

Documentación Tarea1

Instalación de herramientas en servidor Centos

docker-compose.yml utilizado para inicializar un contenedor con Jenkins

```
version: "3"
services:
  jenkins:
    image: jenkins/jenkins
    container_name: jenkins
    ports:
      - "8080:8080"
    volumes:
      - "./jenkins:/var/jenkins_home"
      - "/var/run/docker.sock:/var/run/docker.sock"
      - "/root/.ssh:/root/.ssh"
    user: root
```

docker-compose.yml utilizado para inicializar un contenedor con SonarQube y uno con Postgresql

```
version: "3"
services:
  sonarqube:
    image: sonarqube:lt
    restart: unless-stopped
    environment:
      - SONARQUBE_JDBC_USERNAME=sonarqube
      - SONARQUBE_JDBC_PASSWORD=sonarpass
      - SONARQUBE_JDBC_URL=jdbc:postgresql://db:5432/sonarqube
    ports:
      - "9000:9000"
      - "9092:9092"
    volumes:
      - sonarqube_conf:/opt/sonarqube/conf
      - /sonar/sonar_data/conf/sonar.properties:/opt/sonarqube/conf/sonar.properties
      - sonarqube_data:/opt/sonarqube/data
      - sonarqube_extensions:/opt/sonarqube/extensions
      - sonarqube_bundled-plugins:/opt/sonarqube/lib/bundled-plugins
  db:
    image: postgres:9
    restart: unless-stopped
    environment:
      - POSTGRES_USER=sonarqube
      - POSTGRES_PASSWORD=sonarpass
      - POSTGRES_DB=sonarqube
    volumes:
      - sonarqube_db:/var/lib/postgresql
      - postgresql_data1:/var/lib/postgresql/data
volumes:
  postgresql_data1:
  sonarqube_bundled-plugins:
  sonarqube_conf:
  sonarqube_data:
  sonarqube_db:
  sonarqube_extensions:
```

En ambos casos se inicializan los aplicativos con el comando `docker-compose up -d` ejecutado en el directorio donde se encuentra el archivo `docker-compose.yml` de cada herramienta.

Configuración acceso externo TLS para Docker desde el servidor donde corre Jenkins

Para poder utilizar contenedores como nodos para las construcciones configuré acceso TLS al Docker instalado en el host donde se ejecuta Docker ("Docker in Docker"), se crean los certificados TLS (cliente/servidor) y se configuran los siguientes archivos para permitir dicho acceso:

/etc/systemd/system/docker.service.d/override.conf

```
[Service]
EnvironmentFile=-/etc/sysconfig/docker
ExecStart=
ExecStart=/usr/bin/dockerd $DOCKER_OPTS
```

/etc/sysconfig/docker

```
DOCKER_OPTS=" -H unix:///var/run/docker.sock -H 0.0.0.0:2376 --tlsverify --tlscacert=/jenkins/docker-tls-certs/ca.pem --tlscert=/jenkins/docker-tls-certs/server-cert.pem --tlskey=/jenkins/docker-tls-certs/server-key.pem"
```

Configuración nube Docker en Jenkins

Se creó una credencial del tipo "X.509 Client Certificate" utilizando los certificados que se configuraron para aceptar conexiones externas de Docker

Configure Clouds

Docker

Name ?

docker

Docker Host URI ?

tcp://192.168.248.85:2376

Server credentials

docker_cert

Add

Version = 20.10.9, API Version = 1.41

☒ Enabled

Avanzado...

Test Connection

Configuración Docker Agent Template en Jenkins

Se creó un agente Docker el cual contiene Java 8 (necesario para la compilación del proyecto Maven). La única particularidad de este agente es que utiliza claves SSH las cuales están agregadas en Github para poder publicar los tags y realizar el upgrade de versión en el release.

Docker Agent templates

Docker Agent templates

Labels ?

java-8

☒ Enabled

Name ?

docker

Docker Image ?











openjdk:8-jdk

Registry Authentication...

Credenciales

Se crearon credenciales en Jenkins para utilizar desde el pipeline

Credentials that should be available irrespective of domain specification to requirements matching.

	ID	Name	Kind	Description	
	remote-key	remote_user (Clave de laboratorio jenkins)	SSH Username with private key	Clave de laboratorio jenkins	
	github-pem	ignaciolopez409 (Clave para clonar repos de GitLab)	SSH Username with private key	Clave para clonar repos de GitLab	
	docker_cert	docker_cert	X.509 Client Certificate		
	dockerhubCredentials	ignaciolopezventimiglia/***** (Credenciales para publicar imagenes en hub.docker)	Username with password	Credenciales para publicar imagenes en hub.docker	
	tarea	tarea	Secret text		

Pipeline

Se configuró una tarea la cual solicita parámetros:

- Módulo a construir (orders-service-example para este caso)
- Tarea (build/reléase).

