Tailscale templates

A bit more on the ansible stuff

[[Tailscale Ansible Automation]]

The steps I have got though to setup tail-scale Templates for Proxmox.

- create fresh Ubuntu VM

- user is called 'administrator'

- password will be kept for admin

users setup for guests will have sudo access only for tailscale functions

XOXO- QEMU guest agent installed

- tailscale installed

- subnet router rules applied to system

- udp optimizations setup

- avahi-daemon setup for mdns id of systems

- machine id removed

- ssh host information and keys removed

- cleared out Ethernet device classifications

Onboarding a user:

- create user information for the user in the vault warden, they need to add the install script per linux server onboard steps from tailscale

- Get them to setup tailscale account and one node on their stuff they plan to use for their microspace stuff

- split DNS setup for yeticraft.net (thanks josh)

- need to know what IP range they will be allowed to access

- need to know if they need additional network interfaces to make that happen

- will want their email address to send them the link to authorize their account

- I should figure out how to email...
https://docs.ansible.com/ansible/latest/collections/community/general/mail_module.html

Does the user need access to the system???

Need to experiment...

I don't think so... If we put in the correct information from an admin perspective we should be able to run tailscale up --advertise routes 192.168.x.0/x with all stuff included. Then we paste them the login link. I think that there is a way to

Need to talk over with the fellas if this seems like a safe and or sane way of doing stuff...

avahi-daemon (mDNS linux client) installed to get hostname. tailscale nodes will be updated based on hostname

The link below is the fix for ssh not working (when you wipe out template keys for the template they need to be setup again on the new vm)

https://www.reddit.com/r/Proxmox/comments/sgyv24/ssh_after_clone_doesnt_work/

Vm's that are full clones can be independent of the the Ceph cluster. We are probably going to go with linked clones. Not sure how that works on the backside but it does seem to be functional

Things an admin will need to do.

- create machine based on template

  - users should have a tag with there name on it and

  - select linked clone (unless putting on system not in Ceph. Probably don't do that)

  - add to remote-access pool (haven't created this yet)

  - wait for VM to be created

  - select correct network interfaces before brining system up (this will dictate the IP address the user gets)

- run `sudo dpkg-reconfigure openssh-server` to generate fresh keys

- import ssh keys (gh or other)

- change hostname to correct numerical designation `hostnamectl hostname tailscale-node#` replace # with number

- reboot system to change hostname

- add new host to new guest inventory section

- set administrator user as a no password user (reworking the template may be necessary for this)

- run ansible playbook to setup system (Crowdsec and other stuff)

- walk user though getting authkey for a Linux server. This is not an awesome way to go about this. I have a feeling that getting input from the fellas could lead to better solutions...

-

- tailscale up --advertise-routes *route ip based on allowed addresses*

- ~~give user the login link (need to figure out how long this will wait) I could potentially automate this bit with ansible and email... That would be cool~~

- move to regular guest inventory

- ensure nymph can get to the system for regular updating

- ensure the user can get to services they expect they should be able to get to.

- Will need to implement a full follow along guide for user.

Today I got this mostly working. I don't really know if there are better ways I could do this for now but this seems like an excellent start.

I am going to do a second version. There are a few things I need to get correct

---

After review with the u-space team this is how I'm gonna tweak stuff

A better procedure generally would be such

- user installs tailscale, enorles at least one system, setup split dns

- enroll user in bitwarden (get auth key for user along with having them generate passwords for whatever services they will be using) we should encourage users to use the vault-warden account for all passwords they generate for the project they are in. That way resets are easier for us. This is ultimately where LDAP and SSO will eventually come into play

- create VM for users access including correct network interfaces. There's a bunch of steps here, will outline later

- add VM to ansible creation inventory

- run initial install playbook

- when user has their auth key in the system, paste into ansible var for auth key as part of playbook to spin up system

- the tailscale join should be the last playbook. Should include the subnet routing to users expected resources. (Nextcloud, dev environment like coder, help desk software, redmine, oodoo, WordPress site backend, ECT.)

- user needs to enable subnet routes in the admin console and also the host system the are using (--accept-routes for Linux. Assuming that windows is a toggle. Android auto forwards afaik)

- check with user that they can login to their services

Steps I need to complete.

- Build a fresh template. I think I'm going to try and automate the install and setup of system variables like the tailscale install, setting up the hostname, configuring to use mDNS and other items at the top of this note can all be automated and we can use the cloud init tools to create the VM. Use the techno tim video to get this VM setup.

- probably need to implement logging push up to network monitoring tools (this is something Josh/garth should be consulted on)

- harden nymph

https://cloud-images.ubuntu.com/noble/current/noble-server-cloudimg-amd64.img

Stuff for the ansible playbook

Tailscale optimizations

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

for setting up nodes with this new setup.

1. go to template 8000

2. create a clone

3. target node: mss1/2

4. VM ID: leave alone (let it auto populate)

5. Name: tailscale-node#

6. resource pool: tail-nodes

7. mode: full clone

8. target storage: same as source

9. create clone

10. do not turn on vm once its cloned!

11. set up network devices as needed

12. turn on system

13. get ip address for host (mdns will be installed)

14. ensure nymph can contact hosts

15. run config ansible playbooks (what do those playbooks do?)

Lets talk about password manager integration...

I have yet again, bounced off of working with a password manager and ansible. the issue ive got is that all the implementations I can think of seem incredibly fragile and dificult to setup. I also realised some of the limitations of the community edition of vault warden. my main issue is the lack of granularity when it comes to password user control.

It seems that we would have to add users to the vaults that they need access to, (part of that is sending them an invite link) Once they are in the system they would have access to both their vault and the organization they need access to. they would then add their auth tolken to their

personal account and then share it with an admin user?

