GitLab Cookbook

Over 60 hands-on recipes to efficiently self-host your own Git repository using GitLab

Jeroen van Baarsen
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 "Introduction and Installation"
- A synopsis of the book’s content
- More information on GitLab Cookbook

About the Author

**Jeroen van Baarsen** started programming at the age of 14. His language of choice was PHP. He started his first programming job at the age of 16 and worked in several companies as a PHP developer before he found out about the wonderful language that Ruby is. He then started learning this language and took up a job as a Ruby developer.

Currently, he works at Firmhouse, which is a company that helps build foundations for innovations and new business. Firmhouse has created the Ruby on Rails hosting platform at intercityup.com.

In his spare time, he contributes to GitLab's open source core team as a member. He is responsible for the merge requests that are opened by the community.

I would like to thank my girlfriend for supporting me while I was writing this book.
GitLab Cookbook

GitLab is a popular, open source Git hosting solution implemented by more than 50,000 organizations. Over the last few years, GitLab has evolved with strong community support and growth, handling thousands of users on a single server and several such servers on an active cluster. If you need to set up a Git server, GitLab provides a perfect solution for you!

This book has some carefully chosen recipes to help you decide on the type of GitLab installation that will fit your requirements. You will also explore the benefits of each of these installation types.

Along with covering some of the basic principles of Git, the book covers practical scenarios that will show how you or your organization can effectively manage your proprietary code. You will learn how to manage multiple users, groups, and the permissions GitLab has for them. Updating your GitLab instance, creating backups, and restoring backups are some of the important tasks that are described in detail in order to assist you in maintaining your GitLab server. Moreover, the GitLab API is extensively covered to guide you through the various operations to help you manage your project. Among some of the more complex stuff, you will see how to incorporate the Git workflow and integrate GitLab in your existing LDAP environment. Furthermore, the book takes a peek at the GitLab CI, which is a continuous integration service specially built by people from GitLab for GitLab.

To summarize, this Cookbook will provide you with the knowledge required to work effortlessly toward self-hosting your Git repositories and maintaining them. Whether you are dealing with small or large projects, the recipes conveniently provide you with a thorough learning curve to help you handle the repository with speed and efficiency.

What This Book Covers

*Chapter 1, Introduction and Installation*, looks at the different ways in which you can install GitLab. We will also create your first project on your own GitLab server.

*Chapter 2, Explaining Git*, looks at the basics of Git, how to send your code to your GitLab server, how to get your GitLab project on your computer, and how to squash your commits into one nice commit.

*Chapter 3, Managing Users, Groups, and Permissions*, looks at how you can add new users and what the different permissions that you can grant your users are. We will also look at the power of GitLab groups.

*Chapter 4, Issue Tracker and Wiki*, shows how GitLab has a powerful issue tracker and wiki system. In this chapter, we will take a look at how you can get the most out of it.
Chapter 5, *Maintaining Your GitLab Instance*, shows how you need to maintain a GitLab instance and update backups when running them. We will look it all up!

Chapter 6, *Webhooks, External Services*, and the API, looks at webhooks and shows how you can test them. We will look at linking your GitLab system to external services such as project management tools, and we will look at the powerful API that GitLab ships with.

Chapter 7, *Using LDAP and OmniAuth Providers*, looks at how you can use your LDAP infrastructure to manage your GitLab users. We will also look at how to use external authentication methods called OmniAuth providers.

Chapter 8, *GitLab CI*, talks about GitLab having a powerful Continuous Integration system. In this chapter, we will look at how you can install and link it to your GitLab server.

Appendix, *Tips and Tricks*, looks at some smaller tasks, tips, and tricks that can help you in your daily GitLab usage.
In this chapter, we will cover the following recipes:

- Using the Omnibus package
- Setting up the server dependencies for source installation
- Setting up the database for source installation
- Installing GitLab from source
- Using Chef and GitLab Cookbook
- Logging in for the first time
- Creating your first project

**Introduction**

GitLab is a self-hosted system for managing your code. It was first released in October 2011, and is updated every twenty-second day of the month since then. It was released under the MIT license.

