MikroTik Professionals Conference

Volume Installations

Prague 2024

about the speaker

- Yann Shukor
- Specialised in IT since 1988
- Systems & network maintenance
- Cabling and installation of WIFI networks
- Mikrotik certified since 2012

Azurtem

https://shop.azurtem.com

Volume Installations

introduction

strategy

requirements

lessons learnt

questions

introduction

Small Network

- An increasing number of ISPs in France are adopting Mikrotik not only within their data centres but also in their choice of CPEs.
- Winning such tenders is dependent on our capacity to adapt the proposed Mikrotik devices to the client's environment.
- The ability to provide pre-installed/pre-staged devices is crucial in order to rival traditional brand offerings.

Small Network (cont'd)

- Adapting Mikrotik devices to an ISP's needs may entail multiple processes/steps in order to meet the client's expectations.
- The client usually requests a specific RouterOS version thus requiring the upgrade/downgrade of the purchased devices.
- More often than not the devices/boxes need to be labelled with the ISP's identification e.g. ID, serial, MAC, …

Small Network (cont'd)

- The upside, in exchange for some extra work, is the additional cost that you as a supplier can charge per device prepared.
- Outsource this task would contribute to someone else's growth, potentially complexify the whole process and hinder quality control.
- The objective is to handle the device preparations in the fastest possible manner and with the least impact on the expected revenue.

Considerations

- The proposed approach isn't what one would qualify as "industrial level" but rather "garage workshop" type dimensions.
- You have to assess how many units you can install per session. From a purely Mikrotik perspective, fifty units at a time is feasible.
- Don't forget that things can go wrong. Recovering from a failed installation of fifty units can be quite time consuming.

Considerations (cont'd)

- "Time is money" as Benjamin Franklin once wrote. Therefore keep things simple and follow a predefined step by step procedure.
- The aim is to prepare as many devices as possible in the least amount of time; ideally you must act like a factory 'chain worker'.
- Don't deviate from the procedure unless something goes wrong i.e. if you are faced with unforeseen circumstances.

Azurtem

requirements

Hardware

- A work bench large enough to accommodate fifty devices
- A fifty port PoE switch e.g. CRS354-48P-4S+2Q+RM (2x S-RJ01)
- Ethernet cables to connect the devices up to the switch
- A computer preconfigured to run the installation procedure.
- Labels to identify the units once they have been installed

Azurtem

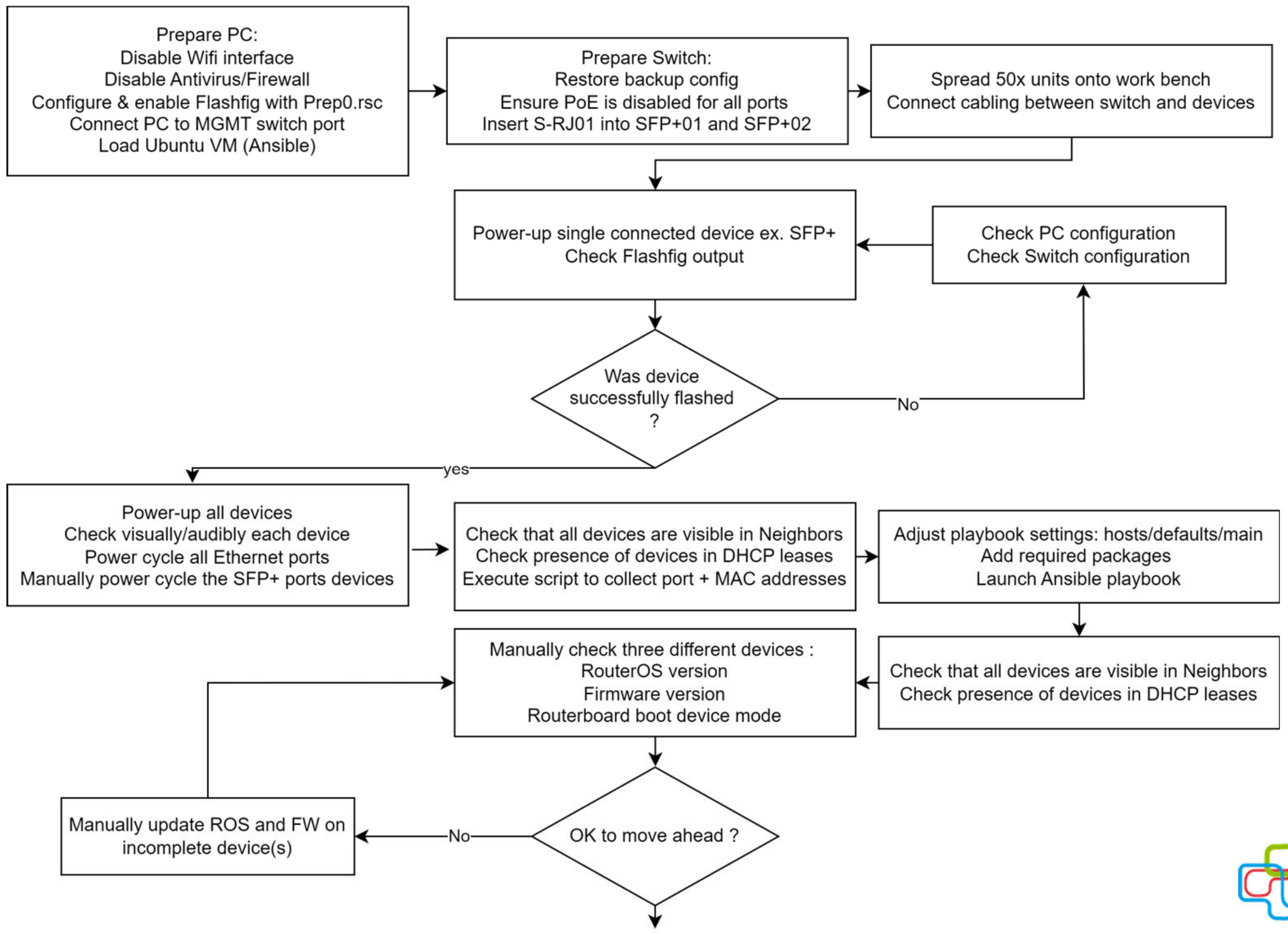
Hardware (cont'd)

Software

- Flashfig : tool that does the magic
- Netinstall : helps when you have to manually install a device
- Winbox/SSH : provides control and oversight of tasks/result
- Linux : because Windows won't suffice
- Ansible : used to upgrade/downgrade the devices
- A means to print the devices' identification labels

