WireGuard VPN using IPv6 and OSPF

Connect your home with the Oracle Cloud Free Tier

DOAG 2022

#whoami

Martin Schmitter, Solutions Architect - Database

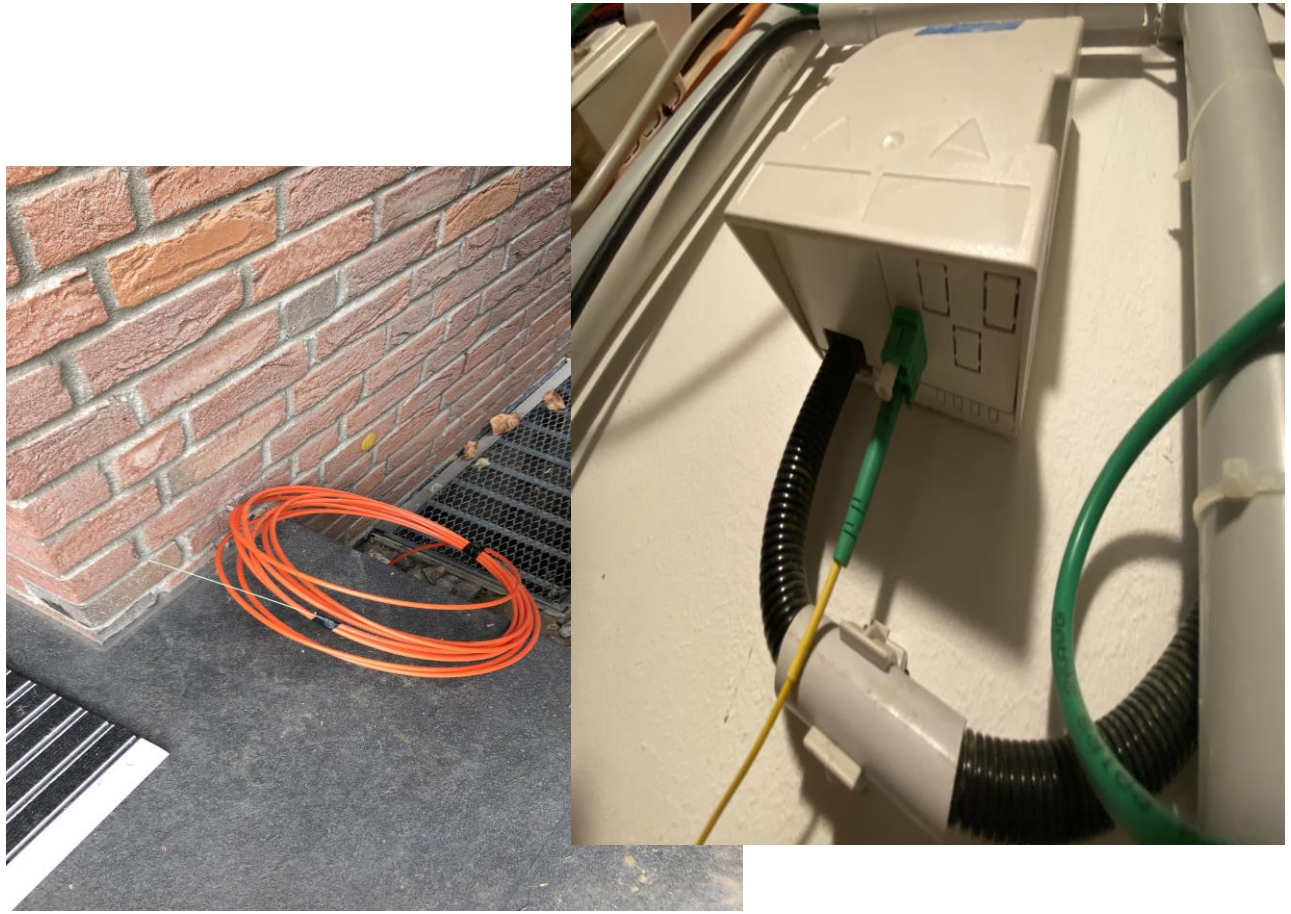
- Information Scientist, application development, Jan 2001
- ~20y in professional IT (~30y non-professional)
- HAM radio amateur (14y)
- My first Oracle version = v7.3
- Cisco Certified Network Associate – CCNA (2002-2005)
- DOAG member >2010
- Local Representative DOAG Regio NRW 2016-2022
- Oracle ACE >2019

Oracle ACE Pro

Safe Harbour Statement

- All statements are representing my opinion and will not represent or reflect any strategy, direction or architecture of any company I’m working with.
- All statements are made in general purpose, based on my own observations and personal experience and will not be specific to an enterprise, project or an individual.

Motivation



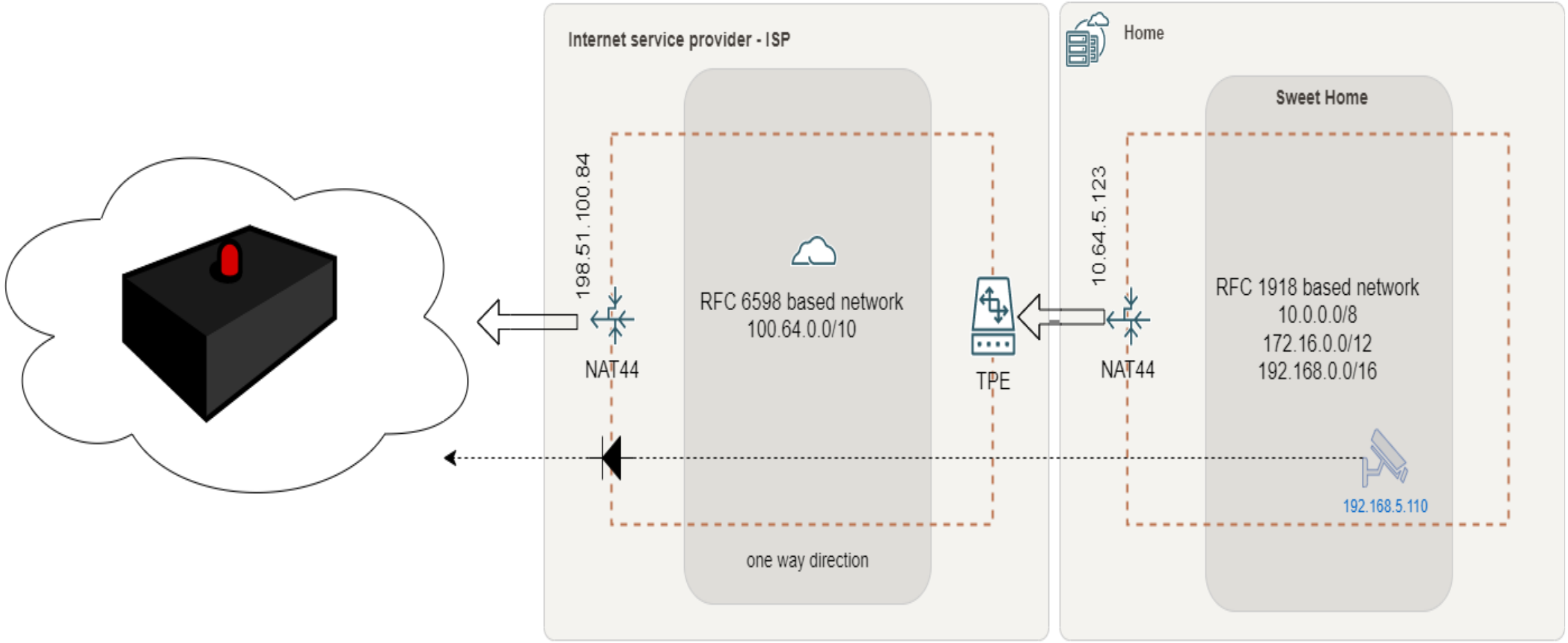
Something has changed!

“It’s not a bug, it’s a feature!”

With the new **FTTH connection** and the new **Internet service provider**, I lost my dynamic and public IPv4 address to access the Internet:

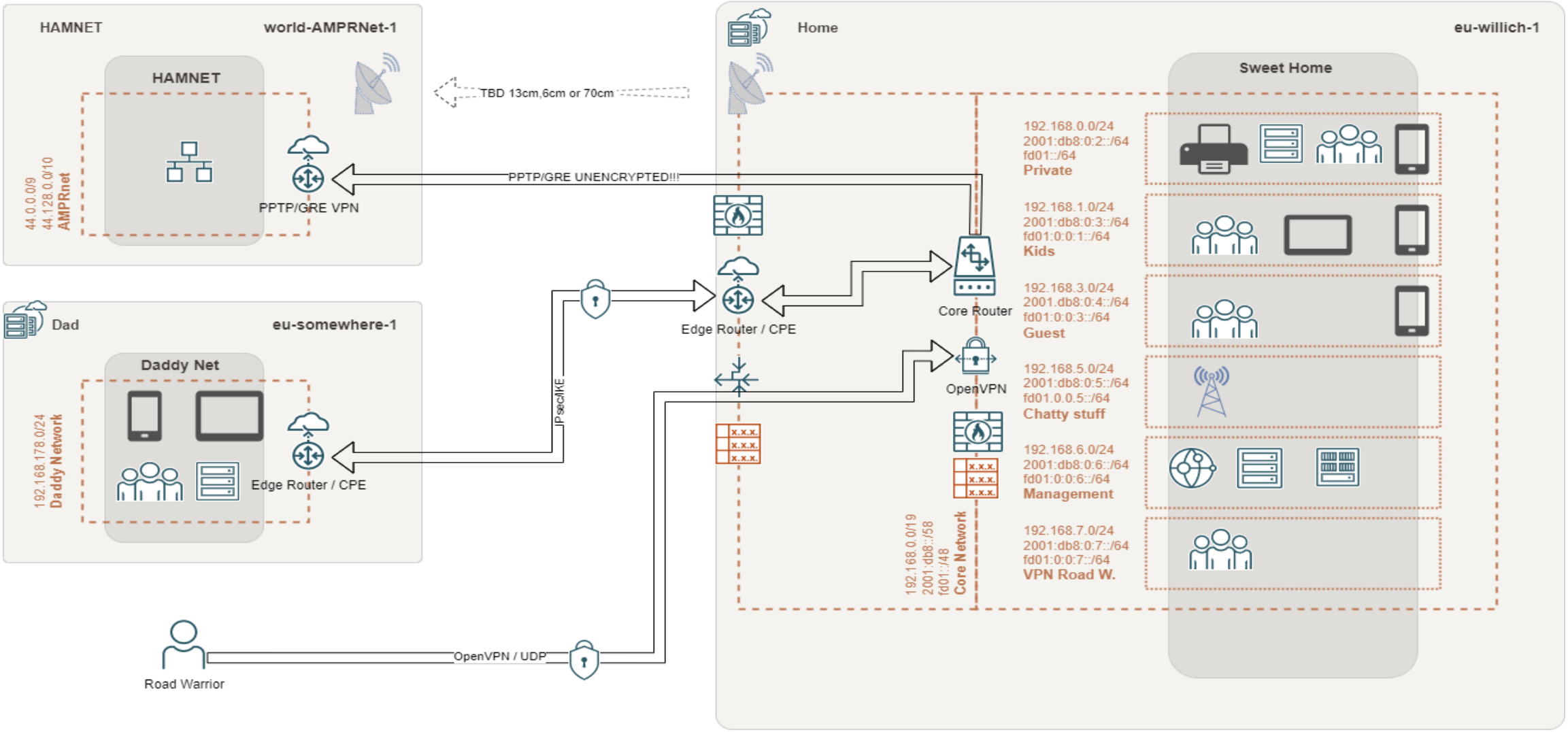
- IPv4 based access is realized based on a **Carrier-grade NAT** using RFC6598 based address space
  - `10.64.0.0/10`
- In consequence, I **can’t** reach my private network!
  - VPN (Site2Site, Road Warrior)
  - Services
  - IoT & smart things
  - VoIP
  - Hamnet
  - …

Carrier-grade NAT



Architecture Overview

My Sweet Home



Deal with it!

Statement of presenter

- The following solutions have been developed over a period of two years. Technology is moving quick! There isn’t doubt, that better solutions might appear, even during the preparation of this presentation.
- E.g., broad implementation of native IPv6 on cell phone carrier, ISP and CSP level solved a lot of issues.

“I’m lucky!”

