**28 - R710 Rebuild run3 host as Ubuntu Workstation**

This describes saving off info from existing run3 host that was built on top of Ubuntu server, Replacing everything in the VM with a fresh install of **ubuntu-22.04-desktop-amd64**, configuring all needed applications and reinstating any data saved from old run3 host.

The need for this arose after some grief with ansible and python3 and I mistakenly deleted python3 and before restoring it I rebooted run3, which lost networking and messed everything up.

So a complete rebuild is now the only option.

This document aims to capture a list of everything that needs to be installed to get back the whole of the run3 host, but starting from a Ubuntu workstation, because adding a GUI to a server configuration did not seem to work so well in so far as I could not select the version of gnome I wanted in remote desktop access.

Here goes …

# 1 – Saving existing data from run3 host

1. Shut run3 host down
2. In Proxmox GUI, configure run3 host to boot Ubuntu GUI ISO:  
   
3. Then in Options set the boot order to boot from the ISO first
4. Click on Console and start run3 VM.
5. This will boot the Ubuntu GUI and as it boots, select Try Ubuntu
6. The reason for doing this is to backup various directories whilst they are not being used in a running run3 VM
7. On the left hand side there are icons. Move mouse over them and scroll down to 50G volume and open this.
8. In the folder view window that opens, open up a terminal at that location.
9. Cd into home directory and create a tar.gz file (preserving permissions, etc) of the ‘rhys’ directory. A file of about 6.7GB is created.
10. Open up firefox and log into my online jupyter notebook and drag and drop the created file into a folder. This will take an hour or so to upload.
11. Once uploaded, download it onto windows machine in folder red-run3
12. Remove the tar file that was created in the VM
13. Cd into /
14. Create a tar.gz of /etc and copy to jupyter, download to windows machine and remove the tar file
15. Close the GUI folder for the 50GB partition
16. Open a GUI directory view of the 200GB volume named S3andSQS
17. Right click in that window and select ‘open in terminal’
18. Create a tar.gz of tmp directory and copy to jupyter, download to windows machine and remove the tar file
19. Close the directory, power off the GUI (and press enter to eject the CD).
20. On windows machine, inspect the saved tar files with 7-zip … if any problems found, then parts of the previous steps may need redoing until backups are OK
21. Take a snapshot in proxmox of the current run3 host, just in case future steps need something else from it.
22. That completes the backup of old data

# 2 – Install Ubuntu-22.04-desktop-amd64.iso

1. From the previous steps, run3 VM is setup to boot from **ubuntu-22.04-desktop-amd64.iso**
2. In proxmox for the run3 VM, select the Console view for the VM and start the VM
3. Then referring to Previous document: “**12 – R710 Proxmox vM – Ubuntu Workstation with Static IP**”, follow step 12 onwards under the section titled: “**VM – Full Virtual Machine ( Ubuntu 20.04 LTS )**” … but note we are now using Ubuntu 22.04
4. Also, when asked ‘Who are you’, use name of ‘**rhys**’ and computer name of ‘**run3**’.
5. NOTE: in the document that is being worked thru, where it sets up a static IP, use the ip of 162.198.124.162 for this new setup
6. Then complete the rest of the steps in that document.

# 3 – Check the remote desktop log in works

1. From windows machine try the remote desktop login … it works … FAB.
2. To set the screen blanking time:

Open the Activities overview and start typing Power.

Click Power to open the panel.

Use the Screen Blank drop-down list under Power Saving Options to set the time until the screen blanks, or disable the blanking completely.

# 4 – The previous S3andSQS volume mount

1. Within the proxmox console Ubuntu GUI login it turns out the new install has mounted the previous hosts S3andSQS mount with all of its contents … FAB.
2. However in the remote desktop, opening up the disks icon in the utilities it is not mounted …
3. So in remote desktop, in a terminal, do:  
   **sudo su -**  
   **mkdir -p /mnt/S3andSQS**Edit **/etc/fstab,** and add line:  
   **LABEL=S3andSQS /mnt/S3andSQS ext4 defaults 0 2**and reboot run3 VM
4. In remote desktop, open a terminal and cd into /mnt/S3andSQS and do ls, where you should see the original tmp directory and lost+found. Within tmp you should see minio and other original directories.

