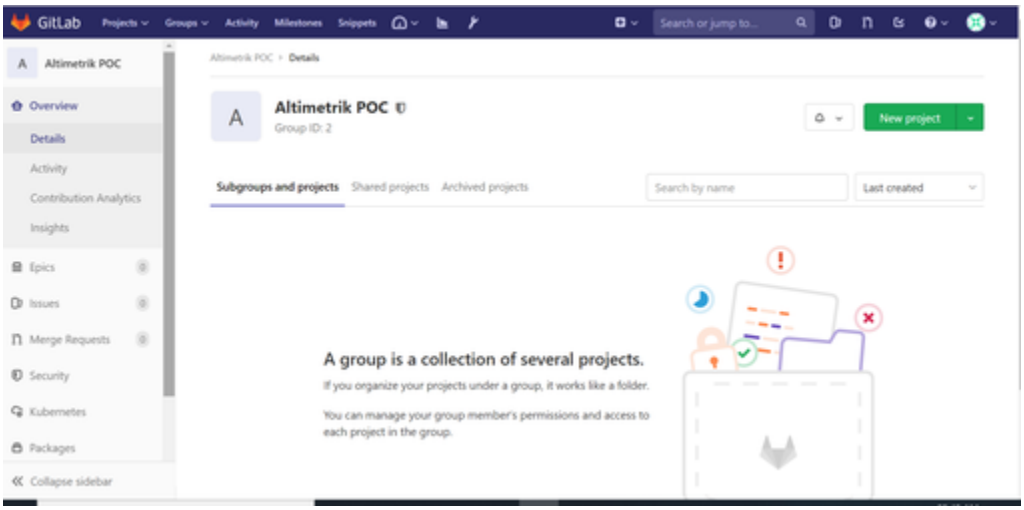
GitLab - Group Definition

Groups:-

With GitLab Groups, you can:

- Assemble related projects together.
- Grant members access to several projects at once.

Find your groups by clicking **Groups > Your Groups** in the top navigation.



The **Groups** page displays:

- All groups you are a member of, when **Your groups** is selected.
- A list of public groups, when **Explore public groups** is selected.

Each group on the **Groups** page is listed with:

- How many subgroups it has.
- How many projects it contains.
- How many members the group has, not including members inherited from parent groups.
- The group's visibility.
- A link to the group's settings, if you have sufficient permissions.
- A link to leave the group, if you are a member.

Namespaces

In GitLab, a namespace is a unique name to be used as a user name, a group name, or a subgroup name.

http://gitlab.example.com/username
http://gitlab.example.com/groupname
http://gitlab.example.com/groupname/subgroup_name

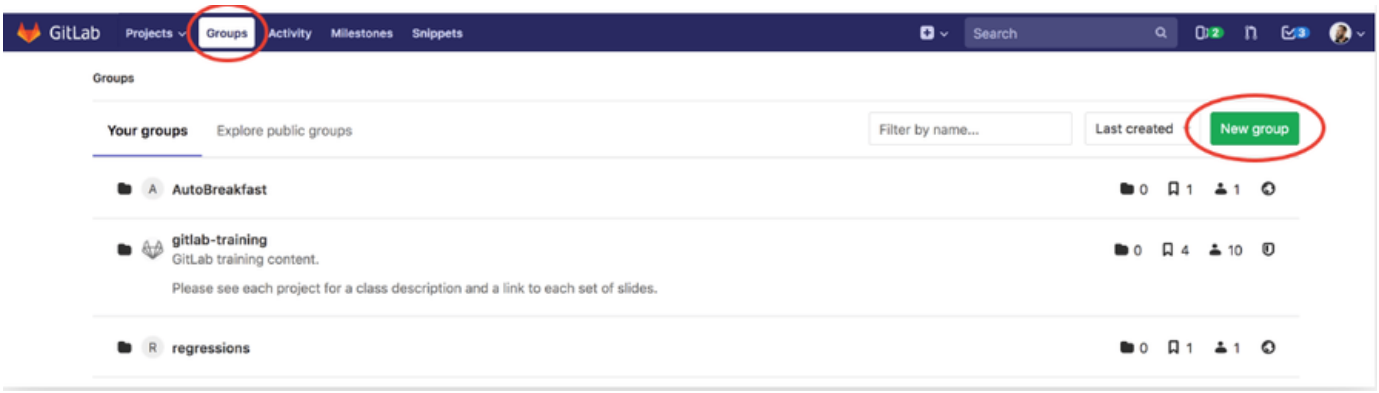
For example, consider a user named Alex:

Alex creates an account on GitLab.com with the username alex; their profile will be accessed under https://gitlab.example.com/alex
Alex creates a group for their team with the group name alex-team; the group and its projects will be accessed under https://gitlab.example.com/alex-team
Alex creates a subgroup of alex-team with the subgroup name marketing; this subgroup and its projects will be accessed under https://gitlab.example.com/alex-team/marketing

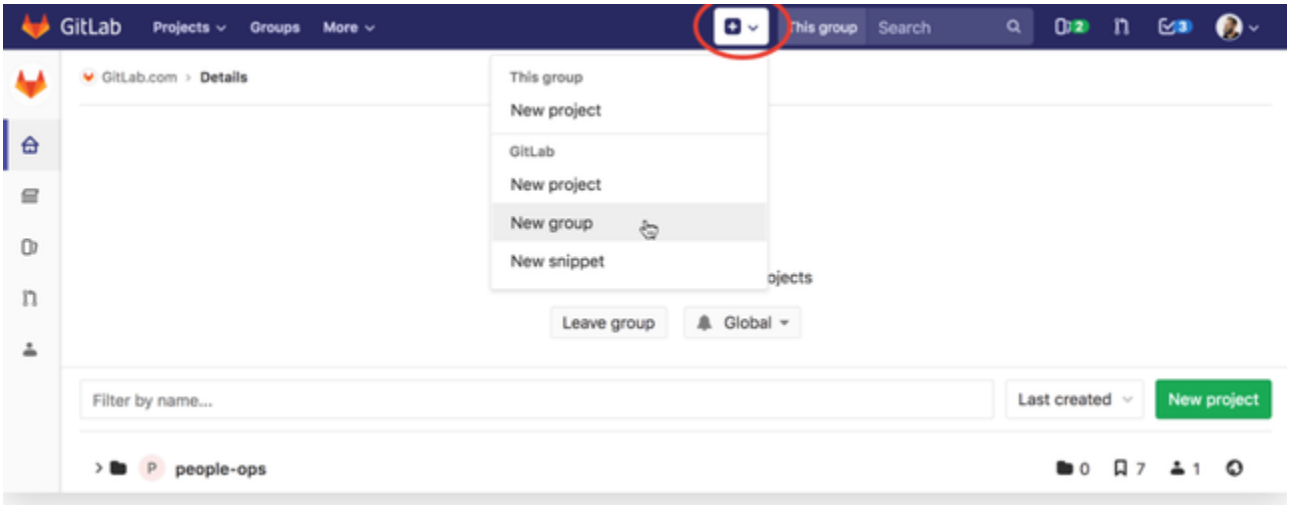
Create a new group

To create a new Group, either:

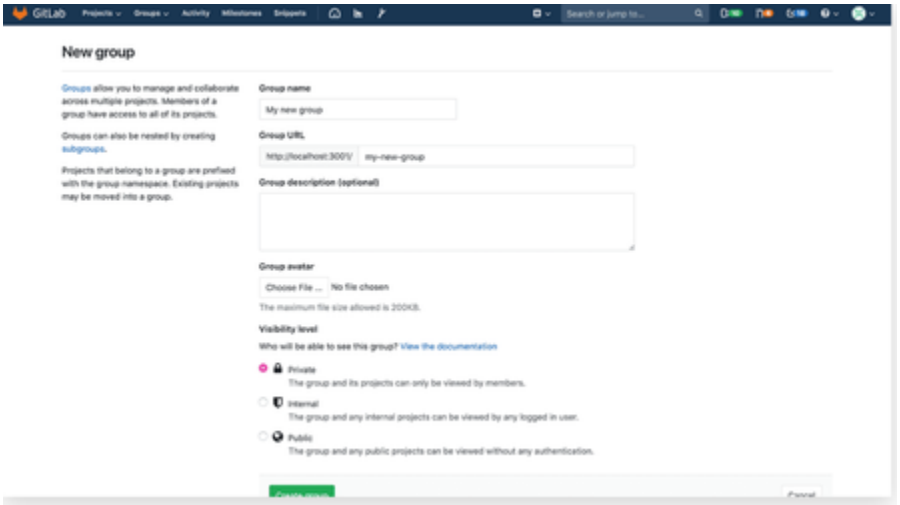
In the top menu, click Groups and then Your Groups, and click the green button New group.



Or, in the top menu, expand the `plus` sign and choose **New group**.



Add the following information:



1. The **Group name** will automatically populate the URL. Optionally, you can change it. This is the name that displays in group views. The name can contain only:
    - Alphanumeric characters
    - Underscores
    - Dashes and dots
    - Spaces
2. The **Group URL** is the namespace under which your projects will be hosted. The URL can contain only:

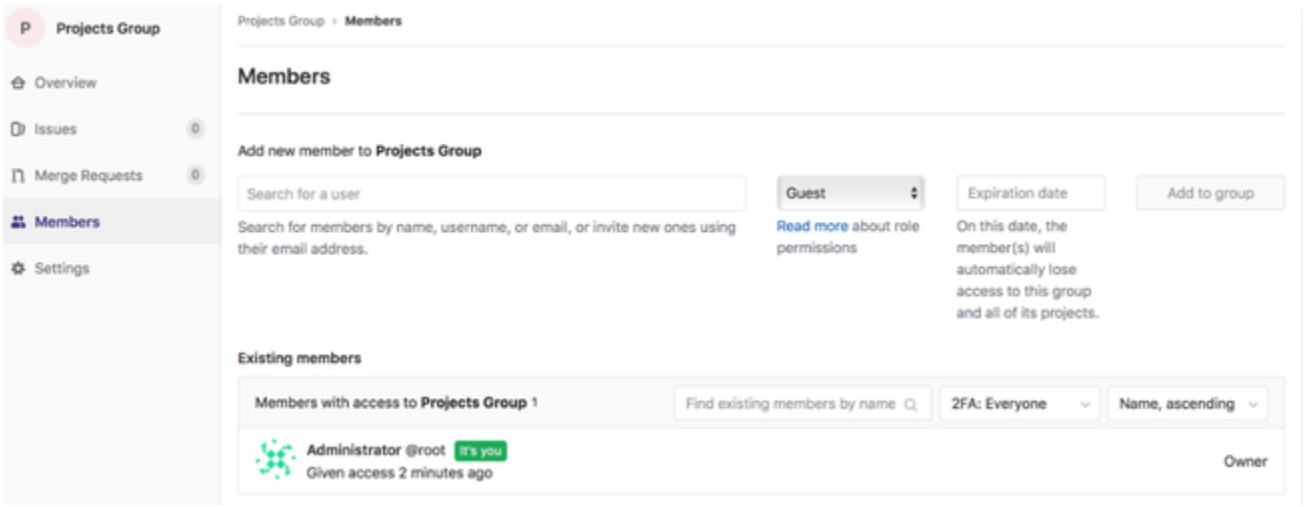
- Alphanumeric characters
- Underscores
- Dashes and dots (it cannot start with dashes or end in a dot)

3. Optionally, you can add a brief description to tell others what this group is about.
4. Optionally, choose an avatar for your group.
5. Choose the visibility level.

