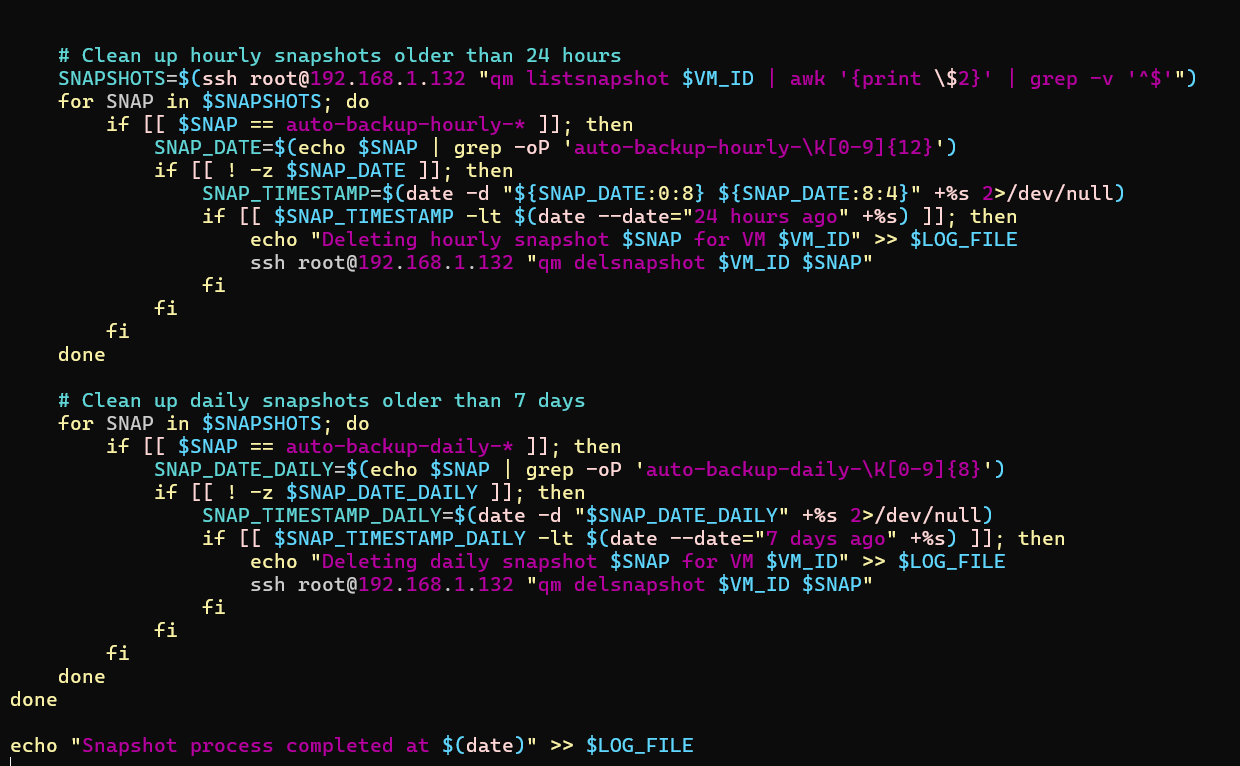
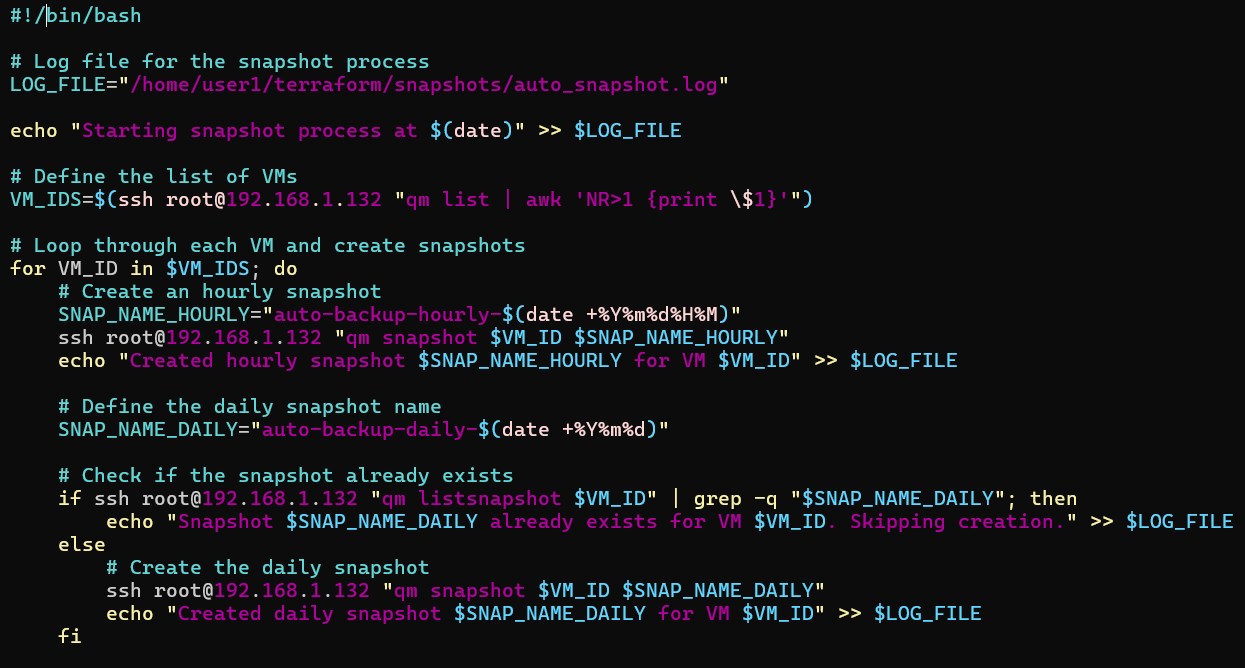
