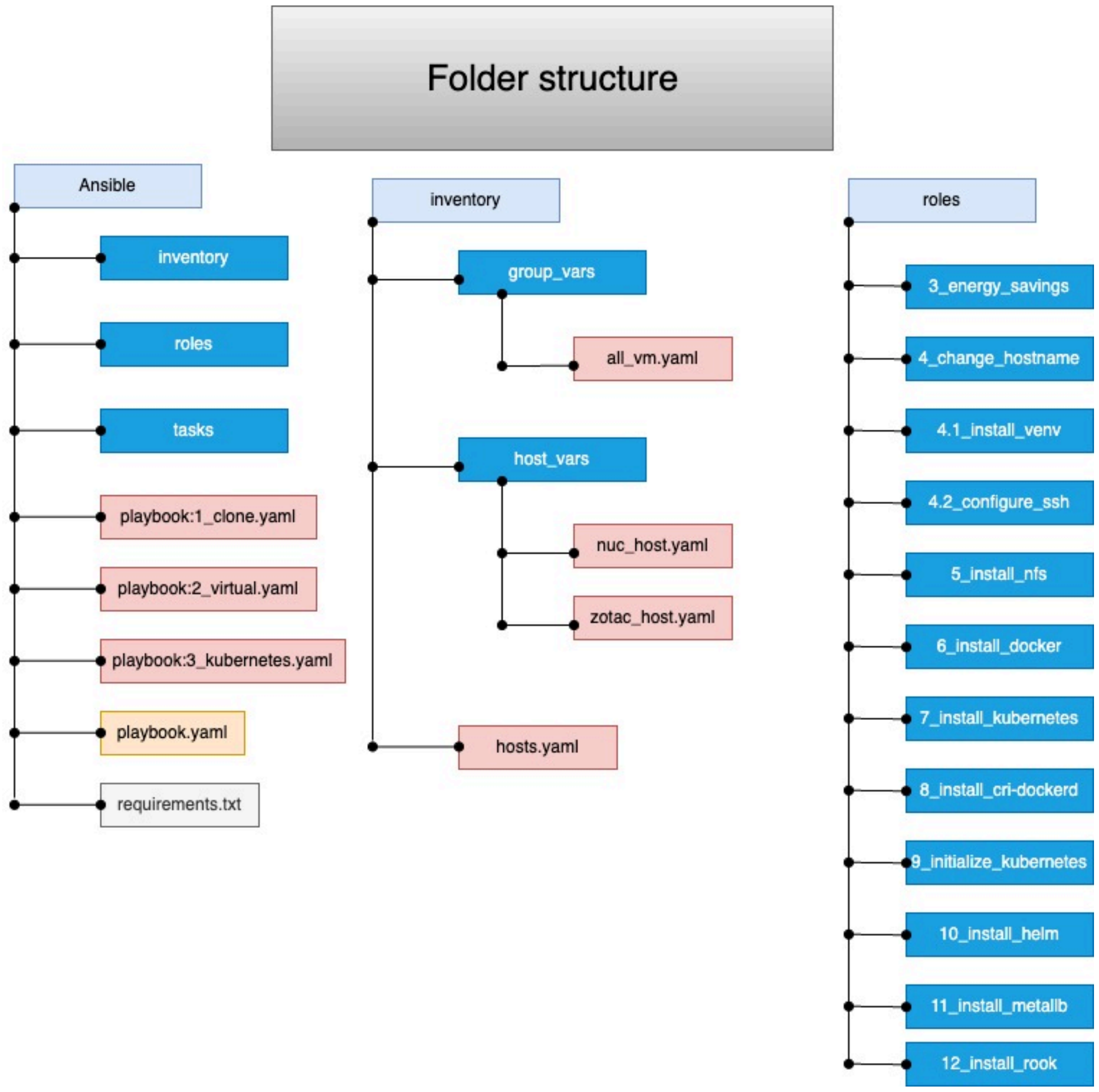
Technical documentation of Ansible script that deploys a Kubernetes cluster on Linux Ubuntu

