**Guía Laboratorio 3 (T Hernandez)**

# **Docker build, push & deploy**

# **Objetivo**

Comprender el funcionamiento de contendores, mediante prácticas con comandos básicos para el despliegue de aplicaciones completas.

# **Duración**

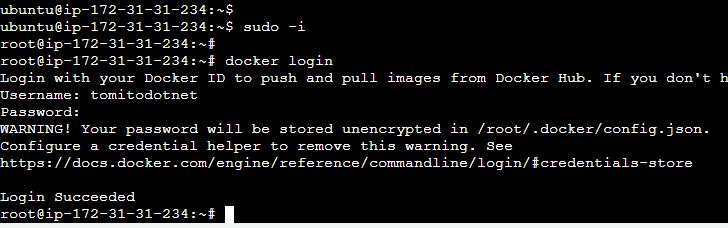
Ciento veinte (80) minutos.

**Actividad 1. Iniciar sesión en Docker HUB.**

1. En el terminal ingrese el comando *docker login* e inicie sesión con sus credenciales.de Docker Hub.

*A computer screen capture

Description automatically generated with low confidence*



**Actividad 2. Crear imágenes de docker**

1. Crear un directorio con el nombre FlaskRedis
2. Dentro del directorio crear un archivo llamado **app.py** (nano app.py)
3. Copiar el siguiente código dentro del archivo:

|  |
| --- |
| from flask import Flask  from redis import Redis  app = Flask(\_\_name\_\_)  redis = Redis(host='redis', port=6379)  @app.route('/')  def hello():  redis.incr('hits')  return 'Hello World! I have been seen %s times.' % redis.get('hits')  if \_\_name\_\_ == "\_\_main\_\_":  app.run(host="0.0.0.0", debug=True) |

1. Dentro del directorio crear otro archivo llamado **requirements.txt** y pegar las siguientes líneas

|  |
| --- |
| flask  redis |

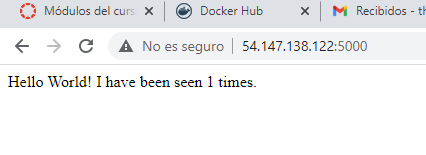
1. Crear un archivo llamado **Dockerfile** , sin extensión. Pegar el siguiente contenido.

|  |
| --- |
| FROM python:2.7  ADD . /code  WORKDIR /code  RUN pip install -r requirements.txt  CMD python app.py |

1. Crear un archivo llamado **docker-compose.yml** . Pegar el siguiente contenido.

|  |
| --- |
| services:  web:  build: .  ports:  - "5000:5000"  volumes:  - .:/code  links:  - redis  redis:  image: redis  volumes:  - persistentRedis:/data  volumes:  persistentRedis: |

1. Levante los contenedores con el comando **docker-compose up -d**
2. En el navegador, abra la aplicación en la IP publica y el puerto correspondiente, adjunte captura de pantalla del resultado.

****

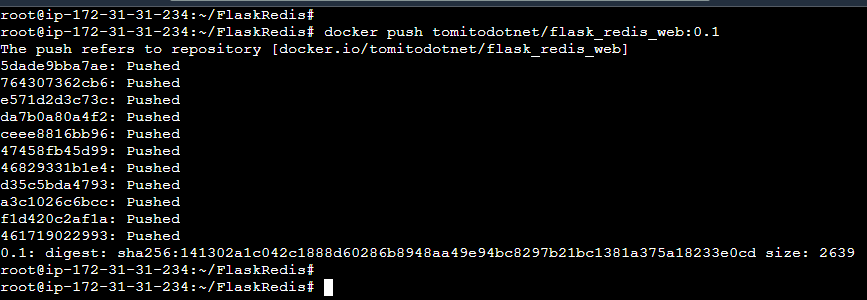
**Actividad 3. Subir imágenes de docker**

