



Documentation

Content:

1. Introduction	3
2. Preparation of the Mikrotik-device	5
3. Installation of the mikrotik.upgrade.server	11
4. Usage of the mikrotik.upgrade.server	19
5. Technical background of the project	28
6. Addendum	30

Disclaimer:

MIKROTIK®, WinBox®, RouterOS, ROS®, hap x2, hap x3, RB5009 and others are or may be trademarks or registered names of SIA Mikrotīkls.

This project is not affiliated with SIA Mikrotīkls and SIA Mikrotīkls is not responsible for this project. Link: <https://mikrotik.com/aboutus>

All names, trademarks or other techniques are only used to illustrate this project.

There is no responsibility for any faults, errors, defects and harm regarding using these images and/or this software.

This is a private project and all information stated here are given as it is and with no responsibility for any defects, errors and harm using these images and/or this software.

Alpine Linux is copyrighted by the Alpine Linux Development Team with all rights reserved.

Also all names and symbols from Alpine Linux are used for illustration purposes only with no responsibility of the Alpine Linux Development Team. Link: <https://www.alpinelinux.org/>

This software is released under the MIT license.

The complete text of this license can be found on the following page.

MIT-License:

Copyright © 2024 Detlef Lampart / DL7DET

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Copyright © 2024 Detlef Lampart / DL7DET

Jedem, der eine Kopie dieser Software und der zugehörigen Dokumentationsdateien (die „Software“) erhält, wird hiermit kostenlos die Erlaubnis erteilt, ohne Einschränkung mit der Software zu handeln, einschließlich und ohne Einschränkung der Rechte zur Nutzung, zum Kopieren, Ändern, Zusammenführen, Veröffentlichen, Verteilen, Unterlizenzieren und/oder Verkaufen von Kopien der Software, und Personen, denen die Software zur Verfügung gestellt wird, dies unter den folgenden Bedingungen zu gestatten:

Der obige Urheberrechtshinweis und dieser Genehmigungshinweis müssen in allen Kopien oder wesentlichen Teilen der Software enthalten sein.

DIE SOFTWARE WIRD OHNE MÄNGELGEWÄHR UND OHNE JEGLICHE AUSDRÜCKLICHE ODER STILLSCHWEIGENDE GEWÄHRLEISTUNG, EINSCHLIEßLICH, ABER NICHT BESCHRÄNKT AUF DIE GEWÄHRLEISTUNG DER MARKTGÄNGIGKEIT, DER EIGNUNG FÜR EINEN BESTIMMTEN ZWECK UND DER NICHTVERLETZUNG VON RECHTEN DRITTER, ZUR VERFÜGUNG GESTELLT. DIE AUTOREN ODER URHEBERRECHTSINHABER SIND IN KEINEM FALL HAFTBAR FÜR ANSPRÜCHE, SCHÄDEN ODER ANDERE VERPFLICHTUNGEN, OB IN EINER VERTRAGS- ODER HAFTUNGSKLAGE, EINER UNERLAUBTEN HANDLUNG ODER ANDERWEITIG, DIE SICH AUS ODER IN VERBINDUNG MIT DER SOFTWARE ODER DER NUTZUNG ODER ANDEREN GESCHÄFTEN MIT DER SOFTWARE ERGEBEN.

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 number of devices, this will take some time.

This is on one hand due to the separate download of the packages for each device through the internet and also that, when someone has a large number 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 try to download it. Fairly we must understand that this peak in update-sessions could not be handled in a very fast way.

We have to accept that MIKROTIK® could not prepare and hold additional download-capacity to handle this peak in an 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). 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 before and during the installation 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 could 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 precious time.

Regards, *Detlef*

*This project is dedicated to Doris Lampart, who listened to all my thoughts during development, even if she didn't understand anything I was talking about.
Thank you and with all my love!*

Here is a rough installation-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 chapter 3. “Installation of the mikrotik.upgrade.server”

As additional steps the installation of a persistent container storage will be described also under chapter 4.

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.

An affordable size of memory (RAM) and some sort of (relatively fast) disk (with enough disk space on it) is needed. Devices with an 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 an 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:

<https://help.mikrotik.com/docs/display/ROS/Container>

“Device-mode limits container use by default, before granting container mode access - make sure your device is fully secured.”

First 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 like the 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 fraudulent 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 !)
```

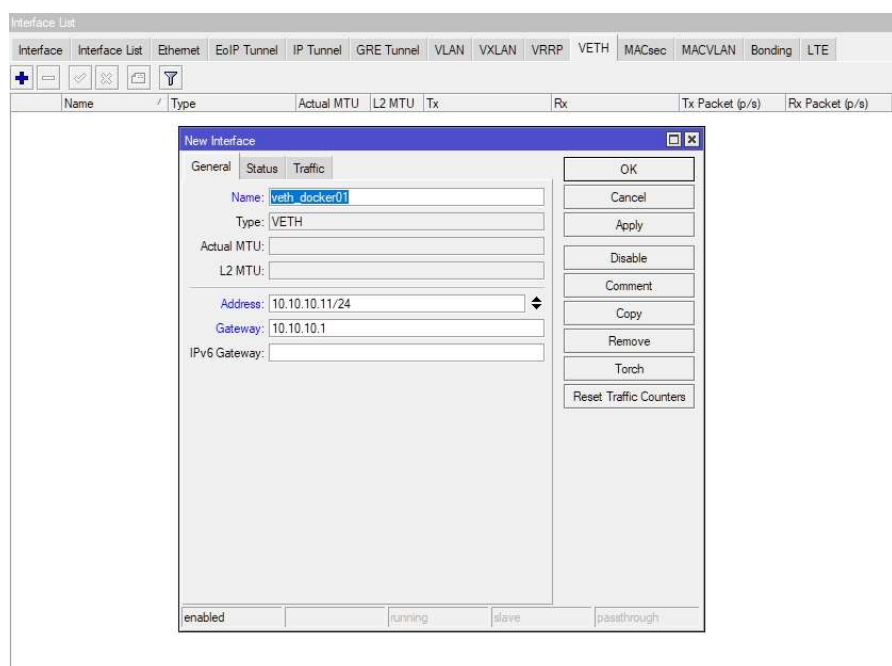
The optional command for setting up the DST-NAT-rule to SSH-access looks like this:

```
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 !)
```

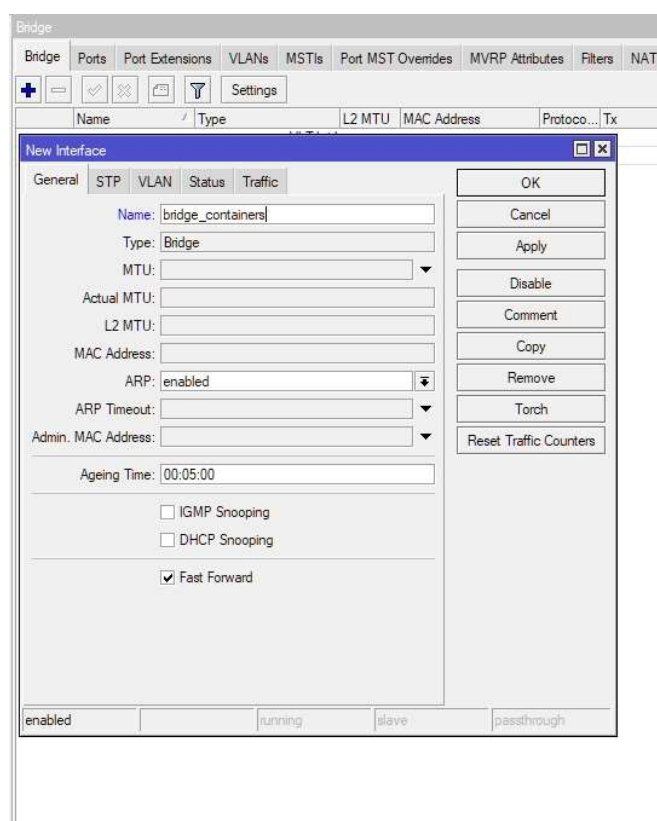
The dst-port=22022 is only an example. Please set it up regarding your preferences. It is advised that you refer to the basic configuration of the device-firewall for a complete understanding how these configurations work.

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

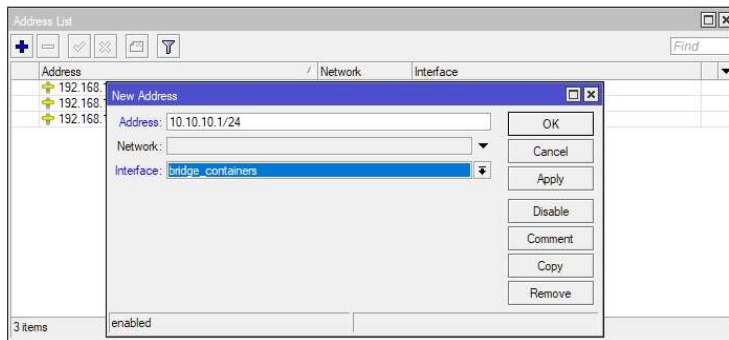
Adding VETH-interface:



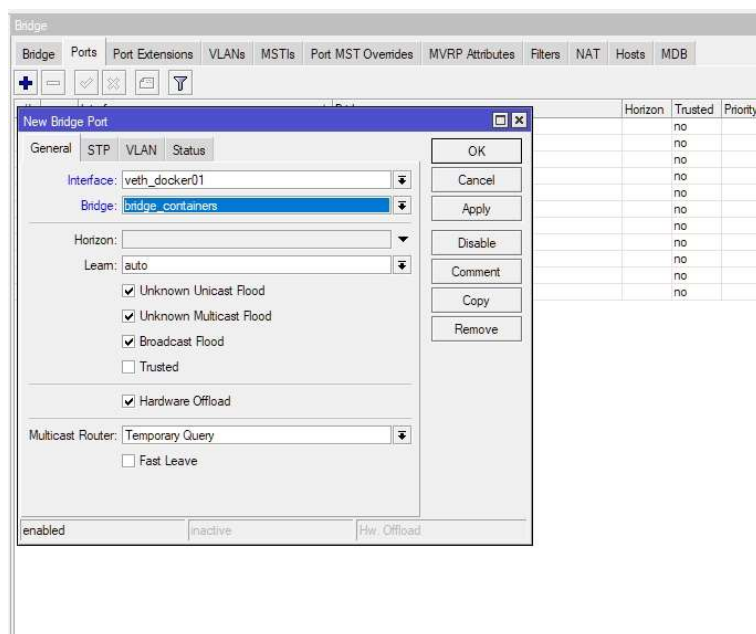
Adding bridge:



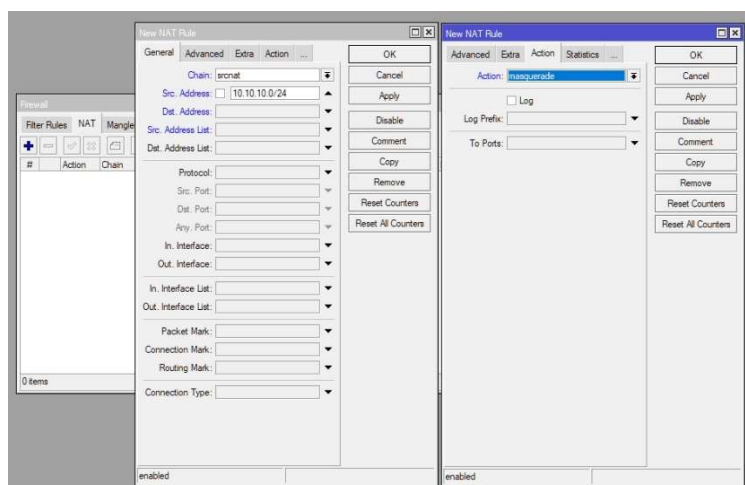
Adding IP to bridge:



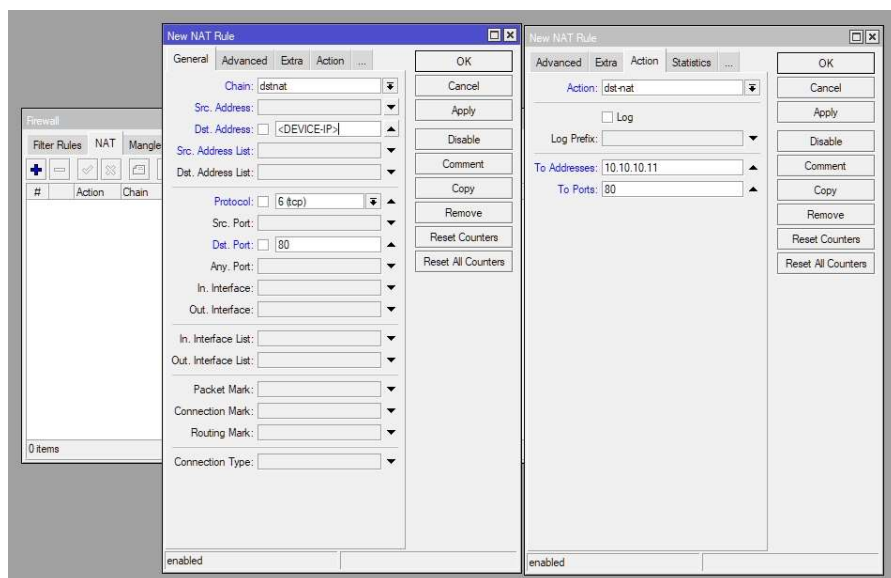
Adding VETH-interface to bridge:



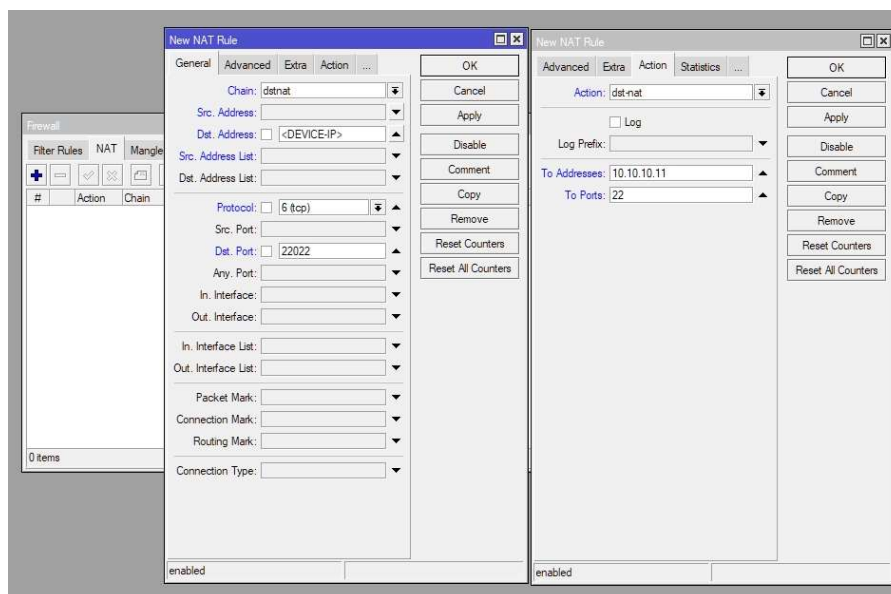
Setup container-masquerading:



DST-NAT rule for webserver:



DST-NAT rule for ssh-access:



Hint:

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.

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

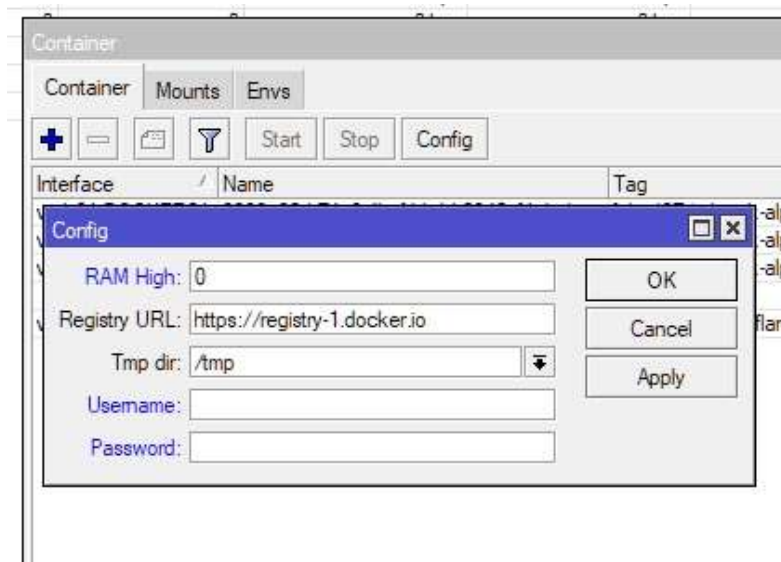
Next is the description to install the container itself on the device.

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 is setting up the “Registry-URL” and the “Tmp dir”. There is a download limit in action on “Docker HUB”, perhaps you need to setup an account on “Docker Hub” to prevent this. Then you need 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 needs 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:



Hint:

All images which I do provide are published publically 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:

The screenshot shows a 'Container' configuration window with the following fields and settings:

- File:** (empty text box)
- Remote Image:** `feltd67/mikrotik-alp_rc_upgrade-server:latest`
- Interface:** `veth01-DOCKER01`
- Envlist:** (empty text box)
- Cmd:** (empty text box)
- Entrypoint:** (empty text box)
- Hostname:** `DOCKER01@RB5009`
- Domain Name:** `network`
- Workdir:** (empty text box)
- Stop Signal:** (empty text box)
- Root Dir:** `usb2/docker01`
- Mounts:** (empty text box)
- DNS:** `10.10.10.1`
- ☒ **Logging**
- ☒ **Start On Boot**
- Name:** (empty text box)
- Tag:** (empty text box)
- OS:** (empty text box)
- Arch:** (empty text box)
- Status:** `none`

On the right side of the window, there are buttons for **OK**, **Cancel**, **Apply**, **Comment**, **Copy**, **Remove**, **Start**, and **Stop**.

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

Here is a picture of this process:

The 'Container' window displays the following configuration:

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

Buttons on the right: OK, Cancel, Apply, Comment, Copy, Remove, Start, Stop.

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

The 'Container' window displays the following configuration after completion:

- Interface: veth07-DOCKER07
- Envlist: (empty)
- Cmd: (empty)
- Entrypoint: (empty)
- Hostname: DOCKER01@RB5009
- Domain Name: network
- Workdir: (empty)
- Stop Signal: (empty)
- Root Dir: usb2/docker07
- Mounts: (empty)
- 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

Buttons on the right: OK, Cancel, Apply, Comment, Copy, Remove, Start, Stop.

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

Check out this screenshot:

Container

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: feltd67/mikrotik-alp_rc_upgrade-server:latest

OS: linux

Arch: arm64

Status: running

OK

Cancel

Apply

Comment

Copy

Remove

Start

Stop

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 “Root Dir” is set to “usb2/docker07” 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.

The “mus” will run as long you stop the container or you reboot your 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 that 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 should be reachable on that IP-address and port 80.

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

The first view looks like this:



mikrotik.upgrade.server v1.0.0

