Research & Development at Limina.Studio

How (and Why) I Switched from JIRA to Clubhouse

2016 September 8
tags: Clubhouse, Go, importer, JIRA
by Tedb0t

My team of ~10-15 people had been using JIRA Cloud, the ticketing/project management system, for a few years, and I had grown to loath it (why?). Earlier this year I began considering alternatives in earnest, and came across Clubhouse.io via a Hacker News post.

Why Clubhouse?

Right off the bat, I was in love: it's beautiful, modern, innovative, tailored to modern agile product development, and above all, FAST. (Oh, and it's got a Clojure/Datomic backend too, which I think is neat!) Despite a proliferation of Kanban Klones, the most well-known being Trello, none of the options I looked at actually worked the way I wanted to work: hierarchical project-epic-story organization, quick filtering, built-in smart Git integrations, etc.. This blog post is a great overview of why to switch to Clubhouse, but needs to be updated, since many of those 'missing' features have since been added. Clubhouse is rapidly and continuously releasing effective improvements, and seem to be passionate about creating a great tool with great customer service to boot. Oh, AND it's free for up to 3 users, AND apparently completely free for qualifying nonprofits.

That being said, it was still a bit of an uphill battle switching: first I had to convince the rest of the management team that it was worthwhile, and then I'd have to figure out how to move our entire JIRA database and get everyone onboard.

I even made a quick Keynote specifically to describe why I wanted to switch to Clubhouse:

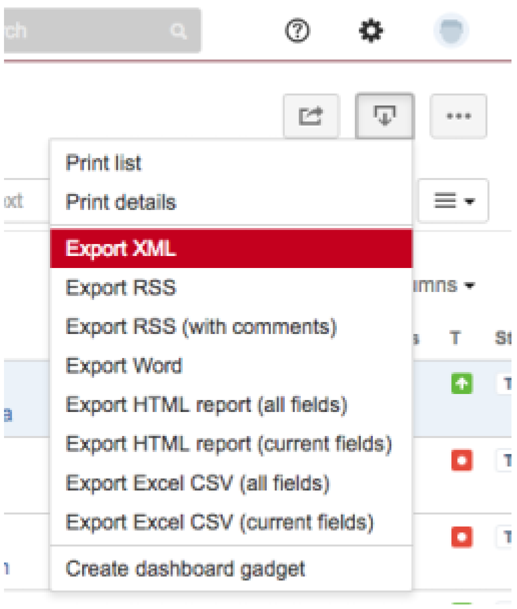
I'm happy to say that I finally did it—here's how.

Importing Tickets

We have over 2000 JIRA tickets, most of which are now completed. I briefly considered starting with just a nice, clean blank slate, but there's too much detail in current open tickets we would lose. Thankfully, I found an importer script, so I forked it and added a number of improvements (and got to learn Go while I was at it!)

Setting Up

First you'll need a Clubhouse organization and an API Token. Once you've got your org set up, go to `https://app.clubhouse.io/[your_organization]/settings/account/api-tokens` to create a token. Copy that and do `export CLUBHOUSE_API_TOKEN=asdflkjaseflkjdf` in your terminal so the token will be ready to go.


Print list
Print details
Export XML
Export RSS
Export RSS (with comments)
Export Word
Export HTML report (all fields)
Export HTML report (current fields)
Export Excel CSV (all fields)
Export Excel CSV (current fields)
Create dashboard gadget

Then you'll need to export your JIRA tickets. The easiest (or least horrible) way to do this is to do an Issue Search for all the tickets you want to export, then click the "Download" button near the top right, then "Export XML". Note: you can right click and "Save As…" right on that link, otherwise it'll open the XML right there in your browser. Yes, thanks, JIRA.

Now that you've got your input file (in the same dir as the importer), you'll need to create a "user mapping" JSON file (say, `userMap.json`) that describes how you want to map JIRA users to Clubhouse projects and User IDs. This looks like:

```
[
	{
		"jiraUsername": "userA",
		"chProjectID": 5,
		"chID": "476257c9-ac5a-46bc-67d6-4bc8bbfde7be"
	},
	{
		"jiraUsername": "userB",
		"chProjectID": 81,
		"chID": "476257c9-ac5a-46bc-67d6-4bc8bbfde7be"
	},
	{
		"jiraUsername": "userC",
		"chProjectID": 6,
		"chID": "476257c9-ac5a-46bc-67d6-4bc8bbfde7be"
	}
]
```

userMaps.json hosted with ❤ by GitHub view raw

If you open up your JIRA export XML file, you'll see elements like:

```
<assignee username="ted">T3db0t</assignee>
```

The importer uses only the `username` attribute, so make sure you use those in your `userMap.json` file.

Since you may want to assign projects by user, you can put the project IDs in as such. If you only have one Clubhouse project, just put that ID in for every user. To obtain your Clubhouse project IDs, you can do:

```
curl -X GET \
-H "Content-Type: application/json" \
-L "https://api.clubhouse.io/api/v1/projects?token=$CLUBHOUSE_API_TOKEN" | python -m
json.tool &gt; projects.json
```

Then look in that output file for the Project IDs.

