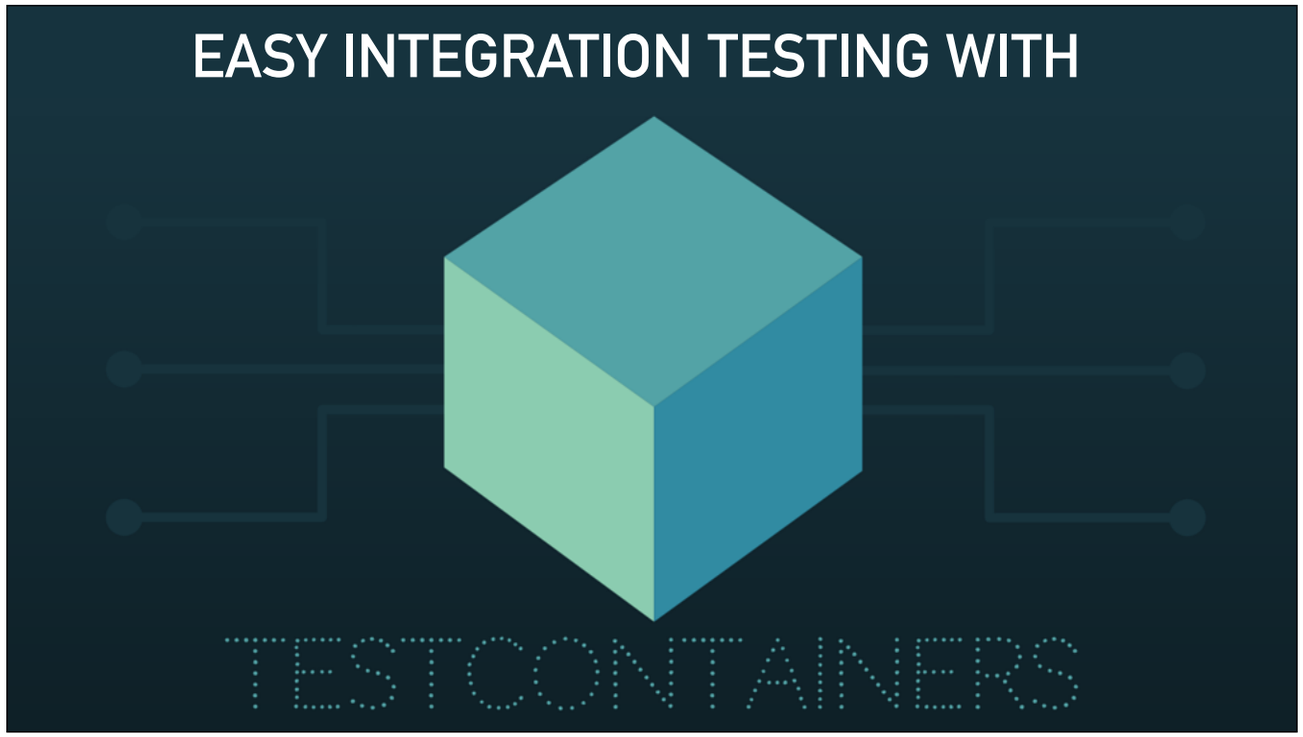


Welcome and introduce yourself:
-Software developer at Unosquare
-Primarily JVM (Java/Kotlin) and some mobile/devops as well

Test containers is a library to help simplify your integration - easier to run, and more reliable
Before we get into that, we'll begin exploring why we should even integration test in the first place

QUICK POLL

HANDS UP IF YOU WRITE TESTS

HANDS UP IF YOU WRITE UNIT TESTS

HANDS UP IF YOU WRITE INTEGRATION TESTS

- System under test
- External/out of process dependencies
- Test suites

HANDS UP IF YOU LIKE WRITING
INTEGRATION TESTS

INTEGRATION TESTING IS HARD!

