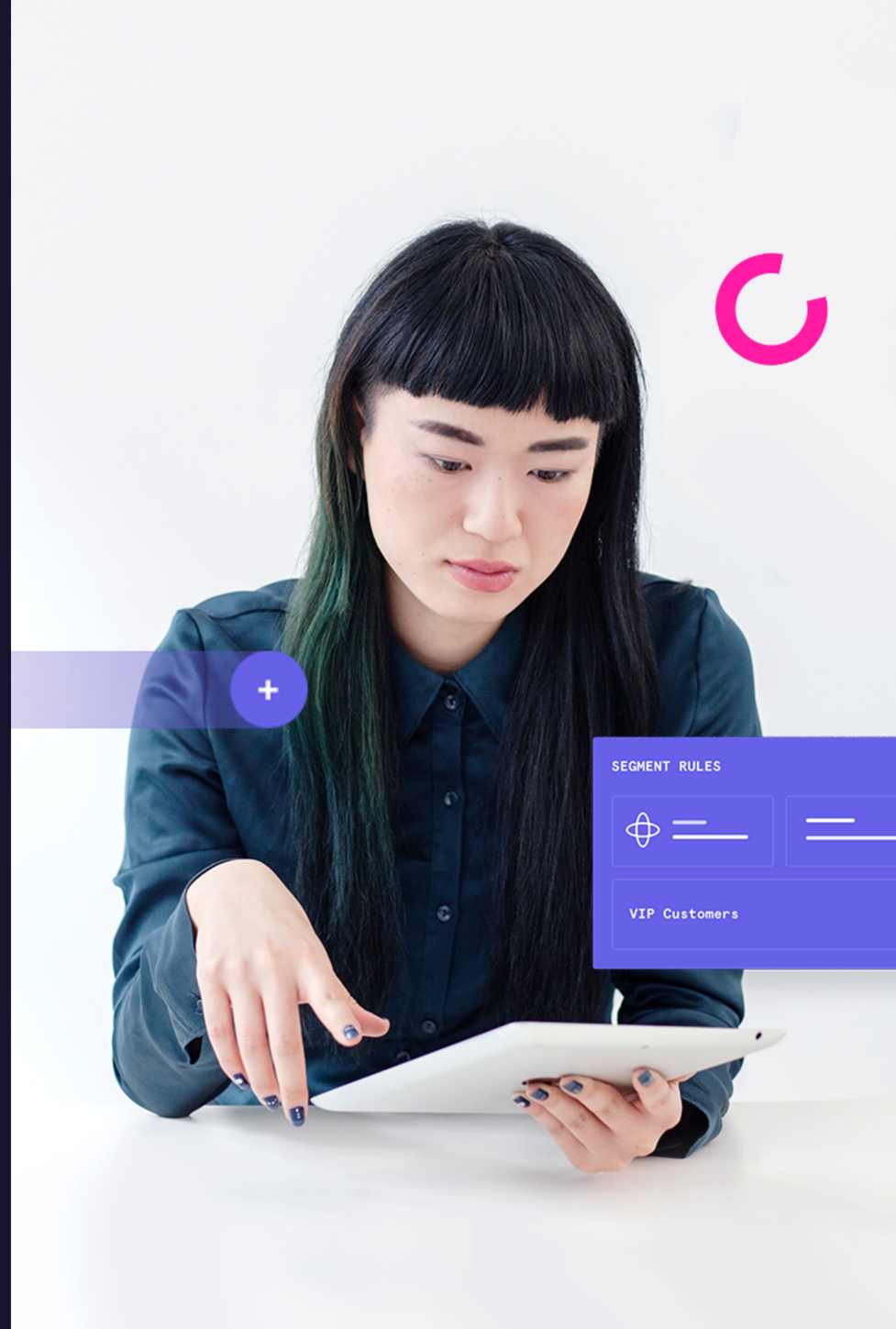
Realistic integration testing with Testcontainers

CHRIS ORTNER

nosto



Agenda

1. What is integration testing?
2. How do Testcontainers work?
3. Testcontainers in action
4. Testcontainers in the wild
5. Burgers