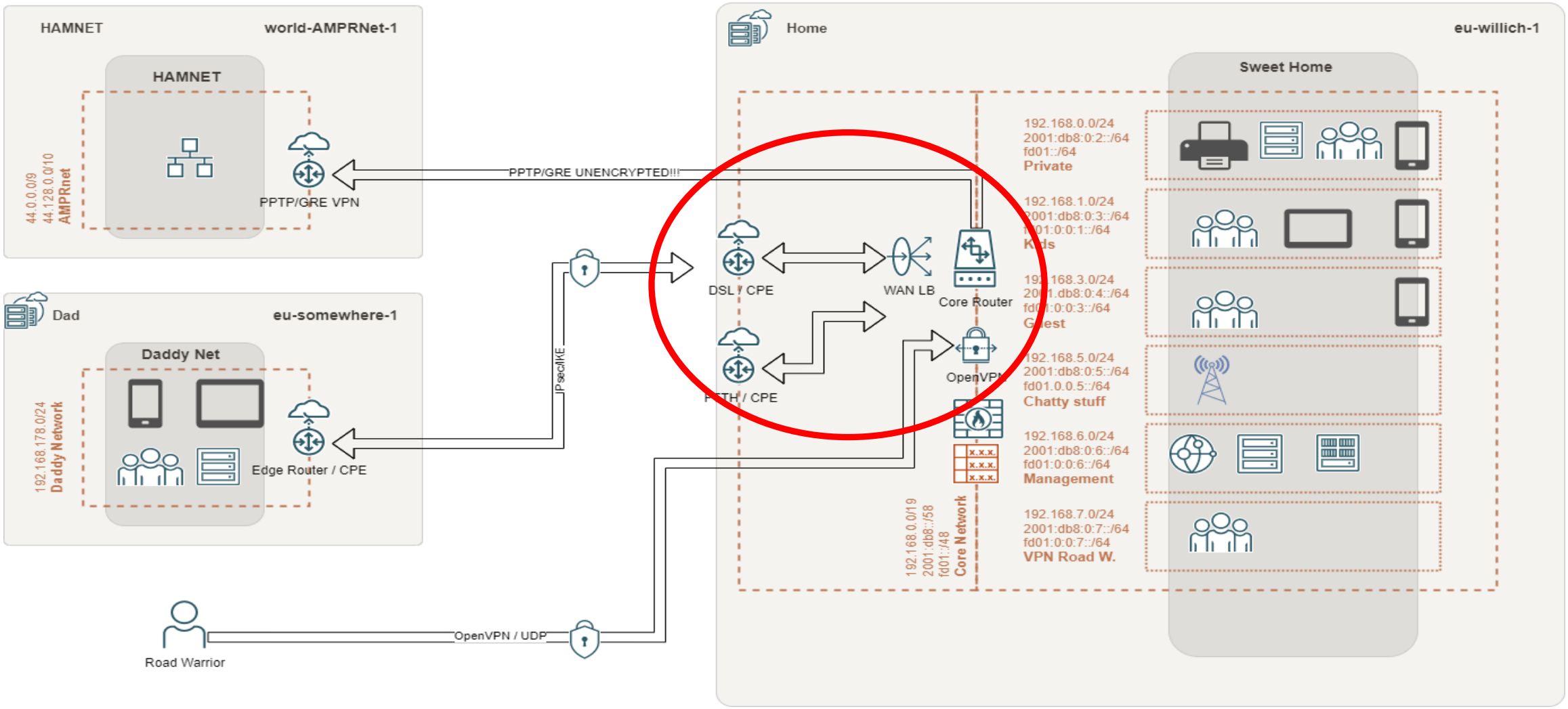
Based on the **contract** with my new ISP, I’ve a period of 6 months with two internet connections in parallel!

- Old connection: 50Mb/s VDSL
- New connection: 400Mb/s FTTH
  - soft limit – 1Gb/s possible

Hold my beer!

Architecture Overview

My Sweet Home - WAN LB



Dual-WAN, Failover and Load-Balancing

A **WAN LB** is quite tricky, but did do the job:

Make yourself familiar with:

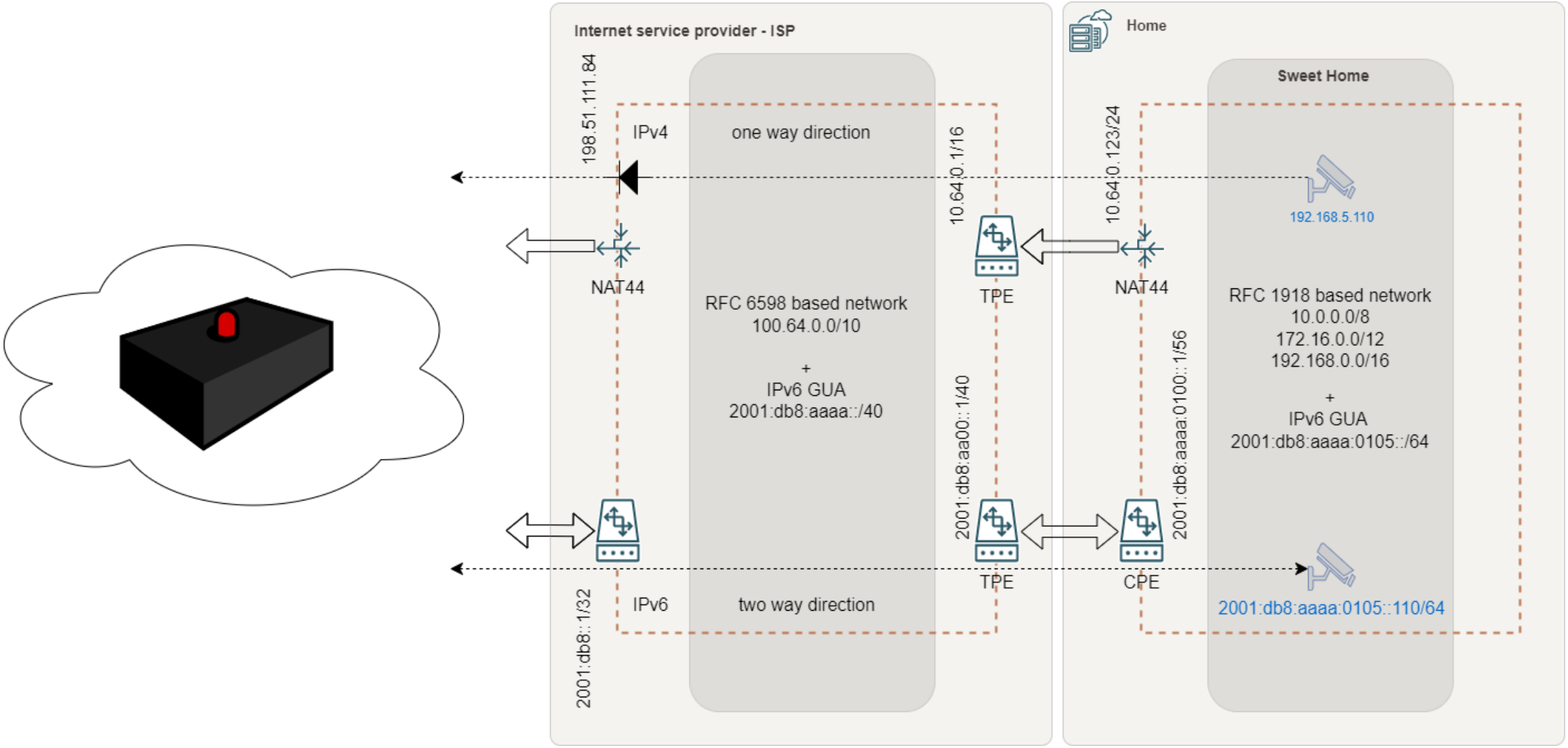
- **Policy-based routing** and connection tagging
- Connectivity checks
- **Load-balancing** policies
- **Stickiness** (email service provider will love you!)

Finally, this will just provide time!

There’s light!: IPv6

Most ISPs will provide each customer **a global IPv6 /56 network**. That gives you 8 bits or **256 /64 subnets** to cascade into your own network. E.g.: IoT, guests, DMZ…

- Change your mind! **Think in networks** and subnets!
- Network addresses are still **dynamic**! They may and will change! ☹
- **Prefix delegation** will enable us to cascade publicly available IPv6 addresses and network segments into our own network
  - Consider **zone-based firewalling**, please!
- Every host in your network **may receive a public IPv6 address**
  - **Global Unicast Address** – GUA (public)
  - It’s helpful to provide **Unique Local Addresses** – ULAs in addition


Internet service provider - ISP
Home
Sweet Home
198.51.111.84
IPv4
one way direction
10.64.0.1/16
10.64.0.123/24
192.168.5.110
NAT44
TPE
NAT44
RFC 6598 based network
100.64.0.0/10
+
IPv6 GUA
2001:db8:aaaa::/40
RFC 1918 based network
10.0.0.0/8
172.16.0.0/12
192.168.0.0/16
+
IPv6 GUA
2001:db8:aaaa:0105::/64
2001:db8:aa00::1/40
2001:db8:aaaa:0100::1/56
TPE
CPE
2001:db8:aaaa:0105::110/64
2001:db8::1/32
IPv6
two way direction

IPv6 – limited availability

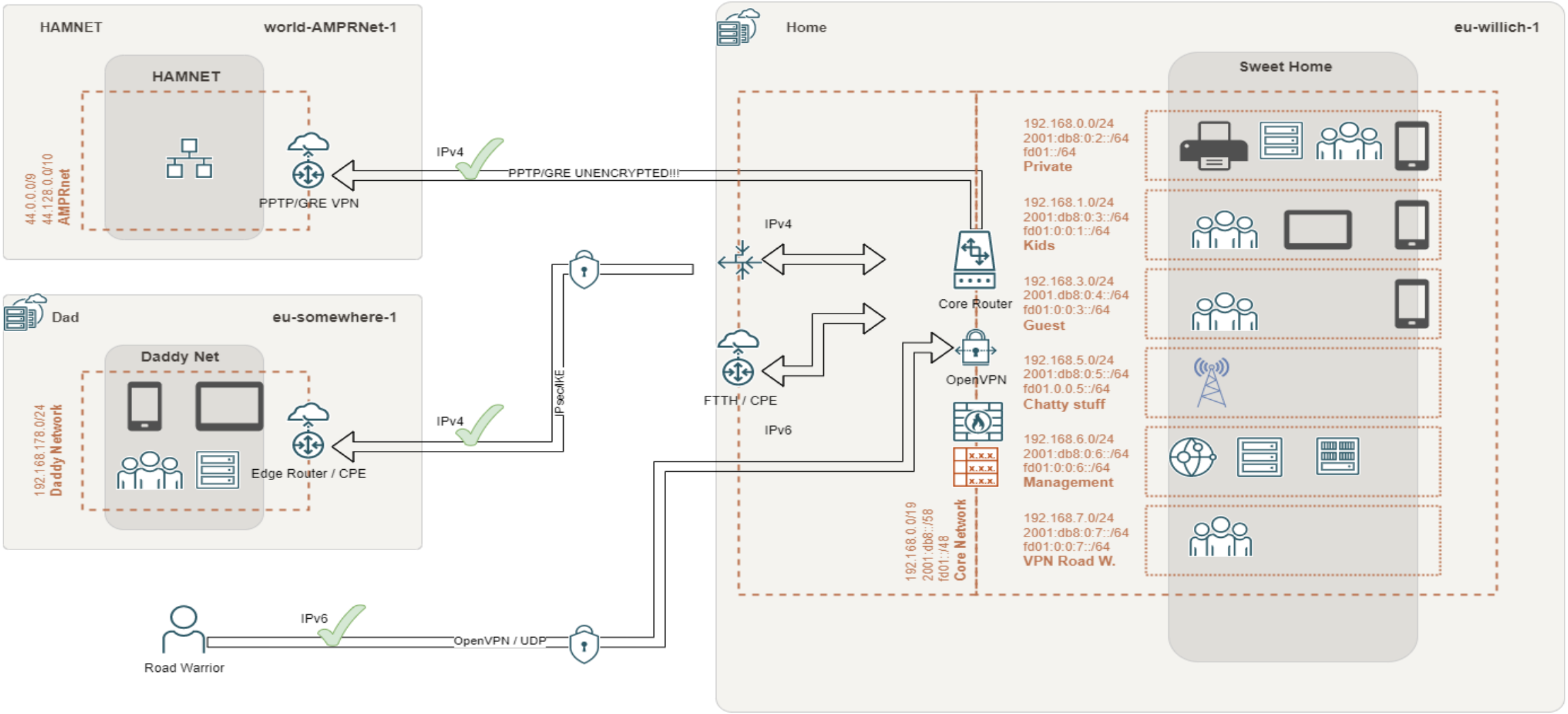
At the given time (around 2019), **IPv6 was not provided broadly**. Cell phone providers, ISPs and office spaces are limited to IPv4, mostly.

- Most **office networks** provide IPv4, only! Still today!
- Cell phone providers just started the implementation of IPv6 (dual stack)
- Some “ugly places” provide **RFC1918** addresses, only. (e.g., beach house)

Let‘s check what we can do so far!

