CAP-GIA (FIB) Laboratory Assignment
Lab 2: Contenidors i Supercomputadors

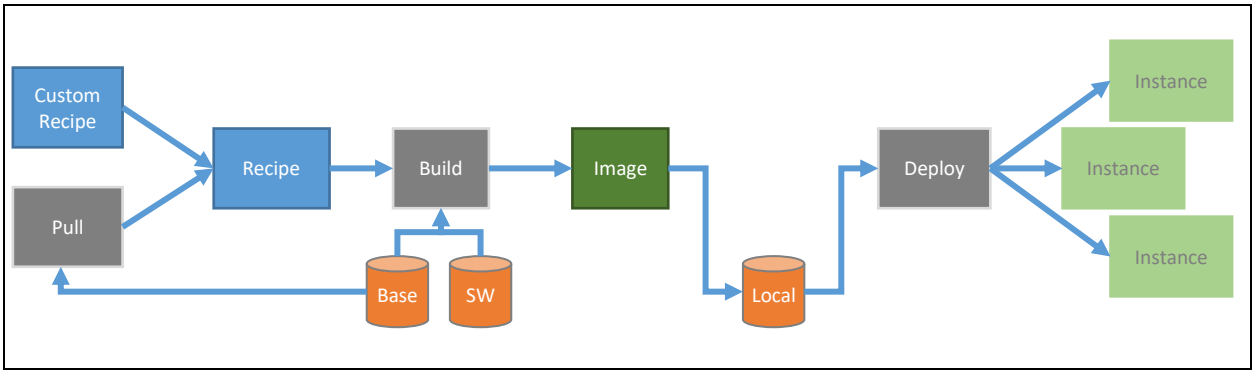
Jordi Torres and Josep Lluís Berral

Fall 2024

Contenidors i Supercomputadors

El fet de poder encapsular serveis en "contenidors" ens permet crear entorns d'execució i serveis llestos per a ser desplegats en gairebé qualsevol entorn, reduint al màxim les dependències i software a instal·lar, ja que els components i serveis queden "auto-continguts" dins el contenidor. Aquesta sessió estarà centrada en 1) el desplegament i experimentació amb contenidors, 2) la creació de contenidors que incloguin aplicacions "custom", i 3) la execució de contenidors a supercomputadors (i.e. MareNostrum).

Singularity és un hypervisor encarregat de construir imatges de contenidor i d'executar instàncies d'aquestes imatges, en espai d'usuari i evitant escalar permisos (com fa Docker). Pel que és idoni per a entorns de computació controlats, com és el Supercomputador MareNostrum. En aquest laboratori prepararem un entorn amb contenidors que incloguin PyTorch, capaços d'executar aplicacions de AI i Deep Learning. En aquest laboratori treballem amb **Ubuntu 19.04 LTS (64bits) + Singularity**, on veurem com crear i instanciar imatges, i veurem com exportar-ho a entorns de supercomputació per tal d'executar-ho a MareNostrum.



Objectius de laboratori

1. Crear i instanciar contenidors en un sistema remot (e.g., usant Singularity)
2. Crear un contenidor que executi una aplicació amb paràmetres (e.g., python3)
3. Crear un contenidor complex que executi eines de Machine Learning
4. Exportar un contenidor a un altre sistema (MareNostrum) i executar-ho

---

Nota important: Compte amb fer "copy & paste", ja que hi ha símbols com comes o guions que es poden copiar malament d'aquest document a consola. Si no us va una comanda, escriviu-la directament!

Índex de continguts

Instruccions de la pràctica

Aquesta sessió de laboratori la farem a una màquina virtual amb Singularity al clúster CROMAI (primera part), i després mourem els contenidors al supercomputador MareNostrum (segona part). A l'inici de la sessió caldrà que creeu una màquina virtual per a treballar amb Singularity i Ubuntu.

Les màquines preparades (Clúster CROMAI-CAP) disposen de Debian 12, i seran compartides per tots els estudiants. Això vol dir que haurem de ser conscients que compartim recursos limitats. Aquestes màquines tenen instal·lat VirtualBox (hypervisor) i Vagrant (gestió d'imatges de VMs).