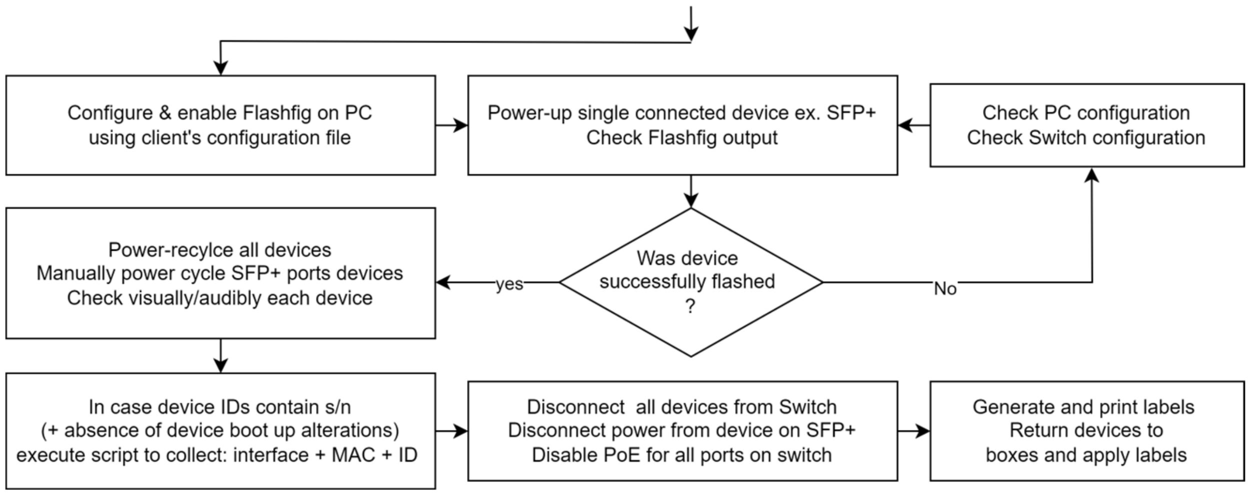
strategy

Task flow


Prepare PC:
Disable Wifi interface
Disable Antivirus/Firewall
Configure & enable Flashfig with Prep0.rsc
Connect PC to MGMT switch port
Load Ubuntu VM (Ansible)
Prepare Switch:
Restore backup config
Ensure PoE is disabled for all ports
Insert S-RJ01 into SFP+01 and SFP+02
Spread 50x units onto work bench
Connect cabling between switch and devices
Power-up single connected device ex. SFP+
Check Flashfig output
Check PC configuration
Check Switch configuration
Was device successfully flashed ?
No
yes
Power-up all devices
Check visually/audibly each device
Power cycle all Ethernet ports
Manually power cycle the SFP+ ports devices
Check that all devices are visible in Neighbors
Check presence of devices in DHCP leases
Execute script to collect port + MAC addresses
Adjust playbook settings: hosts/defaults/main
Add required packages
Launch Ansible playbook
Check that all devices are visible in Neighbors
Check presence of devices in DHCP leases
Manually check three different devices :
RouterOS version
Firmware version
Routerboard boot device mode
OK to move ahead ?
No
Manually update ROS and FW on incomplete device(s)

Azurtem

Task flow (cont'd)


Configure & enable Flashfig on PC using client's configuration file
Power-up single connected device ex. SFP+ Check Flashfig output
Check PC configuration Check Switch configuration
Power-recylce all devices Manually power cycle SFP+ ports devices Check visually/audibly each device
yes
Was device successfully flashed ?
No
In case device IDs contain s/n (+ absence of device boot up alterations) execute script to collect: interface + MAC + ID
Disconnect all devices from Switch Disconnect power from device on SFP+ Disable PoE for all ports on switch
Generate and print labels Return devices to boxes and apply labels

Switch

Prepare the bridge:

```
:for x from 1 to 49 do={/interface bridge port add bridge=bridge interface=("ether".$x )}
:for x from 1 to 49 do={/interface bridge port set number=$x hw=no}
:for x from 1 to 2 do={/interface bridge port add bridge=bridge interface=("sfp-sfpplus".$x)}
```

Control the power:

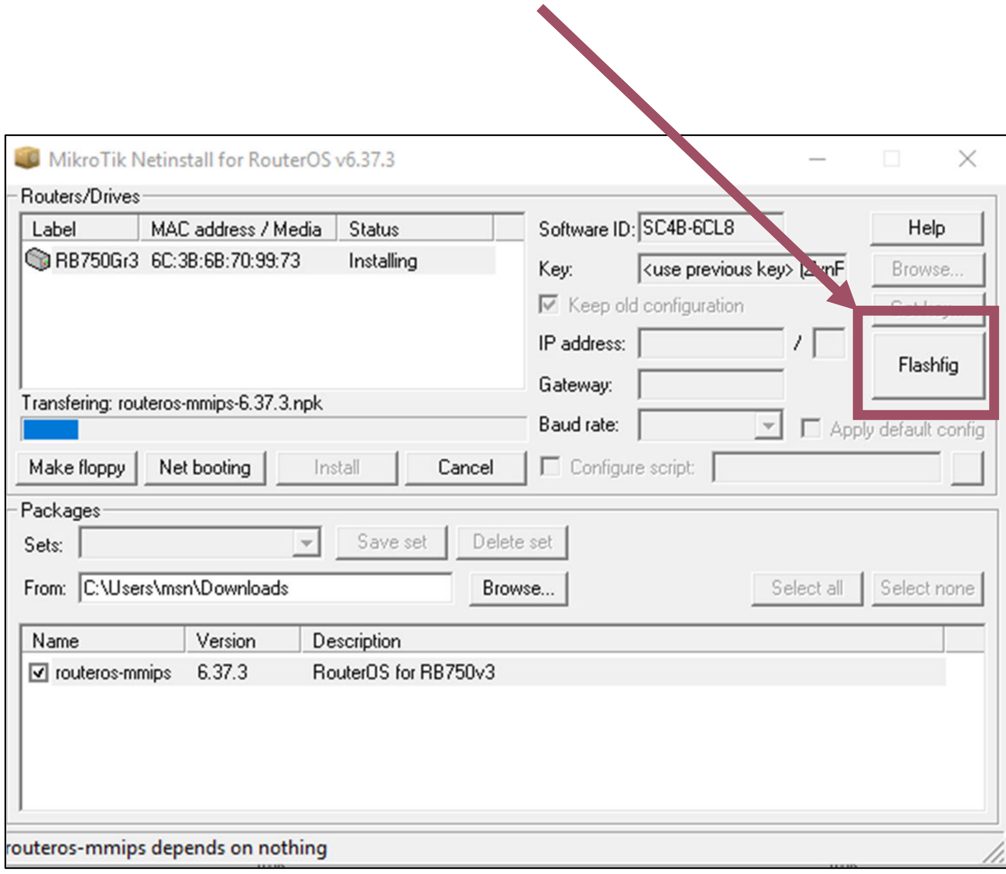
```
:for x from 0 to 47 do={/interface ethernet poe set $x poe-out=off}
:for x from 0 to 47 do={/interface ethernet poe set $x poe-out=auto-on}
:for x from 0 to 47 do={/interface ethernet poe power-cycle $x}
```

*neighbor/hosts lists will allow one to control progress/results

Azurtem

Flashfig

- Previously, Flashfig was launched from within NetInstall
- It has been available as a separate tool since RouterOS v6.47
- Until last May, Flashfig could only replace the 'running configuration' not the default-configuration



Label	MAC address / Media	Status
RB750Gr3	6C:3B:6B:70:99:73	Installing

Transfering: routeros-mmips-6.37.3.npk

Software ID: SC4B-6CL8
Key: <use previous key>
[✓] Keep old configuration
IP address:
Gateway:
Baud rate:
[ ] Apply default config
[ ] Configure script:

Sets:
From: C:\Users\msn\Downloads

Name	Version	Description
[✓] routeros-mmips	6.37.3	RouterOS for RB750v3

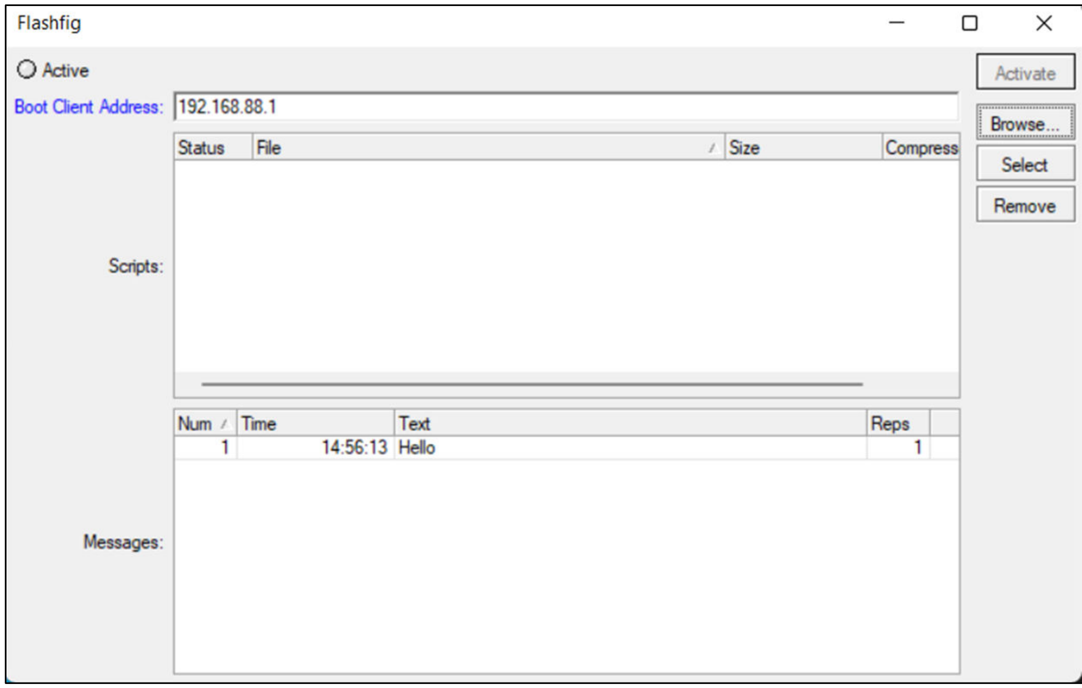
routeros-mmips depends on nothing

Flashfig (cont'd)

- Early May 2023 Druvis published two videos announcing the possibility of using Flashfig to facilitate volume installations :

Flashfig (cont'd)

- Flashfig executes within 3s; before ROS is loaded, therefore no admin password is required
- Flashfig is a Windows executable
- Can only act upon devices located in the same L2 network segment



Boot Client Address: 192.168.88.1

Num	Time	Text	Reps
1	14:56:13	Hello	1

https://help.mikrotik.com/docs/display/ROS/Flashfig

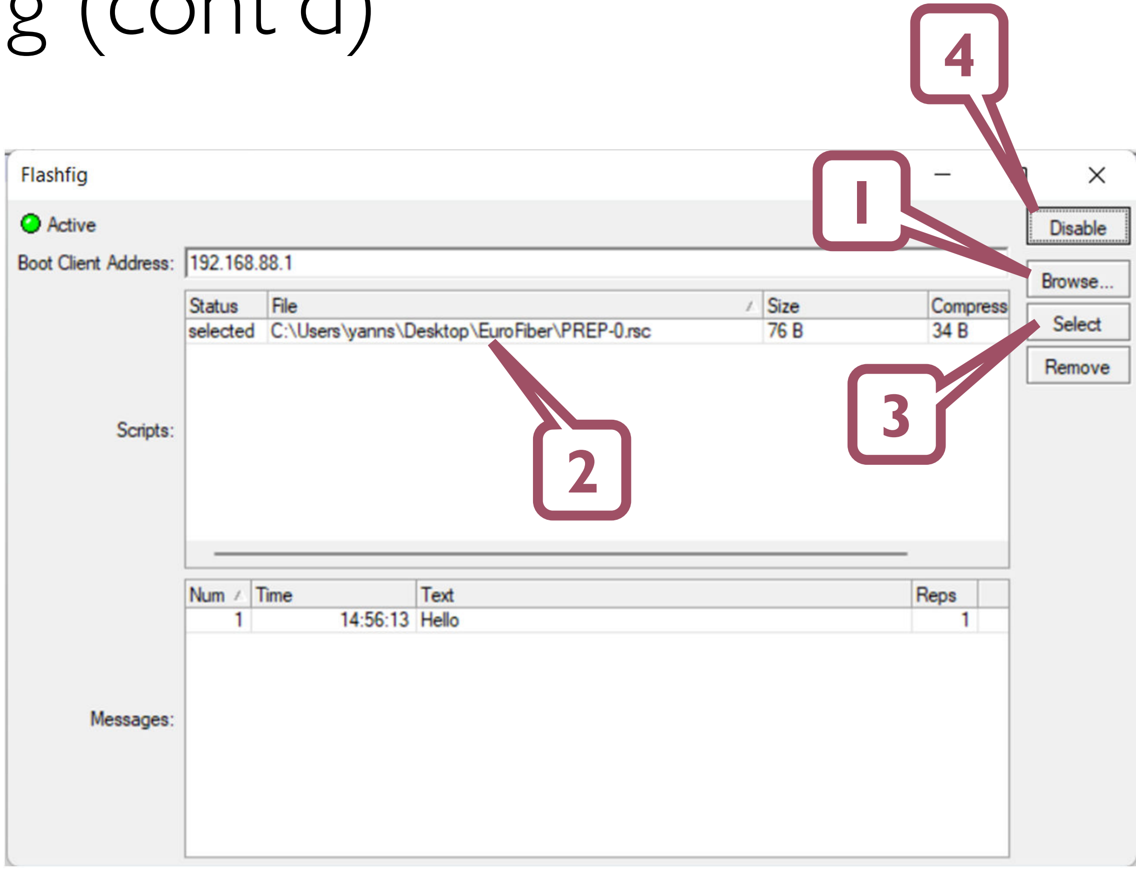
Flashfig (cont'd)

1) Choose your source .rsc file

2) Click on the loaded script

3) Click on "Select"

4) Click on "Enable" to activate



Boot Client Address: 192.168.88.1

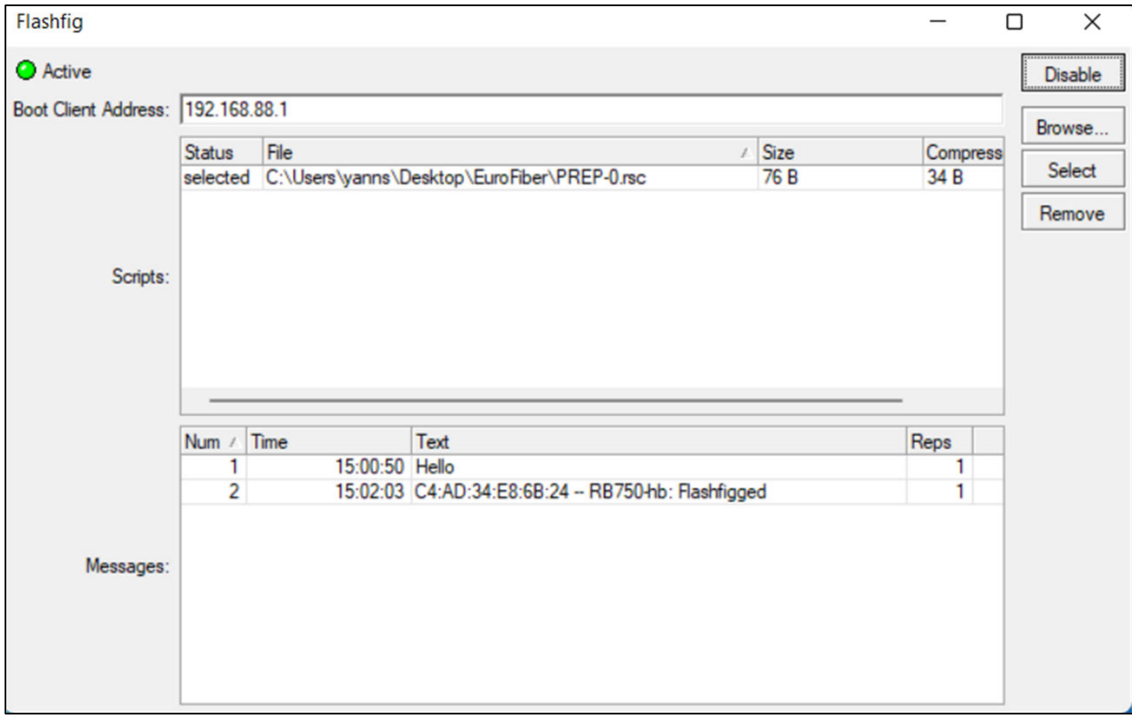
Status	File	Size	Compress
selected	C:\Users\yanns\Desktop\EuroFiber\PREP-0.rsc	76 B	34 B

