



# Portfolio

NOGUEIRA CALDAS Rui Pedro

### Disclaimer:

All work shown in this Portfolio was either written by me or the sources used are referenced.

## Contents

.....	1
Disclaimer: .....	2
Certificates .....	4
.....	4
BTS-CYB Projects .....	6
Semester 1 .....	6
Learn Python Project: .....	6
My Python Project .....	7
Semester 2 .....	10
Cloud computing project .....	10
Semester 3 .....	12
SIEM Project (GORDS + PROMA) .....	12
Internship Project.....	15
1. Acknowledgements.....	16
2. Declaration of honor and validation of the report by the company tutor.....	16
3. Summary .....	16
4. Integration within the company.....	17
4.1. Description of the service and its functions .....	17
4.2. Summary of tasks.....	17
5. Internship project .....	18
5.1. Description of the project .....	18
5.2. Approach and problems encountered .....	22
5.3. Presentation of results and solutions .....	25

## Certificates



Figure 4: PowerPoint 2019 Associate Certificate



Figure 5: Word 2019 Associate Certificate



Figure 6: Word 2019 Expert Certificate



Figure 7: Outlook 2019 Associate Certificate



Figure 8: Excel 2019 Associate Certificate

## BTS-CYB Projects

### Semester 1

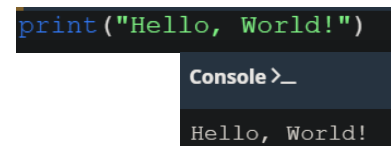
#### Learn Python Project:

At the start of the semester 1 we got the news that we would learn Python by ourselves with a Cisco Netacad Course which was structured in a step-by-step format and designed for beginners. The course was divided into modules that covered a variety of topics, including data types, variables, and control structures. The modules included lessons, interactive quizzes, and practical exercises called Labs, that allowed me to practice my new skills. Those exercises had to be documented with screenshots and uploaded on eduMoodle.

Python is a well-known programming language that is widely used in various applications. One of the key advantages of Python is its simplicity. The language is designed to be easy to understand and its syntax is intuitive. Python also has similarities to Java and since I already had experience with Java from my 1GIN it wasn't a big challenge to me.

The first little program was a print command that prints the phrase "Hello, World!" into the console. In the beginning we learned the basics like the print command or what a variable is and how to use

it. Later, when we had learned the basics, we used them to create lists and do calculations with the numbers in the lists and other more complex programs.



*Figure 9: "Hello World" syntax*

After each module we had to take a test about what we learned in that module to see if we had understood everything that the module had to offer. Those tests consist of multiple-choice questions, and you have to archive 70% to pass the test. We also had to hand in screenshots of the result to our teacher, so that he sees that we had understood that module.

I found the lessons to be well-explained and simple to grasp. I particularly cherished the course's flexibility, as I was able to complete the modules at my own pace. I could even start a module on one day and finish it on another day if I had time issues. This helped me to concentrate on areas where I needed to spend more time without feeling overwhelmed by the content. I was also able to revisit and review earlier courses and tasks as many times as I wanted to strengthen my understanding of the concepts.

After finishing the Cisco Netacad course, I felt confident in my ability to write and understand Python code. I was also able to use my new talents to some personal projects, including a simple text-based game, where you choose the path of the story, more about this project will be in its own section. The hands-on experience that I gained from these projects helped me to further develop my skills and deepen my understanding of the language.

In conclusion, my experience with autonomously learning Python through the Cisco Netacad course was extremely positive. The course structure and content provided me with a solid foundation in the language. I would highly recommend this course to anyone who is interested in learning Python or wants to expand their skills in the field of programming.

(504 words)

## My Python Project

For my Python project was inspired by the game Zork. Zork is a classic text-based adventure game. It is considered one of the earliest examples of interactive fiction and a precursor to modern role-playing and adventure games. In Zork, players navigate a virtual world using text commands, solving puzzles, fighting monsters, and collecting treasures. The game features a richly detailed world filled with humour, wit, and a range of obstacles that require creative problem-solving. Zork has become a classic in the history of gaming and is still enjoyed by many players today. So, I wanted to program something similar, but I didn't want the medieval setup because I find it kind of boring to reuse this setup for role-play games.

So I decided to use the Cyberpunk setup. Cyberpunk is a sub-genre of science fiction that focuses on a future society characterized by advanced technology, a breakdown of the social order, and widespread poverty and decay. Aesthetic cyberpunk refers to the specific visual and cultural elements that are associated with this genre, such as neon lights, cybernetic implants, and a dark, gritty urban environment. This aesthetic has been heavily influenced by cyberpunk literature and movies, and has become a popular style in art, fashion, and popular culture. The aesthetic of cyberpunk often reflects a dystopian vision of the future, where technology has outstripped humanity's ability to control it, and the world is ruled by powerful corporations and corrupt governments.

I started planning the game, I had a story about a head hunter and I was creating a world with a lot of layers but with time I started to see that I was too ambitious and that it was more work than I had thought so I cut some parts of and decided for just two locations, his home and a storage facility where the bad guys are. I thought to myself to finish this little game and then to expand it if I want to.

With the help of GitHub I was able to find a tool that opens a new window. In this window I would put my text and my buttons to take the decisions. I was really happy with the outcome but I had to program a new window for each decision, so I decided to start with less decision and to expand the story after finishing this first game. So it was like a tech demo, so that I could see how the technology to create this kind of things work.

```
import openpyxl
from tkinter import Tk, Label, Entry, Button, IntVar, Radiobutton

#Naming the Character
root = Tk()

#Name Button
label = Label(root, text="Enter your name:")
name_field = Entry(root)

#The submit button
def submit():
    name = name_field.get()
```

Figure 10: How to open a new window with Python

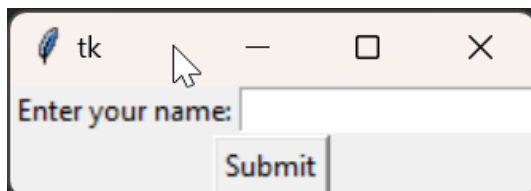


Figure 11: The opened window

So I had the window where the text and the choices would be displayed but I had to save the data so that I could get it later on if needed. I thought about what could be a good and simple way to save data and the think that came to my mind was Excel. Excel is easy to use and it should be easy to program with it because of the rows and columns, they are like coordinates for programs to access the wanted data.

So I searched online to find a way to create a excel sheet and how to save data in to specific cells and how to load the data from specific cells. After some time and a lot of web sites later I found what I had searched for. So I tested it and it worked to save the data for one playthrough but if you restarted the game another name would replace the old one. I went to see the original Zork game that is provided on a site and there it's also the same so I thought to myself that this problem was not a problem but like an historic feature, and also because I didn't want to have to solve it right now but it worked.

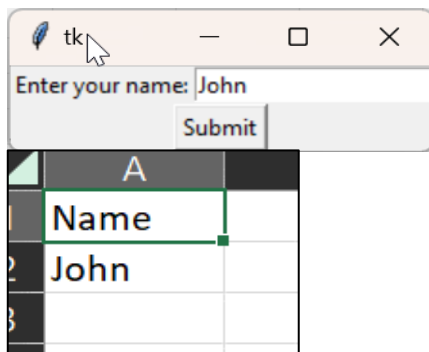


Figure 13: Example of a saved name

```
#The submit button
def submit():
    name = name_field.get()

    #Saving the name
    workbook = openpyxl.Workbook()
    worksheet = workbook.active
    data = [
        ["Name"],
        [name]
    ]
    for row in data:
        worksheet.append(row)
    workbook.save("TextRPG.xlsx")
```

Figure 12: How to save into Excel with Python

So if you put a name like for example "John" into the field and press the submit button. Then it will appear in the excel sheet. After the character is created/named we can start with the story so I wrote the beginning of the story to setup the path of the game. Here came another problem, I am really not that creative and not that good of a storyteller so I had to find something that is exciting to me to write with passion. As a kid was influenced by my father whose favorite actor is Silvester Stallone. We watched a lot of his films like Rambo or Rocky but we watched also films with Arnold Schwarzenegger and Bruce Willis. Those

were my childhood heros so I wanted to give some aspects of those actors to my character, that is why I decided that he would be a Head Hunter in a futuristic setup. So I wanted the old head hunter to be a grumpy man who's only joy in life is to drink his rhum but to get attached to people throughout the story and there I was laready planning ahead of myself so I stepped back and only did the grumpy old man part. So I began to write the setup with that idea in my head. After deciding to make is home the start point of the story I described his room to get a feeling of his

```
#Your chamber --- Start Point
home = Tk()
wb = openpyxl.load_workbook("TextRPG.xlsx")
sheet = wb.active
money = sheet["B2"].value
l0 = Label(home, text=("EuroDollar: ", money))
l1 = Label(home, text="The neon lights of the city blazed through the windows of the high-")
l2 = Label(home, text="You lay in bed for a moment, letting the familiar hum of the city w")
l3 = Label(home, text="")
l4 = Label(home, text="You glance around the sparsely furnished room, taking in the gunmet")
l5 = Label(home, text="This was your home, a place where you can escape the chaos of the c")
l6 = Label(home, text="")
```

Figure 14: Python text for game

grumpiness and I also described a bit the city and how dangerous it is, even for a experienced head hunter.

