Integration Testing with Testcontainers

Martin Nachev

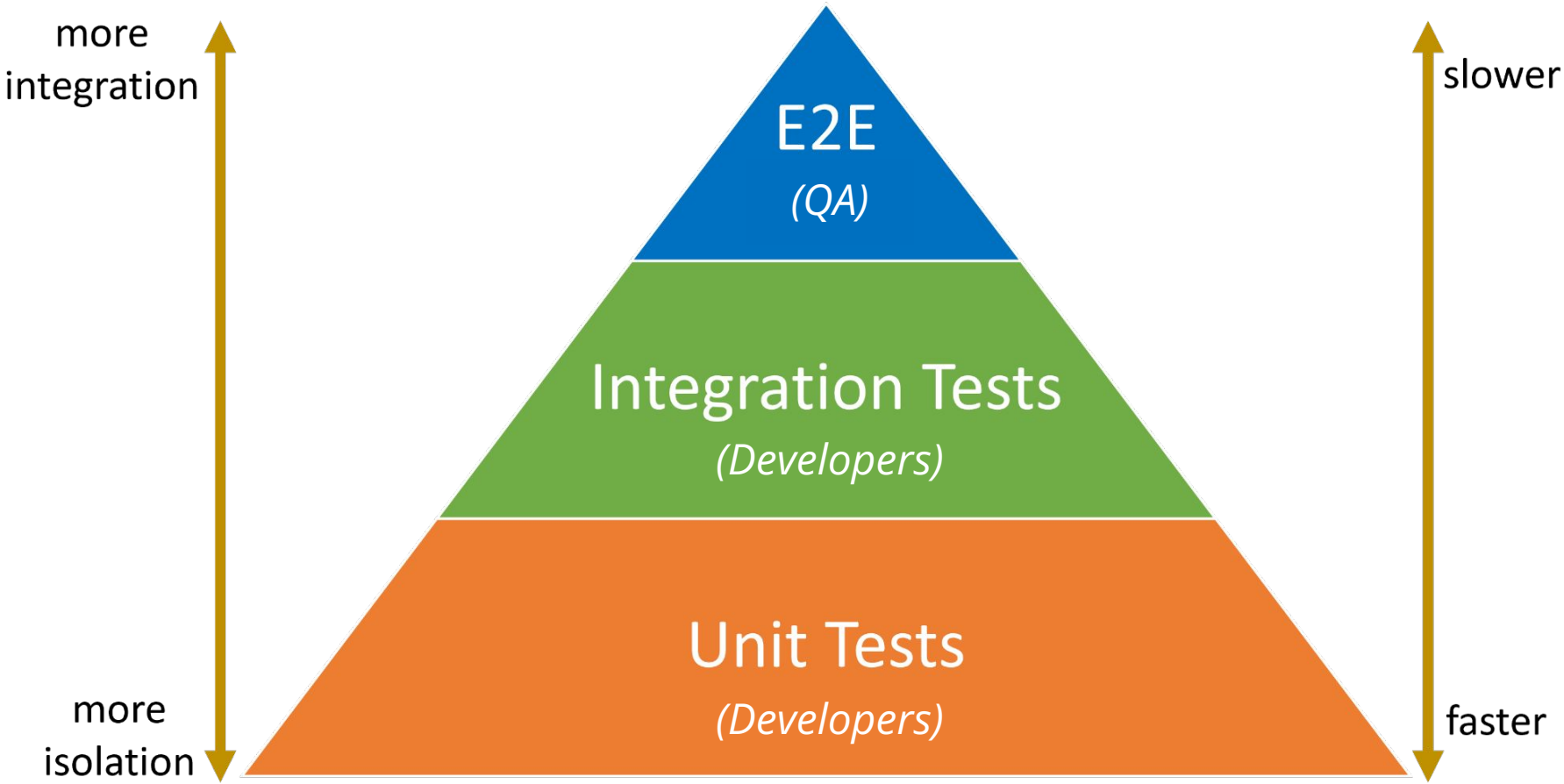
Engineering Manager

personal programmer

propeller

Who here writes tests?

Who here likes maintaining tests?


more integration
more isolation
E2E
(QA)
Integration Tests
(Developers)
Unit Tests
(Developers)
slower
faster

Why Integration Tests

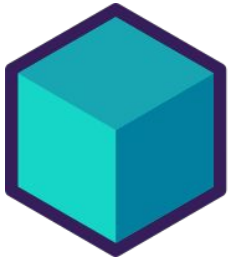
- We make CRUD apps
  - DBs queries contain most of the important logic and performance risk
- Mocking everything tests nothing
  - Mocks make hard-to–maintain tests especially when refactoring

- But integrations tests are slower than unit tests, so don’t overdo it
  - Do not use them for things like validations or pure functions

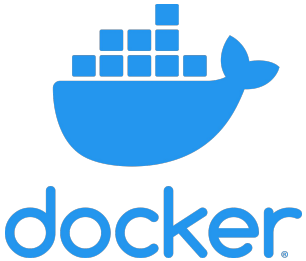
The problem of running today's apps

- Cloud-native infrastructure & microservices
  - First run MySQL, Redis, Kafka and Service B every single time...
  - ... or use mocks/in-memory services that won't act like the real thing
  - ... or connect to Cloud environment's instances
  - ... and make sure there's no existing corrupted data
  - ... and only then you run/test your service locally or in CI

Testcontainers



Testcontainers



- Bootstrap dependencies with Docker containers
- Test/Run using the exact same services you use in production
- Port randomization for parallel test execution
- Automatic cleanup & teardown with Ryuk sidecar container

- Languages: **Node.js**, Java, .NET, Go, Python, Rust, Haskell, Ruby
- Modules: **GenericContainer**, MongoDB, MySQL, Elasticsearch, Kafka etc.

Testing with Jest

```
describe("MongoDB App Test", () => {
 let container: StartedTestContainer;
 // ...
 beforeAll(async () => {
   container = await new GenericContainer("mongo:6").withExposedPorts(27017).start();
   process.env.DB_URI = `mongodb://localhost:${container.getMappedPort(27017)}`;
   // ...
 });

 afterAll(async () => {
   await container.stop();
   // ...
 });

 it("should do something", async () => {
   // ...
 });
```

Running a NestJS app locally

```
// src/main.ts
async function bootstrap() {
 let container;
 if (!process.env.DB_URI) {
   container = await new GenericContainer("mongo:6").withExposedPorts(27017).start();
   process.env.DB_URI = `mongodb://localhost:${container.getMappedPort(27017)}/jedi`;
 }

 const app = await NestFactory.create(AppModule);
 await app.listen(3000);
}

bootstrap();
```

Using Docker Compose

```
// docker-compose.yaml
services:
 mongo:
   image: "mongo:6"
   ports:
     - "27017:27017"
```

```
// src/main.ts
async function bootstrap() {
 if (!process.env.DB_URI) {
   await new DockerComposeEnvironment(
     ".",
     "docker-compose.yaml")
     .up();
   process.env.DB_URI =
     `mongodb://localhost:27017/jedi`;
 }

 const app = await NestFactory.create(AppModule);
 await app.listen(3000);
}

bootstrap();
```

Demo

Questions?

https://github.com/nachevm/integration-testing-with-testcontainers

personal
programmer

propeller