# django-emojiwatch Documentation

Release 0.0.0

**Matt Bogosian** 

## Contents

	Cont	
	1.1	Introduction
		1.1.1 License
		1.1.2 Installation
		1.1.3 Requirements
1.2		Contributing to django-emojiwatch
		1.2.1 Filing Issues
		1.2.2 Submission Guidelines
	1.3	LICENSE
		1.3.1 The MIT License (MIT)
	1.4	CREDITS
		1.4.1 Contributors

Copyright and other protections apply. Please see the accompanying *LICENSE* and *CREDITS* file(s) for rights and restrictions governing use of this software. All rights not expressly waived or licensed are reserved. If those files are missing or appear to be modified from their originals, then please contact the author before viewing or using this software in any capacity.

django-emojiwatch is a bare bones Slack app for posting custom emoji updates to a designated channel. It is licensed under the MIT License. See the *LICENSE* file for details.

Contents 1

2 Contents

## CHAPTER 1

Contents

Copyright and other protections apply. Please see the accompanying *LICENSE* and *CREDITS* file(s) for rights and restrictions governing use of this software. All rights not expressly waived or licensed are reserved. If those files are missing or appear to be modified from their originals, then please contact the author before viewing or using this software in any capacity.

## 1.1 Introduction

django-emojiwatch is a bare bones Slack app for posting custom emoji updates to a designated channel. It is implemented as a Django app. It was loosely inspired by Khan Academy's emojiwatch, which provides similar functionality, but for hosting on on Google App Engine.

#### 1.1.1 License

django-emojiwatch is licensed under the MIT License. See the *LICENSE* file for details. Source code is available on GitHub.

## 1.1.2 Installation

### Django

Installation can be performed via pip (which will download and install the latest release):

```
% pip install django-emojiwatch
...
```

Alternately, you can download the sources (e.g., from GitHub) and run setup.py:

```
% git clone https://github.com/posita/django-emojiwatch
...
% cd django-emojiwatch
% python setup.py install
...
```

Now you can add it to your DJANGO\_SETTINGS\_MODULE:

```
INSTALLED_APPS = (
    # ...
    'emojiwatch',
)

EMOJIWATCH = {
    'slack_verification_token': '...',
}
```

And add it to your site-wide URLs:

```
from django.conf.urls import include, url

urlpatterns = (
    # ...
    url(
        r'^emojiwatch/', # or werever you want
        include('emojiwatch.urls'),
    ),
    # ...
)
```

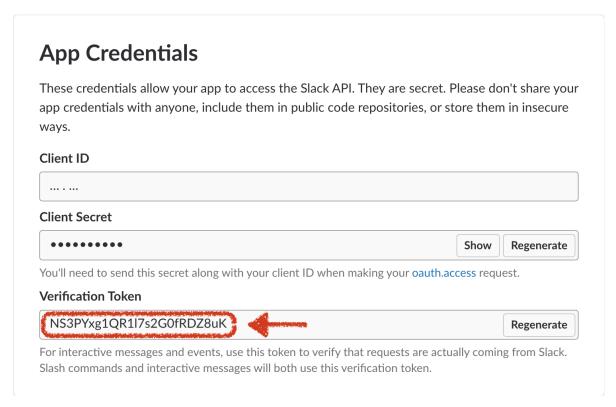
If you haven't already, you'll also need to enable the admin site for your Django installation.

#### Configuring Token Encryption in Django's Database

Auth tokens and notes associated with a watcher are encrypted in the Django database using django-fernet-fields. By default, the encryption key is derived from the SECRET\_KEY Django setting. To override this, use the FERNET\_KEYS and FERNET\_USE\_HKDF settings. See the docs for details.

## **Slack App and Watcher Setup**

- 1. Create a new Slack app or a legacy Slack app.
- 2. Once created, navigate to the Basic Information settings section and copy the Verification Token (e.g., NS3PYxg1QR117s2G0fRDZ8uK):



This is what you'll use as the EMOJIWATCH['slack\_verification\_token'] Django setting.

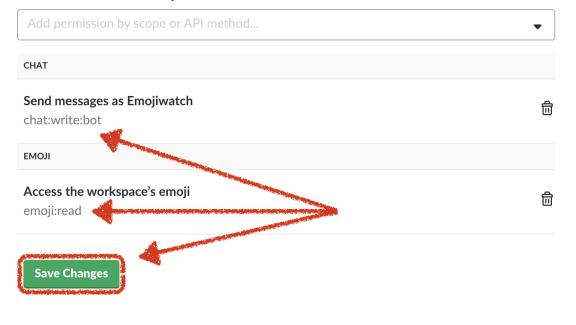
3. Add the emoji:read and chat:write (or chat:write:bot for legacy Slack apps) scopes to your app:

# Scopes define

Scopes define the API methods this app is allowed to call, and thus which infomation and capabilities are available on a workspace it's installed on. Many scopes are restricted to specific resources like channels or files.

