Projects

- Close Lid, Uptime, Auth
- notifications and preload
- new things to do
- webshop
- Overleaf Setup
    - Setup Overleaf on Docker Compose with all Bibtex
- Zeitplan Praxisphase
- Dump

Close Lid, Uptime, Auth

Setup-Guide: Hyprlock, Authelia & Uptime Kuma

TEIL 1: HYPRLOCK AUTO-LOCK BEI LID CLOSE

Hyprland Lid-Event Handler

Methode 1: Mit Hyprland Event (empfohlen!)

Hyprland hat ein eingebautes Event-System!

Config erweitern:

```
nano ~/.config/hypr/hyprland.conf
```

Füge hinzu:

```
# Lid close event
bindl = , switch:on:Lid Switch, exec, hyprlock
```

Das war's!

Test:

- Laptop-Deckel zuklappen
- Sollte sofort sperren

Methode 2: Mit systemd-logind (Alternative)

Falls Methode 1 nicht funktioniert:

Script erstellen:

```
sudo nano /usr/local/bin/lock-on-lid.sh
```

Inhalt:

```
#!/bin/bash

# User und Display finden
USER=$(who | grep tty | awk '{print $1}' | head -1)
DISPLAY_NUM=$(who | grep "($USER)" | awk '{print $5}' | sed 's/[()]//g' | head -1)

if [ -z "$DISPLAY_NUM" ]; then
    DISPLAY_NUM=":0"
fi

# Als User ausführen
export DISPLAY="$DISPLAY_NUM"
sudo -u $USER WAYLAND_DISPLAY=wayland-1 hyprlock
```

Ausführbar machen:

```
sudo chmod +x /usr/local/bin/lock-on-lid.sh
```

Systemd-Service:

```
sudo nano /etc/systemd/system/lock-on-lid@.service
```

Inhalt:

```
[Unit]
Description=Lock screen on lid close
Before=sleep.target

[Service]
Type=oneshot
User=%i
ExecStart=/usr/local/bin/lock-on-lid.sh

[Install]
WantedBy=sleep.target
```

Aktivieren:

```
sudo systemctl enable lock-on-lid@denode.service
```

logind.conf anpassen:

```
sudo nano /etc/systemd/logind.conf
```

Ändere:

```
HandleLidSwitch=lock
HandleLidSwitchExternalPower=lock
HandleLidSwitchDocked=ignore
```

logind neu starten:

```
sudo systemctl restart systemd-logind
```

TEIL 2: AUTHELIA SSO INSTALLATION

Was ist Authelia?

Authelia ersetzt Basic Auth durch moderne Login-Seite mit:

- Schöne Login-UI
- 2FA (TOTP, U2F)
- SSO für alle Services
- Session-Management
- Passwort-Reset

Aussehen: Wie Google/GitHub Login - professionell!

Schritt 1: Verzeichnis erstellen

WICHTIG: Wir nutzen die HDD um Speicherplatz zu sparen!

```
# Auf dem Homeserver - auf der HDD!
sudo mkdir -p /mnt/hdd/authelia/{config,data,data/redis}
sudo chown -R denode:denode /mnt/hdd/authelia
cd /mnt/hdd/authelia
```

Schritt 2: Docker-Compose erstellen

```
nano docker-compose.yml
```

Inhalt:

```
version: '3.8'

services:
  authelia:
    image: authelia/authelia:latest
    container_name: authelia
    restart: unless-stopped
    environment:
      - TZ=Europe/Berlin
```

```
    volumes:
      - ./config:/config
      - ./data:/data
    ports:
      - "9091:9091"
    networks:
      - authelia

  redis:
    image: redis:7-alpine
    container_name: authelia-redis
    restart: unless-stopped
    volumes:
      - ./data/redis:/data
    networks:
      - authelia

networks:
  authelia:
    driver: bridge
```

Schritt 3: Authelia Config

```
nano config/configuration.yml
```

Komplette Config:

```
---
theme: dark
jwt_secret: CHANGE_THIS_TO_RANDOM_STRING_12345678901234567890123456789O

default_redirection_url: https://denode.eu

server:
  host: 0.0.0.0
  port: 9091
```

```
log:
  level: info

totp:
  issuer: denode.eu
  period: 30
  skew: 1

authentication_backend:
  file:
    path: /config/users_database.yml
    password:
      algorithm: argon2id
      iterations: 1
      salt_length: 16
      parallelism: 8
      memory: 64

access_control:
  default_policy: deny
  rules:
    - domain: "*.denode.eu"
      policy: one_factor

session:
  name: authelia_session
  domain: denode.eu
  expiration: 1h
  inactivity: 5m
  remember_me_duration: 1M

  redis:
    host: redis
    port: 6379

regulation:
  max_retries: 3
  find_time: 2m
  ban_time: 5m
```

```
storage:
  local:
    path: /data/db.sqlite3

notifier:
  disable_startup_check: true
  filesystem:
    filename: /data/notification.txt
```

WICHTIG: Ändere `jwt_secret` zu einem zufälligen String!

```
# Generiere Random String
openssl rand -hex 32
# Kopiere Output und ersetze jwt_secret
```

Schritt 4: Users erstellen

```
nano config/users_database.yml
```

Inhalt:

```
---
users:
  denode:
    displayname: "Denode"
    password: "$argon2id$v=19$m=65536,t=3,p=4$HASH_HIER"
    email: admin@denode.local
    groups:
      - admins
      - dev
```

Passwort hashen:

```
docker run --rm authelia/authelia:latest authelia crypto hash generate argon2 --password
'DeinPasswort'
```

Kopiere den Hash (komplette Zeile mit `$argon2id$...`) und ersetze in `users_database.yml`!