It used to be hosted on GitHub, but since January 2014, its main source of hosting is gitlab.com. The fork of GitLab, which is hosted on GitHub, will remain active as a source where you can file issues and merge requests.

GitLab was founded by Dmitriy Zaporozhets in 2013. He has worked on GitLab full-time since 2013. The project consists of two main groups: on one side, the open source core team, and on the other side, the GitLab B.V. team (the second one is the company side of GitLab).
Besides the amazing project management tool for Git projects, the GitLab Continuous Integration (CI) system also exists; this is a CI system that highly integrates with GitLab.

In this book, we will be using the Community Edition (CE) of GitLab. The CE version is the free open source version that you can download. There is also an enterprise version. The enterprise version includes a support subscription where the GitLab B.V. team helps you with problems with the installation of it.

If you want to give GitLab a try, or just don't want to host it yourself, take a look at gitlab.com. You can find the hosted version of GitLab there. It's run by the team behind GitLab B.V., so you know you're in good hands!

**Using the Omnibus package**

The people from GitLab have created an Omnibus package for the major Linux distributions (Ubuntu, Debian, and CentOS). These packages are an easy way of installing your GitLab installation, as they have all the dependencies packaged with them.

For this recipe, we are going to use the Ubuntu version of the Omnibus package.

**Getting ready**

Before we start installing, you need to have a server installed with a Ubuntu 12.04 64-bit system, and have SSH access to the server.

**How to do it...**

Here is how you can install GitLab using Omnibus:

1. Download the package, change the \texttt{X.Y.Z} portion to the version you want to download. At the time of writing, 7.3.2 is the latest version:
   
   
   \texttt{wget https://downloads-packages.s3.amazonaws.com/ubuntu-12.04/gitlab\_X\_Y\_Z-omnibus.5.1.0.ci-1_amd64.deb}

2. Install the OpenSSH server:

   \texttt{sudo apt-get install openssh-server}

3. Install Postfix:

   \texttt{sudo apt-get install postfix}

4. Postfix will ask you what kind of installation you want; choose the Internet Site option.
5. Enter the fully qualified domain name for the domain you want GitLab to send e-mails from (that is, gitlab.example.com).

6. Install the Omnibus package, and replace the X.Y.Z part with your version:
   ```bash
   sudo dpkg -i gitlab_7.3.2-omnibus.5.1.0.ci-1_amd64.deb
   ```

7. Now, we will configure your GitLab instance by running the following command:
   ```bash
   sudo gitlab-ctl reconfigure
   ```

8. Check to see whether all the services are running using the following command:
   ```bash
   sudo gitlab-ctl status
   ```

The output should look like the following screenshot:

```
run: nginx: (pid 2499) 359s; run: log: (pid 2498) 359s
run: postgresql: (pid 2415) 389s; run: log: (pid 2414) 389s
run: redis: (pid 2333) 395s; run: log: (pid 2332) 395s
run: sidekiq: (pid 2484) 365s; run: log: (pid 2483) 365s
run: unicorn: (pid 2473) 367s; run: log: (pid 2472) 367s
```  

You can now log in with the username `root` and password `5iveL!fe`.

**How it works...**

We have just installed GitLab via the Omnibus installer. Omnibus is a way to package your application and install all the dependencies that are required to run it.

GitLab makes good use of this capability. As you only have to install a mail server and an OpenSSH server, all the other parts are automatically installed for you.

GitLab will auto configure itself with the recommended settings. If you want to change your configuration, you have to first create the configuration file. You can do this as follows:

```
$ sudo mkdir -p /etc/gitlab
$ sudo touch /etc/gitlab/gitlab.rb
$ sudo chmod 600 /etc/gitlab/gitlab.rb
```

The settings that you are most likely to change are the ones that are in the `gitlab.yml` file (https://gitlab.com/gitlab-org/gitlab-ce/blob/master/config/gitlab.yml.example).

