



**Universitat Oberta  
de Catalunya**

# IMPLEMENTACIÓ D'UNA INFRAESTRUCTURA WEB AMB COREOS I DOCKER

Memòria de Projecte de Final de Grau

Grau Enginyeria Informàtica

UOC. Universitat Oberta de Catalunya

Autor: Raul Rivero Ramos

Consultor: Francesc Guim Bernat

Professor: Josep Jorba Esteve

## Agraïments

Vull agrair al consultor Francesc Guim Bernat, la seva dedicació, paciència i els consells que m'ha donat tots aquests mesos, ha estat un plaer treballar amb tu.

Vull agrair a la meva família la paciència que han tingut.

Han estat molts anys de compartir nervis, hores d'estudi i moments on hi era físicament però no hi era mentalment.

Han estat quasi 6 anys de compaginar estudi, feina i família, si no hagués estat per vosaltres no hauria acabat els estudis de grau.

Han estat moltes hores explicant-vos com anaven les assignatures, quins nervis tenia en entregar les pacs i mil neguits que anaven entrant quan s'aproximaven els exàmens.

Han estat moltes nits amb la llum de la tauleta oberta per repassar el temari i destorbant el vostre son.

Al final, hem aconseguit el somni que tots aquests anys havia perseguit, graduar-me com a enginyer informàtic, vosaltres sempre heu cregut en mi i això m'ha donat forces per tirar endavant amb les inclemències de tots aquests anys.

Aquest projecte de fi de grau el vull dedicar a les dues persones més importants de la meva vida, a tu Vanesa que ets la meva inspiració i m'has donat suport sempre, mai has dubtat de mi i t'ho agraeixo de tot cor.

A tu Pol, el nen més feliç del món, que encomana la seva felicitat allà on va, ets un crack, hem passat moltes hores junts pintant, escrivint i fent guixadots als apunts de la UOC, m'has donat la força que necessitava per poder acabar els estudis.

Gràcies de tot cor.

## CONTINGUT DEL PROJECTE

Aquest projecte consta dels següents documents:

Memòria descriptiva: A la memòria del projecte estan definits tots els apartats teòrics que han motivat el projecte, així mateix trobareu la introducció, la motivació, la descripció del projecte, l'abast de la proposta, l'organització de les tasques i el seu pla de treball, el pressupost del projecte i el desenvolupament de les tasques especificades.

Annexos: Trobareu un seguit d'annexos on s'especifica els passos que s'han realitzat per poder dur a terme les tasques que s'han realitzat al projecte, els documents annexes són els següents.

-Annex 1 .- En aquest annex s'explica amb captures de pantalles la instal·lació pas a pas d'un servidor KVM Xen server, des de la configuració del servidor físic per poder allotjar màquines virtuals fins a la instal·lació dels service pack necessaris per donar suport al sistema operatiu CoreOs.

-Annex 2 .- En aquest annex s'explica pas a pas la instal·lació del sistema operatiu CoreOS, la configuració dels fitxers Cloud-Config i la configuració i instal·lació d'un clúster fent servir aquest sistema operatiu.

-Annex 3 .- En aquest annex trobareu com executar una aplicació en un contenidor Docker i com fer servir fleet per tenir alta disponibilitat en els serveis executats.

-Annex 4 .- Aquest annex explica pas a pas la manera de configurar un emmagatzematge de xarxa i compartir un directori mitjançant NFS, així mateix s'explica pas a pas com configurar el clúster CoreOS per connectar en iniciar amb els directoris compartits al NAS.

-Annex 5 .- Instal·lació pas a pas del Servei FTP, creació del fitxer DockerFile, creació d'un compte al repository Docker Hub i us del repository per a la creació de les nostres pròpies imatges.

-Annex X .- Trobareu les especificacions tècniques del servidor que s'ha fet servir al projecte un servidor Dell PowerEdge 2950.

Fitxers del projecte: a les carpetes que acompanyen la memòria del projecte trobareu tots els documents que s'han fet servir per a dur a terme la implementació dels serveis, les proves de rendiment i la configuració del clúster.

## ÍNDEX

1.	Introducció.....	6
2.	Motivació.....	7
3.	Descripció del projecte.....	9
	3.1 Objectius.....	9
	3.2 Anàlisi de riscos.....	10
4.	Abast de la proposta.....	11
5.	Organització del projecte.....	12
6.	Pla de treball.....	13
	6.1 Relació d'activitats.....	13
	6.2 Fites principals.....	14
	6.3 Calendari de treball.....	15
7.	Valoració Econòmica.....	16
8.	Infraestructura de proves.....	18
9.	Docker.....	20
10.	CoreOS.....	22
	10.1 Etcad.....	23
	10.2 Fleet.....	24
	10.3 Clúster CoreOs.....	26
	10.4 Escalabilitat de la infraestructura.....	28
	10.4.1 Eliminar node del clúster.....	30
	10.4.2 Afegir node al clúster.....	30

10.4.3	Actualitzar node del clúster.....	31
10.5	Instal·lació de serveis.....	31
10.5.1	Servei Web.....	32
10.5.2	Servei Mysql.....	44
10.5.3	Servei Ftp.....	50
10.6	Disponibilitat dels serveis.....	53
11	Comparativa infraestructura virtual i infraestructura CoreOS.....	54
11.1	Comparativa d'arquitectures.....	54
11.2	Comparativa de rendiments.....	55
11.3	Comparativa d'escalabilitat.....	67
11.4	Comparativa de disponibilitats.....	68
12	Conclusions.....	69
12.1	Objectius assolits.....	70
12.2	Futur del projecte.....	72
13	Bibliografia.....	73
14	Annexos.....	75

Annex 1 Instal·lació Xen Server.

Annex 2 Instal·lació CoreOS.

Annex 3 Implementació de serveis i alta disponibilitat.

Annex 4 Configuració de directori compartit amb NFS al NAS.

Annex 5 Instal·lació de Servei FTP repository Docker Hub.

Annex X Especificacions tècniques servidor Dell PowerEdge 2950

## 1. INTRODUCCIÓ

Una empresa de Hosting molt coneguda ens ha demanat una solució als problemes de disponibilitat dels seus serveis. Tal com tenen muntada la infraestructura actual no poden complir les SLA'S<sup>1</sup> establertes als contractes amb els clients que requereixen un 96% de disponibilitat dels serveis contractats amb un indemnització per part de l'empresa al no complir-les.

Aquestes indemnitzacions estan duent l'empresa a replantejar-se els requeriments de les SLA'S i els preus dels serveis per poder mantenir el negoci. Quan un servidor virtual té un problema, tornar a restablir tots els serveis i clients que gestionava, comporta un temps que no poden reduir amb la infraestructura que tenen actualment.

Cada servidor virtual gestiona uns 500 usuaris, els serveis que ofereix l'empresa són, servidor web (apache), servidor mysql, servidor ftp, a part d'aquest serveis els servidors virtuals tenen instal·lat php. Quan un servidor apache falla, per exemple, l'empresa ha de mirar de recompondre la màquina des de la darrera rèplica<sup>2</sup> o snapshot<sup>3</sup> de la màquina virtual i fer córrer un script que actualitzi els fitxers de configuració de l'apache i dels demés serveis per afegir les altes noves o baixes de clients des del moment que es va fer la còpia fins al moment que va deixar de funcionar.

L'empresa compta amb una base de dades dels clients actius, a cada servidor, i amb quins productes té contractats i quins dominis ha de redirigir a les seves carpetes, les dades d'usuari es guarden en un servidor NFS<sup>4</sup> extern a les màquines virtuals, el qual esta compost per cabines de discos degudament redundades per garantir la fiabilitat i la disponibilitat de les dades en tot moment.

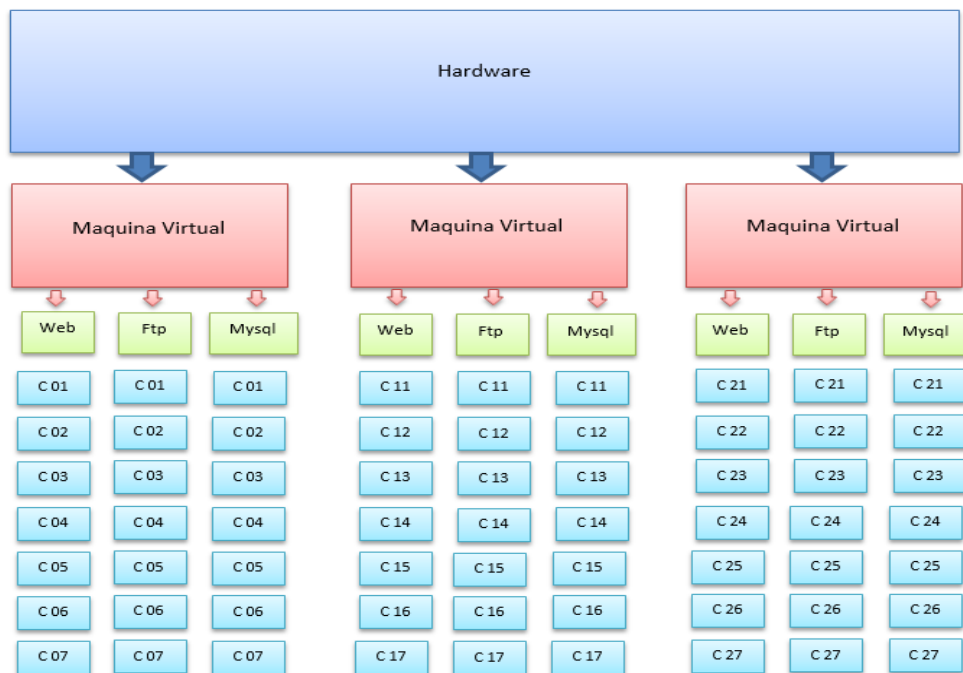
---

<sup>1</sup> **SLA:** Nivell d'acord de servei que dona una empresa a un client al contractar-lo.

<sup>2</sup> **Replica:** Còpia de la màquina virtual a una altra ubicació per poder engegar-la en cas de que l'original deixi de funcionar.

<sup>3</sup> **Snapshot:** Instantània d'una màquina virtual en un punt concret en el temps on podem anar si la màquina fallés.

<sup>4</sup> **NFS:** Sistema de fitxers en xarxa.



Il·lustració 1, Infraestructura Virtual, Font (elaboració pròpia)

## 2. MOTIVACIÓ

La motivació principal d'aquest projecte és intentar trobar una solució a la problemàtica de l'empresa oferint-los una infraestructura més modular on els serveis es puguin recompondre fàcilment i amb una alta disponibilitat.

Les infraestructures actuals basades en màquines virtuals necessiten molt maquinari per donar servei a tots els usuaris que gestionen les empreses de hosting, ja que per cada servidor web, per exemple, necessiten tenir per sota un sistema operatiu.

De la mateixa manera que les grans operadores han optat per noves solucions creant els dispositius de xarxa que fins ara han estat físics, limitant el creixement per motius d'ample de banda, n<sup>o</sup> d'accessos WAN, etc... en dispositius virtuals de software. Aquestes companyies virtualitzen els seus Firewalls, dispositius de balanceig de càrrega, etc... mitjançant NFV (Network Function Virtualized) per tenir un creixement més ràpid i poder adaptar-se a la demanda del client.

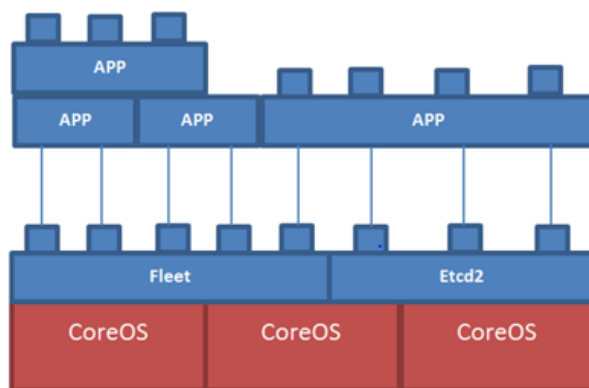
La tecnologia docker<sup>1</sup> permet implementar tots els servidors web, mysql, ftp, etc... que necessitem sobre un sol sistema operatiu, un sol kernel gestiona totes les aplicacions que el nostre hardware sigui capaç de suportar, permetent tenir més serveis cara a l'usuari amb menys recursos.

Fent ús de la tecnologia Docker, el sistema operatiu CoreOS, i alguns sistemes com fleet i etcd podrem crear una infraestructura amb alta disponibilitat reduint notablement la necessitat de hardware i permetent que en un possible desastre, no quedin molts usuaris sense servei evitant, d'aquesta manera, que l'empresa tingui pèrdues.

Aquest projecte realitza un estudi del sistema operatiu CoreOs i la seva implementació en clúster fent servir etcd i fleet per permetre l'alta disponibilitat de les aplicacions que farem córrer en contenidors, d'una manera modular, realitzant una proposta d'infraestructura per garantir la màxima disponibilitat dels serveis que ofereixen.

Convencerem a l'empresa fent una comparativa de les infraestructures fetes sobre màquines virtuals i les infraestructures basades en contenidors, tot comparant la facilitat de moure serveis entre sistemes operatius, la facilitat de recuperar un servei caigut i la diferència en el consum de recursos entre les dues infraestructures.

Per acabar, una de les principals motivacions és la falta d'informació que he trobat envers a la implementació d'aquest tipus d'infraestructures, crec que encara queda molt per fer, crec que les aplicacions basades en Dockers seran molt implementades a la majoria d'empreses del món.



**Il·lustració 2, Infraestructura CoreOs. Font (elaboració pròpia)**

<sup>1</sup> Dockers: Projecte que executa aplicacions dins de contenidors de software.

<sup>2</sup> CoreOS: Sistema operatiu basat en kernel de Linux d'una mida molt reduïda.

<sup>3</sup> etcd: Dimoni que gestiona els clústers de CoreOS

<sup>4</sup> fleet: Dimoni que gestiona els contenidors dins del clúster de CoreOS.



## 3. DESCRIPCIÓ DEL PROJECTE

Dissenyar un prototip d'infraestructura capaç de poder gestionar una empresa hosting amb molts clients, amb les seves respectives pàgines web i serveis contractats, millorant la modularitat, disponibilitat, versatilitat que les actuals infraestructures basades en màquines virtuals.

Dissenyar una infraestructura que permeti minimitzar l'impacte de fallada d'un servei. Si un servei falla no hauria ser motiu perquè 500 usuaris és quedin sense poder fer servir els seus productes contractats. L'objectiu d'aquest projecte és crear aquesta infraestructura fent servir Dockers per crear serveis individualitzats a un sol usuari i que sigui un sol usuari qui tingui el problema i poder engegar el servei en un temps mínim.

Fer el prototip d'aquesta infraestructura i fer proves de rendiment envers a capacitat de computació en les mateixes condicions, versatilitat, facilitat per moure clients d'una màquina a una altra, recuperació de l'entorn davant d'una catàstrofe i facilitat d'actualització de serveis.

### 3.1 OBJECTIUS

Enumerem a continuació els principals objectius del projecte:

- Realitzar una introducció teòrica del sistema operatiu CoreOS i dels programes que permeten fer córrer aplicacions en contenidors de software.
- Realitzar la instal·lació d'un prototip de 3 sistemes operatius CoreOS en clúster sobre una capa de virtualització amb Xen Server<sup>1</sup>.
- Realitzar la instal·lació dels serveis que necessiten els clients de l'empresa hosting, com ara, apache, mysql i ftp.

---

<sup>1</sup> **XenServer:** Hypervisor basat en un microkernel<sup>2</sup> que ens permet tenir molts sistemes operatius corrent sobre el mateix hardware.

<sup>2</sup> **Microkernel:** També conegut com a Micronucli, és la mínima quantitat de software per donar els serveis bàsics d'un sistema operatiu, com ara l'espai d'adreces de memòria, la planificació de processos i la comunicació entre processos.

-Realitzar una comparativa entre les infraestructures basades en màquines virtuals i les basades en Dockers, en matèria de:

- Consum de recursos.
- Facilitat per moure aplicacions.
- Temps de recuperació d'un servei.
- Impacte econòmic de la caiguda d'un servei.
- Escalabilitat.
- Portabilitat.

-Realitzar els informes de les comparatives dutes a terme.

-Documentar la instal·lació del clúster i els serveis per a una possible implantació a l'empresa sol·licitant del projecte.

### 3.2 ANÀLISIS DE RISCOS

Els riscos són els recursos temporals i econòmics. Necessitarem un maquinari capaç de suportar la càrrega de treball dels sistemes virtualitzats, tant de la infraestructura amb CoreOS com la infraestructura amb màquina virtual.

Per poder arribar al nivells de garantia requerits cal prendre les següents mesures preventives:

- Sol·licitar una màquina tipus servidor, a una institució, per garantir la disponibilitat de les infraestructures durant tot el projecte.
- Realitzar un seguiment setmanal de l'estat del projecte seguint la planificació acordada donant prioritat a les tasques amb data límit més propera.
- Centrar-me en l'abast del projecte per evitar pèrdua de recursos temporals i econòmics en fites no previstes.

## 4. ABAST DE LA PROPOSTA

El projecte es centrarà exclusivament en el prototipatge d'una infraestructura basada en Dockers per la posada en marxa de serveis orientats a clients d'una empresa Hosting, així com en la comparació de les característiques principals dels dos tipus d'infraestructures.

Per comparar el rendiment es farà servir una aplicació tipus benchmarking com ApacheBench.

Les comparatives és faran sobre una màquina virtual i el clúster ja que una aplicació sobre un Docker només corre sobre un dels CoreOs del clúster.

Les instal·lacions dels serveis es faran de forma manual, tot documentant el procediment, s'instal·laran servidors web Apache de manera que en el clúster de 3 CoreOS podrem córrer més de 3 servidors web.

Els serveis Mysql és crearan en tipus clúster per evitar el consum excessiu de recursos, sobretot memòria ram. Si fem un contenidor per a cada client els recursos hardware serien tan elevats que no seria una proposta òptima pel client.

La proposta econòmica contemplarà el cost del projecte, incloent els costos del lloguer del maquinari per a crear el pilot, les hores d'investigació, implementació i proves de funcionament, així com el material d'oficina necessari per dur a terme el prototipatge, documentació i presentació al client.

## 5. ORGANITZACIÓ DEL PROJECTE

Recursos necessaris per a la realització del projecte:

### 1. Recursos Humans

Tots els rols necessaris per a la realització del projecte els farà la mateixa persona per cal diferenciar-los.

Cap de projecte (CP): tindrà cura de que totes les tasques es realitzin en dates preestablertes.

Tècnic maquinari (TM): s'encarregarà de la instal·lació del servidor i de les connexions del mateix.

Tècnic Sistemes (TS): s'encarregarà de la instal·lació del sistema base hypervisor (XenServer) i de la configuració de la xarxa, bios del servidor, etc...

Enginyer de Sistemes (ES): s'encarrega del disseny de la infraestructura, prototipatge i implementació de la solució proposada.

Tècnic Qualitat (TQ): s'encarrega de que les tasques que coordina el CP es realitzin correctament.

### 2. Recursos Temporals

Com el nombre de crèdits del TFG és de 12 ECTS, el nombre global d'hores invertides no hauria de superar les 225 hores.

### 3. Recursos Materials

Documentals: accés a internet per realitzar consultes dels sistemes operatius basats en Dockers, sistemes hypervisor, Unix, etc... Aquests recursos són els que crearan l'apartat de bibliografia del projecte.

Tecnològics: Un servidor suficientment potent per gestionar les infraestructures esmentades, internet per descarrega d'aplicacions tant per Dockers com sistemes operatius. Eines per a la creació de la memòria, com ara, eines ofimàtiques, eines de creació d'imatges per il·lustrar idees, eines de creació del diagrama de Gant i eines per creació del pressupost del projecte.

El pla de treball següent determinarà la distribució dels recursos anomenats anteriorment.

## 6.PLA DE TREBALL

### 6.1 RELACIÓ D'ACTIVITATS

Llista ordenada en l'ordre en el que les executarem. Podem diferenciar les tasques pròpies d'implementar el programari base al maquinari per poder muntar el prototip i les tasques pròpies d'implementar la infraestructura proposada i per últim les proves de rendiment envers a les infraestructures basades en màquines virtuals.

1. Cercar informació sobre el sistema operatiu CoreOS, etcd, fleet i dockers, requisits mínims d'instal·lació, arquitectures suportades, protocols de xarxa suportats, etc...
2. Descarregar programari necessari per fer el projecte, Xen Server i CoreOs, en les seves versions estables i actualitzades.
3. Implementació del sistema Hypervisor, Xen Server, base al servidor de test del projecte.
4. Documentació de la instal·lació del Hypervisor.
5. Implementació del sistema operatiu CoreOS al nou servidor i verificació de funcionament.
6. Documentació de la instal·lació del sistema operatiu CoreOS.
7. Implementació del clúster dels sistemes CoreOS i proves de funcionament.
8. Configuració del etcd discover per crear el clúster i verificació del funcionament.
9. Configuració de Fleet per crear alta disponibilitat dels serveis i proves de funcionament.
10. Documentació de la implementació del clúster i els dimonis etcd i fleet.
11. Creació de servei de proves, per exemple apache, per determinar el correcte funcionament del clúster, etcd i fleet aturant una màquina del clúster per comprovar el seu comportament.
12. Implementació del servei web per als clients de l'empresa.
13. Implementació del servei mysql per als clients de l'empresa.
14. Implementació del servei ftp per als clients de l'empresa.
15. Documentació de les implementacions dels serveis realitzades.
16. Instal·lació de servidor LAMP al Hypervisor per fer proves de rendiment.
17. Realitzar proves de rendiment servidor web al clúster.
18. Realitzar proves de rendiment servidor web a la màquina virtual.
19. Documentació dels resultats.
20. Documentar les conclusions a les quals he arribat.
21. Realitzar la memòria del projecte.

## 6.2 FITES PRINCIPALS

El Cap del projecte vetllarà en tot moment perquè les fites principals del projecte quedin assolides tal i com detallem en aquesta proposta.

Taula 1. Fites del projecte

FITA	DESCRIPCIÓ
F1	Instal·lació i configuració de l'entorn Hypervisor
F2	Instal·lació i configuració del clúster CoreOS amb etcd, fleet i docker.
F3	Implementació dels serveis pels usuaris.
F4	Comparativa entre les dues arquitectures.

## 6.3 CALENDARI DE TREBALL

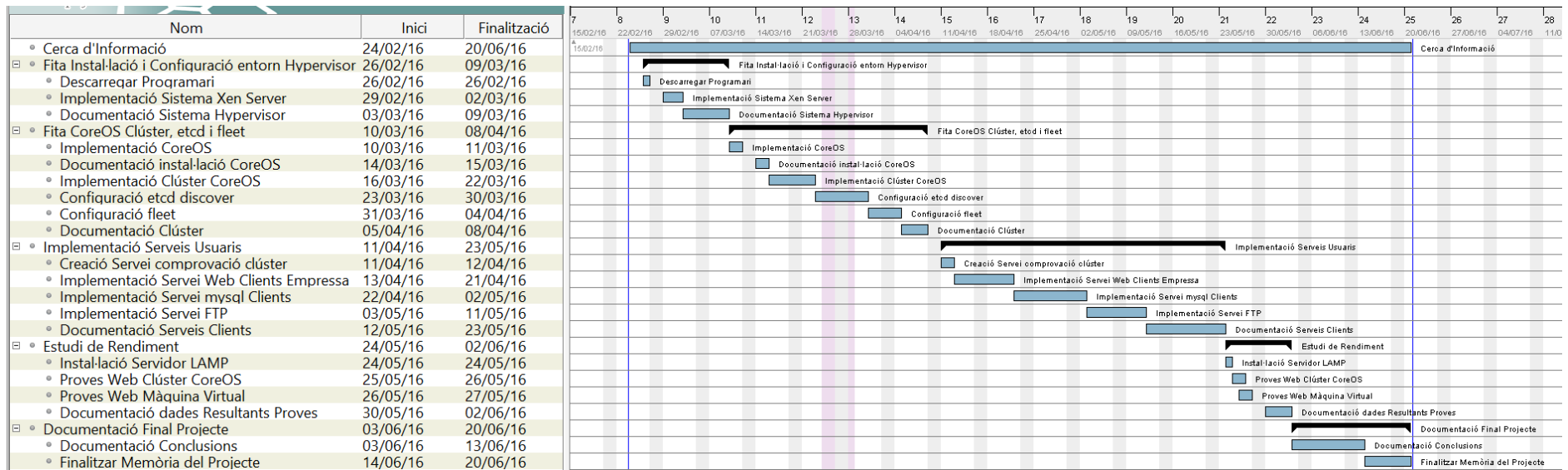
Per arribar a assolir les fites proposades al projecte, he d'establir un calendari de treball. Aquest estarà compost per blocs setmanals de dilluns a divendres tenint en compte els festius.

El semestre comença el dimecres dia 24 de febrer de 2016 i acaba el dilluns 20 de juny també de 2016, això comporta un total de 78 dies laborables treient la Setmana Santa i el 2 de maig.

La jornada laboral serà a la nit de 21:00 a 00:00, en total 78 dies per 3 hores diàries tindrem un còmput total de 234 hores, aproximadament les 225 establertes al punt 5 d'aquest document.

Al diagrama de Gantt següent podrem veure la relació de tasques com s'executen en el temps, les dates d'inici de les activitats a realitzar i les dates del seu tancament.

# IMPLEMENTACIÓ D'UNA INFRAESTRUCTURA WEB AMB COREOS I DOCKER



Il·lustració 3 Diagrama de Gantt del Projecte – Font (elaboració pròpia)

## 7. VALORACIÓ ECONÒMICA

Deixant a banda el cost d'implementació de la nostra infraestructura al client, s'ha de determinar el cost de fer el projecte i la rendibilitat que es suposarà si el pot replicar a més d'un client. El cost de la investigació i la creació de programari per poder implementar en els clients una infraestructura moderna que els permeti tenir major disponibilitat, fiabilitat, portabilitat, etc... no ha incrementar el pressupost del client el cost de fer el projecte.

Ara determinarem quin cost té la investigació i la implementació d'un pilot. Amb aquest cost ja podríem realitzar un estudi de viabilitat del projecte determinant quina quantitat de clients necessitem per rendibilitzar aquesta inversió de temps i recursos que estem realitzant.

Separarem les despeses per partides pressupostàries sent les partides que comencen per l'1 són per despeses de personal, les partides que comencen per 2 per despeses de maquinari i les partides que comencen per 3 són per despeses de material d'oficina.



IMPLEMENTACIÓ D'UNA INFRAESTRUCTURA WEB AMB COREOS I DOCKER

ID	PARTIDA PRESSUPUSTÀRIA	CONCEPTE	COST	HORES
1	11000	Cercar informació sobre el sistema operatiu CoreOS, etcd, fleet i dockers, etc...	2.500,00 €	50
<b>INSTAL·LACIÓ I CONFIGURACIÓ XEN SERVER</b>				
2	11000	Descarregar programari necessari per fer el projecte.	150,00 €	3
3	11000	Implementació del sistema Hypervisor al servidor.	450,00 €	9
4	11000	Documentació de la instal·lació del Hypervisor.	750,00 €	15
<b>INSTAL·LACIÓ I CONFIGURACIÓ COREOS</b>				
5	11000	Implementació del sistema operatiu CoreOS al nou servidor i verificació de funcionament.	300,00 €	6
6	11000	Documentació de la instal·lació del sistema operatiu CoreOS.	300,00 €	6
7	11000	Implementació del clúster dels sistemes CoreOS i proves de funcionament.	750,00 €	15
8	11000	Configuració del etcd discover per crear el clúster i verificació del funcionament.	900,00 €	18
9	11000	Configuració de Fleet per crear alta disponibilitat dels serveis i proves de funcionament.	750,00 €	15
10	11000	Documentació de la implementació del clúster i els dimonis etcd i fleet.	600,00 €	12
<b>INSTAL·LACIÓ I CONFIGURACIÓ DE SERVEIS DE CLIENT</b>				
11	11000	Creació de servei de proves per determinar el correcte funcionament del clúster.	300,00 €	6
12	11000	Implementació del servei web per als clients de l'empresa.	1.050,00 €	21
13	11000	Implementació del servei mysql per als clients de l'empresa.	1.050,00 €	21
14	11000	Implementació del servei ftp per als clients de l'empresa.	1.050,00 €	21
15	11000	Documentació de les implementacions dels serveis realitzades.	1.200,00 €	24
<b>ESTUDI DE RENDIMENT VIRTUALITZAT VERSUS COREOS</b>				
16	11000	Instal·lació de servidor LAMP al Hypervisor per fer proves de rendiment.	150,00 €	3
17	11000	Realitzar proves de rendiment servidor web al clúster.	300,00 €	6
18	11000	Realitzar proves de rendiment servidor web a la màquina virtual.	300,00 €	6
19	11000	Documentació dels resultats.	600,00 €	12
<b>DOCUMENTACIÓ FINAL PROJECTE</b>				
20	11000	Documentar les conclusions a les quals hem arribat.	1.050,00 €	21
21	11000	Finalitzar la memòria del projecte.	750,00 €	15
<b>DESPESES MATERIALS</b>				
22	21000	Lloger de Servidor per implentar prova pilot (DELL PowerEdge 2950)	600,00 €	
23	32002	Material oficina ( consumibles, material presentació, cd, pendrive, etc...)	500,00 €	
<b>Total Pressupost</b>			<b>16.350,00 €</b>	
<b>Impostos (I.V.A 21%)</b>			<b>3433,5</b>	
<b>Total amb I.V.A inclòs</b>			<b>19.783,50 €</b>	

## 8. INFRAESTRUCTURA DE PROVES

L'objectiu del projecte és migrar la infraestructura actual basada en màquines virtuals d'una empresa en una infraestructura que millori la disponibilitat, l'escalabilitat i la modularitat. Per fer aquest estudi farem servir un servidor de proves on podrem implementar les dues infraestructures i, d'aquesta manera, poder comparar-les fent un estudi exhaustiu de les característiques d'ambdues opcions.

Per poder dur a terme aquest estudi s'implementarà un software hypervisor sobre el maquinari. Un programari hypervisor és una capa de programari que permet instal·lar altres sistemes operatius en capes superiors fent un maquinari virtual per a cada sistema operatiu que es vulgui instal·lar.

El programari hypervisor que he escollit és XenServer, un programari amb llicència OpenSource que ens permet realitzar, per una banda, el clúster de CoreOS per implementar els serveis i fer proves de rendiment, i per una altra, instal·lar una màquina Linux amb els mateixos serveis instal·lats que el clúster i comparar els valors obtinguts a les proves.

Existeixen alternatives a XenServer, com ara, VMWARE una alternativa que té una versió lliure però amb moltes més limitacions que XenServer. Un altre programari molt interessant que he trobat a la meua recerca d'informació es diu PROXMOX, un hypervisor OpenSource basat en una distribució Debian de Linux amb una consola d'administració web que ens permet dos tipus de virtualització, una basada en contenidors amb LXC<sup>1</sup> i una altra virtualització completa amb KVM<sup>2</sup>.

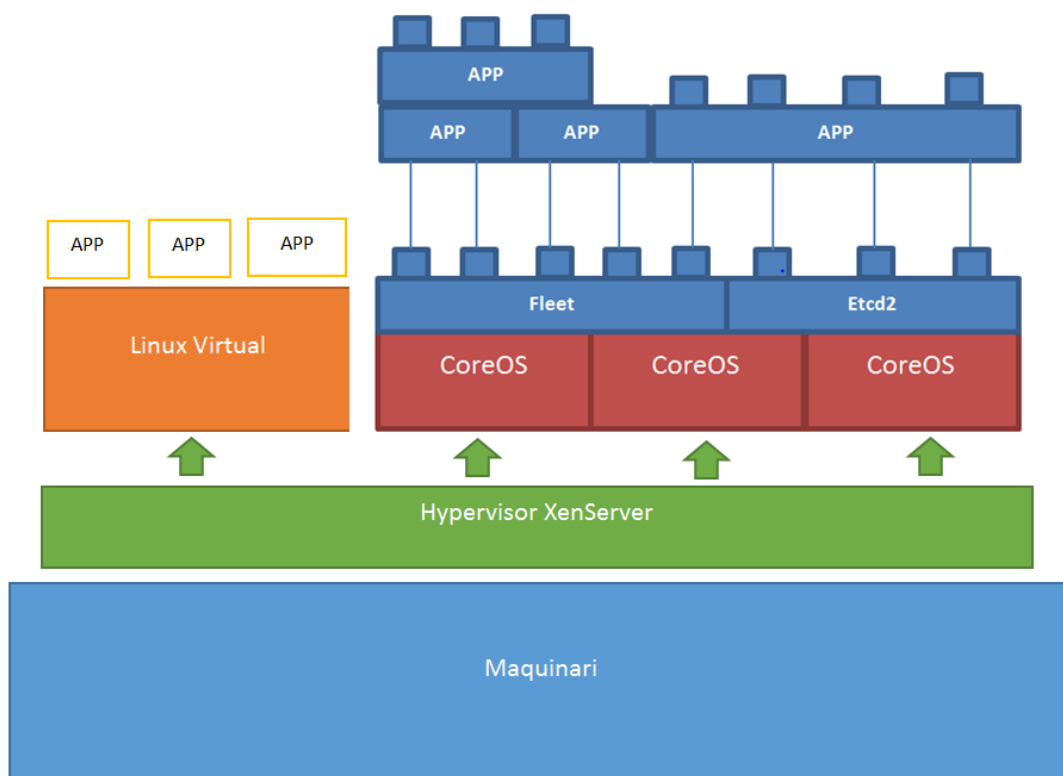
---

<sup>1</sup> LXC: Tecnologia de virtualització a nivell de sistema operatiu per a Linux.

<sup>2</sup> KVM: Virtualització completa basada en el nucli de Linux, és software lliure en la seva totalitat.

El maquinari que s'ha fet servir és un servidor Dell PowerEdge 2950, bàsicament és un servidor amb un processador Xeon 5000, 4 gigues de Ram i dos discos Scsi de 300 GB. Les especificacions del servidor les podeu trobar a [l'Annex X](#). El sistema té configurat un RAID 1, mirroring, en el dos discos i una petició per a tot el sistema operatiu XenServer.

Amb aquesta configuració, un servidor Dell i XenServer es pot crear les dues arquitectures necessàries per a realitzar les activitats esmentades al pla de treball. A la següent imatge es pot veure un esquema de la infraestructura de proves.



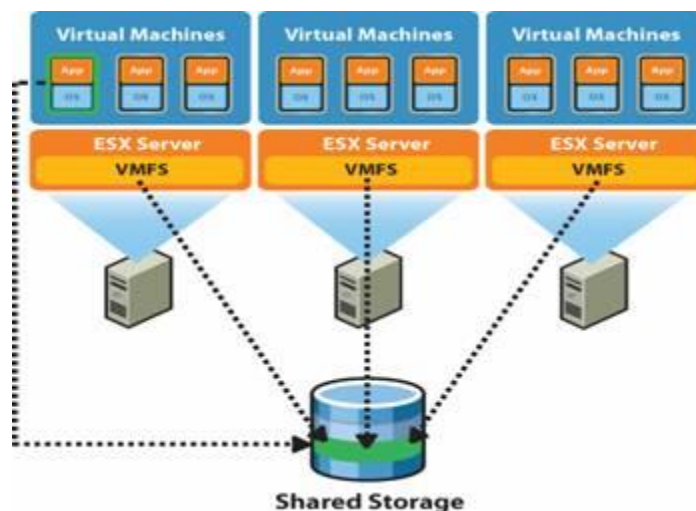
Il·lustració 4. Infraestructura del projecte – Font (elaboració pròpia)

## 9. DOCKER

Els sistemes virtualitzats van irrompre a les empreses, universitats, administracions públiques, etc... com una manera d'aprofitar els recursos de maquinari, podent implementar més d'un servidor a la mateixa màquina.

Després va arribar la necessitat de replicar aquestes màquines virtuals per evitar que una caiguda del servidor físic deixés a l'empresa o entitat sense tots els servidors que tenien corrent a dins.

Van sorgir solucions que actualment es fan servir com l'emmagatzematge de les màquines virtuals a sistemes redundats de dades (NFS, SAN, etc...) i fent que més d'un servidor físic fos capaç, mitjançant un programari, balancejar aquestes màquines virtuals entre els dos o més nodes del clúster.



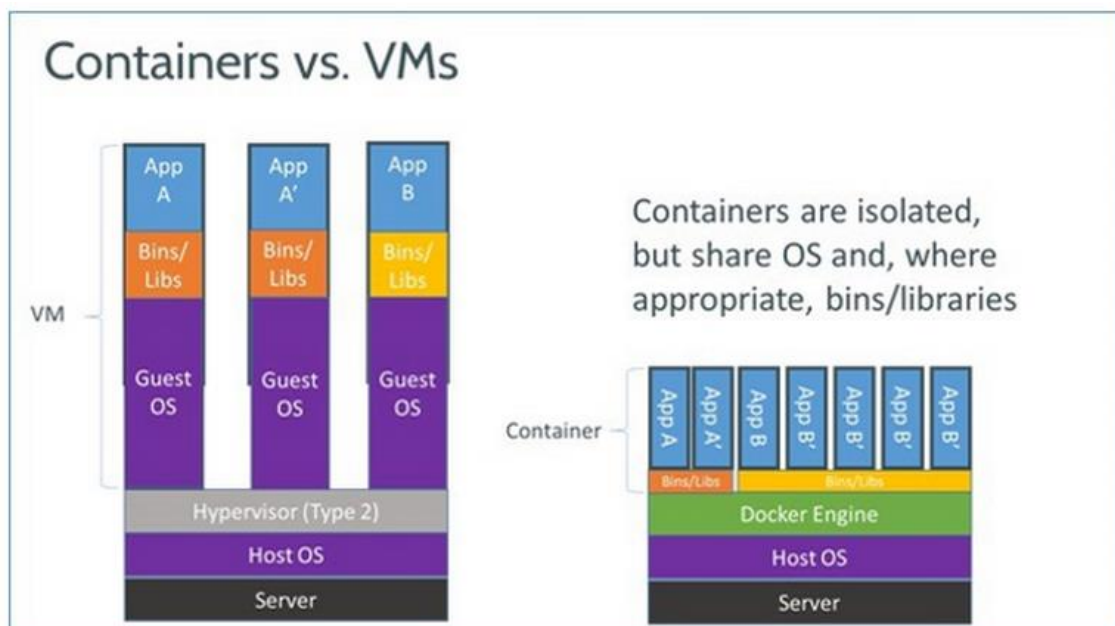
Il·lustració 5. Clúster VmWare - Font [www.vmware.com](http://www.vmware.com)

Les versions de Vmware que fan possible aquests balancejos de càrrega i que donen la possibilitat de moure les màquines virtuals entre els nodes del clúster són versions de pagament i no tenen un cost molt econòmic.

Tampoc vam tenir en compte el cost que té engegar una nova aplicació amb aquesta tecnologia, posant un exemple, si volem publicar una pàgina web a l'empresa necessitem instal·lar un servidor web, ja sigui apache, tomcat o IIS. Es necessari instal·lar un sistema operatiu, amb tots els seus serveis, actualitzacions i configuracions de seguretat que necessita, còpies de seguretat, rèpliques, manteniments necessaris i espais en disc, tant la màquina virtual com les còpies.

Docker permet executar una aplicació fent servir només el necessari per a que funcioni, sense necessitar un sistema operatiu per sota, totes les aplicacions de Docker comparteixen un mateix kernel de Linux. D'aquesta manera estalviem en espai, recursos de maquinari i manteniment dels sistemes operatius.

D'altra banda una aplicació Docker és portable, no és necessari que al lloc on la volem moure tingui la mateixa infraestructura on la vam desplegar. Actualment Docker esta present en les companyies més grans de Cloud que existeixen.



Il·lustració 6. VM vs Docker - Font [www.zdnet.com](http://www.zdnet.com)

## 10. COREOS

CoreOs és un sistema operatiu basat en el kernel Linux que ocupa molt poc espai i és molt lleuger. Està dissenyat per treballar en clúster<sup>1</sup>. És un sistema tant reduït que no té cap gestor de paquets per instal·lar aplicacions; totes les aplicacions han de ser executades dins dels contenidors gràcies a Docker<sup>2</sup>.

CoreOs està compost per vàries capes que permeten descobrir altres nodes a la xarxa i gestionar de manera eficient el volum de càrrega dels diferents components del clúster i dotar les aplicacions arrencades als Dockers d'alta disponibilitat.

La capa que corre per sobre de CoreOS és diu etcd, un dimoni<sup>3</sup> que gestiona gràcies al systemd els etc de diferent CoreOs per fer d'ells un clúster de servidors on podran córrer les aplicacions en contenidors. Etcd descobreix els nodes de la xarxa i gestiona el clúster. Aquest programa està escrit en GO, un llenguatge de programació molt semblant a C++ creat per Google.

Per descobrir els nodes fa servir un identificador de clúster que s'ha de generar posant la mida del clúster que es vol crear a la URL <https://discovery.etcd.io/new?size=3>, on 3 és el número de nodes que volem al clúster. Aquest servei ens retorna un identificador únic, el que he de configurar al fitxer d'arrencada del sistema operatiu CoreOS.

Per sobre de la capa etcd tenim fleet, aquest és un altre dimoni que ens permet tenir alta disponibilitat en les aplicacions que fem córrer. Aquest dimoni quan detecta que un node del clúster cau engega l'aplicació en un altre node. Per sobre de fleet tenim les aplicacions que s'executen gràcies a Docker el qual està incorporat al sistema CoreOS. Aquest programa també està escrit en GO.

---

<sup>1</sup> **CLÚSTER:** el significat de clúster a les TI és un conjunt d'ordinadors amb mateix hardware que es comporten com un sol ordinador.

<sup>2</sup> **DOCKER:** és un projecte de codi obert que permet desplegar aplicacions dins d'un contenidor de software evitant temps d'arrencada i manteniment de màquines virtuals, fa servir els kernel de la màquina host.

<sup>3</sup> **DIMONI;** a les TI un dimoni és un programa resident, un servei que s'executa junt al sistema operatiu, s'executa en segon pla.

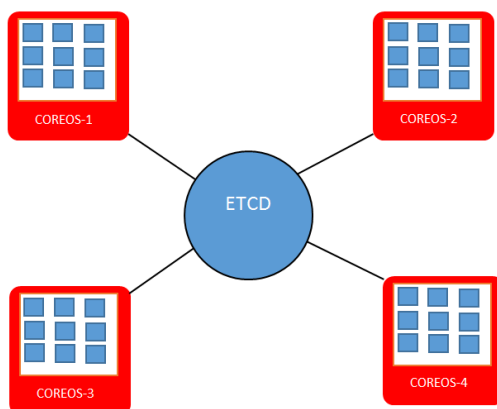
## 10.1 ETCD

Etcd és dimoni que s'executa a tots els nodes del clúster. Aquest dimoni és l'encarregat de replicar i distribuir les dades de configuració, que estan en JSON<sup>1</sup>, per tots els membres del clúster. Etcd fa servir una gestió del node mestre automatitzada, el node testimoni de node mestre va canviant a mesura de que el protocol ho determina, d'aquesta manera evitem que si el node mestre caigués el clúster deixés de funcionar.

D'una banda Etcd incorpora un servidor d'identificació de nodes del clúster. Mitjançant una URL podem sol·licitar un identificador únic per configurar l'arrencada de cada un dels membres del clúster. Els sistemes en arrencar es donaran d'alta a la URL amb l'identificador únic i el servidor els registrarà i enviarà les dades d'altres membres registrats anteriorment. Un cop estiguin tots els membres del clúster registrats i tinguin la configuració de la resta de membres les comunicacions, replicacions i distribucions entre ells es faran mitjançant la xarxa interna.

D'altra banda, etcd, a part de servir per descobrir nodes del clúster i replicar les configuracions, també ens permet un servei de descoberta de serveis, on els mateixos serveis poden anunciar-se a si mateixos. Quan creem un servei el qual es publica a un port en concret del nostre node, he de crear un servei d'auto-descoberta per anunciar al món en quin port està publicat el servei.

Podem trobar dues versions d'etcd, etcd a les versions estables i beta de CoreOS i etcd2 a les versions alpha.