# 5 – Ubuntu Desktop additional software, including ‘go’

1. Follow the steps in this document: “**15 - R710 Proxmox VM – Ubuntu Desktop additional Software**”
2. Edit .bashrc and adjust history size:  
   HISTSIZE=100000

HISTFILESIZE=200000

1. At end of the .bashrc, replace the exporting of the go path with this:  
     
   **myssh\_agent () {**

**umask 077**

**local f=~/.ssh/spy kee=**

**if [[ ! -f $f ]]; then**

**ssh-agent -s | grep --color -v '^echo' > $f**

**fi**

**. $f**

**if [[ -z $SSH\_AGENT\_PID || -z "$(ps -p $SSH\_AGENT\_PID | grep ssh-agent)" ]]; then**

**\rm $f**

**myssh\_agent**

**else**

**if [[ -z "$(ssh-add -l | grep '^[0-9]')" ]]; then**

**ssh-add**

**fi**

**for kee in ~/.ssh/id\_rsa4k ~/.ssh/id\_rsa\_$myHOST; do**

**[[ -f $kee ]] || continue**

**local fing=$(ssh-keygen -l -f $kee | awk '{print $2}')**

**[[ -n $fing && -z "$(ssh-add -l | grep " $fing ")" ]] || continue**

**ssh-add $kee**

**done**

**fi**

**}**

**# Go Global variables**

**export GOROOT="/usr/local/go"**

**export GOPATH="$HOME/Go"**

**export PATH="$PATH:$GOPATH/bin:$GOROOT/bin"**

**alias on='cd ~/public/src/github.com/redhug1'**

**set +e**

**set +o posix**

**complete -C /usr/bin/nomad nomad**

1. Exit and restart the terminal

# 6 – Terraform, ansible and further run3 setup

1. do:  
   **ssh-keygen -t rsa -b 4096**
2. On windows WSL machine, in ubunt terminal we need to clear out old ssh stored info, the last entry in **~/.ssh/known\_hosts** will need to be deleted with:  
   **ssh-keygen -f "/home/rhys/.ssh/known\_hosts" -R "192.168.124.162"**
3. do:  
   **sudo apt-get update && sudo apt-get install -y gnupg software-properties-common**
4. We will install Terraform manually to avoid any unwanted updates:  
   Open Firefox and go to:  
   <https://developer.hashicorp.com/terraform/downloads>  
     
   There select and download:  
   **terraform\_1.3.7\_linux\_amd64.zip**
5. In a terminal, in Downloads folder, do:  
   **unzip terraform\_1.3.7\_linux\_amd64.zip**
6. Then do:  
   **sudo mv terraform /usr/bin**  
   **sudo chown root:root /usr/bin/terraform**
7. Verify terraform installed with:

**terraform version**

1. Install “auto-complete” Terraform extension, with:  
   **terraform -install-autocomplete**  
   **source ~/.bashrc**

Install ansible with:  
**sudo apt install ansible  
sudo apt install sshpass**(as of 4th Feb 2023 this installed version 2.10.8 of ansible, python is 3.10.6)

# 7 – Samba on run3 host:

1. In /home/rhys, create directory: **public**
2. Do:  
   **sudo su -**

**apt install samba  
systemctl stop smbd**

**cd /etc/samba**

**mv smb.conf smb.conf.orig  
touch smb.conf**

1. Open file smb.conf in nano and put the following in it:  
   [global]

server string = File Server

workgroup = WORKGROUP

security = user

map to guest = Bad User

name resolve order = bcast wins

interfaces = lo ens18

bind interfaces only = yes

[run3-rhys-public]