nosto

About me

CHRIS ORTNER

SENIOR SOFTWARE ENGINEER

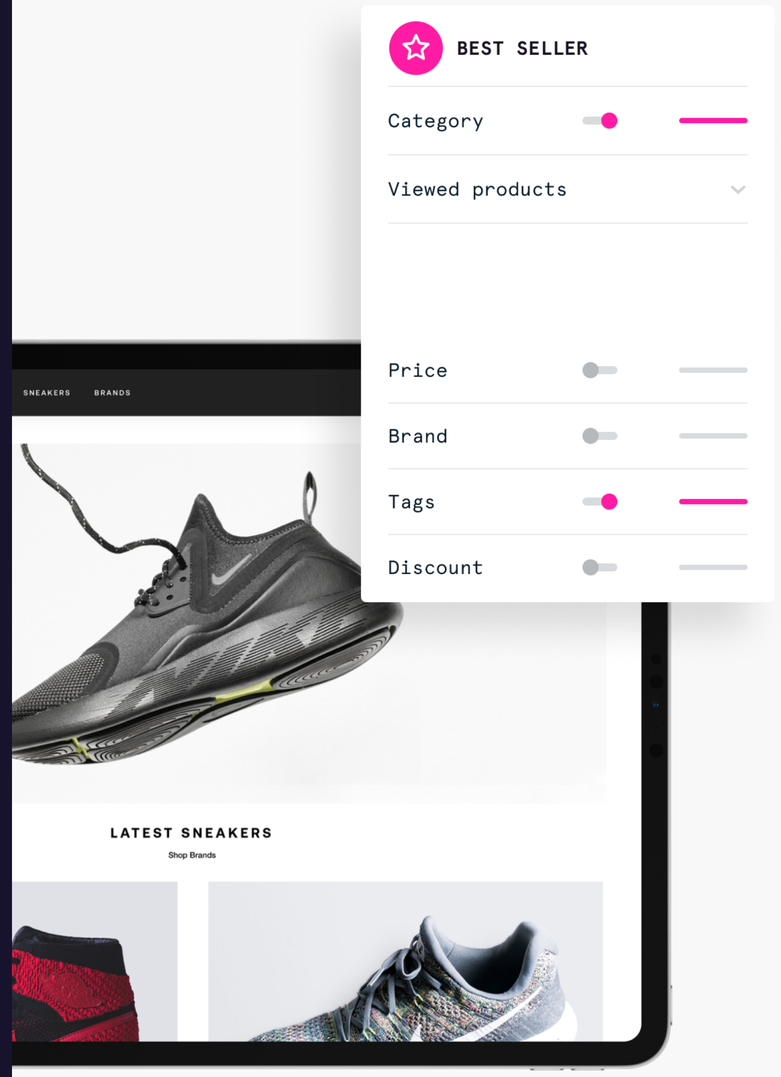
github.com/howard

Scala, Kotlin, TypeScript, PHP...

Cycling, swimming, tinkering, analog/digital gaming

nosto

What is integration testing?



nosto

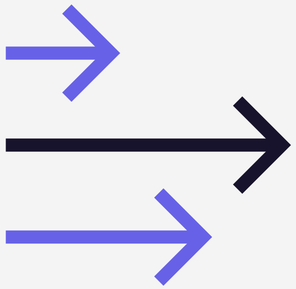
“Testing can only prove the presence of bugs, not their absence.”

– EDSGER W. DIJKSTRA

nosto

Why test?

