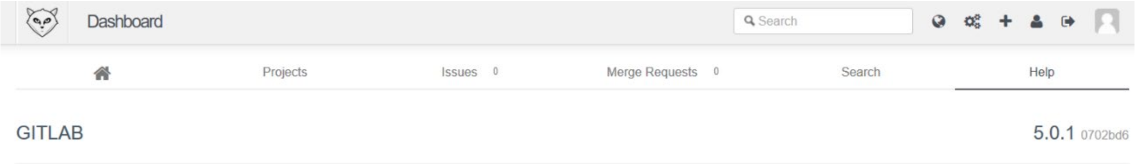
<table>
<tr><th colspan="5">Actualizacion de Gitlab 5.0.2 a 9.3.11</th></tr>
<tr><th>Version inicial</th><th>Version final</th><th>Tiempo empleado</th><th>Migrado</th><th>Ocupado</th></tr>
<tr><td>5.0.2</td><td>5.1.0</td><td>30 “</td><td>24/02/2019</td><td>23,7 %</td></tr>
<tr><td>5.1.0</td><td>6.0.2</td><td>30 ”</td><td>28/02/2019</td><td>23,9 %</td></tr>
<tr><td>6.0.2</td><td>7.14.3</td><td>50”</td><td>07/03/2019</td><td>25,3 %</td></tr>
<tr><td>7.14.3</td><td>8.0.5</td><td>30”</td><td>07/03/2019</td><td>26,0 %</td></tr>
<tr><td>8.0.5</td><td>8.1.4</td><td>10”</td><td>11/03/2019</td><td>22,3 %</td></tr>
<tr><td>8.1.4</td><td>8.2.6</td><td>30”</td><td>12/03/2019</td><td>21,6 %</td></tr>
<tr><td>8.2.6</td><td>8.3.10</td><td>30”</td><td>20/03/2019</td><td>21,7 %</td></tr>
<tr><td>8.3.10</td><td>8.4.11</td><td>10”</td><td>20/03/2019</td><td>22,5 %</td></tr>
<tr><td>8.4.11</td><td>8.5.13</td><td>20”</td><td>21/03/2019</td><td>22,8 %</td></tr>
<tr><td>8.5.13</td><td>8.6.9</td><td>10”</td><td>21/03/2019</td><td>22,8 %</td></tr>
<tr><td>8.6.9</td><td>8.7.9</td><td>25”</td><td>24/03/2019</td><td>23,0 %</td></tr>
<tr><td>8.7.9</td><td>8.8.9</td><td>20”</td><td>25/03/2019</td><td>23,1 %</td></tr>
<tr><td>8.8.9</td><td>8.9.11</td><td>20”</td><td>26/03/2019</td><td>23,4 %</td></tr>
<tr><td>8.9.11</td><td>8.10.13</td><td>30”</td><td>27/03/2019</td><td>23.6 %</td></tr>
<tr><td>8.10.13</td><td>8.11.11</td><td>30”</td><td>28/03/2019</td><td>24,3 %</td></tr>
<tr><td>8.11.11</td><td>8.12.13</td><td>30”</td><td>30/03/2019</td><td>24,4%</td></tr>
<tr><td>8.12.13</td><td>8.13.12</td><td>30”</td><td>30/03/2019</td><td>24,5 %</td></tr>
<tr><td>8.13.12</td><td>8.14.10</td><td>30”</td><td>31/03/2019</td><td>24,5%</td></tr>
<tr><td>8.14.10</td><td>8.15.8</td><td>30”</td><td>31/03/2019</td><td>24,6%</td></tr>
<tr><td>8.15.8</td><td>8.16.9</td><td>45”</td><td>01/04/2019</td><td>24,7%</td></tr>
<tr><td>8.16.9</td><td>8.17.8</td><td>45”</td><td>01/04/2019</td><td>24,8%</td></tr>
<tr><td>8.17.8</td><td>9.0.13</td><td>60”</td><td>02/04/2019</td><td>25,9%</td></tr>
<tr><td>9.0.13</td><td>9.3.11</td><td>45”</td><td>10/05/2019</td><td>26,2%</td></tr>
<tr><th colspan="5">Nuevo servidor ominubus</th></tr>
<tr><td>9.3.11</td><td>11.10.4</td><td>90”</td><td>03/05/2019</td><td>10%</td></tr>
</table>

Actualizar Git 5.0 a 5.1:

https://gitlab.com/gitlab-org/gitlab-ce/blob/11-8-stable/doc/update/5.0-to-5.1.md



1. Stop server

```
sudo service gitlab stop
```

2. Get latest code

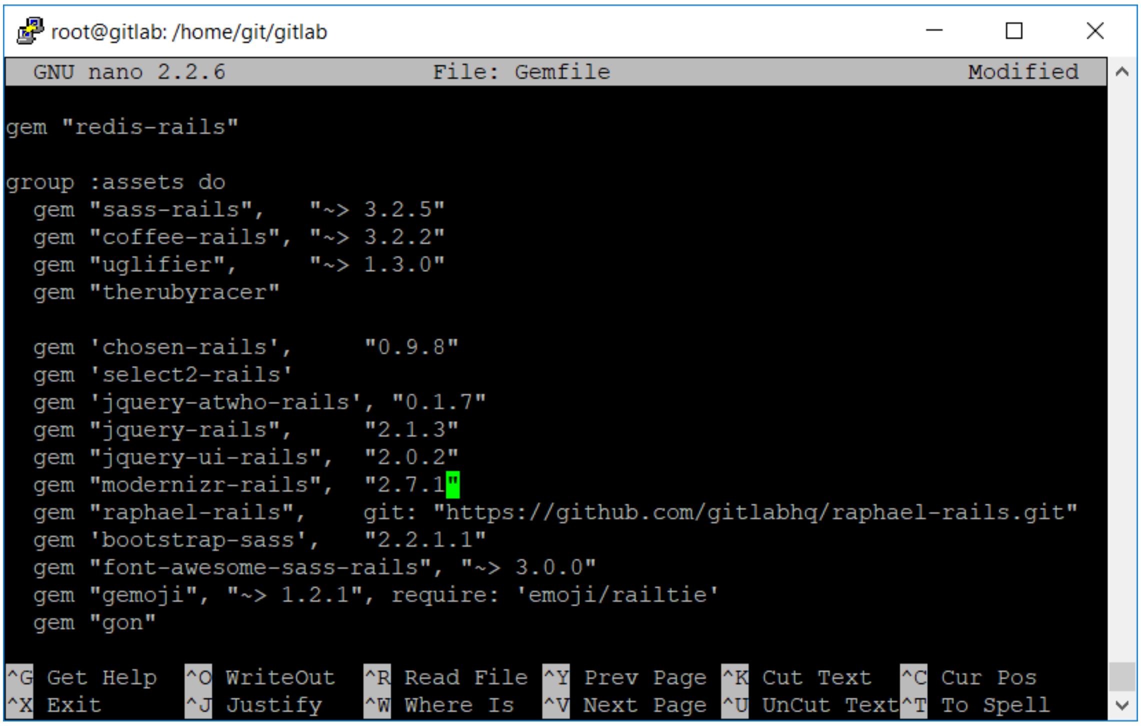
```
cd /home/git/gitlab
sudo -u git -H git fetch
git stash


#sin este comando no aplica los cambios

sudo -u git -H git checkout 5-1-stable
```

3. Update gitlab-shell

```
cd /home/git/gitlab-shell
sudo -u git -H git fetch
sudo -u git -H git checkout v1.3.0

# replace your old config with the new one

sudo -u git -H mv config.yml config.yml.old
sudo -u git -H cp config.yml.example config.yml

# edit options to match old config

sudo -u git -H nano config.yml
```

4. Install libs, migrations etc

```
cd /home/git/gitlab
sudo rm tmp/sockets/gitlab.socket
sudo -u git -H cp config/puma.rb.example config/puma.rb

# The Modernizr gem was yanked from RubyGems. It is required for
GitLab >= 2.8.0
# Edit `Gemfile` and change `gem "modernizr", "2.5.3"` to
# `gem "modernizr-rails", "2.7.1"``

sudo -u git -H nano Gemfile
```



```
root@gitlab: /home/git/gitlab
  GNU nano 2.2.6                  File: Gemfile                       Modified

gem "redis-rails"

group :assets do
  gem "sass-rails",    "~> 3.2.5"
  gem "coffee-rails",  "~> 3.2.2"
  gem "uglifier",      "~> 1.3.0"
  gem "therubyracer"

  gem 'chosen-rails',      "0.9.8"
  gem 'select2-rails'
  gem 'jquery-atwho-rails', "0.1.7"
  gem "jquery-rails",      "2.1.3"
  gem "jquery-ui-rails",   "2.0.2"
  gem "modernizr-rails",   "2.7.1"
  gem "raphael-rails",     git: "https://github.com/gitlabhq/raphael-rails.git"
  gem 'bootstrap-sass',    "2.2.1.1"
  gem "font-awesome-sass-rails", "~> 3.0.0"
  gem "gemoji", "~> 1.2.1", require: 'emoji/railtie'
  gem "gon"

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

```
# Run a bundle install without deployment to generate the new Gemfile

sudo -u root -H bundle install --without development test postgres --
no-deployment

# Install libs (with deployment this time)

sudo -u root -H bundle install --without development test postgres --
deployment

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

5. Update init.d script with a new one

```
# init.d
sudo rm /etc/init.d/gitlab
sudo curl --location --output /etc/init.d/gitlab
https://raw.github.com/gitlabhq/gitlab-recipes/5-1-
stable/init.d/gitlab
sudo chmod +x /etc/init.d/gitlab
```

6. MySQL grant privileges

```
mysql -u root -p
mysql> GRANT LOCK TABLES ON `gitlabhq_production`.* TO
'gitlab'@'localhost';
mysql> \q
```

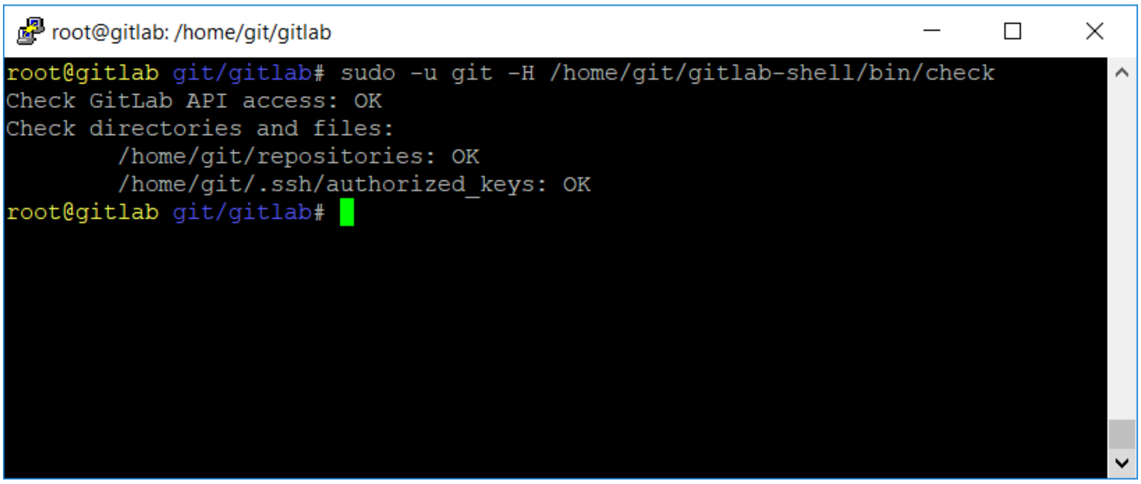
```
# quitar el comando mysql>
```

7. Start application

```
sudo service gitlab start
```

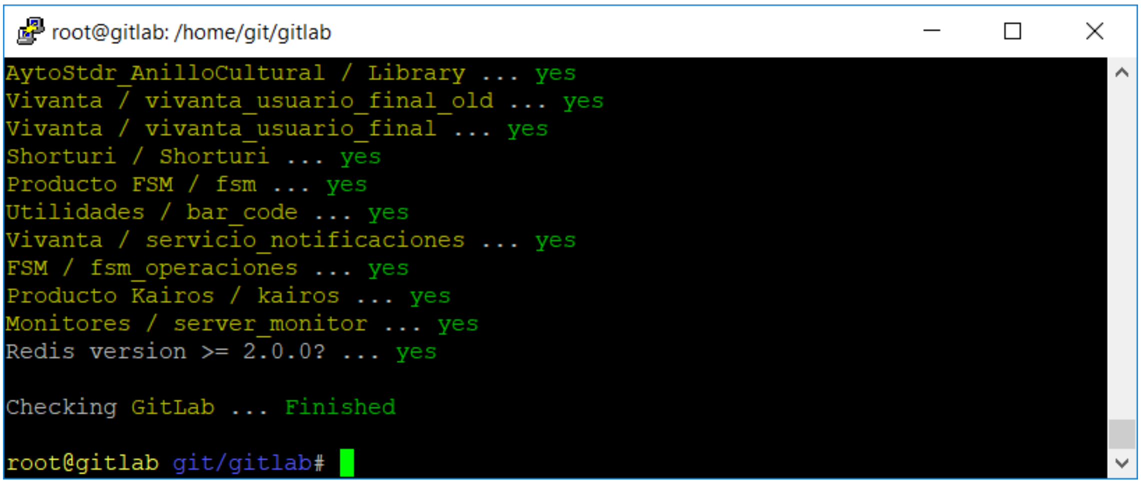
Check installation

```
# In 5-10 seconds lets check gitlab-shell

sudo -u git -H /home/git/gitlab-shell/bin/check
```



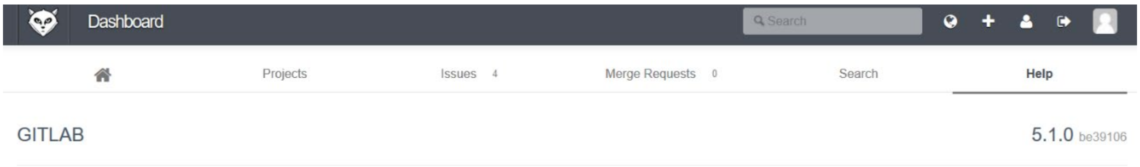
```
root@gitlab git/gitlab# sudo -u git -H /home/git/gitlab-shell/bin/check
Check GitLab API access: OK
Check directories and files:
        /home/git/repositories: OK
        /home/git/.ssh/authorized_keys: OK
root@gitlab git/gitlab#
```

```
# Example of success output
# Check GitLab API access: OK
# Check directories and files:
#         /home/git/repositories: OK
#         /home/git/.ssh/authorized_keys: OK


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



```
AytoStdr_AnilloCultural / Library ... yes
Vivanta / vivanta_usuario_final_old ... yes
Vivanta / vivanta_usuario_final ... yes
Shorturi / Shorturi ... yes
Producto FSM / fsm ... yes
Utilidades / bar_code ... yes
Vivanta / servicio_notificaciones ... yes
FSM / fsm_operaciones ... yes
Producto Kairos / kairos ... yes
Monitores / server_monitor ... yes
Redis version >= 2.0.0? ... yes

Checking GitLab ... Finished

root@gitlab git/gitlab#
```

Ya estamos en la 5.1.0

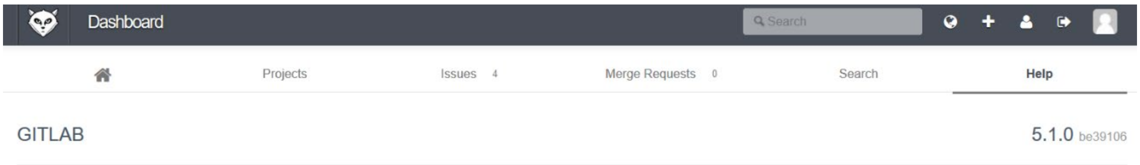


GITLAB

5.1.0 be39106

Actualizar git 5.1 a 6.0

https://gitlab.com/gitlab-org/gitlab-ce/blob/11-8-stable/doc/update/5.1-to-6.0.md



GITLAB 5.1.0 be39106

0. Backup & prepare for update

It's useful to make a backup just in case things go south: (With MySQL, this may require granting "LOCK TABLES" privileges to the GitLab user on the database version)

