DevSecOps

¿Qué es DevSecOps?

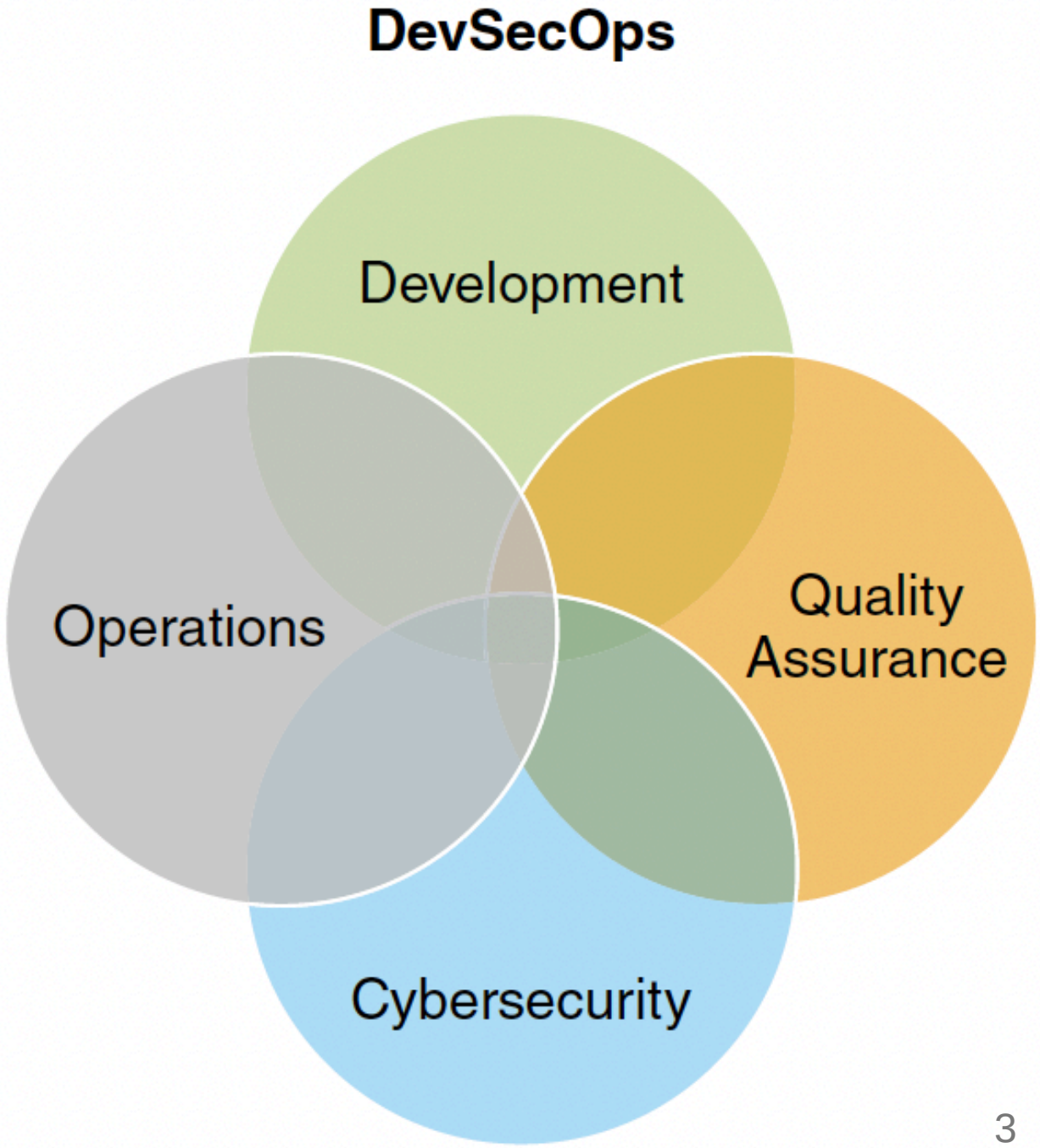
Qué es DevSecOps?

Seguridad integrada en el SLDC

DevSecOps vs SecDevOps

Visiones

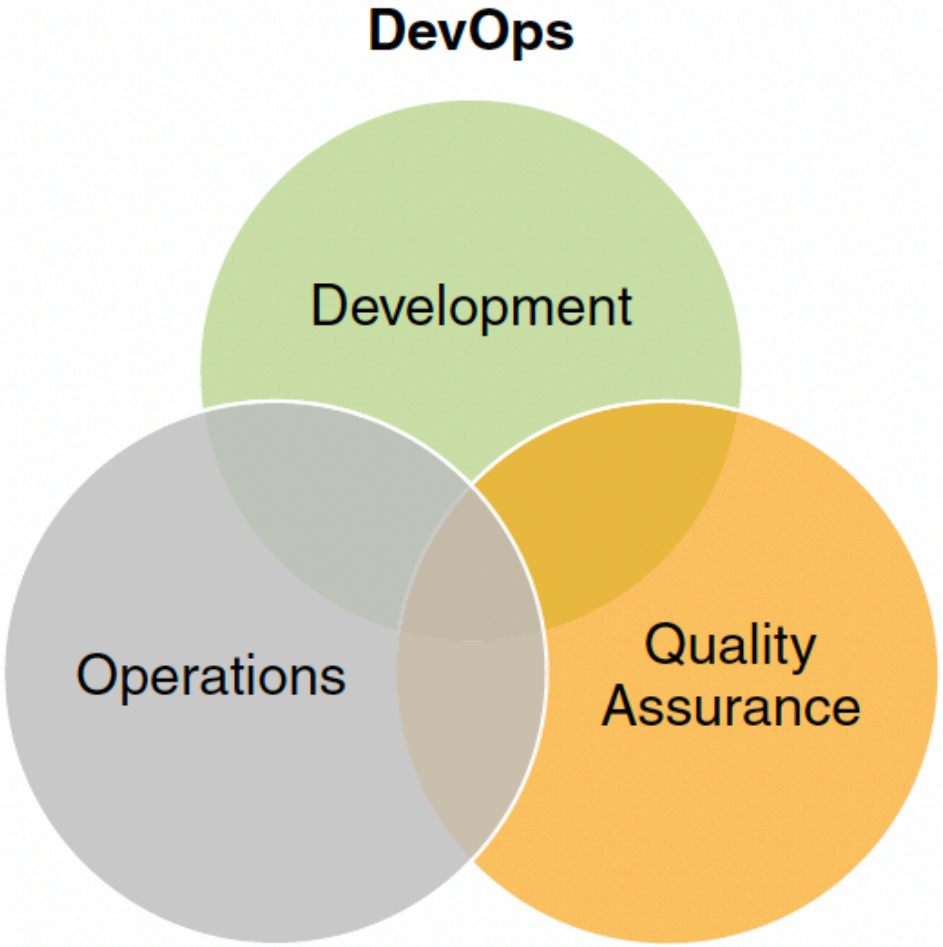
People, Process, Technology, Governance



¿Qué es DevOps?

