# SOFTWARE TESTING

# AGENDA

1. About Akamas & Moviri Group
2. Why Test
3. What to Test

4. Functional Testing
5. Performance Testing
6. Continuous Integration (CI)

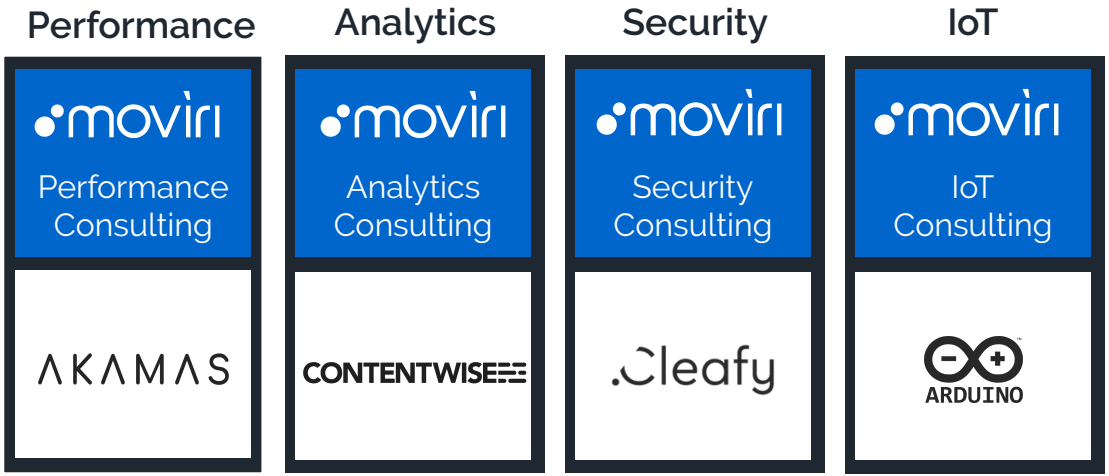
**Giovanni Paolo Gibilisco**
Head of Engineering
**Akamas**

# ABOUT AKAMAS & MOVIRI GROUP

# ONE GROUP, MANY BUSINESSES

## An Agile And Innovative Organization

We are a university start-up that has grown into an international group of five companies. We run specialized consulting services under the Moviri brand in four strategic fields: Performance, Analytics, Security and Internet of Things. Around these areas, we've made investments in four technology companies that fit a cohesive vision: Akamas, ContentWise, Cleafy, and Arduino.



ABOUT US

# WHO WE ARE

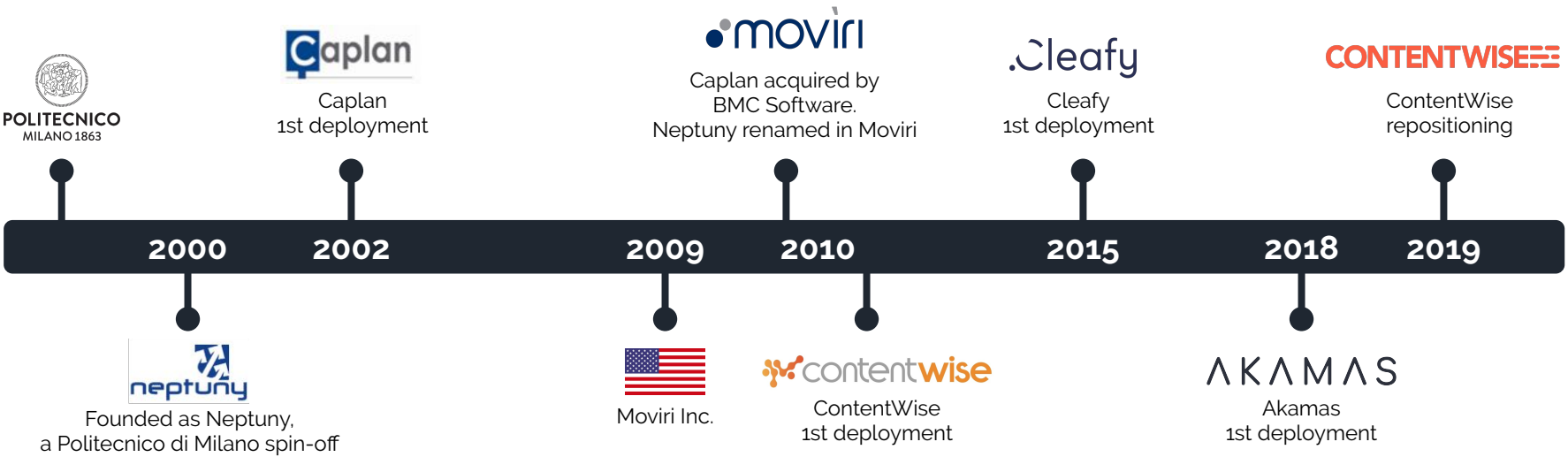
## A Successful Group

We are a global consulting and technology services company with 19 years of expertise in several IT fields, including Performance Engineering, Analytics & Big Data, IT Security and IoT Management Systems.