Schritt 5: Authelia starten

```
cd /mnt/hdd/authelia

docker compose up -d

# Logs ansehen
docker compose logs -f
```

Warte bis: "Authelia is now listening"

Test:

```
curl http://localhost:9091/api/health
# Sollte {"status":"UP"} zurückgeben
```

Schritt 6: Caddy auf Homeserver anpassen

```
sudo nano /etc/caddy/Caddyfile
```

Füge nach dem `homeserver.mora-stairs.ts.net` Block hinzu:

```
# Authelia
auth.denode.eu {
    reverse_proxy localhost:9091
}

http://:80 {
    @from_tunnel remote_ip 10.100.0.1

    # Authelia API
    @authelia host auth.denode.eu
    handle @authelia {
        handle @from_tunnel {
```

```
            reverse_proxy localhost:9091
        }
        handle {
            respond "Access denied" 403
        }
    }

    # ... rest bleibt gleich
}
```

Caddy neuladen:

```
sudo systemctl reload caddy
```

Schritt 7: DNS-Record setzen

Bei spaceship.com:

```
Type: A
Name: auth
Value: 46.224.8.110
TTL: 3600
```

Schritt 8: Caddy auf Hetzner mit Authelia

```
ssh hetzner
nano /etc/caddy/Caddyfile
```

Ändere z.B. den notes.denode.eu Block:

```
notes.denode.eu {
    # Forward Auth zu Authelia
    forward_auth auth.denode.eu {
```

```
        uri /api/verify?rd=https://auth.denode.eu
        copy_headers Remote-User Remote-Groups Remote-Name Remote-Email
    }

    encode gzip
    reverse_proxy 10.100.0.2:4567 {
        header_up Host {host}
        header_up X-Real-IP {remote_host}
        header_up X-Forwarded-For {remote_host}
        header_up X-Forwarded-Proto {scheme}
    }
}
```

Gleich für alle Services (denode.eu, overleaf.denode.eu, etc.):

```
denode.eu {
    forward_auth auth.denode.eu {
        uri /api/verify?rd=https://auth.denode.eu
        copy_headers Remote-User Remote-Groups Remote-Name Remote-Email
    }

    root * /var/www/denode
    file_server
    encode gzip
}

overleaf.denode.eu {
    forward_auth auth.denode.eu {
        uri /api/verify?rd=https://auth.denode.eu
        copy_headers Remote-User Remote-Groups Remote-Name Remote-Email
    }

    encode gzip
    reverse_proxy 10.100.0.2:4568 {
        header_up Host {host}
        header_up X-Real-IP {remote_host}
        header_up X-Forwarded-For {remote_host}
        header_up X-Forwarded-Proto {scheme}
    }
}
```

Caddy neuladen:

```
systemctl reload caddy
```

Schritt 9: Authelia testen!

1. **Öffne:** `https://notes.denode.eu`
2. **Du wirst weitergeleitet zu:** `https://auth.denode.eu`
3. **Login-Seite erscheint!**
4. **Einloggen:**
   - Username: `denode`
   - Passwort: DeinPasswort
5. **Nach Login:** Zurück zu BookStack!

2FA einrichten (optional):

1. Nach Login: Klick auf dein Profil
2. **Two-Factor Authentication**
3. QR-Code mit Authenticator-App scannen (z.B. Authy, Google Authenticator)

Authelia Features

Session-Management:

- Nach Login: 1 Stunde gültig
- "Remember Me": 1 Monat
- Auto-Logout bei Inaktivität: 5 Minuten

Sicherheit:

- 3 Fehlversuche → 5 Minuten Ban
- Passwort-Hashing mit Argon2id
- Optional: 2FA mit TOTP

Aussehen:

- Dark Mode (in Config: `theme: dark`)
- Responsive Design
- Moderne UI

Troubleshooting Authelia

Logs ansehen:

```
cd /mnt/hdd/authelia
docker compose logs authelia -f
```

Container neu starten:

```
docker compose restart authelia
```

Config testen:

```
docker exec authelia authelia validate-config /config/configuration.yml
```

Passwort zurücksetzen:

```
# Neuen Hash generieren
docker run --rm authelia/authelia:latest authelia crypto hash generate argon2 --password
'NeuesPasswort'

# In users_database.yml eintragen
nano config/users_database.yml

# Authelia neu starten
docker compose restart authelia
```

TEIL 3: UPTIME KUMA INSTALLATION

Was ist Uptime Kuma?

Uptime Kuma ist ein modernes Monitoring-Dashboard.