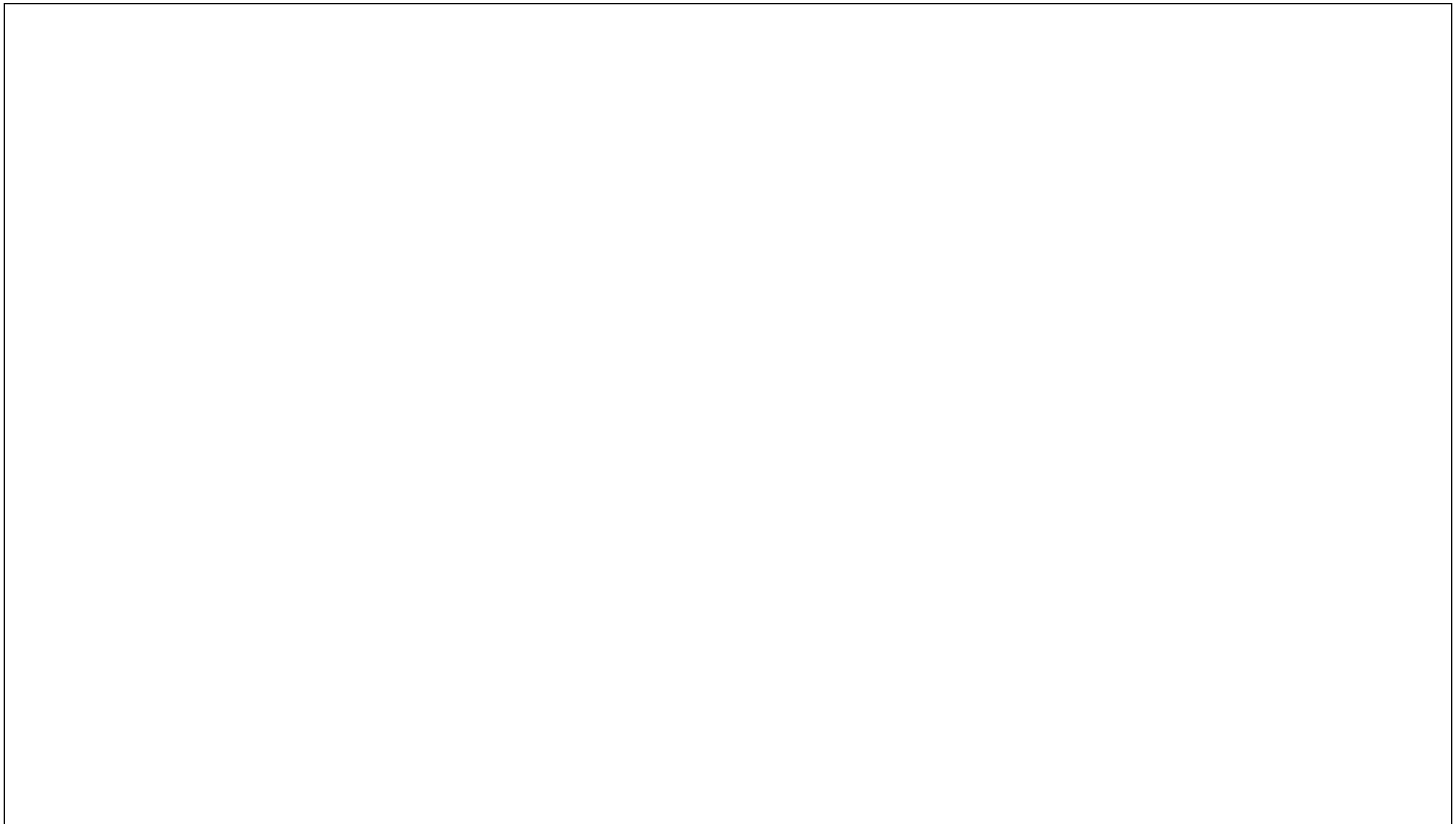
¿Qué es DevOps?

- Development + Operations
  - Concepción > Desarrollo > Entrega
- Procesos ágiles
  - *Lean manufacturing*



	Definiciones
Desarrollo	Se refiere al proceso de crear software, donde los desarrolladores escriben y actualizan el código fuente de las aplicaciones.
Operaciones	Se centran en la gestión y mantenimiento de los sistemas y la infraestructura en los que se ejecuta el software. Incluye tareas como la configuración, el monitoreo y la resolución de problemas.

¿Qué es DevOps?



Elementos clave en DevOps

- Deployment (despliegues) frecuentes
- Pruebas automáticas
  - TDD: *Test-driven design*
  - BDD: *Behavior-driven design*
- CI/CD: *Continuous Integration* + *Continuous Delivery*
- Feedback de usuarios
- Monitorización de apps/infraestructura

¿Qué significan **integración**, **entrega** (*delivery*) y **despliegue** (*deployment*)?

On your marks, get set,... go!

🟧	**Definiciones**
Integración continua	Llevar automáticamente los cambios de varios desarrolladores en el código de una aplicación a un repositorio compartido para cada nueva versión.
Entrega continua	Trasladar la aplicación de software desde el entorno de desarrollo y dejarla disponible para su despliegue en un entorno de producción. Incluye pruebas, empaquetado y preparación de cada *release*.
Despliegue	Instalación de una aplicación en su entorno de producción, ya sea en un servidor, un conjunto de servidores, un contenedor, la nube, etc.


¿DevOps es un nuevo rol?

¿Por qué DevOps?

Motivación

- Sistemas frágiles

  $\Leftarrow$ Falta de comunicación y herramientas

  $\Rightarrow$ Despliegues complejos y propensos a errores

- Deuda técnica
- Arquitectura poco sólida
- Requisitos no funcionales poco solventes

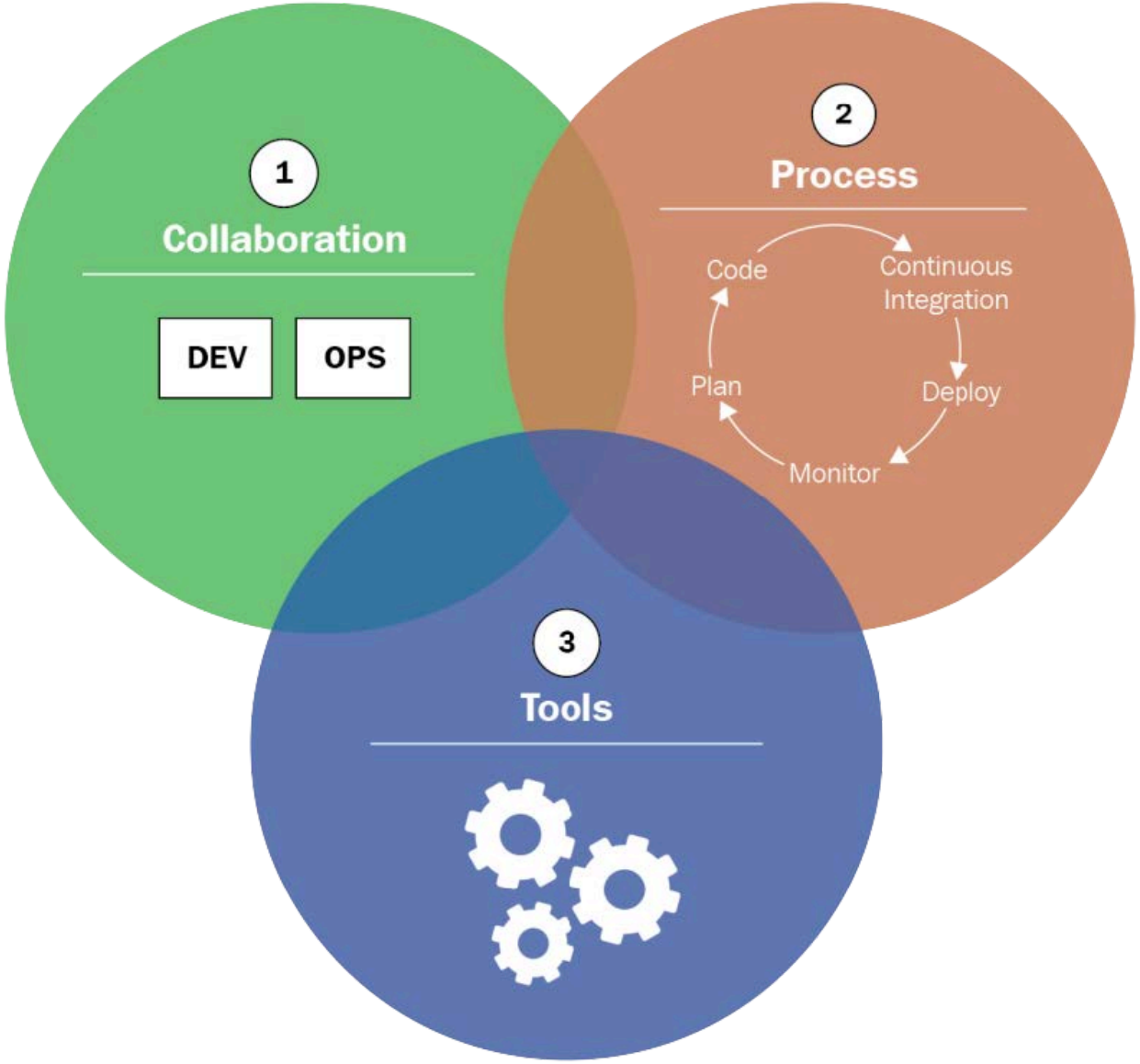
	Definiciones
Deuda técnica	Decisiones tomadas durante el desarrollo de un software que, en el corto plazo, permiten un desarrollo más rápido o una solución temporal, pero que crean problemas a largo plazo en términos de NFR
Requisitos No Funcionales (NFR)	Aspectos que no están relacionados directamente con la funcionalidad de un sistema software, sino con características no directamente vinculados a sus funciones específicas (rendimiento, usabilidad, confiabilidad, seguridad, eficiencia, etc.)
Arquitectura software	Estructura y diseño organizativo de un sistema de software, sobre cómo sus componentes interactúan entre sí y cómo se organizan para lograr sus objetivos de manera efectiva. Proporciona un marco conceptual para abordar aspectos de los NFR.

