rsync

- A powerful tool used to synchronize files & directories.

Can be used to :

- Over the internet via ssh or rsync protocol, etc ...
- Capable of compressing before sending over the network, and decompressing on the other side.
- Sychronize differentially: only copy files that are newer, skip already copied ones.
- Encrypt files sent.
- And more!

Basic Usage :

```
rsync [options] source destination
```

Options:

- `-a` : archive mode equals -rlptgoD (no -H,-A,-X).
- `-r` : recurse into directories.
- `-l` : copy symlinks as symlinks.
- `-p` : preserve permissions. *
- `-t` : preserve modification times. *
- `-g` : preserve group.
- `-o` : preserve owner. *
- `-D` : preserve special and device files. *
- `-v` : verbose.
- `-e` : specify the remote shell to use.
- `-z` : compress files before transfer, decompress after (Useful in remote sync cases).
- `-c` : skip based on checksum, not mod-time & size
- `-u` : skip files that are newer on the receiver (only copies the changes).
- `-P` : --partial && --progress. In case of network corruption rsync will continue from the stopped point and show the progress respectively.
- `-m` : prune empty directory chains from file-list.

* requires super-user privileges.

> Example: Sync directories content in the same computer.

```
$ tree tests/
tests/
├── 01
│   └── 4.txt
├── 1.txt
├── 2.txt
└── 3.txt

1 directory, 4 files
```

- We notice that, it has a subdirectory called 01.

We will sync its content to backups directory.

```
$ rsync -v tests/* backups/
skipping directory 01
1.txt
2.txt
3.txt

sent 197 bytes  received 73 bytes  540.00 bytes/sec
total size is 0  speedup is 0.00
```

As you can see from the verbosed output it skipped the 01 directory, by taking a look into `backups` directory we will see that it only has `1.txt`, `2.txt`, `3.txt`.

```
$ tree backups/
backups/
├── 1.txt
├── 2.txt
└── 3.txt

0 directories, 3 files
```

So in order to avoid this unexpected behavior we will use `-r` option.

> Example:

```
$ rsync -rv tests backups/
sending incremental file list
tests/
tests/1.txt
tests/2.txt
tests/3.txt
tests/01/
tests/01/4.txt
```

```
sent 344 bytes  received 108 bytes  904.00 bytes/sec
total size is 0  speedup is 0.00
```

> `backups` content :

```
$ tree backups/
backups/
└── tests
    ├── 01
    │   └── 4.txt
    ├── 1.txt
    ├── 2.txt
    └── 3.txt

2 directories, 4 files
```

IMPORTANT NOTE:

if you kept the forward slash `/` after the source directory - `source/` - it will sync its content only, but if you removed it `source` it will sync the entire directoy as we saw in the above example `tests` completely synced into `backups`.

You may only need to sync the content instead of sync the entire directory in this case you will append `/` to the source directory.

```
$ rsync -rv tests/ backups/
sending incremental file list
1.txt
2.txt
3.txt
01/
01/4.txt

sent 326 bytes  received 104 bytes  860.00 bytes/sec
total size is 0  speedup is 0.00
```

```
$ tree backups/
backups/
├── 01
│   └── 4.txt
├── 1.txt
├── 2.txt
└── 3.txt

1 directory, 4 files
```

As you can see we synced only the content of the tests directory rather than sync the entire directory.

PRO TIP:

You also use the `--dry-run` option or for short `-n` before performing the sync which will display the content to be synced, this is so handy as it will not perform actual syncing but instead will show the content to be synced.

NOTE:

it needs a verbose to run.

> Example:

```
$ rsync -rvn tests/ backups/
sending incremental file list
1.txt
2.txt
3.txt
01/
01/4.txt

sent 174 bytes  received 32 bytes  412.00 bytes/sec
total size is 0  speedup is 0.00 (DRY RUN)
```

Those displayed files are the ones will be synced. Once everything seems perfect to you can remove `-n` flag and sync.

Suppose you have a configuration files or anything that you need to move it with preserve its permissions, ownership, etc...

In this case, you will add the archive flag `-a`.

`The files are transferred in archive mode, which ensures that symbolic links, devices, attributes, permissions, ownerships, etc. are preserved in the transfer.`

> Basic Comparsion:

> Without `-a` option.

```
$ ls -la tests/
total 12
drwxr-xr-x 3 rakan rakan 4096 Jun  7 11:48 .
drwxr-xr-x 4 rakan rakan 4096 Jun  8 09:01 ..
drwxr-xr-x 2 rakan rakan 4096 Jun  7 11:48 01
-rwxrwxrwx 1 rakan rakan    0 Jun  7 11:48 1.txt
-rw-r--r-- 1 rakan rakan    0 Jun  7 11:48 2.txt
-rw-r--r-- 1 rakan rakan    0 Jun  7 11:48 3.txt
```

Here I've changed the permission for `1.txt` file to `777` just to illustrate the point. And I copied the content using the following command.

```
rsync -rv tests/ backups/
```

Now if we check the permissions:

```
$ ls -la backups/
total 12
drwxr-xr-x 3 rakan rakan 4096 Jun  8 09:03 .
drwxr-xr-x 4 rakan rakan 4096 Jun  8 09:04 ..
drwxr-xr-x 2 rakan rakan 4096 Jun  8 09:03 01
-rwxr-xr-x 1 rakan rakan    0 Jun  8 09:03 1.txt
-rw-r--r-- 1 rakan rakan    0 Jun  8 09:03 2.txt
-rw-r--r-- 1 rakan rakan    0 Jun  8 09:03 3.txt
```

As you can see it has been changed from `777` to `755`. But what if we are satisfied with `777`? :)

In this case we will add the `-a` option.

