



DEVOPS

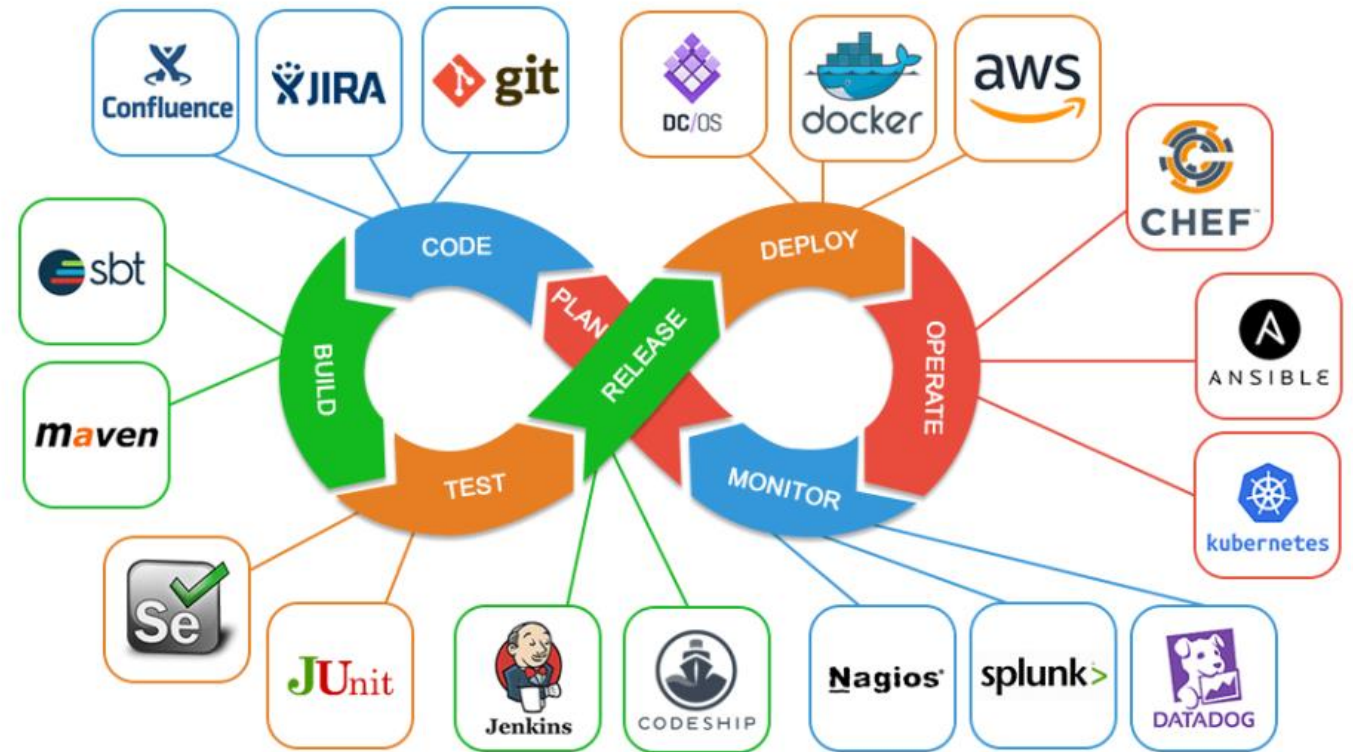
JORGE LOPEZ
BLASCO

INTRODUCCIÓN A TESTING

- Introducción
- Conceptos
- Objetivos
- Principios
- Herramientas
- Versionado
- Calidad del Software



INTRODUCCIÓN

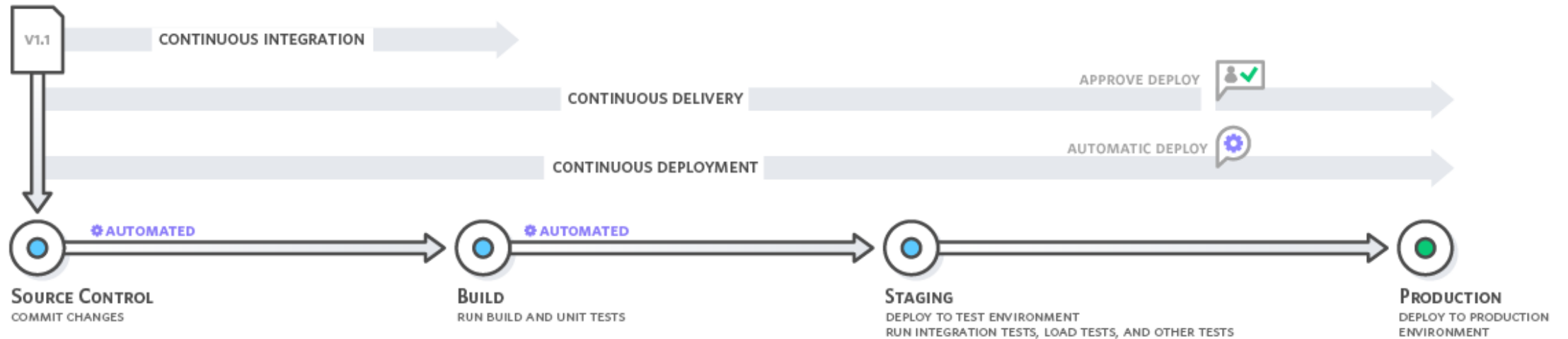


VENTAJAS DEVOPS

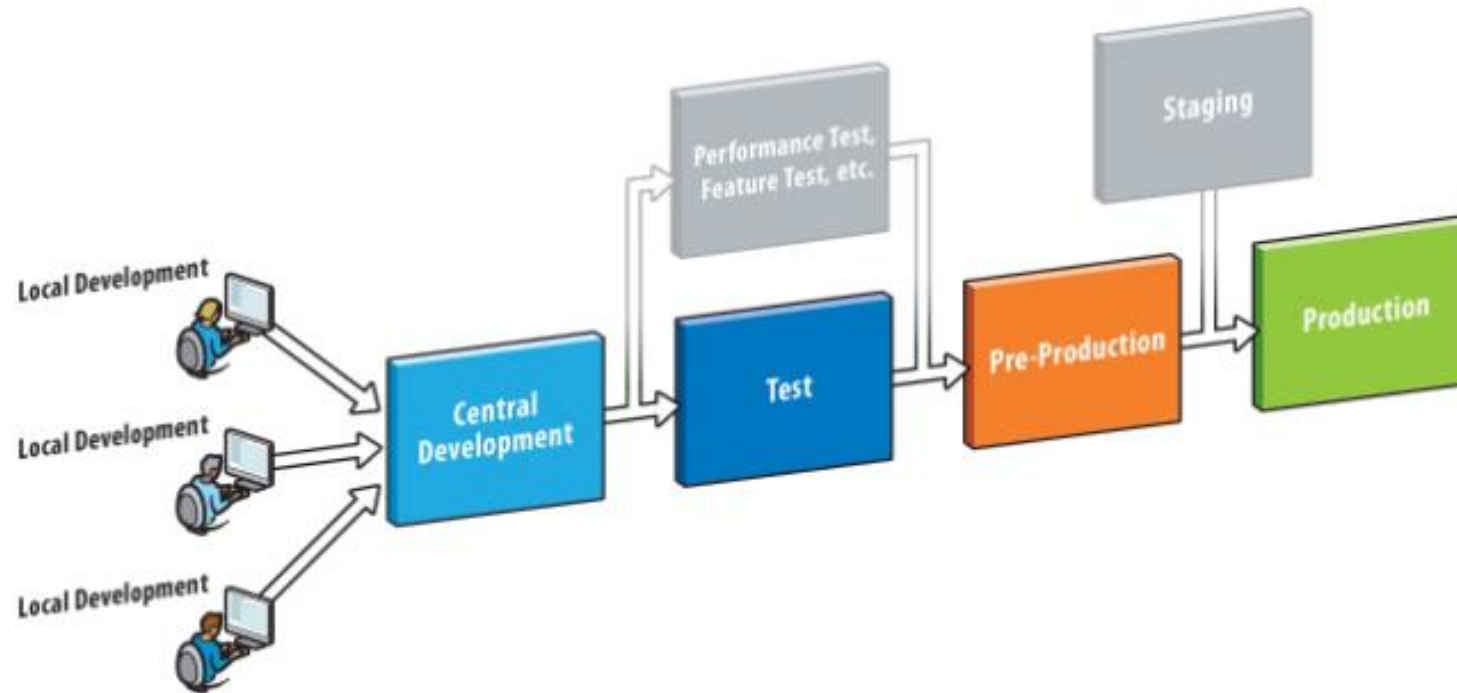
- Una mejor y más rápida entrega de productos
- Resolución de problemas en menos tiempo y con menor complejidad
- Mejor escalabilidad y disponibilidad
- Entornos de funcionamiento más estables
- Mejor utilización de los recursos
- Mayor automatización
- Mayor visibilidad de resultados del sistema
- Mayor innovación