```
# por algún motivo no tengo la carpeta uploads y al hacer el Backup da error, asi que lo primero es crearla
```

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

```
sudo -u git -H mkdir -p public/uploads
sudo chmod -R u+rwX  public/uploads
```

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

The migrations in this update are very sensitive to incomplete or inconsistent data. If you have a long-running GitLab installation and some of the previous upgrades did not work out 100% correct this may bite you now. The following can help you have a more smooth upgrade.

Find projects with invalid project names

MySQL

Login to MySQL:

```
mysql -u root -p
```

Find projects with invalid names:

```
mysql> use gitlab_production;   → Quitar gitlabhq_production;
```

```
# find projects with invalid first char, projects must start with letter
```

```
mysql> select name from projects where name REGEXP '^[^A-Za-z]';
```

```
# find projects with other invalid chars
```

```
## names must only contain alphanumeric chars, underscores, spaces, periods, and dashes
```

```
mysql> select name from projects where name REGEXP '[^a-zA-Z0-9_ .-
]+';
```

```
If any projects have invalid names try correcting them from the web
interface before starting the upgrade. If correcting them from the web
interface fails you can correct them using MySQL:

# e.g. replace invalid / with allowed _

# no hay errores en las tablas ☺


mysql> update projects set name = REPLACE(name,'/','_');

# repeat for all invalid chars found in project names

# Salir de la base de datos

\q
```

PostgreSQL

Make sure all project names start with a letter and only contain alphanumeric chars, underscores, spaces, periods, and dashes (a-zA-Z0-9_ .-).

Find other common errors

```
cd /home/git/gitlab

# Start rails console

sudo -u git -H bin/rails console production

# Make sure none of the following rails commands return results. All
project owners should have an owner:

Project.all.select { |project| project.owner.blank? }

# Every user should have a namespace:

User.all.select { |u| u.namespace.blank? }

# Projects in the global namespace should not conflict with projects
in the owner namespace:
Project.where(namespace_id: nil).select { |p| Project.where(path:
p.path, namespace_id: p.owner.try(:namespace).try(:id)).present? }
```

If any of the above rails commands returned results other than `=> []` try correcting the issue from the web interface. If you find projects without an owner (first rails command above), correct it. For MySQL setups:

```
# get your user id

mysql> select id, name from users order by name;

# set yourself as owner of project

# replace your_user_id with your user id and bad_project_id with the
project id from the rails command

mysql> update projects set creator_id=your_user_id where
id=bad_project_id;
```

Actualizar git 6.0 a 6.0.2

https://gitlab.com/gitlab-org/gitlab-ce/blob/11-8-stable/doc/update/5.1-to-6.0.md

1. Stop server

```
sudo service gitlab stop
```

2. Get latest code

```
cd /home/git/gitlab
sudo -u git -H git fetch
```

```
git stash
```

```
#sin este comando no aplica los cambios
```

```
sudo -u git -H git checkout 6-0-stable
```

3. Update gitlab-shell

```
cd /home/git/gitlab-shell
sudo -u git -H git fetch
sudo -u git -H git checkout v1.7.9
```

4. Install additional packages

```
# For reStructuredText markup language support install required
package:
```

```
sudo apt-get install python-docutils
```

5. Install libs, migrations, etc.

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

```
# The Modernizr gem was yanked from RubyGems. It is required for
GitLab >= 2.8.0
```

```
# Edit `Gemfile` and change `gem "modernizr", "2.5.3"` to
```

```
`gem "modernizr-rails", "2.7.1"``
```

```
sudo -u git -H nano Gemfile
```

```
# MySQL
```

```
# Run a bundle install without deployment to generate the new Gemfile
```

```
sudo -u git -H bundle install --without development test postgres --
no-deployment      → no funciona!!
```

```
git stash
passwd -l git
```

```
# sudo -u root -H bundle install --full-index

sudo -u git -H bundle install --full-index


# Install libs (with deployment this time)

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

# Both MySQL and PostgreSQL


sudo -u git -H bundle exec rake db:migrate RAILS_ENV=production
sudo -u git -H bundle exec rake migrate_groups RAILS_ENV=production
sudo -u git -H bundle exec rake migrate_global_projects RAILS_ENV=production
sudo -u git -H bundle exec rake migrate_keys RAILS_ENV=production
sudo -u git -H bundle exec rake migrate_inline_notes RAILS_ENV=production
sudo -u git -H bundle exec rake gitlab:satellites:create RAILS_ENV=production

# Clear redis cache

sudo -u git -H bundle exec rake cache:clear RAILS_ENV=production

# Clear and precompile assets

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

#Add dealing with newlines for editor

sudo -u git -H git config --global core.autocrlf input
```

6. Update config files

```
#Note: We switched from Puma in GitLab 5.x to unicorn in GitLab 6.0.

cd /home/git/gitlab/config/
mv gitlab.yml gitlab.yml.old
mv unicorn.rb unicorn.rb.old

#Make /home/git/gitlab/config/gitlab.yml the same as https://gitlab.com/gitlab-org/gitlab-ce/blob/6-0-stable/config/gitlab.yml.example but with your settings.

sudo nano /home/git/gitlab/config/gitlab.yml

#Make /home/git/gitlab/config/unicorn.rb the same as https://gitlab.com/gitlab-org/gitlab-ce/blob/6-0-stable/config/unicorn.rb.example but with your settings.

sudo nano /home/git/gitlab/config/unicorn.rb
```

7. Update Init script

```
cd /home/git/gitlab
sudo rm /etc/init.d/gitlab
sudo cp lib/support/init.d/gitlab /etc/init.d/gitlab
sudo chmod +x /etc/init.d/gitlab
```

8. Create uploads directory

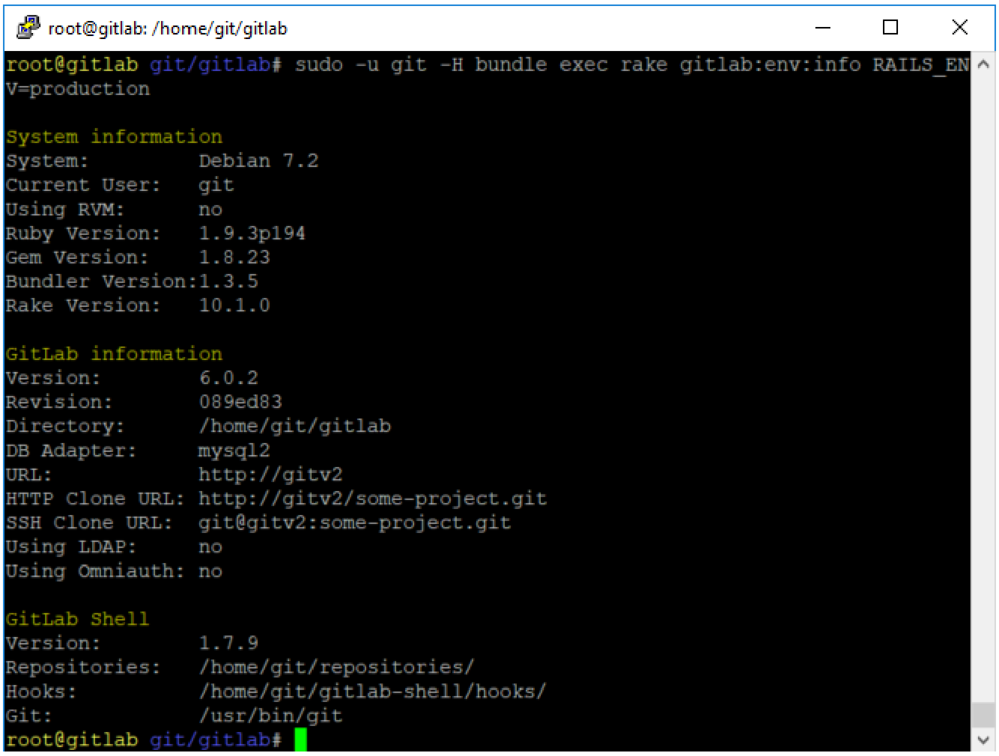
```
cd /home/git/gitlab
sudo -u git -H mkdir -p public/uploads
sudo chmod -R u+rwX  public/uploads
```

9. Start application

```
sudo service gitlab start
sudo service nginx restart
```

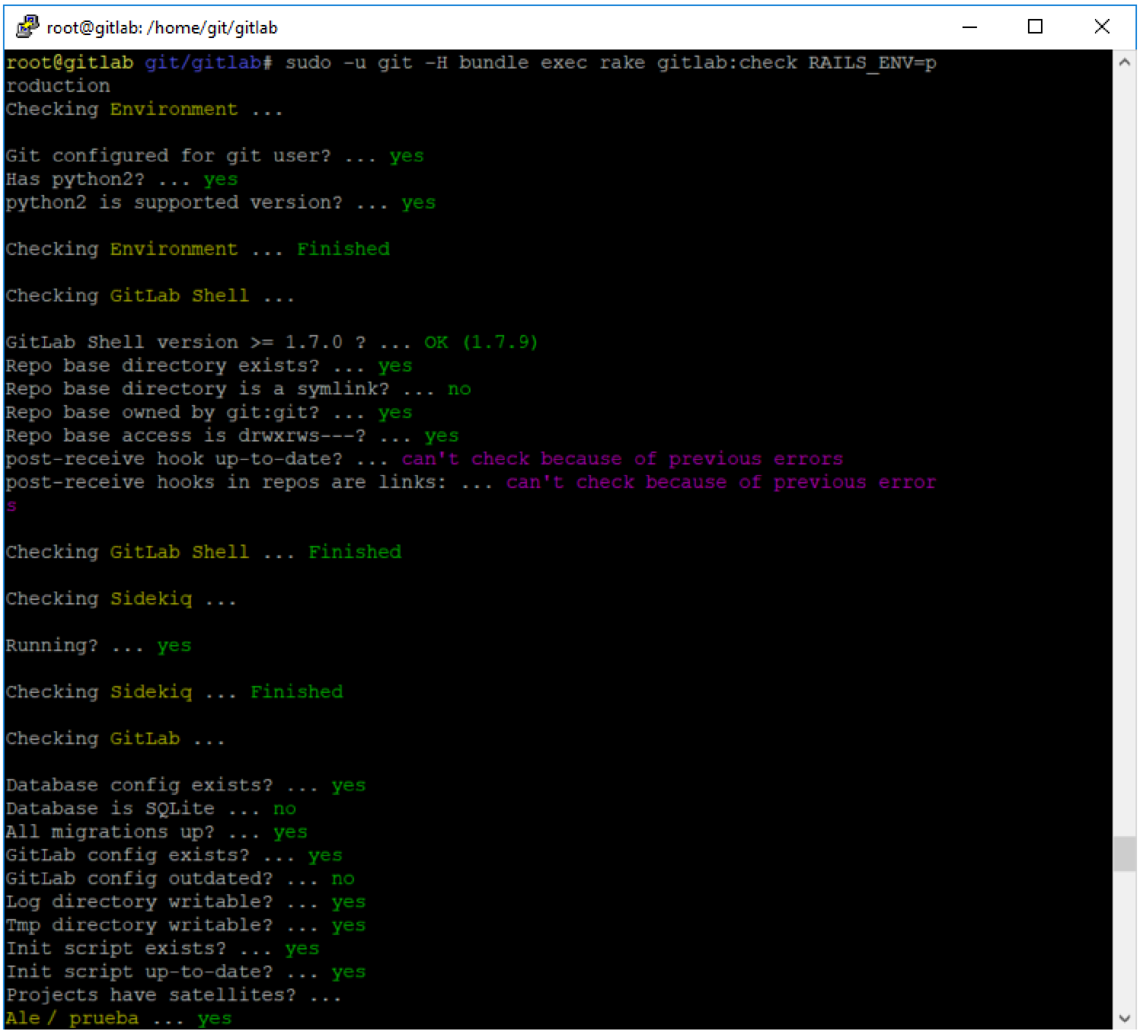
10. Check application status

```
# Check if GitLab and its environment are configured correctly:

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



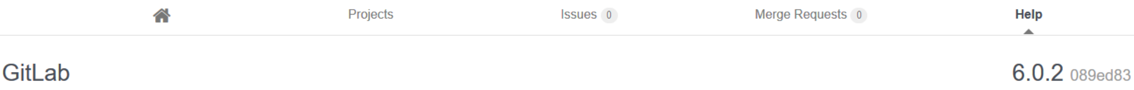
```
root@gitlab git/gitlab# sudo -u git -H bundle exec rake gitlab:env:info RAILS_EN
V=production

System information
System:         Debian 7.2
Current User:   git
Using RVM:      no
Ruby Version:   1.9.3p194
Gem Version:    1.8.23
Bundler Version:1.3.5
Rake Version:   10.1.0

GitLab information
Version:        6.0.2
Revision:       089ed83
Directory:      /home/git/gitlab
DB Adapter:     mysql2
URL:            http://gitv2
HTTP Clone URL: http://gitv2/some-project.git
SSH Clone URL:  git@gitv2:some-project.git
Using LDAP:     no
Using Omniauth: no

GitLab Shell
Version:        1.7.9
Repositories:   /home/git/repositories/
Hooks:          /home/git/gitlab-shell/hooks/
Git:            /usr/bin/git
root@gitlab git/gitlab#
```

```
# 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
```



```
root@gitlab git/gitlab# sudo -u git -H bundle exec rake gitlab:check RAILS_ENV=p
roduction
Checking Environment ...

Git configured for git user? ... yes
Has python2? ... yes
python2 is supported version? ... yes

Checking Environment ... Finished

Checking GitLab Shell ...

GitLab Shell version >= 1.7.0 ? ... OK (1.7.9)
Repo base directory exists? ... yes
Repo base directory is a symlink? ... no
Repo base owned by git:git? ... yes
Repo base access is drwxrws---? ... yes
post-receive hook up-to-date? ... can't check because of previous errors
post-receive hooks in repos are links: ... can't check because of previous error
s

Checking GitLab Shell ... Finished

Checking Sidekiq ...

Running? ... yes

Checking Sidekiq ... Finished

Checking GitLab ...

Database config exists? ... yes
Database is SQLite ... no
All migrations up? ... yes
GitLab config exists? ... yes
GitLab config outdated? ... no
Log directory writable? ... yes
Tmp directory writable? ... yes
Init script exists? ... yes
Init script up-to-date? ... yes
Projects have satellites? ...
Ale / prueba ... yes
```

If all items are green, then congratulations upgrade complete!



GitLab 6.0.2 089ed83

Actualizar git 6.0.2 a 7.14

https://gitlab.com/gitlab-org/gitlab-ce/blob/11-8-stable/doc/update/6.x-or-7.x-to-7.14.md

Global issue numbers

As of 6.1 issue numbers are project specific. This means all issues are renumbered and get a new number in their URL. If you use an old issue number URL and the issue number does not exist yet you are redirected to the new one. This conversion does not trigger if the old number already exists for this project, this is unlikely but will happen with old issues and large projects.

Editable labels

In GitLab 7.2 we replace Issue and Merge Request tags with labels, making it possible to edit the label text and color. The characters `?`, `&` and `,` are no longer allowed however so those will be removed from your tags during the database migrations for GitLab 7.2.