Features:

- Schöne Status-Seite
- Monitort HTTP, TCP, Ping, Docker, etc.
- Benachrichtigungen (Telegram!)
- Öffentliche Status-Page möglich
- Response-Time Graphen
- Uptime-Statistiken

Sieht aus wie: Professionelles Monitoring-Dashboard!

Schritt 1: Installation

WICHTIG: Auf HDD installieren!

```
# Auf dem Homeserver - auf der HDD!
sudo mkdir -p /mnt/hdd/uptime-kuma/data
sudo chown -R denode:denode /mnt/hdd/uptime-kuma
cd /mnt/hdd/uptime-kuma

nano docker-compose.yml
```

Inhalt:

```
version: '3.8'

services:
  uptime-kuma:
    image: louislam/uptime-kuma:latest
    container_name: uptime-kuma
    restart: unless-stopped
    volumes:
      - ./data:/app/data
    ports:
      - "3001:3001"
```

Starten:

```
docker compose up -d

# Logs
docker compose logs -f
```

Schritt 2: Caddy konfigurieren

Auf Homeserver:

```
sudo nano /etc/caddy/Caddyfile
```

Füge in `http://:80` Block hinzu:

```
@uptime host uptime.denode.eu
handle @uptime {
    handle @from_tunnel {
        reverse_proxy localhost:3001 {
            header_up X-Real-IP {remote_host}
            header_up X-Forwarded-For {remote_host}
            header_up X-Forwarded-Proto {scheme}
        }
    }
    handle {
        respond "Access denied" 403
    }
}
```

Auf Hetzner:

```
ssh hetzner
nano /etc/caddy/Caddyfile
```

Hinzufügen:

```
uptime.denode.eu {
    # Optional: Mit Authelia schützen
    forward_auth auth.denode.eu {
        uri /api/verify?rd=https://auth.denode.eu
```

```
        copy_headers Remote-User Remote-Groups Remote-Name Remote-Email
    }

    encode gzip
    reverse_proxy 10.100.0.2:3001 {
        header_up Host {host}
        header_up X-Real-IP {remote_host}
        header_up X-Forwarded-For {remote_host}
        header_up X-Forwarded-Proto {scheme}
    }
}
```

Beide Caddy neu laden:

```
# Homeserver
sudo systemctl reload caddy

# Hetzner
ssh hetzner "systemctl reload caddy"
```

Schritt 3: DNS-Record

Bei spaceship.com:

```
Type: A
Name: uptime
Value: 46.224.8.110
TTL: 3600
```

Schritt 4: Uptime Kuma Setup

1. **Öffne:** `https://uptime.denode.eu`
2. **Erster Login:** Account erstellen
   - Username: admin
   - Passwort: DeinSicheresPasswort
3. **Dashboard öffnet sich!**

Schritt 5: Monitors erstellen

Nextcloud Monitor

1. Klick **"Add New Monitor"**
2. **Monitor Type:** HTTP(s)
3. **Friendly Name:** Nextcloud
4. **URL:** `https://nextcloud.denode.eu`
5. **Heartbeat Interval:** 60 seconds
6. **Retries:** 3
7. **Method:** GET
8. **Accepted Status Codes:** 200-299, 302
9. **Save**

BookStack Monitor

- **Name:** BookStack
- **URL:** `https://notes.denode.eu`
- Rest gleich

Overleaf Monitor

- **Name:** Overleaf
- **URL:** `https://overleaf.denode.eu`

Landing-Page Monitor

- **Name:** Landing Page
- **URL:** `https://denode.eu`

Wireguard-Tunnel Monitor

1. **Monitor Type:** Ping
2. **Name:** Wireguard Tunnel
3. **Hostname:** `10.100.0.1` (Hetzner) oder `10.100.0.2` (Homeserver)
4. **Save**

Schritt 6: Telegram-Benachrichtigungen

1. **Settings** → **Notifications**
2. **Setup Notification** → **Telegram**
3. **Bot Token:** `8446514684:AAEthBJXkEVMw9kOVfrjdtpDuE5S3P9Heas`
4. **Chat ID:** `1916692273`
5. **Test** → sollte Nachricht in Telegram kommen!
6. **Save**

Für jeden Monitor aktivieren:

- Monitor öffnen → **Notifications** Tab
- Telegram aktivieren

Schritt 7: Status-Page erstellen (Optional)

Öffentliche Status-Seite für Freunde/Familie:

1. **Status Pages** (links im Menü)
2. **Add New Status Page**
3. **Title:** Denode Services
4. **Slug:** `public` (wird zu: uptime.denode.eu/status/public)
5. **Description:** "Status meiner Self-Hosted Services"
6. **Theme:** Dark
7. **Add a Group:**
   - Name: "Services"
   - Wähle alle Monitore
8. **Save**

Jetzt können andere `https://uptime.denode.eu/status/public` aufrufen und sehen den Status!

Uptime Kuma Features

Dashboard:

- Live-Status aller Services (□□□□)
- Durchschnittliche Response-Time
- Uptime Prozent (24h, 7d, 30d)
- Incident-History

Benachrichtigungen:

- Telegram
- Discord
- Email
- Slack
- Webhook
- ... und viele mehr!

