

**SOFTWARE DEFINED WIDE AREA NETWORKS**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ZAHRA KHALID

ZEYNEP TEMIZEL

CONVERGED OPTIMIZED NETWORKS 15 HP

HALMSTAD UNIVERSITY

SUPERVISOR: OVE ANDERSSON

**1.ABSTRACT**

This project is about:

* Showing how efficient software-defined wide area networks are by comparing them with traditional wide area networks.
* Creating an SDN topology with the help of ansible in DevNET’s virtual environment sandbox.

In practical work, we configure a whole topology of 10 devices with a couple of commands. It took less than a minute. This amount of configuration could have take hours to do one by one to all devices. New services sometimes took months to be available for all branches. But with SD-WAN, it is fast as a blink.



**2.TABLE OF CONTENTS**

Chapter Page

1. Abstract 2
2. Table of Contents 3
3. Introduction 4
   1. Problem Background and How to Solve it 4
4. Main 4
   1. A Simple Look to MPLS and Comparation to SD-WAN 4
   2. Implementation of SD-WAN 6
5. The Action 6
   1. Step One 7
   2. Step Two 11
6. References 12
7. **INTRODUCTION**
   1. PROBLEM BACKGROUND AND HOW TO SOLVE IT

In today’s world, companies are getting larger and more complicated, so their needs in data trafficking. Configuring hundreds, thousands of devices is money and time consuming, and it is not just that. With the evolving technology, social media platforms, software as a services (SaaS) cloud technology, people realized they need to be as independent as they could be from hardware.

      In traditional networks, every router has a control plane that commands the data plane how to behave. This control plane is the software part of the device where we configure it via CLI and such. On the other hand, data plane is the hardware part where the data coming from other devices routing to one another by the current configuration. These two different entities are placed in the same place in a switch, acting as one entity.

      By that and many, people started to think about "Control plane being on each device is a problem." and began to create the SDN.

      The concept of the SDN is data plane and control plane are separated.

      The software part is pulled to a central point so that it can see and manage the whole network, evaluate it and decide on it.Now the control plane is in a central location and it can see the whole network, it has a command on the neighbor and routing tables of all switches on the network. Decision-making becomes smarter. On the other hand, SDN is abstracting the infrastructure of the network from the applications. In this way, SDN makes it possible for data planes to be programmable.

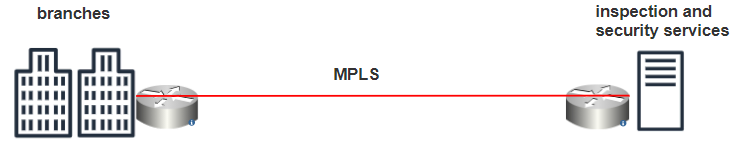
      With a centered, executive control plane, we are free from the dependency of the hardware and head to the software to change the whole network at once.

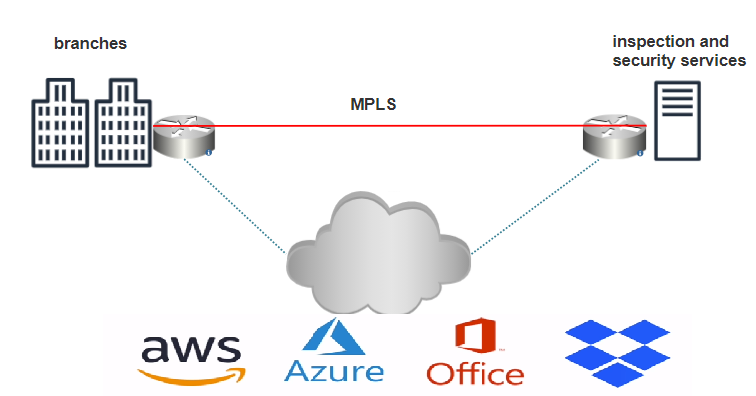
1. **MAIN**
   1. **A SIMPLE LOOK TO MPLS AND COMPARATION TO SD-WAN** [**[0]**](#_[0]_SD-WAN_and)

Traditional solutions on WAN connections allow us to interconnect main offices with branches and that allows us to share resources. We can share access to centrally located data services or applications. Traditionally dedicated circuits have been used to achieve this connectivity things such as a frame relay or MPLS, which uses tags that are assigned to data packets, and the packet transfer process is done only depending on the contents of these tags, regardless of the contents of the packet. Although these provide reliability and security for the connection our modern networks require some rethinking of this cloud usage as we know is on the rise so we need ways to simplify the management of wide-area networks, we also need a way to include our cloud resources.