MANUALLY INSTALLING DEPENDENCIES

BIT ROT – VARIANCE OVER TIME

SHARED STATE

TESTS INTERFERE WITH
EACH OTHER

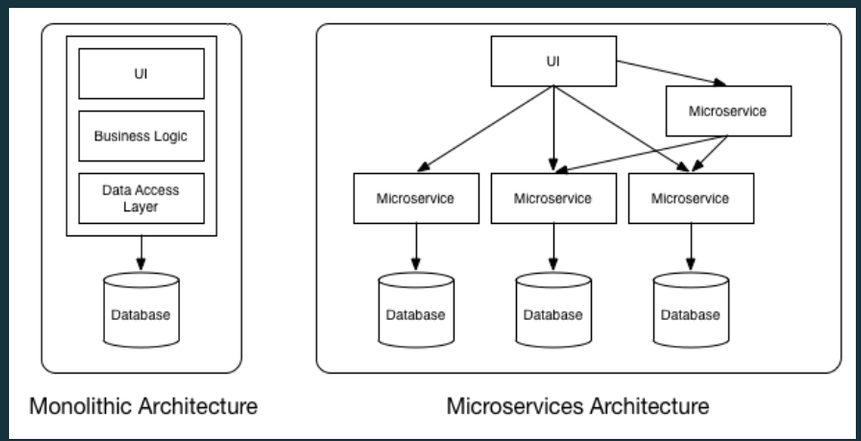
RELIANCE ON MOCKS

MOCKS PROVIDE VARYING ASSURANCE OF COMPATIBILITY

CI PIPELINE CONSISTENCY

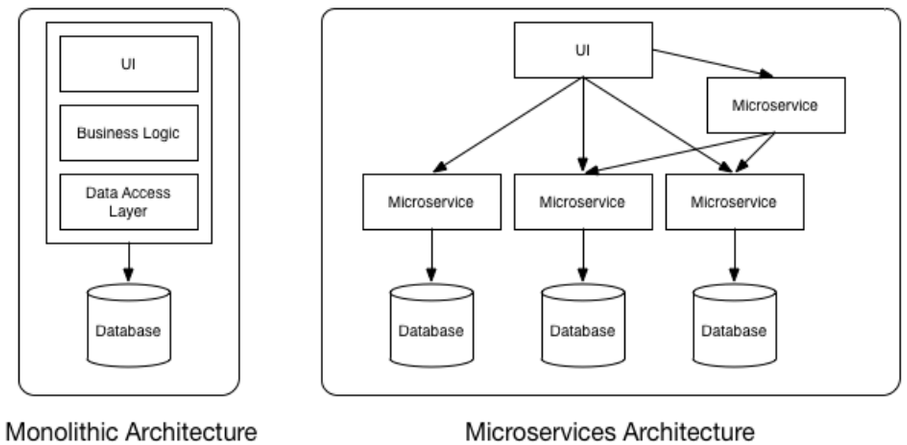
ENVIRONMENTS NOT THE SAME BETWEEN DEV AND CI

INTEGRATION TESTING A MONOLITH VS MICROSERVICE



Source: https://howtodoinjava.com/microservices/microservices-definition-principles-benefits/

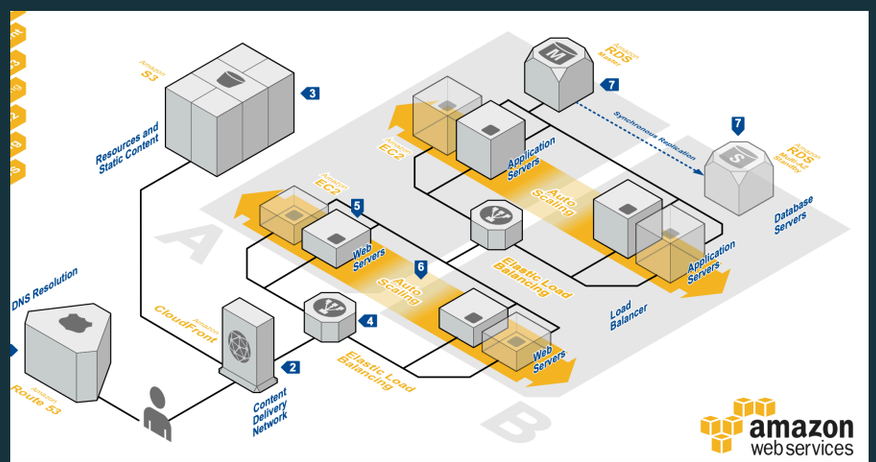
EASY INTEGRATION TESTING WITH TEST CONTAINERS



Source: https://howtodoinjava.com/microservices/microservices-definition-principles-benefits/

