Servicios

- Fichas de Servicio
  - Cloud
  - Social
  - Picto
  - Blog
  - Video
  - Hosting Imágenes
  - Hosting Web
  - Mail
  - Fórum
- Servidores
- Virtualizador
- Borg (Servicio de backup)

Fichas de Servicio

Cloud

Software: Nextcloud
Web oficial: https://nextcloud.com/
Documentación oficial: https://docs.nextcloud.com/server/21/admin_manual/
Foro/Matrix de soporte: https://help.nextcloud.com/categories

Ubicaciones importantes: `/usr/share/nginx/nextcloud /usr/share/nginx/nextcloud-data`
Logs relevantes:

```
/usr/share/nginx/nextcloud/data/nextcloud.log
/usr/share/nginx/nextcloud-data/nextcloud.log
```

Comandos de gestión

Crear un usuario nuevo

```
sudo -u www-data php /usr/share/nginx/nextcloud/occ user:add --display-name="<Nombre Completo>" <user>
```

Actualizar

¡¡Siempre hay que leer la instrucciones en la documentación oficial y leer el changelog de la nueva versión!!
¡¡Es muy recomendable hacer una copia de seguridad antes de actualizar!!

Nextcloud tiene un actualizador en su interficie que se puede usar. Si no se puede, consultar la documentación.

Social

Software: Mastodon
Web oficial: https://joinmastodon.org/
Documentación oficial: https://docs.joinmastodon.org/
Foro/Matrix de soporte: https://discourse.joinmastodon.org/

Ubicaciones importantes: `/home/mastodon/live`
Logs relevantes:``

Comandos de gestión

Crear un usuario nuevo

```
sudo -u mastodon RAILS_ENV=production /home/mastodon/live/bin/tootctl accounts create <user> --email <mail>@anartist.org --confirmed
sudo -u mastodon RAILS_ENV=production /home/mastodon/live/bin/tootctl accounts modify <user> --approve
```

Dar permisos de administrador a una cuenta

```
sudo -u mastodon RAILS_ENV=production /home/mastodon/live/bin/tootctl accounts modify <user> --role admin
```

Actualizar

¡¡Siempre hay que leer la instrucciones en la documentación oficial y leer el changelog de la nueva versión!!
¡¡Es muy recomendable hacer una copia de seguridad antes de actualizar!!

```
su - mastodon
cd /home/mastodon/live
git fetch --tags
git checkout v3.1.2 ## adaptar a la versión que toque
# Aquí seguir las instrucciones del changelog
exit
sudo systemctl restart mastodon-sidekiq
sudo systemctl reload mastodon-web
sudo systemctl restart mastodon-streaming # Esto raramente es necesario, es mejor no hacerlo si no lo es.
```

Picto

Software: Pixelfed
Web oficial: https://pixelfed.org/
Documentación oficial: https://docs.pixelfed.org/
Foro/Matrix de soporte: pixelfed:matrix.org

Ubicaciones importantes: `/usr/share/webapps/pixelfed`
Logs relevantes:

Comandos de gestión

Crear un usuario nuevo

```
cd /usr/share/webapps/pixelfed
sudo php artisan user:create
```

Dar permisos de administrador a una cuenta

```
sudo php artisan user:admin username_here
```

Actualizar

Siempre hay que leer la instrucciones en la documentación oficial y leer el changelog de la nueva versión!!

```
cd /usr/share/webapps/pixelfed
sudo git pull origin dev
sudo composer install
sudo php artisan config:cache
sudo php artisan route:cache
sudo php artisan migrate --force
```

Diario de instalación

Tras la creación de un usuario propio diferente de root, lo primero que observo es que al usar sudo aparece el siguiente error:

```
sudo: unable to resolve host Pixelfed-Test: Name or service not known
```

Se soluciona editando el siguiente fichero y añadiendo el host:

```
sudo vim /etc/hosts
127.0.1.1       Pixelfed-Test
127.0.1.1       Pixelfed
```

El último lo añado para evitar problemas si cambiamos a Pixelfed el nombre del servidor.

