Atte Reenilä

Automating Aruba CX-series switch configuration with Ansible

Metropolia Ammattikorkeakoulu

Bachelor of Engineering

Information Technology

Bachelor’s Thesis

8.4.2023

Abstract

Author:	Atte Reenilä
Title:	Automating Aruba CX-series switch configuration with Ansible
Number of Pages:	37 pages + 1 appendices
Date:	8 April 2023
Degree:	Bachelor of Engineering
Degree Programme:	Information Technology
Professional Major:	Smart Systems
Supervisors:	Marko Uusitalo, Senior Lecturer

The aim of this thesis was to investigate what automation is, how automation is related to DevOps methodology and the Infrastructure as Code model, and what Ansible is and how it can be used in automating the configuration of Aruba CX series edge switches. The purpose of the study was to speed up device management by creating an Ansible Playbook, which can be used as a basis for device configuration and management in the future. In addition, the thesis reviewed competing technologies for Ansible and their differences.

In the practical part of the thesis, the Ansible Playbook was created, which can be used to automate the remote configuration of the Aruba CX 6100 switch using SSH in accordance with the Infrastructure as Code model, also taking into account the dynamic features of the environment and the device. The work also lightly reviews the device manufacturer's support for the device and lists various development proposals for the future.

The work was time-consuming, but the end result was an Ansible Playbook, which made it possible to successfully implement automation for the device.

Keywords: ansible, aruba, automation, CX-switch series, DevOps, IaC, network, switch

Tiivistelmä

Tekijä:	Atte Reenilä
Otsikko:	CX-sarjan kytkimen konfiguroinnin automatisointi Ansiblella
Sivumäärä:	37 sivua + 1 liitettä
Aika:	8.4.2023
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Smart Systems
Ohjaajat:	Lehtori Marko Uusitalo

Tämän opinnäytetyön tavoitteena oli tutkia, mitä automaatio on, miten automaatio liittyy DevOps metodologiaan ja Infrastructure as Code - malliin sekä mikä Ansible on ja miten sitä voidaan hyödyntää Aruba CX -sarjan reunakytkinten konfiguroinnin automatisoinnissa. Työn tarkoituksena oli nopeuttaa laitteiden hallintaa luomalla Ansible Playbook, jota voidaan hyödyntää jatkossa pohjana laitteiden konfiguroinnissa ja hallinnassa. Lisäksi työssä käytiin läpi kilpailevia teknologioita Ansiblelle ja niiden eroavaisuuksia.

Opinnäytetyössä käytännön osuudessa luotiin Ansible Playbook, jonka avulla pystytään automatisoimaan Aruba CX 6100 -kytkimen etäkonfigurointi SSH:n avulla Infrastructre as Code - mallia mukaillen. Automatisoinnissa otettiin huomioon ympäristön sekä laitteen dynaamiset piirteet. Työssä käytiin myös kevyesti läpi laitevalmistajan tarjoama tuki laitteelle ja listattiin erilaisia kehitysehdotuksia tulevaisuutta varten.

Työ vei aikaa, mutta lopputuloksena saavutettiin Ansible Playbook, jonka avulla mahdollistettiin automatisoinnin toteuttaminen laitteen osalta onnistuneesti.

Avainsanat: ansible, aruba, automation, CX-switch series, DevOps, IaC, network, switch

Contents

List of Abbreviations

ACL:	*Access-control list*. List of access rules associated with system resource. In networking, used to control access to system management
ARP:	*Address Resolution Protocol*. Discovers MAC-address associated with IPv4-address on networking
Ansible:	Software tool for automation and IaC development.
CLI:	*Command Line Interface*. User interface in text form.
CPU:	*Central Processing Unit*. Executes machine code on computer.
DevOps:	*Development Operations*. Methodology combining software development and IT operations.
GUI:	*Graphical User Interface. A u*ser interface with graphical elements.
Gbps:	*Gigabits per second.* Measurement of digital data transmission per second on medium.
HTTP:	*Hypertext Transfer Protocol.* Application layer protocol used to distribute web content over the Internet.
HTTPS:	*Hypertext Transfer Protocol Secure*. Application layer protocol with secure component used to distribute web content over the Internet.
ICT:	*Information and Communication Technology*.
IaC:	*Infrastructure as code*. Process of managing and provisioning devices using machine readable definition files.

Jinja2: *Jinja2 templating engine*. Templating engine commonly used in Python programming.

L2: *Layer 2 of the OSI model*.

LLDP: *Link Layer Discovery Protocol*. Network device advertising protocol.

OSI: *Open Systems Interconnection model*. Reference model which splits the networking into seven different abstraction layers.

PoE: *Power over Ethernet*. Standard for providing electric power to devices via Ethernet cabling.

Python: High-level general-purpose programming language.

RAM: *Random Access Memory*. Type of memory close to CPU used in computing.

REST-API: *Representational state transfer application programming interface.* Application programming interface used in the Web. Utilizes HTTP/HTTPS protocol.

Ruby: High-level general-purpose programming language.

SFP+: *Small Form-factor Pluggable*. Hot-pluggable network interface format used in telecommunication.

SNMP: *Simple Network Management Protocol*. Protocol used for collecting managed devices on network.

SSH: *Secure Shell Protocol*. Cryptographic network protocol. Provides remote access to remote host,

Switch: Networking hardware which connect devices within the network.

VLAN: V*irtual Local Area Network*. Allows logical partitioning of physical networking devices. Used in network segmentation.

YAML: Human Readable data-serialization language.

1 Introduction

Automation, which is becoming more and more common in the ICT sector, is almost a necessity today when talking about environments with thousands of managed devices. Managing these without automation is heavy task to do manually and requires several working hours of experts allocated to device configuration and management. Automation works as a solution to this, because with the help of automation it is possible to manage large amounts of devices efficiently while saving unnecessary working hours spent on unproductive tasks.

The purpose of this work is to investigate what automation is, what automation products can be found on the market. Another goal was to study what Ansible is, how automation is related to the DevOps philosophy and the Infrastructure as Code model, and how automation can be used to automate of Aruba CX switches.

The goal of this work is to automate the configuration of the network switch to a point, in which it can be used in the company's service production, thus improving the efficiency of experts and reducing the workload in the form of a new tool.

2 General Theory

2.1 DevOps

DevOps is a cultural philosophy which utilizes practices and tools between two parties: software development and IT team. The name comes from delivery and operation and the idea behind the philosophy is to combine Software Production and Device Management into one “team” in a way it creates a foundation for agile delivery pipeline. [1]

DevOps lifecycle can be defined with a Delivery Pipeline with a Feedback Loop as show in Figure 1.

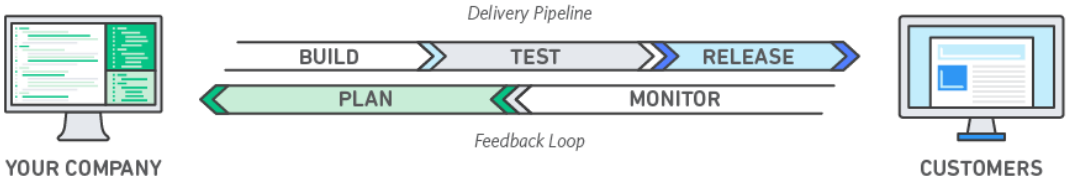

Figure 1: DevOps lifecycle [2].

The underlying idea behind DevOps is to create an environment for the production of the application, through which the intermediate stages of the application, i.e., building, testing, releasing, and delivery, feedback is collected through monitoring and it is then used in the continuous development of the application. This makes it faster and easier for developers to react to problems at earlier stages, and it improves quality observed from the end user perspective. [3]

For example, DevOps philosophy provides following benefits for production [4]:

- Speed
- Improved collaboration

- Rapid deployment and scalability
- Quality and reliability
- Security

Most of the benefits come from the following best practices DevOps [4]:

- Communication and Collaboration
- Configuration Management
- Continuous Delivery
- Continuous Integration
- Microservices and Containers
- Monitoring and Logging

The philosophy of DevOps is a prominent and growing part of the IT automation, as automation in general relies on IaC in infrastructure management [4].