You need to make a small translation change in order to make this work, so let’s say you want to change the port that GitLab is running on; normally, you would change the port value in the GitLab file, but now you have to add an entry to `/etc/gitlab/gitlab.rb`.  

---
So, for the port, the entry in `gitlab.yml` would look like the following code:

```yaml
production: &base
  gitlab:
    port: 80
```

In the `gitlab.rb` file, you need to create the following entry: `gitlab_rails['port'] = 80`.

### Setting up the server dependencies for source installation

To install GitLab from source, we need to install some dependencies on the server. Besides installing the required packages, we will also create the user that will serve our GitLab installation.

#### How to do it...

The installation procedure is applicable for Debian-based systems only. GitLab also supports other Linux distributions, but the installation process is a bit different. For more information, visit the [gitlab.com](http://gitlab.com) website. Perform the following steps:

1. Install the required packages using the following command:
   ```bash
   $ sudo apt-get install -y build-essential zlib1g-dev libyaml-dev libssl-dev libreadline-dev libncurses5-dev libffi-dev curl openssh-server redis-server checkinstall libxml2-dev libxslt-dev libcurl4-openssl-dev libicu-dev logrotate
   ```

2. Install Ruby using the following commands:
   ```bash
   $ mkdir /tmp/ruby && cd /tmp/ruby
   $ ./configure --disable-install-rdoc
   $ make
   $ sudo make install
   ```

3. Install the Bundler gem:
   ```bash
   $ sudo gem install bundler --no-ri --no-rdoc
   ```

4. Create the Git user:
   ```bash
   $ sudo adduser --disabled-login --gecos 'GitLab' git
   ```
Chapter 1

How it works...

At this point, we have installed the server dependencies and installed Ruby on our system. We also installed the Bundler gem; *bundler* is the package manager for Ruby, and as GitLab is written in Ruby, this one will be very important later on, so we can download all the dependencies for GitLab.

The Git user was created so that GitLab and all its dependencies can run under its own user; that way, it is possible to sandbox the installation better and make sure that our system stays secure.

Set up the database for source installation

GitLab can be installed using PostgreSQL or MySQL. In this recipe, I will use PostgreSQL as it is the recommended database engine.

How to do it...

The following steps are applicable for Debian-based systems; they are also possible with RedHat. You can take a look at gitlab.com for the installation instructions.

1. Install PostgreSQL 9.1:
   ```bash
   sudo apt-get install -y postgresql-9.1 postgresql-client libpq-dev
   ```
2. Create the PostgreSQL user for GitLab:
   ```bash
   sudo –u postgresql psql –d template1 –c "CREATE USER git CREATEDB"
   ```
3. Create the database for GitLab and grant all privileges on the database:
   ```bash
   sudo –u postgresql psql –d template1 –c "CREATE DATABASE gitlabhq_production OWNER git"
   ```

Installing GitLab from source

In this recipe, I will help you install GitLab from source. Installing from source means that we will take the source code from gitlab.com and use that to code in order to install it on our server.

At the time of writing, 7.4 is the latest version. If you want to be sure that you have the latest version, please check https://gitlab.com/gitlab-org/gitlab-ce/blob/master/VERSION.

If you want the latest bleeding edge version, you can change 7.4 to master in this recipe.
Getting ready

To install GitLab on your own server, you need to have a few things installed already. Here is a list of prerequisites:

- A server running Debian or Ubuntu; preferably one of the latest versions, and running as 64-bit
- Git Version 1.7 or higher
- A text editor; in the examples, I’ll be using Vim
- Sudo; on Debian this is not installed by default
- A working mail server
- You have to set up the database
- You have to install all the server dependencies

How to do it...

