Servicios Con Docker Compose

Introducción

En esta sección agregare una lista de docker compose de aplicaciones de interés común que previamente haya probado.
Todos los stacks tendrán su archivo docker-compose.yaml y su archivo .env los cuales deben estar dentro de una carpeta con el nombre del stack.

Además, adjunto una lista de comandos de Docker para hacer todas las actividades cotidianas del mantenimiento de servicios de Docker, en caso de que sea Tapatío puedes instalar Portainer para hacer toda esta gestión mediante una Interfaz Web.

Recomendaciones

Para fines de portabilidad te recomiendo crear una carpeta donde almacenaras toda la data de los servicios de docker, de modo que si necesitas exportar todos tus servicios solo necesitaras copiar esta carpeta al nuevo servidor, ejecutar los contenedores de nuevo y estos preservaran el estado anterior.

Carpetas que recomiendo crear

Carpeta relacionada a toda la data de las apps de docker y su estado
Carpeta: /dockerData/appdata
Nombre en los enviroments : PATH_TO_APPDATA

Carpeta relacionada a toda la data de usuarios de los distintos servicios de Docker
Carpeta: /dockerData/data
Nombre en los enviroments: PATH_TO_DATA

Carpeta dedicada a almacenar todos los stacks de los distintos servicios de Docker
Carpeta: /dockerData/stack

Detalles de la infrestuctura

 La documentación esta planeada para el uso de Nginx Proxy Manager para exponer los servicios por lo que es importante que primero hagas la instancia de Nginx Proxy Manager o su red correspondiente para que todos los demás stacks se monten sobre esta red como externos, Todos los servicios en estos docker compose que sean compartidos mediante HTTP tendrán la configuración de red externo con excepción de Nginx Proxy Manager el cual lo tendrá como red interna.

Comandos Esenciales de Docker

Listar contenedores

```
# Contenedores activos
docker ps

# Todos los contenedores (activos e inactivos)
docker ps -a

# Contenedores con formato personalizado
docker ps --format "table {{.Names}}\t{{.Status}}\t{{.Ports}}"
```

Gestionar stacks/compose

```
# Levantar stack desde archivo compose
docker compose -f /ruta/al/docker-compose.yml up -d


# Detener stack sin eliminar contenedores
docker compose -f /ruta/al/docker-compose.yml stop
```

```
# Detener y eliminar stack
docker compose -f /ruta/al/docker-compose.yml down

# Reiniciar stack
docker compose -f /ruta/al/docker-compose.yml restart

# Ver logs del stack
docker compose -f /ruta/al/docker-compose.yml logs -f

# Tambien puedes posicionarte sobre la carpeta para ahorrarte
# escribir -f y la ruta al stack. ejemplo a continuacion

# Ingresa a la ruta del stack
cd /ruta/a/stacks/filebrowser

# Verificar que el archivo compose es correcto y los enviroments se cargan
docker compose config

# Actualizar
docker compose pull
docker compose up -d

# Forzar recreacion de los contenedores del stack
docker compose up -d --force-recreate
```

Gestionar contenedores individuales

```
# Detener contenedor
docker stop nombre_contenedor

# Iniciar contenedor
docker start nombre_contenedor

# Reiniciar contenedor
docker restart nombre_contenedor

# Eliminar contenedor
docker rm nombre_contenedor

# Ver logs en tiempo real
docker logs -f nombre_contenedor
```

```
# Ejecutar comando dentro del contenedor
docker exec -it nombre_contenedor bash
```

Gestionar imágenes

```
# Listar imágenes
docker images

# Listar imágenes no utilizadas (dangling)
docker images -f "dangling=true"

# Eliminar imágenes no utilizadas
docker image prune

# Eliminar todas las imágenes no utilizadas (incluyendo las no referenciadas)
docker image prune -a

# Actualizar imágenes específicas
docker pull nombre_imagen:tag

# Ver el tamaño de las imágenes
docker system df
```

Gestionar redes

```
# Listar redes
docker network ls

# Inspeccionar red específica
docker network inspect nombre_red

# Listar redes no utilizadas
docker network ls --filter dangling=true

# Eliminar redes no utilizadas
docker network prune

# Crear red personalizada
docker network create mi_red
```

