**16 - R710 Proxmox Hypervisor - disable SWAP**

This sets up a bare-bones minimal system with a very small (~2GB disk) Ubuntu box

Most instructions came from this link:

<https://austinsnerdythings.com/2021/08/30/how-to-create-a-proxmox-ubuntu-cloud-init-image/>

from this video:

<https://www.youtube.com/watch?v=1sPG3mFVafE>

Do these steps:

1. Open proxmox root terminal and as **root**, add ‘sudo’ to proxmox:  
   **apt-get install sudo**
2. Also as root, do:  
   **apt update -y && apt install libguestfs-tools -y**
3. Open proxmox root terminal and as root, create user ‘rhys’:  
   **adduser rhys**  
   **usermod -a -G sudo rhys**

**usermod -a -G kvm rhys**

1. Install midnight commander for filesystem easy navigation and file editing:  
   **apt install mc**
2. From new Terminal, log into ‘rhys’ and do:

**ssh-keygen -t rsa -b 4096**

1. Create directory named: ‘**test1**’ and ‘**cd**’ into it.
2. Create directory named: ‘original’ and ‘cd’ into this.  
   Then we download the base Ubuntu image with:  
   **wget** [**https://cloud-images.ubuntu.com/focal/current/focal-server-cloudimg-amd64.img**](https://cloud-images.ubuntu.com/focal/current/focal-server-cloudimg-amd64.img)
3. Do: ‘**cd ..**’
4. Create file: ‘create-template.sh’:

**touch create-template.sh**  
**chmod +x create-template.sh**

in this file put:

