

DevSecOps en aplicaciones Docker y/o Kubernetes

Alumno

Abraham Santana Cebrián

Tutor

Micael Gallego Carrillo

Máster en ciberseguridad y privacidad

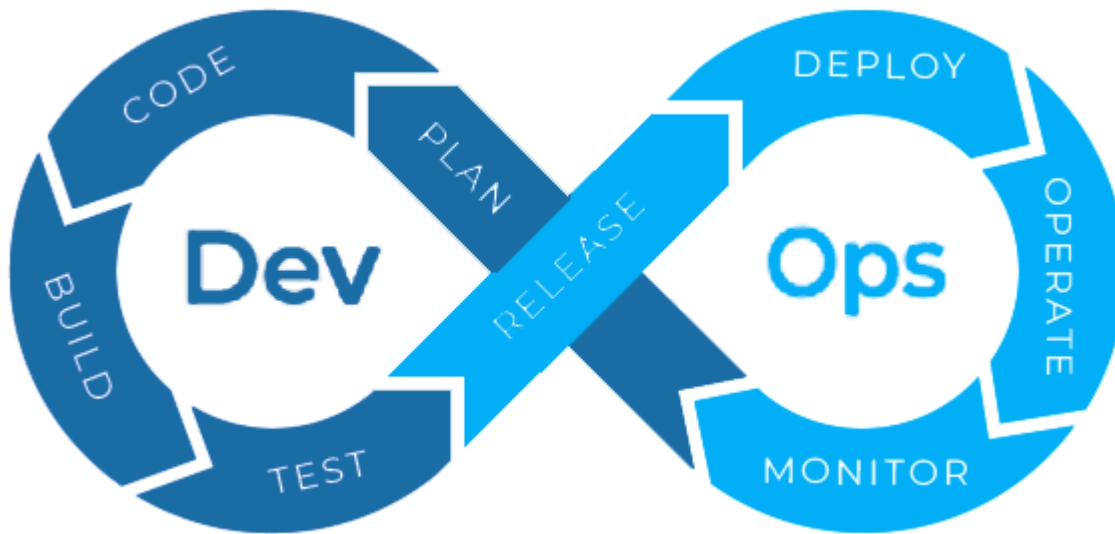
Directora

Marta Beltrán Pardo

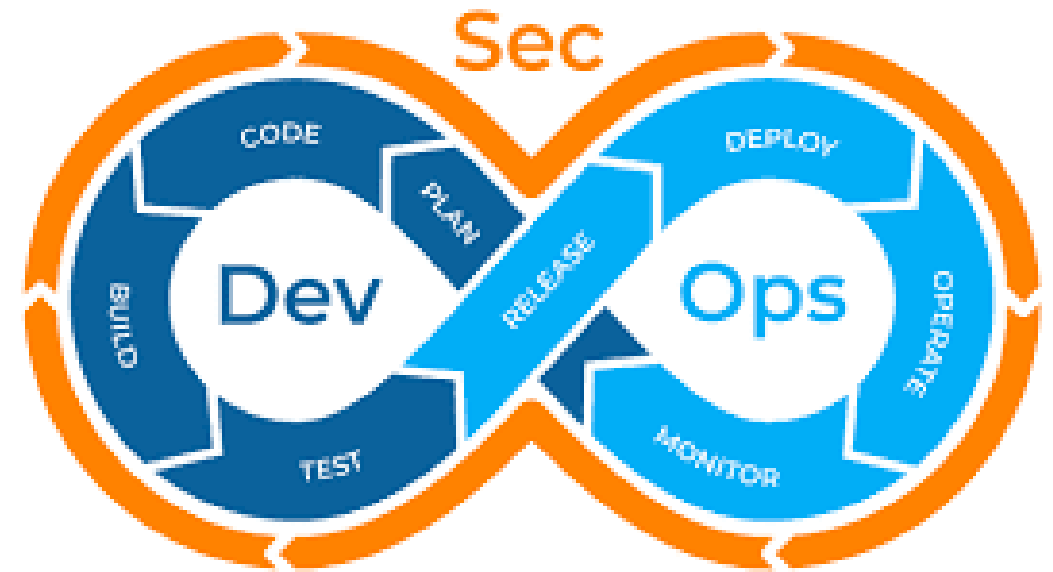
Índice

1. Introducción y contexto
 - Tema del Trabajo
 - Motivación y justificación
2. Organización
 - Objetivos
 - Metodología
 - Planificación
3. Estado del arte
 - Herramientas implementadas
4. Desarrollo del proyecto
 - Análisis
 - Diseño del Ciclo de Vida del Software DevSecOps
 - Implantación
 - Codificación y Pruebas
5. Resultados obtenidos
6. Dificultades encontradas
7. Conclusiones
8. Líneas de trabajo

- Migrar el entorno de Integración Continua / Despliegue Continuo (CI/CD) al proveedor AWS.
- Desacoplar el entorno CI/CD de una máquina física o virtualizada, ofreciendo las herramientas del entorno, para poder configurar y desplegar en AWS dicho ecosistema.

