EASY INTEGRATION TESTING WITH

TESTCONTAINERS

QUICK POLL

HANDS UP IF YOU WRITE TESTS

HANDS UP IF YOU WRITE UNIT TESTS

HANDS UP IF YOU WRITE INTEGRATION TESTS

“Tests which interact with external systems/dependencies”

HANDS UP IF YOU LIKE WRITING INTEGRATION TESTS

INTEGRATION TESTING IS HARD!

MANUALLY INSTALLING DEPENDENCIES

Need to install dependencies just to get tests to run

BIT ROT – VARIANCE OVER TIME

Dependencies and tooling are constantly changing

SHARED STATE

Tests interfere with each other

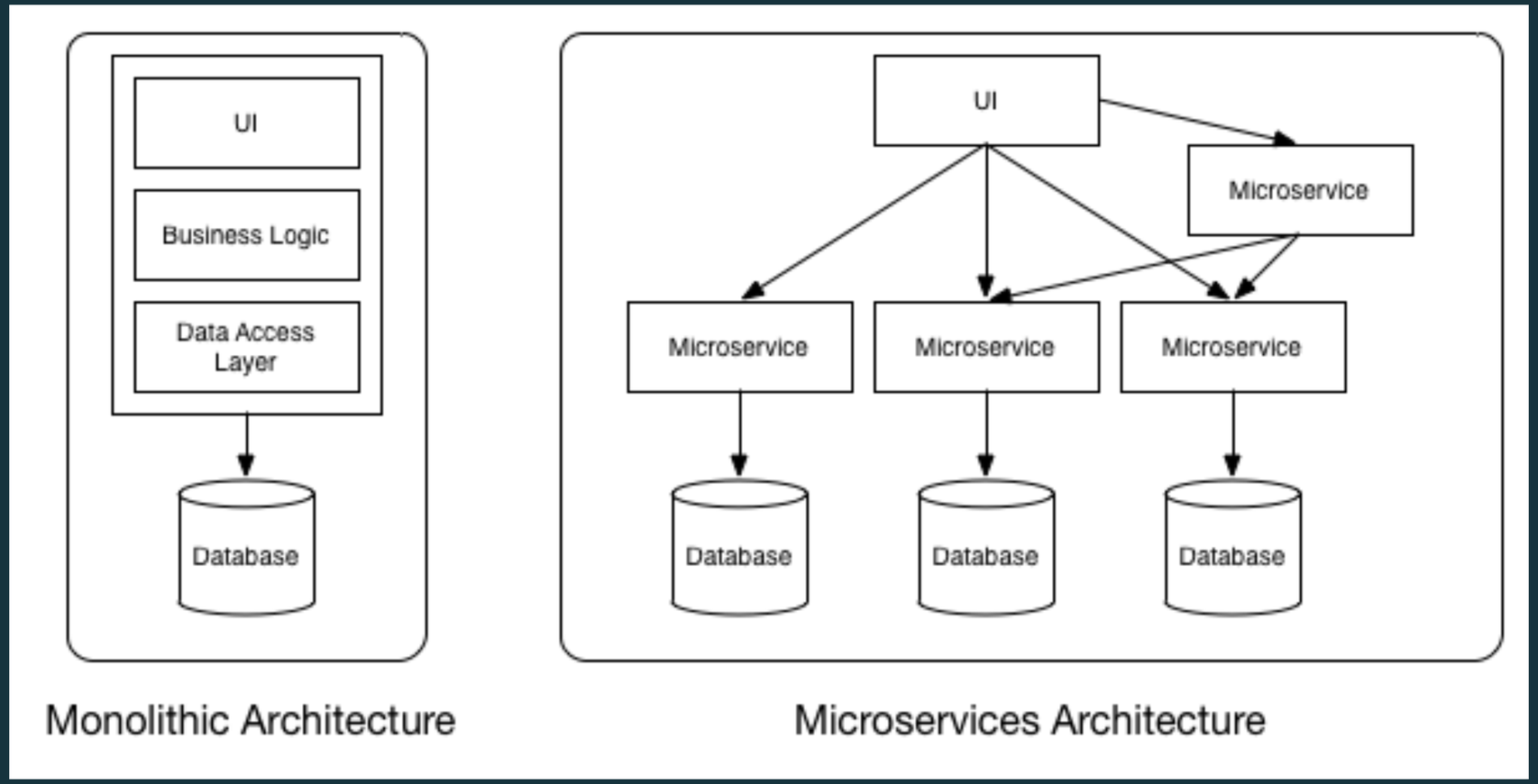
RELIANCE ON MOCKS

Mocks provide varying assurance of compatibility

CI PIPELINE CONSISTENCY

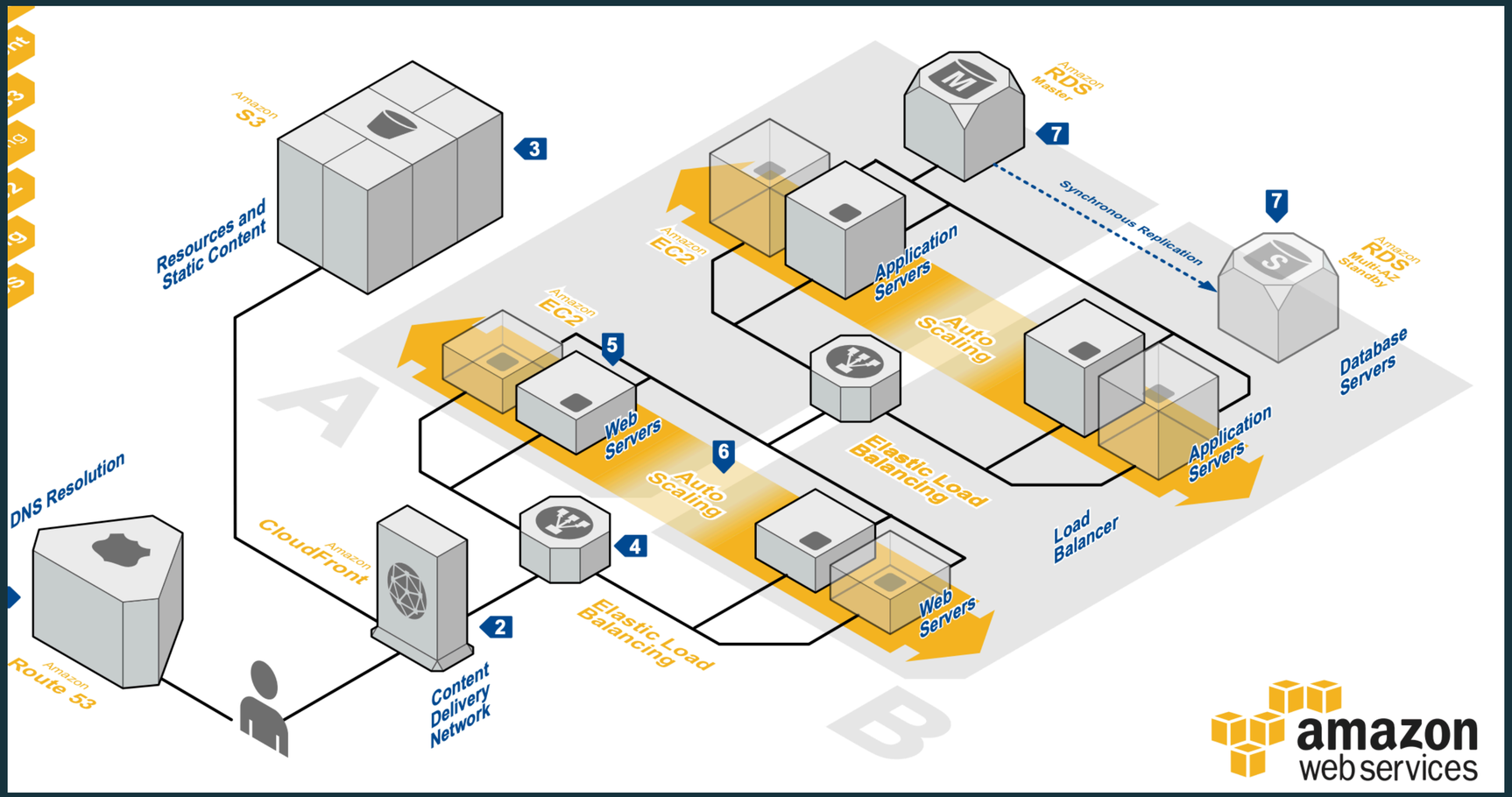
Environments not the same between local and CI

