Contenidors Linux per a entorns de Ciència i Enginyeria

Àngel Linares Zapater
Enginyeria Informàtica, 2n. cicle
Administració de Xarxes i Sistemes Operatius

Eduard Marco Galindo
Pierre Bourdin

7 de Juny de 2017

FITXA DEL TREBALL FINAL

Títol del treball:	*Contenidors Linux per a entorns de Ciència i Enginyeria*
Nom de l'autor:	*Àngel Linares Zapater*
Nom del consultor/a:	*Eduard Marco Galindo*
Nom del PRA:	*Pierre Bourdin*
Data de lliurament (mm/aaaa):	*06/2017*
Titulació o programa:	*Enginyeria Informàtica, 2n. cicle*
Àrea del Treball Final:	*Administració de xarxes i sistemes operatius*
Idioma del treball:	*Català*
Paraules clau	*Linux, containers, scientific computing*

Resum del Treball:

Tot i que el concepte de **contenidor** existeix des de fa molt temps en diferents sistemes operatius (*Solaris zones*, *AIX WPARs*, *BSD jails*, *Linux containers,* etc.), i s'ha emprat com a mètode per a aïllar diferents entorns concurrents en un mateix computador, ha estat la seva utilització recent com a tècnica de desplegament conjunt d'aplicacions i les seves llibreries el que ha popularitzat el seu ús entre els enginyers de software i els de sistemes.

Per altra banda, en el comunitat científica i tècnica, on els treballs estan cada cop més basats en **mètodes computacionals** (càlcul numèric, processament de dades, etc.), el concepte de **reproductibilitat dels resultats numèrics** per tercers està esdevenint cada cop més important per tal de validar els experiments computacionals i els resultats obtinguts.

L'aplicació del contenidors en entorns científics i tècnics podria esdevenir una pràctica potencialment important, tant per al desenvolupament i desplegament dels experiments numèrics i les solucions desenvolupades com per a facilitar la reproducció del seus resultats en entorns informàtics de tercers.

En aquest treball s'analitzen les característiques principals dels entorns informàtics per al treball científic i tècnic en l'àmbit del desenvolupament d'aplicacions i experiments computacionals en general, i en l'aspecte de la reproducció d'aquests resultats per tercers en particular.

Per a validar les solucions disponibles basades en contenidors es dissenya i implanta un cas pràctic d'un entorn representatiu i s'hi apliquen algunes d'aquestes solucions de contenidors Linux per a poder valorar la idoneïtat del seu ús en aquest tipus d'entorns.

Es pot comprovar que els contenidors Linux faciliten els treballs del cicle de la producció i la distribució de soluciones científiques i d'enginyeria i que poden ajudar a millorar la reproductibilitat computacional.

Abstract:

Although the concept of **container** can be found, since long time ago, in different operating systems (*Solaris zones, AIX WPAR, Jail BSD, Linux containers, etc.*), and has been used as a method to isolate different concurrent environments in a single computer, it has been its use as a recent technique for joint deployment of applications and their libraries what has popularized its use among software and systems engineers.

On the other side, in the scientific and technical community, where tasks are increasingly based on **computational methods** (numerical computation, data processing, etc.), the concept of **reproducibility of the numerical results** by third parties is becoming increasingly important in order to validate the computational experiments and its results.

The application of containers in scientific and technical environments could potentially become an important practice for both the development and deployment of numerical experiments and developed solutions, and to facilitate the reproduction of results in third party environments.

This document analyzes the main characteristics of scientific and technical environments, when Linux containers are used, in the development of applications and computational experiments in general, and in the aspect of reproduction of these results by third parties in particular.

In order to validate the available, based on containers solutions a design and implementation is done for a case study of a representative environment and some of these Linux containers solutions are then applied to assess the appropriateness of its use in such environments.

It can be shown that Linux containers facilitate the work cycle of production and distribution of scientific and engineering solutions and can help in improving the reproducibility of computational results.

Índex

1 Introducció....1
1.1 Context i justificació del Treball....1
1.2 Objectius del Treball....2
1.3 Enfocament i mètode seguit....2
1.4 Planificació del Treball....3
1.4.1 Descomposició del Treball....3
1.5 Restriccions del projecte....4
1.5.1 Restriccions tecnològiques....4
1.5.2 Restriccions culturals....4
1.6 Planificació temporal....4
1.6.1 Riscos en el calendari....5
1.6.2 Diagrama de Gantt....5
1.7 Breu sumari de productes obtinguts....7
1.8 Breu descripció dels altres capítols de la memòria....7
2 Estudi Teòric....8
2.1 Estudi de Necessitats....8
2.1.1 Caracterització dels entorns STEM....8
2.1.2 Identificació dels actors i els rols....9
2.1.3 Selecció dels rols característics....12
2.1.4 Les necessitats generals dels entorns STEM....13
2.1.5 Les necessitats particulars dels entorns STEM....14
2.1.6 Necessitats particulars dels «desenvolupadors»....15
2.1.7 Necessitats particulars dels «administradors»....18
2.1.8 Necessitats particulars dels «autors»....21
2.1.9 Necessitats particulars dels «revisors»....23
2.2 Estudi de Solucions amb Contenidors....25
2.2.1 Solucions basades en el propi sistema....25
2.2.2 Solucions basades en màquines virtuals....26
2.2.3 Solucions basades en contenidors....27
2.2.4 Anàlisi de les alternatives basades en contenidors....29
2.2.5 Comparativa de les solucions alternatives....35
3 Aplicació Pràctica....37
3.1 Disseny d'un Cas Representatiu....37
3.1.1 Definició d'un cas representatiu....37
3.1.2 Selecció d'una alternativa basada en contenidors....38
3.1.3 Disseny de l'aplicació pràctica....39
3.2 Implantació del Cas....41
3.2.1 Estació de treball Debian amb Docker....41
3.2.2 Contenidors Docker per a alternar compiladors....46
3.2.3 Contenidors Docker per a alternar llibreries....55
3.2.4 Contenidors Docker per a alternar aplicacions....60
3.2.5 Contenidors Docker per a execucions en paral·lel....64
3.2.6 Servidor Ubuntu compartit amb LXC/LXD....67
3.2.7 Repositori públic....76
3.2.8 Ús de contenidors Docker en Linux....81
3.2.9 Ús de contenidors Docker en Windows....83
3.2.1 0Ús de contenidors Docker en macOS....86
4 Anàlisi dels Resultats....89
4.1 Anàlisi del cas particular....89

4.1.1 Resultats del cas....89
4.1.2 Valoració econòmica....90
4.2 Anàlisi del cas general....91
4.2.1 Valoració de les solucions basades en contenidors....91
4.2.2 Aplicació dels contenidors a la reproductibilitat....92
5 Conclusions....93
5.1.1 El treball realitzat....93
5.1.2 El treball futur....94
6 Glossari....95
7 Bibliografia....96
7.1 Llibres i articles acadèmics....96
7.2 Recursos multimèdia a Internet....96
8 Annexos....98
8.1 GNU Free Documentation License....98

Llista de figures

1 Estructura de descomposició del treball.....3
2 Diagrama de Gantt de la planificació inicial.....6
3 MATLAB i Octave (en un contenidor) executant simultàniament el mateix codi font.....63
4 El contenidor publicat apareix en una cerca pública a DockerHub.....80

1 Introducció

1.1 Context i justificació del Treball

Els contenidors Linux han rebut en els darrers anys un impuls important en els entorns de desenvolupament de software i en els processos de desplegament associats, especialment des de l'aparició, popularització i amplia disponibilitat d'eines multiplataforma com ara *Docker* que faciliten enormement la seva utilització.

De fet, el concepte de **contenidor** (*container*) existeix des de fa força temps i ha estat implantant amb èxit en diferents sistemes operatius (*Solaris zones*, *AIX WPARs*, *BSD jails*, *Linux containers* etc.) com una tècnica fonamental per a organitzar i aïllar diferents entorns d'execució concurrents en un mateix computador. Ha estat, però, la seva utilització recent com a tècnica de desplegament de les aplicacions conjuntament amb les seves llibreries associades, sobretot en entorns d'informàtica en núvol, que ha popularitzat el seu ús entre els enginyers de software i els de sistemes.

Per altra banda, en el comunitat científica i tècnica, on els treballs estan cada cop més basats en mètodes computacionals (des del càlcul numèric i les simulacions al tractament estadístic de grans volums de dades, etc.), el concepte de **reproductibilitat** *(reproducibility)* dels resultats està esdevenint cada cop més un requisit important per a seguir complint amb un principi bàsic del mètode científic com és la repetició dels experiments de forma independent [Nat2016a].

L'aplicació de la tecnologia de contenidors en entorns científics i tècnics, l'àmbit anomenat **STEM** (*science, technology, engineering and mathematics*), seguint el camí iniciat en els entorns de desenvolupament de software i el desplegament conjunt d'aplicacions i llibreries, pot esdevenir una pràctica important. Tot i que les diferents aplicacions dels ordinadors als problemes científics i tècnics són molt diverses, moltes d'aquestes aplicacions es podrien beneficiar fàcilment de la tècnica de contenidors.

Tant pel que fa al desenvolupament en estacions de treball dels experiments numèrics i el seu posterior desplegament en els centres de càlcul propis o aliens, com per a facilitar la distribució i reproducció d'aquests experiments en entorns informàtics de tercers que puguin validar els resultats, la tecnologia dels contenidors podria ajudar en aquest repte.

A més, les branques computacionals de les ciències adquireixen, cada cop més, una importància similar a les branques experimentals, p.e. per a simular determinats experiments que serien impossibles de realitzar en un laboratori. I en l'àmbit de les aplicacions comercials, que es basen en algorismes cada cop

més sofisticats (molts cops sense que es pugui comprendre amb detall com els models generen les seves decisions finals, p.e. en el cas de les xarxes neuronals [Mit2016a]), només serà possible validar la seva idoneïtat general per la via de la replicació per tercers en entorns independents.

És en aquesta direcció que pren sentit analitzar com aquesta tecnologia de virtualització mitjançant contenidors que s'ha popularitzat recentment, especialment degut al fenomen *Docker*, pot tenir un ús efectiu en els entorns de ciència i enginyeria, tot validant-ne la seva utilitat mitjançant una aplicació pràctica en un entorn d'avaluació representatiu.

1.2 Objectius del Treball

La finalitat d'aquest treball es estudiar l'aplicació dels contenidors Linux als entorns STEM i, addicionalment, la viabilitat de l'ús d'aquesta tecnologia com una eina que pugui proporcionar o millorar la desitjada reproductibilitat de resultats computacionals.

Aquest objectiu principal es compon dels següents objectius parcials:

- Identificar la casuística principal per a l'administració d'un entorn informàtic de caire científic o tècnic.
- Comparar les alternatives de software lliure basades en contenidors Linux, tot analitzant la seva aplicabilitat a la casuística de l'entorn STEM.
- Definir un cas particular que pugui ser representatiu. Dissenyar i construir en entorn informàtic d'avaluació de la solució elegida aplicada al cas representatiu.
- Implantar una solució pràctica per a poder analitzar i extreure conclusions, a partir de la seva aplicació al cas representatiu prèviament definit.
- Valorar la viabilitat dels contenidors com a eina per a facilitar la feina STEM i millorar la reproductibilitat d'experiments computacionals.

1.3 Enfocament i mètode seguit

La finalitat del projecte és fonamentalment de naturalesa qualitativa, en estudiar el paper que poden tenir les solucions de contenidors Linux per a entorns STEM.

Malgrat això, es dissenyarà i implantarà una aplicació pràctica que demostri els conceptes analitzats i l'aplicació d'alguna de les solucions alternatives analitzades. Això ha de permetre fer aquesta validació tècnica de la solució elegida quan s'aplica en aquests entorns, així com una anàlisi mes quantitativa, si s'escau, d'aquesta aplicació concreta.

De l'anàlisi dels resultats en el cas particular, que haurà de ser mínimament representatiu, es podran extrapolar conclusions al cas general, si més no de forma qualitativa.

1.4 Planificació del Treball

1.4.1 Descomposició del Treball

El treball total a desenvolupar en el projecte es pot descompondre en 3 blocs de treball: l'Estudi Teòric, l'Aplicació Practica i l'Anàlisi de Resultats, organitzats segons es mostra en la següent figura i descrits a continuació:

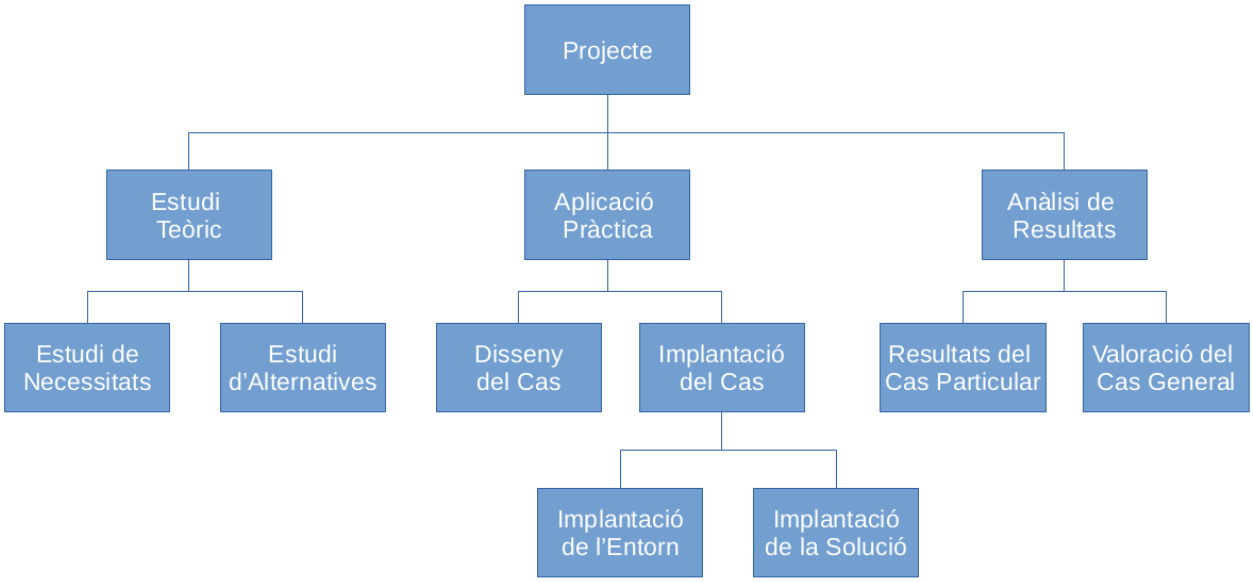


Figura 1: Estructura de descomposició del treball

Un **Estudi Teòric,** que es compondrà de dues parts diferenciades:

- Una anàlisi de les necessitats pròpies dels entorns STEM i les seves casuístiques principals, identificant els actors involucrats com ara els enginyers i científics que han de desenvolupar aplicacions o també els administradors del sistema que han de donar suport.
- La identificació, anàlisi i comparativa de diverses alternatives de solucions de software lliure que, utilitzant tecnologies de contenidors, siguin d'aplicació a les necessitats anteriorment detectades, com *LXC/LXD*, *Docker* o *CoreOS/rkt.*

Una **Aplicació Practica,** a partir de l'estudi anterior, per a obtenir resultats concrets per a la seva posterior anàlisi i valoració, tot fent:

- Primer, una definició d'un cas particular que sigui prou representatiu dels entorns STEM, per a elegir-ne una alternativa, i un disseny detallat de la seva aplicació.
- Segon, la implantació de la solució elegida entre les alternatives abans estudiades. Per a dur a terme això, serà necessari:
  - construir un entorn d'avaluació que representi l'entorn en estudi
  - implantar en aquest entorn els casos amb les solucions basades en contenidors elegides

Una **Anàlisi de Resultats,** basada en l'aplicació pràctica anterior, formada:.

- Per una anàlisi i valoració dels resultats i de l'aplicació de la solució al cas particular.
- Una valoració dels resultats extrapolats, en la mesura del possible, al cas general.

1.5 Restriccions del projecte

1.5.1 Restriccions tecnològiques

- El sistema operatiu de treball en l'entorn d'avaluació haurà de ser una distribució Linux.
- Les alternatives de solucions basades en contenidors que s'analitzaran hauran de ser de codi obert.
- Al menys s'inclouran en l'anàlisi d'alternatives els productes *Docker*, *LXC/LXD* i *CoreOS/rkt* que són les més populars en sistemes Linux.

1.5.2 Restriccions culturals

- La Memòria del projecte, la Presentació, així com la resta de qüestions acadèmiques relacionades amb el treball, es faran en català.
- El codi font informàtic dels lliurables intermedis (incloent els seus comentaris) que s'hagin de desenvolupar en l'aplicació pràctica i la documentació tècnica associada es farà en anglès.

1.6 Planificació temporal

El calendari detallat mostra la seqüenciació temporal de les tasques planificades en les fases d'Estudi Tècnic, Aplicació Pràctica i Anàlisi de Resultats en que s'executa el projecte.

La càrrega de treball planificada durant aquestes fases és de 16 hores setmanals, d'acord amb els 9 crèdits ECTS del Pla Docent, que representa un total de 224 hores de treball.

No es mostren en el diagrama de Gantt la resta de tasques acadèmiques associades al Projecte de Final de Carrera, com ara la preparació de la Presentació o la discussió posterior amb el Tribunal.

Les principals fites del projecte es mostren en relació amb les fites acadèmiques en la següent taula:

	Setmana	Activitat	Memòria
1	**27 feb. - 5 mar.**	Definició de la Proposta	
2	**6 mar. - 12 mar.**	Estudi de les necessitats	Pla de Treball
3	**13 mar. - 19 mar.**	Estudi d'alternatives	
4	**20 mar. - 26 mar.**		
5	**27 mar. - 2 abr.**	Disseny del cas particular representatiu	
6	**3 abr. - 9 abr.**		
7	**10 abr. - 16 abr.**	Creació de l'entorn d'avaluació	Memòria: Estudi Teòric
8	**17 abr. - 23 abr.**		
9	**24 abr. - 30 abr.**	Implantació de la solució	
10	**1 mai. - 7 mai.**		
11	**8 mai. - 14 mai.**	Anàlisi de resultats	
12	**15 mai. - 21 mai.**		Memòria: Aplicació Pràctica
13	**22 mai. - 28 mai.**	Revisió final de la memòria	
14	**29 mai. - 4 jun.**	Preparació de la presentació	
15	**5 jun. - 11 jun.**	Entrega del PFC	Memòria final Presentació
16	**12 jun. - 18 jun.**	Debat	

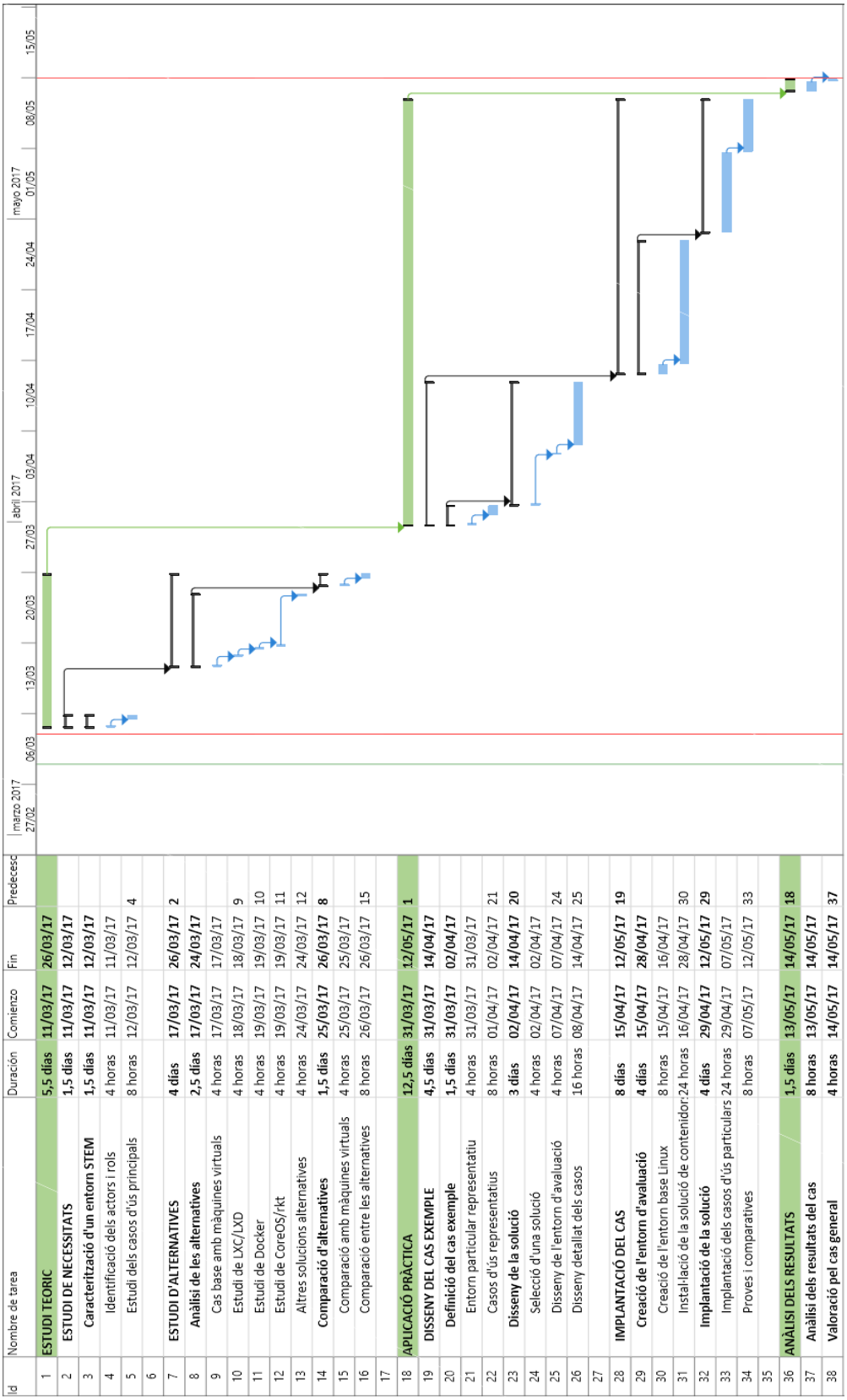
1.6.1 Riscos en el calendari

Per la naturalesa de les activitats a realitzar l'ordenació de les tasques és força seqüencial, la qual cosa significa que totes les tasques formen part del camí crític. Existeix doncs el risc, a l'estar la data final fixada, que qualsevol retràs en alguna activitat pugui afectar el resultat final.

Per a gestionar aquest risc, la col·lecció de casos d'ús a implantar en l'aplicació pràctica es mantindrà prou flexible per així permetre afegir o treure casos i poder respondre a les desviacions que puguin aparèixer, sense afectar però a l'estructura i finalitat del treball.

1.6.2 Diagrama de Gantt

El Diagrama de Gantt corresponent a les fases d'execució del treball es mostra en la següent imatge:



Id	Nombre de tarea	Duración	Comienzo	Fin	Predecesc
1	**ESTUDI TEORIC**	**5,5 días**	**11/03/17**	**26/03/17**	
2	**ESTUDI DE NECESSITATS**	**1,5 días**	**11/03/17**	**12/03/17**	
3	**Caracterització d'un entorn STEM**	**1,5 días**	**11/03/17**	**12/03/17**	
4	Identificació dels actors i rols	4 horas	11/03/17	11/03/17	
5	Estudi dels casos d'ús principals	8 horas	12/03/17	12/03/17	4
6					
7	**ESTUDI D'ALTERNATIVES**	**4 días**	**17/03/17**	**26/03/17**	**2**
8	**Anàlisi de les alternatives**	**2,5 días**	**17/03/17**	**24/03/17**	
9	Cas base amb màquines virtuals	4 horas	17/03/17	17/03/17	
10	Estudi de LXC/LXD	4 horas	18/03/17	18/03/17	9
11	Estudi de Docker	4 horas	19/03/17	19/03/17	10
12	Estudi de CoreOS/rkt	4 horas	19/03/17	19/03/17	11
13	Altres solucions alternatives	4 horas	24/03/17	24/03/17	12
14	**Comparació d'alternatives**	**1,5 días**	**25/03/17**	**26/03/17**	**8**
15	Comparació amb màquines virtuals	4 horas	25/03/17	25/03/17	
16	Comparació entre les alternatives	8 horas	26/03/17	26/03/17	15
17					
18	**APLICACIÓ PRÀCTICA**	**12,5 días**	**31/03/17**	**12/05/17**	**1**
19	**DISSENY DEL CAS EXEMPLE**	**4,5 días**	**31/03/17**	**14/04/17**	
20	**Definició del cas exemple**	**1,5 días**	**31/03/17**	**02/04/17**	
21	Entorn particular representatiu	4 horas	31/03/17	31/03/17	
22	Casos d'ús representatius	8 horas	01/04/17	02/04/17	21
23	**Disseny de la solució**	**3 días**	**02/04/17**	**14/04/17**	**20**
24	Selecció d'una solució	4 horas	02/04/17	02/04/17	
25	Disseny de l'entorn d'avaluació	4 horas	07/04/17	07/04/17	24
26	Disseny detallat dels casos	16 horas	08/04/17	14/04/17	25
27					
28	**IMPLANTACIÓ DEL CAS**	**8 días**	**15/04/17**	**12/05/17**	**19**
29	**Creació de l'entorn d'avaluació**	**4 días**	**15/04/17**	**28/04/17**	
30	Creació de l'entorn base Linux	8 horas	15/04/17	16/04/17	
31	Instal·lació de la solució de contenidor	24 horas	16/04/17	28/04/17	30
32	**Implantació de la solució**	**4 días**	**29/04/17**	**12/05/17**	**29**
33	Implantació dels casos d'ús particulars	24 horas	29/04/17	07/05/17	
34	Proves i comparatives	8 horas	07/05/17	12/05/17	33
35					
36	**ANÀLISI DELS RESULTATS**	**1,5 días**	**13/05/17**	**14/05/17**	**18**
37	**Anàlisi dels resultats del cas**	**8 horas**	**13/05/17**	**14/05/17**	
38	**Valoració pel cas general**	**4 horas**	**14/05/17**	**14/05/17**	**37**

Figura 2: Diagrama de Gantt de la planificació inicial

1.7 Breu sumari de productes obtinguts

El treball del projecte no genera un lliurable principal, com ara un aplicació o un sistema instal·lat.

Amb tot, es produeixen un seguit de lliurables intermedis que poden tenir el seu valor en el context adient. Especialment rellevant és la col·lecció dels principals casos d'ús particulars dels entorns STEM, en forma d'històries d'usuari, que són la base de treball per a l'avaluació de les solucions basades en contenidors en el cas pràctic desenvolupat.

1.8 Breu descripció dels altres capítols de la memòria

El cos de la Memòria s'organitza seguint les etapes de l'execució del projecte.

En el següent capítol s'analitzen les necessitats dels entorn STEM i les possibles alternatives basades en contenidors aplicables, tot comparant-ne les seves característiques.

Un capítol a continuació descriu una aplicació pràctica dels contenidors a un cas particular que pugui ser representatiu d'un entorn STEM.

Finalment es dedica un capítol apart a la valoració dels resultats i les conclusions finals del projecte.

2 Estudi Teòric

2.1 Estudi de Necessitats

2.1.1 Caracterització dels entorns STEM

En l'àmbit de la Ciència i l'Enginyeria, els ordinadors han jugat un paper molt important ja des de les seves primeres implantacions realment operatives a finals de la primera meitat del segle XX. El seu ús s'ha estès fins a tal punt que és difícil imaginar un projecte científic o tecnològic sense la participació dels ordinadors.

Entre la diversitat d'usos i necessitats que hi podem trobar en un entorn STEM, n'hi ha per una banda aquells que són comuns a altres àmbits, com ara l'ús dels ordinadors per a crear documentació o per a comunicar-se per correu electrònic a Internet, i n'hi ha també aquells més propis d'aquests entorns cientificotècnics com poden ser el càlcul numèric, les simulacions, la recopilació i tractament de grans volums de dades i de sèries temporals (p.e. de sensors), etc.

Per altra banda, els entorns STEM poden també ser molt diversos sigui en mida i sigui en complexitat. Així es pot trobar des de l'escenari de l'investigador individual fins als grans laboratoris científics amb xarxes distribuïdes a escala global.

Classificació dels entorns

Per tal de poder fer un estudi de les necessitats dels participants en aquests entorns, elegirem com un primer classificador dels entorns STEM la dimensió de la xarxa d'ordinadors i el nombre d'actors implicats en el projecte.

Atenent aquest criteri, podem definir tres tipus d'entorns, que anomenarem:

- **l'entorn STEM individual**: característic d'un ús personal del sistema informàtic per part del científic/a o enginyer/a, per exemple, per a fer recerca matemàtica amb càlcul simbòlic, per a la resolució d'un problema numèric concret en enginyeria, per a la simulació d'un ambient biològic, etc. treballant de forma individual.
- **l'entorn STEM d'equip**: resultat de la necessitat de compartir dades, algorismes i resultats entre membres d'un mateix equip en un projecte de més gran abast que l'individual. En aquest entorn també hi ha, a més, la necessitat de compartir els recursos informàtics comuns disponibles en la xarxa d'ordinadors, o fins i tot els servidors de càlcul numèric en un centre de càlcul.

- **l'entorn STEM obert**: propi dels projectes, siguin individuals o d'equip, que faran una distribució de les seves solucions d'enginyeria o dels seus resultats científics, que es proporcionaran a terceres pel seu ús o la seva revisió.

Com és evident, els entorns STEM reals són una combinació d'aquestes classes ideals. En cada moment, i segons la perspectiva adoptada, aquests entorns reals poden presentar les característiques d'uns o altres entorns ideals. Aquesta classificació servirà, però, per a poder fer un estudi més sistemàtic dels participants i els implicats en aquests entorns i per tant per a la comprensió de les seves necessitats i els rols que s'hi desenvolupen.

Per a enfocar l'anàlisi i centrar l'abast, en la resta del treball no tindrem en compte aquelles necessitats en entorns STEM que poden ser comunes amb altres més entorns més genèrics[1]. Així no tindrem en compte, entre d'altres necessitats:

- disposar d'eines per a la redacció de textos, documentació, presentacions, etc. conegudes habitualment com eines d'ofimàtica, ni tampoc les solucions de tipografia científica com ara *LaTex*.
- l'ús i accés genèric a internet amb navegadors, aplicacions de correu electrònic, etc. Tampoc l'administració i gestió genèrica de servidors web o de correu electrònic.
- la instal·lació de maquinari i dels sistemes operatius per a l'equip del projecte que són de propòsit general, com ara impressores compartides, emmagatzemament per a ús personal o per a còpies de seguretat, etc. (La instal·lació i configuració dels *runtimes* de contenidors en el sistema operatiu o del software i llibreries científiques pròpies de l'entorn sí que són objecte d'estudi).

2.1.2 Identificació dels actors i els rols

El nombre d'actors implicats en els entorns STEM pot variar molt: des de la persona que fa un treball estrictament individual fins a agrupacions d'equips en institucions arreu del mon. En qualsevol cas aquests actors assumiran diferents rols, sigui com a usuaris sigui com a administradors de les xarxes i sistemes involucrats.

Deixant de banda, com hem dit, els rols d'**usuari genèric** d'ordinador i també el d'**administrador de la xarxa,** comuns a altres entorns genèrics, i atenent a la classificació plantejada en el punt anterior es poden identificar els següents actors i els seus diferents rols.