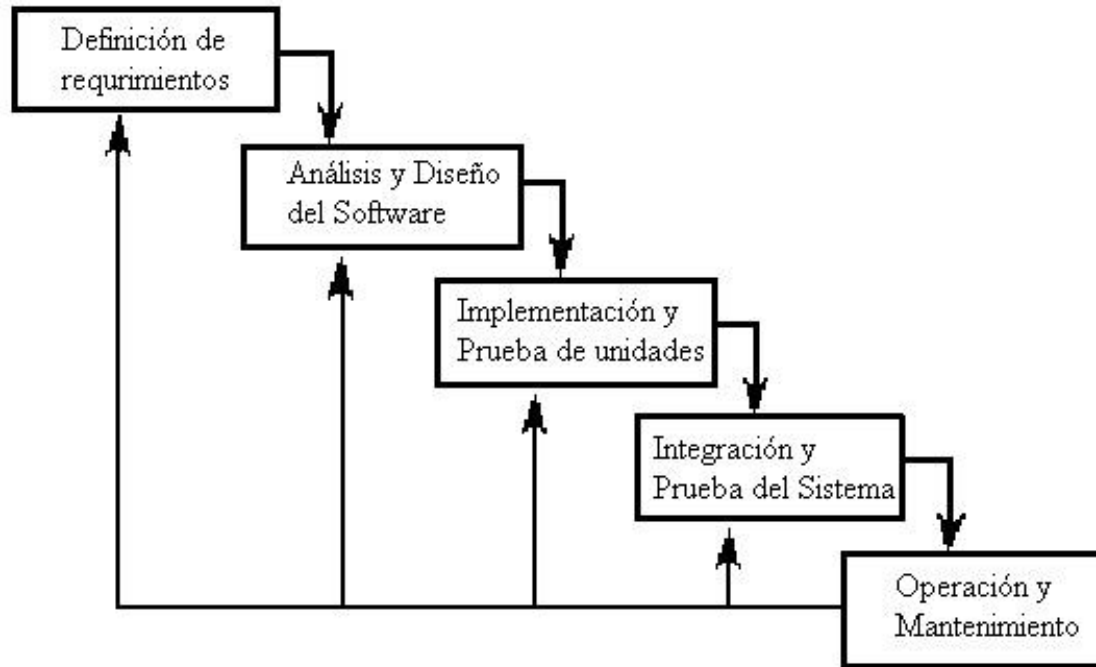


Esquema de funcionamiento de un ciclo DevOps

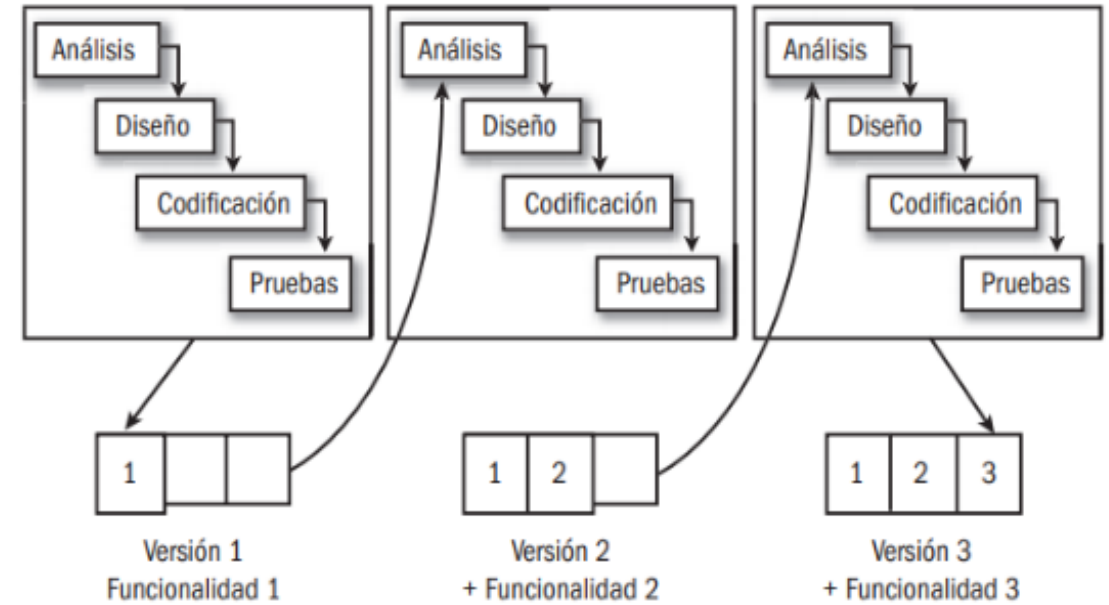


Esquema de funcionamiento de un ciclo DevSecOps

- Tema del Trabajo
- Motivación y justificación



Modelo cascada realimentado para el ciclo de vida



Modelo iterativo incremental para el ciclo de vida

- **montaje, estudio e implementación de un Pipeline de DevSecOps**, integrando en la implementación todas las herramientas de seguridad posibles, asumiendo que el proyecto de desarrollo software es una aplicación web ya desarrollada , además de posibilitar la implementación de herramientas del tipo análisis estático del código y dependencias), análisis dinámico de código, criterios de análisis de la seguridad, análisis de vulnerabilidades Docker, *pen testing*, etc.

- visión global de las herramientas, de las tecnologías utilizadas, y del sistema de integración de los elementos

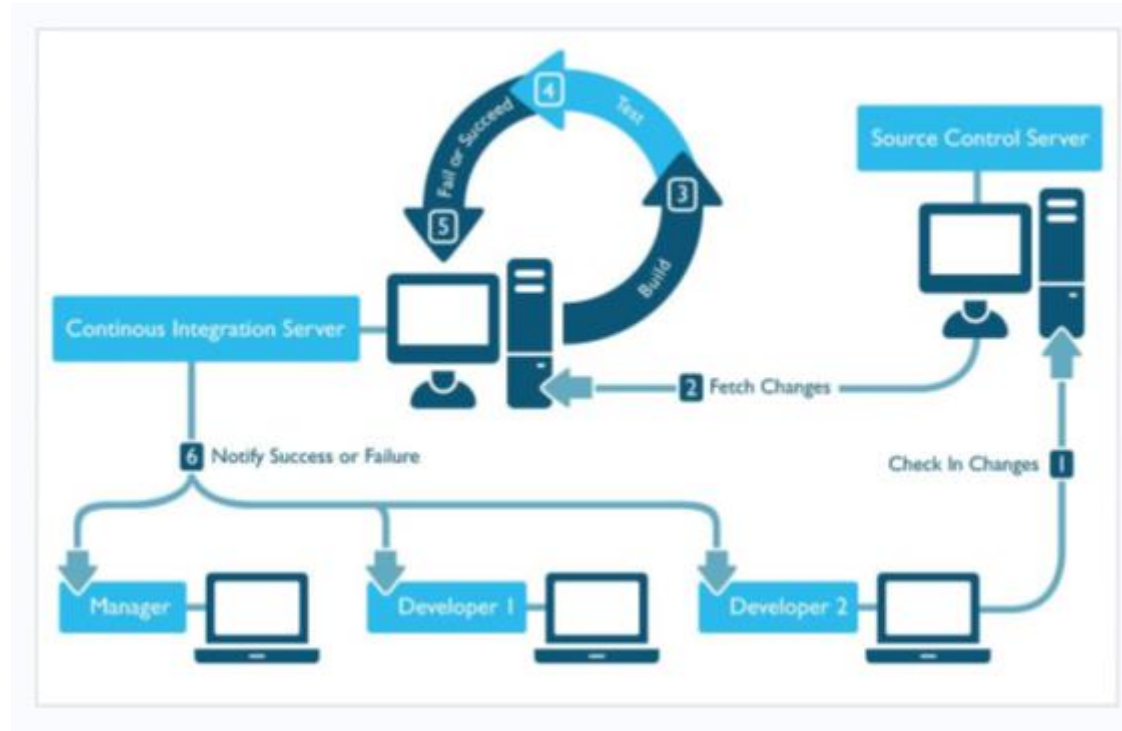
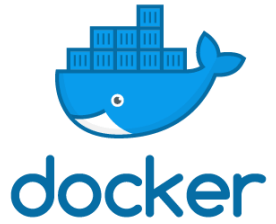
- infraestructura configurada a modo de tutorial, explicando cómo se monta un entorno de Continuous Integration / Continuous Deployment (CI/CD), y la configuración de las herramientas utilizadas en este proyecto

```
networks:
  - mynet
volumes:
  - jenkins_home:/var/jenkins_home
  - /var/run/docker.sock:/var/run/docker.sock
  - /usr/local/bin/docker:/usr/local/bin/docker
sonarqube:
  image: sonarqube:lts
  ports:
    - 9000:9000
  networks:
    - mynet
postgres:
  image: postgres:9.6
  restart: unless-stopped
  volumes:
    - /docker-compose-data/postgres-data:/var/lib/postgresql/data:rw
  environment:
    - POSTGRES_PASSWORD=ChangeMe
    - POSTGRES_USER=clair
    - POSTGRES_DB=clair
  networks:
    - mynet
clair:
  image: quay.io/coreos/clair:v2.0.6
  command: -config=/config/config.yaml
  #command: -log-level=debug
  ports:
    - 6060:6060
    - 6061:6061
  depends_on:
    - postgres
  volumes:
    - /docker-compose-data/clair-config:/config:ro
    - /docker-compose-data/clair-tmp:/tmp:rw
  user: root
  networks:
```

