AQUILA - Atlassian Jira Integration through Oauth 2.0(3LO) and API Key

What are API Token Scopes?

Scopes define what actions an API token is allowed to perform in Atlassian apps such as Jira and Confluence. They enhance security by limiting permissions to only what's needed (e.g., read-only access to audit logs). Always use scoped tokens for AQUILA integrations—unscoped tokens are deprecated for most apps and may not support fine-grained access. For audit logs (events like user actions, config changes, or security incidents), use the specific scopes listed below. Broader scopes may be needed for other integrations (e.g., content indexing) but stick to these for basic monitoring to minimize risk.

Prerequisites

1. Atlassian Admin Email Account
2. Atlest have a read rights linux privilege
3. Broswer (firefox,chrome)

Creating an API Token with Scopes

1. Log in to https://id.atlassian.com/manage-profile/security/api-tokens.
2. Select "Create API token with scopes".
3. Enter a descriptive name for the token (e.g., "AQUILA- Audit Logs Monitoring").

Choose an expiration date for the token (between 1 and 365 days; consider shorter for security).

1. Select the application Jira.
2. Select the scopes or permissions the token should have:
   - For Jira (audit logs): **Classic**: `manage:jira-configuration` or **Granular**: `read:audit-log:jira`, `read:user:jira` to access /rest/api/3/auditing/record.
3. Click "Create".
4. Copy the token and save it securely. You cannot view it again after this step. If lost, generate a new one. Share only with trusted integrations like AQUILA—revoke if compromised.

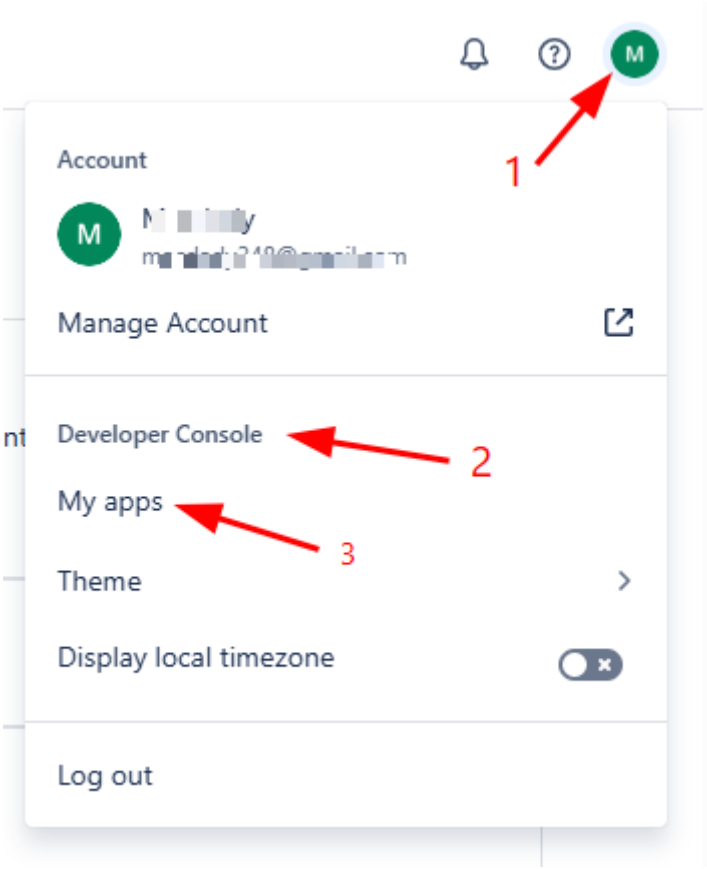
OAuth 2.0(3LO)apps

OAuth 2.0 (3LO) (also known as "three-legged OAuth" or "authorization code grants") apps. OAuth 2.0 (3LO) allows external applications

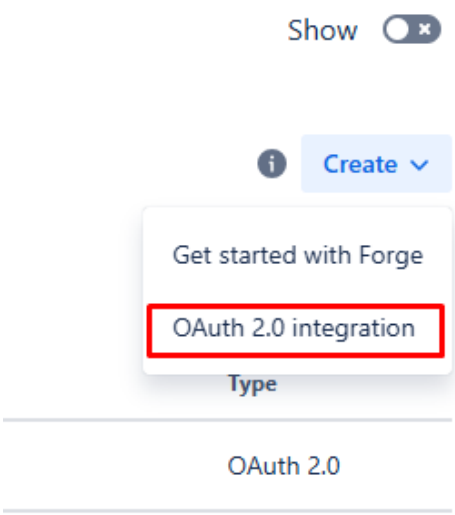
and services to access Atlassian product APIs on a user's behalf. OAuth 2.0 (3LO) apps are created and managed in the [developer console].