INTEGRATION TESTING A MONOLITH VS MICROSERVICE



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

NOW FACTOR IN CLOUD INFRASTRUCTURE...



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

IF IT'S SO DIFFICULT, WHY DO WE EVEN NEED INTEGRATION TESTING?

2 UNIT TESTS,
0 INTEGRATION TESTS

INTEGRATION TEST ADVANTAGES

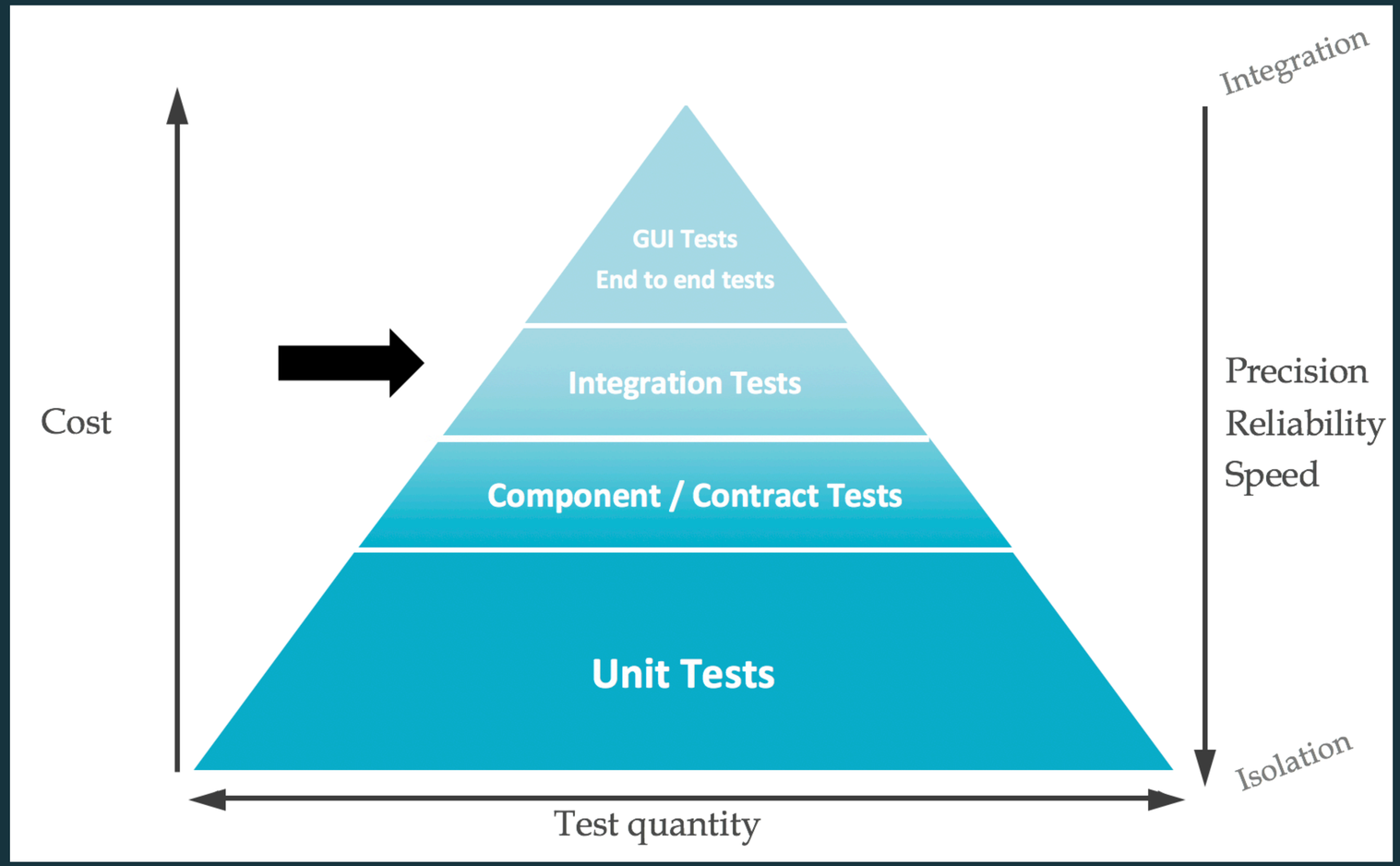
- Real-world, but isolated testing
- Spot the issues before the real environment
- Can be ran during the development