Animación con el script del Pipeline y YAML Docker-Compose



Jenkins

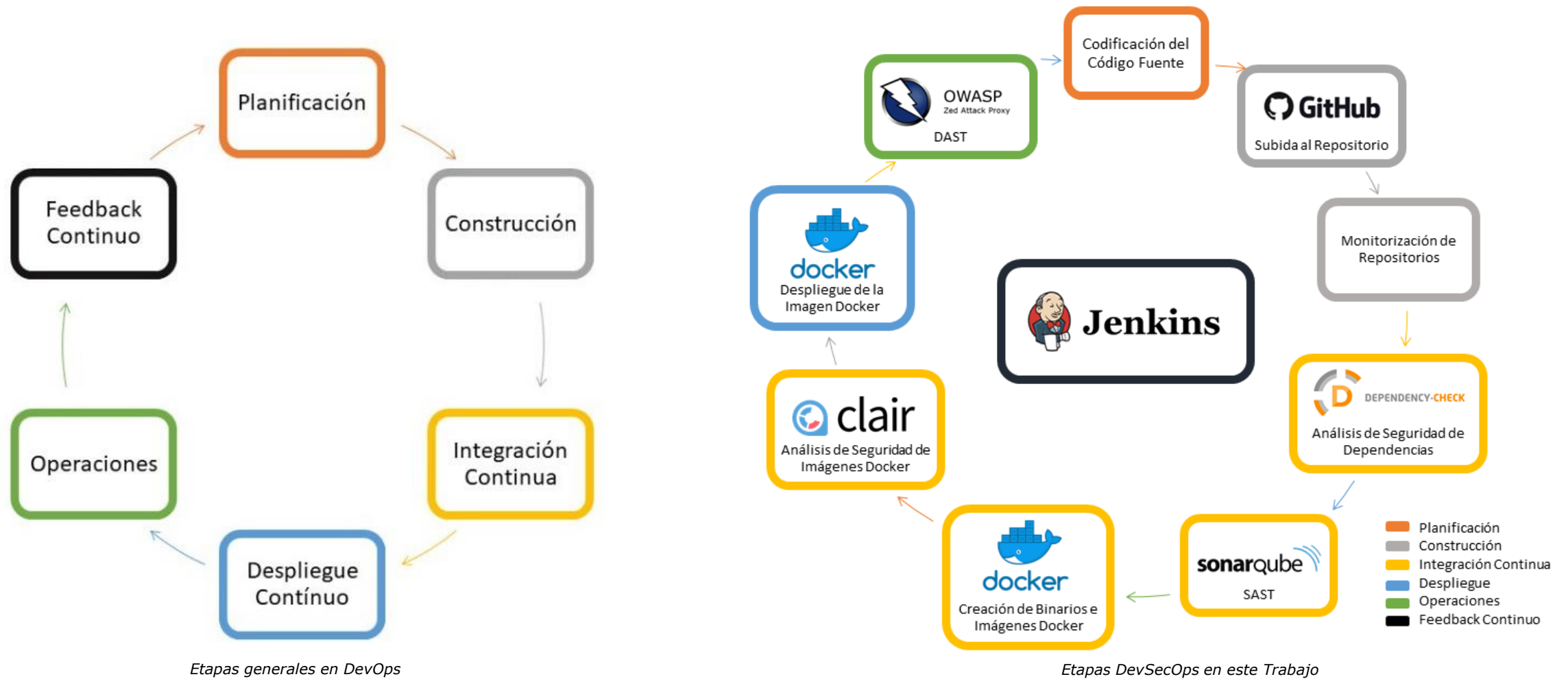


Esquema conceptual de Integración Continua y Despliegue Continuo



kubernetes





- Tema del Trabajo
- Motivación y justificación

- Objetivos

- Herramientas implementadas

- Diseño del Ciclo de Vida
- Implantación

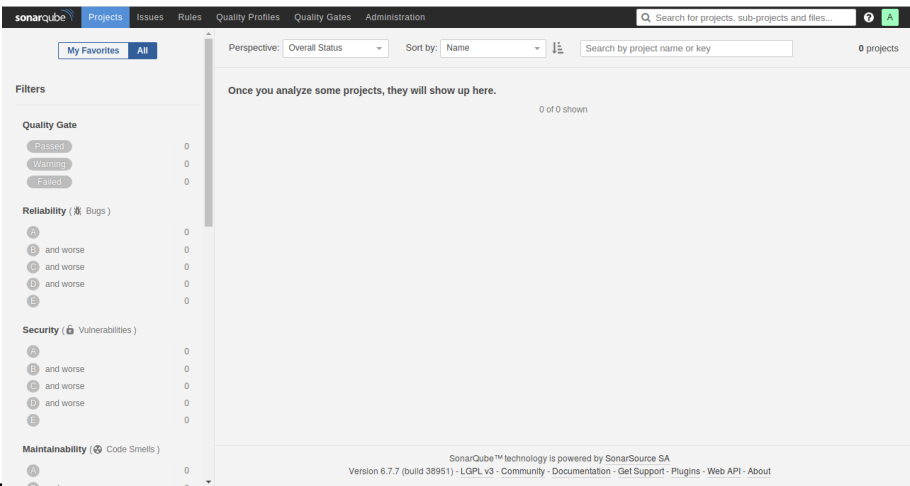
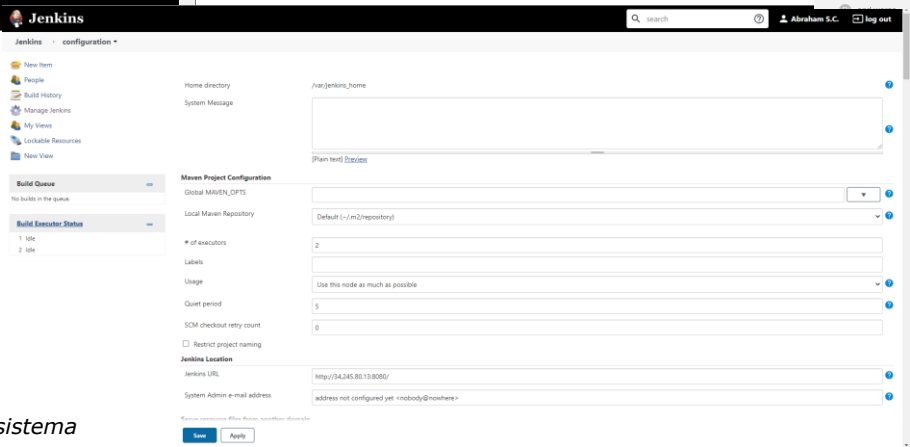
- Docker-Compose: orquestación del entorno DevSecOps

