+++ title = “GitLab OAuth2 Authentication” description = “Grafana OAuthentication Guide” keywords = [“grafana”, “configuration”, “documentation”, “oauth”] weight = 900 +++

GitLab OAuth2 Authentication

To enable GitLab OAuth2 you must register the application in GitLab. GitLab will generate a client ID and secret key for you to use.

Create GitLab OAuth keys

You need to create a GitLab OAuth application. Choose a descriptive *Name*, and use the following *Redirect URI*:

```
https://grafana.example.com/login/gitlab
```

where `https://grafana.example.com` is the URL you use to connect to Grafana. Adjust it as needed if you don't use HTTPS or if you use a different port; for instance, if you access Grafana at `http://203.0.113.31:3000`, you should use

```
http://203.0.113.31:3000/login/gitlab
```

Finally, select *read_api_as the_Scope_and submit the form. Note that if you're not going to use GitLab groups for authorization (i.e. not setting* ***`allowed_groups`****, see below), you can select_read_user* instead of *read_api_as the_Scope*, thus giving a more restricted access to your GitLab API.

You'll get an *Application Id* and a *Secret* in return; we'll call them `GITLAB_APPLICATION_ID` and `GITLAB_SECRET` respectively for the rest of this section.

Enable GitLab in Grafana

Add the following to your Grafana configuration file to enable GitLab authentication:

```
[auth.gitlab]
enabled = true
allow_sign_up = false
client_id = GITLAB_APPLICATION_ID
client_secret = GITLAB_SECRET
scopes = read_api
auth_url = https://gitlab.com/oauth/authorize
token_url = https://gitlab.com/oauth/token
api_url = https://gitlab.com/api/v4
allowed_groups =
```

You may have to set the `root_url` option of `[server]` for the callback URL to be correct. For example in case you are serving Grafana behind a proxy.

Restart the Grafana backend for your changes to take effect.

If you use your own instance of GitLab instead of `gitlab.com`, adjust `auth_url`, `token_url` and `api_url` accordingly by replacing the `gitlab.com` hostname with your own.

With `allow_sign_up` set to `false`, only existing users will be able to login using their GitLab account, but with `allow_sign_up` set to `true`, *any* user who can authenticate on GitLab will be able to login on your Grafana instance; if you use the public `gitlab.com`, it means anyone in the world would be able to login on your Grafana instance.

You can limit access to only members of a given group or list of groups by setting the `allowed_groups` option.

allowed_groups

To limit access to authenticated users that are members of one or more GitLab groups, set `allowed_groups` to a comma- or space-separated list of groups. For instance, if you want to only give access to members of the `example` group, set

```
allowed_groups = example
```

If you want to also give access to members of the subgroup `bar`, which is in the group `foo`, set

```
allowed_groups = example, foo/bar
```

Note that in GitLab, the group or subgroup name doesn't always match its display name, especially if the display name contains spaces or special characters. Make sure you always use the group or subgroup name as it appears in the URL of the group or subgroup.

Here's a complete example with `allow_sign_up` enabled, with access limited to the `example` and `foo/bar` groups. The example also promotes all GitLab Admins to Grafana Admins:

```
[auth.gitlab]
enabled = true
allow_sign_up = true
client_id = GITLAB_APPLICATION_ID
client_secret = GITLAB_SECRET
scopes = read_api
auth_url = https://gitlab.com/oauth/authorize
token_url = https://gitlab.com/oauth/token
api_url = https://gitlab.com/api/v4
allowed_groups = example, foo/bar
role_attribute_path = is_admin && 'Admin' || 'Viewer'
```

Map roles

You can use GitLab OAuth to map roles. During mapping, Grafana checks for the presence of a role using the JMESPath specified via the `role_attribute_path` configuration option.

For the path lookup, Grafana uses JSON obtained from querying GitLab's API `/api/v4/user` endpoint. The result of evaluating the `role_attribute_path` JMESPath expression must be a valid Grafana role, for example, `Viewer`, `Editor` or `Admin`. For more information about roles and permissions in Grafana, refer to [About users and permissions]({{< relref "../administration/manage-users-and-permissions/about-users-and-permissions.md" >}}).

An example Query could look like the following:

```
role_attribute_path = is_admin && 'Admin' || 'Viewer'
```

This allows every GitLab Admin to be an Admin in Grafana.

Team Sync (Enterprise only)

> Only available in Grafana Enterprise v6.4+

With Team Sync you can map your GitLab groups to teams in Grafana so that your users will automatically be added to the correct teams.

Your GitLab groups can be referenced in the same way as `allowed_groups`, like `example` or `foo/bar`.

[Learn more about Team Sync]({{< relref "team-sync.md" >}})