
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
    - Docker
    - Jenkins
    - DVWA
    - SonarQube
    - OWASP ZAP
    - DefectDojo
    - Jenkins + Zap + DefectDojo
    - Jenkins + SonarQube + DefectDojo
    - Pipeline
5. Reflexiones y lecciones aprendidas
6. Referencias

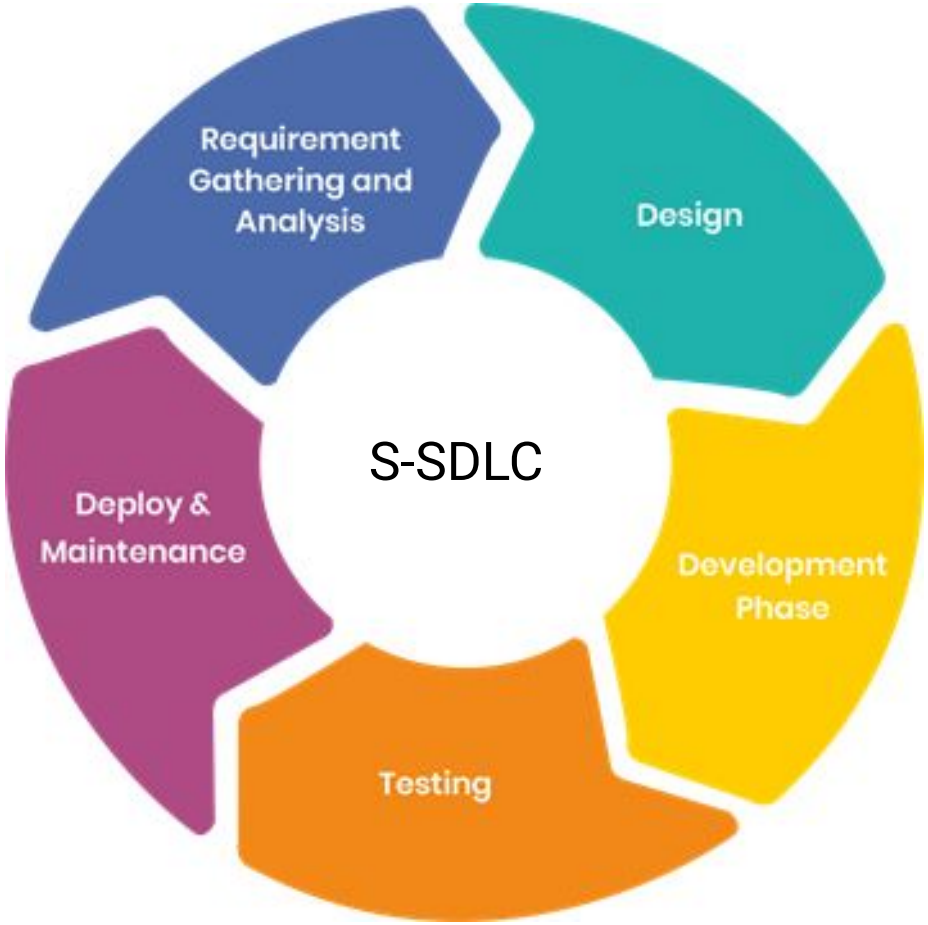
Referencias

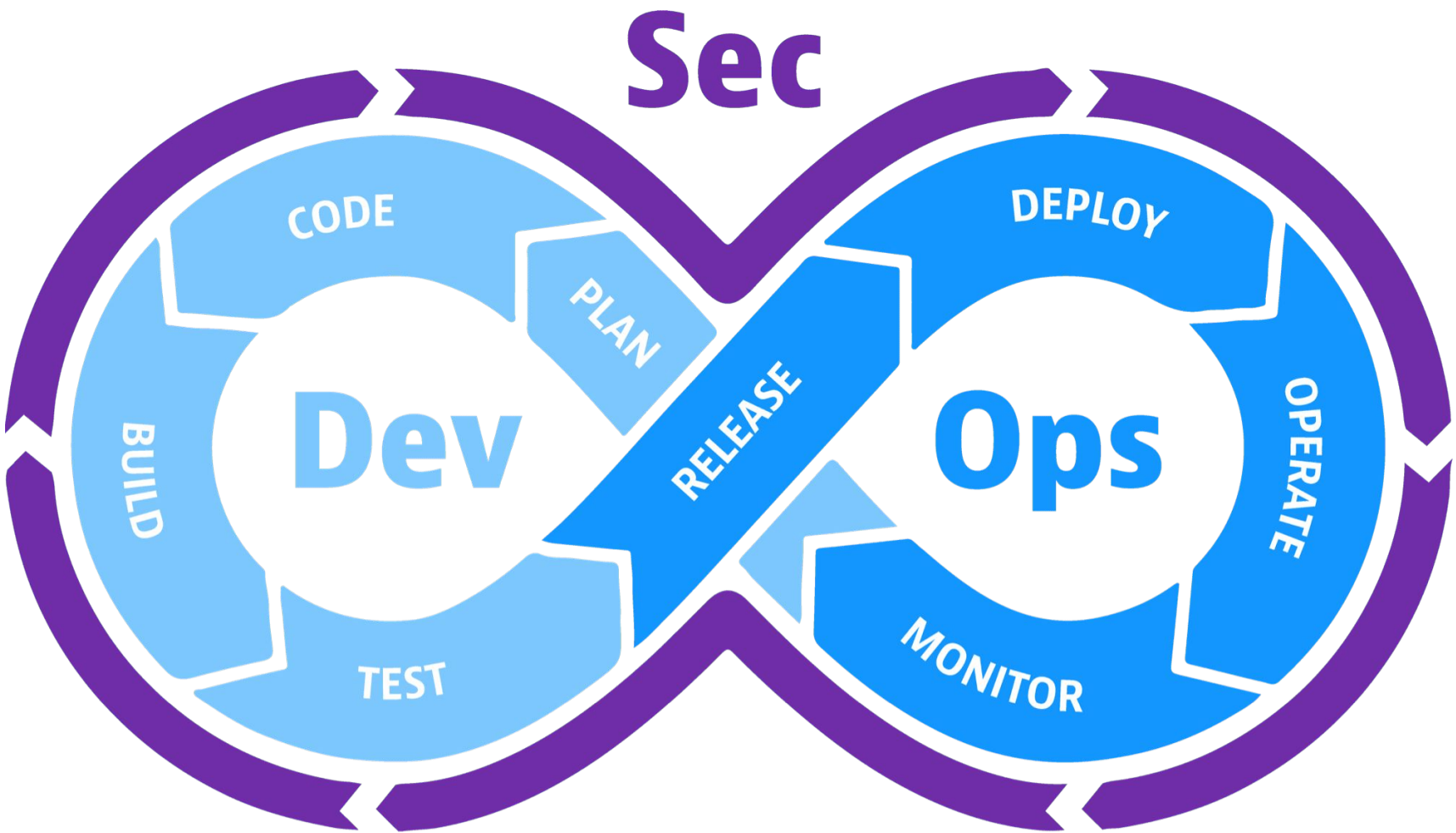
Como usaremos los colores durante el Workshop:

```
nombre-de-archivo.yml
variable: contenido-del-archivo

# comando-id up -X
```

¿Cómo llegamos acá?

Evolución de Desarrollo de Software


Requirement Gathering and Analysis
Design
S-SDLC
Development Phase
Testing
Deploy & Maintenance


