SELF HOSTED SERVER

Proxmox installation

(For a more comprehensive guide visit https://pve.proxmox.com/wiki/Installation or read the attached "Proxmox Install Beginners Guide" article)

1. **Download the Proxmox VE ISO:**
   - **Visit the Proxmox website and download the latest Proxmox VE ISO image[1].**
2. **Create a Bootable USB Drive:**
   - **Use a tool like Rufus or Ventoy to burn the ISO image to a USB flash drive[2].**
3. **Boot from the USB Drive:**
   - **Insert the USB drive into your server and boot from it. You may need to change the boot order in the BIOS/UEFI settings[1].**

4. **Start the Installation Wizard:**
    - **Once the server boots from the USB drive, the Proxmox installation wizard will start. Press ‘Enter’ to begin the installation[1].**
5. **Configure Basic Settings:**
    - **Follow the prompts to configure basic settings such as language, time zone, and keyboard layout[3].**
6. **Partition the Disk:**
    - **Choose the target hard disk for the installation. Note that this will erase all data on the selected disk[3].**
7. **Set Up Network Configuration:**
    - **Configure the network settings, including the hostname, IP address, and DNS settings[3].**
8. **Create an Admin Password:**
    - **Set a password for the ‘root’ user and provide an email address for system notifications[3].**
9. **Complete the Installation:**
    - **Review the summary of your settings and start the installation. This process will take a few minutes[3].**
10. **Reboot the Server:**
    - **After the installation is complete, remove the USB drive and reboot the server[3].**
11. **Access the Web Interface from another PC:**
    - **Once the server has rebooted, you can access the Proxmox web interface by navigating to https://<your-server-ip>:8006 in your web browser[1].**

Uploading ISOs from USB to Proxmox

1. **Prepare Your USB Drive**:
    - Ensure your USB drive is formatted with a file system that Proxmox can read, such as ext4 or NTFS.
    - Copy the ISO files to the USB drive from your local computer.
2. **Insert the USB Drive into the Proxmox Server**:
    - Plug the USB drive into an available USB port on your Proxmox server.
3. **Access the Proxmox Shell**:
    - Open the Proxmox web interface and log in.

- Navigate to the Shell option from the left-hand menu.

4. **Identify the USB Drive**:
   - Run the command lsblk to list all block devices and identify your USB drive (e.g., /dev/sdb).
5. **Mount the USB Drive**:
   - Create a mount point: mkdir /mnt/usb
   - Mount the USB drive: mount /dev/sdb1 /mnt/usb (replace /dev/sdb1 with the correct device identifier).
6. **Copy ISO Files to Proxmox Storage**:
   - Navigate to the directory where Proxmox stores ISOs: cd /var/lib/vz/template/iso
   - Copy the ISO files from the USB drive: cp /mnt/usb/*.iso .
7. **Verify the Upload**:
   - Go back to the Proxmox web interface.
8. **Navigate to Storage**:
   - In the left-hand panel, click on your Proxmox node (e.g., pve), then go to Datacenter > pve > local (pve) > ISO Images.
9. **Upload ISO File**:
   - In the upload window, click Select File and choose the ISO file from your local computer. Click Upload to start the process[1].

Creating a Windows VM with VirtIO Drivers

(For a more comprehensive guide read the attached article "Install Windows In a VM")

1.
2. **Create a New VM**:
   - In the Proxmox web interface, right-click on your node and select Create VM.
3. **General Settings**:
   - Enter a name for your VM and click Next.
4. **OS Settings**:
   - Select the Windows ISO you uploaded earlier from the ISO Image dropdown. Set the Guest OS type to Microsoft Windows and click Next.

5. **System Settings**:
    - Configure the system settings as needed. Ensure the BIOS is set to OVMF (UEFI) and check the Qemu Agent option.
6. **Hard Disk Settings**:
    - Choose VirtIO SCSI as the bus type for better performance. Set the disk size and other parameters as needed.
7. **CPU and Memory Settings**:
    - Allocate the desired number of CPU cores and amount of RAM for your VM.
8. **Network Settings**:
    - Select VirtIO (paravirtualized) for the network adapter type.
9. **Add VirtIO Drivers**:
    - Before finishing, go to the Hardware tab of your VM and add a second CD/DVD drive. Select the VirtIO ISO you uploaded earlier.
10. **Start the VM**:
    - Once the VM is created, start it. During the Windows installation, choose the Custom installation option and load the VirtIO drivers from the second CD/DVD drive to detect the hard disk[2].
11. **Complete Windows Installation**:
    - Follow the prompts to complete the Windows installation. Once done, you can remove the VirtIO ISO from the CD/DVD drive.

These steps should help you get your Windows VM up and running on Proxmox with the necessary VirtIO drivers. If you have any more questions or need further assistance, feel free to ask!

Helper Scripts for LXC installation

Project Overview

Proxmox VE Helper-Scripts is a collection of tools to simplify the setup and management of Proxmox Virtual Environment (VE). Originally created by tteck, these scripts are now continued by the community. Our goal is to preserve and expand upon tteck's work, providing an ongoing resource for Proxmox users worldwide.

Features

- Interactive Setup: Choose between simple and advanced options for configuring VMs and LXC containers.
- Customizable Configurations: Advanced setup for fine-tuning your environment.

- Seamless Integration: Works seamlessly with Proxmox VE for a smooth experience.
- Community-driven: Actively maintained and improved by the Proxmox community.

Requirements

Ensure your system meets the following prerequisites:

- Proxmox VE version: 8.x or higher
- Linux: Compatible with most distributions
- Dependencies: bash and curl should be installed.

Installation

To install the Proxmox Helper Scripts, follow these steps:

1. Visit the Website.
2. Search for the desired script, e.g., "Home Assistant OS VM".
3. Copy the provided Bash command from the "How To Install" section.
4. Open the Proxmox shell on your main node and paste the command.
5. Press enter to start the installation!

Proxmox Helper Scripts: Nextcloud:

Nextcloud Hub is the industry-leading, fully open-source, on premise content collaboration platform. Teams access, share and edit their documents, chat and participate in video calls and manage their mail and calendar and projects across mobile, desktop and web interfaces.

Features:

File Management

- File Sync and Share: Sync files across devices and share them with others securely.
- Folder Upload: Upload entire folders directly through the web interface.
- File List Filters: Quickly navigate files based on criteria like file type.
- Trash Bin: Shows the original location of deleted files and who deleted them, aiding in file recovery.

