Lab Docs, Tutorials/Guides and Notes

This is where most documentation about my lab and experiences will be stored. for archival and as a free knowledge base to anyone who might stumble upon it

Setting up a tailscale/headscale exit-node

On the client

The example below is for linux client

1. install tailscale using your method of choice, I picked the one liner script

   ```
   # one liner script from tailscale
   curl -fsSL https://tailscale.com/install.sh | sh

   # If for some reason you can't use the one liner script to install it, refer to the official docs for manually installing at https://tailscale.com/download/linux/debian-bookworm
   ```

2. run this command to initiate the tailscale client with a custom control server

   ```
   # Sample
   tailscale up --login-server "http://your-headscale-serverip:port" --advertise-exit-node

   # Since I run headscale on the same machine I want to use as an exit-node, I set it as localhost
   tailscale up --login-server "http://localhost:8082" --advertise-exit-node

   # If the node is already registered, then we can advertise it as an exit-node in this manner
   tailscale set --advertise-exit-node
   ```

On the server

1. First we confirm that our nodes are seen and their routes are either `0.0.0.0/0` and/or `::/0`

```
# Check routes using this command
headscale routes list

# Output
ID | Machine            | Prefix    | Advertised | Enabled | Primary
1  | exit-node1         | 0.0.0.0/0 | true       | false   | -
2  | exit-node1         | ::/0      | true       | false   | -
3  | phone1             | ::/0      | false      | false   | -
4  | phone1             | 0.0.0.0/0 | false      | false   | -
5  | phone2             | ::/0      | false      | false   | -
6  | phone2             | 0.0.0.0/0 | false      | false   | -
```

As we can confirm here, the routes are advertised, but not enabled.

2. To enable the routes, we will use the following command to allow the exit-node to advertise itself, you will have to repeat this for every exit-node you want to add.

```
# First select the route you wanna add, in my case I want both IPv4 and IPv6 so I'll
add id 1 and 2
ID | Machine            | Prefix    | Advertised | Enabled | Primary
1  | exit-node1         | 0.0.0.0/0 | true       | false   | -
2  | exit-node1         | ::/0      | true       | false   | -
3  | phone1             | ::/0      | false      | false   | -
4  | phone1             | 0.0.0.0/0 | false      | false   | -
5  | phone2             | ::/0      | false      | false   | -
6  | phone2             | 0.0.0.0/0 | false      | false   | -

# Command
headscale routes enable -r 1
headscale routes enable -r 2

# Confirm we enabled it
headscale routes list

ID | Machine            | Prefix    | Advertised | Enabled | Primary
1  | exit-node1         | 0.0.0.0/0 | true       | true    | -
2  | exit-node1         | ::/0      | true       | true    | -
3  | phone1             | ::/0      | false      | false   | -
4  | phone1             | 0.0.0.0/0 | false      | false   | -
5  | phone2             | ::/0      | false      | false   | -
6  | phone2             | 0.0.0.0/0 | false      | false   | -
```

3. Test with your client to see if it worked, with these steps done any client should be able to utilize your new exit-node

IP Index (WIP)

This Document Outlines plenty of publicly recognized IPv4 and IPv6 Addresses

IPv4 Space

Address Block	Address range	Number of Addresses	Scope	Description
0.0.0.0/8	0.0.0.0-0.255.255.255	16,777,216	Software	Current
10.0.0.0/8	10.0.0.0–10.255.255.255	16,777,216	Private network	Used for local communications within a private network.
100.64.0.0/10	100.64.0.0–100.127.255.255	4,194,304	Private network	Shared address space for communications between a service provider and its subscribers when using a carrier-grade NAT.
127.0.0.0/8	127.0.0.0–127.255.255.255	16,777,216	Host	Used for loopback addresses to the local host.
169.254.0.0/16	169.254.0.0–169.254.255.255	65,536	Subnet	Used for link-local addresses between two hosts on a single link when no IP address is otherwise specified, such as would have normally been retrieved from a DHCP server.
172.16.0.0/12	172.16.0.0–172.31.255.255	1,048,576	Private network	Used for local communications within a private network.