Stash changes

If you deleted the vendors folder during your original installation, you will get an error when you attempt to rebuild the assets in step 7. To avoid this, stash the changes in your GitLab working copy before starting:

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

0. Stop server

```
sudo service gitlab stop
```

1. Backup

```
# It's useful to make a backup just in case things go south: (With
MySQL, this may require granting "LOCK TABLES" privileges to the
GitLab user on the database version)
```

```
cd /home/git/gitlab
sudo -u git -H bundle exec rake gitlab:backup:create
RAILS_ENV=production
```

2. Update Ruby

If you are still using Ruby 1.9.3 or below, you will need to update Ruby. You can check which version you are running with `ruby -v`.

```
# If you are you running Ruby 2.0.x, you do not need to upgrade ruby,
but can consider doing so for performance reasons.
If you are running Ruby 2.1.1 consider upgrading to 2.1.6, because of
the high memory usage of Ruby 2.1.1.
Install, update dependencies:

sudo apt-get install build-essential zlib1g-dev libyaml-dev libssl-dev
libgdbm-dev libreadline-dev libncurses5-dev libffi-dev curl

# Download and compile Ruby:
mkdir /tmp/ruby && cd /tmp/ruby
curl --progress https://cache.ruby-lang.org/pub/ruby/2.1/ruby-
2.1.6.tar.gz | tar xz
cd ruby-2.1.6
./configure --disable-install-rdoc
make
sudo make install

# Install Bundler:
sudo gem install bundler --no-document --version '< 2'
```

3. Get latest code

```
cd /home/git/gitlab
sudo -u git -H git fetch --all
sudo -u git -H git checkout -- db/schema.rb
```

```
# local changes will be restored automatically
```

```
sudo -u git -H git checkout 7-14-stable
```

4. Install additional packages

```
# Add support for logrotate for better log file handling
```

```
sudo apt-get install logrotate
```

```
# Install pkg-config and cmake, which is needed for the latest
versions of rugged
```

```
sudo apt-get install pkg-config cmake
```

```
# If you want to use Kerberos with GitLab EE for user authentication,
install Kerberos header files. If you don't know what Kerberos is, you
can assume you don't need it.
```

```
sudo apt-get install libkrb5-dev
```

```
# Install nodejs, javascript runtime required for assets
```

```
# sudo apt-get install nodejs  → no encuentra el instalador
```

```
curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -
sudo apt-get install -y nodejs
```

5. Configure Redis to use sockets

```
# Configure redis to use sockets
```

```
sudo cp /etc/redis/redis.conf /etc/redis/redis.conf.orig

# Disable Redis listening on TCP by setting 'port' to 0

sed 's/^port .*/port 0/' /etc/redis/redis.conf.orig | sudo tee
/etc/redis/redis.conf

# Enable Redis socket for default Debian / Ubuntu path

echo 'unixsocket /var/run/redis/redis.sock' | sudo tee -a
/etc/redis/redis.conf

# Be sure redis group can write to the socket, enable only if
supported (>= redis 2.4.0).

sudo sed -i '/# unixsocketperm/ s/^# unixsocketperm.*/unixsocketperm
0775/' /etc/redis/redis.conf

# Activate the changes to redis.conf

sudo service redis-server restart

# Add git to the redis group

sudo usermod -aG redis git

# Configure Redis connection settings

sudo -u git -H cp config/resque.yml.example config/resque.yml

# Change the Redis socket path if you are not using the default Debian
/ Ubuntu configuration

sudo -u git -H editor config/resque.yml

# Configure gitlab-shell to use Redis sockets

sudo -u git -H sed -i 's|^  # socket.*|  socket:
/var/run/redis/redis.sock|' /home/git/gitlab-shell/config.yml
```

6. Update gitlab-shell

```
cd /home/git/gitlab-shell
sudo -u git -H git fetch
sudo -u git -H git checkout v2.6.5
```

7. Install libs, migrations, etc.

```
cd /home/git/gitlab

# antes de comenzar hay que modificar el fichero Gemfile.lock para
evitar el Error:

Downloading gollum-lib-4.0.2 revealed dependencies not in the API or
the lockfile (rouge (~> 1.7.4)).
Either installing with `--full-index` or running `bundle update
gollum-lib` should fix the problem.

sudo -u git -H nano Gemfile.lock
```

```
# Buscar (rouge (~> 1.X.X)). Y modificiarlo a (rouge (~> 1.7.4)). Esta
en dos sitios.

gollum-lib (4.0.2)
      github-markup (~> 1.3.1)
      gollum-grit_adapter (~> 0.1, >= 0.1.1)
      nokogiri (~> 1.6.4)
      rouge (~> 1.9)

rest-client (1.8.0)
      http-cookie (>= 1.0.2, < 2.0)
      mime-types (>= 1.16, < 3.0)
      netrc (~> 0.7)
    rinku (1.7.3)
    rotp (1.6.1)
    rouge (1.9.1)


# MySQL installations (note: the line below states '--without ...
postgres')


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


# Run database migrations from 6.0 to 6.1

sudo -u git -H bundle exec rake db:migrate RAILS_ENV=production
VERSION=20130909132950

# Enable internal issue IDs (introduced in GitLab 6.1)

sudo -u git -H bundle exec rake migrate_iids RAILS_ENV=production

# Run left database migrations

sudo -u git -H bundle exec rake db:migrate RAILS_ENV=production


# Clean up assets and cache

sudo -u git -H bundle exec rake assets:clean assets:precompile
cache:clear RAILS_ENV=production

# Close access to gitlab-satellites for others

sudo chmod u+rwx,g+rx,o-rwx /home/git/gitlab-satellites

# Update init.d script

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

8. Update config files

```
# TIP: to see what changed in gitlab.yml.example in this release use
next command:

git diff 6-0-stable:config/gitlab.yml.example 7-14-
stable:config/gitlab.yml.example
```

# Make /home/git/gitlab/config/gitlab.yml the same as https://gitlab.com/gitlab-org/gitlab-ce/blob/7-14-stable/config/gitlab.yml.lsexample but with your settings.

```
cd /home/git/gitlab/config
mv gitlab.yml gitlab.old
sudo nano gitlab.yml
```

# Make /home/git/gitlab/config/unicorn.rb the same as https://gitlab.com/gitlab-org/gitlab-ce/blob/7-14-stable/config/unicorn.rb.example but with your settings.

```
   cd /home/git/gitlab/config
   mv unicorn.rb unicorn.rb.old
   sudo nano unicorn.rb
```

# Make /home/git/gitlab-shell/config.yml the same as https://gitlab.com/gitlab-org/gitlab-shell/blob/v2.6.5/config.yml.example but with your settings.

```
   cd /home/git/gitlab-shell
   mv config.yml config.yml.old
   sudo -u git -H nano config.yml
```

# Copy rack attack middleware config.

```
cd /home/git/gitlab
sudo -u git -H cp config/initializers/rack_attack.rb.example config/initializers/rack_attack.rb
```

# Set up logrotate

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

# Change Nginx settings

# HTTP setups: Make /etc/nginx/sites-available/gitlab the same as https://gitlab.com/gitlab-org/gitlab-ce/blob/7-14-stable/lib/support/nginx/gitlab but with your settings.

```
   cd /etc/nginx/sites-available
   mv gitlab gitlab.old
   sudo nano gitlab
```

# hay que cambiar esta línea del archivo por defecto:

```
listen [::]:80 ipv6only=on default_server;
```

# asi evito el error: Restarting nginx: nginx: [emerg] bind() to [::]:80 failed (98: Address already in use)

# HTTPS setups: Make /etc/nginx/sites-available/gitlab-ssl the same as https://gitlab.com/gitlab-org/gitlab-ce/blob/7-14-stable/lib/support/nginx/gitlab-ssl but with your settings.

```
   sudo nano /etc/nginx/sites-available/gitlab-ssl
```

```
# A new location /uploads/ section has been added that needs to have
the same content as the existing location @gitlab section.
```

Check the version of /usr/local/bin/git
If you installed Git from source into /usr/local/bin/git then please
check your version.

9. Start application

```
sudo service gitlab start
sudo service nginx restart
```

10. Check application status

```
# Check if GitLab and its environment are configured correctly:
cd /home/git/gitlab
sudo -u git -H bundle exec rake gitlab:env:info RAILS_ENV=production

# 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 upgrade complete!
```

11. Update OmniAuth configuration

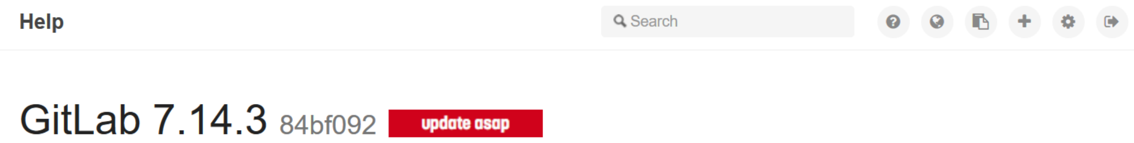
# When using Google omniauth login, changes of the Google account required. Ensure that Contacts API and the Google+ API are enabled in the **Google Developers Console**. More details can be found at the **integration documentation**.

12. Optional optimizations for GitLab setups with MySQL databases

Only applies if running MySQL database created with GitLab 6.7 or earlier. If you are not experiencing any issues you may not need the following instructions however following them will bring your database in line with the latest recommended installation configuration and help avoid future issues. Be sure to follow these directions exactly. These directions should be safe for any MySQL instance but to be sure make a current MySQL database backup beforehand.

```
# Stop GitLab

sudo service gitlab stop

# Secure your MySQL installation (added in GitLab 6.2)

sudo mysql_secure_installation

# Login to MySQL

mysql -u root -p
```

```
# Convert all tables to use the InnoDB storage engine (added in GitLab 6.8)

SELECT CONCAT('ALTER TABLE gitlab_production.', table_name, ' ENGINE=InnoDB;') AS 'Copy & run these SQL statements:' FROM information_schema.tables WHERE table_schema = 'gitlab_production' AND `ENGINE` <> 'InnoDB' AND `TABLE_TYPE` = 'BASE TABLE';

# If previous query returned results, copy & run all shown SQL statements. Convert all tables to correct character set

SET foreign_key_checks = 0;

SELECT CONCAT('ALTER TABLE gitlab_production.', table_name, ' CONVERT TO CHARACTER SET utf8 COLLATE utf8_general_ci;') AS 'Copy & run these SQL statements:' FROM information_schema.tables WHERE table_schema = 'gitlab_production' AND `TABLE_COLLATION` <> 'utf8_unicode_ci' AND `TABLE_TYPE` = 'BASE TABLE';

# If previous query returned results, copy & run all shown SQL statements. Turn foreign key checks back on

SET foreign_key_checks = 1;

# Find MySQL users

SELECT user FROM mysql.user WHERE user LIKE '%git%';


# If both users exist skip to Delete gitlab user
# Create new user for GitLab (changed in GitLab 6.4)
# change $password in the command below to a real password you pick

CREATE USER 'git'@'localhost' IDENTIFIED BY '$password';


# Grant the git user necessary permissions on the database

GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER, LOCK TABLES ON `gitlab_production`.* TO 'git'@'localhost';

# Delete the old gitlab user

DELETE FROM mysql.user WHERE user='gitlab';

# Quit the database session

\q

# Try connecting to the new database with the new user

sudo -u git -H mysql -u git -p -D gitlab_production

# Type the password you replaced $password with earlier
# You should now see a 'mysql>' prompt
# Quit the database sesión

\q

# Update database configuration details
# See config/database.yml.mysql for latest recommended configuration details
#   Remove the reaping_frequency setting line if it exists (removed in GitLab 6.8)
#   Set production -> pool: 10 (updated in GitLab 5.3)
#   Set production -> username: git
#   Set production -> password: the password your replaced $password with earlier

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

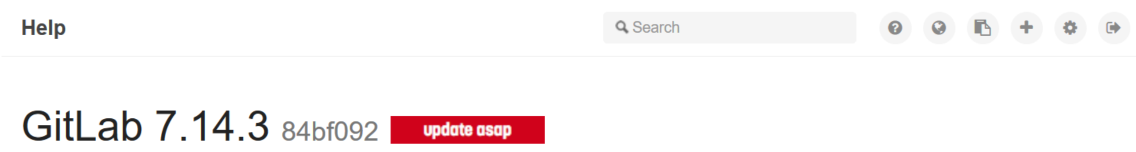


GitLab 7.14.3 84bf092

Desde esta versión no puedo publicar en gitlab -> Se corrije al actualizar a la versión 8.0.5

Actualizar Git de 7.14 a 8.0

https://github.com/turnkeylinux/tracker/wiki/Upgrade-GitLab-in-v14.0-and-v14.1-appliances#gitlab-714x-but-80x



1. Stop server

```
sudo service gitlab stop
```

2.Backup

```
sudo service gitlab stop
cd /home/git/gitlab
sudo -u git -H bundle exec rake gitlab:backup:create
RAILS_ENV=production
```

3.Get the latest GitLab code

```
sudo service gitlab stop
cd /home/git/gitlab

# error: insufficient permission for adding an object to repository
database .git/objects

sudo chown -R git:staff .git

sudo -u git -H git fetch --all
sudo -u git -H git checkout -- db/schema.rb
git stash
sudo -u git -H git checkout 8-0-stable
```

4.Update gitlab-shell

```
cd /home/git/gitlab-shell
sudo -u git -H git fetch
sudo -u git -H git checkout v2.6.5
```

5.Install gitlab-git-http-server

```
cd /opt/
wget http://storage.googleapis.com/golang/go1.5.linux-amd64.tar.gz
sudo tar -C /usr/local -xzf go1.5.linux-amd64.tar.gz
sudo ln -sf /usr/local/go/bin/{go,godoc,gofmt} /usr/local/bin/
rm -rf go1.5.linux-amd64.tar.gz
```

5.1.Gitlab-git-http-server

```
cd /home/git
sudo -u git -H git clone https://gitlab.com/gitlab-org/gitlab-git-http-server.git
cd gitlab-git-http-server
git stash
sudo -u git -H git checkout 0.2.14
sudo -u git -H make

cd /home/git/gitlab
sudo -u git tee -a config/unicorn.rb <<EOF
listen "127.0.0.1:8080", :tcp_nopush => true
EOF
```

6.Copy secrets

```
cd /home/git/gitlab
sudo -u git -H cp config/secrets.yml.example config/secrets.yml
sudo -u git -H chmod 0600 config/secrets.yml
```

7.Install libs, migrations, etc.

```
cd /home/git/gitlab

# antes de comenzar hay que modificar el fichero Gemfile.lock para
evitar el Error:
```

Downloading gollum-lib-4.0.2 revealed dependencies not in the API or the lockfile (rouge (~> 1.7.4)). Either installing with `--full-index` or running `bundle update gollum-lib` should fix the problem.

```
sudo -u git -H nano Gemfile.lock
```

# Buscar **(rouge (~> 1.X.X)).** Y modificiarlo a **(rouge (~> 1.7.4)).** Esta en dos sitios.

```
gollum-lib (4.0.2)
    github-markup (~> 1.3.1)
    gollum-grit_adapter (~> 0.1, >= 0.1.1)
    nokogiri (~> 1.6.4)
    rouge (~> 1.9)

rest-client (1.8.0)
    http-cookie (>= 1.0.2, < 2.0)
    mime-types (>= 1.16, < 3.0)
    netrc (~> 0.7)
  rinku (1.7.3)
  rotp (1.6.1)
  rouge (1.9.1)
```

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

```
sudo -u git -H bundle exec rake db:migrate RAILS_ENV=production
```

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

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

8.Update config files (revisar con los de la 7.14)

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

```
git diff origin/7-14-stable:config/gitlab.yml.example origin/8-0-
stable:config/gitlab.yml.example
```

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

9.Update nginx config:

```
cp lib/support/nginx/gitlab /etc/nginx/sites-available/gitlab
cp lib/support/nginx/gitlab-ssl /etc/nginx/sites-available/gitlab-ssl
cd /etc/nginx/sites-available
sudo nano gitlab
```

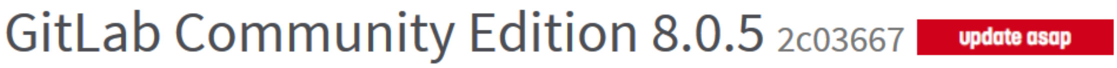
hay que añadir la # en el apartado listen [::]:80 default_server; para evitar problemas con nginx

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

```
cp config/gitlab.yml.example config/gitlab.yml
cp config/database.yml.mysql config/database.yml → te peta el usuario
y hay que editarlo
cd /home/git/gitlab-shell
cp config.yml.example config.yml
```

10.Restart GitLab and test

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


GitLab Community Edition 8.0.5 2c03667 update asap

Actualizar Git de 8.0.5 a 8.1.4

https://gitlab.com/gitlab-org/gitlab-ce/blob/11-8-stable/doc/update/8.0-to-8.1.md

GitLab Community Edition 8.0.5 2c03667 update asap

1. Stop server

```
sudo service gitlab stop
cd /home/git/gitlab
sudo -u git mkdir -m 750 /home/git/gitlab/public/uploads
```

2. Backup

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

3. Get latest code

```
sudo -u git -H git fetch --all
sudo -u git -H git checkout -- db/schema.rb
git stash
sudo -u git -H git checkout 8-1-stable
```

4. Update gitlab-shell

```
cd /home/git/gitlab-shell
sudo -u git -H git fetch
sudo -u git -H git checkout v2.6.5
```

5. Update gitlab-git-http-server

```
cd /home/git/gitlab-git-http-server
sudo -u git -H git fetch origin
sudo -u git -H git checkout 0.3.0
sudo -u git -H make
```

6. Install libs, migrations, etc.

```
cd /home/git/gitlab

# MySQL installations (note: the line below states '--without
postgres')

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

```
# Run database migrations

sudo -u git -H bundle exec rake db:migrate RAILS_ENV=production


# Clean up assets and cache

sudo -u git -H bundle exec rake assets:clean assets:precompile
cache:clear RAILS_ENV=production


# Update init.d script

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

7. Update configuration files

New configuration options for `gitlab.yml`

There are new configuration options available for `gitlab.yml`. View them with the command below and apply them manually to your current `gitlab.yml`:

```
git diff origin/8-0-stable:config/gitlab.yml.example origin/8-1-
stable:config/gitlab.yml.example
```

```
cd /home/git/gitlab/config
mv gitlab.yml gitlab.8.0.5
sudo nano gitlab.yml
```

Nginx configuration

View changes between the previous recommended Nginx configuration and the current one:

```
# For HTTPS configurations
git diff origin/8-0-stable:lib/support/nginx/gitlab-ssl origin/8-1-
stable:lib/support/nginx/gitlab-ssl

# For HTTP configurations
git diff origin/8-0-stable:lib/support/nginx/gitlab origin/8-1-
stable:lib/support/nginx/gitlab
```

If you are using Apache instead of NGINX please see the updated Apache templates. Also note that because Apache does not support upstreams behind Unix sockets you will need to let gitlab-git-http-server listen on a TCP port. You can do this via /etc/default/gitlab.