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

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

MUS-System running on host: *** using arch: ***

Disk-total: *** Bytes * Disk-free: *** Bytes * Disk-usage: *** Bytes

Memory-total: *** * Memory-used: *** * Memory-free: *** * Memory-shared : *** * Memory-buffer/cache: *** * Memory-avail: *** [-Bytes]

Script-status: *** --- Download-status: *** --- Last sync completed: ***

Name	Last modified	Size	Description
doc/	2024-09-01 12:34	-	Documentation
repo/	2024-09-01 12:34	-	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. Released under [MIT-License](#).

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 you are reading now.

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.

Another more useful example after the start may look like this:



mikrotik.upgrade.server v1.0.0

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

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

MUS-System running on host: **MUS@RB5009** using arch: **aarch64** - MUS-System status: **OK**

Disk-total: 28.0G Bytes * Disk-free: 24.1G Bytes * Disk-usage: 2.5G Bytes

Memory-total: 978.4M * Memory-used: 94.7M * Memory-free: 185.5M * Memory-shared : 1.6M * Memory-buffer/cache: 698.0M * Memory-avail.: 778.6M [-[Bytes]

Script-status: **Script(s) is/are NOT running** --- Download-status: **Download is NOT running** --- Last sync completed: 04:26 CEST on Sunday, 01.September 2024

Name	Last modified	Size	Description
 doc/	2024-09-01 01:04	-	Documentation
 repo/	2024-08-31 23:37	-	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. Released under [MIT-License](#).

Some words to this example shown above:

This “mus”-system was installed on a RB5009-device from MIKROTIK®. You might recognize in the fourth line from top the hostname [running on host:] “MUS@RB5009” and the architecture [using arch:] “aarch64” as the RB5009 is a ARM64-based device. If there are any issues with the internet- or dns-configuration, the “MUS-System status:” will change from “OK” to a hint, what may be wrong.

Changing the color to red will stop the background scripts.

There must checked what went wrong and the background-jobs must be started manually as described later in the manual. Only starting the container the first time, (after installation) will initiate the self-configuration- and download-job! After some error occurring here, you have start manually on the cli or wait for the crond-based-job to run nightly.

Below this are stated out the disk statistics. “Disk-total” will give you the complete size of the disk being used, while “Disk-free” and “Disk-usage” give actual sizes. When “Disk-free” falls below 5 GB the value will get red to inform you before getting in the situation of running out of disk-space.

The memory-statistics are mainly informational. All values are system-wide values and not only limited to the actual container. This is a system-limitation. “Memory-used:” is the memory used in the complete system (not only the running container). Please stay calm, if the “Memory-free:”-entry gets nearly to zero or to some 10ers MBytes. This value reflects the not-used memory, which means that this memory is used for nothing at all (even not for the system, programs or this container). After the background-scripts have ended (especially the “download-job”) this value will increase again to a normal value (~ 200 - 500 Mbytes).

The last line before the main download-box will give very useful information what is happening right now in the system.

“Script-status:” will display if the background-scripts are running (red).

“Download-status:” will display if there is a download-job running in background (red).

If both status-fields are displayed in green, then there are no background-jobs running. The system is in idle status and waits until the jobs are initiated via the cli manually or if the jobs are started via the crond-job (this is done at 00:00 h UTC each day).

