Restic ——快速、安全、高效、跨平台的备份程序 官方文档中文翻译（选译）

作者：HaujetZhao

原文链接：https://ld246.com/article/1601527149440

来源网站：链滴

介绍

Restic 是一种快速而安全的备份程序，使用 go 语言编写，高效、跨平台。

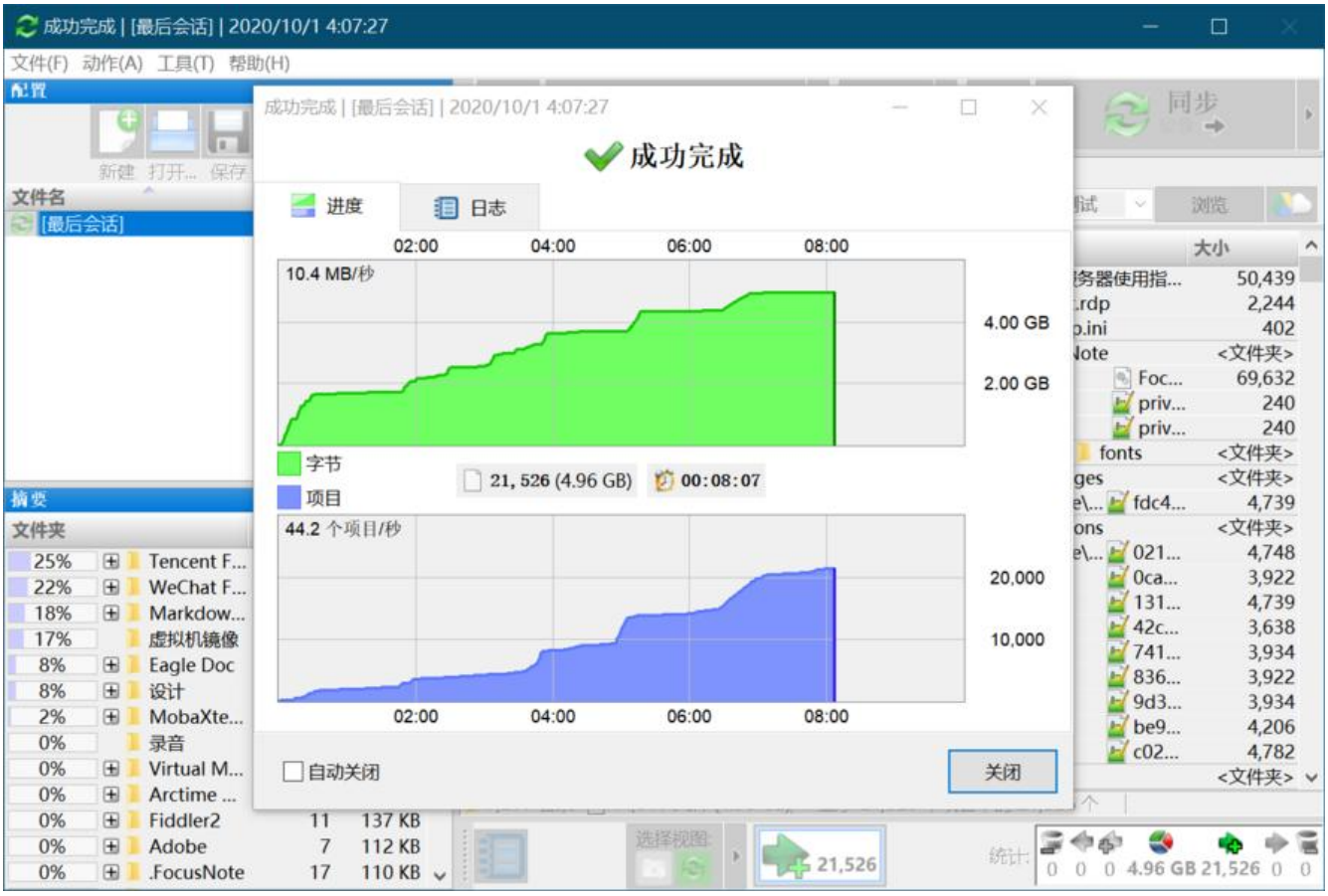
本文是官方文档 0.10.0 版本的部分翻译（2020年10月1日），如果有其它需要查阅的部分，请访问方文档网站。

相比于复制，使用 Restic 可以进行增量备份、加密存储、创建快照、速度快，可以放心地将备份好储存库放到云存储，不用担心隐私文件泄漏（唯一需要担心的就是网盘服务商会不会不好好保存你的密后的存储库文件了）。

下面是我将我固态硬盘的 **文档** 文件夹用：

- FreeFileSync 以复制的方式同步到一块移动硬盘
- 用 Restic 备份到同一移动硬盘的储存库上

两种速度的速度对比（前者用了 8:07，后者用了 2:05）：



```
C:\Windows\System32\cmd.exe
[1:56] 93.43%  12043 files 4.631 GiB, total 14320 files 4.957 GiB, 0 errors ET
[1:56] 93.43%  12044 files 4.631 GiB, total 14320 files 4.957 GiB, 0 errors ET
[1:56] 93.56%  12044 files 4.638 GiB, total 14320 files 4.957 GiB, 0 errors ET
[1:56] 93.64%  12045 files 4.642 GiB, total 14320 files 4.957 GiB, 0 errors ET
[1:56] 93.66%  12046 files 4.643 GiB, total 14320 files 4.957 GiB, 0 errors ET
[1:56] 93.66%  12081 files 4.643 GiB, total 14320 files 4.957 GiB, 0 errors ET
[1:56] 93.66%  12111 files 4.643 GiB, total 14320 files 4.957 GiB, 0 errors ET
[1:56] 93.68%  12131 files 4.644 GiB, total 14320 files 4.957 GiB, 0 errors ET
[1:59] 95.77%  13722 files 4.747 GiB, total 14320 files 4.957 GiB, 0 errors
[1:59] 95.80%  14041 files 4.749 GiB, total 14320 files 4.957 GiB, 0 errors ET
[1:59] 95.80%  14099 files 4.749 GiB, total 14320 files 4.957 GiB, 0 errors ET
[1:59] 95.81%  14155 files 4.749 GiB, total 14320 files 4.957 GiB, 0 errors ET
[1:59] 95.81%  14211 files 4.749 GiB, total 14320 files 4.957 GiB, 0 errors ET
[1:59] 95.81%  14261 files 4.750 GiB, total 14320 files 4.957 GiB, 0 errors ET
[1:59] 95.91%  14291 files 4.755 GiB, total 14320 files 4.957 GiB, 0 errors ET
[2:03] 99.47%  14310 files 4.931 GiB, total 14320 files 4.957 GiB, 0 errors ET
[2:03] 99.47%  14311 files 4.931 GiB, total 14320 files 4.957 GiB, 0 errors ET
[2:03] 99.60%  14312 files 4.937 GiB, total 14320 files 4.957 GiB, 0 errors ET

Files:        14320 new,     0 changed,     0 unmodified
Dirs:          7211 new,     0 changed,     0 unmodified
Added to the repo: 4.299 GiB

processed 14320 files, 4.957 GiB in 2:05
snapshot f93eabc5 saved

E:\>
```

在以下各节中，我们将介绍典型的工作流程，从安装、准备新的存储库以及进行第一次备份开始。

安装

发行包

提示，当你的软件版本过旧的时候，始终可以到 Restic 项目的官方页面下载二进制文件。

这些是最新的二进制文件，以可复制和可验证的方式构建，您可以下载并运行它们，而无需执行其他装工作。

请参阅下面的 官方二进制文件 部分以获取各种下载，也可以使用 restic self-update 命令更新官方进制文件

其它安装方法略

官方二进制包

稳定发行版

您可以从 restic发布页面 下载 restic 的最新稳定版本

准备新存储库

保存备份的位置称为“存储库”。本章介绍如何创建（“init”）这样的存储库。存储库可以存储在地，也可以存储在某个远程服务器或服务上。我们将首先介绍使用本地存储库；本章的其余部分将介所有其他选项。一旦你阅读了这里的相关章节，你可以跳到下一章。

对于自动备份，restic 接受环境变量 RESTIC_REPOSITORY 中的存储库位置。对于密码，有几个选项：

- 通过环境变量 RESTIC_PASSWORD 设置密码
- 通过 --password-file 选项或者环境变量RESTIC_PASSWORD_FILE 指向存储密码的文件
- 通过 --password-command 选项配置程序，或者通过设置环境变量RESTIC_PASSWORD_COMAND，使其可被调用