Collaboration Tools

- Nextcloud Talk: Secure audio/video calls and chat.
- Nextcloud Office: Collaborative document editing with support for various file formats.
- Whiteboard: A canvas-like whiteboard for collaborative sketching, planning, and brainstorming.

Security and Privacy

- End-to-End Encryption: Protects your data with strong encryption.
- Two-Factor Authentication: Adds an extra layer of security.
- Suspicious Login Notifications: Alerts you to potential security issues.

User Management

- Admin Delegation: Allows administrators to delegate user and group management tasks.
- Out-of-Office Replacement: Users can select another user as their replacement during absences.
- Password Hash Management: Facilitates user management and migration scenarios.

AI and Automation

- Nextcloud Assistant: An on-premise AI assistant that can summarize emails, transcribe audio, and more.
- Context Chat and Write: AI-powered tools that can access your data to answer questions or generate text based on existing files.

Customization and Theming

- Custom Themes: Allows for separate primary and background colors.
- Dark Mode Support: Enhances the visual experience in dark mode.

Integration and Extensibility

- App Store: A wide range of apps to extend functionality, from calendars to task management.
- API Access: Integrate Nextcloud with other tools and services.

To use the Nexctcloud Install script, run the command below in the shell.

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

Proxmox Helper Scripts: Jellyfin:

Jellyfin is the volunteer-built media solution that puts *you* in control of your media. Stream to any device from your own server, with no strings attached. Your media, your server, your way.

Features:

Cross-Platform Compatibility

- Runs on Various OS: Jellyfin can be installed on Windows, macOS, Linux, and Docker[1].

Media Management and Organization

- Library Management: Organize your movies, TV shows, music, and photos into libraries.
- Metadata Fetching: Automatically fetches metadata for your media, including posters, descriptions, and ratings.

Transcoding and Streaming

- Real-Time Transcoding: Supports real-time transcoding to ensure smooth playback on various devices.
- Multiple Clients: Stream your media to multiple devices, including smart TVs, smartphones, and web browsers[2].

User Profiles and Parental Controls

- User Profiles: Create multiple user profiles with individual preferences and watch histories.
- Parental Controls: Set restrictions to control what content is accessible to different users[3].

Privacy-Focused and Self-Hosted

- Self-Hosted: Keep your data private by hosting Jellyfin on your own server.
- No Tracking: Jellyfin does not track your usage or collect your data[2].

Extensive Plugin Support

- Plugins: Enhance functionality with a wide range of plugins, such as live TV, music streaming, and more[3].

Community-Driven Development

- Open Source: Developed by a community of volunteers, ensuring continuous improvement and updates.
- Active Community: Engage with a vibrant community for support and contributions[1].

To use the Jellyfin Install script, run the command below in the shell.

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

Proxmox Helper Scripts: The Proxmox Post Install Script

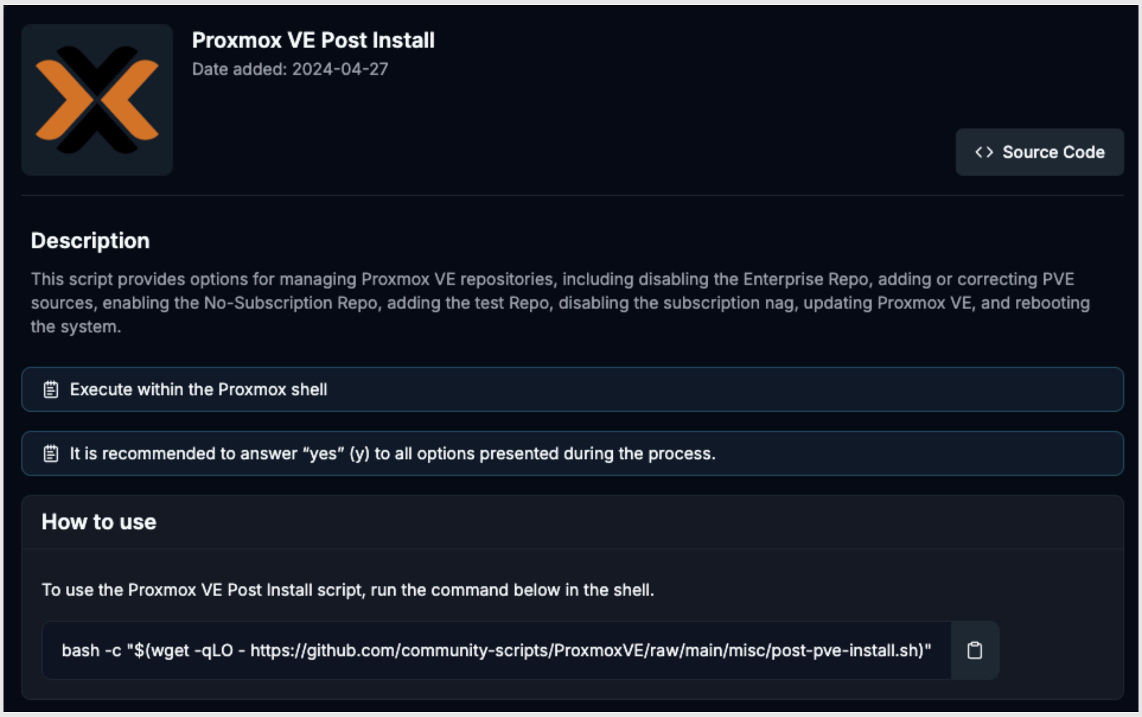
A fresh Proxmox installation often requires some post-installation tweaks. The Proxmox Post Install Script automates these adjustments:

Features:

1. Repository Management:
   - Disables the enterprise repository (if no subscription is used).
   - Enables the no-subscription repository.
   - Optionally adds a test repository (disabled by default).
2. Nag Screen Removal:
   - Disables the subscription warning on login.
3. High Availability Optimization:
   - Disables high availability features on single-node setups to free up resources.
4. System Updates:
   - Updates Proxmox VE and ensures the latest kernel is installed.
5. Reboot Prompt:
   - Reboots the system to apply updates and kernel changes.

How to Execute:

- Review the Proxmox Post Install script.


Proxmox VE Post Install

Date added: 2024-04-27

Description

This script provides options for managing Proxmox VE repositories, including disabling the Enterprise Repo, adding or correcting PVE sources, enabling the No-Subscription Repo, adding the test Repo, disabling the subscription nag, updating Proxmox VE, and rebooting the system.