> With `-a` option:

We don't need to add `-r` since `-a` will do it for us.

```
rsync -av tests/ backups/
```

```
$ ls -la backups/
total 12
drwxr-xr-x 3 rakan rakan 4096 Jun  7 11:48 .
drwxr-xr-x 4 rakan rakan 4096 Jun  8 09:08 ..
drwxr-xr-x 2 rakan rakan 4096 Jun  7 11:48 01
-rwxrwxrwx 1 rakan rakan    0 Jun  7 11:48 1.txt
-rw-r--r-- 1 rakan rakan    0 Jun  7 11:48 2.txt
-rw-r--r-- 1 rakan rakan    0 Jun  7 11:48 3.txt
```

Now we got the desired output and we are satisfied ;)

Suppose we changed/modified one or a couple of files it's insufficient to resync the entire directory/content.
Instead we will use the `-u` option to sync/copy only the modified/newer files.

Let's take our previous example:

```
$ tree tests/
tests/
├── 01
│   └── 4.txt
├── 1.txt
├── 2.txt
├── 3.txt
```

```
└── 4.txt

1 directory, 5 files
```

I've added one file, `4.txt`.

Now let's use `-u` and verify what is going to be synced/copied with dry-run `-n` of cource we will compair with checksum `-c` to be more precise ;)

```
$ rsync -avunc tests/ backups/
sending incremental file list
./
4.txt

sent 293 bytes  received 23 bytes  632.00 bytes/sec
total size is 0  speedup is 0.00 (DRY RUN)
```

As we can see only `4.txt` is going to be synced/copied. Now we can remove `-n` option and sync the content.

What if we want to sync multiple files?

You can achieve this in different ways but I'll explain the easiest way.

Which is specifying the needed files inside `{}` with comma separated.

> Example:

```
rsync -av {tests/,/etc/passwd} backups/
```

> Result :

```
$ tree backups/
backups/
├── 01
│   └── 4.txt
├── 1.txt
├── 2.txt
├── 3.txt
├── 4.txt
└── passwd

1 directory, 6 files
```

---

sync certain extensions/files and ignore the others?

In this case we will use `--include` with `--exclude` to get the job done.

> Suppose we have the following hierarchy and I want to sync all files with `.pdf` extension and ignore anything else.

```
$ tree tests/
tests/
├── 01
│   ├── 4.txt
│   └── book2.pdf
├── 1.txt
├── 2.txt
├── 3.txt
├── 4.txt
├── blah.pdf
├── book1.pdf
└── news.pdf

1 directory, 9 files
```

```
rsync -avm --include="*/" --include='*.pdf' --exclude='*' tests/ backups/
```

```
$ tree backups/
backups/
├── 01
│   └── book2.pdf
├── blah.pdf
├── book1.pdf
└── news.pdf

1 directory, 4 files
```

We got the desired output but let's briefly explain the issued command.

- First, `-avm` a bunch of flags already known for you, the `-m` flag is to avoid go recursive in the empty directories.
- Second, `--include="*/"` means go to the subdirectories in the source directory.
- Third, `--include='*.pdf'` include all files with `.pdf` extension.
- Finally, `--exclude='*'` exclude anyother file that doesn't have `.pdf` extension.

Remotely:

> Example: Copy file from your computer to remote server.

Please note:

ssh/rsync daemon should be up and running on the server or on your computer if it was the dest.

```
rsync -v localFile user@10.10.10.10:~/remoteFile
```

`NOTE:` If you did not specify a location on the remote server. By default, it will be saved on the user's home directory.

> Example:

```
rsync -v localFile user@10.10.10.10:
```

Thus the file will be syned at `/home/user/`

> Example: Copy file from remote server to your computer.

```
rsync -v user@10.10.10.10:~/remoteFile localFile
```

> Example: Copying file from server with custom ssh port.

```
rsync -e 'ssh -p 1337' user@10.10.10.10:remoteFile localFile
```

> Example: Sync entire directory's content. Progress and pratial enabled `-P` and only `-u`pdated files.

```
rsync -urvP dirLocal/ user@10.10.10.10:~/backups/
```

> Copy multiple files from remote server:

```
rsync -zavP user@10.10.10.10:{/etc/passwd,/etc/hostname,/var/www/html/index.php} dirLocal/
```

NOTE:

We used `-z` to compress the files in transmitting and save some bandwidth

---

Conclusion :

rsync is a powerful tool can be used in many situations, it is also recommended to dive and explore more about the tool, use it in combination with your scripts, make cronjobs, etc...

If you have any doubts feel free to contact me on ;)

EOF