Features

Backup Files and Directories Using Snapshots

Kopia creates snapshots of the files and directories you designate, then encrypts these snapshots before they leave your computer, and finally uploads these encrypted snapshots to cloud/network/local storage called a repository. Snapshots are maintained as a set of historical point-in-time records based on policies that you define.

Kopia uses content-addressable storage for snapshots, which has many benefits:

- Each snapshot is always incremental. This means that all data is uploaded once to the repository based on file content, and a file is only re-uploaded to the repository if the file is modified. Kopia uses file splitting based on rolling hash, which allows efficient handling of changes to very large files: any file that gets modified is efficiently snapshotted by only uploading the changed parts and not the entire file.

- Multiple copies of the same file will be stored once. This is known as deduplication and saves you a lot of storage space (i.e., saves you money).

- After moving or renaming even large files, Kopia can recognize that they have the same content and won't need to upload them again.

- Multiple users or computers can share the same repository: if different users have the same files, the files are uploaded only once as Kopia deduplicates content across the entire repository.

  NOTE: Kopia allows one password per repository, and there is currently no access control mechanism when sharing a repository with someone. If you share a repository with someone else, then you must also share your password with them and they will have access to your data. Therefore, make sure you trust all the other users/computers that you share a repository with!

Policies Control What and How Files/Directories are Saved in Snapshots

Kopia allows you to create an unlimited number of policies for each repository. Policies allow you to define what files/directories to backup in a snapshot and other features of a snapshot, including but not limited to:

- how frequently/when Kopia should automatically create snapshots of your data
- whether to exclude certain files/directories from snapshots
- how long to retain a snapshot before expiring it and removing it from the repository
- whether and how to compress the files/directories being backed up

Policies can be applied at multiple different levels:

- `global` (i.e., the policy is applied to all snapshots for the repository)
- `username@hostname:/path` (i.e., the policy is applied only for the specific files/folders being backed up in that particular policy)
- `username@hostname` (i.e., the policy is applied for all policies belonging to the specific user)
- `@hostname` (i.e., the policy is applied to all policies belonging to the specific machine)

Save Snapshots to Cloud, Network, or Local Storage

Kopia performs all its operations locally on your machine, meaning that you do not need to have any dedicated server to run your backups and you can save your snapshots to a variety of storage locations. Kopia supports network and local storage locations, of course, but also many cloud or remote storage locations:

- **Amazon S3** and any **cloud storage that is compatible with S3**
- **Azure Blob Storage**

- **Backblaze B2**
- **Google Cloud Storage**
- Any remote server or cloud storage that supports **WebDAV**
- Any remote server or cloud storage that supports **SFTP**
- Some of the cloud storages supported by **Rclone**
  - Requires you to download and setup Rclone in addition to Kopia, but after that Kopia manages/runs Rclone for you
  - Rclone support is experimental: not all the cloud storages supported by Rclone have been tested to work with Kopia, and some may not work with Kopia; Kopia has been tested to work with **Dropbox**, **OneDrive**, and **Google Drive** through Rclone
- Your own server by setting up a Kopia Repository Server

Read the repositories help page for more information on supported storage locations.

With Kopia you're in full control of where to store your snapshots; you pick the cloud storage you want to use. Kopia plays no role in selecting your storage locations. You must provision and pay (the storage provider) for whatever storage locations you want to use, and then tell Kopia what those storage locations are. The advantage of decoupling the software (i.e., Kopia) from storage is that you can use whatever storage locations you desire -– it makes no difference to Kopia what storage you use. You can even use multiple storage locations if you want to, and Kopia also supports backing up multiple machines to the same storage location.

> NOTE: Different storage providers may operate slightly differently, so you need to make sure whatever storage location you use has enough capacity to store your backups and enough availability to be able to recover the data when needed.

Restore Snapshots Using Multiple Methods

To restore data, Kopia gives you three options:

- mount the contents of a snapshot as a local disk so that you can browse and copy files/directories from the snapshot as if the snapshot is a local directory on your machine

- restore all files/directories contained in a snapshot to any local or network location that you designate

- selectively restore individual files from a snapshot

End-to-End 'Zero Knowledge' Encryption

All data is encrypted before it leaves your machine. Encryption is baked into the DNA of Kopia, and you cannot create a backup without using encryption. Kopia allows you to pick from two state-of-the-art encryption algorithms, AES-256 and ChaCha20.

Kopia encrypts both the content and the names of your backed up files/directories.

The data is encrypted using per-content keys which are derived from the 256-bit master key that is stored in the repository. The master key is encrypted with a password you provide. This means that anyone that does not know the password cannot access your backed up files and will not know what files/directories are contained in the snapshots that are saved in the repository. Importantly, the password you provide is never sent to any server or anywhere outside your machine, and only you know your password. In other words, Kopia provides your backups with end-to-end 'zero knowledge' encryption. However, this also means that you cannot restore your files if you forget your password: there is no way to recover a forgotten password because only you know it. (But you can change your password if you are still connected to the repository that stores your snapshots.)

Compression

Kopia can compress your data to save storage and bandwidth. Several compression methods are supported, including:

- pgzip

- s2

- zstd

Error Correction

Kopia supports Reed-Solomon error correction algorithm to help prevent your snapshots from being corrupted by faulty hardware, such as bitflips or bitrot.

Verifying Backup Validity and Consistency

Backing up data is great, but you also need to be able to restore that data when (if) the time arises. Kopia has built-in functions that enable you to verify the consistency/validity of your backed up files. You can run these consistency checks are frequently as you like (e.g., once a month, once a year, etc.). Read the repository consistency help docs for more information.

Recovering Backed Up Data When There is Data Loss

Although never guaranteed, Kopia can often recover your files even if there is some partial data loss at your repository (e.g., a hard drive failure), because key index information and repository metadata is stored redundantly to prevent single points of failure. Note that Kopia cannot recover data where the actual backed up data file in the repository is corrupt, so make sure to regularly run repository consistency checks (see above discussion)!

Regular Automatic Maintenance of Repositories

Over time, repositories can get bloated to the point of decreased performance and waste of storage space. Kopia runs automatic maintenance that ensures optimal performance and space usage. Read the maintenance help docs for more information.

Caching

Kopia maintains a local cache of recently accessed objects making it possible to quickly browse repository contents without having to download from the storage location (regardless of whether the storage is cloud, network, or local).

Both Command Line and Graphical User Interfaces

Kopia has a rich command-line interface that gives you full access to all Kopia features, including allowing you to create/connect to repositories, manage snapshots and policies, and provides low-level access to the underlying repository, including low-level data recovery.

Do not want to use command-line? No problem. Kopia also comes with a powerful official graphical user interface that allows you to easily create/connect to repositories, manage snapshots and policies, and restore data as needed.

Optional Server Mode with API Support to Centrally Manage Backups of Multiple Machines

Kopia is designed to backup individual machines and you absolutely do not need a server to run Kopia. If you have a handful of machines, you can install and use Kopia on each of them individually, no problem. At the same time, Kopia can also be run in server mode for those that are looking to centrally manage backups of multiple machines, in which case the Kopia server exposes an API that can be used to build client tools to do things like trigger snapshots, get client status, and access snapshotted data. Kopia's server mode makes it incredibly easy to centrally manage backups of multiple computers.

Speed

Kopia. Is. Fast.

Last modified February 6, 2024: Update _index.md: old link redirects to sales page (#3616) (b057b0b9)

Getting Started Guide

This guide will walk you through installing Kopia and setting up Kopia to backup/restore your data. Make sure to familiarize yourself with Kopia features before following this guide, so that you understand the appropriate terminology. As a reminder:

- A `snapshot` is a point-in-time backup of your files/directories; each snapshot contains the files/directories that you can restore when you need to.
- A `repository` is the storage location where your snapshots are saved; Kopia supports cloud/remote, network, and local storage locations and all repositories are encrypted with a password that you designate.
- A `policy` is a set of rules that tells Kopia how to create/manage snapshots; this includes features such as compression, snapshot retention, and scheduling when to take automatically snapshots.

Download and Installation

Read the download and installation guide to learn how to download & install Kopia. As a reminder, Kopia comes in two variants: command-line interface (CLI) and graphical user interface (GUI). Pick the one you like the most.

Setting Up Kopia

Once you have installed Kopia, setting up Kopia is quite easy but varies depending on if you are using Kopia CLI or Kopia GUI (also known as `KopiaUI`).

Kopia GUI (`KopiaUI`)

Setting up Kopia via the GUI is very easy.

Creating and Connecting to a Repository

When you run `KopiaUI` for the first time, you will need to create a `repository`. You will see all supported repository types on-screen within the program interface. Pick the one you want and follow the on-screen directions to get it setup; you will need to enter various different details about the storage location that you selected, and you will pick a password that will be used to encrypt all the snapshots that you store in the repository. (As a reminder, Kopia uses end-to-end zero knowledge encryption, so your password is never sent anywhere and it never leaves your machine!) You can also name the repository whatever you want.

There is absolutely no way to restore snapshots (i.e., your backed up files/directories) from a repository if you forget your password, so do not forget it and keep it secure!

> NOTE: Remember, before you use Kopia, you need to provision, setup, and pay (the storage provider) for whatever storage location you want to use; Kopia will not do that for you. After you have done that, you can create a `repository` for that storage location in Kopia. For example, if you want to use `Backblaze B2`, you need to create a Backblaze account, create a B2 bucket, and get the access keys for the bucket; then you can use the `Backblaze B2` repository option in `KopiaUI` to create a repository.

Defining Snapshot Policy and Creating New Snapshot

Once you have created a repository, you can start backing up your files/directories by creating a new `policy` in `KopiaUI`. You can do this from the `Policies` tab and the process, again, is quite straightforward: enter the `directory` which contains the files you want to backup (you can either manually type in the `directory path` or browse for the `directory`), hit the `Set Policy` button, choose your policy settings from the on-screen options (all policy options are fairly self-explanatory), and hit the `Save Policy` button. Kopia will then automatically begin taking the snapshot following the settings you set for the policy.

After the initial snapshot, for every snapshot afterwards Kopia will rescan the file/directories and only upload file content that has changed. All snapshots in Kopia are always incremental; a snapshot will only upload files/file contents that are not in the repository yet, which saves storage space and upload time. This even applies to files that were moved or renamed. In fact, if two computers have exactly the same file and both computers are backing up to the same `repository`, the file will still be stored only once.

> PRO TIP: If you pick a value for `Snapshot Frequency` when creating a `policy`, then Kopia will automatically take snapshots at that frequency (e.g., every one hour or whatever value you pick), and you do not need to remember to manually run the snapshot. If you do not pick a `Snapshot Frequency`, then Kopia will not automatically take snapshots, and you need to manually run snapshots from the `Snapshots` tab (just click the `Snapshot Now` button as needed).

Note that you can set policies at two levels in `KopiaUI` – at the `global` level, where the settings are applied by default to all policies that do not define their own settings, or at the individual `policy` level, where the settings are applied only to that particular policy. By default, all new policies are set to inherit settings from the `global` policy. The `global` policy is the one that says `*` for `Username`, `Host`, and `Path`.

> PRO TIP: Kopia does not currently support the ability to save one snapshot to multiple different repositories. However, you can use `KopiaUI` to connect to multiple different repositories simultaneously and create identical policies for each repository, which essentially achieves the same outcome of saving one snapshot to multiple different repositories. Connecting to more than one repository in `KopiaUI` is easy: just right-click the icon of the desktop application and select `Connect To Another Repository...`. Currently, this is only available in the desktop version of `KopiaUI` and not the web-based `KopiaUI`. However, if you are using the web-based `KopiaUI`, you can manually run multiple instances of `KopiaUI` to achieve the same outcome.

Restoring Files/Directories from Snapshots

When you want to restore your files/directories from a snapshot, you can do so from the `Snapshots` tab in `KopiaUI`. Just click the `Path` for the files/directories you want to restore and then find the specific `snapshot` you want to restore from. You will then be given the option to either

- `Mount` the snapshot as a local drive so that you can browse, open, and copy any files/directories from the snapshot to your local machine;
- `Restore` all the contents of the snapshot to a local or network location;
- or download individual files from the snapshot (which can be done by browsing the snapshot contents from inside `KopiaUI` and clicking on the file you want to download).

You can restore files/directories using either of these options.

Video Tutorial

Here is a video tutorial on how to use `KopiaUI` (note that the video is of an older version of `KopiaUI` and the interface is different in the current version of `KopiaUI`, but the main principles of how to use `KopiaUI` are the same):

Kopia CLI

Setting up Kopia via the CLI follows similar steps as the GUI, but obviously requires using command-line rather than a graphical user interface.

> NOTE: This guide focuses on simple scenarios. You can learn more about all the command-line features in the command-line reference page.

Creating a Repository

The first thing you need to do is create a `repository`. For a full list of supported types of repositories that you can create, see the repositories page.

To create a repository, use one of the subcommands of `kopia repository create` and follow the on-screen instructions. When creating the repository, you must provide a password that will be used to encrypt all the snapshots and their contents in the repository. (As a reminder, Kopia uses end-to-end zero knowledge encryption, so your password is never sent anywhere and it never leaves your machine!)

There is absolutely no way to restore snapshots (i.e., your backed up files/directories) from a repository if you forget your password, so do not forget it and keep it secure!

As an example, if you want to create a repository in a locally-mounted or network-attached filesystem, you would run the following command:

```
$ kopia repository create filesystem --path /tmp/my-repository
```

You can read more about all the supported `kopia repository create` commands for different repositories from the repositories page.

> NOTE: Remember, before you use Kopia, you need to provision, setup, and pay (the storage provider) for whatever storage location you want to use; Kopia will not do that for you. After you have done that, you can create a `repository` for that storage location in Kopia. For example, if you want to use `Backblaze B2`, you need to create a Backblaze account, create a B2 bucket, and get the access keys for the bucket; then you can use the `kopia repository create b2` command to create a repository.

Connecting to Repository

To connect to a repository after you have created it or to connect to an existing repository, simply use one of the subcommands of `kopia repository connect` instead of `kopia repository create`. You can connect as many computers as you like to the same repository, even simultaneously.

For example:

```
$ kopia repository connect filesystem --path /tmp/my-repository
```

Creating Initial Snapshot

Let's create our first snapshot. That's as simple as pointing `kopia snapshot create` to the directory that contains the files/directories you want to backup, but note you need to make sure to be connected to a repository first (see above). We will create the snapshot of the source code of Kopia itself:

```
$ kopia snapshot create $HOME/Projects/github.com/kopia/kopia
```

After completion, Kopia prints the identifier of the root of the snapshot, which starts with `k`:

```
uploaded snapshot 9a622e33ab134ef440f76ed755f79c2f
  (root kfe997567fb1cf8a13341e4ca11652f70) in 1m42.044883302s
```

Incremental Snapshots

Let's take the snapshot of the same files/directories again. To do so, just rerun the same `kopia snapshot create` command…

```
$ kopia snapshot create $HOME/Projects/github.com/kopia/kopia
```

…and Kopia will rescan the file/directories and only upload the file content that has changed. Assuming we did not make any changes to the files/directories, the snapshot root will be identical, because all object identifiers in Kopia are derived from contents of the underlying data:

```
uploaded snapshot 8a45c3b079cf5e7b99fb855a3701607a
  (root kfe997567fb1cf8a13341e4ca11652f70) in 563.670362ms
```

Notice that snapshot creation was nearly instantaneous. This is because Kopia did not have to upload almost any files to the repository, except tiny piece of metadata about the snapshot itself.

All snapshots in Kopia are always incremental; a snapshot will only upload files/file contents that are not in the repository yet, which saves storage space and upload time. This even applies to files that were moved or renamed. In fact, if two computers have exactly the same file and both computers are backing up to the same `repository`, the file will still be stored only once.

Managing Snapshots

We can see the history of snapshots of a directory using `kopia snapshot list`:

```
$ kopia snapshot list $HOME/Projects/github.com/kopia/kopia
jarek@jareks-mbp:/Users/jarek/Projects/Kopia
  2019-06-22 20:15:51 PDT kb9a8420bf6b8ea280d6637ad1adbd4c5 61.4 MB drwxr-xr-x files:12500 dirs:798 (latest-5)
  + 1 identical snapshots until 2019-06-22 20:15:57 PDT
  2019-06-22 20:21:39 PDT kbb7dd85a55ca79f282b59b57e5f9c479 61.4 MB drwxr-xr-x files:12500 dirs:798 (latest-3)
  2019-06-22 20:21:42 PDT ke2e07d38a8a902ad07eda5d2d0d3025d 61.4 MB drwxr-xr-x files:12500 dirs:798 (latest-2)
  + 1 identical snapshots until 2019-06-22 20:21:44 PDT
```

To compare contents of two snapshots, use `kopia diff`:

```
$ kopia diff kb9a8420bf6b8ea280d6637ad1adbd4c5 ke2e07d38a8a902ad07eda5d2d0d3025d
changed ./content/docs/Getting started/_index.md at 2019-06-22 20:21:30.176230323 -0700 PDT (size 5346 -> 6098)
```

We can list the contents of the directory using `kopia ls`:

```
$ kopia ls -l kb9a8420bf6b8ea280d6637ad1adbd4c5
-rw-r--r--         6148 2019-06-22 19:01:45 PDT aea2fe8e5ed3104806957f48648c957e   .DS_Store
-rw-r--r--           78 2019-05-09 22:33:06 PDT c829f2205d0ba889ebb354464e14c97a   .gitignore
-rw-r--r--         1101 2019-05-09 22:33:06 PDT 5c4da68139ab0a92a56c334988c75e2a   CONTRIBUTING.md
-rw-r--r--        11357 2019-05-09 22:33:06 PDT 28614f260fab7463e3cd9c410a501c3f   LICENSE
-rw-r--r--         1613 2019-06-22 19:01:17 PDT 5c1f9d67a2b1e2d34fc121ba774266b4   Makefile
-rw-r--r--         2286 2019-05-09 22:33:06 PDT 83a5b758d8409550010786e254096606   README.md
drwxr-xr-x        11264 2019-05-09 22:33:06 PDT kc76b1a9ddf378f803f1710df1150ded6  assets/
drwxr-xr-x         6275 2019-06-02 23:08:14 PDT kf3b4b410df41570345dbc2a8043ee29b  cli2md/
-rw-r--r--         3749 2019-05-14 19:00:21 PDT 8c9e27bed2f577b31b07b07da4bdfffb   config.toml
drwxr-xr-x       879721 2019-06-22 20:15:45 PDT k24eb31a05b81d1a83c47c40a4f7b9f0e  content/
-rwxr-xr-x          727 2019-05-09 22:33:06 PDT 2c08f511019f1f5f45f889909c755a9b   deploy.sh
drwxr-xr-x         1838 2019-05-14 19:00:21 PDT k024f1106e0cd56e2c6611cf884a30894  layouts/
drwxr-xr-x     13682567 2019-06-22 18:57:48 PDT k181d6990e75dd783bd50dae36591622a  node_modules/
-rw-r--r--        94056 2019-06-22 18:57:49 PDT ed474fb638d2a3b1c528295d1586466a   package-lock.json
-rw-r--r--          590 2019-05-09 22:33:06 PDT ee85ae1f1cdb70bbd9e335be9762c251   package.json
drwxr-xr-x      7104710 2019-06-22 19:01:38 PDT keb814d92fe795b96795d5bdbfa816ad6  public/
drwxr-xr-x       904965 2019-06-22 20:13:56 PDT k7bf88a7ca076b03f0dafc93ab5fa2263  resources/
drwxr-xr-x     38701570 2019-06-01 20:11:32 PDT kdb9f41fc8db5c45b1aec06df001be995  themes/
```

For each file/directory in a directory, Kopia stores its name, size, attributes and object ID which has the contents of the file or directory.

To examine contents of files, use `kopia show` while passing the object identifier of the file or directory you want to examine:

```
$ kopia show 8c9e27bed2f577b31b07b07da4bdfffb
```

Directories are stored as JSON objects, so it's possible to see their contents as if they were regular files using `kopia content show` along with the the directory's object identifier (the `-j` option displays pretty-printed JSON):

```
$ kopia content show -j kb9a8420bf6b8ea280d6637ad1adbd4c5
```

This command returns:

```
{
  "stream": "kopia:directory",
  "entries": [
    {
      "name": "assets",
      "type": "d",
      "mode": "0755",
      "mtime": "2019-05-14T18:24:15-07:00",
      "uid": 501,
      "gid": 20,
      "obj": "kc76b1a9ddf378f803f1710df1150ded6",
      "summ": {
        "size": 11264,
        "files": 2,
        "dirs": 3,
        "maxTime": "2019-05-09T22:33:06-07:00"
      }
    },
    ...
    {
      "name": "package.json",
      "type": "f",
      "mode": "0644",
      "size": 590,
      "mtime": "2019-05-09T22:33:06-07:00",
      "uid": 501,
      "gid": 20,
      "obj": "ee85ae1f1cdb70bbd9e335be9762c251"
    }
  ],
  "summary": {
    "size": 61414615,
    "files": 12500,
    "dirs": 798,
    "maxTime": "2019-06-22T20:15:45.301289096-07:00"
  }
}
```

Mounting Snapshots and Restoring Files/Directories from Snapshots

We can mount the contents of a snapshot as a local filesystem and examine it using regular file commands to examine the contents using the `kopia mount` command:

```
$ mkdir /tmp/mnt
$ kopia mount kb9a8420bf6b8ea280d6637ad1adbd4c5 /tmp/mnt &
$ ls -l /tmp/mnt/
total 119992
-rw-r--r--  1 jarek  staff      1101 May  9 22:33 CONTRIBUTING.md
-rw-r--r--  1 jarek  staff     11357 May  9 22:33 LICENSE
-rw-r--r--  1 jarek  staff      1613 Jun 22 19:01 Makefile
-rw-r--r--  1 jarek  staff      2286 May  9 22:33 README.md
drwxr-xr-x  1 jarek  staff     11264 May  9 22:33 assets
drwxr-xr-x  1 jarek  staff      6275 Jun  2 23:08 cli2md
-rw-r--r--  1 jarek  staff      3749 May 14 19:00 config.toml
drwxr-xr-x  1 jarek  staff    879721 Jun 22 20:15 content
-rwxr-xr-x  1 jarek  staff       727 May  9 22:33 deploy.sh
drwxr-xr-x  1 jarek  staff      1838 May 14 19:00 layouts
drwxr-xr-x  1 jarek  staff  13682567 Jun 22 18:57 node_modules
-rw-r--r--  1 jarek  staff     94056 Jun 22 18:57 package-lock.json
-rw-r--r--  1 jarek  staff       590 May  9 22:33 package.json
drwxr-xr-x  1 jarek  staff   7104710 Jun 22 19:01 public
drwxr-xr-x  1 jarek  staff    904965 Jun 22 20:13 resources
drwxr-xr-x  1 jarek  staff  38701570 Jun  1 20:11 themes
$ umount /tmp/mnt
```

Mounting is currently the recommended way of restoring files/directories from snapshots. However, you can also use the `kopia snapshot restore` command to restore files/directories from snapshots.

Policies

Policies can be used to specify how Kopia snapshots are taken and retained. We can define various different `policy` options, including:

- which files to ignore
- how many hourly, daily, weekly, monthly and yearly snapshots to maintain
- how frequently snapshots should be made
- whether to compress files or not

To learn read more about what `policy` options are available, see the Kopia `policy` command help docs.

Each `repository` has a `global` policy, which contains the defaults used for all policies if a specific policy does not define its own settings. We can examine the `global` policy by using `kopia policy show --global`:

```
$ kopia policy show --global
Policy for (global):
Keep:
  Annual snapshots:    3           (defined for this target)
  Monthly snapshots:  24           (defined for this target)
  Weekly snapshots:   25           (defined for this target)
  Daily snapshots:    14           (defined for this target)
  Hourly snapshots:   48           (defined for this target)
  Latest snapshots:   10           (defined for this target)

Files policy:
  No ignore rules.
  Read ignore rules from files:
    .kopiaignore                   (defined for this target)
```

We can change policy settings using the `kopia policy set` command. This command allows you to change the `global` policy or change specific policies for a 'user@host', a '@host', a 'user@host:path', or a particular directory. For example, here we tell Kopia to set the policy to ignore two directories from being included in the snapshot of `jarek@jareks-mbp:/Users/jarek/Projects/Kopia/site`:

```
$ kopia policy set --add-ignore public/ --add-ignore node_modules/ .
Setting policy for jarek@jareks-mbp:/Users/jarek/Projects/Kopia/site
 - adding public/ to ignored files
 - adding node_modules/ to ignored files
```

Now when taking snapshot of `jarek@jareks-mbp:/Users/jarek/Projects/Kopia/site`, the directories `public/` and `node_modules/` will be skipped.

The `kopia policy set` command help docs provide more information about all the policy options you have. As another example, we can set a maximum number of weekly snapshots:

```
$ kopia policy set --keep-weekly 30 .
Setting policy for jarek@jareks-mbp:/Users/jarek/Projects/Kopia/site
 - setting number of weekly backups to keep to 30.
```

If you want to examine the policy for a particular directory, use `kopia policy show`:

```
$ kopia policy show .
Policy for jarek@jareks-mbp:/Users/jarek/Projects/Kopia/site:
Keep:
  Annual snapshots:    3           inherited from (global)
  Monthly snapshots:  24           inherited from (global)
  Weekly snapshots:   30           (defined for this target)
  Daily snapshots:    14           inherited from (global)
  Hourly snapshots:   48           inherited from (global)
  Latest snapshots:   10           inherited from (global)

Files policy:
  Ignore rules:
    dist/                             (defined for this target)
    node_modules/                     (defined for this target)
    public/                           (defined for this target)
  Read ignore rules from files:
    .kopiaignore                     inherited from (global)
```

To list all policies for a `repository`, we can use `kopia policy list`:

```
$ kopia policy list
7898f47e36bad80a6d5d90f06ef16de6 (global)
63fc854c283ad63cafbca54eaa4509e9 jarek@jareks-mbp:/Users/jarek/Projects/Kopia/site
2339ab4739bb29688bf26a3a841cf68f jarek@jareks-mbp:/Users/jarek/Projects/Kopia/site/node_modules
```

Finally, you can also import and export policies using the `kopia policy import` and `kopia policy export` commands:

```
$ kopia policy import --from-file import.json
$ kopia policy export --to-file export.json
```

In the above example, `import.json` and `export.json` share the same format, which is a JSON map of policy identifiers to defined policies, for example:

```
{
  "(global)": {
    "retention": {
      "keepLatest": 10,
      "keepHourly": 48,
      ...
    },
    ...
  },
  "foo@bar:/home/foobar": {
     "retention": {
      "keepLatest": 5,
      "keepHourly": 24,
      ...
    },
    ...
  }
}
```

You can optionally limit which policies are imported or exported by specifying the policy identifiers as arguments to the `kopia policy import` and `kopia policy export` commands:

```
$ kopia policy import --from-file import.json "(global)" "foo@bar:/home/foobar"
$ kopia policy export --to-file export.json "(global)" "foo@bar:/home/foobar"
```

Both commands support using stdin/stdout:

```
$ cat file.json | kopia policy import
$ kopia policy export > file.json
```

You can use the `--delete-other-policies` flag to delete all policies that are not imported. This command would delete any policy besides `(global)` and `foo@bar:/home/foobar`:

```
$ kopia policy import --from-file import.json --delete-other-policies "(global)" "foo@bar:/home/foobar"
```

Examining Repository Structure

Kopia CLI provides low-level commands to examine the contents of repository, perform maintenance actions, and get deeper insight into how the data is laid out.

> REMINDER: This guide does not cover all of the commands available via Kopia CLI. Refer to the command-line reference page to learn about all the available commands.

BLOBs

We can list the files in a repository using `kopia blob ls`, which shows how Kopia manages snapshots. We can see that repository contents are grouped into pack files (starting with `p`) and indexed using index files (starting with `n`). Both index and pack files are encrypted:

```
$ kopia blob ls
kopia.repository                             636 2019-06-22 20:03:25 PDT
n16f6b7257610be4826396ce2be1fb302         667941 2019-06-22 20:05:30 PDT
n71861ad243bea4c2010001cf5cba1fbe            110 2019-06-22 20:03:29 PDT
p0b31020332e1fe97856faf8aa9c5cf4c       21015032 2019-06-22 20:05:13 PDT
p0df646619c8d1a25415b1b9dc9329d19       22659618 2019-06-22 20:04:36 PDT
p1bde010809b285b657cc130c34c01687       21606909 2019-06-22 20:04:30 PDT
p4dd3b780360b6889e172776fa65be836       23688119 2019-06-22 20:04:23 PDT
p549295a15c0f481ab46f4e4f18372bfc        6936758 2019-06-22 20:05:30 PDT
p61418c5b66c2ca06a51c4af3205e7006           4232 2019-06-22 20:03:29 PDT
p8a6e4f70c0fbbfda99b2e1100647606f       22930102 2019-06-22 20:04:43 PDT
p8f482528fff99ad5c23264873582545f       21261488 2019-06-22 20:05:21 PDT
p946d1fe528bde84d826cf177168d07b3       21015794 2019-06-22 20:04:15 PDT
pa0223f20fc6776b24d25b122cc9ac5f3       26497385 2019-06-22 20:05:07 PDT
pca01737879ac83f9e613c11f52d58349       21520893 2019-06-22 20:04:59 PDT
pcc4bc6fc2b960091755dc0c4704669f1       21656682 2019-06-22 20:05:27 PDT
pdec2a5b599acef1e1ce13ace25e914cd       21332440 2019-06-22 20:04:02 PDT
pf0239202fea978abf1cae4ca45d395b1       23756797 2019-06-22 20:03:54 PDT
pf1d1f00141f830ee34c89797e56011c2       23663212 2019-06-22 20:04:09 PDT
pfbaba06c70b6aed2c0ed3aa9c709dc47       22823458 2019-06-22 20:04:50 PDT
```

Content-Addressable Block Storage

To list individual contents stored in a repository, use `kopia content list`:

```
$ kopia content list
00020136867452b90a3c65a029e7d08c
00029ee689919a13dbf2d588c0986530
00075db1b05e83ba529bc0e06de76ca0
00077bc1c333ddc475d66c45d4645210
0009d14c0a50ceca3519d522137b3a68
000df512dd55ac9749dd0670efa49c7d
001130562b64643ca255396a95a9f2a4
001375d9065eef57b0f21dfe11590227
0015561b6e70feff623eede49f95da73
00166bad1a69484992e32dbd2b869a8a
001828cc591d0bec1301b15266c5c530
...
kfec1215488eaf6acef4558e87ff343a9
kfec7264b36fabdf072de939f622b4452
kff0a0d64969cde306f4dd2c95ca2df6f
kff646ec33a3c24701aabb22f62d60e43
kff74067fa7a2bf14681aee73eb08330d
kff8824282ccc64f68b7b39aacdbb6ceb
kff99aac1cd37371cfc521753e2a1d424
m08ef40b314fb7f08c7be222a79485cc1
m259ec63a4a0137b7ce2801cc47012ffa
m5cf33f9416a435478dea4040b8049f51
m81bd005052e582e821df831c36138d76
m831980997bceabebefa095914a600a1b
m8401800f69795ed0137365c3e6f627bc
...
```

Manifest Storage

To list manifests (snapshot manifests and policies) stored in a repository, use `kopia manifest list`:

```
$ kopia manifest list
7898f47e36bad80a6d5d90f06ef16de6        170 2019-06-22 20:03:29 PDT type:policy policyType:global
```

```
9a622e33ab134ef440f76ed755f79c2f        802 2019-06-22 20:05:28 PDT type:snapshot hostname:jareks-mbp path:/Users/jarek/Projects/Kopia username:jarek
2d73b31af65d4ac7196641eeea9c475c        755 2019-06-22 20:15:53 PDT type:snapshot hostname:jareks-mbp path:/Users/jarek/Projects/Kopia/site username:jarek
8a45c3b079cf5e7b99fb855a3701607a        762 2019-06-22 20:15:58 PDT type:snapshot hostname:jareks-mbp path:/Users/jarek/Projects/Kopia/site username:jarek
ed30d264bcf795bd6648540fdeebdb31        761 2019-06-22 20:21:39 PDT type:snapshot hostname:jareks-mbp path:/Users/jarek/Projects/Kopia/site username:jarek
a1646120c7a2450cd9e77fd98369d260        761 2019-06-22 20:21:43 PDT type:snapshot hostname:jareks-mbp path:/Users/jarek/Projects/Kopia/site username:jarek
38f1987e1ea434e161111abce86212ed        761 2019-06-22 20:21:45 PDT type:snapshot hostname:jareks-mbp path:/Users/jarek/Projects/Kopia/site username:jarek
2339ab4739bb29688bf26a3a841cf68f         63 2019-06-22 21:19:41 PDT type:policy hostname:jareks-mbp path:/Users/jarek/Projects/Kopia/site/node_modules policyType:path username:ja
856ed6f6cc5cd522d23718e6315cf51e        757 2019-06-22 21:20:37 PDT type:snapshot hostname:jareks-mbp path:/Users/jarek/Projects/Kopia/site username:jarek
63fc854c283ad63cafbca54eaa4509e9        102 2019-06-22 21:22:20 PDT type:policy hostname:jareks-mbp path:/Users/jarek/Projects/Kopia/site policyType:path username:jarek
```

To examine individual manifests, use `kopia manifest show`:

```
$ kopia manifest show 2d73b31af65d4ac7196641eeea9c475c
```

```
// id: 2d73b31af65d4ac7196641eeea9c475c
// length: 755
// modified: 2019-06-22 20:15:53 PDT
// label path:/Users/jarek/Projects/Kopia/site
// label type:snapshot
// label username:jarek
// label hostname:jareks-mbp
{
  "source": {
    "host": "jareks-mbp",
    "userName": "jarek",
    "path": "/Users/jarek/Projects/Kopia/site"
  },
  "description": "",
  "startTime": "2019-06-22T20:15:51.603328-07:00",
  "endTime": "2019-06-22T20:15:53.038247-07:00",
  "stats": {
    "content": {
      "readBytes": 863,
      "decryptedBytes": 799,
      "hashedBytes": 63685985,
      "readContents": 2,
      "hashedContents": 13315
    },
    "dirCount": 798,
    "fileCount": 12500,
    "totalSize": 61414615,
    "excludedFileCount": 0,
    "excludedTotalSize": 0,
    "excludedDirCount": 0,
    "cachedFiles": 0,
    "nonCachedFiles": 12500,
    "readErrors": 0
  },
  "rootEntry": {
    "name": "site",
    "type": "d",
    "mode": "0755",
    "mtime": "2019-06-22T19:01:45.936555202-07:00",
    "uid": 501,
    "gid": 20,
    "obj": "kb9a8420bf6b8ea280d6637ad1adbd4c5",
    "summ": {
      "size": 61414615,
      "files": 12500,
      "dirs": 798,
      "maxTime": "2019-06-22T20:15:45.301289096-07:00"
    }
  }
}
```

Cache

For better performance, Kopia maintains local cache directory where most-recently used blocks are stored. You can examine the cache by using `kopia cache info`:

```
$ kopia cache info
/Users/jarek/Library/Caches/kopia/e470f963ef9528a1/contents: 3 files 7 KB (limit 5.2 GB)
/Users/jarek/Library/Caches/kopia/e470f963ef9528a1/indexes: 12 files 670.8 KB
/Users/jarek/Library/Caches/kopia/e470f963ef9528a1/metadata: 2006 files 3.9 MB (limit 0 B)
```

To clear the cache, use `kopia cache clear`:

```
$ kopia cache clear
```

To set caching parameters, such as maximum size of each cache, use `kopia cache set`:

```
$ kopia cache set --metadata-cache-size-mb=500
21:38:25.024 [kopia/cli] changing metadata cache size to 500 MB
```

More information on `cache` commands is available in the help docs.

