SCM Backup Documentation

Release 1.0.0

Christian Specht

Nov 16, 2018

Contents:

SCM Backup is a tool which makes offline backups of your cloud hosted source code repositories, by cloning them.

It will support backing up from multiple source code hosters *(starting with GitHub)* and backing up multiple users/teams per source code hoster.

- Source code
- Website
- Download latest release

It's written in .NET Core, which means it's supposed to run on Windows, Linux and MacOS.

Note: SCM Backup 1.0 was just released, but the initial documentation is not yet finished. Please check again in a few days!

CHAPTER 1

Introduction

SCM Backup is a tool which makes offline backups of your cloud hosted source code repositories, by cloning them.

1.1 How does it work?

SCM Backup uses the respective hoster's API to get a list of all your repositories hosted there.

Then, it uses the respective SCM (e.g. Git and/or Mercurial, which need to be installed on your machine if you have at least one repository of the given type) to clone every repository into your local backup folder - or just pull the newest changes, if it already **is** in your local backup folder.

CHAPTER 2

Installation

2.1 System Requirements

2.1.1 .NET Core 2.0

SCM Backup is written in .NET Core, the cross-platform version of .NET.

The available releases are framework-dependent deployments, which means that the same download should work on any Windows, Linux and MacOS machine, as long as .NET Core is installed on it.

If it's not on your machine, you can get it from the official download page. You need at least **version 2.0** of the .NET Core **runtime**.

Note: So far, SCM Backup has been written and tested on Windows only. Technically, it should run on Linux and MacOS as well, but this has not been tested yet.

2.1.2 Source control software

SCM Backup doesn't come with its own versions of Git and/or Mercurial, so the respective SCM needs to be installed on your machine if you have at least one repository of the given type.

By default, SCM Backup expects all source control software to be in your path, so it just needs to execute `git`, `hg` etc. without a complete path, although it's possible to *specify the path to the executable in the config*.

Note that at runtime, SCM Backup checks the presence of all required SCMs on your system. It will stop if you have repositories needing a SCM which is not present on your system.

2.2 Download

At the moment, there are only .zip downloads.

Download the .zip file from the latest release and unzip it into a folder of your choice.

2.3 How to run

> **Warning:** You should edit the configuration file before running SCM Backup for the first time!
>
> *Read the guide* for more information.

The actual application is in the `ScmBackup.dll` library. You can execute it with the `dotnet` command:

```
dotnet ScmBackup.dll
```

For Windows, there's a batch file named `ScmBackup.bat` which does exactly that.

CHAPTER 3

Configuration

SCM Backup is configured in YAML, by editing the config file `settings.yml`.

3.1 General Options

3.1.1 localFolder

The folder (on the machine where SCM Backup runs) where all the backups will be stored.

The folder must already exist, SCM Backup won't create it.

Example:

```
localFolder: "c:\\scm-backup"
```

3.1.2 waitSecondsOnError

When an error occurs, SCM Backup will wait that many seconds before exiting the application.

Example:

```
waitSecondsOnError: 5
```

3.1.3 scms

SCM Backup uses the source control software already installed on your system. By default, it assumes that the required SCMs are installed in your path.

If this isn't the case, or if you have multiple versions of the same SCM on your system and want SCM Backup to use a specific one, you can specify the complete path to the executable in the config file.

Example:

```
scms:
  - name: git
    path: "c:\\git\\git.exe"
```

3.2 Sources

SCM Backup will be able to backup from multiple source code hosters, and multiple accounts per hoster.

For example, your GitHub user may be a member of an organization, and you may want to backup all repositories of your user, **and** all repositories of that organization.

In SCM Backup terms, these would be two different **sources**: your GitHub user would be one source, and the organization would be a second one.

You can define as many sources as you want in the config file, in this format:

```
sources:

  - title: some_title
    hoster: github
    type: user
    name: your_user_name

  - title: another_title
    hoster: github
    type: org
    name: your_org_name
```

Each source must have at least those four properties:

`title`

> Must be unique in the whole config file.
>
> For each source, SCM Backup will create a sub-folder named like the source's title in the *main backup folder*.

`hoster`

> The source code hoster from which you want to backup. See the sub-pages for valid values for each hoster.

`type`

> Either `user` or `org`, depending if you want to backup an user or a organization.

`name`

> The name of the user/organization you want to backup.

There are more possible options (for authentication, for example), but these can vary depending on the source code hoster.

See the respective sub-page for detailed documentation per hoster:

3.2.1 GitHub

Configuration settings for backing up repositories from GitHub.

Sources

For the basics, please read the *Sources* section first.

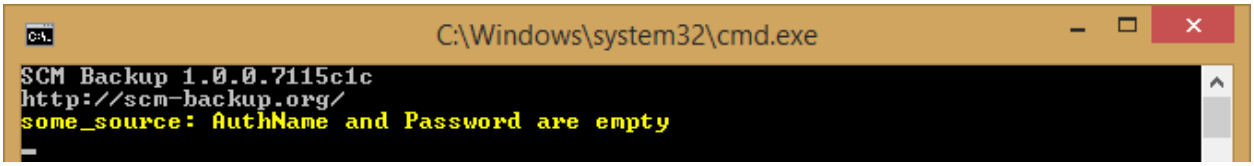
For GitHub, the `hoster` entry in the config file needs to look like this:

```
hoster: github
```

Authentication

Without authentication, SCM Backup can only backup your public repositories.

In this case, it shows a warning:



To backup your private repositories as well, you need to authenticate:

- To backup a user's repositories, you need to authenticate with that user.
- To backup an organization's repositories, you need to authenticate with a user who has sufficient permissions to that organization's repositories.

