Borg Hive

bpereto

Jan 09, 2023

CONTENTS

CHAPTER

ONE

WHAT IS BORG HIVE?

Borg Hive is a management interface for borgbackup repositories.

The main goal of Borg Hive is to provide a easy management of borg repositories and ssh keys, also provide notifications if there is a stale backup. Optionally, it collects some events and statistics what's happening.

I backup my peripherals at home with borgbackup, which works nice on my servers, android phones, laptops, workstations and so on. To keep the overview over my backups and which device haven't done one in a while I decided to write a dashboard for it. The focus is for backups at home, but Borghive should also work in the cloud or in an enterprise.

Warning: This is under active development. It's Alpha!

CHAPTER

TWO

FEATURES

- Repository Managment
- Repository Statistics
- SSH-Key Management
- Notifications of stale backups (E-Mail, Pushover)
- Partially Repository Events (should be improved)
- Basic Object Permissions (Owner & Group) of repositories, SSH-Keys and notifications

CHAPTER

THREE

QUICKSTART

Use the documentation for a quickstart and installation:

borg-hive.readthedocs.io

CHAPTER

FOUR

WHAT IT SHOULD ALSO HAVE IN THE FUTURE / TODO

- More notification types
  - GET/POST Webhooks
  - Other wanted notification types
- REST API (Django Rest Framework)
- Send Logs from borg client / borgmatic to API
- Backup Scheduling & Trigger with Ansible -> AWX/Tower Integration

4.1 Installation

The application is optimized for a containerized setup.

There are different ways to install and run Borg-Hive:

- *Docker* - setup with docker and docker-compose
- *Kubernetes* - easy and fast deployment with helm to kubernetes

4.1.1 Docker

Prerequisites: You should have docker and docker-compose installed and running.

```
# Configure the environment
# Set Admin password: BORGHIVE_ADMIN_PASSWORD=
# optionally set EMAIL or LDAP settings
vi .env

# start app
docker-compose up

# wait untill the app worker is finished setting up
```

Open the browser and navigate to your host: ex. http://localhost:8000

4.1.2 Kubernetes

Prerequisites:

- You should have setup your k8s and kubectl
- You installed helm
- You have an nginx ingress running (the charts must be adjusted to use other ingresses)

Configuration:

- Adjust the DNS-Name in `values.yaml`
- Configure the DNS-Name in your DNS and point it to the Ingress-IP of k8s.

```
# Add helm repo of bitnami
helm repo add bitnami https://charts.bitnami.com/bitnami
helm dep update

# create own namespace in k8s for borg-hive
kubectl create namespace borg-hive

# mariadb should be installed first
helm install mariadb bitnami/mariadb --namespace borg-hive -f values.db.yaml
helm upgrade --install borg-hive . -f values.yaml --namespace borg-hive
```

Important: `helm upgrade` does regenerate the secrets (passwords) of mariadb and openldap. therefore the mariadb is installed sepparate. Keep in mind: on each helm upgrade, the pods of borg-hive should be deleted (and will be recreated) to adjust the secret for openldap in the container.