- Jenkins: configuración y puesta a punto

- SonarQube: configuración y puesta a punto

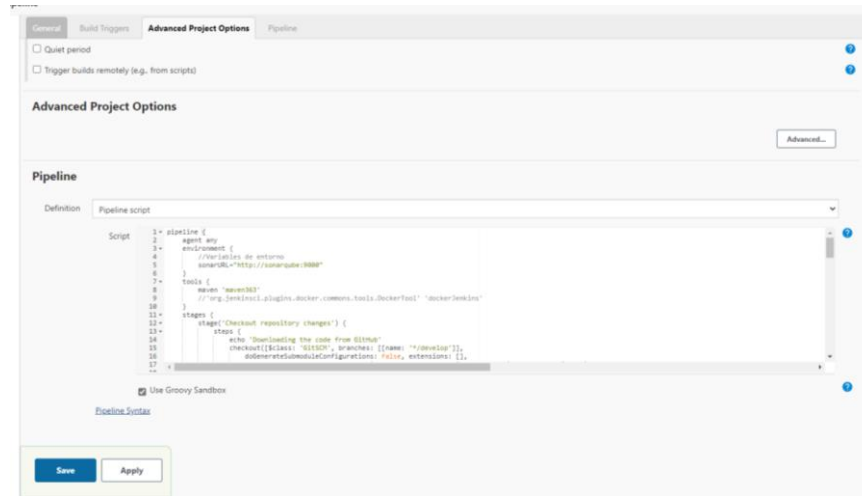
```
PuTTY (inactive)
create mode 100644 jenkins-dockerfile
create mode 100644 pipelineScripts/pipeline03dockerInstalledFromJenkins.txt
rename pipelineScript => pipelineScripts/pipelineScript (100%)
rename pipelineScript02.txt => pipelineScripts/pipelineScript02.txt (100%)
create mode 100644 pipelineScripts/pipelineScript03.txt
ubuntu@ip-172-31-43-98:~/18-19_absace$ docker-compose up
ERROR: Couldn't connect to Docker daemon at http+docker://localhost - is it running?

If it's at a non-standard location, specify the URL with the DOCKER_HOST environment variable.
ubuntu@ip-172-31-43-98:~/18-19_absace$ sudo docker-compose up
Creating network "18-19_absace_mynet" with driver "bridge"
Creating network "18-19_absace_default" with the default driver
Creating volume "18-19_absace_jenkins_home" with default driver
Creating volume "18-19_absace_clair-postgres" with default driver
Creating volume "18-19_absace_zap" with default driver
Building jenkins
Step 1/7 : from jenkins/jenkins:its
its: Pulling from jenkins/jenkins
3192219afd04: Pull complete
17c160265e75: Pull complete
cc4fe40d0e61: Pull complete
9d647f502aa07: Pull complete
d10b98e498aa1: Pull complete
1bfe919b8aa5: Pull complete
dafala7c0751: Pull complete
87e8abcf98a3: Pull complete
7bac295fef3b: Pull complete
b0c7fef45979: Pull complete
4d6fbc8189de: Pull complete
14eac212d69a: Pull complete
e027eb08ee87: Pull complete
```



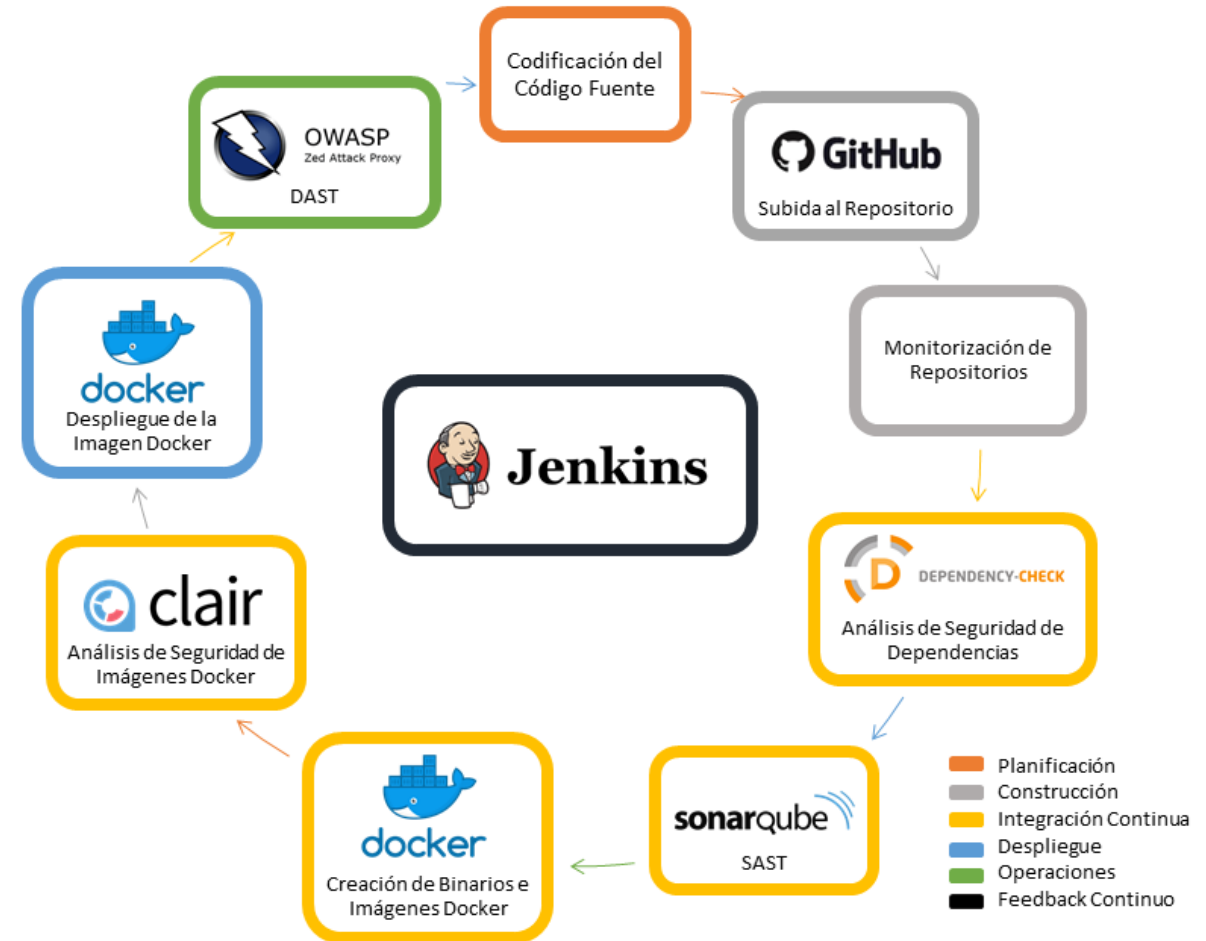
Imágenes de los resultados obtenidos en la implementación del ecosistema basado desplegado en Docker

Configuración del Pipeline



Imágenes de los resultados obtenidos durante la codificación de las pruebas.