Grups de Treball

Per a aquesta pràctica ens distribuïm en parelles (**grups de 2 persones**). Cada parella tindrà un usuari i una màquina assignada a *cromai-cap*. Cada parella farà una sola entrega; una recomanació és que una persona del grup faci el rol d'executor, i l'altre membre del grup faci de secretari, i anar anotant les respostes del qüestionari que cal anar responent.

Qüestionari de Laboratori

Durant la pràctica trobarem preguntes que cal anar contestant a mesura que avanceu per la pràctica. Aquestes formen el **qüestionari de laboratori** que ens servirà de guia durant el laboratori.

Disposareu de la sessió de laboratori per a fer la pràctica, i d'una setmana per a **entregar el qüestionari pel Racó**, on veureu una pràctica oberta per a "Contenidors". Les respostes les entregareu usant el document plantilla de la pràctica, indicant les respostes sota les corresponents preguntes. Assegureu-vos de justificar bé les respostes, i suposeu al voltant d'una línia o dos per resposta, tampoc més. No enganxeu "screenshots". Eviteu entregar al darrer moment, per evitar possibles problemes tècnics, i per a que us pugueu assegurar que heu pujat la pràctica correctament.

Temps de Dedicació

Aquesta pràctica ha estat dissenyada per a que la pugueu realitzar en **aproximadament dues hores**. Hem tingut en compte el que es triga en descarregar el software i les imatges de VM i Contenidors.

Entrega i Avaluació

L'avaluació de la sessió presencial tindrà en compte l'assistència i participació a classe, així com el qüestionari entregat.

Nota Qüestionari → Suma de les puntuacions per cada pregunta.

0. Preparació de l'entorn de la Màquina Virtual

Ja hauríeu d'haver instal·lat i usat VirtualBox i Vagrant a la sessió anterior. Usarem aquestes tecnologies per a descarregar i preparar l'entorn per a aquesta sessió. Primer, entrarem a la màquina *cromai-cap* assignada, amb el *user* i *password* donats:

```
ssh <usuariYYY>@<màquina assignada>
```

Ara teniu accés a l'hypervisor i a un gestor d'imatges preparats per fer servir.

Preparant VM Ubuntu + Singularity

Primer, descarreguem una VM amb Ubuntu i Singularity pre-instal·lats i configurats. Les següents comandes funcionen tant per Linux, MacOs com Windows, des de la consola de comandes:

```
mkdir vm_cap3
cd vm_cap3
vagrant init sylabs/singularity-3.3-ubuntu-bionic64
```

Ara, hem de modificar el descriptor de la VM, per donar-li més RAM, obrir ports per tenir-los disponibles i accedir, i donar-li un nom a la VM. Obrim el fitxer "Vagrantfile", i afegim les següents línies **entre els tags de "config"**:

$HOME/vm_cap3/Vagrantfile

```
  config.vm.provider "virtualbox" do |vb|
        vb.memory = "1024"
        vb.name = "container"
  end
```

Ara, la nostra VM està llesta per ser descarregada (el primer cop) i per iniciar-se:

```
vagrant up
```

Entrant a la VM

Ara podem entrar a la VM usant SSH directament:

```
vagrant ssh
```

Ara entrarem a la màquina virtual, llesta per a fer la sessió.

Recordeu: Per sortir només hem de fer "exit", i per aturar la VM podem fer "vagrant halt" des de fora. Per iniciar-la de nou, només ens cal fer "vagrant up". Si volem esborra la VM, podem fer "vagrant destroy", i fent "vagrant up" després tornarem a descarregar la imatge nova.

A partir d'aquí, ho farem tot **DINS LA VM**.

1. Singularity i Contenidors

La maquina virtual que hem descarregat incorpora “Singularity 3.3” ja instal·lat. Pel que dins podrem preparar i construir imatges de contenidors Singularity.

El nostre objectiu és preparar un contenidor que provist de python, pytorch i scikit-learn, per tal de:

1) Poder executar processos de Deep Learning
2) Enviar-ho a executar a MareNostrum

Creant un Contenidor amb Ubuntu

Un contenidor es construeix a partir d’una “recepta”, que pot contenir diferents apartats **(“pot contenir” significa “no cal que els tingui tots”**) indicant què cal instal·lar dins el contenidor, quines variables calen a l’entorn d’execució de la instància, quins fitxers cal posar o quins cal compartir des de la màquina hoste.