INTEGRATION TEST DISADVANTAGES

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

HOW SHOULD TESTS BE STRUCTURED?

THE TESTING PYRAMID

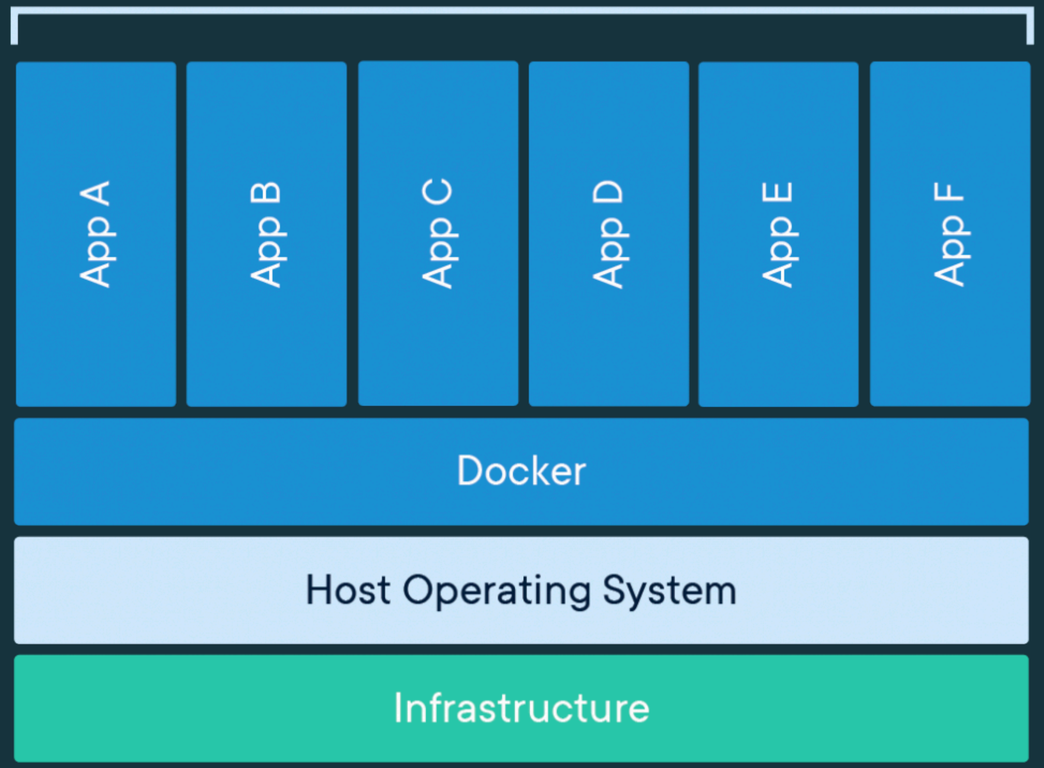


Source: https://blog.octo.com/en/the-test-pyramid-in-practice-5-5/

HOW CAN WE MAKE INTEGRATION TESTING EASIER AND MORE RELIABLE?

