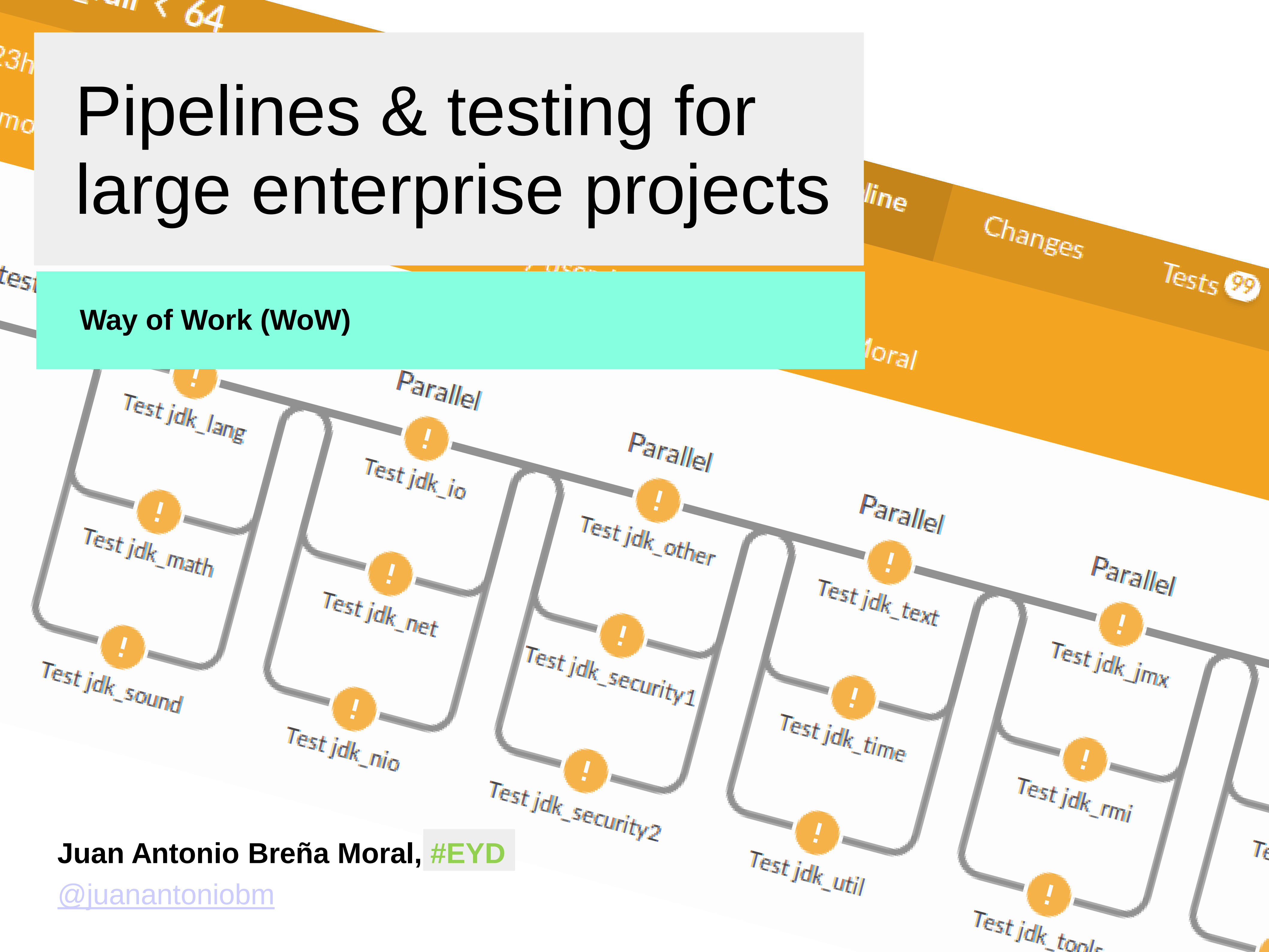


Pipelines & testing for large enterprise projects

Way of Work (WoW)

Juan Antonio Breña Moral, #EYD

@juanantoniobm

Pipelines & testing for large enterprise projects

- Highly productive teams grow their knowledge consciously, practicing continuous learning.
  - Eric Evans (Domain Driving Design)

- Fact 1: The most important factor in software work is not the tools and techniques used by the programmers but rather the quality of the programmers themselves.
  - Robert L. Glass (Object Thinking)

Agenda

- Educational goals
- Concepts
- Develop your Section 9 teams
- The new squad Members: The Pipelines
- Pyramid of Testing
- TDD & Spring Cloud Contract
- Docker & OpenJDK
- Chaos Engineering
- The checklist
- References

Educational goals

- The best asset of any organization is the people
- Pipelines are another digital “member” of your squad
- Spring Cloud Contract offers an interesting way to implement your integration tests
- It is necessary to review the JVM strategy
- Add Chaos experiments are healthy to have a better knowledge about your Distributed system

