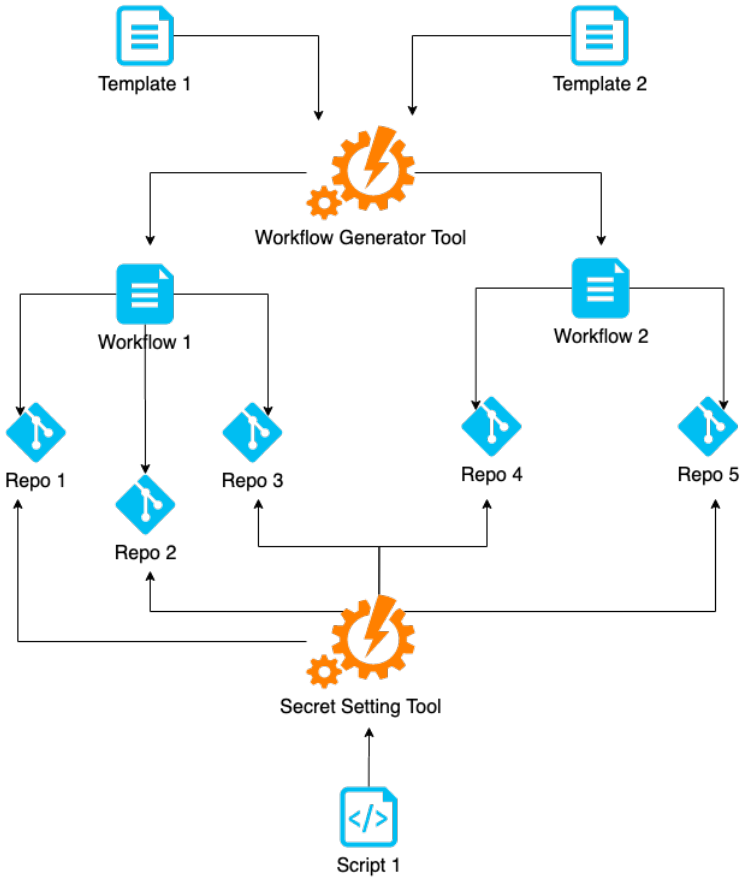
workflow-tools

Vitaly Samigullin

Apr 01, 2020

USER DOCUMENTATION



CLI tools for GitHub Actions.

- Automate writing GitHub workflow configs with a generator tool
- Automate setting GitHub secrets for repositories
- Integrate the tools into your pipeline for setting up new microservices

CHAPTER

ONE

RATIONALE

Microservice architecture may have dozens and dozens of lookalikes services that require similar CI/CD workflows. With infrastructure as code approach taken by the GitHub Actions, why not using workflows generation? Provisioning repository for a new microservice may also be automated. This is where `workflow-tools` come in handy.

CHAPTER

TWO

EXAMPLES

Let's set GitHub Secrets for a repository. First, get a personal access token in GitHub settings. Then set up a secret using `workflow-tools`:

```
workflow_secret --owner=anna-money --repo=workflow-tools \
  --token="YOUR-PERSONAL-ACCESS-TOKEN" \
  update --key=MY_SECRET_KEY --value=MY_VALUE
```

Now let's use a fragment of Jinja2 template for a GitHub Actions workflow to generate resulting config:

```
WORKFLOW_RUNNER_VERSION=ubuntu-18.04 WORKFLOW_PYTHON27=2.7 WORKFLOW_PYTHON37=3.7 \
workflow_generator
# Press Enter to start pasting Jinja2 workflow template into stdin
jobs:
  test:
    runs-on: [[ workflow.runner_version ]]
    strategy:
      matrix:
        python:
          - [[ workflow.python27 ]]
          - [[ workflow.python37 ]]
# Press Ctrl+D to render resulting workflow
# For real workflow templates use reading/writing from/to a file, load variables from␣
↪envfile
jobs:
  test:
    runs-on: ubuntu-18.04
    strategy:
      matrix:
        python:
          - 2.7
          - 3.7
```

See *Examples* for a detailed tour on using `workflow-tools` in the real world.

2.1 Installation

2.1.1 pip

Just use:

```
pip install -U workflow-tools
```

2.1.2 GitHub

You can also install the package from the source code:

```
git clone https://github.com/anna-money/workflow-tools
cd workflow-tools
make install
```

2.2 Tools

`workflow-tools` sticks to the Unix-way's rule *Do One Thing and Do It Well*. That's why once installed the package automatically generate executables for each CLI tool.

2.2.1 workflow_secret

Create, update or list GitHub secrets for the repository

EXAMPLES

Let's work with the repository https://github.com/anna-money/workflow-tools

1. Create a secret HELLO=WORLD

   workflow_secret –owner=anna-money –repo=workflow-tools –token="account:token" –debug update –key=HELLO –value=WORLD

2. Get a list of all secrets

   workflow_secret –owner=anna-money –repo=workflow-tools –token="account:token" list

3. Get info about a secret

   workflow_secret –owner=anna-money –repo=workflow-tools –token="account:token" get –key HELLO

4. Delete a secret HELLO

   workflow_secret –owner=anna-money –repo=workflow-tools –token="account:token" delete –key HELLO

```
workflow_secret [OPTIONS] COMMAND [ARGS]...
```

Options

--owner <owner>
Repository owner

--repo <repo>
Repository name

--token <token>
GitHub access token

--debug, --no-debug
Log debug information

delete

Delete secret

```
workflow_secret delete [OPTIONS]
```

Options

--key <key>
GitHub Secret Name

get

Check details of secret

```
workflow_secret get [OPTIONS]
```

Options

--key <key>
GitHub Secret Name

list

List all secrets for the repository

```
workflow_secret list [OPTIONS]
```

update

Create or update secret in the repository

```
workflow_secret update [OPTIONS]
```

Options

--key <key>
GitHub Secret Name

--value <value>
GitHub Secret Value

2.2.2 workflow_generator

GitHub Workflow Generator based on Jinja2 templates.