TO THE RESCUE

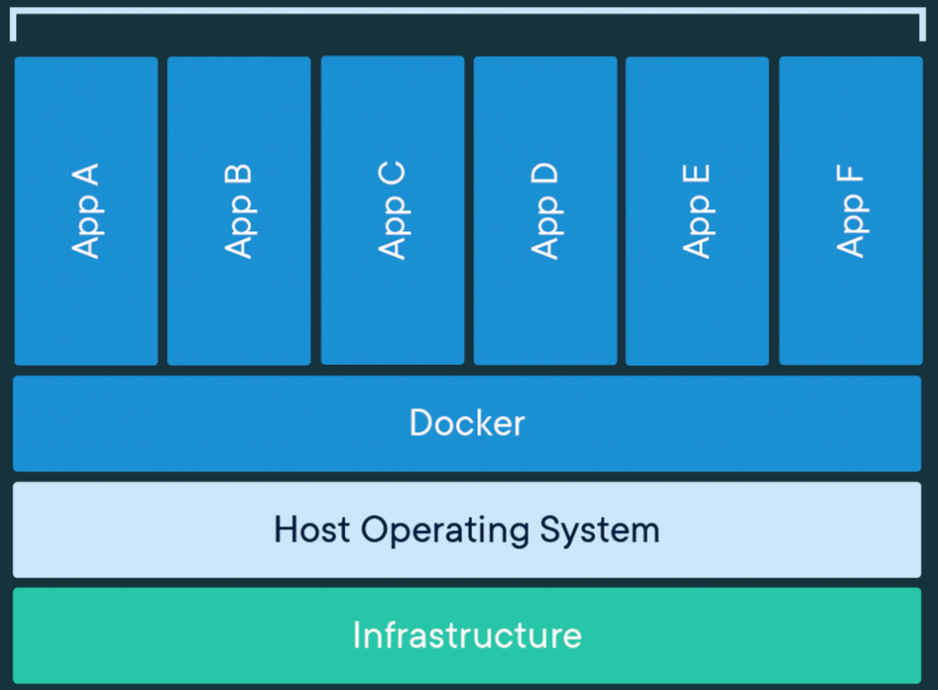
WHAT IS A DOCKER CONTAINER?

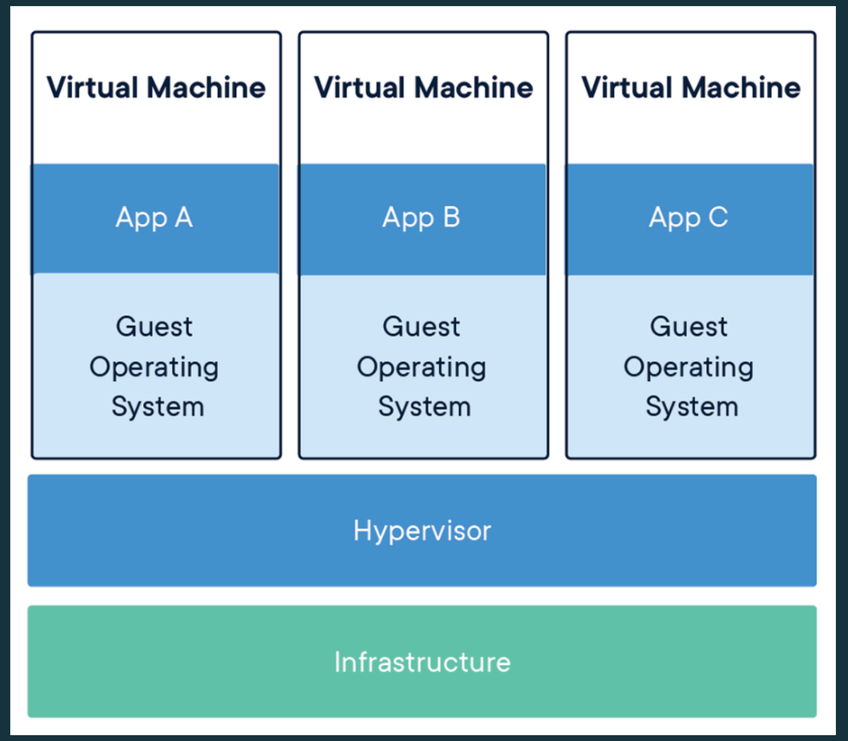


- Standardized way to package up software and dependencies
- Isolate apps from each other
- Share the same OS kernel
- Portable - works on all platforms
- Docker Image - the basis of a Docker container