# public access

path = /home/rhys/public

force user = rhys

force group = rhys

create mask = 0664

force create mode = 0664

directory mask = 0777

force directory mode = 0775

public = yes

writable = yes

1. Then to check that the contents of **smb.conf** are ok, do:  
   **testparm**  
     
   and fix any problems with further editing of **smb.conf**
2. Then, do:  
   **systemctl start smbd**
3. You can now access in windows file explorer the following path: **\\run3**

# 8 – DNS Server on run3 host:

1. From Document: “**21 - R710 Proxmox run host as DNS server and Terraform VMs pointing at it**”  
     
   Do the section: “**Install DNS server on ‘run3’ host**”

# 9 – NTP Server on run3 host:

1. From Document: “**22 - R710 Proxmox Ansible NTP on Terraformed VMs pointing to NTP server on run host**”  
     
   Do the section: “**Setup NTP on run3 for it to serve Terraformed VM’s**”

# 10 – Packer and make

1. We will install Packer manually to avoid any unwanted updates:  
   Open Firefox and go to:  
   <https://developer.hashicorp.com/packer/downloads>  
     
   There select and download:  
   **packer\_1.8.5\_linux\_amd64.zip**
2. In a terminal, in Downloads folder, do:  
   **unzip packer\_1.8.5\_linux\_amd64.zip**
3. Then do:  
   **sudo mv packer /usr/bin**  
   **sudo chown root:root /usr/bin/packer**
4. Verify packer installed with:

**packer version**

1. ‘make’ will be required, so install it with:  
   **sudo apt install make**

!!! may need to do the parts in section “Getting FULL-STACK code and adjustments for your Proxmox:” for ssh stuff from document #23

OR:  
copy current new ssh keys and copy ssh keys from backup and see if I can log into 192.168.124.164

# 11 – 2nd disk cleanup

1. Steps in preparation for **minio** setup later:  
   [  
   see this article for why minio-user is set up with nologin :  
   <https://docs.docker.com/develop/develop-images/dockerfile_best-practices/#user>

]  
  
**sudo useradd --no-log-init -r minio-user -s /sbin/nologin**  
**cd /mnt/S3andSQS/tmp**

**sudo chown -R minio-user:minio-user minio**

# 12 – Docker in run3 host, and add portainer

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 gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg**

**echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null**

**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 Sun 2023-02-05 10:59:33 GMT; 8s ago*

*TriggeredBy: ● docker.socket*

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

*Main PID: 8485 (dockerd)*

*Tasks: 13*

*Memory: 25.2M*

*CPU: 376ms*

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

*└─8485 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock*

1. Do (to avoid typing sudo whenever you run docker):

**sudo usermod -aG docker rhys**

then log out and log back in and do:

**su - rhys**

**groups**

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:
3. **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:2.16.2  
     
   Note:** portainer is remapped to port 9100 because later ‘minio’ is set up and it uses 9000 and 9001

# 13 – Nomad install on run3 host as server and client

1. We will install Nomad manually to avoid any unwanted updates:  
   Open Firefox and go to:  
   <https://developer.hashicorp.com/nomad/downloads>  
     
   There select and download:  
   **nomad\_1.2.15\_linux\_amd64.zip**
2. In a terminal, in Downloads folder, do:  
   **unzip nomad\_1.2.15\_linux\_amd64.zip**
3. Then do:  
   **sudo mv nomad /usr/bin**  
   **sudo chown root:root /usr/bin/nomad**
4. Verify packer installed with:

**nomad version**

1. Setup autocomplete and a data dir:  
   **nomad -autocomplete-install**

Uncomment the line in .bashrc, so that it looks like this:

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

1. Create a unique, non-privileged system user to run Nomad:  
   **sudo useradd --system --home /etc/nomad.d --shell /bin/false nomad**
2. Add user nomad to docker group with:  
   **sudo usermod -G docker -a nomad**
3. 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. Do:

**sudo mkdir --parents /etc/nomad.d**  
**sudo chmod 700 /etc/nomad.d**  
**sudo touch /etc/nomad.d/nomad.hcl**

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"

log\_level = "INFO"

log\_json = true

enable\_syslog = true

#log\_rotate\_duration = "24h" !!! use this if putting logs somewhere other than syslog

