



Certified Tech Developer

The Ultimate Degree

Infraestructura II

Ejercitación GitLab + Docker Hub

Parte 1

OBJETIVO

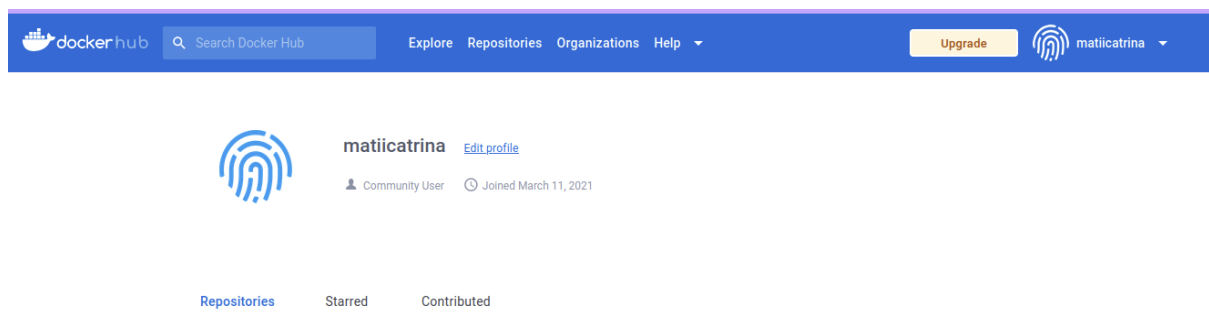
Generar un pipeline en GitLab que me permita pushear el repositorio y generar una imagen de docker hacia el repositorio de Docker Hub.

RESOLUCIÓN

La tarea principal es crear una cuenta en Docker Hub sino la tenemos.

Link : <https://hub.docker.com/>

Deberíamos poder llegar a nuestro perfil:



Vamos a lanzar nuestra infraestructura (desde 0 - sin utilizar lo que ya habíamos hecho -) a través de un Docker Compose.

****Necesitamos instalar docker-compose sino lo tenemos con el comando:**

`sudo apt update`

`sudo apt install docker-compose`

Levantamos GitLab

Creamos un directorio (en la raíz de nuestro usuario) en donde vamos a persistir todos los datos de nuestro servidor

`mkdir gitlab && cd gitlab`

Ahora creamos un archivo (recomiendo que lo hagan en visual studio code) llamado docker-compose.yml y dentro de él colocamos lo siguiente:

version: '3.7'

services:

gitlab:

image: 'gitlab/gitlab-ce:latest'

restart: always

hostname: 'localhost'

container_name: gitlab

environment:

GITLAB_OMNIBUS_CONFIG: |

external_url 'http://localhost'

ports:

- '8080:80'

- '8443:443'

volumes:

- '\$GITLAB_HOME/config:/etc/gitlab'

- '\$GITLAB_HOME/logs:/var/log/gitlab'

- '\$GITLAB_HOME/data:/var/opt/gitlab'

networks:

- gitlab

gitlab-runner:

image: gitlab/gitlab-runner:alpine

container_name: gitlab-runner

restart: always

depends_on:

- gitlab

volumes:

- /var/run/docker.sock:/var/run/docker.sock

- '\$GITLAB_HOME/gitlab-runner:/etc/gitlab-runner'

networks:

- gitlab

networks:

gitlab:

name: gitlab-network

Nos tiene que quedar similar a la siguiente imagen:

```
version: '3.7'
services:
  gitlab:
    image: 'gitlab/gitlab-ce:latest'
    restart: always
    hostname: 'localhost'
    container_name: gitlab
    environment:
      GITLAB_OMNIBUS_CONFIG: |
        external_url 'http://localhost'
    ports:
      - '8080:80'
      - '8443:443'
    volumes:
      - '$GITLAB_HOME/config:/etc/gitlab'
      - '$GITLAB_HOME/logs:/var/log/gitlab'
      - '$GITLAB_HOME/data:/var/opt/gitlab'
    networks:
      - gitlab
  gitlab-runner:
    image: gitlab/gitlab-runner:alpine
    container_name: gitlab-runner
    restart: always
    depends_on:
      - gitlab
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - '$GITLAB_HOME/gitlab-runner:/etc/gitlab-runner'
    networks:
      - gitlab

networks:
  gitlab:
    name: gitlab-network
```

