

## Taller 5

*Link a repositorio de GitHub con código de la imagen de Docker.*

<https://github.com/JaniMariQuesiRami/mlops-taller-5>

*Capturas de pantalla con terminal mostrando ejecución exitosa del pipeline de scikit-learn adentro de la imagen de Docker.*

```
PROBLEMS 6 DEBUG CONSOLE TERMINAL PORTS OUTPUT

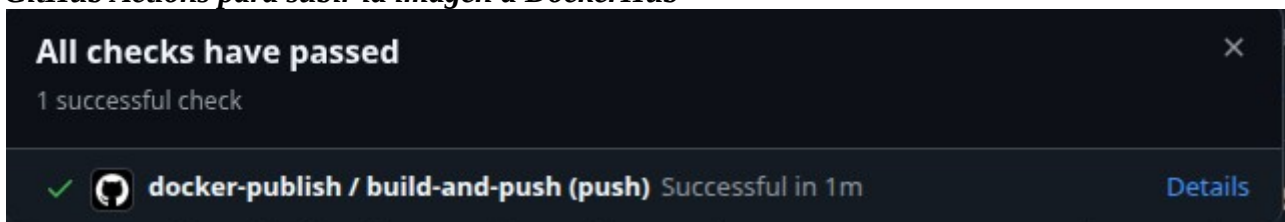
• (venv) janis@janis:~/Documents/mlops-taller-5$ docker build -t jannisce2508/sklearn-pipeline:0.1.0 .
[+] Building 53.9s (11/11) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile                0.1s
=> => transferring dockerfile: 431B                                0.0s
=> [internal] load metadata for docker.io/library/python:3.12-slim 1.3s
=> [internal] load .dockerignore                                   0.0s
=> => transferring context: 111B                                     0.0s
=> [1/6] FROM docker.io/library/python:3.12-slim@sha256:abc799c7ee22b0d66f46c367643088a35e048bbabd81212d73c2323aed 8.4s
=> => resolve docker.io/library/python:3.12-slim@sha256:abc799c7ee22b0d66f46c367643088a35e048bbabd81212d73c2323aed 0.0s
=> => sha256:abc799c7ee22b0d66f46c367643088a35e048bbabd81212d73c2323aed38c64f 10.37kB / 10.37kB 0.0s
=> => sha256:a2dd6679f8a13a533f44782cd31584d4ca4ec7170c84e7e726dc9cc3f862ed42 1.75kB / 1.75kB 0.0s
=> => sha256:f35c889e4f8eeca7a5ccec31926201e4a63df72a87cbd59e7af1bfe2689397354 5.58kB / 5.58kB 0.0s
=> => sha256:ce1261c6d567efa8e3b457673eeeb474a0a8066df6bb95ca9a6a94a31e219dd3 29.77MB / 29.77MB 5.5s
=> => sha256:ae44614ad59ae7d9347036fab246b3ec97ae8dca5a034c8c0057cd787bceb8fa 1.29MB / 1.29MB 1.7s
=> => sha256:48b120ca37b5b627d81d3cb2e63d808078495be97454db590bba47194cb017ee 12.11MB / 12.11MB 3.2s
=> => sha256:debe34bebc01f52b9f6e7fc26f1f74708de668b6088d586352067e5d881087d8 249B / 249B 2.1s
=> => sha256:48b120ca37b5b627d81d3cb2e63d808078495be97454db590bba47194cb017ee 1.4s
=> => extracting sha256:ae44614ad59ae7d9347036fab246b3ec97ae8dca5a034c8c0057cd787bceb8fa 0.1s
=> => extracting sha256:48b120ca37b5b627d81d3cb2e63d808078495be97454db590bba47194cb017ee 0.7s
=> => extracting sha256:debe34bebc01f52b9f6e7fc26f1f74708de668b6088d586352067e5d881087d8 0.0s
=> [internal] load build context                                  6.2s
=> => transferring context: 347.21MB                               6.1s
=> [2/6] WORKDIR /app                                           0.1s
=> [3/6] COPY requirements.txt .                                  0.2s
=> [4/6] RUN pip install --no-cache-dir -r requirements.txt      38.1s
=> [5/6] COPY . .                                                1.9s
=> [6/6] RUN useradd -m appuser                                  0.5s
=> => exporting to image                                           2.9s
=> => exporting layers                                             2.9s
=> => writing image sha256:85aef9979ed447ac2a248e7a1ba9457bcb38f634717949c0508461ae10ed2ad 0.0s
=> => naming to docker.io/jannisce2508/sklearn-pipeline:0.1.0 0.0s
• (venv) janis@janis:~/Documents/mlops-taller-5$ docker run --rm jannisce2508/sklearn-pipeline:0.1.0
Accuracy: 0.9333
Reporte de clasificación:
      precision    recall  f1-score   support

   0       1.0000     1.0000     1.0000        10
   1       0.9000     0.9000     0.9000        10
   2       0.9000     0.9000     0.9000        10

 accuracy          0.9333          0.9333          0.9333         30
 macro avg          0.9333          0.9333          0.9333         30
 weighted avg       0.9333          0.9333          0.9333         30

❖ (venv) janis@janis:~/Documents/mlops-taller-5$
```

*GitHub Actions para subir la imagen a DockerHub*



*Link a Docker Hub de la imagen generada.*

<https://hub.docker.com/r/jannisce2508/sklearn-pipeline>

### ***Link a Documentación de Docker con comando(s) para compartir una imagen.***

#### **Guardar una imagen.**

Usa docker image save para exportar una imagen a un archivo .tar. Conserva capas y tags. Útil para compartir sin usar un registro. Ejemplo conceptual: “guarda <dockerhub\_username>/<image\_name>:<tag> en <archivo>.tar y compártelo por cualquier medio”.

<https://docs.docker.com/reference/cli/docker/image/save/>

#### **Cargar una imagen.**

Usa docker image load para importar una imagen desde un .tar o desde STDIN. Restaura la imagen con sus tags y queda lista para ejecutar.

<https://docs.docker.com/reference/cli/docker/image/load/>

### ***Conclusiones del ejercicio comparando estrategias que conocen para hacer portátil su código (paquetes de Python, pickles, Docker, etc)***

Para portar código hay tres enfoques con roles distintos. Empaquetar como **paquete de Python** facilita reutilizar funciones y pipelines. Requiere recrear el entorno y fijar versiones. Funciona bien dentro del ecosistema Python. **Pickles/joblib** solo mueven el estado del modelo. Son frágiles a versiones y a veces inseguros. No resuelven dependencias del sistema ni de Python. No son un artefacto de despliegue por sí solos. **Docker** congela sistema, Python y librerías con el código. Ejecuta igual en cualquier host con Docker. Pesa más y exige gestionar imágenes, pero entrega reproducibilidad y trazabilidad. Recomendación práctica: código como paquete + dependencias fijadas + contenedor para ejecutar. Exporta el modelo (pickle/ONNX) como artefacto de build, no como mecanismo de portabilidad principal. Para compartir sin registro usa **docker save/load**. En resumen: paquetes portan API, pickles portan estado, Docker porta el runtime completo.