Relay Commander Documentation

Release 0.0.12

LaunchDarkly Solutions Engineering Team

Dec 17, 2020

CONTENTS

A CLI for managing LaunchDarkly Relay Instances.

CHAPTER

ONE

QUICKSTART

1.1 Quickstart

1.1.1 Installation

You can install the latest version of relaycommander with:

```
pip install relaycommander
```

CHAPTER

TWO

RUNBOOK

2.1 RunBook

RelayCommander is a CLI tool that is intended to allow you to manually make a change to feature flags should your applications lose connectivity with the LaunchDarkly service. You must have the LD Relay setup with Redis in order for the CLI tool to work. It works by manually updating the status of a flag directly within Redis, while at the same time recording each update that has taken place. Then, once connection has been re-established with LaunchDarkly, you will then run a command that will update your configuration back to our service via the API. This iteration allows you to change the state of a feature flag to either ON or OFF. Due to the current way that SDK's work with redis, this can only be used to update the status of a backend feature flag.

2.1.1 Setup

- You can install relay commander with `pip install relaycommander`; this will enable the rc command globally.
- LD Relay proxy with Redis is setup
- Backend SDK clients are connected to the relay box

2.1.2 Instructions

Pre-requisites

- **Create a `.env` file similar to the sample file and be sure to update the following:**
  - **REDIS_HOSTS**
    - Update the `.env` file to include the host name(s) and port of the redis instances.

    If there are multiple redis instances running, provide as a CSV list of host names.
  - **LD_API_KEY**
    - LaunchDarkly API token to be used when writing the updates back to LaunchDarkly.

      Note that the API token requires administrative proviliges in order to work.

While there is a disconnect with LaunchDarkly

Update redis by running:

```
rc update -p project -e environment -f feature_flag -s state
```

- Project = the key of the project to be updated
- Environment = the key of the environment to be updated
- Feature = the key of the feature to be updated
- State = the state of the feature flag you would like to change it to. Currently allows you to set it to on or off

Each time this command is run, we will create a new direcoty called playback with a file containing the corresponding that needs to be run using the API

- No changes required: Changes will take effect once the relay cache detects the update and will broadcast the update to the SDK clients
- (Optional) Restart the relay proxy to server to make the updates immediate

Warning: NO CHANGES SHOULD BE MADE ON LAUNCHDARKLY.COM WHILE DISCONNECTED. IT IS RECOMMENDED THAT YOU EITHER HAVE AN INTERNAL PROCESS SO THAT NO ONE MAKES UPDATES DURING THIS TIME OR YOU DISABLE ALL LOGINS VIA SSO

Once LD is reconnected

Run the following command:

```
rc replay
```

Running this command will iterate through all of the files that were created during `rc update` and make the corresponding udpates in LaunchDakly via the API to synch it with the offline changes that were made

Verify that the current state in LaunchDarkly matches that last state that was set using RelayCommander

CHAPTER

THREE

CLI REFERENCE

3.1 CLI

3.1.1 CLI Interface

rc

A CLI for working with LaunchDarkly relay instances.

```
rc [OPTIONS] COMMAND [ARGS]...
```

Options

-v, --verbosity <LVL>
Either CRITICAL, ERROR, WARNING, INFO or DEBUG

--version
Show the version and exit.

--help
Show this message and exit.

generate-relay-config

Generate Relay Proxy Configuration.

Generate a ld-relay.conf file to quickly spin up a relay proxy. Right now this is mostly used for integration testing.

param project LaunchDarkly project key

```
rc generate-relay-config [OPTIONS]
```

Options

-p, --project <project>
    Required

playback

Execute commands in the replay/toDo directory.

```
rc playback [OPTIONS]
```

update-ld-api

Execute command against the LaunchDarkly API.