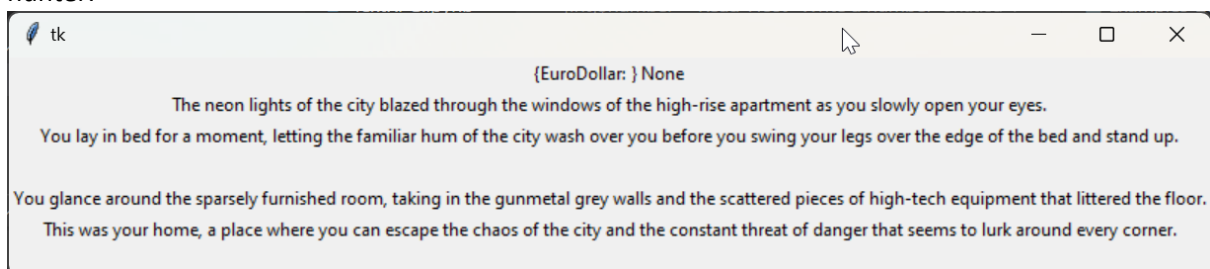


Figure 15: The text in the window



I looked at the first real story window and was happy that it worked. Of course it could be fancier but this is only a tech demo, so I was really happy with the progress. To think that my Python journey started with a Snake joke and with the iconic syntax `print("Hello World!")` and now I am programing a classic video game that might bring some nostalgic feelings for some people. In the end I am programming this game because everybody says that good games with heart aren't produced anymore but most of them don't do the work to create a game how they want to. I am currently still working on the game to expand it and to create more decisions with more than two endings.

(1017 words)

## Semester 2

### Cloud computing project

During the second semester we had a handful of projects to do, every second course would have a project if not two projects. In my opinion the one that I was more interested in and where I learned more new things was the project in the cloud computing course. We were paired randomly into groups of 2, Anmol and I were put in the same group. Our task was it to build a server out of the components that our teacher gave us and setting it up. At the end we had to have a Proxmox server with different virtual machines installed in it. So, we started by inspecting the different components of the server and began to build it up. We installed 12x 2GB Ram modules, 3 HDDs of 160GB each and a 300GB RAID HDD. We had two power supplies for the case that one would not work, or malfunction so that the server could continue running. We also had a RAID controller with his own power battery, to be able to start the server remotely. Before we started the server, we had to do some checks like if both power supplies were working at the start, if the system had detected each piece of hardware correctly. Then we had to configure the RAID controller so that we could access its web interface correctly, so that we could start the server from outside.

After the whole hardware was working and everything was configured, we started to install the OS. For this step we used two USB-sticks. One was the boot device from where the server would start, the other was for installing the OS onto the boot device. We downloaded the Proxmox Hypervisor ISO image on one of the USB-sticks and installed it onto the other USB-stick, this one would stay on the backside of the server. After the installation was finished, we had to do the main configurations, creating a new user and the change network configuration.



Figure 16: The server

After all this was done, we could install the server on the rack. This happened on the 29<sup>th</sup> March 2023. We had to figure out how to do the cable management. The whole class decided on one colour scheme and then we began to connect the cables with the server. The blue cable is the connection to the outside network so that we can access it from outside, the red cable is for the RAID controller network, the yellow cable goes to the motherboard and the white cable is for the RMM controller. We ended that day by testing the remote control to see if our configuration was correct and if we could start the server remotely.

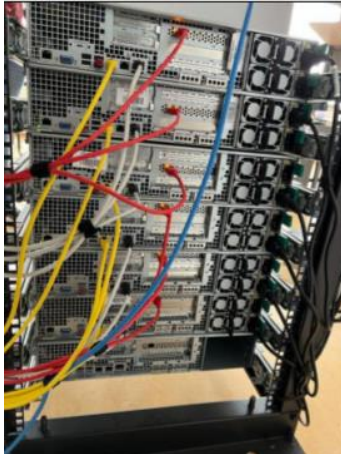


Figure 17: The server rack

After everything was configured correctly, we erased any remaining data that was still on the 3 disks. We then had to create all the directories to save the virtual machines, ISO files and everything else we needed. At this point we were good to go, and we could upload some ISO files.

To upload the ISO files onto the server we had to use FileZilla. At the start we uploaded two ISO files, one Windows 10 and an Ubuntu server ISO file because those were the requirements for the project. We then created some machines to test everything. We did backups of the machines to see if it would work, we also tried to restore the backups to see if they were useable and we cloned a ubuntu machine so that we could create an already running ubuntu machine without losing too much time, which we used for other projects and labs, so it was a nice tool to have.

For this project we also had to write all our steps down and show them with pictures and answer some questions that were asked related to servers and cloud services.

We put the server rack in the classroom of the second-year cybersecurity because they were away for their internship, so the noise and the heat wouldn't affect anyone. The servers were also used for other projects outside of that course. For example, we used the server for a second Cloud project that was about "Infrastructure as Code", where we worked with the tool Chef and had to show how we used a code to let our machines pull different tools from our main server. We also used the servers for the ParseMe project where we got a file type and had to compile it into other file types like from XML to PDF and CSV. So, the servers were useful for the other courses as well. The server was running till the 21<sup>st</sup> June 2023. At this date we disconnected the cables and took all components out.



Figure 18: The interface

As my conclusion, I think the project was interesting because it was my first time building a server, so I was not sure if anything I was doing was right, but it was quite easy and not too difficult, and I learned a lot that could be useful for the future in workplaces or even for me privately.

## Semester 3

### SIEM Project (GORDS + PROMA)

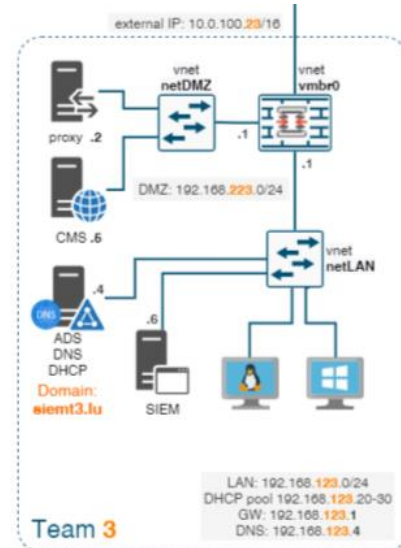
For the third semester we had 2 big projects on which we worked on. One was again a Python project, and the other was the project for the Project Management 3 and for the Governance, Risk and Defensive Security courses. For the project the class was divided into three groups. I was in a group with Mohsen, Noah and Tiago. This project would be marked and would be the majority of the marks for both courses. This project was about having a network with different machines and a SIEM. SIEM stands for Security Information and Event Management, it is a comprehensive approach to cybersecurity that includes the collection, normalization, and analysis of log data from various sources throughout an organization's technology infrastructure. SIEM systems aggregate logs from various devices, applications, and networks to provide a centralized, real-time view of an organization's security. These logs, which record events and activities, are normalized into a consistent format to facilitate analysis. One of SIEM's primary functions is correlation, which involves identifying patterns and relationships between seemingly unrelated events. SIEM tools allow organizations to proactively respond to and mitigate potential threats by detecting anomalies and potential security incidents, thereby improving overall cybersecurity posture. We used Wazuh as our SIEM.