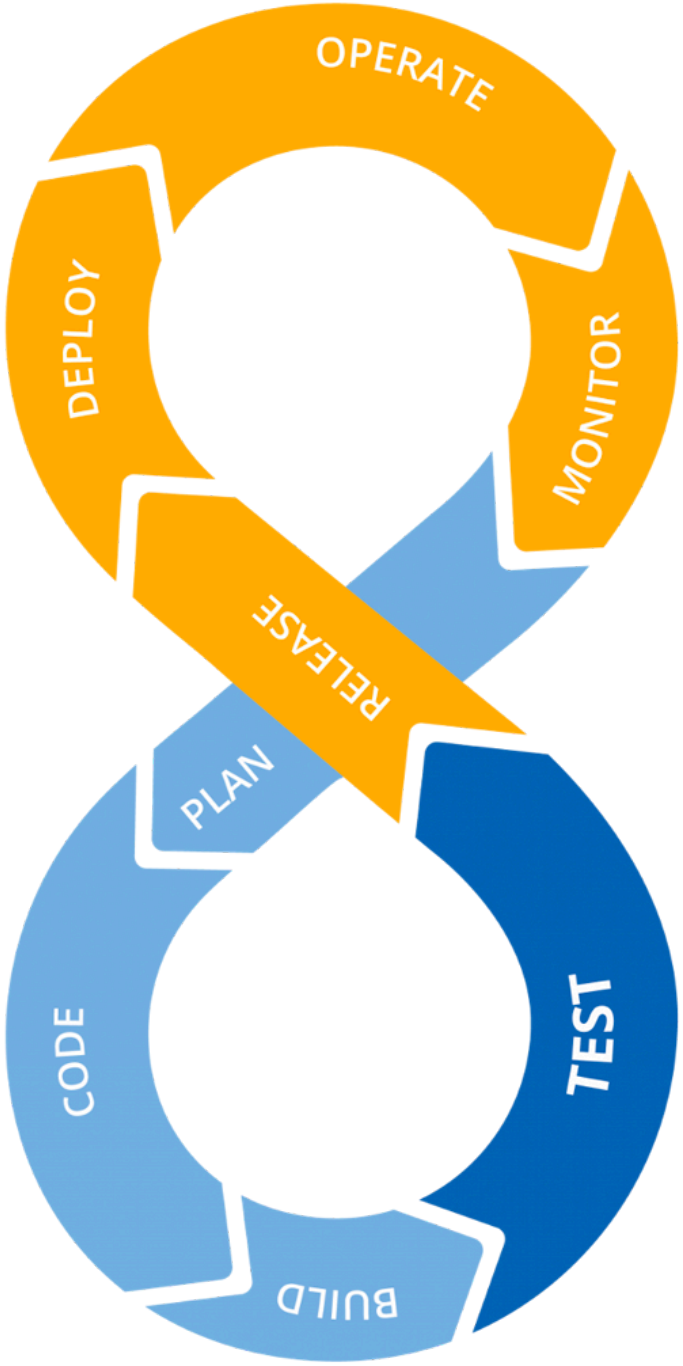
Cultura DevOps

Definición de Donovan Brown

> DevOps es la unión de personas, procesos y productos para una entrega continua de valor a los usuarios finales

- Procesos con Agilidad
- Personas en Colaboración
- Productos con Herramientas



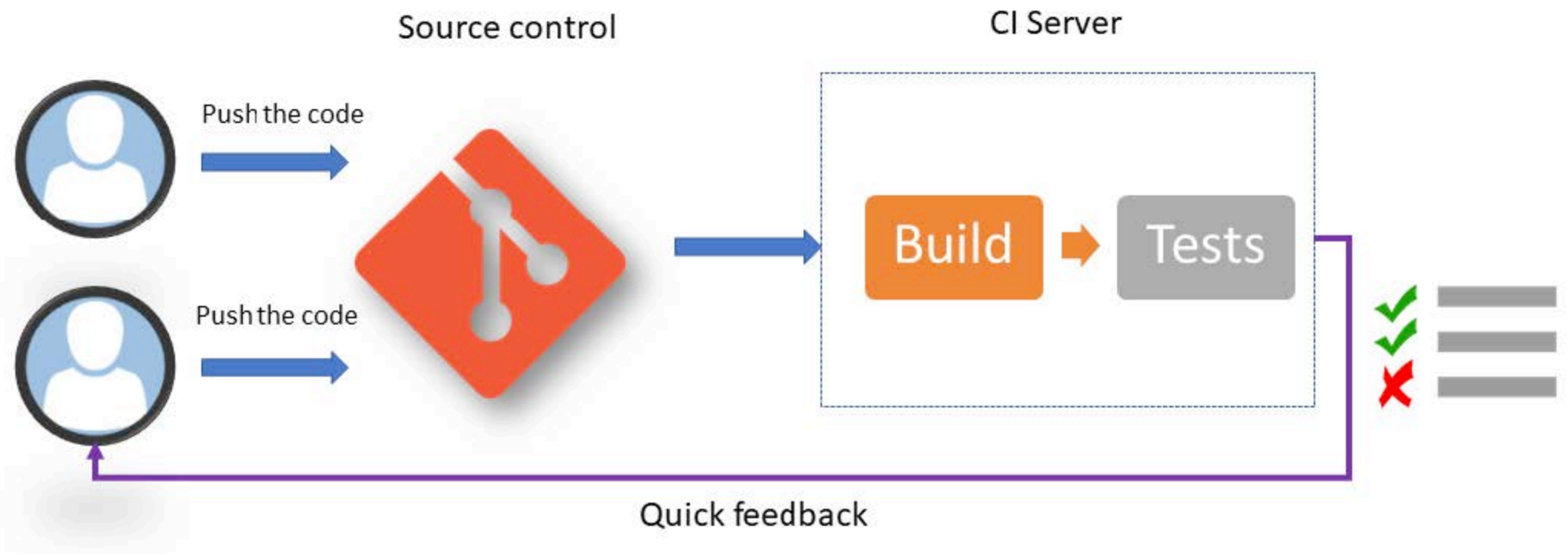


CI/CD: Continous Integration / Continuous Delivery

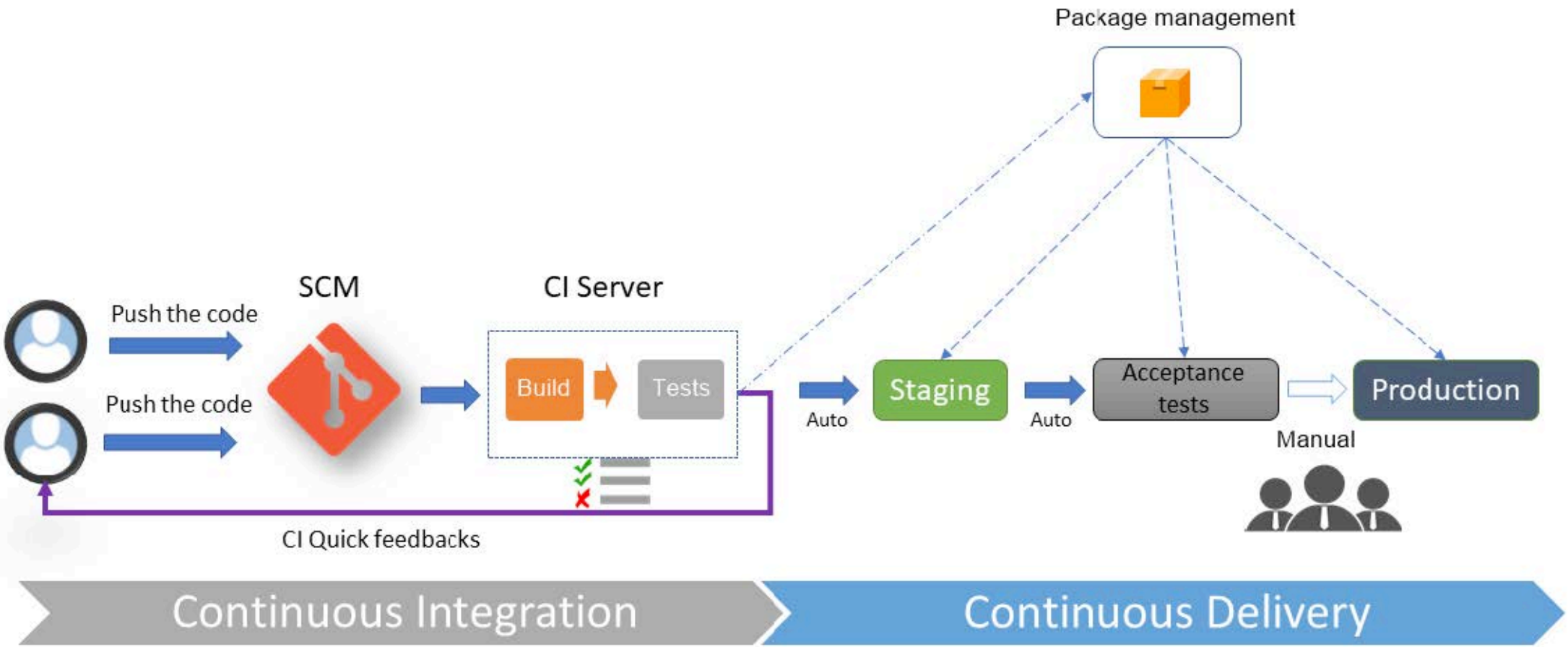
- Continuous integration (CI)
- Continuous delivery (CD)
- Continuous deployment