Concepts

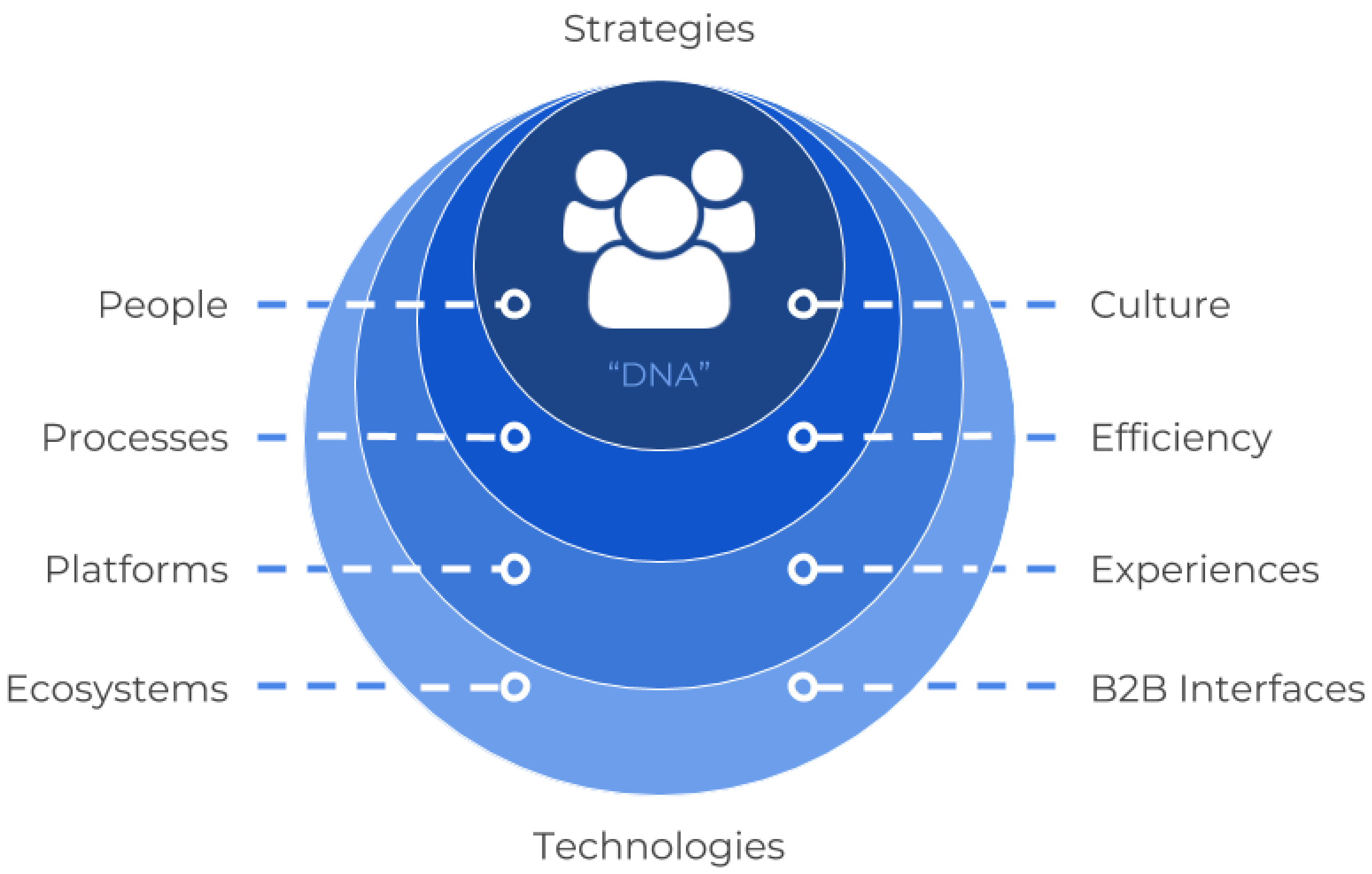
- **Digitization, Digitalization & Digital transformation:** <u>digitization</u> (<u>the conversion</u>, "the conversion of analog information into digital form"), <u>digitalization</u> (<u>the process</u>, “digitalization is the actual 'process' of the technologically-induced change within these industries”) and <u>the digital transformation</u> (<u>the effect</u>, "the total and overall societal effect of digitalization") that are collectively accelerating the global and societal transformation process.
  - *Wikipedia*

Concepts

- **Legacy code:** Code without tests
  - *Michael Feathers*

- **Distributed System:** A distributed system is a system whose components are located on different networked computers, which then communicate and coordinate their actions by passing messages to one other.
  - *Wikipedia*

Concepts

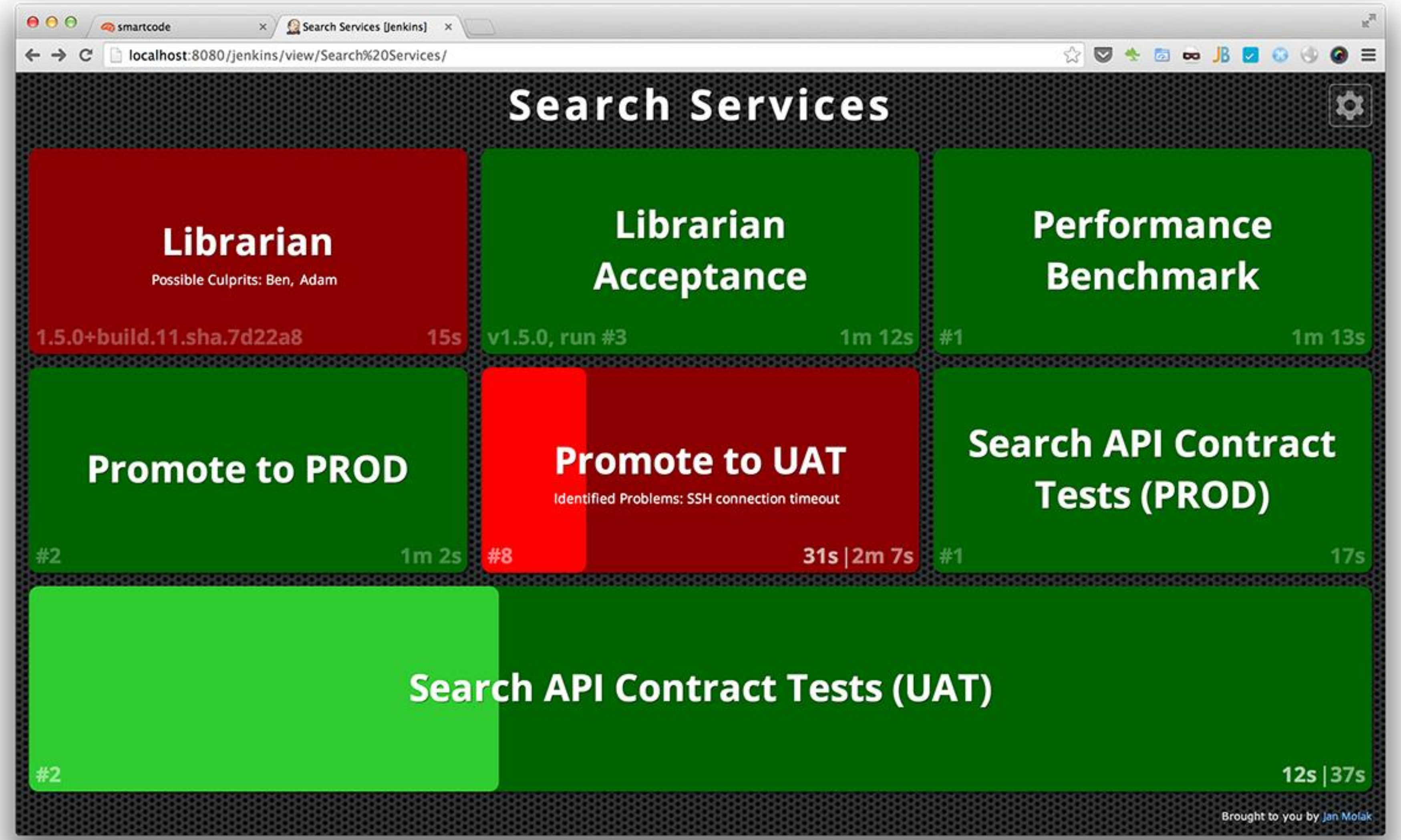

Strategies
People
Processes
Platforms
Ecosystems
"DNA"
Culture
Efficiency
Experiences
B2B Interfaces
Technologies

Develop your Section 9 teams

Develop your section 9 teams

- Every squad should have a Display with information about:
  - Current builds
  - Iteration progress
- Identify Technical debt in the repositories
- Review production support documentation
  - Has everyone the same level of knowledge?
    - Do you have the documentation in some repository or system?

