Install

Everything important for setup of the server in the environment of COe.

Table of Contents:
- Setup VPN
- Install Proxmox
- Format Disks

- Lxc Container
- Website with Nginx
- Firefly-III
- HAOS in VM
- OctoPi
- Grafana
- Antragsgruen
- Vaultwarden
- Mailcow

Lxc Container

Create Container

You can use the helperscript with this command:

```
bash -c "$(wget -qLO - https://github.com/tteck/Proxmox/raw/main/ct/ubuntu.sh)"
```

For the Container ID use a number in the range 200-299. It is useful to give the Container this ID as the ending of the IP-Address later.

In der Cloud ist ein Skript, das folgendes ausführt:

Nutzer erstellen

Create a user:

```
adduser <username>
```

Add user to sudo group:

```
usermod -aG sudo <username>
```

Switch to user.

System vorbereiten

Update the system and install Lazy-Vim using:

```
sudo apt update
sudo apt upgrade -y
sudo apt install -y git build-essential neovim tree curl
```

```
git clone https://github.com/LazyVim/starter ~/.config/nvim
rm -rf ~/.config/nvim/.git
```

Run those Lines also as `root`.

```
git clone https://github.com/LazyVim/starter ~/.config/nvim
rm -rf ~/.config/nvim/.git
```

Install Zsh with autocompletions:

```
sudo apt install -y zsh zsh-syntax-highlighting
sh -c "$(curl -fsSL https://raw.githubusercontent.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"
git clone https://github.com/zsh-users/zsh-history-substring-search ${ZSH_CUSTOM:-~/.oh-my-zsh/custom}/plugins/zsh-history-substring-search
git clone --depth 1 https://github.com/junegunn/fzf.git ~/.config/fzf
~/.config/fzf/install
```

Replace "~/.zshrc" with:

```
export ZSH="$HOME/.oh-my-zsh"
source $ZSH/oh-my-zsh.sh

ZSH_AUTOSUGGEST_STRATEGY=(history)

plugins=(cabal colorize colored-man-pages cp copyfile copypath fzf git gitignore last-working-dir sudo vi-mode web-search zsh-interactive-cd)

ZSH_THEME="jonathan"

HISTSIZE="100000000000"
SAVEHIST="100000000000"

HISTFILE="/home/elias/.local/share/zsh/zsh_history"
mkdir -p "$(dirname "$HISTFILE")"

setopt HIST_FCNTL_LOCK
unsetopt APPEND_HISTORY
setopt HIST_IGNORE_DUPS
unsetopt HIST_IGNORE_ALL_DUPS
setopt HIST_IGNORE_SPACE
setopt HIST_EXPIRE_DUPS_FIRST
```

```
setopt SHARE_HISTORY
unsetopt EXTENDED_HISTORY

PROMPT='%F{green}%n%f@%F{magenta}%m%f %F{blue}%B%~%b%f %# '
RPROMPT='[%F{yellow}%?%f]'
bindkey "^[[A" history-beginning-search-backward
bindkey "^[[B" history-beginning-search-forward

source /usr/share/zsh-syntax-highlighting/zsh-syntax-highlighting.zsh
ZSH_HIGHLIGHT_HIGHLIGHTERS+=()

[ -f ~/.fzf.zsh ] && source ~/.fzf.zsh
```

