Linux

- Mounting & Unmounting
- Symbolic Links
- Users & Permissions
- General Management & commands
- SSH
- Fail2Ban
- wget
- Bash
- UFW
- Swap
- Disk Stuff
- Auto updates
- SSH Tunnel

General Info

General Info

Directory Structure

Directory	Purpose
`/`	Root of the filesystem
`/bin`	Essential command binaries
`/sbin`	Essential system binaries
`/etc`	Configuration files
`/var`	Variable data (logs, cache, etc.)
`/var/log`	System and application logs
`/usr`	User applications and binaries
`/home`	User home directories
`/root`	Root user’s home directory
`/boot`	Boot files and kernel
`/opt`	Optional third-party software
`/tmp`	Temporary files
`/dev`	Device files (disks, terminals)
`/proc`	Process and system info
`/sys`	Kernel-related info
`/lib`, `/lib64`	Shared libraries
`/media`	Auto-mounted removable media
`/mnt`	Temporary mount points
`/srv`	Data for services (web, FTP)

Web server management

Web server management

Apache setup & Virtual servers

Initial setup

Install packages `sudo apt install apache2 mariadb-server libapache2-mod-php php-mysql -y`

Open ports at `cd /etc/apache2/ports.conf` and add the line `Listen NEW_PORT_NUMBER`

Creating virtual servers

Create a file named NAME.conf and add something like this

```
<VirtualHost *:80>
        ServerName pdb.conorbriggs.com.au
        DocumentRoot /mnt/drives/10tb/10tb/web_servers/pdb/

<Directory /mnt/drives/10tb/10tb/web_servers/pdb/>
        Options -Indexes -FollowSymLinks
        AllowOverride All
        Require all granted
</Directory>
ErrorLog /var/log/apache2/pdb-error.log
CustomLog /var/log/apache2/pdb-access.log combined

RewriteEngine on
</VirtualHost>
```

Enable the site & restart apache with `sudo a2ensite site.conf && systemctl restart apache2`

SSL Certificate management

Install certbot and run setup

```
sudo apt install certbot python3-certbot-apache
a2enmod ssl
sudo certbot --apache
```

List all certificates with `certbot certificates`
Delete a certificaite `certbot delete --cert-name CERT_NAME_HERE`

Managing the apache2 service

Start, stop & restart `sudo service apache2 start/restart/stop`

Enable a config `sudo a2ensite SITE_CONF_FILE.conf`
Disable a config `sudo a2dissite SITE_CONF_FILE.conf`

Web server management

SQL

After install

ensure secure access `sudo mysql_secure_installation`

Logging in

`mysql -u USERNAME -p`

add `-h HOSTNAME IP` when logging in remotely

Creating a user

```
CREATE USER 'NEW_USERNAME'@'localhost' IDENTIFIED BY 'NEW_PASSWORD';
```

Creating a database

```
CREATE DATABASE DATABASE_NAME
```

Allow user to modify Database

```
GRANT ALL PRIVILEGES ON DATABASE_NAME.* TO 'USERNAME'@'localhost';
```

Update permissions

`FLUSH PRIVILEGES;`

Open Database to the world

1. Open file at `/etc/mysql/maridadb.conf.d/50-server.cnf`
2. Edit line `bind-address = 127.0.0.1` to -> `bind-address = 0.0.0.0`
3. Restart db service `systemctl restart mariadb`
4. Allow port through firewall with `ufw allow 3306`

Automated backup script

```
#!/bin/bash

# === Config ===
BACKUP_DIR="/var/backups/mysql"
DATE=$(date +"%Y-%m-%d_%H-%M")
DB_USER="backupuser"
DB_PASS="yourpassword"

# Create backup dir if not exist
mkdir -p "$BACKUP_DIR"

# Dump all databases
mysqldump -u "$DB_USER" -p"$DB_PASS" --all-databases --single-transaction > "$BACKUP_DIR/mariadb-$DATE.sql"

# Optional: Compress the backup
gzip "$BACKUP_DIR/mariadb-$DATE.sql"

# Delete backups older than 7 days
find "$BACKUP_DIR" -name "*.sql.gz" -type f -mtime +7 -delete
```

SSL & Certificate management

Install certbot and run setup

```
sudo apt install certbot python3-certbot-apache
a2enmod ssl
sudo certbot --apache
```

List all certificates

```
certbot certificates
```

Delete a certificaite

```
certbot delete --cert-name CERT_NAME_HERE
```

PostgreSQL

Install

Install postgresql and postgis for coordinates data

```
sudo apt install postgresql postgis -y
```

Login

```
sudo -u postgres psql
```

Change password to default account

```
ALTER USER postgres WITH PASSWORD 'new password';
```

Restrict remote access to localhost only

Open the config file

```
nano /etc/postgresql/15/main/postgresql.conf
```

uncomment this line

```
listen_addresses = 'localhost'
```

if you need remote access

```
listen_addresses = '0.0.0.0'
```

and Edit `/etc/postgresql/15/main/pg_hba.conf` and add a line like:

```
host    all             all             192.168.1.0/24          scram-sha-256
```

SQL Reference

Create a table that auto deletes on other table deletion

```
CREATE TABLE tradie_postcode_covered (
    tradie_id INT,
    postcode VARCHAR(10),
    PRIMARY KEY (tradie_id, postcode), -- primary key ensures no tradie can be in the same postcode twice
    FOREIGN KEY (tradie_id) REFERENCES tradies(id) ON DELETE CASCADE
);
```

Mail Server

Initial setup

1 - Updates

```
sudo apt update && sudo apt upgrade -y
```

2 - Install docker and docker compose

```
# Install required packages
sudo apt install -y apt-transport-https ca-certificates curl gnupg lsb-release

# Add Docker's official GPG key
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

# Set up the stable repository
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/debian $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Install Docker
sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin

# Add your user to docker group
sudo usermod -aG docker $USER
```

3 - Create DNS Records

- **A record**: `mail.yourdomain.com` → your server IP
- **MX record**: `@` → `mail.yourdomain.com` (priority 10)
- **TXT record (SPF)**: `v=spf1 mx ~all`

