PRACTICAL PORTS

Samba-based Time Machine Backups

BY BENEDICT REUSCHLING

"I wish I'd saved the bandwidth for doing backups for something useful, as I won't ever need them" — said no one ever. In case of disasters — of which there are plenty, backups are an important part of IT resilience. There are failed drives, stolen laptops, broken firmware drivers that render data unreadable, and much more. Having backups and doing them on a regular basis is vital for continuing business operations. And testing them is just as important. But what happens if the backup solution is no longer supported and will not work on newer systems? How are new systems backed up and how do they integrate with the existing solution?

Having backups and doing them on a regular basis is vital for continuing business operations.

I wrote an article about setting up a FreeBSD Apple Time Machine in the March/April 2022 issue of this Journal. I ran that setup for a long time without issues--neither on the backup side nor that I had to restore from it. As time moved on, Apple changed the underlying Apple Filing Protocol (AFP) underneath. In newer versions of macOS (to which I diligently upgrade for security and feature enhancements), the protocol underwent some changes that made my original instructions not work anymore. As described above, its current setup still works, but for new time machine systems that I backup to, it won't work anymore. Apple integrated Server Message Block (SMB, better known as Samba) into the protocol in macOS 10.9 (Mavericks). The migration completed with macOS 11 (Big Sur), which removed the AFP server part and made SMB the new standard, which Time Machine uses internally.

I noticed this when I set up a new time machine backup system. Many moons ago, I had poured my instructions from the 2022 article into an Ansible playbook for easier deployment. The playbook still works, but results in newer macOS versions not being able to mount the exported drive as a storage location for Time Machine. Lucky me, I found that other people had had the same issue and had solved it. The instructions were not as cohesive has I'd hoped, though. The following setup combines different sources — reddit, per-

sonal blogs, the FreeBSD forums, and the Samba documentation. I have tested it on two different machines, fleshing out some instructions, and adding missing commands to ensure it works as intended. I kept the original "based on ZFS", because that's what I trust my important data to. You can leave this part out and run it on a different filesystem if you want. However, don't blame me if that does not work out so well!

Requirements

For this setup, you need a machine (FreeBSD in my case) for storing the backups. It should have a decent network connection and fast storage. The storage also needs to be redundant in some way, think RAID1 and levels above it. Capacity-wise, it depends on two factors: how many people will backup their data to it and how much data they have. The Time Machine configuration dialog allows you to set a disk quota and encryption, both of which are good options. ZFS employs quotas and reservations, so I set those on the filesystem level to the same value. Time Machine automatically removes older backups when the available storage space won't fit new backup data. The more storage you have, the longer the backup history that you can potentially restore from.

The encryption part is also nice to have, especially since we're sending the data over networks that may not be encrypted or are part of a VPN. On the receiving system, an encrypted ZFS dataset could be added for the data at rest. However, keep in mind that when the backup target needs to restart, the encrypted dataset is not mounted unless someone enters the passphrase for mounting the dataset. You could configure ZFS to get the passphrase from a file somewhere, but I leave that as an exercise to you in securing access to such a file that holds a passphrase in clear text.

On the sending side, any macOS system will be able to mount and configure the drive, protected by individual user credentials. This allows multiple people to backup up to the same Time Machine. Having beefy bandwidth for this server becomes clear when considering that concurrent backups might run at the same time. I learned that when configuring more than one time machine backup location on a Mac, the system is smart enough to not back up to both at the same time — preventing the system I/O from grinding to a halt doing lengthy backups to two (or even more) locations.

Configuring the Backup Server

A base installation of a FreeBSD release of choice (ideally still supported) with the latest security and errata patches installed goes without explicitly writing about it. We chose not to use a jail here, although nothing is stopping us from doing so. First, install the Samba package:

```
# pkg install samba419
```

Next, we configure our backup storage for ZFS. In my case, I have a dedicated pool aptly named backup and mounted at the same location (`/backup`). I create a separate dataset for Time Machine and set a quota and reservation, since I store other data on it and want to reserve some space for those files as well.

```
# zfs create -o quota=1.5T -o reservation=1.5T backup/timemachine
```

I know I will have two users (Tammy and Tim, Alice and Bob are on holiday) backing up their Macs to that destination. I value both of them equally, so both will get the same space reservations and quotas set. Remember that a quota and reservation on a dataset applies to

all datasets below it as well. The 1.5 TB will also apply to their datasets, limiting them already. But it's fine for them to run with 500GB each, so I set a **refquota** and **refreservation** to only apply to that individual dataset.

```
# zfs create -o refquota=500g -o refreservation=500g backup/timemachine/tammy
# zfs create -o refquota=500g -o refreservation=500g backup/timemachine/tim
```

The two of them will never be able to log into my backup server (they don't care anyway), but they still need to have a user on the system to mount the storage for Time Machine. I ran **adduser** for both of them, giving them no home directory (**/var/empty**) and also no shell access (**/usr/sbin/nologin**).