**#!/usr/bin/bash**

**# run this with sudo**

**# change Data1 to local-lvm when needed**

**image\_name = focal-server-cloudimg-amd64.img**

**rm ${ image\_name }**

**cp original/${ image\_name } ${ image\_name }**

**# Modify files to allow login (can’s see what the next line does, look again !!!)**

**virt-edit -a ${image\_name} /etc/cloud/cloud.cfg -e 's/lock\_passwd: [Tt]rue/lock\_passwd: False/'**

**virt-edit -a ${image\_name} /etc/cloud/cloud.cfg -e 's/lock\_passwd: 1/lock\_passwd: 0/'**

**virt-edit -a ${image\_name} /etc/cloud/cloud.cfg -e 's/ssh\_pwauth: 0/ssh\_pwauth: 1/'**

**#virt-edit -a ${image\_name} /etc/ssh/sshd\_config -e 's/PasswordAuthentication no/PasswordAuthentication yes/'**

**virt-edit -a ${image\_name} /etc/ssh/sshd\_config -e 's/PubkeyAuthentication no/PubkeyAuthentication yes/'**

**virt-edit -a ${image\_name} /etc/ssh/sshd\_config -e 's/\#PubkeyAuthentication yes/PubkeyAuthentication yes/'**

**virt-customize -a ${ image\_name } --install qemu-guest-agent**

**# inject the SSH keys into the cloud image itself before turning it into a template and VM.**

**# You need to create a user first and the necessary folders:**

**virt-customize -a ${ image\_name } --run-command 'useradd rhys; echo “rhys\nred” | passwd rhys'**

**virt-customize -a ${ image\_name } --run-command 'mkdir -p /home/rhys/.ssh'**

**virt-customize -a ${ image\_name } --ssh-inject rhys:file:/home/rhys/.ssh/id\_rsa.pub**

**virt-customize -a ${ image\_name } --run-command 'chown -R rhys:rhys /home/rhys'**

**virt-customize -a ${ image\_name } --run-command 'chmod 700 /home/rhys/.ssh'**

**# Create Proxmox VM using modified image**

**qm create 9000 --name "ubuntu-2004-cloudinit-template" --memory 2048 --cores 2 --net0 virtio,bridge=vmbr0**

**qm importdisk 9000 ${ image\_name } Data1**

**qm set 9000 --scsihw virtio-scsi-pci --scsi0 Data1:vm-9000-disk-0**

**qm set 9000 --boot c --bootdisk scsi0**

**qm set 9000 --ide2 Data1:cloudinit**

**#qm set 9000 --serial0 /dev/ttyS0**

**qm set 9000 --serial0 socket --vga serial0**

**qm set 9000 --agent enabled=1**

**# Resize the primary boot disk (otherwise it will be around 2G by default)**

**# This step adds another 8G of disk space, but change this as you need to**

**# NOTE: !!! this does not appear to work !!!**

**qm resize 9000 virtio0 +8G**

**# Convert VM to a template**

**qm template 9000**

**# Clone the template into a full VM and set some parameters**

**qm clone 9000 999 --name test-clone-cloud-init**

**#qm set 999 --sshkey ~/.ssh/id\_rsa.pub**

**qm set 999 --ipconfig0 ip=192.168.124.207/24,gw=192.168.124.1**

**qm start 999**

1. Create a file: ‘delete-template.sh’  
   **touch delete-template.sh**

**chmod +x delete-template.sh**

in this file put:

**#!/usr/bin/bash**

**# run this with sudo**

**qm stop 999**

**qm destroy 999**

**qm destroy 9000**

1. To create the template, create VM from template and run, do:

**sudo ./create-template.sh**The above must be run as root to have access to the ‘**qm**’ commands in the script.

1. To log into the running machine, (from where one is currently logged in – on proxmox):

**ssh** [**rhys@192.168.124.207**](mailto:rhys@192.168.124.207)

**NOTE:** the above log in can not be done from WSL terminal -> see other document for Terraform usage of cloud-init image where it gives the new VM an initial ssh key that matches the one on the ‘machine’ running the terraform script.

1. When done, log out of the machine
2. To delete the VM and template, do:  
   **sudo ./delete-template.sh**
3. Before repeating and running the above process again, the last entry in **~/.ssh/known\_hosts** will need to be deleted
4. **-=-=-**  
   create a Ubuntu server VM in which to do the following, also in this new VM create user rhys and do:  
   **ssh-keygen -t rsa -b 4096**
5. put following into test2 directory:  
     
   in file **1-install-terraform.sh**

**curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -**

**sudo apt-get install software-properties-common**

**sudo apt-get update**

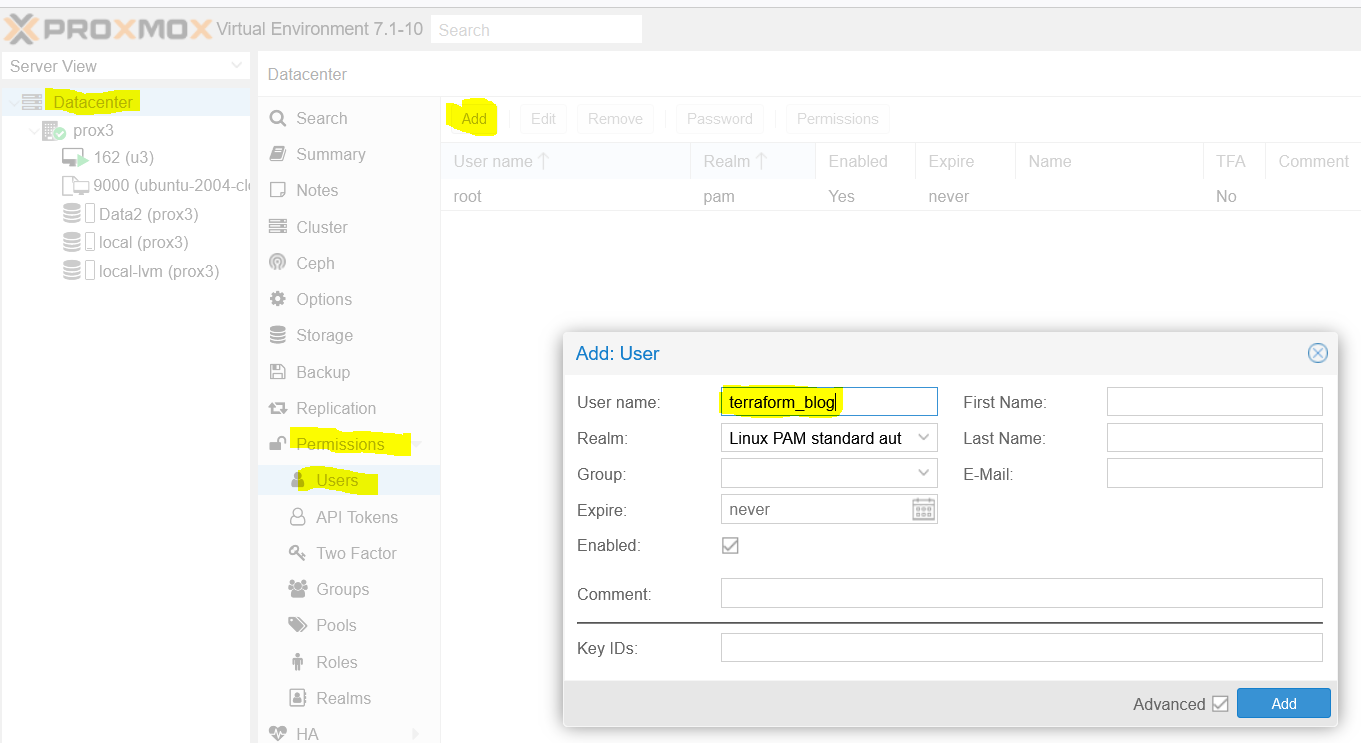
**sudo apt-add-repository "deb [arch=$(dpkg --print-architecture)] https://apt.releases.hashicorp.com $(lsb\_release -cs) main"**

