Ceph Cluster Lab with Ansible

Goal:

Follow along with a recent Network Chuck video on Ceph Clustering but use Ansible as much as possible to implement the lab. Follow along with his setup instructions but use ansible where possible to provision and configure machines.

Definitions

Ceph: (Per https://ceph.io/en/) "Ceph is an open-source, distributed storage system"
Ansible: (Per https://www.ansible.com/) "Ansible is an open source IT automation engine that automates provisioning, configuration management, application deployment, orchestration, and many other IT processes. It is free to use, and the project benefits from the experience and intelligence of its thousands of contributors."

Planned Steps for lab:

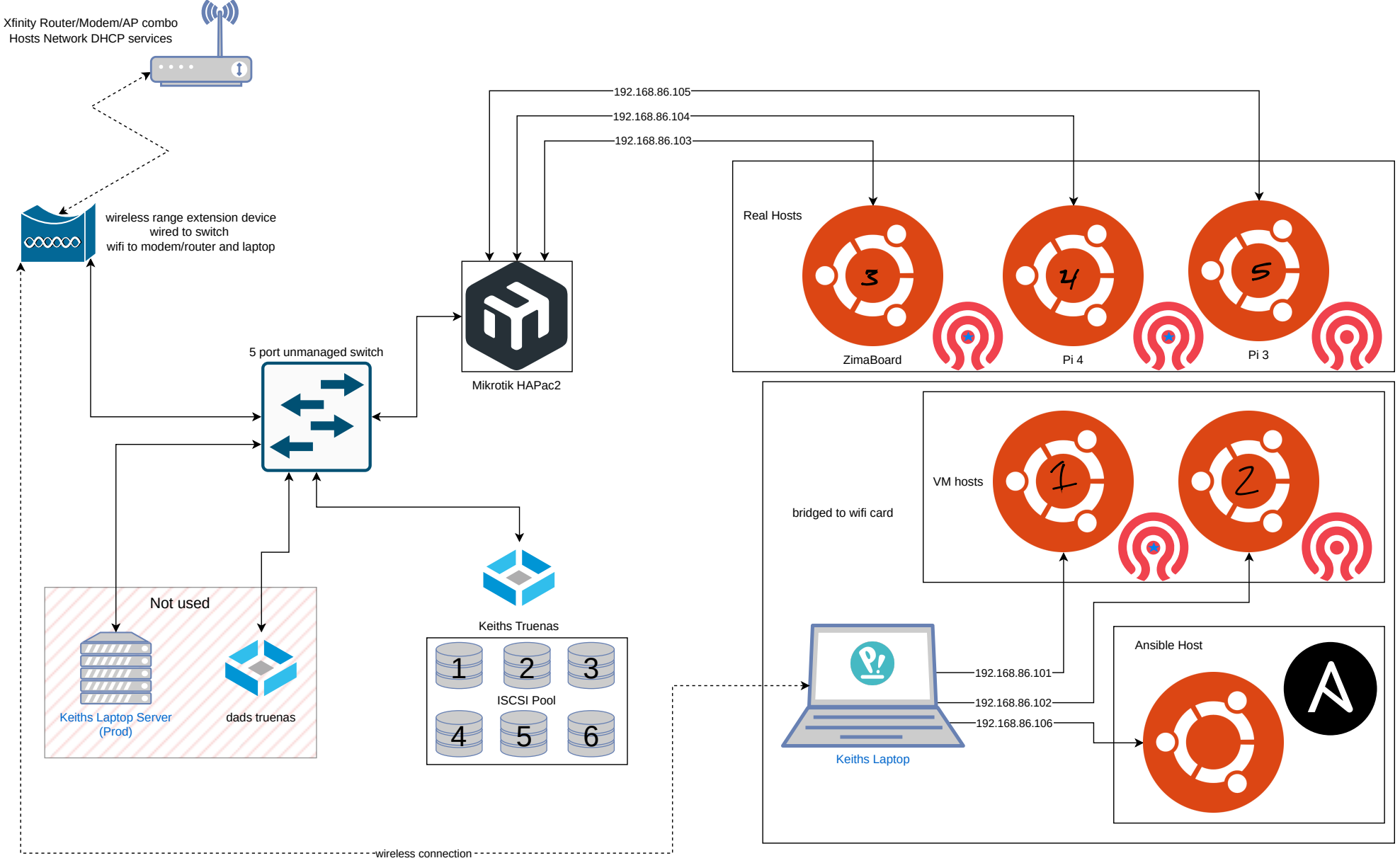
1. Provision Virtual Machines
    1. follow the included spreadsheet
    2. install operating system to each machine per diagram, run updates
    3. setup default user on each machine and import GitHub ssh keys for that user, the keithm username with a simple password to get started and will not be online until ready to deploy new credentials via ansible. Use a secure pasword for the machine that will be running ansible playbooks against other hosts/
    4. setup host-names and IP addresses for each host.
    5. setup ssh connection between the machine that will be the ansible host (ubuntu-server6) and all the CEPH workers/ansible targets (this will require you turn on ssh password connection briefly. turn this back off when done)
    6. once all the basic provisioning is complete, turn off the 3 physical machines, and take a snapshot of the VM's, these will be used while testing the ansible implementation to prevent unintended data loss