2.2 IaC – Infrastructure as Code

IaC utilizes DevOps methodology and versioning with a descriptive model to define infrastructure by code. The code ensures that the functionality is identical regardless of the device and can be replicated in several different devices or environments. This has a huge impact on automation, because device configuration, which is stored as a template in the code repository, consistently replicates the configuration or environment in the system deployment. This makes IaC a key factor in a modern IT automation, as it works as a template for

automation by enabling service growth and scalability in continuous manner. Enforcing an IaC model reduces a risk of inconsistency between environments as manual configuration is avoided. This has a more streamlined effect on, for example, incident management, as it makes sure that the difference between environment objects is minimal or not present. [5] [6]

2.3 Automation

Automation is used to simplify and speed up repetitive tasks by standardizing the operation of the process for the use of information systems. In IT automation, automation is used to reduce manual human interaction with the digital systems, thus decreasing the time used for system management and enabling control of several devices at once. This means that the valuable time can be used for more productive tasks by taking time away from unnecessary manual labor. Automation has gradually become more common in information technology, although initially its roots are in manufacturing, such as the automotive industry. [7]

In IT, automation has taken leaps during the recent years as tools for automation have evolved and societies have digitized. The biggest shift has come from the DevOps philosophy and IaC model, which have emphasis on enabling seamless automation for service production. This shift has generated an increasing pressure on the companies to automate processes in order to gain a competitive advantage in the global market, as automation can be used to cut costs in the company's daily operations. [9] The most well known automation technologies in IT are the following: Ansible, Chef, Puppet, and Terraform [8]. These technologies are used especially in device configuration and management tasks, which is the focus of this thesis [9].

2.3.1 Benefits of automation

Usually, automation has been affiliated with benefits related to manufacturing, such as increased production rate, production growth, better product quality, higher level of manufacturing security and increased material efficiency. In IT automation, the benefits differ slightly, because in the IT market most of the value production is based on service production, which does not involve handling of physical materials but digital assets. [7]

IT automation provides the following benefits [7]:

- Service consistency
- Time efficiency
- Higher level of security
- Easier device handling
- Easier scalability

Benefits listed above cause the processes in service production to be more consistent, easier to scale, more efficiently executed with a higher level of security via easier device handling. This is due to the fact that the human involvement is reduced to minimum. Human involvement can cause variability and risks in the execution of processes through human factors. [7] [9]

These benefits enable IT companies to switch focus to more valuable tasks by reducing human resources in repetitive tasks. This means that IT companies can grow larger with minimal workforce needed and human resources can be allocated to tasks in which are more harder to automate and need more human flexibility. [9]

2.3.2 Issues with automation

Automation also causes challenges. Even though automation reduces human factors in processes, machines may also fail. This changes the focus from process execution to process monitoring and delivery, when automation becomes more common. [7]

The problems can be separated as follows [7]:

- Automation maintenance
- Flexibility
- Safety risks
- Worker displacement

This can be summarized as an increasing need for Human interaction in service production. Machines require maintenance, which can only by done by a professional with required skills. Displacement of workers is a risk as participants in production detach from daily process operation. [7]

2.4 Automation technologies

There are several suitable solutions for infrastructure automation on the market. Most of these are open-source technologies developed by privately owned companies. Because the characteristics and features vary between technologies, careful planning is required. Because this thesis studies available options for network device configuration, the research will be limited to technologies, which are suitable for configuration automation. [8] [10]

2.4.1 Ansible

Ansible is a open source automation tool developed by the Ansible Inc and owned by the RedHat Inc. The tool is created with infrastructure automation in mind and supports, for example, network, security and cloud/server automation via system integration. [11] [12]

Ansible libraries are written in Python, popular high-level programming language and thus do not require any additional code compiling. Ansible is a code-driven language, in which code resides in text-files and language syntax has a big emphasis on human readability. For communication between nodes, Ansible mainly uses OpenSSH protocol, but it also supports a wide range of different communication protocols via downloadable plugins. [13]

Ansible is used by defining Ansible Playbooks and inventory files. These files determine how management is done and in which systems. Inventory files contain node specific variables and Playbook determines how and what is then executed in the managed node. Ansible Roles are used to determine which nodes the Ansible Tasks on Playbooks are executed on. Figure 2 below illustrates the architecture of Ansible by displaying relation between Control- and management nodes. Control Node is used to manage Managed Nodes in question. [13]

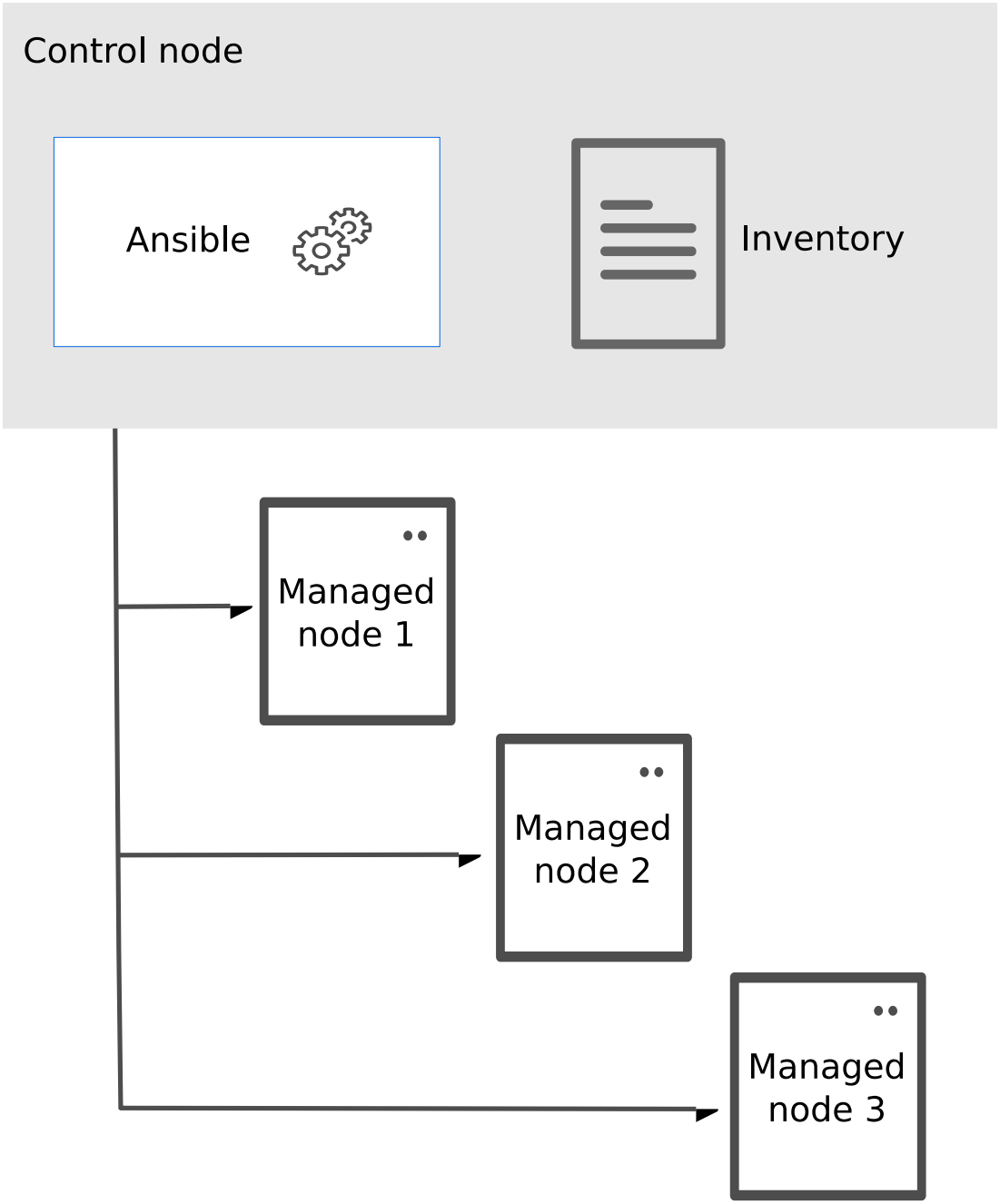

Figure 2: Ansible core architecture [14].

Code is executed agentless in declarative manner on the managed hosts. The management of servers and networking devices differs from each other. As Figure 3 shows, code is executed locally in networking devices. Linux / Windows hosts use remote execution by utilizing system's Python libraries and code copying. [12]

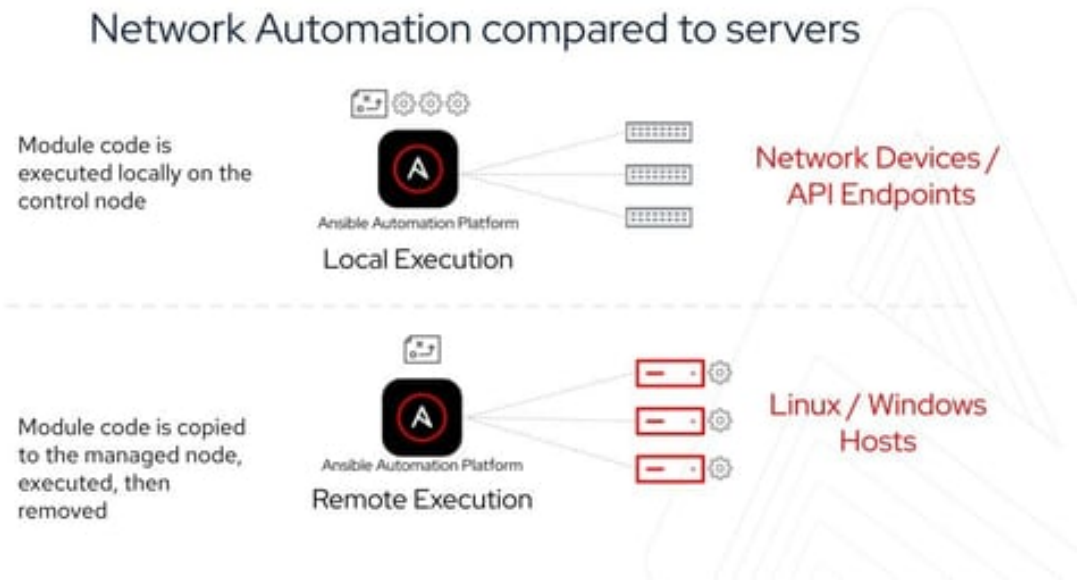

Figure 3: Difference between Ansible automation on different devices [12].

