Amin Karaoui

Automating VLAN management with Infoblox

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor’s Thesis

27 February 2023

Abstract

Author:	Amin Karaoui
Title:	Automating VLAN management with Infoblox
Number of Pages:	34 pages
Date:	27 February 2023
Degree:	Bachelor of Engineering
Degree Programme:	Information Technology
Professional Major:	IoT and Networks
Supervisors:	Marko Uusitalo, Senior Lecturer

The goal of this study was to migrate existing VLANs into Infoblox and create automation around it. The reason was that some VLANs were already managed through Infoblox, but some were stored in excel spreadsheets.

With careful planning and testing in dev environments, the imports were completed smoothly. Automation was also built to speed up manual tasks and new ideas for automatization were inspired throughout the process.

Researching the frameworks built for Infoblox took time but the results were promising and helpful in building knowledge regarding different automation frameworks.

Keywords: Infoblox, VLAN, DDI, network automation

Tiivistelmä

Tekijä:	Amin Karaoui
Otsikko:	VLAN-hallinnan automatisointi Infobloxilla
Sivumäärä:	34 sivua
Aika:	27.2.2023
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	IoT ja tietoverkot
Ohjaajat:	Lehtori Marko Uusitalo

Tutkimuksen tavoite oli siirtää olemassa olevia VLANeja Infobloxiin ja rakentaa automaatiota sen päälle. Syy projektille oli, että joitain VLANeja hallittiin jo Infobloxiin kautta, mutta useat muut olivat vielä hallinnassa Excel-taulukoissa.

Huolellisen suunnittelun ja testauksen ansiosta dev-ympäristössä, siirrot valmistuivat sujuvasti. Automaatiota myös rakennettiin nopeuttamaan manuaalisia tehtäviä ja uusia ideoita automaatiolle syntyi projektia tehdessä.

Infobloxia varten rakennettujen ohjelmistokehyksien tutkimiseen meni aikaa, mutta tulokset olivat loistavia ja hyödyllisiä, kun rakennetaan tietoa, joka koskee erilaisia automaatio-ohjelmistokehyksiä.

Avainsanat: Infoblox, VLAN, DDI, verkkoautomaatio

Contents

1 Introduction 1

2 DDI applications 1

2.1 Infoblox 2
2.2 Other DDI options 3
2.3 NetBox 3
2.4 phpIPAM 4
2.5 Windows Server IPAM 5

3 Setting up the test environment 6

3.1 First time configuration 11
3.2 Navigating Infoblox 16
3.3 Creating data for testing 20

4 Migrating VLANs 22

4.1 Infoblox VLAN schema 22
4.2 Mapping out data 25

5 Automation 26

5.1 WAPI 27
5.2 Getting information 29
5.3 Updating information 30
5.4 Linking VLANs and networks 30

6 Future improvements 32

6.1 Continuous development 32
6.2 Ansible and Terraform 33

7 Conclusion 33

References 35

List of Abbreviations

API:	Application programming interface
CLI:	Command-line interface
CSV:	Comma-separated values
DDI:	DNS, DHCP and IPAM
DHCP:	Dynamic Host Configuration Protocol
DNS:	Domain Name System
GUI:	Graphical user interface
HA:	High availability
HTTP:	Hypertext Transfer Protocol
HTTPS:	Hypertext Transfer Protocol Secure
IP:	Internet Protocol
IPAM:	IP Address Management
JSON:	JavaScript Object Notation
NIOS:	Network Identity Operating System
POST:	HTTP request method
PUT:	HTTP request method
REST:	Representational state transfer

VLAN: Virtual local area network

VM: Virtual machine

WAPI: Infoblox specific REST API

XML: Extensible Markup Language

1 Introduction

This study is the result of a project for a company where the management of all VLANs were migrated to Infoblox. Previously, VLANs were scrubbed through APIs from switches to a website and kept in excel spreadsheets. Additionally, Infoblox was already used to store network information such as DNS records and used networks.

Another goal of the project was to automate manual procedures such as creating new VLANs and linking VLANs to the networks already stored in Infoblox.

2 DDI applications

Traditional data storage for network information is excel or other types of manual applications. You can write down where every network is used and where each VLAN is being used.

An IPAM application can make this work easier and more scalable. In excel, scrolling through hundreds of rows will not only slow down excel but also slow down the work. IPAM applications are designed to make the process seamless and automatable.

On the other hand, traditionally DHCP servers and DNS servers are hosted on servers and often separately. Whether they’re a Linux or Windows server, manual work is needed to make changes or to have any integrations.

A DDI application combines DHCP, DNS and IPAM services and centralizes them. With a DDI application, you can make changes to the IPAM such as changing the subnet of a site, and the DDI application will automatically update the DHCP pools and DNS records. This reduces manual labour and makes

management much easier. Rather than logging into 3 different systems and making changes in each one, changes are only made in one system.

2.1 Infoblox

Infoblox is a DDI application with many extra features. It is the most used DDI application in enterprise networks and held a large 49.9% market share in 2015 according to a study done by Gartner as seen in figure 1. [1.] There were no newer public market studies available at the time of this study, but Infoblox is still the market leader by a large margin. A notable newcomer not mentioned in the study is NetBox which is explored in chapter 2.4.

For the scope of this study, only the VLAN and network management sections of Infoblox are touched upon. Just like keeping track of IP addresses and networks, Infoblox can also keep track of VLANs.

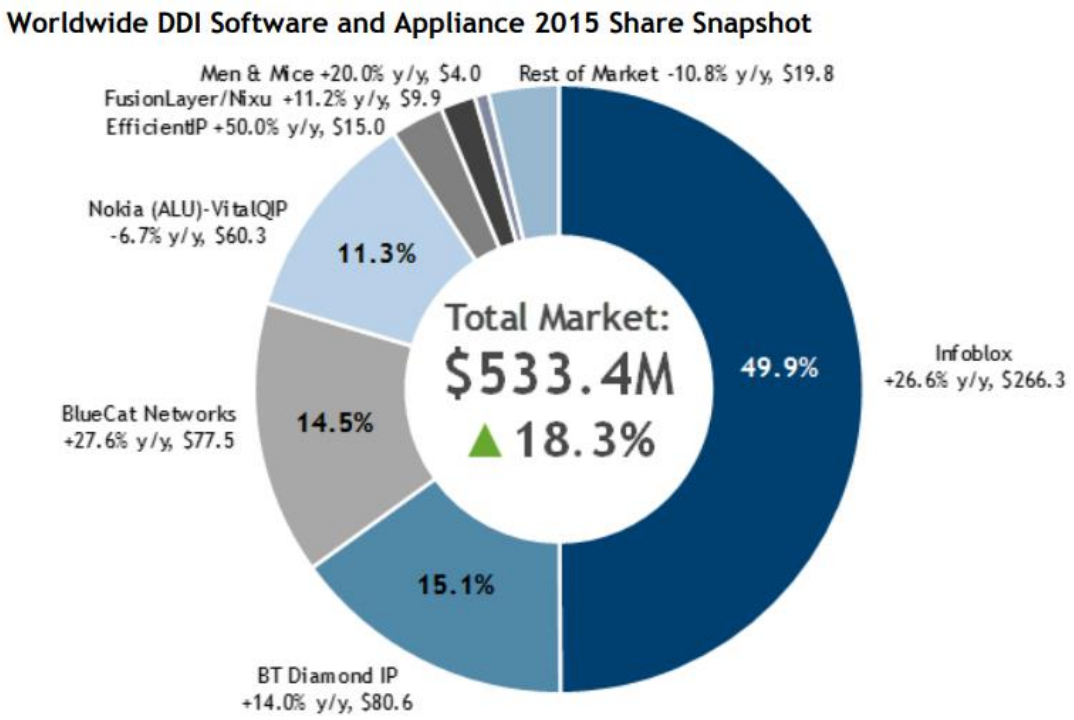

Figure 1. DDI market share in 2015 [1.]

2.2 Other DDI options

Even though using other DDI applications was not an option for this project, I wanted to find other options for different use cases. Infoblox is a great choice, but the steep licensing costs make it not a suitable application for smaller organizations.

Many different IPAM applications exist but only DDI solutions will normally offer VLAN management. For smaller use cases where VLAN tracking is not a requirement, something like a Windows server with the DNS, DHCP and IPAM roles installed will work fine.

In the scope of this project, Infoblox was chosen as it was already in use. If cost was not in consideration, Infoblox would still be my first choice of DDI applications. They've stayed as the market leader for many years for good reason, they keep improving the product and make it stay relevant with adding new features. The cost is also not that overpriced, enterprise support is costly no matter the product.

2.3 NetBox

NetBox is an open-source modelling and documenting application for modern networks designed by DigitalOcean. [2.] While Infoblox provides ready services such as DHCP, DNS and TFTP along with others, NetBox is normally only for documenting information. They provide a beautiful user interface with very extensive APIs in the form of REST APIs and a GraphQL interface. NetBox can be used for free with an on-premises solution, but enterprise support is limited to subscription-based licenses.

While NetBox is by itself just a modelling and documentation application, plugins are available for different services like DHCP and DNS to make it a DDI application.

NetBox would be a prime candidate when only documentation is needed. In this project, only the VLAN management and IP address management (IPAM) are touched upon and NetBox offers both features so this project could have also been implemented on NetBox. Figure 2 shows the VLAN management interface of NetBox.

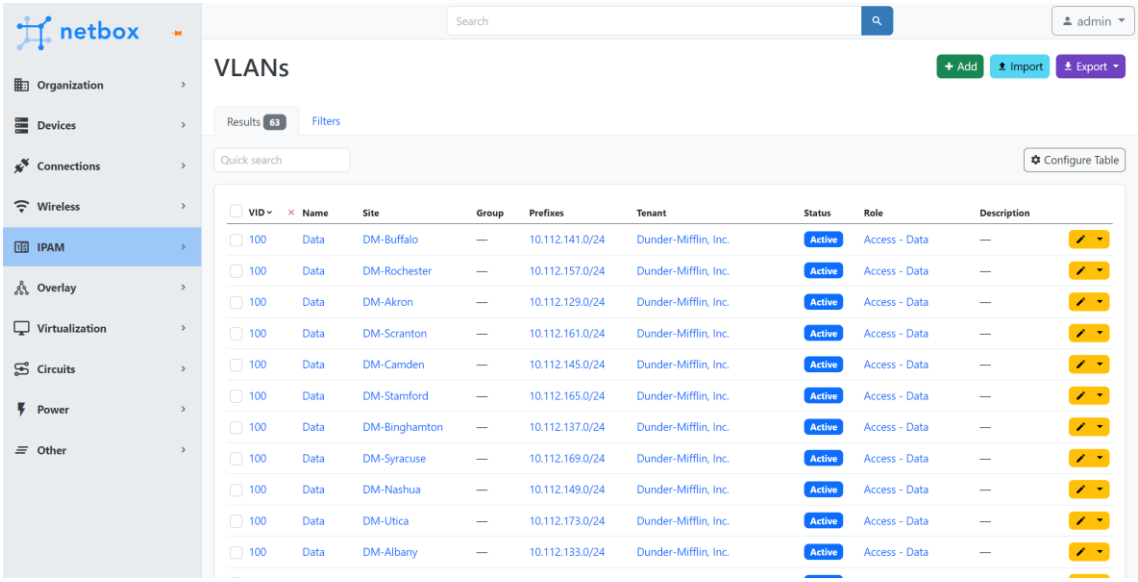

Figure 2. NetBox GUI demo [3.]

2.4 phpIPAM

PhpIPAM is another open-source DDI application. It is another great alternative for smaller networks. There is no official support other than community support on GitHub, therefore it is not suitable for larger enterprises.

