Jipdate Documentation

Linaro, Joakim Bech

Sep 14, 2023

Table of Contents

CHAPTER 1

About

Jipdate is a tool used at Linaro that makes it possible in one go to:

- Update all tickets belonging to a certain user using command line and your favorite editor.
- Create (and save) a status report.
- Move Jira tickets into different states (*To Do*, *In Progess* etc.).
- Retrieve updates on all tickets, to generate weekly or monthly reports

If you want to give it a try, then head over to the *Installation instructions* and then have a look at the *Jipdate Examples* or *Jipstatus Examples*.

CHAPTER 2

Installation instructions

First step is to get the required packages, then you need to clone Jipdate itself and then finally install all the needed Python packages.

The video clip below will show how to install it and how to do a simple update with state changes and adding comments to a Jira ticket. Note that you are **not** required to run it in a Docker container. The only reason for doing that was to try the installation from a clean system to ensure that things are working as supposed to when starting out from scratch.

Note: Jipdate is **Python3** only! So at line 2 (*python3-pip*) and 5 (*pip3*) you have to adjust accordingly. I.e., if Python3 is default on your distro, then it might be that the package is simply `python-pip` (like on Arch Linux for example). The important message is that you **only** use Jipdate with Python3!

2.1 Install jipdate with pip

```
$ pip3 install --user jipdate
```

2.2 Ubuntu / Debian based systems

```
1 $ sudo apt install python3-pip git
2 $ git clone https://github.com/Linaro/jipdate.git
3 $ cd jipdate
4 $ pip3 install --user flit
5 $ flit build
6 $ flit install --symlink
```

2.3 Fedora / Red Hat based systems

```
$ sudo dnf -y install python3-pyyaml python3-jira
$ git clone https://github.com/Linaro/jipdate.git
$ cd jipdate
$ pip3 install --user flit
$ flit build
$ flit install --symlink
```

2.4 Arch Linux

```
$ sudo pacman -Syu
$ sudo pacman -S extra/git extra/python extra/python-pip
$ git clone https://github.com/Linaro/jipdate.git
$ cd jipdate
$ pip3 install --user flit
$ flit build
$ flit install --symlink
```

2.5 Mac OS (v12 Monterey or later)

```
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
$ brew install python git
$ git clone https://github.com/Linaro/jipdate.git
$ cd jipdate
$ pip3 install --user flit
$ flit build
$ flit install --symlink
```

CHAPTER 3

Run Jipdate

The most straight forward way is to simply run the script

```
$ jipdate
```

Running it without any parameters (or with -h) will give you a list of all parameters. If you want to see examples of how to combine flags/parameters, then head over to *Jipdate Examples*.

CHAPTER 4

Run Jipstatus

The most straight forward way is to simply run the script

```
$ jipstatus
```

Running it without any parameters will show all tickets updates for the current user in the last 7 days, including newly created tickets, state transitions, and all updates.

Running with `-h` will give you a list of all parameters. If you want to see examples of how to combine flags/parameters, then head over to *Jipstatus Examples*.

4.1 Environment variables

You can export both the password and the username with environment variables and thereby avoid having to enter them every time you run Jipdate.

```
$ export JIRA_USERNAME="john.doe@linaro.org"
$ export JIRA_PASSWORD="my-super-secret-password"
```

Warning: By exporting `JIRA_PASSWORD` there is a chance that the password end up in the history of your shell (bash, zsh etc).

The same can be achieved by storing the credentials in *username* and *password* in the *Config file*.

CHAPTER 5

Jipdate Examples

On this page you will find a list of examples of things that a Jipdate user might want to do. It should be noted that this is not a complete list, but will probably list the use cases that are most commonly used.c. Likewise each and every combination of flags are not listed here. But you can of course mix flags to include, exclude Jira tickets and so on.

Table of Contents

5.1 Update tickets

5.1.1 I want to update only my Epics

```
$ jipdate -q -e
```

5.1.2 I want to update my Initiatives and Epics

```
$ jipdate -q
```

5.1.3 I want to update all my tickets (Initiatives, Epics, Stories, Sub tasks)

```
$ jipdate -q --all
```

5.1.4 I want to update only my Epics and reuse my previous comment(s)

```
$ jipdate -q -e -l
```

Here it's the `-l` that makes the difference and Jipdate will pull the last comment from the ticket(s) and include that in each section for each and every Jira ticket assigned to you.

5.1.5 I want to update my Initiatives and Epics and reuse my previous comment(s)

```
$ jipdate -q -l
```

Here it's the `-l` that makes the difference and Jipdate will pull the last comment from the ticket(s) and include that in each section for each and every Jira ticket assigned to you.

