napp-it

napp-it ZFS Storageserver as a Cloud-Filer

Setup and First steps

published: 2020-Oct-08

Content:

1. Napp-it and Cloudstorage

1.1 Local Filer

You mainly use napp-it as a local ZFS fileserver on your LAN to share ZFS filesystems via FC/iSCSI, NFS and SMB. You can work directly on the filer mainly via SMB with multiuser filelocking and permission restrictions and a performance that is similar or maybe better than your local disk and much saver due checksums, Copy on Write and snaps.

1.2 Cloud Storage

Cloud storage works different. This is usually storage located elsewhere in the world. Additionally clouds can offer extra services beside storage like calender or multi-user office docs. From a storage view you cannot work directly with the files via a local application just backup, sync and share. From a data security and privacy view, clouds can be a problem as you have only limited control who and where data is located and who has access (Admins, intelligence agencies or related to economic espionage).

You can use napp-it to inhouse share ZFS filesystems to the Internet with the Amazon S3 protocol and minIO. This is an Amazton S3 kompatible OpenSource server application that is supported by napp-it and OmniOS extra repo, https://min.io/product/overview. An S3 share is accessable from the Internet via browser or one of the Amazon S3 compatible backup and sync tools (Google: s3 sync tool). When you enable an S3 share, you must enter a location, a master login with password and the port over which you want to access the service. Additionally you can add S3 users with different permissions. You can use http or preferable https for access.

Main advantage of an S3 cloud server is a quite secure internet access to your files, a superiour performance and the easyness to setup. Enable and it just works. A single binary, no dependencies. . Main disadvantage is that it offers object storage, no authentication or authorisation on a per file level and no filelocking that prevents concurrent editing. This is not a problem for a personal cloud or when used only as a backup/ sync location but is a huge problem in a multiuser mixed filer/cloud environment where you want to have access to files that are editable locally and additional cloud access.

While you can just enable S3 cloud sharing on a filesystem where local SMB sharing is also enabled, you should use this scenario only on a private system with only one or a few users who are aware of the problem that you should not edit/access simulatiously via SMB and S3.

The simple cloud sync and share approach works best if you are the owner of a file that is edited and located ex on your laptop and you want to sync with your desktop or want to share your local files with friends or colleagues.

1.3 Cloud Filer

A Cloud filer is a concept of a traditional local multiuser SMB ZFS filer where your office, company, school or university data are edited and located as primary data source and a cloud storage that allows to sync and share (one or two way) the same files in a multiuser environment. Filer + Cloud = Cloudfiler.

The problems that must be solved:

1.4 Concurrent SMB/S3 access

As concurrent SMB/S3 access is not possible without the problem that concurrent access may result in corrupted files. You cannot allow an S3 cloud client to sync files while they are open ex in a local application and a local application should not open a file via SMB while it is unclear if an S3 client is just updating the file. Best solution to control this is to mirror all files between a S3 and a SMB folder and to sync newer files on demand or a timetable based on snaps in a one or two way sync where you can skip open destination files.

A perfect solution is to hold all files twice, once on an SMB share for local access and once on an S3 share for cloud access.

To update newer files between an SMB folder and an S3 folder use a rclone or rsync job. This can be a one way sync (SMB -> S3) or a two way sync of newer files on demand or based on a timetable and based on snaps to avoid access to open files.

