Continuous Testing

TEST!
regularly!
automated!

Christian Kühn

Software - Developer

Technologie und Architektur Verantwortlicher

spreading happiness in teams for 15+yrs

Interested in:
Security, DevOops, Automation

chris@clowncomputing.de
@CYxChris

dmTECH GmbH

100% subsidiary of dm

full corporate IT
Hardware
Software
Operations

Why should we even test?

learn about the current state

compare to the desired state

"shift left" - learn while actively developing

WARNING!?

"Program testing can be used to show the presence of bugs,
but never to show their absence!"

- Edsger W. Dijkstra

Continuous Integration

"In software engineering, continuous integration (CI)
is the practice of merging all developers' working copies
to a shared mainline several times a day"

(Wikipedia)

Continuous Integration

push

commit

test

feedback

code

transparency

automated

integration

approval

merge

build

deployment

shift left

fail

observability

review

release

Continuous Testing

"Reducing wasted development time via continuous testing"

14th. IEEE International Symposium on Software Reliability Engineering
ISSRE 2003

Continuous Testing

testing as essential part of continuous integration

get to know about problems earlier

react quicker

replace manual testing

Continuous Testing

push

commit

test

feedback

code

transparency

automated

integration

approval

merge

build

deployment

shift left

fail

observability

review

release

regularly check on your classifications

as much testing as possible

as few testing as minimally needed 😇

what and how can we actually test?

Cucumber

behaviour driven development

development process in close communication with all stakeholders

client defines processes together with dev, QA and other Teams

generate a process- and test-description that is "readable" for non-techies

develop the process- and test-spec together

client:
describes business processes

development team:
works on a solution

Q&A-Team:
support dev, ask the important questions "What if?"

describe the test specification in

Given - When - Then

notation

```
Scenario: Customer tries to log in with their password
  Given customer starts checkout fresh process
  When customer with Id "12345" tries to authenticate with pass "hunter2"
  Then the auth-service answers with state "authenticated"
```

cucumber

https://www.cucumber.io

cucumber

```
Scenario: Customer tries to log in with their password
  Given customer starts checkout fresh process
  When customer with Id "12345" tries to authenticate with pass "hunter2"
  Then the auth-service answers with state "authenticated"
```

```java
@Given("customer starts checkout fresh process")
public void prepareCheckout() {
    prepare();
}

@When("customer with Id \"login\" tries to authenticate with pass \"pass\"")
public void authenticate(String login, String pass) {

    RequestSpecification request = given();
    addAuthorizedAuthentication(request);

    Map<Object, Object> authenticateRequest = new HashMap<>();
    authenticateRequest.put("login", login);
    authenticateRequest.put("pass", pass);
    request.contentType(ContentType.JSON).body(authenticateRequest);

    responseHolder.setHttpResponse(
            request
                    .post( s: "/api/authenticate")
                    .andReturn());

     authentication.setIsAuthenticated(responseHolder.getHttpResponse().jsonPath().get("isAuthenticated"));
}

@Then("the auth-service answers with state \"authenticated\"")
public void assertSuccessfulAuthentication() {
    assertThat(authentication.isAuhenticated()).isTrue();

}
```

BDD in Continuous Integration / Testing?

replace manual tests (business logic)

run BDD with TDD (test-driven dev)

makes errors in complex business cases
better understandable for "other" developers

Gatling

continuous performance testing

in every build

different scenarios
depending on build type
(PR vs. main vs. nightly etc)

compare history

Performance - Testing

"Load Test as Code"

Scala - based (Netty / Akka)

Open Source (Apache 2.0)

metric-Export*

https://gatling.io

Scenario

test case description

```
val successfulAuthentication: ScenarioBuilder = scenario("Successful Authenticate")
        .exec(http("auth")
                .post("/api/authenticate")
                .body(StringBody("""{ "account": "12345", "pass": "hunter2" }""")).asJson
                .check(status.is(200))

val failedAuthentication: ScenarioBuilder = scenario("Failed Authenticate")
        .exec(http("auth")
                .post("/api/authenticate")
                .body(StringBody("""{ "account": "12345", "pass": "passw0rd" }""")).asJson
                .check(status.is(401)))
```