Our approach is simple yet our goals are ambitious: identify critical problems in selected IT fields and design and implement innovative solutions that are easy to adopt and work at an enterprise scale.

# MOVIRI TIMELINE

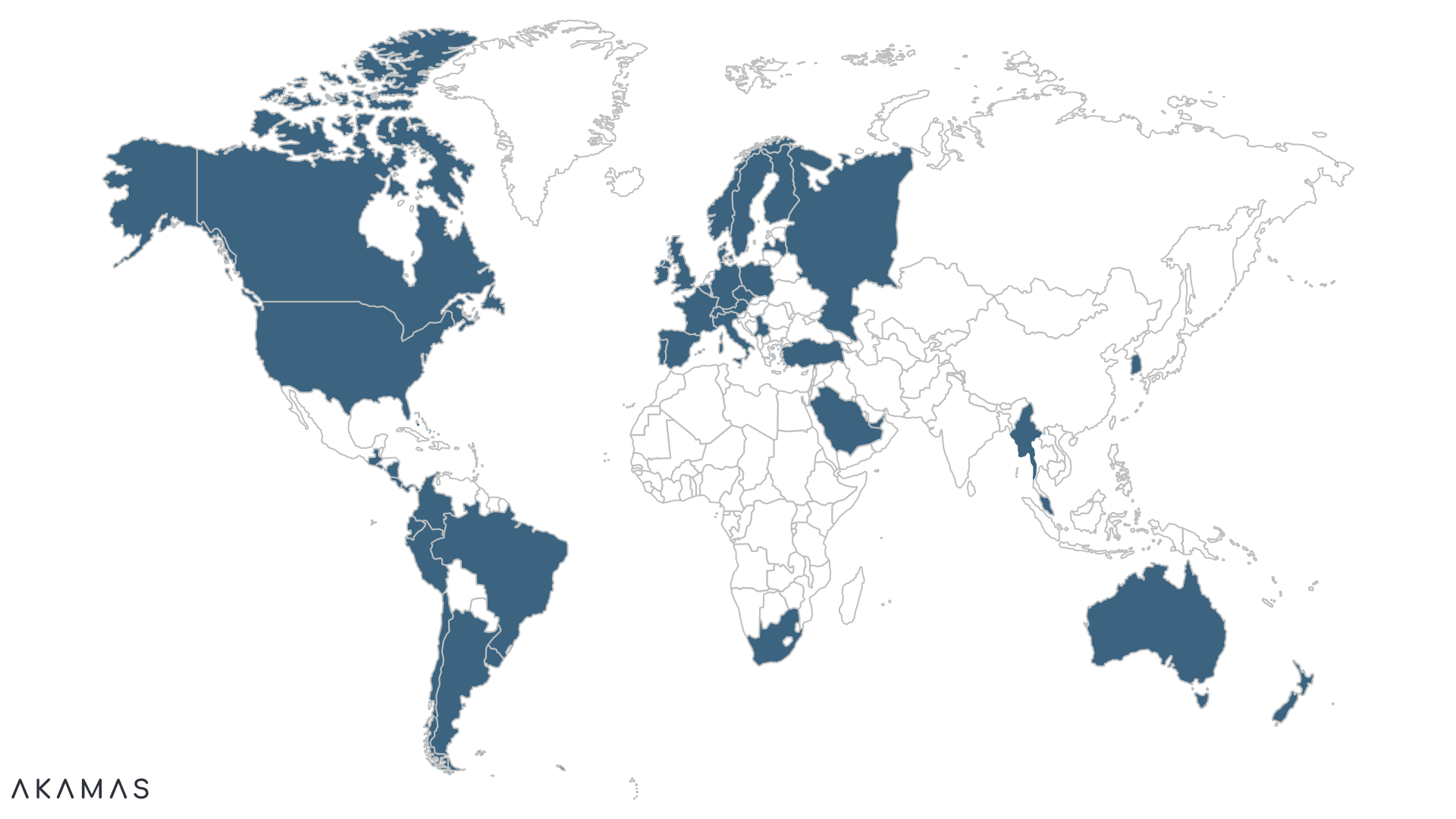
## From Start-Up To International Group



# MAIN ACHIEVEMENTS

## Great Ideas And Big Numbers

We achieved great results so far and we have great ambitions for achieving more in the future. This is why we always strive to deliver the best and love moving out of our comfort zone to make sure that we exceed our customers' expectations.



AKAMAS

# MAIN CUSTOMERS

## Banking, Insurance & Finance


## Tech, Telco & Media


## Manufacturing & Retail


## Utilities & Energy


AKAMAS

PERFORMANCE

## Moviri Performance Engineering Consulting

How to reduce the risks introduced by changes and test applications to ensure safe deployments?
It was the first of our big ideas, it is where we started.

### Our Services

- Performance Testing and Tuning
- Digital Monitoring
- Capacity and Cost Optimization
- AIOps for IT operational analytics

AKAMAS

AKAMAS

The ML-driven solution that optimizes performance and cost by tuning IT stack configurations.

ANALYTICS

## Moviri Analytics Consulting

We are the Data Geeks that help customers in extracting values from data.

