Ansible, tailscale, proxmox automated templates

Stuff for the ansible playbook

Tailscale optimizations generally, mostly for subnet routers

https://tailscale.com/kb/1320/performance-best-practices

Tailscale subnet router setup

https://tailscale.com/kb/1019/subnets

The vault look-ups

this project helped me get a better idea on how ansible looks up variables. this diagram is how I understand this as working.

![[Drawing 2024-12-11 17.47.07.excalidraw.png]]

basicly the way I understand this working is the vars file is used to map in variables from diffrent places including the inventory file and also the vault file. kinda brings everything together.

This was the main addition that this playbook added to my knowledge of ansible. Learning to work with variables and pull information from different sources will be incredibly handy going forward. As far as I can tell this method of pulling variables works but could be turned into a oneliner and hopping over the vars file? It is possible this is wrong as Im still learning about how this all works.

This is the rest of the ansible script written for this project. Beyond the use of vars files the main things I think are cool about this playbook is the use of registering facts about the packages installed using the ansible.builtin.package_facts module to pull information about the systems targeted. I also register the outputs of certain plays to check if other plays should run. this helps when running this play against systems after they have been configured once.

```YAML
---

- name: preliminary steps

  hosts: new_guests

  become: true

  vars_files:

    - ./vars.yaml

    - ./vault.yaml

  tasks:
```

```
- name: Set a hostname

  ansible.builtin.hostname:

    name: "{{ inventory_hostname }}"


- name: gather facts about packages installed

  ansible.builtin.package_facts:

    manager: auto

- name: install packages

  ansible.builtin.apt:

    upgrade: full

- name: Check if reboot is required

  ansible.builtin.stat:
```

```
    path: /var/run/reboot-required
    register: reboot_required_file
- name: Reboot systems to apply kernel updates
  ansible.builtin.reboot:
  when: reboot_required_file.stat.exists == true
```

```
- name: setup crowdsec
  hosts: new_guests
  become: true
  vars_files:
    - ./vars.yaml
```

```
    - ./vault.yaml




  tasks:




    - name: Crowdsec repo's are installed via script

      ansible.builtin.shell: curl -s https://install.crowdsec.net | sh

      register: my_output # <- Registers the command output.

      changed_when: my_output.rc != 0 # <- Uses the return code to define when the task has changed.

      when: "'crowdsec' is not in ansible_facts.packages"

    -

    -

    - name: Crowdsec install

      ansible.builtin.apt:
```

```
    name: Restart crowdsec post enrollment to get stuff working
    ansible.builtin.service:
      name: crowdsec
      state: restarted
    when: "'crowdsec' is not in ansible_facts.packages"


- name: install and enroll tailscale on hosts
  hosts: new_guests
  become: true
  vars_files:
    - ./vars.yaml
```

```
    - ./vault.yaml




  tasks:


  - name: install and enroll host in tailscale

    

    ansible.builtin.shell: curl -fsSL https://tailscale.com/install.sh | sh && sudo tailscale up --auth-key="{{ tailscale_auth_key }}"

    

    when: "'tailscale' is not in ansible_facts.packages"

    

    

    

  - name: setup subnet routers pt 1

    

    ansible.builtin.shell: echo 'net.ipv4.ip_forward = 1' | sudo tee -a /etc/sysctl.d/99-tailscale.conf

    

    # when: "'tailscale' is not in ansible_facts.packages"

    

  - name: setup subnet routers pt 2

    
```

ansible.builtin.shell: echo 'net.ipv6.conf.all.forwarding = 1' | sudo tee -a /etc/sysctl.d/99-tailscale.conf

when: "'tailscale' is not in ansible_facts.packages"

- name: setup subnet routers pt 3

ansible.builtin.shell: sudo sysctl -p /etc/sysctl.d/99-tailscale.conf

when: "'tailscale' is not in ansible_facts.packages"

- name: subnet router optimizations

ansible.builtin.shell: printf '#!/bin/sh\n\nethtool -K %s rx-udp-gro-forwarding on rx-gro-list off \n' "$(ip -o route get 1.1.1.1 | cut -f 5 -d " ")" | sudo tee /etc/networkd-dispatcher/routable.d/50-tailscale && sudo chmod 755 /etc/networkd-dispatcher/routable.d/50-tailscale

- name: advertise routes for systems

ansible.builtin.shell: tailscale up --advertise-routes "{{ subnets }}"

```
```

This is the vars file included with this project. the commented out lines are from earlier iterations of this project that are helpful to understand how things are looked up using variables

```YAML
# ansible_password: "{{ server_passwords.admin }}"

# ansible_become_password: "{{ server_passwords.admin }}"

tailscale_auth_key: "{{ individual_sys[server_id].key}}"

# new_borg_password: "{{ individual_sys[server_id].borg_pass }}"
```

