

Extracción de información de bases de datos de literatura científica



Jaime Andrés Hurtado Giraldo - Colombia



- Ingeniero químico
- Security Developer en Fluid Attacks
- Estudiante Maestría en Ingeniería de Sistemas y Computación. Universidad del Valle. Cali-Colombia
- Investigador en Grupo de Univalle en Inteligencia Artificial

jaime.hurtado@correounivalle.edu.co

jhurtado@fluidattacks.com

www.linkedin.com/in/jandresh



Universidad del Valle

<https://www.univalle.edu.co/>



GUIA
Grupo de Univalle en
Inteligencia Artificial

<http://eisc.univalle.edu.co/index.php/grupos-investigacion/guia>



fluid
attacks

<https://fluidattacks.com/>
<https://gitlab.com/fluidattacks/product>

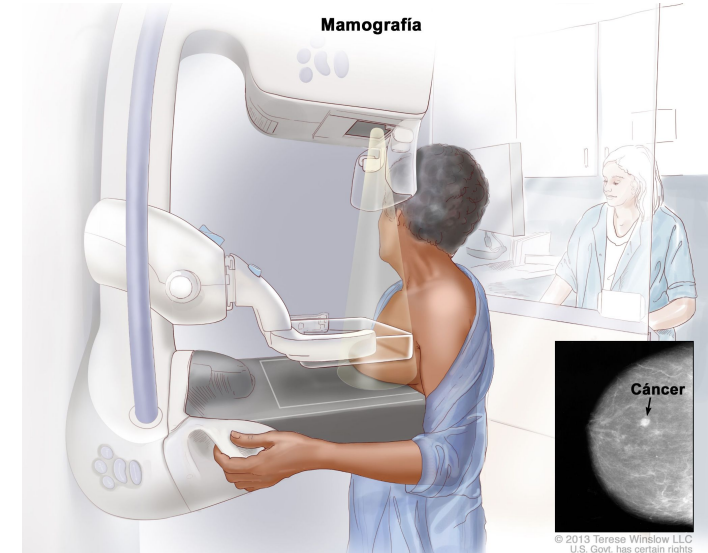
Contenido

- Introducción
- Bases de datos de literatura científica
- Stack tecnológico
- Dockerización de microservicios
- Prototipo local
- Despliegue en la nube
- Preguntas



Introducción

Introducción



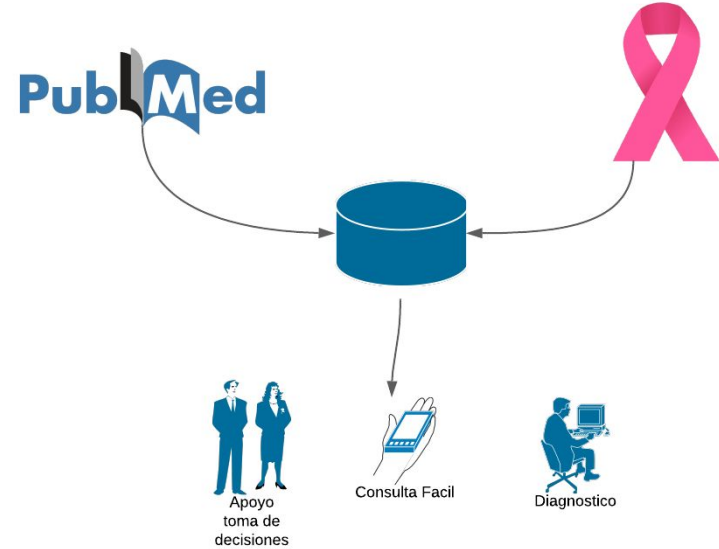
Fuente: Exámenes de detección del cáncer de seno (mama) (PDQ®)-Versión para pacientes

Fuente: El blog de la ingeniería

Introducción



Fuente: [Errores diagnósticos en medicina ¿cómo evitarlos?](#)