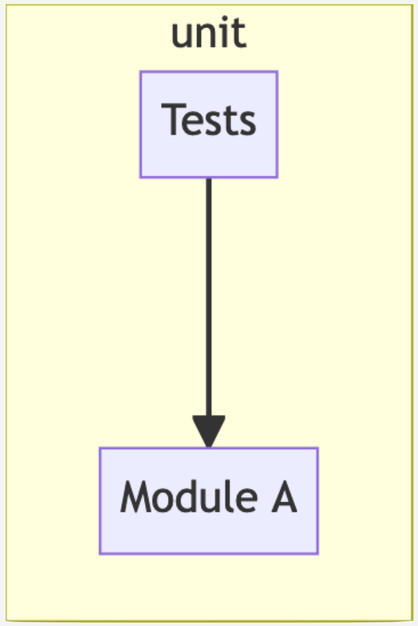
SAFETY

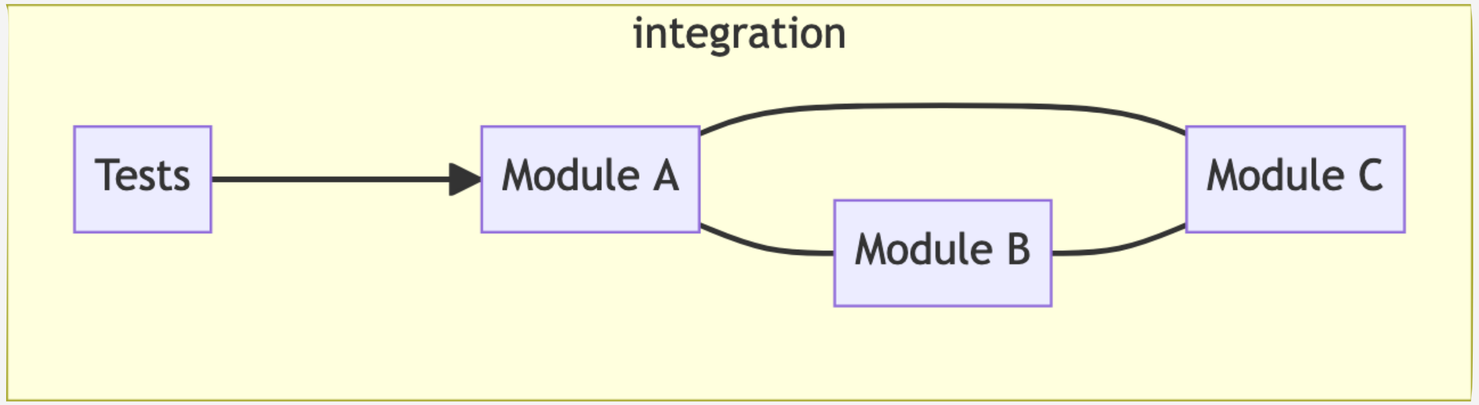


RAPID PROTOTYPING

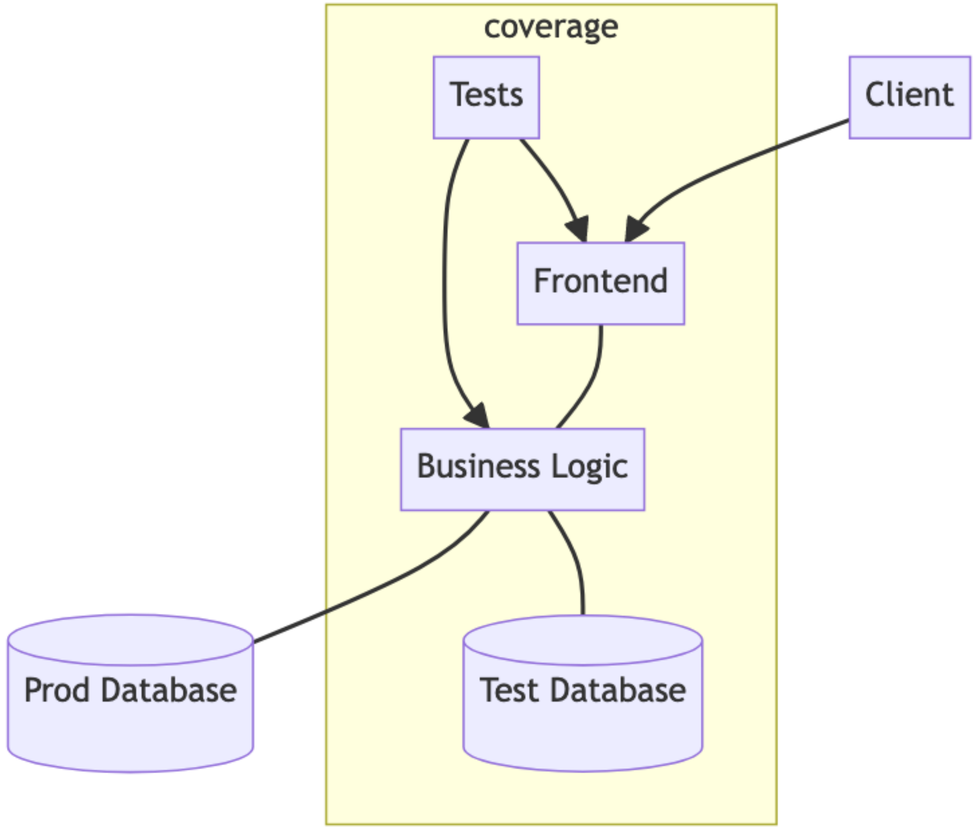
ANSWERING QUESTIONS

