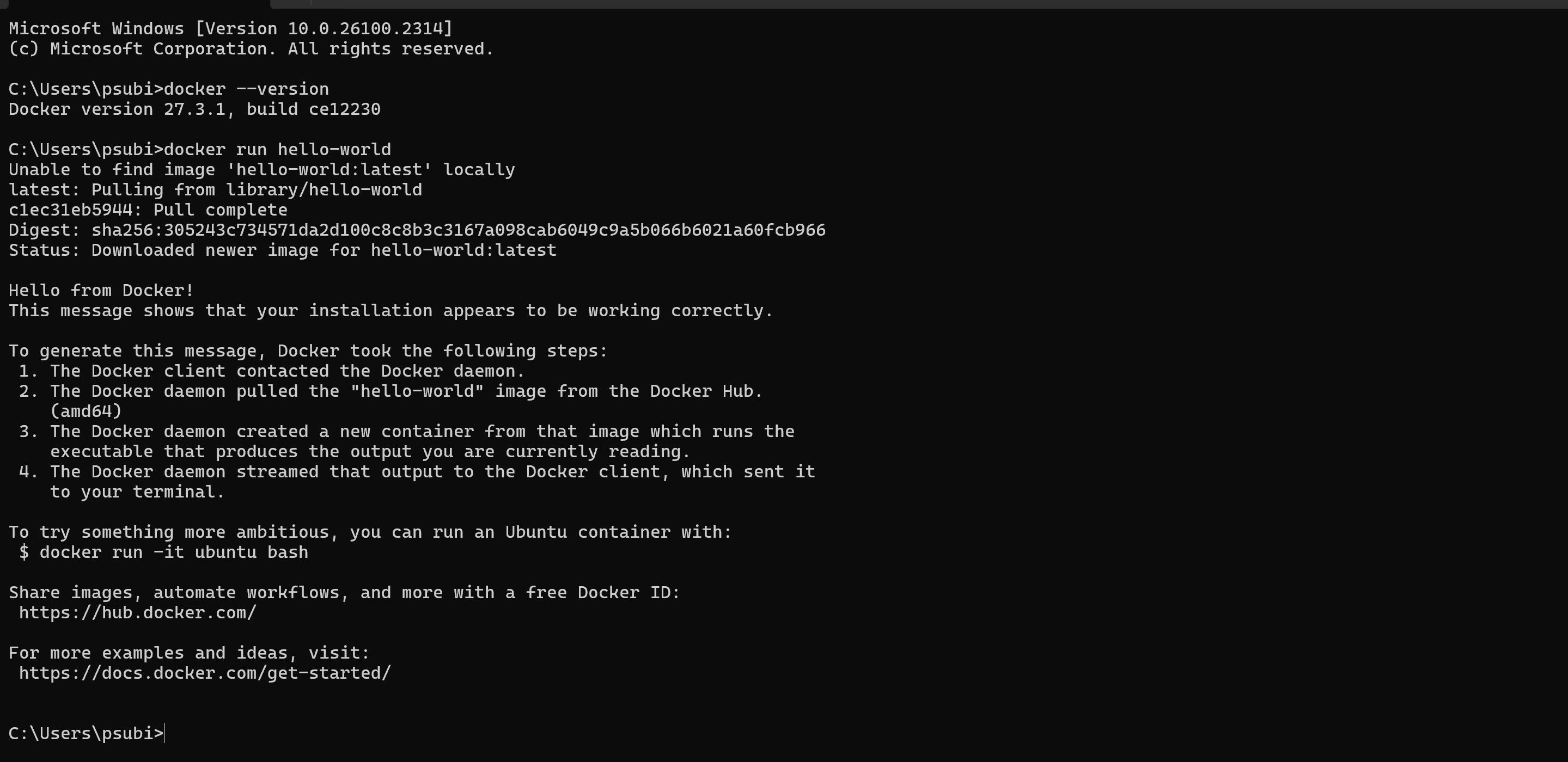
**Docker 기초 실습**

1번



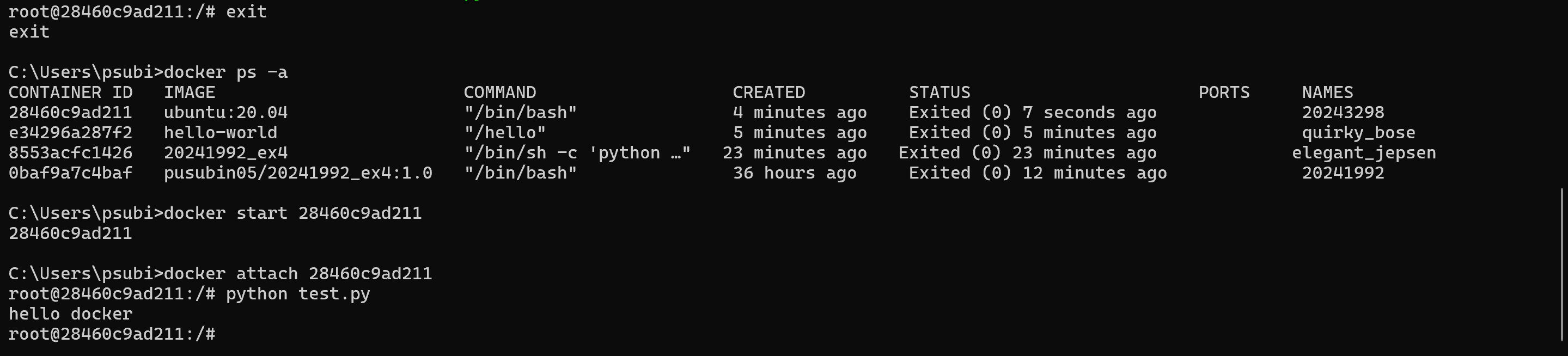
1. Docker 버전 확인: docker --version으로 설치된 Docker 버전 출력.

2. Docker 실행 테스트: docker run hello-world로 테스트, Docker Hub에서 이미지 다운로드 및 실행.

3. 출력 메시지: "Hello from Docker!"로 설치 및 작동 성공 확인.

4. 추가 명령어: docker run -it ubuntu bash로 Ubuntu 컨테이너 실행 가능.

