

Roope Simola

Data Centre Network Design and Development Collapsed Leaf -Design

Bachelor's thesis

Information technology

Cybersecurity

2020



South-Eastern Finland
University of Applied Sciences

Tekijä	Tutkinto	Vuosi
Roope Simola	Insinööri (AMK)	2020
Työn nimi		56 sivua
Datakeskuksen verkon suunnittelu ja kehittäminen: Collapsed Leaf -design		45 sivua liitteitä
Toimeksiantaja		
Ekami		
Työn ohjaaja		
Vesa Kankare		
Tiivistelmä		
<p>Tässä opinnäytetyössä tutkittiin Proxmox-palvelimissa olevan FRRoutingin avulla spine-and-leaf-arkkitehtuuriin perustuvan datakeskuksen suunnittelua, jossa BGP-EVPN olisi sisäistettynä datakeskuksen palvelimiin. Tavoitteena oli suunnitella pienimuotinen datakeskus, joka hyödyntäisi nykyaikaisia datakeskustekniikoita ja ratkaisuja verkon vaivattoman skaalauksen ja sen sisäisen sekä ulkoisen liikenteen mahdollistamiseksi.</p> <p>Opinnäytetyö on kehittämistutkimus, joka suoritettiin virtuaalilaboratoriossa. Opinnäytetyön virtuaalilaboratoriossa testattiin ja kokeiltiin erilaisten iteraatioiden kautta verkon rakennetta ja toimintaa tiedon keräämiseksi. Verkon simuloinnin kautta oli mahdollista testata tarvittavien ohjelmistojen asennus, sekä tutkia verkon käyttäytymistä. Tällä tapaa valittu lähestymistapa muistutti kokemuksellista oppimisteoriaa, sillä jokainen iteraatio perustui kokeelliseen sykliin. Lisäksi tutkimusta tehtiin liittyen spine-and-leaf-arkkitehtuuriin perustuvaan kirjallisuuteen, sekä selvitettiin lähteiden avulla tarvittavien ohjelmien asennusta ja toimintaa.</p> <p>Tämän tutkielman tuloksena oli se, että FRRoutingin avulla on mahdollista suunnitella pienimuotinen datakeskus, jossa spine-and-leaf-arkkitehtuurin leaf -komponentti on sisäistetty palvelimiin fyysisten laitteiden käyttämisen sijasta. Tämän perusteella oli mahdollista hyödyntää Proxmoxilla tehtyä palvelinta siten, että se osallistui suoraan BGP-EVPN:ään, jossa palvelin käytti omia VXLAN-liitäntöjä sisäiseen kommunikaatioon. Tämä tulos merkitsi sitä, ettei leaf-kytkimiä tarvittaisi lopullisessa asennussuunnitelmassa.</p> <p>Tutkimus mahdollisti asennussuunnitelman tekemisen, joka perustui spine-and-leaf-arkkitehtuuriin, jossa VXLAN-ominaisuuksia hyödynnettäisiin käyttämällä BGP-EVPN:ää. Suunnittelun yhteydessä luotiin lisäksi asennusopas, jossa oli dokumentoitu FRRoutingin Proxmox-asennus ja asennukseen liittyvien erilaisten liitosten luonti. Näiden kahden asiankirjan perusteella opinnäytetyö saavutti asetetut tavoitteet tuottaen asennussuunnitelma pienimuotisesta datakeskuksesta, jossa hyödynnettäisiin nykyaikaisia tekniikoita datakeskuksen rakentamiseen.</p>		
Avainsanat		
VXLAN, EVPN, Proxmox, FRRouting, Datakeskus		

Author (authors)	Degree title	Time
Roope Simola	Bachelor of Engineering	December 2020
Thesis title Data centre network design and development: Collapsed leaf -design		56 pages 45 pages of appendices
Commissioned by Ekami		
Supervisor Vesa Kankare		
<p data-bbox="147 806 1464 842">Abstract</p> <p data-bbox="147 842 1464 989">This thesis studied the implementation of FRRouting in Proxmox to create a spine-and-leaf architecture-based data centre design that would have BGP-EVPN introduced into servers. The aim was to create a small-scale data centre, which would have the benefits of modern techniques to scale and use of VXLAN for communication.</p> <p data-bbox="147 1031 1464 1241">A virtual laboratory was used to take the approach of design research, where various iterations were tested and experimented on to gather data. Having simulated the network, observation of behaviour and researching implementation of necessary software was made possible. In this way, it resembled the experiential learning theory. In addition, research into literature sources related to spine-and-leaf -design was done along with studying the various software used.</p> <p data-bbox="147 1283 1464 1461">In the conclusion of this thesis, it was found that implementing FRRouting made it possible to design a small-scale data centre, where the leaf -component was included into the server nodes rather than onto a separate physical switch. This meant that Proxmox -server could be harnessed to participate directly in BGP-EVPN and have its own VXLAN - interfaces.</p> <p data-bbox="147 1503 1464 1682">This allowed the formulation of a design, based on spine-and-leaf architecture, which utilized BGP-EVPN for VXLAN -capabilities. With the design, an installation guide that documented findings regarding implementation of FRRouting on Proxmox was produced with this design. Through these two documents, the thesis achieved the goal of producing a design of a small-scale data centre that utilizes the modern spine-and-leaf architecture.</p>		
Keywords VXLAN, EVPN, Proxmox, FRRouting, Data centre		

CONTENTS

1	INTRODUCTION	7
2	RESEARCH METHODS	9
2.1	Researching the design in a laboratory	9
2.1.1	Virtual Laboratory	10
2.1.2	Testing through iterations	10
2.2	Analysing the results and their reliability	11
3	DATA CENTRE PRINCIPLES	11
3.1	Spine-and-leaf architecture	12
3.2	Virtual Extensible Local Area Network	13
3.3	Underlay	13
3.4	Virtualization	14
4	DESIGNING A DATA CENTRE	14
4.1	Collapsing leaf into a Proxmox hypervisor	14
4.1.1	FRRouting	16
4.1.2	Ifupdown2	16
4.1.3	BGP-EVPN	16
4.2	Spine and Core	17
4.2.1	Switch Virtual Interface	17
5	RESEARCHING DESIGN	18
5.1	Testing environment	18
5.1.1	FRRouting	19
5.1.2	Implementing ifupdown2	20
5.1.3	BGP-EVPN implementation	20
5.1.4	Clustering Proxmox	21
5.2	Iteration two	22

5.2.1	Logs	22
5.2.2	Container to container connection	22
5.2.3	Ceph -storage	23
5.3	Iteration three.....	23
5.3.1	NGINX	24
5.3.2	Storage connected to spine	25
5.4	Iteration four.....	26
5.5	Iteration five	27
5.5.1	VLAN to VTEP	28
5.5.2	OSPF	29
5.5.3	VXLAN -hub bridge.....	29
5.6	Iteration six	29
5.6.1	SVI in spine.....	30
5.7	Iteration seven	31
5.7.1	Routing without VRF	32
5.7.2	VRF routing.....	33
5.7.3	Testing the VRF	34
5.8	Iteration eight.....	35
5.8.1	FRRouting to Arista VRF	36
5.8.2	Testing connection.....	38
6	RESULTS	39
6.1	Data centre -design	39
6.1.1	MX250 connection to spines.....	40
6.1.2	Use of eBGP.....	41
6.1.3	Virtual routing and forwarding	41
6.1.4	Cluster communication and Management.....	42
6.1.5	Storage solutions	42

6.1.6	Choice of switch.....	43
6.2	Installation guide for Proxmox.....	44
6.2.1	Proxmox installation.....	44
6.2.2	Configuration of interfaces.....	45
6.2.3	Variant bridge -interfaces.....	45
6.3	Imitation of MX250.....	47
6.4	Outside of scope.....	47
7	CONCLUSION.....	48
8	DISCUSSION.....	49
8.1	Reliability of research.....	49
8.2	Further research and development.....	50
8.3	Introspection.....	50
	REFERENCES.....	52

LIST OF FIGURES

APPENDICES

Appendix 1. Data centre design document

Appendix 2. Proxmox with FRRouting: Installation guide for data centre purposes

1 INTRODUCTION

Ekami, the Joint Authority of Education of Kotka-Hamina Region Group, has a data centre that is aimed to be used as a learning environment for their students. With the current installation, steps towards virtualization have been made, but the current plan of expanding the capacity and scale of the data centre to house a robust and redundant virtualized environment has been in the works for a while.

Presented with a set of equipment: several Cisco catalyst 2960 -switches, an old Paloalto firewall, PS4100 and PsV2020 storage, with four Dell Poweredge 420 - servers, two Dell Poweredge 630 -servers and a request to design a data centre, the starting off point had to be mapping out what was at hand and what could be done with them. After the initial research into capabilities of the devices, three designs were presented to focus the research on a single design. A design using nothing, but the equipment available, a design introducing two new pieces of equipment and a design introducing five new pieces of equipment were presented to act as the framework for the data centre design based on modern techniques.

As there was no budget, no knowledge of how much cost could occur, the option that utilized two new pieces of equipment was chosen and became the beginning for the designing of the data centre by the choice of the client. After that it became a task to research how to make the design and implement it. Challenge was that the existing equipment should not have gone to waste, as well as the old firewall that was being replaced with a new Meraki MX250 -firewall as a policy device for the data centre that is in active use. The introduction of the new policy device was inevitable, with the deprecation of older firewall to cut costs. Another request was that shifting from use of ESXi to Proxmox would be done.

This would mean that acquiring the switches for testing could incur costs, and some of the devices possibly being in use or not available to be experimented with, which resulted in the need to research the techniques through other means. In addition to not having access to switches that would or could be used for testing, the two Poweredge 630 and PsV2020 were in active use at the start of

the design process, meaning they could not be used directly for testing. So, rather than incur costs for the client for experimenting and purely theoretical speculation on what should and could be done, the research through virtualization was chosen. Through virtualization the design could be tested and experimented on using various applications, as well as see if it was possible to implement the design to begin with.

With the need for streamlining and homogenization of the framework for the data centre, as well as the virtualization environment, a more modern data centre and cloud computing solution would be needed. Utilizing the existing equipment to increase the scalability, redundancy and utility of the installation is to be planned. This entails architectural design, addressing scheme and introduction of necessary components and elements that make it possible to run a functional data centre – design research.

Not only does the data centre need to be functional, operable and robust, the services it will provide need to be accessible by both the students and teachers. To this end, turning the data centre's pooled resources into services that can be accessed through a browser from school premises and outside needs to be considered. However, the initial aim of the data centre is to allow students to access Virtual Machines (VM) that will provide them the ability to interact, learn and practice the use of the different Operating Systems (OS), to which the intention is to provide an access to the VMs as well as access outside the network for the VMs.

However, the design should not be limited to the present, as considering the ability to adjust and shape the data centre into a new design or addition of new solutions should be possible.

With these come many challenges; from design principles to expense, the need to be able to provide a functioning solution will be behind research and development. Especially with the starting point in question that is six servers, with Dell storage devices that are connected with Serial Attached SCSI (SAS) and

simple ethernet RJ45 connections, intended to be connected using several Cisco Catalyst 2960 switches and having the data centre connected to a Meraki MX250 firewall at the edge.

2 RESEARCH METHODS

This research raises several questions at its forefront to be answered.

- How can a cost-efficient data centre be built?
- Does the final design fulfil the data centre's requirements?
- Does this fit a modern data centre solution?
- How to provide access to the resources of the data centre?
- Is the final design scalable?

Based on the questions and the problem presented, the research does not focus on explaining a phenomenon or understanding it. Rather, the research is focused on solving the problem, coming up with a solution and making change. This is the reason why the research would be categorized as interventionistic research, which acts as a hypernym for design-based research, constructive research and action research. As interventionist research acts as a hypernym, the more precise form of research in question would be design-based research, which aims to produce change and amendment to the current design. (Kananen 2017.)

In the design research literature, reports, theories and models, in addition to other sources, related to the field of study are utilized. Problems are observed, defined and the research is adjusted due to these problems if the original outcome becomes unattainable due to them. Design research and action research have a thin line differentiating them, where both can remain as suggestive and not result in actual change. (Kananen 2012, p. 51-84.)

2.1 Researching the design in a laboratory

One of the tools that Koskinen et al. (2011, p. 51-59) bring up in their example is the use of a laboratory to research design. The use of a controlled laboratory

environment to test and experiment on a design, going through several different designs, or iterations, to come to a final design. The intention is to take out variables that might interfere or hinder the studying of a phenomenon and experiment without these variables, allowing the studying to focus on eliminating alternative explanations. A laboratory can also be beneficial for studying, as equipment to provide more accurate observation can be included in it. With detailed documentation, laboratory environments and experiments can be replicated in other laboratories to rule out errors caused by the environment or research culture.

2.1.1 Virtual Laboratory

Southern University of Applied Sciences (XAMK) provides their ICT -students with access to a Virtual Laboratory, VirtualLab (VLAB), for testing and learning purposes. It provides access to numerous firewalls, routers, switches, OS and many other appliances in a virtualized environment for testing and studying. The introduction of OS not yet provided is also possible. As the environment is virtualized, accessing it is possible through a browser, allowing remote studying, research and experiment. (Kettunen 2019.)

VLAB will be used to test various iterations, experiment with the design and components relating to the network. With the possibility to observe traffic from device to device, it is possible to monitor the network traffic, as well as a virtualized environment allows avoiding acquiring physical devices for testing and thus avoids cost.

2.1.2 Testing through iterations

Because the testing is taken in steps, each iteration acts as its own experience and view into the functions of the systems involved and have their own experiment involved, it could be likened to the Experiential Learning Theory. In it, knowledge is produced by transformation of experience; a cycle of concrete experience, reflective observation, abstract conceptualization and active experimentation is formed to attain knowledge. It incurs that experiencing

something leads to reflection of it, before followed by thinking on the concept itself and finally leads to acting on it – until the circle continues once more. (Kolb & Kolb 2011).

2.2 Analysing the results and their reliability

To measure the success of the endeavour, it is important to analyse the results and reflect them against the research questions presented. This means that the data gathered should provide answers, or means to answer, the questions presented. The reliability of the research is also important, as it requires good documentation. This also means that the results and claims made by the research have to be testable and repeatable. (Kananen 2012, 161-172; sb. 191-192)

3 DATA CENTRE PRINCIPLES

Modern data centres have developed greatly, as more and more of the services, applications and platforms are directing themselves towards virtualization to spare the expense of housing and maintaining their own data centre. Due to the constant need to expand or decrease based on demand, as well as the ever-improving and developing hardware, it has become necessary to have a robust, redundant and modular networking environment that is capable of adjusting with ease, as well as ensuring segregated environments for a large number of networks running within a data centre that allows the virtualization to communicate with ease. (Arista s.a.a.)

In cases like these, the typical Virtual Local Area Network (VLAN) no longer manages to handle it due to the limitations of the number of VLANs available. Not only does the relatively small number of VLANs pose an issue, but the need to add VMs, containers hosting services or more networking equipment must to be relatively simple, elegant and easy to do without the need for large scale alterations of the entire network infrastructure. (Arista s.a.b.)

When it comes to designing smaller scale data centres, it is also important to consider the expense of such an endeavour, as high-end networking equipment and high-speed connections can easily add up to a large expense. However, it is also important to consider the future and longevity of the network. Investing in an infrastructure capable of servicing for long and adapting to future changes causes a balancing act of expense and utility. (Al-Shawi 2016.)

3.1 Spine-and-leaf architecture

Spine-and-Leaf architecture is a modern solution for data centre design, which replaces the traditional 3-tier structure of Core, Distribution and Access layers. Whereas in the traditional structure all components are connected through several points to ensure redundancy, in Spine-and-leaf the spine is disjointed with each leaf connected to each spine with a single link. The Core and Distribution levels are usually collapsed into the Spine. (Arista a; Figure 1.)

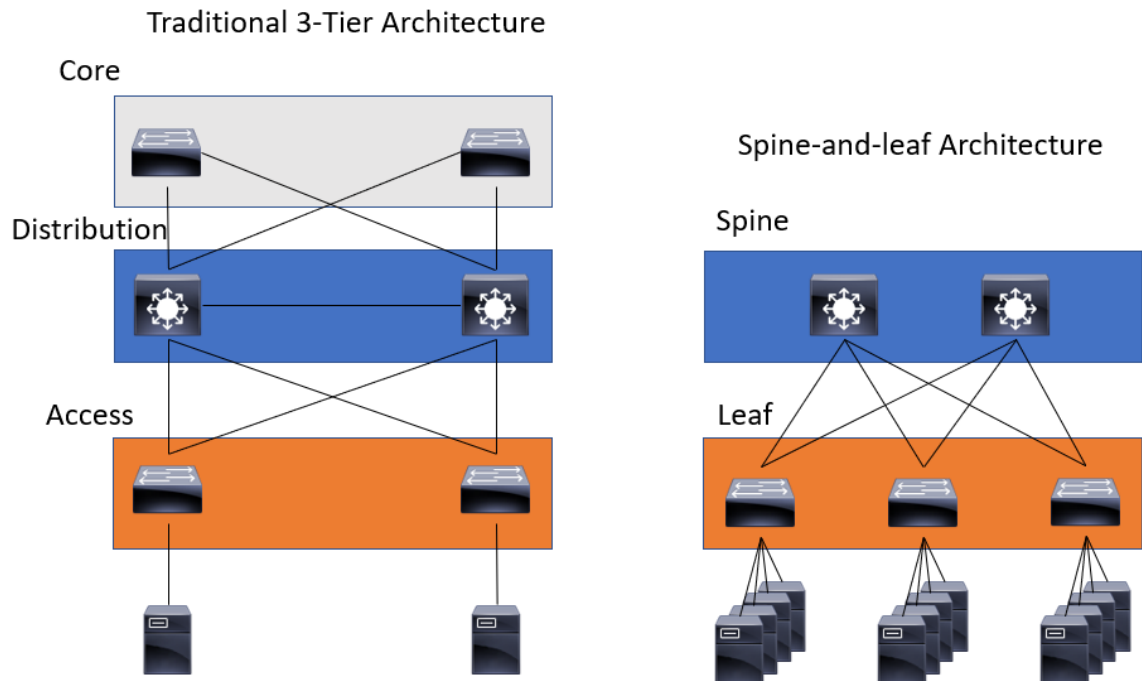


Figure 1 Traditional 3-Tier architecture compared to 2-Tier Spine-Leaf architecture

The redundancy is ensured through the use of Layer 3 (L3) underlay using Open Shortest Path First (OSPF), Intermediary System to Intermediary System (IS-IS) or Border Gateway Protocol (BGP), and an overlay is used to provide Layer 2

(L2) or L3 connectivity from connected hosts utilizing VXLAN and Virtual Routing and Forwarding (VRF). (Cisco s.a.)

3.2 Virtual Extensible Local Area Network

RFC 7348 (2014) worked on developing Virtual Extensible Local Area Network (VXLAN) for data centres to provide virtualized networking, which is referred to as an overlay. VXLAN would be capable of working together with multiple tenants, as well as address the issue of L2 networks use of Spanning-tree protocol to avoid loops in a network leading to sub-optimal use of the entire network. The use of VLANs to provide isolation of networks is also a cause of a problem, as a 12-bit VLAN ID limits the number of isolated networks to 4094 and this can be further limited because of STP.

VXLAN addresses this issue by providing a new 24-bit segment ID, called Virtual Network Identifier (VNI), which allows theoretically the isolation of over 16 million different networks that are unaware of the underlay. This does not completely remove the use of VLANs in certain cases, however, it allows their use in tandem to provide additional solutions to internal connections of the network and the VMs.

To ensure this isolation, as well as connectivity, Virtual Tunnel End Points (VTEP) are used for both ingress and egress of the network tied to a VNI, which means that only the host tied to the correct VNI can participate in the communication of the virtualized network.

3.3 Underlay

Functioning as the backbone of the network, an underlay is provided by utilizing the routing protocols OSPF, IS-IS or BGP which connect the devices of the network together and allow the overlay to travel through the network unaware of underlays existence. Protocol Independent Multicasting (PIM) or Ethernet Private Network (EVPN) could be used to connect the VTEPs together, with the benefit of utilizing Equal Cost Multi-Path (ECMP) routing. ECMP ensures that each link is

utilized for traffic, providing increased optimization of the network and redundancy by allowing several paths to be used simultaneously. (Cisco, Cumulus.)

3.4 Virtualization

With virtualization, there is a need to be able to harness the hardware of a server device to be utilized by virtualized devices, without cost to performance or extraneous tasks performed by obsolete applications. Hypervisors are means to this end, divided into two types, type 1 and type 2, which are either software installed like an OS or simply run like a program on top of a pre-existing operating system. (Opensource s.a.)

Proxmox is a type 1, 'bare-metal', open source hypervisor which is based on Debian GNU/Linux, meaning it is a free hypervisor software that is available to those willing to acquire and install it, which allows, creation of VMs and LXC containers. (Proxmox 2020.)

4 DESIGNING A DATA CENTRE

When it comes to designing a robust, redundant, scalable and futureproof data centre, there are many things to consider. Cost, equipment, maintenance and ease of installation of new servers are relatively important things to consider. A typical data centre design has both spine and leaf switches, which can mean numerous devices that cause expense. The typical design also needs to be able to adjust and endure changes to the infrastructure brought upon by the introduction of new servers or decommission of old ones. Access to storage devices, sharing storage needs to be considered, and enduring of failures of both networking equipment and servers cannot cause the entire system to fail or become inaccessible. (Varnum 2018.)

4.1 Collapsing leaf into a Proxmox hypervisor

Due to the fact that starting point was several Cisco Catalyst 2960 -switches, and Meraki MX250 firewall as the edge, which are not suitable for the creation of a

VXLAN capable infrastructure (Cisco 2014, Cisco Meraki 2020c), and the need to consider expenses as acquisition of several new networking devices of a relatively high cost for the infrastructure, which typically has two spines and three leaf -switches, was not viable. So, a new approach would be needed, an approach that would combine several parts of the network to reduce costs.