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"

}

plugin "docker" {

config {

#endpoint = "unix:///var/run/docker.sock"

volumes {

enabled = true

}

}

}

client {

enabled = true

servers = ["127.0.0.1"]

# 'node\_class' used to ensure jobs meant for host 'run3' do run on 'run3'

node\_class = "run3"

# 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

host\_volume "minio" {

path = "/mnt/S3andSQS/tmp/minio/data"

read\_only = false

}

host\_volume "localSQS" {

path = "/home/rhys/public/nomad-jobs/localSQS"

read\_only = false

}

}

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. Had to do, as root:  
   **chown :nomad /etc/nomad.d**

**chmod 750 /etc/nomad.d**

**chown -R nomad:nomad /opt/nomad**

1. Then in a browser (from widows machine or run3 remote desktop gui browser), go to:  
   **http://192.168.124.162:4646/**  
   to see:  
   
2. To test that To test that nomad runs a job OK, do:  
   in **/home/rhys/public** create a directory called “**nomad-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

}

}

# reserve 1,000 MiB (or 1GB) – as the default is 300 MiB

ephemeral\_disk {

size = 1000

}

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**

# 14 – minio install as Nomad job

1. With the directories and permissions for **minio** set up previously, we first confirm everything is ok by running the **minio** docker container with:  
     
   **docker run -p 9000:9000 -p 9001:9001 --name minio1 -e "MINIO\_ROOT\_USER=minio" -e "MINIO\_ROOT\_PASSWORD=minio123" -e "MINIO\_USERNAME=root" -e "MINIO\_GROUPNAME=root" -v /mnt/S3andSQS/tmp/minio/data:/data minio/minio:latest server --console-address ":9001" /data**

Then open ‘**localhost:9001/login**’ In a browser to confirm.  
  
If all is OK, CTRL-C in the terminal running **minio1** docker container and use **portainer** to remove the container **minio1** and any unused volumes.

1. In **nomad-jobs** directory create file named “**minio.nomad**”, and put the following in it:

job "minio" {

# 'datacenters' is for human convenience and has no binding significance

datacenters = ["run3"]

type = "service"

# All tasks in this job must run on client agent for host 'run3'

constraint {

attribute = "${node.class}"

value = "run3"

}

# The 'Task Group' name has the name of host 'run3' to indicate the host

# (or the type of host) this job is on:

group "run3" {

count = 1

network {

port "http" {

to = 9000

static = 9000

}

port "console" {

to = 9001

static = 9001

}

}

# increase default disk from 300MB to allow for log growth (for if this is left running)

ephemeral\_disk {

size = 1000

sticky = true

migrate = false

}

volume "minio" {

type = "host"

read\_only = false

source = "minio"

}

task "minio" {

driver = "docker"

volume\_mount {

volume = "minio"

destination = "/data"

read\_only = false

}

env {

MINIO\_USERNAME = "minio-user"

MINIO\_GROUPNAME = "minio-user"

MINIO\_ROOT\_USER = "minio"

MINIO\_ROOT\_PASSWORD = "minio123"

}

# NOTE: I can only get minio to work with ‘root’ user

# I was getting this error in nomad:

# API error (500): unable to find user minio-user: no matching entries in passwd file

# I tried doing what this article suggests, with no luck:

# <https://github.com/hashicorp/nomad/issues/11670>

#

user = "root"

config {

#image = "minio/minio"

# pull specific image as otherwise it seems to keep pulling latest at power up.

image = "minio/minio:RELEASE.2022-05-08T23-50-31Z"

#image = "docker.io/bitnami/minio:2022"

force\_pull = false

args = [

"server",

"--console-address=:9001",

"/data",

]

# the following dns line may not be needed if host run3 were to have console on it

dns\_servers = ["192.168.124.162"]

ports = ["http", "console"]

logging {

# the default logger is 'json-file', which has no log-rotation,

# so change to a logging driver that does:

type = "syslog"