Switch to `root` and replace "~/.zshrc" with:

```
ZSH_DISABLE_COMPFIX=true

export ZSH="/home/elias/.oh-my-zsh"
source $ZSH/oh-my-zsh.sh

ZSH_AUTOSUGGEST_STRATEGY=(history)

plugins=(cabal colorize colored-man-pages cp copyfile copypath fzf git gitignore last-working-dir sudo vi-mode
web-search zsh-interactive-cd)

ZSH_THEME="jonathan"

HISTSIZE="100000000000"
SAVEHIST="100000000000"

HISTFILE="/home/elias/.local/share/zsh/zsh_history"
mkdir -p "$(dirname "$HISTFILE")"

setopt HIST_FCNTL_LOCK
unsetopt APPEND_HISTORY
setopt HIST_IGNORE_DUPS
unsetopt HIST_IGNORE_ALL_DUPS
setopt HIST_IGNORE_SPACE
setopt HIST_EXPIRE_DUPS_FIRST
setopt SHARE_HISTORY
```

```
unsetopt EXTENDED_HISTORY

PROMPT='%F{green}%n%f@%F{magenta}%m%f %F{blue}%B%~%b%f %# '
RPROMPT='[%F{yellow}%?%f]'
bindkey "^[[A" history-beginning-search-backward
bindkey "^[[B" history-beginning-search-forward

source /usr/share/zsh-syntax-highlighting/zsh-syntax-highlighting.zsh
ZSH_HIGHLIGHT_HIGHLIGHTERS+=()

[ -f ~/.fzf.zsh ] && source ~/.fzf.zsh
```

Please give the Server a fixed local IP-Adress.

You can add the ssh login, if you run the following command from the computer you want to access with:

```
ssh-copy-id <usr>@<ip>
```

Website with Nginx

Nginx

Install Nginx with

```
sudo apt install nginx
sudo service nginx start
sudo systemctl enable nginx
```

Create a new Configuration File with:
`sudo vim /etc/nginx/sites-available/mywebsite`
and test ist with `sudo nginx -t` and then restart with `sudo service nginx restart`

PHP

Install php 8.3 with

```
apt-get install ca-certificates apt-transport-https software-properties-common
# Add Ondrej's PPA
sudo add-apt-repository ppa:ondrej/php
sudo apt update

# Install new PHP 8.3 packages
sudo apt install php8.3 php8.3-cli php8.3-{bz2,curl,mbstring,intl}

# Install FPM OR Apache module
sudo apt install php8.3-fpm
# OR
# sudo apt install libapache2-mod-php8.2

# On Apache: Enable PHP 8.3 FPM
sudo a2enconf php8.3-fpm
```

Firefly-III

Preparation

PHP

```
sudo apt update
sudo apt upgrade -y
sudo apt install -y nginx curl software-properties-common php8.3 php8.3-{cli,zip,gd,fpm,common,mysql,zip,mbstring,curl,xml,bcmath,imap,ldap,intl} php-json
```

Check to see if php is running

`php -v` and `systemctl status php8.3-fpm`

Adjust some php settings

```
sudo nvim /etc/php/8.3/fpm/php.ini
```

search for and change or enable the following lines of code

```
memory_limit = 512M


[Date]
date.timezone = Europe/Berlin
```

Nginx

stop apache

```
sudo systemctl stop apache2
sudo systemctl disable apache2
```

remove nginx file

```
sudo rm /etc/nginx/sites-enabled/default
```

create "firefly.dodekaeder.name" in sites-enabled folder and then paste in the config below

```
sudo nvim /etc/nginx/sites-enabled/firefly.dodekaeder.name
```

```
server {
        listen       80 default_server;
        listen       [::]:80 default_server;
        #server_name  subdomain.domain.com;
        root         /var/www/firefly-iii/public;
        index index.html index.htm index.php;

        location / {
                try_files $uri /index.php$is_args$args;
                autoindex on;
                sendfile off;
       }

        location ~ \.php$ {
        fastcgi_pass unix:/run/php/php8.3-fpm.sock;
        fastcgi_index index.php;
        fastcgi_read_timeout 240;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
        fastcgi_split_path_info ^(.+.php)(/.+)$;
        }

    }
```

restart

```
sudo systemctl restart nginx php8.3-fpm
```

If you get an error due to duplicate web servers, you need to remove the symbolic link in sites-available (ls -l) and then rm the link