Follow these steps to install GitLab from source:

1. Download the source code:
   ```
   $ cd /home/git
   $ sudo -u git -H git clone https://gitlab.com/gitlab-org/gitlab-ce.git -b 6-9-stable gitlab
   ```

2. In `config/gitlab.yml`, we need to change the host to the fully-qualified domain name of your GitLab instance. Also change the `email_from` to the e-mail address you want to use as a from address for all the e-mails that are sent by GitLab:
   ```
   $ cd /home/git/gitlab
   $ sudo -u git -H cp config/gitlab.yml.example config/gitlab.yml
   $ sudo -u git -H editor config/gitlab.yml
   ```

3. Make sure that GitLab can write to the necessary folders using the following command lines:
   ```
   $ sudo chown -R git log/
   $ sudo chown -R git tmp/
   $ sudo chmod -R u+rwX log/
   $ sudo chmod -R u+rwX tmp/
   $ sudo chmod -R u+rwX tmp/pids/
   $ sudo chmod -R u+rwX tmp/sockets/
   $ sudo chmod -R u+rwX public/uploads
   ```
4. Create the directory for the GitLab satellites:
   
   ```
   $ sudo -u git -H mkdir /home/git/gitlab-satellites
   $ sudo chmod u+rwx,g+rx,o-rwx /home/git/gitlab-satellites
   ```

5. Copy the Unicorn config file:
   
   ```
   $ sudo -u git -H cp config/unicorn.rb.example config/unicorn.rb
   ```

6. Copy the Rack attack config file:
   
   ```
   $ sudo -u git -H cp config/initializers/rack_attack.rb.example config/initializers/rack_attack.rb
   ```

7. Copy the init script and make GitLab start on boot:
   
   ```
   $ sudo cp lib/support/init.d/gitlab /etc/init.d/gitlab
   $ sudo update-rc.d gitlab defaults 21
   ```

8. Set up Logrotate:
   
   ```
   $ sudo cp lib/support/logrotate/gitlab /etc/logrotate.d/gitlab
   ```

9. Set up the Git user. This helps when editing files via the GitLab web interface. Make sure that the `user.email` is the same as the e-mail address you entered in step 8:
   
   ```
   $ sudo -u git -H git config --global user.name "GitLab"
   $ sudo -u git -H git config --global user.email "example@example.com"
   $ sudo -u git -H git config --global core.autocrlf input
   ```

10. Configure your GitLab database:
    
    ```
    $ sudo -u git -H cp config/database.yml.postgresql config/database.yml
    ```

11. Make sure that the `database.yml` file is only readable for the `git` user:
    
    ```
    $ sudo -u git -H chmod o-rwx config/database.yml
    ```

12. Install the GitLab dependencies:
    
    ```
    $ sudo -u git -H bundle install --deployment --without development test mysql aws
    ```

13. Install the GitLab shell:
    
    ```
    ```

14. Initialize the database:
    
    ```
    $ sudo -u git -H bundle exec rake gitlab:setup
    RAILS_ENV=production force=yes
    ```
15. Compile all the assets:
   
   ```bash
   $ sudo -u git -H bundle exec rake assets:precompile
   RAILS_ENV=production
   ```

16. Start your GitLab instance:

   ```bash
   $ sudo /etc/init.d/gitlab restart
   ```

**How it works...**

We have just installed GitLab on our server. We have done this by downloading the source code from gitlab.com and performing the preceding steps.

Let's take a look at what we did in every step.

In step 1, we downloaded the source code for GitLab. We haven't done anything with it yet, just downloaded it.

In step 2, we done the basic configuration of GitLab. We have changed the hostname to the fully-qualified domain name you want to access your GitLab on, for example, gitlab.example.com. Also, we have configured the e-mail addresses that GitLab will send mails from. The `email_from` will be used as the from address of all the e-mails that are sent via GitLab.

Next was the setup for the satellite directory. Satellites are used when you create a merge request, and GitLab has to check whether it is mergeable, and perform the actual merge. A satellite is just checking out of the repository that GitLab has access to.

