```
       _ _         _   _ _
 _ __ (_) |___ _ ___| |_(_) |__
| '  \| | / / '_/ _ \  _| | / /_
|_|_|_|_|_\_\_| \___/\__|_|_\_(_)
                          | |
 _  _ _ __  __ _ _ _ __ _ __| |___
| || | '_ \/ _` | '_/ _` / _` / -_)_
 \_,_| .__/\__, |_| \__,_\__,_\___(_)
     |_|   |___/
 ___ ___ _ ___ _____ _ _
(_-</ -_) '_\ V / -_) '_|
/__/\___|_|  \_/\___|_|
```

Documentation

Content:

Disclaimer:

1.Introduction:

Using the popular devices from Mikrotik® gives at some point the need to update/upgrade these devices. This is usually done by clicking on UPGRADE PACKAGES in the Web-Frontend or in the WINBOX®-client. After that the newest version will be checked and the update/upgrade will be performed. Due to the huge popularity of these devices, an update of one or two devices will be done quite fast, but if someone is in the situation to update a larger amount of devices, this will take some time.

This is on one hand due to the complete download of the packages through the internet and also that, when someone has a large amount of devices, suddenly some kind of fairness-download-speed-limit comes in action. I observed that after the fifth (or more) device-updates: the speed of the download will be decreased. We must keep in mind that when a new release becomes available, everyone will be try to download it. Fairly we must understand that this peak in update-sessions could not be handled in a very fast way.

So we must accept that Mikrotik® could not prepare and hold additional download-capacity to handle this peak in a affordable manner for all the customers. This is definitely no fault or error by Mikrotik®. In that situation you cannot solve this problem in a reasonable way with no extra (and exploding) costs.

So I decided that it will be fine to run a local copy of some sort of download/update/upgrade-service in my network(s). It would be nice to have all packages needed for an extension of the function of the device (LTE/Wireless/LoRa/ROSE-Storage etc.) on a local repository. This brought me to the development of this software.

There are several approaches to hold all needed packages for an installation/update/upgrade locally on a device directly, but in my point of view they are complicated. I thought of a simple installation of a local server, that is updating itself in the background (nightly) and gives the ability to download the actual packages from a web-frontend and also to be available inside the RouterOS® for doing the packages updates/upgrades. But it would be nice to not have to have an additional server-installation in the network, furthermore to integrate it directly in some powerful Mikrotik®-device. I decided to use the CONTAINER-function in RouterOS® available roughly from release 7.5 on.

Using the small Linux-distribution Alpine Linux and so very basic tools inside, this container is very small at first (about 40 Mbytes). Also there are no special programming-tools or languages that are used inside the container. Only using the basic Linux-environment, a webserver and some small additional tools installable via the package manager of the Linux-distro makes the size of that container also very small.

The container itself contains during and after install no packages from the download-area of the Mikrotik®-servers and is completely self-configuring. After a fresh install of the container it will be starting to download all the packages, that are available currently from the master-server based at Mikrotik®. Then the system will check each night, if there is a change in the releases and will download the fresh versions of the files/packages. The last packages downloaded before will be stay there, as long as the container is not deleted and freshly installed. Also using the persistent storage function in the container, all packages will stay permanently even after an upgrade or fresh install of this software. So someone cloud build an archive of past and currently releases with this software.

Using the container-technology makes this software to be nearly a “one-clicker” to run and serve the files/packages to the Mikrotik®-devices in the network.

All preparation and installation steps with additional information about the usage will be described later in this manual.

So at the end the reader will be asking “what is the name of that project/software ?”.
Well, the name is “**mikrotik.upgrade.server**” or short “**mus**”.

I hope that this software will be useful for someone and will speed up your updates/upgrades a little bit. If this is true, I am completely satisfied. If not, sorry for taking your time.

Regards, Detlef

Here is a rough overview of the following steps in short:

1. Install CONTAINER-package and enable CONTAINER-function on the device
2. Create one or several VETH-devices for the container(s) and give them IP-addresses
3. Create a bridge (also with IP-address) for these VETH-devices and the container(s) running on
4. Prepare the FIREWALL, NAT and the PORT-FORWARDING to reach the container(s) from the outside networks (from the view outside the container-bridge)
5. Configure the container with the used VETH-device, the image to be downloaded and some other configurations like DNS, start-on-boot, logging etc.
6. Apply the container to the device
7. Start the container
8. Try to reach the web-interface
9. On access, please wait during the self-configuration and the download of the files/packages.
10. Use the “mus” on your network – have fun

Steps 1 to 4 are regarding to preparation.
All later steps are described under topic 3. “Installation of the mikrotik.upgrade.server”

As additional steps the installation of a persistent container storage will be described also under topic 3.

2.Preparation of the Mikrotik®-device:

Using the “mikrotik.upgrade.server” (or from now on in short form “mus”) needs a Mikrotik®-device with the CONTAINER-function enabled. This makes only x86_64/ARM/ARM64-based devices usable for the “mus”. Other architectures are currently not capable to run a container-image inside the devices.

Also a affordable size of memory (RAM) and some sort of (relatively fast) disk (with enough disk space on it) is needed. Devices with a amount of >256 Mbytes of RAM and a disk (usb/network/local on CHR) with minimum 16 GB are needed.

Devices that support the ROSE-storage-package will give the availability of running the disk as a iSCSI/SMB etc. share. Especially CHR (virtualized installations) will make use of a local disk in the hypervisor or virtual environment. This is the recommended storage for a fast and convenient installation.

Several details will be described in the later manual as they appear.

Preparation in detail:

Citation from the Mikrotik®-Wiki found here:
Device-mode limits container use by default, before granting container mode access - make sure your device is fully secured.

First of all the container-function must be enabled on the device. If you are using WINBOX® as your configuration client, you will see “Container” in the left sidebar.
(Using WEBFIG via a browser will be slightly different, but nearly the same,)

If not, you have to enable this mode as following:

Please make sure that the CONTAINER-package is installed on your device!

Open a console (CLI) using “NEW TERMINAL” on the left sidebar and type in:

```
/system/device-mode/update container=yes
```

After that you must restart the device in the given time without shutting it down.
This means that you have to reset the device via the reset-button or reset the device via the hypervisor/virtual environment (cold-reboot/reset). Do not restart it or shut it via the WINBOX®-Client. This means you have to restart the device the “hard way”, which normally should not be done.