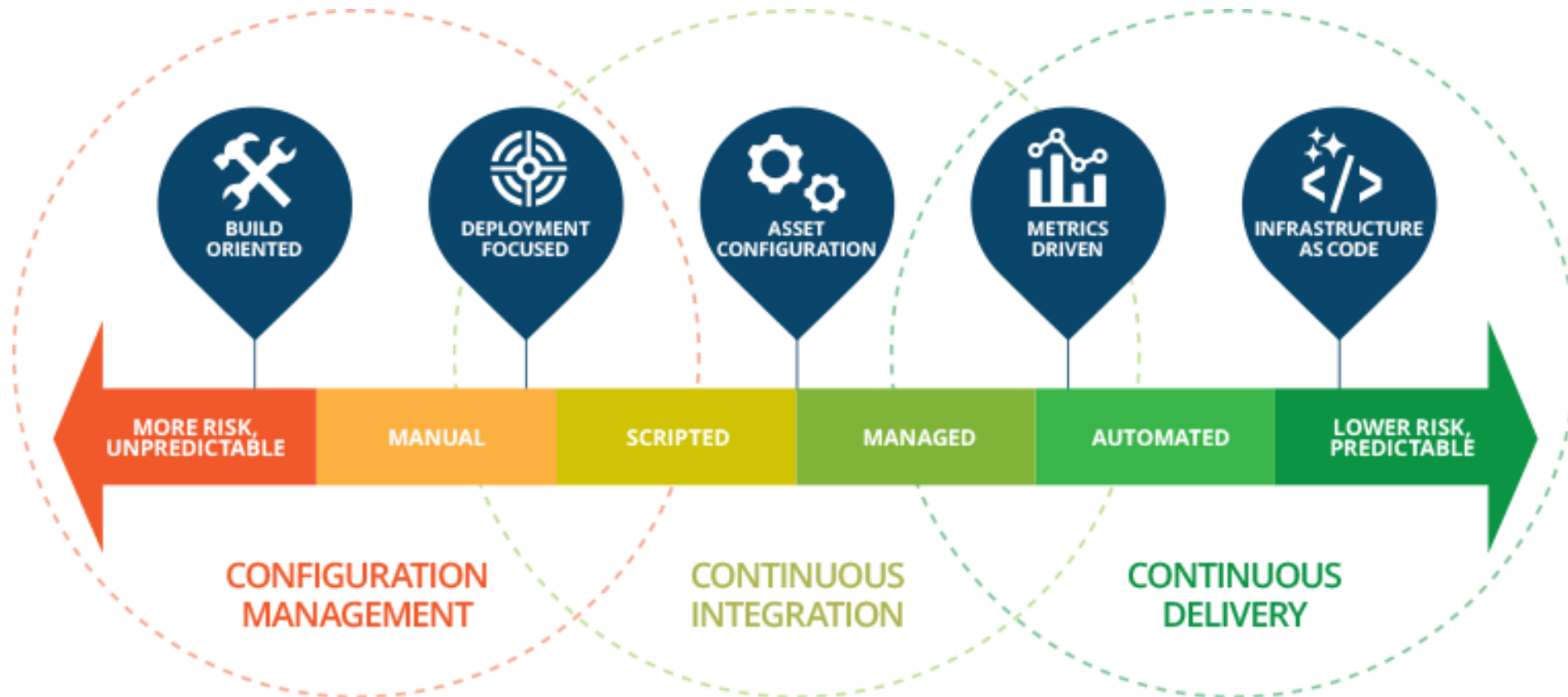
CONCEPTOS: INTEGRACIÓN CONTINUA



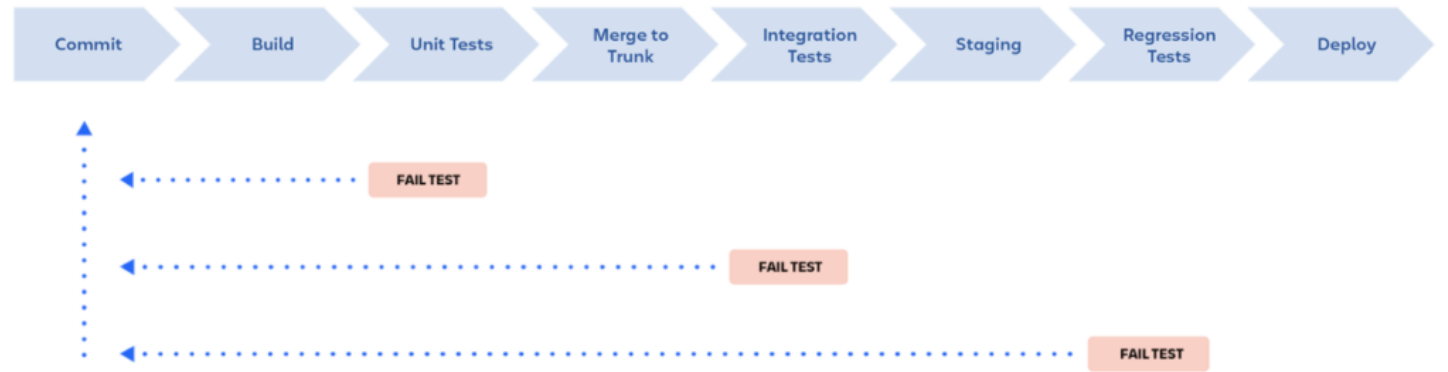
CONCEPTOS: ENTORNOS



CONCEPTOS: ENTREGA CONTINUA



CONCEPTS: PIPELINE



Collaborate

Build

Test

Deploy

Run

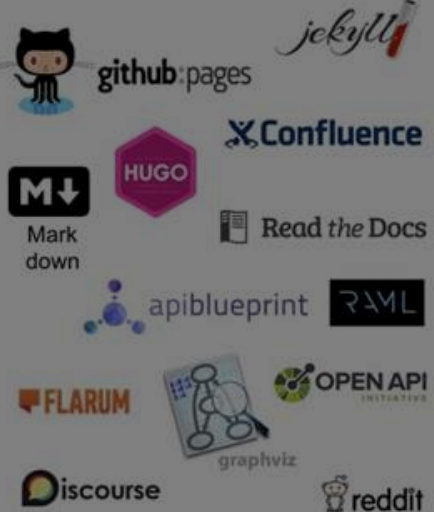
Application Lifecycle Mgmt.



Communication & ChatOps



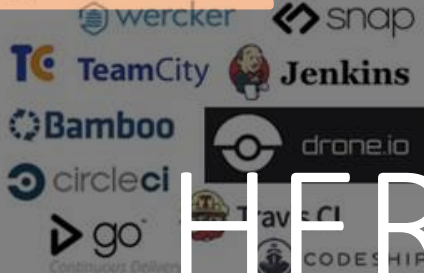
Knowledge Sharing



SCM/VCS



CI



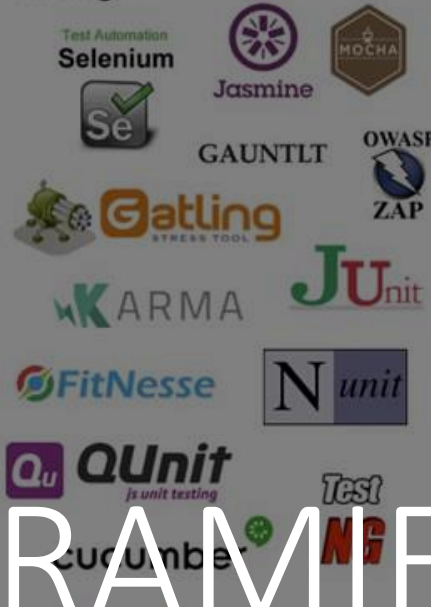
Build



Database Management



Testing



Deployment



Config Mgmt. / Provisioning



Artefact Management



Cloud / IaaS / PaaS



Orchestration & Scheduling

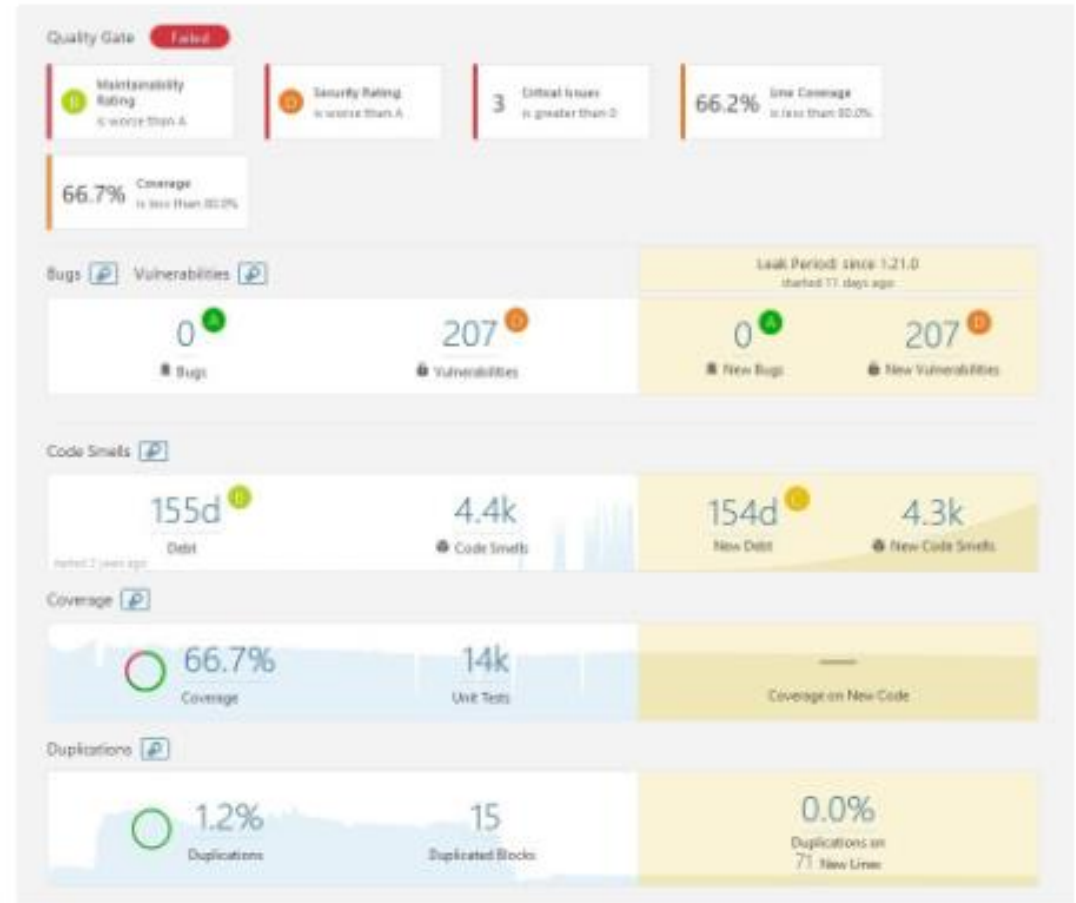


BI / Monitoring / Logging



HERRAMIENTAS

CALIDAD DE SOFTWARE



PRUEBAS

- Tipos de pruebas
- Pruebas de unidad
- Pruebas de integración
- Pruebas de aceptación
- Pruebas de rendimiento



INTRODUCCIÓN A PRUEBAS



vs.