It does have a great interface and a good API and therefore would have been suitable for the project. Figure 3 shows the VLAN management interface of phpIPAM.

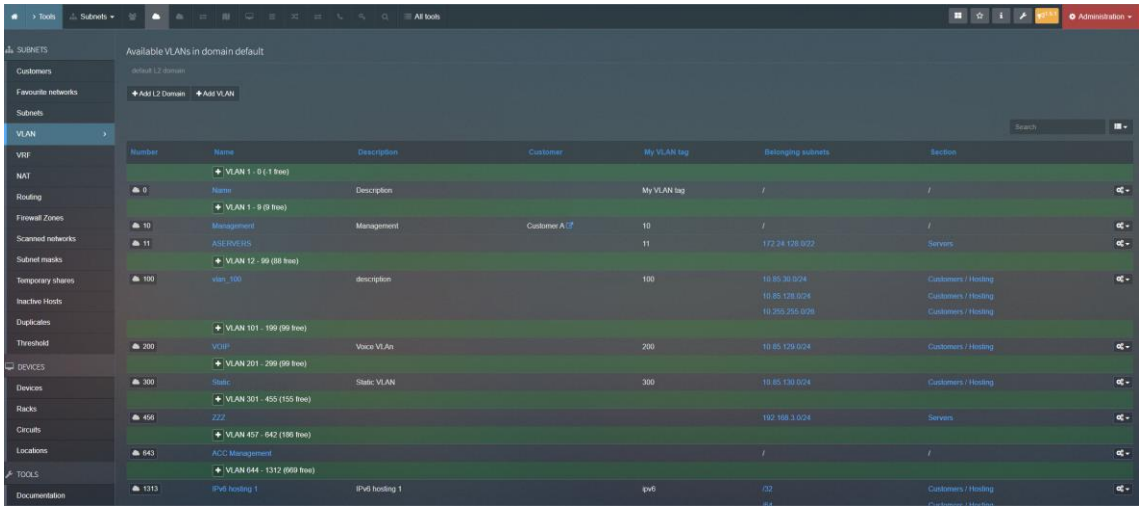

Figure 3. phpIPAM GUI demo [4.]

2.5 Windows Server IPAM

As many networks already make use of Windows Servers for various services, it might be a worthwhile consideration to use the Windows Servers IPAM role. From the other applications considered here, Windows Server is the least flexible and scalable. While it does offer basic IPAM functionalities, it has no linkage to DNS or DHCP services such as Infoblox. It also has no good API interfaces to interact with for workflow automations like NetBox and phpIPAM do.

If only a small amount of networks need to be kept track of and Windows Servers are already used, then the Windows Server IPAM can be a great springboard for testing out IPAM applications in general and to decide whether a larger investment into a specific application would be worthwhile.

3 Setting up the test environment

There are several ways to run Infoblox. There are physical appliances that can be installed like any other hardware. It can also be installed in the following public and private clouds.

- Public Cloud
  - AWS
  - Microsoft Azure
  - Google Cloud
- Private clouds
  - Nutanix
  - OpenStack
  - VMware

The easiest way, however, to run Infoblox is to virtualize it. Infoblox supports most hypervisors like Microsoft Hyper-V, Nutanix AHV, OpenStack KVM, VMware ESXi, and Citrix XenServer. For ease of use, I decided to run a virtualized server through VMware Workstation Player.

For virtualization, Infoblox offers a ready .ova file that can be imported straight into your choice of virtualization application. An .ova file is a virtual machine that has been packaged into one file, multiple virtualization applications such as VMware Workstation and Oracle VM Virtualbox can import it.

When importing, you may decide on which appliance to virtualize as shown in figure 4. The regular Infoblox appliances are the Trinzic series. The trinzic series is further divided into 4 categories, the 800, 1400, 2200 and 4000 series. The higher tiers offer higher performance with more DNS queries and DHCP leases per second for example. Faster network ports and multiple power supplies are also included with the higher tier appliances. All the models are available, but as the performance of the virtual machine is dependent on the resources allocated to it, the choide of model does not matter much. I picked the TE-825 as it’s one of the lower tier models and the default values for the

virtual machine are more reasonable than the higher tier ones. The load on the virtual machine will be minimal with the testing.

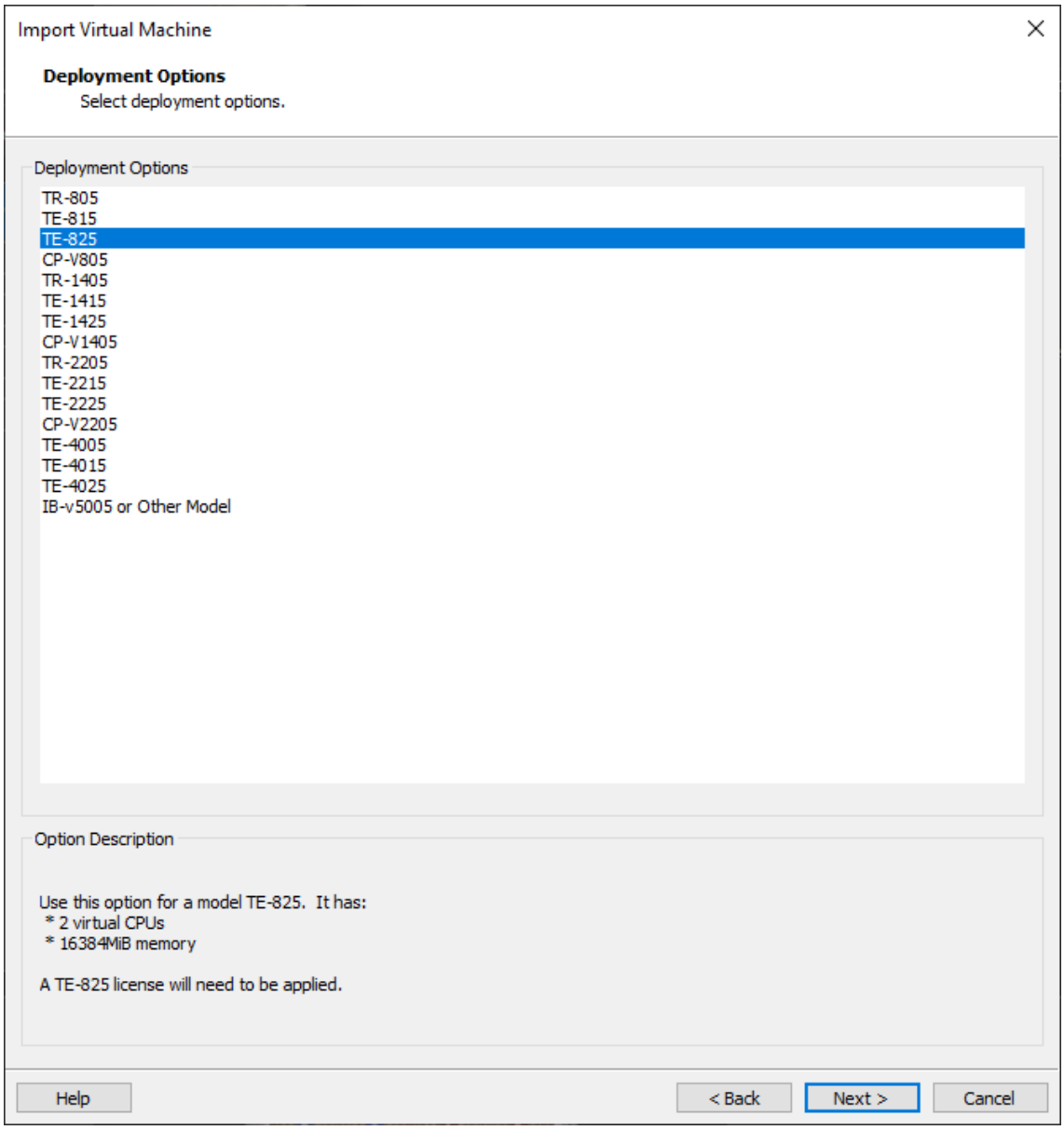

Figure 4. Virtual appliance deployment model selection

For testing purposes, not much needs to be changed from the default settings of the VM. The TE-825 comes with 2 virtual CPUs and 16GB of memory. These are both overkill for testing purposes so lowering the memory usage and going to 1 CPU is acceptable. There is no need to lower them, but my host machine has 24GB of memory so using 16GB on a virtual machine makes my computer

slow down. Therefore, I lowered the memory usage to 8GB as shown in figure 5.

Depending on the host configuration, network settings may need to be changed. For my home network, running the virtual machine network adapter in bridged mode works well. I can set a separate IP address for the virtual machine, and it can be reached independently from wherever in the network.

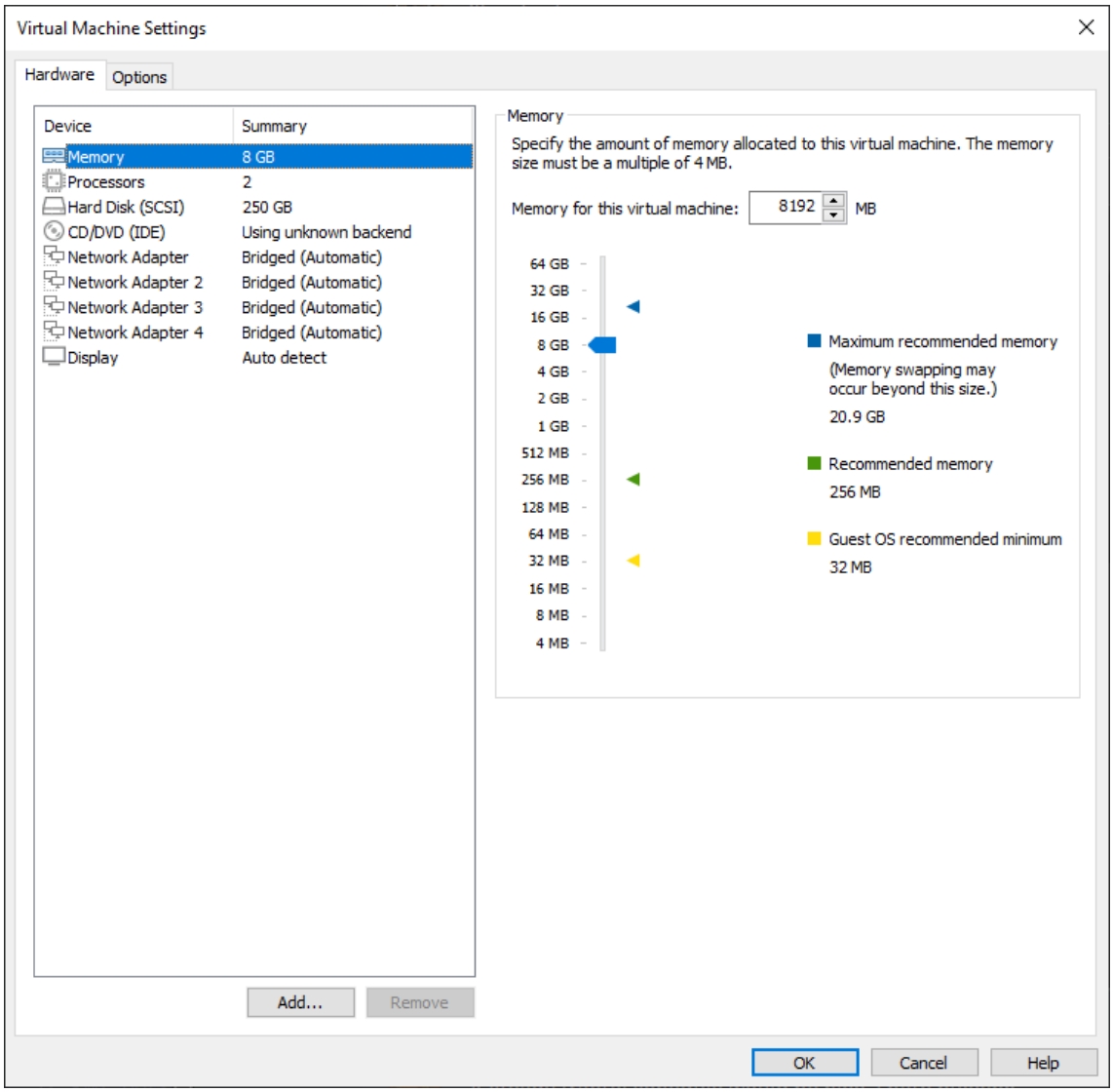

Figure 5. Virtual machine settings

After importing the virtual machine, a few settings must be set on the CLI before being able to access the WebUI. To access the CLI, default credentials

admin:infoblox are used. If running in a production environment, the default credentials should be changed. An IP address needs to be set to conform to the network it is running in. The default IP address of the virtual machine is 192.168.1.2/24 with the default gateway being 192.168.1.1.