Last modified October 28, 2024: feat(cli): add policy import/export commands to im-/export policies from/to json (#4020) (a9e178ed)

Download & Installation

Two Variants of Kopia

Kopia is a standalone binary and can be used through a command-line interface (CLI) or a graphical user interface (GUI).

- If you want to use Kopia via CLI, you will install the `kopia` binary; when you want to use Kopia, you will call the `kopia` binary (along with Kopia commands) in a terminal/command prompt window or within a script.
- If you want to use Kopia via GUI, you will install `KopiaUI`, the name of the Kopia GUI. The installer for KopiaUI comes with the `kopia` binary and a graphical user interface called `KopiaUI` - a wrapper for the `kopia` binary. `KopiaUI` runs the `kopia` binary and associated commands as necessary, so you do not need to use the command-line interface.

  NOTE: `KopiaUI` is available both as a web-based application and a desktop application. The web-based application is available when you run Kopia in server mode. For users who will be using Kopia to backup their individual machines and not running Kopia in server mode, you will use the desktop application. If you do not understand Kopia server mode, do not worry; download `KopiaUI` from the links below, and you will get the desktop application by default.

Both the CLI and GUI versions of Kopia use the same `kopia` binary, so you are getting the same features regardless of which variant you decide to go with (since the `kopia` binary is the workhorse). However, some advanced features are available through CLI but have not yet been added to `KopiaUI`. Right now, `KopiaUI` allows you to access all the essential features of Kopia that are required to backup/restore data: create and connect to repositories (including encryption), set policies (including compression, scheduling automatic snapshots, and snapshot retention), create snapshots, restore snapshots, automatically run maintenance, and install Kopia updates. If you use `KopiaUI` and you want access to advanced features that are not yet available in `KopiaUI`, you can easily run the commands for those features via CLI by calling the `kopia` binary that comes with `KopiaUI`. In other words, using Kopia GUI does not restrict you from using Kopia CLI as well.

Kopia CLI is recommended only if you are comfortable with command-line interfaces (e.g., power users, system administrators, etc.). If you are uncomfortable with the command-line, use Kopia GUI. Although more limited than Kopia CLI, Kopia GUI is still very powerful and allows you to use Kopia to back up/restore your data easily.

Kopia Download Links

The following installation options are available for the latest stable version of Kopia:

- Official Releases
- Windows CLI (Scoop)
- Windows GUI (`KopiaUI`)
- macOS CLI Homebrew
- macOS GUI Homebrew
- macOS GUI (`KopiaUI`)
- Debian/Ubuntu Linux (APT Repository, both CLI and `KopiaUI`)
- RedHat/CentOS/Fedora Linux (Linux YUM Repository, both CLI and `KopiaUI`)
- Arch Linux/Manjaro (AUR)
- OpenBSD
- FreeBSD
- Docker Images

The following options are available if you like to test the beta and unreleased versions of Kopia:

- Test Builds on GitHub
- Windows CLI (Scoop) offers `test-builds` bucket
- macOS CLI Homebrew offers `test-builds` TAP
- Debian/Ubuntu Linux (APT Repository) offers `unstable` channel
- RedHat/CentOS/Fedora Linux (Linux YUM Repository) offers `unstable` channel
- Source Code - see compilation instructions

Installing Kopia

CLI and GUI packages are available for:

- Windows 10 or later, 64-bit (CLI binary, GUI installer {`KopiaUI`}, and Scoop package)
- macOS 10.11 or later, 64-bit (CLI binary, GUI installer {`KopiaUI`}, and Homebrew package)
- Linux - `amd64`, `armhf` or `arm64` (CLI binary and `KopiaUI` available via RPM and DEB repositories)

Windows CLI installation using Scoop

On Windows, Kopia CLI is available as a Scoop package, which automates installation and upgrades.

Using Scoop, installing Kopia is as easy as:

```
> scoop bucket add kopia https://github.com/kopia/scoop-bucket.git
> scoop install kopia
```

See the Scoop Website for more information.

Alternatively, to install the latest unreleased version of Kopia use the following bucket instead:

```
> scoop bucket add kopia https://github.com/kopia/scoop-test-builds.git
```

Windows GUI installation

The installer of `KopiaUI` is available on the releases page. Simply download the file named `KopiaUI-Setup-X.Y.Z.exe` (where `X.Y.Z` is the version number), double click the file, and follow on-screen prompts.

macOS CLI using Homebrew

On macOS, you can use Homebrew to install and keep Kopia up-to-date.

To install:

```
$ brew install kopia
```

To upgrade Kopia:

```
$ brew upgrade kopia
```

Alternatively, to install the latest unreleased version of Kopia use the following TAP instead:

```
$ brew install kopia/test-builds/kopia
```

macOS GUI using Homebrew

On macOS, you can use Homebrew to install and keep Kopia up-to-date.

To install:

```
$ brew install kopiaui
```

To upgrade Kopia:

```
$ brew upgrade kopiaui
```

macOS GUI installer

MacOS package with `KopiaUI` is available in DMG and ZIP formats on the releases page.

Linux installation using APT (Debian, Ubuntu)

Kopia offers APT repository compatible with Debian, Ubuntu and other similar distributions.

To begin, install the GPG signing key to verify authenticity of the releases.

```
curl -s https://kopia.io/signing-key | sudo gpg --dearmor -o /etc/apt/keyrings/kopia-keyring.gpg
```

Register APT source:

```
echo "deb [signed-by=/etc/apt/keyrings/kopia-keyring.gpg] http://packages.kopia.io/apt/ stable main" | sudo tee /etc/apt/sources.list.d/kopia.list
sudo apt update
```

> By default, the **stable** channel provides official stable releases. If you prefer you can also select **testing** channel (which also provides release candidates and is generally stable) or **unstable** which includes all latest changes, but may not be stable.

Finally, install Kopia or KopiaUI:

```
sudo apt install kopia
sudo apt install kopia-ui
```

Linux installation using RPM (RedHat, CentOS, Fedora)

Kopia offers RPM repository compatible with RedHat, CentOS, Fedora and other similar distributions.

To begin, install the GPG signing key to verify authenticity of the releases.

```
rpm --import https://kopia.io/signing-key
```

Install Yum repository:

```
cat <<EOF | sudo tee /etc/yum.repos.d/kopia.repo
[Kopia]
name=Kopia
baseurl=http://packages.kopia.io/rpm/stable/\$basearch/
gpgcheck=1
enabled=1
gpgkey=https://kopia.io/signing-key
EOF
```

> By default, the **stable** channel provides official stable releases. If you prefer you can also select **testing** channel (which also provides release candidates and is generally stable) or **unstable** which includes all latest changes, but may not be stable.

Finally, install Kopia or KopiaUI:

```
sudo yum install kopia
sudo yum install kopia-ui
```

Linux installation using AUR (Arch, Manjaro)

Those using Arch-based distributions have the option of building Kopia from source or installing pre-complied binaries:

To build and install Kopia from source:

```
git clone https://aur.archlinux.org/kopia.git
cd kopia
makepkg -si
```

or if you use an AUR helper such as yay:

```
yay -S kopia
```

To install the binary version:

```
git clone https://aur.archlinux.org/kopia-bin.git
cd kopia-bin
makepkg -si
```

or if you use an AUR helper such as yay:

```
yay -S kopia-bin
```

OpenBSD installation via ports

OpenBSD has kopia in ports, which means it gets built as packages in snapshots for several platforms (amd64, arm64, mips64 and i386).

To install the kopia package, run:

```
# pkg_add kopia
```

To build Kopia from ports yourself, cd /usr/ports/sysutils/kopia and follow the Ports guide on building ports as usual.

FreeBSD installation via ports

FreeBSD now has kopia in ports, which means it gets built as packages in snapshots for several platforms (amd64, arm64 and i386) and will appear as a package for supported versions.

To install the port:

```
cd /usr/ports/sysutils/kopia/ && make install clean
```

To add the package, run one of these commands:

```
pkg install sysutils/kopia
pkg install kopia
```

For more information on ports, see the FreeBSD Handbook.

Docker Images

Kopia provides pre-built Docker container images for `amd64`, `arm64` and `arm` on DockerHub.

The following tags are available:

- `latest` - tracks the latest stable release
- `testing` - tracks the latest stable or pre-release (such as a beta or release candidate)
- `unstable` - tracks the latest unstable nightly build
- `major.minor` - latest patch release for a given major and minor version (e.g. `0.8`)
- `major.minor.patch` - specific stable release

In order to run Kopia in a docker container, you must:

- provide repository password via `KOPIA_PASSWORD` environment variable
- mount `/app/config` directory in which Kopia will look for `repository.config` file
- (recommended) mount `/app/cache` directory in which Kopia will be keeping a cache of downloaded data
- (optional) mount `/app/logs` directory in which Kopia will be writing logs
- (optional), **only** when using `rclone` provider mount `/app/rclone` directory in which RClone will look for `rclone.conf` file
- mount any directory used for locally-attached `repository`
- mount `/tmp` directory to browse mounted snapshots
    - the directory must have `:shared` property, so mounts can be browsable by host system
- for nginx reverse proxy, use: `grpc_pass grpcs://container_ip:container_port` instead of `proxy_pass`

Invocation of `kopia/kopia` in a container will be similar to the following minimal example:

```
$ docker pull kopia/kopia:latest
$ docker run -e KOPIA_PASSWORD \
    -v /path/to/config/dir:/app/config \
    -v /path/to/cache/dir:/app/cache \
    -v /path/to/logs/dir:/app/logs \
    -v /path/to/repository/dir:/repository \
    -v /path/to/tmp/dir:/tmp:shared \
```

In addition to creating the docker container with *docker run*, the following docker-compose provides an example for setting up a minimal container in server mode including the web interface. You can access the interface via http://localhost:51515 or at the server's IP address after starting the container.

> NOTE Kopia provides a vast of parameters to configure the container. Please check our docker-compose for more details.

```
version: '3.7'
services:
    kopia:
        image: kopia/kopia:latest
        hostname: Hostname
        container_name: Kopia
        restart: unless-stopped
        ports:
            - 51515:51515
        # Setup the server that provides the web gui
        command:
            - server
            - start
            - --disable-csrf-token-checks
            - --insecure
            - --address=0.0.0.0:51515
            - --server-username=USERNAME
            - --server-password=SECRET_PASSWORD
        environment:
            # Set repository password
            KOPIA_PASSWORD: "SECRET"
            USER: "User"
        volumes:
            # Mount local folders needed by kopia
            - /path/to/config/dir:/app/config
            - /path/to/cache/dir:/app/cache
            - /path/to/logs/dir:/app/logs
            # Mount local folders to snapshot
            - /path/to/data/dir:/data:ro
            # Mount repository location
            - /path/to/repository/dir:/repository
            # Mount path for browsing mounted snaphots
            - /path/to/tmp/dir:/tmp:shared
```

Because the Docker environment uses random hostnames for its containers, it is recommended to explicitly set them using `hostname`. The name will be persisted in a configuration file and used afterwards.

NOTE Kopia within a container overrides default values of some environment variables, see our dockerfile for more details.

Verifying package integrity

When downloading from GitHub it's recommended to verify SHA256 checksum of the binary and comparing that to `checksums.txt`. For extra security you may want to verify that the checksums have been signed by official Kopia builder, by running GPG:

```
# Import official signing key
$ curl https://kopia.io/signing-key | gpg --import -

# Verify that file checksums are ok
$ sha256sum --check checksums.txt

# Verify signature file
$ gpg --verify checksums.txt.sig
gpg: assuming signed data in 'checksums.txt'
gpg: Signature made Thu Apr 15 22:02:31 2021 PDT
gpg:                using RSA key 7FB99DFD47809F0D5339D7D92273699AFD56A556
gpg: Good signature from "Kopia Builder <[email protected]>" [ultimate]
```

You need to make the download binary executable (Linux/macOS only) and move it to a location that's in your system `PATH` for easy access:

On Linux/macOS run:

```
# make the file executable
$ chmod u+x path/to/kopia
# move to a location in system path
$ sudo mv path/to/kopia /usr/local/bin/kopia
```

Compilation From Source

If you have Go 1.16 or newer, you may download and build Kopia yourself. No special setup is necessary, other than the Go compiler and git. You can simply run:

```
$ go install github.com/kopia/kopia@latest
```

The resulting binary will be available in `$HOME/go/bin`. Note that this will produce basic binary that has all the features except support for HTML-based UI. To build full binary, download the source from GitHub and run:

```
$ make install
```

Additional information about building Kopia from source is available at https://github.com/kopia/kopia/blob/master/BUILD.md

Last modified August 23, 2024: docs(general): update supported Windows version (#4071) (87766e32)

Repositories

Kopia allows you to save your encrypted backups (which are called `snapshots` in Kopia) to a variety of storage locations, and in Kopia a storage location is called a `repository`. Kopia supports all of the following storage locations:

> PRO TIP: You pick the storage locations you want to use. Kopia plays no role in selecting your storage locations. This means you must provision, setup, and pay (the storage provider) for whatever storage locations you want to use **before** you create a `repository` for that storage location in Kopia.

- Amazon S3 and S3-compatible Cloud Storage
  - Kopia supports all cloud storage platforms that support the S3 API
  - Kopia supports object locking and hot, cold, and archive storage classes for any cloud storage that supports the features using the S3 API
- Azure Blob Storage
- Backblaze B2
- Google Cloud Storage
- Google Drive
  - Kopia supports Google Drive natively and through Kopia's Rclone option (see below)
  - Native support for Google Drive in Kopia is currently experimental
  - Native Google Drive support operates differently than Kopia's support for Google Drive through Rclone; you will not be able to use the two interchangeably, so pick one
- All remote servers or cloud storage that support WebDAV
- All remote servers or cloud storage that support SFTP
- Some of the cloud storages supported by Rclone
  - Rclone is a (free and open-source) third-party program that you must download and setup separately before you can use it with Kopia
  - Once you setup Rclone, Kopia automatically manages and runs Rclone for you, so you do not need to do much beyond the initial setup, aside from enabling Rclone's self-update feature so that it stays up-to-date
  - Kopia's Rclone support is experimental: not all the cloud storages supported by Rclone have been tested to work with Kopia, and some may not work with Kopia; Kopia has been tested to work with Dropbox, OneDrive, and Google Drive through Rclone
- Your local machine and any network-attached storage or server
- Your own remote server by setting up a Kopia Repository Server

> PRO TIP: Many cloud storage providers offer a variety of storage tiers that may (or may not) help decrease your cost of cloud storage, depending on your use case. See the storage tiers documentation to learn the different types of files Kopia stores in repositories and which one of these file types you can possibly move to archive tiers, such as Amazon Deep Glacier.

Amazon S3 and S3-compatible Cloud Storage

Creating an Amazon S3 or S3-compatible storage `repository` is done differently depending on if you use Kopia GUI or Kopia CLI.

Kopia GUI

Select the `Amazon S3 and Compatible Storage` option in the `Repository` tab in `KopiaUI`. Then, follow on-screen instructions. You will need to enter `Bucket` name, `Server Endpoint`, `Access Key ID`, and `Secret Access Key`. You can optionally enter an `Override Region` and `Session Token`.

> NOTE: Some S3-compatible cloud storage may have slightly different names for bucket, endpoint, access key, secret key, region, and session token. This will vary between cloud storages. Read the help documentation for the cloud storage you are using to find the appropriate values. You can typically find this information by searching for the S3 API settings for your cloud storage.

You will next need to enter the repository password that you want. Remember, this password is used to encrypt your data, so make sure it is a secure password! At this same password screen, you have the option to change the `Encryption` algorithm, `Hash` algorithm, `Splitter` algorithm, `Repository Format`, `Username`, and `Hostname`. Click the `Show Advanced Options` button to access these settings. If you do not understand what these settings are, do not change them because the default settings are the best settings.

> NOTE: Some settings, such as object locking and actions, can only be enabled when you create a new `repository` using command-line (see next section). However, once you create the `repository` via command-line, you can use the `repository` as normal in Kopia GUI: just connect to the `repository` as described above after you have created it in command-line.

Once you do all that, your repository should be created and you can start backing up your data!

Kopia CLI

Creating a Repository

You must use the `kopia repository create s3` command to create a `repository`:

```
$ kopia repository create s3 \
        --bucket=... \
        --access-key=... \
        --secret-access-key=...
```

At a minimum, you will need to enter the bucket name, access key, and secret access key. If you are not using Amazon S3 and are using an S3-compatible storage, you will also need to enter the endpoint and may need to enter the `region`. There are also various other options (such as object locking and actions) you can change or enable – see the help docs for more information.

You will be asked to enter the repository password that you want. Remember, this password is used to encrypt your data, so make sure it is a secure password!

Connecting to Repository

After you have created the `repository`, you connect to it using the `kopia repository connect s3` command. Read the help docs for more information on the options available for this command.

Azure Blob Storage

Creating an Azure Blob Storage `repository` is done differently depending on if you use Kopia GUI or Kopia CLI.

Kopia GUI

Select the `Azure Blob Storage` option in the `Repository` tab in `KopiaUI`. Then, follow on-screen instructions. You will need to enter `Container` name, `Storage Account` name and either `Access Key` or `SAS Token`. You can optionally enter an `Azure Storage Domain`.

You will next need to enter the repository password that you want. Remember, this password is used to encrypt your data, so make sure it is a secure password! At this same password screen, you have the option to change the `Encryption` algorithm, `Hash` algorithm, `Splitter` algorithm, `Repository Format`, `Username`, and `Hostname`. Click the `Show Advanced Options` button to access these settings. If you do not understand what these settings are, do not change them because the default settings are the best settings.

> NOTE: Some settings, such as actions, can only be enabled when you create a new `repository` using command-line (see next section). However, once you create the `repository` via command-line, you can use the `repository` as normal in Kopia GUI: just connect to the `repository` as described above after you have created it in command-line.

Once you do all that, your repository should be created and you can start backing up your data!

Kopia CLI

Creating a Repository

You must use the `kopia repository create azure` command to create a `repository`:

```
$ kopia repository create azure \
        --container=... \
        --storage-account=... \
        --storage-key=...
```

OR

```
$ kopia repository create azure \
        --container=... \
        --storage-account=... \
        --sas-token=...
```

At a minimum, you will need to enter the container name, storage account name, and either your Azure account access key/storage key or a SAS token. There are also various other options (such as actions) you can change or enable – see the help docs for more information.

You will be asked to enter the repository password that you want. Remember, this password is used to encrypt your data, so make sure it is a secure password!

Connecting to Repository

After you have created the `repository`, you connect to it using the `kopia repository connect azure` command. Read the help docs for more information on the options available for this command.

Backblaze B2

Creating a Backblaze B2 `repository` is done differently depending on if you use Kopia GUI or Kopia CLI.

> NOTE: Currently, object locking is supported for B2 but only through Kopia's S3-compatible storage `repository` and not through the B2 `repository` option. However, B2 is fully S3 compatible, so you can setup your B2 account via Kopia's S3 `repository` option. To use B2 storage with the S3 `repository` option the `--endpoint` argument must be specified with the appropriate B2 endpoint. This endpoint can be found on the buckets page of the B2 web interface and follows the pattern `s3.<region>.backblazeb2.com`.

Kopia GUI

Select the `Backblaze B2` option in the `Repository` tab in `KopiaUI`. Then, follow on-screen instructions. You will need to enter `B2 Bucket` name, `Key ID`, and application `Key`.

You will next need to enter the repository password that you want. Remember, this password is used to encrypt your data, so make sure it is a secure password! At this same password screen, you have the option to change the `Encryption` algorithm, `Hash` algorithm, `Splitter` algorithm, `Repository Format`, `Username`, and `Hostname`. Click the `Show Advanced Options` button to access these settings. If you do not understand what these settings are, do not change them because the default settings are the best settings.

> NOTE: Some settings, such as actions, can only be enabled when you create a new `repository` using command-line (see next section). However, once you create the `repository` via command-line, you can use the `repository` as normal in Kopia GUI: just connect to the `repository` as described above after you have created it in command-line.

Once you do all that, your repository should be created and you can start backing up your data!

Kopia CLI

Creating a Repository

You must use the `kopia repository create b2` command to create a `repository`:

```
$ kopia repository create b2 \
        --bucket=... \
        --key-id=... \
        --key=...
```

There are also various other options (such as actions) you can change or enable – see the help docs for more information.

You will be asked to enter the repository password that you want. Remember, this password is used to encrypt your data, so make sure it is a secure password!

Connecting to Repository

After you have created the `repository`, you connect to it using the `kopia repository connect b2` command. Read the help docs for more information on the options available for this command.

Google Cloud Storage

Creating a Google Cloud Storage `repository` is done differently depending on if you use Kopia GUI or Kopia CLI.

Kopia GUI

Select the `Google Cloud Storage` option in the `Repository` tab in `KopiaUI`. Then, follow on-screen instructions. You will need to enter the `GCS Bucket` name and enter the path to where on your machine you have saved the Google Cloud Storage `Credentials File`. The credentials file can be obtained by creating a Google Cloud Service Account that allows you to access your storage bucket and then downloading the JSON key file for that service account. You enter the path to this JSON key file in the `Credentials File` textbox in `KopiaUI`.

You will next need to enter the repository password that you want. Remember, this password is used to encrypt your data, so make sure it is a secure password! At this same password screen, you have the option to change the `Encryption` algorithm, `Hash` algorithm, `Splitter` algorithm, `Repository Format`, `Username`, and `Hostname`. Click the `Show Advanced Options` button to access these settings. If you do not understand what these settings are, do not change them because the default settings are the best settings.

> NOTE: Some settings, such as actions, can only be enabled when you create a new `repository` using command-line (see next section). However, once you create the `repository` via command-line, you can use the `repository` as normal in Kopia GUI: just connect to the `repository` as described above after you have created it in command-line.

Once you do all that, your repository should be created and you can start backing up your data!

Kopia CLI

Creating a Repository

There are three methods to create a `repository` for Google Cloud Storage: one that requires you to install Google Cloud SDK; the other method allows you to generate credentials without Google Cloud SDK; and the third method allows you to use Google Cloud Storage through Kopia's S3 `repository` option:

Method #1: Installing Google Cloud SDK

1. Create a storage bucket in Google Cloud Console
2. Install Google Cloud SDK
3. Log in with credentials that have permissions to the bucket

```
$ gcloud auth application-default login
```

After these preparations, we can create a Kopia `repository` (assuming bucket named `kopia-test-123`) using the `kopia repository create gcs` command:

```
$ kopia repository create gcs --bucket kopia-test-123
```

There are also various other options (such as actions) you can change or enable – see the help docs for more information.

You will be asked to enter the repository password that you want. Remember, this password is used to encrypt your data, so make sure it is a secure password!

Method #2: Creating a Service Account and Using the JSON Key File

1. Create a storage bucket in Google Cloud Console
2. Create a Google Cloud Service Account that allows you to access your storage bucket. Directions are available on Google Cloud's website. Make sure to download the JSON key file for your service account and keep it safe.

After these preparations, we can create a Kopia `repository` (assuming bucket named `kopia-test-123`) using the `kopia repository create gcs` command:

```
$ kopia repository create gcs --credentials-file="/path/to/your/credentials/file.json" --bucket kopia-test-123
```

There are also various other options (such as actions) you can change or enable – see the help docs for more information.

You will be asked to enter the repository password that you want. Remember, this password is used to encrypt your data, so make sure it is a secure password!

Method #3: Enabling Amazon S3 Interoperability in Google Cloud Storage

1. Create a storage bucket in Google Cloud Console
2. Go to Settings and then Interoperability in your Google Cloud Storage account
3. Enable your project under `Default project for interoperable access` and generate access keys for this project – you will generate both access key and secret key, just like if you were using Amazon S3

After these preparations, we can create a Kopia `repository` (assuming bucket named `kopia-test-123`) using the `kopia repository create s3` command:

```
$ kopia repository create s3 --endpoint="storage.googleapis.com" --bucket="kopia-test-123" --access-key="access/key/here" --secret-access-key="secret/key/here"
```

There are also various other options (such as actions) you can change or enable – see the help docs for more information.

You will be asked to enter the repository password that you want. Remember, this password is used to encrypt your data, so make sure it is a secure password!

Connecting to Repository

After you have created the `repository`, you connect to it using the `kopia repository connect gcs` command or the `kopia repository connect s3` command, depending on whichever way you setup the Google Cloud Storage `repository`. Read the help docs for `repository connect gcs` or the help docs for `repository connect s3` for more information on the options available for these commands.

Credential permissions

The following permissions are required when in readonly mode:

```
storage.buckets.get
storage.objects.get
storage.objects.list
```

When in normal read-write mode the following additional permissions are required:

```
storage.objects.update
storage.objects.create
storage.objects.delete
```

If using ransomware protection then the following additional permission is required:

```
storage.objects.setRetention
```

Google Drive

Kopia supports Google Drive in two ways: natively and through Kopia's Rclone `repository` option. Native Google Drive support is currently only available through Kopia CLI; Kopia GUI users need to use Kopia's Rclone `repository` option.

Below we describe how to setup a `repository` via Kopia CLI using native Google Drive support, and you will need a Google Cloud Storage service account for the process. You do not need a Google Cloud Storage service account if you create a Google Drive `repository` in Kopia through Rclone; to do that, read the Rclone section of this page.

> WARNING: Native Google Drive support is experimental.

Kopia uses a Google Drive folder that you provide to store all the files in the `repository`. Kopia will only access files in this folder, and using Kopia does not impact your other Google Drive files. It is recommended that you let Kopia manage this folder and do not upload any other content to this folder.

Kopia CLI

Creating a Repository

Here's a high-level rundown of what you will need to do to create a Google Drive `repository`:

1. Create or use an existing Google Drive folder for the repository.

2. Create a Google Cloud Service Account for Kopia.

3. Share the Google Drive folder with your new service account so that it allows Kopia to access the folder.

Ready? Here are the step-by-step instructions:

1. Create a Google Cloud project, or use an existing one.

2. Enable the Google Drive API for your project.

3. Create a service account. After enabling the API, you should be now prompted to create credentials. Choose `Service account` from the options, and give it a name. Note down the service account email.

4. Create a key for the service account. You can do this by viewing the service account, navigating to the `Keys` tab, and clicking `Add Key` -> `Create new key`. You should choose `JSON` for the key type. Save the file on your computer.

5. Create or pick an existing Google Drive folder. The browser URL should look something like `https://drive.google.com/drive/u/0/folders/[folder_id]`. Note down the last part of the URL. That's your folder ID.

6. Share the Google Drive folder with the service account. Open the share dialog for the folder by right-clicking the folder in Google Drive, and put in the service account email. You should choose the

> `Editor` as the access role.

After these preparations, we can create a Kopia `repository` (assuming the folder ID is `z63ZZ1Npv3OFvDPwU3dX0w`) using the [`kopia repository create gdrive` command](#):

```
$ kopia repository create gdrive \
        --folder-id z63ZZ1Npv3OFvDPwU3dX0w \
        --credentials-file="<where-you-have-stored-the-json-key-file>"
```

There are also various other options (such as [actions](#)) you can change or enable – see the [help docs](#) for more information.

You will be asked to enter the repository password that you want. Remember, this [password is used to encrypt your data](#), so make sure it is a secure password!

If you view your folder on Google Drive, you should see that Kopia has created the skeleton of the repository with a `kopia.repository` file and a couple of others. Kopia will store all the files for your snapshots in this folder.

Connecting to Repository

After you have created the `repository`, you connect to it using the [`kopia repository connect gdrive` command](#). Read the [help docs](#) for more information on the options available for this command.

WebDAV

Creating a WebDAV `repository` is done differently depending on if you use Kopia GUI or Kopia CLI.

Kopia GUI

Select the `WebDAV Server` option in the `Repository` tab in `KopiaUI`. Then, follow on-screen instructions. You will need to enter `WebDAV Server URL`, `Username`, and `Password.

You will next need to enter the repository password that you want. This password can be whatever you want, it does not need to be the same as your WebDAV password. In fact, it should not be the same! Remember, this [password is used to encrypt your data](#), so make sure it is a secure password! At this same password screen, you have the option to change the `Encryption` algorithm, `Hash` algorithm, `Splitter` algorithm, `Repository Format`, `Username`, and `Hostname`. Click the `Show Advanced Options` button to access these settings. If you do not understand what these settings are, do not change them because the default settings are the best settings.

> NOTE: Some settings, such as [actions](#), can only be enabled when you create a new `repository` using command-line (see next section). However, once you create the `repository` via command-line, you can use the `repository` as normal in Kopia GUI: just connect to the `repository` as described above after you have created it in command-line.

Once you do all that, your repository should be created and you can start backing up your data!