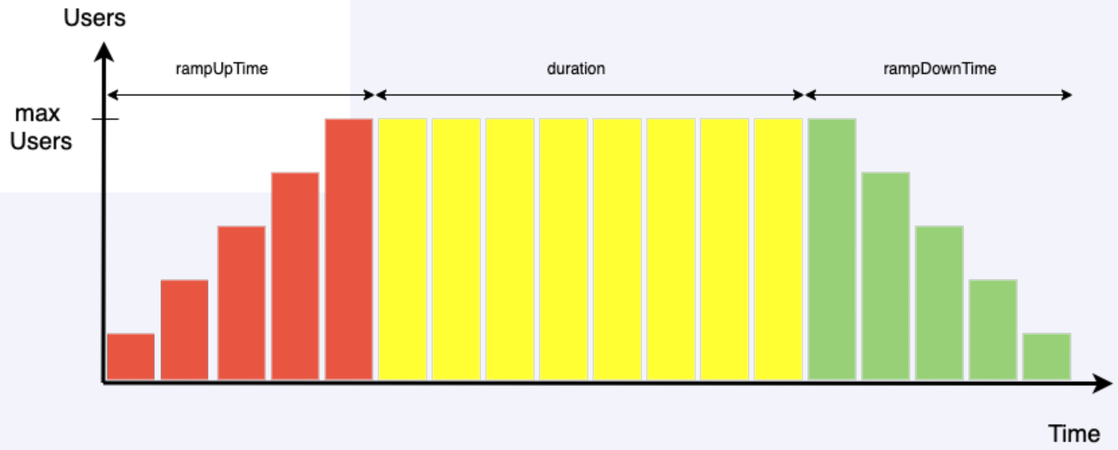
Scenario (2)

combine scenarios

```
val authenticateCombined: ScenarioBuilder = scenario("Authenticate combo")
        .randomSwitch(
            85.0 -> exec(successfulAuthentication),
            15.0 -> exec(failedAuthentication))
```

Simulation (example)

```
setUp(
    AuthenticateScenarios.authenticateCombined.inject(
        rampUsersPerSec(0) to maxUsersPerSecond during (rampupTime seconds),
        constantUsersPerSec(maxUsersPerSecond) during(durationTime seconds) randomized,
        rampUsersPerSec(maxUsersPerSecond) to 0 during (rampdownTime seconds))
).protocols(httpConfig)
        .assertions(
            Seq.apply(global.failedRequests.percent.lte(0.05))
            ++ AuthenticateScenarios.timingAssertions
        )
```




```
val httpConfig: HttpProtocolBuilder = http
        .baseUrl("http://localhost:8080")
        .userAgentHeader("Gatling")
        .acceptEncodingHeader("gzip")
httpConfig.basicAuth(user, password)
```

```
val timingAssertions: Iterable[Assertion] = Seq.apply(
    details("auth").responseTime.percentile4.lte(1000), // 99% below 1000ms
)
```

```
Simulation authservice.AuthServicePerfSimulation completed in 79 seconds
Parsing log file(s)...
Parsing log file(s) done
Generating reports...

================================================================================
---- Global Information --------------------------------------------------------
> request count                                       2085 (OK=2085   KO=0     )
> min response time                                    302 (OK=302    KO=-     )
> max response time                                   3181 (OK=3181   KO=-     )
> mean response time                                  1935 (OK=1935   KO=-     )
> std deviation                                        841 (OK=841    KO=-     )
> response time 50th percentile                       2116 (OK=2116   KO=-     )
> response time 75th percentile                       2640 (OK=2640   KO=-     )
> response time 95th percentile                       2930 (OK=2930   KO=-     )
> response time 99th percentile                       3022 (OK=3022   KO=-     )
> mean requests/sec                                 26.062 (OK=26.062 KO=-     )
---- Response Time Distribution ------------------------------------------------
> t < 800 ms                                           316 ( 15%)
> 800 ms < t < 1200 ms                                 191 (  9%)
> t > 1200 ms                                         1578 ( 76%)
> failed                                                 0 (  0%)
================================================================================

Reports generated in 0s.
Please open the following file: /Volumes/zDev/cy/talk/performance/gatling-demo/build/
Global: percentage of failed events is less than or equal to 0.05 : true
Global: 99th percentile of response time is less than or equal to 1200.0 : false

> Task :gatlingRun FAILED

FAILURE: Build failed with an exception.
```

Testcontainers

next level Integration testing

more realistic testing
based on "real" integration

orchestrate from JUnit

complex test-landscapes
(great for microservice testing)

Testcontainers

tests against third-party services

database operation against real
MySQL, Postgres, MSSQL, …

UI-testing with containerized browsers
(compatible to selenium)

https://www.testcontainers.org/

lots of ready-built containers for instant use
z.B. MySQL, MongoDB, Kafka, Elasticsearch, Neo4j, RabbitMQ

Databases ^

- Database containers
- JDBC support
- R2DBC support
- Cassandra Module
- CockroachDB Module
- Couchbase Module
- Clickhouse Module
- DB2 Module
- Dynalite Module
- InfluxDB Module
- MariaDB Module
- MongoDB Module
- MS SQL Server Module
- MySQL Module
- Neo4j Module
- Oracle-XE Module
- OrientDB Module
- Postgres Module
- Presto Module