Primer de tot, necessitem un “singularity file” (la recepta) on podrem provar que Singularity funciona, i que disposem de tot el que cal per crear contenidors dins la nostra VM. Aquí veiem la plantilla principal de com és un fitxer de Singularity:

```
Bootstrap: docker                # Sing. pot agafar plantilles Docker
From: ubuntu:bionic-20190515     # Ubuntu Bionic

%labels
  Maintainer <Aquí poseu el vostre nom>
  Version <Aquí poseu la versió que li doneu al vostre contenidor>

%help
  # Secció no obligatòria
  # Aquesta secció inclou els textos que sortiran al fer "help" sobre
    el contenidor. Afegiu els textos que us facin falta.
  "Missatge d’ajut"

%post
  # Aquesta secció s’executa al final de la construcció de la imatge
    base, i serveix per instal·lar tots els programes que necessitem
    dins la nostra imatge de contenidor. Només s’executa al fer el
    "build" de la imatge.

  # Per exemple, instal·lar "htop" i netejar al acabar el build:
  export DEBIAN_FRONTEND=noninteractive

  apt-get update --fix-missing
  apt-get install -yq htop

  <Aquí poseu les comandes que facin falta>

  apt-get clean
  apt-get autoclean
  rm -rf /var/lib/apt/lists/*

%startscript
```

```
  # Secció no obligatòria
  # Aquesta secció s'executarà cada cop que s'inicii la instància del
    Contenidor. Poseu les comandes que necessiteu.
  echo "Hola! Estàs al contenidor"

%environment
  # Secció no obligatòria
  # Aquesta secció s'executarà a l'inici de la instància, i serveix
    per indicar totes les variables d'entorn (exports) que siguin
    necessaris dins el vostre contenidor. Aquí poseu les variables
    d'entorn que necessiteu.
  export EXEMPLE='hola'

%apprun <Nom de la Aplicació>
  # Secció no obligatòria
  # Es poden crear tantes "aplicacions" com es vulguin. Cada aplicació
    serveix per a tenir diferents funcionalitats al contenidor. P.e.
    si instal·lem python3 (a %post) i creem una aplicació (%apprun)
    anomenada "python3", que faci exec de "python3", podem crear una
    instància del contenidor que ens obri una sessió de python. Si li
    passem paràmetres (afegint "$@" com a paràmetre per passar
    paràmetres de fora), podem crear una instància de python que
    executi, p.e., un fitxer ".py".
  exec <comanda> <paràmetres>

%setup
  # Secció no obligatòria
  # Aquesta secció s'executa després de tenir la imatge base, abans
    d'escriure la nostra imatge, i abans de la secció %post durant la
    construcció de la imatge. Ens pot servir, p.e., per a crear
    directoris dins la imatge de contenidor.

  # Per exemple, crea el directori "/files" dins la imatge
  mkdir $SINGULARITY_ROOTFS/files

%files
  # Secció no obligatòria
  # Aquesta secció s'executa després de la secció %setup durant la
    construcció de la imatge. Serveix per a copiar fitxers i
    directoris de la màquina hoste a la imatge de contenidor. Els
    fitxers queden dins la imatge, i son persistents.

  # Per exemple, copia el fitxer "hello.txt" (a la màquina hoste) al
    directori "/files" creat durant %setup dins la imatge.
  ./hello.txt /files/
```

Quan volem crear un contenidor, creem un fitxer de text "xxxxxx.singularity", on posem la plantilla i la editem incloent els programes a instal·lar, els fitxers a copiar, variables d'entorn a definir, etc. El següent pas és construït (fer "build") de la imatge. Per exemple (SIF és la imatge de contenidor que volem crear):

```
sudo singularity build <nom_contenidor>.sif <recepta>.singularity
```