Cada proceso tiene su propio pipeline

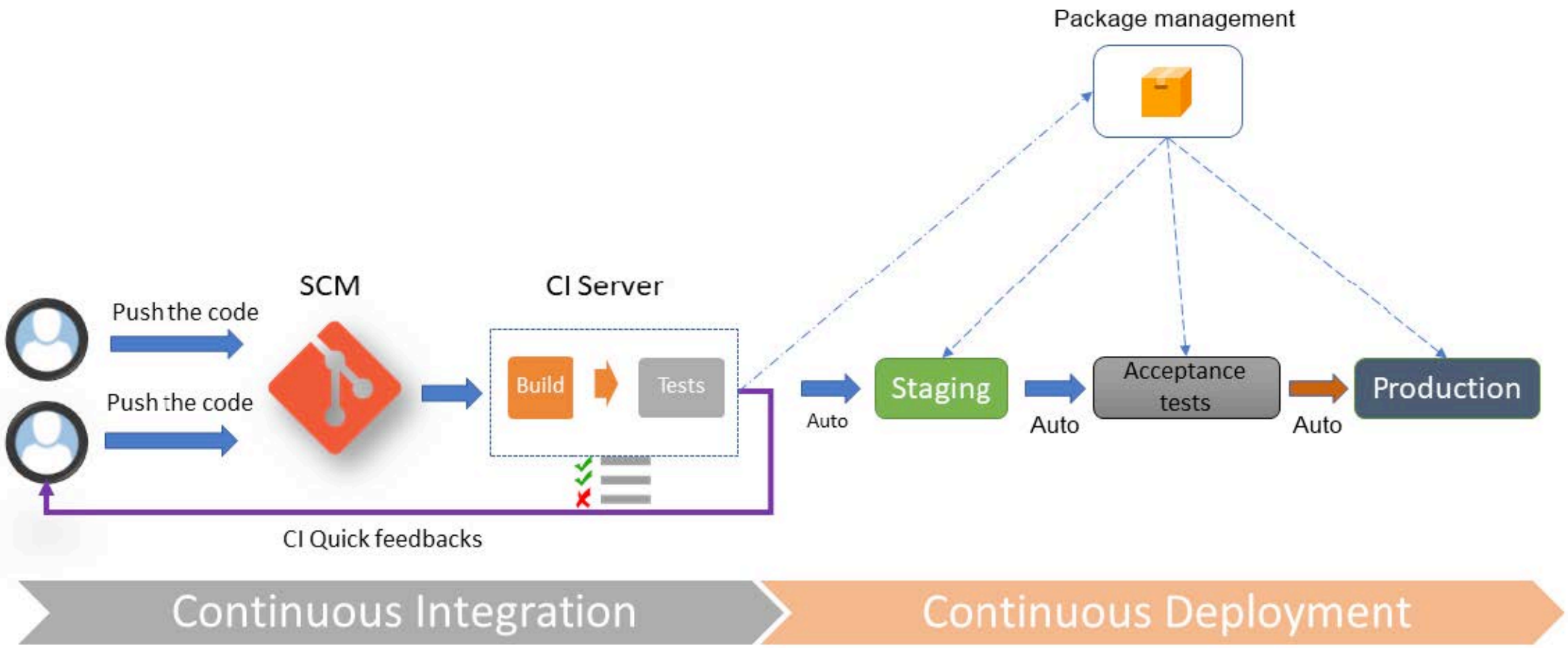
Pipeline de CI



Pipeline de CD



Continuous Deployment



Ejemplo de pipeline CI/CD

```yaml
name: CI Security Pipeline

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Run static analysis (SAST)
        uses: sonarsource/sonarqube-scan-action@v1.0

      - name: Dependency vulnerability scan
        uses: snyk/actions/python@master
        with:
          command: test
```

📙	**Definiciones**
Build	acción de compilar y ensamblar el código fuente de una aplicación en un formato ejecutable o en un conjunto de artefactos que se pueden utilizar en un entorno de ejecución específico
Pipeline	un conjunto automatizado y secuencial de procesos que permiten la ejecución de tareas específicas. Analogía de una línea de montaje de la industria de fabricación
Staging	entorno de prueba que replica el entorno de producción para realizar pruebas finales (con usuarios) antes del despliegue

🟧	**Definiciones**
Artefacto	resultado del *build*. Pueden ser binarios ejecutables, bibliotecas, paquetes de instalación, etc., necesarios para ejecutar la aplicación
Release	una versión específica y completa de una aplicación o software que se considera lista para ser distribuida y utilizada por los usuarios finales
Release Candidate (RC)	*release* con el potencial de convertirse en la versión final o lanzamiento si no se encuentran problemas significativos durante las pruebas

Infrastructure as Code (IaC)

Buenas prácticas de IaC

- Automatizar todo en el código
- SCM: *Source Control Manager*
- Guardar el código junto al de la aplicación
- Código de la IaC debe ser idempotente
- Integrar con CI/CD

Ejemplo usando terraform

- Recurso de Amazon AWS ECS (Elastic Container Service) (sección `aws_ecs_service`)
- El servicio despliega un contenedor Docker (sección `aws_ecs_task_definition`)
- El contenedor Docker ejecuta un servidor nginx en el puerto 80 (configuración de `container_definitions`)
- Solo el tráfico HTTPS está permitido
- El acceso a la instancia de ECS esté restringido solo a direcciones IP específicas (sección `network_configuration`)

```
provider "aws" {
  region = "West Europe" # Cambia esto según tu región de AWS
}

resource "aws_ecs_cluster" "example_cluster" {
  name = "example-cluster"
}

resource "aws_ecs_task_definition" "example_task" {
  family                   = "example-task"
  network_mode             = "bridge"
  requires_compatibilities = ["EC2"]
...
```

```
...
  container_definitions = <<EOF
[
  {
    "name": "example-container",
    "image": "nginx:latest",
    "portMappings": [
      {
        "containerPort": 80,
        "hostPort": 80
      }
    ]
  }
]
EOF
}
...
```

```
...
resource "aws_ecs_service" "example_service" {
  name            = "example-service"
  cluster         = aws_ecs_cluster.example_cluster.id
  task_definition = aws_ecs_task_definition.example_task.arn
  launch_type     = "EC2"
  desired_count   = 1

  network_configuration {
    # Coloca las ID de tus subredes aquí
    subnets         = ["subnet-xxxxxxxxxxxxxx", "subnet-yyyyyyyyyyyyyy"]
    # Coloca la ID de tu grupo de seguridad aquí
    security_groups = ["sg-xxxxxxxxxxxxxxxx"]
  }
}
```

Ejemplo de configuración insegura (Terraform)

```
resource "aws_s3_bucket" "data" {
  bucket = "example-bucket"
  acl    = "public-read"   # ❌ Exposición innecesaria
}
```

Configuración más segura:

```
resource "aws_s3_bucket" "data" {
  bucket = "example-bucket"
  acl    = "private"
  versioning {
    enabled = true
  }
}
```

Provisioning (aprovisionamiento)