Execute within the Proxmox shell

It is recommended to answer “yes” (y) to all options presented during the process.

How to use

To use the Proxmox VE Post Install script, run the command below in the shell.

```
bash -c "$(wget -qLO - https://github.com/community-scripts/ProxmoxVE/raw/main/misc/post-pve-install.sh)"
```

- To use the Proxmox Backup Server Post Install script, run the command below in the shell.

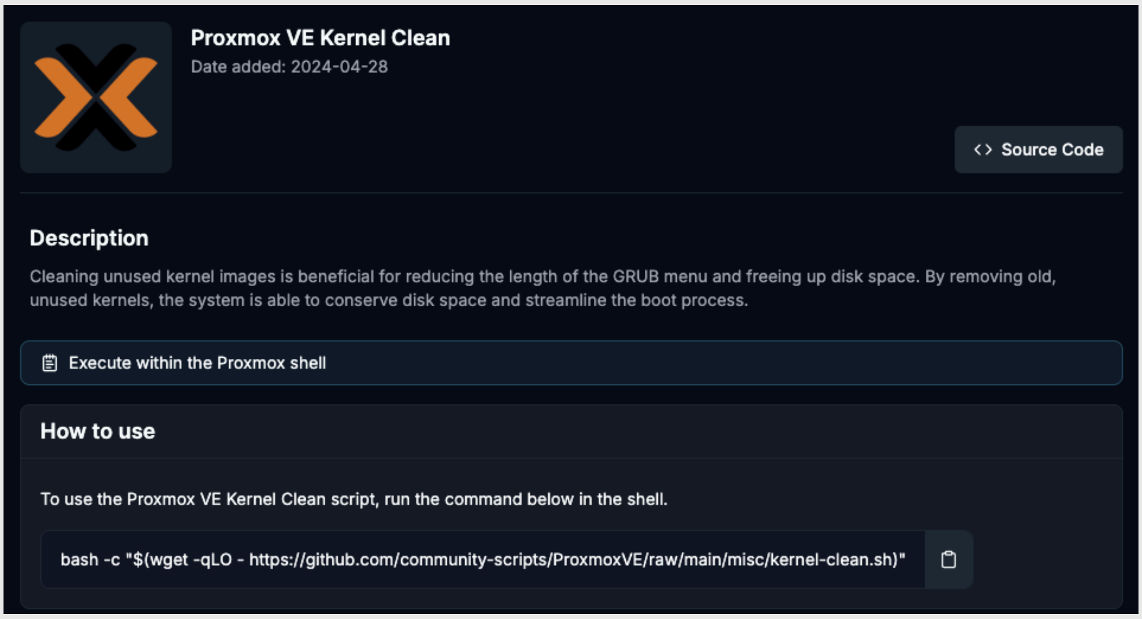
  ```
  bash -c "$(wget -qLO - https://github.com/community-scripts/ProxmoxVE/raw/main/misc/post-pbs-install.sh)"
  ```

Proxmox Helper Scripts: Kernel Cleanup Script

As you update Proxmox over time, old kernels accumulate and consume valuable disk space. The Kernel Cleanup Script simplifies the removal of outdated kernels.

Steps:

1. Review the Kernel Cleanup Script



Proxmox VE Kernel Clean
Date added: 2024-04-28

Description

Cleaning unused kernel images is beneficial for reducing the length of the GRUB menu and freeing up disk space. By removing old, unused kernels, the system is able to conserve disk space and streamline the boot process.

Execute within the Proxmox shell

How to use

To use the Proxmox VE Kernel Clean script, run the command below in the shell.

```
bash -c "$(wget -qLO - https://github.com/community-scripts/ProxmoxVE/raw/main/misc/kernel-clean.sh)"
```

1. Check your current kernel version:
   `uname -r`

2. Run the Kernel Cleanup Script:
   bash -c "$(wget -qLO -
   https://github.com/community-scripts/ProxmoxVE/raw/main/misc/kernel-clean.s
   h)"

3. Safeguards:
   - The script does not delete the currently active kernel.
   - It's recommended to retain the last one or two kernels for fallback purposes.

Proxmox Helper Scripts: Proxmox Host Backup Script

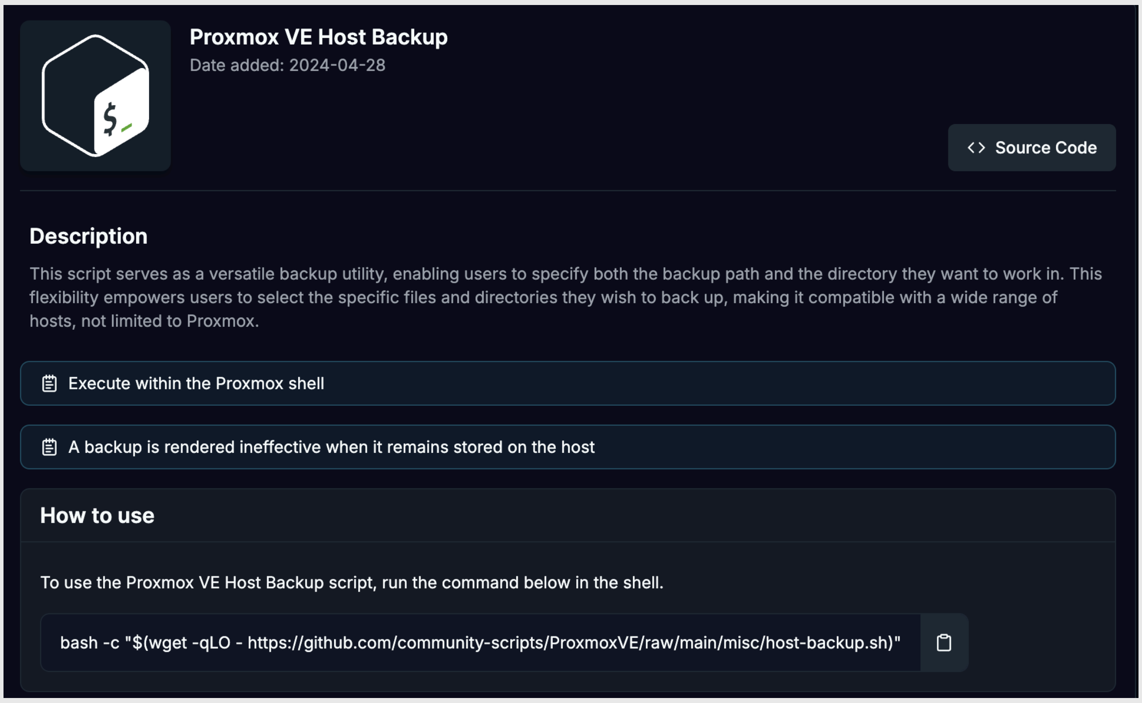
Regular backups of your Proxmox configuration are crucial for disaster recovery. The Proxmox Host Backup Script automates the backup process so that your configurations are safe.