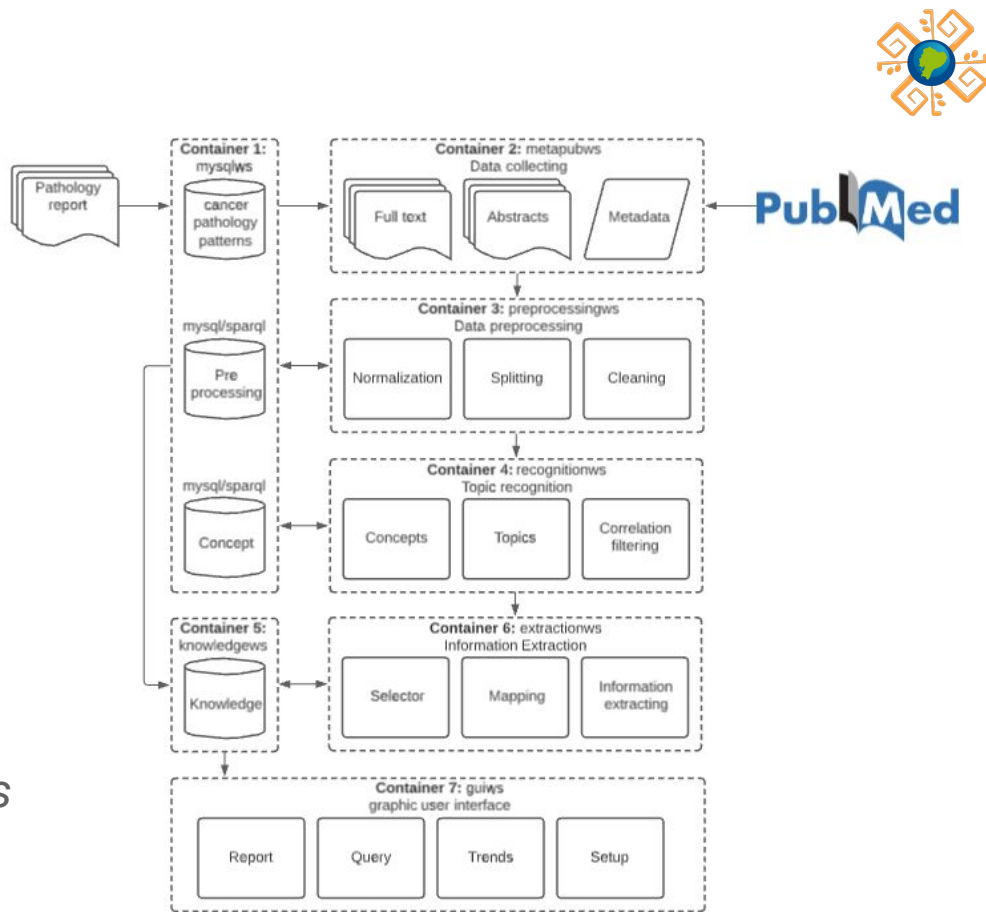


Fuente: Elaboración propia

Introducción

**Objetivo proyecto investigación
analítica de datos de cáncer en Cali:**

*Implementar un sistema prototipo de
enriquecimiento semántico
automático que integre información
de las patologías de cáncer en Cali
con información de literatura
científica publicada en bases de datos*