Num	Time	Text	Reps
1	14:56:13	Hello	1

Azurtem

Flashfig (cont'd)

- If all goes well then you should see the router appear in the 'Messages' pane with an indication of its status : "Flashfigged"



● Active

Boot Client Address: 192.168.88.1

Scripts:

Status	File	Size	Compress
selected	C:\Users\yanns\Desktop\EuroFiber\PREP-0.rsc	76 B	34 B

Messages:

Num	Time	Text	Reps
1	15:00:50	Hello	1
2	15:02:03	C4:AD:34:E8:6B:24 -- RB750-hb: Flashfigged	1

Flashfig (cont'd)

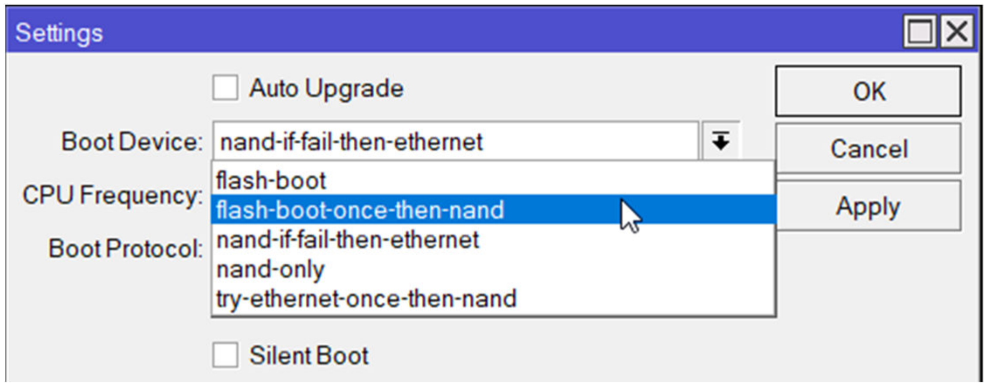
Recommendations

- Ensure that all interfaces on the PC running Flashfig are disabled or else you could run into the "no TFTP request" error.
- Disable any active DHCP server(s) present in the L2 segment
- On the switch, ensure that the stp mode is set to 'none' and that HW-offload is disabled for all ports.

Flashfig (cont'd)

- Whether you're executing Flashfig or NetInstall the targeted device(s) must have the boot mode set to 'FlashBoot'.
- This is set by default on brand new devices

```
system/routerboard/settings/set boot-device=flash-boot-once-then-nand
```


Settings

Auto Upgrade

Boot Device: nand-if-fail-then-ethernet

CPU Frequency:

Boot Protocol:

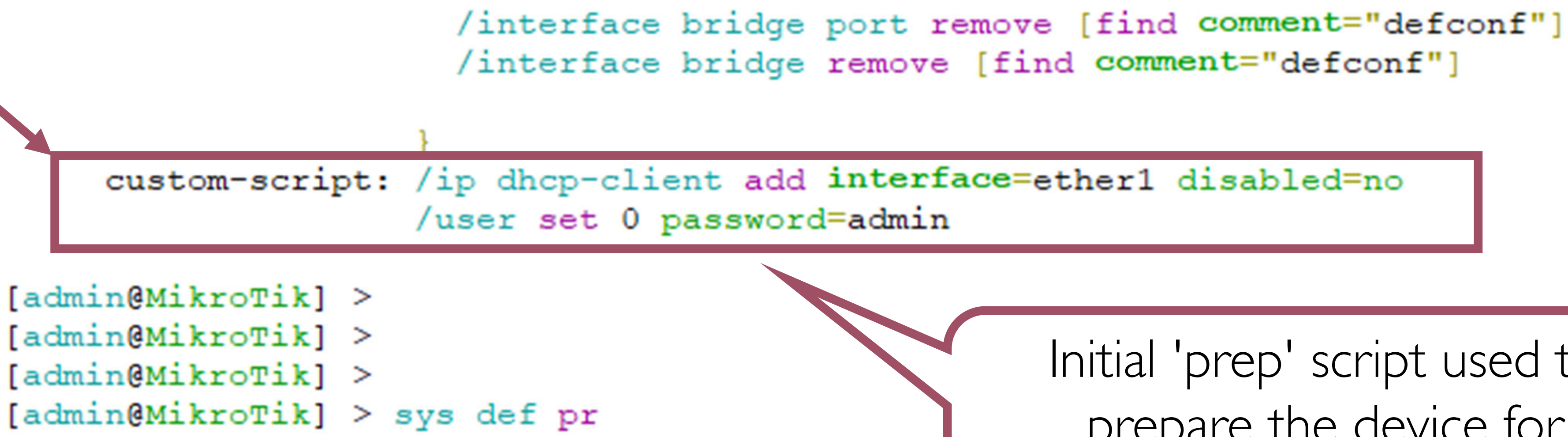
flash-boot
flash-boot-once-then-nand
nand-if-fail-then-ethernet
nand-only
try-ethernet-once-then-nand

Silent Boot

Azurtem

Flashfig (cont'd)

- When injecting your own default configuration script NetInstall/Flashfig will add it to the end of the existing (Mikrotik) default configuration :



```
                  /interface bridge port remove [find comment="defconf"]
                  /interface bridge remove [find comment="defconf"]

                }
   custom-script: /ip dhcp-client add interface=ether1 disabled=no
                  /user set 0 password=admin

[admin@MikroTik] >
[admin@MikroTik] >
[admin@MikroTik] >
[admin@MikroTik] > sys def pr
```

Initial 'prep' script used to prepare the device for RouterOS upgrade/downgrade

Ansible

- Opensource platform with which you can automate : software provisioning, configuration management and application deployment.

- Acquired by Red Hat in 2015, Ansible is designed to configure both Unix-like systems and Microsoft Windows hosts.

- Ansible is agentless, relying on remote connections via SSH or Windows Remote Management which allows PowerShell execution.

Azurtem

Ansible (cont'd)

- An Ansible playbook is publicly available allowing one to automate the upgrade and/or downgrade of RouterOS on Routerboards.
- Thanks to Zach Biles (Github: bile0026), an automation engineer and network architect for sharing this Playbook.

https://github.com/bile0026/mikrotik_upgrade/blob/master/README.md

https://github.com/bile0026/mikrotik_upgrade

ros_upgrade

etc/ansible/**hosts**

A.K.A. the 'inventory' file.