本地

运行以下命令，并输入遍密码，就可以在 /srv/restic-repo 创建存储库：

```
$ restic init --repo /srv/restic-repo
enter password for new repository:
enter password again:
created restic repository 085b3c76b9 at /srv/restic-repo
Please note that knowledge of your password is required to access the repository.
Losing your password means that your data is irrecoverably lost.
```

警告：记住你的密码！如果丢失，你将永远无法访问你的存储库！

警告：在 Linux 上，由于兼容性的原因，在 CIFS (SMB) 共享上保存存储库是不推荐的。要么使用另一个后端，要么通过环境变量 GODEBUG 为 asyncpreemptoff=1。GitHub issue #2659 有更详情解释。

SFTP

要备份数据到 SFTP，你需要首先设置 SSH 。因为如果服务器提示输入密码 restic 就会失败，所以，需要配置无密码登陆方式（密钥）。

当服务器设置好后，通过改变 init 命令中的 URL，就可以初始化 SFTP 中的存储库：

```
$ restic -r sftp:user@host:/srv/restic-repo init
enter password for new repository:
enter password again:
created restic repository f1c6108821 at sftp:user@host:/srv/restic-repo
Please note that knowledge of your password is required to access the repository.
Losing your password means that your data is irrecoverably lost.
```

你也可以指定一个相对路径，这时，这个相对路径指定的就是这个远程路径的用户根目录的相对路径。

另外，如果 SFTP 服务器强制域名限制型用户，你可以这样指定用户： user@domain@host

注意：请知晓，sftp 服务器不会将 ~ 解释为用户根目录。

如果你需要指定 ipv6 的地址，你需要使用 url 语法。例如，在 [::1] 地址的 2222 端口上用 user 用登陆的 /srv/restic-repo 存储库，应当这样指定：

sftp://user@[::1]:2222//srv/restic-repo

注意那个双斜杠，第一个斜杠分割路径的连接设置，第二个斜杠是路径的起始位置。如果要使用相对径，那么只使用一个斜杠。

另外，你可以在 ssh 配置文件（通常在 ~/.ssh/config 或者 /etc/ssh/ssh_config）中创建一个条目：

```
Host foo
    User bar
    Port 2222
```

然后使用指定的 foo 作为服务器名字（这样就不用每次都输入服务器地址了）：

```
$ restic -r sftp:foo:/srv/restic-repo init
```

你也可以添加一个特殊的不存在的域名，只是用于 restic，然后使用 Hostname 选项设置真实的域：

```
Host restic-backup-host
    Hostname foo
    User bar
    Port 2222
```

这是用法：

```
$ restic -r sftp:restic-backup-host:/srv/restic-repo init
```

最后，如果你想用完全不同的程序创建 sftp 连接，你可以通过 -o sftp.command="foobar" 选项，定要运行的命令。

注意：请知晓，sftp 服务器会在没有收到客户端的数据时关闭连接。当 restic 处理海量未发生改动的据时，就有可能中断连接。要避免这个问题，请在客户端的 .ssh/config 文件中添加：

```
ServerAliveInterval 60
ServerAliveCountMax 240
```

REST 服务器

略

Amazon S3

略

Minio Server

略

Wasabi

略

OpenStack Swift

略

Backblaze B2

略

Microsoft Azure Blob Storage

略

Google Cloud Storage

略

通过 rclone 搭建的服务

rclone 可以被用来访问许多其它云存储服务。首先，确保安装并 配置 了 rclone。一般的指定 rclone 后端的格式是： rclone:<remote>:<path>，其中 <remote>:<path> 会被直接传递到 rclone。当配置了一个名叫 foo 的远程路径，你可以用以下的命令让 restic 在 bar 路径中创建一个新储存库：

```
$ restic -r rclone:foo:bar init
```

restic 会管理开始和结束 rclone

作为一个例子，假设你有一个用配置好的 Backblaze B2 类型的远程路径叫 b2prod ，其存储桶叫 yggdrasil，你可以这样使用 rclone 列出存储桶内的文件：

```
$ rclone ls b2prod:yggdrasil
```

如果要在存储桶的根目录下创建存储库，这样运行 restic：

```
$ restic -r rclone:b2prod:yggdrasil init
```

使用 rclone 监听存储库的空目录，应该会返回一列类似这样的结果：

```
$ rclone ls b2prod:yggdrasil/foo/bar/baz
    155 bar/baz/config
    448 bar/baz/keys/4bf9c78049de689d73a56ed0546f83b8416795295cda12ec7fb9465af390044
```

通过 用环境变量配置 rclone 可以配置 rclone，所以可以通过 RCLONE_BWLIMIT 环境变量来限制 rclone 的带宽：

```
$ export RCLONE_BWLIMIT=1M
```

如果要 debug rclone，你可以设置环境变量： RCLONE_VERBOSE=2

rclone 后端还有以下选项：

- -o rclone.program 指定 rclone 的路径

- -o rclone.args 允许设置传递给 rclone 的参数，默认是：serve restic --stdio --b2-hard-delete --drive-use-trash=false

对于最后两个参数（--b2-hard-delete 和 --drive-use-trash=false）的原因可以在 issue #1657 查。

为了启动 rclone，restic 会通过合并以下列表：

- rclone.program
- rclone.args
- 将跟在指定存储库的 rclone: 前缀后的数值作为最后的参数

来创建一系列参数，所以就wqj 这样调用 restic：

```
$ restic -o rclone.program="/path/to/rclone" \
  -o rclone.args="serve restic --stdio --bwlimit 1M --b2-hard-delete --verbose" \
  -r rclone:b2:foo/bar
```

会运行以下命令：

```
$ /path/to/rclone serve restic --stdio --bwlimit 1M --b2-hard-delete --verbose b2:foo/bar
```

手动设置 rclone.program 也可以通过服务器上的 SSH，设置运行一个远程的 rclone 实例，例如：

```
$ restic -o rclone.program="ssh user@host rclone" -r rclone:b2:foo/bar
```

rclone 命令也可以在 SSH 配置或用户的公钥中预设，这时，只要启动 SSH 连接就足够了（这时在 rclone: 后存储库中的参数就不起用了）：

```
$ restic -o rclone.program="ssh user@host" -r rclone:x
```

Windows 上的密码提示

目前，restic 只支持 Windows 默认控制台连接，如果你使用模拟环境（例如使用像是 Mintty 、 rxvt 终端的 MSYS2 、 Cygwin ），你可能得到一个密码错误提示。

你可以通过一个叫 winpty 的特殊工具（ here 、 here 有详情）解决。在 MSYS2 上，你可以这样装 winpty：

```
$ pacman -S winpty
$ winpty restic -r /srv/restic-repo init
```

备份

现在我们已经准备好去备份数据了。在 restic 中，我们把指定点、指定时间的一个目录中的内容叫“照”，运行下列命令，输入存储库的密码：

```
$ restic -r /srv/restic-repo --verbose backup ~/work
open repository
enter password for repository:
password is correct
lock repository
```

```
load index files
start scan
start backup
scan finished in 1.837s
processed 1.720 GiB in 0:12
Files:       5307 new,     0 changed,     0 unmodified
Dirs:        1867 new,     0 changed,     0 unmodified
Added:      1.200 GiB
snapshot 40dc1520 saved
```

你可以看到，restic 非常快地创建了这个目录的备份！这份快照用一列十六进制的字符 40dc1520 为标志。

你可以看到，restic 告诉我们它处理了 1.720 GiB 的数据，这是在 ~/work 中的内容大小。它也告诉们，只有 1.200 GiB 大小的文件被添加到了存储库，这意味着有一点数据是重复的，restic 可以高效避免重复备份。

如果你不传递 --verbose 选项，restic 会打印更少的数据。你仍可以得到这些漂亮的实时数据展示提醒下，实时展示只展示处理过的文件，不展示未传输的数据。传输过的数据（由于反重复机理）可会忽小忽大。