Fuente: Elaboración propia

Bases de datos de literatura científica



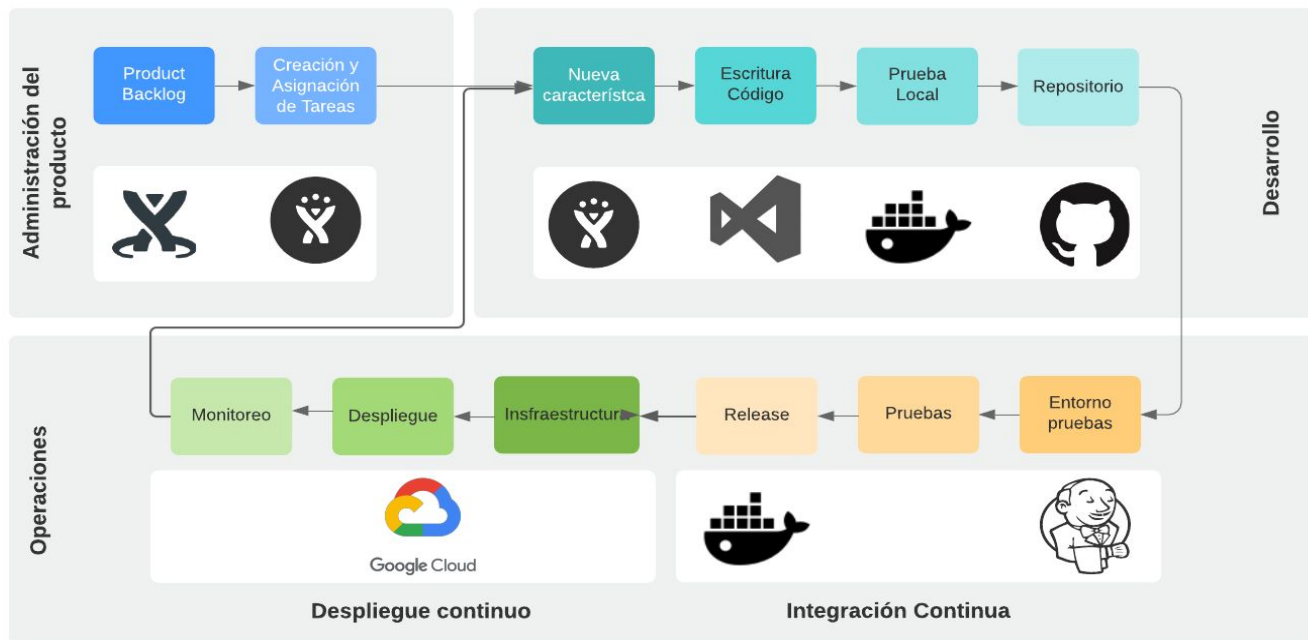
Bases de datos de literatura científica

- Recopilaciones de publicaciones de contenido científico-técnico
- Almacenan artículos de revistas, libros, tesis, congresos
- Tienen como objetivo reunir toda la producción bibliográfica posible sobre un área de conocimiento
- Listado bases de datos

Base de Datos	Url	Librería / API
Arxiv	https://arxiv.org/	https://pypi.org/project/arxiv/
Core	https://core.ac.uk/	https://api.core.ac.uk/docs/v3
Pubmed	https://pubmed.ncbi.nlm.nih.gov/	https://pypi.org/project/metapub/

Stack tecnológico

Stack tecnológico



Fuente: Elaboración propia

Dockerización de microservicios

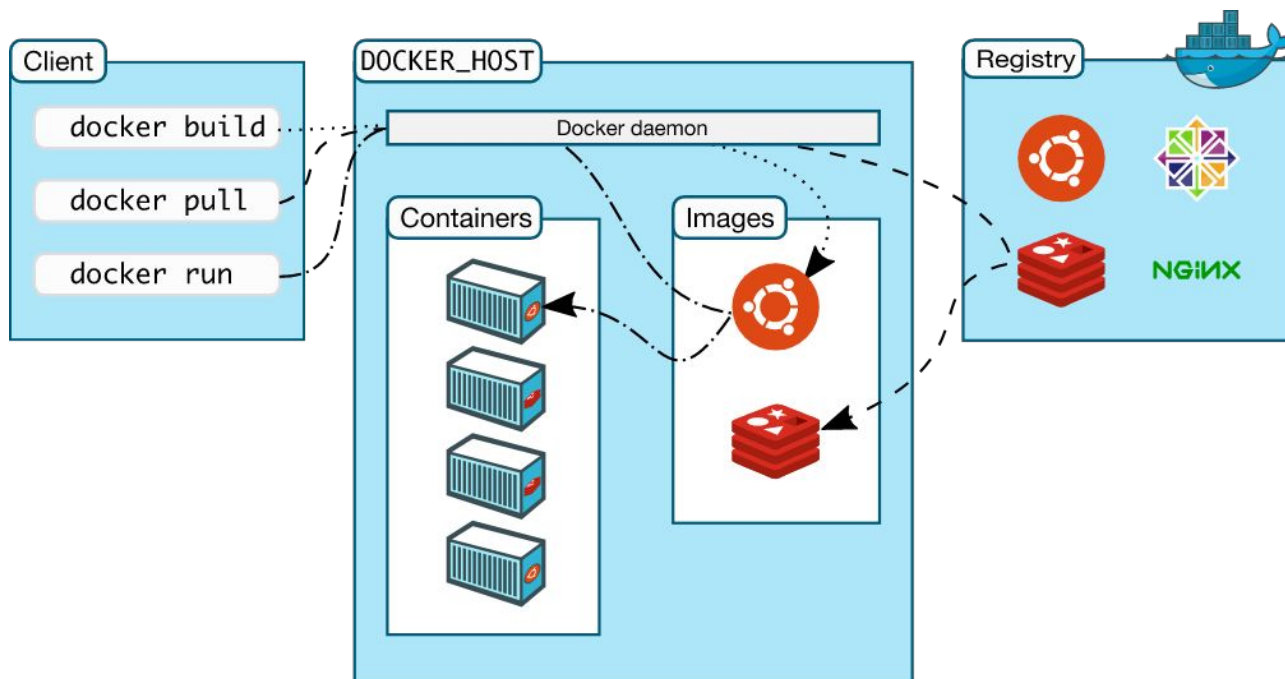


Dockerización de microservicios

Se construye un webservice con [Flask](#) para acceso a la base de datos [Arxiv](#):