8. Start application

```
sudo service gitlab start
sudo service nginx restart
```

9. Check application status

Check if GitLab and its environment are configured correctly:

```
cd /home/git/gitlab
sudo -u git mkdir -m 750 /home/git/gitlab/public/uploads
sudo -u git -H bundle exec rake gitlab:env:info RAILS_ENV=production
```

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

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

If all items are green, then congratulations, the upgrade is complete!

GitLab Community Edition 8.1.4 4d7216a

Actualizar Git de 8.1.4 a 8.2.6

https://gitlab.com/gitlab-org/gitlab-ce/blob/11-8-stable/doc/update/8.1-to-8.2.md

GitLab Community Edition 8.1.4 4d7216a update asap

1. Stop server

```
sudo service gitlab stop
```

2. Backup

```
cd /home/git/gitlab
sudo -u git -H bundle exec rake gitlab:backup:create
RAILS_ENV=production
```

3. Get latest code

```
cd /home/git/gitlab
sudo -u git -H git fetch --all
sudo -u git -H git checkout -- db/schema.rb
sudo -u git -H git checkout 8-2-stable
```

4. Update gitlab-shell

```
cd /home/git/gitlab-shell
sudo -u git -H git fetch
sudo -u git -H git checkout v2.6.8
```

5. Replace gitlab-git-http-server with gitlab-workhorse

Install and compile gitlab-workhorse. This requires Go 1.5 which should already be on your system from GitLab 8.1.

```
cd /home/git
sudo -u git -H git clone https://gitlab.com/gitlab-org/gitlab-
workhorse.git
cd gitlab-workhorse
sudo -u git -H git checkout 0.4.2
sudo -u git -H make
```

Update the GitLab 'default' file.

```
cd /home/git/gitlab
test -e /etc/default/gitlab && \
  sudo sed -i.pre-8.2
's/^\([^=]*\)gitlab_git_http_server/\1gitlab_workhorse/'
/etc/default/gitlab
```

Make sure that you also update your **NGINX configuration** to use the new gitlab-workhorse.socket file.

6. Install libs, migrations, etc.

```
cd /home/git/gitlab

# MySQL installations (note: the line below states '--without
postgres')

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


# Run database migrations

sudo -u git -H bundle exec rake db:migrate RAILS_ENV=production

# Clean up assets and cache

sudo -u git -H bundle exec rake assets:clean assets:precompile
cache:clear RAILS_ENV=production

# Update init.d script

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

7. Update configuration files

New configuration options for `gitlab.yml`

There are new configuration options available for [`gitlab.yml`](). View them with the command below and apply them manually to your current `gitlab.yml`:

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

mv gitlab.yml gitlab.8.1.4

sudo nano gitlab.yml
```

Nginx configuration

View changes between the previous recommended Nginx configuration and the current one:

https://gitlab.com/gitlab-org/gitlab-ce/find_file/8-2-stable

```
# For HTTPS configurations

git diff origin/8-1-stable:lib/support/nginx/gitlab-ssl origin/8-2-stable:lib/support/nginx/gitlab-ssl
```

```
cd /etc/nginx/sites-available
mv gitlab-ssl gitlab-ssl.8.1.4
sudo nano gitlab-ssl
```

```
# For HTTP configurations

git diff origin/8-1-stable:lib/support/nginx/gitlab origin/8-2-stable:lib/support/nginx/gitlab
```

```
cd /etc/nginx/sites-available
mv gitlab gitlab.8.1.4
sudo nano gitlab
```

If you are using Apache instead of NGINX please see the updated Apache templates. Also note that because Apache does not support upstreams behind Unix sockets you will need to let gitlab-workhorse listen on a TCP port. You can do this via /etc/default/gitlab.

8. Start application

```
sudo service gitlab start
sudo service nginx restart
```

9. Check application status

Check if GitLab and its environment are configured correctly:

```
cd /home/git/gitlab
sudo -u git mkdir -m 750 /home/git/gitlab/shared/lfs-objects
sudo -u git -H bundle exec rake gitlab:env:info RAILS_ENV=production
```

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

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

GitLab Community Edition 8.2.6 067b8df

Actualizar Git de 8.2.6 a 8.3.10

https://gitlab.com/gitlab-org/gitlab-ce/blob/11-8-stable/doc/update/8.1-to-8.2.md

GitLab Community Edition 8.2.6 067b8df update asap

0. Double-check your Git version

This notice applies only to /usr/local/bin/git

If you compiled Git from source on your GitLab server then please double-check that you are using a version that protects against CVE-2014-9390. For six months after this vulnerability became known the GitLab installation guide still contained instructions that would install the outdated, 'vulnerable' Git version 2.1.2.

Run the following command to get your current Git version:

```
/usr/local/bin/git --version
```

If you see 'No such file or directory' then you did not install Git according to the outdated instructions from the GitLab installation guide and you can go to the next step 'Stop server' below. If you see a version string then it should be v1.8.5.6, v1.9.5, v2.0.5, v2.1.4, v2.2.1 or newer. You can use the instructions in the GitLab source installation guide to install a newer version of Git.

1. Stop server

```
cd /home/git/gitlab
sudo service gitlab stop
```

2. Backup

```
sudo -u git mkdir -m 750 /home/git/gitlab/shared/lfs-objects
sudo -u git -H bundle exec rake gitlab:backup:create
RAILS_ENV=production
```

3. Get latest code

```
# Para evitar el error insufficient permission for adding an object to
repository database .git/objects necesito aplicar el siguiente comando
```

```
cd /home/git/gitlab
sudo chown -R git .git/*
sudo -u git -H git fetch --all
sudo -u git -H git checkout -- db/schema.rb
sudo -u git -H git checkout 8-3-stable
```

4. Update gitlab-shell

```
cd /home/git/gitlab-shell
sudo -u git -H git fetch --all
sudo -u git -H git checkout v2.6.9
```

5. Update gitlab-workhorse

Install and compile gitlab-workhorse. This requires Go 1.5 which should already be on your system from GitLab 8.1.

```
cd /home/git/gitlab-workhorse
sudo -u git -H git fetch --all
sudo -u git -H git checkout 0.5.4
sudo -u git -H make
```

6. Install libs, migrations, etc.

```
cd /home/git/gitlab

# MySQL installations (note: the line below states '--without postgres')

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


# Run database migrations

sudo -u git -H bundle exec rake db:migrate RAILS_ENV=production

# Clean up assets and cache

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

7. Update configuration files

New configuration options for gitlab.yml

There are new configuration options available for gitlab.yml. View them with the command below and apply them manually to your current gitlab.yml:

```
git diff origin/8-2-stable:config/gitlab.yml.example origin/8-3-stable:config/gitlab.yml.example


cd /home/git/gitlab/config
mv gitlab.yml gitlab.8.2.6
sudo nano gitlab.yml
```

GitLab default file

The value of the gitlab_workhorse_options variable should be updated within the default gitlab file (/etc/default/gitlab) according to the following diff:

```
git diff origin/8-2-stable:lib/support/init.d/gitlab.default.example origin/8-3-stable:lib/support/init.d/gitlab.default.example
```

```
cd /etc/default

sudo nano gitlab
```

Nginx configuration

GitLab 8.3 introduces major changes in the NGINX configuration. Because all HTTP requests pass through gitlab-workhorse now a lot of directives need to be removed from NGINX. During future upgrades there should be much less changes in the NGINX configuration because of this.

View changes between the previous recommended Nginx configuration and the current one:

```
# For HTTPS configurations

git diff origin/8-2-stable:lib/support/nginx/gitlab-ssl origin/8-3-stable:lib/support/nginx/gitlab-ssl
```

```
cd /etc/nginx/sites-available

mv gitlab-ssl gitlab-ssl.8.2.6

sudo nano gitlab-ssl
```

```
# For HTTP configurations

git diff origin/8-2-stable:lib/support/nginx/gitlab origin/8-3-stable:lib/support/nginx/gitlab
```

```
cd /etc/nginx/sites-available

mv gitlab gitlab.8.2.6

sudo nano gitlab
```

If you are using Apache instead of NGINX please see the updated Apache templates. Also note that because Apache does not support upstreams behind Unix sockets you will need to let gitlab-workhorse listen on a TCP port. You can do this via /etc/default/gitlab.

Init script

We updated the init script for GitLab in order to pass new configuration options to gitlab-workhorse. We let gitlab-workhorse connect to the Rails application via a Unix domain socket and we tell it where the 'public' directory of GitLab is.

```
cd /home/git/gitlab
sudo cp lib/support/init.d/gitlab /etc/init.d/gitlab
```

8. Use Redis v2.8.0+

```
# Comprobar Version:
redis-server -v
redis-cli -s /var/run/redis/redis.sock INFO

# Build Redis
cd /home/git/gitlab
wget http://download.redis.io/releases/redis-2.8.23.tar.gz
tar xzf redis-2.8.23.tar.gz
cd redis-2.8.23
make
sudo apt-get install tcl
sudo make install

# Install Redis
cd utils
sudo ./install_server.sh

# Configure redis to use sockets
sudo cp /etc/redis/redis.conf /etc/redis/redis.conf.orig

# Disable Redis listening on TCP by setting 'port' to 0
sed 's/^port .*/port 0/' /etc/redis/redis.conf.orig | sudo tee
/etc/redis/redis.conf

# Enable Redis socket for default Debian / Ubuntu path
echo 'unixsocket /var/run/redis/redis.sock' | sudo tee -a
/etc/redis/redis.conf

# Grant permission to the socket to all members of the redis group
echo 'unixsocketperm 770' | sudo tee -a /etc/redis/redis.conf

# Create the directory which contains the socket
mkdir /var/run/redis
chown redis:redis /var/run/redis
chmod 755 /var/run/redis

# Persist the directory which contains the socket, if applicable
if [ -d /etc/tmpfiles.d ]; then
  echo 'd  /var/run/redis  0755  redis  redis  10d  -' | sudo tee -a
/etc/tmpfiles.d/redis.conf
```

```
# Activate the changes to redis.conf

sudo service redis_6379 restart

# Add git to the redis group

sudo usermod -aG redis git
# Evitar errores de sidekiq, tipo You are using Redis v2.4.14, Sidekiq
requires Redis v2.8.0 or greater. Hay que editar el archivo rescue.yml
y cambiar la línea production: unix:/var/run/redis/redis.sock por
production: redis://localhost:6379

cd /home/git/gitlab/config

sudo nano resque.yml
```

You can check your Redis version with the following command:

```
redis-cli info | grep redis_version
redis-cli -s /var/run/redis/redis.sock INFO
sudo service redis_6379 restart
```

If you need to upgrade, see the [installation guide for Redis](#).

9. Start application

```
sudo service gitlab start

sudo service nginx restart
```

10. Check application status

Check if GitLab and its environment are configured correctly:

```
cd /home/git/gitlab

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

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

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

# comprobar solo sidekiq

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

If all items are green, then congratulations, the upgrade is complete!

GitLab Community Edition 8.3.10 5ab31e2 update asap

Actualizar Git de 8.3.10 a 8.4.11

https://gitlab.com/gitlab-org/gitlab-ce/blob/11-8-stable/doc/update/8.3-to-8.4.md

GitLab Community Edition 8.3.10 5ab31e2

1. Stop server

```
sudo service gitlab stop
```

2. Backup

```
cd /home/git/gitlab
sudo -u git -H bundle exec rake gitlab:backup:create
RAILS_ENV=production
```

3. Get latest code

```
sudo -u git -H git fetch --all
sudo -u git -H git checkout -- db/schema.rb
sudo -u git -H git checkout 8-4-stable
```

4. Update gitlab-shell

```
cd /home/git/gitlab-shell
sudo -u git -H git fetch --all
sudo -u git -H git checkout v2.6.10
```

5. Update gitlab-workhorse

```
cd /home/git/gitlab-workhorse
sudo -u git -H git fetch --all
sudo -u git -H git checkout 0.6.2
sudo -u git -H make
```

6. Install libs, migrations, etc.

```
cd /home/git/gitlab

# MySQL installations (note: the line below states '--without
postgres')

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

```
# Run database migrations

sudo -u git -H bundle exec rake db:migrate RAILS_ENV=production

# Clean up assets and cache

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

7. Update configuration files

```
# New configuration options for gitlab.yml available for gitlab.yml. View them
with the command below and apply them manually to your current gitlab.yml:git
diff origin/8-3-stable:config/gitlab.yml.example origin/8-4-
stable:config/gitlab.yml.example


cd /home/git/gitlab/config

mv gitlab.yml gitlab8.3.10

sudo nano gitlab.yml

# We updated the init script for GitLab in order to set a specific
PATH for gitlab-workhorse.

cd /home/git/gitlab

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

8. Start application

```
sudo service gitlab start
sudo service nginx restart
```

9. Check application status

```
# Check if GitLab and its environment are configured correctly:

cd /home/git/gitlab

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


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

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


# If all items are green, then congratulations, the upgrade is
complete!
```

GitLab Community Edition 8.4.11 e9bef89 update asap

Actualizar Git de 8.4.11 a 8.5.13

https://gitlab.com/gitlab-org/gitlab-ce/blob/11-8-stable/doc/update/8.4-to-8.5.md

GitLab Community Edition 8.4.11 e9bef89 update asap

```
cd /home/git/gitlab/config
sudo nano resque.yml
sudo reboot
```

1. Stop server

```
sudo service gitlab stop
cd /home/git/gitlab/config
sudo -u git -H mkdir -p public/uploads
sudo chmod -R u+rwX  public/uploads
sudo chmod 0700 public/uploads
sudo -u git -H editor /home/git/gitlab-shell/config.yml
```

2. Backup

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

3. Get latest code

```
cd /home/git/gitlab
sudo -u git -H git fetch --all
sudo -u git -H git checkout -- db/schema.rb
sudo -u git -H git checkout 8-5-stable
```

4. Update gitlab-shell

```
cd /home/git/gitlab-shell
sudo -u git -H git fetch --all
sudo -u git -H git checkout v2.6.10
```

5. Update gitlab-workhorse

```
cd /home/git/gitlab-workhorse
sudo -u git -H git fetch --all
sudo -u git -H git checkout 0.6.4
sudo -u git -H make
```

6. Install libs, migrations, etc.