Architecture Overview

My Sweet Home - access path



MARTIN SCHMITTER

6tunnel – calling home: there’s just IPv4

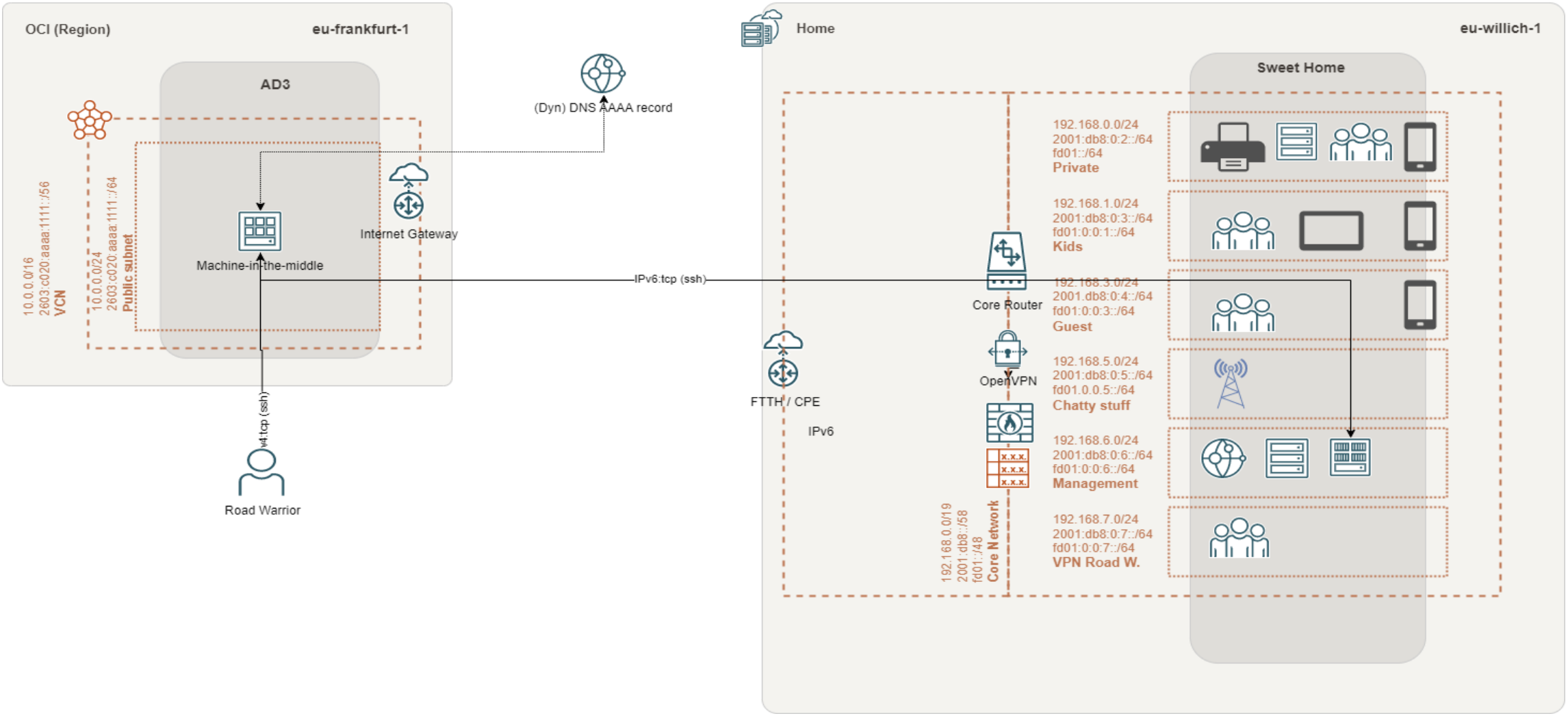
Just having an IPv4 stack:

- We need a **machine-in-the-middle** providing an IPv4 and IPv6 stack
- 6tunnel is a portmapper that will translate **IPv4:tcp** connections into **IPv6:tcp**
- Might be integrated and configured as **daemon**
- Simple syntax

```
$ 6tunnel 23232 myhost.dyn.foo.bar 22
```

Architecture Overview

My Sweet Home - 6tunnel



OpenVPN

In theory it’s possible to make use of **6tunnel** to establish OpenVPN connections, but:

- Tcp will dramatically reduce the **performance** of the VPN
  - Udp is prefered
- Couldn’t push **prefix delegated** IPv6 GUA addresses to VPN clients
  - Well, may be a layer 8 issue
- 6tunnel is capable of **tcp**, only!

What else?

Socket CAT aka. socat – multipurpose relay

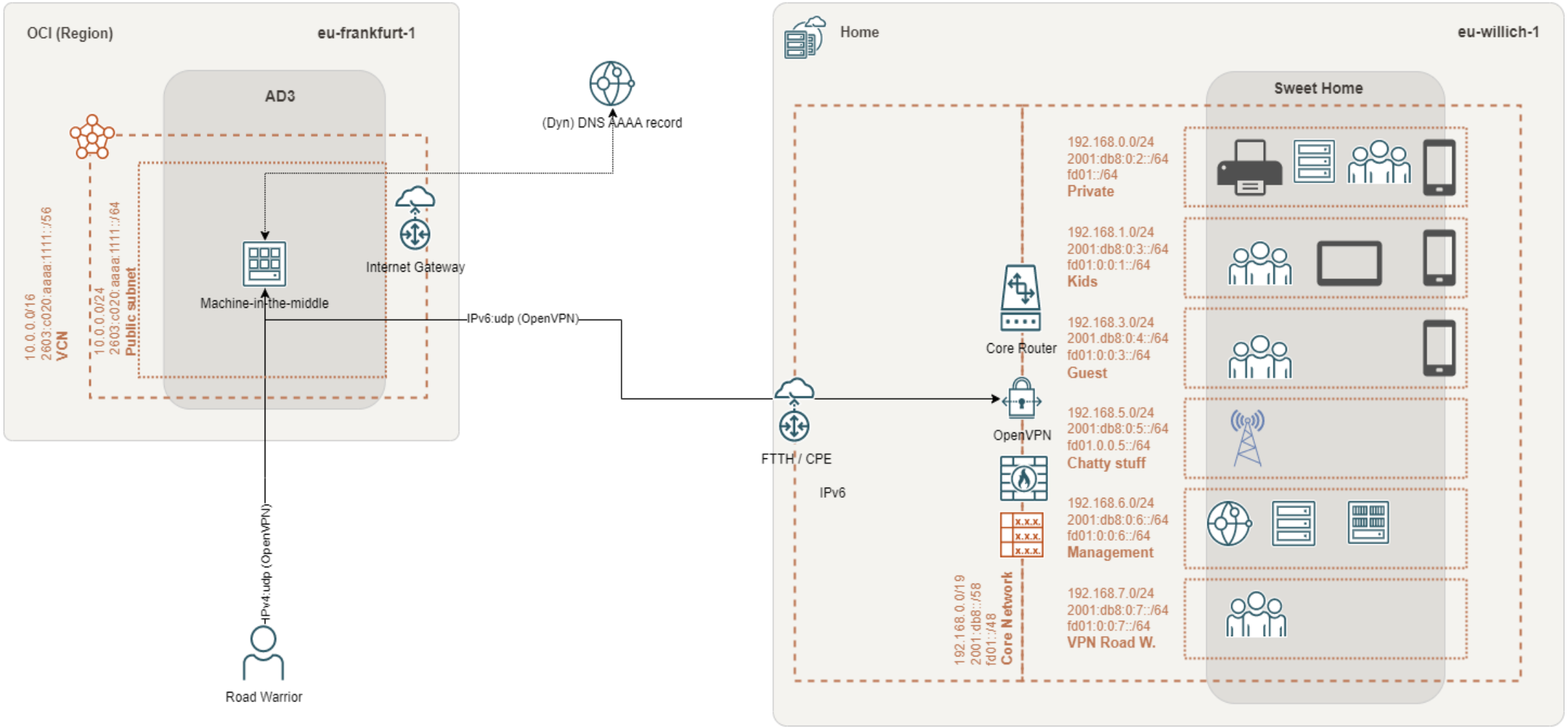
Socat is able to establish **bidirectional byte streams** and may help us to transform and establish udp connections:

- Udp -> much better performance using OpenVPN
- Much more configuration options
- Successful **delegation** of IPv6 addresses (PD) – OpenVPN with udp
- Might be integrated and configured as daemon

```
$ socat UDP4-LISTEN:1194,fork,reuseaddr UDP6:myhost.dyn.foo.bar:1194
```

Architecture Overview

My Sweet Home - OpenVPN

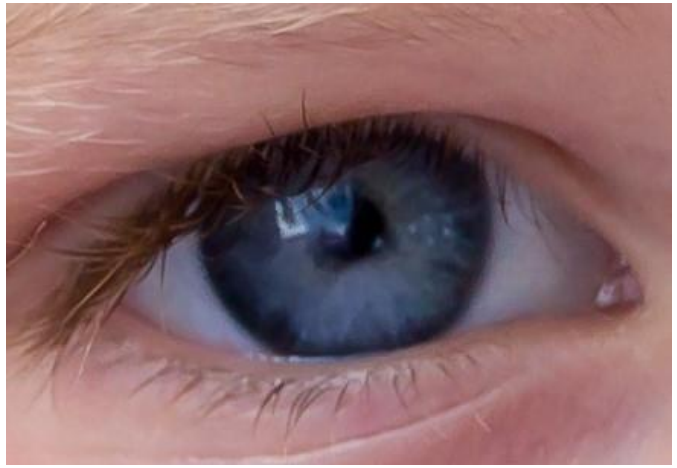


We made it!

I want more!

During the time, Oracle released **Arm-based Ampere A1** compute in the **Oracle Cloud Free Tier**. Time to setup my **Kubernetes** lab.

- But I don’t want to connect over **public Internet**!
- I need a **Site-to-Site VPN** to integrate the Oracle VPC into my network .
- Unfortunately, there isn’t an **out of the box** IPv6 based VPN service available.



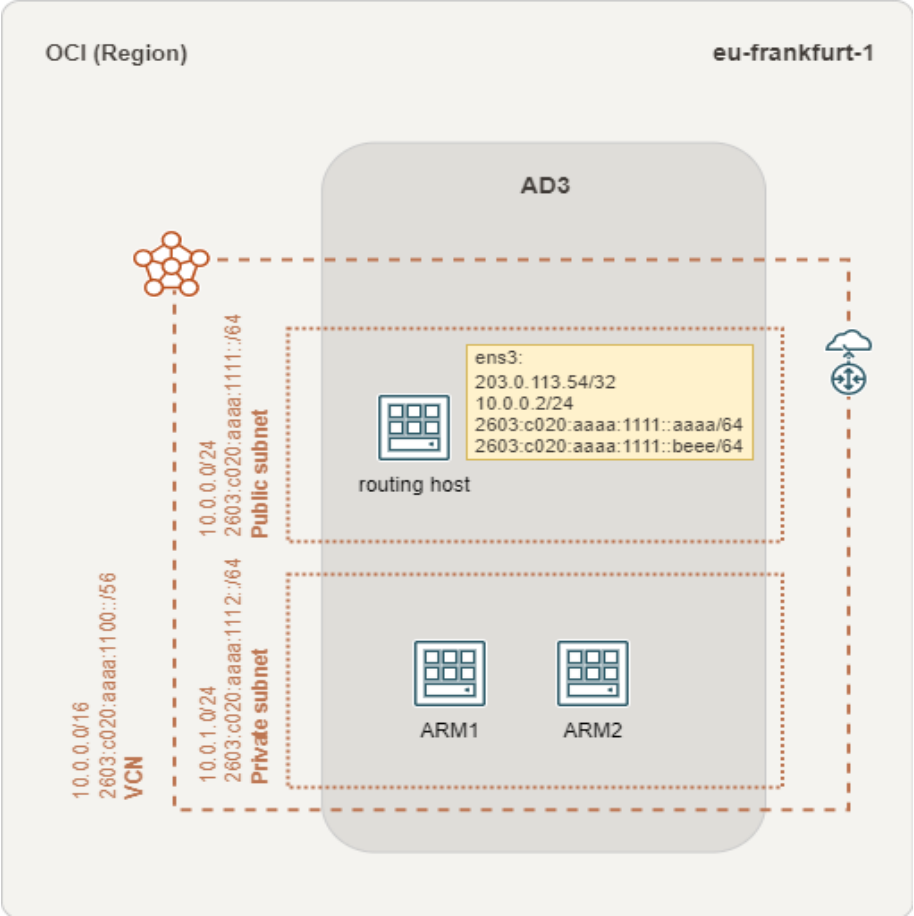
Let`s be creative!

“Which VPN technology?”