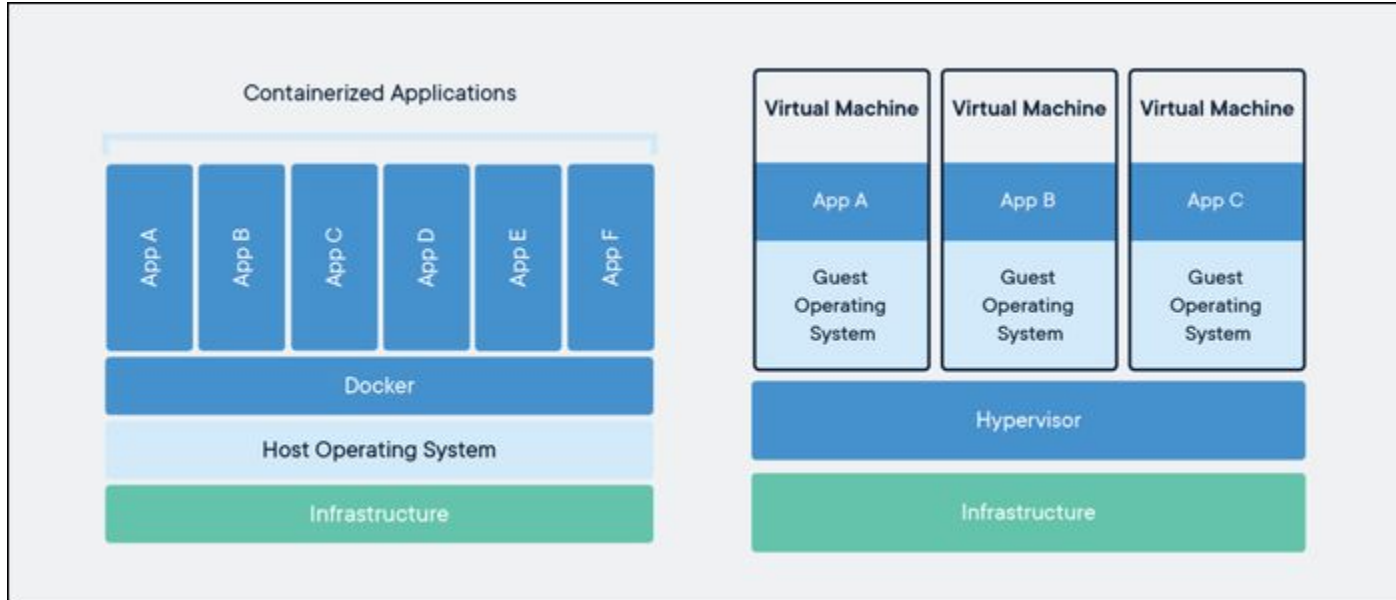
- Instalar distro linux (Sugerida: [Ubuntu](#))
- Instalar [vscode](#)
- Crear repositorio en github e instalar Git:
 `sudo apt-get install git`
 `git --version`
- Instalar [Docker](#) y [docker-compose](#)
- Clonar repositorio [Arxiv-Flask-Docker-compose](#) y pasar a la rama arxiv:
 `git clone https://github.com/josanabr/RunCLI.git`
 `git checkout arxiv`
- Arranque y utilización del microservicio:
 `sudo docker-compose up`

Docker



Fuente: [Docker overview](#)


Docker



Fuente: [What Does Docker Do, and When Should You Use It? – CloudSavvy IT](#)