Table of Contents

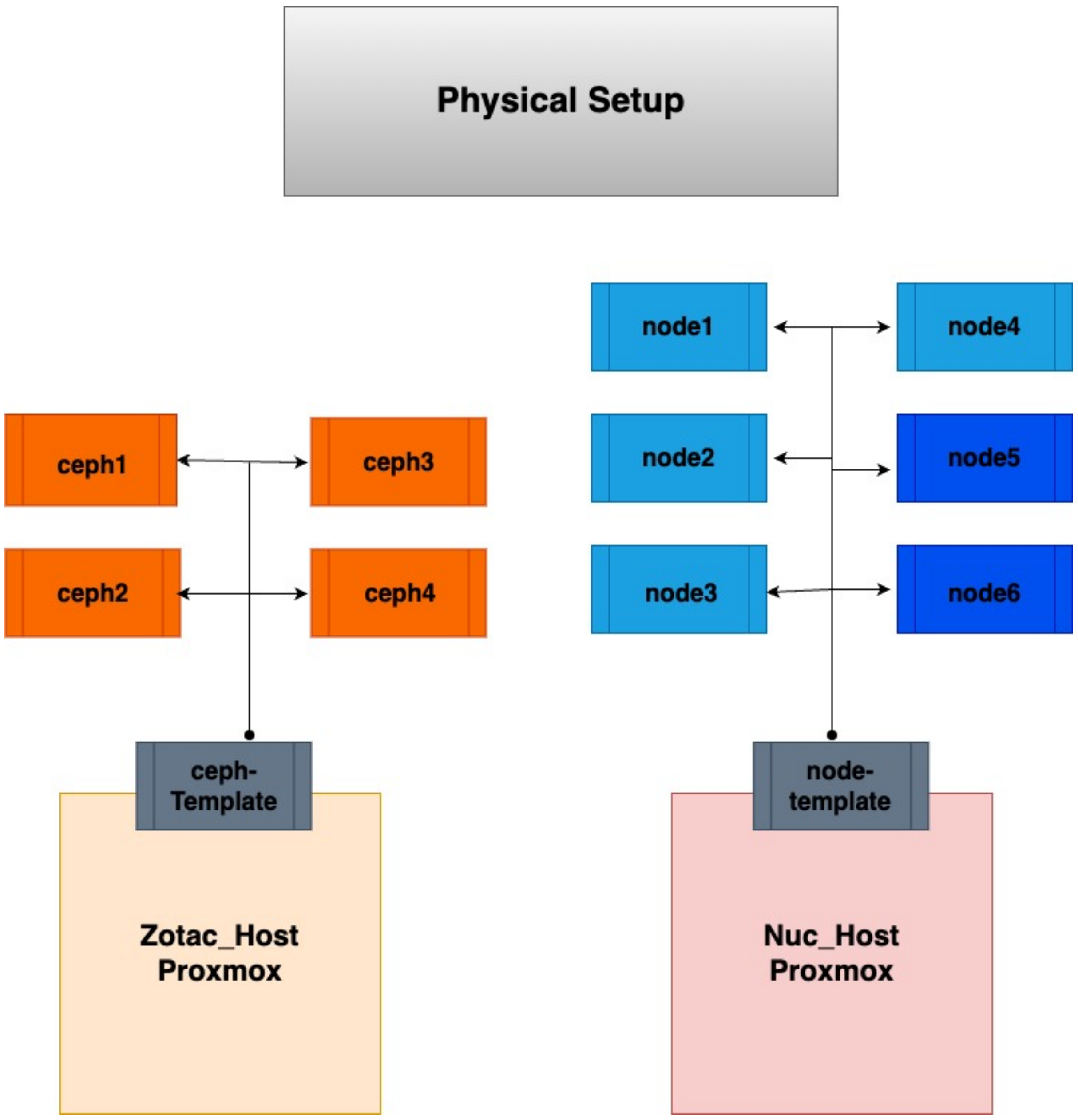

Folder structure
Ansible
inventory
roles
tasks
playbook:1_clone.yaml
playbook:2_virtual.yaml
playbook:3_kubernetes.yaml
playbook.yaml
requirements.txt
inventory
group_vars
all_vm.yaml
host_vars
nuc_host.yaml
zotac_host.yaml
hosts.yaml
roles
3_energy_savings
4_change_hostname
4.1_install_venv
4.2_configure_ssh
5_install_nfs
6_install_docker
7_install_kubernetes
8_install_cri-dockerd
9_initialize_kubernetes
10_install_helm
11_install_metallb
12_install_rook