My preference is **WireGuard**, a quite young open-source implementation:

- Supports IPv6:udp (requires udp)
- Well integrated into most Linux based kernel
- Better **hardware efficiency** –> less hardware consumption
  - Cheaper devices -> good **Wife Acceptance Factor - WAF**!
  - I’m still married! ☺
- Certificate and **encryption handling** is very easy
  - Especially compared with OpenVPN

Setup your OCI



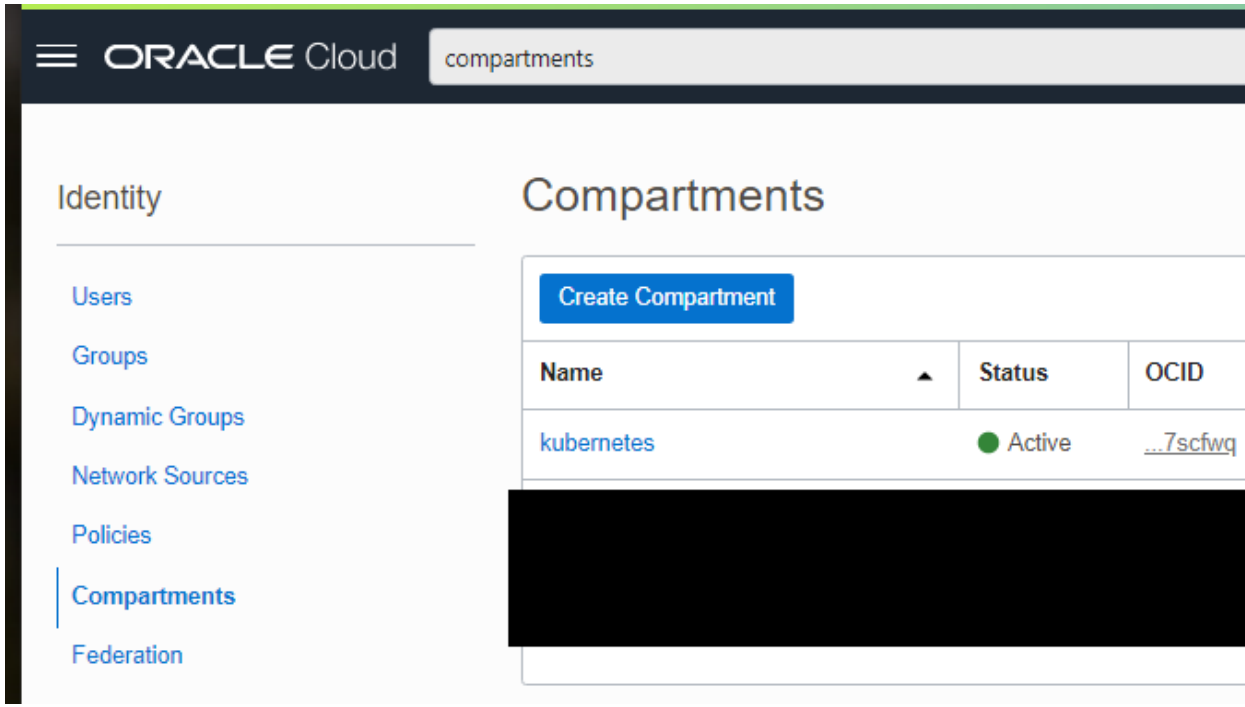
What we need:

- VPC with private and public subnet
- IPv4 addresses (public and private)
- Public IPv6 addresses
- Compute power (routing host)

Setup your OCI: Compartments

Within the OCI, some preparations are required:

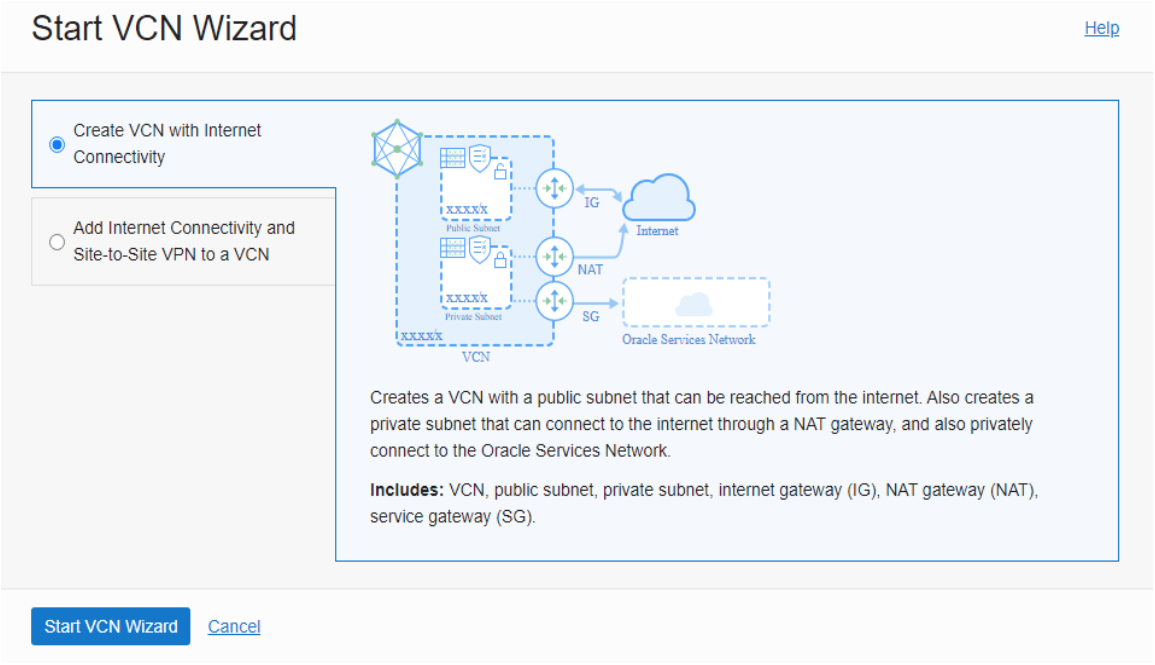
- It's good practice to work with **Compartments**.
  - **Logical segregation** will help us to be more structured
  - In this example all activities are made in the context of a Compartment



Setup your OCI: Virtual Cloud Networks - VCN

To define your Cloud network, make use of **the VCN Wizard**:

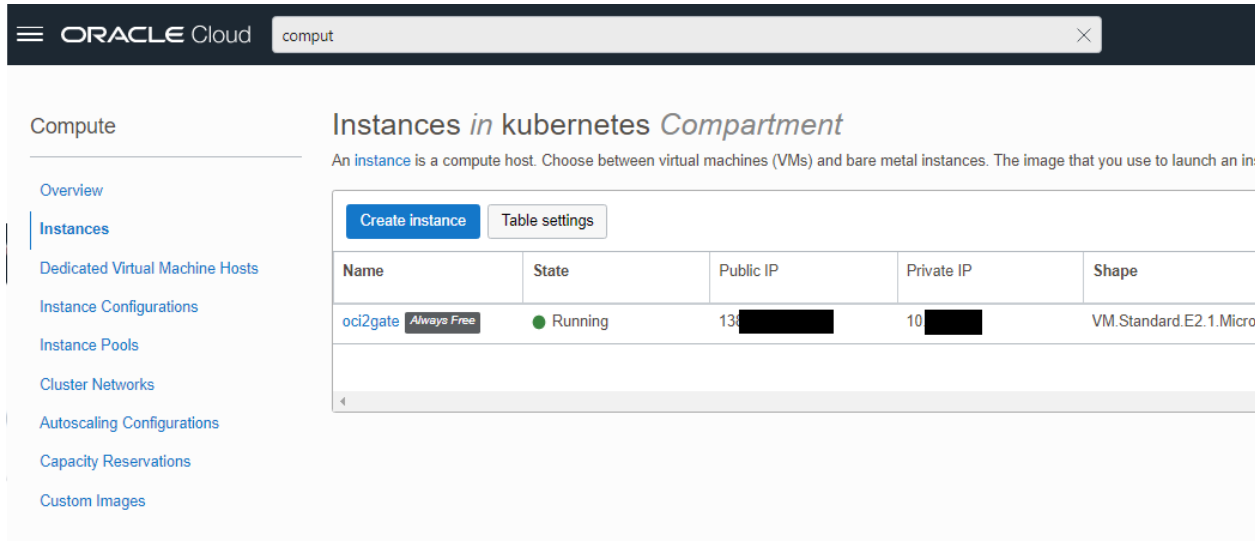
- Select **“VCN with Internet Connectivity”**
- Enable an **IPv6 CIDR Block** attached to the VCN and all subnets
- Prevent **network address conflicts** with your private IPv4 ranges!

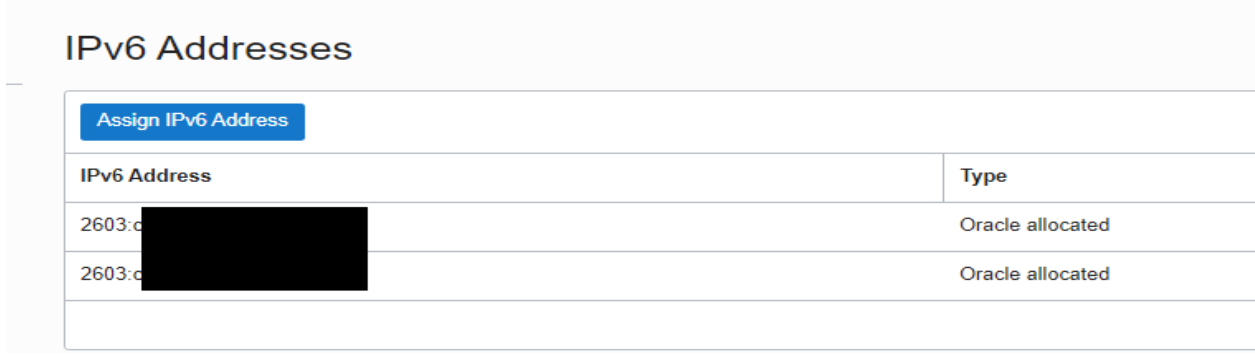


Setup your OCI: Router Compute Instance

To establish **a Site-to-Site VPN**, setup a **compute instance** to provide **VPN endpoint/router** functionality:

- Best choice: AMD based free instance with **Ubuntu minimal**
- To be placed it in the **public subnet**
- Setup VNIC
  - Assign **two IPv6 addresses**, one public IPv4
  - Enable: **“Skip source/destination check”** – We are a router!





Setup Compute Instance: WireGuard

The WireGuard configuration is well documented: Configuring a VPN by Using WireGuard (oracle.com) **F45242-06**. Some of the parameters need to be different:

- WireGuard provides **`wg-quick`** to setup a VPN connection
  - **`Wg-quick`** will update **local routing information**

    ->**`Wg-quick`** MUST NOT be used!
  - Therefore, **`systemd-networkd`** is the best choice to handle the VPN connection/configuration
    - Define **`netdev`** and **`network`** files in **`/etc/systemd/network/`**

Setup Compute Instance: WireGuard

Pro tip:

- Make sure your **endpoint address** has got an **AAAA** record, only!

```
# /etc/systemd/network/50-wireguard.netdev
Endpoint = foobar.dyn.foo.bar
<dynamic DNS of MYSWEETHOME (ipv6 only AAAA record)>
```

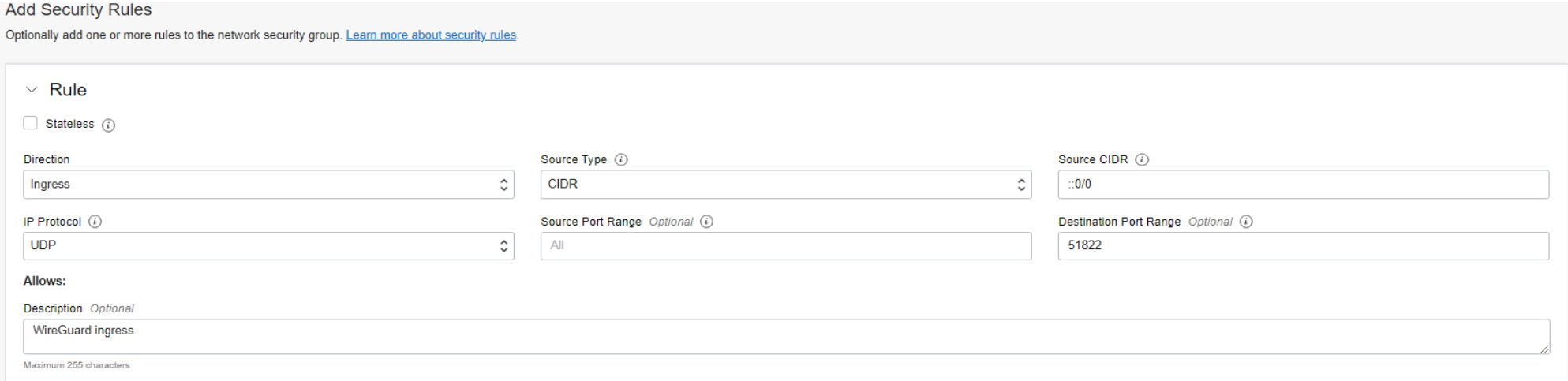
- Don’t forget to include a **`link-local`** address

```
# /etc/systemd/network/50-wireguard.network
[Network]
Address = fd0:0:0:1::2/64 <IPv6 ULA transport network address>
Address = 192.168.250.6/30 <IPv4 transport network address>
Address = fe80::6/64 <link-local>
```

Setup Compute Instance: NSG

Establishing a WireGuard connection, requires an open **`udp`** port. This can be configured with the help of a **Network Security Group - NSG**:

- Create a NSG with your desired udp:port (e.g. **`51822`**)
- Attach the NSG to your **compute instance**, to make the configuration active



Setup Core Router: Security List

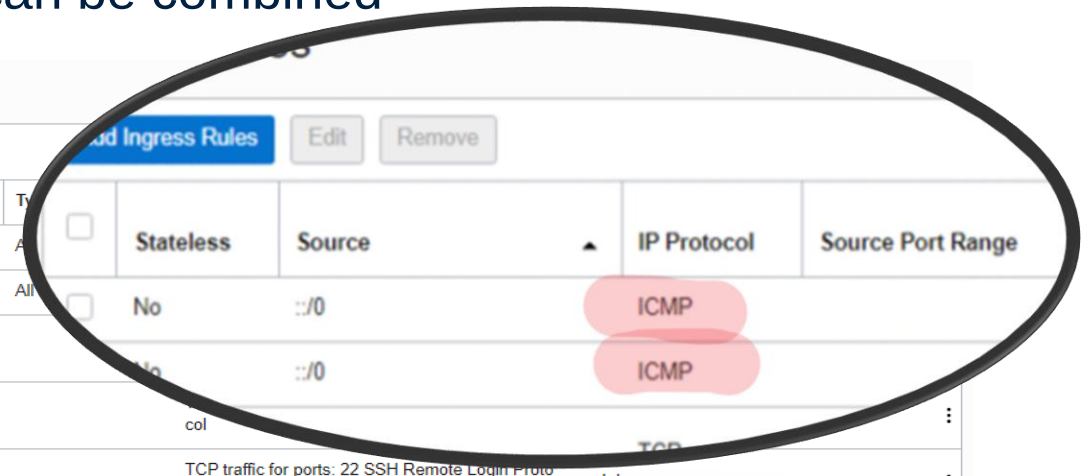
Opposite to a Network Security Group, that's assigned to a host, a **Security List** will take care of a complete network segment:

- Allowing **ICMP ping** between networks, might help to troubleshoot configuration issues
- **Security Lists** and **Network Security Groups** can be combined

Ingress Rules

Add Ingress Rules | Edit | Remove

Stateless	Source	IP Protocol	Source Port Range	Destination Port Range	Ty		
No	::/0	IPv6-ICMP			A		
No	::/0	IPv6-ICMP			All		
No	fd02:[redacted]	TCP	All	22			
No	fd93:[redacted]	TCP	All	22		col	
No	fdd2:[redacted]	TCP	All	22		TCP traffic for ports: 22 SSH Remote Login Proto col	ssh ingress
No	0.0.0.0/0	ICMP			3, 4	ICMP traffic for: 3, 4 Destination Unreachable: Fra gmentation Needed and Don't Fragment was Set	
No	0.0.0.0/0	ICMP			8	ICMP traffic for: 8 Echo	ping
No	10.[redacted]	TCP	All	22		TCP traffic for ports: 22 SSH Remote Login Proto col	ssh ingress



Setup Compute Instance: Default Route Table

To configure the **routing host** as a **gateway** to your **home network**, you need to make other hosts in the **Oracle Cloud** aware about this. This can be archived with the help of the **Default Route Table**:

- `::beee` is the **WG gateway** to home
- ULA addresses are used because of **dynamic GUA addresses** at home -> **predictable**

Route Rules

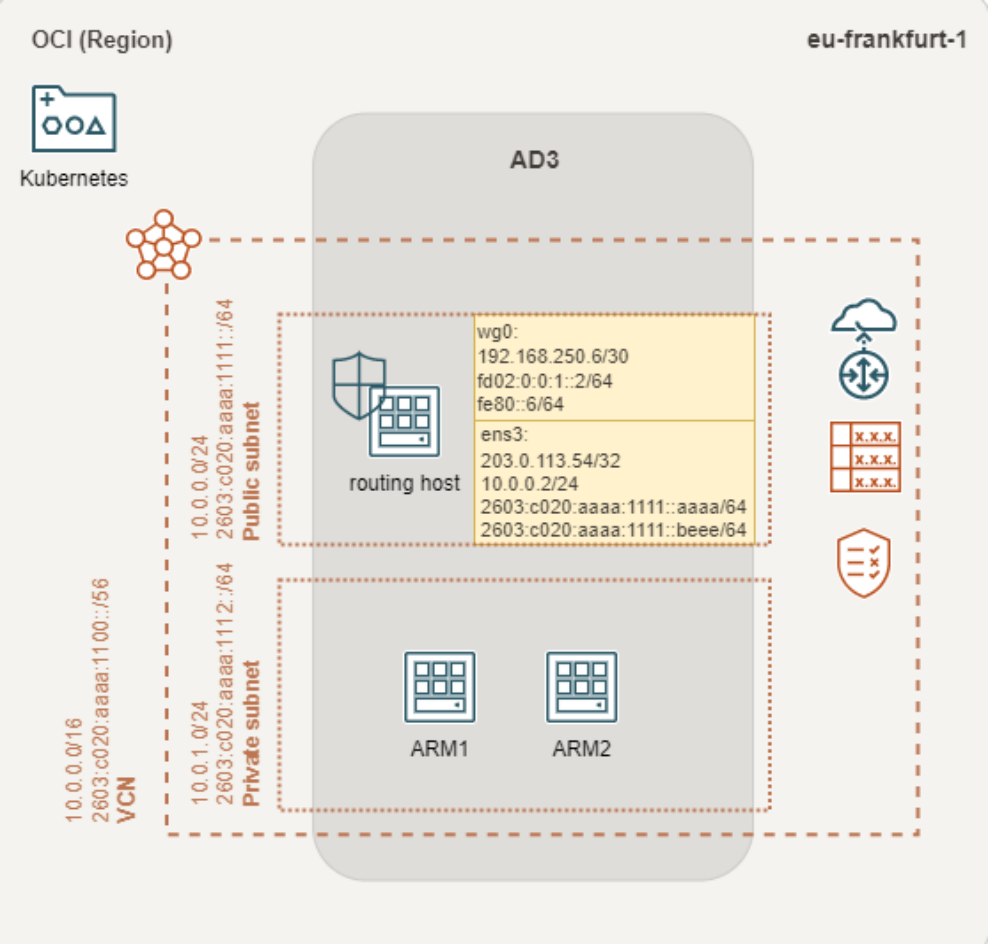
Traffic within the VCN is handled by the VCN's local routing by default. Intra-VCN routing allows you more control over routing between s

Add Route Rules | Edit | Remove

Destination	Target Type	Target
::/0	Internet Gateway	Internet Gateway-vcn_kubernetes
fd02:[redacted]0::/56	IPv6 Address	2603:[redacted]beee
fd93:[redacted]:/48	IPv6 Address	2603:[redacted]:beee
fdd2:[redacted]:/48	IPv6 Address	2603:[redacted]:beee
0.0.0.0/0	Internet Gateway	Internet Gateway-vcn_kubernetes
192.168.0.0/16	Private IP	10.[redacted].2

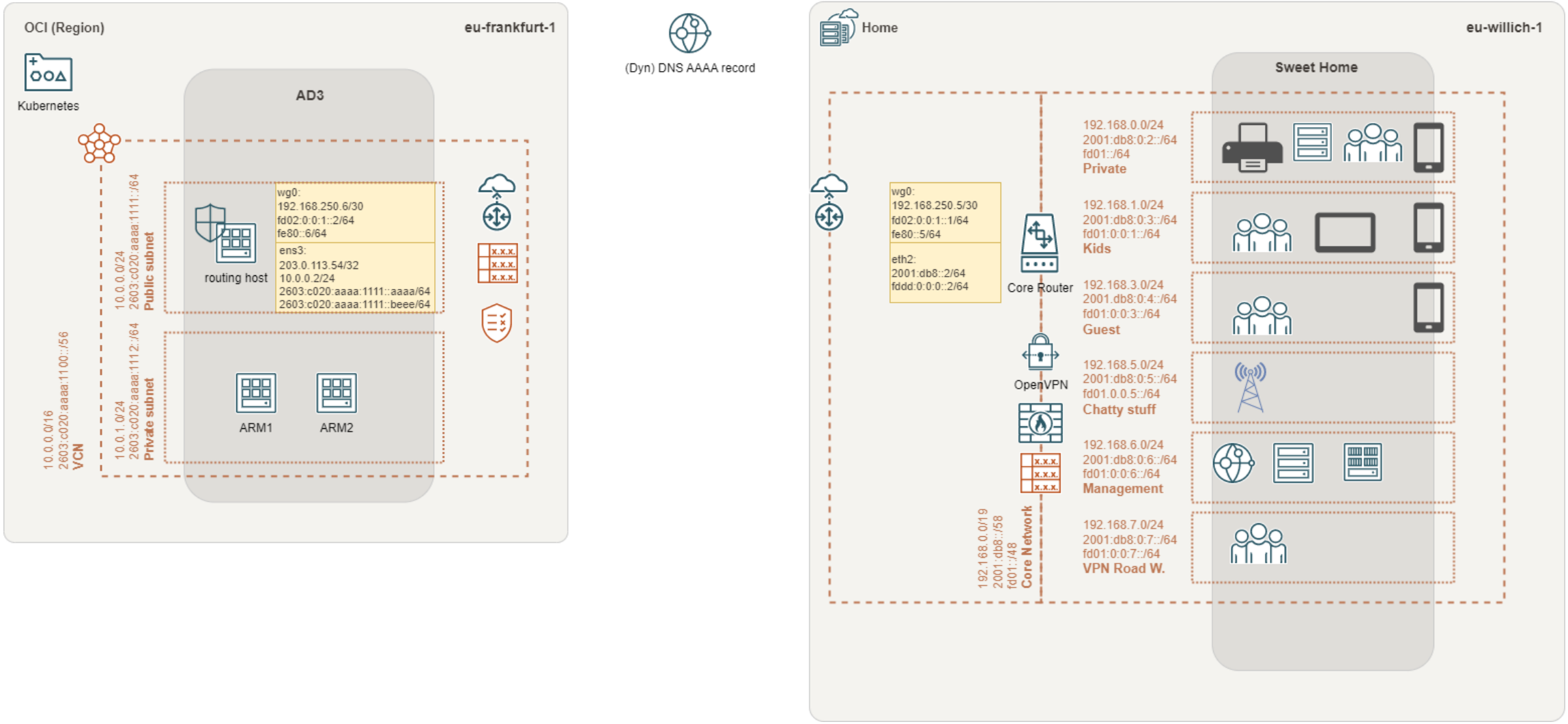
0 Selected

Setup Compute Instance: Review



Finally, there's a configured VPC and a Compute Instance including:

- **WireGuard** interface wg0
- **Default Routing Table**
  - Pointing to VPN gateway
- **NSG** attached to Compute Instance
- **Security List** attached to VCN and subnets
- Optional: Some **ARM Compute Instances**


OCI (Region)
eu-frankfurt-1
Kubernetes
AD3
10.0.0.0/16
2603:c020:aaaa:1100::/56
VCN
10.0.0.0/24
2603:c020:aaaa:1111::/64
Public subnet
routing host
wg0:
192.168.250.6/30
fd02:0:0:1::2/64
fe80::6/64
ens3:
203.0.113.54/32
10.0.0.2/24
2603:c020:aaaa:1111::aaaa/64
2603:c020:aaaa:1111::beee/64
10.0.1.0/24
2603:c020:aaaa:1112::/64
Private subnet
ARM1
ARM2
(Dyn) DNS AAAA record
Home
eu-willich-1
Sweet Home
wg0:
192.168.250.5/30
fd02:0:0:1::1/64
fe80::5/64
eth2:
2001:db8::2/64
fddd:0:0:0::2/64
Core Router
OpenVPN
192.168.0.0/19
2001:db8::/58
fd01::/48
Core Network
192.168.0.0/24
2001:db8:0:2::/64
fd01::/64
Private
192.168.1.0/24
2001:db8:0:3::/64
fd01:0:0:1::/64
Kids
192.168.3.0/24
2001.db8:0:4::/64
fd01:0:0:3::/64
Guest
192.168.5.0/24
2001:db8:0:5::/64
fd01.0.0.5::/64
Chatty stuff
192.168.6.0/24
2001:db8:0:6::/64
fd01:0:0:6::/64
Management
192.168.7.0/24
2001:db8:0:7::/64
fd01:0:0:7::/64
VPN Road W.

Setup Core Router: WireGuard

Now, it’s time to install WireGuard into the **core router**. In my situation it's an **EdgeRouter™-X** with firmware version 2.0.9.

- Download the prebuilded installation file
  - https://www.wireguard.com/install/#edgeos-module-tools
- Follow the installation instructions
  - https://github.com/WireGuard/wireguard-vyatta-ubnt/wiki/EdgeOS-and-Unifi-Gateway

Setup Core Router: WireGuard

Making use of IPv6 and **OSPF**, the standard configuration steps need to be amended:

- Include a **IPv6 link-local** address
- Set **`allowed-ips`** to 0.0.0.0/0 and ::/0
- Set **`route-allowed-ips = false`**
- Take care/reduce your **`mtu`** to e.g., **`1420`**
  - IPv6 doesn't like **fragmentation**!

- Full installation details:
  - https://blog.dieschmitterlinge.de

Setup Core Router: WireGuard

```
set interfaces wireguard wg0 address 192.168.250.5/30          <transport IPv4>
set interfaces wireguard wg0 address 'fd02:0:0:1::1/64'  <transport IPv6>
set interfaces wireguard wg0 address 'fe80::5/64'<link-local>

set interfaces wireguard wg0 description 'Wireguard S2S VPN to OCI'
set interfaces wireguard wg0 listen-port 51822
set interfaces wireguard wg0 mtu 1420

set interfaces wireguard wg0 peer <public key - oci> allowed-ips 0.0.0.0/0
set interfaces wireguard wg0 peer <public key - oci> allowed-ips '::/0'
set interfaces wireguard wg0 peer <public key - oci> endpoint
'2603:c020:aaaa:1111::aaaa:51822'

set interfaces wireguard wg0 private-key /config/auth/wg.key
set interfaces wireguard wg0 route-allowed-ips false
```

Setup Core Router: Firewall Rules

IPv6 concepts are slightly different - you need to **think in segments** not in addresses!

