gl_runner_api Documentation

gl_runner_api developers

Oct 22, 2020

Contents:

An unofficial Python implementation of the API for creating customised GitLab CI runners.

This package provides the basic functionality for registering a Runner. This object can then be used to request a Job and communicate the log, status and artifacts back to GitLab. No functionality is provided to execute the payloads defined in the `.gitlab-ci.yml`.

This package was originally developed to support LHCb's Analysis Productions by providing a CI runner that could submit jobs to LHCbDIRAC. More details can be found in TODO.

CHAPTER 1

Registering a Runner

The simplest way to register a new Runner is with the included command line tool:

For example

```
$ register-runner "https://gitlab.cern.ch/" "MY_REGISTRATION_TOKEN" "my-runner-data.
↪json " --locked
INFO:gitlab_runner_api:gitlab.cern.ch: Successfully registered runner 6602␣
↪(abcdefghij)
INFO:gitlab_runner_api:gitlab.cern.ch: Successfully initialised runner 6602
```

where arguments can be found by navigating to the “CI/CD” page of the desired repository’s settings.

CHAPTER 2

Getting jobs

After a runner has been registered it can be loaded from the `.json` file and used to request jobs:

```
from gitlab_runner_api import Runner
runner = Runner.load("my-runner-data.json")
runner.check_auth()
if job := runner.request_job():
    print("Received a new job, starting executor")
    my_job_executor(job)
else:
    print("No jobs are currently available")
```

CHAPTER 3

Executing jobs

A minimal CI executor might run as follows:

```
from gitlab_runner_api import failure_reasons

job.log += f"Starting job with id {job.id} for branch {job.ref}\n"
do_clone_and_checkout(job.repo_url, job.commit_sha)
success = run_tests(job)
if success:
    job.log += "All tests ran successfully\n"
    job.set_success()
else:
    # ANSI formatting codes can be used to enhance the CI logs
    job.log += "\u001b[31mJob failed!!!\u001b[0m\n"
    job.set_failed(failure_reasons.ScriptFailure())
```

See the reference Job documentation for the full list of available properties.

CHAPTER 4

Persisting jobs

For long running jobs it may be desirable to persist the job object between calls. This can be done using a similar interface to the `pickle` module in the Python standard library:

```
job_data = job.dumps()

from gitlab_runner_api import Job
job = Job.loads(job_data)
```

Note: The job log is included in the persisted data therefore the Job object cannot be persisted once and loaded multiple times without loosing the log messages.

CHAPTER 5

Reference documentation

5.1 Runner

TODO

5.1.1 Runner API

class gitlab_runner_api.**Runner**

classmethod register(*api_url*, *token*, *description=None*, *active=None*, *locked=None*, *run_untagged=None*, *tags=None*, *maximum_timeout=None*, *name=None*, *version=None*, *revision=None*, *platform=None*, *architecture=None*, *executor=None*, *access_level=None*)

Register a new runner in GitLab.

api_url [str] URL for accessing the GitLab API

token [str] Registration token

description [str, optional] Runner's description

active [str, optional] Should Runner be active

locked [bool, optional] Should Runner be locked for current project

run_untagged [bool, optional] Should Runner handle untagged jobs

tags [list of str, optional] List of Runner's tags

maximum_timeout [int, optional] Maximum timeout set when this Runner will handle the job (in seconds)

name [str, optional] The runner's name

version [str, optional] The runner's version

revision [str, optional] The runner's revision

platform [str, optional] The runner's platform

architecture [str, optional] The runner's architecture

executor [str, optional] The runner's executor

access_level [str, optional] Limit the jobs which will be sent to the runner (for security)

Runner

classmethod load(*filename*)
Serialise this runner as a file which can be loaded with *Runner.load*.

filename [str] Path to file that represents the runner to initialise.

Runner

classmethod loads(*data*)
Serialise this runner as a file which can be loaded with *Runner.load*.

data [str] String representing the runner to initialise

Runner

dump(*filename*)
Serialise this runner as a file which can be loaded with *Runner.load*

filename [str] Registration token

dumps()
Serialise this runner as a string which can be loaded with with *Runner.loads*.

str String representation of the job that can be loaded with *Runner.loads*

request_job()
Request a new job to run.

Job or None

5.1.2 Runner properties

class gitlab_runner_api.**Runner**

active

api_url

architecture

check_auth()

description

executor

id

locked

maximum_timeout

name

platform

revision

run_untagged

tags

token

version

5.2 Job

TODO

5.2.1 Job API

class gitlab_runner_api.**Job**

classmethod load(*filename*)
Serialise this job as a file which can be loaded with *Job.load*.

filename [str] Path to file that represents the job to initialise.

Job

classmethod loads(*data*)
Serialise this job as a file which can be loaded with *Job.load*.

data [str] String representing the job to initialise

Job

dump(*filename*)
Serialise this job as a file which can be loaded with *Job.load*.

filename [str] Registration token

dumps()
Serialise this job as a string which can be loaded with with *Job.loads*.

str String representation of the job that can be loaded with *Job.loads*

set_success(*artifacts=None*)

set_failed(*failure_reason=None*, *artifacts=None*)

5.2.2 Job properties

class gitlab_runner_api.**Job**

after_script

auth()

commit_message

commit_sha

get_registry_credential(*image_name*)

id

is_branch

job_url

log

name

pipeline_id

pipeline_url

project_name

project_path

project_url

ref

repo_url

script

stage

state

token

username

variables

Index

T

U

V