1 Per a un exemple d'ús individual però genèric d'eines en entorn STEM veure [Kit2013a]

Entorn STEM individual

L'entorn individual es caracteritza per **una sola persona** que fa un treball cientificotècnic, però que a la vegada realitza personalment les tasques d'administració i configuració del seu sistema. Estem davant del científic o enginyer que programa aplicacions com part de la seva recerca o desenvolupament i que es configura l'ordinador personal per a aquestes tasques.

Aquest individu assumeix tres rols addicionals als genèrics respecte al seu sistema informàtic, que anomenarem:

- **desenvolupador**: entès en el sentit més ampli del terme per incloure tasques des de la concepció del model, a l'anàlisi i el disseny i la programació i prova de l'aplicació. En qualsevol cas, el resultat final és que es genera un codi font que es traduirà a codi objecte per a la seva utilització executant-lo en el mateix o en un altre ordinador.
- **administrador:** qui apart de les tasques comunes amb qualsevol administrador de sistemes (instal·lació del SO, configuració de la xarxa, etc) necessita instal·lar en el sistema operatiu les aplicacions, compiladors i intèrprets, així com llibreries i entorns de treball, que són propis i necessaris del seu àmbit de treball científic o tècnic.
- **revisor:** com a científic o enginyer especialitzat en el seu propi àmbit de coneixement, la mateixa persona s'encarrega de comprovar i revisar els resultats que s'obtenen de la seva solució prèviament desenvolupada.

El rol del desenvolupador no es restringeix només al codificador de programes en llenguatges de programació com C, Java o Python sinó que s'estén a llenguatges específics de domini com MATLAB, R o SPSS i a la programació d'aplicacions i entorns com SPICE o SeSam. En tots els casos existeix un codi font que es desenvolupa i després de ser processat genera uns resultats concrets sobre el problema en estudi.

Cal observar també que el rol d'usuari d'una aplicació d'enginyeria, com ara un sistema CAD/CAM, queda fora de l'estudi doncs es pot considerar un usuari genèric com ho seria un usuari d'una aplicació d'edició de vídeo o un usuari d'un simulador d'un sistema logístic en altres àmbits.

Entorn STEM d'equip

L'entorn d'equip es caracteritza per **un conjunt de persones** que col·laboren en un projecte científic o tècnic usant un **sistema informàtic compartit**. Aquest escenari inclou des del petit equip de dos o tres enginyers o científics fins als equips multidisciplinaris en grans laboratoris, centres de recerca o en els departaments de desenvolupament en empreses i organismes.

En aquesta classe d'entorns es pot observar que els actors involucrats poden, o bé assumir diferents rols vers el sistema, o bé algunes d'aquestes persones en l'equip s'especialitzen en unes tasques determinades. Podem identificar, però, els mateixos rols que en l'entorn individual malgrat que tant la quantitat com la qualitat de les tasques realitzades per cada persona poden ser diferents en cadascun dels aspectes. Així, s'identifiquen els següents rols:

- **desenvolupadors:** de nou entès en el sentit més ampli del terme incloent tots els rols possibles en l'àmbit del desenvolupament de software: analistes, programadors, provadors, etc. En aquest cas més d'un desenvolupador poden treballar de forma coordinada. Només ens interessen les seves necessitats en les vessants tècniques o científiques, i no considerarem d'altres més propis del àmbit del desenvolupament de programar com els repositoris de codi, els sistemes de control de versions, etc..
- **administradors**: que han de proporcionar als desenvolupadors i revisors el programari especialitzat del seu domini: compiladors i intèrprets, llibreries a entorns de treball específics, etc. Quan el rol correspon a una persona especialitzada ho haurà de fer de forma més professionalitzada i automatitzada que en l'entorn individual degut al major nombre de recursos compartits a administrar com servidors de càlcul, d'emmagatzemament, etc. pel conjunt dels membres de l'equip.
- **revisors**: que han de fer una feina especialitzada de revisió i validació del funcionament de la solució desenvolupada per altres rols i des resultats obtinguts però des de la perspectiva cientificotècnica del seu àmbit i no del estricte desenvolupament de codi.

Entorn STEM obert

Un cop es traspassa l'àmbit de l'equip que, per gran que sigui, fa un treball principalment enfocat a proporcionar resultats d'ús intern, sorgeix la necessitat de que aquest treball i els seus resultats es puguin comunicar de forma oberta a tercers. En aquest escenari més ampli direm que estem en un **entorn STEM obert**.

La característica fonamental de l'entorn obert és la presència una part aliena a l'individu o l'equip: els tercers. En aquest entorn les dues parts, equip intern i tercers, operen sota el paradigma de *productor/consumidor.* L'equip (o l'individu, si és el cas) produeix uns resultats que un tercer consumeix posteriorment. Normalment aquesta comunicació es voldrà fer de manera indirecta, és a dir, la comunicació no és punt a punt entre el productor i el consumidor sinó que els resultats del treball es dipositen i comparteixen en algun repositori o sistema intermedi accessible i, de forma asíncrona, seran recollits pel consumidor.

Aquestes parts poden ser, cadascuna d'elles, bé grups d'individus o bé estar formades per una sola persona, però de cara a l'anàlisi de la seva interacció els poden sintetitzar en dos rols:

- **autors:** que dins de l'equip productor s'encarreguen d'empaquetar, fer públics i distribuir els resultats i la resta dels lliurables que s'obtenen de la feina interna del projecte. Aquest rol aïlla doncs tant l'individu com tot un equip i la seva estructura interna, i s'encarrega de presentar a l'exterior el resultats de forma desacoblada a través d'un repositori accessible, sigui públic o privat.
- **revisors**: en aquest cas els actors que assumeixen aquest rol no pertanyen a l'equip productor sinó que obtenen dels autors, a través del repositori intermedi, les aplicacions o els resultats del treball científic o tècnic per al seu ús o revisió especialitzat en el seu propi entorn tercer..

Cal notar que el rol del revisor és lleugerament diferent del rol d'usuari genèric d'una aplicació STEM especialitzada doncs té unes necessitats específiques; concretament, en l'àmbit STEM, *la necessitat de poder reproduir i obtenir els mateixos resultats en el seu entorn particular que els presentats per l'equip productor*.

És en aquest sentir que resulta d'interès poder comprovar si els contenidors Linux poden aportar algun benefici a aquesta dinàmica, que està més enllà de la tècnica informàtica i és la base del mètode científic, entre els productors de contingut cientificotècnic i els tercers revisors.

2.1.3 Selecció dels rols característics

Qualsevol entorn STEM real es trobarà, òbviament, en algun punt intermedi d'aquest continu d'entorns *individual-equip-obert*. En qualsevol cas, però, podem veure que sempre trobarem alguns dels següents rols, que seleccionarem com a rols característics d'un entorn STEM.

- **desenvolupadors**: encarregats de desenvolupar els algorismes, experiments o aplicacions científiques i/o emprar les aplicacions especialitzades de l'àmbit d'enginyeria corresponent per a obtenir uns determinats resultats que representin una nova aportació a la comunitat cientificotècnica.
- **administradors:** encarregats de proveir, mantenir i configurar els equips de la xarxa pel que fa al programari tècnic i científic que es pugui necessitar en el domini especialitzat.
- **autors:** encarregats d'empaquetar, publicar i distribuir el programari i/o els resultats per a tercers en un entorn STEM obert

- **revisors**: aquells actors, siguin interns a l'equip o tercers, interessats en obtenir els resultats publicats i/o obtenir el programari creat per usar-los o per a validar-los en el seu propi sistema informàtic

Respecte al paper dels revisors es pot veure que en entorns compartits de certa dimensió i experiència és molt probable que la revisió interna dels resultats s'estructuri com si els revisors interns fossin tercers aliens. És a dir, un cop proveïda la infraestructura necessària (pel rol administrador) per a la distribució de la solució (pel rol autor) els revisors interns podrien actuar com a tercers, tot obtenint la versió empaquetada del repositori i instal·lant-la en el seus ordinadors. D'aquesta manera, no només es validaran els resultats tècnics del projecte sinó que s'utilitza i per tant es valida la infraestructura de distribució a tercers.

2.1.4 Les necessitats generals dels entorns STEM

Tots aquests rols característics s'afegiran als altres rols comuns amb altres entorns més genèrics, com el rol d'**usuari d'ofimàtica,** que evidentment també es donaran de forma simultània en un entorn cientificotècnic.

Així, altres necessitats generals en els entorns STEM seran les mateixes que les que es donen en altres entorns que també considerarem genèriques. Per exemple, els algorismes que es desenvolupen en entorns STEM poden processar dades que no estan ubicades en el node processador, sigui per motius de capacitat, d'ubicació de la font que els genera, etc. Cal donar accés des del node computador a les bases de dades del node d'emmagatzemament. Exemples en serien les aplicacions que processen dades en temps real, com ara solucions de *Big Data* sobre dades en xarxes socials, o les aplicacions financeres que operen directament sobre el propi mercat, o les aplicacions d'aprenentatge computacional (*machine learning*) sobre grans conjunts de dades. Altres casos són per exemple, en el cas de desenvolupament d'aplicacions científiques i tècniques, l'ús d'eines de control de versions del codi font, seguint les pràctiques del desenvolupament de software comunes. O en el cas de l'administració de xarxes, l'administració de còpies de seguretat o la gestió de la xarxa local entre els nodes que executen un càlcul.

Veiem, per tant, que aquesta caracterització no és totalment diferent a la que podríem trobar en altres àmbits diferents als estrictament científics com per exemple el desenvolupament d'aplicacions per a telèfons mòbils, o la producció de contingut multimèdia, on podríem identificar rols força similars.

És, de fet, aquest similitud que hi ha entre els entorns STEM i els entorns de desenvolupament de software que origina la tesi de partida d'aquest treball en formular la hipòtesi de que les pràctiques recents en l'ús de contenidors es poden traslladar i ser d'aplicació útil en entorns STEM.

Per a poder comprovar aquesta tesi cal doncs aprofundir en la casuística concreta que diferencia un entorn STEM dels altres entorns coneguts i així confirmar o no la hipòtesi sobre l'aplicabilitat dels contenidors en aquest àmbit particular.

2.1.5 Les necessitats particulars dels entorns STEM

Els entorns de ciència i enginyeria es caracteritzen principalment per l'ús intensiu dels recursos computacionals disponibles. Per a comparar i prenent com exemple una solució de comerç electrònic, que si bé aquesta ha d'estar dissenyada per atendre puntes de gran demanda, aquí cada transacció individual requereix normalment d'un processament poc complex i poc intens computacionalment. En canvi en els tractaments tècnics o científics les dades poden ser objecte d'intenses computacions quan no força complexes.

La necessitat d'un ús, tant intens com precís, del càlcul numèric en els problemes cientificotècnics pot estar també acompanyada de la necessitat de processar un gran volum de dades o de llargues sèries temporals, siguin aquestes generades per la pròpia solució o recollia de fonts externes.

Addicionalment, pot existir la necessitat de distribuir els càlculs a realitzar entre multitud de nodes de la xarxa, sigui per aplicar una solució de computació paral·lela a un problema complex, sigui perquè la pròpia naturalesa del problema requereix una arquitectura distribuïda per a la seva simulació o resolució.

Veiem doncs que les demandes del sistema informàtic que fan les aplicacions STEM són, al menys en alguns casos, diferents de la demanda que plantegen altres àmbits. Així les aplicacions HPC (**High Performance Computing**) esperen que el sistema amfitrió (que pot ser en la pràctica tot un clúster de gran capacitat) comparteixi i ofereixi la majoria de les seves capacitats i recursos a l'aplicació. En canvi, en l'àmbit de l'enginyeria del software, tendències recents com la *arquitectura basada en micro-serveis* es caracteritzen per instanciar un gran nombre de petits processos independents (micro-serveis), de caràcter efímer, només quan sigui necessari per a que col·laborin a proporcionar la funcionalitat esperada.

Aquesta distinció és important, doncs la recent difusió i expansió de l'ús dels contenidors es realimenta fortament d'aquestes arquitectures (on cada micro-servei s'implanta com un contenidor diferent). Si a això li sumen que els proveïdors de servis al núvol el que volen és maximitzar la compartició de recursos disponibles per tal de poder allotjar el major nombre de contenidors possible en un mateix sistema amfitrió, però que sense que s'impactin entre ells, veiem que poden haver requisits no funcionals força diferents en els entorns STEM als que originen actualment la difusió massiva de la tecnologia dels contenidors.

Per tant, de les necessitats pròpies dels entorns STEM veurem també que algunes es podran beneficiar de la tecnologia de contenidors mentre que altres no n'obtenen cap avantatge significatiu del seu ús.

Especificació de les necessitats en forma d'històries d'usuari

Un cop identificats els actors principals i més concretament els rols que aquests actors poden assumir en els entorns STEM, es por iniciar un recull i inventariat de les seves necessitats de forma més sistemàtica.

Podem usar les tècniques d'Enginyeria de Requisits per a fer un estudi d'aquestes necessitats. Tractant-se en primer lloc d'identificar les necessitats més bàsiques en entorns STEM, utilitzarem la tècnica simple però eficaç de les **histories d'usuari.** Així per a cada possible necessitat que es podria plantejar per un rol determinat, especificarem una història d'usuari que la descrigui, tot indicant els criteris d'acceptació per a poder validar la seva implantació.

2.1.6 Necessitats particulars dels «desenvolupadors»

El rol de desenvolupador en un entorn STEM té com a principal objectiu resoldre un problema determinat mitjançat la programació, en algun tipus de llenguatge d'alt nivell, d'un algorisme, computació o d'una simulació. El codi creat s'haurà d'executar en el seu entorn corresponent. Aquest entorn d'execució pot ser auto-contingut però moltes vegades farà ús de llibreries compartides instal·lades en el sistema operatiu. Acabada l'execució de la simulació o l'algorisme s'obtenen una sèrie de resultats que poden ser l'objectiu final o bé formar part d'un treball més ampli. Durant el temps que el desenvolupador treballa en desenvolupar aquesta solució o aplicació pot trobar-se en determinades situacions que requeriran de la seva actuació en tasques més enllà del propi problema en el seu àmbit de coneixement.

Alternança de compiladors o intèrprets

En ocasions pot ser necessari provar el codi d'una aplicació o una computació amb una nova versió del compilador o intèrpret del llenguatge de desenvolupament, sempre i quan després es pugui tornar a usar la versió anterior sense problemes. Exemples en serien:

- ha sortit la nova versió 7 del compilador *gcc* i es vol provar un algorisme desenvolupat amb la versió 6.
- es va desenvolupar fa temps una solució usant la versió 2.7 de l'intèrpret de *Python* i es vol provar l'algorisme en la nova versió 3.6 del llenguatge.

Alternança de llibreries científiques

De vegades el que canvia és una de les llibreries de base que usa el llenguatge o l'aplicació especialitzada i es vol actualitzar a una nova versió o bé substituir-

la per una altra sense canviar de compilador i intèrpret i, sobretot, sense haver de recompilar l'aplicació de la que potser no es té el codi font. Exemples serien:

- es vol provar la llibreria del projecte *openBLAS*[2] en substitució de les llibreries *BLAS*[3] estàndard en el sistema disponible
- es vol canviar la versió de les llibreries numèriques *NumPy*[4] o *SciPy*[5]

Alternança de programari especialitzat

Molta feina de caire científic i d'enginyeria es fa usant programari especialitzat, desenvolupat per tercers, que l'administrador instal·la en la estació de treball i el desenvolupador treballa sobre aquesta eina. Quan apareix una aplicació alternativa, p.e. una solució *open source,* es pot necessitar poder avaluar-la però sense alterar equip que ja té aplicacions potser comercials i de pagament instal·lades que no s'haurien d'alterar. Exemples serien:

- s'està usant l'aplicació comercial *MATLAB*[6] per a un desenvolupament específic i es vol avaluar l'alternativa de codi obert *GNU Octave*[7]
- s'està usant *SPSS*[8] per tractament estadístic de dades del problema i es vol avaluar l'alternativa de codi obert *GNU PSPP*[9]

Execucions en paral·lel

Molts desenvolupaments específics dels entorns STEM requereixen d'una gran capacitat de processament numèric que tenen una solució més òptima, quan no més natural, de caire paral·lel. És a dir, l'algorisme que resol el problema es descompon en subproblemes menors que es distribueixen en un nombre gran de nodes per a obtenir la capacitat de computació agregada. Exemples serien:

- una aplicació que ha de processar un gran nombre de dades però que el càlcul sobre cada dada és majorment independent de la resta de dades. Cada subcàlcul sobre un element de dades es pot executar en un node diferent i en paral·lel per a recollir i agregar a posteriori els resultats parcials.
- un algorisme per a un sistema distribuït format per diversos nodes o bé una simulació sobre un conjunt d'agents d'intel·ligència artificial que han d'operar independentment tot i que es comuniquen entre ells per a generar un comportament emergent propi.

2 http://www.openblas.net
3 http://www.netlib.org/blas
4 http://www.numpy.org
5 https://www.scipy.org
6 https://www.mathworks.com/products/matlab.html
7 https://www.gnu.org/software/octave
8 https://www.ibm.com/analytics/us/en/technology/spss
9 https://www.gnu.org/software/pspp

Si bé aquesta necessitat pot ser compartida amb altres àmbits, existeix tot un conjunt de desenvolupaments derivats de les aplicacions cientificotècniques sota aquest paradigma de processament en paral·lel. Un estàndard derivat d'aquest requisit és el **Message Passing Interface (MPI)** que té diverses implantacions com **MPICH**[10] o **Open MPI**[11]. El desenvolupador pot tenir la necessitat de simular o crear multitud de nodes sense la càrrega de recursos que això representa.

Històries d'usuari dels «desenvolupadors»

Aquestes necessitats pels desenvolupadors es poden formalitzar en aquest conjunt d'històries d'usuari:

DEV01	Alternança de compiladors o intèrprets
Com a	desenvolupador de software científic i tècnic
vull	poder alternar el compilador o intèrpret sense alterar l'entorn inicial
per tal de	poder comprovar el codi usant una versió diferent del llenguatge
Criteris d'acceptació:	
es verifica que s'ha actualitzat la versió del compilador o intèrpret	
es pot provar el codi amb el nou compilador o intèrpret	
es pot retornar a la configuració inicial	

DEV02	Alternança de llibreries
Com a	desenvolupador de software científic i tècnic
vull	poder actualitzar una llibreria o canviar a una compatible
per tal de	poder disposar de versions més recents o versions més optimitzades sigui pel meu algorisme o el meu software instal·lat
Criteris d'acceptació:	
es verifica que s'ha actualitzat la llibreria a la nova versió	
es pot comprovar si hi ha canvis en el funcionament del programari	
es pot tornar a la llibreria anterior sense canvis en l'entorn inicial	

DEV03	Alternança de programari
Com a	desenvolupador de software científic i tècnic
vull	poder instal·lar, executar i desinstal·lar programari alternatiu

10 http://www.mpich.org/
11 https://www.open-mpi.org/

per tal de	poder avaluar la seva conveniència sense alterar el meu entorn inicial de treball
Criteris d'acceptació:	
es pot avaluar el nou programari	
es pot retornar al programari inicial sense pèrdua de configuració	

DEV04	**Execucions en paral·lel**
Com a	desenvolupador de software científic i tècnic
vull	poder executar la meva aplicació en múltiples sistemes
per tal de	aprofitar el paral·lelisme del càlcul en varis sistemes concurrents
Criteris d'acceptació:	
es pot desplegar un executable i una configuració de nodes en paral·lel	
es pot comprovar que s'executa en paral·lel en diversos nodes	

2.1.7 Necessitats particulars dels «administradors»

El rol d'administrador en un entorn STEM es troba amb necessitats addicionals a les pròpies de l'administrador genèric de xarxes degut a l'especialització de les eines tècniques del usuaris i dels desenvolupadors que demanen siguin resoltes per l'administrador.

Hi ha doncs dues classes de necessitats: les demandades per la resta de rols per a les seves respectives funcions (que fan referència al procés creatiu dels desenvolupadors i de distribució dels autors) i les pròpies de l'administrador per a la seva gestió del sistema dels entorns STEM (més pròpies de l'administració de les xarxes i els centres de càlcul com proveïment de clústers, accés a recursos de computació de l'amfitrió com **GPUs** dedicades, etc.).

Ens centrarem més en les necessitats STEM més properes al cicle de desenvolupament i publicació de solucions per tercers que no als processos dels serveis interns de proveïment i administració d'entorns d'alt rendiment o altament compartits, com els sistemes HPC (que primen usar tots els recursos del host) o els serveis Cloud (que primen compartir al màxim els recursos del host)

A part, doncs, d'haver d'administrar la xarxa i de proveir els seus usuaris de maquinari de requisits específics, els administradors es poden trobar amb aquestes necessitats particulars:

Actualització coordinada del sistema operatiu

Un cop s'han instal·lat les estacions de treball especialitzades i s'hi ha instal·lat el programari necessari per l'entorn específic, l'administrador ha de mantenir aquests sistemes actualitzats moltes vegades de forma sincronitzada per tots els equips. Més enllà de les tasques genèriques d'actualització del sistema operatiu hi ha determinats components del sistema que, pel seu impacte en les aplicacions científiques, són d'especial interès pel rol d'administrador STEM. Exemples serien:

- gestionar l'actualització del *kernel* Linux en totes les estacions. Aquest nucli que com es veurà està compartit per tots els contenidors ubicats dins d'un mateix amfitrió, i podria estar optimitzat per a la feina pròpia del servidor, i per tant l'impacte de la seva actualització és immediat en tots els contenidors.
- actualitzar de forma sincronitzada un conjunt d'estacions que han de mantenir els mateixos components del sistema operatiu i les mateixes versions, p.e. scripts del sistema d'inici o de tasques programades.

Actualització coordinada de llibreries científiques

De la mateixa manera, en determinats escenaris el manteniment sincronitzat de llibreries compartides en totes les estacions i servidors cau sota la responsabilitat del rol administrador. Exemple seria:

- el rol desenvolupador, després d'alternar llibreries determina que cal actualitzar totes les estacions dels membres del projecte a un conjunt de llibreries alternatives, p.e. *OpenBLAS*.

Desplegar aplicacions en el centre de computació

En entorns d'equip és habitual que es dediquin recursos especialitzats a l'execució i els càlculs de les aplicacions desenvolupades pels equips dels projectes. Aquestes aplicacions un cop desenvolupades en les estacions de treball en despleguen en entorns de producció de major capacitat. Exemples serien:

- el desplegament de solucions i experiments numèrics en centres de supercomputació en la xarxa interna de l'equip
- el desplegament de les aplicacions desenvolupades en proveïdors externs al núvol com *Amazon AWS*[12] o *Google CloudPlatform*[13] pel seu ús públic

12 https://aws.amazon.com
13 https://cloud.google.com

Orquestrar conjunts de contenidors

Quan es comencen a usar contenidors, especialment quan es fa un ús intensiu dels mateixos, sigui en una solució específica aplicant arquitectura basada en micro-serveis o en un centre de computació que executi centenars, sinó milers, de contenidors, apareix una nova necessitat pels administradors: la de manegar conjuntament, sigui en la seva creació, distribució en diferents nodes de computació, configuració simultània, etc., tots aquests contenidors.

Tot un conjunt d'eines s'han anat desenvolupant per a resoldre aquesta necessitat derivada de la popularització dels contenidors, com *Kubernetes*[14] de Google o *Docker Swarm*[15] de Docker.

Aquesta necessitat però no és pròpiament una característica dels entorns STEM sinó una necessitat compartida amb altres entorns, de forma anàloga com ho seria la necessitat d'accedir a grans volums de dades en aplicacions científiques. Per tant no la considerarem a l'hora de definir les històries d'usuari particulars dels administradors en entorns STEM.

Històries d'usuari dels «administradors»

Aquestes necessitats dels administradors es poden formalitzar en aquest conjunt d'històries d'usuari:

ADM01	Actualització coordinada del sistema operatiu
Com a	administrador del sistema
vull	poder actualitzar el sistema operatiu d'un conjunt d'estacions o de servidors de la xarxa de forma simultània i coordinada
per tal de	poder mantenir la xarxa segura i actualitzada
Criteris d'acceptació:	
es verifica que s'ha fet l'actualització al conjunt	
es verifica que cada nou node que s'executa està actualitzat	

ADM02	Actualització coordinada de llibreries científiques
Com a	administrador del sistema
vull	poder actualitzar llibreries de programari científic dels equips de la xarxa
per tal de	que els usuaris disposin de la darrera versió més actualitzada
Criteris d'acceptació:	
es verifica que s'ha fet l'actualització al conjunt	
es verifica que cada nou node que s'executa està actualitzat	

14 https://kubernetes.io/
15 https://docs.docker.com/swarm

ADM03	Desplegar aplicacions en el centre de computació
Com a	administrador del sistema
vull	poder desplegar aplicacions en el centre de computació
per tal de	que es executar en un entorn de major capacitat
Criteris d'acceptació:	
l'aplicació s'instal·la i executa en el centre de computació	
el desenvolupador pot recollir els resultats de l'execució	

2.1.8 Necessitats particulars dels «autors»

Quan en un projecte STEM s'han completat els desenvolupaments interns i es volen publicar els seus treballs per a fer-los disponibles a tercers, el rol d'autor té la necessitat d'empaquetar aquesta solució i distribuir-la.

La estratègia més adient és desacoblar el lliurament a tercers en el temps, de manera que la solució es fa disponible en un repositori intermedi, sigui d'accés públic o restringit, per tal de que els tercers revisors l'obtinguin a la seva conveniència.

Empaquetar aplicacions, llibreries i dades

Sigui amb l'objectiu de la portabilitat o potser més específicament de la reproductibilitat de resultats, l'autor està interessant en generar un paquet únic amb l'aplicació i les seves llibreries. El format del contenidor i el seu sistema d'arxius associar permet distribuir tots els elements necessaris: codi objecte o codi font, llibreries compartides i fins i tot dades per als càlculs, de forma que sigui transportable entre sistemes. Un exemple paradigmàtic es:

- s'ha desenvolupat una aplicació que resol un problema tècnic i es vol oferir a la comunitat per a que la utilitzi. Els autors volen empaquetar la seva solució fins al punt que sigui distribuïble com una sola unitat.

Publicar aplicacions i resultats

Un cop creats aquests paquets de distribució, aquests també es poden fer arribar als tercers que correspongui. En el cas d'entorns compartits es poden fer arribar als revisors interns per a l'avaluació en el seu àmbit de coneixement. En el cas dels entorns oberts, es fan públics i disponibles els paquets de forma estandarditzada per a la seva descàrrega posterior. Exemples serien:

- en el cas d'entorn STEM d'equip, els contenidors (les seves imatges, de fet) es poden deixar en un repositori comú intern com una unitat de disc de la xarxa, etc.

- en el cas d'entorn STEM obert, és una pràctica habitual deixar els paquets en un repositori estàndard (com per exemple, *DockerHub*[16]) amb tota la meta-informació necessària per a descriure i documentar el paquet de cara a tercers.

Si un equip implanta les eines necessàries per fer públics els paquets en un repositori obert serà força habitual que s'acabi utilitzant el mateix sistema per a la distribució interna als revisors de l'equip.

Històries d'usuari dels «autors»

Aquestes necessitats dels autors es poden formalitzar en aquest conjunt d'històries d'usuari:

AUT01	**Empaquetar aplicacions i resultats**
Com a	autor d'aplicacions STEM
vull	poder empaquetar les aplicacions i dades
per tal de	transferir-los a un repositori intermedi estandarditzat
Criteris d'acceptació:	
els conjunt quedi disponible per al públic elegit	
els tercers no hauran d'instal·lar components addicionals	

AUT02	**Publicar aplicacions i resultats**
Com a	autor d'aplicacions
vull	poder publicar les aplicacions
per tal de	que tercers les puguin obtenir i usar en els seus sistemes
Criteris d'acceptació:	
els tercers poden usar el seu sistema operatiu encara que sigui diferent	

Com podem veure en aquest cas, per a validar l'empaquetat és possible que sigui necessari poder-lo publicar i descarregar-lo de nou pels mateixos autors. Les històries d'usuari de desenvolupadors i autors s'encadenen i, de fet, moltes vegades és el mateix actor que va assumint diferents rols en el cicle. Això és propi dels entorns STEM sobretot individuals, i aquí és on la tècnica dels contenidors hauria d'aportar més beneficis a aquests actors: els científics i enginyers que volen centrar-se en la seva feina principal i simplificar els processos auxiliars i que no són especialistes en distribució i desplegament de software.

16 https://hub.docker.com

2.1.9 Necessitats particulars dels «revisors»

Des del punt de vista del revisor (sigui intern o extern) interessat en obtenir una aplicació publicada pels autors per a usar-la en el seu sistema o per a reproduir un resultat computacional, el paradigma del repositori públic de contenidors simplifica també molt les seves necessitats.

Obtenir aplicacions de repositoris

El fet d'estandarditzar els repositoris i usar aquests com a mecanisme de transport de solucions unifica i simplifica les necessitats dels revisors, que són d'una mateixa classe però també poden concretar-se de forma diferent per a cada autor.

En conèixer el format del paquet distribuït en un repositori i disposar de meta-informació en la publicació, la tasca d'obtenir però sobretot d'instal·lar la solució la pot arribar a realitzar el propi revisor sense haver de necessitar la intervenció d'un administrador especialitzat en el seu propi entorn.

Un exemple característic seria:

- un revisor interessat en una determinada publicació de recerca es connecta al repositori on ha estat publicat un contenidor Docker amb el codi i les dades proporcionades pels autors i n'obté una còpia de la imatge del contenidor.

Reproduir els resultats publicats per tercers

Per altra banda, el fet de descarregar un desenvolupament de tercers genera unes noves necessitats en haver d'adaptar els seus sistemes als requisits de l'aplicació. Un exemple seria:

- el revisor crea un nou contenidor en el seu propi sistema que el permet executar el codi publicat en la imatge de contenidor que l'autor abans ha descarregat del repositori.

Un requisit essencial, però, és que el sistema del revisor tingui instal·lat el *runtime* de contenidors capaç d'executar el paquet descarregat. Com que el revisor es podria trobar amb publicacions en diversos sistemes no compatibles, la pressió de la comunitat cientificotècnica anirà en el sentit de simplificar el nombre de formats disponibles.

També en aquest punt, on la gamma d'implantacions, per a múltiples sistemes operatius, ofertes per una solució de contenidors ha de jugar a favor de elegir finalment un sistema *de facto* o un altre, si es vol tenir la llibertat d'elegir el sistema operatiu a l'hora d'usar i validar una solució prèviament publicada.

Històries d'usuari dels «revisors

Aquestes necessitats dels autors es poden formalitzar en aquest conjunt d'històries d'usuari:

REV01	Obtenir aplicacions de repositoris
Com a	revisor i usuari d'aplicacions de tercers (o de l'equip)
vull	poder obtenir aplicacions de repositoris externs (o interns)
per tal de	d'executar-les en el sistemes d'avaluació
Criteris d'acceptació:	
es pot obtenir l'aplicació del repositori	
es pot instal·lar i executar l'aplicació sense instal·lacions addicionals	

REV02	Reproduir els resultats publicats
Com a	revisor de resultats computacionals
vull	poder obtenir aplicacions i dades corresponents a uns resultats
per tal de	poder replicar les computacions i replicar-ne els seus resultats
Criteris d'acceptació:	
es pot obtenir l'aplicació del repositori public	
es pot instal·lar i executar l'aplicació sense instal·lacions addicionals	
es pot verificar la reproductibilitat dels resultats publicats	

2.2 Estudi de Solucions amb Contenidors

2.2.1 Solucions basades en el propi sistema

La primera aproximació per donar solució a aquestes necessitats consisteix en modificar el propi sistema. Aquesta és la solució natural especialment en un entorn individual on el científic o enginyer no està especialitzat en l'administració de sistemes i molt menys en la tecnologia dels contenidors.