Ahora actualizo todos los paquetes:

```
sudo apt update
sudo apt upgrade
```

Dependencias

Siguiendo la documentación de pixelfed, instalo las dependencias que indican:

```
sudo apt install nginx python3-certbot-nginx mariadb-server php-fpm git jpegoptim optipng pngquant ffmpeg
php-gd
```

Instalamos el Composer:

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') ===
'906a84df04cea2aa72f40b5f787e49f22d4c2f19492ac310e8cba5b96ac8b64115ac402c8cd292b8a03482574915
d1a8') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo
PHP_EOL;"
php composer-setup.php
php -r "unlink('composer-setup.php');"

sudo mv composer.phar /usr/local/bin/composer
```

Revisamos con el siguiente comando las extensiones de php que tenemos instaladas e instalamos las que nos faltan respecto a la documentación:

```
php-fpm7.4 -m
```

```
sudo apt install php-bcmath php-curl php-intl php-mbstring php-mysql php-redis php-xml php-zip redis
```

No he tenido que activara las extensiones de las bases de datos, como dice la documentaci'on de pixelfed.

Finalmente, cambio algunos parámetros del php.ini para que no de problemas al subir ficheros:

```
sudo vim /etc/php/7.4/fpm/php.ini


post_max_size = 10M
upload_max_filesize = 10M
max_execution_time = 600
```

Base de Datos

Procedo a crear la base de datos (obviamente con otra contraseña):

```
sudo mysql -u root -p
```

```
create database pixelfed;
grant all privileges on pixelfed.* to 'pixelfed'@'localhost' identified by 'XXXXXXXXX';
flush privileges;
exit;
```

Configuración del PHP-FPM pool

No creo un usuario dedicado. Usaremos www-data porque el usuario dedicado me ha dado problemas en el pasado.

Para configurar el php-fpm pool:

```
cd /etc/php/7.4/fpm/pool.d/
sudo cp www.conf pixelfed.conf
sudo vim pixelfed.conf
```

Sólo cambio el nombre del pool:

`[www-data]`
Corrijo esta ruta:
`listen = /run/php/php-fpm.sock`
Y decomento la siguiente linea:
`listen.mode = 0660`

Descargar Código

Ya podemos descargar el código fuente des de Git:

```
sudo mkdir /usr/share/webapps
cd /usr/share/webapps/
sudo git clone -b dev https://github.com/pixelfed/pixelfed.git pixelfed
```

Definir los permisos correctos

```
cd pixelfed
sudo chown -R www-data:www-data . # change user/group to http user and http group
sudo find . -type d -exec chmod 755 {} \; # set all directories to rwx by user/group
sudo find . -type f -exec chmod 644 {} \; # set all files to rw by user/group
```

Inicializar las dependencias PHP

`sudo composer install --no-ansi --no-interaction --optimize-autoloader`

Edición del fichero de configuración

Copiamos el fichero de ejemplo y lo rellenamos. En algunos casos he usado de referencia el que teníamos en la instalación previa.

```
sudo cp .env.example .env
sudo vim .env
```

Configurando servicios

Aquí ejecutamos una serie de tareas que sólo se requieren una vez siguiendo la documentación de forma fiel:

Una vez sólamente, se necesita generar una APP KEY secreta. Aunque la he generado, al final la he tenido que definir yo. Hay que tener cuidado porque la longitud de la llave tiene que ser 32 caracteres. De una configuración anterior arrastraba una más larga y me daba problemas.

`php artisan key:generate`

Enlazar la carpeta storage/ a la aplicación:

`php artisan storage:link`

Se tiene que migrar la base de datos:

`php artisan migrate --force`

Si se quiere activar el soporte para los datos de localización.

`php artisan import:cities`

Si activas la federación ActivityPub:

`php artisan instance:actor`

Si activas OAuth:

`php artisan passport:keys`

Las rutas se tienen que "cachear" cada vez que hay cambios en el código fuente o cuando cambiamos rutas:

```
php artisan route:cache
php artisan view:cache
```

Cada vez que editamos el fichero .env, tenemos que ejecutar este comando para que los cambios tengan efecto:

`php artisan config:cache`

Instalación de Horizon

```
php artisan horizon:install
php artisan horizon:publish
```

Creación de un servicio

Ahora vamos a crear un servicio para que se ejecute horizon. Creamos el siguiente fichero:

`/etc/systemd/system/pixelfed.service`

Con el siguiente contenido:

```
[Unit]
Description=Pixelfed task queueing via Laravel Horizon
After=network.target
Requires=mariadb
Requires=php7.4-fpm
Requires=redis
Requires=nginx
```