### Our Services

- Big Data solutions & Data Management
- Machine Learning & Artificial Intelligence
- Real-time analytics & Dynamic Reporting

AKAMAS

## CONTENTWISE

Personalization is no longer a nice-to-have.

Our AI automatically manages digital storefronts and enhances catalog metadata to create superior digital experiences for end-customers.

SECURITY

## Moviri Security Consulting

We help clients by actively developing in-depth security competencies and proprietary assets and leveraging best-of-breed technologies.

### Our Services

- Fraud and Risk Intelligence
- Security Intelligence and Response Automation
- User Access and Behavior Analysis
- Infrastructure and Data Protection
- IT Security Governance

AKAMAS

## Cleafy

Cleafy is the market-leader in threat detection and protection solution adopted by leading Banks, Payment Services, Online Lenders and Digital Commerce to protect their online services and end-users from today's advanced targeted attacks and frauds.

INTERNET OF THINGS

## Moviri IoT Consulting

Hardware design, firmware and software development, cloud and edge analytics mixed together to bring automation, live data, intelligence to the physical world.

### Our Services

- IoT cloud platform selection
- Solution design and implementation
- Analytics on Edge
- Sensorization
- Servitization

AKAMAS

ARDUINO

Arduino is an open-source electronics platform based on easy-to-use hardware and software.

Today Arduino is also an IoT product and services company offering industrial grade and certified connected board and a Cloud platform for the enterprise.

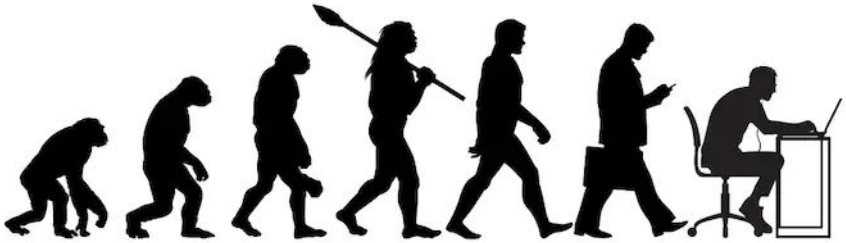
# WHY TESTING

Software changes

Requirements changes

"**Change** has been recognized as the distinguishing feature that makes software different from any other human-produced artifacts"

***C. Ghezzi***



STORY TIME!

*Once upon a time*,

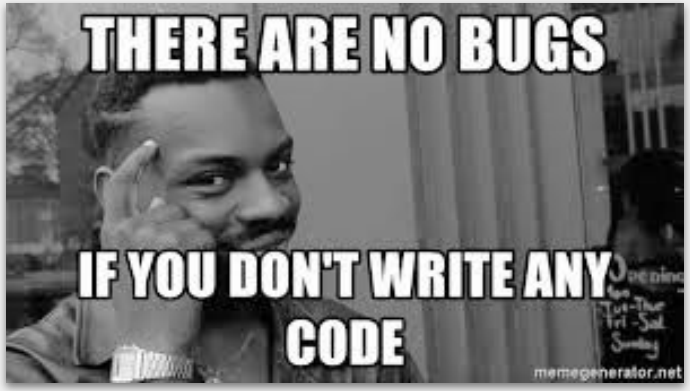
a customer asked us to build a software to provision Virtual Machines on Azure and Openstack

It worked on AWS and VMware

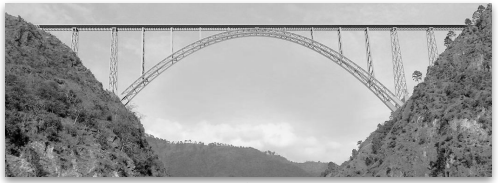
# WHAT TO TEST

# OUR CODE IS OUR RESPONSIBILITY

1. We should rely on external code/services
2. Test our use of external services
3. Other people should rely on our services



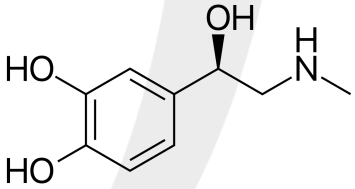
WHY TESTING

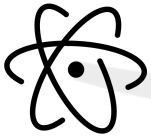


Internals

Connections

User functionality



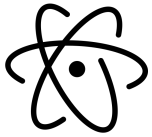
# FUNCTIONAL TESTING

**Internals**

Connections

User functionality

**Unit tests** should be **short**, **focused** and **quick** to run. Developers need feedback in seconds not minutes.



# Internals Unit Testing

JUnit 5

AssertJ

Fluent assertions for java

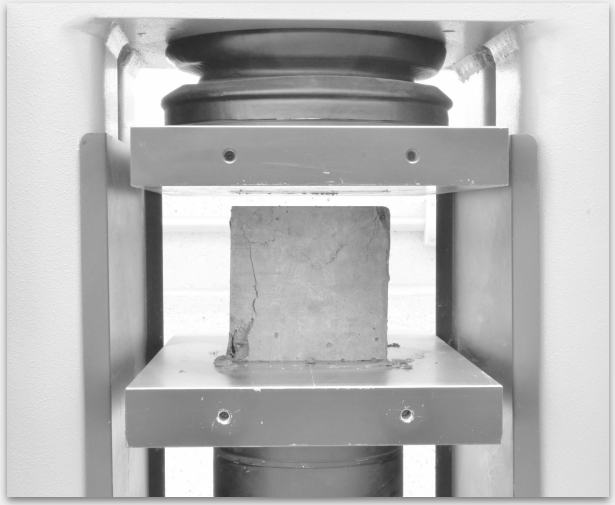
TestNG

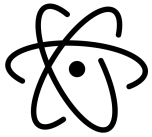
Many unit testing **tools** and **frameworks** they all are **integrated into IDE**s and depend on the **language**