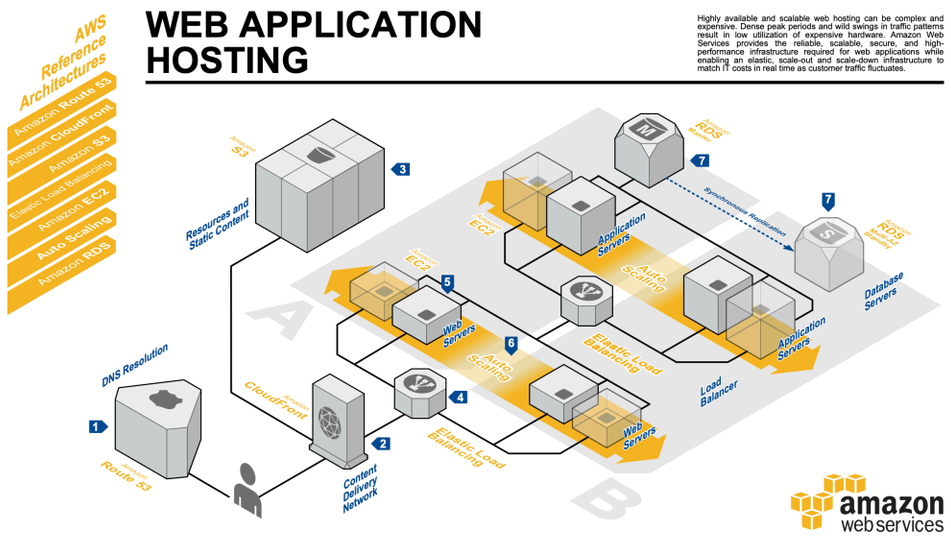
Comparison of integration testing in a monolithic architecture vs micro-service architecture

NOW FACTOR IN CLOUD INFRASTRUCTURE...



Source: https://aws.amazon.com/architecture

Integration testing with cloud infrastructure - getting harder and harder to test



Source: https://aws.amazon.com/architecture

Integration testing with cloud infrastructure - getting harder and harder to test

WHY DO WE EVEN NEED INTEGRATION TESTS?

INTEGRATION TEST ADVANTAGES

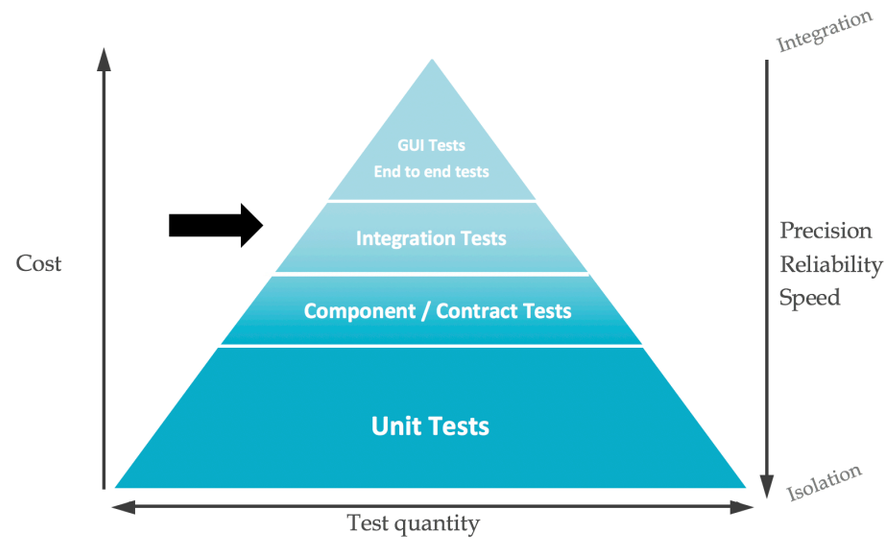
- Real-world, but isolated testing
- Spot the issues before the real environment
- Can be run during the development

INTEGRATION TEST DISADVANTAGES

- You have to start real databases
- Should be cross-platform
- Slower than unit testing

TESTING PYRAMID

EASY INTEGRATION TESTING WITH TEST CONTAINERS



Source: https://blog.octo.com/en/the-test-pyramid-in-practice-5-5/

Slow -> fast
Costly -> cheap
UI -> integration -> unit
Big base of tests

WHY NOT JUST USE EMBEDDED DATABASES?