```
[Service]
Type=simple
ExecStart=/usr/bin/php7.4 /usr/share/webapps/pixelfed/artisan horizon
User=www-data
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Y lo activamos:

`sudo systemctl enable --now pixelfed`

Programación de tareas periódicas

Podemos programar un cron para que ejecute tareas regulares (como hacer backups):

`sudo -u www-data crontab -e`

Y incorporamos al cron:

`* * * * * /usr/bin/php7.4 /usr/share/webapps/pixelfed/artisan schedule:run >> /dev/null 2>&1`

Proxy

Usamos el proxy nginx. Copiamos el modelo y lo editamos;

```
sudo cp contrib/nginx.conf /etc/nginx/modules-available/pixelfed.conf
sudo vim /etc/nginx/sites-available/pixelfed.conf
```

Con las siguientes líneas modificadas;

```
server_name picto.anartist.org;                 # change this to your fqdn
root /usr/share/webapps/pixelfed/public;                    # path to repo/public

fastcgi_pass unix:/run/php/php7.4-fpm.sock; # make sure this is correct

#    listen 443 ssl http2;
#    listen [::]:443 ssl http2;
#    ssl_certificate /etc/nginx/ssl/server.crt;       # generate your own
#    ssl_certificate_key /etc/nginx/ssl/server.key;   # or use letsencrypt
```

```
#    ssl_protocols TLSv1.2;
#    ssl_ciphers EECDH+AESGCM:EECDH+CHACHA20:EECDH+AES;
#    ssl_prefer_server_ciphers on;
```

Obtenemos un certificado de Let's Encrypt:

`sudo certbot --nginx -d picto.anartist.org`

Tras correr el certbot he tenido que revisar el fichero de nginx y comentar el server http, porque me lo habia duplicado.

Finalmente activamos la configuración nginx y reiniciamos el servicio

```
sudo ln -s /etc/nginx/sites-available/pixelfed.conf /etc/nginx/sites-enabled/
sudo systemctl restart nginx.service
```

Blog

Software: Writefreely
Web oficial: https://writefreely.org/
Documentación oficial: https://writefreely.org/docs
Foro/Matrix de soporte: https://discuss.write.as/c/writefreely

Ubicaciones importantes: `/var/www/blog.anartist.org/`
Logs relevantes: `/var/log/syslog`

Comandos de gestión

Crear un usuario nuevo

```
cd /var/www/blog.anartist.org/
sudo ./writefreely --create-user user:password
```

Crear una cuenta de administrador

```
cd /var/www/blog.anartist.org/
sudo ./writefreely --create-admin user:password
```

Actualizar

¡¡Siempre hay que leer la instrucciones en la documentación oficial y leer el changelog de la nueva versión!!
¡¡Es muy recomendable hacer una copia de seguridad antes de actualizar!!

```
# Detener writefreely
# Descargar ficheros de release y sustituirlos por los existentes. Luego ejecutar:
# Arrancar servicio writefreely de nuevo
# Ejecutar:
./writefreely db migrate
```

Diario de Instalación

He seguido las instrucciones de aquí

```
sudo apt update
sudo apt upgrade

sudo apt install nginx certbot python3-certbot-nginx mysql-server

sudo mysql
```

Primero creamos la base de datos:

```
CREATE DATABASE writefreely CHARACTER SET latin1 COLLATE latin1_swedish_ci;
CREATE USER 'writefreely'@'localhost' IDENTIFIED BY 'XXXXXXXXX';
GRANT ALL PRIVILEGES ON writefreely.* to writefreely@'localhost';
FLUSH PRIVILEGES;
exit
```

Descargamos writefreely y configuramos. Es un proceso que te va guiando. En nuestro caso, he puesto **un máximo de 5 blogs por usuario**.

```
wget https://github.com/writeas/writefreely/releases/download/v0.12.0/writefreely_0.12.0_linux_amd64.tar.gz

tar -xf writefreely_0.12.0_linux_amd64.tar.gz
cd writefreely
./writefreely config start
./writefreely keys generate
```

Durante la configuración, le he indicado la opción **Producción detrás de un servidor proxy**, así que configuro *nginx*:

```
cd /etc/nginx/sites-available/
sudo vi writefreely
```