4 - Install and configure mailserver

```
# Create directory for mail server
mkdir -p ~/mailserver
cd ~/mailserver

# Download docker-compose.yml and .env template
wget https://raw.githubusercontent.com/docker-mailserver/docker-mailserver/master/compose.yaml
wget https://raw.githubusercontent.com/docker-mailserver/docker-mailserver/master/mailserver.env

# Rename for easier use
mv compose.yaml docker-compose.yml
mv mailserver.env .env
```

5 - Edit .env variables

- `HOSTNAME=mail`
- `DOMAINNAME=yourdomain.com`
- `OVERRIDE_HOSTNAME=mail.yourdomain.com`
- `ENABLE_SPAMASSASSIN=1`
- `ENABLE_CLAMAV=1`
- `ENABLE_FAIL2BAN=1`
- `SSL_TYPE=letsencrypt` (or `manual` if you have your own certs)
- `ACCOUNT_PROVISIONER=FILE`

6 - Edit docker-compose.yml

```
services:
  mailserver:
    image: ghcr.io/docker-mailserver/docker-mailserver:latest
    container_name: mailserver
    # Provide the FQDN of your mail server here (Your DNS MX record should point to this value)
    hostname: mx.home.conorbriggs.com.au
```

```
    env_file: .env
    # More information about the mail-server ports:
    # https://docker-mailserver.github.io/docker-mailserver/latest/config/security/understanding-the-ports/
    ports:
      - "25:25"    # SMTP  (explicit TLS => STARTTLS, Authentication is DISABLED => use port 465/587 instead)
      - "143:143"  # IMAP4 (explicit TLS => STARTTLS)
      - "465:465"  # ESMTP (implicit TLS)
      - "587:587"  # ESMTP (explicit TLS => STARTTLS)
      - "993:993"  # IMAP4 (implicit TLS)
    volumes:
      - ./mail-data/:/var/mail/
      - ./mail-state/:/var/mail-state/
      - ./mail-logs/:/var/log/mail/
      - ./config/:/tmp/docker-mailserver/
      - /etc/localtime:/etc/localtime:ro
      - /etc/letsencrypt:/etc/letsencrypt:ro
    restart: always
    stop_grace_period: 1m
    # Uncomment if using `ENABLE_FAIL2BAN=1`:
    cap_add:
      - NET_ADMIN
    healthcheck:
      test: "ss --listening --ipv4 --tcp | grep --silent ':smtp' || exit 1"
      timeout: 3s
      retries: 0
    networks:
      - mailserver-network
networks:
  mailserver-network:
    driver: bridge
```

Connecting

IMAP (Incoming):

- Server: `mx.home.conorbriggs.com.au`

- Port: `993` (IMAPS with SSL/TLS)
- Security: SSL/TLS
- Authentication: Normal password

SMTP (Outgoing):

- Server: `mx.home.conorbriggs.com.au`
- Port: `587` (with STARTTLS) or `465` (with SSL/TLS)
- Security: STARTTLS (port 587) or SSL/TLS (port 465)
- Authentication: Normal password

Container Management

Basic Container Operations

Start the mailserver

```
docker compose up -d
```

Stop the mailserver

```
docker compose down
```

Restart the mailserver

```
docker compose restart
```

View logs (live)

```
docker compose logs -f
```

View logs (last 100 lines)

```
docker compose logs --tail=100
```

Check container status

```
docker ps -a
```

Check container health

```
docker inspect mailserver | grep -A 10 Health
```

Email Account Management

Create Email Accounts

Create a new email account (will prompt for password)

```
docker exec -it mailserver setup email add user@home.conorbriggs.com.au
```

Create account with password in command

```
docker exec -it mailserver setup email add user@home.conorbriggs.com.au password123
```

Create account with quota (e.g., 500MB)

```
docker exec -it mailserver setup email add user@home.conorbriggs.com.au password123 500M
```

List Email Accounts

List all email accounts

```
docker exec mailserver setup email list
```

View the accounts file directly

```
docker exec mailserver cat /tmp/docker-mailserver/postfix-accounts.cf
```

Update/Change Passwords

Update password for existing account

```
docker exec -it mailserver setup email update user@home.conorbriggs.com.au new_password
```

Change password (alternative method - will prompt)

```
docker exec -it mailserver setup email update user@home.conorbriggs.com.au
```

Delete Email Accounts

Delete an email account

```
docker exec -it mailserver setup email del user@home.conorbriggs.com.au
```

Delete account and remove mailbox data

```
docker exec mailserver setup email del user@home.conorbriggs.com.au
rm -rf ./mail-data/home.conorbriggs.com.au/user
```

Alias Management

Create Aliases

Create an alias (forward emails from alias to recipient)

```
docker exec mailserver setup alias add alias@home.conorbriggs.com.au recipient@home.conorbriggs.com.au
```

Create alias with multiple recipients

```
docker exec mailserver setup alias add sales@home.conorbriggs.com.au
"user1@home.conorbriggs.com.au,user2@home.conorbriggs.com.au"
```

List Aliases

List all aliases

```
docker exec mailserver setup alias list
```

View aliases file

```
docker exec mailserver cat /tmp/docker-mailserver/postfix-virtual.cf
```

Delete Aliases

Delete an alias

```
docker exec mailserver setup alias del alias@home.conorbriggs.com.au recipient@home.conorbriggs.com.au
```

Quota Management

Set Quotas

Set quota for a user (e.g., 1GB)

```
docker exec mailserver setup quota set user@home.conorbriggs.com.au 1G
```

Set unlimited quota

```
docker exec mailserver setup quota set user@home.conorbriggs.com.au 0
```

Check Quotas

Check quota for specific user

```
docker exec mailserver setup quota get user@home.conorbriggs.com.au
```

List all quotas

```
docker exec mailserver setup quota list
```

Check quota usage

```
docker exec mailserver doveadm quota get -u user@home.conorbriggs.com.au
```

Delete Quotas

Remove quota (sets to default)

```
docker exec mailserver setup quota del user@home.conorbriggs.com.au
```

DKIM (Email Signing)