Unit test vs. Integration test

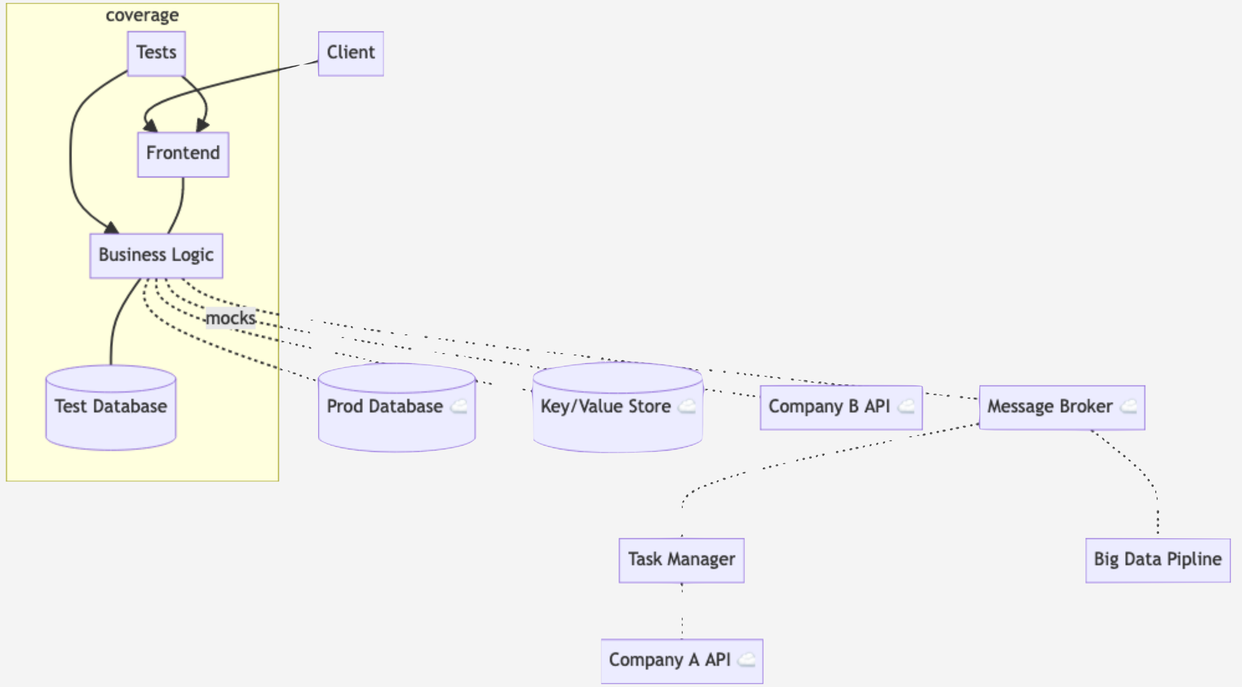




Traditional monolith



2023



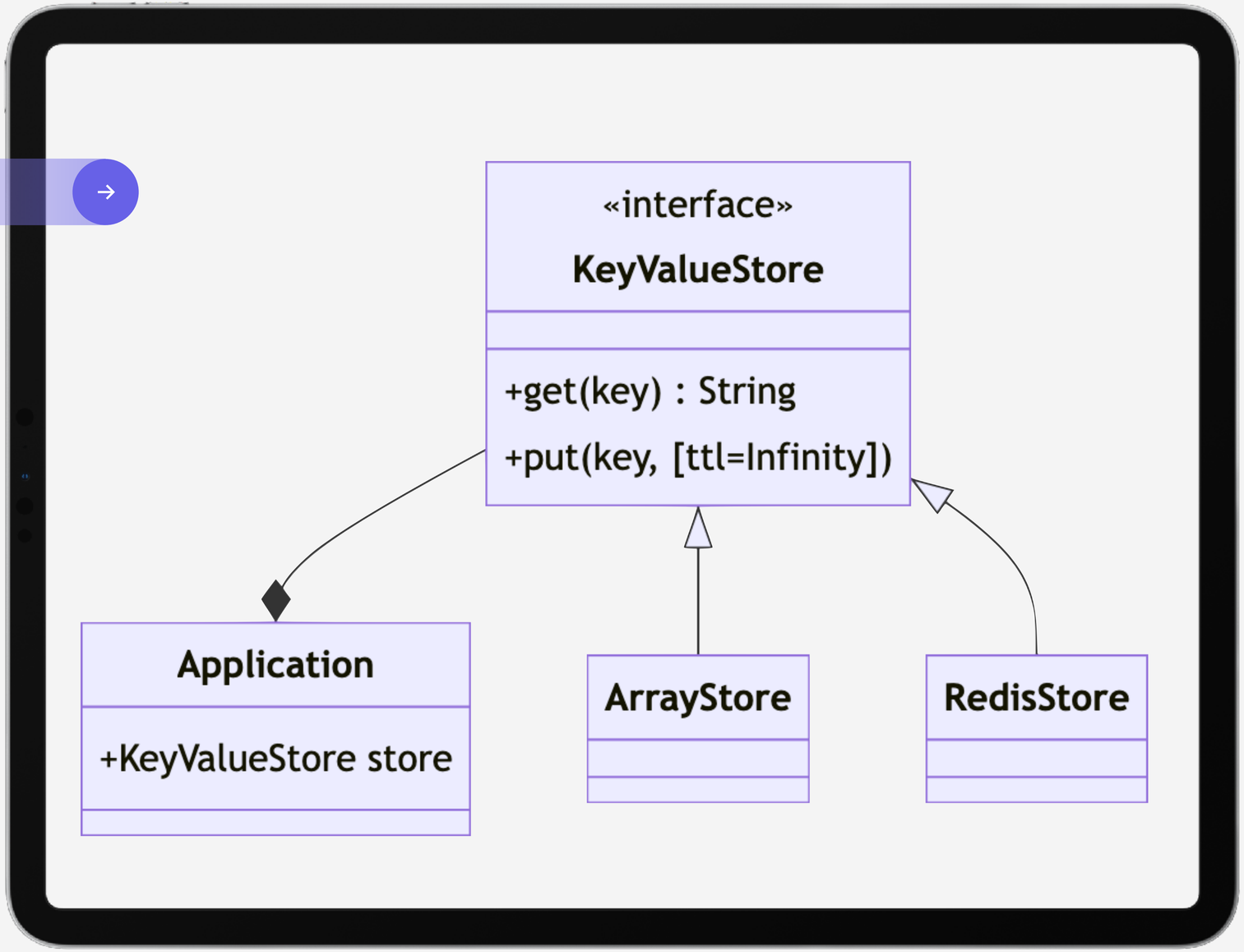
nosto

Mocks