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

CONTAINERS VS VIRTUAL MACHINES





Containers are an app level construct

VMS are an infrastructure level construct to turn one machine into many servers

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

SO WHY USE DOCKER CONTAINERS?

- Speed - No OS to boot
- Portability
- Efficiency - No OS overhead

DOCKER HAS MASSIVELY SIMPLIFIED HOW WE BUILD AND DEPLOY APPLICATIONS

WHY CAN'T WE GET THE SAME BENEFITS FOR OUR TESTS?

INTRODUCING TESTCONTAINERS

WHAT IS TEST CONTAINERS?

- “TestContainers is a Java library that supports JUnit tests, providing lightweight, throwaway instances of common databases, Selenium web browsers, or anything that can run in a Docker container”

WHAT FEATURES DOES IT OFFER?

- Containers as code - allows us to start and stop Docker containers on demand
- Automatic discover of docker environment (Win, Mac, Linux)
- Will start docker-machine if its not started yet
- Port randomisation (no port conflicts)
- Containers cleanup on JVM shutdown (no more “zombie” containers)
- CI friendly - Can be ran inside a container, will detect default gateway’s IP

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!

EXAMPLES

EXAMPLE 1 – GENERIC CONTAINER

REDIS CONTAINER WITH GENERIC CONTAINER

```
@ClassRule
public static GenericContainer redis =
    new GenericContainer( dockerImageName: "redis:3.0.2")
        .withExposedPorts(6379);
```

WHAT WILL TEST CONTAINERS DO HERE?

- Automatic discovery of local docker environment
- Pull images (DockerHub or private registry)
- Start/stop container
- Wait for it to be ready
- Port mapping
- Clean up

```
@ClassRule
public static GenericContainer redis =
    new GenericContainer( dockerImageName: "redis:3.0.2")
        .withExposedPorts(6379);
```

WHAT HAVE WE AVOIDED?

- No need to install the dependency
- No need to keep it running, or make sure its running
- No concern over version or configuration differences
- No difference between what runs on CI and locally
- No port clashes, no shared state (unless we want it)
- No clean up scripts required - clean up done automatically

WHAT HAVE WE GAINED?

- Repeatability - locked version of redis
- Debuggable locally - runnable in IDE
- Parallelizable
- Runs anywhere that Docker runs

EXAMPLE 2 – DATABASE CONTAINERS

POSTGRESQL CONTAINER

```
@ClassRule
public static PostgreSQLContainer postgreSQLContainer = new PostgreSQLContainer( dockerImageName: "postgres:11")
    .withDatabaseName("product_db")
    .withUsername("postgres")
    .withPassword("postgres");
```

```
"spring.datasource.url=" + postgreSQLContainer.getJdbcUrl(),
"spring.datasource.username=" + postgreSQLContainer.getUsername(),
"spring.datasource.password=" + postgreSQLContainer.getPassword()
```

- Set our test properties to point to the container
- Can use flyway migration scripts to set up Postgres to be exactly like our real environments

WHAT'S WRONG WITH JUST USING EMBEDDED DATABASES?

