Pràctica Kerberos

Curs 2019-2020

Authenticació Kerberos

Pràctica1

Imatges Docker

edtasixm11/k19:kserver servidor kerberos detach. Crea els principals pere(kpere) pau(kpau, rol: admin), jordi(kjordi), anna (kanna), marta (kmarta), marta/admin (kmarta rol:admin), julia (kjulia) i admin (kadmin rol:admin). Crear també els principals kuser01...kuser06 amb passwd (kuser01...kuser06). Assignar-li el nom de host: *kserver.edt.org*.

edtasixm11/k19:khost host client de kerberos. Simplement amb eines kinit, klist i kdestroy (no pam). El servidor al que contacta s'ha de dir *kserver.edt.org*. Cal verificar el funcionament de kadmin.

Authenticació

edtasixm11/k19:khostp host amb PAM de kerberos. El servidor al que contacta s'ha de dir *kserver.edt.org*. Aquest host configura el *system-auth* de pam per usar el mòdul *pam_krb5.so*. Crear els usuris local01..local06 (idem nom de passwd) i kuser01..kuser06 (sense passwd). Aquest host utlitza /etc/passwd de IP Information Provider i valida usuaris locals local01... amb pam_unix.so (on /etc/passwd fa de IP i AP) i usuaris locals+principals kuser01... (on /etc/passwd fa de IP i kerberos de AP Authentication Provider).

Verificació:

En una sessió interactiva en el container khostp iniciar amb “su -” sessió com a local01, convertr-se en altre cop amb “su -” en local02 i finalment convertir-se en kuser01. Validar que kuser01 obté un ticket i que pot accedir amb kadmin a l’administració del servidor kerberos (amb independència dels permisos que tingui).

Instal·lació

Eliminar del vostre host físic les particions sda2, sda3 i sda4. Crear una partició sda2 de 8GB. Instal·lar-hi Fedora-27 amb una instal·lació **MINIMAL**.
Refer el GRUB deixant per defecte la partició matí, les etiquetes MATI, TARDA i HISX2-LAB. Cal que el grub que mani (i el fitxer grub.conf) sigui el del matí.

Engegar la màquina a la partició matí (sda5) Fer:

- # cp /boot/grub2/grub.cfg /boot/grub2/grub.hisx2
- # grub2-mkconfig > /boot/grub2/grub.cfg
- # vim /boot/grub2/grub.cfg (veure què cal modificar)
- # grub2-install /dev/sda

Cal modificar:

- set timeout=-1
- set default=0 (o el número corresponent a l'entrada del matí, comencen per zero)
- MATI (posem aquesta etiqueta a la partició matí sda5)
- TARDA (posem aquesta etiqueta a la partició tarda sda6)
- HISX2-LAB (posem aquesta etiqueta a la partició de treball hisx2 sda2)

Pràctica2

Host aula + Kerberos + AWS EC2

Usarem un host real de l'aula, la partició on hem instal·lat un Fedora 27 MINIMAL. Cal configurar la autenticació dels usuaris utilitzant Unix i Kerberos. El servidor kserver.edt.org estarà desplegat a AWS EC2.

Caldrà configurar una AMI a AWS EC2 amb docker i executar el kserver fent un mapping dels ports de kerberos al host de Amazon AWS EC2. També caldrà configurar el firewall. Per fer-ho crearem un *Security groups* propi anomenat *kerberos* que obri els ports del firewall per poder accedir des de l'exterior al kerberos i al ssh. Identifica els ports i de quin tipus són.

Penseu en tot el què cal configurar en el host de l'aula, podeu consultar la configuració que fem en iniciar el curs i planxar els ordinadors a fedora@inf.

En especial cal:

- Selinux
- Authconfig
- <trick>

Problema amb el caché KCM de kerberos:

Problema: pam_krb5[10992]: error updating ccache "KCM:"
Solució:

- Comentar la línia que defineix que utilitzi KCM de caché.
- ull que està en un altre fitxer en els sistemes reals, en els containers no hi és, per això van.
- /etc/krb5.conf.d/kcm_default_ccache

 # cat /etc/krb5.conf.d/kcm_default_ccache

```
# This file should normally be installed by your distribution into a
# directory that is included from the Kerberos configuration file (/etc/krb5.conf)
# On Fedora/RHEL/CentOS, this is /etc/krb5.conf.d/
#
# To enable the KCM credential cache enable the KCM socket and the service:
#   systemctl enable sssd-secrets.socket sssd-kcm.socket
#   systemctl start sssd-kcm.socket
#
# To disable the KCM credential cache, comment out the following lines.

[libdefaults]
        #default_ccache_name = KCM:
```

Pràctica3

Kerberos + LDAP (PAM)

Farem un nou container host client de kerberos i de ldap per verificar que sabem fer un muntatge equivalent al de l'escola. En aquest esquema usem dos containers servidors, un de kerberos i un de ldap (ja els tenim fets). Cal crear el container host client que es descriu a continuació.

edtasixm11/k19:khostpl (khost-pam-ldap) host amb PAM amb autenticació AP de kerberos i IP de ldap. El servidor kerberos al que contacta s'ha de dir *kserver.edt.org*. El servidor ldap s'anomena ldap.edt.org. Aquest host es configura amb *authconfig* (us ajudarà saber que és una configuració mimètica a la que fem en realitzar la instal·lació de les aules)..

Verificar en el host client l'autenticació d'usuaris locals i usuaris globals (ldap+kerberos).
En el host client hi ha usuaris locals (local01…) usuaris locals amb passwd al kerberos (kuser01, etc que en realitat podem eliminar o ignorar) i usuaris de ldap (pere…, user1…). Aquests usuaris cal que tinguin password al kerberos (tipus kpere, kuser01, etc).

Host Aula + Kerberos + LDAP + AWS EC2

Configurar el host de l'aula amb Fedora-27-Minimal per tal de permetre l'autenticació d'usuaris locals amb pam_unix.so i usuaris globals kerberos+ldap. Cal utilitzar *authconfig*. Verificar l'accés d'usuaris locals local01,etc i d'usuaris globals pere, user01, etc.

nota: no confongueu els usuaris de ldap user01 amb els de 'mentida' que vam crear localment al lclient anomenats kuser01.

Caldrà configurar una AMI a AWS EC2 amb docker i executar el kserver.edt.org i el ldap.edt.org fent un mapping dels ports de kerberos i ldap al host de Amazon AWS EC2. També caldrà configurar el firewall. Per fer-ho crearem un *Security groups* propi anomenat

kerberos-ldap que obri els ports del firewall per poder accedir des de l'exterior al kerberos i al ssh. Identifica els ports i de quin tipus són.

Authconfig

Practiqueu la utilització de les opcions --savebackup i --restorebackup de l'ordre authconfig. Recordeu que vam treballar aquesta ordre al fer PAM (HowTo-ASIX_PAM.pdf)a ASIX-M06. Permet desar i restaurar configuracions a /var/lib/authconfig/<nom>.

Creeu tres configuracions amb authconfig:

- ❏ Estàndard unix (la que venia per defecte).
- ❏ Unix amb Kerberos (corresponent a la pràctica 2).
- ❏ Unix, Kerberos i Ldap (corresponent a la pràctica 3).

Seveis Kerberitzats

Pràctica 4

Servei SSH Kerberitzat Bàsic

<salteu al següent exercici si heu fet completament la Pràctica 3 i ja disposeu de un container amb autenticació Kerberos+ldap.>

edtasixm11/k19:sshd Servidor SSHD *kerberitzat*. Servidor ssh que permet l'accés d'usuaris locals i usuaris locals amb autenticació kerberos. El servidor s'ha de dir *sshd.edt.org*.

Primera versió simple (podem usar de base k19:khost) d'un host amb usuaris locals (local01…) i usuaris locals amb passwd al kerberos (kuser01…). A aquest host li afegim el servei ssh per convertir-se en un servidor SSH Kerberitzat. Ha de permetre l'accés tant a usuaris locals (local01) com a usuaris kerberos (kuser01).

El model de funcionament és disposar de un host client de kerberos, per exemple k19:khost i aquest servidor sshd kerberitzat. En el client un usuari 'qualsevol' es pot connectar i iniciar sessió al servidor SSH com a usuari destí local (local01).

En el client un usuari que disposi de ticket kerberos (per exemple kuser01) pot iniciar sessió remota al servidor ssh com a usuari kuser01 automàticament, ja que disposa de les credencials kerberos (similar a iniciar sessió desatesa amb claus pública/privada).

Servei SSH Kerberitzat (Kerberos + LDAP)

Si ja heu fet la Pràctica 3 i heu construït un host amb autenticació kerberos i Ldap que únicament disposa dels usuaris locals local01… i la resta els autentica via Kerberos (AP) i Ldap (IP), podeu usar de base aquesta imatge que s'anomenava k19:khostpl.

L'objectiu és crear un servidor sshd que simplement disposa dels usuaris locals (local01…) i dels usuaris de xarxa (kerberos+ldap). Aquest servidor permet que es connectin remotament tant usuaris locals com usuaris de xarxa.

Als usuaris que disposen d'un ticket de kerberos han de poder fer login automàticament (sense que se'ls demani el password). Per fer-ho caldrà configurar SSH per actuar com un servidor kerberitzat. Podeu consultar als apunts (How-to-ASIX_kerberos.pdf) con configurar un servidor kerberitzat. Bona sort amb l'aprenentatge del Keytab!.

edtasixm11/k19:sshdpl (sshd-pam-kerberos-ldap) Servidor SSH amb PAM amb autenticació AP de kerberos i IP de ldap. El servidor kerberos al que contacta s'ha de dir *kserver.edt.org*. El servidor ldap s'anomena ldap.edt.org. Aquest host es configura amb

authconfig . S'ha generat partint del host edtasixm11/k19:khostpl i se li ha afegit la part del servidor sshd. Conté els fitxers per poder activar el mount del home samba, però no s'ha configurat.

Desplegament SSH a AWS EC2

Desplegeu tots els servidors en una màquina AWS EC2. Cal engegar-hi kserver.edt.org, ldap.edt.org i sshd.edt.org. Poseu atenció a la redirecció de ports necessària per accedir al servei sshd, no podem usar el port 22 perquè és el que ens permet accedir a la AMI. Useu el port 1022 del host AMI per poder accedir al servei sshd (port 22) del container. Genereu un nou Security Groups anomenat kerberos-ldap-sshd.

Recordeu que en el host client també cal configurar el client SSH per indicar-li que utilitzi Kerberos/GSSAPI. Cal que quan usem l'ordre SSH client aquesta transmeti automàticament les credencials de kerberos (si n'hi han).

Recordeu també de configurar apropiadament el fitxers client /etc/hosts indicant els FQDN dels servidors, començant per el sshd.edt.org.

Verifiqueu

- ❏ Des d'un client container host que podeu fer login i podeu fer sessions remotes al sshd un cop disposeu de tiquets de kerberos.
- ❏ Ídem des del host real de l'aula.

Pràctica 5

Afegim Samba

edtasixm11/k19:sshdpls (sshd-pam-kerberos-ldap-home-samba) Servidor SSH amb PAM (kerberos+ldap) que munta els homes dels usuaris (dins del home) via samba.

Samba
edtasixm11/k19:khostpls (khost-pam-ldap-samba) Conté els fitxers per activar el mount del home samba, que munta els homes dels usuaris (dins del home) via samba. Caldrà crear un volum amb els homes dels usuaris. Primer el farem manualment hardcoded i després amb un script de creació.

Volumes / Entrypoint / Scripts

Pràctica 6

Volumes: krb5-data i ldap-data

Volumes krb5-data
Desar la base de dades en un volum anomenat *krb5-data* de manera que les dades de kerberos siguin perdurables. Practiqueu amb kadmin des del client i amb un compte d'administració crear, modificar, esborrar i llistar principals (manteniu els per defecte).

