Things to do with a pi

Setup an HTPC server, pihole, host your cloud and a lot more

- Your own Google services / Nextcloud
- Media streaming on your pi / Your own netflix

Your own Google services / Nextcloud

Now hosting nextcloud is a pretty vanilla setup , all you gotta do is get yourself a compose file and do a docker-compose up and you are pretty much good to go

Docker-compose file

```
version: "2.1"
services:
  nextcloud:
    image: linuxserver/nextcloud
    container_name: nextcloud
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Asia/Kolkata
    volumes:
      - /mnt/hdd/config:/config
      - /mnt/hdd/data:/data
    ports:
      - 993:993
      - 587:587
      - 9443:443
      - 8080:80
    restart: unless-stopped

    # PostgreSQL
  postgres:
    container_name: postgres
    environment:
      - POSTGRES_PASSWORD=PleaseChangeMeandEnterAGoodPasswordHere
      - POSTGRES_USER=nextcloud
```

```
    image: postgres:latest
    restart: always
    volumes:
      - "/mnt/hdd/postgres/init:/docker-entrypoint-initdb.d/"
      - "/mnt/hdd/postgres/data:/var/lib/postgresql/data"
      - "/etc/localtime:/etc/localtime:ro"
    # Redis
  redis:
    container_name: redis
    image: redis:latest
    restart: always
```

> This setup assumes that your disk is mounted at /mnt/hdd which you want to use for your data storage {.is-info}

- Now just do a docker-compose up -d
- While installation just select postgres as a backend with the following details

Nginx config

Here is the nginx config in case you want to make this available publically

```
server {
    listen 80;
    server_name <domain>;
    return 301 https://$host$request_uri;  # redirect http to https
}

server {
    listen 443 ssl http2;
    server_name <domain>;

    ssl_certificate     /etc/letsencrypt/live/<domain>/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/<domain>/privkey.pem;

    port_in_redirect off;
```

```
    proxy_buffering off;
    proxy_http_version 1.1;     # Properly proxy websocket connections
    proxy_read_timeout 300s;    # terminate websockets afer 5min of inactivity
    proxy_set_header X-Forwarded-Host $host;
    proxy_set_header X-Forwarded-Server $host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_set_header X-Forwarded-Protocol $scheme;


    location / {
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-Proto https;
            proxy_pass  http://<ip>:<port>
    }
}
```

Media streaming on your pi / Your own netflix

Basic setup

> The guide here assumes that you are running Ubuntu server 20.04.02 LTS on Raspberry pi 4 with 4gb of RAM . Though since most of this is dockerized it can be replicated on other setups as well {.is-info}

```
version: "2.1"
services:
  jellyfin:
    image: ghcr.io/linuxserver/jellyfin:nightly
    container_name: jellyfin
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Asia/Kolkata
    volumes:
      - /mnt/hdd/media/config:/config
      - /mnt/hdd/media/tvs:/data/tvshows
      - /mnt/hdd/media/movies:/data/movies
      - /mnt/hdd/media/music:/data/music
      - /opt/vc/lib:/opt/vc/lib #optional
    ports:
      - 8096:8096
      - 8920:8920 #optional
    devices:
      - /dev/dri:/dev/dri #optional
      - /dev/vchiq:/dev/vchiq #optional
      - /dev/video10:/dev/video10 #optional
      - /dev/video11:/dev/video11 #optional
```

```
      - /dev/video12:/dev/video12 #optional
    restart: unless-stopped
```

Here are some things you can change

- ghcr.io/linuxserver/jellyfin:nightly to ghcr.io/linuxserver/jellyfin:latest if you don't want to live on the bleeding edge of things like me
- volumes ofcourse need to be changed /mnt/hdd is the local volume where my hdd is mounted on my machine you need to change this to match where your music, TV shows and Movies are located accordingly
- the ports can be changed so you can change 8096:8096 to 8080:8096 that way it would be available on your machine on port 8080
- 8920 is an https port if you want https on the level of jellyfin but since this service is running on our home network it doesnt absolutely need to be on https.
- Coming to devices all the devices you can see here are used for hardware acceleration. If you want to use hardware acceleration(which I recommend using) you can mount these devices onto the container. Below I will provide the explaination for hardware acceleration in detail

Basic steps for setup

- Create a directory called jellyfin
- Paste this compose file in a file called docker-compose.yml(make sure that directory doesn't contain any other docker-compose.yml files)
- docker-compose up -d
- Go to <ip-addr of your pi>:8096 (or whatever port you set in the compose-file)
- Follow the setup

Hardware Acceleration and do you need it ?

