Backups with Borg and Cron

By Stefan Paquay / spaquay@brandeis.edu

This is a short guide on using Borg to make backups and how to automate that using Cron. Some sample scripts are provided.

This guide provides a short overview of making backups using Borg backup and automating this using Cron. It goes over the following:

1. A short guide to Borg terminology
2. Creating a Borg repository
3. Adding a backup to an existing Borg repository
4. Extracting a backup from an existing Borg repository
5. Dealing with remote repositories
6. A short guide to Cron terminology
7. Automating the above with Cron

1. Borg terminology:

Borg uses some jargon/terms:

<table>
<thead>
<tr>
<th>Term</th>
<th>What it is</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>A directory (can be remote, can be local) in which backups are stored.</td>
</tr>
<tr>
<td>Archive</td>
<td>An archive is a single backup entry in a repository.</td>
</tr>
<tr>
<td>Deduplication</td>
<td>This is the act of removing/preventing data duplication in a backup, which leads to less disk usage</td>
</tr>
</tbody>
</table>

2. Creating a repository:

Borg is mainly (only?) driven from the command line, specifically through the "borg" command. To create a Borg repository, the "borg init" command is used:

```
$ borg init --encryption=repokey --progress ~/backup_test/paper_42_data
```

At the time of initializing a repository, you specify encryption (if any) with the "--encryption" switch. By using a so-called repokey, Borg will generate a public/private key pair and will (optionally) encrypt it with a passphrase. It is in general recommended to encrypt your data (because it's better to be safe than sorry) but if for whatever reason you want to opt out, use "--encryption=none".

Borg will create the repository itself (paper_42_data) but not the rest of the path, so make sure it exists before you issue the init command:

```
$ mkdir -p ~/backup_test/
```

Once you issue the "borg init" command, Borg will prompt you for a passphrase. Once entered, it will ask you if it should display your passphrase for verification:

```
[ stefan@hagan-640 ~ ] % borg init --encryption=repokey --progress /home/stefan/backup_test/paper_42_data
```

Enter new passphrase:
Enter same passphrase again:
Do you want your passphrase to be displayed for verification? [yN]: y

This is probably a good idea to do because you might have made a mistake while typing.

With the completion of this command, a Borg repository has been made and is ready for use. It can be inspected with the "borg info" command:

```
$ borg info ~/backup_test/paper_42_data
```

Borg will prompt for the passphrase and will then show some info:
3. Adding a backup to the repository

Now that we have a fresh repository, let's add some data to it to be backed up. In a different directory, we have a bunch of data:

```
$ borg create --stats --progress ~/backup_test/paper_42_data::2018-11-27 ~/results/paper_42
```

To name the specific backup, we use the strange "::"-syntax. In this case, it means "Add ~/results/paper_42 to the repository ~/backup_test/paper_42_data under the name '2018-11-27'". Borg will again prompt for the passphrase if you used encryption. At the end, it will show you some stats about the repository:

```
$ borg info /home/stefan/backup_test/paper_42_data
```

Normally, Borg will achieve reasonably good compression on data. In this case it completely fails because I generated the "data" from the system's random device (/dev/urandom).

4. Extracting data from repositories

Backing up data makes no sense unless you can also extract it later. There are a few ways to do this in Borg, and I will explain two. The first is simply extracting the backup somewhere using "borg extract".

```
$ mkdir ~/extract_backup
$ cd ~/extract_backup
$ borg extract --progress ~/backup_test/paper_42_data::2018-11-27
```

Again, Borg will prompt for a passphrase:

```
$ borg info /home/stefan/backup_test/paper_42_data
```

The extracted data will contain the complete directory path that we added to the repository before:
Therefore, if you add data the way I did before and you extract them in the root filesystem (/), it will overwrite whatever data was in its place. While that is useful for restoring a complete system or for moving data from one system to another, sometimes you just want to grab a few files from a backup rather than all the data. For this, Borg supports mounting the backup. Note that it does require the Python module "llfuse" for this.

5. Dealing with remote repositories

There are in general two ways to interact with remote repositories. One is to simply mount a remote file system somewhere on your local machine and run Borg from your local machine while putting the repository on the mounted file system. This is the only way to use Borg with a server that does not have Borg installed. If a server does have Borg installed, it is recommended to use the Borg protocol instead because it is faster. A remote filesystem to which you have SSH access can be mounted using SSHFS, which might not be available.

Related articles

- Backups with Borg and Cron
- AMANDA Backups