Features:

- Backs up the Proxmox VE configuration files.
- Stores the backup in a user-specified location.
- Can be scheduled for regular execution via cron.

How to Execute:

1. Review the Proxmox VE Host Backup script:



Proxmox VE Host Backup

Date added: 2024-04-28

Description

This script serves as a versatile backup utility, enabling users to specify both the backup path and the directory they want to work in. This flexibility empowers users to select the specific files and directories they wish to back up, making it compatible with a wide range of hosts, not limited to Proxmox.

Execute within the Proxmox shell

A backup is rendered ineffective when it remains stored on the host

How to use

To use the Proxmox VE Host Backup script, run the command below in the shell.

```
bash -c "$(wget -qLO - https://github.com/community-scripts/ProxmoxVE/raw/main/misc/host-backup.sh)"
```

2. Run the Script:
   - Open the Proxmox shell and paste the backup script command.

bash -c "$(wget -qLO -
https://github.com/community-scripts/ProxmoxVE/raw/main/misc/kernel-clean.sh)"

1.
   - The script will prompt you for configuration details.
2. Choose Backup Destination:
   - Select where you want to back up your files.
     - Recommended: Mount a network directory (e.g., NFS or SMB share) and back up there.
     - Alternative: Use a local directory for testing (e.g., /root/backups).
2. Select Files to Back Up:
   - Specify the directory to back up:
     - Recommended: /etc/pve (contains essential configurations for VMs, networks, etc.).
     - Optionally, back up the entire /etc folder for a broader backup.
2. Run the Backup:
   - Confirm the source and destination paths.
   - Let the script run and create a backup archive.

3. Verify the Backup:
    - Navigate to the backup destination (e.g., /root or the mounted network share).
    - Check for the backup archive file (e.g., a .tar.gz file).
4. Optional: Automate the Process
    - Mount a network share permanently.
    - Add the backup script to a cron job for regular execution.

What Are the Best Practices for Using Proxmox Helper Scripts?

1. Review Scripts Before Execution:
    - Always inspect the source code of scripts downloaded from the internet.
    - Validate their integrity by checking commit tags or GitHub issues.
2. Test on a Staging Environment:
    - Before deploying scripts on production servers, test them in a safe environment.
3. Regular Updates:
    - Keep Proxmox VE and the Helper Scripts up to date to ensure compatibility and security.
4. Backup Frequently:
    - Use the Proxmox Host Backup Script to safeguard configurations.
    - Maintain offsite backups for additional redundancy.

More Helper Scripts

https://tteck.github.io/Proxmox/

Reverse-proxy installation

A reverse proxy is a server that sits in front of web servers and forwards client requests (like those from web browsers) to those web servers. Reverse proxies are commonly used to enhance the performance, security, and reliability of web applications.

Here are some key functions and benefits of a reverse proxy:

Key Functions:

- Request Forwarding: It receives client requests and forwards them to the appropriate backend server.
- Response Handling: It collects the responses from the backend servers and sends them back to the clients.

Benefits:

- Security: By hiding the backend servers, it helps protect them from direct attacks[1].
- Load Balancing: Distributes incoming traffic across multiple servers to ensure no single server is overwhelmed[2].
- SSL/TLS Termination: Handles encryption and decryption of traffic, reducing the load on backend servers[2].
- Caching: Stores frequently accessed content closer to the clients, improving response times[2].

Relevant files

1. **Main Configuration File**:
   - **Path**: /etc/nginx/nginx.conf
   - **Description**: This is the main configuration file for NGINX, containing global settings and references to other configuration files.
2. **Site Configuration Files**:
   - **Path**: /etc/nginx/sites-available/
   - **Description**: This directory contains configuration files for different sites or applications. Each file defines the settings for a specific site or application.
3. **Enabled Site Configurations**:
   - **Path**: /etc/nginx/sites-enabled/
   - **Description**: This directory contains symbolic links to the active site configurations in sites-available/. Only the configurations linked here are actually used by NGINX.
4. **Additional Configuration Files**:

   - **Path**: /etc/nginx/conf.d/
   - **Description**: This directory can contain additional configuration files that are included in the main nginx.conf file. It's often used for global settings or additional server blocks.
5. **SSL Certificates and Keys**:
   - **Path**: /etc/nginx/ssl/
   - **Description**: If you're using SSL/TLS, this directory typically contains your SSL certificate and key files.
6. **Log Files**:
   - **Path**: /var/log/nginx/
   - **Description**: This directory contains log files for NGINX, including access logs and error logs. These logs are useful for troubleshooting and monitoring.

Steps to Get a Domain Under Duck DNS

1. **Create an Account**:
   - Visit the Duck DNS website.
   - Sign in using one of the available options (e.g., Google, GitHub, Twitter).
2. **Create a Domain**:
   - Once logged in, you'll see an option to create a new domain.
   - Enter your desired domain name (e.g., mydomain) and click "Add Domain".
   - Duck DNS will create the domain mydomain.duckdns.org.
3. **Update IP Address**:
   - Duck DNS provides a token and a URL to update your IP address.
   - You can use this URL in a script to keep your domain updated with your current IP address.
4. **Subdomains**:
   - You don't need to create subdomains separately on Duck DNS.
   - Once you have mydomain.duckdns.org, any subdomain (e.g., sub.mydomain.duckdns.org) will work automatically.
   -

Example Script to Update IP Address

You can use a simple script to update your IP address with Duck DNS:

1. **Access Your Host Machine**:
   - Log into your host machine (the one running the LXC containers).
2. **Create a Directory for the Script**:
   - Create a directory to store the DuckDNS script:
   - mkdir -p ~/duckdns
3. **Create the Script File**:
   - Navigate to the new directory and create the script file:
   - cd ~/duckdns

     nano duck.sh

1. **Edit the Script File**:
   - Add the following content to the script, replacing yourdomain and yourtoken with your actual DuckDNS domain and token:

```
#!/bin/bash

DOMAIN="mydomain"

TOKEN="your-duckdns-token"

echo url="https://www.duckdns.org/update?domains=$DOMAIN&token=$TOKEN&ip="
| curl -k -o ~/duckdns/duck.log -K -

*/5 * * * * ~/duckdns/duck.sh >/dev/null 2>&1
```

Script Breakdown

- **URL Construction**: The script constructs a URL with your Duck DNS domain and token.
- echo url="https://www.duckdns.org/update?domains=mydomain&token=d9a58a9f-3cfa-43fe-8f0e-e081c8335cdf&ip="
- **Curl Command**: It uses curl to send a request to Duck DNS, updating your IP address.
- | curl -k -o /root/duckdns/duck.log -K -
  - -k: Allows insecure SSL connections and transfers.
  - -o /root/duckdns/duck.log: Outputs the result to a log file.
  - -K -: Reads the URL from the standard input.

1. **Make the Script Executable**:
   - Change the permissions of the script to make it executable:

chmod +x ~/duckdns/duck.sh

Ensuring It Runs Regularly

To ensure this script runs regularly and keeps your IP address updated, you can set up a cron job:

1. **Edit Crontab**:
2. crontab -e
3. **Add Cron Job**:
   - Add the following line to run the script every 5 minutes:

*/5 * * * * /root/duckdns/duck.sh >/dev/null 2>&1

<u>Setting up reverse proxy using an LXC container and NGINX:</u>

Step 1: Create an LXC Container

1. **Log in to Proxmox Web Interface**.
2. **Create a New LXC Container**:
   - Go to Create CT.
   - Fill in the container details (e.g., hostname, password).
   - Choose an appropriate template (e.g., Ubuntu).
   - Allocate resources (CPU, RAM, disk space).
   - Configure network settings (assign an IP address).

Step 2: Access the LXC Container

1. **Start the Container**.
2. **Access the Container**:
   - Use the Proxmox web interface console or SSH into the container using its IP address.

Step 3: Update and Install NGINX

1. **Update Package Lists**:
2. sudo apt-get update
3. sudo apt-get upgrade

4. **Install NGINX**:
5. sudo apt-get install nginx

Step 4: Configure NGINX as a Reverse Proxy

1. **Create a Configuration File** in /etc/nginx/sites-available/:
2. sudo nano /etc/nginx/sites-available/reverse-proxy.conf
3. **Add Reverse Proxy Configuration**:
4. See *File1* at the end of this tutorial.
5. **Enable the Configuration**:
6. sudo ln -s /etc/nginx/sites-available/reverse-proxy.conf /etc/nginx/sites-enabled/
7. **Test NGINX Configuration**:
8. sudo nginx -t
9. **Restart NGINX**:
10. sudo systemctl restart nginx

Step 5: Set Up SSL (Optional)

1. **Install Certbot**:
2. sudo apt-get install certbot python3-certbot-nginx
3. **Obtain SSL Certificate**:
4. sudo certbot --nginx -d yourdomain.com
5. **Follow the Prompts** to complete the SSL setup.

Step 6: Verify the Setup

1. **Access Your Domain**: Open a web browser and navigate to http://yourdomain.com (or https://yourdomain.com if SSL is set up).
2. **Check Logs**: Monitor NGINX logs for any errors or issues:

sudo tail -f /var/log/nginx/access.log /var/log/nginx/error.log

Port forwarding

You'll need to set up port forwarding on your router to direct HTTP (port 80) and HTTPS (port 443) traffic to the IP address of your reverse proxy. Here are the steps:

1. **Access Your Router's Admin Interface**: Open a web browser and enter your router's IP address (commonly 192.168.1.1 or 192.168.0.1).
2. **Log In**: Enter your router's admin username and password.

3. **Find Port Forwarding Settings**: This is usually under sections like “Advanced,” “NAT,” “Applications and games” or “Port Forwarding.”
4. **Create Port Forwarding Rules**:
   - **HTTP (Port 80)**:
     - **External Port**: 80
     - **Internal IP Address**: IP address of your reverse proxy (e.g., 192.168.0.100)
     - **Internal Port**: 80
   - **HTTPS (Port 443)**:
     - **External Port**: 443
     - **Internal IP Address**: IP address of your reverse proxy (e.g., 192.168.0.100)
     - **Internal Port**: 443
5. **Save and Apply**: Save the settings and apply the changes.
6. **Restart Router**: Sometimes, it’s necessary to restart the router for the changes to take effect.

This will ensure that any incoming HTTP and HTTPS requests are forwarded to your reverse proxy, allowing it to handle the traffic and direct it to the appropriate backend servers.

Certificates renewal

Create a script for renewing certificates with Certbot and set up a cron job to automate the process:

1. Create the Script

1. **Choose a Directory**: Decide where to store the script. A common location is /usr/local/bin.
2. **Create the Script**: Use a text editor to create the script. For example, using nano:
3. sudo nano /usr/local/bin/renew_certs.sh
4. **Add the Script Content**: Paste the following content into the editor:

```
#!/bin/bash

# Stop the reverse proxy server (e.g., NGINX)
```

sudo systemctl stop nginx

# Renew the certificates

sudo certbot renew

# Wait for a specified amount of time (e.g., 60 seconds) to ensure the renewal process completes

sleep 60

# Start the reverse proxy server

sudo systemctl start nginx

Save and Exit: Save the file and exit the editor. In nano, press Ctrl+O to save and Ctrl+X to exit.

2. Make the Script Executable: Ensure the script is executable by running:

```
sudo chmod +x /usr/local/bin/renew_certs.sh
```

3. Set Up the Cron Job

1. **Edit the Crontab**: Open the crontab editor:
2. `crontab -e`
3. **Add the Cron Job**: Add the following line to run the script at 2 AM every day:
4. `0 2 * * * /usr/local/bin/renew_certs.sh`
5. **Save and Exit**: Save the changes and exit the crontab editor.

