python-gitlab

How to automate your job

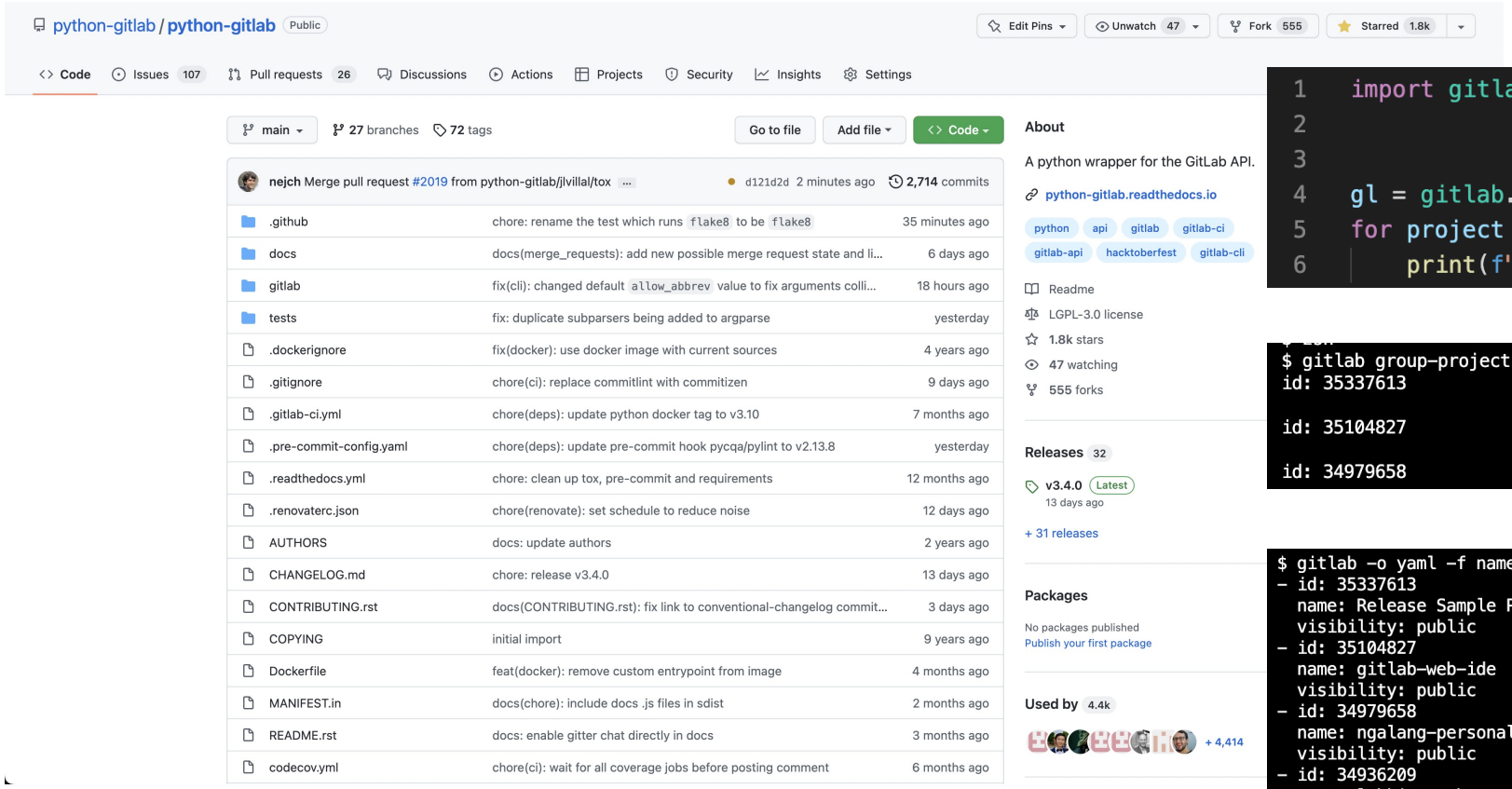
SIEMENS

Agenda

- What is it and why use it?
- Live Demo – Library
- Live Demo – CLI

SIEMENS

What is it?