Generate DKIM Keys

Generate DKIM key for domain

```
docker exec mailserver setup config dkim
```

Generate for specific domain

```
docker exec mailserver setup config dkim domain home.conorbriggs.com.au
```

Generate with custom key size

```
docker exec mailserver setup config dkim keysize 2048
```

View DKIM Public Key

Show DKIM DNS record

```
docker exec mailserver setup config dkim help
```

View the public key directly

```
docker exec mailserver cat /tmp/docker-mailserver/opendkim/keys/home.conorbriggs.com.au/mail.txt
```

Fail2Ban (Security)

Fail2Ban Status

Check fail2ban status

```
docker exec mailserver setup fail2ban status
```

Check banned IPs

```
docker exec mailserver setup fail2ban
```

Unban an IP address

```
docker exec mailserver setup fail2ban unban <IP_ADDRESS>
```

Ban an IP address

```
docker exec mailserver setup fail2ban ban <IP_ADDRESS>
```

Debugging & Diagnostics

Service Status

Check all listening ports

```
docker exec mailserver ss -tlnp
```

Check specific service status

```
docker exec mailserver supervisorctl status
```

Check Postfix status

```
docker exec mailserver postfix status
```

Check Dovecot status

```
docker exec mailserver doveadm service status
```

Mail Queue

View mail queue

```
docker exec mailserver postqueue -p
```

Flush mail queue (retry sending)

```
docker exec mailserver postqueue -f
```

Delete all queued mail

```
docker exec mailserver postsuper -d ALL
```

Delete specific message from queue

```
docker exec mailserver postsuper -d <QUEUE_ID>
```

Logs

View mail logs

```
docker exec mailserver tail -f /var/log/mail/mail.log
```

View mail errors

```
docker exec mailserver tail -f /var/log/mail/mail.err
```

View specific log files

```
docker exec mailserver ls -la /var/log/mail/
```

Search logs for specific email

```
docker exec mailserver grep "user@domain.com" /var/log/mail/mail.log
```

Test Email Delivery

Test SMTP connection

```
docker exec mailserver nc -zv localhost 25
```

Send test email from command line

```
echo "Test email body" | docker exec -i mailserver sendmail test@home.conorbriggs.com.au
```

Test with swaks (if installed)

```
docker exec mailserver swaks --to user@home.conorbriggs.com.au --from test@home.conorbriggs.com.au
```

Connection Testing

Test IMAP connection

```
docker exec mailserver nc -zv localhost 143
docker exec mailserver nc -zv localhost 993
```

Test SMTP connection

```
docker exec mailserver nc -zv localhost 25
docker exec mailserver nc -zv localhost 587
docker exec mailserver nc -zv localhost 465
```

Check TLS/SSL certificates

```
docker exec mailserver openssl s_client -connect localhost:993 -showcerts
docker exec mailserver openssl s_client -connect localhost:465 -showcerts
```

Configuration Management

Reload Configuration

Reload postfix configuration

```
docker exec mailserver postfix reload
```

Reload dovecot configuration

```
docker exec mailserver doveadm reload
```

Restart all services

```
docker compose restart
```

View Configuration

View postfix configuration

```
docker exec mailserver postconf
```

View dovecot configuration

```
docker exec mailserver doveconf
```

View specific postfix setting

```
docker exec mailserver postconf | grep smtp_tls
```

Check all environment variables

```
docker exec mailserver env | grep -E '(SMTP|IMAP|SSL|TLS)'
```

Backup Configuration

Backup all mail data

```
tar -czf mailserver-backup-$(date +%Y%m%d).tar.gz ./mail-data ./mail-state ./config
```

Backup just configuration

```
tar -czf mailserver-config-$(date +%Y%m%d).tar.gz ./config
```

Backup specific user's mailbox

```
tar -czf user-backup-$(date +%Y%m%d).tar.gz ./mail-data/home.conorbriggs.com.au/user
```

Database/User Management

User Database

List all users in Dovecot

```
docker exec mailserver doveadm user '*'
```

Check if user exists

```
docker exec mailserver doveadm user user@home.conorbriggs.com.au
```

View user's mailbox location

```
docker exec mailserver doveadm mailbox status -u user@home.conorbriggs.com.au all '*'
```

Mailbox Management

List mailboxes for user

```
docker exec mailserver doveadm mailbox list -u user@home.conorbriggs.com.au
```

Create mailbox for user

```
docker exec mailserver doveadm mailbox create -u user@home.conorbriggs.com.au Folder.Name
```

Delete mailbox

```
docker exec mailserver doveadm mailbox delete -u user@home.conorbriggs.com.au Folder.Name
```

Rebuild mailbox index

```
docker exec mailserver doveadm force-resync -u user@home.conorbriggs.com.au INBOX
```

Performance & Monitoring

Check Resource Usage

Check container stats

```
docker stats mailserver
```

Check disk usage

```
docker exec mailserver df -h
```

Check memory usage

```
docker exec mailserver free -h
```

Check mail directory size

```
du -sh ./mail-data/*
```

Connection Monitoring

Show active connections

```
docker exec mailserver ss -tn | grep -E ':(25|587|465|143|993)'
```

Count connections by port

```
docker exec mailserver ss -tn | grep -E ':(25|587|465|143|993)' | wc -l
```

Show who's connected to IMAP

```
docker exec mailserver doveadm who
```

SSL/TLS Certificate Management

Check Certificates

Check SSL certificate expiry

```
docker exec mailserver openssl x509 -in /etc/letsencrypt/live/mx.home.conorbriggs.com.au/fullchain.pem -noout -dates
```

View certificate details

```
docker exec mailserver openssl x509 -in /etc/letsencrypt/live/mx.home.conorbriggs.com.au/fullchain.pem -noout -text
```

Test SSL/TLS for SMTP

```
openssl s_client -connect mx.home.conorbriggs.com.au:465 -showcerts
```

Test STARTTLS for SMTP

```
openssl s_client -connect mx.home.conorbriggs.com.au:587 -starttls smtp
```

Troubleshooting

Common Issues

Check if services are running

```
docker exec mailserver supervisorctl status
```

Restart specific service

```
docker exec mailserver supervisorctl restart postfix
docker exec mailserver supervisorctl restart dovecot
```

