GNU/Linux Administration - Support #428

Installing GitLab on Debian Wheezy

08/06/2014 06:01 PM - Daniel Curtis

Status:	Closed	**Start date:**	08/06/2014
Priority:	Normal	**Due date:**	
Assignee:	Daniel Curtis	**% Done:**	100%
Category:	Source Code Management	**Estimated time:**	4.00 hours
Target version:		**Spent time:**	6.00 hours

Description

Install Packages & Dependencies

- sudo is not installed on Debian by default. Make sure your system is up-to-date and install it.

```
su -
apt-get update -y
apt-get upgrade -y
apt-get install sudo -y
```

Note: During this installation some files will need to be edited manually. If you are familiar with vim set it as default editor with the commands below. If you are not familiar with vim please skip this and keep using the default editor.

- Install vim and set as default editor

```
sudo apt-get install -y vim
sudo update-alternatives --set editor /usr/bin/vim.basic
```

- Install the required packages (needed to compile Ruby and native extensions to Ruby gems):

```
sudo apt-get install -y build-essential zlib1g-dev libyaml-dev libssl-dev libgdbm-dev libreadl
ine-dev libncurses5-dev libffi-dev curl openssh-server redis-server checkinstall libxml2-dev l
ibxslt-dev libcurl4-openssl-dev libicu-dev logrotate python-docutils
```

- Make sure you have the right version of Git installed

1. Install Git

```
sudo apt-get install -y git-core
```

2. Make sure Git is version 1.7.10 or higher, for example 1.7.12 or 2.0.0

```
git --version
```

(Optional) Is the system packaged Git too old? Remove it and compile from source.

- Remove packaged Git

```
sudo apt-get remove git-core
```

- Install dependencies

```
sudo apt-get install -y libcurl4-openssl-dev libexpat1-dev gettext libz-dev libssl-dev build-essential
```

- Download and compile from source

```
cd /tmp
curl --progress https://www.kernel.org/pub/software/scm/git/git-2.0.0.tar.gz | tar xz
cd git-2.0.0/
make prefix=/usr/local all
```

- Install into /usr/local/bin

```
sudo make prefix=/usr/local install
```

- When editing config/gitlab.yml (Step 5), change the git bin_path to /usr/local/bin/git

Note: In order to receive mail notifications, make sure to install a mail server. By default, Debian is shipped with exim4 but this has problems while Ubuntu does not ship with one. The recommended mail server is postfix and you can install it with:

```
sudo apt-get install -y postfix
```

Then select 'Internet Site' and press enter to confirm the hostname.

Compile Ruby 2.1.2

The use of ruby version managers such as RVM, rbenv or chruby with GitLab in production frequently leads to hard to diagnose problems. For example, GitLab Shell is called from OpenSSH and having a version manager can prevent pushing and pulling over SSH. Version managers are not supported and we stronly advise everyone to follow the instructions below to use a system ruby.

- Remove the old Ruby 1.8 if present

```
sudo apt-get remove ruby1.8
```

- Download Ruby and compile it:

```
mkdir /tmp/ruby && cd /tmp/ruby
curl --progress ftp://ftp.ruby-lang.org/pub/ruby/2.1/ruby-2.1.2.tar.gz | tar xz
cd ruby-2.1.2
./configure --disable-install-rdoc
make
sudo make install
```

- Install the Bundler Gem:

```
sudo gem install bundler --no-ri --no-rdoc
```

System Users

- Create a git user for Gitlab:

```
sudo adduser --disabled-login --gecos 'GitLab' git
```

Install the PostgreSQL Database Server

We recommend using a PostgreSQL database. For MySQL check MySQL setup guide.

Note: because we need to make use of extensions you need at least pgsql 9.1.

