Host Configuration

- Base Install
- Networking
- Storage & Backups (out-of-date)
- Storage & Backups

Base Install

Operating System

Proxmox Virtual Environment 6.x

Configuration

> Proxmox configuration has been transitioned to being automated by an Ansible Role

Networking

Bridges

Master	Bridge	IP Address	Gateway	Description
man0	-	10.0.3.2/32	-	Management Interface (slower Realtek NIC)
enp0s31f6	vmbr1	-	10.0.1.1	LAN/WAN (faster Intel GbE)
-	vmbr2	-	-	Private network without direct Internet access
enp0s31f6.8	vmbr3	-	-	DMZ (VLAN8 tagged)
enp0s31f6.9	vmbr4	-	-	WARP (VLAN9 tagged)

Storage & Backups (out-of-date)

Create `zpool10` Storage Pool

The host's `rpool` (default ZFS pool used by Proxmox) can be used for templates and whatnot but I want a big pool for storing all my data. So I created a ZFS RAIDZ2 pool using 6x 10TB HDDs.

This is the command I used to build my ZFS pool.

```
# zpool create -o ashift=12 -o failmode=continue -O compression=lz4 -O xattr=sa -O atime=off \
-O encryption=aes-256-gcm -O keyformat=passphrase \
-m /storage/zpool10 zpool10 raidz2 \
/dev/disk/by-id/ata-WDC_WD100EMAZ-00WJTA0_JEKH3DVZ \
/dev/disk/by-id/ata-WDC_WD100EMAZ-00WJTA0_2YK148SD \
/dev/disk/by-id/ata-WDC_WD100EMAZ-00WJTA0_JEKH8RWZ \
/dev/disk/by-id/ata-WDC_WD100EMAZ-00WJTA0_JEK6ESAN \
/dev/disk/by-id/ata-WDC_WD100EMAZ-00WJTA0_JEK53ZHN \
/dev/disk/by-id/ata-WDC_WD100EMAZ-00WJTA0_2YK0HL0D
```

- `ashift=12` to use 4k blocks
- `failmode=continue` to let us keep reading if a drive goes bad
- `compression=lz4` save save space and increase speed
- `xattr=sa` be more efficient for linux attributes
- `atime=off` don't waste time tracking access times for files
- `encryption=aes-256-gcm` use fast/secure encryption algorithm
- `keyformat=passphrase` unlock with a passphrase

Setup ZFS Scrub (Data Integrity)

Automate ZFS scrubbing so the data integrity on disks is actively monitored, repaired if necessary, and I'm alerted if there is a problem with my disks.

Create systemd Service/Timer (source)

Create a simple systemd service template for scrubbing ZFS pools.

```
# /etc/systemd/system/zpool-scrub@.service
+ [Unit]
+ Description=Scrub ZFS Pool
+ Requires=zfs.target
+ After=zfs.target
+
+ [Service]
+ Type=oneshot
+ ExecStartPre=-/usr/sbin/zpool scrub -s %I
+ ExecStart=/usr/sbin/zpool scrub %I
```

Then create a systemd timer template for periodically running that service. I am running the scrub weekly, but semi-monthly or monthly would almost certainly be ok too.

```
# /etc/systemd/system/zpool-scrub@.timer
+ [Unit]
+ Description=Scrub ZFS pool weekly
+
+ [Timer]
+ OnCalendar=weekly
+ Persistent=true
+
+ [Install]
+ WantedBy=timers.target
```

Enable ZFS Scrub

```
systemctl daemon-reload
systemctl enable --now zpool-scrub@rpool.timer
systemctl enable --now zpool-scrub@zpool10.timer
```

Setup Storage Layout

I wanted to logically device my storage pool up into datasets based on their intended usage. This allows me to tweak their paramters if needed and have different snapshot/backup policies.