}

}

resources {

cpu = 500

memory = 500

}

}

}

}

1. Before running the job, we need to clear out old files, so do:  
   **cd /mnt/S3andSQS/tmp/minio/data/.minio.sys  
   sudo chown -R rhys:rhys buckets/**

**sudo chown -R rhys:rhys config/**

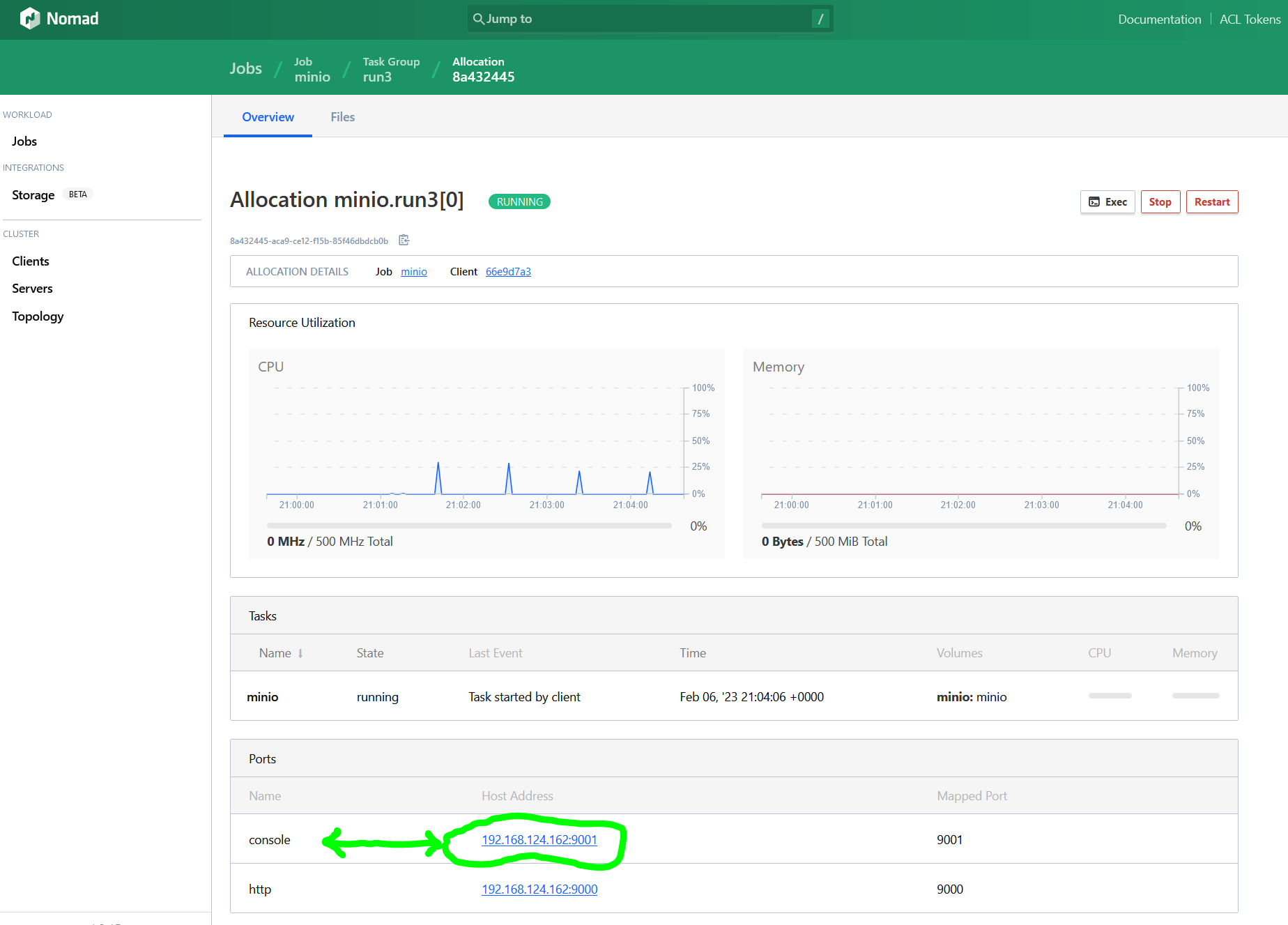
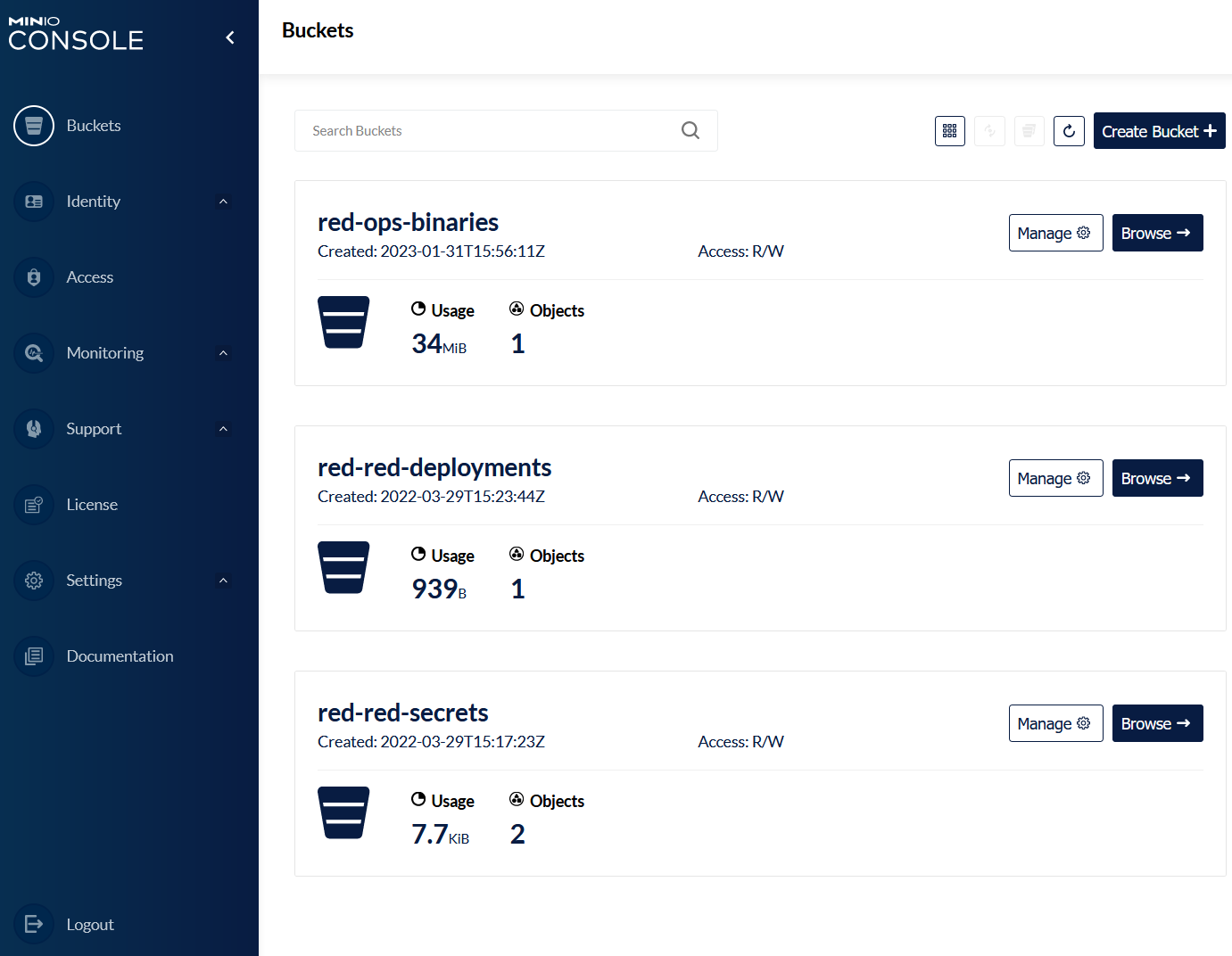
**sudo chown -R rhys:rhys multipart/**