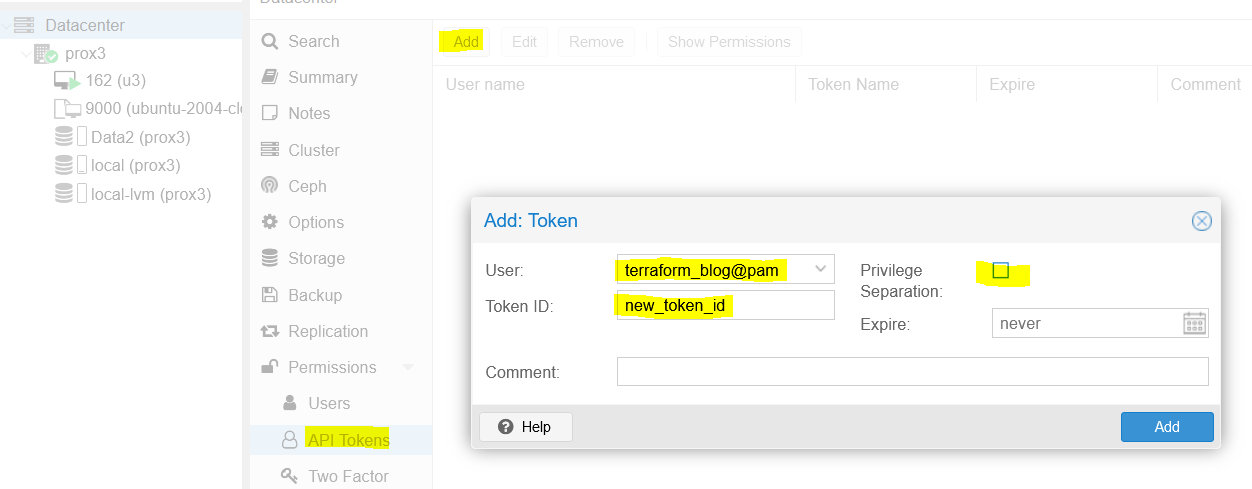
**sudo apt update**

**sudo apt install terraform**

make it executable and run it

1. Copy steps from blog link:  
     
   **https://austinsnerdythings.com/2021/09/01/how-to-deploy-vms-in-proxmox-with-terraform/**  
     
   to create file: notes.txt, where I did:

  
and:



And copy and save the displayed token:

Token Secret

Token ID:

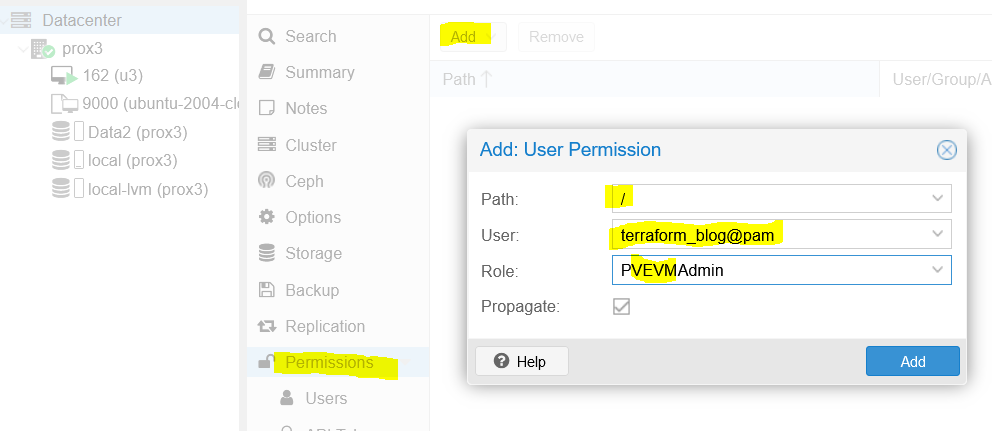
**terraform\_blog@pam!new\_token\_id**

Secret:

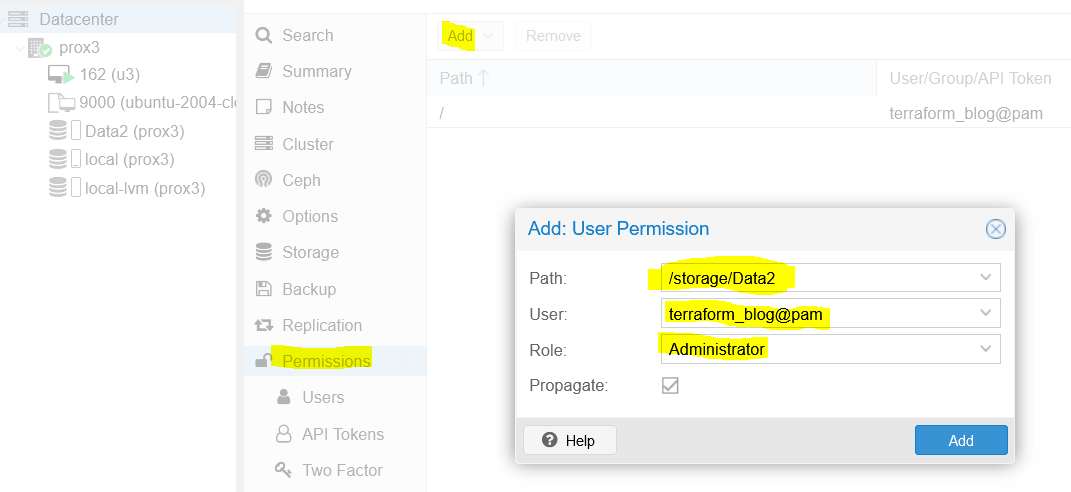
**0bca378d-8ce7-4282-9686-5046207b6ba3**