- `zpool10/backups` place to store disk and time machine backups
- `zpool10/downloads` storage for downloads

- `zpool10/downloads/incomplete` landing zone for incomplete downloads (`recordsize=1M` for bittorrent)
- `zpool10/media` storage for audio/tv/movies
- `zpool10/proxmox` additional storage for proxmox
- `zpool10/proxmox/backups` backup for proxmox instances (in subdirectories by hostname)
- `zpool10/services` storage for services (possibly databases, so use `recordsize=16k`)

```
zfs create zpool10/backups
zfs create zpool10/downloads
zfs create -o recordsize=1M zpool10/downloads/incomplete
zfs create zpool10/media
zfs create zpool10/proxmox
zfs create zpool10/proxmox/backups
zfs create -o recordsize=16K zpool10/services
```

Setup Sanoid/Syncoid (Data Backup)

Run Sanoid for automating snapshots and Syncoid for remote backups. Unfortunately this isn't available in repositories so you have to build it yourself. However the author makes it fairly simple.

Install (source)

```
apt-get install debhelper libcapture-tiny-perl libconfig-inifiles-perl pv lzop mbuffer
sudo git clone https://github.com/jimsalterjrs/sanoid.git
cd sanoid
ln -s packages/debian .
dpkg-buildpackage -uc -us
apt install ../sanoid_*_all.deb
```

Configure Sanoid

I want to take hourly snapshots of both of my ZFS pools because sometimes I am not as careful or thoughtful as I should be about what I am doing at any given moment. But I don't want to snapshot `zpool/backups` because it is a backup destination that will likely already have snapshots (`rpool` snapshots are stored there for example) and I also don't want to snapshot `zpool/downloads` because there is nothing important under there and it is likely to change frquently.

```
# /etc/sanoid/sanoid.conf
+ [template_proxmox]
+         frequently = 0
```

```
+        hourly = 24
+        daily = 7
+        weekly = 4
+        monthly = 1
+        yearly = 0
+        autosnap = yes
+        autoprune = yes
+
+ [rpool]
+        use_template = template_proxmox
+        process_children_only = yes
+        recursive = yes
+
+ [rpool/ROOT]
+        use_template = rpool
+        process_children_only = yes
+        recursive = yes
+
+ [rpool/data]
+        use_template = template_proxmox
+        weekly = 1
+        monthly = 1
+        process_children_only = yes
+        recursive = yes
```

Maybe this is a sin, but I'd like my snapshots to be in local time so I don't have to do the (admittedly simple) conversion in my head.

```
# /usr/lib/systemd/system/sanoid.service
  [Service]
- Environment=TZ=UTC
+ Environment=TZ=EST
```

Eanble Sanoid

```
systemctl daemon-reload
systemctl enable --now sanoid.service
```

Configure Syncoid

Backup `rpool` to `zpool10/proxmox/backups/blackbox`

Right now `rpool` is just running on a single non-redundant 512GB SSD disk. Even though it is only used for Proxmox (config, templates, ISOs) this isn't great practice and I'll work on this in the future. But in the meantime I am backing up everything on a daily timer to my main ZFS pool so I could recover very quickly if the SSD dies.

```
# /etc/systemd/system/rpool-backup.timer
+ [Unit]
+ Description=Backup rpool daily
+
+ [Timer]
+ OnCalendar=daily
+ Persistent=true
+
+ [Install]
+ WantedBy=timers.target
```

```
# /etc/systemd/system/rpool-backup.service
+ [Unit]
+ Description=Use syncoid to backup rpool to zpool10/proxmox/backups/blackbox
+ Requires=zfs.target
+ After=zfs.target
+
+ [Service]
+ Type=oneshot
+ ExecStart=/usr/sbin/syncoid --force-delete --recursive rpool zpool10/proxmox/backups/blackbox/rpool
```

