



DevSecOps \$in un mango

Marcos García & Nico Mayer

Agenda

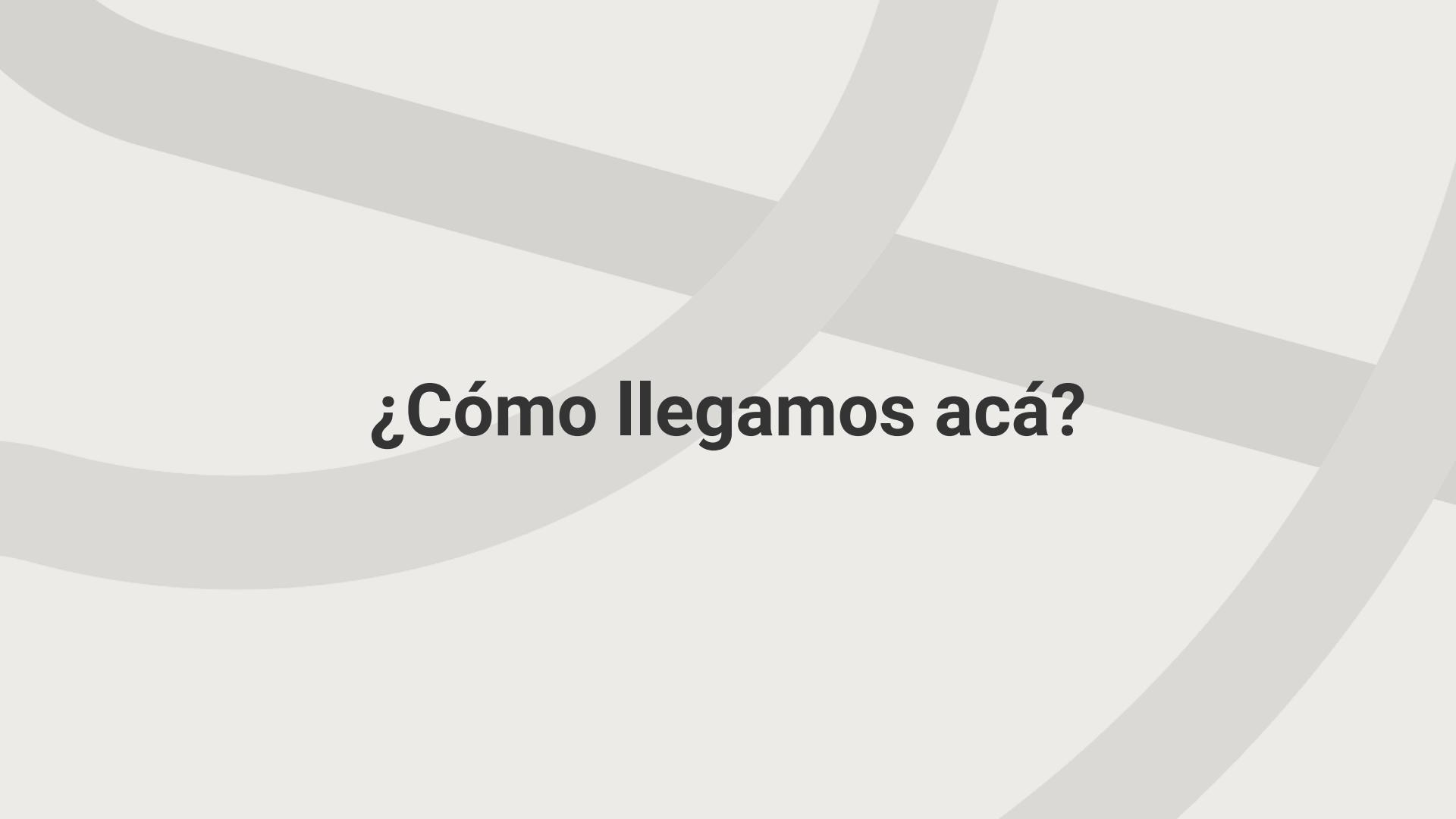
Temas

1. Evolución de Desarrollo de Software
2. OWASP Top 10 Web
3. Workshop sobre DevSecOps
4. Laboratorios
 - o Docker
 - o Jenkins
 - o DVWA
 - o SonarQube
 - o OWASP ZAP
 - o DefectDojo
 - o Jenkins + Zap + DefectDojo
 - o Jenkins + SonarQube +
DefectDojo
 - o Pipeline
5. Reflexiones y lecciones aprendidas
6. Referencias

Referencias

Como usaremos los colores durante el Workshop:

```
nombre-de-archivo.yml  
variable: contenido-del-archivo  
  
# comando-id up -X
```

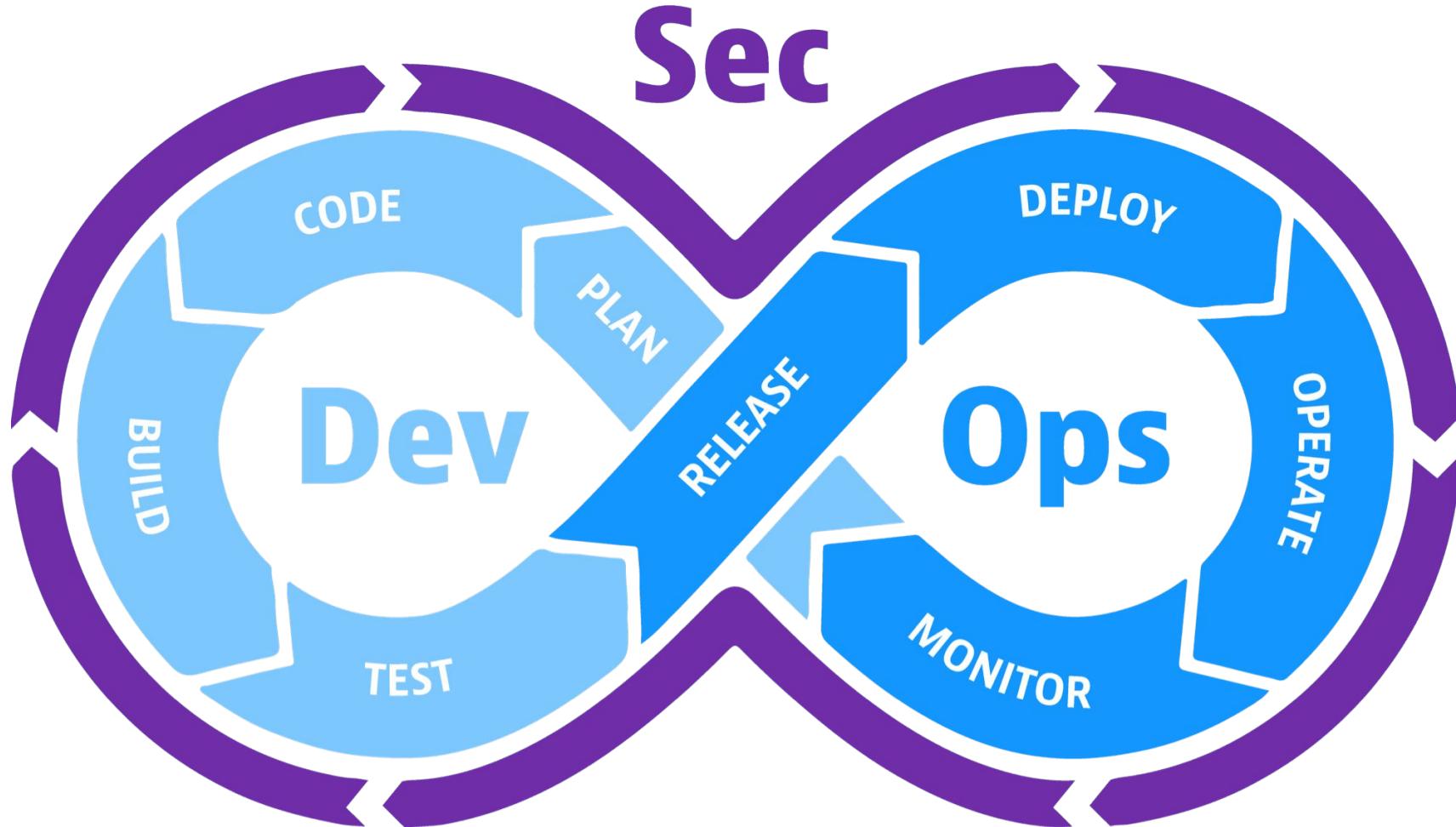


¿Cómo llegamos acá?