如果你再运行这个命令一遍，restic 会创建另一个数据快照，但是这回快了无数倍，因为没有新数据添加到储存库（因为所有的数据都在这了），反重复机制就是这样工作的！

```
$ restic -r /srv/restic-repo backup --verbose ~/work
open repository
enter password for repository:
password is correct
lock repository
load index files
using parent snapshot d875ae93
start scan
start backup
scan finished in 1.881s
processed 1.720 GiB in 0:03
Files:          0 new,     0 changed,  5307 unmodified
Dirs:           0 new,     0 changed,  1867 unmodified
Added:      0 B
snapshot 79766175 saved
```

你也可以在同一个储存库中备份单个文件（不使用 --verbose 表示显示更少的信息）：

```
$ restic -r /srv/restic-repo backup ~/work.txt
enter password for repository:
password is correct
snapshot 249d0210 saved
```

如果你对 restic 做什么感兴趣，传递 --verbose 两次（或者 --verbose=2） 显示 restic 遇到的每个件、文件夹的信息：

```
$ echo 'more data foo bar' >> ~/work.txt

$ restic -r /srv/restic-repo backup --verbose --verbose ~/work.txt
open repository
enter password for repository:
```

```
password is correct
lock repository
load index files
using parent snapshot f3f8d56b
start scan
start backup
scan finished in 2.115s
modified  /home/user/work.txt, saved in 0.007s (22 B added)
modified  /home/user/, saved in 0.008s (0 B added, 378 B metadata)
modified  /home/, saved in 0.009s (0 B added, 375 B metadata)
processed 22 B in 0:02
Files:          0 new,     1 changed,     0 unmodified
Dirs:           0 new,     2 changed,     0 unmodified
Data Blobs:      1 new
Tree Blobs:      3 new
Added:      1.116 KiB
snapshot 8dc503fc saved
```

事实上，几个域名可以使用同一个存储库来备份文件夹和文件，这会有更大的反重复。

提醒，当你备份不同的目录（或者被保存的目录有一个变量名字组件，例如一个 time/date）时，restic 总是需要读取这些文件，然后之后可以计算哪些部分是需要保存的。当你又备份同一个目录（或者新增、改变的文件）时，restic 会在储存库中找到旧的快照，默认只读取这些自从旧快照以来改变或增的文件。这是基于以下文件系统中的文件属性来实现的：

- 类型（文件、符号链接、还是文件夹？）
- 更改时间
- 大小
- 节点数字（在文件系统中用于索引文件的内部数字）

现在是时候去运行 `restic check` 来检验一个所有数据都被适当地存储在存储库了。你应该多多运行个命令，以确保储存库的内部结构没有错误。

排除文件

你可以通过指定排除模式，来排除文件，当前的排除选项有：

- `--exclude` 指定一次或多次，以排除一个或多个条目
- `--iexclude` 与`--exclude` 用法一致，只是忽略大小写
- `--exclude-caches` 指定一次，以排除包含指定文件的文件夹
- `--exclude-file` 指定一次或多次，以排除指定文件中列出的条目
- `--iexclude-file` 与`--exclude-file` 用法一致，只是忽略大小写
- `--exclude-if-present foo` 指定一次或多次，如果一个文件夹包含一个文件叫做`foo` （可选的有一指定的头，对文件名没有支持的通配符），排除这个文件夹的内容
- `--exclude-larger-than size` 指定一次，排除超过指定大小的文件。

使用 `restic help backup` 以查阅更多

例如我们有一个文件叫 `excludes.txt` ，它有以下内容：

```
## exclude go-files
*.go
## exclude foo/x/y/z/bar foo/x/bar foo/bar
foo/**/bar
```

它可以被这样使用：

```
$ restic -r /srv/restic-repo backup ~/work --exclude="*.c" --exclude-file=excludes.txt
```

上述命令会指示 restic 排除匹配下列标准的文件；

- 所有匹配 `*.c` 的文件
- 所有匹配 `*.go` 的文件
- 所有在 `foo` 目录下某处名为`bar` 的文件和子文件夹

模式匹配规则使用了 `filepath.Glob` ，规则会用于检验所有文件的绝对路径（即使你传递的是一个相路径）

在环境变量中保存的排除文件会被使用 `os.ExpandEnv` 展开，所以对于用户 `bob`， `/home/$USER/fo` 会被展开为 `/home/bob/foo` 。要得到一个字面的美元符，请使用 `$$` 。注意， `~` 扩展符不会起作，请使用 `$HOME` 环境变量代替。

模式需要匹配路径的完整组件，例如， `foo`：

- 匹配 `/dir1/foo/dir2/file` 、`/dir/foo`
- 不匹配 `/dir/foobar` 、`barfoo`

跟在路径后面的 `/` 会被忽略，在路径前面的 `/` 表示模式是从根目录开始地。这意味着，`/bin` 匹配 `/bin bash` ，但不会匹配 `/usr/bin/restic`

常规通配符不会被用于匹配目录分割符 `/` ，例如： `b*ash` 匹配 `/bin/bash` 但不匹配 `/bin/ash`

对于这个，特殊的通配符 `**` 可以被用于匹配任意子目录： `foo/**/bar` 就会匹配：

- `/dir1/foo/dir2/bar/file`
- `/foo/bar/file`
- `/tmp/foo/bar`

在模式中的空格列出了一个逐字的例外文件，也就是，为了排除一个叫 `foo bar star.txt` 的文件，把放在一行，以排除这个文件。注意，开始的空格和尾随的空格会被自动裁剪掉 - 为了匹配这些，请在关或末尾使用例如 `*` 的通配符。

在其它排除选项中的空格，（例如命令行中的 `--exclude` ）会以不同的方式应用（取决于操作系统或hell 的不同），restic 自身不做转义，但你的 Shell 可能会先将命令转义，再传递给 restic

在 Unix 类 Shell 上，你可以用引号或反斜杠，例如：

- `--exclude='foo bar star/foo.txt'`
- `--exclude="foo bar star/foo.txt"`
- `--exclude=foo\ bar\ star/foo.txt`

通过指定 --one-file-system 选项，你可以让 restic 只备份中最初指定文件或文件夹时的的文件系统的文件，换句话说，这可以防止 restic 在备份时穿过文件系统的界限。

例如，如果你在备份 / 时使用这个选项，然后你有外置媒体挂载在了 /media/usb ，restic 就不会去管 media/usb 下的文件，因为它和最初指定的 / 文件系统不一样，在用这个选项时，虚拟文件系统例如 proc 也会被认作不一样的文件系统：

```
$ restic -r /srv/restic-repo backup --one-file-system /
```

注意，这不会影响你在一个命令中指定多个文件系统：

```
$ restic -r /srv/restic-repo backup --one-file-system / /media/usb
```

会同时备份 / 和 /media/usb 两文件系统，但不会包括其它文件系统例如 /sys 和 /proc

注意：

--one-file-system 当前在 Windows 上还无法支持，并且会导致备份迅速失败。

通过 –exclude-larger-than 选项，可以排除掉大于指定大小的文件：

```
$ restic -r /srv/restic-repo backup ~/work --exclude-larger-than 1M
```

这会在备份时排除掉 ~/work 中大于 1MB 的文件。

默认的单位大小是 bytes （字节），支持 k/K 、 m/M 、 g/G 、t/T 等单位表示。

包含文件

通过使用 --files-from 选项，你可以从一个或多个文件读取你想要备份的文件。如果有大量文件不同一个文件夹的文件（例如被其它软件分好的文件夹）需要被备份，那么这就十分有用。

例如也许你想备份有匹配以下模式的的文件：

```
$ find /tmp/somefiles | grep 'PATTERN' > /tmp/files_to_backup
```

你就可以使用 restic 备份这些过滤后的文件：

```
$ restic -r /srv/restic-repo backup --files-from /tmp/files_to_backup
```

你也可以顺便将 --files-from 选项与普通文件参数结合：

```
$ restic -r /srv/restic-repo backup --files-from /tmp/files_to_backup /tmp/some_additional_fil
```

在列的路径可以是绝对或相对路径，注意， --files-from 文件中中列出的模式与排除模式匹配方法是同的。

比较快照

Restic 有一个 diff 命令，可以展示两个快照的不同，再显示一个简单的统计，只要传递两个快照的 ID 就行：

```
$ restic -r /srv/restic-repo diff 5845b002 2ab627a6
```

```
password is correct
comparing snapshot ea657ce5 to 2ab627a6:

 C   /restic/cmd_diff.go
+    /restic/foo
 C   /restic/restic

Files:          0 new,     0 removed,     2 changed
Dirs:           1 new,     0 removed
Others:         0 new,     0 removed
Data Blobs:    14 new,    15 removed
Tree Blobs:     2 new,     1 removed
  Added:   16.403 MiB
  Removed: 16.402 MiB
```