Practiqueu a assignar permisos diferents als usuaris, en especial el de poder llistar els principals.

Consulteu els apartats:

- ❏ Docker exec amb ladmin-local
- ❏ Kerberos i volumes
- ❏ Httpd i volums

Volumes ldap-data
La base de dades ldap es desa en un volum anomenat *ldap-data*.
La configuració ldap es desa en un volum anomenat *ldap-config*.

Entrypoint: kserver i ldap

Consulteu l'apartat:

- ❏ Entrypoint versus CMD

Entrypoint kserver
Modificar l'script startup.sh del servidor Kerberos per actuar com a entrypoint amb els següents arguments possibles:

- *res*: engegar el servei kerberos usant la base de dades existent actualment (el volum).
- *Initdb*: inicialitza la base de dades.¡ i engega el servei.
- *initdbedt*: inicialitza la base de dades de kerberos amb els principals per defecte i engega el servei.
- *kadmin*: executa kadmin-local passant-li la resta de parametres que es rebin en l'execució del container.

Entrypoint ldap

Modificar la imatge ldapserver:latest (*ldapserver:entrypoint*) de manera que tingui un script startup.sh de entrypoint que permeti:

- *initdb*: inicialitzar la base de dades ldap sense dades i engegar el servei.
- *initdbedt*: initiclitzar la base de dades ldap amb les dades per defecte usuals i engegar el servei.
- *listdn*: llistar els dn de la base de ddaes ldap usant una comanda de baix nivell "slapcat | grep dn".
- *start*: engegar el servei ldap (la base de dades, amb dades o sense, ha d'existir prèviament). Aquesta és l'opció per defecte.
- *: qualsevol altre conjunt de paràmetres que es passin com a CMDi s'executarà usant eval.

Cal usar un volume anomenat *ldap-data* per a les dades i un volume anomenat *ldap-config* per a la configuració ldap. L'script startup.sh ha de mostrar un debug de tot el que va fent si la *variable d'entorn DEBUG* és superior a zero. Cal passar aquesta variable amb l'ordre docker run amb un valor de 1 per verificar el funcionament de l'script.

Entrypoint kserver useradd/userdel

Ampliar l'script d'administració startup.sh del keserver de manera que contingui les opcions:

- *useradd*: rep les dades necessaries per crear un principal i una entrada d'usuari ldap.
- *userdel*: rep les dades necessaries per eliminar un usuari (principal i entrada ldap).
- *list*: llista els principals.

Kerberos orchestration: docker-compose / swarm

Pràctica 7

docker -compose

Desplegar la app d'autenticació (kerberos + ldap + sshd) en un host AMI de AWS EC2 usant un fitxer docker-compose.yml. Usar el Security Groups anomenat kerberos-ldap-sshd creat
previament.

Consulta l'apartat:

- ❏ docker-compose
- ❏ Docker-compose repliques / scale / deploy

docker-swarm local

Crear un swarm de 2 nodes usant els dos hosts de l'aula que teniu assignats. Desplegar-hi la app d'autenticació (kerberos+ldap+sshd).

Modificar en calent el desplegament fet de l'stack de la app i i afegir-hi un ***visualizer*** (port 8080) per monitorar el desplegament dels nodes i containers.

Modificar en calent el desplegament fet de l'stack de la app i i afegir-hi un ***portainer*** (port 9000) per monitorar el desplegament dels nodes i containers.

Observeu el visualizer i els serveis desplegats en l'stack. Practiqueu modificar l'estat dels nodes (*active*|*paused*|*drain*) i establir *constraints* de col·locació dels serveis (usant *rols* i *labels*).

Consulta els apartats:

- ❏ Docker Swarm

docker-swarm AWS EC2

Desplegar en dues (o tres!) màquines AMI de AWS EC2 la app d'autenticació (kerberos+ldap+sshd) més el visualizer (això dependrà de si les AMI 'aguanten' la càrrega, en general posar-hi el portainer és mala idea...).

Verifiqueu el funcionament des del host client local del funcionament de la app i observeu el desplegament fet amb el visualizer.

Feu modificacions al desplegament modificant els serveis, l'estat dels nodes i establint *constraints* per *rol* i *label*.

docker-swarm local + AWS

Passos a fer:

Desplegament replicated

1. Genereu un swarm amb dues màquines AMI de AWS i dos hosts de l'aula, quatre en total. El manager ha de ser un dels nodes AWS, perquè?.
   Utilitzeu apropidamament l'opció de docker swarm --advertise-ip.
2. Genereu un security groups anomenat *swarm-hello-visualizer* que obri els ports necessaris per a la comunicació dels nodes que formen el swarm i dels serveis visualizer i hello.
3. Modifiqueu el servei hello de manera que la pàgina web que mostra index.html contingui el nom del host, per exemple amb un echo $(hostname) en l'script install.sh.
4. Desplegueu al swarm el servei visualizer amb una constraint que el fixi al node manager i cinc rèpliques del servei hello. Observeu com es distribueixen amb el visualizer.
5. Modifiqueu el desplegament variant el número de rèpliques tant manualment amb l'ordre docker service scale com fent noves versions del deploy.

Global

6. Modific el servei hello passant-lo de replicated a global. Observa com es desplega a tots els nodes.

Nodes

7. Torna a fer el desplegament del servei hello amb replicated amb 6 repliques. Pausa un dels nodes. Fes drain a un dels nodes. Torna a fer actiu un dels nodes.
   Es redistribueixen les rèpliques? Com ho fem?
8. Elimina un dels nodes (un de local) del swarm. Observa el desplegament. Torna'l a afegir al swarm.

Constraints

9. Posa etiquetes *lloc* als nodes AWS amb el valor *aws* i als nodes locals amb el valor *local*.
10. Desplega el servei hello únicament a nodes amb etiquetes *local*.
11. Posa el labe *tipus* amb el valorl *alfa* al node manager i a un dels hosts locals. Posa l'etiqueta *beta* als altres dos nodes.
12. Desplega el servei hello als nodes beta. Ibserva el desplegament amb visualizer.
13. Autra el desplegament.

Teoria

Autenticaction Provider AP

Kerberos proporciona el servei de proveïdor d'autenticació. No emmagatzema informació dels comptes d'usuari com el uid, git, shell, etc. Simplement emmagatzema i gestiona els passwords dels usuaris, en entrades anomenades *principals* en la seva base de dades.

Coneixem els següents AP:

- */etc/passwd* que conté els password (AP) i també la informació dels comptes d'usuari (IP).
- *ldap* el servei de directori ldap conté informació dels comptes d'usuari (IP) i també els seus passwords (AP).
- *kerberos* que únicament actua de AP i no de IP.

Information Provider IP

Els serveis que emmagatzemen la informació dels comptes d'usuari s'anomenen Information providers. Aquests serveis proporcionen el uid, gid, shell, gecos, etc. Els clàssics són /etc/passwd i ldap.

Model de pràctiques

El model que mantindrem a tot el mòdul ASIX M11-SAD és el següent:

- **ldap** al servidor ldap tenim els usuaris habituals pere, marta, anna, julia, pau, jordi. El seu password és el seu propi nom.

- **/etc/passwd** en els containers hi ha els usuaris locals local01, local02 i local03 que tenen assignat com a password el seu mateix nom.

- **kerberos + IP** els usuaris kuser01, kuser02 i kuser03 són principals de kerberos amb passwords tipus kuser01, kuser02 i kuser03. La informació del seu compte d'usuari és local al */etc/passwd* on **no** tenen password assignat.

- **kerberos + ldap** Al servidor kerberos hi ha també principals per als usuaris usuals ldap pere, marta, anna, julia, jordi i pau. Els seus passwords són del tipus kpere, kmarta, kanna, kjulia, kjordi i kpau.

Es resum, podem verificar l'accés/autenticació d'usuaris locals usant el prototipus *local01*, podem fer test de la connectivitat kerberos amb comptes locals amb usuaris tipus *kuser01*. I

finalment podem verificar l'autenticació d'usuaris kerberos amb ldap (fent de IP) amb els clàssics pere (kpere).

Apèndix

Krb5_Cache

Problema: pam_krb5[10992]: error updating ccache "KCM:"
Solució:

- Comentar la línia que defineix que utilitzi KCM de caché.
- ull que està en un altre fitxer en els sistemes reals, en els containers no hi és, per això van.
- /etc/krb5.conf.d/kcm_default_ccache

```
# cat /etc/krb5.conf.d/kcm_default_ccache
# This file should normally be installed by your distribution into a
# directory that is included from the Kerberos configuration file (/etc/krb5.conf)
# On Fedora/RHEL/CentOS, this is /etc/krb5.conf.d/
#
# To enable the KCM credential cache enable the KCM socket and the service:
#   systemctl enable sssd-secrets.socket sssd-kcm.socket
#   systemctl start sssd-kcm.socket
#
# To disable the KCM credential cache, comment out the following lines.

[libdefaults]
        #default_ccache_name = KCM:
```

/etc/krb5.conf (afegim la secció)

```
[appdefaults]
 debug=true
 debug_sensitive=true
 ccache_dir=/tmp
 cred_session=false
```