Kopia CLI

Creating a Repository

You must use the [`kopia repository create webdav` command](#) to create a `repository`:

```
$ kopia repository create webdav \
        --url=... \
        --webdav-password=... \
        --webdav-username=...
```

At a minimum, you will need to enter the WebDAV server URL, username, and password. There are also various other options (such as [actions](#)) you can change or enable – see the [help docs](#) for more information.

You will be asked to enter the repository password that you want. This password can be whatever you want, it does not need to be the same as your WebDAV password. In fact, it should not be the same! Remember, this [password is used to encrypt your data](#), so make sure it is a secure password!

Connecting to Repository

After you have created the `repository`, you connect to it using the [`kopia repository connect webdav` command](#). Read the [help docs](#) for more information on the options available for this command.

SFTP

Creating a SFTP or SSH `repository` is done differently depending on if you use Kopia GUI or Kopia CLI.

Kopia GUI

Select the `SFTP Server` option in the `Repository` tab in `KopiaUI`. Then, follow on-screen instructions. You will need to enter `Host`, `User`, `Path`, and either `Password` or `Path to key file`. You can optionally enter `Path to known_hosts file`.

If the connection to SFTP server does not work, checking the option for `Launch external password-less SSH command` which will launch an external `ssh` process that supports more connectivity options and may be needed for some hosts.

You will next need to enter the repository password that you want. This password can be whatever you want, it does not need to be the same as your SFTP password. In fact, it should not be the same! Remember, this [password is used to encrypt your data](#), so make sure it is a secure password! At this same password screen, you have the option to change the `Encryption` algorithm, `Hash` algorithm, `Splitter` algorithm, `Repository Format`, `Username`, and `Hostname`. Click the `Show Advanced Options` button to access these settings. If you do not understand what these settings are, do not change them because the default settings are the best settings.

> NOTE: Some settings, such as [actions](#), can only be enabled when you create a new `repository` using command-line (see next section). However, once you create the `repository` via command-line, you can use the `repository` as normal in Kopia GUI: just connect to the `repository` as described above after you have created it in command-line.

Once you do all that, your repository should be created and you can start backing up your data!

Kopia CLI

Creating a Repository

You must use the [`kopia repository create sftp` command](#) to create a `repository`:

```
$ kopia repository create sftp \
        --path=... \
        --host=... \
        --username=... \
        --sftp-password=...
```

OR

```
$ kopia repository create sftp \
        --path=... \
        --host=... \
        --username=... \
        --keyfile=...
```

If the connection to SFTP server does not work, try adding `--external` which will launch an external `ssh` process that supports more connectivity options and may be needed for some hosts.

At a minimum, you will need to enter the path, host, username, and either password or path to key file. You may also need to include `--known-hosts`. There are also various other options (such as actions) you can change or enable – see the help docs for more information.

You will be asked to enter the repository password that you want. This password can be whatever you want, it does not need to be the same as your SFTP password. In fact, it should not be the same! Remember, this password is used to encrypt your data, so make sure it is a secure password!

Connecting to Repository

After you have created the `repository`, you connect to it using the `kopia repository connect sftp` command. Read the help docs for more information on the options available for this command.

Rclone

Rclone is an open-source program that allows you to connect to various cloud storage platforms. Many of these platforms are already supported natively by Kopia (see above), but some are not. If you want to use Kopia to backup to cloud storage that Rclone supports but Kopia does not yet, then you can use Kopia's Rclone `repository` feature to do just that. The best part is that once you setup the Rclone `repository`, Kopia manages Rclone for you (including running Rclone when needed), so you do not need to do anything else after setup except make sure you enable Rclone's self-update feature so that it stays up-to-date.

> WARNING: Rclone support is experimental. In theory, all Rclone-supported storage providers should work with Kopia. However, in practice, only Dropbox, OneDrive, and Google Drive have been tested to work with Kopia through Rclone.

Before you can create an Rclone `repository` in Kopia, you first need to download/install Rclone and setup what is called an Rclone `remote` for the cloud storage you want to use. Do the following:

1. Download Rclone from the Rclone website; it is a single executable like Kopia, so you do not need to install it but do remember the path on your machine where you save the Rclone executable file because you will need to know it when setting up your `repository` in Kopia
2. Configure Rclone to setup a `remote` to the storage provider you want to use Kopia with; see Rclone help docs to understand how to do that

Kopia GUI

Select the `Rclone Remote` option in the `Repository` tab in `KopiaUI`. Then, follow on-screen instructions. You will need to enter `Rclone Remote Path` and `Rclone Executable Path`. The `Remote Path` is `my-remote:/some/path`, where you should replace `my-remote` with the name of the Rclone `remote` you created earlier and replace `/some/path` with the directory on the cloud storage where you want Kopia to save your snapshots. The `Executable Path` is the location on your machine where you saved the Rclone executable that you downloaded earlier.

You will next need to enter the repository password that you want. Remember, this password is used to encrypt your data, so make sure it is a secure password! At this same password screen, you have the option to change the `Encryption` algorithm, `Hash` algorithm, `Splitter` algorithm, `Repository Format`, `Username`, and `Hostname`. Click the `Show Advanced Options` button to access these settings. If you do not understand what these settings are, do not change them because the default settings are the best settings.

> NOTE: Some settings, such as actions and arguments to Rclone, can only be enabled when you create a new `repository` using command-line (see next section). However, once you create the `repository` via command-line, you can use the `repository` as normal in Kopia GUI: just connect to the `repository` as described above after you have created it in command-line.

Once you do all that, your repository should be created and you can start backing up your data! Remember, Kopia manages Rclone for you, so you do not need to do anything further with Rclone.

Kopia CLI

Creating a Repository

You must use the `kopia repository create rclone` command to create a `repository` (assuming `my-remote` is the name of the Rclone `remote` you created earlier and `/some/path` is the directory on the cloud storage where you want Kopia to save your snapshots):

```
$ kopia repository create rclone --rclone-exe=/path/to/rclone/executable --remote-path=my-remote:/some/path
```

There are also various other options (such as actions and arguments to Rclone) you can change or enable – see the help docs for more information.

You will be asked to enter the repository password that you want. Remember, this password is used to encrypt your data, so make sure it is a secure password!

Remember, Kopia manages Rclone for you, so you do not need to do anything further with Rclone once you have created the `repository`.

Connecting to Repository

After you have created the `repository`, you connect to it using the `kopia repository connect rclone` command. Read the help docs for more information on the options available for this command.

Local or Network-attached Storage

Kopia allows you to save your snapshots on your local machine, network-attached, or any other readable directory that is attached to your local machine (such as USB device, SMB directory, SSHFS mount, etc.). All of these storages fall under the `filesystem` label.

Creating a filesystem `repository` is done differently depending on if you use Kopia GUI or Kopia CLI.

Kopia GUI

Select the `Local Directory or NAS` option in the `Repository` tab in `KopiaUI`. Then, follow on-screen instructions. You will need to enter `Directory Path`.

You will next need to enter the repository password that you want. Remember, this password is used to encrypt your data, so make sure it is a secure password! At this same password screen, you have the option to change the `Encryption` algorithm, `Hash` algorithm, `Splitter` algorithm, `Repository Format`, `Username`, and `Hostname`. Click the `Show Advanced Options` button to access these settings. If you do not understand what these settings are, do not change them because the default settings are the best settings.

> NOTE: Some settings, such as actions, can only be enabled when you create a new `repository` using command-line (see next section). However, once you create the `repository` via command-line, you can use the `repository` as normal in Kopia GUI: just connect to the `repository` as described above after you have created it in command-line.

Once you do all that, your repository should be created and you can start backing up your data!

Kopia CLI

Creating a Repository

You must use the `kopia repository create filesystem` command to create a `repository`:

```
$ kopia repository create filesystem --path=...
```

There are also various other options (such as actions) you can change or enable – see the help docs for more information.

You will be asked to enter the repository password that you want. Remember, this password is used to encrypt your data, so make sure it is a secure password!

Connecting to Repository

After you have created the `repository`, you connect to it using the `kopia repository connect filesystem` command. Read the help docs for more information on the options available for this command.

Last modified October 23, 2024: list all required permissions for GCS (#4193) (9587d982)

Repository Server

By default, every user of Kopia repository directly connects to an underlying storage using read-write access. If the users who share the repository do not entirely trust each other, some malicious actors can delete repository data structures, causing data loss for others.

Repository Server allows an instance of Kopia to proxy access to the underlying storage and has Kopia clients proxy all access through it, only requiring a username and password to talk to the server without any knowledge of repository storage credentials.

In repository server mode, each user is limited to seeing their own snapshots and policy manifest without being able to access those from another user account.

> NOTE: Only snapshot and policy manifests are access-controlled, not the underlying contents. If two users share the same file, it will be backed using identical content IDs. The consequence is that if a third user can guess the content ID of files in the repository, they can access the files. Because content IDs are one-way salted hashes of contents, it should be impossible to guess content ID without possessing original content.

Starting Repository Server

Before starting the repository server, we must first create and configure a repository. Finally, we must create a list of usernames and passwords that will be allowed to access it. The repository server should be started in a location where:

- all kopia clients can connect directly to the server;
- the latency between the client and the server is low;
- theres is sufficient bandwidth between the client and the server.

Configuring Allowed Users

Starting in Kopia v0.8, allowed repository users can be configured using `kopia server user` commands. Each user is identified by its lowercase `username@hostname` where hostname by default is the name of the computer the client is connecting from (without domain name suffix).

To add a user:

```
$ kopia server user add myuser@mylaptop
Enter new password for user myuser@mylaptop:
Re-enter new password for verification:

Updated user credentials will take effect in 5-10 minutes or when the server is restarted.
To refresh credentials in a running server use 'kopia server refresh' command.
```

Other commands are also available:

- `kopia server user list` - lists user accounts
- `kopia server user set` - changes password
- `kopia server user delete` - deletes user account

Auto-Generated TLS Certificate

To start repository server with auto-generated TLS certificate for the first time:

```
KOPIA_PASSWORD="<password-for-the-repository>" \
KOPIA_SERVER_CONTROL_PASSWORD="<server-control-password>" \
  kopia server start \
    --tls-generate-cert \
    --tls-cert-file ~/my.cert \
    --tls-key-file ~/my.key \
    --address 0.0.0.0:51515 \
    --server-control-username control
```

This will generate TLS certificate and key files and store them in the provided paths (`~/my.cert` and `~/my.key` respectively). It will also print certificate SHA256 fingerprint, which will be used later:

```
SERVER CERT SHA256: 48537cce585fed39fb26c639eb8ef38143592ba4b4e7677a84a31916398d40f7
```

Note that when starting the server again the `--tls-generate-cert` must be omitted, otherwise the server will fail to start.

Custom TLS Certificates

If a user has obtained a custom certificate (for example, from LetsEncrypt or another CA), using it is simply a matter of providing a PEM-formatted certificate and key files on server startup.

To get the SHA256 digest of an existing certificate file, use:

```
$ openssl x509 -in ~/my.cert -noout -fingerprint -sha256 | sed 's/://g' | cut -f 2 -d =
48537CCE585FED39FB26C639EB8EF38143592BA4B4E7677A84A31916398D40F7
```

On Client Computer

Assuming we're on another machine running as `user1@host1`, we can now run the following command to connect to the repository (notice we're using fingerprint obtained before without `:` separators)

```
kopia repository connect server --url https://<address>:51515 \
  --server-cert-fingerprint 48537cce585fed39fb26c639eb8ef38143592ba4b4e7677a84a31916398d40f7
```

Once connected, all snapshot and policy `kopia` commands should work for the current user, but low-level commands such as `repo status` will fail:

```
$ kopia repository status
kopia: error: operation supported only on direct repository, try --help
```

You can override username and hostname with this command :

```
$ kopia repo connect server --url=http://11.222.111.222:51515 --override-username=johndoe --override-hostname=my-laptop
```

Server Access Control (ACL)

Kopia server will check permissions when users try to access contents and manifests based on rules we call ACLs (access control list).

Starting in Kopia v0.8, the ACLs can be controlled by using `kopia server acl` commands.

If no ACLs are explicitly defined, Kopia will use a set of built-in access control rules, which grants all authenticated users identified by `username@hostname` ability to:

- read and write policies for `username@hostname` and `username@hostname:/path`,
- read and write snapshots for `username@hostname:/path`,
- read `global` policy,
- read `host`-level policies for their own `hostname`,
- read and write `user`-level policies for their own `username@hostname`,
- read and write their own `user` account `username@hostname` (to be able to change password),
- read objects if they know their object IDs
- write new `content` to the repository

Access control for individual files or directories

Kopia does not currently perform access control checks to verify that a user trying to access a file or directory by object ID is the original owner of the file (because of Kopia's deduplication, two different users who have the same file will get the same object ID when snapshotting it).

This means that any user who knows of a valid object ID will be able to restore its contents (by `kopia restore <object-id>` or `kopia show <object-id>`, etc.).

Users who currently are (or previously were) in possession of a file can easily determine its object ID from one of the snapshot manifests. However, it is unlikely to guess 128-bit or 256-bit object identifiers for other users.

On the flip side, this allows easy sharing of files between users simply by exchanging object IDs and letting another user restore the object (either a single file or an entire directory) from the repository.

Customizing ACL rules

Sometimes, we want to be able to customize those rules, for example, to allow some users to modify `global` or `host`-level policies, to let one user see another user's snapshots.

To enable ACL mode, run:

```
$ kopia server acl enable
```

This will install ACL entries matching the default rules. Let's take a look at the rules installed:

```
$ kopia server acl list
id:c95e5a13b0b0fc874e550677f5e89262 user:*@* access:APPEND target:type=content
id:633f998af93e00c715e6417cb41df0af user:*@* access:READ target:type=policy,policyType=global
id:de21ac7a55ec1d9cf939b4cb158cc85a user:*@* access:READ target:type=policy,hostname=OWN_HOST,policyType=host
id:2314795d54060ab84cdc7fcb6e6953c8 user:*@* access:FULL target:type=policy,username=OWN_USER,hostname=OWN_HOST
id:0245e5bc1eca296e775e137a9587d09e user:*@* access:FULL target:type=snapshot,hostname=OWN_HOST,username=OWN_USER
id:fb043d9e77299e182e9297d0f4601d9b user:*@* access:FULL target:type=user,username=OWN_USER@OWN_HOST
```

As you can see, all rules have unique identifiers (different for each repository) and each rule defines:

- The `user` the rule applies to (could be a wildcard pattern containing an asterisk (*) instead of `username` or `hostname`)

- The `access` level:

  - `READ` - allows reading but not writing
  - `APPEND` - allows reading and writing, but not deleting
  - `FULL` - allows full read/write/delete access

- The `target`, which specifies the manifests the rule applies to.

  The target specification consists of `key=value` pairs, which must match the corresponding manifest labels. Each target must have a `type` label and (optionally) other labels that are type-specific.

Supported types are:

- `snapshot` with optional labels `username`, `hostname` and `path`
- `policy` with optional labels `username`, `hostname`, `path` and `policyType` (which must be one of `global`, `user`, `host` or `path`)
- `user` with optional label `username`
- `acl`

Only labels specified will be matched. The label values can be literals or one of two special values:

- `OWN_USER` - the user's own `username`
- `OWN_HOST` - the user's own `hostname`

Defining ACL rules

To define an ACL rule use:

```
$ kopia server acl add --user U --access A --target T
```

For example:

1. To grant user `alice@wonderland` the ability to modify global policy:

```
$ kopia server acl add --user alice@wonderland \
      --access FULL --target type=policy,policyType=global
```

2. To allow all users to see all snapshots in the system:

```
$ kopia server acl add --user "*@*" --access READ --target type=snapshot
```

3. To give `princess@tall-tower` the ability to modify that hosts' policy:

```
$ kopia server acl add --user "princess@tall-tower" --access FULL \
    --target type=policy,policyType=host,hostname=tall-tower
```

4. To give `admin@somehost` the ability to add new user accounts and change passwords for existing users:

```
$ kopia server acl add --user "admin@somehost" --access FULL --target type=user
```

5. To give `superadmin@somehost` the ability to define new ACLs:

```
$ kopia server acl add --user "superadmin@somehost" \
    --access FULL --target type=acl
```

Deleting ACL rules

To delete a single ACL rule, use `kopia server acl remove` passing the identifier of the entry:

```
$ kopia server acl remove 77fb0a5706ddbd9848294b36aae47079
```

To delete all ACL rules and revert to default set of rules, use:

```
$ kopia server acl remove --all
```

Both commands default to preview mode and must be confirmed by passing `--delete` for safety.

Reloading server configuration

Kopia server will refresh its configuration by fetching it from repository periodically. To speed up this process after changing access control rules, adding or modifying users or to simply force server to discover new snapshots or policies, you may want to run:

```
$ kopia server refresh \
  --address=https://server:port [--server-cert-fingerprint=FINGERPRINT] \
  --server-username=control \
  --server-password=PASSWORD_HERE
```

To authenticate, pass any valid username and password configured on the server.

Alternatively on Linux/macOS you can send `SIGHUP` signal to a running Kopia process to trigger a refresh.

```
$ ps awx | grep kopia
74189 s003  U+     0:19.19 kopia server start ...

$ kill -SIGHUP 74189
```

or simply:

```
$ killall -SIGHUP kopia
```

Kopia behind a reverse proxy

Kopia server can be run behind a reverse proxy. Here a working example for nginx.

```
server {
  listen 443 ssl http2;
  server_name mydomain.com;

  ssl_certificate_key /path/to/your/key.key;
  ssl_certificate /path/to/your/cert.crt;

  client_max_body_size 0;  # Allow unlimited upload size

  location / {
   grpc_pass grpcs://localhost:51515; # Adapt if your kopia is running on another server
  }
}
```

Make sure you use a recent nginx version (>=1.16) and you start your kopia server with a certificate (`--insecure` does not work, as GRPC needs TLS, which is used by Repository Server), e.g.

```
kopia server start --address 0.0.0.0:51515 --tls-cert-file ~/my.cert --tls-key-file ~/my.key
```

You can now connect to your kopia server via reverse proxy with your domain: `mydomain.com:443`.

Alternatively here is an example nginx/kopia configuration using unix domain sockets:

```

upstream socket {
    server unix:///tmp/kopia.sock;
}
server {
  listen 443 ssl http2;
  server_name mydomain.com;

  ssl_certificate_key /path/to/your/key.key;
  ssl_certificate /path/to/your/cert.crt;

  client_max_body_size 0;  # Allow unlimited upload size

  location / {
   grpc_pass grpcs://socket; # Adapt if your kopia is running on another server
  }
}
```

```
kopia server start --address unix:/tmp/kopia.sock --tls-cert-file ~/my.cert --tls-key-file ~/my.key
```

Kopia with systemd

Kopia can be run as a socket-activated systemd service. While socket activation is not typically needed for Kopia, it can be helpful to run it in a rootless Podman container or to control the permissions of the unix-domain-socket when running behind a reverse proxy.

Kopia will detect socket activation when present and ignore the –address switch.

When using socket activation with Kopia server, it is generally desirable to enable both the socket and the service so that the service starts immediately instead of on-demand (so that the maintenance can run).

An example kopia.socket file using unix domain sockets and permission control may look like:

```
[Unit]
Description=Kopia

[Socket]
ListenStream=%t/kopia/kopia.sock
SocketMode=0666

[Install]
WantedBy=sockets.target
```

Kopia v0.8 usage notes

Configuring Allowed Users

Prior to Kopia v0.8, the user list must be put in a text file formatted using the htpasswd utility from Apache. This method is still supported in v0.8, but it’s recommended to use `kopia server user` to manage users instead. To create password file for two users:

```
$ htpasswd -c password.txt user1@host1
New password:
Re-type new password:
Adding password for user user1@host1

$ htpasswd password.txt user2@host1
New password:
Re-type new password:
Adding password for user user2@host1
```

Auto-Generated TLS Certificate

Prior to Kopia v0.8, the command line for `kopia server start` also needs `--htpasswd-file ~/password.txt`

Server Access Control (ACL)

Prior to Kopia v0.8, the rules were non-configurable and each user could only read and write their own snapshot manifests.

Last modified May 24, 2024: docs(server): add repo password in server start example (#3875) (1aab37dd)

Architecture

Architecture

Kopia stores its data in a data structure called Repository.

Repository adapts simple storage such as NAS filesystem, Google Cloud Storage or Amazon S3 to add features such as encryption, deduplication, content-addressability and ability to maintain rich snapshot history.

The following diagram illustrates the key components of Kopia:

Binary Large Object Storage (BLOB)

BLOB storage is the place where your data is ultimately stored. Any type that implements simple Go API can be used as Kopia's blob storage.

See the Repositories page for a list of currently supported storage backends.

Cloud storage solutions (such as GCS, S3 or Azure Blob Storage) are great choices because they provide high availability and data durability at reasonable prices.

Kopia does not require low-latency storage, it uses caching and other optimizations to be able to work efficiently with high-latency backends.

The API for BLOB storage can be found in https://godoc.org/github.com/kopia/kopia/repo/blob

Content-Addressable Block Storage (CABS)

BLOB storage by itself does not provide the features that Kopia needs (encryption, de-duplication), so we're using content-addressable block storage layer to add those key features.

Content-Addressable Block Storage manages data blocks of relatively small sizes (typically <=20MB). Unlike BLOB storage where we can freely pick a filename, CABS assigns an identifier called **Block ID** to each block of data that gets stored.

Block ID is generated by applying cryptographic hash function such as SHA2 or BLAKE2S to produce a pseudo-random identifier, such as `6a9fc3a464a79360269e20b88cef629a`.

A key property of block identifiers is that two identical block of data will produce exactly the same identifiers, thus resulting in natural de-duplication of data.

After hashing, the block data is encrypted using algorithm such as `AES256-GCM-HMAC-SHA256` or `CHACHA20-POLY1305-HMAC-SHA256`. To make uploads to cloud storage more efficient and cheaper, multiple smaller blocks are combined into larger **Packs** of 20-40MB each. Pack files in blob storage have random names and don't reveal anything about their contents or structure. Their sizes are also generally unrelated to content due to splitting and merging.

To help efficiently find a block in the blob storage, CABS maintains an index that maps block ID to the blob file name, offset within the file and length. In addition, to make data recovery possible in case the index files got corrupted, a local index is also stored at the end of each pack.

File names in the BLOB layer have prefixes to help quickly identify its type:

- `p` represents packs containing data (e.g. `pb4cf8ca179d71478fb8d4b00b79a9a72`)
- `q` represents packs containing metadata (e.g. `q7a9939814e8aba1fdda2d87965f324d3`)
- `x` represents indices (e.g. `xn0_20db7984bd71c4042cea471a61fbcea1`)

CABS is not meant to be used directly, instead it's a building block for object storage (CAOS) and manifest storage layers (LAMS) described below.

The API for CABS can be found in https://godoc.org/github.com/kopia/kopia/repo/content

Content-Addressable Object Storage (CAOS)

Content-Addressable Object Storage allows storing binary objects of arbitrary sizes. Small objects are stored directly as individual CABS blocks, but larger objects need to be split into many smaller blocks before they can be stored.

Object IDs represent CAOS objects and are similar to Block IDs in that they are derived from object data. In fact for small blocks they are both identical: every valid Block ID is also valid Object ID representing the same contents.

Object IDs can also have an optional single-letter prefix `g..z` that helps quickly identify its type:

- `k` represents directory listing (e.g. `kfed1b0498dc54d07cd69f272fb347ca3`)
- `m` represents manifest block (e.g. `m0bf4da00801bd8c6ecfb66cffa67f32c`)
- `x` represents indirect JSON content (e.g. `xac47f7ce280fdd81f04c670fec2353dc`)

Objects without prefixes are stored in the `p` pack while objects with prefixes are stored in the `q` metadata pack by the CABS layer.

To represent objects larger than the size of a single CABS block, Kopia links together multiple blocks via special indirect JSON content. Such blocks are distinguished from regular blocks the `I` virtual prefix. Indirection can be applied to any object type, including metadata such as directory listing. However, when applied to a data block, an `x` prefix is added to ensure than the indirect JSON is stored in a metadata pack instead of a data pack by the CABS layer.

For example large file object might have an identifier such as `Ixac47f7ce280fdd81f04c670fec2353dc` and JSON content:

```
{"stream":"kopia:indirect","entries":[
  {"l":2617867,"o":"e510796ba6ffd15649ea400b67ef6159"},
  {"s":2617867,"l":2751278,"o":"5c090744e0cad69d0d1aecd4ffd69f69"},
  {"s":5369145,"l":2476164,"o":"1f780f1a968fcdaa9e3a51961ce1cd8a"},
  {"s":7845309,"l":2640451,"o":"c03399d1d2fb13c48c062d1243deef91"}
]}
```

The content list command can be used to list CABS blocks in the repository:

```
$ kopia content list
```

The content show command can be used to show the content of a specific CABS block:

```
# Show the content of a directory listing
$ kopia content show kfed1b0498dc54d07cd69f272fb347ca3
# Show the indirect JSON content of a large file object
$ kopia content show xac47f7ce280fdd81f04c670fec2353dc
# Show the actual content of a large file object
$ kopia content show Ixac47f7ce280fdd81f04c670fec2353dc
```

Note that the example above shows that `I` is a virtual prefix. The actual CABS block does not content the `I` prefix but referencing the block with the `I` prefix tells Kopia to interpret the block as an indirect JSON object and return the resolved content instead of the raw content.

The API for CAOS can be found in https://godoc.org/github.com/kopia/kopia/repo/object

Label-Addressable Manifest Storage (LAMS)

While content-addressable storage is a neat idea, dealing with cryptographic hashes is not very convenient for humans to use.

To address that, Kopia supports another type of storage, used to persist small JSON objects called **Manifests** (describing snapshots, policies, etc.) which are identified by arbitrary `key=value` pairs called labels.

Internally manifests are stored as CABS blocks.

The API for LAMS can be found in https://godoc.org/github.com/kopia/kopia/repo/manifest

Last modified March 21, 2023: docs(site): unified headings - created an outline within the advanced section (#2838) (8459e1c0)

Using Different Storage Classes

Storage Classes

Most of the cloud storages supported by Kopia have a feature called storage classes (or storage tiers). Storage classes allow you to trade-off costs for storage and access and, in some cases, data redundancy. Exactly what the trade-offs are vary between cloud providers. However,

- 'hot' storage is typically the most expensive to store, the least expensive to access, and has the highest redundancy;
- 'cold' storage is less expensive to store, more expensive to access, and sometimes has less redundancy than hot storage;
- and 'archive' storage is the least expensive to store, the most expensive to access, often has less redundancy than both hot and cold storage, and most of the time provides access to your files with a delay of several hours (i.e., you cannot instantly download your files).

Some storage classes also have a minimum data retention policy, meaning that they charge you the full price for their whole retention period (e.g. 90 days) even if you delete files before the retention period has ended. This retention policy is typically with cold and archive storage, but some hot storage also have such policies. For example, Wasabi has a 90-day retention policy (at the time of this writing).

The most famous example of archive storage is Amazon Glacier (now called Amazon Glacier Deep Archive).

Kopia works without issue with all storage classes that provide **instant access** to your files, namely hot or cold storage. Kopia will **not** work with archive storage classes that provide *delayed* access to your files, such as Amazon Glacier Deep Archive.

> PRO TIP: If you are not downloading or testing your snapshots regularly, you may save money by using Kopia with some sort of cold storage class; just make sure whatever storage class you use, that storage class provides instant access to your files and not delayed access.

By default, Kopia stores all snapshots using whatever is the standard storage class for your cloud provider; often, that is hot storage. If you want to change storage classes, one way is to create 'lifecycle' rules from within your cloud provider account. This lifecycle rules feature allows you to automatically move snapshots to different storage classes directly from within your bucket. This approach is independent from Kopia; Kopia does not manage lifecycle rules for you, you have to do it from within your cloud provider account.

Another approach that some providers allow is to set a default storage class for a bucket that is other than their standard. This approach, too, is independent from and is not managed by Kopia: you control this from within your cloud provider account. If you follow this approach, Kopia will automatically use the default storage class that you set for your bucket.

Alternatively, Kopia supports the ability to pick a storage class when uploading snapshots to your cloud provider, so that you do not need to create any lifecycle rules or change the default storage for your bucket. However, this feature is only available for the Amazon S3 and S3-compatible cloud storage that have implemented the S3 storage class API (some S3-compatible cloud storage have implemented this API and some have not; you will need to research your cloud provider to find out). To use this feature, you must upload a `.storageconfig` file to the bucket in your cloud provider account that you use to store Kopia snapshots. This feature tells Kopia what storage class to use when uploading your snapshots.

> PRO TIP: `.storageconfig` works for all uploads after you have created the file; it will not change the storage class of files uploaded before you create `.storageconfig`. Thus, if you want to use `.storageconfig` from all data stored in a repository, make sure to upload `.storageconfig` before you create the repository in Kopia.

An example `.storageconfig` looks like this:

```
{
  "blobOptions": [
    { "prefix": "p", "storageClass": "STANDARD_IA" },
    { "storageClass": "STANDARD" }
  ]
}
```

In this file, you need to create an object in the `blobOptions` array for each blob type whose storage class you want to change. In the above example, the `.storageconfig` file is telling Kopia to upload `p` blobs using the `STANDARD_IA` storage class and all other blobs as the `STANDARD` storage class. You can add as many objects in the `blobOptions` array as you desire.

> PRO TIP: You can read about all of Amazon S3's storage classes on Amazon's website. For all S3-compatible storage, you will need to research what storage classes are supported by that provider and what the storage classes are called. Once you know the names of the storage classes, you can use `.storageconfig` as described in this document.

The following are all the blob types that are used by Kopia; you can change the storage class for each of these blob types by creating an object for the respective `prefix` in a `.storageconfig` file:

- `p` blobs store the bulk of snapshot data; they are mostly written once and not accessed again, except during compactions as part of full maintenance or when (optionally) testing your snapshots
- `q` blobs store the metadata for your snapshots (directory listings, manifests, etc.); they are frequently read and written

- `n`, `m`, `l` and `x` blobs are for indexes used in your snapshots; they are frequently read, written, and deleted
- `s` blobs are for sessions; they are frequently read, written, and deleted

You may save money (depending on the cloud storage you use and their pricing) by putting `p` blobs in cold storage because those blobs are infrequently accessed. You likely will not be saving money by putting all other blobs in cold storage, since those blobs are frequently accessed.

> PRO TIP: You can see when files are accessed by Kopia by viewing Kopia's debug logs (`kopia ... --log-level=debug`); look for lines containing `STORAGE`. Note that `q` blobs are very aggressively cached locally by Kopia, so they may appear to not be accessed when performing basic operations like listing snapshots.

Using Archive Storage With Delayed Access

Kopia does **not** currently support cloud storage that provides *delayed* access to your files – namely, archive storage such as Amazon Glacier Deep Archive. Do not try it; things will break.

If you really want to use archive storage with Kopia, consider using Google Cloud Storage's Archive storage class – it is more expensive than Amazon Glacier Deep Archive but still very cheap to store ($0.0012 per GB at the time of this writing) and provides instant access to your files; but remember that, like other archive storage, costs are high for accessing files in Google Cloud Storage's Archive storage class.

Last modified March 21, 2023: docs(site): unified headings - created an outline within the advanced section (#2838) (8459e1c0)

Compression

Compression

Kopia offers internal content compression feature. Each data block stored may be compressed with the algorithm of user choice.

Here is how the whole process works in a summary (with the architecture in your belly first). When Kopia sees a file to backup, it first splits its data into "chunks" with the configured splitter. Then compute its hash with the configured hash algorithm, and compare it against the existing database. If a matching hash is found, this chunk is already stored, thus can be safely discarded. Otherwise, Kopia applies the compression algorithm on the chunk, encrypt it, and pack a few of such into a blob if fit.

Kopia allows user to set compression algorithm, minimum and maximum file size and extensions to compress, on per-policy basis. Below we will discuss how these settings would interact with the data to be backed up.

To see how much data is saved by compression in a repository, use `kopia content stats` command. To see the size before and after compression for each content object, use `kopia content list --json` command then look for ID.

Algorithm

Some algorithms come with different presets. Others are fixed as the code was originally designed. For example, the s2 compression presents 3 variants: default, better and parallel-n. "default" balances between compression ratio and performance, while "better" offers higher compression ratio at the cost of performance. Both of the two have the concurrency equal to the number of logical cores. "parallel-n" is basically "default" with fixed amount of concurrency.