Etapas del Pipeline



Introducción y contexto

- Tema del Trabajo
- Motivación y justificación

Organización

- Objetivos

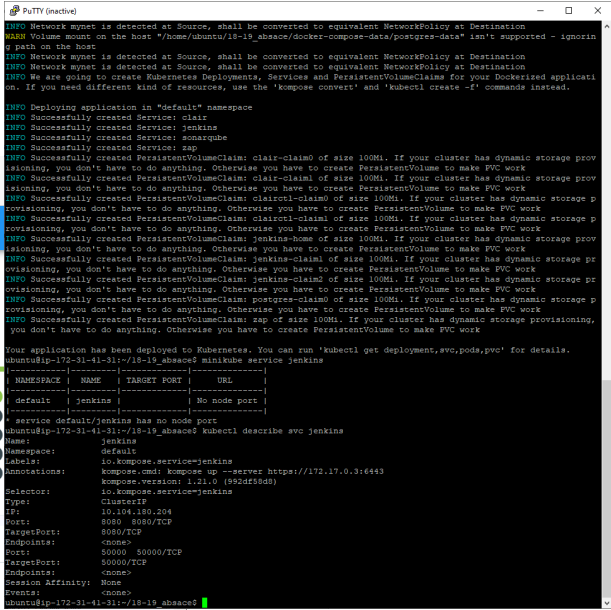
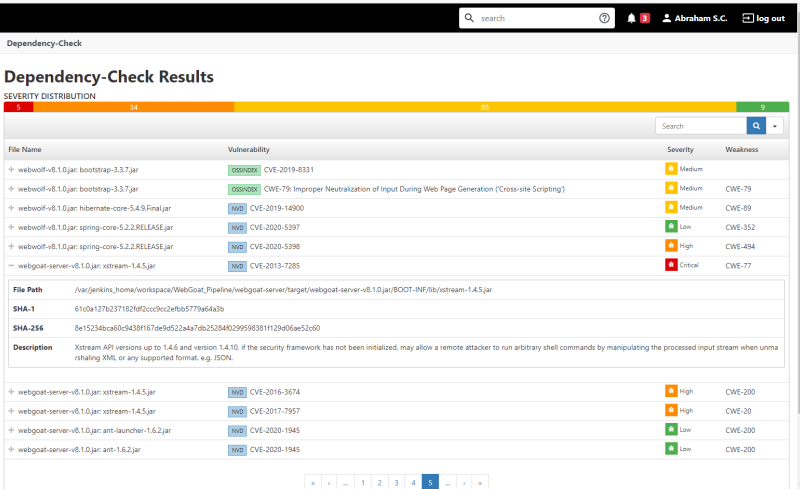
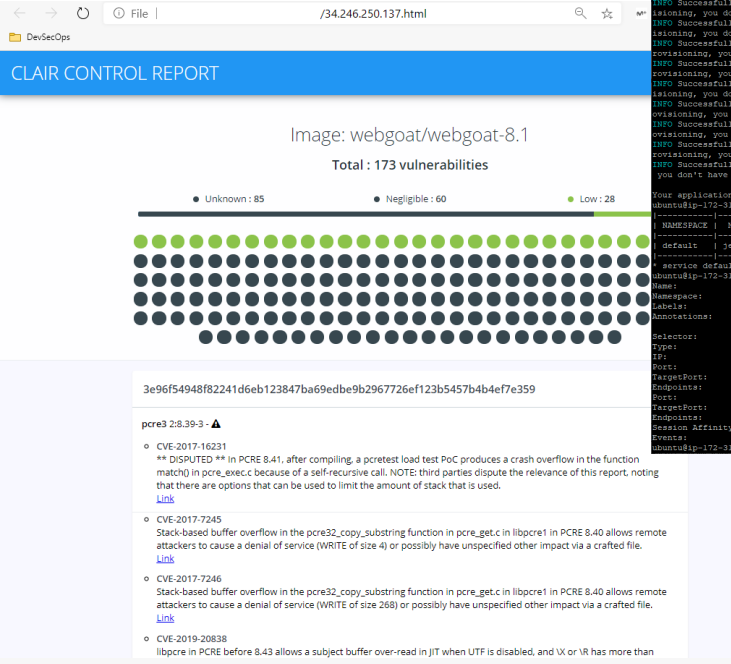
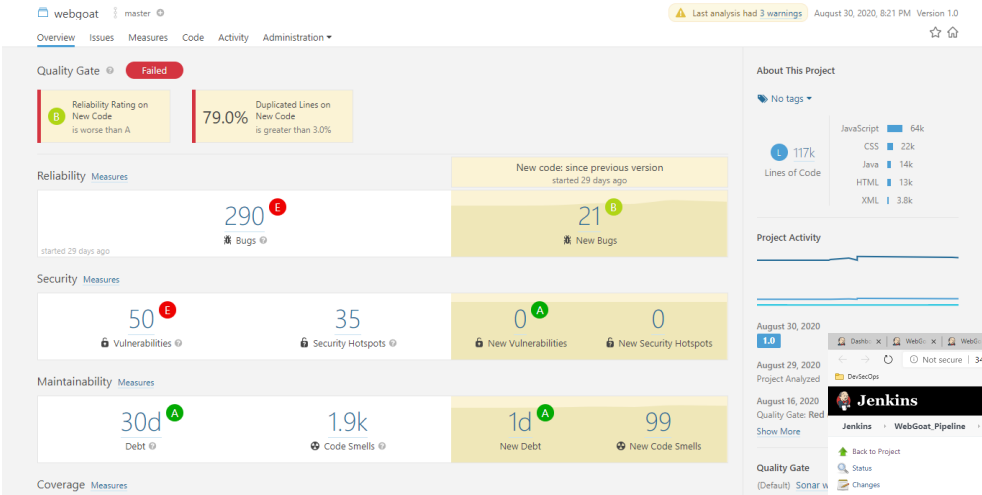
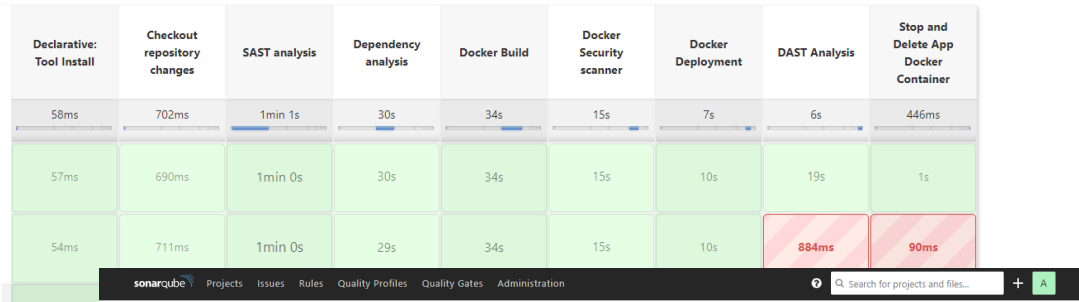
Estado del arte