Pruebas

Pruebas

dinámicas

TIPOS DE PRUEBAS

estáticas

TIPOS DE PRUEBAS

TIPOS DE SOFTWARE TESTING

Pruebas funcionales

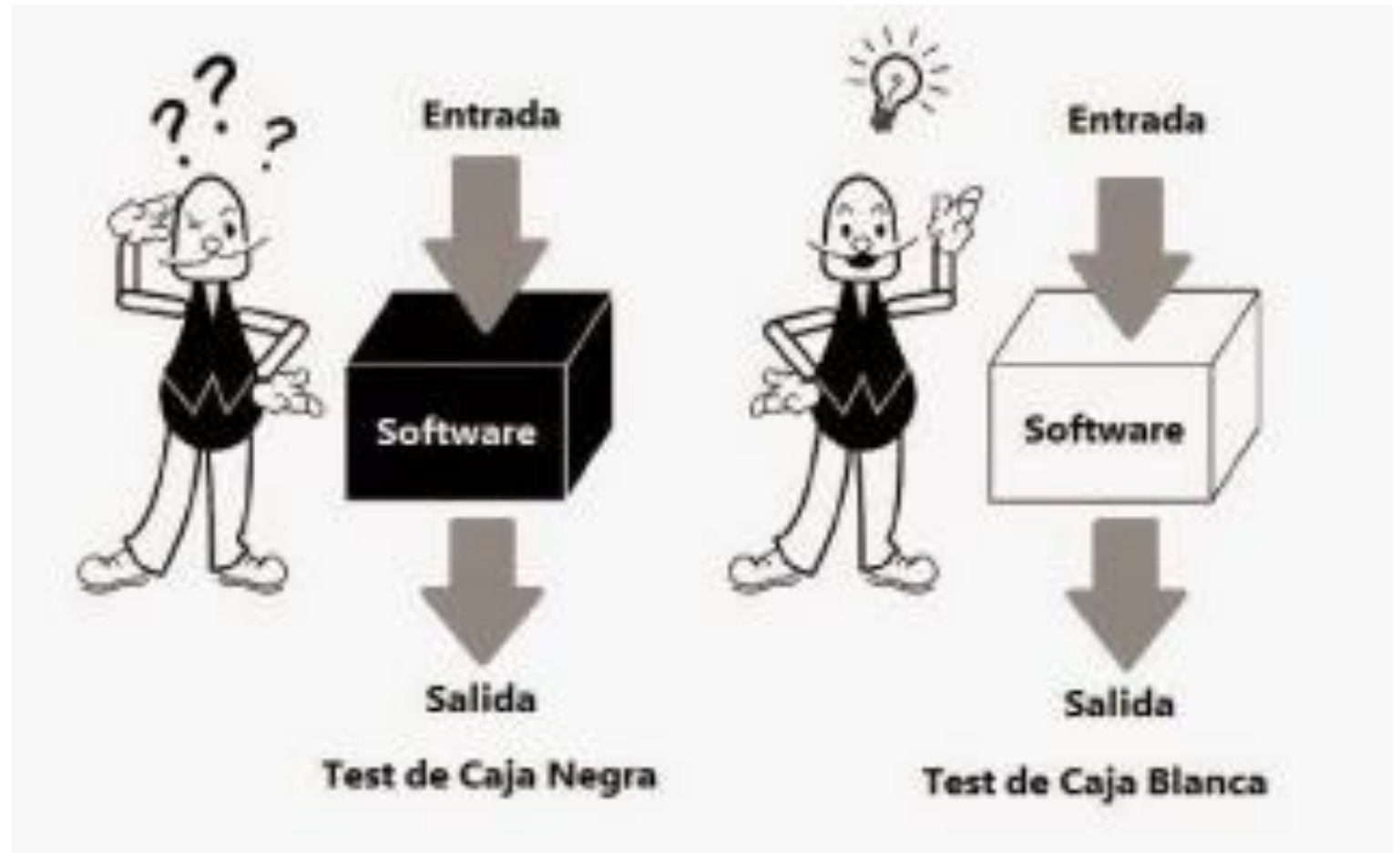
- Pruebas unitarias
- Pruebas de integración
- Pruebas de sistema
- Pruebas de sanidad
- Pruebas de humo
- Pruebas de interfaz
- Pruebas de regresión
- Pruebas de aceptación

Pruebas no funcionales

- Pruebas de rendimiento
- Prueba de carga
- Pruebas de estrés
- Pruebas de volumen
- Pruebas de seguridad
- Pruebas de compatibilidad
- Pruebas de instalación
- Pruebas de recuperación
- Pruebas de confiabilidad
- Pruebas de usabilidad
- Pruebas de conformidad
- Pruebas de localización



TIPOS DE PRUEBAS



ESTRATEGIA DE PRUEBAS



- Requisitos de análisis: recogida de requisitos iniciales
- Planificación de Pruebas
- Desarrollo de Pruebas
- Ejecución de Pruebas
- Informes de Pruebas
- Análisis de los Informes de Pruebas
- Reprueba de Fallos: Cuando se detecta un fallo se prueba esa funcionalidad
- Test de Regresión: paso de pruebas que puedan afectar a la corrección
- Cierre de las Pruebas: Finalización del proyecto

DISEÑO DE PRUEBAS

- Conocimiento de los Requisitos
- Reducción de la funcionalidad a probar
- Definición de Casos de Prueba
- Relacionar con los Casos de Usuario (historias)
- Pensar bien los costes de las pruebas
- Enfocarse en las funcionalidades más críticas
- Cada prueba requiere su herramienta

HERRAMIENTAS

Pruebas Unitarias: Junit

Pruebas de Integración: Mockito

Pruebas de Aceptación/Funcionalidad Web: Selenium

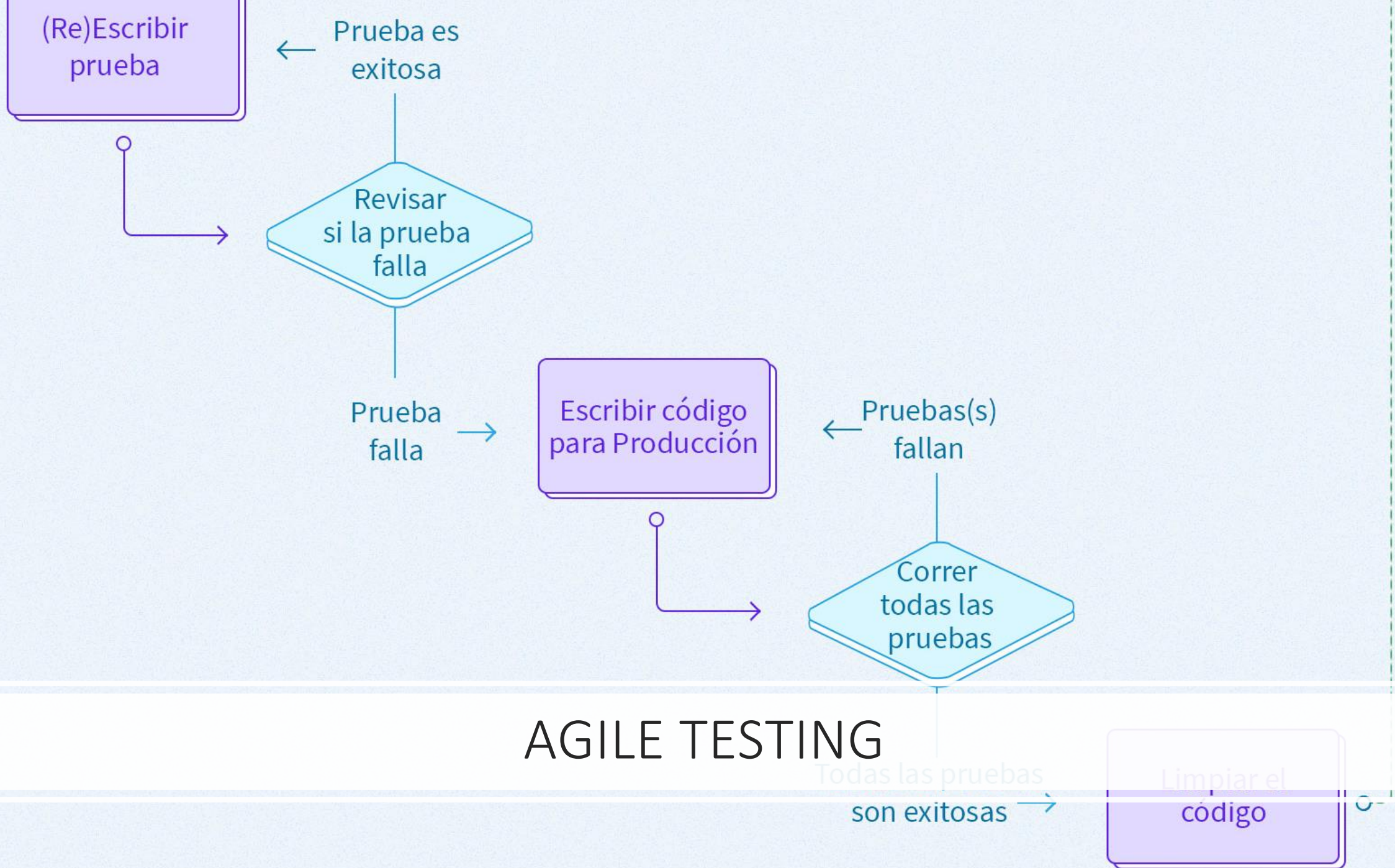
Bases de Datos/Web Rendimiento: Jmeter/Blazemeter/Gatling

Servidor de Integración Continua: Jenkins/TravisCI/CircleCI

Herramienta de Seguimiento de Proyectos e Incidencias: Jira/Mantis

Automatizador de la Compilación, Despliegue y Ejecución de proyectos: Maven/Gradle/NPM

