Proxmox Backup Documentation

Release 1.0.12-1

Proxmox Support Team

Friday, 26 March 2021

TABLE OF CONTENTS

Version 1.0.12 – Friday, 26 March 2021

CHAPTER

ONE

INTRODUCTION

1.1 What is Proxmox Backup Server?

Proxmox Backup Server is an enterprise-class, client-server backup software package that backs up *virtual machine*s, *container*s, and physical hosts. It is specially optimized for the Proxmox Virtual Environment platform and allows you to back up your data securely, even between remote sites, providing easy management with a web-based user interface.

It supports deduplication, compression, and authenticated encryption (AE). Using *Rust* as the implementation language guarantees high performance, low resource usage, and a safe, high-quality codebase.

Proxmox Backup uses state of the art cryptography for both client-server communication and backup content *encryption*. All client-server communication uses TLS, and backup data can be encrypted on the client-side before sending, making it safer to back up data to targets that are not fully trusted.

1.2 Architecture

Proxmox Backup Server uses a client-server model. The server stores the backup data and provides an API to create and manage datastores. With the API, it's also possible to manage disks and other server-side resources.

The backup client uses this API to access the backed up data. With the command line tool `proxmox-backup-client` you can create backups and restore data. For QEMU with Proxmox Virtual Environment we deliver an integrated client.

A single backup is allowed to contain several archives. For example, when you backup a *virtual machine*, each disk is stored as a separate archive inside that backup. The VM configuration itself is stored as an extra file. This way, it's easy to access and restore only important parts of the backup, without the need to scan the whole backup.

1.3 Main Features

Support for Proxmox VE The Proxmox Virtual Environment is fully supported and you can easily backup *virtual machine*s and *container*s.

Performance The whole software stack is written in *Rust*, in order to provide high speed and memory efficiency.

Deduplication Periodic backups produce large amounts of duplicate data. The deduplication layer avoids redundancy and minimizes the storage space used.

Incremental backups Changes between backups are typically low. Reading and sending only the delta reduces the storage and network impact of backups.

Data Integrity The built-in SHA-256 checksum algorithm ensures accuracy and consistency in your backups.

Remote Sync It is possible to efficiently synchronize data to remote sites. Only deltas containing new data are transferred.

Compression The ultra-fast Zstandard compression is able to compress several gigabytes of data per second.

Encryption Backups can be encrypted on the client-side, using AES-256 GCM. This authenticated encryption (AE) mode provides very high performance on modern hardware. In addition to client-side encryption, all data is transferred via a secure TLS connection.

Web interface Manage the Proxmox Backup Server with the integrated, web-based user interface.

Open Source No secrets. Proxmox Backup Server is free and open-source software. The source code is licensed under AGPL, v3.

No Limits Proxmox Backup Server has no artifical limits for backup storage or backup-clients.

Enterprise Support Proxmox Server Solutions GmbH offers enterprise support in form of Proxmox Backup Server Subscription Plans. Users at every subscription level get access to the Proxmox Backup Enterprise Repository. In addition, with a Basic, Standard or Premium subscription, users have access to the *Proxmox Customer Portal*.

1.4 Reasons for Data Backup?

The main purpose of a backup is to protect against data loss. Data loss can be caused by both faulty hardware and human error.

A common mistake is to accidentally delete a file or folder which is still required. Virtualization can even amplify this problem, as deleting a whole virtual machine can be as easy as pressing a single button.

For administrators, backups can serve as a useful toolkit for temporarily storing data. For example, it is common practice to perform full backups before installing major software updates. If something goes wrong, you can easily restore the previous state.

Another reason for backups are legal requirements. Some data, especially business records, must be kept in a safe place for several years by law, so that they can be accessed if required.

In general, data loss is very costly as it can severely damage your business. Therefore, ensure that you perform regular backups and run restore tests.

1.5 Software Stack

Proxmox Backup Server consists of multiple components:

- A server-daemon providing, among other things, a RESTful API, super-fast asynchronous tasks, lightweight usage statistic collection, scheduling events, strict separation of privileged and unprivileged execution environments
- A JavaScript management web interface
- A management CLI tool for the server (*proxmox-backup-manager*)

- A client CLI tool (*proxmox-backup-client*) to access the server easily from any *Linux amd64* environment

Aside from the web interface, most parts of Proxmox Backup Server are written in the Rust programming language.

> "The Rust programming language helps you write faster, more reliable software. High-level ergonomics and low-level control are often at odds in programming language design; Rust challenges that conflict. Through balancing powerful technical capacity and a great developer experience, Rust gives you the option to control low-level details (such as memory usage) without all the hassle traditionally associated with such control."
>
> —The Rust Programming Language

1.6 Getting Help

1.6.1 Enterprise Support

Users with a Proxmox Backup Server Basic, Standard or Premium Subscription Plan have access to the Proxmox Customer Portal. The Customer Portal provides support with guaranteed response times from the Proxmox developers. For more information or for volume discounts, please contact office@proxmox.com.

1.6.2 Community Support Forum

We always encourage our users to discuss and share their knowledge using the Proxmox Community Forum. The forum is moderated by the Proxmox support team. The large user base is spread out all over the world. Needless to say that such a large forum is a great place to get information.

1.6.3 Mailing Lists

Proxmox Backup Server is fully open-source and contributions are welcome! Here is the primary communication channel for developers:

Mailing list for developers PBS Development List

1.6.4 Bug Tracker

Proxmox runs a public bug tracker at https://bugzilla.proxmox.com. If an issue appears, file your report there. An issue can be a bug as well as a request for a new feature or enhancement. The bug tracker helps to keep track of the issue and will send a notification once it has been solved.

1.7 License

This program is distributed in the hope that it will be useful, but `WITHOUT ANY WARRANTY`; without even the implied warranty of `MERCHANTABILITY` or `FITNESS FOR A PARTICULAR PURPOSE`. See the GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License along with this program. If not, see AGPL3.

1.8 History

Backup is, and always has been, a central aspect of IT administration. The need to recover from data loss is fundamental and only increases with virtualization.

For this reason, we've been shipping a backup tool with Proxmox VE, from the beginning. This tool is called `vzdump` and is able to make consistent snapshots of running LXC containers and KVM virtual machines.

However, `vzdump` only allows for full backups. While this is fine for small backups, it becomes a burden for users with large VMs. Both backup duration and storage usage are too high for this case, especially for users who want to keep many backups of the same VMs. To solve these problems, we needed to offer deduplication and incremental backups.

Back in October 2018, development started. We investigated several technologies and frameworks and finally decided to use *Rust* as the implementation language, in order to provide high speed and memory efficiency. The 2018-edition of Rust seemed promising for our requirements.

In July 2020, we released the first beta version of Proxmox Backup Server, followed by the first stable version in November 2020. With support for incremental, fully deduplicated backups, Proxmox Backup significantly reduces network load and saves valuable storage space.

CHAPTER

TWO

INSTALLATION

Proxmox Backup is split into a server and client part. The server part can either be installed with a graphical installer or on top of Debian from the provided package repository.

2.1 System Requirements

We recommend using high quality server hardware when running Proxmox Backup in production. To further decrease the impact of a failed host, you can set up periodic, efficient, incremental *datastore synchronization* from other Proxmox Backup Server instances.

2.1.1 Minimum Server Requirements, for Evaluation

These minimum requirements are for evaluation purposes only and should not be used in production.

- CPU: 64bit (*x86-64* or *AMD64*), 2+ Cores
- Memory (RAM): 2 GB RAM
- Hard drive: more than 8GB of space.
- Network card (NIC)

2.1.2 Recommended Server System Requirements

- CPU: Modern AMD or Intel 64-bit based CPU, with at least 4 cores
- Memory: minimum 4 GiB for the OS, filesystem cache and Proxmox Backup Server daemons. Add at least another GiB per TiB storage space.
- OS storage:
  - 32 GiB, or more, free storage space
  - Use a hardware RAID with battery protected write cache (*BBU*) or a redundant ZFS setup (ZFS is not compatible with a hardware RAID controller).
- Backup storage:
  - Use only SSDs, for best results
  - If HDDs are used: Using a metadata cache is highly recommended, for example, add a ZFS *special device mirror*.
- Redundant Multi-GBit/s network interface cards (NICs)

2.1.3 Supported Web Browsers for Accessing the Web Interface

To access the server's web-based user interface, we recommend using one of the following browsers:

- Firefox, a release from the current year, or the latest Extended Support Release
- Chrome, a release from the current year
- Microsoft's currently supported version of Edge
- Safari, a release from the current year

2.2 Debian Package Repositories

All Debian based systems use APT as a package management tool. The lists of repositories are defined in `/etc/apt/sources.list` and the `.list` files found in the `/etc/apt/sources.d/` directory. Updates can be installed directly with the `apt` command line tool, or via the GUI.

APT `sources.list` files list one package repository per line, with the most preferred source listed first. Empty lines are ignored and a `#` character anywhere on a line marks the remainder of that line as a comment. The information available from the configured sources is acquired by `apt update`.

Listing 1: File: `/etc/apt/sources.list`

```
deb http://ftp.debian.org/debian buster main contrib
deb http://ftp.debian.org/debian buster-updates main contrib

# security updates
deb http://security.debian.org/debian-security buster/updates main contrib
```

In addition, you need a package repository from Proxmox to get Proxmox Backup updates.

2.2.1 SecureApt

The *Release* files in the repositories are signed with GnuPG. APT is using these signatures to verify that all packages are from a trusted source.

If you install Proxmox Backup Server from an official ISO image, the verification key is already installed.

If you install Proxmox Backup Server on top of Debian, download and install the key with the following commands:

```
# wget http://download.proxmox.com/debian/proxmox-ve-release-6.x.gpg -O /etc/apt/trusted.gpg.
↪d/proxmox-ve-release-6.x.gpg
```

Verify the SHA512 checksum afterwards with:

```
# sha512sum /etc/apt/trusted.gpg.d/proxmox-ve-release-6.x.gpg
```

The output should be:

```
acca6f416917e8e11490a08a1e2842d500b3a5d9f322c6319db0927b2901c3eae23cfb5cd5df6facf2b57399d3cfa52ad7769ebdd75d9
↪/etc/apt/trusted.gpg.d/proxmox-ve-release-6.x.gpg
```

and the md5sum:

```
# md5sum /etc/apt/trusted.gpg.d/proxmox-ve-release-6.x.gpg
```

Here, the output should be:

```
f3f6c5a3a67baf38ad178e5ff1ee270c /etc/apt/trusted.gpg.d/proxmox-ve-release-6.x.gpg
```

2.2.2 Proxmox Backup Enterprise Repository

This is the stable, recommended repository. It is available for all Proxmox Backup subscription users. It contains the most stable packages, and is suitable for production use. The `pbs-enterprise` repository is enabled by default:

Listing 2: File: `/etc/apt/sources.list.d/pbs-enterprise.list`

```
deb https://enterprise.proxmox.com/debian/pbs buster pbs-enterprise
```

To never miss important security fixes, the superuser (`root@pam` user) is notified via email about new packages as soon as they are available. The change-log and details of each package can be viewed in the GUI (if available).

Please note that you need a valid subscription key to access this repository. More information regarding subscription levels and pricing can be found at https://www.proxmox.com/en/proxmox-backup-server/pricing

Note: You can disable this repository by commenting out the above line using a # (at the start of the line). This prevents error messages if you do not have a subscription key. Please configure the `pbs-no-subscription` repository in that case.

2.2.3 Proxmox Backup No-Subscription Repository

As the name suggests, you do not need a subscription key to access this repository. It can be used for testing and non-production use. It is not recommended to use it on production servers, because these packages are not always heavily tested and validated.

We recommend to configure this repository in `/etc/apt/sources.list`.

Listing 3: File: `/etc/apt/sources.list`

```
deb http://ftp.debian.org/debian buster main contrib
deb http://ftp.debian.org/debian buster-updates main contrib

# PBS pbs-no-subscription repository provided by proxmox.com,
# NOT recommended for production use
deb http://download.proxmox.com/debian/pbs buster pbs-no-subscription

# security updates
deb http://security.debian.org/debian-security buster/updates main contrib
```

2.2.4 Proxmox Backup Test Repository

This repository contains the latest packages and is heavily used by developers to test new features.

You can access this repository by adding the following line to `/etc/apt/sources.list`:

Listing 4: sources.list entry for `pbstest`

```
deb http://download.proxmox.com/debian/pbs buster pbstest
```

2.3 Server Installation

The backup server stores the actual backed up data and provides a web based GUI for various management tasks such as disk management.

Note: You always need a backup server. It is not possible to use Proxmox Backup without the server part.

The disk image (ISO file) provided by Proxmox includes a complete Debian system ("buster" for version 1.x) as well as all necessary packages for the Proxmox Backup server.

The installer will guide you through the setup process and allow you to partition the local disk(s), apply basic system configurations (e.g. timezone, language, network), and install all required packages. The provided ISO will get you started in just a few minutes, and is the recommended method for new and existing users.

Alternatively, Proxmox Backup server can be installed on top of an existing Debian system.

2.3.1 Install Proxmox Backup with the Installer

Download the ISO from https://www.proxmox.com/downloads. It includes the following:

- The Proxmox Backup server installer, which partitions the local disk(s) with ext4, xfs or ZFS, and installs the operating system
- Complete operating system (Debian Linux, 64-bit)
- Proxmox Linux kernel with ZFS support
- Complete tool-set to administer backups and all necessary resources
- Web based management interface

Note: During the installation process, the complete server is used by default and all existing data is removed.

2.3.2 Install Proxmox Backup Server on Debian

Proxmox ships as a set of Debian packages which can be installed on top of a standard Debian installation. After configuring the sysadmin_package_repositories, you need to run:

```
# apt-get update
# apt-get install proxmox-backup-server
```

The commands above keep the current (Debian) kernel and install a minimal set of required packages.

If you want to install the same set of packages as the installer does, please use the following:

```
# apt-get update
# apt-get install proxmox-backup
```

This will install all required packages, the Proxmox kernel with ZFS support, and a set of common and useful packages.

Caution: Installing Proxmox Backup on top of an existing Debian installation looks easy, but it assumes that the base system and local storage have been set up correctly. In general this is not trivial, especially when LVM or ZFS is used. The network configuration is completely up to you as well.

Note: You can access the web interface of the Proxmox Backup Server with your web browser, using HTTPS on port 8007. For example at `https://<ip-or-dns-name>:8007`

2.3.3 Install Proxmox Backup Server on Proxmox VE

After configuring the sysadmin_package_repositories, you need to run:

```
# apt-get update
# apt-get install proxmox-backup-server
```

Caution: Installing the backup server directly on the hypervisor is not recommended. It is safer to use a separate physical server to store backups. Should the hypervisor server fail, you can still access the backups.

Note: You can access the web interface of the Proxmox Backup Server with your web browser, using HTTPS on port 8007. For example at `https://<ip-or-dns-name>:8007`

2.4 Client Installation

2.4.1 Install Proxmox Backup Client on Debian

Proxmox ships as a set of Debian packages to be installed on top of a standard Debian installation. After configuring the sysadmin_package_repositories, you need to run:

```
# apt-get update
# apt-get install proxmox-backup-client
```

2.4.2 Installing from source

2.4.3 Installing statically linked binary

CHAPTER

THREE

TERMINOLOGY

3.1 Backup Content

When doing deduplication, there are different strategies to get optimal results in terms of performance and/or deduplication rates. Depending on the type of data, it can be split into *fixed* or *variable* sized chunks.

Fixed sized chunking requires minimal CPU power, and is used to backup virtual machine images.

Variable sized chunking needs more CPU power, but is essential to get good deduplication rates for file archives.

The Proxmox Backup Server supports both strategies.

3.1.1 Image Archives: `<name>.img`

This is used for virtual machine images and other large binary data. Content is split into fixed-sized chunks.

3.1.2 File Archives: `<name>.pxar`

A file archive stores a full directory tree. Content is stored using the *Proxmox File Archive Format (.pxar)*, split into variable-sized chunks. The format is optimized to achieve good deduplication rates.

3.1.3 Binary Data (BLOBs)

This type is used to store smaller (< 16MB) binary data such as configuration files. Larger files should be stored as image archive.

Caution: Please do not store all files as BLOBs. Instead, use the file archive to store whole directory trees.

3.1.4 Catalog File: `catalog.pcat1`

The catalog file is an index for file archives. It contains the list of files and is used to speed up search operations.

3.1.5 The Manifest: `index.json`

The manifest contains the list of all backup files, their sizes and checksums. It is used to verify the consistency of a backup.

3.2 Backup Type

The backup server groups backups by *type*, where *type* is one of:

`vm` This type is used for *virtual machine*s. Typically consists of the virtual machine's configuration file and an image archive for each disk.

`ct` This type is used for *container*s. Consists of the container's configuration and a single file archive for the filesystem content.

`host` This type is used for backups created from within the backed up machine. Typically this would be a physical host but could also be a virtual machine or container. Such backups may contain file and image archives, there are no restrictions in this regard.

3.3 Backup ID

A unique ID. Usually the virtual machine or container ID. `host` type backups normally use the hostname.

3.4 Backup Time

The time when the backup was made.

3.5 Backup Group

The tuple `<type>/<ID>` is called a backup group. Such a group may contain one or more backup snapshots.

3.6 Backup Snapshot

The triplet `<type>/<ID>/<time>` is called a backup snapshot. It uniquely identifies a specific backup within a datastore.

Listing 1: Backup Snapshot Examples

```
vm/104/2019-10-09T08:01:06Z
host/elsa/2019-11-08T09:48:14Z
```

As you can see, the time format is RFC3399 with Coordinated Universal Time (UTC, identified by the trailing *Z*).

CHAPTER

FOUR

GRAPHICAL USER INTERFACE

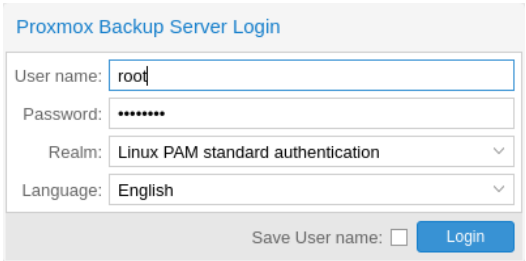
Proxmox Backup Server offers an integrated, web-based interface to manage the server. This means that you can carry out all administration tasks through your web browser, and that you don't have to worry about installing extra management tools. The web interface also provides a built-in console, so if you prefer the command line or need some extra control, you have this option.

The web interface can be accessed via https://youripaddress:8007. The default login is *root*, and the password is the one specified during the installation process.

4.1 Features

- Simple management interface for Proxmox Backup Server
- Monitoring of tasks, logs and resource usage
- Management of users, permissions, datastores, etc.
- Secure HTML5 console
- Support for multiple authentication sources
- Support for multiple languages
- Based on ExtJS 6.x JavaScript framework

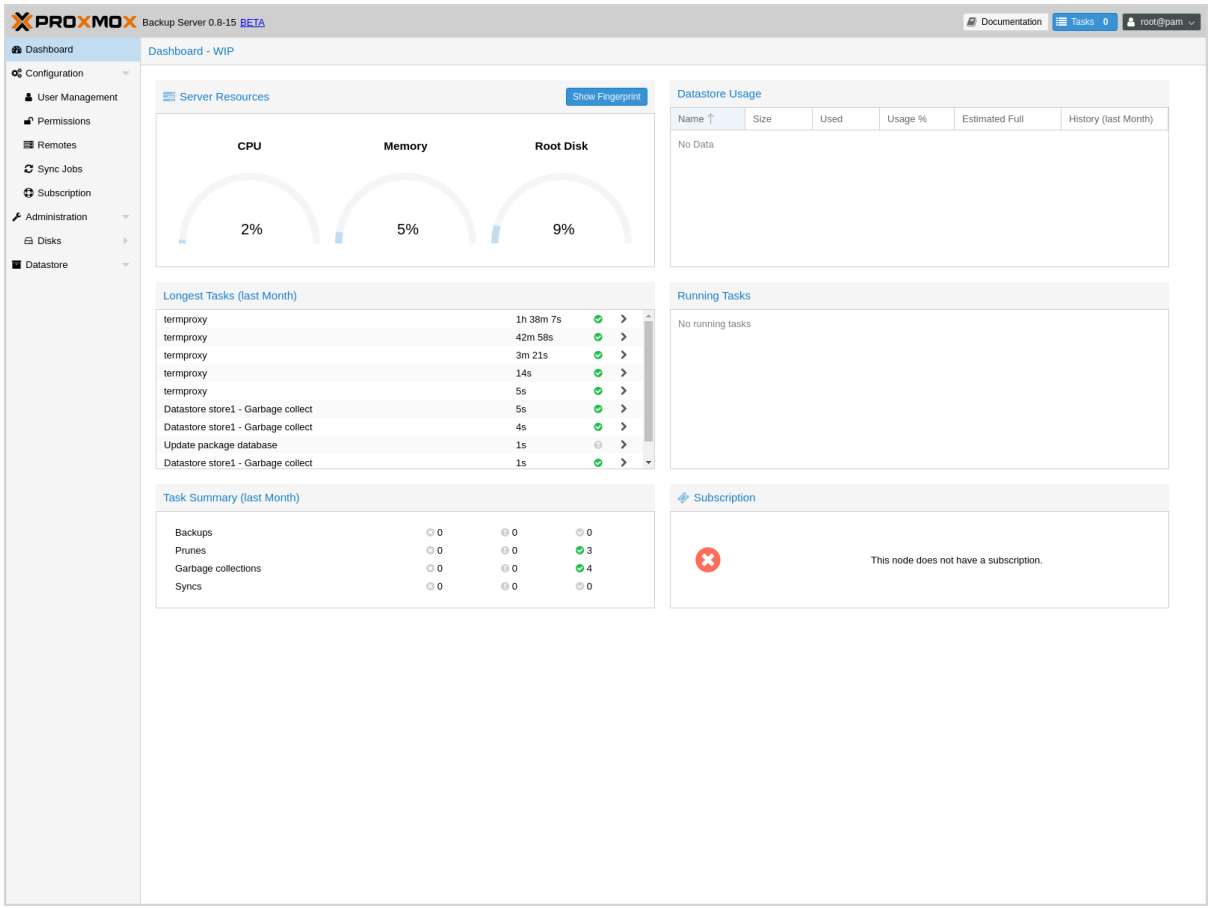
4.2 Login



When you connect to the web interface, you will first see the login window. Proxmox Backup Server supports various languages and authentication back ends (*Realms*), both of which can be selected here.

Note: For convenience, you can save the username on the client side, by selecting the "Save User name" checkbox at the bottom of the window.

4.3 GUI Overview



The Proxmox Backup Server web interface consists of 3 main sections:

- **Header**: At the top. This shows version information, and contains buttons to view documentation, monitor running tasks, set the language and logout.
- **Sidebar**: On the left. This contains the configuration options for the server.
- **Configuration Panel**: In the center. This contains the control interface for the configuration options in the *Sidebar*.

4.4 Sidebar

In the sidebar, on the left side of the page, you can see various items relating to specific management activities.