-=-=-

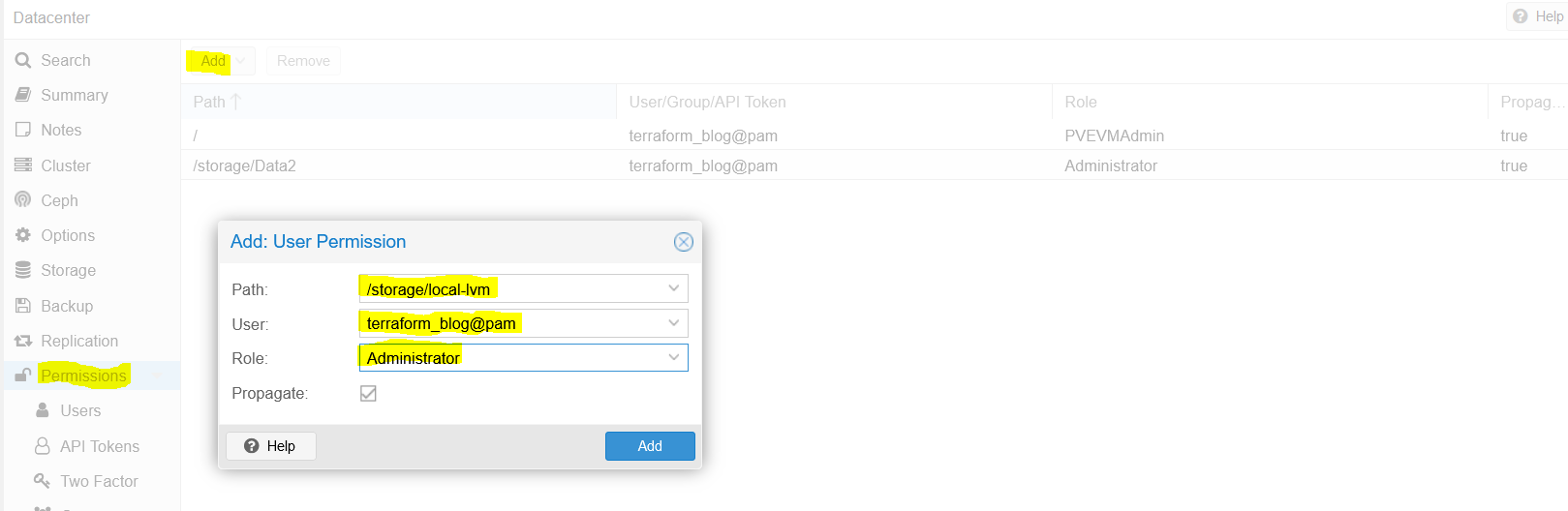
And:



And:



And:



-=-=-

1. In terraform-blog directory, do:

create main.tf as per article above.

then run:

terraform init

terraform plan

terraform apply

-=-=-

I had to copy my ssh.pub file contents over the example in vars.tf

i could then log into the new vm as user 'ubuntu' from the terminal i had opened on prox1 as user 'rhys'

\*\*\* i could not log in from a WSL terminal on A8 as ubuntu ... complained about public key

-=-=-

other articles to investigate:

https://whattheserver.com/proxmox-cloud-init-os-template-creation/

https://registry.terraform.io/providers/Telmate/proxmox/latest/docs/guides/cloud\_init

https://registry.terraform.io/modules/sdhibit/cloud-init-vm/proxmox/latest

// the following seems usefull to work through (**do it all to learn !!!**)

https://yetiops.net/posts/proxmox-terraform-cloudinit-saltstack-prometheus/

https://registry.terraform.io/providers/Telmate/proxmox/latest/docs/resources/vm\_qemu

https://github.com/Telmate/terraform-provider-proxmox/blob/master/examples/cloudinit\_example.tf

See these for tips on enabling ssh password:  
https://serverpilot.io/docs/how-to-enable-ssh-password-authentication/  
  
https://gist.github.com/parente/0227cfbbd8de1ce8ad05

<https://askubuntu.com/questions/988845/how-to-set-passwordauthentication-as-yes-through-automation>

Using Packer: (do this exercise)

<https://salmonsec.com/blogs/home_lab_3>

1. Then in folder: terraform-blog  
     
   in file: main.tf  
     
   terraform {

required\_providers {

proxmox = {

source = "telmate/proxmox"

version = "2.7.4"

}

}

}

provider "proxmox" {

# url is the hostname (FQDN if you have one) for the proxmox host you'd like to connect to to issue the commands. my proxmox host is 'prox-1u'. Add /api2/json at the end for the API

pm\_api\_url = "https://192.168.124.201:8006/api2/json"