192.0.0.0/24	192.0.0.0–192.0.0.255	256	Private network	IETF Protocol Assignments, DS-Lite (/29).
192.0.2.0/24	192.0.2.0–192.0.2.255	256	Documentation	Assigned as TEST-NET-1, documentation and examples.
192.88.99.0/24	192.88.99.0–192.88.99.255	256	Internet	Reserved. Formerly used for IPv6 to IPv4 relay (included IPv6 address block 2002::/16).
192.168.0.0/16	192.168.0.0–192.168.255.255	65536	Private network	Used for local communications within a private network.
198.18.0.0/15	198.18.0.0–198.19.255.255	131,072	Private network	Used for benchmark testing of inter-network communications between two separate subnets.
198.51.100.0/24	198.51.100.0–198.51.100.255	256	Documentation	Assigned as TEST-NET-2, documentation and examples.
203.0.113.0/24	203.0.113.0–203.0.113.255	256	Documentation	Assigned as TEST-NET-3, documentation and examples.
224.0.0.0/4	224.0.0.0–239.255.255.255	268,435,456	Internet	In use for IP multicast. (Former Class D network.)
233.252.0.0/24	233.252.0.0-233.252.0.255	256	Documentation	Assigned as MCAST-TEST-NET, documentation and examples.
240.0.0.0/4	240.0.0.0–255.255.255.254	268,435,455	Internet	Reserved for future use. (Former Class E network.)
255.255.255.255/32	255.255.255.255	1	Subnet	Reserved for the "limited broadcast" destination address.

IPv6 Space

Address Block	First address	Last Address	Number of addresses	Scope	Description
::/128	::	::	1	Software	Unspecified address
::1/128	::1	::1	1	host	Loopback address—a virtual interface that loops all traffic back to itself, the local host
::ffff:0:0/96	::ffff:0.0.0.0	::ffff:255.255.255.255	2^{32}	Software	IPv4-mapped addresses
::ffff:0:0:0/96	::ffff:0:0.0.0.0	::ffff:0:255.255.255.255	2^{32}	Software	IPv4 translated addresses
64:ff9b::/96	64:ff9b::0:0:0:0	64:ff9b::255.255.255.255	2^{32}	Global Internet	IPv4/IPv6 translation
64:ff9b:1::/48	64:ff9b:1::	64:ff9b:1:ffff:ffff:ffff:ffff:ffff	$2,^{80}$ with 2^{48} for each IPv4	Private internets	IPv4/IPv6 translation
100::/64	100::	100::ffff:ffff:ffff:ffff	2^{64}	Routing	Discard prefix
2001:0000::/32	2001::	2001::ffff:ffff:ffff:ffff:ffff:ffff	2^{96}	Global Internet	Teredo tunneling
2001:20::/28	2001:20::	2001:2f:ffff:ffff:ffff:ffff:ffff:ffff	2^{100}	Software	ORCHIDv2

2001:db8::/32	2001:db8::	2001:db8:ffff:ffff:ffff:ffff:ffff:ffff	2^{96}	Documentation	Addresses used in documentation and example source code
2002::/16	2002::	2002:ffff:ffff:ffff:ffff:ffff:ffff:ffff	2^{112}	Global Internet	The 6to4 addressing scheme (deprecated)
fc00::/7	fc00::	fdff:ffff:ffff:ffff:ffff:ffff:ffff:ffff	2^{121}	Private Internets	Unique local address
fe80::/64 from fe80::/10	fe80::	fe80::ffff:ffff:ffff:ffff	2^{64}	Link	Link-local address
ff00::/8	ff00::	ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff	2^{120}	Global Internet	Multicast address

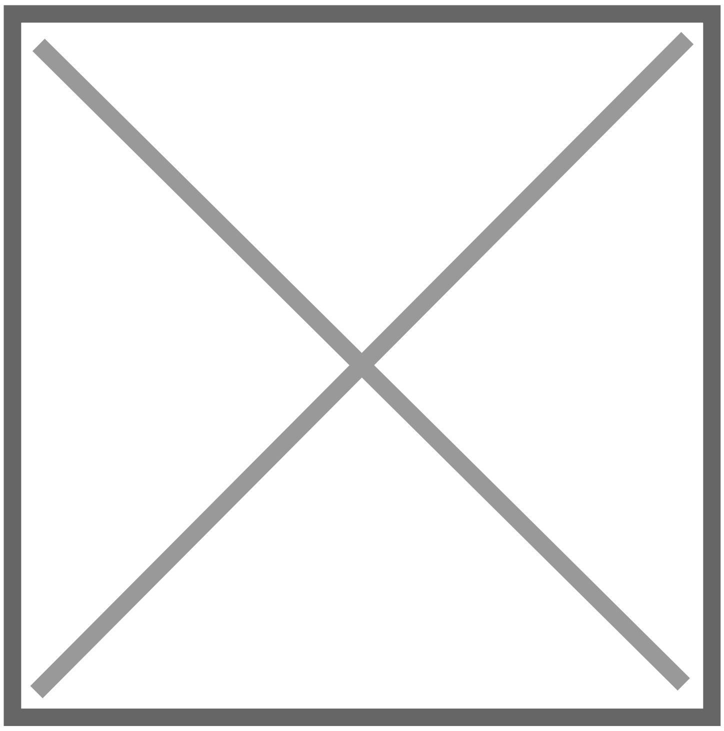
Webhooks

Affiliates Management

On this section, we will go over the steps needed to manage the A-Chan webhook seen in the affiliates channel

