Publicació de serveis a xarxes comunitàries mitjançant contenidors

Estudiant: **Miguel Moreno Real**
Màster d'Enginyeria Informàtica

Consultor: **Félix Freitag**

Data Lliurament: 10/06/2016

FITXA DEL TREBALL FINAL

Títol del treball:	*Publicació de serveis a xarxes comunitàries mitjançant contenidors*
Nom de l'autor:	*Miguel Moreno Real*
Nom del consultor:	*Félix Freitag*
Data de lliurament (mm/aaaa):	*06/2016*
Àrea del Treball Final:	*Sistemes Distribuïts*
Titulació:	*Màster d'Enginyeria Informàtica*

Resum del Treball (màxim 250 paraules):

El treball es centra en analitzar diferents plataformes orientades a la creació de xarxes comunitàries o distribució de serveis per avaluar com es poden combinar per facilitar la publicació de serveis a la xarxa. Sempre tenint en compte la facilitat d'ús amb l'objectiu d'eliminar la barrera tecnològica d'accés a la comunitat.

D'entre els productes analitzats, s'ha identificat que tant Cloudy (distribució Linux per crear xarxes comunitàries) com Docker (Sistema per desplegar aplicacions basat en Linux containers) es poden integrar per potenciar les seves capacitats individuals sempre que es cobreixi el handicap que suposa l'administració de Docker des de la línia de comandes.

En aquest sentit, s'han estudiat diferents entorns d'usuari gràfics compatibles amb Cloudy, però no s'ha trobat cap que s'integri totalment. Per aquest motiu es decideix crear un codi per optimitzar la integració de Cloudy i Docker.

El resultat final és una nova secció al portal web d'administració de Cloudy des d'on administrar els contenidors Docker de forma integral i permetent a l'usuari publicar els serveis continguts a la comunitat.

Abstract (in English, 250 words or less):

This project focuses on analyse different platforms aimed at creating community networks or deploying services to evaluate how they can be combined to facilitate the publication of services at that networks. Keeping in mind the ease of use in order to remove the technological barrier to access to the community.

Some products were analysed, and it's identified that both Cloudy (Linux distribution for community networks) and Docker (Application deployment system based on Linux Containers) can be integrated to enhance their individual capacities if it's possible to reduce or eliminate the needs of command line management.

On that way, some graphical interfaces were tested over Cloudy, but wasn't founded one fully integrated. So it was decided to create a new application to enable the integration between Cloudy and Docker.

The final result is a new section inside the Cloudy administration web portal where manage Docker containers comprehensively and allowing the user to publish content services to the community.

Paraules clau (entre 4 i 8):

Cloudy Docker Publicació Microserveis Sandstorm Xarxes Comunitàries

Índex

Llista de figures

1. Introducció

1.1 Context i justificació del Treball

Cada vegada és més habitual que es creïn xarxes comunitàries per compartir diferents tipus de serveis entre usuaris. Alhora, la tipologia dels usuaris que s'afegeixen a aquestes està canviant i s'està obrint a usuaris que no tenen grans coneixements tècnics.

Per donar suport a aquestes "noves" xarxes han nascut distribucions Linux com Cloudy, que facilita la tasca de unió i administració de nous nodes tot facilitant la instal·lació i publicació de un petit conjunt de nous serveis.

D'altra banda, està apareixent Docker amb molta força al mercat, implementant la consolidada tecnologia de Linux Containers però oferint un valor afegit que està fent replantejar la distribució de serveis actual. Sense oblidar la plataforma Sandstorm, que permet publicar serveis personals a la xarxa des de un entorn de gestió totalment web.

El projecte es centra en l'anàlisi i avaluació de productes OpenSource que permeten als usuaris oferir microserveis en una comunitat pública o cloud.

En el estudi ens basarem en sistemes com Docker o Sandstorm, que permeten distribuir aplicacions fent servir tecnologia de contenidors. Mentre que per la publicació, analitzarem la plataforma Cloudy, on s'avaluarà la possibilitat d'integrar aquestes solucions de contenidors per oferir múltiples serveis.

Degut a la temàtica del projecte, aquest es dividirà en dos fases. A la primera es realitzarà un anàlisi dels productes esmentats i cerca d'altres complementaris que pugin aportar valor afegit definint els objectius de la segona fase, on s'implementarà una prova de concepte del producte proposat.

El resultat que es vol assolir és millorar les capacitats dels productes actuals orientat a la creació de xarxes comunitàries permetent a usuaris no tècnics oferir serveis diversos de forma senzilla.

1.2 Objectius del Treball

Els principals objectius del treball són els següents:

- Entendre l'abast i les possibilitats de Docker, Sandstorm i Cloudy i investigar si hi ha res més que es pugui integrar amb aquestes tecnologies aportant valor.
- Estudiar la integració de Docker sobre Cloudy.

- Estudiar la integració de Sandstorm sobre Ubuntu i/o Cloudy
- Estudiar la possible implementació de Sandstorm sobre un Docker Container per facilitar la seva distribució.
- Estudiar la possible integració de Sandstorm a Cloudy, directament o fent servir Docker
- Proposar una integració entre els productes analitzats.

1.3 Enfocament i mètode seguit

Es determina que existeixen múltiples iniciatives consolidades que aporten bones solucions als problemes que planteja aquest projecte, per aquest motiu es descarta crear un producte nou ja que l'objectiu no és crear una nova iniciativa sinó combinar adequadament les existents.

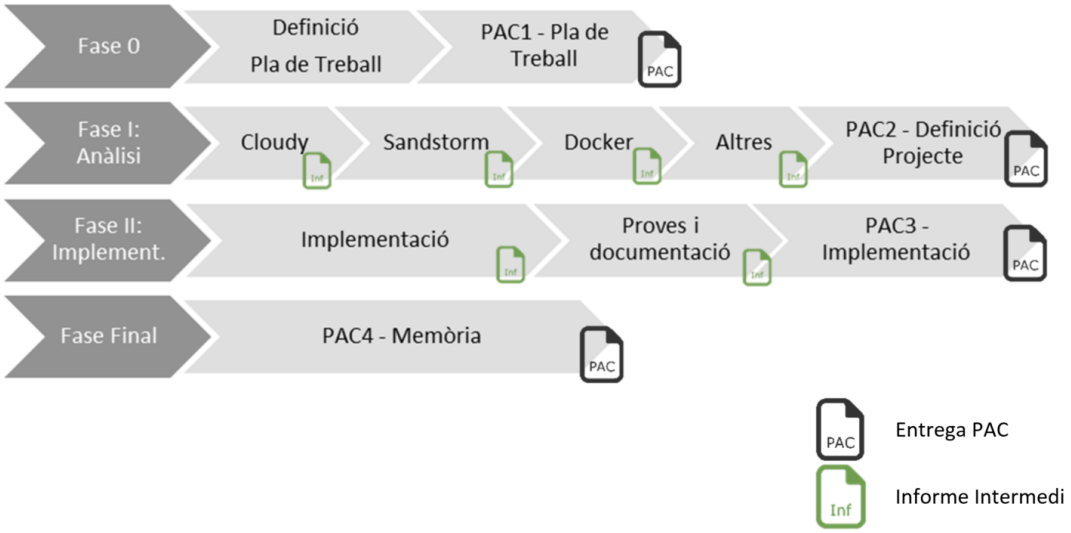
Per aquest motiu es planteja una primera fase d'anàlisi que les estudiarà i definirà l'ecosistema resultat i les integracions que s'hagin de implementar.

D'altra banda, es segueixen conceptes de metodologia àgil per a la fase d'implementació. La codificació s'ha de fer sobre un entorn desconegut on poden aparèixer multitud d'inconvenients. D'aquesta manera ens assegurem d'obtenir un producte funcional, que millorarà segons l'èxit que s'obtingui durant la codificació.

De seguir una planificació tancada durant la implementació, tindríem un alt risc d'acabar el projecte sense cap resultat visible.

1.4 Planificació del Treball

Planificació d'alt nivell identificant les fases i els entregables del projecte:

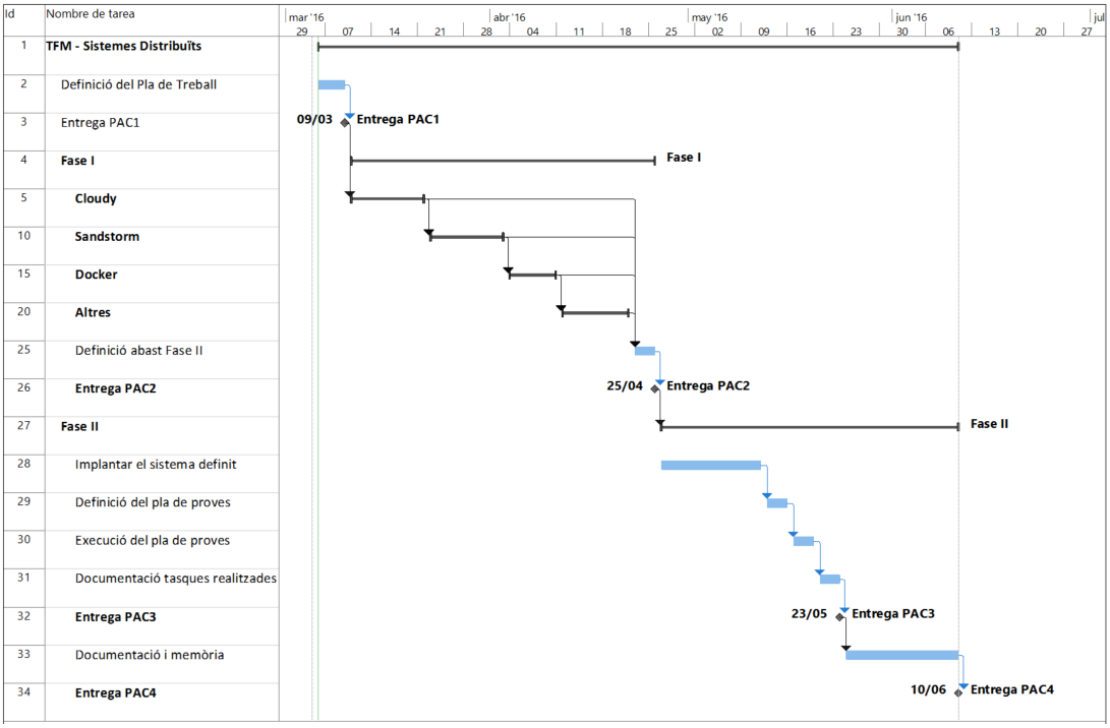


Imatge 1: Planificació d'alt nivell

A la Imatge 1 es pot veure com s'organitza el projecte. Consta d'una primera fase d'introducció on es defineix el treball a realitzar d'on s'obté com a resultat la primera PAC, seguidament comencen les dos fases

principals del projecte, les dues es tanquen amb l'entrega d'una PAC i diferents informes previs. Per últim es defineix una ultima fase, també de durada més curta, per redactar i organitzar els entregables finals.

Diagrama Gantt:



Imatge 2: Diagrama de Gantt

A la Imatge 2 podem veure un resum del diagrama de Gantt definit. Evidentment, el resultat és una programació en cascada, però permet quantificar visualment en pes de cada fase. El detall d'aquest diagrama el podem veure a la següent taula:

Detall Tasques:

Nom de la tasca	Duració	Començament	Fi
TFM - Sistemes Distribuïts	**300 hores**	**sáb 05/03/16**	**vie 10/06/16**
Definició del Pla de Treball	10 hores	sáb 05/03/16	mié 09/03/16
Entrega PAC1	**0 hores**	**mié 09/03/16**	**mié 09/03/16**
Fase I	**100 hores**	**jue 10/03/16**	**lun 25/04/16**
Cloudy	**25 hores**	**jue 10/03/16**	**lun 21/03/16**
Anàlisi i avaluació	3 hores	jue 10/03/16	vie 11/03/16
Instal·lació	4 hores	vie 11/03/16	sáb 12/03/16
Test unitari	10 hores	dom 13/03/16	jue 17/03/16
Conclusions	8 hores	vie 18/03/16	lun 21/03/16
Sandstorm	**25 hores**	**mar 22/03/16**	**sáb 02/04/16**
Anàlisi i avaluació	3 hores	mar 22/03/16	mié 23/03/16
Instal·lació	4 hores	mié 23/03/16	jue 24/03/16
Test unitari	10 hores	vie 25/03/16	mar 29/03/16

	Conclusions	8 hores	mié 30/03/16	sáb 02/04/16
	Docker	**20 hores**	**dom 03/04/16**	**dom 10/04/16**
	Anàlisi i avaluació	4 hores	dom 03/04/16	lun 04/04/16
	Instal·lació	4 hores	lun 04/04/16	mié 06/04/16
	Test unitari	8 hores	mié 06/04/16	vie 08/04/16
	Conclusions	4 hores	sáb 09/04/16	dom 10/04/16
	Altres	**30 hores**	**lun 11/04/16**	**jue 21/04/16**
	Anàlisi i avaluació	8 hores	lun 11/04/16	jue 14/04/16
	Instal·lació	2 hores	vie 15/04/16	vie 15/04/16
	Test unitari	4 hores	sáb 16/04/16	dom 17/04/16
	Conclusions	8 hores	lun 18/04/16	jue 21/04/16
	Definició abast Fase II	8 hores	vie 22/04/16	lun 25/04/16
PAC	**Entrega PAC2**	0 hores	lun 25/04/16	lun 25/04/16
	Fase II	**150 hores**	**mar 26/04/16**	**vie 10/06/16**
	Implantar el sistema definit	90 hores	mar 26/04/16	mié 11/05/16
	Definició del pla de proves	10 hores	jue 12/05/16	dom 15/05/16
	Execució del pla de proves	30 hores	lun 16/05/16	jue 19/05/16
	Documentació tasques realitzades	20 hores	vie 20/05/16	lun 23/05/16
PAC	**Entrega PAC3**	0 hores	lun 23/05/16	lun 23/05/16
	Documentació i memòria	**40 hores**	**mar 24/05/16**	**vie 10/06/16**
PAC	**Entrega PAC4**	0 hores	vie 10/06/16	vie 10/06/16

1.5 Breu sumari de productes obtinguts

A la fase d'anàlisi s'han obtingut tres informes (Cloudy, Docker i Sandstorm), on es detallen les característiques principals de cada tecnologia, quins són els seus punts forts i una proposta de com integrar-la amb la resta de productes analitzats.
Els tres informes es poden trobar annexes a aquest document.

A la fase d'implementació s'ha obtingut un codi que integra l'administració de Docker amb Cloudy i la seva publicació de serveis, als pròxims capítols es detalla el producte obtingut.

Disposem d'un portal a GitHub amb tot el codi generat juntament amb un script que aplica els canvis sobre qualsevol node Cloudy:
https://github.com/mmoreno803/TFM

S'ha creat una presentació del projecte en format PowerPoint que es troba com a adjunt a aquest treball i un vídeo resumint els punts claus de l'anàlisi i el producte final. El vídeo es pot trobar adjunt a aquest treball o als següents links de YouTube:

Català: https://youtu.be/2EZuQnv7rsM
Anglès: https://youtu.be/EwCcWs5-Hn0

Per últim, s'han creat dos vídeos addicionals com a demostració del producte:

DEMO Funcional
Català: https://youtu.be/2WDotGazSG4
Anglès: https://youtu.be/RJkaffhi5nM

DEMO de la facilitat de la instal·lació
Català: https://youtu.be/Xu_LxaqCFm4

1.6 Breu descripció dels altres capítols de la memòria

La resta del document s'organitza en tres capítols os es pot trobar la següent informació:

Productes analitzats (Capítol 2): Es resumeix el resultat de l'anàlisi de cada producte afegint un apartat de conclusions des d'un punt de vista global. A més, es descriuen els objectius detallats de la següent fase tenint en compte els objectius generals del projecte i les conclusions obtingudes a la fase anterior.

Implementació (Capítol 3): Es detalla la estructura de la fase i la seva evolució, tot afegint una secció final de conclusions.

Producte (Capítol 4): Es descriu el producte final obtingut.

2. Productes analitzats

L'anàlisi es centra en els productes Cloudy, Sandstorm i Docker. A continuació es detalla el resultat obtingut.

2.1 Anàlisi Cloudy

¿Què és Cloudy?

Segons el propis desenvolupadors del projecte, Cloudy és una distribució Linux creada per fomentar la adopció d'entorns cloud a les xarxes comunitàries. [1]

La principal motivació d'aquesta distribució és facilitar als usuaris la creació d'una xarxa distribuïda i descentralitzada alhora que facilita el desplegament de determinats serveis sobre la mateixa.

Primera sensació

Després d'instal·lar el primer node, s'aprecia que han aconseguit el seu propòsit. La instal·lació és simple i ràpida, amb poca configuració i sense requerir coneixements tècnics. Un cop instal·lats la resta de nodes i veient que aquests tenen visibilitat entre ells mitjançant una VPN autoconfigurada podem confirmar que Cloudy compleix allò que promet, la creació d'una xarxa comunitària distribuïda i descentralitzada sense necessitat de coneixements tècnics.

Fent una ullada al portal web d'administració, veiem la capacitat que tenim per desplegar diferents serveis sobre els nodes de la xarxa només fent un clic.

Entorn de test

Per poder fer un anàlisi amb més profunditat d'aquest sistema s'han creat dos entorns.

Per al primer entorn fem servir la ISO proporcionada al web de Cloudy (Debian 7.x customitzat) creant una granja de 7 nodes. Amb aquest entorn podem extreure les primeres sensacions de la plataforma, veient la facilitat de la instal·lació i com podem desplegar sistemes com un Filesystem distribuït sense tenir o aplicar coneixements tècnics.

Però aquesta distribució està basada en una distribució Debian de 32-bits. Tenint en compte que més endavant volem fer proves amb contenidors Docker (basats en 64-bits), continuarem les proves amb un entorn amb aquesta arquitectura.

Això és possible gracies a que els desenvolupadors de Cloudy han creat un procediment que anomenen “Cloudynitzar”, que permet desplegar la capa de SW Cloudy sobre qualsevol Debian. Ho aprofitarem per crear una xarxa Cloudy de 64-bits.
Per al segon entorn fem servir l’última versió publicada al web de Debian (64-bits) creant una granja de 5 nodes i procedint a “Cloudynitzar-los”

Observacions

Cloudy és una plataforma molt potent que simplifica extraordinàriament el desplegament de clústers distribuïts sense tenir un ampli coneixement tècnic.

Com els mateixos membres de la comunitat ja ens avançaven, eliminar aquesta barrera tecnològica permet que més usuaris s’animin a fer-la servir.

Cloudy integra un sèrie de serveis que es poden instal·lar i gestionar (de forma bàsica) des de l’entorn web. Reforçant la idea de simplificar l’accés a la gestió.

D’altra banda, ens trobem que el “catàleg” de serveis disponibles per gestionar des del portal web és certament limitat, obligant a fer servir la operació manual si vols treure partit del clúster o tens necessitats específiques.

Conclusions

Cloudy és un bon punt de partida per crear clústers distribuïts ja que simplifica la configuració inicial. En cas d’iniciar un projecte amb aquesta necessitat, és una plataforma a tenir en compte.

Te punts de millora incrementant els serveis que es poden instal·lar des de la gestió web i millorant la gestió que es pot fer sense accedir a la línia de comandes.

Per exemple, podem instal·lar Docker des de el portal web, però res més. Tota la gestió es fa a la línia de comandes. En aquest sentit, Cloudy no aporta gaire valor afegit, ja que si l’usuari està obligat a fer qualsevol operació sobre Docker a la línia de comandes, segur que estarà capacitat per realitzar també la instal·lació.

En qualsevol cas, un cop verificat que Docker funciona sobre la plataforma Cloudy, obre un gran ventall d’oportunitats.

L’informe complert afegeix major nivell de detall, el procés d’Instal·lació del sistema operatiu i el detall de les proves realitzades. Es troba a l’**Annex I: Informe avaluació sistema Cloudy**

2.2 Anàlisi Sandstorm

¿Què és Sandstorm?

Sandstorm és un software que s'instal·la sobre un servidor amb sistema operatiu Linux x86 i permet publicar diferents aplicacions col·laboratives de forma fàcil i efectiva. [2]

Sandstorm proporciona un portal d'aplicacions que permet desplegar apps al el servidor sense coneixements tècnics i permet la ràpida gestió de la compartició de recursos entre usuaris assegurant la gestió d'accés.

El sistema es pot fer servir en cloud, fent servir el servei SaaS de Sandstorm.io o be instal·lant-ho a la teva pròpia infraestructura. Aquesta última opció serà l'escollida per dur a terme aquest anàlisi, tenint en compte que l'objectiu és instal·lar-ho sobre Cloudy.

Proves a realitzar

- Verificar la compatibilitat de Sandstorm sobre l'ecosistema de Cloudy
- Analitzar i verificar les capacitats del sistema Sandstorm.
- Estudiar possibles cassos d'ús de Sandstorm aprofitant la plataforma Cloudy

Entorn de test

Les proves a realitzar es centraran en descobrir i testejar el potencial del sistema alhora que es realitzen proves de compatibilitat sobre la plataforma Cloudy.

Per tant, l'entorn de test on es realitzaran les proves és un clúster Cloudy. S'aprofitarà l'entorn de test existent de la plataforma Cloudy de 64-bit

Observacions

Per disseny, les dades s'emmagatzemen només en local al host os resideix la APP instal·lada. La informació no és accessible des de altres hosts, i això impedeix que les aplicacions pugin ser distribuïdes.

Com Sandstorm planteja l'escalabilitat, és tenint en compte que una mateixa APP la pots instal·lar a tots els nodes de la teva granja, i llavors manualment, repartir l'ús d'aquesta APP entre els nodes.

Per exemple, si volem escalar la APP de compartició de fitxers, podem crear carpetes diferents a diferents nodes fent servir la mateixa APP. Però mai seria la mateixa instància i cada carpeta tindria una URL d'accés.

La versió gratuïta de Sandstorm no permet l'autenticació d'usuaris amb el teu propi directori o fent servir SAML contra un Identity Provider personalitzat. Únicament permet la gestió d'accés segura federant amb Google o GitHub. Tot i que és una clara limitació, totes dues opcions són suficientment bones com per que això no sigui un problema.

Conclusions

Sandstorm no permet el desplegament de aplicacions distribuïdes i per tant no té capacitat per treure profit de Cloudy.

D'altra banda, permet el desplegament de aplicacions col·laboratives de forma molt fàcil i eficaç. I això podria donar un valor afegit als nodes de Cloudy.

Es a dir, si les nostres necessitats són crear un Filesystem, i disposem de la plataforma Cloudy, té més sentit implementar Tahoe-LAFS que una App en Sandstorm, però ens aporta la possibilitat de aprofitar la nostra infraestructura amb aplicacions que no requereixin alta disponibilitat.
Per exemple un chat, un fòrum, un panell kanban col·laboratiu, ...

L'informe complert afegeix major nivell de detall, el procés d'Instal·lació del sistema i el detall de les proves realitzades. Es troba a l'**Annex II: Informe avaluació sistema Sandstorm**

2.3 Anàlisi Docker

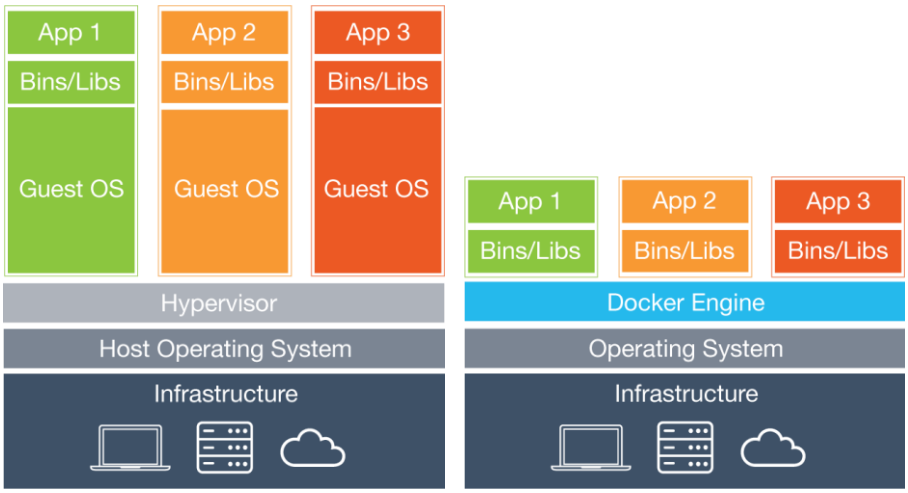
¿Què és Docker?

"Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de Virtualización a nivel de sistema operativo en Linux." [3]
O dit d'una altra manera, Docker ens permet virtualitzar aplicacions dins de contenidors de manera que aquests siguin autònoms i es puguin executar sobre qualsevol plataforma.

El contenidors contenen tot el que necessita la aplicació per funcionar: codi, llibreries, Filesystem, etc, de manera que garanteix el mateix funcionament independentment del host.

Un container Docker no és una màquina virtual, existeix una diferencia que fa que aquest sistema sigui molt interessant. Els contenidors no tenen sistema operatiu, reduint el consum de recursos del host i la complexitat de l'entorn. Per aconseguir-ho, es substitueix l'habitual hypervisor pel Docker Engine.

La següent imatge publicada a la web de Docker clarifica aquest concepte:



Imatge 3: Stack Docker [4]

A la Imatge 3 es compara la pila d'un sistema de virtualització tradicional en front de la pila de Docker. Existeixen dos diferències bàsiques: La primera és el canvi de l'hypervisor pel Docker Engine, això fa possible la segona diferència, que és l'absència a Docker de Sistema operatiu per a cada màquina virtual / Contenidor.

Com a primera conclusió, es pot extreure que Docker podrà executar aplicacions de forma més eficient que qualsevol hypervisor, però també es preveu que la creació d'aquestes aplicacions serà més complexa.

Mentre que en una màquina virtual simplement despleguem aplicacions com si de un entorn físic/tradicional es tractés, amb Docker haurem de construir aquest software per que es pugui interpretar pel seu Engine.

Un altre aspecte a comprovar serà si realment l'Engine de Docker serà més eficient executant contenidors que un hypervisor executant el Sistema Operatiu més l'aplicació.

Docker al mon

Veient com evoluciona aquest sistema, que cada vegada té més referències al mercat i casos d'èxit publicats, a més de la increïble acollida que està tenint per tots els grans fabricants de hardware i software del mercat, podríem avançar una resposta al dubte sobre la seva eficiència, i és que segurament ho sigui.

Referències de Docker:

- The New York Times: https://www.docker.com/customers/new-york-times-delivers-continuous-integration-pipeline-docker
- Amadeus: https://www.docker.com/customers/docker-delivers-greater-mobility-and-better-security-amadeus

- Spotify:
  https://www.docker.com/customers/continuous-delivery-spotify
- I molts altres:
  https://www.docker.com/customers

Esdeveniments on es fa referència a Docker:

- Microsoft dotNet 2016: Running ASP 5 with Docker
  https://blogs.msdn.microsoft.com/webdev/2015/01/14/running-asp-net-5-applications-in-linux-containers-with-docker/
- vmworld 2014 & 2015: VMware and Docker – Better Together
  https://cto.vmware.com/vmware-docker-better-together/
- ciscolive 2016: Multi-datacenter Docker Container Manager
  https://www.ciscolive.com/connect/sessionDetail.ww?SESSION_ID=4736

Aquest són només alguns exemples, però el fet de que Microsoft estigui potenciant l'ús de l'eina, vmware es faci ressò tenint en compte que es tracta de una competència natural i que fins i tot Cisco en parli, són motius per pensar que Docker pot aportar molt al mon de les TI en un curt termini.

¿Què aporta Docker?

Docker no només aporta la possibilitat de crear containers amb aplicacions, disposa d'altres funcionalitats que complementen i fan més atractiu al producte:

- Un cop creat el container aquest es pot desplegar ràpidament sobre nodes amb l'Engine de Docker, tant amb l'objectiu de agilitzar les posades en producció com per escalar la plataforma
- Al estar auto-contingut, elimina dependències i inconsistències.
- Facilitat per compartir contenidors gràcies al Docker Hub. Un portal on es poden publicar contenidors amb tot tipus d'aplicacions i sistemes
- L'Engine de Docker manté actualitzada l'aplicació executant sempre l'ultima versió publicada, sense interacció de l'administrador.

Proves a realitzar

- Verificar la compatibilitat de Docker sobre l'ecosistema de Cloudy
- Analitzar i verificar les capacitats del sistema Docker.
- Estudiar possibles cassos d'ús de Docker aprofitant la plataforma Cloudy
- Estudiar eines gràfiques per la gestió de Docker

Entorn de test

Les proves a realitzar es centraran en descobrir i testejar el potencial del sistema alhora que es realitzen proves de compatibilitat sobre la plataforma Cloudy.

Per tant, l'entorn de test on es realitzaran les proves en un clúster Cloudy. S'aprofitarà l'entorn de test existent de la plataforma Cloudy de 64-bit

Conclusions

Definitivament, Docker pot aportar a Cloudy una gran flexibilitat per desplegar nous serveis. Però, per que sigui realment efectiva en un cloud col·laboratiu, és necessari implementar una eina de gestió gràfica per Docker.

A la presentació de Docker s'explica que un contenidor pot executar-se a qualsevol host gracies a que és totalment autònom alhora que estalvia l'ús de recursos. Però amb les proves que hem fet veiem que això no és sempre així, i un contenidor pot necessitar adaptacions depenent del host.

Tot i això, Docker continua sent una bona opció per distribuir aplicacions. De les aplicacions provades, veiem que ownCloud encaixa perfectament amb la filosofia de Cloudy i seria una eina molt interessant. D'altra banda, podem explotar la capacitat dels contenidors per "autodestruir-se" amb contenidors com Ubuntu per executar proves sobre entorns nets.

L'informe complert afegeix major nivell de detall, el procés d'Instal·lació del sistema, el detall de les proves realitzades i els interfases gràfics avaluats. Es troba a l'**Annex III: Informe avaluació sistema Docker**

2.4 Definició d'objectius:

Després de l'anàlisi realitzat, les opcions de millora detectades i un primer intercanvi d'impressions amb desenvolupadors de la plataforma Cloudy, es defineixen els següents objectius per al TFM, ordenats per prioritat:

- Crear un interface a la plataforma Cloudy entre Docker i Serf per permetre als usuaris publicar a la comunitat els serveis que es despleguin a partir d'un contenidor.
- Implementar una eina de gestió web per gestionar Docker sobre la plataforma Cloudy:
    - Implementar Shipyard com a eina de gestió
    - Implementar un interface web compatible amb LXD
- Permetre desplegar la plataforma Sandstorm com un servei més de Cloudy, integrant la publicació dels serveis a través de Serf.

Aquests objectius es poden veure alterats durant el transcurs de la fase d'implementació ja que es tracten d'objectius dels que no tenim garantia que siguin totalment factibles. A la fase d'anàlisi s'ha identificat la necessitat i s'han provat superficialment alguns productes, però no podem saber si seran totalment compatibles amb Cloudy.

D’altra banda, es planifica una sessió conjunta amb desenvolupadors de la plataforma Cloudy per orientar aquests objectius cap a maximitzar el benefici obtingut per la plataforma al finalitzar aquest TFM.
Tenint aquest punt en compte, es podrien resumir els objectius del TFM en un de sol:

- Enriquir la plataforma Cloudy abordant les opcions de millora detectades.

3. Implementació

En aquest apartat es descriu com s'ha estructurat la fase, com ha evolucionat amb el pas dels Sprints i quines conclusions podem extreure del conjunt del procés

3.1 Estructura de la fase

Tenint en compte que el projecte planteja objectius dels que és difícil valorar el seu esforç, es busca definir fites incrementals intentant maximitzar les funcionalitats implementades a final de projecte sense comprometre l'obtenció de resultats. Per tant, per aquesta fase d'implementació es farà servir una metodologia àgil.

Degut al reduït grup de treball (jo) no té sentit aplicar una metodologia àgil de forma gaire estricta, però ens pot ajudar a definir els objectius i guiar en el procés d'implementació.

En aquesta línia, es defineixen les histories d'usuari detectades, que ajudaran a definir els objectius incrementals a implementar.
Per últim, s'incorporaran les tasques resultats a un panell kanban per fer seguiment de les mateixes. Es farà servir la App *Kanban Panel* que es va provar durant l'anàlisi de Sandstrom i que ja tenim instal·lat.

Aquestes tasques s'executaran en diferents iteracions (Sprints) d'una durada aproximada d'una setmana. Cada Sprint finalitzarà amb un informe de seguiment i la entrega d'un producte completament funcional que implementi els requeriments definits.

Històries d'usuari

Relat d'històries d'usuari a partir dels requeriments detectats i les necessitats a curt.

ID	Descripció
US01	Vull saber els contenidors que tinc de Docker de forma fàcil i visual
US02	Vull poder administrar els contenidors que tinc instal·lats
US03	Vull poder instal·lar nous contenidors d'una llista de recomanacions
US04	Vull poder instal·lar qualsevol contenidor al meu node
US05	Vull poder publicar els serveis dels meus contenidors a la comunitat
US06	Vull poder decidir quins serveis dels meus contenidors publico
US07	Vull poder decidir si publico qualsevol servei que s'executa a Docker
US08	Vull poder instal·lar SandStorm al meu node
US09	Vull poder publicar els serveis de Sandstorm a la comunitat

La llista d'històries d'usuari serà dinàmica en el transcurs de la fase d'implementació. Si en acabar una història es detecta una nova necessitat, s'inclourà a la llista i es generaran les tasques associades.

Fites

A continuació es defineixen les fites associades a les històries d'usuari. Aquestes s'ordenen tenint en compte que el producte final sigui evolutiu i proporcioni noves funcionalitats en acabar cada fase.

L'aparició de noves històries d'usuari poden provocar un canvi en l'ordre d'execució de la resta de tasques definides.

Tenint en compte que no es tenen coneixements suficients per poder quantificar el temps de implementació de cada tasca, s'utilitzarà com a mesura un grau de complexitat definit entre 1 i 5, sent 5 el grau màxim.

ID Tasca	ID US	Descripció Tasca	Complexitat
Tsk01	US01	Identificar com la web de Docker pot interactuar amb el host per rebre informació d'estat de Docker i parsejar-la	2
Tsk02	US05	Configurar Serf en Cloudy per descobrir serveis publicats per altres nodes	1
Tsk03	US05	Descobrir com Cloudy publica serveis a través de Serf	3
Tsk04	US01, US05	Seguir la guia per instal·lar i publicar el software pastecat amb l'objectiu d'entendre tot el procés	4
Tsk05	US01	Obtenir informació dels contenidors instal·lats i mostrar-ho de forma estructurada a la web	3
Tsk06	US02	Afegir un boto per a cada contenidor detectat per parar-ho o arrencar-ho depenent del seu estat	3
Tsk07	US02	Afegir un boto per a cada contenidor detectat per eliminar-lo si està parat	1
Tsk08	US02	Afegir un botó amb un link cap a la plana d'administració del contenidor	1
Tsk09	US05	Afegir un boto per a cada contenidor detectat per publicar-lo	2
Tsk10	US06	Crear una taula que guardi si hem decidit publicar un determinat contenidor	3
Tsk11	US03	Crear una taula de contenidors recomanats amb un botó associat que descarregui i inicií cada contenidor	3
Tsk12	US06	Descobrir com Cloudy detecta canvis en els serveis publicats al node local	4
Tsk13	US06	Si s'arranca un contenidor que hem decidit que volem publicar, publicar-ho directament	4
Tsk14	US06	Si es para o cau un contenidor que hem decidit que volem publicar, despublicar-ho directament	3
Tsk15	US07	Crear un interface per poder afegir contenidors a la llista de contenidors recomanats	3
Tsk16	US08	Crear un nou menú a la web d'administració de Cloudy que permeti instal·lar Sandstorm al node	4
Tsk17	US08	Afegir al nou menú el link d'accés a l'eina de gestió de Sandstorm	1
Tsk18	US09	Investigar com podem detectar les apps que Sandstorm té creades	5

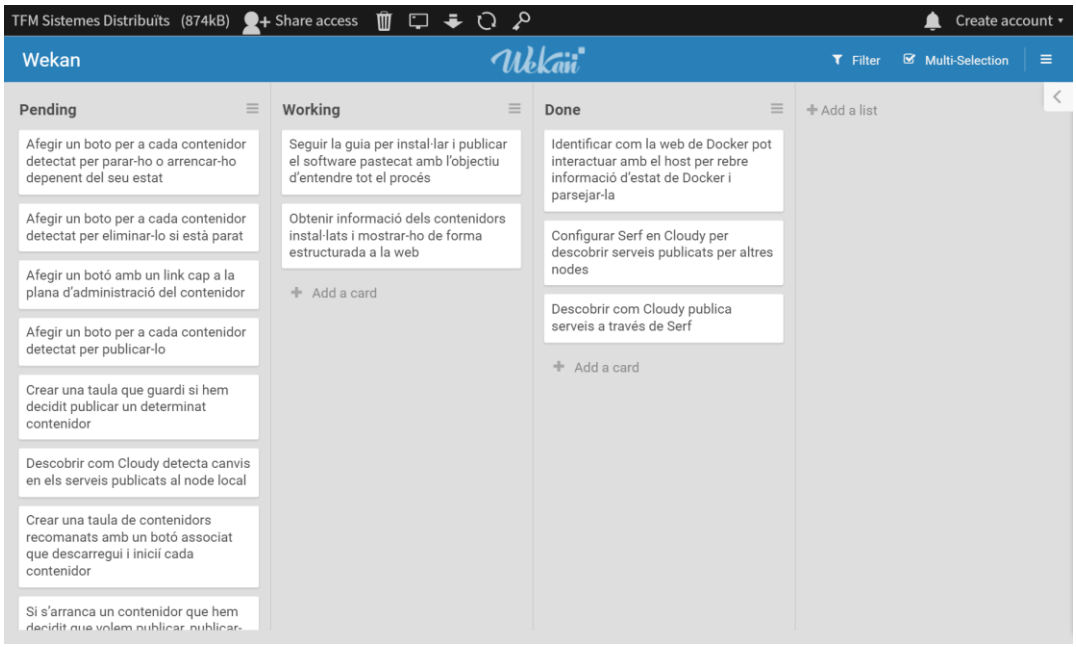
Tsk19	US09	Crear un interface que permeti publicar els serveis de Sandstorm de forma individual creat una taula que emmagatzemi els serveis a publicar	4
Tsk20	US09	Detectar canvis en els serveis de Sandstorm publicats per gestionar si es continuen publicant o no	5

L'objectiu d'aquesta metodologia no és implementar totes les tasques definides en aquesta taula, sinó que pretén forçar l'avanç del projecte amb la dedicació disponible, garantint que a cada tasca implementada obtenim un producte final evolucionat amb noves funcionalitats.