Check for permission issues

```
docker exec mailserver ls -la /var/mail/
docker exec mailserver ls -la /tmp/docker-mailserver/
```

Verify DNS records

```
dig mx home.conorbriggs.com.au
dig txt _dmarc.home.conorbriggs.com.au
dig txt mail._domainkey.home.conorbriggs.com.au
```

Test email authentication

```
docker exec mailserver opendkim-testkey -d home.conorbriggs.com.au -s mail
```

Reset and Clean Up

Remove all mail data (WARNING: deletes all emails)

```
docker compose down
rm -rf ./mail-data/*
rm -rf ./mail-state/*
docker compose up -d
```

Clear logs

```
docker exec mailserver truncate -s 0 /var/log/mail/mail.log
```

Rebuild entire container

```
docker compose down
docker compose pull
docker compose up -d --force-recreate
```

Quick Reference

Setup Script Help

Show all setup commands

```
docker exec mailserver setup help
```

Help for specific command

```
docker exec mailserver setup email help
docker exec mailserver setup alias help
docker exec mailserver setup config help
```

File Locations Inside Container

```
/tmp/docker-mailserver/         - Configuration files (mapped to ./config/)
/var/mail/                      - Mail data (mapped to ./mail-data/)
/var/mail-state/                - State files (mapped to ./mail-state/)
/var/log/mail/                  - Mail logs (mapped to ./mail-logs/)
/etc/letsencrypt/               - SSL certificates (read-only)
/etc/postfix/                   - Postfix configuration
/etc/dovecot/                   - Dovecot configuration
```

Important Configuration Files

```
./config/postfix-accounts.cf      - Email accounts
./config/postfix-virtual.cf       - Aliases
./config/dovecot-quotas.cf        - User quotas
./config/opendkim/                - DKIM keys
```

Advanced Operations

Database Operations

Export all accounts

```
docker exec mailserver cat /tmp/docker-mailserver/postfix-accounts.cf > accounts-backup.txt
```

Import accounts

```
cat accounts-backup.txt | docker exec -i mailserver tee /tmp/docker-mailserver/postfix-accounts.cf
docker compose restart
```

Verify account database

```
docker exec mailserver postmap -q user@home.conorbriggs.com.au /tmp/docker-mailserver/postfix-accounts.cf
```

Custom Scripts

Run custom maintenance script

```
docker exec mailserver /bin/bash -c "your-script-here"
```

Execute interactive shell

```
docker exec -it mailserver /bin/bash
```

Monitoring & Alerts

Set Up Monitoring

Watch logs in real-time

```
docker compose logs -f --tail=100
```

Monitor for failed logins

```
docker exec mailserver tail -f /var/log/mail/mail.log | grep "authentication failed"
```

Monitor mail queue size

```
watch -n 60 'docker exec mailserver postqueue -p | tail -1'
```

Check for errors

```
docker exec mailserver grep -i error /var/log/mail/mail.log | tail -20
```

Common Workflows

Adding a New User

1. Create account

```
docker exec -it mailserver setup email add newuser@home.conorbriggs.com.au
```

2. Set quota (optional)

```
docker exec mailserver setup quota set newuser@home.conorbriggs.com.au 2G
```

3. Verify account created

```
docker exec mailserver setup email list
```

4. Test login: Use an email client to connect via IMAP (993) or SMTP (587)

Migrating Mail

1. Backup old server

```
tar -czf old-mailserver-backup.tar.gz ./mail-data
```

2. Copy to new server

```
scp old-mailserver-backup.tar.gz newserver:/path/to/mailserver/
```

3. Extract on new server

```
tar -xzf old-mailserver-backup.tar.gz
```

4. Fix permissions

```
chown -R 5000:5000 ./mail-data
```

5. Restart mailserver

```
docker compose restart
```

Security Hardening

1. Enable Fail2Ban (in .env file)

```
ENABLE_FAIL2BAN=1
```

2. Check banned IPs regularly

```
docker exec mailserver fail2ban-client status postfix-sasl
```

3. Monitor authentication attempts

```
docker exec mailserver grep "authentication failed" /var/log/mail/mail.log
```

4. Review SSL/TLS settings

```
docker exec mailserver postconf | grep tls
```

Tips & Best Practices

1. **Always backup before major changes:** `tar -czf backup.tar.gz ./mail-data ./config`
2. **Test email flow after changes:** Send test emails in/out
3. **Monitor disk space:** Check `df -h` regularly
4. **Keep certificates updated:** Let's Encrypt certs expire every 90 days
5. **Review logs periodically:** Look for authentication failures or delivery issues
6. **Set appropriate quotas:** Prevent users from filling up disk
7. **Use strong passwords:** Minimum 12 characters for email accounts
8. **Enable DKIM/SPF/DMARC:** Improves deliverability
9. **Regular updates:** `docker compose pull && docker compose up -d`
10. **Document your changes:** Keep notes on custom configurations

Software & packages

Software & packages

Image manipulation

Compression

Install `sudo apt-get install imagemagick`

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

note: this will overwrite images

Software & packages

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`

Software & packages

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
```

Software & packages

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`

Software & packages

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

Software & packages

Screen

Screen lets you make virtual terminals with

```
screen
```

New session with name

```
screen -S session_name
```

Software & packages

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.

Software & packages

smb / samba

Samba Setup on Linux (Home Server)

A practical guide to installing and configuring Samba for a home network, covering a shared folder setup suitable for a home wiki or general file server.

---

1. Install Samba

```
sudo apt update && sudo apt install samba samba-common-bin -y
```

Verify it's running:

```
sudo systemctl status smbd nmbd
```

Both should be active. If not:

```
sudo systemctl enable smbd nmbd --now
```

---

2. Create the Share Directory

```
sudo mkdir -p /srv/samba/wiki
sudo chown -R conor:conor /srv/samba/wiki
sudo chmod 755 /srv/samba/wiki
```

Adjust owner to your actual Linux username.

3. Configure Samba

Back up the default config first:

```
sudo cp /etc/samba/smb.conf /etc/samba/smb.conf.bak
```

Edit the config:

```
sudo nano /etc/samba/smb.conf
```

Global section — replace or update:

```
[global]
   workgroup = WORKGROUP
   server string = Home Server
   server role = standalone server
   log file = /var/log/samba/log.%m
   max log size = 1000
   logging = file
   panic action = /usr/share/samba/panic-action %d

   # Security
   security = user
   map to guest = never
   encrypt passwords = yes

   # Performance
   socket options = TCP_NODELAY IPTOS_LOWDELAY
   read raw = yes
   write raw = yes
   use sendfile = yes
```

Share definition — add to the bottom:

```
[wiki]
   comment = Home Wiki
   path = /srv/samba/wiki
   browseable = yes
   read only = no
   valid users = conor
   create mask = 0664
   directory mask = 0775
   force group = conor
```

---

4. Create Samba User

Samba uses its own password store separate from Linux system passwords. The Linux user must already exist.