3. Test the Script

1. **Run the Script Manually**: Execute the script to ensure it works as expected:
2. `sudo /usr/local/bin/renew_certs.sh`
3. **Monitor the Output**: Watch the terminal output to see if the script stops the reverse proxy server, renews the certificates, waits for the specified time, and then restarts the server.
4. **Check the Logs**: Verify the renewal process by checking the Certbot logs:
5. `sudo cat /var/log/letsencrypt/letsencrypt.log`
6. **Verify the Server Status**: Ensure the reverse proxy server is running correctly:

   ```
   sudo systemctl status nginx
   ```

Bind Mount to an LXC Container

1. **\Identify the Container ID**: Find the ID of the LXC container you want to bind mount the folder to. You can list your containers with:

   ```
   pct list
   ```

2. **Create the Mount Point on the Host**: Ensure the directory /mnt/media exists on the host. If it doesn't, create it:

   ```
   sudo mkdir -p /mnt/media
   ```

3. Create the directory in the container.

   ```
   sudo mkdir -p /mnt/media
   ```

4. **Edit the Container Configuration**: Add the mount point to the container's configuration file (/etc/pve/lxc). You can do this using the pct set command on host:

   ```
   pct set container_id -mp0 /mnt/media,mp=/mnt/media
   ```

   Replace ***container_id*** with the actual ID of your container, for example ***103***.

5. **Restart the Container**: Restart the container to apply the changes:

   ```
   pct restart container_id
   ```

creating a Samba shared folder on your Proxmox host:

Create a Samba Shared Folder on Proxmox host

1. **Install Samba**:

   ```
   sudo apt-get install samba
   ```

2. **Configure Samba**:

   - Open the Samba configuration file:

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

   - Add a new share definition at the bottom of the file:

     ```
     [shared_folder]
     path = /mnt/media
     available = yes
     valid users = jon
     read only = no
     browsable = yes
     guest ok = no
     ```

2. **Create a Samba User/password**:

   - Add a new user:
   - sudo adduser *new_samba_user*
   - Set a Samba password for the user:

- sudo smbpasswd -a new_s

You cannot add a user to Samba if the user does not already exist on the system. The *smbpasswd -a* command requires the user to be a valid system user before it can add them to the Samba password database.

sudo adduser new_samba_user command creates a new user account on your Linux system. It sets up the user's home directory, default shell, and other necessary configurations. This user can log into the system and perform tasks based on the permissions granted.

The command *sudo smbpasswd -a new_samba_user* adds the user *new_samba_user* to the Samba password database and prompts you to set a password for this user. This password will be used for authenticating the user when accessing Samba shares. You will see prompts like:

```
New SMB password:

Retype new SMB password:

Added user new_samba_user.
```

On many Linux distributions, when you create a new user, a group with the same name as the user is also created by default. This is a common practice to simplify user and group management.

You can verify the existence of the group by checking the /etc/group file or using the getent command:

Check Using /etc/group

Open the /etc/group file and look for the group:

cat /etc/group | grep *username*

If you see an entry like *username*:x:1000:, it means the group jon exists.

Check Using getent

Use the getent command to check for the group:

getent group *username*

If the group exists, this command will return information about the group. Example Output: jon:x:1000: This indicates that the group jon with GID (Group ID) 1000 exists.

If you need to create a new group, you can do so with the following command:

```
sudo addgroup groupname
```

Then you can add users to this group:

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

This adds the user *username* to the *groupname*.

3. **Set Permissions**:

- Change ownership of the folder:

  ```
  sudo chown -R username:groupname /mnt/media
  ```

  If you didn`t create a group *username* and *groupname* will be the same. For example, sudo chown -R Johnny:johnny /mnt/media

  sudo: This command is used to run programs with the security privileges of another user, by default the superuser (root). It stands for "superuser do."

  chown: This command changes the ownership of files and directories. It stands for "change owner."

  -R: This option stands for "recursive." It means that the command will apply to all files and subdirectories within the specified directory (/mnt/media).

  username:groupname: This specifies the new owner and group for the files and directories. The format is owner:group.

  /mnt/media: This is the path to the directory whose ownership you want to change.

- Set the appropriate permissions:

  ```
  sudo chmod -R 0755 /mnt/media
  ```

  sudo: This command allows you to run programs with the security privileges of another user, by default the superuser (root). It stands for "superuser do."

  chmod: This command changes the file mode (permissions) of a file or directory. It stands for "change mode."

  -R: This option stands for "recursive." It means that the command will apply to all files and subdirectories within the specified directory (/mnt/media).

  0755: This is the permission setting you are applying. It is an octal representation of the file permissions:

  0: No special permissions.

7: Full permissions (read, write, and execute) for the owner.

5: Read and execute permissions for the group.

5: Read and execute permissions for others.

/mnt/media: This is the path to the directory whose permissions you want to change.

2. **Restart Samba**:

   ```
   sudo systemctl restart smbd
   ```

3. **Access the Samba Share**:
   - From a client machine (Any on the local network), open the file explorer.
   - Enter the network path:

     \\192.168.0.25*shared_folder*
   - Enter the Samba username and password when prompted. If the username and password on the VM match the username and password set for Samba access will be granted directly.

4. **Map Network Drive on Windows (**Optional. This will force login with credentials**)**:
   - Open **File Explorer**.
   - Right-click on **This PC** and select **Map network drive**.
   - Enter the network path (e.g., \\192.168.0.25\shared_folder).
   - Check the box “Connect using different credentials”.
   - Click **Finish** and enter the Samba username and password.

Create a Linux VM for multisession

1. **Create a New Virtual Machine**:
   - In the Proxmox web interface, right-click on your node and select 'Create VM'.
   - Assign a unique VM ID and name for your VM.

2. **Configure VM Settings**:
   - **OS**: Select the uploaded Ubuntu ISO as the installation medium.
   - **System**: Choose BIOS or UEFI firmware (SeaBIOS for BIOS, OVMF for UEFI).
   - **Hard Disk**: Select 'VirtIO Block' for the bus/device and allocate the desired disk size.
   - **CPU**: Assign the number of CPU cores and set the type to 'host'.

    - **Memory**: Allocate the desired amount of RAM.
    - **Network**: Choose 'VirtIO' for the network model.

2. **Start the VM and Install Ubuntu**:
    - After configuring the VM, start it and open the console.
    - Follow the Ubuntu installation prompts to complete the setup.

2. **Post-Installation**:
    - Once Ubuntu is installed, you can update the system and install any additional packages you need.

2. **Download ThinLinc Server Bundle:**
    - Go to the ThinLinc website and download the server bundle.