- Install the database packages

```
sudo apt-get install -y postgresql-9.1 postgresql-client libpq-dev
```

- Login to PostgreSQL

```
sudo -u postgres psql -d template1
```

- Create a user for GitLab.

```
template1=# CREATE USER git CREATEDB;
```

- Create the GitLab production database & grant all privileges on database

```
template1=# CREATE DATABASE gitlabhq_production OWNER git;
```

- Quit the database session

```
template1=# \q
```

- Try connecting to the new database with the new user

```
sudo -u git -H psql -d gitlabhq_production
```

Download GitLab

- We'll install GitLab into home directory of the user "git"

```
cd /home/git
```

- Clone GitLab repository

```
sudo -u git -H git clone https://gitlab.com/gitlab-org/gitlab-ce.git -b 7-1-stable gitlab
```

- Go to gitlab dir

```
cd /home/git/gitlab
```

Note: You can change 7-1-stable to master if you want the bleeding edge version, but never install master on a production server!

Configure GitLab

`cd /home/git/gitlab`

- Copy the example GitLab config

```
sudo -u git -H cp config/gitlab.yml.example config/gitlab.yml
```

Make sure to change "localhost" to the fully-qualified domain name of your host serving GitLab where necessary.
If you want to use https make sure that you set `https` to `true`. See #using-https for all necessary details.

- If you installed Git from source, change the git bin_path to /usr/local/bin/git

```
sudo -u git -H editor config/gitlab.yml
```

- Make sure GitLab can write to the log/ and tmp/ directories

```
sudo chown -R git log/
sudo chown -R git tmp/
sudo chmod -R u+rwX log/
sudo chmod -R u+rwX tmp/
```

- Create directory for satellites

```
sudo -u git -H mkdir /home/git/gitlab-satellites
sudo chmod u+rwx,g=rx,o-rwx /home/git/gitlab-satellites
```

- Make sure GitLab can write to the tmp/pids/ and tmp/sockets/ directories

```
sudo chmod -R u+rwX tmp/pids/
sudo chmod -R u+rwX tmp/sockets/
```

- Make sure GitLab can write to the public/uploads/ directory

```
sudo chmod -R u+rwX  public/uploads
```

- Copy the example Unicorn config

```
sudo -u git -H cp config/unicorn.rb.example config/unicorn.rb
```

*Enable cluster mode if you expect to have a high load instance (ex. change amount of workers to 3 for 2GB RAM server)

```
sudo -u git -H editor config/unicorn.rb
```

- Copy the example Rack attack config

```
sudo -u git -H cp config/initializers/rack_attack.rb.example config/initializers/rack_attack.r
b
```

- Configure Git global settings for git user, useful when editing via web. Edit user.email according to what is set in gitlab.yml

```
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
```

Important Note: Make sure to edit both gitlab.yml and unicorn.rb to match your setup.

Configure GitLab DB settings

- PostgreSQL only:

```
sudo -u git cp config/database.yml.postgresql config/database.yml
```

- MySQL only:

```
sudo -u git cp config/database.yml.mysql config/database.yml
```

MySQL and remote PostgreSQL only:
Update username/password in config/database.yml.
You only need to adapt the production settings (first part).
If you followed the database guide then please do as follows:
Change 'secure password' with the value you have given to $password

- You can keep the double quotes around the password

```
sudo -u git -H editor config/database.yml
```

- PostgreSQL and MySQL: Make config/database.yml readable to git only

```
sudo -u git -H chmod o-rwx config/database.yml
```

Install Gems

Note: As of bundler 1.5.2, you can invoke bundle install -jN (where N the number of your processor cores) and enjoy the parallel gems installation with measurable difference in completion time (~60% faster). Check the number of your cores with nproc. For more information check this post. First make sure you have bundler >= 1.5.2 (run bundle -v) as it addresses some issues that were fixed in 1.5.2.

```
cd /home/git/gitlab
```

- For PostgreSQL (note, the option says "without ... mysql")

```
sudo -u git -H bundle install --deployment --without development test mysql aws
```

- Or if you use MySQL (note, the option says "without ... postgres")

```
sudo -u git -H bundle install --deployment --without development test postgres aws
```

Install GitLab shell

GitLab Shell is an ssh access and repository management software developed specially for GitLab.

- Go to the Gitlab installation folder:

```
cd /home/git/gitlab
```

- Run the installation task for gitlab-shell (replace `REDIS_URL` if needed):

```
sudo -u git -H bundle exec rake gitlab:shell:install[v1.9.6] REDIS_URL=redis://localhost:6379
RAILS_ENV=production
```

By default, the gitlab-shell config is generated from your main gitlab config.