Ejemplo con Docker

Dockerfile:

```
FROM ubuntu
RUN apt-get update
RUN apt-get install -y nginx
ENTRYPOINT ["/usr/sbin/nginx","-g","daemon off;"]
EXPOSE 80
```

Ejemplo de provisioning inseguro (Dockerfile)

```
FROM ubuntu:latest
RUN apt-get update && apt-get install -y python3
COPY . /app
CMD ["python3", "/app/server.py"]
```

Configuración más segura:

```
FROM python:3.11-slim
WORKDIR /app
COPY . /app
RUN pip install --no-cache-dir -r requirements.txt
CMD ["python", "server.py"]
```

Detectar vulnerabilidades:

```
docker scan myapp:latest
# o con Trivy:
trivy image myapp:latest
```

Gestión segura de dependencias

Riesgo:

```
pip install requests==2.18.0
```

Escaneo y actualización:

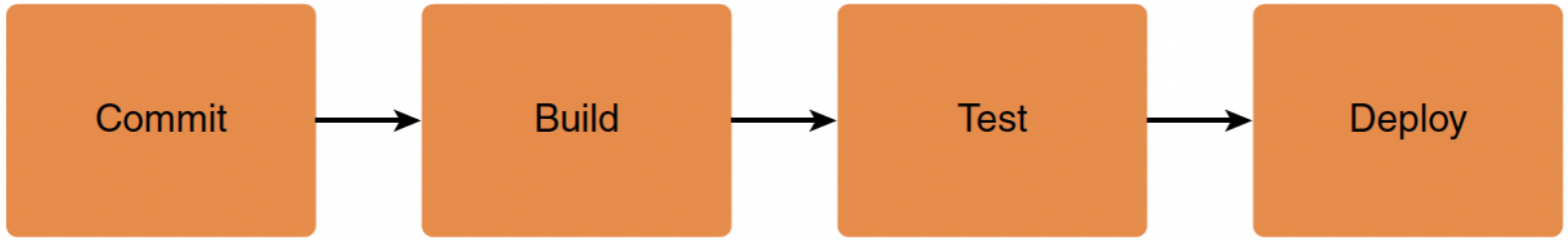
```
pip install safety
safety check
```

Salida:

```
→ requests (<=2.19.1)
   CVE-2018-18074 - Session credential leakage
   fixed in: 2.20.0
```

	Definiciones
IaC	práctica en la que la infraestructura de sistemas, redes y otros recursos tecnológicos se gestiona y aprovisiona utilizando código y archivos de configuración en lugar de realizar configuraciones manuales o a través de interfaces gráficas
Provisioning	proceso de preparar y configurar de manera automática los recursos de infraestructura necesarios para ejecutar una aplicación o servicio

CI/CD pipeline tools para DevSecOps



SAST
- SonarQube
- Veracode
- Mend
- Checkmarx
- Snyk

SCA
- GitHub
- GitLab
- Snyk
- FOSSA

IaC Scanning
- TFLint
- GitLab
- Snyk

SAST
- SonarQube
- Veracode
- Mend
- Checkmarx
- Snyk

DAST
- Checkmarx
- Rapid 7
- HCL AppScan

IAST
- Contrast Security
- Invicti
- Checkmarx
- Veracode

DAST
- Checkmarx
- Rapid 7
- HCL AppScan

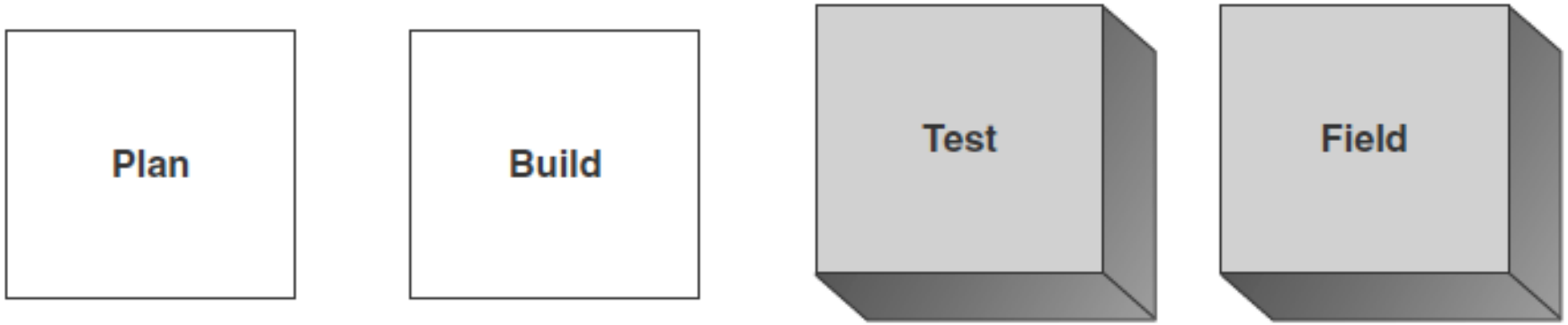
RASP
- Dynatrace
- Signal Science
- JSDefender
- Imperva
- OpenRASP
- Veracode
- Rapid 7

- SAST: Static Application Security Testing
- SCA: Software Composition Analysis (or Dependency Analysis)
- DAST: Dynamic AST
- IAST: Interactive AST
- IaC/Container Vulnerability Scanning

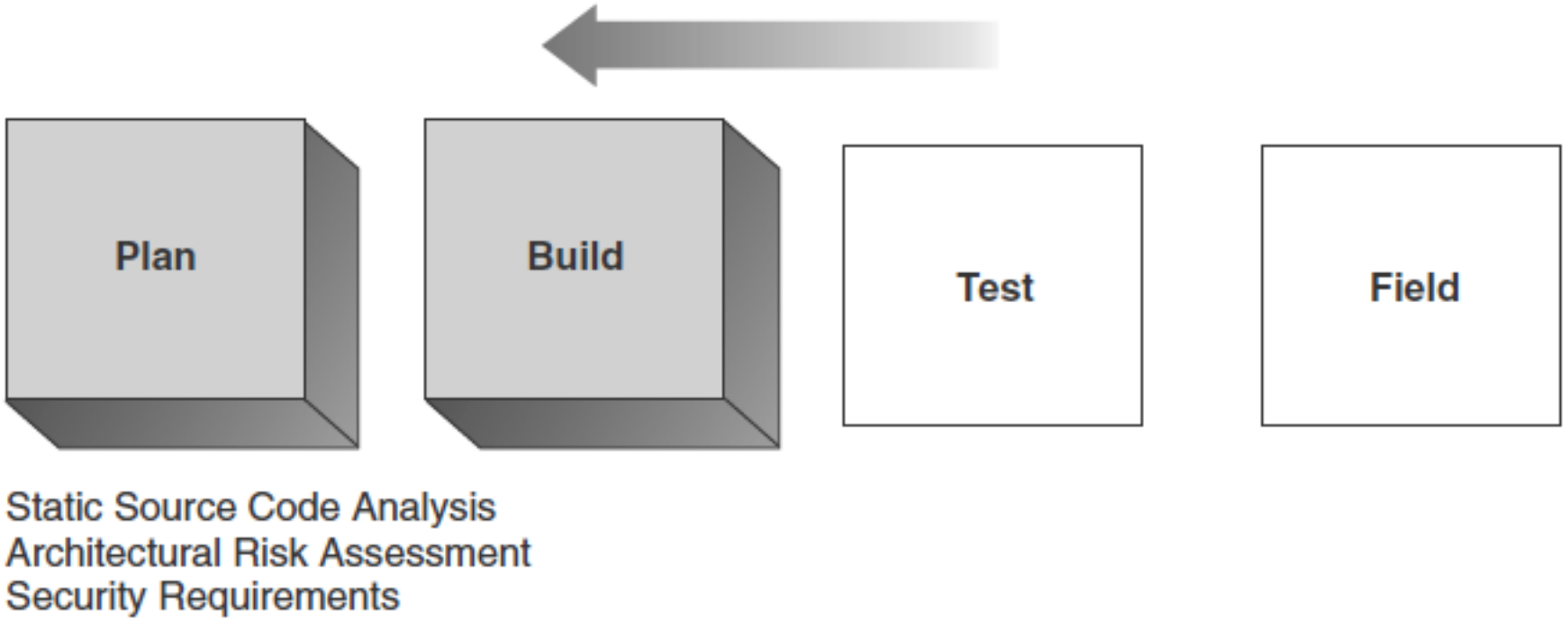
SAST

Papel de SAST en el SDLC

Tratar los síntomas: pensar en la seguridad después de que el software se ha construido no es lo más correcto



Tratar la causa: pensar pronto en la seguridad, durante las actividades dedicadas al desarrollo y construcción del software



Aquí es donde el SAST hace su función para la seguridad

¿Por qué preocuparnos tanto si la IA ya sabe programar?