Interpolate Jinja2 template from INPUT with environment variables and write to OUTPUT.

Template variables to be interpolated in the template should be denoted as follows:

[[workflow.your_variable]]

INPUT and OUTPUT can be files or standard input and output respectively. With no INPUT, or when INPUT is -, read standard input. With no OUTPUT, or when OUTPUT is -, write to standard output.

EXAMPLES

1. Common patterns working with input/output

   workflow_generator input.tmpl output.yaml

   workflow_generator - output.yaml

   workflow_generator input.tmpl -

   workflow_generator input.tmpl

   tail -n 12 input.tmpl | workflow_generator > output.yaml

2. Generate Pull Request GitHub workflow with the values taken from the envfile

   workflow_generator YOUR-TEMPLATES-PATH/pr.tmpl YOUR-WORKFLOWS-PATH/pr.yml -e YOUR-TEMPLATES-PATH/.env.example

3. Override values from envfile by the environment variable

   WORKFLOW_PROJECT=test workflow_generator YOUR-TEMPLATES-PATH/pr.tmpl -e YOUR-TEMPLATES-PATH/.env.example

```
workflow_generator [OPTIONS] [INPUT] [OUTPUT]
```

Options

-p, --prefix <prefix>
Set prefix for envs to be used in template interpolation

-e, --envfile <envfile>
Load env from file. OS envs overwrite file values

--secrets, --no-secrets
Show only secrets needed for the workflow

--vars, --no-vars
Show only user defined variables needed for the workflow

--strict, --no-strict
Throw exceptions for undefined variables

Arguments

INPUT
Optional argument

OUTPUT
Optional argument

2.3 Examples

Let's consider a real life example: setting up a GitHub Actions workflow for the workflow-tools repository itself. We need:

1. Generate a GitHub Action workflow using `workflow_generator` tool
2. Set GitHub Secrets the workflow needs using `workflow_secret` tool

2.3.1 Generating workflow

```
1 WORKFLOW_RUNNER_VERSION=ubuntu-latest \
2 workflow_generator \
3 docs/examples/master.tmpl \
4 ~/PATH-TO-YOUR-REPO/.github/workflows/master.yml \
5 -e docs/examples/envfile
```

First, we define a Jinja2-template for the workflow (see a `file` at line 3). Variables to be substituted should be marked up this way:

```
[[ workflow.your_variable ]]
```

When rendering the resulting file, `workflow-generator` tool substitutes the markup with the value of corresponding environment variable:

```
WORKFLOW_YOUR_VARIABLE
```

The environment variable can be set globally, for a single command run, or can be read from the envfile specified by the option flag `-e` (see a `file` at line 5).

Envfile comes in handy when a template uses many variables at once. It's also easier to share variables between the templates using envfile. The envfile has the *lowest precedence*, it can be overridden (line 1).

2.3.2 Setting secrets

Now that we generated the workflow, let's set a GitHub secret used by the template:

```
1 workflow_secret \
2   --owner=anna-money \
3   --repo=workflow-tools \
4   --token="YOUR-PERSONAL-ACCESS-TOKEN" \
5  update \
6    --key=PYPI_PUSH_USER \
7    --value=YOUR_SECRET_VALUE
```

First, we need to get a personal access token (see line 4). `workflow_secret` tool has multiple commands (see *Tools*). To set up new or update existing secret `update` command is used (line 5). The command accepts `--key` and `--value` options (lines 6, 7). `workflow_secret` also have tool-wide options used for each command: `--owner` (GitHub user, line 2), `--repo` (GitHub repository name, line 3) and `--token`.

Finally, let's check what secrets are set for the repository:

```
1 workflow_secret \
2   --owner=anna-money \
3   --repo=workflow-tools \
4   --token="YOUR-PERSONAL-ACCESS-TOKEN" \
5  list
```

2.4 Contributing

1. Use Python 3.7+, `make` and `virtualenv`
2. Install dependencies:

```
make install
```

3. Install pre-commit hooks:

```
make hooks
```

4. Start contributing!
5. Make sure GitHub Actions pipeline for your Pull Request is passing

2.5 Changelog

2.5.1 0.6.0 (2020-04-01)

- Update readthedocs config (#8) by @pilosus

2.5.2 0.5.0 (2020-04-01)

- Fix autodocs (#7) by @pilosus
- Add check docs step in CI/CD (#7) by @pilosus
- Add badges to README (#7) by @pilosus

2.5.3 0.4.0 (2020-04-01)

- Extend CI/CD flow with package check (#5) by @pilosus
- Fix bugs with setup.py long description by @pilosus

2.5.4 0.3.0 (2020-03-31)

- Add documentation and ReadTheDocs integration (#3) by @pilosus

2.5.5 0.2.0 (2020-03-30)

- Use GitHub Actions for the project (#2) by @pilosus

2.5.6 0.1.0 (2020-03-30)

- Move internal ANNA project to open source (#1) by @pilosus
- Originally developed by Vitaly Samigullin (@pilosus) as internal ANNA project

INDEX

Symbols

I

O

W