Backup `zpool10/services` offsite

All my docker containers store their configuration and data under the `zpool10/services` dataset. It is imperative this is backed up offsite so if anything catastrophic ever happens I don't lose anything important and can get back up and running as quickly as I can download my backup.

```
# /etc/systemd/system/zpool10-services-backup.timer
+ [Unit]
+ Description=Backup zpool10/services daily
+
+ [Timer]
+ OnCalendar=daily
+ Persistent=true
```

```
+
+ [Install]
+ WantedBy=timers.target
```

```
# /etc/systemd/system/zpool10-services-backup.service
+ [Unit]
+ Description=Use syncoid to backup zpool10/services to backedup.swigg.net:bpool/zpool10/services
+ Requires=zfs.target
+ After=zfs.target
+
+ [Service]
+ Type=oneshot
+ ExecStart=/usr/sbin/syncoid --force-delete --recursive zpool10/services
root@backedup.swigg.net:bpool/zpool10/services
```

Enable Syncoid

```
systemctl daemon-reload
systemctl enable --now rpool-backup.timer
```

Setup Restic to Backblaze B2 (Data Backup)

Read more about setting up Restic at https://fedoramagazine.org/automate-backups-with-restic-and-systemd/

Setup Restic Backup

Create the `.service` and `.timer` for the backup service so that it runs everyday.

```
# /etc/systemd/system/restic-backup.service
+ [Unit]
+ Description=Restic backup service
+
+ [Service]
+ Type=oneshot
+ ExecStart=restic backup --verbose --tag systemd.timer $BACKUP_EXCLUDES $BACKUP_INCLUDES
$BACKUP_PATHS
```

```
+ ExecStartPost=restic forget --verbose --tag systemd.timer --group-by "paths,tags" --keep-daily $RETENTION_DAYS --keep-weekly $RETENTION_WEEKS --keep-monthly $RETENTION_MONTHS --keep-yearly $RETENTION_YEARS
+ EnvironmentFile=/etc/systemd/system/restic-backup.service.d/restic-backup.conf
```

```
# /etc/systemd/system/restic-backup.timer
+ [Unit]
+ Description=Backup with restic daily
+
+ [Timer]
+ OnCalendar=daily
+ Persistent=true
+
+ [Install]
+ WantedBy=timers.target
```

```
# /etc/systemd/system/restic-backup.service.d/restic-backup.conf
+ BACKUP_PATHS="/storage/zpool10"
+ BACKUP_EXCLUDES="--exclude-file /etc/systemd/system/restic-backup.service.d/restic-excludes.txt --exclude-if-present .exclude_from_backup"
+ BACKUP_INCLUDES="--files-from /etc/systemd/system/restic-backup.service.d/restic-includes.txt"
+ RETENTION_DAYS=7
+ RETENTION_WEEKS=4
+ RETENTION_MONTHS=6
+ RETENTION_YEARS=3
+ B2_ACCOUNT_ID=xxx
+ B2_ACCOUNT_KEY=xxx
+ RESTIC_REPOSITORY=b2:swigg-backup-blackbox:storage/zpool10
+ RESTIC_PASSWORD=xxx
```

Setup Restic Prune

The backup command above forgets about files when they expire, but to actually delete them once they aren't referenced anymore you need to prune them. The following creates a `.service` and `.timer` for a prune job to be run every month.

```
# /etc/systemd/system/restic-prune.service
+ [Unit]
+ Description=Restic backup service (data pruning)
+
```

```
+ [Service]
+ Type=oneshot
+ ExecStart=restic prune
+ EnvironmentFile=/etc/systemd/system/restic-backup.service.d/restic-backup.conf
```

```
# /etc/systemd/system/restic-prune.timer
+ [Unit]
+ Description=Prune data from the restic repository monthly
+
+ [Timer]
+ OnCalendar=monthly
+ Persistent=true
+
+ [Install]
+ WantedBy=timers.target
```

Storage & Backups

Create Storage Pools

Although my current setup has changed over time I currently settled on the setup below.

