INFORME PRACTICA 2

GESTIÓ D’INFRAESTRUCTURES PER AL PROCESSAMENT DE DADES - CONCEPTES DE XARXA I SERVEIS

Informe realitzat per: Iker Sánchez Catena i Grau Cladera Sensat

[1. Configuració de l’entorn 3](#_Toc500003607)

[1.1 Creació de model-train 3](#_Toc1691396298)

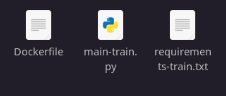
La pràctica s’ha dut a terme a l’usuari gixpd-ged-17 i la MV utilitzada té com a ID “Ubuntu - Practica 2”.

# 1. Configuració de l’entorn

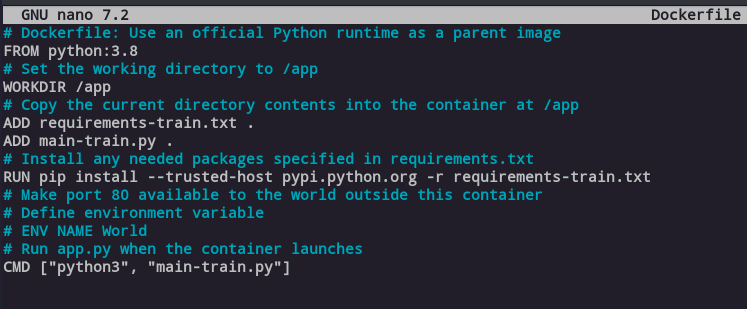
En primer lloc, hem realitzar la tasca 1 seguint les indicacions de les pàgines web indicades a la documentació, per instal·lar *Docker*, *Kubectl* i *Minikube*.

#### 1.1 Creació de model-train

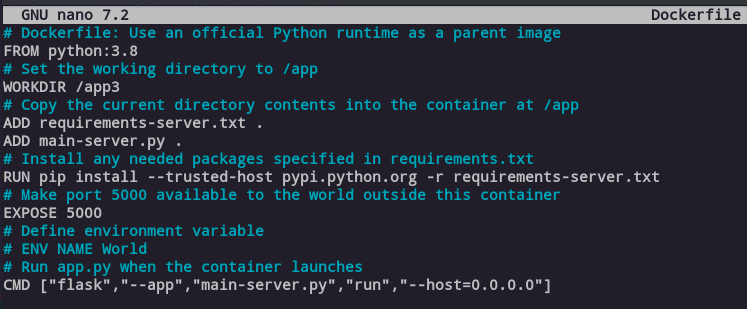
Per crear la primera imatge hem creat una carpeta anomenada “app” i dins d’aquesta un arxiu “Dockerfile”. També hem afegit els arxius “main-train.py” i “requeriments-train.txt” del campus virtual.



Un cop fet això, hem omplert l’arxiu “Dockerfile”:

Docker file train

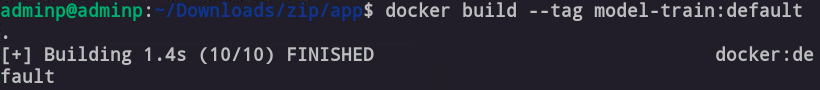
A la primera línia de codi especifiquem que utilitzarem python:3.8. Seguidament, especifiquem el directori de treball dins del contenidor, aquest serà “/app”, i hi afegim el contingut del directori actual a dins d’aquest directori, és a dir, els dos arxius esmentats anteriorment. A continuació, instal·lem les dependències necessàries que es troben a l’arxiu “requirements-train.txt”. Amb l’opció “--trusted-host" permetem que “pypi.python.org” sigui una font confiable per evitar errors. Finalment, l’última línia és la comanda que farà que s’executi el script “main-train.py” quan el contenidor s'iniciï.

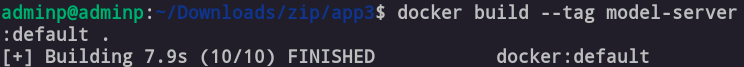
Docker File server

En aquest “Dockerfile” només tenim dues diferències respecte al “model-train:default”. La primera és *EXPOSE 5000*, aquesta línia farà que el contenidor escolti a través del port 5000. La segona és la comanda que executarà el contenidor quan s'iniciï. Aquesta comanda té els següents paràmetres:

* “flask”: inicia el servidor *Flask*. *Flask* permet construir aplicacions web i APIs a través de Python
* “--app”: indica a *Flask* on es troba l’arxiu que conté l’aplicació *Flask*.
* “main-server.py”: és el nom de l’arxiu on es troba l'aplicació *Flask*.
* “run”: inicia el servidor de desenvolupament. Ens permetrà accedir a l’aplicació web a través del port 5000.
* “--host=0.0.0.0”: paràmetre que indica a *Flask* quina adreça IP ha d’escoltar. En el nostre cas especifiquem que ha d’escoltar totes les adreces IP.