For this project we also had to work with team management tools, in our case it was Jira, where we created all the tasks that had to be done, we could also assign steps to each task and assign the person that works on them. For documentation we used Confluence where we created a page for each task. We put all the tasks that had to be done in one week into a sprint that had to be finished each Thursday. If a task was not finished from a sprint we could just take it with us to the next sprint.

Our network was composed of the SIEM, a Firewall with 3 network interfaces, a web application server, a proxy server, an active directory server with DNS, an ubuntu desktop workstation and a windows workstation.

First, we had to install all machines correctly and configure them. We had some problems with the installation of the SIEM but in the end we got it to work. The SIEM is itself composed of different parts. The SIEM has the dashboard, that is the web interface in which you can see all the logs and can filter them for better search. Then you have the indexer, which has the role to index the logs into the right category. At last, there is the server, which is also composed of the manager, which manages the logs from the machines and gives them to the indexer and the API who ensures the connection to the dashboard. After the installations and initial configurations, we had to create rules on the firewall, so that we could access the machines from one to another. For this we divided the Firewall rules into two phases. The first phase was to allow every traffic and in the second phase we would limit the traffic to just what had to be.

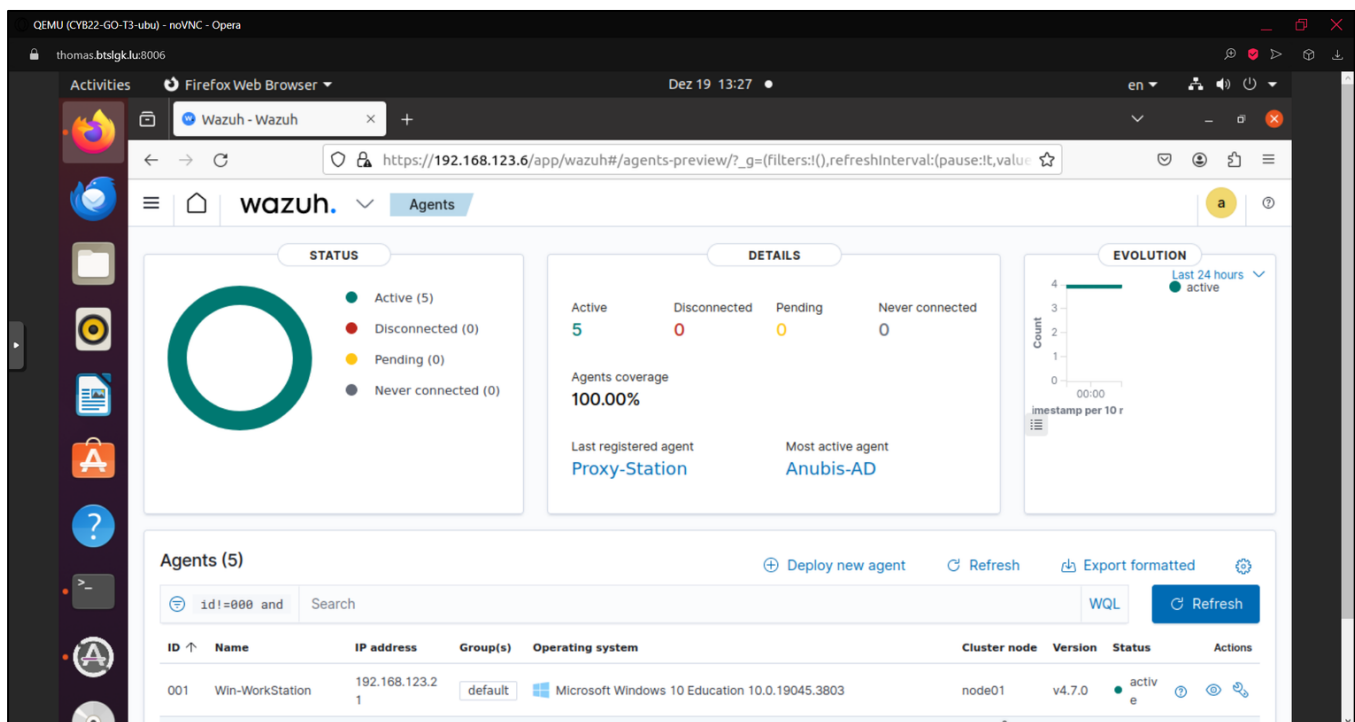
After some test we were ready to start sending logs from the machines to the SIEM. For this we used most of the time agents. Agents are lightweight software components installed on monitored endpoints or servers. These agents are responsible for collecting security-relevant information, such



as logs and events, from the host where they are deployed. They then forward this data to the Wazuh manager, which centrally processes and analyses it.

The only machine for which we could not use an agent was the firewall. For this we got an exercise from Excellium, that also worked with us on this project, giving us information. In this exercise we had to use rsyslog to send the logs from the firewall to the SIEM. We had to enable this function in the firewall, in which we also specified which types of logs should be sent. On the SIEM we also had to enable the port 514, which is used by rsyslog and we had to create a rule to allow rsyslog to be sent from the project network. After enabling everything we could see the logs arrive from the firewall.

On the dashboard we saw that the SIEM was connected to five machines that were using agents. We had then to do some filtering to get also shown the firewall logs on the dashboard.



After the SIEM was receiving all logs from all the machines we just had to create rules for the SIEM so that we would only get logs about the things that were in our interest. We created then one rule that alerted us when a connection was being done to or from an IP address that is on a list of known malicious IP addresses. We tested everything to see if everything was working so we could conclude this project.

At the end of the project, we had to do a documentation, in which every step should be explained with screenshots, and we had to do a presentation for 20-25 minutes in which we would explain our project to the person buying our services. So, we had to do the presentation in a manner that a person that is not in IT would understand it. At the end of the presentation there would also be a Questions and Answers round, where we would have to answer to questions the client would have had. In our case the presentation was about 20 minutes long, it was simple without going into much detail, so it was clear and easy to understand. After the presentation the teacher told us it was clear everything was already explained in the presentation so he could only ask questions about our team management and teamwork. We explained that it could have been better but that we at the end

finished everything in time, we worked well with each other even when 2 people were working on the same machine, it worked out pretty well.  
(1004 words)

# Internship report

Company:



SAASHUP

---

NOGUEIRA CALDAS

Rui Pedro

2024

### 1. Acknowledgements

I would like to express my gratitude to Vehbo for his guidance throughout the project, his willingness to answer my questions, and his assistance when it was required. I am also indebted to Bertrand for providing me with the opportunity to undertake an internship in his company and for allowing me to gain insight into the dynamics of such an organization. Furthermore, I would like to thank Laurent for maintaining a positive atmosphere within the team through his light-hearted remarks while also demonstrating a sense of responsibility when necessary. I would also like to extend my gratitude to François for his profound knowledge of DNS and the infrastructure in place, as well as for his willingness to answer my questions. Furthermore, I would like to thank my colleagues, Arno and Bryan, for their assistance during the internship.

I would like to express my gratitude to Mr. Mathey for serving as my tutor during the BTS program and for providing guidance on how to enhance my academic performance and personal growth. I would also like to acknowledge Mr. Ludwig for facilitating my continued studies in the BTS program and for enabling me to participate in various events, where I had the opportunity to gain valuable insights and engage in competitive learning. I would also like to extend my appreciation to all the teachers who imparted their knowledge and wisdom throughout the program.

### 2. Declaration of honor and validation of the report by the company tutor

I hereby declare that this internship report is the result of my own independent work. All sources and auxiliary tools used have been duly acknowledged. All text is either quoted directly or the nature of the interpretation is indicated by the use of text citation. The reference list also includes internet resources, with URLs provided. This work has not yet been submitted for review by any other agency.

### 3. Summary

My project was to create a workflow that implements DNSSEC on the DNS server in a network. This project aims to improve the security of the DNS server and its service. The ansible playbooks will manage the aspect of the configuration of the machines.