This is the vault file that I created and is useful for understanding how auth-keys are looked up while stored in this secure format. It is sanitized and the keys used are only useful once anyway

```YAML
individual_sys:

  ts-node1:
```

key: tskey-auth-kfjkdeadfkeeCNTRL-ueoiruyaoieuryaiosudfyoaisudyfudh

ts-node2:

key: tskey-auth-kfjkdeadfkeeCNTRL-ueoiruyaoieuryaiosudfyoaisudyfudh

ts-node3:

key: tskey-auth-kfjkdeadfkeeCNTRL-ueoiruyaoieuryaiosudfyoaisudyfudh

```

---

This is the inventory used for this project, Useful for understanding vars and the workflow with this script

```YAML

### uspace stuff

## Once configured, hosts from new_guests get moved to guests

```
you should be able to switch hosts from ip addresses to mdns names

Systems configured like this could also be swapped over to mainline inventorys

##

new_guests:

  hosts:

    tailscale-node3:

    # ansible_host: tailscale-node1.local

    ansible_host: 10.0.0.13

    server_id: ts-node6

    subnets: 192.168.4.0/24

  vars:
```

```
    #ansible_user: administrator

    ansible_user: borg




guests:

  hosts:

    tailscale-node1:

    ansible_host: tailscale-node1.local

   # ansible_host: 10.0.0.11

    server_id: ts-node1

    subnets: 192.168.0.0/24



    tailscale-node2:
```

```
    ansible_host: tailscale-node2.local

  # ansible_host: 10.0.0.12

    server_id: ts-node2

    subnets: 192.168.0.0/24



  vars:

    ansible_user: borg
```

I did do a bit of work for a previous version of this project before i figured out custom cloud init files. below are those files. at this time I was creating a borg user, setting a borg user password, and targeting an admin account that required a password for running sudo commands. I was also passing in the whole enrollment command instead of just an auth-key which is much more cumbersome. ultimately this ended up being unnecessary.

vars:

```yaml
```

ansible_password: "{{ server_passwords.admin }}"

ansible_become_password: "{{ server_passwords.admin }}"

tailscale_auth_command: "{{ individual_sys[server_id].command}}"

new_borg_password: "{{ individual_sys[server_id].borg_pass }}"

```

vault:

```YAML

server_passwords:

admin: password123456!@#$

individual_sys:

tsn1:

command: curl -fsSL https://tailscale.com/install.sh | sh && sudo tailscale up --auth-key=tskey-auth-k8iqCx4jQV11CNTRL-xW3gNm6FuKf7ra49zQ32SfTKRJnVCGbC

borg_pass: ez8aaBkNxRyKGeR9zhdxLRQGoJfoMEEiRuptxamr

tsn2:

command: curl -fsSL https://tailscale.com/install.sh | sh && sudo tailscale up --auth-key=tskey-auth-kBxMEyjHu111CNTRL-3JzrF8UmkRacSdd4yHbuKav7udHYBkeua

borg_pass: aoQCCYphdFvBhzxxnVQW4C6XfrtDtePKUeAhUxj9

```

here's the playbook I wrote for this its basically the same as one I ultimately used with a few slight tweaks. It also includes the steps to bootstrap a borg user.

```YAML

---

- name: setup apt-cache proxy

hosts: new_guests

become: true

vars_files:

- ./vars.yaml

- ./vault.yaml

tasks:

- name: remove apt-cache proxy info

ansible.builtin.file:

path: /etc/apt/apt.conf.d/00proxy

state: absent

register: proxy

- name: reboot all hosts to apply changes

```
ansible.builtin.reboot:

when: proxy.changed

- name: setup mdns and install general use packages

hosts: new_guests

become: true

vars_files:

- ./vars.yaml

- ./vault.yaml

tasks:

- name: gather facts about packages installed
```

```
ansible.builtin.package_facts:

manager: auto

- name: install packages

ansible.builtin.apt:

pkg:

- libnss-mdns

- qemu-guest-agent

update_cache: true

when: "'libnss-mdns' is not in ansible_facts.packages"




- name: apply hardening

hosts: new_guests
```

become: true

vars_files:

- ./vars.yaml

- ./vault.yaml

tasks:

- name: Crowdsec repo's are installed via script

ansible.builtin.shell: curl -s https://install.crowdsec.net | sh

register: my_output # <- Registers the command output.

changed_when: my_output.rc != 0 # <- Uses the return code to define when the task has changed.

when: "'crowdsec' is not in ansible_facts.packages"

```
- name: Crowdsec install

ansible.builtin.apt:

package:

- crowdsec

update_cache: true

when: "'crowdsec' is not in ansible_facts.packages"



- name: Crowdsec install

ansible.builtin.apt:

package:

- crowdsec-firewall-bouncer-iptables
```

update_cache: true

when: "'crowdsec' is not in ansible_facts.packages"