Defines which devices will be targeted by your playbook as well as some other useful parameters :

```
[routerboards]
192.168.89.[200:250]

[routerboards:vars]
ansible_connection=ansible.netcommon.network_cli
ansible_network_os=routeros
ansible_user=admin
ansible_ssh_pass=admin
deprecation_warnings=false
gather_facts=no
```

ros_upgrade (cont'd)

etc/ansible/ansible.cfg

Affects Ansible's behaviour, environment and preferences

```
[defaults]
host_key_checking = False

[ssh_connection]
pipelining = True
retries=2
```

Disables SSH host key checking

States how many times Ansible will attempt to execute a task on a remote host in case of connection difficulties

allows execution of multiple tasks over a single SSH connection. Sends commands/data sequentially without waiting for completion

ros_upgrade (cont'd)

- collections/: package containing reusable content

community.routeros

```
ansible-galaxy collection install -r collections/requirements.yml
```

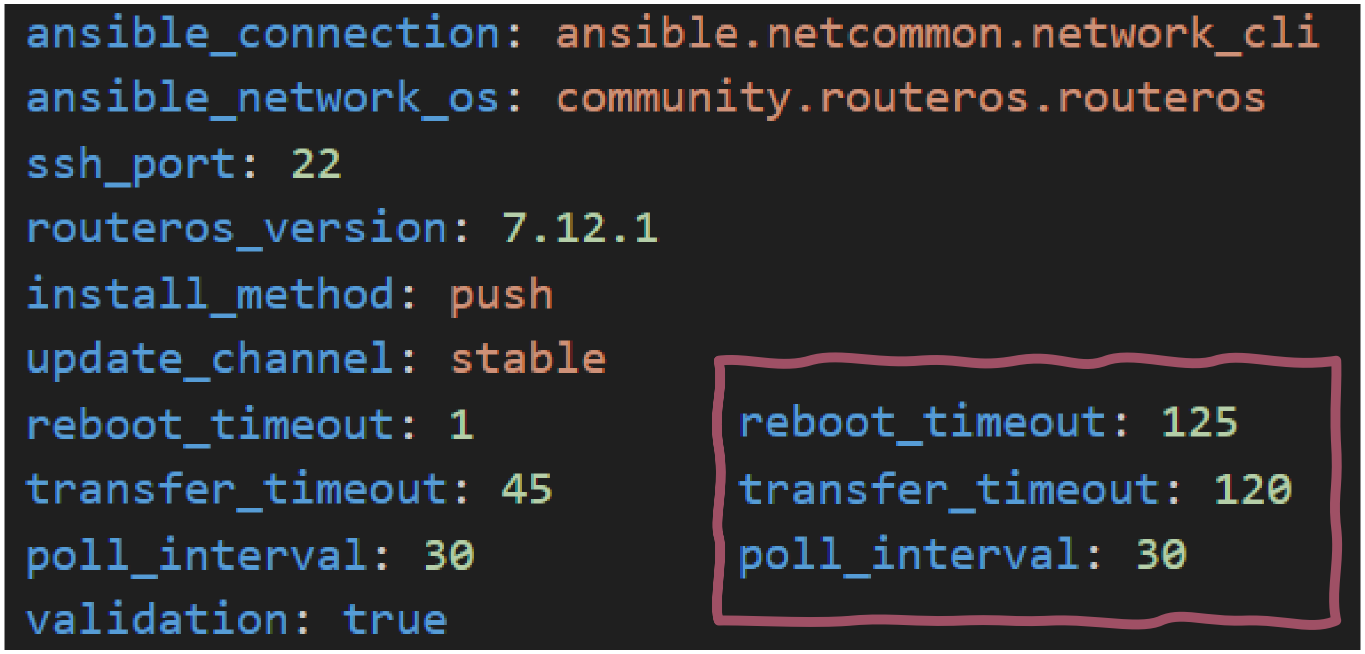
- ros_upgrade/defaults/: execution values used by the playbook e.g. ssh port, ROS version, install method, update channel, timeouts, …
- ros_upgrade/tasks/: install methods available to the playbook
- ros_upgrade/packages/: npk files available to the playbook

Instead of downloading the required npk files they can be stored in ros_upgrade/packages

ros_upgrade (cont'd)

Visual Studio Code with Mikrotik RouterOS script extension

ros_upgrade/defaults/main.yml:



```
ansible_connection: ansible.netcommon.network_cli
ansible_network_os: community.routeros.routeros
ssh_port: 22
routeros_version: 7.12.1
install_method: push
update_channel: stable
reboot_timeout: 1
transfer_timeout: 45
poll_interval: 30
validation: true
```

```
reboot_timeout: 125
transfer_timeout: 120
poll_interval: 30
```

ros_upgrade (cont'd)

ros_upgrade/tasks/

Main: initial method that calls the task that's is indicated in defaults:

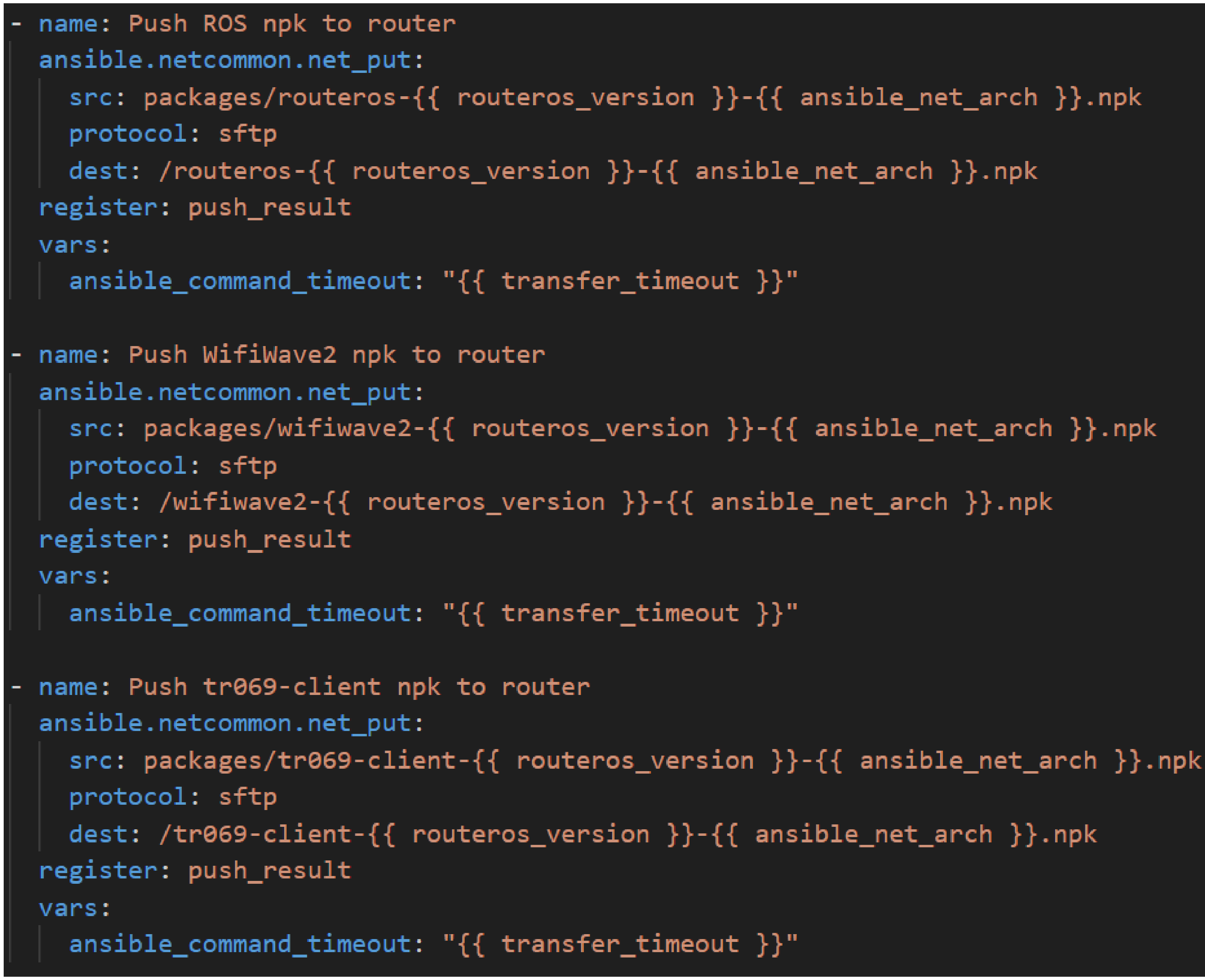
- normal: similar to using */system/packages/check for updates/Download*
- download: uses the router's Internet access for specific upgrade/downgrade
- push: uses Ansible's Internet access for specific upgrade/downgrade
- firmware: used to upgrade Routerboot
- validate: checks devices' resulting RouterOS and Routerboot versions

