Genenate your Github repository structure in seconds with cookiecutter

TLDR:

This story is about using a python **package** that allows any Github practitioner to create repos templates, generate folders structure and set-up few packages to ensure respecting coding best practices.

Introduction:

When starting a new Github repository, we often start by thinking about the folders structure and where each script or file should go in the project tree. Also, as first steps we usually setup the CI/CD to accelerate the development and testing tasks, and initiate the Readme…

Well if you faced this situation at least once in your life you may have thought about ways to **simplify the process** and move faster to the coding part. This is where **Cookiecutter** comes into action.

Cookiecutter is a command-line utility that allows to generate the **structure of the repository from a template**. It works in multiple languages and is widely used among software engineers, data scientists and researchers.

It only requires to:

- Create once **your own template** or **search for an existing** one on Github
- **Configure** the **cookiecutter.json** file on the root folder
- **Run the command-line** and follow the instruction

1. Create a template

Let's start by an example: in this article I wrote about a project structure that I often use in my data science projects. (The repo could be found here). The **main branch** contains the **final result that we will achieve when following the below steps** (ie: the generated project). The **branch cookiecutter** is the template that allows to generate it.

In few words, the template used as example allows to organize folders and files, quickly start a new project with few commands in the makefile to **automatically format code**, **analyze its quality**, **run tests** and finally **generate the README**. In order to transform this structure to a general cookiecutter template we use jinja2 syntax.

The goal behind this article is to **demonstrate the use of Cookiecutter** to build our own template and generate a project based on it. If you are interested in the details of what is **inside the generated example** you can refer to the article above.

First step is to tell cookiecutter the "**variable**" elements in the template that we would like to **overwrite** each time a new project is created. These elements could be the project name, the author, the file paths, the description …

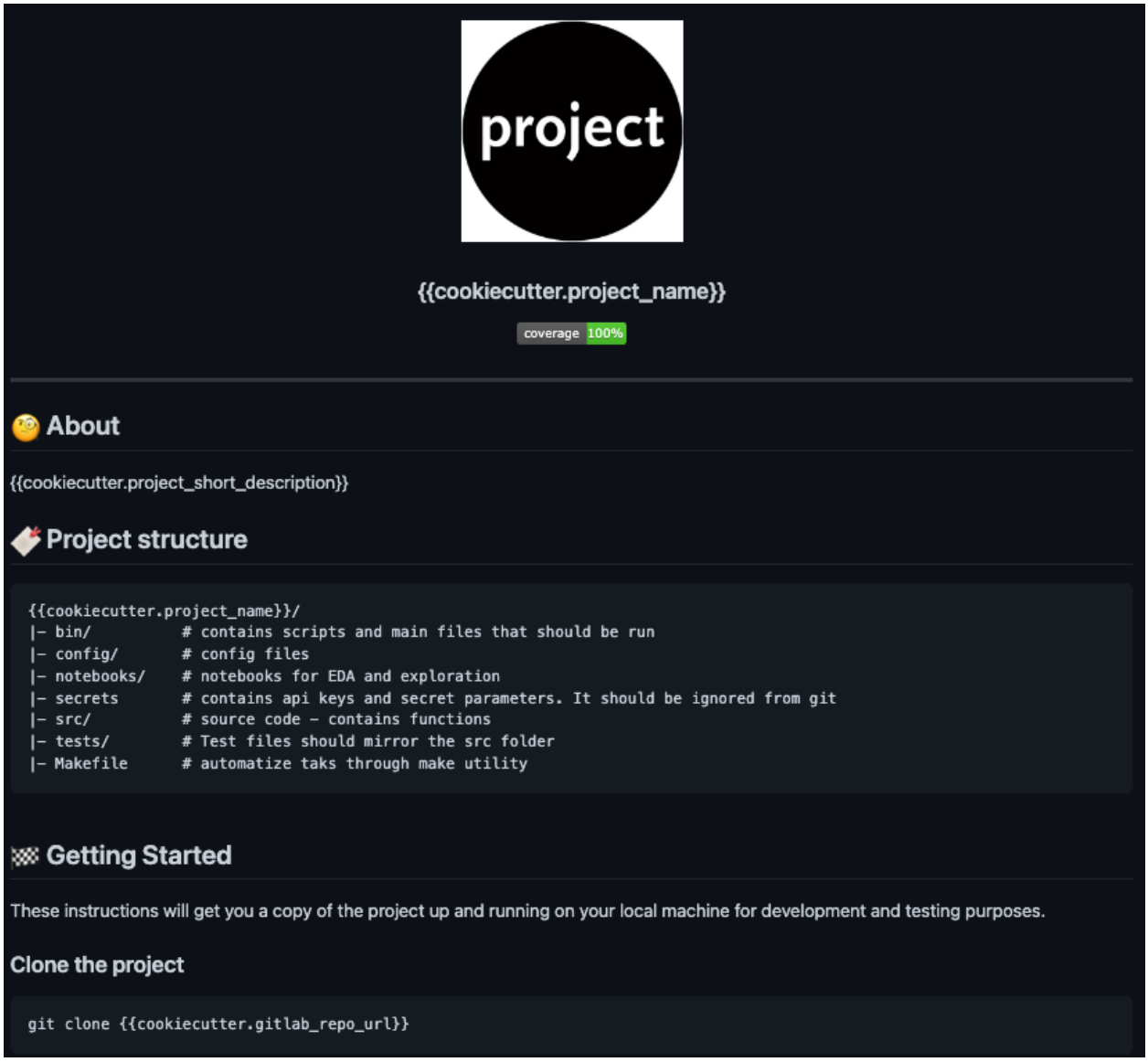
The jinja2 syntax to define a variable is the following:

```
{{cookiecutter.project_name}}{{cookiecutter.python_major_version}}
```