```
sudo smbpasswd -a conor
```

You'll be prompted to set a Samba-specific password. Enable the user:

```
sudo smbpasswd -e conor
```

---

5. Validate Config and Restart

Test the config for syntax errors:

```
testparm
```

If clean:

```
sudo systemctl restart smbd nmbd
```

---

6. Firewall

If UFW is active:

```
sudo ufw allow samba
```

This opens ports 137, 138 (UDP) and 139, 445 (TCP).

---

7. Connect From Clients

Linux (Files / Nautilus):

```
smb://SERVER_IP/wiki
```

Or mount via CLI:

```
sudo mount -t cifs //SERVER_IP/wiki /mnt/wiki -o username=conor,password=yourpass,uid=1000,gid=1000
```

Permanent via `/etc/fstab`:

```
//SERVER_IP/wiki  /mnt/wiki  cifs  credentials=/home/conor/.smbcredentials,uid=1000,gid=1000,_netdev,x-systemd.automount  0  0
```

`/home/conor/.smbcredentials`:

```
username=conor
password=yourpass
```

```
chmod 600 /home/conor/.smbcredentials
```

macOS:

Finder → Go → Connect to Server → `smb://SERVER_IP/wiki`

Windows:

`\\SERVER_IP\wiki` in Explorer address bar, or map as a network drive.

---

8. Tailscale Consideration

Since you're on Tailscale, use the Tailscale IP (`100.x.x.x`) instead of the LAN IP for consistent access across all your machines regardless of which network you're on. This also avoids exposing Samba to the public internet — Samba should **never** be exposed publicly, it has a long CVE history (EternalBlue, etc.).

Bind Samba only to your LAN + Tailscale interfaces to be safe:

```
[global]
   interfaces = lo tailscale0 eth0
   bind interfaces only = yes
```

Replace `eth0` with your actual LAN interface (`ip a` to check).

---

9. Verify the Share is Visible

From the server itself:

```
smbclient -L localhost -U conor
```

From another machine:

```
smbclient -L //SERVER_IP -U conor
```

You should see `wiki` listed under shares.

---

10. Troubleshooting

Symptom	Likely cause	Fix
"Permission denied" on connect	Wrong Samba password or user not enabled	`smbpasswd -e username`
Share not visible	`browseable = no` or firewall	Check config + `ufw status`
Can connect but can't write	Directory permissions	`chmod 775 /srv/samba/wiki`
Works on LAN, not Tailscale	Interfaces binding	Add `tailscale0` to `interfaces`

Symptom	Likely cause	Fix
`testparm` errors	Config syntax	Read the output carefully, it's specific

---

Notes

- Samba passwords and Linux system passwords are **independent** — changing one doesn't change the other.
- For a multi-user setup, create a dedicated group (e.g., `sambashare`) and use `valid users = @sambashare` in the share definition.
- If you're running this on the same machine as your wiki app (e.g., WikiJS, Obsidian vault server), Samba is a good way to access the underlying vault files directly from other machines for editing.

Software & packages

autofs

autofs Setup and Configuration Guide

`autofs` mounts filesystems on-demand when accessed and unmounts them after a configurable idle timeout. This makes it significantly more robust than static `/etc/fstab` mounts for network shares — a missing or offline host won't hang your boot, and stale mounts are cleaned up automatically.

---

1. Install

```
sudo apt update && sudo apt install autofs sshfs -y
```

For NFS shares also install:

```
sudo apt install nfs-common -y
```

---

2. How autofs Works

autofs has two config layers:

File	Purpose
`/etc/auto.master`	Master map — defines mount point directories and which map file handles them
`/etc/auto.<name>`	Detail map — defines individual mounts under that directory

When you access `/mnt/remote/myshare`, autofs intercepts it, reads the map, and mounts the share on the fly. After the timeout with no activity, it unmounts automatically.

---

3. Master Map — `/etc/auto.master`

Each line in `auto.master` follows this format:

```
<base_mount_dir>  <map_file>  [options]
```

Example:

```
/mnt/remote  /etc/auto.sshfs  --timeout=60,--ghost
/mnt/nfs     /etc/auto.nfs    --timeout=120,--ghost
```

Key options:

Option	Effect
`--timeout=N`	Unmount after N seconds of inactivity
`--ghost`	Creates empty placeholder directories so the mount point is visible even when unmounted. Without this, `ls /mnt/remote` shows nothing until a mount is active
`--verbose`	Useful during setup/debugging

Important: auto.master entry placement

The default `/etc/auto.master` contains a `+auto.master` line which is a legacy NIS include directive. Your custom entries must go **above** this line, otherwise they may be ignored on some systems. The `+auto.master` line is safe to remove entirely on a standalone home server with no NIS infrastructure.

Clean minimal auto.master:

```
+dir:/etc/auto.master.d

/mnt/remote  /etc/auto.sshfs  --timeout=60,--ghost
```

The `+dir:/etc/auto.master.d` line is worth keeping — it allows dropping extra `.autofs` map files into that directory without editing auto.master directly.

4. Detail Map Files

SSHFS (SSH/SFTP) — Modern Syntax

> **Important:** Many guides online show legacy syntax using `fstype=fuse` and `:sshfs#user@host` format. This does **not** work on modern Linux kernels. Always use `fstype=fuse.sshfs` with plain `user@host:/path` syntax.

Create `/etc/auto.sshfs`:

```
<mount_name>  -fstype=fuse.sshfs,rw,allow_other,IdentityFile=/home/conor/.ssh/id_ed25519,uid=1000,gid=1000,StrictHostKeyChecking=no,ServerAliveInterval=15,ServerAliveCountMax=3  conor@100.x.x.x:/home/conor
```

Breakdown:

Part	Meaning
`<mount_name>`	The subdirectory created under the base — e.g. `farmdesktop` becomes `/mnt/remote/farmdesktop`
`-fstype=fuse.sshfs`	Correct modern fstype for SSHFS mounts
`rw`	Read/write
`allow_other`	Allows users other than root to access the mount
`IdentityFile=`	Path to SSH private key — **must use key auth, not password**
`uid=1000,gid=1000`	Map remote files to your local user. Check yours with `id conor`
`StrictHostKeyChecking=no`	Required because autofs runs as root which has an empty known_hosts — without this the mount hangs waiting for an interactive host key confirmation that never comes
`ServerAliveInterval=15`	Send keepalive every 15s to prevent silent disconnects
`ServerAliveCountMax=3`	Drop connection after 3 missed keepalives

Full working example:

```
farm-server  -fstype=fuse.sshfs,rw,allow_other,IdentityFile=/home/conor/.ssh/cbcore-key,uid=1000,gid=1000,StrictHostKeyChecking=no,ServerAliveInterval=15,ServerAliveCountMax=3 conor@100.64.0.12:/home/conor
```

Access via: `/mnt/remote/farm-server`

Why legacy syntax fails

Old guides show:

```
farmdesktop  -fstype=fuse,rw,allow_other,...  :sshfs#conor@100.64.0.5:/home/conor
```

This tells autofs to call `mount -t fuse` with `sshfs#` as the device — an approach that relied on the generic FUSE kernel module handling the mount. Modern kernels expect `mount -t fuse.sshfs` with a plain remote path. Using the old syntax results in `signal 22 (SIGABRT)` and a failed mount with no useful error message.

NFS

Create `/etc/auto.nfs`:

```
nas  -fstype=nfs4,rw,soft,intr  100.64.0.10:/exports/data
```

Option	Meaning
`nfs4`	Use NFSv4 (preferred)
`soft`	Return error on timeout instead of hanging indefinitely
`intr`	Allow interrupting a hung NFS call with Ctrl+C

Samba / CIFS

Create `/etc/auto.smb`:

```
wiki  -fstype=cifs,rw,credentials=/home/conor/.smbcredentials,uid=1000,gid=1000,iocharset=utf8 ://100.64.0.10/wiki
```

`/home/conor/.smbcredentials`:

```
username=conor
password=yourpassword
```

```
chmod 600 /home/conor/.smbcredentials
```

---

5. FUSE Prerequisite for SSHFS

`allow_other` requires this line to be uncommented in `/etc/fuse.conf`:

```
sudo nano /etc/fuse.conf
```

Uncomment:

```
user_allow_other
```

Without this, SSHFS mounts via autofs (which runs as root) will be inaccessible to your user account.

---

6. SSH Key Setup

autofs runs as root, so two things must be true:

1. The key file must be readable by root
2. Root must have the remote host in its known_hosts — **or** `StrictHostKeyChecking=no` must be set

Seed root's known_hosts before first use:

```
sudo ssh -i /home/conor/.ssh/your-key conor@100.x.x.x echo "ok"
# Type "yes" when prompted to accept the host key
```

Alternatively, use `StrictHostKeyChecking=no` in the map file options (acceptable on a private Tailscale network).

Option A — use your existing user key (works if permissions allow root to read it):

```
IdentityFile=/home/conor/.ssh/id_ed25519
```

Option B — create a dedicated root-owned key for autofs:

```
sudo ssh-keygen -t ed25519 -f /root/.ssh/autofs_id_ed25519 -N ""
sudo ssh-copy-id -i /root/.ssh/autofs_id_ed25519.pub conor@100.x.x.x
```

Then use `IdentityFile=/root/.ssh/autofs_id_ed25519` in the map file. Cleaner — keeps autofs auth separate from your interactive SSH key.

---

7. Enable and Start autofs

```
sudo systemctl enable autofs --now
```

Reload after map file changes:

```
sudo systemctl reload autofs
```

Full restart required after auto.master changes:

```
sudo systemctl restart autofs
```

---

8. Testing

Trigger a mount by accessing the path directly:

```
ls /mnt/remote/farm-server
```

Check what's currently mounted:

```
mount | grep autofs
```

Verify autofs has loaded your maps correctly:

```
sudo automount --dumpmaps
```

Check status and logs:

```
sudo systemctl status autofs
sudo journalctl -u autofs -f
```

Force unmount manually:

```
sudo umount /mnt/remote/farm-server
```

---

9. Debugging a Failed Mount

If the mount silently fails, run autofs in foreground debug mode:

```
sudo systemctl stop autofs
sudo automount -f --verbose --debug 2>&1 | tee /tmp/autofs-debug.log &
sleep 2
ls /mnt/remote/farm-server
sleep 2
cat /tmp/autofs-debug.log
```

Then kill the debug instance and restart the service:

```
sudo pkill -9 -f automount
sudo rm -f /run/autofs.pid
sudo systemctl start autofs
```

If the mount command itself hangs, test it manually first:

```
# Test SSH connectivity as root
sudo ssh -i /path/to/key user@host echo "ok"