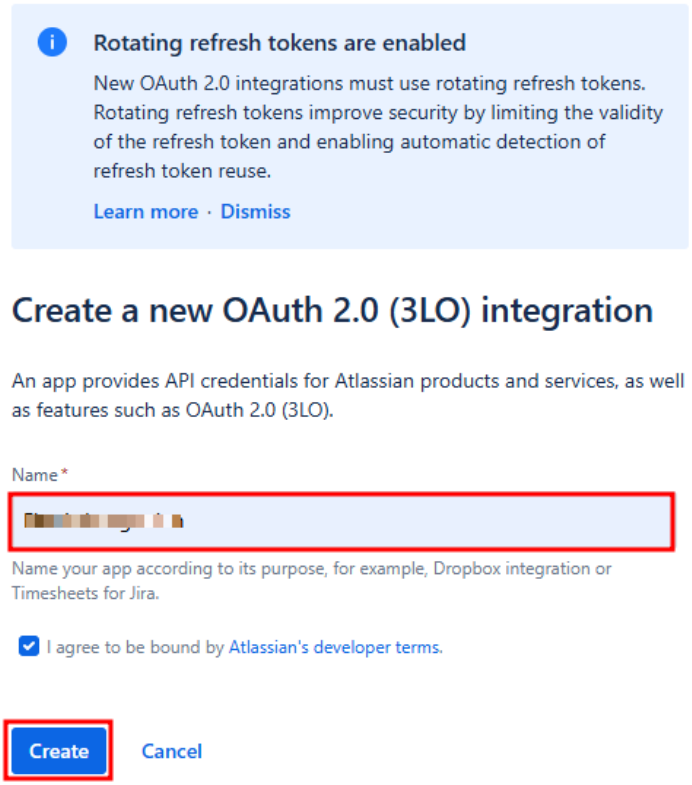
Enabling OAuth 2.0(3LO)

1. Select your profile icon in the top-right corner, and from the dropdown, select Developer console.



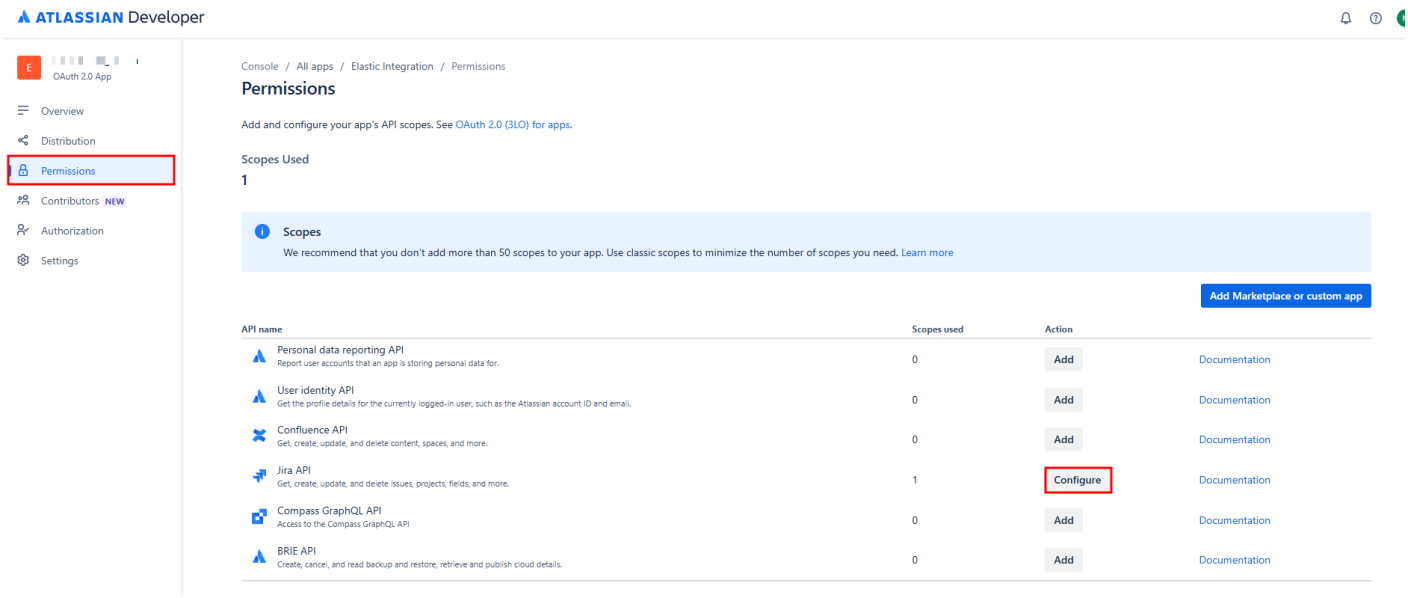
2. Select your app from the list (or create one if you don't already have one).