## 4. Integration within the company

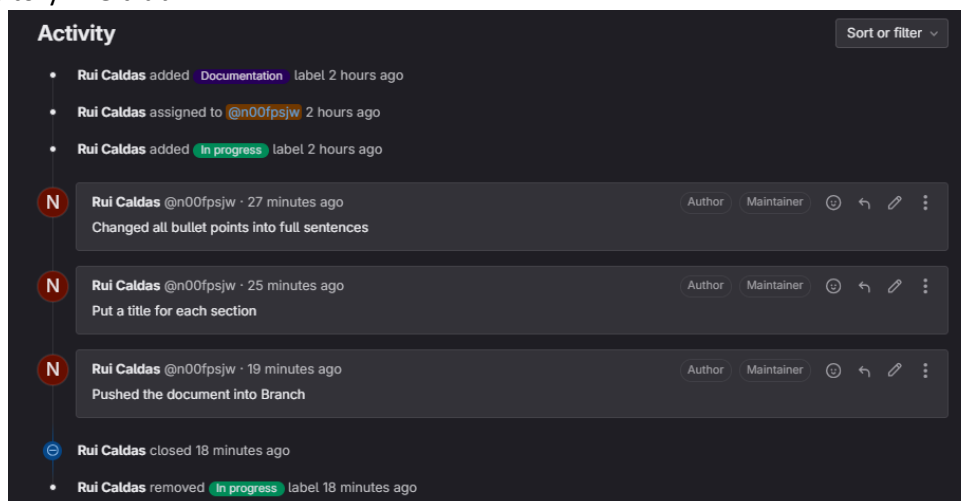
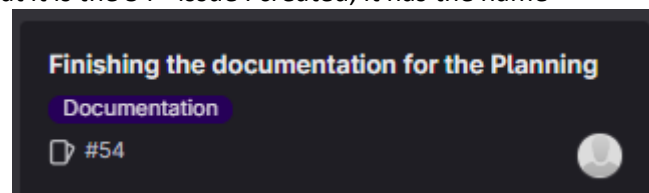
### 4.1. Description of the service and its functions

The service in question was called DevOps Internship, which works in conjunction with the service of the architects. In this service, interns are assigned projects that are relatively straightforward in that they do not necessitate the integration of a multitude of different mechanisms. I am under the supervision of the Architects, with Laurent as the Lead Architect. The architects also work on projects, but most of the time these are larger projects. Projects like mine can then be taken and implemented into more complex projects. The objective of my project is to implement DNSSEC. Consequently, should another project also require the implementation of DNSSEC, the developer can simply take my project and integrate it into theirs.

### 4.2. Summary of tasks

For my project I had some tasks that had to be done daily or weekly.

The daily tasks include creating issues where I write what I do, so that my tutor and the bosses can see what I have done that day. The issues are created and managed in GitLab. This is an issue of mine that I take as an example. Here you see that it is the 54<sup>th</sup> issue I created, it has the name “Finishing the documentation for the Planning”, it has a label which shows that it is an issue concerning the documentation phase and then we have at the bottom left the assignee, which is me. Inside the issue there is a description panel, where the issue is described in better detail. Then there is also the activity list, where it is possible to follow the activities of the issued. After the issue is done, I have to push it to the repository in GitLab.



Another daily task is to write everything down that I did that day, for this I also use my issues because there is already most of it written down, I just have to take all bullet points and sentences and change them into full sentences.

The weekly tasks include presenting what I did to my tutor so that he can see how far I have come during the week. This is done during “Daily Standup Meetings”, sometimes in person at my desk or in the conference room or sometimes over Google Meets. Sometimes we do that 2 times a week when there is a lot of process that has been made so it is easier to have an overview of it all. During this time, he looks over my internship report, tells me with whom I should get in contact to get more

information about specific topics and gives me an indication on what I should take a look on and what procedures should be added/changed.

## 5. Internship project

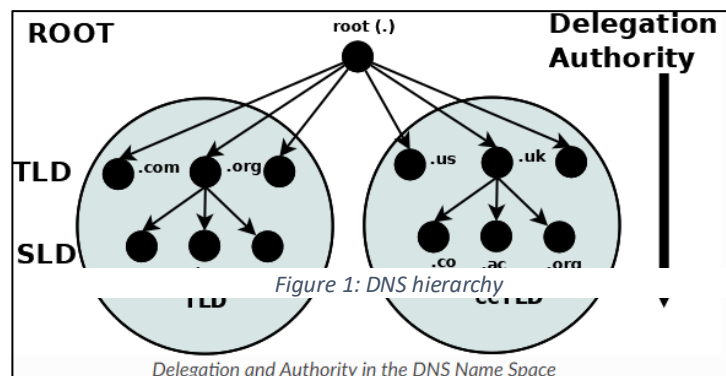
### 5.1. Description of the project

My project was to create an automated DNSSEC implementation system.

First let me explain what DNS and DNSSEC are. DNS, or Domain Name System, is a hierarchical and decentralized naming system for computers, services, or other resources connected to the internet or a private network. It translates human-readable domain names, like `www.example.com`, into IP addresses, such as `192.0.2.1`, which computers use to identify each other on the network. Essentially, DNS functions like a phone book for the internet, allowing users to access websites using easy-to-remember names instead of numerical addresses. It consists of various components, including DNS servers and resolvers, that work together to perform this translation process efficiently.

It is important to note that DNS does not verify the accuracy of the IP address. This can potentially result in users being directed to malicious websites due to DNS spoofing.

#### DNS hierarchy:



Left:

The Domain

(DNS) hierarchy is

initiated by the Root, represented by a dot (".") and containing pointers to Top-Level Domain (TLD) servers. TLDs, such as `.com` or `.org`, are managed by organisations such as the Internet Corporation for Assigned Names and Numbers (ICANN). Below TLDs are the

Second-Level Domains (SLDs), such as "example" in `example.com`, which individuals or organisations can register. This hierarchical structure enables efficient resolution of domain names to IP addresses.

Right:

Name System

The Root serves as the highest level of the

Domain Name System (DNS) hierarchy. Below the Root, ccTLDs, or Country Code Top-Level Domains, represent domains associated with specific countries or territories. Each ccTLD corresponds to a particular country or territory, like `.us` for

United States or `.uk` for the United Kingdom. The Root directs DNS queries to the appropriate ccTLD servers based on the country code portion of the domain name, facilitating access to websites and services specific to each country or territory.

DNSSEC, which stands for Domain Name System Security Extensions, is a set of protocols and standards designed to add a layer of security to the Domain Name System (DNS) infrastructure. DNSSEC introduces several new resource record types, including the following:

- **RRSIG** (digital resource record signature)

These signatures are used by recursive name servers, also known as validating resolvers, to verify the answers received.

- **DNSKEY** (public key)

The public keys are stored in the DNSKEY record types.

- ZSK (Zone-Signing Key): used to sign the zones.
- KSK (Key-Signing Key): used to link the parent and the child zones.

- **DS** (parent-child)

The parent zone vouches for the children zone by publishing its public key, which is then verified.

- **NSEC, NSEC3** (proof of nonexistence)

- **NSEC**

This method returns an answer indicating where the request should be if it existed.

- **NSEC3**

This method performs the same function as NSEC but uses one-way hashing to encrypt the response.

- **CDS, CDNSKEY** (child-parent signalling)

If a record of this type is present in a child zone, the parent zone is informed that it must update its DS records. The parent zone verifies the signed key by checking the CDS/CDNSKEY. The DS is modified to align with the CDS.