Gestionar volúmenes

```
# Listar volúmenes
docker volume ls
```

```
# Inspeccionar volumen
docker volume inspect nombre_volumen

# Listar volúmenes no utilizados
docker volume ls -f dangling=true

# Eliminar volúmenes no utilizados
docker volume prune

# Crear volumen
docker volume create nombre_volumen
```

Comandos de monitoreo

```
# Estadísticas de contenedores en tiempo real
docker stats

# Uso de disco del sistema Docker
docker system df

# Eventos de Docker en tiempo real
docker events

# Información del sistema
docker info
```

Scripts

Para la creación de scripts te recomendamos revisar la sección de Crons y Bash Files

Limpieza completa de Docker

Este script es para limpiar Docker por completo

cleanup-docker.sh

```
#!/bin/bash
echo "=== Limpieza de Docker ==="
echo "1. Deteniendo contenedores..."
docker stop $(docker ps -q)
echo "2. Eliminando contenedores..."
docker rm $(docker ps -aq)
echo "3. Eliminando imágenes no utilizadas..."
```

```
docker image prune -af
echo "4. Eliminando redes no utilizadas..."
docker network prune -f
echo "5. Eliminando volúmenes no utilizados..."
docker volume prune -f
echo "6. Limpieza del sistema..."
docker system prune -af
echo "□ Limpieza completada"
```

Fin

Nginx Proxy Manager

Que hace este servicio

Este servicio es un proxy Inverso para redirigir distintos puertos internos.

docker-compose.yaml

```
services:
    nginx-proxy:
        image: 'jc21/nginx-proxy-manager:latest'
        container_name: nginx-proxy-manager
        ports:
          - '80:80' # Public HTTP Port
          - '443:443' # Public HTTPS Port
          - '81:81' # Admin Web
        networks:
          - sky_net
        environment:
          - NGINX_MAX_BODY_SIZE=200G
          - NGINX_PROXY_READ_TIMEOUT=72000
          - NGINX_PROXY_BUFFERING=off
        volumes:
          - ${PATH_TO_APPDATA}/nginxproxymanager/data:/data
          - ${PATH_TO_APPDATA}/nginxproxymanager/letsencrypt:/etc/letsencrypt
        restart: unless-stopped

networks:
  sky_net:
    driver: bridge
    name: sky_net
```

.env

```
PATH_TO_APPDATA=/dockerData/appdata
```

Menciones

Apps Cloud

Si tienes una app cloud con la cual se transmitirán datos mediante webDav o cualquier protocolo http requerirás agregar la siguiente configuración a ese proxy host

```
client_max_body_size 0;
proxy_buffering off;
proxy_request_buffering off;
proxy_send_timeout 72000s;
proxy_read_timeout 72000s;
proxy_connect_timeout 300s;
```

FileBrowser

Que hace este servicio

Este servicio para administrar, descargar y subir archivos a tu servidor mediante la web.

docker-compose.ymal

```
services:
    filebrowser:
        image: filebrowser/filebrowser:latest
        container_name: filebrowser
        #ports:
        #  - 8009:80
        networks:
            - sky_net
        environment:
          - FB_BASEURL=/filebrowser
          - PUID=${APPUSER_PUID}
          - PGID=${APPUSER_PGID}
        volumes:
          - ${PATH_TO_DOCKERDATA}/filebrowser/database:/database
          - ${PATH_TO_DOCKERDATA}/filebrowser/config:/config
          - ${PATH_TO_DATA}/filebrowser/users:/users
          - ${PATH_TO_DATA}:/filebrowser/srv:/srv
          - ${PATH_TO_OTHER_FOLDERS1}:/srv/folder2
          - ${PATH_TO_OTHER_FOLDERS2}:/srv/folder3
        restart: always

networks:
  sky_net:
    driver: bridge
    external: true
    name: sky_net
```

.env

```
PATH_TO_APPDATA=/dockerData/appdata
PATH_TO_DATA=/dockerData/data
PATH_TO_OTHER_FOLDERS1=/other/location1
PATH_TO_OTHER_FOLDERS2=/other/location2
APPUSER_PGID=1000
APPUSER_PUID=1000
```

Menciones

No es necesario tener PATH_TO_OTHER_USERSDATA si no tendrás más usuarios que puedan acceder a almacenar o editar archivos.