The new squad Members: The Pipelines

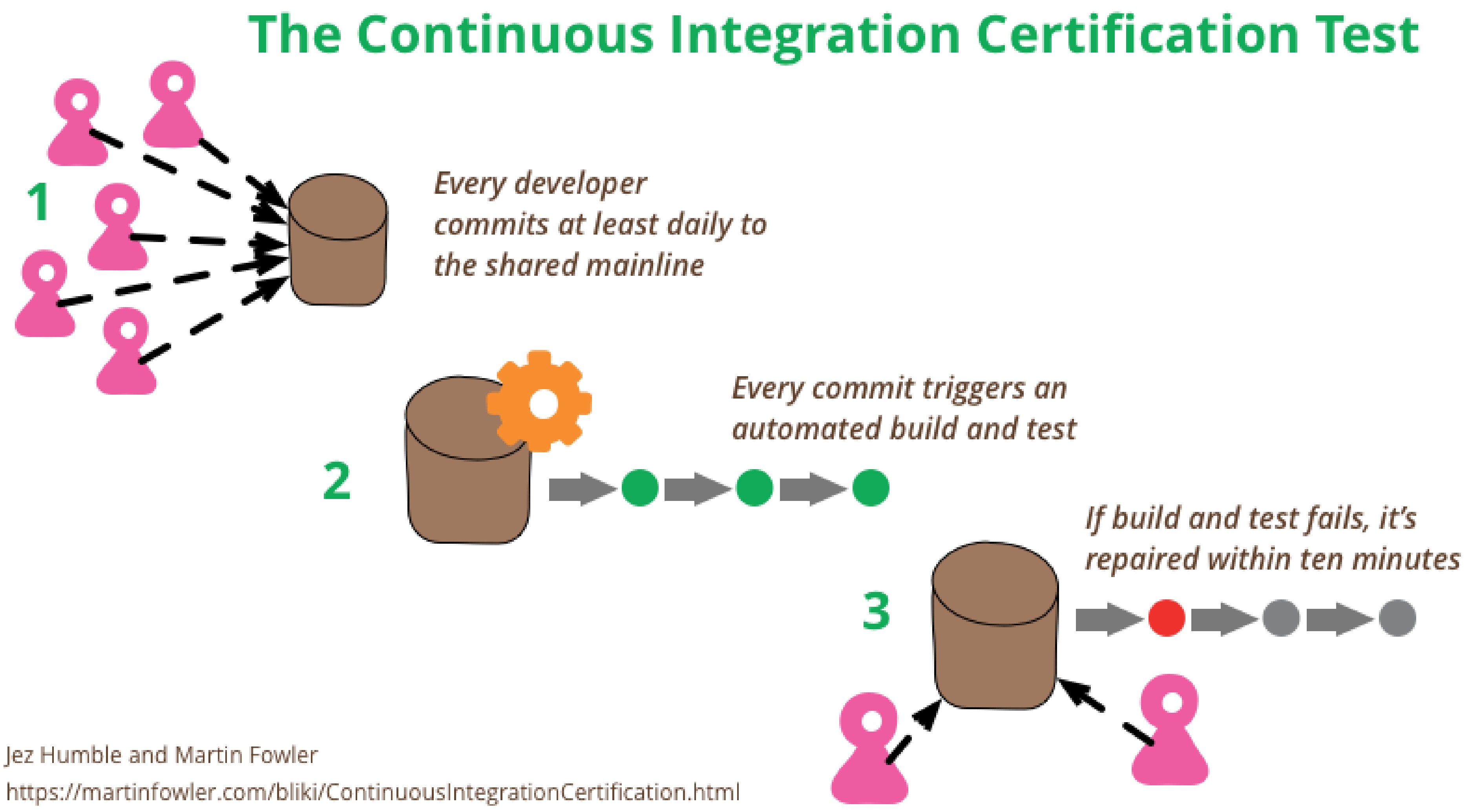

smartcode
Search Services [Jenkins]
localhost:8080/jenkins/view/Search%20Services/

Search Services

Librarian Possible Culprits: Ben, Adam 1.5.0+build.11.sha.7d22a8 — 15s	Librarian Acceptance v1.5.0, run #3 — 1m 12s	Performance Benchmark #1 — 1m 13s
Promote to PROD #2 — 1m 2s	Promote to UAT Identified Problems: SSH connection timeout #8 — 31s | 2m 7s	Search API Contract Tests (PROD) #1 — 17s
Search API Contract Tests (UAT) #2 — 12s | 37s		