After successfully activating the CONTAINER-mode the following printout should be seen on the “NEW TERMINAL”:

```
system/device-mode/print
```

This is the expected output :

```
mode: enterprise
container: yes
```

Important hint:

Enabling the container mode and running containers on the device could be harmful if the device is not properly secured (strong passwords & correct firewall configuration etc).
Anyone who gets access to the device could get root-access to the whole system.
You have been warned !

Now you need to add some virtual interfaces to the device which can be used by the containers. It is advisable to define a new network segment for the containers and the needed bridge. The network-bridge is needed if you want to run more than one container on the device, but I strongly advise to use it. Every container needs one unique VETH-interface and a unique IP-address. This configuration can be done via the CLI (“NEW TERMINAL”) or via the WINBOX®-client.

Here is the syntax for the CLI:

```
/interface/veth/add name=veth_docker01 \
address=10.10.10.11/24 gateway=10.10.10.1
```
(the \ means to put all content in one line !)

Now create a bridge for the container-veth’s:

```
/interface/bridge/add name=bridge_containers
```

```
/ip/address/add address=10.10.10.1/24 interface=bridge_containers
```

```
/interface/bridge/port add bridge=containers interface=veth_docker01
```

After that add a NAT rule to give the containers access to the outside:

```
/ip/firewall/nat/add chain=srcnat action=masquerade src-
address=10.10.10.0/24
```

To reach the container from the outside network we must add several DST-NAT rules.
The exposed ports are 80/tcp for the webserver and optionally 22/tcp for accessing the container via a SSH-client.

Informational hint:
The access of the container is available via port 80 (webserver) and port 22 (SSH-client). The webserver-port is ready to use. SSH-access is possible, but if you do not set a password actively, access via ssh is NOT possible. There is no default password set in the container. Currently the user `root` is used to access the container. Known as a high security risk to access via root-user this will be changed in a future version.

Not setting a DST-NAT-rule for the service SSH in the firewall may be some kind of security-hardening, because if a service is not reachable from the outside network no frauding access will be possible. Please keep in mind, that a proper firewall-configuration is eminent for a secure use of the device!

Here is the command to set up the DST-NAT-rule for the webserver-access via CLI:

```
ip/firewall/nat/add chain=dstnat action=dst-nat dst-address=<DEVICE-IP> \
dst-port=80 \ protocol=tcp to-addresses=10.10.10.11 to-ports=80
```
(the \ means to put all content in one line !)

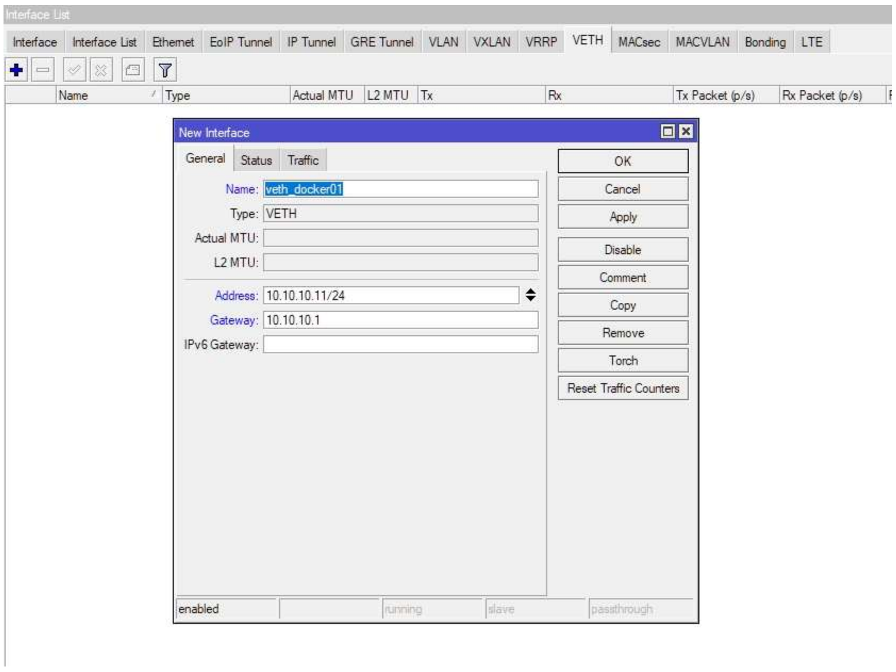
The optional command for setting up the DST-NAT-rule to SSH-access looks like this:

```
ip/firewall/nat/add chain=dstnat action=dst-nat dst-address=<DEVICE-IP> \
dst-port=22022 \ protocol=tcp to-addresses=10.10.10.11 to-ports=22
```
(the \ means to put all content in one line !)

The dst-port=22022 is only an example. Please set it up regarding your preferences.
It is advised that you refer to the basic configuration of the device-firewall for a complete understanding how these configurations work.

It has to be mentioned here that all commands described before could also be setup via the WINBOX®-client. Refer to the following screenshots:

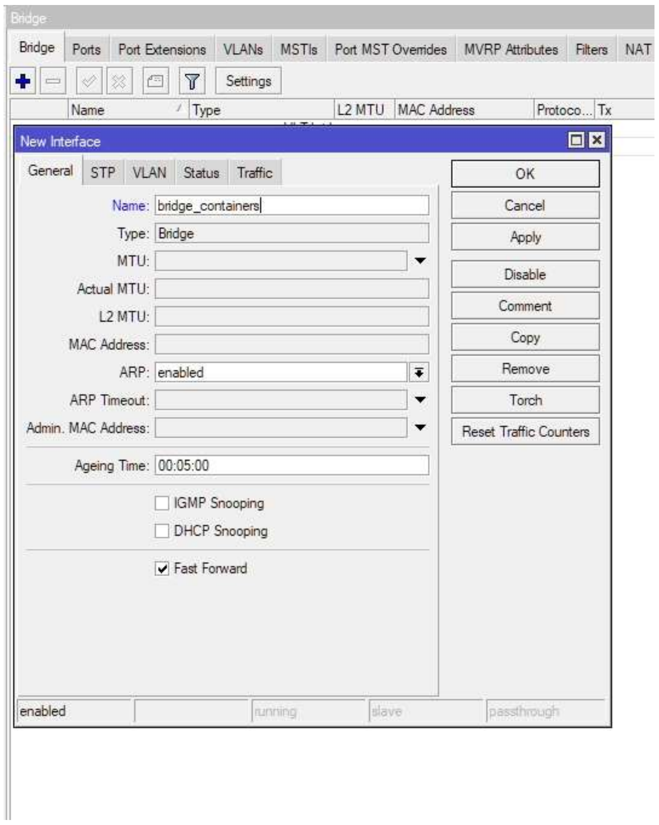
Adding VETH-interface:



New Interface

Name: veth_docker01
Type: VETH
Actual MTU:
L2 MTU:
Address: 10.10.10.11/24
Gateway: 10.10.10.1
IPv6 Gateway:

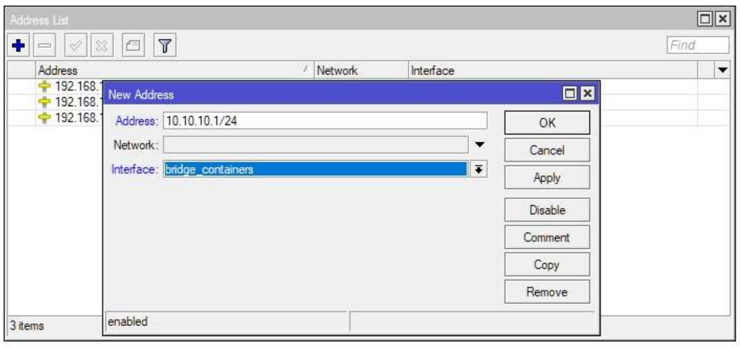
Adding bridge:



New Interface

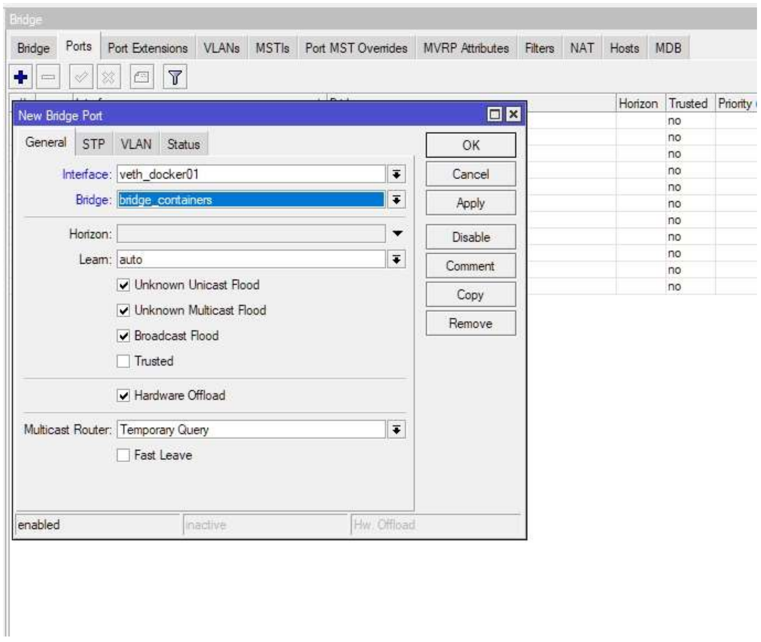
Name: bridge_containers
Type: Bridge
MTU:
Actual MTU:
L2 MTU:
MAC Address:
ARP: enabled
ARP Timeout:
Admin. MAC Address:
Ageing Time: 00:05:00
☐ IGMP Snooping
☐ DHCP Snooping
☑ Fast Forward

Adding IP to bridge:



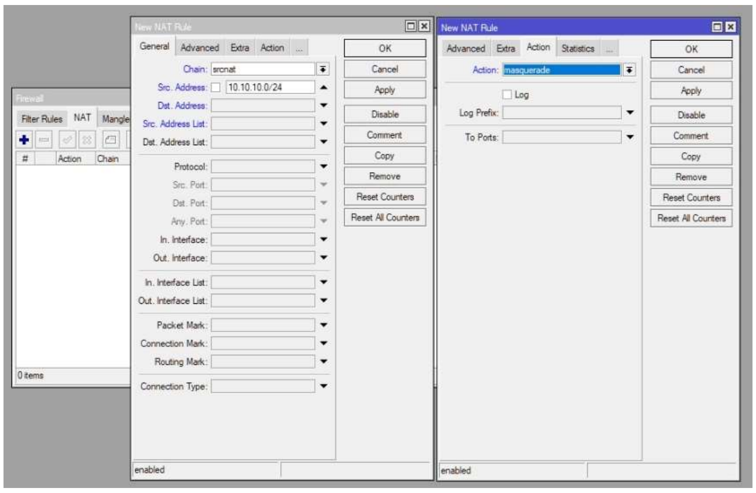
New Address
Address: 10.10.10.1/24
Network:
Interface: bridge_containers
enabled

Adding VETH-interface to bridge:



New Bridge Port
General
Interface: veth_docker01
Bridge: bridge_containers
Horizon:
Learn: auto
☑ Unknown Unicast Flood
☑ Unknown Multicast Flood
☑ Broadcast Flood
☐ Trusted
☑ Hardware Offload
Multicast Router: Temporary Query
☐ Fast Leave
enabled

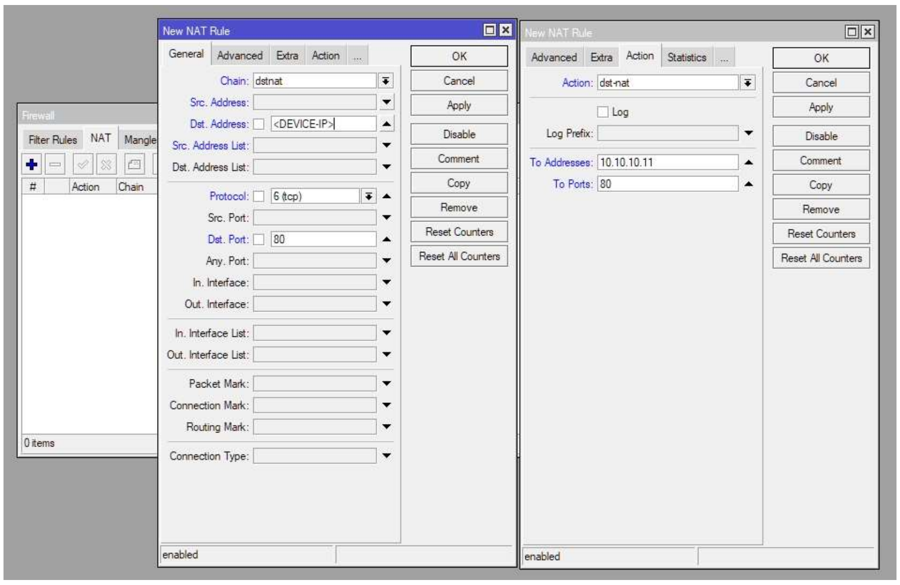
Setup container-masquerading:



New NAT Rule
General
Chain: srcnat
Src. Address: ☐ 10.10.10.0/24
Dst. Address:
Src. Address List:
Dst. Address List:
Protocol:
Src. Port:
Dst. Port:
Any. Port:
In. Interface:
Out. Interface:
In. Interface List:
Out. Interface List:
Packet Mark:
Connection Mark:
Routing Mark:
Connection Type:
enabled

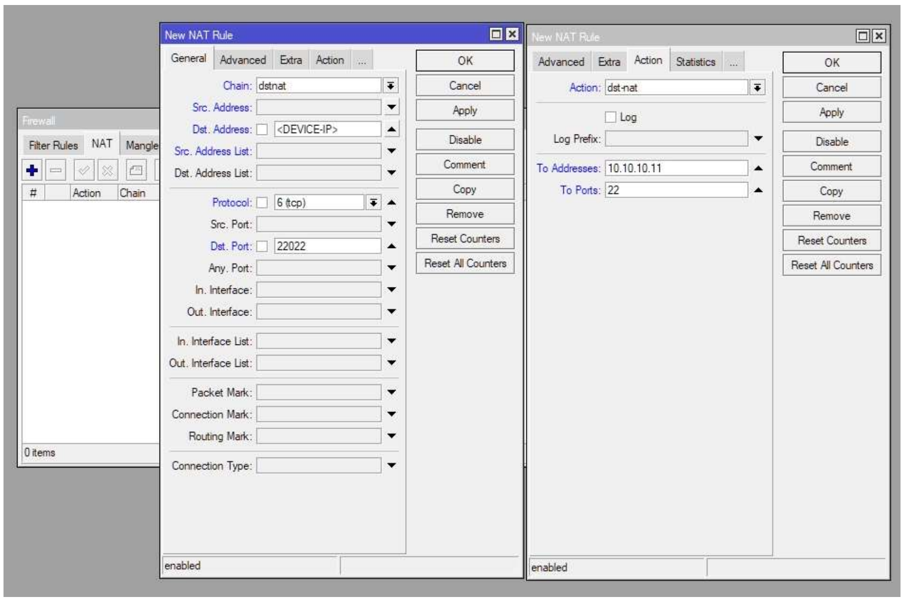
New NAT Rule
Action
Action: masquerade
☐ Log
Log Prefix:
To Ports:
enabled

DST-NAT rule for webserver:



Chain: dstnat
Dst. Address: <DEVICE-IP>
Protocol: 6 (tcp)
Dst. Port: 80
Action: dst-nat
To Addresses: 10.10.10.11
To Ports: 80

DST-NAT rule for ssh-access:



Chain: dstnat
Dst. Address: <DEVICE-IP>
Protocol: 6 (tcp)
Dst. Port: 22022
Action: dst-nat
To Addresses: 10.10.10.11
To Ports: 22

<u>Hint:</u>
<u>For informational purposes the two pictures above show two windows when adding the rules. In a real environment there is only one window, with the “GENERAL” and “ACTION” tab.</u>

At this point all needed (minimal) settings for running the container should be setup.

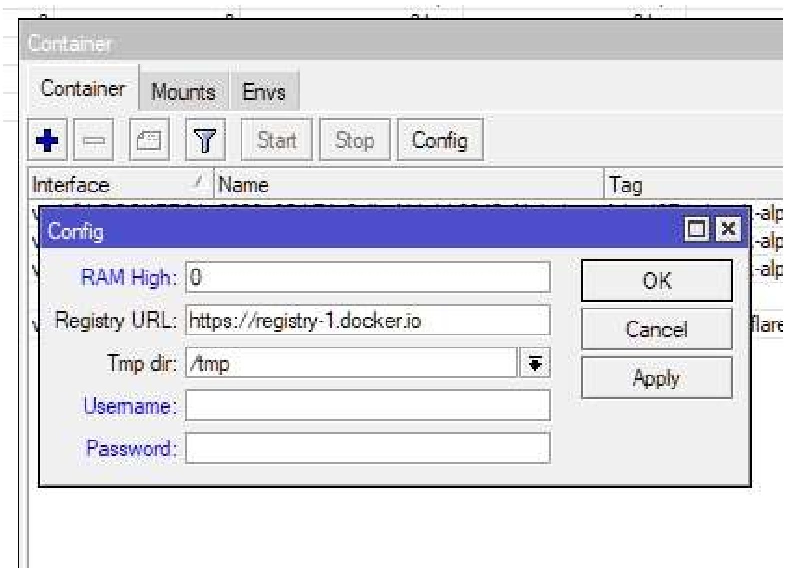
Next is the description to install the container itself on the device.

3.Installation of the mikrotik.upgrade.server:

The “mus” is based on a (docker)-container, which must be installed on the device. Hosted is this image on docker hub, from which it has to be downloaded to the device. Prior the downloading process, there are several configurations to be made on the MIKROTIK®-device. The minimal needed configuration are setting up the “Registry-URL” and the “Tmp dir”. There is a download limit in action on “Docker HUB”, perhaps you need to setup a account on this repository to prevent this. You have to fill in the “Username” and “Password”. If you are running on a device with less memory you could set up a memory limit with “RAM high”. It has to be filled with the bytes-value maximum available for the containers running on the device.

Here is a screenshot from the WINBOX®-client. Using it is little bit more handy than the CLI.

Container base configuration:



RAM High: 0
Registry URL: https://registry-1.docker.io
Tmp dir: /tmp
Username:
Password:

Hint:
All images which I do provide are publish public and could be download without any account.

The next step is to setup and download the container.
First you click on the “+”-sign. The “Interface” will be filled out automatically. If not than you had forget to setup up a VETH-interface like described before. If you set up more than one VETH-interface, you have to choose the right one.
The “Remote Image” will be filled with the following content:

```
felted67/mikrotik-alp_rc_upgrade-server:latest
```

This is the link to the latest, stable version of the “mikrotik.upgrade.server”.
Using other dev-versions needs the links shown on docker hub.

Then you need to setup some “Hostname” and “Domain Name”. Fill it like your preferences.