Not 100% compatible - might fail in production against real database

H2 has a lot of benefits, it’s fast, great integration with spring boot. But its not Postgres - compatibility mode is not enough

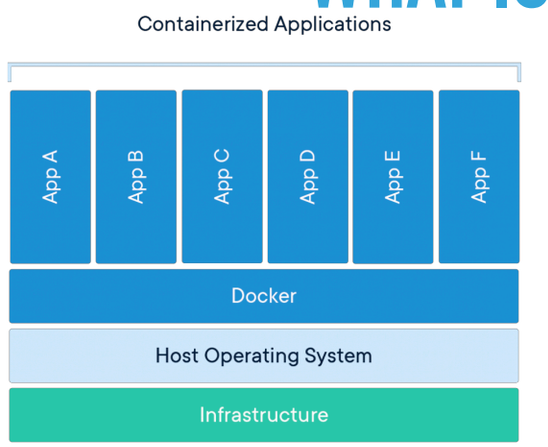
HOW CAN WE MAKE INTEGRATION TESTING EASIER AND MORE RELIABLE?


docker
TO THE RESCUE

DOCKER INTRO

CI Friendly
Cross platform

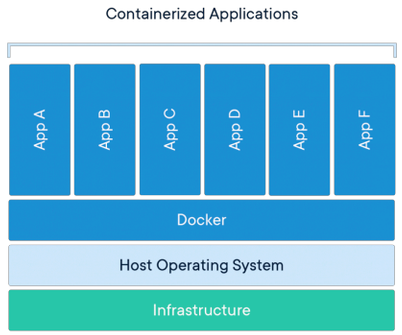
WHAT IS A CONTAINER



- Packages up code and all its dependencies so that an application runs quickly and reliably from one computing environment to another (development, CI, production)
- Lightweight, executable.

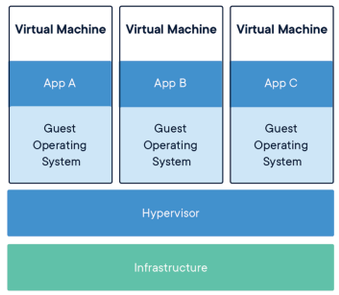
Source: https://www.docker.com/resources/what-container

CONTAINERS VS VIRTUAL MACHINES



Containers are an abstraction at the app layer that packages code and dependencies together. Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space. Containers take up less space than VMs (container images are typically tens of MBs in size), can handle more applications and require fewer VMs and Operating systems.

Source: https://www.docker.com/resources/what-container



Virtual machines (VMs) are an abstraction of physical hardware turning one server into many servers. The hypervisor allows multiple VMs to run on a single machine. Each VM includes a full copy of an operating system, the application, necessary binaries and libraries - taking up tens of GBs. VMs can also be slow to boot.

DOCKER FOR DEV ENVIRONMENTS

Containerization of apps - making them more consistent across environments

WHY CAN'T WE DO THE SAME WITH OUR TESTS (RUN ANYWHERE)

TESTCONTAINERS

WHAT IS TEST CONTAINERS?

- Library that allows us to test against real instances of anything that can run in a Docker container
- Containers as code - allows us to start and stop Docker containers
- Run tests anywhere, with only Docker as a dependency
- Automatic docker environment discovery (Win, Mac, Linux)
- Will start docker-machine if its not started yet
- Containers cleanup on JVM shutdown

WHAT CAN IT BE USED WITH?

- Can be used with any of the following:
  - JVM (Java, Kotlin, Groovy, Scala)
  - Node, Go, Python
  - Some early development forks - .NET, MicroProfile
  - Supports Linux, MacOS and Windows

WHAT CAN IT BE USED FOR?