Add users to a group

A benefit of putting multiple projects in one group is that you can give a user to access to all projects in the group with one action.

Add members to a group by navigating to the group's dashboard and clicking **Members**.



Projects:-

In GitLab, you can create projects for hosting your codebase, use it as an issue tracker, collaborate on code, and continuously build, test, and deploy your app with built-in GitLab CI/CD.

Your projects can be available publicly, internally, or privately, at your choice. GitLab does not limit the number of private projects you creates.

Project features

When you create a project in GitLab, you'll have access to a large number of features:

Repositories:

Issue tracker: Discuss implementations with your team within issues
Issue Boards: Organize and prioritize your workflow
Multiple Issue Boards: Allow your teams to create their own workflows (Issue Boards) for the same project
Repositories: Host your code in a fully integrated platform
Branches: use Git branching strategies to collaborate on code
Protected branches: Prevent collaborators from messing with history or pushing code without review
Protected tags: Control over who has permission to create tags, and prevent accidental update or deletion
Signing commits: use GPG to sign your commits
Deploy tokens: Manage project-based deploy tokens that allow permanent access to the repository and Container Registry.
Web IDE

Issues and merge requests:

Issue tracker: Discuss implementations with your team within issues
Issue Boards: Organize and prioritize your workflow
Multiple Issue Boards: Allow your teams to create their own workflows (Issue Boards) for the same project
Merge Requests: Apply your branching strategy and get reviewed by your team
Merge Request Approvals: Ask for approval before implementing a change
Fix merge conflicts from the UI: Your Git diff tool right from GitLab's UI

Review Apps: Live preview the results of the changes proposed in a merge request in a per-branch basis
Labels: Organize issues and merge requests by labels
Time Tracking: Track estimate time and time spent on the conclusion of an issue or merge request
Milestones: Work towards a target date
Description templates: Define context-specific templates for issue and merge request description fields for your project
Slash commands (quick actions): Textual shortcuts for common actions on issues or merge requests
Autocomplete characters: Autocomplete references to users, groups, issues, merge requests, and other GitLab elements.
Web IDE

GitLab CI/CD:

GitLab CI/CD: GitLab's built-in Continuous Integration, Delivery, and Deployment tool
Container Registry: Build and push Docker images out-of-the-box
Auto Deploy: Configure GitLab CI/CD to automatically set up your app's deployment
Enable and disable GitLab CI
Pipelines: Configure and visualize your GitLab CI/CD pipelines from the UI
Scheduled Pipelines: Schedule a pipeline to start at a chosen time
Pipeline Graphs: View your entire pipeline from the UI
Job artifacts: Define, browse, and download job artifacts
Pipeline settings: Set up Git strategy (choose the default way your repository is fetched from GitLab in a job), timeout (defines the maximum amount of time in minutes that a job is able run), custom path for .gitlab-ci.yml, test coverage parsing, pipeline's visibility, and much more
Kubernetes cluster integration: Connecting your GitLab project with a Kubernetes cluster
Feature Flags: Feature flags allow you to ship a project in different flavors by dynamically toggling certain functionality
GitLab Pages: Build, test, and deploy your static website with GitLab Pages

Repository:-

A repository is what you use to store your codebase in GitLab and change it with version control. A repository is part of a project, which has a lot of other features.

Create a repository

To create a new repository, all you need to do is create a new project.

Once you create a new project, you can add new files via UI or via command line. To add files from the command line, follow the instructions that will be presented on the screen when you create a new project, or read through them in the command line basics documentation.

Create and edit files

Host your codebase in GitLab repositories by pushing your files to GitLab. You can either use the user interface (UI), or connect your local computer with GitLab through the command line.

To configure GitLab CI/CD to build, test, and deploy you code, add a file called `.gitlab-ci.yml` to your repository's root.

