Requests Documentation

Release 0.4

Kenneth Reitz

Apr 03, 2021

Contents

Release v0.4. (*Installation*)

CyBorgBackup is a Web and API Interface to manage Borg Backup solution on multiple servers based on Django and AngularJS frameworks.

CHAPTER 1

Features

- Borg Backup system
- SSH Connection
- Scheduled job
- Local and Remote Borg Repository
- Catalog based on Borg Archive
- Restore Test
- Archive Size statistics
- Client and Repository preparation
- VM Backup Modules
- E-mail notification
- Auto-prune
- Logs system
- REST API

CHAPTER 2

Underground

CyBorgBackup using the following tools

```
- PostgreSQL database
- ElasticSearch
- RabbitMQ messaging system
- Django framework
- Django REST Framework
- Celery and Beat
- AngularJS framework
- BorgBackup
```

CyBorgBackup have been separated in two project

```
- CyBorgBackup => The main API system
- CyBorgBackup-UI => The Web Interface who can manage multiple CyBorgBackup servers
```

CHAPTER 3

The User Guide

3.1 Installation of CyBorgBackup

The following covers three different installation methods of CyBorgBackup.

3.1.1 Debian Package

The latest release of CyBorgBackup is available as a Debian package and can be downloaded from github releases page:

```
# apt install git postgresql-all rabbitmq-server python3-pip python3-virtualenv␣
↪python3-setuptools python3-venv systemd nginx git
# wget https://api.github.com/repos/cyborgbackup/cyborgbackup/releases/latest -O -␣
↪|grep -oP '"browser_download_url": "\K(.*)(?=")' |wget -i -
# dpkg -i cyborgbackup_X.X.X_all.deb
```

To add UI system:

```
# wget https://api.github.com/repos/cyborgbackup/cyborgbackup-ui/releases/latest -O -␣
↪|grep -oP '"browser_download_url": "\K(.*)(?=")' |wget -i -
# dpkg -i cyborgbackup-ui_X.X.X_all.deb
```

Note : MongoDB is not provided by the Debian repository, you need to follow mongodb documentation.

3.1.2 Docker Install

To install CyBorgBackup under Docker, run this command in your terminal of choice:

```
$ wget https://raw.githubusercontent.com/cyborgbackup/cyborgbackup/master/docker-
↪compose.full.yml -O docker-compose.yml
$ cat > .env <<EOF
```

(continues on next page)

(continued from previous page)

```
POSTGRES_PASSWORD=cyborgbackup
POSTGRES_USER=cyborgbackup
POSTGRES_NAME=cyborgbackup
POSTGRES_HOST=postgres
RABBITMQ_DEFAULT_USER=cyborgbackup
RABBITMQ_DEFAULT_PASS=cyborgbackup
RABBITMQ_DEFAULT_VHOST=cyborgbackup
SECRET_KEY=$(openssl rand -base64 47|sed 's/=//g')
EOF
$ docker-compose up
$ docker-compose exec web /bin/bash
web$ python3 /cyborgbackup/manage.py loaddata settings
web$ echo "from django.contrib.auth import get_user_model; User = get_user_model();␣
↪User.objects.create_superuser('admin@cyborg.local', 'admin')" | python3 /
↪cyborgbackup/manage.py shell
web$ exit
```

If you don't have docker-compose or docker installed, head over to the website for installation instructions.

3.1.3 Install from Source Code

CyBorgBackup is developed on GitHub, where the code is always available.

You can either clone the public repository:

```
$ git clone https://github.com/cyborgbackup/cyborgbackup.git
```

Or, download the tarball:

```
$ curl -OL https://github.com/cyborgbackup/cyborgbackup/tarball/master
# optionally, zipball is also available (for Windows users).
```

The UI interface can be found on Github, the the code is always available. And the public repository can be found at the following address:

```
$ git clone https://github.com/cyborgbackup/cyborgbackup-ui.git
```

To build the UI docker image run the following:

```
$ docker build --no-cache --pull -t cyborgbackup/cyborgbackup-ui:latest .
```

Depending of your system, CyBorgBackup needs the following dependencies :

- python3
- python3-pip
- postgresql-server
- rabbitmq-server
- nginx

To use the CyBorgBackup container with Docker, launch the following command:

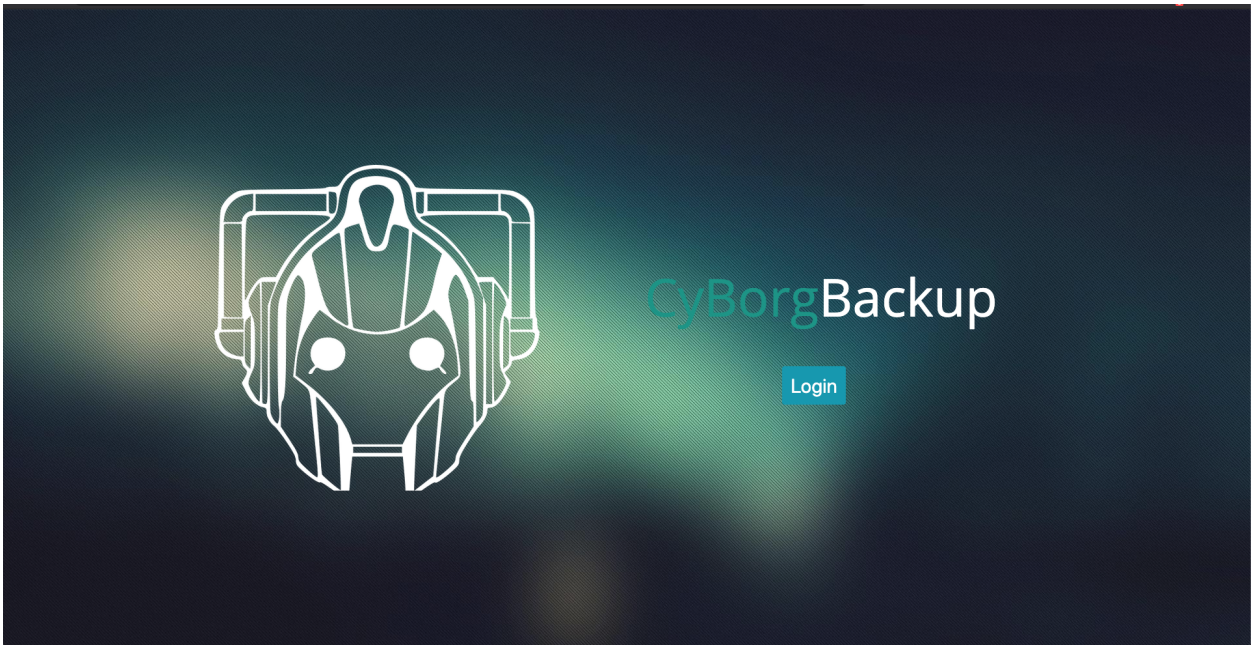
```
$ make cyborgbackup-docker-build
$ make docker-compose-up
```