2번



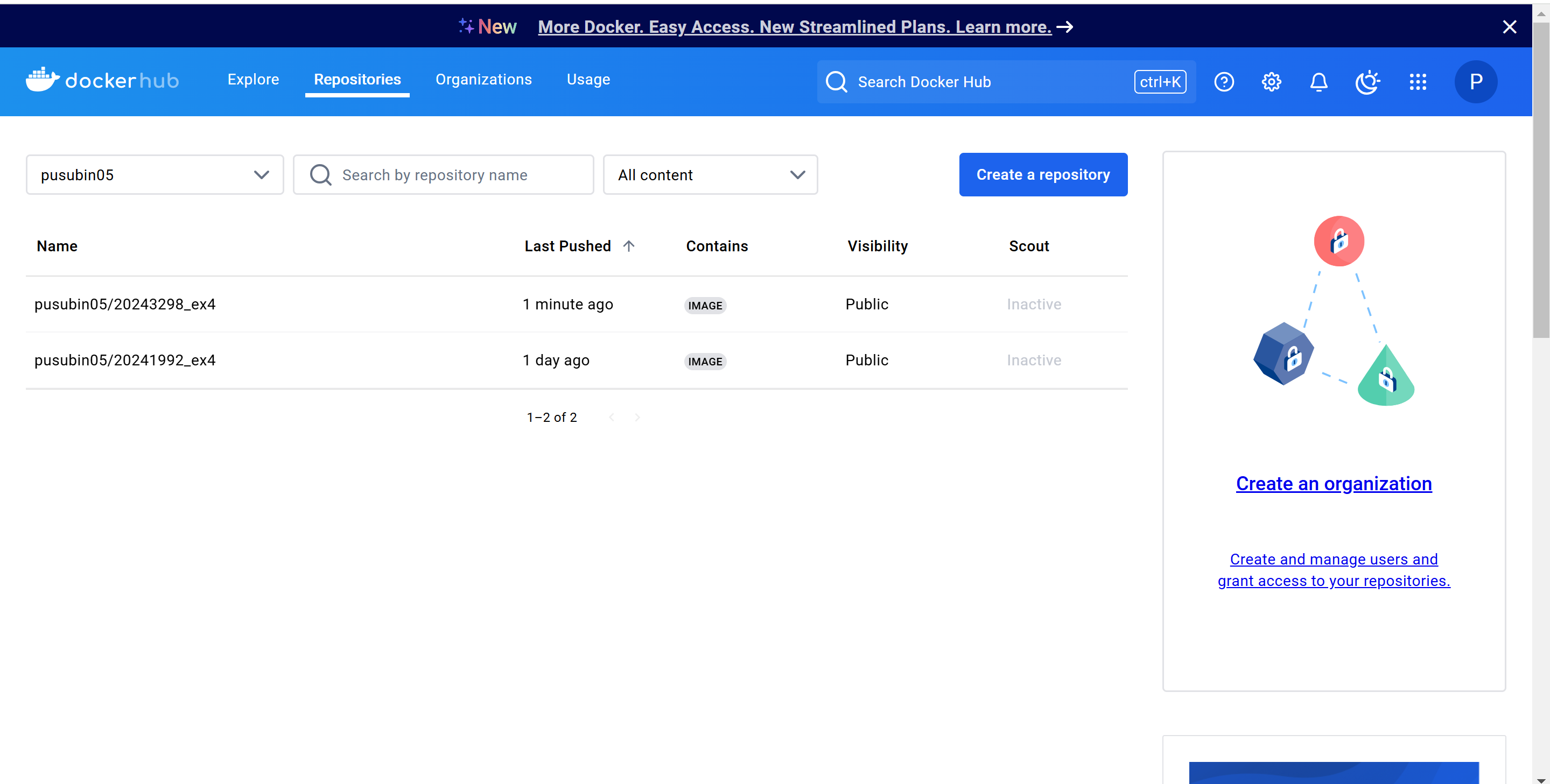
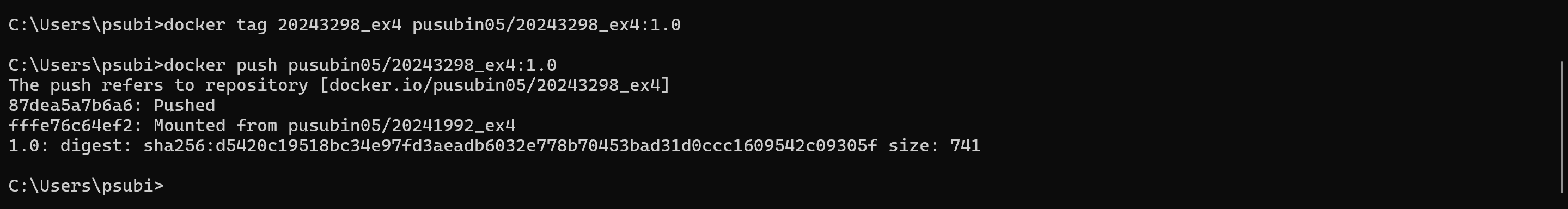
1. 컨테이너 확인: docker ps -a로 종료된 모든 컨테이너와 상태 확인.

2. 컨테이너 재시작: docker start <컨테이너 ID>로 종료된 컨테이너 재시작.

3. 컨테이너 재연결: docker attach <컨테이너 ID>로 컨테이너에 다시 연결.

4. Python 테스트: 컨테이너 내부에서 python test.py 실행하여 "hello docker" 출력 확인.

3번



1. 이미지 태그 지정: docker tag <이미지> 명령어로 로컬 이미지를 새로운 이름(pusubin05/20243298\_ex4:1.0)으로 태그.

2. 이미지 푸시: docker push pusubin05/20243298\_ex4:1.0 명령어로 Docker Hub에 해당 이미지를 업로드.