Create a personal access token for SCM Backup for that user:

1. In the user's settings on GitHub, create a new token. Give it at least the `repo:status` scope.

   This scope allows SCM Backup to get a list of that user's repositories via the GitHub API (read more about scopes).

2. Put the username and the token into the `authName` and `password` properties of the source in the config file.

   Example:

   ```
   sources:

     - title: some_title
       hoster: github
       type: org
       name: your_org_name
       authName: your_user_name
       password: your_token
   ```

   This will backup the repositories of the organization `your_org_name`, but authenticate with the user `your_user_name`.

CHAPTER 4

Restoring your backups

Generally, SCM Backup creates local repositories and pulls from the remote repositories into the local ones.

Those local repositories are **bare repositories**, i.e. they don't contain a working directory.

When you look inside the repository directories, you'll see some directories and files, depending on whether it's a Git/Mercurial/etc. repository.

Your complete history and your source code are in there - you just don't see the actual files!

The repository is backed up without a working directory, because it's not necessary.

All the data already exists inside the repository, a second copy of everything in the working directory would just be a waste of space.

The easiest way to restore your working directory is to clone the bare repository that SCM Backup created *(called* `bare-repo` *in the examples)*, which will create a clone **with** a working directory *(called* `working-repo` *in the examples)*.

For more details, please see the sub-page for the respective SCM:

4.1 Restoring Git repositories

4.1.1 How a bare repository looks like

It contains a few folders (`objects`, `refs...`) and some files.

4.1.2 How to restore

Clone the bare repository into a "regular" one:

```
git clone bare-repo working-repo
```

`working-repo` will have a working directory.

CHAPTER 5

How to Contribute

5.1 Contribute to the application

5.1.1 How to run the integration tests

For each supported hoster, SCM Backup needs to:

- make API calls to get a list of repositories
- use Git/Mercurial etc. to clone repositories

So there are integration tests for each hoster which do these things as well, some of them with authentication.

We created test users especially for these tests (for example: user and organization used for GitHub integration tests), but of course we can't publish their passwords.

So in order to run any of these integration tests, you need to setup your *own* test users and test repositories.

SCM Backup's integration tests read the users, password etc. from a file named `environment-variables.ps1` in the main project directory, which is not in the repository.

You need to create your own by copying/renaming environment-variables.ps1.sample, and changing the values.

5.2 Contribute to the documentation

The documentation is built with Sphinx and hosted on Read the Docs.

The source code is here on GitHub.

5.2.1 Headlines

To make sure that the Read the docs Sphinx theme renders correctly, it's important that the headline styling is consistent across the whole documentation.

SCM Backup's documentation uses these stylings:

- `=====` for level 1 *(the top headline of each page)*
- `-----` for level 2
- `+++++` for level 3

5.3 Making a new release

To make a new release version, the following steps must be followed:

5.3.1 1. Determine the new version number

SCM Backup uses Semantic Versioning.

The new version number **must** be in "three-digit" `MAJOR.MINOR.PATCH` format, for example `1.0.0` !!

5.3.2 2. Release the application

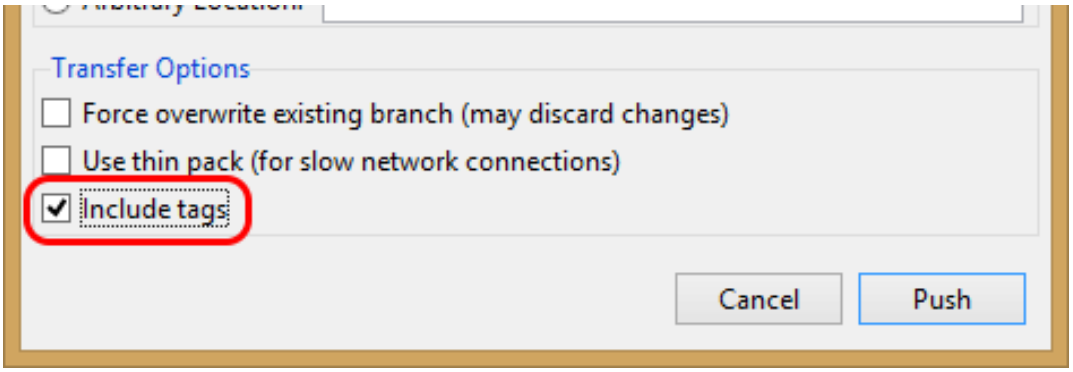
Each push to `master` creates a new CI build on AppVeyor anyway.

Create a new release by creating a Git tag in the main repository with the new version number.

The CI build will recognize this and automatically use this version number to create a new GitHub release.

Note: Don't forget to actually push the tag! Git doesn't do this automatically.

- From the command line, it's `git push origin 1.0.0`.
- In Git GUI, you need to set this checkbox when pushing:



5.3.3 3. Release the docs

- Set the version and release numbers in the Sphinx configuration file `conf.py` to the new version number.

  Set `version` to the short `X.Y` format, e.g. `1.0`.

  Set `release` to the full three-digit format determined in step 1, e.g. `1.0.0`.

  Apparently Read the Docs uses this number at least in the automatically created PDF.

- Create the same "version number" Git tag *(like in the main repository)* in the documentation repository as well.

  This will create a version of the documentation for this release, making use of Read the Docs' versioning capabilities.

CHAPTER 6

Legal Stuff

6.1 License

SCM Backup is licensed under the GPL.

6.2 Acknowledgements

SCM Backup uses the following OSS projects:

- Json.NET
- NLog
- RichardSzalay.MockHttp
- Simple Injector
- xUnit.net
- YamlDotNet

Special thanks to Steven for invaluable advice.