4.4.1 Dashboard

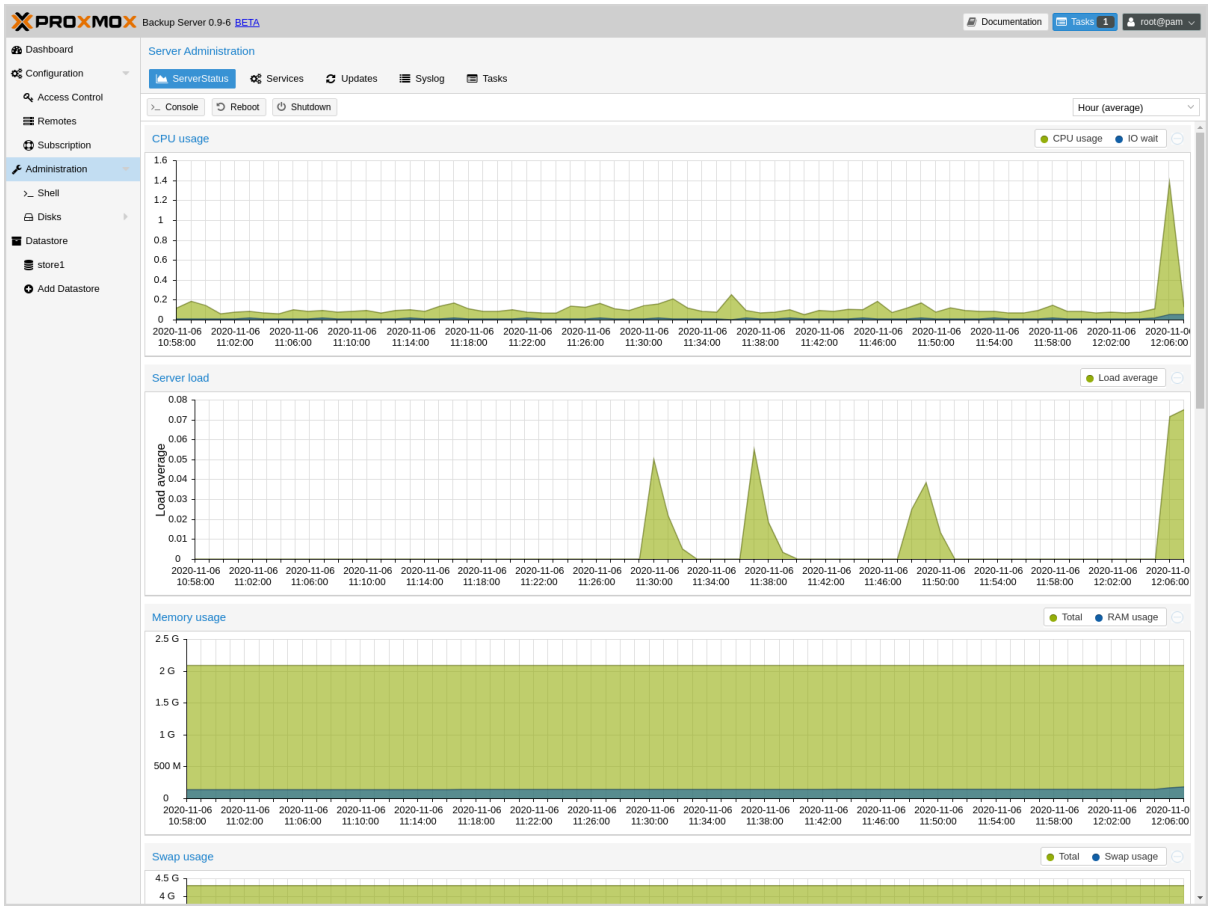
The Dashboard shows a summary of activity and resource usage on the server. Specifically, this displays hardware usage, a summary of previous and currently running tasks, and subscription information.

4.4.2 Configuration

The Configuration section contains some system configuration options, such as time and network configuration. It also contains the following subsections:

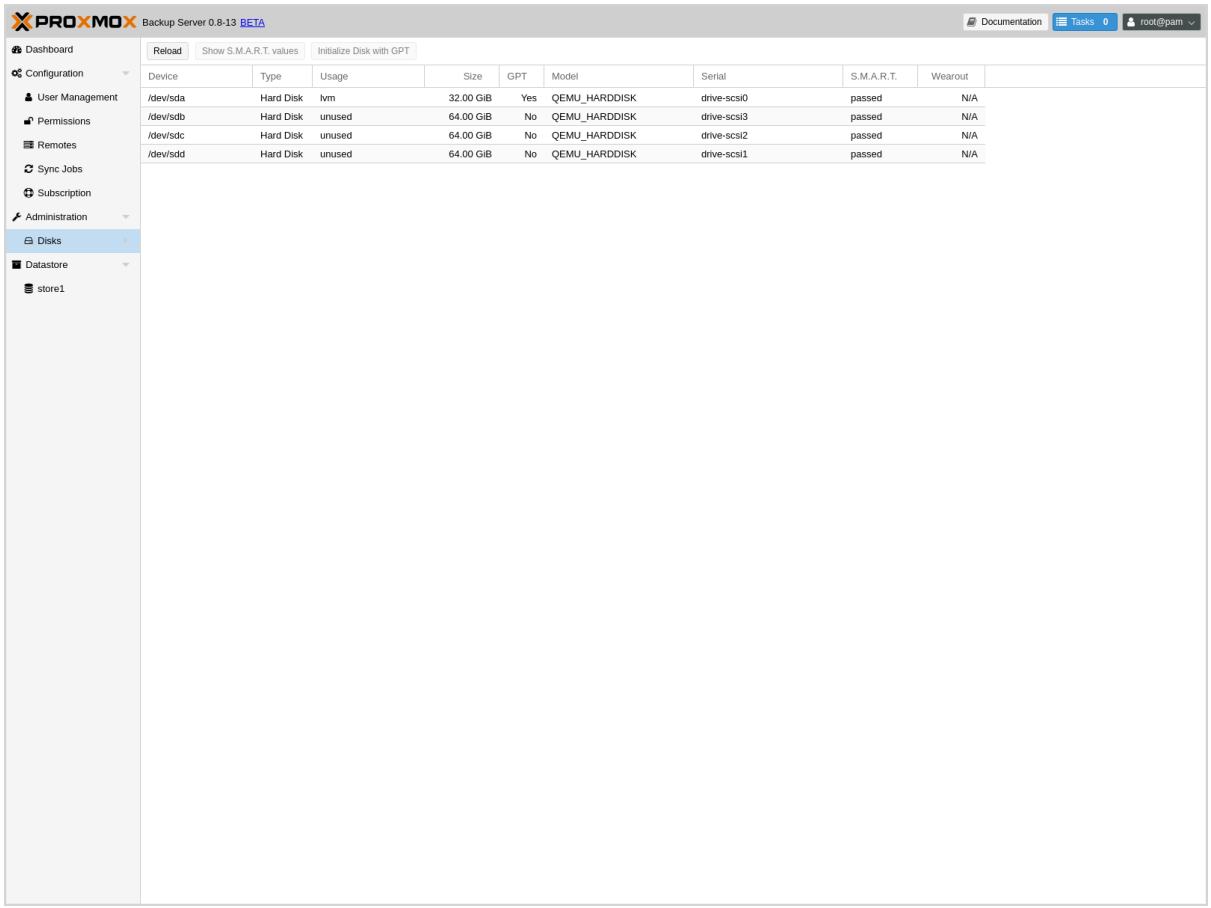
- **Access Control**: Add and manage users, API tokens, and the permissions associated with these items
- **Remotes**: Add, edit and remove remotes (see *Remote*)
- **Subscription**: Upload a subscription key, view subscription status and access a text-based system report.

4.4.3 Administration



The Administration section contains a top panel, with further administration tasks and information. These are:

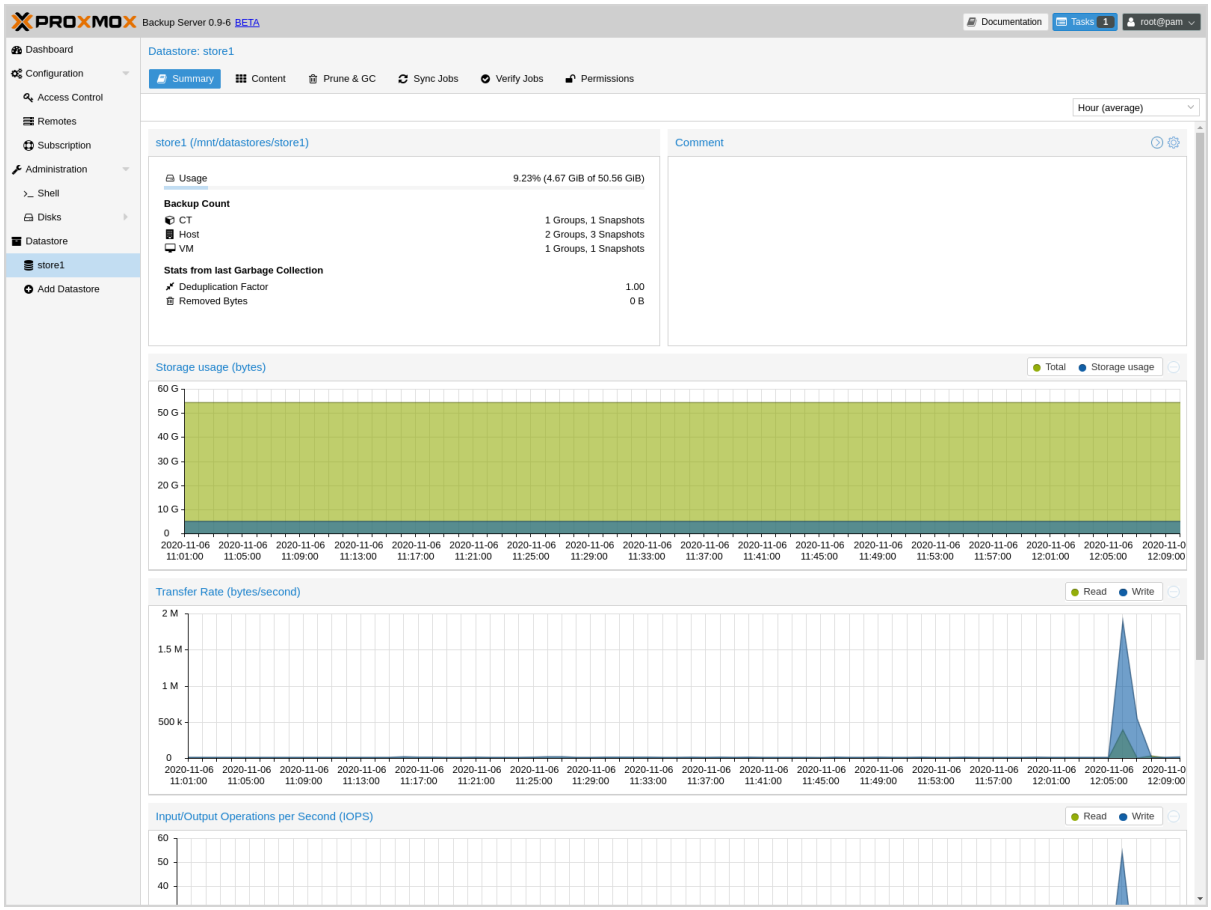
- **ServerStatus**: Provides access to the console, power options, and various resource usage statistics
- **Services**: Manage and monitor system services
- **Updates**: An interface for upgrading packages
- **Syslog**: View log messages from the server
- **Tasks**: Task history with multiple filter options



The administration menu item also contains a disk management subsection:

- **Disks**: View information on available disks
  - **Directory**: Create and view information on *ext4* and *xfs* disks
  - **ZFS**: Create and view information on *ZFS* disks

4.4.4 Datastore



The Datastore section contains interfaces for creating and managing datastores. It contains a button to create a new datastore on the server, as well as a subsection for each datastore on the system, in which you can use the top panel to view:

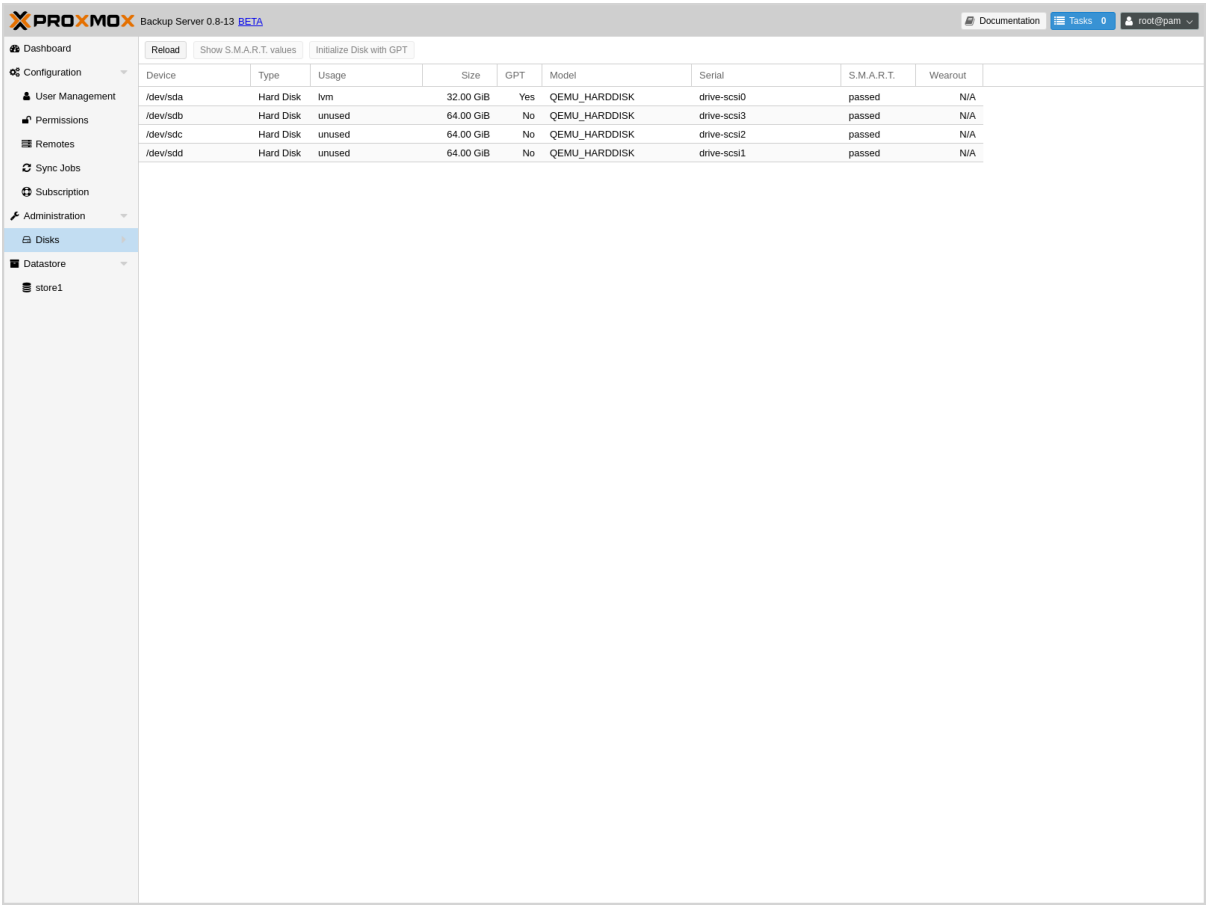
- **Summary**: Access a range of datastore usage statistics
- **Content**: Information on the datastore's backup groups and their respective contents
- **Prune & GC**: Schedule *pruning* and *garbage collection* operations, and run garbage collection manually
- **Sync Jobs**: Create, manage and run *Sync Jobs* from remote servers
- **Verify Jobs**: Create, manage and run *Verification* jobs on the datastore

CHAPTER

FIVE

STORAGE

5.1 Disk Management



Proxmox Backup Server comes with a set of disk utilities, which are accessed using the `disk` subcommand. This subcommand allows you to initialize disks, create various filesystems, and get information about the disks.

To view the disks connected to the system, navigate to **Administration -> Disks** in the web interface or use the `list` subcommand of `disk`:

```
# proxmox-backup-manager disk list
```

name	used	gpt	disk-type	size	model	wearout	status
sda	lvm	1	hdd	34359738368	QEMU_HARDDISK	-	passed
sdb	unused	1	hdd	68719476736	QEMU_HARDDISK	-	passed

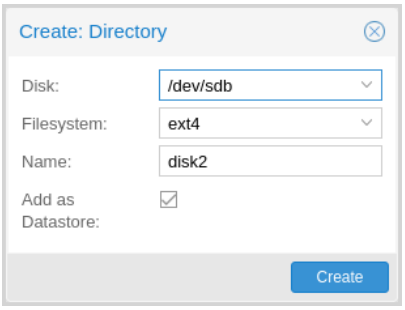
(continues on next page)

(continued from previous page)

```
| sdc | unused |  1 | hdd | 68719476736 | QEMU_HARDDISK |   - | passed |
```

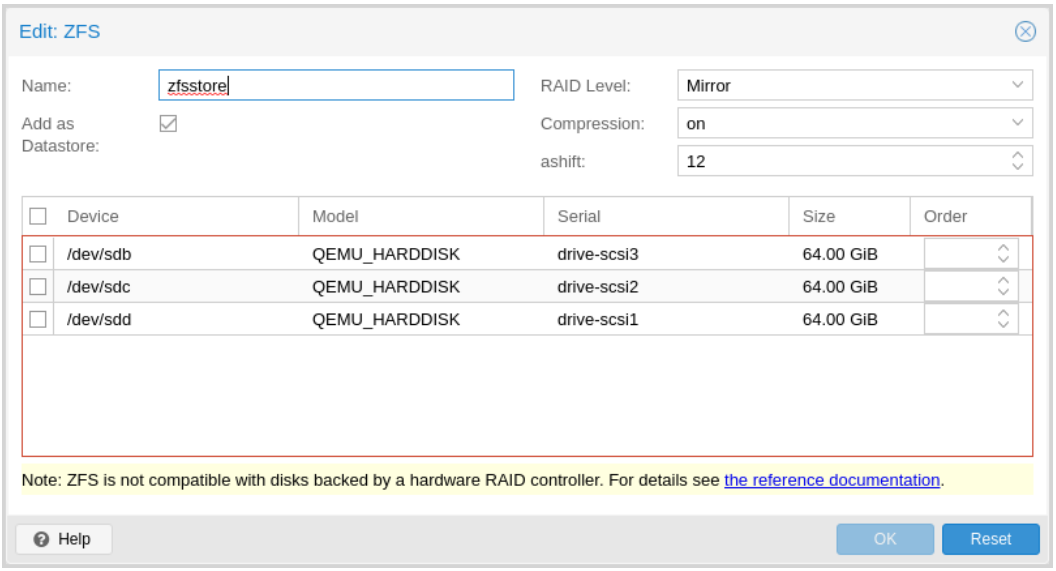
To initialize a disk with a new GPT, use the `initialize` subcommand:

```
# proxmox-backup-manager disk initialize sdX
```



You can create an `ext4` or `xfs` filesystem on a disk using `fs create`, or by navigating to **Administration -> Disks -> Directory** in the web interface and creating one from there. The following command creates an `ext4` filesystem and passes the `--add-datastore` parameter, in order to automatically create a datastore on the disk (in this case `sdd`). This will create a datastore at the location `/mnt/datastore/store1`:

```
# proxmox-backup-manager disk fs create store1 --disk sdd --filesystem ext4 --add-datastore␣
↪true
```



You can also create a `zpool` with various raid levels from **Administration -> Disks -> Zpool** in the web interface, or by using `zpool create`. The command below creates a mirrored `zpool` using two disks (`sdb` & `sdc`) and mounts it under `/mnt/datastore/zpool1`:

```
# proxmox-backup-manager disk zpool create zpool1 --devices sdb,sdc --raidlevel mirror
```

Note: You can also pass the `--add-datastore` parameter here, to automatically create a datastore from the disk.

You can use `disk fs list` and `disk zpool list` to keep track of your filesystems and zpools respectively.

Proxmox Backup Server uses the package smartmontools. This is a set of tools used to monitor

and control the S.M.A.R.T. system for local hard disks. If a disk supports S.M.A.R.T. capability, and you have this enabled, you can display S.M.A.R.T. attributes from the web interface or by using the command:

```
# proxmox-backup-manager disk smart-attributes sdX
```

Note: This functionality may also be accessed directly through the use of the `smartctl` command, which comes as part of the smartmontools package (see `man smartctl` for more details).

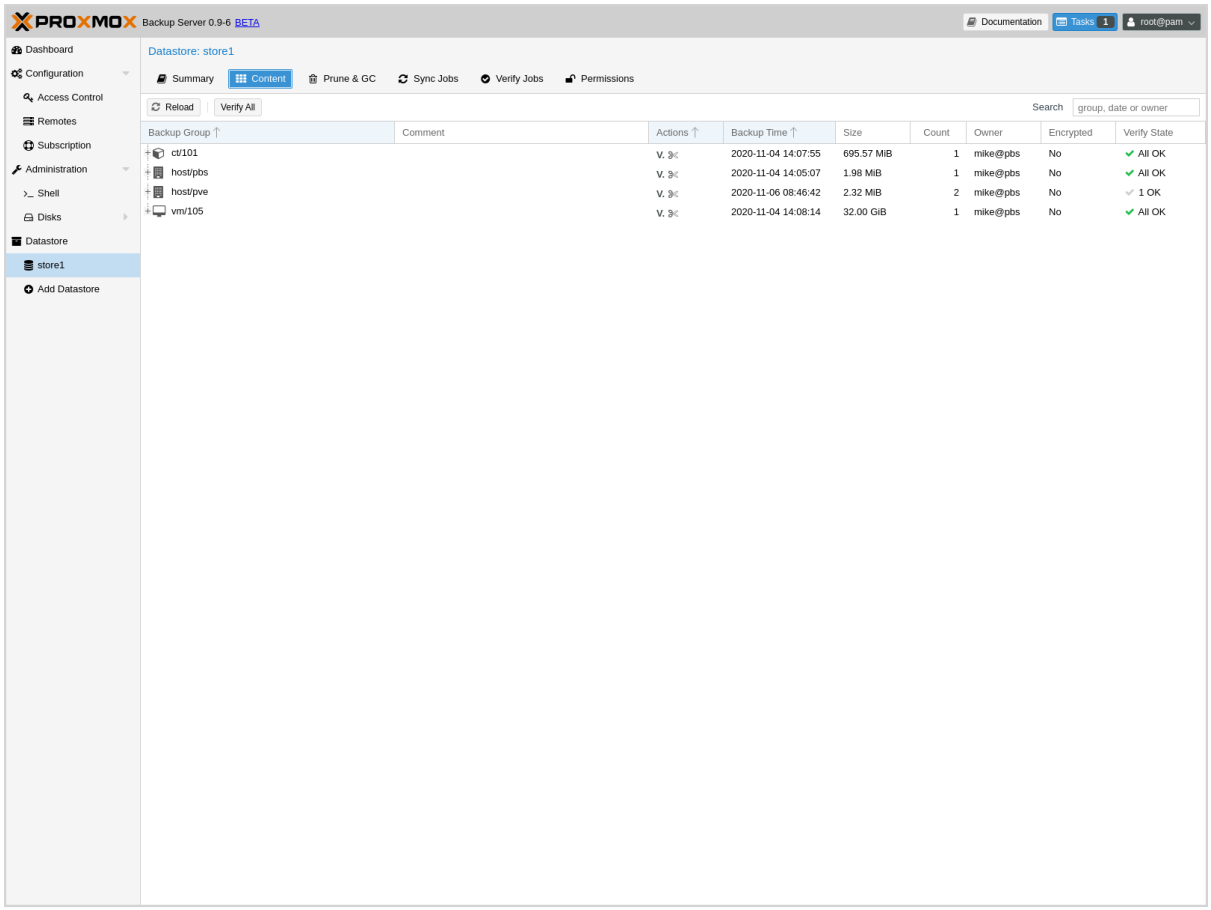
5.2 Datastore

A datastore refers to a location at which backups are stored. The current implementation uses a directory inside a standard Unix file system (`ext4`, `xfs` or `zfs`) to store the backup data.