Bookstack

Que hace este servicio

Este servicio es el que estas viendo, una wiki basada en librerias, libros y paginas

docker-compose.yaml

```
services:
    bookstack:
        image: lscr.io/linuxserver/bookstack:latest
        container_name: bookstack
        #ports:
        #    - 6875:80
        networks:
            - sky_net
        depends_on:
            - bookstack_db
        environment:
            - PUID=1000
            - PGID=1000
            - TZ=America/Mexico_City
            - APP_URL=https://wiki.tinyfox.ddns.me
            - APP_KEY=${APP_KEY}
            - DB_HOST=bookstack_db
            - DB_PORT=3306
            - DB_USERNAME=${MYSQL_USER}
            - DB_PASSWORD=${MYSQL_PASSWORD}
            - DB_DATABASE=${MYSQL_DATABASE}
        volumes:
            - ${PATH_TO_APPDATA}/bookstack/config:/config
        restart: unless-stopped

    bookstack_db:
        image: lscr.io/linuxserver/mariadb
        container_name: bookstack_db
        networks:
```

```
            - sky_net
        environment:
            - PUID=1000
            - PGID=1000
            - MYSQL_ROOT_PASSWORD=${MYSQL_ROOT_PASSWORD}
            - MYSQL_USER=${MYSQL_USER}
            - MYSQL_PASSWORD=${MYSQL_PASSWORD}
            - MYSQL_DATABASE=${MYSQL_DATABASE}
        volumes:
            - ${PATH_TO_APPDATA}/bookstack/db:/config

networks:
  sky_net:
    driver: bridge
    external: true
    name: sky_net
```

.env

```
PATH_TO_APPDATA=/dockerData/appdata
MYSQL_USER=
MYSQL_PASSWORD=
MYSQL_DATABASE=
MYSQL_ROOT_PASSWORD=
API_KEY=
```

La API_KEY la puedes generar con el siguiente comando

```
docker run -it --rm --entrypoint /bin/bash lscr.io/linuxserver/bookstack:latest appkey
```

Brave

Que hace este servicio

Este servicio es un navegador web dentro de tu navegador, es util cuando no puedes instalar una vpn en el dispositivo y quieres no navegar desde tu red actual, si no, navegar desde la red del servidor.

docker-ccompose.yaml

```
services:
    brave:
        image: lscr.io/linuxserver/brave:latest
        container_name: brave
        #ports:
        #    - 3000:3000
        #    - 8008:3001 #Puerto de interfaz web
        networks:
            - sky_net
        environment:
            - PUID=${APPUSER_PUID}
            - PGID=${APPUSER_PGID}
            - TZ=America/Mexico_City
        volumes:
            - ${PATH_TO_APPDATA}/brave/config:/config
        shm_size: "2gb"
```

.env

```
PATH_TO_APPDATA=/dockerData/appdata
APPUSER_PGID=1000
APPUSER_PUID=1000
```

Jellyfin

Que hace este servicio

Este es un servicio para transmitir videos, musica y libros (este ultimo no muy recomendado), incorpora una base de datos para identificar peliculas y series, ademas permite ingresar mas bases de datos y otras utilidades mediante plugins, es completamente autoalojada.

docker-compose.yaml

```
services:
    jellyfin:
        image: 'jellyfin/jellyfin:latest'
        container_name: 'jellyfin'
        #ports:
        #    - '8096:8096/tcp'
        networks:
            - sky_net
        volumes:
            - '${PATH_TO_APPDATA}/jellyfin/config:/config'
            - '${PATH_TO_APPDATA}/jellyfin/cache:/cache'
            - '${PATH_TO_NAS}/videos:/data'
        restart: unless-stopped

networks:
  sky_net:
    driver: bridge
    external: true
    name: sky_net
```

docker-compose.yaml especial para Orange Pi 5 y cualquiera de sus variantes, requisitos.

```
services:
    jellyfin:
        image: 'jellyfin/jellyfin:latest'
        container_name: 'jellyfin'
        ports:
```

```
            - '8002:8096/tcp' #Lo expongo porque quiero acceder a ese puerto en local
        networks:
            - sky_net
        security_opt:
            - systempaths=unconfined
            - apparmor=unconfined
        group_add:
            - 44 # video group
            - 107 #render group
        devices:
            - '/dev/dri:/dev/dri'
            - '/dev/dma_heap:/dev/dma_heap'
            - '/dev/mali0:/dev/mali0'
            - '/dev/rga:/dev/rga'
            - '/dev/mpp_service:/dev/mpp_service'
        volumes:
            - '${PATH_TO_APPDATA}/jellyfin/config:/config'
            - '${PATH_TO_APPDATA}/jellyfin/cache:/cache'
            - '${PATH_TO_NAS}/videos:/data'
        restart: unless-stopped

networks:
  sky_net:
    driver: bridge
    external: true
    name: sky_net
```