Sec
CODE
PLAN
BUILD
Dev
TEST
RELEASE
DEPLOY
OPERATE
Ops
MONITOR

TOP 10

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

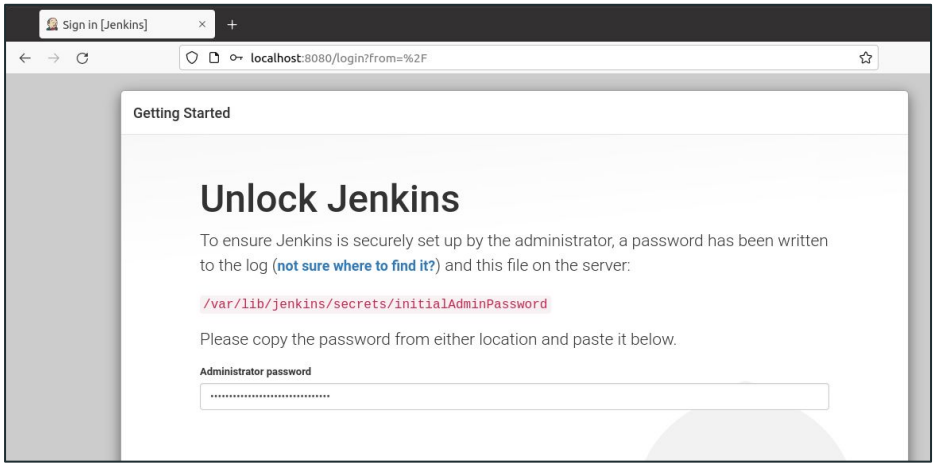
Instalando Jenkins en nuestro servidor

```
# 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

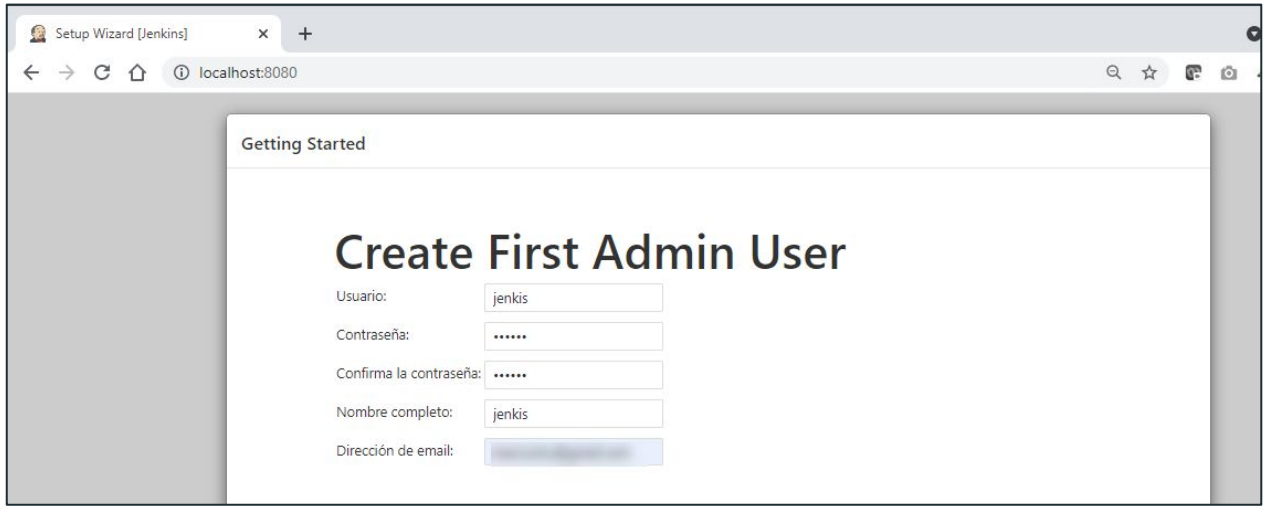
Instalando Jenkins en nuestro servidor docker host
Ingresar a la URL http://localhost:8080/

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



Puesta en marcha de Jenkins

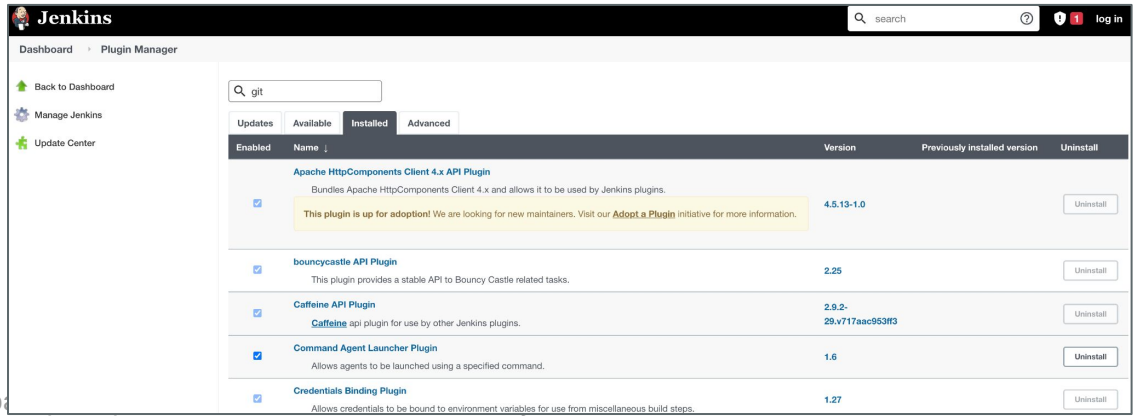
Ingresamos al sitio Web **http://localhost:8080** y seguimos todos los pasos hasta la creación del usuario.



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




DVWA

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)

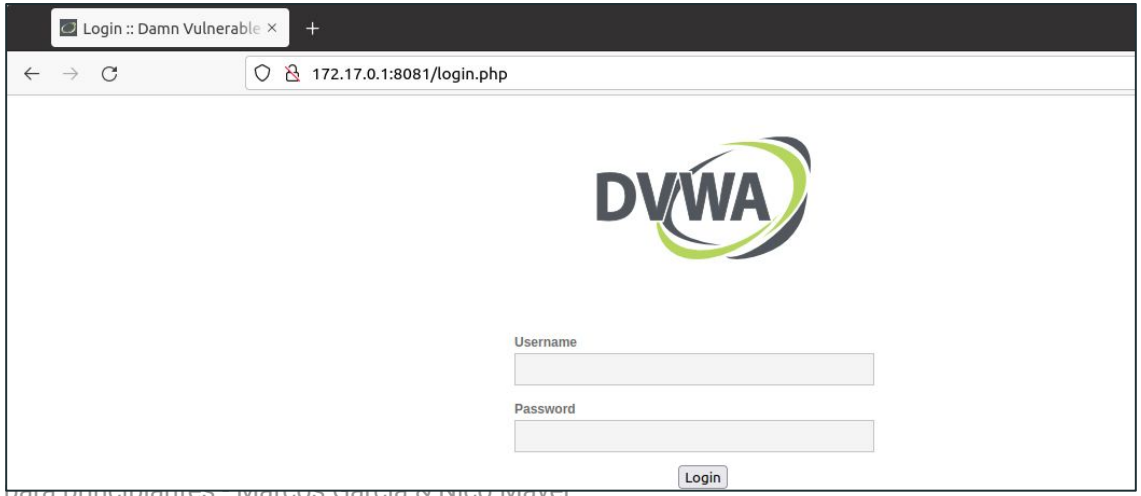
Puesta en marcha de DVWA

Para iniciar e instalar DVWA debemos ejecutar el siguiente comando

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

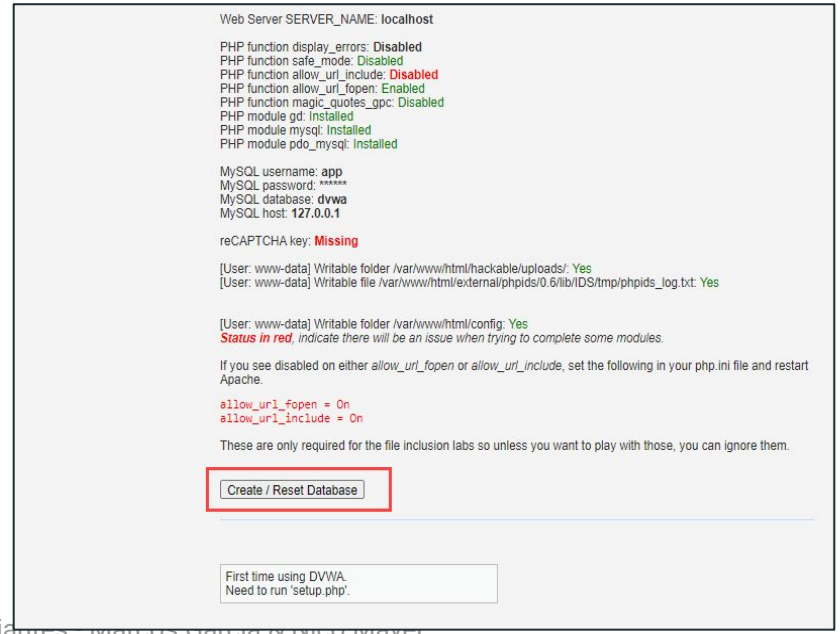
Luego ingresamos a http://localhost:8081 para su configuración

```
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**".