```
cd /home/git/gitlab

# MySQL installations (note: the line below states '--without
postgres')
```

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

# Optional: clean up old gems

sudo -u git -H bundle clean

# Run database migrations

sudo -u git -H bundle exec rake db:migrate RAILS_ENV=production

# Clean up assets and cache

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

7. Update configuration files

There are new configuration options available for [gitlab.yml](). View them with the command below and apply them manually to your current gitlab.yml: `git diff origin/8-4-stable:config/gitlab.yml.example origin/8-5-stable:config/gitlab.yml.example`

```
cd /home/git/gitlab/config
mv gitlab.yml gitlab.8.4.11
sudo nano gitlab.yml
```

Ensure you're still up-to-date with the latest NGINX configuration changes:

```
# For HTTP & HTTPS configurations

git diff origin/8-4-stable:lib/support/nginx/gitlab-ssl origin/8-5-stable:lib/support/nginx/gitlab-ssl

git diff origin/8-4-stable:lib/support/nginx/gitlab origin/8-5-stable:lib/support/nginx/gitlab
```

```
cd /etc/nginx/sites-available
mv gitlab-ssl gitlab-ssl.8.4.11
sudo nano gitlab-ssl
mv gitlab gitlab.8.4.11
sudo nano gitlab
```

If you are using Apache instead of NGINX please see the updated Apache templates. Also note that because Apache does not support upstreams behind Unix sockets you will need to let gitlab-workhorse listen on a TCP port. You can do this via /etc/default/gitlab.

Init script

Ensure you're still up-to-date with the latest init script changes:

```
cd /home/git/gitlab
sudo cp lib/support/init.d/gitlab /etc/init.d/gitlab
```

8. Start application

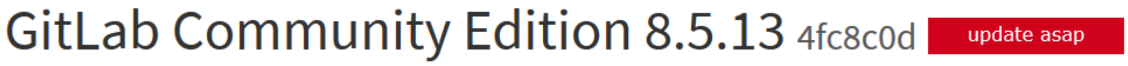
```
sudo service gitlab start
sudo service nginx restart
```

9. Check application status

Check if GitLab and its environment are configured correctly:

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

If all items are green, then congratulations, the upgrade is complete!



GitLab Community Edition 8.5.13 4fc8c0d

GitLab Community Edition 8.5.13 4fc8c0d update asap

Actualizar Git de 8.5.13 a 8.6.9

https://gitlab.com/gitlab-org/gitlab-ce/blob/11-8-stable/doc/update/8.5-to-8.6.md

1. Stop server

```
sudo service gitlab stop
```

2. Backup

```
cd /home/git/gitlab
sudo -u git -H bundle exec rake gitlab:backup:create
RAILS_ENV=production
```

3. Get latest code

```
sudo -u git -H git fetch --all
sudo -u git -H git checkout -- db/schema.rb
sudo -u git -H git checkout 8-6-stable
```

4. Update gitlab-shell

```
cd /home/git/gitlab-shell
sudo -u git -H git fetch --all
sudo -u git -H git checkout v2.6.12
```

5. Update gitlab-workhorse

Install and compile gitlab-workhorse. This requires Go 1.5 which should already be on your system from GitLab 8.1.

```
cd /home/git/gitlab-workhorse
sudo -u git -H git fetch --all
sudo -u git -H git checkout v0.7.1
sudo -u git -H make
```

7. Install libs, migrations, etc.

```
cd /home/git/gitlab

# MySQL installations (note: the line below states '--without
postgres')

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


# Run database migrations

sudo -u git -H bundle exec rake db:migrate RAILS_ENV=production
```

```
# Clean up assets and cache

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

8. Update configuration files

New configuration options for `gitlab.yml`

There are new configuration options available for `gitlab.yml`. View them with the command below and apply them manually to your current `gitlab.yml`: git diff origin/8-5-stable:config/gitlab.yml.example origin/8-6-stable:config/gitlab.yml.example

```
cd /home/git/gitlab/config
mv gitlab.yml gitlab.8.5.13
sudo nano gitlab.yml
```

Nginx configuration

Ensure you're still up-to-date with the latest NGINX configuration changes:

```
# For HTTP Y HTTPS configurations

git diff origin/8-5-stable:lib/support/nginx/gitlab-ssl origin/8-6-
stable:lib/support/nginx/gitlab-ssl

git diff origin/8-5-stable:lib/support/nginx/gitlab origin/8-6-
stable:lib/support/nginx/gitlab

cd /etc/nginx/sites-available

mv gitlab-ssl gitlab-ssl.8.5.13

sudo nano gitlab-ssl

mv gitlab gitlab.8.5.13

sudo nano gitlab
```

Init script: Ensure you're still up-to-date with the latest init script changes:

```
cd /home/git/gitlab
sudo cp lib/support/init.d/gitlab /etc/init.d/gitlab
```

9. Start application

```
cd /home/git/gitlab
sudo service gitlab start
sudo service nginx restart
```

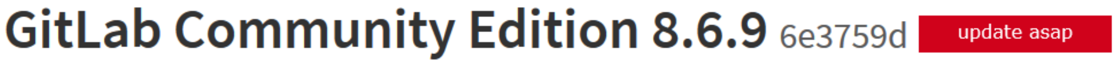
10. Check application status

```
cd /home/git/gitlab
rm -rf public
sudo -u git -H mkdir -p public/uploads
sudo chmod -R u+rwX  public/uploads
sudo chmod 700 /home/git/gitlab/public/uploads
```

Check if GitLab and its environment are configured correctly:

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

If all items are green, then congratulations, the upgrade is complete!



GitLab Community Edition 8.6.9 6e3759d update asap

GitLab Community Edition 8.6.9 6e3759d update asap

Actualizar Git de 8.6.9 a 8.7.9

https://gitlab.com/gitlab-org/gitlab-ce/blob/11-8-stable/doc/update/8.6-to-8.7.md

1. Stop server

```
sudo service gitlab stop
```

2. Backup

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

3. Get latest code

```
cd /home/git/gitlab
sudo -u git -H git fetch --all
sudo -u git -H git checkout -- db/schema.rb
sudo -u git -H git checkout 8-7-stable
```

4. Update gitlab-shell

```
cd /home/git/gitlab-shell
sudo -u git -H git fetch --tags
sudo -u git -H git checkout v2.7.2
```

5. Update gitlab-workhorse

```
cd /home/git/gitlab-workhorse
sudo -u git -H git fetch --all
sudo -u git -H git checkout v0.7.1
sudo -u git -H make
```

6. Install libs, migrations, etc.

```
cd /home/git/gitlab

# MySQL installations (note: the line below states '--without
postgres')

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

# Run database migrations

sudo -u git -H bundle exec rake db:migrate RAILS_ENV=production

# Clean up assets and cache

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

7. Update configuration files

```
#  gitlab.yml : There are new configuration options available for
gitlab.yml. View them with the command below and apply them manually
to your current gitlab.yml:
```

```
git diff origin/8-6-stable:config/gitlab.yml.example origin/8-7-stable:config/gitlab.yml.example
```

```
cd /home/git/gitlab/config
mv gitlab.yml gitlab.8.6.9
sudo nano gitlab.yml
```

```
# Git configuration: Disable git gc --auto because GitLab runs git gc
for us already.
```

```
sudo -u git -H git config --global gc.auto 0
```

Nginx configuration

```
git diff origin/8-6-stable:lib/support/nginx/gitlab-ssl origin/8-7-stable:lib/support/nginx/gitlab-ssl
```

```
git diff origin/8-6-stable:lib/support/nginx/gitlab origin/8-7-stable:lib/support/nginx/gitlab
```

```
cd /etc/nginx/sites-available

mv gitlab-ssl gitlab-ssl.8.6.9

sudo nano gitlab-ssl

mv gitlab gitlab.8.6.9

sudo nano gitlab
```

`Init script: Ensure you're still up-to-date with the latest init script changes`

```
cd /home/git/gitlab
sudo cp lib/support/init.d/gitlab /etc/init.d/gitlab
```

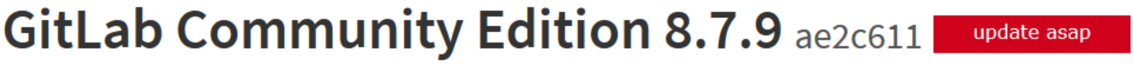
8. Start application

```
sudo service gitlab start
sudo service nginx restart
```

9. Check application status

`Check if GitLab and its environment are configured correctly:`

```
cd /home/git/gitlab

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

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



GitLab Community Edition 8.7.9 ae2c611 update asap

GitLab Community Edition 8.7.9 ae2c611 update asap

Actualizar Git de 8.7.9 a 8.8.9

https://gitlab.com/gitlab-org/gitlab-ce/blob/11-8-stable/doc/update/8.6-to-8.7.md

1. Stop server

```
sudo service gitlab stop
```

2. Backup

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

3. Get latest code

```
cd /home/git/gitlab
sudo -u git -H git fetch --all
sudo -u git -H git checkout -- db/schema.rb
sudo -u git -H git checkout 8-8-stable
```

4. Update gitlab-shell

```
cd /home/git/gitlab-shell
sudo -u git -H git fetch --all --tags
sudo -u git -H git checkout v2.7.2
```

5. Update gitlab-workhorse

```
cd /home/git/gitlab-workhorse
sudo -u git -H git fetch --all
sudo -u git -H git checkout v0.7.1
sudo -u git -H make
```

6. Install libs, migrations, etc.

```
cd /home/git/gitlab

# MySQL installations (note: the line below states '--without
postgres')

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


# Run database migrations

sudo -u git -H bundle exec rake db:migrate RAILS_ENV=production

# Clean up assets and cache
```

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

7. Update configuration files

There are new configuration options available for gitlab.yml. View them with the command below and apply them manually to your current gitlab.yml:

```
git diff origin/8-7-stable:config/gitlab.yml.example origin/8-8-stable:config/gitlab.yml.example
```

Git configuration

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

mv gitlab.yml gitlab.8.7.9

sudo nano gitlab.yml
```

Disable git gc --auto because GitLab runs git gc for us already.

```
sudo -u git -H git config --global gc.auto 0
```

Nginx configuration

# Ensure you're still up-to-date with the latest NGINX configuration changes: gitlab-ssl & gitlab

```
cd /etc/nginx/sites-available

mv gitlab-ssl gitlab-ssl.8.7.9

sudo nano gitlab-ssl

mv gitlab gitlab.8.7.9

sudo nano gitlab
```

Init script: Ensure you're still up-to-date with the latest init script changes:

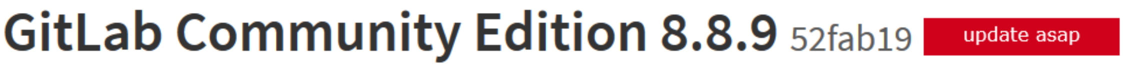
```
cd /home/git/gitlab

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

8. Start application

```
sudo service gitlab start
sudo service nginx restart
```

9. Check application status

```
cd /home/git/gitlab

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



GitLab Community Edition 8.8.9 52fab19 update asap

GitLab Community Edition 8.8.9 52fab19

Actualizar Git de 8.8.9 a 8.9.11

https://gitlab.com/gitlab-org/gitlab-ce/blob/11-8-stable/doc/update/8.8-to-8.9.md

1. Stop server

```
sudo service gitlab stop
```

2. Backup

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

3. Get latest code

```
cd /home/git/gitlab

sudo -u git -H git fetch --all

sudo -u git -H git checkout -- db/schema.rb

sudo -u git -H git checkout 8-9-stable
```

4. Update gitlab-shell

```
cd /home/git/gitlab-shell
sudo -u git -H git fetch --all --tags
sudo -u git -H git checkout v3.0.0
```

5. Update gitlab-workhorse

```
cd /home/git/gitlab-workhorse
sudo -u git -H git fetch --all
sudo -u git -H git checkout v0.7.5
sudo -u git -H make
```

6. Update MySQL permissions

If you are using MySQL you need to grant the GitLab user the necessary permissions on the database:

```
# Login to MySQL

mysql -u root -p

# Grant the GitLab user the REFERENCES permission on the database

GRANT REFERENCES ON `gitlab_production`.* TO 'git'@'localhost';

# Quit the database session
```

```
mysql> \q
```

7. Install libs, migrations, etc.

```
cd /home/git/gitlab

# MySQL installations (note: the line below states '--without postgres')

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

# Run database migrations

sudo -u git -H bundle exec rake db:migrate RAILS_ENV=production

# Clean up assets and cache

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

8. Update configuration files

New configuration options for gitlab.yml. View them with the command below and apply them manually to your current gitlab.yml:

```
git diff origin/8-8-stable:config/gitlab.yml.example origin/8-9-stable:config/gitlab.yml.example

cd /home/git/gitlab/config

mv gitlab.yml gitlab.8.8.9

sudo nano gitlab.yml
```

Git configuration: Disable git gc --auto because GitLab runs git gc for us already.

```
sudo -u git -H git config --global gc.auto 0
```

Nginx configuration:

```
cd /etc/nginx/sites-available

mv gitlab-ssl gitlab-ssl.8.8.9

sudo nano gitlab-ssl

mv gitlab gitlab.8.8.9

sudo nano gitlab
```

If you are using Apache instead of NGINX please see the updated Apache templates. Also note that because Apache does not support upstreams behind Unix sockets you will need to let gitlab-workhorse listen on a TCP port. You can do this via /etc/default/gitlab.

SMTP configuration

If you're installing from source and use SMTP to deliver mail, you will need to add the following line to config/initializers/smtp_settings.rb:

```
ActionMailer::Base.delivery_method = :smtp
```

See smtp_settings.rb.sample as an example.

Init script: Ensure you're still up-to-date with the latest init script changes:

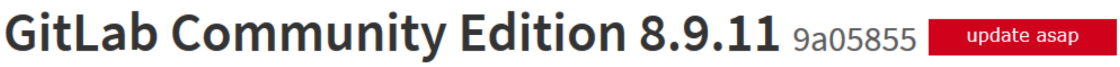
```
cd /home/git/gitlab
sudo cp lib/support/init.d/gitlab /etc/init.d/gitlab
```

9. Start application

```
sudo service gitlab start
sudo service nginx restart
```

10. 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
sudo -u git -H bundle exec rake gitlab:check RAILS_ENV=production
```



GitLab Community Edition 8.9.11 9a05855

GitLab Community Edition 8.9.11 9a05855

Actualizar Git de 8.9.11 a 8.10.13

https://gitlab.com/gitlab-org/gitlab-ce/blob/11-8-stable/doc/update/8.9-to-8.10.md

1. Stop server

```
sudo service gitlab stop
```

2. Backup

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

3. Get latest code

```
cd /home/git/gitlab
sudo -u git -H git fetch --all
sudo -u git -H git checkout -- db/schema.rb
sudo -u git -H git checkout 8-10-stable
```

4. Update gitlab-shell

```
cd /home/git/gitlab-shell
sudo -u git -H git fetch --all --tags
sudo -u git -H git checkout v3.2.1
```

5. Update gitlab-workhorse

```
cd /home/git/gitlab-workhorse
sudo -u git -H git fetch --all
sudo -u git -H git checkout v0.7.8
sudo -u git -H make
```

6. Update MySQL permissions

If you are using MySQL you need to grant the GitLab user the necessary permissions on the database:

```
# Login to MySQL
mysql -u root -p

# Grant the GitLab user the REFERENCES permission on the database
GRANT REFERENCES ON `gitlab_production`.* TO 'git'@'localhost';

# Quit the database session
mysql> \q
```

7. Install libs, migrations, etc.

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

```
Downloading omniauth-google-oauth2-0.4.1 revealed dependencies not in the API or the lockfile (omniauth-oauth2 (>= 1.3.1), jwt (~> 1.5.2)). Either installing with `--full-index` or running `bundle update omniauth-google-oauth2` should fix the problem.
```

```
sudo -u git -H nano Gemfile.lock
```

```
omniauth-google-oauth2 (0.4.1)

  addressable (~> 2.3)

  jwt (~> 1.0)     -→ jwt (~> 1.5.2)

  multi_json (~> 1.3)

  omniauth (>= 1.1.1)

  omniauth-oauth2 (~> 1.3.1) → omniauth-oauth2 (>= 1.3.1)
```

```
# MySQL installations (note: the line below states '--without postgres')

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

```
# Run database migrations

sudo -u git -H bundle exec rake db:migrate RAILS_ENV=production

# Clean up assets and cache

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

8. Update configuration files

New configuration options for **gitlab.yml**. View them with the command below and apply them manually to your current gitlab.yml:

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

mv gitlab.yml gitlab.8.9.11

sudo nano gitlab.yml
```

Git configuration: Disable git gc --auto because GitLab runs git gc for us already.

```
sudo -u git -H git config --global gc.auto 0
```

Nginx configuration:

```
cd /etc/nginx/sites-available
```

```
mv gitlab-ssl gitlab-ssl.8.9.11
sudo nano gitlab-ssl
mv gitlab gitlab.8.9.11
sudo nano gitlab
```

SMTP configuration

If you're installing from source and use SMTP to deliver mail, you will need to add the following line to config/initializers/smtp_settings.rb:

```
ActionMailer::Base.delivery_method = :smtp
```

See [smtp_settings.rb.sample]() as an example.