Monitoring-Typen:

- HTTP(s)
- TCP Port
- Ping/ICMP
- DNS
- Docker Container
- Keyword (prüft ob Text auf Seite)

Troubleshooting Uptime Kuma

Container neu starten:

```
cd /mnt/hdd/uptime-kuma
docker compose restart
```

Logs:

```
docker compose logs -f
```

Daten zurücksetzen:

```
docker compose down
rm -rf data/
docker compose up -d
```

ZUSAMMENFASSUNG

Was du jetzt hast:

1. Hyprlock Auto-Lock

- Laptop sperrt automatisch beim Zuklappen
- Hyprland Event-Handler

2. Authelia SSO

- Moderne Login-Seite für alle Services
- Kein nerviges Basic-Auth mehr!
- Optional: 2FA
- Session-Management
- Sieht professionell aus!

3. Uptime Kuma

- Monitoring aller Services
- Schönes Dashboard
- Telegram-Benachrichtigungen
- Status-Page für andere

URLs:

- **Authelia:** https://auth.denode.eu
- **Uptime Kuma:** https://uptime.denode.eu
- **Status-Page:** https://uptime.denode.eu/status/public

Wichtige Befehle:

Authelia:

```
cd /mnt/hdd/authelia
docker compose ps
docker compose logs -f
docker compose restart
```

Uptime Kuma:

```
cd /mnt/hdd/uptime-kuma
docker compose ps
docker compose logs -f
```

Hyprlock testen:

```
hyprlock
```

Ports:

- Authelia: 9091
- Uptime Kuma: 3001

Backup:

In Backup-Script einfügen:

```
# Authelia Backup (liegt auf HDD!)
tar -czf "$BACKUP_DIR/authelia/authelia_${DATE}.tar.gz" -C /mnt/hdd authelia/

# Uptime Kuma Backup (liegt auf HDD!)
tar -czf "$BACKUP_DIR/uptime-kuma/uptime-kuma_${DATE}.tar.gz" -C /mnt/hdd uptime-kuma/
```

Backup-Verzeichnisse erstellen:

```
mkdir -p /mnt/hdd/backups/authelia
mkdir -p /mnt/hdd/backups/uptime-kuma
```

Landing-Page erweitern:

Füge neue Service-Cards hinzu:

```
<div class="service-card">
    <div class="service-icon">□□</div>
    <h3>Login</h3>
    <p>Single Sign-On für alle Services.</p>
    <a href="https://auth.denode.eu" class="service-link">Öffnen →</a>
</div>

<div class="service-card">
    <div class="service-icon">□□</div>
    <h3>Status</h3>
    <p>Monitoring & Uptime aller Services.</p>
    <a href="https://uptime.denode.eu" class="service-link">Öffnen →</a>
</div>
```

- □ Automatisches Laptop-Lock
- □ Moderne Login-Seite
- □ Professionelles Monitoring

notifications and preload

Notifications

```
yay -S swaync

mkdir ~/.config/swaync

# ~/.config/swaync/config.json
# ~/.config/swaync/style.css
```

in hyprland conf:

```
# Starte swaync
exec-once = swaync

# Keybinds
bind = $mainMod, N, exec, swaync-client -t -sw
bind = $mainMod SHIFT, N, exec, swaync-client -d -sw  # DND toggle
```

in config.json

```
{
  "$schema": "/etc/xdg/swaync/configSchema.json",
  "positionX": "right",
  "positionY": "top",
  "layer": "overlay",
  "control-center-layer": "top",
  "layer-shell": true,
  "cssPriority": "application",
  "control-center-margin-top": 10,
  "control-center-margin-bottom": 10,
  "control-center-margin-right": 10,
  "control-center-margin-left": 0,
  "notification-2fa-action": true,
  "notification-inline-replies": false,
```

```
"notification-icon-size": 64,
"notification-body-image-height": 100,
"notification-body-image-width": 200,
"timeout": 5,
"timeout-low": 3,
"timeout-critical": 0,
"fit-to-screen": true,
"control-center-width": 400,
"control-center-height": 600,
"notification-window-width": 400,
"keyboard-shortcuts": true,
"image-visibility": "when-available",
"transition-time": 200,
"hide-on-clear": false,
"hide-on-action": true,
"script-fail-notify": true,

"widgets": [
  "title",
  "dnd",
  "notifications"
],

"widget-config": {
  "title": {
    "text": "Benachrichtigungen",
    "clear-all-button": true,
    "button-text": "Alle löschen"
  },
  "dnd": {
    "text": "Nicht stören"
  },
  "label": {
    "max-lines": 5,
    "text": "Notification Center"
  },
  "mpris": {
    "image-size": 96,
    "image-radius": 12
  }
```

```
  }
}
```