- Docker Compose Module
- Elasticsearch container
- GCloud Module
- Kafka Containers
- Localstack Module
- Mockserver Module
- Nginx Module
- Apache Pulsar Module
- RabbitMQ Module
- Solr Container
- Toxiproxy Module
- Hashicorp Vault Module
- Webdriver Containers

eigene Container nutzbar

```
private static final GenericContainer authServiceContainer = new GenericContainer(new ImageFromDockerfile()
    .withFileFromClasspath( path: ".",  resourcePath: "/containers/authservice")
    .withFileFromPath( path: "authservice.jar", Paths.get(System.getProperty("authservice.jar"))))
    .withEnv("API_KEY", "lala")
    .withEnv("spring.profiles.active", "testcontainers")
    .withEnv("SSL_ENABLED", "false")
    .withEnv("DB_PROVIDER", "mysql")
```

```
new GenericContainer(DockerImageName.parse("jboss/wildfly:9.0.1.Final"))
```

usage

integrated docker client

ephemeral container-instances per test or "shared"

orchestrate containers manually during testing

```
container.start()
```

container classes are `AutoCloseable`

Wiremock

Http Simulator

match pre-defined responses
to given request

record / playback

JUnit or standalone

mock HTTP-Services

map responses on request match
e.g. `Authentication` - header

configure in Java or JSON

consumer testing for third-party services

test for certain error scenarios and problems

simulate latency

pre-define different stages in a scenario

admin-API for insights and control

provide sandbox of your service(s)
to your consumers


```
{
  "request": {
    "url": "/api/authenticate",
    "method": "POST",
    "bodyPatterns": [
      {
        "matchesJsonPath": "$[?(@.pass == 'hunter2')]"
      }
    ]
  },
  "response": {
    "status": 200,
    "body": "{ \"isAuthenticated\": \"true\" }",
    "fixedDelayMilliseconds": 350,
    "headers": {
      "Content-Type": "application/json;charset=UTF-8",
      "Cache-Control": "no-cache, no-store, max-age=0, must-revalidate",
      "Pragma": "no-cache",
      "Expires": "0",
      "Date": "{{now timezone='GMT' format='EEE, d MMM yyyy HH:mm:ss z'}}"
    }
  },
  "metadata": {
    "mapping-name": "authenticate/successfully_authenticated.json",
    "description": "Nutzer mit richtigem Passwort logt sich ein."
  }
}
```

dependency-check

secure software supply chain

dm TECH

DevSecOps*

find known vulnerabilities

secure your software

(very little security knowledge needed)