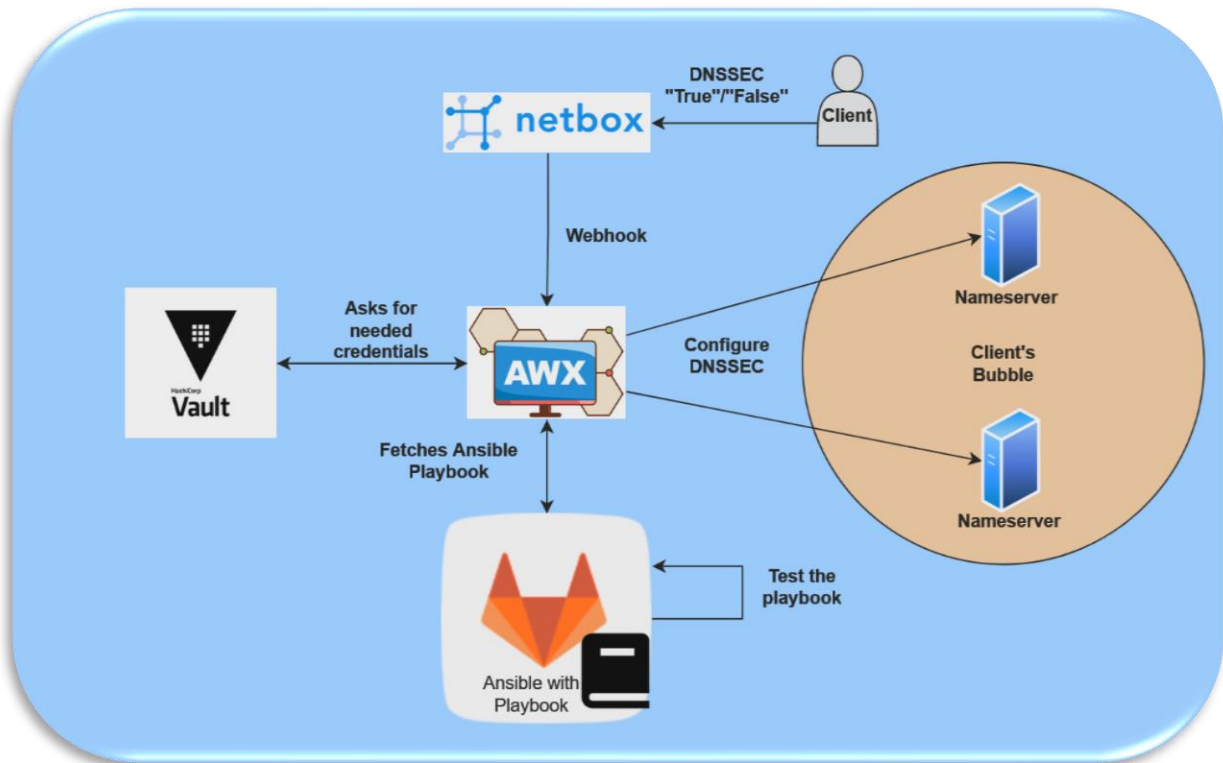
DNSSEC has three main phases.

1. Digital Signing: The domain owners digitally sign their DNS records using cryptographic keys that have been generated with algorithms.
2. Public Key Distribution: The public keys are distributed to the DNS resolver.
3. Validation: The DNS clients verify the digital signatures attached to DNS records to ensure their authenticity and integrity.

In this project I will not work with delegation because I will do DNSSEC just for the zone that the client wants it to be activated; the client can choose which zone is activated when he wants.

The implementation of DNSSEC needs to be fully automated, so that the client just has to press a button and it's all done with my ansible script.

In order to complete this project, it was necessary to utilize a number of new tools that I had not previously used. Here is the infrastructure with each tool and how it interacts with other tools:



**NetBox:** The first of these was NetBox, an open-source tool for managing IP addresses and data



centre infrastructure. It helps document and manage network assets, including IP addresses, subnets, racks, devices, and connections. With features like a user-friendly web interface, customizable fields, and a robust API, NetBox enhances visibility and operational efficiency in network management. NetBox serves as the

authoritative source of truth, meaning all changes must be made here first. It automatically assigns prefixes to subnets. In the planning phase, it's essential to identify current sources of truth, ensuring they are agreed upon by all relevant parties and that their domain is well-defined. Decisions on what to include in NetBox should be clear—if something has a mode, it belongs in NetBox. Validating existing data is a crucial step. While updates in NetBox are made manually, using Webhooks allows it to automatically update AWX.

**GitLab** is a web-based DevOps platform that provides a comprehensive set of tools for software



development and operations. It supports version control using Git, a free and open-source distributed version control system designed to handle projects of any size with speed and efficiency. Git creates snapshots of files that are stored locally, and while Git can be used locally, GitLab leverages these capabilities in a web-based environment. GitLab also includes features such as repositories for

code management and robust CI/CD (Continuous Integration/Continuous Deployment) capabilities. Continuous integration tests code frequently by integrating it into the source code, while continuous delivery ensures that tested software is always ready to be deployed into production. Although GitLab supports both CI and CD, CI is used within GitLab and CD is handled by AWX in your setup. In

addition, GitLab offers project management, issue tracking and various other collaboration tools to increase productivity and streamline the development process.

**Ansible** is an open-source automation engine used for configuration management, application



deployment, and task automation. It uses playbooks, which are configuration files written in YAML. Ansible operates with two categories of computers: the Control Node, which runs Ansible and sends modules via SSH to Managed Nodes, where these modules are executed and removed after finishing. Some examples of Ansible modules can be found here:

<https://opensource.com/article/19/3/developing-ansible-modules>. Managed Nodes are the systems being managed by the Control Node. Ansible Galaxy offers a collection of roles, modules, playbooks, and plugins that you can use, accessible here: <https://galaxy.ansible.com/ui/>. Additionally, there is a Git repository available for storing and managing playbooks.

**BIND9** is a widely used open-source DNS server software that translates human-readable domain



names into IP addresses, essential for locating and identifying computers on the Internet. It fully supports DNSSEC with a mature, full-featured, and

easy-to-use implementation. Once zones are initially signed, BIND9 can automatically re-sign dynamically updated records with inline signing. BIND's Key and Signing Policy utility helps maintain DNSSEC implementations by periodically updating keys and signatures according to the established policy. This makes BIND9 highly configurable and scalable, suitable for both small and large network environments.

**AWX** is an open-source community project that provides a web-based user interface and REST API to



manage Ansible Playbooks, inventories, credentials, and Vaults. It's utilized for Continuous Delivery (CD), facilitating the seamless deployment of tested software to production environments. Additionally, AWX integrates with NetBox, automatically applying changes made in NetBox through events triggered in AWX. This enables streamlined infrastructure management,

allowing non-technical users such as Helpdesk personnel to make changes to their infrastructure without needing deep knowledge of the underlying software or code.

**HashiCorp Vault** is a tool designed to securely manage and access secrets across distributed



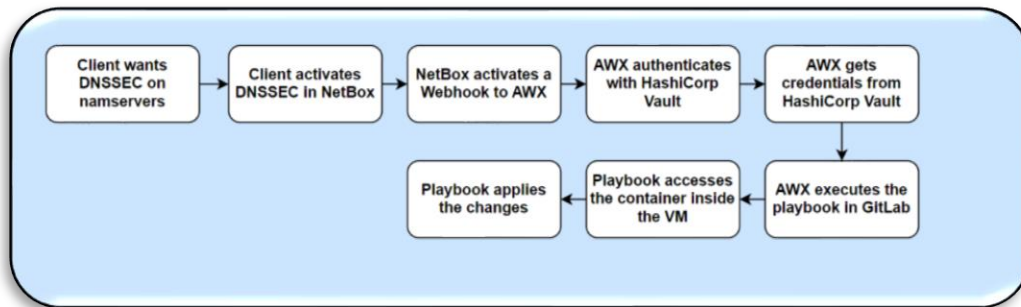
systems, serving the purpose of managing and protecting sensitive information. It accomplishes this through various components, including pluggable secrets engines and authentication methods that integrate with external systems. Secrets, which encompass credentials like API keys and passwords, are often hardcoded into source code, making them readable on platforms like GitLab and Ansible. However, Vault centralizes all secrets into one encrypted repository, safeguarding them at rest and in transit, thereby mitigating the risk of exposure. Access control lists

(ACLs) and auditing capabilities ensure that access to secrets is tightly controlled and monitored.

Vault offers dynamic secrets, which are credentials generated dynamically and usable only for a limited time, adding an additional layer of security. Additionally, Vault can encrypt and decrypt data within external applications, further enhancing security measures. Overall, HashiCorp Vault provides a comprehensive solution for managing secrets, enabling organizations to enhance their security posture and compliance adherence.