.env

```
PATH_TO_APPDATA=/dockerData/appdata
PATH_TO_NAS=/mnt/wdstorage/NAS
PATH_TO_HOME=/home/ubuntu
```

comprobar aceleracion grafica

```
docker exec -it jellyfin /usr/lib/jellyfin-ffmpeg/ffmpeg -v debug -init_hw_device rkmpp=rk -
init_hw_device opencl=ocl@rk
```

Jellyseerr

Que hace este servicio

Este servicio es un catalogo de peliculas, se puede vincular con jellyfin, plex y otras plataformas para extraer los usuarios y hacer que estos te puedan pedir pelicuas y series.

docker-compose.yaml

```
services:
    jellyseerr:
        image: fallenbagel/jellyseerr:latest
        container_name: jellyseerr
        #ports:
        #    - 8012:5055
        networks:
            - sky_net
        environment:
            - LOG_LEVEL=debug
            - TZ=America/Mexico_City
        volumes:
            - ${PATH_TO_APPDATA}/jellyseerr/config:/app/config
        restart: unless-stopped

networks:
  sky_net:
    driver: bridge
    external: true
    name: sky_net
```

.env

```
PATH_TO_APPDATA=/dockerData/appdata
```

Jdownloader

Que hace este servicio

Este servicio es una herramienta de descarga, te ayuda a descargar, descomprimir y administrar tus descargas.

docker-compose.yaml

```
services:
    jdownloader:
        image: jlesage/jdownloader-2:latest
        container_name: jdownloader
        #ports:
        #    - 8006:5800
        networks:
            - sky_net
        environment:
            - PUID=${APPUSER_PUID}
            - PGID=${APPUSER_PGID}
            - DARK_MODE=1
            - TZ=America/Mexico_City
        volumes:
            - ${PATH_TO_APPDATA}/jdownloader/config:/config
            - ${PATH_TO_DOWNLOADS}:/output


networks:
  sky_net:
    driver: bridge
    external: true
    name: sky_net
```

.env

```
PATH_TO_APPDATA=/dockerData/appdata
PATH_TO_NAS=
PATH_TO_DOWNLOADS=
```

```
APPUSER_PGID=1000
APPUSER_PUID=1000
```

Komga

Que hace este servicio

Este servicio es una pagina web para subir libros, mangas y otros documentos de textos con una pagina web muy atractiva y no tiene un formato para almacenar los archivos que soy muy problematico.

docker-compose.yaml

```
services:
    komga:
        image: gotson/komga:latest
        container_name: komga
        user: "${APPUSER_PUID}:${APPUSER_PUID}"
        #ports:
        #  - 8003:25600
        networks:
            - sky_net
        environment:
            - TZ=America/Mexico_City
        volumes:
            - '${PATH_TO_APPDATA}/komga:/config'
            - '${PATH_TO_NAS}/library:/data'
        restart: unless-stopped

networks:
  sky_net:
    driver: bridge
    external: true
    name: sky_net
```

.env

```
PATH_TO_APPDATA=/dockerData/appdata
PATH_TO_NAS=
APPUSER_PGID=1000
```

```
APPUSER_PUID=1000
```

Noip

Que hace este servicio

Este servicio es un cron para actualizar la ip del servicio ddns de noip.

docker-compose.yaml

```
services:
    noip:
        image: ghcr.io/noipcom/noip-duc:latest
        container_name: noip
        environment:
            - NOIP_USERNAME=${GROUP_SERVER}:${USER_SERVER}
            - NOIP_PASSWORD=${PASSWORD_SERVER}
            - NOIP_HOSTNAES=all.ddnskey.com
            - INTERVAL=30
        restart: unless-stopped
```

