Restic 使用手册

Introduction

Restic is a backup program that is fast, efficient and secure. It supports the three major operating systems (Linux, macOS, Windows) and a few smaller ones (FreeBSD, OpenBSD).

> 参考资料：
>
> https://restic.readthedocs.io/en/latest/
>
> https://github.com/restic/restic

Installation

Installation

Packages

Note that if at any point the package you're trying to use is outdated, you always have the option to use an official binary from the restic project.

These are up to date binaries, built in a reproducible and verifiable way, that you can download and run without having to do additional installation work.

Please see the Official Binaries section below for various downloads. Official binaries can be updated in place by using the `restic self-update` command.

RHEL & CentOS

For RHEL / CentOS Stream 8 & 9 restic can be installed from the EPEL repository:

```
$ dnf install epel-release
$ dnf install restic
```

For RHEL7/CentOS there is a copr repository available, you can try the following:

```
$ yum install yum-plugin-copr
$ yum copr enable copart/restic
$ yum install restic
```

If that doesn't work, you can try adding the repository directly, for CentOS6 use:

```
$ yum-config-manager --add-repo
https://copr.fedorainfracloud.org/coprs/copart/restic/repo/epel-6/copart-restic-
epel-6.repo
```

For CentOS7 use:

```
$ yum-config-manager --add-repo
https://copr.fedorainfracloud.org/coprs/copart/restic/repo/epel-7/copart-restic-
epel-7.repo
```

Official Binaries

```
# Stable Releases
# https://github.com/restic/restic/releases/tag/v0.13.1
$ wget
https://github.com/restic/restic/releases/download/v0.13.1/restic_0.13.1_linux_a
md64.bz2

$ yum -y install bzip2
$ bzip2 -d restic_0.13.1_linux_amd64.bz2

$ chmod 777 restic_0.13.1_linux_amd64
$ cp restic_0.13.1_linux_amd64 /usr/local/bin/restic

$ restic

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

Usage:
  restic [command]

Available Commands:
  backup        Create a new backup of files and/or directories
  cache         Operate on local cache directories
  cat           Print internal objects to stdout
  check         Check the repository for errors
  copy          Copy snapshots from one repository to another
  diff          Show differences between two snapshots
  dump          Print a backed-up file to stdout
  find          Find a file, a directory or restic IDs
  forget        Remove snapshots from the repository
  generate      Generate manual pages and auto-completion files (bash, fish,
zsh)
  help          Help about any command
  init          Initialize a new repository
  key           Manage keys (passwords)
  list          List objects in the repository
  ls            List files in a snapshot
  migrate       Apply migrations
  mount         Mount the repository
  prune         Remove unneeded data from the repository
  rebuild-index Build a new index
  recover       Recover data from the repository not referenced by snapshots
```

```
  restore       Extract the data from a snapshot
  self-update   Update the restic binary
  snapshots     List all snapshots
  stats         Scan the repository and show basic statistics
  tag           Modify tags on snapshots
  unlock        Remove locks other processes created
  version       Print version information

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 directory)
      --cleanup-cache             auto remove old cache directories
  -h, --help                      help for restic
      --insecure-tls              skip TLS certificate verification when
connecting to the repo (insecure)
      --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 repository, this allows some
operations on read-only repositories
  -o, --option key=value          set extended option (key=value, can be
specified multiple times)
      --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_PASSWORD_FILE)
  -q, --quiet                     do not output comprehensive progress report
  -r, --repo repository           repository to backup to or restore from
(default: $RESTIC_REPOSITORY)
      --repository-file file      file to read the repository location from
(default: $RESTIC_REPOSITORY_FILE)
      --tls-client-cert file      path to a file containing PEM encoded TLS
client certificate and private key
  -v, --verbose n                 be verbose (specify multiple times or a level
using --verbose=n, max level/times is 3)

Use "restic [command] --help" for more information about a command.
```

> 参考资料:
>
> https://github.com/restic/restic/releases/tag/v0.13.1
>
> https://restic.readthedocs.io/en/latest/020_installation.html

Preparing a new repository

Preparing a new repository

The place where your backups will be saved is called a “repository”. This chapter explains how to create (“init”) such a repository. The repository can be stored locally, or on some remote server or service. We’ll first cover using a local repository; the remaining sections of this chapter cover all the other options. You can skip to the next chapter once you’ve read the relevant section here.

