Software & packages

- Image manipulation
- yt-dlp
- Python packages and commands
- spotdl
- ffmpeg
- IPFS
- Wireguard
- Node & NVM
- Syncthing
- Screen
- Tailscale

Image manipulation

Compression

Install `sudo apt-get install imagemagick`

Compress command `mogrify -quality 80% *.jpg`

note: this will overwrite images

yt-dlp

Download video as mp3

```
yt-dlp -x --audio-format mp3 URL_HERE
```

Convert webm to mp4

```
ffmpeg -i input.webm -c:v libx264 -preset slow -crf 22 -c:a aac -b:a 128k output.mp4
```

Download video as mp4

```
yt-dlp -f "bestvideo[ext=mp4]+bestaudio[ext=m4a]/best[ext=mp4]/best" SOURCE_URL
```

Convert video codec to x264

```
ffmpeg -i input.mp4 -c:v libx264 -crf 23 -preset medium -c:a aac -b:a 192k output.mp4
```

Python packages and commands

Managing Virtual Envrionments

Create a new venv `python3 -m venv /path/to/new/virtual/environment`

Activate the new venv Linux / Mac `source myenv/bin/activate`

Activate an envrionment on windows `myenv\Scripts\activate`

Exit the venv `deactivate`

Installing and uninstalling packages

Installing a package `pip install xyz`

Uninstalling a package `pip uninstall xyz`

Useful packages

- aqlalchemy - SQL Alchemy
- snowflake-sqlalchemy - SQL Alchemy for snowflake

spotdl

Install & Setup

> Best to load a venv first

Installing spotdl `pip install spotdl`

Open the config file at `.spotdl/config.json` and update "output" to be `"{album}/{title}.{output-ext}"`

Updating

`pip install --upgrade spotdl`

ffmpeg

Installing

```
sudo apt install ffmpeg
```

Compress a video to h265

```
ffmpeg -i example_h264.mp4 -c:v libx265 -crf 24 -preset fast -c:a aac -b:a 128k example_h265.mp4
```

CRF Info

For H.264 (x264):

The typical CRF range is 18-28.

CRF 18: Near lossless quality (very high quality, larger file size).

CRF 23: Default value, good balance between quality and file size.

CRF 28: Noticeable quality loss, much smaller file size.

Overwright a videos audio with a new audio track

```
ffmpeg -i video_to_overwright.mkv -i audio_input.mp3 -c:v copy -map 0:v:0 -map 1:a:0 output.mp4
```

MP4 to MP3

```
ffmpeg -i filename.mp4 filename.mp3
```

Change video format

```
ffmpeg -i input.m4v -c copy output.mp4
```

IPFS

Installation for Linux

1 - Download the package

```
wget https://dist.ipfs.tech/kubo/v0.31.0/kubo_v0.31.0_linux-amd64.tar.gz
```

2 - Unzip the file

```
tar -xvzf kubo_v0.39.0_linux-amd64.tar.gz
```

3 - Move into the kubo folder

```
cd kubo
```

4 - Run the install script

```
sudo bash install.sh
```

5 - Test the installation worked

```
ipfs --version
```

Editing the config

Show Config:

```
ipfs config show
```

Change connection count

```
ipfs config Swarm.ConnMgr.HighWater 1000 --json
ipfs config Swarm.ConnMgr.LowWater 500 --json
```

Enable filestore (Allow import with no copy)

```
ipfs config --json Experimental.FilestoreEnabled true
sudo systemctl restart ipfs
ipfs config Experimental.FilestoreEnabled
```

IPFS as a service

Open a new file at `/etc/systemd/system/ipfs.service`

```
[Unit]
Description=IPFS daemon
After=network.target

[Service]
ExecStart=/usr/local/bin/ipfs daemon
Restart=on-failure
User=conor
Group=conor
Environment=IPFS_PATH=/home/conor/.ipfs

[Install]
WantedBy=default.target
```

`WantedBy=default.target` means the service will start up at boot

get it to start at boot

```
sudo systemctl enable ipfs
```

and start it

```
sudo systemctl start ipfs
```

Usage

Check filestore is enabled and working

```
ipfs config Experimental.FilestoreEnabled
```

List files in filestore

```
ipfs filestore ls
```

Import files to filestore

```
ipfs add --nocopy --recursive --cid-version=1 /path/to/your/folder
```