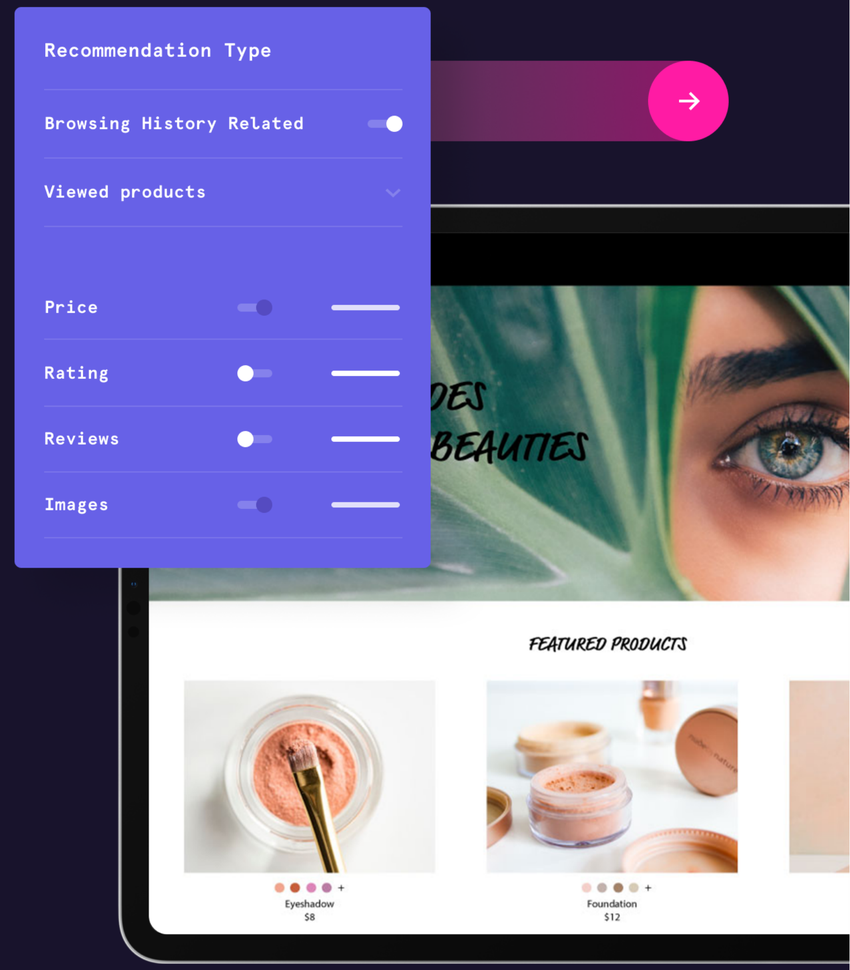
Via Dependency Injection

Other possibilities:

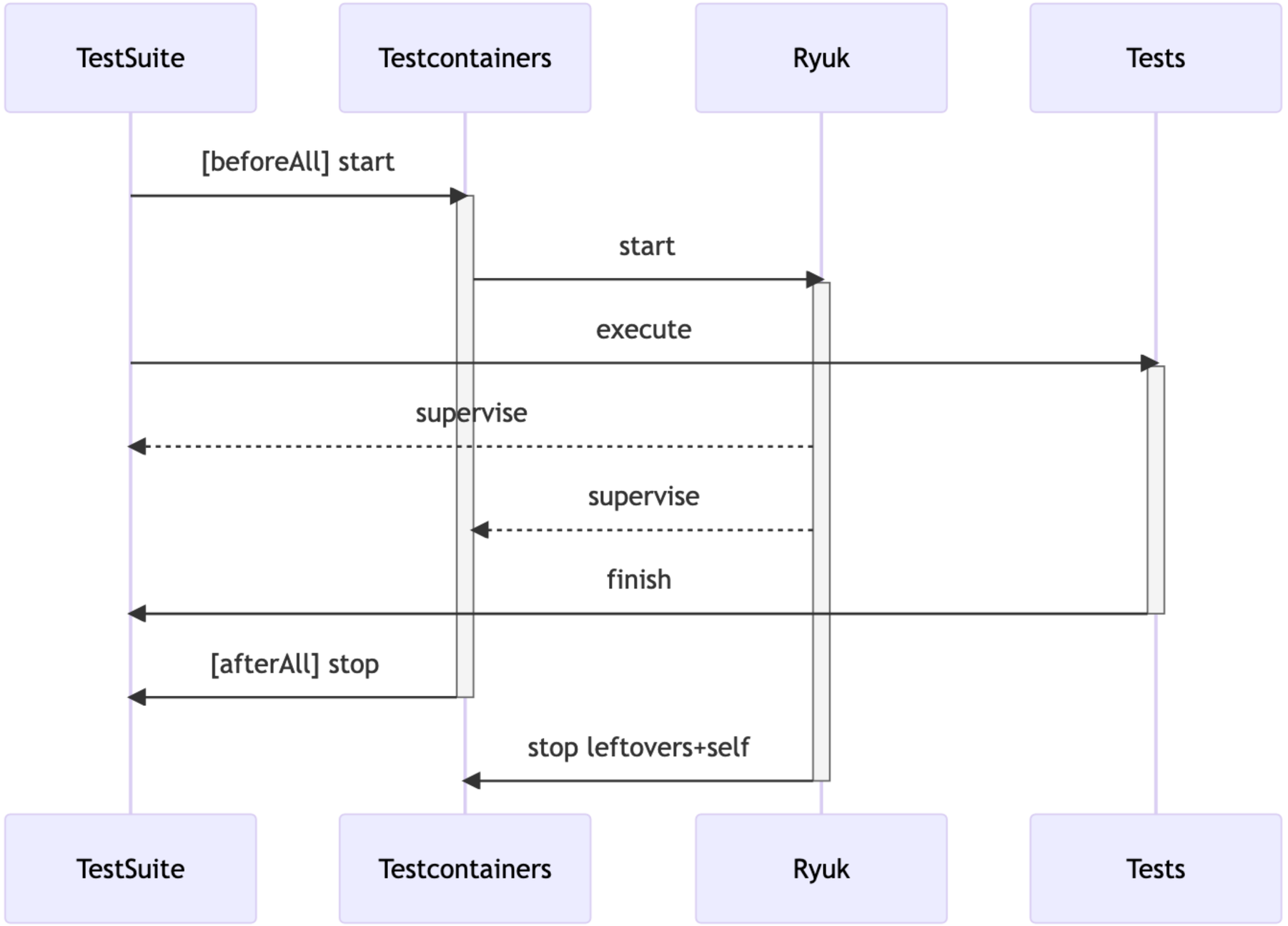
- Module mocks
- Monkey patching
- Code injection
- Spies