3. **Upload and Extract the Bundle:**
    - Upload the downloaded ZIP file to your Linux VM.
    - Right-click on the ZIP file and select "Extract Here" or "Extract to...".

4. **Start the Installation:**
    - Navigate to the extracted directory.
    - Double-click on the install-server file to start the installation process.

5. **Follow the Installation Wizard:**
    - The installation wizard will guide you through the process. Follow the prompts to install the ThinLinc server package suitable for your system.

6. **Run ThinLinc Setup:**
    - After the installation completes, the wizard will ask if you want to run ThinLinc setup. Confirm to start the setup.
    - Follow the setup prompts to configure ThinLinc as a master or agent. For a standalone server, select 'master'.

7. **Verify Installation:**
    - Check the status of ThinLinc services using the terminal:
    - sudo systemctl status vsmserver
    - sudo systemctl status vsmagent
    - sudo systemctl status tlwebaccess
    - Ensure all services are active and running

Users and their passwords for Thinlinc will be those available on the Linux OS in the VM. An administrator user (admin) will be configured during installation. So in order to create (or remove, or modify) a ThinLinc user, you simply use the same tools as you would for a Linux system user. When running ThinLinc in a cluster configuration, a centralised authentication mechanism such as LDAP or Active Directory is normally used.

8. **Creating a User in Linux:**

Open the Terminal from your applications menu or by pressing Ctrl + Alt + T.

Use the useradd command followed by the username. For example, to create a user named newuser, you would run:

```
sudo useradd newuser
```

To create a home directory for the user, use the -m option:

```
sudo useradd -m newuser
```

Use the passwd command to set a password for the new user:

```
sudo passwd newuser
```

You will be prompted to enter and confirm the password.

9. **Adding the User to a Group**

Use the usermod command with the -aG option to add the user to a group. For example, to add newuser to the sudo group, you would run:

```
sudo usermod -aG sudo newuser
```

You can check the groups a user belongs to with the groups command:

```
groups newuser
```

10. **Check desktop environment**

To check which desktop environments you have installed, run this command:

```
ls /usr/share/xsessions
```

Here are some common examples:

GNOME: gnome.desktop

KDE Plasma: plasma.desktop

Xfce: xfce.desktop

LXDE: LXDE.desktop

MATE: mate.desktop

Cinnamon: cinnamon.desktop

If you have multiple desktop environments installed, you might see several .desktop files listed. Note that if you run a ThinLinc cluster, only the Agents need to have working desktop environments.

One common support issue we get is about the error message **“No executable profiles could be found”** that appears when trying to login to a ThinLinc session.

The most common cause of this is that a session can`t start because no working desktop environment is installed on the system, or the one installed is not among the Thinlinc options. If no desktop is installed, you need to install one through the GUI or using commands in terminal.

For example, to install GNOME, you would run:

```
sudo apt install gnome
```

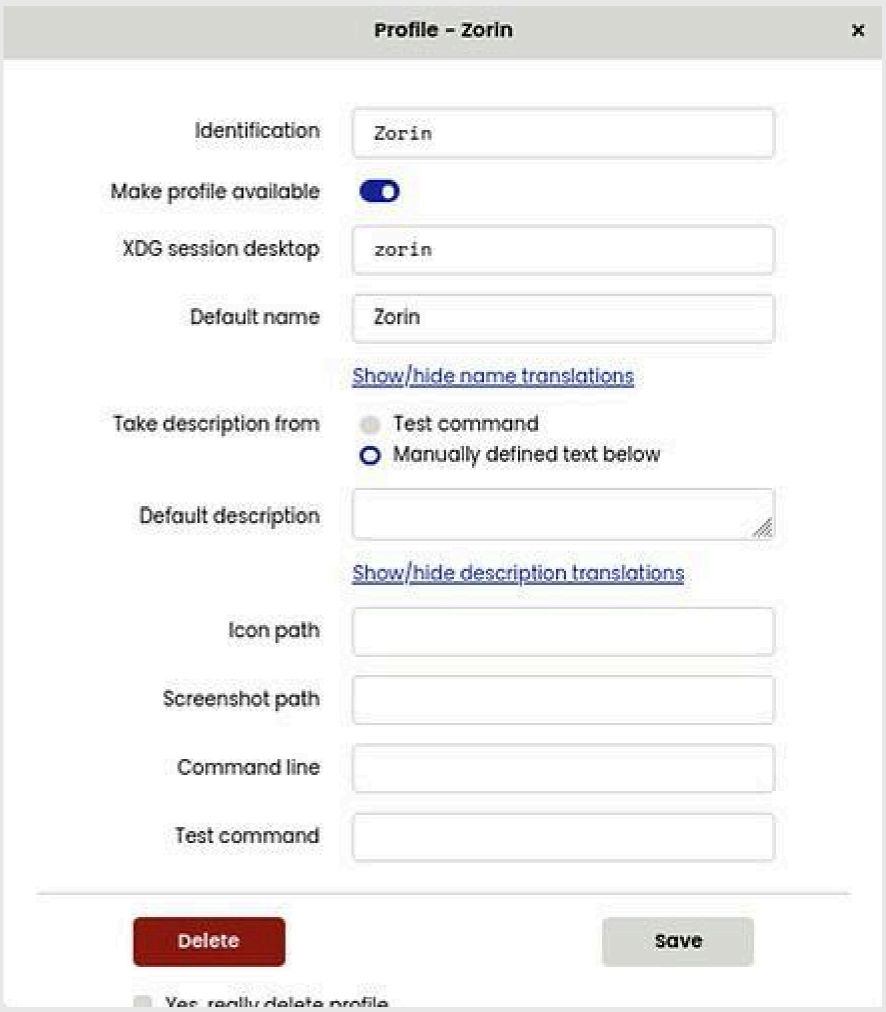
For KDE Plasma, use:

```
sudo apt install kde-plasma-desktop
```

For Xfce, use:

```
sudo apt install xfce4
```

If your desktop is not listed in the Thinlinc server options, go to Profile>Profile list>Configure profile>Add new profile in web administration, and enter the name before the **.desktop** in the 'Identification', 'XDG session desktop' and 'Default name' placeholders. Then press 'save' and try to log in again from client machine. The next example would be for Zorin.desktop output.