The entire process begins with the client pressing the button on NetBox to enable DNSSEC on one of his zones. NetBox then activates a webhook to AWX. AWX has an interface to manage Ansible Playbooks, inventories, credentials, and Vaults. AWX then gets all the needed credentials from

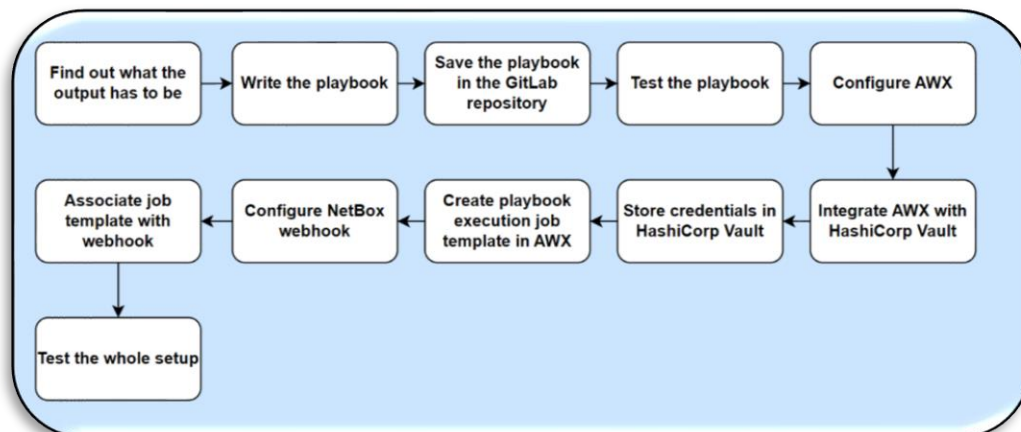
Hashicorp Vault with which it then executes the ansible playbook that is stored in GitLab. Hashicorp is a credentials management tool and GitLab is used as a repository and functions as the project management tool. The playbook is then run and applies the changes to the nameserver of the client.



## 5.2. Approach and problems encountered

First, I had to present the project to my tutor and the bosses. For that I had to do research about the tools I would be using, the procedures I was planning on doing.

After the presentation, they told me to change the order of the procedures and to do a workflow of how the system works. So, I began to create a workflow for how the system will work, so that I know which tools and software are connected and how they communicate with each other. I also created a workflow of the steps I would want to do to be able to succeed with the project.



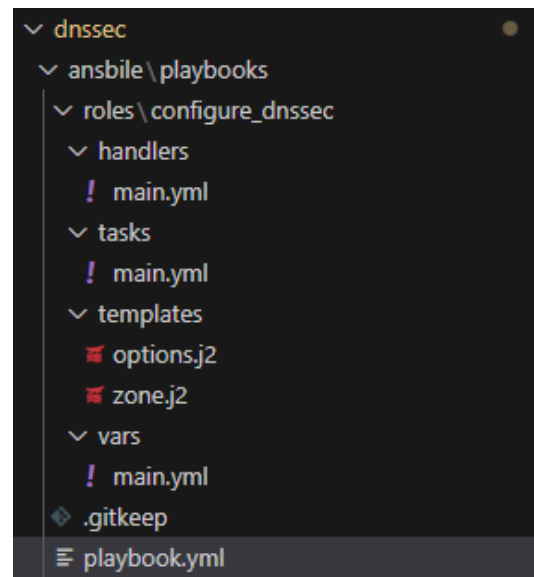
After the research and planning phase was finished, I started working on the ansible playbook. Unfortunately, the person I should be working with took two weeks off, so I had to try the beginning alone. I was able to create a playbook that implements DNSSEC on one of my virtual machines. After the worker came back, we arranged a meeting so that we could exchange information between us, so that he could know what I knew and what I had already done. He walked me through how NetBox works, how the Webhook gets activated, how AWX gets the playbook. I got access to the DNS repository, where other DNS projects are located, so that I could get an insight into the file hierarchy of the server. He also told me that I could create my own branch to be able to develop the playbook on the test servers. After a lot of struggles, I figured out what was needed to be able to implement DNSSEC through a playbook. After finishing the playbook and testing it multiple times on different machines to be sure that it works, I pushed it on to the repository.

Name	Last commit	Last update
Documentation	Added declaration of honor and worked on a part of the...	4 days ago
ansible/playbooks	Corrected by adding 'changed_when: myoutput.rc != 0'	4 days ago
.gitlab-ci.yml	Update .gitlab-ci.yml file	2 weeks ago
README.md	Initial commit	1 month ago

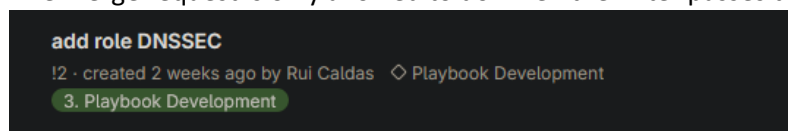
After Vehbo's inspection he told me to install a linter. A linter is a tool used in software development to analyse source code for potential errors, bugs, or stylistic inconsistencies. It works by parsing code and comparing it against predefined rules or coding standards. Linters provide feedback to developers, highlighting issues such as syntax errors, variable naming conventions, indentation problems, and more. By enforcing coding best practices and standards, linters help improve code quality, maintainability, and readability. They are commonly integrated into development workflows, either through command-line interfaces or as plugins within integrated development environments.

With the ansible linter on Virtual Studio Code, I went over my code and corrected everything that was wrong or wrong formatted. Vehbo also told Bryan to arrange a meeting with me to look over the best practices of ansible.

Bryan showed me how to structure the repository so that it is easier to work with the project if it gets upscaled or if it is implemented into another project. Before I had one file with the playbook where all my code was on. Afterwards I had multiple files with each file containing specific code. Here we see all the roles with each having a main.yml. I now also have templates with which I can change file on the target machine instead of going there with commands and trying to modify the files. Bryan also helped me implementing a linter into GitLab so that every time I push some code on to the repository, it gets automatically checked and shows me if it passed or not.

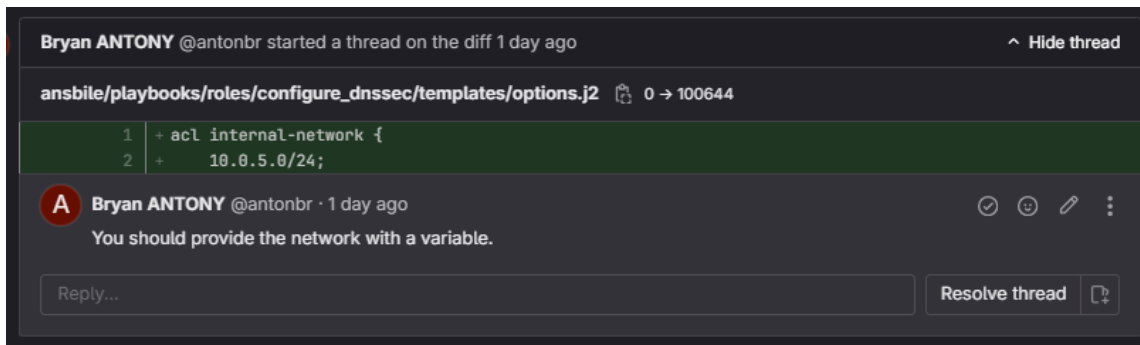


If it does not pass it shows me on which files and on which line, the error is that has to be corrected. After everything seemed to work fine, I did a merge request from my working branch into the main branch. The merge request is only allowed to do when the linter passes the code.



After the merge request is done, one of the supervisors looks at the content of the merge request and comments what should be changed to be accepted. Those comments look like this:





Here you see that the concerned file is shown, along with the line that requires modification and a comment explaining the reason for the change. Once the necessary corrections have been made, the option to "Resolve thread" can be selected, allowing the supervisor to verify that the task has been completed and to review the changes. If the corrections are satisfactory, the two branches will be merged, with the files from the working branch being incorporated into the main branch, ready for deployment.