Despliegue de aplicaciones: Docker/Kubernetes



AGILE TESTING

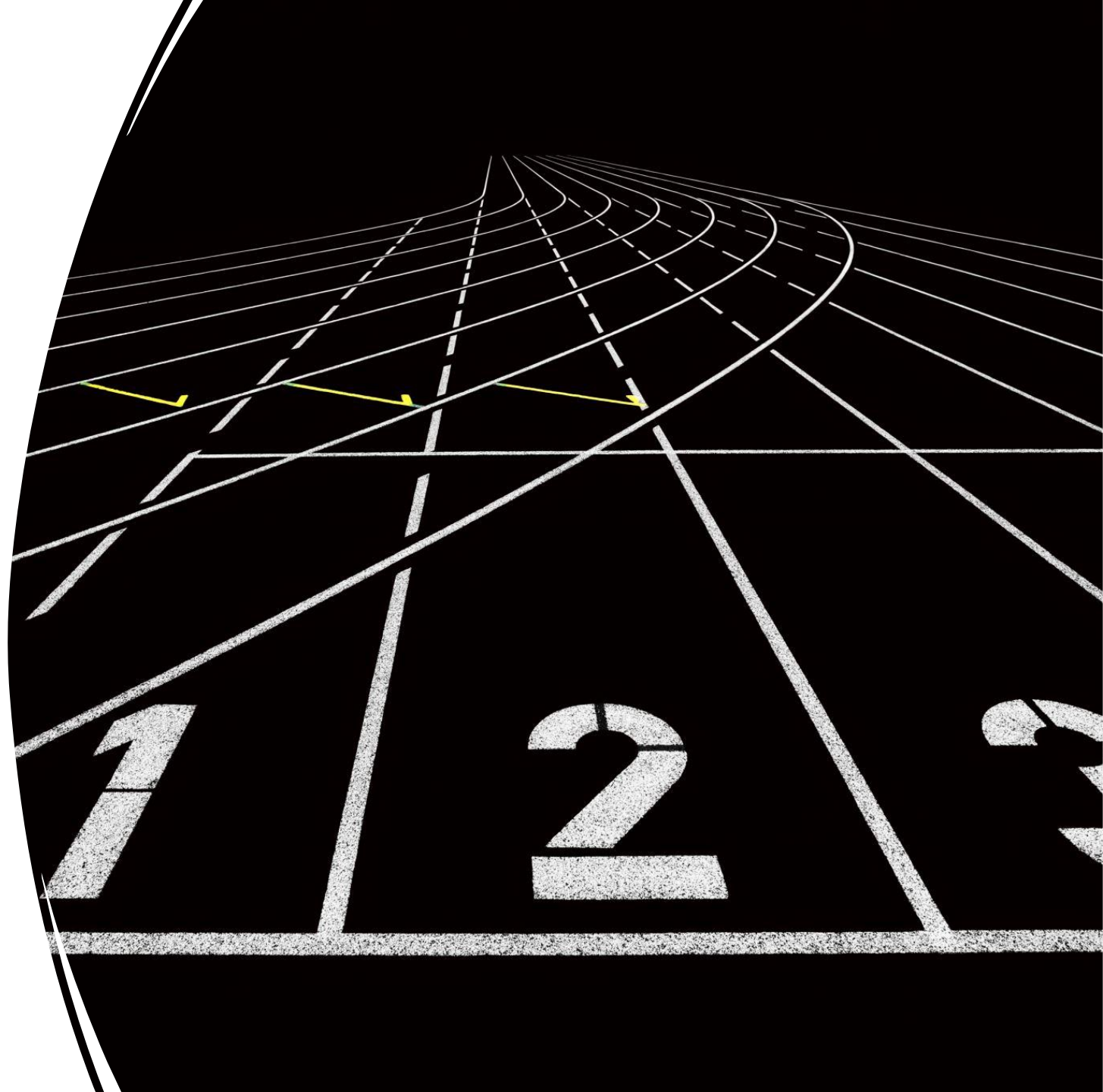
AUTOMATIZACIÓN DE LAS PRUEBAS



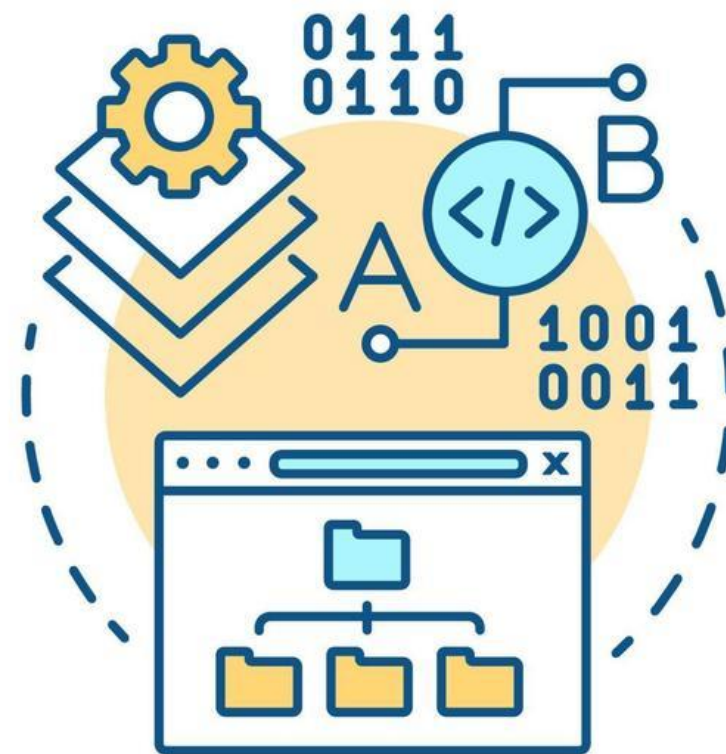


LABORATORIO 1: PRUEBAS UNITARIAS EN PYTHON

EJERCICIO 1



DEMO: PRUEBAS DE INTEGRACIÓN EN PYTHON



Integration Testing



EDITABLE STROKE

EJERCICIO 2



LABORATORIO 2: PRUEBAS DE RENDIMIENTO EN PYTHON



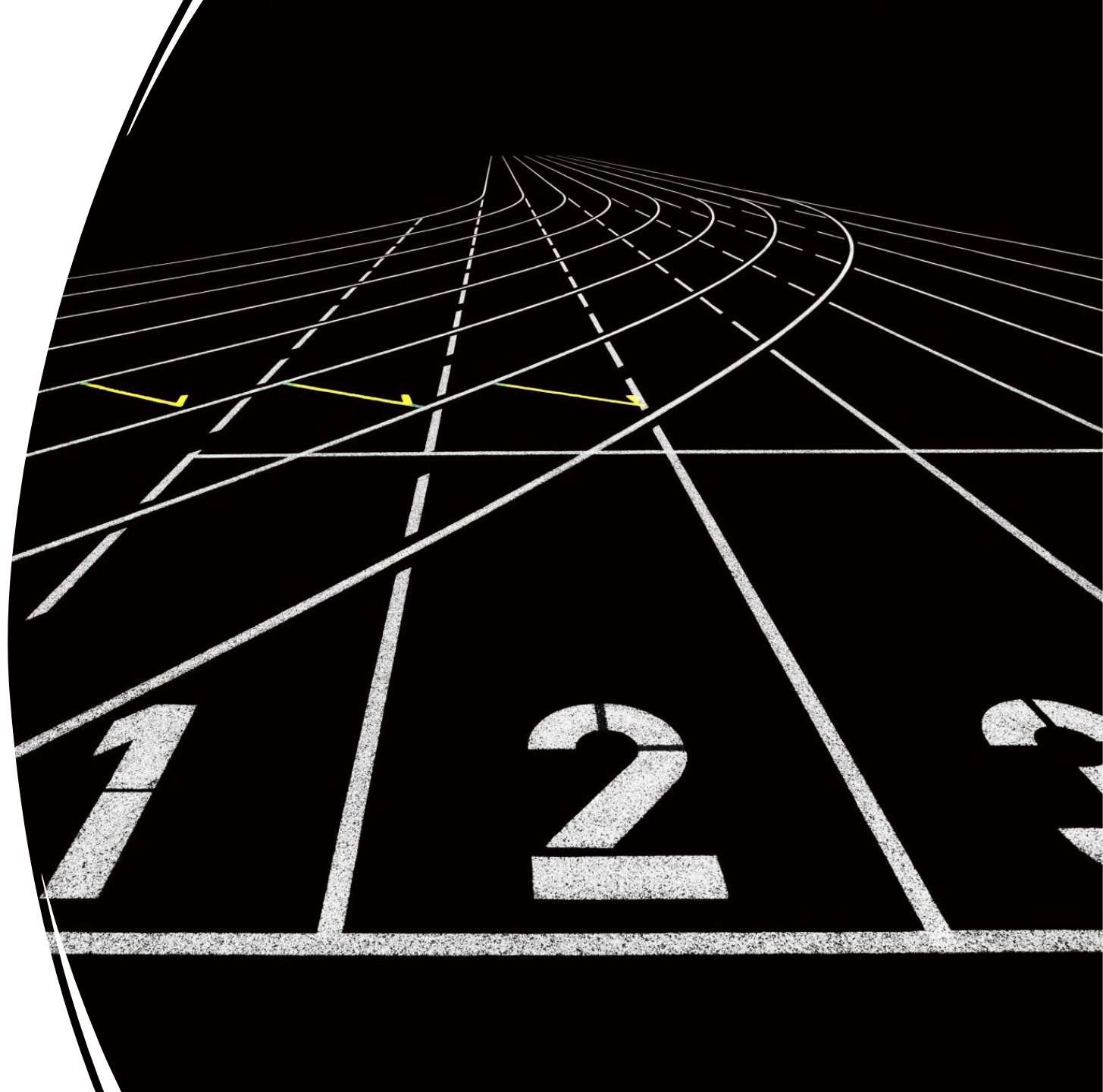
STRESS
TESTING

VOLUME
TESTING

SOAK
TESTING

S

EJERCICIO 3



MAVEN

- Introducción
- Estructura
- Fichero pom.xml
- Compilación del proyecto
- Dependencias
- Repositorios
- Objetivos
- Plugins
- Despliegue
- Integración con IDE's

Maven™

The word "Maven" is written in a bold, black, sans-serif font. The letter "v" is replaced by two crossed feathers. The feathers have a gradient of colors, starting with yellow at the tips, transitioning through orange and red, and ending in purple at the base. The feathers are crossed at their bases, with one feather slightly behind the other. A trademark symbol (TM) is located to the upper right of the word. The entire logo is set against a solid orange background.