For automated backups, restic accepts the repository location in the environment variable `RESTIC_REPOSITORY`. Restic can also read the repository location from a file specified via the `--repository-file` option or the environment variable `RESTIC_REPOSITORY_FILE`. For the password, several options exist:

> - Setting the environment variable `RESTIC_PASSWORD`
> - Specifying the path to a file with the password via the option `--password-file` or the environment variable `RESTIC_PASSWORD_FILE`
> - Configuring a program to be called when the password is needed via the option `--password-command` or the environment variable `RESTIC_PASSWORD_COMMAND`
> - The `init` command has an option called `--repository-version` which can be used to explicitely set the version for the new repository. By default, the current stable version is used. Have a look at the design documentation for details.

Local

In order to create a repository at `/srv/restic-repo`, run the following command and enter the same password twice:

```
$ 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.
```

SFTP

In order to backup data via SFTP, you must first set up a server with SSH and let it know your public key. Passwordless login is important since automatic backups are not possible if the server prompts for credentials.

Once the server is configured, the setup of the SFTP repository can simply be achieved by changing the URL scheme in the `init` command:

```
$ restic -r sftp:user@host:/srv/restic-repo init
$ restic -r sftp:root@shc-sma-cd75.hpeswlab.net:/srv/restic-repo init
root@shc-sma-cd75.hpeswlab.net's password:
created restic repository b8369ceb87 at sftp:root@shc-sma-
cd75.hpeswlab.net:/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.

# back up
$ restic -r sftp:root@shc-sma-cd75.hpeswlab.net:/srv/restic-repo --verbose
backup /tzhong
```

Amazon S3

Restic can backup data to any Amazon S3 bucket. However, in this case, changing the URL scheme is not enough since Amazon uses special security credentials to sign HTTP requests. By consequence, you must first setup the following environment variables with the credentials you obtained while creating the bucket.

```
$ export AWS_ACCESS_KEY_ID=<MY_ACCESS_KEY>
$ export AWS_SECRET_ACCESS_KEY=<MY_SECRET_ACCESS_KEY>
```

You can then easily initialize a repository that uses your Amazon S3 as a backend. If the bucket does not exist it will be created in the default location:

```
$ restic -r s3:s3.amazonaws.com/bucket_name init
enter password for new repository:
enter password again:
created restic repository eefee03bbd at s3:s3.amazonaws.com/bucket_name
Please note that knowledge of your password is required to access the
repository.
Losing your password means that your data is irrecoverably lost.
```

If needed, you can manually specify the region to use by either setting the environment variable `AWS_DEFAULT_REGION` or calling restic with an option parameter like `-o s3.region="us-east-1"`. If the region is not specified, the default region is used. Afterwards, the S3 server (at least for AWS, `s3.amazonaws.com`) will redirect restic to the correct endpoint.

Until version 0.8.0, restic used a default prefix of `restic`, so the files in the bucket were placed in a directory named `restic`. If you want to access a repository created with an older version of restic, specify the path after the bucket name like this:

```
$ restic -r s3:s3.amazonaws.com/bucket_name/restic [...]
```

For an S3-compatible server that is not Amazon (like Minio, see below), or is only available via HTTP, you can specify the URL to the server like this: `s3:http://server:port/bucket_name`.

Note

restic expects path-style URLs like for example `s3.us-west-2.amazonaws.com/bucket_name`. Virtual-hosted–style URLs like `bucket_name.s3.us-west-2.amazonaws.com`, where the bucket name is part of the hostname are not supported. These must be converted to path-style URLs instead, for example `s3.us-west-2.amazonaws.com/bucket_name`.