# Test sshfs as your normal user
mkdir /tmp/testmount
sshfs -o IdentityFile=/path/to/key user@host:/remote/path /tmp/testmount
ls /tmp/testmount
```

If sshfs works as your user but hangs as root, the issue is root's SSH environment (known_hosts, key permissions). Add `StrictHostKeyChecking=no` to the map options and ensure the key is readable by root.

---

10. Troubleshooting

Symptom	Likely cause	Fix
`ls` on mount point returns "No such file or directory"	autofs not reading map, or entry placement in auto.master	Run `sudo automount --dumpmaps` to verify map is loaded; check entry is above `+auto.master`
Mount hangs on access	Root's known_hosts empty — SSH waiting for interactive host key confirmation	Add `StrictHostKeyChecking=no` to map options, or pre-seed with `sudo ssh user@host echo ok`
`signal 22` / SIGABRT in debug output	Legacy `fstype=fuse` + `sshfs#` syntax	Change to `fstype=fuse.sshfs` and plain `user@host:/path`
"Permission denied" accessing mounted path	`allow_other` not set or `user_allow_other` not in fuse.conf	Check `/etc/fuse.conf`
Mount works as root but not as user	`uid=` / `gid=` not set in map file	Add `uid=1000,gid=1000` (check with `id username`)
autofs fails to start after manual debug session	Stale PID file from debug automount process	`sudo pkill -9 -f automount && sudo rm -f /run/autofs.pid`
Changes to map file have no effect	autofs cached old config	`sudo systemctl reload autofs`
`ls /mnt/remote` shows nothing despite ghost mode	Ghost mode requires autofs to be fully running and map parsed	Run `sudo automount --dumpmaps` to confirm map loaded

---

11. Full Example: Tailscale Homelab

`/etc/auto.master`:

```
+dir:/etc/auto.master.d

/mnt/remote  /etc/auto.sshfs  --timeout=60,--ghost
```

`/etc/auto.sshfs`:

```
farm-pi      -fstype=fuse.sshfs,rw,allow_other,IdentityFile=/root/.ssh/autofs_id_ed25519,uid=1000,gid=1000,StrictHostKeyChecking=no,ServerAliveInterval=15,ServerAliveCountMax=3  conor@100.64.0.2:/home/conor
farmdesktop  -
```

```
fstype=fuse.sshfs,rw,allow_other,IdentityFile=/root/.ssh/autofs_id_ed25519,uid=1000,gid=1000,StrictHostKeyChecking=no,ServerAliveInterval=15,ServerAliveCountMax=3  conor@100.64.0.3:/home/conor
homeserver  -
fstype=fuse.sshfs,rw,allow_other,IdentityFile=/root/.ssh/autofs_id_ed25519,uid=1000,gid=1000,StrictHostKeyChecking=no,ServerAliveInterval=15,ServerAliveCountMax=3  conor@100.64.0.4:/home/conor
```

Access:

```
/mnt/remote/farm-pi/
/mnt/remote/farmdesktop/
/mnt/remote/homeserver/
```

All mount on first access, unmount after 60 seconds idle. If a farm machine is offline, accessing its path returns an error immediately rather than hanging the system.

Notes

- autofs mounts do **not** appear in `df` or `mount` output unless currently active — this is normal.
- `/etc/auto.master` changes require a full `systemctl restart autofs`; map file changes only need `systemctl reload autofs`.
- For Ansible management across your fleet, the map files and master config are straightforward to template — a single playbook can deploy consistent autofs config to all five machines.

SMB

Editing SMB shares on linux

1 - Open the config file at `/etc/samba/smb.conf`

2 - Edit the SMB file

```
[ShareName]
   path = /path/to/share
   browseable = yes
   writable = yes
   valid users = user1, user2
```

3 - Save & restart smbd

```
sudo systemctl restart smbd
```

SMB Share options

`browsable = yes/no` Whether the share should be listed when clients request a list of shares.

`Guest ok = yes/no` guests can access without a password, set `read only` to `yes` to just be guest read only `guest ok = yes`

Editing SMB Users

Creating an SMB user `sudo smbpasswd -a new_username`
Listing all SMB users `pdbedit -L`
Changing an SMB user's password `sudo smbpasswd username`

Mounting & Unmounting

Get disk information

List information about all available block devices, including disks and their partitions.

```
lsblk -o NAME,MODEL,SIZE,ROTA,TYPE,MOUNTPOINTS
```

List information about disks and the partitions with UUID

```
lsblk -f
```

List disks with filesystems `sudo parted -l`

Permanent mounting

Edit the file at `/etc/fstab`

```
UUID=xxxxxxxxx-xxxxx-xxxxxxx /mnt/new_drive ext4 defaults X Y
```

X = Backup order - 0 is no backup

Y = Check filesystem at boot - 0 is no check

after editing fstab:

```
sudo mount -a
```

Temporary mounting

`sudo mount -t ext4 /dev/sdXY /mount/path`

X = Disk letter

Y = Disk partition

Symbolic Links

Regular SymLink

```
ln -s /path/to/target /path/to_where_new_shortcut_is_made
```

`-s` = softlink like a windows shortcut

Users & Permissions

Users and groups

List all groups and users in groups

```
getent group
```

Add a user to a group

```
sudo adduser username groupname
```

which is the same as

```
sudo usermod -aG groupname username
```

Create a new user

```
sudo useradd -options_here username
```

Options available:

- m = create home directory

Permissions

To see if a particular user can run a command

```
sudo -u USER COMMAND
```

To change a folders permission

```
sudo chmod -R u+rwx,g+rwx,o-rwx /path/to_folder
```

or change ownership

```
sudo chown -R NEW_USER:NEW_GROUP FOLDER_NAME
```

-R = recursive

Note: user or group can be left blank

Passwords

To change the curretly logged in user's password `passwd`

To change a different users password `sudo passwd USERNAME`

General Management & commands

Get the size of a folder

```
ncdu /path/to/view/
```

Strip all metadata from files

```
exiftool -all= -overwrite_original *
```

SSH

Generate a keypair

1 - Generate the key

```
ssh-keygen -t ed25519 -a 100 -f $HOME/.ssh/SSH_KEY_NAME
```

2 - Copy the key to the server

```
ssh-copy-id -i ~/.ssh/id_rsa.pub USERNAME@SERVER_IP
```

Fail2Ban

Setup

1. Update and install `apt update && apt upgrade -y && apt install -y fail2ban`
2. Start fail2ban and set it to auto start on boot `sudo systemctl start fail2ban && sudo systemctl enable fail2ban`
3. Make a local copy of the config file (why? because jail.conf get wiped after each update) `sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local`
4. Open jail.local and setup stuff

Actions

Unbanning IP `fail2ban-client set YOUR_JAIL_NAME_HERE unbanip IP_ADDRESS`

Banning IP `fail2ban-client JAIL_NAME banip IP_ADDRESS`

Changing max retry attempts

1. Open file located at `etc/fail2ban/jail.local`
2. Add `maxretry = x` under `[sshd]`
3. Restart fail to ban `sudo systemctl restart fail2ban`

wget

General Usage

`wget -Flags URL`

Full site mirror