From the user interface:

GitLab's UI allows you to perform lots of Git commands without having to touch the command line. Even if you use the command line regularly, sometimes it's easier to do so via GitLab UI:

- Create a file
- Upload a file
- File templates
- Create a directory
- Start a merge request

From the command line:

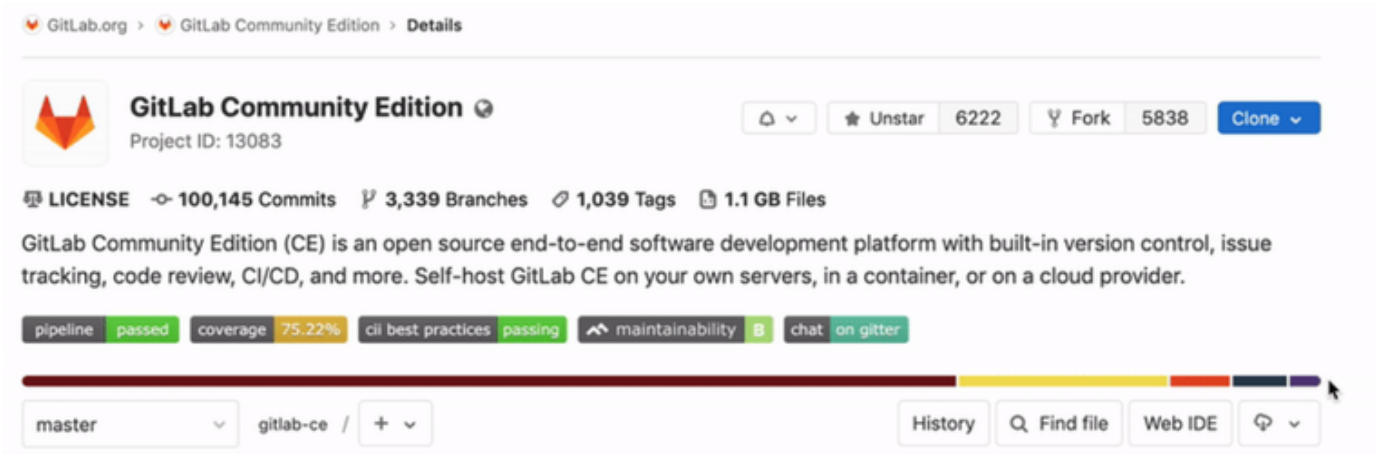
To get started with the command line, please read through the command line basics documentation

Repository graph

Find it under your project's **Repository > Graph**

Repository Languages

For the default branch of each repository, GitLab will determine what programming languages were used and display this on the projects pages. If this information is missing, it will be added after updating the default branch on the project. This process can take up to 5 minutes.



Download Source Code

The source code stored in a repository can be downloaded from the UI. By clicking the download icon, a dropdown will open with links to download the following:

- **Source code:** allows users to download the source code on branch they're currently viewing. Available extensions: `zip`, `tar`, `tar.gz`, and `tar.bz2`.
- **Directory:** only shows up when viewing a sub-directory. This allows users to download the specific directory they're currently viewing. Also available in `zip`, `tar`, `tar.gz`, and `tar.bz2`.
- **Artifacts:** allows users to download the artifacts of the latest CI build