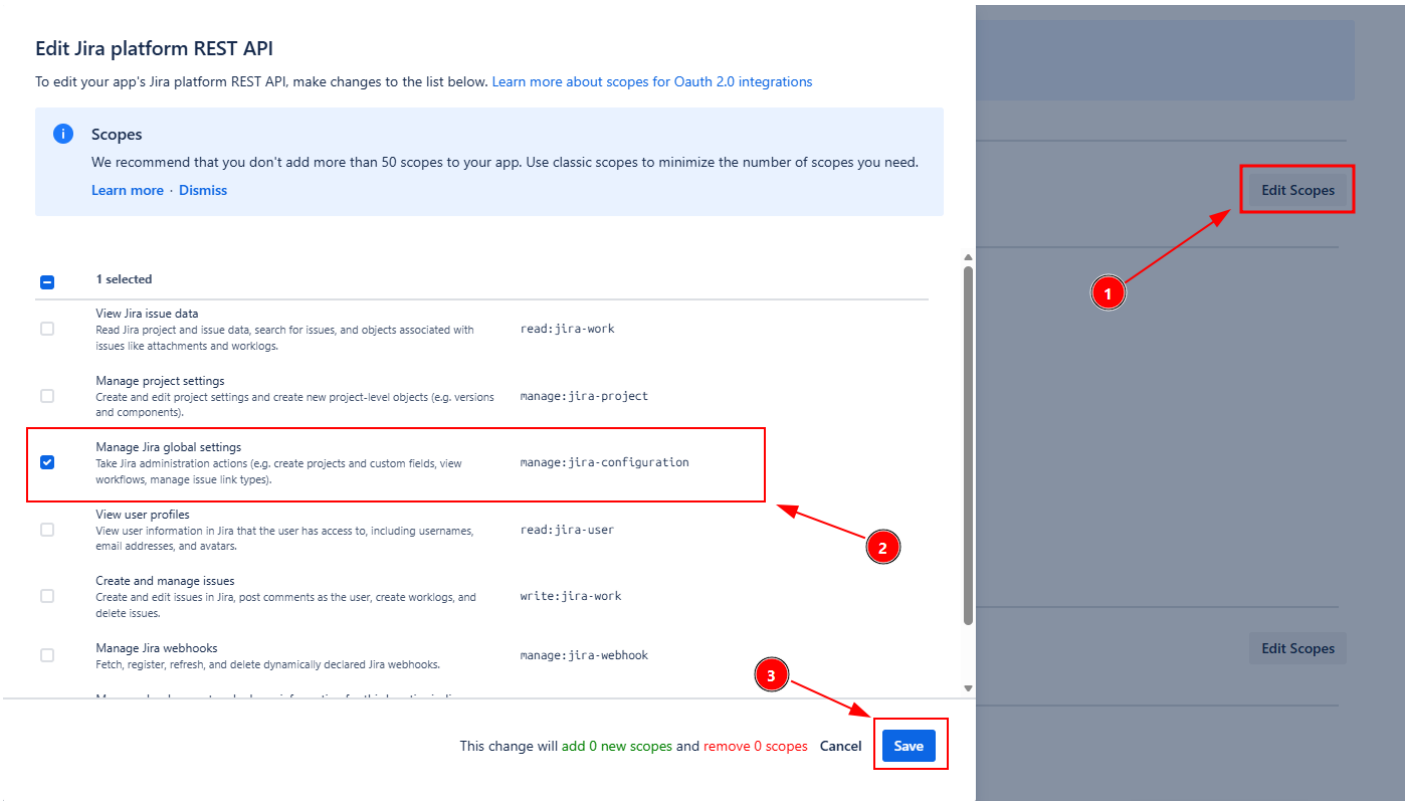


3. Select Permissions in the left menu.

4. Next to the API you want to add, select Configure or Add.

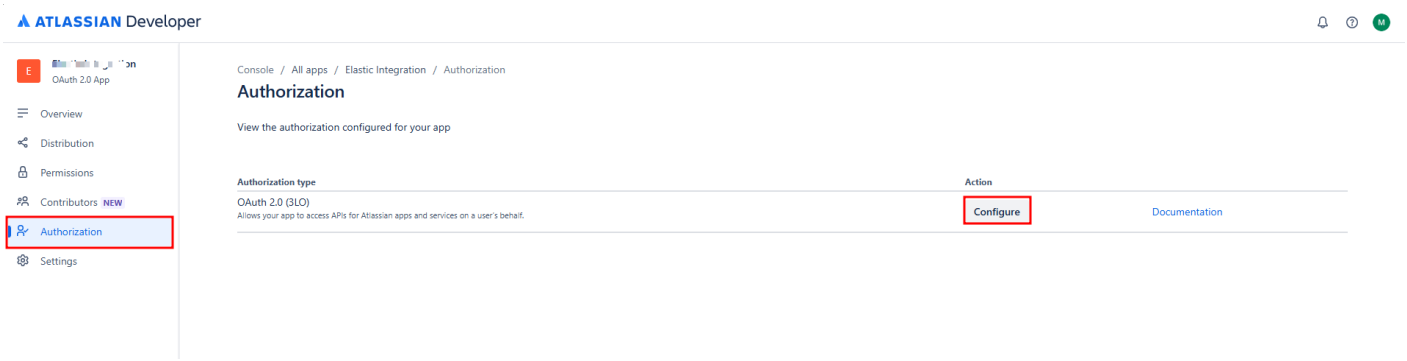