- Herramientas implementadas

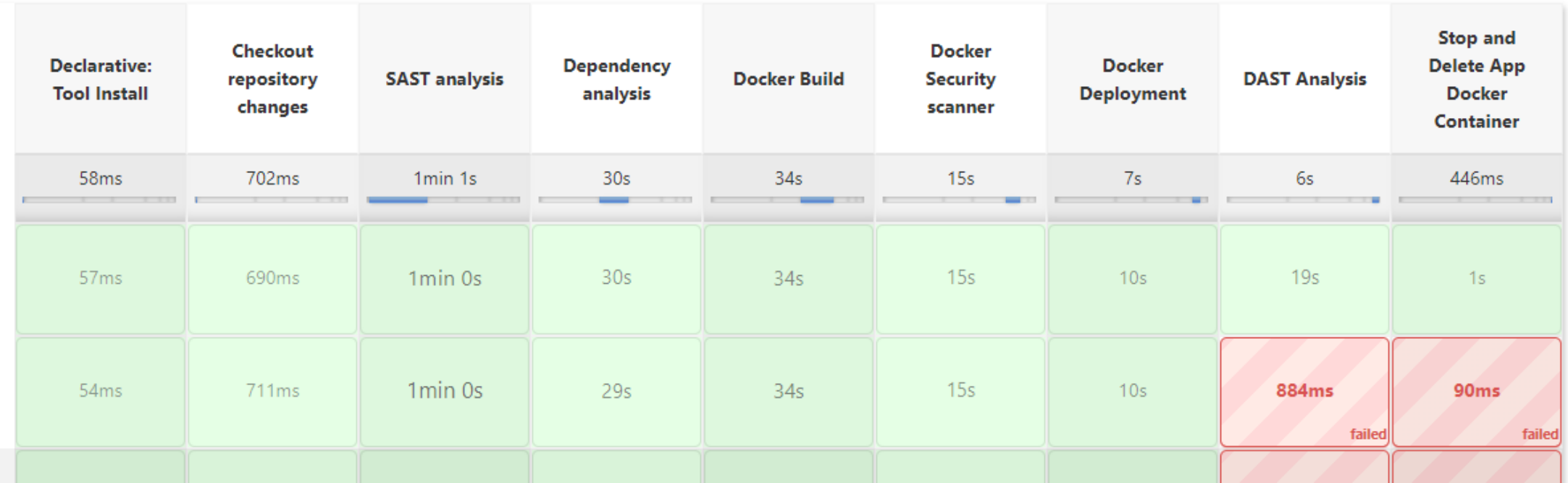
Estado del proyecto

- Diseño del Ciclo de Vida
- Implantación
- Codificación y Pruebas

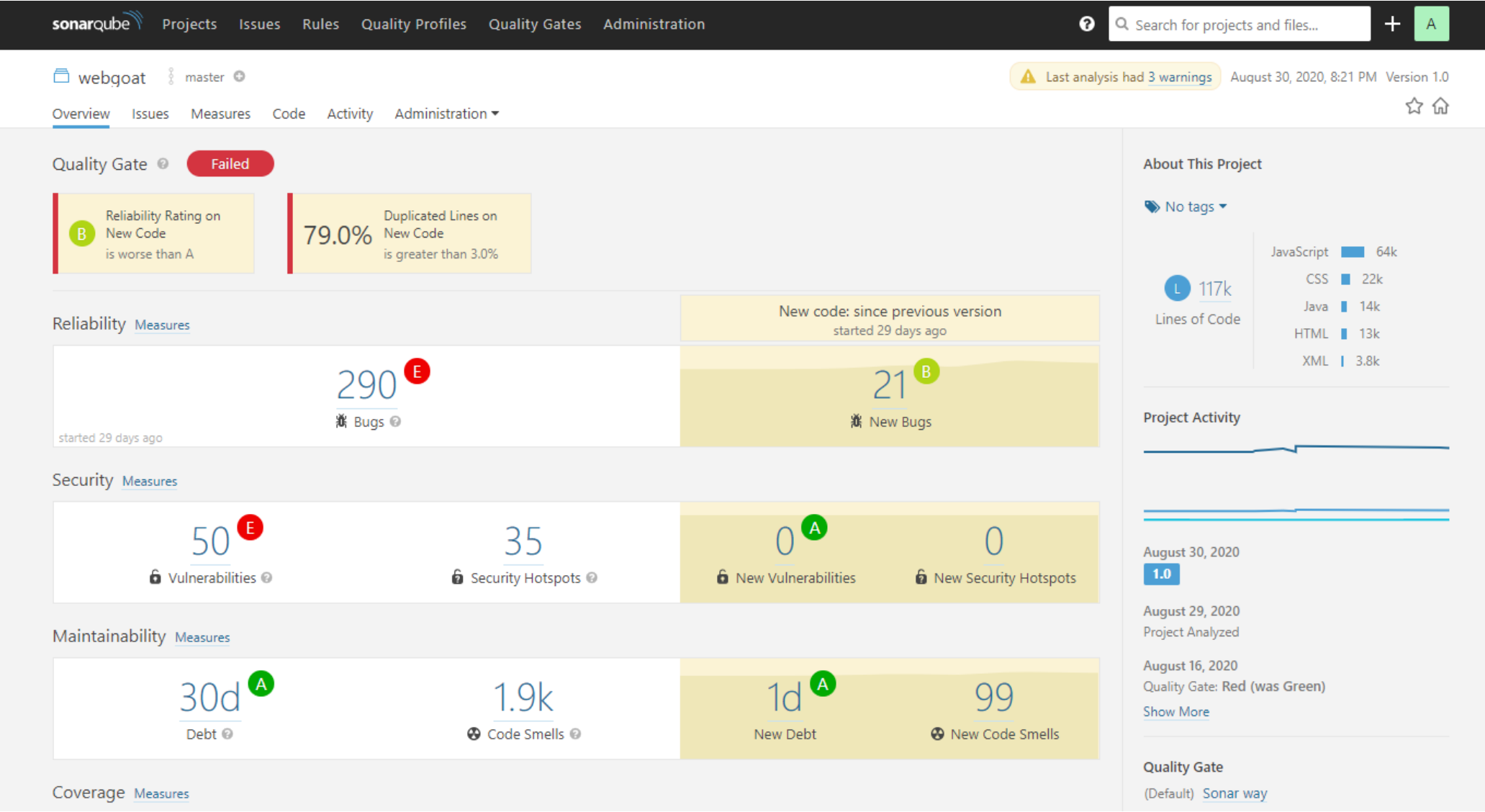
Resultados obtenidos



Imágenes de los resultados obtenidos en las diferentes etapas del Pipeline DevSecOps: Jenkins, Sonarqube, Dependency-Check y Clair de izq. A dcha.



Imágenes de los resultados obtenidos en las diferentes etapas del Pipeline
DevSecOps: Jenkins, Sonarqube, Dependency-Check y Clair de izq. A dcha.



Imágenes de los resultados obtenidos en las diferentes etapas del Pipeline
DevSecOps: Jenkins, Sonarqube, Dependency-Check y Clair de izq. A dcha.

