**24 - R710 Proxmox Add 2nd disk to run host and Docker, Nomad for minio, localstack SQS, local Docker registry and gitea**

These notes cover adding a 2nd drive to the run3 host together with Docker and Nomad.

Nomad will orchestrate:

**mino** that will use to 2nd drive,

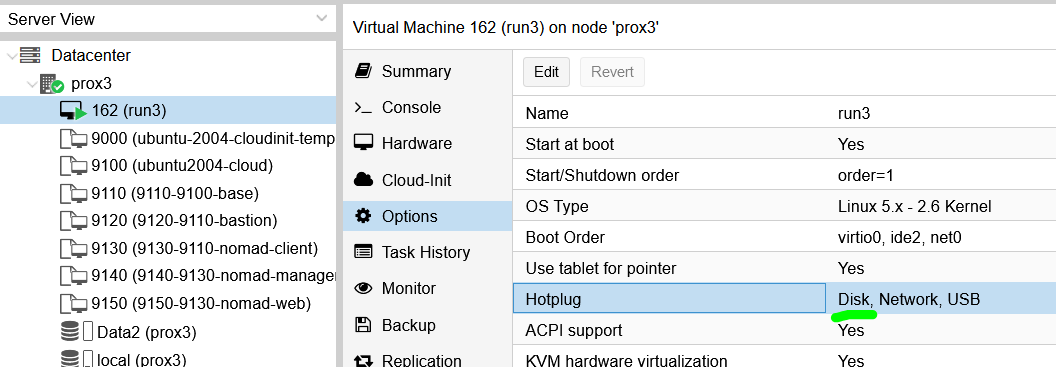
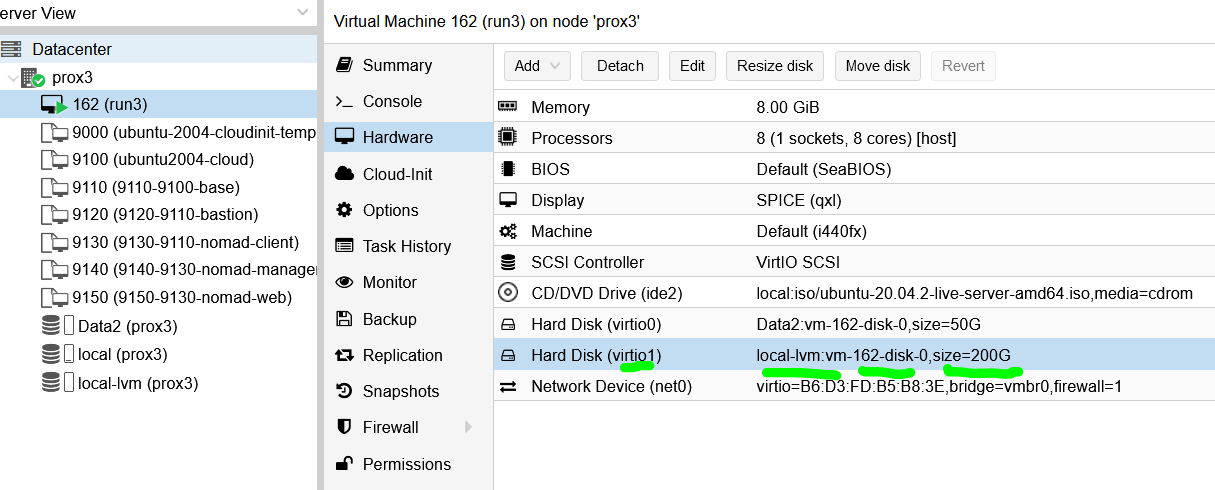
**localstack** to provide **SQS** services,

**Local Docker registry** to save internet bandwidth (using 2nd drive),

**gitea** for local github type repository (using 2nd drive).

This document builds upon the previous documents.