Init script

```
cd /home/git/gitlab
sudo cp lib/support/init.d/gitlab /etc/init.d/gitlab
```

9. Start application

```
sudo service gitlab start
sudo service nginx restart
```

10. Check application status

Check if GitLab and its environment are configured correctly:

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

GitLab Community Edition 8.10.13 c593548

Actualizar Git de 8.10.13 a 8.11.11

https://gitlab.com/gitlab-org/gitlab-ce/blob/11-8-stable/doc/update/8.10-to-8.11.md

GitLab Community Edition 8.10.13 c593548

1. Stop server

```
sudo service gitlab stop
```

2. Backup

```
cd /home/git/gitlab
sudo -u git -H bundle exec rake gitlab:backup:create
RAILS_ENV=production
```

3. Update Ruby

We will continue supporting Ruby < 2.3 for the time being but we recommend you upgrade to Ruby 2.3 if you're running a source installation, as this is the same version that ships with our Omnibus package.

You can check which version you are running with `ruby -v`.

Download and compile Ruby:

```
mkdir /tmp/ruby && cd /tmp/ruby
curl --remote-name --progress https://cache.ruby-
lang.org/pub/ruby/2.3/ruby-2.3.1.tar.gz
echo 'c39b4001f7acb4e334cb60a0f4df72d434bef711  ruby-2.3.1.tar.gz' |
shasum --check - && tar xzf ruby-2.3.1.tar.gz
cd ruby-2.3.1
./configure --disable-install-rdoc
make
sudo make install
```

Install Bundler:

```
sudo gem install bundler --no-document --version '< 2'
```

4. Get latest code

```
cd /home/git/gitlab

sudo -u git -H git fetch --all
sudo -u git -H git checkout -- db/schema.rb
git stash
sudo -u git -H git checkout 8-11-stable
```

5. Update gitlab-shell

```
cd /home/git/gitlab-shell
sudo -u git -H git fetch --all --tags
sudo -u git -H git checkout v3.4.0
```

6. Update gitlab-workhorse

```
cd /home/git/gitlab-workhorse
sudo -u git -H git fetch --all
sudo -u git -H git checkout v0.7.11
sudo -u git -H make
```

7. Install libs, migrations, etc.

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

```
gem install bundler
```

```
Downloading omniauth-google-oauth2-0.4.1 revealed dependencies not in
the API or the lockfile (omniauth-oauth2 (>= 1.3.1), jwt (~> 1.5.2)).
Either installing with `--full-index` or running `bundle update
omniauth-google-oauth2` should fix the problem..
```

```
sudo -u git -H nano Gemfile.lock
```

```
omniauth-google-oauth2 (0.4.1)
      addressable (~> 2.3)
      jwt (~> 1.0)     -→ jwt (~> 1.5.2)
      multi_json (~> 1.3)
      omniauth (>= 1.1.1)
      omniauth-oauth2 (~> 1.3.1) → omniauth-oauth2 (>= 1.3.1)
```

```
# https://bundler.io/guides/bundler_2_upgrade.html
```

```
# MySQL installations (note: the line below states '--without
postgres')
```

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

```
# Run database migrations
```

```
sudo -u git -H bundle exec rake db:migrate RAILS_ENV=production
```

```
# Clean up assets and cache
```

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

8. Update configuration files

New configuration options for gitlab.yml. View them with the command below and apply them manually to your current gitlab.yml:

```
git diff origin/8-10-stable:config/gitlab.yml.example origin/8-11-
stable:config/gitlab.yml.example
```

```
cd /home/git/gitlab/config
mv gitlab.yml gitlab.8.10.13
sudo nano gitlab.yml
```

```
Nginx configuration:

cd /etc/nginx/sites-available

mv gitlab-ssl gitlab-ssl.8.10.13

sudo nano gitlab-ssl

mv gitlab gitlab.8.10.13

sudo nano gitlab
```

```
# Ensure you're still up-to-date with the latest init script changes:

cd /home/git/gitlab
sudo cp lib/support/init.d/gitlab /etc/init.d/gitlab
```

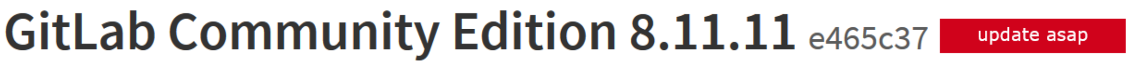
9. Start application

```
sudo service gitlab start
sudo service nginx restart
```

10. Check application status

Check if GitLab and its environment are configured correctly:

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



GitLab Community Edition 8.11.11 e465c37

Actualizar Git de 8.11.11 a 8.12.13

https://gitlab.com/gitlab-org/gitlab-ce/blob/11-8-stable/doc/update/8.11-to-8.12.md

GitLab Community Edition 8.11.11 e465c37 update asap

1. Stop server

```
sudo service gitlab stop
```

2. Backup

```
cd /home/git/gitlab
sudo -u git -H bundle exec rake gitlab:backup:create RAILS_ENV=production
```

3. Update Ruby

```
No hace falta
```

4. Get latest code

```
cd /home/git/gitlab
sudo -u git -H git fetch --all
sudo -u git -H git checkout -- db/schema.rb
git stash
sudo -u git -H git checkout 8-12-stable
```

5. Update gitlab-shell

```
cd /home/git/gitlab-shell
sudo -u git -H git fetch --all --tags
sudo -u git -H git checkout v3.6.1
```

6. Update gitlab-workhorse

```
cd /home/git/gitlab-workhorse
sudo -u git -H git fetch --all
sudo -u git -H git checkout v0.8.2
sudo -u git -H make
```

7. Install libs, migrations, etc.

```
# MySQL installations (note: the line below states '--without postgres')
```

```
cd /home/git/gitlab
sudo -u git -H bundle install --without postgres development test --deployment
```

```
# Run database migrations

sudo -u git -H bundle exec rake db:migrate RAILS_ENV=production

# Clean up assets and cache

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

8. Update configuration files

New configuration options for gitlab.yml. View them with the command below and apply them manually to your current gitlab.yml: `git diff origin/8-11-stable:config/gitlab.yml.example origin/8-12-stable:config/gitlab.yml.example`

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

mv gitlab.yml gitlab.8.11.11

sudo nano gitlab.yml
```

Git configuration

```
sudo -u git -H git config --global repack.writeBitmaps true
```

Nginx configuration

```
cd /etc/nginx/sites-available

mv gitlab-ssl gitlab-ssl.8.11.11

sudo nano gitlab-ssl

mv gitlab gitlab.8.11.11

sudo nano gitlab
```

Init script

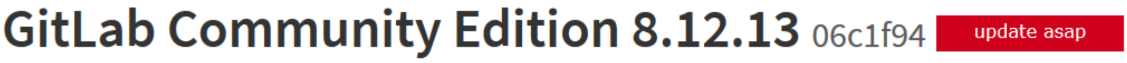
```
cd /home/git/gitlab

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

9. Start application

```
sudo service gitlab start
sudo service nginx restart
```

10. Check application status

```
cd /home/git/gitlab

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

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



GitLab Community Edition 8.12.13 06c1f94 update asap

Actualizar Git de 8.12.13 a 8.13.12

https://gitlab.com/gitlab-org/gitlab-ce/blob/11-8-stable/doc/update/8.12-to-8.13.md

GitLab Community Edition 8.12.13 06c1f94 update asap

1. Stop server

```
sudo service gitlab stop
```

2. Backup

```
cd /home/git/gitlab
sudo -u git -H bundle exec rake gitlab:backup:create RAILS_ENV=production
```

3. Update Ruby

No hace falta

4. Get latest code

```
cd /home/git/gitlab
sudo -u git -H git fetch --all
sudo -u git -H git checkout -- db/schema.rb
sudo -u git -H git checkout 8-13-stable
```

5. Update gitlab-shell

```
cd /home/git/gitlab-shell
sudo -u git -H git fetch --all --tags
sudo -u git -H git checkout v3.6.7
```

6. Update gitlab-workhorse

```
cd /home/git/gitlab-workhorse
sudo -u git -H git fetch --all
sudo -u git -H git checkout v0.8.5
sudo -u git -H make
```

7. Install libs, migrations, etc.

```
# MySQL installations (note: the line below states '--without postgres')

cd /home/git/gitlab
sudo -u git -H bundle install --without postgres development test --deployment
```

```
# Run database migrations

sudo -u git -H bundle exec rake db:migrate RAILS_ENV=production

# Clean up assets and cache

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

8. Update configuration files

New configuration options for gitlab.yml. View them with the command below and apply them manually to your current gitlab.yml:git diff origin/8-12-stable:config/gitlab.yml.example origin/8-13-stable:config/gitlab.yml.example

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

mv gitlab.yml gitlab.8.12.13

sudo nano gitlab.yml
```

Git configuration

```
sudo -u git -H git config --global repack.writeBitmaps true
```

Nginx configuration

```
cd /etc/nginx/sites-available

mv gitlab-ssl gitlab-ssl.8.12.13

sudo nano gitlab-ssl

mv gitlab gitlab.8.12.13

sudo nano gitlab
```

Init script

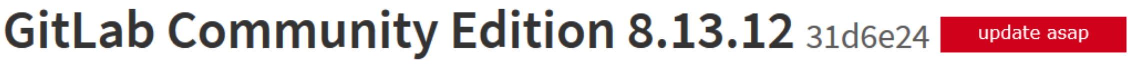
```
cd /home/git/gitlab

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

9. Start application

```
sudo service gitlab start
sudo service nginx restart
```

10. Check application status

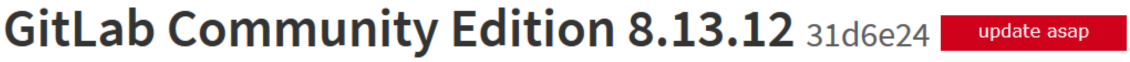
```
cd /home/git/gitlab
sudo -u git -H bundle exec rake gitlab:env:info RAILS_ENV=production
sudo -u git -H bundle exec rake gitlab:check RAILS_ENV=production
```



GitLab Community Edition 8.13.12 31d6e24 update asap

Actualizar Git de 8.13.12 a 8.14.10

https://gitlab.com/gitlab-org/gitlab-ce/blob/11-8-stable/doc/update/8.13-to-8.14.md



GitLab Community Edition 8.13.12 31d6e24 update asap

1. Stop server

```
sudo service gitlab stop
```

2. Backup

```
cd /home/git/gitlab
sudo -u git -H bundle exec rake gitlab:backup:create
RAILS_ENV=production
```

3. Update Ruby

No hace falta, ya tengo esta versión

4. Get latest code

```
cd /home/git/gitlab
error: insufficient permission for adding an object to repository
database .git/objects
sudo chown -R git:staff .git

sudo -u git -H git fetch --all
sudo -u git -H git checkout -- db/schema.rb
sudo -u git -H git checkout 8-14-stable
```

5. Update gitlab-shell

```
cd /home/git/gitlab-shell
sudo -u git -H git fetch --all --tags
sudo -u git -H git checkout v4.1.1
```

6. Update gitlab-workhorse

```
cd /home/git/gitlab-workhorse
sudo -u git -H git fetch --all
sudo -u git -H git checkout v1.0.1
sudo -u git -H make
```

7. Install libs, migrations, etc.

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

```
# MySQL installations (note: the line below states '--without postgres')
```

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

```
# Run database migrations
```

```
sudo -u git -H bundle exec rake db:migrate RAILS_ENV=production
```

```
# Clean up assets and cache
```

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

8. Update configuration files

New configuration options for gitlab.yml.

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

```
mv gitlab.yml gitlab.8.13.12
```

```
sudo nano gitlab.yml
```

Git configuration

```
sudo -u git -H git config --global repack.writeBitmaps true
```

Nginx configuration

```
cd /etc/nginx/sites-available
mv gitlab-ssl gitlab-ssl.8.13.12
sudo nano gitlab-ssl
mv gitlab gitlab.8.13.12
sudo nano gitlab
```

Init script

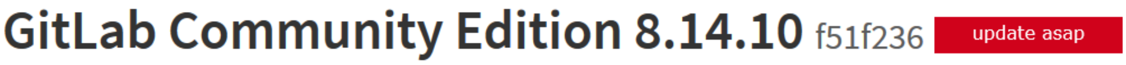
```
cd /home/git/gitlab
sudo cp lib/support/init.d/gitlab /etc/init.d/gitlab
```

9. Start application

```
sudo service gitlab start
sudo service nginx restart
```

10. Check application status

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



GitLab Community Edition 8.14.10 f51f236 update asap

Actualizar Git de 8.13.12 a 8.14

https://gitlab.com/gitlab-org/gitlab-ce/blob/11-8-stable/doc/update/8.14-to-8.15.md

GitLab Community Edition 8.14.10 f51f236 update asap

1. Stop server

```
sudo service gitlab stop
```

2. Backup

```
cd /home/git/gitlab
sudo -u git -H bundle exec rake gitlab:backup:create
RAILS_ENV=production
```

3. Update Ruby

No hace falta actualiar

4. Get latest code

```
cd /home/git/gitlab
sudo -u git -H git fetch --all
sudo -u git -H git checkout -- db/schema.rb
sudo -u git -H git checkout 8-15-stable
```

5. Install libs, migrations, etc.