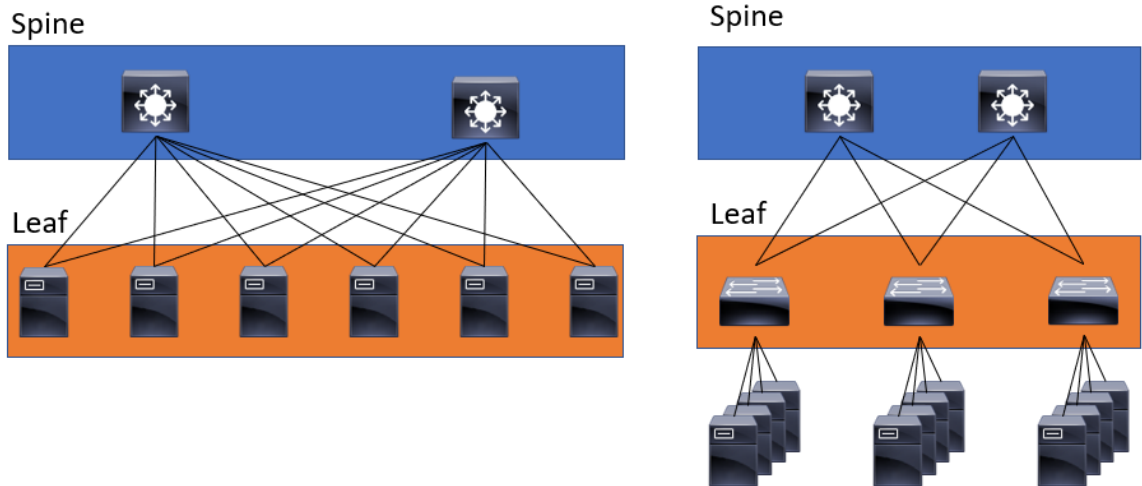


Figure 2 Collapsed -leaf next to traditional spine-and-leaf

Collapsing is a concept already used with the traditional 3-tier network architecture, where usually the Core -level and Distribution -level are combined or collapsed to reduce the number of devices needed for the network. While this reduces the number of devices, it also places more emphasis on devices that now must take on two or more unique tasks previously handled by separate devices. In figure 2, there is an example of the differences that this kind of setup would provide. However, the impact of having Linux act as a router is not too heavy in the end and the idea of using BGP routing in a hypervisor is not a new one (Bernat 2019).

With Proxmox, due to it being based on Debian GNU/Linux, as well as open source, it is possible to collapse the leaf into it and turn it into a hypervisor that also takes on the role of a router. What makes this possible is FRRouting, a free routing software, in unison with using ifupdown2 to introduce VXLAN -interfaces that can be bridged to the VMs (Cumulus).

4.1.1 FRRouting

FRRouting is a free IP routing software that is a fork of Quagga, which has been written primarily in C. With it, standard routing protocols like BGP, OSPF and IS-IS can be implemented, and it is modelled after Linux kernel. The routing table is installed into the OS kernel, which allows the kernel networking stack to make the forwarding decisions. (FRRouting s.a.)

Cumulus Linux utilizes FRRouting, focusing its VXLAN routing on BGP-EVPN that can have PIM added to it in order to optimize the behaviour of the network. It also supports using VRFs for multi-tenant routing (Cumulus s.a.)

Since using internal BGP (iBGP) or external BGP (eBGP) can be used in the same process as BGP-EVPN, it can be used to simplify and unify the network design. (Cumulus.)

4.1.2 Ifupdown2

Ifupdown2 is a network interfaces manager, which is intended to be an improvement over the older Debian Gnu/Linux network interface manager ifupdown. It is used to simplify configuration and provide modules that support creation of VXLAN -interfaces and VRF. (Fortin 2018.)

4.1.3 BGP-EVPN

In RFC 7432 (2015), later updated in RFC 8584 (2019), BGP-EVPN was designed to be based on Multiprotocol Layered Switching (MPLS), which has the intention of introducing the use of Ethernet Tag ID, and automatic VTEP discovery as VXLAN lacks a control plane. The idea is that by adding new overhead at L3 without altering the original packet, the traffic can be directed by using this overhead – rather than the information contained in the original packet sent.

4.2 Spine and Core

Because the VMs need a connection outside of the network, to provide services outside, it is necessary to have means to direct the traffic to an edge device that either does the task routing traffic or redirecting it outside. In a 3-tier network this task is done by the core, however in Spine-and-leaf this task is left to either to the spine or a device capable of handling VTEPs that can also route traffic. (Cumulus.)

In the network in question, the environment is intended to connect to a Meraki MX250 firewall – which in the end present several problems to solve. The firewall has not been designed to act as a VTEP and it does not participate in STP (Cisco Meraki 2020a.), which prevented certain design choices. As networks isolated by VXLAN were intended to be provided access to and through the policy device, a VTEP outside the firewall for the management of traffic would be required. Since the need to provide ‘normal’ outside network connection to the VMs was also required, simply introducing an Application Delivery Controller (ADC) or HTTP/HTTPS proxy would not suffice alone, because VTEP would be necessitated.

4.2.1 Switch Virtual Interface

Because of the situation presented by having Meraki MX250 firewall as the edge device, it is necessary to force the spine to take care of VTEPs regarding networks that are intended to be routed outside and through the firewall. With MX250 not participating in STP but allowing Cisco -based “**switchport trunk**” configuration, it is necessary when a VTEP is bound to a VLAN to terminate the tunnel in the spine that there is a Switch Virtual Interface (SVI) for it. (Cisco Meraki 2020b.).

Because of the typically disjointed nature of spine-and-leaf -architecture, it means that in order to ensure that STP or the firewall do not cause complications regarding traffic engineering, duplication or broadcast storming that the spines are joined together with a trunk that allows formulations of a default gateway SVI

for the VNI that is tied to the VLAN. While this introduces the need for STP, which will make the optimal use of multiple pathways difficult due to its nature of blocking paths (RFC 7348). This means that traffic engineering regarding STP is necessary to ensure optimal use of multiple links to the firewall from the spine. (Cisco Meraki a.).

5 RESEARCHING DESIGN

XAMK provides their ICT-students with access to a Virtual laboratory - environment, which was utilized to research and to study the behaviour and implementation of L3 routing capabilities into Proxmox and implementation of BGP-EVPN. VLAB allowed the research of the network's behaviour when devices suitable for real-world application were unavailable or acquiring them would be expensive for testing purposes. With the virtualization of the testing environment, the benefit of not damaging or losing valuable equipment would be achieved at the cost of performance and limitations presented by hardware resources allocated to the laboratory environment.

5.1 Testing environment

Figure 1 shows the first iteration of the test environment, with three VMs used to host Proxmox VE version 6.2 and two devices designated as spine routers using Arista vEOS 4.23.0F with a third switch with Cisco CSR1000 to act as a management switch and a management PC running Kali Linux. An outside connection is presented directly to the nodes, as difficulties with using management switch resulted in no traffic or loss of all traffic after an undetermined period of time.

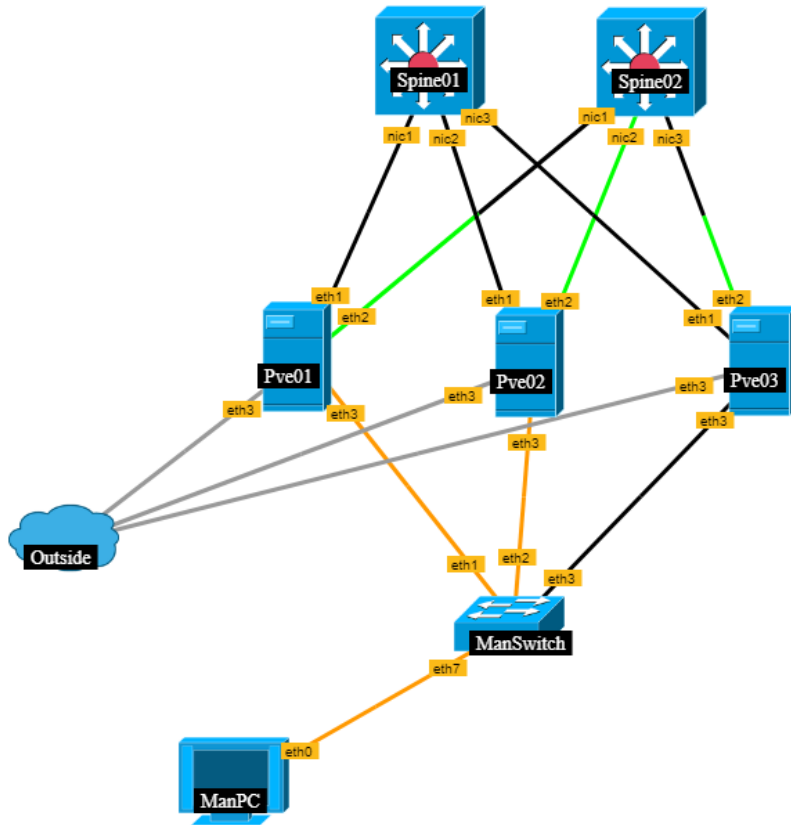


Figure 3 Virtual laboratory testing environment, first iteration

The goal of this initial iteration was to simply introduce FRRouting, version 7.1, and VXLAN -interfaces into Proxmox hypervisor and create a BGP-EVPN connection between the three nodes, while creating a cluster. The Pve -devices have four interfaces in total, 8192 GB of ram, 10 GB of disk size and two CPU cores in qemu64 model. Difference in Spine -devices is the number of interfaces which is eight. Management -switch has been allocated 2048 GB of ram and two CPU cores, with disk size of 10 GB.

5.1.1 FRRouting

The introduction of FRRouting was relatively simple through introduction of the repository into the hypervisor, with its behaviour of allowing direct configuration by altering the `frr.conf` -file and use of `vtsh` to enter configuration mode similar to that of a typical router/switch made it easy to configure and test. Because of the

connection to Kernel, it does not require, but allows the configuration of interfaces through FRRouting or /etc/network/interfaces -file (Appendix 2.)

5.1.2 Implementing ifupdown2

Introducing the VXLAN -interfaces and use of VNI proved to be more difficult, as by default Proxmox is set to use the subscription repository. This will result in behaviour that can lead to either failure of installation of software or guidance by the system which if not observed carefully, will result in purging vital system functions. Altering the repositories to use another repository was a necessity to introduce ifupdown2 to allow creation of VXLAN -interfaces into the etc/network/interfaces -file. Because of the existence of iproute2, it was possible to introduce VXLAN -interfaces through CLI, yet to ensure permanence and the ability to back up the configurations – the use of ifupdown2 became easier. (Appendix 2.)

Another benefit from ifupdown2 was that Cumulus -Linux guides would become somewhat accurate and helpful to the configuration of the network interfaces and introduction of VRF.

5.1.3 BGP-EVPN implementation

In appendix 2 there are examples with the introduction of FRRouting and ifupdown2, the next step was creating BGP connection between the spine and the nodes. While FRRouting allows the use of IP unnumbered lo for BGP, meaning the use of loopback address could be utilized for the interfaces rather than assigning address for each link, the lack of support for this meant that using addressing scheme for the links was necessary. In the testing environment 172.16.0.0/8 -range was chosen, allowing the use of unique link addresses for each node and spine connection. However, due to the use of Debian GNU/Linux as a L3 router a limitation was found. Since typically the links are made using mask /31 to reduce the excess addresses for a point-to-point connection, and the mask /31 was not available in Linux, the mask /30 was used to create connection between the spine and the leaf.

However, in addition to providing the addresses to the interfaces, it was also necessary to take into consideration the new overhead and because of this the MTU of interfaces was also altered from default 1500 to 1550 to ensure that traffic could be handled.

Due to the relatively small differences between Arista, Cisco and FRRouting when it comes to syntax, the implementation of BGP neighbourhoods was very simple. Since the spine was allocated the ASN 65350 and Leafs were given a single ASN 65351 to create the eBGP connection, it was also necessary to introduce the use of **allow-as in** -command so that traffic would not be dropped by BGP from the Leaf seeing its own ASN. This configuration needs to extend to the L2VPN EVPN as well.

A benefit of FRRouting is the command “advertise-all-vni” that allows the VNIs are advertised automatically once configured correctly into the Proxmox. FRRouting will advertise them with the ASN and VNI identifiers, i.e. **import route 65351:600**, creating unique RD with the VTEP (loopback address in this case) for each VNI. However, it is still possible to advertise a VNI by manually setting the RD and importing/exporting of the routes.

5.1.4 Clustering Proxmox

Testing the use of BGP-EVPN network by introducing the cluster communication through a VXLAN -interface was attempted, however, due to the fact that Corosync that takes care of this, is very sensitive to lag, the solution was not considered suitable. While connections were made, the dropping of cluster members was constant and thus using the management switch for communication was necessary. A problem that arose from this was the virtualization, which also added problems with the Cisco OS chosen for the management switch resulting in a similar behaviour of losing communication or communication timing out between nodes.

5.2 Iteration two

Changing the Cisco OS from the management switch to OpenWRT software for the testing environment to allow more stable clustering and testing of node to node container communication using Ubuntu 16.04 container image. The storage size of 10 Gb was deemed too small, especially if FRRouting was creating logs, and shifted to 100 GB to accommodate introduction of containers and container images. Testing of Ceph -storage was also done. Secondary laboratory without topological changes created, only alteration being that the spine has CRV1000 routing software to test BGP-EVPN syntax and connection differences from Arista and Cisco. Only notable change in syntax and requirement of route-map for Cisco routing software.

5.2.1 Logs

Because FRRouting is a routing software, which acts very similar to that of others, it also gathers information and create logs based on it. These logs are stored in Proxmox, especially if logging neighbour changes has been enabled in BGP. An error state, running for several days with the laboratory without interaction, resulted in bloating of logs. Coupled with the introduction of Ifupdown2 and FRRouting, the logs had bloated to a size capable of draining the remaining available space.

5.2.2 Container to container connection

With the introduction of VXLAN -interfaces bridging was necessary to ensure that the VMs and containers could attach to the network as well. A simple VXLAN - interface with a bridge to it allowed the connection between two containers, no addition of VLANs was necessary. The pinging from container to container functions, also attaching the correct VNI responding to the bridge in which the container is attached to. Pinging from bridge to bridge, with bridges having static IP and through Proxmox, there is a possibility of leaking and pinging separate VNI which does not replicate to the containers. (Figure 4.)

No.	Time	Source	Destination	Protocol	Length	Info
→ 61134	492186.27351...	192.168.60.12	192.168.60.13	ICMP	148	Echo (ping) request id
← 61135	492186.27944...	192.168.60.13	192.168.60.12	ICMP	148	Echo (ping) reply id
61136	492187.18088...	00:81:d7:57:7f:da	LLDP Multicast	LLDP	97	Msg = 00:81:d7:57:7f:da
<ul style="list-style-type: none"> ▶ Frame 61135: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits) on interface 0 ▶ Ethernet II, Src: 00:81:d7:57:7f:da (00:81:d7:57:7f:da), Dst: 00:55:46:24:e6:12 (00:55:46:24:e6:12) ▶ Internet Protocol Version 4, Src: 10.0.2.13, Dst: 10.0.2.12 ▶ User Datagram Protocol, Src Port: 48607, Dst Port: 4789 ▼ Virtual eXtensible Local Area Network <ul style="list-style-type: none"> ▶ Flags: 0x0800, VXLAN Network ID (VNI) <ul style="list-style-type: none"> Group Policy ID: 0 VXLAN Network Identifier (VNI): 600 Reserved: 0 ▶ Ethernet II, Src: f2:80:fc:5d:71:c2 (f2:80:fc:5d:71:c2), Dst: 66:07:e7:4d:e5:08 (66:07:e7:4d:e5:08) ▶ Internet Protocol Version 4, Src: 192.168.60.13, Dst: 192.168.60.12 ▶ Internet Control Message Protocol 						

Figure 4 Excerpt of PCAP of ICMP between two containers.

5.2.3 Ceph -storage

While Ceph requires multiple storages with which to operate proper, the intention of this testing was more towards handling the Ceph -communication and the use of static IP -address bridges tied to a VXLAN -interface and ease of installation. In the environment, it was possible to use the combination to achieve this, and the installation of Ceph is supported by Proxmox. However, due to the available equipment in the design, Ceph was ultimately considered an option for later.

5.3 Iteration three

In third iteration, the idea was to test the introduction of an NGINX run on a Debian Buster 10 that has FRRouting on it for connecting it to the BGP-EVPN and to act as a VTEP for connecting to Proxmox -nodes in order to have HTTPS -service up for using the Proxmox WebGUI. In addition to this, testing of introducing a storage device by connecting it directly to both spines was made. Testing would it be possible to utilize VTEP in spine to achieve the connection, as well as using the underlay to form the connection. A fourth node was introduced for testing the time it would take to introduce a new node to the network. (Figure 5.)

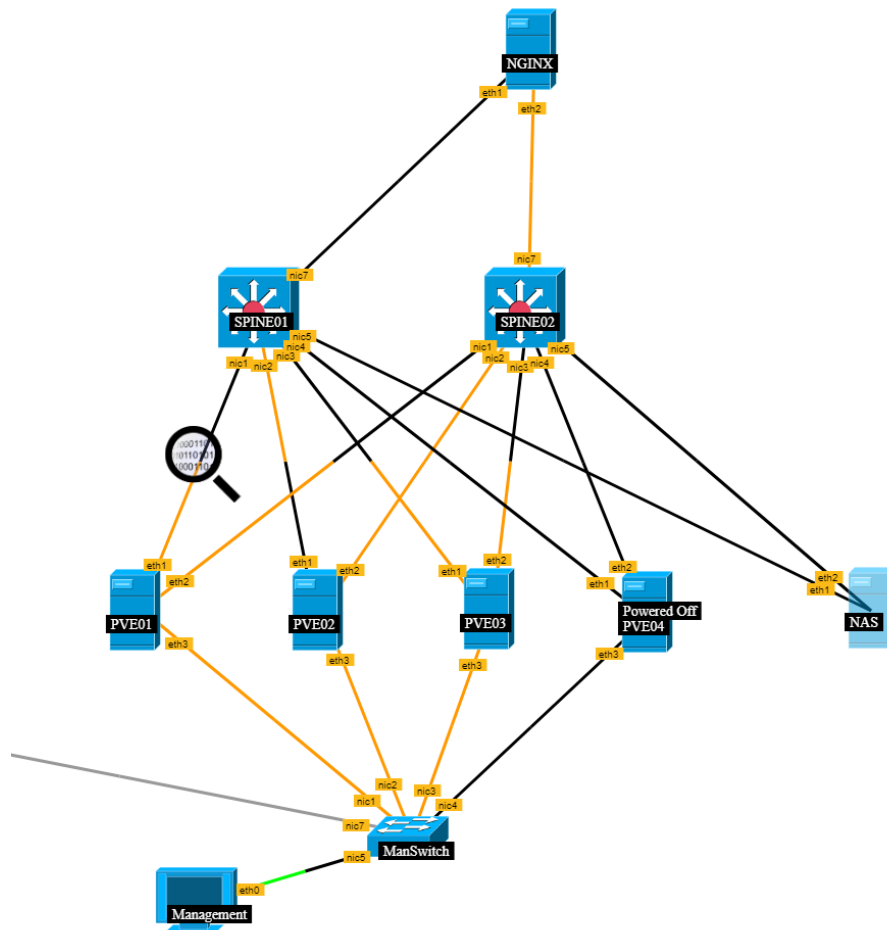


Figure 5 Third iteration: Introduce NGINX and storage

5.3.1 NGINX

Introducing NGINX on Debian Buster 10 was a relatively simple task, as the guide to installing FRRouting onto Proxmox functions the same. With installation of `ifupdown2` also possible, the configuration of a VTEP that connects to static IP bridges at the Proxmox nodes allowed Proxying the port 8006, which is the port of Proxmox WebGUI, from a loopback address assigned to the NGINX. In addition to this, rather than having just an HTTP connection, an HTTPS connection was made by copying the SSL -key from the cluster to the NGINX and this allowed the use of more secure form of connection. However, a problem arose from the inability to utilize noVNC to view the containers or the VMs hosted on a node that was evidently produced by the proxying not sticking. This problem was solved by introducing a simple line of code to the configuration of NGINX,

ip_hash, which ensured that the connection would stick to the instance and ensure noVNC visibility. (Figure 6.)

```

upstream proxmox {
    ip_hash;
    server 10.0.20.11:8006;
    server 10.0.20.12:8006;
    server 10.0.20.13:8006;
}

server {
    listen 80 default_server;
    rewrite ^(.*) https://$host$1 permanent;
}

server {
    listen 443;
    server_name _;
    ssl on;
    ssl_certificate /etc/nginx/ssl/cert.pem;
    ssl_certificate_key /etc/nginx/ssl/key.pem;
    proxy_redirect off;
    location / {
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_pass https://proxmox;
        proxy_buffering off;
        client_max_body_size 0;
        proxy_connect_timeout 3600s;
        proxy_read_timeout 3600s;
        proxy_send_timeout 3600s;
        send_timeout 3600s;
    }
}

```

Figure 6 NGINX configuration for Proxmox WebGUI in laboratory environment

5.3.2 Storage connected to spine

Figure 3 shows an NFS storage, using FreeNAS, which has been connected to the two disjointed spines to test connecting a storage unable to join the BGP-EVPN. While the use of VTEP ended at each SPINE and with the same VLAN and VNI functions, the possibility of miscommunication caused by the disjointed spines connected to a single bonded interface on NFS is a concern. The nodes were able to ping the storage – however due to problems between communication of the NFS and Proxmox, it was not possible to utilize it as the VM hard disk storage during the testing.

The other option of utilizing the underlay was tested, providing much the same result regarding communication problems – indicating that problems are more

between the FreeNAS and Proxmox or related to the virtual environment. Pinging a storage connected through static routing or attached to BGP functions normally.

5.4 Iteration four

While no changes are done to the environment, testing of PIM is performed. With this the idea is to introduce multicasting to see if it was possible to replace the default Head End Replication (HER), which is in use by BGP-EVPN, with flooding of the multicast. FRRouting and ifupdown2 are both capable of supporting multicasting, and it was possible to introduce PIM to the hypervisors. An issue appeared with NGINX as the Kernel in it did not support the use of multicast in VXLAN -interfaces and would lose the entire interface if configured to use multicast. However, FRRouting did manage to form PIM -neighbours between the spines and the hypervisors.

Figure 7 shows an example taken out of FRRouting configuration file, where PIM has been introduced. Since the interfaces had been configured in `/etc/network/interfaces` -file there was no need for introducing more than the simple **IP PIM** -command into each interface in use by BGP-EVPN – this includes the loopback address. In the `/etc/network/interfaces` -file, the change was adding of multicast address for the VNI to use. However, due to FRRouting supporting PIM-sparse mode only, using bridges with static IP stopped working.

