REFERAT DOCKER

fiabilitatea si testarea aplicatiilor software

Cuprins

[Docker 3](#_Toc135126977)

[1.1 Introducere 3](#_Toc135126978)

[1.2 Notiuni si termeni 3](#_Toc135126979)

[1.3 Procesul practic 5](#_Toc135126980)

[1.4 Intrebari 14](#_Toc135126981)

# Docker

## Introducere

Docker este o platforma de virtualizare de nivel superior care utilizeaza containerizarea pentru a ajuta la gestionarea si rularea aplicatiilor. Containerizarea este o tehnologie de virtualizare care permite rularea aplicatiilor in medii izolate, numite containere, care sunt separate de sistemul gazda si de alte aplicatii care ruleaza pe acelasi sistem. Aceasta abordare ofera o serie de beneficii, cum ar fi portabilitatea, scalabilitatea, securitatea si izolarea resurselor, permitand utilizatorilor sa ruleze aplicatii in orice mediu, fara a fi nevoie sa se preocupe de compatibilitatea cu alte aplicatii sau cu sistemul gazda.

Docker a adus o revolutie in domeniul dezvoltarii si livrarii de software, permitand dezvoltatorilor sa livreze aplicatii in mod rapid si eficient, utilizand un set standardizat de instrumente si servicii. In plus, Docker ofera o serie de beneficii, cum ar fi gestionarea si orchestrarea containerelor, depozitarea si distribuirea de imagini, precum si posibilitatea de a utiliza conturi de utilizator personalizate, numite Docker Hub.

Cu Docker, dezvoltatorii pot crea, gestiona si implementa aplicatii intr-un mod eficient si scalabil, folosind o serie de instrumente si servicii care sunt usor de utilizat si de inteles. Docker Compose permite dezvoltatorilor sa gestioneze cluster-uri de containere, permitandu-le sa creeze si sa gestioneze infrastructuri cloud si on-premise. In plus, Docker Swarm si Kubernetes sunt doua instrumente populare pentru orchestrarea containerelor Docker, permitand utilizatorilor sa gestioneze si sa scalabileze aplicatiile lor in mod eficient si automatizat.

Docker este o platforma puternica si populara in comunitatea de dezvoltare software, care ofera o abordare moderna si eficienta pentru gestionarea si rularea aplicatiilor. Cu o serie de instrumente si servicii standardizate, Docker este o alegere populara pentru dezvoltatorii care doresc sa livreze aplicatii in mod eficient si scalabil.

## Notiuni si termeni

Docker Daemon este procesul principal care ruleaza in fundal pe sistemul gazda si gestioneaza crearea, rularea si oprirea containerele Docker. Acesta se executa ca un serviciu si asculta comenzile API pentru a comunica cu daemon-ul Docker. De asemenea, Docker Daemon este responsabil pentru crearea si gestiunea imaginilor Docker si pentru stocarea acestora intr-un registru Docker.

Docker Client este interfata de linie de comanda folosita pentru a interactiona cu daemon-ul Docker. Acesta trimite cereri API la daemon-ul Docker si primeste raspunsurile corespunzatoare. Utilizatorii pot folosi Docker Client pentru a crea, gestiona si executa containere Docker, pentru a gestiona imagini Docker si pentru a comunica cu alte servicii asociate cu Docker.

Interfata grafica a Docker, numita Docker Desktop, ofera o modalitate grafica de a interactiona cu daemon-ul Docker, fara a fi necesar ca linia de comanda sa fie folosita. Acesta vine cu o interfata grafica pentru a gestiona containerele Docker, imaginile si alte componente asociate. De asemenea, Docker Desktop include instrumente pentru a seta optiuni de configurare pentru daemon-ul Docker, inclusiv setarea de retele si volumi pentru containere.

De asemenea, Docker expune patru tipuri de obiecte, fiecare avand un scop specific in ecosistem:

* Containere - obiectele de baza in Docker. Acestea sunt instante ale imaginilor Docker si contin toate componentele necesare pentru a rula o aplicatie sau un serviciu. Acestea sunt izolate de restul sistemului, permitand rularea aplicatiilor fara a interfera cu alte aplicatii sau servicii care ruleaza pe acelasi sistem.
* Imagini - sabloane read-only pentru containere. Acestea contin sistemul de fisiere si toate dependentele necesare pentru a rula aplicatia sau serviciul intr-un container. Imaginile sunt create de la containere sau de la Dockerfiles si sunt stocate intr-un registry, cum ar fi Docker Hub.
* Volumi - obiecte care permit stocarea si partajarea datelor intre containere. Acestea permit containerelor sa aiba acces la date persistente care supravietuiesc ciclurilor de viata ale containerelor.
* Retele - obiecte care permit containerele sa comunice intre ele sau cu alte servicii. Acestea permit configurarea unor topologii complexe de retea si pot fi folosite pentru a izola containerele in retea sau pentru a le conecta la o retea externa.