We then went on to copy some files. The first file was the Unicorn configuration file. Unicorn was used as the application server and we copied the Rack Attack files. Rack Attack is used in order to prevent abusive requests to your GitLab server. One way to make sure that no harmful requests make it to your server is by request throttling. This means that we only allow 10 requests per 60 seconds to certain URLs.

The next important step is the configuration of the database. As we are using PostgreSQL on the same machine, the configuration is really straightforward: just copy the right `database.yml` file and we are done, well almost; we also need to protect our `database.yml` file so that it's only readable for the Git user.

The dependency installation is done via Bundler. Bundler is the package manager for Ruby. We have passed the flags `--deployment` and `--without development test mysql aws`. The first flag is passed to make sure that the dependencies are installed in the context of GitLab and not in a system-wide context. The second flag is passed to make sure that only the necessary dependencies are installed. If you want to learn more about Bundler, take a look at www.bundler.io.
GitLab has its own Git wrapper to perform Git commands on the server. This wrapper is called GitLab Shell. In step 11, we tell GitLab to fetch the code for GitLab Shell and install it.

In step 12, we set up the database. We create the database and load the database schema. We also create the first user, so that you can log in to your server.

**Using Chef and GitLab Cookbook**

You can install GitLab using chef-solo. It allows you to install a server and all of its dependencies through a pre-programmed script. GitLab Cookbook is also used as the base for the Omnibus Package.

If you want more information on Chef, please take a look at [www.getchef.com](http://www.getchef.com).

For this recipe, we are going to use a Ubuntu-based installation.

**Getting ready**

Before we start installing, you need to have a server installed with Ubuntu and have SSH access to the server. Your server needs to have at least 2 GB of RAM to compile all the requirements.

**How to do it...**

1. We start with downloading some server dependencies:
   
   ```
   sudo apt-get update && sudo apt-get install -y build-essential
   git
   curl
   ```

2. Download the chef-solo file:
   
   ```
   ```

3. We have to edit the file we just downloaded so that it fits our needs:
   
   ```
   vi /tmp/solo.json
   ```

4. As we will be using PostgreSQL, you can remove the MySQL part. Also, make sure you change the revision to the latest stable branch, 7.3 at time of writing. After you are done, your file should look like the following code, but with your own host and e-mail addresses:
   
   ```json
   "gitlab": {
     "host": "gitlab.example.com",
     "url": "http://gitlab.example.com/",
     "email_from": "gitlab@example.com",
     "support_email": "support@gitlab.example.com",
     "database_adapter": "postgresql",
   }
   ```


"database_password": "super-secure-password",
"revision": "6-9-stable"
},
"postgresql": {
  "password": {
    "postgres": "psqlpass"
  }
},
"postfix": {
  "mail_type": "client",
  "myhostname": "gitlab.example.com",
  "mydomain": "mydomain.com",
  "myorigin": "gitlab.example.com",
  "smtp_use_tls": "no"
},
"run_list": [
  "postfix",
  "gitlab::default"
]
}

5. Next, we download and install Chef to our server:
   cd /tmp; curl -LO https://www.opscode.com/chef/install.sh;
sudo bash ./install.sh -v 11.4.4; sudo
   /opt/chef/embedded/bin/gem install berkshelf --no-ri --no-rdoc

6. Now, we download the GitLab source from gitlab.com:
   git clone https://gitlab.com/gitlab-org/cookbook-gitlab.git
   /tmp/cookbook-gitlab

7. Install all the GitLab-specific dependencies:
   cd /tmp/cookbook-gitlab; /opt/chef/embedded/bin/berks vendor
   /tmp/cookbooks

8. We need to create one more Chef config file:
   vi /tmp/solo.rb

9. Add the following content to the preceding config file:
   cookbook_path ["/tmp/cookbooks/"]
   log_level :debug

10. Save the file.
11. We are done with configuring everything and now let's install GitLab!
   sudo chef-solo -c /tmp/solo.rb -j /tmp/solo.json
How it works...