5.1.6 I want to change state of my card

Run Jipdate with any parameter that suits your needs. Here we're getting everything.

```
$ jipdate -q --all
```

In your Editor you will see a section for each Jira ticket (based on your given parameters to Jipdate). It could look like this:

```
...
[SWG-368]
# Header: Demo / Test issue three
# Type: Epic
# Status: Open
# No updates since last week.
...
```

Here you can see it in the `Open` state (`# Status Open`). If you want to change this to another state, then simply uncomment the line and write another state for it, i.e., change like we've done at line 4 here.

```
[SWG-368]
# Header: Demo / Test issue three
# Type: Epic
Status: To do
# No updates since last week.
```

Note: Upper/lower case doesn't matter for the status change, nor does spaces before or after matter. But it needs to be written as in Jira otherwise. If you get it wrong, Jipdate will return an error and also show the possible combinations. Example. `todo` is wrong, but `to do` is correct!

```
...
[SWG-368]
# Header: Demo / Test issue three
# Type: Epic
Status: In progress
Time spent: 4h
Updates since last week.
...
```

Here you can see `Status` in `In progress`, and `Time spent` on this issue is ste to 4 hours. There are also some updates, and these updates `Updates since last week.` will be updated under `Comments` and under `Work log` in that ticket.

Note: `Time spent` can be written in the formats: `5m`, `5h`, `5d` and `5w`, and that means `m (minutes), h (hours), d (days), w (weeks).`

5.2 Updates with status reports

5.2.1 I want to update my Epics and create a status report

```
$ jipdate -q -e -f status_report_week_xy.txt
```

When the script has finished running you will have a file `status_report_week_xy.txt` in the folder with your entire status update ready to be sent out via email, for archiving or copy/pasted into a combined status document.

Note: Updating like this with the `-q` (query) will overwrite the file you have specified.

5.2.2 I want to update my issues based on my own status file

```
$ jipdate -f my_status.txt
```

The use case here is that you have a Jipdate status file stored locally that you update on regular basis and you basically never query Jira itself.

5.3 Special use cases

5.3.1 I want to see what tickets person firstname.lastname are working with

```
$ jipdate -q -u john.doe
```

Note: For this you still need to enter your own password even though you make a query about another user.

5.3.2 I want to see the last updates on tickets assigned to firstname.lastname

```
$ jipdate -q -u john.doe -l
```

Note: For this you still need to enter your own password even though you make a query about another user.

5.3.3 I only want to print my status to stdout

```
$ jipdate -q -p
```

This can be combined with other flags (e.g. `--all`, `-e` etc).

5.4 Testing / development

5.4.1 I want to use a test-server / sandbox

```
$ jipdate -t -q
```

Here we provide `-t` which will use Linaro's test server instead of the real Jira instance. This is totally safe to use when playing around and testing Jipdate. You can of course combine this with all other parameters.

5.4.2 I want to do a dry-run

```
$ jipdate -q --dry-run
```

With `--dry-run` you can query the real Jira instance without risking to make any updates. I.e., this can be used as a complement to query the test server.

5.4.3 I want to see more debugging text from Jipdate

```
$ jipdate -q -v
```

CHAPTER 6

Jipcreate Examples

On this page you will find a list of examples of things that a Jipcreate user might want to do. It should be noted that this is not a complete list, but will probably list the use cases that are most commonly used. Likewise each and every combination of keywords that can be specified in the yaml file are not listed here. But you can of course add/remove keywords to set values in the storey created, or remove to use the default value in the story.

Table of Contents

6.1 Create issue

6.1.1 I want to create a story

```
$ jipcreate -f /path/to/file.yaml
```

Fields that can be found in the yaml file are the following for a ‘Story’ or ‘Ticket’. This is an example yaml file.

```
- IssueType: Story
  Project: LKQ
  Summary: Make LKFT work in tuxtest
  Description: |-
```

(continues on next page)

(continued from previous page)

```
    Some text can be written here.
  AssigneeEmail: firstname.surname@linaro.org
  OriginalEstimate: 5h|5d|5w
  Share Visibility: [donald.duck@linaro.org,micky.mouse@linaro.org]
  Sprint: sprint name
```

Fields that can be found in the yaml file are the following for a ‘Epic’. This is an example yaml file.

```
- IssueType: Epic
  Project: LKQ
  Summary: Testing Epic shared visibility
  Epic Name: Different Testing Epic shared visibility
  Description: |-
    Some text can be written here.
  AssigneeEmail: firstname.surname@linaro.org
  Share Visibility: [donald.duck@linaro.org,micky.mouse@linaro.org]
  Due date: 2023-06-31
```

NOTE: Required fields depends on what is configured in your JIRA instance.