DEPENDENCY-CHECK

dm TECH

OWASP - Project

analyse third-party dependencies in your software
find known vulnerabilites

DEPENDENCY-CHECK

```
[INFO] Analysis Started
[INFO] Finished Archive Analyzer (0 seconds)
[INFO] Finished File Name Analyzer (0 seconds)
[INFO] Finished Jar Analyzer (0 seconds)
[INFO] Finished Dependency Merging Analyzer (0 seconds)
[INFO] Finished Version Filter Analyzer (0 seconds)
[INFO] Finished Hint Analyzer (0 seconds)
[INFO] Created CPE Index (2 seconds)
[INFO] Finished CPE Analyzer (3 seconds)
[INFO] Finished False Positive Analyzer (0 seconds)
[INFO] Finished NVD CVE Analyzer (0 seconds)
[INFO] Finished Sonatype OSS Index Analyzer (0 seconds)
[INFO] Finished Vulnerability Suppression Analyzer (0 seconds)
[INFO] Finished Dependency Bundling Analyzer (0 seconds)
[INFO] Analysis Complete (5 seconds)
[WARNING]

One or more dependencies were identified with known vulnerabilities in hello:

jackson-databind-2.9.6.jar (pkg:maven/com.fasterxml.jackson.core/jackson-databind@2.9.6, cpe:2.3:a:fasterxml:jackson:2.9.6:*:*:*:*:*:*:*, cpe:2.3:a:fasterxml:jack
18-14721, CVE-2018-19360, CVE-2018-19361, CVE-2018-19362, CVE-2019-12086, CVE-2019-12384, CVE-2019-12814, CVE-2019-14379, CVE-2019-14439, CVE-2019-14540, CVE-2019
2019-20330, CVE-2020-10672, CVE-2020-10673, CVE-2020-10968, CVE-2020-10969, CVE-2020-11111, CVE-2020-11112, CVE-2020-11113, CVE-2020-11619, CVE-2020-11620, CVE-20
-2020-9546, CVE-2020-9547, CVE-2020-9548
log4j-api-2.10.0.jar (pkg:maven/org.apache.logging.log4j/log4j-api@2.10.0, cpe:2.3:a:apache:log4j:2.10.0:*:*:*:*:*:*:*) : CVE-2020-9488
snakeyaml-1.19.jar (pkg:maven/org.yaml/snakeyaml@1.19, cpe:2.3:a:snakeyaml_project:snakeyaml:1.19:*:*:*:*:*:*:*) : CVE-2017-18640
spring-core-5.0.8.RELEASE.jar (pkg:maven/org.springframework/spring-core@5.0.8.RELEASE, cpe:2.3:a:pivotal_software:spring_framework:5.0.8:release:*:*:*:*:*:*, cpe
rk:5.0.8:release:*:*:*:*:*:*) : CVE-2018-15756, CVE-2020-5398, CVE-2020-5421
spring-security-core-5.0.7.RELEASE.jar (pkg:maven/org.springframework.security/spring-security-core@5.0.7.RELEASE, cpe:2.3:a:pivotal_software:spring_security:5.0.
tomcat-embed-core-8.5.32.jar (pkg:maven/org.apache.tomcat.embed/tomcat-embed-core@8.5.32, cpe:2.3:a:apache:tomcat:8.5.32:*:*:*:*:*:*:*, cpe:2.3:a:apache_software_
18-11784, CVE-2019-0199, CVE-2019-0221, CVE-2019-0232, CVE-2019-10072, CVE-2019-12418, CVE-2019-17563, CVE-2020-11996, CVE-2020-13934, CVE-2020-13935, CVE-2020-13


See the dependency-check report for more details.


[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 10.747 s
[INFO] Finished at: 2020-11-22T01:56:12+01:00
[INFO] ------------------------------------------------------------------------
```

Published Vulnerabilities

CVE-2018-1000873 suppress

Fasterxml Jackson version Before 2.9.8 contains a CWE-20: Improper Input Validation vulnerability in Jackson-Modules-Java8 that can result in Causes a denial-of-service (DoS). This attack appear to be exploitable via The victim deserializes malicio vulnerability appears to have been fixed in 2.9.8.

CWE-20 Improper Input Validation

CVSSv2:
- Base Score: MEDIUM (4.3)
- Vector: /AV:N/AC:M/Au:N/C:N/I:N/A:P

CVSSv3:
- Base Score: MEDIUM (6.5)
- Vector: /AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:N/A:H

References:
- CONFIRM - https://bugzilla.redhat.com/show_bug.cgi?id=1665601
- CONFIRM - https://security.netapp.com/advisory/ntap-20200904-0004/
- MISC - https://github.com/FasterXML/jackson-modules-java8/issues/90
- MISC - https://github.com/FasterXML/jackson-modules-java8/pull/87
- MISC - https://www.oracle.com/security-alerts/cpuoct2020.html
- MISC - https://www.oracle.com/technetwork/security-advisory/cpujul2019-5072835.html
- MISC - https://www.oracle.com/technetwork/security-advisory/cpuoct2019-5072832.html
- MLIST - [drill-dev] 20191017 Dependencies used by Drill contain known vulnerabilities
- MLIST - [drill-dev] 20191021 [jira] [Created] (DRILL-7416) Updates required to dependencies to resolve potential security vulnerabilities
- MLIST - [drill-issues] 20191021 [jira] [Created] (DRILL-7416) Updates required to dependencies to resolve potential security vulnerabilities
- MLIST - [nifi-commits] 20191113 svn commit: r1869773 - /nifi/site/trunk/security.html
- MLIST - [nifi-commits] 20200123 svn commit: r1873083 - /nifi/site/trunk/security.html
- MLIST - [pulsar-commits] 20190416 [GitHub] [pulsar] one70six opened a new issue #4057: Security Vulnerabilities - Black Duck Scan - Pulsar v.2.3.1
- N/A - N/A
- OSSINDEX - [CVE-2018-1000873] Improper Input Validation

Vulnerable Software & Versions: (show all)

- cpe:2.3:a:fasterxml:jackson-databind:*:*:*:*:*:*:*:* versions up to (excluding) 2.9.8
- ...

CVE-2018-14718 suppress

FasterXML jackson-databind 2.x before 2.9.7 might allow remote attackers to execute arbitrary code by leveraging failure to block the slf4j-ext class from polymorphic deserialization.