The Rendezvous point (RP), had been installed onto Spine01 as a separate loopback, with Spine02 having a phantom RP and the addresses had been shared to the Leaf switches.

```

frr version 7.3.1
frr defaults traditional
hostname pve01
log syslog informational
ip pim rp 10.0.0.0 239.0.0.0/8
service integrated-vtysh-config
!
interface ens3
ip pim
!
interface ens4
ip pim
!
interface ipmr-lo
ip pim
!
interface lo
ip pim
!
router bgp 65351
bgp router-id 10.0.2.11
no bgp default ipv4-unicast
neighbor SPINE peer-group
neighbor SPINE remote-as 65350
neighbor SPINE capability extended-nexthop
neighbor 172.16.11.1 peer-group SPINE
neighbor 172.16.21.1 peer-group SPINE
!
address-family ipv4 unicast
network 10.0.2.11/32
neighbor SPINE activate
neighbor SPINE allowas-in
exit-address-family
!
address-family l2vpn evpn
neighbor SPINE activate
neighbor SPINE allowas-in
advertise-all-vni
exit-address-family
!
line vty

```

Figure 7 Iteration 4 with PIM configuration in FRRouting

5.5 Iteration five

With this iteration the idea was ensuring more of the Catalyst 2960 -switches could be utilized to create a storage network, which gains the benefit of ECMP and/or higher connection speed, with the use of OSPF or extending a VLAN to the switch with Spines acting as VTEP to introduce the use of L2 connection between Proxmox and storages. Using a third switch with the same base image as a spine switches was done to imitate a Catalyst -switch. Additional changes of removing NGINX and fourth Proxmox -node necessitated by re-installation of entire environment. (Figure 8.)

In addition to testing separate switch, further testing into creating a VXLAN -hub that allows access to a specific VNI through VLAN access was tested.

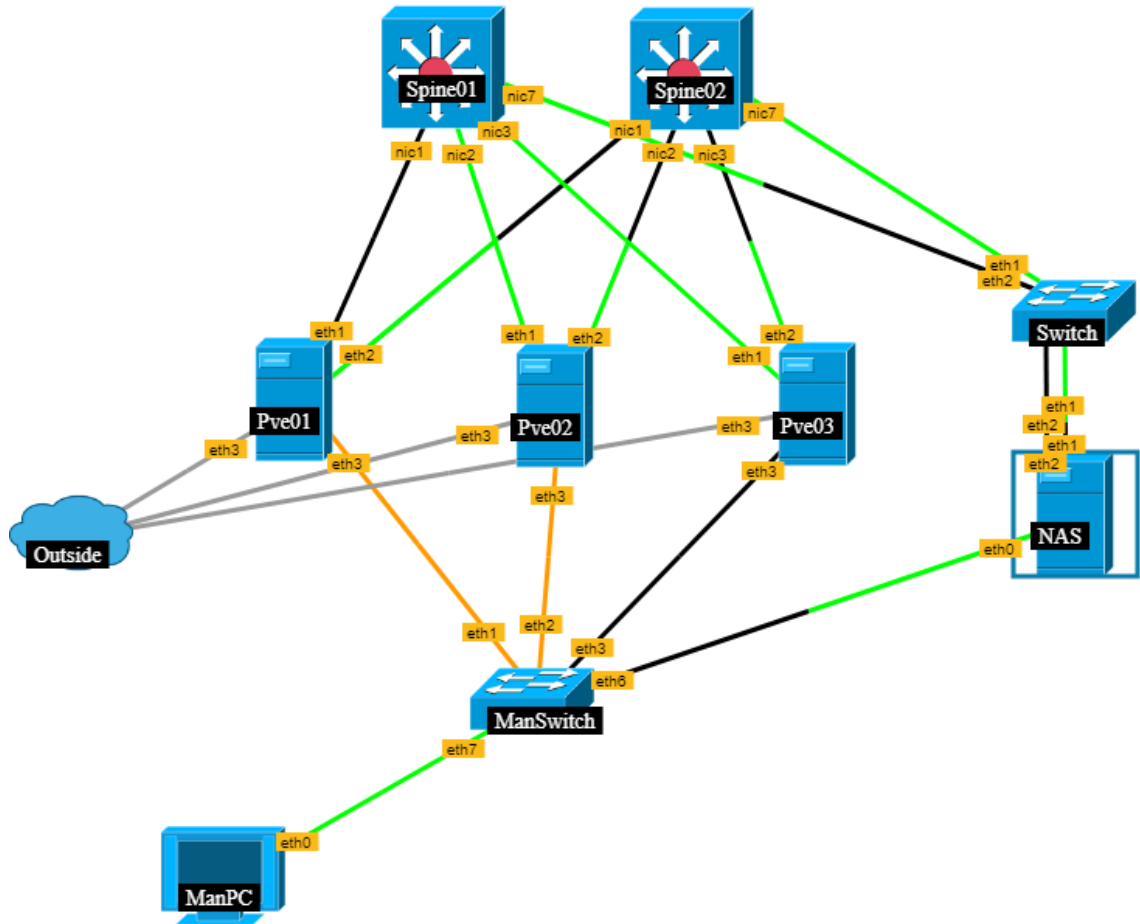


Figure 8 Iteration five with a switch attached to spine using OSPF/Trunk links

5.5.1 VLAN to VTEP

Since Arista binds VNIs to VLANs, the idea was to spread the VLAN through using two trunks towards the third switch to allow extension of the VLAN. This solution functions and provides connection to the NAS, however, it also introduces STP into the equation. In multiple occasions the problem was not the function of the connections, but the introduction of duplication of MAC -addresses which resulted in the spine blacklisting them. Due to the disjointed nature of the iteration, the idea is serviceable with introduction of SVI and joining the spines, if suboptimal due to introduction and use of STP necessitated by the L2 connections.

5.5.2 OSPF

Introducing a third process in the form of OSPF into the spines was not implausible, and the isolated network would not cause notable increase in the size of the routing table. Problems arose with the virtual environment causing connection of Spine01 to the third switch became overwhelmed by OSPF advertisements that only seized once the ports were changed. However, this did not prevent the introduction of redistribution of addresses from BGP to OSPF and vice versa. The Administrative Distance was set to favour BGP routes to eliminate issues relating to routing table entries and suboptimal routing.

5.5.3 VXLAN -hub bridge

In appendix 2 there is an example of how multiple VXLAN -interfaces can be attached to a single bridge that is VLAN aware, with isolation performed by VLAN access on each VXLAN -interface. This way the hub -bridge is capable of understanding VLANs assigned to the VM's or the container's network interfaces, and the use of VLAN access ensures that only certain VLANs can cross through said connections. Even with the hub connecting all VLANs, the isolation is strong enough with the VLAN at the end that changing the VLANs of two separate containers previously attached by same VLAN id and network, become unable to reach one another.

5.6 Iteration six

The idea of this iteration was to test forming L2 connection from the spines, using the spines as an SVI for the containers to reach a third switch not participating in STP through trunk links. Due to no access to Meraki MX250 for testing purposes in the environment, this was done to try and imitate the possible behaviour of the device and the environment. A link between the spines was introduced to create a VRRP SVI into a VLAN that the containers are connected to. Additional alteration was FRRouting update to the version 7.3.1. (Figure 9.)

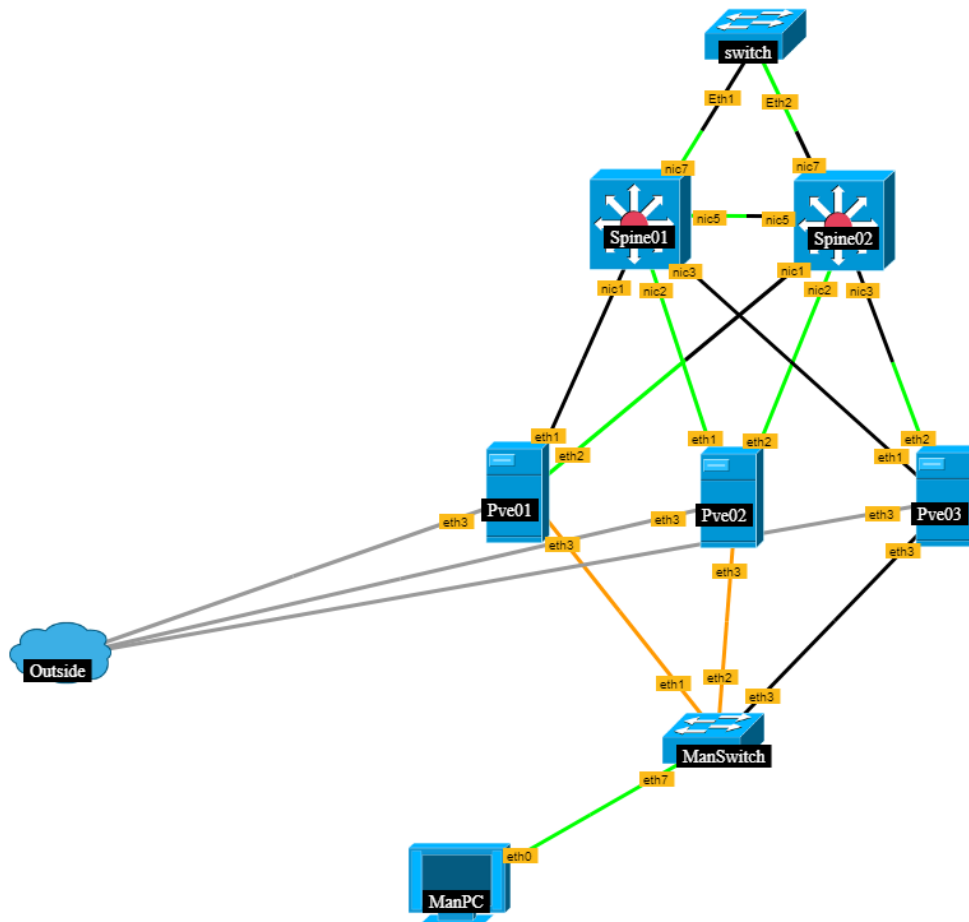


Figure 9 Iteration six: testing SVI and L2 default gateway

5.6.1 SVI in spine

Because the STP is included, defining which spine acts as **root primary** and root secondary was necessary. With the SVI existing on both spines, the connection can recover from failure of links towards the third switch or towards the gateway that is on the VLAN -interface with the use of VRRP. In the example network 192.168.60.0/24 was used to test this. If the VLAN is extended to the third switch through trunking, even without the use of STP, there are occasions where duplication becomes an issue. However, this could be caused by the virtualized environment as the duplication seems to occur depending on how long the instance has been running. With VLAN 60 SVI using VRRP introduced to the spines and VLAN 60 address 192.168.60.200, as **no auto state** address at the third switch, produces duplication. Removing and reinstating this configuration seems to have varying effects.

However, with Multiple Spanning Tree Protocol (MSTP) it could be possible to try and optimize the use of the links by directing a VLAN root to different spines.

5.7 Iteration seven

With testing regarding the spine connections to the imitation policy device resulting in inconclusive results, additional testing was performed with this iteration. However, the intention of this iteration is to experiment with FRRouting version 7.5 by reintroducing the Debian buster 10 to the installation, with the goal of testing if routing of the network outside could be done through this separate edge device. The aim is to see, if NGINX with FRRouting could it be utilized as a router as well. This necessitated changes to topology by disjoining the spines and altering the connection to be only through the new FRRouter to the imitation device (Figure 10). PVE02 was focus of the testing, while PVE01 was disconnected and PVE03 configurations were not touched.

In Pve02 the configurations have not altered, even with the introduction of the FRRouter in different ASN of **35349**, this seems to be largely due to the Cumulus Linux's use of **advertise-all-vni**, which after the spines had been given peering with the FRRouter was able to see VNIs of PVE02. No need to import additional route-tables or change configuration. As no changes were required, the FRRouting version on PVE -nodes is 7.3.1.

In the iteration experimentation with VRF was commenced, as it could provide an avenue to routing outside. What it also required was figuring out how to implement VRF into the network design and how to make it work on Proxmox. The configuration of interfaces and introduction of VRF would allow different type of isolation, and possibly ensure that the traffic would reach outside the network.

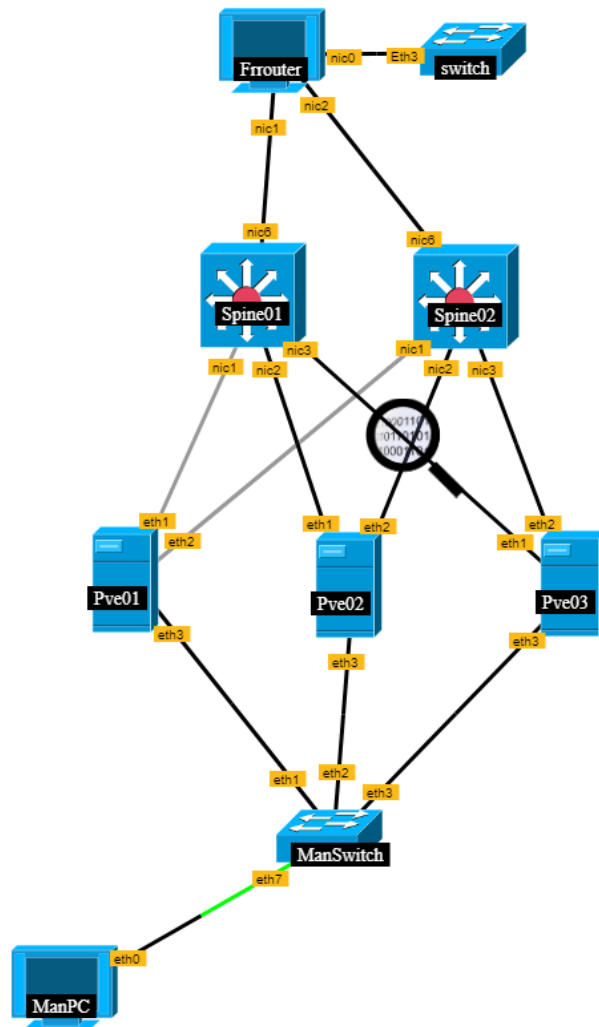


Figure 10 Iteration seven: Frrouting as edge router

5.7.1 Routing without VRF

Testing to see if the FRRouter could be used as a VTEP for L2 traffic, redirecting its traffic out a routed port towards the policy device. Having traffic travel to the router as L2 would simplify the configuration and provide a means to bring the various VXLANs to a single point that either routes or uses them for NGINX. However, despite various attempts the traffic would end up halting at the FRRouter, the traffic did manage to go towards the Switch and return but was not forwarded back to the VM at PVE02. This could have been caused by the FRRouters configurations in the end, the possibility of using a trunk towards the policy device was not tested. As the L2 approach did not seem to work – testing with VRF began.

5.7.2 VRF routing

With the L2 approach not working, turning to use VRF was necessary to ensure that traffic could be directed outside the data centre. However, with this came several problems ranging from difficulties of altering Proxmox to accommodate VRF, and with Debian buster 10 having issues related to Zebra. Zebra would cease to function if configuration of interfaces had syntax errors.

In Proxmox, the introduction of VRF -tables was simple, but trying to bind an interface to said table was causing issues. A VLAN-aware -interface would not become part of the VRF -table created, which meant that no VLAN tagging could be done. However, introducing VRF to FRRouting required only defining the VNI of interconnected VXLAN, which is an additional bridge that is not directly used by any of the VMs or the containers. (Appendix 2).

In FRRouter introducing the VRF -table, designated RED, seemed to cause conflict with Zebra if the table was introduced at the start of the `/etc/network/interfaces` -file. This was resolved by moving the table to the end of the file. With the FRRouting's use of the command **import vrf RED** it was easy to import the routing table entries of the separate VRF into the default routing -table. In figure 11 there is a showcase of the FRRouter's `frr.conf` -file, which displays how VRF is introduced into the system. With the original BGP -processes taking care of most of the connections and connectivity, there is not much configuration needed inside the VRF specific BGP -process.

```

frr version 7.5
frr defaults traditional
hostname FRRouter
log syslog informational
no ipv6 forwarding
service integrated-vtysh-config
!
ip route 0.0.0.0/0 192.168.0.1 sidelink1
!
vrf RED
 vni 3535
 exit-vrf
!
router bgp 65349
 bgp router-id 10.0.2.100
 no bgp ebgp-requires-policy
 neighbor SPINE peer-group
 neighbor SPINE remote-as 65350
 neighbor SPINE ebgp-multihop 255
 neighbor SPINE capability extended-nexthop
 neighbor 172.16.1.1 peer-group SPINE
 neighbor 172.16.2.1 peer-group SPINE
!
 address-family ipv4 unicast
  network 10.0.2.100/32
  import vrf RED
 exit-address-family
!
 address-family l2vpn evpn
  neighbor SPINE activate
  advertise-all-vni
 exit-address-family
!
router bgp 65349 vrf RED
 bgp router-id 10.0.2.100
!
 address-family ipv4 unicast
  redistribute static
 exit-address-family
!
 address-family l2vpn evpn
  advertise ipv4 unicast
  default-originate ipv4
 exit-address-family

```

Figure 11 FRR -configuration of FRRouter.

5.7.3 Testing the VRF

With the VRF configured in PVE02 and FRRouter, a link between FRRouter and the switch acting as an imitation policy device was setup to redirect traffic of the network 192.168.60.0/24 back to the FRRouter. This would allow emulating traffic directed outside of the network. Loopback address, with the address 192.168.10.1/32, was setup for testing traffic over the link 192.168.0.0/31 setup between FRRouter and Switch. Finally, a testing container on PVE02 connected to a VRF RED was made to send traffic toward the loopback address (Figure 12). The traffic was successfully routed between the outside loopback and container.

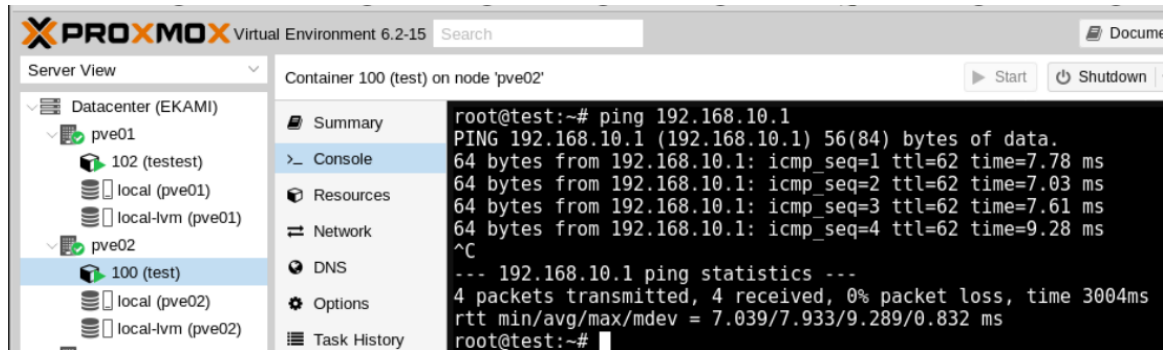


Figure 12 Test container pinging outside Loopback successfully

Observing one of the two links from a spine to FRRRouter, revealed that replies are travelling back inside VNI 3535, which is the VNI of VRF RED. Figure 13 shows the PCAP excerpt from Wireshark.

No.	Time	Source	Destination	Protocol	Length	Info
2602.	147072.59225..	192.168.10.1	192.168.60.12	ICMP	148	Echo (ping) reply id=0x0317, seq=1
2602.	147073.59467..	192.168.10.1	192.168.60.12	ICMP	148	Echo (ping) reply id=0x0317, seq=2
2602.	147074.59612..	192.168.10.1	192.168.60.12	ICMP	148	Echo (ping) reply id=0x0317, seq=3
2602.	147075.59756..	192.168.10.1	192.168.60.12	ICMP	148	Echo (ping) reply id=0x0317, seq=4
2602.	147076.59968..	192.168.10.1	192.168.60.12	ICMP	148	Echo (ping) reply id=0x0317, seq=5
2602.	147077.60915..	192.168.10.1	192.168.60.12	ICMP	148	Echo (ping) reply id=0x0317, seq=6
2602.	147078.60510..	192.168.10.1	192.168.60.12	ICMP	148	Echo (ping) reply id=0x0317, seq=7
2602.	147079.60878..	192.168.10.1	192.168.60.12	ICMP	148	Echo (ping) reply id=0x0317, seq=8
2602.	147080.60802..	192.168.10.1	192.168.60.12	ICMP	148	Echo (ping) reply id=0x0317, seq=9

▶	Frame 260265:	148 bytes on wire (1184 bits), 148 bytes captured (1184 bits) on interface 0
▶	Ethernet II, Src:	00:fd:ca:2d:2d:11 (00:fd:ca:2d:2d:11), Dst: HuthElek_db:0a:e3 (00:22:ba:db:0a:e3)
▶	Internet Protocol Version 4, Src:	10.0.2.100, Dst: 10.0.2.12
▼	User Datagram Protocol, Src Port:	35112, Dst Port: 4789
	Source Port:	35112
	Destination Port:	4789
	Length:	114
	Checksum:	0x3695 [unverified]
	[Checksum Status:	Unverified]
	[Stream index:	94]
▼	Virtual eXtensible Local Area Network	
▶	Flags:	0x0800, VXLAN Network ID (VNI)
	Group Policy ID:	0
	VXLAN Network Identifier (VNI):	3535
	Reserved:	0
▶	Ethernet II, Src:	ee:7d:8d:c9:3e:e4 (ee:7d:8d:c9:3e:e4), Dst: f6:76:8a:b4:f8:c2 (f6:76:8a:b4:f8:c2)
▶	Internet Protocol Version 4, Src:	192.168.10.1, Dst: 192.168.60.12
▶	Internet Control Message Protocol	

Figure 13 PCAP of ICMP -reply between 192.168.60.12 and 192.168.10.1

This shows the container can reach a network outside the data centre through the FRRRouter.

5.8 Iteration eight

With VRF implemented on a separate edge device using FRRouting, and with the purely L2 approach causing issues, if the spines act as the edge, another experiment was needed to see how and if VRF could be implemented between Proxmox running FRRouting and Arista vEOS 4.23.0F. While the FRRouter was

not removed, as it could be used to test if the containers could reach a separate container through the imitation Switch, the spines were joined again and provided trunk links to the Switch. PVE03 would host this new VRF, separate from prior iteration, designated VRF BLUE. (Figure 14).