Un cop completats els passos anteriors, podem crear la imatge Docker.





Explicació de la comanda:

* *Docker build*: s’utilitza per crear la imatge Docker.
* *--tag=model-train:default*: Assignem el nom de la imatge com “model-train” i especifiquem la versió “default”.
* *.* : Indica que la construcció de la imatge es farà en base al directori actual.

Després, comprovem que la imatge s’ha creat correctament amb la comanda *docker image ls*, que llista les imatges Docker que es troben en el sistema. Com podem veure la imatge s’ha creat correctament.

#### 1.2 Creació de la imatge model-server:default

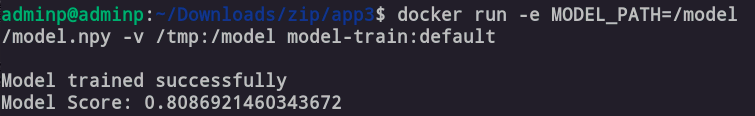
Podríem dir que Flask és un intermediari entre la API i l’usuari. En aquest cas, Flask accepta sol·licituds HTTP i retorna respostes segons els valors que li inclouen com a paràmetres (les accions que ha de dur a terme s’indiquen a main-server.py). En el nostre cas: /, /model i minutes. Flask realitza les crides corresponents a l’API i les retorna en format web, json o similar.

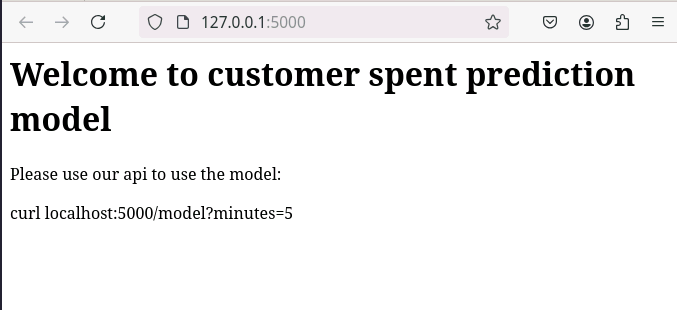
Per realitzar aquesta tasca cal que:

* Fem docker run de train:

Explicar paràmetres, això de que el model\_path ha de ser igual perquè s’han de comunicar entre ells.

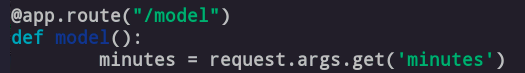
5000:5000 port extern port del sistema i port de la imatge

La línia subratllada correspon a la petició per veure la pàgina de la direcció /model.

Sortida en format HTTP.

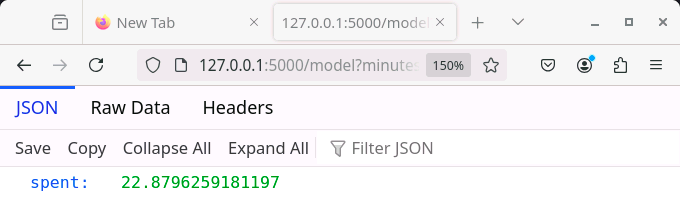
Ens donava error quan intentàvem accedir a /model.

Amb aquest error hem entès que ens faltava incloure el valor del paràmetre minutes.



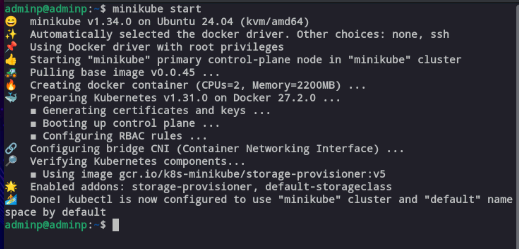
Comprovant el codi, hem vist la comanda de minutes, que espera un valor.

La comanda a inserir al navegador és la següent:  
127.0.0.1:5000/model?minutes=5. Tal i com indica a la pàgina de /, però no ens havíem adonat.

Resultat en format JSON.