Services

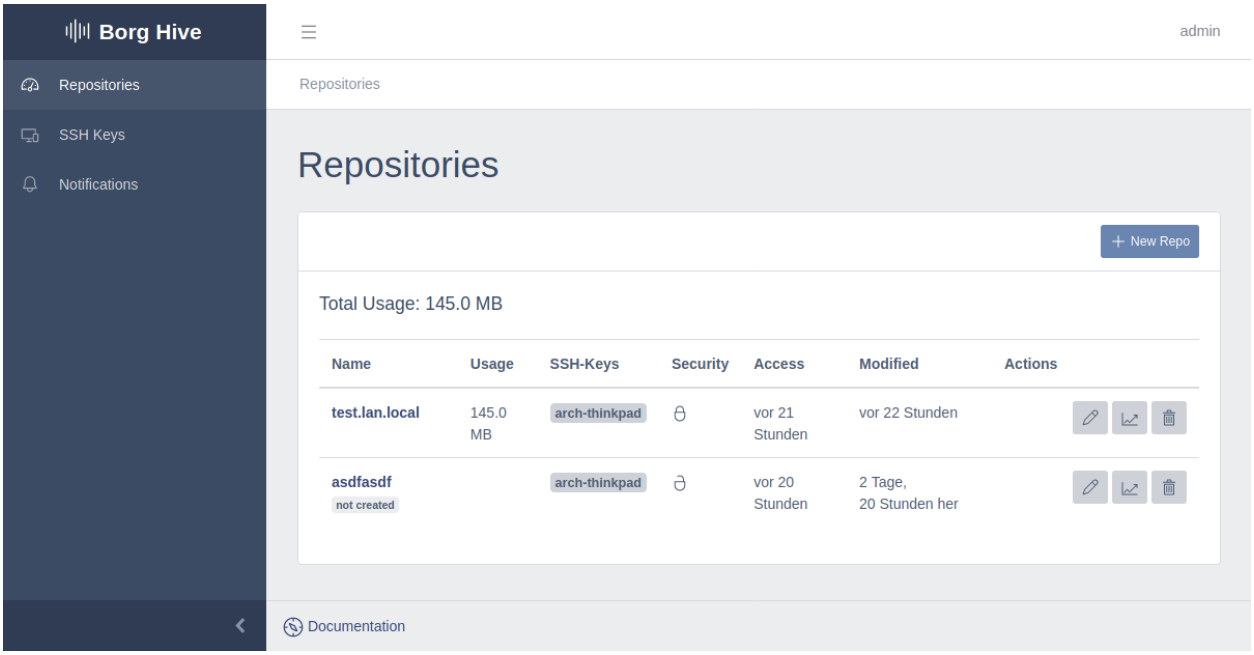
- The web-tier should now be accessible through the ingress. In this example: https://borg-hive.app.local
- borgbackup should now be accessible through the Loadbalancer IP. In this example: 192.168.101.204:22

```
# kubectl get services --namespace borg-hive

borg-hive-app         ClusterIP      10.111.108.79   <none>           8000/TCP         ␣
↪40h
borg-hive-borg        LoadBalancer   10.100.163.89   192.168.101.204  22:30198/TCP     ␣
↪40h
borg-hive-openldap    ClusterIP      10.97.223.166   <none>           389/TCP,636/TCP  ␣
↪38h
mariadb               ClusterIP      10.97.32.101    <none>           3306/TCP         ␣
↪2d14h
mariadb-slave         ClusterIP      10.96.86.61     <none>           3306/TCP         ␣
↪2d14h
redis-headless        ClusterIP      None            <none>           6379/TCP         ␣
↪39h
redis-master          ClusterIP      10.97.123.94    <none>           6379/TCP         ␣
↪39h
redis-slave           ClusterIP      10.96.114.224   <none>           6379/TCP         ␣
↪39h
```

4.2 Quick Start

This chapter will get you started with Borg Hive.



4.2.1 Get started

Installation

- For a development setup look into *Development*
- For a brief test or a standalone installation look into *Docker*
- For a more advanced approach or cloud installation look into the deployment to *Kubernetes*

First Usage

Add or adjust your repository location in the admin interface, which is used to generate the borgbackup destination url: http://localhost:8000/admin/borghive/repositorylocation/

Add an SSH-Key to borg-hive which is needed to create a repository.

Create your first repository and do a backup. Besides plain borgbackup it also works well with borgmatic or vorta.

4.3 Usage

Documentation to the usage of Borg-Hive.

4.3.1 Import or Export existing Repositories

If you already have an existing Borg repository you can import the archive into Borg-Hive with rsync. An export of the Borg repository is also possible in the same way.

Note: Due the nature of rsync it's possible to store arbitrary data besides a Borg repository in Borg-Hive, which is not recommended.

There are different repository modes to control the behaviour of the repository in borg-hive:

- **BORG**: (default) enables borgbackup operations
- **IMPORT**: enables write-only to the given repository location with rsync
- **EXPORT**: enables read-only to the given repository location with rsync

Only one mode can be active. It's not possible to import with rsync and do brogbackups at the same time.

Import existing Borg repository

Create a new repository. Afterwards edit the repository and set the repository mode to `IMPORT`.