Ansible fit well in to the DevOps methodology as a tool. Infrastructure can be handled as a code using Ansible Playbooks and environment can be scaled by adding nodes to the inventory file. Manual configuration is limited and environment consistency is ensured. [15]

2.4.2 Chef

Chef Automate is an infrastructure automation software developed by Progress Chef. It uses Ruby programming language to create domain-specific language for infrastructure provisioning and management. Chef Automate has a high focus on code development, versioning, Infrastructure as Code mentality and DevOps philosophy, which influences its base for its core architecture. Chef Automation software stack provides necessary software for DevOps development. Chef utilizes three systems with their own roles in its workflow: Chef Workstation, Chef Server and Chef infra (nodes), which makes Chef a more complex system than, e,g., Ansible. These components play their own roles in a continuous development cycle. Figure 4 illustrates a simplified model of the Chef core architecture. [16]

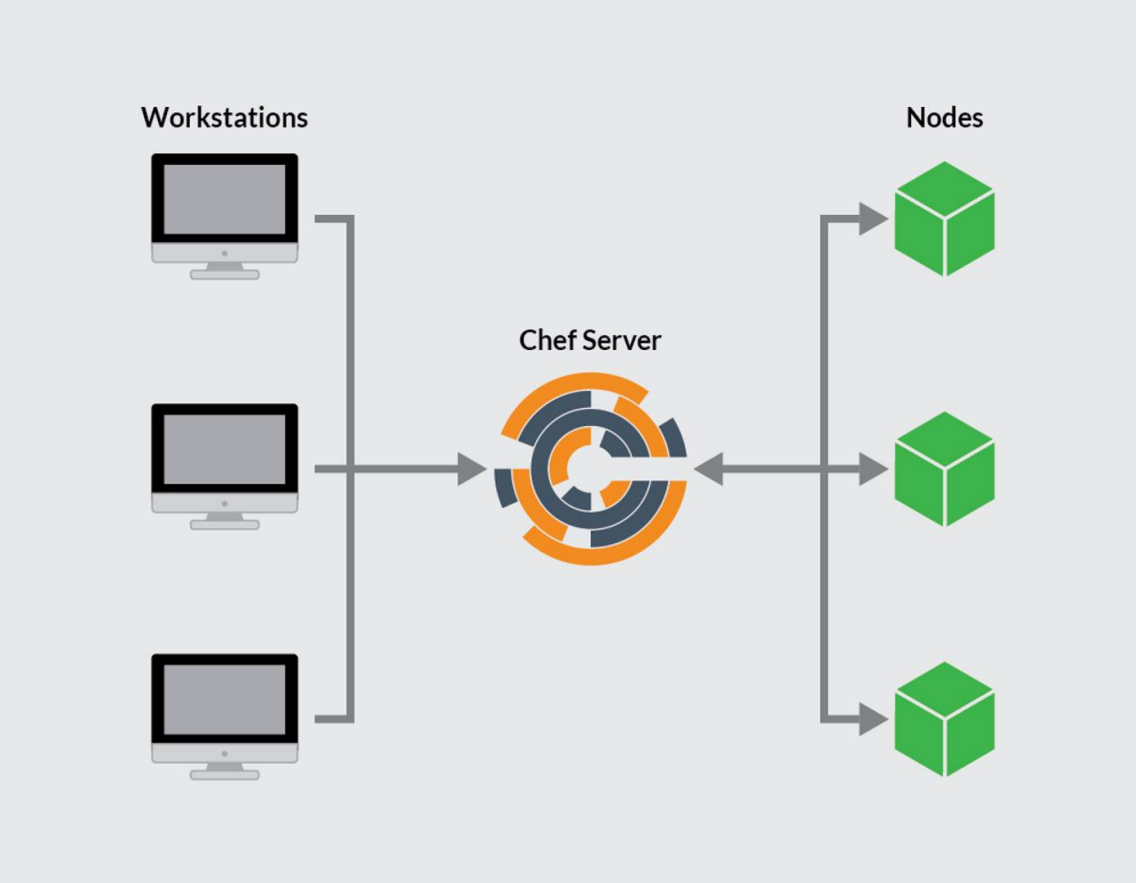

Figure 4: Simplified architecture of Chef Automate [17].

To summarize, Chef workstation is used to develop and maintain Chef environment. Chefs Workstation role is used to develop the Chef Recipes and Chef Cookbooks, which are key building blocks for the Chef environment. These building blocks are used for configuration and policy distribution of Chef Infra. Chef Cookbooks consist of, for instance, all recipes, attributes, custom resources, files and templates needed for Chef Infra node. Chef Recipes works as most important building block of the Cookbook; Chef Recipe consists of all configuration information for the Chef Infra node. Chef Workstation role is to upload Cookbooks to the Chef server using a Chef's knife tool. In other words, Cookbooks contains device-specific attribute libraries for the environment. These attributes specify how system is configured in automation, Chef Infra nodes are distributed with node specific Cookbooks, in question, specify which recipes, files and templates are used. [18] [19]

The Chef Server is responsible for communicating with Chef infra nodes. This communication is done using an agent software called Chef Client, which is installed to the node systems and is responsible for content collection from the Chef Server. Agent is also used to collect data and monitor the status of the nodes by outputting monitoring related messages to the Chef Server regarding incidents, problems and changes. This feedback used to refine in continuous manner in DevOps framework. [20]

Chef has its pros and cons. Centralized management can be used to manage hundreds of servers in a single environment. Chef also maintains blueprints of the entire infrastructure. Complexity comes with its cons; initial setup is complicated because of system complexity and therefore Chef automation requires steep learning curve from the start. The biggest problem compared to Ansible is the use of an agent program, which requires manufacturer support for the device. [21]

2.4.3 Puppet

Puppet is produced by Puppet Inc and shares core philosophy with Chef Automation. Both tools utilize Ruby domain-specific language (DSL) and heavily rely on the agent software for communication between primary server and Agent node. [22]

As show in the Figure 5, Agent software deliveries facts to the Puppet Server (1.), Puppet server distributes Catalog (Puppet Templates) to the Puppet Agent (2.). Puppet Agent reports Agent state to the primary server (3.) in continuous manner. Last, when Catalog matches Puppet Template, Agent workflow goes to the report state (4), which then works as a system monitoring state. [22]

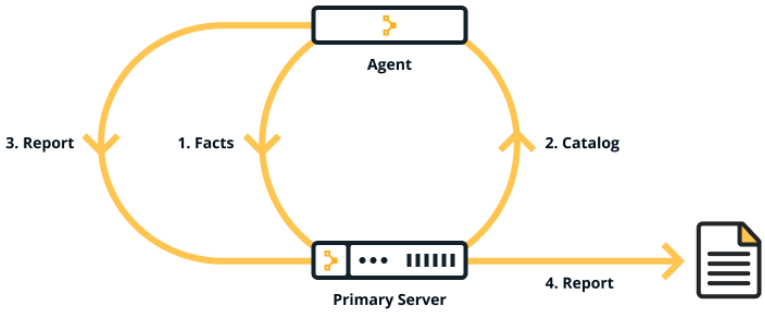

Figure 5: Puppet agent run [22].

Puppet architecture consists of Puppet Server and Puppet agents, which behave in a similar way as in Chef (see Figure 6).