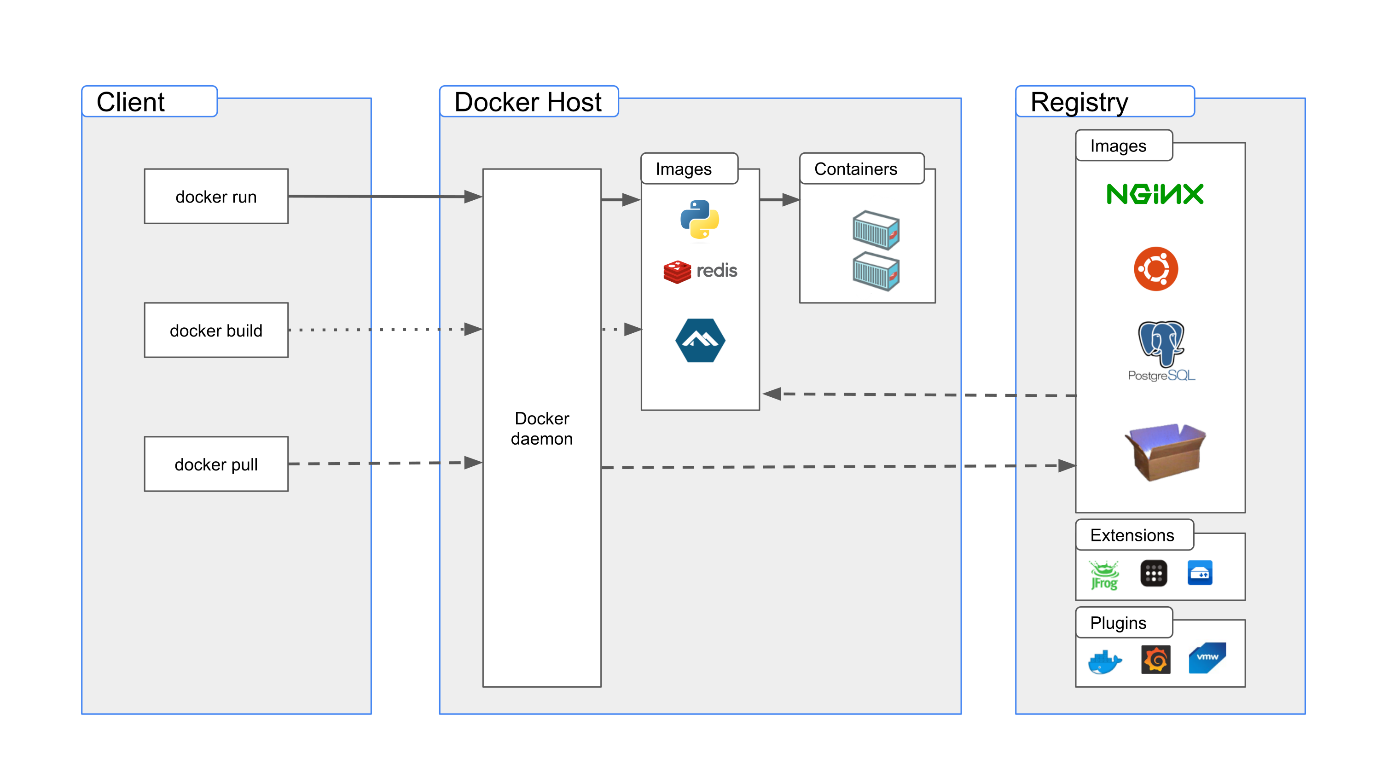


Diagrama arhitecturii Docker

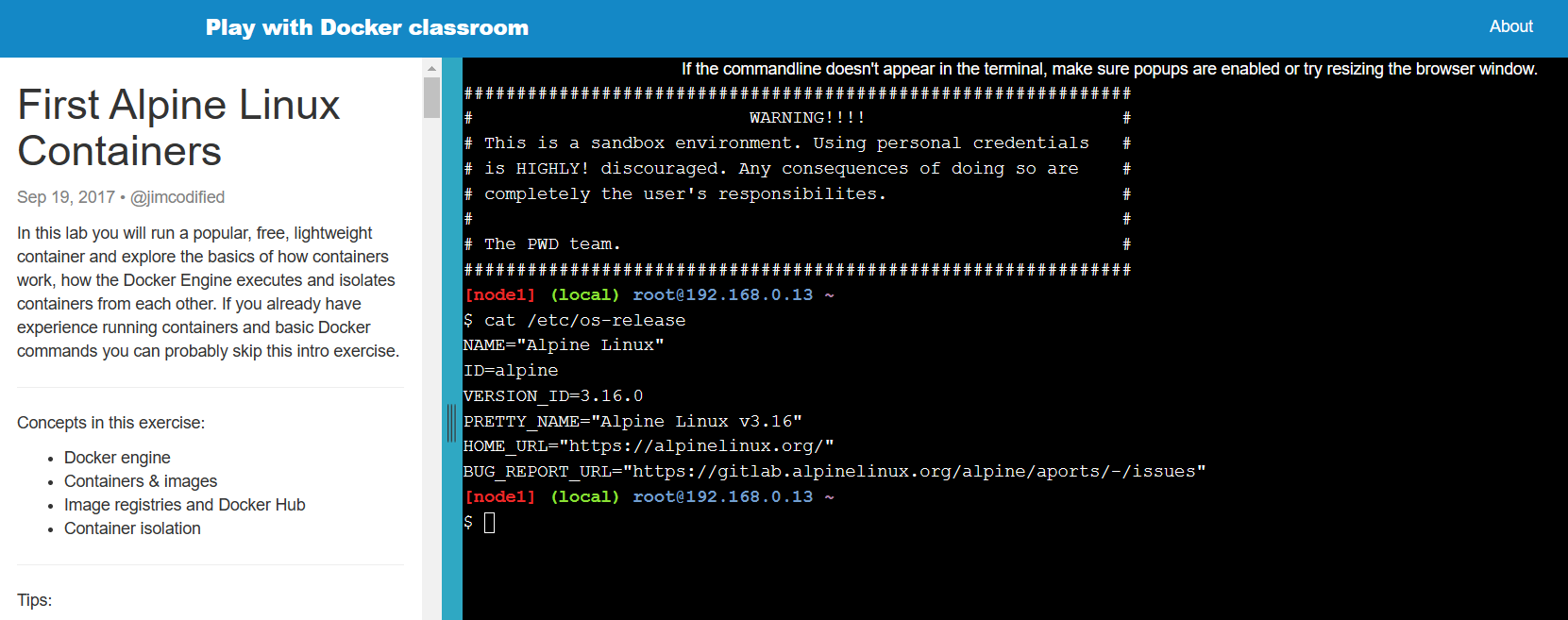
## Procesul practic

**Play with Docker classroom**

Play with Docker Classroom este o platforma online de invatare interactiva pentru Docker, oferind un mediu sandboxed si scenarii de laborator.

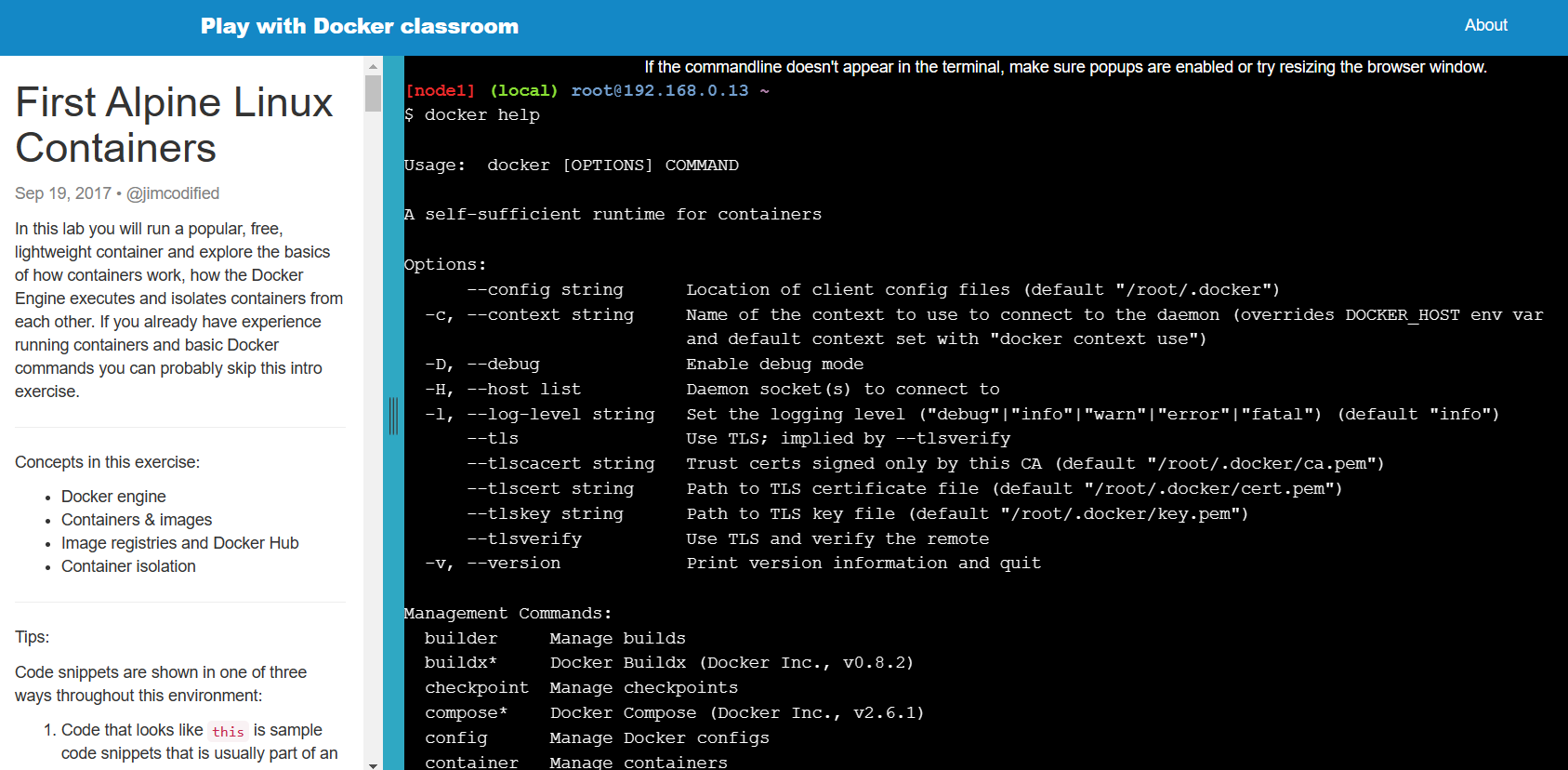
1. **Afisare versiune sistem operare**

Comanda "cat /etc/os-release" este folosita in sistemele de operare bazate pe Linux pentru a afisa informatii despre sistemul de operare instalat. Aceasta comanda afiseaza continutul fisierului care contine informatii precum numele distributiei Linux, versiunea distributiei si alte detalii relevante despre sistemul de operare.