Introduction

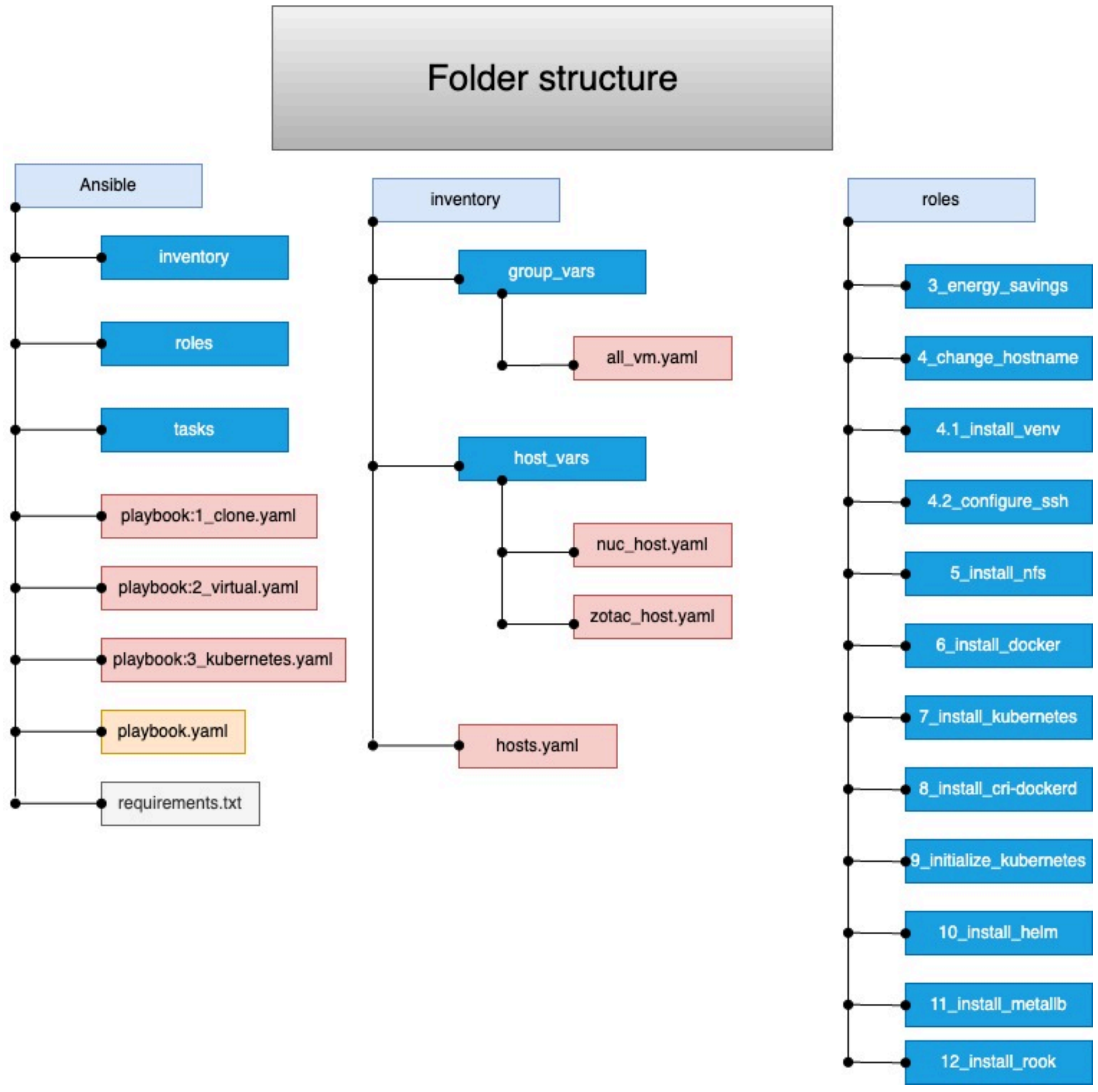
Brief overview of the environment, its purpose and tools used

This document describes the setup and configuration of a Kubernetes environment using Ansible. The environment is based on VMs cloned from templates on a Proxmox servers and includes virtual machine configuration, Kubernetes installation, and additional setup tasks.

The physical setup is illustrated below, and all hostnames are included in the current Ansible script. While these configurations may vary depending on specific setup, the idea is to provide a clear, bird overview of the code and its purpose



The folder structure shown below aligns with the Ansible scripts. Displaying it here serves as a reference point for better understanding when diving deeper into the code. There are more folders within, but the files that will be mentioned in this documentation will all be referenced by the folders displayed here.