Ejemplo con Copilot

1. Abrir código fuente Python con VSCode
   - `server-side-request.py`
   - `temporary-file-creation.py`
   - `cryptographic-key-generation.py`
2. Comprobar las reglas SounarSource:
   - RSPEC-5144: Server-side requests should not be vulnerable to forging attacks
   - RSPEC-5445: Insecure temporary file creation methods should not be used
   - RSPEC-4426: Cryptographic key generation should be based on strong parameters
3. Probar copilot...

Importancia del SAST

- La IA aprende a programar metiendo bugs
- El código con que se entrega a los modelos de ML para generar código es código no está libre de problemas de seguridad
- Hay que analizar todo el código generado y que el programador tenga siempre la última palabra

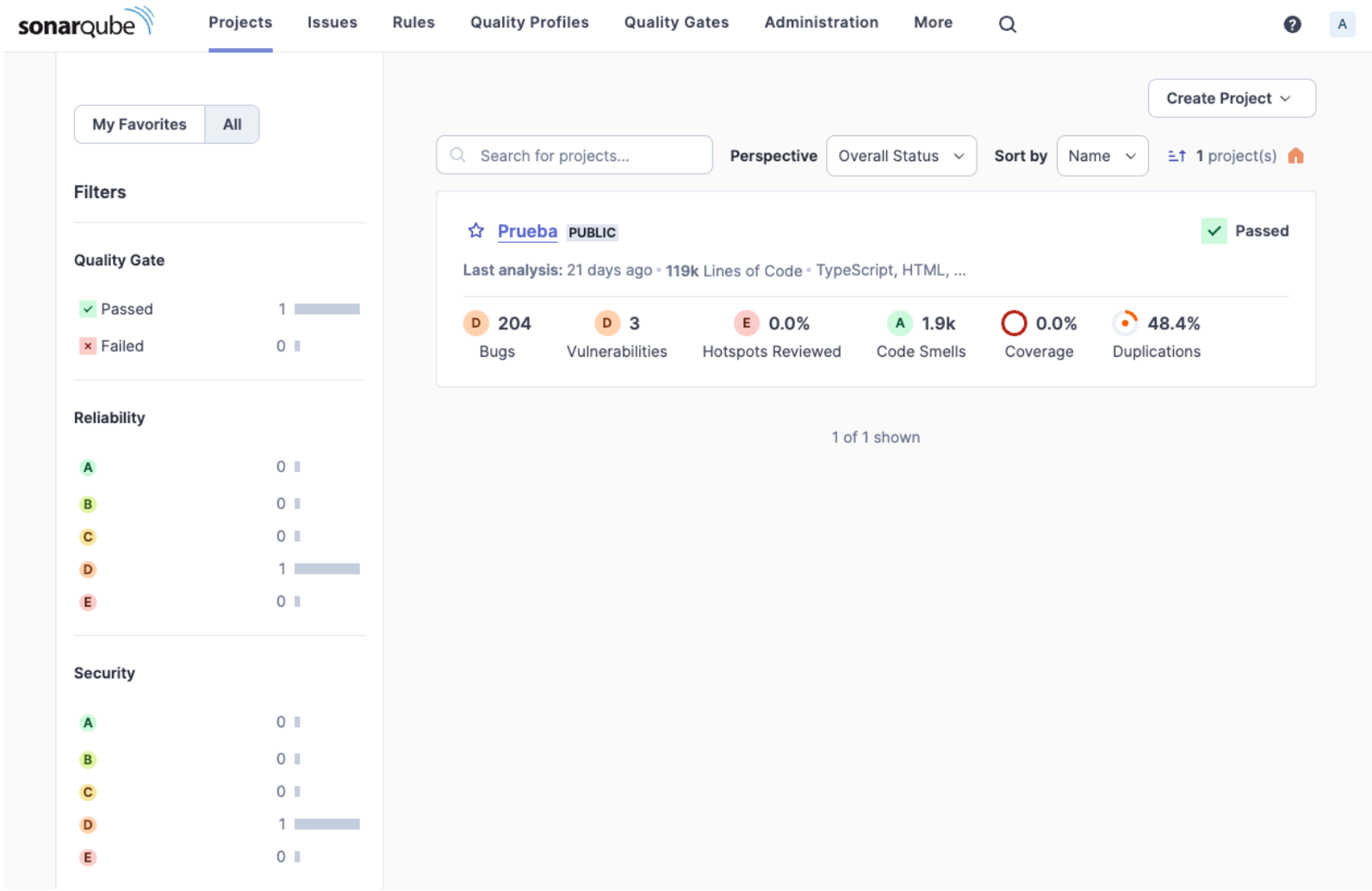
Copilot y su código inseguro o cómo la IA aprende a programar (también) metiendo Bugs

SonarQube

SonarQube

Herramienta automática de revisión de código

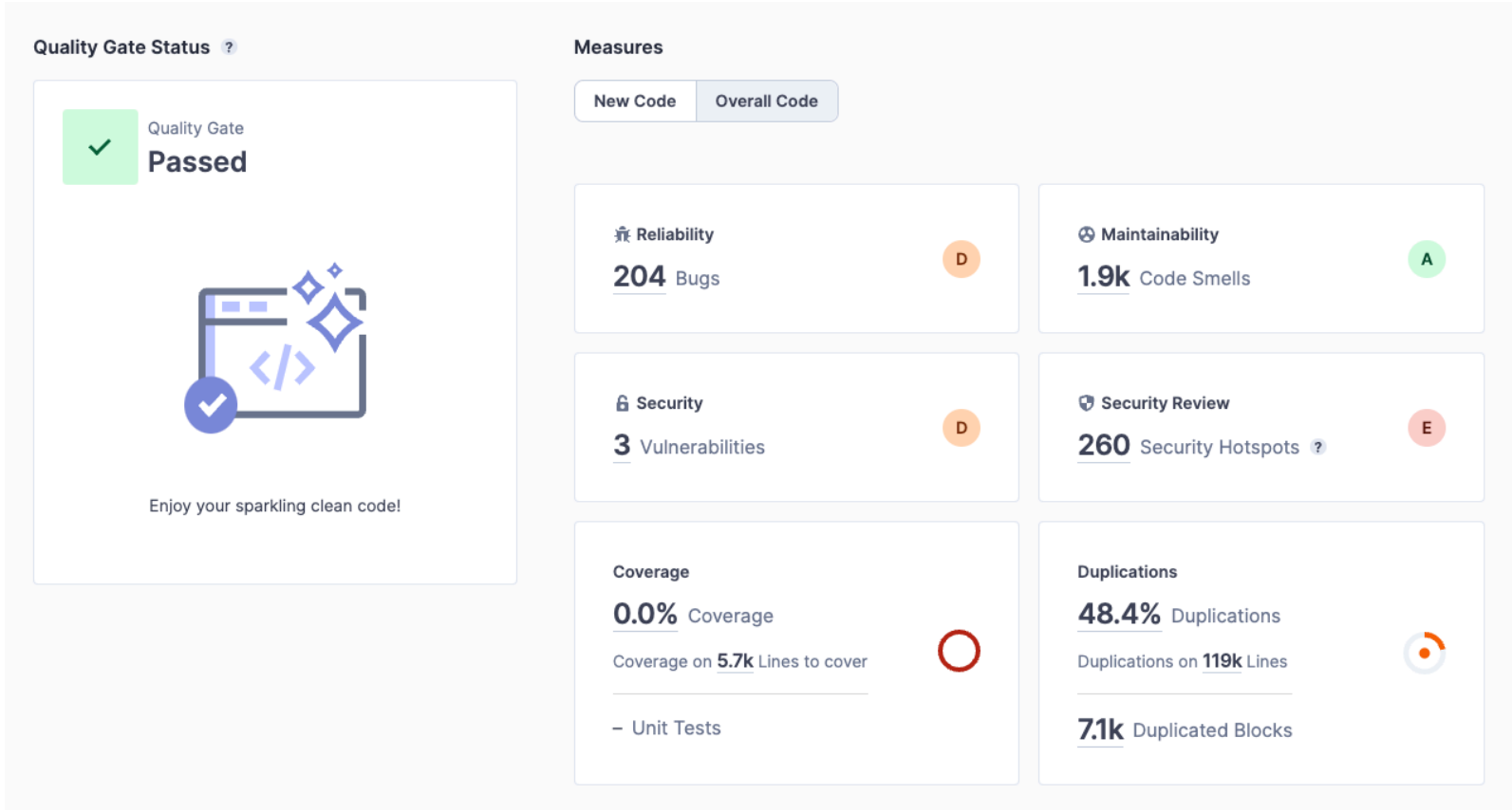
- Análisis estático de métricas de calidad
- Clean code
- Refactoring



Análisis para detectar...