ansible-playbook routeros_upgrade.yml

Azurtem

ros_upgrade (cont'd)

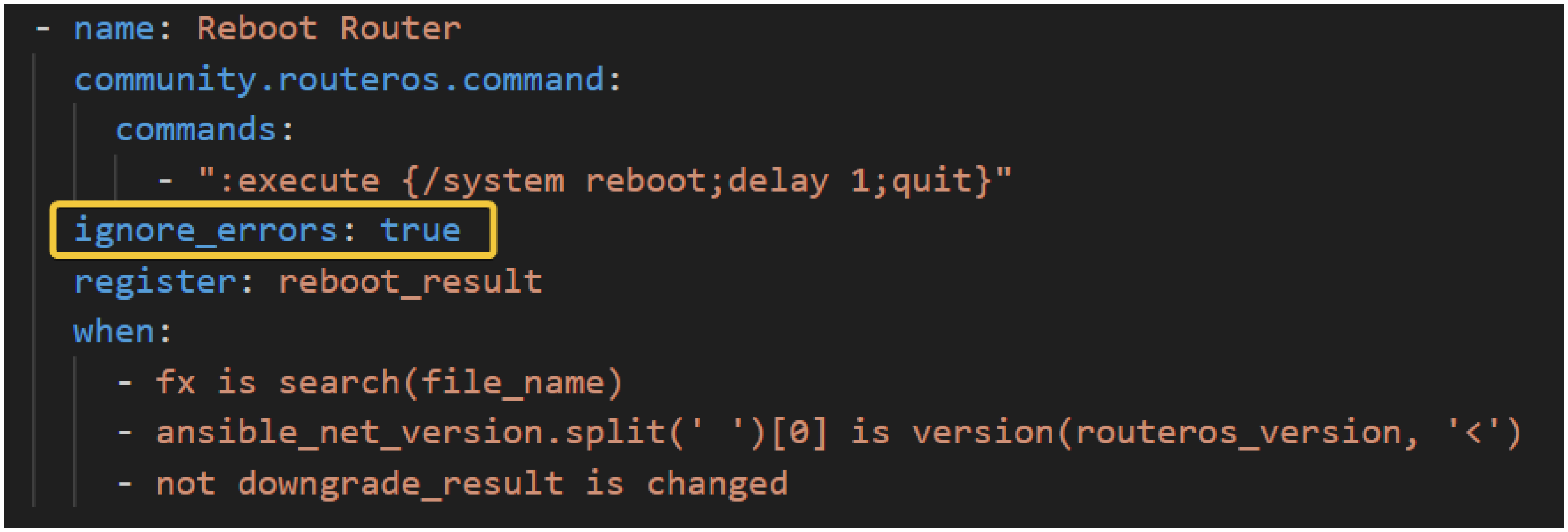
ros_upgrade/push.yml:



```
- name: Push ROS npk to router
  ansible.netcommon.net_put:
    src: packages/routeros-{{ routeros_version }}-{{ ansible_net_arch }}.npk
    protocol: sftp
    dest: /routeros-{{ routeros_version }}-{{ ansible_net_arch }}.npk
  register: push_result
  vars:
    ansible_command_timeout: "{{ transfer_timeout }}"

- name: Push WifiWave2 npk to router
  ansible.netcommon.net_put:
    src: packages/wifiwave2-{{ routeros_version }}-{{ ansible_net_arch }}.npk
    protocol: sftp
    dest: /wifiwave2-{{ routeros_version }}-{{ ansible_net_arch }}.npk
  register: push_result
  vars:
    ansible_command_timeout: "{{ transfer_timeout }}"

- name: Push tr069-client npk to router
  ansible.netcommon.net_put:
    src: packages/tr069-client-{{ routeros_version }}-{{ ansible_net_arch }}.npk
    protocol: sftp
    dest: /tr069-client-{{ routeros_version }}-{{ ansible_net_arch }}.npk
  register: push_result
  vars:
    ansible_command_timeout: "{{ transfer_timeout }}"
```

ros_upgrade (cont'd)

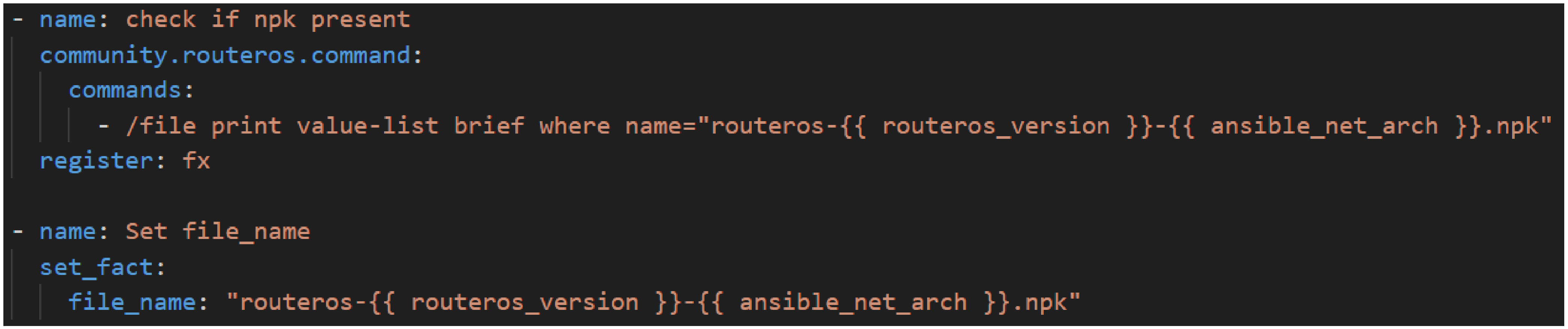
- The reboot caused the task to fail due to the loss of the SSH connection. The workaround was to ignore any errors caused by the action :



```yaml
- name: Reboot Router
  community.routeros.command:
    commands:
      - ":execute {/system reboot;delay 1;quit}"
  ignore_errors: true
  register: reboot_result
  when:
    - fx is search(file_name)
    - ansible_net_version.split(' ')[0] is version(routeros_version, '<')
    - not downgrade_result is changed
```

ros_upgrade (cont'd)