The last completed synchronization is stated out with the “Last sync completed:”-field. The date of the latest sync is displayed and therefore the latest actual sync is given.

There is a third script in the background which is run to fill out the above information. This a cgi-script running in the webserver with a repeat time on 3 minutes. This makes a refresh to all the values in the given time. Although a “RELOAD” in the used browser will make also a refresh, perhaps two “RELOAD”s are needed.

Using the scripts on the cli will be described in a later section of this manual.

Here is a picture of the repo-directory:

mikrotik.upgrade.server v1.0.0

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

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

Name	Last modified	Size	Description
Parent Directory		-	
routeros/	2024-09-01 02:44	-	RouterOS®-packages
winbox/	2024-09-01 02:00	-	WINBOX®-Tools
.type	2024-08-31 23:36	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](#).

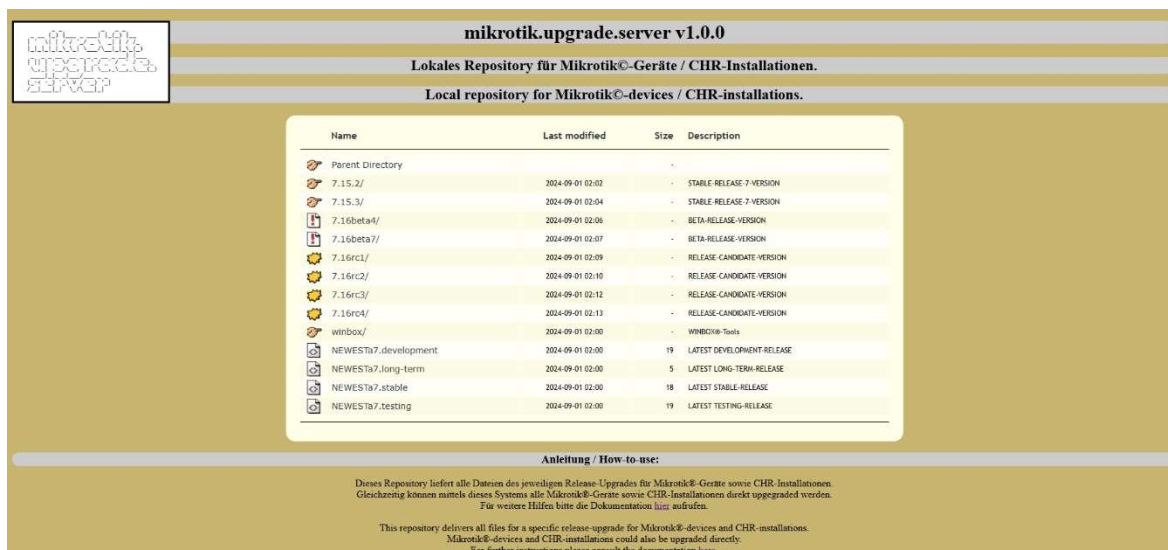
© 2024 by DL7DET. Released under [MIT-License](#).

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. Additionally setting a “persistent mount” will make the config-files and the packages in the repo-directory to survive a restart and also a reinstall (new installation/upgrade) of the “mus-system”.

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 v1.0.0
Lokales Repository für Mikrotik®-Geräte / CHR-Installationen.
Local repository for Mikrotik®-devices / CHR-installations.

Name	Last modified	Size	Description
Parent Directory		-	
7.15.2/	2024-09-01 02:02	-	STABLE-RELEASE-7-VERSION
7.15.3/	2024-09-01 02:04	-	STABLE-RELEASE-7-VERSION
7.16beta4/	2024-09-01 02:06	-	BETA-RELEASE-VERSION
7.16beta7/	2024-09-01 02:07	-	BETA-RELEASE-VERSION
7.16rc1/	2024-09-01 02:09	-	RELEASE-CANDIDATE-VERSION
7.16rc2/	2024-09-01 02:10	-	RELEASE-CANDIDATE-VERSION
7.16rc3/	2024-09-01 02:12	-	RELEASE-CANDIDATE-VERSION
7.16rc4/	2024-09-01 02:13	-	RELEASE-CANDIDATE-VERSION
winbox/	2024-09-01 02:00	-	WINBOX®-Tools
NEWESTa7.development	2024-09-01 02:00	19	LATEST DEVELOPMENT-RELEASE
NEWESTa7.long-term	2024-09-01 02:00	5	LATEST LONG-TERM-RELEASE
NEWESTa7.stable	2024-09-01 02:00	18	LATEST STABLE-RELEASE
NEWESTa7.testing	2024-09-01 02:00	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](#).

There is an entry with “7.15.3” which is the current stable version. A beta-version is identified in the “7.16beta7”-named directory. The “7.16.rc4” stands for the actual release-candidate. All directories contain the 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 these files and will end in generating automatically the download links and to execute the job to download them.

In the default installation now the “mus” will check out every night at 00:00 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”-system 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.

4.Usage of the mikrotik.upgrade.server

After the successful installation the “mus” can be used to download all packages of the current release locally without downloading them via the internet from one of the update-servers or the homepage provided by MIKROTIK®. It provides the actual Windows™ packages of the WINBOX®-utility, too.

Beside the nightly upgrade function the “mus”-system delivers the newest version of the needed packages and utilities without searching for them actively.

Setting optionally a DNS-entry for mus or mikrotik.upgrade.server will also be possible, while these domain-names are also threatet as valid servernames by the webserver-configuration. The safe way will be to reach the „mus“-frontend via the forwarded ip-address defined in the previous setup.

Setting mus in the network-wide dns-server(s) as DNS-entry will look like this:

```
ip/dns/static/add name=mus address=192.168.88.1
```

(“192.168.88.1” is just an example)

Another very important function of the “mus” is the seamless integration of the update-server inside the WINBOX®-utility. This integration is not actively integrated, the “mus” can adapt the update-sources hardcoded in the WINBOX®-utility without changing anything. The only needed configuration is to add a static DNS-entry to the network-wide DNS-services.

Under the hood of the “mikrotik.upgrade.server” there lives a second webserver which listens to each request done on the domain-entry of “upgrade.mikrotik.com”. Adding a static entry to the network-wide DNS-services redirects all requests for updates/upgrades to the second webserver on the “mus” and delivers the packages as needed.

Here are the settings which are needed to redirect the update requests to the “mus”.

Let’s say the device running the “mus” has the internal (but reachable) IP-address of “192.168.88.1” and a proper DST-NAT-configuration to the IP-address of the container set, then the command on the CLI (“New Terminal”) will be:

```
ip/dns/static/add name=upgrade.mikrotik.com address=192.168.88.1
```

Hint:

If you plan to update your device where the “mus” is running on, please add an entry in the DNS-static-list, where the IP-address is that one where the “mus” is running on (docker-container / VETH-IP-address). The reason is in that case the DST-NAT-rules won’t be working correctly.