Inventory Setup

The inventory folder contains a hosts.yaml file and 3 other files with the paths inventory/group_vars/all_vm.yaml, inventory/hosts_vars/nuc_host.yaml and inventory/hosts_vars/zotac_host.yaml. The name of the files correlates to the name of the groups in the inventory/hosts.yaml. Variables set in these files will also be set for the corresponding groups.

hosts.yaml

This file contains a hierarchical structure of groups, hosts and IP addresses of both VM's and the proxmox hosts. Some variables are defined here, these variables comes in second place in terms of priority. First priority is the variables defined in the playbook for the specific play. This will matter later. But it is important for overriding default values. Variables found here are used whitin the VM's and they are

- ansible_ssh_private_key_file: private key used for SSH
- ansible_python_interpreter: /home/administrator/venv/bin/python
- ansible_python_interpreter: /usr/bin/python3.12

This is the file where one would choose how many VM's the cluster should contain and which one that should be master

zotac_host.yaml & nuc_host.yaml

These files contain variables specific to each proxmox physical node(zotac and nuc), they contain configuration details required for interacting with and cloning VMs.

- *ansible_user:* The user account used for managing the proxmox host.
- *vm_id_template:* The template ID of the VM to be cloned.
- *ansible_python_interpreter:* /usr/bin/python3.11.
- *vm_list: Variables to make the VM unique.*

Instead of doing a live ubuntu installation, preconfigured, SSH ready VM templates are cloned. To make the VM unique there are variables in the vm_list, VM ID, MAC address and VM name. This makes the VM unique also in the network and with a preconfigures MAC address, the firewall can house preconfigured IP addresses for the new VMs.

all_vm.yaml

This file houses the variables

- ansible_user: The user for the VMs.
- NSF_IP: The IP address of a possible NFS share.
- NFS_PATH: The path to the shared folder of the NSF share.
- ceph_hosts: The hosts who will install the distributed ceph storage.
- ssh_hosts: The desired VMs who should have SSH connection to each other.

playbook.yaml

All other playbooks that are in the play are imported in playbook.yaml. This is the playbook that is run in the command line with the inventory folder as an argument. In this script there are 3 sub playbooks that are being run.

- playbook:1_clone.yaml, contains the task:
    - clones the VMs, specified in the *vm_list*
- playbook:2_virtual.yaml, contains the roles:
    - 3_energy_savings
    - 4_change_hostname
    - 4.1_install_venv
- Playbook:3_kubernetes.yaml, contains the roles:
    - 4.2_configure_ssh

    - 5_install_nfs
    - 6_install_docker
    - Snapshot task
    - 7_install_kubernetes
    - 8_install_cri-dockerd
    - 9_initialize_kubernetes
    - 10_install_helm
    - 11_install_metallb
    - 12_install_rook

The playbooks are divided into three plays (or more if we examine playbook 3 closely) to ensure tasks are well organize to the corresponding hosts with as few plays as possible. There are also other parameters than the IP address. There is one parameter called strategy which defines how Ansible will execute the commands. This one will differ in different plays depending on if it is possible for the specific installation.

- Serial execution
- Parallel execution

roles

Here is where instructions about what to install is being defined. In theory each role should survive on its own. Meaning that one could remove one role or use the 6_install_docker role on different VM for a different purpose. Installing it on another VM will work but using the role 9_initialize_kubernetes without the 7_install_kubernetes role will not work because of the prerequisites. If the prerequisites are kept in mind then the roles become modular and can be applied as one would need.

3_energy_savings

Disables suspend and hibernation on the VM

4_change_hostname

Changes the hostname with the hostname defined in the inventory/hosts.yaml it also updates the /etc/hosts files with the new information and will write the new IP in there aswell.

In this role we have decided to export the KUBECONFIG in .bashrc because of Kubernetes not doing this in the installation process. By doing this it makes it so Kubernetes and all surrounding programmes knows where to reference the file /etc/Kubernetes/admin.conf and there won't be any unsolvable issues no more.

4.1_install_venv

To use the kubernetes.core.k8s module in Ansible, both the Ansible controller and the remote machine need to have common ansible packages installed. These packages are recommended to be installed in a virtual environment, which is why this role sets up the virtual environment and installs a file called requirements containing 32 specified packages.

Since it is not possible to autoscale the deployment beyond the VMs already prepared (unlike deployments on online services), it is possible to make the master node a act as a second ansible controller. the infrastructure in terms of virtual environment is already in place. However, two things are missing to make this possible.