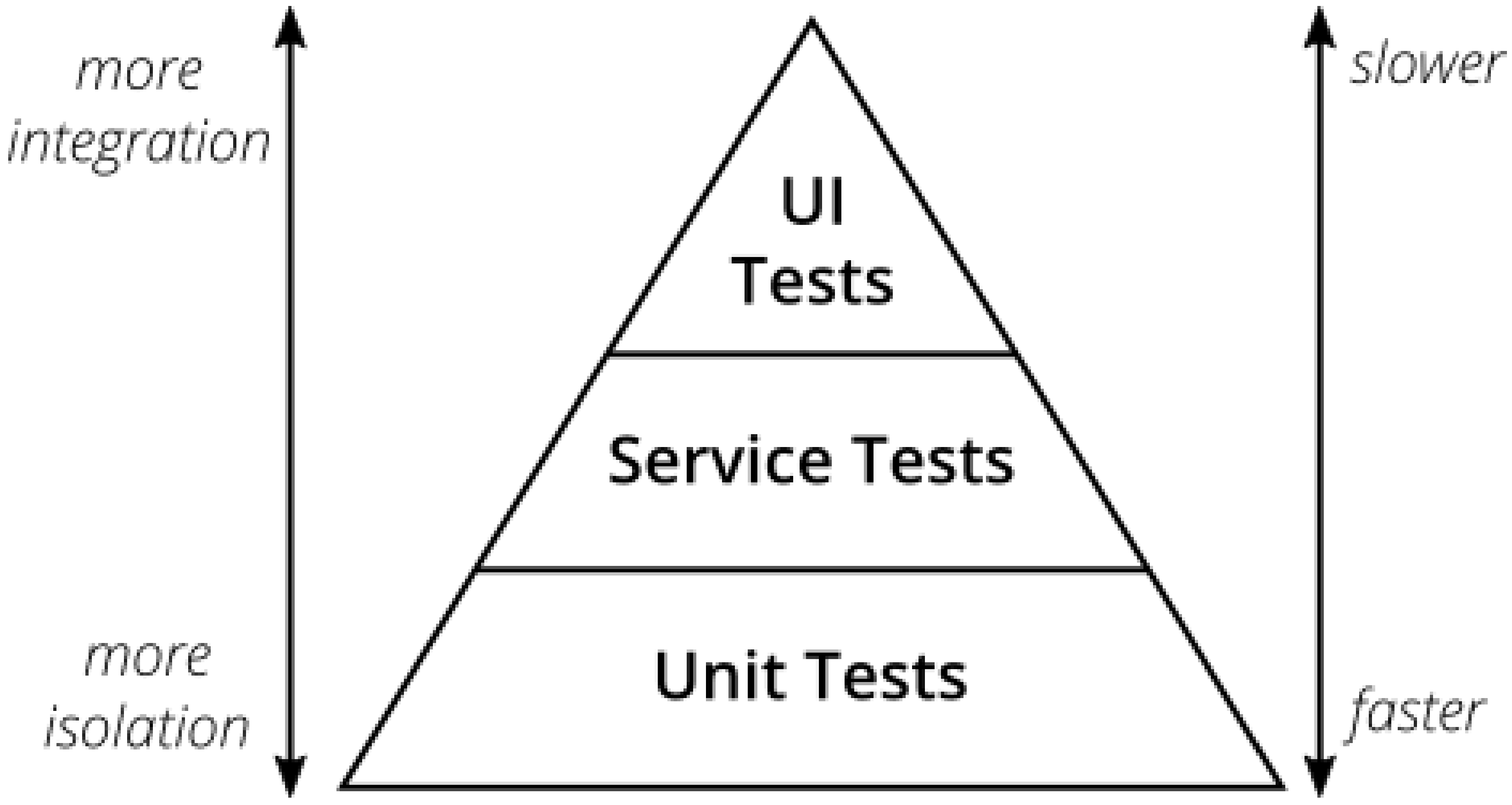
Brought to you by Jan Molak

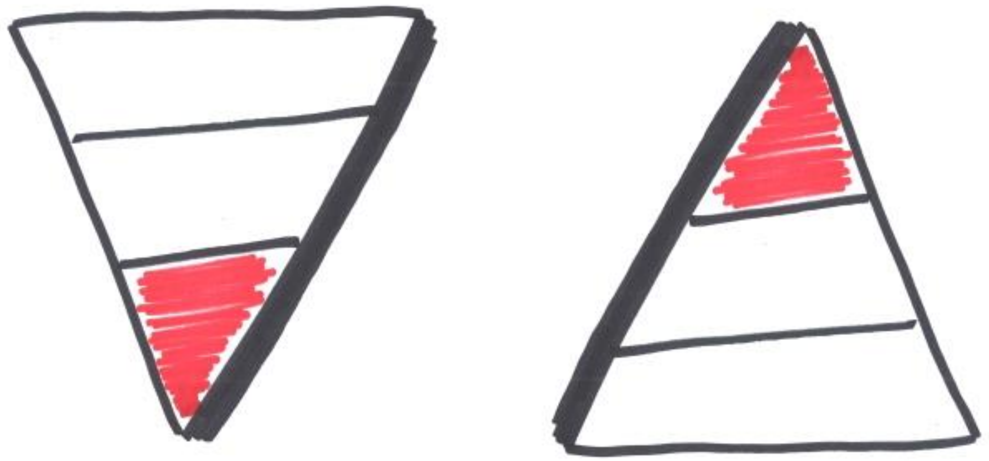
The new squad Members: The Pipelines


The Continuous Integration Certification Test
1
Every developer commits at least daily to the shared mainline
2
Every commit triggers an automated build and test
3
If build and test fails, it's repaired within ten minutes

Jez Humble and Martin Fowler
https://martinfowler.com/bliki/ContinuousIntegrationCertification.html

Pyramid of Testing



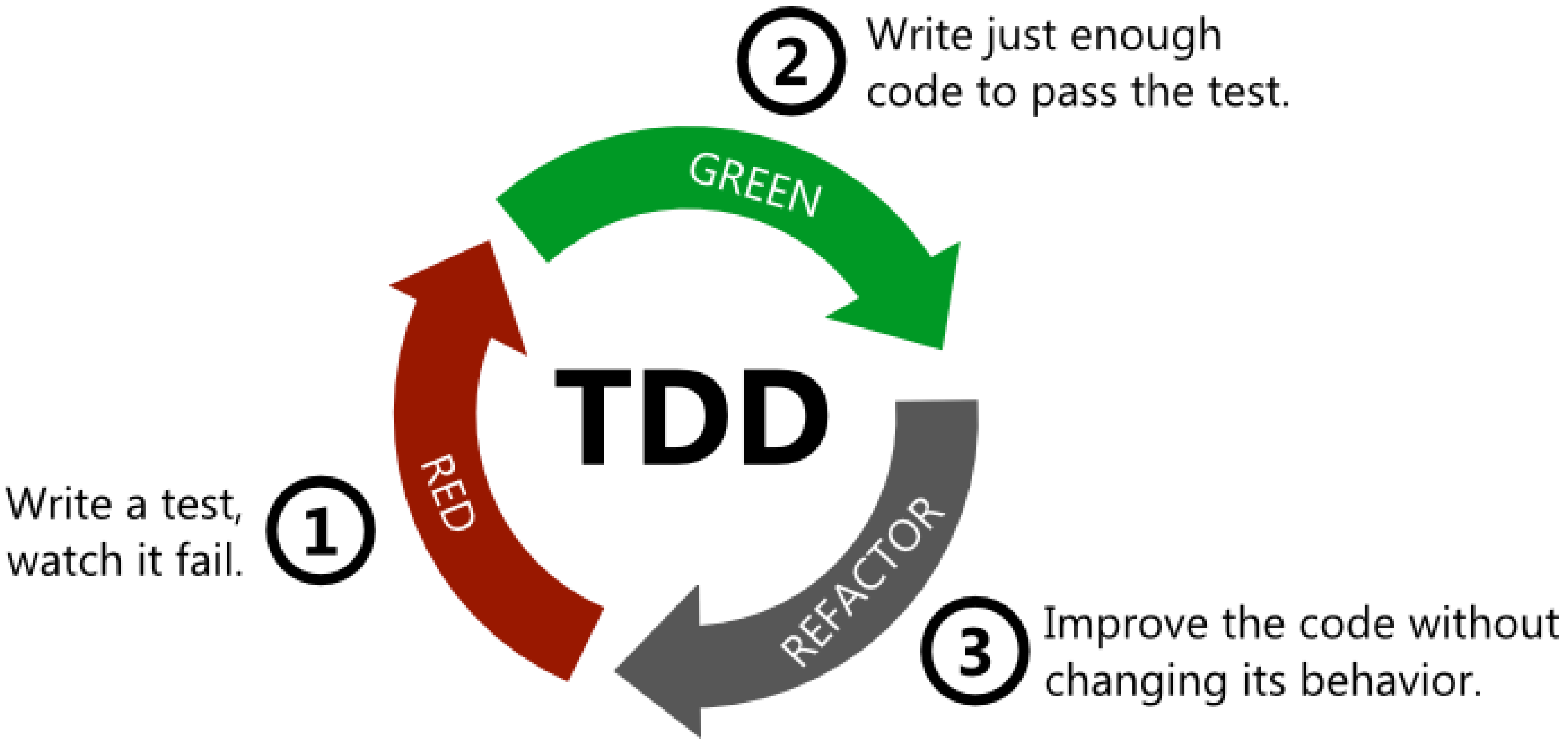


“Invert the Shape of Pyramid of Testing is a real challenge for any big organization”