备份特殊的项目和元数据

Symlinks （符号文件）会作为 synlinks 被存档，restic 不会跟踪他们。当你恢复时，得到的仍是这符号链接，保留着它的坐标和时间戳。

如果在这个目录下有一个绑定的挂载，restic 会深入这个目录保存。

设备文件会被保存和恢复为设备文件，意味着， `/dev/sda` 会被存档为块设备文件。设备内的内容不被读取。

默认，restic 不会保存任何文件和其它项目的访问时间，因为无法可靠地通过 restic 自身更新文件的问时间。如果你实在想保存访问时间，那就使用 `--with-atime` 这个选项。

不支持 inode 连续性的文件系统中，例如基于 FUSE 的文件系统和 pCloud，可以通过传递 `--ignore-node` 选项，在文件改变对比时上忽略 inode。

从 stdin 读取数据

有时，直接保存一个程序的输出非常好用。例如， `mysqldump` ，这样 SQL 之后可以被恢复。Restic 支持这种操作，只要传递 `--stdin` 参数即可：

```
$ set -o pipefail
$ mysqldump [...] | restic -r /srv/restic-repo backup --stdin
```

这会为 `mysqldump` 的输出创建新的快照，你可以使用 fuse 挂载选项挂载这个储存库，然后读取它。

默认，会使用 `stdin` 作为文件名，可以通过 `--stdin-filename` 指定一个不同的文件名：

```
$ mysqldump [...] | restic -r /srv/restic-repo backup --stdin --stdin-filename production.sql
```

高度推荐使用 `pipefail` 选项，这样，程序 pip 的非 0 退出码使整个链返回一个非 0 退出码，详细请考 Use the Unofficial Bash Strict Mode

给备份加标签

快照可以有多个标签，使用 `--tag` 选项添加即可：

```
$ restic -r /srv/restic-repo backup --tag projectX --tag foo --tag bar ~/work
```

[...]

标签可以被用来保存和通过 forget 命令丢弃快照。 tag 命令可以用于改变已有快照的标签。、

空间要求

Restic 目前默认当你备份时，有足够的空间供备份操作。这只是对于大多数云盘服务商的假设，但是于本地硬盘，这个假设就有可能不成立。

当你在备份过程中硬盘空间用完了，会有一些信息添加到储存库，但是不会有快照被创建，因为只有成功后，快照才会被写入。之前的快照会继续工作。

环境变量

为了命令行选项，restic 支持在环境变量中传递变量选项。下列是支持的环境变量：

```
RESTIC_REPOSITORY              本地储存库的位置 (替换 -r)
RESTIC_PASSWORD_FILE           密码文件的位置 (替换 --password-file)
RESTIC_PASSWORD                储存库实际的密码
RESTIC_PASSWORD_COMMAND        为储存库的标准输出打印密码的命令
RESTIC_KEY_HINT                首先解码的 ID 的 key，在其它 key 之前
RESTIC_CACHE_DIR               缓存的目录
RESTIC_PROGRESS_FPS            进度栏更新的帧数

AWS_ACCESS_KEY_ID              Amazon S3 access key ID
AWS_SECRET_ACCESS_KEY          Amazon S3 secret access key
AWS_DEFAULT_REGION             Amazon S3 default region

ST_AUTH                        Auth URL for keystone v1 authentication
ST_USER                        Username for keystone v1 authentication
ST_KEY                         Password for keystone v1 authentication

OS_AUTH_URL                    Auth URL for keystone authentication
OS_REGION_NAME                 Region name for keystone authentication
OS_USERNAME                    Username for keystone authentication
OS_PASSWORD                    Password for keystone authentication
OS_TENANT_ID                   Tenant ID for keystone v2 authentication
OS_TENANT_NAME                 Tenant name for keystone v2 authentication

OS_USER_DOMAIN_NAME            User domain name for keystone authentication
OS_PROJECT_NAME                Project name for keystone authentication
OS_PROJECT_DOMAIN_NAME         Project domain name for keystone authentication

OS_APPLICATION_CREDENTIAL_ID       Application Credential ID (keystone v3)
OS_APPLICATION_CREDENTIAL_NAME     Application Credential Name (keystone v3)
OS_APPLICATION_CREDENTIAL_SECRET   Application Credential Secret (keystone v3)

OS_STORAGE_URL                 Storage URL for token authentication
OS_AUTH_TOKEN                  Auth token for token authentication

B2_ACCOUNT_ID                  Account ID or applicationKeyId for Backblaze B2
B2_ACCOUNT_KEY                 Account Key or applicationKey for Backblaze B2
```

```
AZURE_ACCOUNT_NAME              Account name for Azure
AZURE_ACCOUNT_KEY               Account key for Azure

GOOGLE_PROJECT_ID               Project ID for Google Cloud Storage
GOOGLE_APPLICATION_CREDENTIALS  Application Credentials for Google Cloud Storage (e
g. $HOME/.config/gs-secret-restic-key.json)

RCLONE_BWLIMIT                  rclone 带宽限制
```

作为 restic 特定的环境变量的补充，下述系统级的环境变量也被用作各种操作：

- $XDG_CACHE_HOME/restic,$HOME/.cache/restic:缓存.
- $TMPDIR:临时文件.
- $PATH/fusermount:restic mount 的二进制文件

退出状态码

在备份命令运行退出后，restic 会给出下述退出状态码：

- 0 备份成功时
- 1 有致使错误时（此时快照不会被创建）
- 3 当有输入文件无法被读取时 (剩下的文件会成功创建快照)

使用储存库

列出所有快照

现在你可以列出储存库中所有的快照：

```
$ restic -r /srv/restic-repo snapshots
enter password for repository:
ID        Date                Host    Tags   Directory
----------------------------------------------------------------------
40dc1520  2015-05-08 21:38:30  kasimir        /home/user/work
79766175  2015-05-08 21:40:19  kasimir        /home/user/work
bdbd3439  2015-05-08 21:45:17  luigi          /home/art
590c8fc8  2015-05-08 21:47:38  kazik          /srv
9f0bc19e  2015-05-08 21:46:11  luigi          /srv
```

你可以通过列出文件夹路径来过滤快照：

```
$ restic -r /srv/restic-repo snapshots --path="/srv"
enter password for repository:
ID        Date                Host    Tags   Directory
----------------------------------------------------------------------
590c8fc8  2015-05-08 21:47:38  kazik          /srv
9f0bc19e  2015-05-08 21:46:11  luigi          /srv
```

或者通过保存所在的 host 来过滤：

```
$ restic -r /srv/restic-repo snapshots --host luigi
```

```
enter password for repository:
ID        Date                Host    Tags   Directory
----------------------------------------------------------------------
bdbd3439  2015-05-08 21:45:17  luigi          /home/art
9f0bc19e  2015-05-08 21:46:11  luigi          /srv
```

结合使用过滤也是可行的

进一步，你可以为名字滤镜的输出编组（通过host, paths, tags）：

```
$ restic -r /srv/restic-repo snapshots --group-by host

enter password for repository:
snapshots for (host [kasimir])
ID        Date                Host    Tags   Directory
----------------------------------------------------------------------
40dc1520  2015-05-08 21:38:30  kasimir        /home/user/work
79766175  2015-05-08 21:40:19  kasimir        /home/user/work
2 snapshots
snapshots for (host [luigi])
ID        Date                Host    Tags   Directory
----------------------------------------------------------------------
bdbd3439  2015-05-08 21:45:17  luigi          /home/art
9f0bc19e  2015-05-08 21:46:11  luigi          /srv
2 snapshots
snapshots for (host [kazik])
ID        Date                Host    Tags   Directory
----------------------------------------------------------------------
590c8fc8  2015-05-08 21:47:38  kazik          /srv
1 snapshots
```

在储存库之间复制快照

如果你想在储存库之间复制快照，例如要从本地复制到一个远程储存库，你可以使用 copy 命令：

```
$ restic -r /srv/restic-repo copy --repo2 /srv/restic-repo-copy
repository d6504c63 opened successfully, password is correct
repository 3dd0878c opened successfully, password is correct

snapshot 410b18a2 of [/home/user/work] at 2020-06-09 23:15:57.305305 +0200 CEST)
  copy started, this may take a while...
snapshot 7a746a07 saved

snapshot 4e5d5487 of [/home/user/work] at 2020-05-01 22:44:07.012113 +0200 CEST)
skipping snapshot 4e5d5487, was already copied to snapshot 50eb62b7
```