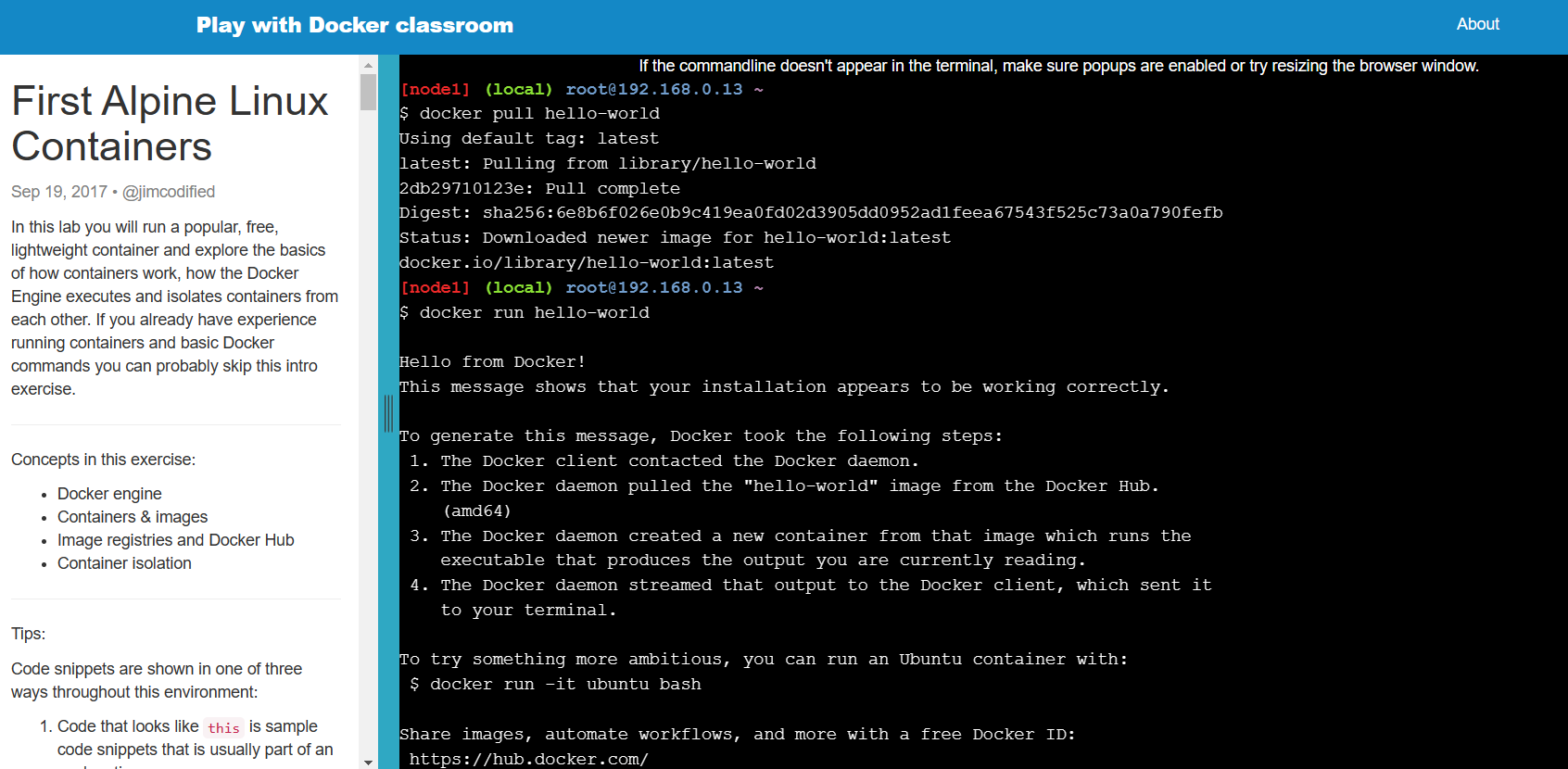
1. **Afisare lista comenzi**

Comanda "docker help" este utilizata in sistemul de operare Linux pentru a afisa informatii despre comenzile disponibile si modul in care acestea pot fi utilizate in platforma Docker.



1. **Verificare functionalitate**

Comanda "docker pull hello-world" este utilizata pentru a descarca imaginea "hello-world" din registrele Docker si pentru a o stoca local pe calculator. Comanda "docker run hello-world" este utilizata pentru a rula containerul asociat cu aceasta imagine si pentru a testa daca platforma Docker functioneaza in mod corect.

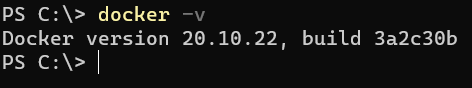


**Docker rulat prin terminal Powershell**

Docker poate fi rulat prin terminalul PowerShell utilizand comenzile specifice Docker, care permit utilizatorilor sa creeze, ruleze si gestioneze containere.

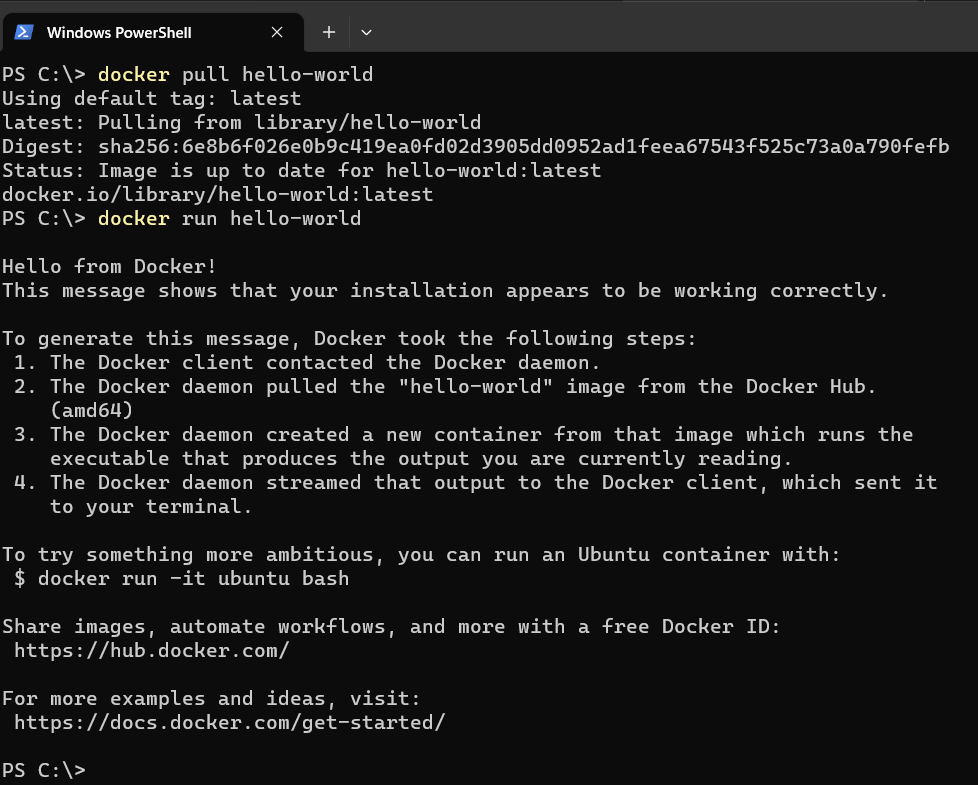
1. **Afisare versiune**

Comanda "docker -v" este utilizata pentru a afisa versiunea instalata a platformei Docker in sistemul de operare utilizat.