Il·lustració 7. Funcionament de ETCD – Font (elaboració pròpia)

---

<sup>1</sup> JSON: acrònim de Java Script Object Notation, és un llenguatge que s'utilitza per intercanviar dades, actualment està substituït a XML.

## 10.2 FLEET

Fleet és un dimoni que s'encarrega de gestionar la càrrega de treball del diferents nodes del clúster, així com detectar possibles fallades tant als nodes com a les aplicacions que estan corrent a dins dels contenidors, és un `systemd`<sup>1</sup> distribuït que treballa a nivell de clúster en comptes de a nivell local com un `systemd`.

Aquest dimoni canvia els contenidors de node si detecta que un dels nodes ha deixat de funcionar. Per dur a terme aquesta tasca fleet engega els serveis que han deixat de funcionar a la resta dels nodes en funcionament repartint els serveis a engegar entre tots ells i d'aquesta manera balanceja la càrrega de treball.

Per poder engegar i aturar serveis, fleet fa servir Docker per engegar i aturar les imatges de les aplicacions que necessitem desplegar al sistema. Per fer ús de fleet s'ha de crear un fitxer on es configurarà el comportament del dimoni en cada moment de l'execució del servei. Podeu veure exemples del fitxer a l'Annex 2.

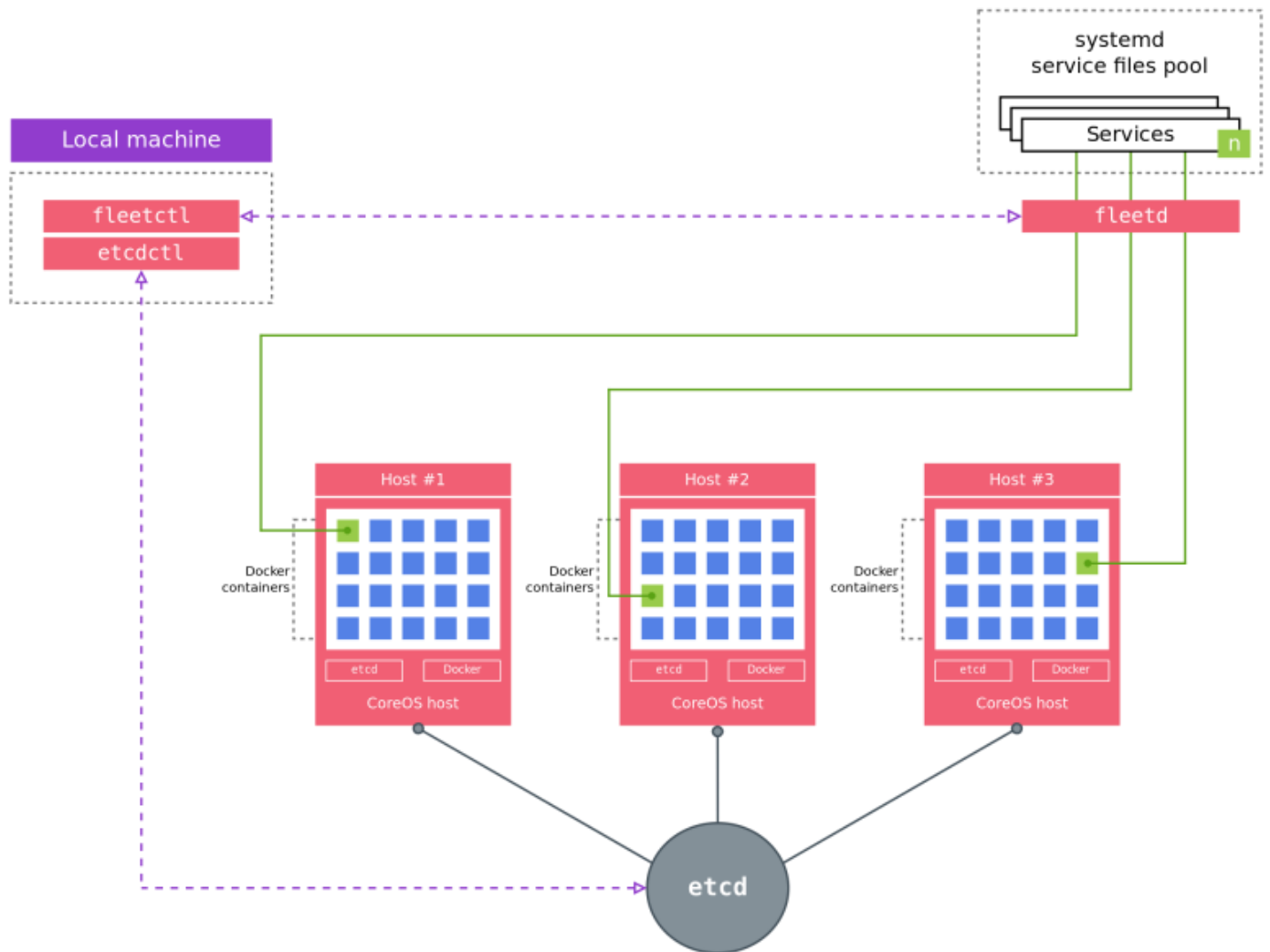
Mitjançant una API executem comandes a fleet per engegar, aturar, saber com estan els serveis engegats etc... l'API s'executa a través de la comanda `fleetctl` on alguns dels flags que podem utilitzar són: `start`, per engegar serveis, `destroy` per destruir els serveis, `list-machines` per veure quines màquines són part del clúster on fleet farà balanceig de càrrega i una de les més importants és `list-units`, aquest flag mostrarà tots els serveis que tenim engegats amb fleet en tots els nodes.

A la il·lustració 8 podem veure una arquitectura completa de CoreOS. Veiem com `systemd` amb la *pool* de serveis engegats a tots els nodes de clúster i gestionats per `fleetd`, que és el dimoni de fleet. També podem veure l'API local tan de fleet que és `fleetctl` com de `etcd` que és `etcdctl`.

---

<sup>1</sup> **SYSTEMD**: conjunt de dimonis d'administració de sistema, bloc de construcció bàsic per un sistema operatiu, s'utilitza com el sistema d'inici de Linux, inicialitza l'espai d'usuari durant el procés d'arrencada i fa la gestió dels processos iniciats al sistema.

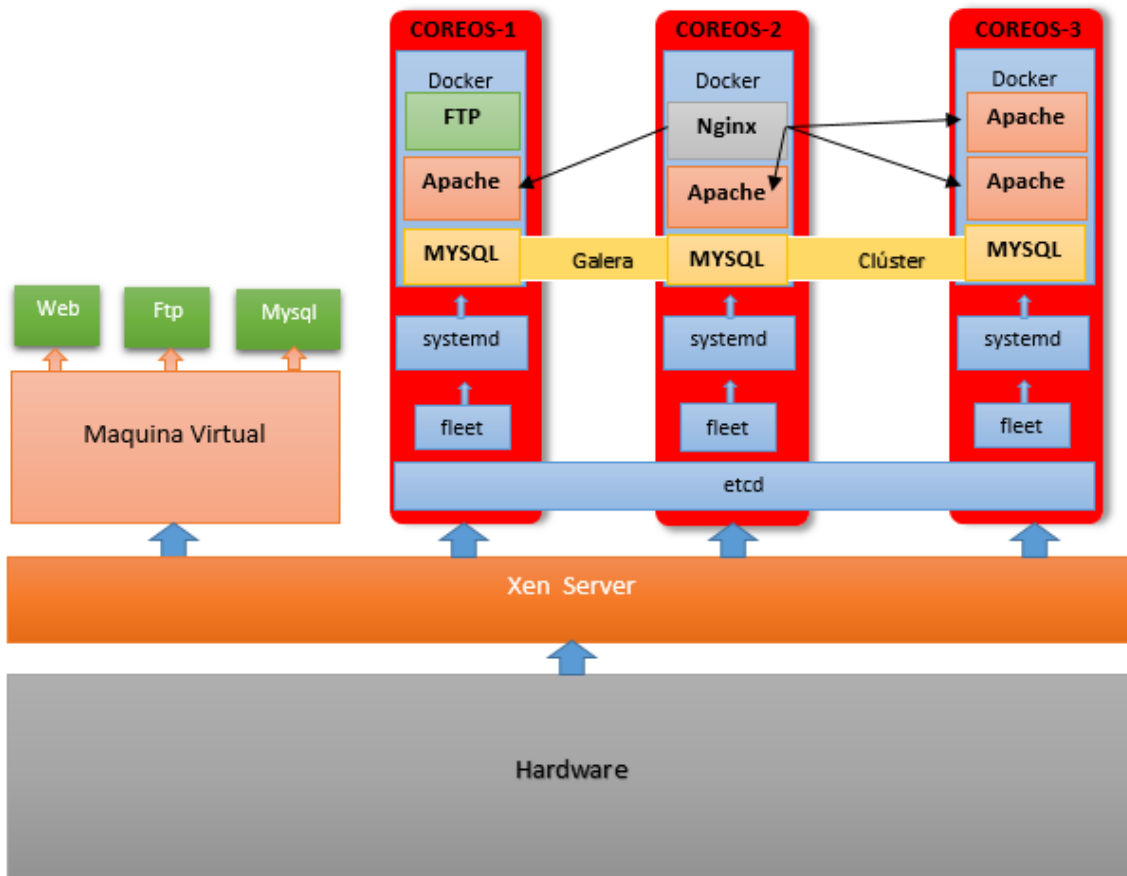




Il·lustració 8, Arquitectura completa COREOS- Font (Wikipedia)

### 10.3 CLÚSTER COREOS

Com podem veure a la il·lustració següent, on es veu tota la infraestructura que es fa servir per dur a terme aquest projecte, el clúster CoreOS és compon de tants nodes com necessitem. Cada un d'ells té implementat una sèries de dimonis per poder engegar i gestionar els contenidors.



Il·lustració 9, Disseny de la infraestructura sencera.

CoreOS fa servir, com ja s'ha comentat anteriorment, Docker per suports a contenidors, etcd per connectar tots els systemd dels CoreOs que formen el clúster, i per sobre tenim fleet que és l'encarregat de gestionar les unitats (contenidors) dins del clúster balancejant i donant alta disponibilitats a les unitats, serveis o contenidors segons vulguem anomenar-los.

Per configurar un clúster simplement s'ha de generar un identificador únic de clúster mitjançant un servei web que proporciona etcd, i configurar els diferents Cloud-config de cada node que vulguem afegir al clúster, tenint en compte que he d'informar del número de nodes que tindrem al nostre clúster en el moment que generem l'identificador únic de clúster.

La URL del servei és `https://discovery.etcd.io/new?size=3` on el nombre 3 és la quantitat de nodes que tindrà el clúster. Aquest servei farà que totes les instàncies de etcd que cada node estiguin connectades entre si, un exemple d'identificador únic és :

`https://discovery.etcd.io/b39feff5c935de2fbbd9235ac12d6dde`

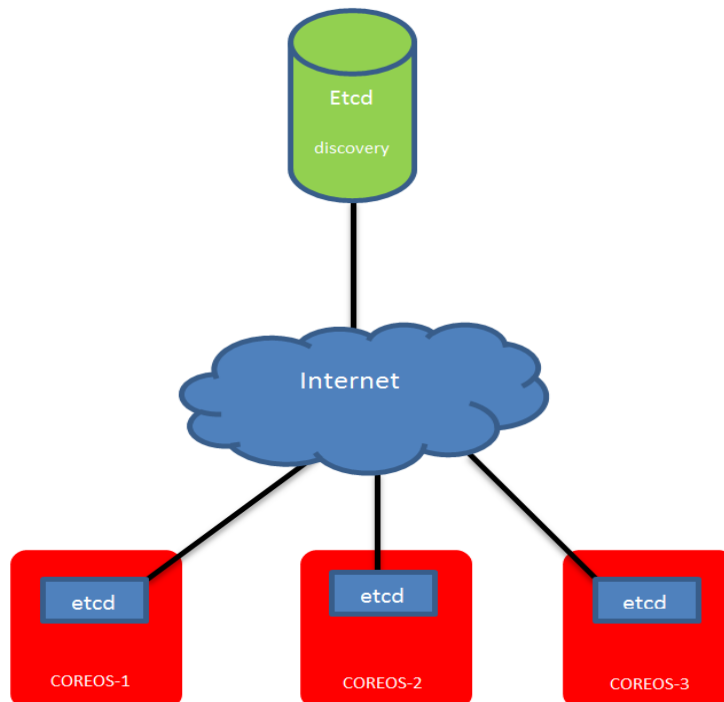
La seqüència d'inici del clúster és la següent:

1. El node 1 inicia i carrega el seu Cloud-config, pregunta al discovery si hi ha algun node al clúster i descarrega la configuració i la llista de nodes, com veu que és l'únic, es configura com a líder.
2. El node 2 arrenca i carrega el seu Cloud-Config, pregunta al discovery i descarrega la configuració i la llista de nodes, com veu que hi ha el node 1 és configura com a seguidor.
3. El node 3 arrenca i carrega el seu Cloud-Config, pregunta al discovery, descarrega la configuració i la llista de nodes del clúster, veu que hi ha el node 1 i el 2 i es configura com a seguidor.

Ja tenim el clúster engegat els etcd connectats i el node 1 fent de líder i el 2 i el 3 de seguidors. Tots el nodes tenen configurats el mateix identificador del discovery als seus Cloud-Config. El node 3 només tenint una ip de la resta de nodes del clúster ja es podria connectar a ell. Cada node conté la llista de nodes vius al clúster.

Un cop tenim un clúster funcionant no podem canviar l'identificador del clúster ja que no funcionaria, discovery està pensat per treballar amb el mateix identificador. Si un node ha deixat de funcionar per culpa del maquinari o del programari podem retirar el node del clúster fent servir l'API del etcd, `etcdctl` amb els flags de "member remove" i informant de l'identificador del clúster a eliminar.

Si volem afegir un nou node cal fer servir també l'API però amb el flag "member add", etcd ens informarà dels paràmetres que s'han de configurar al fitxer Cloud-Config del nou membre i un cop arrenqui el sistema es connectarà al clúster i es posarà al dia de la llista de màquines que el componen, dels serveis publicats, etc...



Il·lustració 10. Servei etcd Discovery - Font (elaboració pròpia)

#### 10.4 ESCALABILITAT DE LA INFRAESTRUCTURA

Els clústers CoreOS són unes infraestructures molt escalables, sempre s'ha de tenir en compte el consum de xarxa i cpu que necessita el clúster per replicar les dades entre tots els nodes que el formen, ja que si fem un clúster de massa nodes el guany que tindrem en concepte de tolerància a fallades, el perdrem en rendiment.

La mida del clúster recomanat és de 3, 5 o 7 nodes. Amb un clúster de 3 nodes tindrem una tolerància a fallades d'un node mentre que amb un clúster de 5 nodes podran fallar dos nodes i amb un de 7 nodes podem tenir 3 nodes sense funcionar que la resta aguantaran la càrrega del clúster.

A la següent taula podem veure la tolerància a fallades dels clústers segons els nodes que els componen. Veiem que és un gran esforç intentar tenir sempre un clúster imparell ja que sempre tindrem un node més de tolerància a fallades que si ens quedem en un node menys i parell, per exemple, si tenim un clúster de 3 nodes és molt millor fer l'esforç per intentar fer un clúster de 5 nodes que no quedar-nos en 4, perquè amb 4 continuarem tenint un sol node de tolerància a fallades, en canvi amb 5 tindrem fins a 2 nodes que poden fallar i el clúster continuaria garantint l'operativitat del sistema.

Taula 2. Tolerància a fallades clúster CoreOS

Mida del Clúster	Nodes en Majoria	Tolerància a Fallades
1	1	0
3	2	1
4	3	1
5	3	2
6	4	2
7	4	3
8	5	3
9	5	4

La columna tolerància a fallades són els nodes que poden deixar de funcionar garantint la integritat del clúster.

Com s'ha dit al 'apartat anterior, per crear un clúster, etcd té un servei de descoberta de nodes online. Això funciona obtenint un identificador de clúster únic per a la quantitat de nodes que vols crear el clúster i configurant-ho al fitxer d'arrencada de CoreOS anomenat Cloud-config.

El primer node que inicia el sistema es connecta a la pàgina del discovery d'etcd amb l'identificador configurat i es descarrega la llista de nodes que han connectat. Com que encara no hi ha cap node connectat es posa com a líder del clúster i espera als altres nodes. El segon node és connecta a la mateixa pàgina i descarrega la llista de nodes. Com que veu que existeix un node líder ho emmagatzema i es connecta amb ell per la xarxa local per sincronitzar les dades i així ho fan la resta de nodes del clúster.

Un cop tots els nodes s'han connectat i descarregat la llista del altres nodes que formen el clúster tots es connecten mitjançant la xarxa local i no fan servir el servei de descoberta d'etcd. Aquests membres no podran formar part d'un altre clúster i si s'ha d'escalar el clúster cal fer servir l'API d'etcd, ja sigui mitjançant json i http o la comanda etcdctl que ens facilita la vida.

10.4.1 Per eliminar un node del clúster s'executa la comanda `etcdctl` per saber la llista de membres del clúster i identificar l'id del que volem eliminar.

```
$ etcdctl member list
6e3bd23ae5f1ea: name=COREOS-01 peerURLs=http://localhost:23802
clientURLs=http://127.0.0.1:23792
924e2e83e93f25: name=COREOS-02 peerURLs=http://localhost:23803
clientURLs=http://127.0.0.1:23793
a8266ecf031671: name=COREOS-03 peerURLs=http://localhost:23801
clientURLs=http://127.0.0.1:23791
```

Un cop sabem l'id del node a eliminar del clúster executem la comanda `etcdctl member remove` per procedir a la seva eliminació.

```
$ etcdctl member remove 6e3bd23ae5f1ea
Removed member 6e3bd23ae5f1ea from cluster
```

- 10.4.2 Per afegir un nou membre al clúster i canviar la mida del clúster s'ha de fer:
1. Afegir un nou membre al clúster mitjançant l'API `etcd` o mitjançant la comanda `#etcdctl member add`
  2. Engegar el nou membre amb la nova configuració del clúster afegint els membres que ja existeixen al clúster.

Fent servir `etcdctl` anem a afegir el nou membre indicant la URL dels nodes anunciats.

```
$ etcdctl member add COREOS-04 http://10.50.0.13:2380
added member 8dfds9df90sdf to cluster

ETCD_NAME="COREOS-04"
ETCD_INITIAL_CLUSTER="COREOS-01=http://10.50.0.109:2380,COREOS-02=http://10.50.0.18:2380,COREOS-03=http://10.50.0.113:2380,COREOS-04=http://10.0.1.13:2380"
ETCD_INITIAL_CLUSTER_STATE=existing
```

`Etcdctl` ha informat del nou membre a la resta de membres del clúster i ha donat les variables d'entorn que s'ha de configurar al nou node abans d'iniciar-ho perquè s'afegeixi al clúster.

S'ha de configurar el Cloud-config del nou node amb les següents variables d'entorn:

```
$ export ETCD_NAME="COREOS-04"
$ export ETCD_INITIAL_CLUSTER="COREOS-01=http://10.50.0.109:2380,COREOS-02=http://10.50.0.18:2380,COREOS-03=http://10.50.0.113:2380,COREOS-04=http://10.50.0.13:2380"
$ export ETCD_INITIAL_CLUSTER_STATE=existing
$ etcd --listen-client-urls http://10.50.0.13:2379 --advertise-client-urls http://10.50.0.13:2379 --listen-peer-urls http://10.50.0.13:2380 --initial-advertise-peer-urls http://10.50.0.13:2380 --data-dir %data_dir%
```

El nou node s'iniciarà i formarà part del clúster i començarà a posar-se al dia amb la resta de membres. Si es vol afegir més d'un node és aconsellable fer-ho d'un en un, un cop el primer node afegit ha engegat i adherit es fa un nou node.

10.4.3 Per actualitzar un ip d'un membre o node del clúster cal fer servir l'API etcd o la comanda etcdctl per saber el llistat de nodes existent al clúster i localitzar el node que es vol actualitzar i el seu id.

```
$ etcdctl member list
6e3bd23ae5f1ea: name=COREOS-01 peerURLs=http://localhost:23802
clientURLs=http://127.0.0.1:23792
924e2e83e93f25: name=COREOS-02 peerURLs=http://localhost:23803
clientURLs=http://127.0.0.1:23793
a8266ecf031671: name=COREOS-03 peerURLs=http://localhost:23801
clientURLs=http://127.0.0.1:23791
```

Un cop se sap el seu id es pot procedir a actualitzar la seva ip, port o el que es necessita actualitzar.

```
$ etcdctl member update a8266ecf031671 http://10.50.0.10:2380
Updated member with ID a8266ecf031671 in cluster
```

## 10.5 INSTAL·LACIÓ DE SERVEIS

La infraestructura actual de l'empresa dona serveis al client relacionats amb les pàgines web. Aquesta empresa té la possibilitat d'allotjament d'una pàgina web estàtica, per exemple amb html, on el client només necessita un servidor web, en el nostre cas apache, i un servidor ftp per poder pujar el codi de la pàgina al seu espai d'usuari.

Altres clients tenen pàgines web dinàmiques fetes amb altres llenguatges de programació. Aquests llenguatges són suportats pel nostre servidor apache standard per a tots els clients, com ara php, perl, etc... aquests tipus de clients necessiten un servidor de base de dades, com mysql a més del servidor web i ftp.

### 10.5.1 SERVEI WEB

La diferència més interessant de la infraestructura amb contenidors és que cada unitat (servei) de servidor web, per exemple, corre una instància per a cada usuari. Si aquest usuari rep un atac de DOS al seu lloc web, el seu servidor web deixarà de funcionar però no la resta de servidors apache dels usuaris. En una infraestructura de màquina virtual un servidor apache és compartit per un nombre molt alt d'usuaris i si aquest deixa de funcionar, molts usuaris veurien afectat el seu servei.

Aquesta manera d'implementar apache és una manera de garantir un servidor apache per a cada usuari però, què passa si un usuari té una quantitat de visites molt reduïda i uns altres tenen moltes visites?, doncs el que podria passar és que un node del clúster estaria sense càrrega si tingués molts servidors sense visites i un altre node estaria molt carregat de peticions si fleet hagués engegat en ell molts servidors amb moltes visites.

D'altra banda també existeix un altre problema. Com que cada node del clúster només té un port 80 cal iniciar els servidors apaches a un port diferent per a cada servidor. Com es pot veure a la figura, i associar al client amb la seva instància d'apache. Aquest manteniment és molt costós i innecessari si muntem la infraestructura de servidors web d'una altra forma.

És per aquest motiu que es decideix fer una implementació diferent. Es fa servir un servidor proxy que balancegi la càrrega entre els diferents servidors apache de tots els apaches del clúster. Hi ha exposat a l'exterior només el port 80 d'un dels nodes del clúster i aquest redirigeix les peticions internament cap a altres ports que els usuaris finals no coneixen i no estan exposat a l'exterior.

Per poder fer aquest tipus d'implementació es necessita que tots els contenidors apache tinguin un directori comú on estiguin ubicades les pàgines web dels usuaris, i un servei que descobreixi que hi ha un servidor apache nou i actualitzi els fitxers de configuració del servidor Proxy.

La manera d'aconseguir que tots els nodes tinguin una carpeta comú per emmagatzemar els fitxers de configuració i les web d'usuari es necessitarà un dispositiu d'emmagatzematge extern. A l'empresa treballaran amb cabines de discos amb fonts i controladores redundants, però al laboratori es fa servir un NAS Iomega StarCenter ix2 amb dos discos SATA de 500GB que formen un RAID1 (mirall) per garantir la integritat de les dades.





Il·lustració 11, Nas lomega StorCenter ix2. Font (<http://www.expertreviews.co.uk>)

S'ha de configurar un directori nfs (network file System). Es pot trobar com implementar-ho a l'annex 4 ( configuració de directori compartit NFS al NAS) i connectar-ho a tots els nodes del clúster mitjançant el cloud-init. Agreguem els paràmetres de connexió i quan els CoreOS arrenquin connectaran el directori extern a un directori intern que se li passa per paràmetre.

```
- name: mnt-nfs.mount
  command: restart
  enabled: true
  content: |
    [Unit]
    Description=NFS Directories
    Documentation=man:hier(7)
    DefaultDependencies=no
    Conflicts=umount.target
    Before=local-fs.target umount.target

    [Mount]
    What=10.50.0.200:/nfs/NFS
    Where=/mnt/nfs
    Type=nfs
    Options=rw,acl
```

Il·lustració 12, Configuració NFS CoreOS, Font (pròpia)

Es pot veure a la il·lustració 12 on es fa una unitat anomenada mnt-nfs.mount on es defineixen els paràmetres a configurar, la ip del servidor NAS que és 10.50.0.200 i la carpeta que comparteix /nfs/NFS, on la es vol muntar al nostre CoreOS a /mnt/nfs i el tipus i les opcions de muntatge.

Agregant aquestes línies al cloud-config es tindrà al clúster un directori comú per poder treballar de manera conjunta amb tots els serveis que es creen. Un cop engeguin els nodes aniran connectant el directori remot 10.50.0.200:/nfs/NFS al directori local /mnt/nfs.

```
core@coreos1 /mnt/nfs/nginx/conf.d $ mount |grep 10.50.0.200
10.50.0.200:/nfs/NFS on /mnt/nfs type nfs (rw,relatime,vers=3,rsize=16384,wsiz
e=16384,namlen=255,hard,proto=tcp,timeo=600,retrans=2,sec=sys,mountaddr=10.50.0.200,mountvers=3,mountport=637,mountproto=udp,local_lock=none,addr=10.50.0.200)
core@coreos1 /mnt/nfs/nginx/conf.d $
```

Il·lustració 13, Directori NFS muntat CoreOS1

Per una banda, tindrem els serveis apache que podrem iniciar, tants com el hardware ens permeti. Cada un estarà escoltant a un port diferent. Aquest comportament es crea al fitxer de servei fleet, indiquem que cada servidor tingui un port diferent i s'evita que si dos servidors inicien en un mateix node amb el mateix port falli l'inici del servei.

```
core@coreos1 /mnt/nfs/services/apache $ cat apache@.service
[Unit]
Description=My Apache Frontend
After=docker.service
Requires=docker.service
After=flanneld.service
Requires=flanneld.service

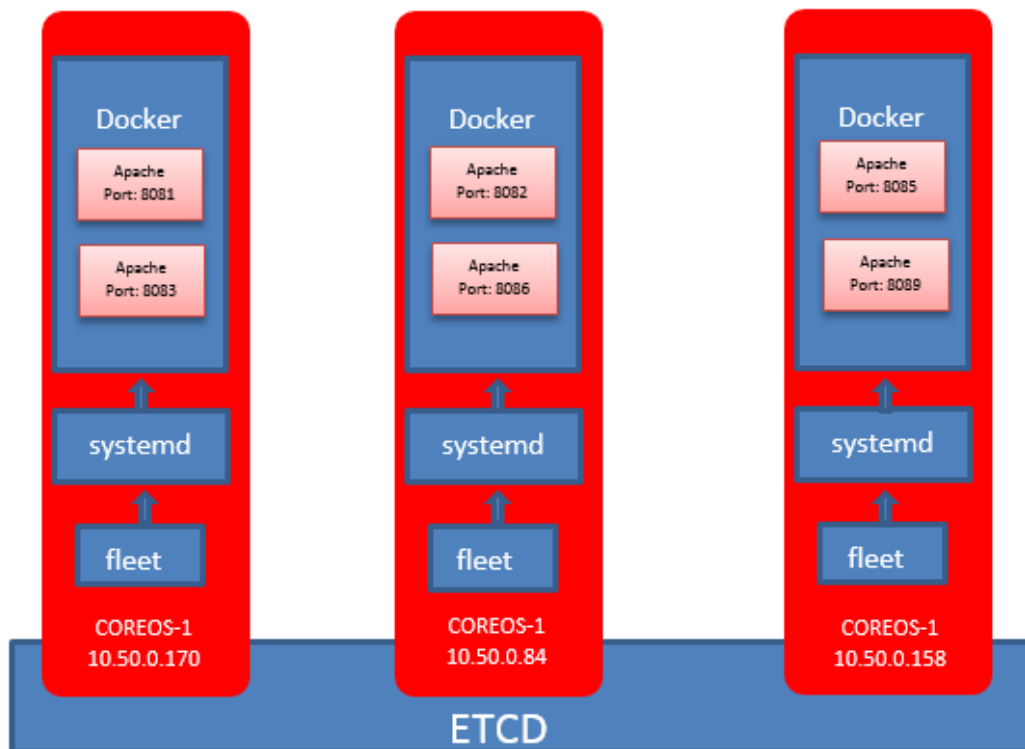
[Service]
TimeoutStartSec=0
ExecStartPre=/usr/bin/docker kill apache%i
ExecStartPre=/usr/bin/docker rm apache%i
ExecStartPre=/usr/bin/docker pull coreos/apache
ExecStart=/usr/bin/docker run --rm --name apache%i -v /mnt/nfs/www:/var/www:ro -p 808%i:80 coreos/apache /usr/sbin/apache2ctl -D FOREGROUND
ExecStop=/usr/bin/docker stop apache%i

core@coreos1 /mnt/nfs/services/apache $
```

Il·lustració 14, Servei Apache, Font (pròpia).

Podem veure a la il·lustració 13 que es crea una unitat anomenada My apache Frontend la qual necessita docker per ser executada i flannel per gestionar la xarxa. El servei en si primer intenta eliminar el contenidor del mateix nom per si existeix o s'ha quedat d'un altre inici penjat; després inicia un contenidor a la línia ExecStart que anomenem apache%i, amb el flag --name, on %i li passem per paràmetre a l'executar el servei amb fleet ( fleet start [apache@1.service](#)). En aquest cas el contenidor es dirà apache1.

Seguidament amb el flag `-v` connectem el nostre directori compartit del node al directori intern del contenidor on s'emmagatzemen les pàgines web d'usuari, si diem en quin port ha de publicar, en aquest cas 808%i, on %i li passem per paràmetre. El servei `apache1` publicarà pel port 8081 del node i el 80 del contenidor i l'`apache2` pel 8082 i així successivament.



Il·lustració 15, Contenedors Apache al Docker.

Ara ja tenim els serveis apache funcionant al servidor, en el cas de la il·lustració 15 tenim 2 apache per cada node del clúster. Si es volgués accedir al servidor apache de CoreOS-1 ho fariem pel port 8081 al primer i pel 8083 al segon, al navegador web o bé amb la comanda `curl http://10.50.0.170:8081`, per exemple.

Ara necessitem un servidor Proxy que ens apunti a tots aquests serveis que tenim iniciats i als que iniciarem si necessitem més servidors per un augment dels clients. Al laboratori s'ha fet servir *nginx*<sup>1</sup> però es podria haver fet servir d'altres, com per exemple *haproxy*. El servei `fleet` de `nginx` inicia el contenidor connectant els directoris de configuració interns de contenidor amb la carpeta compartida que tenim al servidor NFS.

```
core@coreos1 ~ $ cat /mnt/nfs/services/apache/nginx.service
[Unit]
Description=My Apache Frontend
Wants=doker.service network-online.target
After=docker.service network-online.target
Requires=docker.service
#After=flannel.service
#Requires=flannel.service

[Service]
TimeoutStartSec=0
ExecStartPre=/usr/bin/docker kill nginx
ExecStartPre=/usr/bin/docker rm nginx
ExecStartPre=/usr/bin/docker pull nginx
ExecStart=/usr/bin/docker run --rm --name nginx -p 80:80 -v /mnt/nfs/nginx/nginx.conf:/etc/nginx/nginx.conf:ro -v /mnt/nfs/nginx/conf.d:/etc/nginx/conf.d:ro nginx
ExecStop=/usr/bin/docker stop nginx
```

Il·lustració 16, Servei Proxy Nginx

A la figura 16 podem veure que iniciem un contenidor anomenat nginx i que estarà escoltant al port 80 del node on s'engegui, i que tindrà connectat al directori intern del contenidor /etc/nginx/conf.d el nostre directori del NAS /mnt/nfs/nginx/conf.d i que el fitxer /etc/nginx/nginx.conf serà el nostre document /mnt/nfs/nginx/nginx.conf. Els fitxers de configuració els podeu trobar conjuntament a la carpeta del projecte.

A les carpetes de fitxers de configuració tindrem un fitxer de configuració estàtic que es diu nginx.conf on configurem la ruta dels logs, les connexions concurrents, etc...

```
core@coreos1 /mnt/nfs/nginx $ cat nginx.conf
user nginx;
worker_processes 1;

error_log /var/log/nginx/error.log warn;
pid /var/run/nginx.pid;

events {
    worker_connections 1024;
}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
        '$status $body_bytes_sent "$http_referer" '
        '"$http_user_agent" "$http_x_forwarded_for"';

    access_log /var/log/nginx/access.log main;

    sendfile on;
    #tcp_nopush on;

    keepalive_timeout 65;

    #gzip on;

    include /etc/nginx/conf.d/*.conf;
}
core@coreos1 /mnt/nfs/nginx $
```

Il·lustració 17, Fitxer configuració nginx

---

<sup>1</sup> NGINX: servidor web i Proxy invers lleuger que s'encarrega de redirigir les peticions que li arriben al diferents servidors web que tenim a la xarxa interna.

I incloem la carpeta conf.d que és on tindrem el fitxer apache.conf on determinem els ports d'escolta i la localització dels servidors interns, la variable proxy\_pass del fitxer apache.conf ens apunta a un upstream que es diu clúster que serà el document, que també es troba a la carpeta conf.d que es diu upstream.conf que és el fitxer que anirem actualitzant quan un servidor apache inici i s'aturi.

```
core@coreos1 /mnt/nfs/nginx/conf.d $ cat apache.conf

server {

    #server_name;
    listen 80;

    location / {
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_pass http://cluster;
    }
}

core@coreos1 /mnt/nfs/nginx/conf.d $ █
```

Il·lustració 18, Fitxer apache.conf de Nginx

```
core@coreos1 /mnt/nfs/nginx/conf.d $ cat upstream.conf
upstream cluster {
    server 10.50.0.170:8081;
}

core@coreos1 /mnt/nfs/nginx/conf.d $ █
```

Il·lustració 19, Fitxer dinàmic upstream nginx

Com es pot veure a la il·lustració 18 el proxy nginx sap que té un servei apache a la ip 10.50.0.170 al port 8081. Aquesta configuració és dinàmica i es va actualitzant gràcies a un servei de fleet que he creat. Aquest servei l'engegarem per a cada servidor apache i anirà mirant si el contenidor està actiu i crea la línia “server x.x.x.x:port” al fitxer que es troba al directori /mnt/nfs/nginx/conf.d/ el qual està connectat al directori remot del nostre NAS.

El servei “discovery” de fleet fa servir un script quan inicia, li passem per paràmetre l'acció a fer, afegir o eliminar la ip i el port del servidor al qual esta associat.

```

core@coreos1 /mnt/nfs/services/apache $ cat apache-anunce\@.service
[Unit]
Description=Announce Apache1
BindsTo=apache@%i.service
After=apache@%i.service

[Service]
EnvironmentFile=/etc/environment
ExecStart=/bin/sh -c "while true; do /mnt/nfs/nginx/conf.d/upstream_script add ${COREOS_PUBLIC_IPV4}:808%i;sleep 45;done"
ExecStop=/bin/sh -c "su core /mnt/nfs/nginx/conf.d/upstream_script rm ${COREOS_PUBLIC_IPV4}:808%i"

[X-Fleet]
MachineOf=apache@%i.service
core@coreos1 /mnt/nfs/services/apache $ █

```

**Il·lustració 20, Servei Fleet Discovery Apache**

Com es pot veure a la il·lustració 20 que el servei apache-anunce va mirant cada 45 segons si el contenidor està engegat i actualitza el fitxer upstream.conf mitjançant l'script upstream\_script, fem servir la variable d'entornament (\$COREOS\_PUBLIC\_IPV4) per anunciar la màquina on està corrent el servei.

L'script upstream\_script mira si el servidor passat per paràmetre ja existeix al document i sinó hi és l'escriu. Quan el servei apache-anunce veu que el servidor apache al qual esta connectat no funciona crida a l'escript dient-li que esborri el servidor de la llista. Després de cada modificació del fitxer s'ha de fer una recarrega del servidor nginx perquè actualitzi la seva llista interna, com podem veure a la darrera línia de la il·lustració 19.

```

core@coreos1 /mnt/nfs/nginx/conf.d $ cat upstream_script
#Basat en l'script de Sergei Lomakov

#!/bin/bash
if [ -n "$1" -a -n "$2" ]; then
    action="$1";
    target="$2";
else
    echo "Usage: $0 (add|rm) server:port"
    exit 0;
fi;
# Path to upstream config file
CONF="/mnt/nfs/nginx/conf.d/upstream.conf"

SERVERS=`cat $CONF | grep server`

output="upstream cluster {"

if [ $action == "add" ]; then
    echo -e "$output" > $CONF
    if $( echo $SERVERS | grep --quiet $target ); then
        echo "Warning: Server is already enabled."
    else
        SERVERS="$SERVERS\n\tserver $target;"
    fi
    echo -e "$SERVERS" >> $CONF
    echo "}" >> $CONF
elif [ $action == "rm" ]; then
    sed -i "/$target/d" $CONF
else
    echo "Unknown action"
fi

# Actualitzem canvis:
ssh core@10.50.0.84 -i /home/core/id_rsa "sudo docker exec nginx service nginx reload "
core@coreos1 /mnt/nfs/nginx/conf.d $ █

```

Il·lustració 21, Script actualitzar upstream Nginx

A la il·lustració 21 es pot veure que la crida a l'script es fa passant per paràmetre add i x.x.x.x:port per afegir un servidor l'upstream i rm més x.x.x.x:port per eliminar un servidor de l'upstream. A la darrera línia podem veure que executem una comanda mitjançant ssh a un node del clúster. Això és possible perquè fem servir una clau privada per autenticar-nos al node, així executem la comanda de recàrrega de nginx quan el fitxer s'ha modificat.

La ip del servidor nginx també pot canviar, perquè si un node del clúster deixés de funcionar, fleet engegaria el contenidor amb nginx a un altre node del clúster i això suposaria que nginx tingués una ip diferent. Pel que fa al Firewall es farà un pool de ips privades, del rang del clúster, perquè siguin assignades dinàmicament.

Per solucionar aquest problema s'ha creat un servei de descoberta de nginx que simplement crida a un script passant-li per paràmetre la ip del servidor nginx i la modifica al fitxer `upstream_script`. L'script simplement treu la ip que hi ha al fitxer `upstream_script` i la canvia per la nova ip del servidor nginx.

```
core@coreos1 /mnt/nfs/services/apache $ cat nginx-anunce.service
[Unit]
Description=Announce Nginx
BindsTo=nginx.service
After=nginx.service

[Service]
EnvironmentFile=/etc/environment
ExecStart=/bin/sh -c "while true; do /mnt/nfs/nginx/conf.d/canvi_ip_nginx ${COREOS_PUBLIC_IPV4};sleep 45;done"
ExecStop=

[X-Fleet]
MachineOf=nginx.service
```

Il·lustració 22, Script descoberta Nginx `nginx-anunce.service`

A la il·lustració 22 veiem que el servei de descoberta que depèn del servei `nginx.service` fa servir l'script `canvi_ip_nginx` i li passem per paràmetre la ip del node de clúster on esta corrent el servei nginx.

```
core@coreos1 /mnt/nfs/nginx/conf.d $ cat canvi_ip_nginx
#Raul Rivero Ramos

#/bin/bash

if [ -n "$1" ]; then
    novaip="$1";
    # echo nova ip $novaip;

else
    echo "Usage: $0 ip "
    exit 0;
fi;

cerca=`cat /mnt/nfs/nginx/conf.d/upstream_script | grep -oiE '([0-9]{1,3}\.){3}[0-9]{1,3}'`;
#echo ip cercada $cerca;

if [ "$cerca" != "$1" ]; then
    sed -i "s/$cerca/$1/" /mnt/nfs/nginx/conf.d/upstream_script
    echo " S'ha canviat la ip $cerca per $1"

else

    echo " les ip són igual, no es fa cap canvi"

fi;
core@coreos1 /mnt/nfs/nginx/conf.d $ █
```

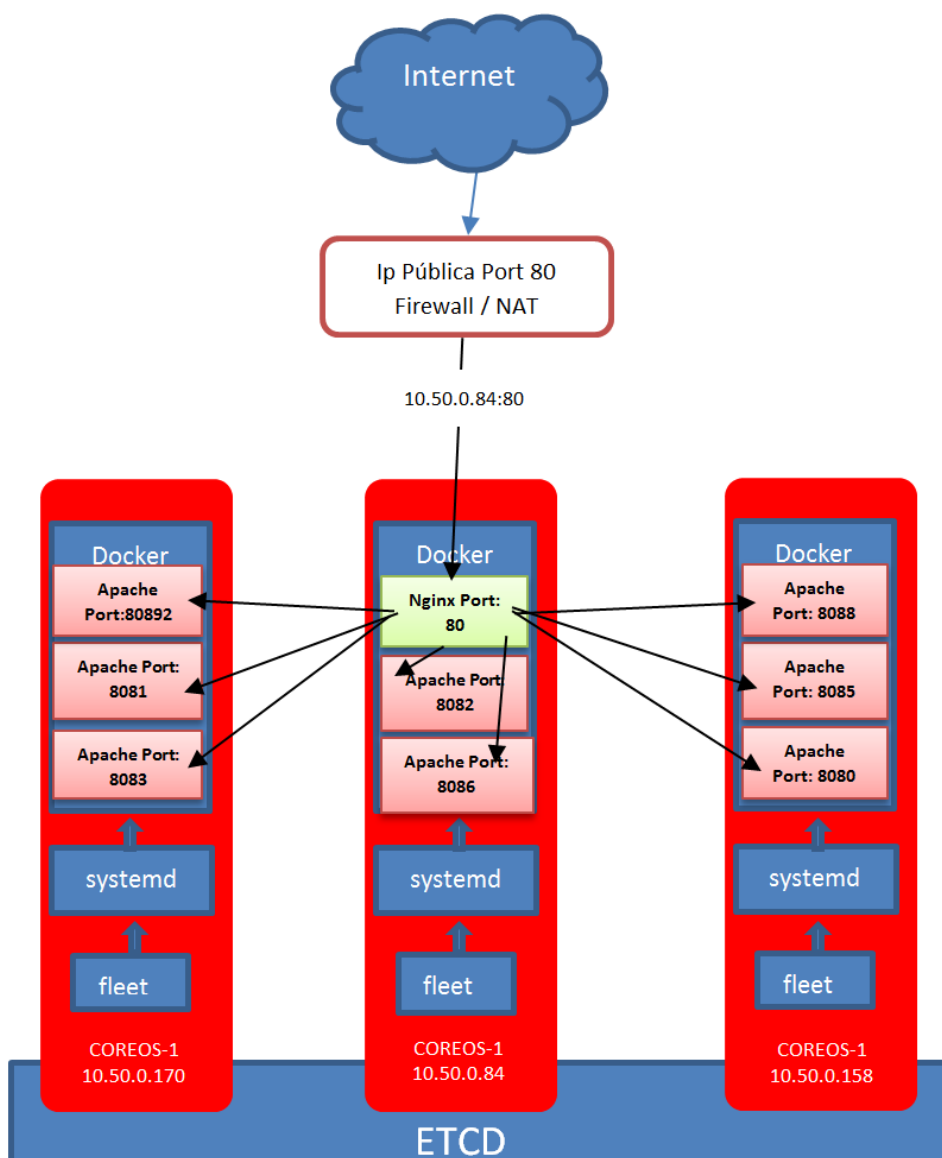
Il·lustració 23, Script Canvi ip nginx



Veiem a la il·lustració 23 l'script que canvia la ip del l'script que gestiona els servidors apache que estan funcionant al clúster. Aquest programa només mira si l'ús del programa és correcte i extreu la ip actual del fitxer upstream\_script per poder fer servir la comana sed i d'aquesta manera substituir-la per la nova ip que ens ha passat el servei de descoberta nginx-announce.service, comprova que les ips siguin diferents i si no ho són no fa cap canvi.