CWE-502 Deserialization of Untrusted Data

CVSSv2:
- Base Score: HIGH (7.5)
- Vector: /AV:N/AC:L/Au:N/C:P/I:P/A:P

CVSSv3:
- Base Score: CRITICAL (9.8)
- Vector: /AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

References:
- BID - 106601
- BUGTRAQ - 20190527 [SECURITY] [DSA 4452-1] jackson-databind security update
- CONFIRM - https://github.com/FasterXML/jackson-databind/commit/87d29af25e82a249ea15858e2d4ecbf64091db44
- CONFIRM - https://github.com/FasterXML/jackson-databind/issues/2097
- CONFIRM - https://github.com/FasterXML/jackson/wiki/Jackson-Release-2.9.7
- CONFIRM - https://security.netapp.com/advisory/ntap-20190530-0003/
- CONFIRM - https://www.oracle.com/technetwork/security-advisory/cpujan2019-5072801.html
- DEBIAN - DSA-4452
- MISC - https://www.oracle.com/security-alerts/cpujan2020.html

SonarQube

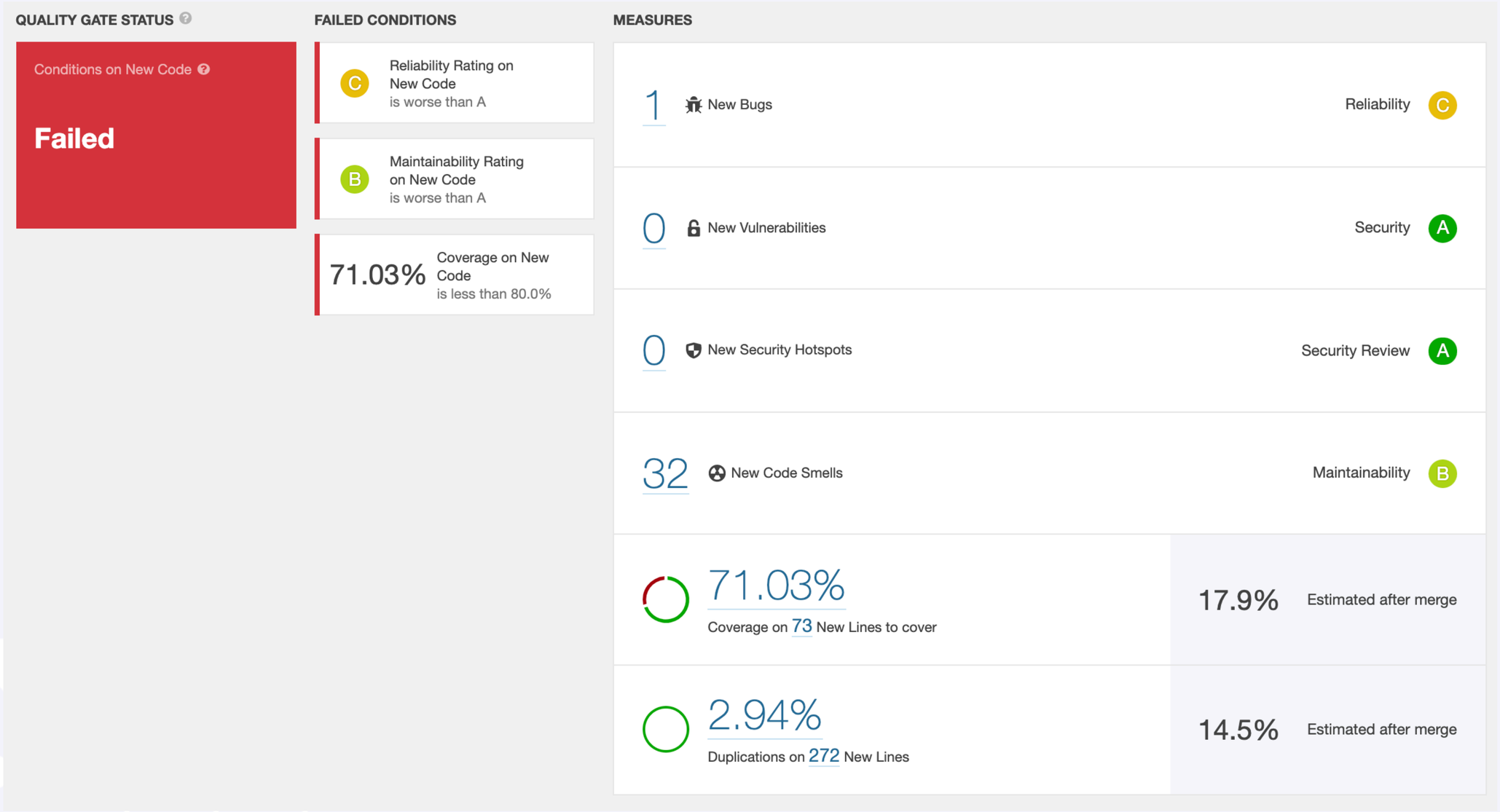
static code-Analysis

scan for technical bugs
and implementation issues

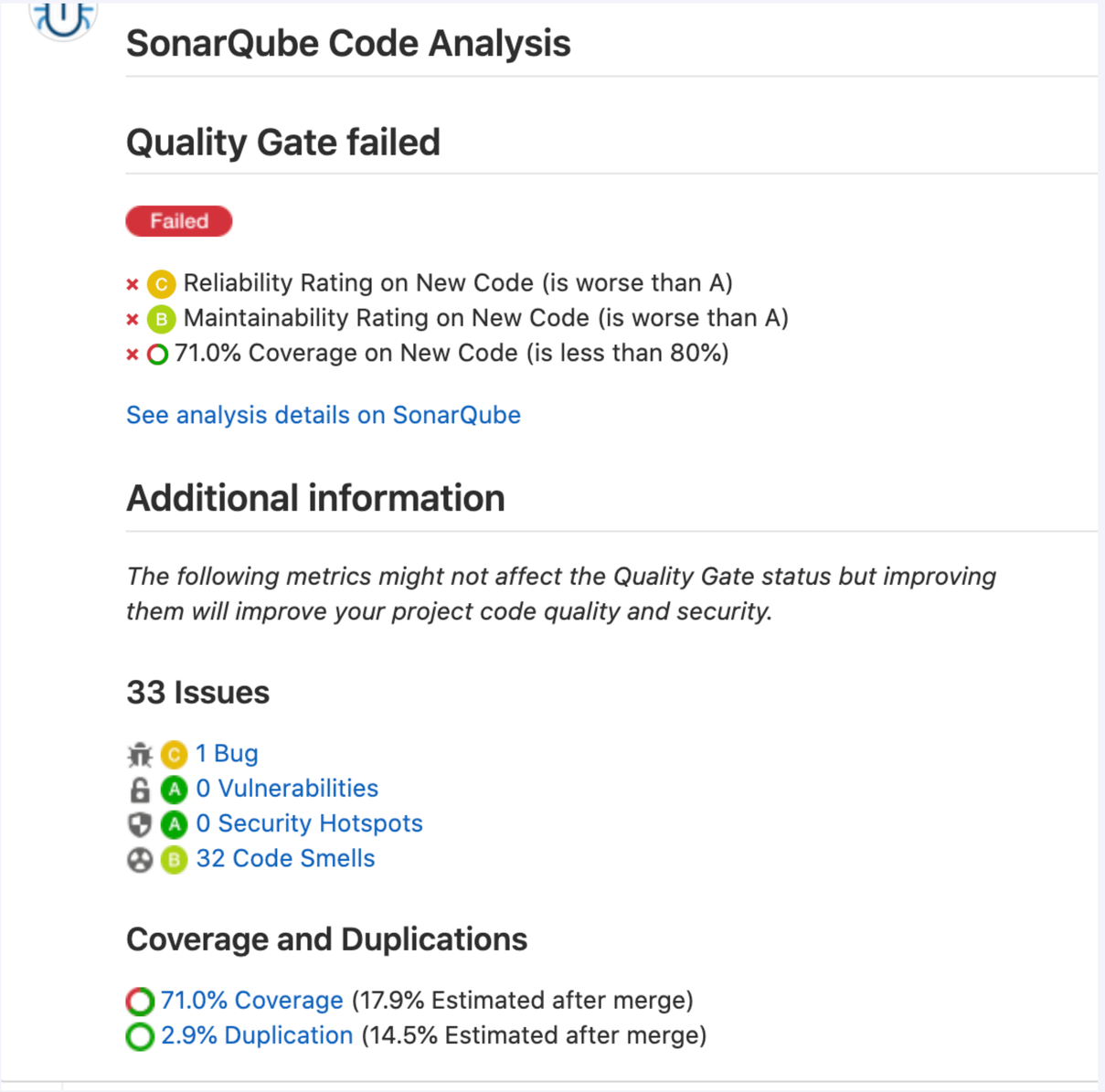
automated code review

Quality Gates in CI/CD

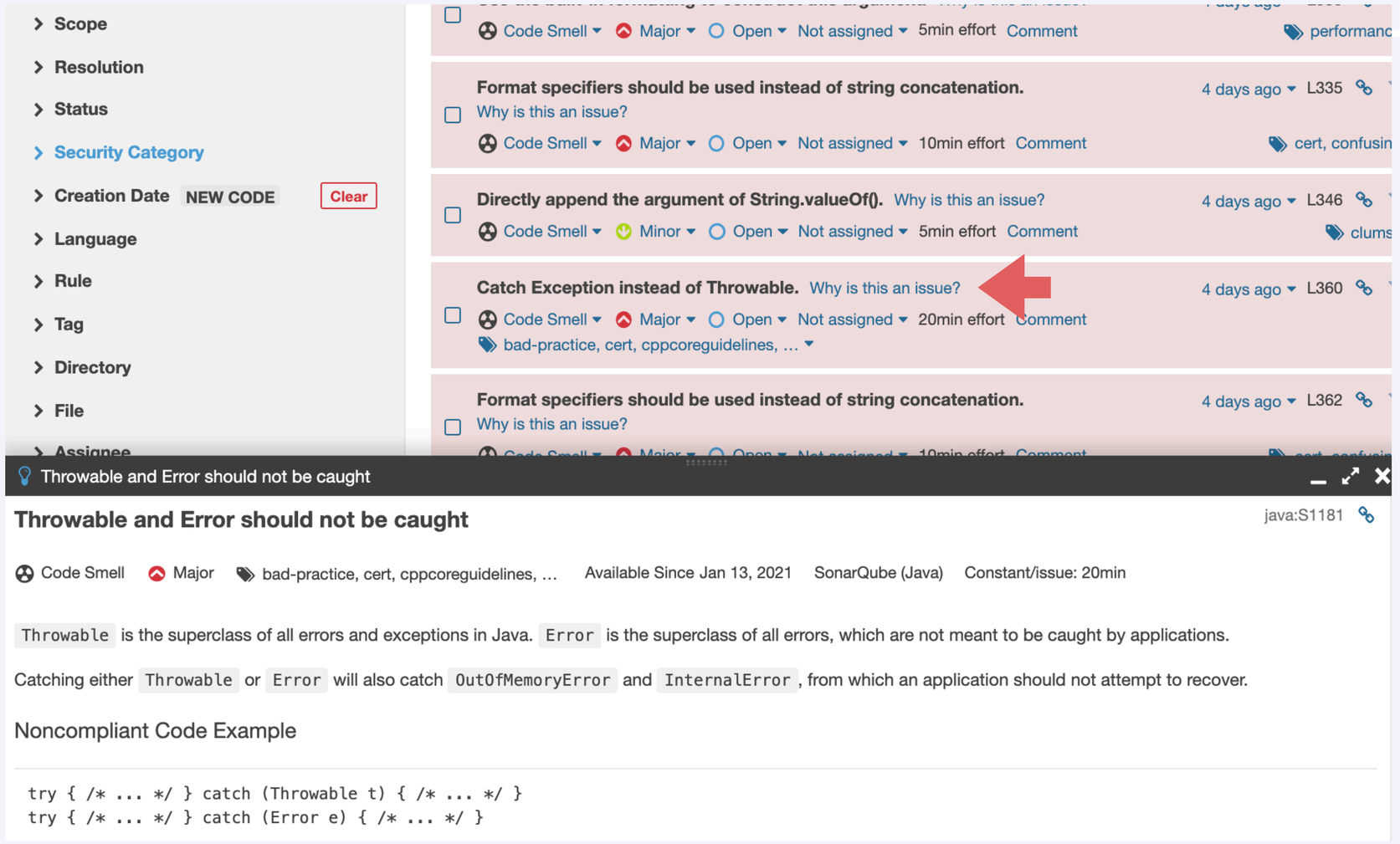
pull-request analysis



comment as
merge-request decorator



explanation and examples



there is MORE!

infrastructure testing

monitoring

"FinOps"