server NUM	Host OS	Host Hardware	Hostname	Ethernet Adapter	IP address	added storage	roles
server 1:	**ubuntu 22.04**	VM, 2GB RAM, 2 Core	ubuntu-server1	enp0s3	**192.168.86.101**	iscsi1, iscsi2	ansible target, testing: ceph manager final: ceph worker
server 2	**ubuntu 22.04**	VM, 2 GB RAM, 2 Core	ubuntu-server2	enp0s3	**192.168.86.102**	iscsi3	ansible target, ceph worker
server 3	**ubuntu 22.04**	Zimaboard 832	ubuntu-server3	enp0s2 enp0s3(dhcp)	**192.168.86.103**	iscsi4, 500gb ssd, 500gb hdd	ansible target, ceph manager/worker
server 4	**ubuntu 20.04**	Raspberry Pi4	ubuntu-server4	eth0	**192.168.86.104**	iscsi5	ansible target, ceph worker
server 5	**ubuntu 20.04 pi3**	Raspberry Pi 3	ubuntu-server5	eth0	**192.168.86.105**	iscsi6	ansible target, ceph worker

server NUM	Host OS	Host Hardware	Hostname	Ethernet Adapter	IP address	added storage	roles
server 6	ubuntu 24.04	VM, 4 GB RAM, 4 Core	ubuntu-server6		**192.168.86.106**	(map in network share for ansible playbooks added to host)	ansible controller

2. provision ISCSI on NAS (TrueNAS-scale) per the below list (I don't have any external drives to use for this project) ISCSI drives are sufficient to test with and adds the challenge of setting up the drives on each host.
    1. iscsi1 - 200gb VM1
    2. iscsi2 - 100 VM1
    3. iscsi3 - 100 pi4
    4. iscsi4 - 100 Pi3
    5. iscsi5 - 120 VM2
    6. iscsi6 - 85 VM2
3. setup HAPac2 Mikrotik router to get physical machines on the network, This is also going to be a useful tool to diagnose network issues as they occur. refer to lab network diagram for network layout. In my case the network is not ideal for lab work but its what I've got
4. learn how to complete the following steps with ansible
    1. run updates and upgrades
    2. setup a new user on all hosts with unique, strong passwords. delete bad password user.
    3. setup ssh as root (modify /etc/sshd.config) add root ssh key from ubuntu-server3 to all hosts (per network chuck ceph video though this is insecure as far as I know)
    4. mount iscsi drives on all systems
    5. prep drives (not the drive with the host os)
        - wipe out the disk (sgdisk --zap-all /dev/sd_)
        - wipe the file system (wipefs)
    6. install dependent software on each system
        - docker
        - lvm2 (should already be installed)
        - open-iscsi (only needed in the case your using iscsi drives in the cluster)
        - timedatectl status needs to be accurate (NTP synchronized and UTC time on all servers)
    7. prep storage for ceph usage on all hosts
    8. set var for ceph_release
    9. install cephadm and make executeable after pulling down ceph repo to root user on
    10. cephadm bootstrap --mon-ip (ip address of manager)
    11. setup mon's (these are monitors or managers of the cluster. you need a few of these for it to have proper redundancy)
    12. add osd's (these are our drives essentially as I understand it)
5. once setup, play around with ceph, look at metrics, setup cephfs

Diagram of network:



Results:

Overall I achieved my goals for this lab.