Step 1 is to have the webhook link copied, the webhook link is provided to **librarians only.**

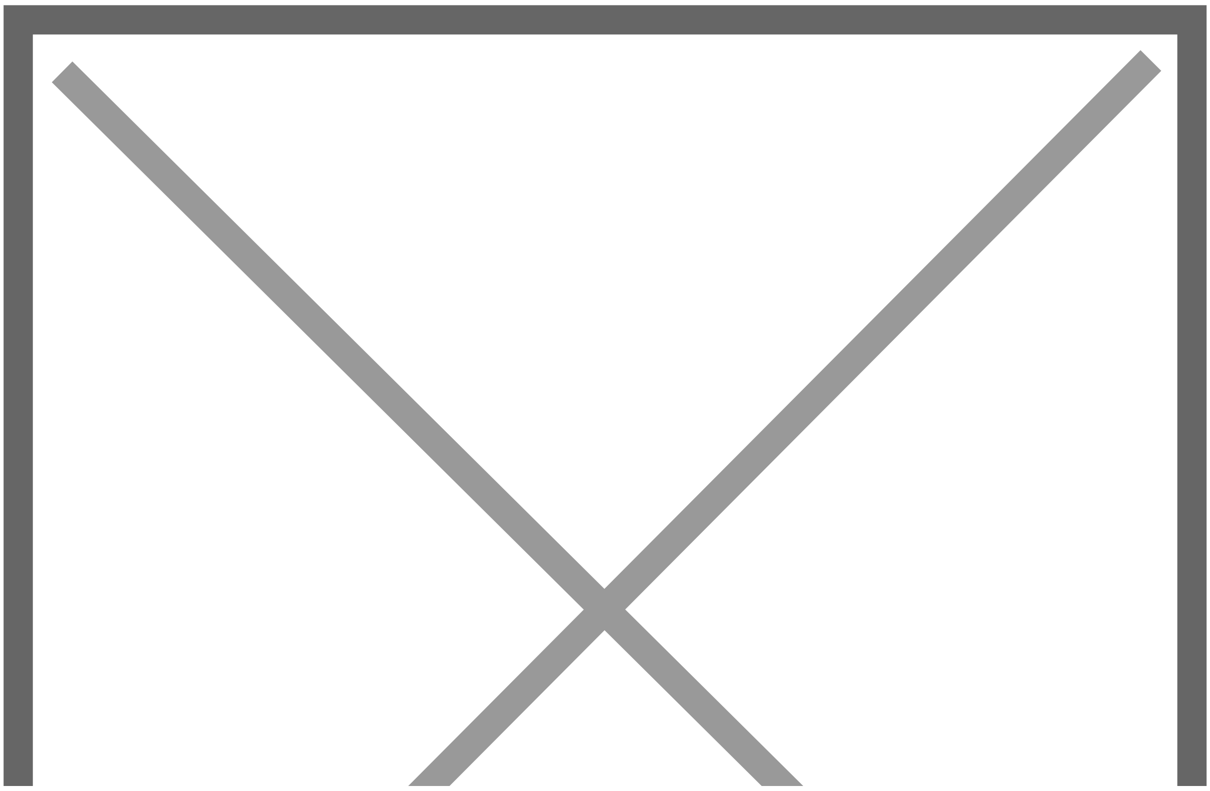
Step 2 is to head over to https://discohook.org, The interface will be a bit intimidating at first but I will go over the steps needed to manipulate the webhook



Step 3, over the top left most part of the website, you will see a Webhook URL text box

You will paste the webhook link that is provided to you in this box

Step 4, Head over to the affiliates channel and select the thread of your choice, then right click the thread title and select copy ID. See pictures below



Step 5, Head back to the discohook site and paste the following after the Webhook URL
?thread_id=<the-thread_id-you-copied>

example:

Step 6, with these steps done, you may now send a new affiliate link with A-Chan webhook, Use the content box for the affiliate link, an example is provided below:
__Example Server__
https://discord.gg/123abc

Step 7 (Optional), IF you need to edit an existing affiliate link, follow this procedure.
Step 7.a Make sure the message is in the same thread that you're editing, right click the message and copy the message link

Step 7.b Paste the message link into the corresponding box in discohook and click the "Load"

Disabling "Predictable" Network Interface names

Introduction

Network interface names in linux are generally named something like eth0, eth1, wlan0, wlan1 and so on...

however with some recent changes, some Operating systems may have this slight annoyance baked in called "Predictable Network Interface Names" or in my book, making my experience **worse**

This guide will cover two ways to disable ifnames and is mainly aimed at **Proxmox VE and Debian**