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'
```

Puesta en marcha de SonarQube

Iniciamos el docker compose

Recomendamos agregar el siguiente parámetro dentro del archivo

`docker-compose.yml`

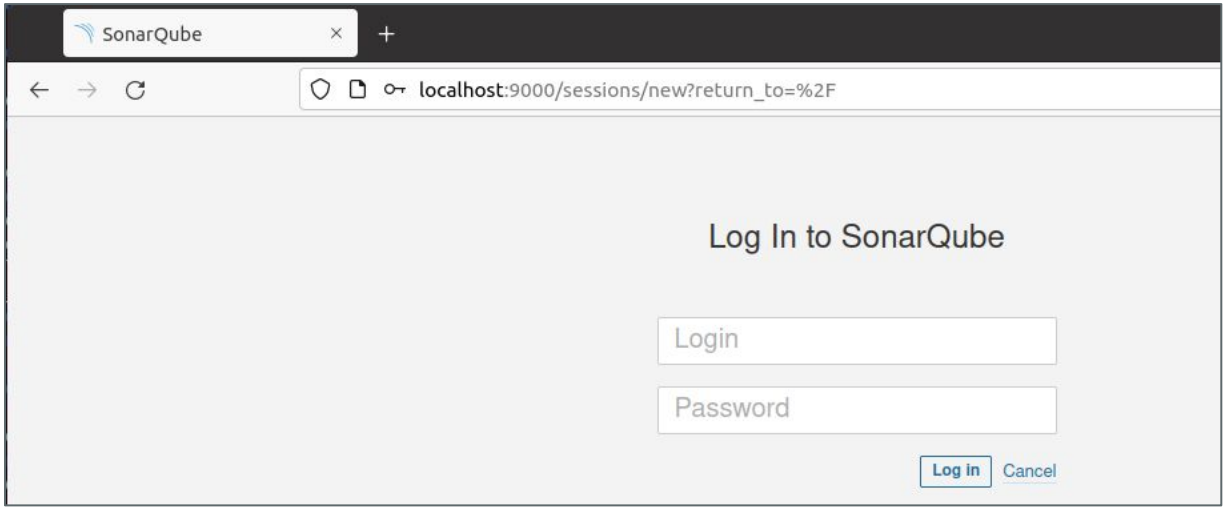
```
restart: always
```

```
# 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

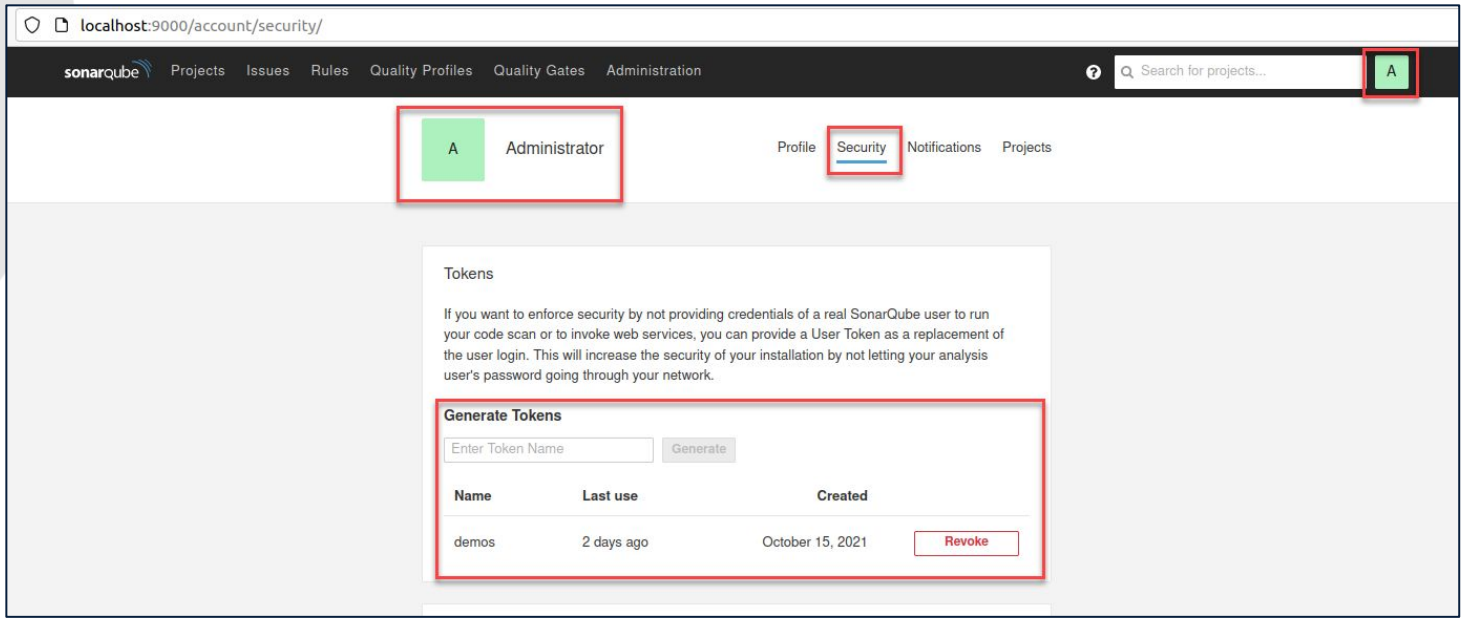
Instalar sonar-scanner

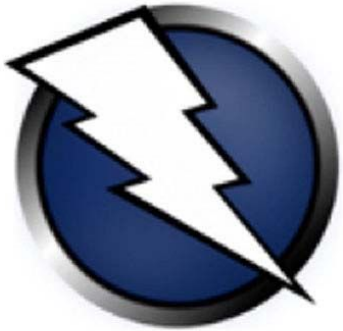
```
# wget
https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cl
i-4.6.2.2472-linux.zip
# unzip sonar-scanner-cli-4.6.2.2472-linux.zip
# echo 'export PATH="~/demos/sonar/sonar-scanner-4.6.2.2472-linux/bin:$PATH"' >>
~/.bashrc
# source ~/.bashrc
```

Puesta en marcha de SonarQube

Cómo configurar la API Key de SonarQube

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





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
```


DEFECT
DOJO

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

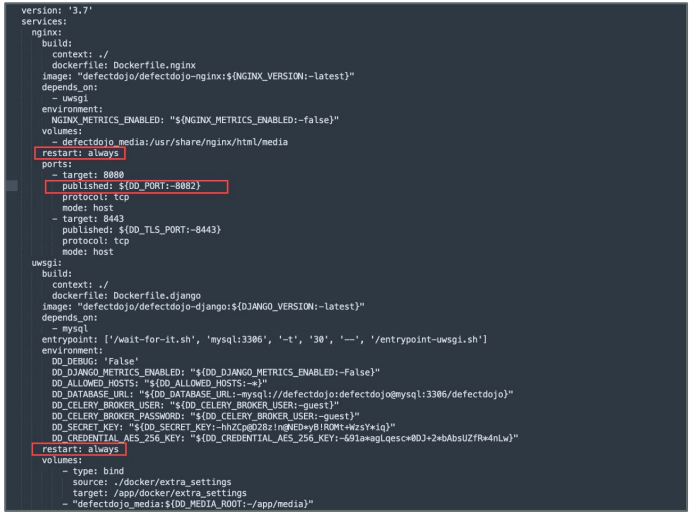
como también agregar el siguiente parámetro dentro

del archivo `docker-compose.yml`

`restart: always`

Ejecutamos docker compose

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



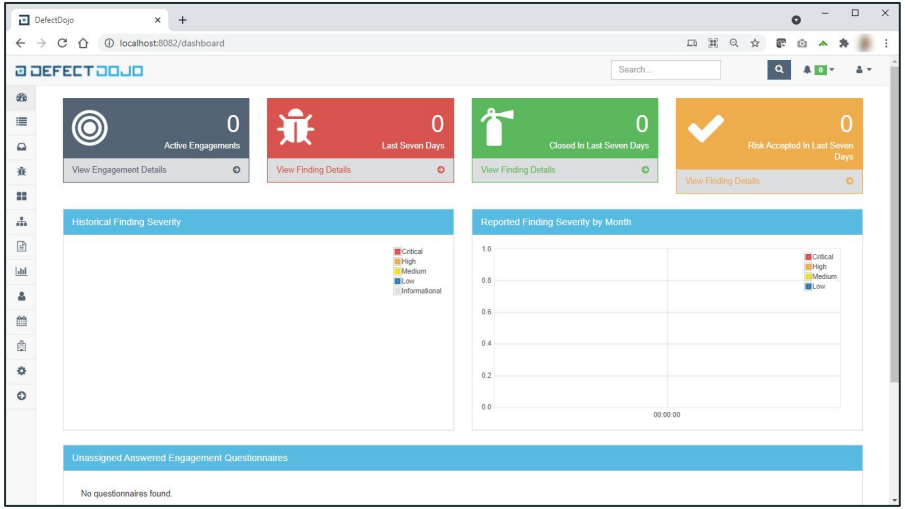
Puesta en marcha de DefectDojo

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
```