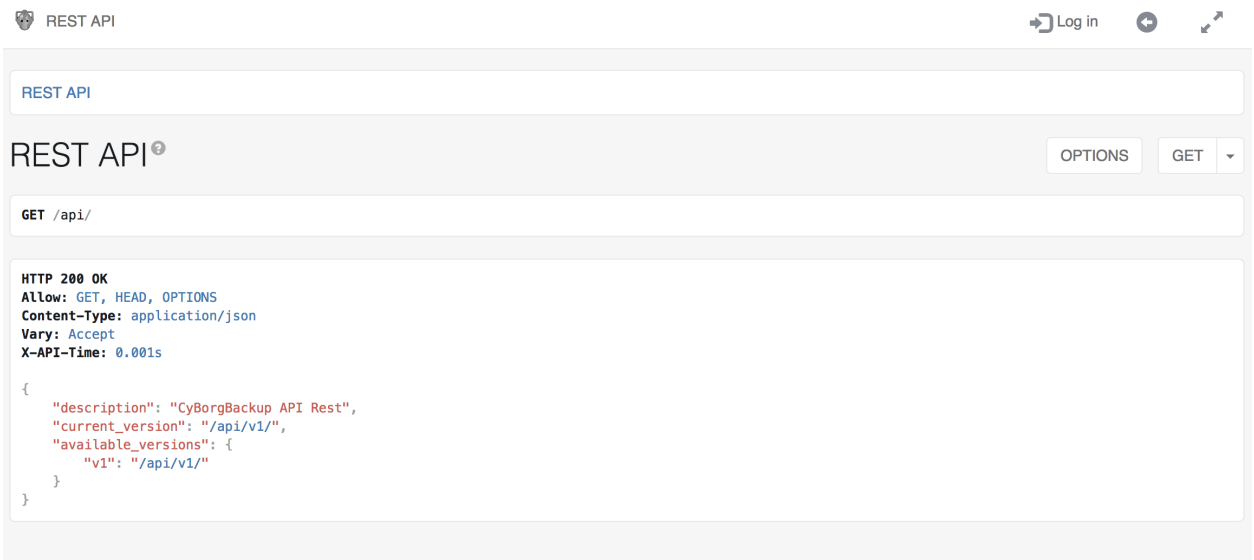
3.1.4 Connecting to the interface

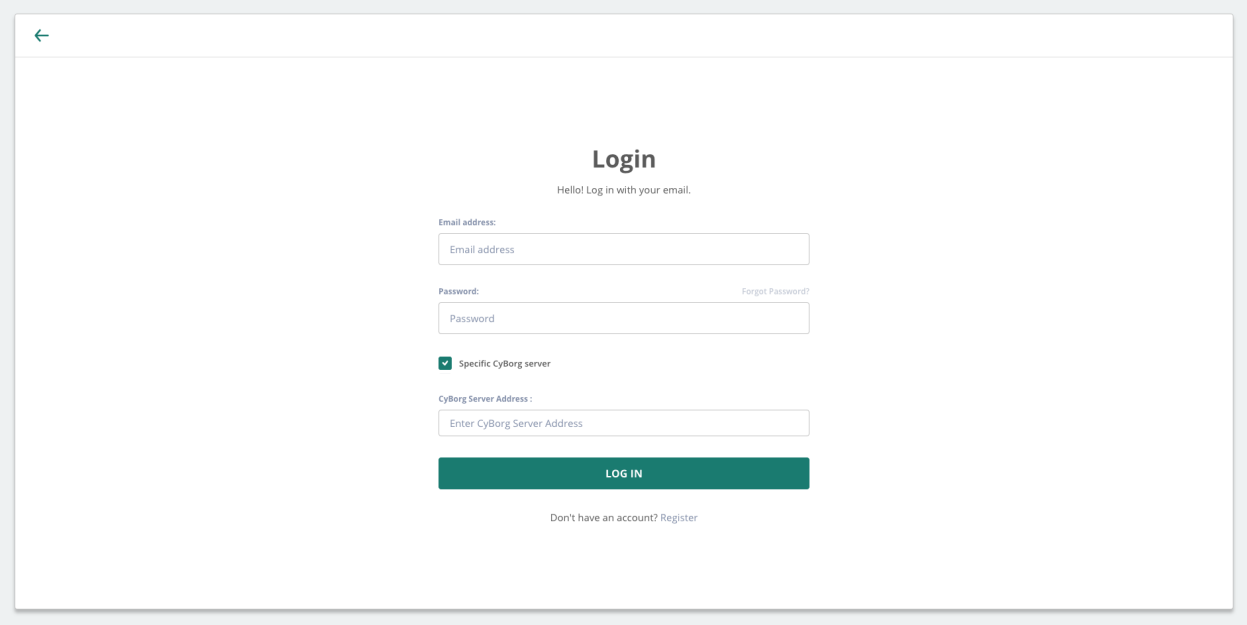
You can connect to the CyBorgBackup interface at : http://localhost:8000
Default account is :

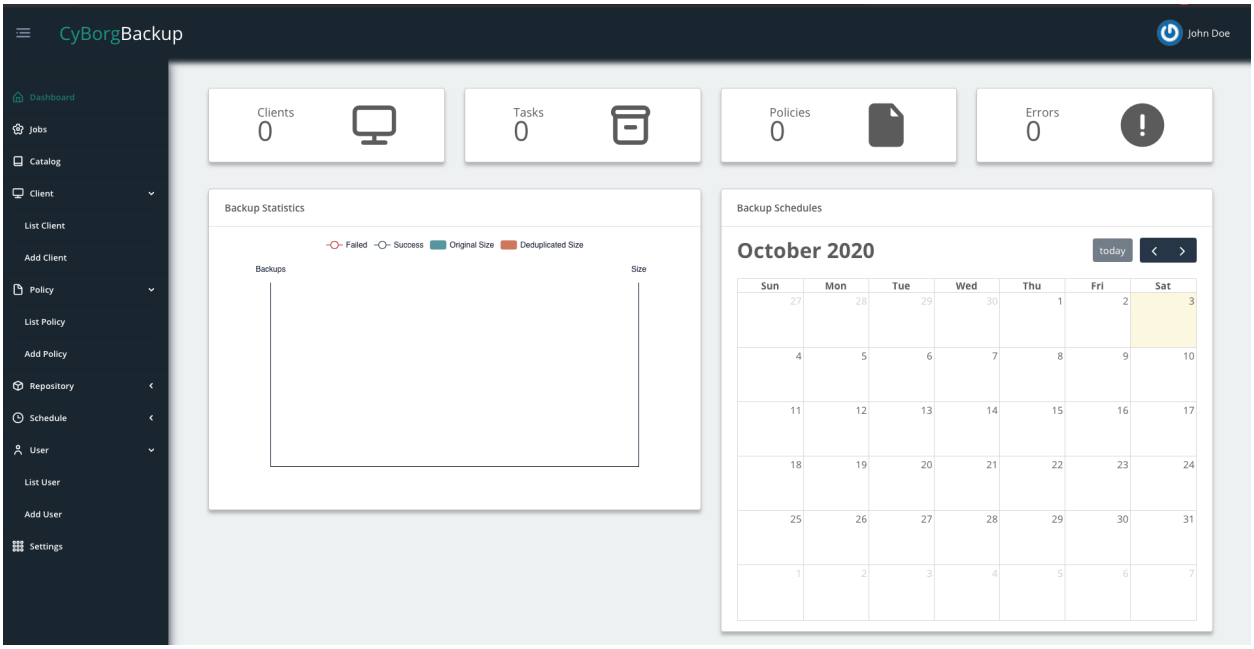
- Login : admin@cyborg.local
- Password : admin