Note: When using GitLab with HTTPS please change the following:

1. Provide paths to the certificates under `ca_file` and `ca_path` options.
2. The `gitlab_url` option must point to the https endpoint of GitLab.
3. In case you are using self signed certificate set `self_signed_cert` to `true`.

- You can review (and modify) the gitlab-shell config as follows:

```
sudo -u git -H editor /home/git/gitlab-shell/config.yml
```

- Initialize Database and Activate Advanced Features

```
sudo -u git -H bundle exec rake gitlab:setup RAILS_ENV=production
```

  1. Type 'yes' to create the database tables.
  2. When done you see 'Administrator account created:'

Install Init Script

- Download the init script (will be /etc/init.d/gitlab):

```
sudo cp lib/support/init.d/gitlab /etc/init.d/gitlab
```

- And if you are installing with a non-default folder or user copy and edit the defaults file:

```
sudo cp lib/support/init.d/gitlab.default.example /etc/default/gitlab
```

If you installed gitlab in another directory or as a user other than the default you should change these settings in /etc/default/gitlab. Do not edit /etc/init.d/gitlab as it will be changed on upgrade.

- Make GitLab start on boot:

```
sudo update-rc.d gitlab defaults 21
```

- Set up logrotate

```
sudo cp lib/support/logrotate/gitlab /etc/logrotate.d/gitlab
```

Check Application Status

- Check if GitLab and its environment are configured correctly:

```
sudo -u git -H bundle exec rake gitlab:env:info RAILS_ENV=production
```

- Compile assets

```
sudo -u git -H bundle exec rake assets:precompile RAILS_ENV=production
```

- Start Your GitLab Instance

```
sudo service gitlab start
```

  1. or

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

Nginx

Note: Nginx is the officially supported web server for GitLab. If you cannot or do not want to use Nginx as your web server, have a look at the GitLab recipes.

- Install nginx

```
sudo apt-get install -y nginx
```

Site Configuration

- Copy the example site config:

```
sudo cp lib/support/nginx/gitlab /etc/nginx/sites-available/gitlab
sudo ln -s /etc/nginx/sites-available/gitlab /etc/nginx/sites-enabled/gitlab
```

Make sure to edit the config file to match your setup:

1. Change YOUR_SERVER_FQDN to the fully-qualified
2. domain name of your host serving GitLab.

```
sudo editor /etc/nginx/sites-available/gitlab
```

Note: If you want to use https, replace the gitlab nginx config with gitlab-ssl. See Using HTTPS for all necessary details.

Restart nginx

```
sudo service nginx restart
```

Done!

Double-check Application Status

- To make sure you didn't miss anything run a more thorough check with:

```
sudo -u git -H bundle exec rake gitlab:check RAILS_ENV=production
```

If all items are green, then congratulations on successfully installing GitLab!

NOTE: Supply SANITIZE=true environment variable to gitlab:check to omit project names from the output of the check command.

Initial Login

Visit YOUR_SERVER in your web browser for your first GitLab login. The setup has created an admin account for you. You can use it to log in:

```
root
5iveL!fe
```

Important Note: Please go over to your profile page and immediately change the password, so nobody can access your GitLab by using this login information later on.

Enjoy!

Advanced Setup Tips

Using HTTPS

- First, create a strong SSL key and CSR:

```
cd /etc/nginx/ssl
openssl req -sha512 -out git.example.com.csr -new -newkey rsa:4096 -nodes -keyout git.example.com.key
```

To recap what is needed to use GitLab with HTTPS:

1. In gitlab/gitlab.yml:
   - Change port to 443
   - Change https to true
2. In the gitlab-shell/config.yml use this new url and give it the certificate of the root CA that signed the servers certificate
   - gitlab_url: "https://gitlab.example.com/"
   - ca_file: /etc/nginx/pki/ca/Root_CA.crt
3. Use the gitlab-ssl nginx example config instead of the gitlab config

```
rm /etc/nginx/sites-available/gitlab
cd /home/git/gitlab
cp lib/support/nginx/gitlab-ssl /etc/nginx/sites-available/gitlab
```

