

# Switching from Jira

Bring your projects, issues, epics, sprints, comments, time and files over from Jira Cloud or Jira Data Center, check it before anything is written, and take it back if you want to.

---

## Overview

The importer is part of Resourcecloud (**Admin → Data Import**, for RC Admins). You export your Jira data into one file, upload it, tell Resourcecloud what each Jira thing becomes, look at what *would* happen, and start. It runs in the background and reports what it did.

### 1. Export

A script reads Jira's REST API and writes one zip.

### 2. Upload

Any size; an interrupted upload carries on where it stopped.

### 3. Map

Types, statuses, projects and people, proposed from what you already have.

### 4. Check, start

A dry run says what would be created, changed and left alone.

- **Repeatable.** Every Jira object is remembered by its id. Importing the same export again changes nothing; importing a newer export updates what changed in Jira and adds what is new. Nothing is doubled.
- **Reversible.** An import can be undone: it removes what it created, and keeps what people have worked on since.
- **Quiet.** No e-mails and no notifications are sent while importing.
- **Yours.** The data goes from Jira to the computer you run the script on, and from there to your own Resourcecloud. It is not sent anywhere else.

---

## Before you start

- **An RC Admin account** in Resourcecloud.
- **Teams.** Each Jira project needs a team in Resourcecloud (who sees and works on its tasks) and a solution (what the work belongs to). Existing ones are matched by name; otherwise you pick them, or let the importer create them (see below).
- **People.** Jira people are matched to Resourcecloud users by e-mail address first, then by user name. Create the accounts you want to keep first, or allow the importer to create the missing ones as *inactive* accounts.
- **Stages.** If the team that gets the work already has a board with stages, name them as your Jira statuses (or choose in the mapping which Jira status goes to which stage).
- **Jira access.** An account that can browse the projects. Jira Cloud: your e-mail address and an API token. Data Center: a personal access token (or user name and password). For the list of all people the account also needs permission to browse users; without it people are taken from the issues.
- **Python 3.9 or newer** on the computer that runs the export script. No other software.
- **Disk space** for the export (attachments are most of it) and for the upload on the Resourcecloud server.

---

## Jira and Resourcecloud words

| In Jira                       | In Resourcecloud | Note                                                                                                                |
|-------------------------------|------------------|---------------------------------------------------------------------------------------------------------------------|
| Project                       | Team + solution  | The team decides who sees the tasks, the solution what the work belongs to. Chosen per project in the mapping.      |
| Issue (Task, Bug, Story, ...) | Task             | The task type comes from the mapping (Bug → Bugfix, ...). A Jira <b>Story</b> becomes a task of type <i>Story</i> . |
| Sub-task                      | Subtask          | Belongs to its task and takes that task's type, unless you choose another.                                          |

|                           |                                             |                                                                                                                                                |
|---------------------------|---------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| Epic                      | Epic                                        | In Resourcecloud an epic holds <i>stories</i> and a story holds tasks, so each epic gets a story of the same name that holds its tasks.        |
| Story (issue type)        | Task of type Story or a Resourcecloud story | The default is a task. You can map an issue type to a Resourcecloud story instead (a container for tasks) in the mapping.                      |
| Board                     | The team's Kanban board and sprint planning | Boards themselves are not copied; they decide which team a sprint belongs to.                                                                  |
| Workflow, status          | Stages of the team's board                  | Category <i>To Do</i> → Assigned (not started), <i>Done</i> → the task is closed, <i>In Progress</i> → a stage you choose (proposed: Working). |
| Sprint                    | Sprint                                      | Name, goal, dates, and whether it is closed, active or in the future. Tasks go into the sprints they were in.                                  |
| Story points              | Story points                                | Rounded to whole numbers.                                                                                                                      |
| Original estimate         | Estimated time                              | Tasks without an estimate get 10 minutes, as in the API.                                                                                       |
| Worklog                   | Time entry                                  | Booked on the person who logged it, on their day. Needs a matching user (see below).                                                           |
| Priority                  | Priority: normal or high                    | Highest, High, Blocker, Critical, Major, Urgent → high. Changeable per priority.                                                               |
| Comment                   | Comment                                     | With the original author and date.                                                                                                             |
| Attachment                | File                                        | Taken from the zip.                                                                                                                            |
| Reporter, assignee        | Created by, assigned to                     | Matched by e-mail, then user name.                                                                                                             |
| Resolved, resolution date | Done, done date                             | A task whose status is mapped to "Done" is closed on its resolution date.                                                                      |
| Description               | Description                                 | Converted to formatted text. Unsafe markup is removed.                                                                                         |

# 1. Export from Jira

Jira has no "export everything" button, so use the script that comes with Resourcecloud: `tools/jira_export.py`. It asks Jira's REST API (it only reads) and writes one zip with projects, issues, comments, worklogs, attachments, boards, sprints and people.