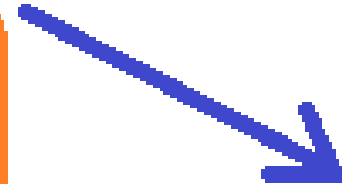
Pom File

- Dependencies
- Plugins
- Life Cycle Phases and Goals
- Build Profiles

Maven
Repository

Maven

Read Pom.xml



LABORATORIO 1: INSTALAR MAVEN




```
-- pom.xml : fichero de configuración de maven
-- src : carpeta principal de código
|-- main: código de la aplicación
|   |-- java : código Java
|   |   |-- com
|   |       |-- mycompany
|   |           |-- app : Paquete principal
|   |               |-- App.java : Código de la aplicación
|-- test : Código de pruebas
|   |-- java : Código Java
|   |   |-- com
|   |       |-- mycompany
|   |           |-- app : Paquete principal
```

Creación de un proyecto Maven

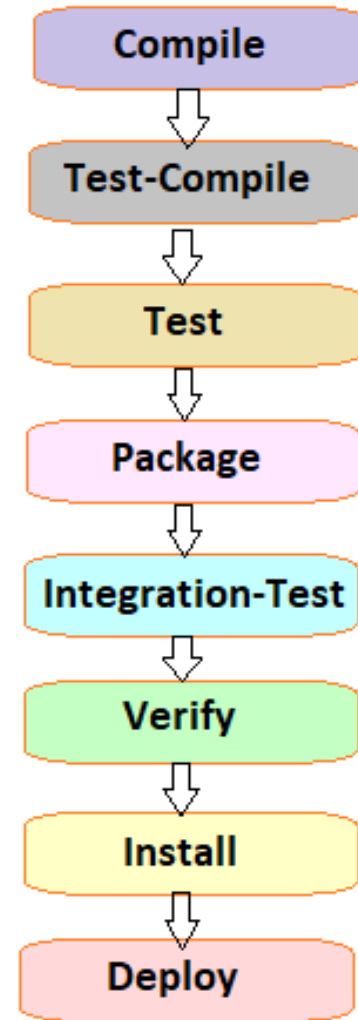
```
.B archetype:generate \
  -D archetypeGroupId=org.apache.maven.archetype \
  -D archetypeArtifactId=com.mycompany.app \
  -D archetypeVersionId=1.0-SNAPSHOT \
  -D groupId=com.mycompany \
  -D artifactId=my-app
```

ARQUETIPOS

POM.XML

```
<?xml version="1.0"?>
- <project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-
  v4_0_0.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://maven.apache.org/POM/4.0.0">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>my-app</name>
  <url>http://maven.apache.org</url>
  - <dependencies>
    - <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

CICLO DE VIDA MAVEN



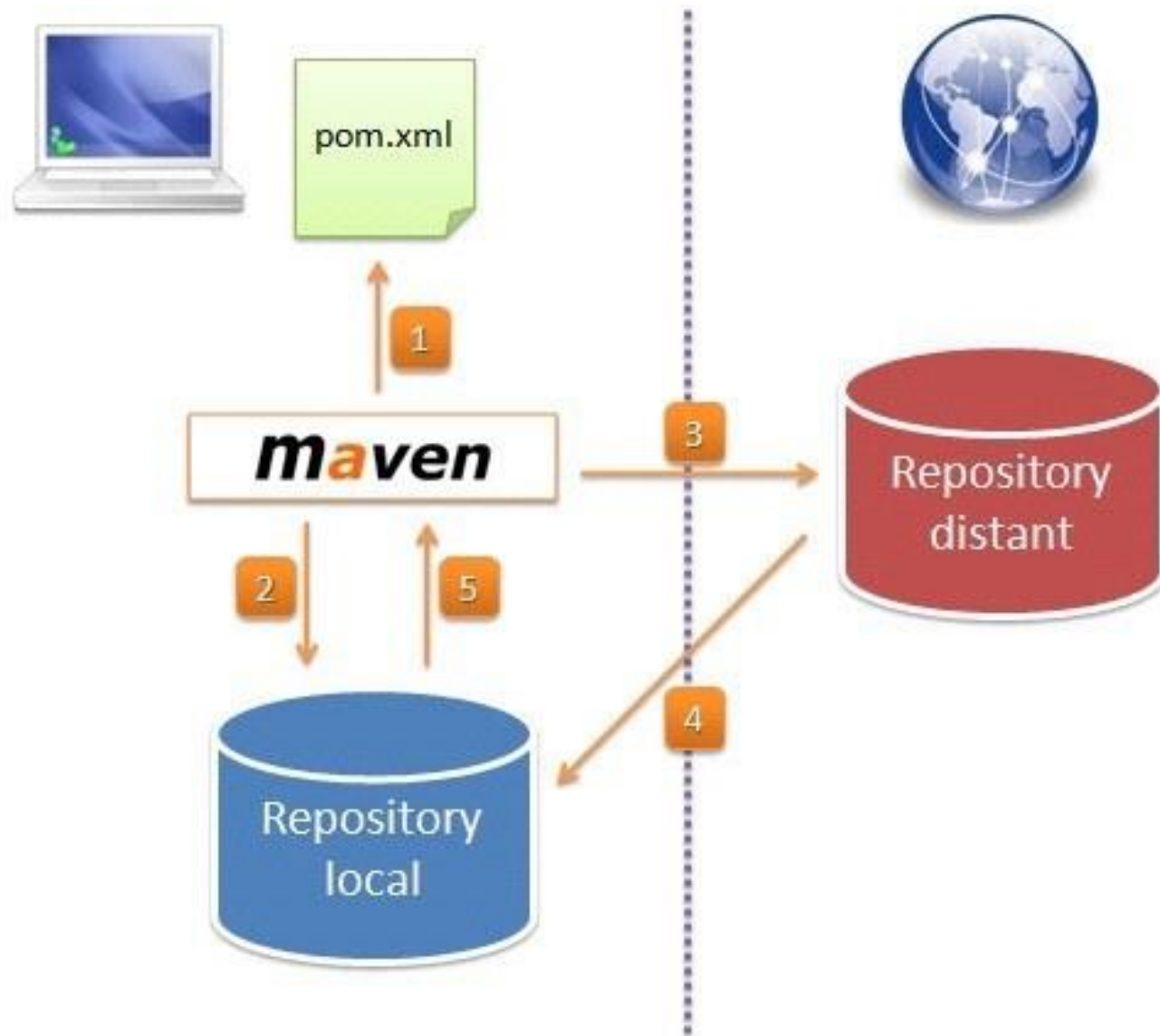
DEPENDENCIAS

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/
4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
<...>
<dependencies>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>3.8.1</version>
<scope>test</scope>
</dependency>
</dependencies>
</project>
```

