Pro Git(https://git-scm.com/book/ko/v2/)를 바탕으로 작성하였습니다.

YoonJoon Lee
SoC KAIST

Fall 2018

What we will take a look in this series

1. Getting Started
2. Git Basics
3. Git Branch
4. Git Server - GitLab

What we will take a look today

1.Getting Started

2. Git Basics

3. Git Branch

4. Git Server - GitLab

What I will talk about in this section

1. Git on the Server

2. GitLab

Git on the Server

In order to do any collaboration in Git, we'll need to have a remote Git repository.

Although we can technically push changes to and pull changes from individuals' repositories, doing so is discouraged because we can fairly easily confuse what they're working on if we're not careful.

Furthermore, we want our collaborators to be able to access the repository even if our computer is offline — having a more reliable common repository is often useful.

The preferred method for collaborating with someone is to set up an intermediate repository that we both have access to, and push to and pull from that.

Running a Git server is fairly straightforward.

1. We choose which protocols we want our server to communicate with.

2. We setup some options using those protocols and how to get our server running with them.

3. We'll go over a few hosted options, if we don't mind hosting our code on someone else's server and don't want to go through the hassle of setting up and maintaining our own server.

A remote repository is generally a *bare repository* — a Git repository that has no working directory.

Because the repository is only used as a collaboration point, there is no reason to have a snapshot checked out on disk; it's just the Git data.

A bare repository is the contents of our project's `.git` directory and nothing else.

GitLab

There are some several open source solutions out there that we can install instead. As GitLab is one of the more popular ones, we'll cover installing and using it as an example.

This is a bit more complex than the GitWeb option and likely requires more maintenance, but it is a much more fully featured option.

Installation

GitLab is a database-backed web application, so its installation is a bit more involved than some other Git servers. Fortunately, this process is very well-documented and supported.

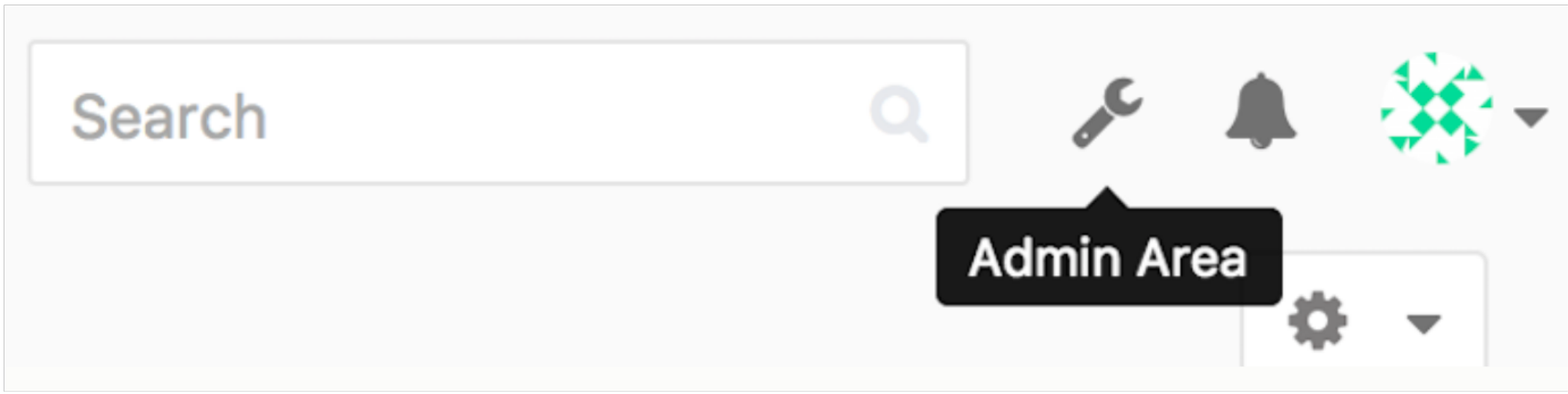

*** Welcome to the BitNami Gitlab Stack ***
*** Built using Ubuntu 12.04 - Kernel 3.2.0-53-virtual (tty2). ***
*** You can access the application at http://10.0.1.17 ***
*** The default username and password is 'user@example.com' and 'bitnami1'. ***
*** Please refer to http://wiki.bitnami.com/Virtual_Machines for details. ***
linux login: _

Administration

GitLab's administration interface is accessed over the web.