This command is generally not used directly, instead it is called as a part of running the `playback()` function.

param project LaunchDarkly project key.

param environment LaunchDarkly environment key.

param feature LaunchDarkly feature key.

param state State for a feature flag.

```
rc update-ld-api [OPTIONS]
```

Options

-p, --project <project>
    Required

-e, --environment <environment>
    Required

-f, --feature <feature>
    Required

-s, --state <state>
    Required

update-redis

Update redis state for a feature flag.

param project LaunchDarkly project key.

param environment LaunchDarkly environment key.

param feature LaunchDarkly feature key.

param state State for a feature flag.

```
rc update-redis [OPTIONS]
```

Options

`-p, --project` `<project>`
Required

`-e, --environment` `<environment>`
Required

`-f, --feature` `<feature>`
Required

`-s, --state` `<state>`
Required

CHAPTER

FOUR

API REFERENCE

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

4.1 API

4.1.1 relay_commander

Relay Commander is a CLI for managing LaunchDarkly relay instances.

copyright

(c) 2018-2019 by LaunchDarkly Solutions Engineering

license Apache 2.0, see LICENSE for more details.

4.1.2 Utilities

relay_commander.generators

This module allows for generating LaunchDarkly relay configurations using Jinja templates.

class `relay_commander.generators.`**`ConfigGenerator`**

Abstract configuration generator using Jinja.

`__init__``()`

Initialize self. See help(type(self)) for accurate signature.

`__weakref__`

list of weak references to the object (if defined)

`generate_relay_config``(`*environments*`)`

Generate ld-relay.conf file.

Given a list of environments of a project, this will generate a `ld-relay.conf` file in the current working directory. The conf file follows the specification that is documented in the main ld-relay documentation.

Parameters **`environments`** (`list`) – list of LaunchDarkly environments.

Return type `None`

relay_commander.validator

This module provides helper functions that validate CLI input.

`relay_commander.validator.`**`_REQUIRED_ENV_VARS = ['LD_API_KEY', 'REDIS_HOSTS']`**
Internal constant that defines required environment variables.

`relay_commander.validator.`**`_VALID_STATES = ['on', 'off']`**
Internal constant that defines a valid `state` argument.

`relay_commander.validator.`**`_check_env_var`**`(`*envvar*`)`
Check Environment Variable to verify that it is set and not empty.

Parameters **`envvar`** (`str`) – Environment Variable to Check.

Return type `bool`

Returns True if Environment Variable is set and not empty.

Raises KeyError if Environment Variable is not set or is empty.

New in version 0.0.12.

`relay_commander.validator.`**`valid_env_vars`**`()`
Validate that required env vars exist.

Return type `bool`

Returns True if required env vars exist.

New in version 0.0.12.

`relay_commander.validator.`**`valid_state`**`(`*state*`)`
Validate State Argument

Checks that either 'on' or 'off' was entered as an argument to the CLI and make it lower case.

Parameters **`state`** (`str`) – state to validate.

Return type `bool`

Returns True if state is valid.

Changed in version 0.0.12: This moethod was renamed from validateState to valid_state to conform to PEP-8. Also removed "magic" text for state and instead reference the _VALID_STATES constant.

relay_commander.replay_builder

This module provides functionality to generate the replay directory and keep track of pending and completed API calls.

As a part of the runbook, when a user makes a change directly to redis we make a copy of the command that they need to run in order to update the API when LaunchDarkly connectivity resumes.

These commands are stored in a directory called `replay` which has the following structure:

```
replay
├── archive
└── toDo
```

`relay_commander.replay_builder.`**`check_local`**`()`
Verify required directories exist.

This functions checks the current working directory to ensure that the required directories exist. If they do not exist, it will create them.

Return type `None`

`relay_commander.replay_builder.`**`create_file`**`(`*project*, *environment*, *feature*, *state*`)`

Create file to replay.

Create file with `rc` command that will be called against the LaunchDarkly API when `rc playback` is called from the main CLI.