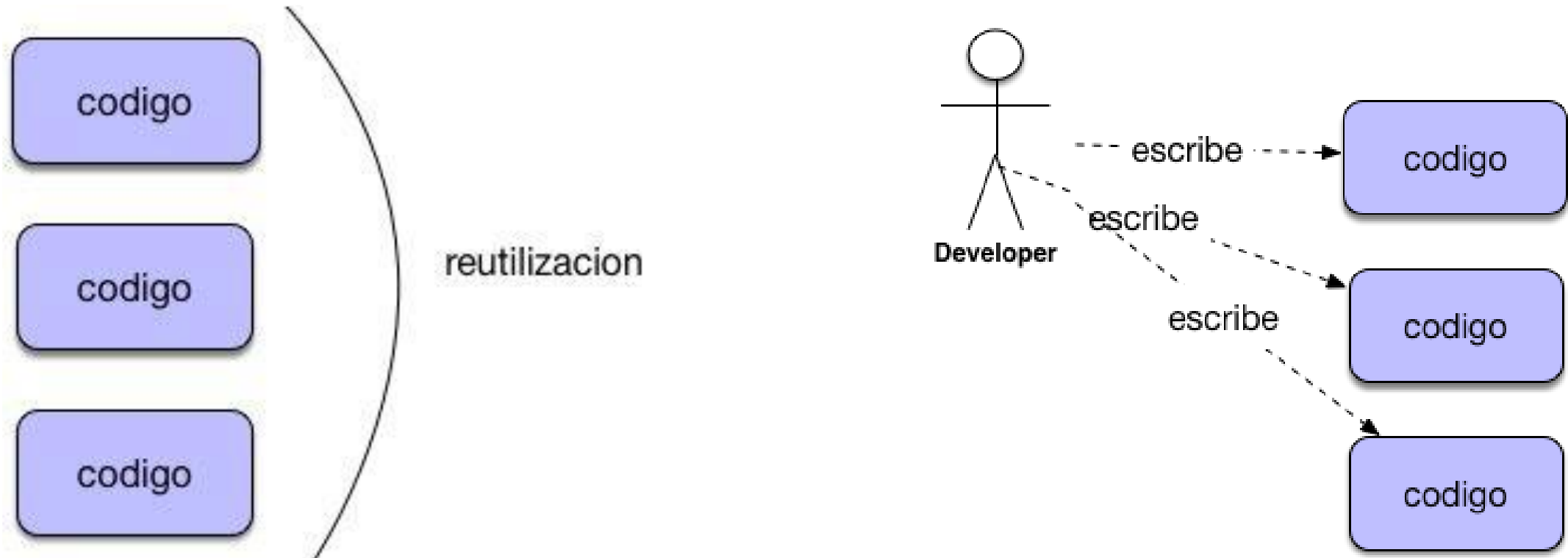

LABORATORIO 2



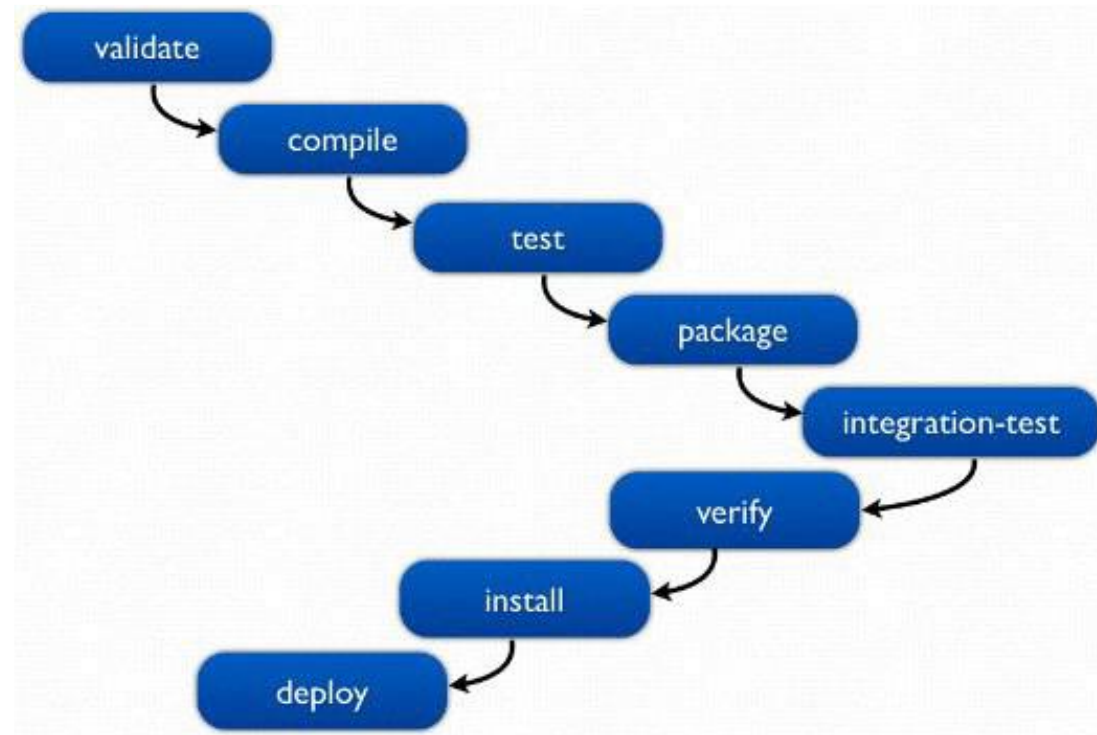
REPOSITARIOS



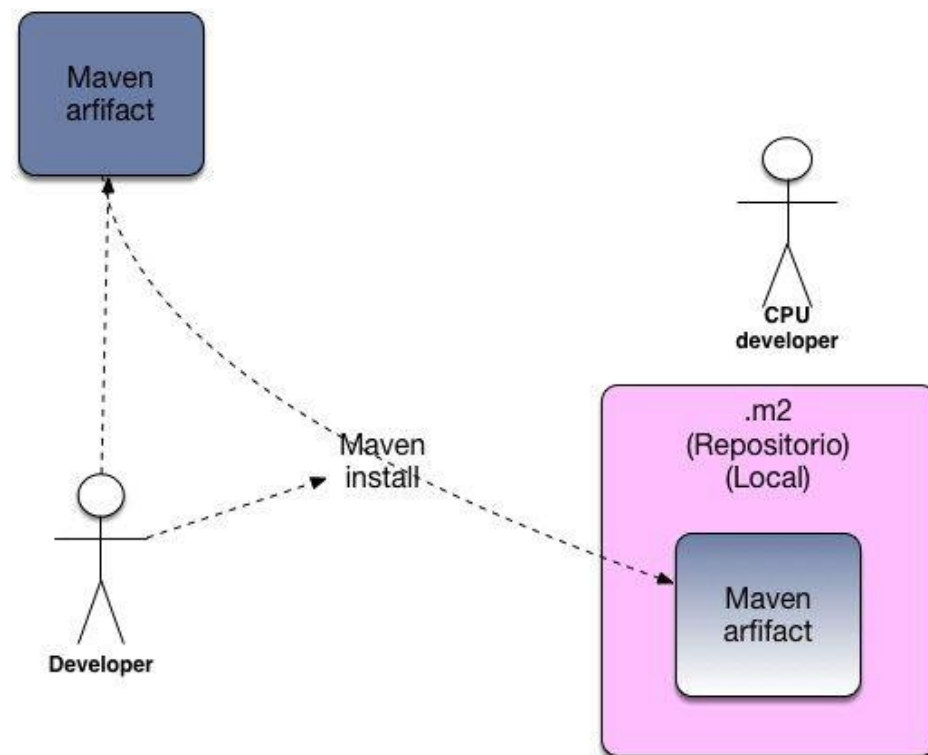
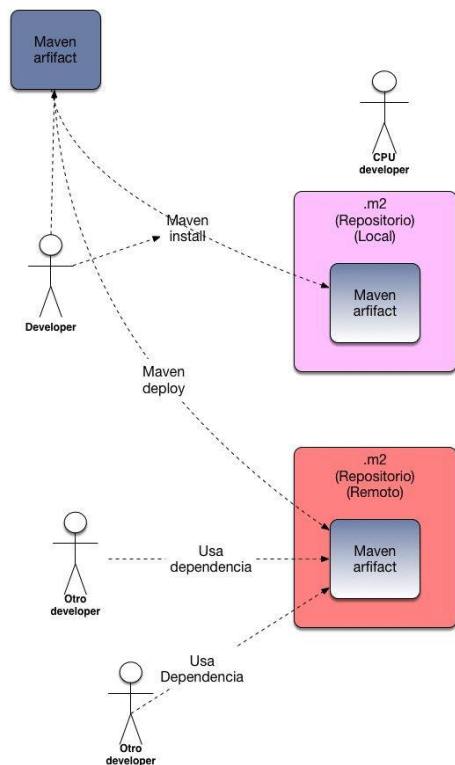
ARTEFACTOS



OBJETIVOS



INSTALL Y DEPLOY





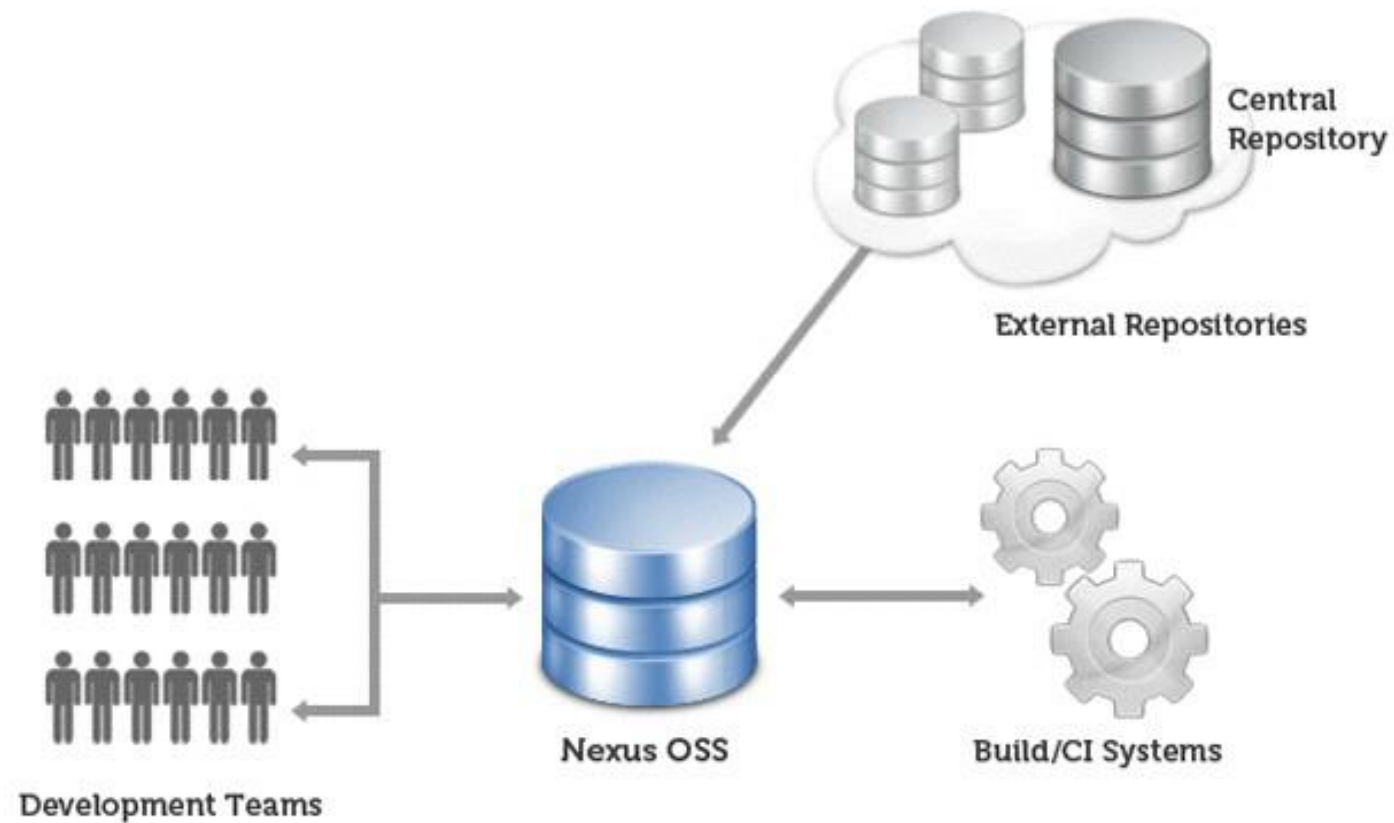
LABORATORIO 3

ARTIFACTORY



LABORATORIO 4

NEXUS



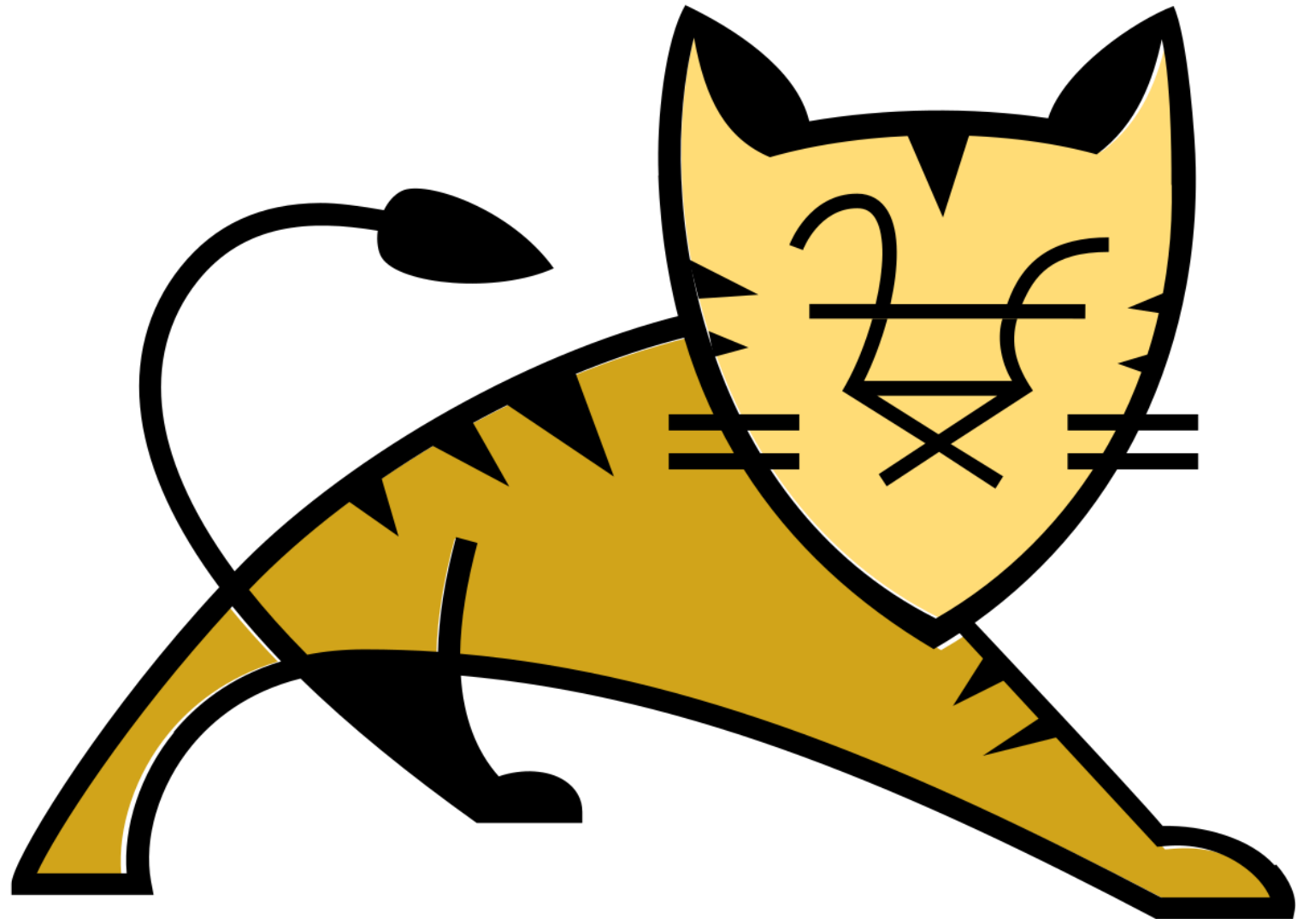
LABORATORIO 5



PLUGINS

```
1 <project>
2   ...
3   <build>
4     ...
5     <plugins>
6       <plugin>
7         <groupId>org.apache.maven.plugins</groupId>
8         <artifactId>maven-compiler-plugin</artifactId>
9         <configuration>
10          <source>1.5</source>
11          <target>1.5</target>
12        </configuration>
13      </plugin>
14    </plugins>
15    ...
16  </build>
17  ...
18 </project>
```