Un cop està creada la imatge, podem crear una instancia (o tantes com vulguem) fent servir comandes com ara "exec", en que executem una comanda dins la instancia del contenidor.

```
# Crea una instància a partir de la imatge del contenidor, i executa
  una comanda dins la instància. Quan acabi el programa en execució,
  el contenidor finalitzarà.
singularity exec <nom_contenidor>.sif <comanda>
```

EXERCICI 1: Ara crearem una imatge senzilla de contenidor Ubuntu (com l'indicat a la plantilla). Dins el nostre directori "home" de la VM crearem el fitxer "contenidor_01.singularity" (podeu usar *nano* o *vim*). A partir de la plantilla, no cal instal·lar cap programa addicional o variable d'entorn, ni aplicació específica. Si voleu provar, aprofiteu per posar a la llista de paquets elements que vulgueu afegir a la imatge de contenidor, i que vulgueu provar. Ara, construïu la imatge ("build"), tal que tinguem una imatge anomenada "container_01.sif", i que puguem instanciar i executar la comanda "sh" (una consola SHELL). Si ha funcionat bé la construcció del contenidor, hem de tenir davant un terminal SHELL, amb el nom de "Singularity". Un cop creada i provada la imatge de Singularity:

a) Quins fitxers podem veure des de dins la instància, i per què? Explica on s'executarà tot el que cridem dins aquesta instància.
b) Sortim de la sessió de SHELL del contenidor ("exit"). On està ara mateix corrent la instància, i com ho sabem?

Instanciant un contenidor com a servei

Un contenidor pot ser instanciat com a "servei". Això vol dir que engegarem la instància, i en comptes d'obrir al moment una sessió interactiva, la deixarem corrent de fons, i on ens podem connectar després. Per a iniciar i gestionar instàncies de contenidor en execució, podem usar les següents comandes:

```
# Llancem la instància
singularity instance start <nom_contenidor>.sif <nom_de_la_instància>

# Comprovem la llista d'instàncies en marxa
singularity instance list

# Connectem-nos a la instància via terminal
singularity shell instance://<nom_de_la_instància>

# Finalitzar una instància en marxa
singularity instance stop <nom_de_la_instància>
```

Al iniciar una instància com a "servei", li podem donar un nom que vulguem per a identificar-la. D'aquesta forma podem identificar-la a l'hora de demanar-li coses (obrir un SHELL dins, executar coses, o aturar-la).

EXERCICI 2: Iniciem una instància del nostre "contenidor_01" com a "servei", donant-li el nom que vulguem. Al acabar de respondre les preguntes, matem la instància, tal que deixi d'executar-se, i ens assegurem de que ja no està corrent.

c) Quines diferències veiem entre aquesta instància i la que hem llençat abans?
d) Sortim de la sessió SHELL del contenidor. On està ara mateix corrent la instància, i com ho sabem?

Afegint aplicacions a una imatge

El nou objectiu és que la nostra imatge tingui Python3 instal·lat, per poder llançar scripts de Python com ara PyTorch, usant l'entorn instal·lat dins el contenidor.

EXERCICI 3: Creeu un nou descriptor d'imatge de Singularity anomenat "contenidor_02.singularity", que podem inicialment copiar de "contenidor_01". Les comandes per instal·lar de python y pytorch son les següents. Afegiu-les on calgui a la nova recepta.

```
  # Install Python3
  apt-get install -y python3 python3-pip python3-matplotlib \
    python3-numpy python3-scipy

  # Upgrade Pip and install Torch + SciKit-Learn
  python3 -m pip install --upgrade pip
  python3 -m pip install torch torchvision scikit-learn
  python3 -m pip cache purge
```

A més a més, li posarem una "aplicació" anomenada "python3", per tal de que puguem iniciar de forma automàtica python3 i ens accepti paràmetres (scripts i datasets). Per a això, la directiva que ens cal afegir a la recepta dins de la app "python3" serà:

```
  exec python3 "$@"
```

I finalment, per tal de provar-ho, crearem un mini-script de Python (e.g., "prova.py") per a fer la prova:

```
print("Hola, classe de GIA-CAP!")
```