security

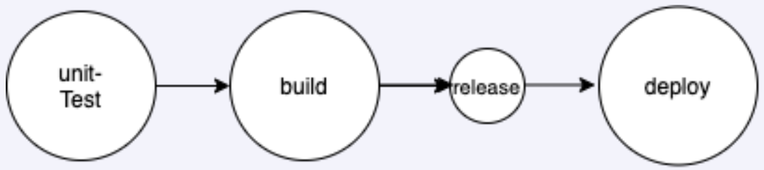
UX/UI

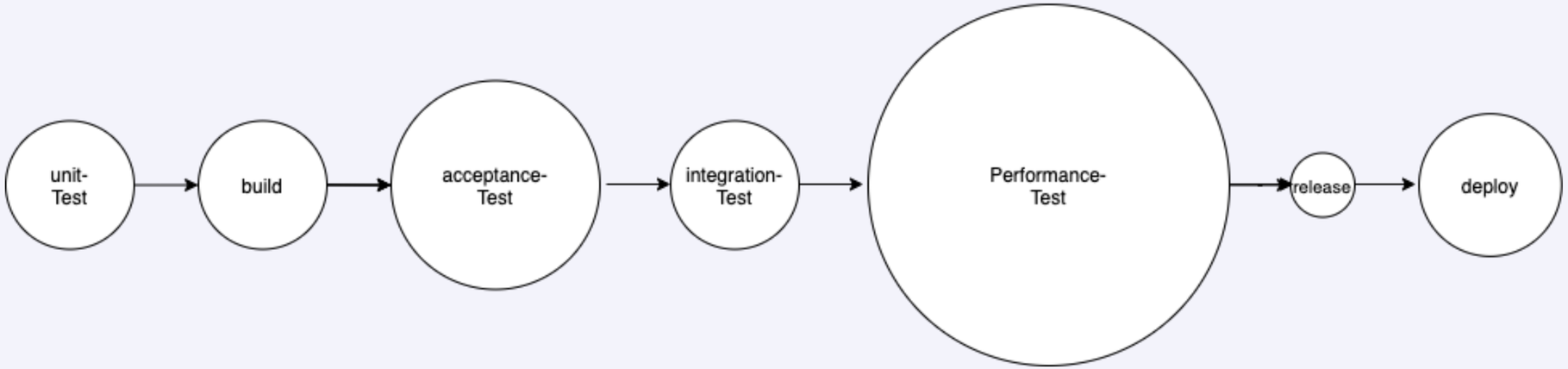
compliance

architecture

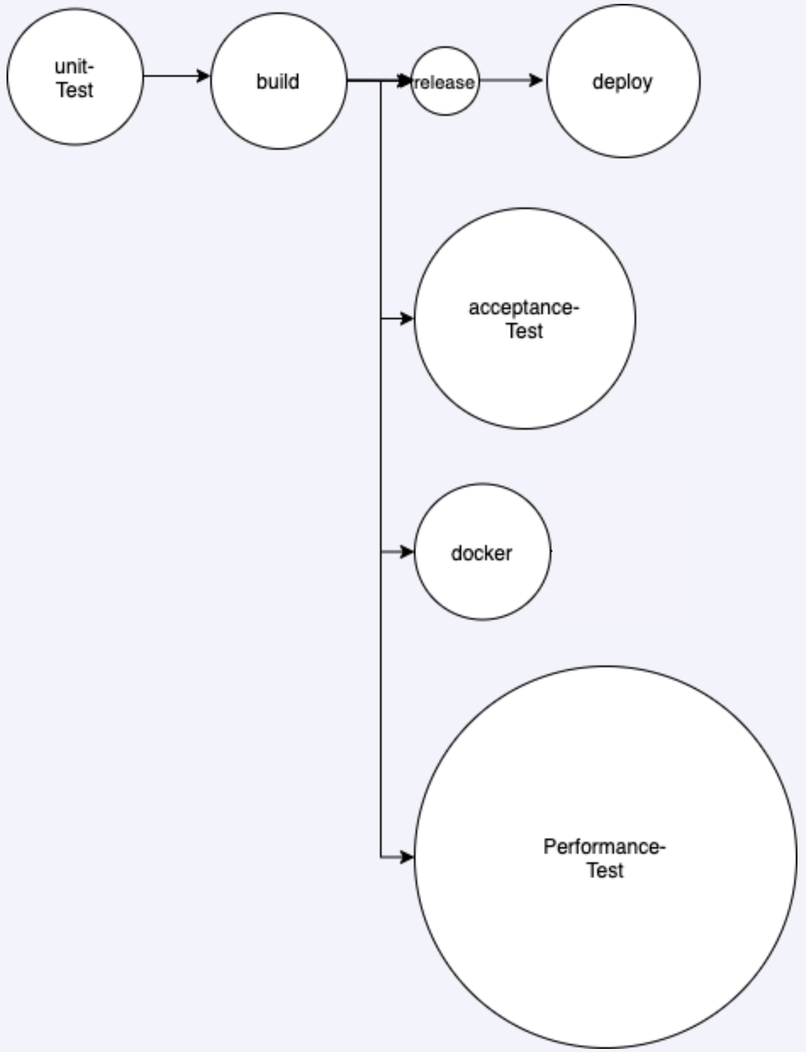
considerations

build-duration





run tests parallel or asynchronously



PROs of continuous testing

tests can not be forgotten / missed

test playbook and parameters are always the same

test runtime does not cost developer time

downsides to continuous testing

¯_(ツ)_/¯

downsides to continuous testing

test setup costs developer time

cost for infrastructure, build systems and skill

Thank you for your interest!

Questions?

join Q&A Lounge heise Developer

happy if you send feedback chris@clowncomputing.de @CYxChris