TOMCAT



LABORATORIO 6

SECURITY TESTING

- Introducción
- SonarQube
- Reglas
- Cobertura de código
- Smells
- Bugs
- Fortify
- Análisis
- Reporting

INTRODUCCIÓN



test-node-js master

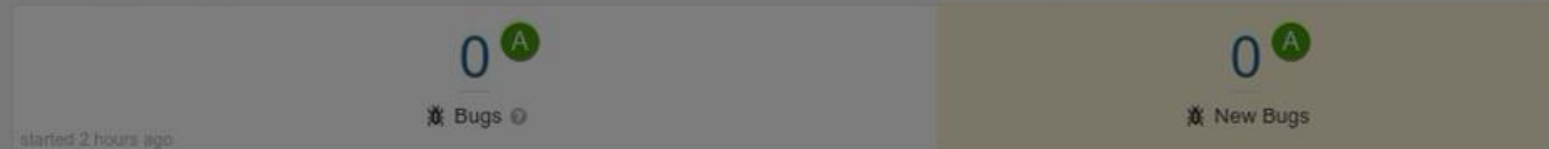
January 20, 2020, 5:07 PM Version not provided

Overview Issues Measures Code Activity Administration

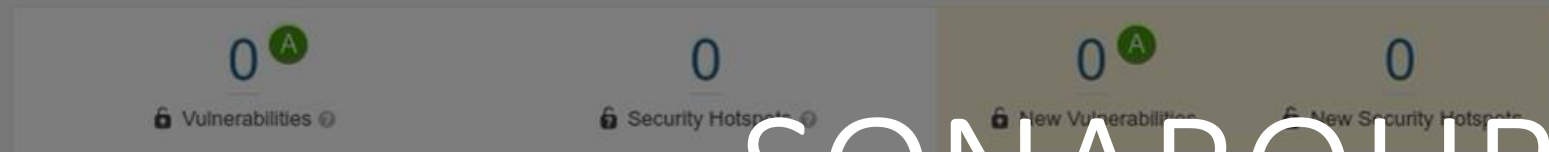
Quality Gate

Passed

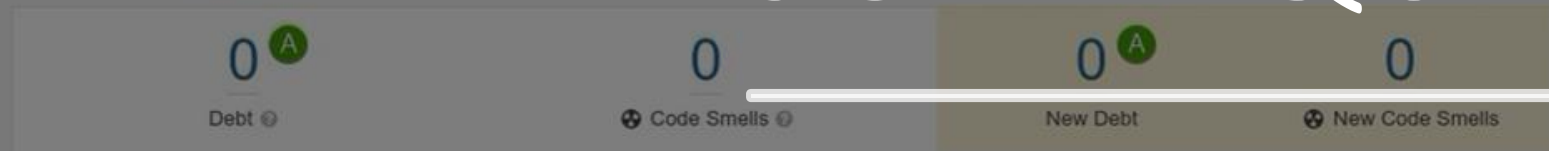
Reliability Measures



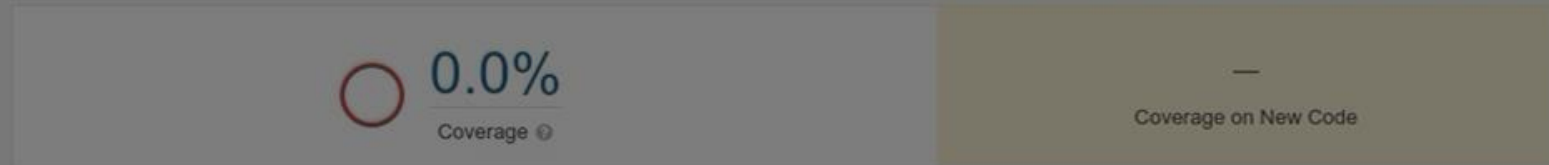
Security Measures



Maintainability Measures



Coverage Measures



Duplications Measures



About This Project

No tags

XS 8

Lines of Code

JavaScript 8

Project Activity

January 20, 2020

Not provided

January 20, 2020

Project Analyzed

January 20, 2020

Project Analyzed

Show More

Quality Gate

(Default) Sonar way

Quality Profiles

(JavaScript) Sonar way

Project Key

test-node-js

Copy

Get project badges

SONARQUBE



T-SQL

<xml />

PL/SQL

ABAP

Apex

COBOL



LENGUAJES

LABORATORIO 1

REGLAS

- SonarQube ejecuta reglas sobre el código fuente para generar problemas. Hay cuatro tipos de reglas:
- Smells
- Bug
- Vulnerabilidad
- Punto de acceso de seguridad

ISSUE

sonarqube

ProjectsIssuesRulesQuality ProfilesQuality GatesAdministration

?

Search for projects, sub-projects and files...

A

trxmaster

November 3, 2018, 5:34 PMVersion 1

OverviewIssuesSecurity ReportsMeasuresCodeActivityAdministration

My IssuesAll

Filters

Clear All Filters

Type

Bug6

Vulnerability0

Code Smell7

Security Hotspot0

Severity

Blocker13

Critical876

Major3.5k

Minor2.2k

Info104

Resolution

Status

Creation Date

Language

Rule

Standard

Tag

Module

Directory

File

Assignee

Author

Bulk Change

to select issues

to navigate

1 / 13 issues

src/.../org/tron/common/runtime/vm/DataWord.java

Remove this "clone" implementation; use a copy constructor or copy factory instead.

Code SmellBlockerOpenNot assigned30min effortComment

4 months agoL362suspicious

src/.../java/org/tron/common/storage/Key.java

Remove this "clone" implementation; use a copy constructor or copy factory instead.

Code SmellBlockerOpenNot assigned30min effortComment

4 months agoL41suspicious

src/.../java/org/tron/common/storage/Type.java

Remove this "clone" implementation; use a copy constructor or copy factory instead.

Code SmellBlockerOpenNot assigned30min effortComment

4 months agoL36suspicious

src/.../java/org/tron/common/storage/Value.java

Remove this "clone" implementation; use a copy constructor or copy factory instead.

Code SmellBlockerOpenNot assigned30min effortComment

4 months agoL60suspicious

src/test/java/org/tron/core/db/TransactionTraceTest.java

Rename field "OwnerAddress" to prevent any misunderstanding/clash with field "ownerAddress" defined on line 65.

Code SmellBlockerOpenNot assigned10min effortComment

3 months agoL76confusing

src/test/java/org/tron/core/net/UdpTest.java

Use try-with-resources or close this "DatagramSocket" in a "finally" clause.

BugBlockerOpenNot assigned5min effortComment

5 months agoL81cert, cwe, denial-of-service, leak

src/.../tron/wallet/common/client/WalletClient.java

Rename method "getBlock" to prevent any misunderstanding/clash with method "GetBlock" defined on line 419.

Code SmellBlockerOpenNot assigned10min effortComment

6 months agoL892confusing

src/.../wallet/common/client/utis/DataWord.java

REGLAS PERSONALIZADAS

Track uses of disallowed dependencies

squid:S3417  

 Code Smell  Major  Main sources  maven  Available Since Jan 13, 2016 SonarAnalyzer (Java) **Rule Template**

Whether they are disallowed locally for security, license, or dependability reasons, forbidden dependencies should not be used.

This rule raises an issue when the group or artifact id of a direct dependency matches the configured forbidden dependency pattern.

Noncompliant Code Example

With a parameter of: `*:.*log4j.*`

```
<dependency> <!-- Noncompliant -->
  <groupId>log4j</groupId>
  <artifactId>log4j</artifactId>
  <version>1.2.17</version>
</dependency>
```

[Extend Description](#)

Parameters

dependencyName Pattern describing forbidden dependencies group and artifact ids. E.G. `'*:.*log4j'` or `'x.y:*'`

version Dependency version pattern or dash-delimited range. Leave blank for all versions. E.G. `'1.3.*'`, `'1.0-3.1'`, `'1.0-*'` or `'*-3.1'`

Custom Rules

[Create](#)

Create a new rule
from this template

Existing custom rule
from this template

["commons-collections" should not be used](#)  Major dependencyName: commons-collections:commons-collections

[Delete](#)

