Jicket Documentation

Release 0.6.2

KWP GmbH & Co. KG

Feb 13, 2019

General

Jicket enables you to create a basic service helpdesk in Jira using Emails. It automatically creates issues for incoming emails and appends responses as comments.

CHAPTER 1

Overview

The goal of jicket was to create a stateless email importer to turn a Jira issue board into a very simple service helpdesk. Stateless means that all necessary information for operating is inferred from the emails themselves and Jira. This makes updating or migrating your jicket instance very easy, as you don't have to migrate any state data.

1.1 Ticket process

Jicket is continuously monitoring a mailbox for incoming emails. It parses those emails and then processes the email depending on the content. If the processing and subsequent import in Jira was successful, the email is moved into a specified folder from where on your service staff can interact with them.

When a mail is processed, jicket checks if the subject contains a `X-Jicket-HashID` header or if the subject line contains a ticket ID. If it does, the email is imported as a reply to an existing issue. If not, a new issue is created from the email.

1.1.1 New issue

When an email is identified as a new communication, jicket generates a new ticket ID and adds a new issue to the configured project. To confirm the creation of the ticket, an email is sent out to the customer and the ticket address which is meant to start an email thread. Also included is a modified subject which contains the ticket ID for this issue.

An example conversation could look like this:

```
Feature XY broken                                          Customer <foo@customer.com>
├── [#JI-ZOZ2P6] Feature XY broken                         Jicket <support@company.com>
│   ├── RE: [#JI-ZOZ2P6] Feature XY broken                 Fred Bobber <f.bobber@company.
↪com>
│   │   ├── RE: RE: [#JI-ZOZ2P6] Feature XY broken         Customer <foo@customer.com>
│   │   ├── RE: RE: [#JI-ZOZ2P6] Feature XY broken         Samantha Else <s.else@company.
↪com>
```

1.1.2 Reply to existing Issue

If the email is identified as a reply to an existing issue, a comment with the email's content is added to the issue. No further confirmation is sent to the customer.

CHAPTER 2

Installation

Jicket can be installed like any other python package. Additionally a convenient docker image is provided.

Jicket requires at least Python 3.6 to run.

2.1 Docker

Running jicket in a docker container is a convenient way to get started quickly or for testing it locally without having to worry about setting up the environment. You need to pass it some minimum configuration (mostly IMAP, SMTP and Jira account data) to get it running.

Jicket on Docker Hub

2.1.1 Running

Create a file `env.list` to store your environment variables. Make sure the rights for accessing the file are set correctly, especially the global read flag (`chmod o-rwx env.list`). Configure the environment variables according to *Configuration* in a `VAR=value` format, e.g.:

Listing 1: env.list

```
JICKET_IMAP_HOST=imap.example.com
JICKET_IMAP_PORT=993
JICKET_IMAP_USER=foo@example.com
JICKET_IMAP_PASS=correcthorsebatterystaple
```

The container is then launched:

```
>>> docker run -it --env-file env.list jicket
```

2.2 pip

Install the jicket package with pip:

```
>>> pip install jicket
```

Afterwards jicket can be launched with

```
>>> jicket
```

CHAPTER 3

Configuration

Jicket can be configured using both environment variables and command line arguments. Command line arguments take precedence over environment variables.

Warning: Using environment variables for configuring the username and password is highly recommended. If you pass them as command line arguments, they show up in the process list and will be readable for anyone with even basic access to the server.

3.1 IMAP

Configuration of the IMAP mailbox that is used to read incoming mails from.

3.1.1 Host

Environment `JICKET_IMAP_HOST`

CLI `--imaphost`

Type `str`

Required Yes

Description URL of IMAP mailbox that is receiving new ticket emails

Example `imap.example.com`

3.1.2 Port

Environment `JICKET_IMAP_PORT`

CLI `--imaphost`

Type `int`

Default `993`

Required No

Description Port of IMAP host

Example `993`

3.1.3 User

Environment `JICKET_IMAP_USER`

CLI `--imapuser`

Type `str`

Required Yes

Description Username for IMAP mailbox

Example `foo@example.com`

3.1.4 Password

Environment `JICKET_IMAP_PASS`

CLI `--imappass`

Type `str`

Required Yes

Description Password for IMAP user

Example `correcthorsebatterystaple`

3.2 SMTP

Configuration of the SMTP server that is used to send emails from.