Select Classic or Granular scope Classic:manage:jira-configuration was selected in this example.



5. Select Authorization in the left menu.

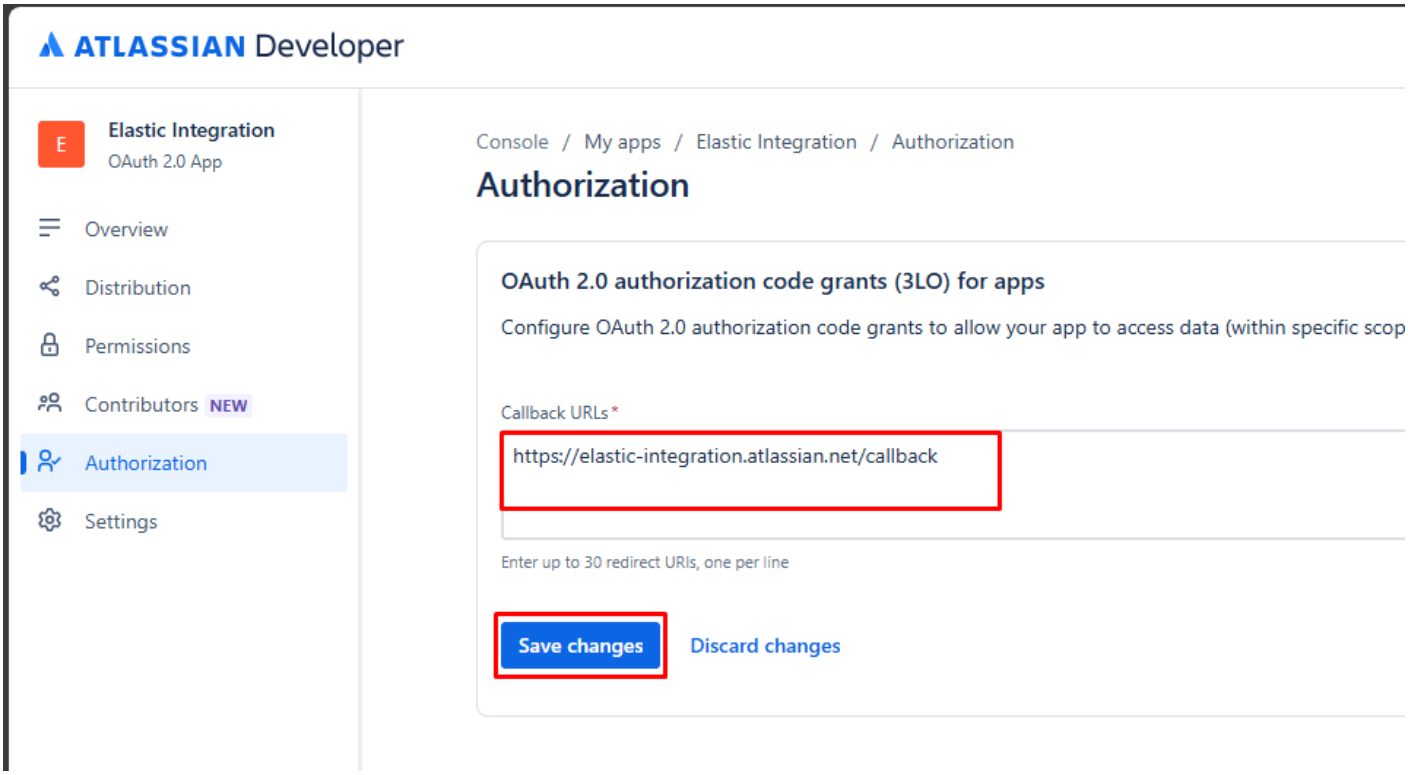
6. Next to OAuth 2.0 (3LO), select Configure or Add.