Luego exportamos la variable que va a utilizar el docker-compose

```
export GITLAB_HOME=$HOME/gitlab
```

Teniendo ya nuestro docker-compose.yml configurado lo vamos a ejecutar para poder desplegar la configuración GITLAB + RUNNER (2 contenedores). Ejecutamos el siguiente comando:

```
sudo -E docker-compose up -d
```

Buscamos una soda y esperamos 10 minutitos que todo se ejecute correctamente, deberíamos llegar a la siguiente situación:

```
matl@matl-virtual-machine: ~/gitlab
matl@matl-virtual-machine:~/gitlab$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
44b53a589ce8	gitlab/gitlab-runner:alpine	"/usr/bin/dumb-init ..."	About an hour ago	Up About an hour	
001a3b970b13	gitlab/gitlab-ce:latest	"/assets/wrapper"	About an hour ago	Up About an hour (healthy)	22/tcp, 0

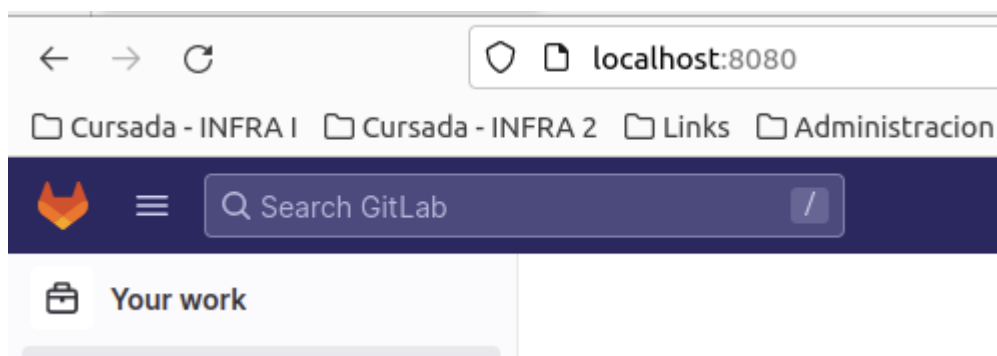
```
matl@matl-virtual-machine:~/gitlab$
```

Luego vamos a poder ejecutar el siguiente comando para obtener nuestra clave de root de GitLab:

```
docker exec -it gitlab grep 'Password:' /etc/gitlab/initial_root_password
```