A needed configuration is the “Root Dir”. When using the internal storage of the device (if possible) or a linked datastorage (disk) on a CHR-installation, you have to make sure that the directory exists. Check it with the “Files”-tab on the left sidebar.

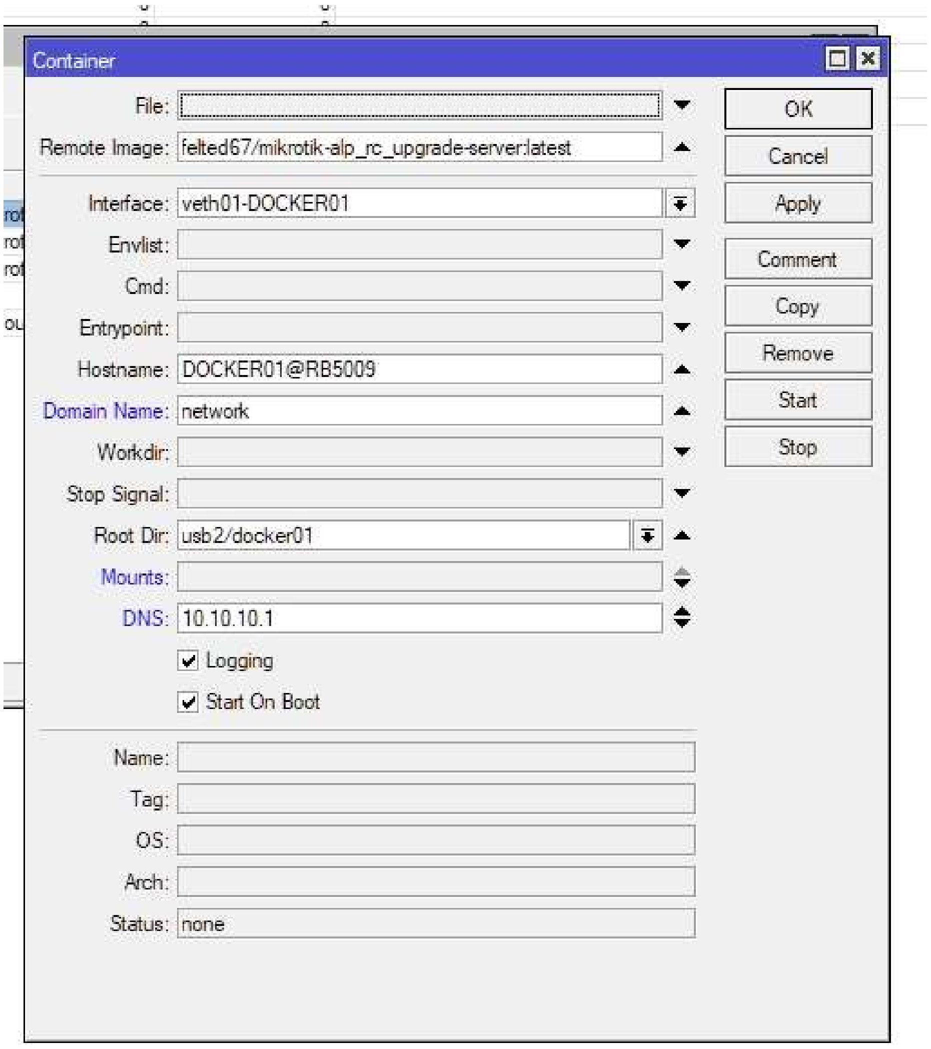
Hint:
Using a NFS-share needs proper configuration of the share in the NFS-server, because some NFS-shares don’t allow to change the user-owner in the share. Direct storage on USB-devices, mounted SMB-share and others works like a charm.

Another needed configuration is the DNS-entry. In our example you have to set this to the IP of the “bridge_containers”, like set before. Also make sure that the firewall allows to access the DNS-server in the device on this IP-address.

Enabling “Logging” will give important information on starting the container. You will find them also in the left sidebar under “Log”.

Setting “Start on boot” will allow to start the “mus” even on reboot of the device.

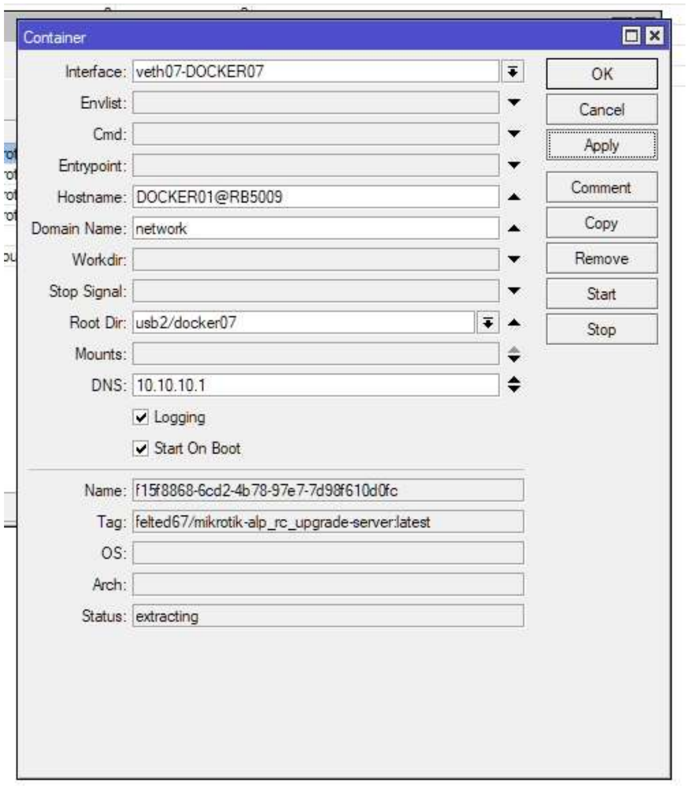
An example of these settings looks like this:



File:	
Remote Image:	felted67/mikrotik-alp_rc_upgrade-server:latest
Interface:	veth01-DOCKER01
Envlist:	
Cmd:	
Entrypoint:	
Hostname:	DOCKER01@RB5009
Domain Name:	network
Workdir:	
Stop Signal:	
Root Dir:	usb2/docker01
Mounts:	
DNS:	10.10.10.1
	✓ Logging
	✓ Start On Boot
Name:	
Tag:	
OS:	
Arch:	
Status:	none