- Bugs
- **Security hotspots**
- **Vulnerabilidades**
- Code smells
- Duplicación de código
- Etc.

Bug vs vulnerability vs code smell: Deprecated!



Security hotspot

Código sensible que no impacta necesariamente en la seguridad

Vulnerabilidad

Problema que impacta en la seguridad y debe ser arreglado

```
export async function run(command: string, options: RunOptions) {
    console.log(`Running ${command} in ${options.cwd}`);
    const process = spawn(command, { shell: true, cwd: options.cwd, stdio: 'inherit' });

    return new Promise<void>((resolve, reject) => {
        process.on('exit', (code) => {
            if (code !== 0) {
                reject(new Error(`Command ${command} exited with code ${code}`));
            } else {
                resolve();
            }
        });
    });
}
```

Make sure that executing this OS command is safe here.

Security hotspot

Código sensible que no impacta necesariamente en la seguridad

Vulnerabilidad

Problema que impacta en la seguridad y debe ser arreglado

```
import { loadMonaco } from "../monaco-loader";
import { IMessageFromRunner, IMessageToRunner, IPreviewState } from "../shared";
import "./style.scss";

window.addEventListener("message", (event) => {
```

Verify the origin of the received message.

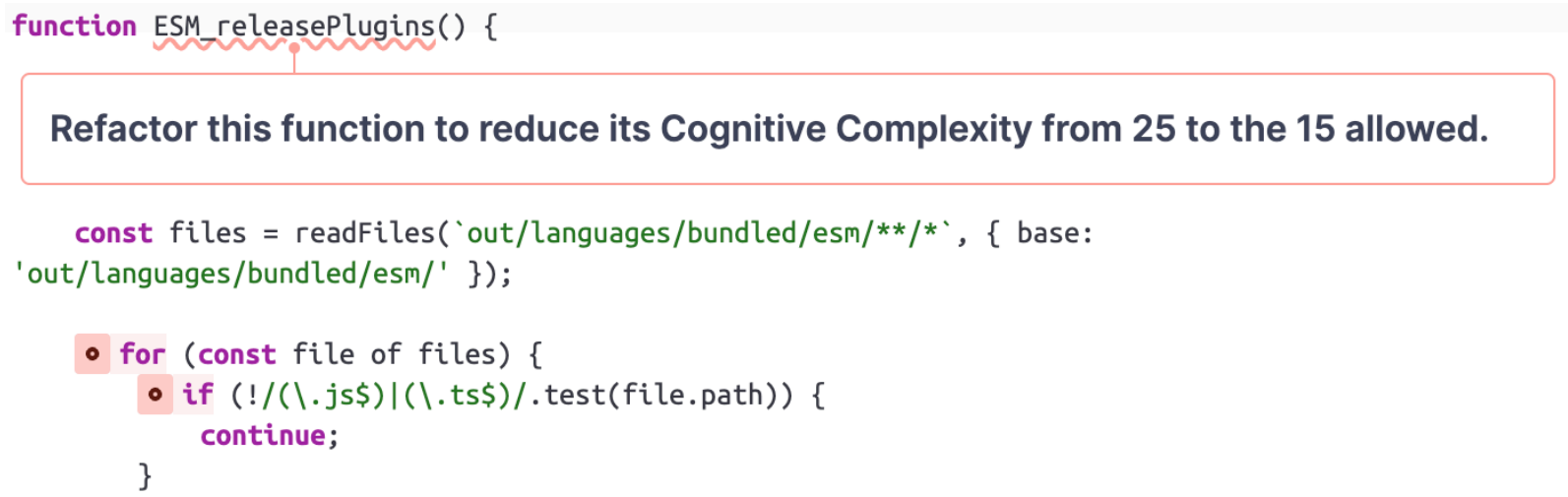
```
    const isInSandbox = window.origin === "null";
    if (!isInSandbox) {

// To prevent someone from using this html file to run arbitrary code in non-sandboxed context
        console.error("not in sandbox");
```

Clean code

SonarQube clean code

- Clean code attributes
- Software qualities
- Issues



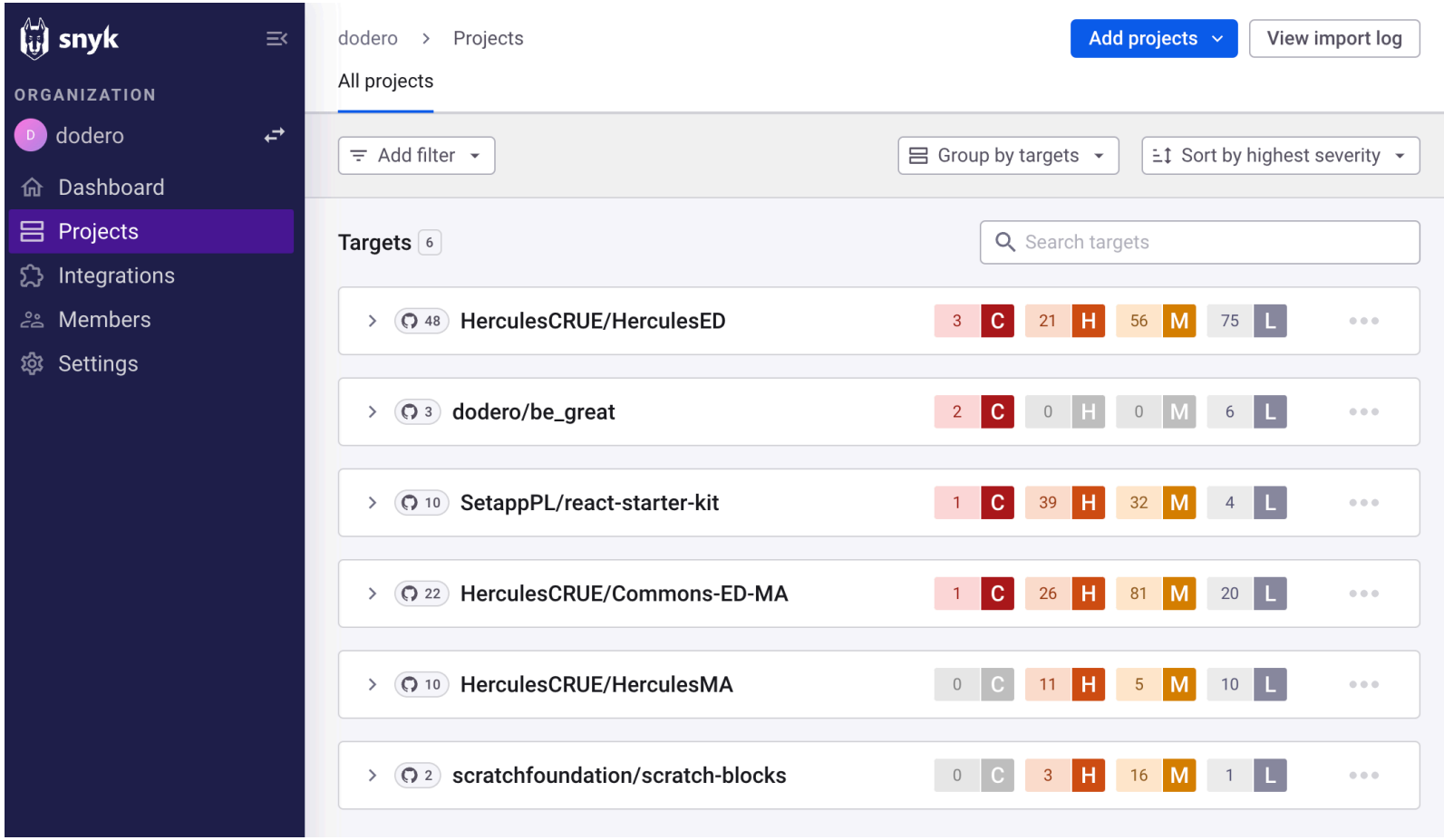
SCA

Software Composition Analysis (SCA)