https://martinfowler.com/articles/microservice-testing/

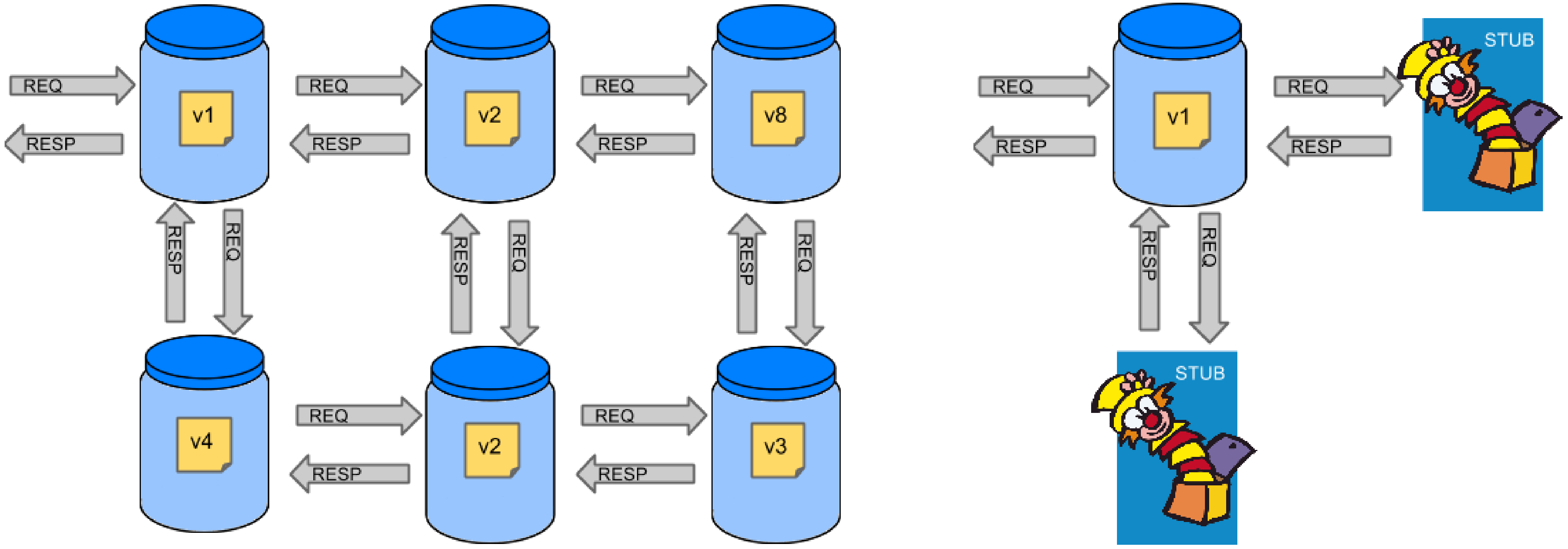
TDD & Spring Cloud Contract

- Spring Cloud Contract Verifier moves TDD to the level of software architecture.


2
Write just enough code to pass the test.
GREEN
TDD
RED
Write a test, watch it fail.
1
REFACTOR
3
Improve the code without changing its behavior.

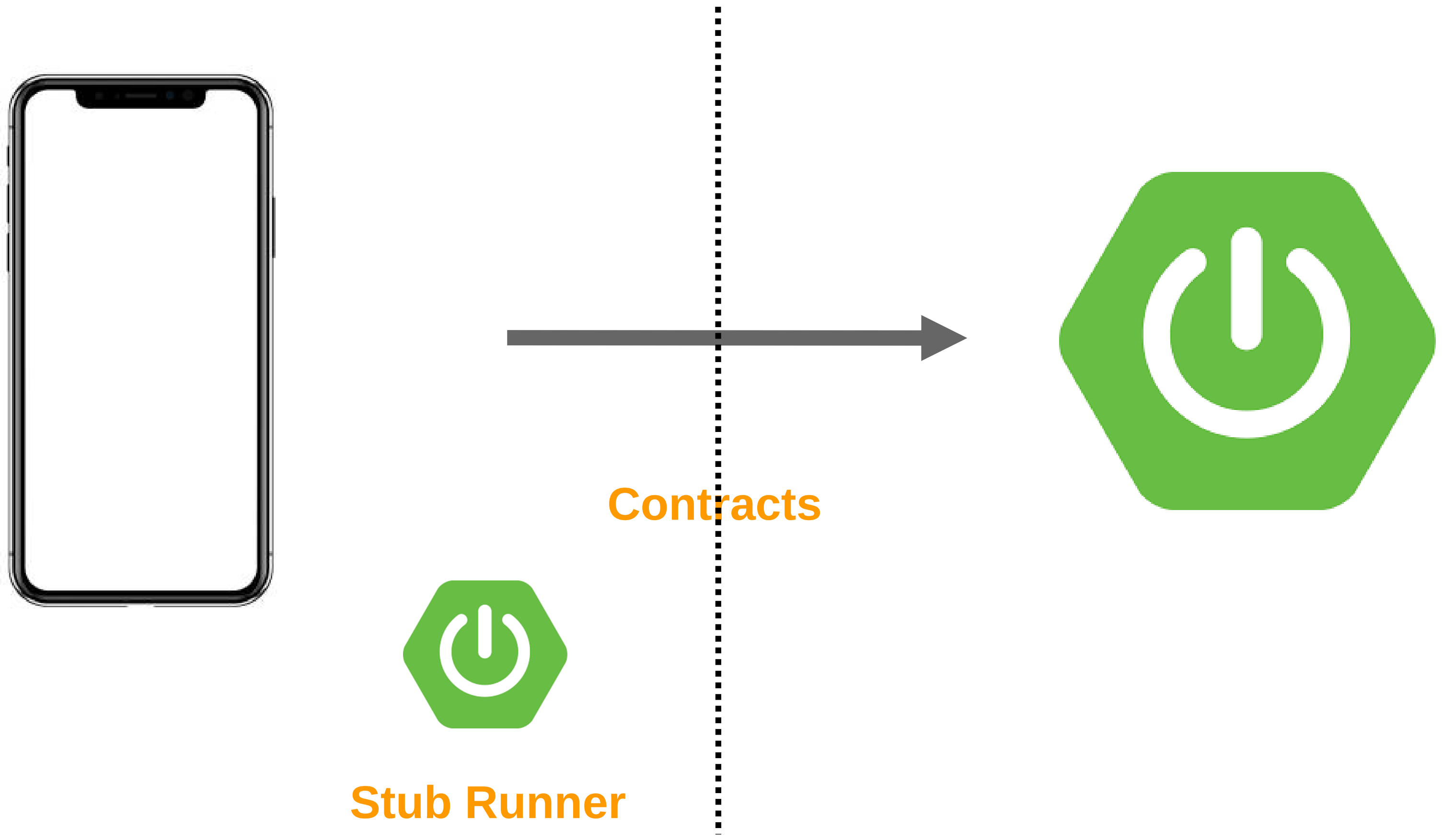
TDD & Spring Cloud Contract

- Original idea:



TDD & Spring Cloud Contract

- Scenario Mobile Application



TDD & Spring Cloud Contract

```
org.springframework.cloud.contract.spec.Contract.make {
    description "should return even when number input is even"
    request{
        method GET()
        url("/api/concept1") {

        }
        headers {
            header 'Content-Type', 'application/json'
        }
    }
    response {
        status 200
        headers {
            headers {
                header 'Content-Type', 'application/json'
            }
        }
        body( """
         {
            "status": "OK"
         }
         """)
    }
}
```

TDD & Spring Cloud Contract

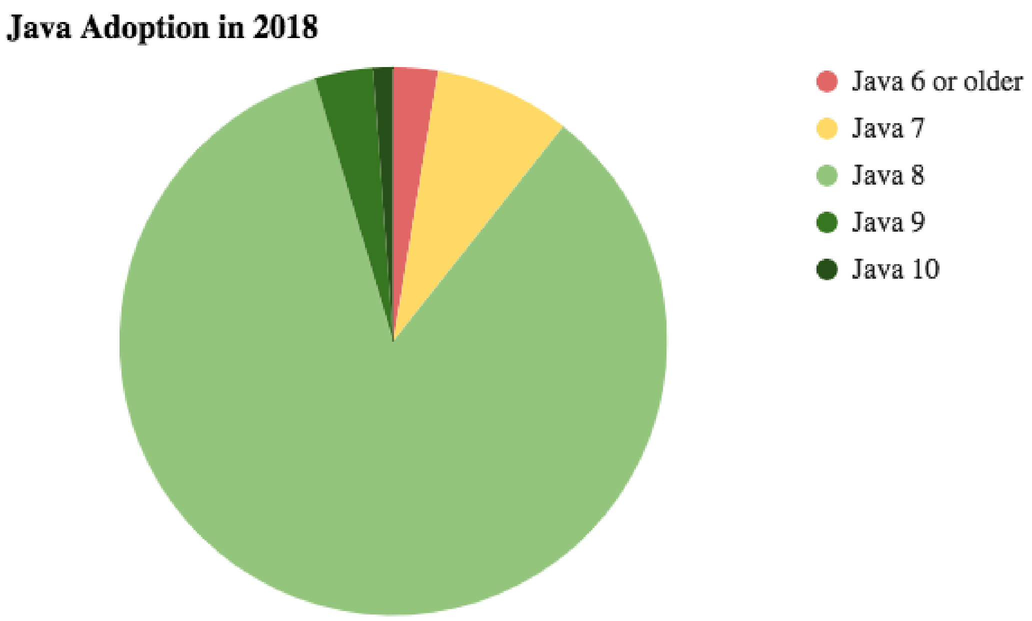
- Scenario Mobile Application
  - SCC decouple the development for both squads
  - With a set “3 amigos” sessions is enough to define the initial contracts
  - The contracts provide integration tests executed by the new dependency:
    - ***spring-cloud-contract-verifier***

