**24 - R710 Proxmox Add 2nd disk to run host and Docker, Nomad for minio**

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

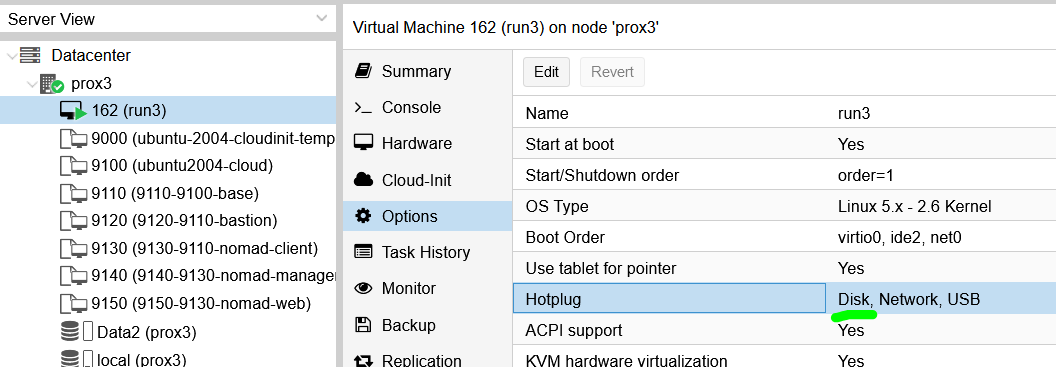
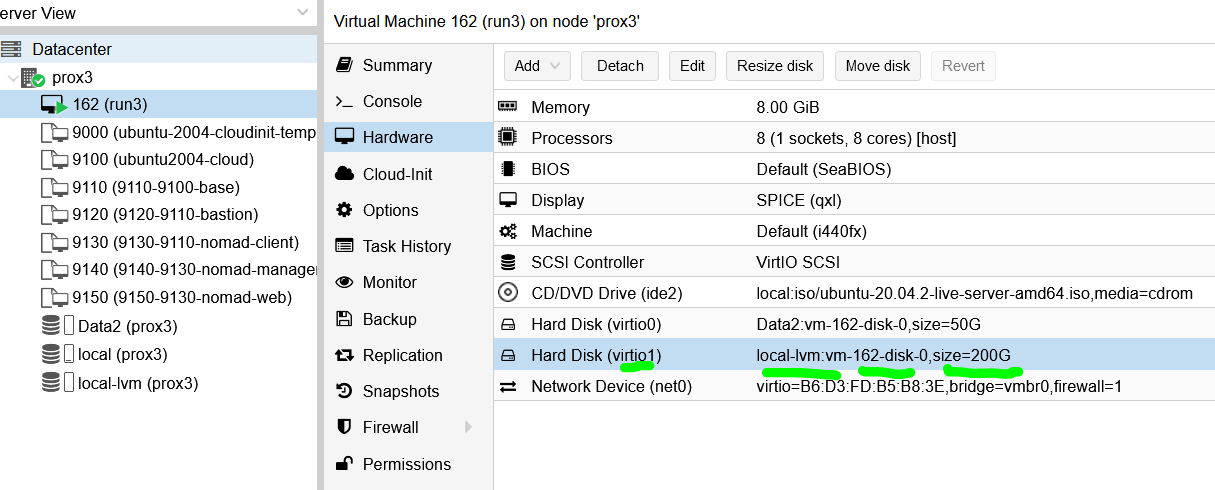
**Nomad** will orchestrate:

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

**Docker** will independently run **portainer** for other container management

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**

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.
2. Now do a few more 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**  
**mkdir tmp**  
**cd tmp**

**mkdir minio**  
**sudo chown minio-user:minio-user minio**

**cd minio**

**mkdir data**  
**sudo chown minio-user:minio-user data**

# 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*

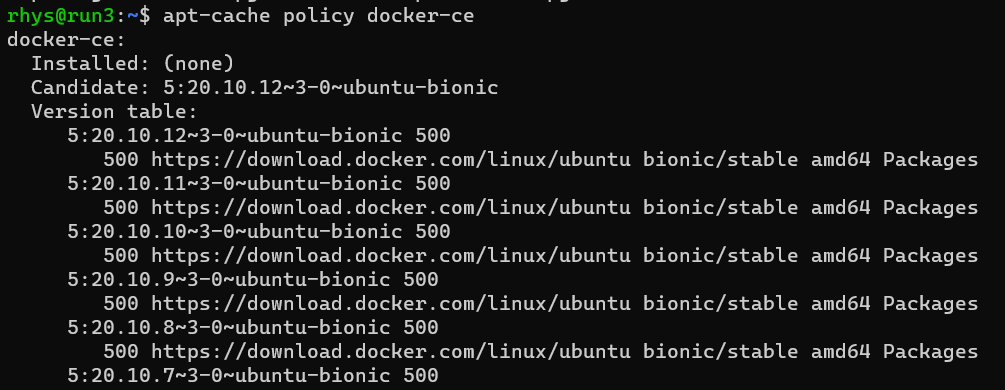
1. Do:

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

# 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**  
     
   to see:  
   *Nomad v1.2.6 (a6c6b475db5073e33885377b4a5c733e1161020c)*
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"

}

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. 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/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**
6. That completes nomad setup on run3 host.