Evolución de Desarrollo de Software







TOP10



OWASP TOP TEN

2017 OWASP TOP 10 LIST

The 2017 OWASP Top 10 features new and classic risks
that should be top-of-mind for every organization

A1 Injection

A2 Broken Authentication

A3 Sensitive Data Exposure

A4 XML External Entities (XXE)

A5 Broken Access Control

A6 Security Misconfiguration

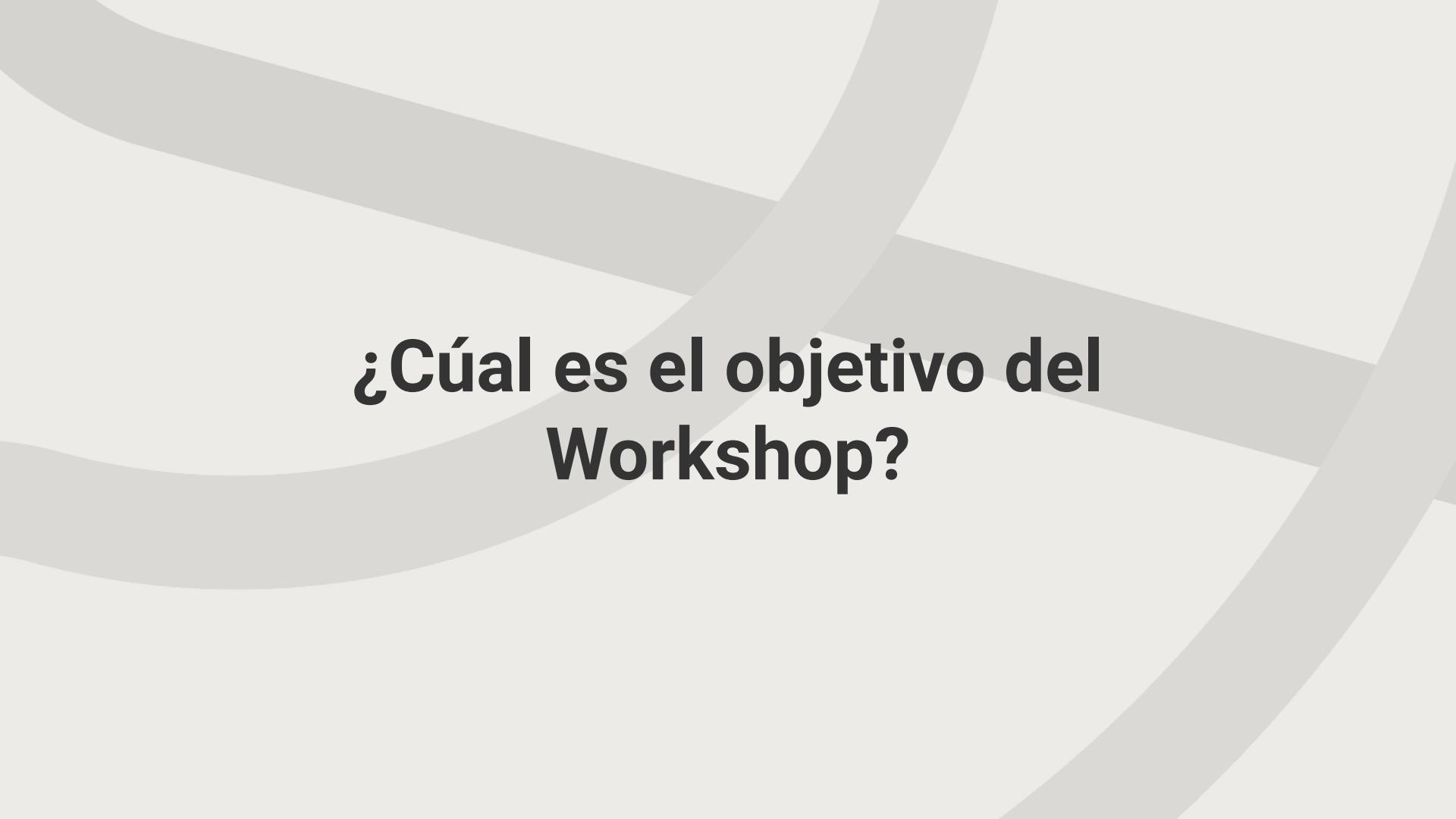
A7 Cross-Site Scripting (XSS)

A8 Insecure Deserialization

A9 Using Components With Known Vulnerabilities

A10 Insufficient Logging & Monitoring

Workshop sobre DevSecOps



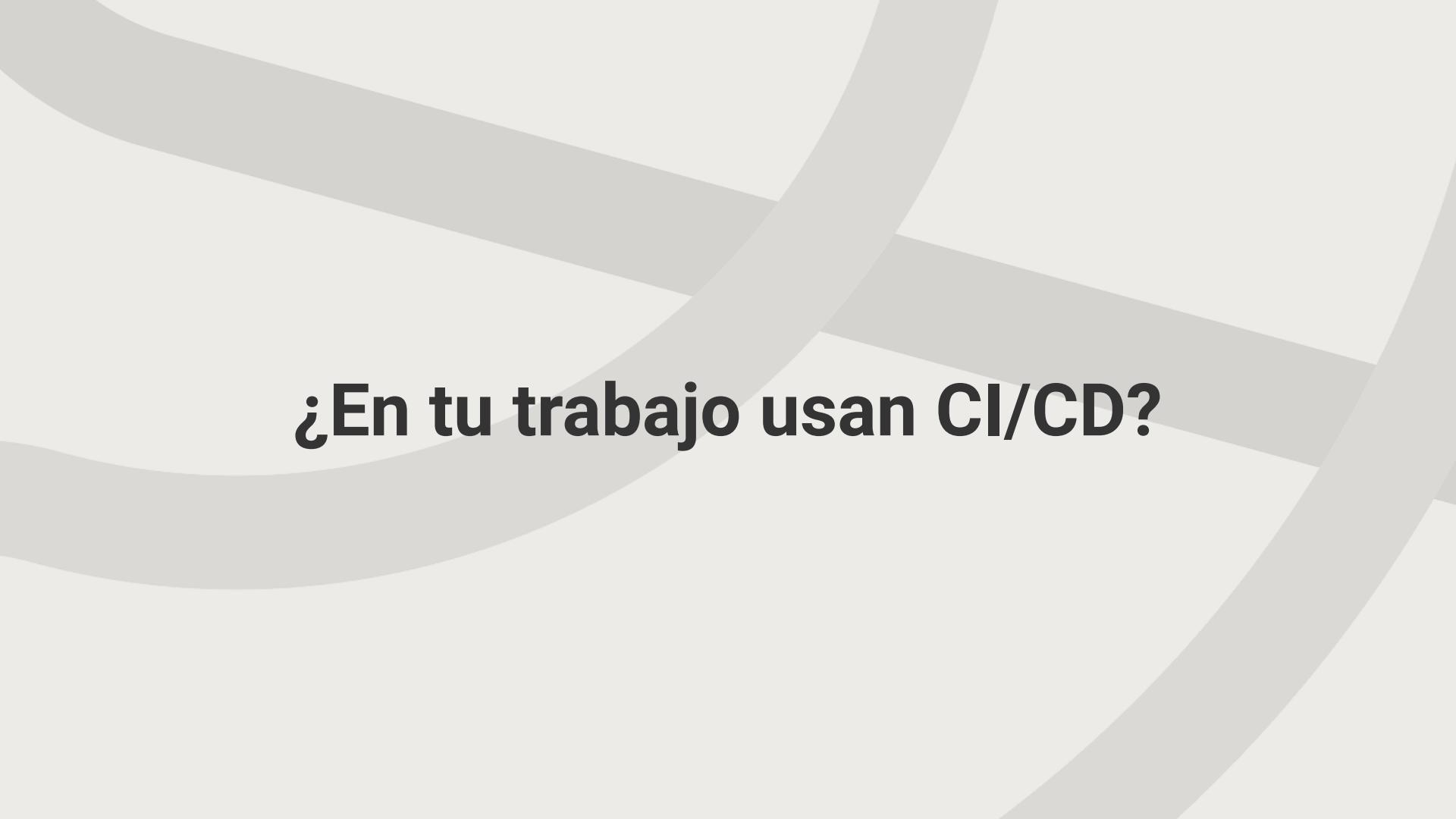
**¿Cúal es el objetivo del
Workshop?**

**¿En tu trabajo existe un pipeline
con DAST/SAST?**

<https://www.menti.com/9rt4dk6cjv>

8141 6420





¿En tu trabajo usan CI/CD?