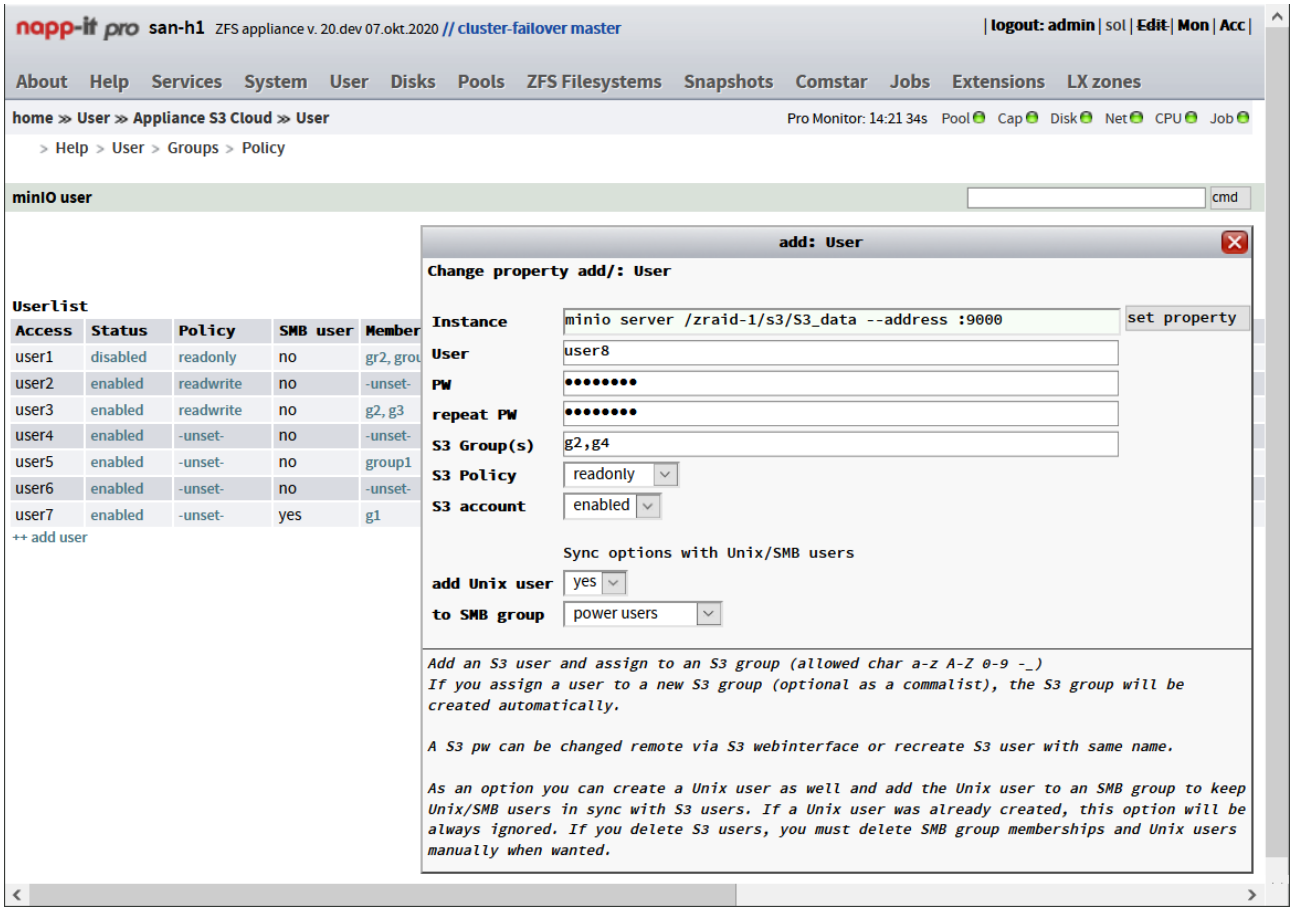
To avoid to store datablock twice, you can use ZFS realtime dedup. To avoid the RAM problem of ZFS dedup, you can use a special vdev ex an NVMe mirror to hold the dedup table. This will effectivley stora datablocks only once while you can access/modify the two copies without problems.

To sync selected files and folders, add a napp-it „other job" with the snap and sync command and start the sync command or script either manually on demand or based on a timetable from once every minute to once a day.

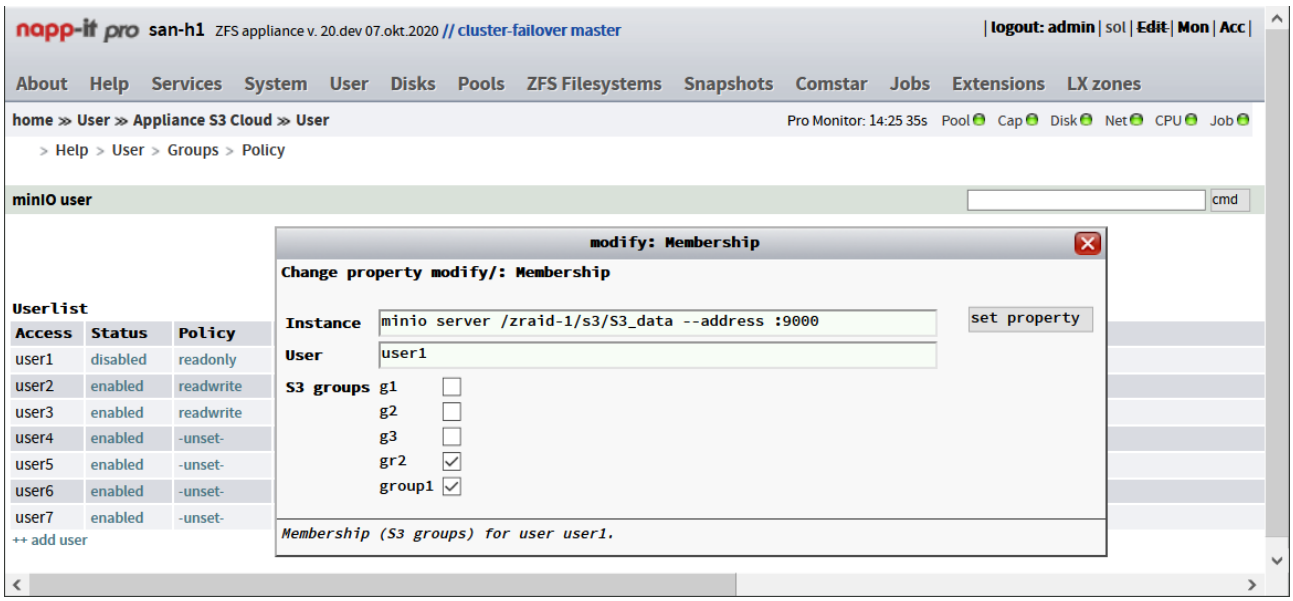
1.4.1 Same user for SMB/S3 access with similar permissions

