
SciLifeLab

EXPLORANDO LA SEGURIDAD Y ORQUESTACIÓN EN LA NUBE CON KUBERNETES

ALVARO REVUELTA

SciLifeLab

¿QUIÉN SOY?

Alvaro Revuelta.

- Egresado de la ETSIINF-UPM.
- Viviendo 3 años en Suecia después de estudiar ahí.
- Trabajando como desarrollador (backend) en SciLifeLab.

SciLifeLab

SCILIFELAB

Laboratorio de ciencias de la vida y biológicas.

1. Data Centre - Enfoque en el desarrollo de soluciones IT para avanzar la investigación.
2. DDS - Data delivery System - Sistema para enviar de forma segura información sensible entre investigadores.
3. Todos los proyectos son Open Source y compromiso con Open Data.

SciLifeLab

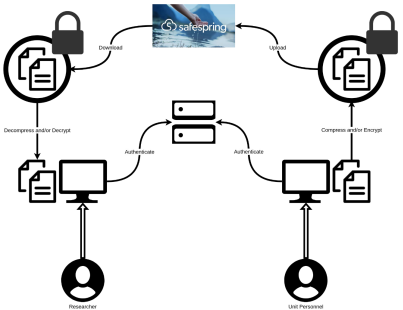
ESTRUCTURA

1. Introducción.
2. Orquestación en Kubernetes.
3. DevOps y GitOps.
4. La seguridad en DDS.
5. Otros proyectos en SciLifeLab.
6. Final - Preguntas.

SciLifeLab

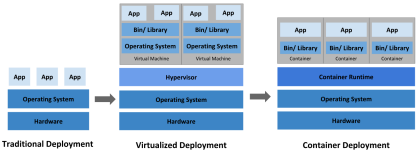
1. INTRODUCCIÓN - DATA DELIVERY SYSTEM

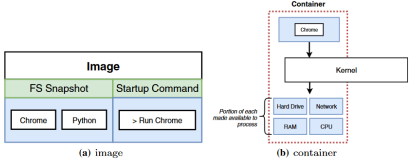
- Los datos permanecen dentro de Suecia en todo momento.
- El sistema se encarga del cifrado y gestión de claves.
- NO es una solución de almacenamiento.



SciLifeLab

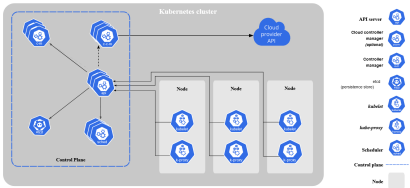
1. INTRODUCCIÓN - CONTENEDORES





1. INTRODUCCIÓN - KUBERNETES

- Coordinación y orquestación de aplicaciones compuestas por contenedores.
- Lenguaje declarativo (archivos YAML).



SciLifeLab

1. INTRODUCCION - CI/CD & GITOPS.

- CI/CD es una serie de prácticas para automatizar el proceso de integrar cambios en el codigo.
- Implica construir y probar **automáticamente** cambios de código a través del sistema de control de versiones (**Git**). Y desplegar automáticamente todos los cambios después de pasar por un pipeline de testing/pruebas.

- GitOps es una metodología para gestionar y automatizar la infraestructura y los despliegues de aplicaciones utilizando Git como **Single source of truth**.

SciLifeLab

1. INTRODUCCION - FUNDAMENTOS DE GIT.

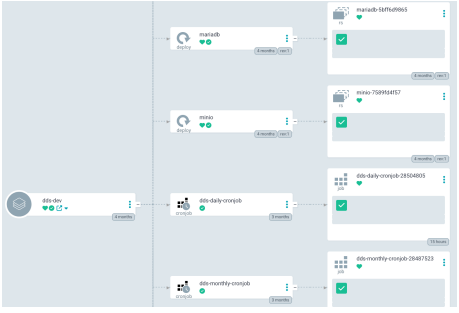
- Un "commit" en Git representa un conjunto de **cambios** realizados en los archivos del proyecto. Cada commit tiene un mensaje descriptivo que resume los cambios realizados.
- Las **ramas** permiten trabajar en paralelo en diferentes características sin afectar la rama principal ("master" o "main").
- Al combinar los commits con la capacidad de ramificar, se facilita el seguimiento de los cambios y la gestión del historial del proyecto.