# Add 2nd disk:

1. For the run3 host, in Proxmox GUI, ensure Disk has been selected for Hotplug  
   
2. Again in Proxmox GUI, for the run3 host, for Hardware, click on Add and fill out the info to end up with the 2nd hard disk shown as per:  
   
3. Ssh into run3 host and switch to root with:

**sudo su -**

1. Do:  
   **fdisk -l**  
     
   to confirm that you can see the new 2nd drive as:  
   **/dev/vdb**
2. Enter the following commands:  
   **parted /dev/vdb mklabel gpt**

**parted -a opt /dev/vdb mkpart primary ext4 0% 100%**  
  
**mkfs.ext4 -L S3andSQS /dev/vdb1  
  
mkdir -p /mnt/S3andSQS \*\*\* may need to not create this as ‘root’, and do as ‘rhys’**

1. Edit:  
   **/etc/fstab**  
     
   and add line:  
     
   **LABEL=S3andSQS /mnt/S3andSQS ext4 defaults 0 2**
2. Doing:  
   **fdisk -l**  
     
   should now show something like:  
   *Disk /dev/vdb: 200 GiB, 214748364800 bytes, 419430400 sectors*

*Units: sectors of 1 \* 512 = 512 bytes*

*Sector size (logical/physical): 512 bytes / 512 bytes*

*I/O size (minimum/optimal): 512 bytes / 512 bytes*

*Disklabel type: gpt*

*Disk identifier: 2F482D94-B19B-4E51-B87C-57CD729FD5F5*

*Device Start End Sectors Size Type*

*/dev/vdb1 2048 419428351 419426304 200G Linux filesystem*

1. Reboot the run3 host and as root confirm the 2nd disk is still there and the S3andSQS directory is still there.

# Docker in run3 host

1. To install Docker into the VM, follow Steps:  
   **sudo apt update**

**sudo apt install apt-transport-https ca-certificates curl software-properties-common**

**curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -**

**sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"**

**sudo apt update**

**apt-cache policy docker-ce**

**sudo apt install docker-ce**

**sudo systemctl status docker**

to see something like:

*● docker.service - Docker Application Container Engine*

*Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)*

*Active: active (running) since Mon 2022-03-07 16:09:06 UTC; 26s ago*

*TriggeredBy: ● docker.socket*

*Docs: https://docs.docker.com*

*Main PID: 4188 (dockerd)*

*Tasks: 13*

*Memory: 33.5M*

*CGroup: /system.slice/docker.service*

*└─4188 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock*

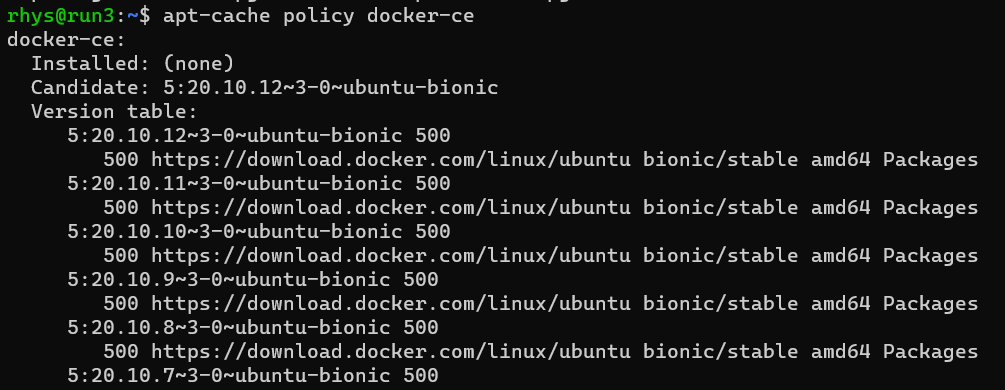
1. Do:

**sudo usermod -aG docker rhys**

then log out and log back in and do:

**id -aG**

to confirm that rhys is in the '**docker**' group.