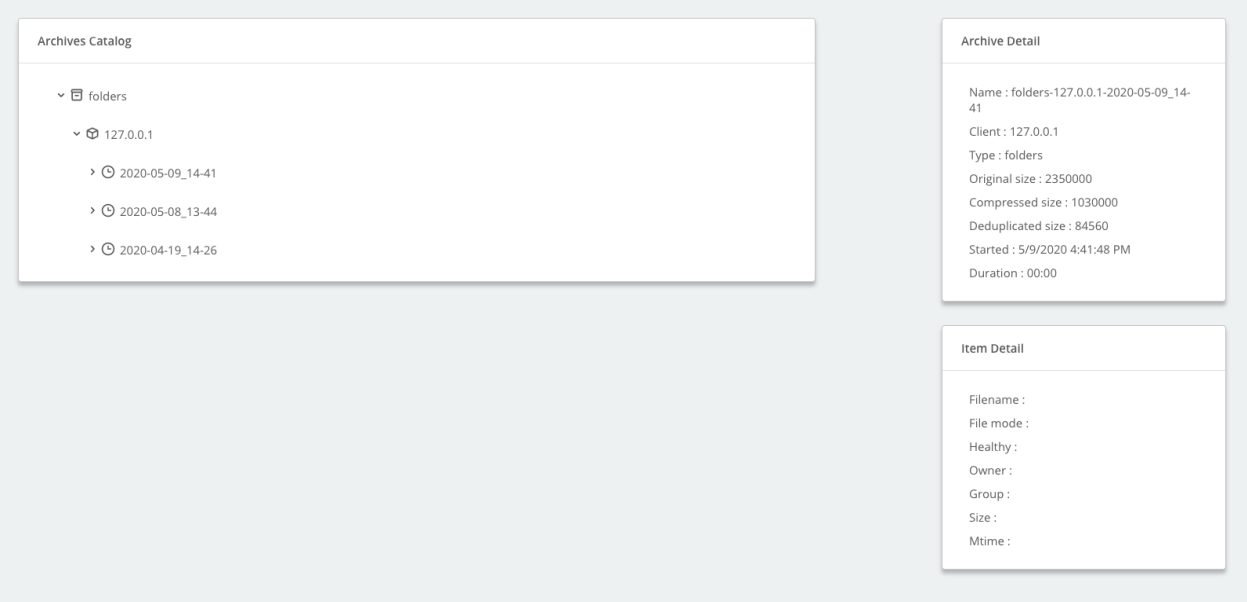
3.2 CyBorgBackup ScreenShots

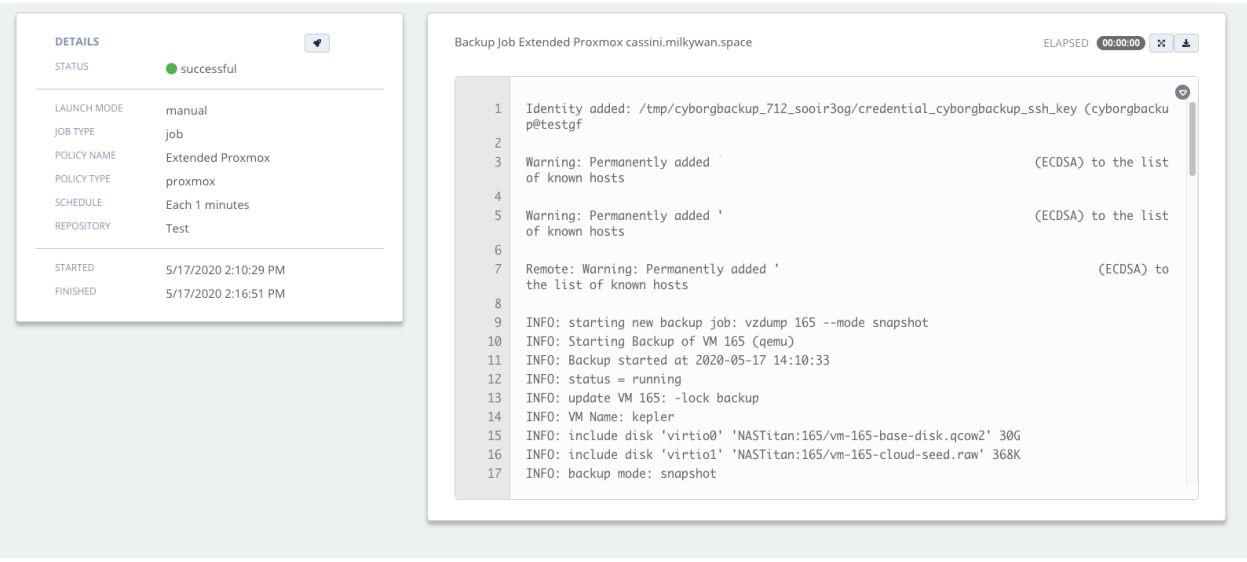


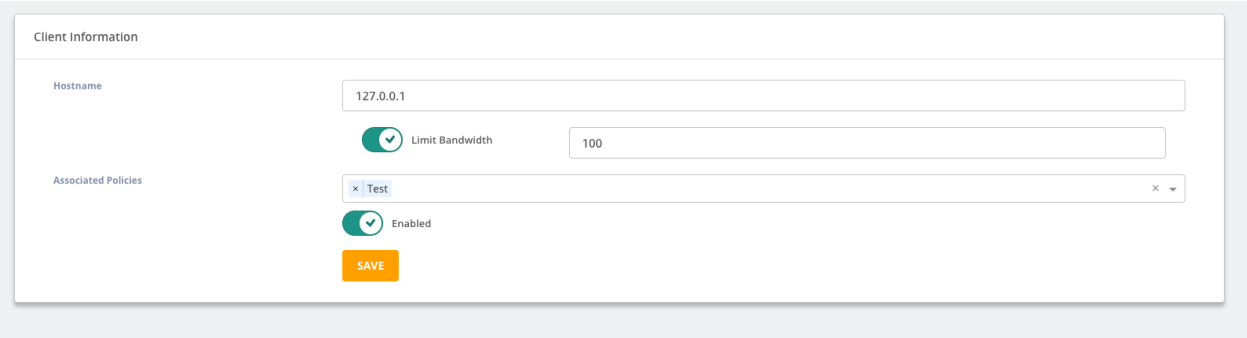


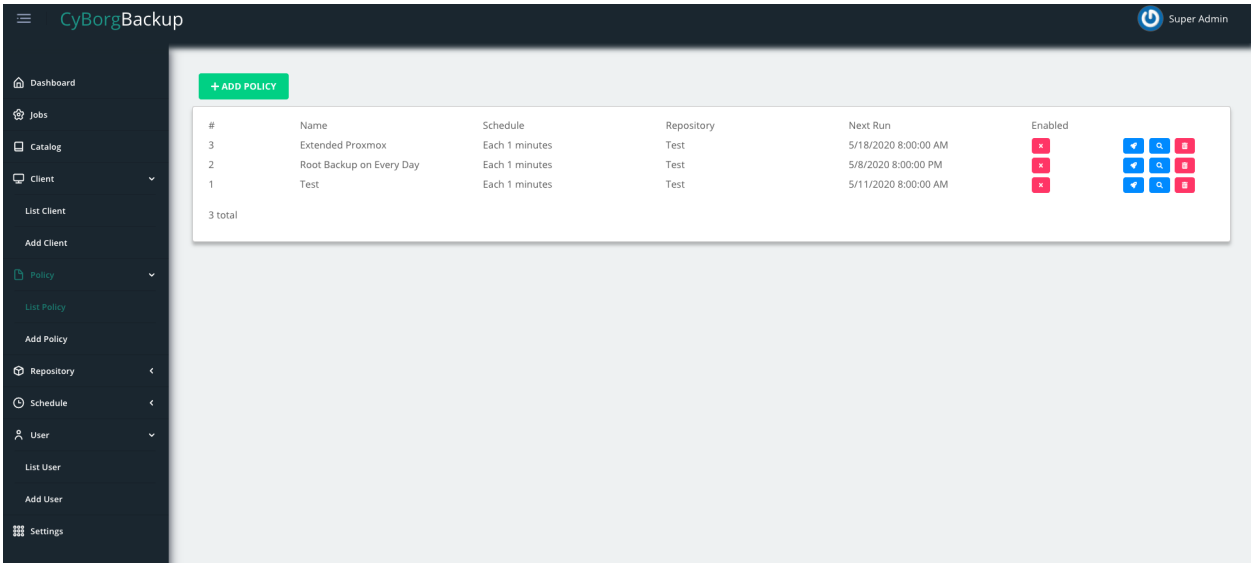

Login
Hello! Log in with your email.
Email address:
Email address
Password:
Forgot Password?
Password
Specific CyBorg server
CyBorg Server Address :
Enter CyBorg Server Address
LOG IN
Don't have an account? Register

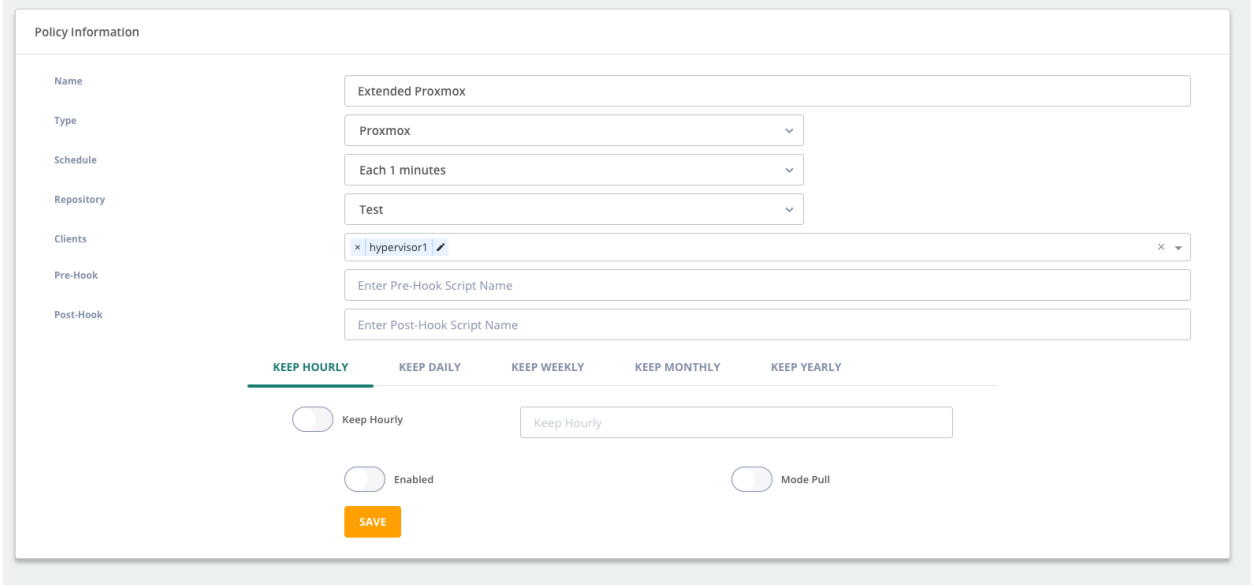

CyBorgBackup
John Doe
Dashboard
Jobs
Catalog
Client
List Client
Add Client
Policy
List Policy
Add Policy
Repository
Schedule
User
List User
Add User
Settings
Clients 0
Tasks 0
Policies 0
Errors 0
Backup Statistics
Failed
Success
Original Size
Deduplicated Size
Backups
Size
Backup Schedules
October 2020
today
Sun
Mon
Tue
Wed
Thu
Fri
Sat

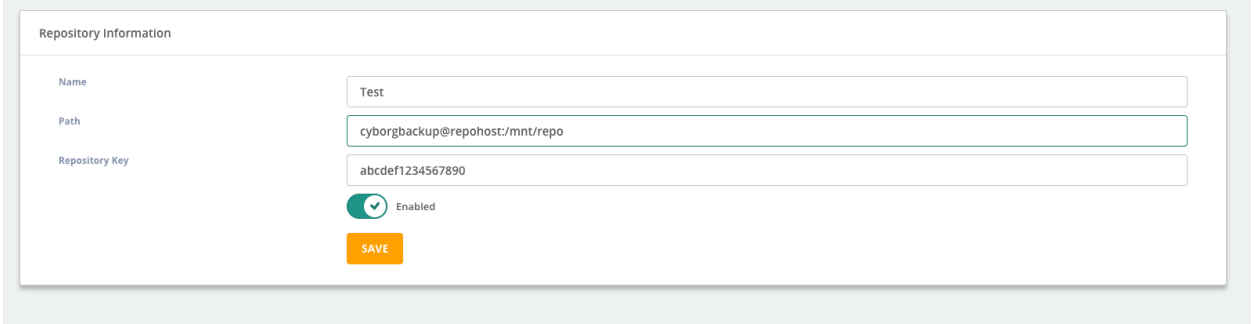

Archives Catalog
folders
127.0.0.1
2020-05-09_14-41
2020-05-08_13-44
2020-04-19_14-26
Archive Detail
Name : folders-127.0.0.1-2020-05-09_14-41
Client : 127.0.0.1
Type : folders
Original size : 2350000
Compressed size : 1030000
Deduplicated size : 84560
Started : 5/9/2020 4:41:48 PM
Duration : 00:00
Item Detail
Filename :
File mode :
Healthy :
Owner :
Group :
Size :
Mtime :