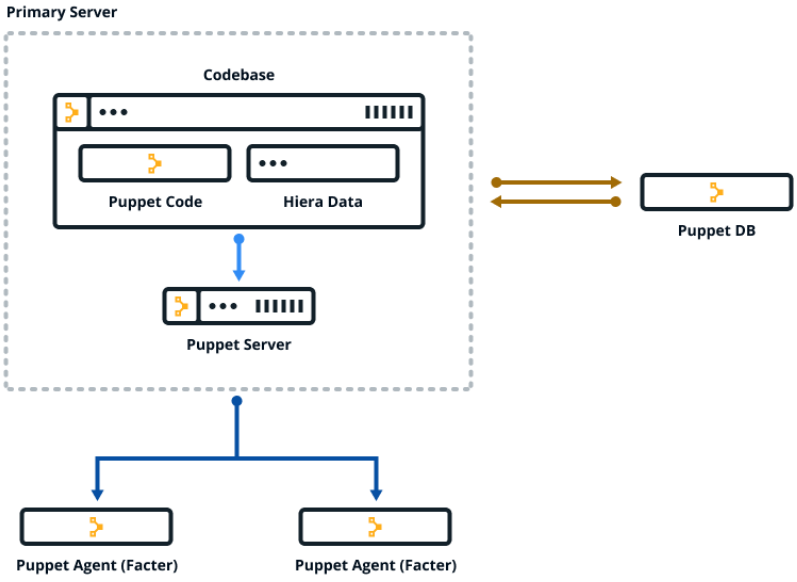

Figure 6: Puppet core architecture [23].

Puppet Server Architecture of Puppet shares the same component roles, Templates work as collection of Puppet manifests, files and modules. These templates share the same roles as Chef Cookbook and are distributed from the

Puppet Master (primary server) to agents (nodes). Role for Manifests is to specify code for the system configuration like with the Chef Recipes. [23] [24]

Puppet Templates are distributed from the Puppet Server, which role in the environment is to communicate and maintain Puppet Agents. [24]

3 Preparations

Ansible was chosen as the technology to perform the practical work because it uses the SSH protocol for communication between devices, and does not require installing a separate agent to the managed devices [20]. This has a huge benefit for environments with a large number of intermediary devices, because the support for agents is usually limited to specific manufacturers [12]. The switch reserved for the practical work has also a manufacturer-implemented support for Ansible, and Ansible does not require a complex system in the background, which thus is a simpler system to use and maintain.

3.1 Ansible Control Node

Ansible Control Node is a host machine which is used to run Ansible tasks on managed nodes. Machine for Control Node role has only few limitations: Ansible code can be run on any machine with support for Python 3.9 or higher. In addition, the Control node role cannot work natively on Windows operating system. Thus, Microsoft Windows devices have to rely on Windows Subsystem for Linux to use Ansible. [24]

3.1.1 System requirements and software dependencies

Before the practical work can be started, a server with Ansible software is needed for the Ansible Control Node role to run Ansible code on managed hosts.

For this practical work, a server with the following properties was allocated from the virtualization platform to the Control Node role (Table 1):

Table 1: List of Control Node hardware properties.

Operating system	Ubuntu LTS Server 22.04.2
CPU	4 cores
RAM	8192 MB
Disk Space	40 GB

Ansible can be run on any Linux machine with Python version +3.9 and thus even commonly used Raspberry Pi, Single Board computer, can be used as a Control Node host [24]. It is recommended to update the server to the latest packages. This can be achieved using distribution package manager. Ubuntu uses apt for package management:

```
sudo apt update
sudo apt upgrade -y
```

Listing 1: Command for updating system

Before Ansible is available on the server, a few packages must be installed on the server. These packages are mainly Python packages and thus it is recommended to use Python Package manager pip as well as to define path for Python environment.

```
echo -e 'export PATH="$PATH:~/.local/bin"' >> .bash_profile
sudo apt install python3-pip
```

Listing 2: Shell commands for pip installation

Pip can be used to install Ansible and its dependencies using the following pip requirements file:

```
ansible
ansible-lint
ansible-pylibssh
paramiko
```

Listing 3: contents of requirements.txt file

Package dependencies can be installed with the following command:

```
pip install -r requirements.txt
```

Listing 4: Shell command for pip package installation from requirements file

File “requirements.txt” makes the installation process easier. After Python dependencies are met, command ansible-galaxy can be used to collect the aoscx collection for Ansible HP Aruba CX-series ansible support:

```
ansible-galaxy collection install arubanetworks.aoscx
```

Listing 5: Shell command for Ansible aoscx plugin

Ansible is now usable and the aoscx collection is installed on the Control Node for the CX-series switch management. [24] [25] [26]

3.1.2 AOSCX Ansible Collection

Ansible Collection is a distribution format for Ansible Playbooks, roles, modules and plugins. These can be installed by using an Ansible Galaxy or a Pulp 3 Galaxy server. [27]

The support for Aruba CX switch series is distributed as a separate collection for Ansible. This collection provides support for SSH or REST-API switch management. [26]

API-support is divided to two different plug-ins, which means that the use of both management protocols are limited for one Ansible Playbook task. REST-API uses arubanetworks.aoscx.aoscx and SSH uses network_cli plugin as a connection method. [26]

SSH can use two different modules, which are aoscx_config and aoscx_command. [25]

3.1.2.1 aoscx_config

Command “aoscx_config” is used to configure “running-config” on the AOS-CX device. Command supports child and parent lines (commands for configuring switch), and can be configured to execute commands after and before main commands, which makes it useful when user is prompted with questions regarding the change. [28]

3.1.2.2 aoscx_command

Command “aoscx_command” executes any AOS-CX CLI command on any AOS-CX device. The only limitation is the situation, in which the switch operating system requires feedback with other than yes/no confirmation to a command after the command execution. “aoscx_config” is more suitable for prompt specific commands configuring “running-config”. [29]

3.1.2.3 REST-API

REST-API plugin has large share of different modules for AOS-CX device configuration. Modules range from setting an ACL rules, creating VLANs and gathering facts about the switch. [30]

3.2 CX6100 series switch

Figure 7: CX6100 switch series [31].

Aruba CX 6100 switch series are marketed as low cost, entry level enterprise access switch series. Series switches have support for wide range of networking technologies, which are commonly expected from enterprise grade networking devices. These features include, for example, 1/10 Gbps SFP+ uplinks, IEEE 802.3at Class 4 PoE support, ACL, 802.1x user authentication, Layer 2 switching including VLAN (IEEE 802.1Q) and jumbo packet size up to 9220 bytes, Layer 3 static routing including support for ARP and LLDP support for neighbor discovery and SNMP for third party monitoring support. [31] [32]

3.2.1 Device Features

A switch supports configuration via GUI and CLI. CLI support is limited to SSH-protocol. A switch also supports REST-API which can be used to remote configuration via HTTP/HTTPS protocol. CX-series switches are equipped with the Aruba AOS-CX operating system. [31] [32]

Table 2: Switch specifications.

Vendor	HPe Aruba
Model	JL677A 6100 24G CL4 4SFP+ Swch
Firmware	PL.10.11.1005
Ports	24 RJ45 + 4 SFP+

Table 2 presents a switch model and a firmware version for the practical work.

The CLI syntax is similar to older HPe Comware devices running Aruba-OS (old HP ProVision OS), but it also shares similarities with Cisco IOS based devices. The syntax itself is out of scope of this thesis, but HPe provides a document which lists all differences between all mentioned manufacturers and ecosystems. [33] [34]

3.2.2 Switch preconfiguration

The switch has to be preconfigured before Ansible can be used to configure the host. Following steps were configured manually on to the device:

- Admin account was configured with a password
- Management VLAN was created
- Management interface was created
    - Static IP-address was set to the management interface
- Default route was set to point to the default gateway on the network

3.3 Lab environment

The lab environment for the practical work of the thesis is simple and structured as follows. At the OSI L2 level (Figure 8), the server is in its own network

segment and the switch management interface is set to dedicated management network. SSH traffic from Ansible Control Node network is allowed to management network.

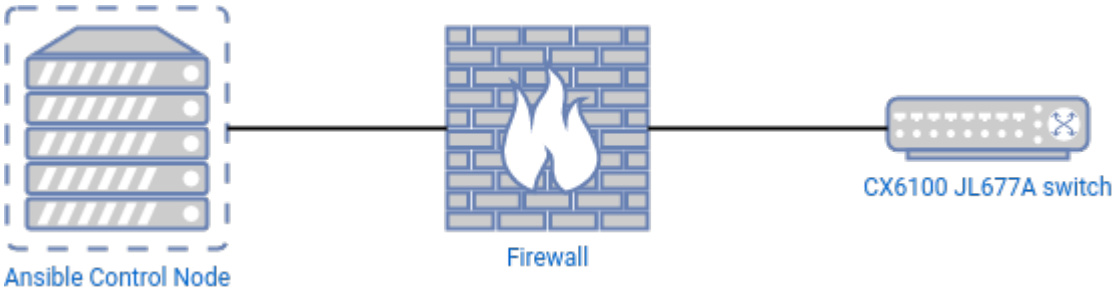

Figure 8: L2 graph of the environment.