Com a resum de la solució tenim una quantitat, dimensionada amb els nombre de clients que tenim, de servidors apache els quals seran clons uns dels altres, tots els servidors comparteixen la carpeta www de totes les web dels clients, on cadascú publicarà per un port diferent a la ip del node on s'està executant.

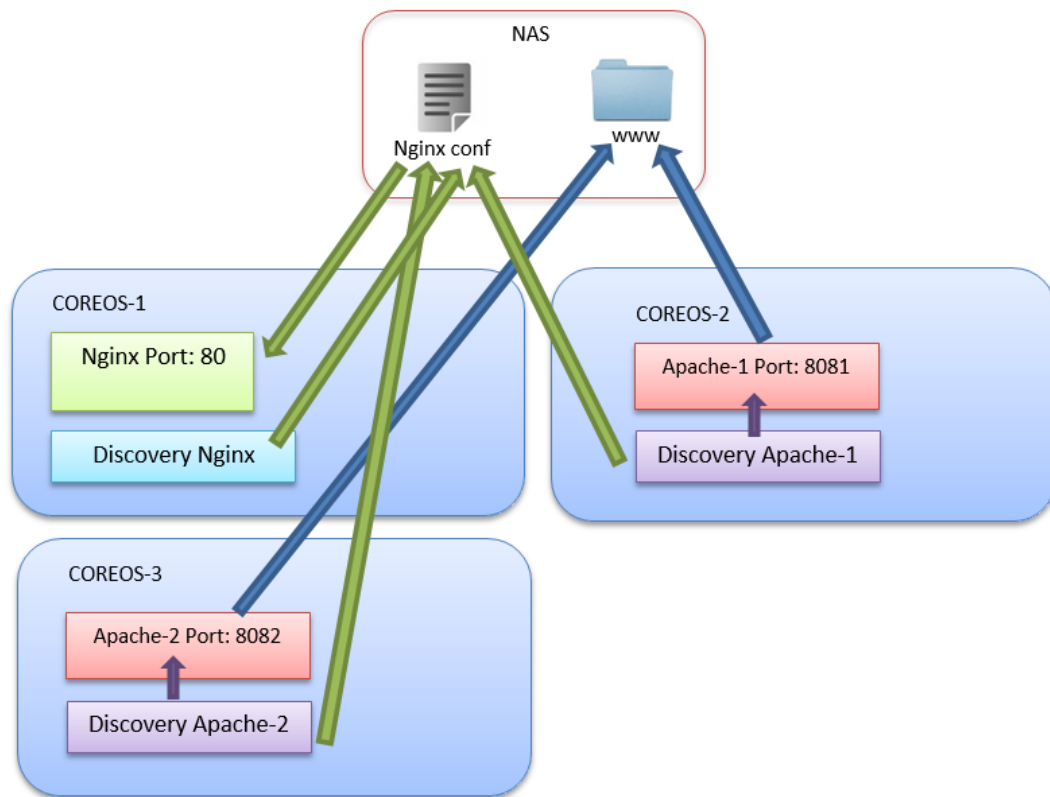
Tenim un servidor proxy-web que publica pel port 80 del node on s'està executant, Aquest servidor proxy obtindrà dinàmicament la llista de servidors actius al clúster mitjançant un servei de descoberta que fa servir un script per anar actualitzant el fitxer de configuració i la recàrrega del servidor proxy-web. Així mateix, si el servidor Proxy-web canvia de node un servei de descoberta actualitza la ip a l'script de manteniment de servidors apache.



Il·lustració 24, Infraestructura Web

De la il·lustració 24 destaquem el funcionament de la infraestructura web. Només tenim un port exposat a l'exterior, el Firewall fa NAT<sup>1</sup> dinàmicament amb una pool de ip privades assignades als nodes del clúster, cap al nostre servidor Proxy-web Nginx. Aquest, mitjançant els fitxers de configuració i els serveis de descoberta, balanceja la càrrega entre tots els servidors apache que tenim al clúster.

<sup>1</sup> NAT: procés que modifica la informació sobre les adreces a la capçalera IP per encaminar una petició a la ip pública de l'empresa cap a una ip privada.



Il·lustració 25, Vista Global de la Solució Web

A la il·lustració 25 podem veure una vista global de la solució, on veiem que els servidor web apache comparteixen la mateixa carpeta de web d'usuaris, on també veiem com els serveis de descoberta escriuen als fitxers de configuració del NAS on està escoltant el nostre servidor Proxy-web per anar actualitzant la llista de servidors disponibles al clúster.

### 10.5.2 SERVEI MYSQL

Pel que fa al servidor mysql, si seguíssim la mateixa filosofia que amb els servidors apache, una unitat fleet amb molts servidors mysql, la infraestructura no podria suportar la càrrega de memòria ram. Un servidor mysql té un consum de memòria ram molt elevat, tot i que podem parametritzar els paràmetres de configuració seria una decisió poc encertada.

Per fer-nos una idea del consum de memòria ram els paràmetres per defecte de mysql són:

Taula 3. Consum de memòria MySQL Font ( /etc/mysql/my.cnf)

Paràmetre	MySQL Default
key_buffer_size	64 MB
+ query_cache_size	64 MB
+ tmp_table_size	32 MB
+ innodb_buffer_pool_size	8 MB
+ innodb_additional_mem_pool_size	1 MB
+ innodb_log_buffer_size	1 MB
+ max_connections	150
*sort_buffer_size	2 MB
+ read_buffer_size	0.128 MB
+ read_rnd_buffer_size	0.256 MB
+ join_buffer_size	0.128 MB
+ thread_stack	0.196 MB
+ binlog_cache_size	0 MB
<b>Total Ram:</b>	<b>576.2 MB</b>

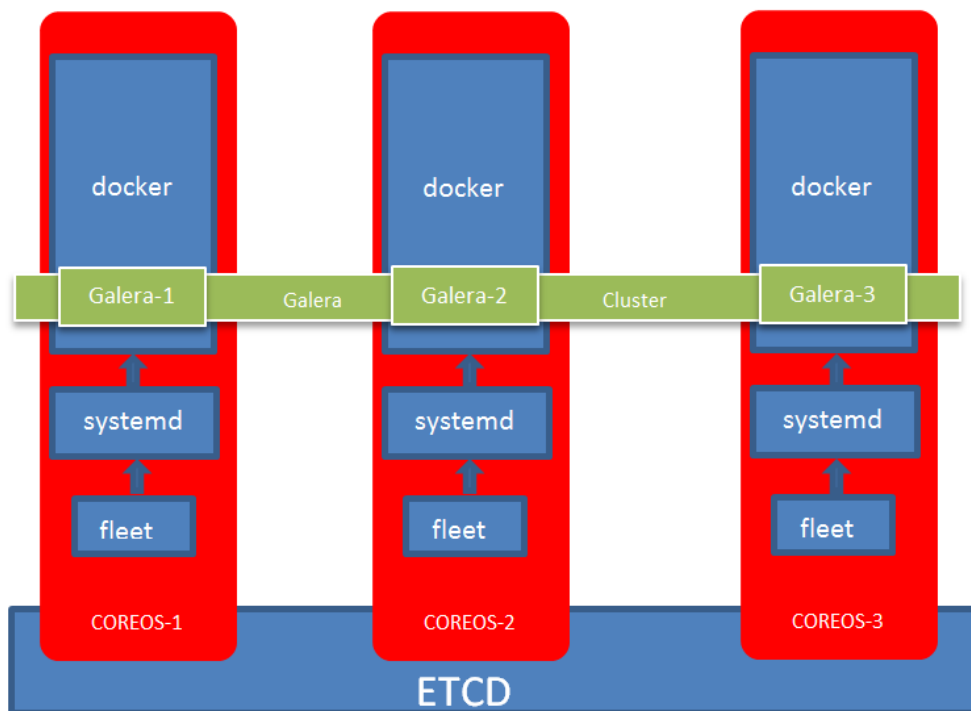
La quantitat de ram obtinguda és la suma de tots els MB més la multiplicació de les connexions màximes que ara tenim `150 x sort_buffer_size` que fa un total de 300 MB. Aquest paràmetres es poden configurar i podem reduir substancialment el consum de memòria del servidor mysql però mai seria convenient fer un servidor mysql per a cada usuari, seria un gran desaprofitament dels recursos de maquinari.

Per aconseguir les dades de configuració i crear aquesta taula és necessari accedir als fitxers de configuració del servidor mysql, el fitxer es troba ubicat al PATH `/etc/mysql/my.cnf` i aquest pot incloure altres fitxers inclosos al directori `/etc/mysql/conf.d/`

Una possibilitat és implementar un servidor mysql dins d'un contenidor docker i garantir la disponibilitat del servei mitjançant fleet. Si el node on el servidor mysql s'està executant caigués, fleet l'executaria en un altre node del clúster, però existeixen diferents possibilitats més elegants i eficients de fer-ho. És pot fer instal·lant dos contenidors amb mysql i un que d'ells sigui el màster i l'altre l'esclau. Amb aquesta solució tindrem replicats tots el canvis que fem al servidor màster, al servidor esclau però no a l'inrevés.

També per garantir la disponibilitat del servei podem implementar un clúster de servidors mysql, en concret 3 servidors, que seran contenidors que s'executaran al clúster CoreOS. Farem servir un clúster mysql anomenat Galera que mitjançant un proxy tindrem un únic port de connexió per a tots els usuaris i aquest proxy redistribuirà les connexions a través dels tres contenidors que executaran mysql. Aquests tres contenidors s'estaran replicant les bases de dades entre ells.

Amb aquesta solució tindrem replicats tots els canvis fets a les bases de dades i, el que és més important, farem un balanceig de càrrega de connexions entre els 3 servidors del clúster, alliberant de càrrega un sol servidor si optéssim per agafar un sol servidor mysql executant-se en un sol node del clúster.



Il·lustració 26. Clúster Galera Mysql- Font (Elaboració Pròpia)

Per instal·lar aquest servei s'han de crear els respectius serveis fleet i els serveis d'auto-descoberta d'etcd. Els serveis fleet executaran mitjançant systemd i docker una imatge del repositori docker creada per fer clúster mysql amb Galera anomenada Percona.

Al servei especifiquem el nom de la imatge que es farà servir com a clúster de mysql, el password del servidor mysql i els ports de comunicació. S'han utilitzat diferents imatges de docker per implementar el clúster, en concret 3, EntropyWorks, Paulczar i Zurmo. Finalment s'ha escollit Zurmo perquè utilitza un sol document fleet per a engegar tots tres nodes i ells mitjançant etcd es van trobant gràcies als serveis de descoberta adjunts a aquesta memòria.

Al dimoni fleet el que fem és el servei que hem creat d'apache, fem la unitat especificant el nom, els requeriments d'engegada, en aquest cas necessita docker.service i galera\_cluster\_discovery, que és el servei de descoberta dels nodes del clúster Galera.

Després configurem el servei pròpiament dit, compartim les variables d'entorn que hem implementat al fitxer cloud-config, i la línia més important és la línia que comença per ExecStart=. Aquest línia és la comanda que executem el contenidor i li passem per paràmetre a la imatge molta configuració del clúster.

Veiem que determinem el password de root del mysql, el nom de la base de dades, l'usuari de myql, el password de l'usuari, quina ip té el servei etcd i a quin port està escoltant, li passem la ip del node on s'està executant el contenidor, l'id del node al clúster, les variables WSREP que són variables de funcionament del clúster, els ports que obrim per escoltar les connexions tant dels clients de base de dades com els altres membres del clúster mysql.

```
core@coreos1 /mnt/nfs/services/galera/zurmo $ cat galera_cluster@.service
[Unit]
Description=MySQL Database
After=docker.service
Requires=docker.service
Before=galera_cluster_discovery@%i.service
#Before=zurmo_log_courier_galera_cluster@%i.service
Wants=galera_cluster_discovery@%i.service
#Wants=zurmo_log_courier_galera_cluster@%i.service

[Service]
EnvironmentFile=/etc/environment
TimeoutStartSec=0
KillMode=none
Restart=always
ExecStartPre=-/usr/bin/docker kill galera_cluster.%i
ExecStartPre=-/usr/bin/docker rm galera_cluster.%i
ExecStartPre=/usr/bin/docker pull icclab/cna/zurmo_galera_cluster
#: ${DOCKER_IMAGE_TAG}
ExecStart=/usr/bin/bash -c 'exec /usr/bin/docker run --name galera_cluster.%i -e "MYSQL_ROOT_PASSWORD=zurmo" -e "MYSQL_DATABASE=zurmo" -e "MYSQL_USER=zurmo" -e "MYSQL_PASSWORD=zurmo" -e "ETCD_ENDPOINT=${COREOS_PUBLIC_IPV4}:4001" -e "COREOS_PRIVATE_IPV4=${COREOS_PUBLIC_IPV4}" -e "GALERA_CLUSTER_NODE_ID=%i" -e "GALERA_CLUSTER=true" -e "WSREP_SST_USER=sst" -e "WSREP_SST_PASSWORD=zurmo" -e "INIT_ZURMO_DB=true" -p 3306:3306 -p 4444:4444 -p 4567:4567 -p 4568:4568 -p 9202:9200 -p 9201:9201 icclab/cna/zurmo_galera_cluster'
ExecStop=/usr/bin/docker stop galera_cluster.%i

[X-Fleet]
Conflicts=galera_cluster@*.service
core@coreos1 /mnt/nfs/services/galera/zurmo $
```

#### Il·lustració 27, Servei Galera Clúster

El servei de descoberta incorpora als directoris del servei etcd i Systemd els valors dels nodes que passaran a formar part del nou clúster, com ara, el uuid del sistema per no tenir dos nodes amb el mateix nom, la ip de la màquina on corre el contenidor docker, el nom del host, el port per on parlaran els diferents nodes, etc...

Aquest servei de descoberta està sempre mirant si el contenidor al qual està vinculat està funcionant i si és així escriu les dades als fitxers d'etcd i systemd. Si el contenidor de galera s'atura el servei esborra les dades del directori.

```
core@coreos1 /mnt/nfs/services/galera/zurmo $ cat galera_cluster_discovery@.service
[Unit]
Description=Announce mysql
BindsTo=galera_cluster@%i.service
After=galera_cluster@%i.service
Requires=galera_cluster@%i.service

[Service]
EnvironmentFile=/etc/environment
Restart=always
ExecStart=/bin/sh -c "ID=`uuidgen`; echo $ID > /tmp/galera_cluster@%i.uuid; TTL=10; SLEEP_TIME=7; etcdctl setdir /components/galera/$ID; etcdctl set /components/galera/$ID/service instance_name galera_cluster@%i.service; while true; do etcdctl updatedir /components/galera/$ID; etcdctl set /components/galera/$ID/host %H; etcdctl set /components/galera/$ID/port 3306; etcdctl set /components/galera/$ID/ip ${COREOS_PUBLIC_IPV4}; etcdctl set /services/database/galera/port 3306; etcdctl set /services/database/galera/ip ${COREOS_PUBLIC_IPV4}; sleep $SLEEP_TIME; done"
ExecStop=/usr/bin/etcdctl rm --recursive /components/galera bin/sh -c "etcdctl rm --recursive /components/galera `cat /tmp/galera_cluster@%i.uuid` && rm -f /tmp/galera_cluster@%i.uuid"

[X-Fleet]
MachineOf=galera_cluster@%i.service
core@coreos1 /mnt/nfs/services/galera/zurmo $
```

Il·lustració 28, Servei descoberta Galera

Un cop engegum un node, el primer es posa com a màster i espera als altres dos per sincronitzar els paràmetres. Per engegar el clúster hem d'iniciar un servei seguit del seu servei de descoberta per tal que quan un altre node engegui pugui consultar els directoris d'etcd i veure els nodes que estan actius al clúster CoreOS.

Per iniciar un servei fem servir fleet amb la comanda `fleet start galera_cluster@1.service`, on 1 és el valor que li passem al servei per crear el nom del node.

```
core@coreos1 /mnt/nfs/services/galera/zurmo $ fleetctl start galera_cluster@1.service
Unit galera_cluster@1.service inactive
Unit galera_cluster@1.service launched on a9e89f8b.../10.50.0.31
core@coreos1 /mnt/nfs/services/galera/zurmo $ fleetctl start galera_cluster_discovery@1.service
Unit galera_cluster_discovery@1.service inactive
Unit galera_cluster_discovery@1.service launched on a9e89f8b.../10.50.0.31
core@coreos1 /mnt/nfs/services/galera/zurmo $
```

Il·lustració 29, Comandes engegar servei

Com podem veure a la il·lustració 29 tant el servei mysql com el servei de descoberta s'inicien al mateix node perquè són dos serveis vinculats.



Mitjançant la comanda: `journalctl -feu galera_cluster@1.service` podem veure que està fent el servei. Aquesta comanda l'hem d'executar al node on està corrent el contenidor, en aquest cas al node amb ip 10.50.0.31 i podrem veure que s'ha posat com a Màster i està esperant als altres nodes.

```
Apr 07 21:22:57 coreos2 bash[2326]: version = 3,
Apr 07 21:22:57 coreos2 bash[2326]: component = PRIMARY,
Apr 07 21:22:57 coreos2 bash[2326]: conf_id = 0,
Apr 07 21:22:57 coreos2 bash[2326]: members = 1/1 (joined/total),
Apr 07 21:22:57 coreos2 bash[2326]: act_id = 0,
Apr 07 21:22:57 coreos2 bash[2326]: last_appl. = -1,
Apr 07 21:22:57 coreos2 bash[2326]: protocols = 0/7/3 (gcs/repl/appl),
Apr 07 21:22:57 coreos2 bash[2326]: group UUID = e60cb081-fd06-11e5-8142-1e6e13e5d4f2
```

```
Apr 07 21:24:45 coreos2 bash[2326]: + echo 'Waiting for cluster to reach size 3'
Apr 07 21:24:45 coreos2 bash[2326]: + sleep 5
Apr 07 21:24:45 coreos2 bash[2326]: Waiting for cluster to reach size 3
```

### Il·lustració 30, Primer node Galera

Si engegarem els altres nodes es connecten al primer i creen un clúster de servidors mysql. El node principal el descobreix, el connecta al clúster i li passa les dades de configuració necessaris, com ara l'identificador de clúster.

```
Apr 07 21:31:36 coreos2 bash[2326]: 2016-04-07 21:31:36 93 [Note] WSREP: Quorum results:
Apr 07 21:31:36 coreos2 bash[2326]: version = 3,
Apr 07 21:31:36 coreos2 bash[2326]: component = PRIMARY,
Apr 07 21:31:36 coreos2 bash[2326]: conf_id = 1,
Apr 07 21:31:36 coreos2 bash[2326]: members = 1/2 (joined/total),
Apr 07 21:31:36 coreos2 bash[2326]: act_id = 0,
Apr 07 21:31:36 coreos2 bash[2326]: last_appl. = 0,
Apr 07 21:31:36 coreos2 bash[2326]: protocols = 0/7/3 (gcs/repl/appl),
Apr 07 21:31:36 coreos2 bash[2326]: group UUID = e60cb081-fd06-11e5-8142-1e6e13e5d4f2
Apr 07 21:31:36 coreos2 bash[2326]: 2016-04-07 21:31:36 93 [Note] WSREP: Flow-control interval: [23, 23]
Apr 07 21:31:36 coreos2 bash[2326]: 2016-04-07 21:31:36 93 [Note] WSREP: New cluster view: global state: e60cb081-
```

### Il·lustració 31, Node nou forma part del clúster mysql

A la il·lustració 31 podem veure com els nodes s'afegeixen al node primari per formar el clúster. I a la il·lustració 32 podem veure tots els serveis engegats i també podem apreciar que cada discovery està a la mateixa màquina que el seu servei.

```
core@coreos1 /mnt/nfs/services/galera/zurmo $ fleetctl start galera_cluster_discovery@1.service
Unit galera_cluster_discovery@1.service inactive
Unit galera_cluster_discovery@1.service launched on a9e89f8b.../10.50.0.31
core@coreos1 /mnt/nfs/services/galera/zurmo $ fleetctl start galera_cluster@2.service
Unit galera_cluster@2.service inactive
^[[AUnit galera_cluster@2.service launched on e2753661.../10.50.0.221
core@coreos1 /mnt/nfs/services/galera/zurmo $ fleetctl start galera_cluster_discovery@2.service
Unit galera_cluster_discovery@2.service inactive
Unit galera_cluster_discovery@2.service launched on e2753661.../10.50.0.221
core@coreos1 /mnt/nfs/services/galera/zurmo $ fleetctl start galera_cluster@3.service
Unit galera_cluster@3.service inactive
Unit galera_cluster@3.service launched on f651d2c6.../10.50.0.96
core@coreos1 /mnt/nfs/services/galera/zurmo $ fleetctl start galera_cluster_discovery@3.service
Unit galera_cluster_discovery@3.service inactive
Unit galera_cluster_discovery@3.service launched on f651d2c6.../10.50.0.96
core@coreos1 /mnt/nfs/services/galera/zurmo $ fleetctl list-units
UNIT                                MACHINE                                ACTIVE SUB
galera_cluster@1.service            a9e89f8b.../10.50.0.31                active running
galera_cluster@2.service            e2753661.../10.50.0.221               active running
galera_cluster@3.service            f651d2c6.../10.50.0.96                active running
galera_cluster_discovery@1.service  a9e89f8b.../10.50.0.31                active running
galera_cluster_discovery@2.service  e2753661.../10.50.0.221               active running
galera_cluster_discovery@3.service  f651d2c6.../10.50.0.96                active running
core@coreos1 /mnt/nfs/services/galera/zurmo $
```

L'opció escollida pel projecte és la de tenir el clúster funcionant a l'empresa en un clúster CoreOS separat de la resta de serveis, com podeu veure a la il·lustració 25. El motiu principal és el consum de memòria RAM que té el motor de base de dades. Podríem tenir correctament dimensionats els servidors de base de dades sense haver de calcular els increments de contenidors de servidors web. L'escenari final serà un clúster CoreOS per implementar el clúster Galera Mysql i un clúster CoreOS per implementar els serveis apache i ftp.

### 10.5.3 SERVEI FTP

Per implementar el servei Ftp s'han estat mirant les imatges del Docker Hub. Si fem una cerca de les imatges de servidor ftp veiem que hi ha 320 repositoris que contenen la paraula ftp i 56 imatges del servidor ftp "vsftpd" que és el que he escollit per instal·lar.

Vsftpd és un servidor ftp segur que ens permet, entre d'altres coses, crear usuaris virtuals, aquest servidor també permet connexions mitjançant claus privades, i connectar-nos mitjançant sftp<sup>1</sup>.

Per implementar el servei he creat un repositori a Docker hub. S'ha creat un document Dockerfile, per crear una imatge pròpia docker, i la he pujat al repositori creat anteriorment. Tota la informació la podreu veure a l'annex 5. El servei té un script d'alta de client. Aquest escript crearà una carpeta al directori de l'usuari i crearà l'usuari a un fitxer segur passwd mitjançant htpasswd.

```
#!/bin/bash
# basat en l'escript de ruo91

username="$1"
userpass="$2"
pwd_file="/etc/vsftpd/users.passwd"

# Main
case $1 in
  $1)
    mkdir /home/$username
    htpasswd -bcd $pwd_file $username $userpass
    echo
    echo "User Name          : $username"
    echo "User Passwd         : $userpass"
    echo "Carpeta Creada       : /home/$username"
    echo "Encrypted Password  : `grep $username $pwd_file |
cut -d ':' -f 2`"
    ;;
  *)
    echo "Usage: $0 [USERNAME] [PASSWORD]"
    ;;
esac
```

#### Il·lustració 32, script usuari FTP

---

<sup>1</sup> SFTP: connexió a ftp mitjançant protocols segurs, com ara, SSL o TLS.

Per executar l'escript faig servir la comanda *Docker exec*, que executa la comanda que li passem per paràmetre al contenidor que li indiquem com hem copiat l'escript a la imatge mitjançant el Dockfile, només hem d'executar la comanda dins del contenidor.

```
core@coreos1 /mnt/nfs/services/ftp/conf $ docker exec vsftpd crea_usuari_ftp.sh prova2 prova2
Adding password for user prova2

User Name      : prova2
User Passwd    : prova2
S'ha creat la carpeta : /home/prova2
Encrypted Password : a7FnfVjEc1S76
core@coreos1 /mnt/nfs/services/ftp/conf $
```

**Il·lustració 33, exemple crear usuari nou al sistema**

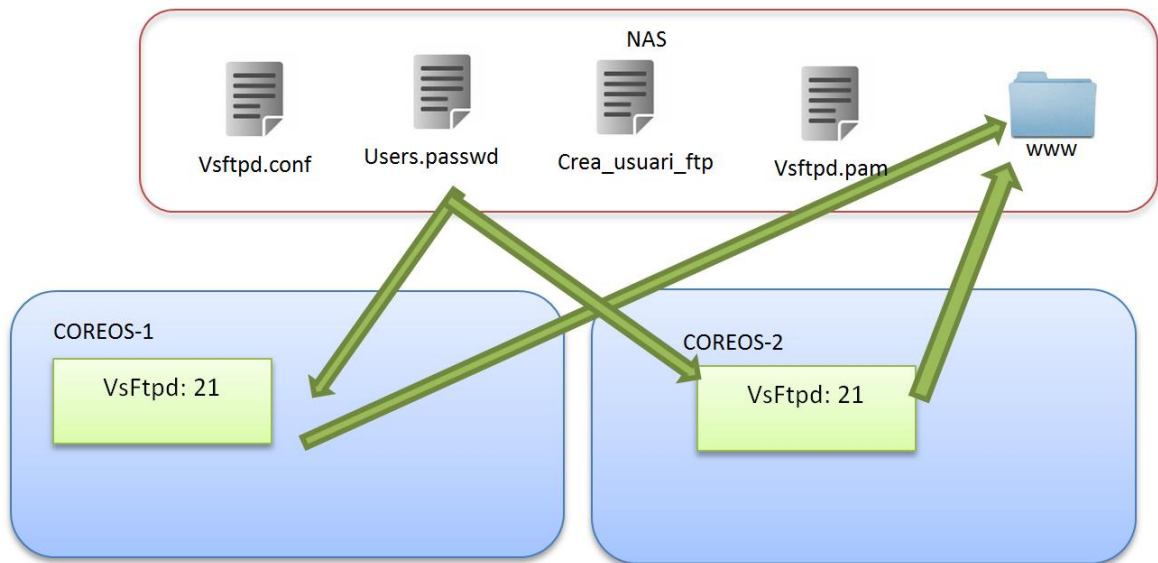
El funcionament és senzill, a l'iniciar el servei connectem el fitxer de configuració del NAS al contenidor, concretament a la ruta `/mnt/nfs/ftp/conf/users.passwd`. D'aquesta manera si iniciem un altre contenidor amb el servei vsftpd podrà llegir les dades de configuració i els usuaris creats del servidor NAS i no haurem de tornar a configurar els usuaris de nou.

```
core@coreos1 /mnt/nfs/services/ftpraul/conf $ docker run -d -p 21:21 --name vsftpd -h "vsftpd" -v /mnt/nfs/www:/home -v /mnt/nfs/services/ftp/conf:/etc/vsftpd raulrivero/vsftpd
d6ddb5199a7abb69e7e1f82a763ad95e3b67657c0f0b47770c8196d464bf0a44
core@coreos1 /mnt/nfs/services/ftpraul/conf $ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
d6ddb5199a7a        raulrivero/vsftpd  "vsftpd"           5 seconds ago      Up 4 seconds       0.0.0.0:21->21/tcp
```

**Il·lustració 34, executar contenidor ftp**

Els directoris que compartim amb els usuaris són els directoris de la seva carpeta web, que com vam veure a l'apartat 10.5.1 del servei apache la vam connectar a una carpeta també emmagatzemada al NAS, concretament a la ruta `/mnt/nfs/www`. Configurarem que el directori dels usuaris ftp sigui aquest mateix directori i així tindrem connectat els dos serveis.

Amb la implementació d'aquest servei ja tindrem les aplicacions necessàries que l'empresa ens demana per substituir l'actual infraestructura web feta amb servidors virtuals basats en Linux, els quals gestionen cadascun un nombre elevat de clients i si un client està sent atacat, per exemple per atacs de denegació de servei, la resta de clients que comparteixen el servidor es queden també sense servei.



Il·lustració 35, Infraestructura Servei FTP

## 10.6 DISPONIBILITAT DELS SERVEIS

Els serveis estan dissenyats per tenir una alta disponibilitat, els servidors formen un clúster i mitjançant fleet ens encarreguem de que els serveis allotjats als nodes estiguin balancejats i executant-se. Si un servei cau fleet s'encarrega de aixecar-ho.

D'altra banda tenim el clúster de servidor mysql. La disponibilitat és també molt alta ja que el servei s'executa als contenidors i agafen les dades d'un repositori extern. Els serveis estan dissenyats en una imatge i només amb aixecar-la (menys d'un minut), tornarem a tenir el servei en marxa.

L'altre servei, ftp, no és un servei crític tot i que hem dissenyat una imatge pròpia que ens garanteix poder modificar-la a la nostra voluntat per implementar seguretat o altres millores que veiem necessàries.

El disseny final de la infraestructura consta d'un clúster de 7 nodes (com a màxim) de CoreOS on estaran corrent els serveis web (apache) i el servei ftp. El primer anirà balancejat de càrrega mitjançant nginx, que és un proxy\_web, i el segon anirem engegant contenidors a mida que els clients vagin augmentant.

Per una altra banda, tenim un clúster de 3 CoreOS amb més memòria RAM que els anteriors que executaran el clúster Garlera Mysql. D'aquesta manera, si hem d'ampliar els servidors, tindrem independència respecte als servidors web i ftp.

Totes aquestes màquines i aplicacions aniran connectades a un sistema d'emmagatzematge extern, com ara una cabina de discos, que implementi seguretat tant a nivell lògic, disposant d'un RAID<sup>1</sup> 5 o 10 sobre les diferents LUN<sup>2</sup> que necessitem, com seguretat física amb redundància de fonts d'alimentació i controladores de discos.

---

<sup>1</sup> RAID: conjunt de discos redundants, evitant pèrdues de dades quan els discos fallen.

<sup>2</sup> LUN: partició virtual dins de un conjunt RAID.

## 11. COMPARATIVA INFRAESTRUCTURES

### 11.1 Comparativa d'arquitectures

#### Arquitectura Servidor Virtuals

Les arquitectures basades en servidors virtuals són arquitectures que sobre un maquinari executem diferents màquines, en el nostre cas sistemes operatius Linux, que a cada sistema operatiu tenim executant-se tres serveis, apache, mysql i ftp.

Cada servidor virtual esta assignat a una sèrie de clients fixes. Si ens donem d'alta a la plataforma de l'empresa de hosting ens assignen una màquina virtual, un espai al dispositiu d'emmagatzematge i configuren els arxius de configuració del servidor al que hem estat assignats perquè validi les nostres peticions i ens proporcioni els serveis contractats.

D'aquesta manera un servidor que gestioni 400 clients tindrà un fitxer de configuració apache i ftp, que serà exclusiu per a aquest servidor. Si s'han de substituir aquest servidor per un altre, haurem de carregar les dades al servidor nou.

En arquitectures de servidor virtual, normalment, cada servidor apache té un fitxer de configuració httpd.conf amb la quantitat de servidors virtuals configurats com clients gestioni, i aquest fitxer no és intercanviable entre servidors virtuals. Aquest fitxer apunta a una sèrie de carpetes web, normalment al directori /var/www, que està redirigit a un dispositiu d'emmagatzemament extern, que no pot ser gestionat per altres servidors virtuals ja que no tenen l'arxiu httpd.conf igual.

Quan un servidor falla, ja sigui apache o el servei que sigui, hem de refer tot el servidor, és a dir, restaurem la màquina des d'un estat anterior i mirem si funciona tot correctament, o s'ha de fer un servidor nou i restaurar els fitxers de configuració dels serveis. Normalment aquestes configuracions es fan amb un script que llegeix la llista d'usuaris donats d'alta al servidor que ha fallat i regenera el fitxer de configuració d'apache i ftp.

### Arquitectura CoreOS

L'arquitectura CoreOS és una arquitectura pensada per a l'escalabilitat, alta disponibilitat dels serveis i sobretot la facilitat de moure serveis entre infraestructures, ja que docker separa l'aplicació de la infraestructura utilitzada, és a dir, tenim els serveis.

Una altra diferència, és la de tenir més instàncies d'un mateix servei corrent sobre el mateix node. Podem tenir 30 servidors web i balancejar la càrrega entre els 30. A l'arquitectura màquina virtual hauríem de tenir 30 màquines per fer córrer els 30 servidors web.

Pel que fa a la programació, l'arquitectura CoreOS amb Docker ens permet crear les aplicacions dins d'un contenidor i compartir-les a través del Docker Hub amb altres col·laboradors, la podran engegar i testejar sigui el que sigui el hardware o sistema que facin servir i, un cop acabada, podrem posar en producció la nostra aplicació tant en els servidor propis o en servidors amb tecnologia Docker com Amazon.

Les aplicacions que fem servir són fàcilment actualitzables, simplement s'ha de crear una nova versió de la imatge Docker amb l'aplicació a actualitzar i engegar una nova instància de l'aplicació i aturar l'antiga.

En resum, les arquitectures que fan servir Docker, com ara CoreOs, són arquitectures molt més escalables, amb més disponibilitat de les aplicacions i molt més portables que les arquitectures basades en Servidors Virtuals.

### 11.2 Comparativa de rendiments

Per poder comparar els rendiments de les dues arquitectures, hem de tenir en compte alguns aspectes claus. Les arquitectures basades en Docker fan servir alguns serveis per gestionar els contenidors, executar els serveis, gestionar la càrrega de treball entre els nodes i els contenidors que s'executen a cada node i gestiona l'ample de banda del node per balancejar la càrrega entre els diferents contenidors.

Aquesta diferència la podem veure engenant un sistema CoreOS sense cap servei a part de Docker, és a dir, no engegarem ni etcd ni fleet i engegarem els serveis mysql i apache en el mateix contenidor, comparant-lo amb un servidor Debian amb apache i mysql funcionant.

```

root@debian:~# ps aux --width 30 --sort -rss | head
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
mysql     1198  0.0  3.6 327392 37384 ?        Ss   12:26   0:04 /usr/sbin/mysqld --basedi
root      887  0.0  2.7 128464 28080 ?        Ss   12:26   0:00 /usr/sbin/apache2 -k star
www-data 1229  0.0  0.7 128488  7216 ?        S    12:26   0:00 /usr/sbin/apache2 -k star
www-data 1230  0.0  0.7 128488  7216 ?        S    12:26   0:00 /usr/sbin/apache2 -k star
www-data 1231  0.0  0.7 128488  7216 ?        S    12:26   0:00 /usr/sbin/apache2 -k star
www-data 1232  0.0  0.7 128488  7216 ?        S    12:26   0:00 /usr/sbin/apache2 -k star
www-data 1233  0.0  0.7 128488  7216 ?        S    12:26   0:00 /usr/sbin/apache2 -k star
root      384  0.0  0.5   9260  5696 ?        Ss   12:26   0:00 dhclient -v -pf /run/dhcl
root     6399  0.3  0.5  11000  5444 ?        Ss   14:35   0:00 sshd: raul [priv]
root@debian:~# █
    
```

Il·lustració 36, Processos amb consum de memòria Debian, Font (pròpia)

```

core@coreosbase ~ $ ps aux --width 30 --sort -rss | head
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
104       1702  3.5  6.4 574532 64812 ?        Ss   12:08   0:00 /usr/sbin/mysqld --basedi
root      874  0.2  3.7 435776 37752 ?        Ss   12:03   0:00 docker daemon --host=fd:/
root     1174  0.2  3.1 354668 32356 ?        S    12:07   0:00 /usr/sbin/apache2 -D FORE
root     1156  0.1  2.7 230184 28216 ?        Ss   12:07   0:00 docker-proxy -proto tcp -
root     1147  0.1  2.5 164648 26216 ?        Ss   12:07   0:00 docker-proxy -proto tcp -
root      597  0.0  1.3 153996 13336 ?        Ss   12:01   0:00 /usr/sbin/update_engine -
root      406  0.0  0.9  39028  9568 ?        Ss   12:01   0:00 /usr/lib/systemd/systemd-
33       1177  0.0  0.9 354692  9496 ?        S    12:07   0:00 /usr/sbin/apache2 -D FORE
33       1178  0.0  0.9 354692  9496 ?        S    12:07   0:00 /usr/sbin/apache2 -D FORE
core@coreosbase ~ $ █
    
```

Il·lustració 37, Processos amb consum de memòria CoreOs, Font (pròpia)

A les il·lustracions 36 i 37 veiem els 10 processos que s'estan executant als servidors amb més consum de memòria, a la il·lustració 36 veiem que mysql i apache són els processos que més consum de memòria tenen i que al node CoreOS, a part d'aquests dos processos que tenen un consum similar de % de memòria que al sistema virtual, tenim un servei anomenat Docker que consumeix una part important de memòria i dos més que són docker-proxy que també tenen consum de memòria.

Aquests serveis, que són necessaris per poder gestionar la infraestructura, fan servir una quantitat de memòria ram i recursos hardware que un sistema operatiu base amb els serveis instal·lats no els consumeix. Per fer les proves de rendiment s'ha fet servir com a servidor virtual un Debian base amb apache, php i mysql instal·lat, sense cap altre servei ni aplicació.



De la banda de CoreOS s'han fet servir diferents modalitats d'implementar els serveis, les quals es numeren a continuació.

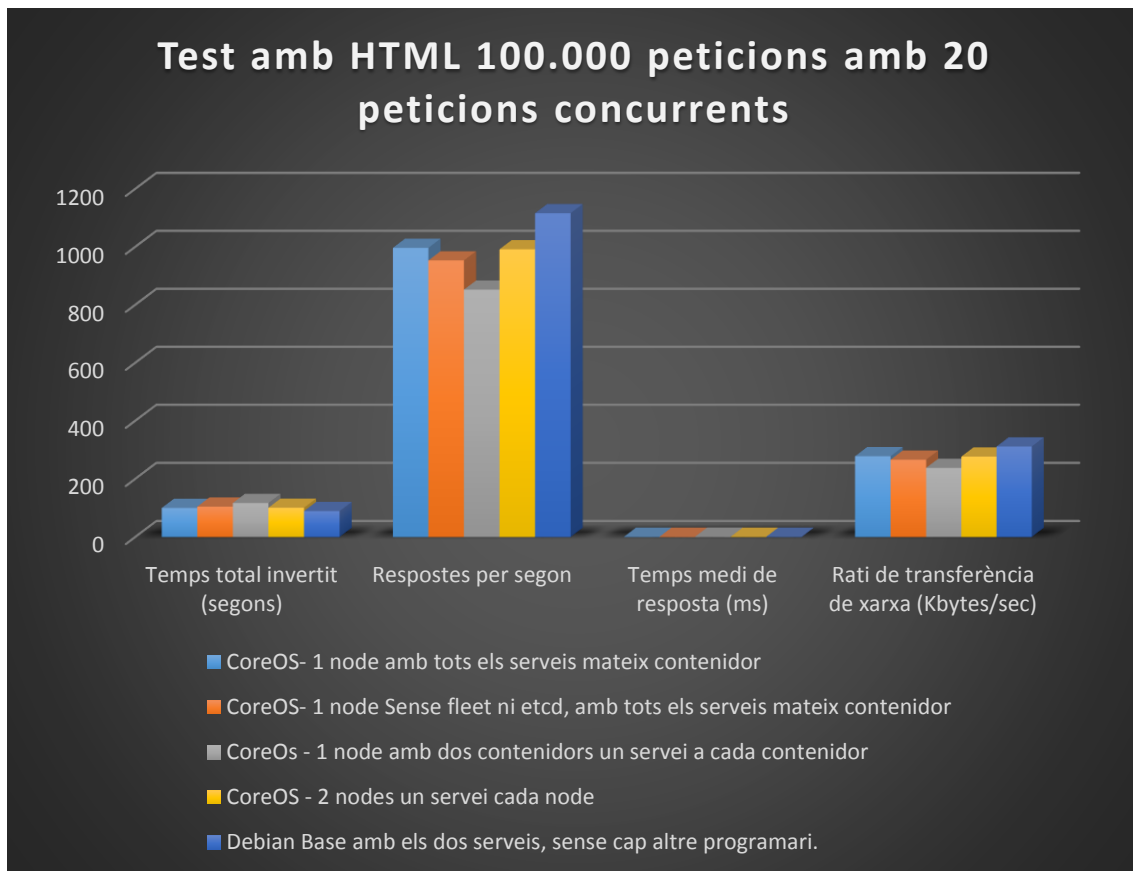
1. Un node CoreOS amb fleet i etcd, amb els serveis corrent en un mateix contenidor
2. Un node CoreOS amb fleet i etcd, amb un contenidor amb apache i un altre amb mysql.
3. Dos nodes CoreOS amb fleet i etcd, amb un contenidor amb apache i l'altre node amb un contenidor amb mysql.
4. Un node CoreOS sense fleet i etcd, amb els serveis corrent en un mateix contenidor.

Les proves de rendiment s'han realitzat amb el programa de BenchMark d'apache anomenat ab. S'han realitzat les proves fent 100.000 peticions a una pàgina web que només tenia contingut html i a una pàgina que feia consultes a una base de dades mysql amb una concurrència de 20, 1000 i 10.000 peticions alhora, s'han tingut en compte els següents paràmetres:

1. Temps total en fer les peticions.
2. Respostes de peticions per segon.
3. Temps de resposta medi d'una petició.
4. Rati de transferència a la xarxa.

S'ha realitzat els estudis tenint en compte el tipus d'infraestructura i un cop aclarit els aspectes importants s'han anat acotant els estudis fins arribar a comparar una màquina virtual amb un sistema operatiu Debian, amb un sistema operatiu CoreOs amb els serveis instal·lats, amb els mateixos paràmetres de configuració, es presenten aquests estudis tot indicant les conclusions a les quals s'arriben.

Estudi 1.- Comparativa entre Debian i les diferents modalitats d'implementar els serveis a CoreOS fent 100.000 peticions HTML.



Gràfic 1, Estudi 1

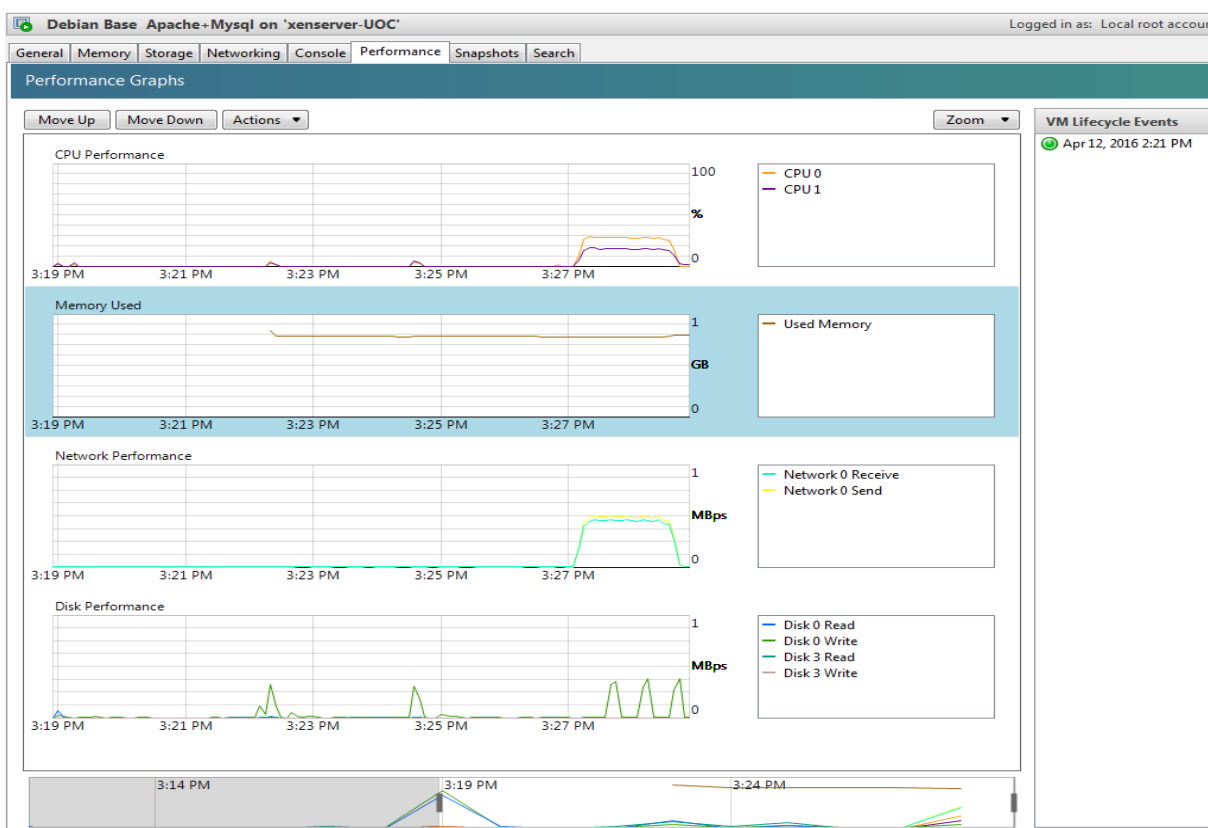
Taula 4, Dades Estudi 1

Sistema / Test	100.000 peticions amb 20 peticions concurrents, amb codi html			
	Temps total invertit (segons)	Respostes per segon	Temps medi de resposta (ms)	Rati de transferència de xarxa (Kbytes/sec)
CoreOS- 1 node amb tots els serveis mateix contenidor	100,322	996,79	1,003	278,4
CoreOS- 1 node Sense fleet ni etcd, amb tots els serveis mateix contenidor	104,875	953,51	1,049	266,31
CoreOs - 1 node amb dos contenidors un servei a cada contenidor	117,291	852,58	1,173	238,12
CoreOS - 2 nodes un servei cada node	100,819	991,87	1,008	277,03
Debian Base amb els dos serveis, sense cap altre programari.	89,566	1116,5	0,896	311,84

Podem veure al gràfic 1 i a la taula 4 que de les diferents modalitats d'implementar els serveis dins del sistema CoreOS, la que més temps necessita per realitzar les 100.000 peticions és la modalitat amb els serveis separats en dos contenidors. El motiu és que Docker reserva un ample de banda per a cada contenidor actiu. Podem veure com el rati de transferència és inferior que en les altres modalitats d'implementació.