I think I got a better idea on using ansible to provision machines with software and setup basics stuff. I also learned about stuff that I will need to learn to get better at all of this.
those tools include:

- Key Management using Hashi-corp Vault, 1password, Bitwarden or Vaultwarden
- Cloud Init
- Teraform (IAC)
- Using ceph and understanding how it is used out in the wild.
- Virt-Manager for Linux based virtual machine management, specifically on remote hosts.
- understanding how ansible variables work and how they're pulled into a playbook. I mostly understood this but there's a lot more there to learn. Looks like there is a vs-code extension that could help with this called Ansible Variable Lookup though I haven't used this yet.

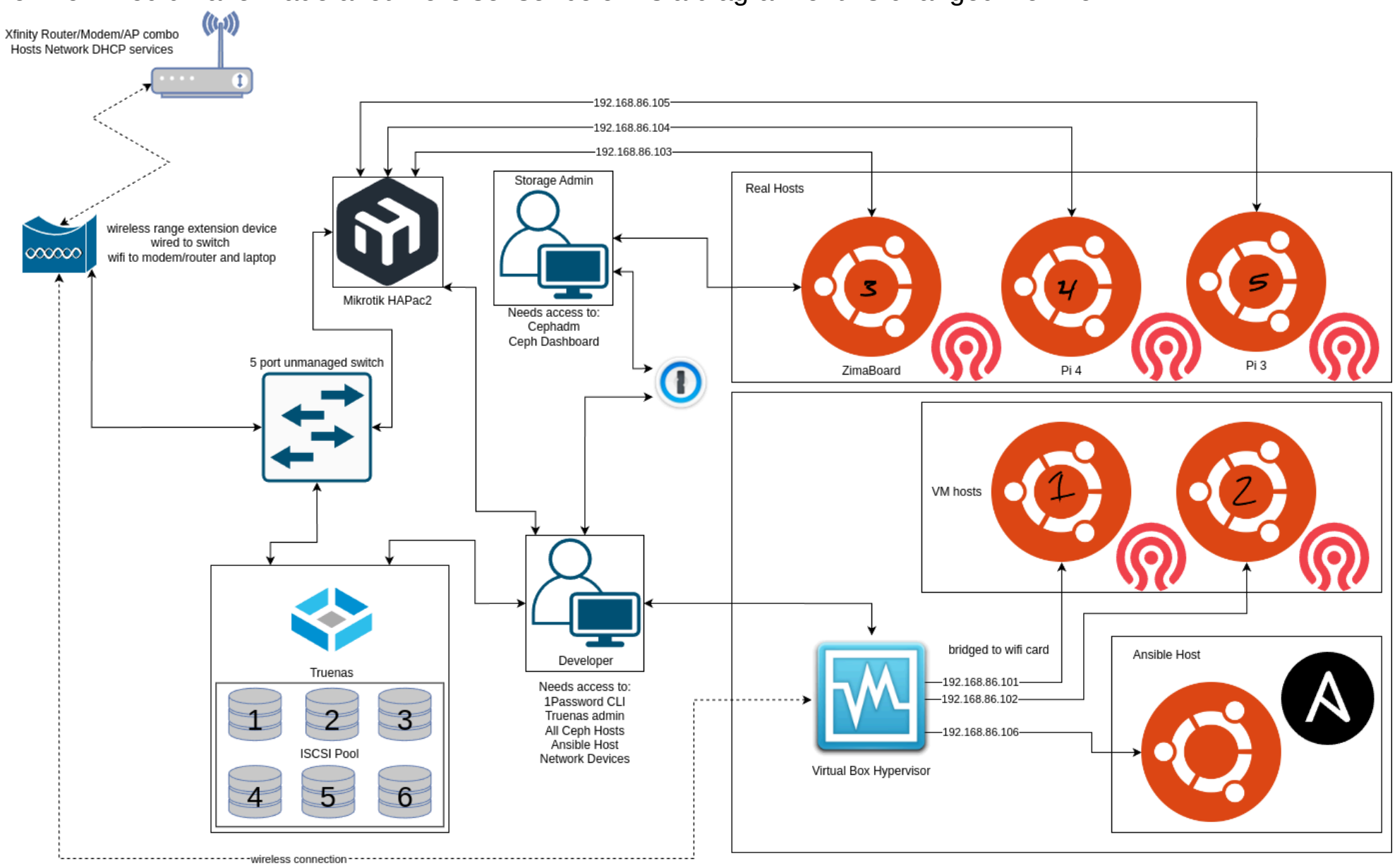
A list of stuff I learned while completing this lab