En el transcurs del projecte poden apareixen noves necessitats, es prioritzarà obtenir un millor producte ajustat a les necessitats que finalitzar aquesta llista de tasques.

Eina de gestió

Com es comentava a la introducció, es fa servir un panell kanban desplegat amb Sandstorm sobre un dels nodes del clúster de test.
http://cloudy64-2.home64:6080



Imatge 4: Panell kanban - gestió del projecte

A la Imatge 4 es pot veure una captura del panell amb el que es gestionarà aquesta fase del projecte, Mantenint l'evolució dels elements implementats i la possibilitat d'incloure de nous.

Conclusió

Seguint aquesta metodologia i definint aquest objectius s'intenta donar granularitat als objectius definits a la PAC2 i confirmats amb els col·laboradors de guifi.net, que eren els següents:

1. Crear un interface a la plataforma Cloudy entre Docker i Serf per permetre als usuaris publicar a la comunitat els serveis que es despleguin a partir d'un contenidor.
2. Implementar una eina de gestió web per gestionar Docker sobre la plataforma Cloudy:
   2.1. Implementar Shipyard com a eina de gestió
   2.2. Implementar un interface web compatible amb LXD
3. Permetre desplegar la plataforma Sandstorm com un servei més de Cloudy, integrant la publicació dels serveis a través de Serf.

3.2 Evolució dels Sprints

El detall sobre com ha evolucionat la implementació d'aquestes tasques i com s'han organitzat es pot veure en detall als informes annexes de cada Sprint:

Annex IV: Informe Sprint 1
Annex V: Informe Sprint 2
Annex VI: Informe Sprint 3

Al finalitzar el segon Sprint de la implementació es va detectar que Docker i el nou portal en desenvolupament podien donar una funcionalitat similar a la que aporta Sandstorm, per aquest motiu es van descartar les històries d'usuari US08 i US09 i les seves tasques associades amb l'objectiu d'aprofundir en noves necessitats detectades per a Docker.

Les noves tasques identificades es centren en analitzar en quins contenidors Docker donen a l'usuari una funcionalitat similar a Sandstorm, en millorar la interface d'usuari i en facilitar el desplegament del nou portal.

Estat final de fites

ID Tasca	Descripció Tasca	Estat final
Tsk01	Identificar com la web de Docker pot interactuar amb el host per rebre informació d'estat de Docker i parsejar-la	**Completada**
Tsk02	Configurar Serf en Cloudy per descobrir serveis publicats per altres nodes	**Completada**
Tsk03	Descobrir com Cloudy publica serveis a través de Serf	**Completada**
Tsk04	Seguir la guia per instal·lar i publicar el software pastecat amb l'objectiu d'entendre tot el procés	**Completada**
Tsk05	Obtenir informació dels contenidors instal·lats i mostrar-ho de forma estructurada a la web	**Completada**
Tsk06	Afegir un boto per a cada contenidor detectat per parar-ho o arrencar-ho depenent del seu estat	**Completada**
Tsk07	Afegir un boto per a cada contenidor detectat per eliminar-lo si està parat	**Completada**
Tsk08	Afegir un botó amb un link cap a la plana	**Completada**

	d'administració del contenidor	
Tsk09	Afegir un boto per a cada contenidor detectat per publicar-lo	**Completada**
Tsk10	Crear una taula que guardi si hem decidit publicar un determinat contenidor	**Completada**
Tsk11	Crear una taula de contenidors recomanats amb un botó associat que descarregui i inicií cada contenidor	**Completada**
Tsk12	Descobrir com Cloudy detecta canvis en els serveis publicats al node local	**Completada**
Tsk13	Si s'arranca un contenidor que hem decidit que volem publicar, publicar-ho directament	**Completada**
Tsk14	Si es para o cau un contenidor que hem decidit que volem publicar, despublicar-ho directament	**Completada**
Tsk15	Crear un interface per poder afegir contenidors a la llista de contenidors recomanats	**Completada**
Tsk16	Crear un nou menú a la web d'administració de Cloudy que permeti instal·lar Sandstorm al node	Descartada
Tsk17	Afegir al nou menú el link d'accés a l'eina de gestió de Sandstorm	Descartada
Tsk18	Investigar com podem detectar les apps que Sandstorm té creades	Descartada
Tsk19	Crear un interface que permeti publicar els serveis de Sandstorm de forma individual creat una taula que emmagatzemi els serveis a publicar	Descartada
Tsk20	Detectar canvis en els serveis de Sandstorm publicats per gestionar si es continuen publicant o no	Descartada
Tsk21 **(Nova)**	Investigar si existeixen contenidors Docker que cobreixin les funcionalitats proporcionades per les Apps de Sandstorm	**Completada**
Tsk22 **(Nova)**	Dissenyar el interfície d'usuari	**Completada**
Tsk23 **(Nova)**	Crear un script d'instal·lació del nou GUI	**Completada**

3.3 Estat de la implementació final

Tot i que els objectius s'han assolit, el producte final encara necessita de proves funcionals exhaustives per passar a producció. Fins a aquest punt s'han fet proves unitàries per assegurar el funcionament de cada component de forma individual.

D'altra banda, tal com es detalla a la secció treball futur del capítol de Conclusions, s'han identificat diferents millores que es consideren importats per millorar substancialment l'experiència d'usuari.

3.4 Conclusions

El balanç final ha estat positiu on:

- S'han realitzat totes les tasques previstes entorn a Cloudy i Docker
- S'han implantat noves funcionalitats no planificades des de l'inici i que s'han identificat en el transcurs dels Sprints.

- Al segon Sprint es van descartar les històries d'usuari referents a Sandstorm ja que les funcionalitats desenvolupades per Docker donen una funcionalitat similar.

Per tant, el resultat final s'ajusta a la planificació i el producte obtingut cobreix les necessitats requerides.

4. Producte final

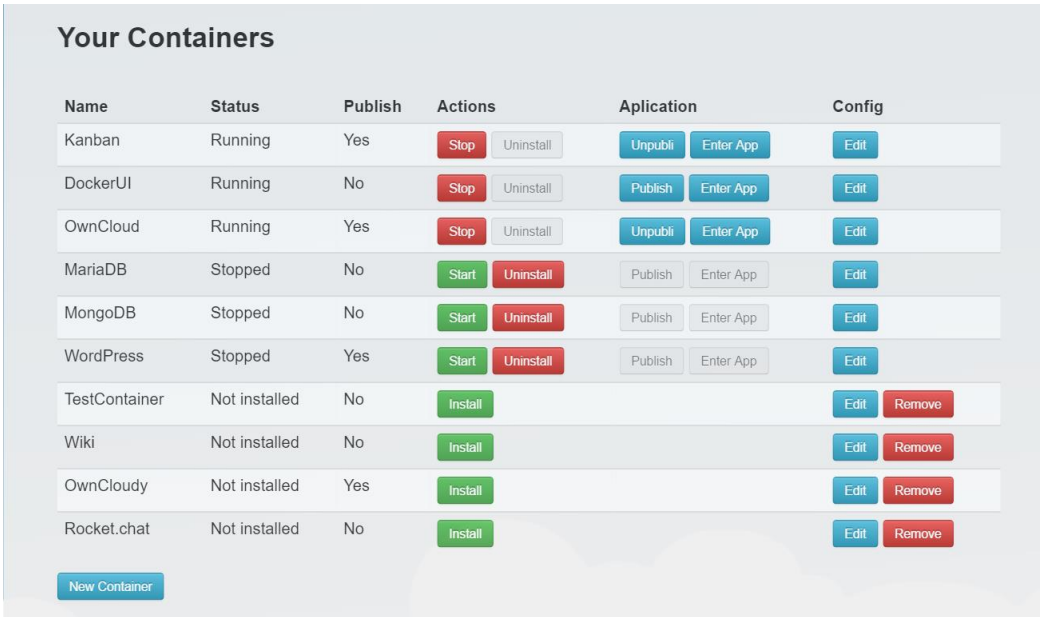
4.1 Descripció del producte

En aquest nou entorn de gestió s'ha creat un nou concepte, el contenidor gestionat.

El contenidor gestionat és la definició i configuració dels paràmetres que un contenidor necessita per ser instal·lat i operat a Cloudy amb el nou entorn de gestió. Per tant, un contenidor gestionat no ha d'estar instal·lat necessàriament, sinó que pot estar simplement definit.

El portal de gestió inclou, per defecte, una sèrie de contenidors gestionats, cap d'ells està instal·lat d'inici.

El resultat final de la implementació es pot veure representat en dues imatges:
Nou panell d'administració de contenidors Docker:

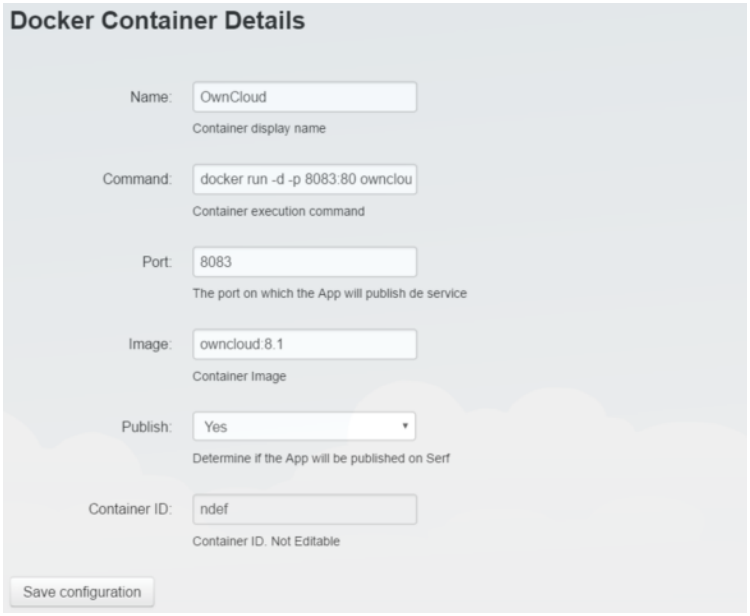


Imatge 5: Panell de Control

A la Imatge 5 es pot veure com l'usuari disposa d'un panell de control on apareixen tots els contenidors gestionats definits a Cloudy. Els contenidors es poden Instal·lar, arrencar, parar i desinstal·lar des d'aquesta mateixa vista.

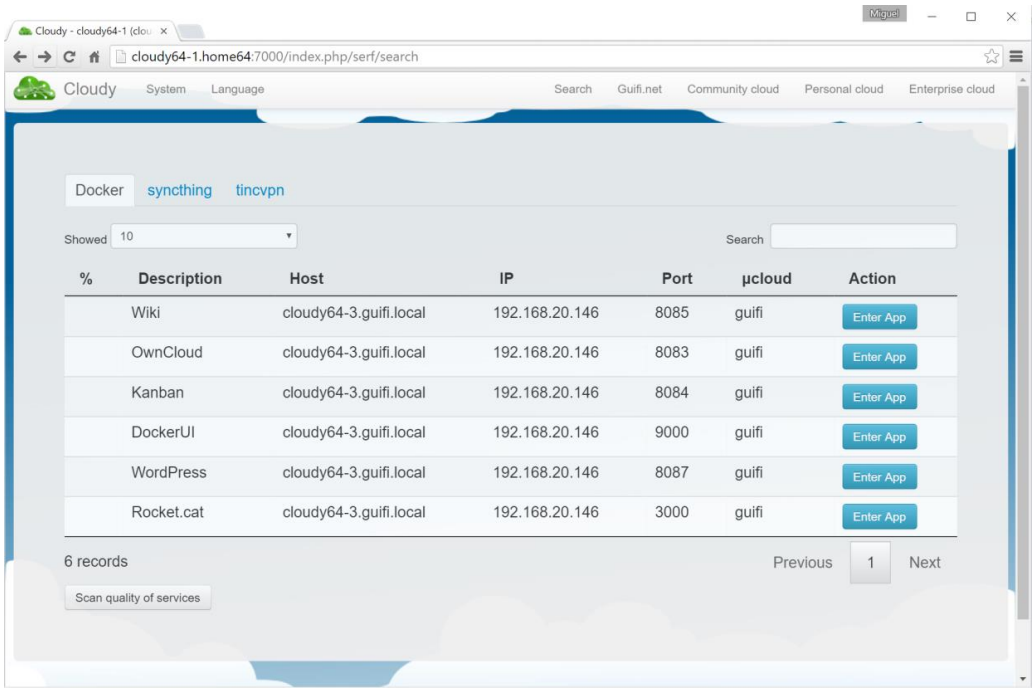
A més, l'usuari pot decidir si publica o no el servei a Serf i se li ofereix l'opció d'accedir a l'aplicació.

Per últim, a la Imatge 6, veiem com es pot editar la configuració dels contenidors gestionats, crear de nous i eliminar els existents. Tot des de un entorn gràfic i sense necessitat de coneixements tècnics.



Imatge 6: Configuració de contenidors gestionats

Vista de la publicació de serveis via Serf:



Imatge 7: Publicació de serveis amb Serf

A la Imatge 7 tenim la vista on es pot observar com tots els serveis publicats apareixen en un altre node de la xarxa. Qualsevol usuari* pot accedir a les aplicacions publicades fent clic al botó d'accés associat.

**Tots els usuaris podran accedir a la URL, la gestió de permisos l'ha de fer cada aplicació.*

A més de la part més visual, cal remarcar les següents funcionalitats:

Es pot instal·lat tot el nou software sobre un node Cloudy a partir d'un script automatitzat. Eliminant així una barrera d'accés a usuaris. Tot el codi i el script d'instal·lació del nou programari està ubicat a un repositori públic de GitHub.

Els serveis publicats estan sota permanent monitorització, si un servei s'atura, aquest es despublica automàticament. Millorant així la monitorització de Serf que només comprova l'estat del servidor.

Es poden definir nous contenidors a través del formulari o executant importacions de un o més fitxers de configuració facilitant la tasca a l'usuari.

4.2 Millores obtingudes amb el nou producte

En el moment d'iniciar el projecte, Cloudy es caracteritzava per ser una distribució Linux molt potent per a la creació de xarxes comunitàries, facilitant al màxim les tasques d'usuari, eliminant la barrera tecnològica d'accés per a nous col·laboradors.

Aquesta facilitat en la creació de nous nodes a la xarxa contrastava amb la seva capacitat per distribuir nous serveis, ja que per defecte es pre-configuren només serveis transversals a la plataforma, importants pel sistema, però limitats en quant a la publicació de nous serveis.

D'aquesta manera, els usuaris de Cloudy que volien publicar serveis personals requerien de coneixements avançats per poder fer-ho.

Amb el nou portal d'administració de Docker s'ha habilitat el desplegament d'aplicacions de forma fàcil i efectiva integrant la seva publicació a la comunitat amb un sol clic.

Amb aquesta aportació transformem una distribució Linux per xarxes comunitàries pensada per donar suport a la mateixa xarxa, en una distribució que a més permet a l'usuari desplegar els serveis que vulgui sense necessitat de coneixements tècnics.

Posem com a exemple a un usuari, sense coneixements tècnics, que disposa d'un HW qualsevol, pot crear un node Cloudy, desplegar un servei de Dropbox* privat i publicar-lo per altres usuaris en menys de 17 minuts (veure vídeos de DEMO)

*La referencia a Dropbox vol definir un sistema d'emmagatzemament i compartició de fitxers al Cloud. En el cas de la demo s'aconsegueix amb l'aplicació OwnCloud.

4.3 Take-up guifi.net

Tal com s'ha comentat anteriorment, a l'inici de la fase d'implementació es va mantenir una reunió amb l'equip de guifi.net per analitzar amb ells la millor manera d'orientar el nou producte per maximitzar la utilitat del resultat final.

Al finalitzar la fase d'implementació, es va compartir el codi generat (a través del portal GitHub) amb l'equip de guifi.net per al seu anàlisi, obtenint com a resultat la creació d'una nova branca de desenvolupament sobre la distribució Cloudy.

L'equip de guifi.net ha adaptat el codi i ha realitzat proves funcionals per garantir la estabilitat en conjunt amb la resta de mòduls i actualment el codi ja forma part de la branca màster de Cloudy.

Això vol dir que actualment, quan qualsevol usuari "Cloudynitza" un nou Debian, aquest ja incorpora les noves funcionalitats creades en aquest treball final de Màster.

Relacionat amb això, s'ha publicat la noticia al web de la comunitat Cloudy [5]:

Post del 09/06/2016:
http://cloudy.community/es/news/

Detall de la publicació:
http://cloudy.community/en/container-management-panel-added-to-cloudy/

A més, està previst que aquesta nova funcionalitat es presenti al SAX (esdeveniment anual de guifi.net) el proper 18 de Juny a Barcelona.

5. Conclusions

Conclusions del treball:

- S'han analitzat diferents productes que poden donar solució a la publicació de microserveis en xarxes comunitàries.
- S'ha identificat com aquests productes poden interactuar entre ells per incrementar les seves possibilitats.
- S'ha implementat un codi per possibilitar la integració complerta dels sistemes d'una manera amigable a l'usuari.
- S'ha integrat la gestió web de contenidors Docker amb la publicació de serveis de Cloudy.
- S'ha implementat un script per facilitar la instal·lació en dos passos.

Assoliment d'objectius:
Els objectius principals s'han assolit satisfactòriament. A la fase d'anàlisi es va identificar una necessitat a cobrir i aquesta s'ha implementat de forma pràctica.

Si be es cert que el codi implementat no és totalment funcional i necessita de una revisió que impliqui tests d'usuari exhaustius, es va prioritzar mostrar les capacitats de la integració més que obtenir un producte més limitat i preparat per entorns productius.

Seguiment de la planificació:
El seguiment de la planificació ha estat satisfactori i s'ha demostrat que seguir una metodologia àgil era la millor opció. Durant la fase d'implementació van aparèixer imprevistos que es van poder corregir durant els Sprints i que ha permès disposar del producte generat.

Treball futur:
Com es comentava anteriorment, el producte necessita passar tests d'usuari intensius. Actualment només s'han fet tests unitaris d'alt nivell per garantir un correcte funcionament per la proba de concepte. Un cop garantida la consistència del producte, aquest estarà preparat per posar-se en producció.

Com a opcions de millora, s'han identificat les següents:

- Millorar el procés d'instal·lació de contenidors mostrant el progrés del procés i donant a l'usuari l'opció de cancel·lar l'operació.
- Crear paquets preconfigurats de contenidors gestionats per donar més propostes als usuaris i facilitar així encara més la publicació de diferents serveis.
- Millorar la publicació de l'accés remot als serveis, actualment es redirigeix als usuaris cap a la IP i port publicat per l'aplicació. Potser seria interessant poder parametritzar altres aspectes com el protocol (http, https, ftp, ...) o la URL complerta cap a una secció concreta de l'aplicació.

- Es podria valorar donar l'opció de gestionar qualsevol container instal·lat al servidor i permetre "convertir-lo" en gestionat si així ho vol l'usuari.

6. Glossari

Linux Container: És una manera d'encapsular aplicacions per distribuir-les sobre diferents servidors Linux. L'encapsulament encara comparteix determinats recursos del sistema operatiu host.

Contenidor gestionat: És la definició d'un contenidor Docker que ha estat afegida al nou portal d'administració de Docker a Cloudy. Aquest container no està instal·lat ni disponible al sistema, es tracta només de la definició i configuració de paràmetres bàsics.

Microservei: És un servei autònom de complexitat reduïda que pot comunicar-se amb d'altres microserveis.

Barrera tecnològica: Concepte que serveix per definir que determinada tecnologia no està a l'abast de tots els usuaris per la seva complexitat tècnica

7. Bibliografia

[1] **Cloudy** «What is Cloudy?» [article en línia]. [Data de consulta: 10 de Març de 2016] <http://cloudy.community/what-is-cloudy/>

[2] **Sandstorm** «Core Features» [article en línia]. [Data de consulta: 20 de Març 2016] <https://es.wikipedia.org/wiki/Docker_(software)>

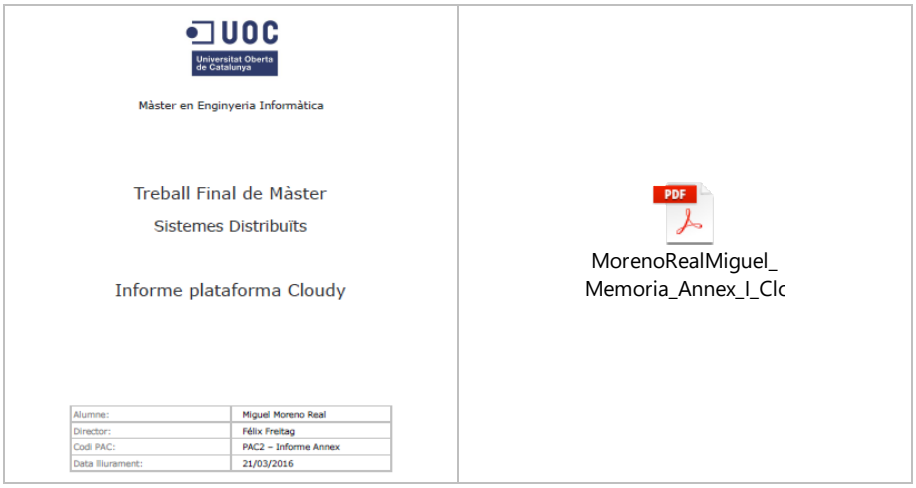
[3] **Wikipedia** «Docker (software)» [article en línia]. [Data de consulta: 7 d'Abril de 2016] <https://es.wikipedia.org/wiki/Docker_(software)>

[4] **Docker** «How is this different from virtual machines?» [article en línia]. [Data de consulta: 10 d'Abril de 2016] <https://www.docker.com/what-docker>

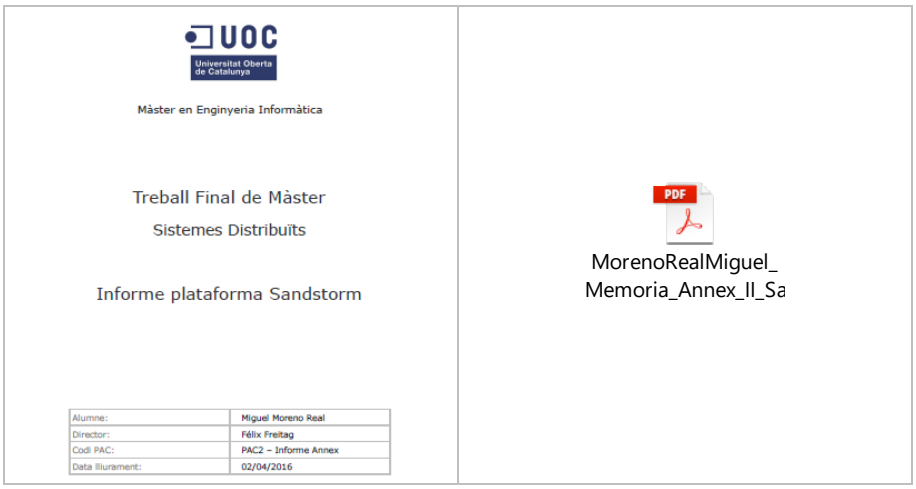
[5] **Cloudy** «Cloudy News» [article en línia]. [Data de consulta: 9 de Juny de 2016] <http://cloudy.community/news/>

8. Annexos

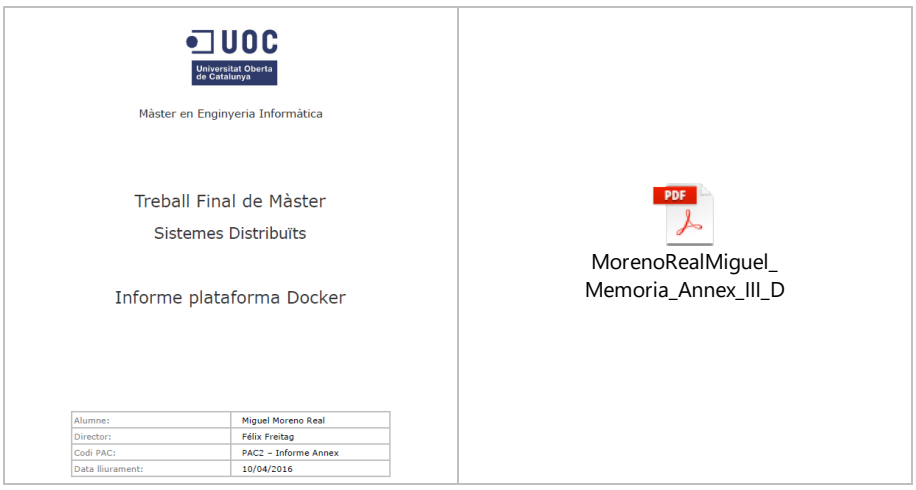
Annex I: Informe avaluació sistema Cloudy



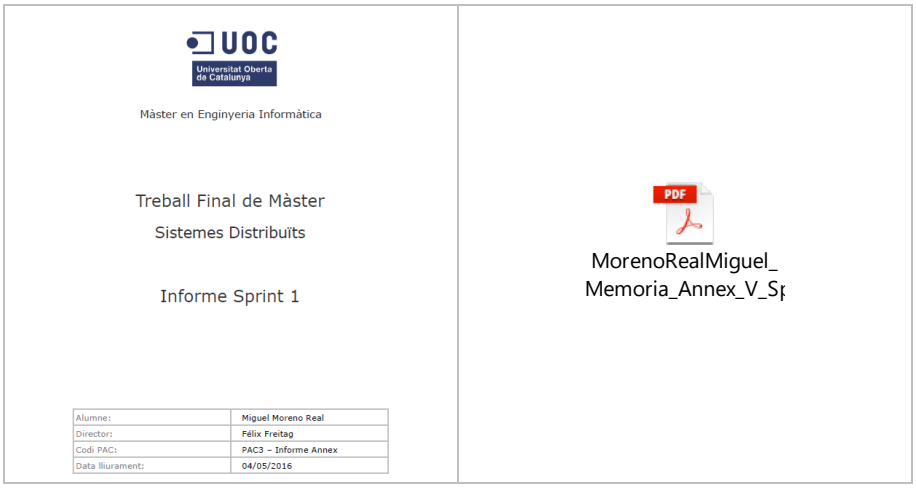
Annex II: Informe avaluació sistema Sandstorm



Annex III: Informe avaluació sistema Docker



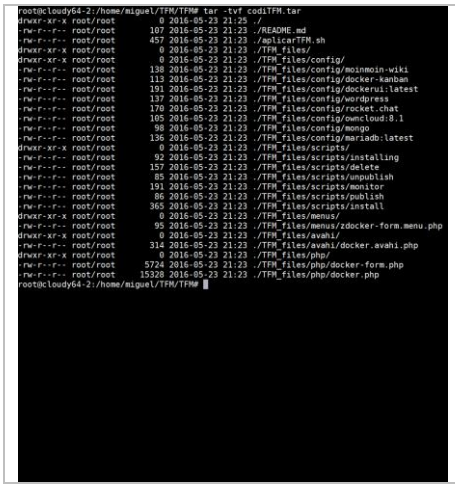
Annex IV: Informe Sprint 1 de la Implementació



Annex V: Informe Sprint 2 de la Implementació

Annex VI: Informe Sprint 3 de la Implementació

Annex VII: Codi implementat



Fitxer .tar

MorenoRealMiguel_
Memoria_Annex_VII_C

Annex I: Informe avaluació sistema Cloudy

1. Introducció

En aquest informe s'analitza la plataforma Cloudy, identificant els seus potencials cassos d'us. Per redactar l'informe s'ha partit de la informació publicada a la web i fòrums de la comunitat i ha estat confirmada i ampliada amb proves sobre un entorn de test creat expressament a tal efecte.

1.1.¿Què és Cloudy?

Segons el propis desenvolupadors del projecte, Cloudy és una distribució Linux creada per fomentar la adopció d'entorns cloud a les xarxes comunitàries.

La principal motivació d'aquesta distribució és facilitar als usuaris la creació d'una xarxa distribuïda i descentralitzada alhora que facilita el desplegament de determinats serveis sobre la mateixa.

1.2.Primeres sensacions

La veritat, és que després d'instal·lar el primer node, s'aprecia que han aconseguit el seu propòsit. La instal·lació és simple i ràpida, amb poca configuració i sense requerir coneixements tècnics. Un cop instal·lats la resta de nodes i veient que aquests tenen visibilitat entre ells mitjançant una VPN autoconfigurada podem confirmar que Cloudy compleix allò que promet, la creació d'una xarxa comunitària distribuïda i descentralitzada sense necessitat de coneixements tècnics.

Fent una ullada al portal web d'administració, veiem la capacitat que tenim per desplegar diferents serveis sobre els nodes de la xarxa només fent un clic.

2. Entorn de test

Per poder fer un anàlisi amb més profunditat d'aquest sistema s'han creat dos entorns.

Característiques del primer entorn:

- SO: Fem servir la ISO proporcionada al web de Cloudy (Debian 7.x customitzat)
- 7 nodes
- Despleguem tahoe-LAFS a tots els nodes

Amb aquest entorn podem extreure les primeres sensacions de la plataforma, veient la facilitat de la instal·lació i com podem desplegar sistemes com un Filesystem distribuït sense tenir o aplicar coneixements tècnics.

Però aquesta distribució està basada en una distribució Debian de 32-bits. Tenint en compte que més endavant volem fer proves amb contenidors Docker (basats en 64-bits), continuarem les proves amb un entorn amb aquesta arquitectura.

Això és possible gracies a que els desenvolupadors de Cloudy han creat un procediment que anomenen "Cloudynitzar", que permet desplegar la capa de SW Cloudy sobre qualsevol Debian. Ho aprofitarem per crear una xarxa Cloudy de 64-bits.