6.2 Testing / development

6.2.1 I want to do a dry-run

```
$ jipcreate --dry-run
```

With `--dry-run` you can query the real Jira instance without risking to make any updates. I.e., this can be used to see what will be sent to the *server*.

6.2.2 I want to see more debugging text from Jipdate

```
$ jipcreate -v
```

CHAPTER 7

Jipstatus Examples

On this page you will find a list of examples of things that a Jipdate user might want to do with the *jipstatus* command. It should be noted that this is not a complete list, but will probably list the use cases that are most commonly used.c.

Table of Contents

7.1 Retrieve my updates

7.1.1 I want to retrieve all my updates

The default behavior for *jipstatus* when no specific arguments are used is to query the Jira server for all the updates for the current user in the last week.

```
$ jipstatus
```

The *–days* argument can be used to query for any arbitrary duration (in days):

```
$ jipstatus --days 30
```

7.1.2 I want to generate an HTML output

The argument *–html* can be used to generate a report in HTML format. By default *jipstatus* will create the file *status.html*, which can be changed by the user:

```
$ jipstatus --html [file.html]
```

7.2 Retrieve updates for a specific Jira project

Instead of querying Jira for a specific user, *–project* can be used to request updates to all tickets from a specific project.

```
$ jipstatus --project <PJT_KEY>
```

Several arguments can be combined together:

```
$ jipstatus --project <PJT_KEY> --days 30 --html
```

7.3 Retrieve updates for a specific Jira team

The *–team* argument can be used to retrieve updates from all users who belong to a specific Jira team (or group):

```
$ jipstatus --team linaro --days 30 --html
```

7.4 Retrieve updates for a specific Jira user

The *–user* argument can be used to retrieve update from a specific user, it can be the user email address, or the short form firstname.lastname.

```
$ jipstatus --user jane.doe --days 30 --html
```

7.5 I want to see more debugging text from Jipstatus

You can use *-v* to request verbose output.

```
$ jipstatus -v
```

CHAPTER 8

Config file

8.1 Location

If you haven't used Jipdate before, then the script will create a `.jipdate.yml` config automatically. It will store it at one of the following locations.

```
$HOME/.config/jipdate/.jipdate.yml
$HOME/.jipdate.yml
$HOME/.../<jipdate-script-dir>/.jipdate.yml
```

You can move it to any of the three folders if you have any preference.

8.2 Example config

Jipdate config files are written in YAML format and a typical config file looks like this:

```
# Config file for jipdate
# For use in future (backwards compatibility)
version: 1

# Jira server information
#server:
#  url: https://linaro.atlassian.net
#  token: abcdefghijkl

#test_server:
#  url: https://<name_of_test_instance>.atlassian.net
#  token: abcdefghijkl

# Extra comments added to each Jira issue (multiline is OK)
comments:
    - "# No updates since last week."
```

(continues on next page)

(continued from previous page)

```
# Header of the file (multiline is OK). It will be followed by
header:
    - |
      Hi,

      This is the status update from me for the
      last week.

      Cheers!

# Set this to 'True' if you want to get the issue header merged with the issue
# number.
use_combined_issue_header: False

# Default separator in the issue header, change to the separator of your own
# preference.
separator: ' | '
text-editor: True
username: john.doe@linaro.org
```

You will need to re-run the script after making a changes to the config file.

8.3 Config options

8.3.1 comments

This can be used to put a default line(s) showing up at each Jira ticket when doing a query. I.e., this is what is shown at line **5** and **11** in the example below.

```
1  [SWG-355]
2  # Header: Centralize and update OP-TEE documentation
3  # Type: Epic
4  # Status: In Progress
5  # No updates since last week.
6
7  [SWG-338]
8  # Header: OP-TEE issues at GitHub
9  # Type: Epic
10 # Status: To Do
11 # No updates since last week.
```

8.3.2 header

This where you can put general information that you want to provide with a status update in email format for example. This will always be put on top in the output from Jira, i.e., before any individual ticket sections. So with the text in the *Example config* above, jipdate will produce this (see line 3-7).

```
1  Subject: [Weekly] Week ending 2019-01-22
2
3  Hi,
4
```

(continues on next page)

(continued from previous page)

```
This is the status update from me for the last week.

Cheers!


John Doe


[SWG-355]
# Header: Centralize and update OP-TEE documentation
# Type: Epic
# Status: In Progress
# No updates since last week.
...
```

