CTRL-Z Documentation

Release 1.0

Sergei Maertens

Jan 20, 2023

Contents:

CTRL-Z (control-zee) is a backup and recovery tool for Django projects.

Its goals are to be operating system agnostic in creating and restoring backups, while being flexible through a yaml configuration file.

CHAPTER 1

Quickstart

1.1 Installation

1.1.1 Requirements

- Python 3.6 or higher
- setuptools 30.3.0 or higher
- PostgreSQL
- Database user must have permissions to drop/create the target database(s) for DB restore

1.1.2 Install

```
pip install ctrl-z

For development:
pip install -e .[tests,pep8,docs,release]
```

1.2 Usage

CTRL-Z exposes a CLI object to hook into your project, for example:

Listing 1: backup/cli.py

```
#!/usr/bin/env python
import os
import sys

```

(continues on next page)

(continued from previous page)

```python
from ctrl_z import cli

def setup():
    # Here you should ensure that your Django project is properly added to
    # ``sys.path``, and any other setup is done (loading ``.env`` for
    # example) so that settings can be imported in ctrl-z and django
    # initialized.
    pass


# assign the setup function to call
cli.setup = setup

if __name__ == '__main__':
    # specify which config file to use
    cli(config_file='/path/to/backup/config.yml')
```

Once the setup around the CLI is done, you can use it.

1.2.1 CLI help

At any time, you can get the CLI help:

```
python backup/cli.py --help

CTRL-Z 0.1.2 - Backup and recovery tool
usage: cli.py [-h] [--config-file CONFIG_FILE] [--base-dir BASE_DIR]
              {generate_config,backup,restore} ...

CTRL-Z CLI

positional arguments:
  {generate_config,backup,restore}
                        Sub commands
    generate_config     Generate a config file from the default config
    backup              Create a backup
    restore             Restore a backup

optional arguments:
  -h, --help            show this help message and exit
  --config-file CONFIG_FILE
                        Config file to use
  --base-dir BASE_DIR   Base directory override
```

1.2.2 Generate a config file

CTRL-Z ships with a default config file that you can use as a starting point.

```
python backup/cli.py generate_config
```

Command options:

- `-o`, `--output-file`: (relative or absolute) path to write the config to. Defaults to stdout.

See *Configuration* for detailed config options documentation.

1.2.3 Generate a backup

```
python backup/cli.py backup
```

By default, database AND file directories (such as `settings.MEDIA_ROOT`) are backed up.

Command options:

- `--no-db`, `--no-database`: do not dump the databases
- `--skip-db`: aliases (the key in `settings.DATABASES`) to skip dumping for. Useful if you have a multi-db setup and only the `default` is important, for example. Use multiple times for each alias to skip.
- `--no-files`: do not backup the (uploaded) files (e.g. `settings.MEDIA_ROOT`)

1.2.4 Restore a backup

```
python backup/cli.py restore /var/backups/2018-06-27-daily/
```

Restore the backup at the specified path.

Command options:

- `--no-db`, `--no-database`: do not restore the databases
- `--skip-db`: aliases (the key in `settings.DATABASES`) to skip restoring for. Useful if you have a multi-db setup and only the `default` is important, for example. Use multiple times for each alias to skip.
- `--no-files`: do not restore the (uploaded) files (e.g. `settings.MEDIA_ROOT`)
- `--db-name`: convenient for loading a different source database name into the target environment. Syntax: `alias:name`, for example `default:project_staging`. Dump files are saved with the database name in the file name, so this allows you to refer to that. Can be used multiple times for multi-db setups.
- `--db-host`: convenient for loading a different source database host into the target environment. Syntax: `alias:host`, for example `default:localhost`. Dump files are saved with the database host in the file name, so this allows you to refer to that. Can be used multiple times for multi-db setups.
- `--db-port`: convenient for loading a different source database port into the target environment. Syntax: `alias:port`, for example `default:5432`. Dump files are saved with the database port in the file name, so this allows you to refer to that. Can be used multiple times for multi-db setups.

CHAPTER 2

Configuration

Configuration is done in YAML format. You can pass the config file to use to the `cli` object (see *Usage*), or pass it as a global CLI option `--config-file`.

2.1 Global config options

2.1.1 `base_dir`

Type: string indicating filesystem path.

The base directory where backups are saved to. This should be an on-disk location, defaults to `/var/backups/`.

Within this directory, date-stamped backup directories will be created.

Old backups in this directory are rotated according to the *retention_policy*.

Note: As end user, you are responsible of setting up a mechanism to transfer backups to off-site storage.

2.1.2 `logging`

Type: object.

CTRL-Z uses stdlib logging to log all its actions. If e-mail notifications are set up, the contents of the log are mailed to indicated receivers.

`logging.filename` name of the log file, will be created inside the date-stamped backup directory.

`logging.level` Log level to control log verbosity. Defaults to INFO. Uses the available stdlib log levels.

2.1.3 `retention_policy`

Type: object

CTRL-Z rotates your backups for you to prevent you from running out of disk space on (production) machines.

`retention_policy.day_of_week` Integer, 0-6, indicating which day counts as weekly backup. Defaults to 0, which is Monday.

`retention_policy.days_to_keep` Number of daily backups to keep, including the backup-to-be-created. Defaults to 7.

`retention_policy.weeks_to_keep` Same as `days_to_keep`, except in weeks.

2.1.4 `report`

Type: object

CTRL-Z can use Django's e-mail machinery to send an e-mail report. Useful to have confirmation that the backup did indeed run/complete without issues.

`report.enabled` Boolean, whether to send reports or not. Defaults to True.

`report.to` List of e-mail address to send the report to. Defaults to `root@localhost`

2.1.5 `database`

Which databases need to be dumped/restored are introspected from `settings.DATABASES`. Database configuration here is related to CTRL-Z internals.

`database.test_function` String, python path. After restoring, CTRL-Z tests if the DB restore was not a failure. By default, the check tests if the `django_migrations` table is not empty. This is not water-tight, and you can provide your own function as long as it can be imported.

The function signature is:

```python
def my_restore_test(using: str='default') -> bool:
    """
    :param using: the alias of the database to check.
    """
    pass
```

2.1.6 `files`

Control how CTRL-Z runs backups of your (uploaded) files.

`files.overwrite_existing_directory` Boolean, defaults to True. If the folder already exists in the backup location, replace it. Useful when running the backup multiple times a day.

`files.directories` List of setting names to include in the backup. Defaults to `['MEDIA_ROOT']`, which means that only `settings.MEDIA_ROOT` will be included.

2.1.7 `pg_dump_binary`

Which binary to use to dump the database. Defaults to `/usr/bin/pg_dump`.

2.1.8 `pg_restore_binary`

Which binary to use to dump the database. Defaults to `/usr/bin/pg_restore`.

2.1.9 `createdb_binary`

Which binary to use to create databases. Defaults to `/usr/bin/createdb`.

2.1.10 `dropdb_binary`

Which binary to use to drop databases. Defaults to `/usr/bin/dropdb`.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search