Software-defined networks' goal is to control and manage the interactions between branch locations and our central resources. One another big advantages is that there's no longer need for backhauling your traffic.

If we look at a very simplified topology,



We can see that we have a branch location connected over an MPLS circuit which is a traditional WAN connection method that's connected back to a data center. At the data center, we have some advanced security and inspection happening so that's why we would tunnel all our traffic from the branch back to the data center for inspection. In this traditional LAN setup, all the traffic from the branch is backhauled to the data center for those security services that include traffic destined to the cloud or the public Internet and traffic destined locally within the organization as well. This can cause a myriad of performance issues, delay and depend on the circuit speed it can some bandwidth issues. On the other hand SD-WAN ,

can interact with all kinds of cloud applications. More and more our applications are going cloud based. For example, how cheap and easy AWS storages now and how office 365 has turned to a cloud platform for enterprise email access. SD wan can easily interact with applications like AWS Dropbox as your office 365 and many more. This means that hosting applications in either public or private clouds will allow direct traffic between the cloud application and the branch location. It will allow that rather than back hauling all the traffic through a central data center. SD WAN can intelligently control the path of traffic in order to optimize traffic flow and to reduce unnecessary bandwidth in our networks. This situation does not have any security issues, because SD-WAN provides end-to-end traffic encryption and inspection, anti-malware system, and botnet intervention.

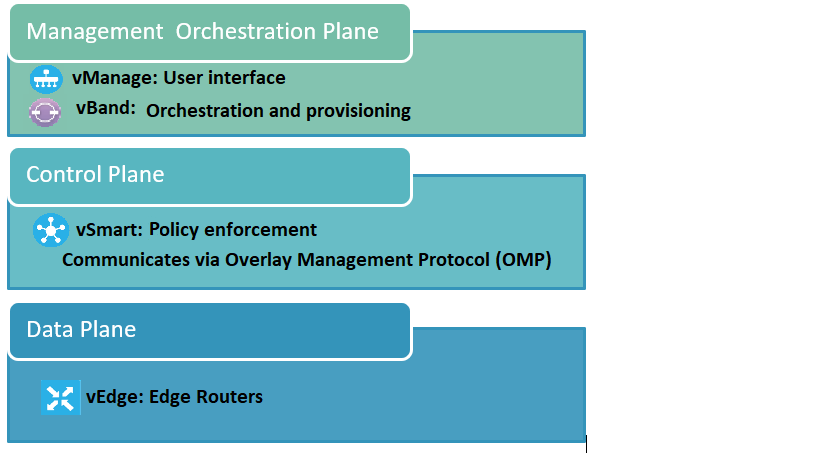
We can say that SDN took these abstract logical plane concepts into actual devices.

[**[1]**](#_[1]_(IEEE)_SD-WAN) In the data plane, there are edge routers called vEdge. **vEdge** is the software or hardware component that sits at your sites. vEdges form control plane connections with vSmart controllers, and not between each other.

vSmart is the brain of the system. vSmart stands in the control plane as a hub device and vEdges connecting with it. It controllers advertise routing, security and data plane policies. vSmart is the device that establishes the control plane component of the architecture.

For the management plane, vManage is the user interface of the system where we can control.

vBond orchestrate connectivity in the whole system. For example it tells our vEdge’s how and where to connect our organization.



* 1. **IMPLEMENTATION OF THE SD-WAN**

To be able to implement SD-WAN ideas, we need the help of some other elements.

[**[2]**](#_[2]_Ansible_Documentation)Ansible is an automation system required in the orchestration plane that provides management of in-house data centers, cloud structure, many servers in physical or virtual environments without the necessity for agent installation. For example, application distribution, software provisioning, configuration management. Ansible uses the playbook to execute commands and jinja2 templates. Those playbooks are files that we write all the tasks we want to automatize in YAML format. Jinja2 is a template developed for. We use it for our topology template.