Docker

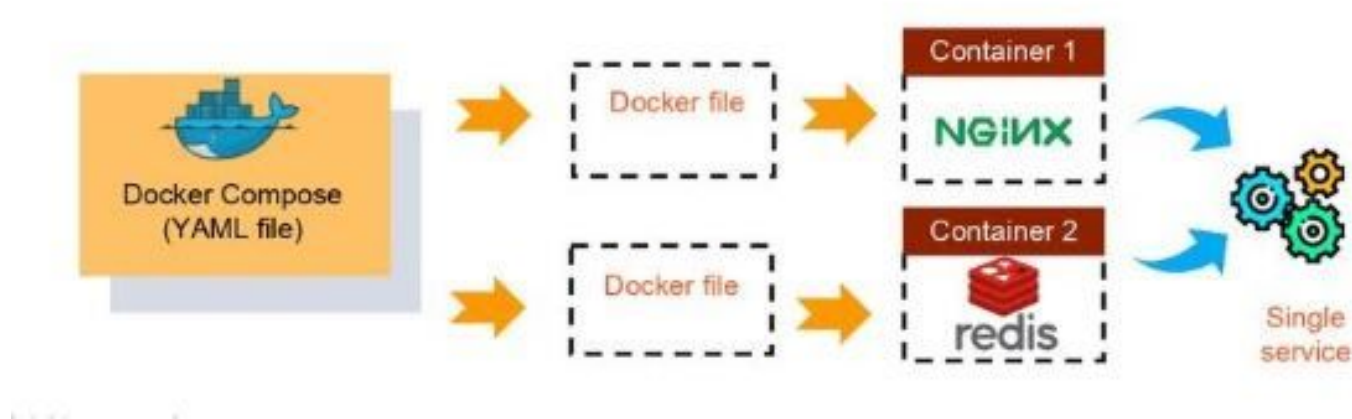


 Cheatsheet for Docker CLI			
Run a new Container	Manage Containers	Manage Images	Info & Stats
<p>Start a new Container from an Image</p> <pre>docker run IMAGE docker run nginx</pre> <p>...and assign it a name</p> <pre>docker run --name CONTAINER_NAME IMAGE docker run --name web nginx</pre> <p>...and map a port</p> <pre>docker run -p HOSTPORT:CONTAINERPORT IMAGE docker run -p 8080:80 nginx</pre> <p>...and map all ports</p> <pre>docker run -P IMAGE docker run -P nginx</pre> <p>...and start container in background</p> <pre>docker run -d IMAGE docker run -d nginx</pre> <p>...and assign it a hostname</p> <pre>docker run --hostname HOSTNAME IMAGE docker run --hostname srv nginx</pre> <p>...and add a dns entry</p> <pre>docker run --add-host HOSTNAME:IP IMAGE</pre> <p>...and map a local directory into the container</p> <pre>docker run -v HOSTDIR:TARGETDIR IMAGE docker run -v ~/.usr/share/nginx/html nginx</pre> <p>...but change the entrypoint</p> <pre>docker run -it --entrypoint EXECUTABLE IMAGE docker run -it --entrypoint bash nginx</pre>	<p>Show a list of running containers</p> <pre>docker ps</pre> <p>Show a list of all containers</p> <pre>docker ps -a</pre> <p>Delete a container</p> <pre>docker rm CONTAINER docker rm web</pre> <p>Delete a running container</p> <pre>docker rm -f CONTAINER docker rm -f web</pre> <p>Delete stopped containers</p> <pre>docker container prune</pre> <p>Stop a running container</p> <pre>docker stop CONTAINER docker stop web</pre> <p>Start a stopped container</p> <pre>docker start CONTAINER docker start web</pre> <p>Copy a file from a container to the host</p> <pre>docker cp CONTAINER:SOURCE TARGET docker cp web:/index.html index.html</pre> <p>Copy a file from the host to a container</p> <pre>docker cp TARGET CONTAINER:SOURCE docker cp index.html web:/index.html</pre> <p>Start a shell inside a running container</p> <pre>docker exec -it CONTAINER EXECUTABLE docker exec -it web bash</pre> <p>Rename a container</p> <pre>docker rename OLD_NAME NEW_NAME docker rename 096 web</pre> <p>Create an image out of container</p> <pre>docker commit CONTAINER docker commit web</pre>	<p>Download an image</p> <pre>docker pull IMAGE[:TAG] docker pull nginx</pre> <p>Upload an image to a repository</p> <pre>docker push IMAGE docker push myimage:1.0</pre> <p>Delete an image</p> <pre>docker rmi IMAGE</pre> <p>Show a list of all Images</p> <pre>docker images</pre> <p>Delete dangling images</p> <pre>docker image prune</pre> <p>Delete all unused images</p> <pre>docker image prune -a</pre> <p>Build an image from a Dockerfile</p> <pre>docker build DIRECTORY docker build .</pre> <p>Tag an image</p> <pre>docker tag IMAGE NEWIMAGE docker tag ubuntu ubuntu:18.04</pre> <p>Build and tag an image from a Dockerfile</p> <pre>docker build -t IMAGE DIRECTORY docker build -t myimage .</pre> <p>Save an image to a tar file</p> <pre>docker save IMAGE > FILE docker save nginx > nginx.tar</pre> <p>Load an image from a tar file</p> <pre>docker load -i TARFILE docker load -i nginx.tar</pre>	<p>Show the logs of a container</p> <pre>docker logs CONTAINER docker logs web</pre> <p>Show stats of running containers</p> <pre>docker stats</pre> <p>Show processes of container</p> <pre>docker top CONTAINER docker top web</pre> <p>Show installed docker version</p> <pre>docker version</pre> <p>Get detailed info about an object</p> <pre>docker inspect NAME docker inspect nginx</pre> <p>Show all modified files in container</p> <pre>docker diff CONTAINER docker diff web</pre> <p>Show mapped ports of a container</p> <pre>docker port CONTAINER docker port web</pre>