Un cop tingueu llesta la recepta de "container_02", construïu-la i instancieu el contenidor executant la aplicació de "python3" dins la instància, passant com a paràmetre "prova.py". Amb la comanda "run" podeu indicar quina "app" del contenidor voleu, i passar els paràmetres:

```
singularity run --app <app> <nom_contenidor>.sif <paràmetres>

# Alternativament, amb "run" també podem cridar comandes senceres
singularity run <nom_contenidor>.sif <comanda>
```

La instància de "container_02" no serà ni interactiva ni un servei, ja que obrirà Python3, executarà el nostre script passat per paràmetre, i s'acabarà. Al executar-la, esperem veure per pantalla el missatge que hem posat al "print".

e) Un cop construïda, com comprovarem que funciona i que podem executar Python3 dins una instància?

f) Com hem pogut comprovar que funciona? Un cop ha fet el que ha de fer, segueix la instància en marxa?

Nota: Podem cridar "python3" com una aplicació dins el contenidor usant "--app", tant sobre una instància nova com sobre una instància que estigui ja en marxa ("instance://<Nom Instància>"), o ho podem cridat sobre una instància nova de la mateixa forma que hem cridat "sh" abans.

Afegint fitxers dins la Imatge

En ocasions ens interessarà incloure fitxers dins la imatge, per a que aquests estiguin disponibles dins de cada instància.

EXERCICI 4: Com a prova, crearem el fitxer "hello_cap.txt", i modificarem la recepta de "container_02" per a que quan es construeixi la imatge aquest fitxer sigui dins. Crearem el fitxer "hello_cap.txt" (p.e., fent servir la comanda següent), i al fer el "build" crearem el directori "/files" dins la imatge i hi copiarem el fitxer. Afegiu a la recepta les comandes per a crear el directori i copiar els fitxers allà on toqui.

```
echo "Hola, classe de GIA-CAP!" > hello_cap.txt
```

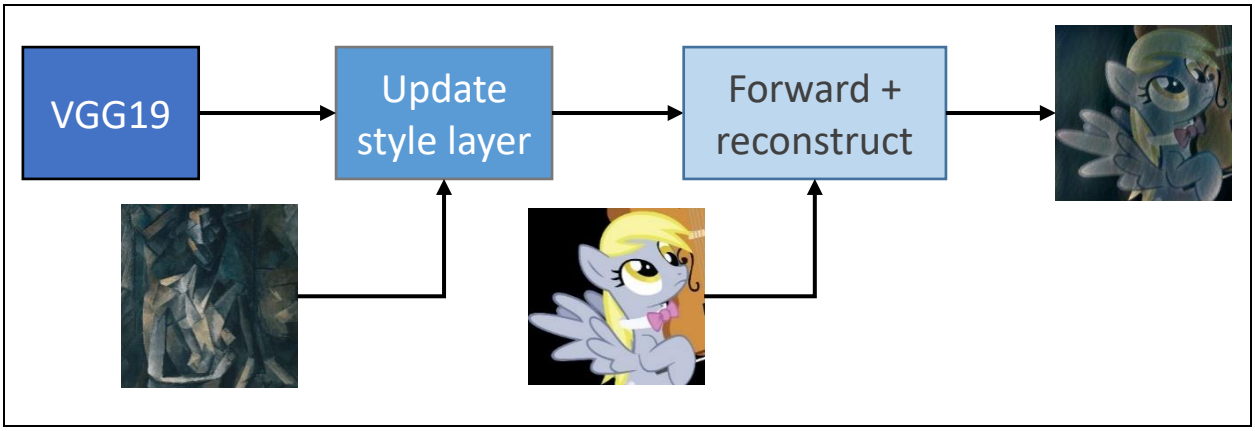
Un cop reconstruïda la imatge, instancieu-la amb un terminal i comproveu que s'ha copiat el fitxer dins del directory "/files". Comproveu també que el directori "/files" NO existeix fora de la instancia.

g) Quina comanda podem fer per assegurar-nos que el fitxer s'ha inserit dins la imatge creada?

NOTA: En cas de que després de diversos "build" Singularity doni problemes d'espai, provar de fer neteja de cache "singularity cache clean --all".