- Proceso automático que identifica y analiza las dependencias de un proyecto
- Busca y repara vulnerabilidades existentes en las dependencias (componentes) de un proyecto
- Monitoriza continuamente las dependencias y permite corregirlas rápidamente cuando se descubren nuevas vulnerabilidades

Snyk.io

Snyk.io

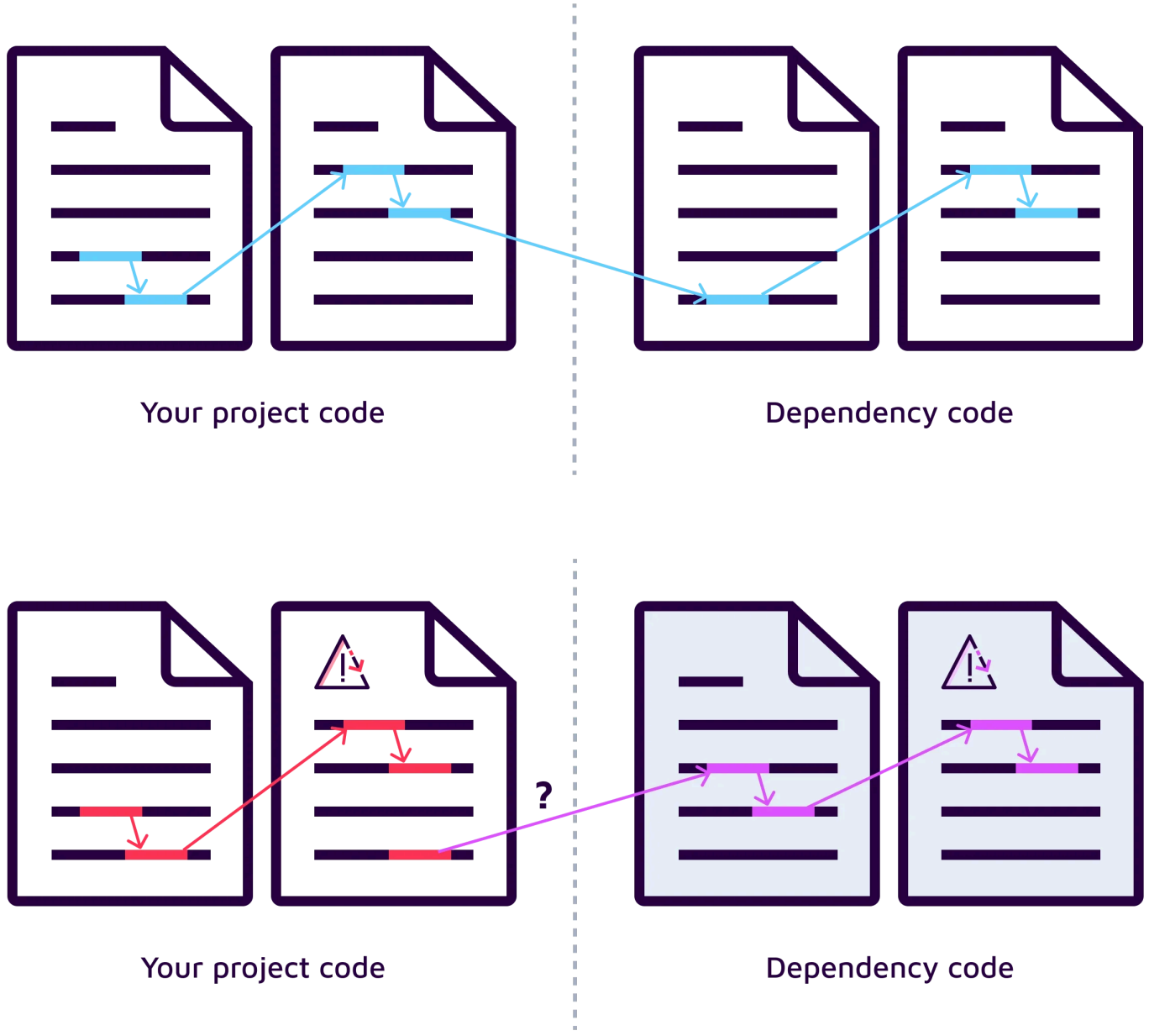

snyk
ORGANIZATION
dodero
Dashboard
Projects
Integrations
Members
Settings
dodero > Projects
All projects
Add projects
View import log
Add filter
Group by targets
Sort by highest severity
Targets 6
Search targets
48 HerculesCRUE/HerculesED 3 C 21 H 56 M 75 L
3 dodero/be_great 2 C 0 H 0 M 6 L
10 SetappPL/react-starter-kit 1 C 39 H 32 M 4 L
22 HerculesCRUE/Commons-ED-MA 1 C 26 H 81 M 20 L
10 HerculesCRUE/HerculesMA 0 C 11 H 5 M 10 L
2 scratchfoundation/scratch-blocks 0 C 3 H 16 M 1 L

Deep SAST

Deep SAST

- Vulnerabilidades *taint* (contaminación)
- Frameworks y bibliotecas
- Dependencias transitivas
- SCA no es suficiente: solo vulnerabilidades en BD públicas

Hidden Security Vulnerabilities



Deep SAST issue 1

SQL injection

src/main/java/com/dashboardmanager/controller/UserController.java

Show 110 more lines

```
111 …               return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
112             }
113         }
114
115 …       @PostMapping("/user/migrate")
116         public ResponseEntity<Resource> migrateUser( 1 @RequestParam(value = "username") String  2 username
    ) {
117             try (var connection = getConnection()) {
118              3 doMigrateUser(username, connection);
119                 return ResponseEntity.status(HttpStatus.OK).build();
120             } catch (SQLException e) {
121                 return ResponseEntity.status(HttpStatus.BAD_REQUEST).build();
122             }
123         }
124
125         private void  4 doMigrateUser( 5 String username, ConnectionImpl connection) throws SQLException {
126             // migration might fail midway, set a savepoint to keep the database consistent
127             Savepoint savepoint =  8 connection.setSavepoint( 7 "before-" +  6 username + "-migration");
```

Change this code to not construct SQL queries directly from user-controlled data.

```
128
129             User newUser = new User();
130             newUser.setName("_placeholder_" + username);
131             newUser.setPassword("_placeholder_");
132             usersRepository.save(newUser);
133
134             User user = usersRepository.findByName(username);
135             if (user == null) {
136                 connection.rollback(savepoint);
```

Show 58 more lines

1 **SOURCE** a user can craft an HTTP request with malicious content

2 This instruction can propagate malicious content

3 This instruction can propagate malicious content

4 This instruction can propagate malicious content

5 This instruction can propagate malicious content

6 The malicious content is concatenated into the string

7 This concatenation can propagate malicious content to the newly created string

8 **SINK** sink (in dependency): tainted value is used to perform a security-sensitive operation

Deep SAST issue 2

Deserialization

src/main/java/com/dashboardmanager/controller/UserController.java

Show 116 more lines

```
117  …          }
118             return new String(org.apache.commons.codec.binary.Base64.encodeBase64(bos.toByteArray()));
119         }
120
121         private SessionHeader getSessionHeader(HttpServletRequest request) {
122          2 String sessionAuth =  1 request.getHeader("Session-Auth");
123             if (sessionAuth != null) {
124                 try {
125                  4 byte[] decoded =  3 decodeBase64(sessionAuth);
126                  7 ObjectInputStream in =  6 new ObjectInputStream( 5 new ByteArrayInputStream(decoded));
127                    return (SessionHeader)  8 in.readObject();
```

Change this code to not deserialize user-controlled data.

```
128                 } catch (Exception e) {
129                     return null;
130                 }
131             }
132             return null;
133         }
134
135         private boolean isAuthenticated(HttpServletRequest request) {
136             SessionHeader sessionHeader = getSessionHeader(request);
```

Show 13 more lines

1 **SOURCE** a user can craft an HTTP request with malicious content

2 A tainted value can be assigned to variable 'sessionAuth'

3 tainted value is propagated (through dependency)

4 A malicious value can be assigned to variable 'decoded'

5 This constructor can propagate malicious content to the newly created object

6 This constructor can propagate malicious content to the newly created object

7 A malicious value can be assigned to variable 'in'

8 **SINK** this invocation is not safe; a malicious value can be injected into