Ansible Control Node is accessible using SSH from network device dedicated for Ansible code development.

4 Practical work

The practical work started with the basics: For Ansible programming, a separate project folder was created with the following files:

- “configure_aoscx_playbook.yaml” as a Ansible Playbook file
- “inventory.yaml” as a Ansible inventory file
- Folder name “templates”
    - “arubacx_switch_conf_template.j2” as a Jinja2 template file

As shown on the list, the project is saved on multiple YAML files. Ansible uses yaml as a file-format for its files. This means that the text in the files is indentation sensitive. [35]

The following goals were set for the program:

- Collect interface information from the device
- Set a login banner on the device
- Create two new VLANs with names test1 and test2
- Set the last six port as trunk ports
- Set rest of the ports as access ports
- Set Spanning tree active on all of the switch access ports

The program also has to be fail-safe to use, for instance it must stop executing if Ansible Play fails. To illustrate the project graphically, following flowchart was created (Figure 9).

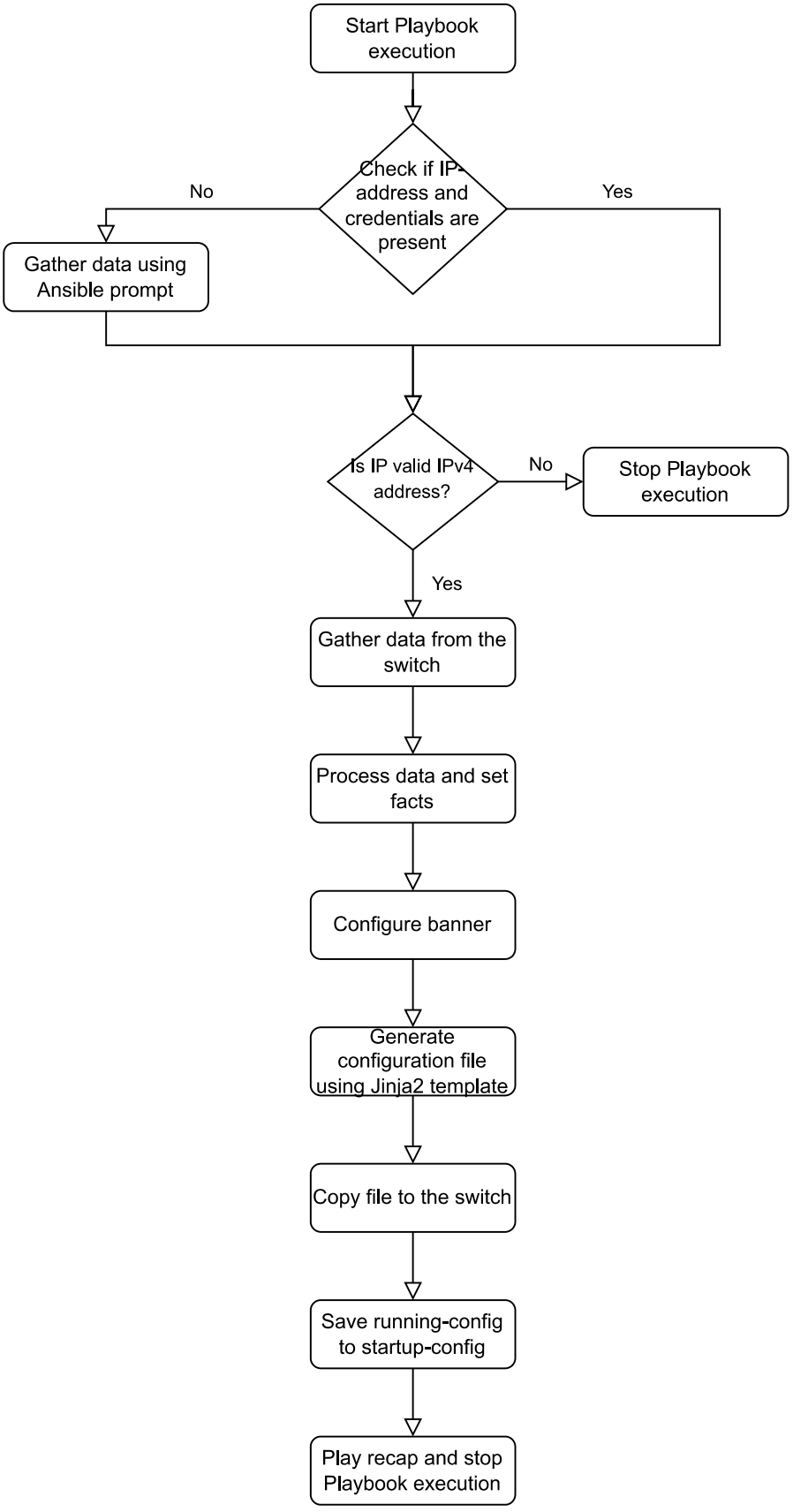

Figure 9: Flowchart of the Ansible program.

In addition, Ansible Playbook has to be coded in such a manner, in which it can be also used manually from the Ansible Control Node and thus provide some sort of prompt for the user.

4.1 Ansible Inventory file

The practical work started by creating an Ansible Inventory file for the project. File follows guidelines detailed in the Ansible aoscx collection and is made to be as simple as possible for demonstration purposes. Content of the “inventory.yaml” is displayed on Listing 6.

```
all:
  hosts:
    aoscx_1:
      ansible_network_os: arubanetworks.aoscx.aoscx # Pointer to
platform specific Ansible collection
      ansible_connection: network_cli  # SSH connection method
      ansible_aoscx_validate_certs: True # BYPASS certificate
validation
      ansible_acx_no_proxy: False # Bypass environment proxies
      ansible_aoscx_use_proxy: False # Set if proxy is needed
```

Listing 6: Ansible inventory file “inventory.yaml”.

Ansible collection type is set to “network_cli” as SSH is used. Switch management interface is not behind any proxy and no additional SSH certificates are used for the practical work, so these boolean values are set accordingly. Switch IP-address is not set in this file, address is passed to the program as a command line flag, or the information is asked from the user via prompt if not present manually.

4.2 Jinja2 template

Ansible supports Jinja2 templating language, which enables the use of dynamic expressions on Ansible. This makes it a great tool for, i.e., the creation of configuration file, when variables with dynamic content are used. [36]

The Jinja2 output content is similar to Python syntax and thus is yaml compatible. AOS-CX series switches “running-config” are stored in a similar format, as indentation is used to define the sections of the device configuration more precisely. [37]

Jinja2 is used to provide a configuration file for the switch, in which it can be used to configure multiple lines per task. This considerably speeds up the configuration of the switch in Ansible automation, as every Ansible task creates a new SSH session for the device.

The Jinja2 template uses the port information collected from the device dynamically and sets the last six ports as trunk ports. Out of these six ports, two are RJ45 ports and four are SFP+ ports. Content of the “arubacx_switch_conf_template.j2” is displayed on Listing 7.

```
!
{% for int_name in sw_ports %}
{% if loop.index0 < loop.length - 6 %}
interface {{ int_name }}
    vlan port access 1
    spanning-tree bpdu-guard
    spanning-tree rpvst-guard
    spanning-tree port-type admin-edge
{% endif %}
{% if loop.index0 >= loop.length - 6 %}
interface {{ int_name }}
    vlan trunk native 1
    vlan trunk allowed all
{% endif %}
{% endfor %}
!
spanning-tree
!
vlan 2
    name test2
vlan 3
    name test3
!
interface vlan1
    shutdown
    no ip dhcp
!
```

Listing 7: Content of arubacx_switch_conf_template.j2 Jinja2 template file.

Interface configuration is done using a basic loop embedded with conditional IF-statements. Spanning tree protocol is set to be active and VLAN creation is done at the end of the file.

The configuration file is created in the "templates" folder and is copied to the device in a separate task.

4.3 Ansible Playbook file

Ansible Playbook contains eight different tasks. Tasks are explained below.

Role on this Playbook is set to “all” like in the Ansible inventory file (Listing 8). This means that all tasks are executed on the host and no role-specific tasks are executed. User prompt is activated if a variable is empty.

```
- hosts: all
  vars_prompt:
    - name: ansible_host
      prompt: Enter IP-address
      private: False

    - name: ansible_user
      prompt: "Enter local User Account"
      private: False

    - name: ansible_password
      prompt: "Enter password"
      private: True
      confirm: True

  collections: # Set collections
    - arubanetworks.aoscx
  gather_facts: False # Disable facts gathering from control node
```

Listing 8: Ansible user input prompt.