1. **THE ACTION** [**[3]**](#_[3]__Cisco)

In this part of the report, we will apply the SD-WAN solution with Ansible using a sandbox and a VIRL supplied by Cisco.

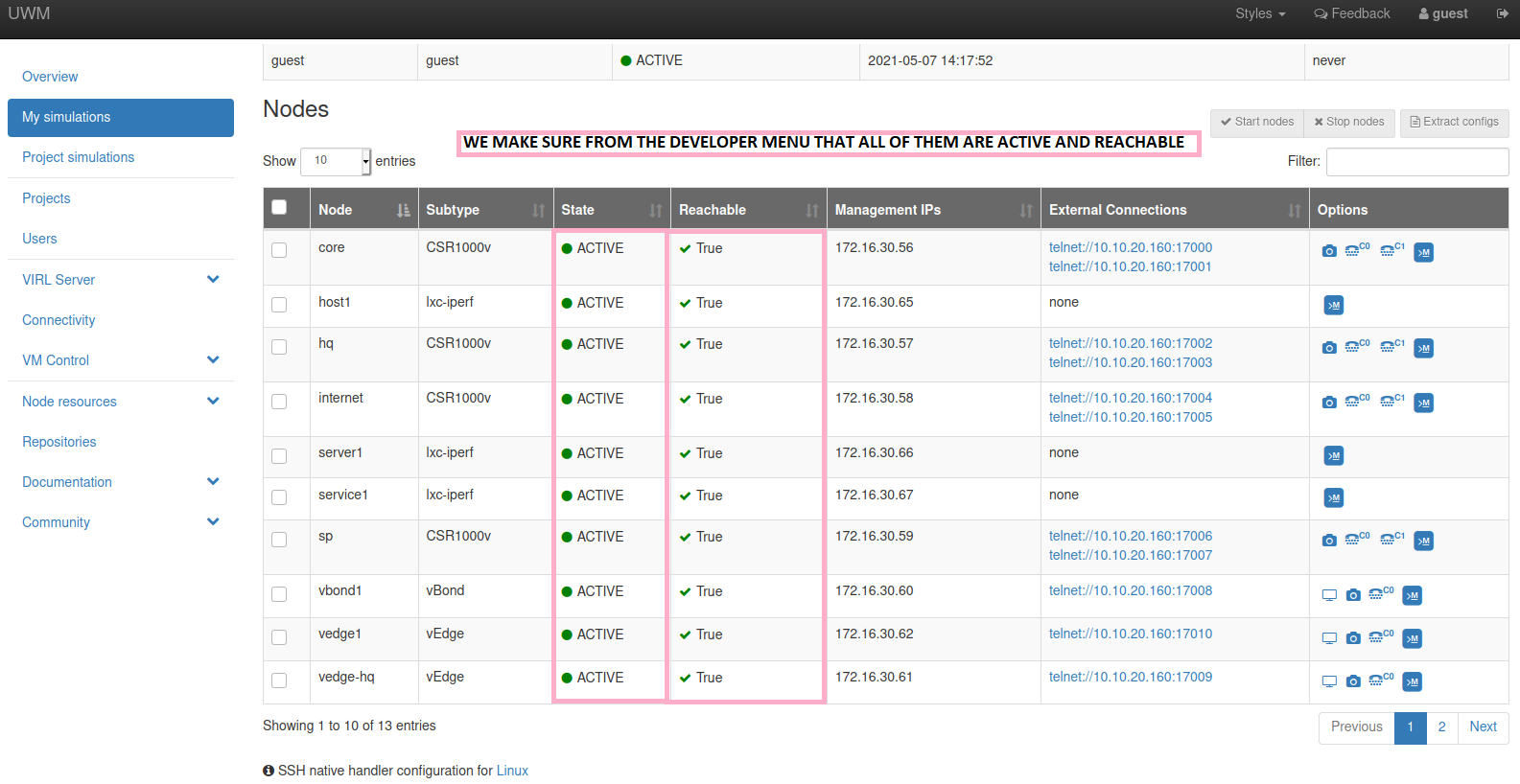
The objective of this act is starting a VIRL simulation and use Ansible to dynamically build a ViRL topology from inventory data.