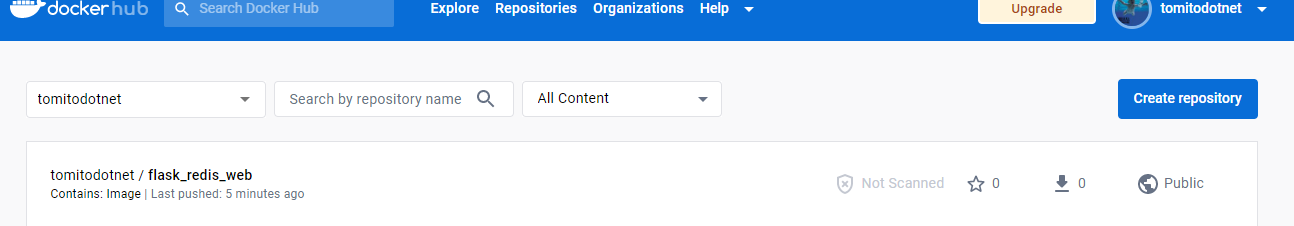
1. Corra el comando *docker image ls* y localiza el contenedor del servicio web creado en la actividad 2.
2. Correr el siguiente comando:

|  |
| --- |
| docker tag <nombre-imagen> <usuario-DockerHub>/<nombre-imagen>[:<tag>] |

**Ej: docker tag flaskredis\_web ceczespol/flask\_redis\_web:0.1**

1. **Docker push <nombre de la imagen>**
2. Adjunte captura de pantalla de su Docker Hub con la nueva imagen cargada.





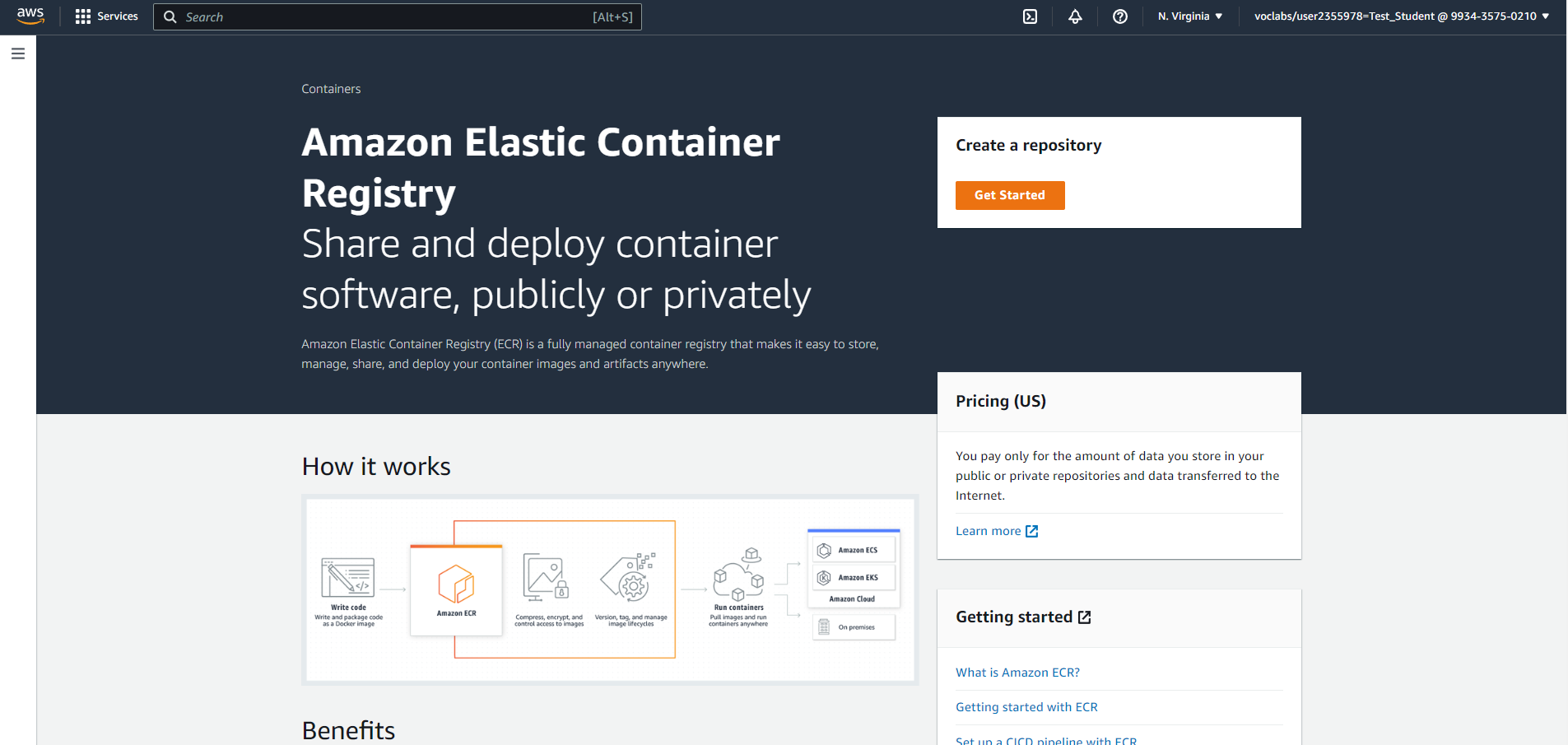
1. Detenga el despliegue local del docker compose con el comando docker-compose down
2. Realice el siguiente cambio en el archivo docker-compose.yml