Name	Disks	Type	Description
rpool	2x 500GB SSD	zpool mirror	Host OS, LXC Templates, ISOs
nocow	2x 1TB NVMe	LVM > LUKS > Linux Raid	LVM PVE Storage for LXCs/VMs, Non-COW disks/volumes
blackmirror	6x 10TB HDD + 4x 18TB HDD + 2x 1TB NVMe + 1x 16GB Optane	zpool + special + log	Primary PVE Storage for all LXCs/VMs

Creation Commands

blackmirror

```
zpool create -o ashift=12 -o failmode=continue -O atime=off -O relatime=on \
-O checksum=on -O compression=zstd -O encryption=aes-256-gcm -O keyformat=passphrase \
-O keylocation=prompt -O special_small_blocks=16K -O snapdir=hidden -O xattr=sa \
-m /storage/blackmirror blackmirror \
mirror \
  /dev/disk/by-id/ata-WDC_WD181KFGX-68AFPN0_4BK0RVSZ \
  /dev/disk/by-id/ata-WDC_WD181KFGX-68AFPN0_4YGUS20H \
mirror \
  /dev/disk/by-id/ata-WDC_WD181KFGX-68AFPN0_4YGUS4AH \
  /dev/disk/by-id/ata-WDC_WD181KFGX-68AFPN0_5DJ20PAV \
mirror \
  /dev/disk/by-id/ata-WDC_WD100EMAZ-00WJTA0_2YK0HL0D \
  /dev/disk/by-id/ata-WDC_WD100EMAZ-00WJTA0_JEK53ZHN \
mirror \
  /dev/disk/by-id/ata-WDC_WD100EMAZ-00WJTA0_2YK148SD \
```

```
  /dev/disk/by-id/ata-WDC_WD100EMAZ-00WJTA0_JEK6ESAN \
mirror \
  /dev/disk/by-id/ata-WDC_WD100EMAZ-00WJTA0_JEKH3DVZ \
  /dev/disk/by-id/ata-WDC_WD100EMAZ-00WJTA0_JEKH8RWZ \
special mirror \
  /dev/disk/by-id/nvme-PC401_NVMe_SK_hynix_1TB_MN8BN16291080BN2A \
  /dev/disk/by-id/nvme-PC401_NVMe_SK_hynix_1TB_MN8BN16291080BN46 \
log \
  /dev/disk/by-id/nvme-INTEL_MEMPEK1W016GA_PHBT72350AVD016D
```

- `-o ashift=12` zpool property: use 4k blocks
- `-o failmode=continue` zpool property: keep reading if a drive goes bad in pool
- `-O atime=off` root filesystem: disable updatnig access time for files when they are read
- `-O relatime=on` root filesystem: enable relatime (relative access time)
- `-O checksum=on` root filesystem: enable checksum used to verify data integrity
- `-O compression=zstd` root filesystem: use zstd compression
- `-O encryption=aes-256-gcm` root filesystem: use fast/secure encryption algorithm
- `-O keyformat=passphrase` root filesystem: unlock with a passphrase
- `-O keylocation=prompt` root filesystem: prompt for key
- `-O special_small_blocks=16K` root filesystem: 16k threshold block size for including small file blocks into the special allocation class
- `-O snapdir=hidden` root filesystem: hide .zfs snapshots directory
- `-O xattr=sa` root filesystem: Enable system attributes for extended attributes
- `-m /storage/blackmirror` root filesystem: mount point for the root dataset

nocow

```
tbd
```

- `attr` description

Setup ZFS Scrub (Data Integrity)

Automate ZFS scrubbing so the data integrity on disks is actively monitored, repaired if necessary, and I'm alerted if there is a problem with my disks.

Create systemd Service/Timer (source)

Create a simple systemd service template for scrubbing ZFS pools.

```
# /etc/systemd/system/zpool-scrub@.service
+ [Unit]
```

```
+ Description=Scrub ZFS Pool
+ Requires=zfs.target
+ After=zfs.target
+
+ [Service]
+ Type=oneshot
+ ExecStartPre=-/usr/sbin/zpool scrub -s %I
+ ExecStart=/usr/sbin/zpool scrub %I
```

Then create a systemd timer template for periodically running that service. I am running the scrub weekly, but semi-monthly or monthly would almost certainly be ok too.

```
# /etc/systemd/system/zpool-scrub@.timer
+ [Unit]
+ Description=Scrub ZFS pool weekly
+
+ [Timer]
+ OnCalendar=weekly
+ Persistent=true
+
+ [Install]
+ WantedBy=timers.target
```

Enable ZFS Scrub

```
systemctl daemon-reload
systemctl enable --now zpool-scrub@rpool.timer
systemctl enable --now zpool-scrub@blackmirror.timer
```

Setup Storage Layout

I wanted to logically device my storage pool up into datasets based on their intended usage. This allows me to tweak their paramters if needed and have different snapshot/backup policies.