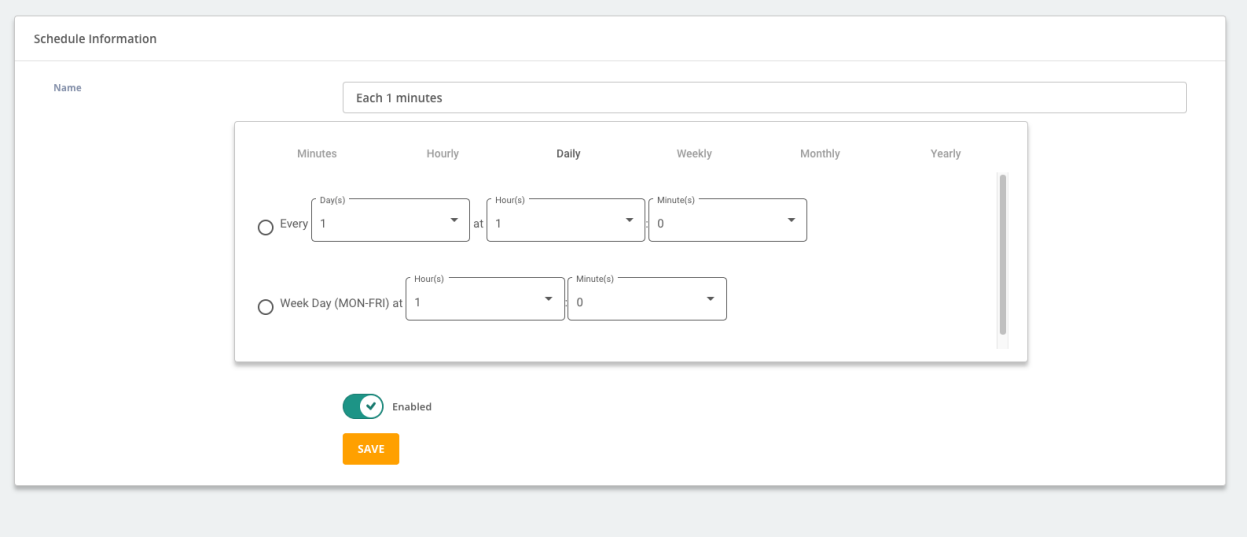

DETAILS
STATUS
successful
LAUNCH MODE
manual
JOB TYPE
job
POLICY NAME
Extended Proxmox
POLICY TYPE
proxmox
SCHEDULE
Each 1 minutes
REPOSITORY
Test
STARTED
5/17/2020 2:10:29 PM
FINISHED
5/17/2020 2:16:51 PM
Backup Job Extended Proxmox cassini.milkywan.space
ELAPSED
00:00:00
1 Identity added: /tmp/cyborgbackup_712_sooir3og/credential_cyborgbackup_ssh_key (cyborgbackup@testgf
2
3 Warning: Permanently added (ECDSA) to the list of known hosts
4
5 Warning: Permanently added ' (ECDSA) to the list of known hosts
6
7 Remote: Warning: Permanently added ' (ECDSA) to the list of known hosts
8
9 INFO: starting new backup job: vzdump 165 --mode snapshot
10 INFO: Starting Backup of VM 165 (qemu)
11 INFO: Backup started at 2020-05-17 14:10:33
12 INFO: status = running
13 INFO: update VM 165: -lock backup
14 INFO: VM Name: kepler
15 INFO: include disk 'virtio0' 'NASTitan:165/vm-165-base-disk.qcow2' 30G
16 INFO: include disk 'virtio1' 'NASTitan:165/vm-165-cloud-seed.raw' 368K
17 INFO: backup mode: snapshot

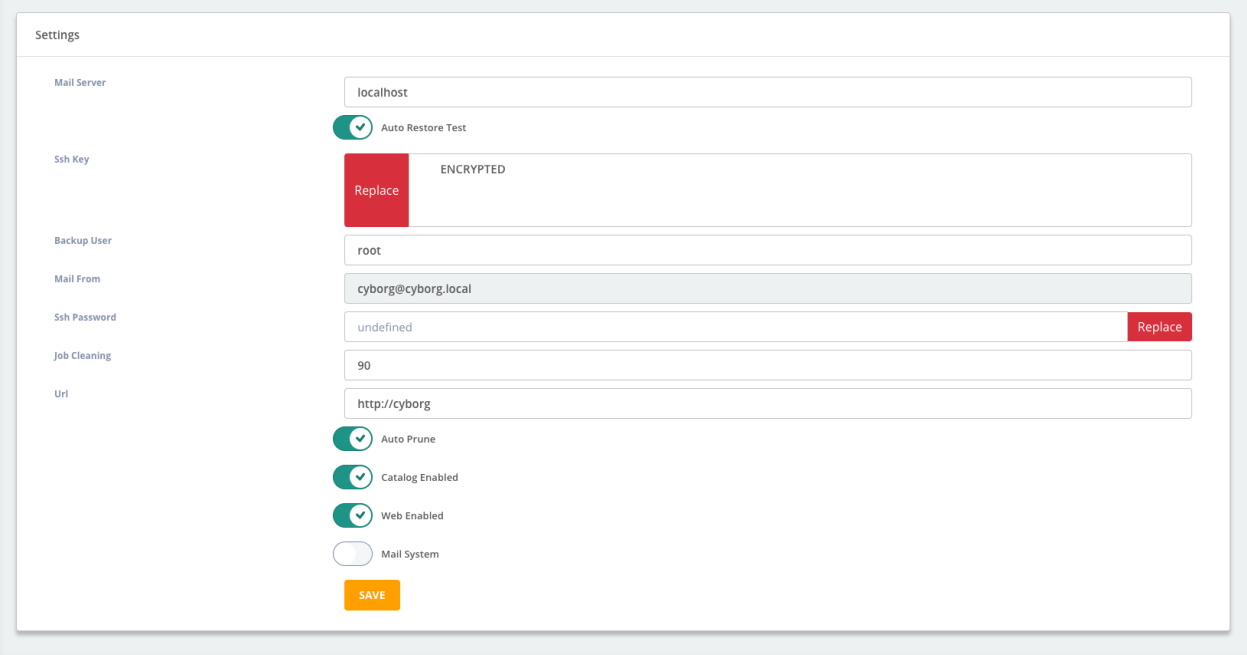

Client Information
Hostname
127.0.0.1
Limit Bandwidth
100
Associated Policies
Test
Enabled
SAVE

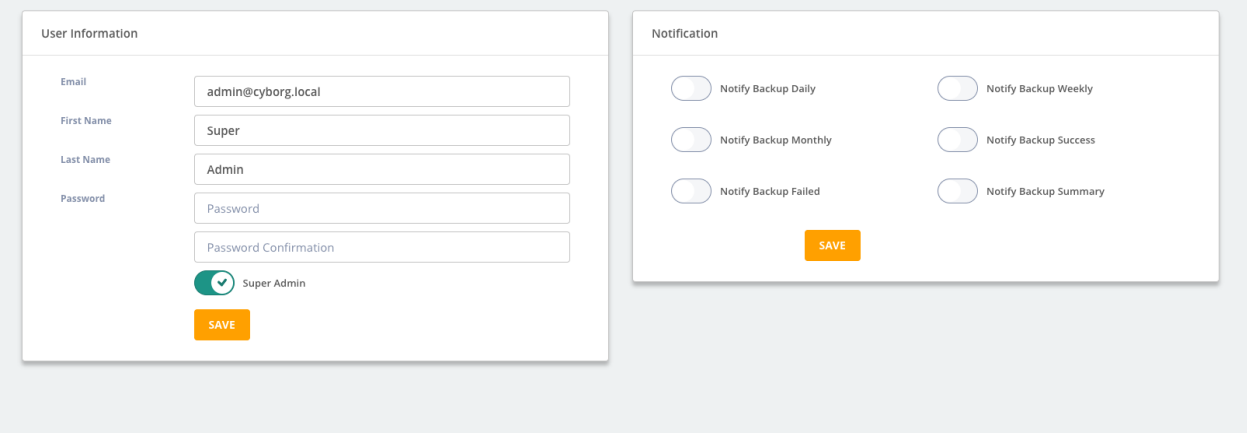

CyBorgBackup
Super Admin
Dashboard
Jobs
Catalog
Client
List Client
Add Client
Policy
List Policy
Add Policy
Repository
Schedule
User
List User
Add User
Settings
+ ADD POLICY
# Name Schedule Repository Next Run Enabled
3 Extended Proxmox Each 1 minutes Test 5/18/2020 8:00:00 AM
2 Root Backup on Every Day Each 1 minutes Test 5/8/2020 8:00:00 PM
1 Test Each 1 minutes Test 5/11/2020 8:00:00 AM
3 total