- Los **Pull Requests** (PR) son una característica clave de los sistemas de control de versiones.
- Un PR es una solicitud que un desarrollador envía para fusionar (**merge**) sus cambios de una rama a otra (generalmente la rama principal).
- Las PR facilitan la revisión del código y la discusión sobre los cambios propuestos.

SciLifeLab

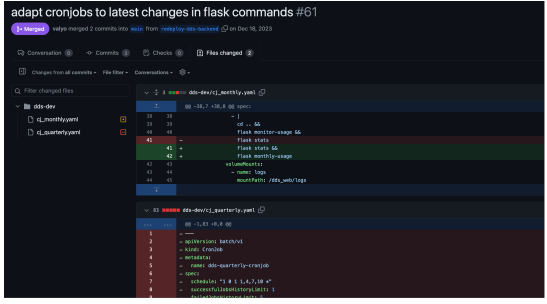
2. ORQUESTACIÓN EN KUBERNETES

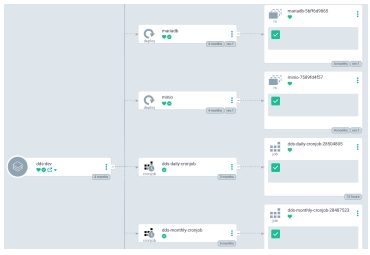
- **ArgoCD** es una herramienta para GitOps en aplicaciones de K8s.
- Se conecta con un repositorio (gitHub) y para cada cambio **automáticamente** actualiza los recursos necesarios.
- Monitorización con **OpenSearch** (alertas y errores) y **Grafana** (recursos).



SciLifeLab

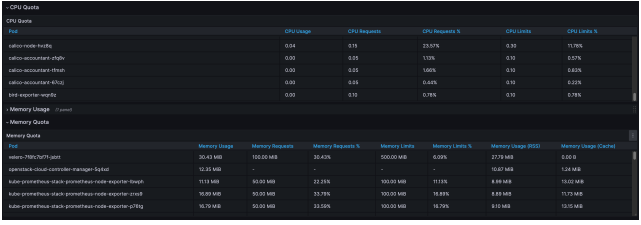
2. - ORQUESTACIÓN EN KUBERNETES





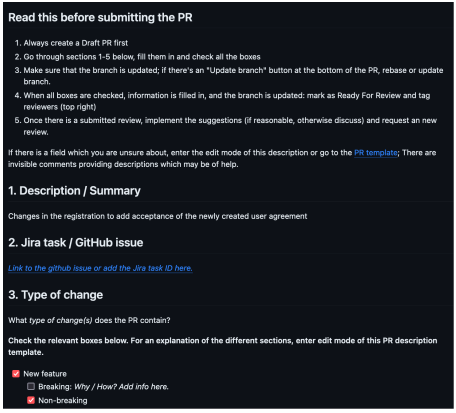
SciLifeLab

2. - ORQUESTACIÓN EN KUBERNETES



3. DEVOPS Y GITOPS

- Branching Strategy: Por cada cambio, nueva rama (**Branch**) y **Pull Request** (PR)
- GitHub Actions: Testing automático.
- Se trabaja en la nueva rama, que será **borrada** al hacer merge.



Read this before submitting the PR

1. Always create a Draft PR first
2. Go through sections 1-5 below, fill them in and check all the boxes
3. Make sure that the branch is updated; if there's an "Update branch" button at the bottom of the PR, rebase or update branch.
4. When all boxes are checked, information is filled in, and the branch is updated: mark as Ready For Review and tag reviewers (top right)
5. Once there is a submitted review, implement the suggestions (if reasonable, otherwise discuss) and request an new review.

If there is a field which you are unsure about, enter the edit mode of this description or go to the PR template; There are invisible comments providing descriptions which may be of help.

1. Description / Summary

Changes in the registration to add acceptance of the newly created user agreement

2. Jira task / GitHub issue

Link to the github issue or add the Jira task ID here.

3. Type of change

What *type of change(s)* does the PR contain?