The basic process for automating VIRL topologies with Ansible is to:

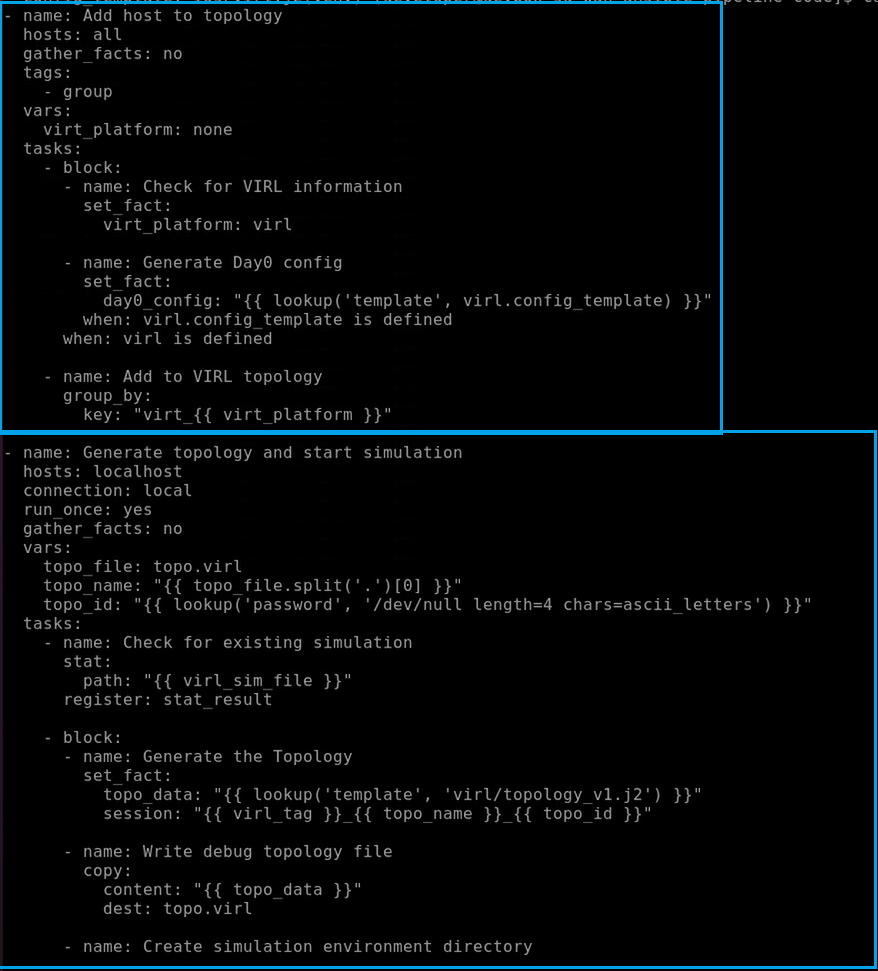
* Create the required inventory data for each node in the simulation (stored in **virl.yml**)
* Run the **build.yml** playbook
  1. **STEP ONE**

We connected to sandbox and VIRL via Cisco VPN so that we can access to SSH and developer mode

With the code below the started the Viptela SD-WAN simulation.

ansible-playbook build.yml

Now let’s look at the playbook, where the magic happens.



For every host in inventory (hosts: all), we check to see if it has any VIRL inventory data (when: virl is defined). If so, then we add it to the list of nodes to simulate in VIRL and generate its Day Zero configuration in the VIRL topology file using the Jinja2 template specified in the virl.config\_template variable.

Now that we have a list of hosts that need to be simulated in VIRL, the **Generate topology and start simulation** play will transform the VIRL data into a valid VIRL topology file and launch the simulation. In the **Generate topology and start simulation** play, view the **Generate the Topology** task. Where we generate the VIRL topology and store it as an Ansible fact (topo\_data) using a Jinja2 template (**topology\_v1.j2**).