in style.css

```
/* Catppuccin Mocha Farbschema - kannst du natürlich anpassen */
* {
  all: unset;
  font-family: "JetBrains Mono", monospace;
  font-size: 14px;
}

/* Notification Window */
.notification-row {
  outline: none;
  margin: 10px;
}

.notification {
  background: rgba(30, 30, 46, 0.95);
  border: 2px solid #89b4fa;
  border-radius: 10px;
  padding: 0;
  margin: 0;
}

.notification-content {
  background: transparent;
  padding: 15px;
  margin: 0;
}

.close-button {
  background: #f38ba8;
  color: #1e1e2e;
  border-radius: 8px;
  margin: 8px;
  padding: 4px 8px;
}
```

```
.close-button:hover {
  background: #eba0ac;
}

.notification-default-action {
  margin: 0;
  padding: 0;
}

/* Summary (Titel) */
.summary {
  color: #cdd6f4;
  font-weight: bold;
  font-size: 15px;
  margin-bottom: 5px;
}

/* Body (Text) */
.body {
  color: #bac2de;
  font-size: 13px;
}

/* Time */
.time {
  color: #6c7086;
  font-size: 11px;
  margin-top: 5px;
}

/* Urgency Levels */
.low {
  border: 2px solid #94e2d5;
}

.normal {
  border: 2px solid #89b4fa;
}
```

```
.critical {
  border: 2px solid #f38ba8;
  background: rgba(243, 139, 168, 0.1);
}

/* Control Center */
.control-center {
  background: rgba(30, 30, 46, 0.98);
  border: 2px solid #89b4fa;
  border-radius: 15px;
  padding: 10px;
}

.control-center-list {
  background: transparent;
}

/* Widget Titel */
.widget-title {
  color: #cdd6f4;
  font-size: 18px;
  font-weight: bold;
  margin: 10px;
}

.widget-title > button {
  background: #89b4fa;
  color: #1e1e2e;
  border-radius: 8px;
  padding: 6px 12px;
  font-size: 13px;
}

.widget-title > button:hover {
  background: #b4befe;
}

/* DND Toggle */
```

```
.widget-dnd {
  margin: 10px;
  padding: 10px;
  background: rgba(137, 180, 250, 0.1);
  border-radius: 10px;
}

.widget-dnd > switch {
  background: #45475a;
  border-radius: 20px;
  padding: 2px;
}

.widget-dnd > switch:checked {
  background: #89b4fa;
}

.widget-dnd > switch slider {
  background: #cdd6f4;
  border-radius: 50%;
}

/* Notification in Control Center */
.control-center .notification {
  margin: 5px 0;
}

/* Scrollbar */
.control-center-list > scrollbar {
  background: transparent;
  width: 8px;
}

.control-center-list > scrollbar > slider {
  background: #585b70;
  border-radius: 8px;
}

.control-center-list > scrollbar > slider:hover {
```

```
  background: #6c7086;
}

/* Leeres Center */
.blank-window {
  background: transparent;
}

/* Image */
.notification-icon {
  border-radius: 8px;
  margin-right: 10px;
}

.body-image {
  border-radius: 8px;
  margin-top: 8px;
}
```

Preload

```
yay -S preload
sudo systemctl enable --now preload
```

```
yay -S ananicy-cpp
sudo systemctl enable --now ananicy-cpp
```

```
yay -S profile-sync-daemon
systemctl --user enable --now psd
psd p  # Zeigt Config
```

```
sudo pacman -S earlyoom
sudo systemctl enable --now earlyoom
```

cat mit syntaxhighlight sudo pacman -S bat bat datei.py # statt cat alias cat='bat'

new things to do

1. Glance

```
https://github.com/glanceapp/glance
```

2. StatPing.ng

```
https://github.com/statping-ng/statping-ng
```

3. DumpDo

```
https://github.com/DumbWareio/DumbDo
```

4. Dump Pad

```
https://github.com/DumbWareio/Dumbpad
```

5. Excalidraw

```
https://docs.excalidraw.com/docs/introduction/development
```

6. FusionAuth

```
https://fusionauth.io/
```

7. Ghost

```
https://docs.ghost.org/hosting
```

8. LocalSend

```
https://localsend.org/de
```

9. Wiki.js

```
https://js.wiki/
```

10. Appsmith

```
https://www.appsmith.com/
```

11. Monica

```
https://github.com/monicahq/monica
```

12. XWiki

```
https://www.xwiki.org/xwiki/bin/view/Main/WebHome
```

13. Snapdrop

```
https://github.com/SnapDrop/snapdrop
```

14. Plane

```
https://plane.so/self-hosted
```

15. Webhook

```
https://github.com/adnanh/webhook
```

webshop

1. Verzeichnisse anlegen:

```
# Auf der SSD
sudo mkdir -p /opt/shop

# Auf der HDD
sudo mkdir -p /mnt/hdd/shop/{uploads,database,backups}
sudo chown -R 33:33 /mnt/hdd/shop/uploads
```