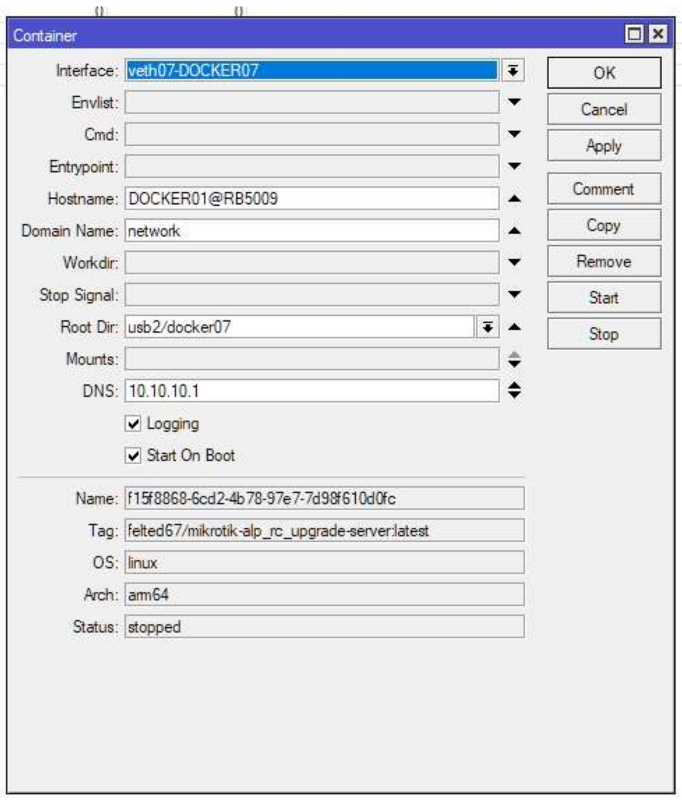
Now you can click on “Apply”. The container will be downloaded and set up.

Here is a picture of this process:



Interface: veth07-DOCKER07
Envlist:
Cmd:
Entrypoint:
Hostname: DOCKER01@RB5009
Domain Name: network
Workdir:
Stop Signal:
Root Dir: usb2/docker07
Mounts:
DNS: 10.10.10.1
☑ Logging
☑ Start On Boot
Name: f15f8868-6cd2-4b78-97e7-7d98f610d0fc
Tag: felted67/mikrotik-alp_rc_upgrade-server:latest
OS:
Arch:
Status: extracting

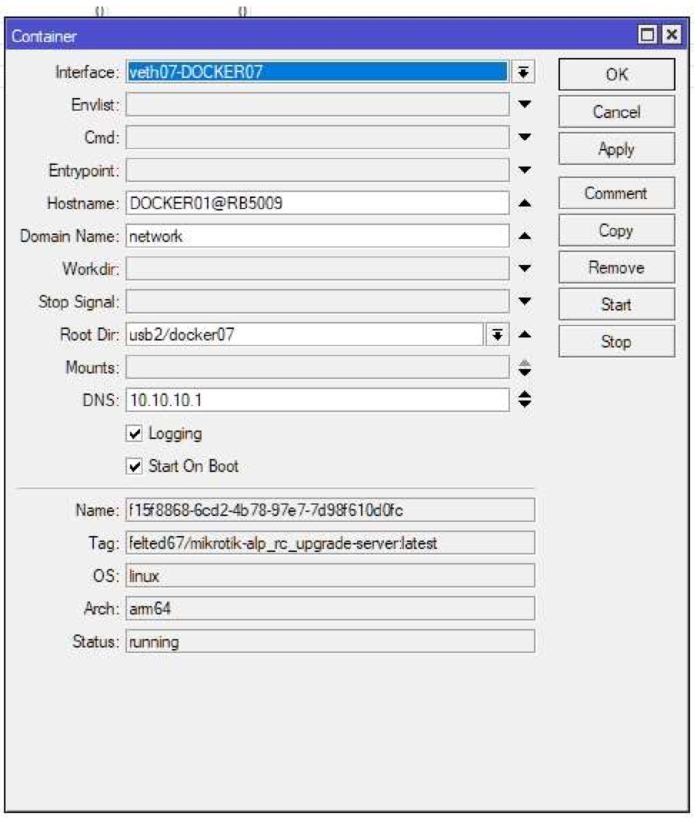
This process takes some time. A completed installation looks like this:



Interface: veth07-DOCKER07
Envlist:
Cmd:
Entrypoint:
Hostname: DOCKER01@RB5009
Domain Name: network
Workdir:
Stop Signal:
Root Dir: usb2/docker07
Mounts:
DNS: 10.10.10.1
☑ Logging
☑ Start On Boot
Name: f15f8868-6cd2-4b78-97e7-7d98f610d0fc
Tag: felted67/mikrotik-alp_rc_upgrade-server:latest
OS: linux
Arch: arm64
Status: stopped

To start the “mus” please click on “Start”. The “Status” will change to running.

Check out this screenshot:



Interface:	veth07-DOCKER07
Envlist:	
Cmd:	
Entrypoint:	
Hostname:	DOCKER01@RB5009
Domain Name:	network
Workdir:	
Stop Signal:	
Root Dir:	usb2/docker07
Mounts:	
DNS:	10.10.10.1
	✓ Logging
	✓ Start On Boot
Name:	f15f8868-6cd2-4b78-97e7-7d98f610d0fc
Tag:	felted67/mikrotik-alp_rc_upgrade-server:latest
OS:	linux
Arch:	arm64
Status:	running

Please wait some time and watch the status. If everything is running fine, the status will stay on “running”.

When the status changes to “stopped” please check out the “Log”-tab in the left sidebar and the log-files for errors.

The “mus” will run as long you stop the container or you reboot you device. And remember: if you clicked “Start On Boot” it will restart automatically if you reboot the device.