Datastores are identified by a simple *ID*. You can configure this when setting up the datastore. The configuration information for datastores is stored in the file `/etc/proxmox-backup/datastore.cfg`.

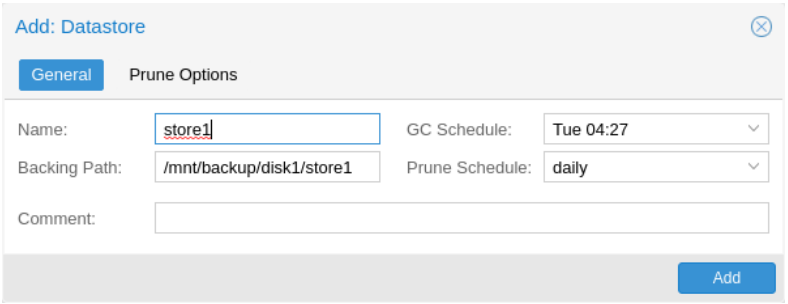
Note: The *File Layout* requires the file system to support at least *65538* subdirectories per directory. That number comes from the 2^{16} pre-created chunk namespace directories, and the `.` and `..` default directory entries. This requirement excludes certain filesystems and filesystem configuration from being supported for a datastore. For example, `ext3` as a whole or `ext4` with the `dir_nlink` feature manually disabled.

5.2.1 Datastore Configuration



You can configure multiple datastores. Minimum one datastore needs to be configured. The datastore is identified by a simple *name* and points to a directory on the filesystem. Each datastore also has associated retention settings of how many backup snapshots for each interval of `hourly`, `daily`, `weekly`, `monthly`, `yearly` as well as a time-independent number of backups to keep in that store. *Pruning and Removing Backups* and *garbage collection* can also be configured to run periodically based on a configured schedule (see *Calendar Events*) per datastore.

Creating a Datastore



You can create a new datastore from the web interface, by clicking **Add Datastore** in the side menu, under the **Datastore** section. In the setup window:

- *Name* refers to the name of the datastore
- *Backing Path* is the path to the directory upon which you want to create the datastore
- *GC Schedule* refers to the time and intervals at which garbage collection runs

- *Prune Schedule* refers to the frequency at which pruning takes place
- *Prune Options* set the amount of backups which you would like to keep (see *Pruning and Removing Backups*).
- *Comment* can be used to add some contextual information to the datastore.

Alternatively you can create a new datastore from the command line. The following command creates a new datastore called `store1` on `/backup/disk1/store1`

```
# proxmox-backup-manager datastore create store1 /backup/disk1/store1
```

Managing Datastores

To list existing datastores from the command line run:

```
# proxmox-backup-manager datastore list
```

name	path	comment
store1	/backup/disk1/store1	This is my default storage.

You can change the garbage collection and prune settings of a datastore, by editing the datastore from the GUI or by using the `update` subcommand. For example, the below command changes the garbage collection schedule using the `update` subcommand and prints the properties of the datastore with the `show` subcommand:

```
# proxmox-backup-manager datastore update store1 --gc-schedule 'Tue 04:27'
# proxmox-backup-manager datastore show store1
```

Name	Value
name	store1
path	/backup/disk1/store1
comment	This is my default storage.
gc-schedule	Tue 04:27
keep-last	7
prune-schedule	daily

Finally, it is possible to remove the datastore configuration:

```
# proxmox-backup-manager datastore remove store1
```

Note: The above command removes only the datastore configuration. It does not delete any data from the underlying directory.

File Layout

After creating a datastore, the following default layout will appear:

```
# ls -arilh /backup/disk1/store1
276493 -rw-r--r-- 1 backup backup       0 Jul  8 12:35 .lock
276490 drwxr-x--- 1 backup backup 1064960 Jul  8 12:35 .chunks
```

.lock is an empty file used for process locking.

The *.chunks* directory contains folders, starting from *0000* and taking hexadecimal values until *ffff*. These directories will store the chunked data after a backup operation has been executed.

```
# ls -arilh /backup/disk1/store1/.chunks
545824 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 ffff
545823 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 fffe
415621 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 fffd
415620 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 fffc
353187 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 fffb
344995 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 fffa
144079 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 fff9
144078 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 fff8
144077 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 fff7
...
403180 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 000c
403179 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 000b
403177 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 000a
402530 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 0009
402513 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 0008
402509 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 0007
276509 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 0006
276508 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 0005
276507 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 0004
276501 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 0003
276499 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 0002
276498 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 0001
276494 drwxr-x--- 2 backup backup 4.0K Jul  8 12:35 0000
276489 drwxr-xr-x 3 backup backup 4.0K Jul  8 12:35 ..
276490 drwxr-x--- 1 backup backup 1.1M Jul  8 12:35 .
```

CHAPTER

SIX

NETWORK MANAGEMENT

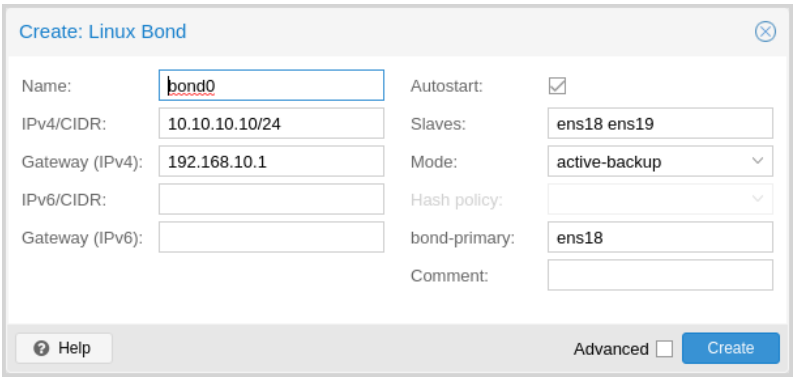
Proxmox Backup Server provides both a web interface and a command line tool for network configuration. You can find the configuration options in the web interface under the **Network Interfaces** section of the **Configuration** menu tree item. The command line tool is accessed via the `network` subcommand. These interfaces allow you to carry out some basic network management tasks, such as adding, configuring, and removing network interfaces.

Note: Any changes made to the network configuration are not applied, until you click on **Apply Configuration** or enter the `network reload` command. This allows you to make many changes at once. It also allows you to ensure that your changes are correct before applying them, as making a mistake here can render the server inaccessible over the network.

To get a list of available interfaces, use the following command:

```
# proxmox-backup-manager network list
```

name	type	autostart	method	address	gateway	ports/slaves
bond0	bond	1	static	x.x.x.x/x	x.x.x.x	ens18 ens19
ens18	eth	1	manual			
ens19	eth	1	manual			



To add a new network interface, use the `create` subcommand with the relevant parameters. For example, you may want to set up a bond, for the purpose of network redundancy. The following command shows a template for creating the bond shown in the list above:

```
# proxmox-backup-manager network create bond0 --type bond --bond_mode active-backup --slaves␣
↪ens18,ens19 --autostart true --cidr x.x.x.x/x --gateway x.x.x.x
```

You can make changes to the configuration of a network interface with the `update` subcommand:

```
# proxmox-backup-manager network update bond0 --cidr y.y.y.y/y
```

You can also remove a network interface:

```
# proxmox-backup-manager network remove bond0
```

The pending changes for the network configuration file will appear at the bottom of the web interface. You can also view these changes, by using the command:

```
# proxmox-backup-manager network changes
```

If you would like to cancel all changes at this point, you can either click on the **Revert** button or use the following command:

```
# proxmox-backup-manager network revert
```

If you are happy with the changes and would like to write them into the configuration file, select **Apply Configuration**. The corresponding command is:

```
# proxmox-backup-manager network reload
```

Note: This command and corresponding GUI button rely on the `ifreload` command, from the package `ifupdown2`. This package is included within the Proxmox Backup Server installation, however, you may have to install it yourself, if you have installed Proxmox Backup Server on top of Debian or Proxmox VE.

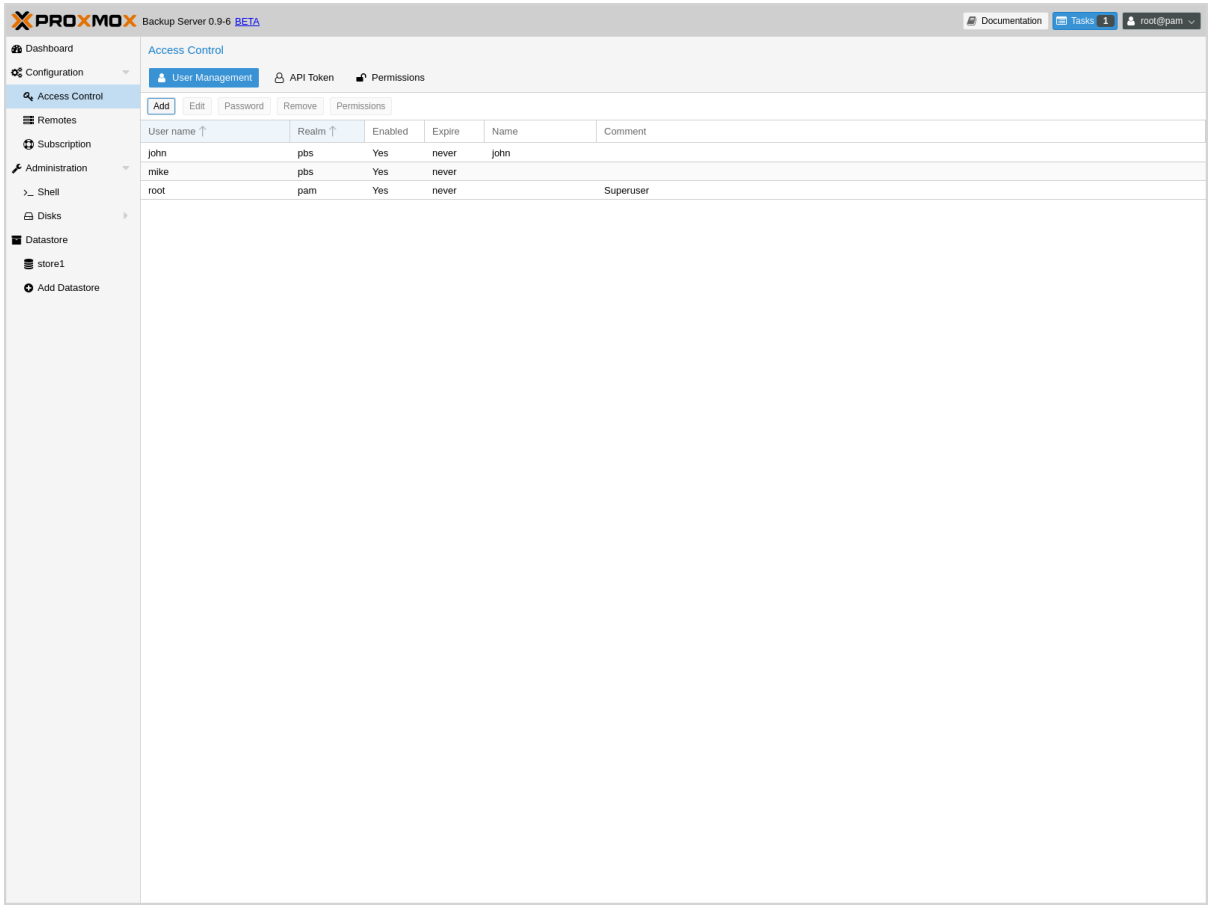
You can also configure DNS settings, from the **DNS** section of **Configuration** or by using the `dns` subcommand of `proxmox-backup-manager`.

CHAPTER

SEVEN

USER MANAGEMENT

7.1 User Configuration



Proxmox Backup Server supports several authentication realms, and you need to choose the realm when you add a new user. Possible realms are:

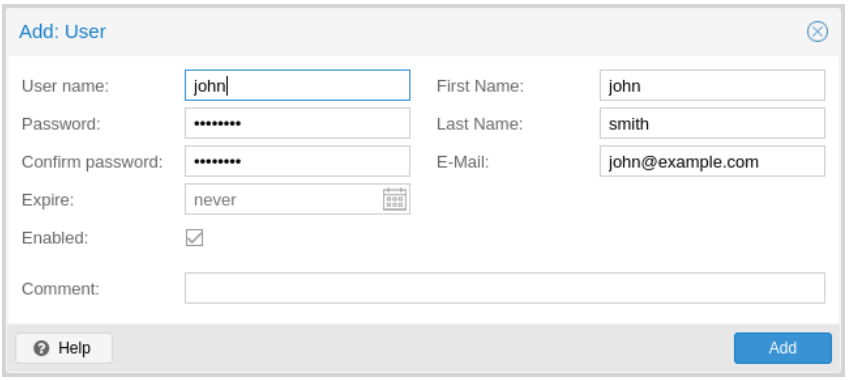
pam Linux PAM standard authentication. Use this if you want to authenticate as Linux system user (Users need to exist on the system).

pbs Proxmox Backup Server realm. This type stores hashed passwords in `/etc/proxmox-backup/shadow.json`.

After installation, there is a single user `root@pam`, which corresponds to the Unix superuser. User configuration information is stored in the file `/etc/proxmox-backup/user.cfg`. You can use the `proxmox-backup-manager` command line tool to list or manipulate users:

```
# proxmox-backup-manager user list
```

userid	enable	expire	firstname	lastname	email	comment
root@pam	1					Superuser



The superuser has full administration rights on everything, so you normally want to add other users with less privileges. You can add a new user with the `user create` subcommand or through the web interface, under the **User Management** tab of **Configuration -> Access Control**. The `create` subcommand lets you specify many options like `--email` or `--password`. You can update or change any user properties using the `update` subcommand later (**Edit** in the GUI):

```
# proxmox-backup-manager user create john@pbs --email john@example.com
# proxmox-backup-manager user update john@pbs --firstname John --lastname Smith
# proxmox-backup-manager user update john@pbs --comment "An example user."
```

The resulting user list looks like this:

```
# proxmox-backup-manager user list
```

userid	enable	expire	firstname	lastname	email	comment
john@pbs	1		John	Smith	john@example.com	An example user.
root@pam	1					Superuser

Newly created users do not have any permissions. Please read the Access Control section to learn how to set access permissions.

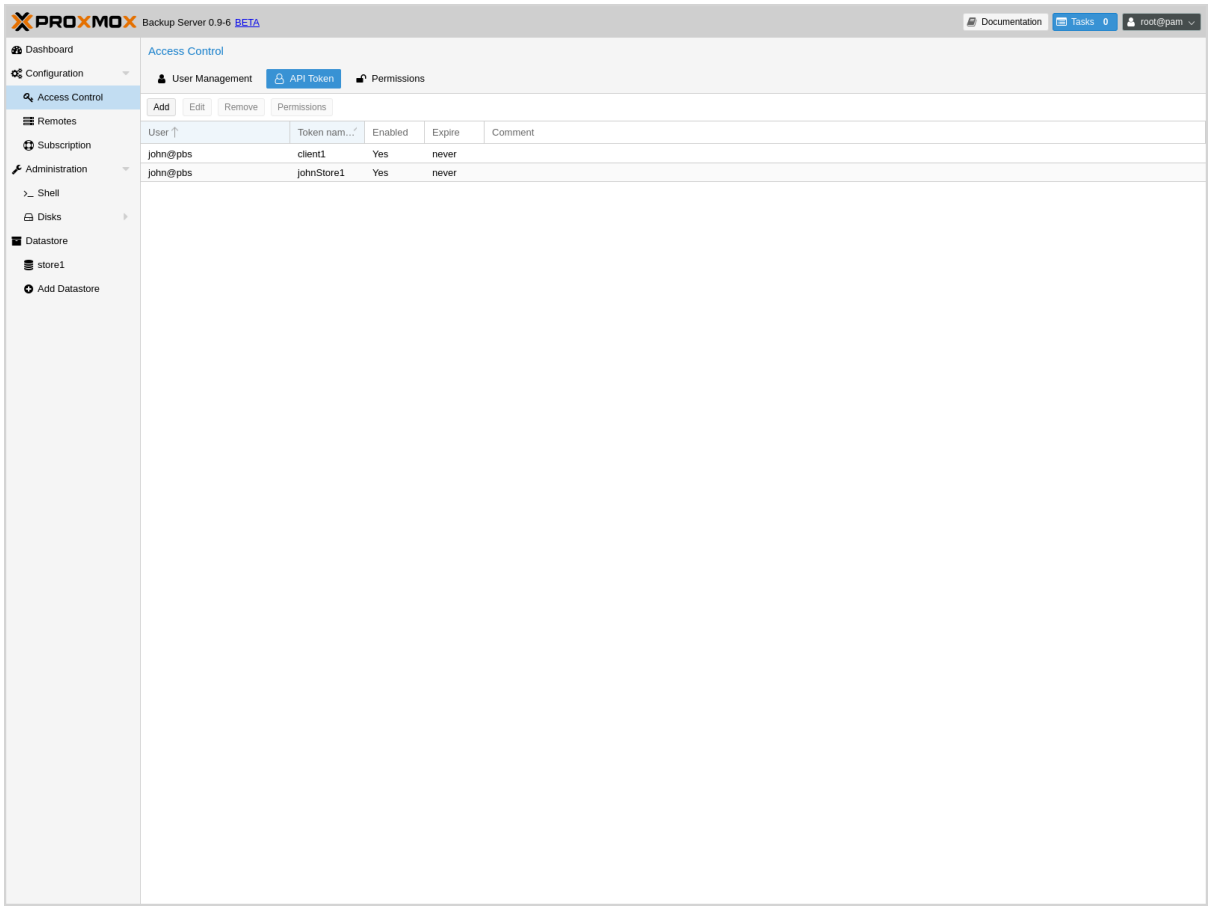
If you want to disable a user account, you can do that by setting `--enable` to `0`

```
# proxmox-backup-manager user update john@pbs --enable 0
```

Or completely remove the user with:

```
# proxmox-backup-manager user remove john@pbs
```

7.2 API Tokens

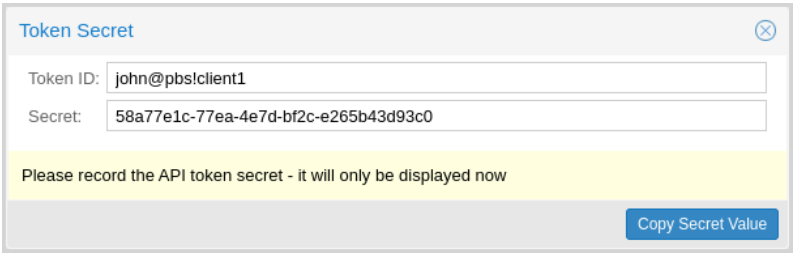


Any authenticated user can generate API tokens which can in turn be used to configure various clients, instead of directly providing the username and password.

API tokens serve two purposes:

1. Easy revocation in case client gets compromised
2. Limit permissions for each client/token within the users' permission

An API token consists of two parts: an identifier consisting of the user name, the realm and a tokenname (`user@realm!tokenname`), and a secret value. Both need to be provided to the client in place of the user ID (`user@realm`) and the user password, respectively.



The API token is passed from the client to the server by setting the `Authorization` HTTP header with method `PBSAPIToken` to the value `TOKENID:TOKENSECRET`.

Generating new tokens can done using `proxmox-backup-manager` or the GUI:

```
# proxmox-backup-manager user generate-token john@pbs client1
Result: {
```

(continues on next page)

(continued from previous page)

```
  "tokenid": "john@pbs!client1",
  "value": "d63e505a-e3ec-449a-9bc7-1da610d4ccde"
}
```

Note: The displayed secret value needs to be saved, since it cannot be displayed again after generating the API token.

The `user list-tokens` sub-command can be used to display tokens and their metadata:

```
# proxmox-backup-manager user list-tokens john@pbs
```

tokenid	enable	expire	comment
john@pbs!client1	1		

Similarly, the `user delete-token` subcommand can be used to delete a token again.

Newly generated API tokens don't have any permissions. Please read the next section to learn how to set access permissions.

7.3 Access Control

By default new users and API tokens do not have any permission. Instead you need to specify what is allowed and what is not. You can do this by assigning roles to users/tokens on specific objects like datastores or remotes. The following roles exist:

NoAccess Disable Access - nothing is allowed.

Admin Can do anything.

Audit Can view things, but is not allowed to change settings.

DatastoreAdmin Can do anything on datastores.

DatastoreAudit Can view datastore settings and list content. But is not allowed to read the actual data.

DatastoreReader Can Inspect datastore content and can do restores.

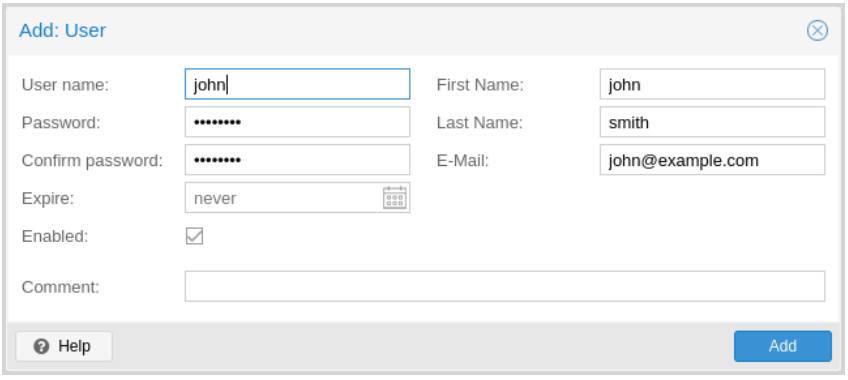
DatastoreBackup Can backup and restore owned backups.

DatastorePowerUser Can backup, restore, and prune owned backups.

RemoteAdmin Can do anything on remotes.

RemoteAudit Can view remote settings.

RemoteSyncOperator Is allowed to read data from a remote.



Access permission information is stored in `/etc/proxmox-backup/acl.cfg`. The file contains 5 fields, separated using a colon (':') as a delimiter. A typical entry takes the form:

`acl:1:/datastore:john@pbs:DatastoreBackup`

The data represented in each field is as follows:

1. `acl` identifier
2. A `1` or `0`, representing whether propagation is enabled or disabled, respectively
3. The object on which the permission is set. This can be a specific object (single datastore, remote, etc.) or a top level object, which with propagation enabled, represents all children of the object also.
4. The user(s)/token(s) for which the permission is set
5. The role being set

You can manage permissions via **Configuration -> Access Control -> Permissions** in the web interface. Likewise, you can use the `acl` subcommand to manage and monitor user permissions from the command line. For example, the command below will add the user `john@pbs` as a **DatastoreAdmin** for the datastore `store1`, located at `/backup/disk1/store1`:

```
# proxmox-backup-manager acl update /datastore/store1 DatastoreAdmin --auth-id john@pbs
```

You can list the ACLs of each user/token using the following command:

```
# proxmox-backup-manager acl list
```

ugid	path	propagate	roleid
john@pbs	/datastore/store1	1	DatastoreAdmin

A single user/token can be assigned multiple permission sets for different datastores.

Note: Naming convention is important here. For datastores on the host, you must use the convention `/datastore/{storename}`. For example, to set permissions for a datastore mounted at `/mnt/backup/disk4/store2`, you would use `/datastore/store2` for the path. For remote stores, use the convention `/remote/{remote}/{storename}`, where `{remote}` signifies the name of the remote (see *Remote* below) and `{storename}` is the name of the datastore on the remote.

7.3.1 API Token permissions

API token permissions are calculated based on ACLs containing their ID independent of those of their corresponding user. The resulting permission set on a given path is then intersected with that of the corresponding user.