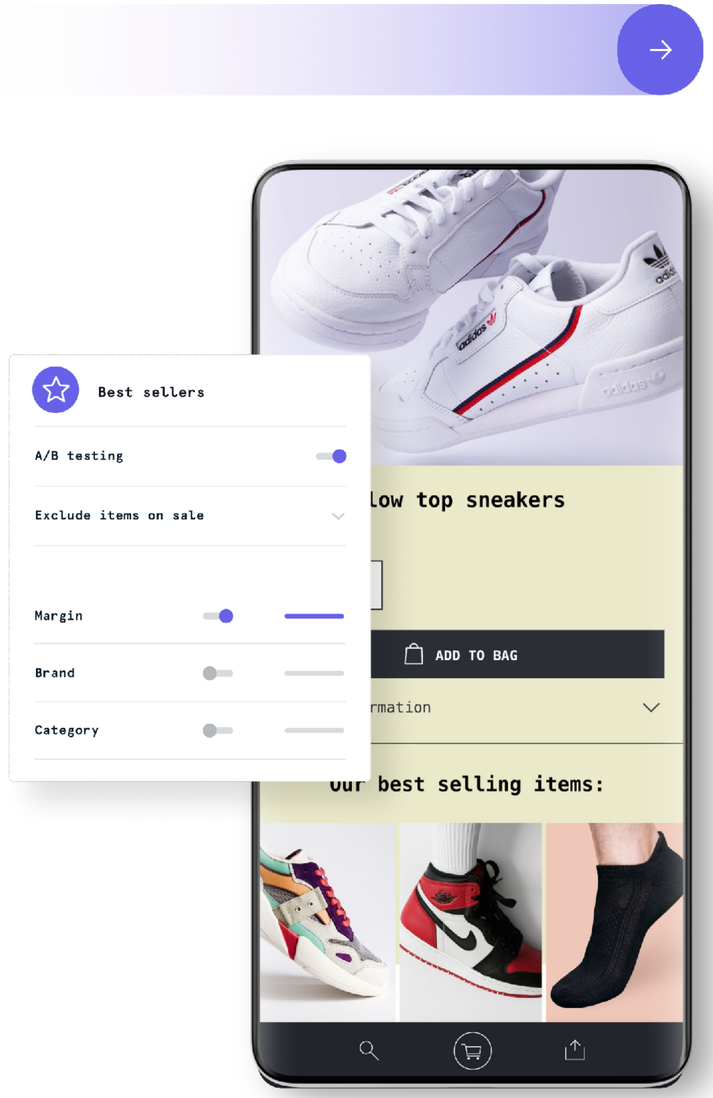
How do Testcontainers work?



nosto


TestSuite
Testcontainers
Ryuk
Tests
[beforeAll] start
start
execute
supervise
supervise
finish
[afterAll] stop
stop leftovers+self
TestSuite
Testcontainers
Ryuk
Tests

Testcontainers in action



nosto

Demo application

Simple Express API for counting terms.

`GET /` (list all terms and counts)

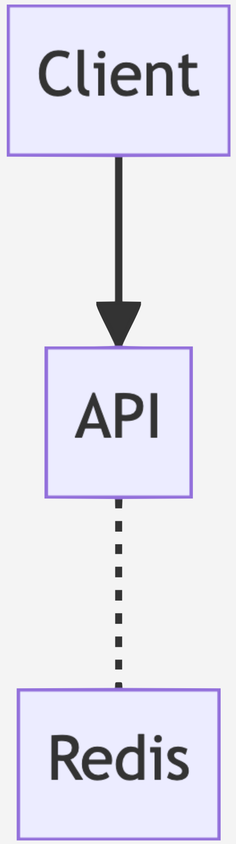
`GET /:term` (list count for this term)

`POST /:term` (increment count for this term)

Example:

```
curl -LX GET mustnot.work/meetup
curl -LX POST mustnot.work/meetup
```