3. 푸시 결과: 이미지가 성공적으로 푸시되었고, Digest와 크기 정보 출력.

4. 목적: Docker Hub에 이미지를 저장해 다른 환경에서도 동일한 이미지를 사용할 수 있도록 공유.

4번



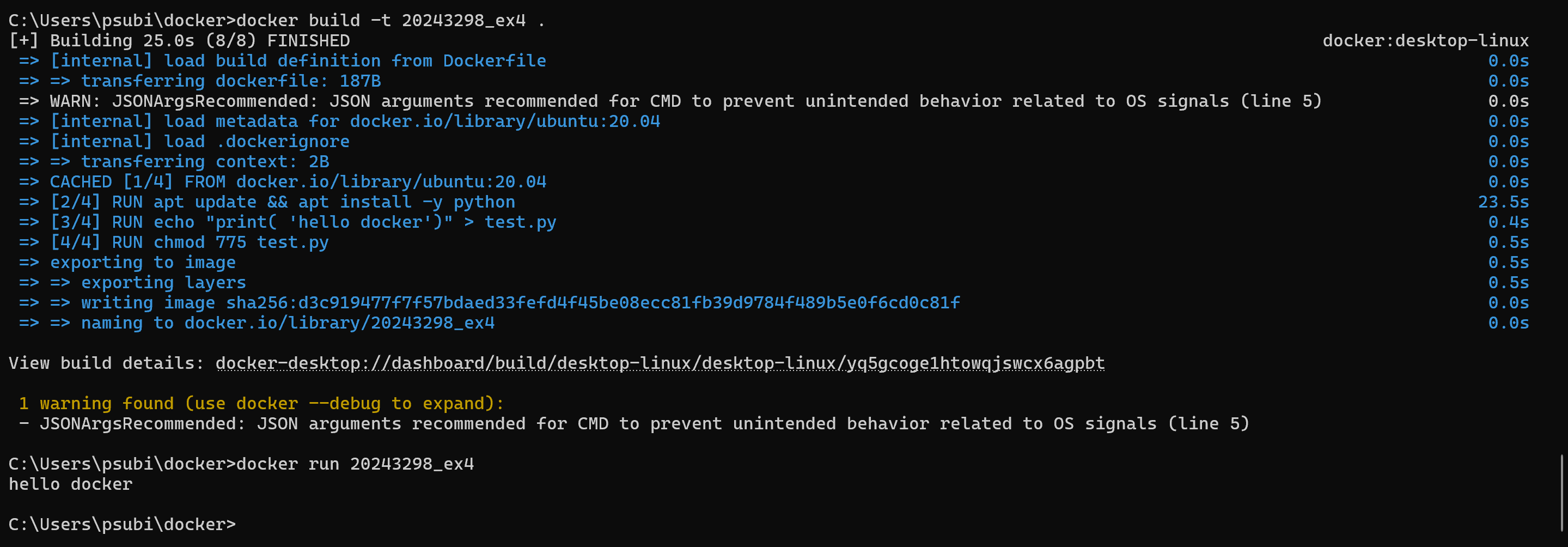
1. 컨테이너 실행: docker run -it --name <컨테이너 이름> <이미지> 명령어로 지정된 이미지(pusubin05/20243298\_ex4:1.0)로 컨테이너 생성 및 실행.

2. Python 스크립트 실행: 컨테이너 내부에서 python test.py 명령어 실행하여 "hello docker" 출력 확인.

3. 컨테이너 종료: exit 명령어로 컨테이너를 종료.

4. 목적: Docker 컨테이너에서 이미지 기반 환경을 실행하고 Python 스크립트를 테스트.

5번



1. 이미지 빌드: docker build -t 20243298\_ex4 . 명령어로 현재 디렉토리에 있는 Dockerfile을 사용해 이미지 생성.

2. 빌드 과정: Dockerfile 명령에 따라 Ubuntu 설치, Python 설치, test.py 복사 및 실행 명령어 포함.

3. 이미지 실행: docker run 20243298\_ex4 명령어로 빌드된 이미지를 실행하여 "hello docker" 출력 확인.