PROBLEMS WITH EMBEDDED DATABASES

- Embedded databases have a lot of benefits (fast, easy setup)
- But they’re still not the same as the real database
- Sometimes database features and efficiencies in production are overlooked if they aren’t compatible with the test embedded database

EXAMPLE 3 – UI AUTOMATION

SELENIUM WEBDRIVER

- Uses containerised web browsers, compatible with Selenium, for conducting automated UI tests
- No need to install specific versions of Chrome/Firefox/etc
- Automatic video recording of each test session, or just when tests fail
- Debugging capabilities

SELENIUM DEMO

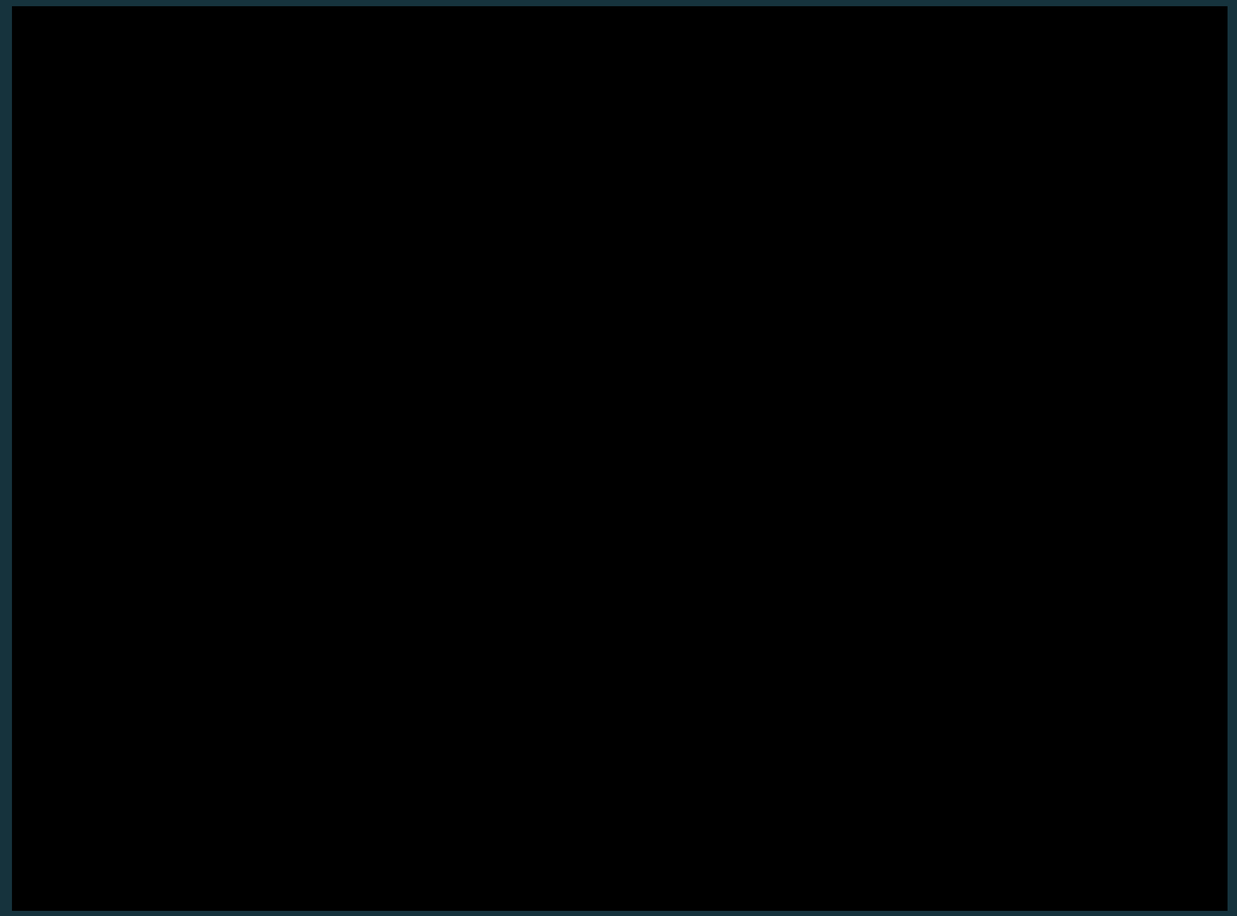
```
@Rule
public BrowserWebDriverContainer chrome = new BrowserWebDriverContainer()
    .withCapabilities(new ChromeOptions())
    .withRecordingMode(RECORD_ALL, new File( pathname: "./out/"));

@Test
public void whenSearchingRickAstleyInChromeExpectMemeFound() {
  RemoteWebDriver driver = chrome.getWebDriver();
  driver.get("https://wikipedia.org");
  WebElement searchInput = driver.findElementByName( using: "search");
  searchInput.sendKeys( ...keysToSend: "Rick Astley");
  searchInput.submit();

  WebElement otherPage = driver.findElementByLinkText( using: "Rickrolling");
  otherPage.click();

  boolean expectedTextFound = driver.findElementsByCssSelector( using: "p")
      .stream()
      .anyMatch(element -> element.getText().contains("meme"));

  assertThat(expectedTextFound).isTrue();
}
```