Contenido del fichero de configuración *nginx*:

```
server {

    server_name blog.anartist.org;

    gzip on;
    gzip_types
```

```
  application/javascript
  application/x-javascript
  application/json
  application/rss+xml
  application/xml
  image/svg+xml
  image/x-icon
  application/vnd.ms-fontobject
  application/font-sfnt
  text/css
  text/plain;
gzip_min_length 256;
gzip_comp_level 5;
gzip_http_version 1.1;
gzip_vary on;

location ~ ^/.well-known/(webfinger|nodeinfo|host-meta) {
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $remote_addr;
    proxy_pass http://127.0.0.1:8080;
    proxy_redirect off;
}

location ~ ^/(css|img|js|fonts)/ {
    root /var/www/blog.anartist.org/static;
    # Optionally cache these files in the browser:
    # expires 12M;
}

location / {
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $remote_addr;
    proxy_pass http://127.0.0.1:8080;
    proxy_redirect off;
}

listen [::]:443 ssl ipv6only=on; # managed by Certbot
listen 443 ssl; # managed by Certbot
```

```
    ssl_certificate /etc/letsencrypt/live/blog.anartist.org/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/blog.anartist.org/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}
server {
    if ($host = blog.anartist.org) {
        return 301 https://$host$request_uri;
    } # managed by Certbot



    listen 80;
    listen [::]:80;

    server_name blog.anartist.org;
    return 404; # managed by Certbot



}
```

```
sudo ln -s /etc/nginx/sites-available/writefreely /etc/nginx/sites-enabled/

sudo systemctl restart nginx.service
sudo certbot --nginx

sudo mv /home/marcel/writefreely /var/www/
cd /var/www/

sudo mv writefreely/ blog.anartist.org/
sudo chown -R root:root blog.anartist.org/
```

Finalmente, creamos el servicio *systemd*:

```
sudo vi /etc/systemd/system/writefreely.service
```

Contenido del fichero de configuració *.service:

```
[Unit]
Description=WriteFreely Instance
```

```
#After=syslog.target network.target
# If MySQL is running on the same machine, uncomment the following
# line to use it, instead.
After=syslog.target network.target mysql.service

[Service]
Type=simple
StandardOutput=syslog
StandardError=syslog
WorkingDirectory=/var/www/blog.anartist.org/
ExecStart=/var/www/blog.anartist.org/writefreely
Restart=always

[Install]
WantedBy=multi-user.target
```

Y arrancamos el servicio. ¡Ya está disponible!

```
sudo systemctl restart writefreely.service
```

Finalmente, creo un usuario administrador y el primer usuario:

```
cd blog.anartist.org/
sudo ./writefreely --create-admin anartist:XXXXXXXXX
sudo ./writefreely --create-user marcel:XXXXXXXX
```

Video

Software: Peertube **Web oficial**: https://joinpeertube.org/

Documentación oficial: https://docs.joinpeertube.org/

Foro/Matrix de soporte: https://framacolibri.org/c/peertube/38

Ubicaciones importantes: `/var/www/peertube`
Logs relevantes:

Comandos de gestión

Crear un usuario nuevo
No hay descripción en la documentación de cómo gestionar usuarios a través de CLI.
Se puede hacer mediante la interfaz web, en esta página:

https://video.anartist.org/admin/users/create

Actualizar

¡¡Siempre hay que leer la instrucciones en la documentación oficial y leer el changelog de la nueva versión!!
¡¡Es muy recomendable hacer una copia de seguridad antes de actualizar!!

```
cd /var/www/peertube/peertube-latest/scripts && sudo -H -u peertube ./upgrade.sh
```

Reiniciar Peertube y comprobar logs:

```
sudo systemctl restart peertube && sudo journalctl -fu peertube
```

Más información **en esta guía**

Diario de Instalación

Servidor XMPP para chat en emisión Live de Peertube

Instalar Prosody, un servidor XMPP:

```
sudo apt install prosody
```

Editar el archivo de configuración de Prosody, de tal manera que concuerde con la instancia

```
sudo nano /etc/prosody/conf.avail/video.anartist.org.cfg.lua
```

Copiar el contenido de aquí: virtualhost.cfg.lua ; y **cambiar** los **nombres** predeterminados por aquellos de la instancia.
Para el apartado *Component*, `room.xxx.xxx.xx`

Guardar el documento y salir

Crear un enlace estático entre dos archivos

```
sudo su -


cd /etc/prosody/conf.d


ln -s ../conf.avail/video.anartist.org.cfg.lua
```