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <configuration>
        <argLine>
            @{argLine} --illegal-access=permit
        </argLine>
        <useSystemClassLoader>false</useSystemClassLoader>
    </configuration>
</plugin>
```

# Unit Testing

Junit

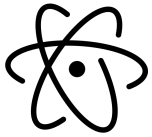




# Internals
# Unit Testing

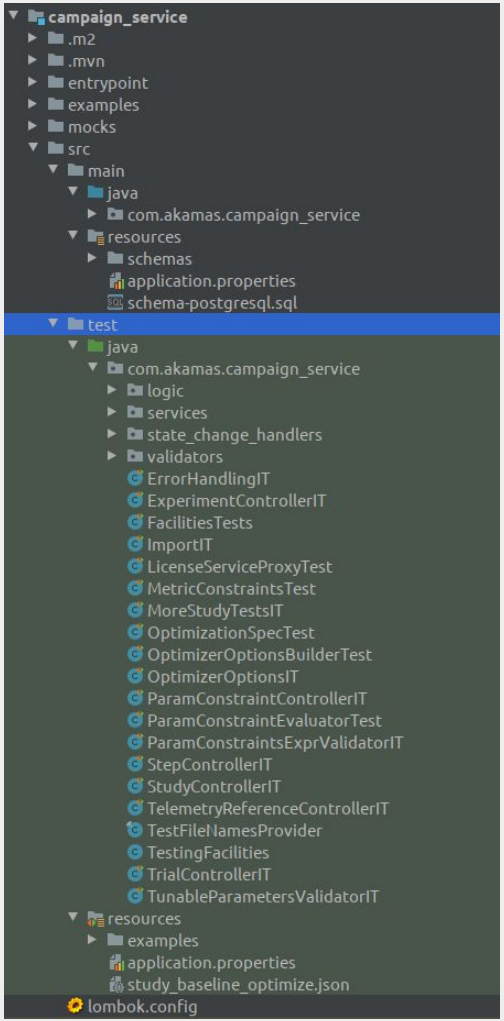
1. Test public functions
2. Test code is code
3. Easy to understand
4. Separate concerns

```java
@RunWith(SpringRunner.class)
public class MetricConstraintsTest {

    @Test
    public void testLower() {
        MetricConstraint constraint = new MetricConstraint();
        constraint.setMetric("metricName");
        constraint.setLowerThan(1.0);
        assertThat(constraint.isValid( value: 0.0)).isTrue();
        assertThat(constraint.isValid( value: 2.0)).isFalse();
    }

    @Test
    public void testGreater() {
        MetricConstraint constraint = new MetricConstraint();
        constraint.setMetric("metricName");
        constraint.setGreaterThan(1.0);
        assertThat(constraint.isValid( value: 2.0)).isTrue();
        assertThat(constraint.isValid( value: 0.0)).isFalse();
    }

    @Test
    public void testIsEqual() {
        MetricConstraint constraint = new MetricConstraint();
        constraint.setMetric("metricName");
        constraint.setEqualTo(1.0);
        assertThat(constraint.isValid( value: 1.0)).isTrue();
        assertThat(constraint.isValid( value: 0.0)).isFalse();
        assertThat(constraint.isValid( value: 2.0)).isFalse();
    }
    @Test
    public void testWithin() {
        MetricConstraint constraint = new MetricConstraint();
        constraint.setMetric("metricName");
        constraint.setLowerThan(2.0);
        constraint.setGreaterThan(1.0);
        assertThat(constraint.isValid( value: 1.5)).isTrue();
        assertThat(constraint.isValid( value: 0.0)).isFalse();
        assertThat(constraint.isValid( value: 3.0)).isFalse();
    }

}
```



# Internals
# Unit Testing

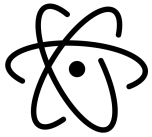
Test code lives with production code. Same version, same interfaces, same scope



# Writing a Test

1. Preparation
2. Code Execution
3. Assertions
4. Tear Down

```
@MockBean
LicenseDao licenseDao;

private String encryptedVersion;
private String encryptedCustomer;
private String encryptedTerm;
private String encryptedCredits;
private UUID id;

@Before
@After
public void resetDb() {
    licenseDao.deleteAll();
    when(licenseDao.existsById(any(UUID.class))).thenReturn(false);
}

@Before
public void init() throws GeneralSecurityException {
    encryptedTerm = testingFacilities.encrypt("2987-11-06");
    encryptedCredits = testingFacilities.encrypt("10");
    encryptedCustomer = testingFacilities.encrypt("Akamas");
    encryptedVersion = testingFacilities.encrypt("1.0.0");
    id = UUID.randomUUID();
}

