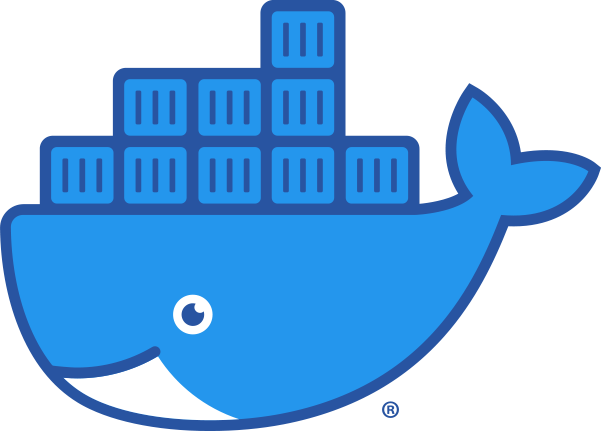
Federico Adria Carrasco

2º DAM Sistemes de gestió empresarial

**UD01.(Presencial) Activitats avaluables 01**



Índex

[Part 03. Cas pràctic 02 - Instal·lant LAMP + WordPress en un contenidor 2](#_Toc82859481)

[Part 03. Cas pràctic 03 - Accedint a la interfície gràfica amb NoVNC 3](#_Toc82859482)

[Part 04. Cas pràctic 01 - Creant imatge Ubuntu amb nano 4](#_Toc82859483)

[Part 04. Cas pràctic 02 - Creant imatge amb APP en Node 5](#_Toc82859484)

[Part 05. Cas pràctic 01 - Wordpress + MySQL 6](#_Toc82859485)

[Part 06. Cas pràctic 01 - Wordpress amb Docker Compose 7](#_Toc82859486)

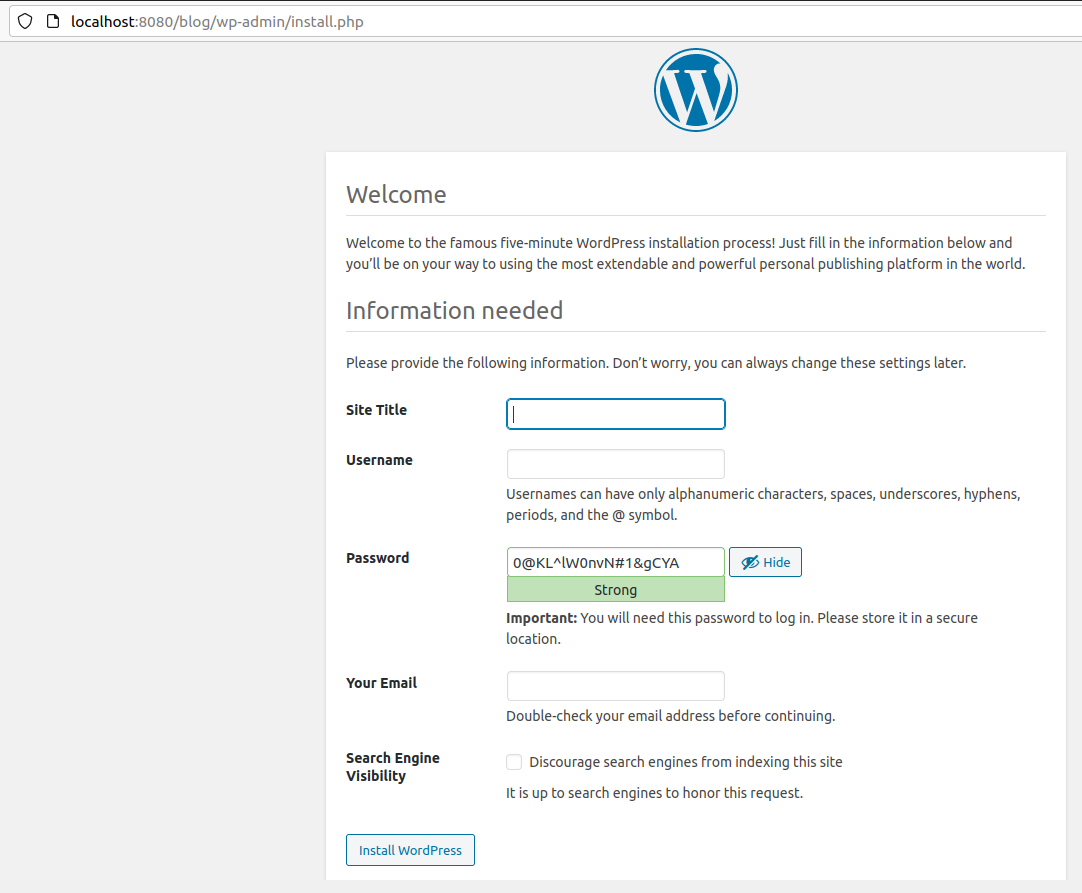
[Part 07. Cas pràctic 01 - Desenvolupant amb Visual Studio Code en un contenidor. 9](#_Toc82859487)

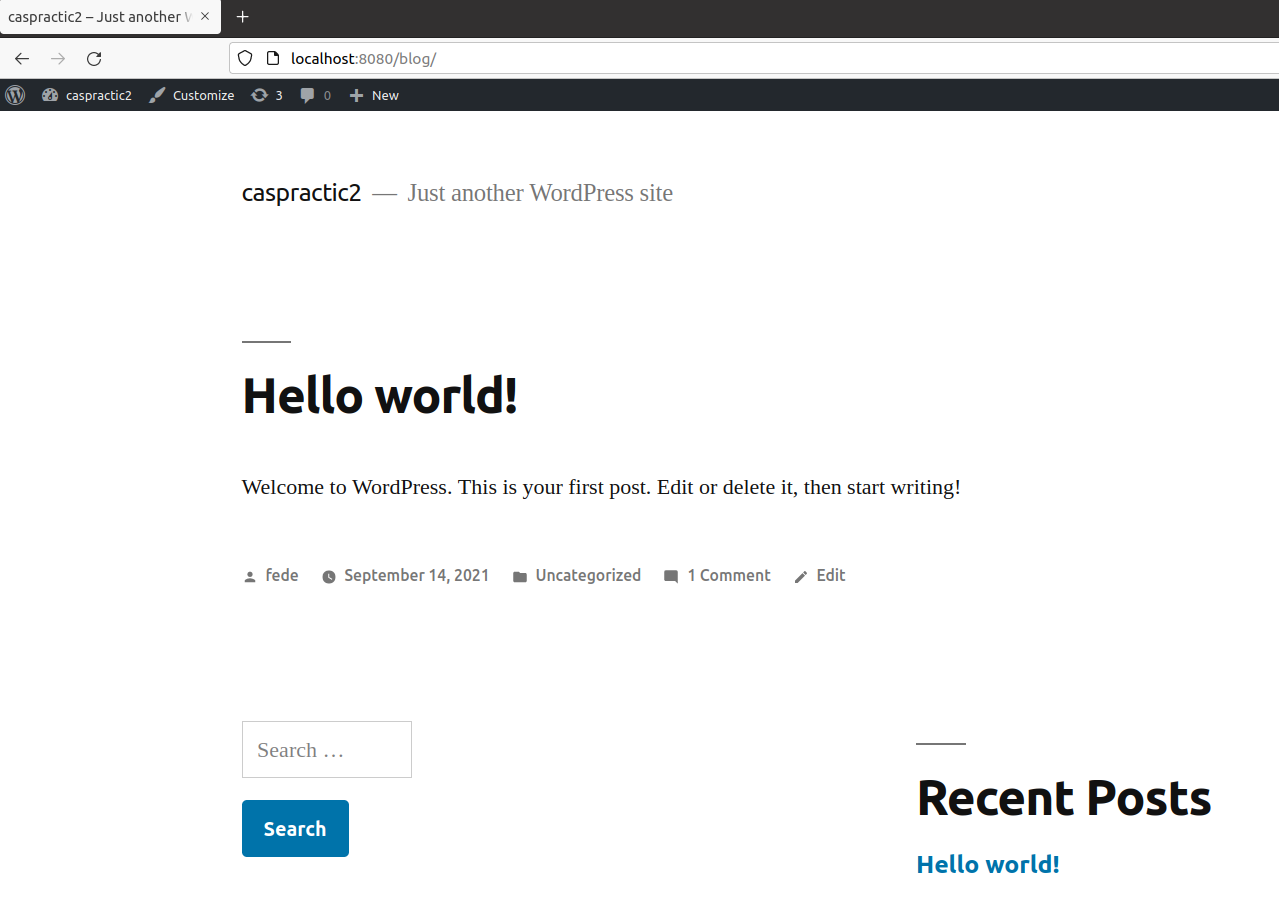
[Part 07. Cas pràctic 02 - Gestionant Docker amb Portainer CE 10](#_Toc82859488)