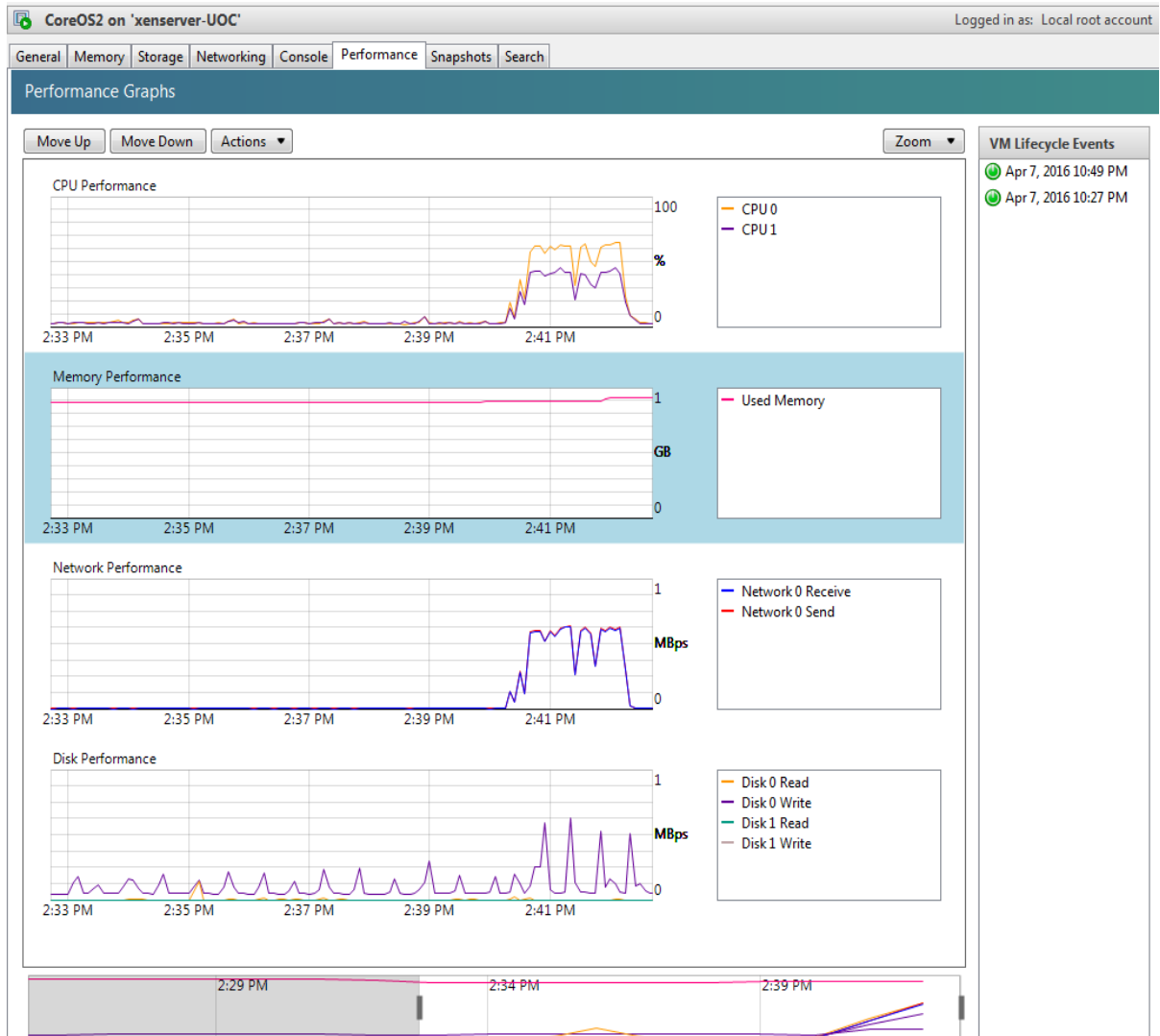
També podem observar que el sistema operatiu Debian amb els mateixos components hardware que les màquines CoreOS, en aquest estudi és més ràpid que els sistemes CoreOS. El motiu és que Docker instal·lat al CoreOS i el servidor apache que està funcionant dins del contenidor es comuniquen mitjançant la xarxa i el rati de resposta és inferior al Debian que no ha de fer servir la xarxa per comunicar-se amb el servidor web apache.

Podem determinar en aquest estudi, on CoreOS i Docker necessiten un ample de banda per gestionar els serveis, veiem que CoreOS és més lent que la màquina virtual. Pel que fa a l'ús de la cpu i de la memòria Ram també veiem que l'increment de recursos de tenir Docker funcionant a les màquines CoreOs és present quan fem l'estudi, com podem veure a les següents gràfiques.



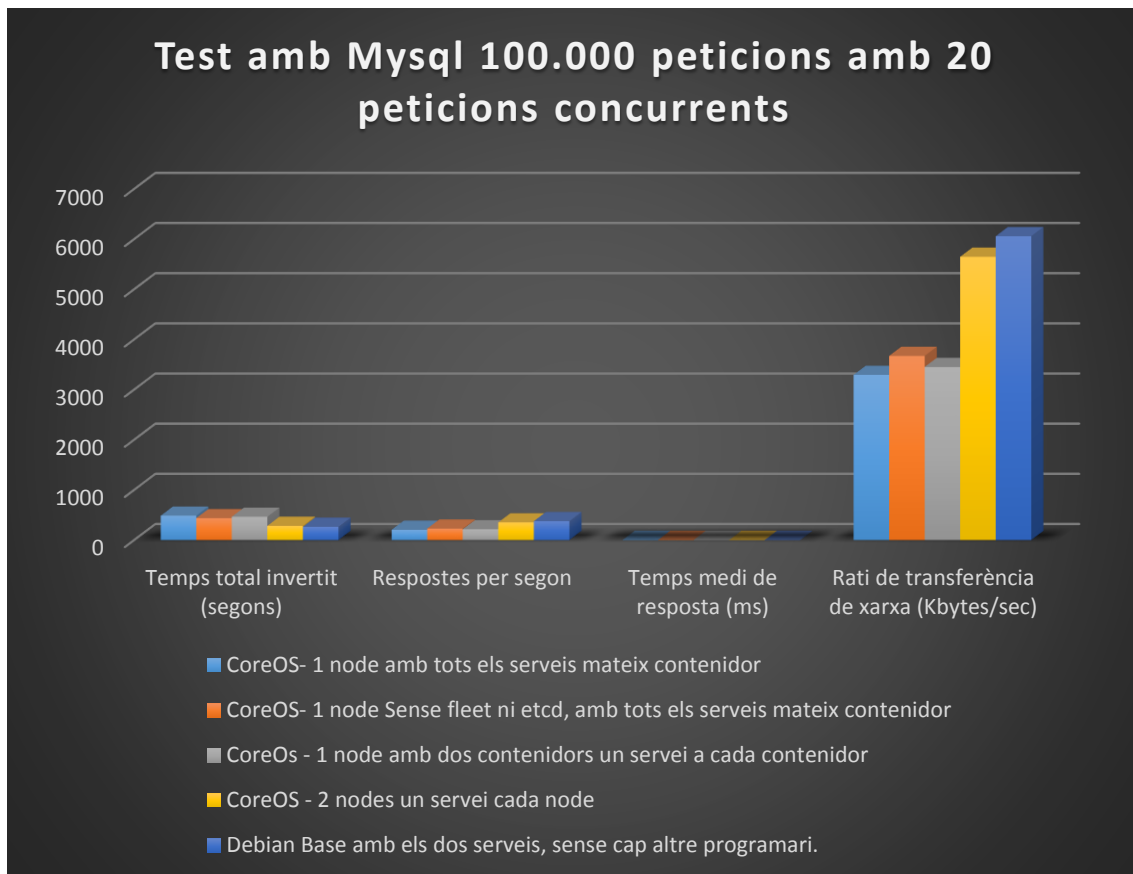
Il·lustració 38, Màquina virtual Debian fent 100.000 peticions HTML.

# IMPLEMENTACIÓ D'UNA INFRAESTRUCTURA WEB AMB COREOS I DOCKER



Il·lustració 39, CoreOS fent 100.000 peticions HTML

Estudi 2.- Comparativa entre Debian i les diferents modalitats d'implementar els serveis a CoreOS fent 100.000 peticions Mysql.



Gràfic 2, Estudi 2

Taula 5, Dades Estudi 2

Sistema / Test	100.000 peticions amb 20 peticions concurrents, amb consulta mysql			
	Temps total invertit (segons)	Respostes per segon	Temps medi de resposta (ms)	Rati de transferència de xarxa (Kbytes/sec)
<b>CoreOS- 1 node amb tots els serveis mateix contenidor</b>	486,952	205,36	4,87	3293,37
<b>CoreOS- 1 node Sense fleet ni etcd, amb tots els serveis mateix contenidor</b>	436,814	228,93	4,368	3671,38
<b>CoreOs - 1 node amb dos contenidors un servei a cada contenidor.</b>	464,761	215,16	4,648	3450,62
<b>CoreOS - 2 nodes un servei cada node</b>	283,936	352,19	2,839	5648,15
<b>Debian Base amb els dos serveis, sense cap altre programari</b>	264,691	377,8	2,647	6058,8

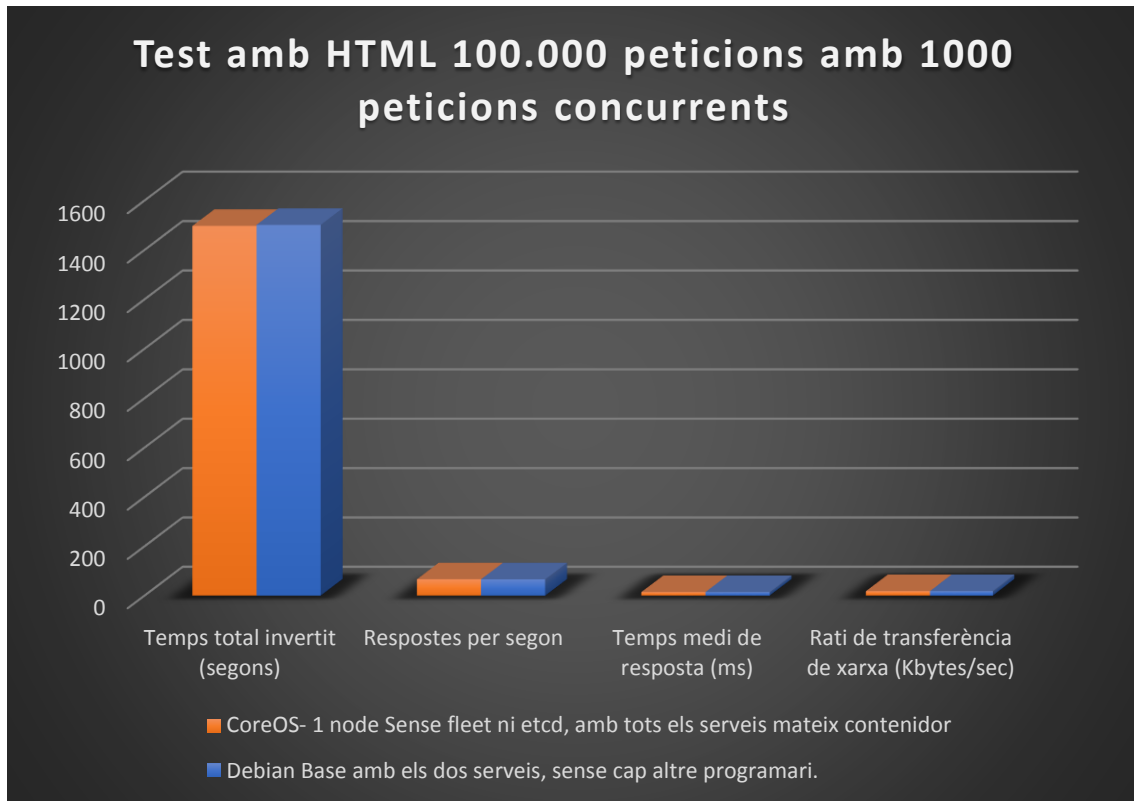
En aquest estudi podem veure que tenir dos serveis que s'hagin de comunicar entre ells, com és aquest cas on el servidor apache ha de comunicar-se amb el servei mysql per fer les peticions, al mateix servidor ja sigui en dos contenidors o en el mateix contenidor incrementa notablement el temps de resposta.

En canvi veiem en el cas de dos servidors CoreOS amb un contenidor amb un servei cadascú s'aproxima molt al rendiment de la màquina virtual. En aquest estudi veiem que Docker també perjudica, lleument, el rendiment del servei envers al servidor virtual.

Un cop plantejats aquests dos estudis podem concretar que en qüestió de rendiment la màquina virtual té una millor resposta que els serveis executats sota Docker en totes les opcions d'implementació que s'han detallat a l'estudi. Podem veure que un dels motius és el menor rati de transferència de la xarxa degut a la necessitat de comunicació entre Docker i els contenidors.

Pel següent estudi s'ha escollit una de les quatre implementacions de CoreOS estudiades amb anterioritat, vistes les quatre s'escull la més similar a la màquina virtual Debian i s'incrementen les peticions concurrents per tal d'estudiar el comportament de CoreOS i Docker en un escenari com aquest.

Estudi 3.- Comparativa entre Debian i CoreOS amb un únic contenidor amb php, apache i mysql instal·lat fent 100.000 peticions HTML i amb 1.000 peticions concurrents.



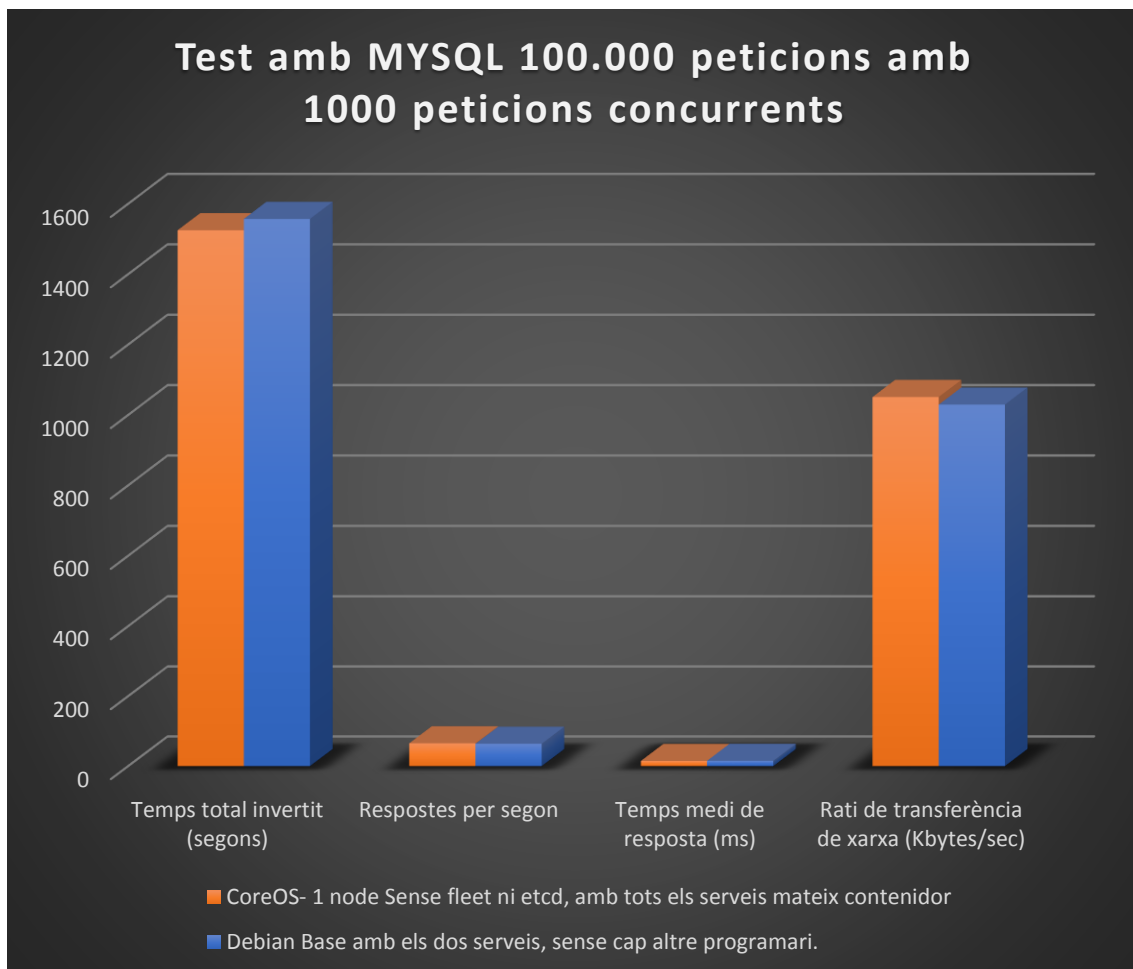
Gràfic 3, Estudi 3

Taula 6, Dades Estudi 3

100.000 peticions amb 1000 peticions concurrents, només html				
Sistema / Test	Temps total invertit (segons)	Respostes per segon	Temps medi de resposta (ms)	Rati de transferència de xarxa (Kbytes/sec)
<b>CoreOS- 1 node amb tots els serveis mateix contenidor</b>	1497,834	66,76	14,978	18,65
<b>Debian Base amb els dos serveis, sense cap altre programari.</b>	1502,072	66,57	15,021	18,59

Al gràfic 3 i a les dades de la taula 6, podem veure que el servidor basat en contenidors amb CoreOS té una millor resposta que el servidor virtual Debian, podem veure que com el rati de transferència necessari és baix, els dos sistemes tenen un comportament molt similar. Després de veure aquestes dades anem a veure com es comporta amb una aplicació que necessiti més rati de transferència i més memòria ram.

Estudi 4.- Comparativa entre Debian i CoreOS amb un únic contenidor amb php, apache i mysql instal·lat fent 100.000 peticions MYSQL i amb 1.000 peticions concurrents.



Gràfic 4, Estudi 4

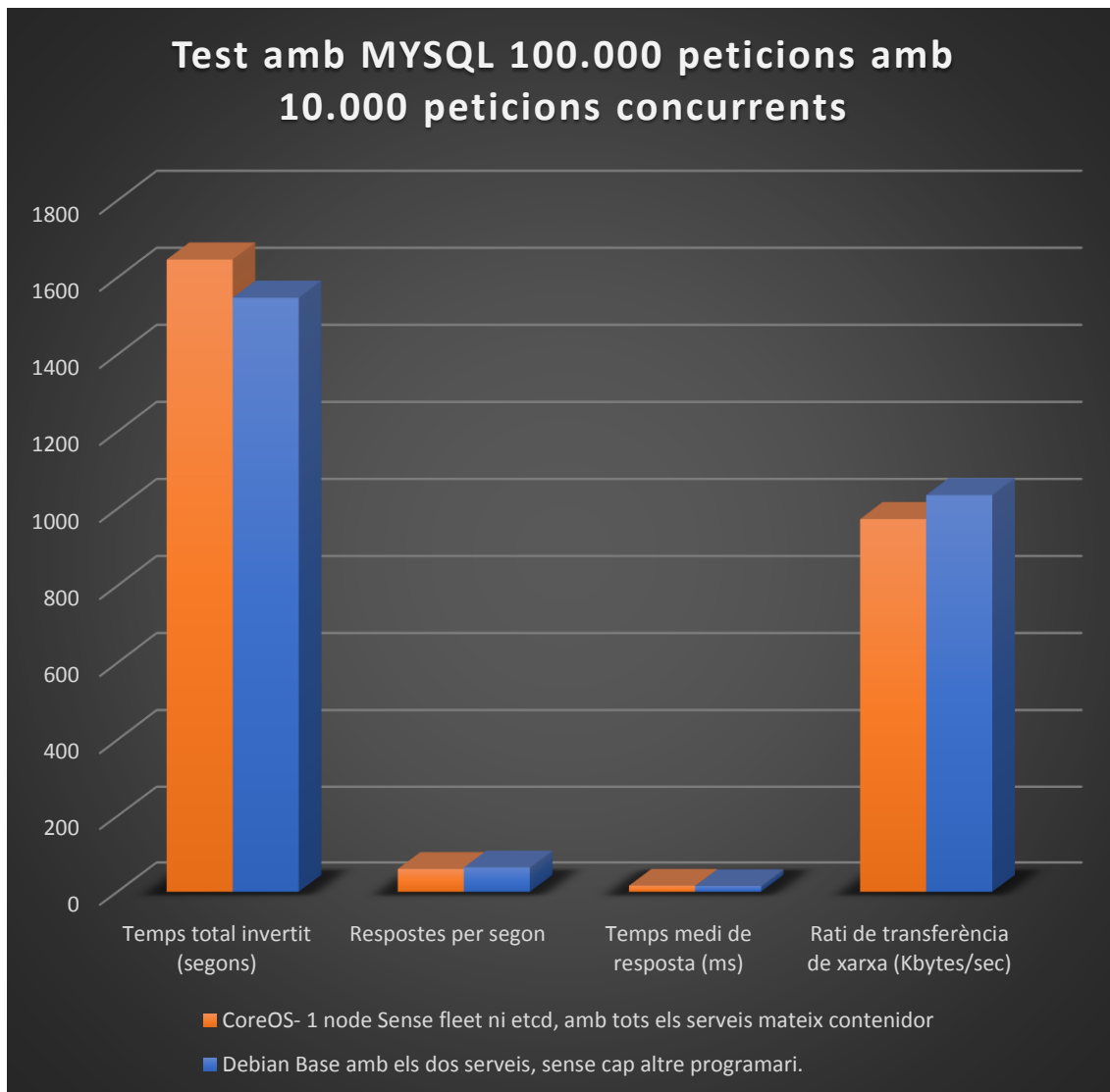


Taula 7, Dades estudi 4

<b>100.000 peticions amb 1000 peticions concurrents amb Mysql</b>				
<b>Sistema / Test</b>	<b>Temps total invertit (segons)</b>	<b>Respostes per segon</b>	<b>Temps medi de resposta (ms)</b>	<b>Rati de transferència de xarxa (Kbytes/sec)</b>
<b>CoreOS- 1 node amb tots els serveis mateix contenidor</b>	1525,227	65,56	15,252	1051,46
<b>Debian Base amb els dos serveis, sense cap altre programari.</b>	1557,366	64,21	15,574	1029,76

Amb l'estudi 4 es manté el resultat que s'ha vist a l'estudi 3. Quan tenim més peticions concurrents, Docker gestiona molt bé la xarxa i és millor en resposta i en rati que la màquina virtual. Tenim una millora substancial quan intentem estressar els servidor fent moltes peticions a la vegada en el servidor basat en contenidors.

Estudi 5.- Comparativa entre Debian i CoreOS amb un únic contenidor amb php, apache i mysql instal·lat fent 100.000 peticions MYSQL i amb 10.000 peticions concurrents.



Gràfic 5, Estudi 5

Veiem que amb 10.000 peticions concurrents la màquina virtual és més ràpida que el sistema operatiu CoreOS, tot i que la diferència és molt baixa. Podem determinar que CoreOS amb Docker ens aporta, a més de molts avantatges en disponibilitat i escalabilitat, ens gestiona d'una manera eficient el maquinari per a donar-nos un rendiment òptim.

Podem veure les dades del gràfic 5 a la taula 8.

Taula 8, Dades Estudi 5

100.000 peticions amb 10.000 peticions concurrents amb Mysql				
Sistema / Test	Temps total invertit (segons)	Respostes per segon	Temps medi de resposta (ms)	Rati de transferència de xarxa (Kbytes/sec)
<b>CoreOS- 1 node amb tots els serveis mateix contenidor</b>	1647,262	60,71	16,473	973,56
<b>Debian Base amb els dos serveis, sense cap altre programari.</b>	1548,25	64,59	15,483	1035,82

### 11.3 Comparativa d'escalabilitat

Els sistemes basats en contenidors, com ara CoreOS amb Docker són sistemes més escalables que les màquines virtuals. Quan es crea una aplicació dins d'un contenidor aquesta aplicació és independent del sistema operatiu en el que l'executem. Podem créixer amb sistemes basats en Docker amb independència de maquinari i sistema operatiu, és més portable.

Hem vist a l'apartat 10.4 com es gestiona un clúster CoreOS. Simplement amb una comanda podem afegir un node al nostre clúster i aquest serà replicat per la resta de membres del node i començarà a allotjar aplicacions d'immediat. Podem créixer tant com vulguem, només tenint en compte que un nombre molt alt de nodes replicant-se no és convenient per l'ús de la xarxa que necessiten.

En infraestructures de màquines virtuals l'escalabilitat comporta moltes més configuracions i estructuració dels clients per poder balancejar els sistemes. Necessitem tenir una base de dades de clients per anar creant els fitxers de configuració dels servidors web i ftp. Si un client ha de canviar de màquina per qualsevol motiu la ip pública de la web del client també canvia.

A la infraestructura que s'ha estudiat en aquest projecte tots els serveis web funcionen sota un proxy web, el que significa que tots els clients fan servir tots els servidor webs que hi ha al clúster i si volem ampliar la infraestructura només cal engegar nous serveis amb la imatge Docker que hem configurat.

No com a les infraestructures de màquines virtuals on una sèrie de clients fan servir un servidor virtual, també es podria muntar un servidor virtual amb un proxy web però continuariem tenint el mateix problema d'un únic punt de fallida. Això amb Docker és molt més senzill de solucionar ja que pots tenir més d'un contenidor amb el servei i si un node cau fleet el mourà a un altre node actiu del clúster.

### 11.4 Comparativa de disponibilitats

La disponibilitat de la infraestructura proposada és una disponibilitat del 100%, ja que el clúster té una sèrie de nodes que, mitjançant fleet, es repliquen els serveis que estan executant i si un node cau, fleet redistribueix el servei a un altre node.

Quan hem d'ampliar el clúster per motius de rendiment, podem afegir nodes sense cap tipus de tall dels serveis. Així mateix, quan un contenidor deixa de funcionar fleet s'encarrega de tornar a engegar un contenidor amb la mateixa imatge o una d'actualitzada al clúster.

Les actualitzacions de la infraestructura CoreOS amb Docker són molt menys "traumàtiques" que amb les màquines virtuals, simplement hem d'actualitzar la imatge Docker que volem modificar, ja sigui el servei web, ftp o mysql, pujar-la al repositori, i anar engegant contenidors nous amb la imatge nova i aturant els contenidors amb imatges antigues. No cal aturar cap servei ni posar en manteniment cap web del clients.

Com el servei que es crea al servidor per engegar els serveis amb fleet, descarreguen la darrera versió de la imatge que tenim al Docker hub, si un contenidor falla fleet torna a engegar el contenidor descarregant la darrera versió de la imatge. Això vol dir que podem actualitzar la imatge del Docker hub i aturar manualment els serveis que volem actualitzar i fleet s'encarregarà d'engegar-los.

A les infraestructures de servidors virtuals has de fer snapshots, actualitzar els serveis i reiniciar-los, el que suposa un tall del servei. Si tot ha anat correctament el tall serà petit però si no funciona haurem de restaurar l'estat anterior de la màquina virtual la qual cosa suposa una aturada dels serveis dels clients i un incompliment dels acords de servei.

## 12. CONCLUSIONS

Com a conclusió del projecte s'ha de puntualitzar que la idea principal de crear una infraestructura amb un sistema operatiu lleuger i amb un sistema de contenidors implementats amb Docker no és un millor rendiment de la infraestructura sinó que s'intenta millorar la disponibilitat, la modularitat i la escalabilitat de la infraestructura basada en servidors virtuals.

Hem vist que creant dos clústers diferenciats, un pels serveis web i ftp i un altre pel servei Mysql garantim que la infraestructura serà redundat i que no tindrem cap punt únic de fallida. Amb la possibilitat d'ampliar el nombre de nodes que formen els clústers i el nombre de contenidors que executen els servies, si l'augment de clients ho determina.

A les proves de rendiment s'ha comprovat que un servidor virtual amb únicament els serveis que s'han dut a estudi té un millor rendiment que el sistema CoreOS ja que aquest últim necessita de més serveis per poder engegar i gestionar els contenidors. Però que aquestes diferències de rendiment no són tant grans com les infinites possibilitats que ens dona CoreOS amb Docker.

Com a conclusió personal crec que aquesta infraestructures basades en contenidors aniran implantant-se a les empreses mitjanes i petites, ja sigui amb una infraestructura física o al núvol, amb la modularitat, disponibilitat i avantatges que ens dona envers a les infraestructures "tradicionals" el futur de les empreses seran amb Dockers. Ha estat un projecte molt enriquidor que m'ha permès aprendre molt sobre un tema el qual no tenia cap experiència.

## 12.1 Objectius assolits

Els objectius plantejats al començament del projecte són els següents:

### Instal·lació i configuració de l'entorn Hypervisor.

S'ha instal·lat un servidor Xen amb els Service Pack1 que dona suport per crear màquines virtuals amb el sistema operatiu CoreOs. Aquest hypervisor ha permès instal·lar un clúster CoreOS amb altes de nodes nous i baixes. Així mateix, ha permès instal·lar una màquina virtual basada en Debian per fer les proves de rendiment.

### Instal·lació i configuració del clúster CoreOS amb etcd, fleet i docker.

S'ha instal·lat un clúster de 3 nodes amb CoreOS. S'han configurat els fitxers Cloud-Config i s'han fet proves de disponibilitat amb fleet creant les imatges Docker. S'ha comprovat la potència de crear un servei amb la tecnologia Docker i s'ha comprovat el funcionament de fleet per mantenir els serveis en marxa quan un servei deixa de funcionar.

### Implementació dels serveis pels usuaris.

S'han implementat tots els serveis que es van plantejar a l'inici del projecte, s'ha instal·lat un servei web que es compon d'un servidor Proxy que balanceja la càrrega de les peticions entre tots els servidors que s'aixequin al clúster. Aquests serveis tenen un mecanisme d'auto descoberta que modifica els fitxers upstream del Proxy per actualitzar la llista de servidors disponibles.

S'ha creat un recurs compartit NFS per tal d'emmagatzemar els scripts, les web i els fitxers de configuració dels serveis. Amb aquests recursos compartits tots els serveis podem veure i llegir els fitxers de configuració i les carpetes de les pàgines web són les mateixes per a tots els nodes del clúster.

S'ha implementat el servei Mysql dins del clúster de 3 nodes. Per aquest servei s'ha recomanat fer un clúster separat dels altres serveis pel consum de memòria RAM que necessita el gestor de base de dades Mysql. El servei s'ha creat amb una imatge Docker creada per Zurmo, un usuari de Git Hub, la qual permet molt fàcilment aixecar el clúster i gestionar-ho.

S'ha implementat un servei Ftp perquè els clients puguin gestionar la seva carpeta web. Aquest servei té un script per crear usuaris, el qual crea la carpeta al recurs compartit amb NFS i dona d'alta l'usuari als fitxers de configuració i d'accés als servidors també allotjats a un recurs compartit mitjançant NFS.

### Comparativa entre les dues arquitectures.

L'estudi de rendiment ens ha aclarit que els servidors implementats amb CoreOS tenen un consum de memòria RAM, CPU i de recursos de xarxa que una màquina virtual amb els serveis bàsics no el té. Tanmateix, la gestió dels nodes CoreOS amb Docker quan fem moltes peticions concurrents s'aproxima molt al rendiment que té la màquina virtual pel que fa a temps de resposta, encara que tingui un major consum de RAM i CPU.

Pel que fa a la disponibilitat s'ha arribat a la conclusió de que els sistemes basats en CoreOS i Docker tenen una major disponibilitat que els servidors virtuals, ja que els serveis com ara fleet s'encarreguen de tenir sempre en marxa els serveis que configurem. Podem tenir serveis independents del maquinari, els quals els podem engegar a qualsevol servidor que tingui instal·lar Docker, ja sigui a les nostres dependències o a una infraestructura al núvol.

També s'ha comparat l'escalabilitat, s'han comparat les dues infraestructures i veiem que CoreOS permet una escalabilitat molt potent permetent afegir nodes al clúster amb independència de maquinari, simplement amb una comanda ens dona la informació necessària per configurar un nou node i que aquest s'integri al clúster.

També s'ha vist que l'escalabilitat de serveis també és molt més eficaç que amb les infraestructures de màquina virtual. Amb CoreOS i Docker podem engegar tants servidors ftp o web, com necessitem fent servir els serveis creats amb fleet sense que s'hagi de fer cap configuració a mida dels serveis. Si els servidors es queden curts de recursos podem inserir un nou node al clúster i continuar creixen sense cap tipus de reestructuració de la infraestructura.

### 12.2 Futur del projecte

Aquest projecte és un exemple d'infraestructura, amb serveis en contenidors, basant-se en les infraestructures que tenen les empreses hosting actualment. Aquesta manera d'implementar els serveis, aplicacions i tot el que necessitem ens proporciona una llibertat absoluta a l'hora de fer servir les nostres aplicacions o serveis implementats.

Podem crear, per exemple, una aplicació basada en web que necessiti apache, mysql, php, python, etc... i empaquetar-la en una imatge Docker. Aquesta imatge la podrem iniciar en qualsevol maquinari que tingui instal·lat Docker, no haurem de patir per la configuració del maquinari i tindrem una aplicació portable a qualsevol servidor.

Per a les escoles és un sistema molt flexible, podem engegar 50 contenidors Docker amb ubuntu perquè els usuaris facin les practiques de sistemes. Podem engegar servidor apache per fer proves de seguretat sense tenir la por de que el servidor caurà o es comprometrà la seva seguretat. Simplement aturant i engegant un altre contenidor tindrem el servidor apache com al començament.

Crec que les empreses de hosting haurien d'anar implementant aquest tipus de solució a les seves infraestructures, ja que dóna una disponibilitat, flexibilitat, modularitat i escalabilitat que amb els sistemes virtuals actuals no tenen. També poden crear nous productes amb aquesta infraestructura, com ara, la possibilitat de poder executar imatges Docker als clients.



### 13. BIBLIOGRAFIA

CoreOS (2016). Web oficial del sistema operatiu CoreOS [web del producte en línia]. CoreOS. [ Febrer- Juny 2016].

<https://coreos.com>

Docker (2016). Web oficial de Docker [web del producte en línia]. Docker. [ Febrer- Juny 2016].

<https://www.docker.com>

Debian (2016). Web oficial del sistema operatiu Debian [web del producte en línia]. Debian. [ Febrer- Juny 2016].

<https://www.debian.org>

Grandi, Andrea (2014). Automatically pull updated Docker images and restart containers with docker-puller [publicació en línia]. Andrea Gandi [ Febrer- Juny 2016].

<https://www.andreagrandi.it/2014/10/25/automatically-pull-updated-docker-images-and-restart-containers-with-docker-puller/>

O'neill, Michael (2009). How to configure an NFS based ISO SR using local storage on a XenServer [publicació en línia]. Citrix - Blog. [ Febrer 2016].

<https://www.citrix.com/blogs/2009/01/26/how-to-configure-an-nfs-based-iso-sr-using-local-storage-on-a-xenserver5-host/>

Guzman (2014). Como configurar un servidor NFS en XenServer [publicació en línia]. GuzmanWeb. [ Març 2016].

<http://guzmanweb.com.ar/blog/?p=1131>

Anònim (2011). Foro StackOverFlow [foro en línia]. Stackoverflow.com. [ Abril- Juny 2016].

<http://stackoverflow.com/questions/4165116/can-include-directive-be-used-within-upstream-block-of-nginx>

EntropyWorks (2016). Git Hub EntropyWorks [repositori en línia]. Git Hub.com. [ Febrer- Juny 2016].

<https://github.com/EntropyWorks/fleet-units-galera-cluster>

Severalnines (2014). How to Deploy Galera Cluster for MySQL using Docker Containers [article en línia]. Severalnines. [ Febrer- Maig 2016].

<http://severalnines.com/blog/how-deploy-galera-cluster-mysql-using-docker-containers>

Yanar, Erkan (2015). Getting Started Galera with Docker [publicació en línia]. GaleraCluster. [ Març - Abril 2016].

<http://galeracluster.com/2015/05/getting-started-galera-with-docker-part-2-2/>

Icclab (2015). Git hub Icclab [publicació en línia]. Git Hub Zurmo Galera cluster. [ Febrer- Juny 2016].

[https://github.com/icclab/cna-seed-project/tree/master/zurmo\\_galera\\_cluster](https://github.com/icclab/cna-seed-project/tree/master/zurmo_galera_cluster)

Galera Cluster (2016). Monitoring clúster status [Publicació en línia]. Galera Cluster. [ Febrer- Juny 2016].

<http://galeracluster.com/documentation-webpages/monitoringthecluster.html>

HScripts (2016). Comandes Linux “Sed” [publicació en línia]. HScripts. [ Febrer- Juny 2016].

<https://www.hscripts.com/es/tutoriales/linux-commands/sed.html>

Ruo91 (2016). Git Hub Ruo91 [web del producte en línia]. Git Hub Repositori Vsftpd. [ Febrer- Juny 2016].

<https://github.com/ruo91/docker-vsftpd>

Alcantarilla, Cristina (2014). Servidor FTP, UBUNTU 12.04 [web del producte en línia]. CoreOS. [ Febrer- Juny 2016].

<https://coreos.com>

Díaz, Eduardo (2014). Instal·lació Vsftpd amb usuaris virtuals [publicació en línia]. Eithel Inside. [ Març- Abril 2016].

<https://coreos.com>

14. ANNEXOS

Índex

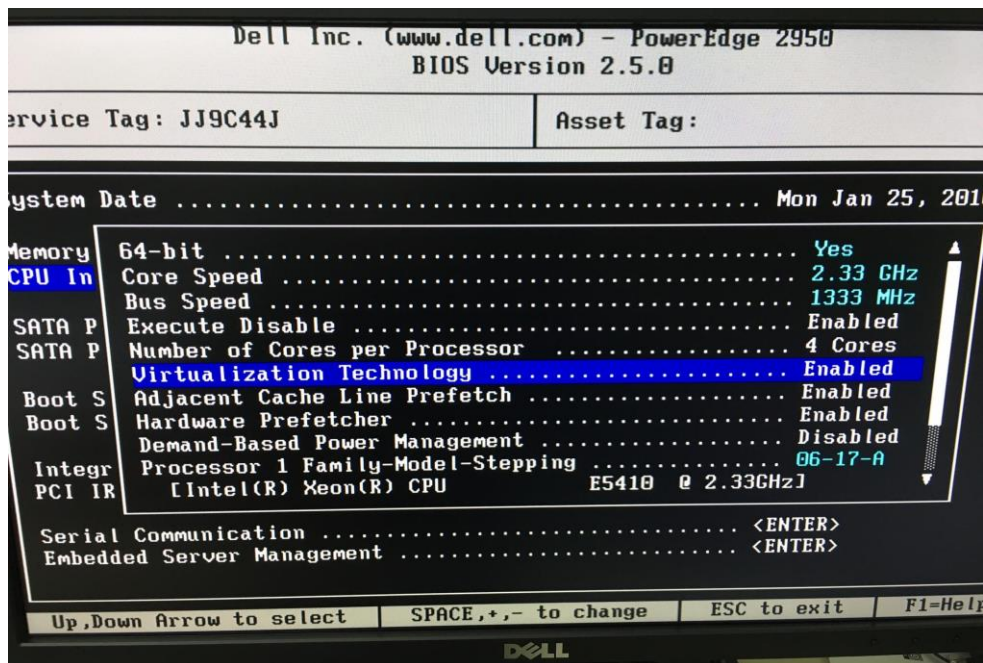
1.Instal·lació de XenServer .....	2
2.Instal·lació de SP1 sobre XenServer 6.5 .....	12
3.Creació d'un repositori per emmagatzemar ISO'S .....	18

## 1. Instal·lació de XenServer.

Per fer el projecte s'ha optat per la solució d'instal·lar XenServer com a capa de virtualització. S'ha escollit aquesta solució entre d'altres com per exemple, Proxmox o Vmware per descobrir un sistema que encara no havia treballat amb ell.

Primerament, cal indicar que he comptat amb un servidor DELL, concretament PowerEdge 2950 amb 1 CPU amb 4 Cores a 2.33 GHZ amb una Bus a 1333 MHZ, el servidor té instal·lat 4 GB de memòria RAM a 667 MHz i amb 4 discos SCSI de 400 GB.

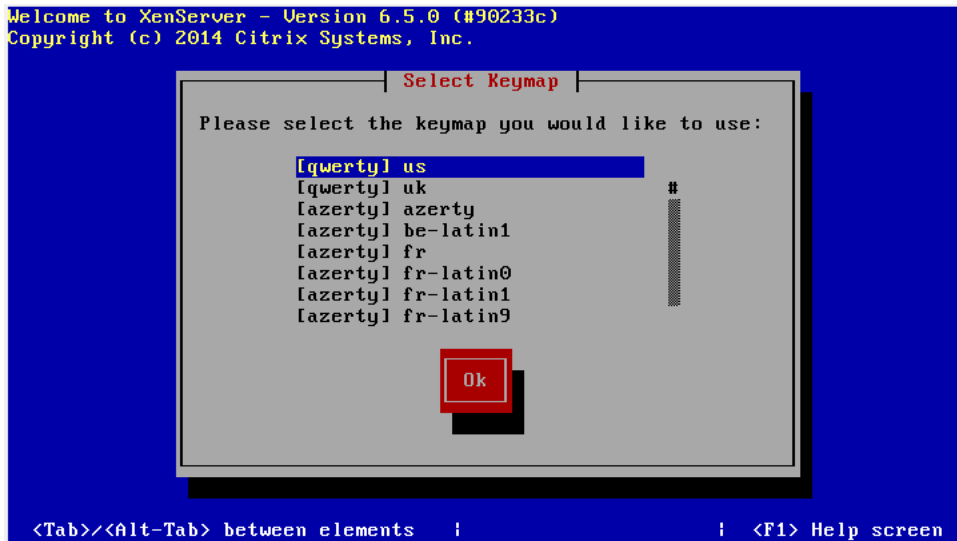
Per fer una instal·lació d'un sistema operatiu KVM necessitem activar l'opció de "Enabled Virtualization Technology" a la bios del nostre servidor. Si aquesta opció no està activada el sistema operatiu no s'instal·larà correctament.