# 2. Desplegar l’aplicació a Kubernetes

En primer lloc, és necessari iniciar el servei minikube amb *minikube start.*

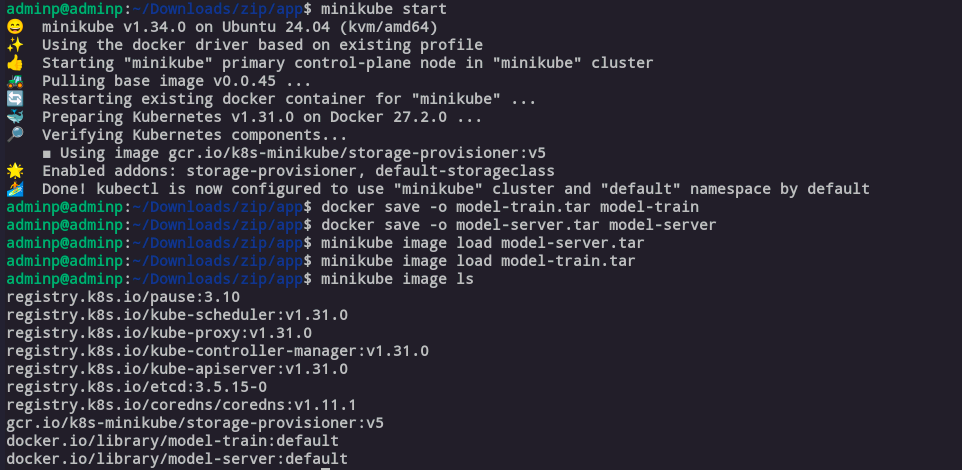


Seguidament reduïm la mida de les imatges,donat que son grans i ens pot donar error de timeout a l’hora de carregar-les.

Fem servir la següent comanda: docker save –o IMAGE.tar IMAGE



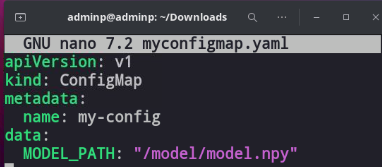
Ara carreguem les imatges amb *minikube image load IMAGE.tar* i amb la comanda *minikube image ls* comprovem que s’han carregat correctament:



Config Map

L’arxiu configmap serveix per guardar els valors de certes variables no confidencials. D’aquesta forma, es millora l’escalabilitat de la nostra aplicació, evitant haver d’entrar en el codi per modificar valors.

En el nostre cas, el contingut de configmap és el següent:



* ApiVersion: es refereix a la versió api de kubernetes
* Kind: especifiquem que es tracta d’un configmap
* Metadata: serveix per donar informació sobre l’arxiu. En aquest cas, indiquem que el seu nom és my-config
* Data: aquí incloem els valors de les variables. En el nostre cas, especifiquem la ruta de MODEL\_PATH.

Amb la seguent comanda kubectl apply -f myconfigmap.yaml apliquem la configuració de configmap al nostre clúster de kubernetes.

Vam tenir problemes al’hora d’iniciar kubernetes ja que ens donava un error de connexió. Seguint les recomanacions del professor, vam fer un reboot a la màquina, però l’error continuava. Ho vam ignorar, perquè ens deixava pujar les imatges.tar. Tanmateix, a l’hora de pujar l’arxiu myconfig, ens deia que no era possible a causa de l’error anterior.

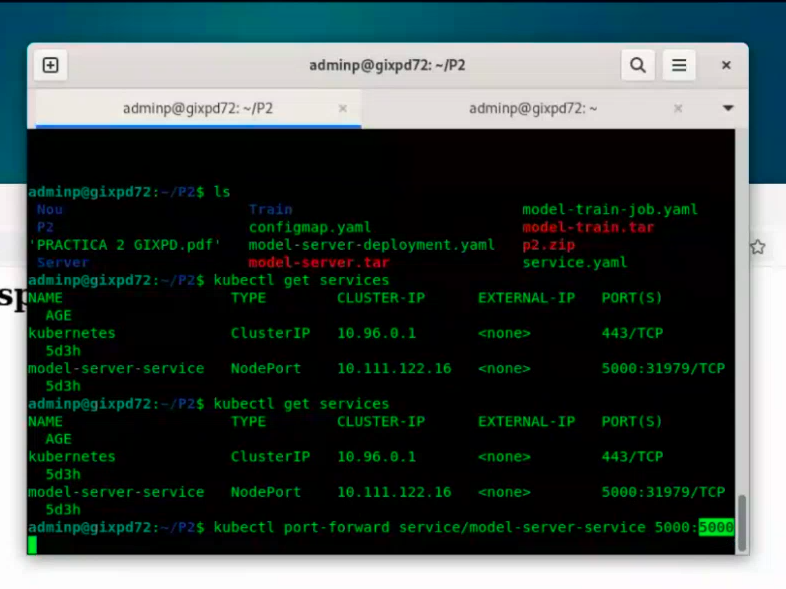
Després d’investigar, vam eliminar la imatge de kubernetes a Docker, per forçar-li a crear-la de nou amb una nova configuració, però res canviava. Finalment, vam eliminar tot el contingut de minikube:

minikube delete

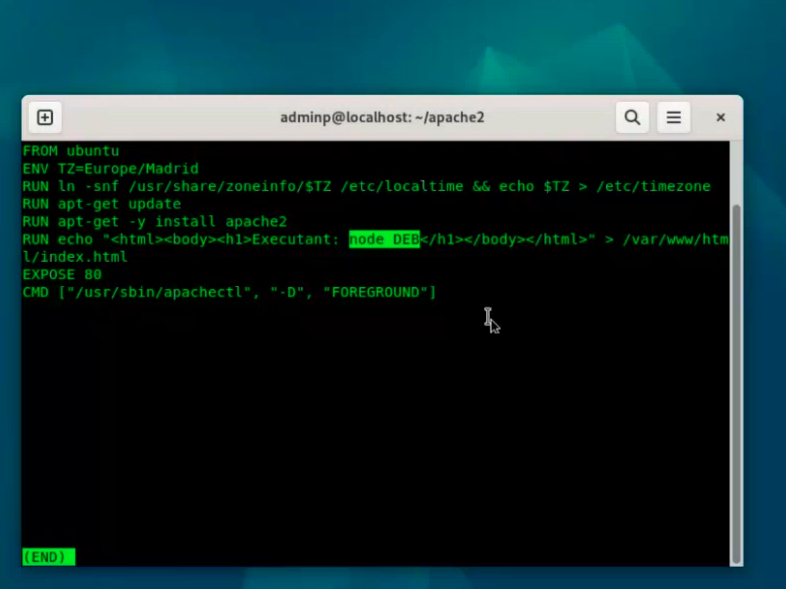
minikube start

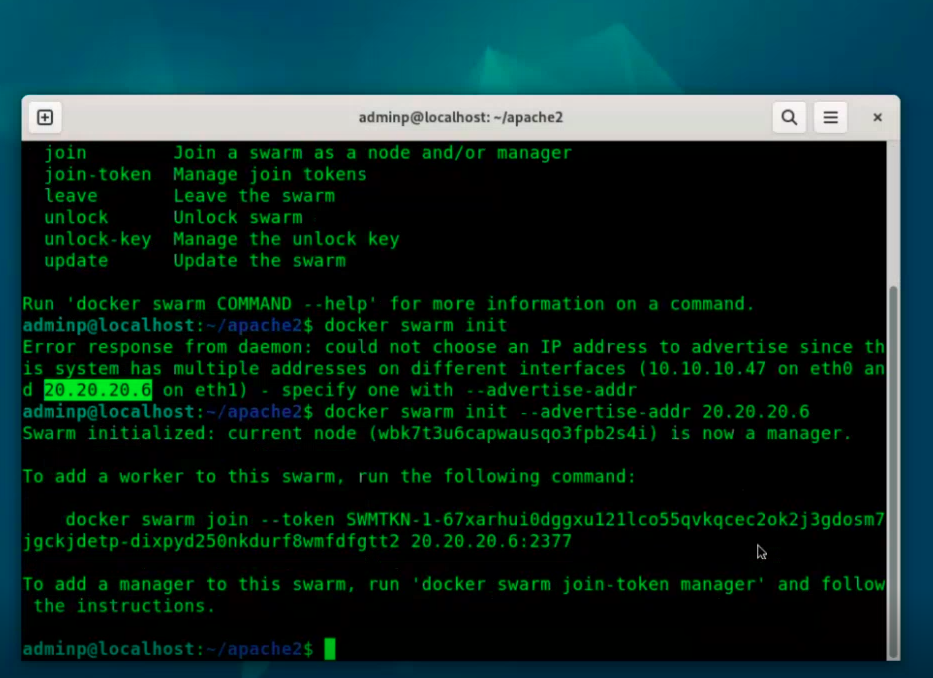
Tot arreglat.