The above is the short default example. You could of course be more creative here and instead list a couple of different sections that are relevant to the status report for your team. For example something like this might be more useful.

```
header:
    - |
      Hi,

      This is the status update from me for the last week.

      * Ongoing:
       ** Jira
          For individual tickets I'm working with, please have a look at the
          Jira sections below.

       ** None Jira:

      * Travels
        No planned travels

      * Vacations / time-off:
        No planned time off in the coming weeks.

      // Regards
```

Which would generate this:

```
Subject: [Weekly] Week ending 2019-01-22

Hi,

This is the status update from me for the last week.

* Ongoing:
  ** Jira
       For individual tickets I'm working with, please have a look at the
           Jira sections below.

  ** None Jira:

* Travels
  No planned travels
```

(continues on next page)

(continued from previous page)

```

* Vacations / time-off:
  No planned time off in the coming weeks.

// Regards
Joakim Bech

[SWG-355]
...
```

8.3.3 use_combined_issue_header

This will decide if `[XYZ-123]` and the name of the ticket name should be merged into a single line or not. I.e.

`use_combined_issue_header: False` gives:

```
[SWG-355]
# Header: Centralize and update OP-TEE documentation
# Type: Epic
# Status: In Progress
# No updates since last week.
...
```

`use_combined_issue_header: True` gives:

```
[SWG-355 | Centralize and update OP-TEE documentation]
# Type: Epic
# Status: In Progress
# No updates since last week.
```

8.3.4 separator

This gives the Jipdate user the ability to use another separate than the default `|`. This is only useful if `use_combined_issue_header: True`. For example if you change it to:

```
separator: ': '
```

Then Jipdate will output Jira sections like this:

```
[SWG-355: Centralize and update OP-TEE documentation]
# Type: Epic
# Status: In Progress
# No updates since last week.
```

8.3.5 text-editor

This is a simple `True` and `False` deciding whether you would like to spawn your preferred text editor with the results after a successfull Jira query. To make use of this, please set the `EDITOR` enviroment variable before invoking Jipdate, e.g., something like:

```
export EDITOR=vim
```

8.3.6 server

Jipdate will use a default (Linaro) Jira server, but the user can set the Jira server in the configuration file. Authentication is required when querying a Jira server, and jipdate supports password based authentication, as well as token based. Using token authentication is considered more secure than password. The *server* configuration must include at least an *url* attribute. A *token* attribute can be added optionally. To create an API token, follow the procedure described in the Atlassian documentation.

When the *token* attribute is present, Jipdate will use token based authentication. Otherwise, it will use password based authentication.

Here is an example to show how to add a *server* entry in the configuration file, assuming token based authentication:

```
# Jira server information
server:
  url: https://linaro.atlassian.net
  token: abcdefghijkl
```

8.3.7 username

If this exists in the config, then jipdate will **not** ask for the username when running the script (see the *Example config* for the syntax).

8.3.8 password

This is similar to the `username` above, i.e., if your Jira password is stored here, then jipdate won't ask for it when running the script.

Warning: Storing your password here in clear text requires some extra precaution, since anyone with access to your computer can rather easy read the contents of this file. I.e., it's not really recommended to use this feature, you're better off just typing the password when running the script or export `JIRA_PASSWORD` as an enviroment variable, when you need to run script multiple times in row and don't want to type it in each and every time.

CHAPTER 9

Problems

The *Installation instructions* has been tested on a brand new and clean docker instance running Ubuntu, so it *should* work. Having that said, software is software and people are using different tools, distros etc. Below is a set of issues that people have seen (and that still can be seen).

9.1 CAPTCHA error

If you login using **wrong** credentials and the authentication has failed a certain number of times, then the Jira server will lock down and there isn't an easy way to reset it, other than successfully logging it via the web interface (or having IT reset the error count.)

When that happens you typically will see a message like this:

```
JIRA - please go to JIRA using your web browser, log out of JIRA, log back
in entering the captcha; after that is done, please re-run the script.
```

9.2 UnicodeEncodeError

Typically you will see an error message similar to this:

```
UnicodeEncodeError: 'ascii' codec can't encode character '\xa0' in position
57: ordinal not in range(128)
```

We have smoked out quite a few of those, but it still shows up from time to time. If you see them, please just file a bug at GitHub (or try to fix it and send at patch). Please submit issues at GitHub Jipdate Issues. The reason for that error message is that there are non ASCII characters in either the Jira ticket itself or in the users username (e.g. *Josè Doe*).