7. Enter the Callback URL. Set this to any URL that is accessible by the app. When you implement OAuth 2.0 (3LO) in your app (see next section),

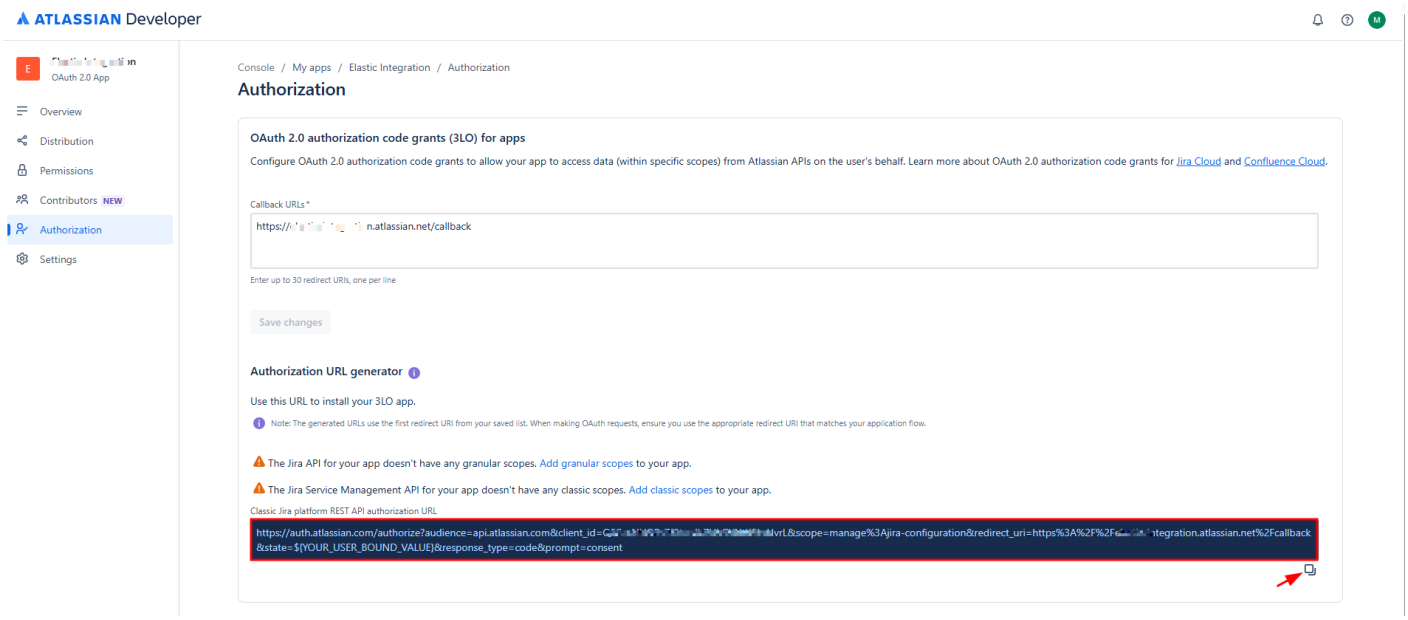
> **The redirect_uri must match this URL.**

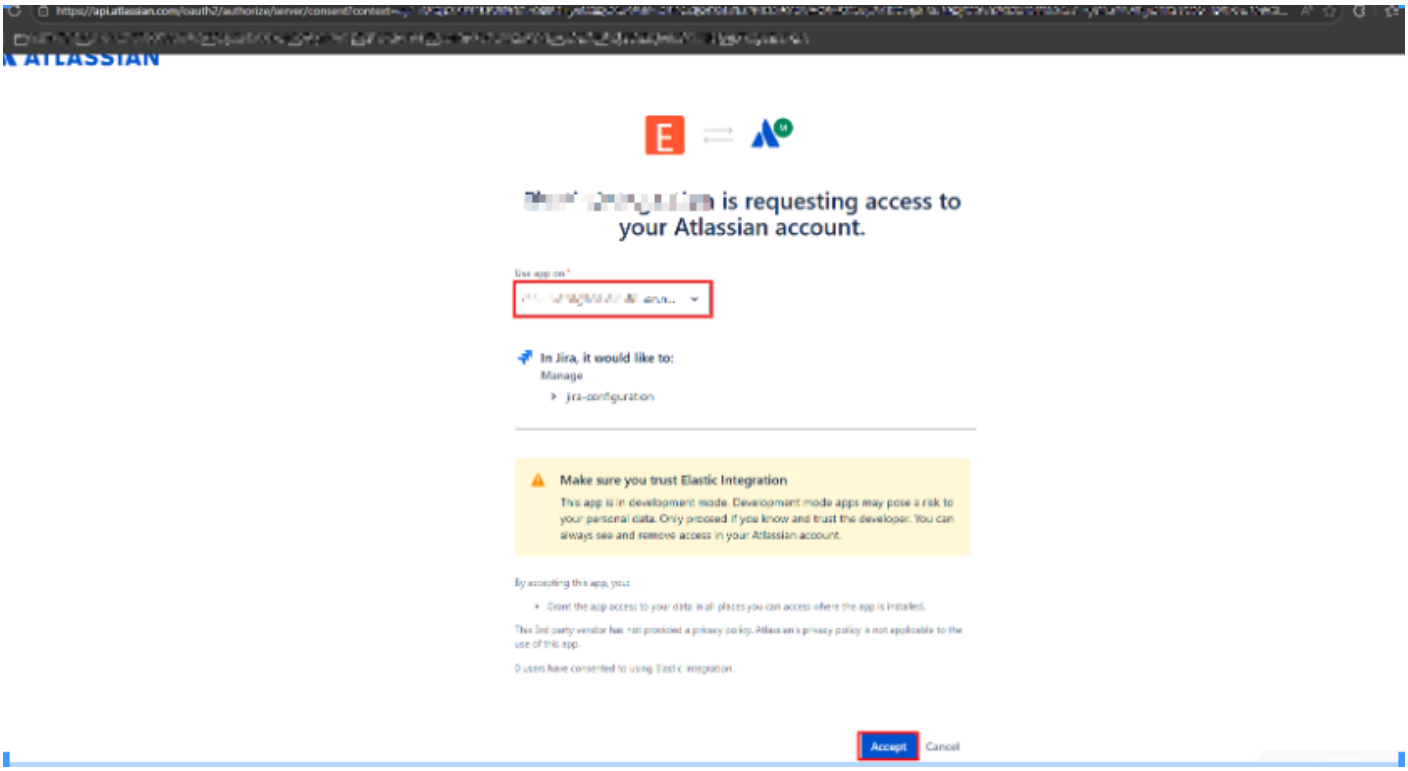
```
https://<client_name>.atlassian.net/callback
```