---

roleplay

Here is my tabletop of a user and admin interaction

In this scenario, a user needs to gain access to several development machines as well as nextcloud for document storage and redmine for project managment. the admin has access to the proxmox cluster to provision this new user a system with access to their cluster of systems. the admin also has access to bitwarden. the project lead sends a message to admin group to establish users in this group.

Project Manager:

Good morning, I need users1, user2, user3 and user4 to have access to my workgroup X. please enroll them. here are their email addresses

Admin Managment:

Check rodger, those users will be added

Admin: *send email to users*

Good morning, Im going to help enroll you in our system. you will recive an invite to join the bitwarden account with access to your workgroups vault. please create your account. Once complete, i will send you an enrolement invite to join the u-space organization along with adding you to relevant password vaults for your work group

Once your account is created you'll also create a tailscale account using this link https://tailscale.com/ you will need to provide an identity provider such as github or google. If you already have a tailscale account you can skip this step.

Once you have tailscale setup on your devices of choice, You will need to go to the settings menu in the tailscale admin dashboard. then select keys, and then generate an auth token. your key for this account should not be re-usable. do not select any of the switches.

That auth-tolken will then be saved under your name in a note with the "enroll tolken vault collection"

From there, I will create your machines. Please reach out to me if you have troubles signing up for tailscale or the bitwarden account

Breif overview of steps

1. sign up for yeticraft vault warden

3. sign up for tailscale

4. admin will enroll you in enroll vault collection and other relevant vaults for your work group

5. generate an auth key and save it to your vault

6. send that item to admin@admin.example

Your remote access will be established to the systems in work group X. if you need additional access please talk to your project manager about provisioning those resources.

Users: do the tasks as outlined

Admin:

- create systems. one for each user. add users systems network interfaces to software defined network for that work group. create entrys for those users in the ansible inventory. then get into the bitwarden account and get the auth tolkens. these should be added to an ansible vault and mapped to the system ID they belong to.

- configure ansible vault with auth tolkens and ip addresses for internal ip's of the systems

- running the playbook to enrole new users will run any hardening steps required, configure a strong administrator password, install tailscale and authorize the system for the users network as well as allowing the subnets needed for the workgroup.

A few notes

Im going to spend some time working on play optimization steps. for example, stripping out the gather facts step if it is not required, dont try to install shit if its already there and dont create users that already exist.

This explains the check_mode tool in the users module which is helpful

https://stackoverflow.com/questions/75211712/ansible-user-module-check-if-a-user-exists-or-not

checkout these modules for checking if stuff is already there

https://docs.ansible.com/ansible/latest/collections/ansible/builtin/package_facts_module.html

---

So how did this all end up?

Well It seems that things worked pretty well. Currently this has all largely come together. What I have now is a system that will allow many users access to internal services for projects. This system still needs a bit of refinement as I had to do a good amount of tweaking to get everything working. I think that I need to run though this whole process again now that the ansible scripts are working. I may also move my access into the microspace to that method and see how it works to access internal services. Everything internal to the machine that Im on should be fairly fast given that there's no real network connections.

As I finished up this project a project came to my attention to host a tailscale proxy system that proxy's all the services in its docker network. This project is fairly easy but I'm not sure if it would work for our needs over all since it seems like its to allow one user to get access to resources. I am going to do a deeper dive into this TSDProxy tool and see if it can be applied to our usecase. It could be that for users who dont need access to full virtual machines and only need access to internal services like nextcloud this could be a more appropriate tool to use.

I have gotten tsdproxy to work, The next step is to see if I can get multiple tailscale accounts link up to a single tailscale proxy stack. that way it can be much more efficent to give access to services for many users we can just setup

Somehow your template got all fucked up. gonna try again using

https://dev.to/minerninja/create-an-ubuntu-cloud-init-template-on-proxmox-the-command-line-guide-5b61

https://blog.themaxtor.boo/post/2024-10-12-how-to-create-proxmox-template-with-cloud-image/

https://blog.themaxtor.boo/post/2024-10-19-how-to-create-vm-with-cloud-init-in-proxmox/

I was able to re-establish the Ubuntu-cloud template. this was an excellent starting point and i was very glad to have it back. I was finally able to crack being able to add customized cloud init user files.

I think the problem I was running into had several causes. Mostly though I think the main issue was that I was incorrectly formatting the cloud init files and expecting things that were not possible. It seems like you can either use the values entered in the proxmox ui for cloud init or you can use the ci-custom command to overwrite those things with a customized version. You only get to use one. This means that I will end up tweaking the procedure for an admin to spin up the tailscale nodes significantly. the new procedure looks like this once all data is gathered from the user.

1. select the Tailscale-Template and create a clone

2. select mss1/mss2

3. leave vmid alone

4. give name of ts-node# (number of the tailscale node your adding)

5. add to Tailscale-Node Resource pool

6. select Full Clone mode

7. leave target striagre at same as source

8. leave format as QEMU image format (qcow2)

9. Wait for system lock to go away

10. Do not start system yet

11. add system to relevant network (default is simple LAN (fartnet), this may or may not be an appropriate option for each user, that system should have some sort of dhcp method for assigning ip addresses.

12. Tag each system with the tsnode tag, the users name as a tag, and the project they are on

13. start new system

14. you can watch the system go though its first boot process, wait for it to finish this process or at least wait for 10 minutes

15. reboot system once cloud init has run

16. the system should now report its ip address to the summary page using the qemu-guest-agent, enter this ip address into the ansible playbook for these new machines in the new guests section.

17. enter user auth tokens into the ansible vault in the keys feild (It is very likely that this portion could be automated)

Revision #1
Created 26 June 2025 04:07:29 by Keith Matthews
Updated 26 June 2025 04:07:53 by Keith Matthews