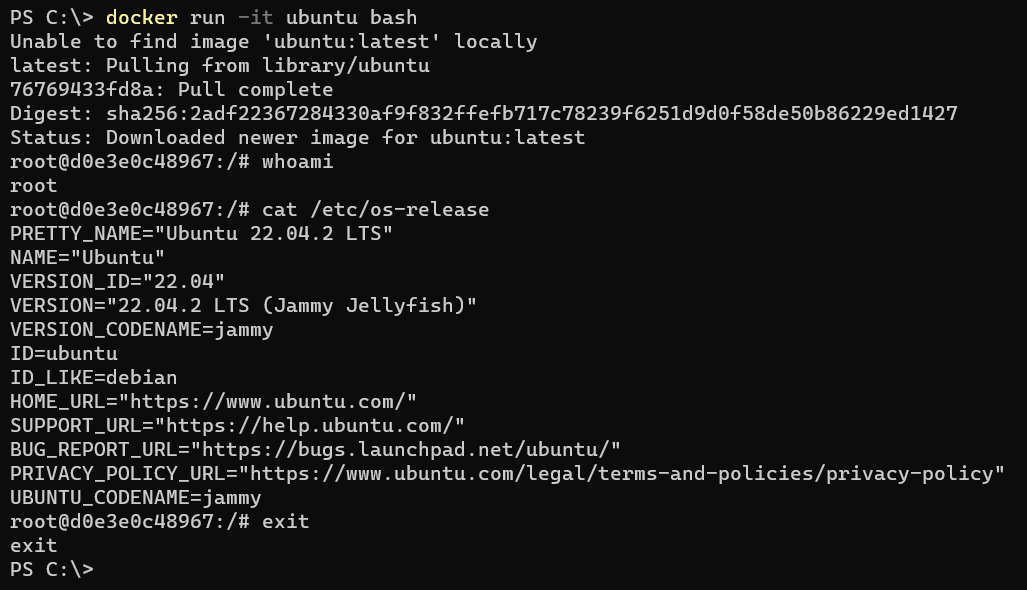
1. **Verificare functionalitate**

Acelasi mod de operare ca in cazul de mai sus, singura diferenta constand in rularea locala, prin intermediul unui terminal de tip Powershell.



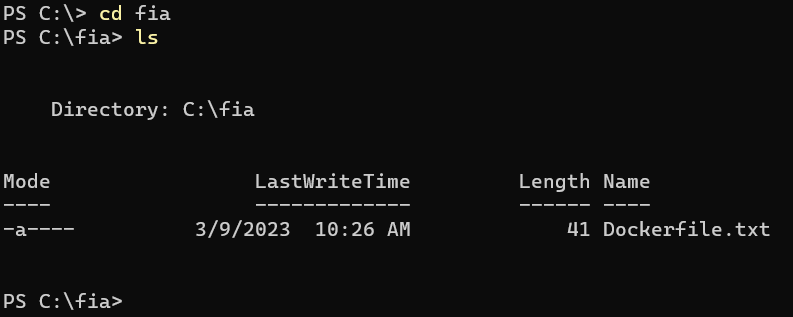
1. **Rulare container, afisare user si versiune sistem operare**

Comanda "docker run -it ubuntu bash" este utilizata pentru a rula un container bazat pe imaginea "ubuntu" si pentru a accesa terminalul interactiv al containerului prin intermediul comenzii "bash", comanda "whoami" este utilizata pentru a afisa numele utilizatorului curent logat in sistemul de operare iar ultima comanda este folosita in acelasi procedeu ca cel prezentat mai sus



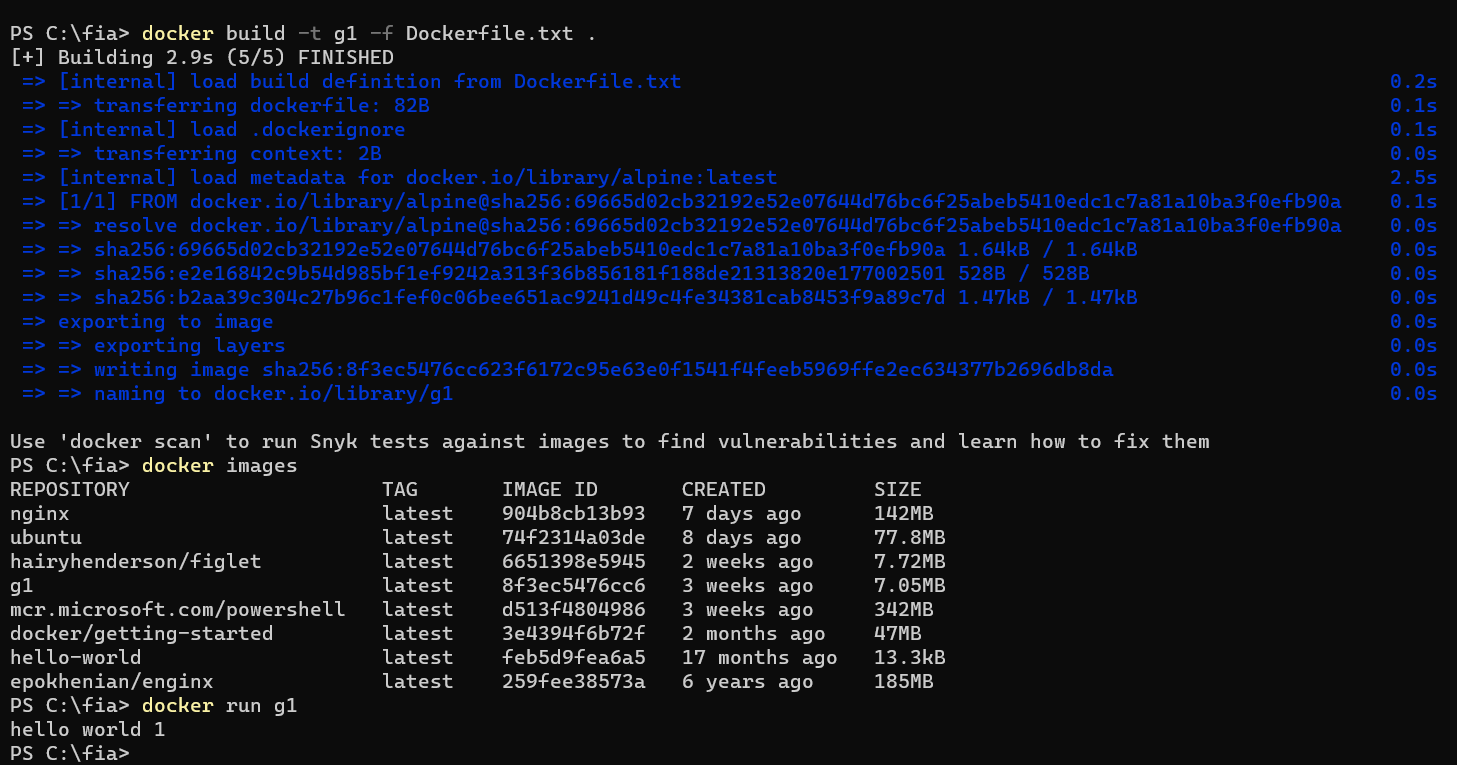
1. **Navigare si afisare fisere/foldere**

Comanda "cd fia" este utilizata pentru a naviga in folderul "fia" din calea curenta, in timp ce comanda "ls" afiseaza lista fisierelor si foldere din directorul curent.