En aquesta situació, les actuacions directes sobre el sistema són les que aniran causant la problemàtica i dificultats que van originar el naixement dels contenidors i el posterior desenvolupament de solucions més complexes i automatitzades per a gestionar-los.

Poden ser problemes típics, que es donarien en prendre aquesta aproximació:

- La modificació del sistema principal de treball provoca la desconfiguració del mateix de manera que no es pot seguir usant com anteriorment al canvi, amb l'impacte sobre la feina inicial del científic o enginyer.
- El sistema de treball no té prou capacitat de càlcul per resoldre el problema en estudi per la qual cosa no es pot concloure si la solució desenvolupada és correcte o no.
- El servidors dedicats a executar la càrrega de treball en estudi s'han d'instal·lar i configurar expressament per a cada problema o experiment limitant-ne la seva capacitat de reutilització en entorns compartits.
- Els resultats obtinguts tenen una forta dependència de la estació de treball o els servidors de càlcul emprats, dificultat la posterior reproductibilitat d'aquests per tercers.

Tots aquests problemes són els que generen la necessitat de buscar i construir altres solucions

L'operació *chroot*

Una de les primeres solucions per a intentar disposar de múltiples entorns aïllats i independents va ser el desenvolupament, en els sistemes UNIX, de l'operació *chroot* (abreviació de *change root filesystem*). Mitjançant aquesta operació es pot canviar el sistema de fitxers arrel d'un procés per a que no sigui el mateix que el del procés pare. Així, si per exemple es té disponible una imatge d'un sistema d'arxius en un subdirectori (p.e. per a un sistema encastat), després de fer *chroot* a aquest subdirectori el procés pensa que aquest subdirectori és l'arrel del sistema d'arxius per la qual cosa ja no pot canviar de directori de treball cap a directoris que estiguin en un nivell superior. Aquesta tècnica s'usa com mecanisme de seguretat, p.e, per aïllar usuaris en

els servidors FTP, que un cop connectats només es poden moure pel seu directori i pels subdirectoris però no accedir a directoris del sistema amfitrió.

El principal inconvenient és que aquests usuaris sí que comparteixen la resta d'espais de noms com usuaris, processos, etc. i un error (p.e. en el procés servidor FTP) o una incorrecta configuració en el sistema amfitrió podria iniciar un escalat de privilegis no desitjat.

2.2.2 Solucions basades en màquines virtuals

Una segona aproximació més complexa consisteix en usar màquines virtuals. Una **màquina virtual** és una simulació parcial o total d'un nou ordinador feta per una aplicació especialitzada anomenada **hipervisor**. Aquesta aplicació emula el maquinari d'un sistema de manera que el codi binari que s'hi executa en aquesta màquina no distingeix que està essent interpretat per un software doncs funciona igual que si s'estès executant en un processador físic.

Els inconvenients de les màquines virtuals es poden resumir en dos aspectes:

- Els hipervisors fan una simulació per software del hardware virtualitzat. Això significa que cada instrucció màquina de la solució executada ha de ser interpretada i emulada en el processador de l'equip amfitrió amb la pèrdua directa i important de rendiment final per l'aplicació emulada.
- La simulació d'una nova màquina virtual requereix una còpia completa no només del sistema d'arxius sinó també del nucli del sistema operatiu simulat. En el cas d'haver de simular un nombre elevat de màquines la suma d'espai en disc necessari creix ràpidament. No menys important des del punt de vista de l'administrador és l'increment de la complexitat d'haver de mantenir tot el programari (aplicacions i llibreries, o el propi sistema operatiu) en un nombre elevat d'equips.

Les solucions basades en màquines virtuals poden ser útil en entorns individuals o en equips de reduïda dimensió però són poc escalables quan l'equip ha de créixer en nombre de participants o quan les solucions s'han de compartir i distribuir a tercers. Tot un conjunt d'eines han aparegut per tal de permetre administrar grans conjunts de màquines virtuals i tots els recursos associats, però en alguns escenaris això representa una sobrecàrrega que podria ser innecessària i contraproduent (p.e. en un solució basada en micro-serveis).

Durant anys, però, aquesta ha estat i segueix sent una solució prou emprada i que funciona adequadament en determinats escenaris. És quan cal distribuir aquestes màquines virtuals, sigui per executar-les en servidors compartits o sigui per compartir-les amb tercers per la seva avaluació quan es manifesten aquests problemes d'escalabilitat en espai i també de velocitat.

2.2.3 Solucions basades en contenidors

Què són els contenidors Linux

Els contenidors són entorns del sistema operatiu encapsulats de manera que els processos que s'executen dins del contenidor no tenen cap informació ni accés directe a la resta del sistema amfitrió que els conté o a d'altres contenidors que s'executen en paral·lel al mateix sistema.

La funció primària dels contenidors, per tant, és aïllar l'entorn contingut de l'entorn del sistema amfitrió p.e. un sistema GNU/Linux. Amb tot aquest nou entorn segueix formant part del sistema operatiu amfitrió i per tant comparteix el seu mateix *kernel*. La diferència és que dins del contenidor es simula un nou **espai d'usuari** (***userland***) que resulta aparentment diferent per als processos que hi corren a dins.

Això s'aconsegueix en Linux gràcies a la capacitat que proporciona el nucli de gestionar els anomenats **espais de noms del nucli** (***kernel namespaces***), que són dominis de recursos del nucli adreçables per nom o identificador.

El nucli Linux suporta una varietat d'espais de noms, essent els principals:

- **identificadors de processos**: cada procés actiu en el sistema operatiu té un identificador únic (**PID, process identificator**). Amb la tècnica dels contenidors aquests identificadors es reanomenen dins del contenidor de manera que només hi apareixen visibles els propis processos del contenidor però no són visibles ni accessibles els processos externs. És més, dins del contenidor es crea de nou un PID 1 virtual que normalment representa el primer procés que inicia el nucli Linux, associat al sistema d'inici sigui /sbin/init de System V o /sbin/systemd en Systemd. Quan s'executa directament un comandament en un contenidor sense cap sistema d'inici, aquest rep el PID 1.
- **identificadors d'usuaris**: de la mateixa manera, tots el usuaris tenen un identificador (**UID, user identificator**) que s'usa entre altres funcions per determinar els privilegis de l'usuari i la seguretat dels seus arxius. Anàlogament tots els grups d'usuaris tenen el seu propi identificador (**GID, group identificator**). Aquest és un dels punts més delicats en la implantació de les solucions de contenidors, doncs un usuari que pogués obtenir el UID 0 en el contenidor (p.e. usant *sudo*) en cas de falles de seguretat podria arribar a tenir privilegis de *root* fora del contenidor.
- **dispositius de xarxa**: una diferència important dels contenidors Linux respecta a la simple operació de *chroot*, (que canvia el sistema de fitxers arrel per un procés en Linux) és que cada contenidor disposa del seu propi espai de noms. Això a la pràctica significa que cada contenidor

disposa dels seu propi conjunt de dispositius de xarxa (*network devices*), amb la seva pròpia pila TCP/IP, les seves adreces IP, configuracions I taules d'enrutament, etc.

- **punts de muntatge:** també els punts de muntatge (*mount points*), com el de sistemes d'arxius, es repliquen en el nou contenidor, però els canvis que allí s'hi facin ja no són visibles en la resta de contenidors. Aquí, tot i que la funció esperada del contenidor és aïllar-se de la resta d'entorns en un mateix sistema, de vegades pot ser convenient tot el contrari per exemple per a poder accedir i/o modificar un directori d'usuari del sistema amfitrió des dels processos del contenidor.

Aquesta gestió dels espais de noms es complementa amb una altra funcionalitat de Linux, els **grups de control** (**control groups, cgroups**) que permet controlar i gestionar els recursos assignats i consumits (sigui temps de CPU, mida de la RAM, accessos a disc o la xarxa, etc) per un grup determinat de processos.

Combinant aquestes característiques es poden crear les abstraccions anomenades contenidors on un grup de processos s'aïlla de la resta de processos i comparteix un conjunt de recursos del sistema amfitrió tot tenint la impressió, dins el contenidor, que es tracta d'una màquina aïllada i diferent d'aquest. És que s'anomena **virtualització a nivell de sistema operatiu**.

Una funció secundària dels contenidors és de la permetre el transport, de forma unitària, de l'entorn desenvolupat a altres sistemes amfitrions. Aquesta aplicació és la que ha popularitzat el projecte Docker en permetre als desenvolupadors empaquetar el seu codi amb les llibreries necessàries per la seva execució, de manera que pugui instal·lar-se i executar-se en un altre amfitrió que no disposi inicialment d'aquestes llibreries i sense haver de modificar l'amfitrió instal·lant-les per una l'aplicació concreta. Aquesta aproximació és coneguda com **virtualització d'aplicacions** a través dels anomenats contenidors d'aplicacions o *apps containers.*

Aplicabilitat dels contenidors Linux a entorns STEM

La reduïda dimensió dels contenidors d'aplicacions, que en principi només contenen l'aplicació i algunes llibreries, és molt interessant pels administradors de sistemes en entorn compartits i fins i tot pels proveïdors d'infraestructures al núvol. Així, en un sol sistema amfitrió es poden executar multitud de sistemes en forma de contenidors sense haver d'aixecar màquines virtuals per a cadascun d'ells. D'aquesta manera es poden estalviar l'espai en disc dels programes de base del sistema operatiu i dels nuclis que estarien replicats en cadascuna de les màquines virtuals.

En els entorns STEM, però, un contenidor pot tenir un requisit addicional d'alta capacitat computacional que cal tenir en compte. Normalment, en una solució

cientificotècnica en un contenidor el que interessa en primer lloc és assegurar-se el màxim de recursos de l'amfitrió (computacionals i emmagatzemament). Això competeix amb la gestió de recursos, que es fa des de l'amfitrió i sense cap control per l'aplicació que corre al contenidor, que tendeix a minimitzar els recursos comuns per a distribuir-los amb altres contenidors. Veiem doncs que en els entorns STEM apareixen dues classes de solucions: aquelles que poden operar en amfitrions altament compartits i aquelles que estarien dissenyades per a utilitzar el màxim de recursos disponibles de l'amfitrió. Es pot donar el cas, fins i tot, que una aplicació virtualitzada en un contenidor hagués de tenir accés directe a elements de maquinari físic en l'amfitrió com p.e. GPUs per a computació massiva en paral·lel.

Un altra avantatge d'aquesta aproximació basada en contenidors és la capacitat de personalitzar l'espai d'usuari del nou sistema virtual a les necessitats de l'aplicació que s'executa, no només a nivell de proporcionar les llibreries que necessita en temps d'execució sinó en forma d'optimitzacions i personalitzacions de fitxers de configuracions i altres aspectes més específics per a l'aplicació. Això és d'interès de nou pels administradors de servidors compartits que així no han de personalitzar el sistema a causa d'una aplicació concreta, canvis que podrien entrar en conflicte amb les personalitzacions necessàries per a altres aplicacions.

Finalment, des del punt de vista dels entorns STEM, interessa aconseguir la màxima compatibilitat dels entorns virtualitzats de manera que els contenidors (les aplicacions instal·lades a dins) s'executin i generin els mateixos resultats que s'obtindrien en amfitrions diferents o en màquines natives.

Les necessitats dels entorns STEM són, doncs, diferents de les necessitats d'altres entorns també pel que fa l'ús dels contenidors, i caldrà tenir en compte aquests requisits a l'hora de seleccionar solucions basades en contenidors.

2.2.4 Anàlisi de les alternatives basades en contenidors

Com hem vist, el concepte de contenidor existeix des de fa molt temps en els sistemes operatius de la família UNIX i derivats. Crear, i sobretot manegar, els contenidors era però una tasca només a l'abast dels administradors més especialitzats, reservada per a escenaris molt concrets.

Amb la proliferació de les aplicacions i serveis a Internet, la necessitat de compartir entorns en un mateix ordinador amfitrió ha crescut exponencialment. Fer que aquests entorns estiguessin aïllats els uns dels altres per a poder multiplexar els recursos de l'amfitrió amb seguretat és un requisit obligatori. I fer que l'administració d'aquests entorns de solucions, creades moltes vegades per tercers, fos escalable pels administradors ha originat tor un seguit de conjunts d'eines per simplificar i automatitzar aquesta casuística.

Existeixen moltes solucions basades en l'abstracció dels contenidors i cada cop són més les noves solucions per manegar conjunts cada cop més grans de contenidors no ja en un sistema amfitrió sinó en clústers de servidors. La popularització dels contenidors ha portat finalment a que diferents actors de la indústria posessin en marxa una iniciativa comuna anomenada **Open Containers Initiative[17] (OCI)** per a definir i difondre estàndards en el format dels contenidors entre diferents solucions.

Com a solucions representatives de l'ampli espectre de solucions basades en contenidors en les distribucions GNU/Linux, i per poder fer un recorregut de la seva evolució i com s'han anat adaptant segons les necessitats de cada moment i cada segment d'usuaris, analitzarem:

- **LXC**, que proporciona un primer conjunt d'eines per l'administrador.
- **Docker**, que automatitza el procés de virtualitzar aplicacions i la seva distribució.
- **rkt**, que és component clau del sistema operatiu basat en contenidors **CoreOS.**
- **Singularity**, una solució de contenidors especialitzada en HPC.
- **runV**, una solució que proporciona un kernel per a cada contenidor.
- **shitter**, que finalment retorna a la solució simple de *chroot* a partir d'imatges de contenidors.

Un aspecte interessant d'aquesta revisió és observar com des de l'objectiu inicial d'aïllar i abstreure totalment els contenidors han aparegut solucions que tornen als orígens, proporcionant capacitat de fer la clàssica operació de *chroot* en imatges de contenidors moderns.

Contenidors amb LXC/LXD

Les solucions LXC/LXD són implantacions de contenidors Linux que han estat desenvolupades inicialment per l'empresa Canonical per a la seva distribució Ubuntu[18].

LXC és un conjunt d'eines per a poder utilitzar les característiques del sistema operatiu (*cgroups, namespaces, etc.*) de manera que es poden construir i executar entorns aïllats, els contenidors, a través d'una **API** normalitzada. El component principal sobre el que s'organitzen aquests contenidors és la llibreria *liblxc* que proporciona a les eines de més alt nivell les interfícies necessàries amb el sistema operatiu.

Aquest conjunt inicial d'eines LXC, tot i representar un avanç de cara a poder crear i manegar contenidors Linux, no resultava prou amigable, de manera que

17 https://www.opencontainers.org/
18 https//linuxcontainers.org

els seus desenvolupadors van crear **LXD**. Amb LXD es proporciona una interfície més elaborada per als administradors, de manera que resulta molt més simple usar els contenidors[19]. A més, aquesta nova evolució de la solució permet interaccionar i integrar-se amb altres sistemes de major nivell d'abstracció com **OpenStack[20],** el que permet crear solucions al núvol amb conjunts de contenidors, entre altres.

LXD ha seguit evolucionant de manera que pot crear els seus contenidors però on també es pot arribar a importar i executar de forma indirecta les més recents imatges Docker. De totes maneres, l'objectiu de LXD és proporcionar contenidors per a virtualitzar sistemes operatius complets (excepte el nucli Linux que és compartit amb el sistema amfitrió) a diferencia d'altres solucions con Docker més orientades a aïllar aplicacions i les seves llibreries.

Contenidors amb Docker

La solució de contenidors Docker[21] va ser desenvolupada per l'empresa dotCloud, un proveïdor de serveis al núvol, inicialment per al seu ús intern. A l'any 2013 va ser alliberada[22] al públic com un conjunt d'eines integrades per a manegar contenidors. Des d'aleshores la solució Docker per a contenidors s'ha popularitzat fins al punt de convertir-se per a moltes persones, especialment desenvolupadors, en sinònim del concepte de contenidors.

La solució **Docker** automatitza la creació del contenidor, tot construint el seu sistema d'arxius, configurant la pila TCP/IP al contenidor i afegint un *bridge* per la seva interconnexió, iniciant els processos sol·licitats i executant-los en un entorn aïllat dins de l'amfitrió. A partir de la versió 0.9, Docker deixa d'usar principalment la llibreria *liblxc* de Canonical i la substitueix per la seva pròpia llibreria *libcontainer,* tot i que manté la capacitat de seguir interaccionant amb *liblxc* i altres llibreries de virtualització com *libvirt* o *systemd-nspawn.*

El funcionament de Docker es basa en les anomenades imatges Docker, una combinació de sistemes d'arxius organitzats en capes sobreposades. D'aquesta manera cada capa del sistema d'arxius conté modificacions sobre la capa inferior. L'avantatge és que si diferents contenidors comparteixin un capa més bàsica, p.e. el sistema d'arxius base d'una distribució, només es necessita disposar d'una única còpia a l'amfitrió. Aquesta capa base es combina després amb cadascuna de les diferents capes superiors de cada contenidor per a obtenir els sistemes d'arxius. Altres configuracions del contenidor es descriuen en l'anomenat d*ockerfile* del contenidor.

19 Per a una discussió detallada de LXC/LXD per part del seu desenvolupador principal, veure [Gra2015a]
20 https://www.openstack.org
21 https://www.docker.com
22 Per a veure la primera presentació de Docker al públic per part dels seus creadors, veure [Doc2013a, min. 3:30]

Aquesta tècnica té beneficis pels administradors dels sistemes amfitrions doncs redueix l'espai total en disc alhora que permet actualitzar simultàniament una capa base per a molts contenidors en una sola operació. Per contra, cal tenir en compte, en un entorn STEM[23] que requereixi garantir la reproductibilitat que si alguna de les capes inferiors s'actualitza, podrien variar els resultats computacionals de l'aplicació que s'executi en el sistema de fitxers resultant de sobreposar les altres capes.

L'orientació de Docker és més cap a la virtualització d'aplicacions que no cap a la virtualització de sistemes operatius complets. La idea bàsica és poder instanciar fàcilment molts contenidors que només continguin el mínim sistema d'arxius capaç de poder executar de forma temporal una aplicació o servei (un punt clau de les arquitectures basades en micro-serveis) mentre sigui necessari aquest servei i després apagar el contenidor quan no sigui necessari per a alliberar recursos a l'amfitrió.

Un dels beneficis que proporciona Docker és aquesta facilitat per a desplegar aplicacions en servidors com un bloc, alliberant als administradors de les tasques de configuració dels servidors d'acord amb les necessitats de les aplicacions. Tant els desenvolupadors, que obtenen un temps de desplegament molt reduït, com els administradors, que reben un paquet d'execució complet i el poden instanciar en els servidor sense configuracions particulars, en surten força beneficiats d'aquesta aproximació.

La solució Docker ha evolucionat força i encara ho fa constantment i de forma pública i oberta de manera que donada la popularitat aconseguida en els passats 4 anys és possible que esdevingui l'estàndard *de facto* en la portabilitat d'aplicacions. A més Docker s'ha implantat en altres sistemes operatius, no només de la família UNIX com en el cas de **macOS,** sinó també en **Windows**, la qual cosa permetria, idealment, que un contenidor desenvolupat en Linux es pugui arribar a executar en un equip Windows sense cap modificació.

Contenidors amb CoreOS/rkt

CoreOS[24] és un sistema operatiu, concretament una distribució GNU/Linux, especialitzat en l'ús de contenidors. La distribució només proporciona un sistema mínim per a poder iniciar el sistema i també la funcionalitat necessària per a instanciar i manegar contenidors. Qualsevol altre funcionalitat típica d'una distribució Linux es delega a cada contenidor per a que aquest desplegui la que necessiti dins del seu entorn aïllat individual.

Coreos proporciona el seu propi *runtime* per a contenidors **rkt** que s'encarrega de descarregar les imatges necessàries i instanciar els contenidors i gestionar el seu cicle de vida. Realitza les tasques més privilegiades com ara crear els

23 Per una revisió de l'ús de Docker per a la reproductibilitat, llegir [Boe2015a]
24 https://coreos.com

cgroups i muntar els directoris del sistema d'arxius. Aquest *runtime* substitueix al propi *runtime* de Docker que inicialment usava CoreOS[25].

Aquesta solució és un exemple de com la popularització de l'ús dels contenidors en els darrers anys ha evolucionat des de la primera necessitat de disposar d'eines per l'administrador d'un sistema per usar contenidors, a la necessitat dels desenvolupadors i administradors que despleguen les aplicacions d'abstreure i simplificar aquest processos, a finalment la necessitat d'administrar sistemes complets basats només en contenidors.

Soluciones especialitzades: Singularity

Es pot entendre en aquest punt que aquesta promesa de portabilitat dels contenidors pot tenir un paper important en termes de reproductibilitat computacional en l'àmbit de la ciència i la tecnologia. Malgrat tot, com hem anat veient, les necessitats específiques dels entorns STEM poden ser prou diferents de les necessitats dels desenvolupadors d'aplicacions al núvol o per a administradors d'arquitectures de micro-serveis. Això dona peu a l'aparició de solucions de contenidors especialitzades per a entorns STEM.

Singularity[26] és una solució de contenidors nascuda en la comunitat científica per a la comunitat científica. Com un dels requisits bàsics, especialment de les aplicacions HPC, el *runtime* dels contenidors està més pensat per a poder compartir els recursos del host per a poder usar-los dins del contenidor (p.e. acceleradors de càlcul basats en GPUs).

Algunes de les característiques de Singularity són[27]:

- **imatge basada en un únic fitxer,** no en unió de capes com Docker. Això permet aprofitar el rendiment i la capacitat de sistemes d'arxius distribuïts típics dels entorns HPC.
- **l'usuari del host és el mateix que el que inicia el contenidor** i a dins ja no pot escalar privilegis (això permet crear un contenidor com *root* a l'entorn de desenvolupament però només es podrà usar com un usuari normal en el sistema HPC compartit, sempre que es tinguin privilegis per a migrar la imatge i instanciar el contenidor)
- **munta automàticament el directori *home* de l'usuari**. A diferència de Docker on els contenidors són inicialment només de lectura (es pot però muntar un volum extern) en *Singularity* l'usuari en entrar en el contenidor té disponible el seu directori personal, on segurament té el projecte on ha de treballar.

25 Per veure la justificació del canvi per l'equip de CoreOS, llegir [Pol2014a]
26 http://singularity.lbl.gov/
27 Per a una excel·lent introducció a Singularity i veure com s'instancien contenidors i es creen i renumeren els processos interns quedant aïllats de l'amfitrió, veure [Kut2016a]

- **proporciona un registre central d'imatges**, al igual que altres sistemes, però amb la finalitat de donar suport a la reproductibilitat amb imatges ben predefinides

Contràriament, Singularity no suporta processos *background,* només interactius ni altres casos d'ús més propis de la filosofia Docker, tot i que sí accepta imatges *Docker*, per raons òbvies de interoperabilitat.

Soluciones especialitzades: runV

Les solucions que cerquen interoperabilitat entre els formats de contenidors poden fer ús dels estàndards ja definits per la OCI, entre ells el *runtime* per a contenidors *runC*[28] que proporciona comandaments per a instanciar i manegar contenidors de forma unificada.

La tendència o moda, però, d'aplicar la tècnica de contenidors a qualsevol escenari ha posat de manifest que no sempre es poden complir els requisits demanats en un projecte només aplicant contenidors.

Un cas es dona quan es necessita que cada contenidor tingui el seu propi nucli. Aquí, una solució recent, **runV**[29], permet utilitzar les imatges dels contenidors OCI i usar-los com a base per a crear màquines virtuals. Així, runV proporciona un nou nucli i l'associa a cada contenidor instanciat, executant-lo en hipervisors «tradicionals» com KVM, Virtualbox o Xen. Al tenir nuclis independents s'obtenen funcionalitats intermèdies entre les abstraccions dels contenidors i les màquines virtuals.

L'aplicabilitat d'aquesta solució pot ser d'ús poc general però posa de manifest que la noció de contenidor (fonamentalment a l'estil Docker) ja forma part de la comunitat informàtica i el seu ús s'esten ràpidament a altres àmbits com els STEM.

Soluciones especialitzades: Shifter (NERSC)

Com a exemple final de solucions altament especialitzades en l'àmbit dels contenidors podem trobar **shifter** del NERSC[30]. Aquesta solució permet, entre altres funcionalitats, extreure el sistema d'arxius d'una imatge Docker en el sistema d'arxius del sistema amfitrió i aleshores fer-hi la clàssica operació *chroot* per a entrar dins del sistema d'arxius i usar-ne la seva funcionalitat.

Shifter és una solució orientada als entorns científics de supercomputació però serveix com a mostra de com les solucions basades en contenidors han evolucionat ràpidament buscant cobrir diversitat de necessitats, fins el punt de tancar un cicle de solucions que començava modificant el propi sistema amfitrió per a poder fer una operació *chroot* en un sistema d'arxius local.

28 https://runc.io/
29 https://github.com/hyperhq/runv
30 http://www.nersc.gov/

2.2.5 Comparativa de les solucions alternatives

Diferències entre contenidors i màquines amfitriones modificades

Els contenidors són una abstracció del sistema operatiu que van evolucionar a partir de les primeres tècniques de modificació del sistema amfitrió, p.e. usant l'operació *chroot* per a poder canviar l'arrel del sistema d'arxius d'un procés.

Tot depenent de la implantació del sistema de contenidors que fa cada solució, en qualsevol cas és d'esperar que els contenidors instanciats aprofitin les capacitats del nucli Linux per a:

- proporcionar major aïllament dels processos dels contenidors dels externs, fins al punt de reanomenar els PIDs dels processos.
- proporcionar configuracions addicionals de xarxa, per cada contenidor
- augmentar la seguretat, gestionant els privilegis que un usuari dins del contenidor por tenir
- permetre l'administració i el monitoratge dels recursos assignats i els seus consums pels processos, mitjançant la funcionalitat dels *cgroups*

És important però recordar que en les solucions de contenidors, el nucli és únic tant pel sistema amfitrió com per a tots i cadascun dels contenidors que s'instanciïn.

Diferències entre contenidors i màquines virtuals

Una diferència entre màquines virtuals i contenidors és que aquests darrers, quan estan al mateix amfitrió poden compartir espais de noms, si així es configura. De fet es poden compartir espais de noms entre un contenidor i el seu sistema amfitrió. Això trenca el principi de separació d'espais de noms però dóna flexibilitat a l'hora d'agrupar contenidors en un mateix desplegament a producció.

En canvi les màquines virtuals que s'executen en un mateix sistema amfitrió són a tots els efectes sistemes diferents per la qual cosa la seva comunicació es farà igual que es faria entre dos altres nodes físics en la xarxa.

Els principals inconvenients que presenten les màquines virtuals respecte als contenidors són:

- la sobrecàrrega de recursos en l'amfitrió, especialment pel que fa a l'ocupació en disc de les imatges, sovint molt similars, dels sistemes d'arxius.
- la pèrdua de rendiment que experimenten els processos degut a la emulació instrucció per instrucció que s'ha de fer per part de l'hipervisor que simula el hardware virtual.

Diferències entre solucions de contenidors

Per tal de poder comparar les diferents solucions de contenidors atendrem a uns criteris generals, que sense ser exhaustius poden servir per a ordenar i classificar aquestes solucions de cara a seleccionar una o algunes d'elles per a un escenari determinat.

El primer criteri de comparació seria, lògicament, el **conjunt de característiques tècniques** que oferix cada solució. Com hem vist en la discussió prèvia, cadascun presenta variacions en aquestes[31]. Una comparació de diferents solucions permet elegir la solució que sigui més adient segons els requisits que tingui un projecte a desenvolupar. Tret que hi hagi un requisit concret a complir aquest criteri no seria suficient per triar una solució.

Un altre criteri podria ser la seva **novetat**. Cal tenir en compte, però, que aquestes solucions estan evolucionant constantment i incorporants noves característiques. Això evidentment té els seus avantatges però també els seus inconvenients. Les versions més recents tindrien més i millors característiques mentre que les més antigues estarien més provades encara que amb menys funcionalitat. En aquest sentit i de cara a l'aplicabilitat de projectes STEM la novetat no seria sempre un criteri important per elegir.

Més important sí podria ser, però, la comparació del nivell d'**especialització en funcionalitats pròpies dels entorns STEM**. En aquest cas, *Singularity* seria una bona elecció com a compromís entre una solució molt especialitzada per un entorn determinat i concret, com *shifter*, o una solució més genèrica i flexible com seria l'ús de *Docker*.

En aquest mateix sentit, es pot usar el criteri de comparació de l'**adaptació la solució al cicle de creació, publicació i revisió** del treball. En aquest sentit, Docker semblaria la solució millor posicionada, donada la seva difusió i precisament per estar dissenyada per al transport de contenidors i compartició d'imatges. En canvi, si es prima l'**adaptació als processos d'administració dels servidors compartits**, com en un centre HPC o un en CPD al núvol, caldria pensar en un sistema operatiu dissenyat ad hoc per aquesta finalitat com *CoreOS* o *RancherOS*[32]

Finalment, des del punt de vista de la **comunitat al voltant de la solució** és Docker, de nou, la solució més popular. Això combinat amb una distribució d'àmplia base com Debian pot ser un bon punt de partida per a projectes de reduïda dimensió sigui en entorns STEM individuals o d'equips petits. Si el que es cerca és **suport comercial** la solució d'Ubuntu Server amb LXC/LXD seria una pas més en aquesta evolució.

31 Per veure una comparació detallada de les característiques tècniques, incloent altres sistemes operatius, veure l'entrada en la Wikipedia sobre virtualització a nivell de sistema operatiu https://en.wikipedia.org/wiki/Operating-system-level_virtualization#Implementations

32 http://rancher.com/rancher-os/

3 Aplicació Pràctica

3.1 Disseny d'un Cas Representatiu

3.1.1 Definició d'un cas representatiu

La diversitat, per no dir la potencial complexitat, que poden tenir els entorns STEM fa difícil poder representar en un únic cas tots els escenaris possibles. De fet, un cas complert que reculli totes les casuístiques possibles, en realitat tampoc seria representatiu del gran nombre de situacions generals on només hi participen un sol o ben pocs actors.

Per a definir un cas que pugui ser avaluat en l'aplicació pràctica dissenyarem un escenari on es puguin implantar i validar les principals històries d'usuari en les que potencialment es poden aplicar contenidors Linux. En aquest escenari es simularan els processos que es podrien donar en un típic cicle de producció, publicació i revisió de solucions STEM, tot això emprant principalment les eines més populars i disponibles i no pas les més especialitzades

Un cas d'entorn STEM

En aquest model representatiu cal començar per incorporar un actor amb el rol de **desenvolupador**. Aquest és un dels primes rols que un científic o enginyer ha d'assumir quan decideix que ha de desenvolupar una solució computacional per a la seva feina. Aquest desenvolupador haurà de disposar de la seva estació de treball on instal·lar bé el compilador o intèrpret d'elecció (p.e. Python 3.x), bé l'aplicació especialitzada que ha d'usar (p.e. MATLAB).

A continuació incorporem al model el rol d'**administrador** per tal de poder configurar i instal·lar les aplicacions i llibreries especialitzades en les estacions dels desenvolupadors. No menys important resulta poder proveir també un sistema compartit per als desenvolupadors d'un mateix equip o un «centre de càlcul» dedicat amb una major capacitat de càlcul. També serà feina d'aquest rol proveir d'un repositori de contenidors.