```
# MySQL installations (note: the line below states '--without
postgres')

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

# Run database migrations

sudo -u git -H bundle exec rake db:migrate RAILS_ENV=production

# Clean up assets and cache

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

6. Update gitlab-workhorse

Comprobar versión https://gitlab.com/gitlab-org/gitlab-ce/blob/8-15-stable/GITLAB_WORKHORSE_VERSION

```
cd /home/git/gitlab-workhorse

sudo -u git -H git fetch --all

sudo -u git -H git checkout v1.2.1

sudo -u git -H make
```

```
cd /home/git/gitlab

sudo -u git -H bundle exec rake
"gitlab:workhorse:install[/home/git/gitlab-workhorse]"
RAILS_ENV=production
```

7. Update gitlab-shell

```
cd /home/git/gitlab-shell
sudo -u git -H git fetch --all --tags
sudo -u git -H git checkout v4.1.1
```

8. Update configuration files

New configuration options for gitlab.yml:

```
cd /home/git/gitlab/config
mv gitlab.yml gitlab.8.14.10
sudo nano gitlab.yml
```

Git configuration

```
sudo -u git -H git config --global repack.writeBitmaps true
```

Nginx configuration

```
cd /etc/nginx/sites-available
mv gitlab-ssl gitlab-ssl.8.14.10
sudo nano gitlab-ssl
mv gitlab gitlab.8.4.10
sudo nano gitlab
```

Init script

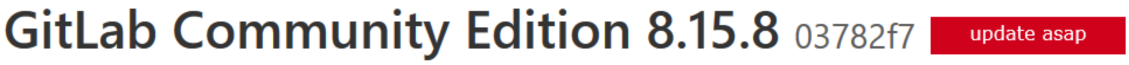
```
cd /home/git/gitlab
sudo cp lib/support/init.d/gitlab /etc/init.d/gitlab
```

9. Start application

```
sudo service gitlab start
sudo service nginx restart
```

10. Check application status

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



GitLab Community Edition 8.15.8 03782f7 update asap

Actualizar Git de 8.15 a 8.16

https://gitlab.com/gitlab-org/gitlab-ce/blob/11-8-stable/doc/update/8.15-to-8.16.md

GitLab Community Edition 8.15.8 03782f7 update asap

1. Stop server

```
sudo service gitlab stop
```

2. Backup

```
cd /home/git/gitlab

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

3. Update Ruby

```
# ruby -v -> actualiar a versión 2.3.3

# Download and compile Ruby:

mkdir /tmp/ruby && cd /tmp/ruby
curl --remote-name --progress https://cache.ruby-
lang.org/pub/ruby/2.3/ruby-2.3.3.tar.gz
echo '1014ee699071aa2ddd501907d18cbe15399c997d  ruby-2.3.3.tar.gz' |
shasum -c - && tar xzf ruby-2.3.3.tar.gz  → doble espacio
cd ruby-2.3.3
./configure --disable-install-rdoc
make
sudo make install

# Install Bundler:

sudo gem install bundler --no-document --version '< 2'
```

4. Get latest code

```
cd /home/git/gitlab

sudo -u git -H git fetch --all
sudo -u git -H git checkout -- db/schema.rb
sudo -u git -H git checkout 8-16-stable
```

5. Install libs, migrations, etc.

```
cd /home/git/gitlab

# MySQL installations (note: the line below states '--without
postgres')

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

```
# Run database migrations

sudo -u git -H bundle exec rake db:migrate RAILS_ENV=production

# Clean up assets and cache

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

MySQL installations:

```
# Install the database packages

sudo apt-get install -y mysql-server mysql-client libmysqlclient-dev


# Ensure you have MySQL version 5.6 or later

mysql --version


# Secure your installation

sudo mysql_secure_installation

# Login to MySQL

mysql -u root -p

# Create a user for GitLab ya estaba creado

CREATE USER 'git'@'localhost' IDENTIFIED BY '$Password';

# Ensure you can use the InnoDB engine which is necessary to support
long indexes

SET storage_engine=INNODB;


# If you have MySQL < 5.7.7 and want to enable utf8mb4 character set
support with your GitLab install, you must set the following NOW:

SET GLOBAL innodb_file_per_table=1, innodb_file_format=Barracuda,
innodb_large_prefix=1;


# If you use MySQL with replication, or just have MySQL configured
with binary logging, you need to run the following to allow the use of
`TRIGGER`:

SET GLOBAL log_bin_trust_function_creators = 1;


# Create the GitLab production database

CREATE DATABASE IF NOT EXISTS `gitlab_production` DEFAULT CHARACTER
SET `utf8` COLLATE `utf8_general_ci`;


# Ojo mi base de datos no se llama gitlabhq_production

# Grant the GitLab user necessary permissions on the database
```

```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, CREATE TEMPORARY TABLES,
DROP, INDEX, ALTER, LOCK TABLES, REFERENCES, TRIGGER ON
`gitlab_production`.* TO 'git'@'localhost';


# Quit the database session

\q

# Try connecting to the new database with the new user

sudo -u git -H mysql -u git -p -D gitlab_production

# Check for InnoDB File-Per-Table Tablespaces

use gitlab_production;


# Check your MySQL version is >= 5.5.3 (GitLab requires 5.5.14+)

SHOW VARIABLES LIKE 'version';

# Note where is your MySQL data dir for later:

select @@datadir;

# Note whether your MySQL server runs with innodb_file_per_table ON or
OFF:

SELECT @@innodb_file_per_table;
```

If SELECT @@innodb_file_per_table returned 1 previously, your server is running correctly.

Now, persist **innodb_file_per_table** **and** **innodb_file_format** **in your my.cnf file. Ensure at this stage that your GitLab instance is indeed *stopped*.**

```
mysql -u root -p

use gitlab_production;

# Safety check: you should still have those values set as follows:

SELECT @@innodb_file_per_table, @@innodb_file_format;

SELECT CONCAT('ALTER TABLE `', TABLE_NAME,'` ENGINE=InnoDB;') AS 'Copy
& run these SQL statements:' FROM INFORMATION_SCHEMA.TABLES WHERE
TABLE_SCHEMA="gitlab_production" AND TABLE_TYPE="BASE TABLE";


# Check for proper InnoDB File Format, Row Format, Large Prefix and
tables conversión

SET GLOBAL innodb_file_format=Barracuda, innodb_large_prefix=1;


# This should return the following:

SELECT @@character_set_database, @@collation_database;

# Convert tables not using ROW_FORMAT DYNAMIC:
```

```
SELECT CONCAT('ALTER TABLE `', TABLE_NAME,'` ROW_FORMAT=DYNAMIC;') AS
'Copy & run these SQL statements:' FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_SCHEMA="gitlab_production" AND TABLE_TYPE="BASE TABLE" AND
ROW_FORMAT!="Dynamic";

# Convert tables/columns not using utf8mb4/utf8mb4_general_ci as
encoding/collation:

SET foreign_key_checks = 0;

SELECT CONCAT('ALTER TABLE `', TABLE_NAME,'` CONVERT TO CHARACTER SET
utf8mb4 COLLATE utf8mb4_general_ci;') AS 'Copy & run these SQL
statements:' FROM INFORMATION_SCHEMA.TABLES WHERE
TABLE_SCHEMA="gitlab_production" AND TABLE_COLLATION !=
"utf8mb4_general_ci" AND TABLE_TYPE="BASE TABLE";


# Turn foreign key checks back on

SET foreign_key_checks = 1;

# You can now quit the database session

\q

cd /home/git/gitlab

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

# Modificar los siguientes campos:

production:
  adapter: mysql2
  encoding: utf8mb4
  collation: utf8mb4_general_ci


# Installations from source

bundle exec rake add_limits_mysql RAILS_ENV=production
```

6. Update gitlab-workhorse

Install and compile gitlab-workhorse. This requires Go 1.5 which should already be on your system from GitLab 8.1.

Comprobar versión https://gitlab.com/gitlab-org/gitlab-ce/blob/8-16-stable/GITLAB_WORKHORSE_VERSION

```
cd /home/git/gitlab-workhorse

sudo -u git -H git fetch --all

sudo -u git -H git checkout v1.3.0

sudo -u git -H make
```

7. Update gitlab-shell

```
cd /home/git/gitlab-shell
sudo -u git -H git fetch --all --tags
sudo -u git -H git checkout v4.1.1
```

8. Update configuration files

```
# New configuration options gitlab.yml.
```

```
cd /home/git/gitlab/config
mv gitlab.yml gitlab.8.15.8
sudo nano gitlab.yml
```

```
#Git configuration
```

```
sudo -u git -H git config --global repack.writeBitmaps true
```

```
# Nginx configuration
```

```
cd /etc/nginx/sites-available
mv gitlab-ssl gitlab-ssl.8.15.8
sudo nano gitlab-ssl
mv gitlab gitlab.8.15.8
sudo nano gitlab
# Init script
```

```
cd /home/git/gitlab
sudo cp lib/support/init.d/gitlab /etc/init.d/gitlab
```

9. Start application

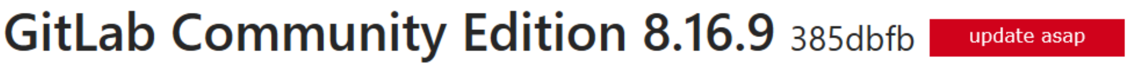
```
sudo service gitlab start
sudo service nginx restart
```

10. Check application status

Check if GitLab and its environment are configured correctly:

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

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



GitLab Community Edition 8.16.9 385dbfb update asap

Actualizar Git de 8.16.9 a 8.17

https://gitlab.com/gitlab-org/gitlab-ce/blob/11-8-stable/doc/update/8.16-to-8.17.md

GitLab Community Edition 8.16.9 385dbfb update asap

1. Stop server

```
sudo service gitlab stop
```

2. Backup

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

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

3. Update Ruby

Actualiado en la versión anterior

4. Update Node

GitLab now runs webpack to compile frontend assets and it has a minimum requirement of node v4.3.0.

You can check which version you are running with `node -v`.

Tengo la versión v6.15.0

5. Get latest code

```
cd /home/git/gitlab
sudo -u git -H git fetch --all
sudo -u git -H git checkout -- db/schema.rb
sudo -u git -H git checkout 8-17-stable
```

6. Install libs, migrations, etc.

```
cd /home/git/gitlab

# MySQL installations (note: the line below states '--without
postgres')

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


# Run database migrations
```

```
sudo -u git -H bundle exec rake db:migrate RAILS_ENV=production
```

# Install/update frontend asset dependencies

```
sudo -u git -H npm install --production
```

# Clean up assets and cache

```
sudo -u git -H bundle exec rake gitlab:assets:clean
gitlab:assets:compile cache:clear RAILS_ENV=production
```

# MySQL installations:

Hecho en la versión anterior (8.16)

7. Update gitlab-workhorse

Es la misma versión que la anterior?

```
sudo -u git -H bundle exec rake
"gitlab:workhorse:install[/home/git/gitlab-workhorse]"
RAILS_ENV=production    → no funciona
```

Comprobar versión https://gitlab.com/gitlab-org/gitlab-ce/blob/8-17-stable/GITLAB_WORKHORSE_VERSION

```
cd /home/git/gitlab-workhorse

sudo -u git -H git fetch --all

sudo -u git -H git checkout v1.3.0

sudo -u git -H make
```

8. Update gitlab-shell

```
cd /home/git/gitlab-shell
sudo -u git -H git fetch --all --tags
sudo -u git -H git checkout v4.1.1
```

9. Update configuration files

# New configuration options for gitlab.yml.

```
cd /home/git/gitlab/config
mv gitlab.yml gitlab.8.16.9
sudo nano gitlab.yml
```

# Git configuration

```
cd /home/git/gitlab
sudo -u git -H git config --global repack.writeBitmaps true
```

```
# Nginx configuration

cd /etc/nginx/sites-available
mv gitlab-ssl gitlab-ssl.8.16.9
sudo nano gitlab-ssl
mv gitlab gitlab.8.16.9
sudo nano gitlab
```

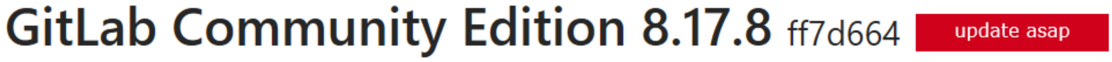
```
# Init script
cd /home/git/gitlab

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

10. Start application

```
sudo service gitlab start
sudo service nginx restart
```

11. Check application status

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



GitLab Community Edition 8.17.8 ff7d664

Actualizar Git de 8.17.8 a 9.0.13

https://gitlab.com/gitlab-org/gitlab-ce/blob/11-8-stable/doc/update/8.17-to-9.0.md

GitLab Community Edition 8.17.8 ff7d664 update asap

0. Update to Debian 8.1

```
# Edit the /etc/apt/sources.list file again:

nano /etc/apt/sources.list


# and replace its content with the following lines:

deb http://ftp.de.debian.org/debian/ jessie main contrib non-free
deb-src http://ftp.de.debian.org/debian/ jessie main contrib non-free


deb http://httpredir.debian.org/debian jessie-updates main contrib non-free
deb-src http://httpredir.debian.org/debian jessie-updates main contrib non-free


deb http://security.debian.org/ jessie/updates main contrib non-free
deb-src http://security.debian.org/ jessie/updates main contrib non-free


# Guardar archivo y ejecutar:

apt-get update
apt-get upgrade

# En las preguntas decir Y a todo
```

1. Stop server

```
sudo service gitlab stop
```

2. Backup

```
cd /home/git/gitlab

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

3. Update Ruby

Ya estoy en la versión 2.3.x

4. Update Node

Estoy en la version 6.15.0

Since 8.17, GitLab requires the use of yarn `>= v0.17.0` to manage JavaScript dependencies.

```
curl --silent --show-error https://dl.yarnpkg.com/debian/pubkey.gpg |
sudo apt-key add -
echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee
/etc/apt/sources.list.d/yarn.list
sudo apt-get update
sudo apt-get install yarn
```

5. Get latest code

```
cd /home/git/gitlab

sudo -u git -H git fetch --all
sudo -u git -H git checkout -- db/schema.rb
sudo -u git -H git checkout 9-0-stable
```

6. Update gitlab-shell

```
cd /home/git/gitlab-shell

sudo -u git -H git fetch --all --tags
sudo -u git -H git checkout v$(</home/git/gitlab/GITLAB_SHELL_VERSION)
```

7. Update gitlab-workhorse

```
cd /home/git/gitlab-workhorse

sudo -u git -H git fetch --all --tags
sudo -u git -H git checkout
v$(</home/git/gitlab/GITLAB_WORKHORSE_VERSION)
sudo -u git -H make
```

8. Update configuration files

```
# New configuration options for gitlab.yml
```

```
cd /home/git/gitlab/config
mv gitlab.yml gitlab.8.17.8
sudo nano gitlab.yml
```

Configuration changes for repository storages

This version introduces a new configuration structure for repository storages. Update your current configuration as follows, replacing with your storages names and paths:

For installations from source

1. Update your `gitlab.yml`, from

```
   repositories:
     storages: # You must have at least a 'default' storage path.
       default: /home/git/repositories
       nfs: /mnt/nfs/repositories
       cephfs: /mnt/cephfs/repositories
```

to

```
   repositories:
        storages: # You must have at least a 'default' storage path.
          default:
            path: /home/git/repositories
          nfs:
            path: /mnt/nfs/repositories
          cephfs:
            path: /mnt/cephfs/repositories
```

```
# Git configuration

cd /home/git/gitlab

sudo -u git -H git config --global repack.writeBitmaps true

# Nginx configuration

cd /etc/nginx/sites-available

mv gitlab-ssl gitlab-ssl.8.17.8

sudo nano gitlab-ssl

mv gitlab gitlab.8.17.8

sudo nano gitlab


# Init script

cd /home/git/gitlab
sudo cp lib/support/init.d/gitlab /etc/init.d/gitlab
```

9. Install libs, migrations, etc.

GitLab 9.0.11 introduced a dependency on the `re2` regular expression library. To install this dependency:

```
sudo apt-get install libre2-dev
# → comando no funciona con wheezy 7 hay que actualizar a 8 si o si
cd /home/git/gitlab

# MySQL installations (note: the line below states '--without
postgres')

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


# Run database migrations

sudo -u git -H bundle exec rake db:migrate RAILS_ENV=production
```

```
# Update node dependencies and recompile assets

sudo -u git -H bundle exec rake yarn:install gitlab:assets:clean
gitlab:assets:compile RAILS_ENV=production NODE_ENV=production

# Clean up cache

sudo -u git -H bundle exec rake cache:clear RAILS_ENV=production
```

MySQL installations:

# cambiado en versión 8.16

10. Install Gitaly

```
# Fetch Gitaly source with Git and compile with Go

cd /home/git/gitlab
sudo -u git -H bundle exec rake
"gitlab:gitaly:install[/home/git/gitaly,/home/git/repositories]"
RAILS_ENV=production


# Restrict Gitaly socket access

sudo chmod 0700 /home/git/gitlab/tmp/sockets/private
sudo chown git /home/git/gitlab/tmp/sockets/private

# If you are using non-default settings you need to update config.toml

cd /home/git/gitaly
sudo -u git -H editor config.toml
```

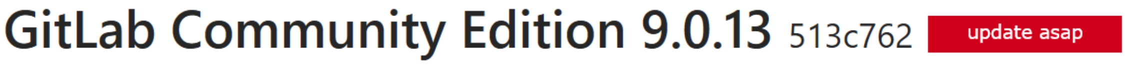
11. Start application

```
sudo service gitlab start
sudo service nginx restart
```

12. Check application status

Check if GitLab and its environment are configured correctly:

```
cd /home/git/gitlab

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



GitLab Community Edition 9.0.13 513c762 update asap

Actualizar Git de 9.0.13 a 9.3

https://gitlab.com/gitlab-org/gitlab-ce/blob/11-8-stable/doc/update/9.2-to-9.3.md

GitLab Community Edition 9.0.13 513c762 update asap

1. Stop server

```
sudo service gitlab stop
```

2. Backup

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

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

3. Update Ruby

`No es necesaria, Actualizado a la versión 2.3.3`

4. Update Node

No es necesaria, Node actualizado a la versión 6.15.0 anteriormente

Since 8.17, GitLab requires the use of yarn `>= v0.17.0` to manage JavaScript dependencies.

```
curl --silent --show-error https://dl.yarnpkg.com/debian/pubkey.gpg |
sudo apt-key add -
echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee
/etc/apt/sources.list.d/yarn.list
sudo apt-get update
sudo apt-get install yarn
```

no actualiza!!