Other algorithms's variants are straightforward. Anything mentioning "speed" or "fast" favor performance over efficiency. Anything mentioning "compression" favors the opposite.

There is no hard rule as which algorithm and which variant is the best. It's all depend on the use case. Fortunately, Kopia comes with a handy benchmark tool to differentiate the choices. `kopia benchmark compression --data-file=<target_file>` is the command to test all supported variants on the target file.

Below is a sample output, generated from a highly compressible 466MB file with AMD Ryzen 7 CPU:

```
Repeating 1 times per compression method (total 466.7 MiB).

     Compression                Compressed   Throughput   Memory Usage
------------------------------------------------------------------------------------------------
  0. s2-default                 127.1 MiB    4 GiB/s      3126   375.4 MiB
  1. s2-better                  120.1 MiB    3.4 GiB/s    2999   351.7 MiB
  2. s2-parallel-8              127.1 MiB    2.8 GiB/s    2981   362.2 MiB
  3. s2-parallel-4              127.1 MiB    2.3 GiB/s    2951   344.1 MiB
  4. pgzip-best-speed           96.7 MiB     2.1 GiB/s    4127   324.1 MiB
  5. pgzip                      86.3 MiB     1.2 GiB/s    4132   298.7 MiB
  6. lz4                        131.8 MiB    458.9 MiB/s  17     321.7 MiB
  7. zstd-fastest               79.8 MiB     356.2 MiB/s  22503  246 MiB
  8. zstd                       76.8 MiB     323.7 MiB/s  22605  237.8 MiB
  9. deflate-best-speed         96.7 MiB     220.8 MiB/s  45     310.8 MiB
 10. gzip-best-speed            94.9 MiB     165 MiB/s    40     305.2 MiB
 11. deflate-default            86.3 MiB     143.1 MiB/s  34     311 MiB
 12. zstd-better-compression    74.2 MiB     104 MiB/s    22496  251.4 MiB
 13. pgzip-best-compression     83 MiB       55.9 MiB/s   4359   299.1 MiB
 14. gzip                       83.6 MiB     40.5 MiB/s   69     304.8 MiB
 15. zstd-best-compression      68.9 MiB     19.2 MiB/s   22669  303.4 MiB
 16. deflate-best-compression   83 MiB       5.6 MiB/s    134    311 MiB
 17. gzip-best-compression      83 MiB       5.1 MiB/s    137    304.8 MiB
```

As you can see, s2 compression clearly favors performance over compression ratio. zstd on the other hand results almost half the size as s2. pgzip arguably offers the best balance on two worlds.

As soon as the throughput of compression is higher than I/O, compression is no longer the bottleneck. Therefore, any higher compression basically comes as free. Take the file above as example, if the I/O throughput is limited to 1 GiB/s, pgzip is a better choice than s2, because using s2 would put the CPU into lots of idle time, while losing compression. However, if the I/O throughput limit is 2.5 GiB/s, it would become a hard choice between the two.

Another kind of metrics shown by the benchmark tool is the memory usage. The first column shows the number of memory allocation, while the second shows the total amount of consumption during the process. In the example above, all algorithms show similar level of memory usage. However, things are slightly different if we change the argument to the benchmark command.

Here is the output if we compress a smaller file repeatedly:

```
Repeating 100 times per compression method (total 12.5 MiB).
```

```
     Compression                   Compressed   Throughput   Memory Usage
------------------------------------------------------------------------------------------------
  0. s2-parallel-4                 43.6 KiB     833.4 MiB/s  533    2.1 MiB
  1. s2-parallel-8                 43.6 KiB     833.3 MiB/s  555    2.1 MiB
  2. s2-default                    43.6 KiB     833.3 MiB/s  579    2.1 MiB
  3. s2-better                     41.3 KiB     500 MiB/s    610    2.1 MiB
  4. zstd-fastest                  28.6 KiB     240.4 MiB/s  925    9.5 MiB
  5. deflate-best-speed            34.3 KiB     198.4 MiB/s  22     874.6 KiB
  6. zstd                          26.8 KiB     165.4 MiB/s  907    18.5 MiB
  7. zstd-better-compression       26.3 KiB     162.3 MiB/s  881    37.3 MiB
  8. gzip-best-speed               33.7 KiB     150.6 MiB/s  28     1.2 MiB
  9. pgzip-best-speed              34.3 KiB     143.7 MiB/s  1649   220.2 MiB
 10. deflate-default               31.2 KiB     126.3 MiB/s  22     1.1 MiB
 11. lz4                           44.7 KiB     112.6 MiB/s  435    816.7 MiB
 12. pgzip                         31.2 KiB     94.6 MiB/s   2634   277.5 MiB
 13. gzip                          30.4 KiB     39.5 MiB/s   26     874.7 KiB
 14. deflate-best-compression      30.4 KiB     25.4 MiB/s   21     1 MiB
 15. gzip-best-compression         30.4 KiB     24.5 MiB/s   27     874.9 KiB
 16. pgzip-best-compression        30.4 KiB     23.1 MiB/s   2646   281.8 MiB
 17. zstd-best-compression         25.1 KiB     19.3 MiB/s   882    99.3 MiB
```

While s2 significantly uses way less memory in this case, pgzip's numbers seem indifferent to the input size. Turns out, pgzip has different memory usage logic. It would quickly allocate necessary memory and plateau, which s2 is more on a linear fashion. Here is rough graph is demonstrate the difference:

Therefore, if your backup target is small, and memory is extremely restricted, s2 might be necessary. Otherwise, all algorithms are valid candidates.

Minimum file size and extensions to compress

As discussed above, some compression algorithms make sense only if the payload is large enough. So it might be beneficial to set a minimum file size when using these algorithms.

On the other hand, if Kopia notices that a data chunk grows size after compression, it will simply use the original data (which explains why you might still see significant "(uncompressed)" in the `kopia content stats` output). So if you don't know the optimal minimum file size for the algorithm with your data set, and you don't mind wasting some CPU time to compress the data then get discarded, you don't need to set a minimum file size.

As for extensions, some file formats are already heavily compressed, such as video files. Applying general-purposed compression would not have much effect, while wasting CPU time. These file extensions are suggested to be set to never compressed.

Side note

We also compared the efficiency of compressing a file as whole using standalone tools versus Kopia (that is, split with default `DYNAMIC-4M-BUZHASH` then compress). Here is the result

Description	Size in MiB	Ratio to original
Original	466.7	100%
s2c, which compress file with s2	119.4	25.59%
Kopia s2-default	133.3	28.56%
7zip using gzip format	80.6	17.27%
Kopia gzip	87.8	18.81%

The loss of compression ratio due to splitting is expected, and we are happy to see the loss is relatively small. Don't forget that in a large enough repository, the deduplication itself (thanks to splitting) would further reduce the backup size.

Last modified March 21, 2023: docs(site): unified headings - created an outline within the advanced section (#2838) (8459e1c0)

Encryption

Encryption

Kopia uses a standard envelope encryption technique to de-couple the repository passphrase from the keys used for encrypting and authenticating the contents of the repository.

Each repository has a format blob containing configuration parameters for the repository. The format blob itself is serialized and persistent as JSON, such as the one below.

```
{
  "tool": "https://github.com/kopia/kopia",
  "buildVersion": "v0.3.0",
  "buildInfo": "unknown",
  "uniqueID": "bm90IGEgZ29vZCByYW5kb20gaW5pdGlhbCB2YWx1ZQ==",
  "keyAlgo": "scrypt-65536-8-1",
  "version": "1",
  "encryption": "AES256_GCM",
  "encryptedBlockFormat": "ZSxqt/gXwVPS6pNvFZOOHlCKr18TmziuUnN8nnuvwQ/+mjbcvEHUfKS11RJl/sWrIOyiYqpSwAZt
BzOxQAUWQt7vHc2ZT4Y75ODrPHhzd0CcBlqHa3HSx8pxIXqpgMy5K3xDvIkBN1qdb/cTyU5s9lZ2
J+rBh2CGQ3phIlKFCCuS3lgB+rnYRrExGg5rm4BUmZZWHfBPQ7QiOJNg86PK+n///dejsAA/+FBj
qIODf7Gr3ppaGZeAHYJuz0BkRSPC4XIAgFj1yPo="
}
```

The corresponding Go struct is the following:

```go
type formatBlob struct {
 Tool         string `json:"tool"`
 BuildVersion string `json:"buildVersion"`
 BuildInfo    string `json:"buildInfo"`

 UniqueID               []byte `json:"uniqueID"`
 KeyDerivationAlgorithm string `json:"keyAlgo"`

 Version               string                  `json:"version"`
 EncryptionAlgorithm   string                  `json:"encryption"`
 EncryptedFormatBytes  []byte                  `json:"encryptedBlockFormat,omitempty"`
 UnencryptedFormat     *format.RepositoryObjectFormat `json:"blockFormat,omitempty"`
}
```

- `Version` identifies the version of the format blob.
- The `tool` and `buildInfo` fields are informational
- `buildVersion` is the version of Kopia, which is also indirectly used as the repository format version.
- `UniqueID` is a randomly generated identifier for the repository. This is also used as the input for various encryption operations.
- `keyAlgo` identifies the password-based key derivation function (PBKDF). Only *scrypt* with the currently recommended cost parameters (N=65536, r=8, p=1) is supported at the moment. The main purpose of this field is to be able to change and extend the key derivation algorithm in the future. For example, by increasing the cost parameters for *scrypt* or using a different algorithm altogether.
- `encryption` identifies the encryption algorithm that was used to encrypt the encryptedBlockFormat field.
- `encryptedBlockFormat` is a ciphertext containing among others, the encryption secrets and parameters used for encrypting the repository content. Below is additional information about its plaintext content and how it is encrypted.
- Alternatively, the unencrypted block format parameters can be specified in the the `blockFormat` field.

The `formatBlob.EncryptedBlockFormat` field is the result of encrypting a JSON-serialized version of the `EncryptedRepositoryConfig` struct shown below. The plaintext version contains the parameters for performing block chunking, as well as for encrypting and authenticating "content" objects.

```go
type EncryptedRepositoryConfig struct {
 Format RepositoryObjectFormat `json:"format"`
}

type RepositoryObjectFormat struct {
 format.ContentFormat
 format.ObjectFormat
}
```

```go
package content

// ContentFormat describes the rules for formatting contents in repository.
type ContentFormat struct {
 Version     int    `json:"version,omitempty"`     // version number, must be "1"
 Hash        string `json:"hash,omitempty"`        // identifier of the hash algorithm used
 Encryption  string `json:"encryption,omitempty"`  // identifier of the encryption algorithm used
 HMACSecret  []byte `json:"secret,omitempty"`      // HMAC secret used to generate encryption keys
 MasterKey   []byte `json:"masterKey,omitempty"`   // master encryption key (SIV-mode encryption only)
 MaxPackSize int    `json:"maxPackSize,omitempty"` // maximum size of a pack object
}
```

```
package object

type Format struct {
 Splitter string `json:"splitter,omitempty"` // splitter used to break objects into pieces of content
}
```

The ciphertext in `formatBlob.encryptedBlockFormat` is obtained by:

1. Serializing to JSON the populated encryptedRepositoryConfig struct.
2. Encrypting the resulting byte stream with the algorithm specified in `formatBlob.encryption`. At the moment, `AES256_GCM` is used by default, which is both encrypted and authenticated. The input parameters are:
    - "Plaintext": serialized JSON byte stream
    - "IV": randomly generated and prepended to the ciphertext in `formatBlob.encryptedBlockFormat`
    - Key: 256-bit AES encryption key derived from the passphrase, details below.
    - Additional Data (AD): Used as "authentication data", the value is derived from the passphrase as well. See below.

The passphrase is used to generate the encryption key and *additional data* (AD) for the encryption and decryption of `formatBlob.encryptedBlockFormat` ciphertext as follows:

- A master key (Km) is derived from the password by using (a) the password-based key derivation function specified in `formatBlob.keyAlgo`, and (b) `formatBlob.UniqueID` as the salt. The resulting key is 32-bytes long (256 bits). `Km = PBKDF( passphrase, formatBlob.UniqueID, … cost parameters).`
- The AES-256 encryption key (Ke) is derived from Km by using a hash-based key derivation function (HKDF), with SHA256 as the hash. `Ke = HKDF(SHA256, Km, formatBlob.UniqueID, "AES", 32)`
- The additional data (AD) is derived using an HKDF as follows: `AD = HKDF(SHA256, Km, formatBlob.UniqueID, "CHECKSUM", 32)`

Last modified March 21, 2023: docs(site): unified headings - created an outline within the advanced section (#2838) (8459e1c0)

Error Correction Algorithm

Error Correction Algorithm

Starting with v0.12.0, Kopia supports the use of error correction using the Reed-Solomon error correction algorithm. Error correction in Kopia is used to mitigate the likelihood that your snapshots become corrupt due to storage errors caused by the hardware your snapshots are saved on (like bitflips). Most, if not all, cloud storage platforms use their own error correction, so using Kopia's error correction for cloud repositories may be overkill. However, the choice is yours.

If you use error correction in Kopia, the `Error Correction Overhead` option controls how much extra storage space is needed for the error correction code. So, for example, if you pick the `1%` option, then your repository will use 1% more storage space with error correction enabled compared to without error correction.

Currently, if you want to use Reed-Solomon error correction with Kopia, you must create a new repository and enable the option when you create the new repository, because there is not yet a way to enable the feature for an existing repository.

This feature is currently experimental. Use at your own risk. You can read more about how the Reed-Solomon algorithm works at Wikipedia.

Last modified March 21, 2023: docs(site): unified headings - created an outline within the advanced section (#2838) (8459e1c0)

Maintenance

Maintenance

Kopia repositories require periodic maintenance to ensure best possible performance and optimal storage usage.

Starting with v0.6.0 the repository maintenance is automatic and will happen occasionally when `kopia` command-line client is used. This document describes maintenance functionality in greater detail.

Maintenance Tasks

Kopia uses the following types of maintenance tasks:

- **Quick Maintenance Tasks** are primarily responsible for keeping the number of frequently accessed blobs (`q` and `n`) low to ensure good performance.

  Quick Maintenance will never delete any metadata from the repository without ensuring that another copy of the same metadata exists. Quick Maintenance Tasks are enabled by default and will execute approximately every hour.

  While the user can disable quick maintenance, it's not recommended as it will lead to reduced performance.

- **Full Maintenance Tasks** are responsible for keeping the repository compact and eliminate deleted files that the user no longer wishes to store.

  The most important task is Snapshot GC which marks for deletion all contents that are no longer reachable from any of the active snapshots. Full Maintenance is also responsible for compaction of data pack blobs (`p`) after contents stored in them have been deleted. Full Maintenance Tasks are enabled by default and will execute every 24 hours.

Maintenance Task Ownership

For correctness reasons, Kopia requires that no more than one instance of certain maintenance operations runs at any given time. To achieve that, one repository `user@hostname` is designated as the Maintenance Owner. Other repository users will not attempt to run maintenance automatically and the designated user will attempt to do so after holding an exclusive lock.

To see the current maintenance owner use `kopia maintenance info` command:

```
$ kopia maintenance info
Owner: root@myhost
```

To change the maintenance owner to either current user or another user use `kopia maintenance set` command:

```
$ kopia maintenance set --owner=me
$ kopia maintenance set --owner=another@somehost
```

Maintenance Task Scheduling

To enable or disable quick or full maintenance:

```
$ kopia maintenance set --enable-quick true
$ kopia maintenance set --enable-full true
```

To change the quick or full maintenance interval:

```
$ kopia maintenance set --quick-interval=2h
$ kopia maintenance set --full-interval=8h
```

It is also possible to pause quick or full maintenance for some time so that it automatically resumes after specified time elapses. To change the quick or full maintenance for some time use:

```
$ kopia maintenance set --pause-quick=48h
$ kopia maintenance set --pause-full=268h
```

Manually Running Maintenance

To run maintenance manually use `kopia maintenance run`:

```
# quick maintenance
$ kopia maintenance run

# full maintenance
```

```
$ kopia maintenance run --full
```

The current user must be the maintenance owner.

Maintenance Safety

Kopia’s maintenance routine follows certain safety rules which rely on passage of time to ensure correctness. This is needed in case other Kopia clients are currently operating on the repository. To guarantee correctness, certain length of time must pass to ensure all caches and transient state are properly synchronized with the repository. Kopia must also account for eventual consistency delays introduced by the blob storage provider.

This means that effects of full maintenance are not immediate - it may take several hours and/or multiple maintenance cycles to remove blobs that are not in use.

Kopia 0.8 adds new flag that can be used to speed up full maintenance if the user can guarantee no kopia snapshots are being created.

> WARNING: As the name implies, the `--safety=none` flag disables all safety features, so the user must ensure that no concurrent operations are happening and repository storage is properly in sync before attempting it. Failure to do so can introduce repository corruption.

Example of running full maintenance with safety disabled:

```
$ kopia maintenance run --full --safety=none
```

Viewing Maintenance History

To view the history of maintenance operations use `kopia maintenance info`, which will display the history of last 5 maintenance runs.

Last modified March 21, 2023: docs(site): unified headings - created an outline within the advanced section (#2838) (8459e1c0)

Verifying Validity of Snapshots/Repositories and Trying to Repair Any Corruption

Consistency

Backing up data is great, but you also need to be able to restore that data when (if) the time arises. That means you need to test snapshots regularly to ensure that a snapshot remains valid and has not become corrupt after you have created the snapshot. (There are various different reasons why a snapshot may become corrupt; see the corruption repair discussion below for more details.)

Verifying Validity of Snapshots

The gold standard for testing the validity of backups is to simply restore the snapshot; if restore is successful, then the backup is valid. However, in many situations it is not feasible to conduct a full restore of a snapshot, such as if you do not have enough local hard drive space to do a full restore or if the egress costs of your cloud storage are extremely high (e.g., Amazon S3 charges $0.09 per GB you download).

If you cannot, or do not want to, do a full test restore of your snapshots, Kopia enables you to verify the consistency/validity of your snapshots/repositories using the `kopia snapshot verify` command. This command ensures that the directory structures in your repository are consistent by walking all files and directories in the snapshots from their roots; it verifies that the content manager index structures are correct and that every index entry is backed by an existing file. All of this is to ensure that your snapshots contain all the necessary metadata and blobs to restore each backed up file, should the need ever arise for you to restore your files.

Both Kopia CLI and KopiaUI automatically run `kopia snapshot verify` during every daily full maintenance, so you do not need to run the command yourself. However, while `kopia snapshot verify` verifies the existence of all blobs necessary to restore your snapshots, the command stops short of downloading, decrypting, and decompressing the blobs. This means that the command does not test whether blobs have been corrupted after they have been uploaded by Kopia, due to bit rot, bit flip, etc.

If you want Kopia to test for bit rot and related data corruption issues and to conduct what is essentially the equivalent of a test restore (i.e., verify the structure of your repository and the existence of all blobs AND download/decrypt/decompress all the files in your snapshots), then you need to use the `--verify-files-percent` option with the `kopia snapshot verify` command, such as `kopia snapshot verify --verify-files-percent=100 --file-parallelism=10 --parallel=10`. `--verify-files-percent=100` tells Kopia to download/decrypt/decompress 100% of the files backed up in your repository. (The `--file-parallelism=10 --parallel=10` options are optional and tell Kopia to use parallelism to speed up the process. You can exclude these two options, if you so desire.) These files are downloaded temporarily and automatically discarded once Kopia has finished with each file, so you do not require a significant amount of hard drive space to use this command. If Kopia throws no errors while running this command, then your snapshots/repositories are valid. If Kopia does throw an error, then you may have a corruption issue, in which case you need to read the corruption repair discussion below.

You can run `kopia snapshot verify --verify-files-percent=100 --file-parallelism=10 --parallel=10` monthly or at whatever interval you desire.

If you do not want Kopia to download 100% of your files, you can set `--verify-files-percent` to any percent you want. For example, `--verify-files-percent=1` will download a random selection of 1% of your data. If you are unable to, or do not want to, regularly run `kopia snapshot verify --verify-files-percent=100`, then it is recommended to at least run `kopia snapshot verify --verify-files-percent=1 --file-parallelism=10 --parallel=10` daily. If you run this command daily, statistically over the course of a year you have a roughly 98% likelihood to have tested 100% of your backed up data.

Currently, `kopia snapshot verify --verify-files-percent=# --file-parallelism=10 --parallel=10` must be run via CLI. KopiaUI does not yet have the ability to run `kopia snapshot verify` with the `--verify-files-percent` option, so all KopiaUI users will need to run the command via CLI.

Repairing Corruption of Snapshots

How Corruption Happens

Kopia data structures and algorithms are designed to maintain data consistency even in the event of a crash or failure, but in rare cases a repository may still get corrupted, usually due to hardware or software misconfiguration. Corruption is typically caused by one of three reasons:

- **Misconfigured or unsupported filesystem** – Kopia requires certain properties of storage subsystems, including POSIX semantics, atomic writes (either native or emulated), strong read-after-write consistency and eventual list-after-write consistency. Many modern filesystems (especially networked ones) are not fully POSIX compliant and don't guarantee certain consistency in case of a crash, power outage, network partitions, high latency, etc. Once such an event occurs, it's possible for the filesystem to misrepresent the set of files it is holding, possibly causing Kopia to assume certain files are unused. Kopia can compensate for such inconsistent behaviors for up to several minutes, but larger inconsistencies can lead to data loss.

  PRO TIP: It is recommended to use the most mature filesystem available for an OS with full POSIX semantics and avoid emulated, layered, or networked filesystems which may not be fully compliant. Alternatively, use one of the cloud storage repositories supported by Kopia.

- **Silent data corruption after write** – most filesystems in use today are susceptible to silent data corruption (known as bit rot, bit flips, etc.) caused by bad hardware, radiation, etc. Modern filesystems such as ZFS or BTRFS use special encodings to be able to detect and recover from such issues, but they come with various trade-offs in terms of performance and capacity, so most filesystem deployments in common use today do not have these features. (RAID storage generally only provides increased availability and not bit rot protection.) In cases when a filesystem corruption is detected, there is very little Kopia can do about it and such event will lead to data loss. It is thus recommended to use the most reliable filesystem available or preferably use cloud storage which is generally better at preventing such issues.

- **Large clock skew** – Kopia requires *reasonable* time synchronization between all clients and storage servers, where clock skews of few minutes are tolerated, but bigger clock skews can lead to major inconsistencies, including Kopia deleting its own data because clocks indicate it is no longer needed. To protect against such issues, it's recommended to run NTP time synchronization on both clients and servers in order to ensure that local clock is up-to-date (if you are using cloud storage, you only need to worry about keeping the clock for your computers up-to-date).

Repairing Corruption

Each type of data corruption is different, so there's not a single approach to data recovery.

There are few tips to try, which are generally safe to try:

1. Look for repository files with zero size and remove them. Kopia will never write such files and if they are present, it indicates filesystem data corruption. Simply removing them may work.

2. Try to open the repository from another computer. If that works, local cache may be corrupted and clearing that should fix the issue. Please do NOT clear the cache on the main computer where the corruption has manifested. Data in cache may help the recovery.

3. Look for files with unusual timestamps (like year 1970, year 2038 or similar), file extensions, etc and try to clean them up.

4. If a repository can't be opened but was working recently and maintenance has not run yet, it may be helpful to try to remove (or stash away) the most recently-written index files whose names start with `x` in the reverse timestamp order one by one until the issue is fixed. This will effectively roll back the repository writes to a prior state. Exercise caution when removing the files.

5. If the steps above do not help, report your issue on https://kopia.discourse.group or https://slack.kopia.io. Kopia has many low-level data recovery tools, but they should not be used by end users without guidance from developers.

   NOTE: Since all data corruption cases are unique, it's generally not recommended to attempt fixes recommended to other users even for possibly similar issues, since the particular fix method may not be applicable.

Last modified March 21, 2023: docs(site): unified headings - created an outline within the advanced section (#2838) (8459e1c0)

Ignoring Files and Folders in Snapshots

Ignoring Files and Folders in Snapshots

Users may want to exclude folders and files not to be saved within the repository when creating snapshots. The benefits of omitting unnecessary files and folders are smaller and faster snapshots while saving only essential data.

Kopia uses `pattern-based` ignore rules to omit folders and files from snapshots. While scanning directories and their content, Kopia looks explicitly for files that contain such rules. If such a file is placed within a directory, `Kopia` omits files and folders `matching` the rules.

> NOTE The default file is called `.kopiaignore`. However, ignore rules can be specified within the global or snapshot-specific `policy` - either directly or by providing a path to file containing such rules.

In the following, we explain different rules and provide examples to create `.kopiaignore` files.

Kopiaignore Files

Let us start with an example directory that contains the following files and folders:

```
thesis/
  --title.png
  --manuscript.tex
  --figures/
 --architecture.png
 --server.png
  --chapters/
 --introduction.tex
 --abstract.tex
 --conclusion.tex
 --logs/
     --chapter.log
  --logs/
 --gen.log
 --fail.log
 --log.db
 --tmp.db
 --tmp.dba
  --tmp.db
  --atmp.db
  --abtmp.db
  --logs.dat
```

The above directory consists of a bunch of `tex` files, figures, and temporary files. Generally, a `.kopiaignore` file is a simple text file where each line represents a single rule. To only save the essential files, we create the following `.kopiaignore` file:

```
# Ignoring all files that end with ".dat"
*.dat

# Ignoring all files and folders within the "thesis/logs" directory
/logs/*

# Ignoring "tmp.db" files within the whole directory
tmp.db
```

The example above contains three simple rules to exclude files and folders from a `snapshot` and some comments. Each line that begins with a `#` is a `comment` and can be used to describe the rule.

- The first rule, `*.dat` contains a `wildcard` and ignores all files with a filename that ends with `.dat`.
- The second rule, `/logs/*` ignores all files within the `logs` directory. Only the `logs` directory at the `root` will be ignored as the rule begins with a `/`.
- The third rule, `tmp.db` ignores the corresponding files within the whole directory. In our example both `tmp.db` files will be ignored.

The example shows that excluding files using `.kopiaignore` from a snapshot is easy. However, there is also the risk of accidentally excluding files when creating rule - leading to incomplete snapshots or data loss.

Supported Patterns

`Kopia` supports a lot of different operators allowing users to precisely exclude unnecessary files or folders. The following table shows special operators used to generate rules.

Special Operator	Explanation

Special Operator	Explanation
`#`	`Comment` that is ignored by `Kopia`
`!`	`Negates` a following rule
`*`	`Wildcard` that matches any character zero or multiple times
`**`	Double `Wildcard` that matches zero or multiple directories
`?`	Matches any character exactly `one` time
`[0-9]`	Matches any single number between `0` and `9`
`[a-z]`	Matches any single character between `a` and `z`
`[A-Z]`	Matches any single character between `A` and `Z`
`[abc]`	Matches one of `a`, `b`, or `c`
`/`	Matches a following rule only at the `root` directory

Examples of Kopiaignore Rules

The following table provides some example rules related to our example. Files and folders that `match` the given rules are excluded from the snapshot.

Rule	Explanation	Matches	Ignores
`logs`	Matches files and folders that are named `logs`	thesis/logs/ thesis/chapters/logs/	2 directories, 6 files
`/logs`	Matches files and folders that are named `logs` only within the parent directory	thesis/logs/	1 directory, 5 files
`*.db`	Matches files with extension `.db`	(…) thesis/tmp.db thesis/logs/log.db	0 directories, 5 files
`*.db*`	Matches files with extension `.db` followed by any other number or character	(…) thesis/tmp.db thesis/logs/tmp.dba	0 directories, 6 files
`**/logs/**`	Matches all occurrences of `logs` within the `thesis` and sub-directories	(…) thesis/logs/ thesis/chapters/logs/	2 directories, 6 files
`chapters/**/*.log`	Matches all files with extension `.log` in all sub-directories within `chapters`	thesis/chapters/logs/chapter.log	0 directories, 1 file
`*.*`	Matches all files in `thesis`	(…) thesis/ thesis/tmp.db	5 directories, 17 files (all)
`!*.*`	Matches no files in `thesis`	-	0 directories, 0 files
`[a-z]?tmp.db`	Matches files beginning with characters between `a` and `z`, followed by a single character, ending with `tmp.db`	thesis/abtmp.db	0 directories, 1 file
`?tmp.db`	Matches files with exactly one character ending with `tmp.db`	thesis/atmp.db	0 directories, 1 file
`[a-z]*tmp.db`	Matches files beginning with characters between `a` and `z`, followed by zero or multiple characters, ending with `tmp.db`	thesis/abtmp.db thesis/atmp.db thesis/logs/tmp.db	0 directories, 3 files

> NOTE Make sure that you have tested your `.kopiaignore` file and the resulting snapshot for correctness. If a file or folder is missing, you will need to adjust the rules to your needs.

Last modified April 17, 2024: chore: fix some typos in comments (#3805) (f125f09d)

Actions

Actions

Starting with v0.8 Kopia supports running custom user-provided commands or scripts before and after snapshot root and also before/after individual folders as they get snapshotted. This supports scenarios such as:

- creating snapshots of filesystems that support it (e.g. ZFS)
- snapshotting databases or virtual machines as part of taking a snapshot
- sending notifications to users, etc.

Actions can optionally modify the directory to be snapshotted or redirect upload to another directory (typically a mountpoint representing filesystem snapshot).

Enabling Actions

To reduce the security risk, actions are an opt-in feature and are not enabled by default.

When using Kopia CLI, actions can be enabled globally at connection time or individually per snapshot:

1. When connecting to repository you can pass `--enable-actions` which will enable actions globally for the client.
2. You can override that decision when taking snapshot by passing `--force-enable-actions` or `--force-disable-actions` to enable or disable actions for the single snapshot session.

When using KopiaUI, actions can be enabled globally by editing your repository.config (it is located in the `Config File` location in KopiaUI under `Repository`) and change `"enableActions": false` to `"enableActions": true`. Save the file and restart KopiaUI.

Configuring actions

To set the script for a directory we can use `kopia policy set` on a directory.

For example:

```
$ kopia policy set /some/dir --before-folder-action /path/to/command
$ kopia policy set /some/dir --after-folder-action /path/to/command
$ kopia policy set /some/dir --before-snapshot-root-action /path/to/command
$ kopia policy set /some/dir --after-snapshot-root-action /path/to/command
```

> NOTE: Unlike all other policy options, `--before-folder-action` and `--after-folder-action` are not inherited and must be set explicitly on target folders, while `--before-snapshot-root-action` and `--after-snapshot-root-action` are inherited from their parents and can be set at global, host, user or directory level.

Actions can be `essential` (must succeed, default behavior), `optional` (failures are tolerated) or `async` (kopia will start the action but not wait for it to finish). This can be set using `--action-command-mode`, for example:

```
$ kopia policy set /some/dir  --action-command-mode=async \
   --before-folder-action /usr/local/bin/notifier.sh
```

Each action has an associated timeout (by default 5 minutes), which specifies how long it will be allowed to run before Kopia kills the process. This can be overridden using `--action-command-timeout`:

```
$ kopia policy set /some/dir  --action-command-timeout=180s \
   --before-folder-action /usr/local/bin/notifier.sh
```

Finally, the action command itself can be stored in a repository, when `--persist-action-script` is passed. To prevent binaries from being stored, the maximum script length can be up to 32000 characters.

Scripts stored like this will be temporarily extracted to a local directory and executed using a shell command, which is:

- On Linux and macOS: `sh -e /path/to/temporary/script/file.sh`
- On Windows: `cmd.exe /c C:\path\to\temporary\script\file.cmd`

On Unix, if the script has `#!` prefix, it will be executed directly, bypassing the `/bin/sh` shell.

Action Environment

When kopia invokes `Before` actions, it passes the following parameters:

Variable	Description
`KOPIA_ACTION`	`before-folder` or `before-snapshot-root`
`KOPIA_SNAPSHOT_ID`	Random 64-bit number
`KOPIA_SOURCE_PATH`	Path being snapshotted
`KOPIA_SNAPSHOT_PATH`	Path being snapshotted
`KOPIA_VERSION`	Version of Kopia (e.g. `0.9.2`)

The action command can modify the contents source directory in place or it can request other directory be snapshotted instead by printing a line to standard output:

```
KOPIA_SNAPSHOT_PATH=<new-directory>
```

This can be used to create point-in-time snapshots - see examples below.