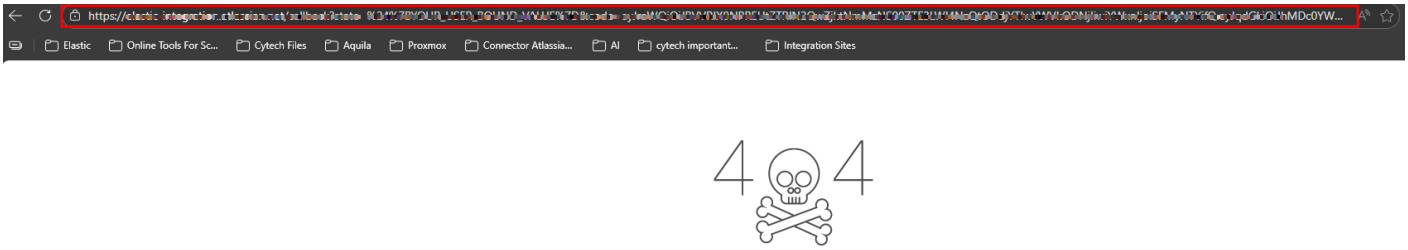
8. Click Save changes.

9. In **Authorization URL generator** Copy and Paste in the browser.

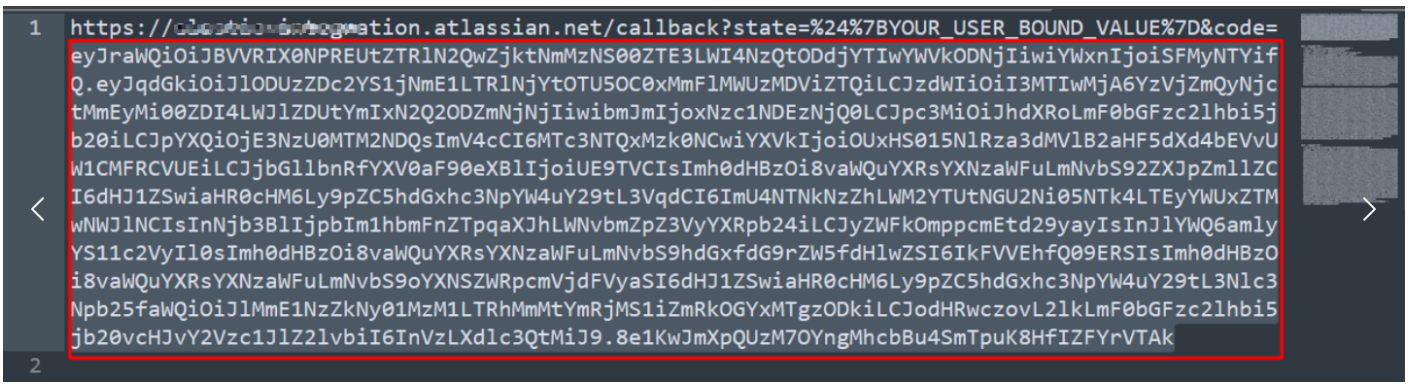




10. Copy the URL and paste in text editor (notepad).



Example of a copied URL (the boxed section contains the authorization code).