Note

Certain S3-compatible servers do not properly implement the `ListObjectsV2` API, most notably Ceph versions before v14.2.5. On these backends, as a temporary workaround, you can provide the `-o s3.list-objects-v1=true` option to use the older `ListObjects` API instead. This option may be removed in future versions of restic.

For more: https://restic.readthedocs.io/en/latest/080_examples.html

Minio Server

Minio is an Open Source Object Storage, written in Go and compatible with Amazon S3 API.

- Download and Install Minio Server.
- You can also refer to https://docs.minio.io for step by step guidance on installation and getting started on Minio Client and Minio Server.

You must first setup the following environment variables with the credentials of your Minio Server.

```
$ export AWS_ACCESS_KEY_ID=<YOUR-MINIO-ACCESS-KEY-ID>
$ export AWS_SECRET_ACCESS_KEY= <YOUR-MINIO-SECRET-ACCESS-KEY>

export AWS_ACCESS_KEY_ID=minio
export AWS_SECRET_ACCESS_KEY=minio123
```

Now you can easily initialize restic to use Minio server as a backend with this command.

```
$ export RESTIC_PASSWORD=Admin_1234

$ ./restic -r s3:http://localhost:9000/restic init

$ restic -r s3:http://shc-sma-cd74.hpeswlab.net:30956/restic init
created restic repository eee650da2d at s3:http://shc-sma-
cd74.hpeswlab.net:30956/restic

Please note that knowledge of your password is required to access
the repository. Losing your password means that your data is
irrecoverably lost.

# back up
restic -r s3:http://shc-sma-cd74.hpeswlab.net:30956/restic backup /tzhong
```

```
$ du -sh /var/vols/itom/
3.2G    /var/vols/itom/

# 3.2G 文件备份所需时间1分04秒。
$ restic -r s3:http://shc-sma-cd74.hpeswlab.net:30956/restic backup
/var/vols/itom
repository eee650da opened successfully, password is correct
no parent snapshot found, will read all files

Files:       20257 new,     0 changed,     0 unmodified
Dirs:        14100 new,     0 changed,     0 unmodified
Added to the repo: 2.584 GiB

processed 20257 files, 3.153 GiB in 1:04
snapshot 264a9584 saved


# restore
restic -r s3:http://shc-sma-cd74.hpeswlab.net:30956/restic restore ab8fb993 --
target /tzhong/restore-work-1

export AWS_ACCESS_KEY_ID=minio
export AWS_SECRET_ACCESS_KEY=minio123
restic -r s3:http://shc-sma-cd74.hpeswlab.net:30956/restic restore 264a9584 --
target /tzhong/restore-work-2
```

> 参考资料:
>
> https://restic.readthedocs.io/en/latest/030_preparing_a_new_repo.html

Backing up

Now we're ready to backup some data. The contents of a directory at a specific point in time is called a "snapshot" in restic. Run the following command and enter the repository password you chose above again:

```
$ mkdir /tzhong
$ cd /tzhong
# 创建 /tzhong/test-file.txt, 大小 500M
$ dd if=/dev/urandom of=/tzhong/test-file.txt count=512000 bs=1024


$ restic -r /srv/restic-repo --verbose backup /tzhong
open repository
enter password for repository:
repository 99bbf28b opened successfully, password is correct
created new cache in /root/.cache/restic
lock repository
```

```
load index files
no parent snapshot found, will read all files
start scan on [/tzhong]
start backup on [/tzhong]
scan finished in 0.206s: 14 files, 614.409 MiB

Files:          14 new,     0 changed,     0 unmodified
Dirs:            5 new,     0 changed,     0 unmodified
Data Blobs:    423 new
Tree Blobs:      6 new
Added to the repo: 614.444 MiB

processed 14 files, 614.409 MiB in 0:07
snapshot b2efd3f0 saved


# If you run the backup command again, restic will create another snapshot of
your data, but this time it's even faster and no new data was added to the
repository (since all data is already there). This is de-duplication at work!
$ restic -r /srv/restic-repo --verbose backup /tzhong
open repository
enter password for repository:
repository 99bbf28b opened successfully, password is correct
lock repository
load index files
using parent snapshot b2efd3f0
start scan on [/tzhong]
start backup on [/tzhong]
scan finished in 0.220s: 14 files, 614.409 MiB

Files:           0 new,     0 changed,    14 unmodified
Dirs:            0 new,     0 changed,     5 unmodified
Data Blobs:      0 new
Tree Blobs:      0 new
Added to the repo: 0 B

processed 14 files, 614.409 MiB in 0:00
snapshot 13e40fc3 saved
```