Tot seguit modelem el rol d'**autor**, per a l'enginyer o científic que empaqueta la solució en un contenidor i que a continuació passa a publicar en el repositori proporcionat per l'administrador.

Finalment, un actor en el rol de **revisor**, pot obtenir la imatge del contenidor del repositori i l'instal·larà en el seu sistema independent per a instanciar-lo i executar-lo i així poder avaluar la seva funcionalitat i la reproductibilitat dels resultats publicats.

Aquest cas o model representaria l'escenari inicial d'un equip STEM que comença a aplicar els contenidors com tecnologia de suport al seu cicle de creació i publicació.

3.1.2 Selecció d'una alternativa basada en contenidors

Per a implantar l'aplicació pràctica cal elegir els components entre les diferents alternatives disponibles. Optarem per les opcions més populars i disponibles, seguint la idea de l'equip STEM que s'inicia en l'ús de contenidors. Un cop aquest equip guanyi experiència i es generin necessitats més especialitzades, els administradors d'aquests entorns ja poden incorporar solucions més complexes per a escalar les capacitats de l'entorn.

Sistemes operatius

Com a sistema de treball típic d'una estació Linux elegirem la distribució **Debian**[33] així com la distribució **Ubuntu Server**[34] pels servidors compartits. Són dues solucions d'abast general que disposen d'una àmplia base d'usuaris i, en el cas d' Ubuntu disposen de suport comercial professional per part de l'empresa creadora Canonical. Pel sistemes dels revisors i per a avaluar la interoperabilitat dels contenidors entre plataformes elegirem **Windows 10** i **macOS** a més de les distribucions Linux.

Programari especialitzat

Per a poder avaluar el funcionament de programari especialitzat científic i tècnic en l'aplicació pràctica elegirem **Python** com a llenguatge popular en els darrers temps en la comunitat científica. Una aplicació típica especialitzada en entorns STEM és **MATLAB**. Aquesta eina disposa d'una alternativa de codi obert, **Octave**, que servirà per simular el cas de les proves de compatibilitat del codi desenvolupat.

Solucions de contenidors

Com principal solució de contenidors elegirem **Docker**, fonamentalment per la seva popularitat actual que pot fer que en molts entorns STEM es plantegin la seva conveniència. Addicionalment, i per a representar entorns de servidors compartits mes complexes, elegirem **LXC/LXD** que pot arribar a treballar amb imatges Docker a més de les natives en el seu format propi. Aquesta solució és tècnicament la més natural per a un servidor Ubuntu i per tant més suportada en cas de que requerís assistència comercial per l'equip.

Repositori de contenidors

Pel que fa al repositori, sigui públic o privat, l'elecció més lògica és usar el servei cloud de Docker, **DockerHub**, que no només disposa d'un ampli catàleg d'imatges base sinó que també por proporcionar solucions comercials des de la seva recent presentació de *Docker Enterprise Edition*.

33 https://www.debian.org
34 https://www.ubuntu.com/server

3.1.3 Disseny de l'aplicació pràctica

La aplicació pràctica del cas STEM haurà d'incloure doncs un seguit de components sobre els que poder realitzar i avaluar els diversos processos usant contenidors.

Tot seguit es desglossen els diferents sistemes, components i processos que cal implantar en l'aplicació pràctica i les proves i validacions que cal realitzar per a comprovar l'aplicabilitat dels contenidors.

Estació de treball Debian amb Docker

- Implantar una estació de treball amb el sistema Debian, com a distribució genèrica GNU/Linux.
- Instal·lar el *runtime* Docker per a crear i manegar contenidors

Contenidors Docker per a alternar compiladors

- Crear una o més imatges Docker per a poder alternar els intèrprets Python 2.x i Python 3.x
- Avaluar la utilitat dels contenidors en aquestes històries d'usuari

Contenidors Docker per a alternar llibreries

- Crear una o més imatges Docker per a poder alternar llibreries científiques
- Avaluar la utilitat dels contenidors en aquestes històries d'usuari

Contenidors Docker per a alternar aplicacions

- Crear una o més imatges Docker per a poder alternar les aplicacions comercials MATLAB i open source Octave
- Avaluar la utilitat dels contenidors en aquestes històries d'usuari

Contenidors Docker per a execucions en paral·lel

- Crear més d'una imatge Docker i executar diversos contenidors en paral·lel
- Avaluar la creació de múltiples instàncies i la concurrència dels contenidors en aquestes històries d'usuari

Servidor Ubuntu compartit amb LXC/LXD

- Implantar un servidor dedicat amb el sistema Ubuntu Server, com a exemple de «centre de càlcul»
- Instal·lar el *runtime LXC/LXD* per a crear i manegar contenidors

- Avaluar la creació de contenidors per a la seva execució en servidors en el «centre de càlcul»
- Avaluar la interoperabilitat de LXC i les imatges Docker

Repositori public

- Crear un repositori d'imatges Docker a DockerHub
- Avaluar la creació i publicació de solucions en contenidors

Ús de contenidors Docker en Linux

- Instal·lar el *runtime* Docker per crear i manegar contenidors
- Obtenir un o més contenidors publicats en el repositori
- Avaluar la interoperabilitat de les imatges Docker entre sistemes Linux

Ús de contenidors Docker en Windows

- Instal·lar el *runtime* Docker per crear i manegar contenidors
- Obtenir un o més contenidors publicats en el repositori
- Avaluar la interoperabilitat de les imatges Docker en sistemes Windows

Ús de contenidors Docker en macOS

- Instal·lar el *runtime* Docker per crear i manegar contenidors
- Obtenir un o més contenidors publicats en el repositori
- Avaluar la interoperabilitat de les imatges Docker en sistemes macOS

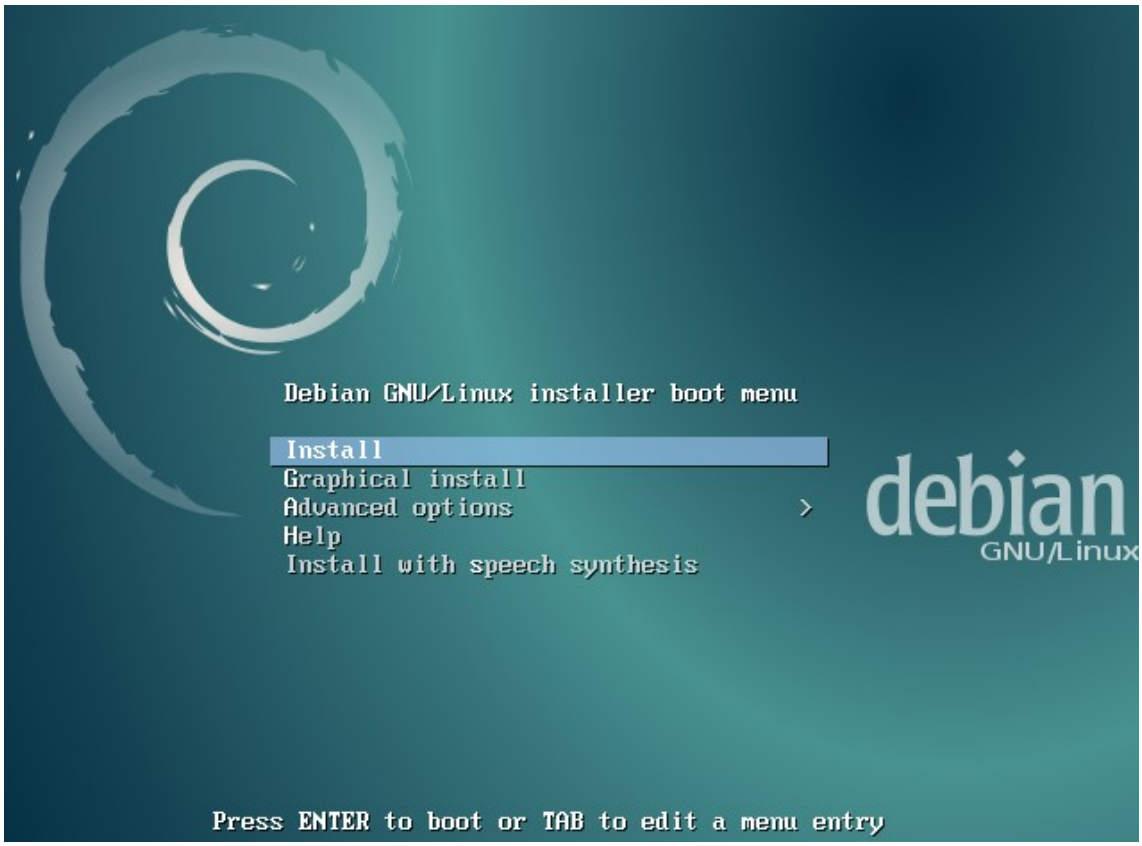
3.2 Implantació del Cas

3.2.1 Estació de treball Debian amb Docker

Instal·lació de Debian

Per a preparar una estació de treball instal·lem la darrera versió disponible de la distribució Debian seguint el procediment estàndard.

Cal tenir en compte que la instal·lació posterior de Docker requerix però treballar amb un sistema operatiu de 64 bits, arquitectura habitual per altra banda en entorns STEM.



Aquesta instal·lació és una tasca comuna per l'administrador de la xarxa. En el nostre cas hem creat una màquina anomenada *STEMstation* amb un usuari *stemdev,* i li hem assignat l'adreça IP 192.168.1.101.

Més característic de l'entorn STEM és la instal·lació en les estacions de treball de les aplicacions pròpies del domini de coneixement del científic o enginyer que l'ha d'usar. Aquestes tasques també son genèriques des del punt de vista de l'administrador. En nostre interès es centra doncs en instal·lar la solució de contenidors Docker en l'estació de treball per a poder usar-los.

Instal·lació de Docker

Un requisit específic per a instal·lar la darrera versió de Docker en un sistema Debian és disposar com a mínim de la versió 3.10 del *kernel* Linux. La darrera versió 8.7.1 de Debian GNU/Linux ja compleix aquest requisit, com podem comprovar connectant-nos a la distribució instal·lada.

```
stemdev@STEMstation:~$ uname -a
Linux STEMstation 3.16.0-4-amd64 #1 SMP Debian 3.16.39-1+deb8u2 (2017-
03-07) x86_64 GNU/Linux
```

En el cas de Debian, també, cal afegir el repositori de software de Docker a la llista de repositoris de l'estació per a poder-lo descarregar i instal·lar. Seguint les darreres instruccions del web de Docker[35] instal·lem el *runtime* en la nostra estació de treball.

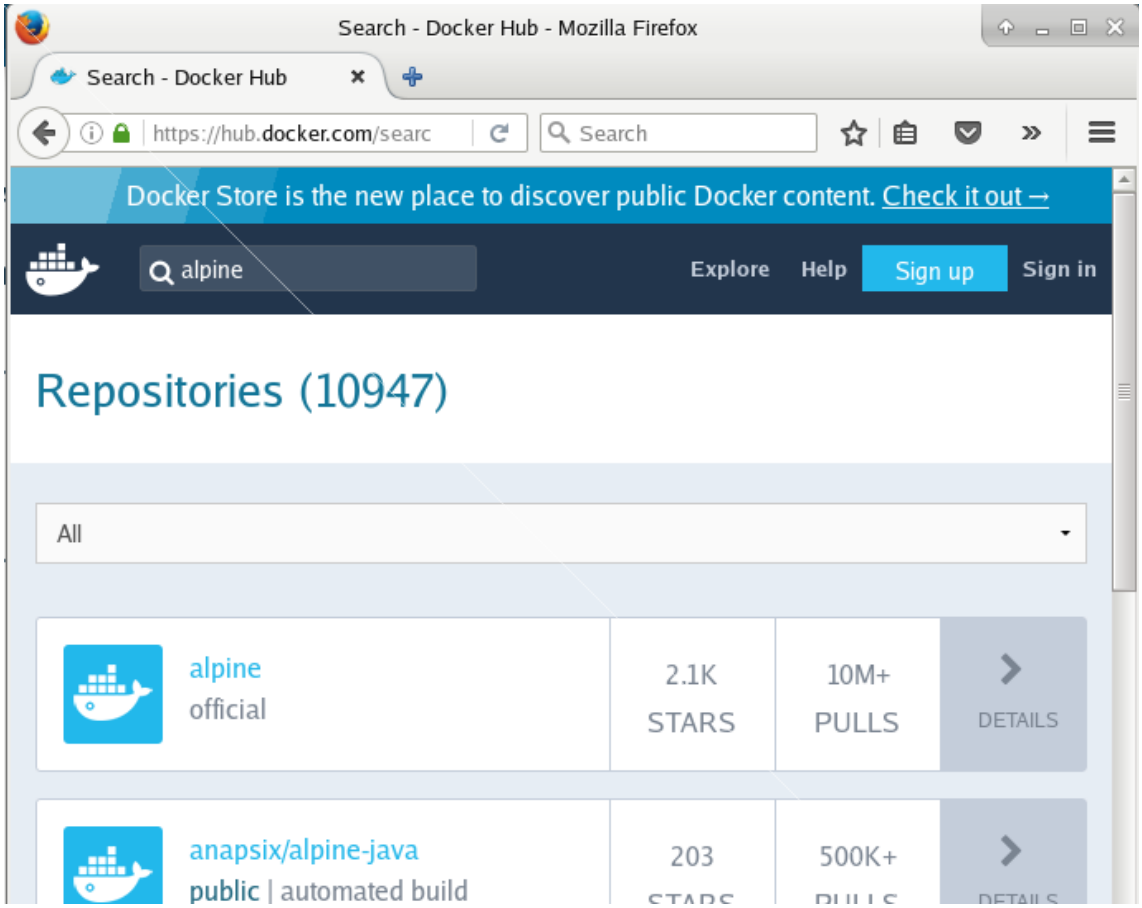
Per a comprovar el seu funcionament, primer habilitem el nou servei *docker* instal·lat i ja podem usar les comandes Docker.

```
stemdev@STEMstation:~$ sudo systemctl enable docker
Synchronizing state for docker.service with sysvinit using update-
rc.d...
Executing /usr/sbin/update-rc.d docker defaults
Executing /usr/sbin/update-rc.d docker enable

stemdev@STEMstation:~$ docker help

Usage:      docker COMMAND

A self-sufficient runtime for containers

Options:
      --config string      Location of client config files (default
"/home/stemdev/.docker")
  -D, --debug              Enable debug mode
      --help               Print usage
  -H, --host list          Daemon socket(s) to connect to (default [])
  -l, --log-level string   Set the logging level ("debug", "info",
"warn", "error", "fatal") (default "info")
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default
"/home/stemdev/.docker/ca.pem")
      --tlscert string     Path to TLS certificate file (default
"/home/stemdev/.docker/cert.pem")
      --tlskey string      Path to TLS key file (default
"/home/stemdev/.docker/key.pem")
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit
```

Docker és una solució de contenidors basada en imatges i la forma més senzilla per poder comprovar la instal·lació i funcionament del *runtime* és descarregar alguna imatge disponible del repositori públic Dockerhub.

35 https://docs.docker.com/engine/installation/linux/debian/

Una imatge molt popular és la del sistema operatiu Alpine Linux[36] optimitzat per a contenidors. Fent una cerca obtenim el nom de la imatge oficial, és a dir mantinguda per l'equip de Docker, que resulta ser simplement *alpine*.



Podem fer una operació *pull* en l'estació per a obtenir la imatge del contenidor.

```
stemdev@STEMstation:~$ sudo docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
627beaf3eaaf: Pull complete
Digest:
sha256:58e1a1bb75db1b5a24a462dd5e2915277ea06438c3f105138f97eb53149673c
4
Status: Downloaded newer image for alpine:latest
```

Un cop descarregada podem comprovar que efectivament la tenim a la llista d'imatges disponibles per a Docker.

```
stemdev@STEMstation:~$ sudo docker images
REPOSITORY      TAG       IMAGE ID            CREATED             SIZE
alpine          latest    4a415e366388        8 weeks ago         3.99 MB
```

36 https://alpinelinux.org/

Amb aquesta imatge disponible ja podem instanciar un contenidor i executar-hi una comanda. Per exemple, si executem la comanda *ls -al* per llistar el directori actual obtenim la sortida des de la perspectiva de dins del contenidor:

```
stemdev@STEMstation:~$ sudo docker run alpine ls -l
total 52
drwxr-xr-x    2 root     root          4096 Mar  3 11:20 bin
drwxr-xr-x    5 root     root           340 Apr 29 15:17 dev
drwxr-xr-x   14 root     root          4096 Apr 29 15:17 etc
drwxr-xr-x    2 root     root          4096 Mar  3 11:20 home
drwxr-xr-x    5 root     root          4096 Mar  3 11:20 lib
drwxr-xr-x    5 root     root          4096 Mar  3 11:20 media
drwxr-xr-x    2 root     root          4096 Mar  3 11:20 mnt
dr-xr-xr-x  156 root     root             0 Apr 29 15:17 proc
drwx------    2 root     root          4096 Mar  3 11:20 root
drwxr-xr-x    2 root     root          4096 Mar  3 11:20 run
drwxr-xr-x    2 root     root          4096 Mar  3 11:20 sbin
drwxr-xr-x    2 root     root          4096 Mar  3 11:20 srv
dr-xr-xr-x   13 root     root             0 Apr 29 15:17 sys
drwxrwxrwt    2 root     root          4096 Mar  3 11:20 tmp
drwxr-xr-x    7 root     root          4096 Mar  3 11:20 usr
drwxr-xr-x   12 root     root          4096 Mar  3 11:20 var
```

que és diferent de la llista del directori actual on s'ha executat el contenidor:

```
stemdev@STEMstation:~$ ls -l
total 32
drwxr-xr-x 2 stemdev stemdev 4096 Apr 29 15:12 Desktop
drwxr-xr-x 2 stemdev stemdev 4096 Apr 29 15:12 Documents
drwxr-xr-x 2 stemdev stemdev 4096 Apr 29 15:12 Downloads
drwxr-xr-x 2 stemdev stemdev 4096 Apr 29 15:12 Music
drwxr-xr-x 2 stemdev stemdev 4096 Apr 29 15:12 Pictures
drwxr-xr-x 2 stemdev stemdev 4096 Apr 29 15:12 Public
drwxr-xr-x 2 stemdev stemdev 4096 Apr 29 15:12 Templates
drwxr-xr-x 2 stemdev stemdev 4096 Apr 29 15:12 Videos
```

Docker ha creat un primer contenidor a partir de la imatge *alpine*, que conté un sistema operatiu minimalista, i ha executat la comanda *ls* en el directori arrel a l'interior del contenidor i que és diferent efectivament del directori actual on s'ha iniciat la comanda *docker run*.

Més interessant és comparar la sortida de les comandes *ps,* fora i dins del contenidor, on es pot veure com el contenidor només mostra un procés, aïllat de tal manera que aparentment és el procés amb PID igual a 1 dins del contenidor:

```
stemdev@STEMstation:~$ ps
  PID TTY          TIME CMD
 3238 pts/0    00:00:00 bash
13013 pts/0    00:00:00 ps

stemdev@STEMstation:~$ sudo docker run alpine ps
PID   USER     TIME   COMMAND
    1 root       0:00 ps
```

No menys important és observar també com l'usuari a dins del contenidor apareix com usuari *root* amb tots els seus privilegis. Serà cada solució de contenidors, en aquest cas Docker, qui té la responsabilitat de garantir que això no serà un problema de seguretat pel sistema amfitrió en el cas de que, per alguna configuració incorrecta o per un error de programació, aquest usuari *root* intern al contenidor pogués accedir fora del contenidor amb tots els seus privilegis.

També es pot veure com Docker ha creat un nou dispositiu de xarxa *docker0* en el sistema amfitrió, tot proveint un *bridge* entre la seva xarxa i la del contenidor, i com ha assignat una adreça IP interna al contenidor (172.17.0.2). D'aquesta manera la connexió a Internet ja està disponible per al contenidor, com es pot veure en la comanda *ping* final.

```
stemdev@STEMstation:~$ ip addr list
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
    link/ether 00:0c:29:00:b4:df brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.101/24 brd 192.168.1.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe00:b4df/64 scope link
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue
state DOWN group default
    link/ether 02:42:23:fd:dc:02 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 scope global docker0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:23ff:fefd:dc02/64 scope link
       valid_lft forever preferred_lft forever

stemdev@STEMstation:~$ sudo docker run alpine ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:11:00:02
          inet addr:172.17.0.2  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::42:acff:fe11:2/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:1 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:90 (90.0 B)  TX bytes:90 (90.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

```
stemdev@STEMstation:~$ sudo docker run alpine ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=57 time=17.357 ms
64 bytes from 8.8.8.8: seq=1 ttl=57 time=16.698 ms
```

Fent una comanda *traceroute* des de dins el contenidor podem observar com els paquets passen pel brigde *docker0* (**172.17.0.1**) per després sortir per la connexió a Internet de l'amfitrió (**192.168.1.1**).

```
stemdev@STEMstation:~$ sudo docker run alpine traceroute -n 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 46 byte packets
 1  172.17.0.1  0.009 ms  0.002 ms  0.001 ms
 2  192.168.1.1  0.637 ms  0.266 ms  0.814 ms
 3  10.80.0.51  12.074 ms  10.588 ms  13.743 ms
 4  10.80.4.201  9.389 ms  7.891 ms  7.910 ms
 5  *  172.29.86.1  8.301 ms  7.848 ms
 6  *  *  *
 7  212.166.147.22  17.694 ms  17.923 ms  19.971 ms
 8  72.14.236.181  16.814 ms  72.14.234.173  15.672 ms  72.14.236.181
15.909 ms
 9  216.239.50.219  15.733 ms  216.239.48.251  16.180 ms
216.239.48.105  17.799 ms
10  8.8.8.8  16.761 ms  17.732 ms  17.902 ms
```

Un benefici immediat, doncs, de les solucions de contenidors com Docker és la simplificació en el proveïment i configuració de la infraestructura de xarxa local, que és automàtica. Un altre benefici és el de proporcionar un repositori d'imatges oficials ja preparades i disponibles per a la comunitat.

3.2.2 Contenidors Docker per a alternar compiladors

Un cop es disposa de la solució de contenidors instal·lada en l'estació de treball es pot usar per a les necessitats pròpies del desenvolupador STEM. Una de les necessitats dels desenvolupadors identificades és la de poder alternar el compilador o intèrpret del llenguatge de desenvolupament sense alterar l'entorn ja configurat (DEV01).

Així, per exemple, l'estació de treball STEMstation creada ja disposa dels intèrprets python2 i python3:

```
stemdev@STEMstation:~$ python2
Python 2.7.9 (default, Jun 29 2016, 13:08:31)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
stemdev@STEMstation:~$ python3
Python 3.4.2 (default, Oct  8 2014, 10:45:20)
[GCC 4.9.1] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

però si es vol treballar amb versions més actuals (o més antigues) que les que proporciona la distribució als seus repositoris oficials caldria desinstal·lar

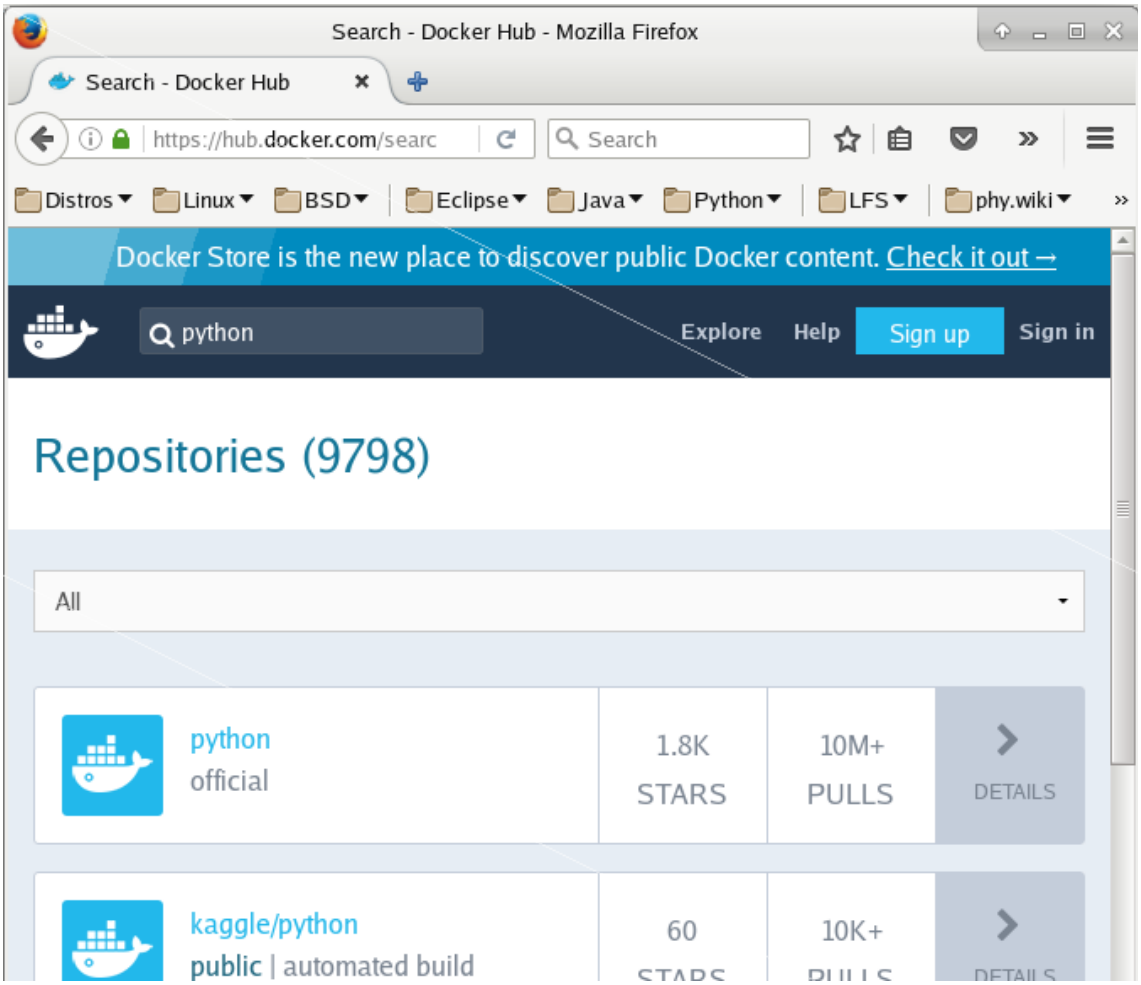
aquestes i instal·lar-ne les altres, i això potser fent-ho des de repositoris mantinguts per tercers amb els possibles conflictes que podria suposar alternar i provar vàries versions diferents per tornar després a la configuració inicial de treball.

Cerca i descàrrega d'imatges de contenidors

Amb la solució de Docker ja instal·lada es poden buscar imatges de contenidors amb *python* disponible usant la comanda *docker search*:

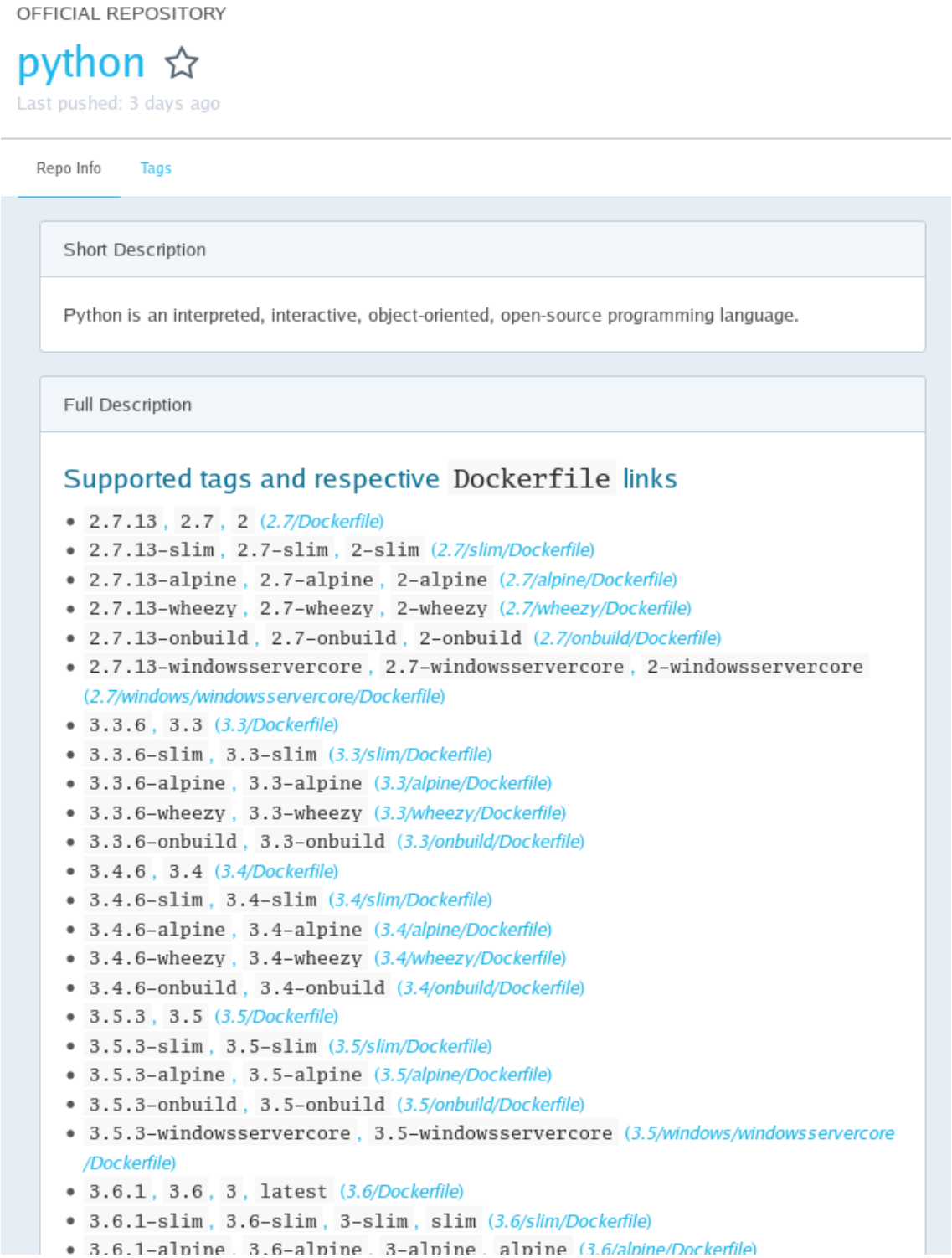
```
stemdev@STEMstation:~$ docker search -f is-official=true python
NAME      DESCRIPTION                                     STARS
OFFICIAL   AUTOMATED
python    Python is an interpreted, interactive, obj...   1814
[OK]
```

És més, si es cerquen contenidors[37] amb python a la web Dockerhub i s'accedeix al detall de la imatge oficial es pot veure que estan disponibles diferents versions, basades en diferents sistemes operatius base.



37 El que es cerca als repositoris són imatges de contenidors. Per abús de llenguatge és habitual parlar de cercar i descarregar contenidors, per referir-se a les seves imatges.

Aquestes versions estan identificades en Docker amb les anomenades etiquetes de la imatge (*tags*):



OFFICIAL REPOSITORY

python ☆

Last pushed: 3 days ago

Repo Info Tags

Short Description

Python is an interpreted, interactive, object-oriented, open-source programming language.