Simply point our browser to the hostname or IP address where GitLab is installed, and log in as an admin user.

Once logged in, click the "Admin area" icon in the menu at the top right.

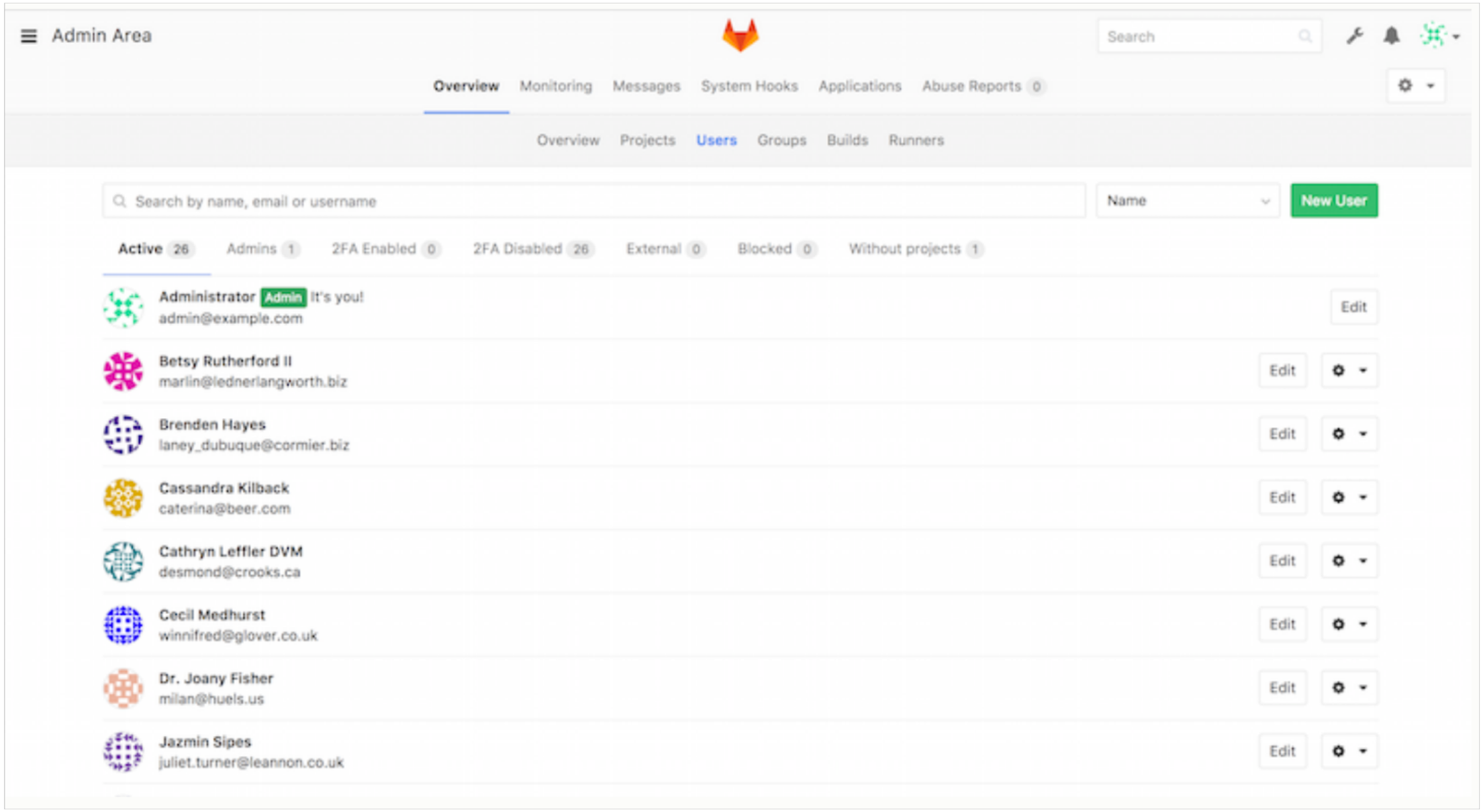


Users

Users in GitLab are accounts that correspond to people.