- Setup TrueNAS served iscsi drives
- Using 1password CLI in tandem with bash scripting
- When a guide or documentation recommends opening up a root terminal before making a change there is probably a good reason for it
- PAM module and Yubi-key setup for a second factor on Linux
- Ansible
  - setup inventory files with hosts, host specific vars, and setting up nested group inventories
  - using ansible vault files for playbook vars, calling specific vars in a loop, and other tasks for setting up users with secure credentials
  - Ansible-Playbooks to:
    - run updates
    - install software
    - install docker

    - install software that isn't already in apts default repos
    - set user time to all use UTC and ensure NTP time is enabled
    - setup users with secure passwords that are stored at rest in encrypted vaults.
    - create ssh keys and import them to all hosts so the ansible machine will always have access to those machines
    - remove users that are on the systems
    - pull in iscsi drives and mount them in the file system (this step the mount part specifically may have been what messed up the raspberry pi's)
    - format specific drives as specified by the host in the inventory file this could probably have been done better but it was towards the end of the project and I was just wanting to blast though.
    - I messed around a bit with crowdsec, I think I have a good idea on what it does but it was distracting me from finishing the project so I put it to the side. Probably worth it to circle back on this.
    - Install cephadm thought I think next time I will change the work flow to actually using cephadm from the machine that it is installed on That would have saved me a ton of time though the real answer is figure out ceph-ansible.
- Setup ansible.cfg
    - learn how to work with ansible-lint even though it sometimes doesn't make much sense.
- bash scripting
- gpg generally and as a part of a script
- documentation and pre-planing/diagramming a network

I looked into the ceph-ansible tool and I struggled for 3 days to understand how to use it. at that point I chose to simply follow along with the Network Chuck video when it came to the steps included in actualy installing ceph on all the prepared hosts. I think that I need to re-approach that particular task with fresh eyes in a month or two. The final results of all this setup was a working ceph cluster though with a few detraction's. for some reason, even though I was able to confirm that the drives were wiped and had nothing on them and they should have been visible to the cluster, the two raspberry pi devices did not want to play nice. They were able to join the cluster but they were not able to share their drives with the cluster. I couldn't figure this out and Im going to chock it up to while it may be possible to use raspberry pi's with ceph it is likely very unstable and probably shouldn't be attempted in anything resembling production. An additional problem was the pi 3 on average took about 2-3 times as long to run any given command.
Once setup cephadm was fairly straight forward to use. I was able to use the cephadm command line tool to bring in other hosts and I was also able to do that task with the dashboard that was setup using cephadm. As I said in the things I learned I think that I should have thought of the flow of this lab as though I had two admins. one doing the ansible side of things and one responsible for administering ceph. this would have the ansible host setup all hosts and got them prepped to be managed by the ceph administrator. This way the ceph admin would get keys to login to the ceph machine where they could have done the administration from. These two people could defiantly be the same person filling two different roles but if I had thought of things like this the

workflow would have made a lot more sense. below is a diagram of this changed workflow.



Things to do next time:

- setup remote hosts targeted by anisble via tail-scale
- scripts for setup should be applicable to different networks and built in a way that makes them scale-able
- use a key-management utility
- update and configure Mikrotik router using ansible
- use more hosts
- do not use raspberry pi's for this

Notes: Ceph Cluster Lab Notes!
Original Write-up: Ceph Cluster Lab with anisble