Importar el certificado TLS para Prosody

```
prosodyctl --root cert import /etc/letsencrypt/live/
```

Reiniciar Prosody y, después, **comprobar su estado**

```
systemctl restart prosody
systemctl status prosody
```

Añadir el código posterior al final del archivo de configuración de Nginx para Peertube, antes de su último `}`

```
nano /etc/nginx/sites-enabled/peertube
```

Este es el código:

```
location /http-bind {
    proxy_pass  http://localhost:5280/http-bind;
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-For $remote_addr;
    proxy_buffering off;
```

```
  tcp_nodelay on;
 }

 location /xmpp-websocket {
  proxy_pass http://localhost:5280/xmpp-websocket;
  proxy_http_version 1.1;
  proxy_set_header Connection "Upgrade";
  proxy_set_header Upgrade $http_upgrade;

  proxy_set_header Host $host;
  proxy_set_header X-Forwarded-For $remote_addr;
  proxy_read_timeout 900s;
 }
```

Comprobar el estado de Nginx (para buscar errores) y **reiniciar Nginx**

```
nginx -t
systemctl reload nginx
```

Luego instalar el plugin `livechat` ***desde la interfaz web de Peertube (>3.0.1)*** *y seguir los pasos especificados* ***aquí***

Errores conocidos

YarnPkg invalid signature

A veces al ejecutar `sudo apt-get update` se produce este error que no deja proseguir con la actualización de los repositorios:

```
[...]
Err:4 https://dl.yarnpkg.com/debian stable InRelease
  The following signatures were invalid: EXPKEYSIG 23E7166788B63E1E Yarn Packaging <yarn@dan.cx>
Hit:8 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease
Fetched 17.1 kB in 1s (22.4 kB/s)
Reading package lists... Done
W: An error occurred during the signature verification. The repository is not updated and the previous index files will be used. GPG error: https://dl.yarnpkg.com/debian stable InRelease: The following signatures were invalid: EXPKEYSIG 23E7166788B63E1E Yarn Packaging <yarn@dan.cx>
W: Failed to fetch https://dl.yarnpkg.com/debian/dists/stable/InRelease  The following signatures were invalid: EXPKEYSIG 23E7166788B63E1E Yarn Packaging <yarn@dan.cx>
```

```
W: Some index files failed to download. They have been ignored, or old ones used instead.
```

El problema surge porque las claves del repositorio `https://dl.yarnpkg.com/debian` no están actualizadas y verificadas. El sistema operativo no deja proseguir.

Es un problema del repositorio de Yarn conocido y he aquí una solución:
`curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -`

Después de esto, el problema debería estar resuelto.

Hosting Imágenes

Software: lutim
Web oficial: https://framagit.org/fiat-tux/hat-softwares/lutim
Documentación oficial: https://framagit.org/fiat-tux/hat-softwares/lutim/-/wikis/home
Foro/Matrix de soporte:

Ubicaciones importantes: `/var/www/lutim/`
Logs relevantes: `/var/www/lutim/log/production.log`

Comandos de gestión

Acceder al contenedor Lutim está dentro de un contenedor LXD en el servidor de blog de anartist (blog.anartist.org). Para acceder tenemos que usar:

```
sudo lxc list #Esto es para listar información de los contenedores, no es necesario
sudo lxc lutim exec
```

Y ya estaremos dentro del servidor.

Actualizar

¡¡Siempre hay que leer la instrucciones en la documentación oficial y leer el changelog de la nueva versión!!
¡¡Es muy recomendable hacer una copia de seguridad antes de actualizar!!

```
cd /var/www/lutim/
git pull
carton install
vimdiff lutim.conf lutim.conf.template
carton exec hypnotoad script/lutim
```

Diario de instalación

Primero he instalado lxd via snap:

```
sudo snap install core
sudo snap install lxd
```

He configurado lxd con todas las opciones por defecto y he creado una imagen con ubuntu 20.04:

```
sudo lxd init
sudo lxc launch images:ubuntu/focal lutim
```

Listamos las imágenes y entramos dentro del contenedor

```
sudo lxc list
sudo lxc exec lutim bash
```