In practice this means:

1. API tokens require their own ACL entries
2. API tokens can never do more than their corresponding user

7.3.2 Effective permissions

To calculate and display the effective permission set of a user or API token you can use the `proxmox-backup-manager user permission` command:

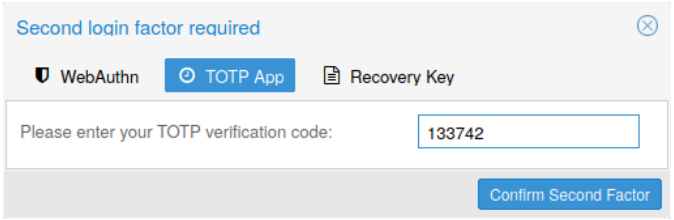
```
# proxmox-backup-manager user permissions john@pbs --path /datastore/store1
Privileges with (*) have the propagate flag set

Path: /datastore/store1
- Datastore.Audit (*)
- Datastore.Backup (*)
- Datastore.Modify (*)
- Datastore.Prune (*)
- Datastore.Read (*)
- Datastore.Verify (*)

# proxmox-backup-manager acl update /datastore/store1 DatastoreBackup --auth-id 'john@pbs!
↪client1'
# proxmox-backup-manager user permissions 'john@pbs!client1' --path /datastore/store1
Privileges with (*) have the propagate flag set

Path: /datastore/store1
- Datastore.Backup (*)
```

7.4 Two-factor authentication

7.4.1 Introduction

With simple authentication, only a password (single factor) is required to successfully claim an identity (authenticate), for example, to be able to log in as *root@pam* on a specific instance of Proxmox Backup Server. In this case, if the password gets stolen or leaked, anybody can use it to log in - even if they should not be allowed to do so.

With two-factor authentication (TFA), a user is asked for an additional factor to verify their authenticity. Rather than relying on something only the user knows (a password), this extra factor requires something only the user has, for example, a piece of hardware (security key) or a secret saved on the user's smartphone. This prevents a remote user from gaining unauthorized access to an account, as even if they have the password, they will not have access to the physical object (second factor).



7.4.2 Available Second Factors

You can set up multiple second factors, in order to avoid a situation in which losing your smartphone or security key locks you out of your account permanently.

Proxmox Backup Server supports three different two-factor authentication methods:

- TOTP (Time-based One-Time Password). A short code derived from a shared secret and the current time, it changes every 30 seconds.
- WebAuthn (Web Authentication). A general standard for authentication. It is implemented by various security devices, like hardware keys or trusted platform modules (TPM) from a computer or smart phone.
- Single use Recovery Keys. A list of keys which should either be printed out and locked in a secure place or saved digitally in an electronic vault. Each key can be used only once. These

are perfect for ensuring that you are not locked out, even if all of your other second factors are lost or corrupt.

7.4.3 Setup

TOTP

There is no server setup required. Simply install a TOTP app on your smartphone (for example, FreeOTP) and use the Proxmox Backup Server web-interface to add a TOTP factor.

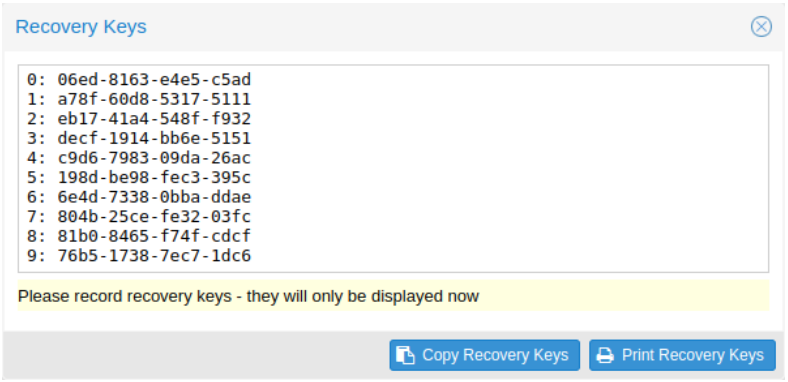
WebAuthn

For WebAuthn to work, you need to have two things:

- a trusted HTTPS certificate (for example, by using Let's Encrypt)
- setup the WebAuthn configuration (see *Configuration -> Authentication* in the Proxmox Backup Server web-interface). This can be auto-filled in most setups.

Once you have fulfilled both of these requirements, you can add a WebAuthn configuration in the *Access Control* panel.

Recovery Keys



Recovery key codes do not need any preparation; you can simply create a set of recovery keys in the *Access Control* panel.

Note: There can only be one set of single-use recovery keys per user at any time.

7.4.4 TFA and Automated Access

Two-factor authentication is only implemented for the web-interface. You should use *API Tokens* for all other use cases, especially non-interactive ones (for example, adding a Proxmox Backup Server to Proxmox VE as a storage).

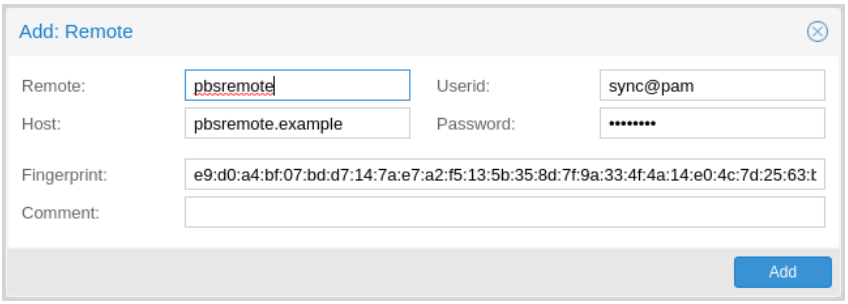
CHAPTER

EIGHT

MANAGING REMOTES

8.1 Remote

A remote refers to a separate Proxmox Backup Server installation and a user on that installation, from which you can *sync* datastores to a local datastore with a *Sync Job*. You can configure remotes in the web interface, under **Configuration -> Remotes**. Alternatively, you can use the `remote` subcommand. The configuration information for remotes is stored in the file `/etc/proxmox-backup/remote.cfg`.



To add a remote, you need its hostname or IP, a userid and password on the remote, and its certificate fingerprint. To get the fingerprint, use the `proxmox-backup-manager cert info` command on the remote, or navigate to **Dashboard** in the remote's web interface and select **Show Fingerprint**.

```
# proxmox-backup-manager cert info |grep Fingerprint
Fingerprint (sha256): 64:d3:ff:3a:50:38:53:5a:9b:f7:50:...:ab:fe
```

Using the information specified above, you can add a remote from the **Remotes** configuration panel, or by using the command:

```
# proxmox-backup-manager remote create pbs2 --host pbs2.mydomain.example --userid sync@pam --
 ↪password 'SECRET' --fingerprint 64:d3:ff:3a:50:38:53:5a:9b:f7:50:...:ab:fe
```

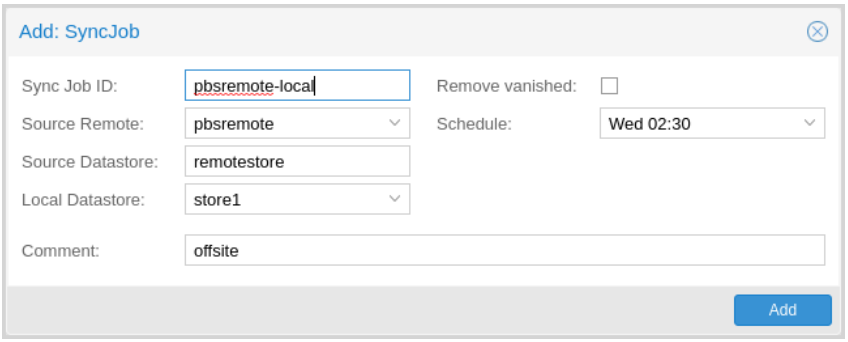
Use the `list`, `show`, `update`, `remove` subcommands of `proxmox-backup-manager remote` to manage your remotes:

```
# proxmox-backup-manager remote update pbs2 --host pbs2.example
# proxmox-backup-manager remote list
```

name	host	userid	fingerprint	comment
pbs2	pbs2.example	sync@pam	64:d3:ff:3a:50:38:53:5a:9b:f7:50:...:ab:fe	

```
# proxmox-backup-manager remote remove pbs2
```

8.2 Sync Jobs



Sync jobs are configured to pull the contents of a datastore on a **Remote** to a local datastore. You can manage sync jobs in the web interface, from the **Sync Jobs** tab of the datastore which you'd like to set one up for, or using the `proxmox-backup-manager sync-job` command. The configuration information for sync jobs is stored at `/etc/proxmox-backup/sync.cfg`. To create a new sync job, click the add button in the GUI, or use the `create` subcommand. After creating a sync job, you can either start it manually from the GUI or provide it with a schedule (see *Calendar Events*) to run regularly.

```
# proxmox-backup-manager sync-job create pbs2-local --remote pbs2 --remote-store local --
↪store local --schedule 'Wed 02:30'
# proxmox-backup-manager sync-job update pbs2-local --comment 'offsite'
# proxmox-backup-manager sync-job list
```

id	store	remote	remote-store	schedule	comment
pbs2-local	local	pbs2	local	Wed 02:30	offsite

```
# proxmox-backup-manager sync-job remove pbs2-local
```

For setting up sync jobs, the configuring user needs the following permissions:

1. `Remote.Read` on the `/remote/{remote}/{remote-store}` path
2. at least `Datastore.Backup` on the local target datastore (`/datastore/{store}`)

If the `remove-vanished` option is set, `Datastore.Prune` is required on the local datastore as well. If the `owner` option is not set (defaulting to `root@pam`) or set to something other than the configuring user, `Datastore.Modify` is required as well.

Note: A sync job can only sync backup groups that the configured remote's user/API token can read. If a remote is configured with a user/API token that only has `Datastore.Backup` privileges, only the limited set of accessible snapshots owned by that user/API token can be synced.

CHAPTER

NINE

MAINTENANCE TASKS

9.1 Pruning

Prune lets you specify which backup snapshots you want to keep. The following retention options are available:

keep-last <N> Keep the last <N> backup snapshots.

keep-hourly <N> Keep backups for the last <N> hours. If there is more than one backup for a single hour, only the latest is kept.

keep-daily <N> Keep backups for the last <N> days. If there is more than one backup for a single day, only the latest is kept.

keep-weekly <N> Keep backups for the last <N> weeks. If there is more than one backup for a single week, only the latest is kept.

> **Note:** Weeks start on Monday and end on Sunday. The software uses the ISO week date system and handles weeks at the end of the year correctly.

keep-monthly <N> Keep backups for the last <N> months. If there is more than one backup for a single month, only the latest is kept.

keep-yearly <N> Keep backups for the last <N> years. If there is more than one backup for a single year, only the latest is kept.

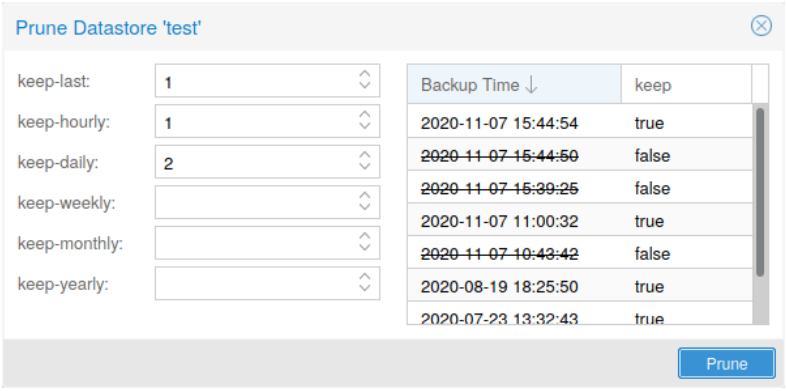
The retention options are processed in the order given above. Each option only covers backups within its time period. The next option does not take care of already covered backups. It will only consider older backups.

Unfinished and incomplete backups will be removed by the prune command unless they are newer than the last successful backup. In this case, the last failed backup is retained.

9.1.1 Prune Simulator

You can use the built-in prune simulator to explore the effect of different retetion options with various backup schedules.

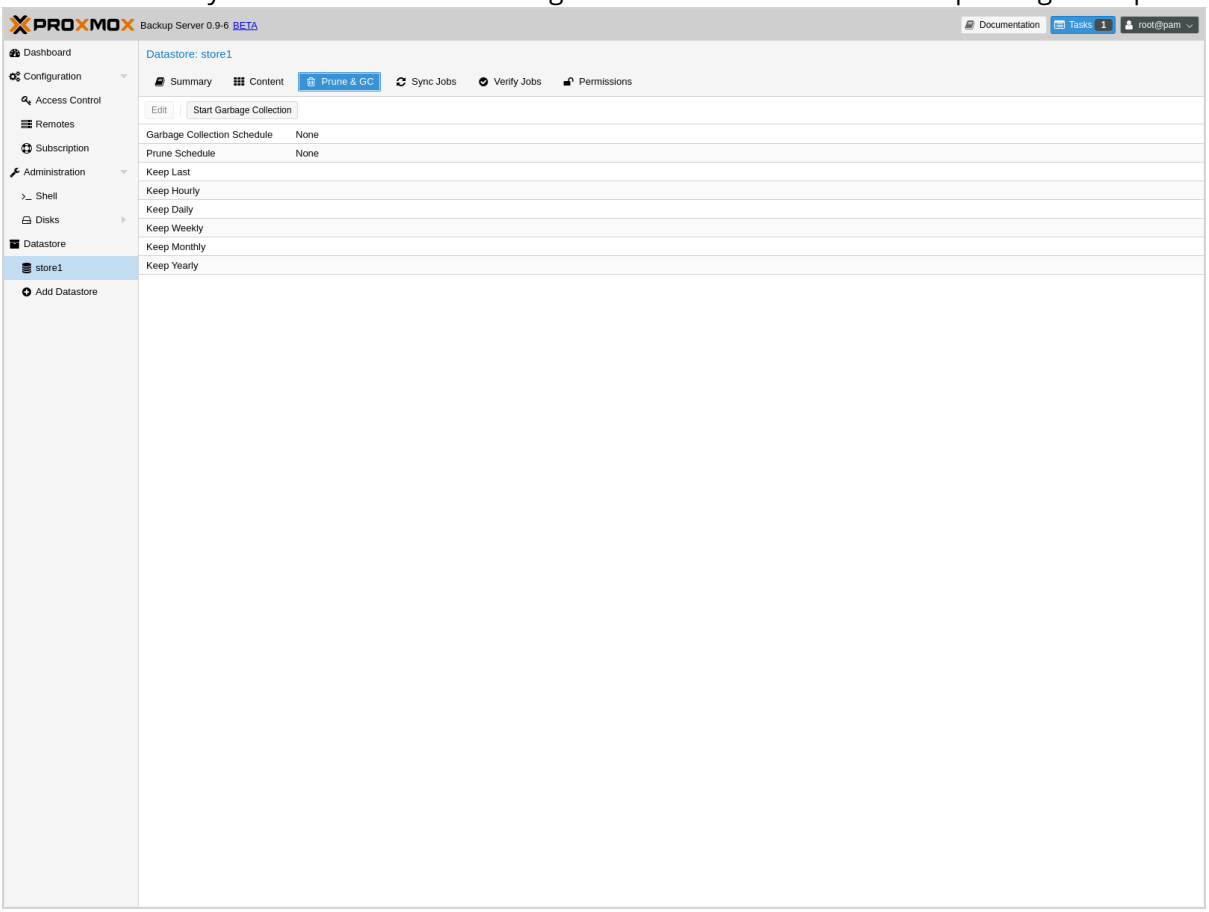
9.1.2 Manual Pruning



To access pruning functionality for a specific backup group, you can use the prune command line option discussed in *Pruning and Removing Backups*, or navigate to the **Content** tab of the datastore and click the scissors icon in the **Actions** column of the relevant backup group.

9.1.3 Prune Schedules

To prune on a datastore level, scheduling options can be found under the **Prune & GC** tab of the datastore. Here you can set retention settings and edit the interval at which pruning takes place.



9.1.4 Retention Settings Example

The backup frequency and retention of old backups may depend on how often data changes, and how important an older state may be, in a specific work load. When backups act as a company's document archive, there may also be legal requirements for how long backup snapshots must be kept.

For this example, we assume that you are doing daily backups, have a retention period of 10 years, and the period between backups stored gradually grows.

- **keep-last:** 3 - even if only daily backups, an admin may want to create an extra one just before or after a big upgrade. Setting keep-last ensures this.
- **keep-hourly:** not set - for daily backups this is not relevant. You cover extra manual backups already, with keep-last.
- **keep-daily:** 13 - together with keep-last, which covers at least one day, this ensures that you have at least two weeks of backups.
- **keep-weekly:** 8 - ensures that you have at least two full months of weekly backups.
- **keep-monthly:** 11 - together with the previous keep settings, this ensures that you have at least a year of monthly backups.
- **keep-yearly:** 9 - this is for the long term archive. As you covered the current year with the previous options, you would set this to nine for the remaining ones, giving you a total of at least 10 years of coverage.

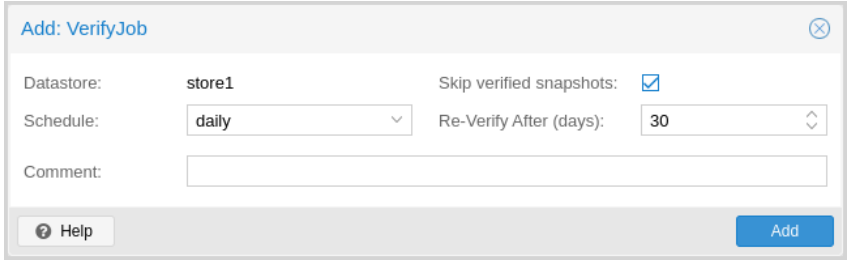
We recommend that you use a higher retention period than is minimally required by your environment; you can always reduce it if you find it is unnecessarily high, but you cannot recreate backup snapshots from the past.

9.2 Garbage Collection

You can monitor and run *garbage collection* on the Proxmox Backup Server using the `garbage-collection` subcommand of `proxmox-backup-manager`. You can use the `start` subcommand to manually start garbage collection on an entire datastore and the `status` subcommand to see attributes relating to the *garbage collection*.

This functionality can also be accessed in the GUI, by navigating to **Prune & GC** from the top panel. From here, you can edit the schedule at which garbage collection runs and manually start the operation.

9.3 Verification



Proxmox Backup offers various verification options to ensure that backup data is intact. Verification is generally carried out through the creation of verify jobs. These are scheduled tasks that run verification at a given interval (see *Calendar Events*). With these, you can set whether already verified snapshots are ignored, as well as set a time period, after which verified jobs are checked again. The interface for creating verify jobs can be found under the **Verify Jobs** tab of the datastore.

Note: It is recommended that you reverify all backups at least monthly, even if a previous verification was successful. This is becuase physical drives are susceptible to damage over time, which can cause an old, working backup to become corrupted in a process known as bit rot/data degradation. It is good practice to have a regularly recurring (hourly/daily) verification job, which checks new and expired backups, then another weekly/monthly job that will reverify everything. This way, there will be no surprises when it comes to restoring data.

Aside from using verify jobs, you can also run verification manually on entire datastores, backup groups, or snapshots. To do this, navigate to the **Content** tab of the datastore and either click *Verify All*, or select the *V.* icon from the *Actions* column in the table.

9.4 Notifications

Proxmox Backup Server can send you notification emails about automatically scheduled verification, garbage-collection and synchronization tasks results.

By default, notifications are send to the email address configured for the *root@pam* user. You can set that user for each datastore.

You can also change the level of notification received per task type, the following options are available:

- Always: send a notification for any scheduled task, independent of the outcome
- Errors: send a notification for any scheduled task resulting in an error
- Never: do not send any notification at all

CHAPTER

TEN

BACKUP CLIENT USAGE

The command line client is called **`proxmox-backup-client`**.

10.1 Repository Locations

The client uses the following notation to specify a datastore repository on the backup server.

[[username@]server[:port]:]datastore

The default value for `username` is `root@pam`. If no server is specified, the default is the local host (`localhost`).

You can specify a port if your backup server is only reachable on a different port (e.g. with NAT and port forwarding).

Note that if the server is an IPv6 address, you have to write it with square brackets (for example, *[fe80::01]*).

You can pass the repository with the `--repository` command line option, or by setting the `PBS_REPOSITORY` environment variable.

Here some examples of valid repositories and the real values

Example	User	Host:Port	Datastore
mydatastore	`root@pam`	localhost:8007	mydatastore
myhostname:mydatastore	`root@pam`	myhostname:8007	mydatastore
user@pbs@myhostname:mydatastore	`user@pbs`	myhostname:8007	mydatastore
user@pbs!token@host:store	`user@pbs!token`	myhostname:8007	mydatastore
192.168.55.55:1234:mydatastore	`root@pam`	192.168.55.55:1234	mydatastore
[ff80::51]:mydatastore	`root@pam`	[ff80::51]:8007	mydatastore
[ff80::51]:1234:mydatastore	`root@pam`	[ff80::51]:1234	mydatastore

10.2 Environment Variables

`PBS_REPOSITORY` The default backup repository.

PBS_PASSWORD When set, this value is used for the password required for the backup server. You can also set this to a API token secret.

PBS_ENCRYPTION_PASSWORD When set, this value is used to access the secret encryption key (if protected by password).

PBS_FINGERPRINT When set, this value is used to verify the server certificate (only used if the system CA certificates cannot validate the certificate).

10.3 Output Format

Most commands producing output supports the `--output-format` parameter. It accepts the following values:

`text` Text format (default). Structured data is rendered as a table.

`json` JSON (single line).

`json-pretty` JSON (multiple lines, nicely formatted).

Also, the following environment variables can modify output behavior:

PROXMOX_OUTPUT_FORMAT Defines the default output format.

PROXMOX_OUTPUT_NO_BORDER If set (to any value), do not render table borders.

PROXMOX_OUTPUT_NO_HEADER If set (to any value), do not render table headers.

Note: The `text` format is designed to be human readable, and not meant to be parsed by automation tools. Please use the `json` format if you need to process the output.

10.4 Creating Backups

This section explains how to create a backup from within the machine. This can be a physical host, a virtual machine, or a container. Such backups may contain file and image archives. There are no restrictions in this case.

Note: If you want to backup virtual machines or containers on Proxmox VE, see *Proxmox VE Integration*.

For the following example you need to have a backup server set up, working credentials and need to know the repository name. In the following examples we use `backup-server:store1`.

```
# proxmox-backup-client backup root.pxar:/ --repository backup-server:store1
Starting backup: host/elsa/2019-12-03T09:35:01Z
Client name: elsa
skip mount point: "/boot/efi"
skip mount point: "/dev"
skip mount point: "/run"
skip mount point: "/sys"
Uploaded 12129 chunks in 87 seconds (564 MB/s).
End Time: 2019-12-03T10:36:29+01:00
```

This will prompt you for a password and then uploads a file archive named `root.pxar` containing all the files in the `/` directory.

Caution: Please note that the proxmox-backup-client does not automatically include mount points. Instead, you will see a short `skip mount point` notice for each of them. The idea is to create a separate file archive for each mounted disk. You can explicitly include them using the `--include-dev` option (i.e. `--include-dev /boot/efi`). You can use this option multiple times for each mount point that should be included.

The `--repository` option can get quite long and is used by all commands. You can avoid having to enter this value by setting the environment variable `PBS_REPOSITORY`. Note that if you would like this to remain set over multiple sessions, you should instead add the below line to your `.bashrc` file.

```
# export PBS_REPOSITORY=backup-server:store1
```

After this you can execute all commands without specifying the `--repository` option.

One single backup is allowed to contain more than one archive. For example, if you want to backup two disks mounted at `/mnt/disk1` and `/mnt/disk2`:

```
# proxmox-backup-client backup disk1.pxar:/mnt/disk1 disk2.pxar:/mnt/disk2
```

This creates a backup of both disks.