- name: Enroll in crowdsec console

ansible.builtin.command: sudo cscli console enroll -n {{ inventory_hostname }} -e context clz8lrn840007lb085o6va59z

register: my_output # <- Registers the command output.

changed_when: my_output.rc != 0 # <- Uses the return code to define when the task has changed.

when: "'crowdsec' is not in ansible_facts.packages"

- name: Add linux collection

ansible.builtin.command: sudo cscli collections install crowdsecurity/linux

register: my_output

changed_when: my_output.rc != 0

when: "'crowdsec' is not in ansible_facts.packages"

- name: Restart crowdsec post enrollment to get stuff working

ansible.builtin.service:

name: crowdsec

state: restarted

when: "'crowdsec' is not in ansible_facts.packages"

- name: Create borg user

hosts: new_guests

become: true

vars_files:

- ./vars.yaml

- ./vault.yaml

tasks:

- name: Check if users exist

ansible.builtin.user:

name: borg

check_mode: true

register: test_users

```
- name: Set user password from vault based on host index for new user borg

ansible.builtin.user:

name: borg

shell: /bin/bash

groups: sudo

password: "{{ new_borg_password }}"

append: true

create_home: yes

no_log: true

when: "test_users is false"



- name: Add borg user to sudoers with no password
```

ansible.builtin.copy:

src: /home/borg/tailscale-project/borgaddin

dest: /etc/sudoers.d/

owner: root

group: root

mode: "0600"

when: "test_users is false"

- name: Turn off cloud init

ansible.builtin.file:

path: /etc/cloud/cloud-init.disabled

mode: "0600"

```
state: touch



- name: Restart cloud-init

ansible.builtin.service:

name: cloud-init

state: restarted



- name: Upload SSH key

ansible.posix.authorized_key:

user: borg

key: "{{ lookup('file', '/home/borg/.ssh/id_rsa.pub') }}"

state: present
```

```
when: "test_users is false"




- name: install and enroll tailscale on hosts

hosts: new_guests

become: true

vars_files:

- ./vars.yaml

- ./vault.yaml




tasks:

- name: install and enroll host in tailscale

ansible.builtin.shell: "{{ tailscale_auth_command }}"
```

when: "'tailscale' is not in ansible_facts.packages"

- name: setup subnet routers pt 1

ansible.builtin.shell: echo 'net.ipv4.ip_forward = 1' | sudo tee -a /etc/sysctl.d/99-tailscale.conf

when: "'tailscale' is not in ansible_facts.packages"

- name: setup subnet routers pt 2

ansible.builtin.shell: echo 'net.ipv6.conf.all.forwarding = 1' | sudo tee -a /etc/sysctl.d/99-tailscale.conf

when: "'tailscale' is not in ansible_facts.packages"

- name: setup subnet routers pt 3

ansible.builtin.shell: sudo sysctl -p /etc/sysctl.d/99-tailscale.conf

when: "'tailscale' is not in ansible_facts.packages"

- name: advertise routes for systems

ansible.builtin.shell: tailscale up --advertise-routes "{{ subnets }}"

```

I also wrote a few scripts to test things...

```YAML

### This playbook is just a tester to see how the when flag(?)

### it also checks out the package_facts module

### This playbook is useful for figuring out

- name: check if packages exist, skip or run

hosts: new_guests

become: true

gather_facts: false

```
tasks:

- name: gather facts about packages installed

ansible.builtin.package_facts:

manager: auto

- name: run arbitrary command if a specific package is installed

ansible.builtin.debug:

msg: this is a test1

when: "'libnss-mdns' is in ansible_facts.packages"

- name: run an arbitrary command if a package is not installed

ansible.builtin.debug:

msg: this is another more diffrent test
```

when: "'nmap' is not in ansible_facts.packages"

- name: run arbitrary command if a specific package is installed

ansible.builtin.debug:

msg: this is a test2

when: "'libnss-mdns' is not in ansible_facts.packages"

- name: run an arbitrary command if a package is not installed

ansible.builtin.debug:

msg: this is another more diffrent test2

when: "'nmap' is in ansible_facts.packages"

```

```YAML

```

This script sets a hostname on systems targeted

matches hostnames to the inventory hostname because why not

- name: set hostname for hosts

hosts: targeted systems

become: true

tasks:

- name: Set a hostname

ansible.builtin.hostname:

name: "{{ inventory_hostname }}"
```

Revision #1
Created 26 June 2025 04:08:08 by Keith Matthews
Updated 26 June 2025 04:08:27 by Keith Matthews