- `zpool10/backups` place to store disk and time machine backups
- `zpool10/downloads` storage for downloads
- `zpool10/downloads/incomplete` landing zone for incomplete downloads (`recordsize=1M` for bittorrent)
- `zpool10/media` storage for audio/tv/movies
- `zpool10/proxmox` additional storage for proxmox
- `zpool10/proxmox/backups` backup for proxmox instances (in subdirectories by hostname)

- `zpool10/services` storage for services (possibly databases, so use `recordsize=16k`)

```
zfs create zpool10/backups
zfs create zpool10/downloads
zfs create -o recordsize=16K zpool10/downloads/incomplete
zfs create zpool10/media
zfs create zpool10/proxmox
zfs create zpool10/proxmox/backups
zfs create -o recordsize=16K zpool10/services
```

Setup Sanoid/Syncoid (Data Backup)

Run Sanoid for automating snapshots and Syncoid for remote backups. Unfortunately this isn't available in repositories so you have to build it yourself. However the author makes it fairly simple.

Install (source)

```
apt-get install debhelper libcapture-tiny-perl libconfig-inifiles-perl pv lzop mbuffer
sudo git clone https://github.com/jimsalterjrs/sanoid.git
cd sanoid
ln -s packages/debian .
dpkg-buildpackage -uc -us
apt install ../sanoid_*_all.deb
```

Configure Sanoid

I want to take hourly snapshots of both of my ZFS pools because sometimes I am not as careful or thoughtful as I should be about what I am doing at any given moment. But I don't want to snapshot `zpool/backups` because it is a backup destination that will likely already have snapshots (`rpool` snapshots are stored there for example) and I also don't want to snapshot `zpool/downloads` because there is nothing important under there and it is likely to change frquently.

```
# /etc/sanoid/sanoid.conf
+ [template_proxmox]
+        frequently = 0
+        hourly = 24
+        daily = 7
+        weekly = 4
+        monthly = 1
+        yearly = 0
```

```
+        autosnap = yes
+        autoprune = yes
+
+ [rpool]
+        use_template = template_proxmox
+        process_children_only = yes
+        recursive = yes
+
+ [rpool/ROOT]
+        use_template = rpool
+        process_children_only = yes
+        recursive = yes
+
+ [rpool/data]
+        use_template = template_proxmox
+        weekly = 1
+        monthly = 1
+        process_children_only = yes
+        recursive = yes
```

Maybe this is a sin, but I'd like my snapshots to be in local time so I don't have to do the (admittedly simple) conversion in my head.

```
# /usr/lib/systemd/system/sanoid.service
  [Service]
- Environment=TZ=UTC
+ Environment=TZ=EST
```

Eanble Sanoid

```
systemctl daemon-reload
systemctl enable --now sanoid.service
```

Configure Syncoid

Backup `rpool` to `zpool10/proxmox/backups/blackbox`

Right now `rpool` is just running on a single non-redundant 512GB SSD disk. Even though it is only used for Proxmox (config, templates, ISOs) this isn't great practice and I'll work on this in the future. But in the meantime I am backing up everything on a daily timer to my main ZFS pool so I could recover very quickly if the SSD dies.

```
# /etc/systemd/system/rpool-backup.timer
+ [Unit]
+ Description=Backup rpool daily
+
+ [Timer]
+ OnCalendar=daily
+ Persistent=true
+
+ [Install]
+ WantedBy=timers.target
```

```
# /etc/systemd/system/rpool-backup.service
+ [Unit]
+ Description=Use syncoid to backup rpool to zpool10/proxmox/backups/blackbox
+ Requires=zfs.target
+ After=zfs.target
+
+ [Service]
+ Type=oneshot
+ ExecStart=/usr/sbin/syncoid --force-delete --recursive rpool zpool10/proxmox/backups/blackbox/rpool
```