MinIO allows a user/group concept to access buckets (=folders on an S3 share). While there are no file based permissions, you can use this to allow restrict access to the share or buckets based on a no access, read or read/write. For simultanious access to same files locally and on the cloud, you should keep SMB users and S3 users in sync. While you cannot exactly mirror file permissions on a cloud access, the cloud options should be enough.

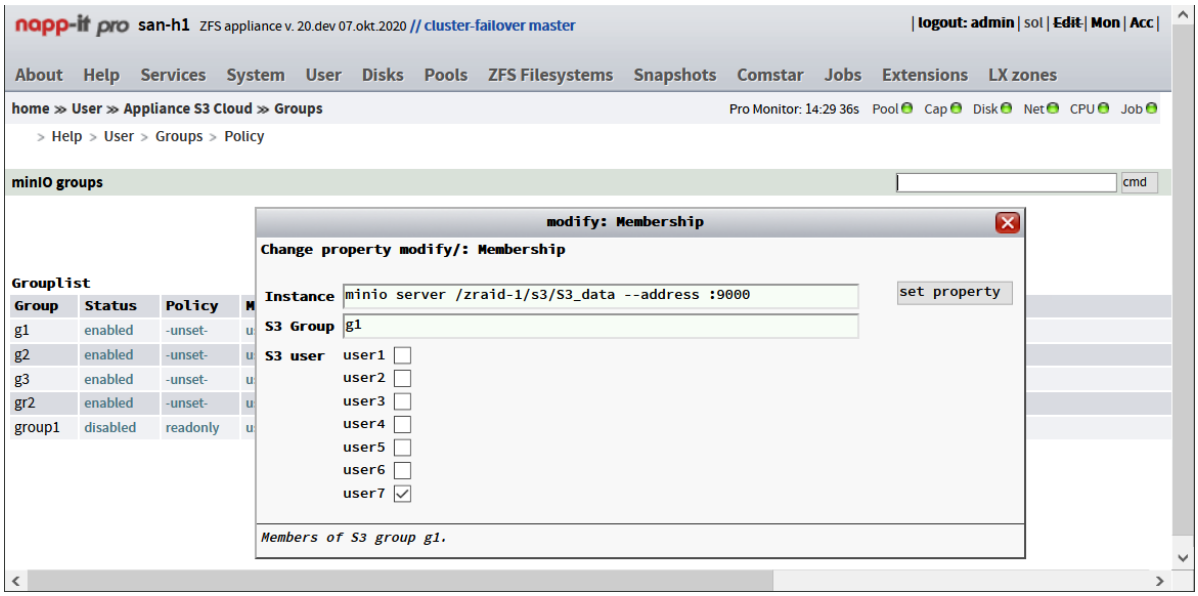
1.5 Create S3 user + SMB user



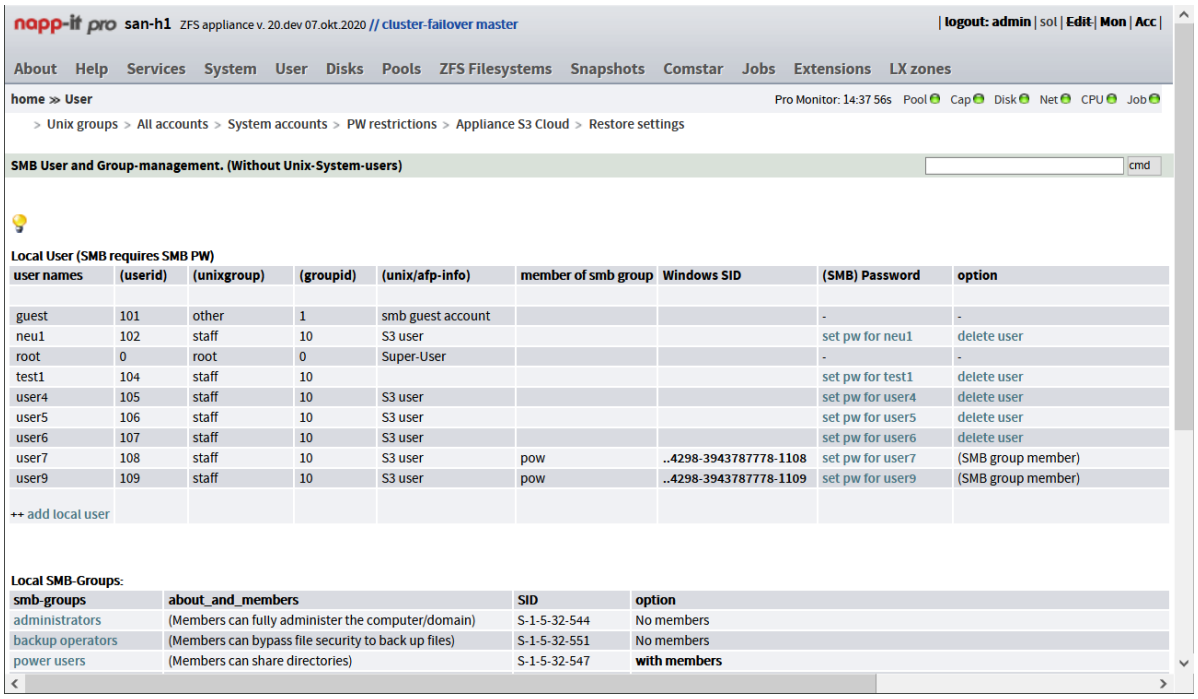
1.6 Edit S3 group membersip for a user



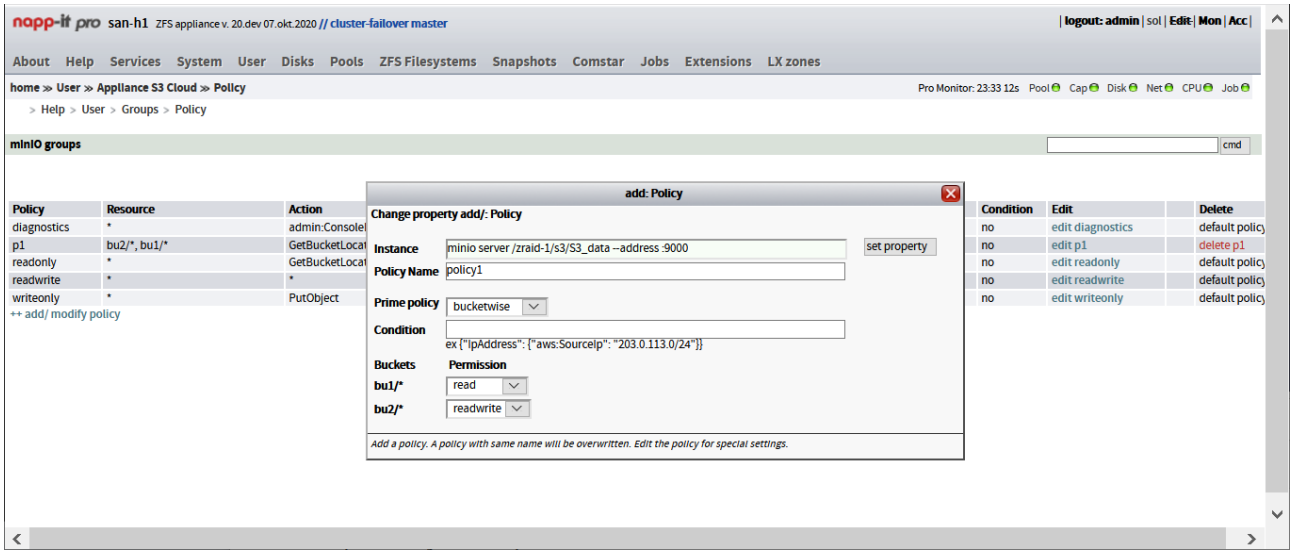
1.7 Edit users of an S3 group