2.docker-compose.yml erstellen: cd /opt/shop nano docker-compose.yml

```
version: '3.8'

services:
  db:
    image: mariadb:10.11-alpine
    restart: unless-stopped
    environment:
      MYSQL_ROOT_PASSWORD: PASSWORT_1
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wpuser
      MYSQL_PASSWORD: PASSWORT_2
    volumes:
      - /mnt/hdd/shop/database:/var/lib/mysql
    networks:
      - shop-internal

  wordpress:
    image: wordpress:6.4-fpm-alpine
    restart: unless-stopped
    depends_on:
      - db
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: wpuser
```

```
      WORDPRESS_DB_PASSWORD: PASSWORT_2
      WORDPRESS_DB_NAME: wordpress
      WORDPRESS_CONFIG_EXTRA: |
        define('WP_HOME', 'https://test.denode.eu');
        define('WP_SITEURL', 'https://test.denode.eu');
    volumes:
      - wordpress-core:/var/www/html
      - /mnt/hdd/shop/uploads:/var/www/html/wp-content/uploads
    networks:
      - shop-internal

  nginx:
    image: nginx:alpine
    restart: unless-stopped
    depends_on:
      - wordpress
    ports:
      - "3050:80"
    volumes:
      - wordpress-core:/var/www/html:ro
      - /mnt/hdd/shop/uploads:/var/www/html/wp-content/uploads:ro
      - ./nginx.conf:/etc/nginx/nginx.conf:ro
    networks:
      - shop-internal

volumes:
  wordpress-core:

networks:
  shop-internal:
```

3. nginx.conf erstellen:

```
user nginx;
worker_processes auto;

events {
    worker_connections 1024;
}
```

```
http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    sendfile on;
    keepalive_timeout 65;
    client_max_body_size 64M;

    upstream php {
        server wordpress:9000;
    }

    server {
        listen 80;
        server_name _;
        root /var/www/html;
        index index.php;

        location / {
            try_files $uri $uri/ /index.php?$args;
        }

        location ~ \.php$ {
            fastcgi_split_path_info ^(.+\.php)(/.+)$;
            fastcgi_pass php;
            fastcgi_index index.php;
            include fastcgi_params;
            fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
            fastcgi_param PATH_INFO $fastcgi_path_info;
        }

        location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg)$ {
            expires max;
            log_not_found off;
        }
    }
}
```

4. docker compose up -d
5. caddy config **WordPress Installation:**

Sprache: Deutsch Seiten-Titel: z.B. "Weihnachtsdeko Shop" Benutzername: Ein sicherer Admin-Name (NICHT "admin") Passwort: PASSWORT_3 (das dritte generierte) E-Mail: Eine gültige E-Mail-Adresse Haken bei "Suchmaschinen davon abhalten..." (für Testphase)

WooCommerce installieren: Nach dem Login:

Links im Menü: Plugins → Installieren Suche nach "WooCommerce" Installieren → Aktivieren Setup-Wizard durchlaufen (kann übersprungen werden, wird später konfiguriert)

Fertig! □□ Der Shop läuft jetzt unter https://test.denode.eu Nächste Schritte für deine Bekannte:

Theme auswählen (Appearance → Themes) Produkte anlegen (Produkte → Neu hinzufügen) Zahlungsarten konfigurieren (WooCommerce → Einstellungen → Zahlungen)

Overleaf Setup

Setup Overleaf on Docker Compose with all Bibtex

Na klar! Hier ist die komplette, bereinigte Anleitung als Markdown-Datei. Du kannst den Text unten einfach kopieren und z.B. als `README.md` oder `INSTALL.md` in deinem Server-Ordner speichern.

□□ Setup-Guide: Overleaf Community Edition v5 (Docker Compose)

Diese Anleitung beschreibt die Installation der **Overleaf Community Edition (v5+)** auf einem eigenen Server mittels Docker Compose. Sie beinhaltet Lösungen für die MongoDB 8.0 Requirements, HTTP-Login-Probleme und die persistente Installation der vollen TeX-Live-Umgebung.

Voraussetzungen

- **Docker** & **Docker Compose** sind installiert.
- **Speicherplatz:** Mindestens **10–15 GB** frei (für `scheme-full`).
- **Zeit:** Ca. 1–2 Stunden (wegen des Downloads der TeX-Pakete).

---

1. Ordnerstruktur erstellen

Wir erstellen eine persistente Struktur, damit Datenbanken und die TeX-Installation auch nach Updates erhalten bleiben.

```
# Hauptordner erstellen
mkdir -p overleaf-server
cd overleaf-server

# Daten-Verzeichnisse anlegen
mkdir -p overleaf_data
mkdir -p mongo_data
mkdir -p redis_data
mkdir -p texlive_data
```

---

2. Die `docker-compose.yml`

Erstelle die Datei `docker-compose.yml`.