Característiques del segon entorn:

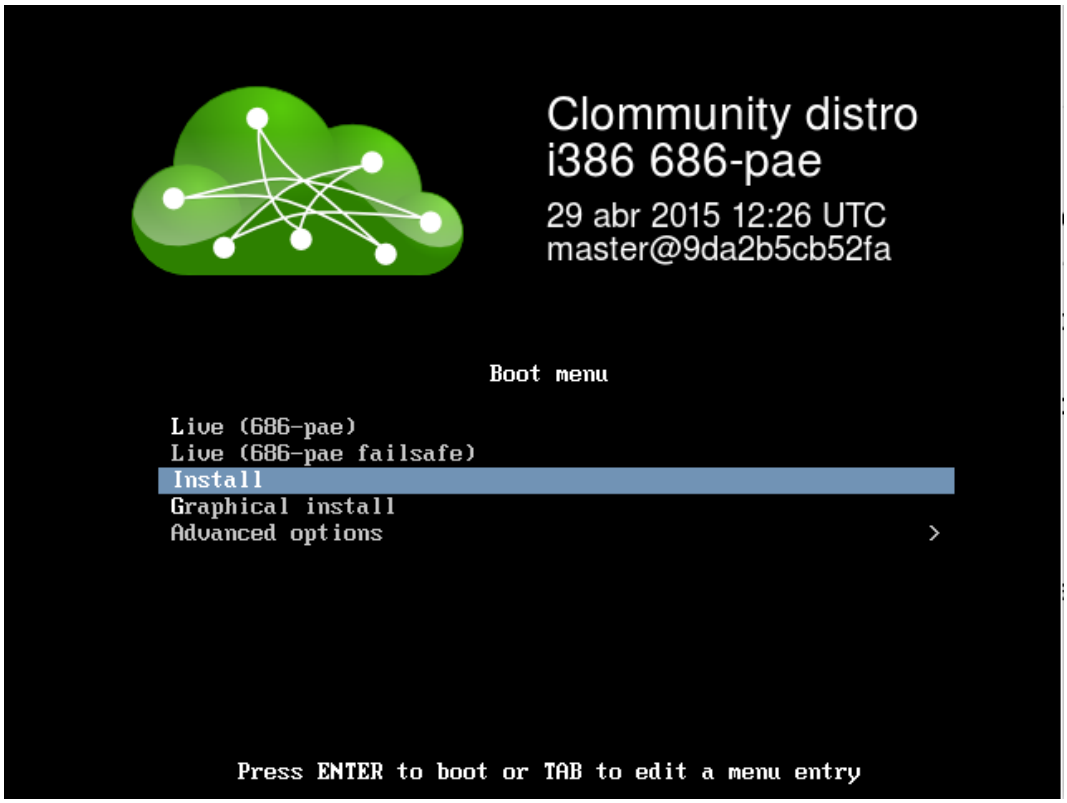
- SO: Fem servir l'última versió publicada al web de Debian (64-bits)
- 5 nodes
- "Cloudynitzem" tots els nodes

2.1. Instal·lació 32-bits

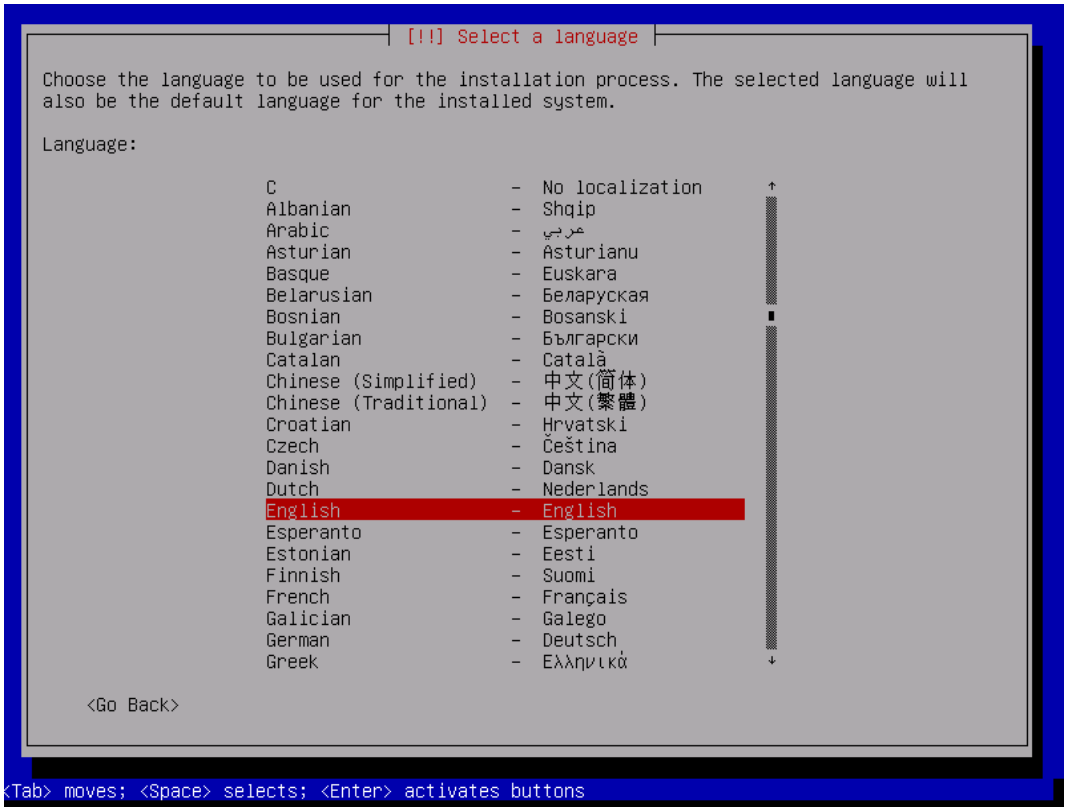
Partint de la imatge cloudy.iso de http://repo.clommunity-project.eu/images/stable/cloudy.iso

Passos a seguir:

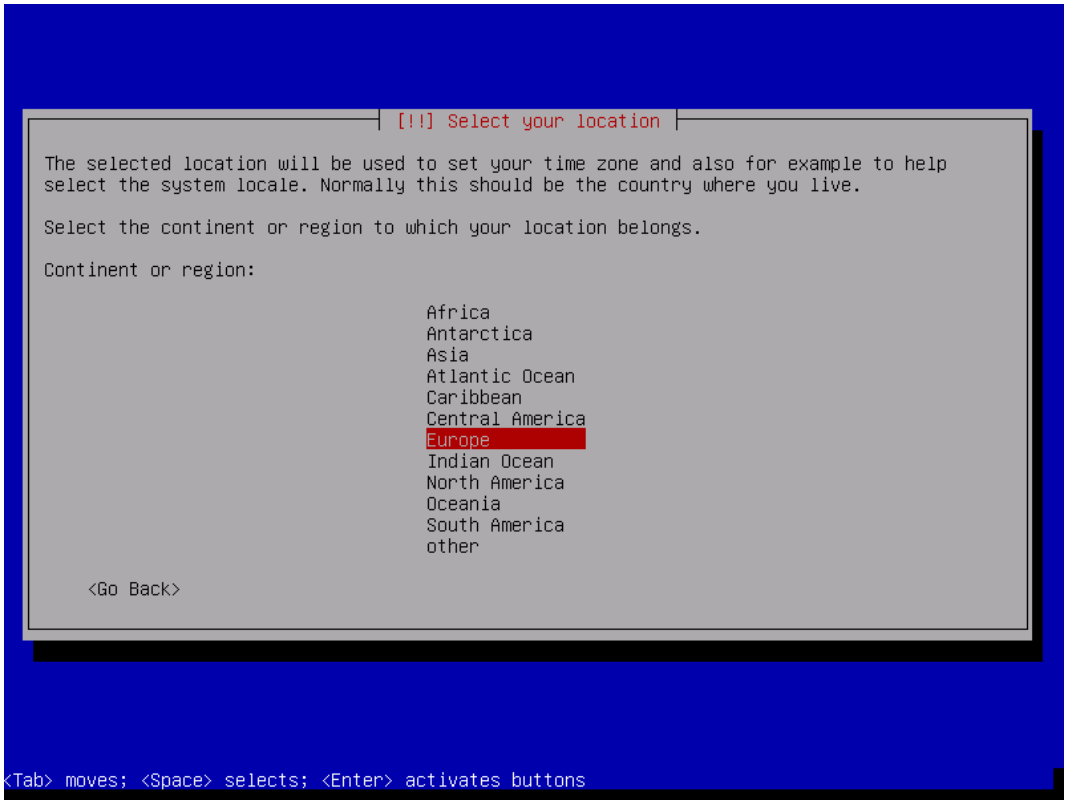
Comencem la instal·lació



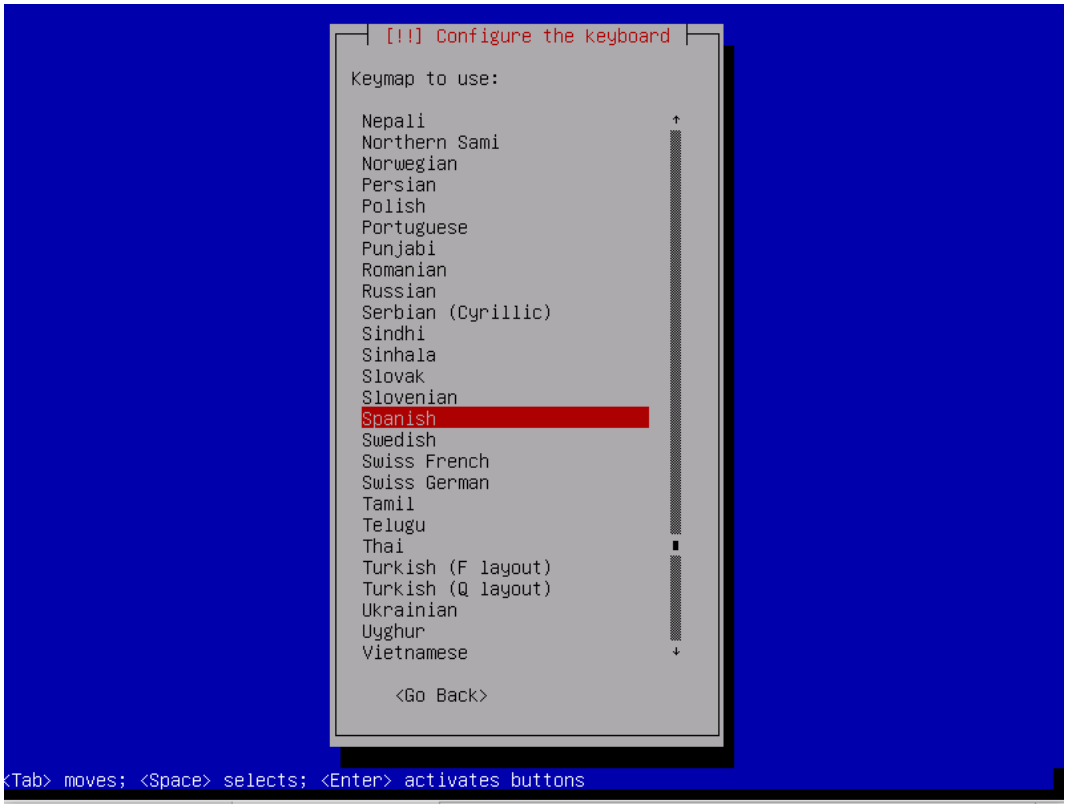
Seleccionem l'idioma



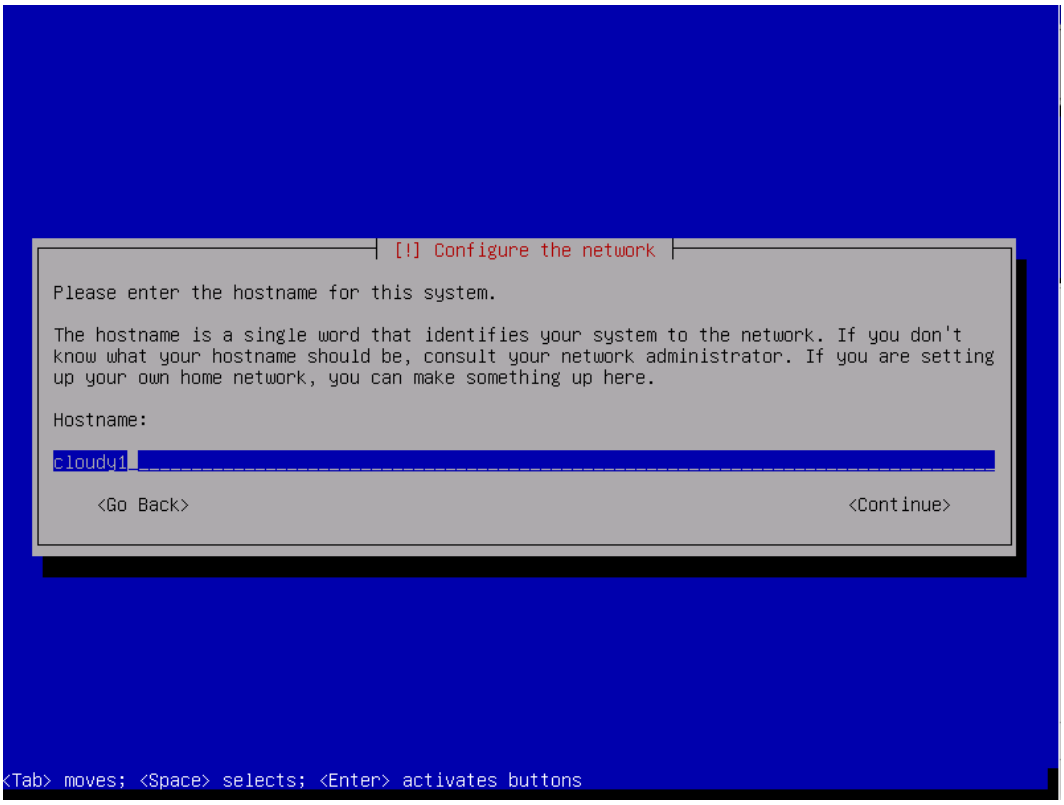
Seleccionem localització



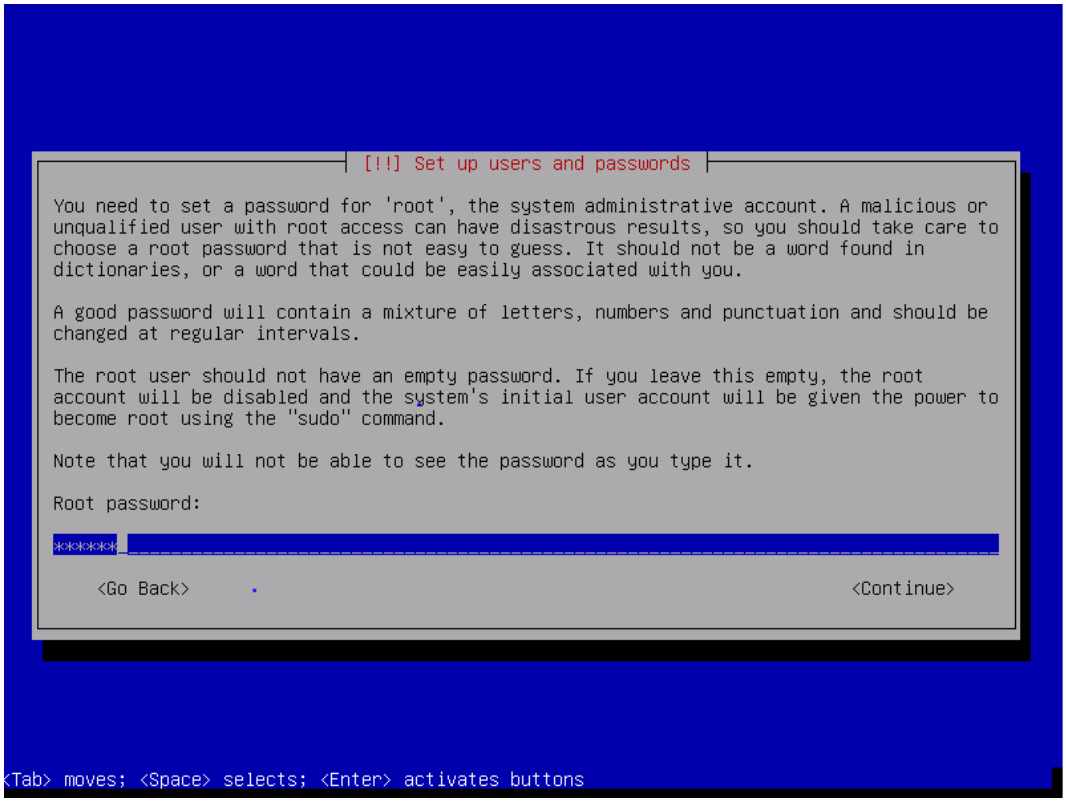
Definim la configuració del teclat



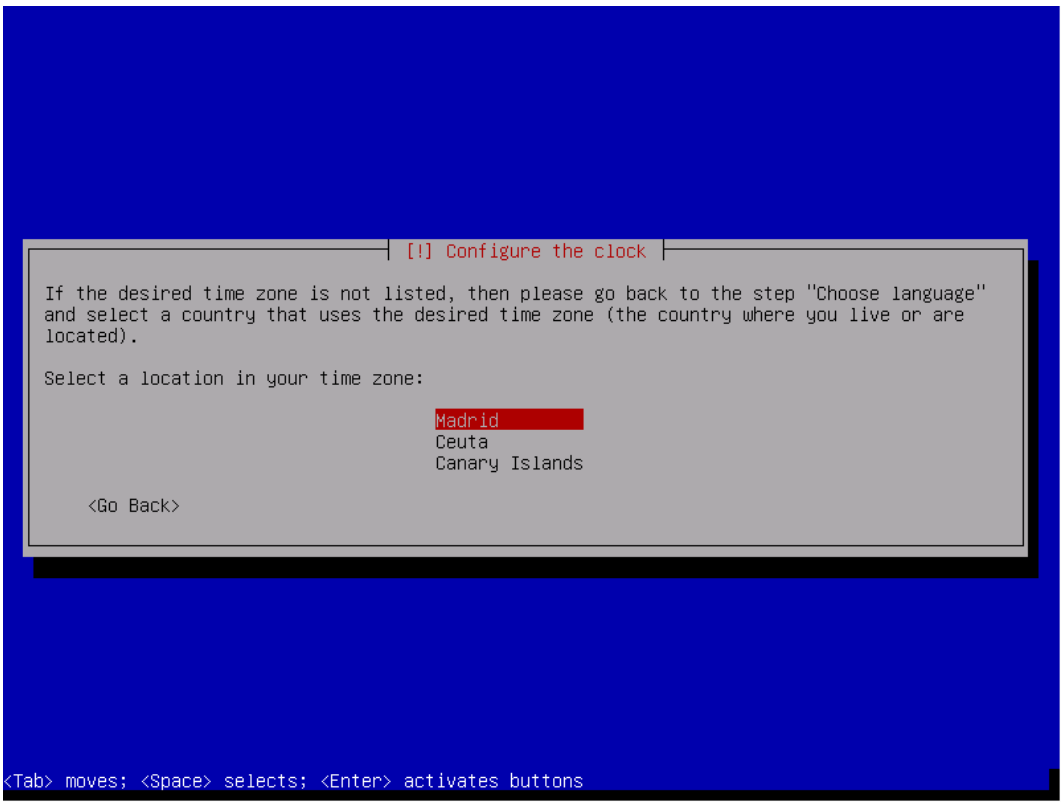
Definim hostname i domini



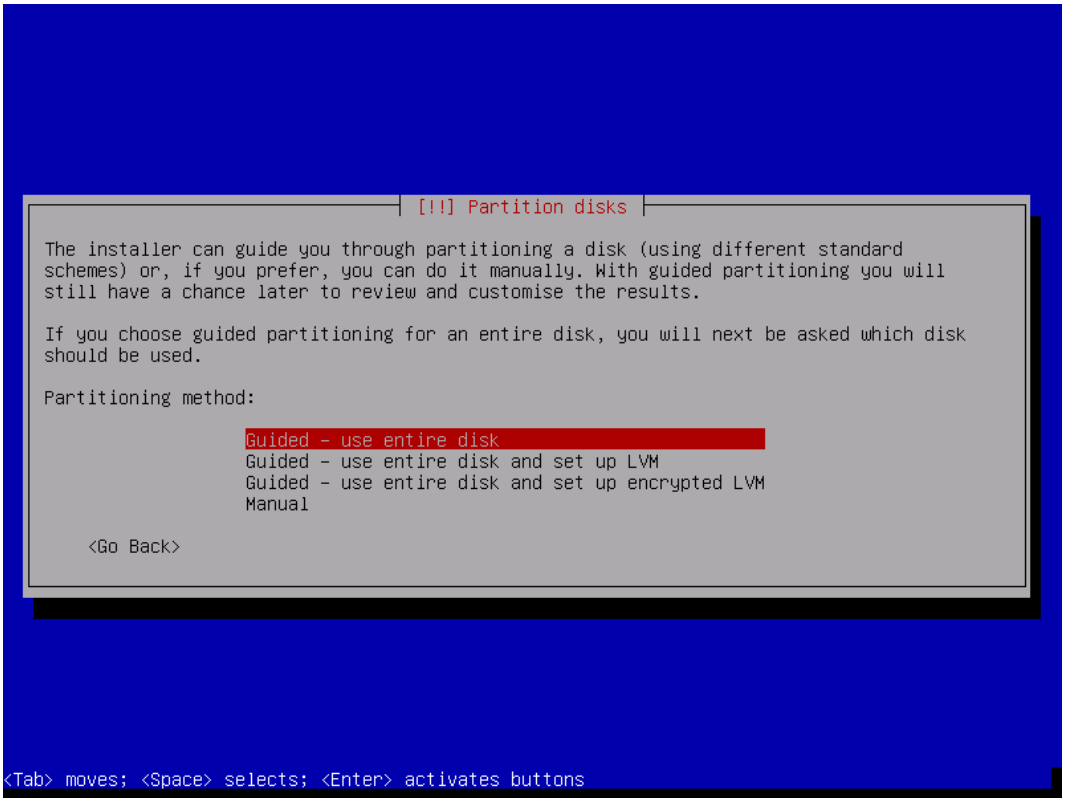
Definim password de root i per un nou usuari



Definim la zona horaria



Configurem el disc per defecte



I ja tenim el node creat i configurat a la xarxa/domini definit

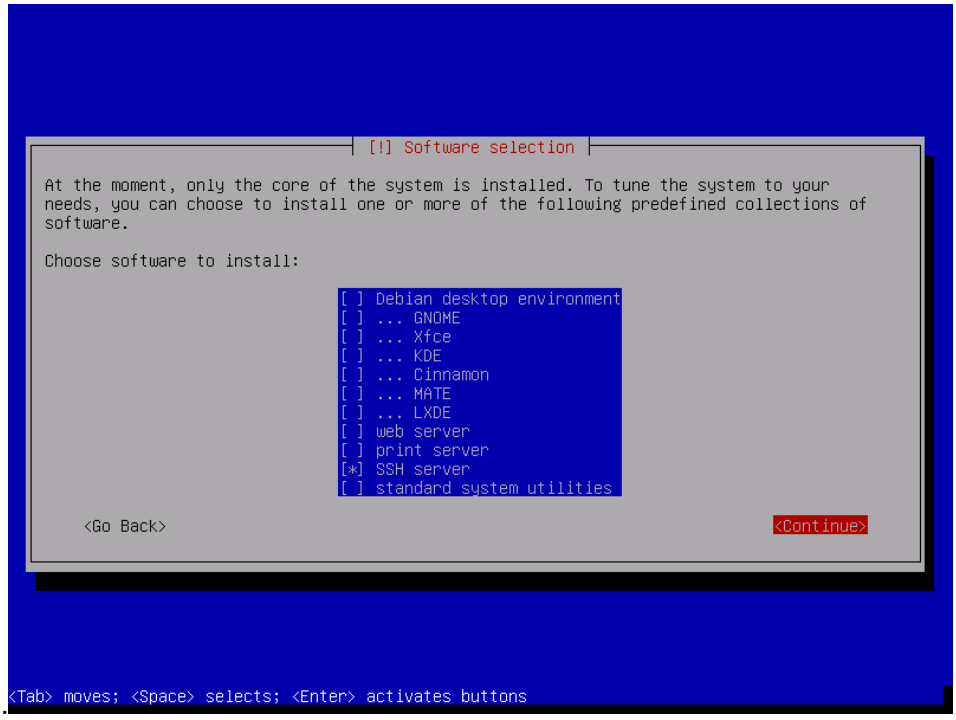
La instal·lació no necessita més de 10 minuts.

2.2. Instal·lació 64-bits

Partint de la ISO publicada al web oficial de Debian:
http://cdimage.debian.org/debian-cd/8.3.0/multi-arch/iso-cd/debian-8.3.0-amd64-i386-netinst.iso

Iniciem la instal·lació en mode 64 bits. Tota el procés és idèntic a la instal·lació de la ISO de Cloudy de 32-bit. Per tant, seguim els mateixos passos fins a arribar al següent punt.

Aquesta és la única pantalla diferent durant la instal·lació. Desmarquem totes les opcions ja que l'objectiu és instal·lar la capa Cloudy, només instal·lem el servidor SSH per habilitar la gestió.



Si no marquem aquesta casella, podem activar SSH més endavant, instal·lant el SW:

```
root@cloudy64-1:~# apt-get install openssh-client
```

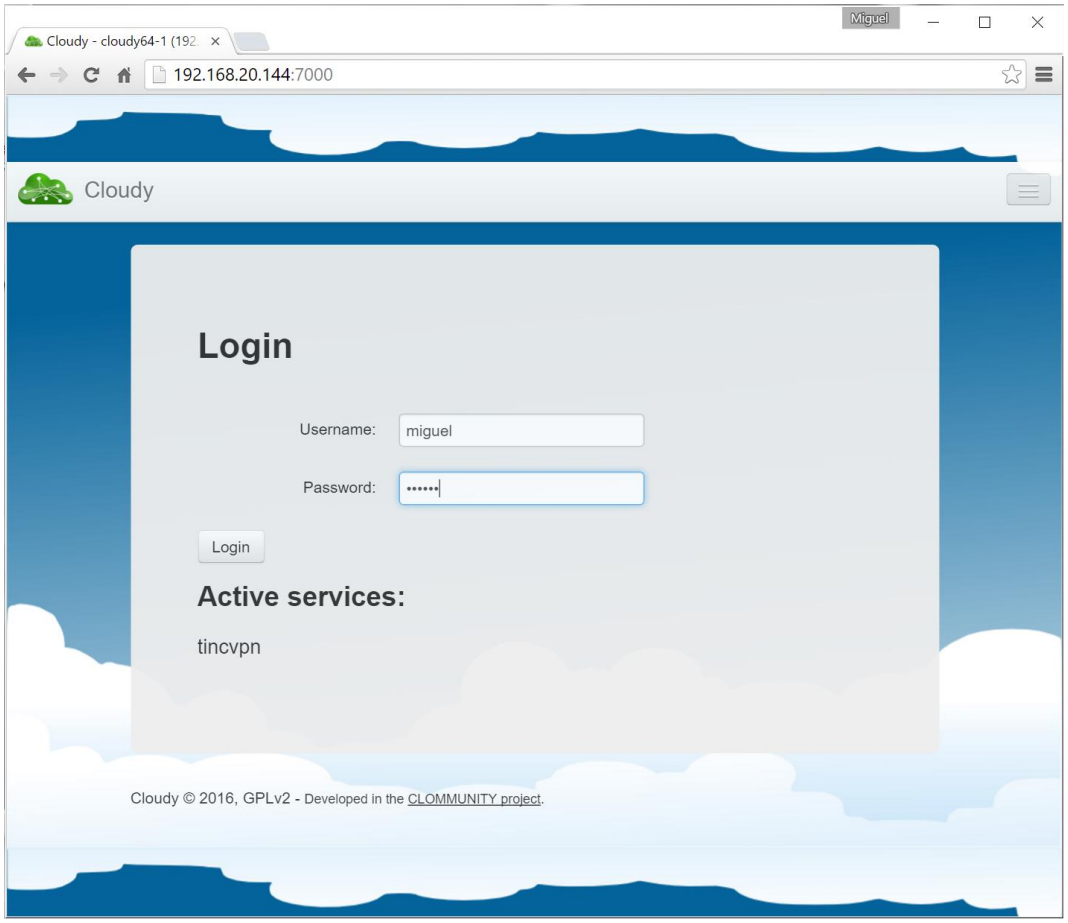
Ja tenim el servidor operatiu i preparat per instal·lar la capa de software de Cloudy

Cloudynizar

Per fer aquest pas seguim la guia publicada a guifi.net: http://es.wiki.guifi.net/wiki/Cloudynizar

```
#Instal·lem Git per descarregar l'ultima versió del script per Cloudynitzar
apt-get update
apt-get install git-core

#Descarreguem el script
git clone https://github.com/Clommunity/cloudynitzar && cd cloudynitzar

#Donem permisos d'execució i executem
chmod +x cloudynitzar.sh
./cloudynitzar.sh
```

Ara ja podem gestionar aquest servidor com un node més de Cloudy:



2.3. Detall dels entorns

Aquest són els detalls dels entorns creats per les proves:

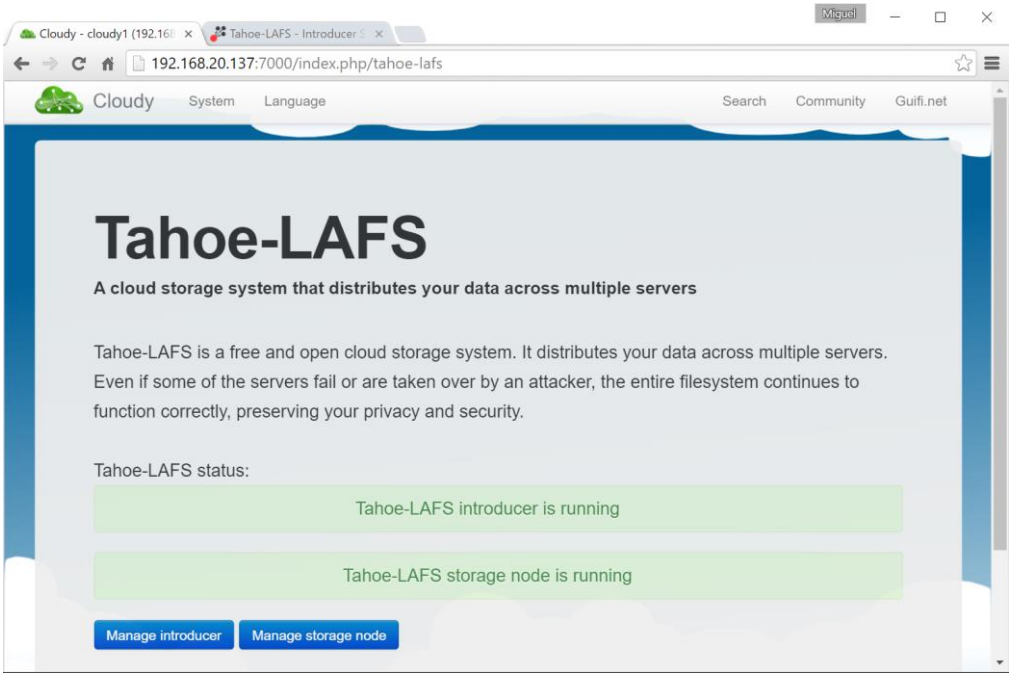
Nom	Domini	SO	Arquitectura	IP	CPU	RAM
Cloudy1	TFM	Debian 7.8	32-bit	192.168.20.137	1 vCPU	512GB
Cloudy2	TFM	Debian 7.8	32-bit	192.168.20.138	1 vCPU	512GB
Cloudy3	TFM	Debian 7.8	32-bit	192.168.20.139	1 vCPU	512GB
Cloudy4	TFM	Debian 7.8	32-bit	192.168.20.140	1 vCPU	512GB
Cloudy5	TFM	Debian 7.8	32-bit	192.168.20.141	1 vCPU	512GB
Cloudy6	TFM	Debian 7.8	32-bit	192.168.20.142	1 vCPU	512GB
Cloudy7	TFM	Debian 7.8	32-bit	192.168.20.143	1 vCPU	512GB
Cloudy64-1	TFM64	Debian 8.3	64-bit	192.168.20.145	1 vCPU	512GB
Cloudy64-2	TFM64	Debian 8.3	64-bit	192.168.20.146	1 vCPU	512GB
Cloudy64-3	TFM64	Debian 8.3	64-bit	192.168.20.147	1 vCPU	512GB
Cloudy64-4	TFM64	Debian 8.3	64-bit	192.168.20.148	1 vCPU	512GB
Cloudy64-5	TFM64	Debian 8.3	64-bit	192.168.20.149	1 vCPU	512GB

Tots el servidors s'han creat amb tecnologia de virtualització vmware i s'executen a un mateix host físic.

3. Proves

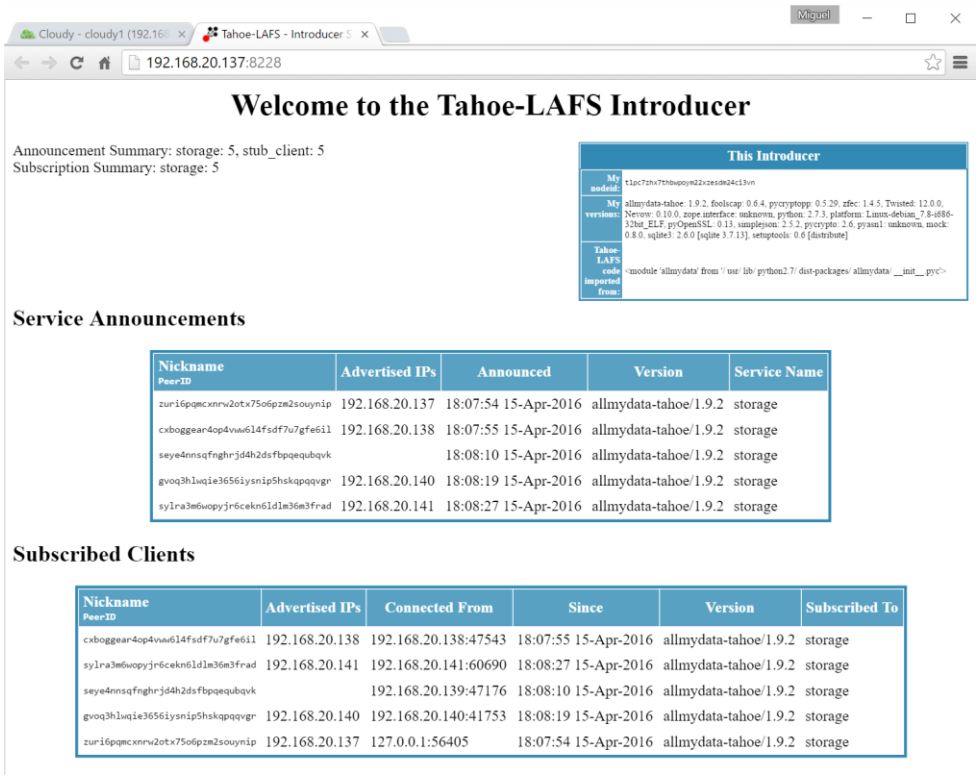
3.1. Desplegament de serveis

Per realitzar proves a la plataforma es desplega sobre la plataforma de 32-bit el servei de Tahoe-LAFS. Accedint a la pestanya d'aquest servei es pot instal·lar al servidor un node "introducer" i/o un "Storage Node", que són els dos rols possibles en aquest FileSystem:



Un cop més, veiem com sense coneixements tècnics és possible desplegar la solució instal·lant un “introducer” al node Cloudy1 i desplegant el rol de “Storage Node” a tots els nodes del clúster (incloent-hi el mateix node Cloudy1)

Des de aquesta mateixa gestió web podem verificar l’estat del Filesystem creat i els Storage Nodes que s’han afegit i estan operatius.

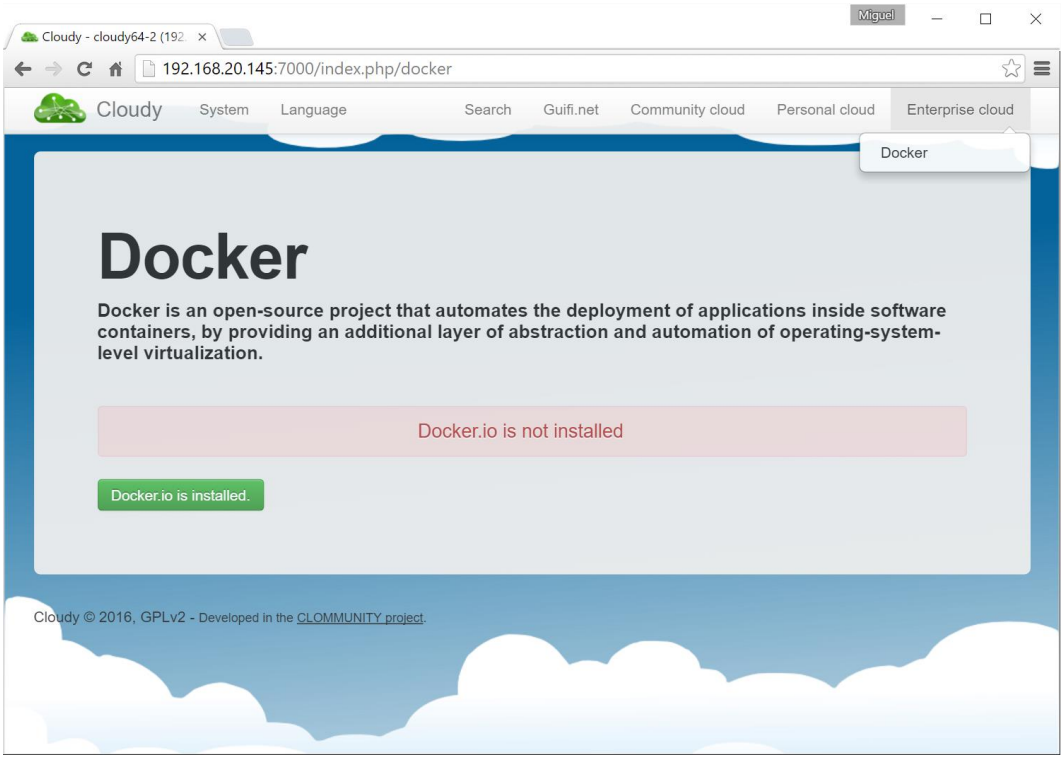


En realitat, el que es pot fer des de aquest portal de gestió web és llençar comandes d’instal·lació i configuració al node que estàs gestionant.

3.2. Docker

Tal com havíem previst, la plataforma Cloudy de 32-bits no té l’opció per instal·lar Docker. A més, si volguéssim instal·lar-ho manualment ens trobaríem amb un seguit d’inconvenients derivats de que la plataforma Docker està creada nativament amb arquitectura de 64-bits.

D’altra banda, al canviar al entorn de test de 64-bits, ens trobem que en aquest cas si que tenim la opció per instal·lar Docker de forma integrada al nostre clúster Cloudy.



Amb un clic, des de la pestanya “Enterprise Cloud”, podem desplegar Docker sobre el node Cloudy.

Accedint des de línia de comandes, confirmem que s’ha instal·lat correctament:

```
root@cloudy64-1:~# docker version
Client version: 1.6.2
Client API version: 1.18
Go version (client): go1.3.3
Git commit (client): 7c8fca2
OS/Arch (client): linux/amd64
Server version: 1.6.2
Server API version: 1.18
Go version (server): go1.3.3
Git commit (server): 7c8fca2
OS/Arch (server): linux/amd64
```

Repetim aquesta operació a tots els nodes i verifiquem el funcionament de Docker. Executem el “Hello World!” bàsic de Docker i comprovem que tot és correcte.

```
root@cloudy64-1:~# docker run docker/whalesay cowsay boo
Unable to find image 'docker/whalesay:latest' locally
latest: Pulling from docker/whalesay

e9e06b06e14c: Pull complete
a82efea989f9: Pull complete
37bea4ee0c81: Pull complete
```

```
07f8e8c5e660: Pull complete
676c4a1897e6: Pull complete
5b74edbcaa5b: Pull complete
1722f41ddcb5: Pull complete
99da72cfe067: Pull complete
5d5bd9951e26: Pull complete
fb434121fc77: Already exists
Digest:
sha256:178598e51a26abbc958b8a2e48825c90bc22e641de3d31e18aaf55f3258ba93b
Status: Downloaded newer image for docker/whalesay:latest
 _____
< boo >
 -----
    \
     \
      \
                    ##        .
              ## ## ##       ==
           ## ## ## ##      ===
       /""""""""""""""""___/ ===
  ~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ /  ===- ~~~
       \______ o          __/
        \    \        __/
          \____\______/
root@cloudy64-1:~#
root@cloudy64-1:~#
root@cloudy64-1:~# docker run docker/whalesay cowsay HelloWorld!
 _____________
< HelloWorld! >
 -------------
    \
     \
      \
                    ##        .
              ## ## ##       ==
           ## ## ## ##      ===
       /""""""""""""""""___/ ===
  ~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ /  ===- ~~~
       \______ o          __/
        \    \        __/
          \____\______/
root@cloudy64-1:~#
```

3.3. Altres proves

A més de les proves documentades en els apartats anteriors, s'han fet proves amb els diferents sistemes disponibles a la plataforma Cloudy amb l'objectiu d'entendre el seu funcionament, objectiu i abast real.