EXAMPLE 4 – DOCKER COMPOSE

DOCKER-COMPOSE

- Can start and orchestrate multiple containers running on the same host
- No port randomisation
- Hard to clean up (zombie containers)

```yaml
version: '3'
services:
  web:
    build: .
    ports:
      - "5000:5000"
    volumes:
      - .:/code
    environment:
      FLASK_ENV: development
  redis:
    image: "redis:alpine"
```

DOCKER-COMPOSE CONTAINER

- Adding more and more containers doesn’t scale:

```
@ClassRule
public GenericContainer db = new GenericContainer( dockerImageName: "mongo:3.0.15");

@ClassRule
public GenericContainer cache = new GenericContainer( dockerImageName: "redis:3.2.8");

@ClassRule
public GenericContainer search = new GenericContainer( dockerImageName: "elasticsearch:5.4.0");
```

DOCKER-COMPOSE CONTAINER

- We can make use of a docker compose file to declare all our dependencies at once:

```
db:
  image: mongo:3.0.15
cache:
  image: redis:3.2.8
search:
  image: elasticsearch:5.4.0
```

```
@ClassRule
public static DockerComposeContainer environment = new DockerComposeContainer(
    new File( pathname: "src/integTest/resources/docker-compose-backend.yml"))
    .withExposedService( serviceName: "db",  servicePort: 21017)
    .withExposedService( serviceName: "cache",  servicePort: 6379)
    .withExposedService( serviceName: "search",  servicePort: 9200);
```

- Allows you to easily orchestrate complex test scenarios

EXAMPLE 5 – DOCKERFILE

USE CASE 4: DOCKER FILE

- Useful if you want to create a custom image that isn't in a registry
- Can read in a docker file or use the 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/

EXAMPLE 6 – LOCALSTACK

LOCALSTACK

- Allows us to run AWS components locally
- LocalStackContainer spins up S3, then we can point to it in using AWS SDK

```
@Rule
public LocalStackContainer localstack = new LocalStackContainer()
    .withServices(S3);

@Test
public void testingS3Setup() {
  AmazonS3 s3 = AmazonS3ClientBuilder
      .standard()
      .withEndpointConfiguration(localstack.getEndpointConfiguration(S3))
      .withCredentials(localstack.getDefaultCredentialsProvider())
      .build();
  s3.createBucket( bucketName: "bucket-name");

  List<Bucket> buckets = s3.listBuckets();
  assertThat(buckets).isNotEmpty();
}
```

EXAMPLE 7 – ADVANCED OPTIONS

ADVANCED OPTIONS

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

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

SOUNDS TOO GOOD TO BE TRUE. WHAT'S THE CATCH?

TEST CONTAINERS DISADVANTAGES AND HOW TO OVERCOME THEM

- “Slow” container startup (~10 seconds) - try to limit starting fresh container for every test.
- There are different patterns that be applied to limit this:
  - @ClassRule to share container within a test class
  - Structuring your tests as a suite, and using static initializer/singleton pattern to reuse the same container instance
- Can disable start up checks (which verify that your machine works fine with Docker)

CREDITS

- www.testcontainers.org
- github.com/testcontainers
- testcontainers.slack.com

QUESTIONS?

Eamon Scullion

https://github.com/escullion/testcontainers-examples