Wichtig: Ersetze `DEINE_SERVER_IP` bei `OVERLEAF_SITE_URL` mit der echten IP oder Domain deines Servers (z.B. `http://192.168.178.50`).

```
version: '3'

services:
  sharelatex:
    image: sharelatex/sharelatex:latest
    container_name: sharelatex
    restart: always
    depends_on:
      - mongo
      - redis
    ports:
      - "80:80"
    links:
      - mongo
      - redis
    volumes:
      # Overleaf Daten (Projekte, User)
```

```
      - ./overleaf_data:/var/lib/overleaf
      # WICHTIG: Persistentes Volume für die TeX-Installation (verhindert Neuinstallation bei
Neustart)
      - ./texlive_data:/usr/local/texlive
    environment:
      # --- App Konfiguration ---
      OVERLEAF_APP_NAME: Overleaf Community
      # MongoDB Replica Set Connection String (Pflicht für v5)
      OVERLEAF_MONGO_URL: mongodb://mongo:27017/sharelatex?replicaSet=rs0
      OVERLEAF_REDIS_HOST: redis
      REDIS_HOST: redis

      # --- Netzwerk & Login Fixes (für HTTP Nutzung) ---
      # Ersetze dies durch deine echte IP! Ohne das werden Cookies oft blockiert.
      OVERLEAF_SITE_URL: "http://DEINE_SERVER_IP"

      # Sicherheitchecks deaktivieren (da kein HTTPS/SSL aktiv ist)
      OVERLEAF_BEHIND_PROXY: "false"
      OVERLEAF_SECURE_COOKIE: "false"
      # Legacy-Variablen ebenfalls deaktivieren (wichtig für Login!)
      SHARELATEX_BEHIND_PROXY: "false"
      SHARELATEX_SECURE_COOKIE: "false"

      # Optional: Cookies toleranter machen
      OVERLEAF_SAME_SITE_COOKIE: "Lax"

      # Feature: User-Registrierung via E-Mail erlauben
      ENABLED_LINKED_FILE_TYPES: 'project_file,project_output_file'

  mongo:
    # Overleaf v5 benötigt zwingend neuere MongoDB Versionen
    image: mongo:8.0
    container_name: mongo
    restart: always
    # Startet MongoDB als Replica Set (Pflicht für Overleaf v5)
    command: "--replSet rs0 --bind_ip_all"
    volumes:
      - ./mongo_data:/data/db

  redis:
```

```
    image: redis:6.2
    container_name: redis
    restart: always
    volumes:
      - ./redis_data:/data
```

---

3. Erster Start & Datenbank-Initialisierung

Da MongoDB als Replica Set läuft, muss es **einmalig** aktiviert werden.

1. **Container starten:**

```
docker-compose up -d
```

(Warte ca. 10–20 Sekunden, bis die Container hochgefahren sind) 2. **Replica Set aktivieren:** Führe diesen Befehl aus, um die Datenbank zu initialisieren:

```
docker exec -it mongo mongosh --eval "rs.initiate({_id:'rs0', members:[{_id:0,
host:'mongo:27017'}]})"
```

Erwartete Ausgabe: `{ "ok" : 1 }` 3. **Overleaf neustarten:** Damit Overleaf die nun fertige Datenbank findet:

```
docker-compose restart sharelatex
```

---

4. TeX Live Installation (Persistent)

Das Docker-Image ist standardmäßig leer. Wir installieren TeX Live manuell in das persistente Volume.

A. Installer laden & Basis installieren

Logge dich in den Container ein:

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

Führe im Container folgende Befehle aus:

```
# 1. In Temp-Ordner wechseln und Installer laden
cd /tmp
wget https://mirror.ctan.org/systems/texlive/tlnet/install-tl-unx.tar.gz

# 2. Entpacken
zcat < install-tl-unx.tar.gz | tar xf -
cd install-tl-*

# 3. Basis-Installation starten (dauert ca. 5 Min)
perl install-tl --scheme=basic --no-interaction

# 4. Pfade verknüpfen (WICHTIG!)
/usr/local/texlive/*/bin/*/tlmgr path add

# 5. Notwendiges Tool installieren
tlmgr install latexmk
```

B. Das "Full Scheme" installieren (Optional, aber empfohlen)

Um alle Pakete zu haben (ca. 4-5 GB Download), installiere `scheme-full`. Da dies lange dauert, nutze am besten `screen` auf deinem Host-Server, damit du SSH schließen kannst.

Verlasse den Container mit `exit`.

Auf dem Host-Server:

```
# 1. Screen Session starten
screen -S latex-install
```

```
# 2. Installation starten
docker exec sharelatex tlmgr install scheme-full
```

Tipp: Du kannst das Fenster nun mit `STRG+A` dann `D` in den Hintergrund schicken ("detachen"). Zurückholen mit `screen -r`.

---

5. Admin-User erstellen

Falls das Web-Formular unter `/launchpad` Probleme macht, erstelle den User über die Konsole:

```
docker exec -it sharelatex /bin/bash -c "cd /overleaf/services/web && node
modules/launchpad/bin/create-admin-user.js --email dein.name@email.com"
```

Kopiere den ausgegebenen Link in deinen Browser, um das Passwort zu setzen.

---

6. Abschluss & Aufräumen

Sobald die Installation von `scheme-full` durchgelaufen ist:

1. **Aufräumen:** Lösche die Installationsdateien im Container, um Platz zu sparen.

```
docker exec sharelatex rm -rf /tmp/install-tl-unx.tar.gz /tmp/install-tl-*
```

2. **Neustart:** Starte den Container einmal neu, um alle Caches zu leeren.

```
docker-compose restart sharelatex
```

Fertig! Dein Overleaf Server ist nun einsatzbereit.