journalcrl -f

```
feb 25 16:58:19 asus unix_chkpwd[10344]: password check failed for user (pere)
feb 25 16:58:19 asus su[10342]: pam_unix(su:auth): authentication failure; logname=root uid=1007 euid=0 tty=pts/1 ruser=local01 rhost=  user=pere
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: flag: debug
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: flag: debug_sensitive
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: flag: don't always_allow_localname
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: flag: no ignore_afs
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: flag: no null_afs
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: flag: no cred_session
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: flag: no ignore_k5login
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: flag: user_check
```

```
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: will try previously set password first
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: will ask for a password if that fails
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: will let libkrb5 ask questions
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: flag: no use_shmem
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: flag: no external
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: flag: multiple_ccaches
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: flag: validate
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: flag: warn
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: banner: Kerberos 5
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: ccache dir: /tmp
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: ccname template: KCM:
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: keytab: FILE:/etc/krb5.keytab
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: token strategy: 2b
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: called to authenticate 'pere', configured realm 'EDT.ORG'
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: authenticating 'pere@EDT.ORG'
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: trying previously-entered password for 'pere', allowing libkrb5 to prompt for more
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: authenticating 'pere@EDT.ORG' to 'krbtgt/EDT.ORG@EDT.ORG'
feb 25 16:58:19 asus su[10342]: pam_krb5[10342]: attempting with password="kpere"
feb 25 16:58:29 asus su[10342]: pam_krb5[10342]: krb5_get_init_creds_password(krbtgt/EDT.ORG@EDT.ORG) returned 0 (Success)
feb 25 16:58:29 asus su[10342]: pam_krb5[10342]: validating credentials
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: error reading keytab 'FILE:/etc/krb5.keytab'
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: TGT verified
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: got result 0 (Success)
feb 25 16:58:34 asus su[10347]: pam_krb5[10347]: no need to create "/tmp"
feb 25 16:58:34 asus su[10347]: pam_krb5[10347]: created ccache "FILE:/tmp/krb5cc_1010_i7hnfq"
feb 25 16:58:34 asus su[10347]: pam_krb5[10347]: created ccache 'FILE:/tmp/krb5cc_1010_i7hnfq' for 'pere'
feb 25 16:58:34 asus su[10347]: pam_krb5[10347]: krb5_kuserok() says "true" for ("pere@EDT.ORG","pere")
feb 25 16:58:34 asus su[10347]: pam_krb5[10347]: destroyed ccache "FILE:/tmp/krb5cc_1010_i7hnfq"
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: 'pere@EDT.ORG' passes .k5login check for 'pere'
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: authentication succeeds for 'pere' (pere@EDT.ORG)
feb 25 16:58:34 asus audit[10342]: USER_AUTH pid=10342 uid=1007 auid=0 ses=4 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
msg='op=PAM:authentication grantors=pam_krb5 acct="pere" exe="/usr/bin/su" hostname=asus addr=? terminal=pts/1 res=success'
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: pam_authenticate returning 0 (Success)
feb 25 16:58:34 asus audit[10342]: USER_ACCT pid=10342 uid=1007 auid=0 ses=4 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
msg='op=PAM:accounting grantors=pam_unix acct="pere" exe="/usr/bin/su" hostname=asus addr=? terminal=pts/1 res=success'
feb 25 16:58:34 asus su[10342]: (to pere) root on pts/1
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: debug
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: debug_sensitive
feb 25 16:58:34 asus audit[10342]: CRED_ACQ pid=10342 uid=1007 auid=0 ses=4 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
msg='op=PAM:setcred grantors=pam_krb5 acct="pere" exe="/usr/bin/su" hostname=asus addr=? terminal=pts/1 res=success'
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: don't always_allow_localname
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: no ignore_afs
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: no null_afs
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: no cred_session
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: no ignore_k5login
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: user_check
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: will try previously set password first
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: will ask for a password if that fails
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: will let libkrb5 ask questions
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: no use_shmem
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: no external
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: multiple_ccaches
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: validate
feb 25 16:58:34 asus audit[10342]: USER_START pid=10342 uid=1007 auid=0 ses=4 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
msg='op=PAM:session_open grantors=pam_unix acct="pere" exe="/usr/bin/su" hostname=asus addr=? terminal=pts/1 res=success'
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: warn
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: banner: Kerberos 5
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: ccache dir: /tmp
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: ccname template: KCM:
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: keytab: FILE:/etc/krb5.keytab
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: token strategy: 2b
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: debug
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: debug_sensitive
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: don't always_allow_localname
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: no ignore_afs
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: no null_afs
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: no cred_session
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: no ignore_k5login
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: user_check
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: will try previously set password first
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: will ask for a password if that fails
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: will let libkrb5 ask questions
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: no use_shmem
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: no external
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: multiple_ccaches
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: validate
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: flag: warn
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: banner: Kerberos 5
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: ccache dir: /tmp
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: ccname template: KCM:
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: keytab: FILE:/etc/krb5.keytab
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: token strategy: 2b
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: creating ccache for 'pere', uid=1010, gid=1010
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: error creating ccache using pattern "KCM:"
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: error creating ccache for user "pere"
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: failed to create ccache for 'pere'
feb 25 16:58:34 asus su[10342]: pam_krb5[10342]: pam_sm_open_session returning 14 (Cannot make/remove an entry for the specified session)
feb 25 16:58:34 asus su[10342]: pam_unix(su:session): session opened for user pere by root(uid=1007)
```

Docker exec i kadmin

Exemples de modificar la base de dades de kerberos externament amb docker exec i les ordres kadmin i kadmin.local:

- ❏ Kadmin.local: ordre de servidor que actua directament al backend, el servei no està engegat (no cal) i no s'utilitza el protocol kerberos. Autenticació via drets d'usuari root.

- ❏ Kadmin: ordre client per connectar amb una identitat existent a la base de dades de principals, utilitza el protocol de xarxa de Kerberos, els serveis han d'estar engegats.

  nota realitzeu les accions següents amb kadmin.local en lloc d'utilitzant kadmin.

Objectius:

- Modificar la base de dades de principals externament amb kadmin.local i kadmin.
- Observar que les dades són efímeres, finalitzen en acabar el container. No perduren.
- Definir diferents rols i permisos als usuaris amb el fitxer kadmin5.acl.
- Practicar ordres en el mode comanda d'administració: listrpincs, addprinc, delprinc, getprinc, changepasswd, modprinc.

$ docker run --rm --name kserver.edt.org -h kserver.edt.org -p 88:88 -p 749:749 -p 464:464 --net mynet -d edtasixm11/k19:kserver

```
[fedora@ip-172-31-90-235 docker-compose]$ docker ps
CONTAINER ID        IMAGE                    COMMAND                  CREATED
STATUS              PORTS                                            NAMES
45521610bb19        edtasixm11/k19:kserver   "/opt/docker/startup…"   44 seconds ago
       Up 43 seconds       0.0.0.0:88->88/tcp, 0.0.0.0:464->464/tcp,
0.0.0.0:749->749/tcp   kserver.edt.org
```

[fedora@ip-172-31-90-235 docker-compose]$ docker exec -it kserver.edt.org kadmin -p admin -q listprincs

```
Authenticating as principal admin with password.
Password for admin@EDT.ORG:
K/M@EDT.ORG
admin@EDT.ORG
anna@EDT.ORG
host/sshd.edt.org@EDT.ORG
jordi@EDT.ORG
```

[fedora@ip-172-31-90-235 docker-compose]$ docker exec -it kserver.edt.org kadmin -p admin -w kadmin -q "addprinc -pw kzztop zztop"

Authenticating as principal admin with password.
WARNING: no policy specified for zztop@EDT.ORG; defaulting to no policy
Principal "zztop@EDT.ORG" created.

[fedora@ip-172-31-90-235 docker-compose]$ docker exec -it kserver.edt.org kadmin -p admin -q "listprincs zztop"
Authenticating as principal admin with password.
Password for admin@EDT.ORG:
zztop@EDT.ORG

[fedora@ip-172-31-90-235 docker-compose]$ docker exec -it kserver.edt.org kadmin -p admin -w kadmin -q "delprinc zztop"
Authenticating as principal admin with password.
Are you sure you want to delete the principal "zztop@EDT.ORG"? (yes/no): yes
Principal "zztop@EDT.ORG" deleted.
Make sure that you have removed this principal from all ACLs before reusing.

[fedora@ip-172-31-90-235 docker-compose]$ docker exec -it kserver.edt.org kadmin -p pere -w kpere -q "listprincs"
Authenticating as principal pere with password.
get_principals: Operation requires ``list'' privilege while retrieving list.

[fedora@ip-172-31-90-235 docker-compose]$ docker exec -it kserver.edt.org vi /var/kerberos/krb5kdc/kadm5.acl

[fedora@ip-172-31-90-235 docker-compose]$ docker exec -it kserver.edt.org cat /var/kerberos/krb5kdc/kadm5.acl
*/admin@EXAMPLE.COM   *
admin@EDT.ORG *
super@EDT.ORG *
pere@EDT.ORG cl

[fedora@ip-172-31-90-235 docker-compose]$ docker exec -it kserver.edt.org kadmin -p pere -w kpere -q "listprincs"
Authenticating as principal pere with password.
get_principals: Operation requires ``list'' privilege while retrieving list.

Atenció: no va perquè cal reinicar el servei i no podem fer-ho sense matar el container

[ecanet@lenovo k19]$ docker run --rm --name kserver.edt.org -h kserver.edt.org --net mynet -it edtasixm11/k19:kserver /bin/bash

[root@kserver docker]# cat kadm5.acl
*/admin@EXAMPLE.COM   *
admin@EDT.ORG *
super@EDT.ORG *
pere@EDT.ORG cl

```
marta@EDT.ORG cdl

[root@kserver docker]# bash install.sh

[root@kserver docker]# /usr/sbin/krb5kdc
[root@kserver docker]# /usr/sbin/kadmind

[root@kserver docker]# kadmin -p pere
Authenticating as principal pere with password.
Password for pere@EDT.ORG:
kadmin:  listprincs
K/M@EDT.ORG
admin@EDT.ORG
anna@EDT.ORG
host/sshd.edt.org@EDT.ORG
jordi@EDT.ORG
julia@EDT.ORG
...
```

Kerberos i volumes

Si volem que les dades dels principals perdurin el més encertat és usar named volumes, creant un volum de dades anomenat per exemple krb5-data.

De fet podem desar les dades de /var/kerberos/krb5kdc a:

- ❏ Un directori del host usant un bind mount
- ❏ Un volum de docker.

Objectius:

- Crear, llistar, i eliminar named volumes.
- Fer inspect de volums i observar la ruta física al host (i llistar) de les dades corresponents al volum. Automatitzar backups de les dades dels volumes.
- Muntar al container named volumes o bind volumes.
- Observar que les dades són perdurables, queden emmagatzemades al volum.

Named volumes

```
$ docker volume create krb5-data

$ docker volume ls
DRIVER              VOLUME NAME
local               kerberos


$ docker run --rm --name kserver.edt.org -h kserver.edt.org --net mynet -v
```

krb5-data:/var/kerberos -it edtasixm11/k19:kserver /bin/bash

[root@kserver docker]# bash install.sh

crear zz01 i zz02
[root@kserver docker]# kadmin.local
Authenticating as principal root/admin@EDT.ORG with password.
kadmin.local: addprinc zz02
WARNING: no policy specified for zz02@EDT.ORG; defaulting to no policy
Enter password for principal "zz02@EDT.ORG":
Re-enter password for principal "zz02@EDT.ORG":
Principal "zz02@EDT.ORG" created.

[root@kserver docker]# exit

$ docker run --rm --name kserver.edt.org -h kserver.edt.org --net mynet -v krb5-data:/var/kerberos -it edtasixm11/k19:kserver /bin/bash

[root@kserver docker]# bash install.sh

[root@kserver docker]# kadmin.local -q listprincs
Authenticating as principal root/admin@EDT.ORG with password.
K/M@EDT.ORG
admin@EDT.ORG
...
zz01@EDT.ORG
zz02@EDT.ORG

[root@kserver docker]# exit

$ docker run --rm --name kserver.edt.org -h kserver.edt.org --net mynet -v krb5-data:/var/kerberos -it edtasixm11/k19:kserver /bin/bash

[root@kserver docker]# bash install.sh

[root@kserver docker]# kadmin.local
kadmin.local: addprinc zz03
WARNING: no policy specified for zz03@EDT.ORG; defaulting to no policy
Enter password for principal "zz03@EDT.ORG":
Re-enter password for principal "zz03@EDT.ORG":
Principal "zz03@EDT.ORG" created.