As shown in figure 6, the IP address can be set with the “set network” command. The network that my virtual machine will be in is 10.0.100.0/24. Setting the network settings is displayed in figure 6. I opted not to configure IPv6 as it’s not relevant in this study. I did not make it a grid member as there is currently no grid master available.

Grid members are essentially multiple Infoblox appliances linked together. To have a grid, a grid master needs to be configured. As there are no Infoblox appliances installed prior to this machine, it cannot join a grid. It also cannot be configured to become a grid master through the CLI. Instead, I will be configuring it to become a grid master later in the web GUI.

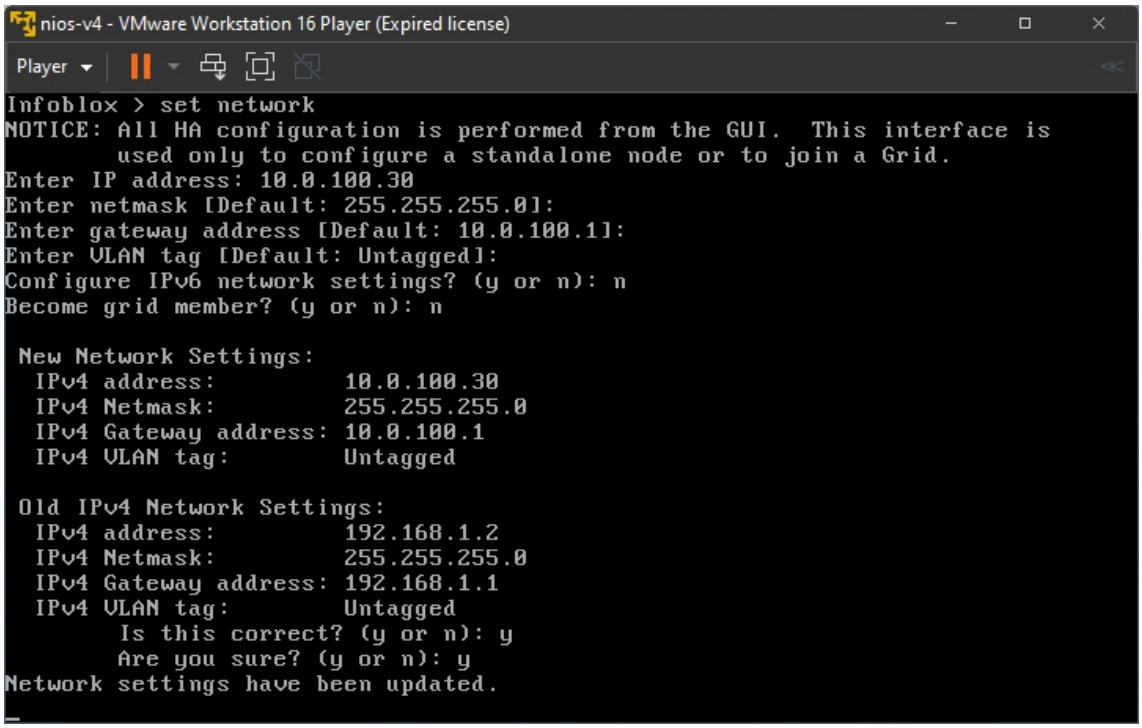

Figure 6. Configuring network settings

Licenses must be installed to any Infoblox service. Thankfully, Infoblox offers 60-day temporary licenses for testing environments or evaluating the products. To continue using a product after the 60 days, either a license must be bought for the service, or the appliance must be reinstalled. This is great for testing environments as you can reinstall the virtual machine after 60 days and losing data is not a big deal. Depending on the usage, different licenses must be installed. [5.] For my testing purposes, the following licenses were installed:

- NIOS License
- DNS Server license
- DHCP Server license
- Grid license

The NIOS license is the basic license for the appliance to work normally. The grid license is for managing grid settings and building HA with multiple grid members. The DHCP and DNS licenses enable their respective services. The licenses may be installed on the CLI with the command “set temp_license” as shown in figure 7.

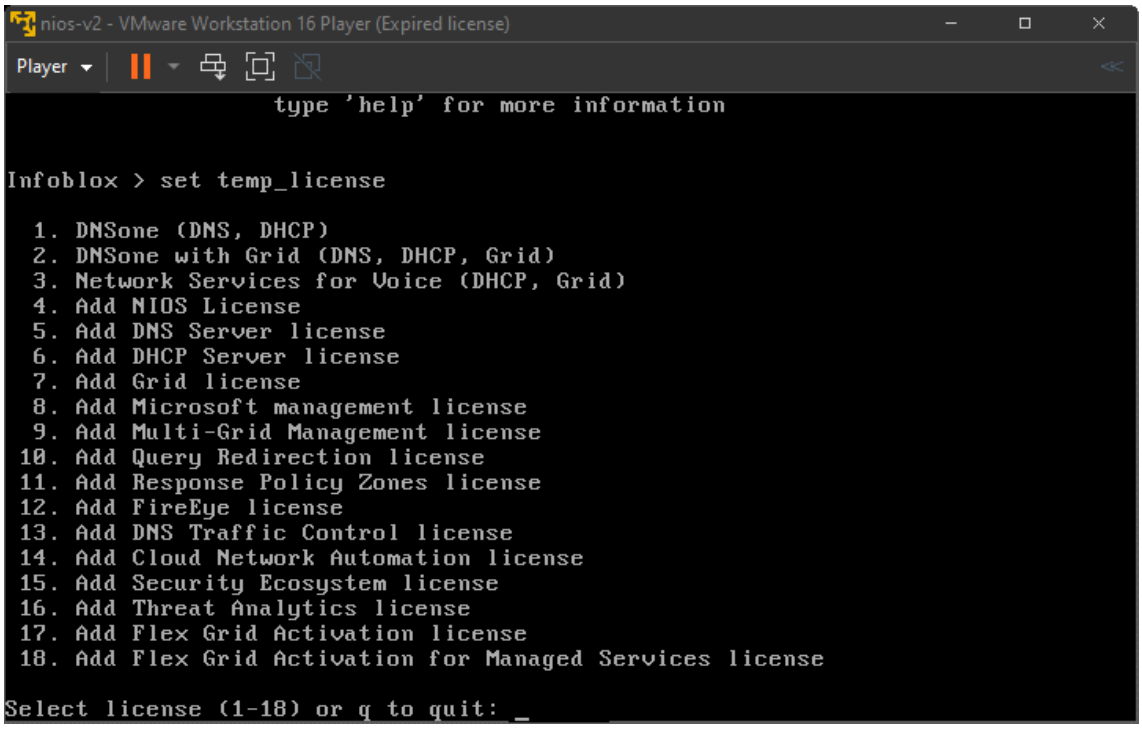

Figure 7. Infoblox CLI interface

3.1 First time configuration

After configuring the IP address and setting the licenses, the Infoblox GUI can be accessed through a browser depending on the IP address. In my case I set the IP as 10.0.100.30, therefore the GUI may be accessed at https://10.0.100.30 as shown in figure 8. The default credentials admin:infoblox also work on the GUI. [6.] As mentioned earlier, if running in production, changing the default credentials is highly recommended.

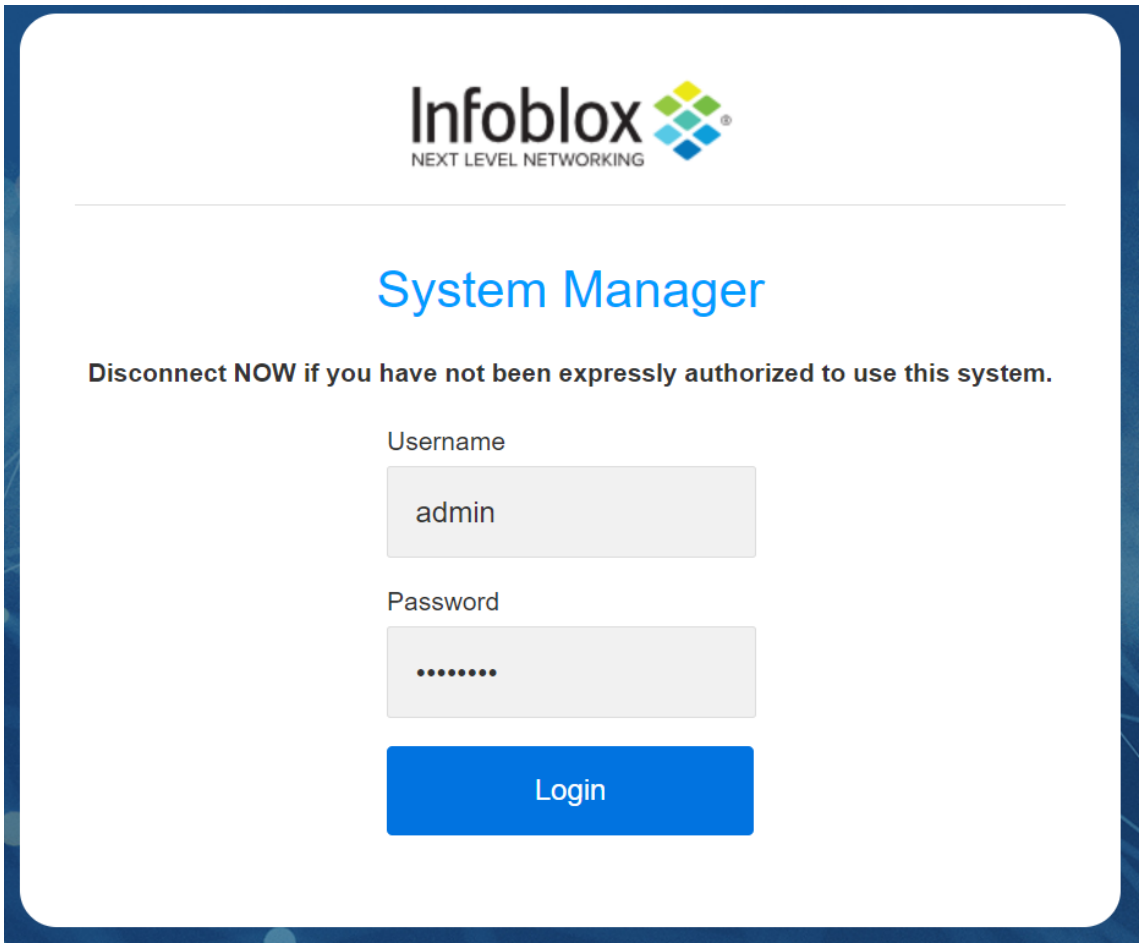

Figure 8. Infoblox login page

A few settings must be configured on the first login into Infoblox. A NIOS setup wizard will appear when logging in for the first time as shown in figure 9.