```
wget -mkprE -np -nc -l inf -e robots=off --restrict-file-names=windows -D example.com --no-check-certificate http://example.com
```

Flags

- `--mirror / -m` Turns on all options to make a mirror copy of a site (`--recursive`, `--timestamping`, `--level=inf`, `--no-remove-listing`)
- `--convert-links / -k` Converts the links in the downloaded documents to make them suitable for offline viewing. This means that all the links will point to the local files instead of the original URLs.
- `adjust-extension / -E` Adjusts the file extension based on the MIME type. For example, if a file is served as HTML but doesn't have an .html extension, this flag will rename the file to include the .html extension
- `--page-requisites / -p` Downloads all the resources that a page requires to display properly, such as images, CSS, and JavaScript files.
- `--restrict-file-names=windows` Modifies filenames so that they are compatible with Windows. This avoids issues with characters that are not allowed in Windows filenames (e.g : or *). `--domains example.com`: - Restricts the download to the specified domain (example.com in this case). wget will only download files from this domain and no other domains.
- `--no-clobber / -nc` Prevents wget from overwriting existing files. If a file already exists, wget will skip the download of that file.
- `--no-check-certificate` Disables SSL/TLS certificate validation. This can be useful if the server's certificate is self-signed or otherwise not trusted by default.
- `-e robots=off` Ignores the robots.txt file on the server, allowing wget to download pages and resources that might otherwise be disallowed by the server's robots.txt file. Use this option responsibly.

- `--recursive / -r` Enables recursive downloading. This means wget will follow links found in the downloaded files and download those files as well. This is necessary for downloading entire websites or sections of websites.
- `--level=inf` Specifies the maximum recursion depth. Setting it to inf (infinity) means wget will follow links at any depth, effectively downloading the entire site as long as links are found.
- `--no-parent / -np` Prevents wget from following links outside the specified directory. This means it won't ascend to parent directories but will stay within the directory structure of the specified URL.

Bash

Edit shortcuts

Open `.bashrc` file and add

```
alias 1="command"
```

Auto run commands on terminal open

Open `.bashrc` file and add command to bottom file

Fix: no bash after ssh login

```
chsh -s /bin/bash
```

UFW

General Info

By default, UFW is set to deny all incoming connections and allow all outgoing connections. This means anyone trying to reach your server would not be able to connect, while any application within the server would be able to reach the outside world.

So **make sure to allow ssh before enabling UFW!!!** UFW requires you to add / deny ports in a certain order, if you want to allow a certain ip access a port but deny access by everyone else, the ip must be allowed first then deny all after

Setup

```
sudo apt update && sudo apt install -y ufw
```

Basic Management

Turning UFW on:

```
ufw enable
```

Turning UFW off:

```
ufw disable
```

Check if UFW is enabled or disabled

```
sudo ufw status
```

Allowing access from a particular IP to ANY port

```
sudo ufw allow from IP_ADDRESS
```

Allowing access from a partiular IP to A SPECIFIED port:

```
sudo ufw allow from IP_ADDRESS to any port PORT_NUMBER
```

Allowing any access to a specified port:

```
sudo ufw allow PORT_NUMBER
```

Denying access to a specific port

```
sudo ufw deny PORT_NUMBER
```

List rule numbers

```
sudo ufw status numbered
```

Swap

Clear swap file

1 - Disable Swap

```
sudo swapoff -a
```

2 - Re-enable swap

```
sudo swapon -a
```

View Swap usage

```
sudo smem -rs swap
```

Swap config

Update swappiness config

```
sudo bash -c "echo 'vm.swappiness = 1' >> /etc/sysctl.conf"
```

Update value immediately

```
sysctl vm.swappiness=1
```

About swap

Swappiness in Linux is a kernel parameter that controls how aggressively the system moves inactive pages from RAM to the swap space on disk. The swappiness value is represented as a percentage from 0 to 100, and it affects the balance between using physical RAM and swap space. Lowering swappiness can help improve performance in certain scenarios, particularly when you have a decent amount of physical memory available.

How Swappiness Works
Low swappiness values (0–30): The kernel avoids swapping as much as possible and will only swap when RAM is close to being fully utilized. This setting is ideal when performance is prioritized and there’s enough RAM.

High swappiness values (60–100): The kernel uses swap space more aggressively, even when RAM is available. This setting can be helpful on systems with limited RAM or on systems running many applications simultaneously.

The default value for swappiness on most Linux systems is 60.

Disk Stuff

List disks:

```
fdisk -l
```

OR

```
df -h
```

Scanning

1 - Check if SMART scans are enabled

```
smartctl -s on /dev/sdX
```

2 - Run a test

```
smartctl -t DURATION /dev/sdX
```

Duration is either "long" or "short"

3 - Check results

```
smartctl -a /dev/sdX
```

S.M.A.R.T scanning

Scan a disk

```
sudo smartctl -t long/short /dev/sdX
```

Check scan results

```
sudo smartctl -a /dev/sdX
```

Check scan progress

```
smartctl -c /dev/sdX
```

Auto updates

1 - Install and setup unattended-upgrades package

```
sudo apt update && sudo apt install unattended-upgrades
```

2 - Enable the auto updates

```
sudo dpkg-reconfigure --priority=low unattended-upgrades
```

3 - Edit the config to allow security updates

Open

```
sudo nano /etc/apt/apt.conf.d/50unattended-upgrades
```

Uncomment this line

```
        "${distro_id}:${distro_codename}-updates";
```

SSH Tunnel

```
ssh -L 127.0.0.1:3389:192.168.4.222:3389 -N conor@cbcore
```

-N = Don't execute command on remote server
-L = Specifies local port forwarding