The `After` action will receive similar parameters as `Before` plus the actual directory that was snapshotted (either `KOPIA_SOURCE_PATH` or `KOPIA_SNAPSHOT_PATH` if returned by the `Before` script).

Variable	Description
`KOPIA_ACTION`	`after-folder` or `after-snapshot-root`
`KOPIA_SNAPSHOT_ID`	Random 64-bit number
`KOPIA_SOURCE_PATH`	Source path being snapshotted
`KOPIA_SNAPSHOT_PATH`	Actual path being snapshotted (returned by the *before* action)
`KOPIA_VERSION`	Version of Kopia (e.g. `0.9.2`)

Examples

Dumping SQL databases before snapshotting:

This script invokes `mysqldump` to create a file called `dump.sql` in the directory that's being snapshotted. This can be used to automate database backups.

```
#!/bin/sh
set -e
mysqldump SomeDatabase --result-file=$KOPIA_SOURCE_PATH/dump.sql
```

ZFS point-in-time snapshotting:

When snapshotting ZFS pools, we must first create a snapshot using `zfs snapshot`, mount it somewhere and tell `kopia` to snapshot the mounted directory instead of the current one.

After snapshotting, we need to unmount and destroy the temporary snapshot using `zfs destroy` command:

Before:

```
#!/bin/sh
set -e
ZPOOL_NAME=tank
zfs snapshot $ZPOOL_NAME@$KOPIA_SNAPSHOT_ID
mkdir -p /mnt/$KOPIA_SNAPSHOT_ID
mount -t zfs $ZPOOL_NAME@$KOPIA_SNAPSHOT_ID /mnt/$KOPIA_SNAPSHOT_ID
echo KOPIA_SNAPSHOT_PATH=/mnt/$KOPIA_SNAPSHOT_ID
```

After:

```
#!/bin/sh
ZPOOL_NAME=tank
umount /mnt/$KOPIA_SNAPSHOT_ID
rmdir /mnt/$KOPIA_SNAPSHOT_ID
zfs destroy $ZPOOL_NAME@$KOPIA_SNAPSHOT_ID
```

LVM Snapshots:

When snapshotting filesystems using LVM snapshots, we must first create a LVM snapshot using `lvcreate` and mount the filesystem inside the LVM snapshot somewhere. Then we tell `kopia` to snapshot the mounted directory instead of the current one. Make sure to match the snapshot size with your requirements. The snapshot grows with the delta to the origin logical volume. You also need to make sure to have enough free space in your volume group, otherwise the snapshot creation will fail.

After snapshotting, we need to unmount and remove the temporary logical volume using `lvremove` command:

Before:

```
#!/bin/sh
set -e
VG_NAME=vg0
LV_NAME=lv-root
SNAPSHOT_SIZE=10G
lvcreate -L ${SNAPSHOT_SIZE} -s -n $KOPIA_SNAPSHOT_ID $VG_NAME/$LV_NAME
mkdir -p /mnt/$KOPIA_SNAPSHOT_ID
mount /dev/$VG_NAME/$KOPIA_SNAPSHOT_ID /mnt/$KOPIA_SNAPSHOT_ID
echo KOPIA_SNAPSHOT_PATH=/mnt/$KOPIA_SNAPSHOT_ID
```

After:

```
#!/bin/sh
VG_NAME=vg0
umount /mnt/$KOPIA_SNAPSHOT_ID
rmdir /mnt/$KOPIA_SNAPSHOT_ID
lvremove -f $VG_NAME/$KOPIA_SNAPSHOT_ID
```

Windows shadow copy

When backing up files opened with exclusive lock in Windows, Kopia would fail the snapshot task because it can't read the file content. One of the popular solutions is taking a shadow copy of the storage volume and ask Kopia to backup that instead.

In this example, we will use PowerShell to take a shadow copy in the "before" action of the target directory and clean everything up in the "after" action. The script also self-elevates as administrator (required to take shadow copy) if Kopia is ran with an unprivileged account.

Make sure `powershell` is reachable in the PATH environment variable.

before.ps1:

```
if ($args.Length -eq 0) {
    $kopiaSnapshotId = $env:KOPIA_SNAPSHOT_ID
    $kopiaSourcePath = $env:KOPIA_SOURCE_PATH
} else {
    $kopiaSnapshotId = $args[0]
    $kopiaSourcePath = $args[1]
}

$sourceDrive = Split-Path -Qualifier $kopiaSourcePath
$sourcePath = Split-Path -NoQualifier $kopiaSourcePath
# use Kopia snapshot ID as mount point name for extra caution for duplication
$mountPoint = "${PSScriptRoot}\${kopiaSnapshotId}"

if (([Security.Principal.WindowsPrincipal][Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole([Security.Principal.WindowsBuiltInRole] 'Administrator')) {
    $shadowId = (Invoke-CimMethod -ClassName Win32_ShadowCopy -MethodName Create -Arguments @{ Volume = "${sourceDrive}\" }).ShadowID
    $shadowDevice = (Get-CimInstance -ClassName Win32_ShadowCopy | Where-Object { $_.ID -eq $shadowId }).DeviceObject
    if (-not $shadowDevice) {
        # fail the Kopia snapshot early if shadow copy was not created
        exit 1
    }

    cmd /c mklink /d $mountPoint "${shadowDevice}\"
} else {
    $proc = Start-Process 'powershell' '-f', $MyInvocation.MyCommand.Path, $kopiaSnapshotId, $kopiaSourcePath -PassThru -Verb RunAs -WindowStyle Hidden -Wait
    if ($proc.ExitCode) {
        exit $proc.ExitCode
    }
}

Write-Output "KOPIA_SNAPSHOT_PATH=${mountPoint}${sourcePath}"
```

after.ps1:

```
if ($args.Length -eq 0) {
    $kopiaSnapshotId = $env:KOPIA_SNAPSHOT_ID
} else {
    $kopiaSnapshotId = $args[0]
}

if (([Security.Principal.WindowsPrincipal][Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole([Security.Principal.WindowsBuiltInRole] 'Administrator')) {
    $mountPoint = Get-Item "${PSScriptRoot}\${kopiaSnapshotId}"
    $mountedVolume = $mountPoint.Target

    cmd /c rmdir $mountPoint
    Get-CimInstance -ClassName Win32_ShadowCopy | Where-Object { "$($_.DeviceObject)\" -eq "\\?\${mountedVolume}" } | Remove-CimInstance
} else {
    Start-Process 'powershell' '-f', $MyInvocation.MyCommand.Path, $kopiaSnapshotId -Verb RunAs -WindowStyle Hidden -Wait
    if ($proc.ExitCode) {
        exit $proc.ExitCode
    }
}
```

To install the actions:

```
kopia policy set <target_dir> --before-folder-action "powershell -WindowStyle Hidden -File <path_to_script>\before.ps1"
kopia policy set <target_dir> --after-folder-action  "powershell -WindowStyle Hidden -File <path_to_script>\after.ps1"
```

Contributions Welcome

Those are just some initial ideas, we're certain more interesting types of actions will be developed using this mechanism, including LVM snapshots, BTRFS Snapshots, notifications and more.

If you have ideas for extending this mechanism, definitely file an Issue on Github.

If you develop a useful action script that you'd like to share with the community, we encourage you to do so by sending us a pull request to add to this web page or you can put them in your own repository and we'll be happy to link it from here.

To get started, click 'Edit This Page' link.

Last modified November 2, 2023: fix(vss): add missing -File for powershell vss script policy setup (#3424) (77bac549)

Caching

Caching

To optimize performance Kopia maintains local cache of contents, metadata, indexes and more. This document provides more insight into this process.

Cache Directory Location

Default Cache Directory Location varies by operating system:

- On Linux - `~/.cache/kopia/{unique-id}`
- On macOS - `~/Library/Caches/kopia/{unique-id}`
- On Windows - `%LocalAppData%\kopia\{unique-id}`

Where `{unique-id}` is a hash of configuration file used and unique identifier that's specific to a connected repository.

The cache directory location can be overridden when connecting to a repository by specifying `--cache-directory` flag or `KOPIA_CACHE_DIRECTORY` environment variable. It will be persisted in the configuration file.

When set `KOPIA_CACHE_DIRECTORY` environment variable takes precedence over location stored in the configuration file.

Cache Types

Kopia maintains several different types of caches for different purposes:

- `metadata` (encrypted in storage) - stores directory listings, manifests and other data stored in `q` blobs in the repository. Each cache entry stores the whole original `q` blob (usually ~20 MB each).

- `contents` (encrypted in storage) - stores contents from `p` blobs. Unlike the metadata cache, elements of the contents cache store sections of the underlying blobs.

- `indexes` (non-encrypted) - contains locally-cached index information (mapping of contents to their location in blobs) in a format that can be directly memory-mapped for efficient access.

- `own-writes` (non-encrypted) - keeps track of names and timestamps of index blobs written by local Kopia to ensure it can see its own writes, even if the underlying storage list mechanism is eventually consistent and written files show up after some delay.

- `blob-list` (non-encrypted) - short-lived cache (by default 30 seconds) to avoid frequent listing of blobs in the underlying storage

- `server-contents` (encrypted locally) - contents downloaded from the repository server.

Setting Cache Parameters

You can override cache sizes, durations and locations by using `kopia cache set`:

```
# set size to 20GB and override location
$ kopia cache set --content-cache-size-mb=20000 --cache-directory=/var/my-cache
```

To override `blob-list` cache duration:

```
$ kopia cache set --max-list-cache-duration=300s
```

Note the cache sizes are not hard limits: cache is swept periodically (every few minutes) to bring the total usage below the defined limit by removing least-recently used cache items.

A hard limit can be set if required via the corresponding `limit` flag:

```
# set the maximum content cache size to 30GB
$ kopia cache set --content-cache-size-limit-mb=30000
# set the maximum metadata cache size to 20GB
$ kopia cache set --metadata-cache-size-limit-mb=20000
```

Clearing Cache

Cache can be cleared on demand by `kopia cache clear` or by simply removing appropriate files. It is always safe to remove files from cache.

Last modified May 7, 2024: docs(cli): cache hard limits flags (#3846) (e5790e33)

Ransomware Protection

Some cloud storage providers provide capabilities targeted at protecting against ransomware attacks. Kopia can be configured to take advantage of those capabilities to provide additional protection for your data.

What is ransomware and why do we need extra protections?

For the context of Kopia protection, ransomware refers to viruses, trojans or other malware that infects a system, and blocks user access to it (often by overwriting files with an encrypted version of themselves), usually requiring to pay for the decryption key to restore access. Because ransomware often targets 'high-value' user data, and often overwrites files in place, an automated backup solution (for instance running Kopia on a nightly cron-job) can cause the overwritten files to be propagated to cloud storage. For the case just described, having multiple snapshots would appear to be sufficient to restore your data from the cloud, however ransomware attacks are getting more sophisticated. A ransomware application may look for your cloud provider access keys while examining your files and use that to permanently delete your snapshots on the cloud! We need to better protect our cloud storage to ensure our snapshots are safe.

Some notes about storage providers

- Kopia's AWS S3 storage engine supports using both restricted access keys as well as object-locks for ransomware protection.
  - Many storage providers with S3 compatibility (for instance Backblaze B2) can take full advantage of Kopia's ransomware protection when used with the S3 storage engine.
  - Note that not all S3 compatible providers provide sufficient access controls to take advantage of these features. Notably, Google Cloud Storage (GCS) (see below).
- Kopia's Backblaze B2 storage engine provides support for using restricted access keys, but not for object locks at the current time.
  - To use storage locks with Backblaze B2, use the S3 storage engine.
- Kopia's Azure & Google storage engines support object-locks for ransomware protection.

Using application keys to protect your data

Some cloud storage solutions provide the ability to generate restricted access keys. These keys can be configured to only allow access to specific data (a specific bucket or path within a bucket), as well as to restrict what capabilities that key has access to. The easiest solution is to generate a key without any `delete` permissions, and to configure Kopia to use that key. Now if a ransomware application finds your key, it can no longer permanently delete any data from the cloud! This might imply that now Kopia can no longer delete old snapshots as part of its maintenance cycle, but that is not the case. When Kopia deletes data from a compatible provider, it is replacing the data with a special file that has a `hidden` marker. This makes the file appear to be deleted, but it can still be accessed by using an older version of the file). Typically, the cloud provider offers 'Lifecycle-Management' to apply a true-deletion of hidden data after a certain period of time. Since this is an automated process executed by the cloud provider, no 'true delete' is ever executed by Kopia. As long as the hidden-to-delete time delay is long enough for you to notice the ransomware, you can still restore the old versions of your data. Enabling restricted keys does not require any changes in your Kopia workflow, since Kopia does not need to change its behavior at all. The cost of this is that data will remain on the cloud provider for extra time before being deleted, potentially incurring additional storage charges.

How to configure restricted access keys

AWS

- Create a new application key
  - Create a IAM user for kopia to use
    - Select 'Attach policies directly'
    - Create a new policy, with the following permissions (paste into JSON form)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1480692207000",
            "Effect": "Deny",
            "Action": [
                "s3:DeleteBucket",
                "s3:DeleteBucketPolicy",
                "s3:DeleteBucketWebsite",
                "s3:DeleteObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::*"
            ]
        }
    ]
}
```

    - Attach created policy to new user
  - Manage user's security-credentials and create a new access key
- Disconnect and reconnect your existing Kopia repo using the new key (or create a new bucket using this key)
- Regenerate (or delete) your root application key if you have one
- Enable Lifecycle management for the Kopia bucket setting the 'Expiration' action to the time you want to ensure your data is protected for

Backblaze

- Backblaze does not allow creation of an application key with restricted permission from the website. Instead, you must use the cli-application to generate a restricted key.
  - Download the appropriate CLI application from Backblaze
  - Generate a new master API key on the website
  - Set the following environment variables from your Master API key
    - B2_APPLICATION_KEY_ID=xxxxxxxxxxxxxxxxxxxxxxxxx
    - B2_APPLICATION_KEY=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  - Run: `b2 create-key --bucket <bucket-name> <key-name> listBuckets,readBuckets,listFiles,readFiles,writeFiles,readBucketEncryption,readBucketReplications,readBucketRetentions,readFileRetentions,writeFileRetentions,readFileLe`
  - The resulting string contains the restricted application key and password…you will need this for Kopia
  - Disconnect and reconnect your existing Kopia repo using the new key (or create a new bucket using this key)
  - Re-generate your API master API key (to deactivate the one you just used)
  - Delete any existing Application keys that do not have restricted access

Even more protection

So far, we have secured your access such that even if a bad actor gets access to your Kopia configuration, they can't do irreparable harm to your cloud backup. However, what if they get access to your login credentials? Your login credentials provide the ability to delete your data and even your entire buckets for all the buckets in your account. But the cloud providers have protection from that too.

Multi-factor-authentication (MFA) is one option. With MFA enabled, an attacker would need access to your password as well as your security device to be able to manipulate your account. All major providers support MFA, and it is recommended to use it to secure your account. Note that it is important to eliminate and root/global acess keys as well, since they can generally be used to execute nearly any task you can do when logged in (effectively bypassing MFA).

An additional layer of protection is `Object Locking' that can be enabled in AWS (and S3 compatible providers). An Object Lock is applied when a file is created (or on an existing file), and it provides a specific retention date. Until that retention date occurs, there is no way to delete the locked file. Even using your login credentials, the file is protected from deletion. It can still be overwritten with a new version or hidden such that it doesn't appear in a file list. But it will always be accessible until its retention date occurs. While Kopia supports applying Object Locks, there are some caveats:

- Object Locks must be enabled when a bucket is created. If you already have backups in the cloud, you will need to create a new bucket with Object Locks turned on (NOTE: On Backblaze S3, object-lock can be enabled on existing buckets via `b2 update bucket`). Once a bucket has Object Lock enabled, it cannot be disabled.
- You must enable Object Lock extension in Kopia. By default, Kopia does not renew Object Lock retention dates, however this can be enabled in the `full-maintenance` options. You must ensure that you run full maintenance at least as frequently as your Object Lock period to ensure Object Locks do not expire, otherwise the Retention date will pass, and your data will no longer be protected by an Object Locks.
- It is strongly recommended to use compliance mode when creating the Kopia repository. Compliance mode ensures that even root users cannot delete files from a bucket, and provides the highest level of security. More information can be found in the S3 documentation
- Kopia currently only supports object locks when using an S3 repo. If you use Backblaze, you will need to use the S3 repo method with Kopia instead of the Backblaze native API.

How to enable Object Locks

- Create a new bucket with Object Locks enabled
  - There is no need to set a default Retention time
  - Backblaze S3 uses can use `b2 update bucket` to enable Object Locks on an existing bucket
- Optional: Create a restricted access key as instructed above for additional protection
- Create a new Kopia repository
  - Run: `kopia repo create s3 --bucket <bucket name> --access-key=<access key> --secret-access-key=<secret access key> --retention-mode COMPLIANCE --retention-period`

`<retention time>`
        - `retention-period` can be specified in days (for instance 30d)
- Enable Object Lock extension
    - Run: `kopia maintenance set --extend-object-locks true`
        - Note that the `full-interval` must be at least 1 day shorter than the `retention-period` or Kopia will not allow you to enable Object Lock extension

How to restore a snapshot that was deleted by ransomware (or some other process)

- If you have data that needs to be restored, make sure that either your retention time will not expire or your lifecycle data expiration is sufficient to ensure you can download your data before the cloud provider removes it
- Disconnect the repo in Kopia
- Reconnect the repo in Kopia using the `--point-in-time` option (ex: `--point-in-time=2021-11-29T01:10:00.000Z`)
    - This option is only currently available when using an 's3' repo

A caveat about ransomware protection

Ransomware can be very sophisticated software. It may run on your system for weeks, slowly encrypting data that is not frequently used, and only hitting actively used data last (to avoid detection for as long as possible). This means that you should ensure your data retention period is long enough that you will be able to recover your data. It also means that recovery may not be easy. You may have many snapshots after the ransomware has started modifying your system, and you may have made changes throughout that time such that there is no single snapshot that represents a good state of all your files. Restoration is likely to be a time-consuming task, and your best bet is to protect your system from these threats, so they don't occur in the first place. But if they do, Kopia can provide some additional security to protect your data.

Additionally note that ransomware could theoretically weaponize object-locks to cost you a lot of money. Because object-locks cannot be reduced or removed, sufficiently malicious ransomware could upload large amounts of data and set a very long object lock that would make it impossible to delete. It is strongly recommended to ensure you have appropriate quotas/limits on your buckets to limit potential storage costs.

An additional note about Lifecycle Management vs retention-time

At first glance, Lifecycle Management and retention-time may seem to serve similar purposes. However, if only using Lifecycle Management, an attacker could still log into your account and delete the entire bucket, or otherwise force-delete a file. Using 'Object Lock' with retention-time provides an additional guarantee that the only way for data to be lost before the retention-time expires would be to delete your account altogether. The S3 provider may allow enabling Object Lock without enabling Lifecycle Management. When retention-time is applied to a file, and that file is deleted, the S3 service will set a `DELETE` marker instead of actually deleting the file. If Lifecycle Management is not enabled, then files may remain in the repository with the `DELETED` tag indefinitely. Thus, it is recommended to enable Lifecycle Management whenever using a retention-time in Kopia to balance protective measures against escalating storage costs.

For simplicity, the recommendation is to use the same time period for Lifecycle Management and for retention-time, however, this is not a hard requirement. It is possible to set a very short Lifecycle Management period and a long retention-time (in which case files will be permanently deleted soon after the retention-time expires. Alternatively, the Lifecycle Management could be set to be significantly longer than the retention time. This would provide additional restore capabilities while allowing for manual cleanup of deleted files should it be necessary (with the understanding that once the retention-time expires, the ransomware protention is reduced). For simplicity, the recommendation is to use the same time period for Lifecycle Management and for retention-time.

Azure protection

Kopia supports ransomware protection for Azure in a similar manner to S3. The container must have version-level immutability support enabled and the related storage account must have versioning enabled. When this is configured, the retention mode can be set to either compliance or governance mode. In both cases the blobs will be in Locked mode.

Follow these steps to enable versioning on the storage account and these steps to enable version-level immutability support on the container or related storage account.

On Kopia side `--retention-mode COMPLIANCE --retention-period <retention time>` should be set like above.

To have continuous protection it is also necessary to run: `kopia maintenance set --extend-object-locks true`

- Note that the `full-interval` must be at least 1 day shorter than the `retention-period` or Kopia will not allow you to enable Object Lock extension

Google protection

Kopia supports ransomware protection for Google in a similar manner to S3. The bucket must have both versioning and object retention enabled. When this is configured, the retention mode can be set to either compliance or governance mode. In both cases the blobs will be in Locked mode.

On Kopia side `--retention-mode COMPLIANCE --retention-period <retention time>` should be set like above.

To have continuous protection it is also necessary to run: `kopia maintenance set --extend-object-locks true`

- Note that the `full-interval` must be at least 1 day shorter than the `retention-period` or Kopia will not allow you to enable Object Lock extension

If using minimal permissions with the credentials, `storage.objects.setRetention` permission is also required.

Last modified October 23, 2024: list all required permissions for GCS (#4193) (9587d982)

Synchronization

Synchronization

Maintaining multiple copies of a repository is important for disaster recovery scenarios. While cloud-based repositories often have better durability than local ones, a local repository copy may help speed up data recovery.

Kopia v0.6.0 adds support for automatic repository replication, which enables incremental copies of the currently connected repository to a separate storage location.

Any repository location can be used as target. For example:

```
$ kopia repository sync-to filesystem --path /dest/repository
$ kopia repository sync-to gcs --bucket my-bucket
```

The calling user must have read access to current repository and read/write access to the destination.

By default, synchronization does not perform any deletions in the destination location, even if the source file has been deleted. Without deletions, the resulting repository will still be correct, but will not benefit from compaction and will run more slowly. To allow deletions, pass `--delete` option:

```
$ kopia repository sync-to filesystem --path /dest/repository --delete
```

When synchronizing to a filesystem location, it is important to check that the filesystem is mounted correctly. To ensure that Kopia will not unnecessarily download potentially large repositories when destination filesystem is accidentally unmounted, pass `--must-exist`:

```
$ kopia repository sync-to filesystem --path /dest/repository --must-exist
```

Last modified March 24, 2023: docs(site): added documentation on kopiaignore (#2844) (c8d6112a)

Sharding

Sharding

Sharding is a feature introduced in Kopia v0.9.0 that allows user to customize the file system structure of a repository. Sharded repository looks like this on disk:

```
│   .shards
│   kopia.maintenance.f
│
├───n00
│   └───255
│           6c89ca6a0337723cbf33b5a198d-s31ab5f6a5cf09672108.f
│
├───n4e
│   └───ad3
│           404342ae8b978e64b7bcfc7d6ff.f
│
├───p00
│   ├───003
│   │       237e0ac3607edbee743a26b0b34.f
│   │
│   ├───011
│   │       d5912b0673ffc2c192753c3d345.f
│   │       b9021cd457094b947de593d2e84.f
│   │
│   ├───023
│   │       a6fcfcf17352c79dc7d81e2656e-saae4f33c0b1ba0c410a.f
│   │
│   ├───03a
│   │       51e5deac78855eb30239d3be7d6.f
│   │       db5f6ba893482ea83b190651c49.f
│   │
```

Notice there are two levels of directories leading to all the data files at the third level. The actual blob hash can be re-constructed by prefixing the filename with all its parent directory names. For example, blob `p00003237e0ac3607edbee743a26b0b34` is stored at `<repo_root>/p00/003/237e0ac3607edbee743a26b0b34.f`.

If multiple blob hashes share the same prefix, they are placed in the same directory.

Motivation

Sharding is introduced to help improve performance of large repositories. It is common to have hundreds of thousands of data files for repositories over 1 TB. Not all file systems handle these many files in the same directory efficiently. By breaking them into multiple levels, with each only a few hundreds, the performance is thus improved.

For small repositories, sharding may not be necessary and can be turned off.

.shards

The layout is controlled by a ".shards" file located in every repository. It is a JSON file with the following structure:

```json
{
    "default": [2, 3],
    "maxNonShardedLength": 20,
    "overrides": [
        { "prefix": "p", "shards": [2, 2] },
        { "prefix": "q", "shards": [3] }
    ]
}
```

`default` is an integer array, that applies the sharding config to all non-overridden blobs. Each element in the array represents one level of directory, and the integer value specifies the length of the directory name. `[2, 3]` means "Take the first 5 characters of each blob hash, split into 2 and 3 as directory names, and put the remaining hash as filename". `[2, 2, 4]` for blob hash `abcdefghijklmn` will become `<repo_root>/ab/cd/efgh/ijklmn.f`.

`maxNonShardedLength` makes blob hashes with length less than its value always unsharded. With value 20, it means if a blob hash is less or equal to 20, such as `e213ff706a0d404e8320`, the file is ignored by the sharding process.

`overrides` is an array that allows user to customize the sharding config for specific prefix. Each entry contains an optional `prefix` for specifying target blobs, and a required `shards` for specifying config.

When choosing the number, the rule of thumb is that larger value leads to more directories in that level, assuming the blob hash is evenly distributed (as they statistically should converge to, ignoring those pre-defined prefixes). Therefore if user notices too many files exist in directories given a `[2, 2]` config, it might be good idea to change it to `[3, 3]`. On the other hand, if too many directories are created, one should consider reducing the value.

Use

New repositories are created with some shard config, whose default value may change between Kopia version. If the default is undesirable, user can change it with `blob shards modify` command. For example,

```
kopia blob shards modify --default-shards=0 --i-am-sure-kopia-is-not-running --path=<repo_root>
```

turns off sharding.

When creating or syncing to a repository backed by cloud storage, user can also consider disabling sharding if needed, since cloud storage by nature is distributed to begin with. Some `repository sync-to` commands come with `--flat` flag to achieve this:

```
kopia repository sync-to rclone --remote-path=<remote_repo> --flat
```

This guarantees the remote repository is flat (`"default": []`) regardless if the local repository is sharded or not.

Last modified March 24, 2023: docs(site): added documentation on kopiaignore (#2844) (c8d6112a)

Compatibility

Compatibility

Kopia uses semantic versioning.

In order to ensure that snapshots created with Kopia will be available as the project evolves, starting with `v0.3.0` release Kopia is offering the following compatibility promise:

1. Each version of Kopia will be able to read snapshots created using current and **at least one previous version** of the software. The *previous version* is to be interpreted as:

- for releases with major version == `v0` (i.e. `v0.x.y`), *previous version* means previous *minor version* (`v0.(x-1).*`)
- for releases with major version == `v1`, *previous version* is the *last minor release* of the `v0` major version.
- for releases with major version >= `v2` (i.e. `vx.y.z`), *previous version* means previous *major* version (`v(x-1)`).

2. While not explicitly guaranteed, it is possible and likely that Kopia will be able to read (but not necessarily write) snapshots created with even older versions of software than explicitly guaranteed. For example, it's likely that Kopia `v0.6.0` will read snapshots created using `v0.3.0`, even though it's three releases behind.

3. In order to avoid corrupting data, Kopia will refuse to mutate repositories it's not designed to safely handle. That means, it will typically only allow reading, but not writing older versions of repository, unless explicitly documented and tested.

4. Kopia may support for writing old repository formats on a best-effort basis.

- For example Kopia `v0.4.x` may write using repository format created by `v0.3.x` in a way that `v0.3.x` versions will continue to understand it.

5. Each version of Kopia will offer migration mechanism to bring old readable repository format to the current format and thus enable full read-write operation.

6. Kopia will never upgrade old repository format to a new version without explicit human action.

Last modified March 24, 2023: docs(site): added documentation on kopiaignore (#2844) (c8d6112a)

Logging

Logging

Kopia maintains diagnostic logging for troubleshooting purposes. This documents describes parameters that can be set to configure logging:

Log File Location

The location of log directory varies by operating system:

- On Linux - `~/.cache/kopia`
- On macOS - `~/Library/Logs/kopia`
- On Windows - `%LocalAppData%\kopia`

Log file location can be overridden by setting flag `--log-dir` or `KOPIA_LOG_DIR` environment variable.

The log directory contains two subdirectories:

- `cli-logs` - contains one log file per each invocation of `kopia` binary and contains general-purpose logging and debugging information and may contain sensitive information like username, hostname, filenames, etc. Please sanitize contents of such log files before filing bug reports.

- `content-logs` - contains one log file per each invocation of `kopia` binary and contains low-level formatting logs but will not contain any sensitive data such as file names, hostnames, etc.

Log Retention

Log retention can be configured using flags and environment variables.

Flag	Environment Variable	Default	Description
`--log-dir-max-files`	`KOPIA_LOG_DIR_MAX_FILES`	1000	Maximum number of log files to retain
`--log-dir-max-age`	`KOPIA_LOG_DIR_MAX_AGE`	720h	Maximum age of log files to retain
`--content-log-dir-max-files`	`KOPIA_CONTENT_LOG_DIR_MAX_FILES`	5000	Maximum number of content log files to retain
`--content-log-dir-max-age`	`KOPIA_CONTENT_LOG_DIR_MAX_AGE`	720h	Maximum age of content log files to retain

Controlling Log Level

The amount of logs can be controlled using log levels:

- `debug` - most detailed logs including potentially verbose debugging information
- `info` - normal output
- `warning` - errors and warnings only
- `error` - errors only

You can control how much data is written to console and log files by using flags:

- `--log-level` - sets log level for console output (defaults to `info`)
- `--file-log-level` - sets log level for file output (defaults to `debug`)

Color Output

By default, console output will be colored to indicate different log levels, this can be disabled (useful when redirecting output to a file) with `--disable-color`. To force color colorized output when redirecting to a file use `--force-color`.

Last modified March 24, 2023: docs(site): added documentation on kopiaignore (#2844) (c8d6112a)

Command Line

Kopia provides a command-line interface (CLI) for accessing all its functions. All commands are accessible through single binary called `kopia` (or `kopia.exe` on Windows).

Kopia functionality is organized into Common Commands for typical use or Advanced Commands for low-level data manipulation or recovery. Click on the above links for more details.

Environment Variables

The following environment variables can be used to configure how Kopia runs:

Variable Name	Default	Description
`KOPIA_BYTES_STRING_BASE_2`	`false`	If set to `true`, Kopia will output storage values in binary (base-2). The default is decimal (base-10).

Connecting to Repository

Most commands require a Repository to be connected first. The first time you use Kopia, repository must be created, later on it can be connected to from one or more machines.

Creating a repository is as simple as:

```
$ kopia repository create <provider> <flags>
```

Examples:

- create repository in local filesystem
- create repository in Google Cloud Storage bucket
- create repository in S3-compatible bucket (e.g. Amazon S3, minio.io, Wasabi)

To connect to an existing repository use the same flags but instead of `create` use `connect`:

```
$ kopia repository connect <provider> <flags>
```

To disconnect:

```
$ kopia repository disconnect
```

Quick Reconnection To Repository

To quickly reconnect to the repository on another machine, you can use `kopia repository status -t`, which will print quick-reconnect command that encodes all repository connection parameters in an opaque token. You can also embed the repository password, by using `kopia repository status -t -s`.

Such command can be stored long-term in a secure location, such as password manager for easy recovery.

```
$ kopia repository status -t -s
...