The next task in Listing 9 tests if the variable “ansible_host” contains a valid IP-address. Condition is saved as a Boolean value to variable “host_ipv4_valid”.

```
  tasks:
  - name: Check if host address is a valid IPv4 address
```

```
    ansible.builtin.set_fact:
      host_ipv4_valid: "{{ ansible_host is ansible.utils.ipv4 }}"
```

Listing 9: Task to test if given address valid IPv4 IP-address.

The task in Listing 10 gathers information about LLDP neighbor and switch interface from the switch and sets the “checkpoint” mode on “auto”. The checkpoint mode works as a fail-safe mechanism: if connection to the switch drops or fails and checkpoint is not cleared during the five minutes timer, switch will rollback “running-config” to the state before any changes.

```
  - name: Gather data from the switch
    aoscx_command:
      commands:
        - 'show lldp neighbor-info' # Get switch neighboring devices
        - 'show interface brief' # Get switch interface information
        - 'checkpoint auto 5'
      output_file_format: json
    register: raw_sw_port_data # register raw data from switch as a
dictionary
    when: host_ipv4_valid is true # condition, only execute if IP-
address valid
```

Listing 10: Task to gather data from the switch.

The task in Listing 11 processes the data gathered from the switch and sets it to Ansible facts. Ansible facts are used to store information about remote system. AOS-CX SSH plug-in does not support automatic gathering of any facts from the switch, which means that the custom fact has to be set by the task manually. [38]

```
  - name: Set facts # Get host port and lldp neighbour data from the
dictionary
    ansible.builtin.set_fact:
      host_lldp_info: "{{ raw_lldp_info }}" # switch lldp neighbor
info
      host_interface_info: "{{ raw_host_interface_info }} " # switch
interface information
    vars:
      raw_lldp_info: "{{ raw_sw_port_data.stdout_lines[0] |
select('match', '([0-9]+(/[0-9]+)+)') |  map('regex_replace', ' +','
') | map('split', ' ') }}" # switch lldp neighbor information
      raw_host_interface_info: "{{ raw_sw_port_data.stdout_lines[1] |
select('match', '([0-9]+(/[0-9]+)+)') | map('regex_replace', ' +',' ')
| map('split',' ') }} " # switch interface information
```

Listing 11: Task to handle raw data gathered from the switch.

In the next task, Ansible continues to process fact variables. In Listing 12 Ansible Control node loops through the “host_interface_info” containing list of a lists and saves the first variable from the list to “sw_ports” list. This list is later used to create dynamic expressions in Jinja2 templating engine in the Listing 14. The template is detailed in chapter 4.2.

```
  - name: Set switch ports to facts
    set_fact:
      sw_ports: "{{ sw_ports | default([]) + [item[0] | string] }}"
    loop: "{{ host_interface_info }}"
```

Listing 12: Task set port information to Ansible fact.

Task in Listing 13 configures banner message to the switch using aoscx_config module. Message is a two-line warning message. The change is not saved to startup-config.

```
  - name: Configure banner message # Configures a multiline banner
message
    aoscx_config:
      lines:
        - AUTHORIZED ACCESS ONLY
        - Disconnect IMMEDIATELY if you are not an authorized user!
      before: "banner motd `"
      after: "`"
      save_when: never
```

Listing 13: Set banner to the switch.

The task in Listing 14 generates a switch configuration file using Jinja2 templating engine. The operation of the template machine is discussed in Section 4.2. Configuration file is saved to the folder “templates”, and file permission is set to mode “0777”, which means that all users of the machine have read, write, and execute rights to the file.

```
  - name: Generate Template for switch configuration # Generates
switch configuration file from jinja2 template
```

```
    template: src="template/arubacx_switch_conf_template.j2"
dest="template/sw_aoscx.conf" mode='0777'
```

Listing 14: Generate template file for the switch.

The configuration file generated in the Listing 14 is copied in the Listing 15 to the switch. The copy process is purely additive, so configuration lines missing from the file are not removed from the switch “running-config”[28].

```
  - name: Copy generated switch configuration file to the switch via
SSH # Appends lines from jinja2 template file
    aoscx_config:
      src: "template/sw_aoscx.conf"
      save_when: never
```

Listing 15: Copy generated configuration file to the switch.

The Control Node confirms “checkpoint auto”, which thus disables rollback function and makes switch to save running-config to startup-config (Listing 16).

```
  - name:  Save configuration
    aoscx_command:
      commands:
- 'checkpoint auto confirm'
```

Listing 16: Save configuration to startup-config.

After the last task is completed, Ansible prints recap to the command line interface. This recap shows whether the Ansible play was successful.