Pipeline tarea1

Esta ejecución requiere parámetros adicionales:

modulo

orders-service-example ▾

Se debe indicar el módulo a construir

tarea

build ▾

build: ejecuta mvn clean package y publica imagen docker de la versión en SNAPSHOT
release: ejecuta mvn release:prepare release:perform y publica la imagen docker de la versión cerrada

Ejecución

La tarea está configurada para obtener el pipeline del repositorio GitHub

Pipeline

Definition

Pipeline script from SCM ▾

SCM

Git ▾

Repositories

Repository URL

git@github.com:ignaciolopez409/tarea1.git

Credentials

ignaciolopez409 (Clave para clonar repos de GitLab) ▾ ➡ Add ▾

Avanzado...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

*/main

Script Path

pipelines/pipeline_tarea1

☒ Lightweight checkout

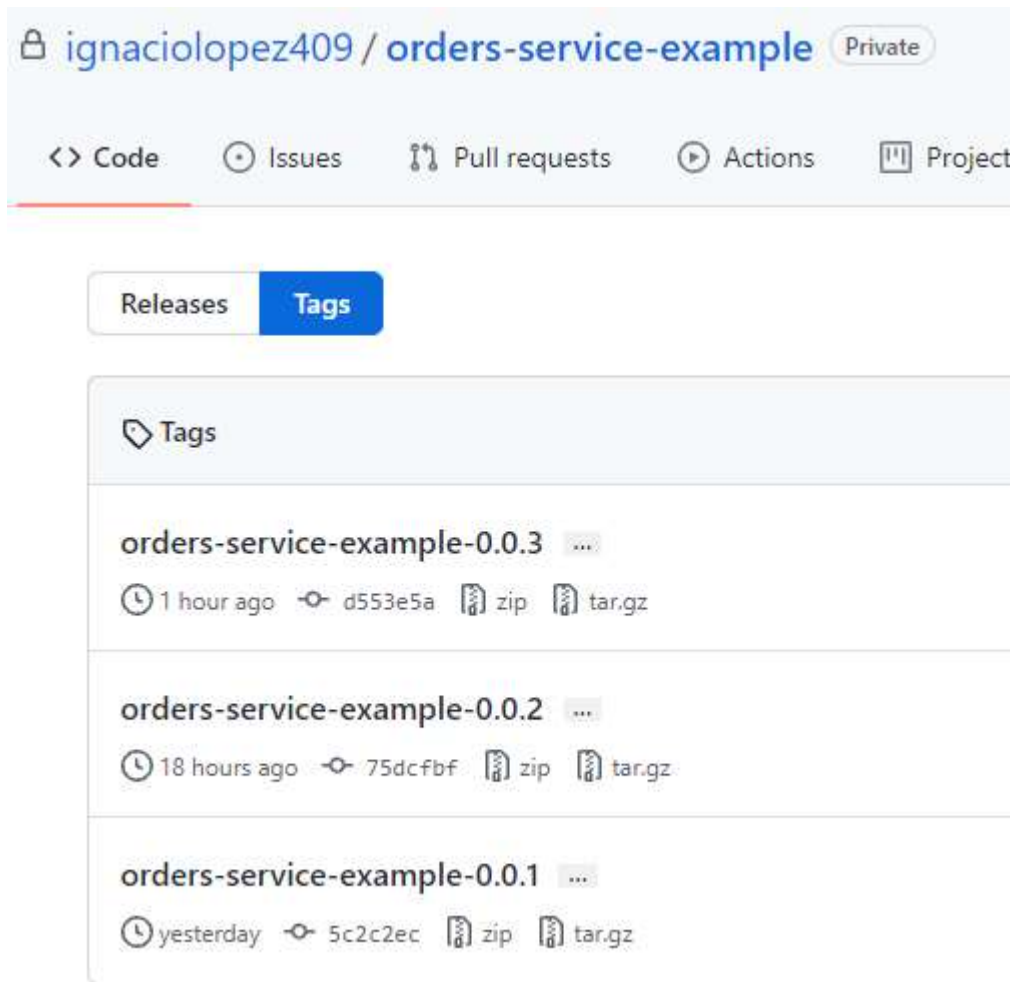
Pipeline Syntax

Evidencias

Orders Service Example

En el siguiente repositorio se puede ver el microservicio utilizado, el incremento de versión luego de cada reléase y el tag de versiones realizado

<https://github.com/ignaciolopez409/orders-service-example>



Imágenes Docker publicadas en Docker Hub luego de cada build

<https://hub.docker.com/r/ignaciolopezventimiglia/orders-service-example/tags>

Resultado de análisis de código realizado con SonarQube en el build