The backup command takes a list of backup specifications, which include the archive name on the server, the type of the archive, and the archive source at the client. The format is:

<archive-name>.<type>:<source-path>

Common types are `.pxar` for file archives, and `.img` for block device images. To create a backup of a block device run the following command:

```
# proxmox-backup-client backup mydata.img:/dev/mylvm/mydata
```

10.4.1 Excluding files/folders from a backup

Sometimes it is desired to exclude certain files or folders from a backup archive. To tell the Proxmox Backup client when and how to ignore files and directories, place a text file called `.pxarexclude` in the filesystem hierarchy. Whenever the backup client encounters such a file in a directory, it interprets each line as glob match patterns for files and directories that are to be excluded from the backup.

The file must contain a single glob pattern per line. Empty lines are ignored. The same is true for lines starting with `#`, which indicates a comment. A `!` at the beginning of a line reverses the glob match pattern from an exclusion to an explicit inclusion. This makes it possible to exclude all entries in a directory except for a few single files/subdirectories. Lines ending in `/` match only on directories. The directory containing the `.pxarexclude` file is considered to be the root of the given patterns. It is only possible to match files in this directory and its subdirectories.

`\` is used to escape special glob characters. `?` matches any single character. `*` matches any character, including an empty string. `**` is used to match subdirectories. It can be used to, for example, exclude all files ending in `.tmp` within the directory or subdirectories with the following pattern `**/*.tmp`. `[...]` matches a single character from any of the provided characters within the brackets. `[!...]` does the complementary and matches any single character not contained within the brackets. It is also possible to specify ranges with two characters separated by `-`. For example, `[a-z]` matches any lowercase alphabetic character and `[0-9]` matches any one single digit.

The order of the glob match patterns defines whether a file is included or excluded, that is to say later entries override previous ones. This is also true for match patterns encountered deeper down the directory tree, which can override a previous exclusion. Be aware that excluded directories will **not** be read by the backup client. Thus, a `.pxarexclude` file in an excluded subdirectory will

have no effect. `.pxarexclude` files are treated as regular files and will be included in the backup archive.

For example, consider the following directory structure:

```
# ls -aR folder
folder/:
.  ..  .pxarexclude  subfolder0  subfolder1

folder/subfolder0:
.  ..  file0  file1  file2  file3  .pxarexclude

folder/subfolder1:
.  ..  file0  file1  file2  file3
```

The different `.pxarexclude` files contain the following:

```
# cat folder/.pxarexclude
/subfolder0/file1
/subfolder1/*
!/subfolder1/file2
```

```
# cat folder/subfolder0/.pxarexclude
file3
```

This would exclude `file1` and `file3` in `subfolder0` and all of `subfolder1` except `file2`.

Restoring this backup will result in:

```
ls -aR restored
restored/:
.  ..  .pxarexclude  subfolder0  subfolder1

restored/subfolder0:
.  ..  file0  file2  .pxarexclude

restored/subfolder1:
.  ..  file2
```

10.5 Encryption

Proxmox Backup supports client-side encryption with AES-256 in GCM mode. To set this up, you first need to create an encryption key:

```
# proxmox-backup-client key create my-backup.key
Encryption Key Password: **************
```

The key is password protected by default. If you do not need this extra protection, you can also create it without a password:

```
# proxmox-backup-client key create /path/to/my-backup.key --kdf none
```

Having created this key, it is now possible to create an encrypted backup, by passing the `--keyfile` parameter, with the path to the key file.

```
# proxmox-backup-client backup etc.pxar:/etc --keyfile /path/to/my-backup.key
Password: *********
Encryption Key Password: **************
...
```

Note: If you do not specify the name of the backup key, the key will be created in the default location `~/.config/proxmox-backup/encryption-key.json`. `proxmox-backup-client` will also search this location by default, in case the `--keyfile` parameter is not specified.

You can avoid entering the passwords by setting the environment variables `PBS_PASSWORD` and `PBS_ENCRYPTION_PASSWORD`.

10.5.1 Using a master key to store and recover encryption keys

You can also use `proxmox-backup-client key` to create an RSA public/private key pair, which can be used to store an encrypted version of the symmetric backup encryption key alongside each backup and recover it later.

To set up a master key:

1. Create an encryption key for the backup:

```
# proxmox-backup-client key create
creating default key at: "~/.config/proxmox-backup/encryption-key.json"
Encryption Key Password: **********
...
```

   The resulting file will be saved to `~/.config/proxmox-backup/encryption-key.json`.

2. Create an RSA public/private key pair:

```
# proxmox-backup-client key create-master-key
Master Key Password: *********
...
```

   This will create two files in your current directory, `master-public.pem` and `master-private.pem`.

3. Import the newly created `master-public.pem` public certificate, so that `proxmox-backup-client` can find and use it upon backup.

```
# proxmox-backup-client key import-master-pubkey /path/to/master-public.pem
Imported public master key to "~/.config/proxmox-backup/master-public.pem"
```

4. With all these files in place, run a backup job:

```
# proxmox-backup-client backup etc.pxar:/etc
```

   The key will be stored in your backup, under the name `rsa-encrypted.key`.

   Note: The `--keyfile` parameter can be excluded, if the encryption key is in the default path. If you specified another path upon creation, you must pass the `--keyfile` parameter.

5. To test that everything worked, you can restore the key from the backup:

```
# proxmox-backup-client restore /path/to/backup/ rsa-encrypted.key /path/to/target
```

   Note: You should not need an encryption key to extract this file. However, if a key exists at the default location (`~/.config/proxmox-backup/encryption-key.json`) the program will prompt you for an encryption key password. Simply moving `encryption-key.json` out of this directory will fix this issue.

6. Then, use the previously generated master key to decrypt the file:

```
# proxmox-backup-client key import-with-master-key /path/to/target --master-keyfile /
↪path/to/master-private.pem --encrypted-keyfile /path/to/rsa-encrypted.key
Master Key Password: ******
New Password: ******
Verify Password: ******
```

7. The target file will now contain the encryption key information in plain text. The success of this can be confirmed by passing the resulting `json` file, with the `--keyfile` parameter, when decrypting files from the backup.

Warning: Without their key, backed up files will be inaccessible. Thus, you should keep keys ordered and in a place that is separate from the contents being backed up. It can happen, for example, that you back up an entire system, using a key on that system. If the system then becomes inaccessible for any reason and needs to be restored, this will not be possible as the encryption key will be lost along with the broken system.

It is recommended that you keep your master key safe, but easily accessible, in order for quick disaster recovery. For this reason, the best place to store it is in your password manager, where it is immediately recoverable. As a backup to this, you should also save the key to a USB drive and store that in a secure place. This way, it is detached from any system, but is still easy to recover from, in case of emergency. Finally, in preparation for the worst case scenario, you should also consider keeping a paper copy of your master key locked away in a safe place. The `paperkey` subcommand can be used to create a QR encoded version of your master key. The following command sends the output of the `paperkey` command to a text file, for easy printing.

```
proxmox-backup-client key paperkey --output-format text > qrkey.txt
```

10.6 Restoring Data

The regular creation of backups is a necessary step to avoiding data loss. More importantly, however, is the restoration. It is good practice to perform periodic recovery tests to ensure that you can access the data in case of problems.

First, you need to find the snapshot which you want to restore. The snapshot list command provides a list of all the snapshots on the server:

```
# proxmox-backup-client snapshot list
```

snapshot	size	files
host/elsa/2019-12-03T09:30:15Z	51788646825	root.pxar catalog.pcat1 index.json
host/elsa/2019-12-03T09:35:01Z	51790622048	root.pxar catalog.pcat1 index.json

```
...
```

You can inspect the catalog to find specific files.

```
# proxmox-backup-client catalog dump host/elsa/2019-12-03T09:35:01Z
...
d "./root.pxar.didx/etc/cifs-utils"
l "./root.pxar.didx/etc/cifs-utils/idmap-plugin"
d "./root.pxar.didx/etc/console-setup"
...
```

The restore command lets you restore a single archive from the backup.

```
# proxmox-backup-client restore host/elsa/2019-12-03T09:35:01Z root.pxar /target/path/
```

To get the contents of any archive, you can restore the `index.json` file in the repository to the target path '-'. This will dump the contents to the standard output.

```
# proxmox-backup-client restore host/elsa/2019-12-03T09:35:01Z index.json -
```

10.6.1 Interactive Restores

If you only want to restore a few individual files, it is often easier to use the interactive recovery shell.

```
# proxmox-backup-client catalog shell host/elsa/2019-12-03T09:35:01Z root.pxar
Starting interactive shell
pxar:/ > ls
bin        boot       dev        etc        home       lib        lib32
...
```

The interactive recovery shell is a minimal command line interface that utilizes the metadata stored in the catalog to quickly list, navigate and search files in a file archive. To restore files, you can select them individually or match them with a glob pattern.

Using the catalog for navigation reduces the overhead considerably because only the catalog needs to be downloaded and, optionally, decrypted. The actual chunks are only accessed if the metadata in the catalog is not enough or for the actual restore.

Similar to common UNIX shells `cd` and `ls` are the commands used to change working directory and list directory contents in the archive. `pwd` shows the full path of the current working directory with respect to the archive root.

Being able to quickly search the contents of the archive is a commonly needed feature. That's where the catalog is most valuable. For example:

```
pxar:/ > find etc/**/*.txt --select
"/etc/X11/rgb.txt"
pxar:/ > list-selected
etc/**/*.txt
pxar:/ > restore-selected /target/path
...
```

This will find and print all files ending in `.txt` located in `etc/` or a subdirectory and add the corresponding pattern to the list for subsequent restores. `list-selected` shows these patterns and `restore-selected` finally restores all files in the archive matching the patterns to `/target/path` on the local host. This will scan the whole archive.

The `restore` command can be used to restore all the files contained within the backup archive. This is most helpful when paired with the `--pattern <glob>` option, as it allows you to restore all files matching a specific pattern. For example, if you wanted to restore configuration files located in `/etc`, you could do the following:

```
pxar:/ > restore target/ --pattern etc/**/*.conf
...
```

The above will scan trough all the directories below `/etc` and restore all files ending in `.conf`.

10.6.2 Mounting of Archives via FUSE

The *FUSE* implementation for the pxar archive allows you to mount a file archive as a read-only filesystem to a mountpoint on your host.

```
# proxmox-backup-client mount host/backup-client/2020-01-29T11:29:22Z root.pxar /mnt/
↪mountpoint
# ls /mnt/mountpoint
bin   dev  home  lib32  libx32      media  opt   root  sbin  sys  usr
boot  etc  lib   lib64  lost+found  mnt    proc  run   srv   tmp  var
```

This allows you to access the full contents of the archive in a seamless manner.

Note: As the FUSE connection needs to fetch and decrypt chunks from the backup server's datastore, this can cause some additional network and CPU load on your host, depending on the oper-

ations you perform on the mounted filesystem.

To unmount the filesystem use the `umount` command on the mountpoint:

```
# umount /mnt/mountpoint
```

10.7 Login and Logout

The client tool prompts you to enter the logon password as soon as you want to access the backup server. The server checks your credentials and responds with a ticket that is valid for two hours. The client tool automatically stores that ticket and uses it for further requests to this server.

You can also manually trigger this login/logout using the login and logout commands:

```
# proxmox-backup-client login
Password: **********
```

To remove the ticket, issue a logout:

```
# proxmox-backup-client logout
```

10.8 Changing the Owner of a Backup Group

By default, the owner of a backup group is the user which was used to originally create that backup group (or in the case of sync jobs, `root@pam`). This means that if a user `mike@pbs` created a backup, another user `john@pbs` can not be used to create backups in that same backup group. In case you want to change the owner of a backup, you can do so with the below command, using a user that has `Datastore.Modify` privileges on the datastore.

```
# proxmox-backup-client change-owner vm/103 john@pbs
```

This can also be done from within the web interface, by navigating to the *Content* section of the datastore that contains the backup group and selecting the user icon under the *Actions* column. Common cases for this could be to change the owner of a sync job from `root@pam`, or to repurpose a backup group.

10.9 Pruning and Removing Backups

You can manually delete a backup snapshot using the `forget` command:

```
# proxmox-backup-client snapshot forget <snapshot>
```

Caution: This command removes all archives in this backup snapshot. They will be inaccessible and unrecoverable.

Although manual removal is sometimes required, the `prune` command is normally used to systematically delete older backups. Prune lets you specify which backup snapshots you want to keep. The following retention options are available:

`--keep-last <N>` Keep the last <N> backup snapshots.

`--keep-hourly <N>` Keep backups for the last <N> hours. If there is more than one backup for a single hour, only the latest is kept.

--keep-daily <N> Keep backups for the last <N> days. If there is more than one backup for a single day, only the latest is kept.

--keep-weekly <N> Keep backups for the last <N> weeks. If there is more than one backup for a single week, only the latest is kept.

> **Note:** Weeks start on Monday and end on Sunday. The software uses the ISO week date system and handles weeks at the end of the year correctly.

--keep-monthly <N> Keep backups for the last <N> months. If there is more than one backup for a single month, only the latest is kept.

--keep-yearly <N> Keep backups for the last <N> years. If there is more than one backup for a single year, only the latest is kept.

The retention options are processed in the order given above. Each option only covers backups within its time period. The next option does not take care of already covered backups. It will only consider older backups.

Unfinished and incomplete backups will be removed by the prune command unless they are newer than the last successful backup. In this case, the last failed backup is retained.

```
# proxmox-backup-client prune <group> --keep-daily 7 --keep-weekly 4 --keep-monthly 3
```

You can use the `--dry-run` option to test your settings. This only shows the list of existing snapshots and what actions prune would take.

```
# proxmox-backup-client prune host/elsa --dry-run --keep-daily 1 --keep-weekly 3
```

snapshot	keep
host/elsa/2019-12-04T13:20:37Z	1
host/elsa/2019-12-03T09:35:01Z	0
host/elsa/2019-11-22T11:54:47Z	1
host/elsa/2019-11-21T12:36:25Z	0
host/elsa/2019-11-10T10:42:20Z	1

Note: Neither the `prune` command nor the `forget` command free space in the chunk-store. The chunk-store still contains the data blocks. To free space you need to perform *Garbage Collection*.

10.10 Garbage Collection

The `prune` command removes only the backup index files, not the data from the datastore. This task is left to the garbage collection command. It is recommended to carry out garbage collection on a regular basis.

The garbage collection works in two phases. In the first phase, all data blocks that are still in use are marked. In the second phase, unused data blocks are removed.

Note: This command needs to read all existing backup index files and touches the complete chunk-store. This can take a long time depending on the number of chunks and the speed of the underlying disks.

Note: The garbage collection will only remove chunks that haven't been used for at least one day (exactly 24h 5m). This grace period is necessary because chunks in use are marked by touching the chunk which updates the `atime` (access time) property. Filesystems are mounted with the `relatime` option by default. This results in a better performance by only updating the `atime` property if the last access has been at least 24 hours ago. The downside is, that touching a chunk within these 24 hours will not always update its `atime` property.

Chunks in the grace period will be logged at the end of the garbage collection task as *Pending removals*.

```
# proxmox-backup-client garbage-collect
starting garbage collection on store store2
Start GC phase1 (mark used chunks)
Start GC phase2 (sweep unused chunks)
percentage done: 1, chunk count: 219
percentage done: 2, chunk count: 453
...
percentage done: 99, chunk count: 21188
Removed bytes: 411368505
Removed chunks: 203
Original data bytes: 327160886391
Disk bytes: 52767414743 (16 %)
Disk chunks: 21221
Average chunk size: 2486565
TASK OK
```

10.11 Benchmarking

The backup client also comes with a benchmarking tool. This tool measures various metrics relating to compression and encryption speeds. If a Proxmox Backup repository (remote or local) is specified, the TLS upload speed will get measured too.

You can run a benchmark using the `benchmark` subcommand of `proxmox-backup-client`:

Note: The TLS speed test is only included if a *backup server repository is specified*.

```
# proxmox-backup-client benchmark
Uploaded 1517 chunks in 5 seconds.
Time per request: 3309 microseconds.
TLS speed: 1267.41 MB/s
SHA256 speed: 2066.73 MB/s
Compression speed: 775.11 MB/s
Decompress speed: 1233.35 MB/s
AES256/GCM speed: 3688.27 MB/s
Verify speed: 783.43 MB/s
```

Name	Value
TLS (maximal backup upload speed)	1267.41 MB/s (103%)
SHA256 checksum computation speed	2066.73 MB/s (102%)
ZStd level 1 compression speed	775.11 MB/s (103%)
ZStd level 1 decompression speed	1233.35 MB/s (103%)
Chunk verification speed	783.43 MB/s (103%)
AES256 GCM encryption speed	3688.27 MB/s (101%)

Note: The percentages given in the output table correspond to a comparison against a Ryzen 7 2700X.

You can also pass the `--output-format` parameter to output stats in `json`, rather than the default table format.

CHAPTER

ELEVEN

PROXMOX VE INTEGRATION

You need to define a new storage with type 'pbs' on your Proxmox VE node. The following example uses `store2` as storage name, and assumes the server address is `localhost`, and you want to connect as `user1@pbs`.

```
# pvesm add pbs store2 --server localhost --datastore store2
# pvesm set store2 --username user1@pbs --password <secret>
```

Note: If you would rather not pass your password as plain text, you can pass the `--password` parameter, without any arguments. This will cause the program to prompt you for a password upon entering the command.

If your backup server uses a self signed certificate, you need to add the certificate fingerprint to the configuration. You can get the fingerprint by running the following command on the backup server:

```
# proxmox-backup-manager cert info | grep Fingerprint
Fingerprint (sha256): 64:d3:ff:3a:50:38:53:5a:9b:f7:50:...:ab:fe
```

Please add that fingerprint to your configuration to establish a trust relationship:

```
# pvesm set store2 --fingerprint  64:d3:ff:3a:50:38:53:5a:9b:f7:50:...:ab:fe
```

After that you should be able to see storage status with:

```
# pvesm status --storage store2
Name             Type     Status           Total            Used       Available        %
store2            pbs     active      3905109820      1336687816      2568422004   34.23%
```

Having added the PBS datastore to Proxmox VE, you can backup VMs and containers in the same way you would for any other storage device within the environment (see PVE Admin Guide: Backup and Restore.

CHAPTER

TWELVE

PXAR COMMAND LINE TOOL

`pxar` is a command line utility to create and manipulate archives in the *Proxmox File Archive Format (.pxar)*. It is inspired by casync file archive format, which caters to a similar use-case. The `.pxar` format is adapted to fulfill the specific needs of the Proxmox Backup Server, for example, efficient storage of hardlinks. The format is designed to reduce storage space needed on the server by achieving a high level of deduplication.

12.1 Creating an Archive

Run the following command to create an archive of a folder named `source`:

```
# pxar create archive.pxar /path/to/source
```

This will create a new archive called `archive.pxar` with the contents of the `source` folder.

Note: `pxar` will not overwrite any existing archives. If an archive with the same name is already present in the target folder, the creation will fail.

By default, `pxar` will skip certain mountpoints and will not follow device boundaries. This design decision is based on the primary use case of creating archives for backups. It makes sense to not back up the contents of certain temporary or system specific files. To alter this behavior and follow device boundaries, use the `--all-file-systems` flag.

It is possible to exclude certain files and/or folders from the archive by passing the `--exclude` parameter with `gitignore`-style match patterns.

For example, you can exclude all files ending in `.txt` from the archive by running:

```
# pxar create archive.pxar /path/to/source --exclude '**/*.txt'
```

Be aware that the shell itself will try to expand all of the glob patterns before invoking `pxar`. In order to avoid this, all globs have to be quoted correctly.

It is possible to pass the `--exclude` parameter multiple times, in order to match more than one pattern. This allows you to use more complex file exclusion/inclusion behavior. However, it is recommended to use `.pxarexclude` files instead for such cases.

For example you might want to exclude all `.txt` files except for a specific one from the archive. This is achieved via the negated match pattern, prefixed by `!`. All the glob patterns are relative to the `source` directory.

```
# pxar create archive.pxar /path/to/source --exclude '**/*.txt' --exclude '!/folder/file.txt'
```

Note: The order of the glob match patterns matters as later ones override previous ones. Permutations of the same patterns lead to different results.

pxar will store the list of glob match patterns passed as parameters via the command line, in a file called .pxarexclude-cli at the root of the archive. If a file with this name is already present in the source folder during archive creation, this file is not included in the archive and the file containing the new patterns is added to the archive instead, the original file is not altered.

A more convenient and persistent way to exclude files from the archive is by placing the glob match patterns in .pxarexclude files. It is possible to create and place these files in any directory of the filesystem tree. These files must contain one pattern per line, again later patterns win over previous ones. The patterns control file exclusions of files present within the given directory or further below it in the tree. The behavior is the same as described in *Creating Backups*.

12.2 Extracting an Archive

An existing archive, archive.pxar, is extracted to a target directory with the following command:

```
# pxar extract archive.pxar /path/to/target
```

If no target is provided, the content of the archive is extracted to the current working directory.

In order to restore only parts of an archive, single files, and/or folders, it is possible to pass the corresponding glob match patterns as additional parameters or to use the patterns stored in a file:

```
# pxar extract etc.pxar /restore/target/etc --pattern '**/*.conf'
```

The above example restores all .conf files encountered in any of the sub-folders in the archive etc.pxar to the target /restore/target/etc. A path to the file containing match patterns can be specified using the --files-from parameter.

12.3 List the Contents of an Archive

To display the files and directories contained in an archive archive.pxar, run the following command:

```
# pxar list archive.pxar
```

This displays the full path of each file or directory with respect to the archives root.

12.4 Mounting an Archive

pxar allows you to mount and inspect the contents of an archive via FUSE. In order to mount an archive named archive.pxar to the mountpoint /mnt, run the command:

```
# pxar mount archive.pxar /mnt
```

Once the archive is mounted, you can access its content under the given mountpoint.

```
# cd /mnt
# ls
bin   dev  home  lib32  libx32      media  opt   root  sbin  sys  usr
boot  etc  lib   lib64  lost+found  mnt    proc  run   srv   tmp  var
```

CHAPTER

THIRTEEN

HOST SYSTEM ADMINISTRATION

Proxmox Backup is based on the famous Debian Linux distribution. That means that you have access to the whole world of Debian packages, and the base system is well documented. The Debian Administrator's Handbook is available online, and provides a comprehensive introduction to the Debian operating system.

A standard Proxmox Backup installation uses the default repositories from Debian, so you get bug fixes and security updates through that channel. In addition, we provide our own package repository to roll out all Proxmox related packages. This includes updates to some Debian packages when necessary.

We also deliver a specially optimized Linux kernel, where we enable all required virtualization and container features. That kernel includes drivers for ZFS, and several hardware drivers. For example, we ship Intel network card drivers to support their newest hardware.

The following sections will concentrate on backup related topics. They either explain things which are different on Proxmox Backup, or tasks which are commonly used on Proxmox Backup. For other topics, please refer to the standard Debian documentation.

13.1 ZFS on Linux

ZFS is a combined file system and logical volume manager designed by Sun Microsystems. There is no need to manually compile ZFS modules - all packages are included.

By using ZFS, it's possible to achieve maximum enterprise features with low budget hardware, but also high performance systems by leveraging SSD caching or even SSD only setups. ZFS can replace cost intense hardware raid cards by moderate CPU and memory load combined with easy management.