<https://www.menti.com/9rt4dk6cjv>

8141 6420



**¿Por qué creemos que es
importante tener un pipeline de
seguridad?**

Laboratorio



docker

Docker

Docker es una plataforma de software que permite crear, probar e implementar aplicaciones rápidamente. Docker empaqueta software en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y tiempo de ejecución.

(De Amazon)

Docker

Instalación y puesta en marcha de docker

```
# sudo apt-get remove docker docker-engine docker.io containerd runc
# sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
# sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
# sudo apt-get install docker-ce docker-ce-cli containerd.io
# sudo apt-get update
# sudo apt install docker-ce
# sudo usermod -aG docker $USER
# sudo chmod 666 /var/run/docker.sock
# /etc/init.d/docker restart
# sudo apt install docker-compose
```



Jenkins

Jenkins

Jenkins es un servidor automatizado de integración continua de código abierto capaz de organizar una cadena de acciones que ayudan a lograr el proceso de integración continua (y mucho más) de manera automatizada.

(De Ciberninjas)

Puesta en marcha de Jenkins

Instalando Jenkins en nuestro servidor

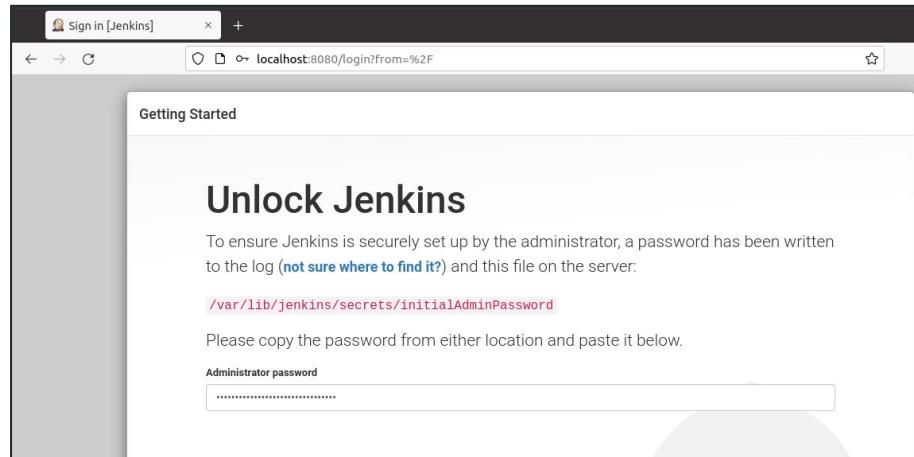
```
# sudo apt install openjdk-11-jdk  
  
# wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -  
  
# sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/' > /etc/apt/sources.list.d/jenkins.list  
  
# sudo apt-get update  
  
# sudo apt-get install jenkins
```

Puesta en marcha de Jenkins

Instalando Jenkins en nuestro servidor docker host

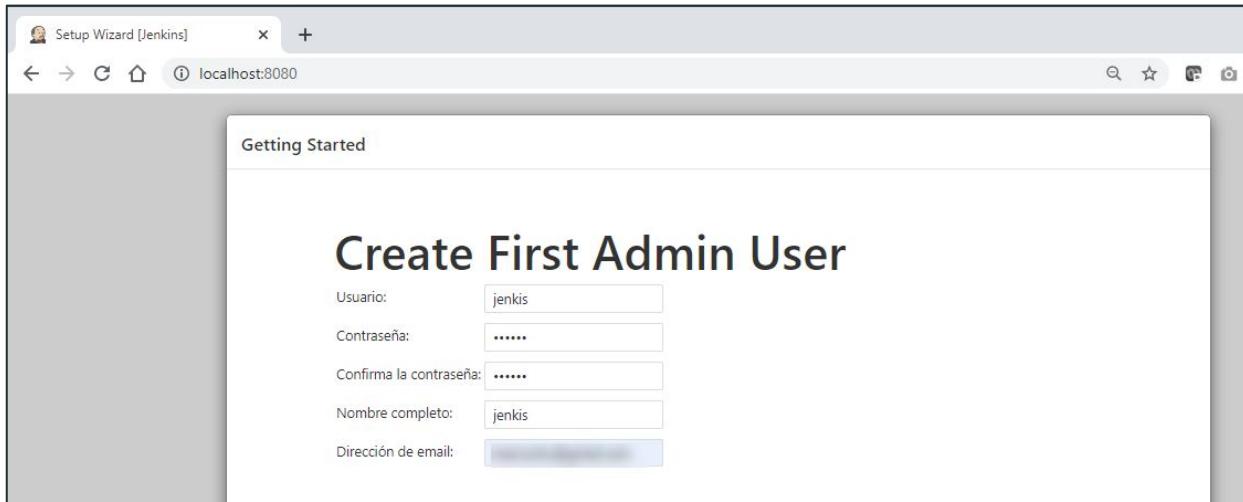
Ingresar a la URL <http://localhost:8080/>

```
# sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```



Puesta en marcha de Jenkins

Ingresamos al sitio Web **http://localhost:8080** y seguimos todos los pasos hasta la creación del usuario.



The screenshot shows a web browser window titled "Setup Wizard [Jenkins]". The address bar displays "localhost:8080". The main content area is titled "Getting Started" and features a large heading "Create First Admin User". Below the heading are five input fields:

Usuario:	<input type="text" value="jenkis"/>
Contraseña:	<input type="password" value="....."/>
Confirma la contraseña:	<input type="password" value="....."/>
Nombre completo:	<input type="text" value="jenkis"/>
Dirección de email:	<input type="text" value="██████████"/>

Puesta en marcha de Jenkins

Instalamos el plugin **HTML Publisher**.

Administrar Jenkins -> Administrar Plugins -> Todos los plugins:

- HTML Publisher
- Git Plugin
- Git Client Plugin
- GitHub Authentication plugin
- SonarQube Scanner for jenkins

The screenshot shows the Jenkins Plugin Manager interface. The search bar at the top contains the text 'git'. Below the search bar, there are tabs for 'Updates', 'Available', 'Installed' (which is selected), and 'Advanced'. A search result for 'git' is displayed, showing the following table:

Enabled	Name	Version	Previously installed version	Uninstall
<input checked="" type="checkbox"/>	Apache HttpComponents Client 4.x API Plugin	4.5.13-1.0		<button>Uninstall</button>
<input checked="" type="checkbox"/>	bouncycastle API Plugin	2.25		<button>Uninstall</button>
<input checked="" type="checkbox"/>	Caffeine API Plugin	2.9.2-29.v717aac953f3		<button>Uninstall</button>
<input checked="" type="checkbox"/>	Command Agent Launcher Plugin	1.6		<button>Uninstall</button>
<input checked="" type="checkbox"/>	Credentials Binding Plugin	1.27		<button>Uninstall</button>



DVWA

Damn **V**ulnerable **W**eb **A**pp es una aplicación de entrenamiento en seguridad Web que se destaca por ser liviano y contener muchas aplicaciones Webs vulnerables a diferentes tipos de técnicas.

(De DVWA)

Puesta en marcha de DVWA

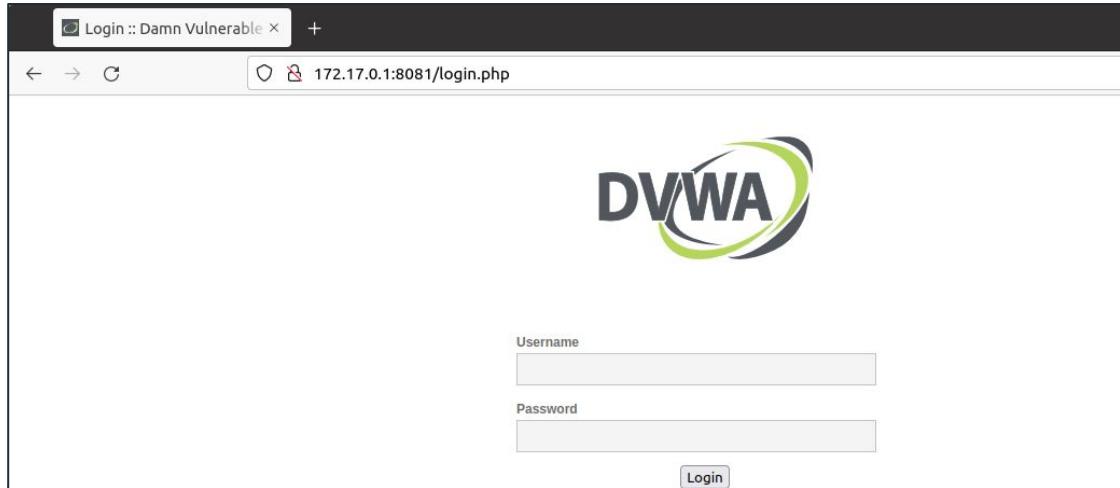
Para iniciar e instalar DVWA debemos ejecutar el siguiente comando