To reconnect to the repository use:

$ kopia repository connect from-config --token 03Fy598cYIqbMlNNDz9VLU0K6Pk9alC...BNeazLBdRzP2MHo0MS83zRb

NOTICE: The token printed above can be trivially decoded to reveal the repository password. Do not store it in an unsecured place.
```

> NOTE: Make sure to safeguard the repository token, as it gives full access to the repository to anybody in its possession.

Configuration File

For each repository connection, Kopia maintains a configuration file and local cache:

By default, the configuration file is located in your home directory under:

- `%APPDATA%\kopia\repository.config` on Windows
- `$HOME/Library/Application Support/kopia/repository.config` on macOS
- `$HOME/.config/kopia/repository.config` on Linux

The location can be overridden using `--config-file`.

The configuration file stores the connection parameters, for example:

```
{
  "storage": {
    "type": "s3",
    "config": {
      "bucket": "some-bucket",
      "endpoint": "s3.endpoint.com",
      "accessKeyID": "...",
      "secretAccessKey": "..."
    }
  },
  "caching": {
    "cacheDirectory": "<path-to>/kopia/7c04ae89ea31e77d-1",
    "maxCacheSize": 524288000,
    "maxListCacheDuration": 600
  }
}
```

The password to the repository is stored in operating-system specific credential storage (KeyChain on macOS, Credential Manager on Windows or KeyRing on Linux).

Flags

Last modified October 24, 2022: feat(cli): Support displaying storage values in base-2 [#2492] (#2502) (c5efed01)

Flags

Flag	Short	Default	Help
`--config-file`		`repository.config`	Specify the config file to use
`--[no-]help`		`false`	Show context-sensitive help (also try –help-long and –help-man).
`--[no-]help-full`		`false`	Show help for all commands, including hidden
`--password`	`-p`		Repository password.
`--[no-]persist-credentials`		`true`	Persist credentials
`--[no-]use-keyring`		`false`	Use Gnome Keyring for storing repository password.
`--advanced-commands`			[ADV] Enable advanced (and potentially dangerous) commands.
`--[no-]auto-maintenance`		`true`	[ADV] Automatic maintenance
`--[no-]caching`		`true`	[ADV] Enables caching of objects (disable with –no-caching)
`--[no-]completion-bash`		`false`	[ADV] Output possible completions for the given args.
`--[no-]completion-script-bash`		`false`	[ADV] Generate completion script for bash.
`--[no-]completion-script-zsh`		`false`	[ADV] Generate completion script for ZSH.
`--[no-]disable-internal-log`		`false`	[ADV] Disable internal log

Flag	Short	Default	Help
`--[no-]dump-allocator-stats`		`false`	[ADV] Dump allocator stats at the end of execution.
`--[no-]enable-jaeger-collector`		`false`	[ADV] (DEPRECATED) Emit OpenTelemetry traces to Jaeger collector
`--[no-]enable-pprof`		`false`	[ADV] Expose pprof handlers
`--[no-]help-long`		`false`	[ADV] Generate long help.
`--[no-]help-man`		`false`	[ADV] Generate a man page.
`--initial-update-check-delay`		`24h`	[ADV] Initial delay before first time update check
`--[no-]list-caching`		`true`	[ADV] Enables caching of list results (disable with –no-list-caching)
`--metrics-directory`			[ADV] Directory where the metrics should be saved when kopia exits. A file per process execution will be created in this directory
`--metrics-listen-addr`			[ADV] Expose Prometheus metrics on a given host:port
`--metrics-push-addr`			[ADV] Address of push gateway
`--metrics-push-format`			[ADV] Format to use for push gateway
`--metrics-push-grouping`			[ADV] Grouping for push gateway
`--metrics-push-interval`		`5s`	[ADV] Frequency of metrics push
`--metrics-push-job`		`kopia`	[ADV] Job ID for to push gateway
`--metrics-push-password`			[ADV] Password for push gateway
`--metrics-push-username`			[ADV] Username for push gateway

Flag	Short	Default	Help
`--[no-]otlp-trace`		`false`	[ADV] Send OpenTelemetry traces to OTLP collector using gRPC
`--[no-]profile-blocking`		`false`	[ADV] Enable block profiling
`--[no-]profile-cpu`		`false`	[ADV] Enable CPU profiling
`--profile-dir`			[ADV] Write profile to the specified directory
`--profile-memory`			[ADV] Enable memory profiling
`--[no-]profile-mutex`		`false`	[ADV] Enable mutex profiling
`--[no-]progress`		`true`	[ADV] Enable progress bar
`--progress-update-interval`		`300ms`	[ADV] How often to update progress information
`--timezone`			[ADV] Format time according to specified time zone (local, utc, original or time zone name)
`--[no-]trace-storage`		`true`	[ADV] Enables tracing of storage operations.
`--track-releasable`			[ADV] Enable tracking of releasable resources.
`--update-available-notify-interval`		`1h`	[ADV] Interval between update notifications
`--update-check-interval`		`168h`	[ADV] Interval between update checks
`--[no-]upgrade-no-block`		`false`	[ADV] Do not block when repository format upgrade is in progress, instead exit with a message.
`--upgrade-owner-id`			[ADV] Repository format upgrade owner-id.

Common Commands

Common Commands

- `diff` - Displays differences between two repository objects (files or directories)
- `list` - List a directory stored in repository object
- `restore` - Restore a directory or a file
- `show` - Displays the contents of a repository object
- `mount` - Mount repository object as a local filesystem

Commands to test performance of algorithms

- `benchmark compression` - Run compression benchmarks
- `benchmark crypto` - Run combined hash and encryption benchmarks
- `benchmark splitter` - Run splitter benchmarks
- `benchmark hashing` - Run hashing function benchmarks
- `benchmark encryption` - Run encryption benchmarks
- `benchmark ecc` - Run ECC benchmarks

Notifications

- `notification profile configure email` - E-mail notification
- `notification profile configure pushover` - Pushover notification
- `notification profile configure webhook` - Webhook notification
- `notification profile delete` - Delete notification profile
- `notification profile test` - Send test notification
- `notification profile list` - List notification profiles
- `notification template list` - List templates
- `notification template set` - Set the notification template
- `notification template show` - Show template
- `notification template remove` - Remove the notification template

Commands to control HTTP API server

- `server start` - Start Kopia server
- `server acl add` - Add ACL entry
- `server acl delete` - Delete ACL entry
- `server acl enable` - Enable ACLs and install default entries
- `server acl list` - List ACL entries
- `server users add` - Add new repository user
- `server users set` - Set password for a repository user
- `server users delete` - Delete user
- `server users hash-password` - Hash a user password that can be passed to the 'server user add/set' command
- `server users info` - Info about particular user
- `server users list` - List users
- `server status` - Status of Kopia server
- `server refresh` - Refresh the cache in Kopia server to observe new sources, etc
- `server flush` - Flush the state of Kopia server to persistent storage, etc
- `server shutdown` - Gracefully shutdown the server
- `server snapshot` - Trigger upload for one or more existing sources
- `server cancel` - Cancels in-progress uploads for one or more sources
- `server pause` - Pause the scheduled snapshots for one or more sources
- `server resume` - Resume the scheduled snapshots for one or more sources
- `server throttle get` - Get throttling parameters for a running server
- `server throttle set` - Set throttling parameters for a running server

Commands to manipulate snapshots

- `snapshot copy-history` - Performs a copy of the history of snapshots from another user or host
- `snapshot move-history` - Performs a move of the history of snapshots from another user or host
- `snapshot create` - Creates a snapshot of local directory or file
- `snapshot delete` - Explicitly delete a snapshot by providing a snapshot ID
- `snapshot estimate` - Estimate the snapshot size and upload time
- `snapshot expire` - Remove old snapshots according to defined expiration policies

- snapshot fix invalid-files - Remove references to any invalid (unreadable) files from snapshots
- snapshot fix remove-files - Remove references to the specified files from snapshots
- snapshot list - List snapshots of files and directories
- snapshot migrate - Migrate snapshots from another repository
- snapshot pin - Add or remove pins preventing snapshot deletion
- snapshot restore - Restore a directory or a file
- snapshot verify - Verify the contents of stored snapshot

Commands to manipulate snapshotting policies

- policy edit - Set snapshot policy for a single directory, user@host or a global policy
- policy list - List policies
- policy delete - Remove snapshot policy for a single directory, user@host or a global policy
- policy set - Set snapshot policy for a single directory, user@host or a global policy
- policy show - Show snapshot policy
- policy export - Exports the policy to the specified file, or to stdout if none is specified
- policy import - Imports policies from a specified file, or stdin if no file is specified

Commands to manipulate repository

- repository connect server - Connect to a repository API Server
- repository connect from-config - Connect to repository in the provided configuration file
- repository connect azure - Connect to repository in an Azure blob storage
- repository connect b2 - Connect to repository in a B2 bucket
- repository connect filesystem - Connect to repository in a filesystem
- repository connect gcs - Connect to repository in a Google Cloud Storage bucket
- repository connect gdrive - Connect to repository in a Google Drive folder
- repository connect rclone - Connect to repository in a rclone-based provided
- repository connect s3 - Connect to repository in an S3 bucket
- repository connect sftp - Connect to repository in an SFTP storage
- repository connect webdav - Connect to repository in a WebDAV storage
- repository create from-config - Create repository in the provided configuration file
- repository create azure - Create repository in an Azure blob storage
- repository create b2 - Create repository in a B2 bucket
- repository create filesystem - Create repository in a filesystem
- repository create gcs - Create repository in a Google Cloud Storage bucket
- repository create gdrive - Create repository in a Google Drive folder
- repository create rclone - Create repository in a rclone-based provided
- repository create s3 - Create repository in an S3 bucket
- repository create sftp - Create repository in an SFTP storage
- repository create webdav - Create repository in a WebDAV storage
- repository disconnect - Disconnect from a repository
- repository repair from-config - Repair repository in the provided configuration file
- repository repair azure - Repair repository in an Azure blob storage
- repository repair b2 - Repair repository in a B2 bucket
- repository repair filesystem - Repair repository in a filesystem
- repository repair gcs - Repair repository in a Google Cloud Storage bucket
- repository repair gdrive - Repair repository in a Google Drive folder
- repository repair rclone - Repair repository in a rclone-based provided
- repository repair s3 - Repair repository in an S3 bucket
- repository repair sftp - Repair repository in an SFTP storage
- repository repair webdav - Repair repository in a WebDAV storage
- repository set-client - Set repository client options
- repository set-parameters - Set repository parameters
- repository status - Display the status of connected repository
- repository sync-to from-config - Synchronize repository data to another repository in the provided configuration file
- repository sync-to azure - Synchronize repository data to another repository in an Azure blob storage
- repository sync-to b2 - Synchronize repository data to another repository in a B2 bucket
- repository sync-to filesystem - Synchronize repository data to another repository in a filesystem
- repository sync-to gcs - Synchronize repository data to another repository in a Google Cloud Storage bucket
- repository sync-to gdrive - Synchronize repository data to another repository in a Google Drive folder
- repository sync-to rclone - Synchronize repository data to another repository in a rclone-based provided
- repository sync-to s3 - Synchronize repository data to another repository in an S3 bucket
- repository sync-to sftp - Synchronize repository data to another repository in an SFTP storage
- repository sync-to webdav - Synchronize repository data to another repository in a WebDAV storage

- `repository throttle get` - Get throttling parameters for a repository
- `repository throttle set` - Set throttling parameters for a repository
- `repository change-password` - Change repository password
- `repository validate-provider` - Validates that a repository provider is compatible with Kopia

diff

```
$ kopia diff <object-path1> <object-path2>
```

Displays differences between two repository objects (files or directories)

Flag	Short	Default	Help
`--diff-command`		`diff -u`	Displays differences between two repository objects (files or directories)
`--[no-]files`	`-f`	`false`	Compare files by launching diff command for all pairs of (old,new)

Argument	Help
`object-path1`	First object/path
`object-path2`	Second object/path

list

```
$ kopia list <object-path>
```

List a directory stored in repository object.

Flag	Short	Default	Help
`--[no-]error-summary`		`true`	Emit error summary
`--[no-]long`	`-l`	`false`	Long output
`--[no-]recursive`	`-r`	`false`	Recursive output
`--[no-]show-object-id`	`-o`	`false`	Show object IDs

Argument	Help
`object-path`	Path

restore

```
$ kopia restore <sources>
```

Restore a directory or a file.

Restore can operate in two modes:

- from a snapshot: restoring (possibly shallowly) a specified file or directory from a snapshot into a target path. By default, the target path will be created by the restore command if it does not exist.
- by expanding a shallow placeholder in situ where the placeholder was created by a previous restore.

In the from-snapshot mode:

The source to be restored is specified in the form of a directory or file ID and optionally a sub-directory path.

For example, the following source and target arguments will restore the contents of the 'kffbb7c28ea6c34d6cbe555d1cf80faa9' directory into a new, local directory named 'd1'

'restore kffbb7c28ea6c34d6cbe555d1cf80faa9 d1'

Similarly, the following command will restore the contents of a subdirectory 'subdir/subdir2' under 'kffbb7c28ea6c34d6cbe555d1cf80faa9' into a new, local directory named 'sd2'

'restore kffbb7c28ea6c34d6cbe555d1cf80faa9/subdir1/subdir2 sd2'

When restoring to a target path that already has existing data, by default the restore will attempt to overwrite, unless one or more of the following flags has been set (to prevent overwrite of each type):

–no-overwrite-files –no-overwrite-directories –no-overwrite-symlinks

If the '–shallow' option is provided, files and directories this depth and below in the directory hierarchy will be represented by compact placeholder files of the form 'entry.kopia-entry' instead of being restored. (I.e. setting '–shallow' to 0 will only shallow restore.) Snapshots created of directory contents represented by placeholder files will be identical to snapshots of the equivalent fully expanded tree.

In the expanding-a-placeholder mode:

The source to be restored is a pre-existing placeholder entry of the form 'entry.kopia-entry'. The target will be 'entry'. '–shallow' controls the depth of the expansion and defaults to 0. For example:

'restore d3.kopiadir'

will remove the d3.kopiadir placeholder and restore the referenced repository contents into path d3 where the contents of the newly created path d3 will themselves be placeholder files.

Flag	Short	Default	Help
`--[no-]consistent-attributes`		`false`	When multiple snapshots match, fail if they have inconsistent attributes
`--[no-]ignore-errors`		`false`	Ignore all errors
`--[no-]ignore-permission-errors`		`true`	Ignore permission errors
`--mode`		`auto`	Override restore mode
`--[no-]overwrite-directories`		`true`	Overwrite existing directories
`--[no-]overwrite-files`		`true`	Specifies whether or not to overwrite already existing files
`--[no-]overwrite-symlinks`		`true`	Specifies whether or not to overwrite already existing symlinks
`--parallel`		`8`	Restore parallelism (1=disable)
`--shallow`			Shallow restore the directory hierarchy starting at this level (default is to deep restore the entire hierarchy.)
`--shallow-minsize`			When doing a shallow restore, write actual files instead of placeholders smaller than this size.
`--[no-]skip-existing`		`false`	Skip files and symlinks that exist in the output
`--[no-]skip-owners`		`false`	Skip owners during restore
`--[no-]skip-permissions`		`false`	Skip permissions during restore
`--[no-]skip-times`		`false`	Skip times during restore

Flag	Short	Default	Help
`--snapshot-time`		`latest`	When using a path as the source, use the latest snapshot available before this date. Default is latest
`--[no-]write-files-atomically`		`false`	Write files atomically to disk, ensuring they are either fully committed, or not written at all, preventing partially written files
`--[no-]write-sparse-files`		`false`	When doing a restore, attempt to write files sparsely-allocating the minimum amount of disk space needed.

Argument	Help
`sources`	Two forms: 1. Source directory ID/path in the form of a

directory ID and optionally a sub-directory path. For example, 'kffbb7c28ea6c34d6cbe555d1cf80faa9' or 'kffbb7c28ea6c34d6cbe555d1cf80faa9/subdir1/subdir2' followed by the path of the directory for the contents to be restored.

2. one or more placeholder files of the form path.kopia-entry |

show

```
$ kopia show <object-path>
```

Displays the contents of a repository object.

Argument	Help
`object-path`	Path

mount

```
$ kopia mount [path] [mountPoint]
```

Mount repository object as a local filesystem.

Flag	Short	Default	Help
`--[no-]browse`		`false`	Open file browser
`--[no-]fuse-allow-non-empty-mount`		`false`	Allows the mounting over a non-empty directory. The files in it will be shadowed by the freshly created mount.
`--[no-]fuse-allow-other`		`false`	Allows other users to access the file system.
`--max-cached-dirs`		`100`	Limit the number of cached directories
`--max-cached-entries`		`100000`	Limit the number of cached directory entries
`--[no-]trace-fs`		`false`	Trace filesystem operations
`--[no-]webdav`		`false`	Use WebDAV to mount the repository object regardless of fuse availability.

Argument	Help
`mountPoint`	Mount point
`path`	Identifier of the directory to mount.

benchmark compression

```
$ kopia benchmark compression \
        --data-file=...
```

Run compression benchmarks

Flag	Short	Default	Help
`--algorithms`			Comma-separated list of algorithms to benchmark
`--[no-]by-alloc`		`false`	Sort results by allocated bytes
`--[no-]by-size`		`false`	Sort results by size
`--data-file`			Use data from the given file
`--[no-]deprecated`		`false`	Included deprecated compression algorithms
`--operations`		`both`	Operations
`--parallel`		`1`	Number of parallel goroutines
`--[no-]print-options`		`false`	Print out options usable for repository creation
`--repeat`		`0`	Number of repetitions
`--[no-]verify-stable`		`false`	Verify that compression is stable

benchmark crypto

```
$ kopia benchmark crypto
```

Run combined hash and encryption benchmarks

Flag	Short	Default	Help
`--block-size`		`1MB`	Size of a block to encrypt
`--[no-]deprecated`		`false`	Include deprecated algorithms
`--parallel`		`1`	Number of parallel goroutines
`--[no-]print-options`		`false`	Print out options usable for repository creation
`--repeat`		`100`	Number of repetitions

benchmark splitter

```
$ kopia benchmark splitter
```

Run splitter benchmarks

Flag	Short	Default	Help
`--block-count`		`16`	Number of data blocks to split
`--data-size`		`32MB`	Size of a data to split
`--parallel`		`1`	Number of parallel goroutines
`--[no-]print-options`		`false`	Print out the fastest dynamic splitter option
`--rand-seed`		`42`	Random seed

benchmark hashing

```
$ kopia benchmark hashing
```

Run hashing function benchmarks

Flag	Short	Default	Help
`--block-size`		`1MB`	Size of a block to hash
`--parallel`		`1`	Number of parallel goroutines
`--[no-]print-options`		`false`	Print out options usable for repository creation
`--repeat`		`10`	Number of repetitions

benchmark encryption

```
$ kopia benchmark encryption
```

Run encryption benchmarks

Flag	Short	Default	Help
`--block-size`		`1MB`	Size of a block to encrypt
`--[no-]deprecated`		`false`	Include deprecated algorithms
`--parallel`		`1`	Number of parallel goroutines
`--[no-]print-options`		`false`	Print out options usable for repository creation
`--repeat`		`1000`	Number of repetitions

benchmark ecc

```
$ kopia benchmark ecc
```

Run ECC benchmarks

Flag	Short	Default	Help
`--block-size`		`10MB`	Size of a block to encrypt
`--parallel`		`1`	Number of parallel goroutines
`--[no-]print-options`		`false`	Print out options usable for repository creation
`--repeat`		`100`	Number of repetitions

notification profile configure email

```
$ kopia notification profile configure email \
        --profile-name=...
```

E-mail notification.

Flag	Short	Default	Help
`--format`			Format of the message
`--mail-cc`			CC address
`--mail-from`			From address
`--mail-to`			To address
`--min-severity`			Minimum severity
`--profile-name`			Profile name
`--[no-]send-test-notification`		`false`	Test the notification
`--smtp-identity`			SMTP identity
`--smtp-password`			SMTP password
`--smtp-port`			SMTP port
`--smtp-server`			SMTP server
`--smtp-username`			SMTP username

notification profile configure pushover

```
$ kopia notification profile configure pushover \
        --profile-name=...
```

Pushover notification.

Flag	Short	Default	Help
`--app-token`			Pushover App Token
`--format`			Format of the message
`--min-severity`			Minimum severity
`--profile-name`			Profile name
`--[no-]send-test-notification`		`false`	Test the notification
`--user-key`			Pushover User Key

notification profile configure webhook

```
$ kopia notification profile configure webhook \
        --profile-name=...
```

Webhook notification.

Flag	Short	Default	Help
`--endpoint`			SMTP server
`--format`			Format of the message
`--method`			HTTP Method
`--min-severity`			Minimum severity
`--profile-name`			Profile name
`--[no-]send-test-notification`		`false`	Test the notification

notification profile delete

```
$ kopia notification profile delete \
        --profile-name=...
```

Delete notification profile

Flag	Short	Default	Help
`--profile-name`			Profile name

notification profile test

```
$ kopia notification profile test \
        --profile-name=...
```

Send test notification

Flag	Short	Default	Help
`--profile-name`			Profile name

notification profile list

```
$ kopia notification profile list
```

List notification profiles

Flag	Short	Default	Help
`--[no-]json`		`false`	Output result in JSON format to stdout
`--[no-]json-indent`		`false`	[ADV] Output result in indented JSON format to stdout
`--[no-]json-verbose`		`false`	[ADV] Output non-essential data (e.g. statistics) in JSON format

notification template list

```
$ kopia notification template list
```

List templates

Flag	Short	Default	Help
`--[no-]json`		`false`	Output result in JSON format to stdout
`--[no-]json-indent`		`false`	[ADV] Output result in indented JSON format to stdout
`--[no-]json-verbose`		`false`	[ADV] Output non-essential data (e.g. statistics) in JSON format

notification template set

```
$ kopia notification template set <template>
```

Set the notification template

Flag	Short	Default	Help
`--[no-]editor`		`false`	Edit template using default editor
`--from-file`			Read new template from file
`--[no-]from-stdin`		`false`	Read new template from stdin

Argument	Help
`template`	Template name

notification template show

```
$ kopia notification template show <template>
```

Show template

Flag	Short	Default	Help
`--format`			Template format
`--[no-]html`		`false`	Convert the output to HTML
`--[no-]original`		`false`	Show original template

Argument	Help
`template`	Template name

notification template remove

```
$ kopia notification template remove <template>
```

Remove the notification template

Argument	Help
`template`	Template name

server start

```
$ kopia server start
```

Start Kopia server

Flag	Short	Default	Help
`--address`		`http://127.0.0.1:51515`	Server address
`--cache-directory`			Cache directory
`--[no-]check-for-updates`		`true`	Periodically check for Kopia updates on GitHub
`--content-cache-size-limit-mb`			Maximum size of local content cache (hard limit)
`--content-cache-size-mb`			Desired size of local content cache (soft limit)
`--content-min-sweep-age`			Minimal age of content cache item to be subject to sweeping
`--[no-]control-api`		`true`	Start the control API
`--description`			Human-readable description of the repository
`--[no-]enable-actions`		`false`	Allow snapshot actions
`--[no-]grpc`		`true`	Start the GRPC server
`--html`			Server the provided HTML at the root URL
`--index-min-sweep-age`			Minimal age of index cache item to be subject to sweeping
`--max-concurrency`		`0`	Maximum number of server goroutines
`--max-list-cache-duration`			Duration of index cache
`--metadata-cache-size-limit-mb`			Maximum size of local metadata cache (hard limit)
`--metadata-cache-size-mb`			Desired size of local metadata cache (soft limit)
`--metadata-min-sweep-age`			Minimal age of metadata cache item to be subject to sweeping
`--[no-]persistent-logs`		`true`	Persist logs in a file
`--[no-]readonly`		`false`	Make repository read-only to avoid accidental changes
`--refresh-interval`		`4h`	Frequency for refreshing repository status
`--server-control-password`			Server control password
`--server-control-username`		`server-control`	Server control username
`--server-password`			HTTP server password (basic auth)
`--server-username`		`kopia`	HTTP server username (basic auth)
`--shutdown-grace-period`		`5s`	Grace period for shutting down the server
`--tls-cert-file`			TLS certificate PEM
`--tls-key-file`			TLS key PEM file
`--[no-]ui`		`true`	Start the server with HTML UI
`--ui-preferences-file`			Path to JSON file storing UI preferences
`--[no-]async-repo-connect`		`false`	[ADV] Connect to repository asynchronously
`--auth-cookie-signing-key`			[ADV] Force particular auth cookie signing key
`--[no-]disable-csrf-token-checks`		`false`	[ADV] Disable CSRF token
`--[no-]disable-repository-format-cache`		`false`	[ADV] Disable caching of kopia.repository format blob
`--htpasswd-file`			[ADV] Path to htpasswd file that contains allowed user@hostname entries
`--[no-]insecure`		`false`	[ADV] Allow insecure configurations (do not use in production)
`--[no-]log-scheduler`		`true`	[ADV] Enable logging of scheduler actions
`--[no-]log-server-requests`		`false`	[ADV] Log server requests
`--min-maintenance-interval`		`60s`	[ADV] Minimum maintenance interval
`--override-hostname`			[ADV] Override hostname used by this repository connection
`--override-username`			[ADV] Override username used by this repository connection
`--[no-]permissive-cache-loading`		`false`	[ADV] Do not fail when loading bad cache index entries. Repository must be opened in read-only mode
`--[no-]random-password`		`false`	[ADV] Generate random password and print to stderr
`--[no-]random-server-control-password`		`false`	[ADV] Generate random server control password and print to stderr

Flag	Short	Default	Help
`--repository-format-cache-duration`			[ADV] Duration of kopia.repository format blob cache
`--[no-]shutdown-on-stdin`		`false`	[ADV] Shut down the server when stdin handle has closed.
`--[no-]tls-generate-cert`		`false`	[ADV] Generate TLS certificate
`--tls-generate-cert-name`		`127.0.0.1`	[ADV] Host names/IP addresses to generate TLS certificate for
`--tls-generate-cert-valid-days`		`3650`	[ADV] How long should the TLS certificate be valid
`--tls-generate-rsa-key-size`		`4096`	[ADV] TLS RSA Key size (bits)
`--[no-]tls-print-server-cert`		`false`	[ADV] Print server certificate
`--ui-title-prefix`			[ADV] UI title prefix
`--[no-]without-password`		`false`	[ADV] Start the server without a password

server acl add

```
$ kopia server acl add \
        --user=... \
        --target=... \
        --access=...
```

Add ACL entry

Flag	Short	Default	Help
`--access`			Access the user gets to subject
`--[no-]overwrite`		`false`	Overwrite existing rule with the same user and target
`--target`			Manifests targeted by the rule (type:T,key1:value1,…,keyN:valueN)
`--user`			User the ACL targets

server acl delete

```
$ kopia server acl delete [id]
```

Delete ACL entry

Flag	Short	Default	Help
`--[no-]all`		`false`	Remove all ACL entries
`--[no-]delete`		`false`	Really delete

Argument	Help
`id`	Entry ID

server acl enable

```
$ kopia server acl enable
```

Enable ACLs and install default entries

Flag	Short	Default	Help
`--[no-]reset`		`false`	Reset all ACLs to default

server acl list

```
$ kopia server acl list
```

List ACL entries

Flag	Short	Default	Help
`--[no-]json`		`false`	Output result in JSON format to stdout
`--[no-]json-indent`		`false`	[ADV] Output result in indented JSON format to stdout
`--[no-]json-verbose`		`false`	[ADV] Output non-essential data (e.g. statistics) in JSON format

server users add

```
$ kopia server users add <username>
```

Add new repository user

Flag	Short	Default	Help
`--[no-]ask-password`		`false`	Ask for user password
`--user-password`			Password
`--user-password-hash`			Password hash

Argument	Help
`username`	Username

server users set

```
$ kopia server users set <username>
```

Set password for a repository user.

Flag	Short	Default	Help
`--[no-]ask-password`		`false`	Ask for user password
`--user-password`			Password
`--user-password-hash`			Password hash

Argument	Help
`username`	Username

server users delete

```
$ kopia server users delete <username>
```

Delete user

Argument	Help
`username`	The username to delete.

server users hash-password

```
$ kopia server users hash-password
```

Hash a user password that can be passed to the ‘server user add/set’ command

Flag	Short	Default	Help
`--user-password`			Password

server users info

```
$ kopia server users info <username>
```

Info about particular user

Argument	Help
`username`	The username to look up.

server users list

```
$ kopia server users list
```

List users

Flag	Short	Default	Help
`--[no-]json`		`false`	Output result in JSON format to stdout
`--[no-]json-indent`		`false`	[ADV] Output result in indented JSON format to stdout
`--[no-]json-verbose`		`false`	[ADV] Output non-essential data (e.g. statistics) in JSON format

server status

```
$ kopia server status
```

Status of Kopia server

Flag	Short	Default	Help
`--address`		`http://127.0.0.1:51515`	Address of the server to connect to
`--[no-]remote`		`false`	Show remote sources
`--server-cert-fingerprint`			Server certificate fingerprint
`--server-control-password`			Server control password
`--server-control-username`			Server control username
`--server-password`			[ADV] Server control password
`--server-username`			[ADV] Server control username

server refresh

```
$ kopia server refresh
```

Refresh the cache in Kopia server to observe new sources, etc.

Flag	Short	Default	Help
`--address`		`http://127.0.0.1:51515`	Address of the server to connect to
`--server-cert-fingerprint`			Server certificate fingerprint
`--server-control-password`			Server control password
`--server-control-username`			Server control username
`--server-password`			[ADV] Server control password
`--server-username`			[ADV] Server control username

server flush

```
$ kopia server flush
```

Flush the state of Kopia server to persistent storage, etc.

Flag	Short	Default	Help
`--address`		`http://127.0.0.1:51515`	Address of the server to connect to
`--server-cert-fingerprint`			Server certificate fingerprint
`--server-control-password`			Server control password
`--server-control-username`			Server control username
`--server-password`			[ADV] Server control password
`--server-username`			[ADV] Server control username

server shutdown

```
$ kopia server shutdown
```

Gracefully shutdown the server

Flag	Short	Default	Help
`--address`		`http://127.0.0.1:51515`	Address of the server to connect to
`--server-cert-fingerprint`			Server certificate fingerprint
`--server-control-password`			Server control password
`--server-control-username`			Server control username
`--server-password`			[ADV] Server control password
`--server-username`			[ADV] Server control username

server snapshot

```
$ kopia server snapshot [source]
```

Trigger upload for one or more existing sources

Flag	Short	Default	Help
`--address`		`http://127.0.0.1:51515`	Address of the server to connect to
`--[no-]all`		`false`	All paths managed by server
`--server-cert-fingerprint`			Server certificate fingerprint
`--server-control-password`			Server control password
`--server-control-username`			Server control username
`--server-password`			[ADV] Server control password
`--server-username`			[ADV] Server control username

Argument	Help
`source`	Source path managed by server

server cancel

```
$ kopia server cancel [source]
```

Cancels in-progress uploads for one or more sources

Flag	Short	Default	Help
`--address`		`http://127.0.0.1:51515`	Address of the server to connect to
`--[no-]all`		`false`	All paths managed by server
`--server-cert-fingerprint`			Server certificate fingerprint
`--server-control-password`			Server control password
`--server-control-username`			Server control username
`--server-password`			[ADV] Server control password
`--server-username`			[ADV] Server control username

Argument	Help
`source`	Source path managed by server

server pause

```
$ kopia server pause [source]
```

Pause the scheduled snapshots for one or more sources

Flag	Short	Default	Help
`--address`		`http://127.0.0.1:51515`	Address of the server to connect to
`--[no-]all`		`false`	All paths managed by server
`--server-cert-fingerprint`			Server certificate fingerprint
`--server-control-password`			Server control password
`--server-control-username`			Server control username
`--server-password`			[ADV] Server control password
`--server-username`			[ADV] Server control username

Argument	Help
`source`	Source path managed by server

server resume

```
$ kopia server resume [source]
```

Resume the scheduled snapshots for one or more sources

Flag	Short	Default	Help
`--address`		`http://127.0.0.1:51515`	Address of the server to connect to
`--[no-]all`		`false`	All paths managed by server
`--server-cert-fingerprint`			Server certificate fingerprint
`--server-control-password`			Server control password
`--server-control-username`			Server control username
`--server-password`			[ADV] Server control password
`--server-username`			[ADV] Server control username

Argument	Help
`source`	Source path managed by server

server throttle get

```
$ kopia server throttle get
```

Get throttling parameters for a running server

Flag	Short	Default	Help
`--address`		`http://127.0.0.1:51515`	Address of the server to connect to
`--[no-]json`		`false`	Output result in JSON format to stdout
`--server-cert-fingerprint`			Server certificate fingerprint
`--server-control-password`			Server control password
`--server-control-username`			Server control username
`--[no-]json-indent`		`false`	[ADV] Output result in indented JSON format to stdout
`--[no-]json-verbose`		`false`	[ADV] Output non-essential data (e.g. statistics) in JSON format
`--server-password`			[ADV] Server control password
`--server-username`			[ADV] Server control username

server throttle set

```
$ kopia server throttle set
```

Set throttling parameters for a running server

Flag	Short	Default	Help
`--address`		`http://127.0.0.1:51515`	Address of the server to connect to
`--concurrent-reads`			Set max concurrent reads
`--concurrent-writes`			Set max concurrent writes
`--download-bytes-per-second`			Set the download bytes per second
`--list-requests-per-second`			Set max lists per second
`--read-requests-per-second`			Set max reads per second
`--server-cert-fingerprint`			Server certificate fingerprint
`--server-control-password`			Server control password
`--server-control-username`			Server control username
`--upload-bytes-per-second`			Set the upload bytes per second
`--write-requests-per-second`			Set max writes per second
`--server-password`			[ADV] Server control password
`--server-username`			[ADV] Server control username

snapshot copy-history

```
$ kopia snapshot copy-history <source> [destination]
```

Performs a copy of the history of snapshots from another user or host. This command will copy snapshot manifests of the specified source to the respective destination. This is typically used when renaming a host, switching username or moving directory around to maintain snapshot history.