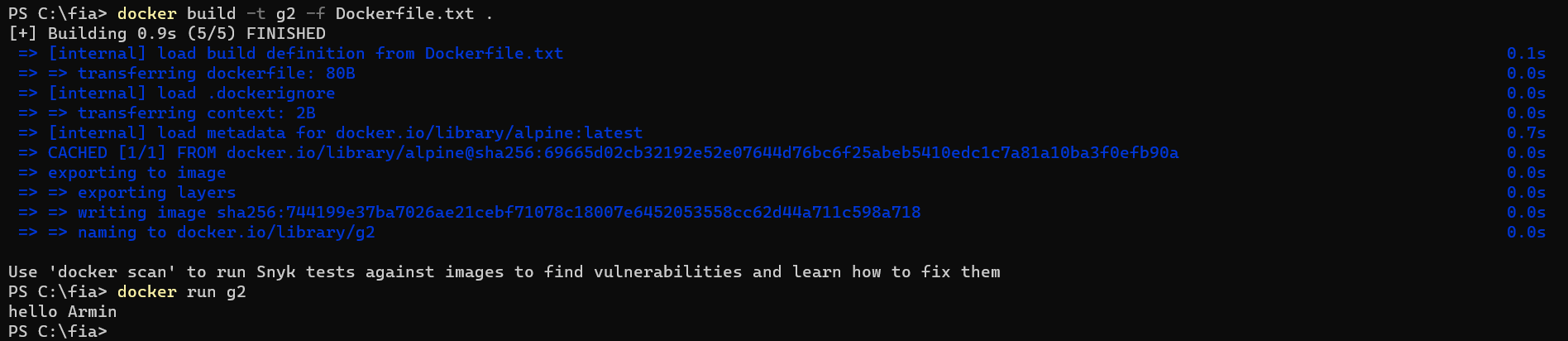
1. **Construire, afisare si rulare imagine**

Prima comanda, "docker build" construieste o imagine Docker cu numele "g1" folosind fisierul "Dockerfile.txt" din directorul curent, indicat de punctul de la sfarsitul comenzii. Cea de a doua comanda, "docker images" este folosita pentru verificarea existentei imaginilor disponibile local. Dupa ce imaginea a fost construita, containerul este rulat utilizand cea de a treia comanda, "docker run".



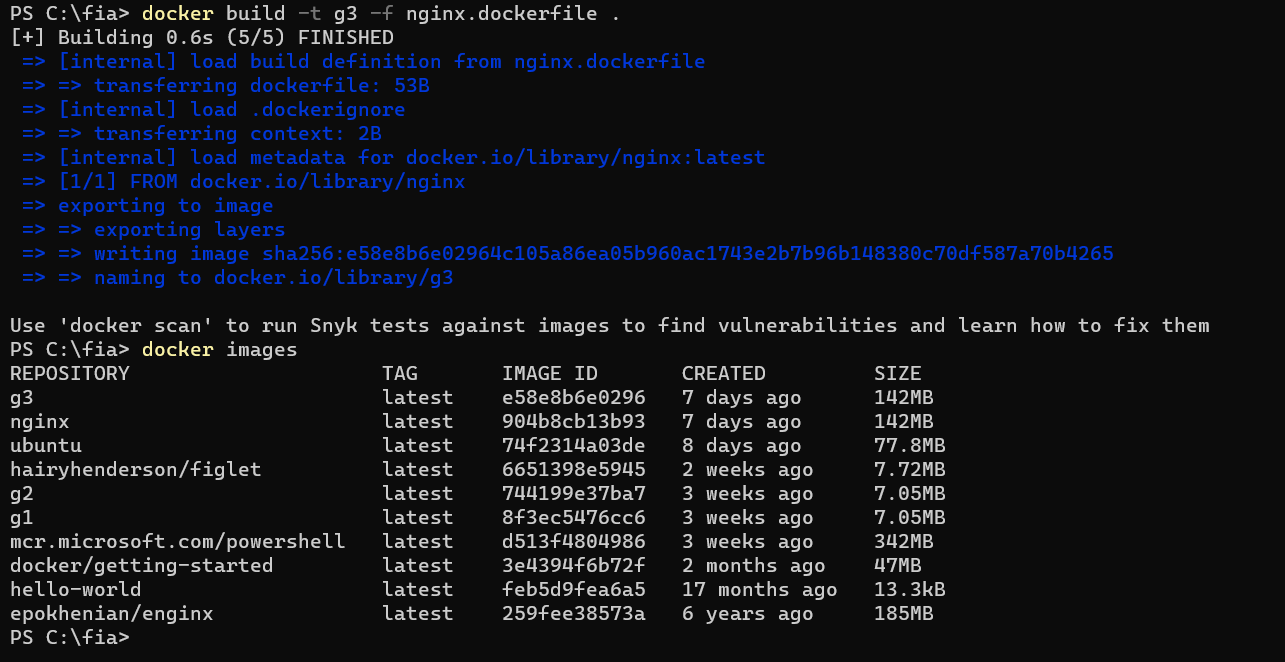
1. **Construire, afisare si rulare imagine aditionala**

Sub acelasi procedeu ca cel anterior, se va crea o a doua imagine aditionala, denumita "g2".