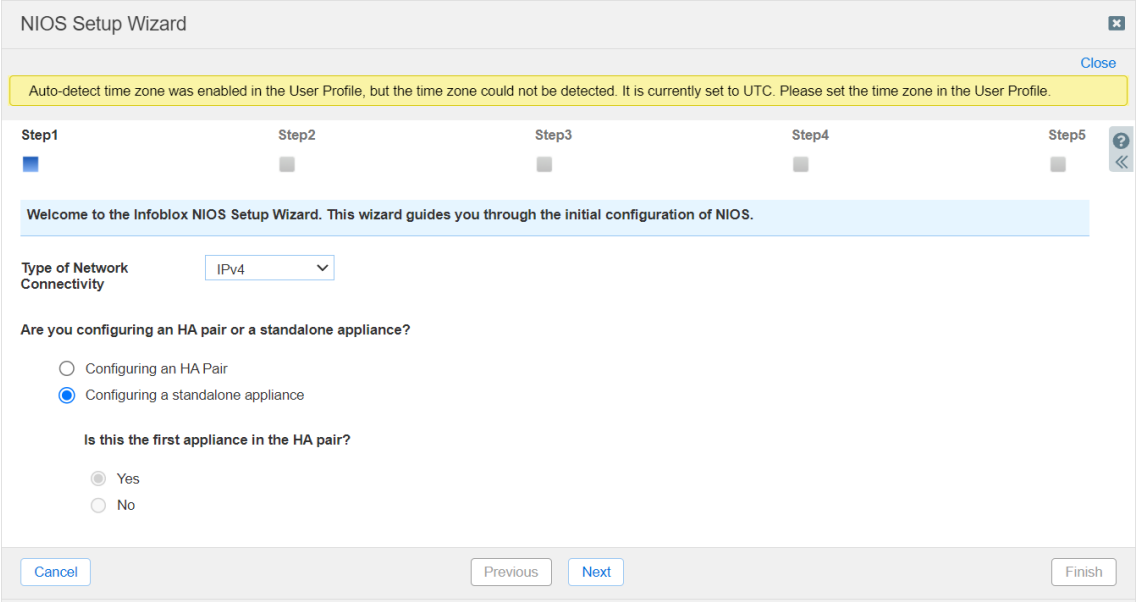

Figure 9. First time configuration step 1

In step 1, the network connectivity and High Availability are configured. I'm going with IPv4 for its ease of use and as there is no need for IPv6 especially when just testing out the application. As this will be the only grid member I'm going to install, no HA will be configured.

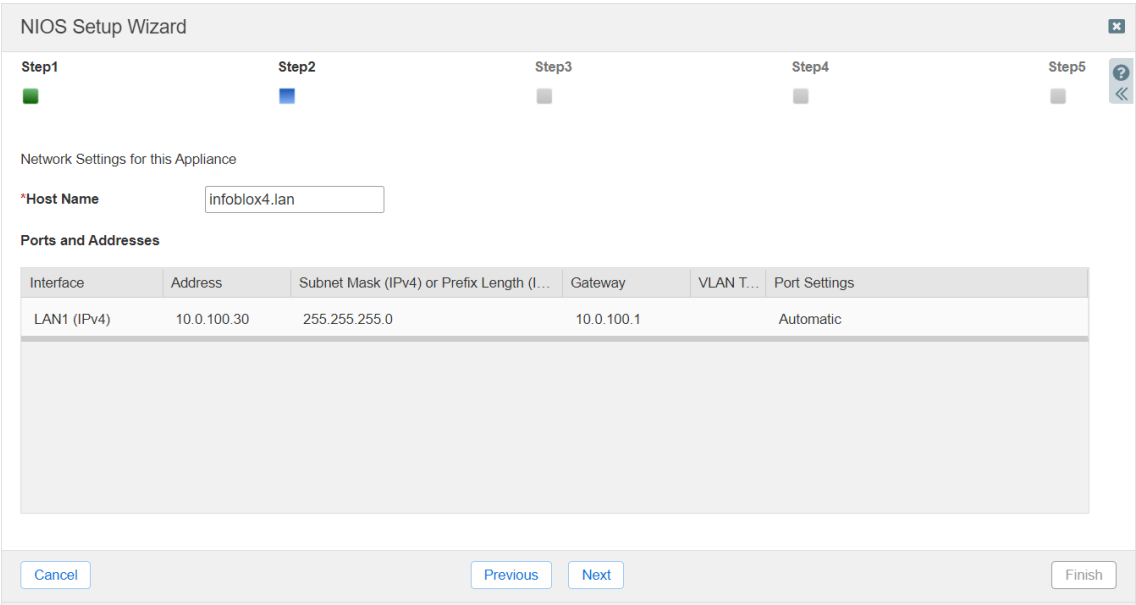

Figure 10. First time configuration step 2

Step 2 contains network settings as shown in figure 10. As I've already set the network settings through the CLI, nothing needs to be done regarding them. I will however change the hostname from the default infoblox.localdomain into infoblox4.lan to match the other devices in my home network.

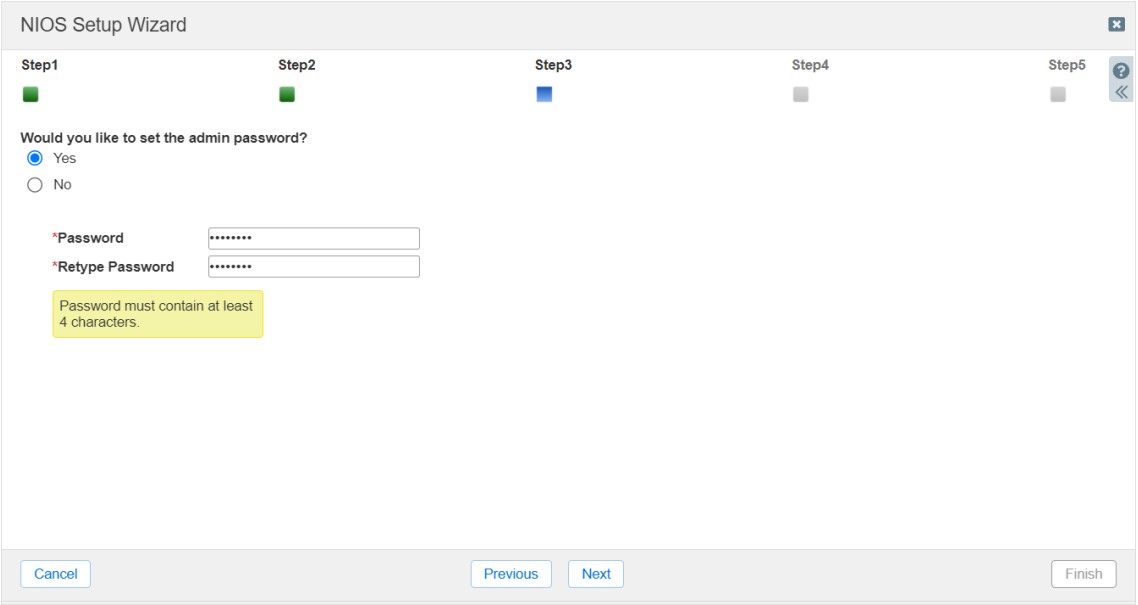

Figure 11. First time configuration step 3

As shown in figure 11, step 3 asks whether you want to change the default admin password. I will change it out of habit as default passwords are never good. Again, this is highly recommended to do in production environments or even in testing environments if any real data will be stored in the appliance.

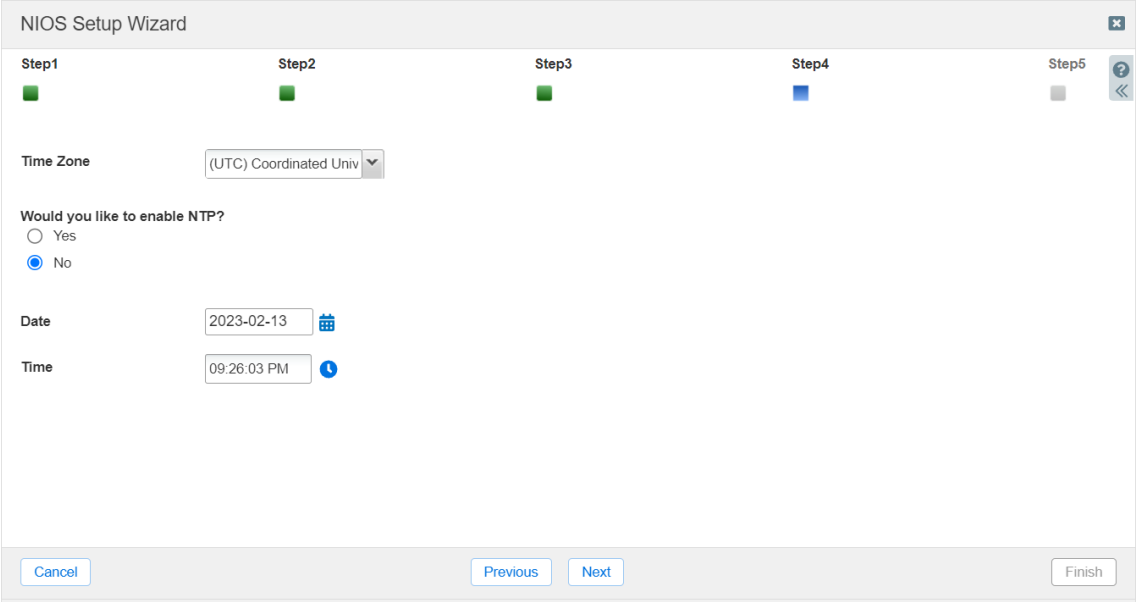

Figure 12. First time configuration step 4

As shown in figure 12, step 4 contains time settings. Accurate time for the appliance is critical in production in order to have accurate logging and auditing. It's also critical for security as most security protocols rely on correct time synchronization. It's not that important while just testing and therefore I've set the correct time zone but opted not to enable NTP.