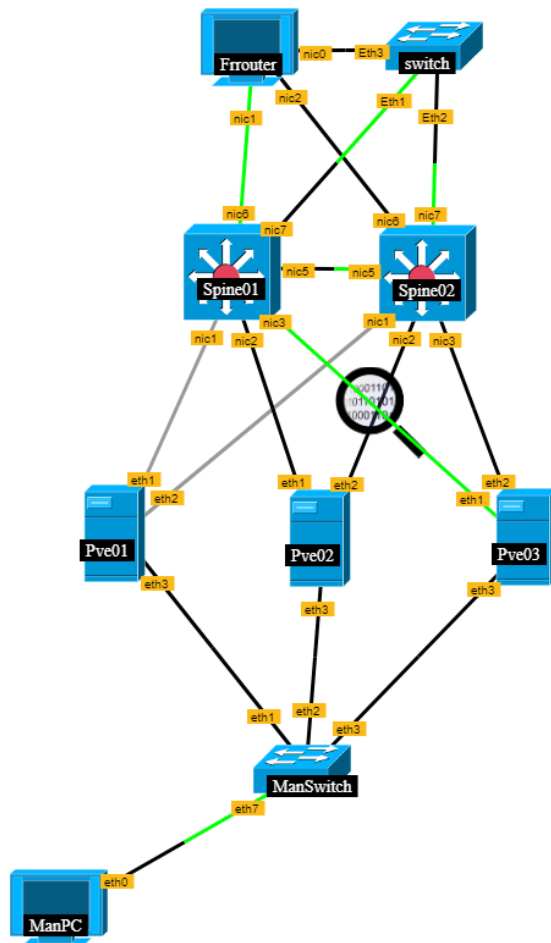


Figure 14 Iteration eight, VRF through spines.

5.8.1 FRRouting to Arista VRF

One of the challenges that the design had and would continue to encounter into was the different routing software in Proxmox and the spines. FRRouting and Arista BGP-EVPN configuration is relatively similar, however, certain aspects like leaking routes from a VRF -table to another have very different ways of to done. While FRRouting uses **import vrf** -command to achieve the leaking, with redistribution of static or dynamic default route in VRF -process, Arista's approach is a bit more complex.

```

ip routing
ip routing vrf BLUE
!
ip prefix-list BLUE seq 10 permit 192.168.160.0/24
ip prefix-list default seq 10 permit 0.0.0.0/0
!
ip route 0.0.0.0/0 Vlan10 192.168.110.100
!
route-map BLUE permit 10
  match ip address prefix-list BLUE
!
route-map DEFAULT permit 10
  match ip address prefix-list default
!
route-map default permit 10
  match ip address prefix-list default
!
router bgp 65350
  router-id 10.0.2.1
  distance bgp 10 10 10
  neighbor EDGE peer group
  neighbor EDGE remote-as 65349
  no neighbor EDGE allowas-in
  neighbor EDGE ebgp-multihop
  neighbor EDGE send-community extended
  neighbor EDGE maximum-routes 12000
  neighbor LEAF peer group
  neighbor LEAF remote-as 65351
  neighbor LEAF allowas-in 3
  neighbor LEAF ebgp-multihop
  neighbor LEAF send-community extended
  neighbor LEAF maximum-routes 12000
  neighbor 172.16.1.2 peer group EDGE
  neighbor 172.16.11.2 peer group LEAF
  neighbor 172.16.12.2 peer group LEAF
  neighbor 172.16.13.2 peer group LEAF
  !
  address-family evpn
    neighbor EDGE activate
    neighbor LEAF activate
  !
  address-family ipv4
    neighbor EDGE activate
    neighbor LEAF activate
    network 10.0.2.1/32
  !
  vrf BLUE
    rd 10.0.2.1:1600
    route-target import evpn 65351:35351
    route-target export evpn 65351:35351
    router-id 10.0.2.1
    neighbor 10.0.2.13 maximum-routes 12000
    network 0.0.0.0/0
    redistribute static
  !
router general
  vrf default
    leak routes source-vrf BLUE subscribe-policy BLUE
  !
  vrf BLUE
    leak routes source-vrf default subscribe-policy DEFAULT

```

Figure 15 Arista configuration for VRF

In figure 15 is an excerpt of Arista's configuration on one of the spines, where VRF BLUE had been implemented, which highlights some of the differences such as the leaking routes from VRF to another. Rather than importing them directly, there is a separate configuration line **router general** and under it the VRFs default and BLUE have defined leaking behaviour based on route-maps. These

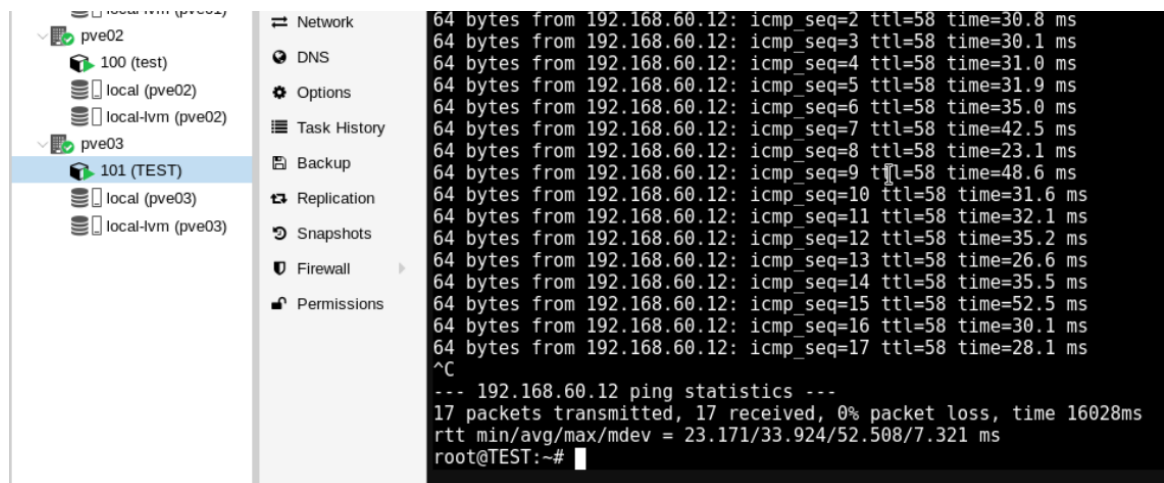
route-maps match an IP address to a prefix-list that has been created for them, imposing what IP addresses will be leaked from the VRF.

In regard to PVE03, only doing the same as with PVE02's VRF RED is needed, with a different VNI for interconnection which in this case was **35351**. This ensured the two VRFs are separate and VRF RED will not learn routes of VRF BLUE.

5.8.2 Testing connection

With introduction of VRF BLUE, the container on PVE03 was given a different network entirely. Using 192.168.160.0/24 for BLUE meant that there would be two distinctly different networks to test connectivity. In addition, a separate VLAN was made on the spines towards the imitation Switch, VLAN 10, which was directed as the default route for VRF BLUE.

Initially, while the routes had been leaked correctly between PVE03 and spines, the traffic only managed to reach the loopback address 192.168.10.1 and return to the spines. There the spines seemed to drop the traffic completely, resulting in no replies reaching the testing container. What appeared to be a partially responsible for this was the mac-address assigned to the container, however, even after changing the mac-address only a single reply would reach the container.



The screenshot shows a network management interface on the left with a tree view of containers. Under 'pve03', the container '101 (TEST)' is selected. A central sidebar lists various network-related settings like Network, DNS, Options, Task History, Backup, Replication, Snapshots, Firewall, and Permissions. On the right, a terminal window displays the output of a ping command from the 'root@TEST' shell. The output shows 17 successful ICMP echo requests from 192.168.60.12 to 192.168.60.12, each receiving a 64-byte reply with varying TTL and RTT values. A summary line indicates 17 packets transmitted, 17 received, 0% packet loss, and a total time of 16028ms. The RTT statistics are: min/avg/max/mdev = 23.171/33.924/52.508/7.321 ms.

```

64 bytes from 192.168.60.12: icmp_seq=2 ttl=58 time=30.8 ms
64 bytes from 192.168.60.12: icmp_seq=3 ttl=58 time=30.1 ms
64 bytes from 192.168.60.12: icmp_seq=4 ttl=58 time=31.0 ms
64 bytes from 192.168.60.12: icmp_seq=5 ttl=58 time=31.9 ms
64 bytes from 192.168.60.12: icmp_seq=6 ttl=58 time=35.0 ms
64 bytes from 192.168.60.12: icmp_seq=7 ttl=58 time=42.5 ms
64 bytes from 192.168.60.12: icmp_seq=8 ttl=58 time=23.1 ms
64 bytes from 192.168.60.12: icmp_seq=9 ttl=58 time=48.6 ms
64 bytes from 192.168.60.12: icmp_seq=10 ttl=58 time=31.6 ms
64 bytes from 192.168.60.12: icmp_seq=11 ttl=58 time=32.1 ms
64 bytes from 192.168.60.12: icmp_seq=12 ttl=58 time=35.2 ms
64 bytes from 192.168.60.12: icmp_seq=13 ttl=58 time=26.6 ms
64 bytes from 192.168.60.12: icmp_seq=14 ttl=58 time=35.5 ms
64 bytes from 192.168.60.12: icmp_seq=15 ttl=58 time=52.5 ms
64 bytes from 192.168.60.12: icmp_seq=16 ttl=58 time=30.1 ms
64 bytes from 192.168.60.12: icmp_seq=17 ttl=58 time=28.1 ms
^C
--- 192.168.60.12 ping statistics ---
17 packets transmitted, 17 received, 0% packet loss, time 16028ms
rtt min/avg/max/mdev = 23.171/33.924/52.508/7.321 ms
root@TEST:~#

```

Figure 16 Successful ping between containers

The problem appeared to be related to vEOS 4.23.0F, since after changing it to 64-bit vEOS 4.24.0F it disappeared. With traffic going back and forth from the loopback to the container, testing to reach the different containers could be done (Figure 16). In figure 17 there is the PCAP of traffic between the FRRRouter and the Switch, showcasing replies and requests going through.

3299...	160343.81749...	192.168.60.12	192.168.160.13	ICMP	98 Echo (ping) reply	id=0x0311, seq
3299...	160344.81489...	192.168.160.13	192.168.60.12	ICMP	98 Echo (ping) request	id=0x0311, seq
3299...	160344.81885...	192.168.60.12	192.168.160.13	ICMP	98 Echo (ping) reply	id=0x0311, seq
3299...	160344.91242...	fe80::2fd:caff:fe2d...	ff02::2	ICMPv6	70 Router Solicitation	from 00:fd:ca:2
3299...	160345.82204...	192.168.160.13	192.168.60.12	ICMP	98 Echo (ping) request	id=0x0311, seq
3299...	160345.82805...	192.168.60.12	192.168.160.13	ICMP	98 Echo (ping) reply	id=0x0311, seq
3299...	160346.57498...	192.168.0.2	224.0.0.251	MDNS	87 Standard query	0x0000 PTR ipps. to
→ 3299...	160346.81875...	192.168.160.13	192.168.60.12	ICMP	98 Echo (ping) request	id=0x0311, seq
← 3299...	160346.83360...	192.168.60.12	192.168.160.13	ICMP	98 Echo (ping) reply	id=0x0311, seq

▶ Frame 329927: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
 ▶ Ethernet II, Src: 00:41:89:da:24:18 (00:41:89:da:24:18), Dst: 00:fd:ca:2d:2d:10 (00:fd:ca:2d:2d:10)
 ▶ Internet Protocol Version 4, Src: 192.168.160.13, Dst: 192.168.60.12
 ▶ Internet Control Message Protocol

Figure 17 Traffic from link between FRRRouter and Switch

6 RESULTS

Based on the data and the research produced by these iterations, it was possible to come up with a design which would allow the use of FRRouting in Proxmox to provide BGP-EVPN capabilities with which to connect the nodes directly to a spine and thus allowing the use of spine-and-leaf architecture. In addition to this another document, based on data gathered from implementing the solutions, detailing the process of introducing FRRouting and ifupdown2 to Proxmox.

6.1 Data centre -design

In Appendix 1 the design has been described in which Proxmox nodes connect directly into the spines, and host FRRouting to achieve BGP-EVPN capability, allowing the use of VXLAN to provide cross node communications and with the use of VRF the connection outside. Since the purely L2 approach, where VRF would not be needed, resulting in blacklisting of mac-addresses and excessive traffic that would drain large amount of the networks resources, not being viable – the use of VRF to achieve connections directed outside of the network has been introduced.

6.1.1 MX250 connection to spines

Since the MX250 is not capable of participating in BGP-EVPN and does not support aggregated links, the spines would be the point of routing by utilizing L2. Two VLANs would be reserved to create a contact surface between the MX250 and the spines, through which the traffic brought in and leaked from VRFs would allow routing towards the policy device. This necessitated the joining of the spines through a redundant trunk link. The trunk link would allow setting up redundant gateways and SVIs which could be used to direct the traffic between the spines and the MX250, where the two separate VLANs would act as next hops for the networks hosted by the data centre.

This will, however, require traffic engineering and it is possible to simply drop the number of VLANs to a singular VLAN taking care of the connection between the MX250 and the spines as the default route.

With this comes a clear caveat in routing capabilities, especially noted during the research, since the use of STP and static routing is required. If the connection between the STP root and the node is cut, in this case being SPINE01, rather than the traffic being directed through SPINE02, the traffic is likely to be bounced through one of the other nodes to SPINE02. This kind of sub-optimal routing would necessitate the introduction of joining the two processes in the spines with OSPF or iBGP for more optimal routing.

It is also important to note that large portion of the increase in traffic was produced by introduction of importing a VLAN to the spines that had **allow-as in** which caused all traffic to be sent to the bridges connected to the VNI. While removing **allow-as in** did remove this excess of traffic, it also ensured that traffic could not be bounced through the Proxmox nodes to the other spine in the same ASN.

6.1.2 Use of eBGP

While the option of using separate ASNs for each node would be viable, the choice of using a single ASN to create a single plane for the nodes would allow far simpler connection of new nodes. This is would be due to the nature of having to import and export routes from the spines to the nodes. If each node had a separate ASN that would mean each newly introduced node would have to either import routes from the spines which has so far not proven to function during testing. With each node in different ASN, each new node would have to be introduced to the processes of both the spines by of importing and exporting routes to the new ASN.

With a single ASN for the nodes, this can be avoided, resulting in less of a need to introduce more than the neighbour link to the process. However, if a physical NGINX/FRRouter is introduced, it should be given a separate ASN as was done during testing. Due to the device being singular and utilizing the same routing software as the Proxmox -nodes, the **advertise-all-vni** -command will to be able to import and export routes of separate ASNs without additional configuration. This is because of the FRRouting using only the VNI, rather than just the whole **ASN:VNI** -type of importing and exporting of routes with eBGP. (Cumulus).

6.1.3 Virtual routing and forwarding

Necessity of the VRF for having a functional solution requires the spines to be able to support VRF. In case this is not possible or would increase costs, through a licence for example, there is the possibility of building an FRRouter. For the initial installation it brings additional complexity, since depending on the spine - switches there can be differences in configuring FRRouting compatible VRF leaking. However, as proven by using Arista as the spine's routing software that acts as the VRF -process endpoint it is possible to use different routing software for the design. With the differences in syntax different routing software, it is going to require some initial investigation and research to make a functional solution.

The use of FRRouting as the edge device to take care of VRF -routing has been shown to work, however, during testing there were certain complications. With the laboratory up for extended periods of time, while active but not actively routing, the router running FRRouting on Debian buster 10 would periodically need to be rebooted to start routing traffic again. This could be caused by either the virtualized environment and lack of resources or possibly by lack of optimization regarding the build based on Debian buster 10. Yet it is a concern that needs to be considered if applied to real-world.

6.1.4 Cluster communication and Management

Due to the sensitivity of the cluster communication, shown in testing to be unviable if done through the spines, the use of two independent switches would likely prove to be most effective for cluster communication and to provide management access to the cluster. With a management switch, it is also possible to attach other devices of the data centre network for remote access. Using catalyst 2960 -switches for this task is intended, as it would allow repurposing some of the old equipment.

Though the cluster communication could not be done in the laboratory environment through the spines using VXLAN in an effective manner, it was proven to be possible. Because the virtualization of each connection, each link, could cause enough delay in traffic to cause the cluster communication to drop or lose other nodes constantly. However, the secure approach would be using separate networks that would not be susceptible to fluctuation of traffic.

6.1.5 Storage solutions

Since there exists a single storage which would be connected to two nodes through use of SAS, it can be easily shared in Proxmox. In this case, it is likely to cause more strain of traffic for these two nodes as other nodes will access the storage shared across the cluster. The addition of the second storage device would necessitate providing a different storage solution as the device has only 4

RJ45 ethernet -ports. Ideally, to allow each node access, either a NAS or SAN solution should be used.

For SAN solution, attaching each node with a single cable to a Catalyst -switch would provide each node 1 Gbe connection to the 4 Gbe connected storage, allowing the storage to be accessed by each node. Easiest to implement, yet perhaps not the most effective. With introduction of NAS solution, there are two possible approaches to consider based on testing.

Using a Catalyst -switch, connected to the spines, could get the benefit of the more powerful upstream ports, but introduces the need for use of OSPF. With OSPF, the traffic would need to go through the underlay. Depending on the Catalyst 2960 -model it would have two 1 Gbe or two 10 Gbe upstream connections to the spines.

One option would be introducing VXLAN -interfaces to the nodes with a static IP address on a bridge to act as a L2 connection to the storage that would be connected directly to the spines. With this, the connection could be either on both of the spines or a single spine. While functional, the solution would take up ports of the spine for a single storage which would not be ideal use of connections in a long run. In addition to this, there is the risk of causing duplicated traffic, if no care is taken when implementing it – so this should be considered the last resort approach.

6.1.6 Choice of switch

Because of the need to utilizing BGP-EVPN, which requires the ability to utilize multiple routing protocols simultaneously, it is necessary that care is taken in choosing the switch. Cumulus, Arista, Cisco and Juniper, including other manufacturers, provide switches intended for data centre use. With the costs of suitable switches that support BGP-EVPN, as well as support modular use of Small Form Pluggable (SFP) -devices tend to be high. However, even a single

spine -switch could be made to service the data centre for long, as the capacity and speed of a dedicated data centre switch is plentiful for a small data centre.

While using Cumulus Linux based switches would allow the use of IP unnumbered BGP, which would eliminate the need for reserving addresses for links between the devices by utilizing loopback as the link address, the choice of device was limited to Cisco based due to the clients wishes to utilize Cisco - devices. (Cumulus.).

Two devices were presented, based on the client desire for Cisco device, chosen with idea of modularity, port speed and BGP-EVPN support that is capable of VTEP. Nexus 93180 yc-FX-24 and Nexus 31108PC-V. SFP -ports would allow connection of several types of modules for fibre connections and module ready copper-cables. However, it is also necessary to consider the licencing costs of capabilities needed for the data centre (Cisco 2020a, Cisco 2020b.).

6.2 Installation guide for Proxmox

Because of the relative scattered nature of information for introducing FRRouting to Proxmox, and configuration of it as a functional participant of BGP-EVPN, a great amount of effort was required in gathering information on how to go through the process. To alleviate this task for the future, collecting the information into single document was done as a part of the research. (Appendix 2.)

6.2.1 Proxmox installation

The guide goes through the steps of installing Proxmox, as well as how to alter the repositories of the unsubscribed version, to make the installation of FRRouting successful. This was because by default Proxmox comes installed with subscribed repository, which causes issues when trying to install applications like ifupdown2 from said repository. It does not conflict with the installation of FRRouting, as this is done from a separate repository. However, the steps presented for the FRRouting installation should be used when installing FRRouting onto other Debian GNU/Linux devices, as it is a reliable way of

installing the latest and stable version. This was noticed during installation of Debian buster 10, which could install FRRouting without additional repository updates, once installed, but only at version 6 at the time of testing.

6.2.2 Configuration of interfaces

In Appendix 2 there is detailed steps to configuring the interfaces used for VXLAN, BGP-EVPN and VRF. Because of the nature of each interface type being very dependent on their configuration, it was necessary to take into notice how each differ and what limitations each type of interface will have.

The uplinks are used for BGP -process to create neighbours between the spines and the nodes. The idea was chosen because it would mean that all configuration could be placed onto a single virtual interface which could then use several interfaces, if necessary, to achieve the goal of connectivity. The bond 0 -mode was chosen because other modes could cause the links to not work properly.

With the VXLAN -interfaces the use of **bridge-arp-nd-suppress** is used to suppress ARP -flooding as much as possible. Bridge-learning is turned off on the VXLAN -interface, as BGP-EVPN will take care of the installation of remote mac-addresses'. In addition to this it was necessary to introduce the various types of bridges that could be connected to the VXLAN -interfaces and the Inter-connecting VXLAN -setup. (Cumulus).

6.2.3 Variant bridge -interfaces

Because the probable need to use VLANs, VRF and connecting the VMs and the containers in the data centre, the various configurations for bridges are needed. Bridges capable of attaching VLAN -tagged containers or VM -interfaces require the use of **bridge-vlan-aware yes** -line. Without it the virtual device will simply refuse to boot at all. With introduction of the VLAN -hub like bridge, it is possible to connect multiple different VXLANs and tie them to a VLAN to allow isolation within the node-to-node communication, in this case the bridge would allow all

VLANs to attach to it. The VXLAN -interface which allows access of a VLAN-tag would prevent wrong VLANs from communicating over it. (Figure 18.)



Figure 18 VLAN-aware bridge with VXLAN access

Use of **bridge-vids** -line would allow making the hub bridge to accept only certain types of VLANs on said bridge, which did not work too well during testing.

When it comes to use of VRF, interfaces cannot be made VLAN-aware as it will make the bridge not participate in the VRF it is intended to be connected to. This means that no VLANs can be attached to these interfaces, as the problem of devices not booting will emerge. Care must also be taken when introducing VRF into Proxmox, or into other Debian GNU/Linux based OS, as there is a possibility of the VRF -table causing FRRouting to halt the use of Zebra. Only way to recover from Zebra getting halted has so far been removal of configurations of the /etc/network/interfaces -file which cause conflict, such as poorly made bridges or VRF -table, prior to restarting the FRR -service.

It is, however, possible to simply connect a VLAN-unaware bridge directly to a VXLAN -interface, allowing the node-to-node communication to exist only through a VNI. This type of connection could be used to make separate communication paths for communication between the VMs and a container hosting services that are intended to be only accessible within the data centre. (Figure 19.)