```
matl@matl-virtual-machine:~/gitlab$ docker exec -it gitlab grep 'Password:' /etc/gitlab/initial_root_password
Password: ZqLYPR1+lBBVYqLhEEM3czrINjt0+SBayIjribJIIdPw=
```

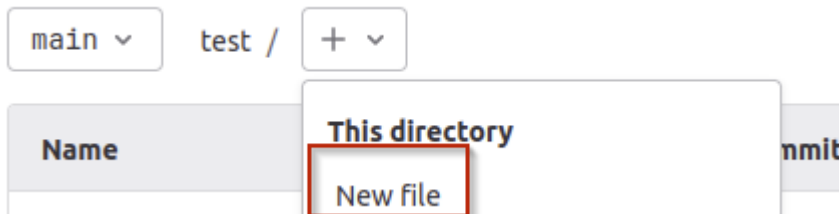
Ya con estas credenciales podemos acceder a nuestro servidor Gitlab a través del localhost:8080



Procedemos a crear un nuevo proyecto y dentro de él vamos a subir una plantilla web, puede ser la misma que estuvieron usando anteriormente o pueden buscar alguna diferente.

En mi caso subí una que encontré en la siguiente web:
<https://plantillashtmlgratis.com/en/home/>

Luego procedemos a crear un Dockerfile que va a ser quien construirá nuestra imagen con el proyecto web ya integrado de la siguiente manera:



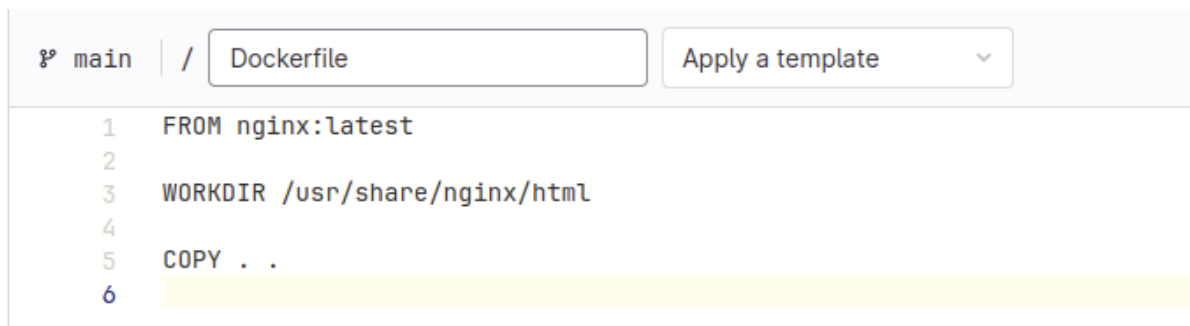
Creamos un archivo Llamado Dockerfile y lo completamos con el siguiente código;

FROM nginx:latest

WORKDIR /usr/share/nginx/html

COPY ..

New file



Guardamos nuestro archivo y nos quedaría de la siguiente manera nuestro Proyecto + el Dockerfile:

main ▾	test / + ▾	Find file	Web IDE	↓ ▾	Clone ▾
README	CHANGELOG	CI/CD configuration	Add LICENSE	Add CONTRIBUTING	Auto DevOps enabled
Add Kubernetes cluster	Add Wiki	Configure Integrations			

Name	Last commit	Last update
css	subiendo repo	49 minutes ago
images	subiendo repo	49 minutes ago
js	subiendo repo	49 minutes ago
.gitlab-ci.yml	Update .gitlab-ci.yml file	38 minutes ago
Dockerfile	subo Dockerfile	43 minutes ago
README.md	Initial commit	1 hour ago
client.html	subiendo repo	49 minutes ago
contact.html	subiendo repo	49 minutes ago
health.html	subiendo repo	49 minutes ago
index.html	subiendo repo	49 minutes ago
medicine.html	subiendo repo	49 minutes ago
news.html	subiendo repo	49 minutes ago

Ya estamos prácticamente listos para que el pipeline corra automáticamente, para ello vamos a tener que configurar nuestro Runner con el token correspondiente al proyecto. Ejecutemos el siguiente comando (modificar solo lo que está en rojo):

```
sudo docker exec -it gitlab-runner gitlab-runner register
--url "http://gitlab" --clone-url "http://gitlab"
--registration-token <TOKEN de nuestro Runner>
```

Pulsamos Enter en todas las opciones, menos cuando nos pida **Enter an executor**, allí tipeamos **docker** y pulsamos Enter nuevamente, la siguiente opción que nos preguntará será **Enter the default Docker image**, allí ingresamos **ruby:2.7** y pulsamos Enter.

Ahora si volvemos a ingresar nuevamente a **Settings** → **CI/CD** → **Runner**, nos debiera aparecer el runner activado, como se muestra en la siguiente imagen:

Assigned project runners

● #1 (X1eAYcxYN) 🔒

44b53a589ce8

  Remove runner

Una vez nuestro runner en ejecución falta configurar algunos parámetros dentro del mismo. Para ello vamos a ingresar al contenedor a través del siguiente comando:

```
sudo docker exec -it gitlab-runner bash
```

Después modificamos el archivo **config.toml** desde la siguiente ruta y con el comando:

```
vi /etc/gitlab-runner/config.toml
```

Los siguientes parámetros:

```
privileged = true
```

```
network_mode = "gitlab-network"
```

Deberíamos tener una configuración similar a la siguiente:

```
[runners.docker]
tls_verify = false
image = "ruby:2.7"
privileged = true
disable_entrypoint_overwrite = false
oom_kill_disable = false
disable_cache = false
volumes = ["/cache"]
shm_size = 0
network_mode = "gitlab-network"
bash-5.1#
```

!!!YA CASI ESTAMOS!!!

Luego dentro de gitlab en la solapa de BUILD vamos a cargar nuestro pipeline:

```
stages:

  - compose

stage_compose:

  stage: compose

  image: docker:stable

  when: manual

  services:

    - docker:dind

    - name: docker:dind

      alias: thedockerhost

  variables:

    DOCKER_HOST: tcp://thedockerhost:2375/

    DOCKER_DRIVER: overlay2

    DOCKER_TLS_CERTDIR: ""

  script:

    - docker login -u $CI_REGISTRY_USER -p
$CI_REGISTRY_PASSWORD $CI_REGISTRY

    - docker build -t test:latest .