4. Observacions

- Cloudy és una plataforma molt potent que simplifica extraordinàriament el desplegament de clústers distribuïts sense tenir un ampli coneixement tècnic. Com els mateixos membres de la comunitat ja ens avançaven, eliminar aquesta barrera tecnològica permet que més usuaris s'animin a fer-la servir.
- Cloudy integra un sèrie de serveis que es poden instal·lar i gestionar (de forma bàsica) des de l'entorn web. Reforçant la idea de simplificar l'accés a la gestió.
- D'altra banda, ens trobem que el "catàleg" de serveis disponibles per gestionar des del portal web és certament limitat, obligant a fer servir la operació manual si vols treure partit del clúster o tens necessitats específiques.

5. Conclusions

- Cloudy és un bon punt de partida per crear clústers distribuïts ja que simplifica la configuració inicial. En cas d'iniciar un projecte amb aquesta necessitat, és una plataforma a tenir en compte.
- Te punts de millora incrementant els serveis que es poden instal·lar des de la gestió web i millorant la gestió que es pot fer sense accedir a la línia de comandes.
    - Per exemple, podem instal·lar Docker des de el portal web, però res més. Tota la gestió es fa a la línia de comandes. En aquest sentit, Cloudy no aporta gaire valor afegit, ja que si l'usuari està obligat a fer qualsevol operació sobre Docker a la línia de comandes, segur que estarà capacitat per realitzar també la instal·lació.
- En qualsevol cas, un cop verificat que Docker funciona sobre la plataforma Cloudy, obre un gran ventall d'oportunitats.

Annex II: Informe avaluació Sandstorm

1. Introducció

En aquest informe s'analitza la plataforma Sandstorm, identificant els seus potencials cassos d'us de forma individual i conjuntament amb la plataforma Cloudy.

Per redactar l'informe s'ha partit de la informació publicada a la web i fòrums de la comunitat i ha estat confirmada i ampliada amb proves sobre un entorn de test creat expressament a tal efecte.

1.1. ¿Què és Sandstorm?

Sandstorm és un software que s'instal·la sobre un servidor amb sistema operatiu Linux x86 i permet publicar diferents aplicacions col·laboratives de forma fàcil i efectiva.

Sandstorm proporciona un portal d'aplicacions que permet desplegar apps al el servidor sense coneixements tècnics i permet la ràpida gestió de la compartició de recursos entre usuaris assegurant la gestió d'accés.

El sistema es pot fer servir en cloud, fent servir el servei SaaS de Sandstorm.io o be instal·lant-ho a la teva pròpia infraestructura. Aquesta última opció serà l'escollida per dur a terme aquest anàlisi, tenint en compte que l'objectiu és instal·lar-ho sobre Cloudy.

2. Proves a realitzar

- Verificar la compatibilitat de Sandstorm sobre l'ecosistema de Cloudy
- Analitzar i verificar les capacitats del sistema Sandstorm.
- Estudiar possibles cassos d'ús de Sandstorm aprofitant la plataforma Cloudy

3. Entorn de test

Les proves a realitzar es centraran en descobrir i testejar el potencial del sistema alhora que es realitzen proves de compatibilitat sobre la plataforma Cloudy.

Per tant, l'entorn de test on es realitzaran les proves és un clúster Cloudy. S'aprofitarà l'entorn de test existent de la plataforma Cloudy de 64-bit:

Nom	Domini	SO	Arquitectura	IP	CPU	RAM
Cloudy64-1	TFM64	Debian 8.3	64-bit	192.168.20.145	1 vCPU	512GB
Cloudy64-2	TFM64	Debian 8.3	64-bit	192.168.20.146	1 vCPU	512GB

Cloudy64-3	TFM64	Debian 8.3	64-bit	192.168.20.147	1 vCPU	512GB
Cloudy64-4	TFM64	Debian 8.3	64-bit	192.168.20.148	1 vCPU	512GB
Cloudy64-5	TFM64	Debian 8.3	64-bit	192.168.20.149	1 vCPU	512GB

3.1. Procés d'instal·lació

Passos de la instal·lació:

1. Instal·lació amb línia de comandes:
   1.1. Executem la comanda al servidor: curl https://install.sandstorm.io | bash
   1.2. Seleccionem la modalitat d'instal·lació per defecte. Es fa servir el directori per defecte, serveis per defecte, els serveis s'iniciaran amb el sistema operatiu i s'aplicaran actualitzacions automàtiques
   1.3. Es substitueix el port 80 del servidor web pel 6080 degut a que el 80 ja ho fa servir el servidor web de Cloudy
   1.4. Es configura el wildcard per defecte[server.domain:port]: *.cloudy64-1.home64:6080
   1.5. A partir d'aquest punt es continua la instal·lació al navegador, a la següent URL: http://cloudy64-1.home64:6080/setup/token/2aa3c0ee4eec496a7b8de69b78349f82c40d50c5
2. Configuració des del navegador:
   2.1. Seleccionem el Identity Provider (En aquest cas GitHub)
   2.2. Creem el comte a GitHub i donem d'alta l'usuari administrador

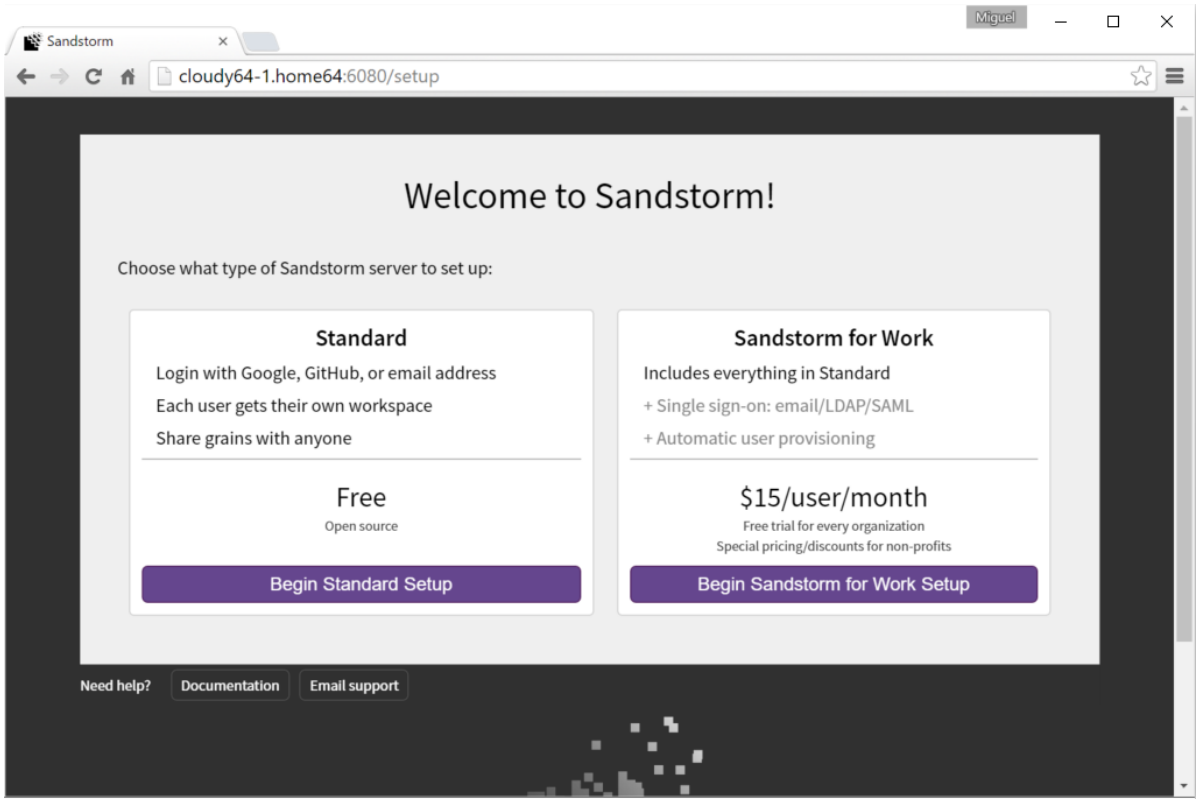
3.2. Detall de l'instal·lació

```
root@cloudy64-1:~# curl https://install.sandstorm.io | bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 74985  100 74985    0     0  92134      0 --:--:-- --:--:-- --:--:-- 92119
Sandstorm makes it easy to run web apps on your own server. You can have:

1. A typical install, to use Sandstorm (press enter to accept this default)
2. A development server, for working on Sandstorm itself or localhost-based app
development

How are you going to use this Sandstorm install? [1] 1
We're going to:
* Install Sandstorm in /opt/sandstorm
* Automatically keep Sandstorm up-to-date
* Create a service user (sandstorm) that owns Sandstorm's files
* Configure Sandstorm to start on system boot (with systemd)

Rest assured that Sandstorm itself won't run as root.
OK to continue? [yes] yes

NOTE: It looks like your system already has some other web server installed
      (port 80 and/or 443 are taken), so Sandstorm cannot act as your main
      web server.

      This script can set up Sandstorm to run on port 6080 instead,
      without HTTPS. This makes sense if you're OK with typing the port number
      into your browser whenever you access Sandstorm and you don't need
      security. This also makes sense if you are going to set up a reverse proxy;
      if so, see https://docs.sandstorm.io/en/latest/administering/reverse-proxy/
```

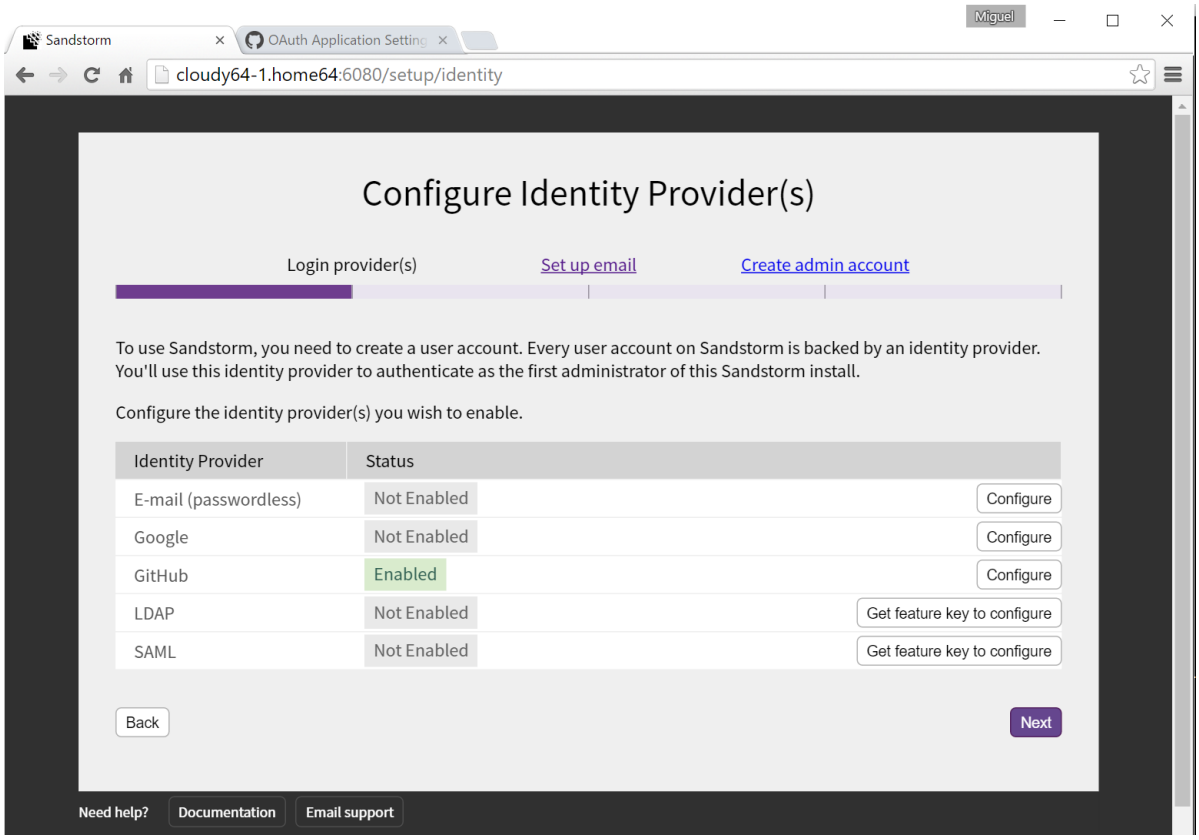
```
      If you want, you can quit this script with Ctrl-C now, and go uninstall
      your other web server, and then run this script again. It is also OK to
      proceed if you want.

OK to skip automatic HTTPS setup & bind to port 6080 instead? [yes] yes
As a Sandstorm user, you are invited to use a free Internet hostname as a subdomain of
sandcats.io,
a service operated by the Sandstorm development team.

Sandcats.io protects your privacy and is subject to terms of use. By using it,
you agree to the terms of service & privacy policy available here:
https://sandcats.io/terms https://sandcats.io/privacy

Choose your desired Sandcats subdomain (alphanumeric, max 20 characters).
Type the word none to skip this step, or help for help.
What *.sandcats.io subdomain would you like? [] none
URL users will enter in browser: [http://cloudy64-1.home64:6080]
Sandstorm requires you to set up a wildcard DNS entry pointing at the server.
This allows Sandstorm to allocate new hosts on-the-fly for sandboxing purposes.
Please enter a DNS hostname containing a '*' which maps to your server. For
example, if you have mapped *.foo.example.com to your server, you could enter
"*.foo.example.com". You can also specify that hosts should have a special
prefix, like "ss-*.foo.example.com". Note that if your server's main page
is served over SSL, the wildcard address must support SSL as well, which
implies that you must have a wildcard certificate. For local-machine servers,
we have mapped *.local.sandstorm.io to 127.0.0.1 for your convenience, so you
can use "*.local.sandstorm.io" here. If you are serving off a non-standard
port, you must include it here as well.
Wildcard host: [*.cloudy64-1.home64:6080]

Config written to /opt/sandstorm/sandstorm.conf.
Finding latest build for dev channel...
Downloading: https://dl.sandstorm.io/sandstorm-158.tar.xz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 23.8M  100 23.8M    0     0  8325k      0  0:00:02  0:00:02 --:--:-- 8322k
GPG signature is valid.
Created symlink from /etc/systemd/system/multi-user.target.wants/sandstorm.service to
/etc/systemd/system/sandstorm.service.

Your server is now online! Visit this link to configure it:

  http://cloudy64-1.home64:6080/setup/token/2aa3c0ee4eec496a7b8de69b78349f82c40d50c5


NOTE: This URL expires in 15 minutes. You can generate a new setup URL by running
'sudo sandstorm admin-token' from the command line.

To learn how to control the server, run:
  sandstorm help
root@cloudy64-1:~#
```

Accedim a la següent URL per continuar amb l'instal·lació:
http://cloudy64-1.home64:6080/setup/token/2aa3c0ee4eec496a7b8de69b78349f82c40d50c5

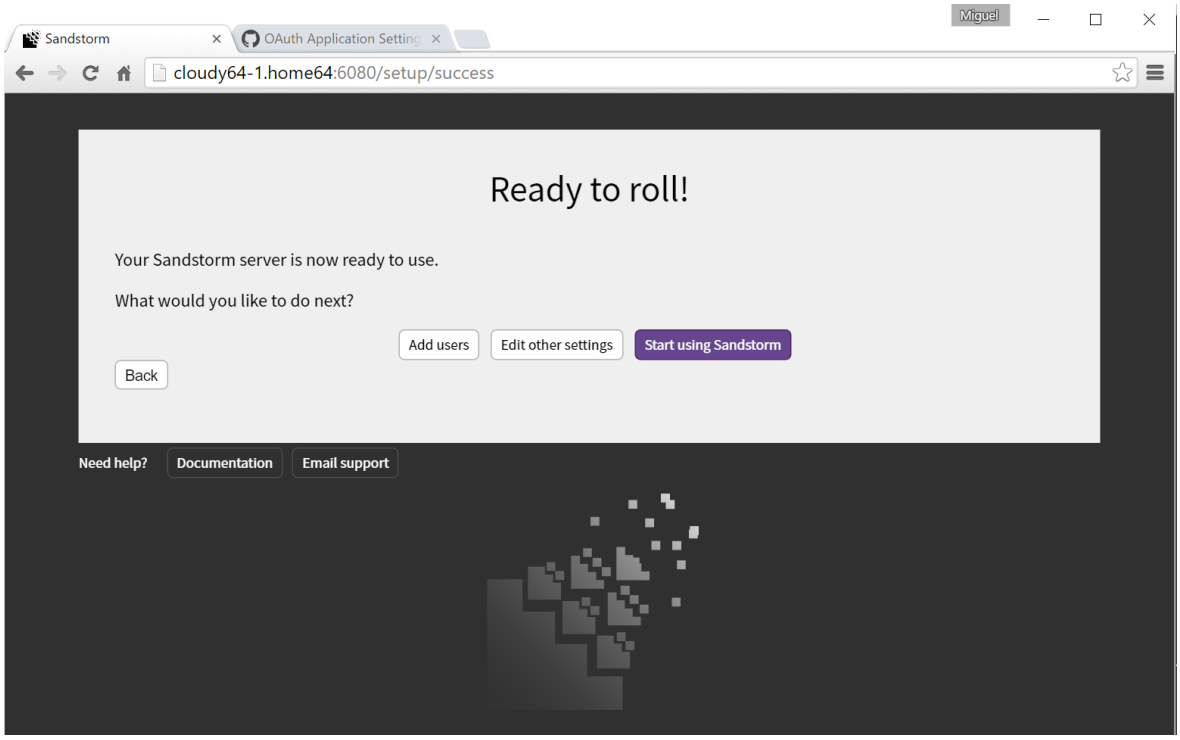
Iniciem el procediment de configuració estàndard.



Seleccionem GitHub com a Identity Provider i el configurem:

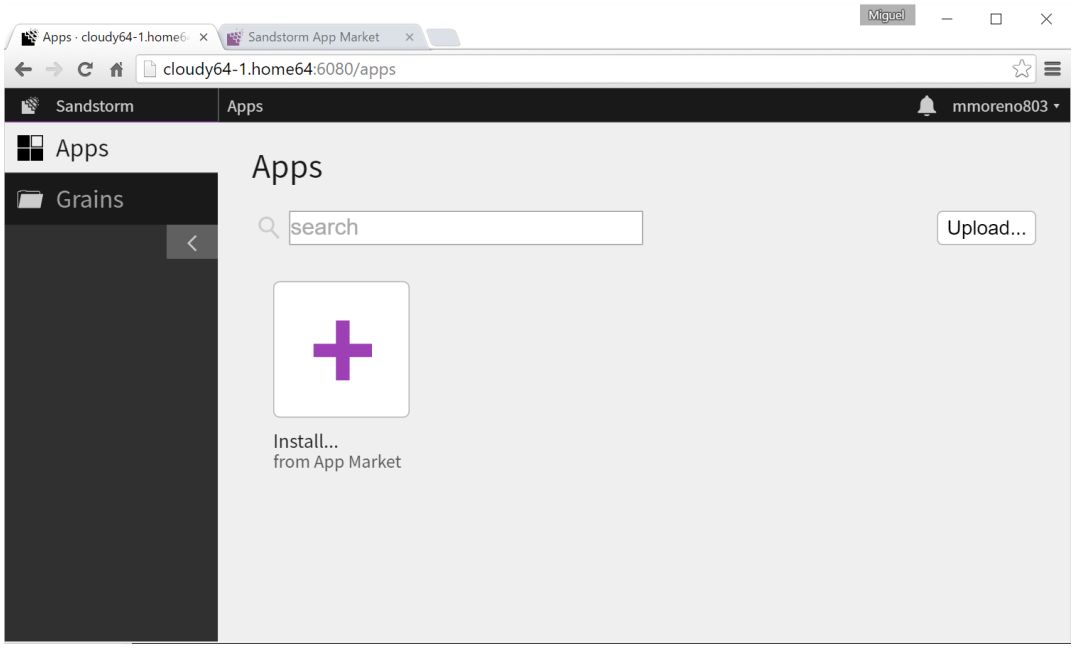


Configurem el compte de l'administrador amb GitHub i es finalitza la configuració:

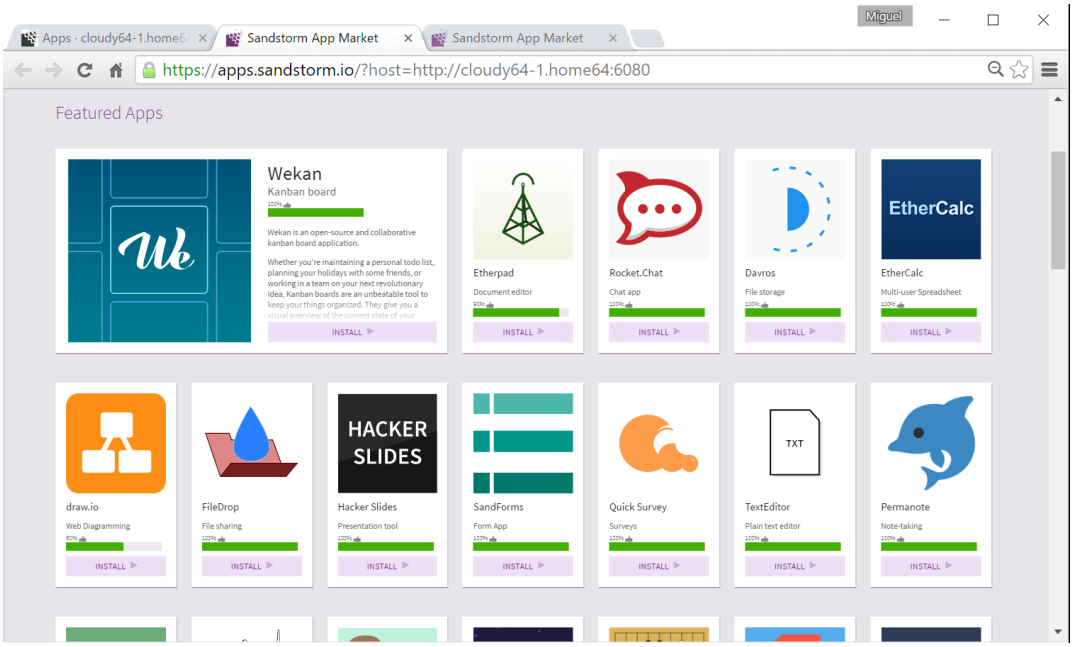


4. Proves realitzades

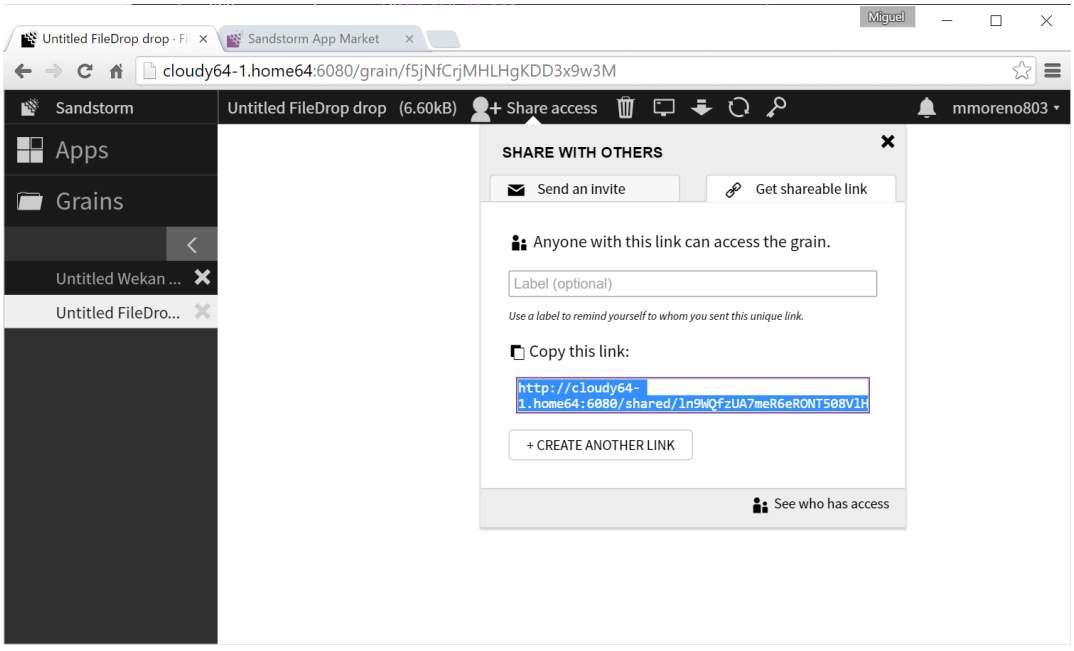
- Verifiquem la funcionalitat del sistema de login federat amb GitHub
    - Gestió de l'usuari administrador
    - Creació de nous usuaris i invitacions
- Es verifica la gestió web, el App Store i la facilitat per disposar de nous serveis al servidor:
    - http://cloudy64-1.home64:6080/apps



- S'instal·len diferents apps sobre el node Cloudy sense problemes. De forma molt ràpida i amb disponibilitat immediata. El catàleg d'aplicacions no és gaire extens, però ofereix un bon ventall d'aplicacions solvents per millorar la productivitat de forma col·laborativa.



- Es fan proves amb la gestió d'accessos i invitacions a usuaris per compartir recursos, comprovant la correcta integració amb GitHub:



- Es realitzen diferents proves amb diferents apps per entendre l'abast real de la solució, com es publiquen i comparteixen els recursos, i com encaixa tot plegat amb la filosofia de la plataforma Cloudy.

5. Observacions

- Per disseny, les dades s'emmagatzemen només en local al host os resideix la APP instal·lada. La informació no és accessible des de altres hosts, i això impedeix que les aplicacions pugin ser distribuïdes.
- Com Sandstorm planteja l'escalabilitat, és tenint en compte que una mateixa APP la pots instal·lar a tots els nodes de la teva granja, i llavors manualment, repartir l'ús d'aquesta APP entre els nodes.
  Per exemple, si volem escalar la APP de compartició de fitxers, podem crear carpetes diferents a diferents nodes fent servir la mateixa APP. Però mai seria la mateixa instància i cada carpeta tindria una URL d'accés.
- La versió gratuïta de Sandstorm no permet l'autenticació d'usuaris amb el teu propi directori o fent servir SAML contra un Identity Provider personalitzat. Únicament permet la gestió d'accés segura federant amb Google o GitHub. Tot i que és una clara limitació, totes dues opcions són suficientment bones com per que això no sigui un problema.

6. Conclusions

- Sandstorm no permet el desplegament de aplicacions distribuïdes i per tant no té capacitat per treure profit de Cloudy.
- D'altra banda, permet el desplegament de aplicacions col·laboratives de forma molt fàcil i eficaç. I això podria donar un valor afegit als nodes de Cloudy.
- Es a dir, si les nostres necessitats són crear un Filesystem, i disposem de la plataforma Cloudy, té més sentit implementar Tahoe-LAFS que una App en Sandstorm, però ens aporta la possibilitat de aprofitar la nostra infraestructura amb aplicacions que no requereixin alta disponibilitat.
- Per exemple un chat, un fòrum, un panell kanban col·laboratiu, ...

Annex III: Informe avaluació sistema Docker

1. Introducció

En aquest informe s'analitza la plataforma Docker, identificant els seus potencials cassos d'us de forma individual i conjuntament amb la plataforma Cloudy.

Per redactar l'informe s'ha partit de la informació publicada a la web i fòrums de la comunitat i ha estat confirmada i ampliada amb proves sobre un entorn de test creat expressament a tal efecte.

1.1. ¿Què és Docker?

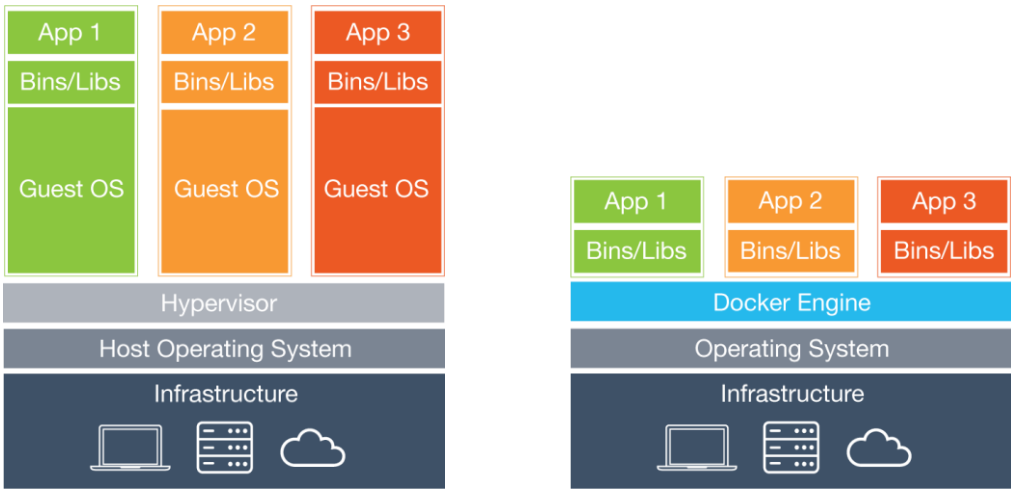
"Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de Virtualización a nivel de sistema operativo en Linux." (wikipedia)

O dit d'una altra manera, Docker ens permet virtualitzar aplicacions dins de contenidors de manera que aquests siguin autònoms i es puguin executar sobre qualsevol plataforma.

El contenidors contenen tot el que necessita la aplicació per funcionar: codi, llibreries, Filesystem, etc, de manera que garanteix el mateix funcionament independentment del host.

Un container Docker no és una màquina virtual, existeix una diferencia que fa que aquest sistema sigui molt interessant. Els contenidors no tenen sistema operatiu, reduint el consum de recursos del host i la complexitat de l'entorn. Per aconseguir-ho, es substitueix l'habitual hypervisor pel Docker Engine.

La següent imatge publicada a la web de Docker clarifica aquest concepte:



Com a primera conclusió, es pot extreure que Docker podrà executar aplicacions de forma més eficient que qualsevol hypervisor, però també es preveu que la creació d'aquestes aplicacions

serà més complexa. Mentre que en una màquina virtual simplement despleguem aplicacions com si de un entorn físic/tradicional es tractés, amb Docker haurem de construir aquest software per que es pugui interpretar pel seu Engine.

Una altre aspecte a comprovar serà si realment l'Engine de Docker serà més eficient executant contenidors que un hypervisor executant el Sistema Operatiu més l'aplicació.

1.2. Docker al mon

Veient com evoluciona aquest sistema, que cada vegada té més referències al mercat i casos d'èxit publicats, a més de la increïble acollida que està tenint per tots els grans fabricants de hardware i software del mercat, podríem avançar una resposta al dubte sobre la seva eficiència, i és que segurament ho sigui.

Referències de Docker:

- The New York Times: https://www.docker.com/customers/new-york-times-delivers-continuous-integration-pipeline-docker
- Amadeus: https://www.docker.com/customers/docker-delivers-greater-mobility-and-better-security-amadeus
- Spotify: https://www.docker.com/customers/continuous-delivery-spotify
- I molts altres: https://www.docker.com/customers

Esdeveniments on es fa referència a Docker:

- Microsoft dotNet 2016: Running ASP 5 with Docker https://blogs.msdn.microsoft.com/webdev/2015/01/14/running-asp-net-5-applications-in-linux-containers-with-docker/
- vmworld 2014 & 2015: VMware and Docker – Better Together https://cto.vmware.com/vmware-docker-better-together/
- ciscolive 2016: Multi-datacenter Docker Container Manager https://www.ciscolive.com/connect/sessionDetail.ww?SESSION_ID=4736

Aquest són només alguns exemples, però el fet de que Microsoft estigui potenciant l'ús de l'eina, vmware es faci ressò tenint en compte que es tracta de una competència natural i que fins i tot Cisco en parli, són motius per pensar que Docker pot aportar molt al mon de les TI en un curt termini.

1.3. ¿Què aporta Docker?

Docker no només aporta la possibilitat de crear containers amb aplicacions, disposa d'altres funcionalitats que complementen i fan més atractiu al producte:

- Un cop creat el container aquest es pot desplegar ràpidament sobre nodes amb l'Engine de Docker, tant amb l'objectiu de agilitzar les posades en producció com per escalar la plataforma
- Al estar auto-contingut, elimina dependències i inconsistències.
- Facilitat per compartir contenidors gràcies al Docker Hub. Un portal on es poden publicar contenidors amb tot tipus d'aplicacions i sistemes
- L'Engine de Docker manté actualitzada l'aplicació executant sempre l'ultima versió publicada, sense interacció de l'administrador.

2. Proves a realitzar

- Verificar la compatibilitat de Docker sobre l'ecosistema de Cloudy
- Analitzar i verificar les capacitats del sistema Docker.
- Estudiar possibles cassos d'ús de Docker aprofitant la plataforma Cloudy
- Estudiar eines gràfiques per la gestió de Docker

3. Entorn de test

Les proves a realitzar es centraran en descobrir i testejar el potencial del sistema alhora que es realitzen proves de compatibilitat sobre la plataforma Cloudy.

Per tant, l'entorn de test on es realitzaran les proves en un clúster Cloudy. S'aprofitarà l'entorn de test existent de la plataforma Cloudy de 64-bit:

Nom	Domini	SO	Arquitectura	IP	CPU	RAM
Cloudy64-1	TFM64	Debian 8.3	64-bit	192.168.20.145	1 vCPU	512GB
Cloudy64-2	TFM64	Debian 8.3	64-bit	192.168.20.146	1 vCPU	512GB
Cloudy64-3	TFM64	Debian 8.3	64-bit	192.168.20.147	1 vCPU	512GB
Cloudy64-4	TFM64	Debian 8.3	64-bit	192.168.20.148	1 vCPU	512GB
Cloudy64-5	TFM64	Debian 8.3	64-bit	192.168.20.149	1 vCPU	512GB

3.1. Procés d'instal·lació

Com es va veure durant l'anàlisi de Cloudy, Docker es pot instal·lar des de pla plataforma de gestió web de Cloudy amb un sol clic.

```
# curl -fsSL https://get.docker.com/ | sh
```

Com alternativa, podem fer servir la següent comanda:

4. Proves realitzades

4.1. Contenidors

4.1.1. Hello World!

A l'anàlisi de Cloudy es va provar l'execució del típic Hello World! de Docker. Per aquesta prova, s'ha creat un nou Hello World! copiant l'original però creant-ho des de una plataforma Windows i pujant-ho al Docker Hub.

Seguint el tutorial de la web de Docker, ens explica com copiar i modificar un contenidor existent. En aquest cas, el que es fa és afegir al Hello World! bàsic la possibilitat de donar missatges aleatoris amb *fortunes.*

Crear nous contenidors és relativament fàcil, tot i que s'ha de complicar com més complexa sigui la nostra aplicació. La possibilitat de publicar-los i compartir-los també és molt senzilla, així com recuperar-los de qualsevol altre host. Per últim, veiem que no importa el SO del host on es crea el container, tot i que aquest exemple és molt bàsic i no hem posat a proba l'Engine de Docker.

Com veiem, el resultat és satisfactori:

```
#Descarreguem la versió modificada del Docker Hub amb la comanda pull
root@cloudy64-1:~# docker pull mmorenoreal/docker-whale
latest: Pulling from mmorenoreal/docker-whale
c9a7e6121448: Pull complete
610b9e11f695: Pull complete
76247d3fc5ab: Pull complete
112ddcdca1d0: Pull complete
26d400839346: Pull complete
6c018392bef4: Pull complete
Digest: sha256:dd8ceb0ab1518d5133305e97e3011198c09395f678b9ab12119916b845cb0e99
Status: Downloaded newer image for mmorenoreal/docker-whale:latest
root@cloudy64-1:~#

#Executem l'aplicació amb el missatge aleatori generat amb fortunes
root@cloudy64-1:~# docker run mmorenoreal/docker-whale
 ______________________________________
/ You'd like to do it instantaneously, \
\ but that's too slow.                 /
 --------------------------------------
    \
     \
      \
                    ##        .
              ## ## ##       ==
           ## ## ## ##      ===
       /""""""""""""""""___/ ===
  ~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ /  ===- ~~~
       \______ o          __/
        \    \        __/
          \____\______/
root@cloudy64-1:~#

#Executem l'aplicació donant-li com a paràmetre el text
root@cloudy64-1:~# docker run mmorenoreal/docker-whale cowsay "Moreno Hello World!"
 ____________________
< Moreno Hello World! >
 --------------------
    \
     \
      \
                    ##        .
              ## ## ##       ==
           ## ## ## ##      ===
       /""""""""""""""""___/ ===
  ~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ /  ===- ~~~
       \______ o          __/
        \    \        __/
          \____\______/
root@cloudy64-1:~#
```

4.1.2. Ubuntu

Gràcies a les característiques dels contenidors de Dockers, es pot crear un container amb Ubuntu molt lleuger.

```
#Descarreguem el contenidor amb Ubuntu
root@cloudy64-1:~# docker pull ubuntu
latest: Pulling from ubuntu
dbcb51e048f9: Pull complete
4e910c38549a: Pull complete
d43cf1f769e9: Pull complete
a572fb20fc42: Pull complete
Digest: sha256:b4dbab2d8029edddfe494f42183de20b7e2e871a424ff16ffe7b15a31f102536
Status: Downloaded newer image for ubuntu:latest
#Executem el contenidor amb Ubuntu
root@cloudy64-1:~# docker run -i -t ubuntu /bin/bash
root@c64ddb1ab8c4:/#
root@c64ddb1ab8c4:/# hostname
c64ddb1ab8c4
root@c64ddb1ab8c4:/# lsb_release -a
Distributor ID: Ubuntu
Description:    Ubuntu 14.04.4 LTS
Release:        14.04
Codename:       trusty
root@c64ddb1ab8c4:/#
```

Al Docker Hub hi ha un publicat amb una ocupació inferior als 60MB.