We just installed GitLab via the Chef cookbook. This way of installation is a little more automated than the installation from source, but it still gives you a bit more control over your installation in comparison to the Omnibus package.

Let's go through the steps that we took to install GitLab in this way.

First, we had to install some server dependencies that were needed to install Chef, and we also cloned the code from GitLab. The dependencies included Curl and Git. We used Curl to download the chef.json file, and in step 4, to download the check installation file. Git was needed to clone the source of GitLab, and to make sure that GitLab, when installed, is able to serve your repositories.

Next, we had to download the config.json file. This JSON file keeps the configuration information for GitLab in order to install itself. You can compare this to the gitlab.yml file from the Installing GitLab from source recipe.

In this recipe, we installed GitLab using PostgreSQL. If you'd prefer to install it with MySQL, that's possible. Just keep in mind that PostgreSQL is the recommended way of running GitLab.

The next step was to download the GitLab source and to install the GitLab dependencies. After we had done that, we created the solo.rb file. This file is used by chef-solo to know where GitLab Cookbook is located.

The last step was to install GitLab itself. This step took a while because the command also downloaded and compiled Ruby for you.

Logging in for the first time

When you have installed your server, you need to log in. GitLab comes with a built-in administrator account.

How to do it...

In this recipe, we will log in and create our own administrator account as follows:

1. Go to your domain where GitLab is installed (that is, gitlab.example.com).
2. Log in using the username root and password 5iveL!fe.
3. You need to choose a new password; pick whatever you like.
4. Log in with the new information.
5. Go to the **Admin area** section, as shown in the following screenshot:

![Admin area screenshot](image)

6. Navigate to **Users** | **New User**.
7. Fill in the information to create your own user. Don't forget to check the **Admin** checkbox.
8. Now, click on **Create user**.

![Access control settings](image)

An e-mail will be sent to the given e-mail address. This e-mail will contain the new password for this account.

9. Log out from GitLab and log in with your new account.
10. You need to choose a new password and log in again.
11. Go to the **Admin area** section and click on **Users**.
12. Click on **Block** for the administrator account.
Chapter 1

**How it works...**

As GitLab ships with a default administrator account, this makes it a bit unsecure by default. Therefore, when you don’t change anything about your setup, everyone will be able to log in.

To ensure that we have a secure GitLab installation, we created a new administrator account, and gave it an administrator access. We also want to be sure that the default shipped account is not useable anymore, so that’s why we logged in with our own account and disabled the account given by GitLab.

We had to log out first, as GitLab does not allow us to disable the account that is currently logged in.

**Creating your first project**

In this recipe, we will take a look at creating a project, and what the different visibility options are. The difficult part for people new to GitLab is the different visibility levels that GitLab offers. There are three visibility levels: private, internal, and public. They are explained as follows:

- **Private projects**: When you set the visibility to Private, the only people who can see this project are the people that you have granted access to this project, or those people who are members of the project's group.
- **Internal projects**: This will cause the project to be visible to all the users who have an account on your GitLab server.
- **Public projects**: GitLab offers a public access directory for projects when you choose Public for your project. This will be visible to everyone who knows the location of your GitLab server. When someone who has no account on your GitLab server views this project, they will have guest permissions.

The visibility level has nothing to do with the permissions someone has on your project. So, when you list your project as Internal, it does not mean that every user can change whatever they want. It just means that users who have an account can clone the project, and view issues and such.

You can always change the visibility of your project after you have created it. Just perform the following steps:

1. Go to your project dashboard.
2. Click on the **Edit** button.
3. Change the **Visibility level** option.
How to do it...

Follow these steps to create your project:

1. Click on the **New Project** button on the right-hand side, as shown in the following screenshot:

![New Project Button](image)

2. Fill in the project title. Let's pick **Cookbook**.
3. You can fill in an optional description.
4. Choose visibility level as **Private**.
5. Click on **Create Project**.
Where to buy this book
You can buy GitLab Cookbook from the Packt Publishing website.
Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.
Click here for ordering and shipping details.