El cas d'ús de Style Transfer – Usant contenidors amb VGG19

La següent tasca serà descarregar-nos un exemple de "Style Transfer" usant Python i PyTorch. El cas d'ús usa la xarxa neuronal VGG-19, dedicada a reconeixement i reconstrucció d'imatges. Si es descarreguen els pesos de la xarxa, i es fa "fine tuning" específicament de la capa 4, la xarxa aprèn l'estil de la imatge de re-entrenament, i reconstrueix qualsevol altra imatge amb l'estil d'aquesta primera imatge.



Dins la VM ens descarregarem els pesos de la VGG19 i un script amb el codi per re-entrenar la xarxa + produir imatges noves a partir d'un model re-entrenat.

```
 wget https://www.dropbox.com/s/tt3a0vjs830n7n8/style-cap.tar.gz?dl=0
-O style-cap.tar.gz
 tar xvzf style-cap.tar.gz
```

Veurem els següents fitxers:

```
 vgg19-dcbb9e9d.pth                  # Matriu de Pesos de la VGG19
 style.py                            # Script de Style Transfer
```

Ara haurem de pujar dues imatges, una d'entrenament i una de test. Busqueu dues imatges que tingueu i que vulgueu fer servir, i seran "image_tr" i "image_ts". Per copiar-les a les màquines de *cromai-cap*, podeu usar la comanda "scp".

```
 scp <fitxer> <usuari>@<màquina>:/home/<usuari>/
```

Després, haurem de retallar les imatges per a que tinguin unes dimensions que la xarxa VGG19 accepti. Podem usar el programa d'edició gràfica que més ens agradi, per tenir imatges de 1000x1000 i sempre en format JPG: Per exemple, podem usar "convert":

```
 convert image_tr.jpg -resize 1000 -crop 1000x1000 imatge_tr_crop.jpg
 convert image_ts.jpg -resize 1000 -crop 1000x1000 imatge_ts_crop.jpg
```

Ara farem una prova en local, per veure que l'script funciona correctament amb les imatges. Proveu de llançar "style.py" a python3, fent servir la opció d'ajuda "./style.py -h" (imprimeix l'ajuda de "style").

EXERCICI 5: Ara modificarem de nou la recepta de "contenidor_02" per a que 1) al construir-la, inclogui dins la imatge la matriu de VGG19 i l'script; i 2) inclogui la aplicació "style", que executi l'script (de dins el contenidor, amb la matriu de dins el contenidor) usant com a paràmetres només la imatge d'entrenament i de test (més híper-paràmetres) . El resultat ens ha de permetre fer:

```
 singularity run --app style container_02.sif -s image_tr.jpg \
   -i image_ts.jpg -o image_result.jpg -w 1000 -e 30
```

h) Podem executar la instància a la nostra màquina local? Justifiqueu el que observeu, i indiqueu quan triga a executar-se una època. (Potser podeu jugar amb el paràmetre "-w" si voleu... indiqueu a la resposta quin valor li heu donat, i quin temps ha trigat)

w	e	Temps
1000	30	
256	30	
1000	5	
256	5	

Nota: És possible que en algun moment us trobeu que... NO HI HA PROU MEMÒRIA! En aquest cas, l'experiment l'haurem de dur a MareNostrum!

2. Contenidors i Supercomputadors

Accedim a MareNostrum

Donat aquest punt, disposem d'una imatge de contenidor que inclou la nostra xarxa neuronal i els scripts i serveis necessaris per a transformar una imatge a partir d'una altra. Com que hem vist que en la nostra màquina és impensable executar-ho amb imatges de bona qualitat i mida, ho voldrem portar al supercomputador MareNostrum.