- Backing up
  - File change detection
  - Dry Runs
  - Excluding Files
  - Including Files
  - Comparing Snapshots
  - Backing up special items and metadata
  - Reading data from stdin
  - Tags for backup
  - Scheduling backups
  - Space requirements
  - Environment Variables
  - Exit status codes

Excluding Files

Let's say we have a file called `excludes.txt` with the following content:

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

It can be used like this:

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

This instructs restic to exclude files matching the following criteria:

> - All files matching `*.c` (parameter `--exclude`)
> - All files matching `*.go` (second line in `excludes.txt`)
> - All files and sub-directories named `bar` which reside somewhere below a directory called `foo` (fourth line in `excludes.txt`)

Including Files

For example, maybe you want to backup files which have a name that matches a certain regular expression pattern (uses GNU `find`):

```
$ find /tmp/some_folder -regex PATTERN -print0 > /tmp/files_to_backup
```

You can then use restic to backup the filtered files:

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

You can combine all three options with each other and with the normal file arguments:

```
$ restic backup --files-from /tmp/files_to_backup /tmp/some_additional_file
$ restic backup --files-from /tmp/glob-pattern --files-from-raw /tmp/generated-
list /tmp/some_additional_file
```

Comparing Snapshots

```
$ restic -r /srv/restic-repo diff 5845b002 2ab627a6
enter password for repository:
repository 99bbf28b opened successfully, password is correct
Fatal: no matching ID found for prefix "5845b002"
[root@shc-sma-cd74 snapshots]#  restic -r /srv/restic-repo diff b2efd3f0
13e40fc3
enter password for repository:
repository 99bbf28b opened successfully, password is correct
comparing snapshot b2efd3f0 to 13e40fc3:
```

```
Files:           0 new,     0 removed,     0 changed
Dirs:            0 new,     0 removed
Others:          0 new,     0 removed
Data Blobs:      0 new,     0 removed
Tree Blobs:      0 new,     0 removed
  Added:   0 B
  Removed: 0 B
```

Tags for backup

Snapshots can have one or more tags, short strings which add identifying information. Just specify the tags for a snapshot one by one with `--tag`:

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

The tags can later be used to keep (or forget) snapshots with the `forget` command. The command `tag` can be used to modify tags on an existing snapshot.

Scheduling backups

Restic does not have a built-in way of scheduling backups, as it's a tool that runs when executed rather than a daemon. There are plenty of different ways to schedule backup runs on various different platforms, e.g. systemd and cron on Linux/BSD and Task Scheduler in Windows, depending on one's needs and requirements. When scheduling restic to run recurringly, please make sure to detect already running instances before starting the backup.

How can I specify encryption passwords automatically?

When you run `restic backup`, you need to enter the passphrase on the console. This is not very convenient for automated backups, so you can also provide the password through the `--password-file` option, or one of the environment variables `RESTIC_PASSWORD` or `RESTIC_PASSWORD_FILE`. A discussion is in progress over implementing unattended backups happens in #533.