1. And when you get to the stage of running command:  
   **apt-cache policy docker-ce**  
     
   it produces something like:  
   
2. Now edit file (to enable IPv4 packet forwarding):  
   **sudo nano /etc/sysctl.conf**  
   and look for line:  
     
   and uncomment line to be:  
     
   and then reboot run3 host for changes to take effect.
3. To check Docker is running:  
   **systemctl status docker**
4. To test Docker:

**docker run hello-world**

1. Add **docker-compose** with:  
   **sudo apt install docker-compose**
2. Add **portainer** for use from Ubuntu desktop to manage docker containers with:

**docker run -d -p 9100:9000 --name=portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer\_data:/data portainer/portainer-ce:2.6.0**

# Nomad install on run3 host as server and client

Nomad on run 3 host will operate in a stand alone manner to only orchestrate services for the terraform stack to use.

1. To install Nomad, on run3 host, do:   
   **curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add –  
     
   sudo apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com $(lsb\_release -cs) main"**  
     
   **sudo apt-get update && sudo apt-get install nomad**
2. Check nomad installed ok with:  
   **nomad –version**
3. To test run nomad and access it in browser, do:  
   **nomad agent -dev -bind 0.0.0.0 -log-level INFO**  
     
   Then in a browser, go to:  
   **http://192.168.124.162:4646/**
4. Stop nomad in command line with **CTRL+C**
5. Setup autocomplete and a data dir:  
   **nomad -autocomplete-install**

**complete -C /usr/bin/nomad nomad**  
  
Check where nomad was installed and adjust the following path to suit:  
**sudo mkdir -p /opt/nomad**

1. Add user nomad to docker group with:  
   **sudo usermod -G docker -a nomad**
2. Configuring nomad to run as a service, do:  
   **sudo nano /etc/systemd/system/nomad.service**  
     
   and put the following into it:  
   [Unit]

Description=Nomad

Documentation=https://www.nomadproject.io/docs

Wants=network-online.target

After=network-online.target

[Service]  
Type=simple

User=nomad

Group=nomad

ExecReload=/bin/kill -HUP $MAINPID

ExecStart=/usr/bin/nomad agent -config /etc/nomad.d

ExecStop=/bin/kill $MAINPID

KillMode=process

KillSignal=SIGINT

LimitNOFILE=65536

LimitNPROC=infinity

Restart=on-failure

RestartSec=2

StartLimitBurst=3

TasksMax=infinity

OOMScoreAdjust=-1000

[Install]

WantedBy=multi-user.target

1. nomad settings, do:  
   **sudo nano /etc/nomad.d/nomad.hcl**  
     
   and check it has the following in it:  
   # Full configuration options can be found at https://www.nomadproject.io/docs/configuration

datacenter = "run3"

data\_dir = "/opt/nomad/data"

bind\_addr = "0.0.0.0"

# This hosts name, just to avoid confusion with terraformed hosts

name = "run3"

# Increase log verbosity

log\_level = "DEBUG"

server {

# license\_path is required as of Nomad v1.1.1+

#license\_path = "/etc/nomad.d/nomad.hcl"

enabled = true

bootstrap\_expect = 1

}

advertise {

http = "192.168.124.162:4646"

rpc = "192.168.124.162:4647"

serf = "192.168.124.162:4648"

}

client {

enabled = true

servers = ["127.0.0.1"]

# run3 host has 8CPU's at 1.8GHz : so limit client to 4 CPU's worth:

cpu\_total\_compute = 7200

# run3 host has 8GB RAM : so limit client to 4GB:

memory\_total\_mb = 4096

}

1. Then start the service:  
   **sudo systemctl enable nomad**

(might also need to do)  
**sudo systemctl start nomad**

1. To check the nomad service status, do:  
   **sudo systemctl status nomad**
2. If there are problems, do:  
   **sudo systemctl stop nomad**  
     
   Fix the previous two files and restart to reload new files, with:  
   **sudo systemctl restart nomad**  
     
   or use:  
   **sudo systemctl daemon-reload**
3. Then in a browser (from widows machine or run3 remote desktop gui browser), go to:  
   **http://192.168.124.162:4646/**  
   to see:  
   