```
# docker run -d --name dvwa -p 8081:80 vulnerables/web-dvwa
```

Luego ingresamos a <http://localhost:8081> para su configuración

USER: admin

PASS: password



Puesta en marcha de DVWA

El siguiente paso es *resetear* la base de datos, para esto debemos hacer clic en “**Create / Reset Database**”.

Web Server SERVER_NAME: localhost
PHP function display_errors: Disabled
PHP function safe_mode: Disabled
PHP function allow_url_include: **Disabled**
PHP function allow_url_fopen: Enabled
PHP function magic_quotes_gpc: Disabled
PHP module gd: Installed
PHP module mysql: Installed
PHP module pdo_mysql: Installed
MySQL username: app
MySQL password: *****
MySQL database: dvwa
MySQL host: 127.0.0.1
reCAPTCHA key: **Missing**
[User: www-data] Writable folder /var/www/html/hackable/uploads/: Yes
[User: www-data] Writable file /var/www/html/external/phpids/0.6/lib/DS/tmp/phpids_log.txt: Yes

[User: www-data] Writable folder /var/www/html/config: Yes
Status in red, indicate there will be an issue when trying to complete some modules.
If you see disabled on either allow_url_fopen or allow_url_include, set the following in your php.ini file and restart Apache.
`allow_url_fopen = On
allow_url_include = On`
These are only required for the file inclusion labs so unless you want to play with those, you can ignore them.

Create / Reset Database

First time using DVWA.
Need to run 'setup.php'.

sonarqube



SonarQube

Es una plataforma para evaluar la calidad del código fuente, realizando un **análisis estático** (SAST) sobre dicho código, con el objetivo de advertirnos sobre diferentes puntos a mejorar y obtener métricas que nos ayudan a mejorar nuestro código.

(De SonarQube)

**3400+ Static Analysis Rules
across 27 programming languages**

	Secrets
	ABAP
	Apex
	C
	C++
	COBOL
	C#
	CSS
	Flex
	Go
	HTML
	Java
	JavaScript
	Kotlin
	Objective C
	PHP
	PL/I
	PL/SQL
	Python
	RPG
	Ruby
	Scala
	Swift
	TypeScript
	T-SQL
	VB.NET
	VB6
	XML

Puesta en marcha de SonarQube

Descargamos el docker compose

```
# curl -sSL  
https://raw.githubusercontent.com/bitnami/bitnami-docker-sonarqube/master/docker-c  
ompose.yml > docker-compose.yml
```

Modificamos el puerto a exponer por el 9000

```
# vim docker-compose.yml
```

```
12  sonarqube:  
13    image: docker.io/bitnami/sonarqube:9  
14    ports:  
15      - '9000:9000'  
16    volumes:  
17      - 'sonarqube_data:/bitnami/sonarqube'  
18
```

Puesta en marcha de SonarQube

Iniciamos el docker compose

Recomendamos agregar el siguiente parámetro dentro del archivo

```
docker-compose.yml
```

```
restart: always
```

```
# sysctl -w vm.max_map_count=262144
```

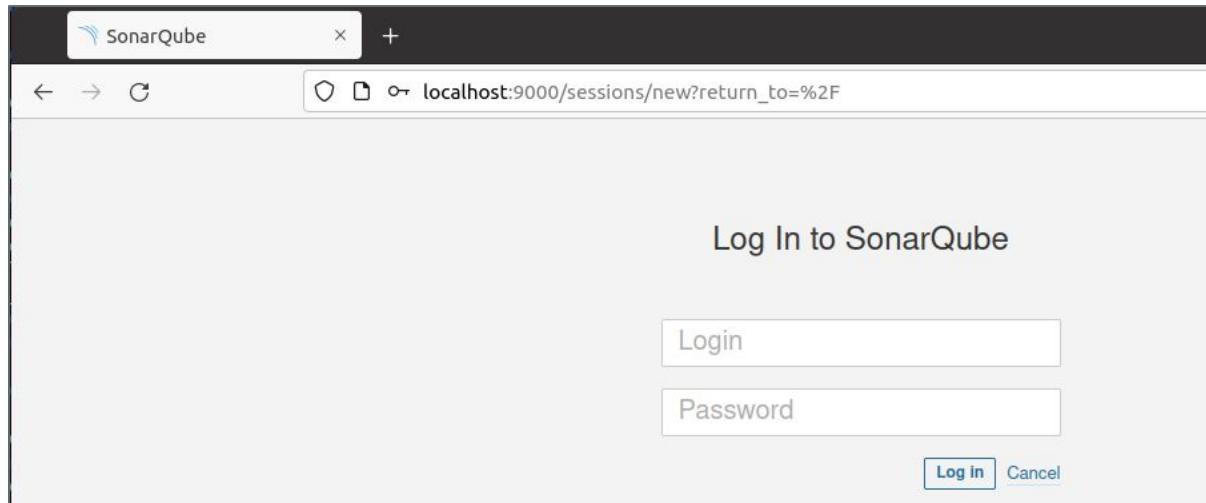
```
# sysctl -p
```

```
# docker-compose up -d
```

Puesta en marcha de SonarQube

Luego ingresar al sitio web <http://127.0.0.1:9000>

- Username: admin
- Password: bitnami



Puesta en marcha de SonarQube

Instalar sonar-scanner

```
# wget  
https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cl  
i-4.6.2.2472-linux.zip  
# unzip sonar-scanner-cli-4.6.2.2472-linux.zip  
# echo 'export PATH="/demos/sonar/sonar-scanner-4.6.2.2472-linux/bin:$PATH"' >>  
~/.bashrc  
# source ~/.bashrc
```

Puesta en marcha de SonarQube

Cómo configurar la API Key de SonarQube

- **Account -> My account-> Security**

The screenshot shows the SonarQube web interface at the URL `localhost:9000/account/security/`. The top navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and a search bar. Below the navigation, there's a user profile section with a green square icon labeled 'A' and the text 'Administrator'. A red box highlights this area. To the right, there are tabs for Profile, **Security** (which is underlined), Notifications, and Projects. Another red box highlights the 'Security' tab. The main content area is titled 'Tokens' and contains a descriptive paragraph about using tokens for security. Below this is a 'Generate Tokens' section with a form to 'Enter Token Name' and a 'Generate' button. A table lists existing tokens, with one row highlighted by a red box. The table columns are Name, Last use, and Created. The listed token is 'demos', with '2 days ago' in the 'Last use' column and 'October 15, 2021' in the 'Created' column. A 'Revoke' button is located next to the 'Created' column for this token.

Name	Last use	Created
demos	2 days ago	October 15, 2021



OWASP ZAP

OWASP ZAP

Es un escáner de seguridad web de código abierto. Pretende ser utilizado como una aplicación de seguridad y como una herramienta profesional para pruebas de penetración.

(De Wikipedia)

Puesta en marcha de OWASP ZAP

Descargamos la imagen

```
# docker pull owasp/zap2docker-stable
```



DefectDojo

Es una aplicación que se encarga de centralizar vulnerabilidades identificadas por diferentes herramientas.

Agiliza el proceso de la gestión de vulnerabilidades ya que nos permite:

- Variedad de inputs
- Identificar vulnerabilidades duplicadas
- Permite realizar metricas
- Histórico de hallazgos por proyecto
- Integración con Jira

(De DefectDojo)

Puesta en marcha de DefectDojo

Descargamos o clonamos el repositorio