First an SSH connection must be established between the host that creates the VMs and the second Ansible controller. Second, an IF statement is needed to check if the number of VMs running is equal to the number of pods deployed. If so, the IF statement will either create another set of VMs or the opposite, shut down a VM. The code for creating the new VMs exists but would need some tweaking to achieve this functionality.

4.2_configure_ssh

This role configures SSH connection between desired VMs in the cluster. It is not currently requested in the order but we at Tedros IT & Associate believe whitout this part the script would be incomplete and hindered for future growth. The script does following tasks:

- Add public key to authorized_keys: This allows passwordless SSH access
- Update the SSH config file: It sets what username and the path to the RSA key to user for a pre specified IP
- Add IPs to known_hosts: This is made to create a trusted connection between the listed IPs. It adds all hosts in the list exept its own IP.

This role dynamically creates an SSH connection between listed hosts. The list is defined in inventory/group_vars/all_vm.yaml under ssh_hosts.

5_install_nfs

Although this part isn't specifically requested, it is common for Kubernetes deployments to have a storage that survives a restart. To be as proactive as possible and ensure quality work, we include a method for connecting an NFS share to potentially future NFS dependent clusters. This role does the following tasks:

- Installs nfs-common
- Creates a directory for the mount path
- Uses ansible.builtin.mount to mount the NFS share

The path for the NFS, the mount point and the IP are variables that can be changed in inventory/group_vars/all_vm.yaml.

6_install_docker

This part is a direct copy from Docker.com Ubuntu documentation. Translated to a Ansible install. The following steps are involved:

- Remove all possible conflicting packages
- Add Dockers official GPG key
- Set up the Docker repository
- Install Docker
- Start and enable Docker service
- Create a directory in the home folder

Docker is used because Kubernetes needs a socket to use and we have found that Docker is the most reliable way of connecting to Kubernetes.

Snapshot task

This snapshot is taken because the tasks ahead of this step might fail. The sole purpose of this snapshot is to speed up the retry process by avoiding the need to repeat the previous steps

7_install_kubernetes

This role installs Kubernetes on all VMs. The procedure can be seen in the Ansible code. The installation follows the documentation for installing Kubernetes. This installation have followed the kubeadm route since it is going to be used on multiple VM.

8_install_cri-dockerd

CRI-Dockerd is used as a way of connecting Kubernetes to Dockers socket. Kubernetes uses CRI-Dockerd as a middleman. There are 3 files that are being copied over to the VMs.

1. cri-dockerd.deb
2. cri-docker.service
3. cri-docker.socket

The first is a .deb package file, and once it is copied, the installation of CRI-Dockerd begins. The second is the service file, which defines how CRI-Dockerd is managed by systemd. Last is the socket which defines the communication interface for CRI-Dockerd.

9_initialize_kubernetes

This role initializes Kubernetes. creating all the necessary files and folders for the cluster. Before this can be done, swap must be disabled on all VMs. The steps taken are:

- Disable swap.
- Initialize the cluster and save the output to a variable.

- Print out the result in the Ansible controller terminal.
- Save the output of the initialization to a file on master node.
- Fetch the file to Ansible controller.
- Perform a regex search and save the result to a variable for (Join_command).
- Send the join command to all the worker nodes.
- Change the permissions of the kubeconfig file .
- Copy the kubeconfig file to Ansible controller (To use the kubernetes.core.k8s module later).
- Apply the Calico network plugin, Defines the network the pods will use. Calico uses BGP.

10_install_helm

- Download Helm apt repository key
- Install prerequisite packages
- Add the Helm repository
- Install Helm
- Create a directory in the users Home folder

11_install_metallb

Metallb provides the Kubernetes cluster with a load balancer that assigns an external IP. External in this context refers to a network address that can be accessed from outside the cluster.

This role also has another focus, it installs a metric server in the cluster which is used to monitor the cluster and pods. This server enables Kubernetes to scale up or down. The components.yaml file have been edited to accept self signed certificates for the pods/Kubernetes network.

12_install_rook

Rook is the second storage alternative we have decided to provide as an option. The storage is distributed and the variable *ceph_hosts* can be found in

- inventory/group_vars/all_vm.yaml

this is where one would define which hosts and hard drives should act as a storage. Rook requires the hard drive to be raw/unformatted.