[Activitat 03: Resum de Kubernetes 13](#_Toc82859489)

# Part 03. Cas pràctic 02 - Instal·lant LAMP + WordPress en un contenidor

Aportem les imatges de la pàgina instal·lada abans i després d’omplir les dades necessàries per a accedir a la pàgina web:

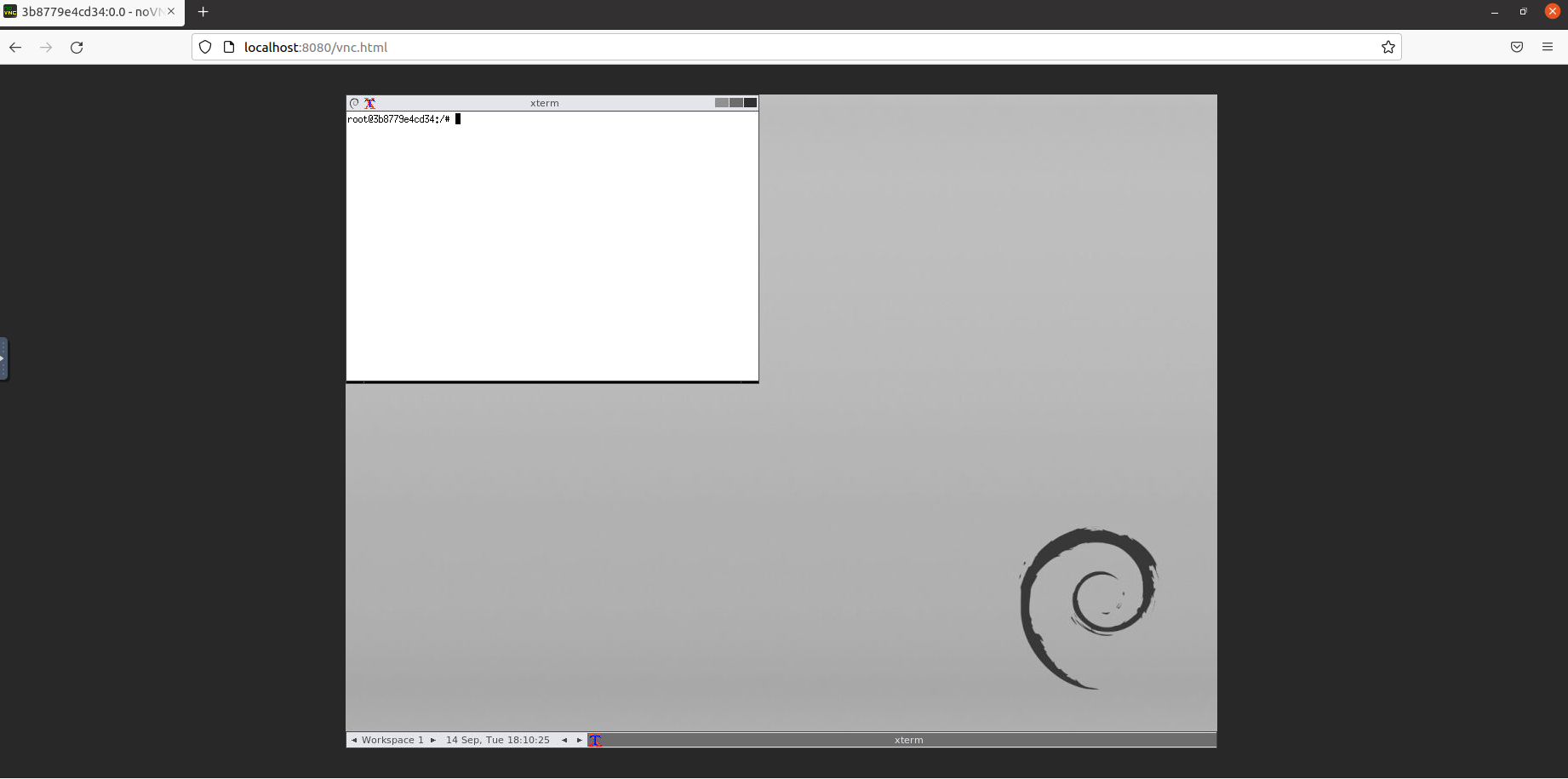




# Part 03. Cas pràctic 03 - Accedint a la interfície gràfica amb NoVNC

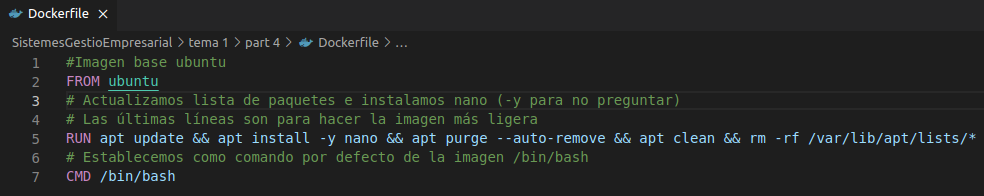
Es poden veure les imatges de com usem el servici noVNC al mateix port que l'exercici d’abans amb WordPress, 8080, per al que és necessari que usem docker stop LAMP per a tancar la màquina, ja que no podem tindre dues màquines al mateix port:



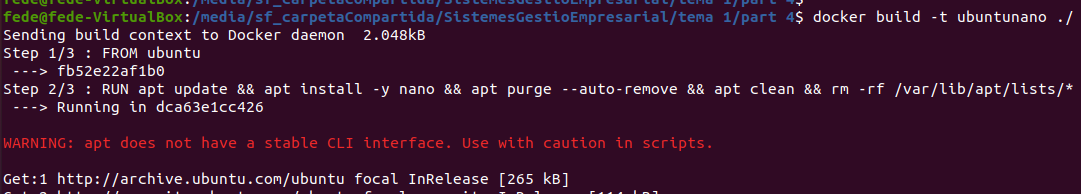


# Part 04. Cas pràctic 01 - Creant imatge Ubuntu amb nano

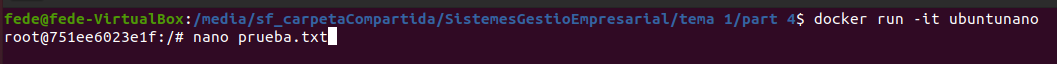
Primer creem un fitxer anomenat Dockerfile amb les instruccions que volem executar. Per exemple, nosaltres anem a usar Visual Studio Code:

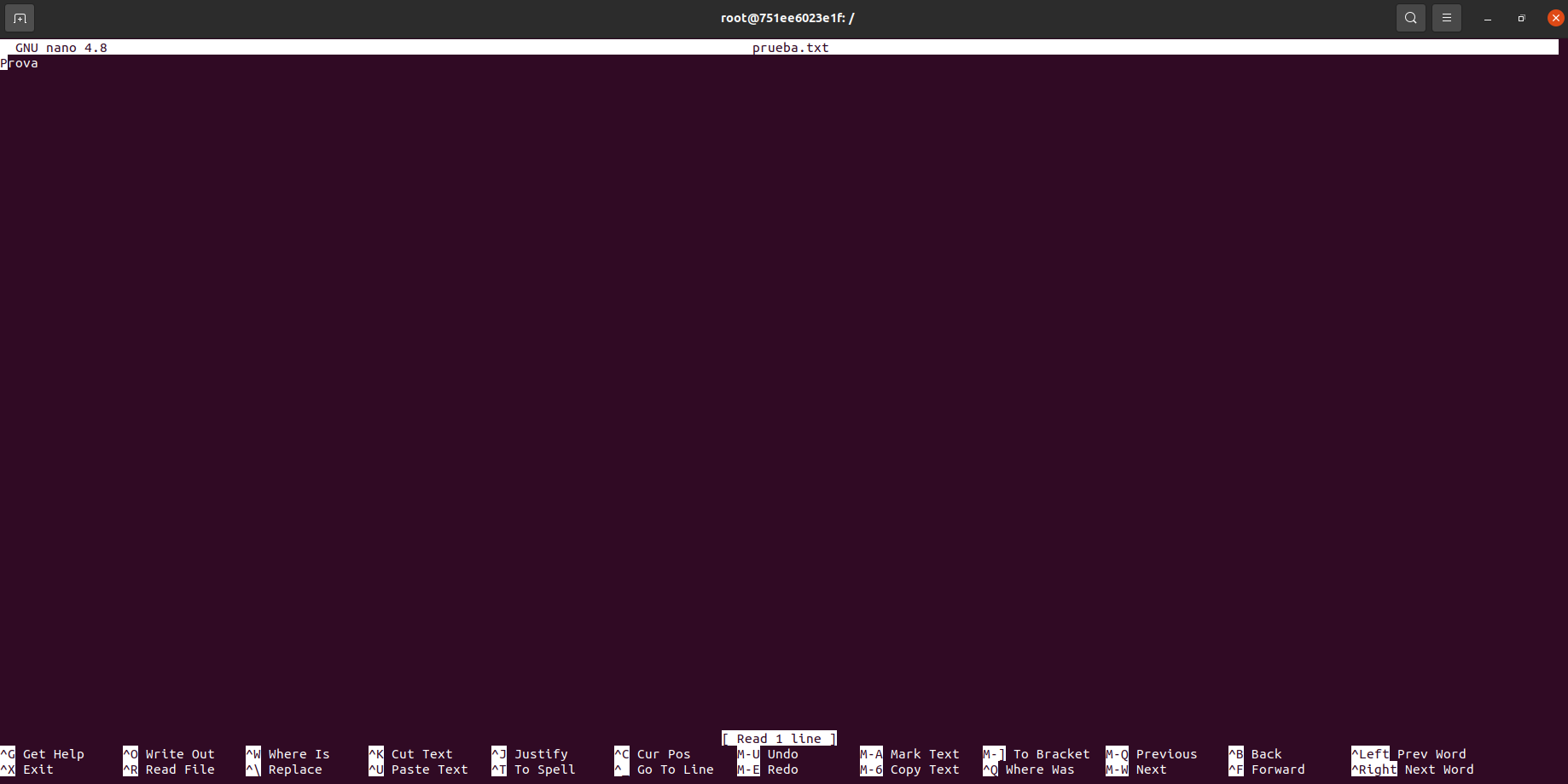


En segon llogar, anem a crear la imatge amb el següent comande on indiquem que el nom de la nova imatge serà ubuntunano i usarem el Dockerfile ubicat en el directori on executem el comande:



Una vegada finalitzen de crear la imatge, el següent pas serà crear el contenidor i fer una xicoteta prova de què finalment, sí que tenim nano instal·lat amb la imatge que hem usat:





# Part 04. Cas pràctic 02 - Creant imatge amb APP en Node

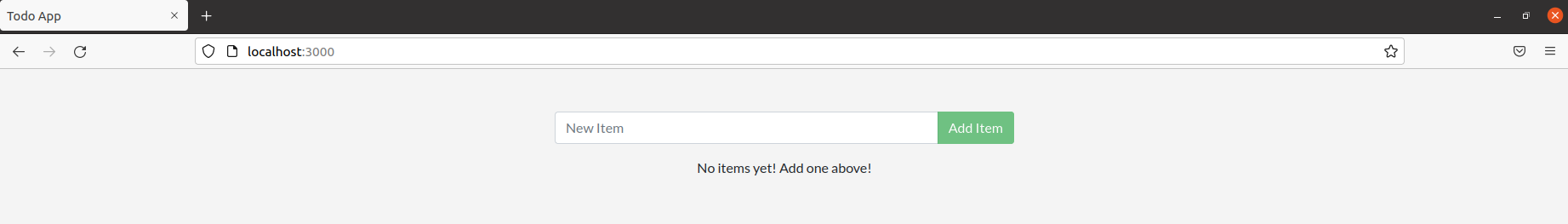
En primer lugar, descarregarem el fitxer des de el següent [enllaç](https://github.com/docker/getting-started/tree/master/app). És molt important que ubiquen el Dockerfile en aquesta carpeta. Una vegada fet, executarem el següent comande:



Finalment, quan usem el següent comande, usarem -d per a indicar que siga un procés en segon pla i -p per a indicar el port del amfitrió i de la màquina virtual:



Una vegada fet, la pàgina web estarà disponible a <http://localhost:3000/>:



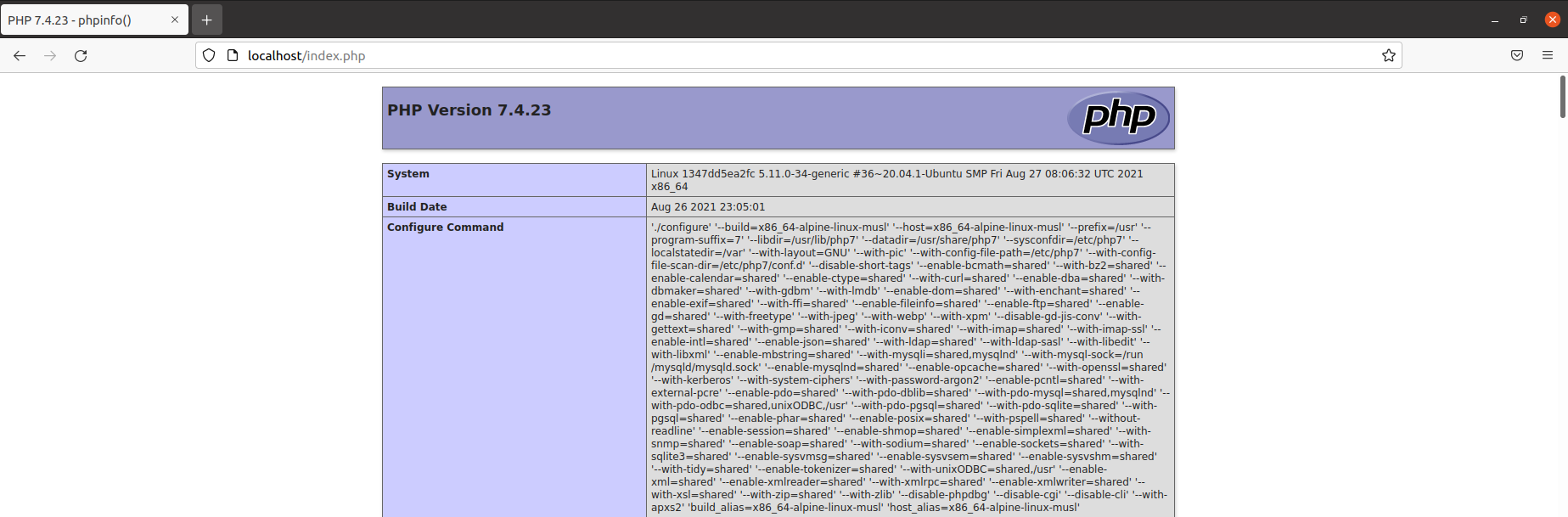
**UD 04. Cas pràctic 03 - Apache2 amb PHP usant Alpine**

