Linux Directory Backup to AWS S3

# Script

See script file

# Detail

## Linux Directory Backup to AWS S3

Compress, encrypt, and backup a Linux directory to AWS S3 using cron.

### Create an AWS S3 Bucket

Create a private Amazon AWS S3 bucket to store your database backups: [AWS 'create bucket' guide](<https://docs.aws.amazon.com/AmazonS3/latest/user-guide/create-bucket.html>).

### Create IAM User

Create an IAM user in your AWS account with access to the S3 bucket created above: [AWS 'create user' guide](https://docs.aws.amazon.com/IAM/latest/UserGuide/id\_users\_create.html)

The script requires, list, put, and delete access on the s3 bucket. So, the S3 policy JSON attached to the IAM user might look like:

```json

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"s3:PutObject",

"s3:ListBucket",

"s3:DeleteObject"

],

"Resource": [

"arn:aws:s3:::<bucket-name>/\*",

"arn:aws:s3:::<bucket-name>"

]

}

]

}

```

Make sure to download or keep hold of the new user security credentials so you can add them to the backup script environment later.

### Create PGP Keys

On a separate (ideally air-gapped) machine, install [GPG](<https://gnupg.org/>):

```sh

apt install gnupg

```

Then create a pair of public and private encryption keys. Using public-key cryptography to encrypt the backup on the server will help prevent the database backup from being compromised if the environment variables are leaked.

- Generate a keypair using your email for ID: ```gpg --gen-key```

- Export the public key: ```gpg --armor --export <your-email>```

- Export the secret key and move to [secure storage](<https://lwn.net/Articles/734767/>).

### Install Dependencies

Install the script dependencies on the VM:

- \*\*GPG\*\* - Install [GPG](https://gnupg.org/) to encrypt backup files: ```apt install gnupg```

- \*\*AWS-CLI\*\* - Install AWS CLI tool to transfer the backup to AWS S3: see [AWS guide](https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-linux.html)

- \*\*date\*\* - Ensure date is GNU core utilities date, not included in alpine linux (busybox) by default: ```apt install coreutils```

### Deploy to Server

#### Add Script

Add backup script (code snippet) to suitable directory:

```sh

mkdir ~/directory-backup

vim ~/directory-backup/directory-backup-s3.sh

```

Modify file permission to prevent unauthorized writes:

```sh

chmod 744 ~/directory-backup/directory-backup-s3.sh

```

#### Add Config

Add config file to contain config and credentials for the backup script, and modify file permission to prevent unauthorized reading of sensitive credentials:

```sh

touch ~/directory-backup/config.env

chmod 700 ~/directory-backup/config.env

```

Add config:

```sh

vim ~/directory-backup/config.env

```

Sample config:

```sh

export ROTATION\_PERIOD= # days to keep backups (exclude to stop backups from deleting)

export BACKUP\_NAME= # name of backup file

export DIR\_PATH= # path to directory to backup

export AWS\_ACCESS\_KEY\_ID= # AWS IAM USER ID

export AWS\_SECRET\_ACCESS\_KEY= # AWS IAM USER KEY

export AWS\_DEFAULT\_REGION= # aws s3 bucket region

export S3\_BUCKET= # aws s3 bucket name

export GPG\_KEY\_ID= # id used in gpg key generation

export GPG\_KEY= # exported amoured GPG key

```

### Create cron Job

Add a new cron job using crontab. The job should periodically load the environment variables and then run the backup script. For example, to run the backup daily at 3.30 am:

```sh

crontab -e

```

```sh

30 3 \* \* \* . $HOME/directory-backup/config.env && $HOME/directory-backup/directory-backup-s3.sh 2>&1 | logger -t psql-backup-s3

```

For more info on setting up a job using crontab, checkout [ubuntu's guide here](https://help.ubuntu.com/community/CronHowto). [crontab guru](https://crontab.guru/) can be helpful for defining schedules.

Note: When setting up, it can be useful to set the job schedule to a short interval such as ```\*/2 \* \* \* \*``` so you can check for any misconfiguration or errors. Follow the ```/var/log/syslog``` to see the logger's output if using the above example.

### Restore

To restore a backup:

- Download the encrypted backup from aws S3

- Copy to the machine containing the private gpg key

- Decrypt downloaded file using gpg: ```gpg --output <decrypted file name>.tar.bz2 --decrypt <downloaded file name>.tar.bz2.gpg```

- Unzip decrypted file using bzip: ```bzip2 -d <decrypted file name>.tar.bz2```

- Untar using ```tar -xvf <decrypted file name>.tar```