# 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=minio-user" -e "MINIO\_GROUPNAME=minio-user" -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 = ["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"

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

args = [

"server",

"--console-address=:9001",

"/data",

]

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 = 1000

memory = 1000

}

}

}

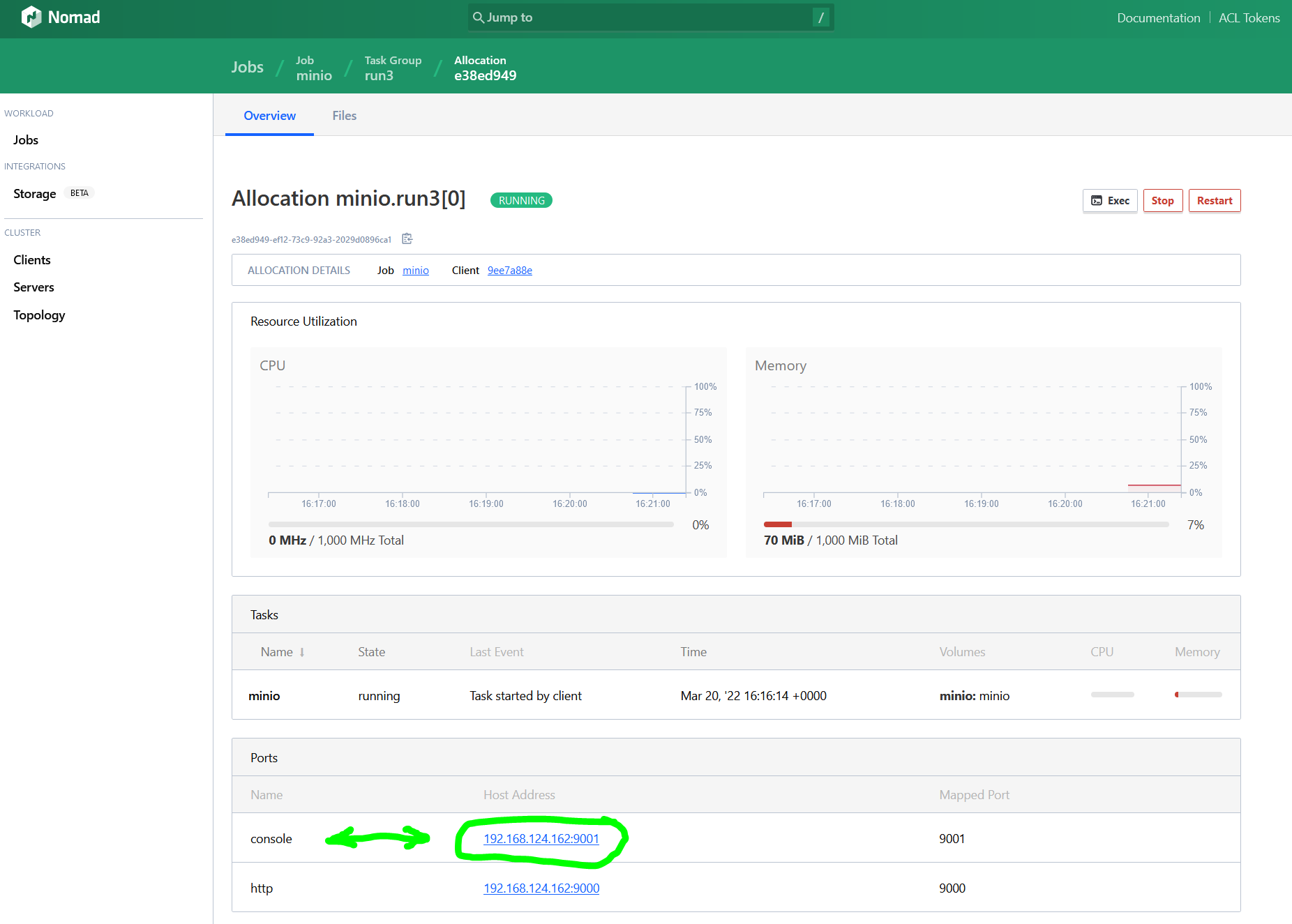
}

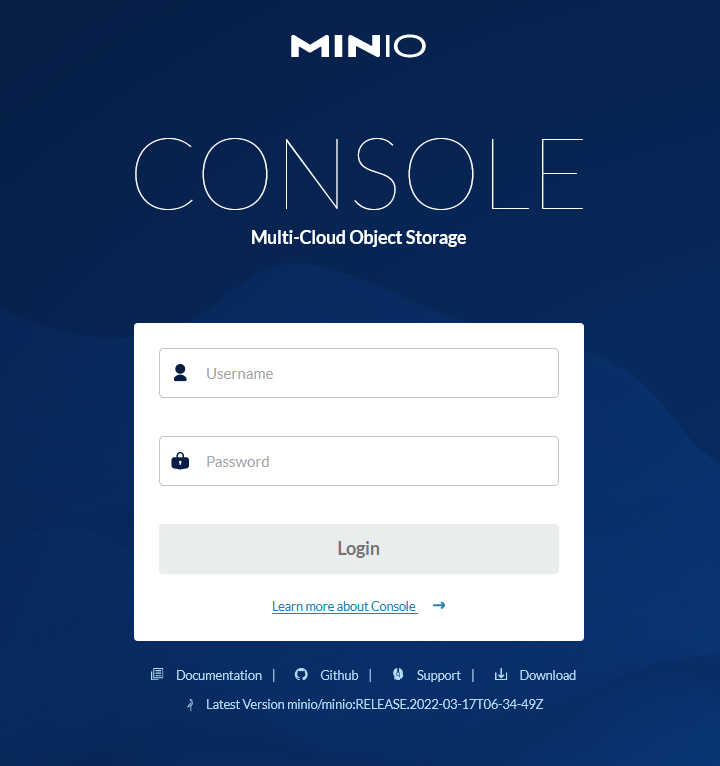
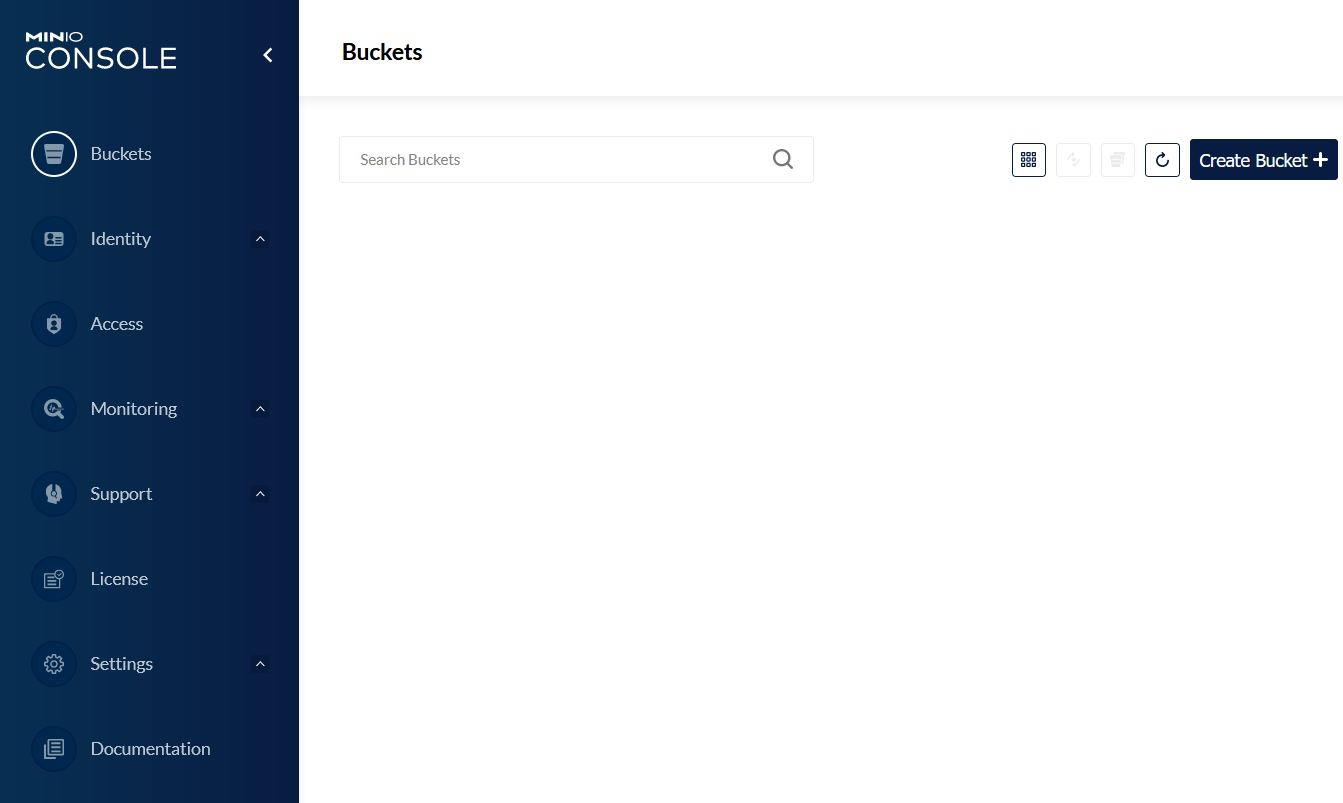
1. Check the job is ok with:  
   **nomad job plan minio.nomad**