## 1 Jira Cloud:

```
python3 jira_export.py --url https://acme.atlassian.net \
    --user you@acme.com --token "$JIRA_TOKEN" --projects ABC,DEF
```

## 2 Jira Data Center / Server:

```
python3 jira_export.py --url https://jira.acme.com \
    --token "$JIRA_TOKEN" --projects ABC,DEF
```

(a personal access token; or `--user` and `--password`)

3 Leave `--projects` out to export every project the account can see. Put secrets in the environment ( `JIRA_USER` , `JIRA_TOKEN` , `JIRA_PASSWORD` ) so they do not end up in your shell history.

4 The result is a folder `jira-export/` and `jira-export.zip`. A `README.txt` inside says what was exported.

**Useful options.** `--no-attachments` leaves files out. `--max-attachment-mb 200` skips larger files. `--delay 0.5` pauses between requests on a busy Jira. `--cafile` trusts an internal certificate. Run it again with the same `--out` to carry on: files already downloaded are not fetched again.

The script waits when Jira asks it to slow down, completes the comment and worklog lists that Jira's search cuts short, never sends your credentials to any other host (Jira Cloud serves files from one), and stops with a clear message if the credentials or project keys are wrong.

**Your own export.** The importer reads what Jira's REST API returns: pages of search results, sprint and board pages, user lists, as `.json`, `.jsonl`, `.gz` or a `.zip` of these. If you already have such files, upload them as they are. Attachments go into the zip as `attachments/<id>/<file name>`; the importer finds them by Jira's attachment id.

## 2. Upload

- 1 Open **Admin → Data Import** and drop the zip on the box (or click it to choose the file).
- 2 The file is sent in pieces of 4 MB with a progress bar. Size does not matter. If the connection drops, the page tries again by itself; if you close the page, open the import from the list, choose **the same file** and it carries on where the server stopped.
- 3 When the last piece is in, Resourcecloud reads the file in the background ("Reading the export...") and the mapping opens.

If Resourcecloud sits behind a reverse proxy that limits request bodies (some default to 1 MB), pieces of 4 MB are refused. Raise the limit for `/backend/import_upload` to at least 8 MB.

## 3. Map

The mapping screen lists everything that needs a decision, with how often it occurs. Everything is already proposed from what exists in Resourcecloud; read it, change what is wrong, and press **Check**. **Back to the proposal** undoes your edits.

| Tab                | You decide                                                                                                                                                                                   |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Issue types</b> | What each Jira issue type becomes: task, subtask, Resourcecloud story or epic, and the task type. Types that do not exist (such as <i>Story</i> ) are created if you allow it under Options. |

|                   |                                                                                                                                                                                                            |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Statuses</b>   | The stage for each Jira status: <i>Assigned</i> (not started), <i>Done</i> (closed) or a stage of the team. A stage the team does not have: tasks go to Assigned, unless you allow adding stages.          |
| <b>Priorities</b> | Normal or high.                                                                                                                                                                                            |
| <b>Projects</b>   | Team, solution and (for a new solution) company per Jira project. Matched by name when possible.                                                                                                           |
| <b>People</b>     | Which Resourcecloud user each Jira person is. Filter, show only the people nobody matched, search for a user, choose <i>Me</i> (comments are attributed to you), <i>Nobody</i> , or <i>Create a user</i> . |
| <b>Options</b>    | What may be created, and what to import (comments, worklogs, attachments, sprints).                                                                                                                        |

## People nobody matches

- Their **comments** are kept. They go under the importing admin, and each says "Comment by <name> in Jira (<date>)".
- Their **worklogs are not imported**. Time is booked on a person; booking it on whoever runs the import would put hours on their timesheet that they never worked. Map such people to a user, or allow creating users, and import again.
- Tasks assigned to them stay **unassigned**.

## Creating users and teams

Off by default. An RC Admin can allow it with the switch on the overview page of the importer; the importer then offers *Create a user* and *Create a virtual team*. Created users are **inactive**, cannot sign in until you activate them, and do not count as licensed seats meanwhile. Created teams are virtual teams that get the stages their issues need and the people who work in them as members. The switch is checked again when the import runs.

---

## 4. Check and start

- 1 **Check** saves the mapping and runs a dry run. It writes nothing. It shows, per kind, how many objects would be *new*, *changed* or *unchanged*, what would be created besides the work (task types, solutions, teams, users, stages), and what needs a look: people nobody matches, stages a team does not have, parents or epics that are not in the export.
- 2 Errors (a project without a team, say) are shown per row and must be fixed. Warnings can be accepted.
- 3 **Start import** asks once more. The import runs in the background; you can close the page.