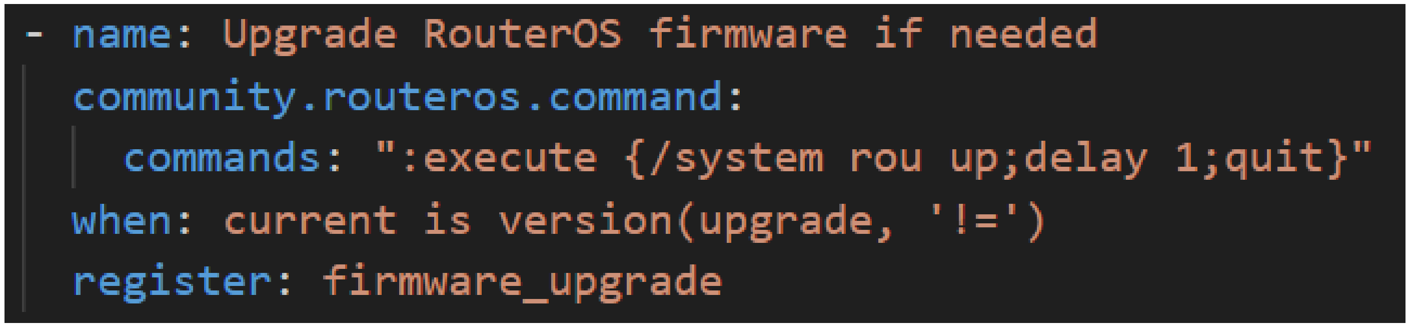
- The reboot wouldn't execute because the verification of the transferred RouterOS npk file would fail. Updated approach :



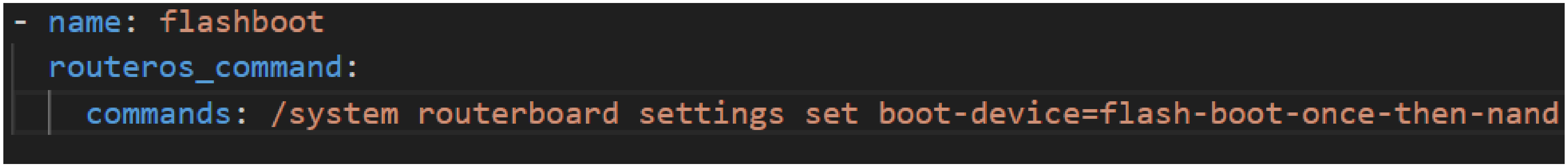
```
- name: check if npk present
  community.routeros.command:
    commands:
      - /file print value-list brief where name="routeros-{{ routeros_version }}-{{ ansible_net_arch }}.npk"
  register: fx

- name: Set file_name
  set_fact:
    file_name: "routeros-{{ routeros_version }}-{{ ansible_net_arch }}.npk"
```

ros_upgrade (cont'd)

ros_upgrade/firmware.yml:



```
- name: Upgrade RouterOS firmware if needed
  community.routeros.command:
    commands: ":execute {/system rou up;delay 1;quit}"
  when: current is version(upgrade, '!=')
  register: firmware_upgrade
```

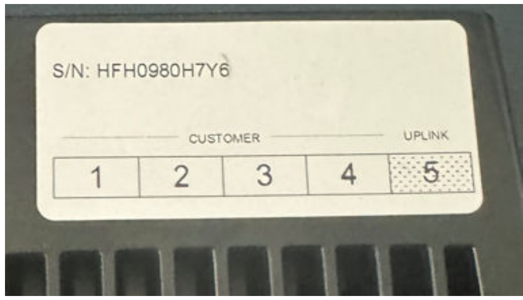


```
- name: flashboot
  routeros_command:
    commands: /system routerboard settings set boot-device=flash-boot-once-then-nand
```

```
ansible-playbook routeros_upgrade.yml -t firmware
ansible-playbook routeros_upgrade.yml -t flashboot
```

Single task execution

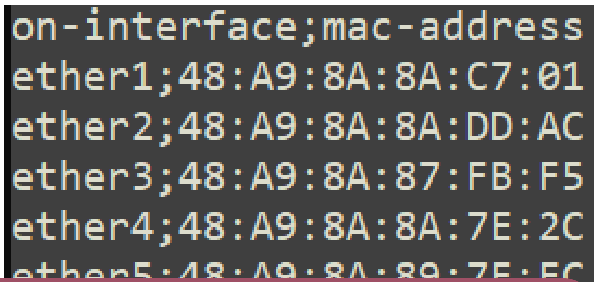
Labels



- Labels allow the client to quickly identify the prepared devices.
- Can be applied on the boxes or on the devices themselves.
- May display an ID, a s/n or a MAC address; even a bar/QR code
- They are generally applied at the end of the preparation so keeping the 50x devices sorted in the proper order is essential.

48A98A40C2AD	48A98A40C73D	48A98A40C287	18FD74B26357
48A98A40CC11	48A98A40C546	48A98A40CBB2	48A98A40C274

Labels (cont'd)



```
on-interface;mac-address
ether1;48:A9:8A:8A:C7:01
ether2;48:A9:8A:8A:DD:AC
ether3;48:A9:8A:87:FB:F5
ether4;48:A9:8A:8A:7E:2C
```

Collect the data:

Once the devices are connected and active collect the 50x Interface + MAC addresses

```
{:local ethlist;
:local buffer;
:local hour [/system clock get time];
:local h [pick $hour 0 2];
:local m [pick $hour 3 5];
:local fileName "address-list ($h-$m)";
:set $buffer ("on-interface".";"."mac-address"."\n");
:foreach i1 in [/interface ethernet find where running=yes] do={ :set $ethlist [/interface ethernet get $i1 name];
:foreach i2 in=$ethlist do={:foreach i3 in=[/interface bridge host print prop=on-interface,mac-address as-value
where interface=$i2 dynamic !local] do={:set $buffer ($buffer.$i3->"on-interface".";".$i3->"mac-address"."\n");}}}
/file print file=$fileName where name="";
delay 1s;
/file set $fileName contents=$buffer;}
```

Collect the 50x Interface + MAC addresses + identities

```
ip neighbor pr fil serial-list.txt proplist="interface,mac-address,identity" without-paging
```

Labels (cont'd)

- Labels can be printed using a thermal printer and then applied on each device/box using a label applicator.
- Microsoft Word can be used to merge and print sheets of labels. The merge however fills the sheet from left to right and then top to bottom.
- We used Barcode Studio from Tec-IT to generate bar/QR codes and print the labels; in an order that suited us best.

Lessons learnt

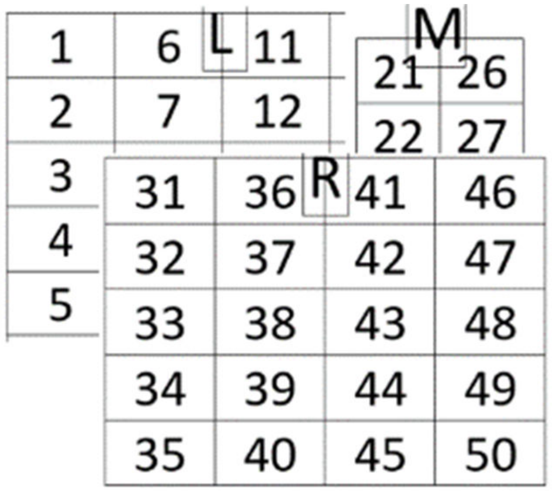
- An installation session 'flow' can be hampered by unforeseen circumstances so plan for potential contingencies.
- Have a second PC on hand to continue the deployment process in case of failure with your main PC.
- Backup the switch's configuration, for future sessions and to quickly recover from a switch break down.
- Have a plan B to print the labels to ensure that the shipment of the prepared devices won't be delayed.

Azurtem

Lessons learnt (cont'd)

- Map your device layout to maintain consistency and to facilitate maintenance when required.