Full Description

Supported tags and respective `Dockerfile` links

- `2.7.13`, `2.7`, `2` (*2.7/Dockerfile*)
- `2.7.13-slim`, `2.7-slim`, `2-slim` (*2.7/slim/Dockerfile*)
- `2.7.13-alpine`, `2.7-alpine`, `2-alpine` (*2.7/alpine/Dockerfile*)
- `2.7.13-wheezy`, `2.7-wheezy`, `2-wheezy` (*2.7/wheezy/Dockerfile*)
- `2.7.13-onbuild`, `2.7-onbuild`, `2-onbuild` (*2.7/onbuild/Dockerfile*)
- `2.7.13-windowsservercore`, `2.7-windowsservercore`, `2-windowsservercore` (*2.7/windows/windowsservercore/Dockerfile*)
- `3.3.6`, `3.3` (*3.3/Dockerfile*)
- `3.3.6-slim`, `3.3-slim` (*3.3/slim/Dockerfile*)
- `3.3.6-alpine`, `3.3-alpine` (*3.3/alpine/Dockerfile*)
- `3.3.6-wheezy`, `3.3-wheezy` (*3.3/wheezy/Dockerfile*)
- `3.3.6-onbuild`, `3.3-onbuild` (*3.3/onbuild/Dockerfile*)
- `3.4.6`, `3.4` (*3.4/Dockerfile*)
- `3.4.6-slim`, `3.4-slim` (*3.4/slim/Dockerfile*)
- `3.4.6-alpine`, `3.4-alpine` (*3.4/alpine/Dockerfile*)
- `3.4.6-wheezy`, `3.4-wheezy` (*3.4/wheezy/Dockerfile*)
- `3.4.6-onbuild`, `3.4-onbuild` (*3.4/onbuild/Dockerfile*)
- `3.5.3`, `3.5` (*3.5/Dockerfile*)
- `3.5.3-slim`, `3.5-slim` (*3.5/slim/Dockerfile*)
- `3.5.3-alpine`, `3.5-alpine` (*3.5/alpine/Dockerfile*)
- `3.5.3-onbuild`, `3.5-onbuild` (*3.5/onbuild/Dockerfile*)
- `3.5.3-windowsservercore`, `3.5-windowsservercore` (*3.5/windows/windowsservercore/Dockerfile*)
- `3.6.1`, `3.6`, `3`, `latest` (*3.6/Dockerfile*)
- `3.6.1-slim`, `3.6-slim`, `3-slim`, `slim` (*3.6/slim/Dockerfile*)
- `3.6.1-alpine`, `3.6-alpine`, `3-alpine`, `alpine` (*3.6/alpine/Dockerfile*)

L'etiqueta *latest* correspon a la versió més actual de la imatge i és la que es descarrega per defecte si no s'especifica cap.

Alternança d'intèrprets

Si es descarrega el contenidor amb la darrera versió, es comprova que es pot usar ara la versió 3.6.1 de Python dins del contenidor sense alterar la versió de Python instal·lada a l'estació des de la distribució:

```
stemdev@STEMstation:~$ docker pull python
Using default tag: latest
latest: Pulling from library/python
cd0a524342ef: Already exists
e39c3ffe4133: Pull complete
85334a7c2001: Pull complete
4c546d9d6a84: Pull complete
a2eb12d55dae: Pull complete
6eb4f90c094f: Pull complete
c9fff2c9f747: Pull complete
Digest:
sha256:13e05c74cb3515f1c1bfe105f6b97e60299ed8ee4e1d7db7ed0a862080d5401
8
Status: Downloaded newer image for python:latest
```

Iniciem un nou contenidor a partir de la imatge *python* i executem la comanda *bash* per a poder treballar de forma interactiva dins del contenidor. Això proporciona el *prompt* de sistema operatiu des d'on es pot comprovar que la versió de python dins del contenidor és la 3.6.1 diferent a la 3.4.2 que proporciona la distribució:

```
stemdev@STEMstation:~$ python3
Python 3.4.2 (default, Oct  8 2014, 10:45:20)
[GCC 4.9.1] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>

stemdev@STEMstation:~$ docker run -ti python bash
root@e974d2503397:/# python
Python 3.6.1 (default, Apr 26 2017, 20:20:42)
[GCC 4.9.2] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Aquest contenidor només conté la versió 3.6.1 de python. Si es vol ara usar la darrera versió de Python2 es pot descarregar la seva imatge corresponent:

```
stemdev@STEMstation:~$ docker pull python:2.7.13
2.7.13: Pulling from library/python
cd0a524342ef: Already exists
e39c3ffe4133: Already exists
85334a7c2001: Already exists
4c546d9d6a84: Already exists
a2eb12d55dae: Already exists
9f5ece3ce497: Pull complete
846446f24ee3: Pull complete
Digest:
sha256:c26eb79cb8334ecd8f63b65edfae0ab11912052ef4b6a45131abcc0224e9989
2
Status: Downloaded newer image for python:2.7.13
```

Ara es pot comprovar en l'estació quines imatges s'han descarregat fins ara i estan disponibles amb la comanda *docker images*:

```
stemdev@STEMstation:~$ docker images
REPOSITORY     TAG       IMAGE ID            CREATED             SIZE
python         latest    a0d32d529a0a        3 days ago          689 MB
python         2.7.13    eb03485f8ec8        3 days ago          678 MB
alpine         latest    4a415e366388        8 weeks ago         3.99 MB
```

Si ara s'instancia un contenidor, a partir de la imatge pyhton i etiqueta 2.7.13, es pot comprovar que aquest nou contenidor conté només la darrera versió de Pyhton2 però no conté cap versió de Python3:

```
stemdev@STEMstation:~$ docker run -ti python:2.7.13 bash

root@dc567cbeed97:/# python
Python 2.7.13 (default, Apr 26 2017, 19:32:55)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> ^D

root@dc567cbeed97:/# python3
bash: python3: command not found
```

Sortint del contenidor i retornant al *prompt* de l'estació de treball es comprova que el sistema original no s'ha alterat i que es pot seguir treballant amb les versions inicials que proporcionava la distribució:

```
root@dc567cbeed97:/# exit
exit

stemdev@STEMstation:~$ python2
Python 2.7.9 (default, Jun 29 2016, 13:08:31)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
stemdev@STEMstation:~$ python3
Python 3.4.2 (default, Oct  8 2014, 10:45:20)
[GCC 4.9.1] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Si bé en el cas de Python el propi llenguatge ja proporciona mecanismes similars com els *virtual environments*[38], per a altres entorns amb la mateixa problemàtica com per exemple el llenguatge Ruby, la tecnologia dels contenidors unifica d'alguna manera la solució a aquesta problemàtica. Així només caldria repetir el procés tot comprovant si hi ha imatges disponibles per a l'entorn de treball desitjat:

```
stemdev@STEMstation:~$ docker search -f is-official=true ruby
NAME      DESCRIPTION                                     STARS
OFFICIAL   AUTOMATED
ruby      Ruby is a dynamic, reflective, object-orie...   995
[OK]
```

38 https://docs.python.org/3/tutorial/venv.html

```
jruby     JRuby (http://www.jruby.org) is an impleme...   55
[OK]
```

Alternança de compiladors

El mateix cas s'aplica també als compiladors, que no cal que estiguin instal·lats en l'estació de treball. Per exemple, per usar la darrera versió del compilador de C, *gcc,* de la col·lecció GNU de compiladors només cal cercar i descarregar la imatge corresponent i instanciar el contenidor adient. Dins del contenidor es podrà crear i compilar el programa en C i executar-lo dins del contenidor:

```
stemdev@STEMstation:~$ gcc
-bash: gcc: command not found

stemdev@STEMstation:~$ docker pull gcc
Using default tag: latest
latest: Pulling from library/gcc
cd0a524342ef: Already exists
e39c3ffe4133: Already exists
85334a7c2001: Already exists
4c546d9d6a84: Already exists
ca5955eb1884: Pull complete
7f1a2329e0fe: Pull complete
339248b9ecfc: Pull complete
26933fdd4fb6: Pull complete
Digest:
sha256:c562497492b2ec337df4e042102ab75242552abb971936bd9b8ff8347404959
b
Status: Downloaded newer image for gcc:latest

stemdev@STEMstation:~$ docker run -ti gcc bash
root@d3d0e2a562e2:/# cd
root@d3d0e2a562e2:~# pwd
/root

root@d3d0e2a562e2:~# cat > prova.c <<EOF
> #include <stdio.h>
> void main()
> {
>   printf("Hello, World!\n");
> }
> EOF

root@d3d0e2a562e2:~# more prova.c
#include <stdio.h>
void main()
{
  printf("Hello, World!\n");
}

root@d3d0e2a562e2:~# gcc -o prova prova.c

root@d3d0e2a562e2:~# ./prova
Hello, World!
```

Un detall important és que tota aquesta alternança d'eines es pot fer sense la intervenció de l'administrador del sistema. Només cal que l'usuari

desenvolupador (en aquest cas l'usuari *stemdev* que s'ha creat a la STEMstation) tingui els privilegis suficients en el seu equip de treball (en aquest cas concret se l'ha afegit als grups d'usuaris *docker* de manera que pot executar una comanda *docker* amb els privilegis necessaris).

Muntatge de volums als contenidors

Els contenidors de la solució Docker, però, són efímers en el sentit que un cop acaba la seva execució el seu contingut desapareix. Així, mentre s'està executant el contenidor gcc anterior podem accedir-hi, p.e. iniciant una altra comanda *bash* des d'un altre terminal a l'amfitrió i hi trobarem el contingut creat (p.e. el programa prova.c anterior).

Per a comprovar quins contenidors hi ha en marxa a l'estació de treball es pot usar la comanda *docker ps* i veure que amb la imatge gcc s'ha instanciat el contenidor amb ID **d3d0e2a562e2** (Docker a més li ha donat automàticament un nom més «humà», com sleepy_clarke per a usar-lo en altres comandes).

```
stemdev@STEMstation:~$ docker ps
CONTAINER ID IMAGE COMMAND CREATED         STATUS        PORTS NAMES
d3d0e2a562e2 gcc   "bash"  32 minutes ago  Up 32 minutes       sleepy_clarke
```

Contra aquest contenidor es pot executar una altra comanda *bash* per accedir interactivament al directori de l'usuari *root* on s'ha creat el programa *prova.c* i on es pot executar la compilació prèvia. Usant la comanda *docker exec* tot referenciant el contenidor pels primers caràcters del seu ID es comprova:

```
stemdev@STEMstation:~$ docker exec -ti d3d0 bash

root@d3d0e2a562e2:/# cd /root
root@d3d0e2a562e2:~# ls -al
total 36
drwx------  4 root root 4096 Apr 30 08:31 .
drwxr-xr-x 56 root root 4096 Apr 30 08:01 ..
-rw-------  1 root root   16 Apr 30 08:31 .bash_history
-rw-r--r--  1 root root  570 Jan 31  2010 .bashrc
drwx------  2 root root 4096 Apr 25 09:05 .gnupg
-rw-r--r--  1 root root  140 Nov 19  2007 .profile
-rwxr-xr-x  1 root root 6536 Apr 30 08:01 prova
-rw-r--r--  1 root root   65 Apr 30 08:01 prova.c
root@d3d0e2a562e2:~# ./prova
Hello, World!
```

Si ara, però, es surt del contenidor i es torna a instanciar un nou contenidor a partir de la mateixa imatge tots els canvis fets dins el contenidor anterior s'hauran perdut:

```
root@d3d0e2a562e2:~# exit
exit

stemdev@STEMstation:~$ docker run -ti gcc bash
root@09d62bf22a6e:/# cd /root
root@09d62bf22a6e:~# ls -al
```

```
total 20
drwx------  4 root root 4096 Apr 25 09:05 .
drwxr-xr-x 54 root root 4096 Apr 30 08:44 ..
-rw-r--r--  1 root root  570 Jan 31  2010 .bashrc
drwx------  2 root root 4096 Apr 25 09:05 .gnupg
-rw-r--r--  1 root root  140 Nov 19  2007 .profile
```

Aquesta característica dels contenidors immutables és una funcionalitat de Docker per disseny. Per tal de poder mantenir canvis fets dins del contenidor de forma persistent, Docker proporciona la possibilitat de muntar directoris del sistema amfitrió dins del contenidor (aprofitant la capacitat de reanomenar els punts de muntatge anàlogament a com es fa amb els IDs dels processos).

Així, el desenvolupador pot treballar normalment en la seva estació i en crear el contenidor amb una versió alternativa del compilador, muntar el directori de treball dins el contenidor i executar programes o usar el codi font dins aquest altre entorn aïllat. Per exemple, l'usuari *stemdev* ha creat en el directori *~/Development/project1* dos programes en python: un que funciona en la versió 2 del llenguatge i l'altre en la versió 3:

```
stemdev@STEMstation:~/Development/project1$ ls -al
total 16
drwxr-xr-x 2 stemdev stemdev 4096 Apr 30 10:52 .
drwxr-xr-x 3 stemdev stemdev 4096 Apr 29 21:18 ..
-rw-r--r-- 1 stemdev stemdev   33 Apr 30 10:53 test2.py
-rw-r--r-- 1 stemdev stemdev   34 Apr 30 10:53 test3.py

stemdev@STEMstation:~/Development/project1$ more test2.py
for i in range(5):
    print i

stemdev@STEMstation:~/Development/project1$ more test3.py
for i in range(5):
    print(i)
```

En el nou contenidor que es creï a partir de la imatge python no existeix encara el directori /opt/project1:

```
stemdev@STEMstation:~/Development/project1$ docker run python \
ls -al /opt/project1
ls: cannot access /opt/project1: No such file or directory
```

Però si en la comanda de creació del contenidor s'afegeix el paràmetre -v, per a muntar aquest directori de treball en un directori dins del contenidor, aleshores:

```
stemdev@STEMstation:~/Development/project1$ docker run \
-v ~/Development/project1:/opt/project1 python ls -al /opt/project1
total 16
drwxr-xr-x 2 1000 1000 4096 Apr 30 08:52 .
drwxr-xr-x 3 root root 4096 Apr 30 09:00 ..
-rw-r--r-- 1 1000 1000   33 Apr 30 08:53 test2.py
-rw-r--r-- 1 1000 1000   34 Apr 30 08:53 test3.py
```

es pot veure un directori /opt/project1 amb els fitxers del sistema amfitrió i ara ja es poden fer proves dins del contenidor tot usant directament el codi original de l'estació de treball:

```
stemdev@STEMstation:~/Development/project1$ docker run \
-v ~/Development/project1:/opt/project1 python \
python /opt/project1/test2.py
  File "/opt/project1/test2.py", line 2
    print i
          ^
SyntaxError: Missing parentheses in call to 'print'

stemdev@STEMstation:~/Development/project1$ docker run \
-v ~/Development/project1:/opt/project1 python \
python /opt/project1/test3.py
0
1
2
3
4
```

En aquest cas la imatge *python* conté la darrera versió de Python 3, que falla en executar la prova creada amb Python2 (*print* és una funció a partir de Python 3 i per tant necessita parèntesis) però sí que pot executar la prova amb Python 3. Alternativament, el mateix programa Pyhton2 funciona perfectament en un contenidor basat en la imatge python amb etiqueta 2.7.13:

```
stemdev@STEMstation:~/Development/project1$ docker run \
-v ~/Development/project1:/opt/project1 python:2.7.13 \
python /opt/project1/test2.py
0
1
2
3
4
```

Aplicació a les necessitats STEM

Es pot veure doncs que amb la tecnologia dels contenidors Docker es pot resoldre la necessitat que apareix en entorns STEM d'alternar versions de compiladors i intèrprets tot usant els projectes desenvolupats a l'estació de treball però sense alterar de cap manera l'entorn original (per a no permetre als processos del contenidor alterar el sistema d'arxius de l'amfitrió els volums Docker es poden muntar amb permisos de només lectura[39] i no escriptura com en l'exemple anterior).

Combinant totes aquestes tècniques, la solució Docker permet complir amb els criteris d'acceptació establerts per a la història d'usuari **DEV01** definida en l'estudi de necessitats.

39 Veure totes les opcions de la comanda *docker run* en la direcció https://docs.docker.com/engine/reference/commandline/run/

3.2.3 Contenidors Docker per a alternar llibreries

Els contenidors Docker, com els anteriors creats a partir de les imatges python o gcc, tenen l'inconvenient de que no són persistents. Excepte en els volums muntants sobre directoris del sistema amfitrió (amb les consideracions de seguretat que això implica) tots els canvis que es facin en el contenidor, p.e. per actualitzar les llibreries del sistema es perdran al tancar-lo.

Suposem que necessitem desenvolupar una aplicació que utilitzi la llibreria científica *GNU Scientific Library*[40] tot usant el compilador gcc. Partint del contenidor bàsic de la imatge gcc anterior s'hi poden fer les modificacions necessàries per compilar un programa que l'usuari *stemdev* tingui en el seu sistema:

```
stemdev@STEMstation:~$ mkdir Development/project2
stemdev@STEMstation:~$ cd Development/project2
stemdev@STEMstation:~/Development/project2$ cat > test-gsl.c <<EOF
#include <stdio.h>
#include <gsl/gsl_sf_bessel.h>

int
main (void)
{
  double x = 5.0;
  double y = gsl_sf_bessel_J0 (x);
  printf ("J0(%g) = %.18e\n", x, y);
  return 0;
}
EOF

stemdev@STEMstation:~/Development/project2$ docker run -ti \
-v $(pwd):/root/project2 gcc
root@50a10f961ef0:/# cd /root/project2
root@50a10f961ef0:~/project2# ls -al
total 12
drwxr-xr-x 2 1000 1000 4096 May  6 14:38 .
drwx------ 5 root root 4096 May  6 14:37 ..
-rw-r--r-- 1 1000 1000  173 May  6 14:11 test-gsl.c
```

Un cop dins del contenidor s'hi pot instal·lar la llibreria des del repositori de la distribució i així poder compilar i executar el programa:

```
root@50a10f961ef0:~/project2# apt-get update
Get:1 http://security.debian.org jessie/updates InRelease [63.1 kB]
Get:2 http://security.debian.org jessie/updates/main amd64 Packages

<<< ... part de la sortida suprimida ... >>

Fetched 9933 kB in 3s (3036 kB/s)
Reading package lists... Done

root@50a10f961ef0:~/project2# apt-get -y install libgsl0-dev
Reading package lists... Done
Building dependency tree
```

40 https://www.gnu.org/software/gsl

```
Reading state information... Done
The following extra packages will be installed:
  libgsl0ldbl
Suggested packages:
  gsl-ref-psdoc gsl-doc-pdf gsl-doc-info gsl-ref-html
The following NEW packages will be installed:
  libgsl0-dev libgsl0ldbl
0 upgraded, 2 newly installed, 0 to remove and 8 not upgraded.

<<< ... part de la sortida suprimida ... >>

Setting up libgsl0ldbl (1.16+dfsg-2) ...
Setting up libgsl0-dev (1.16+dfsg-2) ...
Processing triggers for libc-bin (2.19-18+deb8u7) ...

root@50a10f961ef0:~/project2# gcc -I/usr/include/gsl -c test-gsl.c

root@50a10f961ef0:~/project2# gcc test-gsl.o -lgsl -lgslcblas -o test-
gsl

root@50a10f961ef0:~/project2# ./test-gsl
J0(5) = -1.775967713143382642e-01
```

Persistència dels canvis fets dins del contenidor

Si ara es surt i torna a entrar al contenidor resulta que el programa compilat sí que es troba disponible (perquè s'ha guardat en un volum muntat sobre sistema amfitrió), però que el programa ara no s'executa perquè dins del nou contenidor no està instal·lada la llibreria:

```
stemdev@STEMstation:~/Development/project2$ docker run -ti \
-v $(pwd):/root/project2 gcc
root@7a889a7f66ea:/# cd /root/project2
root@7a889a7f66ea:~/project2# ls -al
total 32
drwxr-xr-x 2 1000 1000 4096 May  6 14:43 .
drwx------ 5 root root 4096 May  6 14:50 ..
-rwxr-xr-x 1 root root 7216 May  6 14:43 test-gsl
-rw-r--r-- 1 1000 1000  173 May  6 14:11 test-gsl.c
-rw-r--r-- 1 root root 1656 May  6 14:42 test-gsl.o
root@7a889a7f66ea:~/project2# ./test-gsl
./test-gsl: error while loading shared libraries: libgsl.so.0: cannot
open shared object file: No such file or directory
```

Caldria, per tant, tornar a repetit totes les passes anteriors per a poder usar la llibreria amb l'executable anterior dins de cada nou contenidor.

Per a resoldre aquesta problemàtica de no persistència Docker proporciona un mecanisme per a crear noves imatges a partir de les imatges disponibles. Usant la comanda *docker build,* que llegeix un fitxer d'instruccions anomenat Dockerfile, es crea una nova imatge que sí conté les modificacions fetes.

Així, si es crea un fitxer Dockerfile amb les instruccions d'instal·lació de les llibreries a partir de la imatge base es podran crear després nous contenidors a

partir d'aquesta nova imatge que, aquests sí, ara ja tinguin les llibreries necessàries:

```
stemdev@STEMstation:~/Development/project2$ cat > Dockerfile <<EOF
FROM gcc
RUN apt-get update && apt-get -y install libgsl0-dev
EOF
```

Ara executant la comanda *docker build* es pot crear la nova imatge a partir del Dockerfile[41] present en el directori actual tot donant-li, p.e. el nom *gcc-gsl*:

```
stemdev@STEMstation:~/Development/project2$ docker build -t gcc-gsl .
Sending build context to Docker daemon 22.02 kB
Step 1/2 : FROM gcc
 ---> ecc92f2be252
Step 2/2 : RUN apt-get update && apt-get -y install libgsl0-dev
 ---> Running in 6ca7d33a4da1
Get:1 http://security.debian.org jessie/updates InRelease [63.1 kB]
Ign http://deb.debian.org jessie InRelease
Get:2 http://deb.debian.org jessie-updates InRelease [145 kB]

<<< ... part de la sortida suprimida ... >>

 ---> 04be6c0624eb
Removing intermediate container 6ca7d33a4da1
Successfully built 04be6c0624eb
```

Es pot veure com Docker va executant una a una les instruccions del fitxer Dockerfile per a acabar construint una nova imatge que ara conté les llibreries necessàries. Això es pot comprovar amb la comanda *docker images* i també executant el programa creat anteriorment però en aquest nou contenidor:

```
stemdev@STEMstation:~/Development/project2$ docker images
REPOSITORY     TAG       IMAGE ID            CREATED             SIZE
gcc-gsl        latest    04be6c0624eb        7 seconds ago       1.52 GB
python         latest    a0d32d529a0a        9 days ago          689 MB
python         2.7.13    eb03485f8ec8        9 days ago          678 MB
gcc            latest    ecc92f2be252        11 days ago         1.49 GB
alpine         latest    4a415e366388        2 months ago        3.99 MB

stemdev@STEMstation:~/Development/project2$ docker run -ti \
-v $(pwd):/root/project2 gcc-gsl

root@6d4fb2bc1079:/# /root/project2/test-gsl
J0(5) = -1.77596771314338264e-01

root@6d4fb2bc1079:/# gsl-config --version
1.16

root@6d4fb2bc1079:/# find /usr/lib -name libgsl.so*
/usr/lib/libgsl.so
/usr/lib/libgsl.so.0.17.0
/usr/lib/libgsl.so.0
```

41 La clàusula FROM indica a partir de quina imatge es construeix la nova, i la clàusula RUN executa comandes dins del contenidor. Per revisar totes les possibilitats del Dockerfile veure https://docs.docker.com/engine/reference/builder/

Noves imatges basades en capes

El sistema de creació d'imatges de Docker permet alterar només una de les múltiples capes de canvis que configuren una imatge final. Així, si ara es vol crear una imatge de contenidor que contingui la darrera versió de la llibreria *gsl* i no la subministrada per la distribució de nou es pot crear un altre arxiu Dockerfile més elaborat amb les instruccions específiques per a la descàrrega del codi font de la llibreria i seva compilació i posterior instal·lació en la imatge:

```
stemdev@STEMstation:~/Development/project2$ cat > Dockerfile <<EOF
FROM gcc
RUN wget ftp://ftp.gnu.org/gnu/gsl/gsl-2.3.tar.gz && \
    tar xvf gsl-2.3.tar.gz                        && \
    cd gsl-2.3                                    && \
    ./configure --prefix=/usr --disable-static    && \
    make                                          && \
    make install                                  && \
    ln -s libgsl.so.19.3.0 /usr/lib/libgsl.so.0
EOF
```

Executant la comanda *docker build* amb el nou Dockerfile es genera una imatge amb una llibreria *gsl* més actual i personalitzada:

```
stemdev@STEMstation:~/Development/project2$ docker build -t gcc-gsl-
2.3 .
Sending build context to Docker daemon 271.4 MB

Step 1/2 : FROM gcc
 ---> ecc92f2be252
Step 2/2 : RUN wget ftp://ftp.gnu.org/gnu/gsl/gsl-2.3.tar.gz &&
tar xvf gsl-2.3.tar.gz && cd gsl-2.3 && ./configure --prefix=/usr
--disable-static && make && make install && ln -s libgsl.so.19.3.0
/usr/lib/libgsl.so.0
 ---> Running in 029e8eccb197
converted 'ftp://ftp.gnu.org/gnu/gsl/gsl-2.3.tar.gz' (ANSI_X3.4-1968)
-> 'ftp://ftp.gnu.org/gnu/gsl/gsl-2.3.tar.gz' (UTF-8)

<<< ... part de la sortida suprimida ... >>

/usr/bin/install -c -m 644 gsl_math.h gsl_pow_int.h gsl_nan.h
gsl_machine.h gsl_mode.h gsl_precision.h gsl_types.h gsl_version.h
gsl_minmax.h gsl_inline.h '/usr/include/gsl'
make[2]: Leaving directory '/gsl-2.3'
make[1]: Leaving directory '/gsl-2.3'
 ---> e7f5d27051b5
Removing intermediate container 029e8eccb197
Successfully built 6eecb93fd8a6

stemdev@STEMstation:~/Development/project2$ docker images
REPOSITORY     TAG      IMAGE ID           CREATED             SIZE
gcc-gsl-2.3    latest   6eecb93fd8a6       2 minutes ago       1.59 GB
gcc-gsl        latest   04be6c0624eb       About an hour ago   1.52 GB
python         latest   a0d32d529a0a       9 days ago          689 MB
python         2.7.13   eb03485f8ec8       9 days ago          678 MB
gcc            latest   ecc92f2be252       11 days ago         1.49 GB
alpine         latest   4a415e366388       2 months ago        3.99 MB
```

Aquesta nova imatge permet iniciar un contenidor que executa el programa anterior per enllaçant dinàmicament amb la nova llibreria personalitzada sense haver de recompilar el programa prèviament generat:

```
stemdev@STEMstation:~/Development/project2$ docker run -ti \
-v $(pwd):/root/project2 gcc-gsl-2.3 bash

root@5caa9d2a2dfe:/# /root/project2/test-gsl
J0(5) = -1.77596771314338264e-01

root@5caa9d2a2dfe:/# gsl-config --version
2.3

root@5caa9d2a2dfe:/# find /usr/lib -name libgsl.so*
/usr/lib/libgsl.so
/usr/lib/libgsl.so.19
/usr/lib/libgsl.so.19.3.0
/usr/lib/libgsl.so.0
```

Cache d'imatges i múltiples capes

El sistema d'imatges de Docket manté una memòria cache per tal de no haver de reconstruir les capes que no han canviat. En la instrucció anterior es pot veure com la imatge base gcc (**6eecb93fd8a6**) de la clàusula FROM no es reconstrueix. I si es torna a demanar la creació de la imatge sense canvis en el Dockerfile no es tornen tampoc a executar les instruccions de la clàusula RUN:

```
sstemdev@STEMstation:~/Development/project2$ docker build -t gcc-gsl-
2.3 .
Sending build context to Docker daemon 271.8 MB
Step 1/2 : FROM gcc
 --->  ecc92f2be252
Step 2/2 : RUN wget ftp://ftp.gnu.org/gnu/gsl/gsl-2.3.tar.gz &&
tar xvf gsl-2.3.tar.gz && cd gsl-2.3 && ./configure --prefix=/usr
--disable-static && make && make install && ln -s libgsl.so.19.3.0
/usr/lib/libgsl.so.0
 ---> Using cache
 --->  6eecb93fd8a6
Successfully built  6eecb93fd8a6
```

Aquest sistema de capes permet niuar els contenidors i els seus Dockerfile corresponents. Si per exemple es vol afegir l'editor *GNU nano*[42] per a poder fer canvis als programes en desenvolupament dins del contenidor es pot partir de la darrera imatge a la clàusula FROM i només afegir una nova capa amb el canvi puntual:

```
stemdev@STEMstation:~/Development/project2$ cat > Dockerfile <<EOF
FROM gcc-gsl-2.3
RUN apt-get update && apt-get install -y nano
EOF

stemdev@STEMstation:~/Development/project2$ docker build \
-t gcc-stemdev .
Sending build context to Docker daemon 271.8 MB
```

42 https://www.nano-editor.org

```
Step 1/2 : FROM gcc-gsl-2.3
 ---> 6eecb93fd8a6
Step 2/2 : RUN apt-get update && apt-get install -y nano

<<< ... part de la sortida suprimida ... >>

Setting up nano (2.2.6-3) ...
update-alternatives: using /bin/nano to provide /usr/bin/editor
(editor) in auto mode
update-alternatives: using /bin/nano to provide /usr/bin/pico (pico)
in auto mode
 ---> 7df891118e20
Removing intermediate container e6c56b2275b1
Successfully built 7df891118e20

stemdev@STEMstation:~/Development/project2$ docker images
REPOSITORY     TAG      IMAGE ID            CREATED             SIZE
gcc-stemdev    latest   7df891118e20        3 minutes ago       1.61 GB
gcc-gsl-2.3    latest   6eecb93fd8a6        51 minutes ago      1.59 GB
gcc-gsl        latest   04be6c0624eb        14 hours ago        1.52 GB
python         latest   a0d32d529a0a        10 days ago          689 MB
python         2.7.13   eb03485f8ec8        10 days ago          678 MB
gcc            latest   ecc92f2be252        11 days ago         1.49 GB
alpine         latest   4a415e366388        2 months ago        3.99 MB
```

Aplicació a les necessitats STEM

Amb la tecnologia Docker i la utilització dels Dockerfiles per a construir a mida noves imatges de contenidors es pot resoldre la necessitat STEM de poder instal·lar i alternar les versions de les llibreries científiques necessàries per un programa (que en aquest cas es comprova que ni cal recompilar) i/o avaluar el seu funcionament amb unes llibreries més actualitzades o alternatives.

La diferència fonamental en aquest darrer cas respecte l'anterior és que aquí el programa executable està fixat i són les llibreries del sistema (del contenidor en aquest cas) les que es canvien sense alterar l'executable disponible, mentre que en l'anterior cas el que estava fixat era el codi del programa (disponible pel muntatge d'un directori al sistema amfitrió) però l'executable era el que variava (fos un compilador o intèrpret).

En aquest 'exemple es comprova doncs que els criteris d'acceptació la història d'usuari **DEV02** es poden satisfer amb aquesta solució de contenidors.

3.2.4 Contenidors Docker per a alternar aplicacions

La flexibilitat oferta pel sistema de contenidor permet que el desenvolupador pugui mantenir entorns de treball alternatius força complexos. Això permet disposar tant de l'estació de treball «oficial», instal·lada i mantinguda per l'administrador de l'equip, i a la vegada disposar en contenidors locals de les aplicacions alternatives que es desitgi. Per exemple, si en l'estació de treball ja està instal·lada l'aplicació comercial MATLAB però es vol valorar o usar l'aplicació alternativa GNU Octave, de codi lliure i gratuïta, aquesta es pot mantenir en un contenidor a part.