---

## 5. Report and result

While it runs you see a progress bar and a log. Afterwards the report lists how many objects were created, updated, unchanged, skipped and failed per kind, each failed item with its reason, and the log (warnings such as people nobody matched, files that were not in the zip). One failed item does not stop the others.

Then look at the result where your people work: the backlog and Kanban board of each team (open work lands in the backlog, or in the stage you mapped), the sprint planning, the epics and stories, a task's comments and time, its files. **Clean up** removes the uploaded file and the staged data from the server; the report and the record of what was imported stay.

**Speed.** On a development machine the importer wrote about 1,000 objects (tasks, comments, worklogs) per second, so 50,000 issues with 150,000 comments take a few minutes. Reading the file is faster still. Exporting from Jira is the slow part, and depends on Jira.

---

## Importing again

You can import the same project as often as you like: the first time to look, again for a final run just before you switch.

- An object Jira did not change since last time is **left alone**. A new one is created. A changed one is **updated in place**.
- An update writes Jira's version over what is in Resourcecloud, including edits made here in the meantime (title, description, assignee, stage, dates). The dry run warns you with the number of objects affected. If people already work in Resourcecloud, do the final import before they start, or accept that those tasks are overwritten.
- Comments, worklogs and files that are new in Jira are added; ones that exist are not doubled.

---

## Undoing an import

On the report of a finished import, **Undo this import** shows a preview and, when you confirm, runs in the background.

- It removes what **that import created**: tasks with their comments, time entries, files and links, stories, epics, sprints, and the users, teams, solutions, task types and stages it made — the last ones only when nothing uses them any more.
- It does **not** take back updates the import made to things an earlier import created.
- A task people have **worked on since** (comments, files or chat added in Resourcecloud) is **kept**, whole, and the report says why. Tick *Also remove tasks that have been worked on since* to remove those too.
- A task that something else depends on — an invoice, leave, a lead, a deployment, a meeting, a forked task or a subtask from elsewhere — is **never removed**.
- Kept things keep their record, so importing the export again updates them instead of making a second copy.
- It is refused while another import is running, and while a later import has updated what this one created (the message names it; undo that one first).

---

## What is not imported

- Labels and components as fields: Resourcecloud tasks have none, so they are written as a line at the end of the description (searchable, but no filter).

- Fix versions on anything but epics: an epic is put into a release of the first fix version's name (a further version is reported, not taken over); tasks are not. Release dates are not imported.
- Issue links other than "Blocks" (relates to, duplicates, ...) and watchers, votes. A "Blocks" link between two imported tasks becomes a dependency: the blocked task waits for the other. A link removed in Jira later is not removed here by an update.
- Custom fields other than story points, the epic link and sprints.
- The change history of issues, Jira workflows and their rules, screens, permission schemes, notification schemes, automations.
- Dashboards, filters, saved views, and Marketplace apps' data.
- Jira's issue security levels: imported tasks are visible to their team like any other task.
- Comments and worklogs written on an *epic* (an epic has no comments in Resourcecloud); the report lists them.
- Attachments that are not in the zip (the issue is imported; the report names the missing file).

Recreate the rules you need on the Resourcecloud side: stages and working model of each team, roles, notifications.

---

## Cut-over plan

- 1 **Rehearse.** Export one project, import it into a test team, look around, undo it. Fix the mapping until it is what you want.
- 2 **Import everything** a week or two before the switch. Let the team members look at their tasks; fix names and stages that surprise them.
- 3 **Freeze Jira** on the day: make it read-only, or ask everyone to stop. Export again and import again; only what changed is touched.
- 4 **Switch.** Everyone works in Resourcecloud from now on. Keep Jira read-only for as long as your records require; the export zip is a complete copy of what was imported.
- 5 Run **Clean up** on the imports when you are done, so the uploads are not kept on the server.

---

## Questions

► Why is a Jira Story a task and not a Resourcecloud story?

► Some people were not matched. What now?

► Worklogs are missing.

► An attachment was reported as missing.

► Emoji are gone from titles.

► A title lost text between < and >.

► Can I import into a team that already has a board?

► Can I import two Jira installations?

► Does the import send mail?

► The upload stops at the same point every time.

---

## Security

- Only an RC Admin can use the importer; every action is checked on the server, and every change needs the usual security token.
- The uploaded file is kept in a folder the web server does not serve, and is removed by **Clean up**.

- Whatever an export contains is treated as untrusted: markup and script are removed from titles, names and formatted text; file names are cleaned; an attachment that claims to be gigabytes is refused before it is read.
- Time is booked, and comments are attributed, only to people who exist in Resourcecloud. Every imported object can be traced to the import, the admin and the time.