Click-through for custom rule details

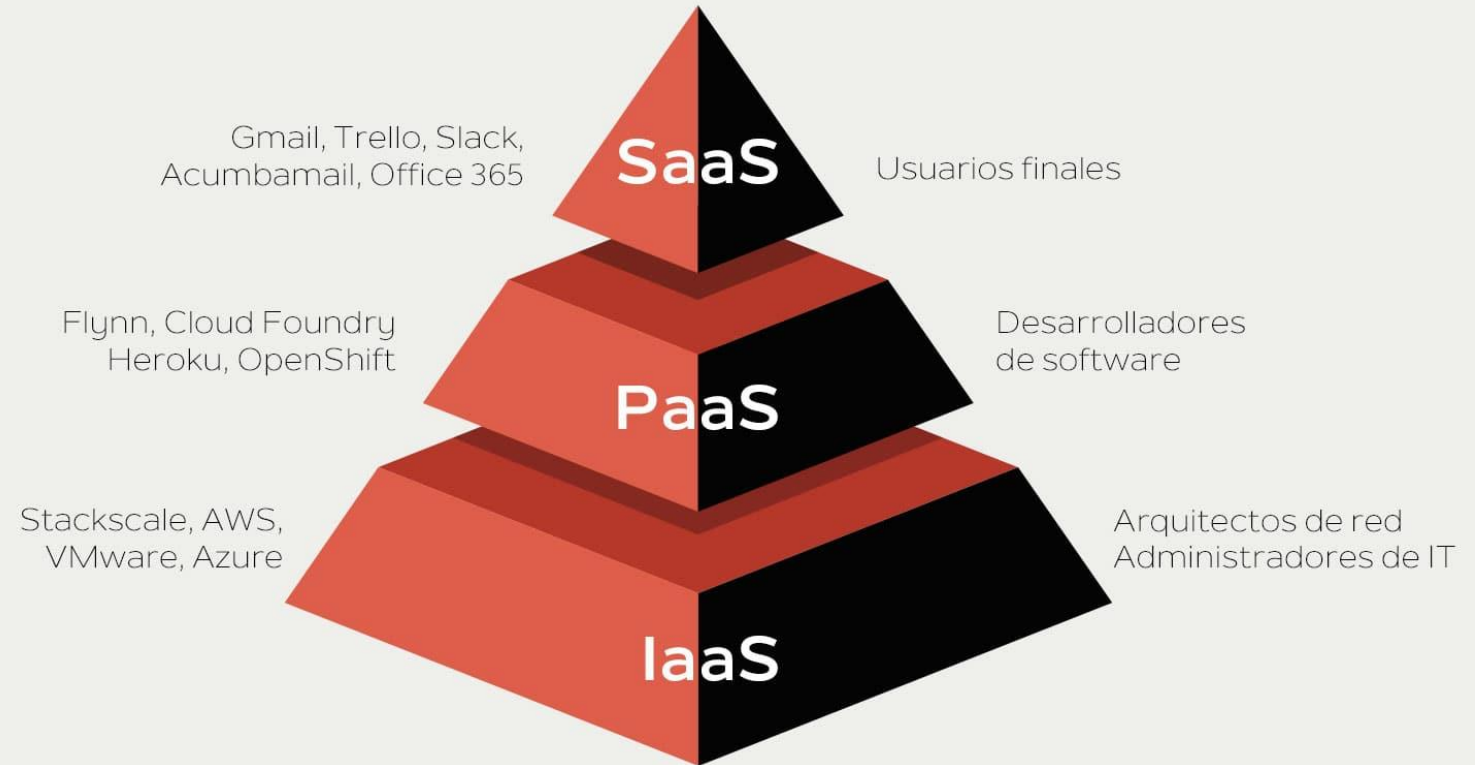
LABORATORIO 2

SONARCLOUD

sonarcloud 

MODELOS DE SERVICIO

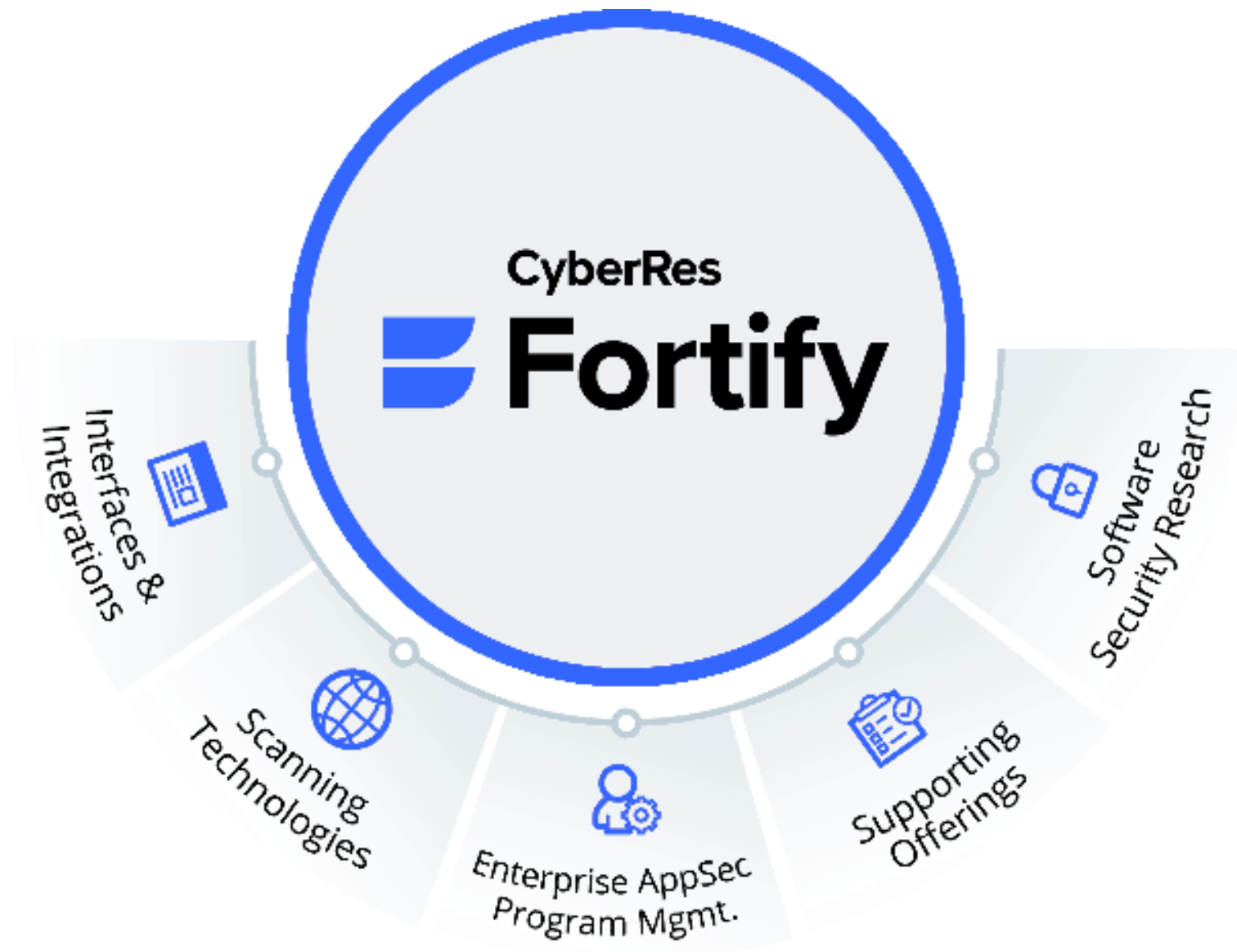
Modelos de servicio cloud



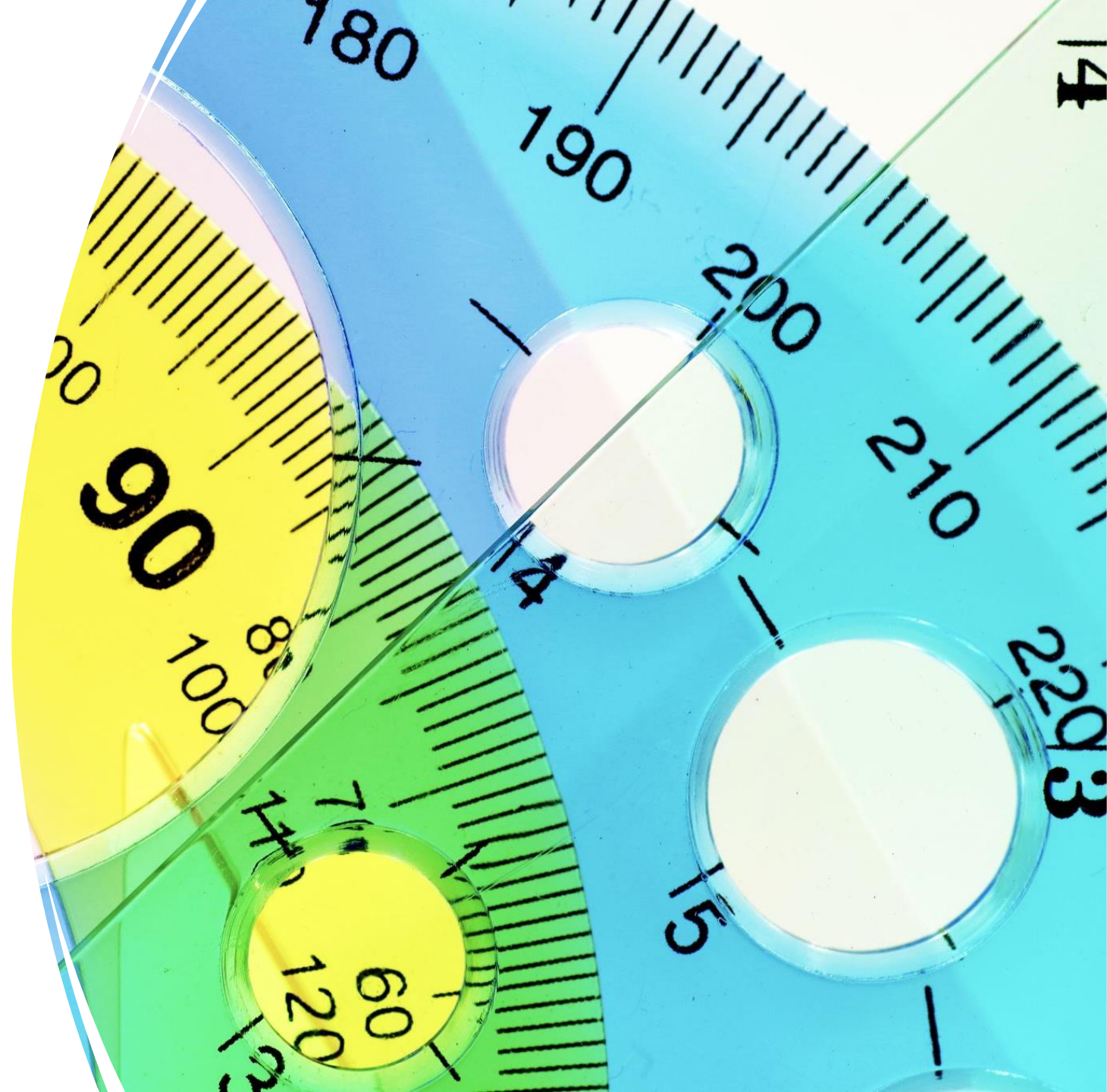
LABORATORIO 3

LABORATORIO 4

FORTIFY



SONAR VS FORTIFY



LABORATORIO 5

EJERCICIO 4