Fuente: [The Ultimate Docker Cheat Sheet | dockerlabs](#)



docker-compose



Fuente: [Docker Compose file vs Dockerfile - InfoHubBlog](#)

Prototipo local



Prototipo local

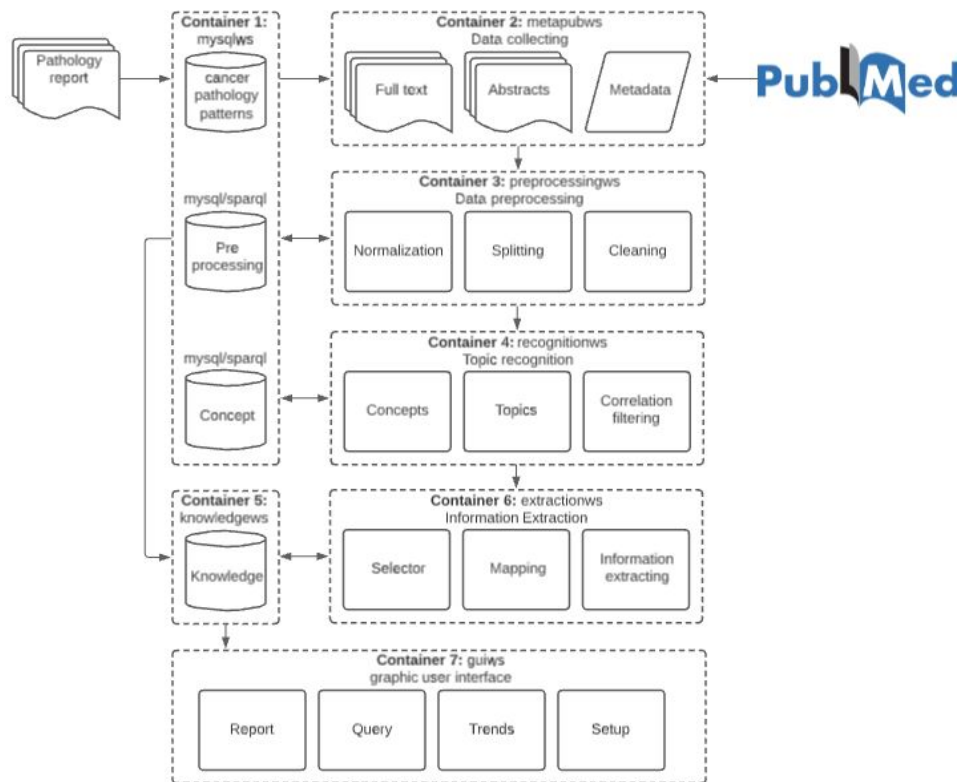
Se despliega localmente una arquitectura basada en microservicios:

- Clonar repositorio [madesoft2](https://github.com/jandresh/madesoft2) y pasar a la rama arxiv:
`git clone https://github.com/jandresh/madesoft2.git`
- Arranque y utilización de los microservicios ingresando a cada carpeta:
`sudo docker-compose build`
`sudo docker-compose up`
`sudo docker-compose down`
`Sudo docker volume prune`
- Iniciar base de datos:
`curl localhost:5001/init`
`curl localhost:5001/txt2patterns`
- Consumo de los microservicios