MariaDB

Install mariaDB

```
curl -LsS https://r.mariadb.com/downloads/mariadb_repo_setup | sudo bash -s -- \
  --mariadb-server-version="mariadb-10.11" --os-type="ubuntu" --os-version="noble"
sudo apt install -y mariadb-server mariadb-client
sudo mysql_secure_installation
```

test installation

```
mysql -u root -p
```

in the mysql shell, check version with the command in bold

```
SELECT VERSION();
```

while still inside the mariaDB shell:

```
CREATE DATABASE firefly_db;
CREATE USER 'fireflyuser'@'localhost' IDENTIFIED BY 'yourpasswordhere';
GRANT ALL PRIVILEGES ON firefly_db.* TO 'fireflyuser'@'localhost';
FLUSH PRIVILEGES;
exit;

cd ~
curl -sS https://getcomposer.org/installer -o composer-setup.php
sudo php composer-setup.php --install-dir=/usr/local/bin --filename=composer
```

Install

test composer

```
composer -V
```

Download the latest Version from here.
Unpack it with

```
unzip -o FireflyIII-v6.1.24.zip -d /var/www/firefly-iii
sudo chown -R elias /var/www/firefly-iiicd /var/www/firefly-iii
```

Change in the ".env" file those lines:

```
DB_HOST=127.0.0.1
DB_DATABASEE=firefly_db
DB_USERNAME=<usr>
DB_PASSWORD=<pwd>

TZ=Europe/Berlin
```

Set up Composer and install

```
composer install --no-dev --no-scripts
php artisan key:generate
php artisan migrate --seed
php artisan firefly-iii:decrypt-all
php artisan cache:clear
php artisan view:clear
php artisan firefly-iii:upgrade-database
php artisan firefly-iii:laravel-passport-keys
```

change the owner of the folder with

```
sudo chown -R www-data:www-data /var/www/firefly-iii
sudo chmod -R 775 /var/www/firefly-iii/storage
```

uncomment those lines in the file "/etc/locale.gen"

```
de_DE ISO-8859-1
de_DE.UTF-8 UTF-8
de_DE@euro ISO-8859-15
```

and generate the locale using

```
sudo locale-gen
```

HAOS in VM

Install

Create a VM and install Home Assistant OS on it using his command:

```
bash -c "$(wget -qLO - https://github.com/tteck/Proxmox/raw/main/vm/haos-vm.sh)"
```

Source

TTeck Installer Script

OctoPi

We will use the Raspberry Pi Imager for installation.

Open the Imager and select the Pi Version and OctoPi image under “Choose OS”, by selecting “Other Specific Purpose OS” > “3D printing” > “OctoPi” and then the “stable” version.

Open advanced options by using the keyboard shortcut ctrl+shift+x and then:

- Configure your wifi options: Set your SSID, password and WiFi country.
- Change the system password in “Set username and password” by entering a new password to use for the system user “pi”. This is not the password you’ll use for logging into OctoPrint but one that you’ll have to use to log into your Pi via SSH should you ever need to.
- Optionally: Change the configured timezone in “Set locale settings”
- Optionally: Change the hostname in “Set hostname”

Install the image to your SD card, then plug everything in to your Raspberry Pi and boot it up. Do not format the SD card after installing, even if prompted to do so. This will break the installation and you will have to start over!

Access OctoPrint from your browser via http://octopi.local or the hostname you chose (if your computer supports bonjour) or http://<your pi's ip address>. https is available too, with a self-signed certificate (which means your browser will warn you about it being invalid - it isn’t, it’s just not recognized by your browser).

Grafana

Install Grafana

Add Grafana to the sources list:

```
sudo apt-get install -y apt-transport-https
sudo apt-get install -y software-properties-common wget
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list
sudo apt-get update
```

Install Grafana:

```
sudo apt-get install grafana
```

Install the .deb package

```
sudo apt-get install -y adduser libfontconfig1
wget https://dl.grafana.com/oss/release/grafana_7.1.1_amd64.deb
sudo dpkg -i grafana_7.1.1_amd64.deb
sudo systemctl daemon-reload
sudo systemctl start grafana-server
sudo systemctl status grafana-server
```

Source

Source

Antragsgruen

Preparation

Database

Install MySQL

```
sudo apt install mysql-server -y
sudo systemctl enable mysql
sudo systemctl start mysql
```

Setup the installation with `sudo mysql_secure_installation` and choose:

- VALIDATE PASSWORD component: Y
- Password Validation Policy: 3
- Remove anonymous users?: Y
- Disallow root login remotely?: Y
- Remove test database and access to it?: Y
- Reload privilege tables now?: Y

Enter MySQL with `sudo mysql -u root -p` and create Database and user:

```
CREATE DATABASE antragsgruen;
CREATE USER '<my_user>'@'localhost' IDENTIFIED BY '<my_password>';
GRANT ALL PRIVILEGES ON antragsgruen.* TO '<my_user>'@'localhost';
FLUSH PRIVILEGES;
Exit
```

PHP8.4

Add PHP8.4 Repository and install it:

```
sudo apt install software-properties-common
sudo add-apt-repository ppa:ondrej/php
sudo apt update
sudo apt-get install php8.4 php8.4-cli php8.4-fpm php8.4-intl php8.4-gd php8.4-mysql php8.4-opcache php8.4-curl php8.4-xml php8.4-mbstring php8.4-zip php8.4-iconv
```

Apache2

Install Apache2:

```
sudo apt install apache2
```

Change the default apache2 site (`/etc/apache2/sites-available/000-default.conf`) to:

```
RewriteEngine On

<Directory /var/www/antragsgruen/web>
    AllowOverride all
    Require all granted
</Directory>

<VirtualHost *:80>
    DocumentRoot /var/www/antragsgruen/web
    ServerName antragsgruen.example.org

    # Other directives here
</VirtualHost>
```

Enable the RewriteEngine and restart Apache2:

```
sudo a2enmod rewrite && sudo service apache2 restart
```

Nodejs 20 & npm

Download ans execute the NodeSource setup script:

```
curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
```

Install Node.js and npm:

```
sudo apt-get install -y nodejs
```

Install

Install the sources and dependencies from the repository:

```
sudo git clone https://github.com/CatoTH/antragsgruen.git
sudo chmod 777 antragsgruen
cd antragsgruen
curl -sS https://getcomposer.org/installer | php
./composer.phar install --prefer-dist
npm install
npm run build
```

To enable the web-based installer:

```
touch config/INSTALLING
```

Set the permissions:

```
sudo chown -R www-data:www-data web/assets
sudo chown -R www-data:www-data runtime
sudo chown -R www-data:www-data config
```

Sources

- MySQL Guide
- PHP 8.4
- Nodejs 20 & npm
- Antragsgruen

Vaultwarden

Preparation

Docker Compose

Install Docker

```
sudo apt install docker.io
```

Install Docker Compose

```
curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose
```

Install

Create the file `compose.yaml` with this content:

```
services:
  vaultwarden:
    image: vaultwarden/server:latest
    container_name: vaultwarden
    restart: unless-stopped
    environment:
      DOMAIN: "https://vw.domain.tld"
    volumes:
      - ./vw-data/:/data/
    ports:
      - 80:80
```

Start the Container with:

```
docker-compose -f compose.yaml up -d
```

Sources

- Docker-Compose
- Vaultwarden

Mailcow

Preparation

Docker Compose

Install Docker

Prep

Portforwads:

https://docs.mailcow.email/getstarted/prerequisite-system/#firewall-ports

```
sudo apt install docker.io
```

Install Docker Compose

```
curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose
```

Check SELinux specifics¶

Install SELinux

```
sudo apt install policycoreutils selinux-utils selinux-basics
sudo selinux-activate
```

```
sudo selinux-config-enforcing
```

Enable SELinux by creating the file `/etc/docker.daemon.json` with the content:

```
{
  "selinux-enabled": true
}
```

Install

Create the file `compose.yaml` with this content:

```
services:
  vaultwarden:
    image: vaultwarden/server:latest
    container_name: vaultwarden
    restart: unless-stopped
    environment:
      DOMAIN: "https://vw.domain.tld"
    volumes:
      - ./vw-data/:/data/
    ports:
      - 80:80
```

Start the Container with:

```
docker-compose -f compose.yaml up -d
```

Sources

- Docker-Compose
- SELinux