> 参考资料:
>
> https://restic.readthedocs.io/en/latest/040_backup.html#
>
> https://restic.readthedocs.io/en/latest/faq.html#how-can-i-specify-encryption-passwords-automatically

Working with repositories

Listing all snapshots

```
$ export RESTIC_PASSWORD=Admin_1234
$ restic -r /srv/restic-repo snapshots
enter password for repository:
repository 99bbf28b opened successfully, password is correct
ID        Time                 Host                        Tags        Paths
----------------------------------------------------------------------------
b2efd3f0  2022-05-23 18:11:28  shc-sma-cd74.hpeswlab.net               /tzhong
13e40fc3  2022-05-23 18:14:42  shc-sma-cd74.hpeswlab.net               /tzhong
----------------------------------------------------------------------------
2 snapshots
```

Copying snapshots between repositories

In case you want to transfer snapshots between two repositories, for example from a local to a remote repository, you can use the `copy` command:

```
$ 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
```

Checking integrity and consistency

Imagine your repository is saved on a server that has a faulty hard drive, or even worse, attackers get privileged access and modify the files in your repository with the intention to make you restore malicious data:

```
$ echo "boom" > /srv/restic-
repo/index/de30f3231ca2e6a59af4aa84216dfe2ef7339c549dc11b09b84000997b139628
```

Trying to restore a snapshot which has been modified as shown above will yield an error:

```
$ 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
```

In order to detect these things before they become a problem, it's a good idea to regularly use the `check` command to test whether your repository is healthy and consistent, and that your precious backup data is unharmed. There are two types of checks that can be performed:

- Structural consistency and integrity, e.g. snapshots, trees and pack files (default)
- Integrity of the actual data that you backed up (enabled with flags, see below)

To verify the structure of the repository, issue the `check` command. If the repository is damaged like in the example above, `check` will detect this and yield the same error as when you tried to restore:

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

If the repository structure is intact, restic will show that no errors were found:

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

By default, the `check` command does not verify that the actual pack files on disk in the repository are unmodified, because doing so requires reading a copy of every pack file in the repository. To tell restic to also verify the integrity of the pack files in the repository, use the `--read-data` flag:

```
$ 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
```

Note

Since `--read-data` has to download all pack files in the repository, beware that it might incur higher bandwidth costs than usual and also that it takes more time than the default `check`.

Alternatively, use the `--read-data-subset` parameter to check only a subset of the repository pack files at a time. It supports three ways to select a subset. One selects a specific part of pack files, the second and third selects a random subset of the pack files by the given percentage or size.

Use `--read-data-subset=n/t` to check a specific part of the repository pack files at a time. The parameter takes two values, `n` and `t`. When the check command runs, all pack files in the repository are logically divided in `t` (roughly equal) groups, and only files that belong to group number `n` are checked. For example, the following commands check all repository pack files over 5 separate invocations:

```
$ 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
```

Use `--read-data-subset=x%` to check a randomly choosen subset of the repository pack files. It takes one parameter, `x`, the percentage of pack files to check as an integer or floating point number. This will not guarantee to cover all available pack files after sufficient runs, but it is easy to automate checking a small subset of data after each backup. For a floating point value the following command may be used:

```
$ restic -r /srv/restic-repo check --read-data-subset=2.5%
```

When checking bigger subsets you most likely want to specify the percentage as an integer:

```
$ restic -r /srv/restic-repo check --read-data-subset=10%
```

Use `--read-data-subset=nS` to check a randomly chosen subset of the repository pack files. It takes one parameter, `nS`, where 'n' is a whole number representing file size and 'S' is the unit of file size (K/M/G/T) of pack files to check. Behind the scenes, the specified size will be converted to percentage of the total repository size. The behaviour of the check command following this conversion will be the same as the percentage option above. For a file size value the following command may be used:

```
$ restic -r /srv/restic-repo check --read-data-subset=50M
$ restic -r /srv/restic-repo check --read-data-subset=10G
```

> 参考资料:
>
> https://restic.readthedocs.io/en/latest/045_working_with_repos.html

Restoring from backup

Restoring from a snapshot

Restoring a snapshot is as easy as it sounds, just use the following command to restore the contents of the latest snapshot to `/tmp/restore-work`:

```
$ restic -r /srv/restic-repo restore b2efd3f0 --target /tmp/restore-work
repository 99bbf28b opened successfully, password is correct
restoring <Snapshot b2efd3f0 of [/tzhong] at 2022-05-23 18:11:28.589557815 +0800
CST by root@shc-sma-cd74.hpeswlab.net> to /tmp/restore-work
```