Similarly, to obtain Clubhouse User IDs, you can do:

```
curl -X GET -H "Content-Type: application/json" -L "https://api.clubhouse.io/api/v1/users?
token=$CLUBHOUSE_API_TOKEN" | python -m json.tool &gt; users.json
```

Lastly, you need to setup your workflow mappings, i.e. JIRA "Selected for Development" -> Clubhouse "Selected for Sprint" or however you want it. To obtain your Clubhouse workflow state IDs, do:

```
curl -X GET \
-H "Content-Type: application/json" \
-L "https://api.clubhouse.io/api/v1/workflows?token=$CLUBHOUSE_API_TOKEN"
```

Right now the workflow mapping is only done in the script (no external config file, sorry!), so you'll need to go to `jiraStructs.go` and update this switch statement with your Clubhouse state IDs:

```go
switch item.Status {
    case "Ready for Test":
        // ready for test
        state = 500000010
    case "Task In Progress":
        // in progress
        state = 500000015
    case "Selected for Review/Development":
        // selected
        state = 500000011
    case "Task backlog":
        // backlog
        state = 500000014
    case "Done":
        // Completed
        state = 500000012
    case "Verified":
        // Completed
        state = 500000012
    case "Closed":
        state = 500000021
    default:
        // backlog
        state = 500000014
}
```

workflows.go hosted with ❤ by GitHub view raw

Pull the Trigger

OK, on to the main event: using the importer. With your `userMap.json` file ready, do:

```
go run *.go import --in SearchRequest.xml --map userMap.json --token $CLUBHOUSE_API_TOKEN --
test
```

This will run the tool in test mode, which will parse all the input files and show you what it's going to do, but won't upload data to Clubhouse. Once you're satisfied, run the tool without the test flag, open up your Clubhouse workspace and watch those tickets roll in!

Also included are a few utility scripts in Python: createTestStory.py, deleteArchivedStories.py and deleteEmptyEpics.py. Use these like so:

```
python deleteArchivedStories.py $CLUBHOUSE_API_TOKEN
```

The importer will add a “JIRA” label to every ticket it imports. If something gets messed up and you need to redo an import, this makes it easy to select all imported tickets, archive them and use the above command to delete them.

Transitioning

If you’ve got a team of more than a few people, you’ll probably want to schedule a day to do the import and transition, so that they can stop updating JIRA tickets while you verify the import. Then you can do a little team meeting and show off your slick new interface!

Make sure to read the README for full details and create a Github issue if you run into any problems!

Lastly:

“The only thing that other project management tool did better than Clubhouse was suck and die”—Leela

Related Posts:

- Why I Finally Ditched JIRA
- Importing GitHub issues into JIRA OnDemand

from → Commentary, Programming, Tutorials

Free Experi

3 Responses

1. Nick S permalink
   September 19, 2016

   Awesome walkthrough, super helpful!

   I submitted a request to Clubhouse support to create an automated tool that does these steps for you. Should significantly reduce onboarding friction if they decide to implement it.

2.
   Jeffrey Fermin permalink
   March 28, 2017

   Was doing some research and stumbled across this post.

   Just wanted to thank you for the kind words!

   – Jeffrey (from Clubhouse)

Trackbacks and Pingbacks

1. Why I Finally Ditched JIRA | Limina.Log

Comments are closed.

- ## Recent Articles
  - How to Draw Anything With SVG Paths in D3
  - Extracting text from the Kindle Cloud Reader
  - How to push your new GitHub repo when there's already a README
  - Visualizing a quantum wavefunction with macOS's Grapher.app
  - How to make a 'tier' or bracket formula in Google Sheets

- ## Google Ads

  Get great deals on Surface

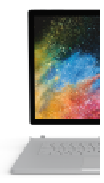
  Surface Book 2 for Business - 15" Display / 512GB / Intel Core i7

  ~~$2,899.00~~

  $2,399.00 NOW 17% OFF

  SHOP NOW >

  Free shipping

- ## Tedb0t Events

  Live near NYC? Like technological & experimental art & music? Subscribe to the **Tedb0t NYC Events** mailing list! Subscribe

- ## Tags

architecture arduino art AS3 Bash bluetooth C++ Code development electronics essay Fiction Flash graffiti Hacking Hardware Installation Art ITP JIRA Language Linux metaforms Music Networked Objects Networking Node.js Performance Physical Computing poetry Processing Programming Prototypes PureData Python RaspberryPi script Sound stream of consciousness Ubuntu video Visual Music web.py Windows wireless xbee

- ## Donate to Limina.Log

  If you like what you read and would like to keep the server running, consider a donation, or click a few ads! Every bit helps :)
  Donate
  Or donate Bitcoin to my address: 15f9er3dHk3ZsgU68Jr4LoN8mzDFZP5bAx

- ## Meta

  - Log in
  - Entries RSS
  - Comments RSS
  - WordPress.org

- ## Links

  - Limina.Studio | Art & Design Consulting
  - OlberBlog
  - Plastic Blog
  - The Messenger Said
  - Unwed Human Female

- ## Google Ads

  Free Experimentation eBoo
  Successfully experiment and integrate product features in yo software applications