I gained access to the repository of the DNS project, which was being developed by other workers. There, they were also engaged in implementing DNSSEC, but ultimately discontinued that project. This afforded me the opportunity to gain a deeper understanding of the infrastructure of the DNS server and the way their playbooks were written. This enabled me to draw inspiration from their code and to identify the differences between our approaches and the reasons for those differences. Consequently, I extracted a few lines of code from their code and tested them until they functioned as intended. A search was conducted to ascertain how the nameserver was incorporated into variables. This information was then used to implement a similar approach in the project. It was discovered that the variables were sourced from AWX. They put certain variables in AWX, to hide them from the public eyes. If we put important and confidential information into the code, people can see this information, because the code will be open source. So, we must include all sensitive information as variables coming from the AWX. They put certain variables in AWX, to hide them from the public eyes. If we put important and confidential information into the code, people can see this information, because the code will be open source. So, we must include all sensitive information as variables coming from the AWX. On AWX, we can place various variables to be included into the code for the deployment. After knowing this I wrote on a separate file all the variables that the program needs but can't be open to the public, so that I can add it later to AWX without forgetting them. I then started to implement all the other factors that I saw in the playbooks from the DNS project into my project, so that my project would be more whole. For this I started to create a Docker container with Bind9 on it to test my final program. I saved the container after having done

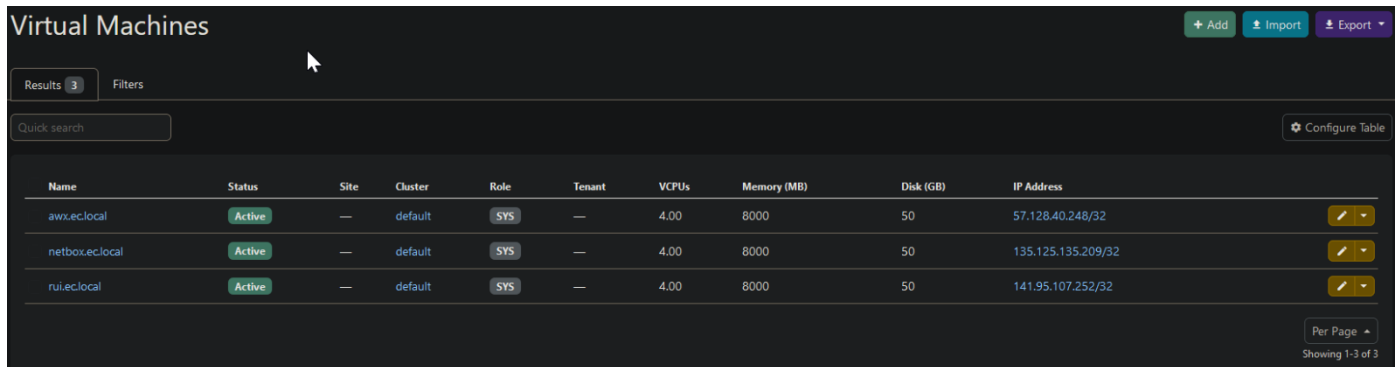


the standard configuration of the DNS server so that I could easily and fast start the docker machine at that point if the program would crash my machine or even if it would break something.

### 5.3. Presentation of results and solutions

#### 5.3.1. The infrastructure

Here we see the infrastructure, with the AWX machine, the NetBox machine and the DNS server called rui.ec.local.



Name	Status	Site	Cluster	Role	Tenant	VCPUs	Memory (MB)	Disk (GB)	IP Address
awx.ec.local	Active	—	default	SYS	—	4.00	8000	50	57.128.40.248/32
netbox.ec.local	Active	—	default	SYS	—	4.00	8000	50	135.125.135.209/32
rui.ec.local	Active	—	default	SYS	—	4.00	8000	50	141.95.107.252/32

```
dnssec
├── ansible
│   ├── playbooks
│   │   ├── roles
│   │   │   ├── handlers
│   │   │   │   └── main.yml
│   │   │   ├── tasks
│   │   │   │   └── main.yml
│   │   │   ├── templates
│   │   │   │   ├── options.j2
│   │   │   │   └── zone.j2
│   │   │   └── vars
│   │   │       └── main.yml
│   │   └── playbook.yml
│   └── collections
│       └── requirements.yml
```

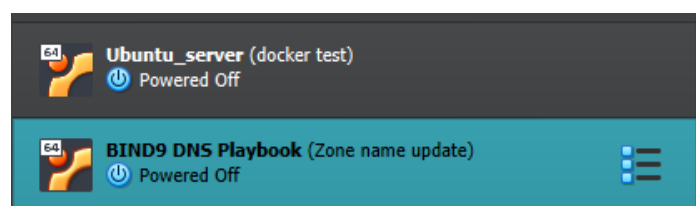
Here we have the file structure of the playbook with all the files it needs.

#### 5.3.2. The Playbook

So, during the internship I had to do and redo an ansible playbook that would implement DNSSEC automatically.

#### 5.3.3. Testing on local machines

When I had the playbook finished, I tested it first on some local machines of mine that I created for that purpose. So I created the Ubuntu\_server for executing the playbook and the BIND9 DNS Playbook is the machine where the playbook will configure DNSSEC. These machines are basic ubuntu machines. On the the BIND9 machine I have already installed BIND9 and configured a zone for test purposes. We can see



the state of the zone with the command “dig DNSKEY <zone name> @<ip of the nameserver> +multiline”, in our case we have “dig DNSKEY test.local @localhost +multiline” and this is the outcome:

```
beastii@boy:~$ dig DNSKEY test.local @localhost +multiline

; <>> DiG 9.18.18-0ubuntu0.22.04.2-Ubuntu <>> DNSKEY test.local @localhost +multiline
;; global options: +cmd
;; Got answer:
;; WARNING: .local is reserved for Multicast DNS
;; You are currently testing what happens when an mDNS query is leaked to DNS
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62429
;; flags: qr aa rd; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 85702f9e70e522310100000066470d1dc297e5ec2bca206d (good)
;; QUESTION SECTION:
;test.local.                IN DNSKEY

;; AUTHORITY SECTION:
test.local.                86400 IN SOA primary.test.local. root.primary.test.local. (
                                2022072651 ; serial
                                3600      ; refresh (1 hour)
                                1800      ; retry (30 minutes)
                                604800    ; expire (1 week)
                                604600    ; minimum (6 days 23 hours 56 minutes 40 seconds)
                                )

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(localhost) (UDP)
;; WHEN: Fri May 17 07:54:05 UTC 2024
;; MSG SIZE rcvd: 116
```

So here we see that the zone is not yet signed so there is no DNSSEC implemented.

For the local test I put the code all in the playbook.yml file because it is easier to test it because it is not that big that I would lose oversight of where is which part of the code. So, on the ubuntu\_server machine we have this playbook:

```
- name: Configure DNSSEC
  hosts: myhosts
  become: true
  tasks:
    - name: Ensure the necessary DNSSEC configuration is in place
      template:
        src: templates/options.j2
        dest: /etc/bind/named.conf.options

    - name: Create Zone Signing Key
      command: dnssec-keygen -a NSEC3RSASHA1 -b 2048 -n ZONE test.local
      args:
        chdir: /var/cache/bind
      register: zsk_key

    - name: Create Key Signing Key
      command: dnssec-keygen -f KSK -a NSEC3RSASHA1 -b 4096 -n ZONE test.local
      args:
        chdir: /var/cache/bind
      register: ksk_key

    - name: Ensure the zone file includes DNSKEY records
      shell: |
        for key in $(ls Ktest.local*.key); do
          echo "\$INCLUDE $key" >> /var/lib/bind/forward.test.local
        done
      args:
        chdir: /var/cache/bind

    - name: Generate random salt for DNSSEC signing
      shell: head -c 1000 /dev/random | shasum | cut -b 1-16
      register: dnssec_salt

    - name: Sign the zone
      command: >
        dnssec-signzone -A -3 (( dnssec_salt.stdout )) -N INCREMENT -o test.local -t /var/lib/bind/forward.test.local
      args:
        chdir: /var/cache/bind
      register: signed_zone

    - name: Update named.conf.local
      template:
        src: templates/zone.j2
        dest: /etc/bind/named.conf.local

    - name: Reload BIND
      service:
        name: bind9

"playbook.yml" 50L, 1412B                                     43,4      Top
```

So here we have all the parts of the code in one file instead of it being spread like in the real repository. In the inventory.yml file I only have the one machine listed.

```
beastii@boy:~/ansible_project$ cat inventory.yml
[myhosts]
10.0.5.5
```

To execute the playbook, I use the command “ansible-playbook -i inventory.yml playbook.yml -k --ask-become-pass”.

```
beastii@boy:~/ansible_project$ ansible-playbook -i inventory.yml playbook.yml -k --ask-become-pass
SSH password:
BECOME password(defaults to SSH password):

PLAY [Configure DNSSEC] *****
TASK [Gathering Facts] *****
ok: [10.0.5.5]
TASK [Ensure the necessary DNSSEC configuration is in place] *****
changed: [10.0.5.5]
TASK [Create Zone Signing Key] *****
changed: [10.0.5.5]
TASK [Create Key Signing Key] *****
changed: [10.0.5.5]
TASK [Ensure the zone file includes DNSKEY records] *****
changed: [10.0.5.5]
TASK [Generate random salt for DNSSEC signing] *****
changed: [10.0.5.5]
TASK [Sign the zone] *****
changed: [10.0.5.5]
TASK [Update named.conf.local] *****
changed: [10.0.5.5]
TASK [Reload BIND] *****
changed: [10.0.5.5]
PLAY RECAP *****
10.0.5.5 : ok=9 changed=8 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
```