To handle the situation that public IPv6 (segments) **addresses may change randomly** (based on prefix delegation), I highly recommend to make use of **zone-based** firewalling.

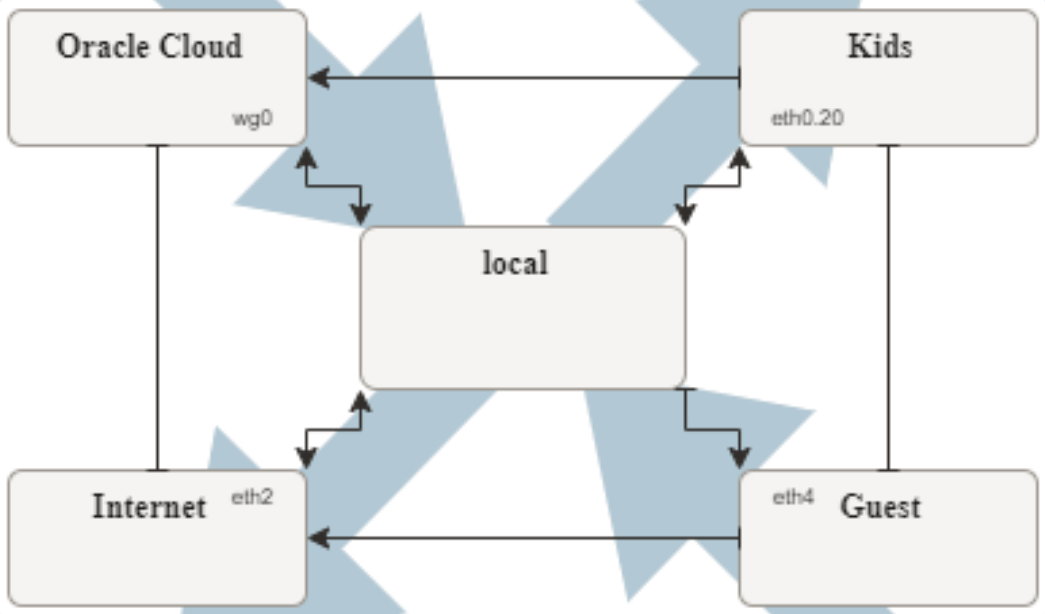
- Define a rule for WireGuard ingress and attach the rule to your **`LOCAL`** zone-based policy

```
set firewall ipv6-name wan-local-6 rule 110 action accept
set firewall ipv6-name wan-local-6 rule 110 description 'allow Wireguard VPN'
set firewall ipv6-name wan-local-6 rule 110 destination port 51822
set firewall ipv6-name wan-local-6 rule 110 log enable
set firewall ipv6-name wan-local-6 rule 110 protocol udp
```

Setup Core Router: Firewall Rules

Zone-based firewalling is a huge benefit. **Don't mind** about addresses! Mind about destinations!

- Define rules
- Declare **zones** and **zone policies** with the help of rules
- Attach **interfaces** to zones



Setup Core Router: Dynamic GUA addresses

How to deal with **dynamic GUA addresses**?

- IPv6 **destination address rules** might be masked to **interface addresses**.

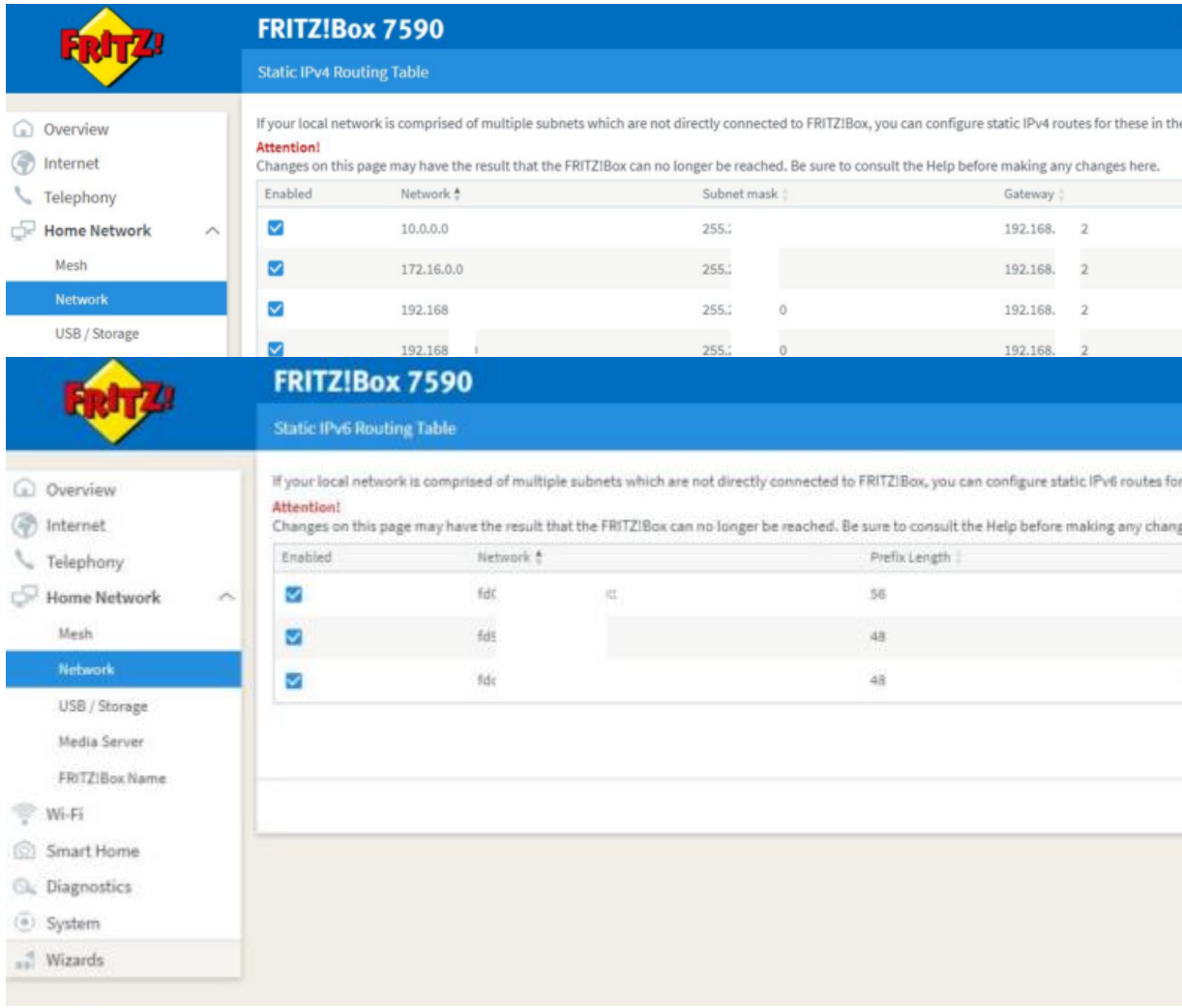
```
set firewall ipv6-name WAN-to-LAN-6 rule 123 destination address
'::ba27:ebff:fede:e2c7/::ffff:ffff:ffff:ffff'
```

- Combine the **ingress rule** with a **zone policy**
  - Yes, it feels ugly, in the first moment
  - Just used **internally**

Setup Edge Router: Routing

Terminating the VPN at the **core router,** requires a configuration of firewall- and routing rules at the **edge router**, as well. Required rules and routing need to point to the **internal core router**.

- Enable WireGuard:udp ingress to the core router
- Define routing rules for the internal- and cloud networks
  - Core router acts as gateway
  - Don’t forget to make use of ULA addresses!



Send IPv6 traffic through the VPN

In the **Oracle Cloud Free Tier** I couldn’t make use of **IPv6 ULA addresses**. What will raise the question, how will it be possible to send IPv6 traffic trough the VPN connection?

- Define a **static route** to the OCI router

```
set protocols static route6 '2603:c020:aaaa:1111::aaaa/128' next-hop 'fe80::6' distance 1
set protocols static route6 '2603:c020:aaaa:1111::aaaa/128' next-hop 'fe80::6' interface eth2
```

- Define a static route to the **Oracle Cloud network** segment through to the VPN

```
set protocols static route6 '2603:c020:aaaa:1100::/56' next-hop ‘fd02:0:0:1::2' distance 10
set protocols static route6 '2603:c020:aaaa:1100::/56' next-hop ‘fd02:0:0:1::2' interface wg2
```

Send IPv6 traffic through the VPN: validate

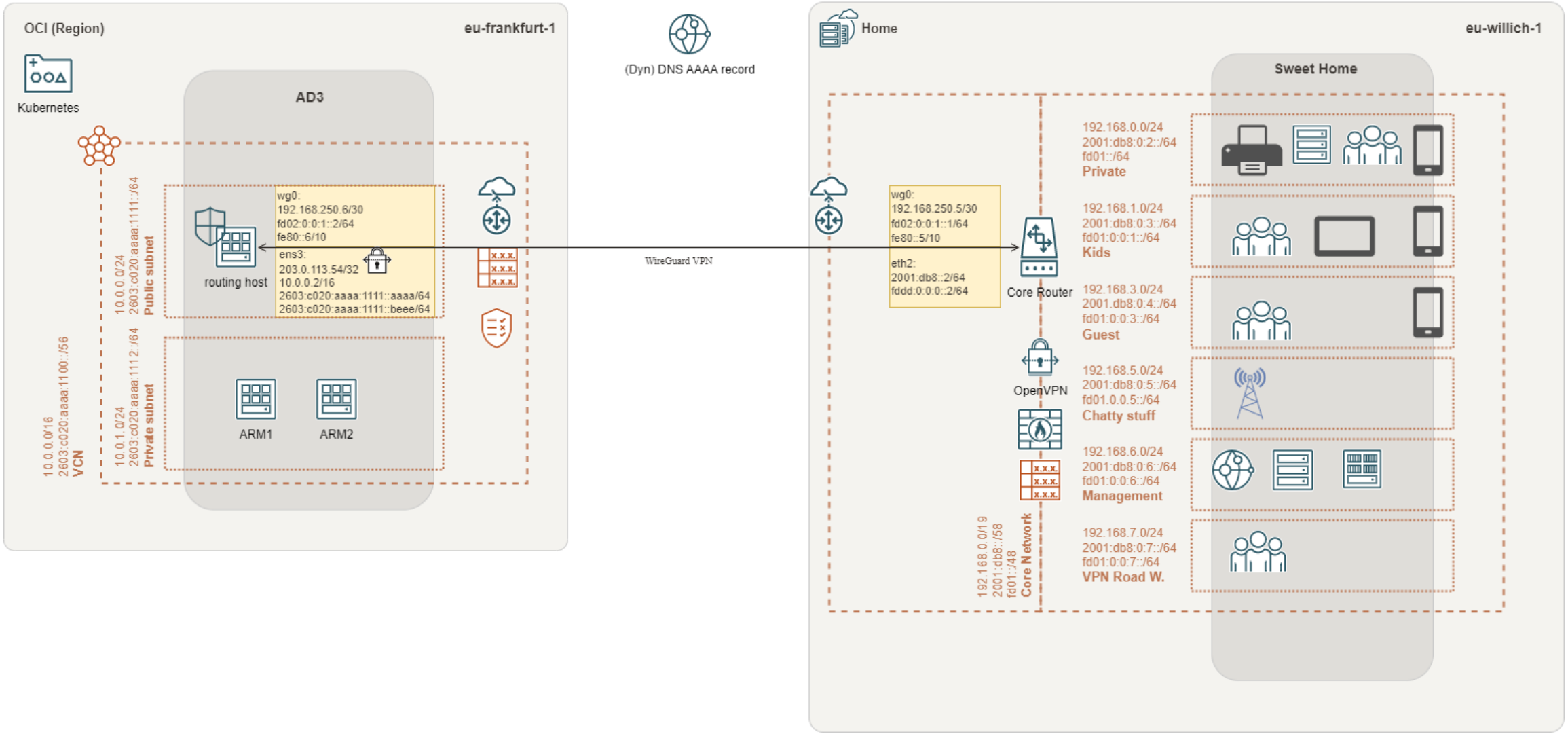
As more specific, has got priority:

- Validating static route to the router in OCI

```
#ip route get 2603:c020:aaaa:1111::aaaa
2603:c020:aaaa:1111::aaaa from :: via fe80::6 dev eth2 proto zebra src 2001:db8::2
metric 1024 pref medium
```