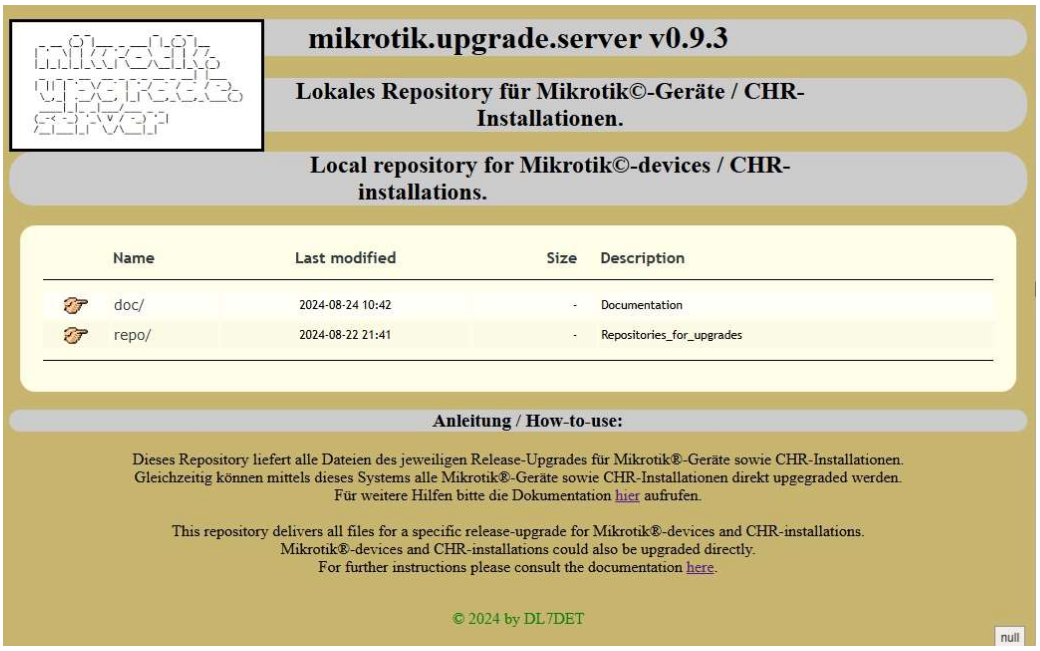
Next we do a first test of the running “mikrotik.upgrade.server” described on the next page.

First test after installation:

To test to everything gone right, please use a web-browser like “Firefox” or “Google Chrome” (or something similar) and open the address you configured in the firewall DST-NAT-configuration. The webserver is running on the IP-address and port 80.

Please open: http://<IP-you-configured>

The first view looks like this:



mikrotik.upgrade.server v0.9.3

Lokales Repository für Mikrotik©-Geräte / CHR-Installationen.

Local repository for Mikrotik©-devices / CHR-installations.

Name	Last modified	Size	Description
doc/	2024-08-24 10:42	-	Documentation
repo/	2024-08-22 21:41	-	Repositories_for_upgrades

Anleitung / How-to-use:

Dieses Repository liefert alle Dateien des jeweiligen Release-Upgrades für Mikrotik®-Geräte sowie CHR-Installationen.
Gleichzeitig können mittels dieses Systems alle Mikrotik®-Geräte sowie CHR-Installationen direkt upgegraded werden.
Für weitere Hilfen bitte die Dokumentation hier aufrufen.

This repository delivers all files for a specific release-upgrade for Mikrotik®-devices and CHR-installations.
Mikrotik®-devices and CHR-installations could also be upgraded directly.
For further instructions please consult the documentation here.

© 2024 by DL7DET

When you see a webpage like this all is running fine. Clicking on the link “doc” will give the chance to open the current documentation like this one.

Choosing the link “repo” will contain in the beginning nearly nothing, because the “mus” is downloading the latest version of the packages for routeros and the WINBOX®-software in the background. Depending on your internet-speed this may take a while.

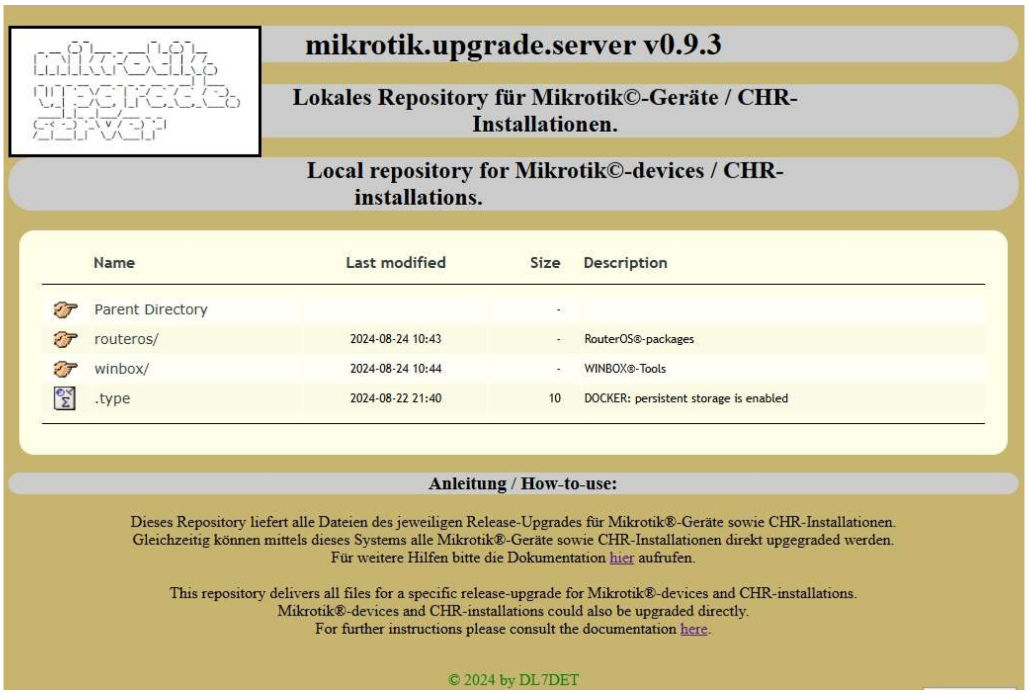
At first the WINBOX®-packages will be available. After that the routeros-directory will be filled.

Hint:
Currently we set up a container running the “mus” the “simple” way. That means all data and configurations inside the container will live as long as the container will exist. If you click “Remove” all data and configurations will be lost. You could setup up “Mounts” and make the data and configurations of the “mus” to survive even on a complete reinstallation (including a “Remove”)! This will be described in a later topic.

Some words to this example shown above. This example was tested on a RB5009-device from MIKROTIK®. You might recognize the “Arch:” as arm64 as the RB5009 is a ARM64-based device. The “Root Dir” is set to “usb2/docker02” which means there is a

USB-stick inserted in the device with a directory set to “docker07”. It is advisable to setup unique directories for every container you are running on the device.

Here is a picture of the repo-directory:



mikrotik.upgrade.server v0.9.3

Lokales Repository für Mikrotik©-Geräte / CHR-Installationen.

Local repository for Mikrotik©-devices / CHR-installations.