.env

```
GROUP_SERVER=
USER_SERVER=
PASSWORD_SERVER=
```

Odoo 18

Que hace este servicio

Este servicio es un ERP con Punto de Venta, pese a ser de pago tiene una gran parte Open Source que es incluso exesivo para pequeñas empresas y suficiente para medianas empresas

docker-compose.yaml

```
services:
    odoo-web:
        image: odoo:18
        container_name: odoo-web
        depends_on:
            - odoo-db
        #ports:
        #    - "8500:8069"
        networks:
            - sky_net
        volumes:
            - ${PATH_TO_APPDATA}/odoo/var:/var/lib/odoo
            - ${PATH_TO_APPDATA}/odoo/etc:/etc/odoo
            - ${PATH_TO_APPDATA}/odoo/extra-addons:/mnt/extra-addons
        environment:
            - HOST=odoo-db
            - USER=${POSTGRES_USER}
            - PASSWORD=${POSTGRES_PASSWORD}
        restart: unless-stopped

    odoo-db:
        image: postgres:15
        container_name: odoo-db
        networks:
            - sky_net
        environment:
```

```
            - POSTGRES_DB=${POSTGRES_DB}
            - POSTGRES_USER=${POSTGRES_USER}
            - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
        volumes:
            - ${PATH_TO_APPDATA}/odoo/db:/var/lib/postgresql/data
        restart: unless-stopped

networks:
  sky_net:
    driver: bridge
    external: true
    name: sky_net
```

.env

```
PATH_TO_APPDATA=/dockerData/appdata
POSTGRES_DB=
POSTGRES_USER=
POSTGRES_PASSWORD=
```

Qbittorrent

Que hace este servicio

Este servicio es una aplicacion para descargar y transmitir Torrents

docker-compose.yaml

```
services:
    qbittorrent:
        image: lscr.io/linuxserver/qbittorrent:latest
        container_name: qbittorrent
        #ports:
        #    - 8005:8082 #este es el puerto para la pagina http
        #    - 6881:6881
        #    - 6881:6881/udp
        networks:
            - sky_net
        environment:
            - PUID=${APPUSER_PUID}
            - PGID=${APPUSER_PGID}
            - TZ=America/Mexico_City
            - WEBUI_PORT=8082
        volumes:
            - ${PATH_TO_APPDATA}/qbittorrent/config:/config
            - ${PATH_TO_DOWNLOADS}:/downloads

networks:
  sky_net:
    driver: bridge
    external: true
    name: sky_net
```

.env

```
PATH_TO_APPDATA=/dockerData/appdata
PATH_TO_DOWNLOADS=
```

```
APPUSER_PGID=1000
APPUSER_PUID=1000
```

Sonarr

Que hace este servicio

Este servicio es para buscar, descargar, renombrar y organizar series, en mi caso solo lo uso para renombrar series.

docker-compose.yaml

```
services:
    sonarr:
        image: lscr.io/linuxserver/sonarr:latest
        container_name: sonarr
        #ports:
        #    - 8010:8989
        networks:
            - sky_net
        environment:
            - PUID=${APPUSER_PUID}
            - PGID=${APPUSER_PGID}
            - TZ=America/Mexico_City
        volumes:
            - ${PATH_TO_APPDATA}/radarr/config:/config
            - ${PATH_TO_NAS}/jellyfin:/tv
            - ${PATH_TO_DOWNLOADS}/downloads:/downloads



networks:
  sky_net:
    driver: bridge
    external: true
    name: sky_net
```

.env

```
PATH_TO_APPDATA=/dockerData/appdata
PATH_TO_NAS=
```

```
PATH_TO_DOWNLOADS=
APPUSER_PGID=1000
APPUSER_PUID=1000
```

vaultwarden

Que hace este servicio

Este servicio es un servidor para almacenar contraseñas y documentos personales codificados por una contraseña maestra, se puede vincular con la aplicacion bitwarden.

docker-compose.yaml

```
services:
    vaultwarden:
        image: vaultwarden/server:latest
        container_name: vaultwarden
        #ports:
        #    - 8007:80
        environment:
            - SIGNUPS_ALLOWED=true
        volumes:
            - ${PATH_TO_APPDATA}/vaultwarden:/data
        restart: unless-stopped
        networks:
            - sky_net


networks:
  sky_net:
    driver: bridge
    external: true
    name: sky_net
```