- Database containers - MySQL, Postgres, Oracle and many more
- Selenium web driver- Chrome and Firefox
- Docker compose - to orchestrate numerous containers
- DockerFile - anything expressible in a docker file
- Generic container - any image from DockerHub or a private registry
- And many more!

SOUNDS GREAT, BUT WHY NOT JUST USE DOCKER COMPOSE?

DOCKER-COMPOSE ADVANTAGES

- Can start and orchestrate multiple containers
-

DOCKER-COMPOSE DISADVANTAGES

- No port randomisation
- Fighting with docker environment - docker for mac, docker toolbox
- No clean up
- Lack of IDE support

EXAMPLES

Pumba - chaos testing - simulate network failures

USE CASE 1 – TESTING MICRO SERVICES

- REST service
- Java, Spring Boot
- Redis and PostgreSQL
- Calls some other micro-services

```
exposedPort - containers perspective, from host perspective it is exposed on a random free port
Integer firstMappedPort = container.getMappedPort(2424);

String ipAddress = container.getContainerIpAddress();
```

USE CASE 1 – TESTING MICRO SERVICES

- 1

USE CASE 2: DOCKER AS SELENIUM DRIVER

- Uses containerised web browsers, compatible with Selenium, for conducting automated UI tests.
- No need to install chrome/firefox/etc
- CI friendly
- Automatic video recording of each test session, or just where tests failed.

USE CASE 3: DOCKER COMPOSE

- 3 - localstack

USE CASE 4: DOCKER FILE

- Creating images on-the-fly with Dockerfile DSL:

```
new GenericContainer(
        new ImageFromDockerfile()
                .withDockerfileFromBuilder(builder ->
                        builder
                                .from("alpine:3.2")
                                .run("apk add --update nginx")
                                .cmd("nginx", "-g", "daemon off;")
                                .build()))
                .withExposedPorts(80);
```

Source: https://www.testcontainers.org/features/creating_images/

USE CASE 5: GENERIC CONTAINER

- DockerHub and private registry

TEST CONTAINERS DISADVANTAGES AND HOW TO OVERCOME THEM

- Slowness of container setup (10 seconds) - limit starting fresh container for every test, can structure your tests into suites:
- ClassRule/Static initialiser/Singleton
- Can disable start up checks (which very that your machine works fine with Docker)
- Relies on underlying docker internals, so will improve as more features are adding to Docker (Checkpoint restore for example)

Before running any containers Testcontainers will perform a set of startup checks to ensure that your environment is configured correctly. Usually they look like this:

```
ℹ Checking the system...
✔ Docker version should be at least 1.6.0
✔ Docker environment should have more than 2GB free disk space
✔ File should be mountable
✔ A port exposed by a docker container should be accessible
```

It takes a couple of seconds, but if you want to speed up your tests, you can disable the checks once you have everything configured. Add checks.disable=true to your $HOME/.testcontainers.properties to completely disable them.

ADVANCED OPTIONS

- If there is an advanced feature you want to use that Testcontainers API doesn't expose, you can docker-java API directly by passing in commands using withCreateContainerCmdModifier:

```
@Rule
public GenericContainer theCache = new GenericContainer<>("redis:3.0.2")
        .withCreateContainerCmdModifier(cmd -> cmd.withHostName("the-cache"));
```

Before running any containers Testcontainers will perform a set of startup checks to ensure that your environment is configured correctly. Usually they look like this:

```
ℹ Checking the system...
✔ Docker version should be at least 1.6.0
✔ Docker environment should have more than 2GB free disk space
✔ File should be mountable
✔ A port exposed by a docker container should be accessible
```

It takes a couple of seconds, but if you want to speed up your tests, you can disable the checks once you have everything configured. Add checks.disable=true to your $HOME/.testcontainers.properties to completely disable them.

ADVANCED OPTIONS

- Docker “wormhole” - test containers can be used from inside a container.
- This is useful for CI scenarios

Source: https://www.testcontainers.org/supported_docker_environment/continuous_integration/dind_patterns/

CREDITS

QUESTIONS?