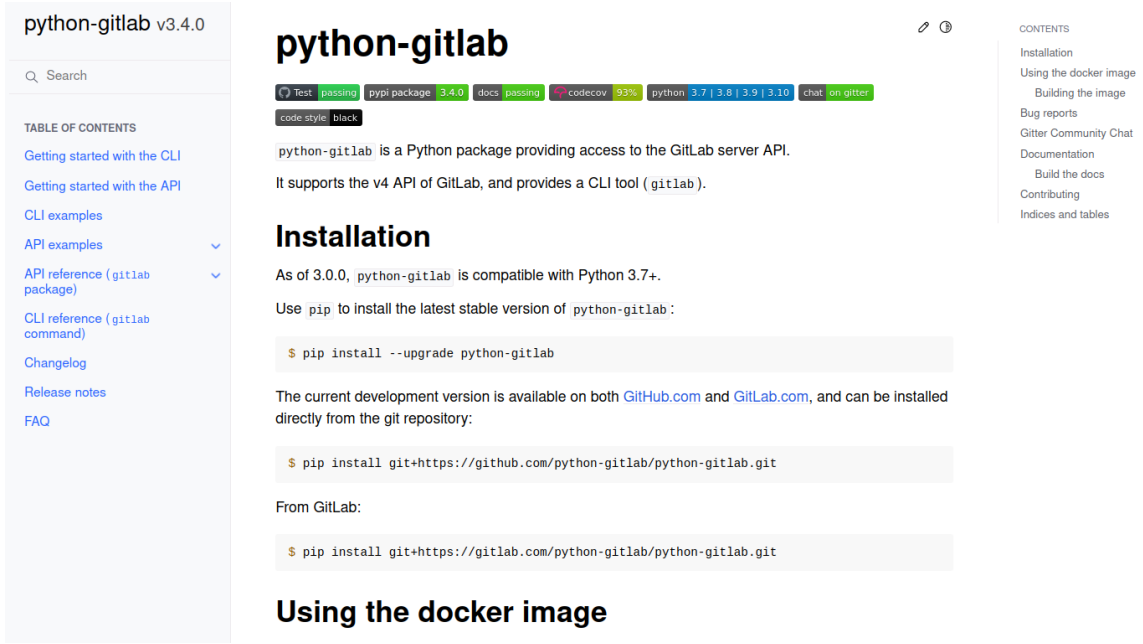

```python
import gitlab


gl = gitlab.Gitlab(per_page=100)
for project in gl.projects.list(as_list=False):
    print(f"{project.id} - {project.path_with_namespace}")
```

```
$ gitlab group-project list --group-id gitlab-org
id: 35337613

id: 35104827

id: 34979658
```

```
$ gitlab -o yaml -f name,id,visibility group-project list --group-id gitlab-org
- id: 35337613
  name: Release Sample Projects
  visibility: public
- id: 35104827
  name: gitlab-web-ide
  visibility: public
- id: 34979658
  name: ngalang-personal
  visibility: public
- id: 34936209
  name: labkit-python
  visibility: public
- id: 34770409
  name: GitLab Metrics Exporter
  visibility: public
```


SIEMENS

What is it?



- Python library and CLI tool
- ~1.4m monthly package installs
- Available from PyPI, source or as Docker image
- Powers IaC and automation tooling (Ansible modules, python-semantic-release, ..)

SIEMENS

Why?

"It's just a few curl calls, I don't need a library"

Why?