Backup `zpool10/services` offsite

All my docker containers store their configuration and data under the `zpool10/services` dataset. It is imperative this is backed up offsite so if anything catastrophic ever happens I don't lose anything important and can get back up and running as quickly as I can download my backup.

```
# /etc/systemd/system/zpool10-services-backup.timer
+ [Unit]
+ Description=Backup zpool10/services daily
+
+ [Timer]
+ OnCalendar=daily
+ Persistent=true
+
+ [Install]
+ WantedBy=timers.target
```

```
# /etc/systemd/system/zpool10-services-backup.service
+ [Unit]
```

```
+ Description=Use syncoid to backup zpool10/services to backedup.swigg.net:bpool/zpool10/services
+ Requires=zfs.target
+ After=zfs.target
+
+ [Service]
+ Type=oneshot
+ ExecStart=/usr/sbin/syncoid --force-delete --recursive zpool10/services
root@backedup.swigg.net:bpool/zpool10/services
```

Enable Syncoid

```
systemctl daemon-reload
systemctl enable --now rpool-backup.timer
```

Setup Restic to Backblaze B2 (Data Backup)

Read more about setting up Restic at https://fedoramagazine.org/automate-backups-with-restic-and-systemd/

Setup Restic Backup

Create the `.service` and `.timer` for the backup service so that it runs everyday.

```
# /etc/systemd/system/restic-backup.service
+ [Unit]
+ Description=Restic backup service
+
+ [Service]
+ Type=oneshot
+ ExecStart=restic backup --verbose --tag systemd.timer $BACKUP_EXCLUDES $BACKUP_INCLUDES
$BACKUP_PATHS
+ ExecStartPost=restic forget --verbose --tag systemd.timer --group-by "paths,tags" --keep-daily
$RETENTION_DAYS --keep-weekly $RETENTION_WEEKS --keep-monthly $RETENTION_MONTHS --keep-yearly
$RETENTION_YEARS
+ EnvironmentFile=/etc/systemd/system/restic-backup.service.d/restic-backup.conf
```

```
# /etc/systemd/system/restic-backup.timer
+ [Unit]
+ Description=Backup with restic daily
+
+ [Timer]
+ OnCalendar=daily
+ Persistent=true
+
+ [Install]
+ WantedBy=timers.target
```

```
# /etc/systemd/system/restic-backup.service.d/restic-backup.conf
+ BACKUP_PATHS="/storage/zpool10"
+ BACKUP_EXCLUDES="--exclude-file /etc/systemd/system/restic-backup.service.d/restic-excludes.txt --exclude-if-present .exclude_from_backup"
+ BACKUP_INCLUDES="--files-from /etc/systemd/system/restic-backup.service.d/restic-includes.txt"
+ RETENTION_DAYS=7
+ RETENTION_WEEKS=4
+ RETENTION_MONTHS=6
+ RETENTION_YEARS=3
+ B2_ACCOUNT_ID=xxx
+ B2_ACCOUNT_KEY=xxx
+ RESTIC_REPOSITORY=b2:swigg-backup-blackbox:storage/zpool10
+ RESTIC_PASSWORD=xxx
```

Setup Restic Prune

The backup command above forgets about files when they expire, but to actually delete them once they aren't referenced anymore you need to prune them. The following creates a `.service` and `.timer` for a prune job to be run every month.

```
# /etc/systemd/system/restic-prune.service
+ [Unit]
+ Description=Restic backup service (data pruning)
+
+ [Service]
+ Type=oneshot
+ ExecStart=restic prune
+ EnvironmentFile=/etc/systemd/system/restic-backup.service.d/restic-backup.conf
```

```
# /etc/systemd/system/restic-prune.timer
+ [Unit]
+ Description=Prune data from the restic repository monthly
+
+ [Timer]
+ OnCalendar=monthly
+ Persistent=true
+
+ [Install]
+ WantedBy=timers.target
```