Aquest contenidor ens aporta moltes solucions, com per exemple, el test d'aplicacions o procediments.

Aquesta imatge de Ubuntu es "neteja" cada vegada que sortim del contenidor ja que els canvis

```
#Executem el contenidor amb Ubuntu, escrivim un fitxer i sortim
root@cloudy64-1:~# docker run -i -t ubuntu /bin/bash
root@c995c38d975f:/# echo prova > prova.txt
root@c995c38d975f:/# cat prova.txt
prova
root@c995c38d975f:/# exit
#Tornem a executar el mateix contenidor i mostrem el fitxer
root@cloudy64-1:~# docker run -i -t ubuntu /bin/bash
root@a78f220a8461:/# cat prova.txt
cat: prova.txt: No such file or directory   #El fitxer no existeix
root@a78f220a8461:/#
```

no són permanents.

4.1.3. Sandstorm

Després de realitzar l'anàlisi de Sandstorm es planteja la possibilitat de tractar com alternativa la seva distribució en un contenidor Docker.

Al Docker Hub hi ha publicades una desena de contenidors amb Sandstorm, però cap d'aquestes a funcionat sobre l'entorn de test. A més, la documentació relacionada a aquests contenidors era inexistent impossibilitant l'opció de depurar els possibles errors.

Entrant al bash d'alguns d'aquests contenidors:

```
root@cloudy64-3:~# docker exec -t -i 3f18ce5c1926 /bin/bash
```

Es pot veure com existeix el procés arrencat, però sense les credencials del contenidor, no és possible reiniciar-lo o parametritzar-lo.

Es deixa per més endavant la possibilitat de crear un contenidor amb Sandstorm propi i optimitzat per funcionar sobre Cloudy.

4.1.4. Owncloud

Owncloud és un sistema d'emmagatzemament de fitxers que implementa sincronització, accés remot i compartició de dades.

A diferencia del sistema de fitxers Tahoe-LAFS que s'inclou a Cloudy, ownCloud no és un Filesystem distribuït, ja que les dades només s'emmagatzemen a un node. D'altra banda, permet la federació entre diferents sistemes ownCloud per facilitar la compartició de dades.

Tot i no ser una sistema distribuït, crec que s'adapta a la filosofia de Cloudy en entorns com Guifi, on un usuari té el seu propi node i comparteix recursos amb la resta. ownCloud facilita aquesta tasca al compartir fitxers.

S'instal·la el contenidor i s'inicia:

```
#Descarreguem el contenidor de owncloud
root@cloudy64-5:~# docker pull owncloud

#Executem el contenidor indicant la versió 8.1 de owncloud i que utilitzi el port 8062
root@cloudy64-5:~# docker run -d -p 8062:80 owncloud:8.1
```

Verifiquem que el contenidor s'ha aixecat al port correcte:

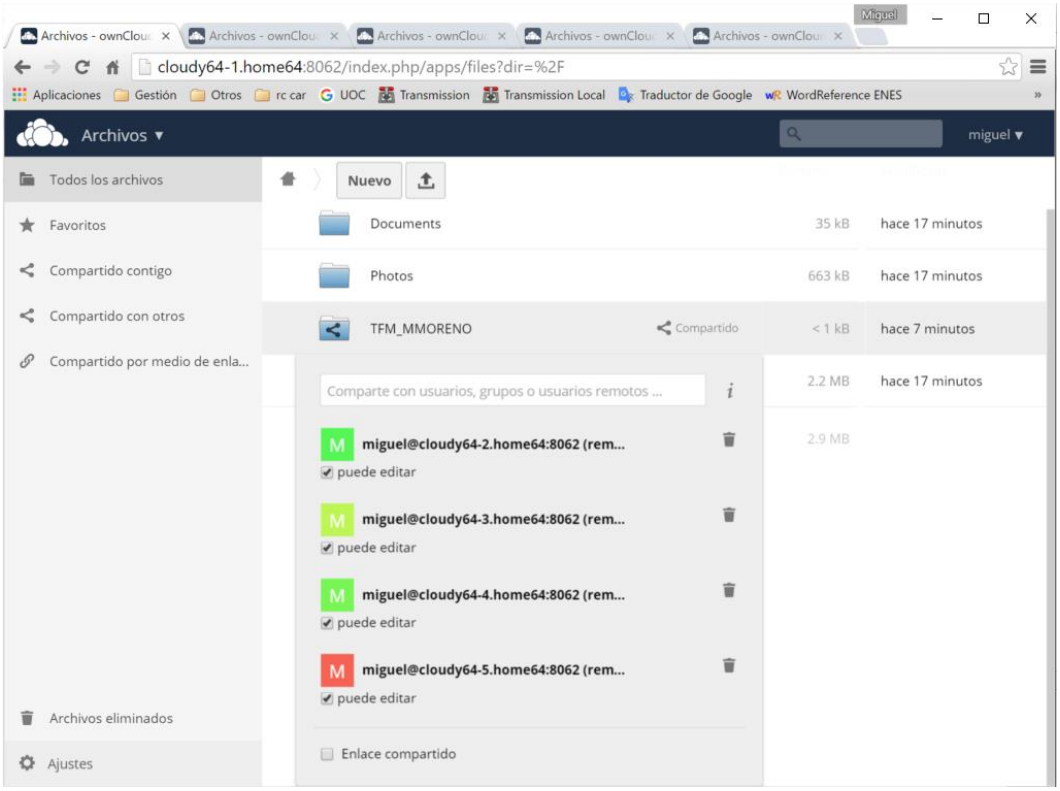
```
root@cloudy64-3:~# docker ps
CONTAINER ID   IMAGE                      COMMAND                  STATUS     PORTS
3f18ce5c1926   mplewis/sandstorm:latest   "/bin/sh -c '/home/s   Up         6080/tcp
f5658066a1a1   owncloud:8.1               "/entrypoint.sh apac   Up         0.0.0.0:8062->80/tcp
root@cloudy64-3:~#
```

Repetim la instal·lació als 5 nodes de test i es continua la configuració des del navegador:

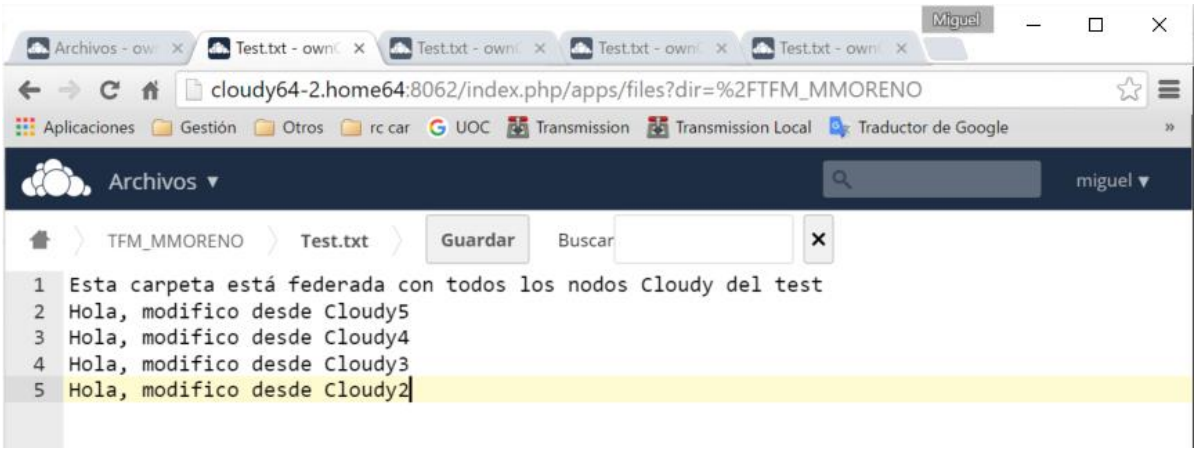
El contenidor emmagatzema les dades al següent path per defecte: /var/www/html/data.

Per defecte es fa servir una base de dades SQLite, que proporciona una pitjor experiència d'usuari i es recomana fer servir una BBDD externa. Per a la nostra prova ho deixem per defecte.

Per les proves, crearem una carpeta i la federarem amb la resta de nodes:



I de forma molt àgil ja tenim accés a aquesta carpeta des dels 5 nodes de Cloudy. Els fitxers de la carpeta són editables des de tots 5 nodes:



Durant les proves, es decideix apagar el contenidor que conté ownCloud del node Cloudy64-1. Des d'aquest moment, la informació de la carpeta federada entre tots el nodes, ja no és accessible des de la resta.

Recordem que federar permet compartir fitxers, però les dades només resideixen al ownCloud origen.

D'altra banda, al tornar a arrencar el contenidor al node Cloudy64-1 veiem que les dades ja no existeixen. Al realitzar la configuració per defecte, no hem seleccionat un Filesystem persistent per les dades i aquestes s'han perdut.

4.2. Entorns gràfics de Gestió

4.2.1. Kitematic

Kitematic és l'entorn gràfic oficial de Docker i es pot trobar informació al web. Es tracta d'una aplicació d'escriptori i està disponible per Windows i mac os.

Segons alguns fòrums a Docker i d'altres especialitzats, és possible arribar a arrencar aquest GUI també en entorns Linux. Existeixen tutorials per arrencar-lo sobre Ubuntu, però no funciona sobre el Debian que fa servir Cloudy.

Després d'algunes proves es desestima continuar intentant-ho ja que en el cas de que arribi a funcionar, segurament no sigui una opció que aporti valor ja que la gestió de Cloudy no està orientada a fer servir les X dels nodes, sinó que es decanta per la gestió web ja que facilita la connexió remota.

4.2.2. DockerUI

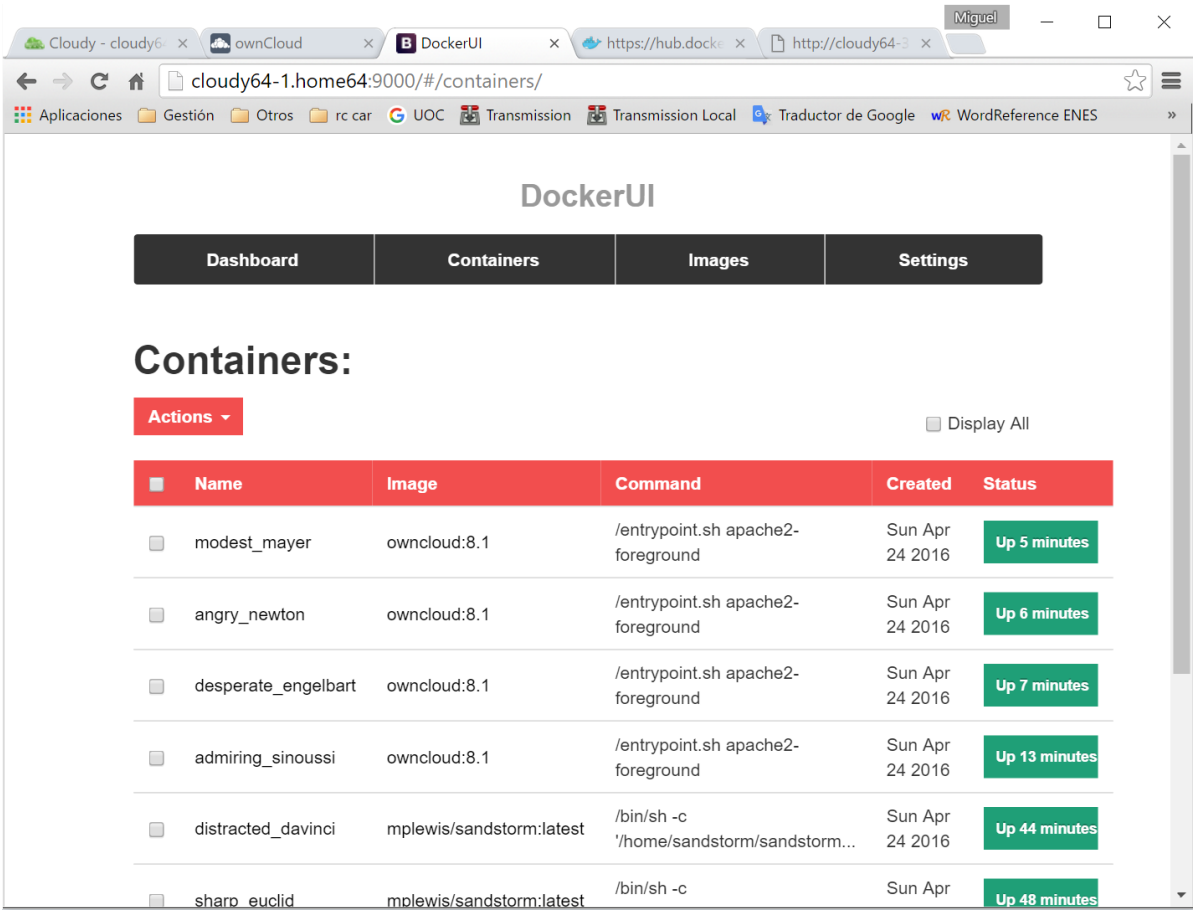
DockerUI és una eina de monitorització i gestió bàsica que funciona realment be. A diferencia de Kitematic, DockerUI permet la gestió de Docker des de un navegador web en remot.

Com a gran avantatge veiem que es pot desplegar en forma de contenidor, per contra trobem diferents limitacions, com per exemple que només podem gestionar contenidors instal·lats (Arrencar, parar, esborrar), però no podem realitzar noves instal·lacions.

```
#Descarreguem el contenidor
docker pull abh1nav/dockerui:latest

#L'executem
docker run -d -p 9000:9000 -v /var/run/docker.sock:/docker.sock \
--name dockerui abh1nav/dockerui:latest -e="/docker.sock"
```

La trobem al Docker Hub: https://hub.docker.com/r/abh1nav/dockerui/

I ja es pot gestionar des del navegador:



Durant les proves es fa servir aquesta GUI per parar i arrencar contenidors amb resultats satisfactoris. Però no cobreix totes les necessitats per administrar Docker, sent necessari accedir a la línia de comandes molt sovint.

4.2.3. Shipyard

Com en el cas de DockerUI, Shipyard és una aplicació de gestió web per Docker basada en contenidors.

Però en aquest cas es tracta d'una aplicació molt més complerta, que permet gestionar de forma gairebé integral Docker al host local i als hosts remots.

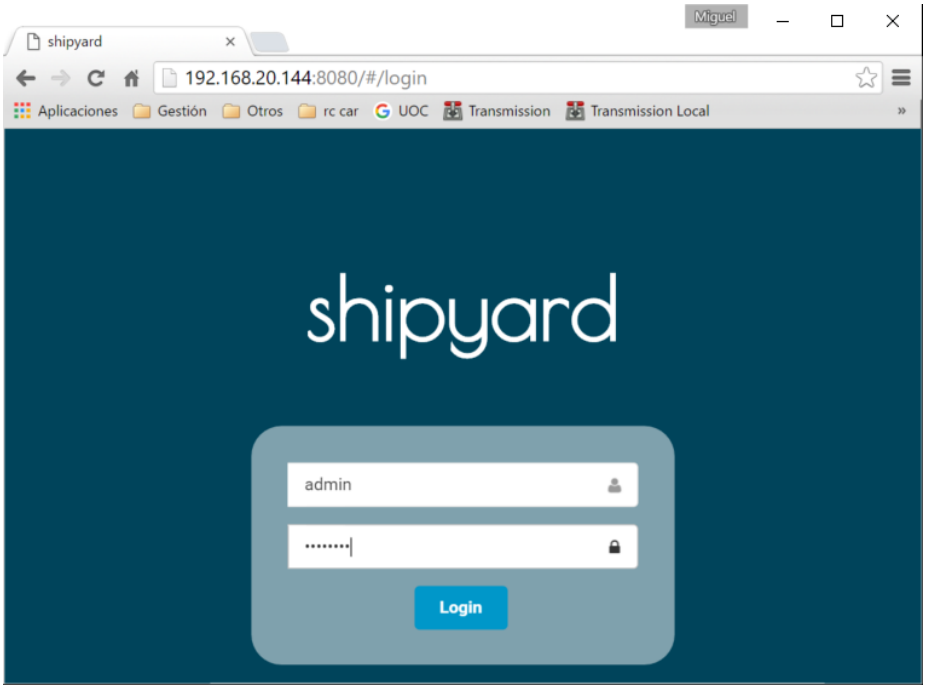
La instal·lació és fàcil per al node principal ja que es posa a la nostra disposició un script que automatitza la descarrega de contenidors i la seva parametrització:

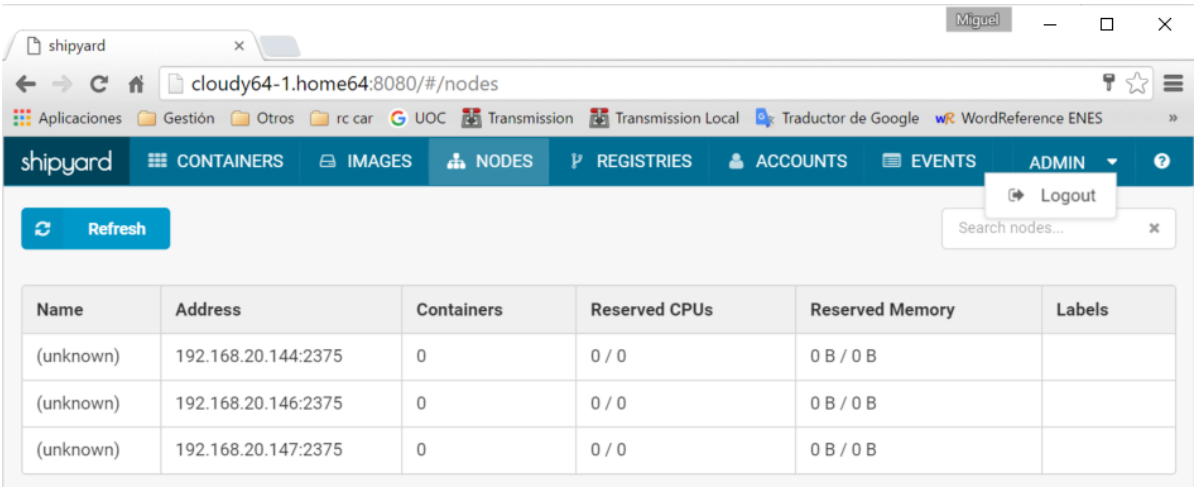
```
#Iniciem el script que descarrega els contenidors i parametritza l'entorn
root@cloudy64-1:~# curl -sSL https://shipyard-project.com/deploy | bash -s
```

Per afegir la resta de nodes a la mateixa consola, executem la següent comanda:

```
#Iniciem el script que descarrega els contenidors i parametritza l'entorn
root@cloudy64-2:~# curl -sSL https://shipyard-project.com/deploy | ACTION=node
DISCOVERY=etcd://192.168.20.144:4001 bash -s
```

Ja podem accedir a l'eina de gestió:





Tot i això, shipyard no acaba de funcionar en l'entorn Cloudy. Tot i que és possible veure els nodes afegits per gestionar, no es permet realitzar tasques d'administració ni desplegament d'aplicacions ja que sempre mostra aquest missatge d'error:

> **Error...**
> No healthy node available in the cluster

Investigant a la xarxa, s'associa aquest error a problemes de comunicació entre nodes, però des de la línia de comandes verifiquem que la comunicació és correcta:

```
#Des de el node 1 consultem la informació del node 3
root@cloudy64-1:~# docker -H cloudy64-3:2375 info
Containers: 9
Images: 112
Storage Driver: aufs
 Root Dir: /var/lib/docker/aufs
 Backing Filesystem: extfs
 Dirs: 130
 Dirperm1 Supported: true
Execution Driver: native-0.2
Kernel Version: 3.16.0-4-amd64
Operating System: Debian GNU/Linux 8 (jessie)
CPUs: 1
Total Memory: 481 MiB
Name: cloudy64-3
ID: X3RR:USM3:B4RE:YJL3:WTLY:WSDG:EETD:7YYH:JLCS:URBI:HQFP:55BH
WARNING: No memory limit support
WARNING: No swap limit support
root@cloudy64-1:~#
```

Tot i aquest problema, sembla que és l'eina que millor cobreix les necessitats identificades, per tant s'investigarà en aquesta línia per depurar els errors existents.

5. Conclusions

- Definitivament, Docker pot aportar a Cloudy una gran flexibilitat per desplegar nous serveis. Però, per que sigui realment efectiva en un cloud col·laboratiu, és necessari implementar una eina de gestió gràfica per Docker.
- A la presentació de Docker s'explica que un contenidor pot executar-se a qualsevol host gracies a que és totalment autònom alhora que estalvia l'ús de recursos. Però amb les proves que hem fet veiem que això no és sempre així, i un contenidor pot necessitar adaptacions depenent del host.
  Tot i això, Docker continua sent una bona opció per distribuir aplicacions.
- De les aplicacions provades, veiem que ownCloud encaixa perfectament amb la filosofia de Cloudy i seria una eina molt interessant. D'altra banda, podem explotar la capacitat dels contenidors per "autodestruir-se" amb contenidors com Ubuntu per executar proves sobre entorns nets.

Annex IV: Informe Sprint 1

1. Introducció

Aquest informe té com a objectiu el seguiment de les tasques de projecte enregistrant la seva evolució. Amb aquesta finalitat, es documentaran les tasques finalitzades i es seleccionaran de noves per a executar al següent Sprint.

La periodicitat estimada dels Sprints és d'una setmana.

2. Tasques planificades

Aquest és el llistat de tasques planificades per al primer Sprint:

ID Tasca	Descripció Tasca	Estat
Tsk01	Identificar com la web de Docker pot interactuar amb el host per rebre informació d'estat de Docker i parsejar-la	Completada
Tsk02	Configurar Serf en Cloudy per descobrir serveis publicats per altres nodes	Completada
Tsk03	Descobrir com Cloudy publica serveis a través de Serf	Completada
Tsk04	Seguir la guia per instal·lar i publicar el software pastecat amb l'objectiu d'entendre tot el procés	Completada
Tsk05	Obtenir informació dels contenidors instal·lats i mostrar-ho de forma estructurada a la web	Completada
Tsk06	Afegir un boto per a cada contenidor detectat per parar-ho o arrencar-ho depenent del seu estat	Completada
Tsk07	Afegir un boto per a cada contenidor detectat per eliminar-lo si està parat	Completada

A continuació es detalla el resultat de cada tasca:

Tsk04: Pastecat

Es decideix executar en primer lloc aquesta tasca, ja que l'objectiu final no és obtenir un resultat tangible sinó adquirir coneixements necessaris per a resta de tasques.

Es segueix la guia publicada a: http://wiki.clommunity-project.eu/howto:installpastecat

Seguint la guia pas a pas s'aconsegueix l'objectiu i es descobreix l'estructura bàsica de directoris i fluxos de treball.

Al finalitzar la tasca s'esborren els fitxers creats ja que no es considera que aquesta aplicació aporti res a la resta d'implementació planificada.

Tsk01: Interface Docker-web

Es detecta que tots els fitxers per gestionar les aplicacions que inclou la distribució Cloudy estan ubicats al path: `var/local/cDistro/plug/`

`/var/local/cDistro/plug/menús:` Un fitxer per menú de la web
`/var/local/cDistro/plug/controller:` Un fitxer per implementar la lògica de cada aplicació

```
root@cloudy64-5:/var/local/cDistro/plug/controllers# ls -la
total 240
drwxr-sr-x 2 root staff  4096 Mar 20 14:51 .
drwxr-sr-x 6 root staff  4096 Mar 20 14:51 ..
-rw-r--r-- 1 root staff  3483 Mar 20 14:51 avahi.php
-rw-r--r-- 1 root staff 11352 Mar 20 14:51 cloudyupdate.php
-rw-r--r-- 1 root staff  4154 Mar 20 14:51 default.php
-rw-r--r-- 1 root staff  2074 Mar 20 14:51 docker.php
-rw-r--r-- 1 root staff  5119 Mar 20 14:51 getinconf.php
-rw-r--r-- 1 root staff  9545 Mar 20 14:51 guifi-dnss.php
-rw-r--r-- 1 root staff 12739 Mar 20 14:51 guifi-proxy3.php
-rw-r--r-- 1 root staff  8089 Mar 20 14:51 guifi-snps.php
-rw-r--r-- 1 root staff 28293 Mar 20 14:51 guifi-web.php
-rw-r--r-- 1 root staff  4190 Mar 20 14:51 kimchi.php
-rw-r--r-- 1 root staff  3518 Mar 20 14:51 owp.php
-rw-r--r-- 1 root staff 12023 Mar 20 14:51 peerstreamer.php
-rw-r--r-- 1 root staff 11105 Mar 20 14:51 serf.php
-rw-r--r-- 1 root staff  7163 Mar 20 14:51 settings.php
-rw-r--r-- 1 root staff  2603 Mar 20 14:51 sshkeys.php
-rw-r--r-- 1 root staff  1712 Mar 20 14:51 ssl.php
-rw-r--r-- 1 root staff  7082 Mar 20 14:51 syncthing.php
-rw-r--r-- 1 root staff 54504 Mar 20 14:51 tahoe-lafs.php
-rw-r--r-- 1 root staff 13327 Mar 20 14:51 webdav.php
```

Investigant en aquests fitxers de “controladors” veiem que és possible enviar comandes al host amb el següent codi al fitxer PHP:

```
$ret = execute_program($cmd);
```

On $cmd és la comanda a executar i $ret emmagatzema la sortida de l’execució.

Per fer proves s’insereix la següent línia a la funcionalitat d’un botó existent:

```
$cmd = “touch /home/miguel/test/file.test”
$ret = execute_program($cmd);
```

I es comprova que efectivament el fitxer existeix al host:

```
miguel@cloudy64-3:/$ ls -l /home/miguel/test
-rw-r--r-- 1 root 0 May  1 03:05 file.test
```

Tsk02: Configuració Serf

Serf és una eina descentralitzada per gestionar els nodes d'un clúster, incloent la detecció de fallades i orquestració. L'eina és molt lleugera i utilitza molt poc tràfic de xarxa gracies a la implementació de un protocol gossip. Aquesta manera de funcionar fa que els nodes tinguin informació possiblement desfasada dels altres nodes més allunyats, fet que és assumible tenint en compte que una sincronització més exhaustiva pot ser inviable.

Serf permet la propagació de missatges entre nodes. Cloudy aprofita aquesta funcionalitat per publicar els serveis que té cada node.

El funcionament és simple, cada node distribueix un missatge amb els serveis que publica amb el seu estat i cada node manté una taula actualitzada amb els serveis que han publicat la resta de nodes.

Configuració de serf a Cloudy:

Serf està instal·lat i preconfigurat als nodes Cloudy, només hem d'arrencar el servei i decidir qui serà el node principal (tots poden ser-ho, farem servir el node Cloudy64-1):

```
#Iniciem el servei al node Cloudy1-64
root@cloudy64-1:~# /opt/serf/serf agent
```

```
#Iniciem el servei a la resta de nodes indicant el node principal
root@cloudy64-2:~# /opt/serf/serf agent -join=cloudy64-1
```

Als pocs segons, serf ja està rebent informació dels serveis publicats a tots els nodes:

```
#Mostrem els membres del clúster [El text services apareix truncat]
root@cloudy64-1:~# /opt/serf/serf members
cloudy64-1 192.168.20.144:7946 alive services=QlpoOT...  ...weM1bg==
cloudy64-2 192.168.20.145:7946 alive services=QlpoOT...  ...w6dzAA==
cloudy64-3 192.168.20.146:7946 alive services=QlpoOT...  ...oSAABSIa
cloudy64-4 192.168.20.147:7946 alive services=QlpoOT...  ...YLaeoA==
cloudy64-5 192.168.20.148:7946 alive services=QlpoOT...  ...USciiwA=
```

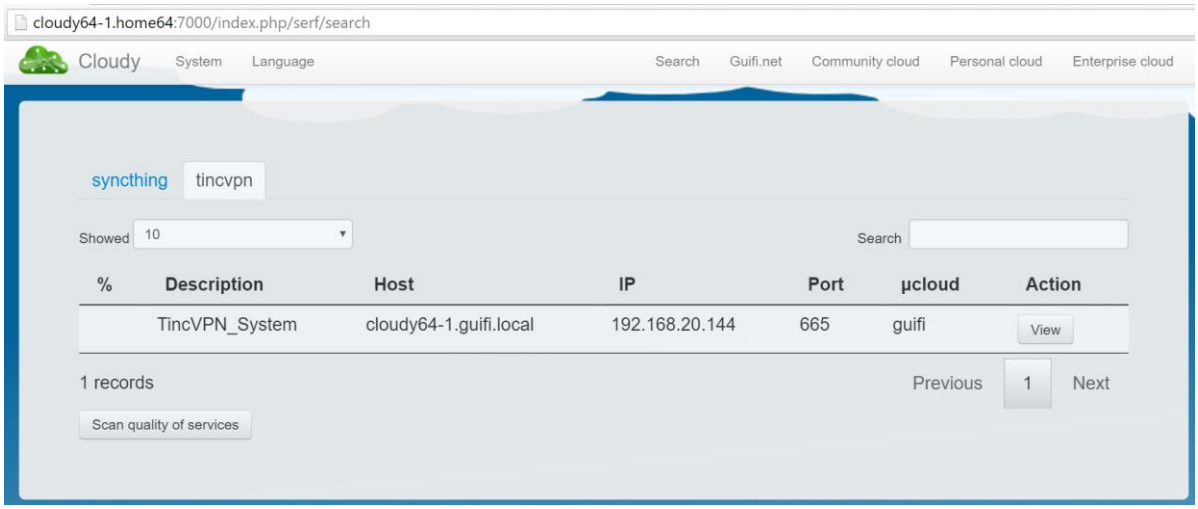
Es pot veure que els serveis es publiquen codificats amb base64 i comprimits amb bzip, per poder llegir el contingut podem executar la següent comanda (Facilitada per l'Agustí Moll)

```
root@cloudy64-1:~# echo
"QlpoOTFBWSZTWf6pmrcAAGVfgAAQEId/8if/ffqv598qMADWzYaim9Im1D1MhoaDQAPUDyaEAA1T0
p6ZJp4p6QANAAABtRp6jIEqmETQbRAAYgA0AaAaHqbpHMmc1yA80/hEz0hoIoKSM2e2wzFoBLWtbGJ
CwRBCGJTDCOWRxrCfbknmzu3SQykpeFrEIkJQiizCh1WzgLg1iyDsPdX+DCVAHSRoBBBjKMbSueiJq
ckea05A6T34abzUQaqHg6KaVSQ+8+VbB2lsYa/hU1RSFUuFjjpDafYPbkNAY3E9IOfVFEkCKBCOAhH
05HIXIwHooqlDQIwoV1K9spcBIVWPKoFoH1tgUp19KkUEAq0Wwau4PTDaKdT+LuSKcKEh/VM1bg=="
| base64 -d - | bunzip2 -
[{"s":"tincvpn","d":"TincVPN_System","m":"cloudy64-
1.guifi.local","i":"192.168.20.144","p":"665","e":"guifi","t":""},{'s':'syncth
ing','d':'Syncthing_instance_running','m':'cloudy64-
1.guifi.local','i':'192.168.20.144','p':'22000','e':'guifi','t':'node_id=JLJI3
HX-H5E2NJY-7VJLD7A-GXDQ64D-WADSI3Y-ZJJPS4N-DNV6FMK-JVPFRA5'}]
```

Es a dir, el node Cloudy64-1 està publicant els següents serveis:

Servei	Descripció	IP	Port	e (?)	t (?)
tincvpn	TincVPN_System	192.168.20.144	665	guifi	""
syncthing	Syncthing_instance_running	192.168.20.144	22000	guifi	node_id=JLJI3HX-H5E2NJY-7VJLD7A-GXDQ64D-WADSI3Y-ZJJPS4N-DNV6FMK-JVPFRA5

O des del web:



Tsk03: Publicació de serveis

A la documentació sobre la instal·lació de pastecat, es troben referencies a l'eina avahi.

Fent una cerca es troba la següent comanda: /usr/sbin/avahi-ps amb la següent funcionalitat:

root@cloudy64-3:~# /usr/sbin/avahi-ps

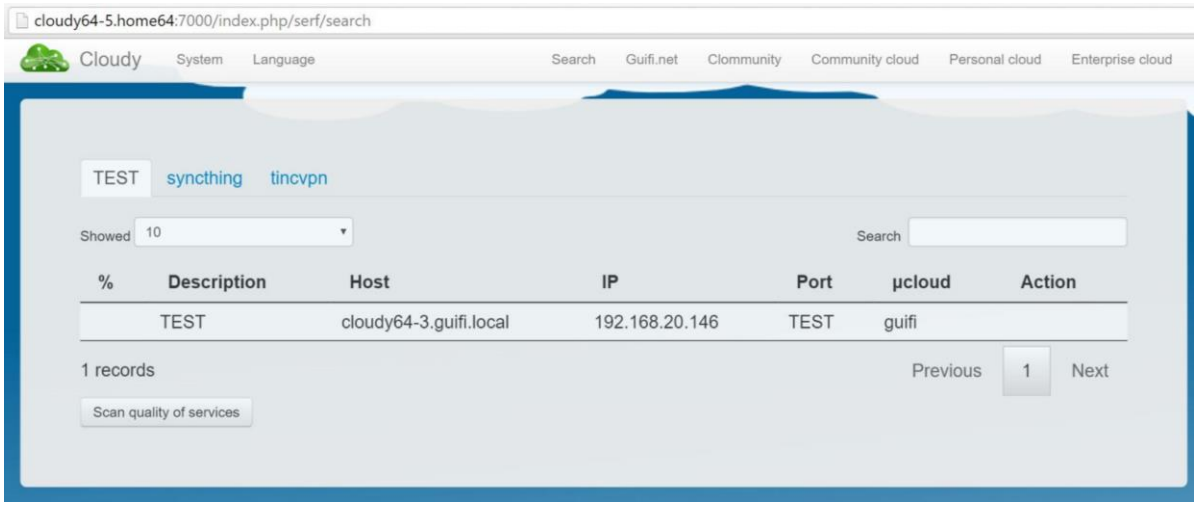
```
root@cloudy64-3:~# /usr/sbin/avahi-ps
Avahi - Publish and Search: System to publish avahi services or Search
existents services.

Use: /usr/sbin/avahi-ps publish|search|unpublish|info <options>
Use: /usr/sbin/avahi-ps publish <describe> <type> <port> [txt]
       <describe>: Text describe service.
       <type>: Service type.
       <port>: Service port.
      <txt>:              Other              information,              format
            'attibute1=value1&attribute2=value2&...&attributeN=valueN'.
Use: /usr/sbin/avahi-ps search [type] [hostname]
Use: /usr/sbin/avahi-ps unpublish [type] [port]
Use: /usr/sbin/avahi-ps info <variable>
       <variable>: informations ip, cloud, tincdev, communitydev.
```

Bàsicament veiem com té la capacitat de publicar i despublicar indicant una sèrie de paràmetres. Verifiquem el seu funcionament executat una prova:

```
root@cloudy64-3:~# /usr/sbin/avahi-ps publish TEST TEST TEST TEST
```

On els paràmetres “Descripció”, “Tipus”, “Port” i “txt” s’informen amb el literal “TEST”

Obtenint el següent resultat:



A la imatge veiem com des del node Cloudy64-5 estem veient una nova pestanya de serveis anomenada TEST on trobem el servei TEST publicat al node Cloudy64-3 al Port TEST. Verificant, per tant, que la publicació de serveis es pot gestionar fent servir aquest comanda.