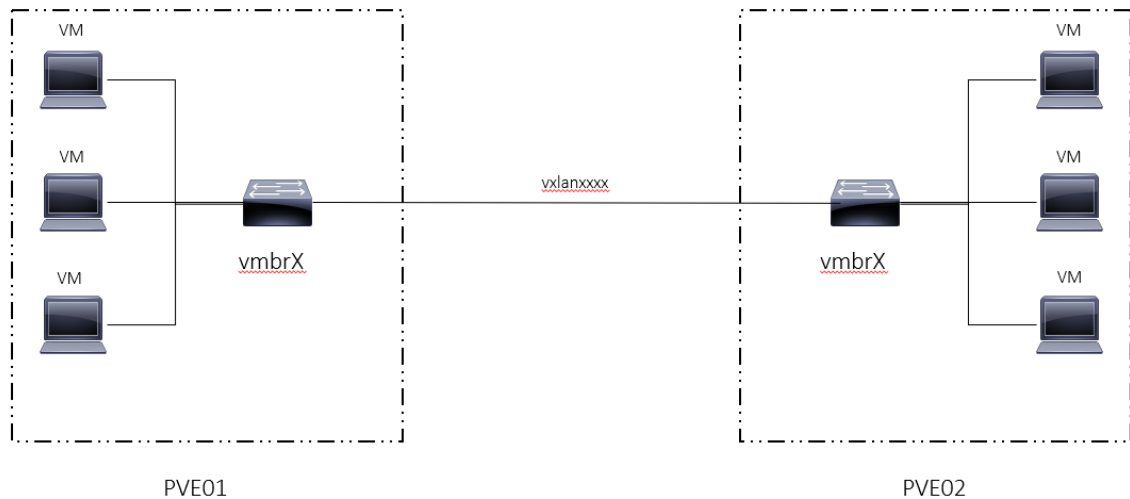


Figure 19 VBR without VLAN

6.3 Imitation of MX250

While in testing it was possible to imitate the MX250, this imitation has limitations to it. Because of this, it is difficult to speculate how the actual device will behave when connected to the two spines. However, the testing should give some reference to the possible behaviour -yet it cannot precisely predict the interactions of the data centre with the policy device. While having the separate FRRouter connected through a single routed port is likely to work, connecting two separate switches should not cause immense conflict if they are not being tied together as a single aggregated port.

6.4 Outside of scope

As no work was done on implementing techniques such as Multi-chassis Link Aggregation (MLAG) or virtual Port Channel (vPC), it is not possible to out rule the use of such techniques to improve the design. In addition, the research has been done completely in IPv4, it does not speculate on how to implement IPv6 into the design.

While the study into the design has noted that logs produced by FRRouting can cause issues with storage, further investigation into means of preventing this

outside of not using logging of neighbour states in BGP have not been investigated.

The study also has been done completely utilizing BGP, meaning it does not take into consideration the use of OSPF as a replacement for BGP to connect the BGP-EVPN processes.

7 CONCLUSION

The goal of the research was to design a data centre, using modern data centre techniques in a cost-effective manner. A design that could be used for hosting various VMs and containers to provide services that could be used as a learning environment. The final design should be able to scale with the growth of the network, as well as allow further development of the network environment to support the further virtualization of a learning space.

With the introduction of spine-and-leaf -architecture and BGP-EVPN into the data centre network, the use of modern data centre techniques has been achieved. With the choice of architecture, the possibility to scale the data centre by adding new servers has been made possible. Introduction of switches specifically designed to act as Leaf -switches can be done, allowing the creation of separate clusters to increase the volume or dedicate a cluster to a specific purpose.

In regard to cost-effectiveness, it is difficult to scope when no budget exists. With the elimination of leaf -switches from the initial design, the design can start off by introducing only a single spine -switch at the beginning. Each server presented can be introduced to the network, with or without the introduction of 10 Gbe ports. In addition, several of the Catalyst switches already present can be used in the design.

Additionally, the access to the data centre's resources can be provided through two ways. With the use of VRF, it is possible to either focus the routing onto the spines or introduce a separate edge device that could be used as a router. This means the VMs that are hosted by the data centre can be accessed through the

MX250 -firewall. In addition, the design considers the use of NGINX either in virtual or physical form, in which it could be used to host http/https -services. This would allow for more options to access the resources of the data centre while keeping the management side of the Proxmox cluster separate.

In comparison to the starting point in which there were only two servers with a single storage in use, with four servers and a single storage not in use, the design utilizes all the servers and storages. With the possibility to increase the capacity of the data centre relatively easily due to the spine-and-leaf - architecture.

8 DISCUSSION

With the conclusion of the research, there still remains the reliability of the research to be discussed, as well as the future development of the research done. Introspection and self-reflection on part of the research done.

8.1 Reliability of research

With the research being done in a virtualized laboratory environment, repeating the testing and behaviour of the design should be possible because of the documentation on Proxmox installation. Because of the differences in configuring of routing software between Arista and FRRouting, introducing a different routing software as the spine will require research into the interactions between FRRouting and the new routing software. This should not, however, prevent the formulation of neighbours or leaking of VRF -tables.

However, it is important to note that updates that come with changes from versions used -such as FRRouting including the bidirectional sparse-mode for PIM (Github 2019), could affect the results. There is a possibility that later versions of Proxmox support the turning of nodes into leaf -switches more in the future. Or it is possible that the development of the hypervisor results in changes that prevent the use of FRRouting entirely.

8.2 Further research and development

For the future of the design, research into optimization of the routing behaviour and usage of pathways must be considered. In addition, the design has room for further utilization of the Catalyst 2960 -switches to act as a connection surface for IoT -devices which could be accessed by the VMs. This would provide another use for the switches, while granting additional avenues for educational use of the environment.

One thing to consider should be for the addition of new storage and servers, especially regarding new Proxmox -nodes, as to how ease and expedite the process. Because the process is largely based on altering text files and downloading set applications, it could be possible to automate most of the process through a script that can perform most of the task on its own.

Since the research has not taken a stance on using MLAG or vPC, it should be studied as a viable option to improve the design or how it would impact the function.

Due to the relative complexity, turning implementation of the design into another thesis project should be considered.

8.3 Introspection

At the start of the research, I had very little experience in dealing with Proxmox or FRRouting. BGP was familiar to me from studying, while there had been just a glimpse into EVPN, which gave me the illusion that this task would be relatively easy to accomplish. As the research progressed, the starting point of even figuring out problems regarding Linux and repositories was new and provided me with great insight into the functions of the OS. FRRouting and working with Arista, BGP-EVPN in general, made me realize that there are numerous factors that can affect and alter the nature of the network. Because of this the experience gained from this research will be valuable to for the rest of my life.

Having produced a document that compiles the introduction of FRRouting and the various interfaces that could be used in it, provides me with a sense of pride. Scavenging for this information, finding the means to implement them to the design and figuring out problems has been quite the task. It has, however, provided me with great amount of experience about creating and inner functions of data centre networks, VRF and BGP in general.

REFERENCES

Al-Shawi, M. 2016. CCDE Study Guide. E-book. Indianapolis, Indiana: Cisco Press. Available at: <https://www.oreilly.com/> [Accessed: 19 November 2020]

Arista. No date a. Layer 2 Leaf & Spine Design and Deployment Guide. PDF document. Available at: http://allvpc.net/L2LS_Design_Guide.pdf [Accessed: 16 November 2020]

Arista. No date b. Layer 3 Leaf & Spine Design and Deployment Guide. PDF document. Available at: http://allvpc.net/Arista_L3LS_Design_Deployment_Guide.pdf [Accessed: 19 November 2020]

Bernat V. 2018. L3 routing to the hypervisor with BGP. Blog. Available at: <https://vincent.bernat.ch/en/blog/2018-l3-routing-hypervisor> [Accessed: 16 November 2020]

Cisco. No date. Cisco Data Center Spine-and-Leaf Architecture: Design Overview White Paper. WWW document. Available at: <https://www.cisco.com/c/en/us/products/collateral/switches/nexus-7000-series-switches/white-paper-c11-737022.html> [Accessed: 15 November 2020]

Cisco. 2014. Cisco Catalyst 2960-Plus Series Switches Data Sheet. WWW document. Available at: https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-2960-plus-series-switches/data_sheet_c78-728003.html [Accessed: 26 November 2020]

Cisco. 2020a. Cisco Nexus 9300-FX Series Switches Data Sheet. WWW document. Available at: <https://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-series-switches/datasheet-c78-742284.html> [Accessed: 19 November 2020]

Cisco. 2020b. Cisco Nexus 3100-V Platform Switches Data Sheet. WWW document. Available at: <https://www.cisco.com/c/en/us/products/collateral/switches/nexus-3000-series-switches/datasheet-c78-736608.html> [Accessed: 19 November 2020]

Cisco Meraki. 2020. MX Layer 2 Functionality. WWW document. Available at: https://documentation.meraki.com/MX/Networks_and_Routing/MX_Layer_2_Functionality [Accessed: 19 November 2020]

Cisco Meraki. 2020b. Configuring VLANs on the MX security Appliance. WWW document. Available at: https://documentation.meraki.com/MX/Networks_and_Routing/Configuring_VLANs_on_the_MX_Security_Appliance [Accessed: 19 November 2020]

Cisco Meraki. 2020c. Networks and Routing. WWW document. Available at: https://documentation.meraki.com/MX/Networks_and_Routing [Accessed: 26 November 2020]

Cumulus. Cumulus Linux 4.2 User Guide. PDF document. Available at: <https://docs.cumulusnetworks.com/cumulus-linux-42/cumulus-linux-42.pdf> [Accessed: 16.11.2020]

FRRouting. 2017. FRRouting Project. WWW document. Available at: <https://frrouting.org/> [Accessed 15 November 2020]

Fortien, J. 2018. Welcome to ifupdown2's documentation! WWW document. Available at: <https://ifupdown2.readthedocs.io/en/latest/> [Accessed: 16 November 2020]

Github. 2019. FRR 7.2 Release. WWW document. Available at: <https://github.com/FRRouting/frr/releases/tag/frr-7.2> [Accessed: 26 November 2020]

Kananen, J. 2012. Kehittämistutkimus opinnäytetyönä: Kehittämistutkimuksen kirjoittamisen käytännön opas. Jyväskylä: Jyväskylän ammattikorkeakoulu, p. 51-84, 161-172, 191-192.

Kananen, J. 2017. Kehittämistutkimus interventiotutkimuksen muotona: Opas opinnäytetyön ja pro gradun kirjoittajalle. E-book. Jyväskylä: Jyväskylän ammattikorkeakoulu. Available at: <https://kaakkuri.finna.fi/> [Accessed: 19.5.2020]

Kettunen, M. 2019. XAMK Virtuaalilaboratio VirtualLab. WWW document. Available at:

<https://www.ictlab.fi/index.php/fi/informaatioteknologia/hankkeet/kybervaliot/233-xamk-virtuaalilaboratorio-virtuallab> [Accessed: 19 November 2020]

Kolb, A., Kolb, D. 2011. Experiential Learning Theory: A Dynamic, Holistic Approach to Management Learning, Education and Development. PDF document. Available at:

https://www.researchgate.net/publication/267974468_Experiential_Learning_Theory_A_Dynamic_Holistic_Approach_to_Management_Learning_Education_and_Development [Accessed: 30 November 2020]

Koskinen, I., Zimmerman, J., Binder, T., Redström, J. Wensveen, S. (2011). Design Research Through Practice: from the lab, field and showroom. Waltman, MA: Morgan Kaufman, p. 51-59.

Opensource. What is virtualization? WWW document. Available at:

<https://opensource.com/resources/virtualization> [Accessed: 16 November 2020]

Proxmox. 2020. Proxmox VE. WWW document. Available at:

https://pve.proxmox.com/mediawiki/index.php?title=Main_Page&oldid=10654 [Accessed 15 November 2020]

RFC 7348. 2014. Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks. PDF

document. Available at: <https://www.rfc-editor.org/rfc/pdf/rfc7348.txt.pdf>
[Accessed: 15 November 2020]

RFC 7432. 2015. BGP MPLS-Based Ethernet VPN. PDF document. Available at:
<https://tools.ietf.org/pdf/rfc7432.pdf> [Accessed: 17 November 2020]

RFC 8584. 2019. Framework for Ethernet VPN Designated Forwarder Election
Extensibility. PDF document. Available at: <https://tools.ietf.org/pdf/rfc8584.pdf>
[Accessed: 17 November 2020]

Varnum, D. 2018. 7 guiding principles for leading data center networks. Blog.
Available at: <https://cumulusnetworks.com/blog/data-center-network-principles/>
[Accessed: 19 November 2020]

LIST OF FIGURES

Figure 1 Traditional 3-Tier architecture compared to 2-Tier Spine-Leaf architecture.....	12
Figure 2 Collapsed -leaf next to traditional spine-and-leaf.....	15
Figure 3 Virtual laboratory testing environment, first iteration.....	19
Figure 4 Excerpt of PCAP of ICMP between two containers.....	23
Figure 5 Third iteration: Introduce NGINX and storage	24
Figure 6 NGINX configuration for Proxmox WebGUI in laboratory environment .	25
Figure 7 Iteration 4 with PIM configuration in FRRouting	27
Figure 8 Iteration five with a switch attached to spine using OSPF/Trunk links...	28
Figure 9 Iteration six: testing SVI and L2 default gateway	30
Figure 10 Iteration seven: Frrouting as edge router	32
Figure 11 FRR -configuration of FRRouter.....	34
Figure 12 Test container pinging outside Loopback successfully	35
Figure 13 PCAP of ICMP -reply between 192.168.60.12 and 192.168.10.1.....	35
Figure 14 Iteration eight, VRF through spines.....	36
Figure 15 Arista configuration for VRF	37
Figure 16 Successful ping between containers	38
Figure 17 Traffic from link between FRRouter and Switch	39
Figure 18 VLAN-aware bridge with VXLAN access.....	46
Figure 19 VMBR without VLAN	47

Datacentre design document

2020



South-Eastern Finland
University of Applied Sciences

CONTENTS

1	INTRODUCTION.....	3
2	DESIGN	4
2.1	Spine-and-Leaf connection	5
2.2	Spine to Firewall.....	6
2.3	Storage.....	7
3	NETWORK SCHEME.....	10
3.1	Loopback addressing	10
3.2	Spine to node addressing.....	11
3.3	Reserved addressing	11
4	EQUIPMENT	12
4.1	Cabling	12
5	LOGICAL TOPOLOGY.....	13
5.1	Introducing NGINX with FRRouting.....	13
	Figure 1 Spine and server connection topology	4
	Figure 2 Spine to Leaf connections	5
	Figure 3 Spine to Firewall.....	6
	Figure 4 Storage solutions.....	7
	Figure 5 OSPF for storage switch	8
	Figure 6 SAS and SAN -storage.....	8
	Figure 7 Spine connected storage with VTEP and VLAN.....	9
	Figure 8 Loopback addressing	10
	Figure 9 Spine and Leaf link addressing	11
	Figure 10 Reserved addressing	11
	Figure 11 Logical topology of the network.....	13
	Figure 12 NGINX with FRRouting	14

1 INTRODUCTION

This document goes over the design of a planned Data centre environment for Ekami utilizing FRRouting to produce a “collapsed Leaf” -type of a network environment to provide VXLAN capabilities to the network and virtualization environment. The design in question has been planned and researched based on client’s requests, pre-existing equipment and suggested new equipment.

Intention with design is to create a robust, redundant and scalable network which is capable of adjusting and being developed further based on future needs.

2 DESIGN

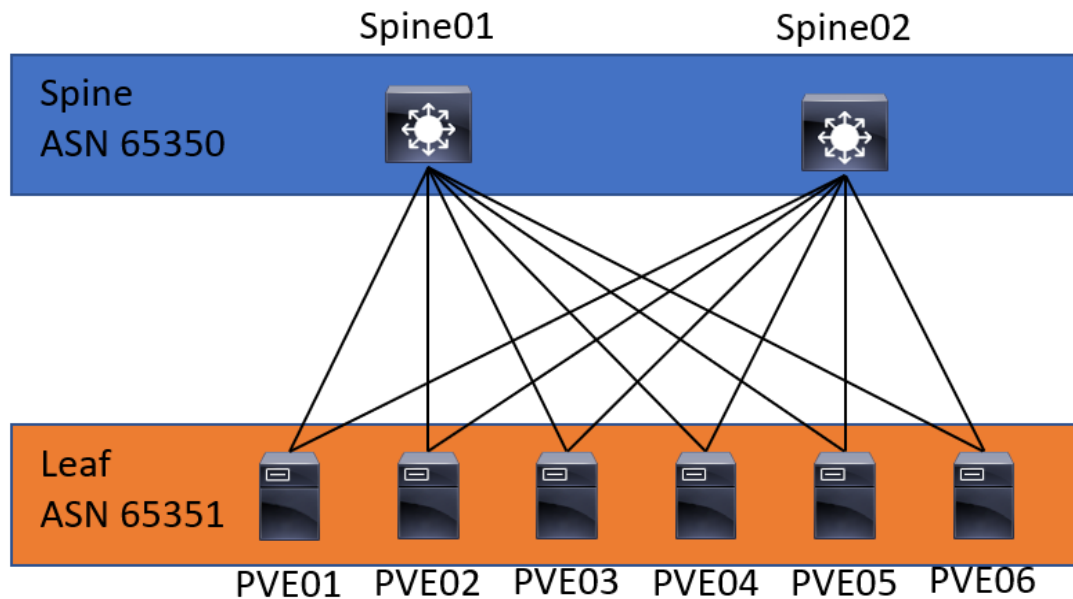


Figure 1 Spine and server connection topology

The design is based on using only two Spine -switches, which provide BGP-EVPN connection to each server. In figure 1 can be seen the logical topology regarding the devices available and planned for the network at the time of this design. Further in the document will have more detailed information regarding storage options and network design. Leaf will exist in a single ASN due to the relatively small size of the cluster, however adjusting it to have unique ASN for each node is not impossible – as the Spine will remain in a single ASN even then.

2.1 Spine-and-Leaf connection

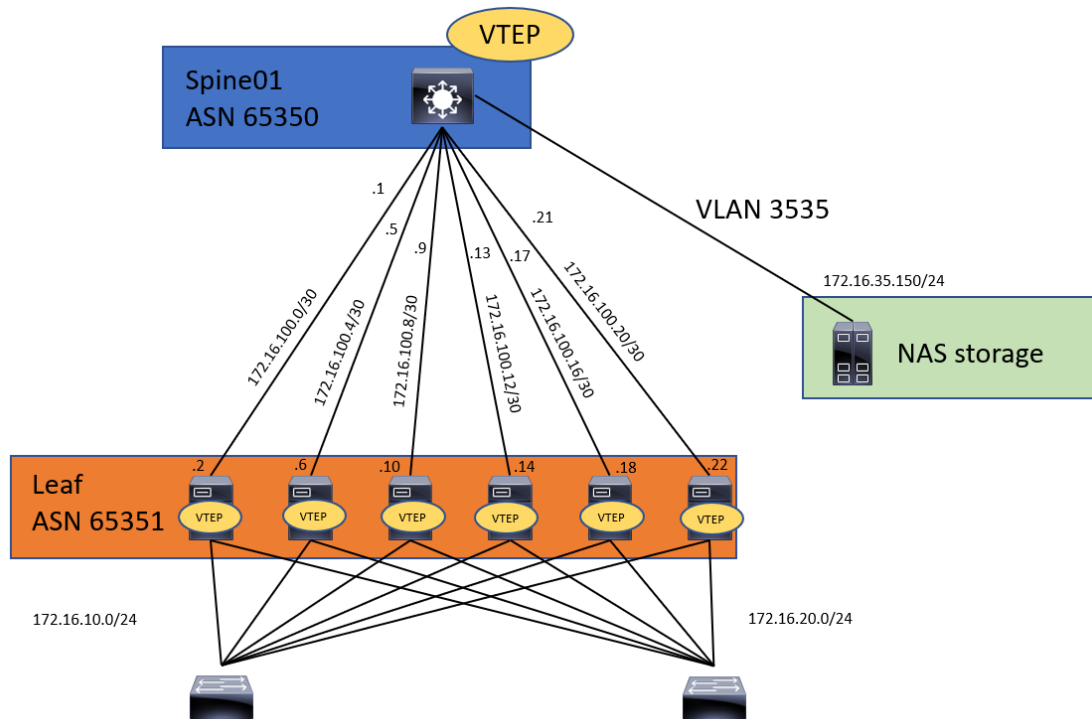


Figure 2 Spine to Leaf connections

The Spine -switch will be providing connection to each node, with two spines acting as redundancy and load balancing element. With FRRouting installed onto the Proxmox -hypervisors, they are able to take part in the communication and create their own Virtual Tunnel End Points (VTEPs) and VXLAN interfaces that allow segregation of the network traffic, providing the overlay; with BGP acting as the underlay and EVPN being used to create the overlay. Storage will be connected to the nodes using overlay, where nodes have their own network interface to connect to it, and the Spine has a VTEP that connects a specific VLAN through access ports to the nodes.

eBGP was chosen due to the relatively small size of the network, with two separate Autonomous System Numbers (ASN) for limiting the routing tables.

Two switches are also connected to the nodes, acting as management and cluster communication pathways; cluster communication is very sensitive to lag,

which means using a switch as a hub for their communication is the ideal solution, and providing a second switch for cluster communication redundancy ensures that the network will endure failure of a link or a switch. For this, the ports 1 and 2 should be reserved.

2.2 Spine to Firewall

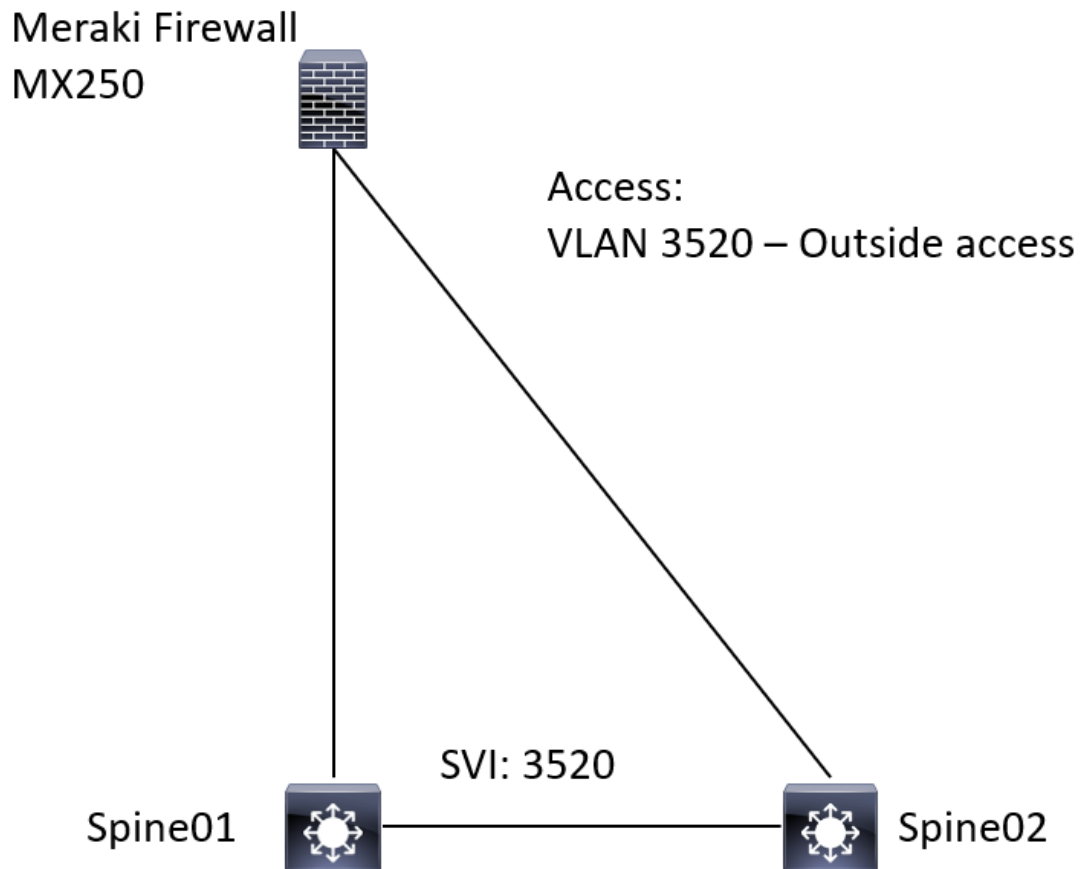


Figure 3 Spine to Firewall

The spine to firewall connection will be using 10 GbE connections over fibre, which will provide redundancy and a connection to the VMs as well as separate connection outside of the entire network for the VMs. This provides a local access to the VMs using VNC or SPICE, as well as ensures that the VMs can be used for connecting to the internet for educational purposes. A VLAN(s) will be dedicated to exist between Meraki and the spine, to which VRF connections will be routed to for outside connectivity.