Jenkins + OWASP ZAP + DefectDojo

Jenkins + OWASP ZAP + DefectDojo

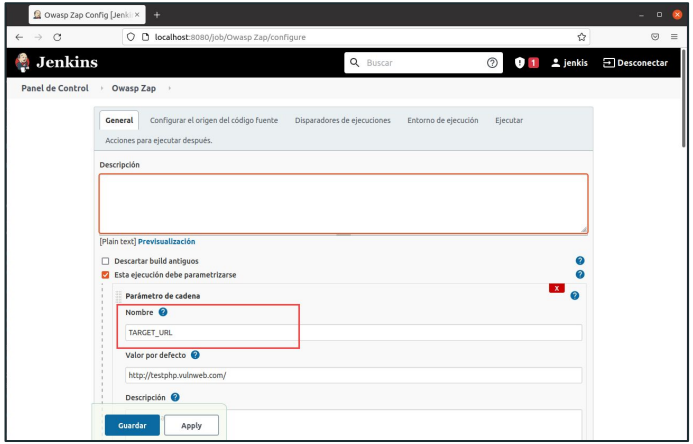
Si tenemos problemas de permisos, debo agregar al usuario Jenkins dentro del grupo docker

```
# 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

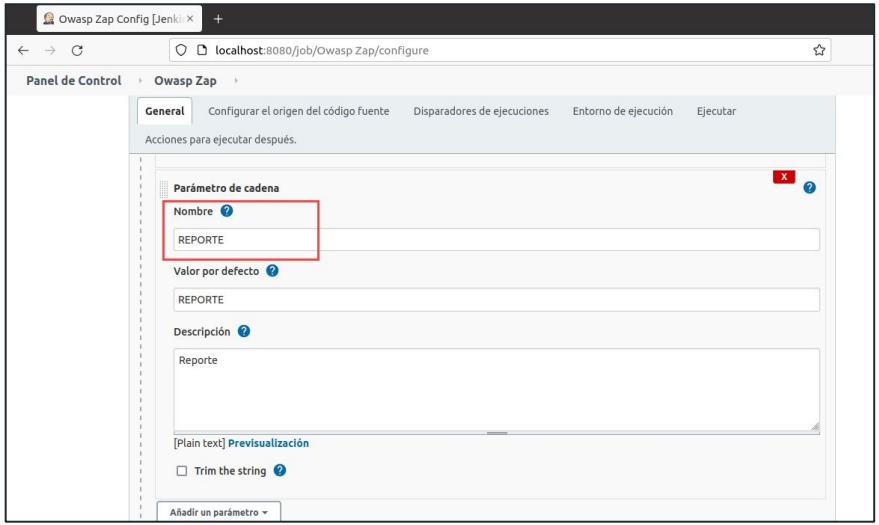


Jenkins + OWASP ZAP + DefectDojo

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.

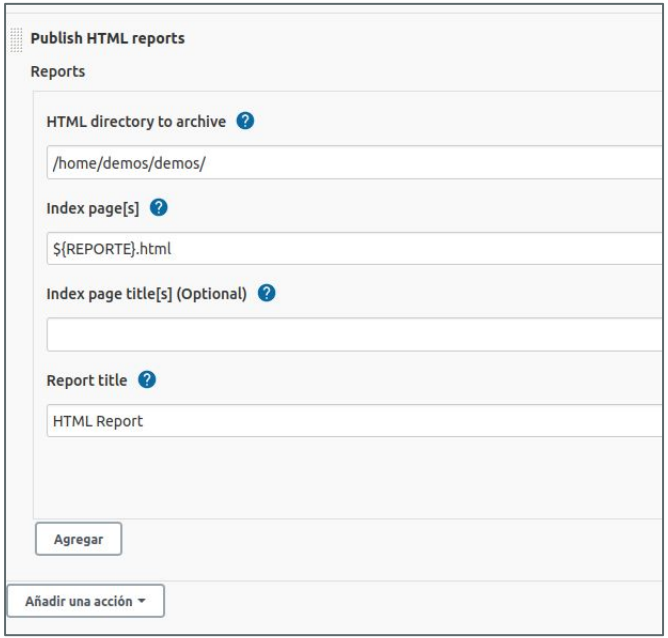
Ejecutar:
“Comando”
/home/demos/demos/scan_zap_munual.sh $TARGET_URL $REPORTE

Jenkins + OWASP ZAP + DefectDojo

Creamos una nueva tarea

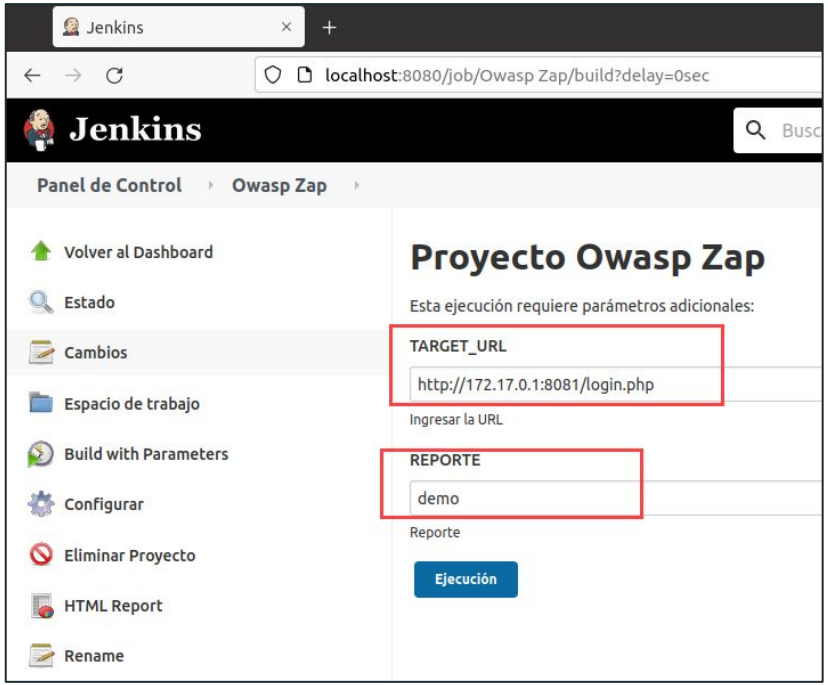
Agregamos una acción “Publish HTML reports”
Reports -> Agregar