If your app is submitted to the Slack App Directory, we'll review your reasons for requesting each scope. After your app is listed in the Directory, it will only be able to use permission scopes Slack has approved.

## **Select Permission Scopes**



4. Navigate to the OAuth & Permission features section. If necessary, click Install App to Workspace:

## OAuth Tokens & Redirect URLs

These OAuth Tokens will be automatically generated when you finish connecting the app to your workspace. You'll use these tokens to authenticate your app.



You'll be asked to authorize your new app in your workspace:

## Access your workspace's emoji

•

Emojiwatch will be able to access the names and images of custom emoji on posita.

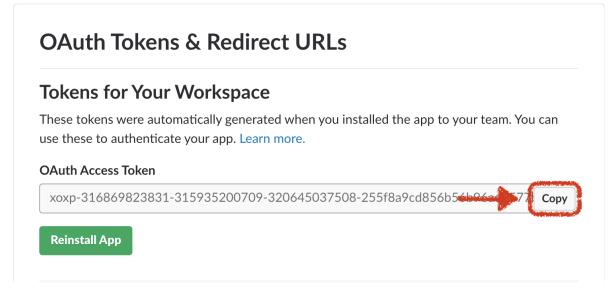
## Send messages as Emojiwatch



Emojiwatch will be able to send messages to posita.

Click Authorize.

5. Copy the OAuth Access Token (e.g., xoxp-3168...db0b5):



This is what you'll use when creating the Slack Workspace Emoji Watcher below.

6. If you haven't already, install emojiwatch into your Django project. (See the Django installation section above.) Navigate to your Django project's admin interface and add a new Slack Workspace Emoji Watcher with your Slack team ID, your OAuth access token, and the Slack channel ID to which you'd like Emojiwatch to post messages:

## Add Slack Workspace Emoji Watcher

Team ID:	T4P09SCHKT
Access Token:	xoxp-316869823831-315935200709-32064!
Channel ID:	C8VSYSEQ22
Icon Emoji:	:robot_face:

Your Slack team ID can be determined by navigating to any channel within your workspace, and looking at boot\_data.team\_id in your browser's JavaScript console:

```
>> boot_data.team_id
"T4P09SCHKT"
```

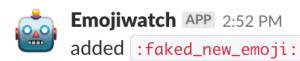
Your Slack channel ID can be found in the URL when navigating to that channel:

```
https://<workspace-name>.slack.com/messages/C8VSYSEQ22/details/
```

7. Once your Slack Workspace Emoji Watcher is saved, you should be able to test your configuration by faking a minimalist emoji\_changed event via curl:

```
curl --verbose --data '{
    "token": "NS3PYxg1QR117s2G0fRDZ8uK",
    "team_id": "T4P09SCHKT",
    "type": "event_callback",
    "event": {
        "type": "emoji_changed",
        "subtype": "add",
        "name": "faked-new-emoji",
        "value": "<some-img-url>"
    }
}' https://<django-project-base>/emojiwatch/event_hook
```

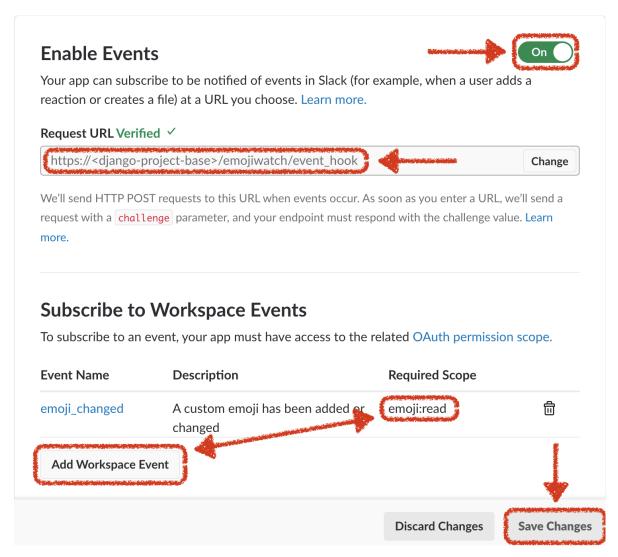
<django-project-base> is your domain, and optionally any path to your top-level Django project. If your Django project provides your root path, this will just be a domain name. Assuming everything has been set up correctly so far, this should result in a post to your Slack channel (e.g., C8VSYSEQ22):





If not, examine the output from your curl call for any clues as to what went wrong. See the *Troubleshooting* section below for additional suggestions.

8. Now you're ready to start receiving events. Navigate to your Slack app's Event Subscriptions features section. Turn events on and add your Django project's publicly-visible HTTPS URL. (This is the same URL you used with your curl command above.) Slack will attempt to post to that URL to verify its accessibility. Once verified, subscribe to the emoji:read event and click Save Changes.