Així, en primer lloc es pot crear una nova imatge, amb l'aplicació Octave instal·lada, i configurar-la per a acceptar connexions SSH[43] de l'usuari *stemdev*:

```
stemdev@STEMstation:~/Development/project4$ cat > Dockerfile <<EOF
FROM debian:latest

RUN apt-get update && \
    apt-get update -y && \
    apt-get install -y octave && \
    apt-get install -y ssh && \
    mkdir /var/run/sshd

RUN useradd stemdev -m -U -s /bin/bash && \
    echo "stemdev:stemdev" | chpasswd

EXPOSE 22
CMD ["/usr/sbin/sshd", "-D"]
EOF

stemdev@STEMstation:~/Development/project4$ \
docker build -t stem-octave .
Sending build context to Docker daemon 3.072 kB
Step 1/5 : FROM debian:latest
 ---> 054abe38b1e6
Step 2/5 : RUN apt-get update && apt-get update -y && apt-get install
-y octave && apt-get install -y ssh && mkdir /var/run/sshd
 ---> Running in 5b9ddb9bd40b
Ign http://deb.debian.org jessie InReleaseSending build context to
Docker daemon 3.072 kB
Step 1/5 : FROM debian:latest
 ---> 054abe38b1e6
Step 2/5 : RUN apt-get update &&     apt-get update -y &&     apt-get
install -y octave &&     apt-get install -y ssh &&     service ssh
start &&     useradd stemdev -m -U -s /bin/bash
 ---> Running in d633b3983898
Get:1 http://security.debian.org jessie/updates InRelease [63.1 kB]

<<< ... part de la sortida suprimida ... >>

Step 4/5 : EXPOSE 22
 ---> Running in f4a6c8e94dbd
 ---> 623a3d6e046c
Removing intermediate container f4a6c8e94dbd
Step 5/5 : CMD /usr/sbin/sshd -D
 ---> Running in 064c41779bd4
 ---> 85123cde19a4
Removing intermediate container 064c41779bd4
Successfully built 85123cde19a4
```

Ara aquest contenidor es pot iniciar en background i fer connexions per usar Octave des de l'entorn gràfic de l'estació:

```
stemdev@STEMstation:~/Development/project4$ docker run --name octave
-v $(pwd):/home/stemdev -d stem-octave
52d13f64ba7ea470be0185d484344b3cd4caf34d0b2cb10f9da88dc4660cc338
```

43 En un entorn real les configuracions de seguretat serien evidentment molt més estrictes que en aquest exemple.

```
stemdev@STEMstation:~/Development/project4$ docker ps
CONTAINER ID  IMAGE        COMMAND               CREATED
STATUS              PORTS               NAMES
52d13f64ba7e  stem-octave  "/usr/sbin/sshd -D"   5 seconds ago
Up 4 seconds        22/tcp              octave
```

Aquest contenidor, que està actiu en *background* i esperant connexions SSH, ara es pot usar per a executar Octave des de l'entorn gràfic X11 de l'estació de treball. Després de consultar l'adreça IP, es pot establir una connexió SSH i iniciar la interfície gràfica de Octave:

```
stemdev@STEMstation:~/Development/project4$ docker exec octave ip addr
list
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
166: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UP group default
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.2/16 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:acff:fe11:2/64 scope link
       valid_lft forever preferred_lft forever

stemdev@STEMstation:~/Development/project4$ matlab &
[2] 13319
stemdev@STEMstation:~/Development/project4$ MATLAB is selecting
SOFTWARE OPENGL rendering.

stemdev@STEMstation:~/Development/project4$ ssh 172.17.0.2
stemdev@172.17.0.2's password:

The programs included with the Debian GNU/Linux system are free
software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun May 21 11:56:58 2017 from 172.17.0.1

stemdev@41fad24fd892:~$ octave --force-gui
```

En la imatge següent es mostra l'execució simultània de l'aplicació comercial *MATLAB* i de codi obert *Octave*. La primera s'ha iniciat en l'estació de treball amb la comanda *matlab*. La segona s'ha iniciat dins del contenidor *octave* anterior, després de connectar-s'hi per SSH amb redirecció del protocol X11 de manera que ambdues aplicacions presenten la seva interfície gràfica simultàniament:

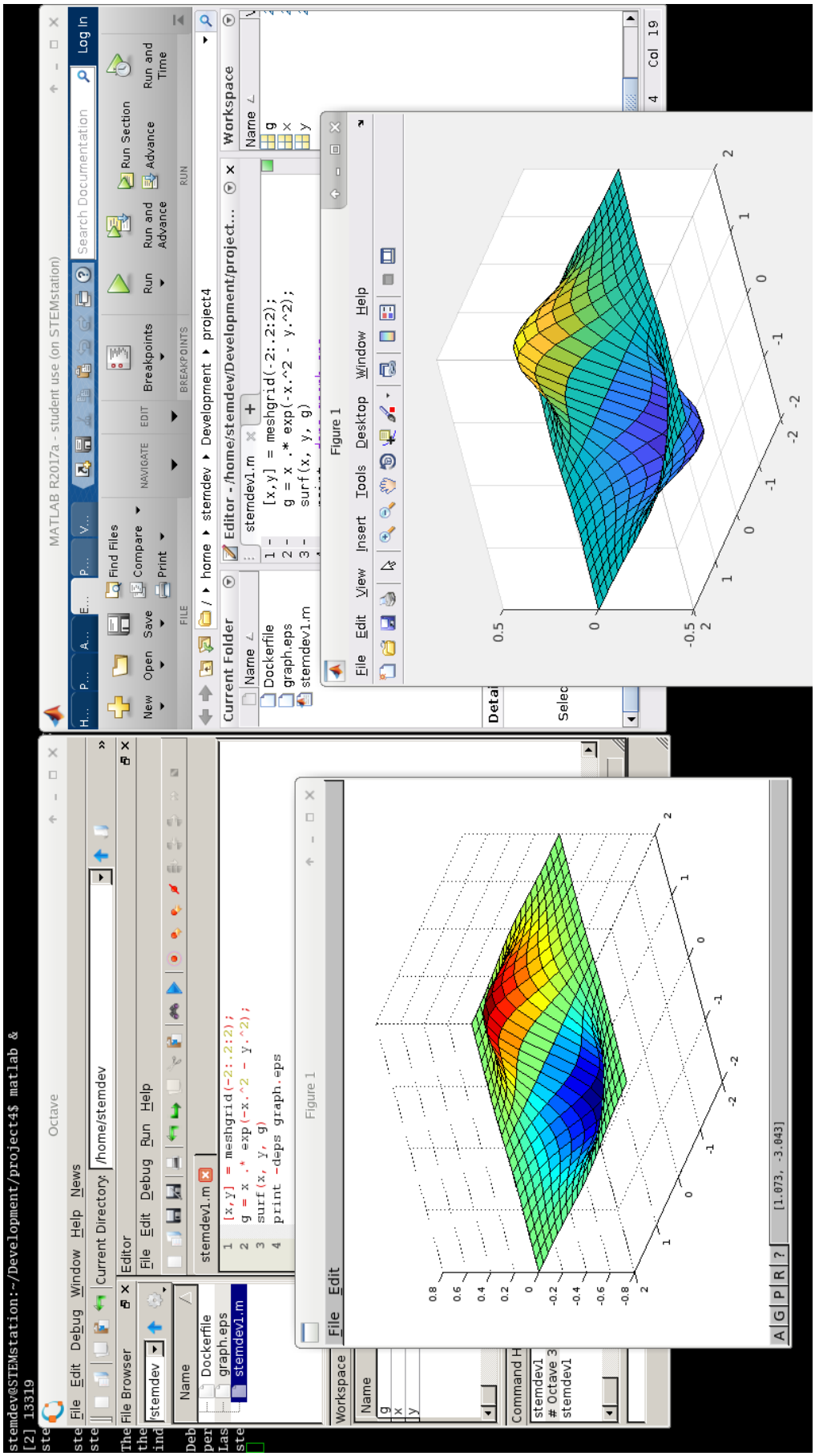


Figura 3: MATLAB i Octave (en un contenidor) executant simultàniament el mateix codi font

Aplicació a les necessitats STEM

En aquest darrer exemple comprovem la complexitat que poden tenir les solucions obtingudes aplicant la tecnologia de contenidors Docker. En aquest exemple s'executen simultàniament dues aplicacions especialitzades sense cap interferència entre elles, doncs una corre en l'estació de forma nativa i l'altre en un contenidor aïllat, i que a la vegada estan usant el mateix codi font en un directori de l'estació i presentant de forma interactiva la seva sortida gràfica. Es comprova per tant que els criteris d'acceptació la història d'usuari **DEV03** es poden satisfer amb la tècnica de contenidors.

3.2.5 Contenidors Docker per a execucions en paral·lel

Els contenidors vistos fins ara s'han executat de forma puntual per a necessitats especifiques. La solució Docker permet iniciar els contenidors en mode desacoblat de manera que no bloquegin el procés que els inicia i per tant poden arribar a córrer en paral·lel amb altres contenidors que s'iniciïn mentre encara s'executen els anteriors.

Per a comprovar aquesta funcionalitat preparem un programa *parallel.c (*que es pot compilar usant el contenidor *gcc* i executar després en un contenidor sense compilador instal·lat) :

```
stemdev@STEMstation:~/Development/project3$ cat > parallel.c <<EOF
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <netinet/in.h>
#include <net/if.h>
#include <arpa/inet.h>
#include <time.h>

#include <sys/sysinfo.h>

void up_time()
{
struct sysinfo info;
sysinfo(&info);
printf("%ld \t", info.uptime);
}

void main(int argc, char *argv[])
{
 long id=(long) argv[1];

 char hostname[1024];
 hostname[1023] = '\0';
 gethostname(hostname, 1023);
 up_time();
 printf("C0%s: starting container %s ", id, hostname);
```

```
 sleep(1);

 int fd;
 struct ifreq ifr;
 fd = socket(AF_INET, SOCK_DGRAM, 0);
 ifr.ifr_addr.sa_family = AF_INET;
 strncpy(ifr.ifr_name, "eth0", IFNAMSIZ-1);
 ioctl(fd, SIOCGIFADDR, &ifr);
 close(fd);
 printf("with IP %s\n", inet_ntoa((
                      (struct sockaddr_in*)&ifr.ifr_addr)->sin_addr));

 srand(id);
 int s=1+rand() % 10;
 up_time();
 printf("C0%s: ready to sleep for %d seconds...\n", id, s);
 sleep(s);

 up_time();
 printf("C0%s: ending container.\n", id);
}
EOF
```

A continuació crearem un *script bash* que arrenqui en paral·lel més d'un contenidor, amb noms C0x i basat en el sistema *debian,* usant sempre el mateix executable disponible en un directori del sistema amfitrió:

```
stemdev@STEMstation:~/Development/project3$ cat > parallel.sh <<EOF
#/bin/bash
for i in {1..5}
do
   echo "run container $i"
   docker run -d -v $(pwd):/root --name C0$i debian /root/parallel $i
done
EOF
```

Si ara s'executa l'script *parallel.sh*, aquest crearà 5 contenidors, en cadascun dels qual s'executarà el programa *parallel*:

```
stemdev@STEMstation:~/Development/project3$ bash parallel.sh
run container 1
69f100c69fc82d7079e196455bf5bb1e904bef74f4be86070ad5f11f14b6f528
run container 2
b1b71989ca6c7c76375e644af8ed91447cbb5f81e6c76832a831d2caa249fa58
run container 3
4e9e45cfa338ad11d4f59dcb0ca34d474f2578a7c2a5d142a4e58f9873a09f1d
run container 4
ca41d66be8560573349a01db3958e23e30373a4d772fa44accbec8ad8f0f793f
run container 5
d7b76ef9f02a2e122ece238838977e38c9b817e6739ed66132a6a0b4085ced4f

stemdev@STEMstation:~/Development/project3$ docker ps
CONTAINER ID  IMAGE COMMAND            CREATED       STATUS       PORTS NAMES
d7b76ef9f02a  gcc   "/root/parallel 5" 4 seconds ago Up 3 seconds       C05
ca41d66be856  gcc   "/root/parallel 4" 4 seconds ago Up 4 seconds       C04
4e9e45cfa338  gcc   "/root/parallel 3" 5 seconds ago Up 4 seconds       C03
b1b71989ca6c  gcc   "/root/parallel 2" 5 seconds ago Up 4 seconds       C02
69f100c69fc8  gcc   "/root/parallel 1" 5 seconds ago Up 5 seconds       C01
```

Aquest programa usarà el paràmetre que rep per a mostrar en quin contenidor s'executa. La sortida de cada contenidor anirà quedant enregistrada per Docker de forma independent per a cada contenidor i es pot recuperar amb la comanda *docker logs*:

```
stemdev@STEMstation:~/Development/project3$ docker logs C01
27210       C01: starting container 51cbbbbcd81d with IP 172.17.0.3
27211       C01: ready to sleep for 8 seconds...
27219       C01: ending container.
```

Si ara es recupera el contingut de totes les sortides anteriors i s'ordenen pel moment de la seva execució es pot veure com els contenidors s'han creat i executat de forma concurrent, a més de comprovar que cada contenidor ha rebut una adreça IP diferent de forma automàtica dins de la xarxa interna dels contenidors:

```
stemdev@STEMstation:~/Development/project3$ for i in {1..5}; \
do docker logs C0$i; done | sort
27210       C01: starting container 51cbbbbcd81d with IP 172.17.0.3
27210       C02: starting container 784a4e6fdf53 with IP 172.17.0.4
27210       C03: starting container 1ecf01866cb4 with IP 172.17.0.5
27211       C01: ready to sleep for 8 seconds...
27211       C02: ready to sleep for 2 seconds...
27211       C03: ready to sleep for 7 seconds...
27211       C04: starting container 09c66e2b0bb1 with IP 172.17.0.6
27211       C05: starting container f3ac477c2bc6 with IP 172.17.0.7
27212       C04: ready to sleep for 2 seconds...
27212       C05: ready to sleep for 3 seconds...
27213       C02: ending container.
27214       C04: ending container.
27215       C05: ending container.
27218       C03: ending container.
27219       C01: ending container.
```

Aplicació a les necessitats STEM

Amb aquest exemple es pot veure com amb la solució de contenidors, un desenvolupador STEM pot simular en el seu propi sistema amfitrió múltiples sistemes virtuals independents, tot això sense necessitat d'haver de crear, instal·lar i configurar els sistemes operatius i xarxa en cada un d'ells com es faria en una solució de màquines virtuals.

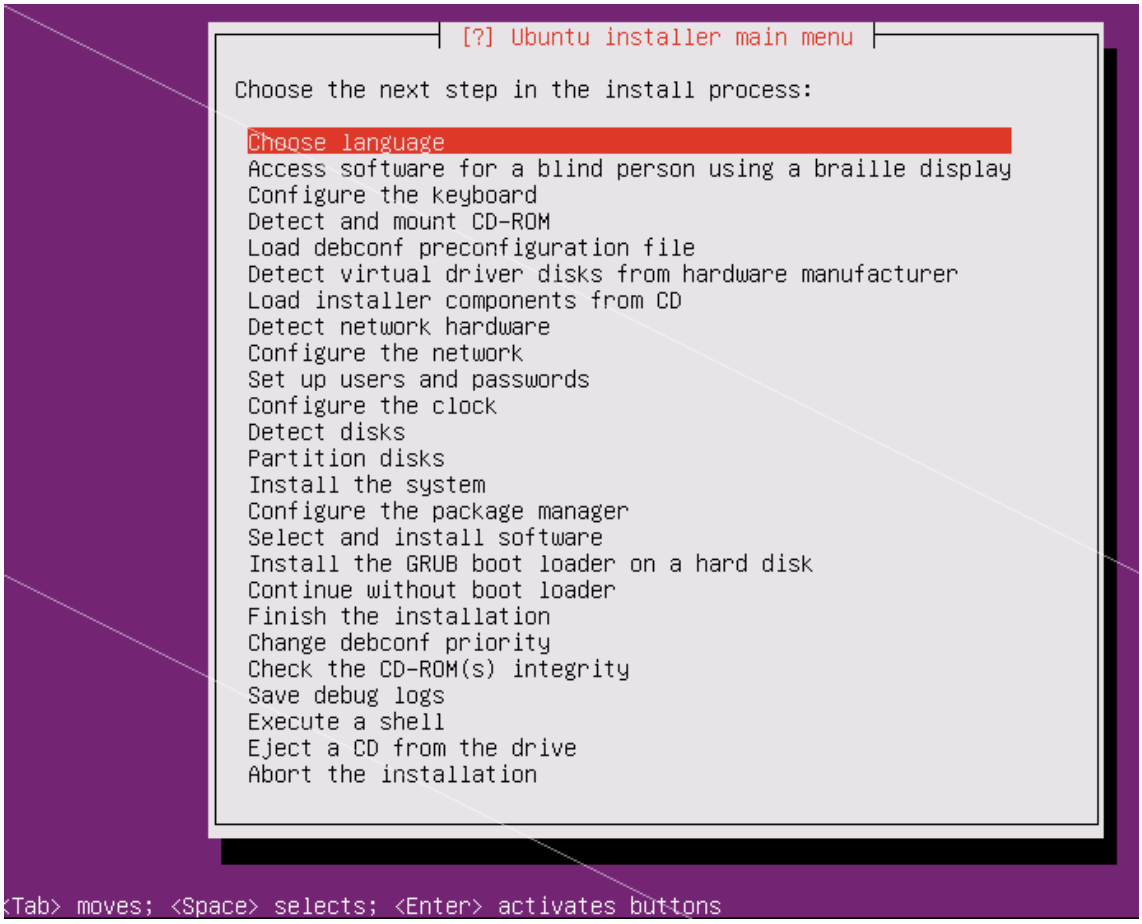
A més el rendiment d'aquests processos, a l'executar-se com processos ordinaris del sistema amfitrió, és gairebé el rendiment natiu amb la única sobrecàrrega de la gestió dels espais de noms, números de processos o els volums muntats que proveeix automàticament el *runtime* dels contenidors i de la indirecció de la xarxa local construida.

Aquest 'exemple verifica que els criteris d'acceptació la història d'usuari **DEV04** també es poden satisfer amb una solució de contenidors.

3.2.6 Servidor Ubuntu compartit amb LXC/LXD

Instal·lació de Ubuntu Server

Per a preparar un servidor compartit que simuli el tipus de recursos disponibles per l'equip en un «centre de càlcul» instal·lem la darrera versió de la distribució Ubuntu Server, seguint el procediment d'instal·lació d'expert que ens permet configurar al detall la instal·lació.



Aquesta instal·lació també és una tasca comuna per a un administrador de la xarxa i inclouria la configuració de tots els recursos compartits que el servidor hagi d'oferir a la xarxa. Pel cas pràctic configurem un servidor anomenat *STEMserver* en l'adreça IP 192.168.1.110 i creem l'usuari *stemadm* per a poder treballar-hi.

En elegir la versió més recent d'Ubuntu Server per al servidor automàticament ja es té instal·lada la solució de contenidors LXD de Canonical (que usa internament LXC), per la qual cosa ja es poden usar contenidors LXC en el servidor i avaluar la seva interoperabilitat amb les imatges Docker.

Configuració de LXC/LXD

Un cop instal·lat el servidor STEMserver podem connectar-nos des de l'estació STEMstation per a utilitzar els contenidors:

```
stemdev@STEMstation:~$ ssh stemadm@STEMserver
stemadm@stemserver's password:

Welcome to Ubuntu 17.04 (GNU/Linux 4.10.0-19-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

Last login: Mon May  1 08:09:20 2017 from 192.168.1.101
```

La primera tasca és comprovar la configuració correcta de LXC:

```
stemadmSTEMserver:~$ lxc info
If this is your first time using LXD, you should also run: lxd init
To start your first container, try: lxc launch ubuntu:16.04
config: {}
api_extensions:
- storage_zfs_remove_snapshots
- container_host_shutdown_timeout
- container_syscall_filtering
- auth_pki
- container_last_used_at

<<< ... part de la sortida suprimida ... >>

    D6zytDT4P/UcPUPBv3OC0W6jJioVLfJHnUcplL3eLzZphjhbuDy3bbHK4XgZqHIZ
    f9eJizHU05gg/gxZamNi4EjF76egozrvzeCjf6vbgtepq3pOp0R4dyIGqcEp4Zcf
    sbZtqm9P89d+GRRiFpRqxXtFr6mSxaYvHvYXUOyOMm3CdZAvs24prA529rGfZWX/
    rHjzcfSlOwYbRA/fF0xYgQ==
    -----END CERTIFICATE-----
  certificate_fingerprint:
daea0c1063f19bdbb89f4483da562d896da3fa866b3a970175826d040306840c
  driver: lxc
  driver_version: 2.0.7
  kernel: Linux
  kernel_architecture: x86_64
  kernel_version: 4.10.0-19-generic
  server: lxd
  server_pid: 1789
  server_version: "2.12"
  storage: ""
  storage_version: ""
stemadm@STEMserver:~$
```

Es pot comprovar que tant LXC (versió **2.0.7**) com LXD (versió **2.12**) ja estan efectivament instal·lats en el sistema i preparats per a ser emprats.

Seguint les indicacions del missatge de la comanda *lxc info*, procedim a iniciar LXD i instanciar un primer contenidor usant la imatge *ubuntu:16.04.*

En primer lloc es configura el servidor LXD que permetrà accedir al *runtime* de contenidors LXC usant la nova interfície LXD més simplificada que la de les versions inicials de LXC:

```
stemadm@STEMserver:~$ lxd init
Do you want to configure a new storage pool (yes/no) [default=yes]?
Name of the new storage pool [default=default]:
Name of the storage backend to use (dir, btrfs, lvm) [default=dir]:
Would you like LXD to be available over the network (yes/no)
[default=no]?
Would you like stale cached images to be updated automatically
(yes/no) [default=yes]?
Would you like to create a new network bridge (yes/no) [default=yes]?
What should the new bridge be called [default=lxdbr0]?
What IPv4 address should be used (CIDR subnet notation, “auto” or
“none”) [default=auto]? 10.1.1.1/24
Would you like LXD to NAT IPv4 traffic on your bridge? [default=yes]?
What IPv6 address should be used (CIDR subnet notation, “auto” or
“none”) [default=auto]? none
LXD has been successfully configured.
```

Deixant les opcions de configuració per defecte (excepte el rang de la xarxa pels contenidors i les adreces IPv6 que desactivarem[44]), podem comprovar com es crea un bridge **lxcbr0** per a connectar els nous contenidors a la xarxa de l'amfitrió a través de la IP de xarxa privada **10.1.1.1** especificada .

```
stemadm@STEMserver:~$ ip addr list
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
    link/ether 00:0c:29:82:2b:29 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.110/24 brd 192.168.1.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe82:2b29/64 scope link
       valid_lft forever preferred_lft forever
3: lxdbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UNKNOWN group default qlen 1000
    link/ether b6:6c:ea:78:69:f9 brd ff:ff:ff:ff:ff:ff
    inet 10.1.1.1/24 scope global lxdbr0
       valid_lft forever preferred_lft forever
    inet6 fe80::b46c:eaff:fe78:69f9/64 scope link
       valid_lft forever preferred_lft forever
```

44 No hi ha cap raó especial per elegir aquest rang 10.1.1.1/24 o suprimir les adreces IPv6. La raó de fer-ho és que les sortides en pantalla de les comandes *lxc list* i altres estan formatades com taules que fan difícil la seva presentació en aquest document escrit. Suprimint les adreces IPv6 es redueix la mida de les línies i es fa presentable la sortida en text escrit. Quelcom similar passa amb els noms per defecte dels contenidors que posa LXD de l'estil *driving-mastiff* que no usarem i donarem nom explícit als contenidors.

Creació i ús dels contenidors LXD

Procedim a crear el primer contenidor LXC amb la comanda *lxc launch* que descarregarà la imatge elegida si no està disponible, instanciarà un nou contenidor amb el nom que s'hi indiqui i deixarà el contenidor funcionant en *background*:

```
stemadm@STEMserver:~$ lxc launch ubuntu:16.04 c01
Creating c01
Starting c01
```

Per tal de comprovar que el contenidor està en marxa es pot usar la comanda *lxc list* que presenta una taula dels contenidor actius:

```
stemadm@STEMserver:~$ lxc list
+------+---------+--------------------+------+------------+-----------+
| NAME |  STATE  |        IPV4        | IPV6 |    TYPE    | SNAPSHOTS |
+------+---------+--------------------+------+------------+-----------+
| c01  | RUNNING | 10.1.1.125 (eth0) |      | PERSISTENT | 0         |
+------+---------+--------------------+------+------------+-----------+
```

LXD està més orientat a virtualitzar sistemes operatius sencers, al contrari que Docker que està més orientat a virtualitzar aplicacions, i proporciona alguns mecanismes més sofisticats, com la comanda *lxc shell,* per tal d'iniciar una connexió i sessió completa al contenidor (amb docker s'executa directament cada comanda com ara *bash o* /bin/*sh*). Si s'accedeix al contenidor amb aquesta comanda s'obté un *login* complet i es pot verificar que s'ha iniciat també tot el sistema d'inici *systemd* al contenidor i una comanda *login*:

```
stemadm@STEMserver:~$ lxc shell c01
run-parts: /etc/update-motd.d/98-fsck-at-reboot exited with return
code 1

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

mesg: ttyname failed: Success

root@c01:~# ps -eH
  PID TTY          TIME CMD
  444 ?        00:00:00 login
  477 ?        00:00:00   bash
  500 ?        00:00:00     ps
    1 ?        00:00:00 systemd
   54 ?        00:00:00   systemd-udevd
   56 ?        00:00:00   systemd-journal
  231 ?        00:00:00   dhclient
  316 ?        00:00:00   accounts-daemon
  317 ?        00:00:00   atd
  321 ?        00:00:00   snapd
  325 ?        00:00:00   sshd
```

```
  326 ?        00:00:00   cron
  327 ?        00:00:00   dbus-daemon
  331 ?        00:00:00   systemd-logind
  333 ?        00:00:00   rsyslogd
  344 ?        00:00:00   polkitd
  378 console  00:00:00   agetty
  473 ?        00:00:00   systemd
  474 ?        00:00:00     (sd-pam)
```

En tot cas, des de dins del contenidor aquest processos trobaran reanomenats els seus PIDs, i així el procés *systemd* apareix com PID 1 quan en realitat és un procés natiu del sistema amfitrió. Això es pot comprovar connectant al servidor *STEMserver* i llistant els processos que s'hi executen. Amb la comanda *ps -eH* des del servidor s'observen els PIDs reals dels processos de LXC/LXD (**1785**, **1947**, ...) i també els dels processos interns al contenidor amb els seus PIDs reals (*systemd* és el PID 1971 i no pas el PID 1, i login és 2790 i no pas el 444):

```
1785 ?        00:00:14   lxd
2785 ?        00:00:00     lxd
2790 pts/3    00:00:00       login
2837 pts/3    00:00:00         bash
1912 ?        00:00:00   dnsmasq
1947 ?        00:00:00   lxd
1971 ?        00:00:00     systemd
2096 ?        00:00:00       systemd-udevd
2098 ?        00:00:00       systemd-journal
2373 ?        00:00:00       dhclient
2496 ?        00:00:00       accounts-daemon
2501 ?        00:00:00       atd
2505 ?        00:00:00       snapd
2510 ?        00:00:00       sshd
2512 ?        00:00:00       cron
2513 ?        00:00:00       dbus-daemon
2524 ?        00:00:00       systemd-logind
2528 ?        00:00:00       rsyslogd
2547 ?        00:00:00       polkitd
2625 pts/2    00:00:00       agetty
2825 ?        00:00:00       systemd
2833 ?        00:00:00         (sd-pam)
```

Un cop s'està executant el contenidor al servidor, en aquest cas amb una distribució Ubuntu 16.04, aquest sistema també està disponible a la xarxa per a compartir els recursos i els serveis que publiqui. En les estacions, de cara a poder usar els recursos dels contenidors que s'estan executant en el servidor, només cal afegir la ruta correcta per a poder arribar a aquests nous sistemes:

```
stemdev@STEMstation:~$ ip route list
default via 192.168.1.1 dev eth0  proto static  metric 1024
172.17.0.0/16 dev docker0  proto kernel  scope link  src 172.17.0.1
192.168.1.0/24 dev eth0  proto kernel  scope link  src 192.168.1.101

stemdev@STEMstation:~$ sudo ip route add 10.1.1.0/24 via 192.168.1.110
```

```
stemdev@STEMstation:~$ ip route list
default via 192.168.1.1 dev eth0  proto static  metric 1024
10.1.1.0/24 via 192.168.1.110 dev eth0
172.17.0.0/16 dev docker0  proto kernel  scope link  src 172.17.0.1
192.168.1.0/24 dev eth0  proto kernel  scope link  src 192.168.1.101

stemdev@STEMserver:~$ ping 10.1.1.125
PING 10.1.1.125 (10.1.1.125) 56(84) bytes of data.
64 bytes from 10.1.1.125: icmp_seq=1 ttl=64 time=0.265 ms
64 bytes from 10.1.1.125: icmp_seq=2 ttl=64 time=0.112 ms
```

Totes aquestes ja són tasques habituals de l'administrador de la xarxa i dependran de la configuració necessària en cada instal·lació. Així, si es configura adientment el servidor STEMserver, l'usuari de l'estació STEMstation ja podria accedir als seus recursos i serveis, però també als recursos i serveis oferts pels contenidors instanciats. Per exemple, si en el *STEMserver* afegim l'usuari *stemdev:*

```
stemadm@STEMserver:~# sudo groupadd stem
stemadm@STEMserver:~# sudo useradd -m -g stem -s /bin/bash
```

i es configura el servidor *ssh (*segons els convenis de seguretat de la instal·lació en concret), aquest podrà connectar-s'hi des de l'estació al servidor compartit:

```
stemdev@STEMstation:~$ ssh STEMserver
stemdev@stemserver's password:
Welcome to Ubuntu 17.04 (GNU/Linux 4.10.0-19-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```

Però el mateix s'aplica als serveis que s'executen en el contenidor C01 que ara ja és un node més de la xarxa compartida i es pot configurar per ús dels usuaris de la xarxa.

Podem veure doncs com el desenvolupador des de l'estació de treball pot accedir tant als seus propis contenidors com als recursos dels servidors de la xarxa o fins i tot als recursos i serveis que proporcionen els contenidors que corren en el servidor compartir. Des del punt de vista del desenvolupador no hi ha diferència, però des del punt de vista de l'administrador cal notar que s'ha pogut proveir tot un sistema virtual (disponible en el contenidor C01) només amb unes poques instruccions sense d'haver d'instal·lar una màquina virtual completa amb el seu sistema operatiu i haver de configurar la xarxes.

Interoperabilitat de LXD amb Docker

En el moment actual de popularització de Docker l'administrador d'un entorn STEM pot trobar-se fàcilment amb la necessitat d'atendre les sol·licituds per part dels desenvolupadors d'executar els seus contenidors Docker en un servidor LXD.

LXD proporciona capacitats d'executar contenidors Docker mitjançant la instal·lació del *runtime* Docker dins d'un contenidor LXD. La manera més directa d'implantar aquesta solució és doncs crear un contenidor dedicat a Docker en el servidor i oferir aquest contenidor per a executar les imatges Docker.

Per a fer això, l'administrador pot crear un nou contenidor D01 en el servidor *STEMserver* tot afegint el paràmetre *-p docker* per a indicar que aquest contenidor requereix configuracions addicionals específiques de Docker a més de les del contenidor LXC::