En primer lloc, hem de crear el Dockerfile i el fitxer start.sh que tenim en els apunts del tema. Crearem la imatge amb el següent comande:



Finalment, llançarem l’aplicaciò i podriem accedir a ella en l’url localhost/index.php:



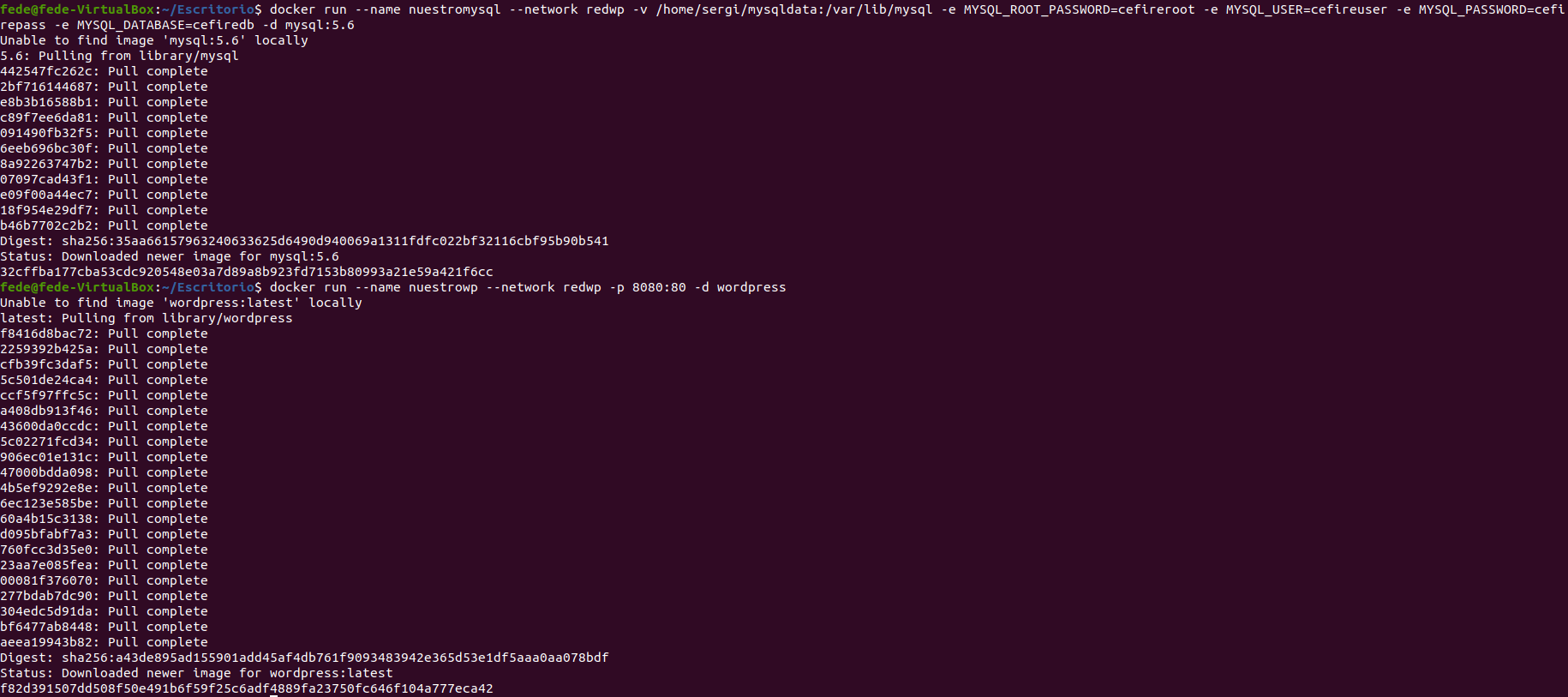


# Part 05. Cas pràctic 01 - Wordpress + MySQL

En aquest exercici, usarem una xarxa, ja que usarem dos contenidors, ho farem con el comande:



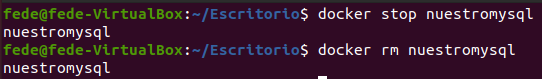
Crearem els dos contenidors en la mateixa xarxa.



Accedim a l’URL localhost:8080 y omplim els camps amb les següents dades:



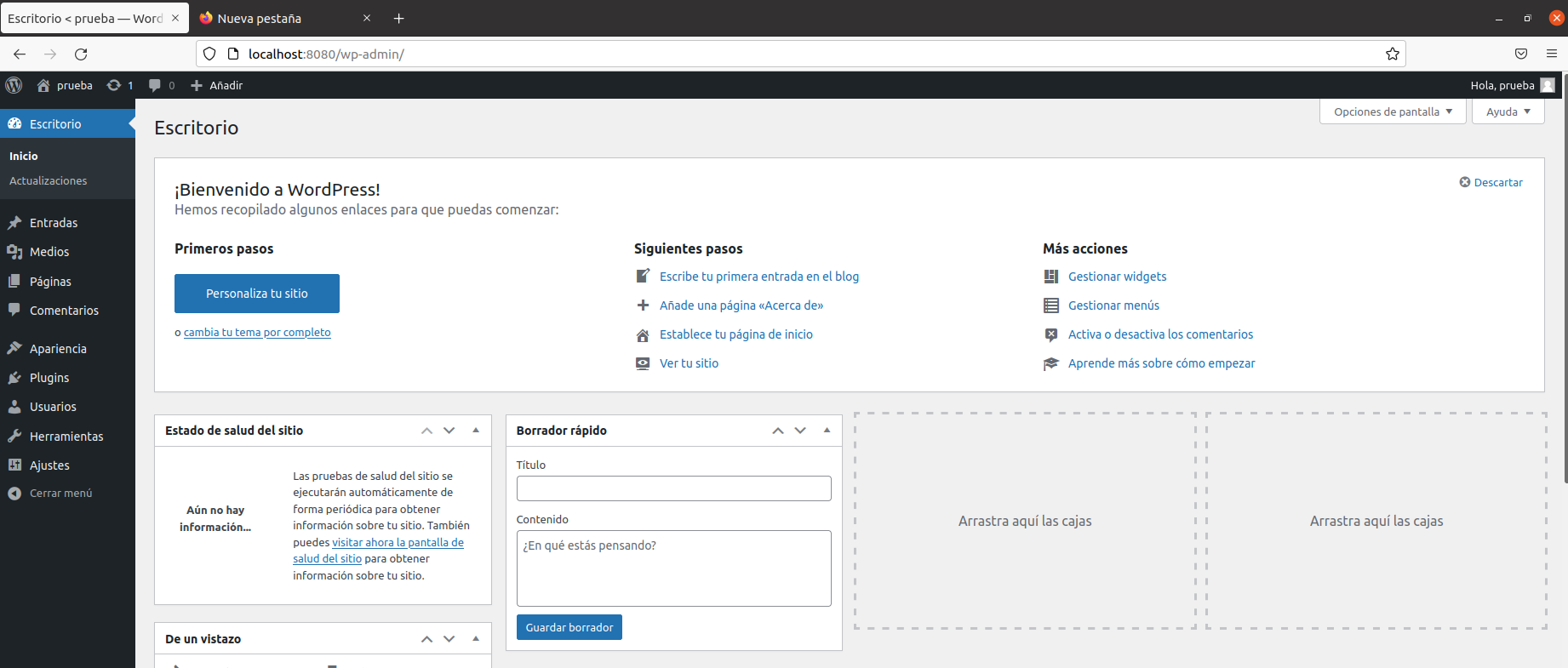
Amb això, ja podríem configurar i usar WordPress com hauríem normalment. El següent pas que anem a fer, és actualitzar el MySQL a una versió mas nova. Primer pararem el servici i asprés l’eliminarem.