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:



1. To get to the Minio login console:  
   
2. Enter “**minio**” for Username and “**minio123**” for Password to get to:  
   

1. From there you can create buckets, upload files into bucket from the machine that the browser is running from and also download, together with other management tasks.
2. In the above gui, add a bucket called ‘test-bucket’ and upload a few small files into it.
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**  
  
to get a response similar to (after you created that bucket in minio gui):  
*[2022-03-17 20:42:36 UTC] 0B test-bucket/*  
  
**mc ls myminio/test-bucket**  
  
to get (for example):  
*[2022-03-19 09:05:35 UTC] 4.5KiB STANDARD autoscale plan.txt*

*[2022-03-17 20:43:49 UTC] 3.2KiB STANDARD m1-1.nomad*

*[2022-03-18 17:39:29 UTC] 1.2KiB STANDARD m1.nomad*

1. To copy a local file into the test-bucket:  
     
   **mc cp minio.nomad myminio/test-bucket/minio.nomad**  
   Then go and check that it is there in the minio console gui.
2. A few other ‘**mc**’ commands:

**mc tree myminio**

**mc du myminio**

# ‘go’ code to test access to minio:

On a windows system with WSL installed and goland installed in that (or another linux host with golang) …

1. Create test directory called ‘**minio-test**’
2. In ‘**minio-test**’ create file called ‘**m-test.go**’ and put the following into it:  
     
   package main

import (

"context"

"log"

"github.com/minio/minio-go/v7"

"github.com/minio/minio-go/v7/pkg/credentials"

)

func main() {

endpoint := "192.168.124.162:9000"

accessKeyID := "minio"

secretAccessKey := "minio123"

useSSL := false

// Initialize minio client object.

minioClient, err := minio.New(endpoint, &minio.Options{

Creds: credentials.NewStaticV4(accessKeyID, secretAccessKey, ""),

Secure: useSSL,

})

if err != nil {

log.Fatalln(err)

}

//log.Printf("Success:\n%#v\n", minioClient) // minioClient is now set up

buckets, err := minioClient.ListBuckets(context.Background())

if err != nil {

log.Fatalln(err)

}

for \_, bucket := range buckets {

log.Println(bucket)

}

}

1. In ‘**minio-test**’ create file called ‘**go.mod**’ and put the following into it:  
     
   module fred/m-test

go 1.18

require github.com/minio/minio-go/v7 v7.0.23

require (

github.com/dustin/go-humanize v1.0.0 // indirect

github.com/google/uuid v1.1.1 // indirect

github.com/gopherjs/gopherjs v0.0.0-20220221023154-0b2280d3ff96 // indirect

github.com/json-iterator/go v1.1.10 // indirect

github.com/jtolds/gls v4.20.0+incompatible // indirect

github.com/klauspost/compress v1.13.5 // indirect

github.com/klauspost/cpuid v1.3.1 // indirect

github.com/minio/md5-simd v1.1.0 // indirect

github.com/minio/sha256-simd v0.1.1 // indirect

github.com/mitchellh/go-homedir v1.1.0 // indirect

github.com/modern-go/concurrent v0.0.0-20180306012644-bacd9c7ef1dd // indirect

github.com/modern-go/reflect2 v1.0.1 // indirect

github.com/rs/xid v1.2.1 // indirect

github.com/sirupsen/logrus v1.8.1 // indirect

github.com/smartystreets/assertions v1.2.1 // indirect

golang.org/x/crypto v0.0.0-20210711020723-a769d52b0f97 // indirect

golang.org/x/net v0.0.0-20210226172049-e18ecbb05110 // indirect

golang.org/x/sys v0.0.0-20210630005230-0f9fa26af87c // indirect

golang.org/x/text v0.3.3 // indirect

gopkg.in/ini.v1 v1.57.0 // indirect

)

1. Then do:  
   **cd minio-test**

**go mod tidy  
  
go run m-test.go**  
  
to see a result similar to:  
  
*2022/03/19 10:34:32 {test-bucket 2022-03-17 20:42:36.189 +0000 UTC}*

1. That completes the setup and testing of **minio**.