El primer pas és veure quins fitxers calen pujar al supercomputador des de la nostra màquina. Quan ho feu, useu el node DataTransfer en comptes de Login.

```
scp <FITXER> <USUARI>@<TRANSFER>.bsc.es:/home/<NCT>/<USER>/<DIR>
```

Després, connecteu-vos normal a MareNostrum mitjançant el node de Login. Aquí usarem el clúster de CPUs de MareNostrum, que disposa de màquines amb 48 CPUs i entre 94GB i 384GB de memòria principal. Connecteu-vos tal com ho heu fet a les anteriors pràctiques, però tenint en compte que usarem MareNostrum CPU i no CTE:

```
ssh <USUARI>@<LOGIN>.bsc.es
```

Preparar l'entorn d'execució SLURM

El següent exercici consistirà en preparar l'script d'SLURM per a executar una instància de contenidor, passant les imatges per a entrenar VGG19 i generar noves imatges. Crearem un fitxer "start_style.sh", que modificarem i completarem com calgui:

```
#!/bin/sh

#SBATCH --output=output_%j.out
#SBATCH --error=output_%j.err
#SBATCH --nodes=1
#SBATCH --exclusive
#SBATCH --job-name=style_cap
#SBATCH --time=00:10:00

module load intel/2018.1 singularity/3.5.2

# THIS CREATES A SINGLE-NODE CLUSTER IN MARENOSTRUM-IV
# MN-IV NODES -> 48 CORES; 96GB RAM

IMAGE=container_02.sif

# COPY THE EXPERIMENT FILES INTO WORK DIRECTORY
HOME_DIR=/gpfs/home/nctXX/nctXXYYY/style_cap
WORK_DIR=/gpfs/scratch/nctXX/nctXXYYY/style_cap

mkdir -p $WORK_DIR
cd $WORK_DIR

# COPY EXPERIMENT, DATA FILES, AND IMAGE (IF NEEDED)
```

```
if [ ! -d $WORK_DIR/$IMAGE ]; then
        cp $HOME_DIR/$IMAGE $WORK_DIR/ ;
fi
cp $HOME_DIR/image_tr.jpg $WORK_DIR/
cp $HOME_DIR/image_ts.jpg $WORK_DIR/
<<AQUI COPIEU ALTRES FITXERS QUE PUGUEU NECESSITAR>>

START_TIME=`date +%s`

<<AQUI LA COMANDA PER EXCUTAR EL CONTENIDOR>>

echo "Exec Time:" $(expr `date +%s` - $START_TIME) " seconds"

# COPY RESULTING IMAGES BACK TO HOME
mkdir -p $HOME_DIR/experiment_${SLURM_JOBID}
mv $WORK_DIR/*.jpg $HOME_DIR/experiment_${SLURM_JOBID}/
mv $WORK_DIR/*.png $HOME_DIR/experiment_${SLURM_JOBID}/
<<AQUI COPIEU DE TORNADA EL QUE CALGUI>>

# THAT'S ALL, FOLKS!
echo "Bye!"
exit
```

Assegureu-vos d'indicar els directoris correctes a l'script (grup 'id -gn' i usuari 'id -un'). Recordeu d'usar les comandes de 'sbatch', 'squeue' i 'scancel' per a controlar l'execució.

Proves de Rendiment i Resultats

Finalment, comproveu el temps d'execució amb els diferents paràmetres indicats prèviament, però aquest cop usant el supercomputador en comptes del vostre portàtil.

EXERCICI 6: Un cop executat el contenidor a MareNostrum:

i) Quins son els fitxers mínims que hem hagut de pujar a MareNostrum des de la nostra màquina per a poder executar aquest darrer exercici?

j) Quan triga a executar-se la instància al supercomputador, usant els paràmetres originals "w" i "e"?

k) Podeu observar la nova imatge "test" reconstruïda? Adjunteu les tres imatges a la pràctica dins el document d'entrega. (Entrenament, test, test reconstruïda.)

IMAGE TRAINING	IMAGE TEST	IMAGE RESULTANT

l) Quina avantatge podem tenir de llançar-ho al supercomputador en comptes de fer-ho al nostre portàtil?

Final de la Pràctica

Tanqueu totes les VMs que tingueu en marxa. Responeu les PREGUNTES de forma breu, convertiu el document a **PDF**, i finalment **entregueu el qüestionari** (un per parella). A més a més, per a aquesta entrega, feu un **ZIP** o **TAR.GZ** amb:

1) El qüestionari, convertit a **PDF**.
2) La recepta "**container_02.singularity**"
3) L'script SLURM "**start_style.sh**".

Finalment, pugeu-ho a l'entrega corresponent del Racó de l'Estudiant.