4.4 Ansible Playbook execution

Ansible prints the progress of the application to the command line interface. Progress and Play Recap can be found in Listing 17 below.

```
PLAY [all]
*********************************************************************

TASK [Check if host address is a valid IPv4 address]
*************************
ok: [aoscx_1]
```

```
TASK [Gather data from the switch]
*****************************************
ok: [aoscx_1]

TASK [Set facts]
************************************************************
ok: [aoscx_1]

TASK [Set switch ports to facts]
*******************************************
ok: [aoscx_1] => (item=['1/1/1', '1', 'access', '1GbT', 'yes', 'down',
'Waiting', 'for', 'link', '--', '--'])
ok: [aoscx_1] => (item=['1/1/2', '1', 'access', '1GbT', 'yes', 'down',
'Waiting', 'for', 'link', '--', '--'])
ok: [aoscx_1] => (item=['1/1/3', '1', 'access', '1GbT', 'yes', 'down',
'Waiting', 'for', 'link', '--', '--'])
ok: [aoscx_1] => (item=['1/1/4', '1', 'access', '1GbT', 'yes', 'down',
'Waiting', 'for', 'link', '--', '--'])
ok: [aoscx_1] => (item=['1/1/5', '1', 'access', '1GbT', 'yes', 'down',
'Waiting', 'for', 'link', '--', '--'])
ok: [aoscx_1] => (item=['1/1/6', '1', 'access', '1GbT', 'yes', 'down',
'Waiting', 'for', 'link', '--', '--'])
ok: [aoscx_1] => (item=['1/1/7', '1', 'access', '1GbT', 'yes', 'down',
'Waiting', 'for', 'link', '--', '--'])
ok: [aoscx_1] => (item=['1/1/8', '1', 'access', '1GbT', 'yes', 'down',
'Waiting', 'for', 'link', '--', '--'])
ok: [aoscx_1] => (item=['1/1/9', '1', 'access', '1GbT', 'yes', 'down',
'Waiting', 'for', 'link', '--', '--'])
ok: [aoscx_1] => (item=['1/1/10', '1', 'access', '1GbT', 'yes',
'down', 'Waiting', 'for', 'link', '--', '--'])
ok: [aoscx_1] => (item=['1/1/11', '1', 'access', '1GbT', 'yes',
'down', 'Waiting', 'for', 'link', '--', '--'])
ok: [aoscx_1] => (item=['1/1/12', '1', 'access', '1GbT', 'yes',
'down', 'Waiting', 'for', 'link', '--', '--'])
ok: [aoscx_1] => (item=['1/1/13', '1', 'access', '1GbT', 'yes',
'down', 'Waiting', 'for', 'link', '--', '--'])
ok: [aoscx_1] => (item=['1/1/14', '1', 'access', '1GbT', 'yes',
'down', 'Waiting', 'for', 'link', '--', '--'])
ok: [aoscx_1] => (item=['1/1/15', '1', 'access', '1GbT', 'yes',
'down', 'Waiting', 'for', 'link', '--', '--'])
ok: [aoscx_1] => (item=['1/1/16', '1', 'access', '1GbT', 'yes',
'down', 'Waiting', 'for', 'link', '--', '--'])
ok: [aoscx_1] => (item=['1/1/17', '1', 'access', '1GbT', 'yes',
'down', 'Waiting', 'for', 'link', '--', '--'])
ok: [aoscx_1] => (item=['1/1/18', '1', 'access', '1GbT', 'yes',
'down', 'Waiting', 'for', 'link', '--', '--'])
ok: [aoscx_1] => (item=['1/1/19', '1', 'access', '1GbT', 'yes',
'down', 'Waiting', 'for', 'link', '--', '--'])
ok: [aoscx_1] => (item=['1/1/20', '1', 'access', '1GbT', 'yes',
'down', 'Waiting', 'for', 'link', '--', '--'])
ok: [aoscx_1] => (item=['1/1/21', '1', 'access', '1GbT', 'yes',
'down', 'Waiting', 'for', 'link', '--', '--'])
ok: [aoscx_1] => (item=['1/1/22', '1', 'access', '1GbT', 'yes',
'down', 'Waiting', 'for', 'link', '--', '--'])
ok: [aoscx_1] => (item=['1/1/23', '1', 'trunk', '1GbT', 'yes', 'up',
'1000', '--'])
ok: [aoscx_1] => (item=['1/1/24', '1', 'trunk', '1GbT', 'yes', 'up',
'1000', '--'])
```

```
ok: [aoscx_1] => (item=['1/1/25', '1', 'access', '--', 'yes', 'down',
'No', 'XCVR', 'installed', '--', '--'])
ok: [aoscx_1] => (item=['1/1/26', '1', 'access', '--', 'yes', 'down',
'No', 'XCVR', 'installed', '--', '--'])
ok: [aoscx_1] => (item=['1/1/27', '1', 'access', '--', 'yes', 'down',
'No', 'XCVR', 'installed', '--', '--'])
ok: [aoscx_1] => (item=['1/1/28', '1', 'access', '--', 'yes', 'down',
'No', 'XCVR', 'installed', '--', '--'])

TASK [Configure banner message]
*********************************************
changed: [aoscx_1]

TASK [Generate Template for switch configuration]
*****************************
changed: [aoscx_1]

TASK [Copy generated switch configuration file to the switch via SSH]
**********
changed: [aoscx_1]

TASK [Save configuration]
***************************************************
ok: [aoscx_1]

PLAY RECAP
*********************************************************************
aoscx_1                    : ok=8    changed=3    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

Listing 17: Ansible Playbook execution.

As seen from the Ansible “PLAY RECAP” on Listing 17, all eight tasks returned with a status “ok”, and three changes were made to the remote system. Ansible Play was successful. The end result of the Ansible Play is detailed in Appendix 1 by running “show running-config” command on the switch CLI. It can be verified in Appendix 1 that the configuration of the device was successful.

5 Results and future improvements

The practical work of this thesis demonstrated well how Ansible can be used as a tool to configure and manage environment with Aruba AOS-CX devices. The code is easily scalable, and the future development using Ansible is almost unlimited due to the fact that Ansible as a tool provides simple yet powerful features for the DevOps developer. Tools can be used exceptionally well to match more challenging needs set by the network environment. Ansible and RedHat provide solutions for Ansible Control Node management and managed host automation.

Jinja2 enables the use of dynamic expressions in Ansible, which also makes it a key role in Ansible automation when implementing, for example, more site specific differences within the network environment. However, using Jinja2 introduces more complexity and requires more collection of data from the managed host, or at least needs to rely on some sort of database which stores necessary information about the managed host differences.

The thesis and, more specifically, the practical work did not compare the time difference between the automated and manual device configuration. From the point of view of further development, it would be good to research further how the obtained benefits and disadvantages compare to each other. My estimation is that the more complex the configuration is, the greater the benefit is achieved from automating the configuration of the device.

In addition, from the point of view of the future project improvement, utilizing Ansible Roles and developing the security of the automation, such as using Ansible Vault, would be preferred. The Playbook for the switch configuration could be improved, for example, by adding LLDP information to the ports where a device that supports the LLDP protocol, creating ACL rules for the device management access, implementing the radius for management, and using the REST API for collecting Ansible Facts, for example.

6 Conclusion

The practical work achieved the goals set for this thesis. Ansible Control Node was successfully used to configure network device remotely. Practical work followed the DevOps methodology and IaC model. Configuration of the network device can be automated using the Ansible Playbook created for the project. Also, Playbook can be used on different devices and it can replicate the end result of the device used in this thesis.

It is good to note that if the work is viewed from a DevOps perspective, this thesis reviewed only a small part of the DevOps philosophy. The philosophy is based on the continuous development and contains several stages and best practices taken in the development pipeline.

Chapter 5 pointed out several future improvements for the topic. In my opinion the improvements in question are worth to be implemented in the future.

I personally think that I benefited a lot from this thesis. Although the subject is familiar and has been covered in studies, the research done for this thesis has helped me to understand more clearly the benefits of automation and the DevOps mentality.

References

1 Jacobs Mike, EE Kathryn, Petersen Theano, Kaim Ed, Danielson Steve, Hellem Dan. 2023. What is DevOps? - Azure DevOps. Microsoft Learn. Online. <https://learn.microsoft.com/en-us/devops/what-is-devops>. Accessed 27.3.2023.

2 Amazon.com Inc. 2023. What is DevOps?. Online. <https://aws.amazon.com/devops/what-is-devops/>. Accessed 27.3.2023.

3 Kim Gene, Allspaw John, Debois Patrick, Humble Jez, Willis John. 2016. The DevOps Handbook. E-Book. Portland IT Revolution Press. ISBN 1-942788-00-2

4 Courtemanche Meredith, Mell Emily, Gillis Alexander S. 2021. What is DevOps? The ultimate guide. TechTarget IT Operations. Online. <https://www.techtarget.com/searchitoperations/definition/DevOps>. Accessed 27.3.2023.

5 Jacobs Mike, Kulla-Mader Julia, Petersen Theano, Kaim Ed. 2022. What is infrastructure as code (IAC)? - Azure DevOps. Microsoft Learn. <https://learn.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code>. Accessed 28.3.2023.

6 Red Hat Inc. 2022. What is infrastructure as code (IaC)?. Online. <https://www.redhat.com/en/topics/automation/what-is-infrastructure-as-code-iac>. Accessed 28.3.2023.

7 Groover Mikell. 2023. Automation. Encyclopedia Britannica. Online. <https://www.britannica.com/technology/automation>. Accessed 28.3.2023.

8 Froehlich Andrew. 2022. Compare 8 network automation tools. TechTarget Networking. Online. https://www.techtarget.com/searchnetworking/feature/The-8-leading-options-in-network-automation-tools>. Accessed 05.4.2023.

9 Red Hat Inc. 2022. Understanding automation. Online. <https://www.redhat.com/en/topics/automation>. Accessed 28.3.2023.

10 Red Hat Inc. 2022. Ansible vs. Terraform, clarified. Online. <https://www.redhat.com/en/topics/automation/ansible-vs-terraform>. Accessed 29.3.2023.

11 Freeman James, Keating Jesse. 2016. Mastering Ansible - Fourth Edition. Packt Publishing. ISBN 1-80181-878-9

12 Ansible Inc. 2023. How it works. Online. <https://www.ansible.com/overview/how-ansible-works>. Accessed 30.3.2023.

13 Ansible Project. 2023. Basic concepts - Ansible Documentation. Online. <https://docs.ansible.com/ansible/latest/network/getting_started/basic_concepts.html>. Accessed 30.3.2023.

14 Ansible Project. 2023. Getting started with Ansible - Ansible Documentation. Online. <https://docs.ansible.com/ansible/latest/getting_started/index.html>. Accessed 30.3.2023.

15 Ansible Inc. 2023. Ansible for configuration management. Online. <https://www.ansible.com/use-cases/configuration-management>. Accessed 30.3.2023.

16 Tutorials Point. Chef - Overview. Online. <https://www.tutorialspoint.com/chef/chef_overview.htm>. Accessed 03.4.2023.

17 Progress Software Corporation. 2022. Chef Infra Overview. Online. <https://docs.chef.io/chef_overview/>. Accessed 03.4.2023.

18 Krout Elle. 2023. A beginner's Guide to Chef. Linode Guides & Tutorials. Online. <https://www.linode.com/docs/guides/beginners-guide-chef/>. Accessed 03.4.2023.

19 Progress Software Corporation. 2022. About Cookbooks. Online. <https://docs.chef.io/cookbooks/>. Accessed 03.4.2023.

20 Red Hat Inc. Ansible vs. chef: What you need to know. Online. <https://www.redhat.com/en/topics/automation/ansible-vs-chef>. Accessed 03.4.2023.

21 Intellipaat Blog. 2023. What is chef? DevOps tool for Configuration Management. Online. <https://intellipaat.com/blog/what-is-chef/>. Accessed 03.4.2023.

22 Cadman Claire. 2022. What is Puppet. Introduction to Puppet. Online. <https://www.puppet.com/docs/puppet/6/puppet_overview.html#what_is_puppet>. Accessed 04.4.2023.

23 Cadman Claire. 2022. The Puppet platform. Introduction to Puppet. Online. <https://www.puppet.com/docs/puppet/6/puppet_overview.html#platform_components>. Accessed 04.4.2023.

24 Ansible Project. 2023. Installing Ansible - Ansible Documentation. <https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#control-node-requirements>. Accessed 04.4.2023.

25 Venugopal Madhusudan Pranav, Liu Yang, Chiapuzio-Wong Tiffany, Wang Derek, Gutierrez Melvin, Hernandez Rodrigo Jose, Bonilla Daniel Alvarado/ Aruba Networks. 2023. Aruba/AOSCX-ansible-collection: Ansible collections for AOS-CX switches. GitHub. Aruba Networks. Online. <https://github.com/aruba/aoscx-ansible-collection>. Accessed 05.4.2023.

26 Venugopal Madhusudan Pranav, Liu Yang, Chiapuzio-Wong Tiffany, Wang Derek, Gutierrez Melvin, Hernandez Rodrigo Jose, Bonilla Daniel Alvarado/ Aruba Networks. 2023. AOSCX Ansible Collection. Ansible Galaxy. Online. <https://galaxy.ansible.com/arubanetworks/aoscx>. Accessed 05.4.2023.

27 Ansible Project. 2023. Using Ansible collections - Ansible Documentation. Online. <https://docs.ansible.com/ansible/latest/collections_guide/index.html>. Accessed 05.4.2023.

28 Chiapuzio-Wong Tiffany / Aruba Networks. 2020. AOSCX-Ansible-role/aoscx_config.MD at master · Aruba/AOSCX-ansible-role. GitHub. Online. <https://github.com/aruba/aoscx-ansible-role/blob/master/docs/aoscx_config.md>. Accessed 05.4.2023.

29 Chiapuzio-Wong Tiffany / Aruba Networks. 2020. AOSCX-Ansible-role/AOSCX_COMMAND.MD at master · Aruba/AOSCX-ansible-role. GitHub. Online. <https://github.com/aruba/aoscx-ansible-role/blob/master/docs/aoscx_command.md>. Accessed 05.4.2023.

30 Chiapuzio-Wong Tiffany / Aruba Networks. 2022. AOSCX-Ansible-role/docs at master · Aruba/AOSCX-ansible-role. GitHub. Online. <https://github.com/aruba/aoscx-ansible-role/tree/master/docs>. Accessed 05.4.2023.

31 Hewlett Packard Enterprise Development LP. 2023. Aruba CX 6100 switch series. Online. <https://www.hpe.com/psnow/doc/a00021859enw.html?jumpid=in_pdp-psnow-qs>. Accessed 06.4.2023.

32 Hewlett Packard Enterprise Development LP. 2023. Aruba 6100 Switch Series - Overview. Document Display | HPE Support Center. Online. <https://support.hpe.com/hpesc/public/docDisplay?docId=a00110388en_us&docLocale=en_US>. Accessed 06.4.2023.

33 Hewlett Packard Enterprise Development. 2023. AOS-CX Overview. Online. <https://www.arubanetworks.com/techdocs/central/latest/content/nms/aos-cx/cfg/conf-cx-switches.htm>. Accessed 06.4.2023.

34 Hewlett Packard Enterprise Development LP. 2019. CLI Reference Guide for arubaos-CX, ArubaOS-Switch, and Cisco Ios. Online. <https://www.hpe.com/psnow/doc/c04793912>. Accessed 06.4.2023.

35 Ansible Project. 2023. Glossary - YAML. Glossary - Ansible Documentation. Online. <https://docs.ansible.com/ansible/latest/reference_appendices/glossary.html#term-YAML>. Accessed 07.4.2023.

36 Ansible Project. 2023. Templating (Jinja2) - Ansible Documentation. Online. <https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_templating.html>. Accessed 07.4.2023.

37 Pallets. Jinja2. Online. <https://jinja.palletsprojects.com/en/3.1.x/>. Accessed 08.4.2023.

38 Ansible Project. 2023. Discovering variables: Facts and magic variables - Ansible Documentation. Online. <https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_vars_facts.html>. Accessed 08.4.2023.

Switch “running-config”