HTML directory to archive:
/home/demos/demos/

Index page[s]:
${REPORTE}.html
Y guardamos la tarea



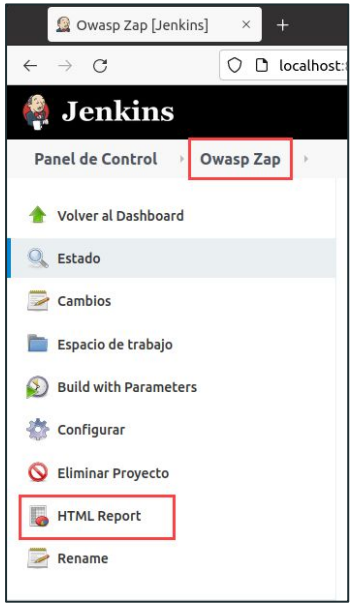
Jenkins + OWASP ZAP + DefectDojo

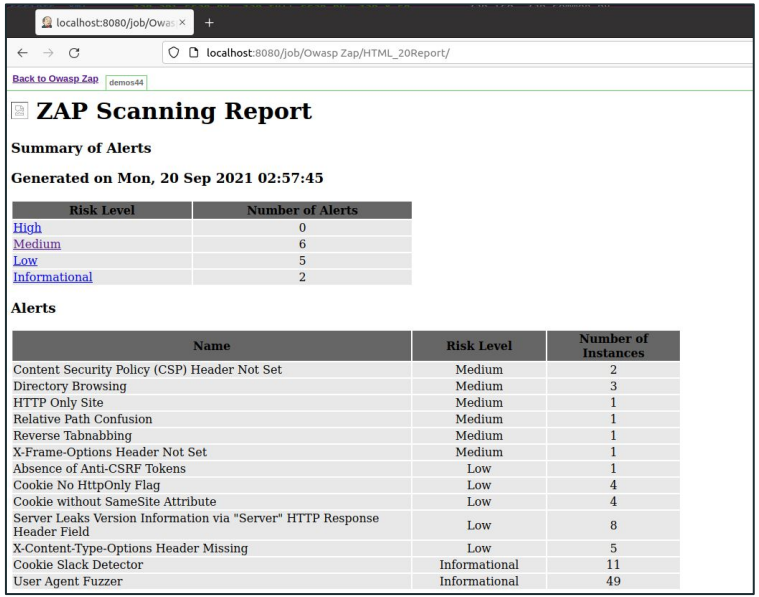
Ejecutamos la tarea



Jenkins + OWASP ZAP + DefectDojo

Visualizamos el reporte





localhost:8080/job/Owasp Zap/HTML_20Report/

Back to Owasp Zap demos44

ZAP Scanning Report

Summary of Alerts

Generated on Mon, 20 Sep 2021 02:57:45

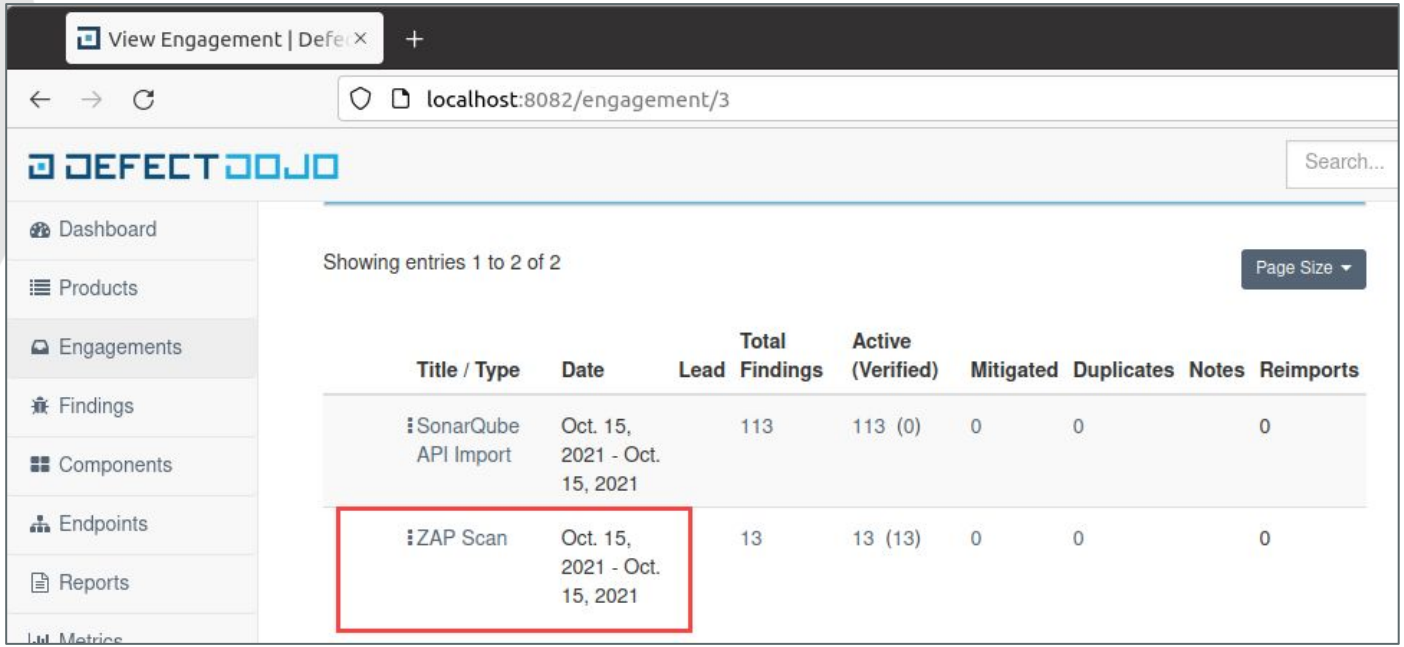
Risk Level	Number of Alerts
High	0
Medium	6
Low	5
Informational	2

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

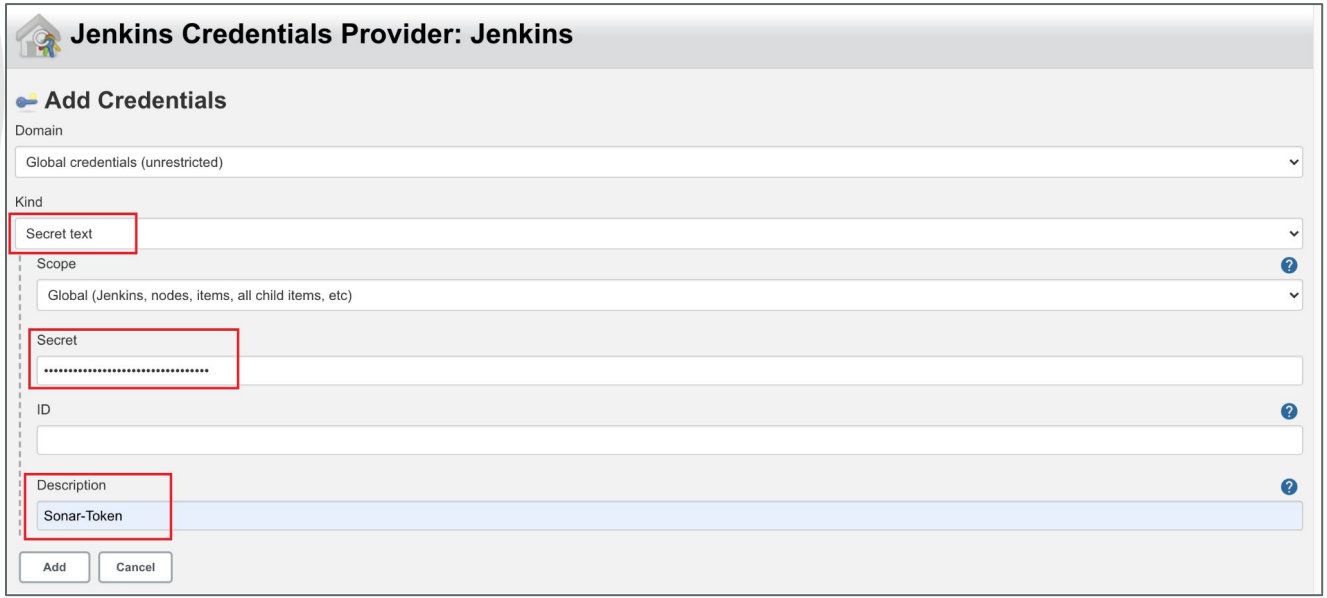