这个命令例子从源储存库 /srv/restic-repo 复制所有的快照到目标储存库 /srv/restic-repo-copy，前已经复制过的储存库会在运行时跳过。

注意

由于源和目标储存库的加密密码不一样，这个进程会需要上传和下载整个快照。另外，传输的文件不被重新分块，重新分块会在已经存储在目标储存库和源储存库的文件之间破坏反重复性。看下节以避这个问题。

对于目标储存库 --repo2 ，可以从 --password-file2 指定的文件中读取它的密码，或者使用 --password-command2 选项。可选的，也可以使用环境变量 $RESTIC_PASSWORD_COMMAND2 和 $RESTIC_PASSWORD_FILE2 。也可以通过 $RESTIC_PASSWORD2 变量直接传递密码。应该用于解密密码可以被通过使用 --key-hint2 传递它的 ID 或者使用环境变量 $RESTIC_KEY_HINT2 来选择。

当源和目标储存库都使用相同的后端时，设置这个后端的选项和环境变量会同时作用于两个储存库。如可以为源和目标两个库指定不同的账户。你可以通过使用在 rclone 中配置的远程路径后端避免这限制。

要复制的快照可以被通过 host、路径、逗号分割的标签列过滤：

```
$ restic -r /srv/restic-repo copy --repo2 /srv/restic-repo-copy --host luigi --path /srv --tag foo bar
```

也可以直接指定要复制哪几个快照：

```
$ restic -r /srv/restic-repo copy --repo2 /srv/restic-repo-copy 410b18a2 4e5d5487 latest
```

保存复制快照时的反重复

虽然复制命令在两个储存库之间传输快照，但反重复方法在不同的储存库的快照间是不可用的。为了保正确的反重复，两个储存库都需要使用相同的参数用于分割大文件为小的块，也就要求额外的设置骤。当有相同的参数时，restic 就会为两个库分割出相同的块，也就能使用反重复了。

块参数在创建储存库时就生成了。这是在创建远程储存库时要求使用与已存在储存库相同块大小的命：

```
$ restic -r /srv/restic-repo-copy init --repo2 /srv/restic-repo --copy-chunker-params
```

restic 无法改变一个已存在的储存库的块大小。

检查储存库的完整和连续性

想象你的储存库保存在一个硬盘有问题的服务器上，或者更糟糕，攻击者有了访问你的储存库的权限还想让你恢复恶意数据：

```
$ echo "boom" > /srv/restic-repo/index/de30f3231ca2e6a59af4aa84216dfe2ef7339c549dc1109b84000997b139628
```

要恢复带有以上修改的快照会产生错误：

```
$ restic -r /srv/restic-repo --no-cache restore c23e491f --target /tmp/restore-work
...
Fatal: unable to load index de30f323: load <index/de30f3231c>: invalid data returned
```

为了在发生问题前检测这些毛病，日常运行 check 命令测试储存库是否完好是有必要的。有两种检查做：

- 结构连续性和完整性，例如快照、树、块文件（默认）
- 实际数据的完整性（要用参数启用，见下文）

使用 check 命令检查储存库的完整性，如果像上面那样发现了损坏，check 会检测到并发出错误：

```
$ restic -r /srv/restic-repo check
...
load indexes
error: error loading index de30f323: load <index/de30f3231c>: invalid data returned
Fatal: LoadIndex returned errors
```

这是正常的输出：

```
$ restic -r /src/restic-repo check
...
load indexes
check all packs
check snapshots, trees and blobs
no errors were found
```

默认 check 选项不会检查实际每个未被修改的包的完整性，因为那样需要读取储存库中的每一个包让 restic 验证实际包的完整性，使用 --read-data 选项：

```
$ restic -r /srv/restic-repo check --read-data
...
load indexes
check all packs
check snapshots, trees and blobs
read all data
[0:00] 100.00%  3 / 3 items
duration: 0:00
no errors were found
```

你也可以使用 --read-data-subset=n/t 参数，检查所有数据了子集。当命令运行时，所有的包会被成 t 组，只有属于组序号 n 的块会被检查，例如，以下命令会在检查所有储存库的文件时调用 5 分组：

```
$ restic -r /srv/restic-repo check --read-data-subset=1/5
$ restic -r /srv/restic-repo check --read-data-subset=2/5
$ restic -r /srv/restic-repo check --read-data-subset=3/5
$ restic -r /srv/restic-repo check --read-data-subset=4/5
$ restic -r /srv/restic-repo check --read-data-subset=5/5
```

从备份恢复

从快照恢复

用下面的命令可以将最后一份快照的内容恢复到 /tmp/restore-work：

```
$ restic -r /srv/restic-repo restore 79766175 --target /tmp/restore-work
enter password for repository:
restoring <Snapshot of [/home/user/work] at 2015-05-08 21:40:19.884408621 +0200 CEST>
o /tmp/restore-work
```

使用 latest可以恢复最后一个快照，你也可以结合 latest 和 --host 和 --path 过滤器选择指定 host 路径的最后一个快照：

```
$ restic -r /srv/restic-repo restore latest --target /tmp/restore-art --path "/home/art" --host lu
gi
```

```
enter password for repository:
restoring <Snapshot of [/home/art] at 2015-05-08 21:45:17.884408621 +0200 CEST> to /tmp
restore-art
```

使用 --exclude 和 --include 来限制恢复的快照的子文件，例如，只恢复一个文件：

```
$ restic -r /srv/restic-repo restore 79766175 --target /tmp/restore-work --include /work/foo
enter password for repository:
restoring <Snapshot of [/home/user/work] at 2015-05-08 21:40:19.884408621 +0200 CEST>
o /tmp/restore-work
```

这就只会将 foo 恢复到 /tmp/restore-work/work/foo.

你可以使用命令 restic ls latest 或 restic find foo 来查找在快照中指向这个文件的路径，你可以将这路径传递给 --include 以只恢复这一个文件。

对 --exclude and --include 还有忽略大小写的 --iexclude and --iinclude。

从挂载恢复

用下述命令可以将储存库以 FUSE 的形式挂载:

```
$ mkdir /mnt/restic
$ restic -r /srv/restic-repo mount /mnt/restic
enter password for repository:
Now serving /srv/restic-repo at /mnt/restic
When finished, quit with Ctrl-c or umount the mountpoint.
```

在 OpenBSD, Solaris/illumos and Windows 上通过 FUSE 挂载储存库是不可用的。在 Linux 上，要安装 fuse 内核模块。在 FreeBSD 上，你需要安装和启用这个内核模块(kldload fuse)

通过挂载恢复，能否恢复硬链接，取决于你使用的程序， rsync 的 –hard-links 选项可以恢复硬链接。

将文件打印到标准输出

例子：

```
$ restic -r /srv/restic-repo dump latest production.sql | mysql
```

如果你在一个储存库中有很多不同的事情要做， latest 快照可能就不是你想用的。比如：在储存库中如下快照：

```
$ restic -r /srv/restic-repo snapshots
ID        Date                Host        Tags        Directory
----------------------------------------------------------------------
562bfc5e  2018-07-14 20:18:01  mopped                  /home/user/file1
bbacb625  2018-07-14 20:18:07  mopped                  /home/other/work
e922c858  2018-07-14 20:18:10  mopped                  /home/other/work
098db9d5  2018-07-14 20:18:13  mopped                  /production.sql
b62f46ec  2018-07-14 20:18:16  mopped                  /home/user/file1
1541acae  2018-07-14 20:18:18  mopped                  /home/other/work
----------------------------------------------------------------------
```

这里，restic 将会把 latest 指向 1541acae 快照，后者不包含我们要打印的文件(production.sql)，

是，我们可以传递快照 ID：

```
$ restic -r /srv/restic-repo dump 098db9d5 production.sql | mysql
```

或者可以传递一个用于选择最后快照的路径，这个路径必须匹配打印在 Directory 一栏的路径：

```
$ restic -r /srv/restic-repo dump --path /production.sql latest production.sql | mysql
```

你也可以 dump 一整个文件夹内的结构到 stdout，为了保留文件和文件夹的信息，restic 会以 tar 格式输出：

```
$ restic -r /srv/restic-repo dump latest /home/other/work > restore.tar
```