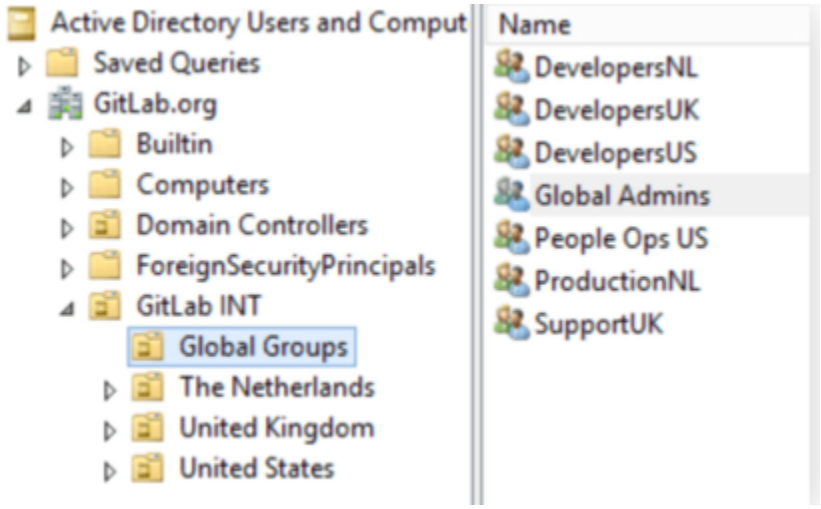
How to configure LDAP with GitLab EE

GitLab Enterprise Edition - LDAP features

GitLab Enterprise Edition (EE) has a number of advantages when it comes to integrating with Active Directory (LDAP):

- Administrator Sync: As an extension of group sync, you can automatically manage your global GitLab administrators. Specify a group CN for `admin_group` and all members of the LDAP group will be given administrator privileges.
- Group Sync: This allows GitLab group membership to be automatically updated based on LDAP group members.
- Multiple LDAP servers: The ability to configure multiple LDAP servers. This is useful if an organization has different LDAP servers within departments. This is not designed for failover. We're working on supporting LDAP failover in GitLab.
- Daily user synchronization: Once a day, GitLab will run a synchronization to check and update GitLab users against LDAP. This process updates all user details automatically.

On the following section, you'll find a description for each of these features. Read through LDAP GitLab EE docs for complementary information.



All members of the group `Global Admins` will be given **administrator** access to GitLab, allowing them to view the `/admin` dashboard

Creating group links

As an example, let's suppose we have a "UKGov" GitLab group, which deals with confidential government information. Therefore, it is important that users of this group are given the correct permissions to projects contained within the group. Granular group permissions can be applied based on the AD group.

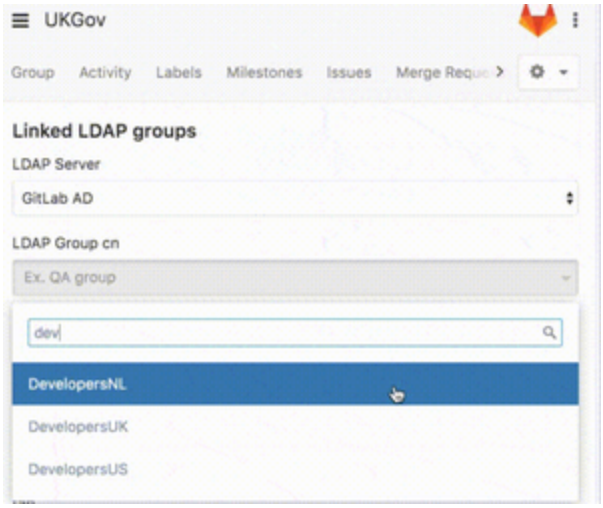
UK Developers of our "UKGov" group are given **"developer"** permissions.

The developer permission allows the development staff to effectively manage all project code, issues, and merge requests.