```
# git clone https://github.com/DefectDojo/django-DefectDojo  
# cd django-DefectDojo
```

Recomendamos modificar el puerto 8080 por 8082 como también agregar el siguiente parámetro dentro del archivo `docker-compose.yml`

```
restart: always
```

Ejecutamos docker compose

```
# docker-compose build  
# docker-compose up -d
```

```
version: '3.7'  
services:  
  nginx:  
    build:  
      context: ./  
      dockerfile: Dockerfile.nginx  
      image: "defectdojo/defectdojo-nginx:${NGINX_VERSION:-latest}"  
      depends_on:  
        - uwsgi  
      environment:  
        NGINX_METRICS_ENABLED: "${NGINX_METRICS_ENABLED:-false}"  
      volumes:  
        - defectdojo_media:/usr/share/nginx/html/media  
      restart: always  
    ports:  
      - target: 8080  
        published: ${DD_PORT:-8082}  
        protocol: tcp  
        mode: host  
      - target: 8443  
        published: ${DD_TLS_PORT:-8443}  
        protocol: tcp  
        mode: host  
    uWSGI:  
      build:  
        context: ./  
        dockerfile: Dockerfile.django  
        image: "defectdojo/defectdojo-django:${DJANGO_VERSION:-latest}"  
      depends_on:  
        - mysql  
      environment:  
        DJANGO_DEBUG: "False"  
        DD_DJANGO_METRICS_ENABLED: "${DD_DJANGO_METRICS_ENABLED:-false}"  
        DD_ALLURED_ROOT: "${DD_ALLURED_ROOT:-$DD_ALLURED_ROOT}"  
        DD_DATABASE_URL: "${DD_DATABASE_URL:-mysql://defectdojo:password@mysql:3306/defectdojo}"  
        DD_CELERY_BROKER_USER: "${DD_CELERY_BROKER_USER:-guest}"  
        DD_CELERY_BROKER_PASSWORD: "${DD_CELERY_BROKER_PASSWORD:-$DD_CELERY_BROKER_USER:-guest}"  
        DD_SECRET_KEY: "${DD_SECRET_KEY:-hhZcp0z8z1ngNEDwB1R0Mt-WzsY*iq}"  
        DD_CREDENTIAL_AES_256_KEY: "${DD_CREDENTIAL_AES_256_KEY:-$91a+aglqes+c8Dj+2wbAbsUzfR+4nLw}"  
      restart: always  
    volumes:  
      - type: bind  
        source: ./docker/extr_settings  
        target: /app/docker/extr_settings  
      - "defectdojo_media:${DD_MEDIA_ROOT:-/app/media}"
```

Puesta en marcha de DefectDojo

Identificamos la clave de inicialización ejecutando el siguiente comando en la misma carpeta del proyecto, debemos esperar 5 minutos luego del comando anterior.

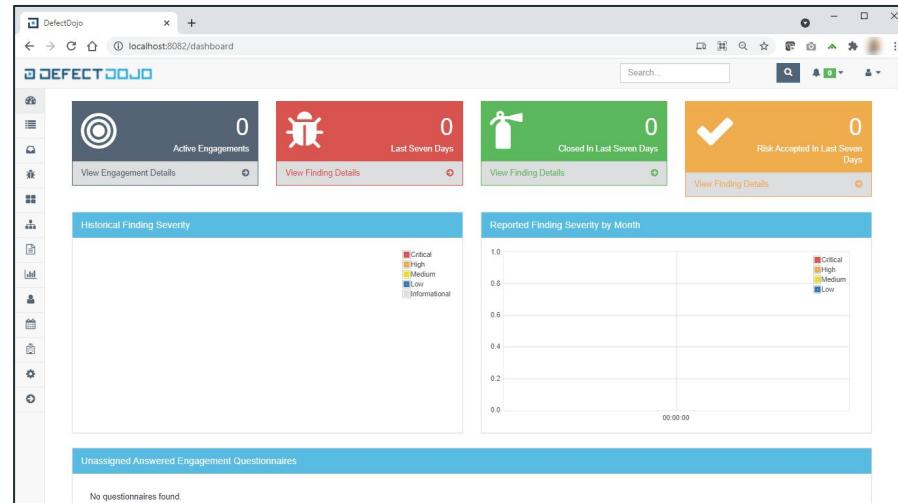
```
# docker-compose logs initializer | grep "Admin password:"
```

Ingresamos al portal

<http://localhost:8082>

User: admin

Pass:3jdm0WnvvBZjkhzhH4ccns



Puesta en marcha de DefectDojo

Identificación de la API

`http://localhost:8082/api/key-v2`

The screenshot shows the DefectDojo web interface. The left sidebar has a dark theme with white icons and text. The main content area has a light background. The title 'API v2 Key' is centered above a message block. The message block contains two lines of pink text: 'Your current API key is' followed by a long hex string, and 'Your current API Authorization Header value is' followed by another long hex string. Below the message block is a question: 'Has your key been exposed? Are you ready for a new one?'. At the bottom is a blue button labeled 'Generate New Key'.

DEFECTDOJO

Dashboard Products Engagements Findings Components Endpoints Terminal Reports

Home / API Key

API v2 Key

Your current API key is `9010a9fe648b30b59bdb72a156edaf0285c90a8e`

Your current API Authorization Header value is `Token 9010a9fe648b30b59bdb72a156edaf0285c90a8e`

Has your key been exposed? Are you ready for a new one?

Generate New Key

Jenkins + OWASP ZAP + DefectDojo

Jenkins + OWASP ZAP + DefectDojo

Si tenemos problemas de permisos, debo agregar al usuario Jenkins dentro del grupo docker

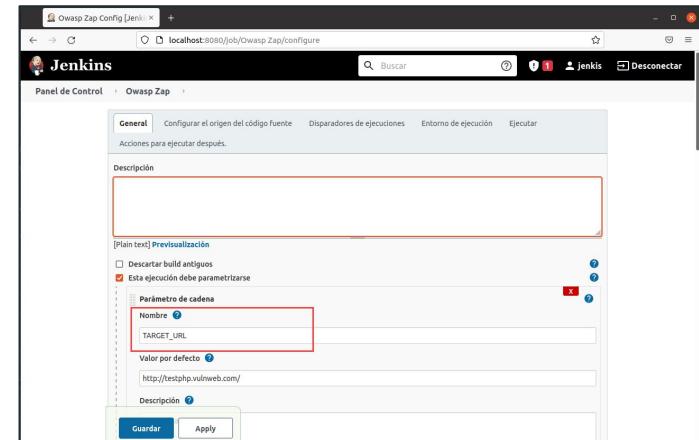
```
# sudo usermod -aG docker jenkins  
# sudo service jenkins restart
```

Creamos una nueva tarea

Debemos tildar “Esta ejecución debe parametrizarse” seleccionamos “Parámetro de cadena”.

Esto nos va a permitir ingresar la URL y el nombre del reporte.

Nombre: TARGET_URL



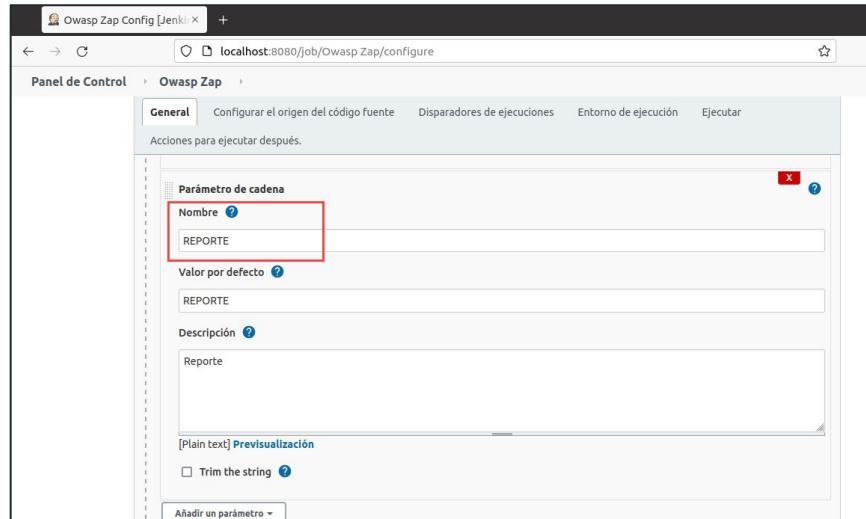
Jenkins + OWASP ZAP + DefectDojo

Creamos una nueva tarea

Debemos tildar “Esta ejecución debe parametrizarse”, seleccionamos “Parámetro de cadena”.

Esto nos va a permitir ingresar la URL y el nombre del reporte.

Nombre: REPORTE



Jenkins + OWASP ZAP + DefectDojo

Creamos una nueva tarea

Debemos tildar “Esta ejecución debe parametrizarse”, esto no va a permitir ingresar la URL y el nombre del reporte.

Ejecutar:

“Comando”