```
ip/dns/static/add name=upgrade.mikrotik.com address=10.10.10.101
```

(“10.10.10.101” is just an example)

The entry for other devices (not the one that “mus” is running on) looks like this in the WINBOX®-client:

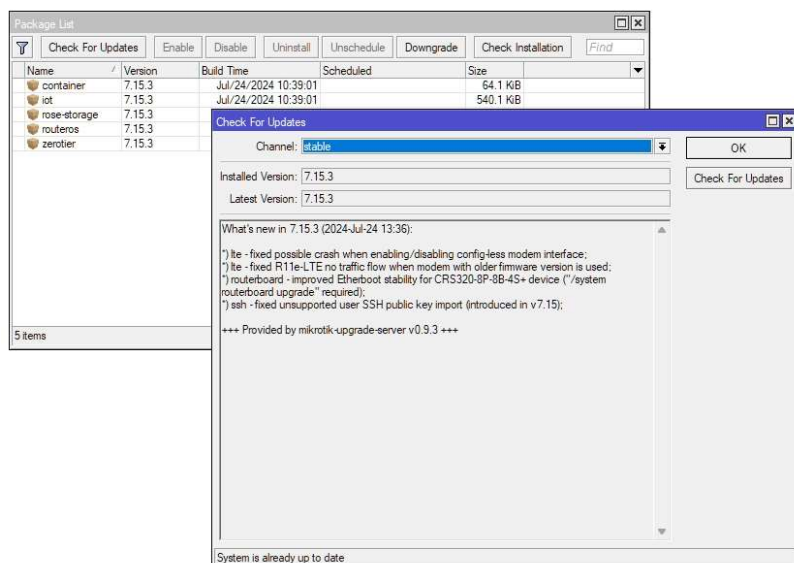


This entry could also be added from the cli of the WINBOX®-client, too.

Please keep in mind that if there are secondary or more DNS-servers in your network, the static entry must be set in addition there accordingly.

Done these settings you could use the update function inside the WINBOX®-client to check for new updates/upgrades.

Please click in the left sidebar of the WINBOX®-client on “System” => “Packages” to display the “Package List”. Then click on “Check For Updates” and you will get the following display:



Important is to recognize the last line in “Changelog”-windows which appears:

```
routerboard upgrade" required);
*) ssh - fixed unsupported user SSH public key import (introduced in v7.15);
+++ Provided by mikrotik-upgrade-server v0.9.3 +++
```

“+++ Provided by mikrotik-upgrade-server v0.9.3 +++” (v0.9.3 is only an example reflecting the current version of the “mus”-system) means that the packages will be served from the “mus” and not from the official update-servers by MIKROTIK®.

If this line is not shown, then you are probably not loading the update-information from the “mus”. Please check then the DNS-settings and additionally the DST-NAT-configuration.

After an update of the “mus”-system to a newer release this version-number will be updated after restarting the updated version and the first recheck of the available packages.

The update channels “stable”, “testing” and “development” are currently available for upgrades. The “long-term” channel will give you a little different information, telling you to choose another channel.

If these update-informations for the different channels are not displayed as expected, please change the channel from on to another channel and back and check again. There may be a caching problem when displaying the correct update/upgrade-information.

From here you could use the “Download” or “Download & Install” functions as known with the official upgrade-servers.

Extended storage by using “persistent storage” of the container

This feature is very handy if you want to build an archive from all latest releases-versions of the packages and files provided by the “mikrotik.upgrade.server”.

Because of the volatile nature of the containers and the depended storage, all data and configurations will be lost (deleted) when killing and removing the container. This is a meant to be a normal feature and can be overcome by using “persistent storage”.

In normal mode all storage lives in the container and therefore it will be lost if the container is deleted. The solution is to “map out” the inside storage to a storage living in the outside world (the base system). This is done in RouterOS® by setting up “Mounts” and adding them to the container configuration.

“Mounts” are mappings from a system storage (preferable on an additional disk) directing to a unique directory on that storage. They could be setup described like this:

Creating mounts for configuration- and data-storage on the CLI:

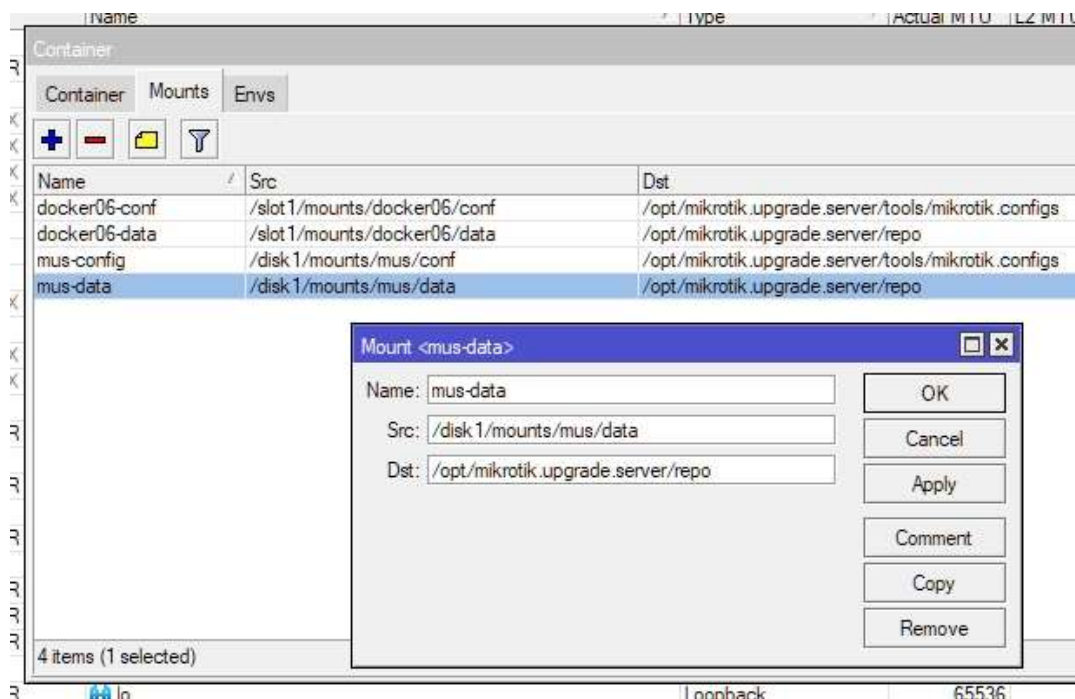
```
container/mounts/add name=mus-config src=/disk1/mounts/mus/conf \
dst=/opt/mikrotik.upgrade.server/tools/mikrotik.configs
```