Se puede visualizar que el scan fue importado en Defectdojo



Jenkins + SonarQube + DefectDojo

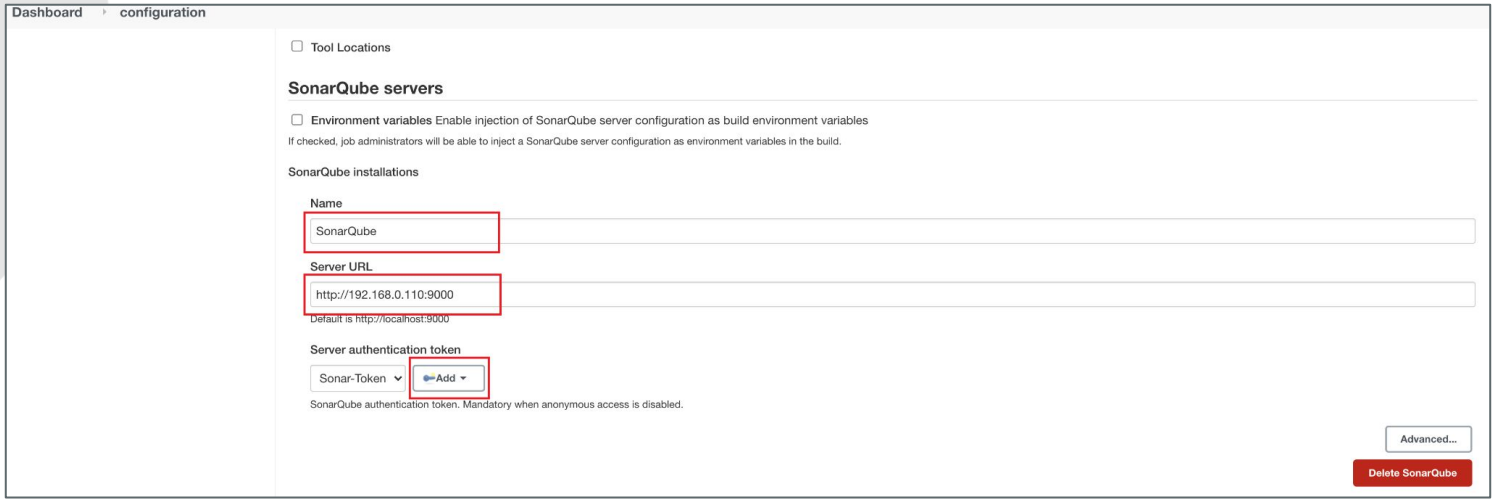
Jenkins + SonarQube + DefectDojo

Elegimos “Secret Text” y pegamos nuestro Token de SonarQube



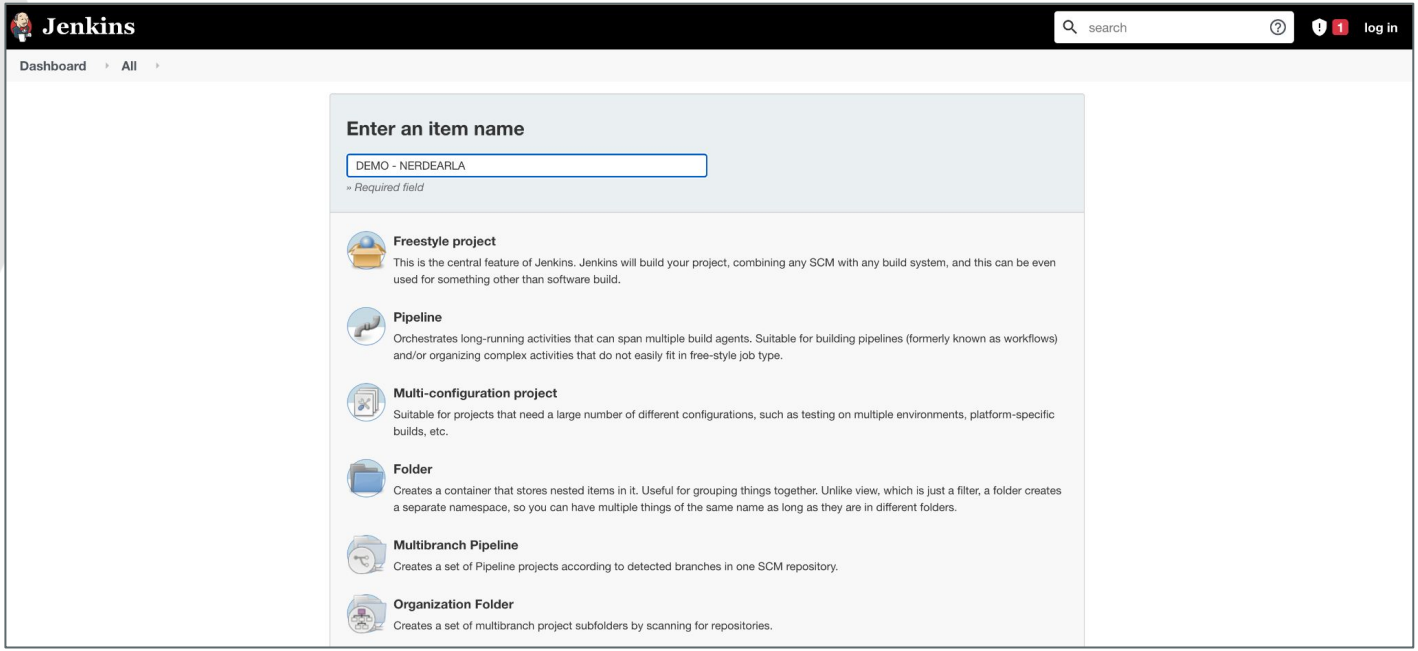
Jenkins + SonarQube + DefectDojo

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



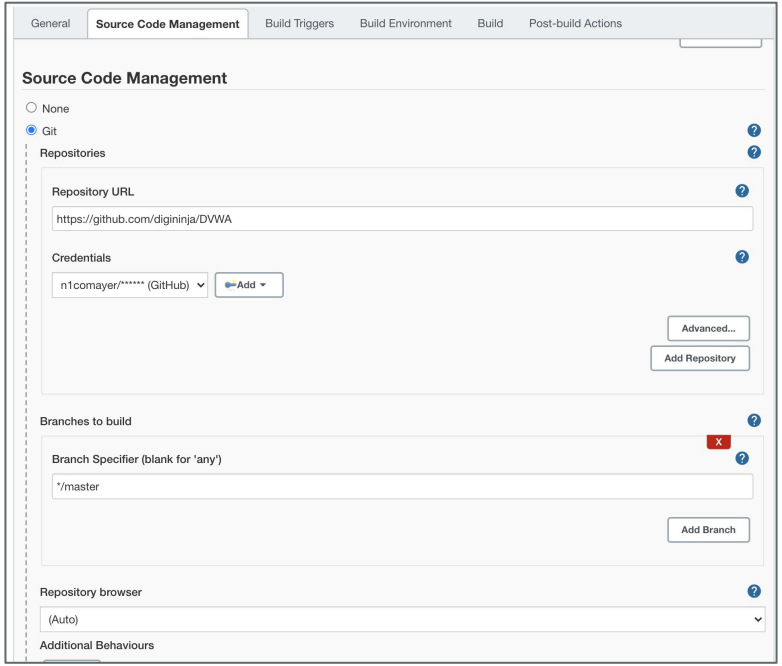
Jenkins + SonarQube + DefectDojo

Creamos un nuevo proyecto “vacío” en Jenkins



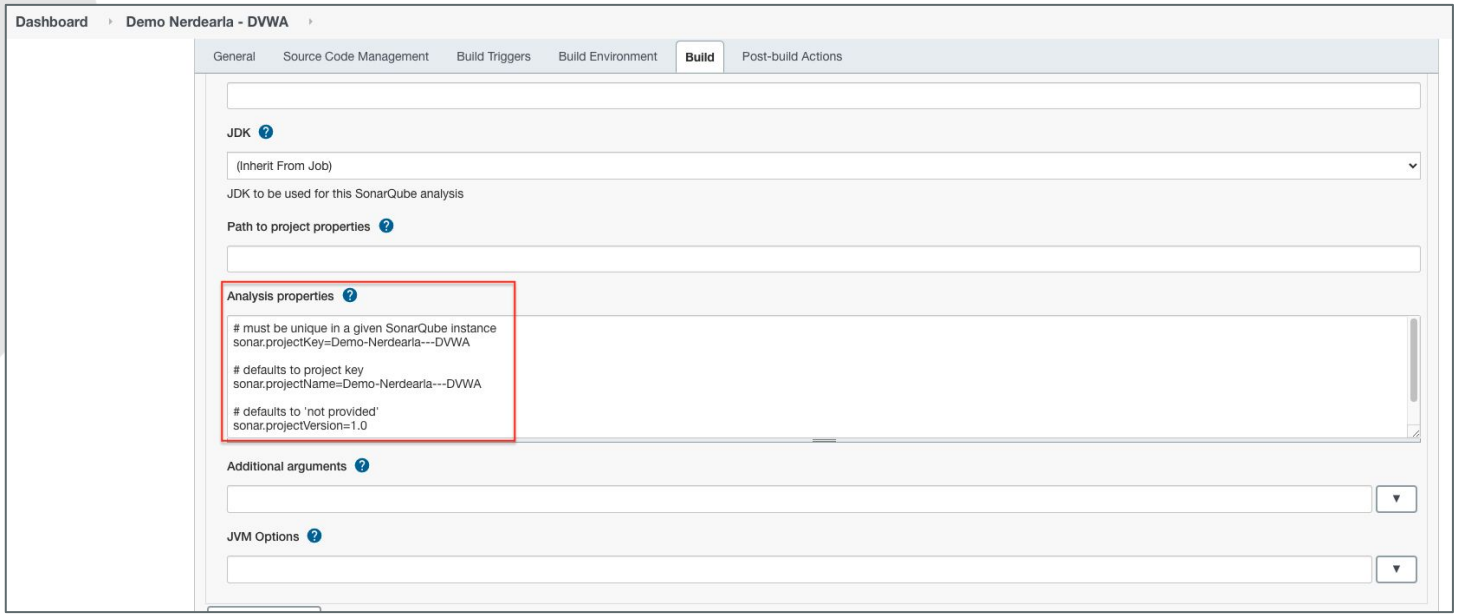
Jenkins + SonarQube + DefectDojo

Agregamos el repo con el proyecto que vamos a analizar



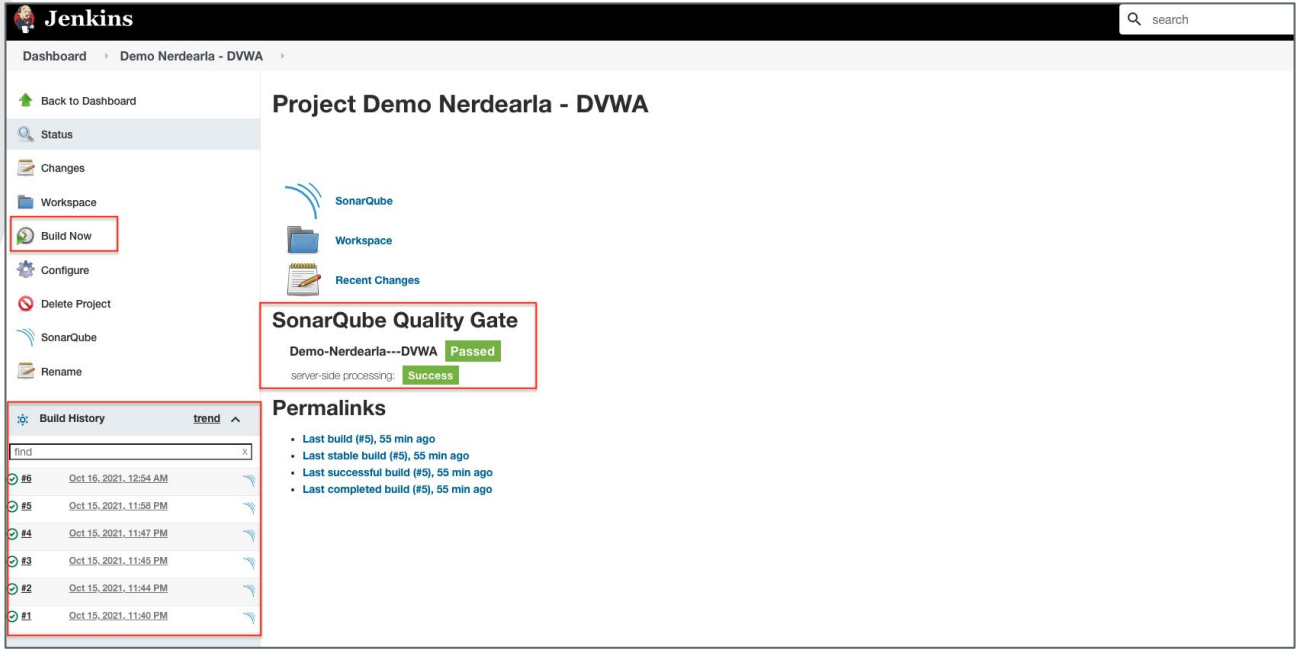
Jenkins + SonarQube + DefectDojo