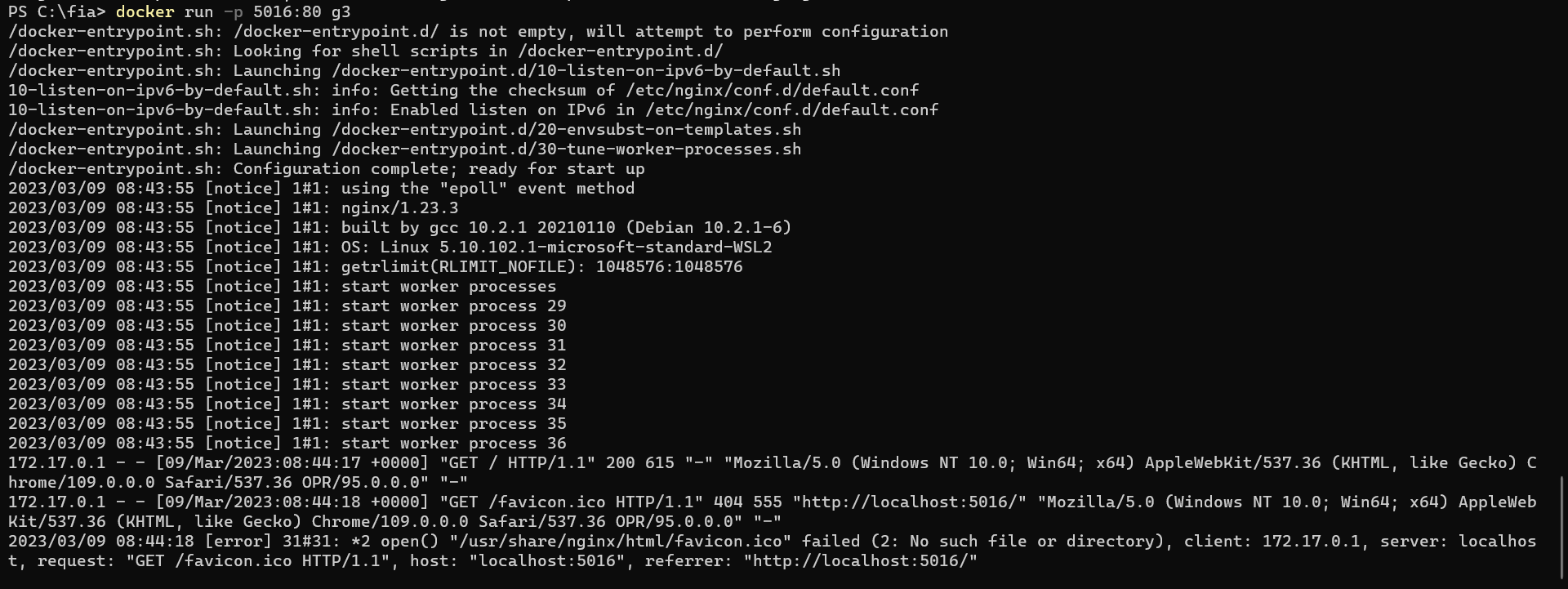
1. **Construire, afisare si rulare imagine aditionala**

Sub acelasi procedeu ca cele anteriore, se va crea o a treia imagine aditionala, denumita "g3", cu diferenta ca este folosit fisierul "nginx.dockerfile" din directorul curent.



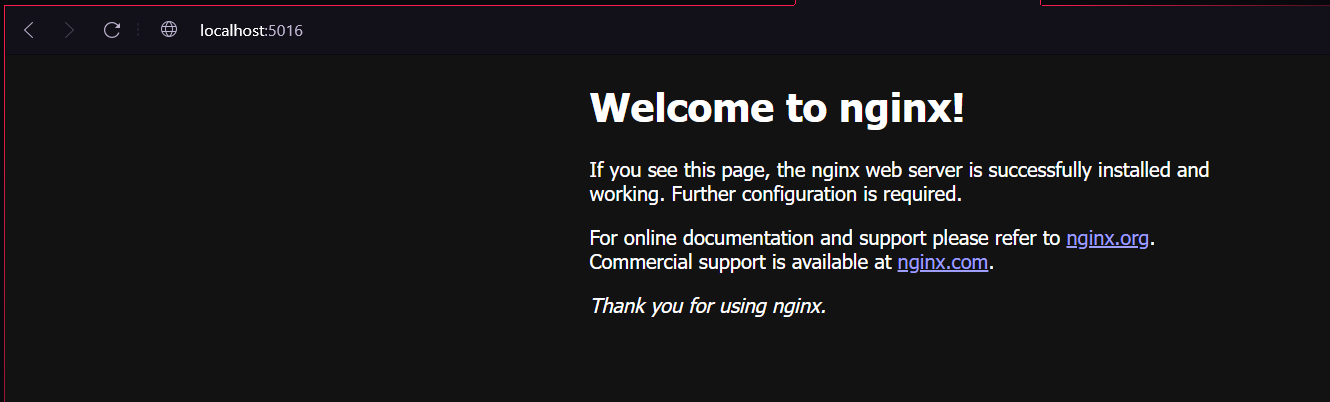
1. **Rularea unui container pentru accesarea la distanta**

Comanda "docker run -p 5016:80 g3" ruleaza un container folosind imaginea "g3" pe portul 80 si il leaga la portul 5016 local, al gazdei (16 in acest context este reprezentat de ID-ul de ordine al studentului). Aceasta reprezinta o modalitate de a face containerul disponibil pentru a fi accesat de la distanta prin intermediul browser-ului, daca este necesar.



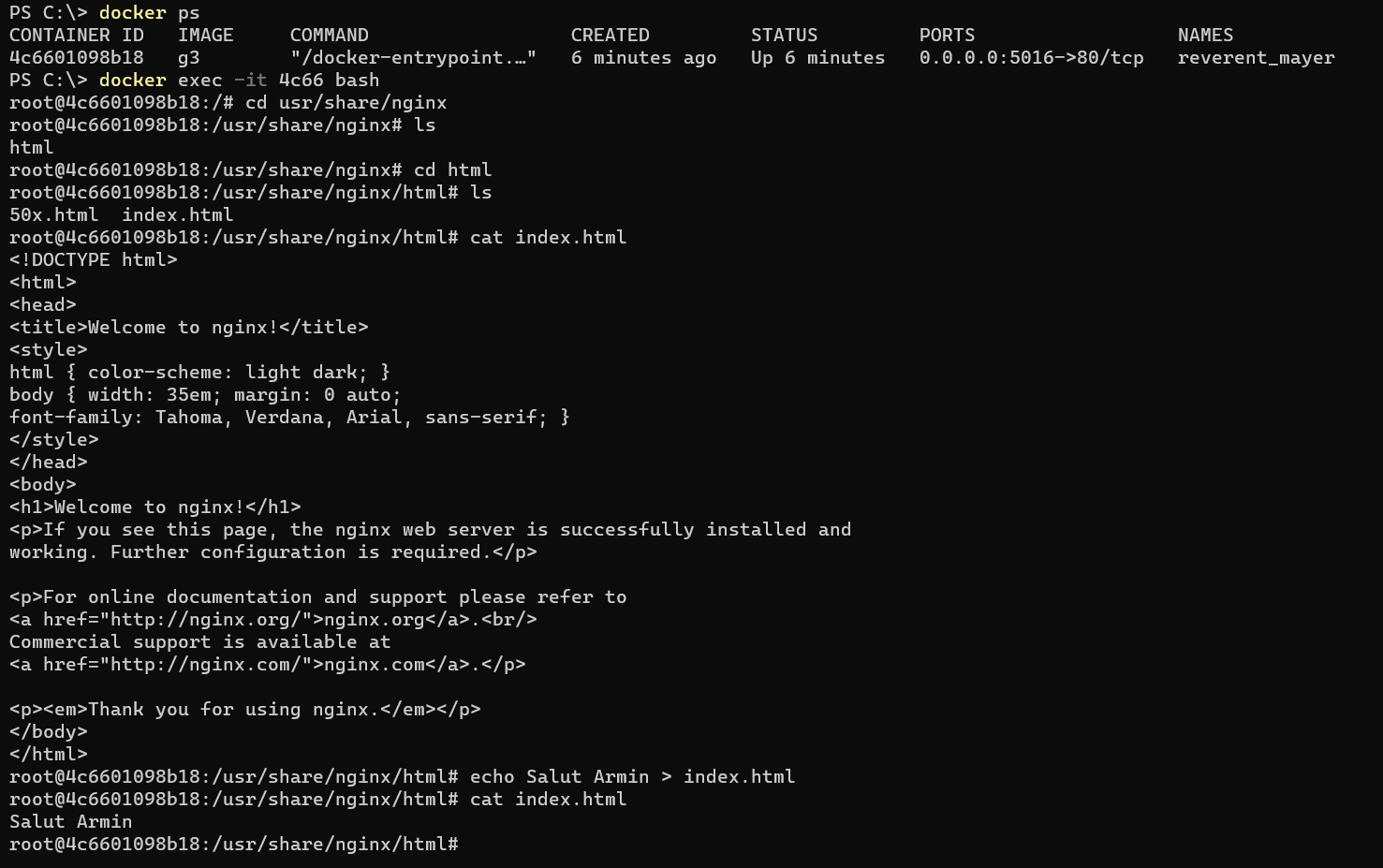
1. **Conectare la server Nginx**

Se efectueaza conectarea de pe browser prin adresa locala "localhost:5016" la serverul ngxinx.