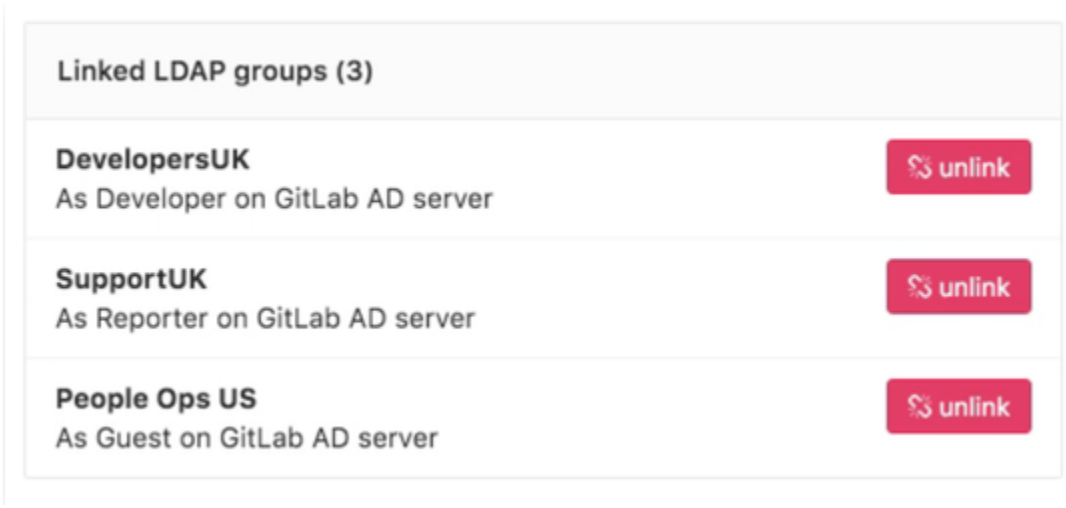
UK Support staff of our "UKGov" group are given **"reporter"** permissions.

The reporter permission allows support staff to manage issues, labels, and review project code.

US People Ops of our "UKGov" group are given **"guest"** permissions.

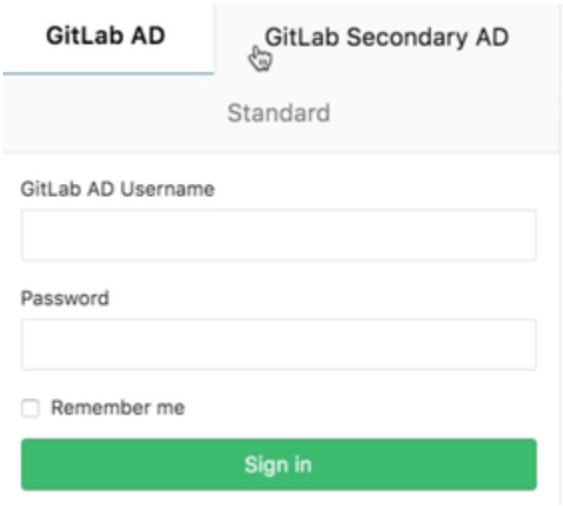


Group permissions - example



Using this permission structure in our example allows only UK staff access to sensitive information stored in the projects code, while still allowing other teams to work effectively. As all permissions are controlled via AD groups new users can be quickly added to existing groups. New group members will then automatically inherit the required permissions.

Multiple LDAP servers



Considering the example illustrated on the image above, our `gitlab.rb` configuration would look like:

```
gitlab_rails['ldap_enabled'] = true
gitlab_rails['ldap_servers'] = {
'main' => {
  'label' => 'GitLab AD',
  'host' =>  'ad.example.org',
  'port' => 636,
  'uid' => 'sAMAccountName',
  'method' => 'ssl',
  'bind_dn' => 'CN=GitLabSRV,CN=Users,DC=GitLab,DC=org',
  'password' => 'Password1',
  'active_directory' => true,
  'base' => 'OU=GitLab INT,DC=GitLab,DC=org',
  'group_base' => 'OU=Global Groups,OU=GitLab INT,DC=GitLab,DC=org',
  'admin_group' => 'Global Admins'
  },

'secondary' => {
  'label' => 'GitLab Secondary AD',
  'host' =>  'ad-secondary.example.net',
  'port' => 636,
  'uid' => 'sAMAccountName',
  'method' => 'ssl',
  'bind_dn' => 'CN=GitLabSRV,CN=Users,DC=GitLab,DC=com',
  'password' => 'Password1',
  'active_directory' => true,
  'base' => 'OU=GitLab Secondary,DC=GitLab,DC=com',
  'group_base' => 'OU=Global Groups,OU=GitLab INT,DC=GitLab,DC=com',
  'admin_group' => 'Global Admins'
  }
}
```