User accounts don't have a lot of complexity; mainly it's a collection of personal information attached to login data.

Each user account comes with a **namespace**, which is a logical grouping of projects that belong to that user. If the user `jane` had a project named Registra`Mgt`, that project's url would be `http://server/jane/` Registra`Mgt`.

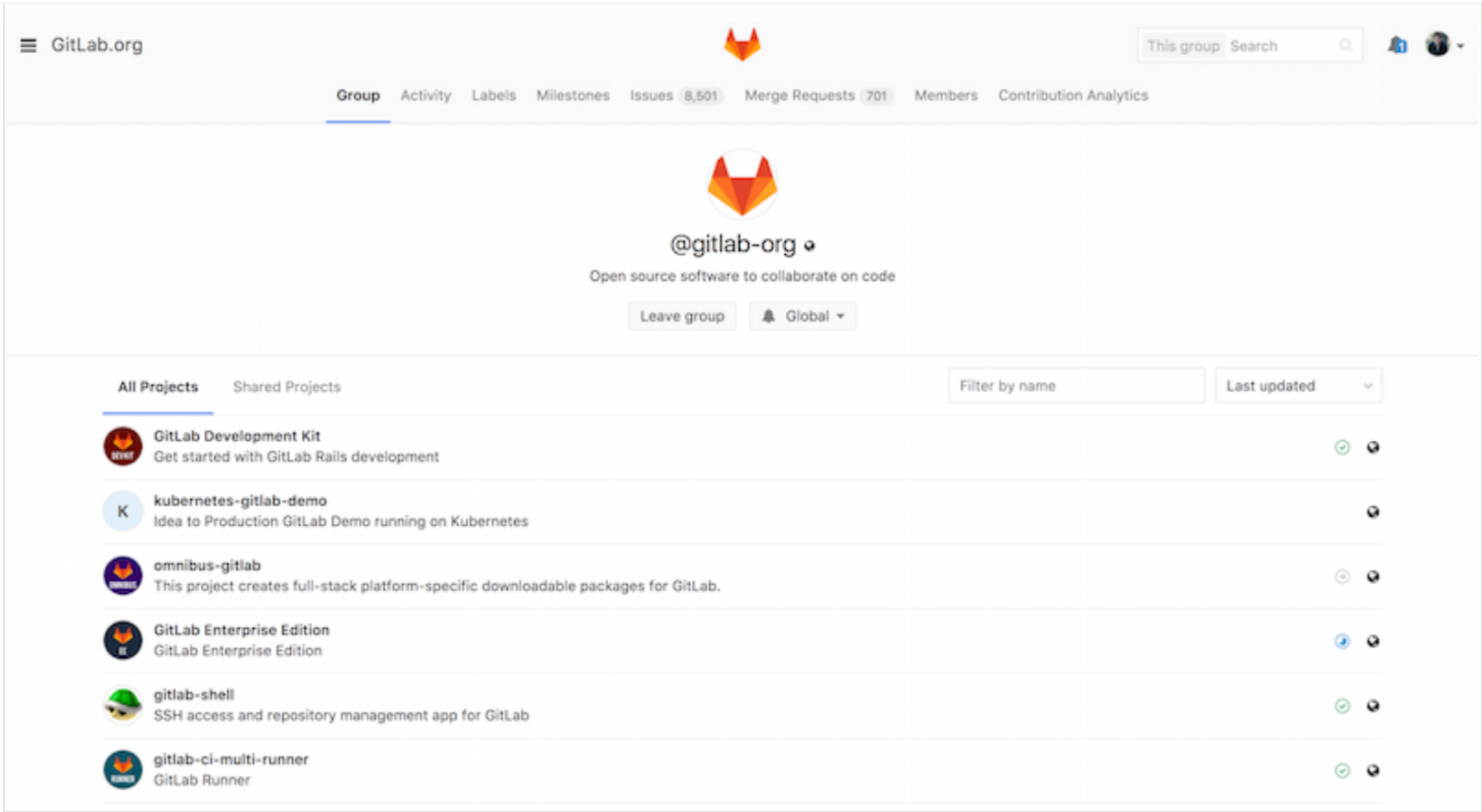

Admin Area
Search
Overview
Monitoring
Messages
System Hooks
Applications
Abuse Reports 0
Overview
Projects
Users
Groups
Builds
Runners
Search by name, email or username
Name
New User
Active 26
Admins 1
2FA Enabled 0
2FA Disabled 26
External 0
Blocked 0
Without projects 1
Administrator Admin It's you!
admin@example.com
Edit
Betsy Rutherford II
marlin@lednerlangworth.biz
Edit
Brenden Hayes
laney_dubuque@cormier.biz
Edit
Cassandra Kilback
caterina@beer.com
Edit
Cathryn Leffler DVM
desmond@crooks.ca
Edit
Cecil Medhurst
winnifred@glover.co.uk
Edit
Dr. Joany Fisher
milan@huels.us
Edit
Jazmin Sipes
juliet.turner@leannon.co.uk
Edit

Groups

A GitLab group is an assemblage of projects, along with data about how users can access those projects.

Each group has a project namespace (the same way that users do), so if the group `Lilac` has a project RegistraMgt, its url would be http://server/`Lilac`/RegistraMgt.

Each group is associated with a number of users, each of which has a level of permissions for the group's projects and the group itself. These range from "Guest" (issues and chat only) to "Owner" (full control of the group, its members, and its projects).


GitLab.org
This group Search
Group Activity Labels Milestones Issues 8,501 Merge Requests 701 Members Contribution Analytics
@gitlab-org
Open source software to collaborate on code
Leave group
Global
All Projects
Shared Projects
Filter by name
Last updated
GitLab Development Kit
Get started with GitLab Rails development
kubernetes-gitlab-demo
Idea to Production GitLab Demo running on Kubernetes
omnibus-gitlab
This project creates full-stack platform-specific downloadable packages for GitLab.
GitLab Enterprise Edition
GitLab Enterprise Edition
gitlab-shell
SSH access and repository management app for GitLab
gitlab-ci-multi-runner
GitLab Runner

Projects

A GitLab project roughly corresponds to a single Git repository.

Every project belongs to a single namespace, either a user or a group.