|  |
| --- |
| services:  web:  image: reemplace-por-suimagen ej: ceczespol/flask\_redis\_web  ports:  - "5000:5000"  volumes:  - .:/code  links:  - redis  redis:  image: redis  volumes:  - persistentRedis:/data  volumes:  persistentRedis: |

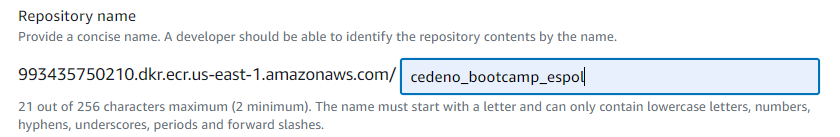
1. Verifique que todo funciona igual.

**Actividad 4. Registro en AWS ECR**

1. En la consola de su máquina virtual correr el comando *sudo apt install awscli*
2. Siga los pasos de la sección AWS Details en el Learner LAB, botón AWS CLI. Deberá crear un archivo en el directorio ~/.aws/credentials y poner los datos ahí mostrados.
3. En la consola de servicios de AWS busque Amazon Elastic Container Registry. De click en Create a repository.

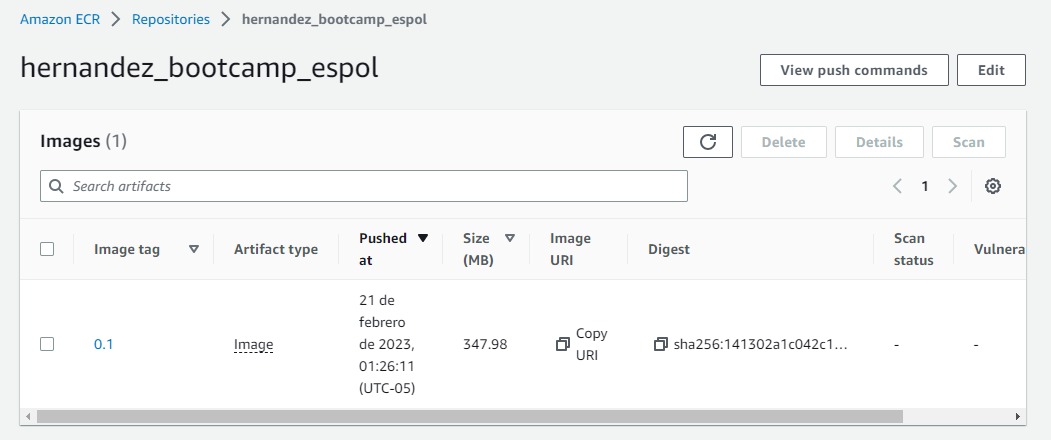


1. En la consola de servicios de AWS busque Amazon Elastic Container Registry. De click en Create a repository. Configure las siguientes opciones:
   1. Visibility Settings: Privado
   2. Repository name: apellido\_bootcamp\_espol