Removing backup snapshots

Remove a single snapshot

```
[root@shc-sma-cd74 itom]# restic -r s3:http://shc-sma-
cd74.hpeswlab.net:30956/restic snapshots
repository eee650da opened successfully, password is correct
ID        Time                 Host                        Tags        Paths
```

```
----------------------------------------------------------------------------
---
ab8fb993  2022-05-23 20:55:52  shc-sma-cd74.hpeswlab.net              /tzhong
264a9584  2022-05-23 21:01:25  shc-sma-cd74.hpeswlab.net
/var/vols/itom
----------------------------------------------------------------------------
---
2 snapshots

# In order to remove the snapshot of /var/vols/itom, use the forget command and
specify the snapshot ID on the command line:
restic -r s3:http://shc-sma-cd74.hpeswlab.net:30956/restic forget 264a9584

# But the data that was referenced by files in this snapshot is still stored in
the repository. To cleanup unreferenced data, the prune command must be run:
restic -r s3:http://shc-sma-cd74.hpeswlab.net:30956/restic prune
```

> 参考资料:
>
> https://restic.readthedocs.io/en/latest/060_forget.html

Rsync

Rsync (**Remote Sync**) is the most commonly used command for copying and synchronizing files and directories remotely as well as **locally** in **Linux/Unix** systems.

With the help of the **rsync** command, you can copy and synchronize your data remotely and locally across directories, disks, and networks, perform data backups, and mirror between two Linux machines.

Some Advantages and Features of Rsync Command

- It efficiently copies and sync files to or from a remote system.
- Supports copying links, devices, owners, groups, and permissions.
- It's faster than scp (**Secure Copy**) because **rsync** uses a remote-update protocol which allows transferring just the differences between two sets of files. The first time, it copies the whole content of a file or a directory from source to destination but from next time, it copies only the changed blocks and bytes to the destination.
- Rsync consumes less bandwidth utilization as it uses compression and decompression method while sending and receiving data on both ends.

The basic syntax of the rsync command

```
# rsync options source destination
```

Some common options used with rsync commands

- **-v** : verbose
- **-r** : copies data recursively (but don't preserve timestamps and permission while transferring data.
- **-a** : archive mode, which allows copying files recursively and it also preserves symbolic links, file permissions, user & group ownerships, and timestamps.

- **-z** : compress file data.
- **-h** : human-readable, output numbers in a human-readable format.

Install Rsync in Linux System

We can install the **rsync** package with the help of the following command in your Linux distribution.

```
$ sudo apt-get install rsync   [On Debian/Ubuntu & Mint]
$ pacman -S rsync              [On Arch Linux]
$ emerge sys-apps/rsync        [On Gentoo]
$ sudo dnf install rsync       [On Fedora/CentOS/RHEL and Rocky Linux/AlmaLinux]
$ sudo zypper install rsync    [On openSUSE]
```

Copy a Directory from Local Server to a Remote Server

```
yum install -y rsync

rsync -avzh /tzhong root@shc-sma-cd75.hpeswlab.net:/tzhong/

# Copy `/var/vols/itom` from cd74 to cd75.
# 3.2G 文件备份所需时间3分11秒。
rsync -avzh /var/vols/itom root@shc-sma-cd75.hpeswlab.net:/tzhong
```

Copy/Sync a Remote Directory to a Local Machine

```
rsync -avzh root@shc-sma-cd75.hpeswlab.net:/tzhong/tzhong /tmp/tzhong
```

> 参考资料:
>
> https://www.tecmint.com/rsync-local-remote-file-synchronization-commands/

Rsync 和 Restic 备份文件时间对比

	文件大小	Rsync	Restic
（跨Host）备份所花时间	3.2G	3分11秒	48秒
（跨Host）还原所花时间	3.2G	N/A	38秒

Restic 存储在 MinIO上的文件会被压缩，3.2G文件实际所用空间为2.6G，压缩比81%。