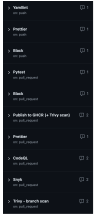
Check the relevant boxes below. For an explanation of the different sections, enter edit mode of this PR description template.

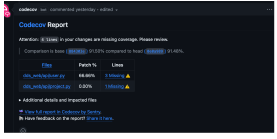
- [x] New feature
  - [ ] Breaking: *Why / How? Add info here.*
  - [x] Non-breaking

SciLifeLab

3. DEVOPS Y GITOPS

- Por cada PR, una suite de tests serán ejecutados.
- Si alguno falla, la PR será **bloqueada**.

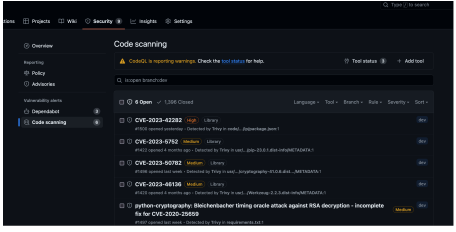




SciLifeLab

3. DEVOPS Y GITOPS

- Algunas acciones solo se activarán **después** del merge. Algunas buscarán continuamente **vulnerabilidades de seguridad**.
- La última fase construirá una nueva imagen con el tag de la versión que se específica y la publicará en el repositorio de **Docker Hub**.
  Ver dds_web/blob/dev/.github/workflows/publish_and_trivyscan.yml



RESUMEN HASTA AHORA

Dos repositorios:

1. El principal, en el que el código base de la aplicación se mantiene. Técnica de branching y PR. Automatización con acciones y tests. Publicación de las imágenes con la aplicación dockerizada.
2. Un segundo repositorio conectado con ArgoCD que contiene los YAML para Kubernetes. Y al que se le informa que la aplicación ha cambiado.

SciLifeLab

4. SEGURIDAD Y CIFRADOS

- Sistema de clave pública-privada por cada usuario. La clave privada se cifra con la contraseña.
- Cuando un proyecto se crea, un nuevo par es generado. La clave privada del projecto se cifra con la clave publica del usuario.
- Por tanto, la clave privada del proyecto solo se puede descifrar con la clave privada del usuario, que a su vez se deriva de la contraseña del usuario.

SciLifeLab

4. SEGURIDAD Y CIFRADOS.

- Para las claves del projecto el algoritmo utilizado es **X25519** Curva elíptica de Diffie Helman - Paquete "Crypthopgraphy" de Python.
  Enlace
- La subsecuente encriptación de la clave privada generada se realiza con el algoritmo RSA.
  Enlace.

```python
def generate_project_key_pair(user, project):
    private_key = asymmetric.x25519.X25519PrivateKey.generate()
    private_key_bytes = private_key.private_bytes(
        encoding=serialization.Encoding.Raw,
        format=serialization.PrivateFormat.Raw,
        encryption_algorithm=serialization.NoEncryption(),
    )
    public_key_bytes = private_key.public_key().public_bytes(
        encoding=serialization.Encoding.Raw,
        format=serialization.PublicFormat.Raw,
    )
    project.public_key = public_key_bytes
    for unit_user in user.unit.users:
        _init_and_append_project_user_key(unit_user, project, private_key_bytes)
    del private_key_bytes
    del private_key
    gc.collect()
```

SciLifeLab

4. SEGURIDAD Y CIFRADOS.

- Para acceder a la aplicación, un **token temporal** se tiene que generar mediante **2FA**.
- Primero se pregunta por el usuario y contraseña, una vez correctos, se envía un código a través de email o authenticator app y se genera un token con una duración limitada.
- En este caso es un **JWT** (JSON Web Token).

- El cifrado de los datos que sé suben se realiza con el algoritmo **ChaCha20-Poly1305** https://datatracker.ietf.org/doc/html/rfc7539

SciLifeLab

5. OTROS PROYECTOS

EN SCILIFELAB

SciLifeLab

6. FINAL

CONTACTO:
LINKEDIN - ALVARO REVUELTA MARTINEZ
ALVARO.REVUELTA@SCILIFELAB.UU.SE

SciLifeLab

6. FINAL

PREGUNTAS

SciLifeLab