2.3 Storage

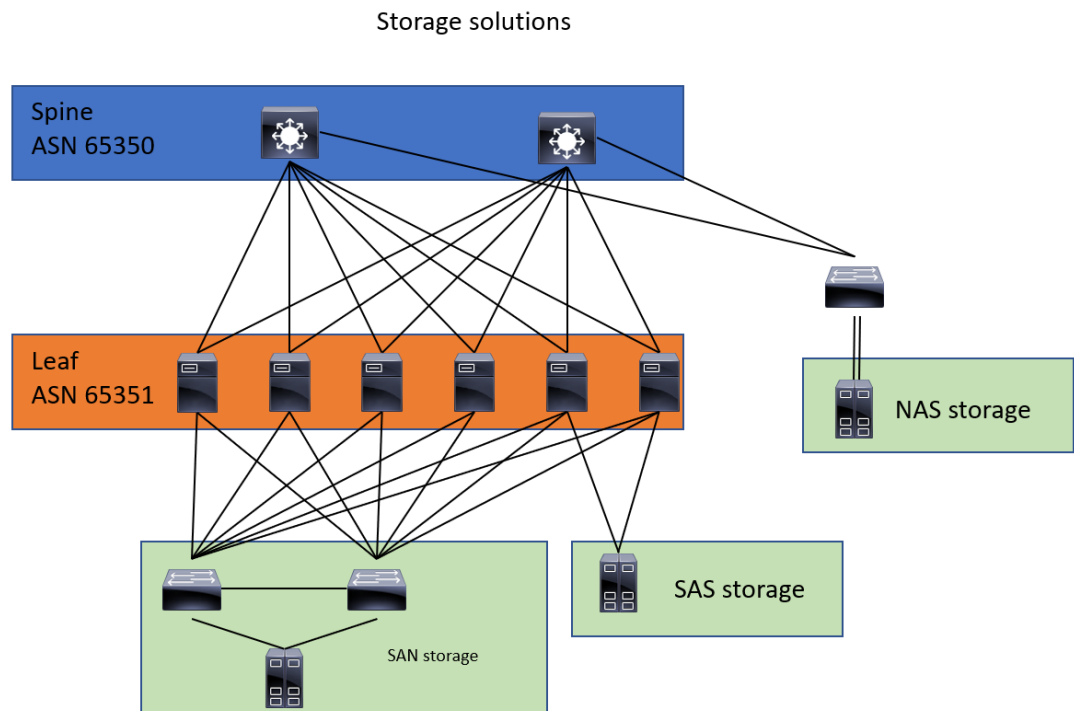


Figure 4 Storage solutions

Storage can be connected to the network in several ways. An option to connect directly to the servers a storage that is shared, creation of a Network Access Storage (NAS) or Storage Area Network (SAN). For use of NAS, it is possible to connect a Catalyst 2960 -switch to the spine, utilizing OSPF to isolate it and prevent Spanning-tree protocol (STP) from causing issues, broadcast storms or unpredictable looping behaviour from disjointed L2 layers.

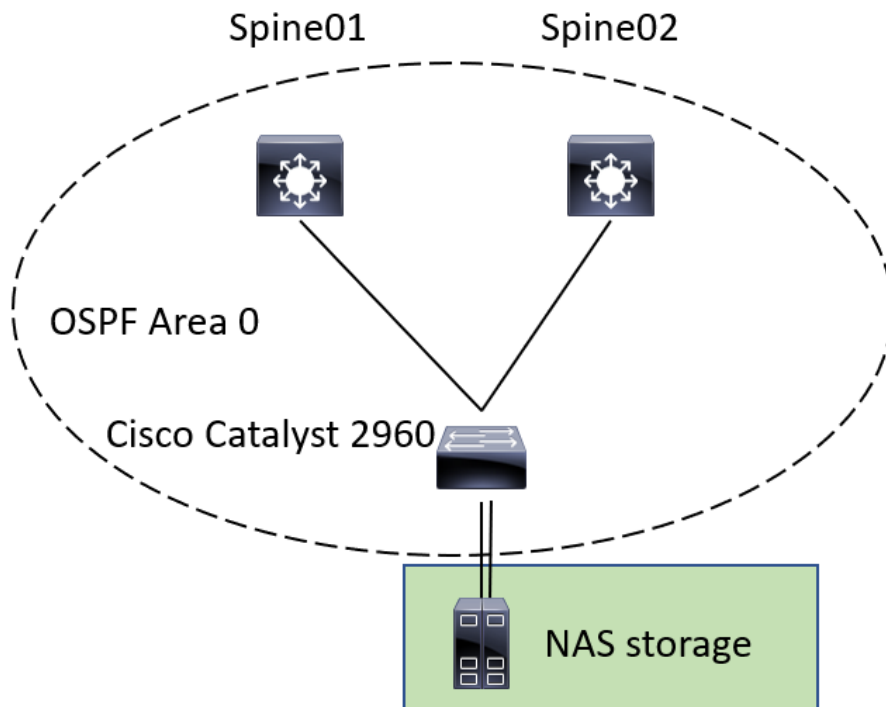


Figure 5 OSPF for storage switch

With this the VLAN dedicated to storage area should be possible to extend to the Cisco Catalyst 2960 switch that operates using OSPF lite; that way the need for STP can be removed.

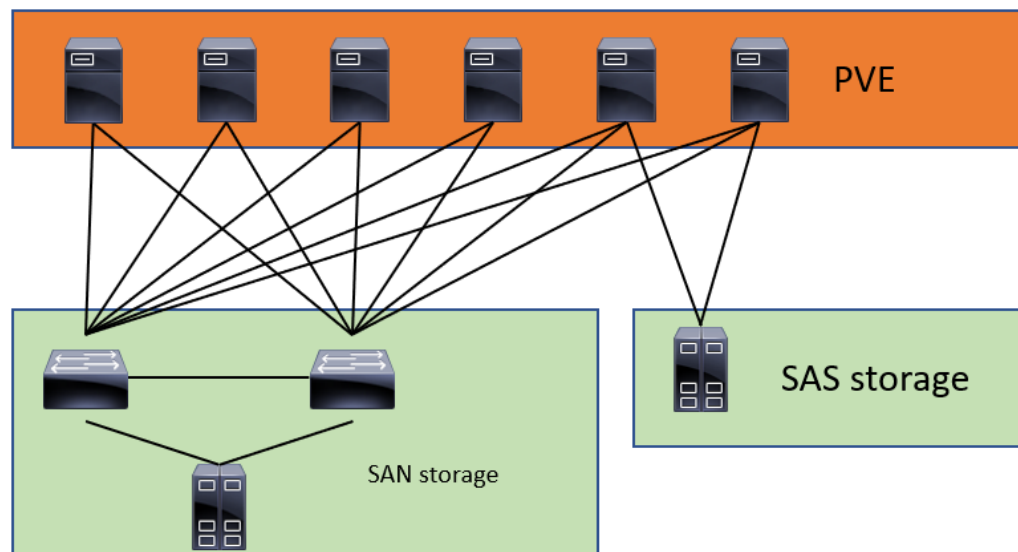


Figure 6 SAS and SAN -storage

It is also possible to utilize the unused ethernet ports of the servers to create a SAN area, to which storage could be connected to in order to provide redundancy for the connection. This option doesn't require more than utilization of two Catalyst 2960 switches as simple Layer 2, with interfaces in PVE nodes designated to connect to them and have a shared storage.

Another storage form that is present is the use of Serial Attached SCSI (SAS), which makes the storage be part of the server nodes. Sharing this storage is an option and if more SAS based storages are added, the use of CEPH should be considered.

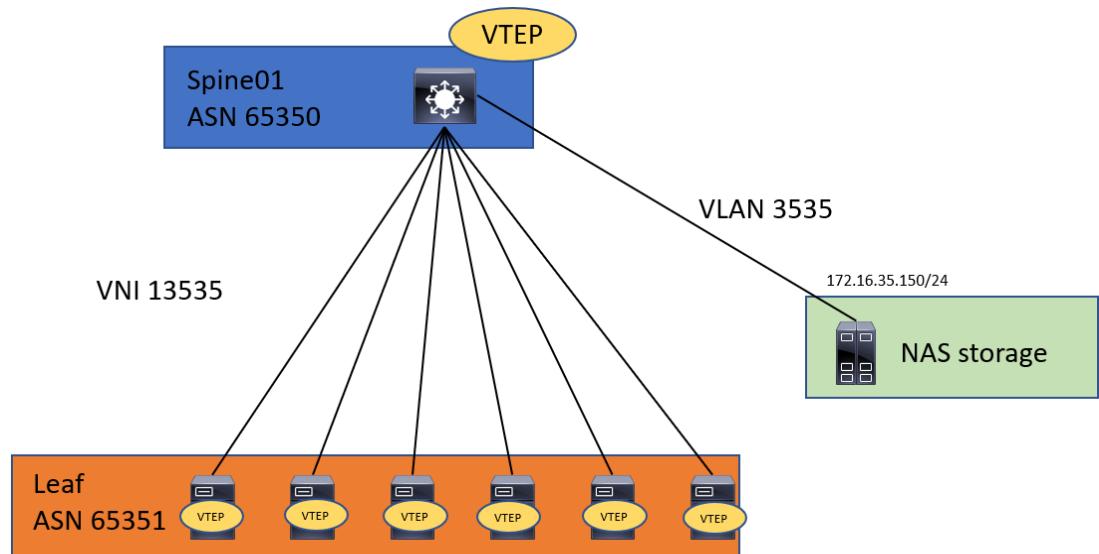


Figure 7 Spine connected storage with VTEP and VLAN

The design also allows the use of creating bridges in the server nodes, which can be connected to the Spine through VXLAN and in Spine allowed to connect to directly connected storage that exists in a VLAN that is tied to the VNI of the VXLAN. This acts as a Layer 2 connection between the nodes and the storage.

3 NETWORK SCHEME

Because the networks underlay is going to be relatively isolated, it provides ease in planning to the links between nodes and the Spine, however it is important that care is taken in consideration to the overlay networks that are intended to take part and be accessible to the rest of the network.

3.1 Loopback addressing

Device	Loopback
Spine01	10.0.2.1/32
Spine02	10.0.2.2/32
PVE001	10.0.2.11/32
PVE002	10.0.2.12/32
PVE003	10.0.2.13/32
PVE004	10.0.2.14/32
PVE005	10.0.2.15/32
PVE006	10.0.2.16/32

Figure 8 Loopback addressing

Loopbacks for each device need to be unique and follow a simple pattern to identify them. As they are not routed, a private addressing scheme of 10.0.0.0/8 has been used, as is typical, for loopback addresses. Spines range from 10.0.2.1/32 to 10.0.2.10/32 as the number of Spines is not expected to exceed ten units in short period of time; in perspective nodes are given the range from 10.0.2.11/32 to 10.0.2.255/32, however number of nodes is not expected to exceed twenty in a short period of time.

3.2 Spine to node addressing

Spine end			Leaf end		
Device	port	address	address	port	Device
Spine01	1	172.16.100.1/30	172.16.100.2/30	1	PVE001
Spine01	3	172.16.100.5/30	172.16.100.6/30	1	PVE002
Spine01	5	172.16.100.9/30	172.16.100.10/30	1	PVE003
Spine01	7	172.16.100.13/30	172.16.100.14/30	1	PVE004
Spine01	9	172.16.100.17/30	172.16.100.18/30	1	PVE005
Spine01	11	172.16.100.21/30	172.16.100.22/30	1	PVE006
Spine02	1	172.16.200.1/30	172.16.200.2/30	2	PVE001
Spine02	3	172.16.200.5/30	172.16.200.6/30	2	PVE002
Spine02	5	172.16.200.9/30	172.16.200.10/30	2	PVE003
Spine02	7	172.16.200.13/30	172.16.200.14/30	2	PVE004
Spine02	9	172.16.200.17/30	172.16.200.18/30	2	PVE005
Spine02	11	172.16.200.21/30	172.16.200.22/30	2	PVE006

Figure 9 Spine and Leaf link addressing

Figure 9 showcases the ports and addresses for each port in regard to Spine and each individual Leaf. The intention of utilizing only the uneven numbered ports, is to allow – in case it becomes necessary – the use of paired up connection to each node.

3.3 Reserved addressing

Reserved addresses				
VXLAN	VNI	Network	Vlan	Name
vxlan35100	35100	172.20.135.0/24	3510	VM_outside
vxlan35150	35150	172.18.15.0/24	3515	NGINX
vxlan35200	35200	172.19.0.0/16	3520	VM_remote_access
vxlan35035	35035	172.16.35.0/24	3535	storage
		172.16.10.0/24	3500	Cluster
		172.16.20.0/24	3500	Cluster_backup
vxlan35300-400	35300-35400	VLAN HUB	300-400	Internal vlan
		10.0.0.0/8		Network loopbacks

Figure 10 Reserved addressing

Figure 10 consists of planned and intended reservations in regard to the network, taking in consideration the possible use of NGINX as an http/https server for future use, as well as the few intended VXLAN interfaces that are required for use. Note that for interconnected VRFs, there is no addressing required, but a VNI is needed for each VRF created.

4 EQUIPMENT

The network equipment that are intended to be used for the installation have been chosen are several of the pre-existing Catalyst 2960 -switches, as well as two Cisco Nexus 93180 yc-FX-24 -switch and will be connected to the MX250 Cisco Meraki firewall.

Total of five Catalyst -switches should be reserved, with the Nexus -switches acting as Spine and Core. In addition, it is necessary to acquire Network Interfaces Cards (NIC) that have two SFP ports of 10Gbe speed.

4.1 Cabling

For connections between Spine and the servers, due to the relatively short distance intended in two server racks, use of fibre with transmitters is not the ideal solution. Using Small Form Pluggable (SFP) copper cables due to their relatively low price in comparison is intended for these connections. What needs to be taken into consideration is the compatibility of the cables, as using Cisco devices presents a requirement to have Cisco compatible SFP cables.

The connections from Spine to Meraki MX250 firewall will be over fibre, in which the use of Cisco compatible 10Gbe transceivers is necessary for each link. In order to ensure the possibility of using multiplexers, if the need arises later, the transceivers should be chosen to use different wavelengths.

5 LOGICAL TOPOLOGY

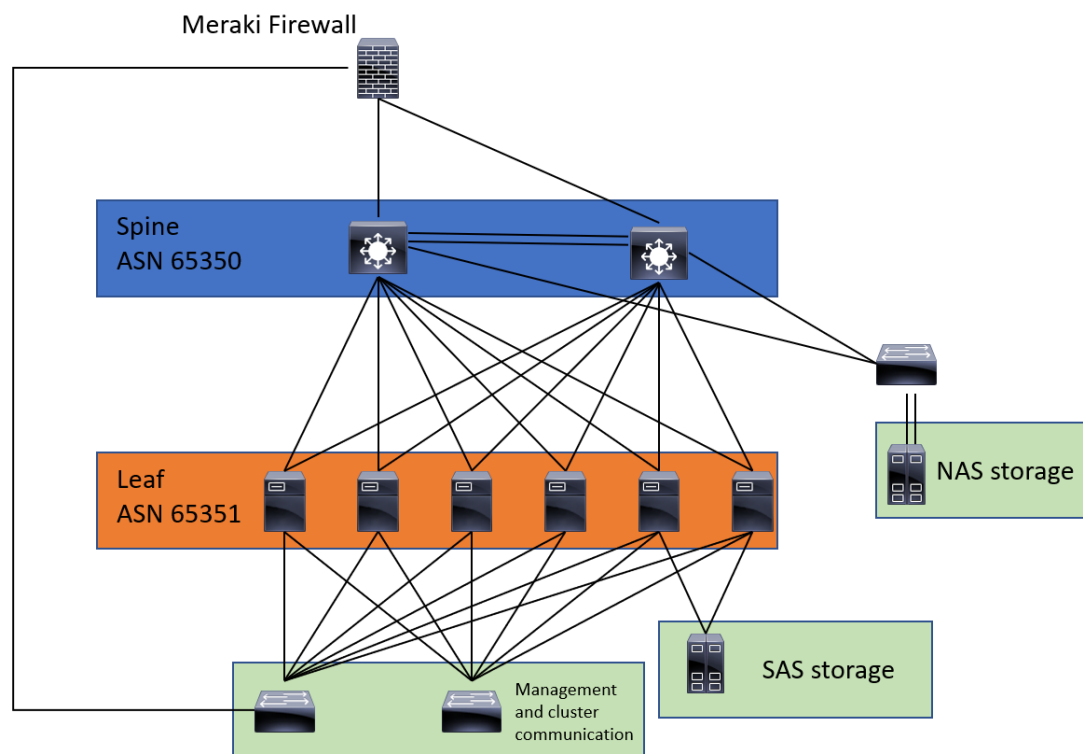


Figure 11 Logical topology of the network

5.1 Introducing NGINX with FRRouting

The BGP-EVPN provides a possibility of introducing NGINX as an edge device for services and as HTTP/HTTPS -proxy for the VXLAN environment. This can be done by hosting NGINX on a Debian GNU/Linux that has FRRouting installed onto it and provided a connection to the spine so that it can participate in the network. Through this the VXLANs can be brought to the NGINX with a VTEP.

Requirements: 4 cores or more on CPU
 Debian GNU/Linux OS
 8 GB of memory
 4 x 10 Gbe SFP ports

This allows connecting the NGINX into the Meraki MX250 firewall through redundant link, providing a single point that can allow redirecting of traffic to

HTTP/HTTPS -services through the NGINX on different overlays. This way it is possible to provide an IP -address to forward traffic directed to certain ports that is then redirected to the correct servers hosting the service in question. (figure 12.)

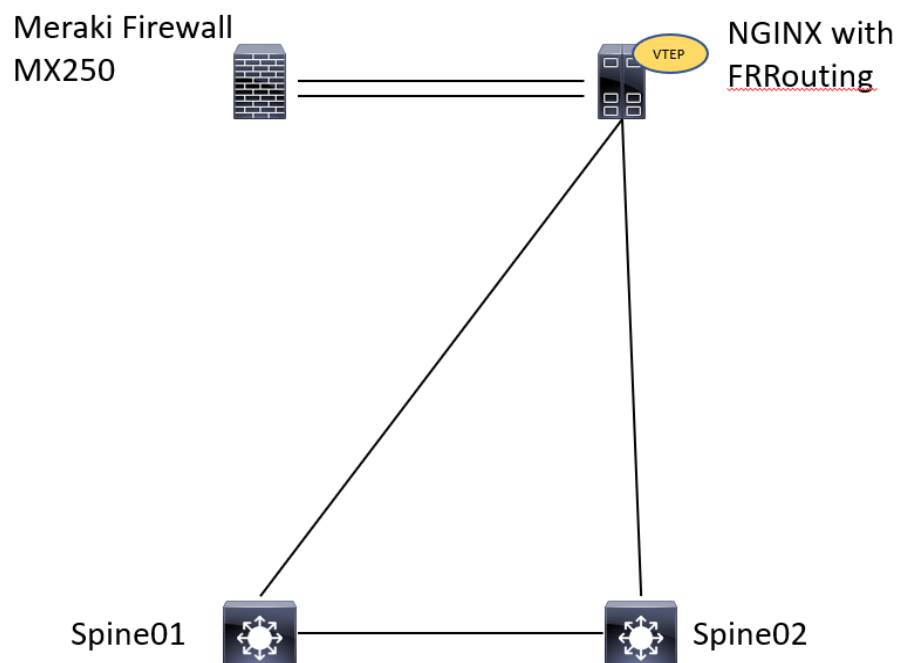


Figure 12 NGINX with FRRouting

Note that if PIM is to be introduced for multicasting, it is necessary to ensure that the Debian GNU/Linux has a kernel version that supports and enables the use of multicasting.

Roope Simola

Proxmox with FRRouting

Installation guide for data centre purposes

2020



South-Eastern Finland
University of Applied Sciences

CONTENTS

1	INTRODUCTION	4
2	PROXMOX SETUP	5
2.1	PROXMOX VE basic installation	5
2.2	Setting up Proxmox for installation of packages	10
3	NETWORK INTERFACES.....	15
3.1	Uplinks and MTU	15
3.2	VXLAN -interfaces	18
3.3	VRF in Proxmox.....	22
4	FRRROUTING CONFIGURATION	24
5	PROXMOX CLUSTER.....	26
5.1	Creating a cluster.....	27
	REFERENCE.....	31
	Figure 1 Initiation of installation	6
	Figure 2 storage options	7
	Figure 3 Time zone, Country and Keyboard layout.....	7
	Figure 4 Password and E-mail.....	8
	Figure 5 Management configuration	10
	Figure 6 Login screen	11
	Figure 7 Enterprise repository commented out.	11
	Figure 8 new repository added into the source list.....	12
	Figure 9 procuring keys for FRRouting	13
	Figure 10 adding FRRouting repository	13
	Figure 11 bgpd -daemon activation	14
	Figure 12 Basic setup in network configuration.	16
	Figure 13 Uplink configuration example.....	17
	Figure 14 Management interface and Source directive.....	18
	Figure 15 VXLAN -interface example.....	19

Figure 16 VXLAN -interface example.....	20
Figure 17 Bridge meant to visible through etc/network/interfaces -file.....	20
Figure 18 VXLAN -interfaces with VLAN access.	21
Figure 19 VLAN aware bridge for connecting to VXLAN -interfaces.....	21
Figure 20 VRF-table and VRF -participant bridge.....	22
Figure 21 Example of VRF inter-VXLAN -interface.....	23
Figure 22 FRRouting example setup.....	24
Figure 23 Excerpt from example configuration of FRRouting.	25
Figure 24 Except of Arista Spine configuration.	26
Figure 25 etc/hosts -file with correct IP for host.	27
Figure 26 Example of Proxmox cluster setup.	27
Figure 27 Navigating WebUI.....	28
Figure 28 Creating cluster TestingData.	28
Figure 29 Cluster join information.	29
Figure 30 Joining PVE02 into the cluster.....	29
Figure 31 Three nodes added to the cluster.	30

1 INTRODUCTION

This document is meant to be a step by step guide regarding installation of a Proxmox -node into Ekami data centre environment. It will go through the process of installing the hypervisor, any necessary components for it to operate and configuration of network interfaces and FRRouting to create a Proxmox Virtualized Environment -node (PVE -node) with LEAF -switch capabilities.