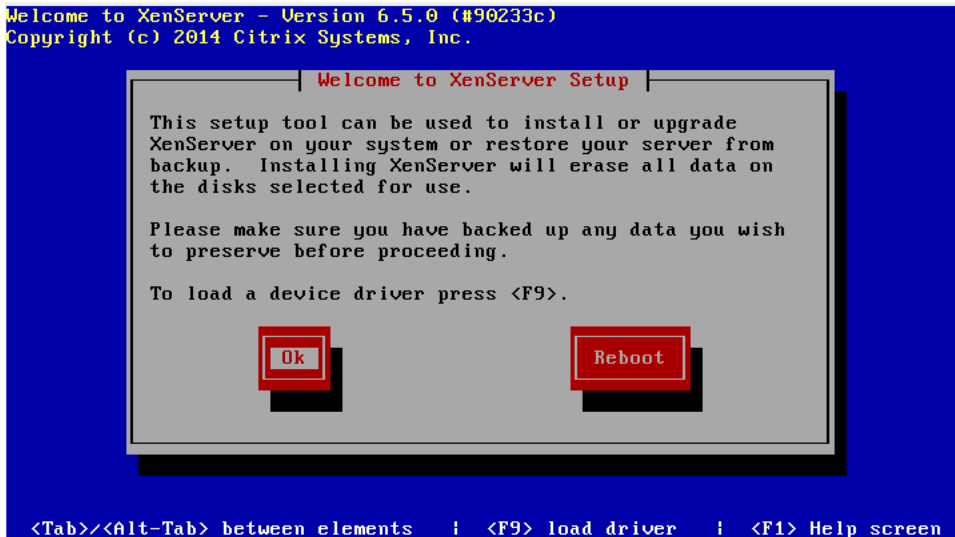
La instal·lació del servidor és molt intuïtiva, cal configurar els paràmetres de disc, xarxa, etc...

1. Iniciem el servidor i botem des de l'usb o cd depenent el medi d'instal·lació que necessitem.

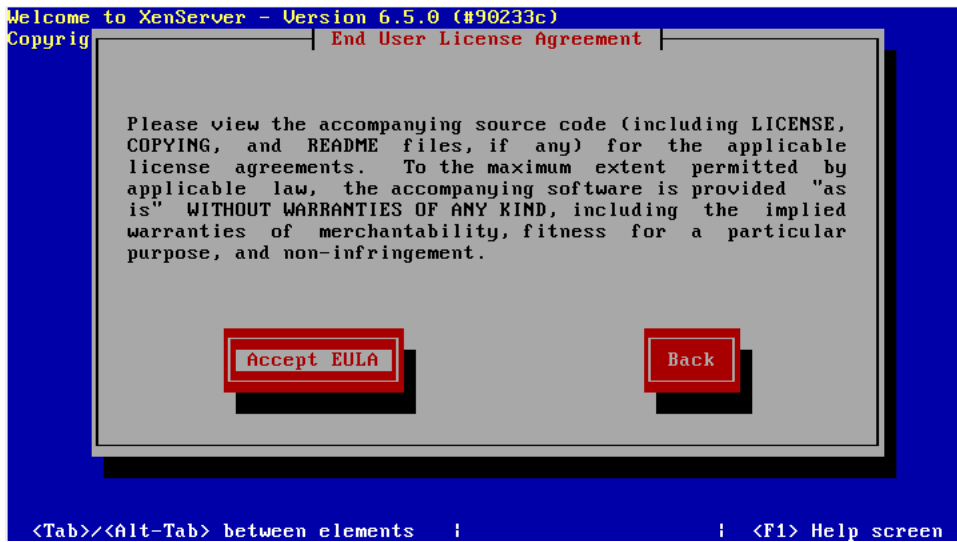
2. Seleccionem el teclat, en el meu cas he seleccionat “[qwerty] -es”.



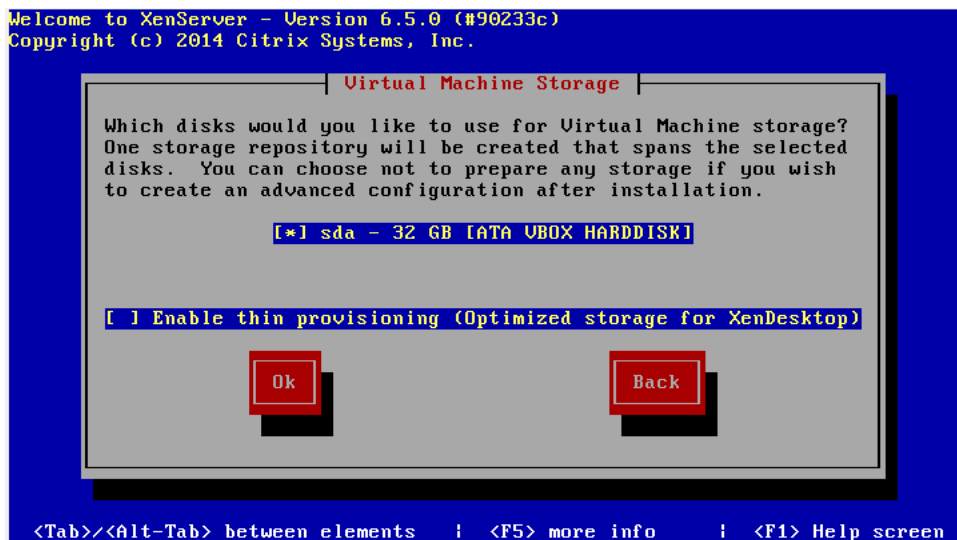
3. Mitjançant la tecla F9 podem instal·lar drivers de controladors de discos, ethernet o quelcom que necessitem per poder posar el sistema en marxa.



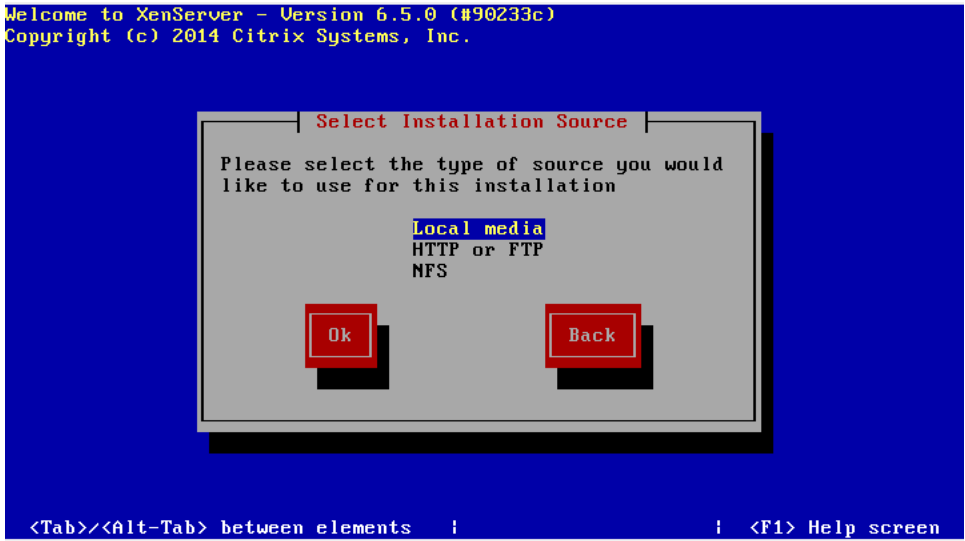
4. Acceptem la llicència d'ús.



5. Seleccionem els discos els quals s'utilitzaran per l'emmagatzematge de les màquines virtuals que creem.



6. Seleccionem l'origen de la instal·lació. Com podem veure a la imatge pot ser directament del cd o usb amb el qual hem arrencat l'instal·lador, a través d'internet per http o ftp i a través d'un sistema de fitxers de xarxa NFS.

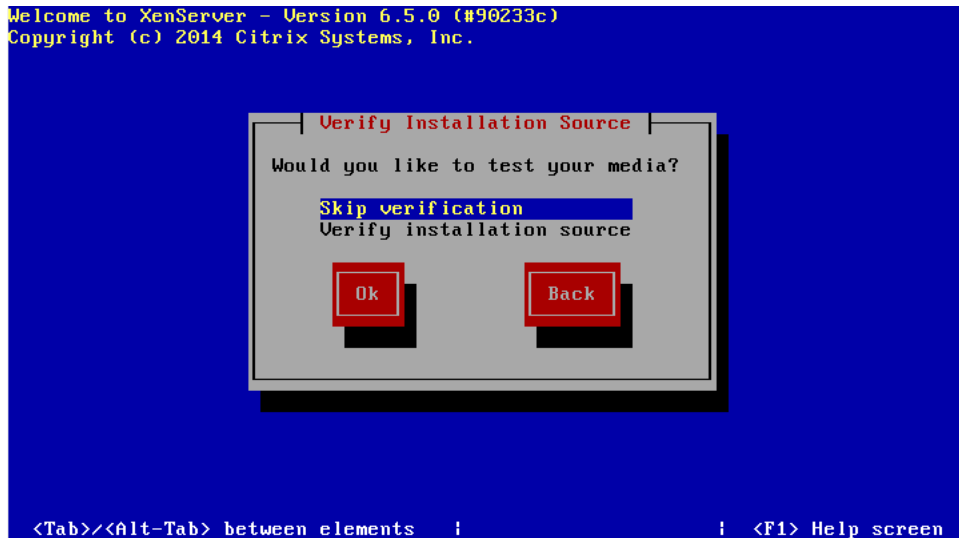


7. Ens demana si volem instal·lar algun paquet addicional, com ara un service pack, que el necessitem per donar suport a CoreOS, però que l'instal·larem més endavant.

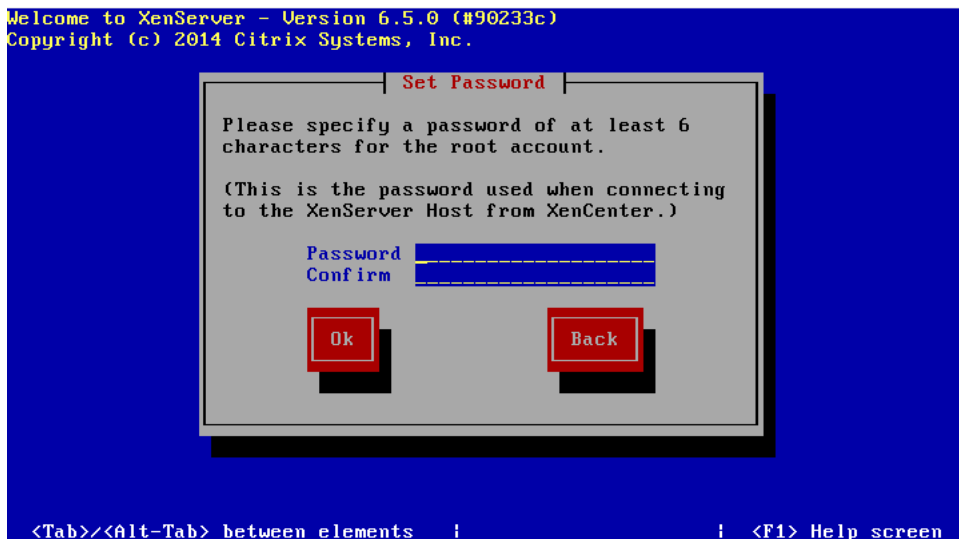




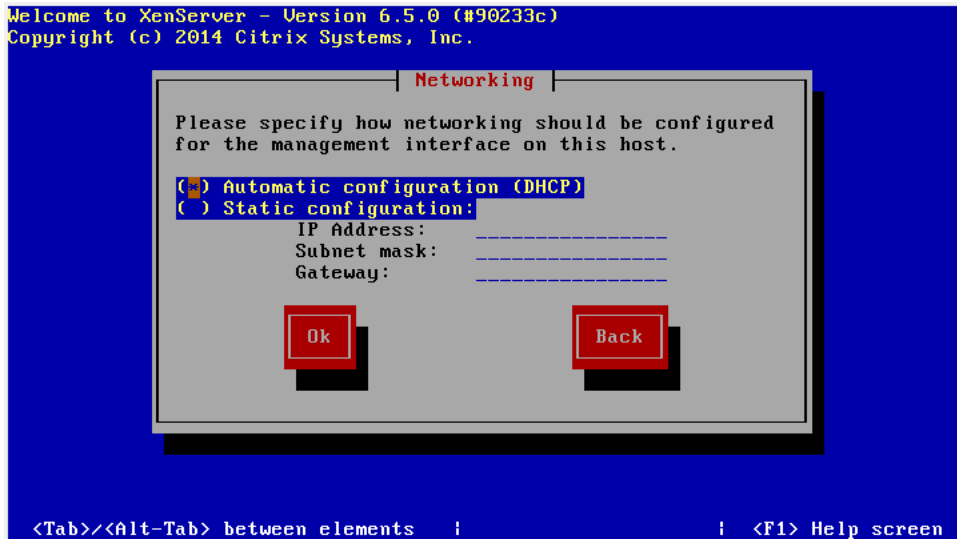
8. El sistema ens demana si volem verificar la font des d'on s'instal·larà el sistema operatiu.



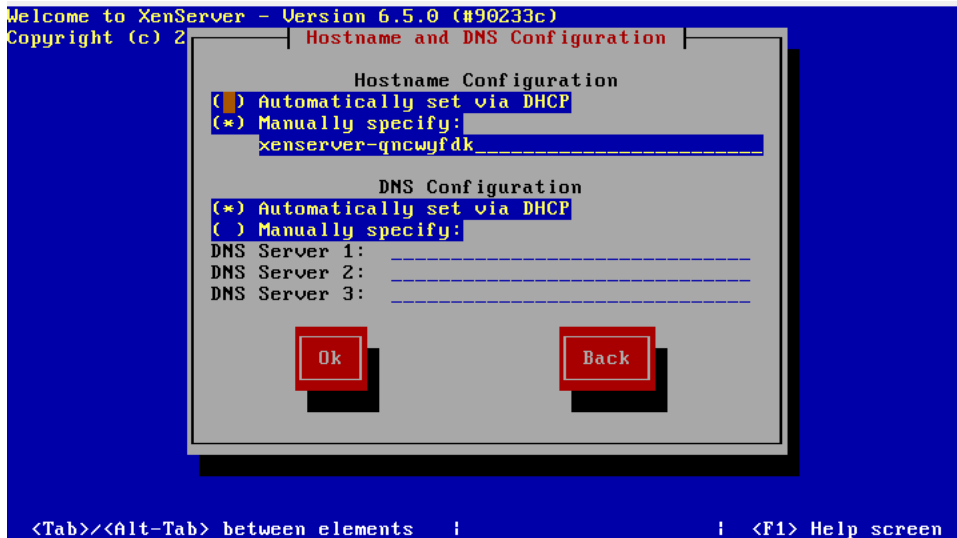
9. Hem de configurar la contrasenya que tindrà el sistema.



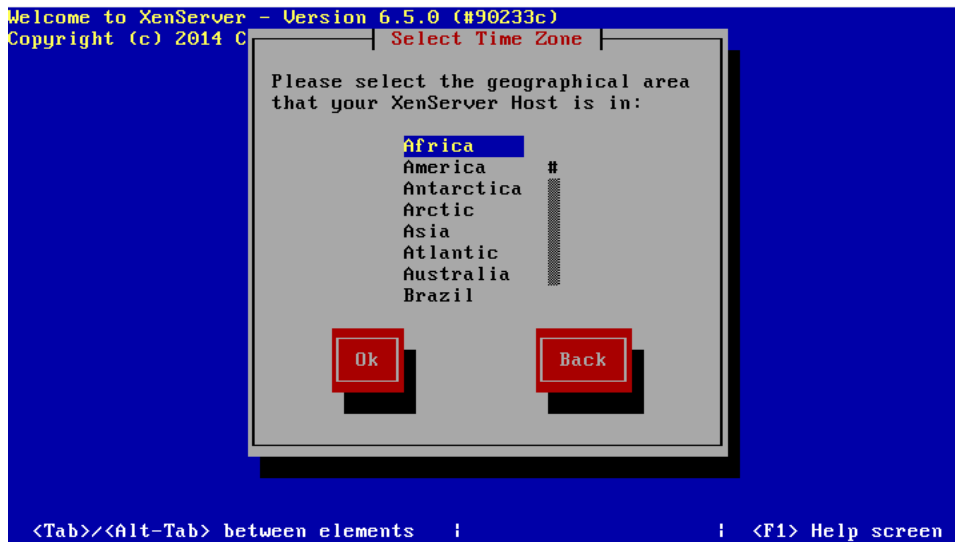
10. Ara hem de configurar la xarxa, si tenim un servidor DHCP a la xarxa podem posar automàtic, però aquest tipus de sistemes és més recomanable posar una ip estàtica, tot i que un cop instal·lat et surt una pantalla on indica quina ip té el sistema.



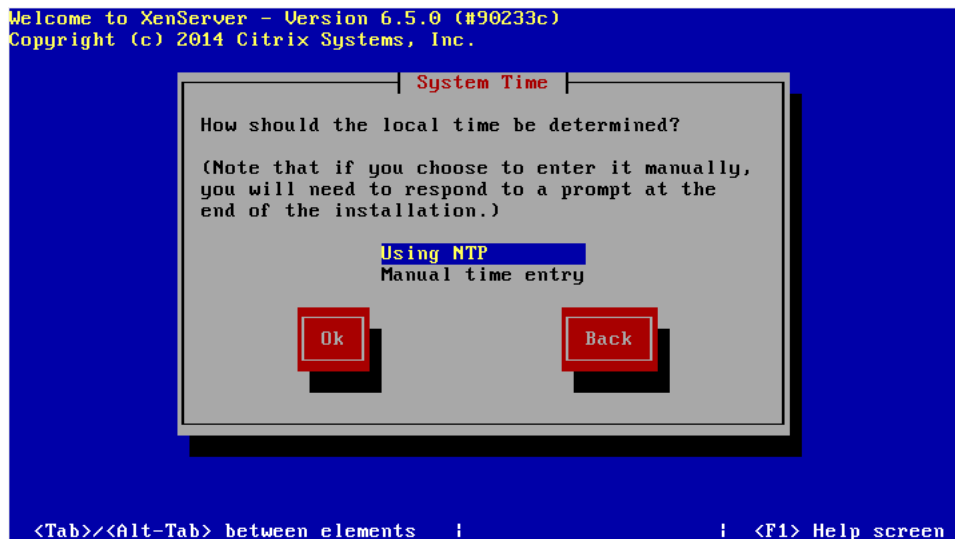
11. Indiquem el nom del host.



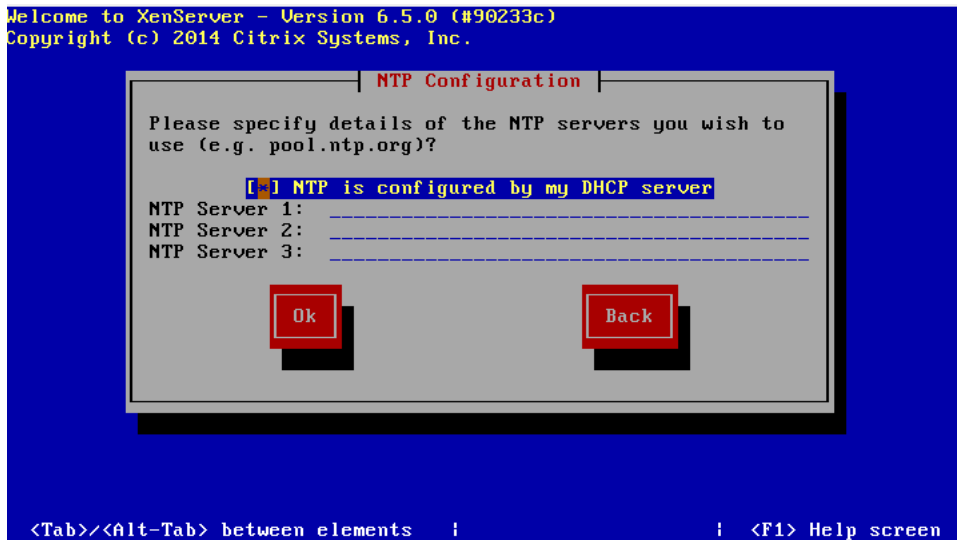
12. Seleccionem l'àrea geogràfica on està instal·lat el sistema, en el nostre cas Europe.



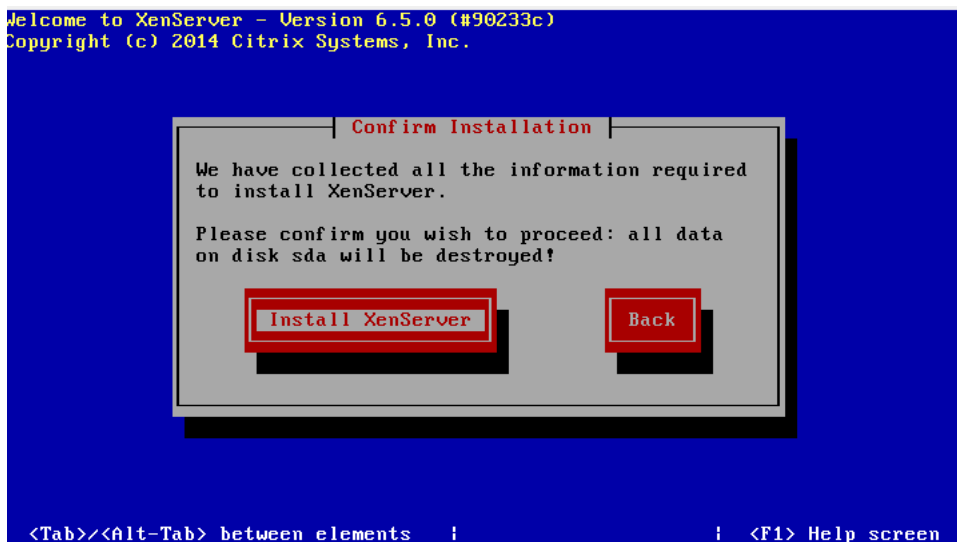
13. Seleccionem si volem configurar l'hora amb un servidor NTP, automàticament, o la volem posar manualment.

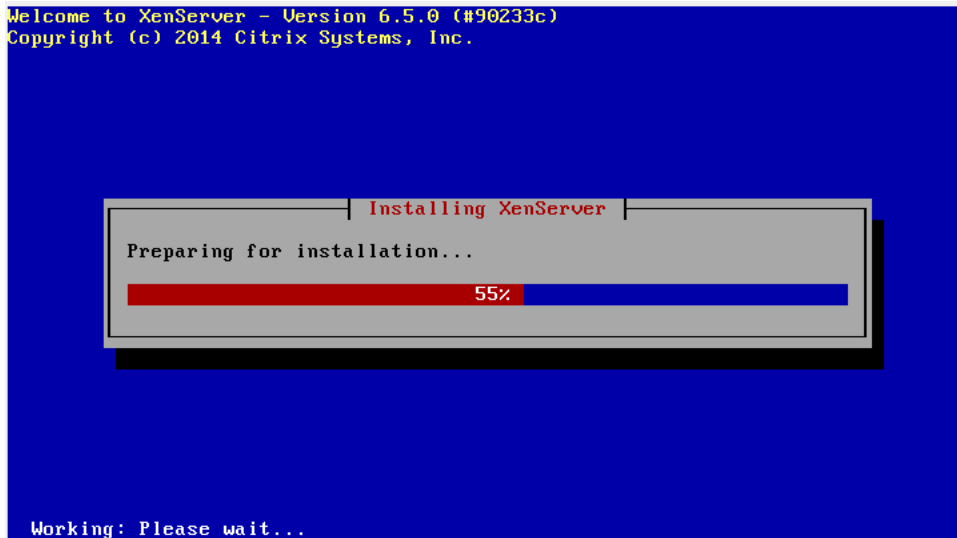


14. Si seleccionem automàtic ens demana que configurem els servidors, un molt conegut és pool.ntp.org, que és com el seu nom indica una pool de servidor de temps.

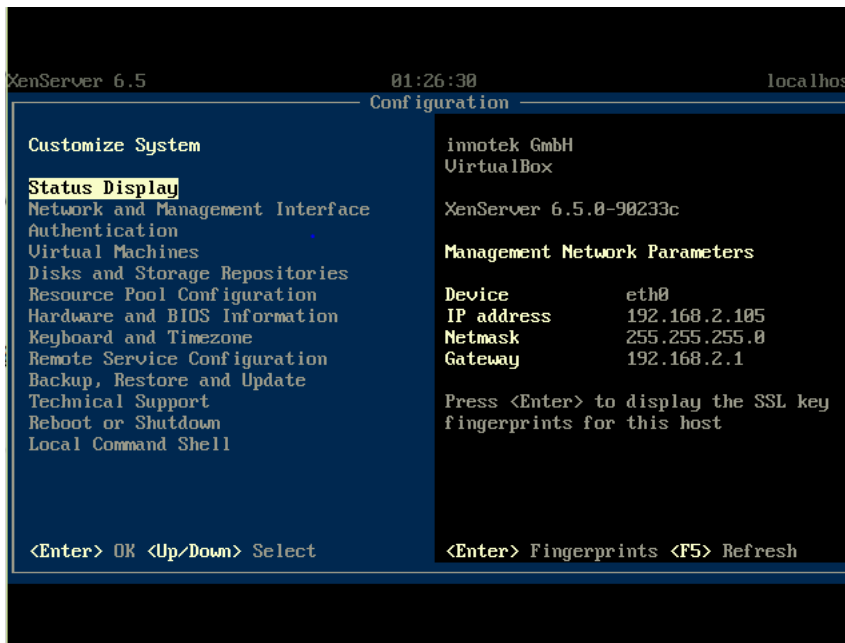


15. A la fi, ja podem instal·lar Xen Server al maquinari.



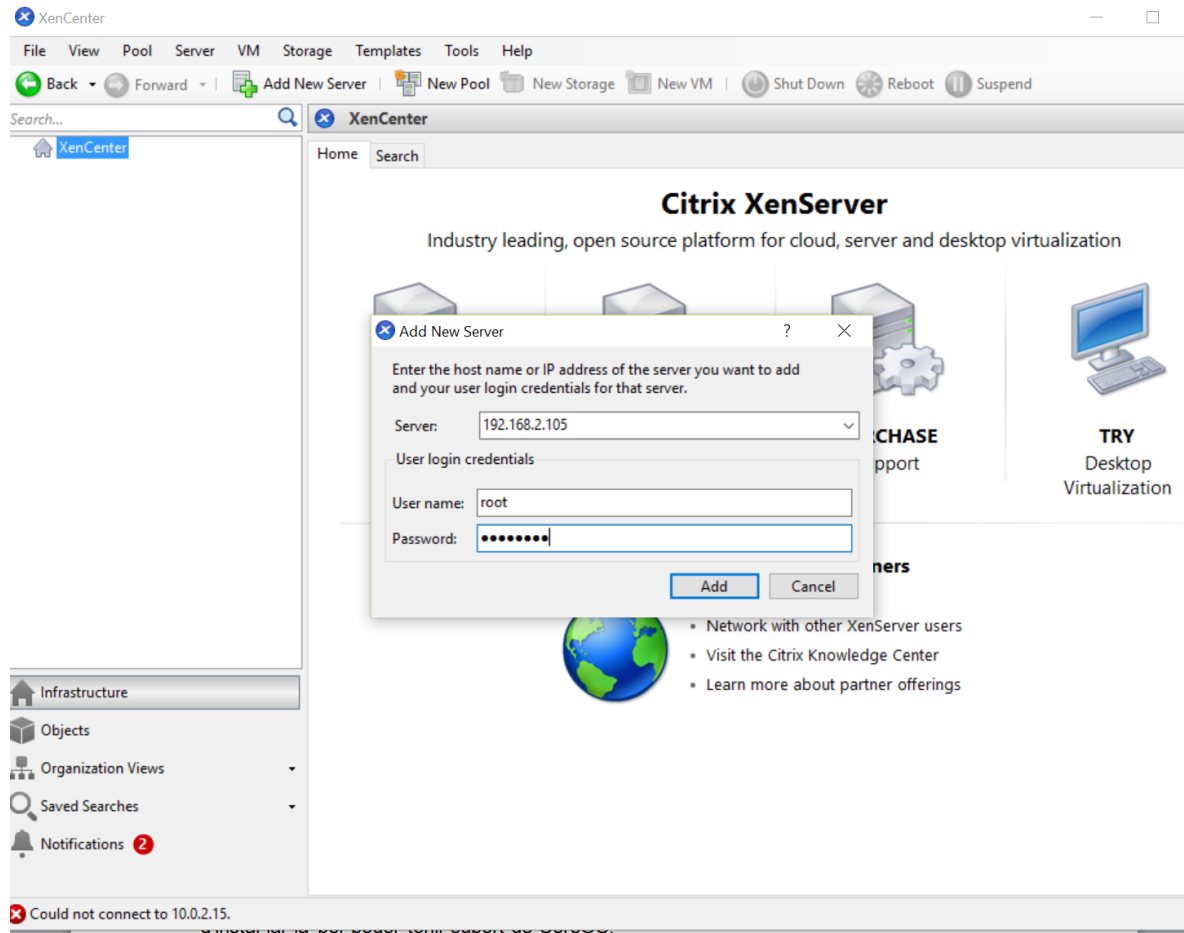


16. Un cop instal·lat tindrem una pantalla indicant la versió, ip, etc... i un menú on podrem configurar el nostre sistema, canviar ip, reiniciar, fer repositoris, etc...



Per accedir al Xen Server existeix una eina d'escriptori anomenada XenCenter que ens permetrà veure el nostre servidor, les màquines que tenim instal·lades, crear noves màquines, eliminar-les, en definitiva, gestionar el nostre entorn virtual.

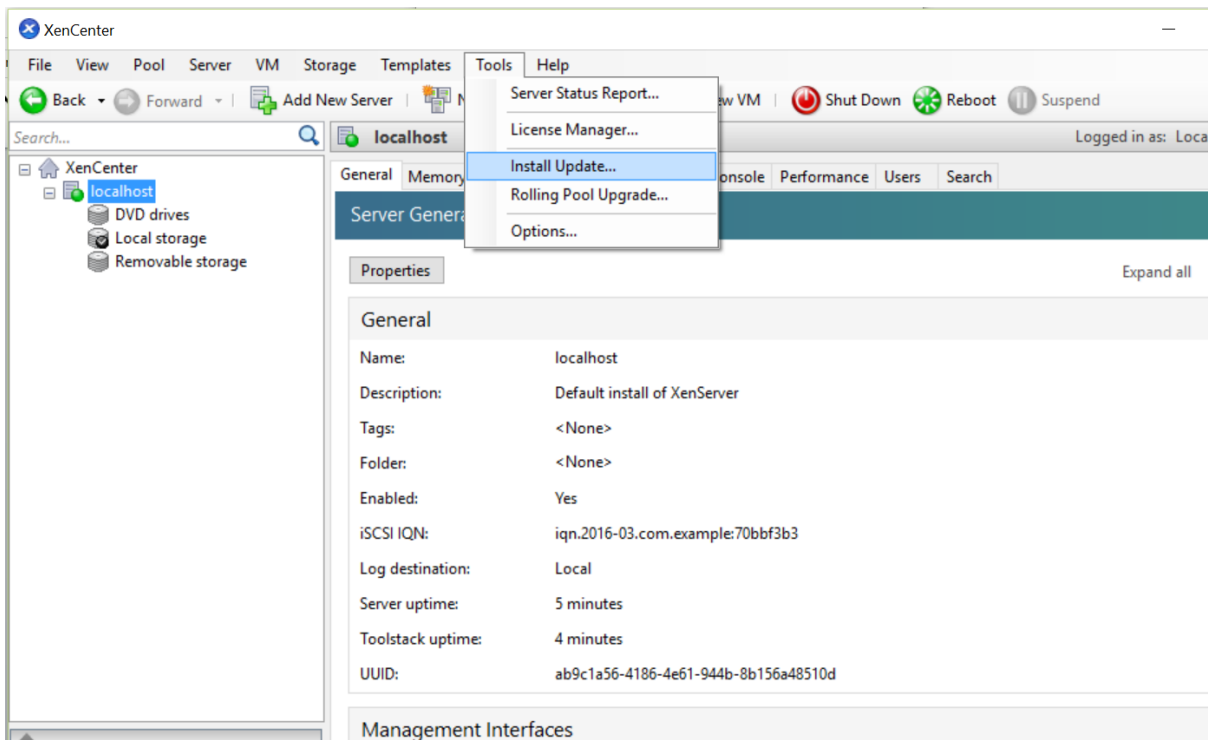
Xen Center el podem descarregar de la web de xen, només posant la ip del nostre servidor, en el nostre cas 192.168.2.105 i la contrasenya que hem posat a la instal·lació ja podem gestionar el nostre entorn.



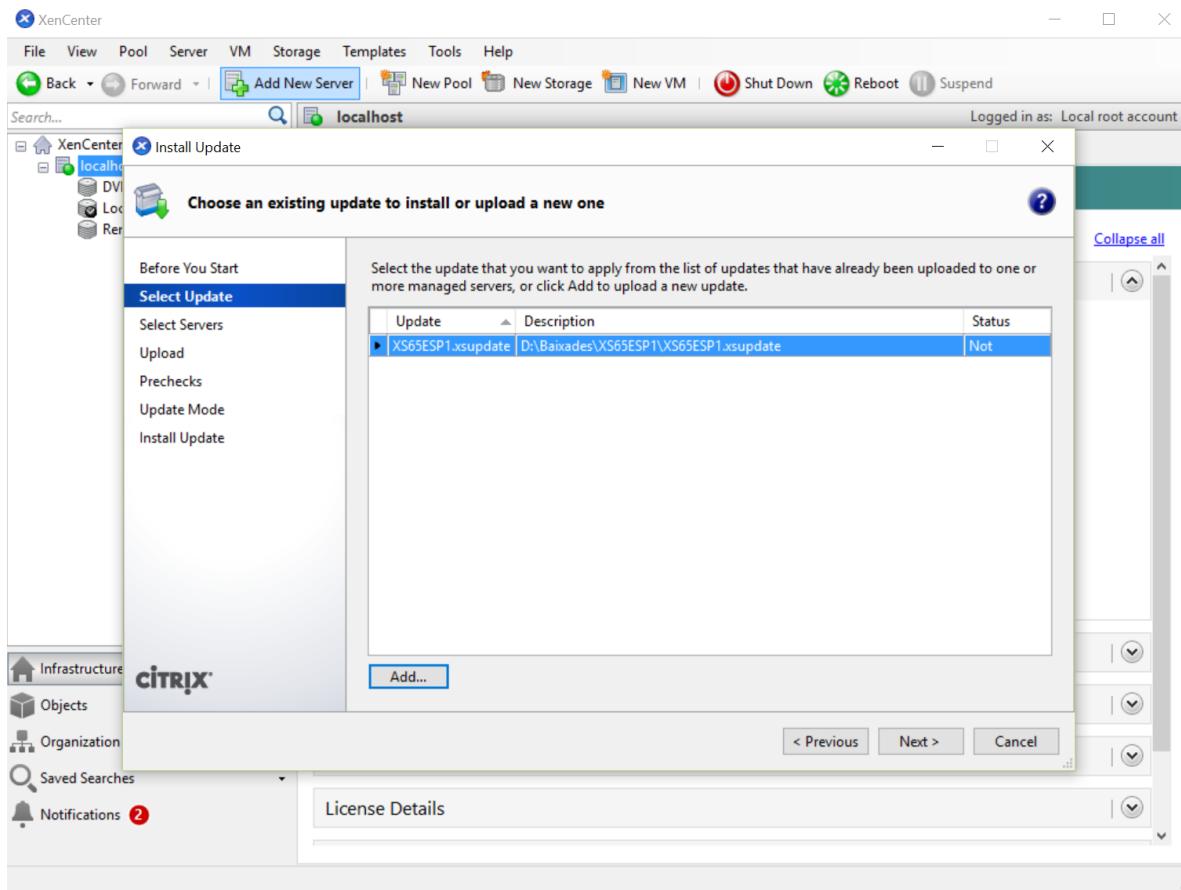
## 2. Instal·lació de SP1 a XenServer 6.5.

Ara que tenim el sistema instal·lat i podem crear màquines virtuals si intentem fer una màquina tipus CoreOS veiem que no existeix la plantilla. Això és degut a que Xen no va donar suport per CoreOS fins que no va treure el service pack 1 de la versió 6.5, haurem d'instal·lar-la per poder tenir suport de CoreOS.

Per instal·lar el sp1 de Xen l'hem de descarregar de la pàgina web del fabricant, [www.xenserver.org](http://www.xenserver.org), i un cop el tenim descarregat anem al XenCenter i cliquem a tools -> install Update.

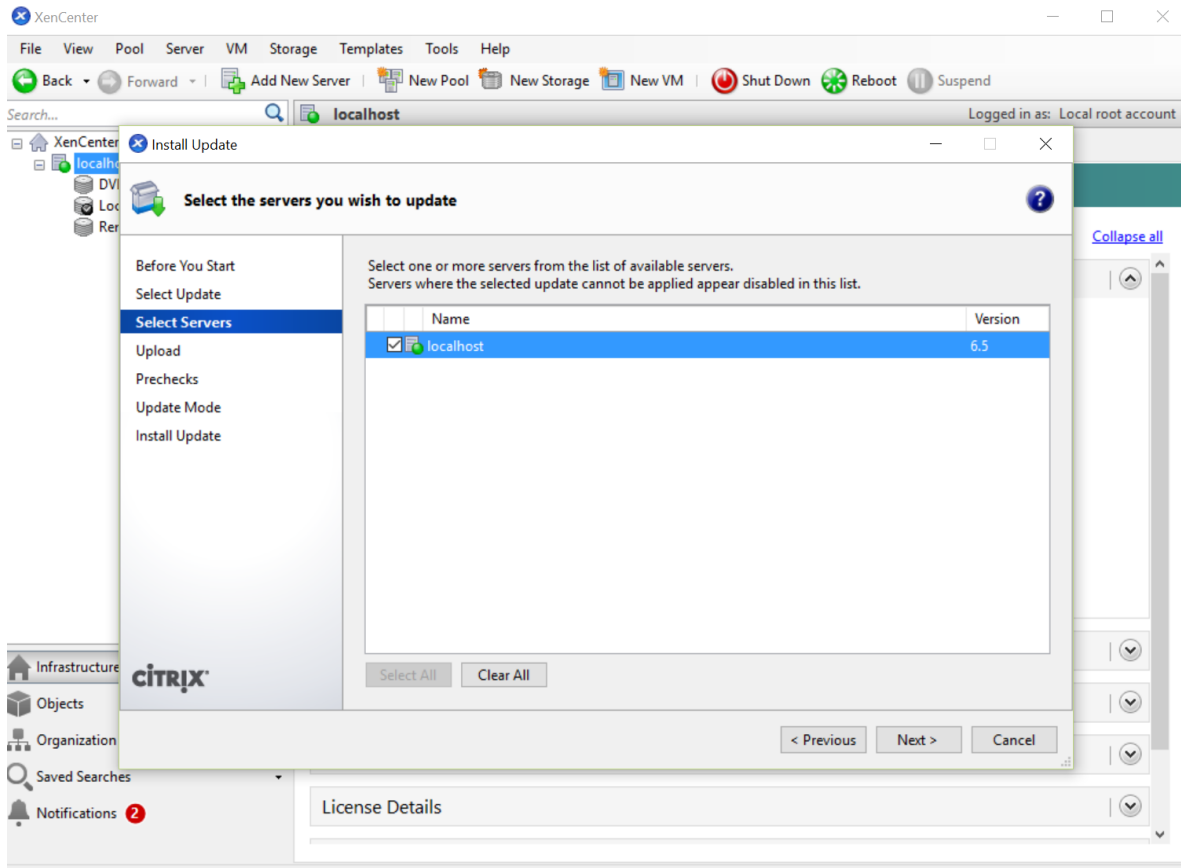


Seleccionem el fitxer del Sp1.

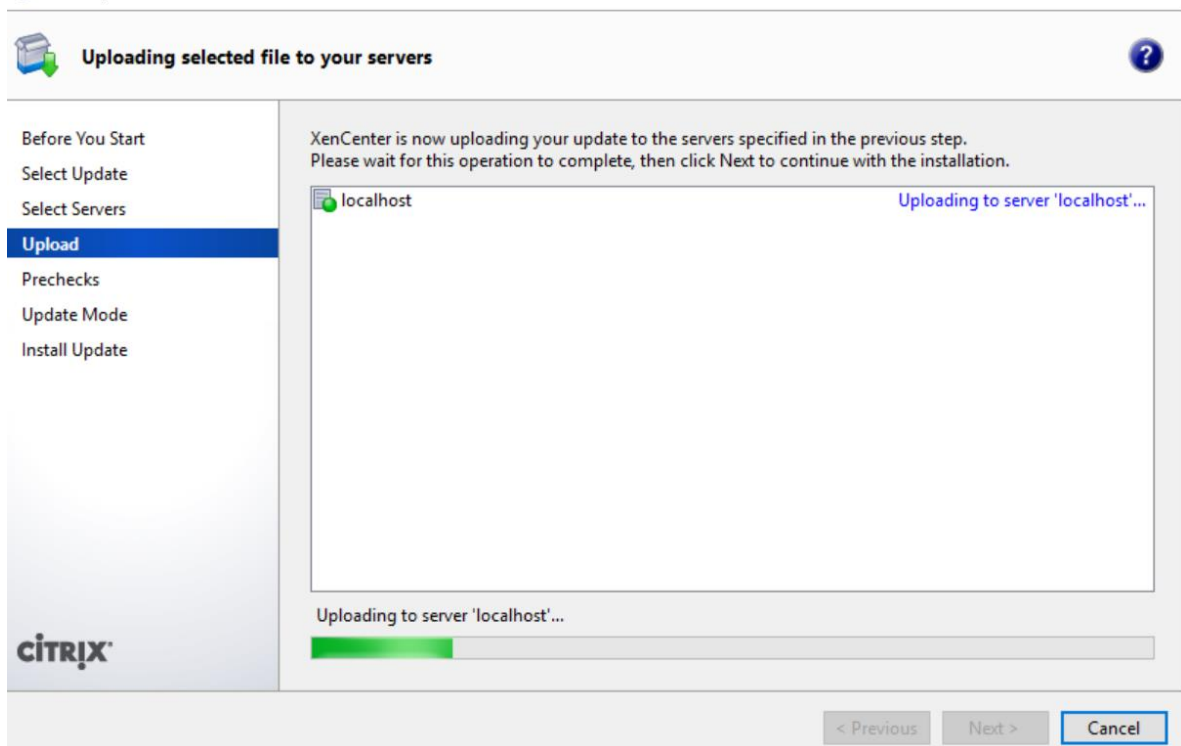


Seleccionem el servidor o servidors, en el cas que tinguem un clúster de Xenserver, on volem instal·lar el service pack.