Now let’s view the topology by this command,

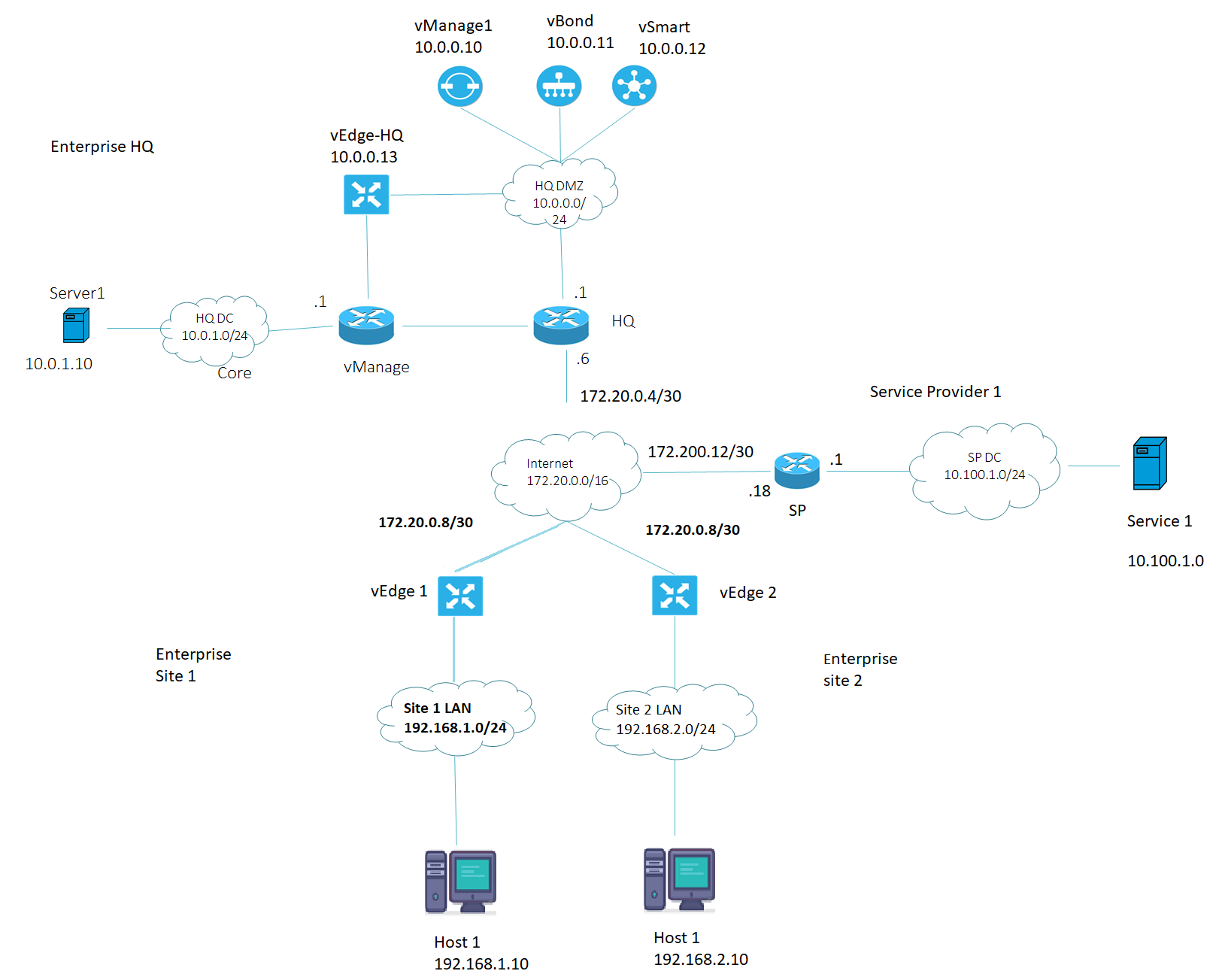
cat templates/virl/topology\_v1.j2

This template is what assembles all the VIRL inventory data into a XML format needed to launch a VIRL simulation. The template includes three main sections that are required for a valid VIRL topology:

* + **Network Nodes** - the nodes and their Day Zero configuration.
  + **Networks** - the networks.
  + **Connections** - the connections between nodes and networks.

Most of the work is done in the Network Nodes section were we:

* + Iterate over all hosts with virl data defined.
  + Set the VIRL node type (e.g. CSR1000v, vEdge, etc.).
  + Set the Day Zero config, if defined.
  + Create a list of required networks.
  + Create a list of required connections between nodes and networks.