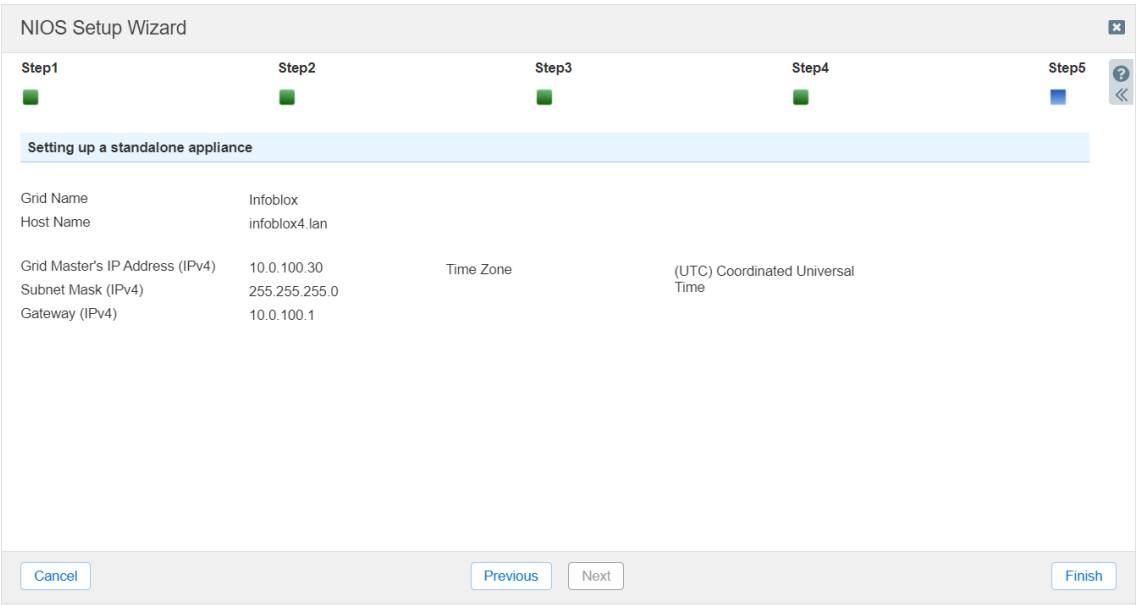

Figure 13. First time configuration step 5

As shown in figure 13, step 5 is a review of the configurations made. With this finished, Infoblox is now ready to be used.

3.2 Navigating Infoblox

There are 5 tabs in Infoblox: Dashboards, Data Management, Smart Folders, System and Administration. Figure 14 shows the default login page of Infoblox. The dashboards tab is open and the status windows there will show statistics of the appliance such as the system status, network usage and active WebUI users. Dashboards may be customized for needed usage. Different tasks may also be added to dashboards for quick access such as creating a new VLAN or adding a DNS host record.

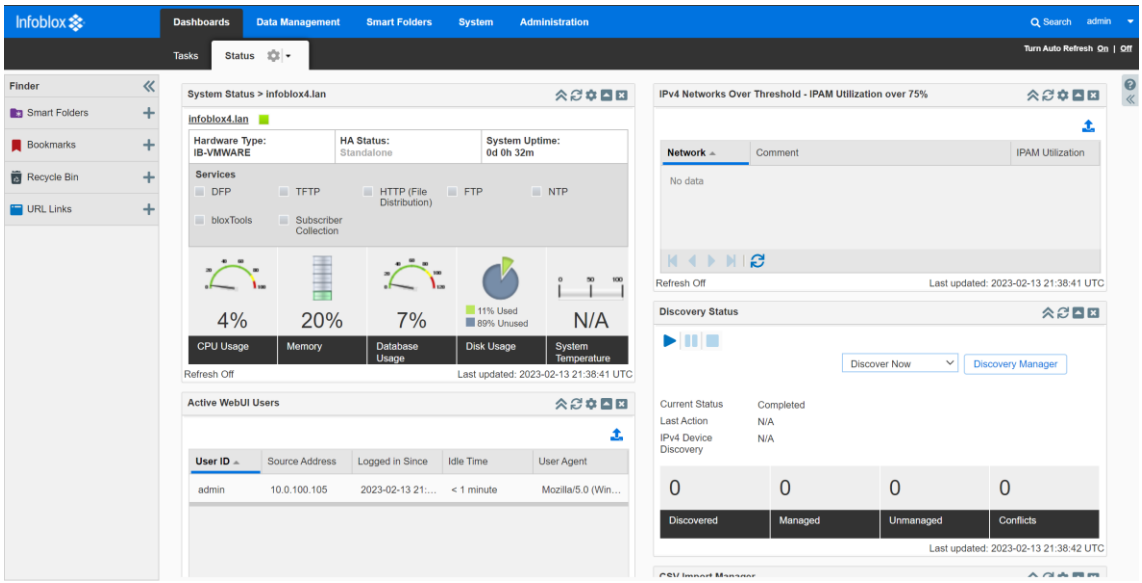

Figure 14. Infoblox dashboard tab

The Data Management tab is shown in figure 15. This is where the actual usage of Infoblox comes in. By default, it includes the IPAM, VLANs, Super Host and File Distribution tabs. Different tabs like DNS or DHCP may be added with licenses, but they are not available by default.

The only relevant tabs in this study will be the IPAM and VLAN tabs. Networks are added to IPAM and VLANs are added to the VLAN tab. These will both be further discussed in later chapters.

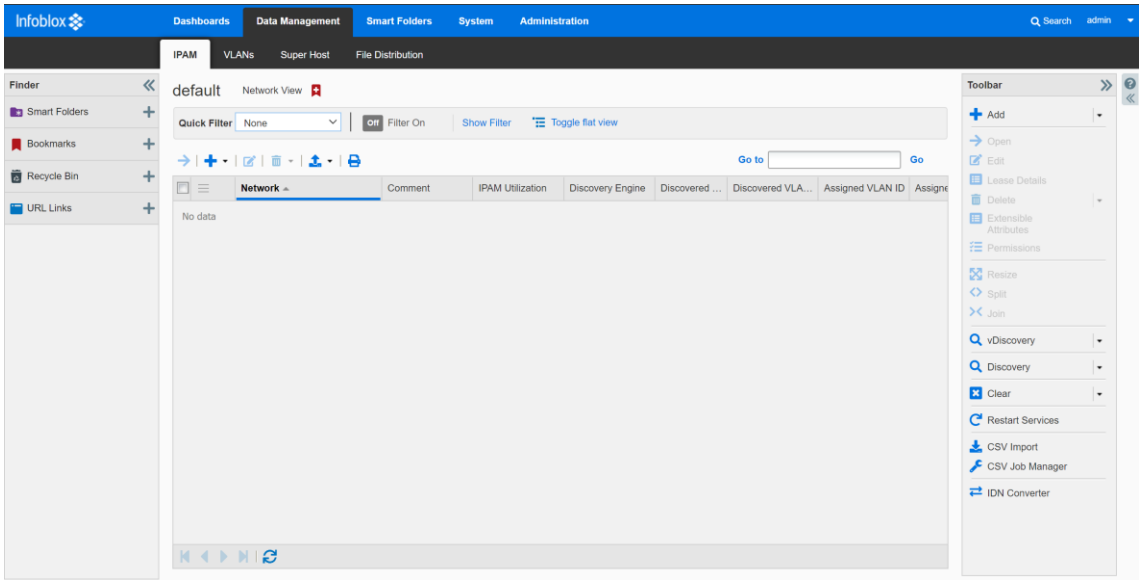

Figure 15. Infoblox data management tab

The Smart Folders tab is shown in figure 16. It is used for filtering data with predefined attributes. For example, if a network contains different networks in multiple countries, the networks may be labelled with their respective countries with extensible attributes. After labelling, the networks may be grouped in Smart Folders by filtering for those attributes. With this example you may create folders for “Finnish networks” and “Swedish networks” for example.

Smart folders may be used for any objects within Infoblox. They may be filtered and grouped together depending on needs. Some default smart folders that exist are “Active Directory Sites” and “Apple Mac OS Devices”.

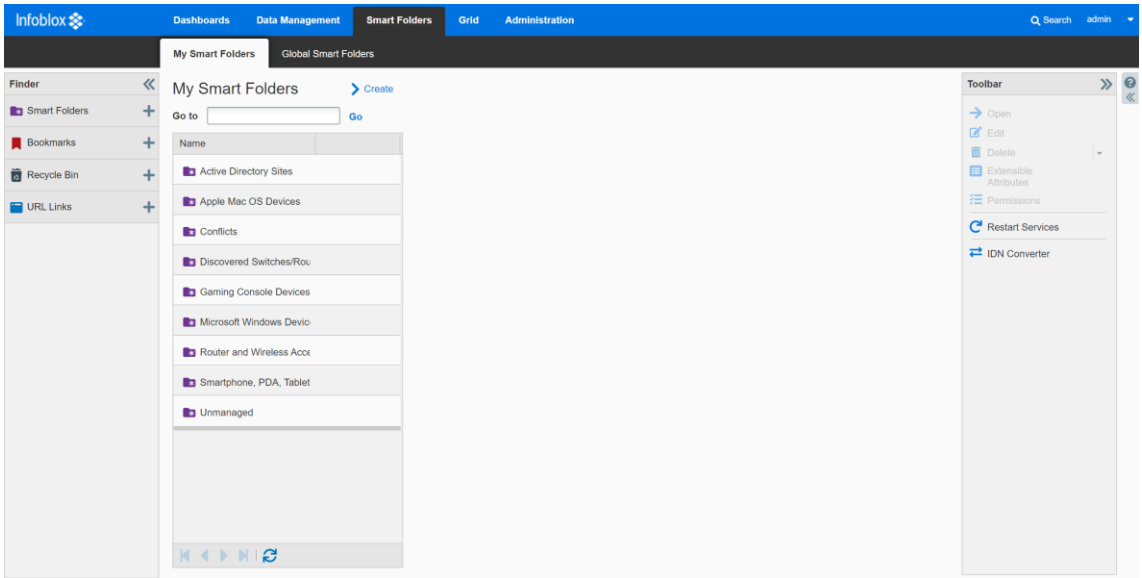

Figure 16. Infoblox smart folders tab

The Grid tab is shown in figure 17. It is used for managing all the grid members. As mentioned before, a grid master must be set in all Infoblox environments and further grid members may be added. All the grid configuration is done in the Grid tab.

Different roles may be added for grid members to balance the load of the network or guarantee high availability. For example, if Infoblox managed the DNS of a network, it would be smart to have 2 grid members in different locations handling the DNS service. The data is replicated between them and they both respond to DNS requests but if something happens to one node, the other will still continue to serve the network.

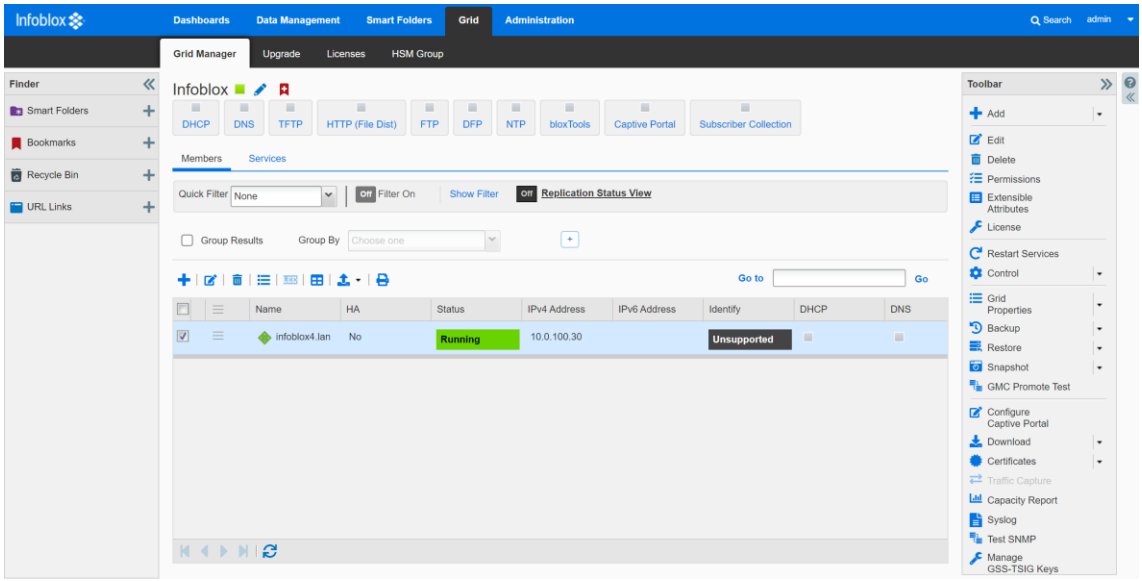

Figure 17. Infoblox grid tab

Lastly, the administration tab is shown in figure 18. It contains all the individual configurations of the grid members and other administrative data such as logs and extensible attributes. Here is where new users may be created, and permissions can be configured for each user. All logs can also be found in the log tab. For multiple grid members, you may choose which member to send the logs to. The log locations are configured in the different services settings, so DHCP log locations need to be configured in the DHCP settings of the members using it.