Execute rsync to transfer the data to borg-hive:

```
$ rsync -Paz --stats REPO-TO-IMPORT/ xxxxxx@borg-hive-app.local:
```

Note: The `/` is required to copy the content of the folder and not the folder itself.

Warning: An import can also be executed on an existing repository, but be aware that the stored data will be overwritten or deleted.

Export existing Borg repository

Edit the repository and set the repository mode to `EXPORT`.

Execute rsync to transfer the data from borg-hive:

```
$ rsync -Paz --stats xxxxxx@borg-hive-app.local: /home/my-local-repo
```

4.4 Backup Borg Hive

4.4.1 LDAP

If using the version with the openldap-backup container, the LDAP container backs up it's data via a cron job that can be configured in your *.env* file:

```
LDAP_BACKUP_CONFIG_ENV="5 * * * *"
LDAP_BACKUP_DATA_ENV="*/5 * * * *"
```

For a backup of the config every 5th minute past the hour, a data backup every 5 minutes.

4.4.2 Restoring an LDAP backup

To restore an LDAP backup to an openldap container, run the following commands:

```
docker exec -it openldap-backup bash
sv stop /container/run/process/slapd
rm -r /etc/ldap/slapd.d/*
slapd-restore-config 20190502T082301-config
sv stop /container/run/process/slapd
rm /var/lib/ldap/*
slapd-restore-data 20190502T080901-data
```

4.4.3 MySQL/MariaDB

Find the IP of the MariaDB container: `sudo docker ps` and look for the container ID of MariaDB .

`sudo docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' <Docker Container ID>`

This will give you the IP of the MariaDB container.

To get a dump from this, you can use MySQLDump, with the -h flag: `mysqldump -h <IP of Container> -u <username> -p<password> borghive > /var/backup/mybackup.sql`

Note to replace the username, IP, password and possibly the database name (borghive) and the target file, with your own configuration.

If you use Borgmatic to back up the Borg Hive, the MySQL/MariaDB config can be entered in the YAML file of Borgmatic.

4.5 Internals

Some detail informations about the internals of borg hive.

4.5.1 Repository User

A unique Repository User is generated per Repository to ensure that only one Repository can be accessed.

So the structure on the filesystem looks like this: `/repos/<repo user>/<repo name>`

On Repository User create, a corresponding in the ldap backend through *django-ldapdb* is generated, that sshd allows a login for this user. The SSHD Service queries the ldap passwd and shadow entries through the PAM ldap module.

The PAM configuration creates the directories on first user login.

4.5.2 Repository Events

Due to Borg's design to distrust the server as little information as possible is emitted on the server part and therefore its not easy to obtain correct informations about the repository.

At the moment the following is detected:

- Borg Repository open: `lock.exclusive` is created
- Borg Repository close: `lock.exclusive` is removed
- Last Access: Modification Time of repository directory (because of create/delete of the lock file)
- Last Update: Modification Time of the index.* files.
- Backup Usage: Usage on Filesystem (du)

A future enhancement could be a plugin to borgmatic to submit the information and logs via API to Borg Hive.

The management command `watch_repositories` runs inotify on the repository directory and a combination of files and paths results in repository events.

4.5.3 Repository Statistic

The Repository Statistic is obtained each day, when a repsitory is refreshed and after a "Repository Updated" Event is emitted.

4.5.4 SSH Authentication

To prevent managing the ssh keys in authorized keys files per repo user on the filesystem, the ssh-keys are retrieved from the database. When the User logs in, the management command `authorized_keys_check` is executed by the sshd-server trough the statement: `AuthorizedKeysCommand`

```
Match User *
  AuthorizedKeysCommand /app/manage.py authorized_keys_check --user %u
  AuthorizedKeysCommandUser borg
```

The command expects on stdout lines of the format of the authorized keys.

After SSH-Key authentication, the user must be allowed through PAM.

4.6 Development

Setup the development docker containers.

The Database needs time to initialize the first time:

```
docker-compose -f docker-compose.dev.yml up -d db
docker-compose -f docker-compose.dev.yml up -d
docker exec -it borg-hive_app_1 /bin/bash
./manage.py createsuperuser
```

Open the browser and navigate to your host: ex. http://localhost:8000