General ZFS advantages

- Easy configuration and management with GUI and CLI.
- Reliable
- Protection against data corruption
- Data compression on file system level
- Snapshots
- Copy-on-write clone
- Various raid levels: RAID0, RAID1, RAID10, RAIDZ-1, RAIDZ-2 and RAIDZ-3
- Can use SSD for cache
- Self healing
- Continuous integrity checking
- Designed for high storage capacities

- Asynchronous replication over network
- Open Source
- Encryption

13.1.1 Hardware

ZFS depends heavily on memory, so you need at least 8GB to start. In practice, use as much you can get for your hardware/budget. To prevent data corruption, we recommend the use of high quality ECC RAM.

If you use a dedicated cache and/or log disk, you should use an enterprise class SSD (e.g. Intel SSD DC S3700 Series). This can increase the overall performance significantly.

IMPORTANT: Do not use ZFS on top of hardware controller which has its own cache management. ZFS needs to directly communicate with disks. An HBA adapter is the way to go, or something like LSI controller flashed in `IT` mode.

13.1.2 ZFS Administration

This section gives you some usage examples for common tasks. ZFS itself is really powerful and provides many options. The main commands to manage ZFS are *zfs* and *zpool*. Both commands come with great manual pages, which can be read with:

```
# man zpool
# man zfs
```

Create a new zpool

To create a new pool, at least one disk is needed. The *ashift* should have the same sector-size (2 power of *ashift*) or larger as the underlying disk.

```
# zpool create -f -o ashift=12 <pool> <device>
```

Create a new pool with RAID-0

Minimum 1 disk

```
# zpool create -f -o ashift=12 <pool> <device1> <device2>
```

Create a new pool with RAID-1

Minimum 2 disks

```
# zpool create -f -o ashift=12 <pool> mirror <device1> <device2>
```

Create a new pool with RAID-10

Minimum 4 disks

```
# zpool create -f -o ashift=12 <pool> mirror <device1> <device2> mirror <device3> <device4>
```

Create a new pool with RAIDZ-1

Minimum 3 disks

```
# zpool create -f -o ashift=12 <pool> raidz1 <device1> <device2> <device3>
```

Create a new pool with RAIDZ-2

Minimum 4 disks

```
# zpool create -f -o ashift=12 <pool> raidz2 <device1> <device2> <device3> <device4>
```

Create a new pool with cache (L2ARC)

It is possible to use a dedicated cache drive partition to increase the performance (use SSD).

As *<device>* it is possible to use more devices, like it's shown in "Create a new pool with RAID*".

```
# zpool create -f -o ashift=12 <pool> <device> cache <cache_device>
```

Create a new pool with log (ZIL)

It is possible to use a dedicated cache drive partition to increase the performance (SSD).

As *<device>* it is possible to use more devices, like it's shown in "Create a new pool with RAID*".

```
# zpool create -f -o ashift=12 <pool> <device> log <log_device>
```

Add cache and log to an existing pool

If you have a pool without cache and log. First partition the SSD in 2 partition with *parted* or *gdisk*

Important: Always use GPT partition tables.

The maximum size of a log device should be about half the size of physical memory, so this is usually quite small. The rest of the SSD can be used as cache.

```
# zpool add -f <pool> log <device-part1> cache <device-part2>
```

Changing a failed device

```
# zpool replace -f <pool> <old device> <new device>
```

Changing a failed bootable device

Depending on how Proxmox Backup was installed it is either using *grub* or *systemd-boot* as bootloader.

The first steps of copying the partition table, reissuing GUIDs and replacing the ZFS partition are the same. To make the system bootable from the new disk, different steps are needed which depend on the bootloader in use.

```
# sgdisk <healthy bootable device> -R <new device>
# sgdisk -G <new device>
# zpool replace -f <pool> <old zfs partition> <new zfs partition>
```

Note: Use the *zpool status -v* command to monitor how far the resilvering process of the new disk has progressed.

With *systemd-boot*:

```
# pve-efiboot-tool format <new disk's ESP>
# pve-efiboot-tool init <new disk's ESP>
```

Note: *ESP* stands for EFI System Partition, which is setup as partition #2 on bootable disks setup by the {pve} installer since version 5.4. For details, see xref:sysboot_systemd_boot_setup[Setting up a new partition for use as synced ESP].

With *grub*:

Usually *grub.cfg* is located in */boot/grub/grub.cfg*

```
# grub-install <new disk>
# grub-mkconfig -o /path/to/grub.cfg
```

Activate E-Mail Notification

ZFS comes with an event daemon, which monitors events generated by the ZFS kernel module. The daemon can also send emails on ZFS events like pool errors. Newer ZFS packages ship the daemon in a separate package, and you can install it using *apt-get*:

```
# apt-get install zfs-zed
```

To activate the daemon it is necessary to edit */etc/zfs/zed.d/zed.rc* with your favorite editor, and uncomment the *ZED_EMAIL_ADDR* setting:

```
ZED_EMAIL_ADDR="root"
```

Please note Proxmox Backup forwards mails to *root* to the email address configured for the root user.

IMPORTANT: The only setting that is required is *ZED_EMAIL_ADDR*. All other settings are optional.

Limit ZFS Memory Usage

It is good to use at most 50 percent (which is the default) of the system memory for ZFS ARC to prevent performance shortage of the host. Use your preferred editor to change the configuration in */etc/modprobe.d/zfs.conf* and insert:

```
options zfs zfs_arc_max=8589934592
```

This example setting limits the usage to 8GB.

Important: If your root file system is ZFS you must update your initramfs every time this value changes:

```
# update-initramfs -u
```

SWAP on ZFS

Swap-space created on a zvol may generate some troubles, like blocking the server or generating a high IO load, often seen when starting a Backup to an external Storage.

We strongly recommend to use enough memory, so that you normally do not run into low memory situations. Should you need or want to add swap, it is preferred to create a partition on a physical disk and use it as swap device. You can leave some space free for this purpose in the advanced options of the installer. Additionally, you can lower the *swappiness* value. A good value for servers is 10:

```
# sysctl -w vm.swappiness=10
```

To make the swappiness persistent, open */etc/sysctl.conf* with an editor of your choice and add the following line:

```
vm.swappiness = 10
```

Table 1: Linux kernel *swappiness* parameter values :widths:auto

Value	Strategy
vm.swappiness = 0	The kernel will swap only to avoid an 'out of memory' condition
vm.swappiness = 1	Minimum amount of swapping without disabling it entirely.
vm.swappiness = 10	Sometimes recommended to improve performance when sufficient memory exists in a system.
vm.swappiness = 60	The default value.
vm.swappiness = 100	The kernel will swap aggressively.

ZFS Compression

To activate compression: .. code-block:: console

    # zpool set compression=lz4 <pool>

We recommend using the *lz4* algorithm, since it adds very little CPU overhead. Other algorithms such as *lzjb* and *gzip-N* (where *N* is an integer *1-9* representing the compression ratio, 1 is fastest and 9 is best compression) are also available. Depending on the algorithm and how compressible the data is, having compression enabled can even increase I/O performance.

You can disable compression at any time with: .. code-block:: console

    # zfs set compression=off <dataset>

Only new blocks will be affected by this change.

ZFS Special Device

Since version 0.8.0 ZFS supports *special* devices. A *special* device in a pool is used to store metadata, deduplication tables, and optionally small file blocks.

A *special* device can improve the speed of a pool consisting of slow spinning hard disks with a lot of metadata changes. For example workloads that involve creating, updating or deleting a large

number of files will benefit from the presence of a *special* device. ZFS datasets can also be configured to store whole small files on the *special* device which can further improve the performance. Use fast SSDs for the *special* device.

Important: The redundancy of the *special* device should match the one of the pool, since the *special* device is a point of failure for the whole pool.

Warning: Adding a *special* device to a pool cannot be undone!

Create a pool with *special* device and RAID-1:

```
# zpool create -f -o ashift=12 <pool> mirror <device1> <device2> special mirror <device3>
↪<device4>
```

Adding a *special* device to an existing pool with RAID-1:

```
# zpool add <pool> special mirror <device1> <device2>
```

ZFS datasets expose the *special_small_blocks=<size>* property. *size* can be *0* to disable storing small file blocks on the *special* device or a power of two in the range between *512B* to *128K*. After setting the property new file blocks smaller than *size* will be allocated on the *special* device.

Important: If the value for *special_small_blocks* is greater than or equal to the *recordsize* (default *128K*) of the dataset, *all* data will be written to the *special* device, so be careful!

Setting the *special_small_blocks* property on a pool will change the default value of that property for all child ZFS datasets (for example all containers in the pool will opt in for small file blocks).

Opt in for all file smaller than 4K-blocks pool-wide:

```
# zfs set special_small_blocks=4K <pool>
```

Opt in for small file blocks for a single dataset:

```
# zfs set special_small_blocks=4K <pool>/<filesystem>
```

Opt out from small file blocks for a single dataset:

```
# zfs set special_small_blocks=0 <pool>/<filesystem>
```

Troubleshooting

Corrupted cachefile

In case of a corrupted ZFS cachefile, some volumes may not be mounted during boot until mounted manually later.

For each pool, run:

```
# zpool set cachefile=/etc/zfs/zpool.cache POOLNAME
```

and afterwards update the *initramfs* by running:

```
# update-initramfs -u -k all
```

and finally reboot your node.

Sometimes the ZFS cachefile can get corrupted, and *zfs-import-cache.service* doesn't import the pools that aren't present in the cachefile.

Another workaround to this problem is enabling the *zfs-import-scan.service*, which searches and imports pools via device scanning (usually slower).

13.2 Service Daemons

13.2.1 `proxmox-backup-proxy`

This daemon exposes the whole Proxmox Backup Server API on TCP port 8007 using HTTPS. It runs as user `backup` and has very limited permissions. Operation requiring more permissions are forwarded to the local `proxmox-backup` service.

13.2.2 `proxmox-backup`

This daemon exposes the Proxmox Backup Server management API on `127.0.0.1:82`. It runs as `root` and has permission to do all privileged operations.

NOTE: The daemon listens to a local address only, so you cannot access it from outside. The `proxmox-backup-proxy` daemon exposes the API to the outside world.

CHAPTER

FOURTEEN

TECHNICAL OVERVIEW

14.1 Datastores

A Datastore is the logical place where *Backup Snapshots* and their chunks are stored. Snapshots consist of a manifest, blobs, dynamic- and fixed-indexes (see *Terminology*), and are stored in the following directory structure:

<datastore-root>/<type>/<id>/<time>/

The deduplication of datastores is based on reusing chunks, which are referenced by the indexes in a backup snapshot. This means that multiple indexes can reference the same chunks, reducing the amount of space needed to contain the data (even across backup snapshots).

14.2 Chunks

A chunk is some (possibly encrypted) data with a CRC-32 checksum at the end and a type marker at the beginning. It is identified by the SHA-256 checksum of its content.

To generate such chunks, backup data is split either into fixed-size or dynamically sized chunks. The same content will be hashed to the same checksum.

The chunks of a datastore are found in

<datastore-root>/.chunks/

This chunk directory is further subdivided by the first four byte of the chunks checksum, so the chunk with the checksum

a342e8151cbf439ce65f3df696b54c67a114982cc0aa751f2852c2f7acc19a8b

lives in

<datastore-root>/.chunks/a342/

This is done to reduce the number of files per directory, as having many files per directory can be bad for file system performance.

These chunk directories ('0000'-'ffff') will be preallocated when a datastore is created.

14.2.1 Fixed-sized Chunks

For block based backups (like VMs), fixed-sized chunks are used. The content (disk image), is split into chunks of the same length (typically 4 MiB).

This works very well for VM images, since the file system on the guest most often tries to allocate files in contiguous pieces, so new files get new blocks, and changing existing files changes only their own blocks.

As an optimization, VMs in Proxmox VE can make use of 'dirty bitmaps', which can track the changed blocks of an image. Since these bitmap are also a representation of the image split into chunks, there is a direct relation between dirty blocks of the image and chunks which need to get uploaded, so only modified chunks of the disk have to be uploaded for a backup.

Since the image is always split into chunks of the same size, unchanged blocks will result in identical checksums for those chunks, so such chunks do not need to be backed up again. This way storage snapshots are not needed to find the changed blocks.

For consistency, Proxmox VE uses a QEMU internal snapshot mechanism, that does not rely on storage snapshots either.

14.2.2 Dynamically sized Chunks

If one does not want to backup block-based systems but rather file-based systems, using fixed-sized chunks is not a good idea, since every time a file would change in size, the remaining data gets shifted around and this would result in many chunks changing, reducing the amount of deduplication.

To improve this, Proxmox Backup Server uses dynamically sized chunks instead. Instead of splitting an image into fixed sizes, it first generates a consistent file archive (*pxar*) and uses a rolling hash over this on-the-fly generated archive to calculate chunk boundaries.

We use a variant of Buzhash which is a cyclic polynomial algorithm. It works by continuously calculating a checksum while iterating over the data, and on certain conditions it triggers a hash boundary.

Assuming that most files of the system that is to be backed up have not changed, eventually the algorithm triggers the boundary on the same data as a previous backup, resulting in chunks that can be reused.

14.2.3 Encrypted Chunks

Encrypted chunks are a special case. Both fixed- and dynamically sized chunks can be encrypted, and they are handled in a slightly different manner than normal chunks.

The hashes of encrypted chunks are calculated not with the actual (encrypted) chunk content, but with the plaintext content concatenated with the encryption key. This way, two chunks of the same data encrypted with different keys generate two different checksums and no collisions occur for multiple encryption keys.

This is done to speed up the client part of the backup, since it only needs to encrypt chunks that are actually getting uploaded. Chunks that exist already in the previous backup, do not need to be encrypted and uploaded.

14.3 Caveats and Limitations

14.3.1 Notes on hash collisions

Every hashing algorithm has a chance to produce collisions, meaning two (or more) inputs generate the same checksum. For SHA-256, this chance is negligible. To calculate such a collision, one can use the ideas of the 'birthday problem' from probability theory. For big numbers, this is actually infeasible to calculate with regular computers, but there is a good approximation:

$$p(n, d) = 1 - e^{-n^2/(2d)}$$

Where *n* is the number of tries, and *d* is the number of possibilities. For a concrete example lets assume a large datastore of 1 PiB, and an average chunk size of 4 MiB. That means $n = 268435456$ tries, and $d = 2^{256}$ possibilities. Inserting those values in the formula from earlier you will see that the probability of a collision in that scenario is:

$$3.1115 * 10^{-61}$$

For context, in a lottery game of guessing 6 out of 45, the chance to correctly guess all 6 numbers is only $1.2277 * 10^{-7}$, that means the chance of collission is about the same as winning 13 such lotto games *in a row*.

In conclusion, it is extremely unlikely that such a collision would occur by accident in a normal datastore.

Additionally, SHA-256 is prone to length extension attacks, but since there is an upper limit for how big the chunk are, this is not a problem, since a potential attacker cannot arbitrarily add content to the data beyond that limit.

14.3.2 File-based Backup

Since dynamically sized chunks (for file-based backups) are created on a custom archive format (pxar) and not over the files directly, there is no relation between files and the chunks. This means that the Proxmox Backup client has to read all files again for every backup, otherwise it would not be possible to generate a consistent independent pxar archive where the original chunks can be reused. Note that there will be still only new or change chunks be uploaded.

14.3.3 Verification of encrypted chunks

For encrypted chunks, only the checksum of the original (plaintext) data is available, making it impossible for the server (without the encryption key), to verify its content against it. Instead only the CRC-32 checksum gets checked.

CHAPTER

FIFTEEN

FAQ

15.1 What distribution is Proxmox Backup Server (PBS) based on?

Proxmox Backup Server is based on Debian GNU/Linux.

15.2 Which platforms are supported as a backup source (client)?

The client tool works on most modern Linux systems, meaning you are not limited to Debian-based backups.

15.3 Will Proxmox Backup Server run on a 32-bit processor?

Proxmox Backup Server only supports 64-bit CPUs (AMD or Intel). There are no future plans to support 32-bit processors.

15.4 How long will my Proxmox Backup Server version be supported?

Proxmox Backup Version	Debian Version	First Release	Debian EOL	Proxmox Backup EOL
Proxmox Backup 1.x	Debian 10 (Buster)	2020-11	tba	tba

15.5 Can I copy or synchronize my datastore to another location?

Proxmox Backup Server allows you to copy or synchronize datastores to other locations, through the use of *Remotes* and *Sync Jobs*. *Remote* is the term given to a separate server, which has a datastore that can be synced to a local store. A *Sync Job* is the process which is used to pull the contents of a datastore from a *Remote* to a local datastore.

15.6 Can Proxmox Backup Server verify data integrity of a backup archive?

Proxmox Backup Server uses a built-in SHA-256 checksum algorithm, to ensure data integrity. Within each backup, a manifest file (index.json) is created, which contains a list of all the backup files, along with their sizes and checksums. This manifest file is used to verify the integrity of each backup.

15.7 When backing up to remote servers, do I have to trust the remote server?

Proxmox Backup Server transfers data via Transport Layer Security (TLS) and additionally supports client-side encryption. This means that data is transferred securely and can be encrypted before it reaches the server. Thus, in the event that an attacker gains access to the server or any point of the network, they will not be able to read the data.

Note: Encryption is not enabled by default. To set up encryption, see the Encryption section of the Proxmox Backup Server Administration Guide.

15.8 Is the backup incremental/deduplicated?

With Proxmox Backup Server, backups are sent incremental and data is deduplicated on the server. This minimizes both the storage consumed and the network impact.

APPENDIX

A

COMMAND SYNTAX

A.1 proxmox-backup-client

`proxmox-backup-client backup {<backupspec>} [OPTIONS]`

Create (host) backup.

<backupspec> [<array>] List of backup source specifications ([<label.ext>:<path>] ...)

Optional parameters:

--all-file-systems <boolean> Include all mounted subdirectories.

--backup-id <string> Backup ID.

--backup-time <integer> (1547797308 - N) Backup time (Unix epoch.)

--backup-type vm|ct|host Backup type.

--chunk-size <integer> (64 - 4096) (default=4096) Chunk size in KB. Must be a power of 2.

--crypt-mode none|encrypt|sign-only (default=encrypt) Defines whether data is encrypted (using an AEAD cipher), only signed, or neither.

--entries-max <integer> (default=1048576) Max number of entries to hold in memory.

--exclude <array> List of paths or patterns for matching files to exclude.

--include-dev <array> Include mountpoints with same st_dev number (see `man fstat`) as specified files.

--keyfd <integer> (0 - N) Pass an encryption key via an already opened file descriptor.

--keyfile <string> Path to encryption key. All data will be encrypted using this key.

--master-pubkey-fd <integer> (0 - N) Pass a master public key via an already opened file descriptor.

--master-pubkey-file <string> Path to master public key. The encryption key used for a backup will be encrypted using this key and appended to the backup.

--repository <string> Repository URL.

--skip-lost-and-found <boolean> Skip lost+found directory.

--verbose <boolean> Verbose output.

`proxmox-backup-client benchmark [OPTIONS]`

Run benchmark tests

Optional parameters:

--keyfile <string> Path to encryption key. All data will be encrypted using this key.

--output-format text|json|json-pretty Output format.

--repository <string> Repository URL.

--verbose <boolean> Verbose output.

`proxmox-backup-client catalog dump <snapshot> [OPTIONS]`

Dump catalog.

<snapshot> [<string>] Snapshot path.

Optional parameters:

--keyfd <integer> (0 - N) Pass an encryption key via an already opened file descriptor.

--keyfile <string> Path to encryption key.

--repository <string> Repository URL.

`proxmox-backup-client catalog shell <snapshot> <archive-name> [OPTIONS]`

Shell to interactively inspect and restore snapshots.

<snapshot> [<string>] Group/Snapshot path.

<archive-name> [<string>] Backup archive name.

Optional parameters:

--keyfd <integer> (0 - N) Pass an encryption key via an already opened file descriptor.

--keyfile <string> Path to encryption key.

--repository <string> Repository URL.

`proxmox-backup-client change-owner <group> <new-owner> [OPTIONS]`

Change owner of a backup group

<group> [<string>] Backup group.

<new-owner> [<string>] Authentication ID

Optional parameters:

--repository <string> Repository URL.

`proxmox-backup-client garbage-collect [OPTIONS]`

Start garbage collection for a specific repository.

Optional parameters:

--output-format text|json|json-pretty Output format.

--repository <string> Repository URL.

`proxmox-backup-client help [{<command>}] [OPTIONS]`

Get help about specified command (or sub-command).

<command> [<array>] Command. This may be a list in order to spefify nested sub-commands.

Optional parameters:

--verbose <boolean> Verbose help.

`proxmox-backup-client key change-passphrase [<path>] [OPTIONS]`

Change the encryption key's password.

<path> [<string>] Key file. Without this the default key's password will be changed.

Optional parameters:

--hint <string> Password hint.

--kdf none|scrypt|pbkdf2 (default=scrypt) Key derivation function for password protected encryption keys.

`proxmox-backup-client key create [<path>] [OPTIONS]`

Create a new encryption key.

<path> [<string>] Output file. Without this the key will become the new default encryption key.

Optional parameters:

--hint <string> Password hint.

--kdf none|scrypt|pbkdf2 (default=scrypt) Key derivation function for password protected encryption keys.

`proxmox-backup-client key create-master-key`

Create an RSA public/private key pair used to put an encrypted version of the symmetric backup encryption key onto the backup server along with each backup.

`proxmox-backup-client key import-master-pubkey <path>`

Import an RSA public key used to put an encrypted version of the symmetric backup encryption key onto the backup server along with each backup.

The imported key will be used as default master key for future invocations by the same local user.

<path> [<string>] Path to the PEM formatted RSA public key.

`proxmox-backup-client key import-with-master-key [<path>] --encrypted-keyfile <string> --master-keyfile <string> [OPTIONS]`

Import an encrypted backup of an encryption key using a (private) master key.

<path> [<string>] Output file. Without this the key will become the new default encryption key.

--encrypted-keyfile <string> RSA-encrypted keyfile to import.

--master-keyfile <string> (Private) master key to use.

Optional parameters:

--hint <string> Password hint.

--kdf none|scrypt|pbkdf2 (default=scrypt) Key derivation function for password protected encryption keys.

`proxmox-backup-client key paperkey [<path>] [OPTIONS]`

Generate a printable, human readable text file containing the encryption key.

This also includes a scanable QR code for fast key restore.

<path> [<string>] Key file. Without this the default key's will be used.

Optional parameters:

--output-format text|html Paperkey output format

--subject <string> Include the specified subject as title text.

---

proxmox-backup-client key show [<path>] [OPTIONS]

Print the encryption key's metadata.

<path> [<string>] Key file. Without this the default key's metadata will be shown.

Optional parameters:

--output-format text|json|json-pretty Output format.

---

proxmox-backup-client key show-master-pubkey [<path>] [OPTIONS]

List information about master key

<path> [<string>] Path to the PEM formatted RSA public key. Default location will be used if not specified.

Optional parameters:

--output-format text|json|json-pretty Output format.

---

proxmox-backup-client list [OPTIONS]

List backup groups.

Optional parameters:

--output-format text|json|json-pretty Output format.

--repository <string> Repository URL.

---

proxmox-backup-client login [OPTIONS]

Try to login. If successful, store ticket.

Optional parameters:

--repository <string> Repository URL.

---

proxmox-backup-client logout [OPTIONS]

Logout (delete stored ticket).

Optional parameters:

--repository <string> Repository URL.

---

proxmox-backup-client map <snapshot> <archive-name> [OPTIONS]

Map a drive image from a VM backup to a local loopback device. Use 'unmap' to undo. WARNING: Only do this with *trusted* backups!

<snapshot> [<string>] Group/Snapshot path.

<archive-name> [<string>] Backup archive name.

Optional parameters:

--keyfile <string> Path to encryption key.

--repository <string> Repository URL.