Prototipo local



```
▼ madesoft2
  > corews
  > dbws
  > guiws
  > guiws2
  > jenkins
  > metapubws
  > orchestratorws
  > preprocessingws
  $ container-publish.sh
  $ functional.sh
  📄 Jenkinsfile
  ≡ out.txt
  ! pod-template.yaml
  $ post-pipeline.sh
  $ test-environment1.sh
  $ test-environment2.sh
```



Fuente: Elaboración propia

Despliegue en la nube



Despliegue en la nube

Se despliega en la nube utilizando [GCloud](#), integrador [Jenkins](#) y un cluster [kubernetes](#):

- Paso a paso para el despliegue de repositorio [madesoft2](#) ([Documento](#))
- Acceso a Kubernetes
- Revisión de pipelines de integración en Jenkins
- Revision servicios en el cluster kubernetes
- Consumo de microservicios desde la nube

```
kubectl get svc -A
```

```
kubectl get pods --namespace=<namespace>
```

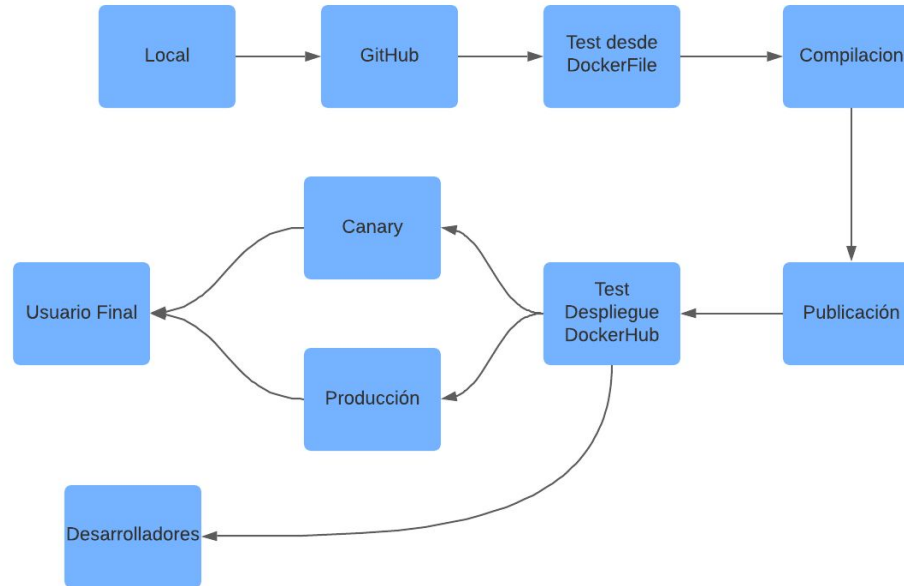
```
kubectl exec --stdin --tty <pod-name> --namespace="<namespace>" -it sh
```

```
kubectl logs <pod-name> --namespace=<namespace>
```

```
kubectl cp <namespace>/<pod-name>:<filepath> <file-name>
```