Per canviar el port i IP

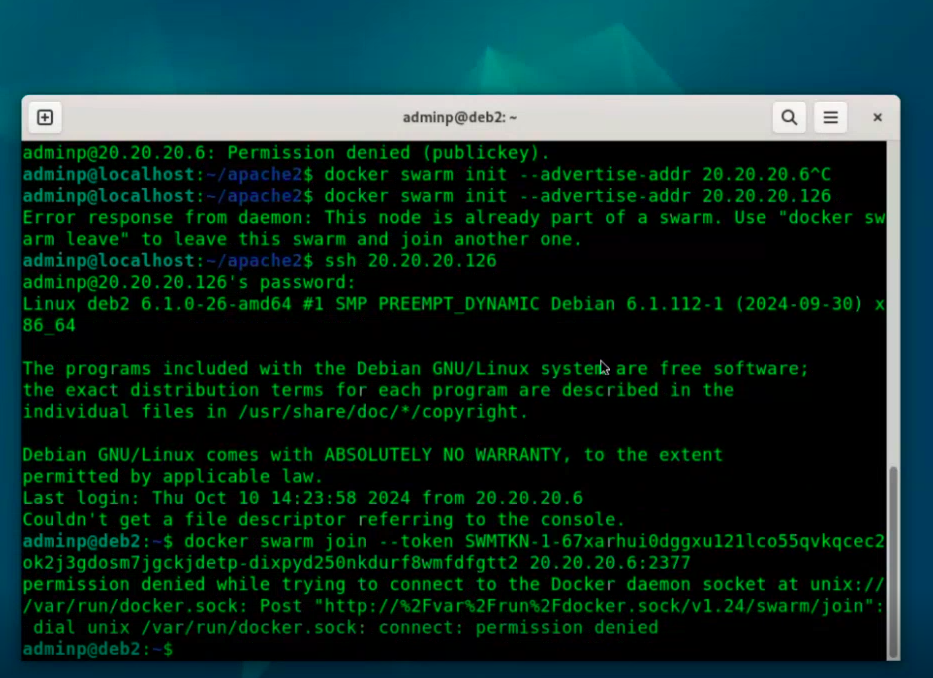


Contenidor amb apache2

Canvia perquè no apareixi al localhost crec

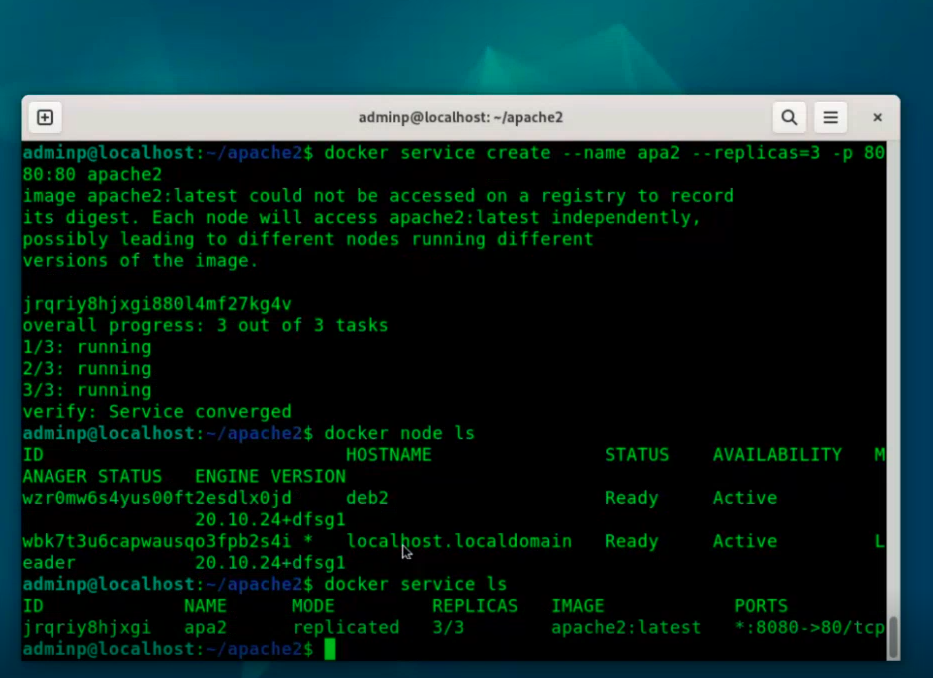


Ssh a l’altre maquina

Copies el swarm

Aconsegueixes que aquesta maquina formi part del swarm

Docker node ls -> \* indica que es el master

Per fer repliques:  


Enrecordar-se de l’entorn gràfic de minikube, link al fer el init