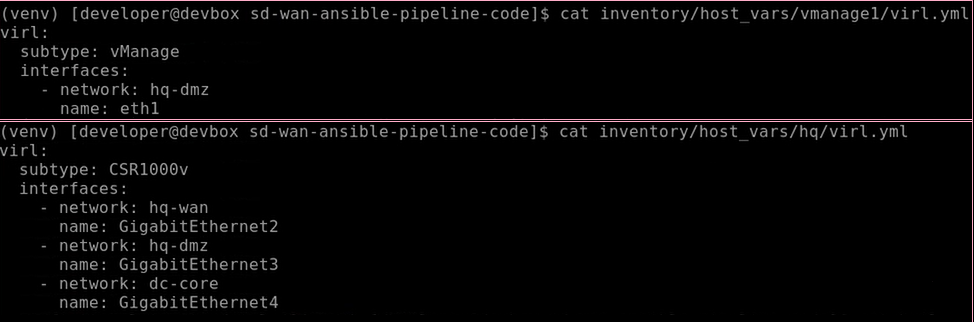
The final topology we created by build.yml ;

We created this topology by executing a few commands with the help of jinja2 template.

* 1. **STEP TWO**

The Jinja2 templates used by the build.yml playbook to dynamically generate a VIRL topology file based on Ansible inventory data are the 'magic' of the playbook. As a result, the Ansible inventory data becomes the foundation for our Infrastructure-as-code; if the inventory data changes, so does the resulting VIRL topology file. The relationship between Ansible inventory data and the VIRL topology is investigated in this step.

Each host in the Ansible inventory that we want to simulate in VIRL should have a **virl.yml** data file. In the photo down below, you can see the VIRL inventory data for vmanage1 and the HQ router.



We specified a vManage node with its eth1 interface connected to the network hq-dmz. Note that we specify several properties under the 'virl:' block*:*

* *VIRL subtype*
* *interfaces/networks this node will have*
* *Day Zero template to use, if any*

The HQ node is type CSR1000v and has interface GigabitEthernet3 connected to the hq-dmz network. This is how we make connections in the VIRL topology between nodes. If two or more nodes have interfaces in the same network, then they will be directly connected. In this case, vmanage1 will be directly connected to the HQ router.

By combining the subtype, the interfaces, and the config\_template for each node, we can create any arbitrary VIRL topology with Ansible inventory data. In the next step, we examine how this data is converted into a VIRL topology file using the **build.yml** playbook and Jinja2 templates.

1. **REFERENCES**

# **[0]** [SD‑WAN and SASE: The new landscape of networking (cisco.com)](https://www.cisco.com/c/m/en_us/solutions/enterprise-networks/sd-wan/new-landscape-of-networking.html?oid=ebketr024847)

[SD-WAN Solution - Cisco SD-WAN At-a-Glance - Cisco](https://www.cisco.com/c/en/us/solutions/collateral/enterprise-networks/sd-wan/nb-07-enterprise-grade-wp-cte-en.html#BenefitsofCiscoSDWAN)

[SD-WAN vs. MPLS: Why SD-WAN is a Better Choice in 2020 (fortinet.com)](https://www.fortinet.com/blog/business-and-technology/advantage-of-sdwan-over-mpls" \l ":~:text=MPLS%3A%20Why%20SD%2DWAN%20is%20a%20Better%20Choice%20in%202020,-By%20Nirav%20Shah&text=Compared%20to%20MPLS%2C%20SD%2DWAN,from%20vulnerabilities%20that%20MPLS%20cannot.)

# **[1]** P. Segeč, M. Moravčik, J. Uratmová, J. Papán and O. Yeremenko, "SD-WAN - architecture, functions and benefits," 2020 18th International Conference on Emerging eLearning Technologies and Applications (ICETA), 2020, pp. 593-599, doi: 10.1109/ICETA51985.2020.9379257.

# **[2]** [Ansible Documentation — Ansible Documentation](https://docs.ansible.com/ansible/latest/index.html?extIdCarryOver=true&sc_cid=701f2000001OH7EAAW)

# **[3]** [Cisco DevNet Learning Labs](https://developer.cisco.com/learning/tracks/sd-wan_programmability/sd-wan-ansible-pipeline/sdwan-automation-virl-topologies/step/1)