GRUB method

For Proxmox VE, we have two choices for boot methods. We have the traditional Legacy BIOS boot which is usually accompanied by GRUB and the second option is EFI boot which is usually handled by efibootmgr

to disable "Predictable" Names, use your text editor of choice, in my case it's nano, we will edit the default GRUB configuration which is usually found in the /etc/default directory

```
nano /etc/default/grub
```

my configuration looks like this

```
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
GRUB_TIMEOUT=3
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet"
GRUB_CMDLINE_LINUX=""
```

```
# Uncomment to enable BadRAM filtering, modify to suit your needs
# This works with Linux (no patch required) and with any kernel that obtains
# the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"

# Uncomment to disable graphical terminal (grub-pc only)
#GRUB_TERMINAL=console

# The resolution used on graphical terminal
# note that you can use only modes which your graphic card supports via VBE
# you can see them in real GRUB with the command `vbeinfo'
#GRUB_GFXMODE=640x480

# Uncomment if you don't want GRUB to pass "root=UUID=xxx" parameter to Linux
#GRUB_DISABLE_LINUX_UUID=true

# Uncomment to disable generation of recovery mode menu entries
#GRUB_DISABLE_RECOVERY="true"

# Uncomment to get a beep at grub start
#GRUB_INIT_TUNE="480 440 1"
```

However what we're really interested in is the following lines

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet"
GRUB_CMDLINE_LINUX=""
```

We want to add `net.ifnames=0` to `GRUB_CMDLINE_LINUX_DEFAULT` which handles the default commandline options and passes it to our Proxmox GRUB entry. Your new config should look something like this

```
GRUB_DEFAULT=0
GRUB_TIMEOUT=3
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet net.ifnames=0"
GRUB_CMDLINE_LINUX=""
```

After applying these changes, you have to update your GRUB entries by using the following command

```
update-grub
```

After applying the changes using the `update-grub` command you should apply some changes to your Proxmox VE node before rebooting, this will be outlined on the last step on this guide.

EFI method

If you are using UEFI then this guide is for you.

The guide for EFI is pretty simple as well. just like before It involves just editing a single config and updating boot entries

For EFI boot on Proxmox VE, We need to edit the `/etc/kernel/cmdline` config. use your choice of text editor.

```
nano /etc/kernel/cmdline
```

Upon viewing the file you should see your default config. In my case it contains ZFS entries as I used ZFS as the boot filesystem

```
root=ZFS=rpool/ROOT/pve-1 boot=zfs
```

We only need to append to the end `net.ifnames=0` like so

```
root=ZFS=rpool/ROOT/pve-1 boot=zfs net.ifnames=0
```

After making this change, since we're on Proxmox VE just use the built in command which is

```
proxmox-boot-tool refresh
```

This should automatically update entries for you, after which you need to apply network config changes

Updating Network config to reflect changes

After updating our boot entries to use the traditional naming scheme, we must also update our network changes to use the old names.

to do this we have to edit the `/etc/network/interfaces` file and replace the old interface with the new one

```
source /etc/network/interfaces.d/*

auto lo
iface lo inet loopback

auto enp8s0
iface enp8s0 inet manual

auto enp9s0
iface enp9s0 inet manual

auto bond0
iface bond0 inet manual
        bond-slaves enp8s0 enp9s0
        bond-miimon 100
        bond-mode active-backup
        bond-primary enp8s0

auto vmbr0
iface vmbr0 inet static
        address 192.2.0.3/24
        gateway 192.2.0.1
        bridge-ports bond0
        bridge-stp off
        bridge-fd 0
        bridge-vlan-aware yes
        bridge-vids 2-4094
        mtu 9000
#physical network
```

In my configuration, I have the `enp8s0` and `enp9s0` interfaces from my supermicro board with 2 LAN ports bonded for redundancy via a software bond. all I have to do is substitute the new values/names into the config. If you're confused which one is your actual interface then use the command `ip a` this will show you all your interfaces and their altnames aswell

```
2: eth0: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master bond0 state
UP group default qlen 1000
    link/ether aa:bb:cc:dd:ee:ff brd ff:ff:ff:ff:ff:ff
    altname enp8s0
3: eth1: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master bond0 state
```

```
UP group default qlen 1000
    link/ether aa:bb:cc:dd:ee:f0 brd ff:ff:ff:ff:ff:ff permaddr aa:bb:cc:dd:ee:f0
    altname enp9s0
```

now my configuration looks like this

```
source /etc/network/interfaces.d/*

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet manual

auto eth1
iface eth1 inet manual

auto bond0
iface bond0 inet manual
        bond-slaves eth0 eth1
        bond-miimon 100
        bond-mode active-backup
        bond-primary eth0

auto vmbr0
iface vmbr0 inet static
        address 192.2.0.3/24
        gateway 192.2.0.1
        bridge-ports bond0
        bridge-stp off
        bridge-fd 0
        bridge-vlan-aware yes
        bridge-vids 2-4094
        mtu 9000
#physical network
```

After updating your network config, you can safely reboot your Proxmox VE node and connect to it as usual!