@Test
public void licenseSignatureTest() throws GeneralSecurityException {
    String hash = MessageFormat.format( pattern: "id:{0}:term:{1}:credits:{2}:customer:{3}:version:{4}",
            ...arguments: id,
            "2987-11-06",
            "10",
            "Akamas",
            "1.0.0");
    String signature = testingFacilities.encrypt(hash);

    License license = new License(id, encryptedTerm, encryptedCredits, encryptedCustomer, encryptedVersion);
    ValidationOutcome validation = licenseValidator.validateSignature(license, signature);
    assertThat(validation.isValid()).isTrue();
}

@Test
public void licenseInvalidSignatureTest() {
    License license = new License(id, encryptedTerm, encryptedCredits, encryptedCustomer, encryptedVersion);
    ValidationOutcome validation = licenseValidator.validateSignature(license, signature: "something_that_is_not_correctly_encrypted");
    assertThat(validation.isValid()).isFalse();
    assertThat(validation.getMessage()).isEqualTo("unable to decode base64 string: invalid characters encountered in base64 data");
}
```

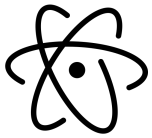


# Assertion

Assertions define the **test conditions**.
AssertJ provides a **fluent api** to specify them.

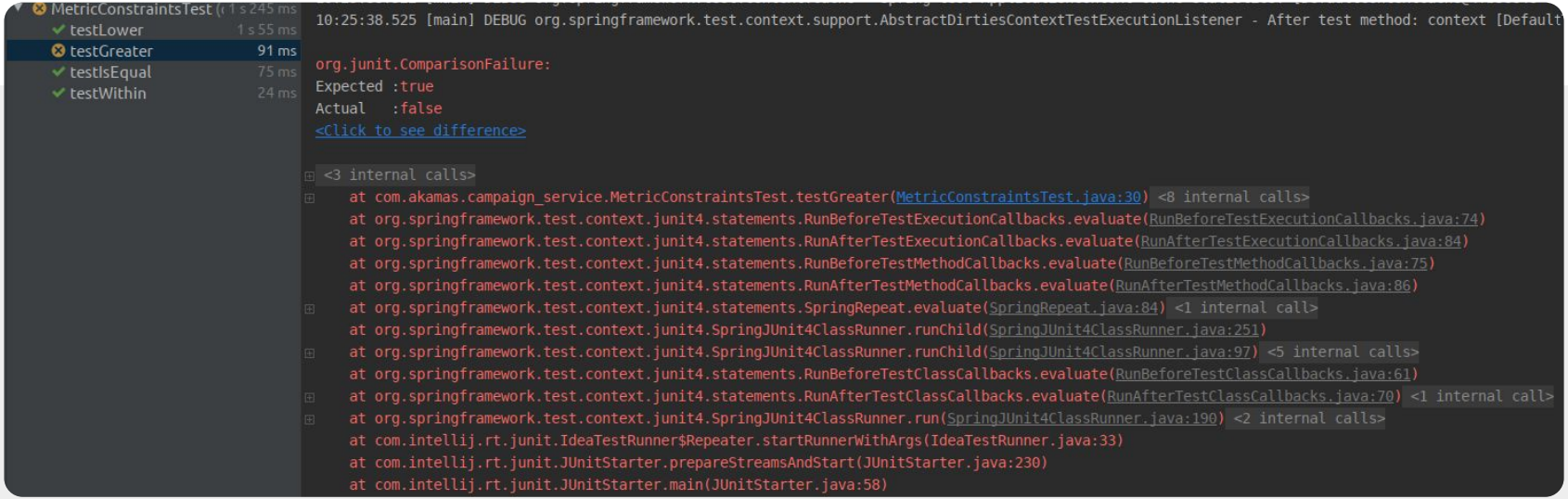
```
import static org.assertj.core.api.Assertions.in;
import static org.assertj.core.api.Assertions.not;
import static org.assertj.core.api.Assertions.notIn;
...

// filters use introspection to get property/field values
assertThat(fellowshipOfTheRing).filteredOn("race", HOBBIT)
                               .containsOnly(sam, frodo, pippin, merry);

// nested properties are supported
assertThat(fellowshipOfTheRing).filteredOn("race.name", "Man")
                               .containsOnly(aragorn, boromir);

// you can apply different comparison
assertThat(fellowshipOfTheRing).filteredOn("race", notIn(HOBBIT, MAN))
                               .containsOnly(gandalf, gimli, legolas);

assertThat(fellowshipOfTheRing).filteredOn("race", in(MAIA, MAN))
                               .containsOnly(gandalf, boromir, aragorn);

assertThat(fellowshipOfTheRing).filteredOn("race", not(HOBBIT))
                               .containsOnly(gandalf, boromir, aragorn, gimli, legolas);