3.2.1 Host

Environment `JICKET_SMTP_HOST`

CLI `--smtphost`

Type `str`

Required Yes

Description URL of SMTP server used to send out emails

Example `smtp.example.com`

3.2.2 Port

Environment `JICKET_SMTP_PORT`

CLI `--smtphost`

Type `int`

Default `587`

Required No

Description Port of SMTP server

Example `587`

3.2.3 User

Environment `JICKET_smtp_USER`

CLI `--smtpuser`

Type `str`

Required No

Description Username for SMTP server. If it is not explicitly provided, IMAP username will be used.

Example `foo@example.com`

3.2.4 Password

Environment `JICKET_SMTP_PASS`

CLI `--smtppass`

Type `str`

Required No

Description Password for SMTP user. If it is not explicitly provided, IMAP password will be used.

Example `correcthorsebatterystaple`

3.3 Jira

Configuration of jira instance on which new issues shall be created from incoming emails.

3.3.1 URL

Environment `JICKET_JIRA_URL`

CLI `--jiraurl`

Type `str`

Required Yes

Description URL of Jira instance that shall be used

Example `jira.example.com`

3.3.2 User

Environment `JICKET_JIRA_USER`

CLI `--jirauser`

Type `str`

Required Yes

Description Username for Jira access

Example `foo@example.com`

3.3.3 Password

Environment `JICKET_JIRA_PASS`

CLI `--jirapass`

Type `str`

Required Yes

Description Password for Jira user

Example `correcthorsebatterystaple`

3.3.4 Project

Environment `JICKET_JIRA_PROJECT`

CLI `--jiraproject`

Type `str`

Required Yes

Description The Project key in which new issues shall be created. It can be found in the URL of your project.

Example `SHD`

3.4 Email

Configuration regarding the mailbox and emails in general

3.4.1 Inbox

Environment `JICKET_FOLDER_INBOX`

CLI `--folderinbox`

Type `str`

Default `INBOX` (This is the name for the default IMAP inbox)

Required No

Description Folder from which emails shall be fetched for parsing. Using the default IMAP inbox is recommended unless you know what you're doing.

Example `mycoolfolder`

3.4.2 Success

Environment `JICKET_FOLDER_SUCCESS`

CLI `--foldersuccess`

Type `str`

Default `jicket`

Required No

Description Imap folder to which successfully imported emails shall be moved. The folder must exist and must not be the same as `JICKET_FOLDER_INBOX`.

Example `myothercoolfolder`

3.4.3 Thread template

Environment `JICKET_THREAD_TEMPLATE`

CLI `--threadtemplate`

Type `str`

Required Yes

Description Path to HTML file containing template for ticket thread emails. Can be absolute or relative path. See *Mail Template* on how to format the template.

Example `/etc/jicket/threadtemplate.html`

3.4.4 Ticket Address

Environment `JICKET_TICKET_ADDRESS`

CLI `--ticketaddress`

Type `str`

Required Yes

Description Email address of ticket system. This is the address your customers should contact, and from which they will in turn receive the ticket creation confirmation.

Example `support@example.com`

3.4.5 Ticket Address

Environment `JICKET_FILTER_CONFIG`

CLI `--filterconfig`

Type `str`

Required No

Description Path to a JSON file containing the config for the email filter. See *Filtering*

Example `/etc/jicket/filter.json`

3.5 Operation

Configuration of jicket operation

3.5.1 Loopmode

Environment `JICKET_LOOPMODE`

CLI `--loopmode`

Type `str`

Default `dynamic`

Required No

Description How the main loop shall operate.

dynamic After finishing with fetching and processing the main loop will sleep for `JICKET_LOOPTIME` before fetching again.

interval Tries to run the main loop exactly every `JICKET_LOOPTIME` seconds. If main loop execution takes longer than that, there is no break between subsequent executions.

singleshot Program runs exactly once and then exits. This is particularily useful if you run jicket as a serverless function, for example on AWS Lambda

Example `interval`

3.5.2 Looptime

Environment `JICKET_LOOPTIME`

CLI `--looptime`

Type `float`

Default `60`

Required No

Description Length between loop execution. Also see `JICKET_LOOPMODE` how exactly this time is applied.

Example `120`

3.6 Ticket ID

Miscellaneous configuration

3.6.1 Prefix

Environment `JICKET_ID_PREFIX`

CLI `--idprefix`

Type `str`

Default `JI-`

Required No

Description A prefix that is prepended to ticket IDs. This could for example be your company initials.