```
stemadm@STEMserver:~$ lxc launch ubuntu:16.04 d01 -p default -p docker
Creating d01
Starting d01

stemadm@STEMserver:~$ lxc list
+------+---------+--------------------+------+------------+-----------+
| NAME |  STATE  |        IPV4        | IPV6 |    TYPE    | SNAPSHOTS |
+------+---------+--------------------+------+------------+-----------+
| c01  | RUNNING | 10.1.1.125 (eth0)  |      | PERSISTENT | 0         |
+------+---------+--------------------+------+------------+-----------+
| d01  | RUNNING | 10.1.1.64 (eth0)   |      | PERSISTENT | 0         |
+------+---------+--------------------+------+------------+-----------+
```

Tot seguit només cal actualitzar el contenidor per a instal·lar-hi la darrera versió del paquet *docker.io* dins del contenidor i aquest estarà preparat per a usar contenidors Docker :

```
stemadm@STEMserver:~$ lxc exec d01 -- apt update
stemadm@STEMserver:~$ lxc exec d01 -- apt dist-upgrade -y
stemadm@STEMserver:~$ lxc exec d01 -- apt install -y docker.io

<<< ... part de la sortida suprimida ... >>

Setting up bridge-utils (1.5-9ubuntu1) ...
Setting up cgroupfs-mount (1.2) ...
Setting up runc (1.0.0~rc2-0ubuntu2~16.04.1) ...
Setting up containerd (0.2.5-0ubuntu1~16.04.1) ...
Setting up docker.io (1.12.6-0ubuntu1~16.04.1) ...
Adding group `docker' (GID 116) ...
Done.
Setting up ubuntu-fan (0.9.2) ...
Processing triggers for systemd (229-4ubuntu17) ...
Processing triggers for ureadahead (0.100.0-19) ..
```

Aquest contenidor LXC ara ja executa una versió de Ubuntu que té configurat el *runtime* Docker, amb la seva xarxa interna (ha rebut una nova adreça IP dins

del rang de Docker). Ara es poden usar les comandes habituals per a descarregar imatges i executar contenidors Docker:

```
stemadm@STEMserver:~$ lxc list
+------+---------+---------------------+------+------------+---------+
| NAME |  STATE  |        IPV4         | IPV6 |  TYPE      |SNAPSHOTS|
+------+---------+---------------------+------+------------+---------+
| c01  | RUNNING | 10.1.1.125 (eth0)   |      | PERSISTENT | 0       |
+------+---------+---------------------+------+------------+---------+
| d01  | RUNNING | 127.17.0.1 (docker0)|      | PERSISTENT | 0       |
|      |         | 10.1.1.64 (eth0)    |      |            |         |
+------+---------+---------------------+------+------------+---------+

stemadm@STEMserver:~$ lxc shell d01

root@d01:~# docker --version
Docker version 1.12.6, build 78d1802

root@d01:~# docker search -f is-official=true alpine
NAME      DESCRIPTION                                     STARS
OFFICIAL   AUTOMATED
alpine    A minimal Docker image based on Alpine Lin...   2161
[OK]

root@d01:~# docker run -ti alpine sh
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
627beaf3eaaf: Pull complete
Digest:
sha256:58e1a1bb75db1b5a24a462dd5e2915277ea06438c3f105138f97eb53149673c
4
Status: Downloaded newer image for alpine:latest

/ # ls -al
total 60
drwxr-xr-x    1 root     root          4096 May  7 15:09 .
drwxr-xr-x    1 root     root          4096 May  7 15:09 ..
-rwxr-xr-x    1 root     root             0 May  7 15:09 .dockerenv
drwxr-xr-x    2 root     root          4096 Mar  3 11:20 bin
drwxr-xr-x    5 root     root           380 May  7 15:09 dev
drwxr-xr-x    1 root     root          4096 May  7 15:09 etc
drwxr-xr-x    2 root     root          4096 Mar  3 11:20 home
drwxr-xr-x    5 root     root          4096 Mar  3 11:20 lib
drwxr-xr-x    5 root     root          4096 Mar  3 11:20 media
drwxr-xr-x    2 root     root          4096 Mar  3 11:20 mnt
dr-xr-xr-x  242 nobody   nobody           0 May  7 15:09 proc
drwx------    1 root     root          4096 May  7 15:09 root
drwxr-xr-x    2 root     root          4096 Mar  3 11:20 run
drwxr-xr-x    2 root     root          4096 Mar  3 11:20 sbin
drwxr-xr-x    2 root     root          4096 Mar  3 11:20 srv
dr-xr-xr-x   13 nobody   nobody           0 May  7 15:09 sys
drwxrwxrwt    2 root     root          4096 Mar  3 11:20 tmp
drwxr-xr-x    7 root     root          4096 Mar  3 11:20 usr
drwxr-xr-x   12 root     root          4096 Mar  3 11:20 var
/ #
```

Com es pot observar, la darrera comanda *ls -al* s'ha executat dins un sistema Alpine en un contenidor Docker, que està dins un contenidor LXC amb sistema Ubuntu 16.04, que a la vegada corre en el servidor Ubuntu *STEMserver*.

Aplicació a les necessitats STEM

Aquesta configuració combinada de LXD amb Docker proporciona a l'administrador STEM prou flexibilitat com per a oferir serveis d'execució de contenidors de dues classes: la nativa LXD del sistema Ubuntu i la més popular del sistema Docker.

Els desenvolupadors a la seva vegada poden enviar els seus contenidors per a que l'administrador els desplegui en el servidor compartit, que possiblement disposi de més recursos i rendiment per a la seva execució. Amb aquesta funcionalitat es pot atendre la necessitat **ADM03** de l'administrador de «Desplegar aplicacions en el centre de computació» fins i tot delegant aquestes tasques als desenvolupadors que poden crear contenidors Docker en un contenidor LXD privat.

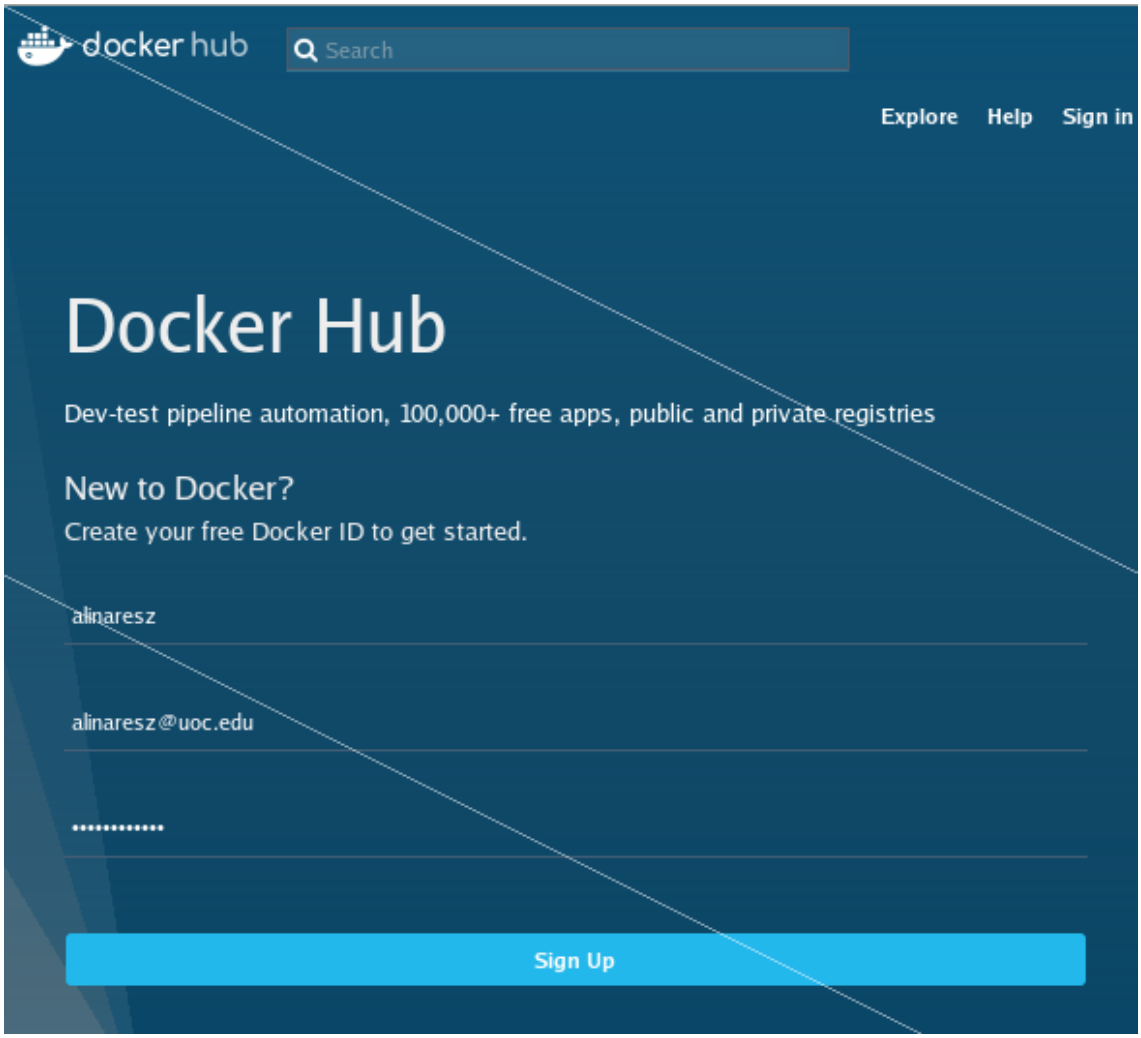
Per altra banda aprofitant la capacitat de gestió d'imatges (i el sistema de cache de Docker), l'administrador pot actualitzar de forma coordinada multitud de sistemes virtuals (en forma de contenidors pels usuaris) per sol fet d'actualitzar una imatge base dels contenidors. Tots els contenidors dels desenvolupadors basats en aquesta imatge genèrica s'actualitzen automàticament en haver-se modificat una de les capes inferiors. Combinant totes aquests tècniques l'administrador de l'entorn STEM pot satisfer les necessitats d'aquest entorn de mantenir actualitzats un nombre elevat de sistemes virtuals (necessitat **ADM01** i **ADM02**) sense haver d'actualitzar-los un per un, com s'hauria de fer en el mateix escenari si fos construït usant màquines virtuals.

3.2.7 Repositori públic

Crear un repositori d'imatges Docker

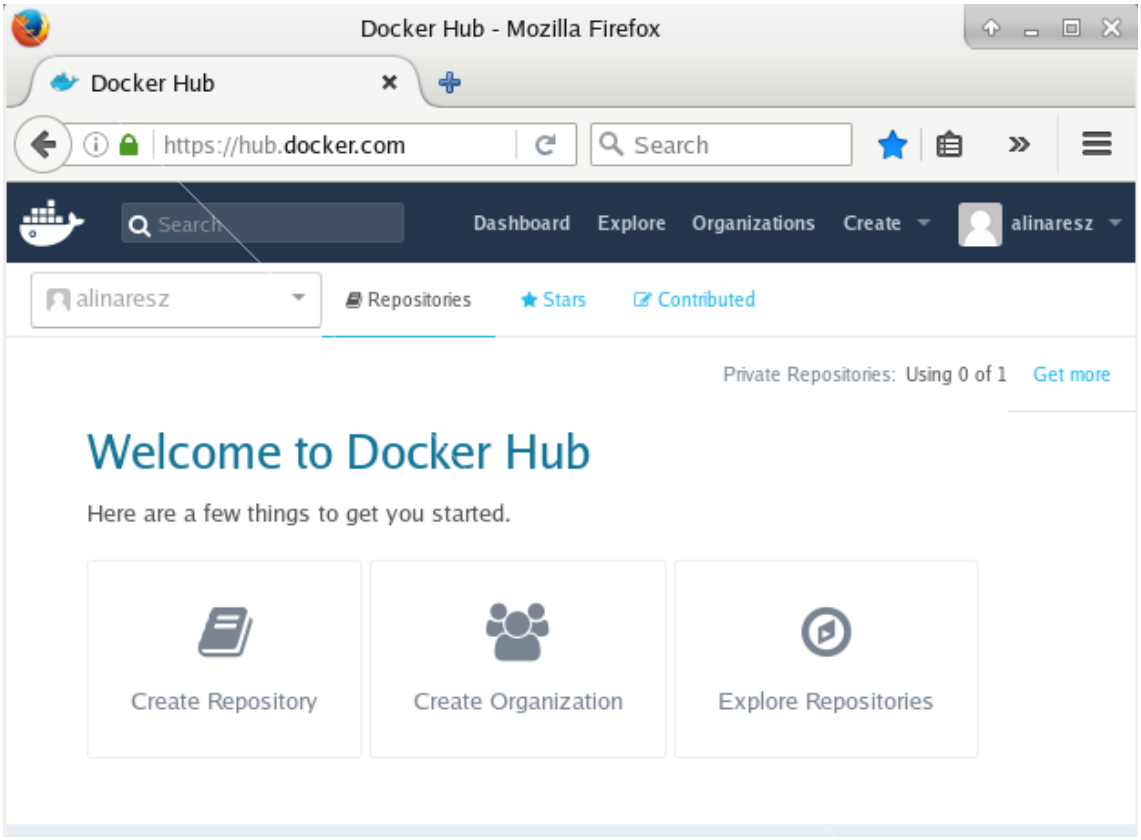
Una de les tasques assignades al rol d'administrador de l'entorn STEM és proporcionar al autors un repositori d'imatges per a compartir contenidors. Aquest repositori pot ser totalment obert i públic i bé d'ús intern o limitat a determinats tercers revisors. Algunes de les solucions més modernes de contenidors proporcionen una infraestructura per a aquest propòsit.

La solució Docker ofereix el *Docker Hub* per a publicar i obtenir imatges de contenidors. L'alta d'un nou usuari al registre[45] és simple i les característiques disponibles com nombre de repositoris, controls d'usuaris i grups, permisos, etc. ja depenen de cada solució concreta.



45 En la terminologia específica de la solució Docker, el repositori principal dels usuaris es coneix com *registre* i per a cada possible imatge i totes les seves diferents versions o *etiquetes* es crea un *repositori*. En aquest treball ens referirem de forma genèrica al lloc on publicar i compartir imatges de contenidors com el *repositori* independentment de la seva implantació en una solució determinada.

Un cop afegit el nou usuari al servei de Docker Hub ja es poden crear nous repositoris a on publicar imatges de contenidors i fer altres operacions que ens permeti el servei:



Publicar una imatge Docker al seu repositori

Els autors, per la seva banda, ara ja poden publicar les seves imatges en el repositori que els ha proporcionat l'administrador. Per fer-ho amb Docker, cal crear o marcar les imatges amb el nom del repositori i enviar-les, un cop s'ha fet *login* amb el nom d'usuari corresponent.

Per a comprovar el funcionament, agafarem una de les imatges creades en les seccions anteriors i la marcarem amb una nova etiqueta associada al repositori acabat de crear:

```
stemdev@STEMstation:~$ docker images
REPOSITORY    TAG     IMAGE ID            CREATED             SIZE
gcc-parallel  latest  f0de41057bdb        6 days ago          1.49 GB
gcc-stemdev   latest  7df891118e20        7 days ago          1.61 GB
gcc-gsl-2.3   latest  e7f5d27051b5        7 days ago          1.59 GB
gcc-gsl       latest  04be6c0624eb        7 days ago          1.52 GB
python        latest  a0d32d529a0a        2 weeks ago         689 MB
python        2.7.13  eb03485f8ec8        2 weeks ago         678 MB
gcc           latest  ecc92f2be252        2 weeks ago         1.49 GB
debian        latest  054abe38b1e6        2 weeks ago         123 MB
alpine        latest  4a415e366388        2 months ago        3.99 MB
```

Per exemple, reanomenem la imatge *gcc-gsl-2.3* amb el nom del repositori *alinaresz/gcc-gsl* i li donem l'etiqueta per defecte *latest:*

```
stemdev@STEMstation:~$ docker tag gcc-gsl-2.3 alinaresz/gcc-gsl

stemdev@STEMstation:~$ docker images
REPOSITORY            TAG    IMAGE ID      CREATED             SIZE
gcc-parallel          latest f0de41057bdb  6 days ago          1.49 GB
gcc-stemdev           latest 7df891118e20  7 days ago          1.61 GB
alinaresz/gcc-gsl     latest e7f5d27051b5  7 days ago          1.59 GB
gcc-gsl-2.3           latest e7f5d27051b5  7 days ago          1.59 GB
gcc-gsl               latest 04be6c0624eb  7 days ago          1.52 GB
python                latest a0d32d529a0a  2 weeks ago         689 MB
python                2.7.13 eb03485f8ec8  2 weeks ago         678 MB
gcc                   latest ecc92f2be252  2 weeks ago         1.49 GB
debian                latest 054abe38b1e6  2 weeks ago         123 MB
alpine                latest 4a415e366388  2 months ago        3.99 MB
```

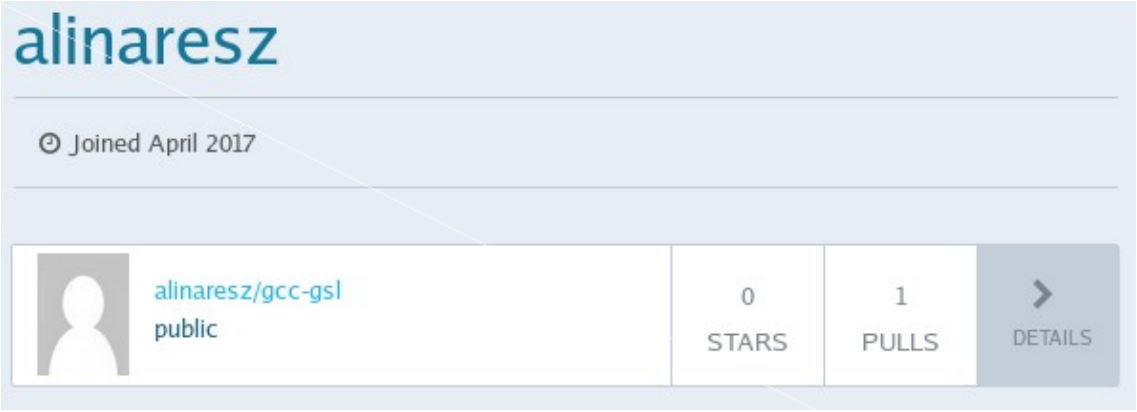
Es pot observar que l'identificador de la imatge (**e7f5d27051b5**) és el mateix. El sistema d'imatges de Docker no ha creat cap capa addicional per a fer aquesta nova imatge.

Ara l'autor es pot connectar al repositori, on l'administrador li hagi donat permisos, i enviar aquesta imatge des de la seva estació de treball:

```
stemdev@STEMstation:~$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If
you don't have a Docker ID, head over to https://hub.docker.com to
create one.
Username: alinaresz
Password:
Login Succeeded

stemdev@STEMstation:~$ docker push alinaresz/gcc-gsl
The push refers to a repository [docker.io/alinaresz/gcc-gsl]
065203a55518: Pushed
f666b2722e0d: Pushed
9549a9dc266c: Pushed
c9b47193bfe8: Pushed
39b80ff7f17e: Pushed
eda5b29538df: Pushed
d359ab38b013: Pushed
682e7cee9d37: Pushed
295d6a056bfd: Pushed
latest: digest:
sha256:a83bf0b8006c5a4bbe840d3262c992bc69dd3acf023dc8f9e07058b9c99d05f
5 size: 2218
```

Un cop pujada al repositori la imatge (el temps d'enviament dependrà òbviament de la mida de la imatge i de la velocitat de la connexió) ja serà visible des de les interfícies gràfiques dels serveis de Docker Hub.

Com l'usuari Docker pot comprovar la imatge està disponible de forma pública:



L'autor pot afegit descripcions addicionals al repositori per tal que altres usuaris de Docker puguin trobar la imatge i descarregar-la:

Si ara es fa una cerca genèrica amb la comanda *docker search* buscant el terme *gsl* , que busca per tot el Docker Hub, ja apareix la imatge publicada:

```
stemdev@STEMstation:~$ docker search gsl
NAME                    DESCRIPTION             STARS OFFICIAL AUTOMATED
gslin/tor               Tor on Debian (jessie)      1
jaewonpark/gsl          gsl auto-build              0            [OK]
oscarraig/gsl-boost     gsl boost                   0            [OK]
alinaresz/gcc-gsl-2.3 GCC with GSL 2.3              0

<<< ... resta de la sortida suprimida ... >>
```

i la mateixa cerca del terme *gsl* per un usuari qualsevol a la web proporciona:

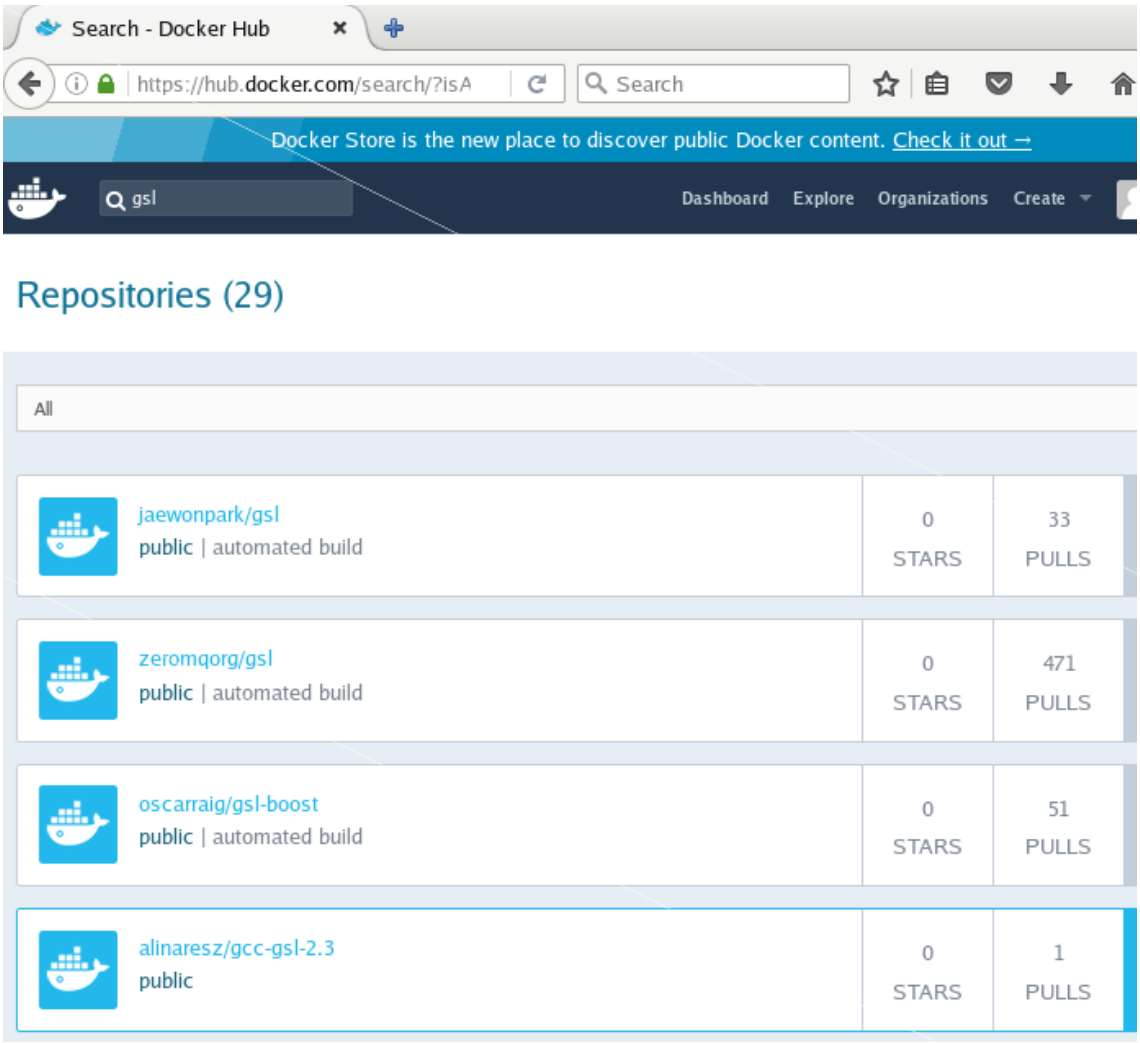


Figura 4: El contenidor publicat apareix en una cerca pública a DockerHub

Aplicació a les necessitats STEM

Mitjançant el servei de repositori proporcionat per Docker s'ha pogut comprovar com un autor pot publicar amb força facilitat una imatge de contenidor en el repositori compartit. Aquesta funcionalitat permet atendre la necessitat **AUT02** de publicar aplicacions.

En la mesura que el contingut del contenidor és de plena disposició per a l'autor, qui pot crear les imatges que calgui afegint capes i copiant en la imatge tots els executables, arxius de dades de treball o codi font que sigui necessari, també es compleix amb la necessitat **AUT01** d'empaquetar aplicacions, dades i resultats per a l'avaluació de tercers. El gran avantatge de la virtualització d'aplicacions és que aquestes es poden publicar junt amb les llibreries que necessitin, o bé tot un sistema operatiu de base si fos el cas.

3.2.8 Ús de contenidors Docker en Linux

En les seccions anteriors ja hem vist com instal·lar i usar contenidors Docker des de Linux. El contenidor Docker D01 que corre en el servidor STEMserver és un sistema Linux i per tant es pot descarregar aquesta nova imatge enviada al repositori, tal com faria un administrador desplegant el contenidor en un centre de càlcul o un revisor avaluant una imatge concreta en la seva estació de treball. De fet qualsevol usuari de Docker pot accedir al servei DockerHub i fer una cerca per trobar la imatge i ja podria instanciar un contenidor a partir de la imatge i executar-lo.

Descarregar i executar un contenidor publicat

Veiem com ho faria l'administrador en el supòsit que vulgues executar el contenidor Docker *alinaresz/gcc-gls* en el *STEMServer* que executa la solució LXC on a la seva vegada s'està executant un contenidor amb un sistema Ubuntu que ja hi té el *runtime* Docker instal·lat:

```
stemadm@STEMserver:~$ lxc list
+------+---------+---------------------+------+------------+---------+
| NAME |  STATE  |        IPV4         | IPV6 |   TYPE     |SNAPSHOTS|
+------+---------+---------------------+------+------------+---------+
| c01  | RUNNING | 10.1.1.125 (eth0)   |      | PERSISTENT | 0       |
+------+---------+---------------------+------+------------+---------+
| d01  | RUNNING | 127.17.0.1 (docker0)|      | PERSISTENT | 0       |
|      |         | 10.1.1.64 (eth0)    |      |            |         |
+------+---------+---------------------+------+------------+---------+

stemadm@STEMserver:~$ lxc shell d01
Last login: Sun May 14 10:38:37 UTC 2017 on UNKNOWN
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.10.0-20-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

7 packages can be updated.
0 updates are security updates.

root@d01:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
```

Es comprova que encara no hi ha cap contenidor Docker executant-se dins el contenidor D01, però ara es pot cercar i descarregar la imatge desitjada:

```
root@d01:~# docker search alinaresz
NAME                DESCRIPTION         STARS     OFFICIAL   AUTOMATED
alinaresz/gcc-gsl   GCC with GSL 2.3    0

root@d01:~# docker pull alinaresz/gcc-gsl
```

```
Using default tag: latest
latest: Pulling from alinaresz/gcc-gsl
cd0a524342ef: Pull complete
e39c3ffe4133: Pull complete
85334a7c2001: Pull complete
4c546d9d6a84: Pull complete
ca5955eb1884: Pull complete
7f1a2329e0fe: Pull complete
339248b9ecfc: Pull complete
26933fdd4fb6: Pull complete
8cf4015a690c: Pull complete
Digest:
sha256:a83bf0b8006c5a4bbe840d3262c992bc69dd3acf023dc8f9e07058b9c99d05f
5
Status: Downloaded newer image for alinaresz/gcc-gsl:latest

root@d01:~# docker images
REPOSITORY        TAG     IMAGE ID        CREATED          SIZE
alinaresz/gcc-gsl latest  e7f5d27051b5    7 days ago       1.593 GB
alpine            latest  4a415e366388    10 weeks ago     3.984 MB
```

Un cop la imatge ja està disponible en el sistema Ubuntu es pot executar qualsevol programa que estigui instal·lat en el contenidor que es crei a partir d'aquesta imatge:

```
root@d01:~# docker run alinaresz/gcc-gsl gcc --version
gcc (GCC) 6.3.0
Copyright (C) 2016 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.