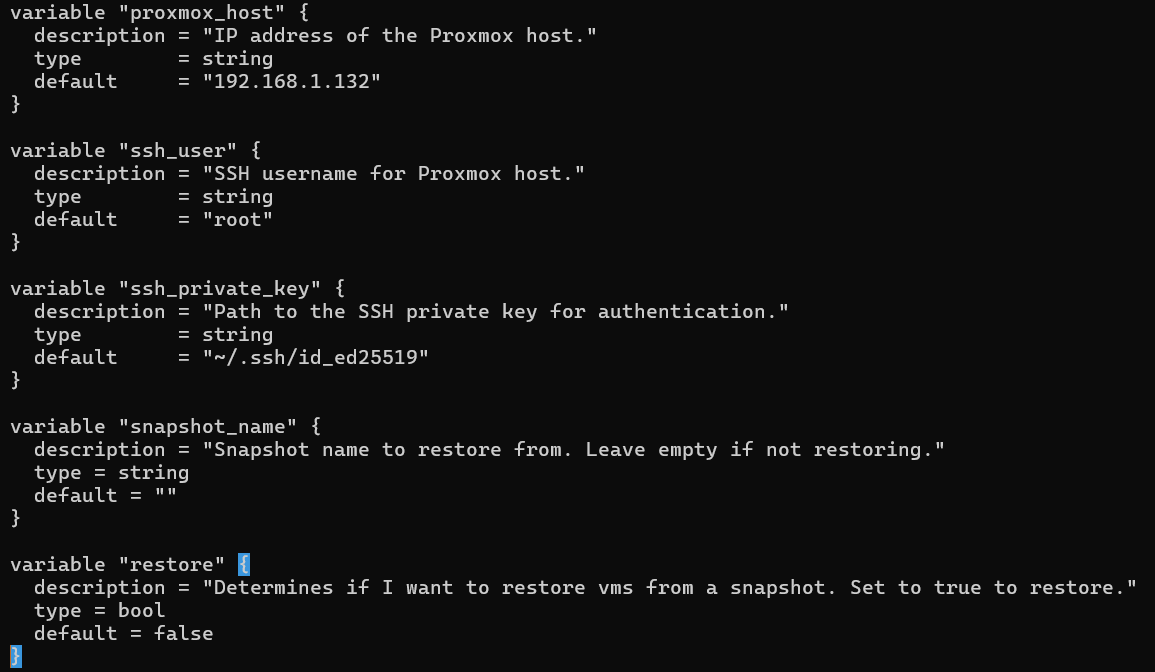
Snapshot Automation Process