We can also use **if structure** and **loops** inside the script files. An example we could use this statement:

```
{% if cookiecutter.python_major_version != "3" %}
 echo "Upgrade your python version!"
{% endif %}
```

Knowing these elements we can proceed with the modifications and the creation of the template. Cookiecutter expects to have all project files and directories inside one folder named **{{cookiecutter.project_name}}**. Few modifications are also needed in the project readme, ci file,… You can check the final result here and see the differences by **comparing the two branches**.



The cookiecutter template of a git project

2. Configure cookiecutter.json file

Second step is to adapt your json config file. This is where cookiecutter looks for the **variables used in the template.** This file contains a mapping between **variable names** and **their true values**.

```
{
    "full_name": "Your Name",
    "email": "you@yourdomain.com",
    "project_name": "project_name",
    "gitlab_repo_url": "https or ssh url to clone the project",
    "project_short_description": "Sample project .....",
    "python_major_version": 3,
    "python_minor_version": 10
}
```

3. Generate a new project

Last step consists in generating the repo structure. Actually, this is simple **given the template and the config file**. All we have to do is to **install** the cookiecutter package from pypi and **run the utility** with the template repo link on github or locally. You can also specify the branch as an argument. Here is an example:

```
pip install cookiecuttercookiecutter
git@github.com:kaislar/ds_template.git --checkout cookiecutter
```

Cookiecutter will then guide you through the configuration and builds the structure for you:

```
(template)                                  % cookiecutter git@github.com:kaislar/ds_template.git --checkout cookiecutter
You've downloaded /Users/kais.laribi/.cookiecutters/ds_template before. Is it okay to delete and re-download it? [yes]:
full_name [Your Name]: test
email [you@yourdomain.com]: test@test.test
project_name [project_name]: hello_world_ds
gitlab_repo_url [https or ssh url to clone the project]: git@github.com:kaislar/hello_world_ds.git
project_short_description [Sample project .....]: This project is generated by cookiecutter
python_major_version [3]:
python_minor_version [10]:
```

The cookiecutter utility in action

Conclusion:

In this article, we saw how we can create our own template to **quickly generate the initial structure** of a Github project. We based this

article on an example that I often use in my own **data science projects** described in more depth here.