Agregamos los datos del proyecto que generamos en SonarQube



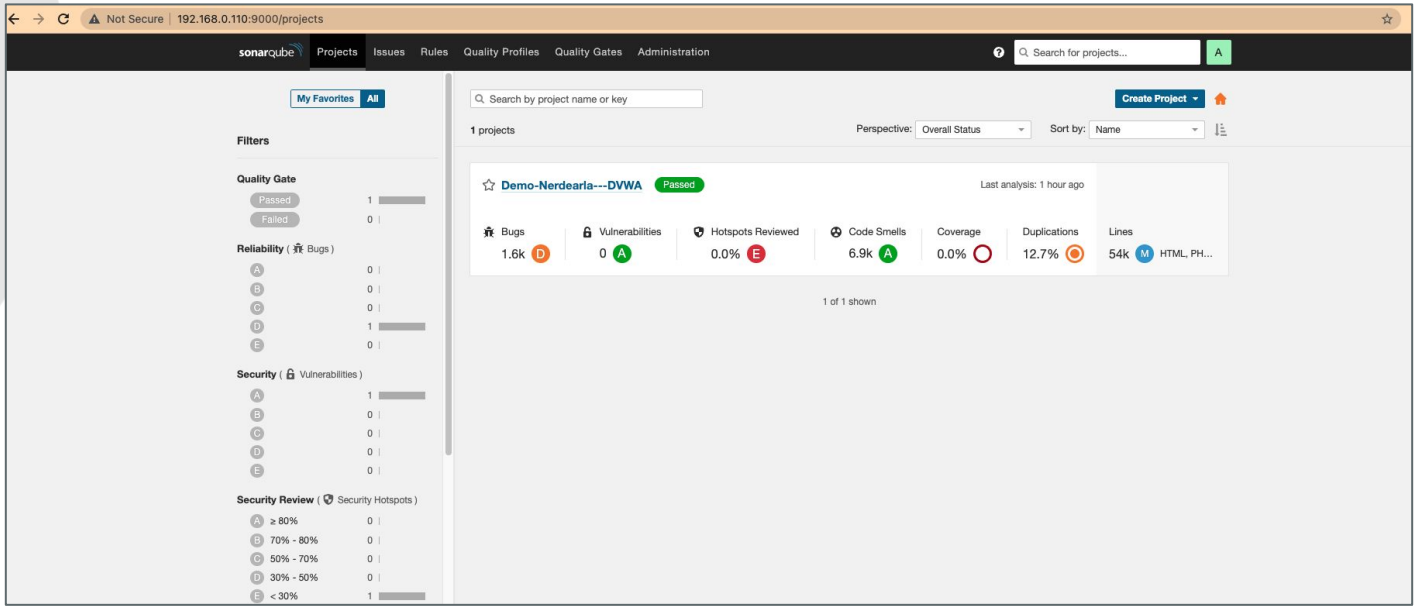
Jenkins + SonarQube + DefectDojo

Hacemos Click en “Build Now” y verificamos los resultados



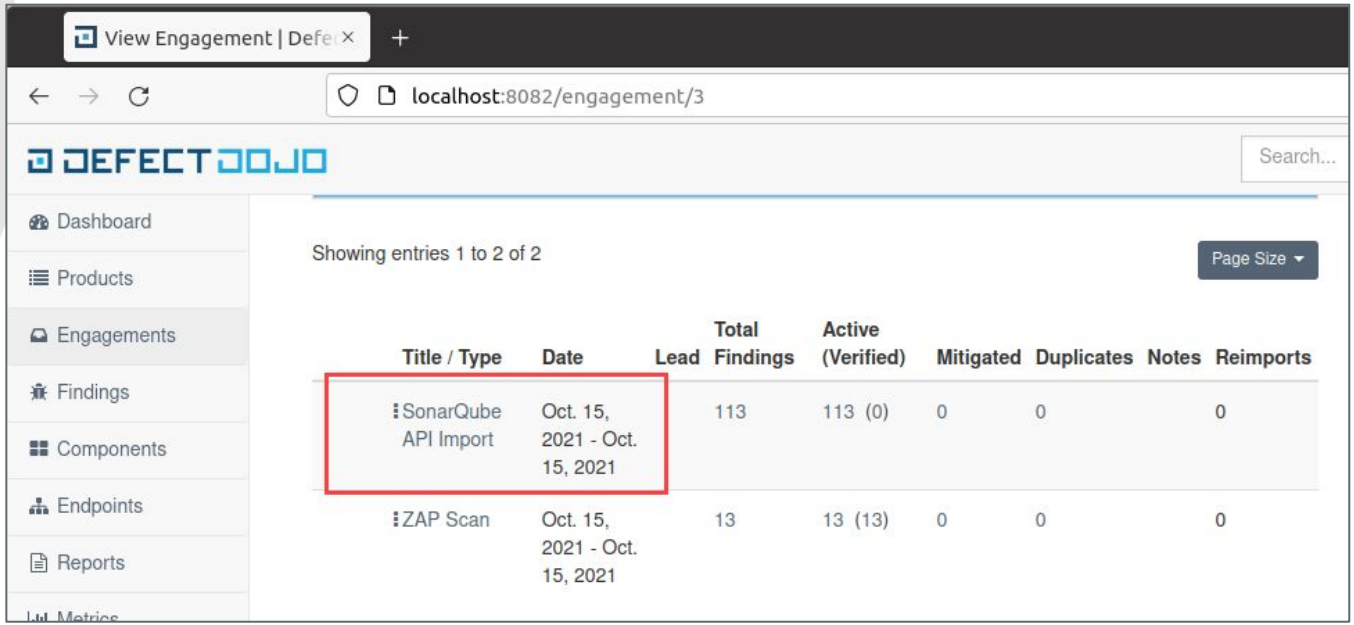
Jenkins + SonarQube + DefectDojo

Verificamos el resultado en SonarQube



Jenkins + SonarQube + DefectDojo

Se observa que el SCAN fue importado correctamente

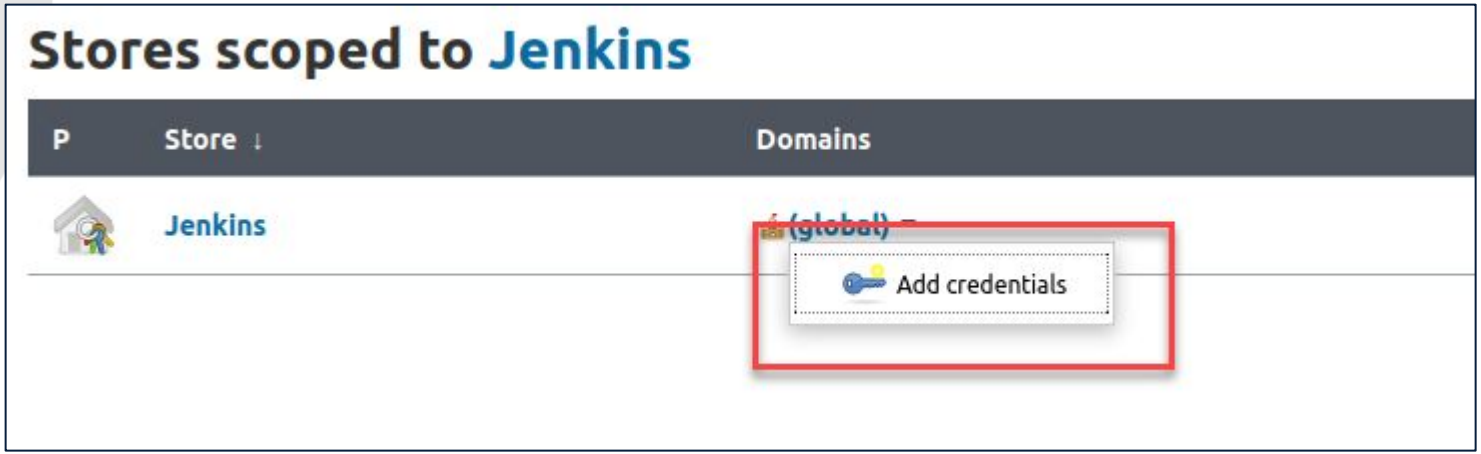


Pipeline

Pipeline

Agrego una nueva credencial

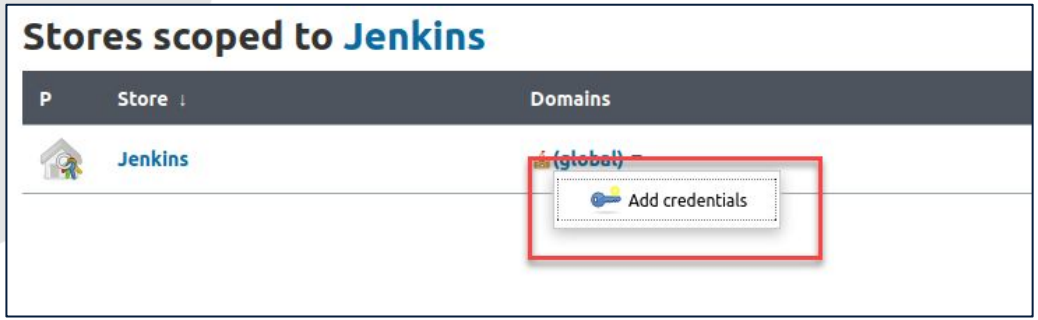
- ***Panel de control -> Administrar jenkis -> Manage credentials***



Pipeline

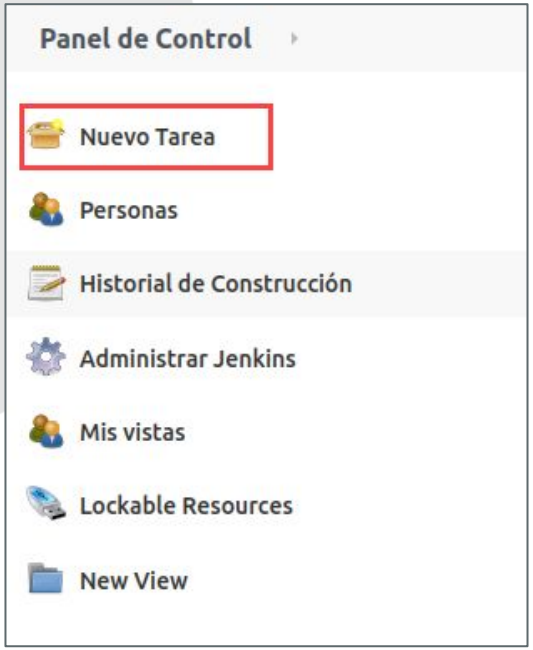
Agrego una nueva credencial