Exchange authorization code for access token

Paste this Curl command into terminal.

```
curl --request POST \
  --url 'https://auth.atlassian.com/oauth/token' \
  --header 'Content-Type: application/json' \
  --data '{"grant_type": "authorization_code","client_id": "YOUR_CLIENT_ID","client_secret":
"YOUR_CLIENT_SECRET","code": "YOUR_AUTHORIZATION_CODE","redirect_uri":
"https://YOUR_APP_CALLBACK_URL"}'
```

Change all fields:

- `client_id`: (*required*) Set this to the **Client ID** for your app. Find this in **Settings** for your app in the developer console.
- `client_secret`: (*required*) Set this to the **Secret** for your app. Find this in **Settings** for your app in the developer console.
- `code`: (*required*) Set this to the authorization code received from the initial authorize call (described above).
- `redirect_uri`: (*required*) Set this to the callback URL configured for your app in the developer console.

If successful, this call returns an access token similar to this:

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
  "access_token": <string>,
  "expires_in": <expiry time of access_token in second>,
  "scope": <string>
}
```

Make calls to the API using the access token Get the `cloudid` for your site

Your app now has an access token that it can use to authorize requests to the APIs for the Atlassian site. To make requests, do the following:

1. **Get the `cloudid` for your site.**
2. Construct the request URL using the `cloudid`.
3. Call the API, using the access token and request URL.

Get the `cloud_id` for your site

Make a GET request to https://api.atlassian.com/oauth/token/accessible-resources passing the access token as a bearer token in the header of the request. For example:

> **Replace "ACCESS_TOKEN" with your newly generated token.**

```
curl --request GET \
  --url https://api.atlassian.com/oauth/token/accessible-resources \
  --header 'Authorization: Bearer ACCESS_TOKEN' \
  --header 'Accept: application/json'
```

This will retrieve the sites that have scopes granted by the token (see Check site access for the app below for details). Find your site in the response and copy the `id`. This is the `cloud_id` for your site.

> **sample output: a Jira site:**

```
[
  {
    "id": "1324a887-45db-1bf4-1e99-ef0ff456d421",
    "name": "Site name",
    "url": "https://your-domain.atlassian.net",
```

```
    "scopes": [
      "write:jira-work",
      "read:jira-user",
      "manage:jira-configuration"
    ],
    "avatarUrl": "https:\/\/site-admin-avatar-cdn.prod.public.atl-
paas.net\/avatars\/240\/flag.png"
  }
]
```

Creating the Bash Wrapper Script:

> **The wrapper script manages event collection, logging, and ensures only one instance runs at a time.**

1. **Create required directories for data, logs:**

> **Create a directory named `jira` (for this example):**

```
mkdir jira
```

2. Create the wrapper script file:

```
sudo nano /usr/local/bin/jira_audit.sh
```

3. Add the following content to the file:

> **Replace the file path `FLAT_FILE` , `CHECKPOINT_FILE` as well as `CLOUD_ID`, `USER_EMAIL`, and `API_KEY`, with their actual values.**

```
Ask Cytech Support for the Source Code
```

4. **Make the script executable:**

```
sudo chmod +x /usr/local/bin/jira_audit.sh
```

5. **Set ownership to the dedicated user**:

> **Replace the "user:group" in this example both `testing-jira:testing-jira`.**

```
sudo chown testing-jira:testing-jira /usr/local/bin/jira_audit.sh
```

Creating the Systemd Service File:

> **The systemd service ensures the event collection runs continuously and restarts automatically on failure.**

1. Create the service file:

```
sudo nano /etc/systemd/system/jira-audit.service
```

2. Add the following content to the file:

> **Replace WorkingDirectory file path.**

```
Ask Cytech Support for The Source Code
```

Create a Systemd Timer to handle looping and run automatically in the background:

A systemd timer is a feature of systemd used to schedule tasks to run automatically at specific times or intervals.

1. Create the timer file:

```
sudo nano /etc/systemd/system/jira-audit.timer
```

2. Add the following content to the file:

```
[Unit]
Description=Run Atlassian Jira Audit every 10 minutes
```

```
[Timer]
OnBootSec=60
OnUnitActiveSec=600
Persistent=true
Unit=jira-audit.service

[Install]
WantedBy=timers.target
```

Configuring Log Rotation

To manage log file sizes and prevent disk space issues, configure log rotation for Jira Audit Logs.

Step 1: Create a logrotate configuration file:

```
sudo nano /etc/logrotate.d/jira_audit
```

Step 2: Add the following configuration to the file:

> **Replace `/home/your_user/` with your actual path, and update `user_owner` and `user_group` to match the correct user and group.**