Parameters

- **`project`** (`str`) – LaunchDarkly Project
- **`environment`** (`str`) – LaunchDarkly Environment
- **`feature`** (`str`) – LaunchDarkly Feature
- **`state`** (`str`) – State to update feature flag

Return type `None`

`relay_commander.replay_builder.`**`execute_replay`**`()`

Execute all commands.

For every command that is found in replay/toDo, execute each of them and move the file to the replay/archive directory.

Return type `None`

4.1.3 Wrappers

relay_commander.ld

This module provides a wrapper for the LaunchDarkly API.

Reference API - https://pypi.org/project/launchdarkly-api/

Changed in version 0.0.12: Refactor module to make it PEP-8 and PEP-484 compliant.

`class` `relay_commander.ld.`**`LaunchDarklyApi`**`(`*api_key*, *project_key=None*, *environment_key=None*`)`

Wrapper for the LaunchDarkly API

`__init__``(`*api_key*, *project_key=None*, *environment_key=None*`)`

Instantiate a new LaunchDarklyApi instance.

Parameters

- **`api_key`** (`str`) – API Access Key for LaunchDarkly.
- **`project_key`** (`Optional[str]`) – Key for project.
- **`environment_key`** (`Optional[str]`) – Environment in which to pull state from.

`__weakref__`

list of weak references to the object (if defined)

`get_environments``(`*project_key*`)`

Retrieve all environments for a given project.

Includes name, key, and mobile key.

Parameters **`project_key`** (`str`) – Key for project.

Return type `dict`

Returns dictionary of environments.

`update_flag`(*state*, *feature_key*)

Update the flag status for the specified feature flag.

Parameters

- **`state`** (`str`) – New feature flag state
- **`featureKey`** – Feature flag key

Return type `FeatureFlag`

Returns FeatureFlag object.

relay_commander.redis

This module provides an interface for working with redis.

`class` `relay_commander.redis_wrapper.`**`RedisWrapper`**(*host*, *port*, *project_key*, *environment_key*)

A wrapper around the redis library.

This class implements some general data access patterns as well as LaunchDarkly relay specific functionality.

Parameters

- **`host`** – redis hostname.
- **`port`** – redis port.
- **`project_key`** – LaunchDarkly project key
- **`environment_key`** – LaunchDarkly environment key.

`__init__`(*host*, *port*, *project_key*, *environment_key*)

Initialize self. See help(type(self)) for accurate signature.

`__weakref__`

list of weak references to the object (if defined)

`_format_key_name`()

Return formatted redis key name.

Return type `str`

`static connection_string_parser`(*uri*)

Parse Connection string to extract host and port.

Parameters **`uri`** (`str`) – full URI for redis connection in the form of host:port

Return type `list`

Returns list of RedisConnection objects

`get_flag_record`(*feature_key*)

Get feature flag record from redis.

Parameters **`feature_key`** (`str`) – key for feature flag

Return type `str`

Returns value of feature flag key in redis.

Raises KeyError if key is not found.

`update_flag_record`(*state*, *feature_key*)
Update redis record with new state.

Parameters

- **`state`** (`str`) – state for feature flag.
- **`feature_key`** (`str`) – key for feature flag.

Return type `None`

`relay_commander.redis_wrapper.`**`_DEFAULT_REDIS_PORT = 6379`**
Internal constant that defines the default redis port.

`class` `relay_commander.redis_wrapper.`**`_RedisConnection`**(*host*, *port*)
Private data class that represents a redis connection.

Parameters

- **`host`** (`str`) – hostname for redis
- **`port`** (`int`) – port for redis

Changed in version 0.0.12: Refactored to become private, and renamed to fix typo.

`__init__`(*host*, *port*)
Initialize self. See help(type(self)) for accurate signature.

`__weakref__`
list of weak references to the object (if defined)

CHAPTER

FIVE

DEVELOPER DOCUMENTATION

5.1 Developer Documentation

5.1.1 Installing Development Environment

This project uses pip for dependency management. You'll have a better time if you use a virtualenv.