.env

```
PATH_TO_APPDATA=/dockerData/appdata
```

Vikunja

Que hace este servicio

Este servicio es para organizar las actividades mediante tarjetas

docker-compose,yaml

```
services:
    vikunja:
        image: vikunja/vikunja:latest
        container_name: vikunja
        #ports:
        #  - "8004:3456"
        networks:
            - sky_net
        environment:
          VIKUNJA_SERVICE_ENABLEREGISTRATION: 0
          VIKUNJA_SERVICE_PUBLICURL: https://tu-dominio
          VIKUNJA_SERVICE_JWTSECRET: "${VIKUNJA_SERVICE_JWTSECRET}"
          VIKUNJA_DATABASE_PATH: "/db/vikunja.db"
        volumes:
          - ${PATH_TO_APPDATA}/vikunja/files:/app/vikunja/files
          - ${PATH_TO_APPDATA}/vikunja/db:/db
          - ${PATH_TO_APPDATA}/vikunja/.cache:/.cache
        restart: unless-stopped


networks:
  sky_net:
    driver: bridge
    external: true
    name: sky_net
```

.env

```
PATH_TO_APPDATA=/dockerData/appdata
VIKUNJA_SERVICE_JWTSECRET=
```

wireguard

Que hace este servicio

Este servicio es una vpn

docker-compose.yaml

```
services:
    wireguard:
        image: lscr.io/linuxserver/wireguard:latest
        container_name: wireguard
        network_mode: host
        cap_add:
          - NET_ADMIN
          - SYS_MODULE
        environment:
          - PUID=${APPUSER_PUID}
          - PGID=${APPUSER_PGID}
          - TZ=America/Mexico_City
          - PEERS=4
          - LOG_CONFS=true
          - SERVERURL=wg.tinyfox.ddns.me
          - SERVERPORT=51820
        volumes:
          - ${PATH_TO_APPDATA}/wireguard/config:/config
          - ${PATH_TO_APPDATA}/wireguard/lib/modules:/lib/modules
        # ports:
        #   - 51820:51820/udp  # inecesario por que use la propiead network_mode: host
        restart: unless-stopped
```

.env

```
PATH_TO_APPDATA=/dockerData/appdata
APPUSER_PGID=1000
```

```
APPUSER_PUID=1000
```

DuckDns

docker-compose.yaml

```
services:
  duckdns:
    image: lscr.io/linuxserver/duckdns:latest
    container_name: duckdns
    environment:
      - SUBDOMAINS=ejemplo1,ejemplo2,ejemplo3
      - TOKEN=${TOKEN_DUCKDNS}
      - LOG_FILE=false #optional
    volumes:
      - ${PATH_TO_APPDATA}/duckdns/config:/config #optional
    restart: unless-stopped
```

.env

```
PATH_TO_APPDATA=
TOKEN_DUCKDNS=
```

Nextcloud

docker-compose.yaml

```
services:
  mariadb:
    image: mariadb:lts-ubi9
    user: "${APPUSER_PUID}:${APPUSER_PGID}"
    #ports:
    #  - 3306:3306
    networks:
      - sky_net
    command: --transaction-isolation=READ-COMMITTED --log-bin=binlog --binlog-format=ROW
    volumes:
      - ${PATH_TO_APPDATA}/nextcloud/db:/var/lib/mysql
    environment:
      - MYSQL_ROOT_PASSWORD=nextcloud
      - MYSQL_PASSWORD=nextcloud
      - MYSQL_DATABASE=nextcloud
      - MYSQL_USER=nextcloud
    restart: always

  nextcloud:
    image: lscr.io/linuxserver/nextcloud:latest
    container_name: nextcloud
    #ports:
    #  - 8083:443
    networks:
      - sky_net
    depends_on:
      - mariadb
    environment:
      - PUID=${APPUSER_PUID}
      - PGID=${APPUSER_PGID}
      - TZ=America/Mexico_City
    volumes:
      - ${PATH_TO_APPDATA}/nextcloud/config:/config
```

```
      - ${PATH_TO_DATA}/nextcloud/data:/data
      - ${PATH_TO_NAS}:/mnt/storage
    restart: unless-stopped

networks:
  sky_net:
    driver: bridge
    external: true
    name: sky_net
```