--verbose <boolean> (default=false) Verbose output and stay in foreground.

`proxmox-backup-client mount <snapshot> <archive-name> <target> [OPTIONS]`

Mount pxar archive.

<snapshot> [<string>] Group/Snapshot path.

<archive-name> [<string>] Backup archive name.

<target> [<string>] Target directory path.

Optional parameters:

--keyfile <string> Path to encryption key.

--repository <string> Repository URL.

--verbose <boolean> (default=false) Verbose output and stay in foreground.

`proxmox-backup-client prune <group> [OPTIONS]`

Prune a backup repository.

<group> [<string>] Backup group.

Optional parameters:

--dry-run <boolean> Just show what prune would do, but do not delete anything.

--keep-daily <integer> (1 - N) Number of daily backups to keep.

--keep-hourly <integer> (1 - N) Number of hourly backups to keep.

--keep-last <integer> (1 - N) Number of backups to keep.

--keep-monthly <integer> (1 - N) Number of monthly backups to keep.

--keep-weekly <integer> (1 - N) Number of weekly backups to keep.

--keep-yearly <integer> (1 - N) Number of yearly backups to keep.

--output-format text|json|json-pretty Output format.

--quiet <boolean> Minimal output - only show removals.

--repository <string> Repository URL.

`proxmox-backup-client restore <snapshot> <archive-name> <target> [OPTIONS]`

Restore backup repository.

<snapshot> [<string>] Group/Snapshot path.

<archive-name> [<string>] Backup archive name.

<target> [<string>] Target directory path. Use '-' to write to standard output.

We do not extract '.pxar' archives when writing to standard output.

Optional parameters:

--allow-existing-dirs <boolean> Do not fail if directories already exists.

--crypt-mode none|encrypt|sign-only (default=encrypt) Defines whether data is encrypted (using an AEAD cipher), only signed, or neither.

--keyfd <integer> (0 - N) Pass an encryption key via an already opened file descriptor.

--keyfile <string> Path to encryption key. All data will be encrypted using this key.

--repository <string> Repository URL.

`proxmox-backup-client snapshot files <snapshot> [OPTIONS]`

List snapshot files.

<snapshot> [<string>] Snapshot path.

Optional parameters:

--output-format text|json|json-pretty Output format.

--repository <string> Repository URL.

`proxmox-backup-client snapshot forget <snapshot> [OPTIONS]`

Forget (remove) backup snapshots.

<snapshot> [<string>] Snapshot path.

Optional parameters:

--repository <string> Repository URL.

`proxmox-backup-client snapshot list [<group>] [OPTIONS]`

List backup snapshots.

<group> [<string>] Backup group.

Optional parameters:

--output-format text|json|json-pretty Output format.

--repository <string> Repository URL.

`proxmox-backup-client snapshot notes show <snapshot> [OPTIONS]`

Show notes

<snapshot> [<string>] Snapshot path.

Optional parameters:

--output-format text|json|json-pretty Output format.

--repository <string> Repository URL.

`proxmox-backup-client snapshot notes update <snapshot> <notes> [OPTIONS]`

Update Notes

<snapshot> [<string>] Snapshot path.

<notes> [<string>] The Notes.

Optional parameters:

--repository <string> Repository URL.

`proxmox-backup-client snapshot upload-log <snapshot> <logfile> [OPTIONS]`

Upload backup log file.

<snapshot> [<string>] Group/Snapshot path.

<logfile> [<string>] The path to the log file you want to upload.

Optional parameters:

--crypt-mode none|encrypt|sign-only (default=encrypt) Defines whether data is encrypted (using an AEAD cipher), only signed, or neither.

--keyfd <integer> (0 - N) Pass an encryption key via an already opened file descriptor.

--keyfile <string> Path to encryption key. All data will be encrypted using this key.

--repository <string> Repository URL.

`proxmox-backup-client status [OPTIONS]`

Get repository status.

Optional parameters:

--output-format text|json|json-pretty Output format.

--repository <string> Repository URL.

`proxmox-backup-client task list [OPTIONS]`

List running server tasks for this repo user

Optional parameters:

--all <boolean> Also list stopped tasks.

--limit <integer> (1 - 1000) (default=50) The maximal number of tasks to list.

--output-format text|json|json-pretty Output format.

--repository <string> Repository URL.

`proxmox-backup-client task log <upid> [OPTIONS]`

Display the task log.

<upid> [<string>] Unique Process/Task ID.

Optional parameters:

--repository <string> Repository URL.

`proxmox-backup-client task stop <upid> [OPTIONS]`

Try to stop a specific task.

<upid> [<string>] Unique Process/Task ID.

Optional parameters:

--repository <string> Repository URL.

`proxmox-backup-client unmap [<name>]`

Unmap a loop device mapped with 'map' and release all resources.

<name> [<string>] Archive name, path to loopdev (/dev/loopX) or loop device number. Omit to list all current mappings and force cleaning up leftover instances.

`proxmox-backup-client version [OPTIONS]`

Show client and optional server version

Optional parameters:

--output-format text|json|json-pretty Output format.

--repository <string> Repository URL.

A.1.1 Catalog Shell Commands

Those command are available when you start an interactive restore shell:

```
proxmox-backup-client shell <snapshot> <name.pxar>
```

`cd [<path>]`

Change the current working directory to the new directory

<path> [<string>] target path.

`clear-selected`

Clear the list of files selected for restore.

`deselect <path>`

Deselect an entry for restore.

This will return an error if the entry was not found in the list of entries selected for restore.

<path> [<string>] path to entry to remove from list.

`exit`

Exit the shell

`find <pattern> [OPTIONS]`

Find entries in the catalog matching the given match pattern.

<pattern> [<string>] Match pattern for matching files in the catalog.

Optional parameters:

--select <boolean> (default=false) Add matching filenames to list for restore.

`help [{<command>}] [OPTIONS]`

Get help about specified command (or sub-command).

<command> [<array>] Command. This may be a list in order to spefify nested sub-commands.

Optional parameters:

--verbose <boolean> Verbose help.

`list-selected [OPTIONS]`

List entries currently selected for restore.

Optional parameters:

`--patterns <boolean> (default=false)` List match patterns instead of the matching files.

`ls [<path>]`

List the content of working directory or given path.

`<path>` [`<string>`] target path.

`pwd`

List the current working directory.

`restore <target> [OPTIONS]`

Restore the sub-archive given by the current working directory to target.

By further providing a pattern, the restore can be limited to a narrower subset of this sub-archive. If pattern is not present or empty, the full archive is restored to target.

`<target>` [`<string>`] target path for restore on local filesystem.

Optional parameters:

`--pattern <string>` match pattern to limit files for restore.

`restore-selected <target>`

Restore the selected entries to the given target path.

Target must not exist on the clients filesystem.

`<target>` [`<string>`] target path for restore on local filesystem.

`select <path>`

Select an entry for restore.

This will return an error if the entry is already present in the list or if an invalid path was provided.

`<path>` [`<string>`] target path.

`stat <path>`

Read the metadata for a given directory entry.

This is expensive because the data has to be read from the pxar archive, which means reading over the network.

`<path>` [`<string>`] target path.

A.2 proxmox-backup-manager

`proxmox-backup-manager acl list [OPTIONS]`

Access Control list.

Optional parameters:

--output-format text|json|json-pretty Output format.

`proxmox-backup-manager acl update <path> <role> [OPTIONS]`

Update Access Control List (ACLs).

<path> [<string>] Access control path.

<role> [<role>] Enum representing roles via their [PRIVILEGES] combination.

> Since privileges are implemented as bitflags, each unique combination of privileges maps to a single, unique *u64* value that is used in this enum definition.

Optional parameters:

--auth-id <string> Authentication ID

--delete <boolean> Remove permissions (instead of adding it).

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

--group <string> Group ID

--propagate <boolean> (default=true) Allow to propagate (inherit) permissions.

`proxmox-backup-manager cert info`

Display node certificate information.

`proxmox-backup-manager cert update [OPTIONS]`

Update node certificates and generate all needed files/directories.

Optional parameters:

--force <boolean> Force generation of new SSL certifate.

`proxmox-backup-manager datastore create <name> <path> [OPTIONS]`

Create new datastore config.

<name> [<string>] Datastore name.

<path> [<string>] Directory name

Optional parameters:

--comment <string> Comment (single line).

--gc-schedule <calendar-event> Run garbage collection job at specified schedule.

--keep-daily <integer> (1 - N) Number of daily backups to keep.

--keep-hourly <integer> (1 - N) Number of hourly backups to keep.

--keep-last <integer> (1 - N) Number of backups to keep.

--keep-monthly <integer> (1 - N) Number of monthly backups to keep.

--keep-weekly <integer> (1 - N) Number of weekly backups to keep.

--keep-yearly <integer> (1 - N) Number of yearly backups to keep.

--notify [[gc=<enum>] [,sync=<enum>] [,verify=<enum>]] Datastore notification setting

--notify-user <string> User ID

--prune-schedule <calendar-event> Run prune job at specified schedule.

`proxmox-backup-manager datastore list [OPTIONS]`

Datastore list.

Optional parameters:

--output-format text|json|json-pretty Output format.

`proxmox-backup-manager datastore remove <name> [OPTIONS]`

Remove a datastore configuration.

<name> [<string>] Datastore name.

Optional parameters:

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

`proxmox-backup-manager datastore show <name> [OPTIONS]`

Show datastore configuration

<name> [<string>] Datastore name.

Optional parameters:

--output-format text|json|json-pretty Output format.

`proxmox-backup-manager datastore update <name> [OPTIONS]`

Update datastore config.

<name> [<string>] Datastore name.

Optional parameters:

--comment <string> Comment (single line).

--delete <array> List of properties to delete.

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

--gc-schedule <calendar-event> Run garbage collection job at specified schedule.

--keep-daily <integer> (1 - N) Number of daily backups to keep.

--keep-hourly <integer> (1 - N) Number of hourly backups to keep.

--keep-last <integer> (1 - N) Number of backups to keep.

--keep-monthly <integer> (1 - N) Number of monthly backups to keep.

--keep-weekly <integer> (1 - N) Number of weekly backups to keep.

--keep-yearly <integer> (1 - N) Number of yearly backups to keep.

--notify [[gc=<enum>] [,sync=<enum>] [,verify=<enum>]] Datastore notification setting

--notify-user <string> User ID

--prune-schedule <calendar-event> Run prune job at specified schedule.

--verify-new <boolean> (default=false) If enabled, all new backups will be verified right after completion.

`proxmox-backup-manager disk fs create <name> --disk <string> [OPTIONS]`

Create a Filesystem on an unused disk. Will be mounted under '/mnt/datastore/<name>'.

<name> [<string>] Datastore name.

--disk <string> Block device name (/sys/block/<name>).

Optional parameters:

--add-datastore <boolean> Configure a datastore using the directory.

`--filesystem ext4|xfs`

`proxmox-backup-manager disk fs list [OPTIONS]`

List systemd datastore mount units.

Optional parameters:

--output-format text|json|json-pretty Output format.

`proxmox-backup-manager disk initialize <disk> [OPTIONS]`

Initialize empty Disk with GPT

<disk> [<string>] Block device name (/sys/block/<name>).

Optional parameters:

--uuid <string> UUID for the GPT table.

`proxmox-backup-manager disk list [OPTIONS]`

Local disk list.

Optional parameters:

--output-format text|json|json-pretty Output format.

`proxmox-backup-manager disk smart-attributes <disk> [OPTIONS]`

Show SMART attributes.

<disk> [<string>] Block device name (/sys/block/<name>).

Optional parameters:

--output-format text|json|json-pretty Output format.

```
proxmox-backup-manager disk zpool create <name> --devices [<string>, ...]
--raidlevel single|mirror|raid10|raidz|raidz2|raidz3 [OPTIONS]
```

create a zfs pool

<name> [<string>] Datastore name.

--devices [<string>, ...] A list of disk names, comma separated.

--raidlevel single|mirror|raid10|raidz|raidz2|raidz3 The ZFS RAID level to use.

Optional parameters:

--add-datastore <boolean> Configure a datastore using the zpool.

--ashift <integer> (9 - 16) (default=12) Pool sector size exponent.

--compression gzip|lz4|lzjb|zle|on|off (default=On) The ZFS compression algorithm to use.

`proxmox-backup-manager disk zpool list [OPTIONS]`

Local zfs pools.

Optional parameters:

--output-format text|json|json-pretty Output format.

`proxmox-backup-manager dns get [OPTIONS]`

Read DNS settings

Optional parameters:

--output-format text|json|json-pretty Output format.

`proxmox-backup-manager dns set [OPTIONS]`

Update DNS settings.

Optional parameters:

--delete <array> List of properties to delete.

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

--dns1 <string> First name server IP address.

--dns2 <string> Second name server IP address.

--dns3 <string> Third name server IP address.

--search <string> Search domain for host-name lookup.

`proxmox-backup-manager garbage-collection start <store> [OPTIONS]`

Start garbage collection for a specific datastore.

<store> [<string>] Datastore name.

Optional parameters:

--output-format text|json|json-pretty Output format.

`proxmox-backup-manager garbage-collection status <store> [OPTIONS]`

Show garbage collection status for a specific datastore.

<store> [<string>] Datastore name.

Optional parameters:

--output-format text|json|json-pretty Output format.

`proxmox-backup-manager help [{<command>}] [OPTIONS]`

Get help about specified command (or sub-command).

<command> [<array>] Command. This may be a list in order to spefify nested sub-commands.

Optional parameters:

--verbose <boolean> Verbose help.

`proxmox-backup-manager network changes`

Show pending configuration changes (diff)

`proxmox-backup-manager network create <iface> [OPTIONS]`

Create network interface configuration.

<iface> [<string>] Network interface name.

Optional parameters:

--autostart <boolean> Autostart interface.

--bond-primary <string> Network interface name.

--bond_mode balance-rr|active-backup|balance-xor|broadcast|802.3ad|balance-tlb|balance-a
Linux Bond Mode

--bond_xmit_hash_policy layer2|layer2+3|layer3+4 Bond Transmit Hash Policy for LACP (802.3ad)

--bridge_ports [<string>, ...] A list of network devices, comma separated.

--bridge_vlan_aware <boolean> Enable bridge vlan support.

--cidr <string> IPv4 address with netmask (CIDR notation).

--cidr6 <string> IPv6 address with netmask (CIDR notation).

--comments <string> Comments (inet, may span multiple lines)

--comments6 <string> Comments (inet5, may span multiple lines)

--gateway <string> IPv4 address.

--gateway6 <string> IPv6 address.

--method manual|static|dhcp|loopback Interface configuration method

--method6 manual|static|dhcp|loopback Interface configuration method

--mtu <integer> (46 - 65535) (default=1500) Maximum Transmission Unit.

--slaves [<string>, ...] A list of network devices, comma separated.

--type loopback|eth|bridge|bond|vlan|alias|unknown Network interface type

---

`proxmox-backup-manager network list [OPTIONS]`

Network device list.

Optional parameters:

--output-format text|json|json-pretty Output format.

`proxmox-backup-manager network reload`

Reload network configuration (requires ifupdown2).

`proxmox-backup-manager network remove <iface> [OPTIONS]`

Remove network interface configuration.

<iface> [<string>] Network interface name.

Optional parameters:

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

`proxmox-backup-manager network revert`

Revert network configuration (rm /etc/network/interfaces.new).

`proxmox-backup-manager network update <iface> [OPTIONS]`

Update network interface config.

<iface> [<string>] Network interface name.

Optional parameters:

--autostart <boolean> Autostart interface.

--bond-primary <string> Network interface name.

--bond_mode balance-rr|active-backup|balance-xor|broadcast|802.3ad|balance-tlb|balance-a
Linux Bond Mode

--bond_xmit_hash_policy layer2|layer2+3|layer3+4 Bond Transmit Hash Policy for LACP (802.3ad)

--bridge_ports [<string>, ...] A list of network devices, comma separated.

--bridge_vlan_aware <boolean> Enable bridge vlan support.

--cidr <string> IPv4 address with netmask (CIDR notation).

--cidr6 <string> IPv6 address with netmask (CIDR notation).

--comments <string> Comments (inet, may span multiple lines)

--comments6 <string> Comments (inet5, may span multiple lines)

--delete <array> List of properties to delete.

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

--gateway <string> IPv4 address.

--gateway6 <string> IPv6 address.

--method manual|static|dhcp|loopback Interface configuration method

--method6 manual|static|dhcp|loopback Interface configuration method

--mtu <integer> (46 - 65535) (default=1500) Maximum Transmission Unit.

--slaves [<string>, ...] A list of network devices, comma separated.

--type loopback|eth|bridge|bond|vlan|alias|unknown Network interface type

---

`proxmox-backup-manager pull <remote> <remote-store> <local-store> [OPTIONS]`

Sync datastore from another repository

<remote> [<string>] Remote ID.

<remote-store> [<string>] Datastore name.

<local-store> [<string>] Datastore name.

Optional parameters:

--output-format text|json|json-pretty Output format.

--remove-vanished <boolean> (default=true) Delete vanished backups. This remove the local copy if the remote backup was deleted.

`proxmox-backup-manager remote create <name> --auth-id <string> --host <string> --password <string> [OPTIONS]`

Create new remote.

<name> [<string>] Remote ID.

--auth-id <string> Authentication ID

--host <string> DNS name or IP address.

--password <string> Password or auth token for remote host.

Optional parameters:

--comment <string> Comment (single line).

--fingerprint <string> X509 certificate fingerprint (sha256).

--port <integer> (default=8007) The (optional) port.

`proxmox-backup-manager remote list [OPTIONS]`

List configured remotes.

Optional parameters:

--output-format text|json|json-pretty Output format.

`proxmox-backup-manager remote remove <name> [OPTIONS]`

Remove a remote from the configuration file.

<name> [<string>] Remote ID.

Optional parameters:

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

`proxmox-backup-manager remote show <name> [OPTIONS]`

Show remote configuration

<name> [<string>] Remote ID.

Optional parameters:

--output-format text|json|json-pretty Output format.

`proxmox-backup-manager remote update <name> [OPTIONS]`

Update remote configuration.

<name> [<string>] Remote ID.

Optional parameters:

--auth-id <string> Authentication ID

--comment <string> Comment (single line).

--delete <array> List of properties to delete.

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

--fingerprint <string> X509 certificate fingerprint (sha256).

--host <string> DNS name or IP address.

--password <string> Password or auth token for remote host.

--port <integer> The (optional) port.

`proxmox-backup-manager report`

System report

`proxmox-backup-manager subscription get [OPTIONS]`

Read subscription info.

Optional parameters:

--output-format text|json|json-pretty Output format.

`proxmox-backup-manager subscription remove`

Delete subscription info.

`proxmox-backup-manager subscription set <key>`

Set a subscription key and check it.

<key> [<string>] Proxmox Backup Server subscription key.

`proxmox-backup-manager subscription update [OPTIONS]`

Check and update subscription status.

Optional parameters:

--force <boolean> (default=false) Always connect to server, even if information in cache is up to date.

`proxmox-backup-manager sync-job create <id> --remote <string> --remote-store <string> --store <string> [OPTIONS]`

Create a new sync job.

<id> [<string>] Job ID.

--remote <string> Remote ID.

--remote-store <string> Datastore name.

--store <string> Datastore name.

Optional parameters:

--comment <string> Comment (single line).

--owner <string> Authentication ID

--remove-vanished <boolean> (default=true) Delete vanished backups. This remove the local copy if the remote backup was deleted.

--schedule <calendar-event> Run sync job at specified schedule.

`proxmox-backup-manager sync-job list [OPTIONS]`

Sync job list.

Optional parameters:

--output-format text|json|json-pretty Output format.

`proxmox-backup-manager sync-job remove <id> [OPTIONS]`

Remove a sync job configuration

<id> [<string>] Job ID.

Optional parameters:

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

`proxmox-backup-manager sync-job show <id> [OPTIONS]`

Show sync job configuration

<id> [<string>] Job ID.

Optional parameters:

--output-format text|json|json-pretty Output format.

`proxmox-backup-manager sync-job update <id> [OPTIONS]`

Update sync job config.

<id> [<string>] Job ID.

Optional parameters:

--comment <string> Comment (single line).

--delete <array> List of properties to delete.

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

--owner <string> Authentication ID

--remote <string> Remote ID.

--remote-store <string> Datastore name.

--remove-vanished <boolean> (default=true) Delete vanished backups. This remove the local copy if the remote backup was deleted.

--schedule <calendar-event> Run sync job at specified schedule.

--store <string> Datastore name.

`proxmox-backup-manager task list [OPTIONS]`

List running server tasks.

Optional parameters:

--all <boolean> Also list stopped tasks.

--limit <integer> (1 - 1000) (default=50) The maximal number of tasks to list.

--output-format text|json|json-pretty Output format.

`proxmox-backup-manager task log <upid>`

Display the task log.

<upid> [<string>] Unique Process/Task ID.

`proxmox-backup-manager task stop <upid>`

Try to stop a specific task.

<upid> [<string>] Unique Process/Task ID.

`proxmox-backup-manager user create <userid> [OPTIONS]`

Create new user.

<userid> [<string>] User ID

Optional parameters:

--comment <string> Comment (single line).

--email <string> E-Mail Address.

--enable <boolean> (default=true) Enable the account (default). You can set this to '0' to disable the account.

--expire <integer> (0 - N) (default=0) Account expiration date (seconds since epoch). '0' means no expiration date.

--firstname <string> First name.

--lastname <string> Last name.

--password <string> User Password.

`proxmox-backup-manager user delete-token <userid> <tokenname> [OPTIONS]`

Delete a user's API token

<userid> [<string>] User ID

<tokenname> [<string>] The token ID part of an API token authentication id.

This alone does NOT uniquely identify the API token - use a full *Authid* for such use cases.

Optional parameters:

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

`proxmox-backup-manager user generate-token <userid> <tokenname> [OPTIONS]`

Generate a new API token with given metadata

<userid> [<string>] User ID

<tokenname> [<string>] The token ID part of an API token authentication id.

This alone does NOT uniquely identify the API token - use a full *Authid* for such use cases.

Optional parameters:

--comment <string> Comment (single line).

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

--enable <boolean> (default=true) Enable the account (default). You can set this to '0' to disable the account.

--expire <integer> (0 - N) (default=0) Account expiration date (seconds since epoch). '0' means no expiration date.

`proxmox-backup-manager user list [OPTIONS]`

List configured users.

Optional parameters:

--output-format text|json|json-pretty Output format.

`proxmox-backup-manager user list-tokens <userid> [OPTIONS]`

List tokens associated with user.

<userid> [<string>] User ID

Optional parameters:

--output-format text|json|json-pretty Output format.

`proxmox-backup-manager user permissions <auth-id> [OPTIONS]`

List permissions of user/token.

<auth-id> [<string>] Authentication ID

Optional parameters:

--output-format text|json|json-pretty Output format.

--path <string> Access control path.

`proxmox-backup-manager user remove <userid> [OPTIONS]`

Remove a user from the configuration file.

<userid> [<string>] User ID

Optional parameters:

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

`proxmox-backup-manager user update <userid> [OPTIONS]`

Update user configuration.

<userid> [<string>] User ID

Optional parameters:

--comment <string> Comment (single line).

--delete <array> List of properties to delete.

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

--email <string> E-Mail Address.

--enable <boolean> (default=true) Enable the account (default). You can set this to '0' to disable the account.

--expire <integer> (0 - N) (default=0) Account expiration date (seconds since epoch). '0' means no expiration date.

--firstname <string> First name.

--lastname <string> Last name.

--password <string> User Password.

`proxmox-backup-manager verify <store> [OPTIONS]`

Verify backups

<store> [<string>] Datastore name.