Often to make the environment centralized, all logs will be sent to the grid master where they can be inspected easily. Logging into multiple grid members to inspect different logs creates unnecessary complexity. Extensible attributes are also configured in this tab, they will be further reviewed in later chapters.

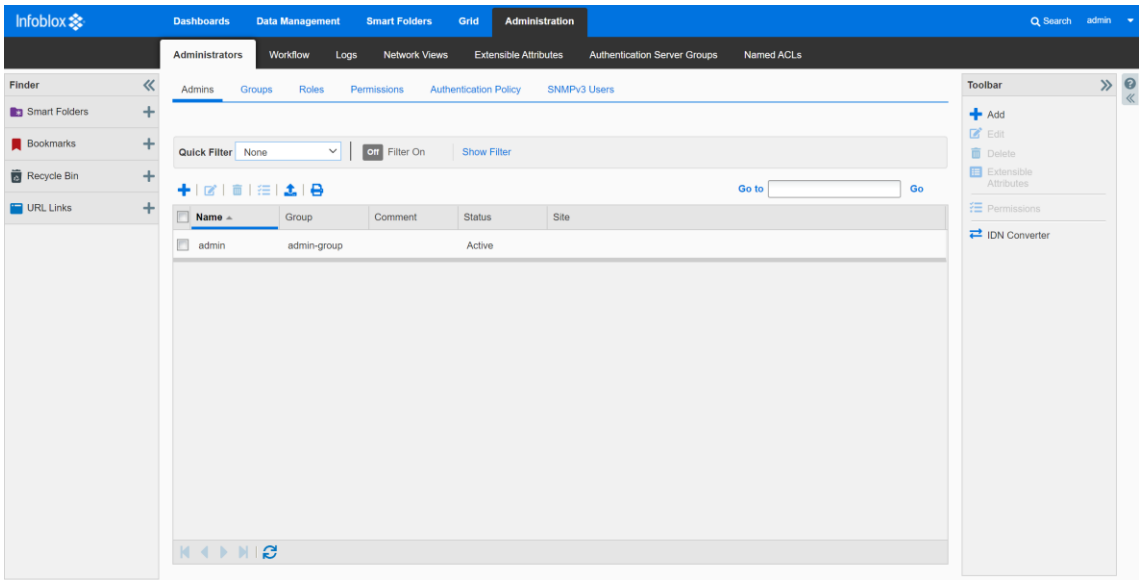

Figure 18. Infoblox administration tab

3.3 Creating data for testing

VLAN management in Infoblox consists of three parts. [7.] VLAN views, VLAN ranges and VLANs themselves. Normally when managing VLANs, you cannot have overlapping VLANs.

A default VLAN view that contains the full range of 1-4094 VLAN ids is created by default. VLAN views make it possible to have multiple VLANs with the same ID managed by Infoblox. [8.] As can be seen in figure 19, extra VLAN views internal and external have been created with the same range as the default VLAN view. This allows multiple VLANs with the same id to co-exist in Infoblox depending on the needs of the organization.

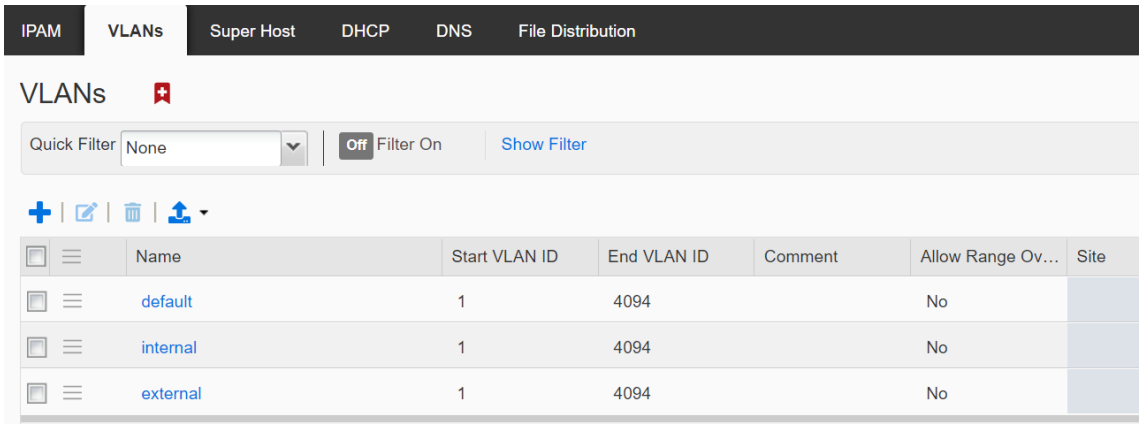

Figure 19. Vlan views

VLAN ranges reside in a VLAN view and are a range of VLANs as indicated by the name. [9.] A test range of 500-999 can be seen in figure 20.

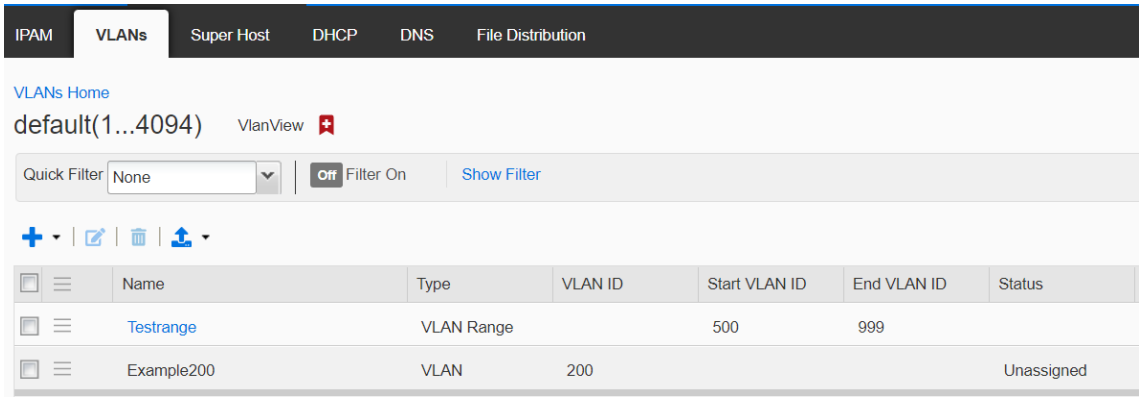

Figure 20. VLAN example

Lastly, VLANs can either be configured directly into a VLAN view or placed in a VLAN range. [10.] In figure 20, VLAN 200 has been configured directly in the default VLAN view. Meanwhile in figure 21, VLAN 600 has been configured in the test range created earlier. As the range of “Testrange” is 500-999, the VLAN configured must be within that range.

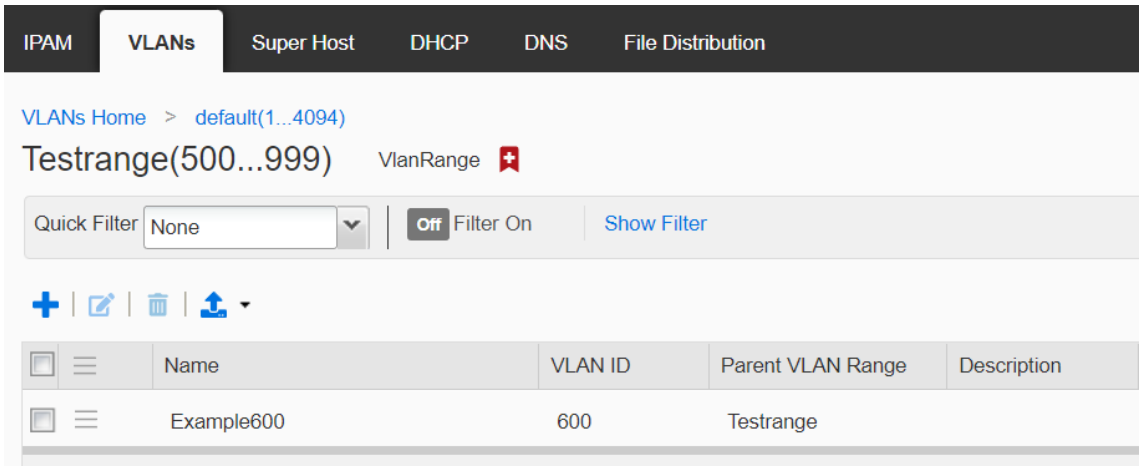

Figure 21. VLAN range example

4 Migrating VLANs

4.1 Infoblox VLAN schema

The main goal of the project was to migrate VLANs from a large excel spreadsheet. The spreadsheet contained multiple ranges of VLANs with different attributes. The requirements to add a VLAN into Infoblox are the VLAN ID as well as a name for the VLAN. Multiple ranges in the spreadsheet contained additional attributes such as which customer it belonged to. This was solved through extensible attributes.

As seen in figure 22, a VLAN has multiple attributes you can configure by default, these include:

- name
- whether the VLAN is a reservation or in use
- description
- contact
- department
- status
- comment
- what networks it is assigned to.

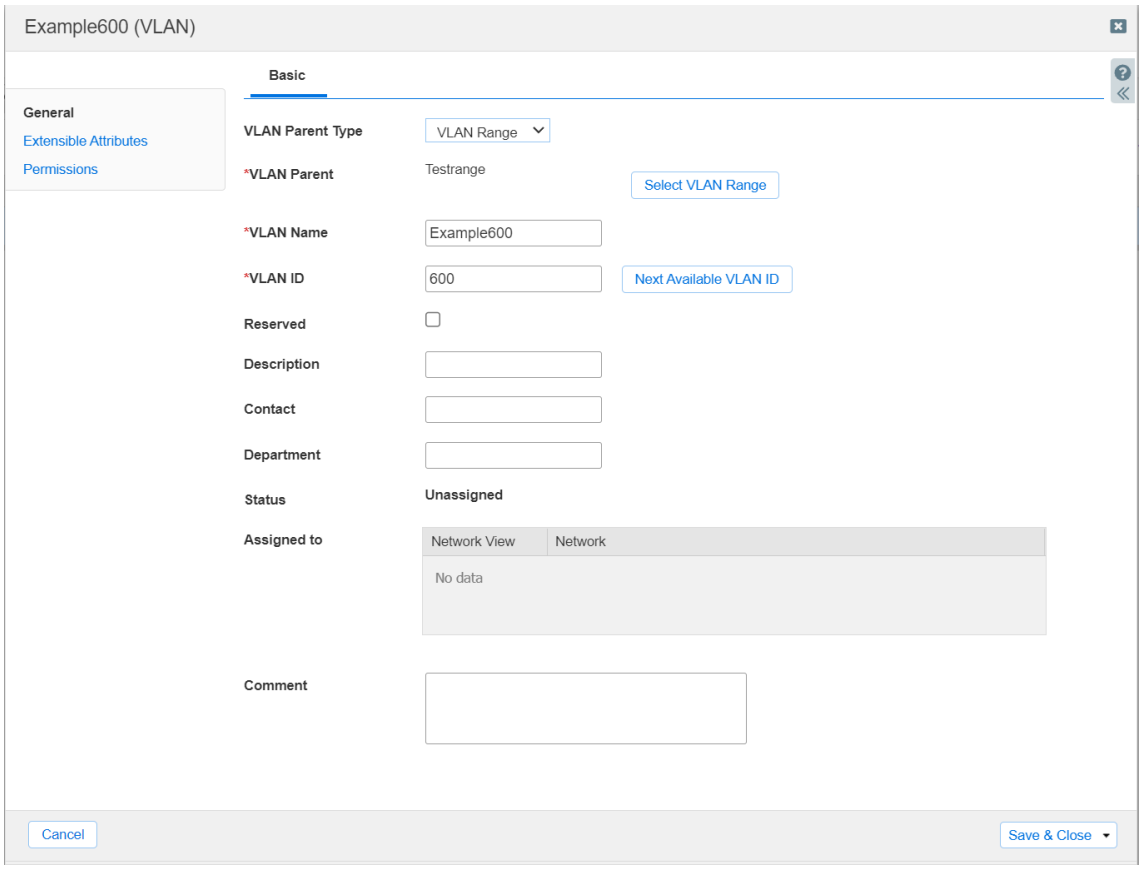

Figure 22. VLAN settings

Additionally, extensible attributes may be added to the VLAN configuration. These are essentially custom attributes that can be made for whatever needs may arise. [11.]