Here we see that it asked me once for the SSH password because of the “-k” and once for the sudo password because of the “--ask-become-pass”. We also see that everything was executed without problem.

```
;; ANSWER SECTION:
test.local.      86400 IN DNSKEY 256 3 7 (
    AwEAAadnuJUPrgzQGC05b2qJgLSkn5zUYxmUv+MKb72EH
    uAQq/H0cB9znMnID1VCvB6FHeqv+/MNTJGruhB1yFLh5
    EvR26oACA0Apy6LF0EAXI9nJ7wP5bKpBNCXV4nMQ0BoR
    KJ/ISV77yh4oLsH0Ys+ADbin9vvaBfR7Wo91QbN7BQqP
    8I5bDxs14m28tWfJSJH0MB0ApXce80rs0oUJW23MoSea
    R/Gb1VeFw8WVG65z/KH2uUpXVovApPvvLMJzFti00U/I
    oHeJJEpodKxRAY6NeWfKuYw9w2GcB++hHp088//ofoh0
    r7Fz4JuBgh16TD071r+EIiPgoKfunEQOI7as8vM=
    ) ; ZSK; alg = NSEC3RSASHA1 ; key id = 19650
test.local.      86400 IN DNSKEY 257 3 7 (
    AwEAAZi4WQKoSduksNkAAge6Ewp4fTnRR5YYVwtXEUnC
    RLIO6X7W05Djmt9JvV3g5PhyQ5UYSCYBt4xBHXfkITK
    4ZF8fybH8qneK96njQdXU4gB2Qdx4T1KSPKEA1USDIIb
    bHr2N+NwXYDhVxpzQFtGCN+gJBsbU1ESP5g8unNg+hPD
    DuweeV+xd7//CmxTtkNvINOR5bCXHmHA33Sh76QP5+fs
    P8Q3mv28cx3mbXaf/tItUsRv40jKfzZ0AUXs9YpY3HzR
    TLGW722MxiKA7hS8d8spz2WixNtwh5bHYIiuk0o2br8Z
    17CRWxzWqv1+3hv435KFBo/bTONxLAFAGN67VHT01Nr
    ERoKu7CDUI+eN0z0QPg9J7rgtGc3MXsdqk4k0uRwGjv
    4xIz/c5m2bEtaTTO2T0WhWBO+PliwfYhmEhLdlcoAdhh
    ijaT05ap4if4Z/Cr10iGeDshY6C5ue1Ver1UA9hHbMag
    qE8nag7WmwviK62dh1yGh98TSS22qpKQqia/jeqYNz+y
    R1SWagt7bAFiI1P6rvSQh18Bh0cjpYYmLM9Hm3XhYafU
    PV90q9/jIkioTc75HStCLSKFRrPtIH0MbTckA1wicQGW
    DtH1z9YFcTaxgGYiA3f3D/zAI7ek7VptQwwCs65njTMv
    bcvBp30UMtrYHqAv1jKQBjYQcmB
    ) ; KSK; alg = NSEC3RSASHA1 ; key id = 39243

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(localhost) (UDP)
;; WHEN: Mon Jun 03 14:00:13 UTC 2024
;; MSG SIZE rcvd: 875
```

If we do the “dig” command now we see that the zone is signed with the DNSKEY.

#### 5.3.4. Testing with AWX on a VM on NetBox

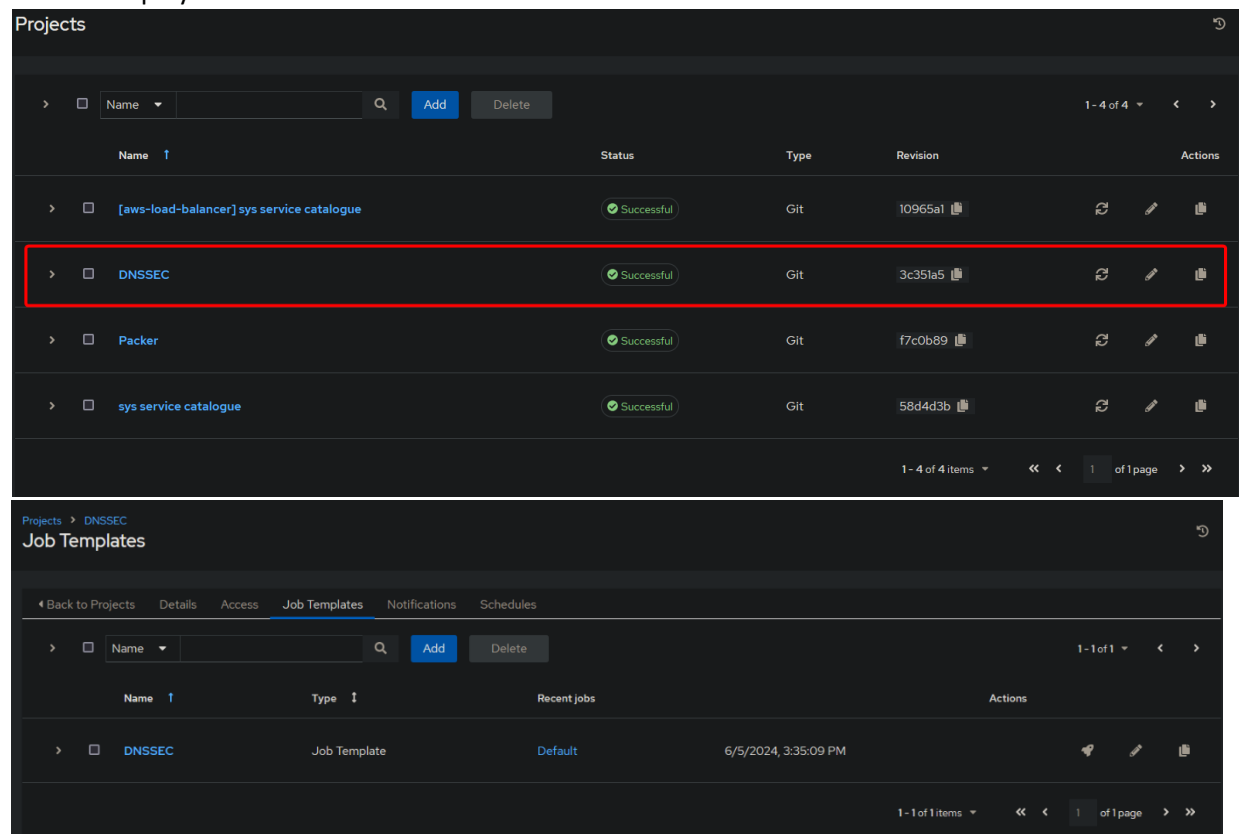
After having successfully tested the playbook on a local machine it is time to get it to real environment.

First, we will link AWX to NetBox, so that it can get all the machines that are shown on NetBox to work with. To do that we have to go to AWX and add a new source with the source variables.

```
Source variables  ⓘ  YAML  JSON
1  ---
2  NETBOX_API: http://135.125.135.209:8080/
3  NETBOX_TOKEN: 3f4cc03c92fe1fd2ddcc569c5077bab32db7b60a
```

On AWX, we will first link the GitLab repository to AWX, so it can get the playbook and all the necessary files.

On AWX we create a new project, which I called DNSSEC. There we can add the job template, that is where the playbook will be executed and where we can add more variables.



In the job template I added some variables that I need to run the playbook. Those are variables that should not be made public in the source code or variables that change from client to client.

```
Variables  ⓘ  YAML  JSON
1  ---
2  dnssec_zone: "test.local"
3  dnssec_zone_file: "/var/lib/bind/forward.test.local"
4  forwarders:
5    - 8.8.8.8
```

It is also necessary to configure the job template to get it right. You have to give it a name, say what it should do with the job, say which file it should use and for which project it is.