// you can chain multiple filter criteria
assertThat(fellowshipOfTheRing).filteredOn("race", MAN)
                               .filteredOn("name", not("Boromir"))
                               .containsOnly(aragorn);
```

AssertJ



# IDE

Tests should be run often during development to anticipare bugs
IDE should report them clearly showing which test failed and the condition that led to the failure

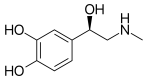


Internals

**Connections**

User functionality

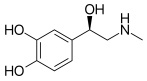
**Integration tests** target groups of modules and validates their interactions.




# Mocking Scaffolding

Mock mimic the behavior of real objects in controlled ways.

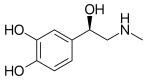
Like a crash test dummy mimic the behavior of a human in a car accident


# Connections Integration Testing

1. Test components
2. Interacts with real components or mocks
3. Interacts with external services

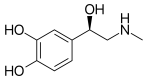
```
@RunWith(SpringRunner.class)
@SpringBootTest
@ActiveProfiles("test")
public class OptimizationServiceIT {

    @MockBean
    private OptimizationDAO optimizationDAO;

    @MockBean
    private OptimizationEngineProxy optimizationEngineProxy;

    @MockBean
    private ContainerBridgeDAO containerBridgeDAO;

    @Autowired
    private OptimizationService optimizationService;
```




# Connections
# Integration Testing

1. Mock specify the **expected** behavior of external **components**
2. Test should validate the **external behavior** of our component
3. Also in case of **internal errors** we should behave nicely

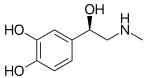
```
@Test
public void testReadOptimization() throws ApiException {
    when(optimizationDAO.findById(Mockito.eq( value: 1L))).thenReturn(Optional.empty());
    when(optimizationDAO.findById(AdditionalMatchers.not(eq( value: 1L)))).thenReturn(Optional.of(new Optimization()));
    try {
        optimizationService.readOptimization( id: 1L);
        Fail.fail( failureMessage: "An api exception should be thrown");
    } catch (ApiException e) {
        assertThat(e).extracting( ...propertiesOrFields: "httpStatus", "message").containsExactly(
                HttpStatus.NOT_FOUND,
                MessagesProvider.OPTIMIZATION_NOT_FOUND
        );
    }
    assertThat(optimizationService.readOptimization( id: 2L).getClass()).isEqualTo(Optimization.class);
}
```




# Mocking Scaffolding

Mockito acts at the **component level** (e.g. classes) and can be used to mock internal components.


```java
@RunWith(SpringRunner.class)
@SpringBootTest
@ActiveProfiles("test")
public class OptimizationServiceIT {

    @MockBean
    private OptimizationDAO optimizationDAO;

    @MockBean
    private OptimizationEngineProxy optimizationEngineProxy;

    @MockBean
    private ContainerBridgeDAO containerBridgeDAO;

    @Autowired
    private OptimizationService optimizationService;
```




# Mocking Scaffolding

Wiremock acts at the **API level** and can be used to mock **external services** or internal services in a microservice architecture.

WIREMOCK