Dashboard

WebGoat

WebGoat

Dependency-Check

Update

System

GitHub

Add the

Instances

Key pairs

AMIs

Cost M

Billing

Speed

bra1

New tab

Not secure | 34.246.250.137:8080/job/WebGoat_Pipeline/65/dependency-check-findings/

DevSecOps

Jenkins

search

3

Abraham S.C.

log out

Back to Project

Status

Changes

Console Output

View as plain text

Edit Build Information

Delete build '#65'

Git Build Data

No Tags

Dependency-Check

Restart from Stage

Replay

Pipeline Steps

Workspaces

Previous Build

Dependency-Check Results

SEVERITY DISTRIBUTION

5

34

85

9

Search

Q

File Name	Vulnerability	Severity	Weakness
+ webwolf-v8.1.0.jar: bootstrap-3.3.7.jar	OSSINDEX CVE-2019-8331	Medium	
+ webwolf-v8.1.0.jar: bootstrap-3.3.7.jar	OSSINDEX CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	Medium	CWE-79
+ webwolf-v8.1.0.jar: hibernate-core-5.4.9.Final.jar	NVD CVE-2019-14900	Medium	CWE-89
+ webwolf-v8.1.0.jar: spring-core-5.2.2.RELEASE.jar	NVD CVE-2020-5397	Low	CWE-352
+ webwolf-v8.1.0.jar: spring-core-5.2.2.RELEASE.jar	NVD CVE-2020-5398	High	CWE-494
- webgoat-server-v8.1.0.jar: xstream-1.4.5.jar	NVD CVE-2013-7285	Critical	CWE-77

File Path

/var/jenkins_home/workspace/WebGoat_Pipeline/webgoat-server/target/webgoat-server-v8.1.0.jar/BOOT-INF/lib/xstream-1.4.5.jar

SHA-1

61c0a127b237182fdf2ccc9cc2efbb5779a64a3b

SHA-256

8e15234bca60c9438f167de9d522a4a7db25284f0299598381f129d06ae52c60

Description

Xstream API versions up to 1.4.6 and version 1.4.10, if the security framework has not been initialized, may allow a remote attacker to run arbitrary shell commands by manipulating the processed input stream when unmarshaling XML or any supported format. e.g. JSON.

+ webgoat-server-v8.1.0.jar: xstream-1.4.5.jar	NVD CVE-2016-3674	High	CWE-200
+ webgoat-server-v8.1.0.jar: xstream-1.4.5.jar	NVD CVE-2017-7957	High	CWE-20
+ webgoat-server-v8.1.0.jar: ant-launcher-1.6.2.jar	NVD CVE-2020-1945	Low	CWE-200
+ webgoat-server-v8.1.0.jar: ant-1.6.2.jar	NVD CVE-2020-1945	Low	CWE-200

«

<

...

1

2

3

4

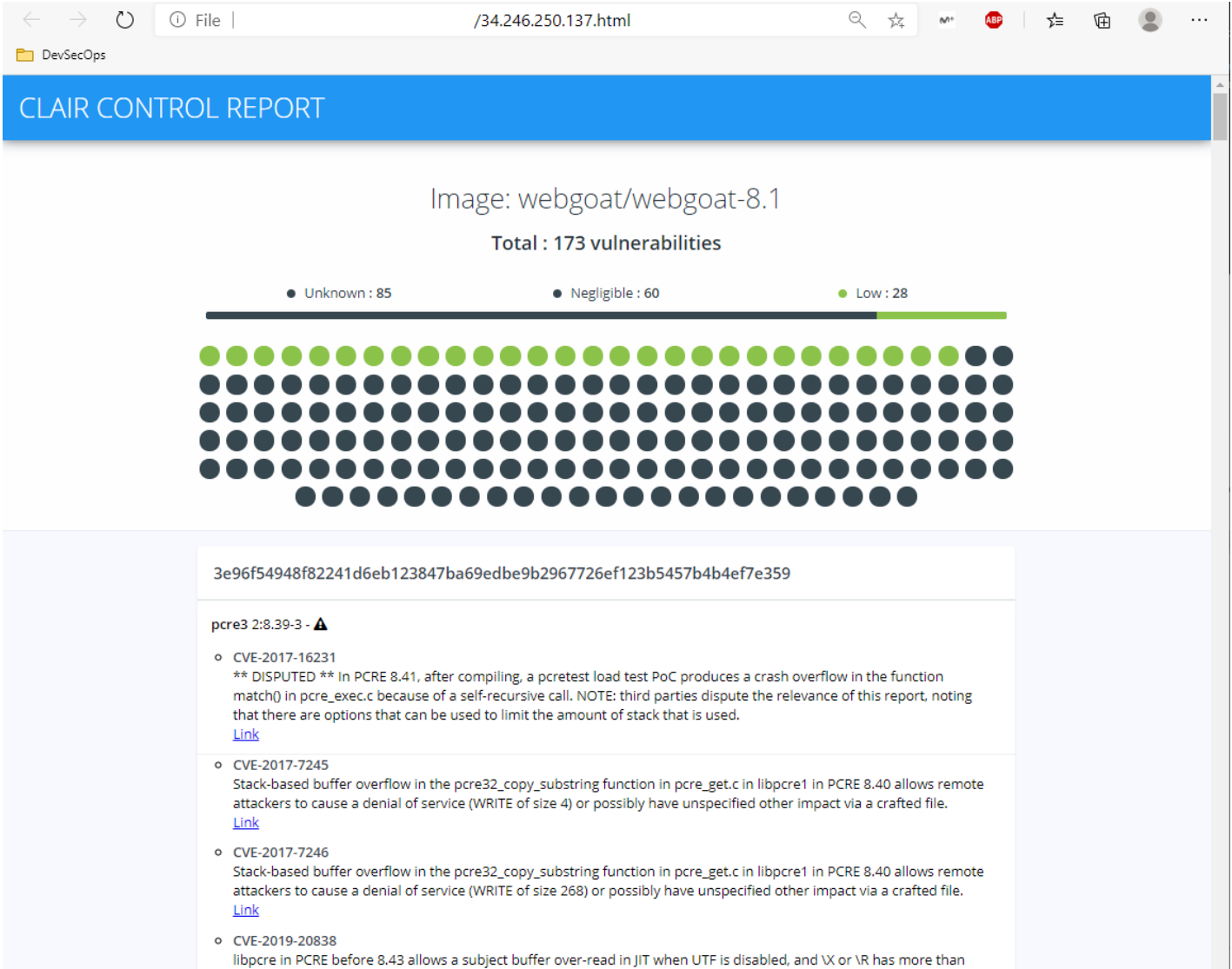
5

...

>

»

Imágenes de los resultados obtenidos en las diferentes etapas del Pipeline
DevSecOps: Jenkins, Sonarqube, Dependency-Check y Clair de izq. A dcha.



Imágenes de los resultados obtenidos en las diferentes etapas del Pipeline
DevSecOps: Jenkins, Sonarqube, Dependency-Check y Clair de izq. A dcha.