The intention of this document is to provide the means for the first-time setup of Proxmox -nodes and later addition of new ones. It will provide information regarding addition of interfaces, as well as backing up of important configuration files that are necessary for the function of the nodes in a cluster. It will also go over the introduction of storage to the cluster in the form of Network File System (NFS) but will not go over the configuration required for the SPINE -switches.

The network and infrastructure of this setup will be based on modern SPINE and LEAF -solution that is common in data centres, with the LEAF being implemented onto the PVE -nodes and SPINE -switches as the actual Layer 3 -switches.

The document is tailor made for this specific environment, holding IP -scheme and configurations specific for this environment to function.

2 PROXMOX SETUP

This portion of the document will go over the basic installation steps of Proxmox VE. In this guide the version of Proxmox used is 6.2, with could lead to some steps regarding later configurations not being applicable to earlier or later versions of the hypervisor.

To begin following this guide, it is necessary to create an installation media for Proxmox, which is recommended to be an USB -device – rather than CD/DVD. This is due to the increasing rarity of CD/DVD -drives, but commonality of USB -ports.

Note that Proxmox is a Debian based system, with a minimalistic approach, which means that until the system has been setup up to a certain point, there will be no graphical interface, aside from installation for initial installation, and everything thereafter is done in Command line interface (CLI).

It is also necessary to enable support for Kernel-based Virtual Machine (KVM) in BIOS, for optimal performance of the system. For the entire process to be successful, a network connection is required as to acquire necessary software and packages.

2.1 PROXMOX VE basic installation

Once the installation media has been introduced to the server that Proxmox is to be installed onto, and it has been selected as the bootable device to initiate the installation, the first thing that should be faced on the screen is the following image of figure 1.

Proxmox VE 6.2 (iso release 1) - <https://www.proxmox.com/>



Welcome to Proxmox Virtual Environment

Install Proxmox VE

Install Proxmox VE (Debug mode)

Rescue Boot

Test memory (Legacy BIOS)

Figure 1 Initiation of installation

Getting to this step means that the installation can be started as normal by choosing the “Install Proxmox VE” -option and proceeding onwards. Once the option has been chosen, the installation will begin setup and introduce the End User License Agreement (EULA) and once it has been agreed to the initial configuration of the system can commence.

The next step will be choosing the storage upon which the installation is made, along with options of the type of storage as well as size. By default, the installation will use ext4 -format and allocate all of the available space on the disk for use. The size and filesystem can be altered through options menu, and it is to be noted that if the size is lowered – the rest of the space will require partitioning to become available (Figure 2).



Figure 2 storage options

The next step is to choose the time zone, country and keyboard layout of the system; the following process is made easy through automation of the installation process, where only the country needs to be chosen for the rest of the options to be filled out correctly – with the possibility to alter the time zone and keyboard layout if necessary (figure 3).

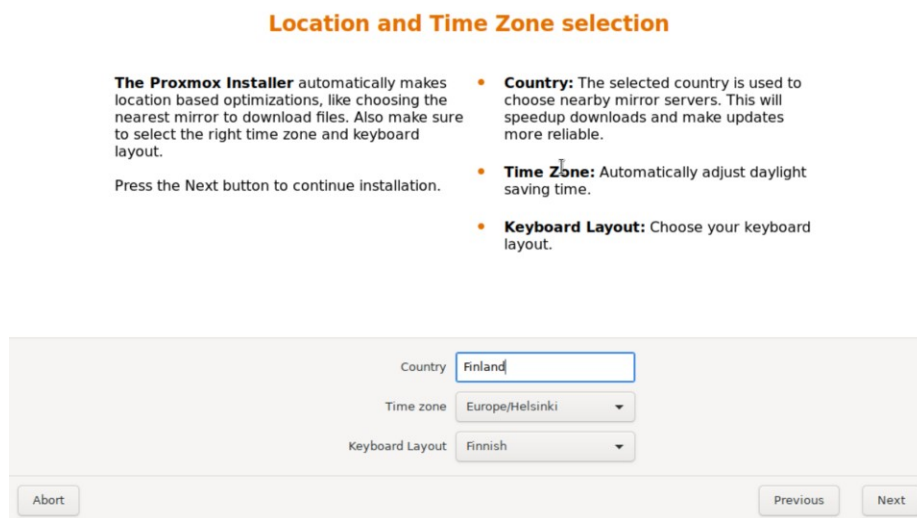


Figure 3 Time zone, Country and Keyboard layout

Once the locations and time zone has been set, it is necessary to input a password and email address. The password that is provided should be considered

carefully, made secure through best practices of password generation and kept known only by the individuals that manage and maintain the servers as it will be a root password. This means that in case the password is broken or obtained by a third party – there is a great chance of damage to be done to the entire system, as well as breach of privacy. Each server should be provided with a different password to mitigate access in case a breach occurs.

The email that is to be provided at this stage is for important alerts and notifications regarding the server, and as such it should be directed to the individual that is in charge of maintaining and supervising the servers and network (figure 4).

The screenshot shows a configuration window titled "Administration Password and E-Mail Address". It contains the following text and form elements:

- Proxmox Virtual Environment** is a full featured highly secure GNU/Linux system based on Debian.
- Please provide the *root* password in this step.
- Password:** Please use a strong password. It should have 8 or more characters. Also combine letters, numbers, and symbols.
- E-Mail:** Enter a valid email address. Your Proxmox VE server will send important alert notifications to this email account (such as backup failures, high availability events, etc.).
- Press the Next button to continue installation.

The form fields are:

- Password: [Masked with 8 dots]
- Confirm: [Masked with 8 dots]
- E-Mail: EmailofAlert@Employer.fi

Buttons at the bottom: Abort, Previous, Next.

Figure 4 Password and E-mail

After this comes the final step of the initial configuration process of the Proxmox - server, configuration of the management address and interface. This should be properly in order to provide Proxmox network connectivity for the duration of the initial setup; these configurations can, and will be, changed after the initial installation through network configuration files contained within the system.

Management Interface: This is the interface that the bridge that holds the network configurations will be attached to, and it should be chosen as the one directed to the management switch or device. (figure 5).

Hostname: This is the name of the server, in the example the name has been designated as “Ekami.Pve04.Example”, however it is ideal that the naming convention begins with the PVE followed by the number that identifies the server to enforce clarity.

IP Address and Netmask: This is the IP address with which the management interface will communicate and needs to be unique for each server introduced to the network. Netmask defines the size of the network and should already be pre-ordained – by default it is /24 (255.255.255.0).

Gateway: The address of the device that is used to manage traffic that is unknown to the network/device. In this circumstance it refers to the address of an interface (vlan or physical) which is responsible for routing traffic. I.e. Firewall or a router.

DNS server: Domain Name Service -server, which is responsible for handling domain name searches. By default Proxmox has a loop-back address for this, which defines itself as the DNS; the address should be the DNS -server of the network that Proxmox is able to reach through Management network.

Management Network Configuration

Please verify the displayed network configuration. You will need a valid network configuration to access the management interface after installation.

Afterwards press the Next button. You will be shown a list of the options that you chose during the previous steps.

- **IP address:** Set the IP address for your server.
- **Netmask:** Set the netmask of your network.
- **Gateway:** IP address of your gateway or firewall.
- **DNS Server:** IP address of your DNS server.

Management Interface:	ens5 - 00:8a:49:62:a6:13 (e1000) ▼
Hostname (FQDN):	Ekami.Pve04.Example
IP Address:	192.168.10.14
Netmask:	255.255.255.0
Gateway:	192.168.10.1
DNS Server:	127.0.0.1

Abort
Previous
Next

Figure 5 Management configuration

Once the configuration is done, there will be a summary until finally the installation process can be progressed to the actual installation of the system. This will take a moment, after which the system will request to be rebooted and removal of the installation media for the entire process to be complete.

2.2 Setting up Proxmox for installation of packages

From here on out, until the configuration of clusters and installation of virtual machines, the configuration process of Proxmox will occur in CLI. The commands necessary for the steps will be displayed and provided simple explanation and it is necessary that the steps are followed in correct order to avoid causing harm to the system or having to start over. Once the system has completed the installation, the first thing that is presented should be the following imagine with the first part of the hostname of the device (figure 6). Provide the root -username and password to access the system.


```
Welcome to the Proxmox Virtual Environment. Please use your web browser to
configure this server - connect to:

https://192.168.100.2:8006/

-----

Ekami login: _
```

Figure 6 Login screen

In order to begin to process of configuring and installing the necessary software, it is important to alter the repositories of Proxmox. These are locations where Linux systems can retrieve updates and applications, and by default Proxmox is set to use the Enterprise repository. The current repository choice will not work without a subscription to Proxmox, however there are repositories that are available to the free version of Proxmox. Proxmox itself encourages not using these for production purposes, yet for the purposes of the network that uses FRRouting and VXLAN it is necessary to use these to obtain the needed applications.

The first step in rectifying the repositories for use in this particular instance is to disable the enterprise repository from being used. This is a very simple task that does not require much more than using of VI -text editor on a correct file that contains the repository address. In there, a single line needs to be commented.

```
vi /etc/apt/sources.list.d/pve-enterprise.list
```

The following command above will use VI to edit the text file, where we can simply comment the line present by inserting a “#” before it (figure 7).

```
#deb https://enterprise.proxmox.com/debian/pve buster pve-enterprise
```

Figure 7 Enterprise repository commented out.

Next it is necessary to add the needed repositories for FRRouting and the free repository. The free repository will be added into the “sources.list” -file, where as like the enterprise -file, the FRRouting will be made itself its own file for clarity. However, it is necessary to have internet connection as before FRRouting -repos-

itory can be added, “lsb-release” application needs to be installed and this requires the system to update its repositories after the free repository has been inserted into the correct file.

Lsb-release is a simple Linux tool to help identify the Linux distribution in use, which makes finding the correct version of FRRouting easier.

Adding the free “pve-no-subscription” -repository next, we have to add the following line into the file “sources.list” – “deb http://download.proxmox.com/debian buster pve-no-subscription”.

```
Vi /etc/apt/sources.list
```

The above command will get us into the file, after which we can add the new repository path – without removing any of the other previous repositories (figure 8).

```
deb http://ftp.fi.debian.org/debian buster main contrib
deb http://ftp.fi.debian.org/debian buster-updates main contrib
deb http://download.proxmox.com/debian buster pve-no-subscription
# security updates
deb http://security.debian.org buster/updates main contrib
```

Figure 8 new repository added into the source list

With the new repository added, the following command needs to be run for the system to update to use it. In case issues emerge, check files for spelling errors or ensure you have connectivity outside of the network.

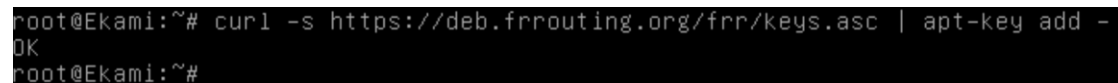
```
apt-get update
```

Once it has successfully ran through the update process, it is possible to install the following applications: Ifupdown2, lsb-release, pve-manager. Installing these at this point is advisable. Ifupdown2 is needed for creating and using VXLAN interfaces, where as pve-manager is a Proxmox management tool.

```
Apt-get install -y ifupdown2
Apt-get install -y lsb-release
Apt-get install -y pve-manager
```

Once at least lsb-release has been installed, the adding of FRRouting repository becomes easy. Beginning with the following command, the keys necessary to access it are procured and added into the system. If the process is successful, there should be a simple “OK” -prompt in the CLI (figure 9.)

```
curl -s https://deb.frrouting.org/frr/keys.asc | apt-key add -
```



```
root@Ekami:~# curl -s https://deb.frrouting.org/frr/keys.asc | apt-key add -
OK
root@Ekami:~#
```

Figure 9 procuring keys for FRRouting

With the keys obtained, the next step is to run few simple commands. The first one will simply make it so that the statement FRRVER contains the “frr-stable” line within it for brevity, while the echo will add the repository line into “sources.list.d” and creates a file named “frr.list” while utilizing lsb-release to acquire the rest of the repository name and path.

```
FRRVER="frr-stable"
Echo deb https://deb.frrouting.org/frr $(lsb_release -s -c) $FRRVER |
tee -a /etc/apt/sources.list.d/frr.list
```



```
root@Ekami:~# FRRVER="frr-stable"
root@Ekami:~# echo deb https://deb.frrouting.org/frr $(lsb_release -s -c) $FRRVER
R | tee -a /etc/apt/sources.list.d/frr.list
deb https://deb.frrouting.org/frr buster frr-stable
root@Ekami:~# _
```

Figure 10 adding FRRouting repository

In figure 10 is the final output of a successful addition of the repository, which should be followed by “apt update” -command to update the repositories again. Once the process has been successful, it is possible to finally install FRRouting and pythontools related to it.

```
apt-get install -y frr frr-pythontools
```

With FRRouting successfully installed, there are two more steps that are necessary to take care of. Enabling FRRouting to start on boot, so as to avoid having to start up the routing software upon every reboot or unexpected outage, as well as enabling the correct daemons within FRRouting that are to be used for the current installation. In this example the bgpd daemon will be enabled.

```
systemctl enable frr
```

The command above will tell the system to enable FRRouting and make sure that it will be start-up on boot. The daemons that control the various parts of FRRouting are within the FRRouting -file and will require VI to edit it. Changing the “No” statement to “Yes” will enable the daemon once FRRouting has been restarted (figure 11). However, in this case the system will be rebooted to enable the various components installed prior.

```
vi /etc/frr/daemons
```

```
# ATTENTION:
#
# When activating a daemon for the first time, a config
# empty, has to be present *and* be owned by the user
# the daemon will not be started by /etc/init.d/frr.
# be u=rw,g=r,o=.
# When using "vtysh" such a config file is also needed
# group "frrvty" and set to ug=rw,o= though. Check /e
#
# The watchfrr and zebra daemons are always started.
#
bgpd=yes
ospfd=no
ospf6d=no
ripd=no
ripngd=no
isisd=no
pimd=no
ldpd=no
```

Figure 11 bgpd -daemon activation

Now the final step to finalize the entire process is to reboot the system. Once rebooted, configuration of VXLAN interfaces and FRRouting can be done.

3 NETWORK INTERFACES

The network interfaces that are required to implement VXLAN and the “Spine and Leaf” -structure are necessary to implement into the Proxmox network configuration files. Ifupdown2 -application is needed for the VXLAN interfaces, which will be providing the Layer 2 connection between Virtual Machines (VM) and Proxmox servers and storage solutions and the Edge Firewall/Router.

There are several different types of interfaces that can be implemented and FRRouting also supports the use of Virtual Routing and Forwarding (VRF), however this is not necessary to implement for unless more advanced design solutions are planned to be implemented.

It is also important to note that the typical “Ifconfig” -command will not function in the Proxmox CLI, instead it will utilize “ip link” and “ip add” -commands as it utilizes uproute2.

3.1 Uplinks and MTU

In order to provide connectivity upwards towards the spine switches, there needs to be links and addresses configured onto the network configuration file. The first step is to go to “etc/network/interfaces” – file to alter the behaviour of the network connections. In figure 12 there is shown what should be at the top of the configuration file, since it is important that the order of interfaces is correct. This is due to the fact that Linux systems read their configuration files from top to bottom, left to right, leading to interfaces being raised in the order of topmost being first and bottom last. If in the order of the interfaces there is something bonding or bridging to an interface that is raised after it, the action will fail on the initial boot or restart of the network daemon.

Note that in order for the interfaces that are intended to be seen in the webUI need to have the naming convention of “vmbr”, i.e. “vmbr100”, otherwise they do not show are not available for use.

```
auto lo
iface lo inet loopback
        address 10.0.2.13/32

iface ens2 inet manual

iface ens3 inet manual

iface ens4 inet manual

iface ens5 inet manual
```

Figure 12 Basic setup in network configuration.

The first one is the Loopback, which will be essential for directing flow of traffic and VXLANs in the network, followed by the physical interfaces in manual state. Manual state creates a link that has no IP address of its own, so it will take its address and attributes from bridges or bond master assigned/connected to it.

These lines should be followed up the links that have actual addresses and MTU values that the interfaces should use, called uplinks in this case, so as to avoid the issue of interfaces not going up properly. The figure 13 showcases an example of two uplinks, which are bond masters and will convey their attributes to the physical interfaces designated as their slaves. The configuration is done like this to ensure that addresses are on a virtual interfaces that will never go down or stop functioning, and also allows the use of one or multiple different physical interfaces with relative ease to either increase or decrease the redundancy and capacity of the link in question.

```
auto uplink1
iface uplink1 inet static
    mtu 9182
    address 172.16.13.2/30
    bond-slaves ens3
    bond-miimon 100
    bond-updelay 200
    bond-downdelay 200
    bond_mode 0

auto uplink2
iface uplink2 inet static
    mtu 9182
    address 172.16.23.2/30
    bond-slaves ens4
    bond-miimon 100
    bond-updelay 200
    bond-downdelay 200
    bond_mode 0
```

Figure 13 Uplink configuration example

MTU: set to 9182 (jumbo sized frames), defines the MTU value that is by default 1500 and should be at least 1550.

Bond-slaves: designates which interfaces are tied to this virtual interface

Bond-miimon: set to 100, defines how often the link state of the slaves is inspected in milliseconds.

Bond-updelay: set to 200, defines how long to wait until the slave is enabled after link has recovered in milliseconds.

Bond-downdelay: set to 200, defines how long to wait until the slave is disabled upon link failure in milliseconds

Bond-mode: set to 0, balance-rr (round-robin), defines which type of bond is in use and how the physical interfaces are utilized in the setup.

Finally, at the bottom should be placed the lines regarding the management interface. This should be there by default in some capacity due to the initial installation process, however altering it to fit the nature of the network is advised in some cases.

And after the management interface there should be added the following line “source /etc/network/interfaces.d/* “. This line will be used to search and read other network configurations in another location, targeted at “/etc/network/interfaces.d/ “, the “*” is there to include all files found in this directory. This is to keep the main configuration file clean, short and simple, as after inclusion of this line the VXLAN -interfaces can be individually added into the interfaces.d -directory (figure 14.)

```
auto vmbr0
iface vmbr0 inet static
    bridge-ports ens5
    bridge-stp off
    bridge-fd 0
    address 192.168.10.13
    netmask 255.255.255.0
    gateway 192.168.10.1

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

Figure 14 Management interface and Source directive

3.2 VXLAN -interfaces

The VXLAN -interfaces which will be responsible for delivering the traffic of the overlay network of a “Spine and Leaf” -network solution, can be implemented in several different ways into a Proxmox server with the use of ifupdown2. There can be interfaces that can be dedicated for communication between the servers, towards storage or simply for the VMs and Containers that are housed in the cluster. The unique identifiers ensure that the traffic is isolated from one another like with the typical VLAN. This document will not cover the addition of VLANs into the networking tied to VXLAN, as VLANs can be used in tandem with VXLANs to some capacity.

To ensure that the interfaces remain clear and easily managed, the VXLANs should be configured under the interface.d -directory, as well as to ensure that

they are not visible options for network interfaces that could be used for VMs and Containers. The bridges that utilize these interfaces will need to go to the “etc/network/interfaces” -file like the others, but after the “source” -directive.

Note that this does cause conflict with the webUI -interface, meaning configuration and adding of new interfaces through it might become unavailable as the webUI only searches primarily the “etc/network/interfaces” -file.

```
root@Ekami:~# cat /etc/network/interfaces.d/vxlan1500
auto vxlan1500
iface vxlan1500
    vxlan-id 500
    vxlan-local-tunnelip 10.0.2.13
    bridge-learning off
    bridge-arp-nd-suppress on

auto vubr500
iface vubr500 inet static
    address 10.0.50.13/24
    bridge-stp off
    bridge-fd 0
    bridge-ports vxlan1500
```

Figure 15 VXLAN -interface example.

Figure 15 showcases the path in which VXLAN -interface, named “vxlan1500”, has been created and the contents of the file. The bridge, vubr500 in this case, connected to it, which has an address, could be utilized for communication to a storage network.

Vxlan-id: The identifier of the particular VXLAN tunnel.

Vxlan-local-tunnelip: The Virtual Tunnel End Point (VTEP) that the vxlan interface is directed towards. In this case it is the loopback address of the Proxmox server being configured.

Bridge-learning: set to off, controls does the interface learn mac-addresses. This is not ideal for the vxlan -interface to be on.

Bridge-arp-nd-suppress: set to on, allows to suppress ARP flooding as much as possible.

Bridge-stp: set to off, defines is the Spanning Tree Protocol (STP) in use. Should not be used in tandem to VXLAN unless more advanced configurations are planned.

Bridge-fd: set to 0, defines the forwarding delay of the interfaces joining this bridge.

Bridge-ports: The port(s) that the bridge is connected to.

```
auto vxlan1100
iface vxlan1100
    vxlan-id 100
    vxlan-local-tunnelip 10.0.2.13
    bridge-learning off
    bridge-arp-nd-suppress on
```

Figure 16 VXLAN -interface example.

In figure 16 the interface is almost identical, aside from the fact that there is no bridge included into the file. This bridge will need to be added into the “etc/network/interfaces” -file (figure 17).

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

auto vubr100
iface vubr100 inet manual
    bridge-stp off
    bridge-fd 0
    bridge-ports vxlan1100
```

Figure 17 Bridge meant to visible through etc/network/interfaces -file.