Hardware acceleration in the most layman of terms means using something for something that was it specialized to do. For example a GPU or Graphics Procesing Unit does what it says on the box. It processes graphics . Now you can do the same thing on your CPU but you wouldn't get the same performence from it. A GPU is specefically meant to process graphics hence the perfomance.

Hardware acceleration in case of jellyfin comes in use when you want to transcode the video . Transcoding means decoding from one format into another and then encoding it again . So for

example a lot of browsers dont support H.265 encoded video and if you ask jellyfin to play that directly it will refuse . It will instead transcode it into another format and then send the video to your browser.

The transcoding part is where you need to Hardware acceleration and trust me when you are trying to transcode a big video hardware acceleration really really helps.

How to enable hardware acceleration on Ubuntu 20 on raspi

Hardware acceleration is disabled by default since the pi isn't really meant to be used as a device which might be used for graphics intensive purposes such as playing games.

> Try doing and ls /dev/dri if you see some devices here and then hardware acceleration is already enabled and you don't need to do anything else {.is-info}

Add this to `/boot/firmware/usercfg.txt`

```
# Place "config.txt" changes (dtparam, dtoverlay, disable_overscan, etc.) in
# this file. Please refer to the README file for a description of the various
# configuration files on the boot partition.
dtoverlay=vc4-fkms-v3d
max_framebuffers=2
gpu_mem=128
hdmi_enable_4kp60=1
```

Now reboot your pi and you should be good to go

Enable hardware accel. in Jellyfin

To do that you shoudl go to Dashboard -> Playback and then select Video Acceleration API from the dropdown under hardware acceleration Rest everything can be default.

Exposing Jellyfin to the web

The not so secure way

- The easiest but not so secure way to expose jellyfin to the web is to open ports in your router and get a DDNS setup . Then you can access jellyfin over your domain:8096 , which btw looks straight up ugly
- You can also open port 80 and 443 and use nginx and certbot to use as reverse proxy and then expose jellyfin to the web over ssl using certbot(letsencrypt) (Only applicable in case your ISP allows you to do that which most ISPs don't)

The more secure way

- Patching holes in the firewall that protects your home network isn't considered very secure but there is another way
- Get a small cloud VM from whatever provider and host a VPN on it and then that way you will be easily able to connect your pi to this VPN and serve whatever content onto the dark place we call internet
- When you have a VPN configured you can use nginx as a reverse proxy to serve that via ssl

Sample configuration for nginx

Here is a sample configuration for nginx if you are using the VPN route

```
server {
    listen 443 ssl;
    listen [::]:443 ssl;

    server_name <website domain here>;

    #include /config/nginx/ssl.conf;
    ssl_trusted_certificate /etc/letsencrypt/live/<domainhere>/chain.pem;
    ssl_stapling on;
```

```
    ssl_stapling_verify on;
    ssl_certificate /etc/letsencrypt/live/<domainhere>/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/<domainhere>/privkey.pem; # managed by
Certbot

    client_max_body_size 0;

    location / {
        include /etc/nginx/proxy.conf;
        #resolver 127.0.0.11 valid=30s;
        set $upstream_app <ip of your jellyfin server>;
        set $upstream_port 8096;
        set $upstream_proto http;
        proxy_pass $upstream_proto://$upstream_app:$upstream_port;

        proxy_set_header Range $http_range;
        proxy_set_header If-Range $http_if_range;
    }
    # to be able to use syncplay over nginx
    location ~ (/jellyfin)?/socket {
        include /etc/nginx/proxy.conf;
        #resolver 127.0.0.11 valid=30s;
        set $upstream_app <ip of your jellyfin server>;
        set $upstream_port 8096;
        set $upstream_proto http;
        proxy_pass $upstream_proto://$upstream_app:$upstream_port;

    }
}
```

> Go through the nginx config file and change whatever needs to be changed this will not work as is {.is-warning}

Here is the proxy.conf file save this in `/etc/nginx/proxy.conf`

```
proxy_next_upstream error timeout invalid_header http_500 http_502 http_503;

# Proxy Connection Settings
proxy_buffers 32 4k;
```

```
proxy_connect_timeout 240;
proxy_headers_hash_bucket_size 128;
proxy_headers_hash_max_size 1024;
proxy_http_version 1.1;
proxy_read_timeout 240;
proxy_redirect  http://  $scheme://;
proxy_send_timeout 240;

# Proxy Cache and Cookie Settings
proxy_cache_bypass $cookie_session;
#proxy_cookie_path / "/; Secure"; # enable at your own risk, may break certain apps
proxy_no_cache $cookie_session;

# Proxy Header Settings
proxy_set_header Host $host;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Host $host;
proxy_set_header X-Forwarded-Proto https;
proxy_set_header X-Forwarded-Ssl on;
proxy_set_header X-Real-IP $remote_addr;
```