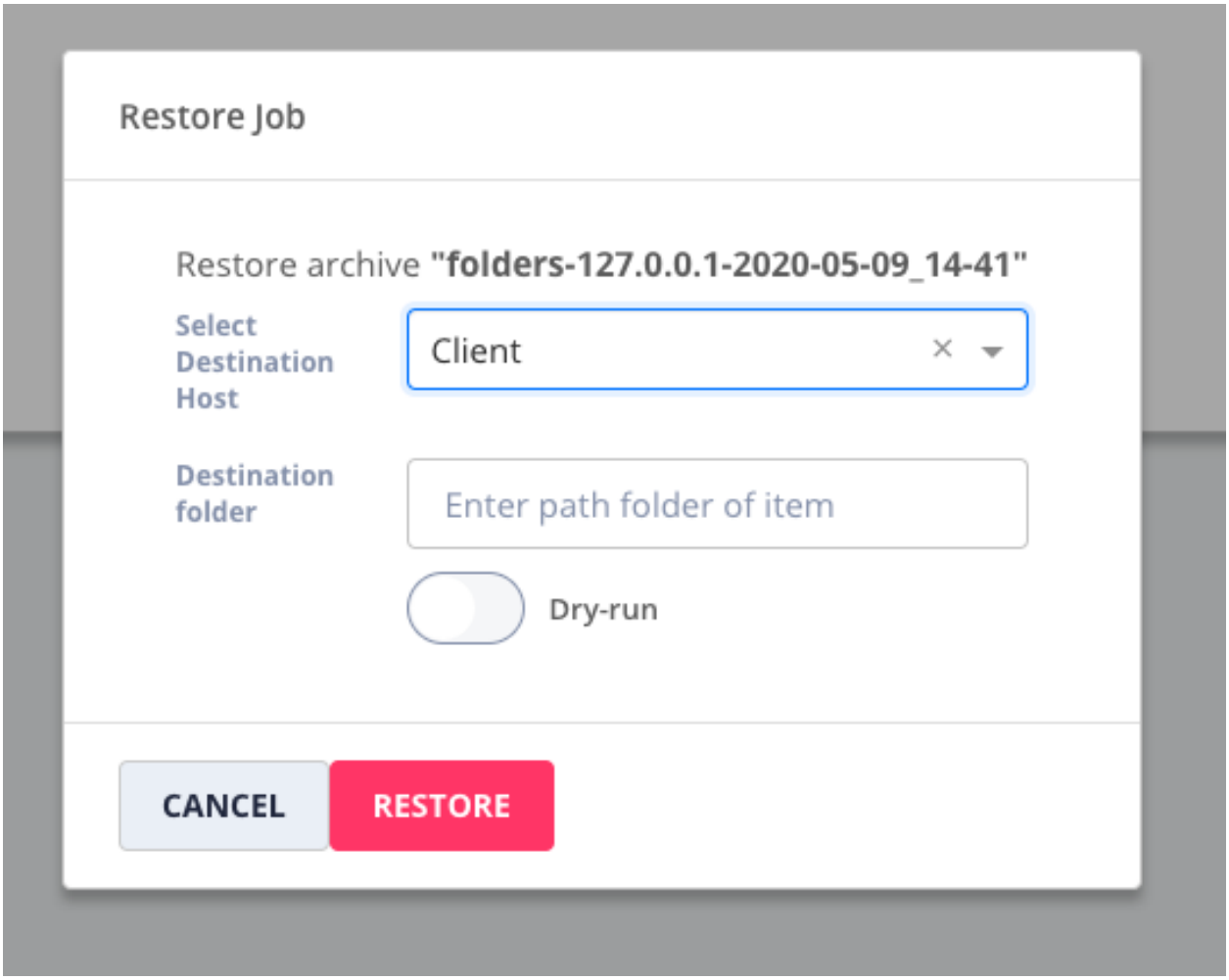

Policy Information
Name
Extended Proxmox
Type
Proxmox
Schedule
Each 1 minutes
Repository
Test
Clients
hypervisor1
Pre-Hook
Enter Pre-Hook Script Name
Post-Hook
Enter Post-Hook Script Name
KEEP HOURLY
KEEP DAILY
KEEP WEEKLY
KEEP MONTHLY
KEEP YEARLY
Keep Hourly
Keep Hourly
Enabled
Mode Pull
SAVE

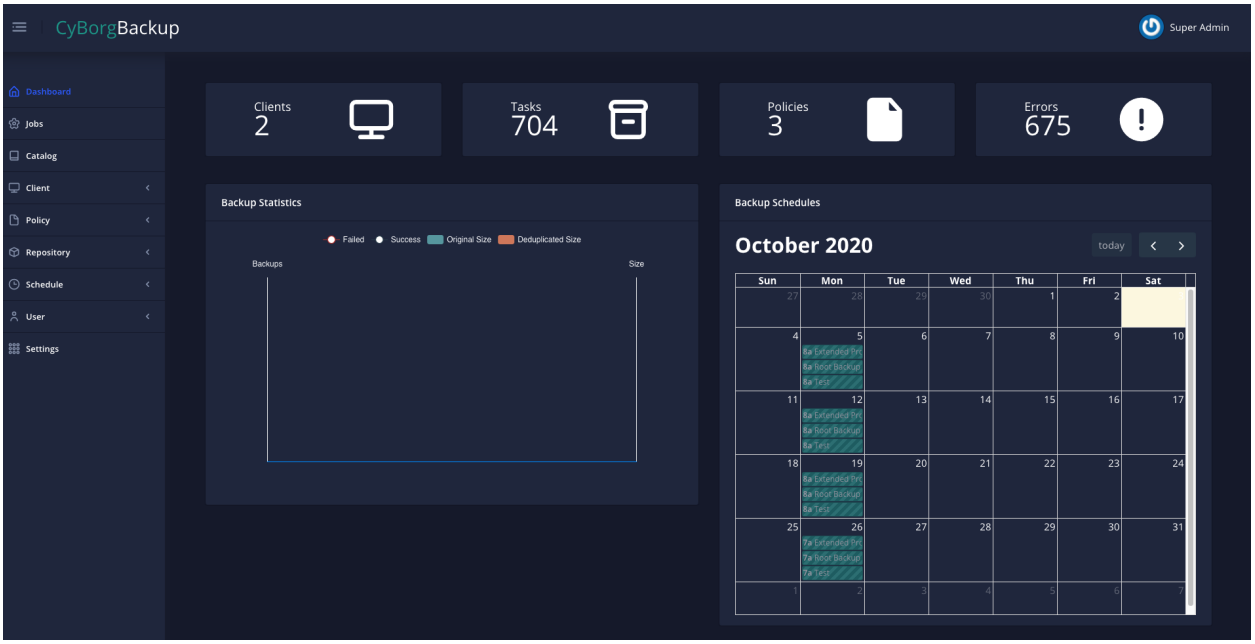

Repository Information
Name
Test
Path
cyborgbackup@repohost:/mnt/repo
Repository Key
abcdef1234567890
Enabled
SAVE

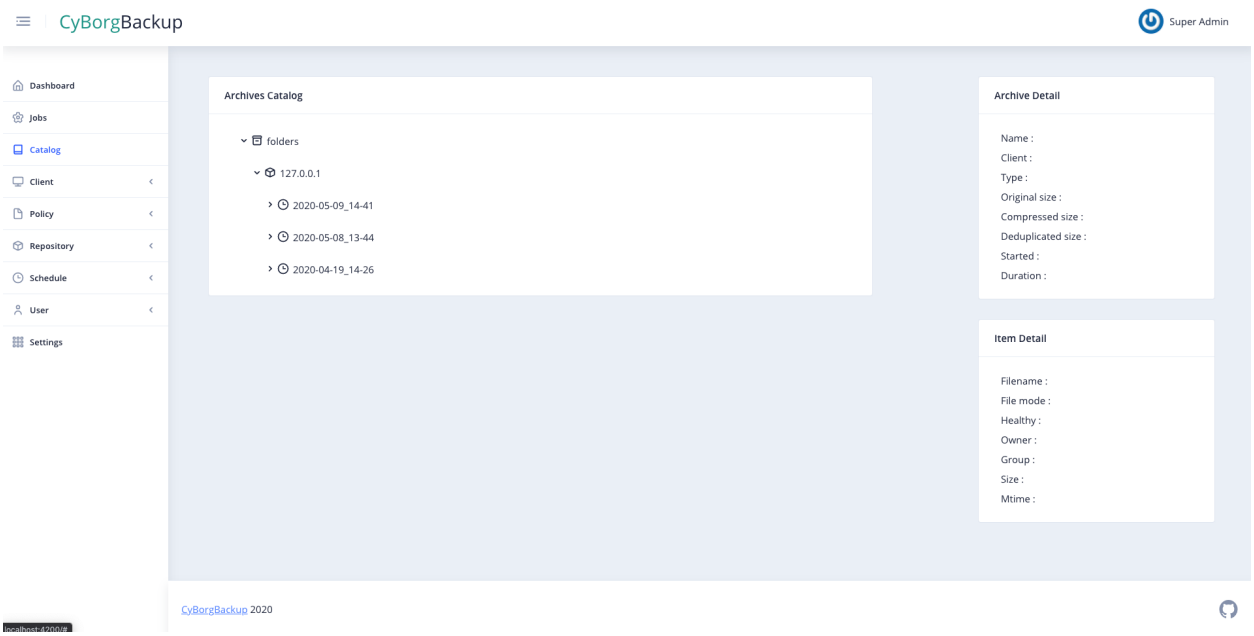

Schedule Information
Name
Each 1 minutes
Minutes
Hourly
Daily
Weekly
Monthly
Yearly
Every
Day(s)
1
at
Hour(s)
1
Minute(s)
0
Week Day (MON-FRI) at
Hour(s)
1
Minute(s)
0
Enabled
SAVE


Settings
Mail Server
localhost
Auto Restore Test
Ssh Key
Replace
ENCRYPTED
Backup User
root
Mail From
cyborg@cyborg.local
Ssh Password
undefined
Replace
Job Cleaning
90
Url
http://cyborg
Auto Prune
Catalog Enabled
Web Enabled
Mail System
SAVE


User Information
Email
admin@cyborg.local
First Name
Super
Last Name
Admin
Password
Password
Password Confirmation
Super Admin
SAVE
Notification
Notify Backup Daily
Notify Backup Weekly
Notify Backup Monthly
Notify Backup Success
Notify Backup Failed
Notify Backup Summary
SAVE


Restore Job
Restore archive "folders-127.0.0.1-2020-05-09_14-41"
Select Destination Host
Client
Destination folder
Enter path folder of item
Dry-run
CANCEL
RESTORE


CyBorgBackup
Super Admin
Dashboard
Jobs
Catalog
Client
Policy
Repository
Schedule
User
Settings
Clients 2
Tasks 704
Policies 3
Errors 675
Backup Statistics
Failed
Success
Original Size
Deduplicated Size
Backups
Size
Backup Schedules
October 2020
today
Sun
Mon
Tue
Wed
Thu
Fri
Sat



3.3 Quickstart

Eager to get started? This page gives a good introduction in how to get started with CyBorgBackup.

3.3.1 CyBorgBackup System

CyBorgBackup use some framework to give a system with capabilities of backup system based on BorgBackup with ease.

The Web Interface and API populate the Policy, Schedule, Repository, Client and Job object.

We use Celery Beat to manage the schedule of internal task of CyBorgBackup like task_manager and to notify celery to start task. Task_manager is responsible to launch job if he's ready and not blocked by another job.

Celery is responsible to execute task like task_manager or other. It's the element who execute ssh connection and execute borg backup.

Channels Workers is responsible of the websocket message between each element and the web interface.

Callback receivers all stdout messages from celery program output to put them in database and emit signal under the websocket.

3.3.2 Object Relation

A schedule describe the policy schedule. They can be affected to multiple policies. The schedule is describe using the crontab format.

A repository describe the borg backup repository with the path and the repository encryption key. The path must be a valid uri like SCP uri.

A client describe a backup client with their hostname only. IP and Borg Version are filled by the preparation script.

A policy is the element who made the relation between each element. A policy have a schedule, a repository, a type, and some clients. We can also define the retention policy of each client backup. They can also launch post ou pre hook.

A job is the backup job based on a policy and a client. When you launch a backup job, the system create as many jobs as there are client defined in the policy.