# api token id is in the form of: <username>@pam!<tokenId>

pm\_api\_token\_id = "blog\_example@pam!new\_token\_id"

# this is the full secret wrapped in quotes. don't worry, I've already deleted this from my proxmox cluster by the time you read this post

pm\_api\_token\_secret = "bb215dc4-cef6-4b89-8ec3-31fd2acc117c"

# leave tls\_insecure set to true unless you have your proxmox SSL certificate situation fully sorted out (if you do, you will know)

pm\_tls\_insecure = true

}

# resource is formatted to be "[type]" "[entity\_name]" so in this case

# we are looking to create a proxmox\_vm\_qemu entity named test\_server

resource "proxmox\_vm\_qemu" "test\_server" {

count = 1 # just want 1 for now, set to 0 and apply to destroy VM

name = "test-vm-${count.index + 1}" #count.index starts at 0, so + 1 means this VM will be named test-vm-1 in proxmox

# this now reaches out to the vars file. I could've also used this var above in the pm\_api\_url setting but wanted to spell it out up there. target\_node is different than api\_url. target\_node is which node hosts the template and thus also which node will host the new VM. it can be different than the host you use to communicate with the API. the variable contains the contents "prox-1u"

target\_node = var.proxmox\_host

# another variable with contents "ubuntu-2004-cloudinit-template"

clone = var.template\_name

# full\_clone = 1 # create machine and record image sizes, then put this in and repeat

# basic VM settings here. agent refers to guest agent

agent = 1

os\_type = "cloud-init"

cores = 2

sockets = 1

cpu = "host"

memory = 2048

scsihw = "virtio-scsi-pci"

bootdisk = "scsi0"

disk {

# set disk size here. leave it small for testing because expanding the disk takes time.

size = "5G"

type = "scsi"

storage = "Data1"

iothread = 1

ssd = 1 # set this if its on ssd

}

disk {

# set disk size here. leave it small for testing because expanding the disk takes time.

size = "1M"

type = "scsi"

storage = "Data1"

iothread = 1

ssd = 1 # set this if its on ssd

}

disk {

size = "2G"

type = "scsi"

storage = "Data1"

iothread = 1

ssd = 1 # set this if its on ssd

}

# if you want two NICs, just copy this whole network section and duplicate it

network {

model = "virtio"

bridge = "vmbr0"

}

# not sure exactly what this is for. presumably something about MAC addresses and ignore network changes during the life of the VM

lifecycle {

ignore\_changes = [

network,

]

}

# Cloud Init Settings

# the ${count.index + 1} thing appends text to the end of the ip address

# in this case, since we are only adding a single VM, the IP will

# be 10.98.1.91 since count.index starts at 0. this is how you can create

# multiple VMs and have an IP assigned to each (.91, .92, .93, etc.)

ipconfig0 = "ip=192.168.124.21${count.index + 1}/24,gw=192.168.124.1"

# sshkeys set using variables. the variable contains the text of the key.

sshkeys = <<EOF

${var.ssh\_key}

EOF

# play around with the following to enable public ssh  
 provisioner "remote-exec" {  
 inline = [  
 "ip a"  
 ]  
 }

}  
  
-=-=-  
in file: vars.tf  
  
variable "ssh\_key" {

default = "ssh-rsa  rhys@prox1"

}

variable "proxmox\_host" {

default = "**prox1**"

}

variable "template\_name" {

default = "ubuntu-2004-cloudinit-template"

}

1. If Terraform can’t seem to create / access a VM it might be that I’ve confused things by interrupting a previous creation / destroy.  
   Determine the proxmox ID of the problematic machine that can’t be stopped or deleted, say for example an ID of **224**, using this command:

**fuser /var/lock/qemu-server/lock-224.conf**  
  
to get its PID, with a return value of say **1872**, then do:

**kill -9 1872**

**qm stop 224**  
  
The VM should then be able to be deleted.