[root@kserver docker]# kadmin.local -q listprincs
Authenticating as principal root/admin@EDT.ORG with password.
K/M@EDT.ORG
admin@EDT.ORG
...
zz01@EDT.ORG
zz02@EDT.ORG

```
zz03@EDT.ORG

$ docker volume inspect krb5-data
[
        {
        "CreatedAt": "2020-03-02T09:15:58+01:00",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/krb5-data/_data",
        "Name": "krb5-data",
        "Options": {},
        "Scope": "local"
        }
]

$ sudo tree /var/lib/docker/volumes/krb5-data/_data
```

```
[ecanet@lenovo k19]$ docker run --rm -v kerberos:/mnt -it alpine
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
c9b1b535fdd9: Pull complete
Digest:
sha256:ab00606a42621fb68f2ed6ad3c88be54397f981a7b70a79db3d1172b11c4367d
Status: Downloaded newer image for alpine:latest
/ # ls /mnt/
krb5     krb5kdc

/ # ls -l  /mnt/krb5kdc/
total 52
-rw-------    1 root     root            55 Feb 29 17:50 kadm5.acl
-rw-------    1 root     root           447 Feb 29 17:50 kdc.conf
-rw-------    1 root     root         32768 Feb 29 17:49 principal
-rw-------    1 root     root          8192 Feb 29 17:46 principal.kadm5
-rw-------    1 root     root             0 Feb 29 17:46 principal.kadm5.lock
-rw-------    1 root     root             0 Feb 29 17:50 principal.ok
```

Bind mount

```
$ docker run --rm -v /usr/share/man:/mnt -it alpine
/ # ls /mnt/
ca      es      id      man0p   man2    man3x   man5x   man7x   man9x   pt      ru      uk
cs      fr      it      man1    man2x   man4    man6    man8    mann    pt_BR   sk      zh
da      hr      ja      man1p   man3    man4x   man6x   man8x   nl      pt_PT   sv      zh_CN
de      hu      ko      man1x   man3p   man5    man7    man9    pl      ro      tr      zh_TW
```

```
$ mkdir /tmp/krb5
```

```
$ chmod 777 /tmp/krb5

$ docker run --rm --name kserver.edt.org -h kserver.edt.org --net mynet -v
/tmp/krb5:/var/kerberos/krb5kdc -it edtasixm11/k19:kserver /bin/bash

[root@kserver docker]# bash install
```

```
[ecanet@lenovo kerberos]$ ls -l /tmp/krb5/
total 44
-rw-r--r--. 1 root root    55 29 feb 19:02 kadm5.acl
-rw-r--r--. 1 root root   447 29 feb 19:02 kdc.conf
-rw-------. 1 root root 28672 29 feb 19:02 principal
-rw-------. 1 root root  8192 29 feb 19:02 principal.kadm5
-rw-------. 1 root root     0 29 feb 19:02 principal.kadm5.lock
-rw-------. 1 root root     0 29 feb 19:03 principal.ok
```

```
< afegir manualment dos usuaris zz01 zz02>

[root@kserver docker]# exit

$ docker run --rm --name kserver.edt.org -h kserver.edt.org --net mynet -v
/tmp/krb5:/var/kerberos/krb5kdc -it edtasixm11/k19:kserver /bin/bash

Authenticating as principal root/admin@EDT.ORG with password.
K/M@EDT.ORG
admin@EDT.ORG
...
zz01@EDT.ORG
zz02@EDT.ORG
```

```
[ecanet@lenovo kerberos]$ ls -l /tmp/krb5/
total 48
-rw-r--r--. 1 root root    55 29 feb 19:09 kadm5.acl
-rw-r--r--. 1 root root   447 29 feb 19:09 kdc.conf
-rw-------. 1 root root 32768 29 feb 19:08 principal
-rw-------. 1 root root  8192 29 feb 19:02 principal.kadm5
-rw-------. 1 root root     0 29 feb 19:02 principal.kadm5.lock
-rw-------. 1 root root     0 29 feb 19:09 principal.ok
```

Httpd i volums

Exemple de treballar amb un container editant interactivament des del host contingut que fa al container comportar-se diferent. Exemples clàssics:

- ❏ Desenvolupem una app de python, en el container hi ha totes les llibreries necessàries per a executar el python i tots els gadgets que necessiti, conté un volum que es munta a /app que realment és el directori del host on hi ha el codi de la app que estem desenvolupant. Modificar el codi modifica la app executant-se en el container.

- ❏ Desenvolupar un servidor qeb amb el directori de publicació muntat amb un bind mount d'un directori del host. Editar els fitxers html (entre d'altres) del host amfitrió implica canvis en la visualització de l'aplicació web.

Exemple servidor web

```
[root@lenovo k19:hello]# cat Dockerfile
FROM fedora:27
LABEL author="@edt ASIX M11-SAD"
LABEL description="SSH server amb autenticació PAM: kerb5+ldap  2019-2020"
RUN dnf -y install  httpd nmap
RUN mkdir /opt/docker
COPY * /opt/docker/
RUN chmod +x /opt/docker/startup.sh /opt/docker/install.sh
WORKDIR /opt/docker
CMD [ "/opt/docker/startup.sh" ]

[ecanet@lenovo kerberos]$ mkdir /tmp/html
[ecanet@lenovo kerberos]$ chmod 777 /tmp/html

[ecanet@lenovo k19:hello]$ docker build -t edtasixm11/k19:hello .

[ecanet@lenovo k19:hello]$ docker run --rm -v /tmp/html:/var/www/html -d
edtasixm11/k19:hello
7ad0ed9d637aa5978555da44080ad066e4f1792a200c6a1ffde9b596c0e321f6
[ecanet@lenovo k19:hello]$
[ecanet@lenovo k19:hello]$ docker ps
CONTAINER ID        IMAGE                    COMMAND              CREATED
STATUS              PORTS          NAMES
7ad0ed9d637a        edtasixm11/k19:hello   "/opt/docker/startup…"   7 seconds ago
Up 3 seconds                  inspiring_neumann

[ecanet@lenovo tmp]$ curl http://172.17.0.2:80
hola bon dia a tothom
```

```
[ecanet@lenovo html]$ ls -l /tmp/html/
total 4
-rw-r--r--. 1 root root 23 29 feb 20:04 index.html

[ecanet@lenovo html]$ su -
Password:
```

```
[root@lenovo ~]# vim /tmp/html/index.html
[root@lenovo ~]# cat /tmp/html/index.html
hola bon dia a tothom
això és una actualitzacicató
[root@lenovo ~]# curl http://172.17.0.2
hola bon dia a tothom
això és una actualització
```

Editeu la pàgina i feu-ne canvis interactivament:

```
<html>
<title> exemple de pàgina </title>
<body>
        <h1>hola bon dia a tothom</h1>
això és una actualització
<body>
</html>
```

Mentre la visualitzeu en un navegador: http://172.170.2

Entrypoint versus CMD

Docker Documentation:

Understand how CMD and ENTRYPOINT interact

Both CMD and ENTRYPOINT instructions define what command gets executed when running a container. There are few rules that describe their cooperation.

1. Dockerfile should specify at least one of CMD or ENTRYPOINT commands.
2. ENTRYPOINT should be defined when using the container as an executable.
3. CMD should be used as a way of defining default arguments for an ENTRYPOINT command or for executing an ad-hoc command in a container.
4. CMD will be overridden when running the container with alternative arguments.

CMD

```
[ecanet@lenovo k19]$ docker run --rm -it alpine
/ # uname -a
Linux ecb957911be4 4.18.19-100.fc27.x86_64 #1 SMP Wed Nov 14 22:04:34 UTC 2018
x86_64 Linux
```

```
[ecanet@lenovo k19]$ docker run --rm -it alpine date
Mon Mar  2 16:00:41 UTC 2020
```

ENTRYPOINT

```
[ecanet@lenovo k19:calendar]$ vim Dockerfile
FROM fedora:27
LABEL author="@edt ASIX M11-SAD"
LABEL description="Exemple utilització entrypoint  2019-2020"
ENTRYPOINT [ "/usr/bin/cal" ]

[ecanet@lenovo k19:calendar]$ docker build -t edtasixm11/k19:calendar .
[ecanet@lenovo k19:calendar]$ docker push edtasixm11/k19:calendar
```

Executar ell container que per defecte executa l'ordre definida al entrypoint. Usar entrypoint quan el container executa una ordre i els arguments que especifiquem al CMD del docker run volem que siguin arguments a aquesta ordre:

```
[ecanet@lenovo k19:calendar]$ docker run --rm -it edtasixm11/k19:calendar
      March 2020
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31

[ecanet@lenovo k19:calendar]$ docker run --rm -it edtasixm11/k19:calendar 2019
                                2019

      January               February              March
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
          1  2  3  4  5                  1  2            1  2
 6  7  8  9 10 11 12   3  4  5  6  7  8  9   3  4  5  6  7  8  9
13 14 15 16 17 18 19  10 11 12 13 14 15 16  10 11 12 13 14 15 16
20 21 22 23 24 25 26  17 18 19 20 21 22 23  17 18 19 20 21 22 23
27 28 29 30 31        24 25 26 27 28        24 25 26 27 28 29 30
                      31
      April                 May                   June
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
          1  2  3  4  5  6            1  2  3  4            1
 7  8  9 10 11 12 13   5  6  7  8  9 10 11   2  3  4  5  6  7  8
14 15 16 17 18 19 20  12 13 14 15 16 17 18   9 10 11 12 13 14 15
21 22 23 24 25 26 27  19 20 21 22 23 24 25  16 17 18 19 20 21 22
28 29 30              26 27 28 29 30 31     23 24 25 26 27 28 29
                      30
      July          August                September
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
          1  2  3  4  5  6            1  2  3   1  2  3  4  5  6  7
 7  8  9 10 11 12 13   4  5  6  7  8  9 10   8  9 10 11 12 13 14
14 15 16 17 18 19 20  11 12 13 14 15 16 17  15 16 17 18 19 20 21
21 22 23 24 25 26 27  18 19 20 21 22 23 24  22 23 24 25 26 27 28
28 29 30 31           25 26 27 28 29 30 31  29 30

      October               November              December
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
```

```
                1  2  3  4  5                  1  2           1  2  3  4  5  6  7
 6  7  8  9 10 11 12           3  4  5  6  7  8  9           8  9 10 11 12 13 14
13 14 15 16 17 18 19          10 11 12 13 14 15 16          15 16 17 18 19 20 21
20 21 22 23 24 25 26          17 18 19 20 21 22 23          22 23 24 25 26 27 28
27 28 29 30 31                24 25 26 27 28 29 30          29 30 31
```

[ecanet@lenovo k19:calendar]$ docker run --rm -it edtasixm11/k19:calendar cacota
cal: failed to parse timestamp or unknown month name: cacota

[ecanet@lenovo k19:calendar]$ docker run --rm -it edtasixm11/k19:calendar 06 2020

```
      June 2020
Su Mo Tu We Th Fr Sa
          1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30
```

No podem executar una altra ordre, mana entrypoint:

[ecanet@lenovo k19:calendar]$ docker run --rm -it edtasixm11/k19:calendar /bin/bash
cal: failed to parse timestamp or unknown month name: /bin/bash

[ecanet@lenovo k19:calendar]$ docker run --rm -it edtasixm11/k19:calendar date
cal: failed to parse timestamp or unknown month name: date

Redefinir un nou entrypoint, l'ordre a executar en iniciar el container:

[ecanet@lenovo k19:calendar]$ docker run --rm --entrypoint /bin/bash -it edtasixm11/k19:calendar
[root@96c88dba0017 /]# date
Mon Mar 2 16:18:05 UTC 2020
[root@96c88dba0017 /]#exit

[ecanet@lenovo k19:calendar]$ docker run --rm --entrypoint /usr/bin/date -it edtasixm11/k19:calendar
Mon Mar 2 16:18:44 UTC 2020

[ecanet@lenovo k19:calendar]$ docker run --rm --entrypoint /bin/ls -it edtasixm11/k19:calendar
bin dev home lib64   media opt   root sbin sys usr
boot etc lib   lost+found mnt   proc run   srv   tmp var

Els arguments CMD són interpretats com a paràmetres a passar a l'ordre del entrypoint:

[ecanet@lenovo k19:calendar]$ docker run --rm --entrypoint /bin/ls -it

edtasixm11/k19:calendar /root
anaconda-ks.cfg anaconda-post.log original-ks.cfg

[ecanet@lenovo k19:calendar]$ docker run --rm --entrypoint /bin/ls -it edtasixm11/k19:calendar uname -a
/bin/ls: cannot access 'uname': No such file or directory

[ecanet@lenovo k19:calendar]$ docker run --rm --entrypoint /bin/ls -it edtasixm11/k19:calendar /bin/bash
/bin/bash

Usar entrypoint quan el container executa una ordre i els arguments que especifiquem al CMD del docker run volem que siguin arguments a aquesta ordre. Exemple amb tot indicat a la línia de comandes:

[ecanet@lenovo k19:calendar]$ docker run --rm --entrypoint /usr/bin/cal -it fedora:27 2019

```
                               2019

      January               February                March
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
          1  2  3  4  5                  1  2            1  2
 6  7  8  9 10 11 12   3  4  5  6  7  8  9   3  4  5  6  7  8  9
13 14 15 16 17 18 19  10 11 12 13 14 15 16  10 11 12 13 14 15 16
20 21 22 23 24 25 26  17 18 19 20 21 22 23  17 18 19 20 21 22 23
27 28 29 30 31        24 25 26 27 28        24 25 26 27 28 29 30
                      31
       April                  May                   June
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
          1  2  3  4  5  6            1  2  3  4            1
 7  8  9 10 11 12 13   5  6  7  8  9 10 11   2  3  4  5  6  7  8
14 15 16 17 18 19 20  12 13 14 15 16 17 18   9 10 11 12 13 14 15
21 22 23 24 25 26 27  19 20 21 22 23 24 25  16 17 18 19 20 21 22
28 29 30              26 27 28 29 30 31     23 24 25 26 27 28 29
                      30
        July                 August               September
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
          1  2  3  4  5  6            1  2  3   1  2  3  4  5  6  7
 7  8  9 10 11 12 13   4  5  6  7  8  9 10   8  9 10 11 12 13 14
14 15 16 17 18 19 20  11 12 13 14 15 16 17  15 16 17 18 19 20 21
21 22 23 24 25 26 27  18 19 20 21 22 23 24  22 23 24 25 26 27 28
28 29 30 31           25 26 27 28 29 30 31  29 30

      October               November              December
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
          1  2  3  4  5                  1  2   1  2  3  4  5  6  7
 6  7  8  9 10 11 12   3  4  5  6  7  8  9   8  9 10 11 12 13 14
13 14 15 16 17 18 19  10 11 12 13 14 15 16  15 16 17 18 19 20 21
20 21 22 23 24 25 26  17 18 19 20 21 22 23  22 23 24 25 26 27 28
27 28 29 30 31        24 25 26 27 28 29 30  29 30 31
```

[ecanet@lenovo k19:calendar]$ docker run --rm --entrypoint /bin/uname -it fedora:27 -a
Linux bf290c0422b9 4.18.19-100.fc27.x86_64 #1 SMP Wed Nov 14 22:04:34 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux

docker-compose

```
[fedora@ip-172-31-90-235 docker-compose]$ docker-compose up -d
Creating network "docker-compose_mynet" with the default driver
Creating kserver.edt.org ... done
Creating sshd.edt.org ... done
Creating ldap.edt.org  ... done
[fedora@ip-172-31-90-235 docker-compose]$
```

```
[fedora@ip-172-31-90-235 docker-compose]$ docker-compose ps
          Name                 Command            State                 Ports
-------------------------------------------------------------------------------------------------------
kserver.edt.org   /opt/docker/startup.sh            Up         0.0.0.0:464->464/tcp, 0.0.0.0:749->749/tcp,
                                                                0.0.0.0:88->88/tcp
ldap.edt.org      /bin/sh -c /opt/docker/sta ...   Up         0.0.0.0:389->389/tcp
sshd.edt.org      /opt/docker/startup.sh           Up         0.0.0.0:1022->22/tcp
```

```
[fedora@ip-172-31-90-235 docker-compose]$ docker-compose top ldap
ldap.edt.org
UID     PID     PPID   C   STIME   TTY       TIME                 CMD
-----------------------------------------------------------------------------------
root   3634   3599   0   16:10   ?       00:00:00   /bin/bash /opt/docker/startup.sh
root   3827   3634   0   16:10   ?       00:00:00   /sbin/slapd -d0
[fedora@ip-172-31-90-235 docker-compose]$

[fedora@ip-172-31-90-235 docker-compose]$ docker-compose top kserver
kserver.edt.org
UID     PID     PPID   C   STIME   TTY       TIME                 CMD
-----------------------------------------------------------------------------------
root   3679   3645   0   16:10   ?       00:00:00   /bin/bash /opt/docker/startup.sh
root   3993   3679   0   16:10   ?       00:00:00   /usr/sbin/krb5kdc
root   3994   3679   0   16:10   ?       00:00:00   /usr/sbin/kadmind -nofork

[fedora@ip-172-31-90-235 docker-compose]$ docker-compose top sshd
sshd.edt.org
UID     PID     PPID   C   STIME   TTY       TIME                 CMD
-----------------------------------------------------------------------------------
root   3686   3641   0   16:10   ?       00:00:00   /bin/bash /opt/docker/startup.sh
65     3965   3686   0   16:10   ?       00:00:00   /usr/sbin/nslcd
28     3972   3686   0   16:10   ?       00:00:00   /usr/sbin/nscd
root   3985   3686   0   16:10   ?       00:00:00   /usr/sbin/sshd -D
```

```
[fedora@ip-172-31-90-235 docker-compose]$ docker-compose images
  Container              Repository              Tag      Image Id        Size
-------------------------------------------------------------------------------
kserver.edt.org   edtasixm11/k19              kserver   efa6f3dd7181   450.1 MB
```

```
ldap.edt.org    edtasixm06/ldapserver19   latest      c290f57cb792   453.8 MB
sshd.edt.org    edtasixm11/k19            sshd        6e7e79610f20   492.6 MB
```

```
[fedora@ip-172-31-90-235 docker-compose]$ docker-compose port sshd 22
0.0.0.0:1022
```

```
[fedora@ip-172-31-90-235 docker-compose]$ docker-compose push
[fedora@ip-172-31-90-235 docker-compose]$ docker-compose pull
Pulling kserver ... done
Pulling ldap    ... done
Pulling sshd    ... done
```

```
[fedora@ip-172-31-90-235 docker-compose]$ docker-compose logs sshd
Attaching to sshd.edt.org
sshd.edt.org | Changing password for user local01.
sshd.edt.org | passwd: all authentication tokens updated successfully.
sshd.edt.org | Changing password for user local02.
sshd.edt.org | passwd: all authentication tokens updated successfully.
sshd.edt.org | Changing password for user local03.
sshd.edt.org | passwd: all authentication tokens updated successfully.
sshd.edt.org | sh: /bin/domainname: No such file or directory
sshd.edt.org | getsebool:  SELinux is disabled
sshd.edt.org | Failed to connect to bus: No such file or directory
sshd.edt.org | getsebool:  SELinux is disabled
sshd.edt.org | ssh-keygen: generating new host keys: RSA DSA ECDSA ED25519
sshd.edt.org | kadmin: Cannot contact any KDC for realm 'EDT.ORG' while initializing
kadmin interface
sshd.edt.org | Authenticating as principal admin with password.
sshd.edt.org | nslcd Ok
sshd.edt.org | nscd Ok
```

```
[fedora@ip-172-31-90-235 docker-compose]$ docker-compose ps
          Name                        Command               State                        Ports
-------------------------------------------------------------------------------------------------------------
kserver.edt.org   /opt/docker/startup.sh          Up      0.0.0.0:464->464/tcp,
0.0.0.0:749->749/tcp,
                                                  0.0.0.0:88->88/tcp
ldap.edt.org      /bin/sh -c /opt/docker/sta ...   Up      0.0.0.0:389->389/tcp
sshd.edt.org      /opt/docker/startup.sh           Up      0.0.0.0:1022->22/tcp

[fedora@ip-172-31-90-235 docker-compose]$ docker-compose pause sshd
Pausing sshd.edt.org ... done

[fedora@ip-172-31-90-235 docker-compose]$ docker-compose ps
          Name                        Command               State                        Ports
-------------------------------------------------------------------------------------------------------------
```

```
kserver.edt.org   /opt/docker/startup.sh        Up       0.0.0.0:464->464/tcp,
0.0.0.0:749->749/tcp,
                                               0.0.0.0:88->88/tcp
ldap.edt.org      /bin/sh -c /opt/docker/sta ...   Up       0.0.0.0:389->389/tcp
sshd.edt.org      /opt/docker/startup.sh           Paused   0.0.0.0:1022->22/tcp
```

```
[fedora@ip-172-31-90-235 docker-compose]$ docker-compose unpause sshd
Unpausing sshd.edt.org ... done
[fedora@ip-172-31-90-235 docker-compose]$ docker-compose ps
         Name                       Command               State                    Ports
-------------------------------------------------------------------------------------------------------
kserver.edt.org   /opt/docker/startup.sh        Up       0.0.0.0:464->464/tcp,
0.0.0.0:749->749/tcp,
                                               0.0.0.0:88->88/tcp
ldap.edt.org      /bin/sh -c /opt/docker/sta ...   Up       0.0.0.0:389->389/tcp
sshd.edt.org      /opt/docker/startup.sh           Up       0.0.0.0:1022->22/tcp
```

```
[fedora@ip-172-31-90-235 docker-compose]$ docker-compose ps
         Name                       Command               State                    Ports
-------------------------------------------------------------------------------------------------------
kserver.edt.org   /opt/docker/startup.sh        Up       0.0.0.0:464->464/tcp,
0.0.0.0:749->749/tcp,
                                               0.0.0.0:88->88/tcp
ldap.edt.org      /bin/sh -c /opt/docker/sta ...   Up       0.0.0.0:389->389/tcp
sshd.edt.org      /opt/docker/startup.sh           Up       0.0.0.0:1022->22/tcp

[fedora@ip-172-31-90-235 docker-compose]$ docker-compose stop sshd
Stopping sshd.edt.org ... done

[fedora@ip-172-31-90-235 docker-compose]$ docker-compose ps
         Name                       Command               State                    Ports
-------------------------------------------------------------------------------------------------------
kserver.edt.org   /opt/docker/startup.sh        Up       0.0.0.0:464->464/tcp,
                                               0.0.0.0:749->749/tcp, 0.0.0.0:88->88/tcp
ldap.edt.org      /bin/sh -c /opt/docker/sta ...   Up       0.0.0.0:389->389/tcp
sshd.edt.org      /opt/docker/startup.sh           Exit 137
```

```
[fedora@ip-172-31-90-235 docker-compose]$ docker-compose start sshd
Starting sshd ... done
[fedora@ip-172-31-90-235 docker-compose]$ docker-compose ps
         Name                       Command               State                    Ports
-------------------------------------------------------------------------------------------------------
kserver.edt.org   /opt/docker/startup.sh        Up       0.0.0.0:464->464/tcp,
0.0.0.0:749->749/tcp,
                                               0.0.0.0:88->88/tcp
ldap.edt.org      /bin/sh -c /opt/docker/sta ...   Up       0.0.0.0:389->389/tcp
sshd.edt.org      /opt/docker/startup.sh           Up       0.0.0.0:1022->22/tcp
```

```
[fedora@ip-172-31-90-235 docker-compose]$ docker-compose scale sshd=2
WARNING: The scale command is deprecated. Use the up command with the --scale flag instead.
WARNING: The "sshd" service is using the custom container name "sshd.edt.org". Docker requires each container to have a unique name. Remove the custom name to scale the service.
WARNING: The "sshd" service specifies a port on the host. If multiple containers for this service are created on a single host, the port will clash.
Starting sshd.edt.org ... done
Creating sshd.edt.org ... error

ERROR: for sshd.edt.org  Cannot create container for service sshd: Conflict. The container name "/sshd.edt.org" is already in use by container "fbcdc03511b3510d8fb0055596b1ff839ff1b7be92bd08a097a87ab1c3440181". You have to remove (or rename) that container to be able to reuse that name.
ERROR: Cannot create container for service sshd: Conflict. The container name "/sshd.edt.org" is already in use by container "fbcdc03511b3510d8fb0055596b1ff839ff1b7be92bd08a097a87ab1c3440181". You have to remove (or rename) that container to be able to reuse that name.
```

```
[fedora@ip-172-31-90-235 docker-compose]$ docker-compose down
Stopping sshd.edt.org... done
Stopping kserver.edt.org ... done
Stopping ldap.edt.org ... done
Removing sshd.edt.org          ... done
Removing kserver.edt.org ... done
Removing ldap.edt.org          ... done
Removing network docker-compose_mynet
```