1. **Gestionarea unui container Nginx cu Docker: listare, accesare si editare de fisiere prin terminal**

Comanda "docker ps" afiseaza toate containerele care ruleaza in prezent, "docker exec -it 4c66 bash" deschide o fereastra de terminal in containerul cu ID-ul 4c66, "cd usr/share/nginx" schimba directorul curent la /usr/share/nginx, "ls" afiseaza continutul directorului curent, "cd html" schimba directorul curent la /usr/share/nginx/html, "cat index.html" afiseaza continutul fisierului index.html, "echo Salut Armin > index.html" rescrie continutul fisierului index.html cu textul "Salut Armin", iar "cat index.html" afiseaza noul continut al fisierului index.html.



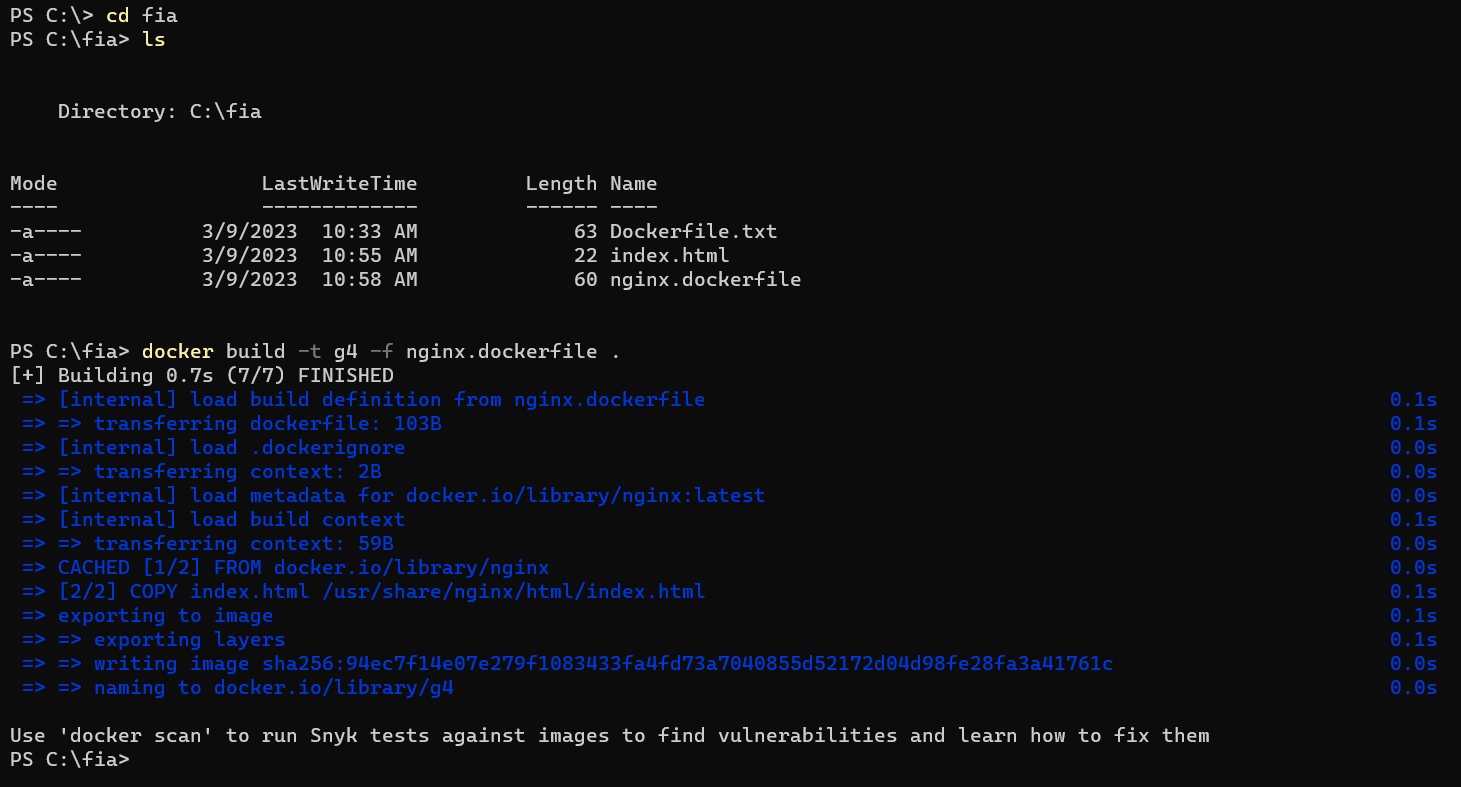
1. **Conectare la server Nginx in urma modificarilor**

Se efectueaza conectarea de pe browser prin adresa locala "localhost:5016" la serverul ngxinx, de aceasta data cu modificarile de mai sus efectuate.



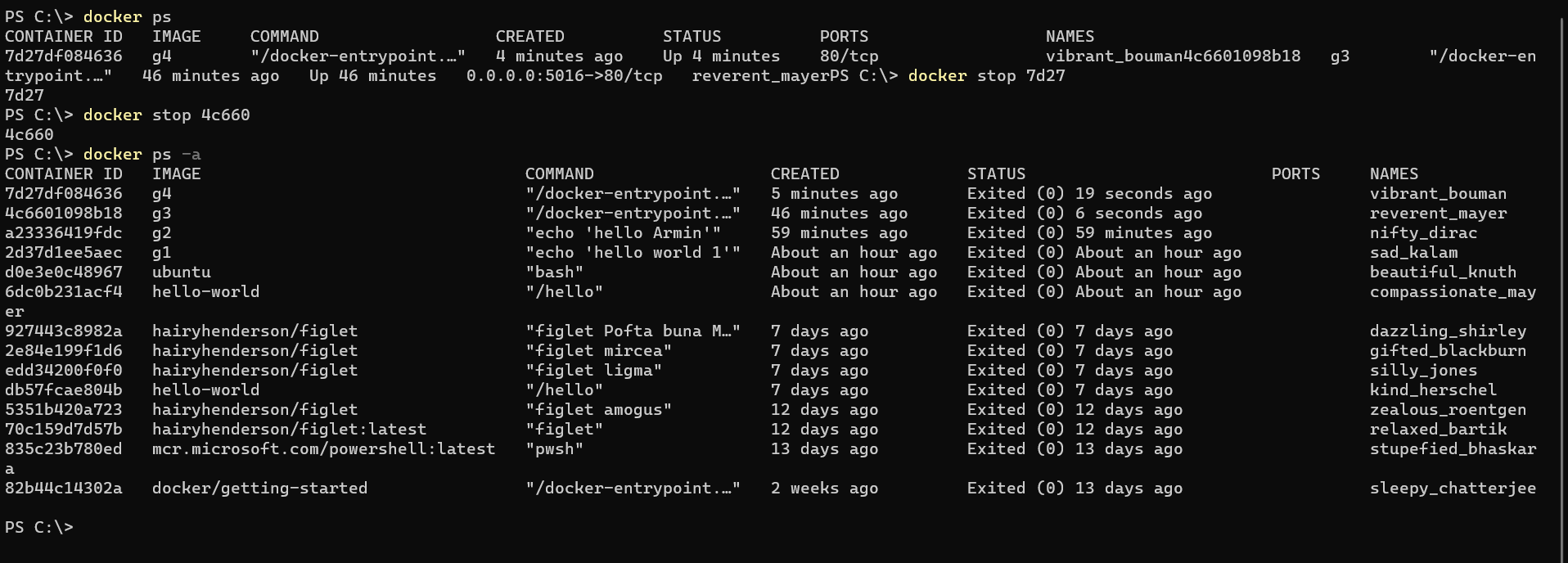
1. **Construire, afisare si rulare imagine aditionala**