- Group your cables together to avoid confusion and potential device identification errors



L

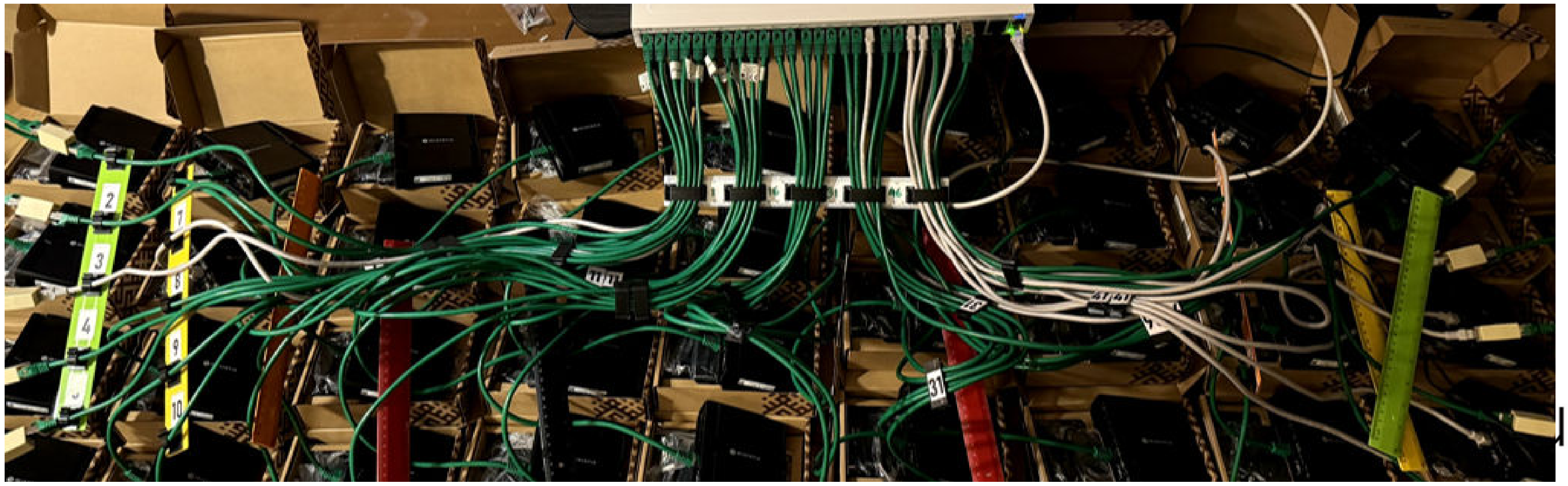
1	6	11
2	7	12
3		
4		
5		

M

21	26
22	27

R

31	36	41	46
32	37	42	47
33	38	43	48
34	39	44	49
35	40	45	50



Lessons learnt (cont'd)

- If the switch is started up with the client devices already plugged in then by default these will also be powered up.

  The switch however won't be quick enough to activate the L2 network segment and thus Flashfig won't be able to flash the devices.

  The result is that devices will no longer be in Flashboot mode. To fix this reboot each device using the reset button to enable NetInstall mode.

Begin a new session by setting "poe-out=off" on all Ethernet ports

Azurtem

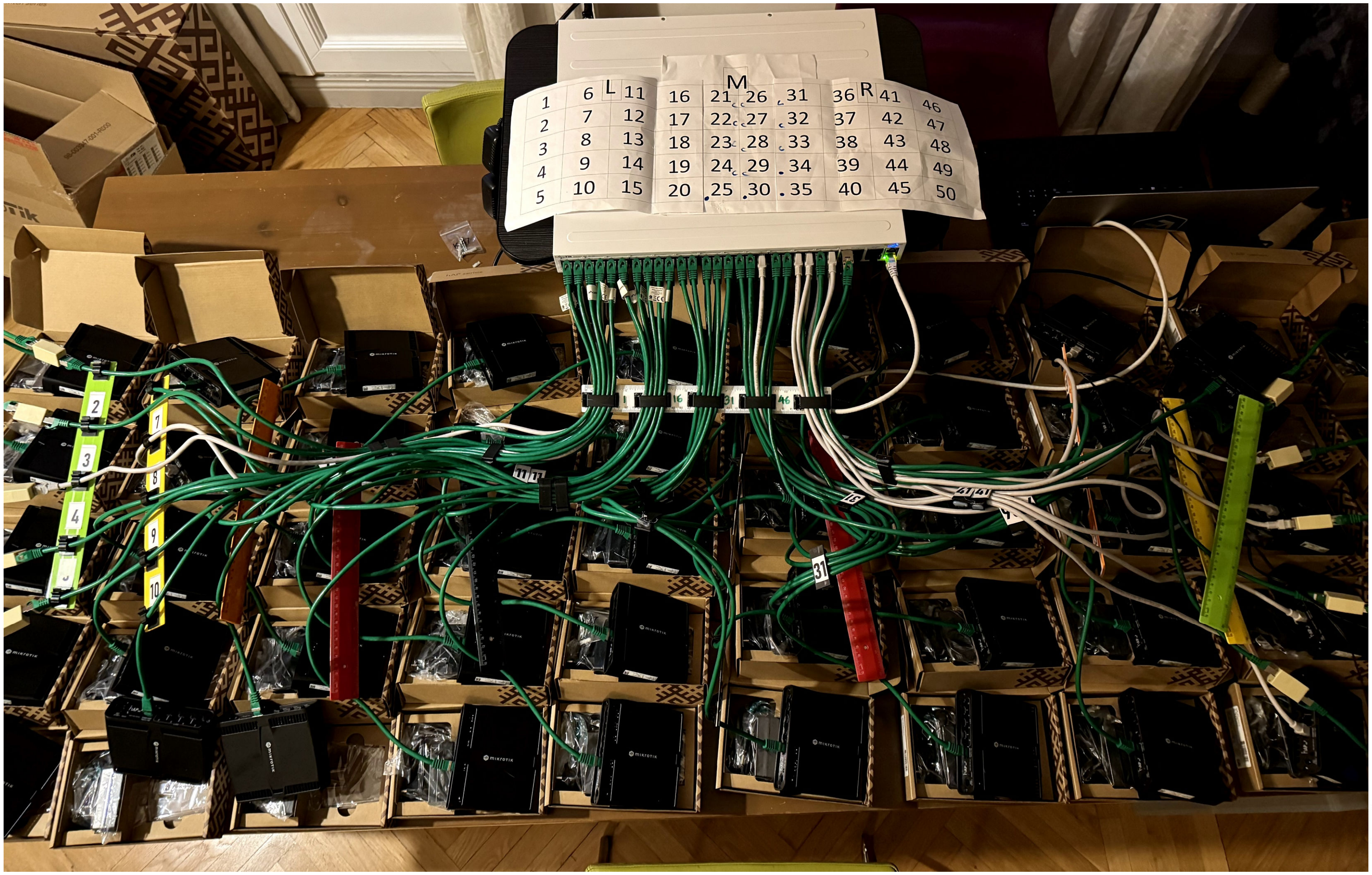
Lessons learnt (cont'd)

- Execute Flashfig with a single unit first to ensure success. The two SFP+ ports are prime candidates for such tests.

- Some devices won't respond (well) to Flashfig/Ansible – you may have to deal with them manually/individually (NetInstall).

- Check whether you can reboot the devices after flashing the customer's configuration; the initial boot sequence may be programmed to adapt the router's configuration based on available links.

Visuals



rtem

Visuals (cont'd)

questions ?