Figure 23 showcases the creation of an extensible attribute. In the default example, I've created a university string attribute that is restricted to only VLANs. I've left the default value empty and set it as an optional attribute. It could also be made required. If it were required, every VLAN would have to have the university attribute configured.

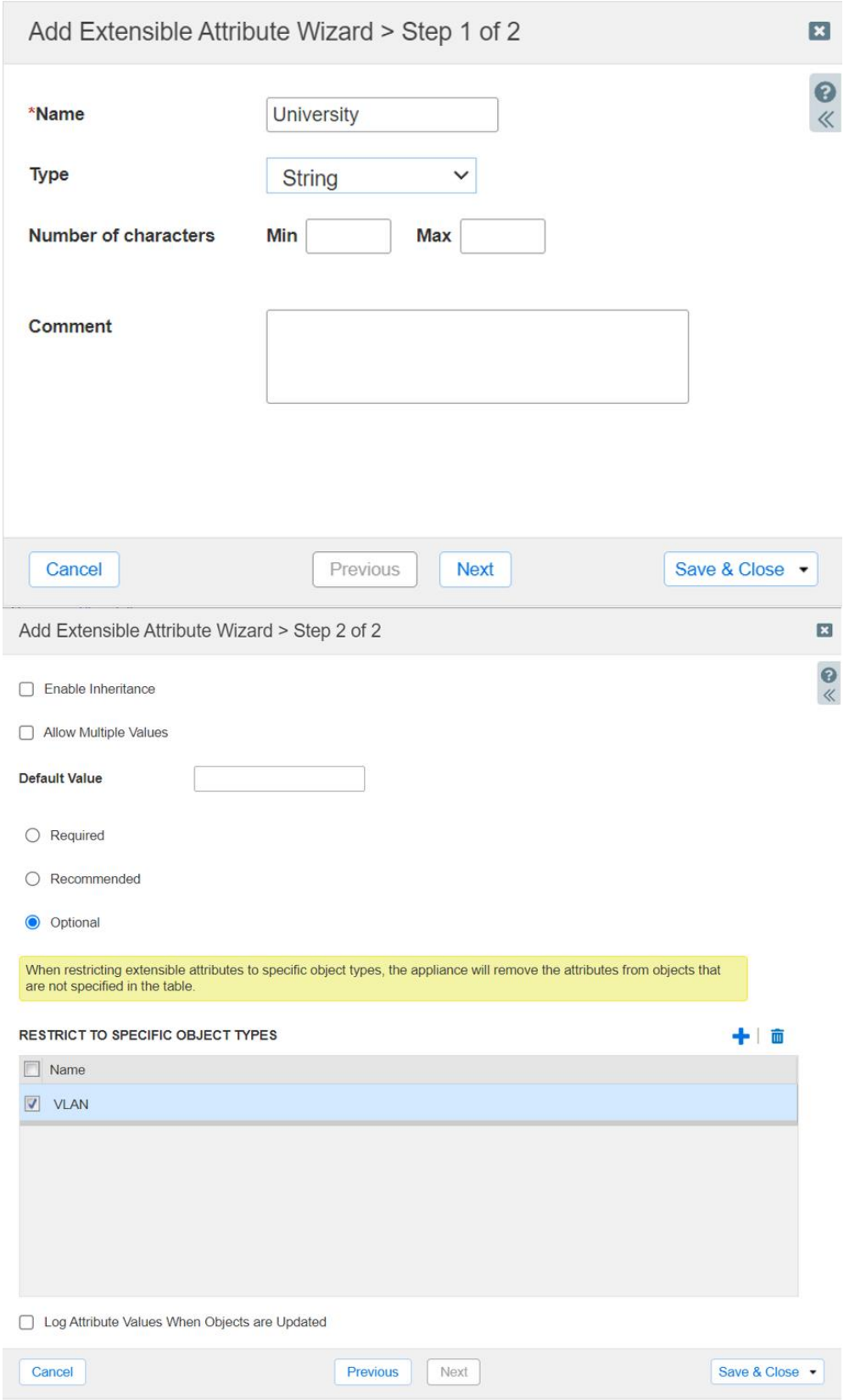

Figure 23. Extensible attribute creation

After creating the attribute, it may be freely assigned to any VLAN. In figure 24, I've configured Metropolia as the university attribute for VLAN 600. Extensible attributes may be created and used for essentially all objects within Infoblox. In the scope of VLANs, all 3 important objects may have them, VLAN views, VLAN ranges and VLANs themselves.

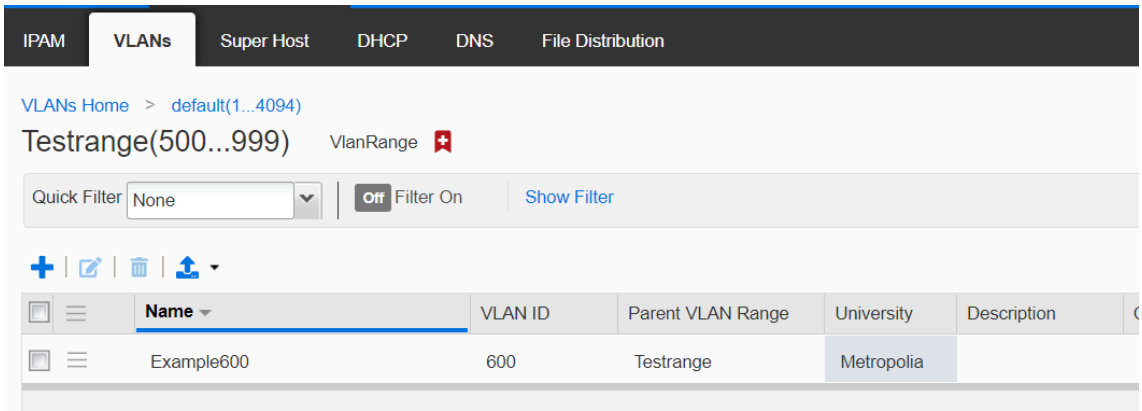

Figure 24. VLAN with an extensible attribute

4.2 Mapping out data

To begin the migration process, meetings were held to decide on relevant data as well as a schema to hold the VLANs. Overlapping VLANs are used so multiple VLAN views were created.

There were 3 ways to migrate the data. Manually where each VLAN entry is created separately. This was out of consideration due to the sheer number of VLANs. The two viable options were importing via a CSV file [12.] or adding them through the REST API. It was decided to use a CSV file as it could easily be made from the existing excel spreadsheet.

To create a correct form of CSV file, an export is first made from Infoblox and opened in excel. The columns are then filled with relevant data from the spreadsheet with the actual data. The CSV is saved normally and can then be imported into Infoblox. Thanks to this, the actual importing of the VLANs was done in a few hours after all the planning was done. Figure 25 shows an example CSV file opened in excel.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	header-staticvlan	id*	_new_id	name*	_new_name	parent*	_new_parent	comment	contact	department	description	reserved	EA-University
2	staticvlan	###		Example600		Testrange,500,999,default.1.4094						False	Metropolia

Figure 25. CSV file for importing.

5 Automation

Another goal of the project was to create automation around VLAN management in general. A process that was immediately recognized was the linking of networks and VLANs automatically. As Infoblox is also used to manage networks, it only makes sense to link the networks used by VLANs into the VLAN objects themselves. This makes it possible to immediately know what network and VLANs are linked to each other by just browsing their relevant listings.

Figure 26 shows the network object view where you can see the VLAN that has been assigned to the network. Meanwhile, figure 27 shows which network VLAN 600 has been assigned to. [13.]

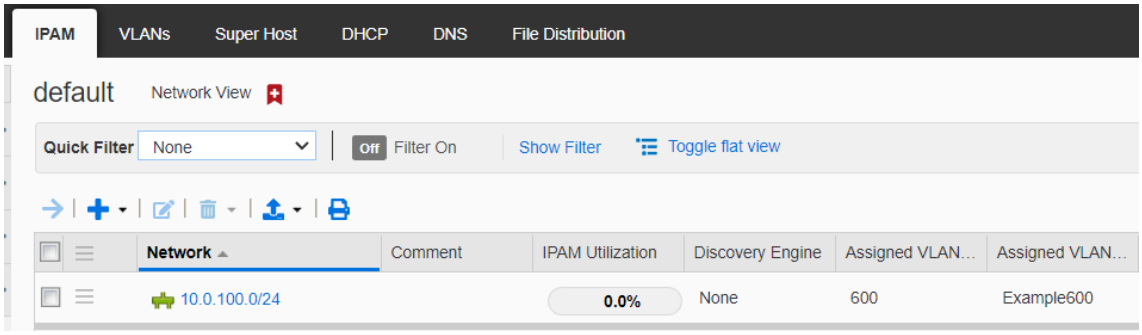

Figure 26. Network linked to VLAN

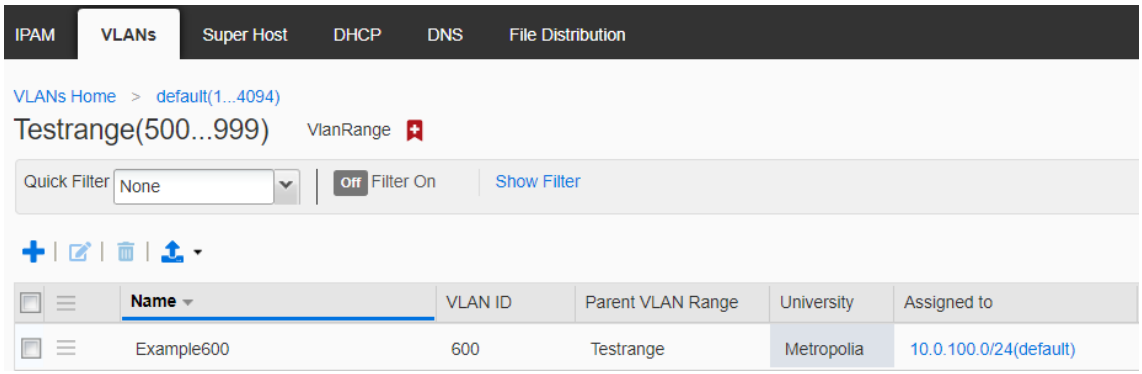

Figure 27. VLAN assigned to a network

Manually configuring the links between VLANs and networks can become very time consuming in enterprise networks where there are thousands of VLANs and networks. This is where the API comes in handy.