github.com/howard/testcontainers-demo



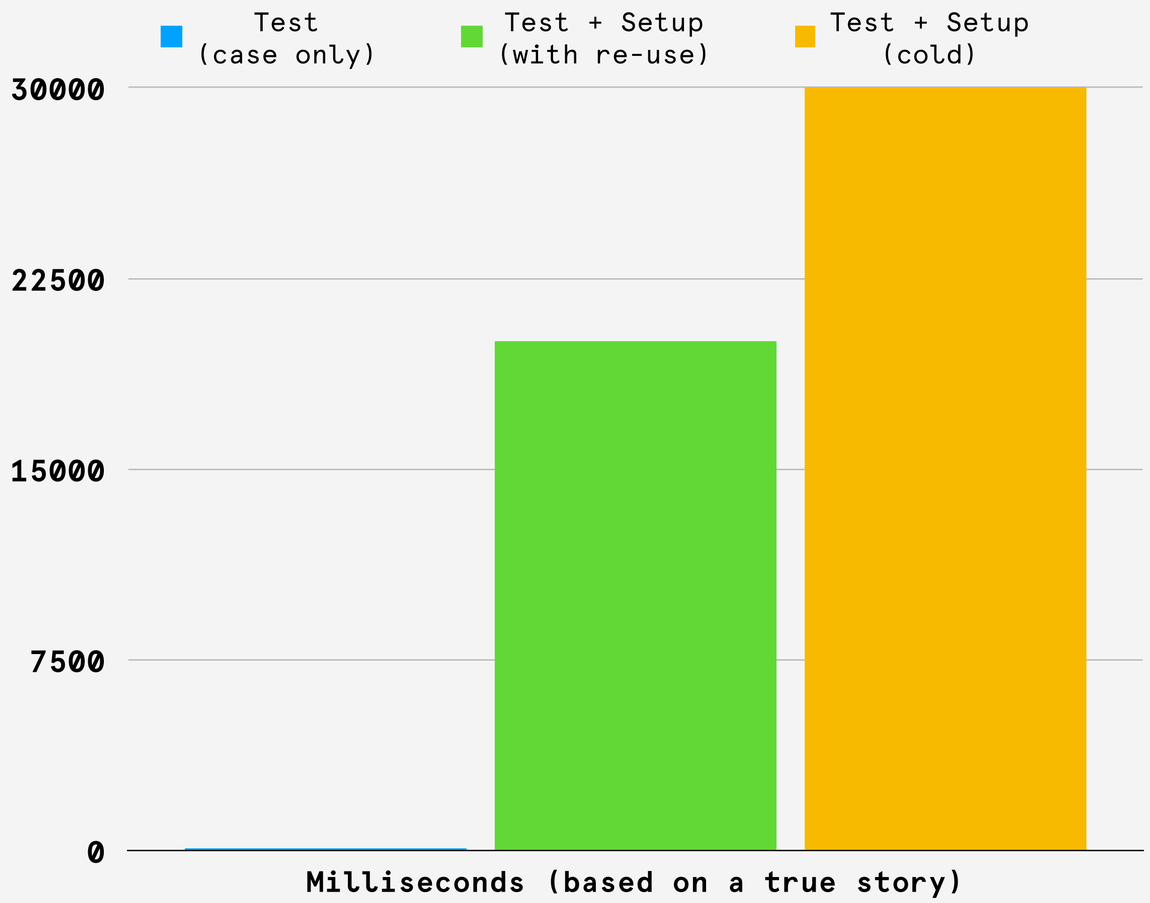
nosto

Testcontainers in the wild

nosto

Speed

- Can be really bad, if not careful
- Beware of:
  - Image builds
  - Bad wait strategies
  - Heavy images
- Mitigate with re-use, in `~/.testcontainers.properties`: `testcontainers.reuse.enable=true`