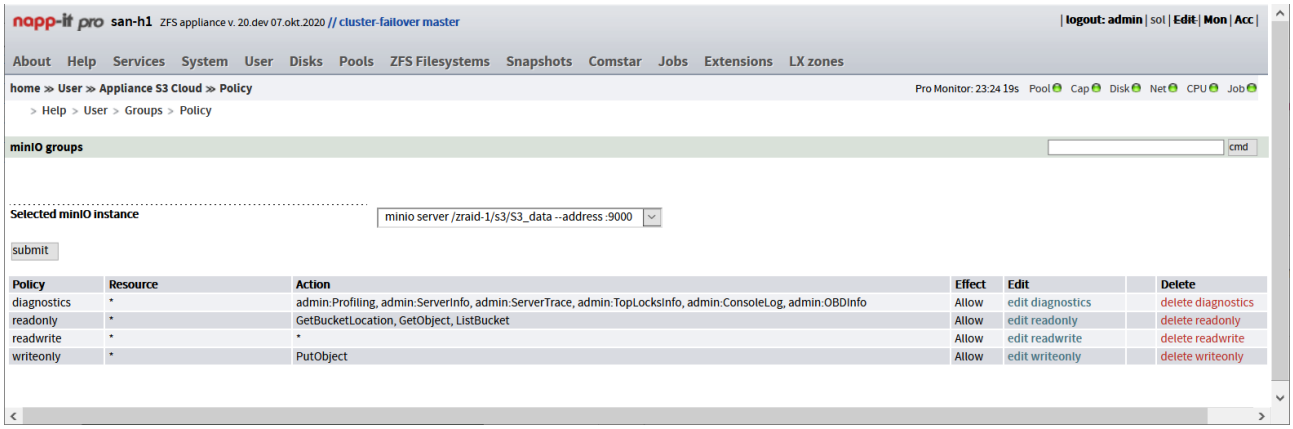
1.8 SMB/ Unix user



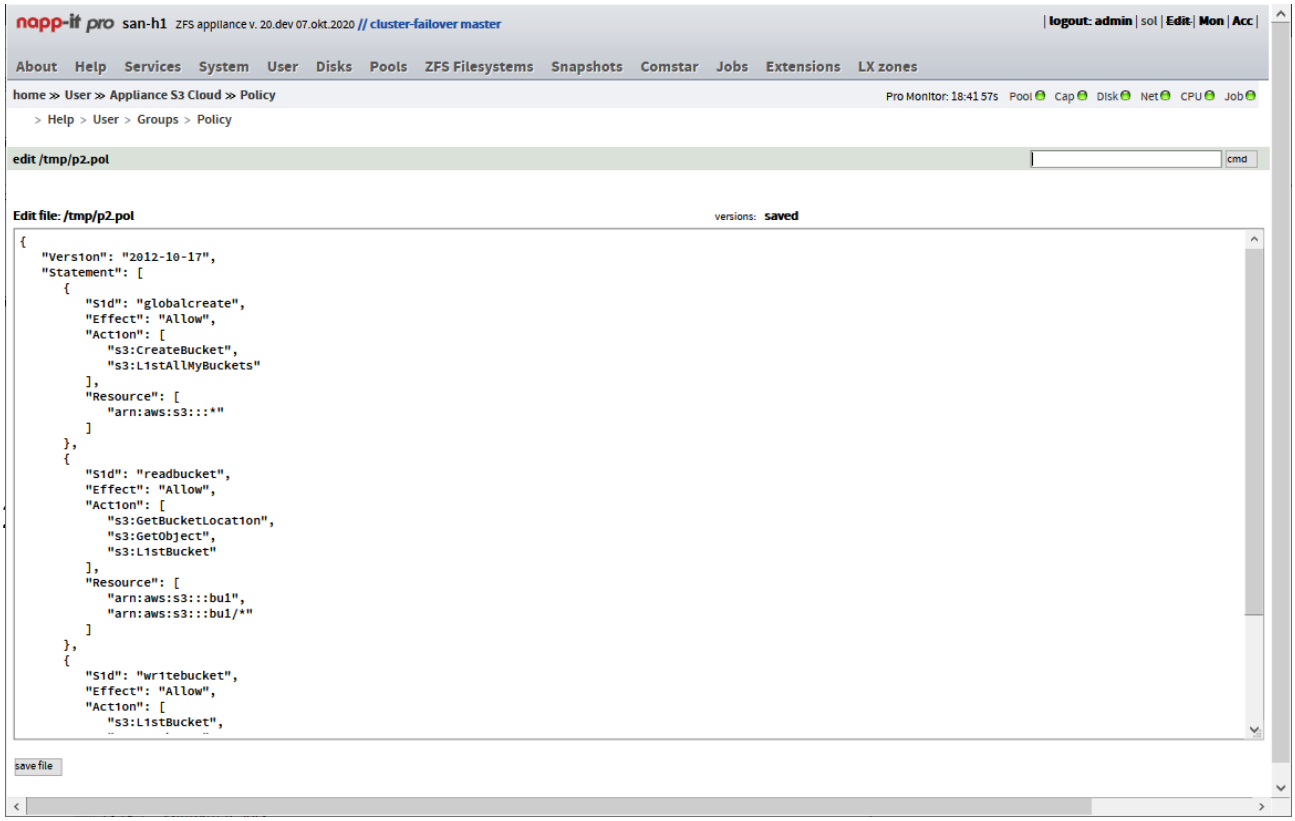
1.9 create Policies



1.10 Show policies



Edit policy

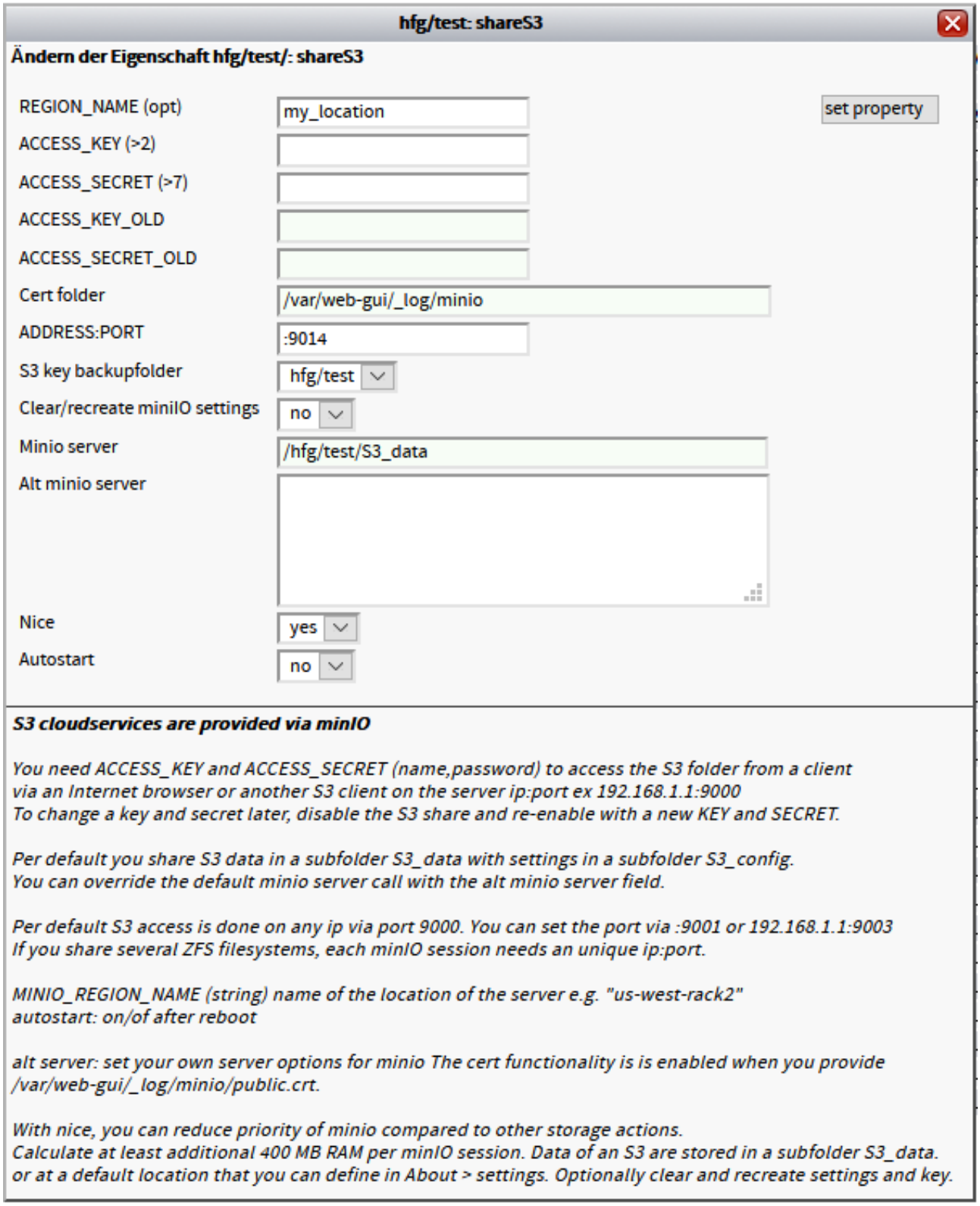


MinIO and rclone are provided by the OmniOS extra repository.
Install via pkg (or napp-it menu Services > S3)

```
pkg set-publisher-g https://pkg.omniosce.org/r151032/extra extra.omnios
pkg install minio
pkg install minio-mc
pkg install rclone
#optionally enable minio as a service
# (I prefer a manual start with options, you can start multiple instances with a different port)
svcadm enable minio
# service manifest, see /lib/svc/manifest/application/application-minio.xml
```