```
Both source and destination can be specified using user@host, @host or user@host:/path
where destination values override the corresponding parts of the source, so both targeted
and mass copy is supported.

Source:             Destination         Behavior
---------------------------------------------------
@host1              @host2              copy snapshots from all users of host1
@host1              user2@host2         (disallowed as it would potentially collapse users)
@host1              user2@host2:/path2  (disallowed as it would potentially collapse paths)
user1@host1         @host2              copy all snapshots to user1@host2
user1@host1         user2@host2         copy all snapshots to user2@host2
user1@host1         user2@host2:/path2  (disallowed as it would potentially collapse paths)
user1@host1:/path1  @host2              copy to user1@host2:/path1
user1@host1:/path1  user2@host2         copy to user2@host2:/path1
user1@host1:/path1  user2@host2:/path2  copy snapshots from single path.
```

Flag	Short	Default	Help
`--[no-]dry-run`	`-n`	`false`	Do not actually copy snapshots, only print what would happen

Argument	Help
`destination`	Destination (defaults to current user@host)
`source`	Source (user@host or user@host:path)

snapshot move-history

```
$ kopia snapshot move-history <source> [destination]
```

Performs a move of the history of snapshots from another user or host. This command will move snapshot manifests of the specified source to the respective destination. This is typically used when renaming a host, switching username or moving directory around to maintain snapshot history.

```
Both source and destination can be specified using user@host, @host or user@host:/path
where destination values override the corresponding parts of the source, so both targeted
and mass move is supported.

Source:             Destination         Behavior
---------------------------------------------------
@host1              @host2              move snapshots from all users of host1
@host1              user2@host2         (disallowed as it would potentially collapse users)
@host1              user2@host2:/path2  (disallowed as it would potentially collapse paths)
user1@host1         @host2              move all snapshots to user1@host2
user1@host1         user2@host2         move all snapshots to user2@host2
user1@host1         user2@host2:/path2  (disallowed as it would potentially collapse paths)
user1@host1:/path1  @host2              move to user1@host2:/path1
user1@host1:/path1  user2@host2         move to user2@host2:/path1
user1@host1:/path1  user2@host2:/path2  move snapshots from single path.
```

Flag	Short	Default	Help
`--[no-]dry-run`	`-n`	`false`	Do not actually copy snapshots, only print what would happen

Argument	Help
`destination`	Destination (defaults to current user@host)
`source`	Source (user@host or user@host:path)

snapshot create

```
$ kopia snapshot create [source]
```

Creates a snapshot of local directory or file.

Flag	Short	Default	Help
`--[no-]all`		`false`	Create snapshots for files or directories previously backed up by this user on this computer. Cannot be used when a source path argument is also specified.
`--description`			Free-form snapshot description.
`--end-time`			Override snapshot end timestamp.
`--[no-]fail-fast`		`false`	Fail fast when creating snapshot.
`--force-hash`		`0`	Force hashing of source files for a given percentage of files [0.0 .. 100.0]
`--[no-]json`		`false`	Output result in JSON format to stdout
`--log-dir-detail`			Override log level for directories
`--log-entry-detail`			Override log level for entries
`--override-source`			Override the source of the snapshot.
`--parallel`		`0`	Upload N files in parallel
`--pin`			Create a pinned snapshot that will not expire automatically
`--start-time`			Override snapshot start timestamp.
`--stdin-file`			File path to be used for stdin data snapshot.
`--tags`			Tags applied on the snapshot. Must be provided in the : format.
`--upload-limit-mb`		`0`	Stop the backup process after the specified amount of data (in MB) has been uploaded.
`--checkpoint-interval`			[ADV] Interval between periodic checkpoints (must be <= 45 minutes).
`--[no-]flush-per-source`		`false`	[ADV] Flush writes at the end of each source
`--[no-]force-disable-actions`		`false`	[ADV] Disable snapshot actions even if globally enabled on this client
`--[no-]force-enable-actions`		`false`	[ADV] Enable snapshot actions even if globally disabled on this client
`--[no-]json-indent`		`false`	[ADV] Output result in indented JSON format to stdout
`--[no-]json-verbose`		`false`	[ADV] Output non-essential data (e.g. statistics) in JSON format

Argument	Help
`source`	Files or directories to create snapshot(s) of.

snapshot delete

```
$ kopia snapshot delete <id>
```

Explicitly delete a snapshot by providing a snapshot ID.

Flag	Short	Default	Help
`--[no-]all-snapshots-for-source`		`false`	Delete all snapshots for a source
`--[no-]delete`		`false`	Confirm deletion
`--[no-]unsafe-ignore-source`		`false`	[ADV] Alias for –delete

Argument	Help
`id`	Snapshot ID or root object ID to be deleted

snapshot estimate

```
$ kopia snapshot estimate <source>
```

Estimate the snapshot size and upload time.

Flag	Short	Default	Help
`--max-examples-per-bucket`		`10`	Max examples per bucket
`--[no-]quiet`	`-q`	`false`	Do not display scanning progress
`--[no-]show-files`		`false`	Show files
`--upload-speed`		`10`	Upload speed to use for estimation

Argument	Help
`source`	File or directory to analyze.

snapshot expire

```
$ kopia snapshot expire [path]
```

Remove old snapshots according to defined expiration policies.

Flag	Short	Default	Help
`--[no-]all`		`false`	Expire all snapshots
`--[no-]delete`		`false`	Whether to actually delete snapshots

Argument	Help
`path`	Expire snapshots for given paths only

snapshot fix invalid-files

```
$ kopia snapshot fix invalid-files
```

Remove references to any invalid (unreadable) files from snapshots.

Flag	Short	Default	Help
`--[no-]commit`		`false`	Update snapshot manifests
`--invalid-directory-handling`		`stub`	Handling of invalid directories
`--invalid-file-handling`		`stub`	How to handle invalid files
`--manifest-id`			Manifest IDs
`--parallel`			Parallelism
`--source`			Source to target (username@hostname:/path)
`--verify-files-percent`		`0`	Verify a percentage of files by fully downloading them [0.0 .. 100.0]

snapshot fix remove-files

```
$ kopia snapshot fix remove-files
```

Remove references to the specified files from snapshots.

Flag	Short	Default	Help
`--[no-]commit`		`false`	Update snapshot manifests
`--filename`			Remove files by filename (wildcards are supported)
`--invalid-directory-handling`		`stub`	Handling of invalid directories
`--manifest-id`			Manifest IDs
`--object-id`			Remove files by their object ID
`--parallel`			Parallelism
`--source`			Source to target (username@hostname:/path)

snapshot list

```
$ kopia snapshot list [source]
```

List snapshots of files and directories.

Flag	Short	Default	Help
`--[no-]all`	`-a`	`false`	Show all snapshots (not just current username/host)
`--[no-]delta`	`-d`	`false`	Include deltas.
`--[no-]human-readable`		`true`	Show human-readable units
`--[no-]incomplete`	`-i`	`false`	Include incomplete.
`--[no-]json`		`false`	Output result in JSON format to stdout
`--[no-]manifest-id`	`-m`	`false`	Include manifest item ID.
`--max-results`	`-n`		Maximum number of entries per source.
`--[no-]mtime`		`false`	Include file mod time
`--[no-]owner`		`false`	Include owner
`--[no-]retention`		`true`	Include retention reasons.
`--[no-]reverse`		`false`	Reverse sort order
`--[no-]show-identical`	`-l`	`false`	Show identical snapshots
`--[no-]storage-stats`		`false`	Compute and show storage statistics
`--tags`			Tag filters to apply on the list items. Must be provided in the : format.
`--[no-]json-indent`		`false`	[ADV] Output result in indented JSON format to stdout
`--[no-]json-verbose`		`false`	[ADV] Output non-essential data (e.g. statistics) in JSON format

Argument	Help
`source`	File or directory to show history of.

snapshot migrate

```
$ kopia snapshot migrate \
        --source-config=...
```

Migrate snapshots from another repository

Flag	Short	Default	Help
`--[no-]all`		`false`	Migrate all sources
`--[no-]apply-ignore-rules`		`false`	When migrating also apply current ignore rules
`--[no-]latest-only`		`false`	Only migrate the latest snapshot
`--[no-]overwrite-policies`		`false`	Overwrite policies
`--parallel`		`1`	Number of sources to migrate in parallel
`--[no-]policies`		`true`	Migrate policies too
`--source-config`			Configuration file for the source repository
`--sources`			List of sources to migrate

snapshot pin

```
$ kopia snapshot pin <id>
```

Add or remove pins preventing snapshot deletion

Flag	Short	Default	Help
`--add`			Add pins
`--remove`			Remove pins

Argument	Help
`id`	Snapshot ID or root object ID

snapshot restore

```
$ kopia snapshot restore <sources>
```

Restore a directory or a file.

Restore can operate in two modes:

- from a snapshot: restoring (possibly shallowly) a specified file or directory from a snapshot into a target path. By default, the target path will be created by the restore command if it does not exist.
- by expanding a shallow placeholder in situ where the placeholder was created by a previous restore.

In the from-snapshot mode:

The source to be restored is specified in the form of a directory or file ID and optionally a sub-directory path.

For example, the following source and target arguments will restore the contents of the 'kffbb7c28ea6c34d6cbe555d1cf80faa9' directory into a new, local directory named 'd1'

'restore kffbb7c28ea6c34d6cbe555d1cf80faa9 d1'

Similarly, the following command will restore the contents of a subdirectory 'subdir/subdir2' under 'kffbb7c28ea6c34d6cbe555d1cf80faa9' into a new, local directory named 'sd2'

'restore kffbb7c28ea6c34d6cbe555d1cf80faa9/subdir1/subdir2 sd2'

When restoring to a target path that already has existing data, by default the restore will attempt to overwrite, unless one or more of the following flags has been set (to prevent overwrite of each type):

–no-overwrite-files –no-overwrite-directories –no-overwrite-symlinks

If the '–shallow' option is provided, files and directories this depth and below in the directory hierarchy will be represented by compact placeholder files of the form 'entry.kopia-entry' instead of being restored. (I.e. setting '–shallow' to 0 will only shallow restore.) Snapshots created of directory contents represented by placeholder files will be identical to snapshots of the equivalent fully expanded tree.

In the expanding-a-placeholder mode:

The source to be restored is a pre-existing placeholder entry of the form 'entry.kopia-entry'. The target will be 'entry'. '–shallow' controls the depth of the expansion and defaults to 0. For example:

'restore d3.kopiadir'

will remove the d3.kopiadir placeholder and restore the referenced repository contents into path d3 where the contents of the newly created path d3 will themselves be placeholder files.

Flag	Short	Default	Help
`--[no-]consistent-attributes`		`false`	When multiple snapshots match, fail if they have inconsistent attributes
`--[no-]ignore-errors`		`false`	Ignore all errors
`--[no-]ignore-permission-errors`		`true`	Ignore permission errors
`--mode`		`auto`	Override restore mode
`--[no-]overwrite-directories`		`true`	Overwrite existing directories
`--[no-]overwrite-files`		`true`	Specifies whether or not to overwrite already existing files
`--[no-]overwrite-symlinks`		`true`	Specifies whether or not to overwrite already existing symlinks
`--parallel`		`8`	Restore parallelism (1=disable)
`--shallow`			Shallow restore the directory hierarchy starting at this level (default is to deep restore the entire hierarchy.)
`--shallow-minsize`			When doing a shallow restore, write actual files instead of placeholders smaller than this size.
`--[no-]skip-existing`		`false`	Skip files and symlinks that exist in the output
`--[no-]skip-owners`		`false`	Skip owners during restore
`--[no-]skip-permissions`		`false`	Skip permissions during restore
`--[no-]skip-times`		`false`	Skip times during restore

Flag	Short	Default	Help
`--snapshot-time`		`latest`	When using a path as the source, use the latest snapshot available before this date. Default is latest
`--[no-]write-files-atomically`		`false`	Write files atomically to disk, ensuring they are either fully committed, or not written at all, preventing partially written files
`--[no-]write-sparse-files`		`false`	When doing a restore, attempt to write files sparsely-allocating the minimum amount of disk space needed.

Argument	Help
`sources`	Two forms: 1. Source directory ID/path in the form of a

directory ID and optionally a sub-directory path. For example,
'kffbb7c28ea6c34d6cbe555d1cf80faa9' or
'kffbb7c28ea6c34d6cbe555d1cf80faa9/subdir1/subdir2'
followed by the path of the directory for the contents to be restored.

2. one or more placeholder files of the form path.kopia-entry |

snapshot verify

```
$ kopia snapshot verify [snapshot-ids]
```

Verify the contents of stored snapshot

Flag	Short	Default	Help
`--directory-id`			Directory object IDs to verify
`--file-id`			File object IDs to verify
`--file-parallelism`			Parallelism for file verification
`--file-queue-length`		`20000`	Queue length for file verification
`--max-errors`		`0`	Maximum number of errors before stopping
`--parallel`		`8`	Parallelization
`--sources`			Verify the provided sources
`--verify-files-percent`		`0`	Randomly verify a percentage of files by downloading them [0.0 .. 100.0]
`--[no-]all-sources`		`false`	[ADV] Verify all snapshots (DEPRECATED)

Argument	Help
`snapshot-ids`	snapshot IDs to verify

policy edit

```
$ kopia policy edit [target]
```

Set snapshot policy for a single directory, user@host or a global policy.

Flag	Short	Default	Help
`--[no-]global`		`false`	Select the global policy.

Argument	Help
`target`	Select a particular policy (‘user@host’,’@host’,‘user@host:path’ or a local path). Use –global to target the global policy.

policy list

```
$ kopia policy list
```

List policies.

Flag	Short	Default	Help
`--[no-]json`		`false`	Output result in JSON format to stdout
`--[no-]json-indent`		`false`	[ADV] Output result in indented JSON format to stdout
`--[no-]json-verbose`		`false`	[ADV] Output non-essential data (e.g. statistics) in JSON format

policy delete

```
$ kopia policy delete [target]
```

Remove snapshot policy for a single directory, user@host or a global policy.

Flag	Short	Default	Help
`--[no-]dry-run`	`-n`	`false`	Do not remove
`--[no-]global`		`false`	Select the global policy.

Argument	Help
`target`	Select a particular policy (‘user@host’,’@host’,‘user@host:path’ or a local path). Use –global to target the global policy.

policy set

```
$ kopia policy set [target]
```

Set snapshot policy for a single directory, user@host or a global policy.

Flag	Short	Default	Help
`--action-command-mode`		`essential`	Action command mode
`--action-command-timeout`		`5m`	Max time allowed for an action to run in seconds
`--add-dot-ignore`			List of paths to add to the dot-ignore list
`--add-ignore`			List of paths to add to the ignore list
`--add-never-compress`			List of extensions to add to the never compress list
`--add-only-compress`			List of extensions to add to the only-compress list
`--after-folder-action`		`-`	Path to after-folder action command ('none' to remove)
`--after-snapshot-root-action`		`-`	Path to after-snapshot-root action command ('none' to remove or 'inherit')
`--before-folder-action`		`-`	Path to before-folder action command ('none' to remove)
`--before-snapshot-root-action`		`-`	Path to before-snapshot-root action command ('none' to remove or 'inherit')
`--[no-]clear-dot-ignore`		`false`	Clear list of paths in the dot-ignore list
`--[no-]clear-ignore`		`false`	Clear list of paths in the ignore list
`--[no-]clear-never-compress`		`false`	Clear list of extensions in the never compress list
`--[no-]clear-only-compress`		`false`	Clear list of extensions in the only-compress list
`--compression`			Compression algorithm
`--compression-max-size`			Max size of file to attempt compression for
`--compression-min-size`			Min size of file to attempt compression for
`--enable-volume-shadow-copy`			Enable Volume Shadow Copy snapshots ('never', 'always', 'when-available', 'inherit')
`--[no-]global`		`false`	Select the global policy.
`--ignore-cache-dirs`			Ignore cache directories ('true', 'false', 'inherit')
`--ignore-dir-errors`			Ignore errors reading directories while traversing ('true', 'false', 'inherit
`--ignore-file-errors`			Ignore errors reading files while traversing ('true', 'false', 'inherit')
`--ignore-identical-snapshots`			Do not save identical snapshots (or 'inherit')
`--ignore-unknown-types`			Ignore unknown entry types in directories ('true', 'false', 'inherit
`--inherit`			Enable or disable inheriting policies from the parent
`--keep-annual`			Number of most-recent annual backups to keep per source (or 'inherit')
`--keep-daily`			Number of most-recent daily backups to keep per source (or 'inherit')
`--keep-hourly`			Number of most-recent hourly backups to keep per source (or 'inherit')
`--keep-latest`			Number of most recent backups to keep per source (or 'inherit')
`--keep-monthly`			Number of most-recent monthly backups to keep per source (or 'inherit')
`--keep-weekly`			Number of most-recent weekly backups to keep per source (or 'inherit')
`--log-dir-ignored`			Log detail when a directory is ignored (or 'inherit')
`--log-dir-snapshotted`			Log detail when a directory is snapshotted (or 'inherit')
`--log-entry-cache-hit`			Log detail on entry cache hit (or 'inherit')
`--log-entry-cache-miss`			Log detail on entry cache miss (or 'inherit')
`--log-entry-ignored`			Log detail when an entry is ignored (or 'inherit')
`--log-entry-snapshotted`			Log detail when an entry is snapshotted (or 'inherit')
`--[no-]manual`		`false`	Only create snapshots manually
`--max-file-size`			Exclude files above given size
`--max-parallel-file-reads`			Maximum number of parallel file reads
`--max-parallel-snapshots`			Maximum number of parallel snapshots (server, KopiaUI only)
`--metadata-compression`			Metadata Compression algorithm
`--one-file-system`			Stay in parent filesystem when finding files ('true', 'false', 'inherit')
`--parallel-upload-above-size-mib`			Use parallel uploads above size
`--[no-]persist-action-script`		`false`	Persist action script

Flag	Short	Default	Help
`--remove-dot-ignore`			List of paths to remove from the dot-ignore list
`--remove-ignore`			List of paths to remove from the ignore list
`--remove-never-compress`			List of extensions to remove from the never compress list
`--remove-only-compress`			List of extensions to remove from the only-compress list
`--run-missed`			Run missed time-of-day or cron snapshots ('true', 'false', 'inherit')
`--snapshot-interval`			Interval between snapshots
`--snapshot-time`			Comma-separated times of day when to take snapshot (HH:mm,HH:mm,…) or 'inherit' to remove override
`--snapshot-time-crontab`			Semicolon-separated crontab-compatible expressions (or 'inherit')
`--splitter`			Splitter algorithm override

Argument	Help
`target`	Select a particular policy ('user@host','@host','user@host:path' or a local path). Use –global to target the global policy.

policy show

```
$ kopia policy show [target]
```

Show snapshot policy.

Flag	Short	Default	Help
`--[no-]global`		`false`	Select the global policy.
`--[no-]json`		`false`	Output result in JSON format to stdout
`--[no-]json-indent`		`false`	[ADV] Output result in indented JSON format to stdout
`--[no-]json-verbose`		`false`	[ADV] Output non-essential data (e.g. statistics) in JSON format

Argument	Help
`target`	Select a particular policy ('user@host','@host','user@host:path' or a local path). Use –global to target the global policy.

policy export

```
$ kopia policy export [target]
```

Exports the policy to the specified file, or to stdout if none is specified.

Flag	Short	Default	Help
`--[no-]global`		`false`	Select the global policy.
`--[no-]overwrite`		`false`	Overwrite the file if it exists
`--to-file`			File path to export to
`--[no-]json-indent`		`false`	[ADV] Output result in indented JSON format

Argument	Help
`target`	Select a particular policy (‘user@host’,’@host’,‘user@host:path’ or a local path). Use –global to target the global policy.

policy import

```
$ kopia policy import [target]
```

Imports policies from a specified file, or stdin if no file is specified.

Flag	Short	Default	Help
`--[no-]allow-unknown-fields`		`false`	Allow unknown fields in the policy file
`--[no-]delete-other-policies`		`false`	Delete all other policies, keeping only those that got imported
`--from-file`			File path to import from
`--[no-]global`		`false`	Select the global policy.

Argument	Help
`target`	Select a particular policy ('user@host','@host','user@host:path' or a local path). Use –global to target the global policy.

repository connect s3

```
$ kopia repository connect s3 \
        --bucket=... \
        --access-key=... \
        --secret-access-key=...
```

Connect to repository in an S3 bucket

Flag	Short	Default	Help
`--access-key`			Access key ID (overrides AWS_ACCESS_KEY_ID environment variable)
`--bucket`			Name of the S3 bucket
`--cache-directory`			Cache directory
`--[no-]check-for-updates`		`true`	Periodically check for Kopia updates on GitHub
`--content-cache-size-limit-mb`			Maximum size of local content cache (hard limit)
`--content-cache-size-mb`			Desired size of local content cache (soft limit)
`--content-min-sweep-age`			Minimal age of content cache item to be subject to sweeping
`--description`			Human-readable description of the repository
`--[no-]disable-tls`		`false`	Disable TLS security (HTTPS)
`--[no-]disable-tls-verification`		`false`	Disable TLS (HTTPS) certificate verification
`--[no-]enable-actions`		`false`	Allow snapshot actions
`--endpoint`		`s3.amazonaws.com`	Endpoint to use
`--index-min-sweep-age`			Minimal age of index cache item to be subject to sweeping
`--max-download-speed`			Limit the download speed.
`--max-list-cache-duration`			Duration of index cache
`--max-upload-speed`			Limit the upload speed.
`--metadata-cache-size-limit-mb`			Maximum size of local metadata cache (hard limit)
`--metadata-cache-size-mb`			Desired size of local metadata cache (soft limit)
`--metadata-min-sweep-age`			Minimal age of metadata cache item to be subject to sweeping
`--point-in-time`			Use a point-in-time view of the storage repository when supported
`--prefix`			Prefix to use for objects in the bucket. Put trailing slash (/) if you want to use prefix as directory. e.g my-backup-dir/ would put repository contents inside my-backup-dir directory
`--[no-]readonly`		`false`	Make repository read-only to avoid accidental changes
`--region`			S3 Region
`--root-ca-pem-base64`			Certificate authority in-line (base64 enc.)
`--root-ca-pem-path`			Certificate authority file path
`--secret-access-key`			Secret access key (overrides AWS_SECRET_ACCESS_KEY environment variable)
`--session-token`			Session token (overrides AWS_SESSION_TOKEN environment variable)
`--[no-]disable-repository-format-cache`		`false`	[ADV] Disable caching of kopia.repository format blob
`--override-hostname`			[ADV] Override hostname used by this repository connection
`--override-username`			[ADV] Override username used by this repository connection
`--[no-]permissive-cache-loading`		`false`	[ADV] Do not fail when loading bad cache index entries. Repository must be opened in read-only mode
`--repository-format-cache-duration`			[ADV] Duration of kopia.repository format blob cache

repository connect from-config

```
$ kopia repository connect from-config
```

Connect to repository in the provided configuration file

Flag	Short	Default	Help
`--cache-directory`			Cache directory
`--[no-]check-for-updates`		`true`	Periodically check for Kopia updates on GitHub
`--content-cache-size-limit-mb`			Maximum size of local content cache (hard limit)
`--content-cache-size-mb`			Desired size of local content cache (soft limit)
`--content-min-sweep-age`			Minimal age of content cache item to be subject to sweeping
`--description`			Human-readable description of the repository
`--[no-]enable-actions`		`false`	Allow snapshot actions
`--file`			Path to the configuration file
`--index-min-sweep-age`			Minimal age of index cache item to be subject to sweeping
`--max-list-cache-duration`			Duration of index cache
`--metadata-cache-size-limit-mb`			Maximum size of local metadata cache (hard limit)
`--metadata-cache-size-mb`			Desired size of local metadata cache (soft limit)
`--metadata-min-sweep-age`			Minimal age of metadata cache item to be subject to sweeping
`--[no-]readonly`		`false`	Make repository read-only to avoid accidental changes
`--token`			Configuration token
`--token-file`			Path to the configuration token file
`--[no-]token-stdin`		`false`	Read configuration token from stdin
`--[no-]disable-repository-format-cache`		`false`	[ADV] Disable caching of kopia.repository format blob
`--override-hostname`			[ADV] Override hostname used by this repository connection
`--override-username`			[ADV] Override username used by this repository connection
`--[no-]permissive-cache-loading`		`false`	[ADV] Do not fail when loading bad cache index entries. Repository must be opened in read-only mode
`--repository-format-cache-duration`			[ADV] Duration of kopia.repository format blob cache

repository create s3

```
$ kopia repository create s3 \
        --bucket=... \
        --access-key=... \
        --secret-access-key=...
```

Create repository in an S3 bucket

Flag	Short	Default	Help
`--access-key`			Access key ID (overrides AWS_ACCESS_KEY_ID environment variable)
`--block-hash`		`BLAKE2B-256-128`	Content hash algorithm.
`--bucket`			Name of the S3 bucket
`--cache-directory`			Cache directory
`--[no-]check-for-updates`		`true`	Periodically check for Kopia updates on GitHub
`--content-cache-size-limit-mb`			Maximum size of local content cache (hard limit)
`--content-cache-size-mb`			Desired size of local content cache (soft limit)
`--content-min-sweep-age`			Minimal age of content cache item to be subject to sweeping
`--[no-]create-only`	`-c`	`false`	Create repository, but don't connect to it.
`--description`			Human-readable description of the repository
`--[no-]disable-tls`		`false`	Disable TLS security (HTTPS)
`--[no-]disable-tls-verification`		`false`	Disable TLS (HTTPS) certificate verification
`--ecc`		`REED-SOLOMON-CRC32`	[EXPERIMENTAL] Error correction algorithm.
`--ecc-overhead-percent`		`0`	[EXPERIMENTAL] How much space overhead can be used for error correction, in percentage. Use 0 to disable ECC.
`--[no-]enable-actions`		`false`	Allow snapshot actions
`--encryption`		`AES256-GCM-HMAC-SHA256`	Content encryption algorithm.
`--endpoint`		`s3.amazonaws.com`	Endpoint to use
`--format-block-key-derivation-algorithm`		`scrypt-65536-8-1`	Algorithm to derive the encryption key for the format block from the repository password
`--format-version`			Force a particular repository format version (1, 2 or 3, 0==default)
`--index-min-sweep-age`			Minimal age of index cache item to be subject to sweeping
`--max-download-speed`			Limit the download speed.
`--max-list-cache-duration`			Duration of index cache
`--max-upload-speed`			Limit the upload speed.
`--metadata-cache-size-limit-mb`			Maximum size of local metadata cache (hard limit)
`--metadata-cache-size-mb`			Desired size of local metadata cache (soft limit)
`--metadata-min-sweep-age`			Minimal age of metadata cache item to be subject to sweeping
`--object-splitter`		`DYNAMIC-4M-BUZHASH`	The splitter to use for new objects in the repository
`--point-in-time`			Use a point-in-time view of the storage repository when supported
`--prefix`			Prefix to use for objects in the bucket. Put trailing slash (/) if you want to use prefix as directory. e.g my-backup-dir/ would put repository contents inside my-backup-dir directory
`--[no-]readonly`		`false`	Make repository read-only to avoid accidental changes
`--region`			S3 Region
`--retention-mode`			Set the blob retention-mode for supported storage backends.
`--retention-period`			Set the blob retention-period for supported storage backends.
`--root-ca-pem-base64`			Certificate authority in-line (base64 enc.)
`--root-ca-pem-path`			Certificate authority file path
`--secret-access-key`			Secret access key (overrides AWS_SECRET_ACCESS_KEY environment variable)

Flag	Short	Default	Help
`--session-token`			Session token (overrides AWS_SESSION_TOKEN environment variable)
`--[no-]disable-repository-format-cache`		`false`	[ADV] Disable caching of kopia.repository format blob
`--override-hostname`			[ADV] Override hostname used by this repository connection
`--override-username`			[ADV] Override username used by this repository connection
`--[no-]permissive-cache-loading`		`false`	[ADV] Do not fail when loading bad cache index entries. Repository must be opened in read-only mode
`--repository-format-cache-duration`			[ADV] Duration of kopia.repository format blob cache

repository create from-config

```
$ kopia repository create from-config
```

Create repository in the provided configuration file

Flag	Short	Default	Help
`--block-hash`		`BLAKE2B-256-128`	Content hash algorithm.
`--cache-directory`			Cache directory
`--[no-]check-for-updates`		`true`	Periodically check for Kopia updates on GitHub
`--content-cache-size-limit-mb`			Maximum size of local content cache (hard limit)
`--content-cache-size-mb`			Desired size of local content cache (soft limit)
`--content-min-sweep-age`			Minimal age of content cache item to be subject to sweeping
`--[no-]create-only`	`-c`	`false`	Create repository, but don't connect to it.
`--description`			Human-readable description of the repository
`--ecc`		`REED-SOLOMON-CRC32`	[EXPERIMENTAL] Error correction algorithm.
`--ecc-overhead-percent`		`0`	[EXPERIMENTAL] How much space overhead can be used for error correction, in percentage. Use 0 to disable ECC.
`--[no-]enable-actions`		`false`	Allow snapshot actions
`--encryption`		`AES256-GCM-HMAC-SHA256`	Content encryption algorithm.
`--file`			Path to the configuration file
`--format-block-key-derivation-algorithm`		`scrypt-65536-8-1`	Algorithm to derive the encryption key for the format block from the repository password
`--format-version`			Force a particular repository format version (1, 2 or 3, 0==default)
`--index-min-sweep-age`			Minimal age of index cache item to be subject to sweeping
`--max-list-cache-duration`			Duration of index cache
`--metadata-cache-size-limit-mb`			Maximum size of local metadata cache (hard limit)
`--metadata-cache-size-mb`			Desired size of local metadata cache (soft limit)
`--metadata-min-sweep-age`			Minimal age of metadata cache item to be subject to sweeping
`--object-splitter`		`DYNAMIC-4M-BUZHASH`	The splitter to use for new objects in the repository
`--[no-]readonly`		`false`	Make repository read-only to avoid accidental changes
`--retention-mode`			Set the blob retention-mode for supported storage backends.
`--retention-period`			Set the blob retention-period for supported storage backends.
`--token`			Configuration token
`--token-file`			Path to the configuration token file
`--[no-]token-stdin`		`false`	Read configuration token from stdin
`--[no-]disable-repository-format-cache`		`false`	[ADV] Disable caching of kopia.repository format blob
`--override-hostname`			[ADV] Override hostname used by this repository connection
`--override-username`			[ADV] Override username used by this repository connection
`--[no-]permissive-cache-loading`		`false`	[ADV] Do not fail when loading bad cache index entries. Repository must be opened in read-only mode
`--repository-format-cache-duration`			[ADV] Duration of kopia.repository format blob cache

repository disconnect

```
$ kopia repository disconnect
```

Disconnect from a repository.

repository repair s3