Docker-compose repliques / scale / deploy

```
[ecanet@lenovo k19:hello]$ cat docker-compose.yml
version: "3"
services:
  hello:
        image: edtasixm11/k19:hello
        ports:
        - "80"
        networks:
        - mynet
networks:
  mynet:
```

```
[ecanet@lenovo k19:hello]$ docker-compose up -d
Creating network "k19hello_mynet" with the default driver
```

```
Creating k19hello_hello_1 ... done

[ecanet@lenovo k19:hello]$ docker-compose scale hello=2
WARNING: The scale command is deprecated. Use the up command with the --scale flag
instead.
Starting k19hello_hello_1 ... done
Creating k19hello_hello_2 ... done

[ecanet@lenovo k19:hello]$ docker-compose port hello 80
0.0.0.0:32768

[ecanet@lenovo k19:hello]$ docker-compose scale hello=4
WARNING: The scale command is deprecated. Use the up command with the --scale flag
instead.
Starting k19hello_hello_1 ... done
Starting k19hello_hello_2 ... done
Creating k19hello_hello_3 ... done
Creating k19hello_hello_4 ... done

[ecanet@lenovo k19:hello]$ docker-compose port hello 80
0.0.0.0:32768
<però en realitat publica un port per a cada container>

[ecanet@lenovo k19:hello]$ docker-compose down
Stopping k19hello_hello_4 ... done
Stopping k19hello_hello_3 ... done
Stopping k19hello_hello_2 ... done
Stopping k19hello_hello_1 ... done
Removing k19hello_hello_4 ... done
Removing k19hello_hello_3 ... done
Removing k19hello_hello_2 ... done
Removing k19hello_hello_1 ... done
Removing network k19hello_mynet
```

Amb portainer:

```
$ cat docker-compose.yml
version: "3"
services:
  hello:
        image: edtasixm11/k19:hello
        ports:
        - "80"
        networks:
        - mynet
  portainer:
        image: portainer/portainer
        ports:
        - "9000:9000"
        volumes:
```

```
      - "/var/run/docker.sock:/var/run/docker.sock"
      networks:
      - mynet
networks:
  mynet:
```

```
[ecanet@lenovo k19:hello]$ docker-compose up -d
Creating network "k19hello_mynet" with the default driver
Creating k19hello_hello_1      ... done
Creating k19hello_portainer_1 ... done
[ecanet@lenovo k19:hello]$
[ecanet@lenovo k19:hello]$ docker-compose scale hello=3
WARNING: The scale command is deprecated. Use the up command with the --scale flag
instead.
Starting k19hello_hello_1 ... done
Creating k19hello_hello_2 ... done
Creating k19hello_hello_3 ... done
[ecanet@lenovo k19:hello]$ docker-compose port hello 80
0.0.0.0:32772

[ecanet@lenovo k19:hello]$ docker-compose ps
          Name                        Command               State            Ports
-------------------------------------------------------------------------------
k19hello_hello_1       /opt/docker/startup.sh   Up      0.0.0.0:32772->80/tcp
k19hello_hello_2       /opt/docker/startup.sh   Up      0.0.0.0:32774->80/tcp
k19hello_hello_3       /opt/docker/startup.sh   Up      0.0.0.0:32773->80/tcp
k19hello_portainer_1   /portainer               Up      0.0.0.0:9000->9000/tcp

[ecanet@lenovo k19:hello]$ docker-compose down
Stopping k19hello_hello_2     ... done
Stopping k19hello_hello_3     ... done
Stopping k19hello_hello_1     ... done
Stopping k19hello_portainer_1 ... done
Removing k19hello_hello_2     ... done
Removing k19hello_hello_3     ... done
Removing k19hello_hello_1     ... done
Removing k19hello_portainer_1 ... done
Removing network k19hello_mynet
```

Docker swarm

Swarm: definir conjunt de hosts 'nodes' que formen el swarm'.
Stack: definir una aplicació a desplegar en el swarm. Un stack es composa d'un conjunt de serveis. Es desplega usant un fitxer docker-compose (o un bundle) i es desplega en un orchestrator com swarm (o kubernetes). Un cop desplegat el podem modificar amb la propia ordre deploy i amb noves versions dels fitxer compose.

Service: tot stack està format per un conjunt de serveis, podem usar l'ordre service per examinar individualment els serveis de l'stack.
Container: podem fer servir totes les ordres docker standard per actuar amb els containers que formen part del desplegament del stack.
Node: els hosts que formen part del swarm.

Gestionar el swarm

manager:
$ docker swarm init
$ docker swarm join-token worker
$ docker swarm join-token manager

worker:
$ docker swarm join --token SWMTKN-1-5pz5j3duk6b83fe3htyax0td7ok3eoqbi347ejzsr3dzkmplbs-8ufvz85obbcssyvjx10dhvp7p 192.168.1.50:2377

Manager:
$ docker node ls

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE VERSION
wf4a01vm0dqv7f145zeicipj0	asus	Ready	Active		18.09.0
2a8jk6gooa4rbmz5q6iff7b3y *	lenovo	Ready	Active	Leader	18.09.0

```
version: "3"
services:
  hello:
        image: edtasixm11/k19:hello
        deploy:
          replicas: 3
        ports:
        - "80:80"
        networks:
        - mynet
  portainer:
        image: portainer/portainer
        ports:
        - "9000:9000"
        volumes:
        - "/var/run/docker.sock:/var/run/docker.sock"
        networks:
        - mynet
networks:
  mynet:
```

Gestionar Stack

manager:

$ docker stack ps myhello

```
ID             NAME                IMAGE                          NODE      DESIRED STATE  CURRENT STATE
ERROR          PORTS
thmg752wdfqj   myhello_portainer.1  portainer/portainer:latest     lenovo    Running        Starting 13 seconds ago
wu833mlt0y8e   myhello_hello.1      edtasixm11/k19:hello           asus      Running        Preparing about a minute ago

66ltcvqgsf8v   myhello_hello.2      edtasixm11/k19:hello           asus      Running        Preparing about a minute ago

np777jvwmy6s   myhello_hello.3      edtasixm11/k19:hello           lenovo    Running        Starting 11 seconds ago
```

[root@lenovo k19:hello]# docker stack ls

```
NAME            SERVICES            ORCHESTRATOR
myhello         2                   Swarm
```

[root@lenovo k19:hello]# docker stack services myhello

```
ID              NAME            MODE          REPLICAS        IMAGE
PORTS
klri1twoqvv7    myhello_portainer   replicated    1/1
portainer/portainer:latest   *:9000->9000/tcp
uy8pc0sq982n    myhello_hello       replicated    1/3
edtasixm11/k19:hello         *:80->80/tcp
```

Gestionar container

manager:

[root@lenovo k19:hello]# docker ps

```
CONTAINER ID   IMAGE                      COMMAND                  CREATED          STATUS          PORTS
NAMES
4ca3353e24a4   portainer/portainer:latest  "/portainer"             35 seconds ago   Up 27 seconds   9000/tcp
myhello_portainer.1.7s0de7mg0b6m7sd4s53vzqhcq
fcba03a9145f   edtasixm11/k19:hello       "/opt/docker/startup…"   6 minutes ago    Up 5 minutes    80/tcp
myhello_hello.3.np777jvwmy6sui6m09o3kr2rt
```

[root@lenovo k19:hello]# docker top myhello_portainer.1.7s0de7mg0b6m7sd4s53vzqhcq

```
UID             PID             PPID            C               STIME           TTY
TIME            CMD
root            5780            5763            0               19:20           ?
00:00:00        /portainer
```

[root@lenovo k19:hello]# docker top myhello_hello.3.np777jvwmy6sui6m09o3kr2rt

```
UID             PID             PPID            C               STIME           TTY
TIME            CMD
root            4799            4762            0               19:15           ?
00:00:00        /bin/bash /opt/docker/startup.sh
root            5076            4799            0               19:15           ?
00:00:00        /usr/sbin/httpd -DFOREGROUND
apache          5147            5076            0               19:15           ?
00:00:00        /usr/sbin/httpd -DFOREGROUND
apache          5148            5076            0               19:15           ?
00:00:00        /usr/sbin/httpd -DFOREGROUND
apache          5149            5076            0               19:15           ?
```

```
00:00:00    /usr/sbin/httpd -DFOREGROUND
apache      5150        5076        0        19:15        ?
00:00:00    /usr/sbin/httpd -DFOREGROUND
```

worker:

[root@asus ~]# docker ps

```
CONTAINER ID   IMAGE                  COMMAND                  CREATED              STATUS              PORTS
NAMES
e9113f66338a   edtasixm11/k19:hello   "/opt/docker/startup…"   About a minute ago   Up About a minute   80/tcp
myhello_hello.2.66ltcvqgsf8vp99dvjsewlotc
6375d7490baf   edtasixm11/k19:hello   "/opt/docker/startup…"   About a minute ago   Up About a minute   80/tcp
myhello_hello.1.wu833mlt0y8ew7i15qscmjtwv
```

Podem connectar als serveis web 80 i portainer 9000 a qualsevol dels dos hosts, amb independència de si s'hi executa el servei o no.

Gestionar Serveis

Manager:

[root@lenovo k19:hello]# **docker service**

create inspect logs ls ps rm rollback scale update

[root@lenovo k19:hello]# docker service ls

```
ID             NAME               MODE         REPLICAS   IMAGE                          PORTS
uy8pc0sq982n   myhello_hello      replicated   3/3        edtasixm11/k19:hello           *:80->80/tcp
klri1twoqvv7   myhello_portainer  replicated   1/1        portainer/portainer:latest     *:9000->9000/tcp
```

[root@lenovo k19:hello]# docker service ps myhello_hello

```
ID             NAME                IMAGE                  NODE     DESIRED STATE   CURRENT STATE            ERROR
               PORTS
wu833mlt0y8e   myhello_hello.1     edtasixm11/k19:hello   asus     Running         Running 17 minutes ago
66ltcvqgsf8v   myhello_hello.2     edtasixm11/k19:hello   asus     Running         Running 17 minutes ago
np777jvwmy6s   myhello_hello.3     edtasixm11/k19:hello   lenovo   Running         Running 21 minutes ago
```

[root@lenovo k19:hello]# docker service ps myhello_portainer

```
ID             NAME                IMAGE                      NODE     DESIRED STATE   CURRENT STATE            ERROR
               PORTS
8gsf2wmbtrp5   myhello_portainer.1  portainer/portainer:latest  lenovo   Running         Running 11 minutes ago

7s0de7mg0b6m   \_ myhello_portainer.1  portainer/portainer:latest  lenovo   Shutdown        Failed 12 minutes ago    "task: non-zero
exit (1)"
thmg752wdfqj   \_ myhello_portainer.1  portainer/portainer:latest  lenovo   Shutdown        Failed 17 minutes ago    "task: non-zero
exit (1)"
```

[root@lenovo k19:hello]# docker service inspect myhello_hello

```
[
        {
        "ID": "uy8pc0sq982nqtztuf66jsvh5",
        "Version": {
        "Index": 58
        },
        "CreatedAt": "2020-03-01T18:15:02.7932269Z",
        "UpdatedAt": "2020-03-01T18:15:03.043307335Z",
        "Spec": {
        "Name": "myhello_hello",
        "Labels": {
        "com.docker.stack.image": "edtasixm11/k19:hello",
        "com.docker.stack.namespace": "myhello"
        },
    …
        "Ports": [
        {
        "Protocol": "tcp",
        "TargetPort": 80,
```

```
                "PublishedPort": 80,
                "PublishMode": "ingress"
                }
                ],
                "VirtualIPs": [
                {
                "NetworkID": "mihuzxd504m7spxdkewoxhqgl",
                "Addr": "10.255.0.4/16"
                },
                {
                "NetworkID": "wvyptx71e7fqq92h2qiexmzw8",
                "Addr": "10.0.0.2/24"
                }
                ]
                }
                }
]
```