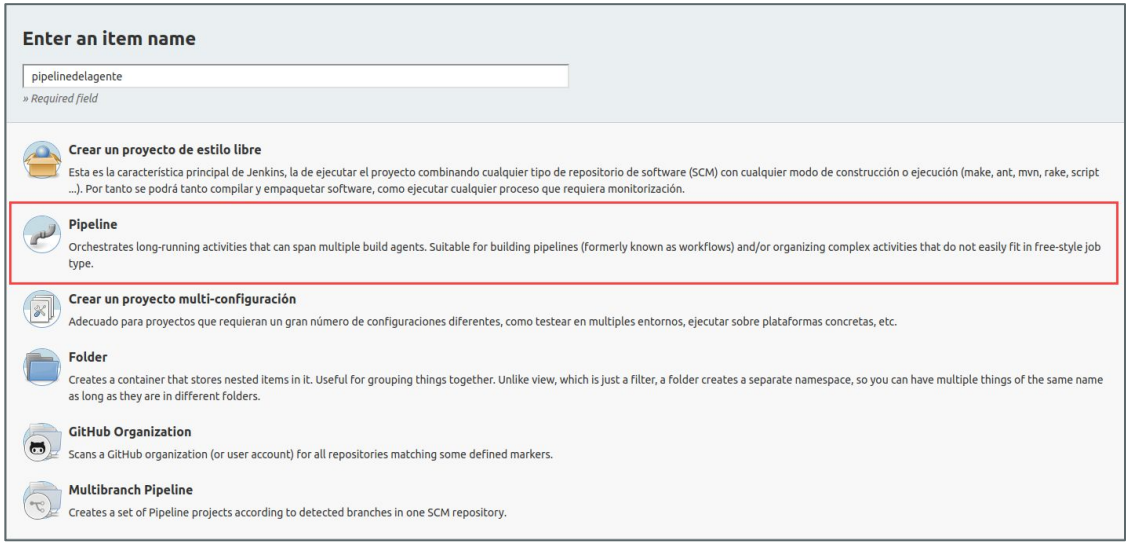
- ***Panel de control -> Administrar jenkis -> Manage credentials***



Pipeline

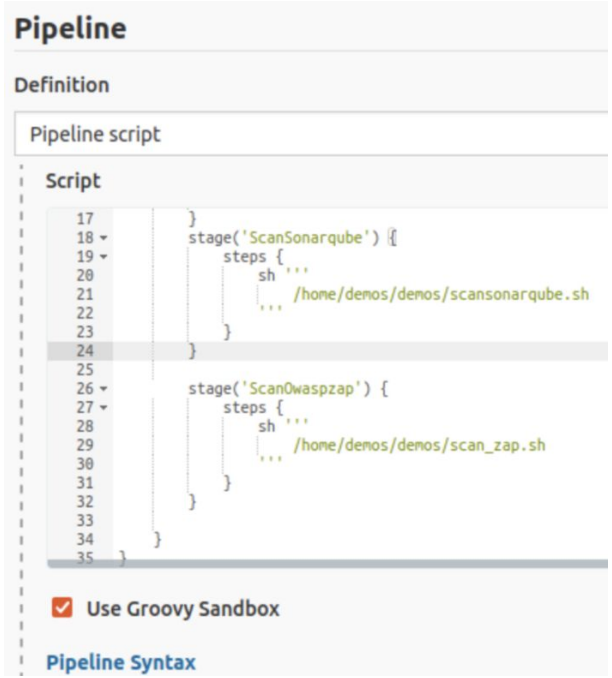
Debemos crear un pipeline para ejecutar los Scans de forma automatizada:





Pipeline

Debemos crear un pipeline e ingresar como secuencia lo siguiente:



Pipeline

En el pipeline definimos los siguientes stages:

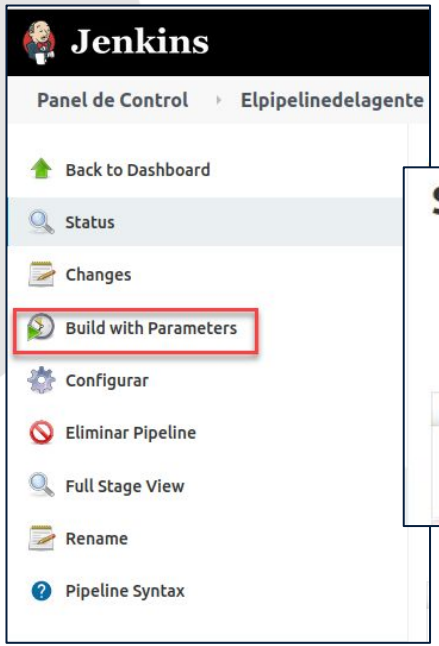
- `CrearProducto`
- `CrearEngagement`
- `ScanSonarqube`
- `ScanOwaspzap`

Para descargarlo ir a:

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

Pipeline

Ejecutamos el pipeline desde "*Build with Parameters*":



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