```
{
  "request": {
    "method": "POST",
    "url": "/score_computation_requests",
    "headers": {
      "X-User-Id": {
        "matches": "[0-9A-Fa-f]{8}-[0-9A-Fa-f]{4}-[1-5][0-9A-Fa-f]{3}-[89ABab][0-9A-Fa-f]{3}-[0-9A-Fa-f]{12}"
      }
    },
    "bodyPatterns": [
      {"matchesJsonPath": "$.windowPolicyName"},
      {"matchesJsonPath": "$.windowPolicyLabelSelectors"},
      {"matchesJsonPath": "$.windowPolicyArgs"},
      {"matchesJsonPath": "$.metricsToAggregate"},
      {"matchesJsonPath": "$.scoreSpec"},
      {"matchesJsonPath": "$.trialStart"},
      {"matchesJsonPath": "$.trialEnd"},
      {"matchesJsonPath": "$.workflowRunId"},
      {"matchesJsonPath": "$.workflowId"}
    ]
  },
  "response": {
    "status": 200,
    "jsonBody": {
      "content": {
        "score": 11.0,
        "window": {
          "from": "2018-06-30T21:00:00.000",
          "to": "2018-06-30T21:01:00.000"
        },
        "aggregatedMetrics": {
          "comp1.metric1": {
            "avg": 30.31,
            "variance": 0.4
          }
        }
      }
    },
    "headers": {
      "Content-Type": "application/json"
    }
  }
}
```

AKAMAS

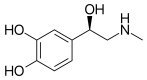
FUNCTIONAL TESTING

Internals

Connections

**User functionality**

**End to End (E2E) tests** validates the behavior of the entire application from a user perspective.

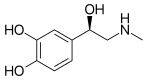



# User functionality E2E Testing

E2E is the only kind of testing that **relates to the end user** and is probably the most important one.

It is also the most **expensive complex** and **slow**.

Usually done manually should definitely be automated

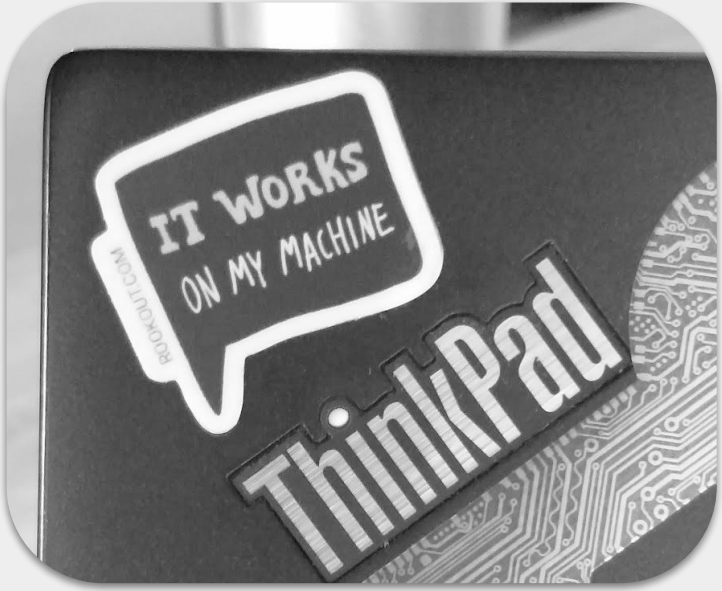



# User functionality E2E Testing

Runs in a different Environment than developer laptop.

Model user behavior and requirements (coming from RASD)

Key in finding functional regressions before releasing



# PERFORMANCE TESTING

PERFORMANCE TESTING

# Why

User Behavior

Scenarios

**Performance Tests** measure **non functional** properties of the application related to **speed**, **supported load** and **efficiency**

# Load Testing Why

Functional tests verify **what** the application do.

Performance tests verify **how** it does so.