2.1 Enable a ZFS filesystem for S3 internet access

Open menu ZFS filesystems and click under S3_cloud (unset) to enable a S3 share

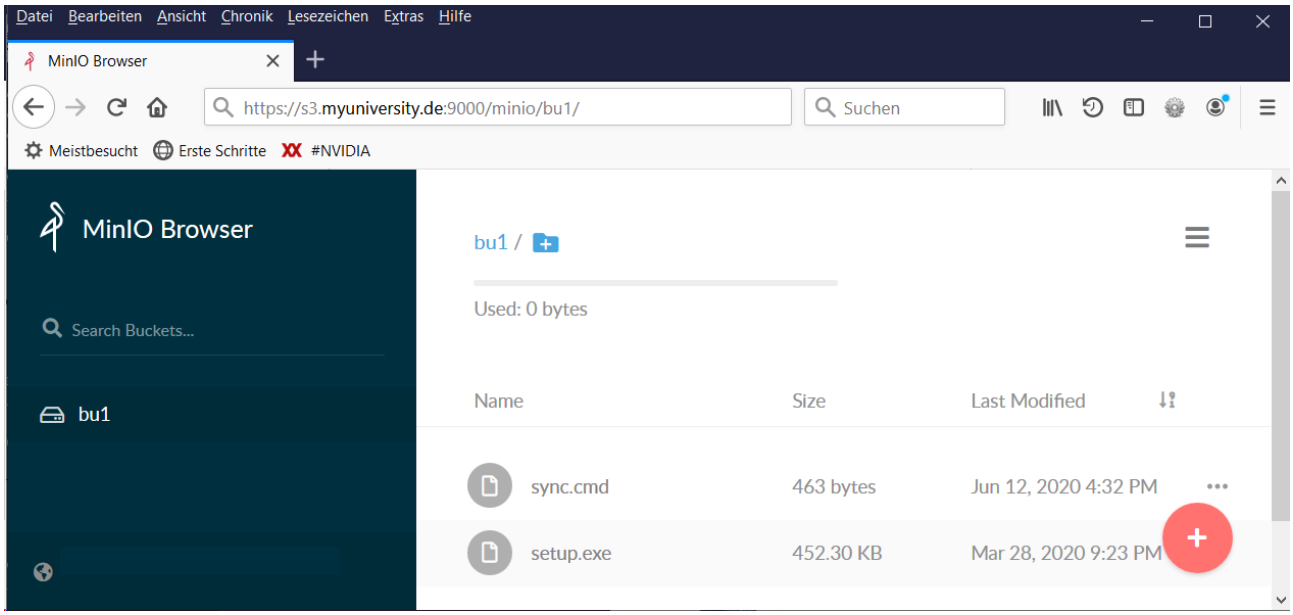


Enter REGION_NAME, ACCESS_KEY, ACCESS_SECRET and port.
You need them later for access.

3. Access an S3 share via browser

Enter the hostname and the port into your browser.

MinIO will ask for access key and secret password. You can then access the ZFS filesystem from elsewhere and upload/download files or create an anonymous link for a file (valid up to seven days)



4. Access an S3 share via GUI tools

To access an S3 share, you can use GUI backup tools ex Duplicati or GUI sync tools like CloudBerry, CyberDuck, S3 Browser, DragonDisk or Arq.

5. Access an S3 share (or other cloudservices) via rclone (CLI tool)

Rclone is in the OmniOS extra repo and installed with minIO in menu Services > S3.
You find the binaries in /opt/ooce/. To pass parameters to rclone, you can create a script where you pass parameters via export ex (use the parameter from your napp-it S3 share)
You can use any parameter from a rclone config file, just use it with uppercase char ex region -> REGION

```
#!/usr/bin/sh
export RCLONE_CONFIG_MYS3_TYPE=s3
export RCLONE_CONFIG_MYS3_ACCESS_KEY_ID=loginname
export RCLONE_CONFIG_MYS3_SECRET_ACCESS_KEY=secetpassword
export RCLONE_CONFIG_MYS3_ENDPOINT=https://s3.university.org:9000
export RCLONE_CONFIG_MYS3_REGION=my_location

#sync (a bucket (folder) bu1 must exist on your S3 share for this example
/opt/ooce/bin/rclone copy /anyfolder/ MYS3:/bu1

#list files in folder bu1
/opt/ooce/bin/rclone ls MYS3:/bu1
```

5. Config rclone for any Cloudservice

Start interactive setup for rclone, https://rclone.org/commands/rclone_config/
/opt/ooce/bin/rclone config and select and configure a Cloud service

Tipps
If you want to configure a remote where you want to copy/paste from console:
use Putty (as root) and copy/pasted from/to console with a mouse right click.

5.1 Config rclone for Amazon S3 compatible (minIO)

Create one or more remote shares with options. To access a napp-it S3 share, you need at least the parameters type, provider, access_key, secret_key, endpoint and region

When rclone is configured, you can get the location of the config file via
/opt/ooce/bin/rclone config file

example: config file: /root/.config/rclone/rclone.conf

Check config file via
cat /root/.config/rclone/rclone.conf

[cloud]
type = s3
provider = Minio
#env_auth = false
access_key_id = loginname
secret_access_key = secretpassword
endpoint = https://s3.university.org:9000
region = my_location

You can now use cloud: as source or desitination in rclone commands like copy or sync

ex enter at console:
/opt/ooce/bin/rclone ls cloud:

If you want to start rclone as a job, enter the rclone command as jobaction or call a script. You can enter several ongoing commands cmd1; cmd2; etc for a single line multi-command. If you want to keep a local filesystem in sync with a remote filesystem, use sync. This will delete remote files no longer in the local folder.