Run **chmod** and **chown** commands on their dataset mountpoints to protect those directories from prying eyes.

```
chmod -R 0700 /backup/timemachine/tammy
chmod -R 0700 /backup/timemachine/tim
chown -R tim /backup/timemachine/tim
chown -R tammy /backup/timemachine/tammy
```

The last thing to do is setting a password on the Samba side for those two users. This is the password needed for the prompt on macOS when mounting the time machine backup volume.

```
# smbpasswd -a tim
# smbpasswd -a tammy
```

Avahi Configuration

That's all for required software — straightforward and easy enough. Next we need to create two configuration files. One for the time machine service and the other one for the Samba configuration. The time machine service runs with Avahi — no not the lemurs from Madagascar, the software. This zeroconf network implementation enables programs to publish and discover services (like our time machine) in the local network. The configuration file is XML based, resides in **/usr/local/etc/avahi/services/timemachine.service** (create the file when it does not exist yet) and looks like this:

```
<?xml version="1.0" standalone='no'?>
<!DOCTYPE service-group SYSTEM "avahi-service.dtd">
<service-group>
 <name replace-wildcards="yes">%h</name>
 <service>
   <type>_smb._tcp</type>
   <port>445</port>
</service>
<service>
   <type>_device-info._tcp</type>
   <port>0</port>
   <txt-record>model=RackMac</txt-record>
</service>
<service>
   <type>_adisk._tcp</type>
   <txt-record>sys=waMa=0,adVF=0x100</txt-record>
```

```
    <txt-record>dk0=adVN=FreeBSD TimeMachine,adVF=0x82</txt-record>
  </service>
</service-group>
```

The file defines which ports to listen on for Samba services (445), how the icon looks for the mounted drive (**RackMac**) and what the display name for it should be (**FreeBSD Time-Machine**). It's fine to change the latter to give it a more descriptive name. I did not change any other parts of this file.

Samba Configuration

Samba is the open source implementation of the SMB protocol and has been around for 32 years. In the beginning, its main purpose was for compatibility between Unix systems and Windows. Active Directory integration, Domain Controller and other functionalities were added as Windows grew those appendixes. Since there is not a single Windows box involved in this setup (you can hear the sound of a sigh of relieve here), we use Samba's file sharing functionality as part of AFP here.

The samba configuration file is under **/usr/local/etc/smb4.conf** and contains this:

```
[global]
workgroup = WORKGROUP
security = user
passdb backend = tdbsam
fruit:aapl = yes
fruit:model = MacSamba
fruit:advertise_fullsync = true
fruit:metadata = stream
fruit:veto_appledouble = no
fruit:nfs_aces = no
fruit:wipe_intentionally_left_blank_rfork = yes
fruit:delete_empty_adfiles = yes

[TimeMachine]
path = /backup/timemachine/%U
valid users = %U
browseable = yes
writeable = yes
vfs objects = catia fruit streams_xattr zfsacl
fruit:time machine = yes
create mask = 0600
directory mask = 0700
```

Two sections (**global** and **TimeMachine**) define the necessary options for the backup destination. The lines prefixed with **fruit:** establish compatibility with macOS. Individual documentation for these lines is in the Samba documentation (see link in References at the end of the article). Change the path line in the **[TimeMachine]** section to the one created earlier on FreeBSD. The %U part is a placeholder for individual user names (tammy and tim in our case) backing up their files. That way, when adding another user later, we do not need to change this line at all. The create and directory masks ensure proper permissions so that the files don't get intermixed and users can't see or change each others backups.

Starting up

The remaining steps boil down to enabling and starting the **dbus** (avahi) and **samba** services.

```
# service dbus enable
# service dbus start
# service samba_server enable
# service samba_server start
```

On the macOS side (the backup clients), go to the finder and press CMD-K (Shortcut for "Connect to Server"). Enter **smb://server.ip.or.dns**. If all goes well, enter the username and password. This is the one for tim or tammy that we entered in the **smbpasswd** dialog earlier. If that succeeds, the share gets mounted into the system. Next, head to the time machine configuration dialog and add a new time machine volume. Make sure to visit the options button in there and check the box for the encrypted backup. You can only set this once before the initial backup and not afterward. You can also limit the amount of disk space the backups should consume, but that is optional, since we did it on the ZFS level already. Next, the long initial first backup can take place. After that finishes, time machine will backup up the Mac to this location by automatically mounting and unmounting the share when reachable on the network.

Summary

That's all. The samba configuration is easy enough to do and users should be able adapt it to their own needs. I found the solution just as reliable as the old one. I've adjusted my Ansible playbooks to use the new Samba-based setup. I still enjoy the fire-and-forget approach to backup up my Mac, knowing that I can pull individual files or the whole installation back when I need to, with the most current files that I had worked with.

References and Sources

- Samba Documentation
- Reddit Post
- FreeBSD Forums Post
- Dan Langille's blog

BENEDICT REUSCHLING is a documentation committer in the FreeBSD project and member of the documentation engineering team. In the past, he served on the FreeBSD core team for two terms. He administers a big data cluster at the University of Applied Sciences, Darmstadt, Germany. He's also teaching a course "Unix for Developers" for undergraduates. Benedict is one of the hosts of the weekly bsdnow.tv podcast.