移除备份快照

所有的备份空间都是有限的，所以 restic 允许移除旧的快照。可以手动（指定 ID）或者使用一定的则移除。对于移除操作，两个命令需要按序调用： forget 用于移除快照， prune 用于实际移除快照引用的文件。实际只要在使用 forget 时加上 --prune 选项即可。

警告：

Prune 快照可能非常费时，需要几乎与备份相同的时间。在这个操作期间，整个索引会被锁定，无法份。提升这方面的性能的计划还在规划中。

建议在 prune 完后运行一下 restic check 以确保储存库的完整性。

移除单个快照

snapshots 命令可以列出指定储存库中的快照：

```
$ restic -r /srv/restic-repo snapshots
enter password for repository:
ID        Date                Host      Tags  Directory
----------------------------------------------------------------------
40dc1520  2015-05-08 21:38:30  kasimir        /home/user/work
79766175  2015-05-08 21:40:19  kasimir        /home/user/work
bdbd3439  2015-05-08 21:45:17  luigi          /home/art
590c8fc8  2015-05-08 21:47:38  kazik          /srv
9f0bc19e  2015-05-08 21:46:11  luigi          /srv
```

要移除 /home/art 的快照，使用 forget 命令，并指定 IDC

```
$ restic -r /srv/restic-repo forget bdbd3439
enter password for repository:
removed snapshot d3f01f63
```

然后这个快照就被移除了：

```
$ restic -r /srv/restic-repo snapshots
enter password for repository:
ID        Date                Host      Tags  Directory
----------------------------------------------------------------------
40dc1520  2015-05-08 21:38:30  kasimir        /home/user/work
79766175  2015-05-08 21:40:19  kasimir        /home/user/work
```

```
590c8fc8  2015-05-08 21:47:38  kazik          /srv
9f0bc19e  2015-05-08 21:46:11  luigi          /srv
```

但是他索引的文件还会被保存，如果要清除这些未被索引的数据，就需要使用 prune命令：

```
$ restic -r /srv/restic-repo prune
enter password for repository:

counting files in repo
building new index for repo
[0:00] 100.00%  22 / 22 files
repository contains 22 packs (8512 blobs) with 100.092 MiB bytes
processed 8512 blobs: 0 duplicate blobs, 0B duplicate
load all snapshots
find data that is still in use for 1 snapshots
[0:00] 100.00%  1 / 1 snapshots
found 8433 of 8512 data blobs still in use
will rewrite 3 packs
creating new index
[0:00] 86.36%  19 / 22 files
saved new index as 544a5084
done
```

然后储存库就小点了。

你也可以这样一步完成：

```
$ restic forget --keep-last 1 --prune
snapshots for host mopped, directories /home/user/work:

keep 1 snapshots:
ID        Date              Host       Tags       Directory
----------------------------------------------------------------------
4bba301e  2017-02-21 10:49:18  mopped                /home/user/work

remove 1 snapshots:
ID        Date              Host       Tags       Directory
----------------------------------------------------------------------
8c02b94b  2017-02-21 10:48:33  mopped                /home/user/work

1 snapshots have been removed, running prune
counting files in repo
building new index for repo
[0:00] 100.00%  37 / 37 packs
repository contains 37 packs (5521 blobs) with 151.012 MiB bytes
processed 5521 blobs: 0 duplicate blobs, 0B duplicate
load all snapshots
find data that is still in use for 1 snapshots
[0:00] 100.00%  1 / 1 snapshots
found 5323 of 5521 data blobs still in use, removing 198 blobs
will delete 0 packs and rewrite 27 packs, this frees 22.106 MiB
creating new index
[0:00] 100.00%  30 / 30 packs
saved new index as b49f3e68
done
```

以一定的规则移除快照

手动移除快照是非常无聊并且容易出错的。所以 restic 允许按照指定规则自动移除快照。你可以指定少小时，多少天，多少星期，几个月，几年以内的快照可以被保留，把其他快照删掉。最重要的参数是 `--dry-run` ，这个选项可以让 restic 不要删掉任何快照，而是把会被删除的快照打印出来

当带有规则运行 `forget` 时，restic 会载入所有快照，以 host name 和文件夹列表编组，也可以使用 `--group-by` 设置编组依据，例如 `--group-by paths,tags` 。然后规则会被分别应用到每个快照组上。是安全特性。

`forget` 命令接受如下参数：

- `--keep-last n` 永远不要删除最后n个快照
- `--keep-hourly n` 对，最后n个小时内创建的快照，只保存每个小时内最后一个快照
- `--keep-daily n` 最后n天内创建的快照，只保存每天内最后一个快照
- `--keep-weekly n` 最后n周内创建的快照，只保存每周内最后一个快照
- `--keep-monthly n` 最后n月内创建的快照，只保存每月内最后一个快照
- `--keep-yearly n` 最后n年内创建的快照，只保存每年内最后一个快照
- `--keep-tag` 只保留具有该选项指定的标签的快照（可以重复多次，以指定多个快照）
- `--keep-within duration` 只保留距离最后一个快照指定时间内创建的快照，`duration` 需要是年月小时的数字，例如：`2y5m7d3h` 会保留距离最后一个快照两年，5个月，7天三小时之内创建的所有照

多个规则，则取其并集，保留尽可能多的快照。

你也可以通过 `--host` `--tag` 指定拥有指定 host name、tag 的快照。当指定了多个快照时，只有同拥有这些标签的快照会被删除。例如：

```
$ restic forget --tag foo --keep-last 1

$ restic forget --tag foo --tag bar --keep-last 1

$ restic forget --tag foo,bar --keep-last 1
```

所有上述 `--keep-*` 选项只会计数拥有快照的 小时/天/星期/月/年 ，那些没有快照的时间不会被计数。

为了安全，restic 会拒绝执行清空的规则，例如，如果指定 `--keep-last 0` 会 forget 所有快照，restic 会回应说不会有快照被移除。要删除这些快照，使用 `--keep-last` 再使用 ID 手动删除最后一个快照。

除了匹配 `--keep-*` 数的快照，其它快照都会被计数。

All snapshots are evaluated against all matching `--keep-*` counts. A single snapshot on 2017-9-30 (Sat) will count as a daily, weekly and monthly.

Let’s explain this with an example: Suppose you have only made a backup on each Sunday f r 12 weeks:

```
$ restic snapshots
repository f00c6e2a opened successfully, password is correct
ID        Time                 Host        Tags        Paths
----------------------------------------------------------------
```

```
0a1f9759  2019-09-01 11:00:00  mopped              /home/user/work
46cfe4d5  2019-09-08 11:00:00  mopped              /home/user/work
f6b1f037  2019-09-15 11:00:00  mopped              /home/user/work
eb430a5d  2019-09-22 11:00:00  mopped              /home/user/work
8cf1cb9a  2019-09-29 11:00:00  mopped              /home/user/work
5d33b116  2019-10-06 11:00:00  mopped              /home/user/work
b9553125  2019-10-13 11:00:00  mopped              /home/user/work
e1a7b58b  2019-10-20 11:00:00  mopped              /home/user/work
8f8018c0  2019-10-27 11:00:00  mopped              /home/user/work
59403279  2019-11-03 11:00:00  mopped              /home/user/work
dfee9fb4  2019-11-10 11:00:00  mopped              /home/user/work
e1ae2f40  2019-11-17 11:00:00  mopped              /home/user/work
----------------------------------------------------------------
12 snapshots
```

Then forget --keep-daily 4 will keep the last four snapshots for the last four Sundays, but rem ve the rest:

```
$ restic forget --keep-daily 4 --dry-run
repository f00c6e2a opened successfully, password is correct
Applying Policy: keep the last 4 daily snapshots
keep 4 snapshots:
ID        Time                 Host       Tags       Reasons        Paths
-----------------------------------------------------------------------------
8f8018c0  2019-10-27 11:00:00  mopped                daily snapshot  /home/user/work
59403279  2019-11-03 11:00:00  mopped                daily snapshot  /home/user/work
dfee9fb4  2019-11-10 11:00:00  mopped                daily snapshot  /home/user/work
e1ae2f40  2019-11-17 11:00:00  mopped                daily snapshot  /home/user/work
-----------------------------------------------------------------------------
4 snapshots

remove 8 snapshots:
ID        Time                 Host       Tags       Paths
----------------------------------------------------------------
0a1f9759  2019-09-01 11:00:00  mopped              /home/user/work
46cfe4d5  2019-09-08 11:00:00  mopped              /home/user/work
f6b1f037  2019-09-15 11:00:00  mopped              /home/user/work
eb430a5d  2019-09-22 11:00:00  mopped              /home/user/work
8cf1cb9a  2019-09-29 11:00:00  mopped              /home/user/work
5d33b116  2019-10-06 11:00:00  mopped              /home/user/work
b9553125  2019-10-13 11:00:00  mopped              /home/user/work
e1a7b58b  2019-10-20 11:00:00  mopped              /home/user/work
----------------------------------------------------------------
8 snapshots
```