```
/home/demos/demos/scan_zap_munual.sh $TARGET_URL $REPORTER
```



Jenkins + OWASP ZAP + DefectDojo

Creamos una nueva tarea

Agregamos una acción “Publish HTML reports”

Reports -> Agregar

HTML directory to archive:

`/home/demos/demos/`

Index page[s]:

`${REPORTE}.html`

Y guardamos la tarea

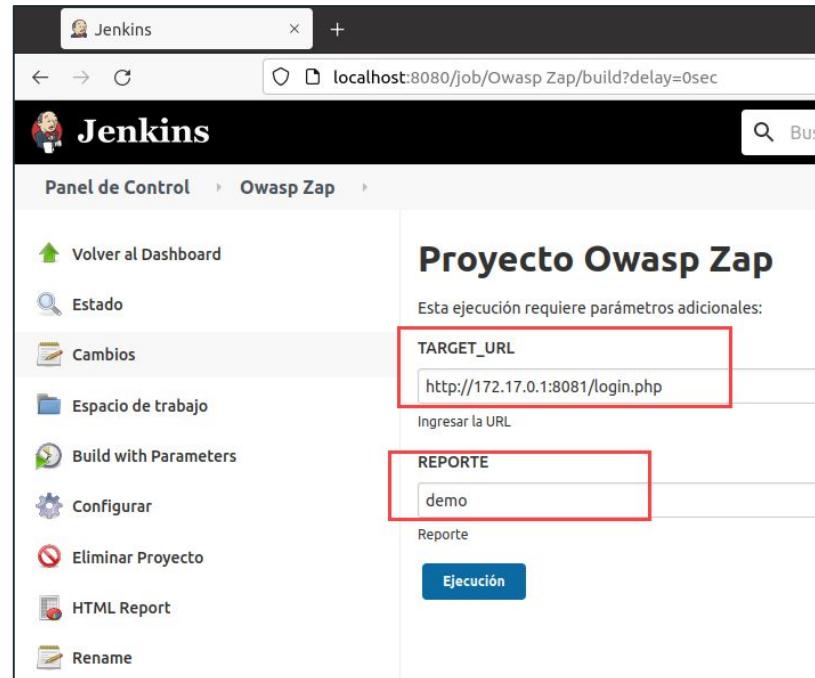
The screenshot shows the Jenkins configuration interface for the 'Publish HTML reports' action. The form fields are as follows:

- HTML directory to archive**: `/home/demos/demos/`
- Index page[s]**: `${REPORTE}.html`
- Index page title[s] (Optional)**: (empty field)
- Report title**: `HTML Report`

At the bottom of the form is a large blue 'Agregar' button.

Jenkins + OWASP ZAP + DefectDojo

Ejecutamos la tarea

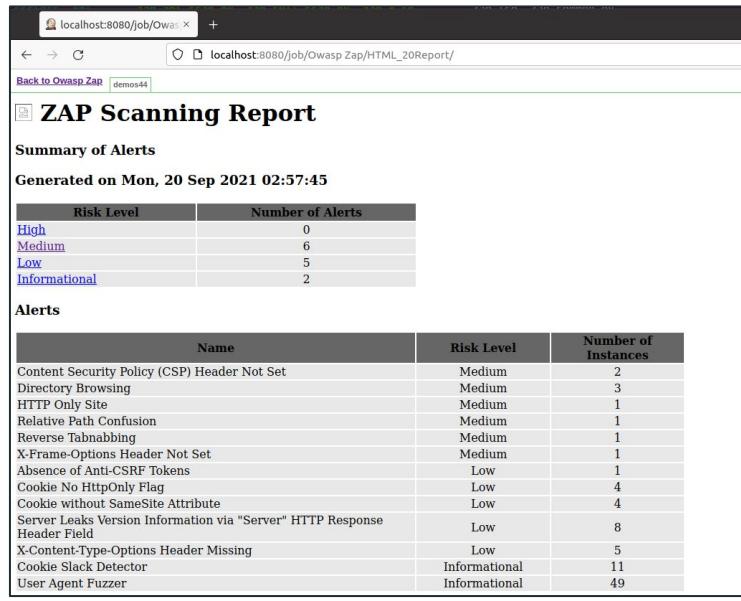


Jenkins + OWASP ZAP + DefectDojo

Visualizamos el reporte



The Jenkins dashboard shows the 'Owasp Zap' job. The 'HTML Report' option under the 'Owasp Zap' job is highlighted with a red box.



The OWASP ZAP Scanning Report is displayed. It includes a summary of alerts generated on Monday, 20 Sep 2021 at 02:57:45, and a detailed table of findings categorized by risk level.

Risk Level	Number of Alerts
High	0
Medium	6
Low	5
Informational	2

Alerts

Name	Risk Level	Number of Instances
Content Security Policy (CSP) Header Not Set	Medium	2
Directory Browsing	Medium	3
HTTP Only Site	Medium	1
Relative Path Confusion	Medium	1
Reverse Tabnabbing	Medium	1
X-Frame-Options Header Not Set	Medium	1
Absence of Anti-CSRF Tokens	Low	1
Cookie No HttpOnly Flag	Low	4
Cookie without SameSite Attribute	Low	4
Server Leaks Version Information via "Server" HTTP Response Header Field	Low	8
X-Content-Type-Options Header Missing	Low	5
Cookie Slack Detector	Informational	11
User Agent Fuzzer	Informational	49

Jenkins + OWASP ZAP + DefectDojo

Se puede visualizar que el scan fue importado en Defectdojo

The screenshot shows the DefectDojo application running at `localhost:8082/engagement/3`. The left sidebar has a navigation menu with items: Dashboard, Products, Engagements (selected), Findings, Components, Endpoints, Reports, and Metrics. The main content area displays a table of findings. The table has columns: Title / Type, Date, Lead, Total Findings, Active (Verified), Mitigated, Duplicates, Notes, and Reimports. There are two entries:

Title / Type	Date	Lead	Total Findings	Active (Verified)	Mitigated	Duplicates	Notes	Reimports
SonarQube API Import	Oct. 15, 2021 - Oct. 15, 2021		113	113 (0)	0	0		0
ZAP Scan	Oct. 15, 2021 - Oct. 15, 2021		13	13 (13)	0	0		0

Jenkins + SonarQube + DefectDojo

Jenkins + SonarQube + DefectDojo

Elegimos “Secret Text” y pegamos nuestro Token de SonarQube

The screenshot shows the Jenkins Credentials Provider interface. The title bar says "Jenkins Credentials Provider: Jenkins". Below it, there's a section titled "Add Credentials" with a house icon. The "Kind" dropdown is set to "Secret text", which is highlighted with a red box. The "Scope" dropdown is set to "Global (Jenkins, nodes, items, all child items, etc)". The "Secret" field contains a series of dots, also highlighted with a red box. The "ID" field is empty. The "Description" field contains "Sonar-Token", also highlighted with a red box. At the bottom, there are "Add" and "Cancel" buttons.

Jenkins + SonarQube + DefectDojo

Configuramos SonarQube con los datos de nuestro servidor y luego seleccionamos en agregar un token.

The screenshot shows the Jenkins configuration interface for SonarQube servers. The 'Name' field contains 'SonarQube' and the 'Server URL' field contains 'http://192.168.0.110:9000'. Both fields are highlighted with red boxes. The 'Server authentication token' dropdown is set to 'Sonar-Token' and has an 'Add' button next to it, also highlighted with a red box. A note below states: 'SonarQube authentication token. Mandatory when anonymous access is disabled.' At the bottom right are 'Advanced...' and 'Delete SonarQube' buttons.

Jenkins + SonarQube + DefectDojo

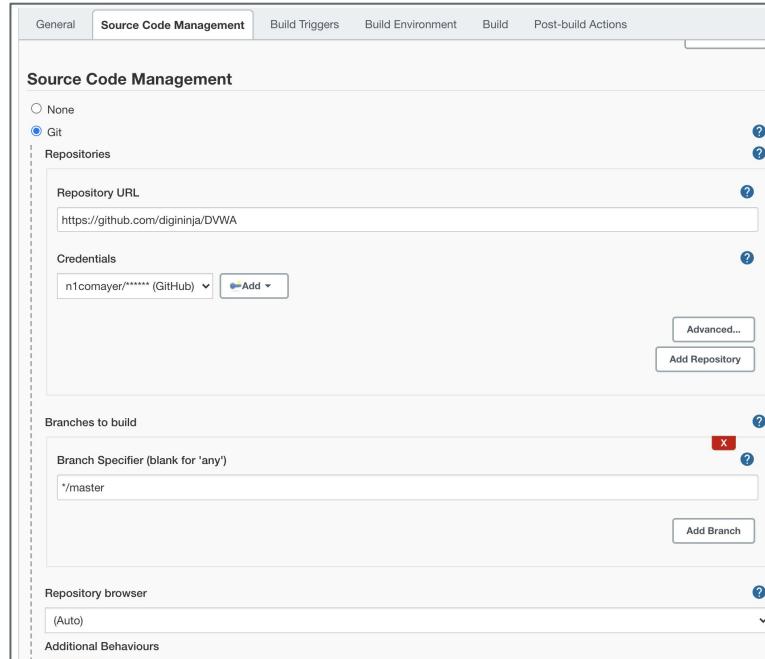
Creamos un nuevo proyecto “vacío” en Jenkins

The screenshot shows the Jenkins dashboard with a search bar and a 'log in' button at the top right. Below the header, there's a breadcrumb navigation: 'Dashboard > All'. A central panel titled 'Enter an item name' contains a text input field with the placeholder 'DEMO - NERDEARLA' and a note '» Required field'. Below this, there's a list of project types with their icons and descriptions:

- Freestyle project**: This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Pipeline**: Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**: Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder**: Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- Multibranch Pipeline**: Creates a set of Pipeline projects according to detected branches in one SCM repository.
- Organization Folder**: Creates a set of multibranch project subfolders by scanning for repositories.

Jenkins + SonarQube + DefectDojo

Agregamos el repo con el proyecto que vamos a analizar



Jenkins + SonarQube + DefectDojo

Agregamos los datos del proyecto que generamos en SonarQube

The screenshot shows the Jenkins 'Build' configuration page for a job named 'Demo Nerdearla - DVWA'. The 'Build' tab is selected. In the 'Analysis properties' section, there is a red box highlighting the following configuration:

```
# must be unique in a given SonarQube instance
sonar.projectKey=Demo-Nerdearla--DVWA

# defaults to project key
sonar.projectName=Demo-Nerdearla--DVWA

# defaults to 'not provided'
sonar.projectVersion=1.0
```

Jenkins + SonarQube + DefectDojo

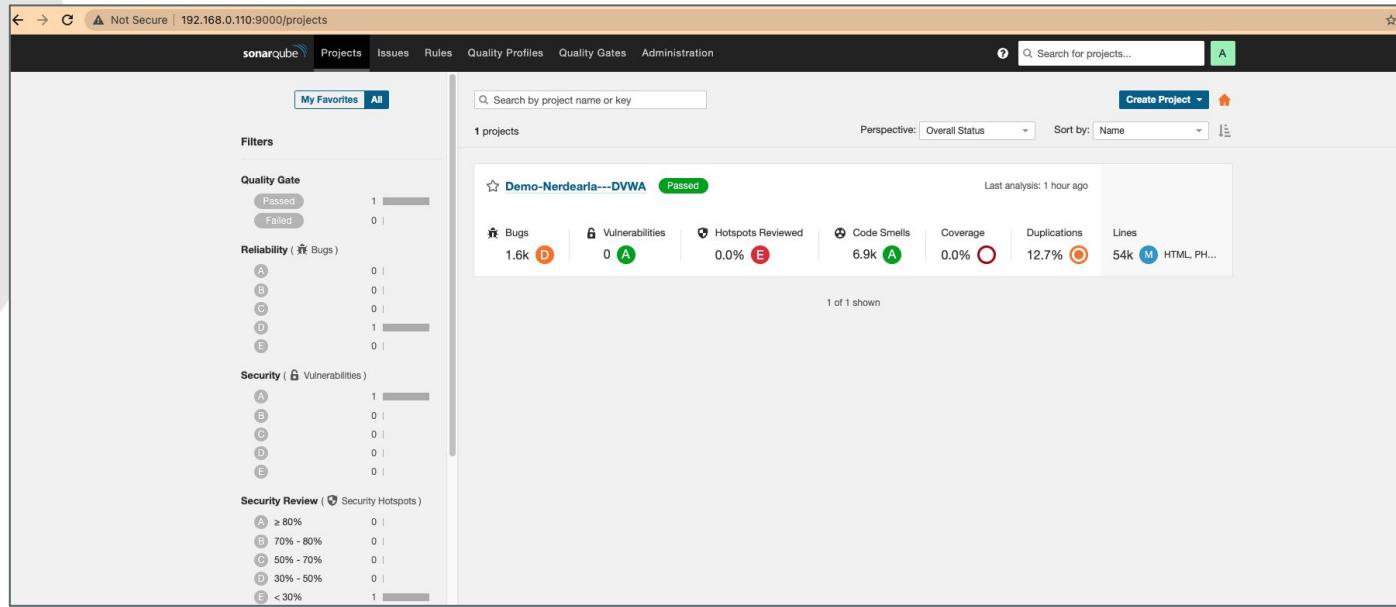
Hacemos Click en “Build Now” y verificamos los resultados

The screenshot shows the Jenkins dashboard for the 'Demo Nerdearla - DVWA' project. The sidebar menu includes 'Back to Dashboard', 'Status', 'Changes', 'Workspace', **Build Now** (highlighted with a red box), 'Configure', 'Delete Project', 'SonarQube', and 'Rename'. The main content area displays the 'Project Demo Nerdearla - DVWA' page. It features a 'SonarQube Quality Gate' section with a summary for 'Demo-Nerdearla---DVWA' showing 'Passed' status and 'Success' server-side processing. Below this is a 'Permalinks' section with a bulleted list of recent builds: 'Last build (#5), 55 min ago', 'Last stable build (#5), 55 min ago', 'Last successful build (#5), 55 min ago', and 'Last completed build (#5), 55 min ago'. At the bottom left is a 'Build History' table with columns 'Build' and 'trend'. The table lists builds #6 through #1, all of which are green and show a downward trend arrow, indicating they were successful.

Build	trend
#6 Oct 16, 2021, 12:54 AM	▼
#5 Oct 15, 2021, 11:58 PM	▼
#4 Oct 15, 2021, 11:47 PM	▼
#3 Oct 15, 2021, 11:45 PM	▼
#2 Oct 15, 2021, 11:44 PM	▼
#1 Oct 15, 2021, 11:40 PM	▼

Jenkins + SonarQube + DefectDojo

Verificamos el resultado en SonarQube



Jenkins + SonarQube + DefectDojo

Se observa que el SCAN fue importado correctamente

The screenshot shows the DefectDojo application running on localhost:8082. The left sidebar has navigation links: Dashboard, Products, Engagements, Findings, Components, Endpoints, Reports, and Metrics. The main content area displays a table of findings. The table has columns: Title / Type, Date, Lead, Total Findings, Active (Verified), Mitigated, Duplicates, Notes, and Reimports. Two rows are present:

Title / Type	Date	Lead	Total Findings	Active (Verified)	Mitigated	Duplicates	Notes	Reimports
SonarQube API Import	Oct. 15, 2021 - Oct. 15, 2021		113	113 (0)	0	0	0	
ZAP Scan	Oct. 15, 2021 - Oct. 15, 2021		13	13 (13)	0	0	0	

The first row, representing the SonarQube API Import, is highlighted with a red box.

Pipeline

Pipeline

Agrego una nueva credencial

- *Panel de control -> Administrar jenkins -> Manage credentials*

The screenshot shows the Jenkins 'Manage Credentials' interface. At the top, there's a header bar with tabs for 'P' (Pipeline), 'Store' (sorted by name), and 'Domains'. Below the header, there's a table with one row. The first column contains a Jenkins icon and the text 'Jenkins'. The second column contains a 'global' label and a 'Add credentials' button. A red rectangular box highlights the 'Add credentials' button.

Pipeline

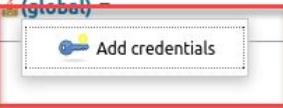
Agrego una nueva credencial

- *Panel de control -> Administrar jenkins -> Manage credentials*

Stores scoped to Jenkins

P	Store	Domains
	Jenkins	(global)

Add credentials



Kind	Secret text
Scope	Global (Jenkins, nodes, items, all child items, etc)
Secret
ID	sonar
Description	sonar

Pipeline

Debemos crear un pipeline para ejecutar los Scans de forma automatizada:

Panel de Control

- Nuevo Tarea**
- Personas
- Historial de Construcción
- Administrar Jenkins
- Mis vistas
- Lockable Resources
- New View

Enter an item name

» Required field

Crear un proyecto de estilo libre
Esta es la característica principal de Jenkins, la de ejecutar el proyecto combinando cualquier tipo de repositorio de software (SCM) con cualquier modo de construcción o ejecución (make, ant, mvn, rake, script ...). Por tanto se podrá tanto compilar y empaquetar software, como ejecutar cualquier proceso que requiera monitorización.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Crear un proyecto multi- configuración
Adecuado para proyectos que requieren un gran número de configuraciones diferentes, como testear en múltiples entornos, ejecutar sobre plataformas concretas, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Pipeline

Debemos crear un pipeline e ingresar como secuencia lo siguiente:

Pipeline

Definition

Pipeline script

Script