Executant el següent codi es despublica el servei:

```
root@cloudy64-3:~# /usr/sbin/avahi-ps publish TEST TEST
```

Tsk05: Mostrar informació Docker

Per realitzar aquesta tasca aprofundim en la tècnica provada a la tasca Tsk01, on es veia com executar comandes al host obtenint el resultat.

Amb aquest objectiu afegim la següent línia al codi existent del controlador de Docker (/var/local/cDistro/plug/controllers/Docker.php):

A la funció index() s’avaluen tres casuístiques:

- Docker no està instal·lat -> Mostra un botó per instal·lar-ho
- Docker no està arrencat -> Mostra un botó per arrencar-ho
- Docker està arrencat -> Mostra informació i un botó per parar-ho

En aquesta última opció afegim la crida a una nova funció amb l’objectiu de mostrar informació estructurada de Docker:

```
function index() {
        global $title, $urlpath, $docker_pkg, $staticFile;

        $page = hlc(t("docker_title"));
        $page .= hl(t("docker_desc"), 4);

        if (!isPackageInstall($docker_pkg)) {
                [...]

        } elseif (!isRunning()) {
                [...]
        } else {
                [...]

                //Codi modificat: Docker GUI
                $page = docker_GUI($page);

                return array('type' => 'render','page' => $page);
        }
}
```

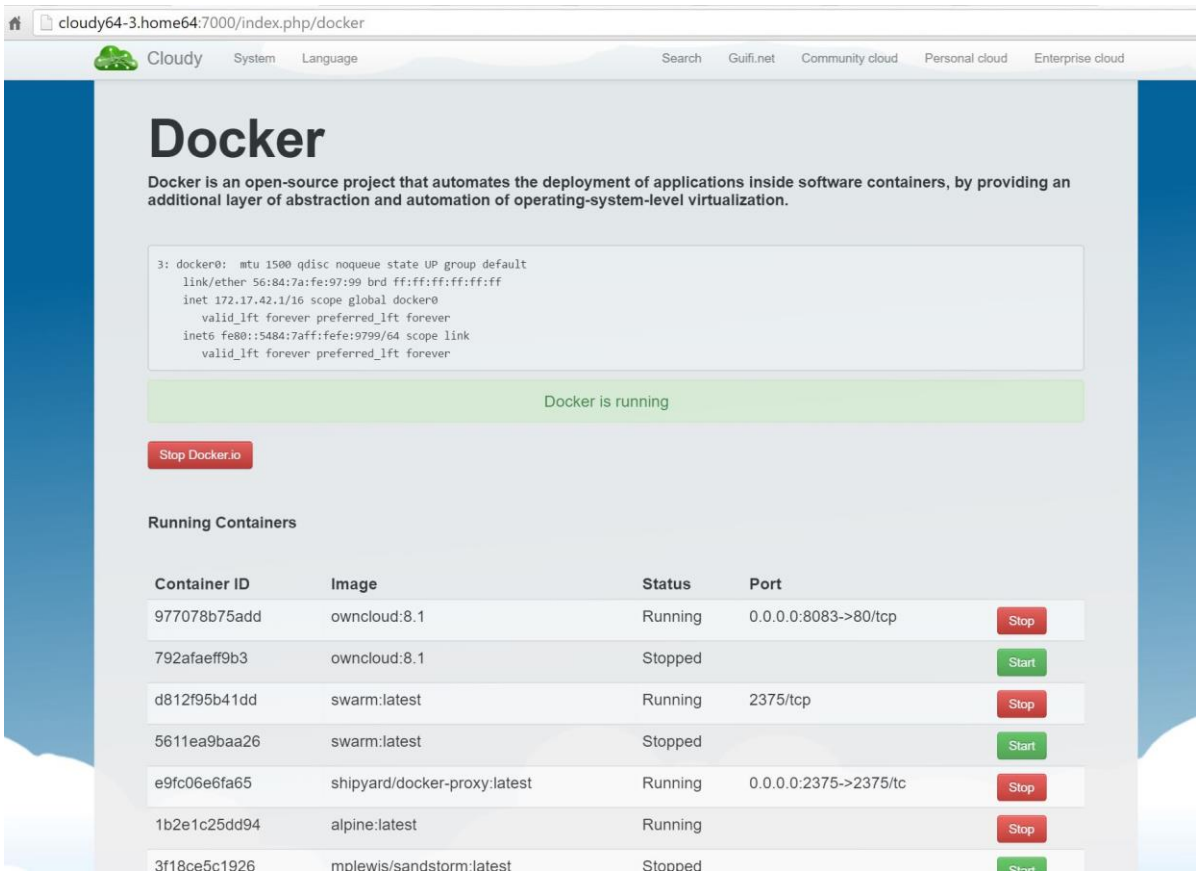
Amb aquesta crida estem afegint a la plana web la informació resultant de l'execució de la funció Docker_GUI().

Implementació de la funció **docker_GUI()**:

```
function docker_GUI($page_tmp){
  global $dev,$urlpath;

  //Executem la comanda per obtenir informació de contenidors en execució
  $cmd = "docker ps -a";
  $ret = execute_program($cmd);

  //Mostrem el títol de la secció
  $page_tmp .= hl("", 4);
  $page_tmp .= hl("Running Containers", 4);

  //Creem la taula on mostrarem tota la informació obtinguda
  //es fa servir l'estil de la resta de la web
   $page_tmp    .=    addTableHeader(array("Container    ID","Image","Status","Port"),
       array('class'=>'table table-striped'));

  //Fem un recorregut per cada línia del output de l'execució
  $first=true;
  foreach ($ret['output'] as $container){
       //Ometem la primera línia del output amb informació de capçaleres
       if($first){ $first=false; }
       else{
               $cId=substr($container,0,12);   //Obtenim ID
               $cIm=substr($container,20,30);  //Obtenim Imatge
               $cSt=substr($container,94,2);   //Obtenim Status
               $cPr=substr($container,120,22); //Obtenim Port
```

```
            if($cSt=="Up"){
                //Si l'estatus és UP mostrem botó per parar
                $cSt="Running";;
                $boton=addButton(array('label'=>"Stop",'class'=>'btn  btn-danger',
                    'href'=>"$urlpath/containerStop/".$cId."_".$cIm));
                $boton2=addButton(array('label'=>"Delete",   'class'=>'btn   btn-
                    default disabled'));
            }else{
                //Si no, mostrem botó per arrencar
                $cSt="Stopped";
                $boton=addButton(array('label'=>"Start",'class'=>'btn        btn-
                    success', 'href'=>"$urlpath/containerStart/".$cId."_".$cIm));
                $boton2=addButton(array('label'=>"Delete",'class'=>'btn      btn-
                    danger', 'href'=>"$urlpath/containerDelete/".$cId));
            }

            //Afegim la fila a la taula amb informació del contenidor
            $page_tmp   .=   addTableRow(array($cId,   $cIm,   $cSt,   $cPr,   $boton,
                $boton2));
        }
    }
    //Tanquem la taula
    $page_tmp .= addTableFooter();

    return ( $page_tmp );
}
```

Obtenint el següent resultat:



Tsk06: Parar-arrencar Contenidors

S'implementen les funcions que criden els botons de la Tsk05:

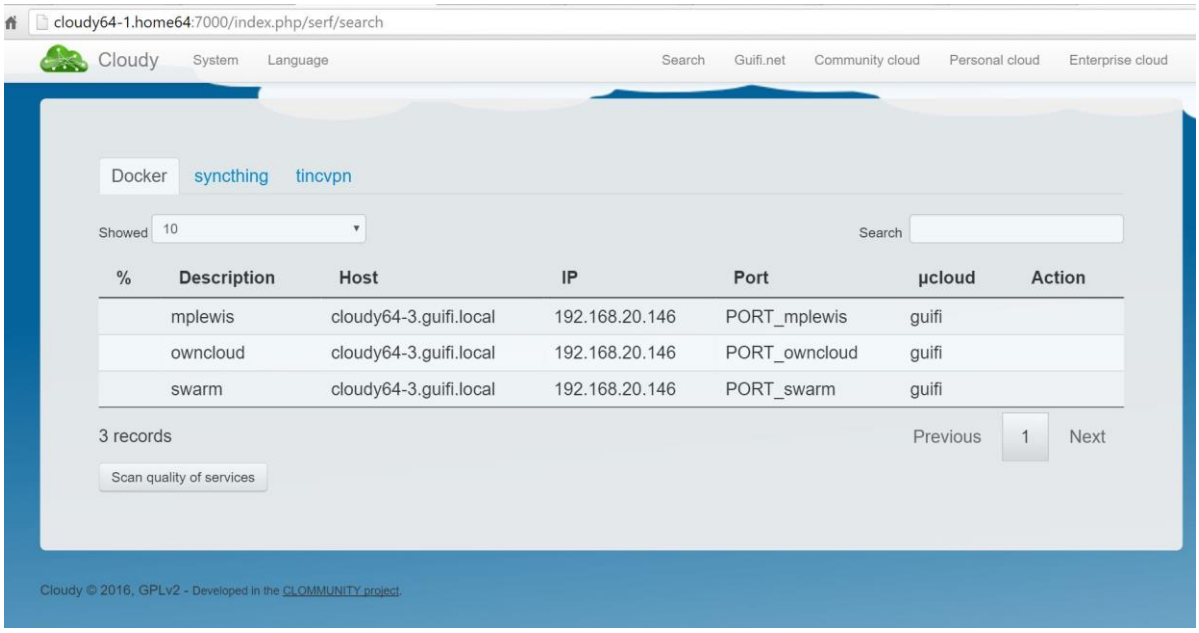
containerStart() rep com a paràmetre el ID del contenidor i el nom de la imatge. Fa servir el ID per arrencar el contenidor, i el nom de la imatge per publicar el servei a Serf.

```
function containerStart() {
        global $urlpath,$Parameters;

        //Obtenim els dos paràmetres que estan separats per "_"
        $param = $Parameters[0];
        $p = explode("_",$param);
        $cId = $p[0];           //Obtenim ID
        $cImTemp = explode(":",$p[1]);
        $cIm = $cImTemp[0];     //Obtenim Imatge

        //Executem la comanda docker start
        execute_program_detached("docker start CONTAINER ".$cId);

        //Mostrem un missatge per pantalla mostrant el success
        setFlash("Container Started","success");

        //Publiquem el nom del contenidor a Serf
        execute_program_detached("/usr/sbin/avahi-ps     publish     ".$cIm."     Docker
            PORT_".$cIm);

        //Mostrem un missatge per pantalla confirmant la publicació
        setFlash("Container Published ".$cIm."success");

        return(array('type'=> 'redirect', 'url' => $urlpath));
}
```

containerStop() rep com a paràmetre el ID del contenidor i el nom de la imatge. Fa servir el ID per parar el contenidor, i el nom de la imatge per despublicar el servei a Serf.

```
function containerStop() {
        global $urlpath,$Parameters;

        //Obtenim els dos paràmetres que estan separats per "_"
        $param = $Parameters[0];
        $p = explode("_",$param);
        $cId = $p[0];           //Obtenim ID
        $cImTemp = explode(":",$p[1]);
        $cIm = $cImTemp[0];     //Obtenim Imatge

        //Para el contenidor i despublica a Serf
        execute_program_detached("docker stop CONTAINER ".$cId);
        execute_program_detached("/usr/sbin/avahi-ps unpublish Docker PORT_".$cIm);
        setFlash("Container Stopped. ","success");
        return(array('type'=> 'redirect', 'url' => $urlpath));
}
```

Al arrencar contenidors al node Cloudy64-3 es publiquen a Serf (vist de del node Cloudy64-1):



Tsk07: Eliminar Contenidors

S'implementa la última funció que criden els botons de la Tsk05:

containerDelete() rep com a paràmetre el ID del contenidor i el fa servir per eliminar el contenidor.

```
function containerDelete() {
        global $urlpath,$Parameters;
        $cId = $Parameters[0]; //Obtenim ID

        //Executem l'eliminació del contenidor
        execute_program_detached("docker rm ".$cId);

        setFlash("Container Deleted. ","success");
        return(array('type'=> 'redirect', 'url' => $urlpath));
}
```

3. Tasques pendents

Aquesta és la llista de tasques pendents, es marquen en verd les seleccionades per implementar al següent Sprint. Es marquen en blau noves tasques afegides:

ID Tasca	Descripció Tasca	Estat
Tsk08	Afegir un botó amb un link cap a la plana d'administració del contenidor	Planificada
Tsk09	Afegir un boto per a cada contenidor detectat per publicar-lo	Planificada
Tsk10	Crear una taula que guardi si hem decidit publicar un determinat contenidor	Planificada
Tsk11	Crear una taula de contenidors recomanats amb un botó associat que descarregui i inicií cada contenidor	Planificada
Tsk12	Descobrir com Cloudy detecta canvis en els serveis publicats al node local	Pendent

Tsk13	Si s'arranca un contenidor que hem decidit que volem publicar, publicar-ho directament	Pendent
Tsk14	Si es para o cau un contenidor que hem decidit que volem publicar, despublicar-ho directament	Pendent
Tsk15	Crear un interface per poder afegir contenidors a la llista de contenidors recomanats	Pendent
Tsk21*	Investigar si existeixen contenidors Docker que cobreixin les funcionalitats proporcionades per les Apps de Sandstorm	Pendent (Nova)
Tsk16	Crear un nou menú a la web d'administració de Cloudy que permeti instal·lar Sandstorm al node	Pendent
Tsk17	Afegir al nou menú el link d'accés a l'eina de gestió de Sandstorm	Pendent
Tsk18	Investigar com podem detectar les apps que Sandstorm té creades	Pendent
Tsk19	Crear un interfase que permeti publicar els serveis de Sandstorm de forma individual creat una taula que emmagatzemi els serveis a publicar	Pendent
Tsk20	Detectar canvis en els serveis de Sandstorm publicats per gestionar si es continuen publicant o no	Pendent

*S'afegeix una nova tasca: "Investigar si existeixen contenidors Docker que cobreixin les funcionalitats proporcionades per les Apps de Sandstorm". La necessitat sorgeix després d'una conversa amb el director de projecte, on es detecta la possibilitat de que els contenidors Docker ens pugin donar totes (o bona part) de les funcionalitats que aporta Docker, i per tant tindria més sentit potenciar encara més aquesta eina que no pas implementar Sandstorm.

Es planifica per executar just abans de la tasca Tsk16, on es plantejava iniciar les tasques d'integració de Sandstorm. El resultat d'aquesta tasca (en cas d'arribar-se a executar) serà determinar si es continua amb les tasques definides o es defineixen unes alternatives.

4. Resum

4.1.Sprint tancat

El resultat obtingut durant el primer Sprint ha estat satisfactori, s'han assolit les tasques planificades i s'ha avançat molt en la comprensió de la estructura de la distribució Cloudy. Actualment, es disposa d'un entorn gràfic per a la gestió bàsica de contenidors integrat amb la publicació de serveis de Cloudy mitjançant Serf.

Aquest entorn gràfic és només un esbós i segurament dista del seu disseny i funcionalitats finals, però estableix una línia base.

4.2.Sprint obert

Les tasques seleccionades al Sprint que s'inicia estan orientades a millorar l'experiència d'usuari al gestionar contenidors i com aquests es publiquen.

Annex V: Informe Sprint 2

1. Introducció

Aquest informe té com a objectiu el seguiment de les tasques de projecte enregistrant la seva evolució. Amb aquesta finalitat, es documentaran les tasques finalitzades i seleccionaran de noves per a executar al següent Sprint.

La periodicitat estimada dels Sprint és d'una setmana.

2. Tasques planificades

Aquest és el llistat de tasques planificades per al segon Sprint:

ID Tasca	Descripció Tasca	Estat
Tsk10	Crear una taula que guardi si hem decidit publicar un determinat contenidor	Completada
Tsk22	(Nova tasca) Dissenyar el interfície d'usuari	Completada
Tsk08	Afegir un botó amb un link cap a la plana d'administració del contenidor	Completada
Tsk09	Afegir un boto per a cada contenidor detectat per publicar-lo	Completada
Tsk11	Crear una taula de contenidors recomanats amb un botó associat que descarregui i inicií cada contenidor	Completada
Tsk21	Investigar si existeixen contenidors Docker que cobreixin les funcionalitats proporcionades per les Apps de Sandstorm	Completada

Com es pot veure, s'inicia en primer lloc la Tsk10. El motiu és que aquesta taula que es pretén crear pot servir de suport per als objectius de la resta de tasques del Sprint.

A més, s'ha afegit una tasca nova a la llista. Es degut a que després de crear aquesta taula auxiliar amb informació dels contenidors es detecta la necessitat de definir el disseny del interfície d'usuari, en cas contrari podríem haver de definir algunes funcions per aquest motiu.

A continuació es detalla el resultat de cada tasca:

Tsk10: Dissenyar taula amb informació de contenidors

Per continuar evolucionant l'entorn gràfic per Docker sorgeix la necessitat d'emmagatzemar informació auxiliar sobre els contenidors instal·lats. Dades com per exemple: si un contenidor s'ha de publicar o no, la URL d'administració, la URL de accés a la App, ...

S'avaluen dues opcions de base de dades per emmagatzemar la informació d'aquesta taula:

- Buscar una Base de Dades lleugera per instal·lar-la sobre la distribució Cloudy per donar suport a aquesta taula.

- Aprofitar l'entorn Docker disponible per desplegar un contenidor lleuger amb una base de dades bàsica.

En cap cas s'ha trobat un sistema de base de dades que consumeixi uns recursos que justifiquin la seva utilitat. Per tant, s'opta per fer servir un sistema de carpetes i fitxers per emmagatzemar la informació:

- Es crea la carpeta: /var/local/cDistro/plug/resources/docker/containers
- Aquí apareixerà un fitxer per cada container configurat
- I cada fitxer contindrà la informació del container concret

Com a exemple, es defineixen dos fitxers de test amb informació de dos contenidors que s'han provat fins ara: OwnCloud i DockerUI. El resultat és el següent:

```
root@cloudy64-3:/var/local/cDistro/plug/resources/docker/containers# ls -l
total 8
-rw-r--r-- 1 root staff 193 May  7 20:30 dockerui:lastest
-rw-r--r-- 1 root staff 105 May  7 20:30 owncloud:8.1
root@cloudy64-3:/var/local/cDistro/plug/resources/docker/containers# cat owncloud:8.1
       Name;OwnCloud
       Run;docker run -d -p 8083:80 owncloud:8.1
       Id;977078b75add
       Port;8083
       Img;owncloud:8.1
       Pub;Y
root@cloudy64-3:/var/local/cDistro/plug/resources/docker/containers# cat dockerui:last
       Name;DockerUI
       Run;docker run -d -p 9000:9000 -v /var/run/docker.sock:/docker.sock --name
           dockerui abh1nav/dockerui:latest -e="/docker.sock"
       Id;ndef
       Port;9000
       Img;abh1nav/dockerui:lastest
       Pub:N
root@cloudy64-3:/var/local/cDistro/plug/resources/docker/containers#
```

D'inici, es proposen els següents camps, però s'afegiran més paràmetres si es necessiten:

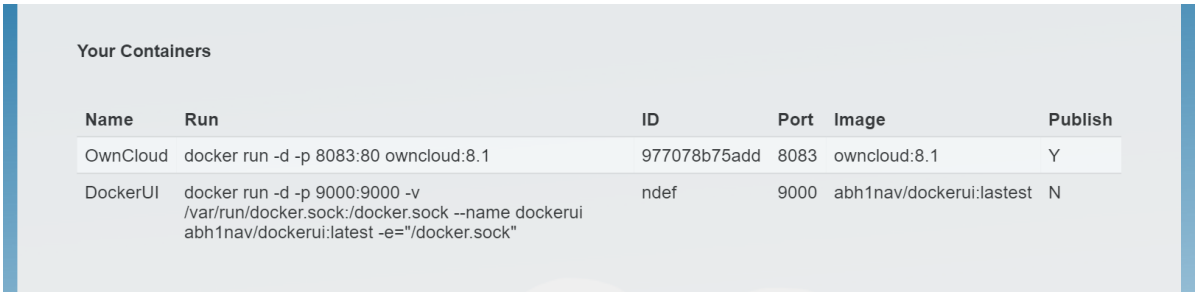
- Name: Nom arbitrari assignat al contenidor
- Run: Conté la comanda a executar per descarregar i iniciar el contenidor
- Id: Conté el identificador del contenidor.
- Port: Port de publicació extern
- Img: Nom de la imatge
- Pub: Defineix si es publica o no

Es defineix la següent funció per obtenir els paràmetres definits al fitxer donat un contenidor:

```
function get_container_info($image){
        global $dev,$urlpath;
      $containerspath="/var/local/cDistro/plug/resources/docker/containers/";

        //Llegim el fitxer de configuració
        $cmd = "cat ".$containerspath.$image;
        $properties = execute_program($cmd);
```

```
        //Existeix el fitxer de configuració?
        if(count($properties['output'])==1){
                return("0"); //Retornem Error, el fitxer no existeix
        }else{
                //Es recuperen les propietats
                foreach ($properties['output'] as $prop){
                        $prop=explode(";",$prop);
                        switch($prop[0]){
                                case "Name":
                                        $name=$prop[1]; break;
                                case "Run":
                                        $run=$prop[1]; break;
                                case "Id":
                                        $id=$prop[1]; break;
                                case "Port":
                                        $port=$prop[1]; break;
                                case "Img":
                                        $img=$prop[1]; break;
                                case "Pub":
                                        $pub=$prop[1]; break;
                        }
                }
        }
       //Retornem un array amb totes les propietats definides
       return(array('name'=>$name,'run'=>$run,'id'=>$id,'port'=>$port,'img'=>$
       img,'pub'=>$pub));
}
```

Es desenvolupa un petit codi de prova per mostrar per pantalla el resultat d'aquesta funció amb el següent resultat:



Your Containers

Name	Run	ID	Port	Image	Publish
OwnCloud	docker run -d -p 8083:80 owncloud:8.1	977078b75add	8083	owncloud:8.1	Y
DockerUI	docker run -d -p 9000:9000 -v /var/run/docker.sock:/docker.sock --name dockerui abh1nav/dockerui:latest -e="/docker.sock"	ndef	9000	abh1nav/dockerui:lastest	N

Tsk22: Disseny GUI

Partim de la base que l'objectiu d'aquesta plana no és ser un entorn gràfic per parar i arrencar contenidors ja instal·lats, per això ja existeixen eines com DockerUI, per exemple.

L'objectiu és facilitar als usuaris la gestió dels seus contenidors, facilitant la instal·lació dels mateixos i donant una visió enriquida que permeti accedir a les aplicacions i publicar-les a la comunitat.

Per tant, a partir d'ara no es farà servir com a origen de dades la informació que ens mostra Docker, sinó que farem servir els fitxers que s'acaben de crear a la tasca anterior. Cridant a Docker per enriquir la informació mostrada.

D'aquesta manera, ja no es mostraran els contenidors existents instal·lats des de línia de codi. Podríem resumir-ho dient: els contenidors que s'instal·len de d'aquest entorn gràfic, es poden gestionar des d'aquí, si s'instal·len per línia de comandes, s'han de gestionar igualment per línia de comandes.

Es crea el següent codi per consultar dades a Docker:

```
function get_docker_info($image){
        global $dev,$urlpath;

        //Consultem els contenidors existents
        $cmd = "docker ps -a";
        $ret = execute_program($cmd);

        //Fem servir la capçalera per trobar la posició de Status
        $status_pos = strpos($ret['output'][0],"STATUS");

        //Inicialitzem variables per si no trobem el contenidor
        $id=0;
        $status="Not Installed";
        //Busquem a cada contenidor
        foreach($ret['output'] as $container){
                if(strpos($container, $image)){
                        $id=substr($container,0,12);  //Obtenim ID
                        $status=substr($container,$status_pos,2); //Status
                        if($status=="Up") $status="Running";
                        else $status="Stopped";
                }
        }
        return(array('id'=>$id, 'status'=>$status));
}
```

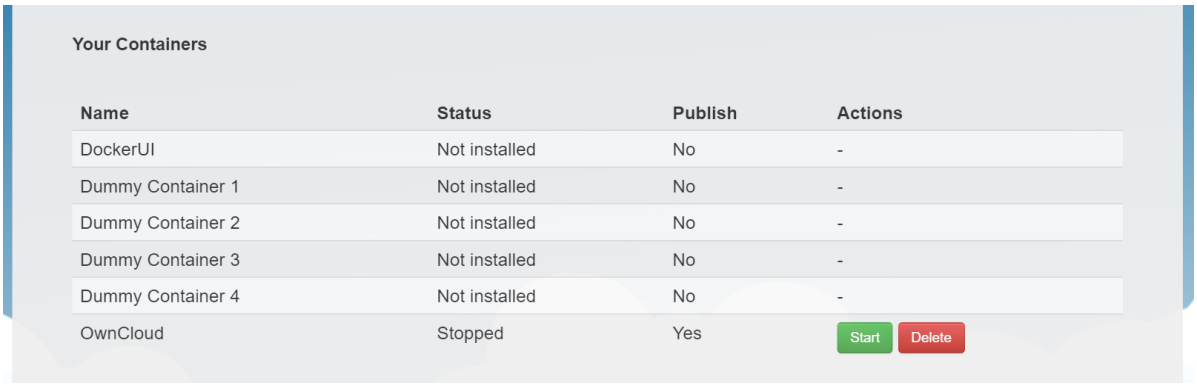
Es crea el següent codi per mostrar la taula resultant:

```
function docker_Admin($page_tmp){
   global $dev,$urlpath;
   $containerspath="/var/local/cDistro/plug/resources/docker/containers/";

   $page_tmp .= hl("", 4);
   $page_tmp .= hl("Your Containers", 4);

   //llistem els fitxers existents
   $cmd = "ls ".$containerspath;
   $llista = execute_program($cmd);
```

```
    //Iniciem la taula
    $page_tmp    .=    addTableHeader(array("Name","Status","Publish","Actions"),
        array('class'=>'table table-striped'));

    //Per a cada fitxer trobat
    foreach($llista['output'] as $container){
       //Obtenim info dels fitxers
       $cInfo = get_container_info($container);
       //Obtenim info de Docker
       $cDocker = get_docker_info($container);
       //Definim botons
       if($cDocker['status']=="Running"){
           $boton=addButton(array('label'=>"Stop",'class'=>'btn            btn-
              danger','href'=>"$urlpath/containerStop/".$cDocker['id']."_".$con
              tainer));
           $boton2=addButton(array('label'=>"Delete", 'class'=>'btn btn-default
              disabled'));
       }else{
           $boton=addButton(array('label'=>"Start",'class'=>'btn           btn-
              success','href'=>"$urlpath/containerStart/".$cDocker['id']."_".$c
              ontainer));
           $boton2=addButton(array('label'=>"Delete",'class'=>'btn         btn-
              danger','href'=>"$urlpath/containerDelete/".$cDocker['id']));
       }

       //Si existeix, mostrem botons
       if($cDocker['id']!=0){
               $page_tmp                                                    .=
              addTableRow(array($cInfo['name'],$cDocker[status],$cInfo['pub'],$
              boton.$boton2));
       }else{
               $page_tmp        .=        addTableRow(array($cInfo['name'],"Not
              installed",$cInfo['pub'],"-"));
       }
    }
    //Tanquem la taula i retornem resultat
    $page_tmp .= addTableFooter();
    return ( $page_tmp );
}
```

A continuació es mostra el resultat obtingut, S'han creat una sèrie de fitxers addicionals amb informació fictícia per donar cos a l'exemple.

Com es pot veure, a la taula es mostren tots els contenidors definits als fitxers, a més, pels contenidors existents a Docker, s'enriqueix la informació amb l'estat actual del contenidor i s'afegeixen els botons definits al primer Sprint:

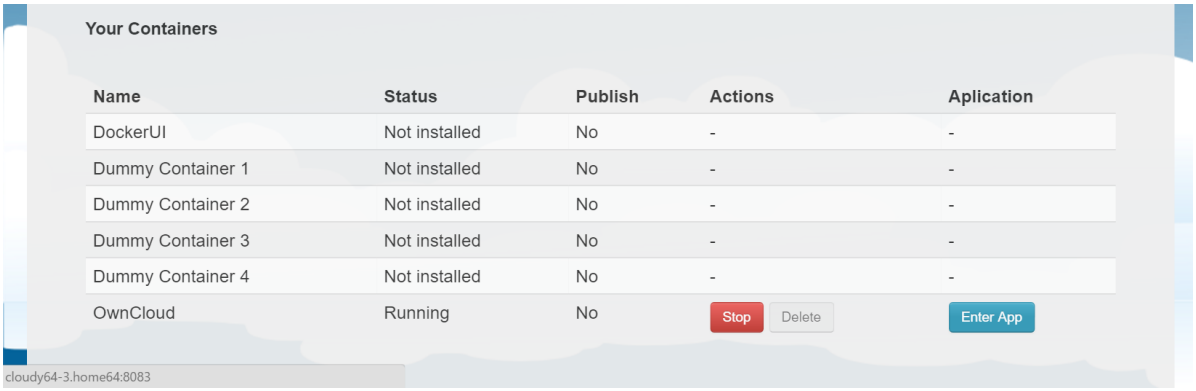


Tsk08: Afegir botó amb link a la App

Per facilitar el seguiment del codi, es decideix crear una funció per a definir el text, color i acció de cada botó, al següent codi es defineixen els botons Start/Stop i Delete (ja existents) i Accedir App:

```
function botonStartStop($cInfo,$cDocker){
    global $dev,$urlpath;
    if($cDocker['status']=="Running"){
         return(addButton(array('label'=>"Stop",'class'=>'btn                btn-danger',
             'href'=>"$urlpath/containerStop/".$cDocker['id']."_".$cInfo['name'])));
    }else{
         return($botonOn=addButton(array('label'=>"Start",'class'=>'btn    btn-success',
             'href'=>"$urlpath/containerStart/".$cDocker['id']."_".$cInfo['name'])));
        }
}

function botonDelete($cInfo,$cDocker){
    global $dev,$urlpath;
    if($cDocker['status']=="Running"){
         return(addButton(array('label'=>"Del",'class'=>'btn btn-default disabled')));
    }else{
         return(addButton(array('label'=>"Delete",'class'=>'btn              btn-danger',
              'href'=>"$urlpath/containerDelete/".$cDocker['id'])));
        }
}

function botonApp($cInfo,$cDocker){
    //obtenim el hostname
    $cmd="hostname -f";
    $hostname=execute_program($cmd);

    //Si el contenidor està en execució, mostrem el botó
    if($cDocker['status']=="Running"){
         return(addButton(array('label'=>"Enter      App",'class'=>'btn      btn-info',
              'href'=>"http://".$hostname['output'][0].":".$cInfo['port'])));
    }else{
         return(addButton(array('label'=>"Enter      App",'class'=>'btn      btn-default
               disabled')));
        }
}
```

Amb el següent resultat:



On el botó “Enter App”, que només es mostra si el contenidor està arrancat, apunta al nom del node + Port identificat per crear la URL d’accés

Tsk09: Afegir botó per publicar contenidor

Es defineix el botó amb la següent funció:

```
function botonPublish($cInfo,$cDocker){
    global $dev,$urlpath;
    if($cDocker['status']=="Running" && $cInfo['pub']=="No"){
          return(addButton(array('label'=>"Publish",'class'=>'btn     btn-info',
                'href'=>"$urlpath/containerPublish/Pub".$container)));
    }else if($cDocker['status']=="Running" && $cInfo['pub']=="Yes"){
          return(addButton(array('label'=>"Unpublish",'class'=>'btn   btn-info',
                'href'=>"$urlpath/containerPublish/Unpub".$container)));
    }else{
          return(addButton(array('label'=>"Publish",'class'=>'btn    btn-default
                disabled')));
    }
 }
```

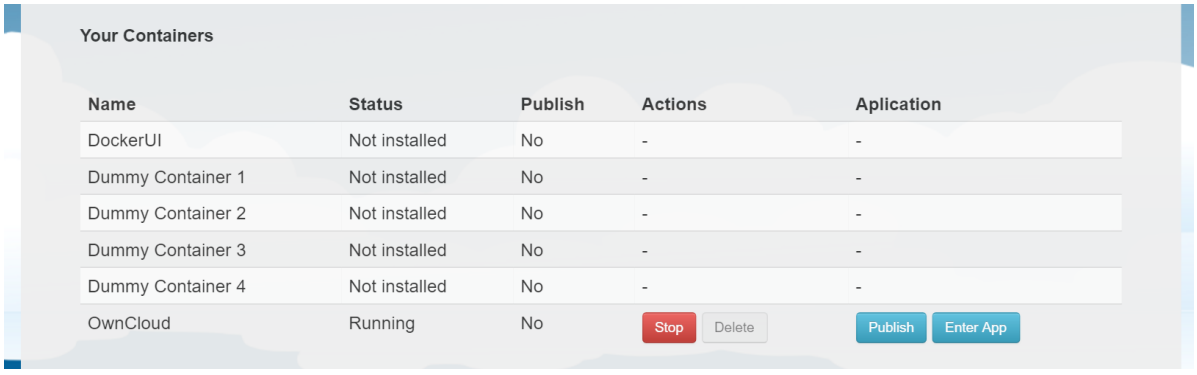
Aquest botó només es mostra habilitat quan el contenidor s’està executant.

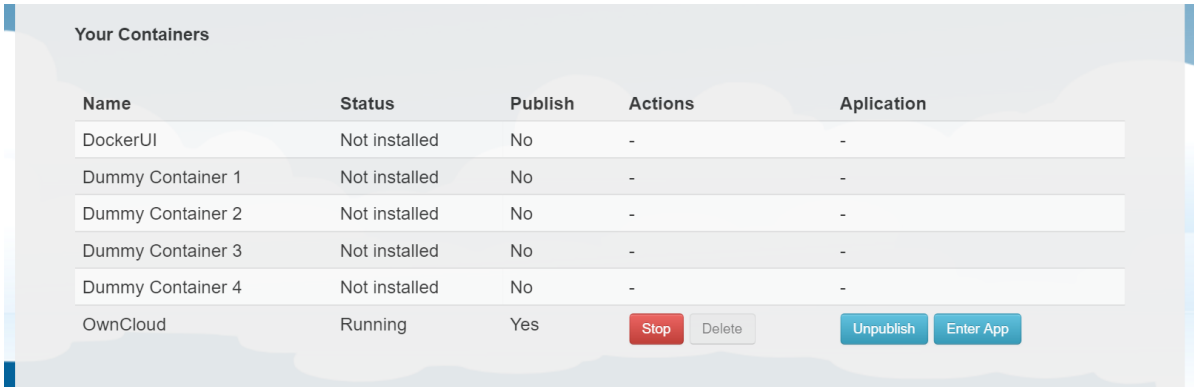
Y la acció associada al botó:

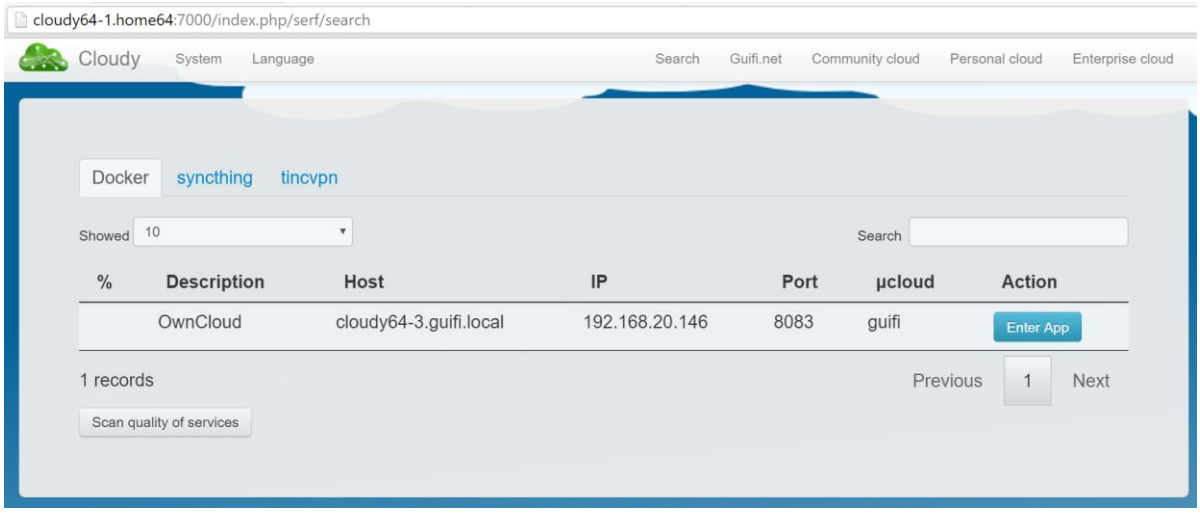
```
function containerPublish() {

        global $urlpath,$Parameters;
        $respath="/var/local/cDistro/plug/resources/docker/";

        //Obtenim el contenidor a Publicar (Paràmetre de la funció)
        $con = $Parameters[0];
        //Obtenim informació associada al contenidor
        $conInfo=get_container_info($con);
```

```
        //Publiquem el servei
        execute_program_detached("/usr/sbin/avahi-ps                    publish
             ".$conInfo['name']." Docker ".$conInfo['port']);
        //Modifiquem el paràmetre de configuració
        execute_program_detached($respath."publish ".$conInfo['img']);

        setFlash("Container Published. ","success");
        return(array('type'=> 'redirect', 'url' => $urlpath));
}

function containerUnpublish() {
        global $urlpath,$Parameters;
        $respath="/var/local/cDistro/plug/resources/docker/";

        //Obtenim el contenidor a Publicar (Paràmetre de la funció)
        $con = $Parameters[0];
        //Obtenim informació associada al contenidor
        $conInfo=get_container_info($con);

        //Publiquem el servei
        execute_program_detached("/usr/sbin/avahi-ps      unpublish      Docker
             ".$conInfo['port']);
        //Modifiquem el paràmetre de configuració
        execute_program_detached($respath."unpublish ".$conInfo['img']);

        setFlash("Container Unpublished. ","success");
        return(array('type'=> 'redirect', 'url' => $urlpath));
}
```

El botó publica o despublica el contenidor i modifica el paràmetre del fitxer de configuració.