```
container/mounts/add name=mus-data src=/disk1/mounts/mus/data \
dst=/opt/mikrotik.upgrade.server/repo
```

(the \ means to put all content in one line !)

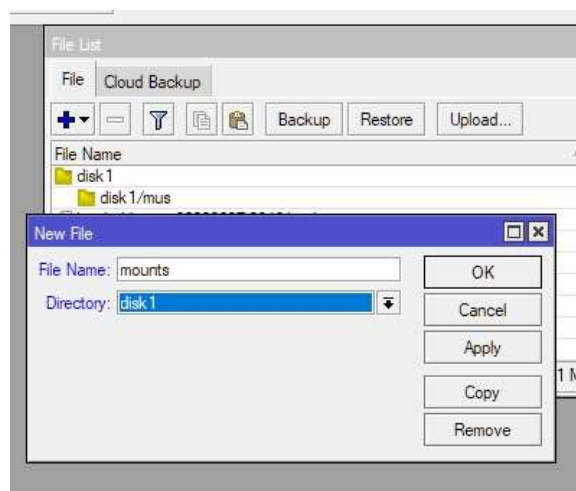
The “src=”-entry could be chosen as needed. The “dst=”-entry is mandatory and must NOT be changed!

This entry could also be added via the WINBOX®-client. In both cases the mounts will be shown as in the example below:



Hint:

It may be needed to create a directory in the “File List” to make the mounts work like expected. That is due to an issue, that currently RouterOS® creates only the next subdirectory leaving a gap in this example:



Now the mounts can be added to the container configuration. Choose the appropriate boxes and add the two newly created entries to “Mounts”:

The screenshot shows the 'Container' configuration window. The 'Mounts' field is expanded, showing two entries: 'mus-config' and 'mus-data'. The 'Root Dir' is set to 'disk 1/mus'. The 'DNS' is set to '10.10.10.1'. The 'Logging' and 'Start On Boot' checkboxes are checked. The 'Status' is set to 'none'.

Now start the container to apply the configuration and check if it is running (and that it stays running!):

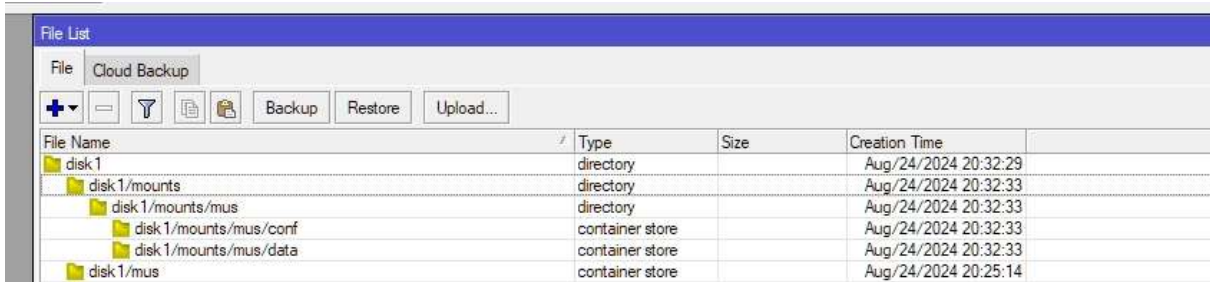
The screenshot shows two windows. The top window is the 'File List' showing the container's file system structure. The bottom window is the 'Container' configuration window, which is now showing the container's status as 'running'.

File Name	Type	Size	Creation Time
disk 1	directory		Aug/24/2024 20:25:09
disk 1/mounts	directory		Aug/24/2024 20:25:13
disk 1/mounts/mus	directory		Aug/24/2024 20:25:13
disk 1/mounts/mus/conf	container store		Aug/24/2024 20:26:07
disk 1/mounts/mus/data	container store		Aug/24/2024 20:25:42
disk 1/mus	container store		Aug/24/2024 20:25:14

The 'Container' window shows the following configuration:

- Interface: veth07-DOCKER07
- Envlist: (empty)
- Cmd: (empty)
- Entrypoint: (empty)
- Hostname: mus@RB5009
- Domain Name: mynetwork
- Workdir: (empty)
- Stop Signal: (empty)
- Root Dir: disk 1/mus
- Mounts: mus-config, mus-data
- DNS: 10.10.10.1
- Logging: ☒
- Start On Boot: ☒
- Name: a62f546b644-43aa-bf05-55d5d254c633
- Tag: felted67/mikrotik-alp_rc_upgrade-server.latest
- OS: linux
- Arch: amd64
- Status: running

It is important that the mounts created before are shown with the “Type” as “container store” in the “File List”:



File Name	Type	Size	Creation Time
disk1	directory		Aug/24/2024 20:32:29
disk1/mounts	directory		Aug/24/2024 20:32:33
disk1/mounts/mus	directory		Aug/24/2024 20:32:33
disk1/mounts/mus/conf	container store		Aug/24/2024 20:32:33
disk1/mounts/mus/data	container store		Aug/24/2024 20:32:33
disk1/mus	container store		Aug/24/2024 20:25:14

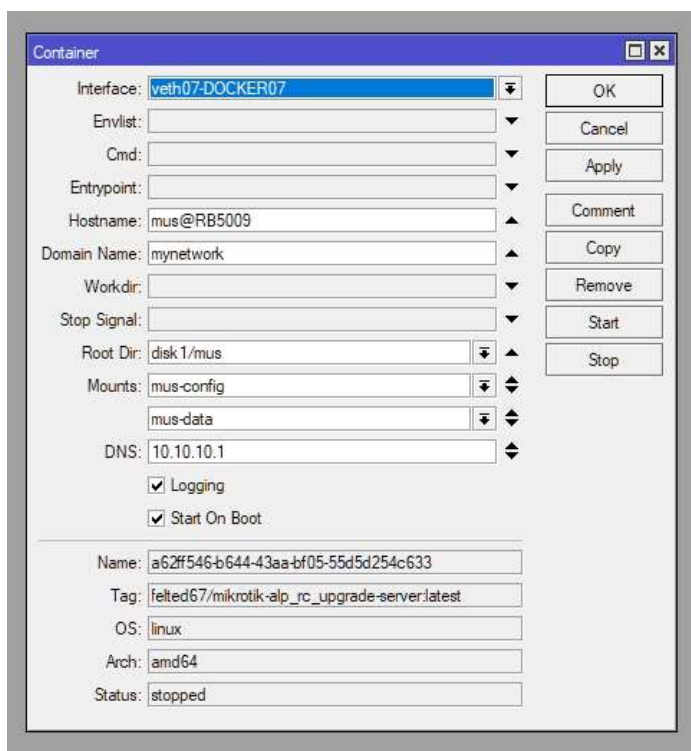
The successful configuration of the “persistent storage” gives you now the ability to upgrade or completely remove and reinstall the “mikrotik.upgrade.server” without losing the prior data and configuration as long as the “Mounts” will be untouched and will not be deleted.

This is very useful when upgrading the “mus” itself like described now.

Upgrading or reinstalling the “mikrotik.upgrade.server” to a newer or specific version

Because of the ongoing development and permanent process of adding of new functionality it may be possible that the “mus”-system must be updated or reinstalled. Here comes the “nearly-one-click”-functionality in action. Beside removing the container completely (remind to leave the mounts untouched as described before), there is a way to do that update in a very easy way.

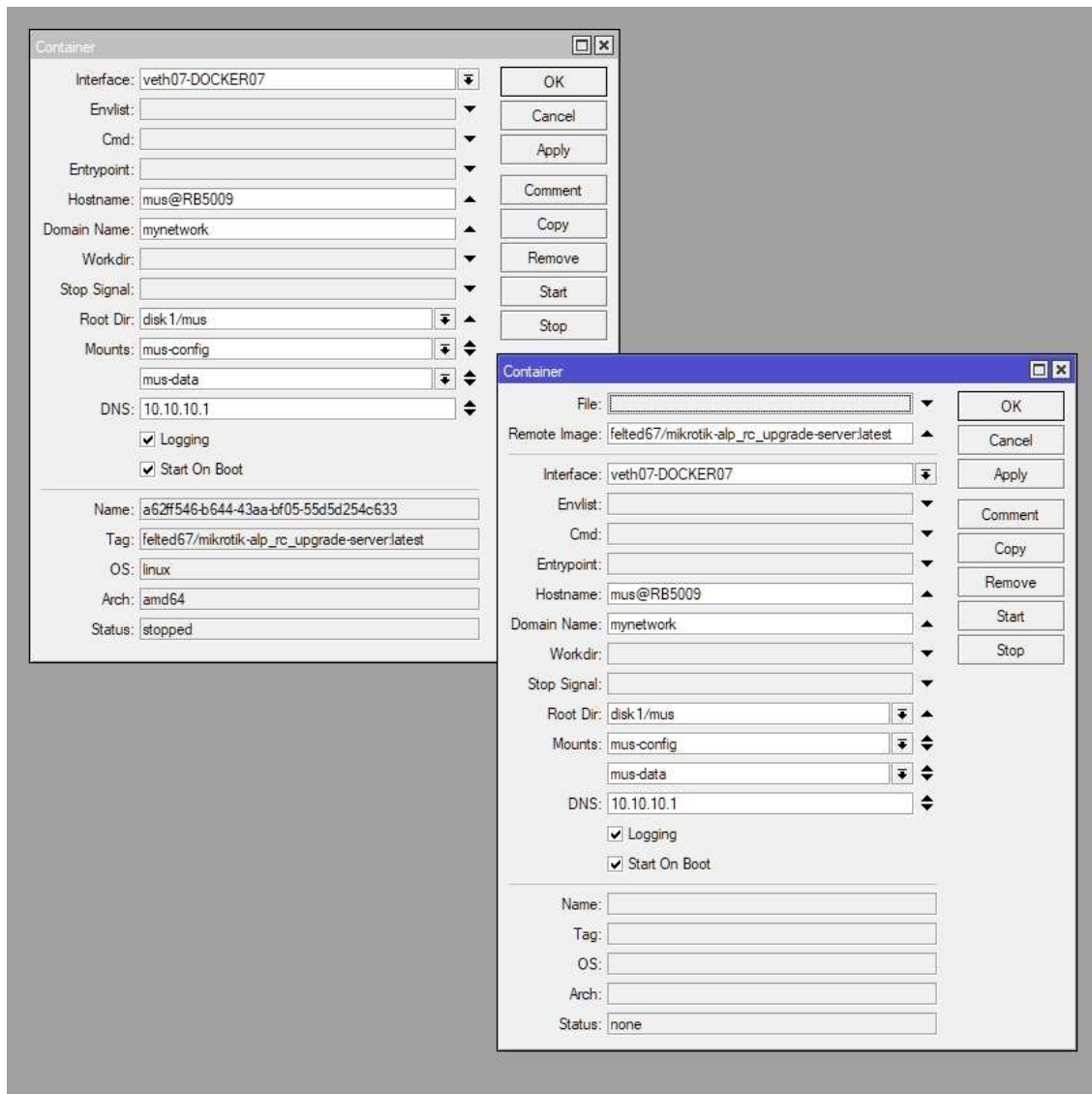
At first stop the running container. The “Status” will be changed via “stopping” and then to “stopped”. This looks like this:



Now click on “Copy” to copy the complete entry. A new “Container”-configuration will appear.

Please check that in the “Container”-configuration the “Remote Image”-field is populated and not empty. If it is empty, please copy the “Tag”-entry from the previous, stopped container-entry and insert it on the “Remote Image”-field. This can occur especially on ARM/ARM64-architectures. Check again that all fields in the new configuration are properly filled as in the previous configuration.

Please check the following image:



Now click on the previous (stopped) configuration the “Remove”-button but leave the new configuration untouched. The “old” configuration will disappear (and maybe some entries in the “File List” will also disappear if this window is open). This is a normal behavior. During this process leave the new, left configuration-windows untouched. If not, you will have fill in all the configurations you have done before.

Now click on “Apply” in the left-over configuration-box. That lets the device to download the newest version and reinstall it with optionally using the mounts you have done before.

In the example above we have used the “Remote Image” with the entry

feltd67/mikrotik-alp_rc_upgrade-server:latest

This will install the latest stable version on your device, but perhaps you want to install a more specific version from the docker hub repository. Then you can choose the adequate version and fill in the entry from docker hub’s “pull <image-name>” on the right side of the docker hub website.

It may be needed to install devel-versions or older stable versions, if there are any problems with the current stable release. Please choose the version as you like.

Hint:

During installation of the container you must not choose a corresponding version of the image for the needed architecture. All builds are done parallel to the three available architectures needed for the different MIKROTIK®-devices. The device chooses automatically the right image for the architecture the device is running on.

After an update/upgrade of the “mus” the system will run the actual updated version regardless of a stop/restart. The system will begin automatically to check for new releases and download and serve them, if new versions are available from the master-servers at MIKROTIK®. After completing this “first-start”-update the system will recheck on 00:00 h UTC using a cronjob integrated in the system.

Accessing the “mikrotik.upgrade.server” via SSH-client

Hint:

WARNING ! This feature is for advanced users and needs to set a root-password on the container. If the firewall and the DST-NAT-rule are not properly set, a direct access to the container with administrative rights may be possible to anyone who has access to the container or the device running the container and knows the password of the user “root”.

This feature is a left free feature from the previous development stages. Currently the access is open as a root user to the “mus”, but there is no password or an authentication-key set up in the container. If the DST-NAT-rule to the SSH-port is not setup in the devices firewall, then there will be also no access possible.

If you are interested in looking inside the container or run in several troubles or you are more curious than a “normal” user, you can access the “mus” via SSH-protocol and SCP/SFTP-protocol. First you need to set up a root password into the container.

There is a system-wide way to access the container from the RouterOS®-CLI or the WINBOX®-client. Click in WINBOX® on the left panel on “New Terminal”.

Then insert the following:

/container print

The output will show a list of the current installed containers. If you have installed more than one container beside the “mus”, please lookup for the number in front of the correct container-entry. This is the container-number you will need now.

Attach via CLI to the “mus”-container with using the number from the step before:

```
/container shell num=<number>
```

This will open a shell accessing directly the container.

Now you can set the “root”-password needed for accessing the “mus” from outside via SSH-protocol.

You can type the following in the console:

```
$ passwd root
```

Choose an adequate password (remember this is your last barrier for direct administrative access to the system) and fill it in. As usual repeat it. Then you can exit the shell and the CLI.

Now you can access the “mus” via SSH-protocol (in Windows™ with WinSCP for example). There you find the complete Alpine Linux environment running inside the container. For your convenience there is a “Midnight Commander” [mc] installed. The “mus”-environment is located under /opt/mikrotik.upgrade.server/*. All other services and they corresponding configurations are in the Alpine-Linux-default locations (ssh / apache / openrc).

Manual start of the “mus”-scripts

The scripts, which implement all functionality, can be started manually. These are

```
/opt/mikrotik.upgrade.server/tools/mus-start.sh ,
```

```
/opt/mikrotik.upgrade.server/tools/mus-sync.sh and
```

```
/opt/mikrotik.upgrade.server/tools/mus-gen-status.sh .
```

Additionally, there are three scripts executable from everywhere in the system, as they are added to the path, but in a little different way. Please understand that this system-wide-files are called **without** the .sh-extension.

The first script to start is **mus-start**, where this one does first downloads and (re-)generates the config-files.

The second script named **mus-sync** will download the packages and will integrate them to the repo-structure.

The third script which is named **mus-gen-status** gives some useful information of the system-status, but also generates the status information on the web-interface.

Running these three scripts will also help in debugging the “mus”-system, while they display some information about what they are doing. Please start the first script at the beginning of a manual start, as both (**mus-start** & **mus-sync**) are mainly running non-destructive. These two scripts can’t be started if another instance of them is running (also the crond-job).

Running these scripts in parallel will harm the file structure, as both of them make use of the same `./temp-directory`. The “mus”-system recognizes the parallel start automatically and prevent them from further execution.

Further information and some tweaking examples are described in the next chapter.

At this point be warned again – all things you are doing from this point on will occasionally break the “mus”-system and furthermore the whole MIKROTIK®-device and its security features!

Hint:

One more security hint: Beside that the “mus”-environment is currently running as the root-user, the webserver mapped to the outside will run with secure features as “apache”-user. This root-user-behavior will be changed in one of the next versions to use the “root”-user only for preconfiguring processes and then switch to a non-root-user for running the “mus”-scripts. Because of the complexity of the preparation process for the container, that feature will be on in one of the next releases. The reason for that complexity is that the container is not running only one process for which the container is meant to be, but furthermore it is running several processes in parallel to give the functionality as needed. Some of them must set up a root-user.

5. Technical background of the project

This chapter is meant to read for the advanced user or someone else, who wants to get some specific information about in the project from under the hood.

Basic functions of the “mikrotik.upgrade.server”:

This project/software provides in a small Alpine Linux-environment the openrc-init-system, a cron-service, a webserver, a ssh-server, several scripts using the bash-shell and some small tools for manipulation strings and variables on bash-level. The main functionality is written with bash-scripts. To be precise all functions are handled via three bash scripts, where the two scripts for creating the file-structure and the download of the needed packages are activated manually or via a cronjob at 0:00 h UTC. The third script is used to give some status-information and to fill in the status-lines in the web-frontend. It is started every 180 seconds and also on a browser-refresh done when viewing the web-frontend.

The base system consists of a container with running the openrc-init-system as a process with id 1, the main process. This gives the ability to run additional jobs inside the container controlled by the openrc-process. This breaks the standard definition of a container, which normally runs only one job. But this “enhancement” gives the freedom to start other jobs and stop/restart them without killing the container at all. This is also very handy when you need the ability to run some processes in parallel without creating additional containers. In a “normal” container-environment it is possible and advised to create one container for each process and wire them together with the use of docker-compose or other third-party-tools.

Using containers with RouterOS® from MIKROTIK® gives only the chance to run one container without the use of docker-compose. There may be exist the possibility to run one container for each process you need for your project and wire them together outside of these containers in RouterOS®, but with the need of additional work and surely some tricks.

When planning to build containers for RouterOS® I decided to choose the openrc-init-system and define a default container which can be extended with the process-functions I needed. Some functions can be implemented seamlessly, but when a service needs cgroup-functions from the kernel, it is nearly not implementable. Some more simpler services as a webserver, ssh-server or some easy scripts are positively usable.

One aspect of a container which is running on a device is the size of the container. Alpine Linux is a very small distribution when it comes to image-size and it is the default building-environment when creating (docker-)containers. To minimize the size of a usable container-image I decided to use no explicit programming language inside the container right now, but only using pure bash-scripts. These bash-scripts are very limited in the functions, so perhaps I will switch to python in the future.

Now back to the specific functions of the “mus”-system. I wanted to build a system which will configure itself with only a minimal pre-configuration. This is done by only downloading the release-status-files from the master-servers at MIKROTIK® at the first stage. These are the “NEWEST7.*”-files and they will give the latest release-versions which are available for download. Then I created a template-file with all names of the files which must be downloaded, but with a DUMMY-release-number. Putting the template-data and the release-number from the “NEWEST7.*”-files together gives the concrete filenames to download from the download-area at the MIKROTIK®-master-servers. These functions generate several *.config-files which contains the file-lists for downloading the needed packages.

Older configuration-files can be copied manually via SCP/SFTP-access (see before) into **/opt/mikrotik.upgrade.server/tools/mikrotik.configs** to download these specific versions also. These can be found at **/aux_files** at the Github source-repository. There is also the template-(.raw)-file available for your reference.

The handling of the WINBOX®-packages is very similar to the above.

Another function is to download the all-packages-files and extracted all files from these archives. There are several situations when it is needed to install a specific function-package-file from the all-packages-archives. The template-file defines some files additionally which are needed for updating the Dude® for example. As the template-file is editable the download-process is easy expandable. Linking this generated repo from all downloaded files to the webserver makes them available for download with fancy-indexing on the Apache-webserver. Some nice formatting makes the website with the repository and other files attractive for downloading the needed files easily.

The second main function of this “mikrotik.upgrade.server” is the ability to integrate the update-service from WINBOX® to perform all upgrades from inside of the WINBOX®-client. This is done with a second webserver-virtual-host-configuration which serves webservices with the domain “upgrade.mikrotik.com”. Setting an DNS-entry in the DNS-service inside the customer network devices that points with the domain “upgrade.mikrotik.com” to the local IP-address enables to update the device from the local “mus”-system.

All these functions for updating/upgrading or downloading the actual releases will become very useful when the customer-network consists of many MIKROTIK®-devices.

I found out that for fairness reasons several sequential downloads via the same public-IP-address will slow down the download-process significantly. This is understandable when putting the sold devices by MIKROTIK® in relation to the download-requests when releasing a new version. This can't be satisfied to get an all-fast download speed. Running several server-systems with access from many (yes many!) customers gives me the point of view, that someone can't provide all the time the adequate download speed if several peaks occur.

Beside the handiness of a local repo-server (in technical nature view) the system must be as easy to setup and run as smooth as possible. This goal is nearly completely satisfied.

In the end I would like to give some information about the build process and the distribution of the container-images. The complete build process is done via a Gitlab-system using CI/CD-techniques. The pipeline builds all arch-versions available now (x86_64, ARM, ARM64) and publishes the build images directly to docker hub. While the Gitlab-system is a closed system with no public access, the sources of the different commits are copied to a corresponding Github-repository. At this point the sources are available at the Github-repository while the container-images are pullable via docker hub. Some short instructions for installing and updating the "mus" are also available at docker hub. The complete instructions, the sources and this documentation is available on the Github-repository.

A wiki and perhaps a helpdesk/forum are planned to be installed in the future for handling questions, feature-request and/or issues.

All links and web-addresses will be published in the addendum of this documentation.

6.Addendum

The (docker-)images are available on Docker Hub:

https://hub.docker.com/repository/docker/felted67/mikrotik-alp_rc_upgrade-server

The source code is available on Github:

https://github.com/felted67/mikrotik-alp_rc_upgrade-server

The wiki from MIKROTIK® regarding CONTAINER-functions:

<https://help.mikrotik.com/docs/display/ROS/Container>

Contact details:

Detlef Lampart / DL7DET / felted67

E-Mail: **detlef(at)lampart.de**