```
17 }  
18 }  
19 } stage('ScanSonarqube') {  
20 steps {  
21     sh '''  
22         ... /home/demos/demos/scansonarqube.sh  
23     }  
24 }  
25  
26 } stage('ScanOwaspzap') {  
27 steps {  
28     sh '''  
29         ... /home/demos/demos/scan_zap.sh  
30     }  
31 }  
32 }  
33 }  
34 }  
35 }
```

Use Groovy Sandbox

Pipeline Syntax

Pipeline

En el pipeline definimos los siguientes stages:

- CrearProducto
- CrearEngagement
- ScanSonarqube
- ScanOwaspzap

Para descargarlo ir a:

<https://github.com/marcositu/workshop-decsecops/blob/main/pipeline>

Pipeline

Ejecutamos el pipeline desde “Build with Parameters”:



The screenshot shows the Jenkins interface for a pipeline named "Elpipelinodelagente". On the left, a sidebar menu includes options like "Back to Dashboard", "Status", "Changes", "Build with Parameters" (which is highlighted with a red box), "Configurar", "Eliminar Pipeline", "Full Stage View", "Rename", and "Pipeline Syntax". The main area is titled "Stage View" and displays four stages: "Crearproducto", "Crearengagement", "ScanSonarqube", and "ScanOwaspzap". Below these stages, a summary states "Average stage times: (Average full run time: ~8min 25s)". A specific build entry for "#60" is shown, dated "Oct 21" at "00:57", with a status of "No Changes". The "ScanSonarqube" stage has a blue progress bar indicating it is still running.

Crearproducto	Crearengagement	ScanSonarqube	ScanOwaspzap
1s	1s	2min 4s	2min 5s
1s	1s	4min 7s	4min 11s

Reflexiones y lecciones aprendidas

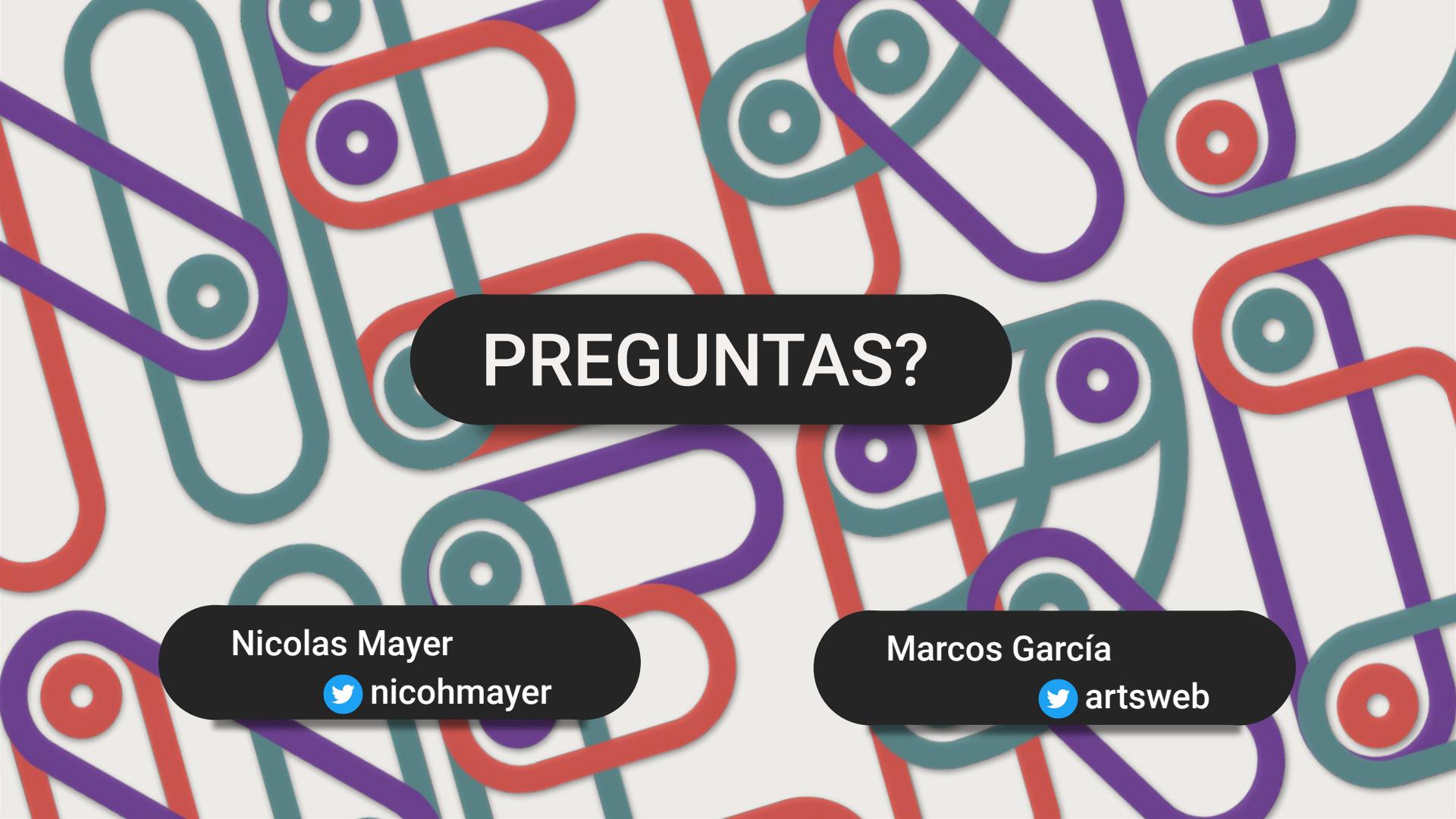
Reflexiones y lecciones aprendidas

- Encontrar vulnerabilidades antes de llegar a producción ahorra tiempo y \$.
- Aplicaciones más seguras y con mayor calidad.
- Seguimiento continuo de las aplicaciones.
- Mayor escalabilidad.
- El cliente medio tarda **174** días en corregir una vulnerabilidad encontrada mediante DAST. Sin embargo, quienes implementaron DevSecOps lo hacen en solo **92** días.
- Si miramos las vulnerabilidades encontradas en el desarrollo usando SAST, una empresa promedio tarda **113** días, mientras que las empresas DevSecOps tardan solo **51** días.

Referencias

Referencias

- <https://www.jenkins.io>
- <https://dvwa.co.uk>
- <https://www.sonarqube.org>
- <https://www.defectdojo.org>
- <https://owasp.org>
- <https://github.com/marcositu/workshop-devsecops>



PREGUNTAS?

Nicolas Mayer
 nicohmayer

Marcos García
 artsweb