If you want to have newer files synced both ways, start a local copy -> Cloud followed by a copy Cloud > local with -u (newer files only). To delete files then you must delete locally and remote. You can not only sync napp-it ex to/from S3 or Google, you can also sync Google with Amazon S3 or compatible.

5.2 Config rclone for Google Gdrive

https://github.com/Cloudbox/Cloudbox/wiki/Google-Drive-API-Client-ID-and-Client-Secret

Login into Google (Chrome), Connect to https://console.developers.google.com/apis/
- create a project rclone
- enable Google Drive API
- create credidentials OAuth with addittional drive.file API
  (OaAuth Client ID and other or TV and limited Imput devices)
  This give the client id and client secret

Start rclone config
- create a new remote (Google drive) ex gdrive:
- enter client id and secret from above
- For verification, select headless mode
  this displays a verification link

- Copy/paste the verification link into a browser and allow access
  This will return a verification value
- Enter the verification value into rclone config and save

Now verify:
rclone ls gdrive

This should gave a listing of your Google drive content

6. Security

6.1 Encrypted transfer with https

To enable https for minIO, place a public and private key in /var/web-gui/_log/minio/
- private.key
- public.crt

Content of private.key
```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAt//67pLLarmyUldJet+YzlSJhl9w9DD8OK2A61NODIVztv3V
.....
iy6a+6g7uzR6/RISPWvvC05SXZXeMu/uMx7svtC9z88QfmVYWNDI5Q==
-----END RSA PRIVATE KEY-----
```

Content of public.crt
```
subject=/CN=s3.university.de/OU=it/O=University...
-----BEGIN CERTIFICATE-----
MIIEpAIBA....
```

6.2 Encrypt/ Decrypt files

Start minio config and create two entries:
s3cloud (the regular unencrypted cloud) and an encrypted device that points to the unencrypted remote and enc_s3: the device that does the encryptions and forward to s3cloud:

examples:
-name: s3cloud
- type: s3
- provider: Minio
- key: xxxx
- secret: yyyyyyyy
- endpoint: http(s)://xxx.yy:9000
- region: my_location (from s3 share settings)

and

- name: enc_s3 (use a name that identify the crypt property and the undelying remote)
- type of storage: 10 (enc/decrypt a remote)
- remote: s3:/bucket1 (s3: and s3:/bucket1 must exist)
- pw: 22222222 (enter twice)
- enc file and directoty names: optional
- salt (optional)

Save the password and the rclone.conf file with the hashed password and settings at a save place !!!

result: (rclone.conf) ex:

```
[s3cloud]
type = s3
provider = Minio
access_key_id = loginname
secret_access_key = wnkiPWs3#+
endpoint = https://s3.hfg-gmuend.de:9000
region = my_location

[enc_s3]
type = crypt
remote = s3cloud:/bu2
filename_encryption = standard
directory_name_encryption = true
password = HW8linj-vFk4XFXNF0v-ii2p1O0wSgr3
```

The above password is a hash value. Save the config file with your settings at a save place

You can now upload files to enc_s3:
This will save the files encrypted according the settings (a bucket ex bu2 must exist)
/opt/ooce/bin/rclone copy /folder/ enc_s3:

or download and decrypt:
/opt/ooce/bin/rclone copy /folder/ enc_s3: /folder/

Save your passwords and rclone config file on a save place. On Problems:
Restore the config file (or the needed remote entry) from backup

rclone config file	returns path to config file
rclone config show	returns content of config file

7. Cloudsync as a job (napp-it „other job")

Test a command at CLI or create a script with the wanted actions ex sync.sh:
#!/bin/sh
/opt/ooce/bin/rclone copy -u /pool/filesystem gdrive:/backup1

If you want to have new files locally and cloud (delete must be done manually both sides)
/opt/ooce/bin/rclone copy -u /pool/filesystem gdrive:/backup1
/opt/ooce/bin/rclone copy -u gdrive:/backup1 /pool/filesystem

If the action works as expected, create an other job with the command ex
/opt/ooce/bin/rclone copy -u /pool/filesystem gdrive:/backup1
or a script ex „sh /path/script.sh" as action.

8. Manuals and Questions

https://napp-it.org/manuals/index_en.html
https://forums.servethehome.com/index.php?threads/amazon-s3-compatible-zfs-cloud-with-minio.27524/
https://www.napp-it.org/doc/downloads/dreamteam.pdf