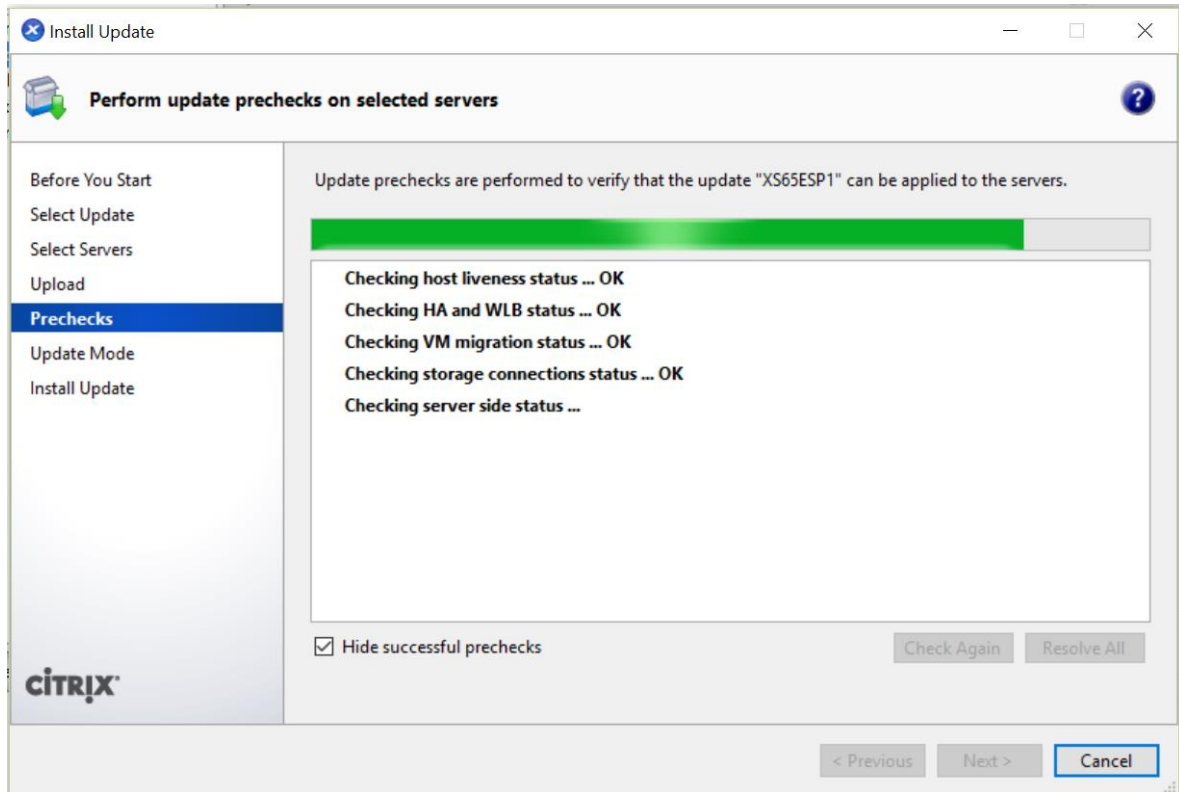




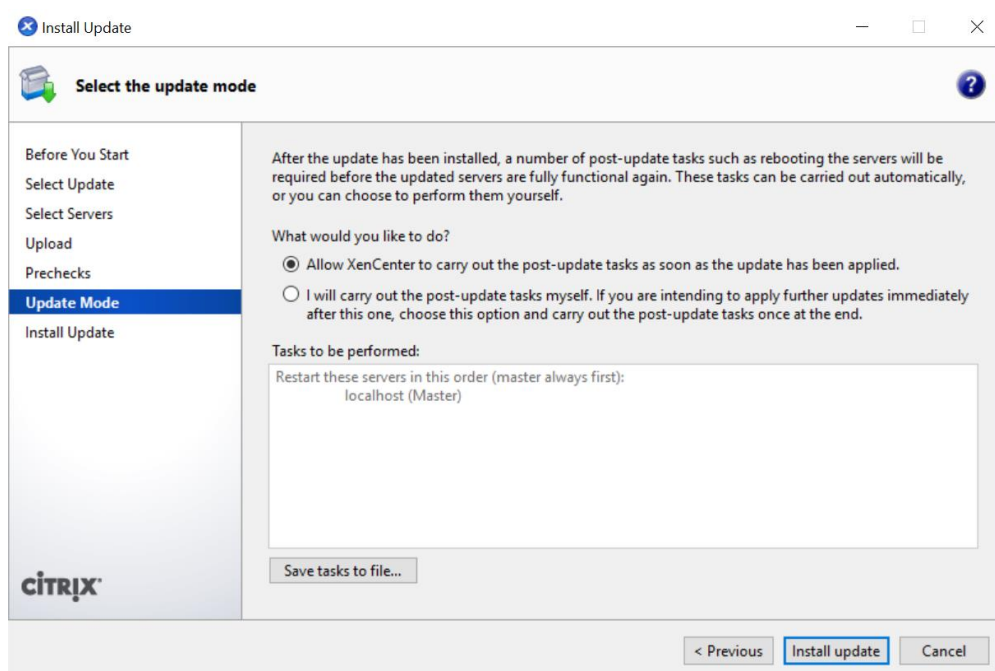
Es posa a treballar per actualitzar-ho, pujant el service pack al servidor.



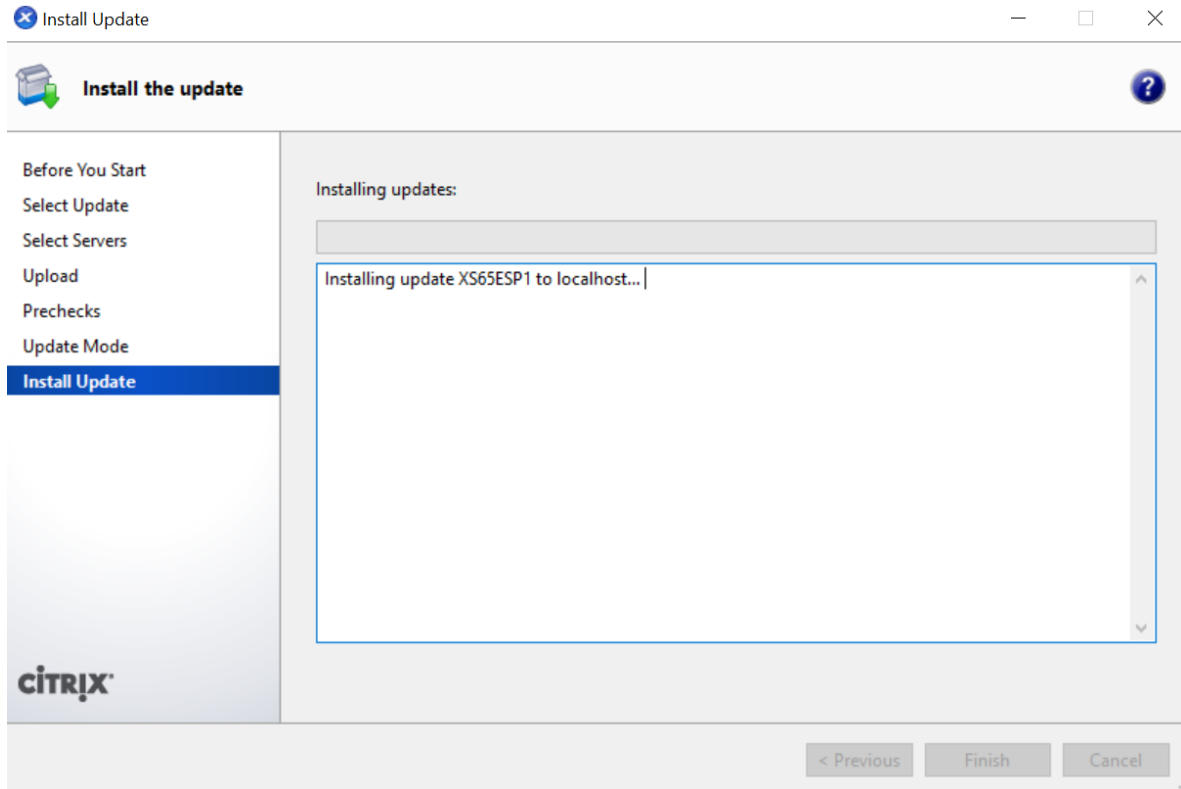
Comença a fer comprovacions pre-instal·lació, del host de l'alta disponibilitat, de les màquines virtuals instal·lades, dels magatzems de dades connectats, etc...



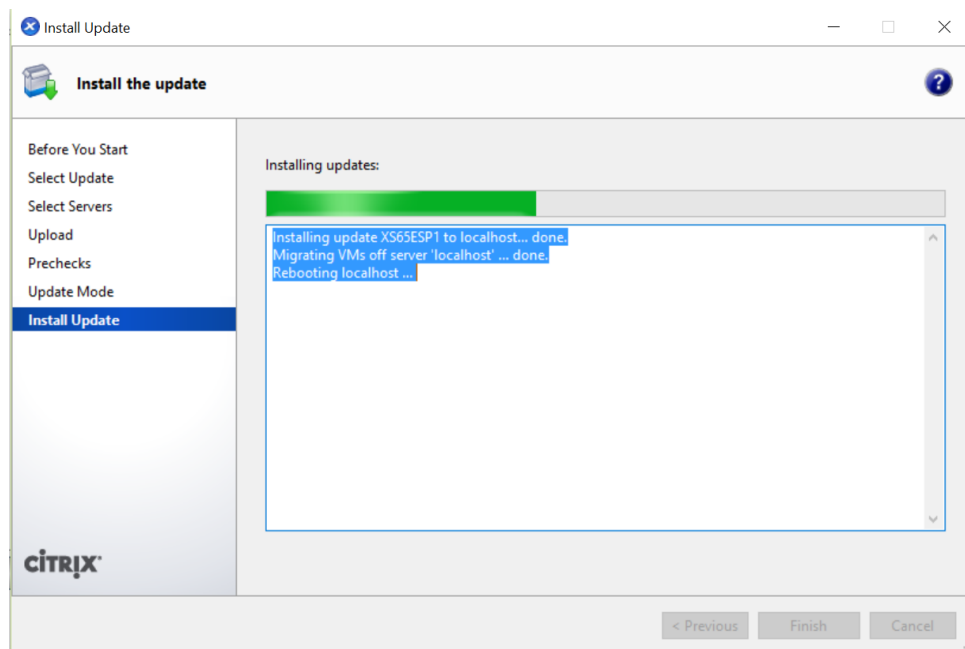
Posem els servidors en mode actualització i seleccionem l'opció que XenServer s'encarregui de fer les tasques post-update i seleccionem Install Update.



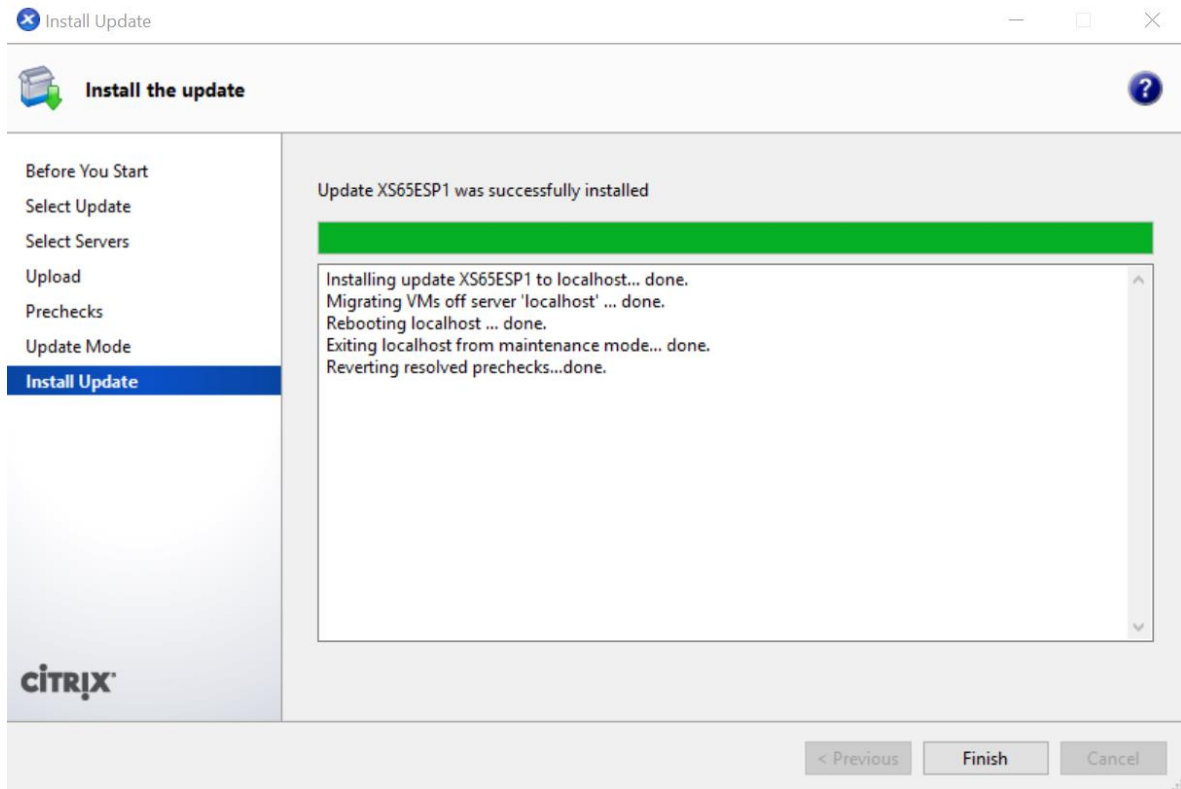
El sistema instal·la l'actualització i les plantilles dels nous sistemes operatius suportats, entre d'altres tenim CoreOs que és el que necessitem.



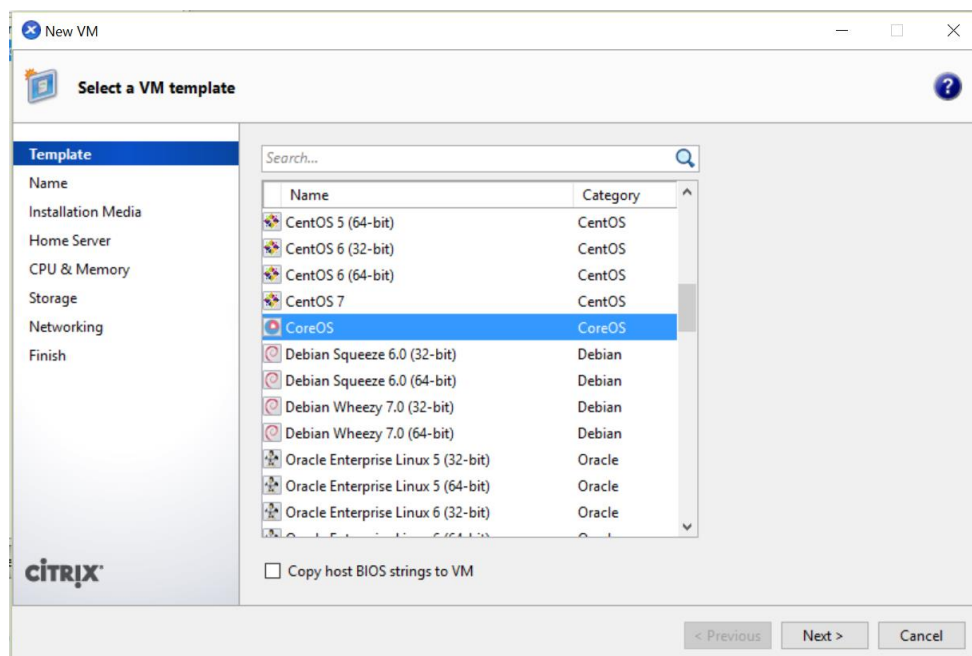
Al seleccionar que el mateix fes les tasques post-update, reinicia el servidor automàticament.



Després de reiniciar el servidor la instal·lació ha finalitzat.



I podem comprovar que tenim la plantilla que necessitem per fer les nostres màquines virtuals CoreOs i crear el nostre clúster.



### 3. Creació de repositori per emmagatzemar ISO'S.

Si volem instal·lar un sistema operatiu a una màquina virtual necessitem el CD d'instal·lació, anar cada vegada al servidor físic i ficar el CD, o podem fer un repositori local o a la xarxa on emmagatzemar totes les ISO'S dels sistemes operatius que volem instal·lar al nostre entorn virtual i cada vegada que fem una màquina nova podem accedir al repositori per connectar el cdrom virtual a les nostre ISO'S.

Per fer aquest repositor jo he optat per fer servir el repositori local. Per poder crear-ho hem de connectar-nos al servidor mitjançant SSH, amb la utilitat putty si fem desde Windows o Mac i directament de la Shell si ho fem des de Linux.

Creem el directori.

```

root@localhost:~
login as: root
root@192.168.2.105's password:

XenServer dom0 configuration is tuned for maximum performance and reliability.

Configuration changes which are not explicitly documented or approved by Citrix
Technical Support, may not have been tested and are therefore not supported. In
addition, configuration changes may not persist after installation of a hotfix
or upgrade, and could also cause a hotfix or upgrade to fail.

Third party tools, which require modification to dom0 configuration, or
installation into dom0, may cease to function correctly after upgrade or hotfix
installation. Please consult Citrix Technical Support for advice regarding
specific tools.

Type "xsconsole" for access to the management console.
[root@localhost ~]# mkdir /var/opt/isos
[root@localhost ~]#

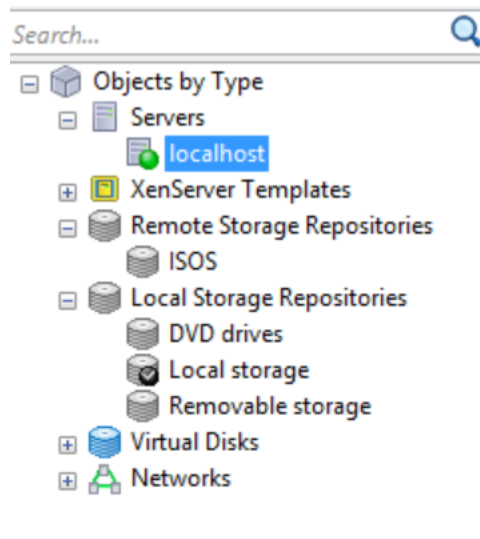
```

I ara ja podem crear el repositori perquè surti al XenCenter.

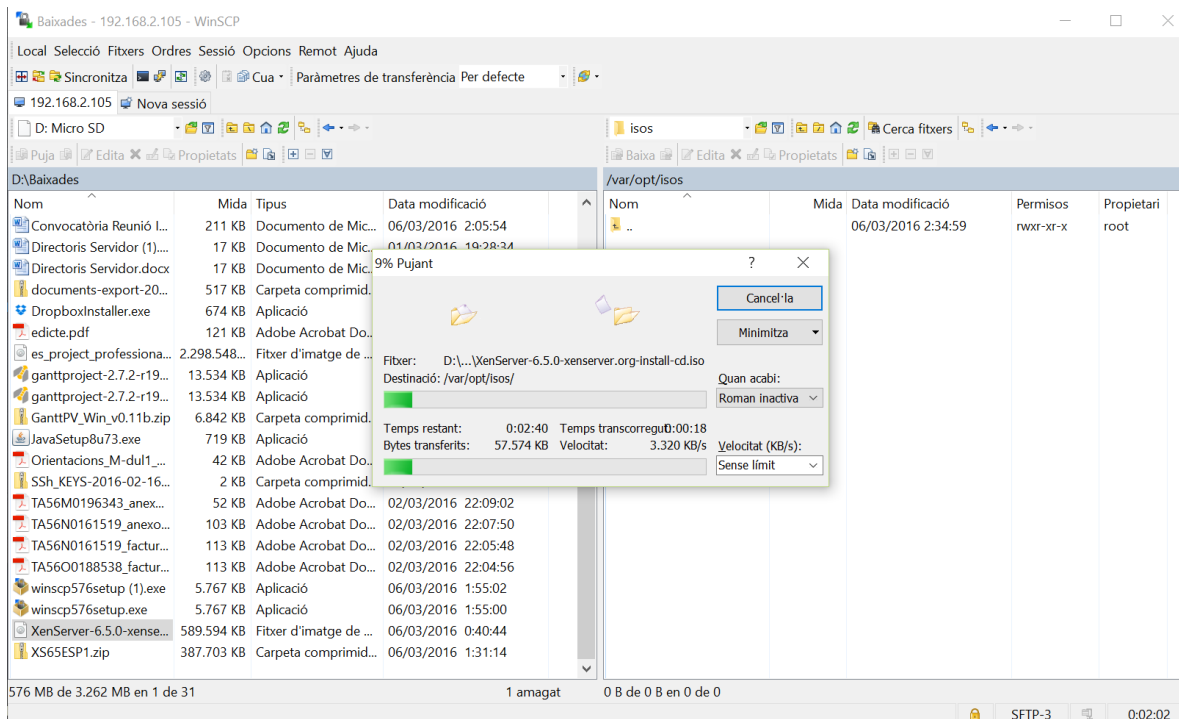
```
#xe sr-create name-label=ISOS type=iso device-config:location=/var/opt/isos device-
config:legacy_mode=true content-type=iso
```

```
9c89ada6-479b-322e-dd69-584b32dd47f3
```

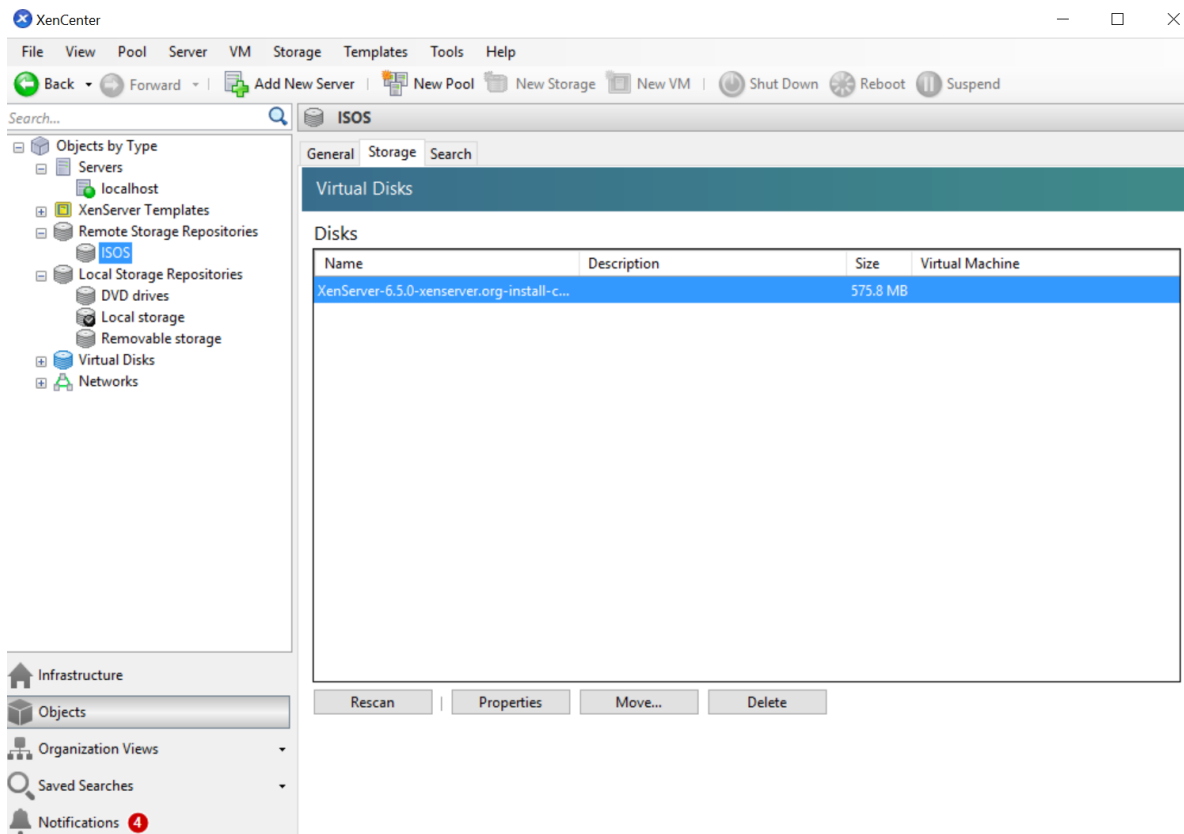
I ja tindrem el repositori creat que podrem veure des del XenCenter



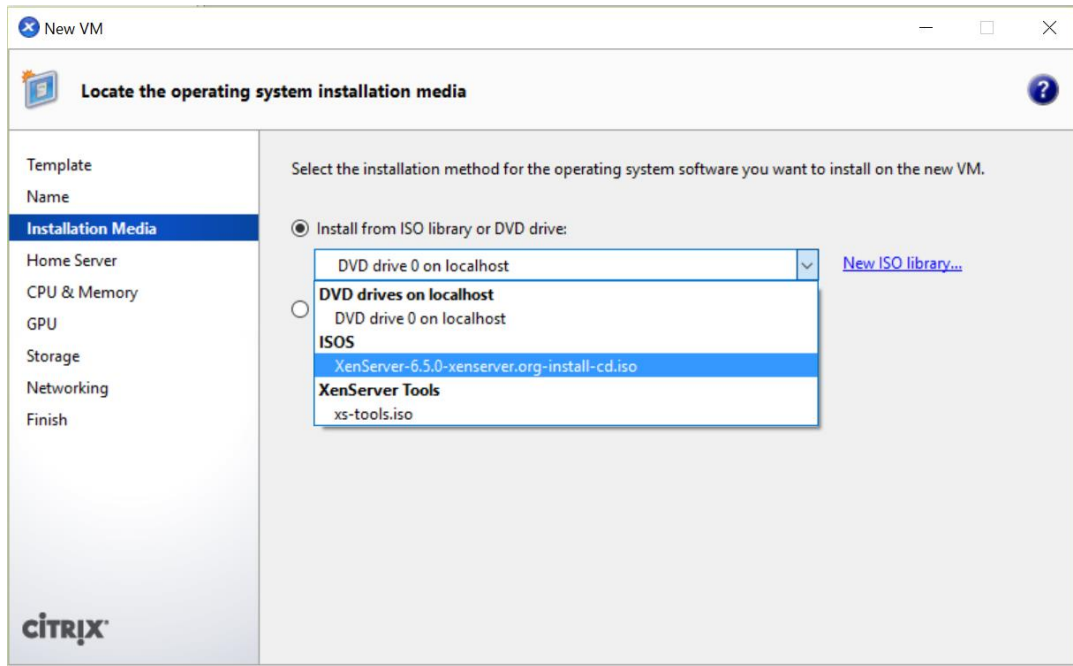
Ara amb el programa WinSCP o amb la comanda scp de Linux podrem pujar les isos al nostre repositori i les podrem fer servir per crear màquines virtuals.



Si anem al repositori i fem “rescan” podrem veure la imatge que hem pujat.



I la podrem fer servir per connectar a un cd d'una màquina virtual.



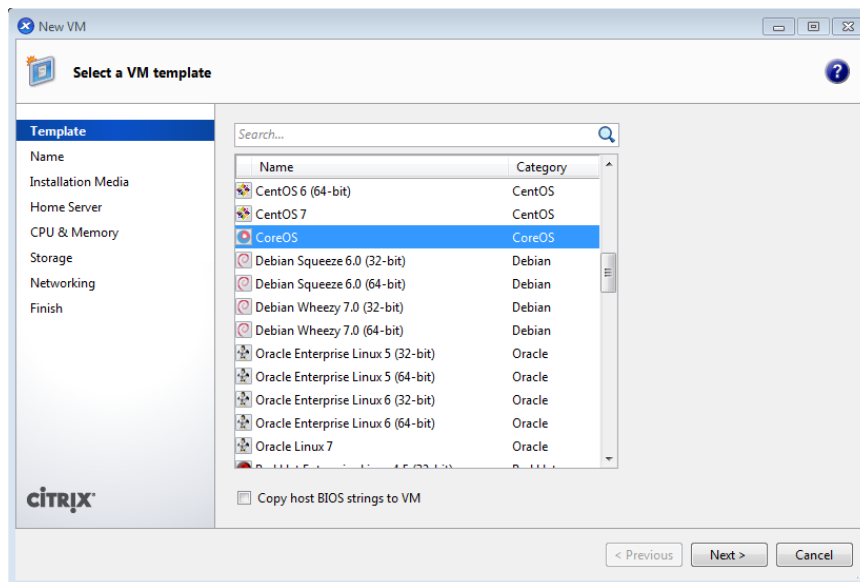


Índex

1. Creació de la màquina virtual a Xen Server.....	2
2. Configuració del fitxer Cloud Config .....	5
3. Instal·lació de CoreOS al disc dur.....	12
4. Comprovació d'estat del clúster.....	13

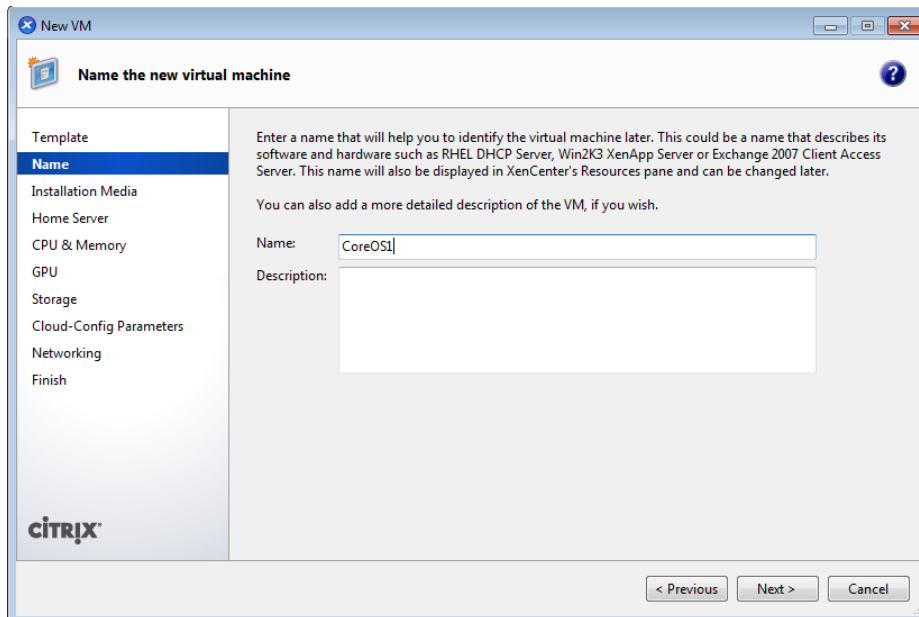
1. Creació de la màquina virtual a XenServer

Per crear la màquina virtual al XenServer hem de connectar-nos al servidor Xen mitjançant l'eina XenCenter i crear nova màquina virtual, gràcies a que s'ha instal·lat el SP1 de la versió 6.5 de XenServer tenim disponible la plantilla CoreOs.

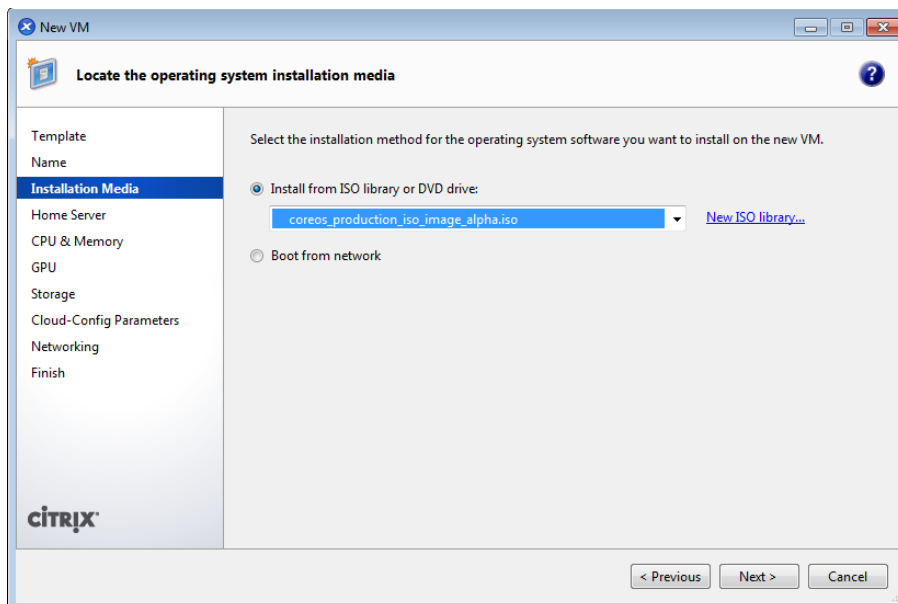


Indiquem el nom de la màquina, jo he posat CoreOS i el nº que tindrà al clúster, per a la primera màquina CoreOS1, la segona CoreOS2 i així successivament.

## Annex2 Instal·lació CoreOS

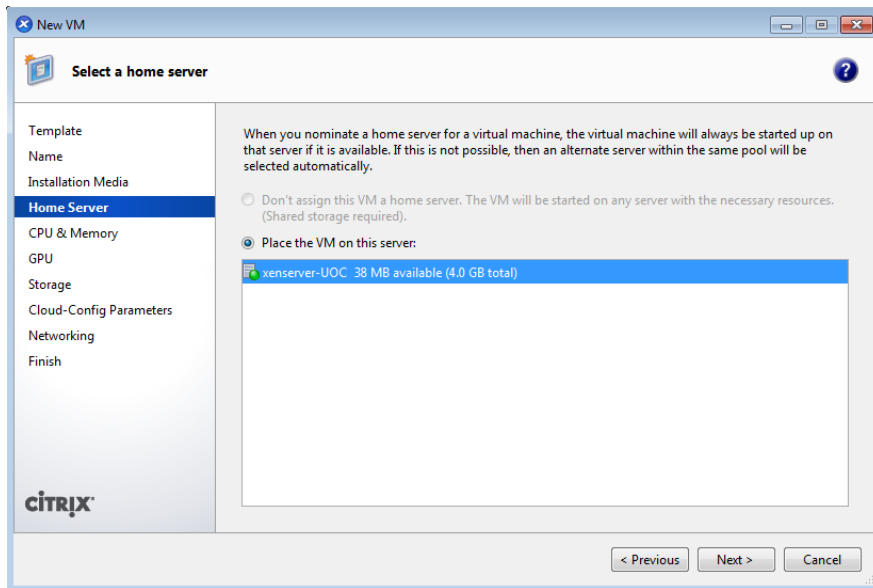


Seguidament seleccionem la iso que tenim emmagatzemada al nostre repositori que hem creat al Annex1 d'aquest document, la qual s'ha copiat mitjançant scp o si fem servir Windows o Mac mitjançant WinSCP, en aquest cas hem seleccionat la versió alpha, unes de les diferències d'aquesta versió és que fa servir etcd2 i no etcd.

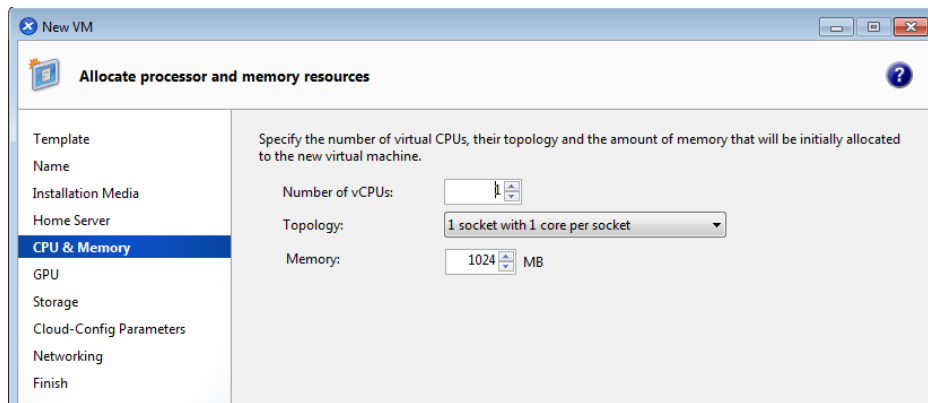


Seleccionem el servidor on serà executada la màquina virtual, en el nostre cas només en tenim un però podríem tenir un clúster de XenServers.

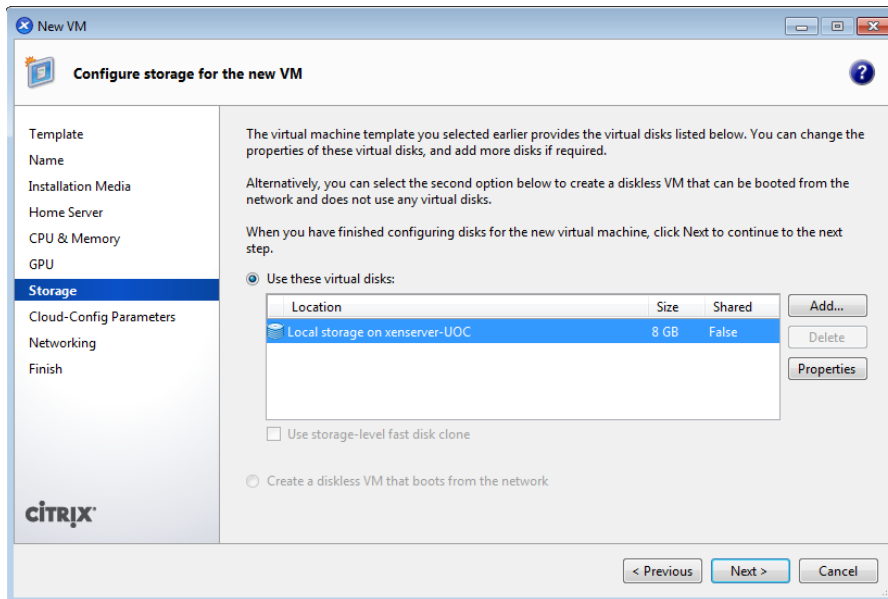
## Annex2 Instal·lació CoreOS



Seguidament hem de configurar la CPU i la memòria ram que tindrà la nostra màquina virtual.



El proper pas és configurar on estarà emmagatzemat el disc dur virtual de la nostra màquina. S'ha fet servir l'emmagatzemament intern del servidor però s'hauria de fer servir un emmagatzemament extern, com ara, una cabina de discos, un clúster de servidors nfs, etc... D'aquesta manera podríem tenir un clúster de servidors XenServer i si un fallés els altres aixecarien les nostres màquines virtuals.



S'ha de tenir en compte que el disc dur virtual de la màquina CoreOS ha de tenir un mínim de tamany de 8 GB, si no té aquesta mida el sistema operatiu no s'instal·larà.

## 2. Configuració del fitxer Cloud Config

El fitxer de configuració Cloud-Config és el fitxer que llegeix el sistema operatiu CoreOS cada vegada que arrenca el sistema, és on configurem el comportament que tindrà, configurant quin serveis s'iniciaran i en quins ports escoltarà, etc...

Aquest fitxer el pots crear manualment quan arrenca el sistema operatiu des de la ISO i fer que l'utilitzi quan executem l'ordre d'instal·lació al disc dur.

```
# sudo coreos-install -d /dev/sda -C alpha -c cloud-config-file
```

Taula 1: Instal·lació de CoreOS directament a un servidor.

## Annex2 Instal·lació CoreOS

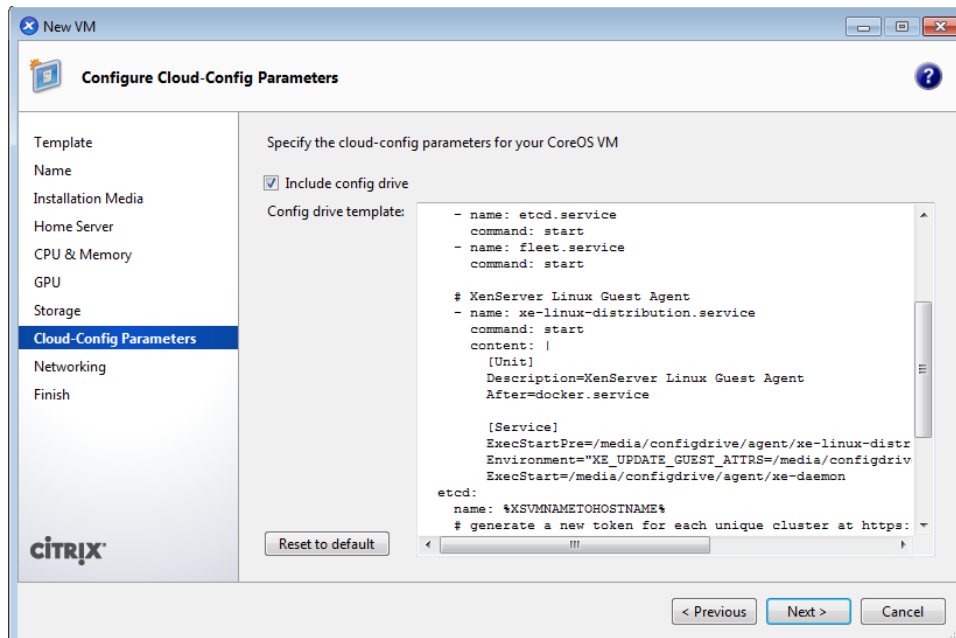
O com en el cas d'aquest projecte XenServer et dona una plantilla i l'emmagatzema en un "Cloud-Drive", amb aquesta opció no hem de dir a la instal·lació on es troba el nostre fitxer, s'encarrega de fer-ho Xen.

```
# sudo coreos-install -d /dev/xvda --o xen --C alpha
```

Taula 2: Instal·lació de CoreOS a XenServer

El fitxer Cloud-config és el fitxer més important de CoreOS. És per aquest motiu que explicarem pas per pas quines parts s'han de configurar per arribar a l'objectiu d'aquest projecte.

Si mirem l'estructura del fitxer que ens proporciona Xen veurem que hem de configurar una sèrie de paràmetres:



```

#cloud-config

hostname: %XSVMNAMETOHOSTNAME%
ssh_authorized_keys:
  # - ssh-rsa <Your public key>
  # The following entry will automatically be replaced with a public key
  # generated by XenServer's container management. The key-entry must exist,
  # in order to enable container management for this VM.
  - ssh-rsa %XSCONTAINERRSAPUB%

coreos:
  units:
    - name: etcd.service
      command: start
    - name: fleet.service
      command: start

  # XenServer Linux Guest Agent
  - name: xe-linux-distribution.service
    command: start
    content: |
      [Unit]
      Description=XenServer Linux Guest Agent
      After=docker.service

      [Service]
      ExecStartPre=/media/configdrive/agent/xe-linux-distribution /var/cache/xe-linux-distribution
      Environment="XE_UPDATE_GUEST_ATTRS=/media/configdrive/agent/xe-update-guest-attrs"
      ExecStart=/media/configdrive/agent/xe-daemon

  etcd:
    name: %XSVMNAMETOHOSTNAME%
    # generate a new token for each unique cluster at https://discovery.etcd.io/new
    # discovery: https://discovery.etcd.io/<token>

  write_files:
    # Enable ARP notifications for smooth network recovery after migrations
    - path: /etc/sysctl.d/10-enable-arp-notify.conf
      permissions: 0644
      owner: root
      content: |
        net.ipv4.conf.all.arp_notify = 1

# Template loaded from /usr/lib/python2.4/site-packages/xscontainer/data/cloud-config.template

```

El primer pas és assegurar-nos de que una vegada instal·lem el CoreOS al disc dur local podrem accedir-hi per treballar. Aquesta part requereix crear una parella de claus, pública i privada, per xifrar les connexions i garantir que som nosaltres els que estem accedint.

Per crear la parella de claus la fem des del nostre sistema, de d'on accedirem a cadascun dels CoreOS.

Per a crear la parella de claus executem les següents comandes.

```
$ ssh-keygen -t rsa -b 2048 -C email@domini.com

#Ens demanarà el nom del fitxers de claus, per defecte es diuen id_rsa i id_rsa.pub
Enter a file in which to save the key (/home/usuari/.ssh/id_rsa): [Press enter]

#També ens demana la contrasenya per la clau privada
Enter passphrase (empty for no passphrase): [Type a passphrase]
Enter same passphrase again: [Type passphrase again]

# I ens dona el nostre key fingerprint
nsdlfsdfEljsjfldsfjdEljkdlsdkfjdsfidsifduutT dkfj= email@domini.com
```

Ens demana quin nom volem donar-li al fitxer, per defecte és id\_rsa per a la clau privada i id\_rsa.pub per a la clau pública.

I per acabar ens demanarà la contrasenya de la clau privada, les claus s'emmagatzemen a la carpeta /home/usuari/.ssh . Aquests fitxers els hem de guardar per no perdre'ls ja que són les claus d'accés al nostre clúster.