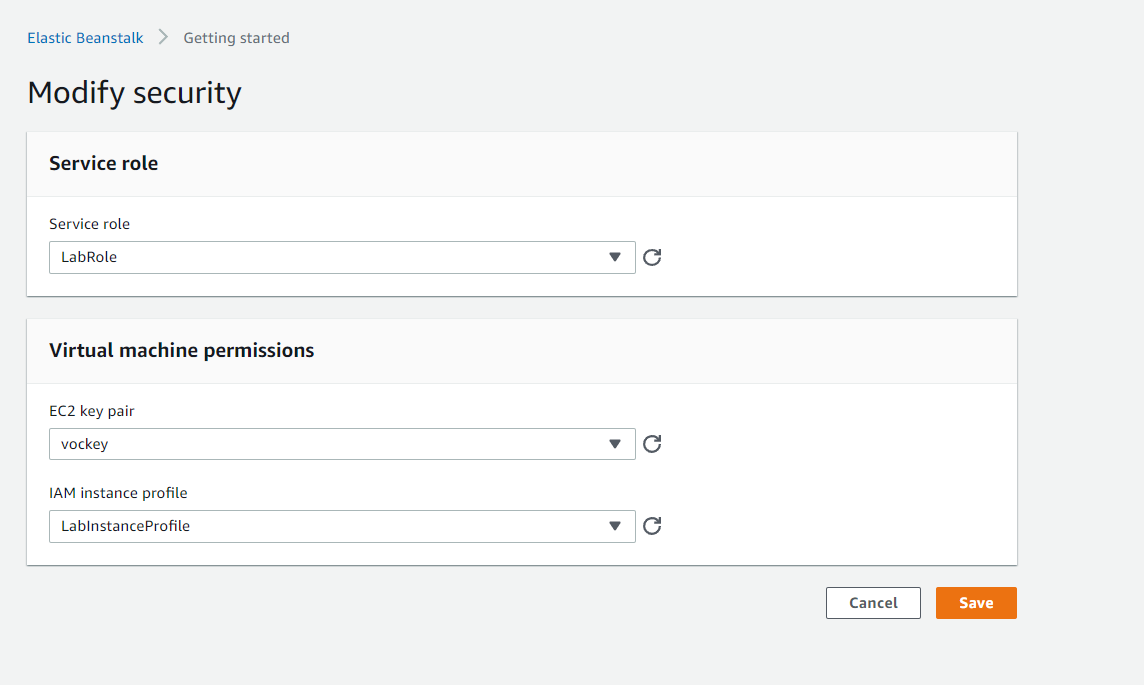
* 1. Deje el resto de los campos sin configurar y dar click en *Create repository* al final de la página.

1. Una vez creado el repositorio, ingrese al mismo y escoja la opción *View push commands.* Siga los pasos ahí mostrados para subir la imagen que se creó en la actividad 3 ahora en AWS.
2. Adjunte captura de pantalla de su AWS Container Registry con la nueva imagen cargada.



**Actividad 5. Despliegue en AWS Beanstalk**

1. De las actividades anteriores, guarde el docker-compose.yml en su máquina haciendo los siguientes cambios:
   1. Elimine los volúmenes
   2. Cambie el atributo *links* por *depens\_on*
2. En la consola de servicios de AWS busque Elastic Beanstalk. De click en Create application.
3. Configure las siguientes opciones
   1. Application name: a su elección.
   2. Platform
      1. Platform: Docker
      2. Platform branck: Docker running on 64bit
      3. Platform version: 3.5.4
   3. Applicacion code: upload your code
   4. Source code origin
      1. Version label: 0.1
      2. Local file: Subir el archivo docker-compose.yml
   5. Configure more options
      1. En la sección security dele click a editar y configure las credenciales como la imagen



* + 1. En la sección network modifique las opciones como en la siguiente imagen:

Graphical user interface, application

Description automatically generated

* 1. Finalizar dando clic en Create application.
  2. El proceso de deploy puede tardar hasta 5 minutos. Adjunte captura de pantalla similar al siguiente ejemplo

Graphical user interface, application

Description automatically generated



1. ¿Pudo ingresar a la web? ¿Cuáles son los posibles escenarios de falla? ¿Como lo solucionaría?

Click or tap here to enter text.