Depending of the settings value, some job are created in dependencies of them like catalog job, prepare job and prune job.

3.3.3 Enabled object

Each object have a enabled/disabled field. This field enable the object in each relation. For example, if you disable a schedule, each policies who used this schedule will be unusable. The same for the client or the repository.

3.3.4 Preparation job

Client and repository came with a preparation job to install and prepare borg. On the preparation script, the final step is to prevent CyBorgBackup system that the object is ready to be used. If the hook didn't work, the job stdout show the curl command to launch to enable the object. Or you can use the API to set the ready field to true.

Warning: The URL settings must be defined correctly with an accessible URL of each client of repository nodes.

3.3.5 Policy Type

At this time, 9 policy types has been defined in CyBorgBackup.

rootfs

The rootfs policy backup all files present in the root directory of the server except some useless file like /dev, /proc and other.

vm

The vm policy type backup directly the Virtual Machine using the hypervisor. They will backup the hard drive device of the virtual machine.

mysql

The mysql policy type will create a mysql dump and backup them using Borg.

You can to specify user,password and database/s in extra vars:

```
{"user":"backupuser","password":"backupass"}
```

By default, mysql policy type will backup all databases. To specify a database, you need to add database entry in extra_vars:

```
{"databases":"mydb"}
```

To specify multiple databases, you need to add databases list in extra_vars:

```
{"databases":["mydb1","mydb2"]}
```

Command to create backup user on MySQL:

```
GRANT LOCK TABLES, SELECT ON *.* TO 'backupuser'@'%' identified by 'backuppass';
FLUSH PRIVILEGES
```

postgresql

The postgresql policy type will create a postgresql dump and backup them using Borg.

You can to specify database in extra vars:

```
{"database":"mydb"}
```

Command to create backup user on PostgreSQL:

```
CREATE USER backupuser SUPERUSER password 'backuppass';
ALTER USER cyborgbackup set default_transaction_read_only = on;
```

piped

The piped policy type permit to launch a command on the client and backup the output of the command to Borg.

You need to specify extra vars with piped command:

```
{"command":"mypipedcommand"}
```

config

The config policy type will backup only the /etc folder of the server.

mail

The mail policy type will backup only the /var/mail or /var/lib/mail folder of the server.

folders

The folders policy type will backup specified folder of the server. You need to specify extra vars with piped command:

```
{"folders":["folder1","folder2"]}
```

proxmox

The folders policy type will backup specified folder of the server. You need to specify extra vars with piped command:

```
{"folders":["folder1","folder2"]}
```

3.4 Configuration

CyBorgBackup need some configuration before be ready to use. They will be defined in the next section.

3.4.1 Settings

Under the settings page, you must configure the following element :

- URL => Contains the CyBorgBackup accessible URL. It will be use by client or repository node for the preparation step.
- Mail Server => You must configure the Mail server to use the mail report of CyBorgBackup
- SSH Key => An ssh key pair must be configured in CyBorgBackup. CyBorgBackup can generate them for you. He will use them to connect on each client and on each repository. The public key must also be configured on each authorized_keys ssh file manually.

Warning: The SSH Key must be protected with a password. The password will be encrypted before stored under CyBorgBackup database.

Warning: The SSH Public Key must be configured on the "Backup User" authorized_keys file. If backup user is different of "root", CyBorgBackup will use 'sudo' command for the backup

Note: If you have a specific configuration for SSH, different port or ciphers for example. You can use the **.ssh/config** file on the CyBorgBackup folder.

3.4.2 Repositories

A repository must be defined to start backup. The path must be in URI format as the following : **user@fqdn:path**

Warning: Same as the SSH Key configuration, the user defined in the path must have the CyBorgBackup SSK Key configured in authorized_keys file.

3.4.3 Schedules

The schedule system use the CRON format.

3.4.4 Clients

To use the CyBorgBackup system, you must create your first client. The hostname must be a resolvable hostname by CyBorgBackup. It will use them to connect using SSH protocol.

3.4.5 Policies

The policy is the element for permit to CyBorgBackup to backup client. It's a relation between a schedule, a repository and clients.

Pre-hook and post-hook entries are script or command launch of each client. The script or command must be exist on them. CyBorgBackup will not pull or install them.

The "Keep" section is used to specify on many archive will be keep by the system before pruning. For example, if you configure 7 Keep Daily and 4 Keep Weekly, CyBorgBackup will keep archives of the last 7 days made each day and one archive each week for the last 4 weeks.

By default, CyBorgbackup will connect on each client and each client will push the backup to the repository using ssh connection. The "Pull Mode" permit to change the method, CyBorgBackup will connect on the repository node and from them, connect to each client. Be careful, the client need to connect to the repository node.

3.4.6 Ready ?

Settings, Schedule, Repository, Client and Policy are configured ? CyBorgBackup is now ready to backup them. For each new repository or client, CyBorgBackup will launch a preparation script before backup them.

You can now launch a policy to launch the backup workflow or wait for the schedule.

3.5 VM Module Provider

CyBorgBackup can use custom VM module provider to backup virtual machine based on hypervisor.

The module need to respect some prerequisites to be used by CyBorgBackup.

3.5.1 Module Name

This function return the module name displayed by CyBorgBackup:

```
def module_name():
    return 'Proxmox'
```

3.5.2 Module Type

This function return the policy type code used by CyBorgBackup to identify the utility of this module. For VM backup provider, it must be set to 'vm':

```
def module_type():
    return 'vm'
```

3.5.3 Get Client

This function return the hostname of the hypervisor of the VM. It will be used to connect them and launch backup script

```
def get_client(client):
    return 'hypervisor.example.com'
```

3.5.4 Get Script

This function return a string that represent the script send to the hypervisor and used to backup the virtual machine. The script must return data on stdout. Data received will be directly send to borg create:

```
def get_script():
    return '''#!/bin/bash
echo "Hello World"
'''
```

3.5.5 Example Proxmox Script

You will find bellow an example script used to backup Proxmox VirtualMachine from her hypervisor

CHAPTER 4

The API Documentation / Guide

If you are looking for information on a specific function, class, or method, this part of the documentation is for you.

4.1 CyBorgBackup API

Actually, only the V1 version of the API is available. They can be access on the url /api of the webserver and the V1 api on the url /api/v1.

4.1.1 Main API V1

GET /api/v1
Retrieve all available submodule of the API.

Example request:

```
$ curl https://cyborgbackup.local/api/v1/
```

Example response:

```
{
    "ping": "/api/v1/ping",
    "config": "/api/v1/config/",
    "me": "/api/v1/me/"
}
```

GET /api/v1/ping
Test the api and get the version

```
{
    "version": "1.0"
    "ping": "pong"
}
```

GET /api/v1/config
Retrieve some configuration of the CyBorgBackup instance.

```
{
    "version": "1.0"
    "debug": true,
    "allowed_hosts" : ["127.0.0.1"]
}
```

GET /api/v1/me
Retrieve information about the current logged user.

```
{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
        "id": 1,
        "type": "user",
        "url": "/api/v1/users/1/",
        "related": {},
        "summary_fields": {},
        "created": "2018-11-08T16:13:29.370148Z",
        "first_name": "",
        "last_name": "",
        "email": "admin@milkywan.fr",
        "is_superuser": true
    ]
}
```

4.1.2 Users API V1

GET /api/v1/users/
Retrieve a list of all users.

```
{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [USERS]
}
```

Response JSON Object

- **next** (*string*) – URI for next set of Users.
- **previous** (*string*) – URI for previous set of Users.
- **count** (*integer*) – Total number of Users.
- **results** (*array*) – Array of `Users` objects.

GET /api/v1/users/(int: *id*)/
Retrieve details of a single user.

```
{
    "id": 3,
    "type": "user",
    "url": "/api/v1/users/3/",
    "related": {},
    "summary_fields": {},
    "created": "2018-11-11T19:43:24.261706Z",
    "first_name": "",
    "last_name": "",
    "email": "cyborg@agent.local",
    "is_superuser": true
}
```

Response JSON Object

- **id** (*integer*) – The ID of the user
- **type** (*string*) – The object type under cyborgbackup system.
- **url** (*string*) – The URL access of the user object.
- **related** (*dict*) – Related property of mapped object
- **summary_fields** (*dict*) – Some summary field of object relation
- **created** (*string*) – The creation date of the user
- **first_name** (*string*) – First name of the user
- **last_name** (*string*) – Last name of the user
- **email** (*string*) – Email of the user
- **is_superuser** (*boolean*) – Super User

Status Codes

- 200 OK – no error
- 404 Not Found – There is no `User` with this ID

POST /api/v1/users/

Create a single user.

```
{
    "first_name": "",
    "last_name": "",
    "email": "cyborg@agent.local",
    "is_superuser": true
}
```

Response JSON Object

- **first_name** (*string*) – First name of the user
- **last_name** (*string*) – Last name of the user
- **email** (*string*) – Email of the user
- **is_superuser** (*boolean*) – Super User

Status Codes

- 200 OK – no error

- 404 Not Found – There is no `User` with this ID

PATCH /api/v1/users/(**int:** *id*)**/**
Update a single user.

```
{
    "first_name": "",
    "last_name": "",
    "email": "cyborg@agent.local",
    "is_superuser": true
}
```

Response JSON Object

- **id** (*integer*) – The ID of the user
- **first_name** (*string*) – First name of the user
- **last_name** (*string*) – Last name of the user
- **email** (*string*) – Email of the user
- **is_superuser** (*boolean*) – Super User

Status Codes

- 200 OK – no error
- 404 Not Found – There is no `User` with this ID

DELETE /api/v1/users/(**int:** *id*)**/**
Delete a single user.