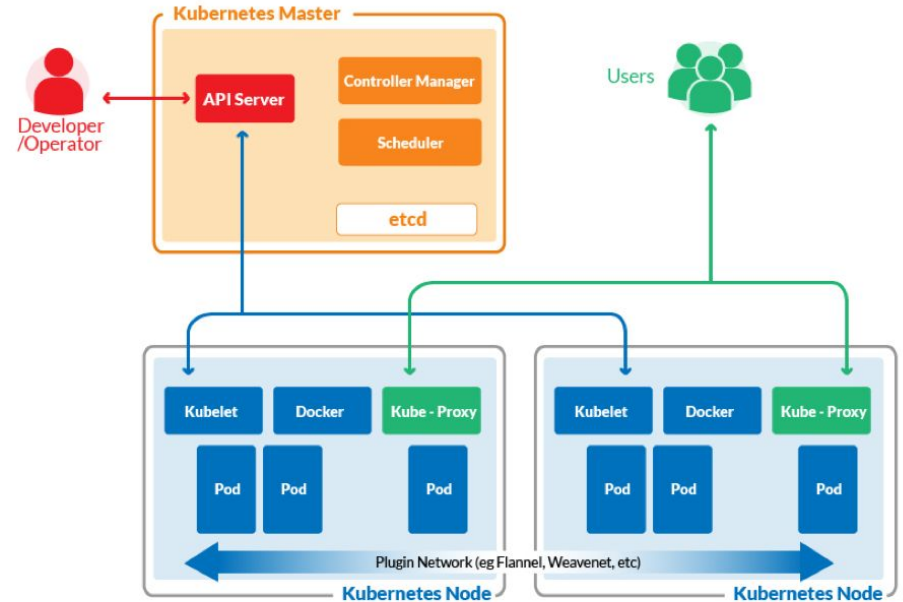
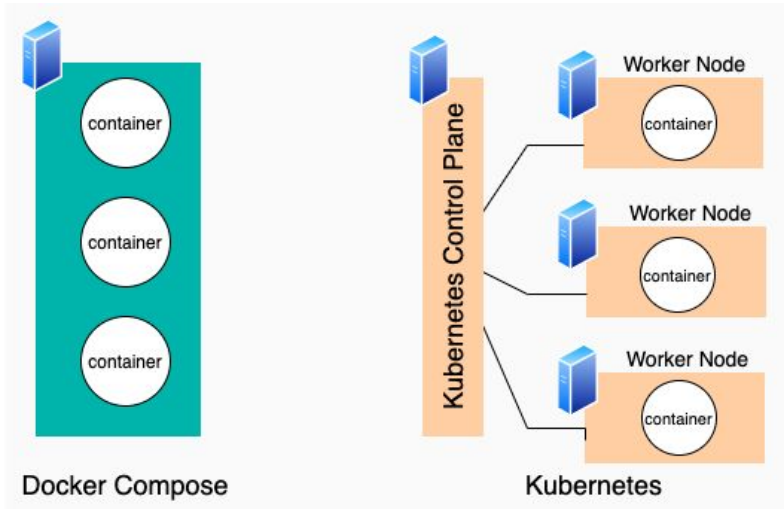
Estrategia canary



Fuente: Elaboración propia



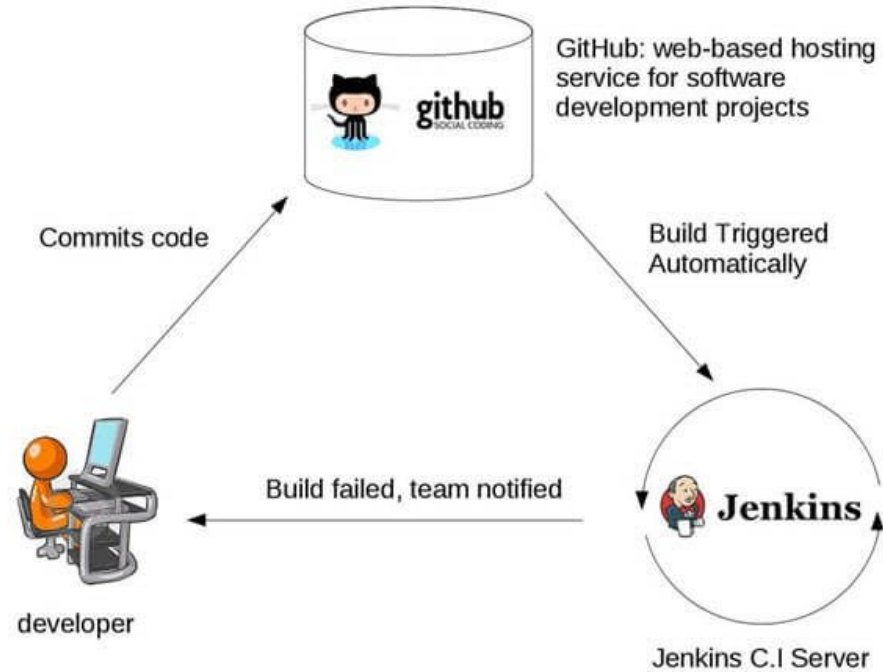
Kubernetes



Fuente: [What is Kubernetes vs Docker Compose? How these DevOps tools compare - Coffee Talk: Java, News, Stories and Opinions](#)

Fuente: [Kubernetes vs. Docker: A Primer - Container Journal](#)

Jenkins



Fuente: [What is Jenkins? Why Use Continuous Integration \(CI\) Tool?](#)

Preguntas