Wartung & Updates

- **Overleaf Update (Web-Oberfläche):**

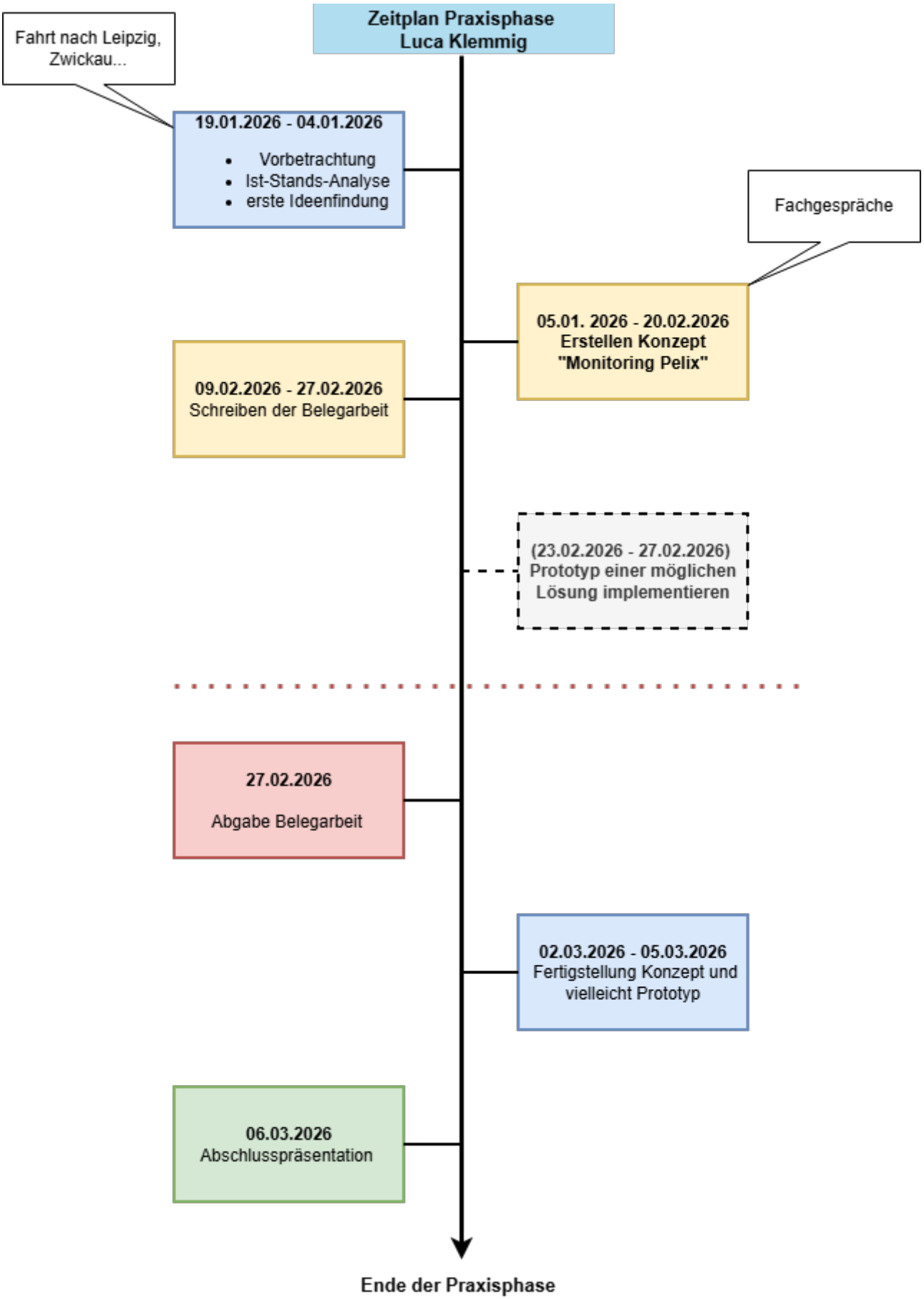
```
docker-compose pull
docker-compose up -d
```

- **LaTeX Pakete Update (Inhalt):**

```
docker exec -it sharelatex tlmgr update --self --all
```

Zeitplan Praxisphase

Zeitplan für die Praxisphase im Referat 45


Zeitplan Praxisphase
Luca Klemmig
Fahrt nach Leipzig, Zwickau...
19.01.2026 - 04.01.2026
Vorbetrachtung
Ist-Stands-Analyse
erste Ideenfindung
Fachgespräche
05.01. 2026 - 20.02.2026
Erstellen Konzept
"Monitoring Pelix"
09.02.2026 - 27.02.2026
Schreiben der Belegarbeit
(23.02.2026 - 27.02.2026)
Prototyp einer möglichen
Lösung implementieren
27.02.2026
Abgabe Belegarbeit
02.03.2026 - 05.03.2026
Fertigstellung Konzept und
vielleicht Prototyp
06.03.2026
Abschlusspräsentation
Ende der Praxisphase

Dump

1. Ipod-Workflow

- iOpenPod
- spotdl download https://link.hier.einfuegen_lol
- syncen