Sub acelasi procedeu ca cele anteriore, se va accesa folderul "fia", se vor afisa fisierele disponibile, si in cele din urma se va crea o a patra imagine aditionala, denumita "g4", pe baza fisierului "nginx.dockerfile" din directorul curent.



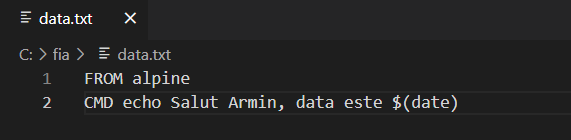
1. **Construire, afisare si rulare imagine aditionala**

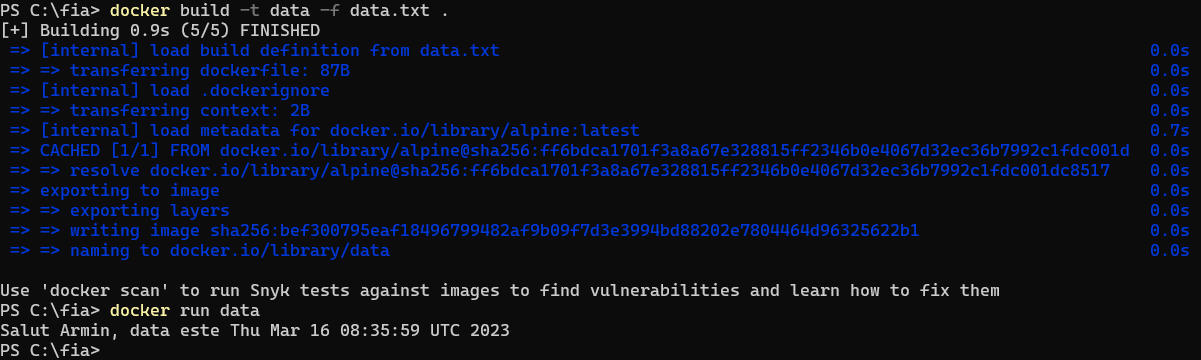
Comanda "docker ps" afiseaza containerele care ruleaza in prezent, impreuna cu informatii precum ID-ul, imaginea folosita, porturile expuse si alte detalii. Comanda "docker stop 4c660" opreste containerul cu ID-ul "4c660". Pentru a vedea toate containerele (inclusiv cele oprite), se foloseste comanda "docker ps -a". Aceasta comanda afiseaza informatii despre toate containerele existente, inclusiv cele oprite, impreuna cu ID-ul lor si alte detalii.



1. **Imagine personalizata cu data curenta**

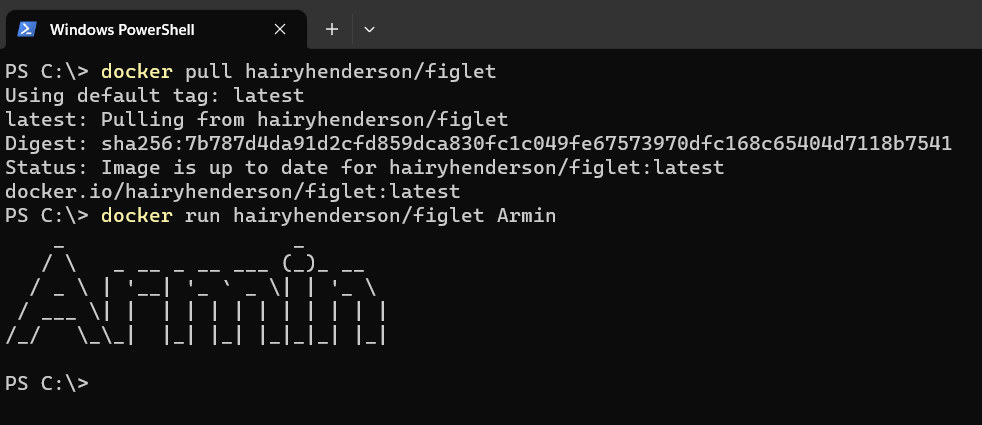
Comanda "docker build -t data -f data.txt ." construieste o noua imagine Docker denumita "data" folosind fisierul de construire "data.txt" din director. Acest fisier include o instructiune "FROM" care specifica imaginea de baza "alpine" (o imagine de baza care serveste ca sistem de operare minimal si contine doar pachetele de baza necesare pentru rularea aplicatiilor), si o instructiune "CMD" care afiseaza un mesaj "Salut, Armin data este" urmat de data curenta "$(date)", atunci cand containerul este rulat. Dupa ce imaginea este construita, este rulat un container utilizand comanda "docker run data". Aceasta comanda va crea un container din imaginea "data" si va afisa mesajul "Salut, Armin data este" urmat de data curenta.





1. **Construire, afisare si rulare imagine aditionala**

Comanda "docker pull hairyhenderson/figlet" descarca imaginea "hairyhenderson/figlet" care contine un utilitar numit Figlet ce permite afisarea de text in stil ASCII artistic. Comanda "docker run hairyhenderson/figlet Armin" ruleaza un container din aceasta imagine si afiseaza textul "Armin" sub forma de arta ASCII utilizand utilitarul Figlet.



## Intrebari

**1.** Care este diferenta intre Docker si Vagrant?

Docker este o platforma de virtualizare la nivelul sistemului de operare, care utilizeaza containere pentru a impacheta si rula aplicatii impreuna cu dependintele lor, avand un accent pe portabilitatea si reproducibilitatea aplicatiilor. Pe de alta parte, Vagrant este o unealta de gestionare a mediilor virtuale, care se bazeaza pe tehnologii de virtualizare precum VirtualBox sau VMware pentru a rula masini virtuale izolate, oferind un mediu de dezvoltare uniform si reproducibil. In timp ce Docker se concentreaza pe containere si distributia aplicatiilor, Vagrant se axeaza pe crearea si configurarea mediilor de dezvoltare portabile.

**2.** Ce tipuri de hypervisors exista?

Exista doua categorii principale:

* Tipul 1: Sunt instalate direct pe hardware-ul fizic al sistemului gazda, fara a avea nevoie de un sistem de operare gazda separat. Acestea ofera performante si control maxim asupra resurselor hardware. Exemple: VMware ESXi, Microsoft Hyper-V, Xen și KVM (Kernel-based Virtual Machine).
* Tipul 2: Ruleaza pe un sistem de operare gazda existent si sunt instalate ca aplicatii pe acesta. Acestea utilizeaza resursele si serviciile sistemului de operare gazda pentru a gestiona masinile virtuale. Exemple: VMware Workstation, Oracle VirtualBox si Microsoft Virtual PC.