gsheet-keyring Documentation

Release 1.0.0

Oliver Steele

Jun 10, 2018

Contents:

This package provides a Keyring backend that stores passwords in a Google Sheet.

CHAPTER 1

Motivation

This package allows ipython-secrets to be used on Google Colaboratory, and on other hosted services that don't support the standard Keyring backends.

The **ipython-secrets** package uses Keyring to store secrets for use in a Jupyter notebook. However, none of the standard Keyring backends works in Google Colaboratory, since that service provides neither durable file storage, nor the native operating system services that the standard keyring backends require.

CHAPTER 2

Usage

To use this package, install it via `pip3 install gsheet-keyring`, and use the Keyring API as normal. If one of the built-in Keyring backends is available, Keyring will use that backend in preference to this one (as it should). However, if a platform-specific backend is not available, Keyring will automatically detect and use this package instead.

Use `keyring.set_keyring` to force Keyring to use this package even, if other backends are available:

```
import keyring
from gsheet_keyring import GoogleSheetKeyring
keyring.set_keyring(GoogleSheetKeyring())
```

By default, this backend searches for a Google Sheet named "keyring". If there's no sheet with this name, one is created.

You can override this default by specifying a Google Sheet name, a Google Sheet key, or a `Worksheet` from the gspread package.

CHAPTER 3

Alternatives

If you're running in an environment where any other Keyring backend is available, use that instead. (This should happen automatically.)

If you require either greater performance or security than this package provides (see the notes in the API documentation), you probably want to instead create or use a backend that uses a secret management service such AWS Secrets Manager, Google Cloud AMS, or or Hashicorp Vault.

The keyring-vault-backend package is a Keyring backend interface to Hashicorp Vault. I haven't used it.

CHAPTER 4

License

MIT

CHAPTER 5

API

This package provides a Keyring backend that is backed by a Google Sheet.

Example

`example.py` is an example of using this package outside of Google Colaboratory, Google Compute Engine, or another environment that sets Google OAuth2 credentials automatically.

5.1 Credentials

A `GoogleCredentials` instance is required in order to access the Google Sheet. This can be supplied in one of the following ways:

1. Follow the instructions in Getting Started with Authentication. Set the `GOOGLE_APPLICATION_CREDENTIALS` environment variable to the path to the service account key file.
2. If the code is running in Google Compute Engine or certain other Google environments, the code uses the instance's service account via `oauth2client.client.GoogleCredentials.get_application_default()`.
3. If the code is running in a Colaboratory notebook, the user will be asked to sign in via `google.colab.auth.authenticate_user()`.
4. Use `oauth2client` to instantiate a `GoogleCredentials`. Pass it as the *credential* parameter to *`GoogleSheetKeyring()`*. See the gspread wiki for additional information on getting started with Google Authentication.

If you use gsheet-keyring from Google Colaboratory, Google Compute Engine, or certain other Google hosting services, you don't need to obtain a credentials file. In these cases, you can instantiate *`GoogleSheetKeyring`* without a `credentials` argument.

5.2 Caching

Access to Google Sheets is very slow. This package performs minimal caching — just enough to optimize these cases:

- The caller sets and then gets a password.
- The caller gets a password multiple times.

In order to minimize the risk of using stale data when a notebook is left running in a background browser tab while you interact with another tab. The cache expires quickly.[1]

Caution: Passwords are stored *unencrypted* in your Google Sheet. Standard security warnings apply:

- Don't share your "keyring" Google Sheet more widely than you want your passwords shared.
- Anyone with access to your Google account has access to these passwords.
  - This includes anyone who can sign into a laptop or phone that is signed into your Google account.
- If you open the spreadsheet in a public place, you are vulnerable to shoulder surfing.
- If you open it within view of a camera, you have leaked your passwords to (today) anyone who can view the stream, or (going forwards) anyone who gains access to a server that stores the stream. (Hello, Nest!)
- Even if you open the spreadsheet in a private place, you're only as secure as the physical security of the lines of sight (including through windows) to your screen.

Caution: This package's use of Google Sheets is neither Atomic, nor Consistent, nor Isolated.

Simultaneous calls to a single writer (a single call to either `set_password()` or `delete_password()`) and/or multiple readers (any number of calls to `get_password()`) should be fine.

Simultaneous calls (for example, from different Jupyter notebooks) to `set_password()` and/or `delete_password()` can easily corrupt the spreadsheet. If you need this capability, this package (and Google Sheets) is not the right technology to build it on top of. In this case, consider using a hosted database as a back end, or using a hosted key management service instead of Keyring.

class gsheet_keyring.**GoogleSheetKeyring**(*, *sheet_key=None*, *sheet_title='keyring'*, *sheet_url=None*, *credentials=None*, *worksheet=None*)

A Keyring back end backed by a Google Sheet.

The Google Sheet may be specified with a variety of parameters. They have the precedence *worksheet* > *sheet_url* > *sheet_key* > *sheet_title*. The first truthy parameter is used. Lower-precedence parameters are silently ignored. For example, if *sheet_url* is truthy, *sheet_key* and *sheet_title* are ignored. If the only truthy parameter is *sheet_title* and no sheet with this title is found, a new sheet is created. This is in the only circumstance in which this class will create a new sheet.

Parameters

- **credentials** (`oauth2client.client.GoogleCredentials`, optional) – An instance of `oauth2client.client.GoogleCredentials`.
- **sheet_key** (*str, optional*) – A Google Sheet document key.

[1] Data is currently cached for a minute, counting from the last access (to any password, not just the requested password). This is slow by Python execution speed, but fast by human standards. This matches the intended use of the package.

- **sheet_title** (*str, optional*) – A Google Sheet document title. Defaults to "keyring".
- **sheet_url** (*str, optional*) – A Google Sheet document URL.
- **worksheet** (gspread.models.Worksheet, optional) – A gspread.models.Worksheet instance.

credentials
An instance of oauth2client.client.GoogleCredentials.

This has the value of the credentials initialization parameter.

If this parameter isn't specified, the credentials are computed as described in the module documentation.

delete_password(*servicename*, *username*)
Delete the password for the username of the service.

get_password(*servicename*, *username*)
Get password of the username for the service

set_password(*servicename*, *username*, *password*)
Set password for the username of the service

sheet
The gspread.models.Worksheet that is used as a backing store.

Python Module Index

g

Index