Status Codes

- 200 OK – no error
- 404 Not Found – There is no `User` with this ID

4.1.3 Clients API V1

GET /api/v1/clients/
Retrieve a list of all clients.

```
{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [CLIENTS]
}
```

Response JSON Object

- **next** (*string*) – URI for next set of Clients.
- **previous** (*string*) – URI for previous set of Clients.
- **count** (*integer*) – Total number of Clients.
- **results** (*array*) – Array of `Clients` objects.

GET /api/v1/clients/(int: *id*)/
Retrieve details of a single client.


```
{
    "id": 1,
    "type": "client",
    "url": "/api/v1/clients/1/",
    "related": {},
    "summary_fields": {},
    "created": "2018-11-22T18:17:51.831221Z",
    "modified": "2018-11-22T19:21:16.011127Z",
    "created_by": null,
    "modified_by": null,
    "hostname": "lab.example.com",
    "ip": "",
    "version": "",
    "ready": false,
    "hypervisor_ready": false,
    "hypervisor_name": "",
    "enabled": true,
    "uuid": "fa3462e3-57da-430e-bca5-3bc60d4ba5a2"
}
```


Response JSON Object

- **id** (*integer*) – The ID of the client
- **type** (*string*) – The object type under cyborgbackup system.
- **url** (*string*) – The URL access of the client object.
- **related** (*dict*) – Related property of mapped object
- **summary_fields** (*dict*) – Some summary field of object relation
- **created** (*string*) – The creation date of the client
- **modified** (*string*) – The modification date of the client
- **created_by** (*string*) – User responsible of the creation of the client
- **modified_by** (*string*) – User responsible of the last modification
- **hostname** (*string*) – Client Hostname
- **ip** (*string*) – IP Addresses of the client
- **version** (*string*) – Borg Client Version
- **ready** (*boolean*) – Client prepared to be use with borg
- **hypervisor_name** (*string*) – Hypervisor name of the client
- **hypervisor_ready** (*boolean*) – Hypervisor prepared to be use with borg
- **enabled** (*boolean*) – Client enabled
- **uuid** (*string*) – Auto generated UUID

Status Codes

- 200 OK – no error
- 404 Not Found – There is no `Client` with this ID

POST /api/v1/clients/

Create a single client.

```
{
    "hostname": "lab.example.com",
    "ip": "",
    "version": "",
    "ready": false,
    "hypervisor_ready": false,
    "hypervisor_name": "",
    "enabled": true,
}
```

Response JSON Object

- **hostname** (*string*) – Client Hostname
- **ip** (*string*) – IP Addresses of the client
- **version** (*string*) – Borg Client Version
- **ready** (*boolean*) – Client prepared to be use with borg
- **hypervisor_name** (*string*) – Hypervisor name of the client
- **hypervisor_ready** (*boolean*) – Hypervisor prepared to be use with borg
- **enabled** (*boolean*) – Client enabled

Status Codes

- 200 OK – no error
- 404 Not Found – There is no `Client` with this ID

PATCH /api/v1/clients/(int: *id*)/

Update a single client.

```
{
    "hostname": "lab.example.com",
    "ip": "",
    "version": "",
    "ready": false,
    "hypervisor_ready": false,
    "hypervisor_name": "",
    "enabled": true
}
```

Response JSON Object

- **id** (*integer*) – The ID of the client
- **hostname** (*string*) – Client Hostname
- **ip** (*string*) – IP Addresses of the client
- **version** (*string*) – Borg Client Version
- **ready** (*boolean*) – Client prepared to be use with borg
- **hypervisor_name** (*string*) – Hypervisor name of the client
- **hypervisor_ready** (*boolean*) – Hypervisor prepared to be use with borg

- **enabled** (*boolean*) – Client enabled

Status Codes

- 200 OK – no error
- 404 Not Found – There is no `Client` with this ID

DELETE /api/v1/clients/(int: *id*)/
Delete a single client.

Status Codes

- 200 OK – no error
- 404 Not Found – There is no `Client` with this ID

4.1.4 Schedules API V1

GET /api/v1/schedules/
Retrieve a list of all schedules.

```
{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [SCHEDULES]
}
```

Response JSON Object

- **next** (*string*) – URI for next set of Schedules.
- **previous** (*string*) – URI for previous set of Schedules.
- **count** (*integer*) – Total number of Schedules.
- **results** (*array*) – Array of `Schedule` objects.

GET /api/v1/schedules/(int: *id*)/
Retrieve details of a single schedule.

```
{
    "id": 1,
    "type": "schedule",
    "url": "/api/v1/schedules/1/",
    "related": {},
    "summary_fields": {},
    "created": "2018-11-22T18:17:51.831221Z",
    "modified": "2018-11-22T19:21:16.011127Z",
    "created_by": null,
    "modified_by": null,
    "name": "Every Minutes",
    "crontab": "*/1 * * * * *",
    "enabled": true,
    "uuid": "fa3462e3-57da-430e-bca5-3bc60d4ba5a2"
}
```

Response JSON Object

- **id** (*integer*) – The ID of the schedule
- **type** (*string*) – The object type under cyborgbackup system.
- **url** (*string*) – The URL access of the schedule object.
- **related** (*dict*) – Related property of mapped object
- **summary_fields** (*dict*) – Some summary field of object relation
- **created** (*string*) – The creation date of the schedule
- **modified** (*string*) – The modification date of the schedule
- **created_by** (*string*) – User responsible of the creation of the schedule
- **modified_by** (*string*) – User responsible of the last modification
- **name** (*string*) – Schedule name
- **crontab** (*string*) – Crontab schedule
- **enabled** (*boolean*) – Schedule enabled
- **uuid** (*string*) – Auto generated UUID

Status Codes

- 200 OK – no error
- 404 Not Found – There is no `Schedule` with this ID

POST /api/v1/schedules/

Create a single schedule.

```
{
    "name": "Every Minutes",
    "crontab": "*/1 * * * * *",
    "enabled": true
}
```

Response JSON Object

- **name** (*string*) – Schedule name
- **crontab** (*string*) – Crontab schedule
- **enabled** (*boolean*) – Schedule enabled

Status Codes

- 200 OK – no error
- 404 Not Found – There is no `Schedule` with this ID

PATCH /api/v1/schedules/(int: *id*)/

Update a single schedule.

```
{
    "name": "Every Monday",
    "crontab": "0 5 * * MON *",
    "enabled": true
}
```

Response JSON Object

- **id** (*integer*) – The ID of the schedule
- **name** (*string*) – Schedule name
- **crontab** (*string*) – Crontab schedule
- **enabled** (*boolean*) – Schedule enabled

Status Codes

- 200 OK – no error
- 404 Not Found – There is no `Schedule` with this ID

DELETE /api/v1/schedules/(int: *id*)/
Delete a single schedule.

Status Codes

- 200 OK – no error
- 404 Not Found – There is no `Schedule` with this ID

4.1.5 Repositories API V1

GET /api/v1/repositories/
Retrieve a list of all repositories.

```
{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [REPOSITORIES]
}
```

Response JSON Object

- **next** (*string*) – URI for next set of Repositories.
- **previous** (*string*) – URI for previous set of Repositories.
- **count** (*integer*) – Total number of Repositories.
- **results** (*array*) – Array of `Repository` objects.

GET /api/v1/repositories/(int: *id*)/
Retrieve details of a single repository.

```
{
    "id": 1,
    "type": "repository",
    "url": "/api/v1/repositories/1/",
    "related": {},
    "summary_fields": {},
    "created": "2018-11-22T18:17:51.831221Z",
    "modified": "2018-11-22T19:21:16.011127Z",
    "created_by": null,
    "modified_by": null,
    "name": "Main Repo",
    "path": "cyborgbackup@backup:/repository",
```

(continues on next page)

(continued from previous page)

```
    "repository_key": "0123456789abcdef",
    "original_size": 722,
    "compressed_size": 747,
    "deduplicated_size": 747,
    "ready": true,
    "enabled": true,
    "uuid": "fa3462e3-57da-430e-bca5-3bc60d4ba5a2"
}
```

Response JSON Object

- **id** (*integer*) – The ID of the repository
- **type** (*string*) – The object type under cyborgbackup system.
- **url** (*string*) – The URL access of the repository object.
- **related** (*dict*) – Related property of mapped object
- **summary_fields** (*dict*) – Some summary field of object relation
- **created** (*string*) – The creation date of the repository
- **modified** (*string*) – The modification date of the repository
- **created_by** (*string*) – User responsible of the creation of the repository
- **modified_by** (*string*) – User responsible of the last modification
- **name** (*string*) – Repository name
- **path** (*string*) – URI path to access the repository from each client
- **repository_key** (*string*) – Key used to encrypt the repository
- **original_size** (*integer*) – Calculated size of all archives
- **compressed_size** (*integer*) – Calculated compressed size of all archives
- **deduplicated_size** (*integer*) – Calculated deduplicated size of all archives
- **ready** (*boolean*) – Repository prepared to be use with borg
- **enabled** (*boolean*) – Repository enabled
- **uuid** (*string*) – Auto generated UUID

Status Codes

- 200 OK – no error
- 404 Not Found – There is no `Repository` with this ID

POST /api/v1/repositories/

Create a single repository.

```
{
    "name": "Main Repo",
    "path": "cyborgbackup@backup:/repository",
    "repository_key": "0123456789abcdef",
    "ready": true,
    "enabled": true
}
```

Response JSON Object

- **name** (*string*) – Repository name
- **path** (*string*) – URI path to access the repository from each client
- **repository_key** (*string*) – Key used to encrypt the repository
- **ready** (*boolean*) – Repository prepared to be use with borg
- **enabled** (*boolean*) – Repository enabled

Status Codes

- 200 OK – no error
- 404 Not Found – There is no `Repository` with this ID

PATCH /api/v1/repositories/(int: *id*)/

Update a single repository.

```
{
    "name": "Main Repo",
    "path": "cyborgbackup@backup:/repository",
    "repository_key": "0123456789abcdef",
    "ready": true,
    "enabled": true
}
```

Response JSON Object

- **id** (*integer*) – The ID of the repository
- **name** (*string*) – Repository name
- **path** (*string*) – URI path to access the repository from each client
- **repository_key** (*string*) – Key used to encrypt the repository
- **ready** (*boolean*) – Repository prepared to be use with borg
- **enabled** (*boolean*) – Repository enabled

Status Codes

- 200 OK – no error
- 404 Not Found – There is no `Repository` with this ID

DELETE /api/v1/repositories/(int: *id*)/

Delete a single repository.