A primera vista, podem pensar que l’informació de les bases de dades ha sigut eliminada, però quan crearem el contenidor MySQL vam indicar que l’informació la tindríem en una altra carpeta, pel que anem a copiar-la de nou:



Asprès fer això, wordpress funciona normalment, no tenim que repetir ninguna acciò en la base de dades:



# Part 06. Cas pràctic 01 - Wordpress amb Docker Compose

Asprés fer això, WordPress funciona normalment, no hem de repetir cap acció en la base de dades:

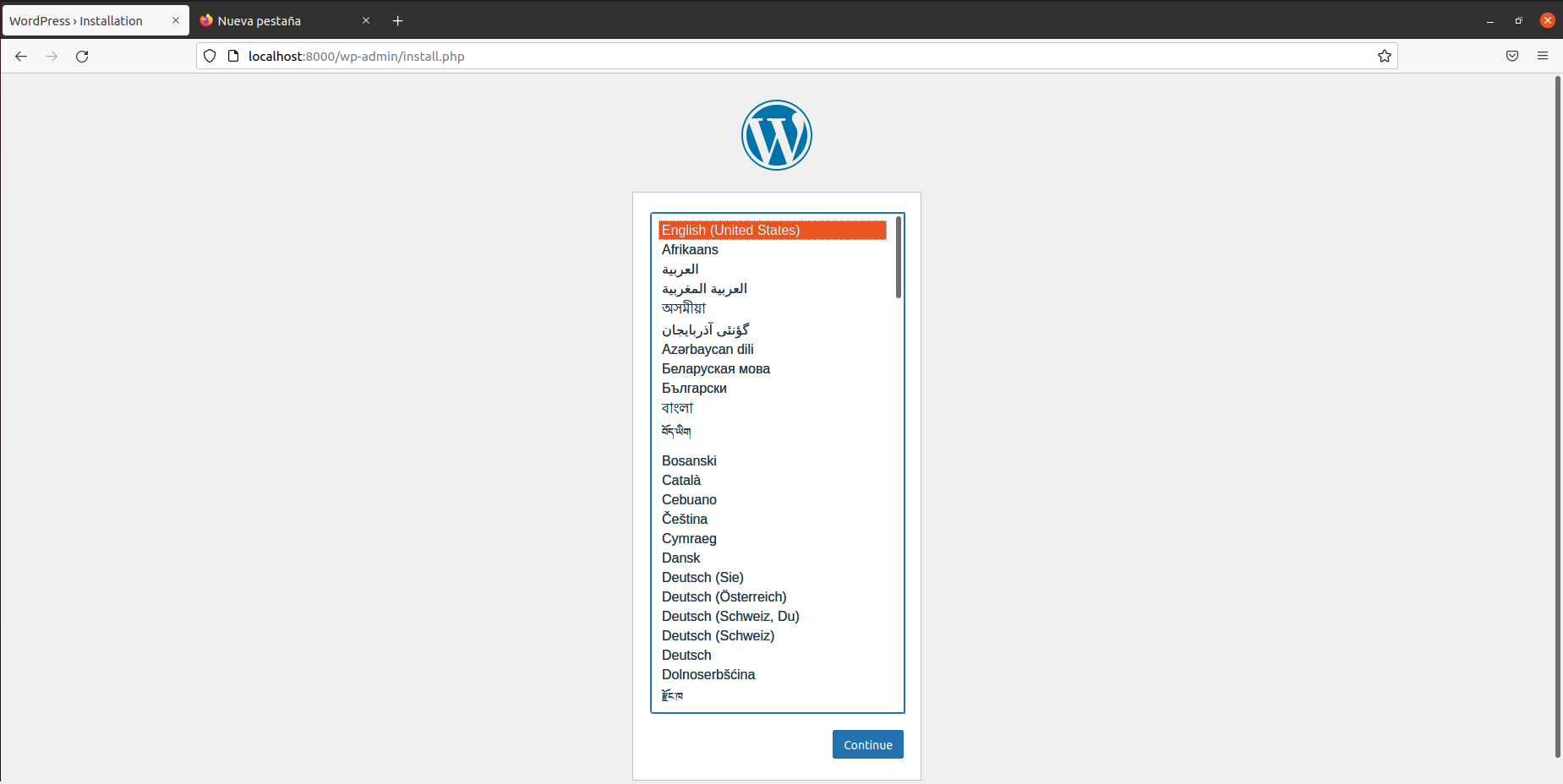




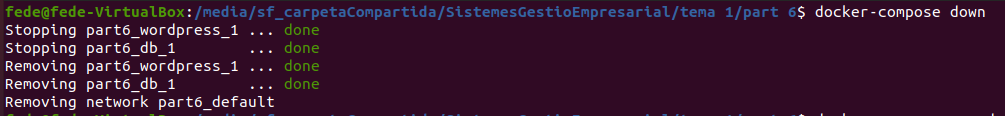
Per a llançar l’aplicació, hem d'usar el següent comande, que usa -d per a ser executat en segon pla i up per a obtindre les imatges indicades, si no els teníem ja.



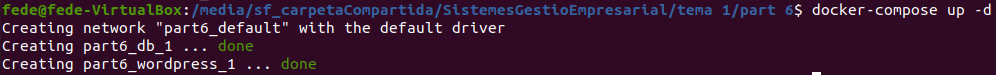
Una vegada executar, podrem accedir a la web a localhost:8000:



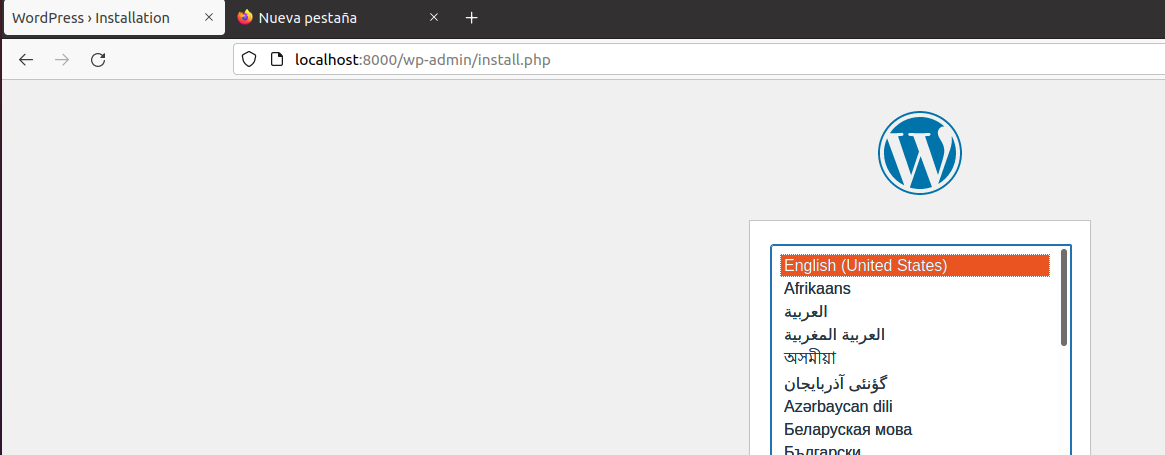
Per a parar el contenidor usarem el comande:



I per a encendre-ho de nou usarem:



Observem que la pàgina segueix disponible:



# Part 07. Cas pràctic 01 - Desenvolupant amb Visual Studio Code en un contenidor.

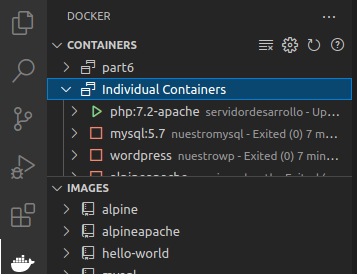
En primer lloc, buscarem l’extensió de Docker a Visual Studio Code i l’instal·larem:



També necessitem la següent extensió:



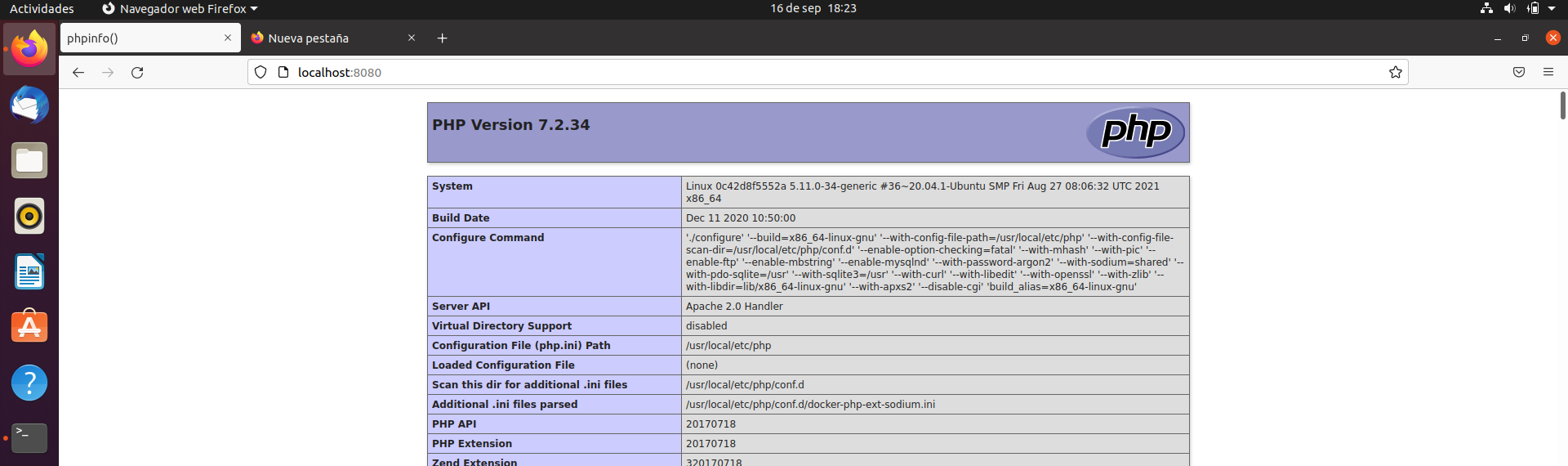
Una vegada fet, podrem veure l’icona de la balena. Buscarem el contenidor que hem creat:



Amb clic dret, serà possible accedir als fitxers de la màquina polsant Attach Visual Studio Code:



Creem un fitxer amb phpInfo a var/www i quan tornem a accedir a l’url http://localhost:8080/ ja veuriem el resultat del phpinfo en vegada de l'error d’abans:



# Part 07. Cas pràctic 02 - Gestionant Docker amb Portainer CE

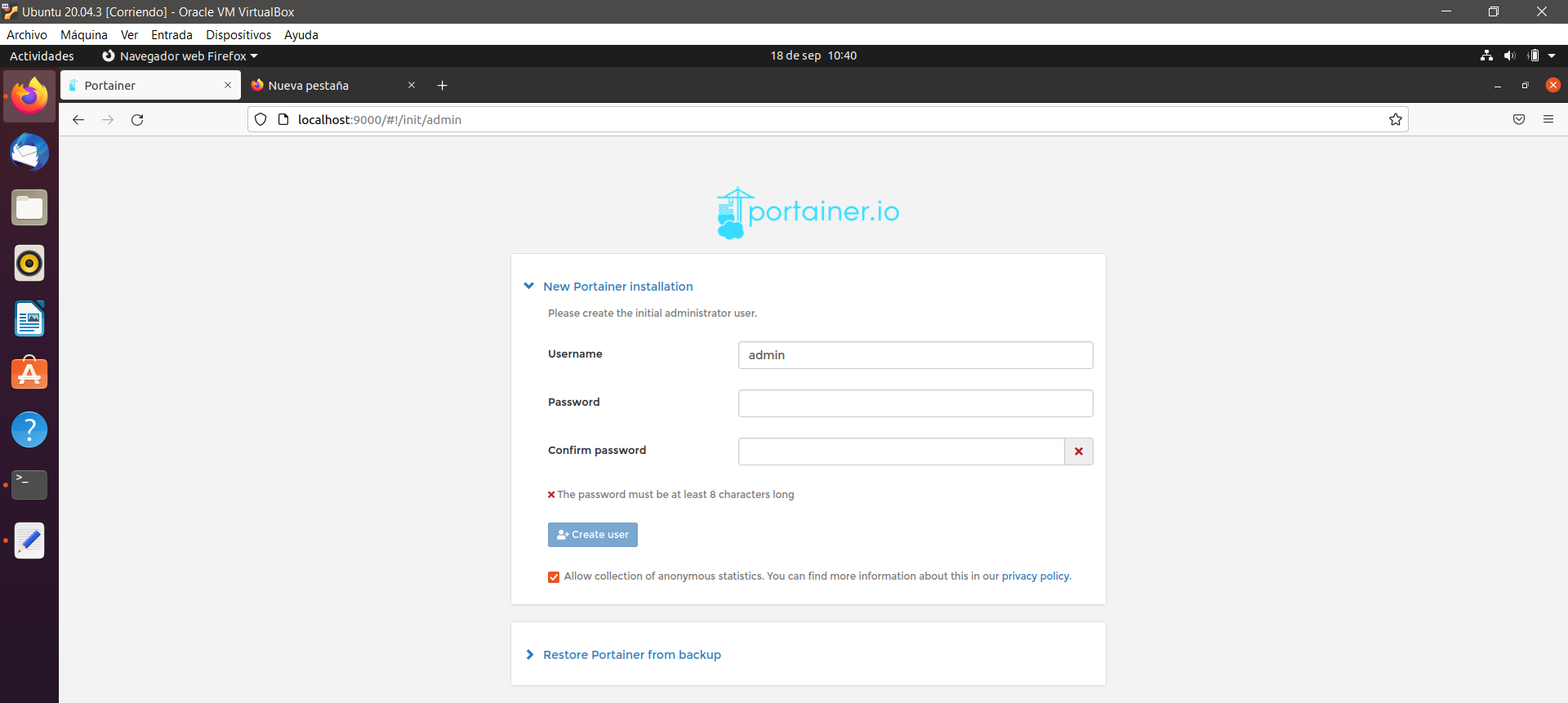
El primer pas que hem fet, és la creació d’un volum. On emmagatzemarem la informació del contenidor:



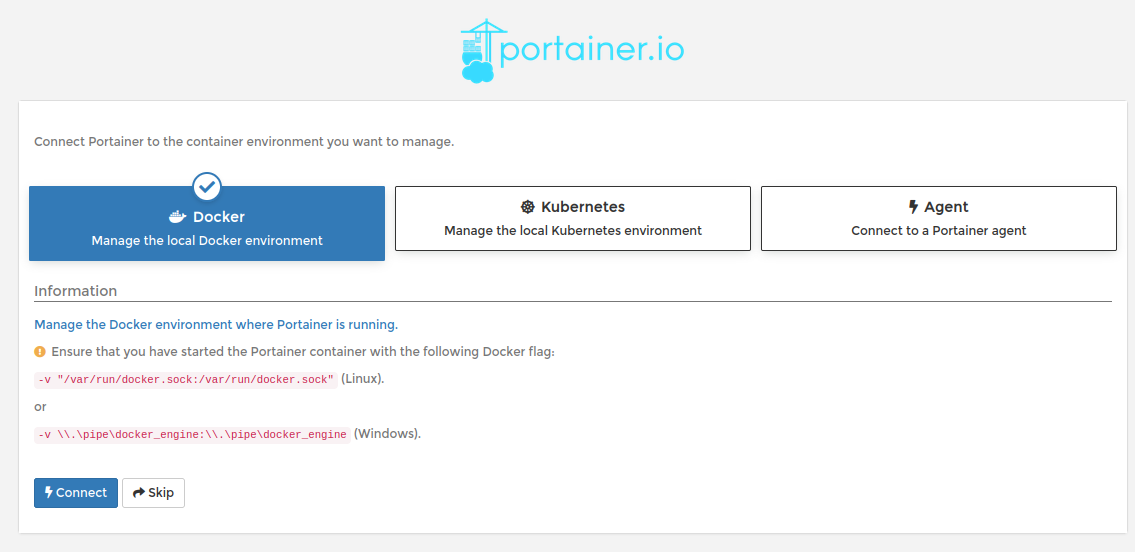
Una vegada creat, llançarem el contenidor:

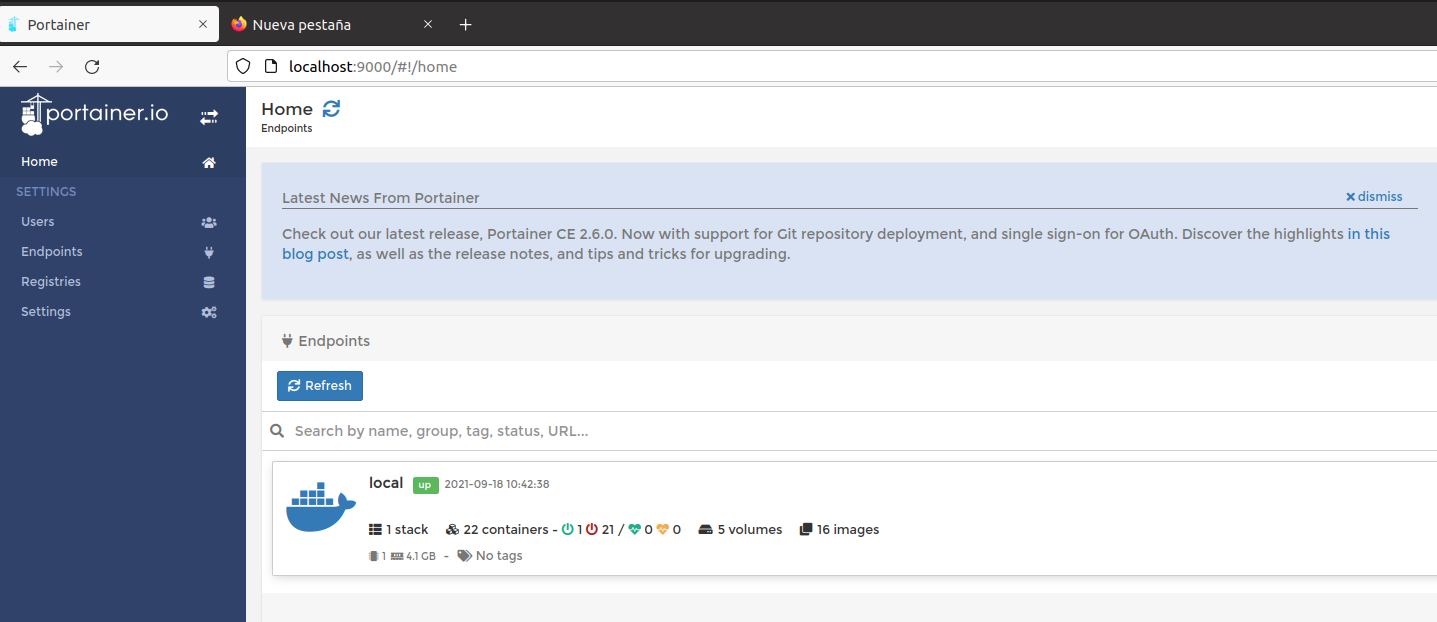


A localhost:9000 podríem veure l’interfície Portainer:

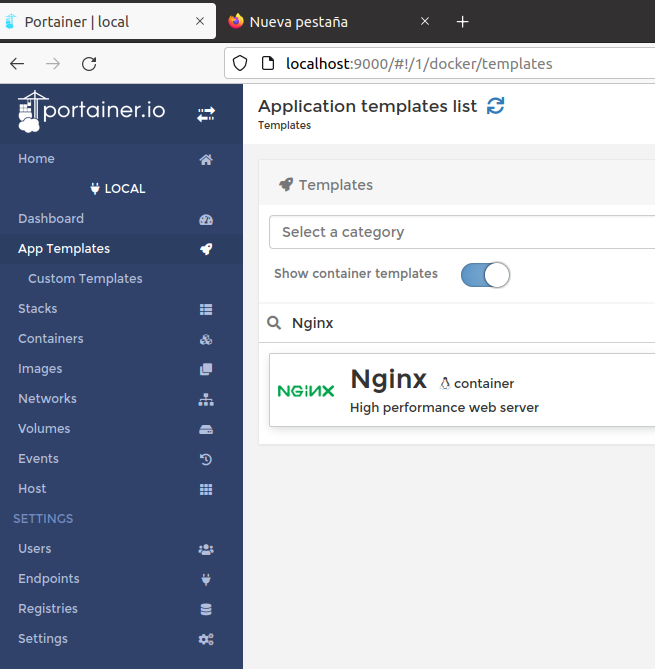


En la següent pantalla indicarem que usarem Docker:

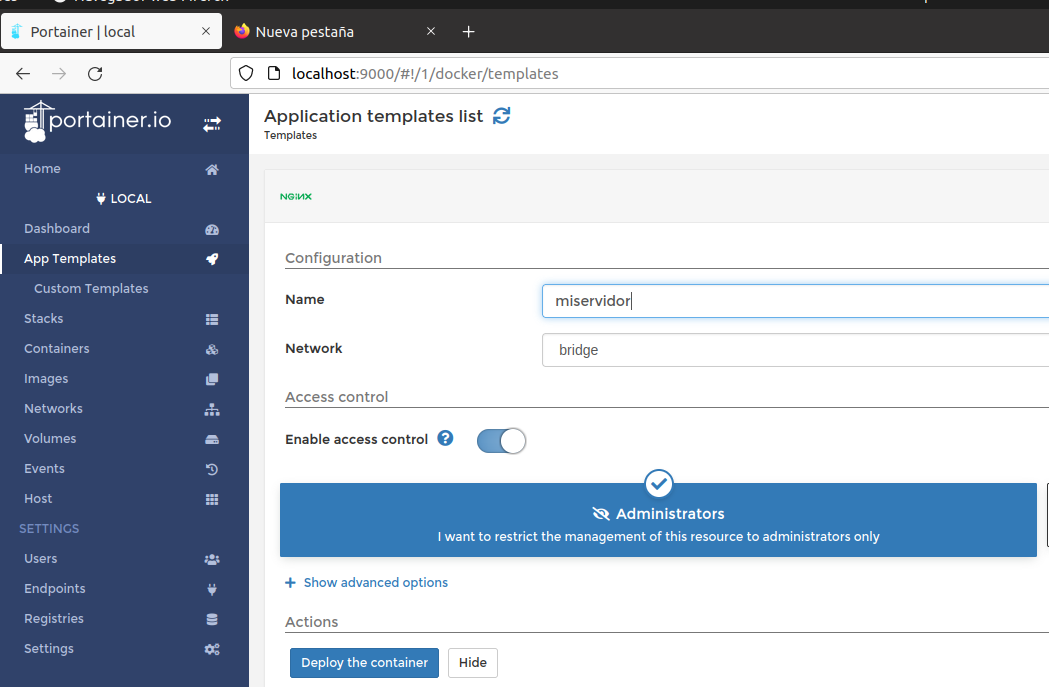




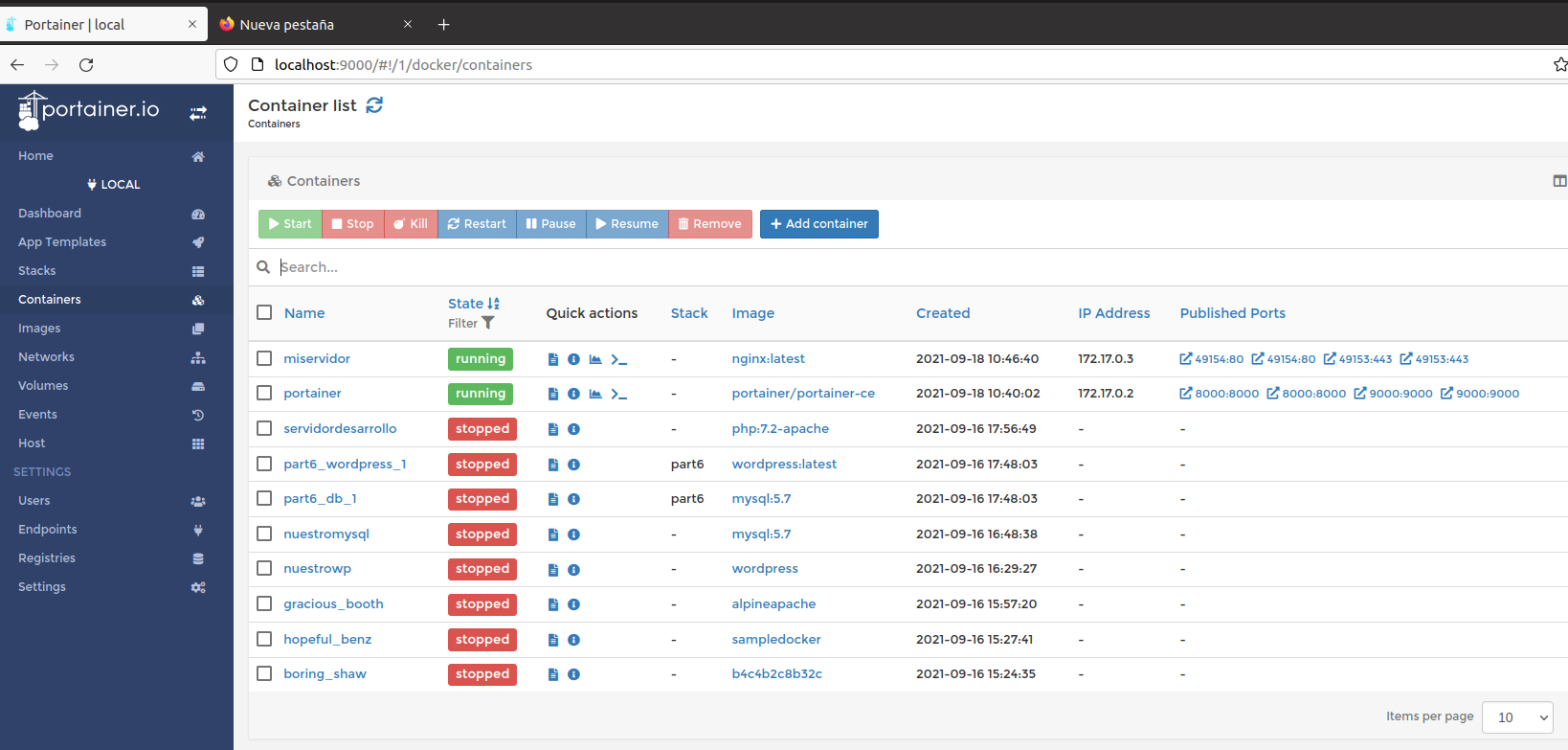
Accedim i busquem nginx en App Templates:

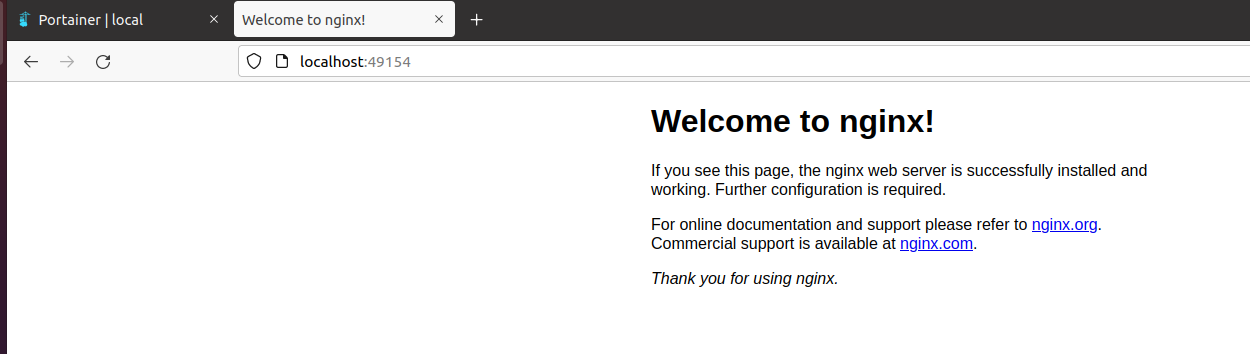


Omplirem les dades i polsarem sobre Deploy Container:



Una vegada desplegat, ens apareixia aquesta pantalla i a http://localhost:49154/ encontraríem el servidor Nginx:





# Activitat 03: Resum de Kubernetes

Llegiu el comic de “Kubernetes” que está disponible en la següent url: <https://cloud.google.com/kubernetes-engine/kubernetes-comic?hl=es-419>

Inclou al document de l’activitat anterior un apartat resumint en un màxim de 8 línies que es Kubernetes i perquè raons pot ser útil.

Kubernetes és una plataforma on disposarem de diferents avantatges al desplegar contenidors. Una característica molt important és l'escalabilitat, pel que podríem fer els servicis créixer a mesura que el necessitem. La portabilitat també es millora en gran mesura, ja que és compatible amb la gran majoria de sistemes. Tots els contenidors són distribuïts d’una manera eficient i també Kubernetes és encarrega de mantindre els servicis corrent contínuament, pel que estalvia als desenrotlladors molt de temps. Finalment, no hem d’oblidar que Kubernetes també ens aporta abstracció, pel que l'ús serà més senzill que amb altres opcions.