**sudo chown -R rhys:rhys tmp/**

1. And then this may be needed before the job will run OK:  
   **cd /mnt/S3andSQS/tmp/minio  
   sudo chmod -R 777 data**
2. Check the job is ok with:  
   **nomad job plan minio.nomad**

1. Then if all ok, run with:  
   **nomad job run minio.nomad**
2. If you need to totally get rid of the minio job, do:

**nomad job stop -purge minio**

1. Check in Nomad web gui that all is OK and navigate to page that looks like this and click on the console link:  
   
2. To get to the Minio login console.  
   If this works, close that window and check the shortcut in firefox browser on windows machine for minio works and that you can still see previous buckets and their contents:  
   
3. Install the minio command line tool for performing minio tasks on run3 host with the following commands:

**wget https://dl.min.io/client/mc/release/linux-amd64/mc**

**chmod +x mc**

**sudo mv mc /usr/local/bin/mc**

1. To talk to the object server with ‘**mc**’, do:  
   **mc config host add myminio http://192.168.124.162:9000 minio minio123**
2. A few simple mc commands are:  
   **mc ls myminio  
     
   mc ls myminio/red-ops-binaries**  
     
   **mc tree myminio**  
     
   **mc du myminio**
3. From Document “**24 – R710 Proxmox Add 2nd disk to run host and Docker, Nomad for minio**” do the go code section named “**‘go’ code to test access to minio**”, and the expected output should match the minio pick above with something like this from running from run3 (the code may take a number of secs to build and run):  
   2023/02/07 16:19:35 {red-ops-binaries 2023-01-31 15:56:11.284 +0000 UTC}

2023/02/07 16:19:35 {red-red-deployments 2022-03-29 15:23:44.198 +0000 UTC}

2023/02/07 16:19:35 {red-red-secrets 2022-03-29 15:17:23.049 +0000 UTC}

# 15 – localstack SQS install as Nomad job

1. Work through the section ‘**“localstack” install as a Nomad job to provide SQS**’ of the document named: “**25 - R710 Proxmox Add Nomad job - localstack SQS**”
2. Check the initially set up queues exist with:  
   **aws --endpoint-url=http://192.168.124.162:4566 sqs list-queues**

Which should show something like:  
  
**{**

**"QueueUrls": [**

**"http://localhost:4566/000000000000/develop-deadletter-service-events",**

**"http://localhost:4566/000000000000/develop-inbound-service-events",**

**"http://localhost:4566/000000000000/develop-deadletter-deployer-messages",**

**"http://localhost:4566/000000000000/develop-inbound-deployer-messages",**

**"http://localhost:4566/000000000000/develop-service-event-results"**

**]**

**}**

!!! sort out ssh stuff -> backup existing stuff and copy the backed up stuff in its place and see if ssh’ing into run3 from w8 works and sshing from w8 into other terraformed boxes works and sshing into terraformed boxes from run 3 works

# 16 – add vscode extensions and other apps

1. From the document “26 - R710 Proxmox run host - add GUI, vscode extensions and other apps”, work through the sections:  
   “**‘vscode’ – adding extensions**”,  
   “**Other apps**”

!!! still to do the gitkraken bit from doc #26

As the following steps are followed, also detail the restoration of any data

!!! also after each step take a snapshot … and at some point if possible clean out previous snapshots.

Restore all other folders that were in /home/rhys

* Including checking for any dot type folders in the backup to see that they are replicated in restoration

Install and check operation of Gitkraken

* Book in latest code …

Copy bits needed from document 27

* Fix this document 27 and rename it to only cover getting docker registry installed

Take backup of this machine, then do snapshot and remove older snapshots … read proxmox PDF user guide on how to do this.

Do document 29 to cover gitea …

Take regular snapshots … see if I can setup some app on W8 that kicks off a reminder to do this and any other reminders …

Do a readme in red-setup to cover running up mini dev with everything working …