Optional parameters:

--output-format text|json|json-pretty Output format.

`proxmox-backup-manager verify-job create <id> --store <string> [OPTIONS]`

Create a new verification job.

<id> [<string>] Job ID.

--store <string> Datastore name.

Optional parameters:

--comment <string> Comment (single line).

--ignore-verified <boolean> (default=true) Do not verify backups that are already verified if their verification is not outdated.

--outdated-after <integer> (1 - N) Days after that a verification becomes outdated

--schedule <calendar-event> Run verify job at specified schedule.

`proxmox-backup-manager verify-job list [OPTIONS]`

List all verification jobs

Optional parameters:

--output-format text|json|json-pretty Output format.

`proxmox-backup-manager verify-job remove <id> [OPTIONS]`

Remove a verification job configuration

<id> [<string>] Job ID.

Optional parameters:

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

`proxmox-backup-manager verify-job show <id> [OPTIONS]`

Show verification job configuration

<id> [<string>] Job ID.

Optional parameters:

--output-format text|json|json-pretty Output format.

`proxmox-backup-manager verify-job update <id> [OPTIONS]`

Update verification job config.

<id> [<string>] Job ID.

Optional parameters:

--comment <string> Comment (single line).

--delete <array> List of properties to delete.

--digest <string> Prevent changes if current configuration file has different SHA256 digest. This can be used to prevent concurrent modifications.

--ignore-verified <boolean> (default=true) Do not verify backups that are already verified if their verification is not outdated.

--outdated-after <integer> (1 - N) Days after that a verification becomes outdated

--schedule <calendar-event> Run verify job at specified schedule.

--store <string> Datastore name.

`proxmox-backup-manager versions [OPTIONS]`

List package versions for important Proxmox Backup Server packages.

Optional parameters:

--output-format text|json|json-pretty Output format.

--verbose <boolean> (default=false) Output verbose package information. It is ignored if output-format is specified.

A.3 pxar

`pxar create <archive> <source> [OPTIONS]`

Create a new .pxar archive.

<archive> [<string>] Archive name.

<source> [<string>] Source directory.

Optional parameters:

--all-file-systems <boolean> (default=false) Include mounted sudirs.

--entries-max <integer> (0 - 9223372036854775807) (default=1048576) Max number of entries loaded at once into memory

--exclude <array> List of paths or pattern matching files to exclude.

--no-acls <boolean> (default=false) Ignore access control list entries.

--no-device-nodes <boolean> (default=false) Ignore device nodes.

--no-fcaps <boolean> (default=false) Ignore file capabilities.

--no-fifos <boolean> (default=false) Ignore fifos.

--no-sockets <boolean> (default=false) Ignore sockets.

--no-xattrs <boolean> (default=false) Ignore extended file attributes.

--verbose <boolean> (default=false) Verbose output.

`pxar extract <archive> [<target>] [OPTIONS]`

Extract an archive.

<archive> [<string>] Archive name.

<target> [<string>] Target directory

Optional parameters:

--allow-existing-dirs <boolean> (default=false) Allows directories to already exist on restore.

--files-from <string> File containing match pattern for files to restore.

--no-acls <boolean> (default=false) Ignore access control list entries.

--no-device-nodes <boolean> (default=false) Ignore device nodes.

--no-fcaps <boolean> (default=false) Ignore file capabilities.

--no-fifos <boolean> (default=false) Ignore fifos.

--no-sockets <boolean> (default=false) Ignore sockets.

--no-xattrs <boolean> (default=false) Ignore extended file attributes.

--pattern <array> List of paths or pattern matching files to restore

--strict <boolean> (default=false) Stop on errors. Otherwise most errors will simply warn.

--verbose <boolean> (default=false) Verbose output.

`pxar help [{<command>}] [OPTIONS]`

Get help about specified command (or sub-command).

<command> [<array>] Command. This may be a list in order to spefify nested sub-commands.

Optional parameters:

--verbose <boolean> Verbose help.

`pxar list <archive> [OPTIONS]`

List the contents of an archive.

<archive> [<string>] Archive name.

Optional parameters:

--verbose <boolean> (default=false) Verbose output.

`pxar mount <archive> <mountpoint> [OPTIONS]`

Mount the archive to the provided mountpoint via FUSE.

<archive> [<string>] Archive name.

<mountpoint> [<string>] Mountpoint for the file system.

Optional parameters:

--verbose <boolean> (default=false) Verbose output, running in the foreground (for debugging).

APPENDIX

B

CONFIGURATION FILES

All Proxmox Backup Server configuration files resides inside directory `/etc/proxmox-backup/`.

B.1 `acl.cfg`

B.1.1 File Format

This file contains the access control list for the Proxmox Backup Server API.

Each line starts with `acl:`, followed by 4 additional values separated by collon.

propagate Propagate permissions down the hierachrchy

path The object path

User/Token List of users and token

Role List of assigned roles

Here is an example list:

```
acl:1:/:root@pam!test:Admin
acl:1:/datastore/store1:user1@pbs:DatastoreAdmin
```

You can use the `proxmox-backup-manager acl` command to manipulate this file.

B.1.2 Roles

The following roles exist:

`Admin` Administrator

`Audit` Auditor

`NoAccess` Disable Access

`DatastoreAdmin` Datastore Administrator

`DatastoreReader` Datastore Reader (inspect datastore content and do restores)

`DatastoreBackup` Datastore Backup (backup and restore owned backups)

`DatastorePowerUser` Datastore PowerUser (backup, restore and prune owned backup)

`DatastoreAudit` Datastore Auditor

`RemoteAudit` Remote Auditor

`RemoteAdmin` Remote Administrator

`RemoteSyncOperator` Syncronisation Opertator

TapeAudit Tape Auditor

TapeAdmin Tape Administrator

TapeOperator Tape Operator

TapeReader Tape Reader

B.2 datastore.cfg

B.2.1 File Format

The file contains a list of datastore configuration sections. Each section starts with a header `datastore: <name>`, followed by the datastore configuration options.

```
datastore: <name1>

   path <path1>
   <option1> <value1>
   ...

datastore: <name2>

   path <path2>
   ...
```

You can use the `proxmox-backup-manager datastore` command to manipulate this file.

B.2.2 Options

Required properties:

path [<string>] Directory name

Optional properties:

comment [<string>] Comment (single line).

gc-schedule [<calendar-event>] Run garbage collection job at specified schedule.

keep-daily [<integer> (1 - N)] Number of daily backups to keep.

keep-hourly [<integer> (1 - N)] Number of hourly backups to keep.

keep-last [<integer> (1 - N)] Number of backups to keep.

keep-monthly [<integer> (1 - N)] Number of monthly backups to keep.

keep-weekly [<integer> (1 - N)] Number of weekly backups to keep.

keep-yearly [<integer> (1 - N)] Number of yearly backups to keep.

notify [[gc=<enum>] [,sync=<enum>] [,verify=<enum>]] Datastore notification setting

gc = never|always|error When do we send notifications

sync = never|always|error When do we send notifications

verify = never|always|error When do we send notifications

notify-user [<string>] User ID

prune-schedule [<calendar-event>] Run prune job at specified schedule.

verify-new [<boolean>] If enabled, all backups will be verified right after completion.

B.3 user.cfg

B.3.1 File Format

This file contains the list of API users and API tokens.

Each user configuration section starts with a header `user: <name>`, followed by the user configuration options.

API token configuration starts with a header `token: <userid!token_name>`, followed by the token configuration. The data used to authenticate tokens is stored in a separate file (`token.shadow`).

```
user: root@pam

      comment Superuser
      email test@example.local
      ...

token: root@pam!token1

      comment API test token
      enable true
      expire 0
user: ...
```

You can use the `proxmox-backup-manager user` command to manipulate this file.

B.3.2 Options

Section type 'token': ApiToken properties.

Optional properties:

comment [<string>] Comment (single line).

enable [<boolean> (default=true)] Enable the account (default). You can set this to '0' to disable the account.

expire [<integer> (0 - N) (default=0)] Account expiration date (seconds since epoch). '0' means no expiration date.

Section type 'user': User properties.

Optional properties:

comment [<string>] Comment (single line).

email [<string>] E-Mail Address.

enable [<boolean> (default=true)] Enable the account (default). You can set this to '0' to disable the account.

expire [<integer> (0 - N) (default=0)] Account expiration date (seconds since epoch). '0' means no expiration date.

firstname [<string>] First name.

lastname [<string>] Last name.

B.4 remote.cfg

B.4.1 File Format

This file contains information used to access remote servers.

Each entry starts with a header `remote: <name>`, followed by the remote configuration options.

```
remote: server1

      host server1.local
      auth-id sync@pbs
      ...

remote: ...
```

You can use the `proxmox-backup-manager remote` command to manipulate this file.

B.4.2 Options

Required properties:

`auth-id` [`<string>`] Authentication ID

`host` [`<string>`] DNS name or IP address.

`password` [`<string>`] Password or auth token for remote host.

Optional properties:

`comment` [`<string>`] Comment (single line).

`fingerprint` [`<string>`] X509 certificate fingerprint (sha256).

`port` [`<integer>`] The (optional) port

B.5 sync.cfg

B.5.1 File Format

Each entry starts with a header `sync: <name>`, followed by the job configuration options.

```
sync: job1

      store store1
      remote-store store1
      remote lina

sync: ...
```

You can use the `proxmox-backup-manager sync-job` command to manipulate this file.

B.5.2 Options

Required properties:

`remote` [`<string>`] Remote ID.

`remote-store` [`<string>`] Datastore name.

`store` [`<string>`] Datastore name.

Optional properties:

comment [<string>] Comment (single line).

owner [<string>] Authentication ID

remove-vanished [<boolean> (default=true)] Delete vanished backups. This remove the local copy if the remote backup was deleted.

schedule [<calendar-event>] Run sync job at specified schedule.

B.6 verification.cfg

B.6.1 File Format

Each entry starts with a header `verification: <name>`, followed by the job configuration options.

```
verification: verify-store2

      ignore-verified true
      outdated-after 7
      schedule daily
      store store2
verification: ...
```

You can use the `proxmox-backup-manager verify-job` command to manipulate this file.

B.6.2 Options

Required properties:

store [<string>] Datastore name.

Optional properties:

comment [<string>] Comment (single line).

ignore-verified [<boolean> (default=true)] Do not verify backups that are already verified if their verification is not outdated.

outdated-after [<integer> (1 - N)] Days after that a verification becomes outdated

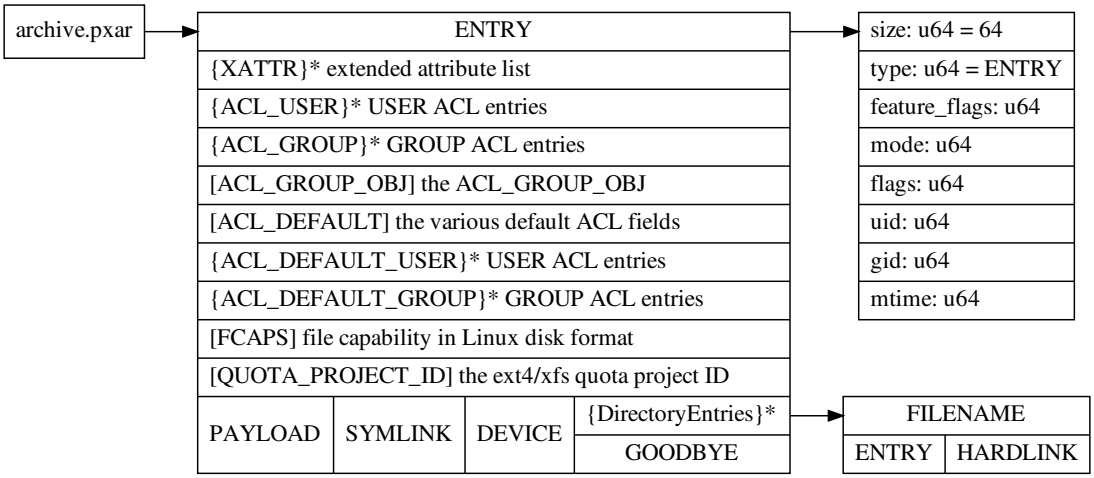
schedule [<calendar-event>] Run verify job at specified schedule.

APPENDIX

C

FILE FORMATS

C.1 Proxmox File Archive Format (`.pxar`)



C.2 Data Blob Format (`.blob`)

The data blob format is used to store small binary data. The magic number decides the exact format:

`[66, 171, 56, 7, 190, 131, 112, 161]`	unencrypted	uncompressed
`[49, 185, 88, 66, 111, 182, 163, 127]`	unencrypted	compressed
`[123, 103, 133, 190, 34, 45, 76, 240]`	encrypted	uncompressed
`[230, 89, 27, 191, 11, 191, 216, 11]`	encrypted	compressed

Compression algorithm is `zstd`. Encryption cipher is `AES_256_GCM`.

Unencrypted blobs use the following format:

`MAGIC: [u8; 8]`
`CRC32: [u8; 4]`
`Data: (max 16MiB)`

Encrypted blobs additionally contains a 16 byte IV, followed by a 16 byte Authenticated Encyryption (AE) tag, followed by the encrypted data:

MAGIC: [u8; 8]
CRC32: [u8; 4]
ÌV: [u8; 16]
TAG: [u8; 16]
Data: (max 16MiB)

C.3 Fixed Index Format (.fidx)

All numbers are stored as little-endian.

MAGIC: [u8; 8]	[47, 127, 65, 237, 145, 253, 15, 205]
uuid: [u8; 16],	Unique ID
ctime: i64,	Creation Time (epoch)
index_csum: [u8; 32],	Sha256 over the index (without header) SHA256(digest1||digest2||...)
size: u64,	Image size
chunk_size: u64,	Chunk size
reserved: [u8; 4016],	overall header size is one page (4096 bytes)
digest1: [u8; 32]	first chunk digest
digest2: [u8; 32]	next chunk
...	next chunk ...

C.4 Dynamic Index Format (.didx)

All numbers are stored as little-endian.

MAGIC: [u8; 8]	[28, 145, 78, 165, 25, 186, 179, 205]
uuid: [u8; 16],	Unique ID
ctime: i64,	Creation Time (epoch)
index_csum: [u8; 32],	Sha256 over the index (without header) SHA256(offset1||digest1||offset2||digest2||...)
reserved: [u8; 4032],	Overall header size is one page (4096 bytes)
offset1: u64	End of first chunk
digest1: [u8; 32]	first chunk digest
offset2: u64	End of second chunk
digest2: [u8; 32]	second chunk digest
...	next chunk offset/digest

APPENDIX

D

BACKUP PROTOCOL

Proxmox Backup Server uses a REST based API. While the management interface use normal HTTP, the actual backup and restore interface use HTTP/2 for improved performance. Both HTTP and HTTP/2 are well known standards, so the following section assumes that you are familiar on how to use them.

D.1 Backup Protocol API

To start a new backup, the API call `GET /api2/json/backup` needs to be upgraded to a HTTP/2 connection using `proxmox-backup-protocol-v1` as protocol name:

```
GET /api2/json/backup HTTP/1.1
UPGRADE: proxmox-backup-protocol-v1
```

The server replies with HTTP 101 Switching Protocol status code, and you can then issue REST commands on that updated HTTP/2 connection.

The backup protocol allows you to upload three different kind of files:

- Chunks and blobs (binary data)
- Fixed Indexes (List of chunks with fixed size)
- Dynamic Indexes (List of chunk with variable size)

The following section gives a short introduction how to upload such files. Please use the API Viewer for details about available REST commands.

D.1.1 Upload Blobs

Uploading blobs is done using `POST /blob`. The HTTP body contains the data encoded as *Data Blob*).

The file name needs to end with `.blob`, and is automatically added to the backup manifest.

D.1.2 Upload Chunks

Chunks belong to an index, so you first need to open an index (see below). After that, you can upload chunks using `POST /fixed_chunk` and `POST /dynamic_chunk`. The HTTP body contains the chunk data encoded as *Data Blob*).

D.1.3 Upload Fixed Indexes

Fixed indexes are use to store VM image data. The VM image is split into equally sized chunks, which are uploaded individually. The index file simply contains a list to chunk digests.

You create a fixed index with `POST /fixed_index`. Then upload chunks with `POST /fixed_chunk`, and append them to the index with `PUT /fixed_index`. When finished, you need to close the index using `POST /fixed_close`.

The file name needs to end with `.fidx`, and is automatically added to the backup manifest.

D.1.4 Upload Dynamic Indexes

Dynamic indexes are use to store file archive data. The archive data is split into dynamically sized chunks, which are uploaded individually. The index file simply contains a list to chunk digests and offsets.

You create a dynamic sized index with `POST /dynamic_index`. Then upload chunks with `POST /dynamic_chunk`, and append them to the index with `PUT /dynamic_index`. When finished, you need to close the index using `POST /dynamic_close`.

The file name needs to end with `.didx`, and is automatically added to the backup manifest.

D.1.5 Finish Backup

Once you have uploaded all data, you need to call `POST /finish`. This commits all data and ends the backup protocol.

D.2 Restore/Reader Protocol API

To start a new reader, the API call `GET /api2/json/reader` needs to be upgraded to a HTTP/2 connection using `proxmox-backup-reader-protocol-v1` as protocol name:

```
GET /api2/json/reader HTTP/1.1
UPGRADE: proxmox-backup-reader-protocol-v1
```

The server replies with HTTP 101 Switching Protocol status code, and you can then issue REST commands on that updated HTTP/2 connection.

The reader protocol allows you to download three different kind of files:

- Chunks and blobs (binary data)
- Fixed Indexes (List of chunks with fixed size)
- Dynamic Indexes (List of chunk with variable size)

The following section gives a short introduction how to download such files. Please use the API Viewer for details about available REST commands.

D.2.1 Download Blobs

Downloading blobs is done using `GET /download`. The HTTP body contains the data encoded as *Data Blob*.

D.2.2 Download Chunks

Downloading chunks is done using `GET /chunk`. The HTTP body contains the data encoded as *Data Blob*).

D.2.3 Download Index Files

Downloading index files is done using `GET /download`. The HTTP body contains the data encoded as *Fixed Index* or *Dynamic Index*.

APPENDIX

E

CALENDAR EVENTS

E.1 Introduction and Format

Certain tasks, for example pruning and garbage collection, need to be performed on a regular basis. Proxmox Backup Server uses a format inspired by the systemd Time and Date Specification (see systemd.time manpage) called *calendar events* for its schedules.

Calendar events are expressions to specify one or more points in time. They are mostly compatible with systemd's calendar events.

The general format is as follows:

Listing 1: Calendar event

```
[WEEKDAY] [[YEARS-]MONTHS-DAYS] [HOURS:MINUTES[:SECONDS]]
```

Note that there either has to be at least a weekday, date or time part. If the weekday or date part is omitted, all (week)days are included. If the time part is omitted, the time 00:00:00 is implied. (e.g. '2020-01-01' refers to '2020-01-01 00:00:00')

Weekdays are specified with the abbreviated English version: *mon, tue, wed, thu, fri, sat, sun*.

Each field can contain multiple values in the following formats:

- comma-separated: e.g., 01,02,03
- as a range: e.g., 01..10
- as a repetition: e.g, 05/10 (means starting at 5 every 10)
- and a combination of the above: e.g., 01,05..10,12/02
- or a * for every possible value: e.g., *:00

There are some special values that have specific meaning:

Value	Syntax
minutely	**-*-* *:*:00*
hourly	**-*-* *:00:00*
daily	**-*-* 00:00:00*
weekly	*mon *-*-* 00:00:00*
monthly	**-*-01 00:00:00*
yearly or *annually*	**-01-01 00:00:00*
quarterly	**-01,04,07,10-01 00:00:00*
semiannually or *semi-annually*	**-01,07-01 00:00:00*

Here is a table with some useful examples:

Example	Alternative	Explanation
mon,tue,wed,thu,fri	*mon..fri*	Every working day at 00:00
sat,sun	*sat..sun*	Only on weekends at 00:00
mon,wed,fri	–	Monday, Wednesday, Friday at 00:00
12:05	–	Every day at 12:05 PM
*:00/5	*0/1:0/5*	Every five minutes
*mon..wed *:30/10*	*mon,tue,wed *:30/10*	Monday, Tuesday, Wednesday 30, 40 and 50 minutes after every full hour
mon..fri 8..17,22:0/15	–	Every working day every 15 minutes between 8 AM and 6 PM and between 10 PM and 11 PM
fri 12..13:5/20	*fri 12,13:5/20*	Friday at 12:05, 12:25, 12:45, 13:05, 13:25 and 13:45
12,14,16,18,20,22:5	*12/2:5*	Every day starting at 12:05 until 22:05, every 2 hours
:	*0/1:0/1*	Every minute (minimum interval)
**-05*	–	On the 5th day of every Month
*Sat *-1..7 15:00*	–	First Saturday each Month at 15:00
2015-10-21	–	21st October 2015 at 00:00

E.2 Differences to systemd

Not all features of systemd calendar events are implemented:

- no Unix timestamps (e.g. *@12345*): instead use date and time to specify a specific point in time
- no timezone: all schedules use the set timezone on the server
- no sub-second resolution
- no reverse day syntax (e.g. 2020-03~01)
- no repetition of ranges (e.g. 1..10/2)

E.3 Notes on scheduling

In Proxmox Backup scheduling for most tasks is done in the *proxmox-backup-proxy*. This daemon checks all job schedules if they are due every minute. This means that even if *calendar events* can contain seconds, it will only be checked once a minute.

Also, all schedules will be checked against the timezone set in the Proxmox Backup server.

APPENDIX

F

GLOSSARY

Virtual machine A virtual machine is a program that can execute an entire operating system inside an emulated hardware environment.

Container A container is an isolated user space. Programs run directly on the host's kernel, but with limited access to the host resources.

Datastore A place to store backups. A directory which contains the backup data. The current implementation is file-system based.

Rust Rust is a new, fast and memory-efficient system programming language. It has no runtime or garbage collector. Rust's rich type system and ownership model guarantee memory-safety and thread-safety. I can eliminate many classes of bugs at compile-time.

Sphinx Is a tool that makes it easy to create intelligent and beautiful documentation. It was originally created for the documentation of the Python programming language. It has excellent facilities for the documentation of software projects in a range of languages.

reStructuredText Is an easy-to-read, what-you-see-is-what-you-get plaintext markup syntax and parser system.

FUSE Filesystem in Userspace (FUSE) defines an interface which makes it possible to implement a filesystem in userspace as opposed to implementing it in the kernel. The fuse kernel driver handles filesystem requests and sends them to a userspace application.

Remote A remote Proxmox Backup Server installation and credentials for a user on it. You can pull datastores from a remote to a local datastore in order to have redundant backups.

APPENDIX

G

GNU FREE DOCUMENTATION LICENSE

Version 1.3, 3 November 2008

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document ”free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of ”copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The ”Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as ”you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A ”Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A ”Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The ”Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The ”Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A ”Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not ”Transparent” is called ”Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The ”Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, ”Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The ”publisher” means any person or entity that distributes copies of the Document to the public.

A section ”Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as ”Acknowledgements”, ”Dedications”, ”Endorsements”, or ”History”.) To ”Preserve the Title” of such a section when you modify the Document means that it remains a section ”Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled ”Endorsements”. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled ”Endorsements” or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled ”Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled ”History” in the various original documents, forming one section Entitled ”History”; likewise combine any sections Entitled ”Acknowledgements”, and any sections Entitled ”Dedications”. You must delete all sections Entitled ”Endorsements”.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an ”aggregate” if the

copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See https://www.gnu.org/licenses/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those

works. A public wiki that anybody can edit is an example of such a server. A ”Massive Multiauthor Collaboration” (or ”MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

”CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

”Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is ”eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

INDEX