Introducción y contexto

- Tema del Trabajo
- Motivación y justificación

Organización

- Objetivos

Estado del arte

- Herramientas implementadas

Desarrollo del proyecto

- Diseño del Ciclo de Vida
- Implantación
- Codificación y Pruebas

Resultados obtenidos

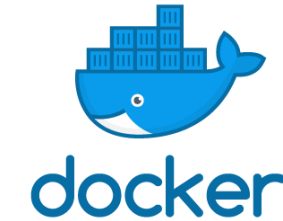
```
PutTY (inactive)
INFO Network mynet is detected at Source, shall be converted to equivalent NetworkPolicy at Destination
WARN Volume mount on the host "/home/ubuntu/18-19_absace/docker-compose-data/postgres-data" isn't supported - ignorin
g path on the host
INFO Network mynet is detected at Source, shall be converted to equivalent NetworkPolicy at Destination
INFO Network mynet is detected at Source, shall be converted to equivalent NetworkPolicy at Destination
INFO We are going to create Kubernetes Deployments, Services and PersistentVolumeClaims for your Dockerized applicati
on. If you need different kind of resources, use the 'kompose convert' and 'kubectl create -f' commands instead.

INFO Deploying application in "default" namespace
INFO Successfully created Service: clair
INFO Successfully created Service: jenkins
INFO Successfully created Service: sonarqube
INFO Successfully created Service: zap
INFO Successfully created PersistentVolumeClaim: clair-claim0 of size 100Mi. If your cluster has dynamic storage prov
isioning, you don't have to do anything. Otherwise you have to create PersistentVolume to make PVC work
INFO Successfully created PersistentVolumeClaim: clair-claim1 of size 100Mi. If your cluster has dynamic storage prov
isioning, you don't have to do anything. Otherwise you have to create PersistentVolume to make PVC work
INFO Successfully created PersistentVolumeClaim: clairctl-claim0 of size 100Mi. If your cluster has dynamic storage p
rovisioning, you don't have to do anything. Otherwise you have to create PersistentVolume to make PVC work
INFO Successfully created PersistentVolumeClaim: clairctl-claim1 of size 100Mi. If your cluster has dynamic storage p
rovisioning, you don't have to do anything. Otherwise you have to create PersistentVolume to make PVC work
INFO Successfully created PersistentVolumeClaim: jenkins-home of size 100Mi. If your cluster has dynamic storage prov
isioning, you don't have to do anything. Otherwise you have to create PersistentVolume to make PVC work
INFO Successfully created PersistentVolumeClaim: jenkins-claim1 of size 100Mi. If your cluster has dynamic storage pr
ovisioning, you don't have to do anything. Otherwise you have to create PersistentVolume to make PVC work
INFO Successfully created PersistentVolumeClaim: jenkins-claim2 of size 100Mi. If your cluster has dynamic storage pr
ovisioning, you don't have to do anything. Otherwise you have to create PersistentVolume to make PVC work
INFO Successfully created PersistentVolumeClaim: postgres-claim0 of size 100Mi. If your cluster has dynamic storage p
rovisioning, you don't have to do anything. Otherwise you have to create PersistentVolume to make PVC work
INFO Successfully created PersistentVolumeClaim: zap of size 100Mi. If your cluster has dynamic storage provisioning,
you don't have to do anything. Otherwise you have to create PersistentVolume to make PVC work

Your application has been deployed to Kubernetes. You can run 'kubectl get deployment,svc,pods,pvc' for details.
ubuntu@ip-172-31-41-31:~/18-19_absace$ minikube service jenkins
|-----|-----|-----|-----|
| NAMESPACE | NAME | TARGET PORT | URL |
|-----|-----|-----|-----|
| default | jenkins | | No node port |
|-----|-----|-----|-----|
* service default/jenkins has no node port
ubuntu@ip-172-31-41-31:~/18-19_absace$ kubectl describe svc jenkins
Name: jenkins
Namespace: default
Labels: io.kompose.service=jenkins
Annotations: kompose.cmd: kompose up --server https://172.17.0.3:6443
             kompose.version: 1.21.0 (992df58d8)
Selector: io.kompose.service=jenkins
Type: ClusterIP
IP: 10.104.180.204
Port: 8080 8080/TCP
TargetPort: 8080/TCP
Endpoints: <none>
Port: 50000 50000/TCP
TargetPort: 50000/TCP
Endpoints: <none>
Session Affinity: None
Events: <none>
ubuntu@ip-172-31-41-31:~/18-19_absace$
```

Imágenes de los resultados obtenidos en las diferentes etapas del Pipeline
DevSecOps: Jenkins, Sonarqube, Dependency-Check y Clair de izq. A dcha.

- Implementación de la herramienta Docker dentro de la imagen y contenedor Jenkins utilizado en este Trabajo



- Error de codificación del fichero Dockerfile para la empaquetación de la app WebGoat en una imagen Docker



- Fichero de configuración en Clair e integración de Clair y Clairctl en el ecosistema DevSecOps dockerizado



- Incompatibilidades en la migración del ecosistema DevSecOps hacia Kubernetes utilizando la herramienta Kompose



- implementar un sistema CI/CD teniendo en cuenta los requisitos y las políticas de seguridad, así como la implementación de seguridad SAST, DAST, análisis de dependencias, análisis de imágenes Docker, nos proporcionan un nivel de información muy elevado sobre la calidad del producto software que se está desplegando, así como los riesgos de seguridad que tiene su entrega al mercado
- la capacidad de recursos del Host donde se despliegue este sistema es limitante
- el desarrollo de un proyecto software con un Pipeline DevSecOps será mucho más sencilla que con un simple DevOps si se logra integrar herramientas en flujo de Pipeline de Jenkins
- la importancia de añadir la seguridad en todas las etapas, resaltando la idea de que seguridad y desarrollo son incompatibles

```

networks:
  - mynet
volumes:
  - jenkins_home:/var/jenkins_home
  - /var/run/docker.sock:/var/run/docker.sock
  - /usr/local/bin/docker:/usr/local/bin/docker
sonarqube:
  image: sonarqube:lts
  ports:
    - 9000:9000
  networks:
    - mynet
postgres:
  image: postgres:9.6
  restart: unless-stopped
  volumes:
    - ./docker-compose-data/postgres-data:/var/lib/postgresql/data:rw
  environment:
    - POSTGRES_PASSWORD=ChangeMe
    - POSTGRES_USER=clair
    - POSTGRES_DB=clair
  networks:
    - mynet
clair:
  image: quay.io/coreos/clair:v2.0.6
  command: -config=/config/config.yaml
  #command: -log-level=debug
  ports:
    - 6060:6060
    - 6061:6061
  depends_on:
    - postgres
  volumes:
    - ./docker-compose-data/clair-config:/config:ro
    - ./docker-compose-data/clair-tmp:/tmp:rw
  user: root
  networks:

```

- Realizar un estudio y la integración adecuada de la herramienta Notary, la cual se utiliza para analizar y asegurar que las imágenes Docker que se están utilizando han sido firmadas, comprobando así su integridad, confidencialidad, y disponibilidad.
- Realizar un estudio e implementación más completo del realizado en este trabajo para el análisis DAST utilizando la herramienta OWASP ZAP, ya que no se han conseguido resultados lo suficientemente claros en este Trabajo.
- Realizar un estudio y mejora continua del script del Pipeline, ya siempre es posible encontrar mejoras en el rendimiento y funcionalidad de este, siguiendo la metodología DevSecOps.
- Realizar un estudio necesario para la migración y adaptación de todo el ecosistema DevSecOps creado en este Trabajo a un entorno Kubernetes, analizando la viabilidad de hacer uso de la herramienta Kompose u otras herramientas.



notary



OWASP
Zed Attack Proxy



Jenkins



kubernetes

DevSecOps en aplicaciones Docker y/o Kubernetes

Demo del ecosistema y ejecución del Pipeline DevSecOps

<https://youtu.be/HkFP6tAobp0>

¿Preguntas?

a.santanac.2018@alumnos.urjc.es

Alumno

Abraham Santana Cebrián

Tutor

Micael Gallego Carrillo

Máster en ciberseguridad y privacidad

Directora

Marta Beltrán Pardo



DevSecOps en aplicaciones Docker y/o Kubernetes

Muchas gracias por su atención

Alumno

Abraham Santana Cebrián

Tutor

Micael Gallego Carrillo

Máster en ciberseguridad y privacidad

Directora

Marta Beltrán Pardo