nosto

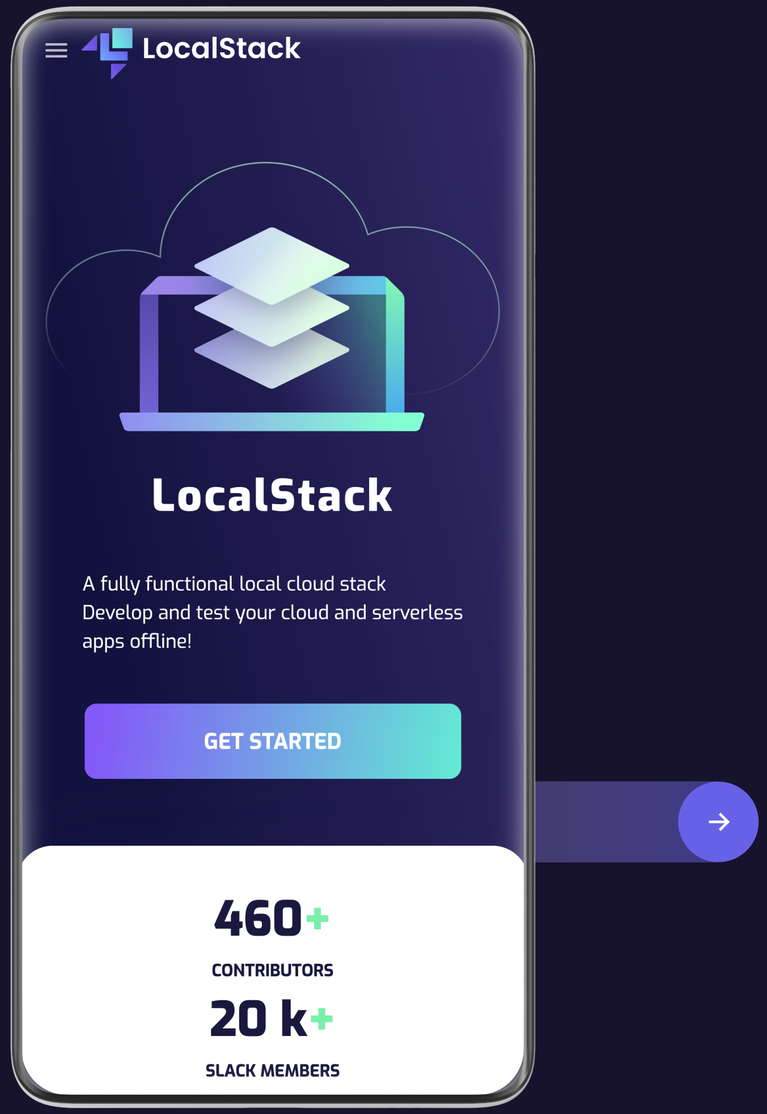
Cloud

LocalStack emulates AWS.

- S3
- EC2
- Kinesis
- ...

Also available:

- Azure
- Google Cloud
- Kafka
- Many more

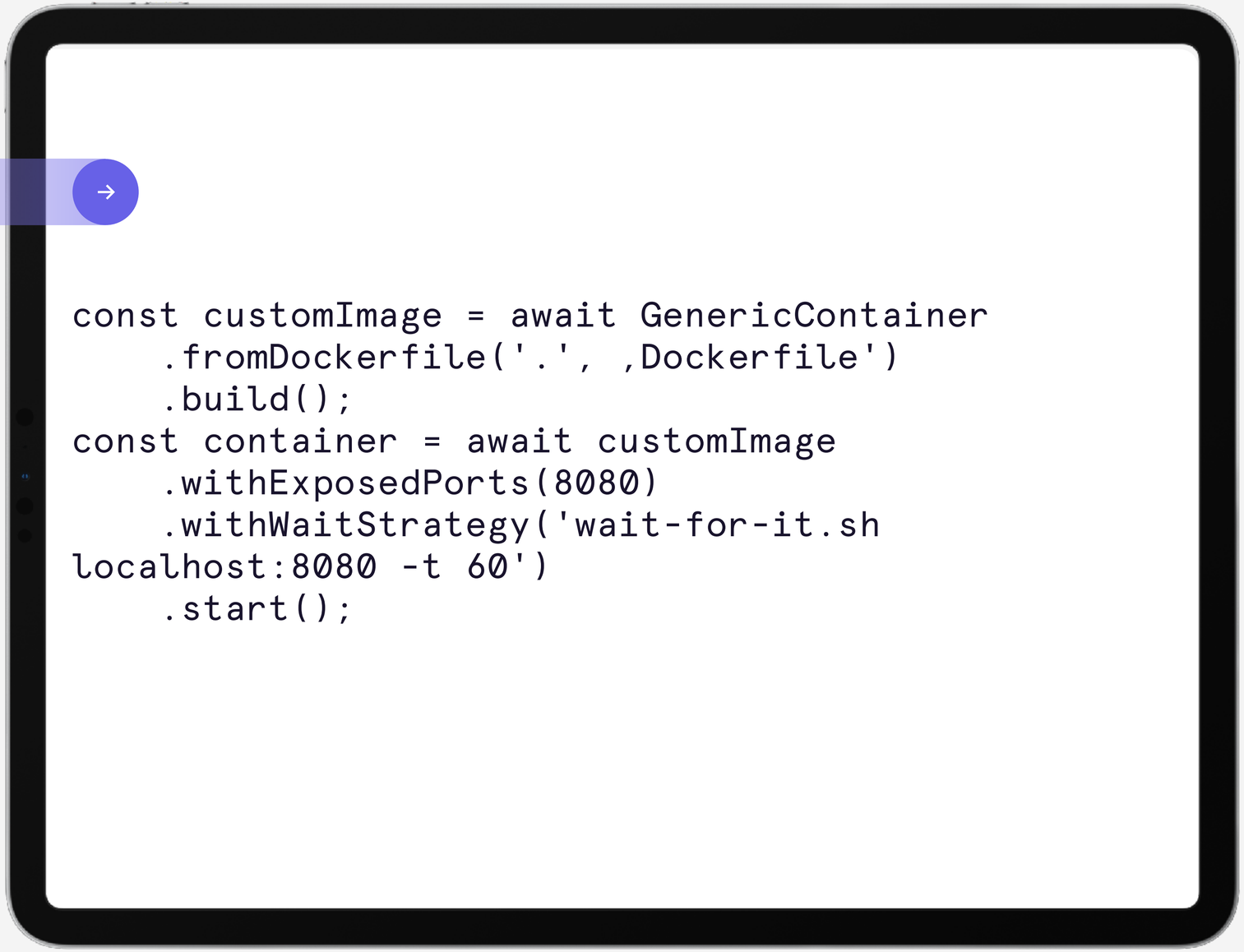


nosto

Your own images

Very easy.

Good wait strategy critical for stable tests.



```
const customImage = await GenericContainer
    .fromDockerfile('.', ,Dockerfile')
    .build();
const container = await customImage
    .withExposedPorts(8080)
    .withWaitStrategy('wait-for-it.sh
localhost:8080 -t 60')
    .start();
```

Thank you!

nosto

NOSTO.COM NOSTO @NOSTOSOLUTIONS NOSTO