1. **Create SSH key pair and deploy**
   1. All scripts are run from my terraform VM but to manage snapshots I must run commands from the host node pve1. To do this I created an ssh key pair so I can authenticate to the host node and not worry about passwords.
   2. Command: ssh-keygen
      1. This generates the key pair
   3. Command: ssh-copy-id [root@192.168.1.132](mailto:root@192.168.1.132)
      1. This command installs the public key on the host pve1
2. **Create script auto\_snapshot.sh**
   1. This script gets a list of VM ids and creates an hourly and daily snapshot for each VM. Then deletes hourly snapshots older than 24 hours and daily snapshots older than 7 days. It also sends some minor data to log file auto\_snapshot.log.

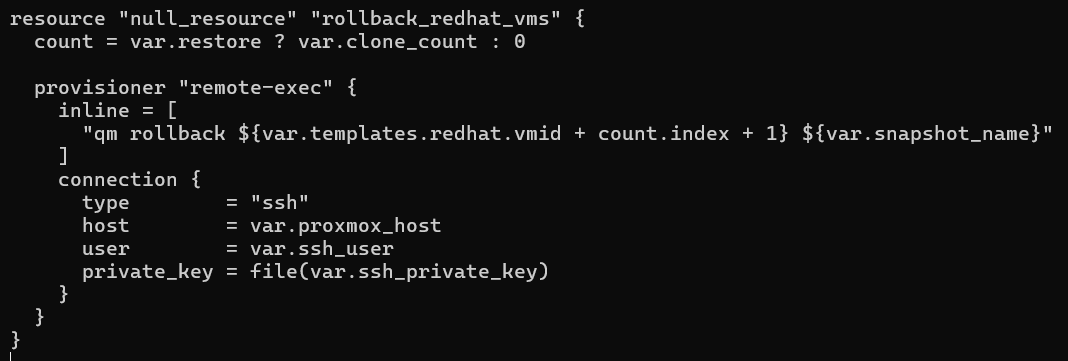


1. **Set execute permissions for auto\_snapshot.sh**
   1. Command: chmod +x auto\_snapshot.sh
   2. There will now be an asterisk after the file name showing all can execute it “auto\_snapshot.sh\*”.
2. **Schedule cronjob for auto\_snapshot.sh**
   1. Command: Crontab –e
   2. 0 \* \* \* \* /home/user1/terraform/snapshots/auto\_snapshot.sh
      1. Insert this line into the crontab file it will schedule auto\_snapshot.sh to run at the top of the hour every hour of everyday.
3. **Add variables to variables.tf**
   1. Create variables proxmox\_host, ssh\_user, ssh\_private\_key, snapshot\_name, and restore. Variables proxmox\_host, ssh\_uer, and ssh\_private\_key are used to authenticate to pve1 host server so that terraform resources can look up and rollback to specified snapshots.
   2. proxmox\_host: The ip address for host pve1.
   3. ssh\_user: Which user I want to authenticate as on pve1.
   4. ssh\_private\_key: Path to my private key so that terraform can complete the 3-way handshake with pve1.
   5. snapshot\_name: The name of the snapshot I want to rollback to, this should be blank if I don’t want to rollback.
   6. Restore: This is a Boolean value to determine if I want to rollback or not.



1. **Add a null\_resource for each current resource** 
   1. A null\_resource is useful to trigger a provisioner which can run a script. By using the null\_resource and referencing the snapshot\_name and restore variables I can run the qm rollback for each VM. Because terraform is triggering the script it sees the state change of each VM and updates its state file to reflect the changes. This has prevented the issue of terraform losing control of its VMS.
   2. **This process can be done externally to terraform through a script if you would prefer but then you must take steps to integrate it with terraform which can be tricky.**

Example null\_resource



1. **Notes on the finished process**
   1. Terraform is tricky and running something with terraform/telmate that it wasn’t intended to do can have odd results. So, when you execute the rollback process terraform plan will say creating 8 VMs. Then when you turn off the rollback process and try to make additional changes it will say destroying 8 VMs. This is incorrect. What it is doing is not running the null\_resources, for terraform that means removing them but they only run scripts so nothing will be removed. The only ones that matter are the main resources so if terraform plan is only mentioning destroying null\_resources its running fine.