.env

```
PATH_TO_APPDATA=
PATH_TO_DATA=
PATH_TO_NAS=
APPUSER_PUID=
APPUSER_PGID=
```

Openssh Server

docker-compose.yaml

```
services:
  openssh-server:
    image: lscr.io/linuxserver/openssh-server:latest
    container_name: openssh-server
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=America/Mexico_City
      - USER_PASSWORD=${USER_PASSWORD} #optional
      - USER_NAME=${USER_NAME} #optional
      - PASSWORD_ACCESS=true #optional
      #- SUDO_ACCESS=true #optional
    volumes:
      - ${PATH_TO_APPDATA}/opensshserver/config:/config
      - ${PATH_TO_NAS}/data/movies:/mnt/movies:ro
    ports:
      - 2222:2222
    restart: unless-stopped
```

.env

```
PATH_TO_APPDATA=
PATH_TO_NAS=
USER_NAME=
USER_PASSWORD=
```

Minecraft Server

docker-compose.yaml

```
services:
  mc-server:
    image: itzg/minecraft-server:java24-graalvm
    container_name: mc-server
    restart: unless-stopped
    tty: true
    stdin_open: true
    ports:
      - "25565:25565"
    environment:
      EULA: "TRUE"
      VERSION: "LATEST"
      ONLINE_MODE: "TRUE" # set false to play whit out mojang acount
      TYPE: "PAPER"
      # WORLD: "WORLD_NAME"
      # SEED: "SEED"
      SNOOPER_ENABLED: "false"
      INIT_MEMORY: "2G"
      MAX_MEMORY: "8G"
      ENABLE_COMMAND_BLOCK: "false"
      DIFFICULTY: "hard"
      SERVER_NAME: "SERVER_NAME"
      WHITELIST: "acount1,acount2,acount3"
      ENABLE_WHITELIST: "true"
    volumes:
      - ${PATH_TO_APPDATA}/minecraft/data:/data
```

.env

```
PATH_TO_APPDATA=
```

NFS Server

Servidor:

```
  nfs-server:
    image: itsthenetwork/nfs-server-alpine:latest-arm
    container_name: nfs-server
    privileged: true                  # Necesario para NFS
    restart: unless-stopped
    # network_mode: "host"              # Exponer directamente el puerto 2049
    ports:
      - "2049:2049/tcp"               # Puerto NFS
      - "2049:2049/udp"               # Puerto NFS
    environment:
      TZ: America/Mexico_City
      SHARED_DIRECTORY: /nfsshare     # Directorio que se comparte por NFS
      # READ_ONLY: "true"            # Opcional: solo lectura
      # SYNC: "true"                 # Opcional: modo síncrono
      # PERMITTED: "192.168.1.*"    # Opcional: restringir IPs
    volumes:
      - /mnt/nas/backups/jellyfin:/nfsshare # Carpeta a compartir
      # - ./config/nfs/exports:/etc/exports:ro # Permisos avanzados
```

Cliente:

1. Install NFS Client Packages:
First, install the necessary NFS client packages on your Debian machine:

```
sudo apt update
sudo apt install nfs-common
```

2. Create a Local Mount Point:
Create a directory on your Debian machine where you want to mount the NFS share. This will be the local access point for the remote files.

```
sudo mkdir /mnt/nfs
```

3. Mount the NFS Share:
Now, you can mount the NFS share from the server to your local mount point. You'll need the IP address or hostname of the NFS server and the path to the shared directory on the server.

```
sudo mount -t nfs <NFS_SERVER_IP_OR_HOSTNAME>:/path/to/shared/directory /mnt/nfs


mount -t nfs 192.168.31.100:/ /mnt/nfs
```

4. Verify the Mount:
You can verify that the NFS share has been successfully mounted using the `df -h` command:

```
df -h
```

This command will display a list of mounted filesystems, including your newly mounted NFS share.
5. Mount NFS at Boot (Optional):
To automatically mount the NFS share every time your Debian machine boots, you can add an entry to the `/etc/fstab` file.
Open `/etc/fstab` with a text editor (e.g., `nano` or `vim`):

```
sudo nano /etc/fstab
```

Add a line similar to the following, replacing the placeholders with your specific details:

```
<NFS_SERVER_IP_OR_HOSTNAME>:/path/to/shared/directory /mnt/nfs_share nfs defaults


192.168.31.100:/      /mnt/nfs     nfs    defaults,noatime,_netdev,x-systemd.automount
0     0
```