另一个例子：假设你已经每天备份保持了100年， forget --keep-daily 7 --keep-weekly 5 --keep-onthly 12 --keep-yearly 75 会保留最近 7 天的快照，然后是 4 个（加上之前的七天就是 5 周）每最后一天、11 个每月最后一天、75 个每年最后一天的快照。其他快照都会被移除。

加密

“或许这个设计不是完美的，但他是优秀的。加密是第一类的特性，实施看起来正常，并且我猜反重复代价是值得的，所以，我会使用 restic 用做个人备份。” Filippo Valsorda

管理加密库密码

key 命令允许你设置多个访问密码，你可以使用 list、 add、 remove、 passwd 子命令精细地管理码：

```
$ restic -r /srv/restic-repo key list
enter password for repository:
 ID        User       Host       Created
----------------------------------------------------------------------
*eb78040b    username    kasimir   2015-08-12 13:29:57

$ restic -r /srv/restic-repo key add
enter password for repository:
enter password for new key:
enter password again:
saved new key as <Key of username@kasimir, created on 2015-08-12 13:35:05.316831933 +0
00 CEST>

$ restic -r /srv/restic-repo key list
enter password for repository:
 ID        User       Host       Created
----------------------------------------------------------------------
 5c657874    username    kasimir   2015-08-12 13:35:05
*eb78040b    username    kasimir   2015-08-12 13:29:57
```

脚本

略

例子

略

参与

略

参考

略

交流

略

FAQ

略

手册

使用帮助

```
$ ./restic --help
restic is a backup program which allows saving multiple revisions of files and
directories in an encrypted repository stored on different backends.

用法:
  restic [command]

可用命令:
  backup        创建一个文件或文件夹的备份
  cache         操作本地缓存夹
  cat           打印内部对象到 stdout
  check         检查储存库的错误
  copy          从一个储存库复制快照到另一个
  diff          展示两个储存库的不同
  dump          打印一个已备份的文件到 stdout
  find          找到一个文件、目录、restic ID
  forget        从储存库移除快照
  generate      生成手册页，自动完成文件 (bash, zsh)
  help          显示帮助
  init          初始化新储存库
  key           管理密码 (passwords)
  list          列出储存库内的对象
  ls            列出快照内的文件
  migrate       使用迁移
  mount         挂载储存库
  prune         从储存库移除不需要的数据
  rebuild-index 新建一个新的索引文件
  recover       从储存库恢复数据
  restore       从快照提取数据
  self-update   升级 restic 二进制文件
  snapshots     列出所有快照
  stats         扫描存储库并显示基本统计信息
  tag           修改快照的标签
  unlock        移除其它进程创建的锁
  version       打印版本信息

flags:
      --cacert file              用于加载 root 验证的文件（使用系统验证）
      --cache-dir directory      设置缓存文件夹. (默认: 使用系统默认缓存路径)
      --cleanup-cache            自动清除旧的缓存目录
  -h, --help                     显示帮助
      --json                     输出模式改为 json
      --key-hint key             尝试解码的第一个 key 的 key ID (默认: $RESTIC_KEY_HINT)
      --limit-download int       下载最大带宽，单位 KiB/s. (默认: 无限制)
      --limit-upload int         下载最大带宽，单位 KiB/s. (默认: 无限制)
      --no-cache                 不使用本地缓存
      --no-lock                  不锁定储存库，这允许一些只读储存库的操作
  -o, --option key=value         设置扩展选项 (key=value, 可以被使用多次)
      --password-command command  抓取储存库密码的 shell 命令 (默认: $RESTIC_PASSWORD
COMMAND)
  -p, --password-file file       得到储存库密码的文件 (默认: $RESTIC_PASSWORD_FILE)
  -q, --quiet                    不要输出进度报告
```

```
  -r, --repo repository          要备份到/自的储存库 (default: $RESTIC_REPOSITORY)
      --repository-file file     得到储存库路径所需读取的文件 (默认: $RESTIC_REPOSITORY_FILE)
      --tls-client-cert file     得到 PEM 编码的 TLS 客户端验证和私钥所需读取的文件
  -v, --verbose n                verbose 等级 (specify --verbose multiple times or level --verbose=
)

使用 "restic [command] --help" 以得到命令更详细的帮助。

$ ./restic backup --help
The "backup" command creates a new snapshot and saves the files and directories
given as the arguments.

EXIT STATUS
===========

Exit status is 0 if the command was successful.
Exit status is 1 if there was a fatal error (no snapshot created).
Exit status is 3 if some source data could not be read (incomplete snapshot created).

Usage:
  restic backup [flags] FILE/DIR [FILE/DIR] ...

Flags:
  -e, --exclude pattern                  exclude a pattern (can be specified multiple times)
      --exclude-caches                   excludes cache directories that are marked with a CACHE
IR.TAG file. See https://bford.info/cachedir/ for the Cache Directory Tagging Standard
      --exclude-file file                read exclude patterns from a file (can be specified multiple
imes)
      --exclude-if-present filename[:header]   takes filename[:header], exclude contents of direc
ories containing filename (except filename itself) if header of that file is as provided (can be s
ecified multiple times)
      --exclude-larger-than size         max size of the files to be backed up (allowed suffixes:
k/K, m/M, g/G, t/T)
      --files-from file                  read the files to backup from file (can be combined with file
args/can be specified multiple times)
  -f, --force                            force re-reading the target files/directories (overrides the "pa
ent" flag)
  -h, --help                             help for backup
  -H, --host hostname                    set the hostname for the snapshot manually. To prevent
an expensive rescan use the "parent" flag
      --iexclude pattern                 same as --exclude pattern but ignores the casing of filen
mes
      --iexclude-file file               same as --exclude-file but ignores casing of filenames in pa
terns
      --ignore-inode                     ignore inode number changes when checking for modified
files
  -x, --one-file-system                  exclude other file systems
      --parent snapshot                  use this parent snapshot (default: last snapshot in the re
o that has the same target files/directories)
      --stdin                            read backup from stdin
      --stdin-filename filename          filename to use when reading from stdin (default "stdi
")
      --tag tag                          add a tag for the new snapshot (can be specified multiple ti
es)
```

```
      --time time                          time of the backup (ex. '2012-11-01 22:08:41') (default: now)
      --with-atime                         store the atime for all files and directories

Global Flags:
      --cacert file              file to load root certificates from (default: use system certificates)
      --cache-dir directory        set the cache directory. (default: use system default cache direct
ry)
      --cleanup-cache            auto remove old cache directories
      --json                   set output mode to JSON for commands that support it
      --key-hint key             key ID of key to try decrypting first (default: $RESTIC_KEY_HINT)
      --limit-download int         limits downloads to a maximum rate in KiB/s. (default: unlimited

      --limit-upload int          limits uploads to a maximum rate in KiB/s. (default: unlimited)
      --no-cache                 do not use a local cache
      --no-lock                  do not lock the repo, this allows some operations on read-only repo

  -o, --option key=value          set extended option (key=value, can be specified multiple time
)
      --password-command command   shell command to obtain the repository password from
(default: $RESTIC_PASSWORD_COMMAND)
  -p, --password-file file         file to read the repository password from (default: $RESTIC_PAS
WORD_FILE)
  -q, --quiet                    do not output comprehensive progress report
  -r, --repo repository           repository to backup to or restore from (default: $RESTIC_REPOS
TORY)
      --repository-file file       file to read the repository location from (default: $RESTIC_REPOSI
ORY_FILE)
      --tls-client-cert file       path to a file containing PEM encoded TLS client certificate and pri
ate key
  -v, --verbose n                 be verbose (specify --verbose multiple times or level --verbose=n)
```

Subcommand that support showing progress information such as backup, check and prune wi l do so unless the quiet flag -q or --quiet is set. When running from a non-interactive console progress reporting will be limited to once every 10 seconds to not fill your logs. Use backup w th the quiet flag -q or --quiet to skip the initial scan of the source directory, this may shorten he backup time needed for large directories.