4. To test that nomad runs a job OK, do:  
   in /home/rhys create a directory called “jobs”  
     
   In that create a file called “http-echo.nomad”, with contents:  
   job "http-echo" {

datacenters = ["run3"]

group "echo" {

count = 1

network {

port "http" {

static = 8080

}

}

task "server" {

driver = "docker"

config {

image = "hashicorp/http-echo:latest"

args = [

"-listen", ":8080",

"-text", "Hello and welcome to 192.168.124.162 running on port 8080",

]

ports = ["http"]

}

resources {

cpu = 100

memory = 100  
 }

}

}

}

1. Then do:  
   **export NOMAD\_ADDR=**[**http://192.168.124.162:4646**](http://192.168.124.162:4646)
2. Then to check file is ok, do:  
   **nomad job plan http-echo.nomad**
3. Then to run the file, do:  
   **nomad job run http-echo.nomad**  
     
   That should complete OK and in the browser, you should see:  
   
4. You should be able to then see 192.168.124.162:8080 in remote desktop browser and similarly from windows browser:  
   
5. To clear the job out, do:  
   **nomad job stop -purge http-echo**
6. That completes nomad setup on run3 host.

# minio install as Nomad job:

1. !!! Fill in this section …

# localstack install to provide SQS as a Nomad job:

1. !!! Fill in this section …
2. To use / test localstack, awscli will be needed to be installed, with:

**sudo apt install awscli  
  
\*\*\* copy out instructions from below articles \*\*\***

1. Now work thru this to setup SQS and SNS and test them:  
   <https://onexlab-io.medium.com/localstack-sns-to-sqs-47a38f33b8f4>
2. Then test that commands can access the SQS, etc on run3 host from other terraformed VM’s  
   (add notes on what was done here to test)
3. Use any other commands from this:  
   <https://onexlab-io.medium.com/localstack-sqs-a0c36fd13108>  
     
   to do further checks.  
   (add notes on what was done here to test)
4. Do the same for S3 with this:  
   <https://onexlab-io.medium.com/localstack-s3-e28ad393c09>

1. And check I can access from other terraformed machines.  
   (add notes on what was done here to test)
2. Figure out that S3 files are in some s3 directory in the 2nd disk
3. Is there any app / web app that I can use to inspect S3 bucket and also the SQS and SNS queues ?
4. Sus running the docker compose in a detached manner such that it will run at power up / reset of run3 host, so that its always running (or not in a detached state, but running at boot)

1. Work through this article, to run in run3 and also on terraformed host:

<https://joerg-pfruender.github.io/software/docker/microservices/testing/2020/01/25/Localstack_in_Docker.html>

1. Also double check through this to see if its got anything of use:  
   <https://towardsaws.com/sns-and-sqs-with-localstack-using-golang-16b291f45e0b>
2. S3 setup with:  
   <https://onexlab-io.medium.com/localstack-s3-e28ad393c09>
3. In browser, for status of localstack on run3, go to address:

<http://192.168.124.162:4566/health>

This may only show services as available, and not running …

1. Need to run:  
   aws configure  
     
   and enter these dummy parameters:  
   AWS Access Key ID [None]: fred

AWS Secret Access Key [None]: fred2  
Default region name [None]: us-east-1

Default output format [None]:

1. Then to create a testqueue:  
   AWS\_DEFAULT\_REGION=us-east-1 aws --endpoint-url=http://localhost:4566 sqs create-queue --queue-name testqueue  
     
   this responds with:  
   {

"QueueUrl": "http://localhost:4566/000000000000/testqueue"

}

1. Then to create a testbucket:  
   AWS\_DEFAULT\_REGION=us-east-1 aws --endpoint-url=http://localhost:4566 s3 mb s3://testbucket  
     
   this responds with:  
   make\_bucket: testbucket
2. To copy file to bucket:  
   aws --endpoint-url=http://localhost:4566 s3 cp docker-compose.yml s3://testbucket
3. To verify the copy, do:  
   aws --endpoint-url=http://localhost:4566 s3 ls s3://testbucket

# Docker registry install as a Nomad job:

1. !!! Fill in this section …

# gitea install as a Nomad job:

1. !!! Fill in this section …