    - docker tag test:latest $CI_REGISTRY_USER/test:latest

    - docker push $CI_REGISTRY_USER/test:latest
```

Nos quedaría de la siguiente manera:

CI/CD

Pipelines

Editor

Jobs

Schedules

Security & Compliance

Deployments

Packages and registries

Infrastructure

Monitor

Analytics

Wiki

Snippets

Settings

✓ Pipeline syntax is correct. [Learn more](#)

Edit

Visualize

Validate

NEW

View merged YAML

Browse templates

Help

```

1 build-push--job:
2   image: docker:stable
3   services:
4     - docker:dind
5     - name: docker:dind
6       alias: thedockerhost
7   variables:
8     DOCKER_HOST: tcp://thedockerhost:2375/
9     DOCKER_DRIVER: overlay2
10    DOCKER_TLS_CERTDIR: ""
11
12   script:
13     - docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD $CI_REGISTRY
14     - docker build -t test:latest .
15     - docker tag test:latest matiicatrina/test:latest
16     - docker push matiicatrina/test:latest
17   only:
18     - main
19
20

```

Si prestamos atención, hay 3 variables que vamos a tener que cargar en nuestra sección de variables para que GitLab las utilice. Para ello dentro de nuestro proyecto vamos a Settings > CI/CD > Variables , Una vez ahí cargamos nuestras credenciales de la siguiente manera:

- CI_REGISTRY = docker.io
- CI_REGISTRY_PASSWORD = tupassoword (de dockerhub)
- CI_REGISTRY_USER = tuusuario (de dockerhub)

Variables

Collapse

Variables store information, like passwords and secret keys, that you can use in job scripts. Each project can define a maximum of 8000 variables. [Learn more](#).

Variables can have several attributes. [Learn more](#).

- **Protected:** Only exposed to protected branches or protected tags.
- **Masked:** Hidden in job logs. Must match masking requirements.
- **Expanded:** Variables with \$ will be treated as the start of a reference to another variable.

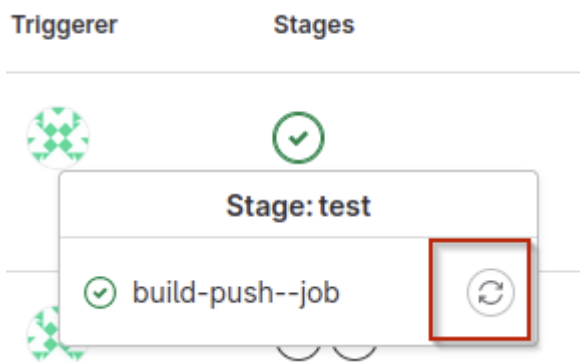
Environment variables are configured by your administrator to be **protected** by default.

Type	↑ Key	Value	Options	Environments	
Variable	CI_REGISTRY	*****	Protected, Expanded	All (default)	
Variable	CI_REGISTRY_PASSWORD	*****	Protected, Expanded	All (default)	
Variable	CI_REGISTRY_USER	*****	Protected, Expanded	All (default)	

Nuestro pipeline debería pasar a **passed**:

All 2	Finished	Branches	Tags	Clear runner caches	CI lint	Run pipeline
Filter pipelines				Q	Show Pipeline ID ▾	
Status	Pipeline	Triggerer	Stages			
<p>passed</p> <p>00:00:55</p> <p>52 minutes ago</p>	<p>Update .gitlab-ci.yml file</p> <p>#2 main 75729637</p> <p>latest</p>			<p>Download ▾</p>		


Sino lo volvemos a lanzar desde el siguiente botón;



Ahora deberíamos tener nuestra imagen en Docker Hub

Repositories Starred Contributed

Displaying 1 to 3 repositories



matricatrina/test • 1 • 0

By [matricatrina](#) • Updated 12 minutes ago

Image

FELICITACIONES