Siguiendo las instrucciones, actualizo el servidor y voy instalando lo necesario:

```
apt update
apt upgrade
apt install cpan
apt install build-essential
cpan Carton
cpan inc::Module::Install::DSL
apt install libssl-dev shared-mime-info libpq-dev zlib1g-dev git
git clone https://framagit.org/fiat-tux/hat-softwares/lutim.git
cd lutim/
carton install --deployment --without=test --without=postgresql --without=ldap --without=htpasswd --
without=cache --without=memcached
cp lutim.conf.template lutim.conf
vi lutim.conf
carton exec hypnotoad script/lutim
cp utilities/lutim.service /etc/systemd/system/
vi /etc/systemd/system/lutim.service
sudo mkdir /var/www
mv lutim/ /var/www/
systemctl restart lutim
```

Configuramos nginx:

```
apt install nginx
rm /etc/nginx/sites-enabled/default
vi /etc/nginx/sites-available/lutim
```

Fichero de configuracíón de nginx dentro del contenedor:

```
server {
    listen 80;
    # No need to have a `root` parameter.

    # This is important for user's privacy !
    access_log off;
    error_log /var/log/nginx/lutim.error.log;

    # This is important ! Make it OK with your Lutim configuration
    client_max_body_size 40M;

    location ~* ^/(img|css|font|js)/ {
        try_files $uri @lutim;
        add_header Expires "Thu, 31 Dec 2037 23:55:55 GMT";
        add_header Cache-Control "public, max-age=315360000";

        # HTTPS only header, improves security
        #add_header Strict-Transport-Security "max-age=15768000";
    }

    location / {
        try_files $uri @lutim;

        # HTTPS only header, improves security
        #add_header Strict-Transport-Security "max-age=15768000";
    }

    location @lutim {
        # Adapt this to your configuration
        proxy_pass http://127.0.0.1:8080;

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

        # If you want to log the remote port of the image senders, you'll need that
        proxy_set_header X-Remote-Port $remote_port;
```

```
        proxy_set_header X-Forwarded-Proto $scheme;

        # We expect the downsteam servers to redirect to the right hostname, so don't do any rewrites here.
        proxy_redirect off;
    }
}
```

Continuamos con la configuración de nginx:

```
ln -s /etc/nginx/sites-available/lutim /etc/nginx/sites-enabled/
systemctl restart nginx
```

En el servidor host, también he configurado el nginx con el siguiente fichero:

```
server {

    listen 80;
    listen [::]:80;

    server_name    lutim.anartist.org;

    location / {
        proxy_pass          http://10.253.31.53:80;
        proxy_set_header    Host    $host;
        proxy_set_header    X-Forwarded-For $remote_addr;
        proxy_set_header    X-Real-IP   $remote_addr;
        proxy_pass_header   server;
    }
```

Y configuramos el certificado de Let's Encrypt:

```
sudo systemctl restart nginx
sudo certbot --nginx
```

Hosting Web

Software: HestiaCP

Web oficial: hestiacp.com

Documentación oficial: docs.hestiacp.com **Foro/Matrix de soporte**:

Ubicaciones importantes:
Logs relevantes:

Comandos de gestión

Actualizar

¡¡Siempre hay que leer la instrucciones en la documentación oficial y leer el changelog de la nueva versión!!
¡¡Es muy recomendable hacer una copia de seguridad antes de actualizar!!

```

```

Diario de Instalación

La instalación es muy sencilla.

```
apt update && apt upgrade
wget https://raw.githubusercontent.com/hestiacp/hestiacp/release/install/hst-install.sh
bash hst-install.sh
```

Durante la instalación nos irá pidiendo alguna información y nos creará una cuenta de administrador.

Podemos hacer una instalación personalida introduciendo algunos parámetros al script, en el servidor de anartist hemos utilizado la siguiente

```
bash hst-install.sh -a yes -n yes -w yes -o yes -v yes -k no -g no -t no -c no -x yes -z no -q yes -b yes -i yes -m yes
-l es -r 1984 \ -s XXXXXX.org -e XXXX@XXXXXX.com -p XXXXXX`
```

Con esta configuración del script evitamos instalar el servidor de DNS ya que utilizamos Don Dominio; el servidor de correo, ya tenemos uno para eso; el antivirus, solo se utiliza para revisar el correo; instalamos la opción de multiPHP sobre NGINX como proxy y Apache con PHP-FPM y cambiamos el puerto del panel al 1984.

Y nos conectamos a https://135.181.14.114:1984 y nos registramos con las credenciales de administrador generadas durante la instalación.