root@d01:~# docker run alinaresz/gcc-gsl gsl-config --version
2.3
```

En aquest cas s'han executat els programes *gcc* i *gsl-version* dins d'un contenidor Docker, en un sistema Ubuntu que a la seva vegada és també un contenidor LXC que corre dins del servidor STEMserver.

Aplicació a les necessitats STEM

Un cop una imatge està publicada en el repositori, aquesta pot ser usada per l'administrador per a desplegar aplicacions en el centre del càlcul (**ADM03**) o bé per un revisor per tal d'obtenir una aplicació o un experiment numèric (**REV01**) i poder-lo avaluar.

Aquesta avaluació per un tercer revisor dependrà del sistema usat en el seu entorn que no té perquè ser l'original. Hem vist doncs com fer una revisió amb un sistema també Linux (**REV02**).

3.2.9 Ús de contenidors Docker en Windows

Instal·lació de Docker en Windows

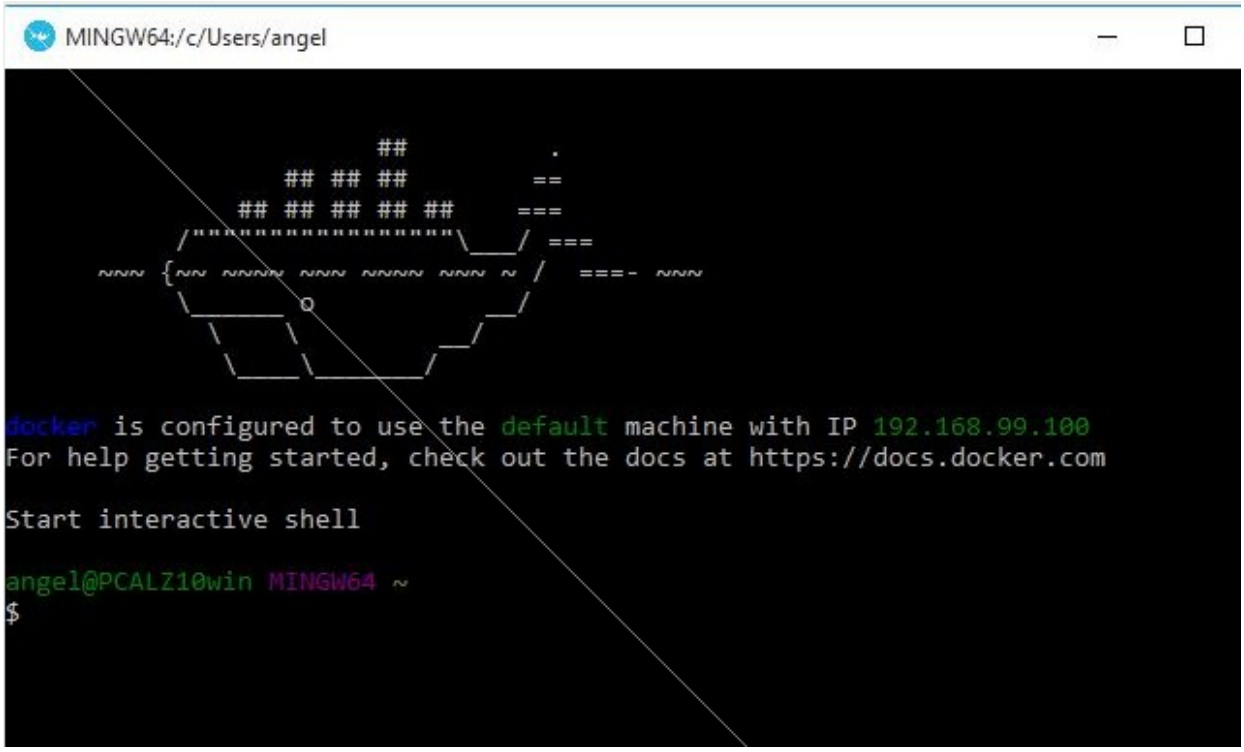
Docker també permet treballar amb contenidors des del sistema operatiu *Microsoft Windows*. En un entorn STEM, això permet que els contenidors publicats pels autors puguin descarregar-se i executar-se per revisors amb aquest sistema operatiu.

Donat que els contenidors Linux fan ús de les capacitats pròpies del nucli Linux, per a poder-se instal·lar en un sistema Windows, Docker proporciona un entorn Linux addicional en una màquina virtual. Aquests entorns Docker en Windows estan evolucionant constantment amb l'objectiu de fer que siguin el més propers possibles a un entorn natiu Linux i canvien també en funció de la versió de Windows disponible i dels requisits necessaris que demana la solució Docker en cada cas. Des de la pàgina web de Docker es pot descarregar el software més actualitzat i segons la versió de Windows pot ser necessari instal·lar en Windows el *Docker Toolbox* que afegeix les eines necessàries per a obtenir un entorn Linux on córrer els contenidors:

Veurem un exemple d'instal·lació del *Docker Toolbox* en Windows 10 per a executar els contenidors descarregats del Docker Hub com en la secció anterior.

En primer lloc cal instal·lar el *Docker Toolbox*:

Aquest software crea i configura una màquina virtual proporcionada per Docker que un cop instal·lada i configurada ens presenta una nova finestra Windows amb un *prompt* de Linux:

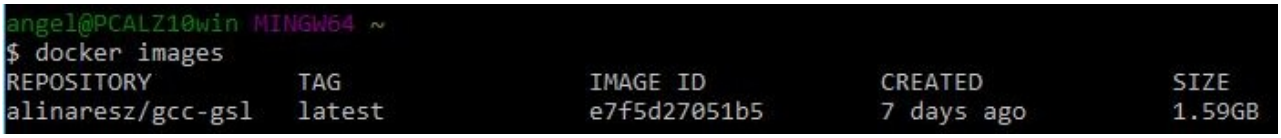


Ús dels contenidors Docker en Windows

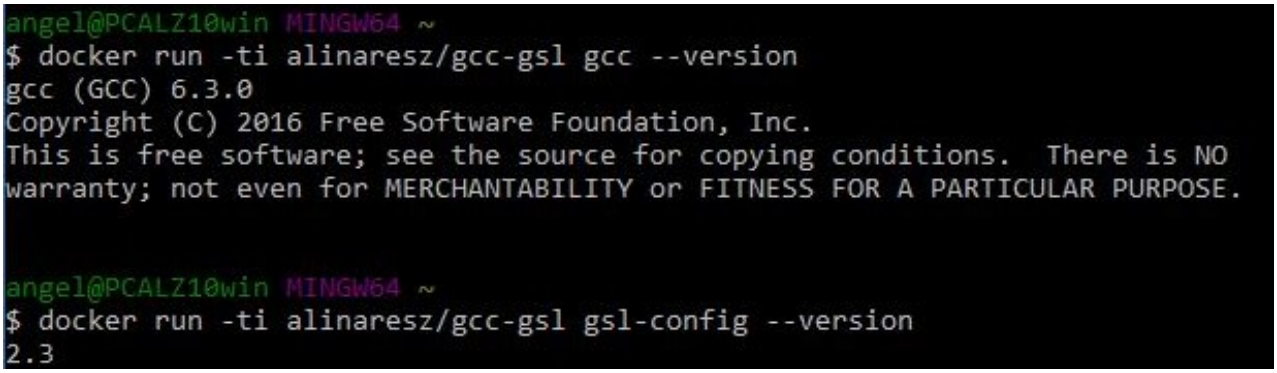
Ara ja està disponible Docker per a treballar en Windows i es pot, per exemple, cercar en el Docker Hub la imatge que es vol avaluar. En la següent captura de

pantalla es pot veure com es cerca la imatge i es descarrega amb la comanda habitual *docker pull*:

La imatge descarregada del repositori públic està disponible en el sistema com es pot comprovar amb la comanda *docker images*:



A continuació ja es poden executar els contenidors usant les mateixes comandes *docker run* que en la secció anterior usant Linux. Docker crearà els contenidors a partir de la imatge descarregada i executarà les comandes d'exemple per a mostrar les versions instal·lades del GCC i GSL:

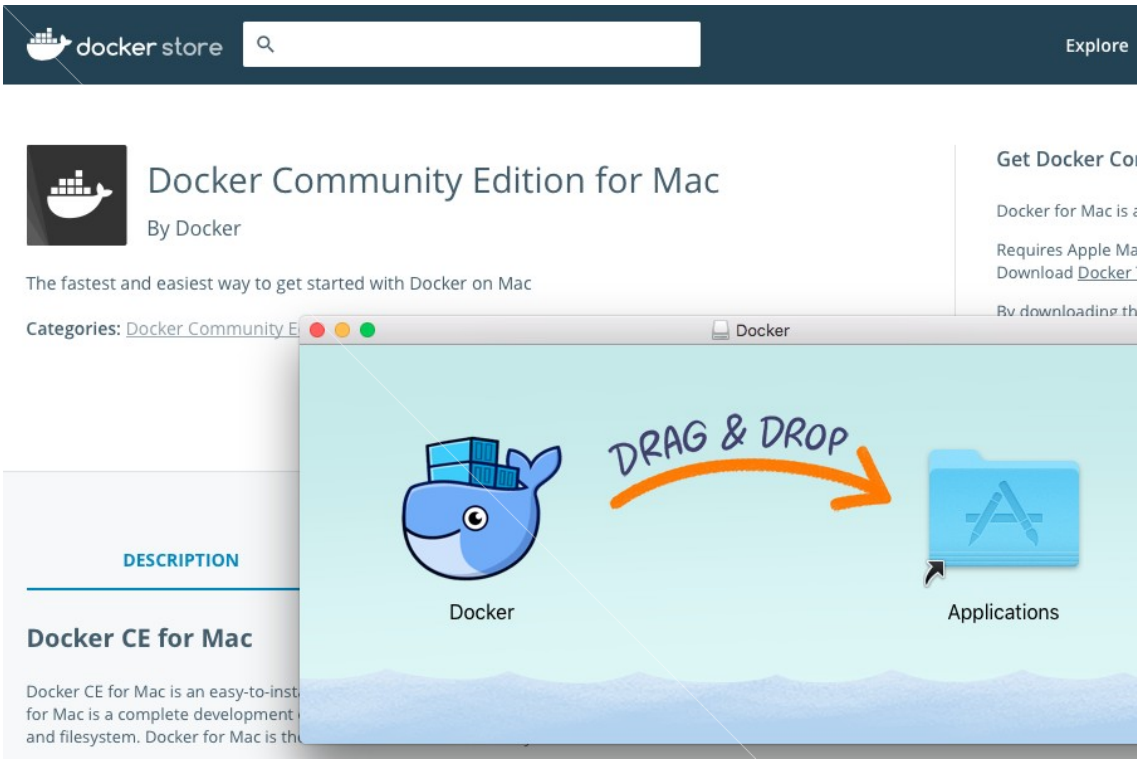


3.2.10 Ús de contenidors Docker en macOS

Instal·lació de Docker Toolbox en macOS

Docker permet treballar també des del sistema operatiu Apple *macOS* amb la seva solució de contenidors.

Malgrat ser un sistema derivat de UNIX i per tant proper a Linux, l'ús específic de les capacitats del *kernel* Linux fa que els contenidors no s'executin sobre el nucli Darwin, sinó que es proporciona també un entorn Linux simulat.



El mecanisme és completament anàleg al vist amb Windows i *Docker Toolbox* i funciona de la mateixa manera.

Instal·lació de Docker en una màquina virtual en macOS

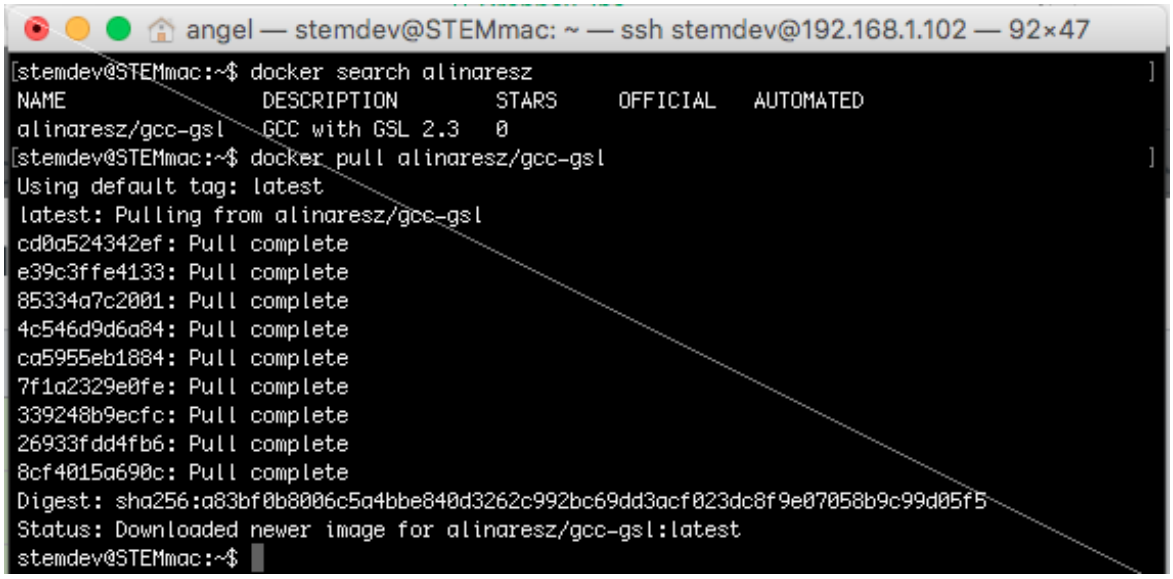
Per altra banda, l'administrador de la xarxa també té la possibilitat de fer directament el mateix que fa *Docker Toolbox* i instal·lar explícitament una màquina virtual per a tenir més control sobre l'administració de la mateixa i fer el manteniment de la mateixa manera que amb la resta dels equips de la instal·lació.

En l'exemple a continuació, l'administrador crea una màquina virtual Debian en un entorn VirtualBox sobre macOS que, un cop preparat, en connectar-se ja es comporta com un sistema Linux:

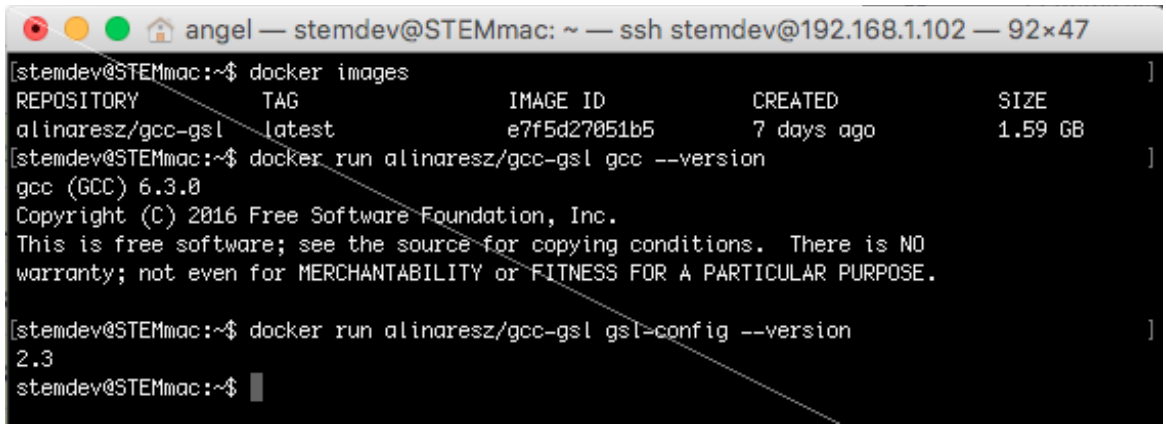
Des del Teminal de OS X es pot obrir una connexió SSH a la màquina virtual Linux i treballar com en les seccions anteriors:

```
PCALZ7osx:~ angel$ ssh stemdev@192.168.1.102
The authenticity of host '192.168.1.102 (192.168.1.102)' can't be established.
ECDSA key fingerprint is SHA256:a6gC8hcUi5Xb1DWF1mbBiSbLLZJ9LoDNvVIhz8njns8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.102' (ECDSA) to the list of known hosts.
stemdev@192.168.1.102's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun May 14 14:39:57 2017 from 192.168.1.8
stemdev@STEMmac:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
stemdev@STEMmac:~$ docker search alinaresz
NAME                DESCRIPTION         STARS     OFFICIAL   AUTOMATED
alinaresz/gcc-gsl   GCC with GSL 2.3    0
stemdev@STEMmac:~$
```

A partir d'aquest moment està disponible la solució de contenidors Docker en un sistema operatiu diferent de Linux (malgrat fer-ho en una màquina virtual, igual que en el sistema Ubuntu Server es feia sobre un contenidor LXD) i es poden realitzar les operacions habituals amb les imatges i els contenidors:



I finalment treballar amb les comandes tot emprant executables i llibreries Linux:



Aplicació a les necessitats STEM

La solució Docker proporciona doncs certa capacitat d'executar contenidors en sistemes operatius aliens a Linux permeten als revisors arribar a fer una avaluació de l'aplicació o els resultats numèrics publicats (necessitat **REV02**). La solució actual, però, es mitjançant l'ús de màquines virtuals intermèdies per la qual cosa tot i ser funcionals en termes d'avaluació, no ho seran tant en termes de rendiment computacional respecte al contenidor natiu.

4 Anàlisi dels Resultats

4.1 Anàlisi del cas particular

4.1.1 Resultats del cas

Satisfacció de les necessitats STEM

En l'aplicació pràctica s'ha pogut comprovar que les necessitats identificades per als diferents rols en un entorn STEM poden ser satisfetes. Seguint el model plantejat del cicle productor-consumidor, i aprofitant el paral·lelisme entre la producció cientificotècnica i el cicle bàsic del desenvolupament del programari, es pot comprovar que es pot fer ús de les pràctiques més actuals basades contenidors i adaptar-les als entorns STEM.

Els **desenvolupadors** STEM, en la part de la seva feina directament relacionada amb la creació de solucions computacionals, poden aprofitar les solucions de contenidors, en aquest cas Docker, per a disposar de total flexibilitat a l'hora d'alternar les eines informàtiques, siguin aquestes els compiladors o intèrprets, les llibreries científiques o les aplicacions d'alt nivell. També s'ha comprovat com amb l'ús dels contenidors es poden crear entorns de gran paral·lelisme per a simular xarxes de nodes concurrents.

Els **administradors** STEM, a la seva vegada, poden aprofitar la tecnologia dels contenidors per a proporcionar i mantenir actualitzada tota una biblioteca compartida d'imatges prèviament construïdes i altament optimitzades per als desenvolupadors. A més, poden delegar les tasques de configuració interna dels entorns computacionals als mateixos desenvolupadors i rebre els contenidors ja preparats per la seva execució en el centre de computació. Fins i tot, poden configurar entorns accessibles als desenvolupadors per a que ells mateixos facin el desplegament dels contenidors en els servidors de major rendiment.

Els **autors**, amb la tecnologia de contenidors, disposen de la infraestructura necessària per a empaquetar les solucions o experiments computacionals i poder-los distribuir a tercers de forma asíncrona a través d'un repositori.

Aquests paquets computacionals, que poden constar de codi executable o codi font, dades inicials o resultats finals i la seva documentació, són els que els **revisors** podran descarregar dels repositoris a la seva conveniència per a executar la solució que necessiten o per a validar experiments i resultats numèrics.

En una mesura o altra, totes les necessitats bàsiques inicialment identificades en aquest model STEM representatiu s'han pogut implantar i validar comprovant que superen els seus respectius criteris d'acceptació.

4.1.2 Valoració econòmica

Cost del cas pràctic

La implantació del cas pràctic s'ha pogut realitzar a cost zero en relació a l'adquisició del programari necessari. Totes les solucions emprades són de codi obert i gratuïtes, per la qual cosa el seu impacte en els pressupostos dels projectes STEM és nul.

A més la inversió en temps necessària per a instal·lar el *runtime* Docker en cada estació és fixat i únic, de manera que un cop preparada cada estació són els propis usuaris els que es poden proveir de les imatges necessàries sense la participació de l'administrador.

En la banda del centre computacional i els servidors compartits, tant la solució LXC/LXD com la capacitat d'interoperabilitat amb Docker també requereix només una inversió fixa en temps inicial i, amb la configuració adient i tenint en compte els requisits específics de seguretat de cada instal·lació, els administradors poden arribar a oferir una solució que els desenvolupadors i els autors usin en mode auto-servei.

Amb les solucions actuals de contenidors també es disposa d'infraestructures al núvol de cost zero de manera que la publicació i descàrrega dels contenidors no té cap cost per les parts. Els autors i revisors també poden fer ús dels seus sistemes actuals per avaluar contenidors Linux tot mantenint els seus sistemes propietaris Windows o macOS.

El cost de les aplicacions comercials que es puguin usar en la instal·lació STEM és un cost que no queda alterat per la incorporació de solucions de contenidors. De fet, el que pot arribar a succeir és que algunes d'aquestes aplicacions puguin ser substituïdes per alternatives gratuïtes de codi obert un cop s'hagi avaluat que l'aplicació alternativa és totalment operativa i funcional.

Cost d'explotació amb suport empresarial

En determinades instal·lacions pot ser un requisit necessari el poder disposar de manteniment i suport de nivell empresarial per a les solucions implantades. Tant la solució Docker com la solució LXD disposen de suport de nivell empresarial.

En el cas de Docker és recent la disponibilitat de l'edició *Docker Enterprise* que proporciona suport a partir d'uns 750$/any per node. En el cas del sistema Ubuntu Server també està disponible el servei de suport *Advantage for Servers*, que proporciona suport de LXD a partir d'uns 750$/any per servidor.

Els administradors poden doncs elegir quines parts de la seva xarxa cobrir amb manteniment de nivell empresarial i quines parts mantenir de forma interna amb els seus propis recursos.

4.2 Anàlisi del cas general

4.2.1 Valoració de les solucions basades en contenidors

Les solucions de contenidors Linux, aplicades als entorns de ciència i enginyeria, tenen un potencial que ara només s'està començant a explotar. En la mesura que el cicle de producció científica es sembli al de producció de programari, i això és cada cop més habitual, es poden aprofitar les pràctiques i les tècniques d'aquest darrer domini en els entorns STEM.

Per altra banda, és propi dels àmbits de desenvolupament de programari treballar amb solucions on la innovació és constant. Aquesta evolució de la tecnologia és acceptable en projectes software de durada relativament curta on s'accepta fins a cert grau treballar amb versions «a millorar». Això no sempre resultarà del tot positiu en entorns com els STEM que poden requerir de certa estabilitat durant el període de recerca i desenvolupament que en determinats projectes pot ser de llarga durada[46]. Els administradors STEM hauran de trobar l'equilibri entre satisfer la demanda dels desenvolupadors de disposar de les darreres solucions i garantir l'estabilitat i sobretot la seguretat necessària en els entorns compartits.

Addicionalment, la promesa d'execució universal dels contenidor no resulta tal ara per ara. En el cas de clients Windows o macOS en realitat s'instal·len màquines virtuals i per tant el rendiment com a mínim, sinó els propis resultats computacionals encara no són els mateixos en aquestes entorns que en sistemes Linux natius. A mida que es segueixin popularitzant les solucions de contenidors s'haurien d'anar millorant aquestes implantacions. En *macOS* semblaria que l'adaptació dels *runtimes* al seu nucli seria més simple, mentre que en *Windows* també hi ha una tendència recent a intentar proporcionar suport més directe a les crides del nucli Linux.

El punt més feble de les solucions de contenidors Linux és a la vegada el seu principal avantatge: els contenidors són entorns simulats que compateixen el mateix nucli Linux que el sistema amfitrió. Això té implicacions en les solucions cientificotècniques, de forta càrrega computacional, doncs aquest únic nucli que inicialment dona servei a l'amfitrió (p.e. el servidor LXD) no pot estar optimitzat a la vegada tant per a la càrrega de treball d'un servidor com per a les necessitats computacionals dels contenidors. En els entorns STEM els contenidors executaran aplicacions que tenen requisits de màxim ús dels recursos i d'alt rendiment, que són contradictoris amb els requisits de màxima compartició de recursos i de temps de CPU dels servidors.

46 Serveixi com exemple que durant la realització d'aquest mateix treball, Docker ha modificat la seva oferta de solucions passant a proporcionar dues versions del producte: *Docker Community* i *Docker Enterprise.*

4.2.2 Aplicació dels contenidors a la reproductibilitat

La tesi inicial d'aquest treball és que les similituds cada cop majors entre bona part del treball cientificotècnic i del treball de desenvolupament de programari fa possible que, en el cas concret dels contenidors, es puguin aplicar aquestes tècniques els entorns STEM. És més, la promesa de transport i execució ubiqua i inalterada dels contenidors podria ser potencialment de gran utilitat en la cerca d'una major reproductibilitat científica, si més no en termes de resultats computacionals.

La realitat és que els contenidors, com a abstracció d'entorns més o menys aïllats de grups de processos dins d'un sistema operatiu, segueixen depenen òbviament de l'arquitectura de processador de l'amfitrió que instancia el contenidor i que executa aquests processos de forma nativa. Això té l'avantatge respecte a les solucions de màquines virtual que s'obté un rendiment gairebé natiu però contràriament no proporciona cap independència del llenguatge màquina del processador amb les implicacions, molt importants, que això té en aspectes derivats de l'aritmètica implantada en la CPU. Els resultats numèrics d'un determinat experiment computacional poden dependre fortament de l'aritmètica disponible en l'arquitectura de la CPU, per la qual cosa es podrien arribar a obtenir fins i tot resultats diferents per un mateix codi font.

En aquest sentit els contenidors proporcionen mecanismes de publicació i distribució d'aplicacions empaquetades amb les seves dependències, però no abstreuen, al contrari que les màquines virtuals, del maquinari de base. Les solucions basades en contenidors milloren la reproductibilitat en termes d'empaquetat i de transport de les solucions i els resultats però no poden independitzar els processos de càlcul de l'arquitectura física. Així, revisors treballant en arquitectures de computadors diferents podrien obtenir resultats diferents quan es treballa al nivell de precisió numèrica que es requereix en determinats entorns STEM.

5 Conclusions

5.1.1 El treball realitzat

Hipòtesi inicial

La realització del present treball ha permès posar a prova i finalment validar la hipòtesi inicial de que les solucions de contenidors Linux poden incorporar-se als processos de desenvolupament en els entorns cientificotècnics. Això, de fet, ja està succeint en els entorns reals degut a la pressió de la popularitat d'eines com Docker sobre la comunitat STEM. És important, però, que en aquests tipus d'aplicacions cientificotècniques, que poden tenir una repercussió a llarg termini per a la societat molt més gran que una nova aplicació mòbil o un nou web, la incorporació d'aquestes solucions no es faci a costa de la qualitat i el rigor.

Més enllà de la millora en els processos de creació i distribució de solucions, l'aplicació pràctica realitzada posa de manifest que l'aportació real en termes de reproductibilitat computacional són interessants però, en cap cas, representen un salt qualitatiu en aquest problema ben actual per a la comunitat científica i tècnica.

Lliçons apreses

Entre les lliçons apreses del treball es pot destacar la relativa facilitat en la incorporació d'aquestes tècniques als entorns STEM i el benefici pràctic immediat que se'n pot obtenir.

Sempre i quan es tingui present que l'aplicació d'aquestes tècniques en els projectes STEM té el seu lloc com una eina més i no pas com una finalitat en sí mateixa (a diferència del que es promociona moltes vegades en els entorns de desenvolupament de software), l'ús de solucions basades en contenidors Linux proporciona determinats beneficis i pot millorar tant els processos de creació i distribució com els de revisió cientificotècnica.

Gestió del projecte

El treball realitzat, des del punt de vista de la gestió de projectes, posa de manifest que, en els estadis inicials en l'adopció de solucions de contenidors en entorns STEM, és millor fer una aproximació iterativa que no pas implantar una solució totalment planificada de forma prèvia. La naturalesa del treball realitzat, tant en la seva durada com en la seva forma, no ha permès realitzar vàries iteracions, que en un entorn real resultarien molt més productives. El model plantejat d'un escenari STEM genèric i ideal és només un primer punt de partida i les necessitats identificades són prou genèriques com per a poder plantejar i avaluar les solucions proposades però que en cada escenari STEM concret, òbviament, caldria adaptar.

5.1.2 El treball futur

Partint del treball actual, hi ha diverses línies de treball que es podrien desenvolupar.

Cicle de desenvolupament de solucions STEM

Per la part del procés de desenvolupament, publicació i distribució de solucions cientificotècniques, una anàlisi més fina dels rols i necessitats dels actors implicats en un entorn STEM permetria una millor especificació, disseny, i implantació de les solucions disponibles a casos més realistes.

Administració de solucions STEM

Per altra banda, des del punt de vista de l'administració de les xarxes i els sistemes propis dels entorns STEM caldria avançar en dues línies.

Per un costat, la proliferació de contenidors i per tant la seva gestió tant immediata (desplegament) com a llarg termini (manteniment) crea nous reptes per a l'administrador de la xarxa. Tota una generació de noves eines de gestió i orquestració dels contenidors instanciats ja estan disponibles en l'àmbit del desenvolupament de software i caldria analitzar, en un treball similar al present, com es podrien aplicar aquestes als entorns STEM.

Per l'altre costat, les clares diferències en els requisits demandats per les solucions STEM vers la simple i immediata virtualització d'aplicacions que propicien algunes solucions actuals, fa necessari analitzar a fons si les eines genèriques, que inicialment s'introduirien a través dels entorns dels desenvolupadors STEM, són realment d'ús i aplicació pràctica en sistemes d'alt rendiment gestionats pels administradors.

Reproductibilitat computacional

Una línia addicional de treball podria analitzar amb detall les implicacions que pot tenir l'aparent facilitat en la portabilitat de solucions basades en contenidors, tant en termes de precisió numèrica dels resultats com en garantir que realment s'avança en la reproductibilitat de resultats computacionals.

6 Glossari

API: acrònim de *application programming interface*. Conjunt de funcions oferts per una component de programari per al seu ús per tercers.

Big Data: tècniques i procediments per a la manipulació de grans quantitats de dades amb la finalitat de detectar patrons i fer inferències sobre els conjunts de dades.

BLAS: especificació d'un conjunt de llibreries científiques d'àlgebra lineal amb diferents implantacions disponibles.

Cloud Computing: paradigma de computació on la funcionalitat s'ofereix en forma de serveis a través d'Internet.

Cluster: agrupació de computadors que es presenten com una unitat per a oferir recursos i funcionalitat de gran capacitat.

CPD: acrònim de c*entre de processament de dades.*

CPU*: acrònim de central processing unit.* El processador central dels ordinadors.

GPU: acrònim de *graphics processing unit*. En contraposició a CPU, aquesta unitat té per missió el processament especialitzat de gràfics. En entorns STEM també s'usen les seves característiques técniques per a implantar solucions de computació d'alt rendiment.

Hipervisor: aplicació que crea màquines virtuals i executa codi simulant que aquest corre en un processador físic.

HPC: acrònim de *high performance computing*. Fa referència al maquinari i programari i als mètodes de computació d'alt rendiment emprats en aplicacions de ciència i enginyeria.

Micro-serveis, arquitectura basada en: model d'arquitectura d'aplicacions on cada component es considera un servei independent de la resta, especialment en termes de desplegament.

Runtime: entorn o subsistema encarregat de l'execució i gestió d'altres programes, p.e. dels contenidors Linux.

STEM: acrònim de *Science, Technology, Engineering and Mathematics.*

7 Bibliografia

7.1 Llibres i articles acadèmics

(Boe2015a) *Boettiger, Carl «***An introduction to Docker for reproducible research***».* ACM SIGOPS Operating Systems Review, - Special Issue on Repeatability and Sharing of Experimental Artifacts, vol. 49, núm. 1, pàg. 71-79. Gener 2015. *doi:10.1145/2723872.2723882*

(Gal2015a) *Gallagher, Scott «**Mastering Docker».*** Packt Publishing, Birmingham 2015

(McK2016a) *McKendrick, Russ «**Extending Docker».*** Packt Publishing, Birmingham 2016

(Mit2016a) MIT news (2016, 20 d'octubre). «***Making computers explain themselves***».
(Data de consulta 4 de març de 2017)
http://news.mit.edu/2016/making-computers-explain-themselves-machine-learning-1028

(Nat2016a) Nature eds. (2016, 25 de maig). «**Reality check on Reproducibility**». Nature 533, 437.
doi:10.1038/533437a

(Nic2016a) *Nickoloff, Jeff «***Docker in action***».* Manning Publications, New York 2016

7.2 Recursos multimèdia a Internet

(Gra2015a) *Graber, Stéphan (2015, setembre). «**An introduction to LXD, the container lighter-viso**r».* Container Camp. LDN Conference 2015
(Data de consulta: 24 de març de 2017)
https://youtu.be/yEr_EIZG0ZM

(Hyk2013a) *Hykes, Solomon (2013, març). «**The future of Linux Containers**».* PyCon Conference *2013*
(Data de consulta: 24 de març de 2017)
https://youtu.be/wW9CAH9nSLs

(Hyk2015a) *Hykes, Solomon (2015, juny). «***Introducing runC: a lightweight universal container runtime***».* Docker Blog
(Data de consulta: 24 de març de 2017)
https://blog.docker.com/2015/06/runc/

(Kit2013a) *Kitchin, John (2013, juliol). «***Emacs + org-mode + python in reproducible research***».* SciPy Conference 2013
(Data de consulta: 1 d'abril de 2017)
https://youtu.be/1-dUkyn_fZA

(Kni2017a) *Kniep, Christian (2017, febrer). «***Best Practices: State of Linux Containers***.».* HPC Advisory Council Stanford Conference 2017

(Data de consulta: 24 de març de 2017)
https://youtu.be/9Ck0okEzRp4

(Kut2016a) *Kutzer, Gregory (2016, maig). «***Linux containers with Singularity: introduction and demonstration***».*
(Data de consulta: 24 de març de 2017)
https://youtu.be/xuIQoth0r4E

(Pol2014a) *Polvi, Alex (2014, desembre). «***CoreOS is building a container runtime, rkt***».* CoreOs Blog
(Data de consulta: 13 d'abril de 2017)
https://coreos.com/blog/rocket.html

8 Annexos

8.1 GNU Free Documentation License

Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published

at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those

notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  YEAR  YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled "GNU
Free Documentation License".
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with … Texts." line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.