Un cop tenim la parella de claus hem de configurar la nostra clau pública al fitxer Cloud-Config per treure la clau pública hem de fer un cat del fitxer, copiar la sortida al fitxer Cloud\_config.

```
# cat /home/usuari/.ssh/id_rsa.pub
```

La sortida de la comanda és una cosa així:

```
[root@xenserver-UOC ~]# cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAAnYS1USanI8NIEYutR+yZGW6knRjXkX0PNsfQNYRMtR4n
K4G1XLE9U2/LZ2MvVLJYm9Ws9wiv7chs1Emuhly8bBrKnmHx4k5sp+tIIL82EDJzeD6VyX4WlpBhRk7E
7HVoNCetBYcqTTnzSBBvDiXa                          byd0RMcQ/hTocsEr0uBpPW93bfEWbf8eBveZO
44Q85RrwBZj2jt05RQBxK1IJkwzez5GCQeP3I1UUiCdtUBjjsRYpV0hg68jMQYXUzu9rvKFI46+ds8pC
TKS6j76xc436zAkc09xxmgmpuILVuc81PBo60c2v3IxJ7Da7DnoPbIXVw== root@xenserver-UOC
[root@xenserver-UOC ~]#
```



Hem de copiar des de ssh-rsa ..... fins root@xenserver-UOC, o el nostre correu electrònic al fitxer de configuració del XenServer.

```
hostname: %XSVMMNAME%TOHOSTNAME%
ssh_authorized_keys:
  - ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA... ..
```

Hem de treure el coixinet “#” que fa que la línia sigui un comentari i enganxar la nostra clau pública, podem veure a sobre que hostname té una variable, això ho deixarem tal i com està, ja que XenServer assignarà el mateix nom que s’ha donat a la màquina virtual, aquesta configuració haurà de ser manual si optem per instal·lar-lo manualment.

També podem veure a l’apartat de ssh\_authorized\_keys la línia - ssh-rsa %XSCONTAINERRSAPUB%

Aquesta línia també la deixarem com està ja que serveix perquè el nostre servidor XenServer pugui monitoritzar els contenidors que estem executant al nostre clúster.

Continuem configurant el fitxer i ens trobem l’apartat d’units. Aquí configurarem els serveis que necessitem que s’executin a l’iniciar el sistema. En el nostre cas farem servir etcd2 i fleet, com hem descrit a la memòria etcd2 farà el descobriment dels CoreOS que tinguem corrent amb el mateix identificador de clúster i fleet ens ajudarà a balancejar i tenir aixecats els contenidors que iniciarem al clúster.

Haurem de substituir la línia que posa etcd.service per la nostra versió de etcd2.

```
coreos:
  units:
    - name: etcd2.service
      command: start
    - name: fleet.service
      command: start
```

Només ens quedarà la part d'etcd i fleet per configurar, els ports per on estaran escoltant i els serveis que desplegaran.

```
etcd2:
  name: %XSVMNAMETOHOSTNAME%
  # generate a new token for each unique cluster at https://discovery.etcd.io/new
  discovery: https://discovery.etcd.io/d8582c076452aaa599be502
  # private networking need to use $public_ipv4:
  advertise-client-urls: http://\$public\_ipv4:2379
  initial-advertise-peer-urls: http://\$public\_ipv4:2380
  # listen on the official ports 2379, 2380 and one legacy port 4001:
  listen-client-urls: http://0.0.0.0:2379,http://0.0.0.0:4001
  listen-peer-urls: http://\$public\_ipv4:2380,http://10.50.0.211:7001
fleet:
  public-ip: $public_ipv4 # used for fleetctl ssh command
```

Una part molt important la trobem a la línia 4 de l'anterior taula és el servei discovery, aquest servei és el que fa servir etcd i etcd2 per descobrir els clústers. Quan volem crear un nou clúster hem d'accedir a la pàgina web <https://discovery.etcd.io/new?size=3> on 3 és el número de màquines que tindrem al clúster, si no posem ?size=3 ens farà un identificador de 3 màquines igualment, és el nombre de màquines que fa per defecte.

Podrem modificar el nombre de màquines segons les necessitats que tinguem, la pàgina web ens retorna un identificador que hem d'introduir a la línia discovery, perquè etcd pugui descobrir nodes a la xarxa.

En el nostre cas tenim: discovery: <https://discovery.etcd.io/d8582c076452aaa599be502>

Ja tenim el Cloud-Config configurat per a les nostres necessitats, aquest fitxer serà el mateix per a tots els sistemes que tenim, és a dir guardem la configuració en un fitxer a part i la podrem copiar quan generem les màquines noves. El fitxer ha de quedar d'aquesta manera:

```
#cloud-config
hostname: %XSVMNAMETOHOSTNAME%
ssh_authorized_keys:
  - ssh-rsa
  AAAAB3NzaC1yc2EAAAABIwAAAQEAnYS1USanl8NIEYutR+yZGW6knRjXkXOPNsfQNYRMtR4nK4
  GIXLE9U2/+ds8pCTKS6j76xc436zAkcO9xxmgmppuLVuc8IPBo60c2v3lxJ7Da7DnoPbIXVw==
  root@xenserver-UOC
  # The following entry will automatically be replaced with a public key
  # generated by XenServer's container management. The key-entry must exist,
  # in order to enable container management for this VM.
  - ssh-rsa %XSCONTAINERRSAPUB%
coreos:
  units:
    - name: etcd2.service
      command: start
    - name: fleet.service
      command: start
    # XenServer Linux Guest Agent
    - name: xe-linux-distribution.service
      command: start
      content: |
        [Unit]
        Description=XenServer Linux Guest Agent
        After=docker.service
        [Service]
        ExecStartPre=/media/configdrive/agent/xe-linux-distribution /var/cache/xe-linux-
distribution
        Environment="XE_UPDATE_GUEST_ATTRS=/media/configdrive/agent/xe-update-
guest-attrs"
        ExecStart=/media/configdrive/agent/xe-daemon
    etcd2:
      name: %XSVMNAMETOHOSTNAME%
      # generate a new token for each unique cluster at https://discovery.etcd.io/new
      discovery: https://discovery.etcd.io/d8582c076452aaa599be502
      # private networking need to use $public_ipv4:
      advertise-client-urls: http://\$public\_ipv4:2379
      initial-advertise-peer-urls: http://\$public\_ipv4:2380
      # listen on the official ports 2379, 2380 and one legacy port 4001:
      listen-client-urls: http://0.0.0.0:2379,http://0.0.0.0:4001
      listen-peer-urls: http://\$public\_ipv4:2380,http://\$public\_ipv4:7001
```

```
fleet:
  public-ip: ·$public_ipv4 # used for fleetctl ssh command
write_files:
  # Enable ARP notifications for smooth network recovery after migrations
  - path: /etc/sysctl.d/10-enable-arp-notify.conf
    permissions: 0644
```

### 3. Instal·lació al disc dur.

Per instal·lar aquest sistema operatiu només ens cal una comanda, el sistema té un script anomenat coreos-install que descarrega la versió que li passem per paràmetre i la instal·la al disc dur que li diem. Per veure el codi font d'aquest script el podeu veure al següent repositori github <https://github.com/coreos/init/blob/master/bin/coreos-install> .

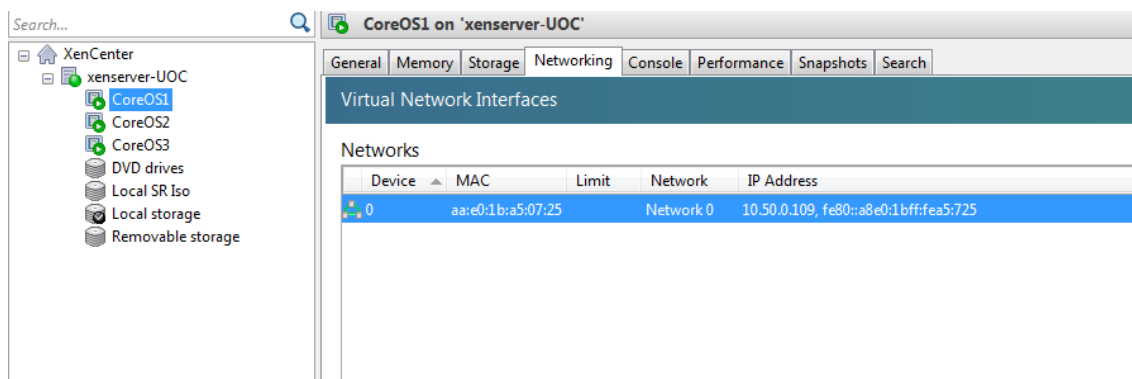
```
# sudo coreos-install -d /dev/xvda -o xen -C alpha
```

Com es pot veure executem la comanda amb permisos de root.

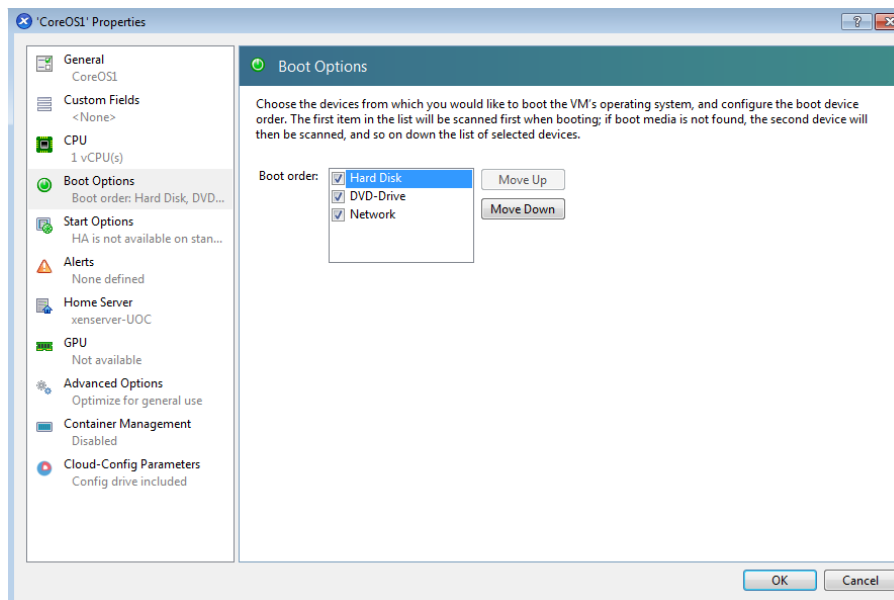
Li passem per paràmetre la unitat on instal·larem el sistema amb el flag -d.

I li diem quina versió volem instal·lar, en el nostre cas serà la versió alpha, però podria ser beta o estable.

Per acabar configurem la xarxa, podem seleccionar les targetes que tenim físiques al servidor i configurar les ip's dinàmiques o estàtiques. Sempre podrem veure quina ip té assignat el node des de la pestanya networking del XenCenter.



4. Un cop tinguem el sistema instal·lat al disc dur ja podem reiniciar el sistema. Abans de reiniciar, configurar la màquina perquè arrenqui des del disc dur i no des del CD.



La instal·lació s'ha de fer amb totes les màquines/nodes que componen el nostre clúster, en el cas que ens ocupa són 3. Un cop tinguem engegades totes les màquines podrem accedir a elles.

La manera que podem accedir a les màquines és accedint al XenServer per SSH i des d'allà accedir mitjançant la següent comanda.

```
Type "xsconsole" for access to the management console.
[root@xenserver-UOC ~]# ssh core@10.50.0.109 -i id_rsa
Last login: Mon Feb 22 21:56:52 2016 from 10.50.0.250
CoreOS alpha (976.0.0)
core@coreos1 ~ $
```

Com podem veure a la imatge anterior fem servir la comanda ssh i li diem que volem accedir amb l'usuari core fent core@ip\_servidor i amb el flag -i li diem que volem fer servir la nostra clau privada, la que hem fet al començament de l'annex 1 .

Ara que estem a un dels nodes del clúster podem mirar quins nodes estan donats d'alta amb la comanda fleetctl list-machines, que ens mostrarà els nodes del clúster.

## Annex2 Instal·lació CoreOS

```
core@coreos1 ~ $ fleetctl list-machines
MACHINE      IP           METADATA
04b97016...  10.50.0.211 -
44a02419...  10.50.0.109 -
7e0b0ae0...  10.50.0.18  -
core@coreos1 ~ $
```

Índex

1. Creació de Contenidor “Docker” apache.....	2
2. Creació de servei Fleet per contenidor.....	3
3. Execució del Contenidor.....	4
4. Proves Alta disponibilitat.....	6

### 1. Creació de Contenidor “Docker” Apache

Docker té un repositori d’imatges creades les quals podem modificar i tornar a empaquetar per personalitzar-les; les imatges disponibles són molt diverses.

Per crear un contenidor amb Docker i CoreOS el primer que hem de fer és iniciar el contenidor amb la imatge que vulguem modificar, instal·lar, configurar el que necessitem i tornar a empaquetar-la. En el nostre cas per fer les proves he fet servir una imatge d’ubuntu, he instal·lat l’apache i s’ha tornat a empaquetar la imatge per fer-la servir tantes vegades com necessitem.

```
# docker run -t -i ubuntu /bin/bash
```

Els flags `-t` i `-i` ens permeten fer servir el contenidor com una màquina virtual assignant-li un `tty` en la Shell que executem la comanda. Podem veure que fem servir la imatge `ubutu`, la qual la descarregarà del repositori de `docker` i executarà la comanda `/bin/bash` per poder interactuar amb el contenidor.

Un cop tenim la màquina engegada amb el prompt, instal·lem el servidor `apache`.

```
# apt-get install apache2
```

Ara tenim un contenidor amb `ubuntu` i `apache` instal·lat, el que hem de fer és crear una imatge d’aquest contenidor. Existeixen imatges ja creades amb `apache`, `mysql` i fins i tot `wordpress` instal·lat, però he fet aquest annex per veure com podem personalitzar una imatge.

Per veure quin identificador té el nostre contenidor que acabem d’iniciar hem de executar:

```
# docker ps
```

Aquesta s’ha d’executar a fora del contenidor, és a dir al nostre CoreOS. Per fer-ho podem obrir un altre Shell al node de CoreOS i ens retornarà l’ID del contenidor `ubuntu`.

```
# docker commit "identificador" usuari/apache
```



Amb aquesta comanda estem desant les modificacions que hem fet de la imatge docker d'ubuntu a una nova imatge anomenada usuari/apache. Ja tenim una imatge d'un contenidor que executarà un apache cada cop que l'executem.

## 2. Creació de servei Fleet per contenidor

Per comprovar l'alta disponibilitat del nostre clúster fent servir fleet, hem de crear un servei que executi fleet amb el docker que acabem de crear. El servei és un document on configurem quin gestor de contenidors fem servir, quina imatge executarem, quina configuració farà servir aquesta imatge, etc...

```
core@coreos1 ~ $ cat apache.service
[Unit]
Description=My Apache Frontend
After=docker.service
Requires=docker.service

[Service]
TimeoutStartSec=0
ExecStartPre=/usr/bin/docker kill apache%i
ExecStartPre=/usr/bin/docker rm apache%i
ExecStartPre=/usr/bin/docker pull coreos/apache
ExecStart=/usr/bin/docker run --rm --name apache%i -p 80%i:80 coreos/apache /usr/sbin/apache2ctl -D FOREGROUND
ExecStop=/usr/bin/docker stop apache%i

[X-Fleet]
#Conflicts=apache@*.service
core@coreos1 ~ $
```

Il·lustració 1. Annex 3 Servei Apache Fleet

Anem a comentar el fitxer de configuració del servei. A la secció [Unit] configurem el nom del servei i el gestor de contenidors que farà servir, en aquest cas Docker.

A la secció [Service] configurem el temps que trigarà en arrencar el contenidor, la comanda que executarà abans de crear el contenidor, podem veure que mata qualsevol contenidor que tingui el mateix nom que el que estem creant. Això és molt útil perquè si tenim un contenidor que executa apache i configurem fleet perquè faci comprovacions del port 80, per exemple, per saber si aquell apache està operatiu i no el troba, torna a executar el contenidor i mata el que no funciona.

Veiem que la comanda ExecStart executa run de docker, inicia el contenidor, assignant-li el nom, el port que farà servir, quina imatge utilitzarà i quina ordre executarà quan iniciï el contenidor, en aquest cas apache en foreground.

Veiem comentada la línia [Conflits=apache@\\*.service](#) perquè hem posat que el port per on escoltarà l'apache serà 80% i la variable vindrà donada pel número que posarem al crear el contenidor. Si per exemple fem:

```
# fleetctl start apache@1.service
```

Fleet crearà un contenidor anomenat apache1 i que estarà escoltant pel port 8081, d'aquesta manera podem posar més d'un contenidor amb apache al mateix node del clúster perquè cada apache escoltarà per un port diferent.

Si no es fes servir aquesta variable, només podríem tenir un apache per node, ja que només tenim un port 80 i hauríem de descomentar l'última línia que diu [Conflits=apache@\\*.service](#).

### 3. Execució del Contenedor

Per comprovar l'alta disponibilitat llencem un contenidor amb apache amb fleet i mirem a quina màquina s'està executant.

```
# fleetctl list-machines
```

Amb aquesta comanda veiem quines màquines del node estan detectant i gestionant fleet, aquestes són les màquines on fleet llançarà els contenidors tenint en compte la seva càrrega de treball. Aquesta comanda la podem executar a qualsevol node del clúster.

```
core@coreos1 ~ $ fleetctl list-machines
MACHINE      IP           METADATA
04b97016...  10.50.0.211 -
44a02419...  10.50.0.109 -
7e0b0ae0...  10.50.0.18  -
```

**Il·lustració 2 Annex 3. Llista màquines al clúster.**

```
# fleetctl start apache@1.service
```

Aquesta comanda ens llançarà un apache a un dels tres nodes del clúster i estarà escoltant pel port 801.

```
# fleetctl list-units
```

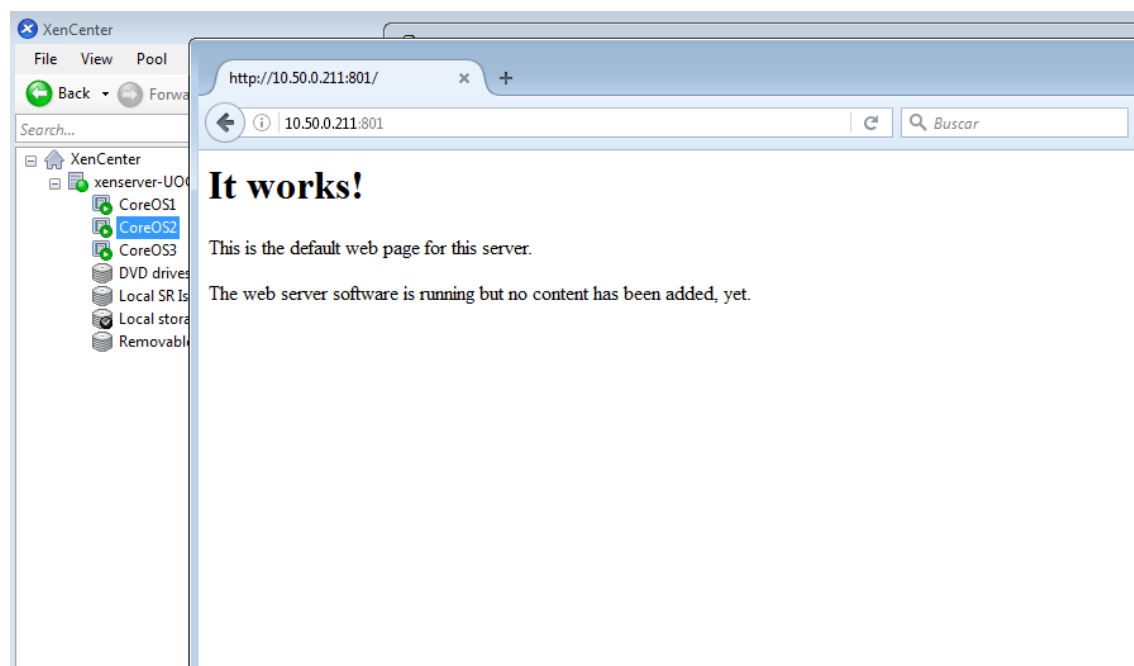
Ens mostrarà quins contenidors tenim corrent al clúster.

```
core@coreos1 ~ $ fleetctl list-units
UNIT                                MACHINE                                ACTIVE  SUB
apache@1.service                    04b97016.../10.50.0.211             active  running
core@coreos1 ~ $
```

Il·lustració 3 Annex 3. Llista unitats al fleet

A la imatge anterior podem veure que tenim un contenidor anomenat apache1 que s'està executant al node que té la ip 10.50.0.211.

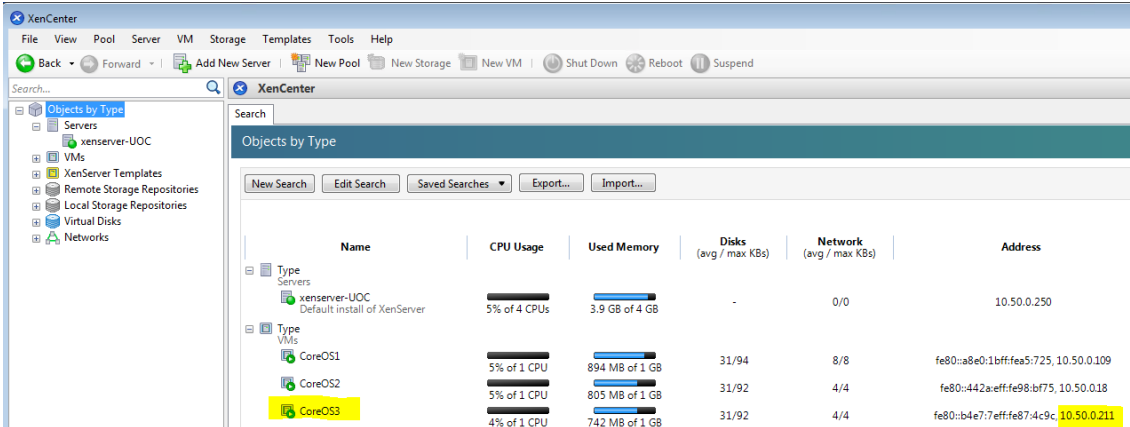
Ja sabem que està escoltant pel port 8081 perquè és el n<sup>o</sup> 1 i ho podem comprovar al navegador.



## Annex 3 Alta Disponibilitat amb Fleet

### 4. Proves Alta disponibilitat

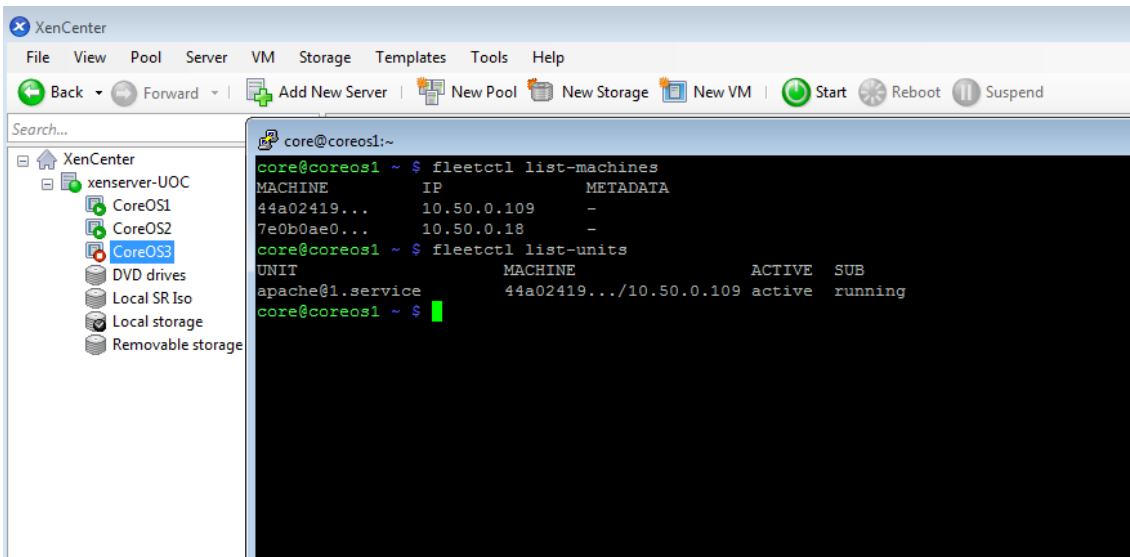
Podem veure a la següent imatge que el node que té la ip 10.50.0.211 és el node 3, ara simularem una caiguda del node3 per veure que fa fleet amb el contenidor que està corrent a aquest node.



Name	CPU Usage	Used Memory	Disks (avg / max KBs)	Network (avg / max KBs)	Address
Type Servers					
xenserver-UOC	5% of 4 CPUs	3.9 GB of 4 GB	-	0/0	10.50.0.250
Type VMs					
CoreOS1	5% of 1 CPU	894 MB of 1 GB	31/94	8/8	fe80::a8e0:1bff:fea5:725, 10.50.0.109
CoreOS2	5% of 1 CPU	805 MB of 1 GB	31/92	4/4	fe80::442a:efff:e98:bf75, 10.50.0.18
CoreOS3	4% of 1 CPU	742 MB of 1 GB	31/92	4/4	fe80::b4e7:7eff:fe87:4c9c, 10.50.0.211

Il·lustració 4 Annex 3. Xen Center

Parem el node per simular una caiguda.



```
core@coreos1:~  
core@coreos1 ~$ fleetctl list-machines  
MACHINE IP METADATA  
44a02419... 10.50.0.109 -  
7e0b0ae0... 10.50.0.18 -  
core@coreos1 ~$ fleetctl list-units  
UNIT MACHINE ACTIVE SUB  
apache@1.service 44a02419.../10.50.0.109 active running  
core@coreos1 ~$
```

Il·lustració 5 Annex3 Alta disponibilitat

A la imatge anterior podem veure com hem apagat el node 3 del clúster i si fem la comanda `fleetctl list-machines`, només ens mostra les dues que estan aixecades i si fem la comanda `fleetctl list-units` podem apreciar que ha mogut el contenidor al node que té la ip 10.50.0.109.

Podem veure amb aquesta arquitectura que és molt més fàcil moure aplicacions d'un servidor a un altre, podem tenir més d'una aplicació amb el mateix servei al mateix servidor, cosa que ens ajuda evitar que molts usuaris estiguin afectats per la caiguda d'un servei compartit entre tots.

Podem engegar 12 apaches en 3 nodes executant la mateixa imatge que hem generat i amb el mateix servei que hem creat de fleet:

```
core@coreos1 ~ $ fleetctl start apache@1.service
Unit apache@1.service inactive
^[[A^[[DUunit apache@1.service launched on 04b97016.../10.50.0.211
core@coreos1 ~ $ fleetctl start apache@2.service
Unit apache@2.service inactive
Unit apache@2.service launched on 44a02419.../10.50.0.109
core@coreos1 ~ $ fleetctl start apache@3.service
Unit apache@3.service inactive
Unit apache@3.service launched on 7e0b0ae0.../10.50.0.18
core@coreos1 ~ $ fleetctl start apache@4.service
Unit apache@4.service inactive
Unit apache@4.service launched on 04b97016.../10.50.0.211
core@coreos1 ~ $ fleetctl start apache@5.service
Unit apache@5.service inactive
Unit apache@5.service launched on 44a02419.../10.50.0.109
core@coreos1 ~ $ fleetctl start apache@6.service
Unit apache@6.service inactive
Unit apache@6.service launched on 7e0b0ae0.../10.50.0.18
core@coreos1 ~ $ fleetctl start apache@7.service
Unit apache@7.service inactive
Unit apache@7.service launched on 04b97016.../10.50.0.211
core@coreos1 ~ $ fleetctl start apache@8.service
Unit apache@8.service inactive
Unit apache@8.service launched on 44a02419.../10.50.0.109
core@coreos1 ~ $ fleetctl start apache@9.service
Unit apache@9.service inactive
Unit apache@9.service launched on 7e0b0ae0.../10.50.0.18
core@coreos1 ~ $ fleetctl start apache@10.service
Unit apache@10.service inactive
Unit apache@10.service launched on 04b97016.../10.50.0.211
core@coreos1 ~ $ fleetctl start apache@11.service
Unit apache@11.service inactive
Unit apache@11.service launched on 44a02419.../10.50.0.109
core@coreos1 ~ $ fleetctl start apache@12.service
Unit apache@12.service inactive
Unit apache@12.service launched on 7e0b0ae0.../10.50.0.18
```

### Il·lustració 6 Annex 3. Arrancant serveis

Engueguem els onze nous contenidors amb apache i podem veure que fleet balanceja la càrrega entre els 3 nodes del clúster.

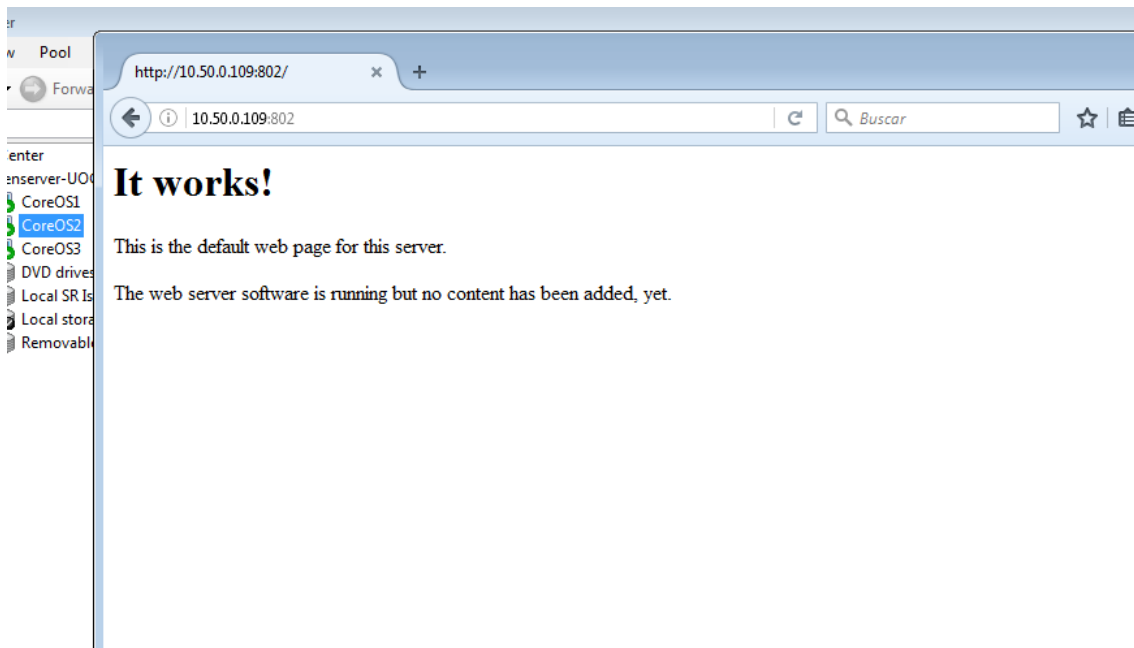
## Annex 3 Alta Disponibilitat amb Fleet

```
core@coreos1 ~ $ fleetctl list-units
UNIT                                MACHINE                                ACTIVE SUB
apache@1.service                    04b97016.../10.50.0.211               active running
apache@10.service                   04b97016.../10.50.0.211               active running
apache@11.service                   44a02419.../10.50.0.109               active running
apache@12.service                   7e0b0ae0.../10.50.0.18                active running
apache@2.service                    44a02419.../10.50.0.109               active running
apache@3.service                    7e0b0ae0.../10.50.0.18                active running
apache@4.service                    04b97016.../10.50.0.211               active running
apache@5.service                    44a02419.../10.50.0.109               active running
apache@6.service                    7e0b0ae0.../10.50.0.18                active running
apache@7.service                    04b97016.../10.50.0.211               active running
apache@8.service                    44a02419.../10.50.0.109               active running
apache@9.service                    7e0b0ae0.../10.50.0.18                active running
core@coreos1 ~ $
```

Il·lustració 7 Annex 3. Llista unitats Fleet

Els ports per on està escoltant els apaches és el 80 més el n<sup>o</sup> de contenidor, així que el contenidor apache1 escolta pel port 801, el apache2 pel 802 i així successivament.

Podem veure uns exemples amb els ports.

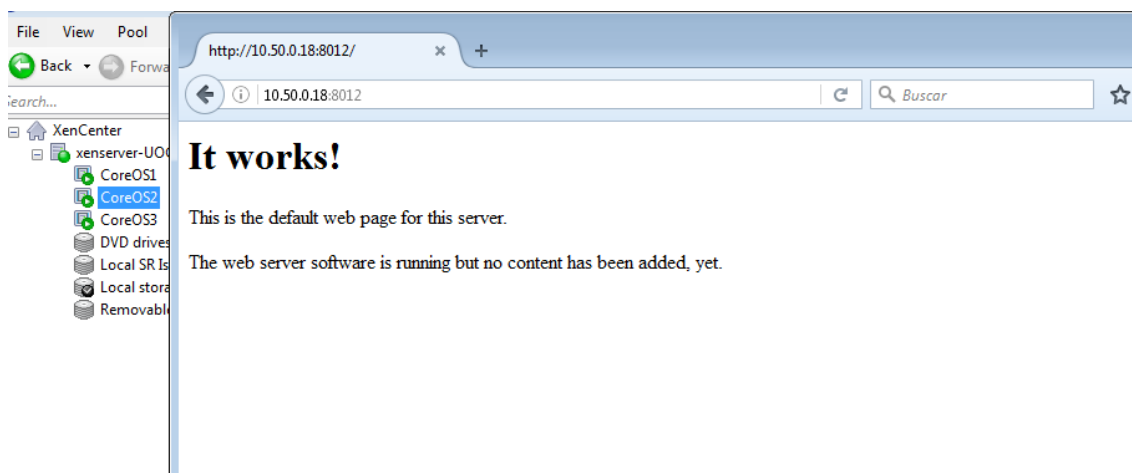


Il·lustració 8 Annex 3. Apache funcionant.

## Annex 3 Alta Disponibilitat amb Fleet

A la imatge anterior veiem que al node que té ip 10.50.0.109 hi ha el contenidor apache2 que esta escoltant pel port 802 d'aquest node.

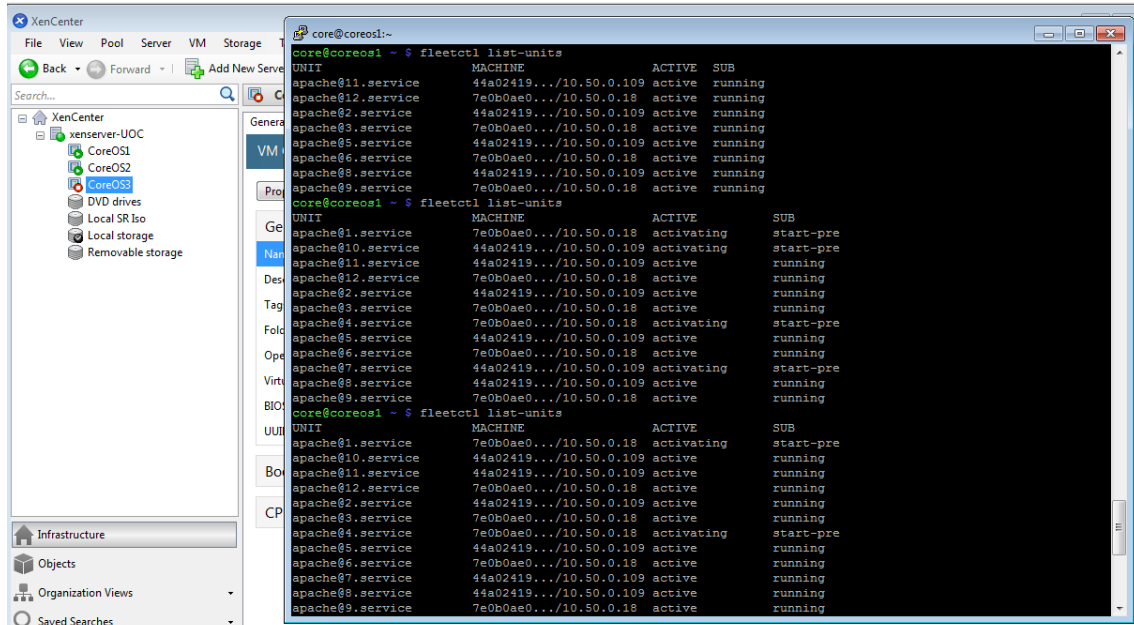
El contenidor apache12 està publicant per la ip del node 10.50.0.18 pel port 8012.



Il·lustració 9 Annex 3. Apache funcionant node 2.

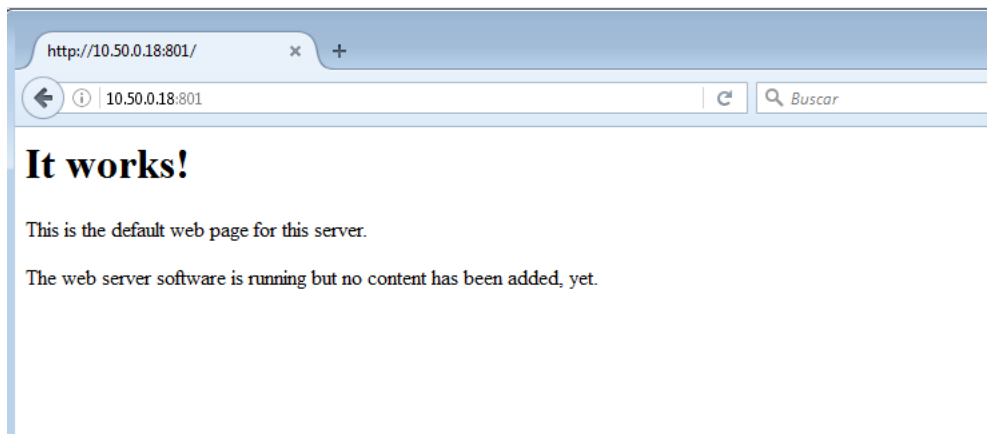
## Annex 3 Alta Disponibilitat amb Fleet

Si apaguem un node fleet mou els contenidors d'aquest node als nodes que segueixen engegats.



Il·lustració 10 Annex 3. Serveis canviant de node.

A la captura anterior podem veure com tots els contenidors que estaven engegats al node amb ip 10.50.0.211 que pertany al node3, hi ha hagut un intent de temps que no estaven a la llista del fleet i seguidament les ha tornat a engegar als altres nodes. Aquesta operació la fa en segons, quasi indetectable pel client. Ara podem veure com el contenidor apache1 està publicant pel port 801 del node amb ip 10.50.0.18.



Il·lustració 11 Annex 3. Prova funcionament Apache



### Annex 3 Alta Disponibilitat amb Fleet

Ja hem creat el servei web amb aplicacions diferenciades per a cada client, si un apache falla per un atac només aquell client tindrà problemes amb la seva pàgina web, problema que fleet solucionarà de manera automàtica eliminant el contenidor i creant-ne un altre.

Índex

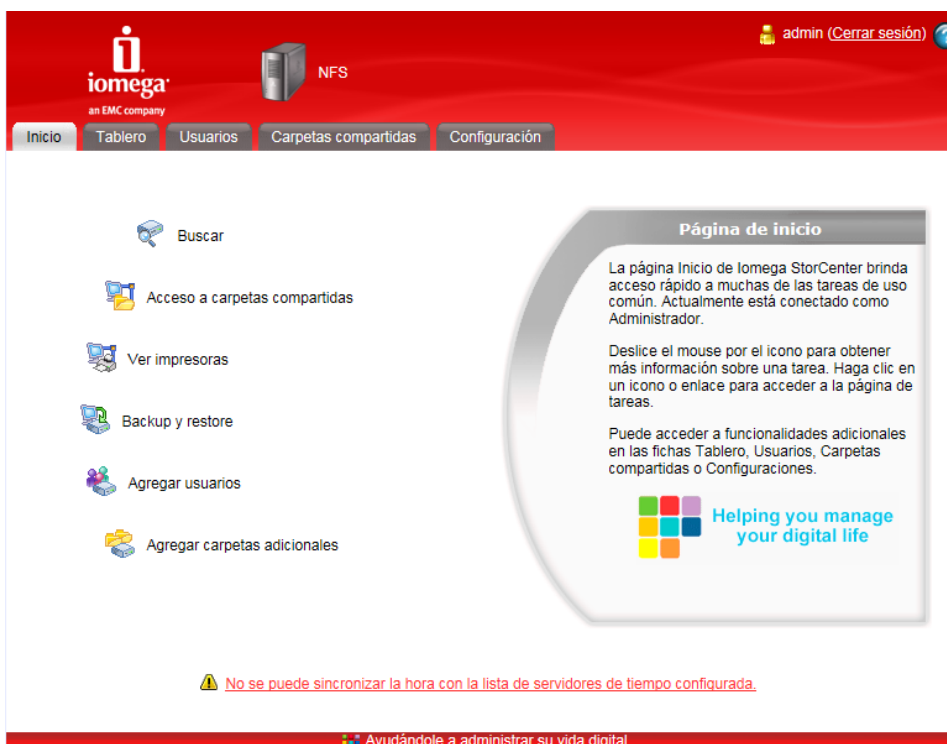
1. Configuració de la xarxa.....	2
2. Configuració del servei NFS .....	5
3. Compartir carpeta mitjançant NFS.....	6

## 1. Configuració de la xarxa.

Per poder compartir una carpeta a la xarxa mitjançant NFS hem de tenir el servidor o bé a la mateixa xarxa del clúster CoreOS o en una xarxa on la porta d'enllaç del clúster sàpigia arribar.

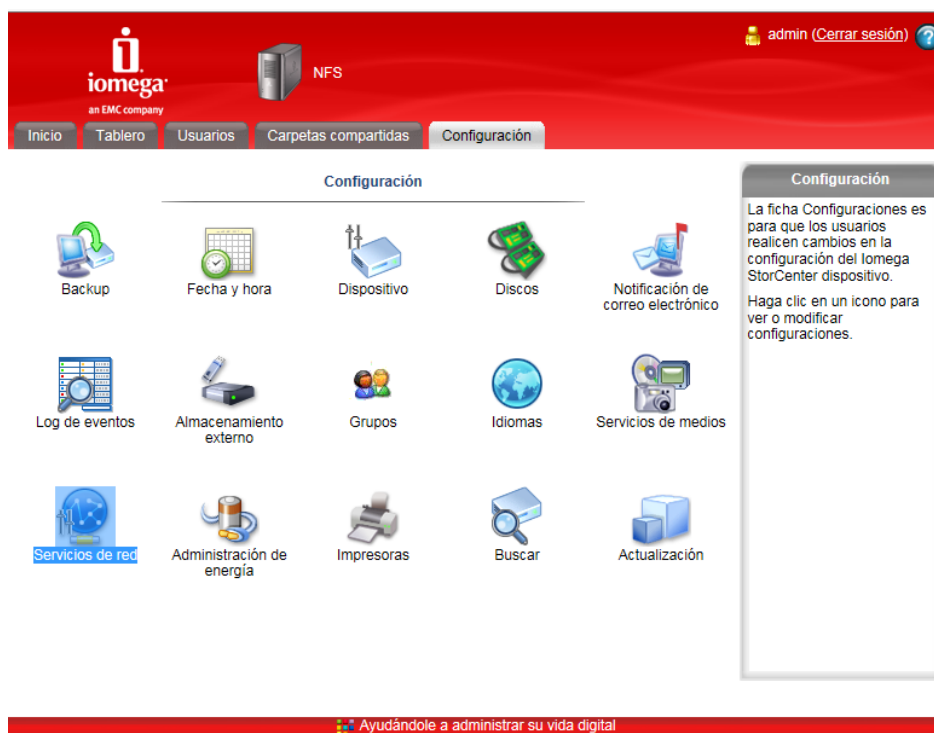
En el cas que ens ocupa hem configurat el servidor NAS a la mateixa xarxa que el servidor Xen, on tenim tots els nodes, i li hem donat la ip 10.50.0.200. Per configurar el Nas primerament hem d'entrar al configurador web mitjançant la ip que porta de fàbrica posant al navegador web [https://ip\\_fabrica](https://ip_fabrica) i si no la sabem podem descarregar una utilitat de Iomega que busca el dispositiu fent un broadcast a nivell de mac address.

Quan entrem per primera vegada en obliga a posar una contrasenya d'administrador. Per configurar la xarxa un cop que hem entrat al nostre NAS, s'ha d'anar a configuració.



Un cop hem entrat a configuració cliquem a Servicios de red.

## Annex 4 Configuració NAS i Servei NFS



Un cop hem entrat a serveis de xarxa hem de prémer l'opció Configuració de Xarxa.