```
$ kopia repository repair s3 \
        --bucket=... \
        --access-key=... \
        --secret-access-key=...
```

Repair repository in an S3 bucket

Flag	Short	Default	Help
`--access-key`			Access key ID (overrides AWS_ACCESS_KEY_ID environment variable)
`--bucket`			Name of the S3 bucket
`--[no-]disable-tls`		`false`	Disable TLS security (HTTPS)
`--[no-]disable-tls-verification`		`false`	Disable TLS (HTTPS) certificate verification
`--[no-]dry-run`	`-n`	`false`	Do not modify repository
`--endpoint`		`s3.amazonaws.com`	Endpoint to use
`--max-download-speed`			Limit the download speed.
`--max-upload-speed`			Limit the upload speed.
`--point-in-time`			Use a point-in-time view of the storage repository when supported
`--prefix`			Prefix to use for objects in the bucket. Put trailing slash (/) if you want to use prefix as directory. e.g my-backup-dir/ would put repository contents inside my-backup-dir directory
`--recover-format`		`auto`	Recover format blob from a copy
`--recover-format-block-prefixes`			Prefixes of file names
`--region`			S3 Region
`--root-ca-pem-base64`			Certificate authority in-line (base64 enc.)
`--root-ca-pem-path`			Certificate authority file path
`--secret-access-key`			Secret access key (overrides AWS_SECRET_ACCESS_KEY environment variable)
`--session-token`			Session token (overrides AWS_SESSION_TOKEN environment variable)

repository repair from-config

```
$ kopia repository repair from-config
```

Repair repository in the provided configuration file

Flag	Short	Default	Help
`--[no-]dry-run`	`-n`	`false`	Do not modify repository
`--file`			Path to the configuration file
`--recover-format`		`auto`	Recover format blob from a copy
`--recover-format-block-prefixes`			Prefixes of file names
`--token`			Configuration token
`--token-file`			Path to the configuration token file
`--[no-]token-stdin`		`false`	Read configuration token from stdin

repository set-client

```
$ kopia repository set-client
```

Set repository client options.

Flag	Short	Default	Help
`--description`			Change description
`--[no-]disable-repository-format-cache`		`false`	Disable caching of kopia.repository format blob
`--hostname`			Change hostname
`--[no-]read-only`		`false`	Set repository to read-only
`--[no-]read-write`		`false`	Set repository to read-write
`--repository-format-cache-duration`			Duration of kopia.repository format blob cache
`--username`			Change username
`--[no-]permissive-cache-loading`		`false`	[ADV] Do not fail when loading bad cache index entries. Repository must be opened in read-only mode

repository set-parameters

```
$ kopia repository set-parameters
```

Set repository parameters.

Flag	Short	Default	Help
`--epoch-advance-on-count`			Advance epoch if the number of indexes exceeds given threshold
`--epoch-advance-on-size-mb`			Advance epoch if the total size of indexes exceeds given threshold
`--epoch-checkpoint-frequency`			Checkpoint frequency
`--epoch-cleanup-safety-margin`			Epoch cleanup safety margin
`--epoch-delete-parallelism`			Epoch delete parallelism
`--epoch-min-duration`			Minimal duration of a single epoch
`--epoch-refresh-frequency`			Epoch refresh frequency
`--index-version`			Set version of index format used for writing
`--max-pack-size-mb`			Set max pack file size
`--retention-mode`			Set the blob retention-mode for supported storage backends.
`--retention-period`			Set the blob retention-period for supported storage backends.
`--[no-]upgrade`		`false`	Upgrade repository to the latest stable format

repository status

```
$ kopia repository status
```

Display the status of connected repository.

Flag	Short	Default	Help
`--[no-]json`		`false`	Output result in JSON format to stdout
`--[no-]reconnect-token`	`-t`	`false`	Display reconnect command
`--[no-]reconnect-token-with-password`	`-s`	`false`	Include password in reconnect token
`--[no-]json-indent`		`false`	[ADV] Output result in indented JSON format to stdout
`--[no-]json-verbose`		`false`	[ADV] Output non-essential data (e.g. statistics) in JSON format

repository sync-to s3

```
$ kopia repository sync-to s3 \
        --bucket=... \
        --access-key=... \
        --secret-access-key=...
```

Synchronize repository data to another repository in an S3 bucket

Flag	Short	Default	Help
`--access-key`			Access key ID (overrides AWS_ACCESS_KEY_ID environment variable)
`--bucket`			Name of the S3 bucket
`--[no-]delete`		`false`	Whether to delete blobs present in destination but not source.
`--[no-]disable-tls`		`false`	Disable TLS security (HTTPS)
`--[no-]disable-tls-verification`		`false`	Disable TLS (HTTPS) certificate verification
`--[no-]dry-run`	`-n`	`false`	Do not perform copying.
`--endpoint`		`s3.amazonaws.com`	Endpoint to use
`--max-download-speed`			Limit the download speed.
`--max-upload-speed`			Limit the upload speed.
`--[no-]must-exist`		`false`	Fail if destination does not have repository format blob.
`--parallel`		`1`	Copy parallelism.
`--point-in-time`			Use a point-in-time view of the storage repository when supported
`--prefix`			Prefix to use for objects in the bucket. Put trailing slash (/) if you want to use prefix as directory. e.g my-backup-dir/ would put repository contents inside my-backup-dir directory
`--region`			S3 Region
`--root-ca-pem-base64`			Certificate authority in-line (base64 enc.)
`--root-ca-pem-path`			Certificate authority file path
`--secret-access-key`			Secret access key (overrides AWS_SECRET_ACCESS_KEY environment variable)
`--session-token`			Session token (overrides AWS_SESSION_TOKEN environment variable)
`--[no-]times`		`false`	Synchronize blob times if supported.
`--[no-]update`		`true`	Whether to update blobs present in destination and source if the source is newer.

repository sync-to from-config

```
$ kopia repository sync-to from-config
```

Synchronize repository data to another repository in the provided configuration file

Flag	Short	Default	Help
`--[no-]delete`		`false`	Whether to delete blobs present in destination but not source.
`--[no-]dry-run`	`-n`	`false`	Do not perform copying.
`--file`			Path to the configuration file
`--[no-]must-exist`		`false`	Fail if destination does not have repository format blob.
`--parallel`		`1`	Copy parallelism.
`--[no-]times`		`false`	Synchronize blob times if supported.
`--token`			Configuration token
`--token-file`			Path to the configuration token file
`--[no-]token-stdin`		`false`	Read configuration token from stdin
`--[no-]update`		`true`	Whether to update blobs present in destination and source if the source is newer.

repository throttle get

```
$ kopia repository throttle get
```

Get throttling parameters for a repository

Flag	Short	Default	Help
`--[no-]json`		`false`	Output result in JSON format to stdout
`--[no-]json-indent`		`false`	[ADV] Output result in indented JSON format to stdout
`--[no-]json-verbose`		`false`	[ADV] Output non-essential data (e.g. statistics) in JSON format

repository throttle set

```
$ kopia repository throttle set
```

Set throttling parameters for a repository

Flag	Short	Default	Help
`--concurrent-reads`			Set max concurrent reads
`--concurrent-writes`			Set max concurrent writes
`--download-bytes-per-second`			Set the download bytes per second
`--list-requests-per-second`			Set max lists per second
`--read-requests-per-second`			Set max reads per second
`--upload-bytes-per-second`			Set the upload bytes per second
`--write-requests-per-second`			Set max writes per second

repository change-password

```
$ kopia repository change-password
```

Change repository password

Flag	Short	Default	Help
`--new-password`			New password

repository validate-provider

```
$ kopia repository validate-provider
```

Validates that a repository provider is compatible with Kopia

Flag	Short	Default	Help
`--concurrency-test-duration`			Duration of concurrency test
`--get-blob-workers`			Number of GetBlob workers
`--get-metadata-workers`			Number of GetMetadata workers
`--num-storage-connections`			Number of storage connections
`--put-blob-workers`			Number of PutBlob workers

Advanced Commands

Commands to manipulate BLOBs

- `blob delete` - Delete blobs by ID
- `blob gc` - Garbage-collect unused blobs
- `blob list` - List BLOBs
- `blob shards modify` - Perform low-level resharding of blob storage
- `blob show` - Show contents of BLOBs
- `blob stats` - Blob statistics

Commands to manipulate local cache

- `cache clear` - Clears the cache
- `cache info` - Displays cache information and statistics
- `cache prefetch` - Prefetches the provided objects into cache
- `cache set` - Sets parameters local caching of repository data
- `cache sync` - Synchronizes the metadata cache with blobs in storage

Commands to manipulate content in repository

- `content delete` - Remove content
- `content list` - List contents
- `content rewrite` - Rewrite content using most recent format
- `content show` - Show contents by ID
- `content stats` - Content statistics
- `content verify` - Verify that each content is backed by a valid blob

Commands to manipulate content index

- `index epoch list` - List the status of epochs
- `index inspect` - Inspect index blob
- `index list` - List content indexes
- `index optimize` - Optimize indexes blobs
- `index recover` - Recover indexes from pack blobs

Commands to manipulate logs stored in the repository

- `logs cleanup` - Clean up logs
- `logs list` - List logs
- `logs show` - Show contents of the log. When no flags or arguments are specified, only the last log is shown

Session commands

- `session list` - List sessions

Low-level commands to manipulate manifest items

- `manifest delete` - Remove manifest items
- `manifest list` - List manifest items
- `manifest show` - Show manifest items

Maintenance commands

- `maintenance info` - Display maintenance information
- `maintenance run` - Run repository maintenance
- `maintenance set` - Set maintenance parameters

Commands to manipulate repository

- `repository upgrade begin` - Begin upgrade
- `repository upgrade rollback` - Rollback the repository upgrade
- `repository upgrade validate` - Validate the upgraded indexes

blob delete

```
$ kopia blob delete <blobIDs>
```

Delete blobs by ID

Argument	Help
`blobIDs`	Blob IDs

blob gc

```
$ kopia blob gc
```

Garbage-collect unused blobs

Flag	Short	Default	Help
`--delete`			Whether to delete unused blobs
`--parallel`		`16`	Number of parallel blob scans
`--prefix`			Only GC blobs with given prefix
`--safety`		`full`	Safety level

blob list

```
$ kopia blob list
```

List BLOBs

Flag	Short	Default	Help
`--[no-]data-only`		`false`	Only list data blobs
`--exclude-prefix`			Blob ID prefixes to exclude
`--[no-]json`		`false`	Output result in JSON format to stdout
`--max-size`			Maximum size
`--min-size`			Minimum size
`--prefix`			Blob ID prefix
`--[no-]json-indent`		`false`	[ADV] Output result in indented JSON format to stdout
`--[no-]json-verbose`		`false`	[ADV] Output non-essential data (e.g. statistics) in JSON format

blob shards modify

```
$ kopia blob shards modify \
        --i-am-sure-kopia-is-not-running=... \
        --path=...
```

Perform low-level resharding of blob storage

Flag	Short	Default	Help
`--default-shards`			Default specification 'n1,..nN' or 'flat')
`--[no-]dry-run`		`false`	Dry run
`--[no-]i-am-sure-kopia-is-not-running`		`false`	Confirm that no other instance of kopia is running
`--override`			Override specification 'prefix=n1,..nN')
`--path`			Sharded directory path
`--remove-override`			Override specification 'prefix=n1,..nN')
`--unsharded-length`			Minimum sharded length

blob show

```
$ kopia blob show <blobID>
```

Show contents of BLOBs

Flag	Short	Default	Help
`--[no-]decrypt`		`false`	Decrypt blob if possible

Argument	Help
`blobID`	Blob IDs

blob stats

```
$ kopia blob stats
```

Blob statistics

Flag	Short	Default	Help
`--prefix`			Blob name prefix
`--[no-]raw`	`-r`	`false`	Raw numbers

cache clear

```
$ kopia cache clear
```

Clears the cache

Flag	Short	Default	Help
`--partial`			Specifies the cache to clear

cache info

```
$ kopia cache info
```

Displays cache information and statistics

Flag	Short	Default	Help
`--[no-]path`		`false`	Only display cache path

cache prefetch

```
$ kopia cache prefetch <object>
```

Prefetches the provided objects into cache

Flag	Short	Default	Help
`--hint`			Prefetch hint

Argument	Help
`object`	Object ID to prefetch

cache set

```
$ kopia cache set
```

Sets parameters local caching of repository data

Flag	Short	Default	Help
`--cache-directory`			Directory where to store cache files
`--content-cache-size-limit-mb`			Maximum size of local content cache (hard limit)
`--content-cache-size-mb`			Desired size of local content cache (soft limit)
`--content-min-sweep-age`			Minimal age of content cache item to be subject to sweeping
`--index-min-sweep-age`			Minimal age of index cache item to be subject to sweeping
`--max-list-cache-duration`			Duration of index cache
`--metadata-cache-size-limit-mb`			Maximum size of local metadata cache (hard limit)
`--metadata-cache-size-mb`			Desired size of local metadata cache (soft limit)
`--metadata-min-sweep-age`			Minimal age of metadata cache item to be subject to sweeping

cache sync

```
$ kopia cache sync
```

Synchronizes the metadata cache with blobs in storage

Flag	Short	Default	Help
`--parallel`		`16`	Fetch parallelism

content delete

```
$ kopia content delete <id>
```

Remove content

Argument	Help
`id`	IDs of content to remove

content list

```
$ kopia content list
```

List contents

Flag	Short	Default	Help
`--[no-]compression`	`-c`	`false`	Compression
`--[no-]deleted`		`false`	Include deleted content
`--[no-]deleted-only`		`false`	Only show deleted content
`--[no-]human`	`-h`	`false`	Human-readable output
`--[no-]json`		`false`	Output result in JSON format to stdout
`--[no-]long`	`-l`	`false`	Long output
`--[no-]non-prefixed`		`false`	Apply to content IDs without prefix
`--prefix`			Content ID prefix
`--[no-]prefixed`		`false`	Apply to content IDs with (any) prefix
`--[no-]summary`	`-s`	`false`	Summarize the list
`--[no-]json-indent`		`false`	[ADV] Output result in indented JSON format to stdout
`--[no-]json-verbose`		`false`	[ADV] Output non-essential data (e.g. statistics) in JSON format

content rewrite

```
$ kopia content rewrite [contentID]
```

Rewrite content using most recent format

Flag	Short	Default	Help
`--[no-]dry-run`	`-n`	`false`	Do not actually rewrite, only print what would happen
`--format-version`		`-1`	Rewrite contents using the provided format version
`--[no-]non-prefixed`		`false`	Apply to content IDs without prefix
`--pack-prefix`			Only rewrite contents from pack blobs with a given prefix
`--parallelism`		`16`	Number of parallel workers
`--prefix`			Content ID prefix
`--[no-]prefixed`		`false`	Apply to content IDs with (any) prefix
`--safety`		`full`	Safety level
`--[no-]short`		`false`	Rewrite contents from short packs

Argument	Help
`contentID`	Identifiers of contents to rewrite

content show

```
$ kopia content show <id>
```

Show contents by ID.

Flag	Short	Default	Help
`--[no-]json`	`-j`	`false`	Pretty-print JSON content
`--[no-]unzip`	`-z`	`false`	Transparently decompress the content

Argument	Help
`id`	IDs of contents to show

content stats

```
$ kopia content stats
```

Content statistics

Flag	Short	Default	Help
`--[no-]non-prefixed`		`false`	Apply to content IDs without prefix
`--prefix`			Content ID prefix
`--[no-]prefixed`		`false`	Apply to content IDs with (any) prefix
`--[no-]raw`	`-r`	`false`	Raw numbers

content verify

```
$ kopia content verify
```

Verify that each content is backed by a valid blob

Flag	Short	Default	Help
`--download-percent`			Download a percentage of files [0.0 .. 100.0]
`--[no-]full`		`false`	Full verification (including download)
`--[no-]include-deleted`		`false`	Include deleted contents
`--[no-]non-prefixed`		`false`	Apply to content IDs without prefix
`--parallel`		`16`	Parallelism
`--prefix`			Content ID prefix
`--[no-]prefixed`		`false`	Apply to content IDs with (any) prefix
`--progress-interval`		`3s`	Progress output interval

index epoch list

```
$ kopia index epoch list
```

List the status of epochs.

index inspect

```
$ kopia index inspect [blobs]
```

Inspect index blob

Flag	Short	Default	Help
`--[no-]active`		`false`	Inspect all active index blobs
`--[no-]all`		`false`	Inspect all index blobs in the repository, including inactive
`--content-id`			Inspect all active index blobs
`--parallel`		`8`	Parallelism

Argument	Help
`blobs`	Names of index blobs to inspect

index list

```
$ kopia index list
```

List content indexes

Flag	Short	Default	Help
`--[no-]json`		`false`	Output result in JSON format to stdout
`--sort`		`time`	Index blob sort order
`--[no-]summary`		`false`	Display index blob summary
`--[no-]superseded`		`false`	Include inactive index files superseded by compaction
`--[no-]json-indent`		`false`	[ADV] Output result in indented JSON format to stdout
`--[no-]json-verbose`		`false`	[ADV] Output non-essential data (e.g. statistics) in JSON format

index optimize

```
$ kopia index optimize
```

Optimize indexes blobs.

Flag	Short	Default	Help
`--[no-]all`		`false`	Optimize all indexes, even those above maximum size.
`--drop-contents`			Drop contents with given IDs
`--drop-deleted-older-than`			Drop deleted contents above given age
`--max-small-blobs`		`1`	Maximum number of small index blobs that can be left after compaction.

index recover

```
$ kopia index recover
```

Recover indexes from pack blobs

Flag	Short	Default	Help
`--blob-prefixes`			Prefixes of pack blobs to recover from (default=all packs)
`--blobs`			Names of pack blobs to recover from (default=all packs)
`--[no-]commit`		`false`	Commit recovered content
`--[no-]delete-indexes`		`false`	Delete all indexes before recovering
`--[no-]ignore-errors`		`false`	Ignore errors when recovering
`--parallel`		`1`	Recover parallelism

logs cleanup

```
$ kopia logs cleanup
```

Clean up logs

Flag	Short	Default	Help
`--[no-]dry-run`		`false`	Do not delete
`--max-age`		`720h`	Maximal age
`--max-count`		`10000`	Maximal number of files to keep
`--max-total-size-mb`		`1024`	Maximal total size in MiB

logs list

```
$ kopia logs list
```

List logs.

Flag	Short	Default	Help
`--[no-]all`		`false`	Show all logs
`--latest`	`-n`		Include last N logs, by default the last one is shown
`--older-than`			Include logs older than X (e.g. '1h')
`--younger-than`			Include logs younger than X (e.g. '1h')

logs show

```
$ kopia logs show [session-id]
```

Show contents of the log. When no flags or arguments are specified, only the last log is shown.

Flag	Short	Default	Help
`--[no-]all`		`false`	Show all logs
`--latest`	`-n`		Include last N logs, by default the last one is shown
`--older-than`			Include logs older than X (e.g. ‘1h’)
`--younger-than`			Include logs younger than X (e.g. ‘1h’)

Argument	Help
`session-id`	Log Session ID to show

session list

```
$ kopia session list
```

List sessions

manifest delete

```
$ kopia manifest delete <item>
```

Remove manifest items

Argument	Help
`item`	Items to remove

manifest list

```
$ kopia manifest list
```

List manifest items

Flag	Short	Default	Help
`--filter`			List of key:value pairs
`--[no-]json`		`false`	Output result in JSON format to stdout
`--sort`			List of keys to sort by
`--[no-]json-indent`		`false`	[ADV] Output result in indented JSON format to stdout
`--[no-]json-verbose`		`false`	[ADV] Output non-essential data (e.g. statistics) in JSON format

manifest show

```
$ kopia manifest show <item>
```

Show manifest items

Argument	Help
`item`	List of items

maintenance info

```
$ kopia maintenance info
```

Display maintenance information

Flag	Short	Default	Help
`--[no-]json`		`false`	Output result in JSON format to stdout
`--[no-]json-indent`		`false`	[ADV] Output result in indented JSON format to stdout
`--[no-]json-verbose`		`false`	[ADV] Output non-essential data (e.g. statistics) in JSON format

maintenance run

```
$ kopia maintenance run
```

Run repository maintenance

Flag	Short	Default	Help
`--[no-]full`		`false`	Full maintenance
`--safety`		`full`	Safety level
`--[no-]force`		`false`	[ADV] Run maintenance even if not owned (unsafe)

maintenance set

```
$ kopia maintenance set
```

Set maintenance parameters

Flag	Short	Default	Help
`--enable-full`			Enable or disable full maintenance
`--enable-quick`			Enable or disable quick maintenance
`--extend-object-locks`			Extend retention period of locked objects as part of full maintenance.
`--full-interval`			Set full maintenance interval
`--list-parallelism`			Override list parallelism.
`--max-retained-log-age`			Set maximum age of log sessions to retain
`--max-retained-log-count`			Set maximum number of log sessions to retain
`--max-retained-log-size-mb`			Set maximum total size of log sessions
`--owner`			Set maintenance owner user@hostname
`--pause-full`			Pause full maintenance for a specified duration
`--pause-quick`			Pause quick maintenance for a specified duration
`--quick-interval`			Set quick maintenance interval

repository upgrade begin

```
$ kopia repository upgrade begin
```

Begin upgrade.

Flag	Short	Default	Help
`--io-drain-timeout`		`15m0s`	Max time it should take all other Kopia clients to drop repository connections
`--max-permitted-clock-drift`		`5m0s`	The maximum drift between repository and client clocks
`--status-poll-interval`		`60s`	An advisory polling interval to check for the status of upgrade
`--[no-]allow-unsafe-upgrade`		`false`	[ADV] Force using an unsafe io-drain-timeout for the upgrade lock
`--commit-mode`			[ADV] Change behavior of commit. When not set, commit on validation success. ‘always’: always commit. ’never’: always exit before commit.
`--[no-]lock-only`		`false`	[ADV] Advertise the upgrade lock and exit without actually performing the drain or upgrade

repository upgrade rollback

```
$ kopia repository upgrade rollback
```

Rollback the repository upgrade.

Flag	Short	Default	Help
`--[no-]force`		`false`	Force rollback the repository upgrade, this action can cause repository corruption

repository upgrade validate

```
$ kopia repository upgrade validate
```

Validate the upgraded indexes.

Upgrading to New Version

Upgrading Kopia from one version to the next is a seamless process except for the upgrade paths discussed in this document. If your Kopia upgrade path is not mentioned here, then you are safe to upgrade Kopia as normal.

Upgrading Kopia v0.8 and Earlier to Newer Version

Kopia v0.9 adds support for several new features thanks to a brand-new index format.

If your repository was created in a version older than v0.9, please follow the steps below to upgrade.

Notes

It is critical to follow the process outlined before exactly and to verify that during the upgrade steps no instance of `kopia` is connected to the repository.

This includes:

- `kopia` or `KopiaUI` running interactively on in scripts,
- running as a scheduled background tasks (e.g. using `crontab`),
- running in server mode either as the current user or system-wide daemon (e.g. using `systemd`),
- running in Docker containers and similar.

Also note, that after the upgrade, kopia v0.8 and earlier will not be able to open the repository anymore. Once upgraded all new v0.9 features will be supported except password change, which is only available for newly-created repositories.

Upgrade Process

1. Select one kopia client that will perform the upgrade, if there are more clients, pick the one that is currently the owner of maintenance process, which is typically the client that first created the repository.

2. Disconnect all other kopia clients:

- using CLI run:

```
$ kopia repository disconnect
```

- using KopiaUI, click `Repository` | `Disconnect`.

- make sure to stop any running `kopia server` instances and disable all background kopia tasks, such as periodic snapshots in `crontab`.

3. Upgrade all kopia clients to the latest version >=v0.9.x

4. Using the designated `kopia` client, run:

```
$ kopia repository set-parameters --upgrade
```

5. Verify upgrade by:

```
$ kopia repository status
```

You should see `Format version: 2`

5. Reconnect kopia clients that were disconnected in step 2 and re-enable all disabled background jobs.

6. When in doubt, it's better not to guess, but post a question on https://kopia.discourse.group

Last modified February 12, 2024: docs(site): improvements related to spelling and structure (#3618) (25437ae5)

Frequently Asked Questions

Questions

Is your question not answered here? Please ask in the Kopia discussion forums for help!

Answers

What is a Snapshot?

A `snapshot` is a point-in-time backup of your files/directories; each snapshot contains the files/directories that you can restore when you need to.

What is a Repository?

A `repository` is the storage location where your snapshots are saved; Kopia supports cloud/remote, network, and local storage locations and all repositories are encrypted with a password that you designate.

See the repository help docs for more information.

What is a Policy?

A `policy` is a set of rules that tells Kopia how to create/manage snapshots; this includes features such as compression, snapshot retention, and scheduling when to take snapshots automatically.

How to Restore My Backed Up Files/Directories?

Files/directories are restored from the snapshots you create. To restore the data you backed up in a snapshot, Kopia gives you three options:

- mount the contents of a snapshot as a local disk so that you can browse and copy files/directories from the snapshot as if the snapshot is a local directory on your machine;
- restore all files/directories contained in a snapshot to any local or network location that you designate;
- or selectively restore individual files from a snapshot.

The Getting Started Guide provides directions on how to restore files/directions when using Kopia GUI and when using Kopia CLI.

How Do I Define Files And Folders To Be Ignored By Kopia?

Files and directories can be ignored from snapshots by adding `ignore rules` to the `policy` or creating `.kopiaignore` files. For more information, please refer to our guide on creating ignore rules.

How Do I Enable Encryption?

Encryption is at the `repository` level, and Kopia encrypts all snapshots in all repositories by default. Kopia asks for a password when creating your `repository`. This password is used to encrypt your backups.

By default, Kopia uses the `AES256-GCM-HMAC-SHA256` encryption algorithm for all repositories, but you can choose `CHACHA20-POLY1305-HMAC-SHA256` if you want to. Picking an encryption algorithm is done when you initially create a `repository`. In `KopiaUI`, to pick the `CHACHA20-POLY1305-HMAC-SHA256` encryption algorithm, you need to click the `Show Advanced Options` button at the screen where you enter your password when creating a new `repository`. For Kopia CLI users, you need to use the `--encryption=CHACHA20-POLY1305-HMAC-SHA256` option when creating a `repository` with the `kopia repository create` command.

Currently, encryption algorithms cannot be changed after a `repository` has been created.

> NOTE There is no way to recover it or the files and folders within that repository. Store your repository password in a safe place, such as a password manager, so you can retrieve it later.

How Do I Enable Compression?

Compression is controlled by policies and is disabled by default. If compression is not working for a snapshot, it is likely that you have not enabled compression yet in your `policy`.

Enabling compression when using `KopiaUI` is easy; edit the `policy` you want to add compression to and pick a `Compression Algorithm` in the `Compression` section. Kopia CLI users need to use the `kopia policy set` command as shown in the Getting Started Guide. You can set compression on a per-source-directory basis…

```
kopia policy set </path/to/source/directory/> --compression=<none|deflate-best-compression|deflate-best-speed|deflate-default|gzip|gzip-best-compression|gzip-best-speed|pgzip|pgz
```

…or globally for all source directories:

```
kopia policy set --global --compression=<none|deflate-best-compression|deflate-best-speed|deflate-default|gzip|gzip-best-compression|gzip-best-speed|pgzip|pgzip-best-compression|
```

If you enable or disable compression or change the compression algorithm, the new setting is applied going forward and not reteroactively. In other words, Kopia will not modify the compression for files/directories already uploaded to your repository.

If you are unclear about what compression algorithm to use, `zstd` is considered one of the top algorithms currently.

How Do I Enable Data Deduplication?

Data deduplication is enabled automatically by Kopia for all repositories, regardless of whether you use the GUI or CLI. You do not need to do anything.

How Do I Change My Repository Password?

You must use Kopia CLI if you want to change your `repository` password; changing password is not currently supported via Kopia GUI. The `kopia repository change-password` command is used to change your password.

Before changing your password, you must be connected to your `repository`. This means that you **can** reset your password if you forget your password AND you are still connected to your `repository`. But this also means that you **cannot** reset your password if you forget your password and you are NOT still connected to your `repository`, because you will need your current password to connect to the `repository`.

Remember to select a secure *repository password*. The password is used to decrypt and access the data in your snapshots.

Does Kopia Support Storage Classes, Like Amazon Glacier?

Yes. Please read the storage classes guide to learn more.

How Do I Decrease Kopia's CPU Usage?

It is difficult to know what is causing high CPU use on your machine without details about your unique issue (which you can post about on the community forums). However, generally speaking, there are two `policy` settings you should change if you want to decrease the amount of CPU Kopia uses while creating snapshots:

1. Maximum Parallel Snapshots: This setting controls how many snapshots Kopia runs simultaneously. The lower this number, the less CPU Kopia will use when running multiple snapshots – with the caveat that the total time it takes to run all your snapshots will be higher. This setting is not applicable if you only ever run one snapshot at a time.
    - Kopia CLI users can change this setting by running the `kopia policy set --global --max-parallel-snapshots=#` command, where `#` is the number of snapshots you want Kopia to run simultaneously.
    - Kopia GUI users can change this setting from the `Upload` tab when editing the `global` policy in `KopiaUI`.
2. Maximum Parallel File Reads: This setting controls how many files Kopia uploads simultaneously when running a snapshot. The lower this number, the less CPU Kopia will use when running a snapshot – with the caveat that it will take longer to complete a snapshot. By default, Kopia sets this setting to the number of logical cores your machine's CPU has, so make sure to lower it to a smaller number than the default.
    - Kopia CLI users can change this setting by running the `kopia policy set [target] --max-parallel-file-reads=#` command, where `#` is the number of file uploads you want Kopia to run simultaneously. This setting can be set per policy or globally, in which case make sure to use `--global` in your `kopia policy set` command.
    - Kopia GUI users can change this setting from the `Upload` tab when editing policies in `KopiaUI`, either the `global` policy or individual policies.

An added benefit of decreasing these settings is that Kopia's memory usage will also decrease.

How Do I Decrease Kopia's Memory (RAM) Usage?

It is difficult to know what is causing high memory use on your machine without details about your unique issue (which you can post about on the community forums). However, generally speaking, compression and parallelism (i.e., simultaneous snapshots or simultaneous uploads) are the two big culprits of memory usage in Kopia when creating snapshots. If you want to decrease the amount of memory used by Kopia, you should tweak these settings:

- Disabling compression will result in less memory usage than enabling compression. If you want to keep compression, compression benchmarks suggest `s2`, `deflate`, and `gzip` are the most memory-friendly compression algorithms when backing up small files. When backing up large files, all the compression algorithms have similar memory usage. Thus, if your machine has low memory, try `s2`, `deflate`, or `gzip`. Read the FAQ on enabling compression to learn how to change or remove compression in Kopia.
- Decreasing the number of parallel snapshots and parallel file reads will decrease Kopia's memory usage because Kopia will run fewer simultaneous processes. Read the FAQ on decreasing Kopia's CPU usage to learn how to decrease parallelism in Kopia.

What are Incomplete Snapshots?

When creating snapshots from large files or folders, `Kopia` sometimes marks snapshots as incomplete. This is because `Kopia` creates `checkpoints` at predefined time intervals. If a snapshot takes longer than the predefined checkpoint interval, `Kopia` creates a temporary **incomplete** snapshot, preventing the snapshot from being garbage-collected by the maintenance tasks. *Kopia* will remove incomplete snapshots once a **complete** snapshot of the files and directories has been created.

For more information on the `checkpoint interval`, please refer to the command-line reference.

What is a Kopia Repository Server?

See the Kopia Repository Server help docs for more information.

KopiaUI and Multiple Repositories

When KopiaUI starts up, it will look for configuration files in Kopia's configuration directory (`%APPDATA%\kopia` on Windows; `$HOME/.config/kopia` on linux; `$HOME/Library/Application Support/kopia` on macOS). KopiaUI will look for all files ending in `*.config` and use these configurations to determine the set of repositories to connect to.

KopiaUI will always look for a `repository.config` file, even if that file does not exist, in which case it will try to start up a connection which will never succeed.

Be aware that if you create multiple config files for testing purposes, eg, `repository.orig.config`, `repository.test1.config`, `repository.test2.config`, etc., KopiaUI will try to connect to ALL of them at startup, even if they are not intended to be valid. Thus, if you don't want KopiaUI to use a config file, make sure it ends in something other than `.config`.

Last modified September 20, 2024: docs(ui): Document how KopiaUI Desktop handles multiple config files and repositories (#4049) (8e37cea7)