Additionally on Unix systems if restic receives a SIGUSR1 signal the current progress will be wr tten to the standard output so you can check up on the status at will.

管理标签

Managing tags on snapshots is done with the tag command. The existing set of tags can be r placed completely, tags can be added or removed. The result is directly visible in the snapsho s command.

Let’s say we want to tag snapshot 590c8fc8 with the tags NL and CH and remove all other t gs that may be present, the following command does that:

```
$ restic -r /srv/restic-repo tag --set NL --set CH 590c8fc8
create exclusive lock for repository
modified tags on 1 snapshots
```

Note the snapshot ID has changed, so between each change we need to look up the new ID o

the snapshot. But there is an even better way, the tag command accepts --tag for a filter, so e can filter snapshots based on the tag we just added.

So we can add and remove tags incrementally like this:

```
$ restic -r /srv/restic-repo tag --tag NL --remove CH
create exclusive lock for repository
modified tags on 1 snapshots

$ restic -r /srv/restic-repo tag --tag NL --add UK
create exclusive lock for repository
modified tags on 1 snapshots

$ restic -r /srv/restic-repo tag --tag NL --remove NL
create exclusive lock for repository
modified tags on 1 snapshots

$ restic -r /srv/restic-repo tag --tag NL --add SOMETHING
no snapshots were modified
```

底层

浏览储存库对象

Internally, a repository stores data of several different types described in the design documen ation. You can list objects such as blobs, packs, index, snapshots, keys or locks with the follow ng command:

```
$ restic -r /srv/restic-repo list snapshots
d369ccc7d126594950bf74f0a348d5d98d9e99f3215082eb69bf02dc9b3e464c
```

The find command searches for a given pattern in the repository.

```
$ restic -r backup find test.txt
debug log file restic.log
debug enabled
enter password for repository:
found 1 matching entries in snapshot 196bc5760c909a7681647949e80e5448e276521489558
25680acf1bd428af36
  -rw-r--r--   501    20      5 2015-08-26 14:09:57 +0200 CEST path/to/test.txt
```

The cat command allows you to display the JSON representation of the objects or their raw c ntent.

```
$ restic -r /srv/restic-repo cat snapshot d369ccc7d126594950bf74f0a348d5d98d9e99f321508
eb69bf02dc9b3e464c
enter password for repository:
{
  "time": "2015-08-12T12:52:44.091448856+02:00",
  "tree": "05cec17e8d3349f402576d02576a2971fc0d9f9776ce2f441c7010849c4ff5af",
  "paths": [
    "/home/user/work"
  ],
```

```
  "hostname": "kasimir",
  "username": "username",
  "uid": 501,
  "gid": 20
}
```

Metadata handling

Restic saves and restores most default attributes, including extended attributes like ACLs. Spa se files are not handled in a special way yet, and aren’ t restored.

The following metadata is handled by restic:

- Name
- Type
- Mode
- ModTime
- AccessTime
- ChangeTime
- UID
- GID
- User
- Group
- Inode
- Size
- Links
- LinkTarget
- Device
- Content
- Subtree
- ExtendedAttributes

Getting information about repository data

Use the stats command to count up stats about the data in the repository. There are different counting modes available using the --mode flag, depending on what you want to calculate. T e default is the restore size, or the size required to restore the files:

- restore-size (default) counts the size of the restored files.
- files-by-contents counts the total size of unique files as given by their contents. This can be useful since a file is considered unique only if it has unique contents. Keep in mind that a smal change to a large file (even when the file name/path hasn’ t changed) will cause them to loo like different files, thus essentially causing the whole size of the file to be counted twice.
- raw-data counts the size of the blobs in the repository, regardless of how many files referen

e them. This tells you how much restic has reduced all your original data down to (either for a single snapshot or across all your backups), and compared to the size given by the restore-siz mode, can tell you how much deduplication is helping you.

- blobs-per-file is kind of a mix between files-by-contents and raw-data modes; it is useful for knowing how much value your backup is providing you in terms of unique data stored by file. Like files-by-contents, it is resilient to file renames/moves. Unlike files-by-contents, it does no balloon to high values when large files have small edits, as long as the file path stayed the s me. Unlike raw-data, this mode DOES consider how many files point to each blob such that t e more files a blob is referenced by, the more it counts toward the size.

For example, to calculate how much space would be required to restore the latest snapshot (f om any host that made it):

```
$ restic stats latest
password is correct
Total File Count:   10538
      Total Size:   37.824 GiB
```

If multiple hosts are backing up to the repository, the latest snapshot may not be the one you want. You can specify the latest snapshot from only a specific host by using the --host flag:

```
$ restic stats --host myserver latest
password is correct
Total File Count:   21766
      Total Size:   481.783 GiB
```

There we see that it would take 482 GiB of disk space to restore the latest snapshot from "m server".

In case you have multiple backups running from the same host so can also use --tag and --pa h to be more specific about which snapshots you are looking for.

But how much space does that snapshot take on disk? In other words, how much has restic's deduplication helped? We can check:

```
$ restic stats --host myserver --mode raw-data latest
password is correct
Total Blob Count:   340847
      Total Size:   458.663 GiB
```

Comparing this size to the previous command, we see that restic has saved about 23 GiB of s ace with deduplication.

Which mode you use depends on your exact use case. Some modes are more useful across all snapshots, while others make more sense on just a single snapshot, depending on what you re trying to calculate.

脚本

Restic supports the output of some commands in JSON format, the JSON data can then be pr cessed by other programs (e.g. jq). The following example lists all snapshots as JSON and uses q to pretty-print the result:

```
$ restic -r /srv/restic-repo snapshots --json | jq .
[
  {
    "time": "2017-03-11T09:57:43.26630619+01:00",
    "tree": "bf25241679533df554fc0fd0ae6dbb9dcf1859a13f2bc9dd4543c354eff6c464",
    "paths": [
      "/home/work/doc"
    ],
    "hostname": "kasimir",
    "username": "fd0",
    "uid": 1000,
    "gid": 100,
    "id": "bbeed6d28159aa384d1ccc6fa0b540644b1b9599b162d2972acda86b1b80f89e"
  },
  {
    "time": "2017-03-11T09:58:57.541446938+01:00",
    "tree": "7f8c95d3420baaac28dc51609796ae0e0ecfb4862b609a9f38ffaf7ae2d758da",
    "paths": [
      "/home/user/shared"
    ],
    "hostname": "kasimir",
    "username": "fd0",
    "uid": 1000,
    "gid": 100,
    "id": "b157d91c16f0ba56801ece3a708dfc53791fe2a97e827090d6ed9a69a6ebdca0"
  }
]
```

临时文件

During some operations (e.g. backup and prune) restic uses temporary files to store data. The e files will, by default, be saved to the system’ s temporary directory, on Linux this is usually l cated in /tmp/. The environment variable TMPDIR can be used to specify a different directory, e.g. to use the directory /var/tmp/restic-tmp instead of the default, set the environment varia le like this:

```
$ export TMPDIR=/var/tmp/restic-tmp
$ restic -r /srv/restic-repo backup ~/work
```

缓存

Restic keeps a cache with some files from the repository on the local machine. This allows fas er operations, since meta data does not need to be loaded from a remote repository. The cac e is automatically created, usually in an OS-specific cache folder:

- Linux/other: ~/.cache/restic (or$XDG_CACHE_HOME/restic)
- macOS: ~/Library/Caches/restic
- Windows: %LOCALAPPDATA%/restic

The command line parameter --cache-dir or the environment variable $RESTIC_CACHE_DIR c n be used to override the default cache location. The parameter --no-cache disables the cache entirely. In this case, all data is loaded from the repo.

The cache is ephemeral: When a file cannot be read from the cache, it is loaded from the repo itory.

Within the cache directory, there’ s a sub directory for each repository the cache was used wi h. Restic updates the timestamps of a repo directory each time it is used, so by looking at the imestamps of the sub directories of the cache directory it can decide which sub directories are old and probably not needed any more. You can either remove these directories manually, or un a restic command with the --cleanup-cache flag.

开发者信息

略