   - Edit the /etc/nginx/sites-available/gitlab file and change the following paramaters
     1. server_name: git.example.com;
     2. ssl_certificate /etc/nginx/ssl/gitlab.crt;
     3. ssl_certificate_key /etc/nginx/ssl/gitlab.key;

NOTE: The ssl_certificate should be a chained certificate, this means that if you have a Intermediate CA (which you should have) you will need to

```
cat gitlab.domain.crt intermeiate_ca.crt > gitlab.domain.chained.crt
```

- Now it's time to restart Gitlab/nginx and check that it's all working:

```
sudo service gitlab restart
```

```
sudo service nginx restart
cd /home/git/gitlab
sudo -u git -H bundle exec rake gitlab:check RAILS_ENV=production
```

You should have a working Gitlab instance running over HTTPS. You shouldn't get any certificate errors whatsoever, if you do it most likely means that you haven't imported the Root CA's certificate into your OS's key store or you haven't trusted it. Remember Firefox has its own certificate store, if that what you're using, and you'll almost definitely need to restart your browser after installing certificates.

Additional markup styles

Apart from the always supported markdown style there are other rich text files that GitLab can display. But you might have to install a dependency to do so. Please see the github-markup gem readme for more information.

Custom Redis Connection

If you'd like Resque to connect to a Redis server on a non-standard port or on a different host, you can configure its connection string via the config/resque.yml file.

- Example

  production: redis://redis.example.tld:6379

If you want to connect the Redis server via socket, then use the "unix:" URL scheme and the path to the Redis socket file in the config/resque.yml file.

- Example

  production: unix:/path/to/redis/socket

Custom SSH Connection

If you are running SSH on a non-standard port, you must change the gitlab user's SSH config.

- Add to /home/git/.ssh/config

```
host localhost          # Give your setup a name (here: override localhost)
    user git            # Your remote git user
    port 2222           # Your port number
    hostname 127.0.0.1; # Your server name or IP
```

You also need to change the corresponding options (e.g. ssh_user, ssh_host, admin_uri) in the config\gitlab.yml file.

LDAP authentication

You can configure LDAP authentication in config/gitlab.yml. Please restart GitLab after editing this file.

Using Custom Omniauth Providers

GitLab uses Omniauth for authentication and already ships with a few providers preinstalled (e.g. LDAP, GitHub, Twitter). But sometimes that is not enough and you need to integrate with other authentication solutions. For these cases you can use the Omniauth provider.
Steps

These steps are fairly general and you will need to figure out the exact details from the Omniauth provider's documentation.

1. Stop GitLab:

```
sudo service gitlab stop
```

2. Add the gem to your Gemfile:

```
gem "omniauth-your-auth-provider"
```

   - If you're using MySQL, install the new Omniauth provider gem by running the following command:

```
sudo -u git -H bundle install --without development test postgres --path vendor/bundle --no-deployment
```

   - If you're using PostgreSQL, install the new Omniauth provider gem by running the following command:

```
sudo -u git -H bundle install --without development test mysql --path vendor/bundle --no-deployment
```

3. These are the same commands you used in the Install Gems section with --path vendor/bundle --no-deployment instead of --deployment.
4. Start GitLab:

```
sudo service gitlab start
```

Examples

If you have successfully set up a provider that is not shipped with GitLab itself, please let us know.

You can help others by reporting successful configurations and probably share a few insights or provide warnings for common errors or pitfalls by sharing your experience in the public Wiki.

While we can't officially support every possible auth mechanism out there, we'd like to at least help those with special needs.

Resources

- https://github.com/gitlabhq/gitlabhq/blob/master/doc/install/installation.md

History

#1 - 08/06/2014 07:23 PM - Daniel Curtis

- Status changed from New to Resolved

- % Done changed from 0 to 90

#2 - 08/07/2014 08:48 AM - Daniel Curtis

- Description updated

#3 - 08/07/2014 09:19 AM - Daniel Curtis

- Description updated

#4 - 08/07/2014 03:16 PM - Daniel Curtis

- Status changed from Resolved to Closed

- % Done changed from 90 to 100

#5 - 02/15/2015 08:38 PM - Daniel Curtis

- Project changed from 98 to GNU/Linux Administration

- Category set to Source Code Management