[root@lenovo k19:hello]# docker service logs myhello_hello

```
myhello_hello.1.wu833mlt0y8e@asus    | Ok install
myhello_hello.1.wu833mlt0y8e@asus    | AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 10.0.0.5. Set the 'ServerName' directive globally to suppress this message
myhello_hello.2.66ltcvqgsf8v@asus    | Ok install
myhello_hello.2.66ltcvqgsf8v@asus    | AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 10.0.0.3. Set the 'ServerName' directive globally to suppress this message
myhello_hello.3.np777jvwmy6s@lenovo | Ok install
myhello_hello.3.np777jvwmy6s@lenovo | AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 10.0.0.4. Set the 'ServerName' directive globally to suppress this message
```

[root@lenovo k19:hello]# docker service scale myhello_hello=4

```
myhello_hello scaled to 4
overall progress: 4 out of 4 tasks
1/4: running   [==================================================>]
2/4: running   [==================================================>]
3/4: running   [==================================================>]
4/4: running   [==================================================>]
verify: Service converged
```

[root@lenovo k19:hello]# docker stack services myhello

```
ID            NAME              MODE         REPLICAS   IMAGE                    PORTS
klri1twoqvv7  myhello_portainer  replicated   1/1        portainer/portainer:latest
*:9000->9000/tcp
uy8pc0sq982n  myhello_hello      replicated   4/4        edtasixm11/k19:hello     *:80->80/tcp
```

[root@lenovo k19:hello]# docker service scale myhello_hello=2

```
myhello_hello scaled to 2
overall progress: 2 out of 2 tasks
1/2: running   [==================================================>]
2/2: running   [==================================================>]
verify: Service converged
```

[root@lenovo k19:hello]# docker stack services myhello

```
ID            NAME              MODE         REPLICAS   IMAGE                    PORTS
klri1twoqvv7  myhello_portainer  replicated   1/1        portainer/portainer:latest
*:9000->9000/tcp
uy8pc0sq982n  myhello_hello      replicated   2/2        edtasixm11/k19:hello     *:80->80/tcp
```

[root@lenovo k19:hello]# docker stack ps myhello

```
ID            NAME              IMAGE                    NODE     DESIRED STATE  CURRENT STATE          ERROR
              PORTS
8gsf2wmbtrp5  myhello_portainer.1  portainer/portainer:latest  lenovo  Running  Running 21 minutes ago

wu833mlt0y8e  myhello_hello.1    edtasixm11/k19:hello     asus     Running  Running 28 minutes ago

np777jvwmy6s  myhello_hello.3    edtasixm11/k19:hello     lenovo   Running  Running 31 minutes ago
```

Gestionar nou Deploy:

```
version: "3"
services:
  hello:
        image: edtasixm11/k19:hello
        deploy:
        replicas: 3
        ports:
        - "80:80"
        networks:
        - mynet
  portainer:
        image: portainer/portainer
        ports:
        - "9000:9000"
        volumes:
        - "/var/run/docker.sock:/var/run/docker.sock"
        networks:
        - mynet
  visualizer:
        image: dockersamples/visualizer:stable
        ports:
        - "8080:8080"
        volumes:
        - "/var/run/docker.sock:/var/run/docker.sock"
        deploy:
        placement:
        constraints: [node.role == manager]
        networks:
        - mynet
networks:
  mynet:
```

```
[root@lenovo k19:hello]# docker stack deploy -c docker-compose.yml myhello
Updating service myhello_hello (id: uy8pc0sq982nqtztuf66jsvh5)
Updating service myhello_portainer (id: klri1twoqvv723675k9ruetca)
Creating service myhello_visualizer

[root@lenovo k19:hello]# docker stack services myhello
ID                  NAME                MODE                REPLICAS            IMAGE                             PORTS
klri1twoqvv7        myhello_portainer   replicated          1/1                 portainer/portainer:latest
*:9000->9000/tcp
uy8pc0sq982n        myhello_hello       replicated          3/3                 edtasixm11/k19:hello              *:80->80/tcp
xqosrtk4wovg        myhello_visualizer  replicated          1/1                 dockersamples/visualizer:stable
*:8080->8080/tcp
```

```
[root@lenovo k19:hello]# docker stack ps  myhello | grep Running
1bttkxp64g7j   myhello_visualizer.1   dockersamples/visualizer:stable   lenovo   Running   Running 2 minutes ago
8gsf2wmbtrp5   myhello_portainer.1    portainer/portainer:latest        lenovo   Running   Running 31 minutes ago
wu833mlt0y8e   myhello_hello.1        edtasixm11/k19:hello              asus     Running   Running 38 minutes ago
t9arb2ijwy3c   myhello_hello.2        edtasixm11/k19:hello              asus     Running   Running 4 minutes ago
np777jvwmy6s   myhello_hello.3        edtasixm11/k19:hello              lenovo   Running   Running 42 minutes ago
```

Accediu al visualizer al port 8080 del node worker (on no s'està executant).

Proveu de modificar replicas a 2 del servei hello modificant el fitxer docker-compose.yml i fent el deploy de nou. Llavors amb docker service torneu-lo a posar a 3.

Podria engegar ara un altre stack anomenat mystack2 amb els mateixos serveis?
No perquè xocaran de port!

Gestionar nodes

```
[root@lenovo k19:hello]# docker node
demote   inspect  ls         promote  ps         rm       update

[root@lenovo k19:hello]# docker node ls
ID                            HOSTNAME   STATUS   AVAILABILITY   MANAGER STATUS   ENGINE
VERSION
wf4a01vm0dqv7f145zeicipj0     asus       Ready    Active                          18.09.0
2a8jk6gooa4rbmz5q6iff7b3y *   lenovo     Ready    Active         Leader           18.09.0

[root@lenovo k19:hello]# docker node update --availability pause asus
asus

[root@lenovo k19:hello]# docker node ls
ID                            HOSTNAME   STATUS   AVAILABILITY   MANAGER STATUS   ENGINE
VERSION
wf4a01vm0dqv7f145zeicipj0     asus       Ready    Pause                           18.09.0
2a8jk6gooa4rbmz5q6iff7b3y *   lenovo     Ready    Active         Leader           18.09.0

[root@lenovo k19:hello]# docker stack services myhello
ID             NAME                MODE         REPLICAS   IMAGE                          PORTS
klri1twoqvv7   myhello_portainer   replicated   1/1        portainer/portainer:latest     *:9000->9000/tcp
uy8pc0sq982n   myhello_hello       replicated   3/3        edtasixm11/k19:hello           *:80->80/tcp
xqosrtk4wovg   myhello_visualizer  replicated   1/1        dockersamples/visualizer:stable  *:8080->8080/tcp
```

pausat, no accepta noves tasques però es continuen executant les existents

```
[root@lenovo k19:hello]# docker node update --availability drain asus
asus

[root@lenovo k19:hello]# docker node ls
ID                            HOSTNAME   STATUS   AVAILABILITY   MANAGER STATUS   ENGINE VERSION
```

```
wf4a01vm0dqv7f145zeicipj0          asus      Ready     Drain                18.09.0
2a8jk6gooa4rbmz5q6iff7b3y *   lenovo    Ready     Active    Leader     18.09.0
```

[root@lenovo k19:hello]# docker stack ps myhello | grep Running

```
xjtezfq7y06v   myhello_hello.1        edtasixm11/k19:hello                  lenovo    Running          Running 45 seconds ago
1bttkxp64g7j   myhello_visualizer.1   dockersamples/visualizer:stable   lenovo    Running          Running 25 minutes ago
8gsf2wmbtrp5   myhello_portainer.1    portainer/portainer:latest    lenovo    Running          Running about an hour ago
v7pptutf433h   myhello_hello.2        edtasixm11/k19:hello                  lenovo    Running          Running 45 seconds ago
np777jvwmy6s   myhello_hello.3        edtasixm11/k19:hello                  lenovo    Running          Running about an hour ago
```

[root@lenovo k19:hello]# docker stack services myhello

```
ID             NAME                MODE          REPLICAS    IMAGE                          PORTS
klri1twoqvv7   myhello_portainer   replicated    1/1         portainer/portainer:latest     *:9000->9000/tcp
uy8pc0sq982n   myhello_hello       replicated    3/3         edtasixm11/k19:hello           *:80->80/tcp
xqosrtk4wovg   myhello_visualizer  replicated    1/1         dockersamples/visualizer:stable   *:8080->8080/tcp
```

tots els containers que s'executaven al node worker han passat a l0altre node. Eren containerrs del servei hello perquè els serveis visualizer i portainer sempre s'estaven executant en el node manager.

[root@lenovo k19:hello]# docker node update --availability active asus
asus

[root@lenovo k19:hello]# docker stack ps myhello | grep Running

```
xjtezfq7y06v   myhello_hello.1        edtasixm11/k19:hello                  lenovo    Running          Running 4 minutes ago
1bttkxp64g7j   myhello_visualizer.1   dockersamples/visualizer:stable   lenovo    Running          Running 29 minutes ago
8gsf2wmbtrp5   myhello_portainer.1    portainer/portainer:latest    lenovo    Running          Running about an hour ago
v7pptutf433h   myhello_hello.2        edtasixm11/k19:hello                  lenovo    Running          Running 4 minutes ago
np777jvwmy6s   myhello_hello.3        edtasixm11/k19:hello                  lenovo    Running          Running about an hour ago
```

es continua tot executant en el node manager encara que torna a haver-hi els dos nodes.

[root@lenovo k19:hello]# docker node ls

```
ID                            HOSTNAME   STATUS    AVAILABILITY     MANAGER STATUS   ENGINE VERSION
wf4a01vm0dqv7f145zeicipj0     asus       Ready     Active                            18.09.0
2a8jk6gooa4rbmz5q6iff7b3y *   lenovo     Ready     Active           Leader           18.09.0
```

#?? Com es fan reorganitzar els containers en els nodes un cop afegit un nou node?

Gestionar labels i constraints

 # docker node update --label-add lloc=local asus
asus

[root@lenovo k19:hello]# docker node inspect asus

```
[
        {
```

```
"ID": "wf4a01vm0dqv7f145zeicipj0",
"Version": {
"Index": 174
},
"CreatedAt": "2020-03-01T18:07:34.335326677Z",
"UpdatedAt": "2020-03-01T19:40:00.73727218Z",
"Spec": {
"Labels": {
"lloc": "local"
},
"Role": "worker",
"Availability": "active"
},
```

```
version: "3"
services:
  hello:
        image: edtasixm11/k19:hello
        deploy:
        replicas: 3
        placement:
          constraints: [node.labels.lloc == local ]
        ports:
        - "80:80"
        networks:
        - mynet
```

```
[root@lenovo k19:hello]# docker stack deploy -c  docker-compose.yml myhello
Updating service myhello_portainer (id: klri1twoqvv723675k9ruetca)
Updating service myhello_visualizer (id: xqosrtk4wovg8r43gyrsk2dg9)
Updating service myhello_hello (id: uy8pc0sq982nqtztuf66jsvh5)

[root@lenovo k19:hello]# docker stack ps myhello | grep Run
oqbtj5w91io9        myhello_hello.1       edtasixm11/k19:hello                asus      Running        Running about a minute ago
1bttkxp64g7j        myhello_visualizer.1  dockersamples/visualizer:stable     lenovo    Running        Running about an hour ago
8gsf2wmbtrp5        myhello_portainer.1   portainer/portainer:latest          lenovo    Running        Running about an hour ago
78aeaqt5lop5        myhello_hello.2       edtasixm11/k19:hello                asus      Running        Running about a minute ago
z5awipunw6xb        myhello_hello.3       edtasixm11/k19:hello                asus      Running        Running about a minute ago
```

Tancar el stack i el swarm

```
[root@lenovo k19:hello]# docker stack rm myhello
Removing service myhello_hello
Removing service myhello_portainer
Removing service myhello_visualizer
Removing network myhello_mynet
Failed to remove network wvyptx71e7fqq92h2qiexmzw8: Error response from daemon:
rpc error: code = FailedPrecondition desc = network wvyptx71e7fqq92h2qiexmzw8 is in
```

use by task 1bttkxp64g7jwk14ealkrtum1Failed to remove some resources from stack: myhello

[root@lenovo k19:hello]# docker stack rm myhello
Removing network myhello_mynet

[root@asus ~]# docker swarm leave
Node left the swarm.