Las difrentes opciones del script son las siguientes

```
-a, --apache          Install Apache        [yes|no]  default: yes
-n, --nginx           Install Nginx         [yes|no]  default: yes
-w, --phpfpm          Install PHP-FPM       [yes|no]  default: yes
-o, --multiphp        Install Multi-PHP     [yes|no]  default: no
-v, --vsftpd          Install Vsftpd        [yes|no]  default: yes
-j, --proftpd         Install ProFTPD       [yes|no]  default: no
-k, --named           Install Bind          [yes|no]  default: yes
-m, --mysql           Install MariaDB       [yes|no]  default: yes
-g, --postgresql      Install PostgreSQL    [yes|no]  default: no
-x, --exim            Install Exim          [yes|no]  default: yes
-z, --dovecot         Install Dovecot       [yes|no]  default: yes
-c, --clamav          Install ClamAV        [yes|no]  default: yes
-t, --spamassassin    Install SpamAssassin  [yes|no]  default: yes
-i, --iptables        Install Iptables      [yes|no]  default: yes
-b, --fail2ban        Install Fail2ban      [yes|no]  default: yes
-q, --quota           Filesystem Quota      [yes|no]  default: no
-d, --api             Activate API          [yes|no]  default: yes
-r, --port            Change Backend Port             default: 8083
-l, --lang            Default language                default: en
-y, --interactive     Interactive install   [yes|no]  default: yes
-s, --hostname        Set hostname
-e, --email           Set admin email
-p, --password        Set admin password
-f, --force           Force installation
-h, --help            Print this help
```

Web Anartist

Los archivos del CMS Grav de la web de anartist están alojados en `/home/anartist/web/anartist.org/public_html`

Con esta configuración de servidor tendremos que añadir contenido a .htaccess

Mail

Software: iRedMail
Web oficial: https://iredmail.org/
Documentación oficial: https://docs.iredmail.org/index.html
Foro/Matrix de soporte: https://forum.iredmail.org/

Ubicaciones importantes: `/root/iRedMail-1.6.2/`
Logs relevantes: `/var/log/syslog`

Comandos de gestión

Crear un usuario nuevo

```
sudo su
cd /root/iRedMail-1.6.2/tools/
bash ./create_mail_user_OpenLDAP.sh anartist.org <user>
```

Copias de seguridad
Ubicación: `/var/vmail/backup/backups`

Comandos cron para copia de seguridad diaria

```
# m h  dom mon dow   command
0   3   *   *   *   /bin/bash /var/vmail/backup/backup_openldap.sh

# iRedMail: Backup MySQL databases on 03:30 AM
30   3   *   *   *   /bin/bash /var/vmail/backup/backup_mysql.sh

# iRedAPD: Clean up expired tracking records hourly.
1   *   *   *   *   python2 /opt/iredapd/tools/cleanup_db.py >/dev/null

# iRedAPD: Convert SPF DNS record of specified domain names to IP
#          addresses/networks hourly.
2   *   *   *   *   python2 /opt/iredapd/tools/spf_to_greylist_whitelists.py >/dev/null

# iRedMail: Cleanup Amavisd database
```

```
1   2   *   *   *   python2 /opt/www/iredadmin/tools/cleanup_amavisd_db.py >/dev/null

# iRedAdmin: Clean up sql database.
1   *   *   *   *   python2 /opt/www/iredadmin/tools/cleanup_db.py >/dev/null 2>&1

# iRedAdmin: Delete mailboxes on file system which belong to removed accounts.
1   *   *   *   *   python2 /opt/www/iredadmin/tools/delete_mailboxes.py
# iRedMail: Cleanup Roundcube SQL database
2   2   *   *   *   /usr/bin/php /opt/www/roundcubemail/bin/cleandb.sh >/dev/null

# iRedMail: Cleanup Roundcube temporary files under 'temp/' directory
2   2   *   *   *   /usr/bin/php /opt/www/roundcubemail/bin/gc.sh >/dev/null
# iRedMail: Backup SOGo data databases on 04:01AM
1   4   *   *   *   /bin/bash /var/vmail/backup/backup_sogo.sh

# iRedMail: Backup completo con mantenimiento de tres días
30   4   *   *   *   /bin/bash /var/vmail/backup/backup_correos.sh
```

Actualizar

¡¡Siempre hay que leer la instrucciones en la documentación oficial y leer el changelog de la nueva versión!!
¡¡Es muy recomendable hacer una copia de seguridad antes de actualizar!!