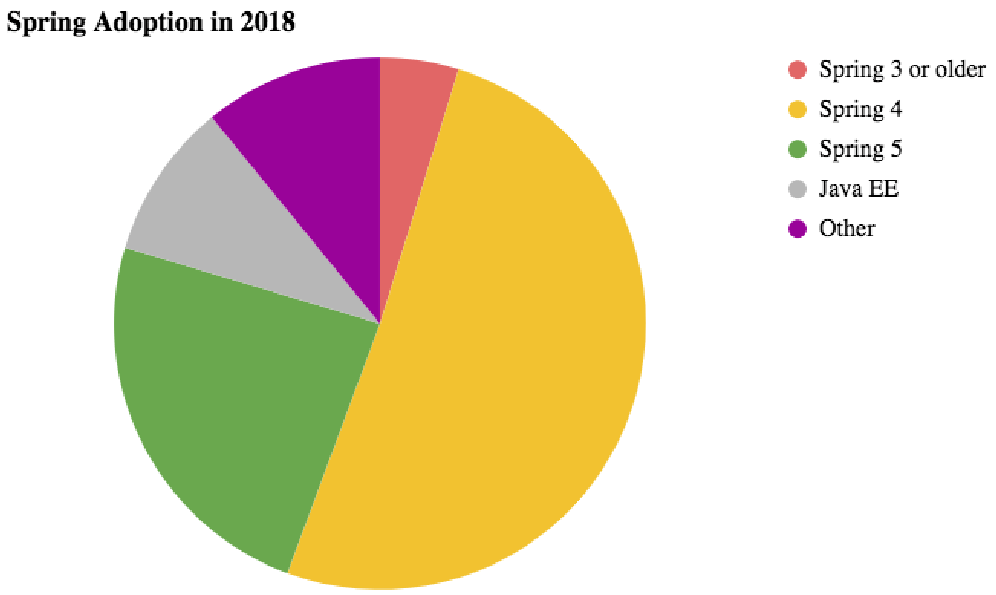
TDD & Spring Cloud Contract

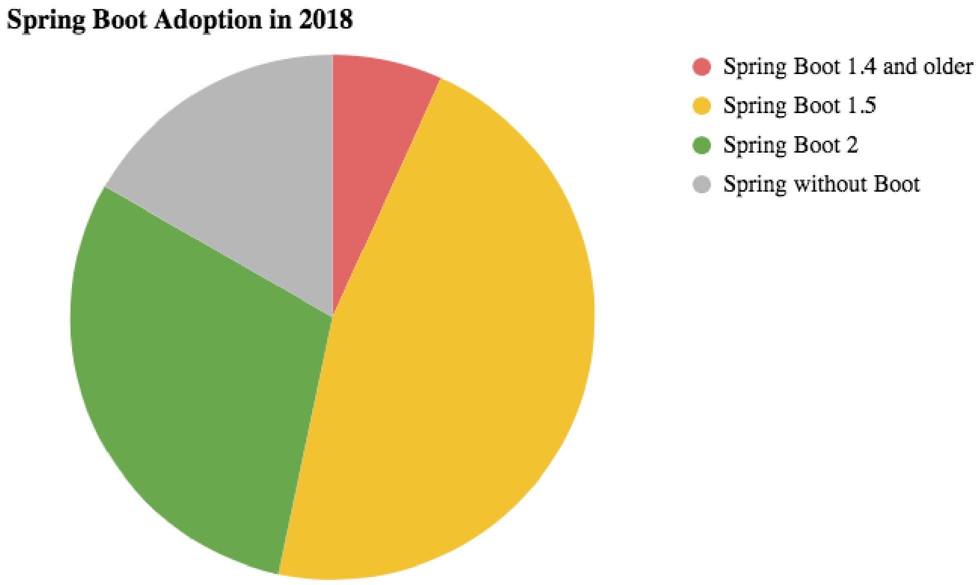
- Scenario API

Contracts
Contracts

Docker & OpenJDK

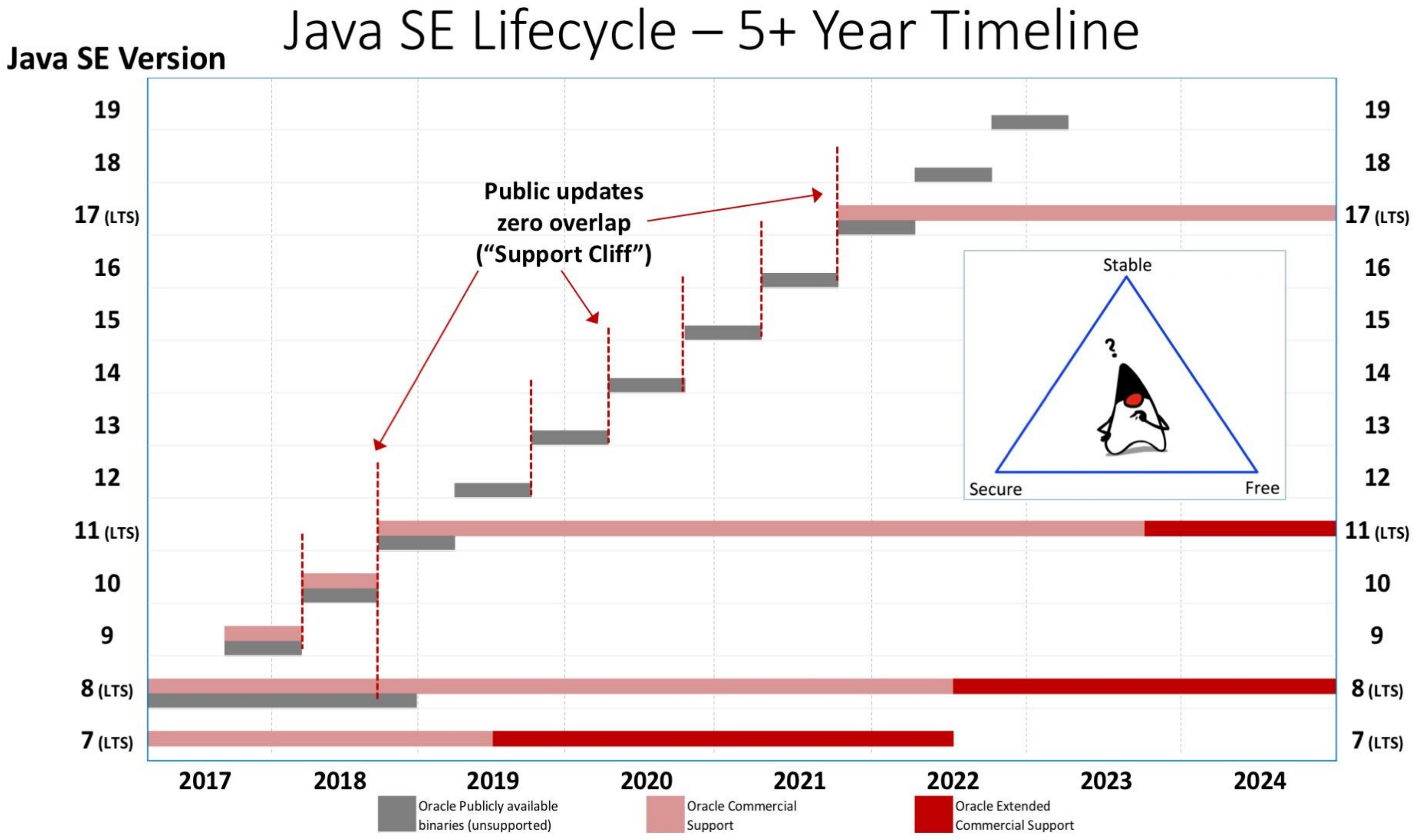






Docker & OpenJDK

- JEP 322: Time-Based Release Versioning



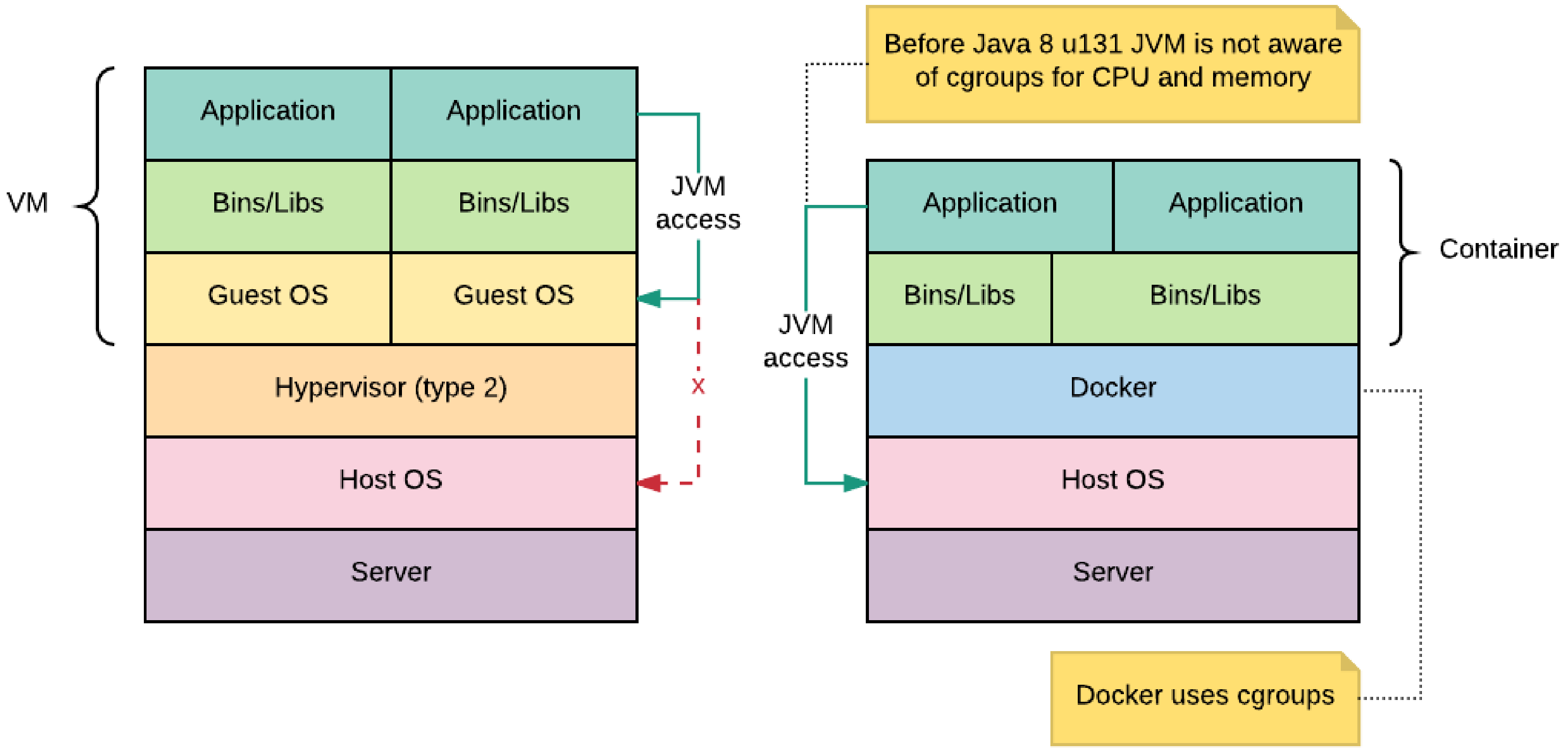
Docker & OpenJDK

Enterprise **Desktop** **Education**

AdoptOpenJDK

Docker & OpenJDK

- JDK-8196595: Java Improvements for Docker Containers



Docker & OpenJDK

- JDK-8196595: Java Improvements for Docker Containers
  - adhering to memory limits set in the container
  - setting available cpus in the container
  - setting cpu constraints in the container

```
docker container run -it -m512 --entrypoint bash openjdk:latest
```

```
docker container run -it --cpu-shares 2048 openjdk:10-jdk
```

```
docker run -it --cpuset-cpus="1,2,3" openjdk:10-jdk
```

Chaos Engineering

- Building confidence in system behavior through experiments

I Plan an Experiment
II Learn from a test
III Scale an Experiment

Chaos Engineering

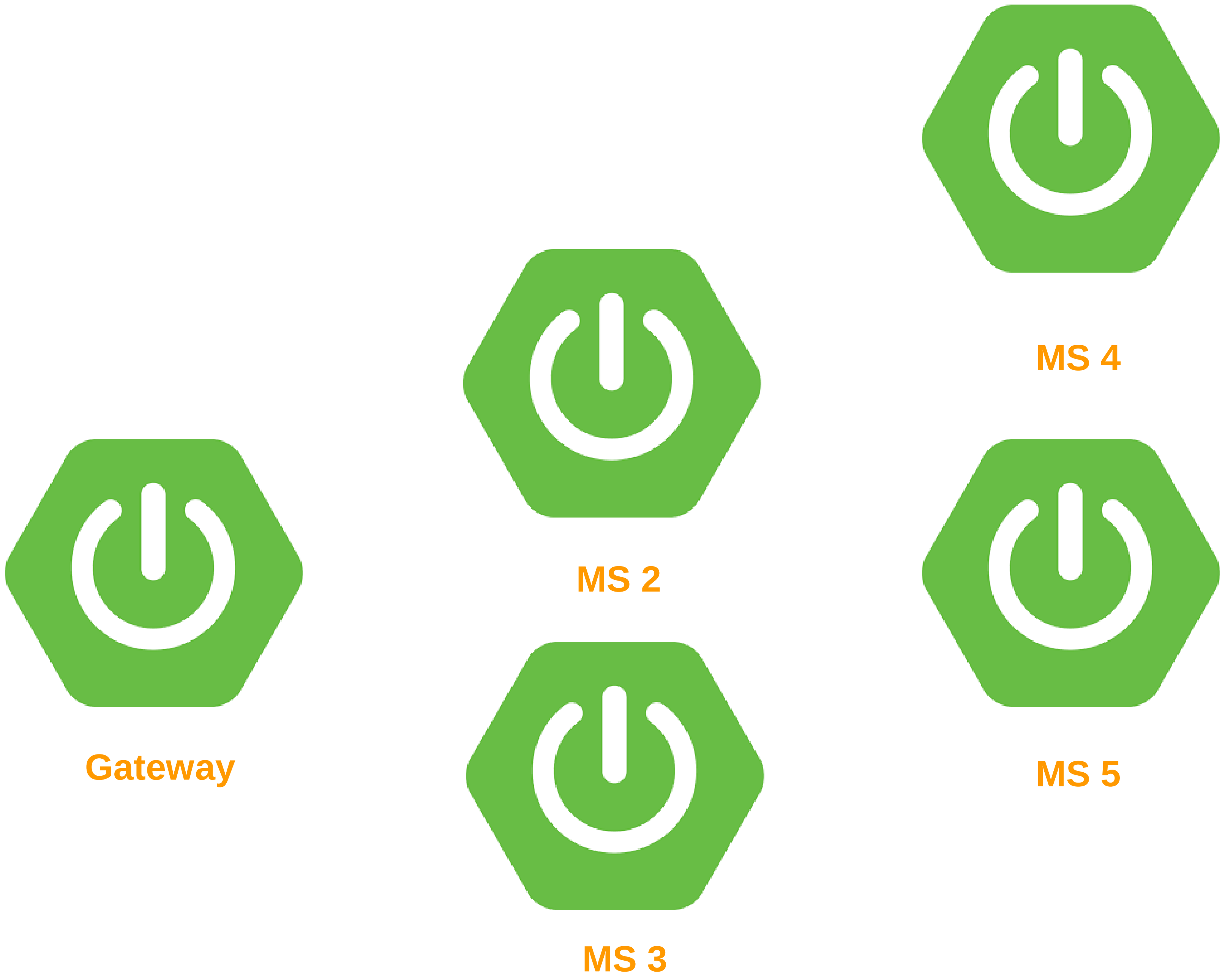
Chaos Engineering is the discipline of experimenting on a distributed system in order to build confidence in the system’s capability to withstand turbulent conditions in production.

Principles of Chaos

Chaos Engineering

- Simulating the failure of an entire region or datacenter.
- Injecting latency between services for a select percentage of traffic over a predetermined period of time.
- Function-based chaos (runtime injection): randomly causing functions to throw exceptions.
- Executing a routine in driver code emulating I/O errors.

Chaos Engineering



The checklist

- Review the usage of CI Dashboards in the daily work by anyone in the squad
- Review Docker images and JVM versions
- Review your request POJOS to improve the security. (JSR 303)
- Review the configuration of Beans to ensure that you have configured timeouts for your http connections
- Review squad trainings process to improve the coding skills
- Organize sessions with multiple squads to execute chaos experiments with your Microservices

References

- https://www.amazon.com/FEATHERS-WORK-EFFECT-LEG-CODE/dp/0131177052
- https://martinfowler.com/bliki/ContinuousIntegrationCertification.html
- https://github.com/spring-cloud-samples/spring-cloud-contract-samples/
- https://www.baeldung.com/java-in-2018
- https://blog.docker.com/2018/04/improved-docker-container-integration-with-java-10/
- https://adoptopenjdk.net/
- https://medium.com/chaos-toolkit/chaos-toolkit-loves-chaos-monkey-for-spring-boot-548352985c8f

Thanks