```
/home/your_user/jira/flattened.json {
    daily
    rotate 7
    missingok
    notifempty

    copytruncate

    compress
    compressoptions -1
    delaycompress

    dateext
    dateformat -%Y%m%d-%H%M%S
```

```
    create 0640 user_owner user_group
}
```

Enabling and Starting the Service

Step 1: Reload systemd to recognize the new service file:

```
sudo systemctl daemon-reload
```

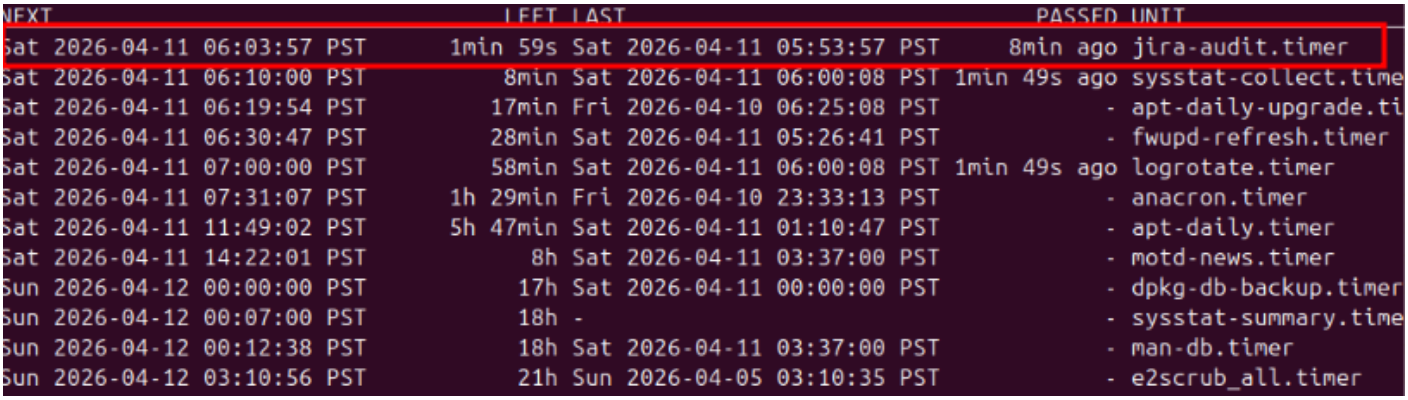
Step 2: Enable the service to start automatically on boot:

```
sudo systemctl enable --now jira-audit.timer
```

Step 3: Verify the service is running:

```
sudo systemctl list-timers
```

> **You should see something like this.**



```
NEXT                            LEFT LAST                            PASSED UNIT
Sat 2026-04-11 06:03:57 PST      1min 59s Sat 2026-04-11 05:53:57 PST     8min ago jira-audit.timer
Sat 2026-04-11 06:10:00 PST         8min Sat 2026-04-11 06:00:08 PST 1min 49s ago sysstat-collect.time
Sat 2026-04-11 06:19:54 PST        17min Fri 2026-04-10 06:25:08 PST            - apt-daily-upgrade.ti
Sat 2026-04-11 06:30:47 PST        28min Sat 2026-04-11 05:26:41 PST            - fwupd-refresh.timer
Sat 2026-04-11 07:00:00 PST        58min Sat 2026-04-11 06:00:08 PST 1min 49s ago logrotate.timer
Sat 2026-04-11 07:31:07 PST      1h 29min Fri 2026-04-10 23:33:13 PST            - anacron.timer
Sat 2026-04-11 11:49:02 PST      5h 47min Sat 2026-04-11 01:10:47 PST            - apt-daily.timer
Sat 2026-04-11 14:22:01 PST            8h Sat 2026-04-11 03:37:00 PST            - motd-news.timer
Sun 2026-04-12 00:00:00 PST           17h Sat 2026-04-11 00:00:00 PST            - dpkg-db-backup.timer
Sun 2026-04-12 00:07:00 PST           18h -                                      - sysstat-summary.time
Sun 2026-04-12 00:12:38 PST           18h Sat 2026-04-11 03:37:00 PST            - man-db.timer
Sun 2026-04-12 03:10:56 PST           21h Sun 2026-04-05 03:10:35 PST            - e2scrub_all.timer
```

Monitoring Live Logs:

To view real-time logs from the jira-audit service:

```
journalctl -u jira-audit -f
```

> **Please provide the following information to CyTech.**

- **Flattened.json file path example "/home/testing-jira/jira/flattened.json"**

Revision #13
Created 6 April 2026 16:33:09 by Benjie Janlay Jr.
Updated 28 May 2026 10:23:21 by Benjie Janlay Jr.