If the project belongs to a user, the owner of the project has direct control over who has access to the project; if the project belongs to a group, the group's user-level permissions will also take effect.

Every project also has a visibility level, which controls who has read access to that project's pages and repository.

If a project is *Private*, the project's owner must explicitly grant access to specific users. An *Internal* project is visible to any logged-in user, and a *Public* project is visible to anyone.

Note that this controls both `git fetch` access as well as access to the web UI for that project.

Hooks

GitLab includes support for hooks, both at a project or system level.

For either of these, the GitLab server will perform an HTTP POST with some descriptive JSON whenever relevant events occur.

This is a great way to connect your Git repositories and GitLab instance to the rest of your development automation, such as CI servers, chat rooms, or deployment tools.

Basic Usage

The first thing we'll want to do with GitLab is create a new project.

This is accomplished by clicking the "+" icon on the toolbar.

We'll be asked for the project's name, which namespace it should belong to, and what its visibility level should be.

Most of what we specify here isn't permanent, and can be re-adjusted later through the settings interface.

Click "Create Project", and we're done.

Once the project exists, we'll probably want to connect it with a local Git repository.

Each project is accessible over HTTPS or SSH, either of which can be used to configure a Git remote.

The URLs are visible at the top of the project's home page.

For an existing local repository, this command will create a remote named `gitlab` to the hosted location:

```
$ git remote add gitlab https://server/namespace/project.git
```

```
$ git clone https://server/namespace/project.git
```

Working Together

The simplest way of working together on a GitLab project is by giving another user direct push access to the Git repository.

We can add a user to a project by going to the "Members" section of that project's settings, and associating the new user with an access level (the different access levels are discussed a bit in Groups).

By giving a user an access level of "Developer" or above, that user can push commits and branches directly to the repository with impunity.

Another, more decoupled way of collaboration is by using merge requests.

This feature enables any user that can see a project to contribute to it in a controlled way.