Example `EC-` will produce ticket IDs like `[#EC-XXXXX]`

3.6.2 Hash salt

Environment `JICKET_ID_SALT`

CLI `--idsalt`

Type `str`

Default `JicketSalt`

Required No

Description The salt for hashing ticket IDs. Only needs to be set if you don't want your users to be able to find out the true ID of the ticket (which is the email's UID).

Example `VerySecretSalt`

3.6.3 Hash alphabet

Environment `JICKET_ID_ALPHABET`

CLI `--idalphabet`

Type `str`

Default `ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890`

Required No

Description Alphabet for hashing. The generated hash will only consist of letters from this alphabet.

Example `ABCD1234`

3.6.4 Hash minimum length

Environment `JICKET_ID_ALPHABET`

CLI `--idalphabet`

Type `int`

Default `6`

Required No

Description Minimum length of generated hash. If the email uid is low, a hash might consist of only one character if no minimum length is set. Must be positive or zero.

Example `0`

CHAPTER 4

Mail Template

The contents of the confirmation mail is generated from a template. Some variables can be accessed to dynamically generate a response to incoming emails.

4.1 Template syntax

The template should be written as valid HTML, just as if you would write a regular mail. You can place named substitutes for use with string interpolation in your template. The syntax for them is `%(NAME)TYPE`. For example, if you want the subject as a string, you'd put `$(subject)s` at the appropriate location in your template. See *Interpolation variables* for a list of available variables.

An example template could look like this:

```
<html>
  <head></head>
  <body>
    <p>Hello!<br>
        <br>
        Thank you for contacting the support. This mail indicates that your ticket␣
↪has been successfully created and will be processed soon.<br>
        Please always keep the Ticket-ID in the subject, otherwise we won't be able␣
↪to track your issue properly.<br>
        <br>
        <br>
        Ticket ID: %(ticketid)s<br>
        Ticket Subject: %(subject)s<br>
        <br>
        <br>
        This mail was automatically generated by <a href="https://github.com/kwp-
↪communications/jicket">Jicket</a>
    </p>
  </body>
</html>
```

4.2 Interpolation variables

4.2.1 Subject

Name `subject`

Type `s`

Description Subject of ticket

Example Re: The Website Is Down

4.2.2 Ticket ID

Name `ticketid`

Type `s`

Description Hashed ID of ticket

Example K6NPD4

CHAPTER 5

Filtering

Jicket offers some capabilities to blacklist emails by their address and subject lines using regex patterns. Additionally, exceptions to blacklistings can be added with a whitelist.

5.1 Filter Configuration File

The filter configuration file is a JSON formatted file. The root objects contains two lists, `blacklist` and `whitelist`. Each entry in the lists is an object, consisting of the properties `description`, `addresspattern` and `subjectpattern`.

5.1.1 Description

Property `description`

Type `str`

Required Yes

Description Description of the filter rule, which will be printed to the logs when the filter applies.

Example `Block emails matching .*@spameridoo.com for excessive spam`

5.1.2 Address pattern

Property `addresspattern`

Type `str`

Required No

Description Python regex pattern which is matched with the address.

Example `.*@spameridoo.com`

5.1.3 Subject Pattern

Property `subjectpattern`

Type `str`

Required No

Description Python regex pattern which is matched with the subject.

Example `buy my spam`

5.1.4 Ignore case

Property `ignorecase`

Type `bool`

Required No

Description Whether regex shall be case insensitive

Example `true`

Default `false`

Blacklisting and Whitelisting

Each mail is matched against all available blacklist filters. If any of them matches, the mail is also checked against all whitelist filters. If only blacklisting rules match, the mail is marked ased filtered and will not be further processed and will be moved. If a whitelist filter matches, this mark is reset and the mail import continues as usual.

Example

The following example contains a simple filter setup. The only blacklist rule filters out all emails coming from the domain `test.com`. However, two whitelist rules will let through emails from foo.com if they either come from `foo@test.com`, or if the subject contains the magic keyword `sesame`.

```
{
  "blacklist": [
    {
      "description": "Deny mails from domain test.com",
      "addresspattern": ".*@test\\.com"
    }
  ],
  "whitelist": [
    {
      "description": "Allow foo@test.com",
      "addresspattern": "foo@test\\.com"
    },
    {
      "description": "Allow because of magic word 'sesame' in subject",
      "subjectpattern": "sesame"
    }
  ]
}
```

CHAPTER 6

Indices and tables

- genindex
- modindex
- search