Hay que actualizar varios elementos. Consulta la documentación.

Fórum

Software: Discourse
Web oficial: http://discourse.org/
Documentación oficial: https://github.com/discourse/discourse/blob/master/docs/INSTALL-cloud.md
Foro/Matrix de soporte: https://meta.discourse.org/

Ubicaciones importantes: `/var/discourse`
Logs relevantes:

Comandos de gestión

La mayor parte de la gestión de este software se hace a través de la UI web.

Acceder a la configuración

```
ssh -o "ServerAliveInterval 60" -p [[Puerto]] [[Usuario]]@[[IP]]
```

Crear un usuario nuevo

En este caso los usuarios se registran solos, aunque se puede crear un usuario así:

```
cd /var/discourse
sudo ./launcher enter app
rake admin:create RAILS_DB=discourse
```

Copias de seguridad

Ubicación: `/var/discourse/shared/standalone/backups/default/`

Se configuran a través del panel de administrador (/admin/backups).

Actualizar

¡¡Siempre hay que leer la instrucciones en la documentación oficial y leer el changelog de la nueva versión!!
¡¡Es muy recomendable hacer una copia de seguridad antes de actualizar!!

En este caso se actualiza a través de la web, en el panel de administrador (/admin/upgrade).

```
cd /var/discourse
./launcher rebuild app
```

Servidores

Hetzner

Este es el servidor principal de anartist

Aspectos técnicos

Lorem ipsum

Time4VPS

Este es el servidor de backup de anartist

Aspectos técnicos

Lorem ipsum

Virtualizador

Usamos un virtualizador cuyo nombre no recuerdo jejejeje @marcel me podrías ayudar un poco con esto XD o igual ya está documentado en otro lau

Borg (Servicio de backup)

Software: Borg
Web oficial: https://www.borgbackup.org
Documentación oficial: https://borgbackup.readthedocs.io/en/stable/
Conversación inicial en el foro de anartist: https://forum.anartist.org/t/backups-incrementales-con-borg/793

Ubicaciones importantes: `Lorem/ipsum`

Comandos de gestión

Listado de comandos creados por @marcel: https://cloud.anartist.org/s/fKiXt4ytMtp3p4E

```
#!/bin/bash

sshfs anartist@XX.XXX.XXX.XX:/home/anartist/borgbackup /root/borgbackup/

export BORG_REPO=/root/borgbackup/cloud
export BORG_PASSPHRASE=''

info () { printf "\n%s %s\n\n" "$( date )" "$*" >&2; }
trap 'echo $( date ) Backup interrupted >&2; exit 2' INT TERM

info "Entering Maintenance Mode"
sudo -u www-data php /usr/share/nginx/nextcloud/occ maintenance:mode --on

info "Starting DB backup"

dbdate=`date +"%Y%m%d-%H%M%S"`
mysqldump --opt --password=XXXXXXXXXX --user=root nextcloud | gzip > /backup/base/nextcloud-sqlbkp_${dbdate}.sql.gz

info "Starting Remote backup"

/usr/bin/borg create --verbose --stats ::backup_$(date +%Y%m%d) /usr/share/nginx/nextcloud-data /usr/share/nginx/nextcloud /backup/base/nextcloud-sqlbkp_${dbdate}.sql.gz /etc/nginx/conf.d/nextcloud.conf
```

```
backup_exit=$?

/usr/bin/borg prune --list --show-rc --keep-daily 7 --keep-weekly 4 --keep-monthly 6

prune_exit=$?

# use highest exit code as global exit code
global_exit=$(( backup_exit > prune_exit ? backup_exit : prune_exit))

if [ ${global_exit} -eq 0 ]; then
    info "Backup and Prune finished successfully"
elif [ ${global_exit} -eq 1 ]; then
    info "Backup and/or Prune finished with warnings"
else
    info "Backup and/or Prune finished with errors"
fi

fusermount -u /root/borgbackup/

info "Eliminating old db"
ls /backup/base/* -trd | head -n -2 | xargs --no-run-if-empty rm

info "Deactivating Maintenance Mode"
sudo -u www-data php /usr/share/nginx/nextcloud/occ maintenance:mode --off

info "Backup Finished"

exit ${global_exit}
```