1. Create a new virtualenv with `python -m venv venv`.
2. Activate the new virtualenv with `. venv/bin/activate`.
3. Install all of the project dependencies with `pip install -r dev-requirements.txt`.
4. Install relayCommander in editable form with `pip install -e .`
5. Try out some cli commands with `rc`.

5.1.2 Running Tests

Tests can be found in the `tests` directory.

You can run tests with `make tests`.

If you want to run a specific test file you can do so with:

```
python -m unittest tests/relay_commander/test$MODULE.py
```

Code Coverage

This project attempts to have 100% code coverage. when you run `make test` code coverage is automatically ran. You can view the code coverage report locally by opening up the index.html file in the `htmlcov` directory that gets created when you run `make test`.

5.1.3 Documentation

This project uses sphinx for documentation. You can generate the latest docs locally by running `make docs`. You can then view them by opening up the `index.html` file in the `docs/build/html` directory.

5.1.4 Linting and Style

This project follows the PEP 8 style guidelines. You can install `pylint` in order to ensure that all of your code is compliant with this standard.

5.1.5 Release Checklist

- update VERSION in version.py
- make sure CHANGELOG has release date and relevant changes
- git tag with the new version (make sure it matches version.py)

CHAPTER

SIX

CHANGELOG

6.1 Changelog

Here you can see the full list of changs between each relaycommander release.

6.1.1 Version 0.0.12

UNRELEASED

Core CLI:

- Implement environment variable validator to ensure that required environment variables are set in the environment.
- Implement helpers for version checking and logging.
- Add try/except logic for all calls to the LaunchDarkly API. Any exceptions coming from the LaunchDarkly API were previously uncaught.

Developer Experience:

- Add sphinx-click extension which allows us to automatically generate documentation for our click based CLI using sphinx.
- Switch away from `pipenv`. This was causing a few more problems than it was solving, so we moved back to using a standard virtualenv, pip, and requirements.txt file.
- Refactor everything for PEP-8 and PEP-484.

6.1.2 Version 0.0.11

Released on March 5, 2019

Warning: This release fixes a critical bug where commands were not automatically being replayed via the CLI. All users are urged to upgrade to this version as soon as possible if it is being used in production.

- Refactor CLI to accept `on` or `off` instead of `true` or `false`. This better aligns with the experience that a user would have if they were to flip a kill switch in the LaunchDarkly UI.
- Fix a critical bug where commands were not being replayed properly via the playback command.
- Add support for python 3.5, 3.6, and 3.7 - due to an upstream issue with swagger and python 3.7 we were not able to use the LD API wrapper in versions greater than 3.6. This is no longer the case.

- Add integration tests to test out the general flow of the run book on all supported python versions during CI.
- Add new command to generate a relay proxy configuration for a given project.

6.1.3 Version 0.0.10

Released on January 28, 2019

- **Add feature that allows users to override the default redis port as** a part of the configuration.

6.1.4 Version 0.0.9

Released on January 25, 2019

- Add feature to allow users to define multiple redis destinations. The CLI will then attempt to update all hosts and report back the status via the console.

Warning: All of the versions below have been unpublished from pypi to reduce confusion. If you need a specific version from below you can download the appropriate git tag and build the package from there.

6.1.5 Version 0.0.8

Released on January 17, 2019

- Configure Logging
- Fix bug that failed to update redis due to missing key

6.1.6 Version 0.0.7

Released on January 16, 2019

- Fix bug in CLI command that does not allow you to update redis.

6.1.7 Version 0.0.6

Released on January 16, 2019

- Added sphinx documentation

6.1.8 Version 0.0.5

Released on Dec 19, 2018

- First alpha release including base functionality, code coverage, and unit tests.

PYTHON MODULE INDEX

INDEX