Status Codes

- 200 OK – no error
- 404 Not Found – There is no `Repository` with this ID

4.1.6 Policies API V1

GET /api/v1/policies/

Retrieve a list of all policies.

```
{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [POLICIES]
}
```

Response JSON Object

- **next** (*string*) – URI for next set of Policies.
- **previous** (*string*) – URI for previous set of Policies.
- **count** (*integer*) – Total number of Policies.
- **results** (*array*) – Array of `Policy` objects.

GET /api/v1/policies/(int: *id*)/

Retrieve details of a single policy.

```
{
    "id": 1,
    "type": "policy",
    "url": "/api/v1/policies/1/",
    "related": {
        "launch": "/api/v1/policies/1/launch/",
        "calendar": "/api/v1/policies/1/calendar/",
        "schedule": "/api/v1/schedules/1/",
        "repository": "/api/v1/repositories/1/"
    },
    "summary_fields": {
        "repository": {
            "id": 1,
            "name": "Main Repo",
            "path": "cyborgbackup@backup:/repository"
        },
        "schedule": {
            "id": 1,
            "name": "Each Monday",
            "crontab": "0 5 * * MON *"
        }
    },
    "created": "2018-11-22T18:51:22.894984Z",
    "modified": "2018-11-23T20:54:51.013495Z",
    "created_by": null,
    "modified_by": null,
    "uuid": "3a67b010-bbc4-43de-937e-11270c710aad",
    "name": "Full Features",
    "extra_vars": "",
    "clients": [
        1
    ],
    "repository": 1,
    "schedule": 1,
    "policy_type": "vm",
    "keep_hourly": 1,
    "keep_yearly": null,
    "keep_daily": null,
```

(continues on next page)

(continued from previous page)

```
    "keep_weekly": null,
    "keep_monthly": null,
    "vmprovider": "proxmox",
    "next_run": "2018-11-26T05:00:00Z",
    "mode_pull": false,
    "enabled": true
}
```

Response JSON Object

- **id** (*integer*) – The ID of the policy
- **type** (*string*) – The object type under cyborgbackup system.
- **url** (*string*) – The URL access of the policy object.
- **related** (*dict*) – Related property of mapped object
- **summary_fields** (*dict*) – Some summary field of object relation
- **created** (*string*) – The creation date of the policy
- **modified** (*string*) – The modification date of the policy
- **created_by** (*string*) – User responsible of the creation of the policy
- **modified_by** (*string*) – User responsible of the last modification
- **name** (*string*) – Policy name
- **extra_vars** (*string*) – JSON Dictionnary of variable used by the system
- **clients** (*array*) – Array of `Client` ID
- **repository** (*integer*) – `Repository` ID
- **schedule** (*integer*) – `Schedule` ID
- **policy_type** (*string*) – Policy Backup Type
- **keep_hourly** (*integer*) – Number of hourly archives to keep
- **keep_daily** (*integer*) – Number of daily archives to keep
- **keep_weekly** (*integer*) – Number of weekly archives to keep
- **keep_monthly** (*integer*) – Number of monthly archives to keep
- **keep_yearly** (*integer*) – Number of yearly archives to keep
- **vmprovider** (*string*) – Name of the VM module provider
- **next_run** (*string*) – Date of the next run of the backup job
- **mode_pull** (*boolean*) – Backup in pull mode
- **enabled** (*boolean*) – Policy enabled
- **uuid** (*string*) – Auto generated UUID

Status Codes

- 200 OK – no error
- 404 Not Found – There is no `Policy` with this ID

POST /api/v1/policies/

Create a single policy.

```
{
    "name": "Full Features",
    "extra_vars": "",
    "clients": [
        1
    ],
    "repository": 1,
    "schedule": 1,
    "policy_type": "vm",
    "keep_hourly": 1,
    "keep_yearly": null,
    "keep_daily": null,
    "keep_weekly": null,
    "keep_monthly": null,
    "vmprovider": "proxmox",
    "mode_pull": false,
    "enabled": true
}
```

Response JSON Object

- **name** (*string*) – Policy name
- **extra_vars** (*string*) – JSON Dictionnary of variable used by the system
- **clients** (*array*) – Array of `Client` ID
- **repository** (*integer*) – `Repository` ID
- **schedule** (*integer*) – `Schedule` ID
- **policy_type** (*string*) – Policy Backup Type
- **keep_hourly** (*integer*) – Number of hourly archives to keep
- **keep_daily** (*integer*) – Number of daily archives to keep
- **keep_weekly** (*integer*) – Number of weekly archives to keep
- **keep_monthly** (*integer*) – Number of monthly archives to keep
- **keep_yearly** (*integer*) – Number of yearly archives to keep
- **vmprovider** (*string*) – Name of the VM module provider
- **mode_pull** (*boolean*) – Backup in pull mode
- **enabled** (*boolean*) – Policy enabled

Status Codes

- 200 OK – no error
- 404 Not Found – There is no `Policy` with this ID

PATCH /api/v1/policies/(int: *id*)/

Update a single repository.

```
{
    "name": "Main Repo",
    "path": "cyborgbackup@backup:/repository",
    "repository_key": "0123456789abcdef",
    "ready": true,
    "enabled": true
}
```

Response JSON Object

- **name** (*string*) – Policy name
- **extra_vars** (*string*) – JSON Dictionnary of variable used by the system
- **clients** (*array*) – Array of `Client` ID
- **repository** (*integer*) – `Repository` ID
- **schedule** (*integer*) – `Schedule` ID
- **policy_type** (*string*) – Policy Backup Type
- **keep_hourly** (*integer*) – Number of hourly archives to keep
- **keep_daily** (*integer*) – Number of daily archives to keep
- **keep_weekly** (*integer*) – Number of weekly archives to keep
- **keep_monthly** (*integer*) – Number of monthly archives to keep
- **keep_yearly** (*integer*) – Number of yearly archives to keep
- **vmprovider** (*string*) – Name of the VM module provider
- **mode_pull** (*boolean*) – Backup in pull mode
- **enabled** (*boolean*) – Policy enabled

Status Codes

- 200 OK – no error
- 404 Not Found – There is no `Policy` with this ID

DELETE /api/v1/policies/(int: *id*)/

Delete a single repository.

Status Codes

- 200 OK – no error
- 404 Not Found – There is no `Policy` with this ID

POST /api/v1/policies/(int: *id*)/launch/

Launch a backup job based on the policy.

Status Codes

- 200 OK – no error
- 404 Not Found – There is no `Policy` with this ID

GET /api/v1/policies/(int: *id*)/calendar/

Get all datetime of the current month for each run of the policy

```
[DATETIME]
```

Status Codes

- 200 OK – no error
- 404 Not Found – There is no `Policy` with this ID

4.1.7 Catalogs API V1

GET /api/v1/catalogs/
Retrieve a list of all catalogs entries.

```
{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [CATALOGS]
}
```

Response JSON Object

- **next** (*string*) – URI for next set of Catalogs.
- **previous** (*string*) – URI for previous set of Catalogs.
- **count** (*integer*) – Total number of Catalogs.
- **results** (*array*) – Array of `Catalog` objects.

GET /api/v1/catalogs/(int: *id*)/
Retrieve details of a single catalog entry.

```
{
    "id": 1,
    "url": "/api/v1/catalogs/1/",
    "archive_name": "vm-lab.example.com-2018-11-23_22-02",
    "path": "stdin",
    "job": 1,
    "mode": "-rw-rw----",
    "mtime": "2018-11-23T23:03:52Z",
    "owner": "root",
    "group": "root",
    "size": 12,
    "healthy": true
}
```

Response JSON Object

- **id** (*integer*) – The ID of the catalog entry
- **url** (*string*) – The URL access of the repository object.
- **archive_name** (*string*) – The Borg Backup archive name
- **path** (*string*) – Full path of the file in the archive
- **job** (*integer*) – `Job` ID catalog entry related

- **mode** (*string*) – Unix mode of the file
- **mtime** (*string*) – Latest modification date of the file
- **owner** (*string*) – Owner of the file
- **group** (*string*) – Group of the file
- **size** (*integer*) – Size of the file in Bytes
- **healthy** (*boolean*) – Healthy state of the file

Status Codes

- 200 OK – no error
- 404 Not Found – There is no `Repository` with this ID

CHAPTER 5

The Community Guide

This part of the documentation details the CyBorgBackup community.

5.1 Support

If you have questions or issues about CyBorgBackup:

5.1.1 Send a Tweet

If your question is less than 280 characters, feel free to send a tweet to @Gaetan_F,

5.1.2 File an Issue

If you notice some unexpected behaviour in CyBorgBackup, file an issue on GitHub.

5.1.3 Mailing Lists

Current Mailing Lists :

- Developers : developers@lists.cyborgbackup.dev
- Users : users@lists.cyborgbackup.dev

CHAPTER 6

The Contributor Guide

If you want to contribute to the project, this part of the documentation is for you.

6.1 How to Help

CyBorgBackup is under active development, and contributions are more than welcome!

1. Check for open issues or open a fresh issue to start a discussion around a bug.
2. Fork the repository on GitHub and start making your changes to a new branch.
3. Send a pull request and bug the maintainer until it gets merged and published. :)

6.1.1 Runtime Environments

CyBorgBackup currently supports the following versions of Python:

- Python 3.5
- Python 3.6
- Python 3.7

HTTP Routing Table

/api