Profile - Zorin ×

Identification	Zorin
Make profile available	●
XDG session desktop	zorin
Default name	Zorin
	Show/hide name translations
Take description from	Test command Manually defined text below
Default description	
	Show/hide description translations
Icon path	
Screenshot path	
Command line	
Test command	

Delete   Save

Yes, really delete profile

Access a Samba shared folder from a Linux VM

1. **Install Samba Client**:
    - Install the Samba client package using the following command:

      ```
      sudo apt update
      sudo apt install smbclient cifs-utils
      ```

2. **Create a Mount Point**:
    - Create a directory where you want to mount the shared folder. For example:

      ```
      sudo mkdir /mnt/shared_folder
      ```

2. **Mount the Shared Folder**:
    - Use the mount command to mount the Samba share.
      Replace SHARE_NAME, SERVER_IP, USERNAME,
      and PASSWORD with your actual share name, server IP address, username (for Samba service), and password:

      ```
      sudo mount -t cifs //SERVER_IP/SHARE_NAME /mnt/shared_folder -o
      username=USERNAME,password=PASSWORD
      ```

    - If your Samba share requires a domain, you can add the domain option:
    - sudo mount -t cifs //*SERVER_IP*/*SHARE_NAME* /mnt/sharedfolder -o username=*USERNAME*,password=*PASSWORD*,domain=*DOMAIN*

2. **Verify the Mount**:
    - Check if the shared folder is mounted correctly by listing the contents of the mount point:
    - ls /mnt/*shared_folder*

2. **Automate the Mount (Optional)**:
    - To automatically mount the shared folder at boot, add an entry to the /etc/fstab file. Open the file with a text editor:

      ```
      sudo nano /etc/fstab
      ```

    - Add the following line, replacing the placeholders with your actual details:
    - //*SERVER_IP*/*SHARE_NAME* /*mnt*/*shared_folder* cifs username=*USERNAME*,password=*PASSWORD*,uid=1000,gid=1000 0 0
    - Save and close the file.

--- File1---

```
# Main domain
# This block redirects HTTP requests to HTTPS for the domain example.duckdns.org
server {
    listen 80;
    server_name example.duckdns.org;
    if ($host = example.duckdns.org) {
        return 301 https://$host$request_uri;  # Redirect HTTP to HTTPS
    }                                                                          # managed by
Certbot
    return 404;                                                                # managed by
Certbot
}
# This block handles HTTPS requests for the Proxmox server
server {
    listen 443 ssl;                                                            # managed by
Certbot
    server_name example.duckdns.org;                            # Replace with your domain or IP
address
    # SSL configuration
    ssl_certificate /etc/letsencrypt/live/example.duckdns.org/fullchain.pem;           # managed by
Certbot
    ssl_certificate_key /etc/letsencrypt/live/example.duckdns.org/privkey.pem;         # managed by
Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf;                           # managed by
Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;                             # managed by
Certbot

    location / {
        proxy_pass https://192.168.0.25:8006;                   # Replace with your Proxmox host IP
and port
        proxy_ssl_verify off;                   # Disable SSL verification if Proxmox uses self-signed
certificates
        # Proxy settings
        proxy_set_header Host $host;                                           # Pass the
original Host header
        proxy_set_header X-Real-IP $remote_addr;                               # Pass the real
client IP
```

```
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;            # Pass the client IP
through proxies

        proxy_set_header X-Forwarded-Proto $scheme;                     # Pass the original protocol (http or
https)

        proxy_set_header Upgrade $http_upgrade;                                 # Support WebSocket
connections

        proxy_set_header Connection "upgrade";                                          # Support WebSocket
connections

    }

}



# Jellyfin subdomain

# This block handles HTTPS requests for the Jellyfin server

server {

    listen 443 ssl;

    server_name jellyfin.example.duckdns.org;

    # SSL configuration

    ssl_certificate /etc/letsencrypt/live/jellyfin.example.duckdns.org/fullchain.pem;

    ssl_certificate_key /etc/letsencrypt/live/jellyfin.example.duckdns.org/privkey.pem;

    ssl_protocols TLSv1.2 TLSv1.3;                                                  # Use secure TLS
protocols

    ssl_ciphers
'ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GC
M-SHA384';                                  # Use secure ciphers. These 2 lines are 1 command.

    ssl_prefer_server_ciphers on;                                                       # Prefer server
ciphers

    ssl_dhparam /etc/ssl/certs/dhparam.pem;                         # Optional: Generate for stronger DH
keys

    error_page 497 https://$host$request_uri;

    location / {

        proxy_pass http://192.168.0.3:8096/;                            # Jellyfin's IP and port (8096 by
default)

        # Proxy settings

        proxy_set_header Host $host;                                                    # Pass the
original Host header

        proxy_set_header X-Real-IP $remote_addr;                                        # Pass the real
client IP

        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  # Pass the client IP through
proxies

        proxy_set_header X-Forwarded-Proto $scheme;  # Pass the original protocol (http or https)
        proxy_http_version 1.1;                                                  # Use HTTP/1.1 for WebSocket
support
        proxy_set_header Upgrade $http_upgrade;                                   # Support WebSocket
connections
        proxy_set_header Connection 'upgrade';                                             # Support
WebSocket connections
    }
}
```

```
# Nextcloud subdomain
# This block handles HTTPS requests for the Nextcloud server
server {
    listen 443 ssl;
    server_name nextcloud.example.duckdns.org;
    # SSL configuration
    ssl_certificate /etc/letsencrypt/live/nextcloud.jongfad.duckdns.org/fullchain.pem;   # not placed by
Certbot.
        Both are possible. Certbot always places certs in correct folder but only add lines to config file if
specified.
    ssl_certificate_key /etc/letsencrypt/live/nextcloud.example.duckdns.org/privkey.pem;
    ssl_protocols TLSv1.2 TLSv1.3;                                                  # Use secure TLS
protocols
    ssl_ciphers
'TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:ECDHE-RSA-AES128-GCM-SHA256:EC
DHE-RSA-AES256-GCM-SHA384';                    # Use secure ciphers. These 2 lines are 1 command.
    ssl_prefer_server_ciphers on;                                                      # Prefer server
ciphers

    location / {
        proxy_pass https://192.168.0.32:443;                # Replace with the internal IP of your Nextcloud
container
```

```
        # Proxy settings
        proxy_set_header Host $host;                                    # Pass the original Host header
        proxy_set_header X-Real-IP $remote_addr;                        # Pass the real client IP
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  # Pass the client IP through proxies
        proxy_set_header X-Forwarded-Proto https;                       # Pass the original protocol (https)
    }
}
```

---End Of File---