Name	Last modified	Size	Description
Parent Directory		-	
routeros/	2024-08-24 10:43	-	RouterOS®-packages
winbox/	2024-08-24 10:44	-	WINBOX®-Tools
.type	2024-08-22 21:40	10	DOCKER: persistent storage is enabled

Anleitung / How-to-use:

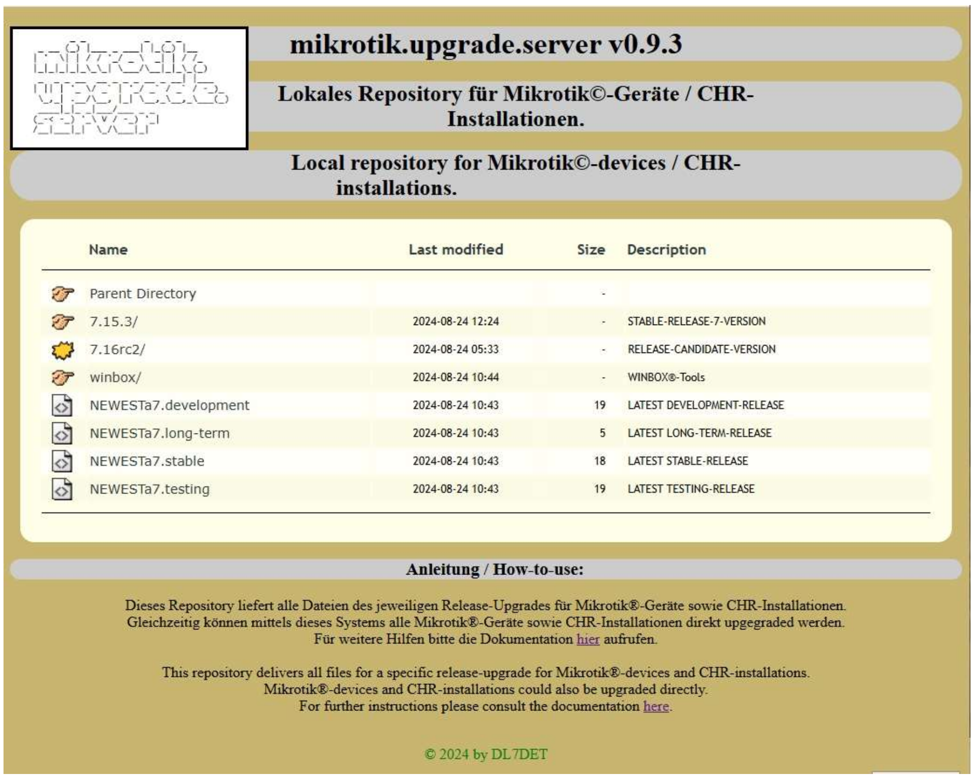
Dieses Repository liefert alle Dateien des jeweiligen Release-Upgrades für Mikrotik®-Geräte sowie CHR-Installationen.
Gleichzeitig können mittels dieses Systems alle Mikrotik®-Geräte sowie CHR-Installationen direkt upgegraded werden.
Für weitere Hilfen bitte die Dokumentation hier aufrufen.

This repository delivers all files for a specific release-upgrade for Mikrotik®-devices and CHR-installations.
Mikrotik®-devices and CHR-installations could also be upgraded directly.
For further instructions please consult the documentation here.

© 2024 by DL7DET

Perhaps you have recognized the “.type”-entry at the bottom of the screen.
This will be shown, if a “persistent mount” is used for storing the data and configuration.
Configuring this will be described later.

The “routeros”-directory will be filled – as described before – after some time and may look like as the following:



mikrotik.upgrade.server v0.9.3

Lokales Repository für Mikrotik©-Geräte / CHR-Installationen.

Local repository for Mikrotik©-devices / CHR-installations.

Name	Last modified	Size	Description
Parent Directory		-	
7.15.3/	2024-08-24 12:24	-	STABLE-RELEASE-7-VERSION
7.16rc2/	2024-08-24 05:33	-	RELEASE-CANDIDATE-VERSION
winbox/	2024-08-24 10:44	-	WINBOX®-Tools
NEWESTa7.development	2024-08-24 10:43	19	LATEST DEVELOPMENT-RELEASE
NEWESTa7.long-term	2024-08-24 10:43	5	LATEST LONG-TERM-RELEASE
NEWESTa7.stable	2024-08-24 10:43	18	LATEST STABLE-RELEASE
NEWESTa7.testing	2024-08-24 10:43	19	LATEST TESTING-RELEASE

Anleitung / How-to-use:

Dieses Repository liefert alle Dateien des jeweiligen Release-Upgrades für Mikrotik®-Geräte sowie CHR-Installationen.
Gleichzeitig können mittels dieses Systems alle Mikrotik®-Geräte sowie CHR-Installationen direkt upgegraded werden.
Für weitere Hilfen bitte die Dokumentation hier aufrufen.

This repository delivers all files for a specific release-upgrade for Mikrotik®-devices and CHR-installations.
Mikrotik®-devices and CHR-installations could also be upgraded directly.
For further instructions please consult the documentation here.

© 2024 by DL7DET

There is an entry with “7.15.3” which is the current stable version. The “7.16.rc2” stands for the actual release-candidate. Both directories contain all files available for that release plus some extra files needed for further updates.

The “winbox” entry will be shown twice (also on the “repo”-directory). This is for convenience reasons to reach out the WINBOX®-packages in a fast way.

The additional “NEWEST7.*”-entries are used for informational purposes, but also needed for the internal system to checkout the needed package-version for downloading them. These ones are the only (!) information from the MIKROTIK® master-servers, which need to be downloaded to create the current directories with the actual versions.

All other needed configurations are derived from this files and will end in generating automatically the download links and to execute them.

In the default installation now the “mus” will check out every night at 0:00 h UTC for new versions and will download them. All previous versions will stay there and will not be overwritten. This make the “mus” also useable as an archive-server when running several versions of the packages in a non-homogenous network.

Keep in mind that all data will be lost, if the container is removed and recreated. Use the “Mount”-feature described later for overcoming this behavior.

At this point the first main function of the “mikrotik.upgrade.server” has been set up and is running now = a local web repository hosting all actual packages.

Another very useful function is now that the “mus” is able to work as an upgrade source directly from the WINBOX®- or WEBFIG-client to apply all update in a very easy way.

This will be described in the next topic following now.