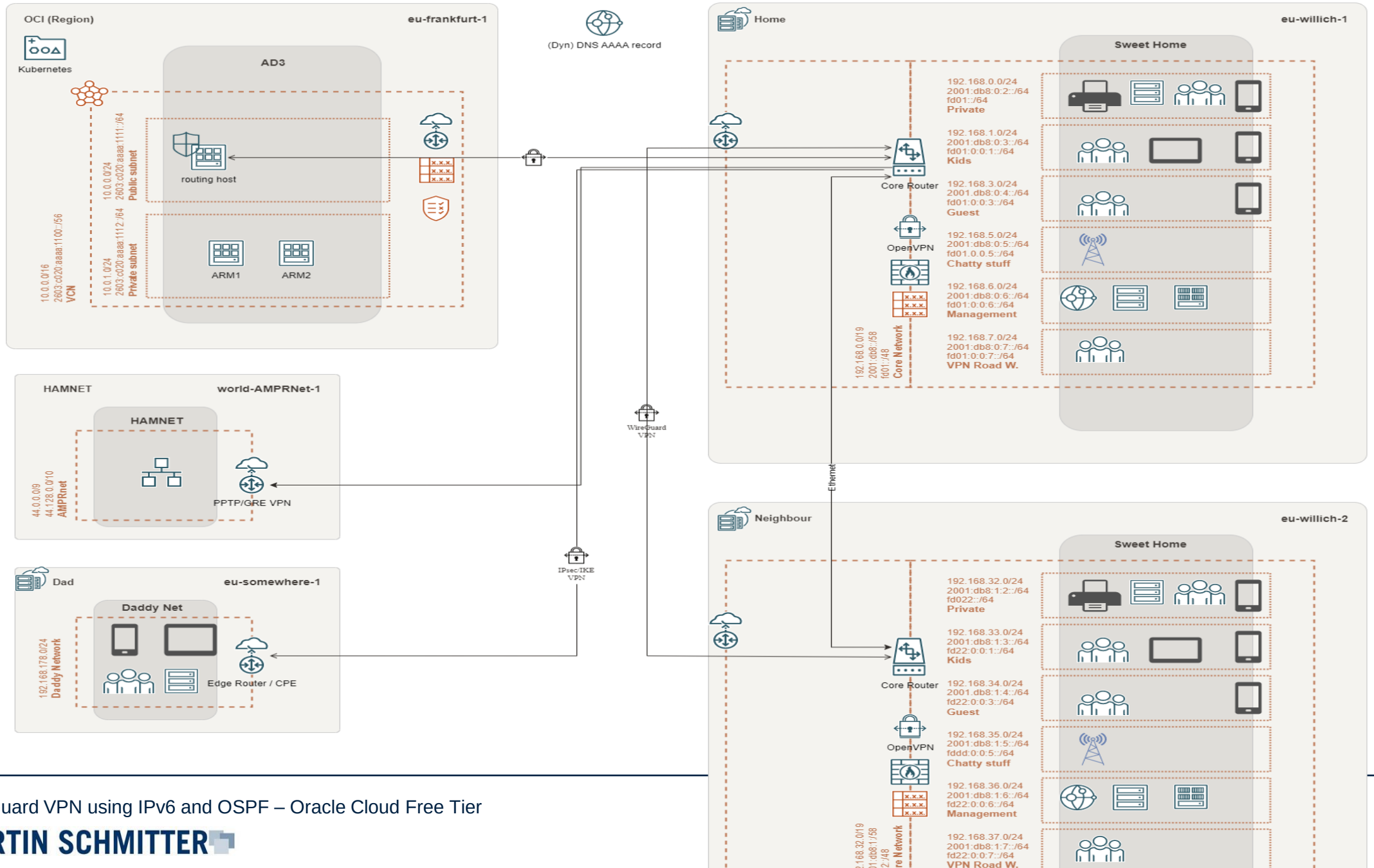
- Define a static route to the Oracle Cloud network segment through to the VPN

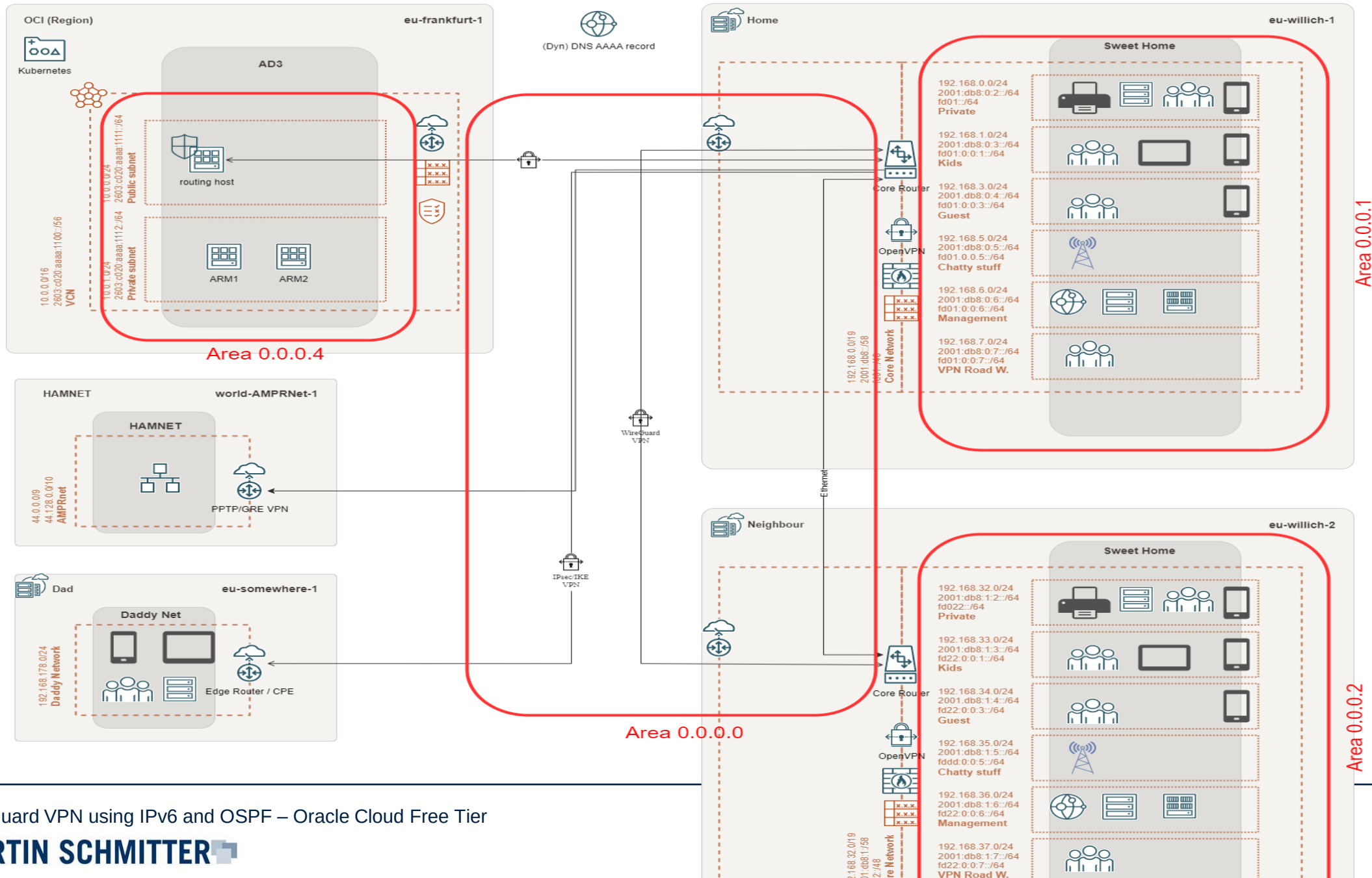
```
#ip route get 2603:c020:aaaa:1111::beee
2603:c020:aaaa:1111::beee from :: via fd02:0:0:1::2 dev wg2 proto zebra src
fd02:0:0:1::1 metric 1024 pref medium
```


OCI (Region)
eu-frankfurt-1
Kubernetes
AD3
10.0.0.0/16
2603:c020:aaaa:1100::/56
VCN
10.0.0.0/24
2603:c020:aaaa:1111::/64
Public subnet
routing host
wg0:
192.168.250.6/30
fd02:0:0:1::2/64
fe80::6/10
ens3:
203.0.113.54/32
10.0.0.2/16
2603:c020:aaaa:1111::aaaa/64
2603:c020:aaaa:1111::beee/64
10.0.1.0/24
2603:c020:aaaa:1112::/64
Private subnet
ARM1
ARM2
(Dyn) DNS AAAA record
WireGuard VPN
Home
eu-willich-1
Sweet Home
wg0:
192.168.250.5/30
fd02:0:0:1::1/64
fe80::5/10
eth2:
2001:db8::2/64
fddd:0:0:0::2/64
Core Router
OpenVPN
192.168.0.0/19
2001:db8::/58
fd01::/48
Core Network
192.168.0.0/24
2001:db8:0:2::/64
fd01::/64
Private
192.168.1.0/24
2001:db8:0:3::/64
fd01:0:0:1::/64
Kids
192.168.3.0/24
2001.db8:0:4::/64
fd01:0:0:3::/64
Guest
192.168.5.0/24
2001:db8:0:5::/64
fd01.0.0.5::/64
Chatty stuff
192.168.6.0/24
2001:db8:0:6::/64
fd01:0:0:6::/64
Management
192.168.7.0/24
2001:db8:0:7::/64
fd01:0:0:7::/64
VPN Road W.

VPN connected!

What’s about routing?


OCI (Region)
eu-frankfurt-1
Kubernetes
AD3
routing host
10.0.0.0/24
2603:c020:aaaa:1111::/64
Public subnet
10.0.1.0/24
2603:c020:aaaa:1112::/64
Private subnet
10.0.0.0/16
2603:c020:aaaa:1100::/56
VCN
ARM1
ARM2
(Dyn) DNS AAAA record
Home
eu-willich-1
Sweet Home
192.168.0.0/24
2001:db8:0:2::/64
fd01::/64
Private
192.168.1.0/24
2001:db8:0:3::/64
fd01:0:0:1::/64
Kids
Core Router
192.168.3.0/24
2001:db8:0:4::/64
fd01:0:0:3::/64
Guest
OpenVPN
192.168.5.0/24
2001:db8:0:5::/64
fd01:0:0:5::/64
Chatty stuff
192.168.6.0/24
2001:db8:0:6::/64
fd01:0:0:6::/64
Management
192.168.7.0/24
2001:db8:0:7::/64
fd01:0:0:7::/64
VPN Road W.
192.168.0.0/19
2001:db8::/58
fd01::/48
Core Network
HAMNET
world-AMPRNet-1
HAMNET
44.0.0.0/9
44.128.0.0/10
AMPRnet
PPTP/GRE VPN
WireGuard VPN
IPsec/IKE VPN
Ethernet
Dad
eu-somewhere-1
Daddy Net
192.168.178.0/24
Daddy Network
Edge Router / CPE
Neighbour
eu-willich-2
Sweet Home
192.168.32.0/24
2001:db8:1:2::/64
fd022::/64
Private
192.168.33.0/24
2001:db8:1:3::/64
fd22:0:0:1::/64
Kids
Core Router
192.168.34.0/24
2001.db8:1:4::/64
fd22:0:0:3::/64
Guest
OpenVPN
192.168.35.0/24
2001:db8:1:5::/64
fddd:0:0:5::/64
Chatty stuff
192.168.36.0/24
2001:db8:1:6::/64
fd22:0:0:6::/64
Management
192.168.37.0/24
2001:db8:1:7::/64
fd22:0:0:7::/64
VPN Road W.
192.168.32.0/19
2001:db8:1/58
fd22::/48
Core Network


OCI (Region)
eu-frankfurt-1
Kubernetes
AD3
routing host
ARM1
ARM2
10.0.0.0/16
2603:c020:aaaa:1100::/56
VCN
10.0.0.0/24
2603:c020:aaaa:1111::/64
Public subnet
10.0.1.0/24
2603:c020:aaaa:1112::/64
Private subnet
Area 0.0.0.4
(Dyn) DNS AAAA record
Home
eu-willich-1
Sweet Home
192.168.0.0/24
2001:db8:0:2::/64
fd01::/64
Private
192.168.1.0/24
2001:db8:0:3::/64
fd01:0:0:1::/64
Kids
192.168.3.0/24
2001:db8:0:4::/64
fd01:0:0:3::/64
Guest
192.168.5.0/24
2001:db8:0:5::/64
fd01.0.0.5::/64
Chatty stuff
192.168.6.0/24
2001:db8:0:6::/64
fd01:0:0:6::/64
Management
192.168.7.0/24
2001:db8:0:7::/64
fd01:0:0:7::/64
VPN Road W.
Core Router
OpenVPN
192.168.0.0/19
2001:db8::/58
Core Network
Area 0.0.0.1
HAMNET
world-AMPRNet-1
HAMNET
44.0.0.0/9
44.128.0.0/10
AMPRnet
PPTP/GRE VPN
WireGuard VPN
Ethernet
IPsec/IKE VPN
Dad
eu-somewhere-1
Daddy Net
192.168.178.0/24
Daddy Network
Edge Router / CPE
Area 0.0.0.0
Neighbour
eu-willich-2
Sweet Home
192.168.32.0/24
2001:db8:1:2::/64
fd022::/64
Private
192.168.33.0/24
2001:db8:1:3::/64
fd22:0:0:1::/64
Kids
192.168.34.0/24
2001.db8:1:4::/64
fd22:0:0:3::/64
Guest
192.168.35.0/24
2001:db8:1:5::/64
fddd:0:0:5::/64
Chatty stuff
192.168.36.0/24
2001:db8:1:6::/64
fd22:0:0:6::/64
Management
192.168.37.0/24
2001:db8:1:7::/64
fd22:0:0:7::/64
VPN Road W.
Core Router
OpenVPN
192.168.32.0/19
2001:db8:1::/58
2:/48
Area 0.0.0.2

Quagga Routing Suite & OSPF

OSPF is an **interior gateway protocol** and **will help** use to mange the routing over the networks. With multiple router involved, management becomes **more complex** and will implement a risk of failure. A gateway protocol will help us to mitigate the risk of **configuration failure** and reduce the configuration effort.