Wireguard

Setup Script

```
#!/bin/bash

# Checks to see if script is being run as root
if [ "$EUID" -ne 0 ]; then
  echo "Please run as root"
  exit
fi

sudo apt install wireguard

# Deletes keys in case this script is being run again
rm -f /etc/wireguard/*

# Create necessary directories, including parents
mkdir -p /etc/wireguard/client



# Create and store server private and public key
wg genkey | tee /etc/wireguard/server_priv.key | wg pubkey | tee /etc/wireguard/server_pub.key

# Get the server public and private key as well as the network interface
server_priv_key=$(cat /etc/wireguard/server_priv.key)
server_pub_key=$(cat /etc/wireguard/server_pub.key)
network_interface=$(ip -o -4 route show to default | awk '{print $5}')



# Client
# Create client public and private key and store it in vars for later
wg genkey | tee /etc/wireguard/client/client_priv.key | wg pubkey | tee /etc/wireguard/client/client_pub.key
client_priv_key=$(cat /etc/wireguard/client/client_priv.key)
client_pub_key=$(cat /etc/wireguard/client/client_pub.key)
```

```
# Create initial client config
echo "[Interface]
PrivateKey = $client_priv_key
Address = 10.0.0.2/24

[Peer]
PublicKey = $server_pub_key
Endpoint = servername_or_ip:51820
AllowedIPs = 0.0.0.0/0
" > /etc/wireguard/client/wg0.conf



# Create the config file for wireguard
echo "[Interface]
Address = 10.0.0.1/24
SaveConfig = true
ListenPort = 51820
PrivateKey = $server_priv_key
PostUp = iptables -A FORWARD -i %i -j ACCEPT; iptables -t nat -A POSTROUTING -o $network_interface -j
MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT; iptables -t nat -D POSTROUTING -o $network_interface -j
MASQUERADE

[Peer]
PublicKey = $client_pub_key
AllowedIPs = 10.0.0.2/24
" > /etc/wireguard/wg0.conf

# Change permissions so only root can access the files
chmod 600 /etc/wireguard/server_priv.key
chmod 600 /etc/wireguard/wg0.conf

# Allow 51820 through ufw
ufw allow 51820/udp

# Start wireguard and set it to auto start on boot
wg-quick up wg0
systemctl enable wg-quick@wg0
```

Setup - Manual

1 - Update system and install wireguard

```
apt update && apt upgrade && apt install wireguard
```

2 - Make the necessary folders

```
mkdir -p /etc/wireguard/client
```

3 - Create the server's public and private keys

```
wg genkey | tee /etc/wireguard/server_priv.key | wg pubkey | tee /etc/wireguard/server_pub.key
```

4 - Find the servers network interface

```
ip -o -4 route show to default | awk '{print $5}
```

5 - Nano into

```
/etc/wireguard/wg0.conf
```

(Make sure to add network interface and server private key)

```
[Interface] Address = 10.0.0.1/24
SaveConfig = true
ListenPort = 51820
PrivateKey = SERVER_PRIVATE_KEY
PostUp = iptables -A FORWARD -i %i -j ACCEPT; iptables -t nat -A POSTROUTING -o NETWORK_INTERFACE_HERE -j MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT; iptables -t nat -D POSTROUTING -o NETWORK_INTERFACE_HERE -j MASQUERADE
```

6 - Update file permissions so only root can read the config file and private key

```
chmod 600 /etc/wireguard/server_priv.key chmod 600 /etc/wireguard/wg0.conf
```

7 - Allow the vpn port through UFW

```
ufw allow 51820/udp
```

8 - Start wireguard and set it to auto start at boot

```
wg-quick up wg0 systemctl enable wg-quick@wg0
```

9 - Create the client keys

```
wg genkey | tee /etc/wireguard/client/priv.key | wg pubkey | tee /etc/wireguard/client/pub.key
```

10 - Create the client config file (copy to client device)

```
[Interface] PrivateKey = CLIENT_PRIVATE_KEY
Address = 10.0.0.2/24 [Peer]
PublicKey = SERVER_PUBLIC_KEY
Endpoint = SERVER_IP_OR_DOMAIN:51820
AllowedIPs = 0.0.0.0/0
```

11 - Add peer info to the server config

```
[Interface] Address = 10.0.0.1/24
SaveConfig = true
ListenPort = 51820
PrivateKey = SERVER_PRIVATE_KEY
PostUp = iptables -A FORWARD -i %i -j ACCEPT; iptables -t nat -A POSTROUTING -o NETWORK_INTERFACE_HERE -j MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT; iptables -t nat -D POSTROUTING -o NETWORK_INTERFACE_HERE -j MASQUERADE

# ================
[Peer]
PublicKey = PUBLIC_KEY
AllowedIPs = 10.0.0.2/24
```

Setup with SeaBee's setup script

```
wget https://raw.githubusercontent.com/seabee33/wireguard_helper/refs/heads/main/wg_helper.py && chmod +x wg_helper.py && sudo python3 wg_helper.py
```

Auto start at boot

1 - ensure the client config is at `/etc/wireguard/wg0.conf`
2 - enable it to start at boot with `sudo systemctl enable wg-quick@wg0`

Node & NVM

1 - install node version manager (NVM)

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.3/install.sh | bash
```

2 - install node

```
nvm install node
```

General steps

1 - install components

```
npm create vite@latest my-charting-site --template react
cd my-charting-site
npm install

npm install -D tailwindcss@3 postcss autoprefixer
npx tailwindcss init -p
```

2 - test

```
npm run dev
```

build

```
npm run build
```

Syncthing

Screen

Screen lets you make virtual terminals with

```
screen
```

New session with name

```
screen -S session_name
```

Tailscale

Client setup

1. Install Tailscale client

```
curl -fsSL https://tailscale.com/install.sh | sh
```

2. Create a pre-auth key on your Headscale server

```
docker exec headscale headscale preauthkeys create --user 1 --reusable --expiration 24h
```

Copy the key that's generated.

3. Connect to your Headscale server

```
sudo tailscale up --login-server=https://headscale.conorbriggs.com.au --authkey=YOUR_PREAUTH_KEY --hostname=rpi3
```

Replace YOUR_PREAUTH_KEY with the key from step 2.

4. Verify connection

```
# On Raspberry Pi:
sudo tailscale status

# On your Headscale server:
docker exec headscale headscale nodes list
```

You should see your Raspberry Pi listed with the hostname "raspberry-pi" and an IP like 100.64.0.2.