[root@lenovo k19:hello]# docker swarm leave
Error response from daemon: You are attempting to leave the swarm on a node that is participating as a manager. Removing the last manager erases all current state of the swarm. Use `--force` to ignore this message.

[root@lenovo k19:hello]# docker swarm leave --force
Node left the swarm.

El model swarm / compose

Al llarg del curs s'ha estudiat el desplegament de containers amb les ordres bàsiques de docker. Un cop tenim varis containers, networks i volumes a gestionar veiem que es poden desplegar conjuntament amb docker-compose. Quan aquest desplegament es realitza en múltiples nodes estem treballant amb swarm. La gestió de múltiples containers, networks, volumes, etc s'anomena ***orchestration***. Les dues tecnologies d'orchestration més importants actualment són *swarm* i *Kubernetes*.

docker -compose

Usem l'ordre docker-compose per desplegar aplicacions en un sol host. Per exemple l'aplicació auth (kerberos+ldap+sshd+portainer). Es composa de serveis, networks i volumes. En lloc d'engegar un a un els serveis o usar un script bash per fer-ho és molt més pràctic engegar els serveis conjuntament amb docker-compose.

Podem gestionar el desplegament amb l'ordre docker-compose, els serveis amb l'ordre docker service i els container individuals amb les ordres docker.

En un desplegament amb compose els serveis poden estar formats per un sol container o per múltiples rèpliques. Segons la versió del fitxer docker-compose s'utilitza scale, replicas, deploy, etc. Ara bé, cal tenir present que:

- No podem replicar container que tenen noms fixes assignats (hostname o name).

- No podem replicar containers que tenen mapping de port origen:destí assignats. Per exemple assignar el port 1022 del host al 22 del container. Això no té sentit si hi ha més de una rèplica del container perquè no hi ha més de un port 1022 en el host amfitrió.
  Per tant quan usem docker-compose amb rèpliques l'assignació de ports al host amfitrió serà dinàmica.
- No té sentit utilitzar volumes individuals a cada rèplica perquè no hi haurà coherència de dades. Si es desplega un kerberos amb tres rèpliques i cada un d'ells té associat un volume krb5-data, cada container té una còpia de les dades que no és coherent amb la dels altres containers.
  Caldrà un altre disseny de la app o utilitzar mecanismes d'emmagatzemament de xarxa per tal de que cada container kerberos desi les dades en un únic repositori de dades de xarxa.
- Quan es despleguen rèpliques d'un container que publica un port, per exemple el 80, Docker publica en el host amfitrió un port dinàmic per a cada rèplica, que permet un accés al port 80 del container corresponent. És una relació un a un, un port dinàmic al host amfitrió per a cada container.

Són moltes les característiques que es poden 'tunejar' en fer desplegaments amb docker-compose, i un cop fet un desplegament també podem modificar-ne la configuració. Podem, per exemple:

- Pausar, unpausar, aturar, reiniciar i iniciar serveis dins del desplegament.
- Modificar el número de rèpliques d'un servei (cal que el servei estigui 'pensat' per treballar amb rèpliques).
- Indicar característiques físiques i límits de memoria, cpu, etc que pot usar el servei.
- Establir health-checks per verificar el bon funcionament del servei i definir polítiques de reinici del servei en cas de mal funcionament o aturada.
- Observeu que no té sentit establir polítiques de col·locació perquè en aquest model no tenim nodes, el docker-compose es desplega en un únic host amfitrió.

En veure el funcionament del desplegament de docker-compose observem que:

- s'utilitza com a nom virtual de la app que estem desplegant el directori actual des d'on s'ha fet el desplegament.
- cada servei s'identifica amb un nom de tipus nomapp_nomservei.
- cada container s'identifica amb un nom tipus nomapp_nomservei_replica_id.
- cada xarxa s'identifica amb nomapp_nomxarxa.
- les xarxes que s'utilitzen són locals al desplegament i hi ha un nivell de isolació entre els containers del desplegament de manera que l'accés, els hostnames i la xarxa és compartit només entre ells.

docker-swarm

El model de docker-swarm consisteix en un conjunt d'ordinadors actuant conjuntament com un cluster, que en docker s'anomena swarm (un eixam). Cada un dels hosts que participa en un swarm s'anomena node. Els nodes poden ser hosts locals, hosts del cloud (com per

exemple màquines AMI de AWS EC2), màquines virtuals (per exemple de Virtualbox) i màquines de docker-machine (si encara no està deprecated…), tots ells es poden combinar.

Els nodes que formen un swarm contenen:

- ❏ Nodes manager i nodes Worker. Un swarm mínim és el format per un sol node que és un node manager.
- ❏ Manager: en tot swarm hi ha almenys un node manager. N'hi pot haver més. El node manager és des d'on es despleguen les ordres i manté la informació de funcionament del swarm. Si hi ha varis manager se n'escull un d'ells com a Leader (ho fa automàticament Docker aplicant l'algorisme de Raft Consensus). Tot manager és també un worker on s'hi poden desplegar containers.
- ❏ Worker: els workers són nodes on es despleguen els containers. Des d'un node worker no es pot governar el desplegament, només des del manager.

En un swarm podem desplegar una o més aplicacions que en Docker s'anomenen Stack. Així per exemple usant dos nodes podem desplegar la app "auth" que consta dels serveis ldap, kerberos, sshd i portainer. Aquesta app en llenguatge docker s'anomena stack.

Un stack correspon a un docker-compose file on indiquem els recursos: serveis; volums i xarxes a utilitzar. Quan es desplega un stack els serveis es despleguen en els nodes seguint el criteri intern d'optimització que realitzi docker (o seguint les ordres de col·locació que se li ha donat).

Un stack (conceptualment una app i físicament un fitxer docker-compose.yml) està format per serveis (a part de networks i volumes) que poden ser individuals, replicated o globals:

- ❏ Un servei individual és un servei que correspon a un únic container (o replicated.1).
- ❏ Un servei replicated és un servei que consta de múltiples rèpliques o instàncies del mateix container. Tal i com ja s'ha comentat amb anterioritat els containers han de ser dissenyats per poder funcionar apropiadament com a rèpliques.
- ❏ Un servei global és un servei que desplega un container a cada node, no s'indica el número de rèpliques sinó que significa desplegar una rèplica per a cada node del swarm.

El funcionament dels ports i la xarxa és significativement diferent de docker-compose i és un dels avantatges del swarm. Es crea una xarxa ***Ingress*** de ***routing mesh*** que permet l'accés als serveis per els ports que publiquen. Es pot accedir a tots els serveis a tots els nodes amb independència de si el container s'està executant en el node o no. Docker swarm genera una xarxa ingress que funciona de ***load balancer*** i distribueix les peticions entre tots els nodes.

- ❏ És a dir, si per exemple tenim el servei visualizer desplegat al node manager al port 8080 podem accedit al host worker al port 8080 i accedirem al servei visualizer, tot i que no s'hi executa!.
- ❏ Si en un swarm de 3 nodes tenim un apache httpd desplegat amb dos repliques a un node, tres rèpliques a un altre node i cap replica a l'últim d'ells, tots tres nodes ofereixen el servei httpd al port 80. Quan des del client ens connectem a qualsevol dels nodes al port 80 (el que publiquen) la xarxa ingress dirigirà la petició al node i

container que consideri, no podem assegurar que el container que contesti sigui el del node al que el client s'ha dirigit.

- ❏ Una diferència important amb docker-compose és que amb docker-compose amb rèpliques no es poden publicar els ports al host amfitrió i s'utilitzen diversos ports dinàmics. Amb Swarm els ports interns del container es publiquen al node, amb independència de si el servei s'executa concretament en aquell node o no. I les peticions entrants són redistribuïdes al swarm, amb independència de que anaven dirigides a un node concret.

Amb la gestió de nodes podrem:

- afegir i eliminar nodes al swarm, indicant si realitzen el rol de manager o worker.
- modificar l'estat del node a *active*, *pause* o *drain*. Un node actiu funciona amb normalitat. En estat de pause no accepta noves tasques però continua executant les que té assignades. En estat de drain el node no accepta noves tasques i deixa d'executar les que té assignades (es mouen a un altre node).
- definir etiquetes o *labels* als nodes, tantes com es requereixin. Podem per exemple etiquetar els nodes com a locals o remots. Etiquetar nodes de asia o de europa. Nodes vermells i nodes blaus. Nodes de potència alta, potencia mitjana o potència baixa.

Amb la gestió de l'stack podem:

- engegar (desplegar) i aturar una app o stack.
- modificar el desplegament de l'stack fent un re-desplegament. Simplement cal fer un nou deploy passant-li una nova versió del fitxer docker-compose que defineix l'estat desitjat de l'stack.
- manualment amb l'ordre docker service també es pot modificar el número de rèpliques (o scale) d'un servei.
- observar els serveis que formen l'stack i els processos que s'executen en l'stack.

En veure el funcionament del desplegament de l'stack observem que:

- s'utilitza el nom de l'stack com a nom de la app.
- cada servei s'identifica amb un nom de tipus nomapp_nomservei.
- cada container s'identifica amb un nom tipus nomapp_nomservei_replica_id.
- cada xarxa s'identifica amb nomapp_nomxarxa.

En la configuració dels serveis a desplegar en un stack podem:

- indicar en el deployment el número de rèpliques.
- indicar *constraints* (restrinccions) i *placements*, a quins nodes es pot executar o no. Les restriccions als nodes es poden establir per:
  - node.role manager o role worker.
  - node.label.nometiqueta.
  - node.id, node.hostname, node.platform.os, node.platform.arc, engine.labels.
- indicar restriccions d'uilització de memòria, cpu, etc.

docker service

```
$ docker service create \
  --name redis_2 \
  --constraint node.platform.os==linux \
  --constraint node.labels.type==queue \
  redis:3.0.6

$ docker service create \
  --name web \
  --constraint node.labels.region==east \
  nginx:alpine

$ docker service create \
  --replicas 9 \
  --name redis_2 \
  --placement-pref spread=node.labels.datacenter \
  redis:3.0.6

$ docker service create \
  --replicas 9 \
  --name redis_2 \
  --placement-pref 'spread=node.labels.datacenter' \
  --placement-pref 'spread=node.labels.rack' \
  redis:3.0.6

$ docker service create --reserve-memory=4GB --name=too-big nginx:alpine

$ docker service create \
  --name nginx \
  --replicas 2 \
  --replicas-max-per-node 1 \
  --placement-pref 'spread=node.labels.datacenter' \
  nginx

$ docker service create \
  --replicas 3 \
  --network my-network \
  --name my-web \
  nginx
```