```
6100# show running-config
Current configuration:
!
!Version ArubaOS-CX PL.10.11.1005
!export-password: default
banner motd #
AUTHORIZED ACCESS ONLY
Disconnect IMMEDIATELY if you are not an authorized user!
#
user admin group administrators password ciphertext
AQBapblCNtN3NyMkkHd/EE5slNNmGupfuW5RTtFBCLsXramxYgAAAO8pnp+4i79gMWMfVow2QFLO/
rQhMWNqJAm/JTHILVNp/
3anozQI+v4bWqgkzjjwm+bl7x3Y0aJxVyGysrwtSrvhLUY6z3MVUwDpIE0C8fSoL1ktOqhgJvaQ5yv1g
Z/DXogl
ntp server pool.ntp.org minpoll 4 maxpoll 4 iburst
ntp enable
!
!
!
!
!
!
ssh server vrf default
vlan 1
vlan 2
    name test2
vlan 3
    name test3
spanning-tree
interface 1/1/1
    no shutdown
    vlan access 1
    spanning-tree bpdu-guard
    spanning-tree rpvst-guard
    spanning-tree port-type admin-edge
interface 1/1/2
    no shutdown
    vlan access 1
    spanning-tree bpdu-guard
    spanning-tree rpvst-guard
    spanning-tree port-type admin-edge
interface 1/1/3
    no shutdown
    vlan access 1
    spanning-tree bpdu-guard
    spanning-tree rpvst-guard
    spanning-tree port-type admin-edge
interface 1/1/4
    no shutdown
    vlan access 1
    spanning-tree bpdu-guard
    spanning-tree rpvst-guard
```

```
    spanning-tree port-type admin-edge
interface 1/1/5
    no shutdown
    vlan access 1
    spanning-tree bpdu-guard
    spanning-tree rpvst-guard
    spanning-tree port-type admin-edge
interface 1/1/6
    no shutdown
    vlan access 1
    spanning-tree bpdu-guard
    spanning-tree rpvst-guard
    spanning-tree port-type admin-edge
interface 1/1/7
    no shutdown
    vlan access 1
    spanning-tree bpdu-guard
    spanning-tree rpvst-guard
    spanning-tree port-type admin-edge
interface 1/1/8
    no shutdown
    vlan access 1
    spanning-tree bpdu-guard
    spanning-tree rpvst-guard
    spanning-tree port-type admin-edge
interface 1/1/9
    no shutdown
    vlan access 1
    spanning-tree bpdu-guard
    spanning-tree rpvst-guard
    spanning-tree port-type admin-edge
interface 1/1/10
    no shutdown
    vlan access 1
    spanning-tree bpdu-guard
    spanning-tree rpvst-guard
    spanning-tree port-type admin-edge
interface 1/1/11
    no shutdown
    vlan access 1
    spanning-tree bpdu-guard
    spanning-tree rpvst-guard
    spanning-tree port-type admin-edge
interface 1/1/12
    no shutdown
    vlan access 1
    spanning-tree bpdu-guard
    spanning-tree rpvst-guard
    spanning-tree port-type admin-edge
interface 1/1/13
    no shutdown
    vlan access 1
    spanning-tree bpdu-guard
    spanning-tree rpvst-guard
    spanning-tree port-type admin-edge
interface 1/1/14
    no shutdown
    vlan access 1
```

```
    spanning-tree bpdu-guard
    spanning-tree rpvst-guard
    spanning-tree port-type admin-edge
interface 1/1/15
    no shutdown
    vlan access 1
    spanning-tree bpdu-guard
    spanning-tree rpvst-guard
    spanning-tree port-type admin-edge
interface 1/1/16
    no shutdown
    vlan access 1
    spanning-tree bpdu-guard
    spanning-tree rpvst-guard
    spanning-tree port-type admin-edge
interface 1/1/17
    no shutdown
    vlan access 1
    spanning-tree bpdu-guard
    spanning-tree rpvst-guard
    spanning-tree port-type admin-edge
interface 1/1/18
    no shutdown
    vlan access 1
    spanning-tree bpdu-guard
    spanning-tree rpvst-guard
    spanning-tree port-type admin-edge
interface 1/1/19
    no shutdown
    vlan access 1
    spanning-tree bpdu-guard
    spanning-tree rpvst-guard
    spanning-tree port-type admin-edge
interface 1/1/20
    no shutdown
    vlan access 1
    spanning-tree bpdu-guard
    spanning-tree rpvst-guard
    spanning-tree port-type admin-edge
interface 1/1/21
    no shutdown
    vlan access 1
    spanning-tree bpdu-guard
    spanning-tree rpvst-guard
    spanning-tree port-type admin-edge
interface 1/1/22
    no shutdown
    vlan access 1
    spanning-tree bpdu-guard
    spanning-tree rpvst-guard
    spanning-tree port-type admin-edge
interface 1/1/23
    no shutdown
    vlan trunk native 1
    vlan trunk allowed all
interface 1/1/24
    no shutdown
    vlan trunk native 1
```

```
    vlan trunk allowed all
interface 1/1/25
    no shutdown
    vlan trunk native 1
    vlan trunk allowed all
interface 1/1/26
    no shutdown
    vlan trunk native 1
    vlan trunk allowed all
interface 1/1/27
    no shutdown
    vlan trunk native 1
    vlan trunk allowed all
interface 1/1/28
    no shutdown
    vlan trunk native 1
    vlan trunk allowed all
interface vlan 1
    shutdown
    no ip dhcp
interface vlan 999
    ip address 10.0.99.10/24
ip route 0.0.0.0/0 10.0.99.1
!
!
!
!
!
no https-server vrf default
```