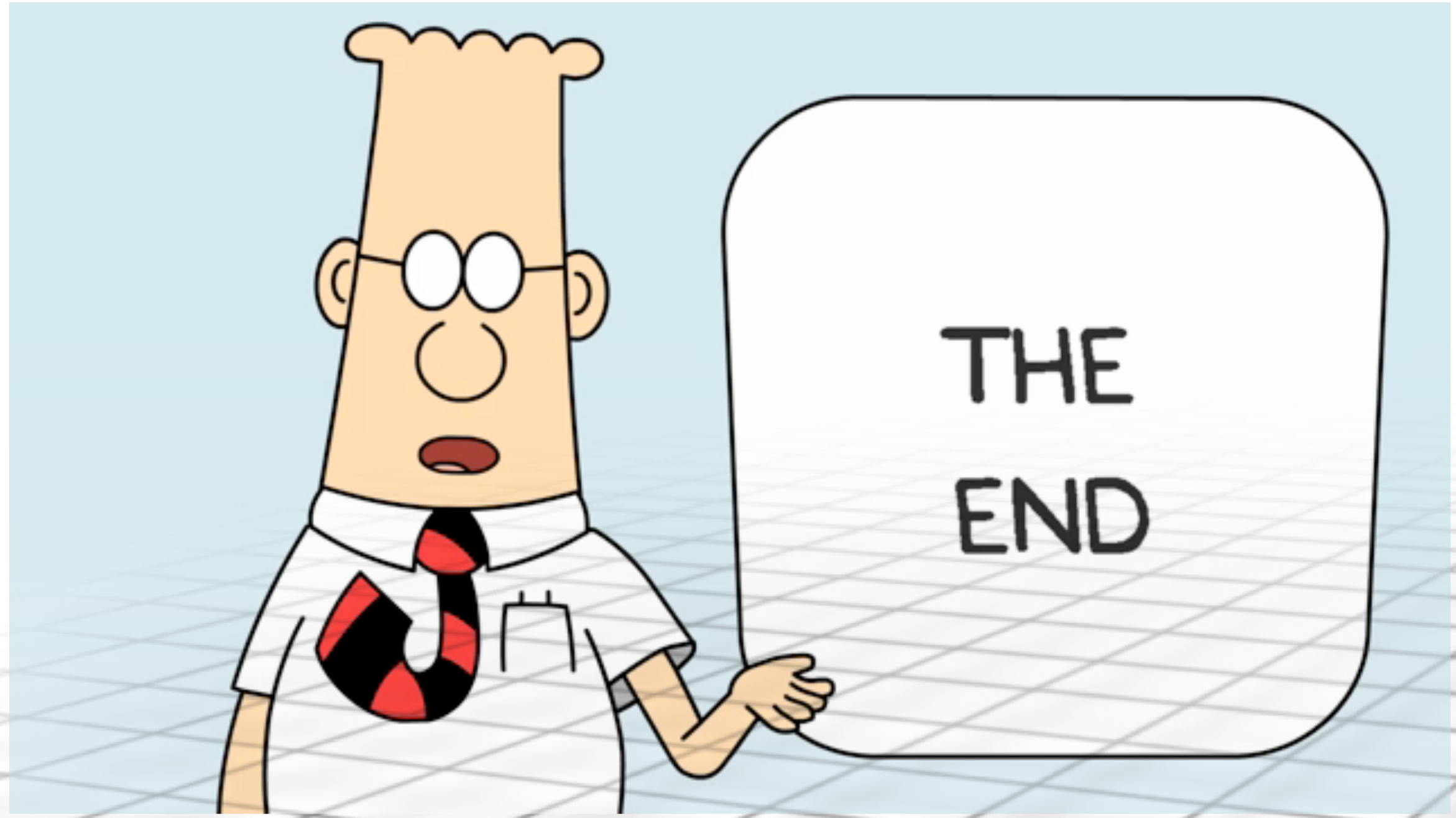
Users with direct access can simply create a branch, push commits to it, and open a merge request from their branch back into `master` or any other branch.

Users who don't have push permissions for a repository can "fork" it (create their own copy), push commits to *that* copy, and open a merge request from their fork back to the main project. This model allows the owner to be in full control of what goes into the repository and when, while allowing contributions from untrusted users.

Merge requests and issues are the main units of long-lived discussion in GitLab.

Each merge request allows a line-by-line discussion of the proposed change (which supports a lightweight kind of code review), as well as a general overall discussion thread.

Both can be assigned to users, or organized into milestones.



출처: metachannels.com

감사합니다