9. That's it! You should now get notices to your designated channel whenever you add or remove custom Emojis to your workspace.

## **Troubleshooting**

If your curl command is succeeding, but you're still unable to see a post to your Slack channel, try turning on logging output via your Django settings. Here's a minimalist configuration if you don't already have one:

```
import logging
LOGGING = {
  'version': 1,
  'disable_existing_loggers': False,
  'formatters': {
    'standard': {
      'format': '%(asctime)s\t%(levelname)s\t%(name)s\t%(filename)s:
      \%(lineno)d\t%(message)s',
      },
    },
    'handlers': {
```

```
'default': {
      'class': 'logging.StreamHandler',
      'level': 'DEBUG',
      'formatter': 'standard',
    },
 },
  'loggers': {
    '': {
      'handlers': ['default'],
      'level': 'DEBUG',
      'propagate': False,
    'django': {
      'level': 'INFO',
      'propagate': True,
    },
 },
}
```

Try your curl command again. The Django console log should provide some clue as to what's wrong.

Some common causes are:

- Not properly adding or configuring the emojiwatch app in your Django project.
- Omitting or using an incorrect value for your EMOJIWATCH['slack\_verification\_token'] Django setting.
- Using an incorrect URL for your Django project instance or the django-emojiwatch event handler. (Note: Slack requires event handlers to support HTTPS.)
- Not creating (or neglecting to save) your Slack Workspace Emoji Watcher object via your Django project's admin interface.
- Using incorrect values for your team ID, access token, or channel ID.
- Failing to properly format a faked emoji\_changed event when invoking curl.

## 1.1.3 Requirements

You'll need a Slack account (and admin approval) for setting up your Slack app. A modern version of Python is required:

- cPython (2.7 or 3.4+)
- PyPy (Python 2.7 or 3.4+ compatible)

django-emojiwatch has the following dependencies (which will be installed automatically):

- Django (1.8 or higher)
- django-fernet-fields
- future
- slacker

Copyright and other protections apply. Please see the accompanying *LICENSE* and *CREDITS* file(s) for rights and restrictions governing use of this software. All rights not expressly waived or licensed are reserved. If those files are missing or appear to be modified from their originals, then please contact the author before viewing or using this software in any capacity.

## 1.2 Contributing to django-emojiwatch

There are several ways you can contribute.

## 1.2.1 Filing Issues

You can file new issues as you find them. Please avoid duplicating issues. "Writing Effective Bug Reports" by Elisabeth Hendrickson (PDF) may be helpful.

#### 1.2.2 Submission Guidelines

If you're willing and able, consider submitting a pull request (PR) with a fix. There are only a few guidelines:

• If it isn't already there, please add your name (and optionally your GitHub username, email, website address, or other contact information) to the *CREDITS* file:

```
...

* `Gordon the Turtle <https://github.com/GordonTheTurtle>`_
...
```

- Try to follow the source conventions as you observe them. (Note: I have purposely avoided aspects of PEP8, in part because I have adopted conventions developed from my experiences with other languages, but mostly because I'm growing older and more stubborn.)
- Provide tests where feasible and appropriate. At the very least, existing tests should not fail. (There are exceptions, but if there is any doubt, they probably don't apply.)

```
Unit tests live in ./tests. Tests can be run with tox [-e TOX_ENV] (requires Tox) or "${PYTHON:-python}" setup.py test.
```

There are two helper scripts that may be of interest. To set up a virtual environment (via virtualenv) for development and to run unit tests using Tox from that virtual environment, you can do the following:

```
( . ./helpers/venvsetup.sh && ./helpers/runtests.sh [-e TOX_ENV] )
```

- If you need me, mention me (@posita) in your comment, and describe specifically how I can help.
- If you want feedback on a work-in-progress (WIP), create a PR and prefix its title with something like, "NEED FEEDBACK -".
- If your PR is still in progress, but you aren't blocked on anything, prefix the title with something like, "WIP -".
- Once you're ready for a merge, resolve any merge conflicts, squash your commits, and provide a useful commit message. (This and this may be helpful.) Then prefix the PR's title to something like, "READY FOR MERGE -". I'll try to get to it as soon as I can.

## 1.3 LICENSE

## 1.3.1 The MIT License (MIT)

Copyright © 2015-2018 Matt Bogosian (@posita).

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use,

copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

#### 1.4 CREDITS

#### 1.4.1 Contributors

The following individuals or entities have contributed to this software.

By adding your name to this list, you grant a nonexclusive, perpetual license to your contributions to this software under the same terms as its *LICENSE*. Further, you warrant that your contributions to this software are exclusively your own creations and no one else has any superior right or claim to them. Finally, you agree to indemnify and hold harmless this software's owner against any colorable claim of infringement by a third party for this software's owner's otherwise lawful use of your contribution, whether or not such use was contemplated by you at the time you made it.

1.4. CREDITS