Com es pot observar a la imatge, al mostrar els serveis actius als nodes veins, ens mostra un botó amb l'opció d'accedir a l'aplicació. Això s'ha aconseguit afegint els següent fitxers a tots els nodes del clúster:

```
funcroot@cloudy64-1:/# cat /var/local/cDistro/plug/avahi/docker.avahi.php
<?php
// plug/avahi/docker.avahi.php

addAvahi('Docker','fDocker');

function fDocker($dates){
        global $staticFile;

      //Crea un botó apuntant cap a IP:Port del node que publica el servei
        return(addButton(array('label'=>"Enter  App",'class'=>'btn  btn-info',
        'href'=>"http://".$dates['ip'].":".$dates['port'])));
}
```

Aquesta funció serà valida per a tots els serveis que es publiquin a la categoria "Docker"

Per últim, s'ha modificat la funció que arrenca el contenidor a que només publiqui el servei si aquest està configurat per publicar-se. `containerStart()`

```
[...]
//Publiquem el servei si aplica
$conInfo=get_container_info($cIm);
if($conInfo['pub']=="Yes"){
     execute_program_detached("/usr/sbin/avahi-ps publish ".$conInfo['name']."
          Docker ".$conInfo['port']);
     setFlash("Container Published ".$cIm."success");
}
[...]
```

Tsk21: Analitzar Docker Containers + GUI vs Sandstorm

Com s'ha vist fins ara, s'ha simplificat la gestió dels contenidors amb el GUI disponible, però per arribar a l'agilitat que proporciona Sandstorm en el desplegament de serveis, s'han de proposar diferents aplicacions a instal·lar com a contenidor.

Amb aquest objectiu, es revisen les aplicacions publicades al repositori públic de Docker (hub.docker.com) per proposar la següent llista:

Funcionalitat	Execució
Panell Kanban	`docker run -d -p 8084:80 mkodockx/docker-kanban`
BBDD MongoDB	`docker run --name db -d mongo --smallfiles`
Rocket.chat (Requereix mongodb)	`docker run --name rocketchat -p 3000:3000 --env ROOT_URL=http://localhost --link db -d rocket.chat`
Wiki	docker run -it -p 8085:443 -p 8086:80 --name my_wiki olavgg/moinmoin-wiki
BBDD MySQL	docker run --name mariadb -e MYSQL_ROOT_PASSWORD=1234 -d mariadb:tag
Wordpress (Requereix mysql)	docker run --name Wordpress --link mariadb:mysql -p 8087:80 -d wordpress

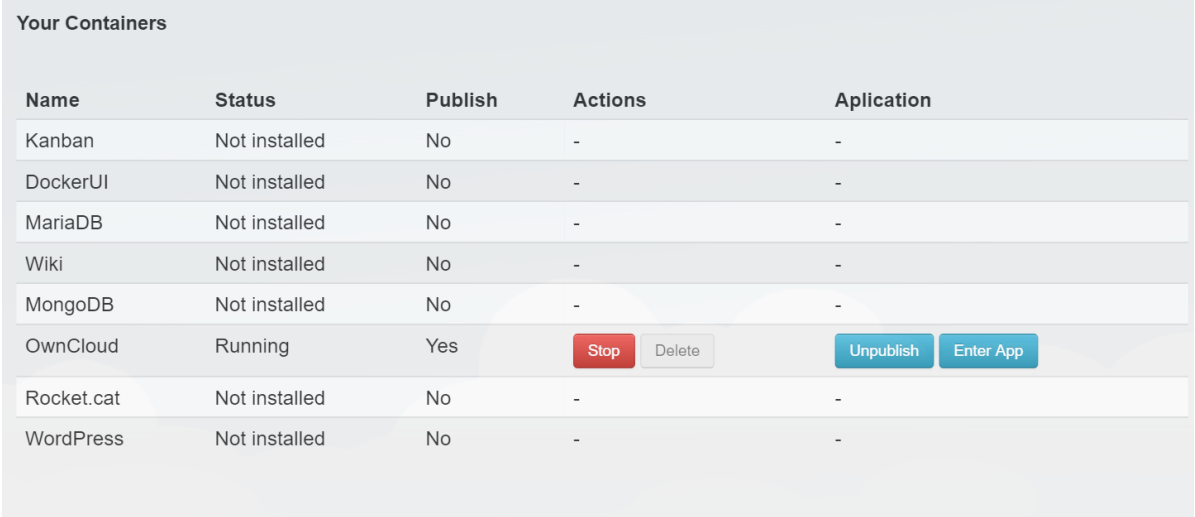
No s'ha validat el correcte funcionament intern de aquestes aplicacions, per ara es pretén tenir un subconjunt de contenidors per poder fer proves amb la instal·lació.

Tsk11: Informar taula de contenidors i afegir instal·lació

Es creen els fitxers de configuració per donar suport a les aplicacions de la taula anterior.

```
root@cloudy64-3:/var/local/cDistro/plug/resources/docker/containers# ls -l
-rw-r--r-- 1 root staff 107 May 13 09:58 docker-kanban
-rw-r--r-- 1 root staff 194 May  8 06:19 dockerui:lastest
-rw-r--r-- 1 root staff 124 May 13 10:04 mariadb:tag
-rw-r--r-- 1 root staff 131 May 13 10:03 moinmoin-wiki
-rw-r--r-- 1 root staff  92 May 13 10:01 mongo
-rw-r--r-- 1 root staff 107 May 13 09:57 owncloud:8.1
-rw-r--r-- 1 root staff 158 May 13 10:00 rocket.chat
-rw-r--r-- 1 root staff 131 May 13 10:05 wordpress
```

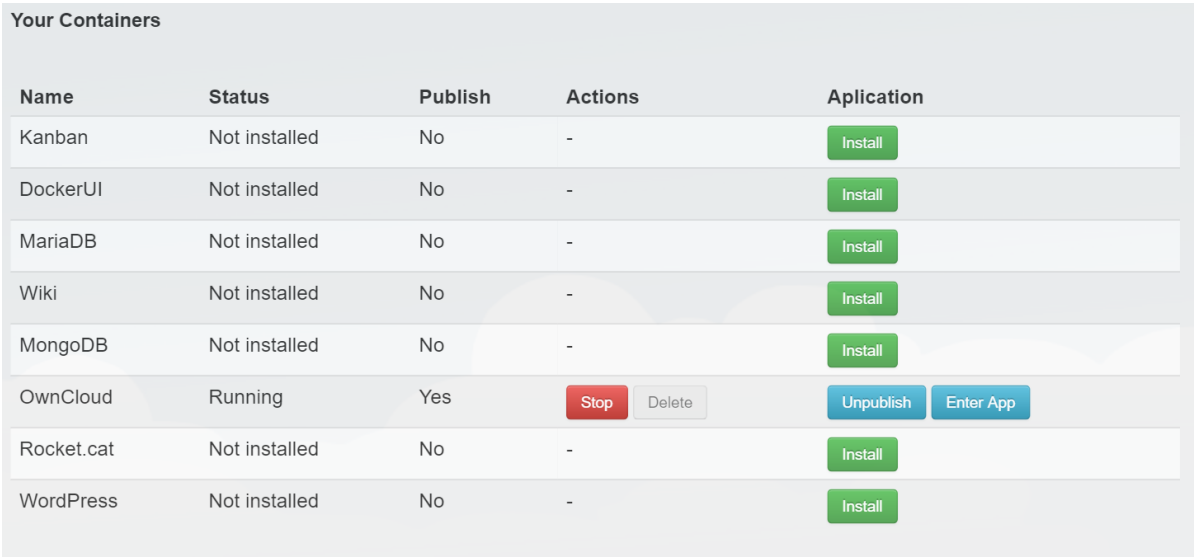
Es poden veure al panell d'administració de Docker:



Per últim, s'afegeix el botó d'instal·lació si no troba el contenidor instal·lat amb el següent codi associat:

```
function botonInst($cInfo,$cDocker){
    return(addButton(array('label'=>"Install",'class'=>'btn      btn-success',
        'href'=>"$urlpath/containerInstall/".$cInfo['img'])));
}
```

Amb el següent resultat:



Queda pendent que el botó d'instal·lació emmagatzemi al fitxer de configuració el ID del contenidor, paràmetre imprescindible pel correcte funcionament del sistema.

3. Tasques pendents

Aquesta és la llista de tasques pendents, es marquen en verd les seleccionades per implementar al següent Sprint. Es marquen en blau noves tasques afegides:

ID Tasca	Descripció Tasca	Estat
Tsk11b*	Completar la funció d'instal·lació	En curs
Tsk12	Descobrir com Cloudy detecta canvis en els serveis publicats al node local	Pendent
Tsk13	Si s'arranca un contenidor que hem decidit que volem publicar, publicar-ho directament	Completada a la Tsk10
Tsk14	Si es para o cau un contenidor que hem decidit que volem publicar, despublicar-ho directament	Pendent
Tsk15	Crear un interface per poder afegir contenidors a la llista de contenidors recomanats	Pendent
Tsk16	Crear un nou menú a la web d'administració de Cloudy que permeti instal·lar Sandstorm al node	Descartat
Tsk17	Afegir al nou menú el link d'accés a l'eina de gestió de Sandstorm	Descartat
Tsk18	Investigar com podem detectar les apps que Sandstorm té creades	Descartat
Tsk19	Crear un interfase que permeti publicar els serveis de Sandstorm	Descartat

	de forma individual creat una taula que emmagatzemi els serveis a publicar	
Tsk20	Detectar canvis en els serveis de Sandstorm publicats per gestionar si es continuen publicant o no	Descartat

4. Resum

4.1.Sprint tancat

Aquest Sprint ha suposat un gran esforç en comprendre alguns mecanismes interns de Cloudy, com per exemple el sistema de publicació de serveis i com associar una acció en la finestra de serveis publicats.

En tot cas, el resultat és satisfactori i actualment la plana de Docker disposa de funcionalitats avançades a un interface simple, que era l'objectiu.

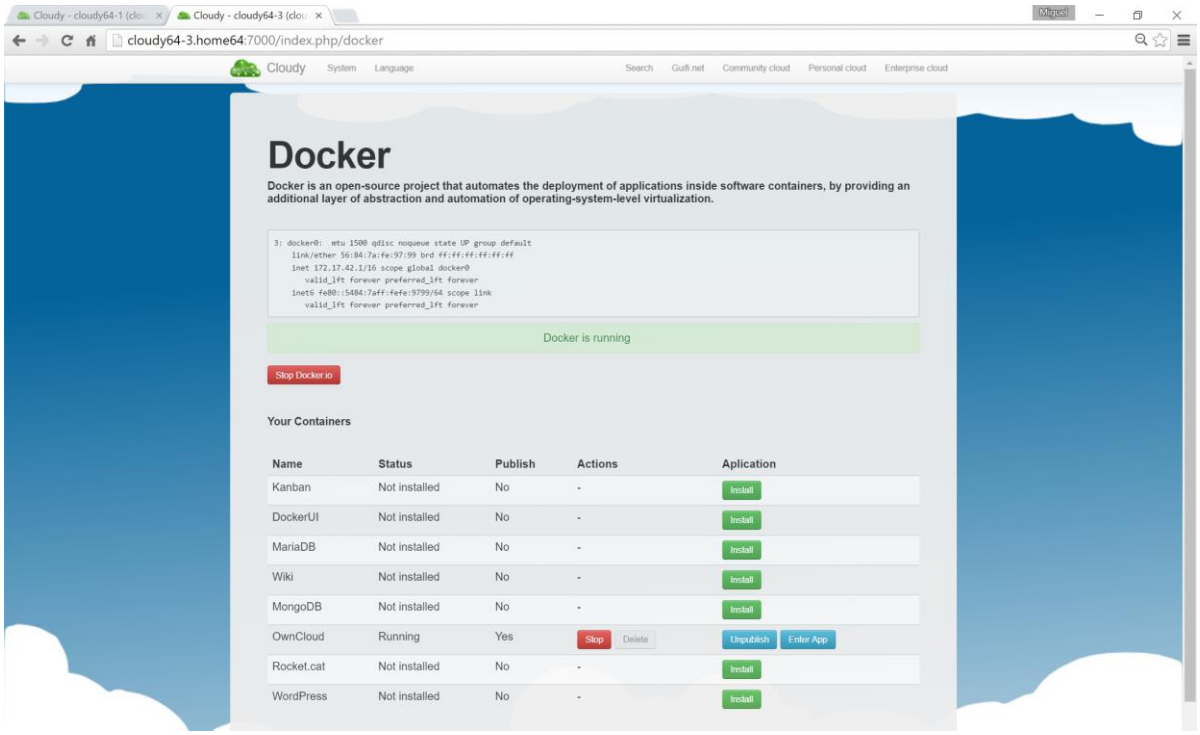
Es descarten les iniciatives de integració amb Sandstorm ja que Docker amb el nou GUI implementat dona molt més valor a l'usuari.

4.2.Sprint obert

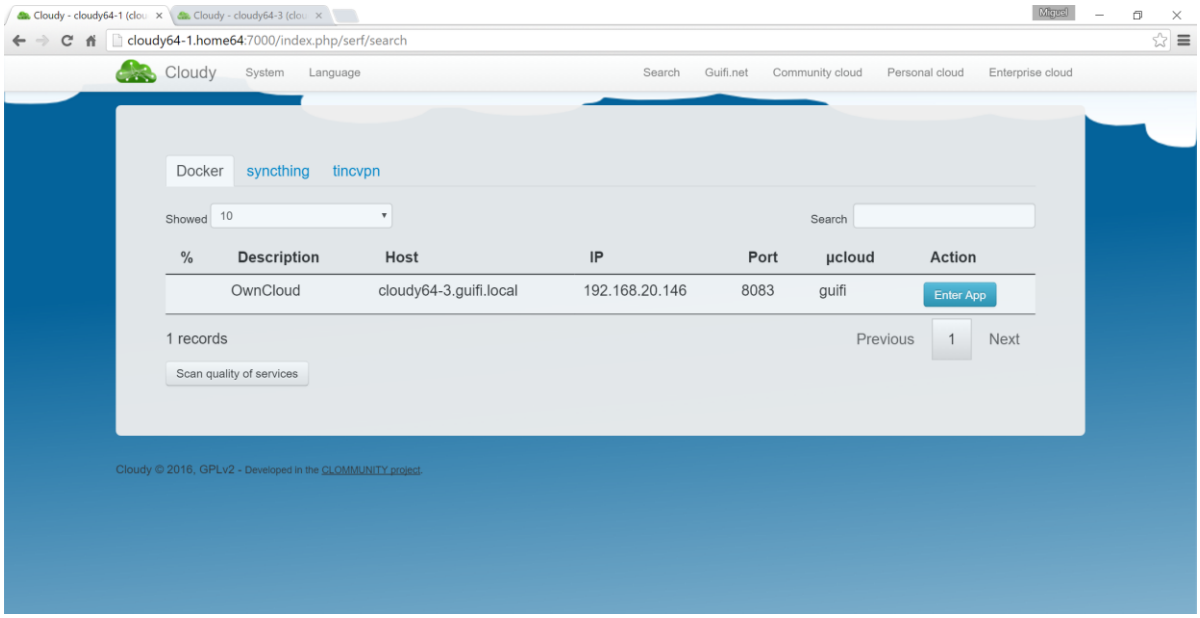
Les tasques seleccionades al Sprint que s'inicia estan orientades a millorar el comportament de la publicació i despublicació de serveis reaccionant envers a esdeveniments no planificats. A més, s'evolucionarà la taula de contenidors per fer-la dinàmica.

4.3. Vista actual

Estat actual de la plana d'administració de Docker:



I de la vista de serveis publicats des de un altre node del clúster:



Annex VI: Informe Sprint 3

1. Introducció

Aquest informe té com a objectiu el seguiment de les tasques de projecte enregistrant la seva evolució. Amb aquesta finalitat, es documentaran les tasques finalitzades i seleccionaran de noves per a executar al següent Sprint.

La periodicitat estimada dels Sprint és d'una setmana.

2. Tasques planificades

Aquest és el llistat de tasques planificades per al tercer Sprint:

ID Tasca	Descripció Tasca	Estat
Tsk11b*	Completar la funció d'instal·lació	Completada
Tsk12	Descobrir com Cloudy detecta canvis en els serveis publicats al node local	Completada
Tsk14	Si es para o cau un contenidor que hem decidit que volem publicar, despublicar-ho directament	Completada
Tsk15	Crear un interface per poder afegir contenidors a la llista de contenidors recomanats	Completada
Tsk23	(Nova tasca) Crear un script d'instal·lació del nou GUI	Completada

Per començar aquest Sprint, es completarà la funció d'instal·lació de contenidors, seguit d'un estudi i implementació de monitoritzadors d'estat per als serveis publicats al contenidor i acabant amb un interface web per afegir i modificar els contenidors predefinits.

Addicionalment s'afegeix una tasca per automatitzar l'aplicació de tots els desenvolupaments que s'han creat durant el TFM sobre una imatge neta de Cloudy. (Funcionalitat similar al script cloudinitzar)

A continuació es detalla el resultat de cada tasca:

Tsk11b: Completar la funció d'Instal·lació

Aquesta tasca té dos reptes principals:

- Fer saber a l'usuari que el contenidor s'està instal·lant ja que aquest procés pot trigar minuts al descarregar la imatge per primera vegada.
- Obtenir el ID del contenidor al finalitzar la instal·lació per poder gestionar-lo posteriorment.

Es decideix que la millor opció és canviar el test del botó d'instal·lació per "Installing...", d'aquesta manera l'usuari pot saber que ha d'esperar.

D'altra banda es planteja modificar les funcions de inici i parada d'un contenidor per a que no sigui necessari emmagatzemar el seu ID.

Amb aquest objectiu, el procés d'instal·lació crida a aquesta funció:

```
function containerInstall() {
        global $urlpath,$Parameters;
        $respath="/var/local/cDistro/plug/resources/docker/";

        //Obtenim el contenidor a Instal·lar (Paràmetre de la funció)
        $con = $Parameters[0];
        //Obtenim informació associada al contenidor
        $conInfo=get_container_info($con);

        //executem la instal·lació
        execute_program_detached($respath."install ".$conInfo['run']);

        //Alertem que el contenidor està en fase d'instal·lació
        execute_program_detached($respath."installing ".$conInfo['img']);

        setFlash("Installing Container... ".$conInfo['run'],"success");
        return(array('type'=> 'redirect', 'url' => $urlpath));
}
```

I es creen aquests dos scripts com auxiliars a la instal·lació:

```
root@cloudy64-3:/var/local/cDistro/plug/resources/docker# cat install
#!/bin/bash
#Espera com a paràmetre la comanda a executar
path="/var/local/cDistro/plug/resources/docker/install.log"

echo ""  >> $path
echo PROCÉS INSTAL·LACIÓ NOU CONTENIDOR >> $path
echo Comanda: "          " $* >> $path
echo Data inici: "       " `date` >> $path
echo ""  >> $path

$* >> $path

echo ""  >> $path
echo Data fi: "          " `date` >> $path


root@cloudy64-3:/var/local/cDistro/plug/resources/docker# cat installing
#!/bin/bash
echo "Id;Installing" >> /var/local/cDistro/plug/resources/docker/containers/$1
```

El primer és qui executa realment la instal·lació i escriu el procés a un fitxer de log.

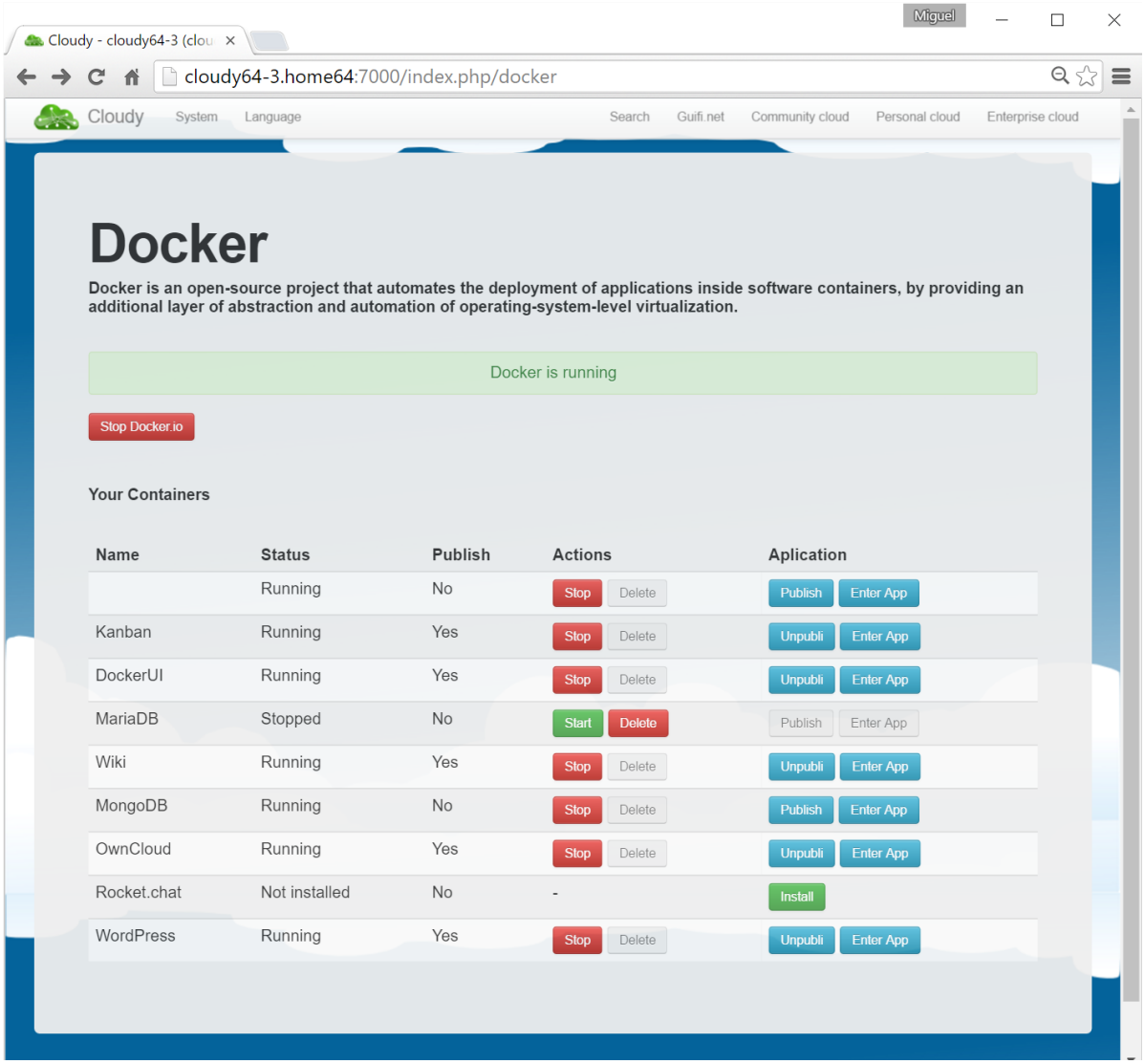
Pel que fa a la dependència d'obtenir el ID durant el procés d'instal·lació:
Es modifiquen les funcions de iniciar i parar contenidors, on anteriorment obtenien per paràmetre el ID del contenidor i el nom de la imatge.

Es substitueix això i ara només es passa per paràmetre el nom de la imatge, per obtenir posteriorment el ID del contenidor:

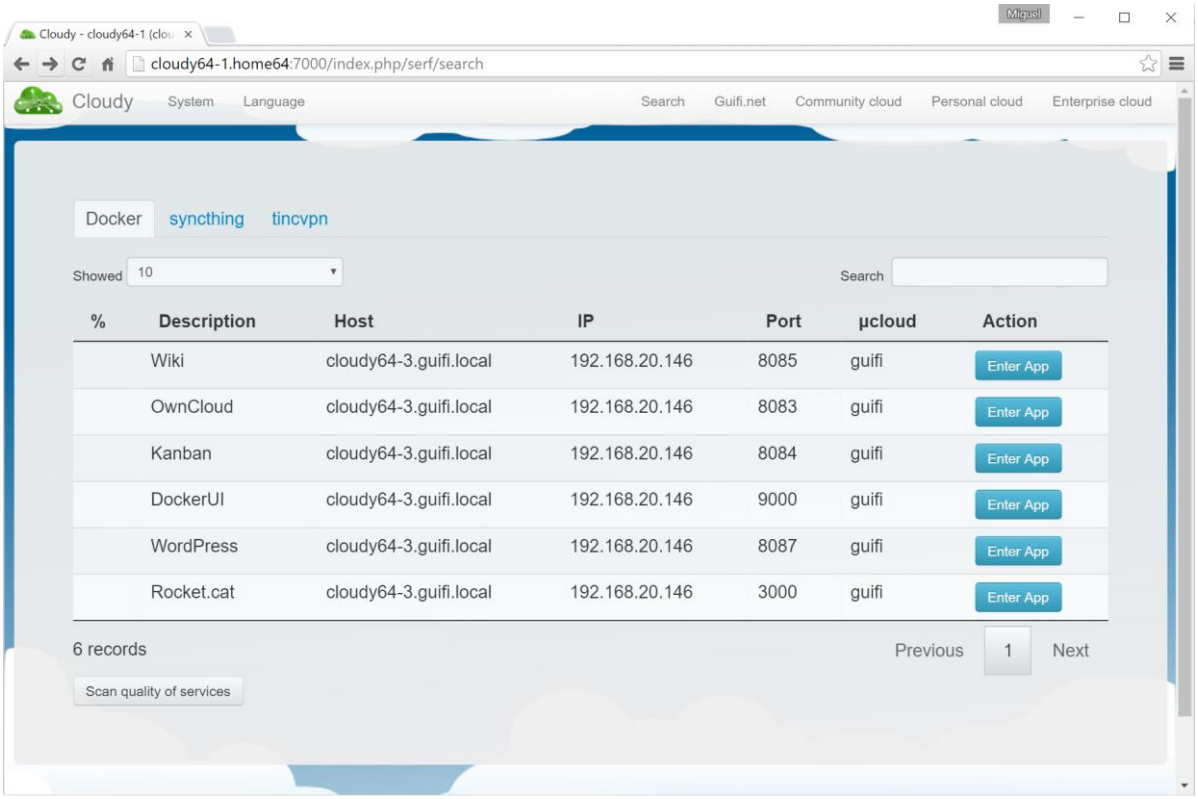
```
[...]

        //Recollim com paràmetre el nom de la imatge
        $cIm = $Parameters[0];

        //Obtenir informació de Docker
        $con = get_docker_info($cIm);
        $cId = $con['id'];

[...]
```

El resultat és que tots els contenidors es poden Instal·lar i desinstal·lar, arrencar i parar, publicar i despublicar i eliminar des del nou interface gràfic sense necessitat de tenir coneixements tècnics.

Aquest és el resultat final de la vista del GUI:



I Aquesta la vista dels serveis publicats des d'un altre node:

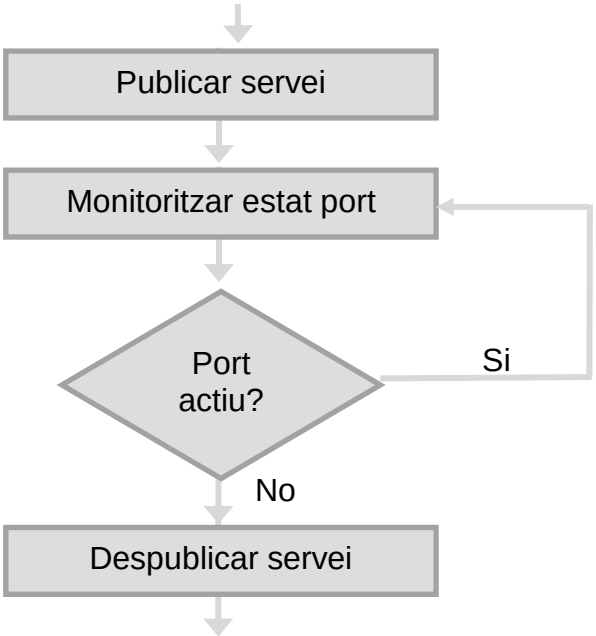


Tsk12: Analitzar com Cloudy monitoritza serveis

No hi ha disponible gaire documentació de com Cloudy gestiona la monitorització de serveis per despublicar-los en cas de que el servei esdevingui indisponible tot i que el node estigui actiu.

Per aquest motiu i aprofitant que els serveis que publicarem sempre tindran un port obert, es decideix investigar la via de com monitoritzar els ports actius en debian.

Podem aprofitar aquesta comanda per definir el següent flux de publicació de serveis:



Es troba la comanda “nc”, que és nativa en aquesta distribució i que permet quedar-se en standby fent un keepalive mentre el port està obert.

Tsk14: Implantar monitoritzador de serveis publicats

Es crea el següent script a la carpeta de recursos:

```
root@cloudy64-3:/var/local/cDistro/plug/resources/docker# cat monitor
#!/bin/bash
#Es rep el port com a paràmetre: $1
#Esperem mentre el port està obert
nc -k localhost $1
#Si deixa d'estar obert, despubliquem el servei
/usr/sbin/avahi-ps unpublish Docker $1
```

Aquest script es crida cada vegada que un servei es publica. Mentre el port està obert, es queda monitoritzant l’estat, si el port es tanca, despublica el servei automàticament.

Tsk15: Crear interface per modificar contenidors gestionats

Per poder editar la configuració dels contenidors gestionats es crea una nova plana PHP “*docker-form.php*”, que mostrarà un formulari amb totes les característiques emmagatzemades d’un contenidor i permetrà modificar-les.

Aquesta mateixa plana, en cas de no rebre com a paràmetre cap contenidor a modificar, servirà per crear un de nou.

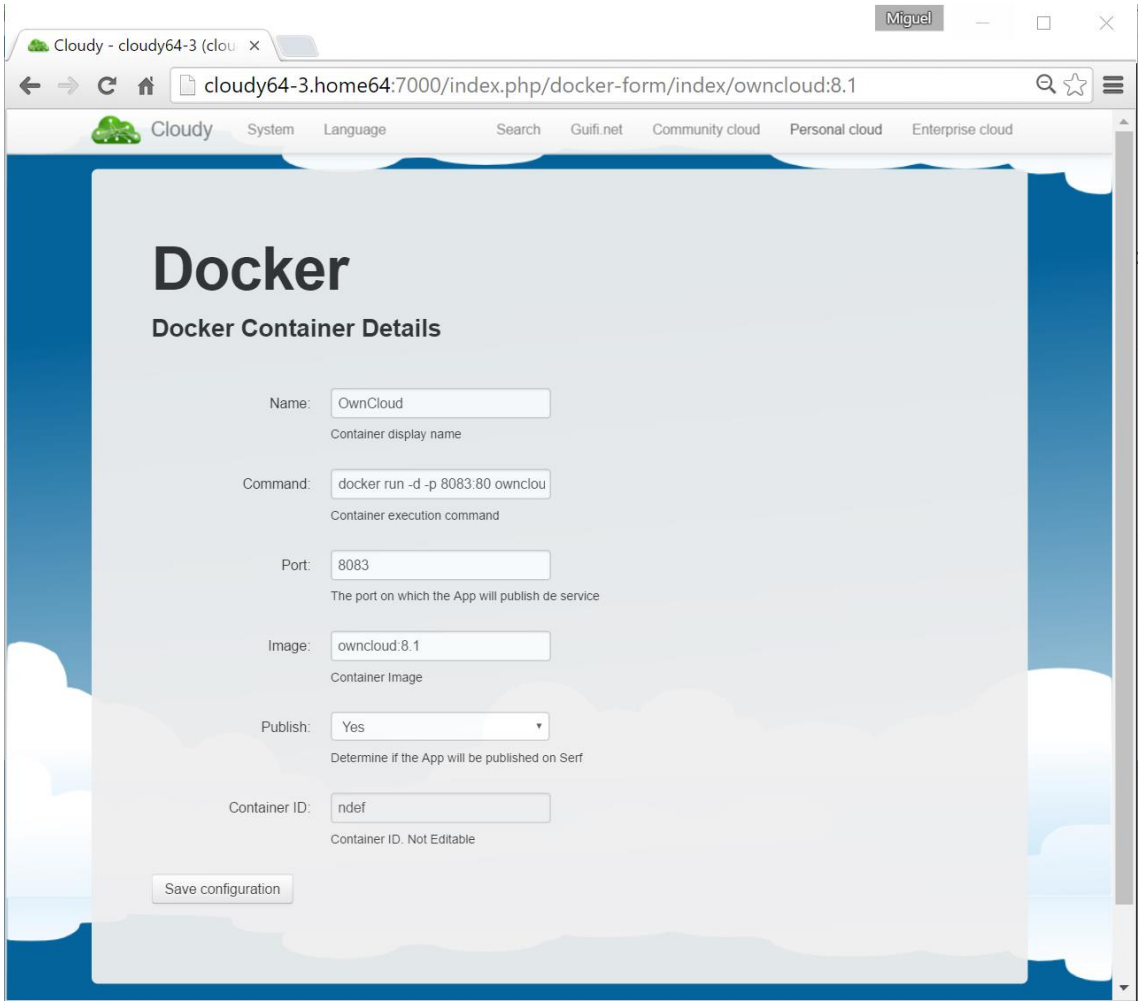
El codi definit per la nova plana és el següent:

```
<?php
$urlpath="$staticFile/docker";
$docker_pkg = "docker.io";
$dev = "docker0";

function index() {
     global $title, $urlpath, $docker_pkg, $staticFile, $Parameters;

     //Recollim els paràmetres
     $image = $Parameters[0];
     if($image==""){
    $con=array();
     }else{
    $con = get_container_info($image);
     }

     //Capçalera
     $page = hlc(t("docker_title"));
     $page .= hl("Docker Container Details", 3);
     //Formulari
     $page .= createForm(array('class'=>'form-horizontal'));
```

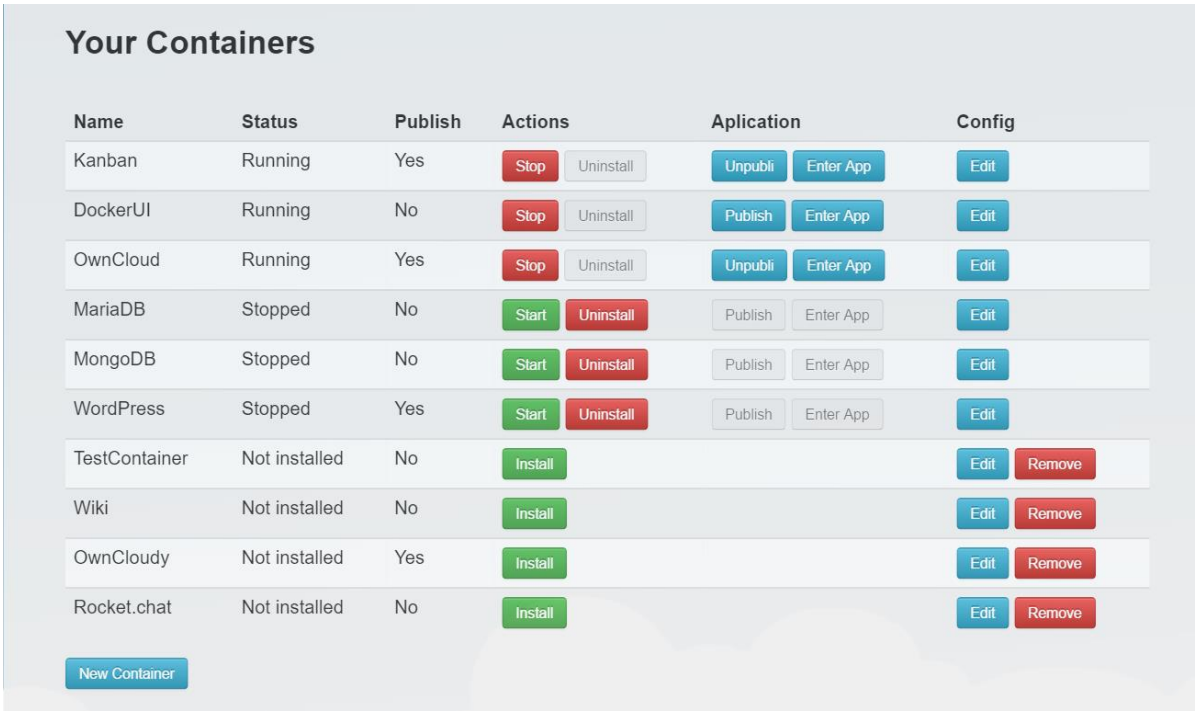
```
    $page                                                                                .=
       addInput('Name',"Name",$con['name'],array('type'=>'text','required'=>''),"","Co
       ntainer display name");
    $page                                                                                .=
       addInput('Run',"Command",$con['run'],array('type'=>'text','size'=>200,'required
       '=>''),"","Container execution command");
    $page   .=   addInput('Port',"Port",$con['port'],array('type'=>'number',   'min'   =>
       '1024', 'max' => '65535', 'required'=>''),"","The port on which the App will
       publish de service");
    $page                                                                                .=
       addInput('Img',"Image",$con['img'],array('type'=>'text','required'=>''),"","Con
       tainer Image");
    if($con['pub']=="Yes"){
            $page                                                                        .=
       addCheckbox2('Pub',"Publish",array("Yes","No"),array('type'=>'text','required'=
       >''),"","Determine if the App will be published on Serf");
    }else{
            $page                                                                        .=
       addCheckbox2('Pub',"Publish",array("No","Yes"),array('type'=>'text','required'=
       >''),"","Determine if the App will be published on Serf");
    }
    $page                              .=                              addInput('Id',"Container
       ID","ndef",array('type'=>'text','size'=>'200'),"readonly","Container   ID.   Not
       Editable");
    $page    .=    addSubmit(array('label'=>t("serf_button_save"),'class'=>'btn    btn-
       default','divOptions'=>array('class'=>'btn-group')));

    return array('type' => 'render','page' => $page);
}


function index_post(){
        global $staticFile, $staticPath, $urlpath;
        global $avahipsetc_config;
        $datesToSave = array();
        foreach ($_POST as $key => $value) {
                $datesToSave[$key] = $value;
        }

       $file="/var/local/cDistro/plug/resourc/docker/containers/".$datesToSave['Img'];
        write_conffile($file,$datesToSave,"","",'"');

        setFlash("Config -> OK","info");
        return(array('type'=> 'redirect', 'url' => $staticFile."/docker"));
}
```

NOTA: En aquest fitxer s'ha reescrit la funció addCheckbox de la llibreria per defecte *(lib/form.php)*. El motiu és que aquesta retornava el identificador de l'opció seleccionada i no el text. A la nova funció es retorna el text de l'opció seleccionada.