5.1 WAPI

WAPI is the REST API interface for Infoblox. [14.] It can be used to get information and configure anything through HTTPS requests. It supports input and output in JSON and XML. Authentication can be done through HTTP basic authentication or using cookies.

The WAPI has very extensive documentation that seems to include essentially every aspect of Infoblox. For some reason, there is not much community information on the API or troubleshooting posts found online though. This made troubleshooting and figuring out certain syntaxes difficult as the documentation does not contain a lot of examples usages. [15.]

```
import requests
from requests.auth import HTTPBasicAuth


url = "https://10.0.100.60/wapi/v2.12/vlan"
query = {
    "id":"600",
    "_return_fields":"name,id,parent,extattrs"
}

response = requests.get(url, params=query,
auth=HTTPBasicAuth(username='admin', password='infoblox'), verify=False)
print(response.text)
```

Listing 1. WAPI request for VLAN 600

Listing 1 showcases python code to request information from VLAN 600. The query includes the id and _return_fields variables. Id refers to the VLAN id and _return_fields dictate what the API should return. In this case I want the name, id, the parent which in this case is the 500-999 VLAN range and the extensible attributes set for the VLAN.


```
[
    {
        "_ref":
"vlan/ZG5zLnZsYW4kLmNvbS5pbmZvYmxveC5kbnMudmxhbl9yYW5nZSRkZWZhdWx0LjEuNDA5NC5U
ZXN0cmFuZ2UuNTAwLjk5OS42MDA:default/Testrange/Example600/600",
        "extattrs": {
            "University": {
                "value": "Metropolia"
            }
        },
        "id": 600,
        "name": "Example600",
        "parent": {
            "_ref":
"vlanrange/ZG5zLnZsYW5fcmFuZ2UkZGVmYXVsdC4xLjQwOTQuVGVzdHJhbmdlLjUwMC45OTk:def
ault/Testrange/500/999"
        }
    }
]
```


Listing 2. API return values for listing 1 request

The return result of the API is showcased in listing 2. The _ref value is the unique reference value of this specific VLAN. The fields requested in the _return_fields variable can be seen in the response. The name, the VLAN id, the extra attributes which contain the University value and the parent of the VLAN which in this case is the VLAN range of 500-999 created earlier.

5.2 Getting information

The WAPI is great, but it is a very low level of requesting information from the API. It directly deals with HTTP requests and all logic must be coded manually. Directly dealing with HTTP requests is fine for smaller projects but a better alternative exists in the form of Infoblox-client. [16.]

Infoblox-client is an open-source python framework built on top of the WAPI and maintained by Infoblox. It has a very basic documentation but contains functions to directly obtain information and update it easily.

```
from infoblox_client import connector
from infoblox_client import objects

conn = connector.Connector({
     'host': "10.0.100.60",
     'username': 'admin',
     'password': 'infoblox'
})
vlan = objects.Vlan.search(connector=conn, id=600, return_fields=[
     'name',
     'id',
     'parent',
     'extattrs'
])
print(vlan)
```

Listing 3. Infoblox-client request for VLAN 600

As seen in listing 3, the code for requesting becomes much simpler. The framework objectifies all possible objects you could request from Infoblox. The returned objects are defined in the framework and have different fields depending on the type of object.

Finding out about this framework made work a lot easier. Dealing with HTTP requests in code is normally not that difficult but the reference system used in WAPI makes it, so you must keep track of the reference of the requested VLAN, the reference of the parent range and possibly even the reference of the parent VLAN view. Objectifying all of this makes it so the parents can be accessed as simply as “variable.parent”.

5.3 Updating information

With pure HTTP requests, the correct reference needs to be retrieved and then either a POST or PUT request may be made to the API with new values. With Infoblox-client, the object can be retrieved with search parameters, its values may be changed freely, and it can be updated with a function. Listing 4 shows an example of updating the comment field of a retrieved VLAN object.

```
from infoblox_client import connector
from infoblox_client import objects

conn = connector.Connector({
     'host': "10.0.100.60",
     'username': 'admin',
     'password': 'infoblox'
})
vlan = objects.Vlan.search(connector=conn, id=600, return_fields=[
     'name',
     'id',
     'parent',
     'extattrs'
])

vlan.comment = 'test'
vlan.update()
```

Listing 4. Updating VLAN 600 comment with Infoblox-client

With this method, multiple VLANs can be created or modified based on a dataset. The easiest way I found of updating multiple objects was to create a CSV file with the needed columns and loop through that with API requests.

5.4 Linking VLANs and networks

Linking VLANs and networks turned out to be simple after learning about the python framework. The longest part was creating the CSV file with the correct data.

```
NETWORK_NAME;NETWORK_SUBNET;VLAN
customer0;10.0.100.0/24;500
customer1;10.0.101.0/24;501
customer2;10.0.102.0/24;502
customer3;10.0.103.0/24;503
customer4;10.0.104.0/24;504
customer5;10.0.105.0/24;505
customer6;10.0.106.0/24;506
customer7;10.0.107.0/24;507
customer8;10.0.108.0/24;508
customer9;10.0.109.0/24;509
```

Listing 5. Example customer data CSV

```
import csv
from infoblox_client import connector
from infoblox_client import objects

conn = connector.Connector({
    'host': "10.0.100.60",
    'username': 'admin',
    'password': 'infoblox'
})

data = []
with open('/home/amin/infoblox/data.csv', newline='') as csvfile:
    reader = csv.DictReader(csvfile, dialect='excel', delimiter=';')
    for row in reader:
        data.append(row)

vlanRange =
"vlanrange/ZG5zLnZsYW5fcmFuZ2UkZGVmYXVsdC4xLjQwOTQuVGVzdHJhbmdlLjUwMC45OTk:def
ault/Testrange/500/999"

for net in data:
    vlan = objects.Vlan.search(conn, id=net['VLAN'], parent=vlanRange)

    network = objects.NetworkV4.create(conn,
                                       comment=net['NETWORK_NAME'],
                                       update_if_exists=True,
                                       cidr=net['NETWORK_SUBNET'],
                                       vlans=[{'vlan': vlan.ref}])
```

Listing 6. Python code to link VLANs to networks

In listing 5, I've included what the CSV file looked like when connecting the VLANs with networks. All that's needed is a name for the network, the actual network subnet and the VLAN number.

The code in listing 6 is what does the actual linking. The CSV file is stored into the data variable. Later each row in the file is looped through and in the loop, the data points may be accessed with for example: net['NETWORK_NAME']. The same column names that were set in the CSV file.

For each row, the corresponding VLAN is searched for with the ID provided from the CSV file and referenced through net[‘VLAN’]. The VLAN must be created in advance. Manual creation of VLANs is preferred as creating many is very quick even through the GUI and the risk of incorrect VLANs being created is minimized.

After the VLAN is found, a network is created according to the data found in the CSV file, and crucially the “vlans” attribute is added to reference the just searched VLAN. Additionally, an “update_if_exists=True” is added so even if a network is already created, it links still links the VLAN to it. Lastly, the comment field for the network is essentially the name of the network object. This is also defined in the CSV file but could even be left empty as networks do not require a name.

6 Future improvements

6.1 Continuous development

The project was completed successfully but the API was only used for linking VLANs and networks together. The API is capable of much more and could be utilized to a much further effect in automatization. This has already saved a lot of manual labour and additionally made categorization and troubleshooting much easier with the linking’s.

Further steps include creating more readily accessible tools for people using Infoblox on the daily. This includes CLI based tools and possibly a website for ease of use. Another priority is further integration into the ServiceNow ITSM platform that is already being used at the company. This is an ongoing project and will be continued.

6.2 Ansible and Terraform

The next step that came to mind in automation was to move away from direct API usage and use tools such as terraform and ansible.

In ansible there is an Infoblox module that makes use of the Infoblox-client framework, but no modules have been developed for VLAN management. [17.] It contains a lot of useful material to manage networks and DNS in general but doesn’t really have anything more. Some community modules are available, but they are not official so usage in production environments would be questionable. Ansible usage would require the writing of custom modules.

Terraform faced the same problem. Only network and DNS objects were supported by the official plugin. [18.] This was a bit odd as the API is very extensible and has sufficient documentation. Unfortunately, the community support for Infoblox is lacking in general.

7 Conclusion

The goal of this study was to originally just import the VLANs into Infoblox. This could have been done very quickly but careful planning made the process go very smoothly without problems.

In addition, automatization was introduced with a simple solution. With the framework in place and a little bit done, further automatization will be simple as the lessons learned here will help any upcoming projects. The only resource needed for more automatization is time.

The greatest obstacle was originally dealing with the API through direct HTTP requests as it introduced unnecessary complexity. Finding and utilizing the Infoblox-client framework made development go smoothly.

In conclusion, this was an interesting project that taught me a lot about REST API and different automatization tools. Even though I didn’t end up using

ansible or terraform, I still tried the modules and learned about both tools in general.

References

1 Infoblox market share study. 2015. https://www.infoblox.com/wp-content/uploads/idc-worldwide-ddi-market-update_0.pdf. Accessed 10.2.2023.

2 Okasha, Karim. 2020. Network Automation Cookbook. Packt Publishing.

3 Netbox, Netbox GUI Demo. https://demo.netbox.dev/. Accessed 22.2.2023

4 phpIPAM, phpIPAM GUI Demo. https://demo.phpipam.net/. Accessed 22.2.2023.

5 Infoblox, Managing Licenses. https://docs.infoblox.com/space/nios85/35417954/Managing+Licenses. Accessed 10.2.2023.

6 Infoblox, Appliance Administration. https://docs.infoblox.com/space/nios85/35480477/Appliance+Administration. Accessed 10.2.2023.

7 Infoblox, VLAN Management. https://docs.infoblox.com/space/nios85/35480165/VLAN+Management. Accessed 10.2.2023.

8 Infoblox, Configuring VLAN Views. https://docs.infoblox.com/space/nios85/35751132/Configuring+VLAN+Views. Accessed 10.2.2023.

9 Infoblox, Configuring VLAN Ranges. https://docs.infoblox.com/space/nios85/35481119/Configuring+VLAN+Ranges. Accessed 10.2.2023.

10 Infoblox, Configuring VLAN Objects. https://docs.infoblox.com/space/nios85/35481101/Configuring+VLAN+Objects. Accessed. 10.2.2023.

11 Infoblox, Managing Extensible Attributes. https://docs.infoblox.com/space/nios85/35448912/Managing+Extensible+Attributes. Accessed 10.2.2023.

12 Infoblox, Importing and Exporting Data using CSV Import. https://docs.infoblox.com/space/nios85/35478602/Importing+and+Exporting+Data+using++CSV+Import. Accessed 10.2.2023.

13 Infoblox, Assigning VLANs to a Network. https://docs.infoblox.com/space/nios85/35481082/Assigning+VLANs+to+a+Network. Accessed 10.2.2023.

14 Infoblox WAPI documentation. https://ipam.illinois.edu/wapidoc/index.html#. Accessed 10.2.2023.

15 Infoblox WAPI documentation, examples accessing WAPI using Curl. https://ipam.illinois.edu/wapidoc/additional/sample.html. Accessed 10.2.2023.

16 Infoblox-Client documentation. https://infoblox-client.readthedocs.io/en/stable/readme.html. Accessed 10.2.2023.

17 Ansible, Infoblox plugin. https://docs.ansible.com/ansible/latest/scenario_guides/guide_infoblox.html. Accessed 10.2.2023.

18 Terraform, Infoblox IPAM Driver for Terraform. https://registry.terraform.io/providers/infobloxopen/infoblox/latest/docs. Accessed 10.2.2023