- Quagga Routing Suite as software router will help us.
  - **Reduce complexity**
  - **Central point** for local routing configuration
  - Is in use at the core router, already
- Full installation details:
  - https://blog.dieschmitterlinge.de/

Quagga Routing Suite: Installation

Check the OCI „Know Issues“ document!

Known issues:

- Don’t forget the **kernel settings** (forwarding)

```
net.ipv4.ip_forward = 1
net.ipv6.conf.all.forwarding = 1
```

- Preconfigured **host firewall rules** on Oracle Ubuntu image
  - Don’t make use of Uncomplicated Firewall (UFW)
  - To modify rules, make use of `/etc/iptables/rules.v[4|6]`

```
$ sudo su -
# iptables-restore < /etc/iptables/rules.v4
```

Quagga Routing Suite: iptables

- Example for IPv4

```
#allow OSPF, just an example
root@oci:~# iptables -A INPUT --protocol OSPF -j ACCEPT
root@oci:~# iptables -L -v -n
chain INPUT (policy ACCEPT 342 packets, 24562 bytes)
 pkts bytes target     prot opt in     out     source
destination
    1    84 ACCEPT     89   --  *      *       0.0.0.0/0            0.0.0.0/0
…
root@oci:~# iptables -A OUTPUT --protocol OSPF -j ACCEPT
root@oci:~# iptables -A FORWARD --protocol OSPF -j ACCEPT
```

Quagga Routing Suite: ip6tables

- Example for IPv6

```
#allow OSPFv3, just an example
root@oci:~# ip6tables -A INPUT --protocol OSPF -j ACCEPT
root@oci:~# ip6tables -L -v -n
Chain INPUT (policy ACCEPT 1120 packets, 184K bytes)
 pkts bytes target     prot opt in     out     source
destination
   25  1900 ACCEPT     89       *      *       ::/0                ::/0
…
root@oci:~# iptables -A OUTPUT --protocol OSPF -j ACCEPT
root@oci:~# iptables -A FORWARD --protocol OSPF -j ACCEPT
```

Quagga Routing Suite: interface configuration

```
#!/etc/quagga/zebra.conf
…
!
interface ens3
 link-detect
 ip address 10.0.0.2/24
 ipv6 address 2603:c020:aaaa:1111::aaaa/64
 ipv6 address 2603:c020:aaaa:1111::beee/64
!
interface wg0
 link-detect
 ip address 192.168.250.6/30
 ipv6 address fd02:0:0:1::2/64
 ipv6 address fe80::6/64
!
…
```

Quagga Routing Suite: OSPF configuration

```
!/etc/quagga /ospfd.conf
…
!
router ospf
 ospf router-id 192.168.250.6
 redistribute connected
 network 10.0.0.0/16 area 0.0.0.4          <OCI>
 network 192.168.250.4/30 area 0.0.0.0         <transport network>
 area 0.0.0.0 authentication message-digest
 area 0.0.0.4 authentication message-digest
 area 0.0.0.4 stub                             <OCI>
!
…
```

Core Router: OSPF configuration

```
set protocols ospf parameters abr-type cisco
set protocols ospf parameters router-id 1.1.1.1
set protocols ospf passive-interface default
set protocols ospf passive-interface-exclude wg0

set protocols ospf area 0 area-type normal
set protocols ospf area 0 network 192.168.250.4/30
```

Core Router: OSPF configuration

```
#local area of your private network
set protocols ospf area 1 area-type stub
set protocols ospf area 1 network 192.168.0.0/19
set protocols ospf redistribute connected metric-type 1
#set protocols ospf redistribute static metric-type 1

#WireGuard
set interfaces wireguard wg0 ip ospf authentication md5 key-id 1 md5-key
<password>
set interfaces wireguard wg0 ip ospf dead-interval 40
set interfaces wireguard wg0 ip ospf hello-interval 10
set interfaces wireguard wg0 ip ospf priority 1
set interfaces wireguard wg0 ip ospf retransmit-interval 5
set interfaces wireguard wg0 ip ospf transmit-delay 1
```

The big final!

The big final: start daemons

If there isn't a configuration mistake, everything **should be ready**. Let's start all related daemons on the OCI routing host:

```
sudo systemctl restart zebra
sudo systemctl restart ospf6d
sudo systemctl restart ospfd
```

The big final: check OSPF neighborship

Validate that OSPF communication works:

```
oci# sudo vtysh

Hello, this is Quagga (version 1.2.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

oci# show ip ospf neighbor

Neighbor ID     Pri State           Dead Time Address         Interface
1.1.1.1           1 Full/DROther      35.914s 192.168.250.5   wg0:192.168.250.6
```

The big final: check routing

Validate that OSPF routes are provided:

```
oci# show ip route
O>* 192.168.0.0/24 [110/11] via 192.168.250.5, wg0, 1d22h17m…
O>* 192.168.1.0/24 [110/11] via 192.168.250.5, wg0, 1d22h17m
O>* 192.168.2.0/24 [110/11] via 192.168.250.5, wg0, 1d22h17m
O>* 192.168.3.0/24 [110/11] via 192.168.250.5, wg0, 1d22h17m
O>* 192.168.4.0/24 [110/11] via 192.168.250.5, wg0, 1d22h17m
```

```
oci# show ip ospf route
============ OSPF network routing table ============
N IA 192.168.0.0/24         [21] area: 0.0.0.0
                            via 192.168.250.5, wg0
N    10.0.0.0/24            [10] area: 0.0.0.4
                            directly attached to ens3
…
```

The big final: check OSPFv3 (IPv6) neighborship

Validate that **OSPFv3** communication works:

```
oci# show ipv6 ospf neighbor
Neighbor ID     Pri    DeadTime  State/IfState          Duration I/F[State]
1.1.1.1           1    00:00:34   Init/PointToPoint  5d00:22:23 wg0[PointToPoint]
```

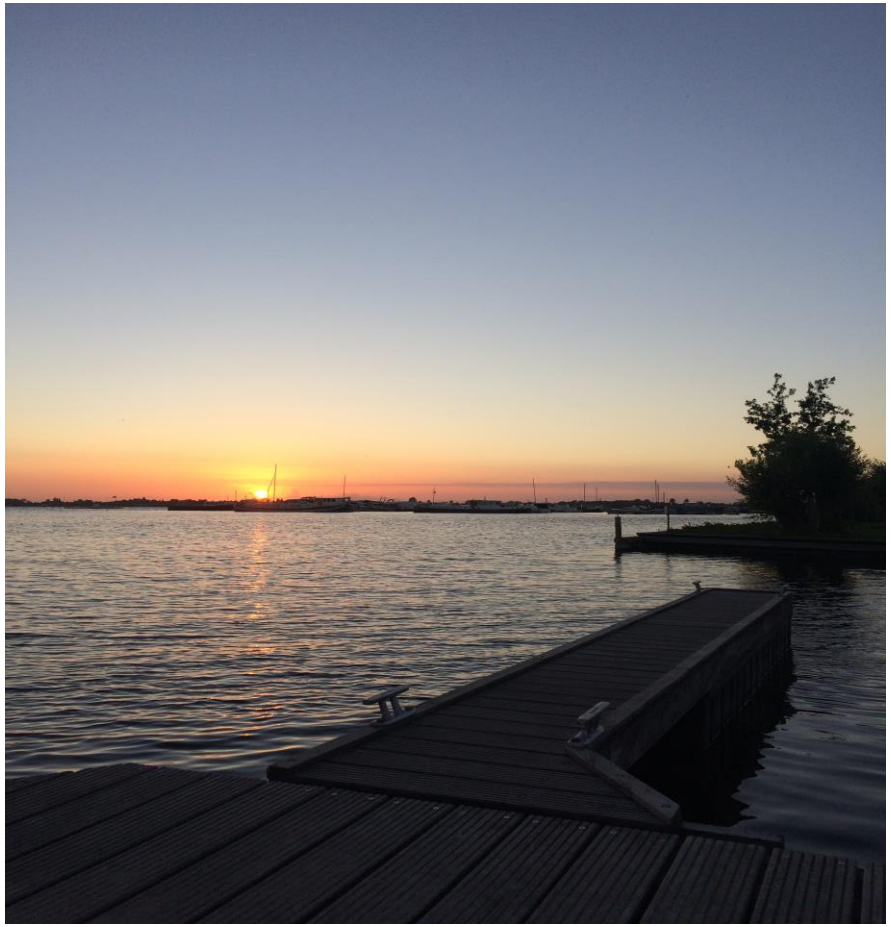
- Unfortunately, I couldn't establish a neighborship via OSPFv3 (WireGuard)
  - Turns out to be an **implementation issue** with the router.
  - **Support request** has been **closed** by the vendor -> WireGuard is **community driven**
  - OSPFv3 via LAN (ethX) works fine
  - **Static routes** did help for the moment

The big final: OSPFv3 over ethernet

Just providing evidence!

```
admin@thewall:~$ show ipv6 ospfv3 neighbor
OSPFv3 Process (*null*)
Neighbor ID     Pri   State           Dead Time   Interface  Instance ID
192.168.   .2     1   Full/DR         00:00:29    eth1       0
admin@thewall:~$ show ipv6 route ospf6
IP Route Table for VRF "default"
O E2   ::/0 [110/20] via fe80::[redacted]:526, eth1, 02w4d08h
O E2   2001:[redacted]28 [110/20] via fe80::2[redacted]6, eth1, 02w4d10h
O E2   2001:[redacted]28 [110/20] via fe80::2[redacted]6, eth1, 02w4d10h
O E2*> 2a00:[redacted]64 [110/20] via fe80::2[redacted]6, eth1, 02w4d08h
O IA*> 2a00:[redacted]64 [110/2] via fe80::26[redacted], eth1, 00:06:48
O IA*> 2a00:[redacted]64 [110/2] via fe80::26[redacted], eth1, 00:06:48
O IA*> 2a00:[redacted]64 [110/2] via fe80::26[redacted], eth1, 00:06:50
O IA*> 2a00:[redacted]64 [110/2] via fe80::26[redacted], eth1, 00:06:48
O IA*> 2a00:[redacted]64 [110/2] via fe80::26[redacted], eth1, 00:06:48
O      2a00:[redacted]64 [110/1] via ::, eth0, 00:06:48
O      2a00:[redacted]64 [110/1] via ::, eth0.20, 00:06:48
O      2a00:[redacted]64 [110/1] via ::, eth3.30, 00:06:48
O      2a00:[redacted]64 [110/1] via ::, eth3.40, 00:06:48
O      2a00:[redacted]64 [110/1] via ::, eth4, 00:06:48
O      2a00:[redacted]64 [110/1] via ::, eth4.60, 00:06:48
O      2a00:[redacted]64 [110/1] via ::, eth3, 00:06:48
O      2a00:[redacted]64 [110/10] via ::, vtun0, 00:06:50
O    *> fd02:[redacted]56 [110/0] via ::, Null, 02w4d10h
O      fd02:[redacted]64 [110/1] via ::, eth0, 02w4d10h
O      fd02:[redacted]64 [110/1] via ::, eth0.20, 02w4d10h
O      fd02:[redacted]64 [110/1] via ::, eth3.30, 02w4d10h
O      fd02:[redacted]64 [110/1] via ::, eth3.40, 02w4d10h
O      fd02:[redacted]64 [110/1] via ::, eth4, 02w4d10h
O      fd02:[redacted]64 [110/1] via ::, eth4.60, 02w4d10h
O      fd02:[redacted]64 [110/1] via ::, eth3, 02w4d10h
O      fd02:[redacted]64 [110/10] via ::, vtun0, 02w4d10h
O E2   fd93:[redacted]10/20] via fe80::2[redacted] eth1, 02w4d10h
O      fd93:[redacted][110/3] via ::, wg1, 02w4d10h
O      fd93:[redacted][110/3] via ::, wg2, 02w4d10h
O      fd93:[redacted][110/1] via ::, eth1, 02w4d10h
O IA*> fdd2:[redacted]10/2] via fe80::2[redacted] eth1, 02w4d10h
O E2*> fdd2:[redacted]10/20] via fe80::[redacted], eth1, 02w4d10h
admin@thewall:~$
```

Summary



Conclusion

- It’s possible to connect the **Oracle Cloud Free Tier** to your home network with the help of **IPv6** and a **WireGuard VPN**
- An implementation of **OSPF** did work for IPv4
- Unfortunately, there are **issues** with the WireGuard and **OSPFv3** (IPv6) implementation using an EdgeRouter
  - https://github.com/WireGuard/wireguard-vyatta-ubnt/issues/86
  - Looks like the issue is specific to Vyatta and the EdgeRouter implementation
- Find all detailed installation steps at:
  - https://blog.dieschmitterlinge.de

What next?

- **AVM** announced WireGuard for **FRITZ!OS** > 7.39 (lab)
  - Might **make things easier** for small networks
- As time allows:
  - Flash **OpenWRT** into my EdgeRouter-X
  - Consider **BGP** as replacement for the missing OSPFv3 capabilities
  - Learn **Perl** and fix the code ☺
- May be, a version 2.0 of the presentation

Thanks for your attention!