## Annex 4 Configuració NAS i Servei NFS

Ja podem configurar la nostra ip, màscara, porta d'enllaç i dns, podem deixar que la ip la doni un servidor DHCP però haurem de fer una reserva per MAC ADDRESS per garantir que sempre tingui la mateixa ip que configurem al cloud config dels nodes CoreOS.

The screenshot shows the iomega NFS configuration interface. The top navigation bar includes 'Inicio', 'Tablero', 'Usuarios', 'Carpetas compartidas', and 'Configuración'. The main content area is titled 'Configuración de red' and contains the following settings:

- Configurar automáticamente todas las configuraciones de red
- Servidores DNS: 8.8.8.8
- Servidores WINS: (empty fields)
- Configurar automáticamente la dirección IP
- ID de hardware (dirección MAC): 00:D0:B8:02:A0:F7
- Dirección IP: 10.50.0.200
- Máscara de subred: 255.255.255.0
- Gateway: 10.50.0.1

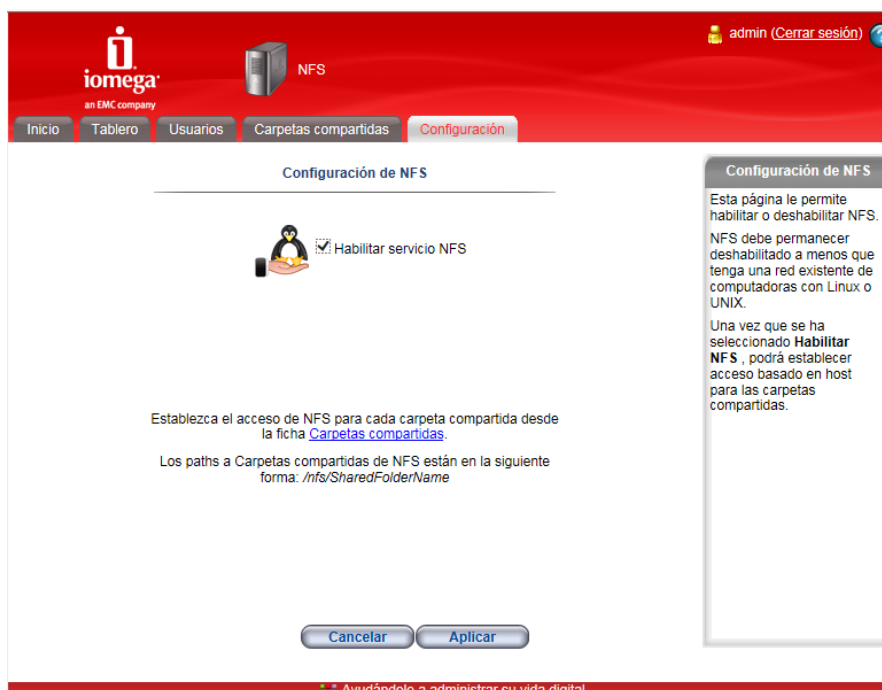
Buttons for 'Cancelar' and 'Aplicar' are located at the bottom. A help box on the right provides instructions: 'La dirección IP y otras configuraciones de la red normalmente se configuran en forma automática. Si no hay servidor DHCP disponible, puede configurar manualmente las configuraciones de la red. Haga clic en **Aplicar** para guardar su configuración.'

## 2. CONFIGURACIÓ DEL SERVEI NFS

Per configurar el servei NFS hem d'anar a configuració i serveis de xarxa igual que hem fet per configurar l'adreça ip del NAS. Un cop hem entrat a la configuració de la xarxa hem de clicar sobre la icona del pingüi sobre una mà.

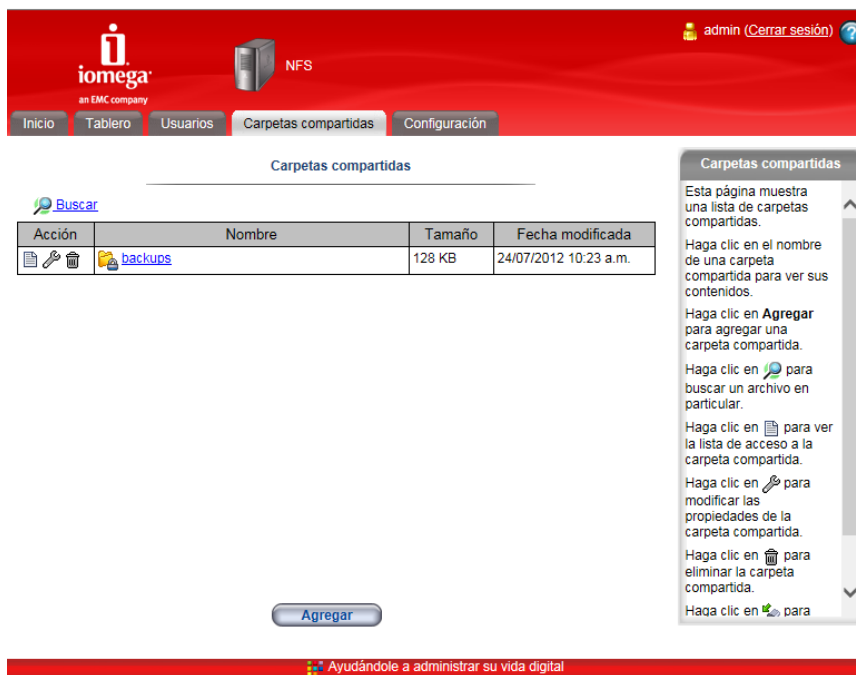


Un cop que hem entrat a la configuració del servei NFS hem d'activar el check box que diu "Habilitar Servicio NFS". Ens informa que les carpetes compartides seran compartides dins de la carpeta /nfs/nom\_carpeta\_compartida.



### 3. COMPARTIR CARPETA MITJANÇANT NFS

Quan tenim activat el servei ja podem clicar sobre l'enllaç carpetes compartides i configurar-la clicant sobre el botó Agregar.



Posem el nom de la carpeta a compartir i habilitem la seguretat si ho veiem necessari a la nostra xarxa, normalment NFS configura les ip de les màquines a les que volem donar permisos d'accés.



## Annex 4 Configuració NAS i Servei NFS

Un cop configurada podem clicar sobre el botó Aplicar i ja es tindrà la carpeta compartida



Al clúster s'ha de configurar a l'arxiu Cloud-Config dels nodes que connecti a la carpeta NFS a la ruta 10.50.0.200:/nfs/NFS a la carpeta local que es crearà per aquest efecte.

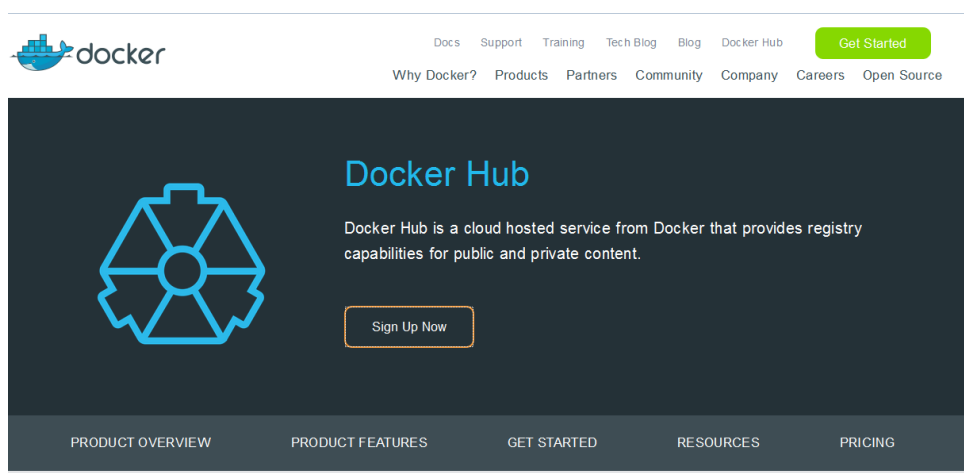


Índex

1. Creació de Repositori Docker.....	2
2. Creació de fitxer Docker DockerFile .....	3
3. Compilació de la imatge docker i pujada al repositori.....	5
4. Explicació i execució del servei.....	6

## 1. Creació de Repositori Docker

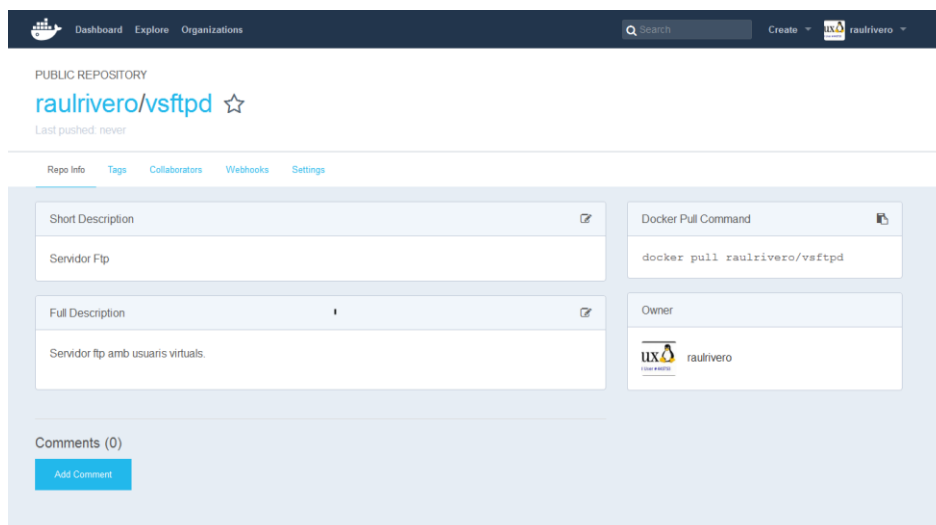
Per aquest servei s'ha creat un repositori al docker hub per poder pujar la imatge creada localment i que estigui accessible des de qualsevol node del clúster o des de qualsevol sistema que tingui instal·lat docker.



Il·lustració 1, Docker hub

Per crear el repositori s'ha accedit a la pàgina web de docker <http://hub.docker.com> i s'ha creat un compte d'usuari. Per crear el compte et demana un nom d'usuari el correu electrònic i una contrasenya, que necessitarem per poder pujar les imatges posteriorment.

Un cop tinguem l'usuari creem un repositori. En aquest cas s'ha fet un repositori públic on tothom podrà veure la nostra imatge i descarregar-la per fer-la servir.



Il·lustració 2, Repositori Docker

## 2. Creació de fitxer Docker DockerFile

Un cop tenim el repositori creat s'ha de crear la imatge del nostre servei. Per crear una imatge Docker s'ha de fer un fitxer amb els paràmetres que volem que docker executi al crear-la. La imatge que s'ha creat pel servei ftp fa servir la imatge ubuntu com a base i executa una sèrie de comandes abans de ser empaquetada.

```
core@coreos1 /mnt/nfs/services/ftpraul $ cat Dockerfile
# debian-based container for vsftpd
# VERSIO 0.1 FROM ubuntu

FROM ubuntu:14.04
MAINTAINER Raul Rivero Ramos <rrivero@uoc.edu>

#Instal·lar paquets

RUN apt-get update && apt-get install -y --no-install-recommends vsftpd libpam-pwdfilere apache2-utils
RUN apt-get clean

#Configuració del fitxer de vsftpd

RUN mkdir -p /var/run/vsftpd/empty \
&& mkdir /etc/vsftpd \
&& useradd --home /home --gid nogroup -m --shell /bin/false vsftpd

ADD conf/vsftpd.conf /etc/vsftpd.conf
ADD conf/crea_usuari_ftp.sh /bin/crea_usuari_ftp.sh
ADD conf/vsftpd.pam /etc/pam.d/vsftpd
RUN chmod a+x /bin/crea_usuari_ftp.sh

#port exposat

EXPOSE 21

CMD ["vsftpd"]
core@coreos1 /mnt/nfs/services/ftpraul $
```

Il·lustració 3, Dockerfile imatge vsftpd

El fitxer Dockerfile amb la comanda FROM li diu a docker que faci servir la imatge ubuntu com a base, després amb la comanda RUN executa la instal·lació dels paquets necessaris. Seguidament creem els directoris necessaris per la configuració del servei i afegim els documents de configuració a les carpetes creades.

El fitxer vsftpd.conf configura el servei per tenir usuaris virtuals, és a dir, que no estan al passwd del sistema, ja que farem servir el servei en diferents nodes i els usuaris del sistema no seran els mateixos en tots els nodes, el fitxer crea\_usuari\_ftp.sh crea un usuari nou pel ftp, i el fitxer vsftpd.pam és el fitxer de configuració del programa que ens crea el fitxer de passwd segur del nostre servei.

```
core@coreos1 /mnt/nfs/services/ftp/conf $ cat vsftpd.pam
# Virtual user
auth required pam_pwdfile.so pwdfile /etc/vsftpd/users.passwd
account required pam_permit.so
core@coreos1 /mnt/nfs/services/ftp/conf $
```

Il·lustració 4, fitxer vsftpd.pam

La comanda EXPOSE informa a docker de quin port ha de tenir obert el contenidor i per acabar la comanda CMD ens executarà el que posem entre claudàtors. En el nostre cas volem que executi vsftpd perquè és el servei ftp que volem executar al contenidor.

### 3. Compilació de la imatge docker i pujada al repositori

Per compilar la imatge i pujar-la al repositori cal executar les següents comandes:

```
core@coreos1 /mnt/nfs/services/ftp $ docker build --rm -t raulrivero/vsftpd ./
Sending build context to Docker daemon 9.216 kB
Step 1 : FROM ubuntu:14.04
--> b549a9959a66
Step 2 : MAINTAINER Raul Rivero Ramos <rrivero@uoc.edu>
--> Using cache
--> 3981bf58c389
Step 3 : RUN apt-get update && apt-get install -y --no-install-recommends vsftpd libpam-pwdfilere apache2-utils
--> Using cache
--> a1e20ea97e40
Step 4 : RUN apt-get clean
--> Using cache
--> d90a9a97ded5
Step 5 : RUN mkdir -p /var/run/vsftpd/empty && mkdir /etc/vsftpd && useradd --home /home --gid nogroup -m --shell /bin/false vsftpd
--> Using cache
--> f428b86beee3
Step 6 : ADD conf/vsftpd.conf /etc/vsftpd.conf
--> Using cache
--> d77450808d41
Step 7 : ADD conf/crea_usuari_ftp.sh /bin/crea_usuari_ftp.sh
--> Using cache
--> 2d3dab4f1c93
Step 8 : ADD conf/vsftpd.pam /etc/pam.d/vsftpd
--> Using cache
--> d727a508e54d
Step 9 : RUN chmod a+x /bin/crea_usuari_ftp.sh
--> Using cache
--> d8f34da4f92a
Step 10 : EXPOSE 21
--> Using cache
--> a0dc7eaf2493
Step 11 : CMD vsftpd
--> Using cache
--> a43d15a276ae
Successfully built a43d15a276ae
core@coreos1 /mnt/nfs/services/ftp $
```

#### Il·lustració 5, Docker Build de la imatge

A la il·lustració 4 es pot veure la comanda, `docker build --rm -t raulrivero/vsftpd ./`. Aquesta comanda s'executa al directori on està el fitxer Dockerfile o sinó passar-li la ruta d'on està. Un cop s'executa docker executa totes les comandes que li hem dit i genera una imatge amb el nom \$usuari/\$servei, en el nostre cas `raulrivero/vsftpd`.

Un cop tenim la imatge creada ja podem pujar-la al nostre repositori amb la comanda:

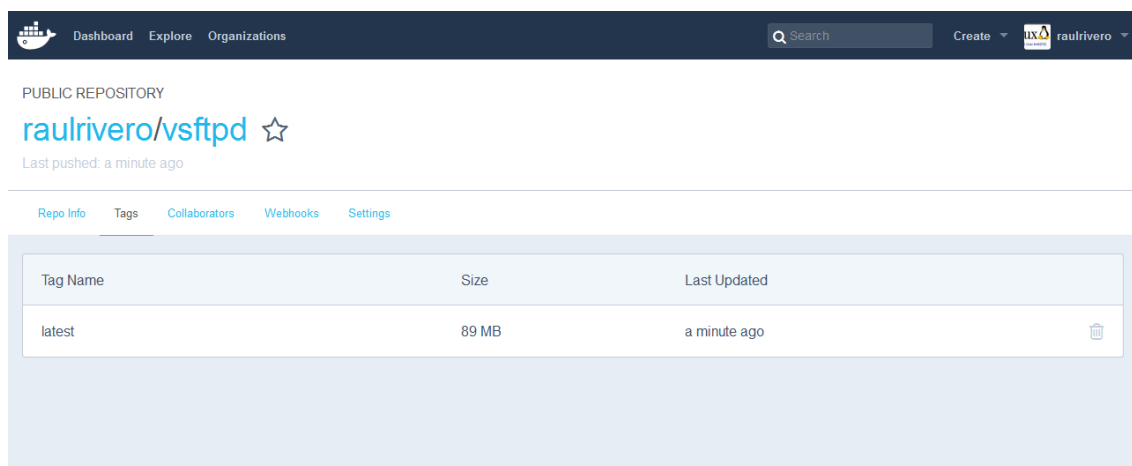
```
core@coreos2 /mnt/nfs/services/ftpraul $ docker login --username=raulrivero --email=raulrivero46@gmail.com
Password:
WARNING: login credentials saved in /home/core/.docker/config.json
Login Succeeded
core@coreos2 /mnt/nfs/services/ftpraul $ docker push raulrivero/vsftpd
The push refers to a repository [docker.io/raulrivero/vsftpd]
952222876d8c: Pushed
33bc641ca519: Pushed
cc667eca2145: Pushed
306c32155a1e: Pushed
4d2d121fd943: Pushed
5bd3269b81f7: Pushing [=====>] 16.59 MB/26.36 MB
5f70bf18a086: Pushed
348c57e94a93: Pushed
4ac0f65a29bb: Pushed
dcd5e38e7975: Pushing [=====>] 39.7 MB/187.7 MB
```

#### Il·lustració 6, push imatge docker

## Annex5 Servei FTP

A la imatge 5 es pot veure que abans de poder executar la comanda docker push, que puja la imatge al repositori, hem d'executar docker login per connectar-nos al nostre compte d'usuari del docker hub.

Ara ja tenim la imatge al repositori i podem descarregar-la amb la comanda docker pull des de qualsevol node del clúster.



Il·lustració 7, Repositori Docker després del PUSH

#### 4. Explicació i execució del servei

El servei vsftpd és un servei de ftp que ens deixa configurar-lo per fer usuaris virtuals, com ja s'ha citat a l'inici d'aquest annex, aquesta particularitat ens és de molta ajuda ja que el servei ftp que executarem serà a un contenidor i en qualsevol node del clúster, fins i tot podríem tenir més d'un contenidor executant-se depenent del nombre de clients que tinguem a l'empresa.

Per aquesta particularitat, el de tenir més d'un contenidor executant-se, serà necessari que el directori de configuració /etc/vsftpd estigui al nostre servidor NAS i també hem de connectar el directori /mnt/nfs/www, que és on tenim els directoris de les pàgines web que vam configurar amb el servei apache, com als directoris dels usuaris del servei ftp.

Per executar el servei farem servir la comanda Docker run passant per paràmetre els ports que obrirà el node, els directoris a connectar i la imatge que hem de posar al contenidor.

```

core@coreos1 /mnt/nfs/services/ftpraul/conf $ docker run -d -p 21:21 --name vsftpd -h "vsftpd" -v /mnt/nfs/www:
/home -v /mnt/nfs/services/ftp/conf:/etc/vsftpd raulrivero/vsftpd
d6ddb5199a7abb69e7e1f82a763ad95e3b67657c0f0b47770c8196d464bf0a44
core@coreos1 /mnt/nfs/services/ftpraul/conf $ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
d6ddb5199a7a        raulrivero/vsftpd  "vsftpd"           5 seconds ago      Up 4 seconds       0.0.0.0:21-
>21/tcp    vsftpd

```

Il·lustració 8, Docker run

A la il·lustració 8 es pot veure la comanda doker run on el flag `-p` diem que el port 21 del node es redirigeixi cap al port 21 del contenidor, amb el flag `-v` és amb la comanda que connectem els directoris del NAS cap a les carpetes del contenidor. D'aquesta manera si obrim dos serveis ftp, un a cada node, tots dos llegiran la configuració del NAS i si fem un usuari a un node, l'altre llegirà el fitxer `user.passwd` del NAS i permetrà entrar a l'usuari.

Per crear un usuari al servei tenim un script anomenat `crea_usuari_ftp.sh` que simplement amb la comanda, `htpasswd -bcd $pwd_file $username $userpass`, crea un usuari al fitxer `/mnt/nfs/services/ftp/conf/users.passwd`. Per poder executar l'escript que esta dins del contenidor hem de fer servir la comanda `docker exec`, que executa la comanda que volem al contenidor que li diem.

```

core@coreos1 /mnt/nfs/services/ftp/conf $ docker exec vsftpd crea_usuari_ftp.sh prova prova
Adding password for user prova

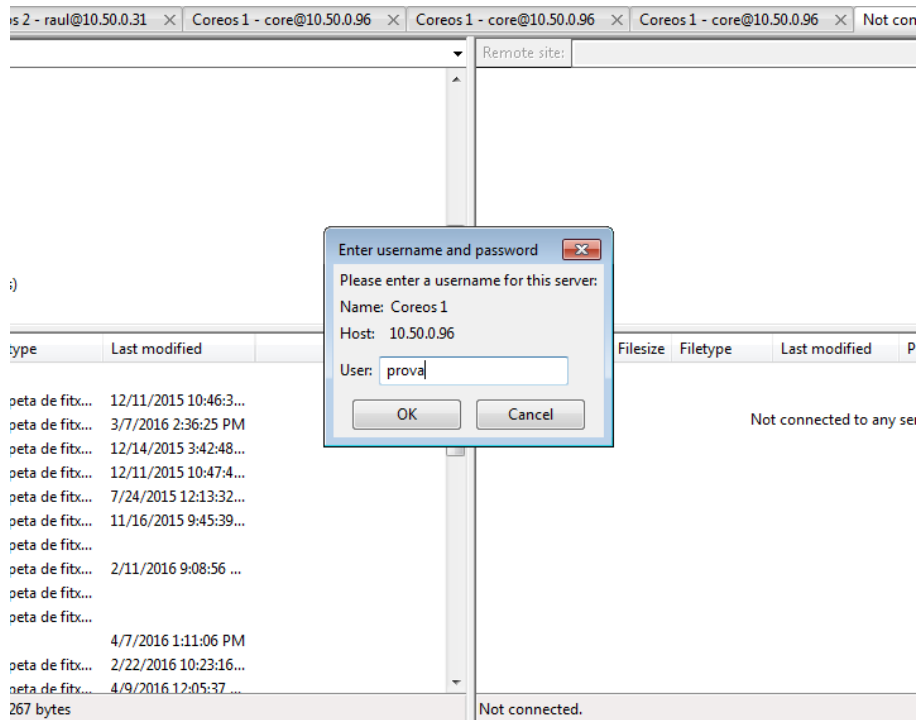
User Name      : prova
User Passwd    : prova
Encrypted Password : 882C8QeeZZdVI
core@coreos1 /mnt/nfs/services/ftp/conf $ cat users.passwd
prova:882C8QeeZZdVI
core@coreos1 /mnt/nfs/services/ftp/conf $ mkdir /mnt/nfs/www/prova
core@coreos1 /mnt/nfs/services/ftp/conf $ mkdir /mnt/nfs/www/prova/ftp
core@coreos1 /mnt/nfs/services/ftp/conf $ mkdir /mnt/nfs/www/prova/ftp1
core@coreos1 /mnt/nfs/services/ftp/conf $ mkdir /mnt/nfs/www/prova/ftp2
core@coreos1 /mnt/nfs/services/ftp/conf $ █

```

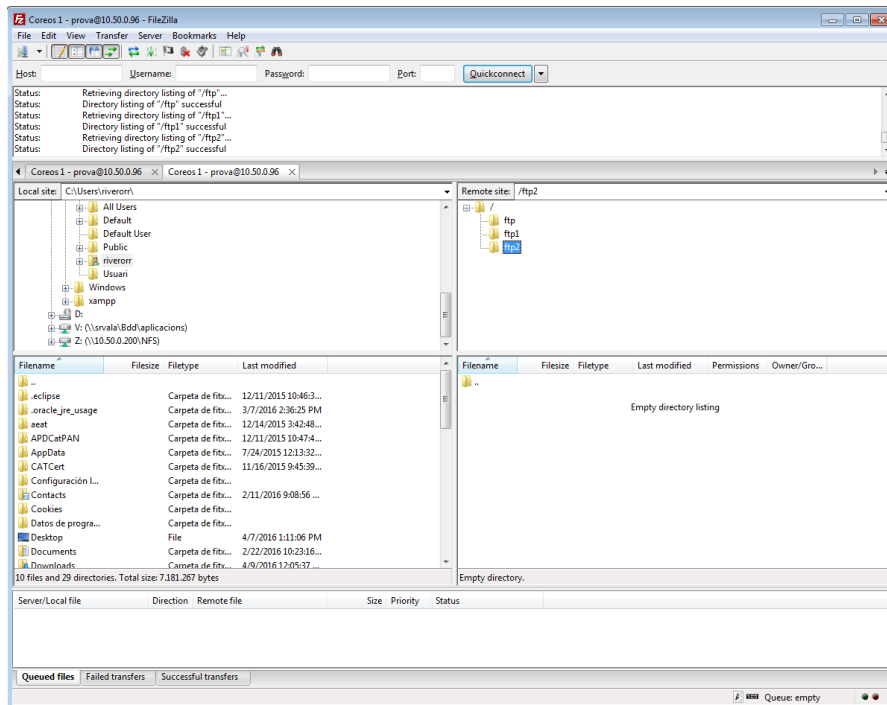
Il·lustració 9, comanda Docker exec

Veiem que executem al contenidor anomenat `vsftpd` l'escript `crea_usuari_ftp.sh` i li passem per paràmetre el nom d'usuari `prova` i la contrasenya `prova`, l'escript ens retorna l'usuari creat i seguidament fem la carpeta d'usuari `prova` al directori `www` del nostre NAS, he creat les carpetes `ftp`'s per comprovar la connexió amb un client ftp.

## Annex5 Servei FTP



Il·lustració 10, connexió a ftp usuari



Il·lustració 11, connexió ftp



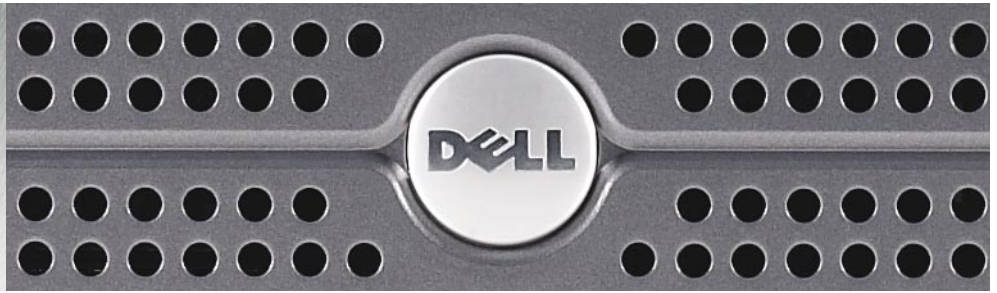
## Annex5 Servei FTP

Podem veure que la connexió i el llistat del directori ha estat satisfactori. Si ara mirem el fitxer users.passwd podrem veure l'usuari que ha creat la comanda htpasswd.

```
core@coreos1 ~ $ cat /mnt/nfs/services/ftp/conf/users.passwd
prova:882C8QeeZZdVI
core@coreos1 ~ $
```

Il·lustració 12, passwd

## SERVIDOR DELL POWEREDGE 2950



**En un factor de forma de 2U optimizado en rack, el servidor Dell™ PowerEdge™ 2950 ofrece un equilibrio perfecto entre un rendimiento, una disponibilidad y una flexibilidad excelentes para aplicaciones de infraestructura de red en crecimiento así como para la consolidación Web, de mensajería, de bases de datos y de archivos/impresión**

### **9ª generación de innovadores servidores PowerEdge de Dell**

Gracias a un diseño de hardware innovador, a la compatibilidad de software y al continuo enfoque en minimizar las actualizaciones del sistema, la 9ª generación de servidores PowerEdge de Dell ayuda a reducir la complejidad que implica la administración de datos, tanto si se trata de una empresa grande o pequeña. Estos servidores están diseñados para una Especificación conductual desarrollada por Dell™ que define un formato de hardware y una interacción del usuario uniformes en todos los modelos de esta generación y de futuras generaciones de PowerEdge. Además, una imagen del sistema principal compartida con 1950 y 2900 permite las actualizaciones del BIOS, los drivers del sistema, el firmware, los sistemas operativos y las aplicaciones desde una plantilla fácil de copiar para una administración de software simplificada. Con los últimos procesadores Intel® Xeon®, la 9ª generación de servidores PowerEdge ofrecen la potencia y el rendimiento que espera de Dell.

### **Dell PowerEdge 2950 reúne flexibilidad y capacidad de almacenamiento en formato 2U**

El servidor Dell PowerEdge 2950 ofrece flexibilidad de configuración en un chasis de 2U para organizaciones que requieren una capacidad de almacenamiento interno reducido en lugar de un sistema de almacenamiento externo. Seis compartimentos internos de unidad de disco duro que proporcionan hasta 1,8 TB<sup>1</sup> de almacenamiento interno que ayudan a ahorrar espacio en los centros de datos a la vez que ofrecen suficiente espacio de almacenamiento para admitir aplicaciones cada vez mayores.

Diseñado para ayudar a las organizaciones a satisfacer las necesidades en constante cambio, el servidor incluye distintas funciones integradas, como las tarjetas duales NIC Gigabit y el driver de almacenamiento SAS integrado, que admite tres ranuras PCI Express™. De esta forma, es posible una expansión sencilla que admitirá una amplia variedad de cargas de trabajo del centro de datos.

### **Alto rendimiento y disponibilidad para maximizar el tiempo de actividad**

El servidor Dell PowerEdge 2950 admite hasta dos de los últimos procesadores Intel Xeon de cuatro núcleos y el conjunto de chips Intel 5000X. Estos últimos procesadores proporcionan un rendimiento y un rendimiento por vatio mejorados<sup>2</sup>. Con la nueva capacidad para admitir ocho unidades SAS de 2,5", el 2950 también proporciona suficiente espacio para fragmentar los datos en varias unidades para obtener un rendimiento mayor en entornos exigentes.

Además, se obtiene un rendimiento y una capacidad de memoria excepcionales gracias a los 32 GB de la memoria DIMM con memoria intermedia completa. El servidor incluye la tecnología PCI Express para un rendimiento de E/S excelente, un motor de carga TCP/IP (TOE) que transfiere el procesamiento TCP/IP a un procesador dedicado en la tarjeta NIC integrada para aumentar el rendimiento de la CPU. Y con funciones tales como las fuentes de alimentación/ventiladores conectables en marcha, configuraciones RAID con caché con reserva de memoria por batería y una opción de unidad en cinta interna para la copia de seguridad de datos locales, el servidor Dell PowerEdge 2950 ayuda a garantizar que sus datos están protegidos y que se puede acceder a los mismos.

### **Facilidad de administración para una complejidad reducida**

El servidor Dell PowerEdge 2950 está equipado con un Driver de administración de la placa base (BMC) que incluye un conjunto de herramientas completo que supervisa el hardware de servidor, le avisa cuando se producen fallos en el servidor y permite las operaciones remotas básicas. Para entornos con servidores ubicados en centros de datos seguros o en sitios que carecen de personal de TI, Dell ofrece una característica opcional para los servidores PowerEdge, el Driver de acceso remoto de Dell (DRAC). Funciona mediante una interfaz de usuario gráfica basada en, DRAC puede admitir la supervisión, la solución de problemas, la reparación, las actualizaciones y el acceso remoto del estado del sistema operativo. Un software común con la misma familia de servidores de 9ª generación PowerEdge ayudan a simplificar aún más la administración. Además, la Especificación conductual de Dell proporciona una plataforma conocida para una facilitan la implementación, la administración y los servicios e implica una reducción del coste total de la propiedad (TCO) en diversas generaciones de servidores PowerEdge.



Dell PowerEdge 2950



## SERVICIOS DE INFRAESTRUCTURA DE TI DE DELL

Dell aporta ejecución pura a los Servicios de TI. La planificación, implementación y mantenimiento de su infraestructura de TI no merece menos. La variabilidad en la ejecución puede afectar a la productividad del usuario, los recursos de TI y, en definitiva, a su reputación. Al aprovechar nuestra herencia en la calidad de dirección del proceso, en Dell Services podemos ofrecer un método más inteligente.

No pretendemos hacerlo todo. Nos encontramos en los servicios de infraestructura de TI. Y tomamos un enfoque dirigido hacia el cliente, basándonos en la filosofía de que usted conoce su negocio mejor que nadie. Por eso Dell no intenta tomar decisiones clave sin su conocimiento o le ofrece más de lo que necesita. Todo lo contrario, aplicamos nuestra administración de procesos a nivel mundial y nuestra cultura "sin excusas" para ofrecer lo que nuestros clientes necesitan más actualmente: flexibilidad y calidad constante. Esto es pura ejecución. Esto es Dell en estado puro.

### Servicios de evaluación, diseño e implementación

Los departamentos de TI continuamente se enfrentan al reto de evaluar e implementar nuevas tecnologías. Los servicios de evaluación, diseño e implementación de Dell pueden reestructurar su entorno de TI para mejorar el rendimiento, escalabilidad y eficacia al tiempo que contribuyen a maximizar su inversión y minimizar la interrupción de su negocio.

### Servicios de distribución

La implementación del sistema es un mal necesario que invade casi todas las organizaciones. Debe implementar nuevos sistemas para ayudar a mejorar el rendimiento y satisfacer los requisitos del usuario. Con los servicios de implementación de Dell, ayudamos a simplificar y acelerar la implementación y el uso de nuevos sistemas para maximizar el tiempo de actividad en su entorno de TI.

### Servicios de recuperación y reciclado de activos

La eliminación, reventa y donación correctas del equipo informático constituyen una larga tarea que suele encontrarse al final de muchas listas de tareas informáticas. Dell simplifica los procesos de caducidad del equipo informático de modo que maximiza el valor para los clientes.

### Servicios de formación

Aporte a sus empleados los conocimientos y habilidades que necesitan para ser tan productivos como sea posible. Dell ofrece extensos servicios de formación que incluyen formación en hardware y software, así como clases de desarrollo profesional. Con la formación de Dell, puede contribuir a mejorar la fiabilidad del sistema, maximizar la productividad y reducir las peticiones del usuario final y el tiempo de inactividad.

### Servicios de asistencia técnica a empresas

Con Dell, puede obtener el máximo rendimiento y disponibilidad de su servidor y sistemas de almacenamiento Dell. Los Servicios de asistencia de nuestra empresa ofrecen un mantenimiento proactivo para ayudar a evitar problemas y para responder y solucionar rápidamente los problemas cuando se produzcan. Hemos construido una infraestructura global que ofrece distintos niveles de asistencia para los sistemas de su infraestructura.

Para ayudarle a obtener el máximo provecho de sus sistemas Dell, visite [www.dell.com/services](http://www.dell.com/services).

Los servicios varían según la zona.

## CARACTERÍSTICAS DESCRIPCIÓN

<b>Formato</b>	Altura en rack de 2U
<b>Procesadores</b>	Hasta dos procesadores de secuencia de doble núcleo Intel® Xeon® 5000 con 3.0 GHz de frecuencia de reloj o hasta dos procesadores de secuencia de doble núcleo Intel Xeon 5100 con 3.0 GHz de frecuencia de reloj o hasta dos procesadores de secuencia de cuatro núcleos Intel Xeon 5300 con 2.66 GHz de frecuencia de reloj
<b>Bus frontal</b>	Secuencia 5000: 667 MHz o 1066 MHz Secuencia 5100: 1066 MHz o 1333 MHz Secuencia 5300: 1066 MHz o 1333 MHz
<b>Caché</b>	Secuencia 5000: Caché de nivel 2 de 2 x 2 MB por procesador Secuencia 5100: caché de nivel 2 de 4 MB por procesador Secuencia 5300: caché de nivel 2 de 2 x 4MB por procesador
<b>Conjunto de chips</b>	Intel 5000X
<b>Memoria</b>	Módulos DIMM de 256 MB/512 MB/1 GB/2 GB/4 GB con memoria intermedia completa (FBD) en pares coincidentes; 533 MHz o 667 MHz; 8 zócalos para admitir hasta 32 GB
<b>Ranuras de E/S</b>	Seis en total: tres ranuras PCI, con aumento PCIe con tres ranuras PCI Express (una de 1 x 4 y dos de 1 x 8) o dos ranuras PCI-X de 64 bits/133 MHz y una ranura PCI Express de 1 x 8; 2 tarjetas NIC Gigabit integradas; puerto de administración con DRAC5 opcional
<b>Driver de almacenamiento integrado</b>	PERC 5/i (opcional): driver RAID SAS de 3 Gb/s RAID con procesador Intel IOP333 y caché de 256 MB; SAS 5/i (base): driver de 4 puertos con procesador ARM966 (no admite RAID)
<b>Driver RAID complementario</b>	PERC 4e/DC opcional (driver RAID PCI Express de canal dual); Adaptador PERC 5/E opcional para almacenamiento RAID externo
<b>Compartimentos de disco duro</b>	3 opciones base de disco duro: Opción de 8 discos duros de 2,5": Opción de disco duro de 2,5": hasta 8 discos duros SAS (a 10.000 rpm) Opción de 4 discos duros de 3,5": Opción de disco duro de 3,5": hasta 4 unidades SAS (a 10.000/15.000 rpm) o unidades SATA (7.200) Opción de 6 discos duros de 3,5": Opción de disco duro de 3,5": hasta 6 unidades SAS (a 10.000/15.000 rpm) o unidades SATA (7.200) Opciones de compartimentos para periféricos: Unidad de disquete, unidad en cinta DAT72 (no disponible con base de 6 discos duros de 3,5") Compartimento para unidad óptica delgado con opción de unidad de CD-ROM, de DVD-ROM o unidad combinada de CD-RW/DVD-ROM
<b>Almacenamiento interno máximo</b>	Hasta 1,8 TB
<b>Unidades de disco duro conectables en marcha<sup>1</sup></b>	SAS de 2,5" (a 10.000 rpm): unidades de disco duro de 36 GB o 73 GB conectables en marcha; SAS de 3,5" (a 10.000 rpm): unidades de disco duro de 73 GB, 146 GB, 300 GB conectables en marcha; SAS de 3,5" (a 15.000 rpm): unidades de disco duro de 36 GB, 73 GB, 146 GB conectables en marcha; SATA de 3,5" (a 7.200 rpm): unidades de disco duro de 80 GB, 160 GB, 250 GB conectables en marcha <sup>2</sup>
<b>Almacenamiento interno</b>	4 o 6 unidades SAS de 3,5" conectables en marcha (a 10.000 y 15.000 rpm)/unidades SATA (7.200) o 8 unidades SAS de 2,5" conectables en marcha (a 10.000 rpm)
<b>Almacenamiento externo</b>	Dell PowerVault™ 22xS, PowerVault MD1000, productos Dell/EMC
<b>Opciones de copia de seguridad en cinta</b>	Internas: PV100T (DAT 72) con varios compartimentos Externas: PowerVault DAT 72, 110T, 114T, 122T, 124T, 132T, 136T, 160T y ML6000
<b>Tarjeta de interfaz de red</b>	NIC Gigabit Ethernet Broadcom® NetXtreme II™ 5708 dual integrada <sup>3</sup> NIC Ethernet con compensación de carga y capacidad de recuperación. TOE (motor de carga TCP/IP) compatible con Microsoft Windows Server 2003, SP1 o superior con Scalable Networking Pack. Tarjetas NIC complementarias opcionales: Adaptador de puerto dual Intel® PRO/1000 PT, Gigabit, Copper, PCI-E x4; adaptador de servidor de un solo puerto Intel® PRO/1000 PT, Gigabit, Copper, PCI-E x1; adaptador de servidor de un solo puerto Intel® PRO/1000 PF, Gigabit, óptico, PCI-E x4; NIC Gigabit Broadcom® NetXtreme™ 5721 de un solo puerto, Copper, PCI-E x1; NIC Gigabit Ethernet Broadcom® NetXtreme II™ 5708 de un solo puerto con TOE, Copper, PCI-E x4
<b>Fuente de alimentación</b>	Fuente de alimentación estándar de 750 vatios conectable en marcha, fuente de alimentación redundante opcional de 750 vatios conectable en marcha; conmutación automática universal de 110/220 voltios
<b>Disponibilidad</b>	Unidades de disco duro conectables en marcha, fuente de alimentación redundante conectable en marcha; refrigeración redundante conectable en marcha; memoria ECC; banco de reserva; Single Device Data Correction (SDDC); tarjeta secundaria PERC 5/i integrada con caché con reserva de memoria por batería; soporte de conmutación por error de alta disponibilidad; DRAC5; soporte para dispositivos de cinta interno; chasis sin necesidad de herramientas; compatibilidad con clústeres
<b>Vídeo</b>	ATI ES1000 integrada con memoria de 16 MB
<b>Administración remota</b>	Driver de administración de la placa base estándar compatible con IMPI 2.0; DRAC5 opcional para funciones avanzadas
<b>Administración de sistemas</b>	Dell OpenManage™
<b>Compatibilidad con rack</b>	4 postes (rack Dell), 2 postes y guías Versa de terceros, guías móviles y brazo para la manipulación de cables
<b>Sistemas operativos</b>	Microsoft® Windows Server 2003 R2, Standard, Enterprise y Web Edition, x64, Standard y Enterprise Edition; Microsoft® Windows® Storage Server 2003 R2, Workgroup, Standard, Enterprise Edition; Red Hat® Linux® Enterprise v4, ES y ES EM64T; SUSE Linux Enterprise Server 9 EM64T, SP3

<sup>1</sup> Para las unidades de disco duro, GB significa 1.000.000.000 de bytes, la capacidad total accesible varía en función del material preconfigurado y el entorno operativo y será inferior.  
<sup>2</sup> Basado en las pruebas realizadas por los laboratorios de Dell en mayo de 2006 mediante el banco de pruebas SPECintb2000 en un PE2950 con dos procesadores duales Intel Xeon 5080 (3,73 GHz Dempsey) de doble núcleo, 4 módulos de memoria FBD de 1 GB a 533 MHz, 2 unidades de disco duro SAS de 73 GB a 15.000 rpm, sistema operativo Windows Server 2003 Enterprise x64 Edition comparados con un PE2850 con dos procesadores duales Intel Xeon de doble núcleo de 2,8 GHz (Paxville), 4 módulos de memoria DDR2 de 1 GB a 400 MHz, 2 unidades de disco duro SCSI de 36 GB a 15.000 rpm y sistema operativo Windows Server 2003 Enterprise x64 Edition. El rendimiento real y el consumo de energía varían en función de la variabilidad en la configuración, el uso y la fabricación.  
<sup>3</sup> Soporte de disco duro SATA de 250 GB en Q3CY06.  
<sup>4</sup> Este término no conlleva una velocidad de funcionamiento real de 1 GB/seg. Para la transmisión de alta velocidad se necesita una conexión a un servidor Gigabit Ethernet e infraestructura de red.

Dell no se hace responsable de ningún error tipográfico ni fotográfico. Dell, el logotipo de Dell y PowerEdge son marcas comerciales de Dell Inc. Intel y Xeon son marcas comerciales registradas de Intel Corporation. PCI Express es una marca comercial y PCI-X es una marca comercial registrada de PCI-SIG. El resto de marcas registradas y nombres comerciales se pueden usar en este documento para hacer referencia a entidades que reclaman las marcas, los nombres o sus productos. Dell renuncia a cualquier interés en la propiedad de las marcas y los nombres de terceros.  
© Copyright 2006 Dell Inc. Todos los derechos reservados. Queda totalmente prohibida cualquier tipo de reproducción sin el permiso por escrito de Dell Inc. Para obtener más información, póngase en contacto con Dell. Mayo de 2006.