Aquesta és l'aparença del nou formulari quan volem editar un contenidor existent: (S'ha fet servir l'estil habitual per crear formularis al web)



A la vista general, es pot observar com s'ha afegir un botó a cada filera amb l'objectiu de poder editar la configuració. A més, s'ha afegit un nou botó sota la taula per crear nous contenidors gestionats:



Tsk23: Script d'aplicació dels canvis desenvolupats

Revisem els fitxers modificats de la plana d'administració de Cloudy:

```
root@cloudy64-3:/var/local/cDistro# find . -mtime -60 -ls
```

Trobem els següents:

Fitxers PHP:
/var/local/cDistro/plug/avahi/docker.avahi.php
/var/local/cDistro/plug/controllers/docker.php
/var/local/cDistro/plug/controllers/docker-form.php

Directoris:
/var/local/cDistro/plug/resources/Docker
/var/local/cDistro/plug/resources/docker/containers

Scripts de suport:
/var/local/cDistro/plug/resources/docker/installing
/var/local/cDistro/plug/resources/docker/unpublish
/var/local/cDistro/plug/resources/docker/monitor
/var/local/cDistro/plug/resources/docker/publish
/var/local/cDistro/plug/resources/docker/install
/var/local/cDistro/plug/resources/docker/delete

Fitxers de configuració de contenidors per defecte:
/var/local/cDistro/plug/resources/docker/containers/mongo

/var/local/cDistro/plug/resources/docker/containers/rocket.chat
/var/local/cDistro/plug/resources/docker/containers/mariadb:latest
/var/local/cDistro/plug/resources/docker/containers/moinmoin-wiki
/var/local/cDistro/plug/resources/docker/containers/docker-kanban
/var/local/cDistro/plug/resources/docker/containers/dockerui:latest
/var/local/cDistro/plug/resources/docker/containers/wordpress
/var/local/cDistro/plug/resources/docker/containers/owncloud:8.1

Menus:
/var/local/cDistro/plug/menus/zdocker-form.menu.php

Creem una carpeta amb tots els fitxers modificats agrupats en carpetes segons tipologia:

```
drwxr-xr-x root/root      0 2016-05-21 19:55 ./TFM_files/
drwxr-xr-x root/root      0 2016-05-21 19:57 ./TFM_files/scripts/
-rwxr-xr-x root/root     92 2016-05-21 19:56 ./TFM_files/scripts/installing
-rwxr-xr-x root/root     85 2016-05-21 19:56 ./TFM_files/scripts/unpublish
-rwxr-xr-x root/root    191 2016-05-21 19:56 ./TFM_files/scripts/monitor
-rwxr-xr-x root/root     86 2016-05-21 19:56 ./TFM_files/scripts/publish
-rwxr-xr-x root/root    365 2016-05-21 19:56 ./TFM_files/scripts/install
-rwxr-xr-x root/root    157 2016-05-21 19:56 ./TFM_files/scripts/delete
drwxr-xr-x root/root      0 2016-05-21 19:57 ./TFM_files/config/
-rw-r--r-- root/root    136 2016-05-21 19:57 ./TFM_files/config/mongo
-rw-r--r-- root/root    100 2016-05-21 19:57 ./TFM_files/config/owncloud:8.2
-rw-r--r-- root/root    244 2016-05-21 19:57 ./TFM_files/config/rocket.chat
-rw-r--r-- root/root    174 2016-05-21 19:57 ./TFM_files/config/mariadb:latest
-rw-r--r-- root/root    147 2016-05-21 19:57 ./TFM_files/config/moinmoin-wiki
-rw-r--r-- root/root    144 2016-05-21 19:57 ./TFM_files/config/docker-kanban
-rw-r--r-- root/root    229 2016-05-21 19:57 ./TFM_files/config/dockerui:latest
-rw-r--r-- root/root    153 2016-05-21 19:57 ./TFM_files/config/wordpress
-rw-r--r-- root/root     99 2016-05-21 19:57 ./TFM_files/config/owncloud:8.1
-rw-r--r-- root/root     81 2016-05-21 19:57 ./TFM_files/config/DummyImage:3.0
drwxr-xr-x root/root      0 2016-05-21 19:55 ./TFM_files/php/
-rw-r--r-- root/root    314 2016-05-21 19:55 ./TFM_files/php/docker.avahi.php
-rwxr-xr-x root/root  15328 2016-05-21 19:55 ./TFM_files/php/docker.php
-rw-r--r-- root/root   5724 2016-05-21 19:55 ./TFM_files/php/docker-form.php
drwxr-xr-x root/root      0 2016-05-21 19:57 ./TFM_files/menus/
-rw-r--r-- root/root     95 2016-05-21 19:57 ./TFM_files/menus/zdocker-form.menu.php
```

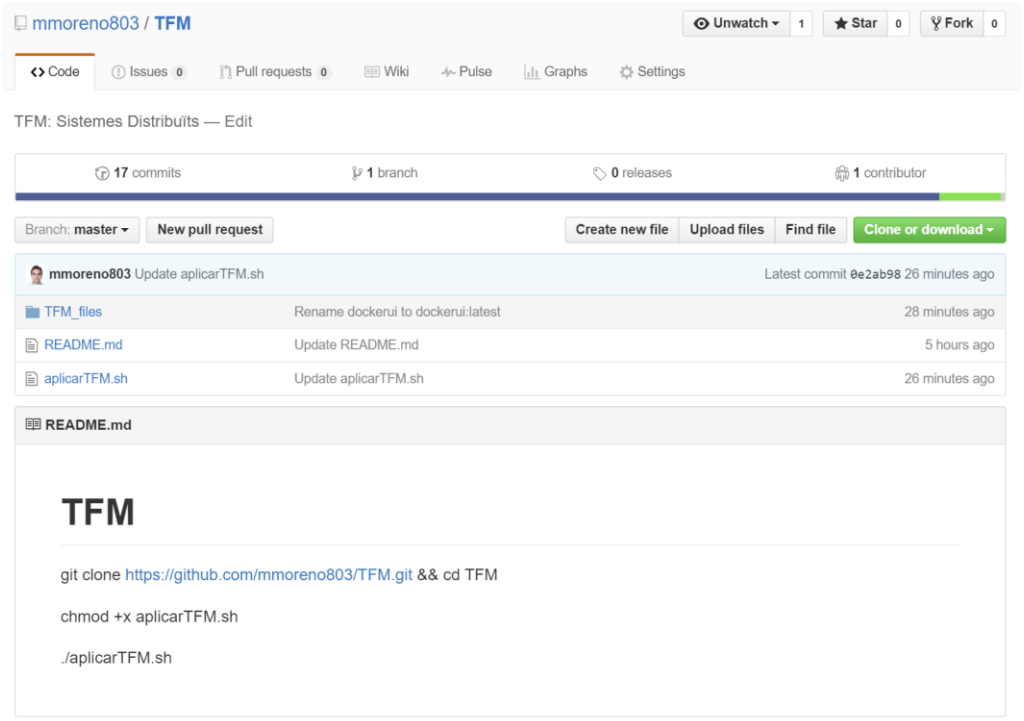
Definim un nou repositori a GitHub on depositem tots els fitxers necessaris juntament amb un script de deploy a l'estil del script *cloudynitzar.sh*

Contingut del nou script:

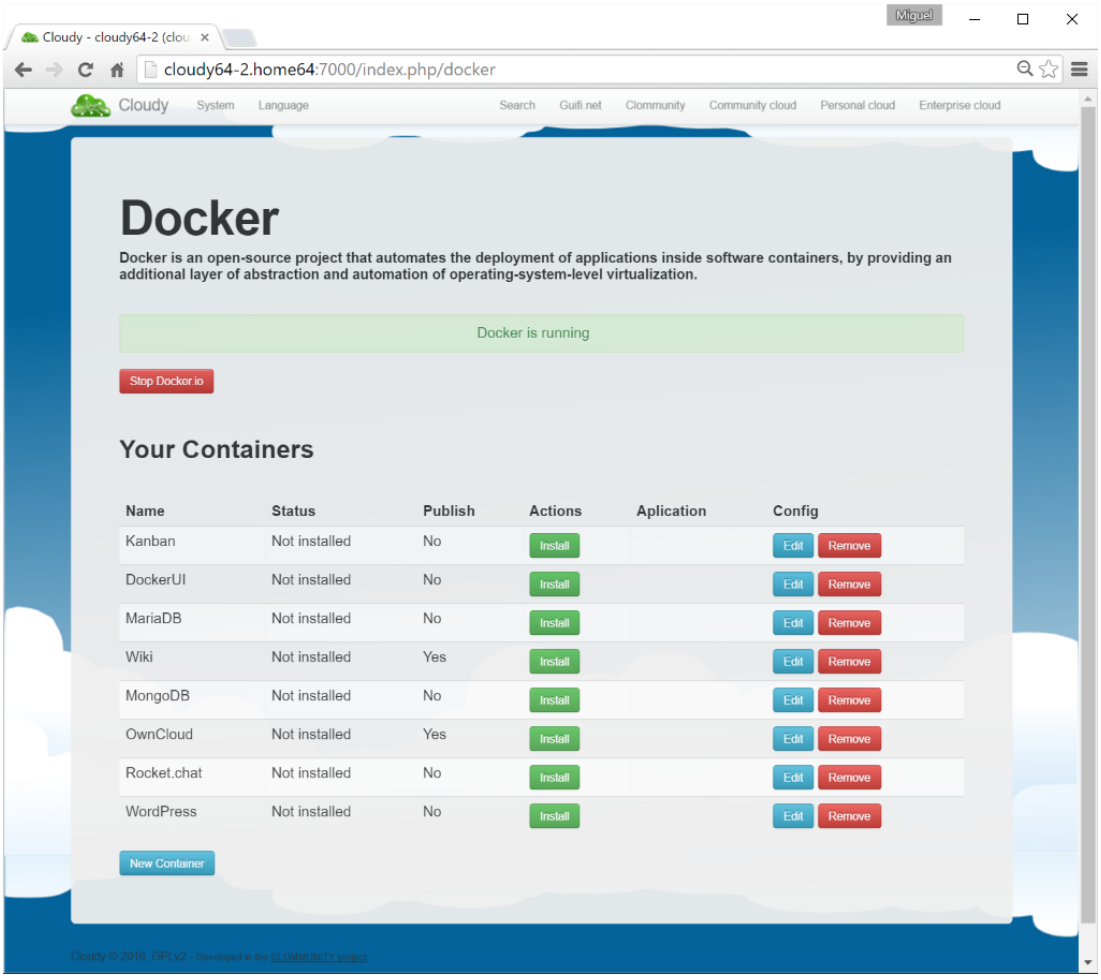
```
cp TFM_files/avahi/* /var/local/cDistro/plug/avahi/
cp TFM_files/php/* /var/local/cDistro/plug/controllers/
cp TFM_files/menus/* /var/local/cDistro/plug/menus/
mkdir /var/local/cDistro/plug/resources/docker
mkdir /var/local/cDistro/plug/resources/docker/containers
cp TFM_files/scripts/* /var/local/cDistro/plug/resources/docker/
chmod +x /var/local/cDistro/plug/resources/docker/*
cp TFM_files/config/* /var/local/cDistro/plug/resources/docker/containers/
```

El script és molt senzill i es limita a copiar fitxers a la seva ubicació definitiva, crear estructura de carpetes necessària i donar permisos als scripts que ho necessiten.

A la plana principal del repository a GitHub s'ha afegit una petita guia per instal·lar aquest software:



I aquest és el resultat d'executar-ho en un node "net":



3. Tasques pendents

No queden tasques pendents:

ID Tasca	Descripció Tasca	Estat

4. Resum

4.1.Sprint tancat

Amb aquest Sprint es tanca la fase d'implementació.

El balanç ha estat positiu on:

- S'han realitzat totes les tasques previstes entorn a Cloudy i Docker
- S'han implantat noves funcionalitats no planificades des de l'inici i que s'han identificat en el transcurs dels Sprints
- Al segon Sprint es van descartar les històries d'usuari referents a Sandstorm ja que les funcionalitats desenvolupades per Docker donen una funcionalitat similar.

Per tant, el resultat final s'ajusta a la planificació i el producte obtingut cobreix les necessitats requerides.

Annex VII: Codi

Docker.php

```
<?php
$urlpath="$staticFile/docker";
$docker_pkg = "docker.io";
$dev = "docker0";

function index() {
        global $title, $urlpath, $docker_pkg, $staticFile;
        $page = hlc(t("docker_title"));
        $page .= hl(t("docker_desc"), 4);

        if (!isPackageInstall($docker_pkg)) {
                $page .= "<div class='alert alert-error text-center'>".t("docker_not_installed")."</div>\n";
                $page .= addButton(array('label'=>t("docker_install"),'class'=>'btn btn-success', 'href'=>"$urlpath/install"));
                return array('type'=>'render','page'=>$page);
        } elseif (!isRunning()) {
                $page .= "<div class='alert alert-error text-center'>".t("docker_not_running")."</div>\n";
                $page .= addButton(array('label'=>t("docker_start"),'class'=>'btn btn-success', 'href'=>"$urlpath/start"));
                $page .= addButton(array('label'=>t('docker_remove'),'class'=>'btn btn-danger', 'href'=>$staticFile.'/default/uninstall/'.$docker_pkg));
                return array('type'=>'render','page'=>$page);
        } else {
                //$page .= ptxt(info_docker());
                $page .= "<div class='alert alert-success text-center'>".t("docker_running")."</div>\n";
                $page .= addButton(array('label'=>t("docker_stop"),'class'=>'btn btn-danger', 'href'=>"$urlpath/stop"));

              //Codi modificat: Docker GUI
              $page = docker_Admin($page);

                return array('type' => 'render','page' => $page);
        }
}

function isRunning(){
        $cmd = "/usr/bin/docker ps";
        $ret = execute_program($cmd);
  return ( $ret['return'] ==  0 );
}
function install(){
  global $title, $urlpath, $docker_pkg;
  $page = package_not_install($docker_pkg,t("docker_desc"));
  return array('type' => 'render','page' => $page);
}
```

```
function start() {
        global $urlpath;

        execute_program_detached("service docker start");
        setFlash(t('docker_start_message'),"success");
        return(array('type'=> 'redirect', 'url' => $urlpath));
}
function stop() {
        global $urlpath;

        execute_program_detached("service docker stop");
        setFlash(t('docker_stop_message'),"success");
        return(array('type'=> 'redirect', 'url' => $urlpath));
}
function info_docker(){
        global $dev;

        $cmd = "/sbin/ip addr show dev $dev";
        $ret = execute_program($cmd);
  return ( implode("\n", $ret['output']) );

}


////////////////////////////////////////////////////////////////////
//Codi afegit per:                                                //
//   Miguel Moreno - mmorenoreal@uoc.edu - mmoreno803@gmail.com   //
//   UOC - MEI - TFM: Sistemes DistribuÃ¯ts)                        //
////////////////////////////////////////////////////////////////////


//Funcions Principals//
//-------------------//

//FunciÃ³ que crea la taula amb la informaciÃ³ a mostrar i els botons
//associats a cada contenidor
function docker_Admin($page_tmp){
        global $dev,$urlpath;
       $containerspath="/var/local/cDistro/plug/resources/docker/containers/";

        $page_tmp .= hl("", 4);
        $page_tmp .= hl("Your Containers", 2);

       //llistem els fitxers existents
       $cmd = "ls ".$containerspath;
       $llista = execute_program($cmd);
```

```
	//Iniciem la taula
	 $page_tmp .= addTableHeader(array("Name","Status","Publish","Actions","Aplication","Config"), array('class'=>'table table-striped'));
	$taula_running="";
	$taula_stopped="";
	$taula_notinstalled="";

	//Per a cada fitxer trobat
	foreach($llista['output'] as $container){
		//Obtenim info dels fitxers i de docker
		$cInfo = get_container_info($container);
		$cDocker = get_docker_info($container);
		//Definim botons
		$bStart=botonStartStop($cInfo,$cDocker);
		$bDelete=botonDelete($cInfo,$cDocker);
		$bPublish=botonPublish($cInfo,$cDocker);
		$bApp=botonApp($cInfo,$cDocker);
		$bInstall=botonInst($cInfo,$cDocker);
		$bEdit=botonEdit($cInfo,$cDocker);
		$bRemove=botonRemove($cInfo,$cDocker);
		$bNew=botonNouContainer();

		//Si està  instal·lat, es mostra la botonera
		if($cDocker['id']!="NotFound"){
			if($cDocker['status']=="Running"){
				$taula_running .= addTableRow(array($cInfo['name'],$cDocker[status],$cInfo['pub'],$bStart.$bDelete,$bPublish.$bApp,$bEdit));
			}else{
				$taula_stopped .= addTableRow(array($cInfo['name'],$cDocker[status],$cInfo['pub'],$bStart.$bDelete,$bPublish.$bApp,$bEdit));
			}
		}else{  //Si no ho estÃ , es dona l'opciÃ³ de instalÂ·lar-lo
			if($cInfo['id']=="Installing"){
				//Si s'estÃ  instalÂ·lant
				$taula_notinstalled .= addTableRow(array($cInfo['name'],"Installing..."," "," ","Wait please..."));
			}else{
				//Si no ho estÃ , es dona l'opciÃ³ de instalÂ·lar-lo
				$taula_notinstalled .= addTableRow(array($cInfo['name'],"Not installed",$cInfo['pub'],$bInstall,"",$bEdit.$bRemove));
			}
		}
	}
	//Taula: primer es mostren els containers en execuciÃ³
	$page_tmp .= $taula_running;
	$page_tmp .= $taula_stopped;
	$page_tmp .= $taula_notinstalled;
	//Tanquem la taula i retornem resultat
	$page_tmp .= addTableFooter();
	$page_tmp .= $bNew;
	 return ( $page_tmp );
}
```

```
//FunciÃ³ auxiliar per obtenir la informaciÃ³ d'una imatge en concret
//Si existeix el fitxer de configuraciÃ³, retorna tota la informaciÃ³
//Si no existeix, retorna 0
function get_container_info($image){
        global $dev,$urlpath;
        $containerspath="/var/local/cDistro/plug/resources/docker/containers/";

        //Llegim el fitxer de configuraciÃ³
        $cmd = "cat ".$containerspath.$image;
        $properties = execute_program($cmd);

        //Existeix el fitxer de configuraciÃ³?
        if(count($properties['output'])==1){
                return("0"); //Retornem Error
        }else{
                //Es recuperen les propietats
                foreach ($properties['output'] as $prop){
                        $prop=explode("=",$prop);
                        switch($prop[0]){
                                case "Name":
                                        $name=$prop[1]; break;
                                case "Run":
                                        $run=$prop[1]; break;
                                case "Id":
                                        $id=$prop[1]; break;
                                case "Port":
                                        $port=$prop[1]; break;
                                case "Img":
                                        $img=$prop[1]; break;
                                case "Pub":
                                        $pub=$prop[1]; break;
                        }
                }
        }
        return(array('name'=>$name,'run'=>$run,'id'=>$id,'port'=>$port,'img'=>$img,'pub'=>$pub));
}


//FunciÃ³ auxiliar per obtenir la informaciÃ³ del contenidor docker en execuciÃ³
//per una determinada imatge.
//Si la imatge no existeix al sistema Docker retorna ("0","Not Installed)
//Si existeix, retorna el seu ID i Status
function get_docker_info($image){
        global $dev,$urlpath;

        //Consultem els contenidors existents
        $cmd = "docker ps -a";
```

```
        $ret = execute_program($cmd);

        //Fem servir la capÃ§alera per trobar la posiciÃ³ de Status
        $status_pos = strpos($ret['output'][0],"STATUS");

        //Inicialitzem variables per si no trobem el contenidor
       $id="NotFound";
        $status="Not Installed";
        //Busquem a cada contenidor
        foreach($ret['output'] as $container){
                if(strpos($container, $image)){
                        $id=substr($container,0,12);  //Obtenim ID
                        //$id=1;
                     $status=substr($container,$status_pos,2); //Status
                        if($status=="Up") $status="Running";
                        else $status="Stopped";
                     $info=substr($container,0,12);
                }
        }
        return(array('id'=>$id, 'status'=>$status));
}


//DefiniciÃ³ dels botons//
//--------------------//

function botonStartStop($cInfo,$cDocker){
       global $dev,$urlpath;
       if($cDocker['status']=="Running"){
              return(addButton(array('label'=>"Stop",'class'=>'btn btn-danger', 'href'=>"$urlpath/containerStop/".$cInfo['img'])));
       }else{
              return($botonOn=addButton(array('label'=>"Start",'class'=>'btn btn-success', 'href'=>"$urlpath/containerStart/".$cInfo['img'])));
       }
}

function botonDelete($cInfo,$cDocker){
       global $dev,$urlpath;
       if($cDocker['status']=="Running"){
              return(addButton(array('label'=>"Uninstall", 'class'=>'btn btn-default disabled')));
        }else{
              return(addButton(array('label'=>"Uninstall",'class'=>'btn btn-danger', 'href'=>"$urlpath/containerDelete/".$cInfo['img'])));
        }
}

function botonPublish($cInfo,$cDocker){
       global $dev,$urlpath;
       if($cDocker['status']=="Running" && $cInfo['pub']=="No"){
```

```
                return(addButton(array('label'=>"Publish",'class'=>'btn btn-info', 'href'=>"$urlpath/containerPublish/".$cInfo['img'])));
        }else if($cDocker['status']=="Running" && $cInfo['pub']=="Yes"){
                 return(addButton(array('label'=>"Unpubli",'class'=>'btn btn-info', 'href'=>"$urlpath/containerUnpublish/".$cInfo['img'])));
        }else{
                return(addButton(array('label'=>"Publish",'class'=>'btn btn-default disabled')));
        }
}
function botonApp($cInfo,$cDocker){
        //obtenim el hostname
        $cmd="hostname -f";
        $hostname=execute_program($cmd);

        //Si el contenidor estÃ  en execuciÃ³, mostrem el botÃ³
        if($cDocker['status']=="Running"){
                return(addButton(array('label'=>"Enter App",'class'=>'btn btn-info', 'href'=>"http://".$hostname['output'][0].":".$cInfo['port'])));
        }else{
                return(addButton(array('label'=>"Enter App",'class'=>'btn btn-default disabled')));
        }
}

function botonInst($cInfo,$cDocker){
        global $dev,$urlpath;
        return(addButton(array('label'=>"Install",'class'=>'btn btn-success', 'href'=>"$urlpath/containerInstall/".$cInfo['img'])));
}

function botonEdit($cInfo,$cDocker){
        global $dev,$urlpath;

                return(addButton(array('label'=>"Edit",'class'=>'btn btn-info', 'href'=>"$urlpath_/index.php/docker-form/index/".$cInfo['img'])));

}

function botonNouContainer($cInfo,$cDocker){
        global $dev,$urlpath;

                return(addButton(array('label'=>"New Container",'class'=>'btn btn-info', 'href'=>"$urlpath_/index.php/docker-form/index/")));

}

function botonRemove($cInfo,$cDocker){
        global $dev,$urlpath;
        return(addButton(array('label'=>"Remove",'class'=>'btn btn-danger', 'href'=>"$urlpath/removeConfig/".$cInfo['img'])));
}
```

```
//Accions dels botons//
//-------------------//

function containerStart() {
        global $urlpath,$Parameters;
       $respath="/var/local/cDistro/plug/resources/docker/";

       //Recollim els parÃ metres
       $cIm = $Parameters[0];

       //Obtenir informaciÃ³ de Docker
        $con = get_docker_info($cIm);
        $cId = $con['id'];

       //Arranquem el contenidor
        execute_program_detached("docker start CONTAINER ".$cId);
       setFlash("Container Started","success");

       //Publiquem el servei si aplica
        $conInfo=get_container_info($cIm);
       if($conInfo['pub']=="Yes"){
               execute_program_detached("/usr/sbin/avahi-ps publish ".$conInfo['name']." Docker ".$conInfo['port']);
              execute_program_detached($respath."monitor ".$conInfo['port']);
              setFlash("Container Published ".$cIm."success");
       }
        return(array('type'=> 'redirect', 'url' => $urlpath));
}

function containerStop() {
        global $urlpath,$Parameters;

       //Recollim els parÃ metres
       $cIm = $Parameters[0];

       //Obtenir informaciÃ³ de Docker
       $con = get_docker_info($cIm);
       $cId = $con['id'];

       //Es para el contenidor i es despublica
       execute_program_detached("docker stop CONTAINER ".$cId);
        setFlash("Container Stopped. ","success");
       $conInfo=get_container_info($cIm);
       execute_program_detached("/usr/sbin/avahi-ps unpublish Docker ".$conInfo['port']);

       sleep(2); //Esperem dos segons per donar temps al contenidor per parar-se
       return(array('type'=> 'redirect', 'url' => $urlpath));
}
```

```
function containerDelete() {
        global $urlpath,$Parameters;
       $respath="/var/local/cDistro/plug/resources/docker/";

       //Recollim els parÃ metres
        $cIm = $Parameters[0];

        //Obtenir informaciÃ³ de Docker
        $con = get_docker_info($cIm);
        $cId = $con['id'];
       $conInfo=get_container_info($cIm);

        execute_program_detached("docker rm ".$cId);
       execute_program_detached($respath."delete ".$conInfo['img']);

        setFlash("Container Deleted. ","success");
        return(array('type'=> 'redirect', 'url' => $urlpath));
}

function containerPublish() {
        global $urlpath,$Parameters;
        $respath="/var/local/cDistro/plug/resources/docker/";

        //Obtenim el contenidor a Publicar (ParÃ metre de la funciÃ³)
        $con = $Parameters[0];
        //Obtenim informaciÃ³ associada al contenidor
        $conInfo=get_container_info($con);

        //Publiquem el servei
        execute_program_detached("/usr/sbin/avahi-ps publish ".$conInfo['name']." Docker ".$conInfo['port']);
        //Modifiquem el parÃ metre de configuraciÃ³
        execute_program_detached($respath."publish ".$conInfo['img']);

        setFlash("Container Published. ","success");
        return(array('type'=> 'redirect', 'url' => $urlpath));
}

function containerUnpublish() {
        global $urlpath,$Parameters;
        $respath="/var/local/cDistro/plug/resources/docker/";

        //Obtenim el contenidor a Publicar (ParÃ metre de la funciÃ³)
        $con = $Parameters[0];
        //Obtenim informaciÃ³ associada al contenidor
        $conInfo=get_container_info($con);

        //Publiquem el servei
```

```
        execute_program_detached("/usr/sbin/avahi-ps unpublish Docker ".$conInfo['port']);
        //Modifiquem el parÃ metre de configuraciÃ³
        execute_program_detached($respath."unpublish ".$conInfo['img']);

        setFlash("Container Unpublished. ","success");
        return(array('type'=> 'redirect', 'url' => $urlpath));
}


function removeConfig() {
        global $urlpath,$Parameters;
        $respath="/var/local/cDistro/plug/resources/docker/";

        //Obtenim el contenidor a Publicar (ParÃ metre de la funciÃ³)
        $con = $Parameters[0];
        //Obtenim informaciÃ³ associada al contenidor
        $conInfo=get_container_info($con);

        //Esborrem el fitxer de configuraciÃ³
        execute_program_detached($respath."remove ".$conInfo['img']);

        setFlash("Container Removed. ","success");
        return(array('type'=> 'redirect', 'url' => $urlpath));

}

function containerInstall() {
       global $urlpath,$Parameters;
       $respath="/var/local/cDistro/plug/resources/docker/";

        //Obtenim el contenidor a InstalÂ·lar (ParÃ metre de la funciÃ³)
        $con = $Parameters[0];
        //Obtenim informaciÃ³ associada al contenidor
        $conInfo=get_container_info($con);

       //executem la instalÂ·laciÃ³
       execute_program_detached($respath."install ".$conInfo['run']);
       //execute_program_detached("docker run -d -p 8084:80 mkodockx/docker-kanban");
       //Alertem que el contenidor estÃ  en fase d'instalÂ·laciÃ³
       execute_program_detached($respath."installing ".$conInfo['img']);

       setFlash("Installing Container... ".$conInfo['run'],"success");
       return(array('type'=> 'redirect', 'url' => $urlpath));
}
```

Docker-form.php

```
<?php
$urlpath="$staticFile/docker";
$docker_pkg = "docker.io";
$dev = "docker0";

function index() {
        global $title, $urlpath, $docker_pkg, $staticFile, $Parameters;

       //Recollim els parÃ metres
        $image = $Parameters[0];
       if($image==""){
              $con=array();
       }else{
              $con = get_container_info($image);
       }

       //CapÃ§alera
       $page = h1c(t("docker_title"));
        $page .= h1("Docker Container Details", 3);

       //Formulari
       $page .= createForm(array('class'=>'form-horizontal'));
       $page .= addInput('Name',"Name",$con['name'],array('type'=>'text','required'=>''),"","Container display name");
       $page .= addInput('Run',"Command",$con['run'],array('type'=>'text','size'=>200,'required'=>''),"","Container execution command");
       $page .= addInput('Port',"Port",$con['port'],array('type'=>'number', 'min' => '1024', 'max' => '65535', 'required'=>''),"","The port on which the App
                will publish de service");
       $page .= addInput('Img',"Image",$con['img'],array('type'=>'text','required'=>''),"","Container Image");
       if($con['pub']=="Yes"){
              $page .= addCheckbox2('Pub',"Publish",array("Yes","No"),array('type'=>'text','required'=>''),"","Determine if the App will be published on
                     Serf");
       }else{
              $page .= addCheckbox2('Pub',"Publish",array("No","Yes"),array('type'=>'text','required'=>''),"","Determine if the App will be published on
                     Serf");
        }
       $page .= addInput('Id',"Container ID","ndef",array('type'=>'text','size'=>'200'),"readonly","Container ID. Not Editable");
       $page .= addSubmit(array('label'=>t("serf_button_save"),'class'=>'btn btn-default','divOptions'=>array('class'=>'btn-group')));


        return array('type' => 'render','page' => $page);
}
```

```
function index_post(){

        global $staticFile, $staticPath, $urlpath;
        global $avahipsetc_config;


        $datesToSave = array();
        foreach ($_POST as $key => $value) {
                $datesToSave[$key] = $value;
        }

       $file="/var/local/cDistro/plug/resources/docker/containers/".$datesToSave['Img'];
        write_conffile($file,$datesToSave,"","","'");

        setFlash("Config -> OK","info");
        return(array('type'=> 'redirect', 'url' => $staticFile."/docker"));
}

//FunciÃ³ auxiliar per obtenir la informaciÃ³ d'una imatge en concret
//Si existeix el fitxer de configuraciÃ³, retorna tota la informaciÃ³
//Si no existeix, retorna 0
function get_container_info($image){
        global $dev,$urlpath;
        $containerspath="/var/local/cDistro/plug/resources/docker/containers/";

        //Llegim el fitxer de configuraciÃ³
        $cmd = "cat ".$containerspath.$image;
        $properties = execute_program($cmd);

        //Existeix el fitxer de configuraciÃ³?
        if(count($properties['output'])==1){
                return("0"); //Retornem Error
        }else{
                //Es recuperen les propietats
                foreach ($properties['output'] as $prop){
                        $prop=explode("=",$prop);
                        switch($prop[0]){
                                case "Name":
                                        $name=$prop[1]; break;
                                case "Run":
                                        $run=$prop[1]; break;
                                case "Id":
                                        $id=$prop[1]; break;
                                case "Port":
                                        $port=$prop[1]; break;
                                case "Img":
                                        $img=$prop[1]; break;
```

```
                                case "Pub":
                                        $pub=$prop[1]; break;
                        }
                }
        }
        return(array('name'=>$name,'run'=>$run,'id'=>$id,'port'=>$port,'img'=>$img,'pub'=>$pub));
}

//Redefinim la funciÃ³  addCheckbox de la llibreria lib/form.php
//L'objectiu Ã©s canviar el valor de cada option
function addCheckbox2($name=null, $label= null, $value = null, $options = null, $attributes = null, $tooltip = null, $nobr = null){
        if (!is_null($name)){
                $options['name'] = $name;
        }

        $str = "";
        $str .= "<div class='control-group'>\n";
        $str .="<label class='control-label'>$label:</label>\n";
        $str .="<div class='controls'>\n";
        $str .= "<select ";
        if (is_array($options)){
                foreach($options as $k=>$v){
                        $str .= " $k='".$v."'";
                }
        }
        if (!is_null($attributes))
                $str .= $attributes;
        $str .= ">\n";

        foreach ($value as $k => $v) {
              //codi de la llibreria
                //$str .= "<option value='".$k."'>".$v."</option>";
              //nou codi
              $str .= "<option value='".$v."'>".$v."</option>";
        }

        $str .= "</select>";

        if (!is_null($tooltip))
                if (is_null($nobr))
                        $str .= '<br/>';
                $str .= '<span style="font-size: smaller;"><span style="font-size: smaller;">'.$tooltip.'</span></span>';
        $str .="</div>\n";
        $str .="</div>\n";

        return $str;
}
```

Docker.avahi.php

```
<?php
// plug/avahi/Docker.avahi.php

addAvahi('Docker','fDocker');

function fDocker($dates){
	global $staticFile;

	//Crea un botÃ³ apuntant cap a IP:Port del node que publica el servei
	return(addButton(array('label'=>"Enter App",'class'=>'btn btn-info', 'href'=>"http://".$dates['ip'].":".$dates['port'])));
}
```

La resta de fitxers creats i modificats al TFM es poden trobar al fitxer adjunt de l'annex VII (Fitxer *.tar)