The addition of VLANs is a simple process, which requires Virtual Machine Bridge (VMBR) which acts as a hub to the VMs and Containers connected to it, as well as the various VXLAN interfaces that guide the traffic forwards. Each VXLAN interface can be designated a single, multiple or a range of VLANs that may access it – allowing the use of local VLANs as well as cluster wide VLANs.

```
auto vxlan1200
iface vxlan1200
    vxlan-id 200
    vxlan-local-tunnelip 10.0.2.11
    bridge-learning off
    bridge-arp-nd-suppress on
    bridge-access 200

auto vxlan1300
iface vxlan1300
    vxlan-id 300
    vxlan-local-tunnelip 10.0.2.11
    bridge-learning off
    bridge-arp-nd-suppress on
    bridge-access 300
```

Figure 18 VXLAN -interfaces with VLAN access.

In figure 18, there is an example configuration of two VXLAN -interfaces that have both their unique VXLAN identifiers, 200 and 300, as well as designated access of VLANs 200 and 300 (bridge-access). To them is connected the hub interface, to which all VMs and Containers that are planned to utilize it are connected (figure 19). This means that VMs and Containers that have been connected to the hub bridge, as well as given the tagged traffic of 200 or 300, will be isolated from one another and move through the network with their own unique VXLAN identifiers (VNI).

In short, it means that the VLANs ensure that locally, in the same node, connected VMs or Containers cannot communicate with one another unless in the same VLAN or two separate instances (on different nodes) cannot communicate with one another unless in the same VLAN.

```
auto vubr1430
iface vubr1430 inet manual
    bridge-fd 0
    bridge-stp off
    bridge-ports vxlan1200 vxlan1300
    bridge-vlan-aware yes
```

Figure 19 VLAN aware bridge for connecting to VXLAN -interfaces.

3.3 VRF in Proxmox

With implementation of VRF there is a need to create a new routing table, which will be used as the routing table for the VNIs and through that for VMs and Containers to elevate the L2VPN to L3VPN. This means that routing becomes involved. The introduction of a VRF into Proxmox is simple but does cause certain complications if VLAN is considered – VLAN aware -bridges do not become a part of a VRF at this current version. Configuring VRF is done in the `/etc/network/interfaces` -file with the help of `ifupdown2` and inclusion of a bridge into VRF requires small addition to the bridge; in figure 20 can be seen how the VRF -table is made, as well as the additions to the bridge. The bridge interface is given a handmade mac-address and IP -address belonging to the subnet that is used by the VMs, typically a gateway address, which should be identical in each node for that bridge. Meaning that `vmbr1430`, in each node, would have the same configuration scheme. VXLAN -interface must not have the “**bridge access**” -line in it.

```
auto RED
iface RED
    vrf-table auto

auto vmbr1430
iface vmbr1430 inet static
    address 192.168.60.1/24
    hwaddress 00:00:00:00:00:1a
    bridge-stp off
    bridge-fd 0
    bridge-ports vxlan1600
    vrf RED
```

Figure 20 VRF-table and VRF -participant bridge

With VRF there is also need for additional interface with a unique VNI, which will be added into FRR on its own. This connection does not need to appear to others, and as such can safely be placed into the `/etc/network/interface.d/` -directory (or the sourced directory in use). (figure 21).

```
auto vxlan3535
iface vxlan3535
    vxlan-id 3535
    vxlan-local-tunnelip 10.0.2.12
    bridge-learning off
    bridge-arp-nd-suppress on

auto vubr3535
iface vubr3535 inet manual
    bridge-ports vxlan3535
    bridge-stp off
    bridge-fd 0
    vrf RED
```

Figure 21 Example of VRF inter-VXLAN -interface

In this example a VRF, named RED, would be created and in FRRouting it would be assigned the VNI 3535. A gateway node or edge device would then be able to import the VRF into its routing table and provide a default gateway (0.0.0.0/0) into the VRF and install (or leak) the addresses from RED into its default VRF. Default VRF in this case would refer to the routing table that is used normally by the router to direct traffic.

4 FRRROUTING CONFIGURATION

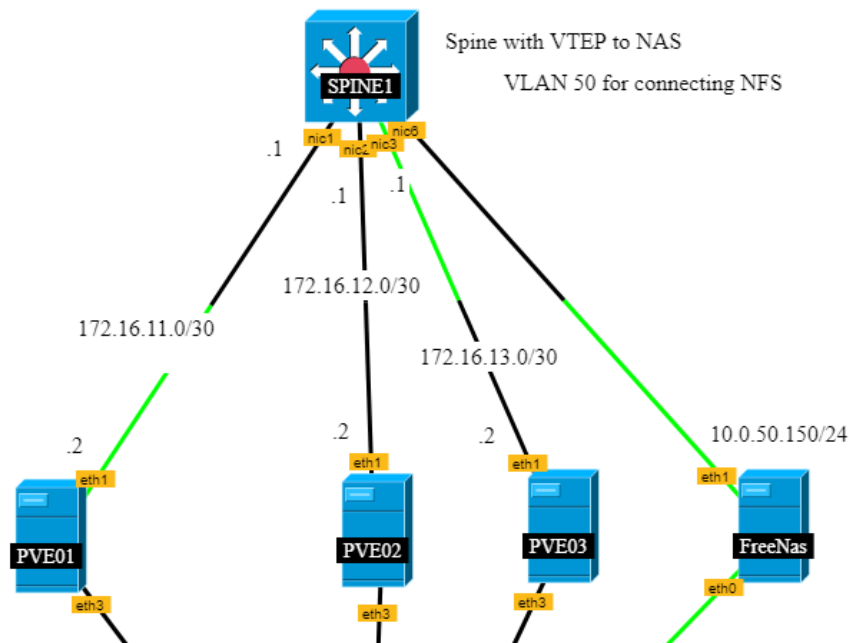


Figure 22 FRRouting example setup

FRRouting is the software which will take place of a Leaf -switch that would be typically found in large scale or modern datacentre, as it acts as an access point to the servers and connects them together through the Spine. Since no Leaf -switches are present in the setup, as they have been collapsed into the Proxmox -servers, FRRouting is necessary to implement use of VXLAN.

Note that FRRouting does not supersede some of the Linux configurations, such as those regarding IP -forwarding. To ensure that FRRouting has IP -forwarding allowed, it must be allowed in the `/etc/sysctl.conf` -file by uncommenting the respective lines regarding IPv4 and IPv6.

Configuration of FRRouting can be done through two different ways; with “vtysh” -command it is possible to enter a configuration state similar to that of a Cisco OS, the other is by altering the contents of “`etc/frr/frr.conf`” -file directly (figure 23).

When configuring through “vtysh”, it is important to remember that command “wr mem” is needed to ensure that the configuration is saved, retaining it through re-boots. Also important is to note that due to the way the interfaces have been configured through and into “etc/network/” -directory, there is no need to make additional configurations to the interfaces unless more advanced actions are planned. FRRouting will learn the addresses and interfaces through Kernel.

```
!
router bgp 65351
  bgp router-id 10.0.2.13
  no bgp default ipv4-unicast
  neighbor SPINE peer-group
  neighbor SPINE remote-as 65350
  neighbor SPINE capability extended-next-hop
  neighbor 172.16.13.1 peer-group SPINE
  neighbor 172.16.23.1 peer-group SPINE
  !
  address-family ipv4 unicast
    network 10.0.2.13/32
    neighbor SPINE activate
    neighbor SPINE allow-as-in
  exit-address-family
  !
  address-family l2vpn evpn
    neighbor SPINE activate
    advertise-all-vni
  exit-address-family
```

Figure 23 Excerpt from example configuration of FRRouting.

As EVPN is used, the address family “l2vpn evpn” needs to be configured, in which there is a particularly unique command to FRRouting. The command “advertise-all-vni” can be used to replace the need to individually add each VNI and their Route Distinguisher (RD), and does communicate this information to the SPINE just as it does to the other parts of the network without additional steps needed at the spine.

In case there are VTEPs in the Spine, addressing them down towards the servers is enough to enable the communication. In figure 24 is an example of a configuration within an Arista Spine -switch, using external BGP (eBGP) solution. A VLAN 50 has been created, and then in BGP has been given an RD and made to advertise the routes to the Leaf -switches.

```

interface Vlan160
  no autostate
  vrf BLUE
  ip address 192.168.160.253/24
  vrrp 160 priority-level 253
  vrrp 160 advertisement interval 3
  vrrp 160 ipv4 192.168.160.1
!
interface Vxlan1
  vxlan source-interface Loopback0
  vxlan udp-port 4789
  vxlan vlan 70 vni 700
  vxlan vlan 160 vni 1600
  vxlan vrf BLUE vni 35351
!
ip routing
ip routing vrf BLUE
!
ip prefix-list BLUE seq 10 permit 192.168.160.0/24
ip prefix-list default seq 10 permit 0.0.0.0/0
!
ip route 0.0.0.0/0 Vlan10 192.168.110.100
!
route-map BLUE permit 10
  match ip address prefix-list BLUE
!
route-map default permit 10
  match ip address prefix-list default
!
router bgp 65350
  router-id 10.0.2.2
  no bgp default ipv4-unicast
  distance bgp 10 10 10
  neighbor EDGE peer group
  neighbor EDGE remote-as 65349
  no neighbor EDGE allowas-in
  neighbor EDGE ebgp-multihop
  neighbor EDGE send-community extended
  neighbor EDGE maximum-routes 12000
  neighbor LEAF peer group
  neighbor LEAF remote-as 65351
  neighbor LEAF allowas-in 3
  neighbor LEAF ebgp-multihop
  neighbor LEAF send-community extended
  neighbor LEAF maximum-routes 12000
  neighbor 172.16.2.2 peer group EDGE
  neighbor 172.16.21.2 peer group LEAF
  neighbor 172.16.22.2 peer group LEAF
  neighbor 172.16.23.2 peer group LEAF
!
  address-family evpn
    neighbor EDGE activate
    neighbor LEAF activate
  !
  address-family ipv4
    neighbor EDGE activate
    neighbor LEAF activate
    network 10.0.2.2/32
  !
  vrf BLUE
    rd 10.0.2.2:1600
    route-target import evpn 65351:35351
    route-target export evpn 65351:35351
    router-id 10.0.2.2
    network 0.0.0.0/0
    redistribute connected
    redistribute static
!
router general
  vrf default
    leak routes source-vrf BLUE subscribe-policy BLUE
  !
  vrf BLUE
    leak routes source-vrf default subscribe-policy default

```

Figure 24 Except of Arista Spine configuration.

5 PROXMOX CLUSTER

Creating a cluster with Proxmox servers (nodes) is a relatively simple task that can be accomplished utilizing the CLI or the webUI. Due to the relatively sensitiv-

ity to lag, it is necessary that the traffic that is utilized by the cluster to communicate with one another is handled as quickly and through the shortest path possible. Typically, a switch that connects the management ports, as is used in this demonstration, or a separate switch entirely, should be used for this.

Note that if the management interface has been reconfigured after initial installation, it is necessary to check that in “etc/hosts” -file the IP-address is the same as the new address meant to be used by the Proxmox server (figure 25).

```
root@Ekami:~# cat /etc/hosts
127.0.0.1 localhost.localdomain localhost
192.168.10.13 Ekami.Pve03 Ekami
```

Figure 25 etc/hosts -file with correct IP for host.

5.1 Creating a cluster

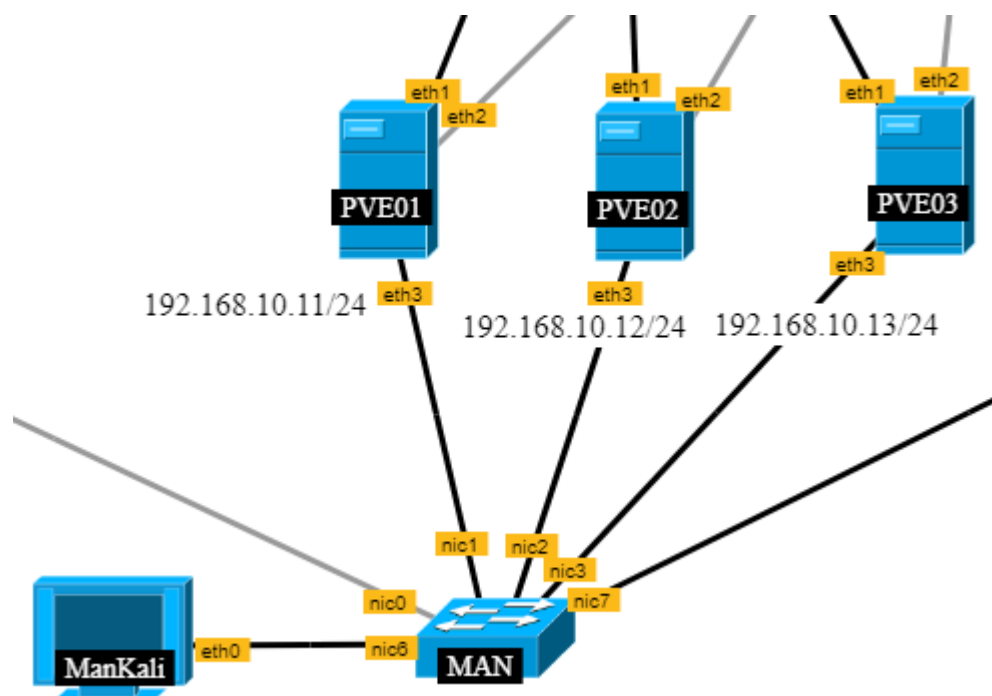


Figure 26 Example of Proxmox cluster setup.

Connecting each and every Proxmox node to switch is needed, and in the figure 26 there is an example of what the setup looks like. Each “eth3” responds to the management port of each Proxmox server, which has been connected to a switch

labelled as “MAN” – where the management device “ManKali” is used to interact with the webUI. To reach this webUI, using a preferred browser, connecting to “[https://\(ip-of-server\):8006](https://(ip-of-server):8006)” allows access to the webUI. Accepting the certificates through advanced section allows access into the server where root user and password is then used to log into the server.

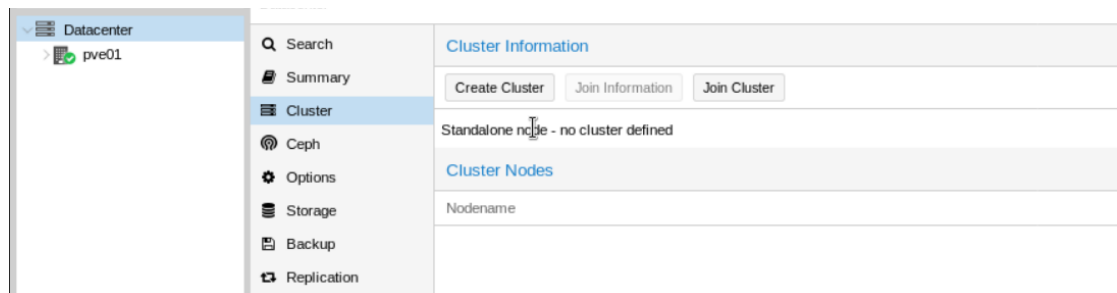


Figure 27 Navigating WebUI.

In this example the PVE01 has been chosen for creation of the cluster, which begins by choosing the datacentre option on the left side, after which it is necessary to navigate to the section “cluster”. There it will provide several options, from which two are available: “create cluster” and “join cluster” (figure 27). Choosing the “create cluster” option, we can then dictate the name of the cluster and what interface (IP-address) is used for the communication (figure 28).

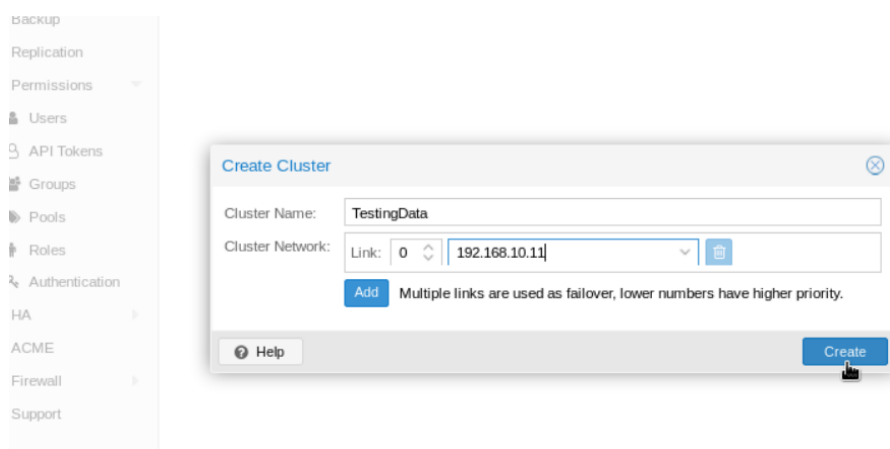


Figure 28 Creating cluster TestingData.

After the cluster has been created, a new option will become available “Join information”. Selecting it and copying the information allows the joining of other nodes to the newly formed cluster (figure 29).

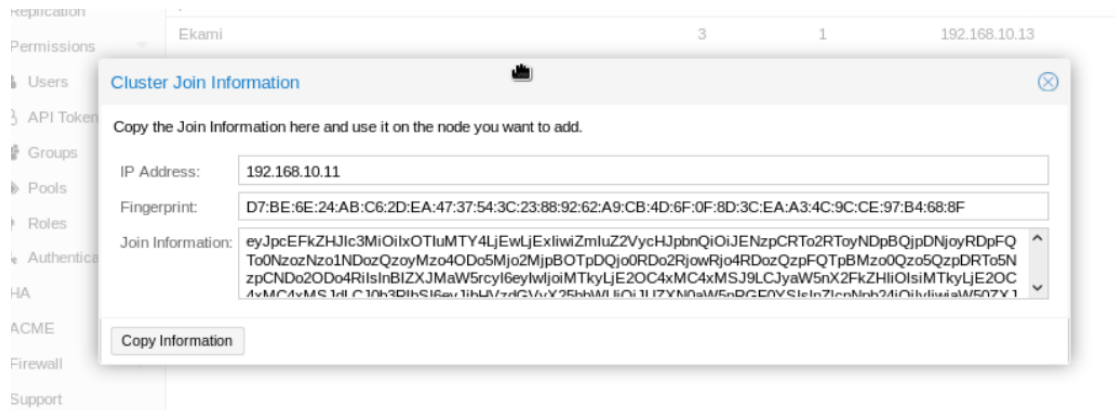


Figure 29 Cluster join information.

Once it has been copied, connecting to the next node that is meant to be a part of the cluster should be done. Navigating there to the same location as with creating the cluster, this time the option “Join cluster” is utilized. Pasting the copied join information of the cluster, the root password of the cluster creating node is used and the link that joins this cluster communication is selected (figure 30). This step is repeated for each node that is meant to join the cluster.

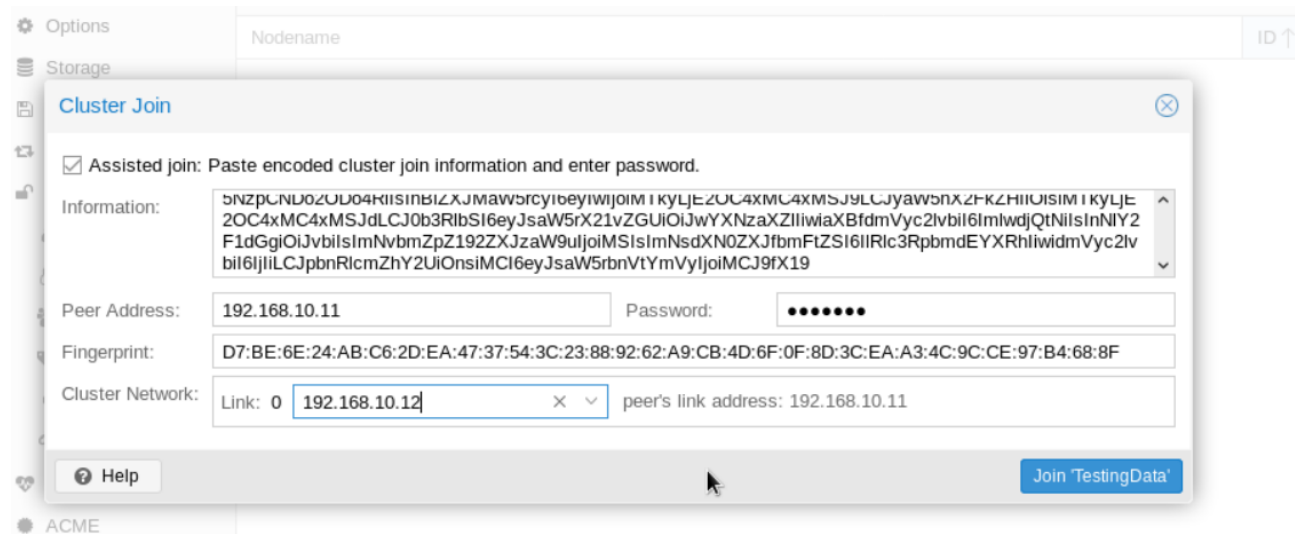


Figure 30 Joining PVE02 into the cluster.

Once each node has been added, note that adding each one will disable the functions of the webUI until reconnecting to the original address (not refreshing) and will showcase each node underneath the “Datacentre” -section (figure 31).

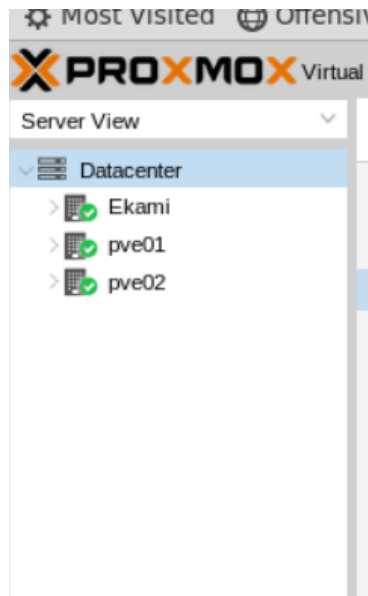


Figure 31 Three nodes added to the cluster.

REFERENCE

Proxmox. 2020. Proxmox VE. WWW document. Available at: https://pve.proxmox.com/mediawiki/index.php?title=Main_Page&oldid=10654 [Accessed 22.10.2020]

FFrouting. 2017. FRRouting Project. WWW document. Available at: <https://frrouting.org/> [Accessed 21.10.2020]

Git Proxmox. No date. Git Proxmox. WWW document. Available at: https://git.proxmox.com/?p=pve-docs.git;a=blob_plain;f=vxlan-and-evpn.adoc;hb=HEAD [Accessed: 22.11.2020]