"It's just a few curl calls, I don't need a library"

Two curl calls later:

```
# prepare urlencode <url> function, allowing converting provided string
urlencode: |
  urlencode() {
    # urlencode <string>

    old_lc_collate=$LC_COLLATE;
    LC_COLLATE=C;

    local length="${#1}"
    for i in `seq 1 $length`
    do
      local c=$(expr substr "$1" $i 1)
      case $c in
        [a-zA-Z0-9.~_-]) printf '%s' "$c" ;;
        *) printf '%%%02X' "'$c" ;;
      esac
    done

    LC_COL
  }
```

`jq` doesn't fetch pages. You need to fetch pages with multiple `curl` calls, JSON outputs. For example you could get a JSON array of the combined `.`

```
url=https://mywebsite.com/api/cars.json
jq -s '.[0].items + .[1].items' <(curl "$url&page=1" | jq .) <
```

128

The following statement will retry 5 times or a maximum
5 seconds, and no exponential backoff policy

```
curl --connect-timeout 5 \
    --max-time 10 \
    --retry 5 \
    --retry-delay 0 \
    --retry-max-time 40 \
    'http://your_url'
```


SIEMENS

Why?

"It's just a few curl calls, I don't need a library"

Two curl calls later:

```
# prepare urlencode <url> function, allowing converting provided string
urlencode: |
  urlencode() {
    # urlencode <string>

    old_lc_collate=$LC_COLLATE;
    LC_COLLATE=C;

    local length="${#1}"
    for i in `seq 1 $length`
    do
      local c=$(expr substr "$1" $i 1)
      case $c in
        [a-zA-Z0-9.~_-]) printf '%s' "$c" ;;
        *) printf '%%%02X' "'$c" ;;
      esac
    done

    LC_COL
  }
```

`jq` doesn't fetch pages. You need to fetch pages with multiple `curl` calls, JSON outputs. For example you could get a JSON array of the combined `.`

```
url=https://mywebsite.com/api/cars.json
jq -s '.[0].items + .[1].items' <(curl "$url&page=1" | jq .) <
```

128

The following statement will retry 5 times or a maximum
5 seconds, and no exponential backoff policy

```
curl --connect-timeout 5 \
    --max-time 10 \
    --retry 5 \
    --retry-delay 0 \
    --retry-max-time 40 \
    'http://your_url'
```

python-gitlab:

```
projects = gl.projects.list(all=True)
```


SIEMENS

Why?

"It's just a few curl calls, I don't need a library"

Two curl calls later:

```
# prepare urlencode <url> function, allowing converting provided string
urlencode: |
  urlencode() {
    # urlencode <string>

    old_lc_collate=$LC_COLLATE;
    LC_COLLATE=C;

    local length="${#1}"
    for i in `seq 1 $length`
    do
      local c=$(expr substr "$1" $i 1)
      case $c in
        [a-zA-Z0-9.~_-]) printf '%s' "$c" ;;
        *) printf '%%%02X' "'$c" ;;
      esac
    done

    LC_COL
  }
```

`jq` doesn't fetch pages. You need to fetch pages with multiple `curl` calls, JSON outputs. For example you could get a JSON array of the combined `.`

```
url=https://mywebsite.com/api/cars.json
jq -s '.[0].items + .[1].items' <(curl "$url&page=1" | jq .) <
```

128

The following statement will retry 5 times or a maximum
5 seconds, and no exponential backoff policy

```
curl --connect-timeout 5 \
     --max-time 10 \
     --retry 5 \
     --retry-delay 0 \
     --retry-max-time 40 \
     'http://your_url'
```

python-gitlab does:

- Proper URL encoding
- Pagination
- Rate limits
- Retries on network & server errors
- Complex API attributes (arrays, delimited strings)
- Generic methods for arbitrary endpoints
- Passing arbitrary API parameters (in most cases)

SIEMENS

Live Demo - Library

Live Demo - CLI

python-gitlab without python

SIEMENS

Contacts

Nejc Habjan – nejc.habjan@siemens.com

Max Wittig – max.wittig@siemens.com

Issue tracker: https://github.com/python-gitlab/python-gitlab

Docs: https://python-gitlab.readthedocs.io/en/stable/

Chat: https://gitter.im/python-gitlab/Lobby

SIEMENS