Related to the **user experience**.

# Why **User Behavior** Scenarios

**Performance Tests** simulate the interaction of **real users** with the system.

# Load Testing User Behavior

Users behave differently from what expected.

User behave differently one another

Users have different quality requirements


AKAMAS

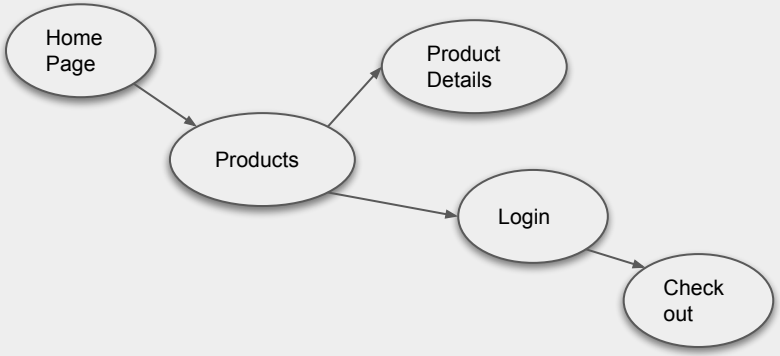
# Load Testing User Behavior

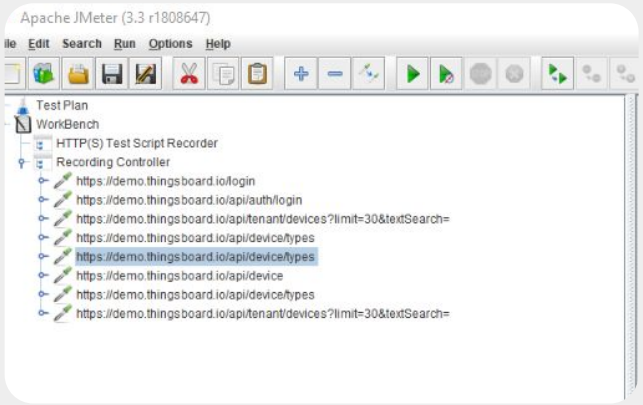
Analyze production metrics & logs

Model user behavior

Build a navigation representing users

Add think time




AKAMAS

Why

User Behavior

**Scenarios**

**Performance Tests** are used in a variety of scenarios to assess **peak performance** level, **critical conditions** or **predict failures**

# Load Testing Scenarios

How much concurrent users can my system support?

What is the main bottleneck?

Are there any memory leak?


AKAMAS

# CONTINUOUS INTEGRATION

**Why**

How

Test are valuable and we should use them. Tests are complex and hard to run

Why

**How**

Continuous Integration (CI) platforms and automated testing tools reduce the burden of manual testing

# Continuous Integration

A platform to automate the execution and reporting of different tests.

Lives with the code.

Easy to assess which commit broke a certain functionality.

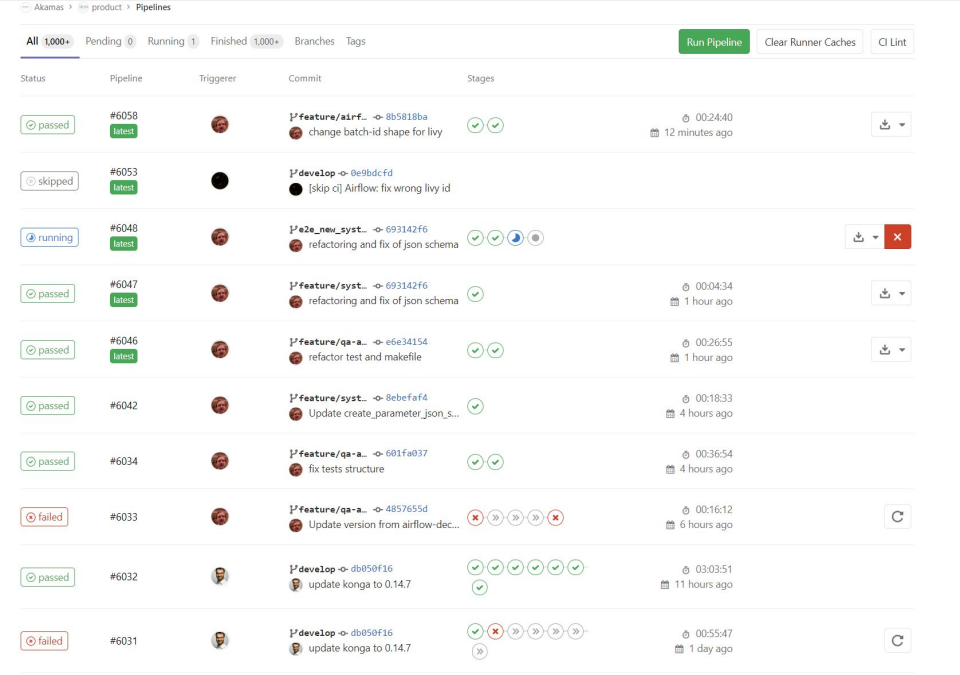
Travis CI

# Continuous Integration

Allows to select which tests should be run according to the different branches of a repository.

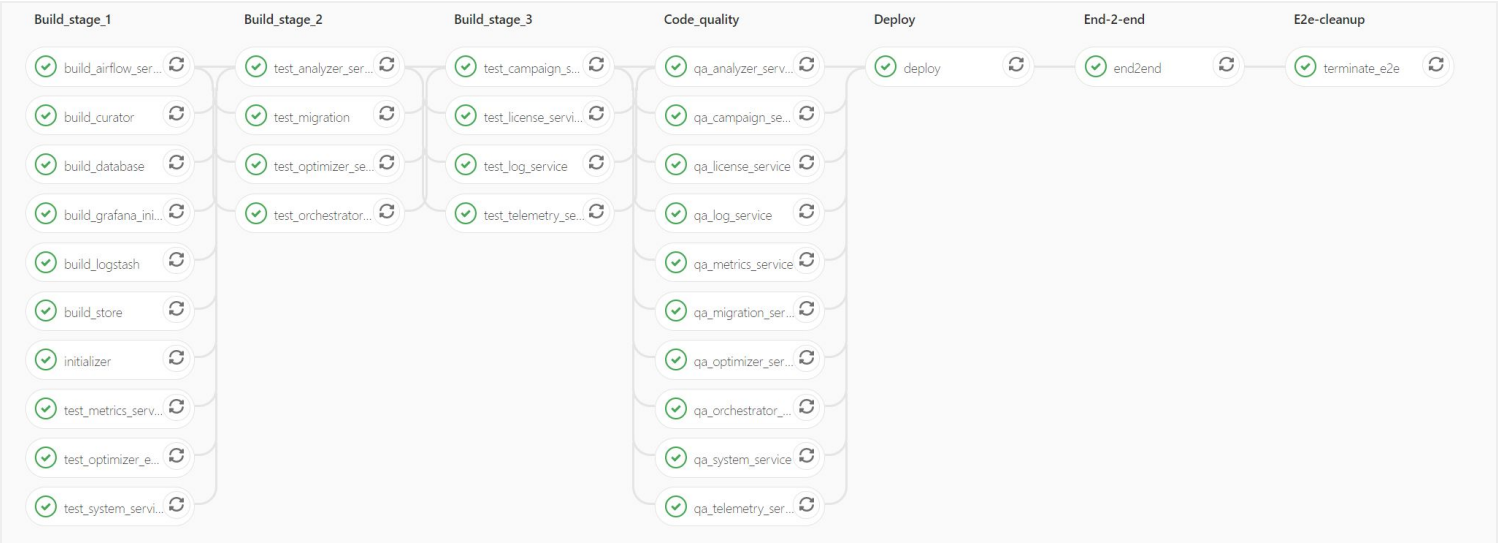
Report test results for all commits.

Nightly run longer tests



# Continuous Integration

A testing pipeline is composed by multiple steps.
Each module has its own testing strategy.
Code quality perform static analysis
Finally The application is deployed and tested End to End




AKAMAS

# ANY QUESTIONS?

moviri


Italy HQ
Via Schiaffino, 11
20158 Milan
T: +39 02 4951 7001

Boston
211 Congress Street
Boston, MA 02110
T: +1 617 936 0212

Los Angeles
12655 W. Jefferson Blvd
Los Angeles, CA 90066
T: +1 323 524 0524

Singapore
5 Temasek Boulevard,
Singapore 038985

@moviri
@contentwisetv
@AkamasLabs
@Cleafy

@Moviri
@ContentWise
@Akamas
@Cleafy

@movirigroup
@contentwise.tv
@akamaslabs
@cleafy

@movirigroup


# CONTACTS