More information can be found on the yarn website.

5. Update Go

NOTE: GitLab 9.2 and higher only supports Go 1.8.3 and dropped support for Go 1.5.x through 1.7.x. Be sure to upgrade your installation if necessary.

You can check which version you are running with `go version`.

Download and install Go:

```
# Remove former Go installation folder
sudo rm -rf /usr/local/go
```

```
curl --remote-name --progress
https://storage.googleapis.com/golang/go1.8.3.linux-amd64.tar.gz
echo '1862f4c3d3907e59b04a757cfda0ea7aa9ef39274af99a784f5be843c80c6772
go1.8.3.linux-amd64.tar.gz' | shasum -a256 -c - && \
  sudo tar -C /usr/local -xzf go1.8.3.linux-amd64.tar.gz
sudo ln -sf /usr/local/go/bin/{go,godoc,gofmt} /usr/local/bin/
rm go1.8.3.linux-amd64.tar.gz
```

6. Get latest code

```
cd /home/git/gitlab

sudo -u git -H git fetch --all
sudo -u git -H git checkout -- db/schema.rb
sudo -u git -H git checkout -- locale
sudo -u git -H git checkout 9-3-stable
```

7. Update gitlab-shell

```
cd /home/git/gitlab-shell

sudo -u git -H git fetch --all --tags
sudo -u git -H git checkout v$(</home/git/gitlab/GITLAB_SHELL_VERSION)
sudo -u git -H bin/compile
```

8. Update gitlab-workhorse

```
cd /home/git/gitlab-workhorse

sudo -u git -H git fetch --all --tags
sudo -u git -H git checkout
v$(</home/git/gitlab/GITLAB_WORKHORSE_VERSION)
sudo -u git -H make
```

9. Update Gitaly

```
cd /home/git/gitaly
sudo -u git -H git fetch --all --tags
sudo -u git -H git checkout
v$(</home/git/gitlab/GITALY_SERVER_VERSION)
sudo -u git -H make

sudo mv env env.old
sudo -u git cp config.toml.example config.toml
```

10. Update MySQL permissions

If you are using MySQL you need to grant the GitLab user the necessary permissions on the database:

```
mysql -u root -p -e "GRANT TRIGGER ON \`gitlab_production\`.* TO
'git'@'localhost';"
```

If you use MySQL with replication, or just have MySQL configured with binary logging, you will need to also run the following on all of your MySQL servers:

```
mysql -u root -p -e "SET GLOBAL log_bin_trust_function_creators = 1;"
```

You can make this setting permanent by adding it to your `my.cnf`:

```
log_bin_trust_function_creators=1
```

11. Update configuration files

New configuration options for gitlab.yml.:

```
cd /home/git/gitlab/config
mv gitlab.yml gitlab.9.0.13
sudo nano gitlab.yml
```

Nginx configuration

```
cd /etc/nginx/sites-available
mv gitlab-ssl gitlab-ssl.9.0.13
sudo nano gitlab-ssl
mv gitlab gitlab.9.0.13
sudo nano gitlab
```

Init script

```
cd /home/git/gitlab

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

12. Install libs, migrations, etc.

```
cd /home/git/gitlab

# MySQL installations (note: the line below states '--without postgres')

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

# Run database migrations

sudo -u git -H bundle exec rake db:migrate RAILS_ENV=production

# Compile GetText PO files


sudo -u git -H bundle exec rake gettext:compile RAILS_ENV=production

# Update node dependencies and recompile assets

sudo -u git -H bundle exec rake yarn:install gitlab:assets:clean gitlab:assets:compile RAILS_ENV=production NODE_ENV=production
```

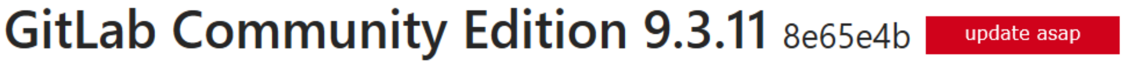
```
# Clean up cache

sudo -u git -H bundle exec rake cache:clear RAILS_ENV=production
```

13. Start application

```
sudo service gitlab start
sudo service nginx restart
```

14. Check application status

```
cd /home/git/gitlab

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



GitLab Community Edition 9.3.11 8e65e4b update asap

1. Instalar versión 9.3 en el [New Server]

We Need version (v9.3.11 - to match your old server). To repost the steps to downgrade GitLab version:

```
gitlab-ctl cleanse
apt remove gitlab-ce
apt update
apt upgrade
apt install gitlab-ce=9.3.11-ce.0
rm -rf /opt/gitlab/etc/gitaly/env
sudo gitlab-ctl reconfigure
```

2. Hacer Backup en [Old Server]

```
cd /home/git/gitlab


sudo -u git -H bundle exec rake gitlab:backup:create SKIP=db
RAILS_ENV=production
```

se guarda en cd /home/git/gitlab/tmp/backups

ver lo que ocupa ***ls -lh***

3. Transferir la copia de seguridad al [New Server] desde el [Old Server]

# Once you have your backup, then transfer it, and your config files to the new server. I'll show copying from the OLD server to the new one:

```
scp
/home/git/gitlab/tmp/backups/1555502686_2019_04_17_9.3.11_gitlab_backu
p.tar root@172.22.0.190:/root/

scp /home/git/gitlab/config/gitlab.yml root@172.22.0.190:/root/
scp /home/git/gitlab/config/secrets.yml root@172.22.0.190:/root/
scp /home/git/gitlab/.secret root@172.22.0.190:/root/
scp /home/git/gitlab/.gitlab_shell_secret root@172.22.0.190:/root/
```

4. Migrate DB - install pgloader [New Server]

Next update pgloader v3.4.1+. Do that like this:

```
apt update
cd /tmp
```

```
wget
http://apt.postgresql.org/pub/repos/apt/pool/main/p/pgloader/pgloader_
3.5.2-3.pgdg90%2b1_amd64.deb
apt install /tmp/pgloader_3.5.2-3.pgdg90+1_amd64.deb

rm -rf pgloader_3.5.2-3.pgdg90+1_amd64.deb
```

en el viejo

```
# Add repository
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/
$(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'

# Add key

sudo apt-get install wget ca-certificates
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc |
sudo apt-key add -

# Install package
sudo apt-get update
sudo apt-get install pgloader
```

5. Migrar DB – Crear base de datos [New Server]

```
# Install the database packages:

cd /tmp
sudo apt-get install -y postgresql postgresql-client libpq-dev
postgresql-contrib

# Create a database user for GitLab:

sudo -u postgres psql -d template1 -c "CREATE USER git CREATEDB;"

# Create the pg_trgm extension (required for GitLab 8.6+):

sudo -u postgres psql -d template1 -c "CREATE EXTENSION IF NOT EXISTS
pg_trgm;"

# Create the GitLab production database and grant all privileges on
database:

sudo -u postgres psql -d template1 -c "CREATE DATABASE
gitlabhq_production OWNER git;"

# Try connecting to the new database with the new user:

sudo -u git -H psql -d gitlabhq_production

# Check if the pg_trgm extension is enabled:

SELECT true AS enabled
FROM pg_available_extensions
WHERE name = 'pg_trgm'
AND installed_version IS NOT NULL;
```

```
# If the extension is enabled this will produce the following output:
 enabled
 ---------
  t
 (1 row)

# Quit the database session:
gitlabhq_production> \q


# Activar la base de datos de postgresql. Stop GitLab:

gitlab-ctl stop

# Edit /etc/gitlab/gitlab.rb to enable bundled PostgreSQL:

sudo nano /etc/gitlab/gitlab.rb

postgresql['enable'] = true


gitlab_rails['db_adapter'] = "postgresql"
gitlab_rails['db_encoding'] = "unicode"
gitlab_rails['db_collation'] = nil
gitlab_rails['db_database'] = "gitlabhq_production"
gitlab_rails['db_pool'] = 10
gitlab_rails['db_username'] = "gitlab"
gitlab_rails['db_password'] = nil
gitlab_rails['db_host'] = nil
gitlab_rails['db_port'] = 5432
gitlab_rails['db_socket'] = nil
gitlab_rails['db_sslmode'] = nil
gitlab_rails['db_sslrootcert'] = nil
gitlab_rails['db_prepared_statements'] = true
gitlab_rails['db_statements_limit'] = 1000



# preparar la base de datos para no tener problemas con los permisos:

sudo gitlab-ctl reconfigure

sudo su - gitlab-psql
/bin/bash
psql -h /var/opt/gitlab/postgresql -d template1
ALTER USER gitlab CREATEDB;
\q
exit


# Reconfiguramos Gitlab

sudo gitlab-ctl reconfigure
gitlab-ctl stop

# Start Unicorn and PostgreSQL so that we can prepare the schema:

sudo gitlab-ctl start unicorn
sudo gitlab-ctl start postgresql
```

```
# Run the following commands to prepare the schema:

sudo gitlab-rake db:create db:migrate

# Stop Unicorn to prevent other database access from interfering with
the loading of data:

sudo gitlab-ctl stop unicorn
```

6. Habilitar conexiones remotas en mysql en el [Old Server]:

```
sed -i "s/^bind-address/#bind-address/" /etc/mysql/my.cnf


# Configure MySQL to accept remote root connections (from any host
('%')): (added line breaks for readability)

MYSQL_BATCH="mysql --user=root --password=$mypassword --batch"

$MYSQL_BATCH --execute "INSERT INTO mysql.user ( Host , User ,
Password , Select_priv ,
    Insert_priv , Update_priv , Delete_priv , Create_priv , Drop_priv
, Reload_priv ,
    Shutdown_priv , Process_priv , File_priv , Grant_priv ,
References_priv ,
    Index_priv , Alter_priv , Show_db_priv , Super_priv ,
Create_tmp_table_priv ,
    Lock_tables_priv , Execute_priv , Repl_slave_priv ,
Repl_client_priv ,
    Create_view_priv , Show_view_priv , Create_routine_priv ,
Alter_routine_priv ,
    Create_user_priv , ssl_type , max_questions , max_updates ,
max_connections ,
    max_user_connections) VALUES ( '%', 'root', '', 'Y', 'Y', 'Y',
'Y', 'Y', 'Y',
    'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y','Y', 'Y',
'Y', 'Y', 'Y',
    'Y', 'Y', 'Y', 'Y', '', '0', '0', '0', '0');"


# Restart MySQL for the changes to take effect

service mysql restart
```

Migrate DB - test access to MySQL from NEW server

```
apt install mysql-client
mysql -u root -h 172.22.0.99 gitlab_production

# ver tablas (87 rows)

SHOW TABLES;
```

```
# en postgres: sudo -u gitlab-psql psql -h /var/opt/gitlab/postgresql
-d gitlabhq_production

\dt  → ver relaciones
\l  → ver bases de datos

# (Swap out 'USERNAME', 'PASSWORD' and 'OLD_SERVER_IP' for the real
values you're using)

You should see something like this:

Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 10.1.26-MariaDB-0+deb9u1 Debian 9.1

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

MariaDB [(none)]>

Assuming that works ok, you can close the client (Ctrl-C to exit).
```

7. Migrate DB - prepare to migrate data [New Server]

```
# create the pgloader commands.load file

MYSQL_USER=root
MYSQL_PASS=
OLD_SERVER_IP=172.22.0.99

cat > /tmp/commands.load <<EOF
LOAD DATABASE
     FROM mysql://$MYSQL_USER@$OLD_SERVER_IP/gitlab_production
     INTO postgresql://gitlab-
psql@unix://var/opt/gitlab/postgresql:/gitlabhq_production

 WITH include no drop, truncate, disable triggers, create no tables,
      create no indexes, preserve index names, no foreign keys,
      data only

 ALTER SCHEMA 'gitlab_production' RENAME TO 'public'

 ;
EOF
```

8. Migrate DB - do the migration Nuevo

First ensure that everyone has access to the commands.load file, then to start the DB migration:

```
cd /tmp

chmod 777 /tmp/commands.load
sudo -u gitlab-psql pgloader /tmp/commands.load
```

9. Restore your your backup

```
# start GitLab again

gitlab-ctl start

# copy your backup into the backup dir & grant correct ownership
cp /root/1556531672_2019_04_29_9.3.11_gitlab_backup.tar
/var/opt/gitlab/backups/

chown git:git
/var/opt/gitlab/backups/1556531672_2019_04_29_9.3.11_gitlab_backup.tar

# Stop the processes that are connected to the database. Leave the
rest of GitLab running:
sudo gitlab-ctl stop unicorn
sudo gitlab-ctl stop sidekiq

# Verify

sudo gitlab-ctl status

# Next, restore the backup, specifying the timestamp of the backup you
wish to restore. This command will overwrite the contents of your
GitLab database!

sudo gitlab-rake gitlab:backup:restore
BACKUP=1556531672_2019_04_29_9.3.11

# Next, restore /etc/gitlab/gitlab-secrets.json if necessary as
mentioned above.

sudo cp gitlab-secrets.json /etc/gitlab  # → no funciona

# Restart and check GitLab:

sudo gitlab-ctl restart
sudo gitlab-rake gitlab:check SANITIZE=true

# If there is a GitLab version mismatch between your backup tar file
and the installed version of GitLab, the restore command will abort
with an error. Install the correct GitLab version and try again.
```

Restore for Docker image and GitLab helm chart installations

For GitLab installations using the Docker image or the GitLab helm chart on a Kubernetes cluster, the restore task expects the restore directories to be empty. However, with docker and Kubernetes volume mounts, some system level directories may be created at the volume roots, like `lost+found` directory found in Linux operating systems. These directories are usually owned by `root`, which can cause access permission errors since the restore rake task runs as `git` user. So, to restore a GitLab installation, users have to confirm the restore target directories are empty.

For both these installation types, the backup tarball has to be available in the backup location (default location is `/var/opt/gitlab/backups`).

For docker installations, the restore task can be run from host:

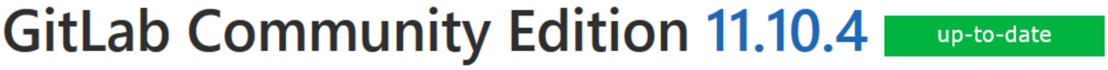
```
docker exec -it <name of container> gitlab-rake gitlab:backup:restore
```

Update to latest GitLab

Final step is to update to the latest GitLab. This should now be super easy! Just like this:

```
sudo gitlab-rake gitlab:env:info
sudo gitlab-ctl restart
```

```
#  actualizar a 9.5

apt install gitlab-ce=9.5.10-ce.0
apt install gitlab-ce=10.8.7-ce.0
apt install gitlab-ce=11.10.4-ce.0
```



GitLab Community Edition 11.10.4 up-to-date

Comprobar versión gitlab disponible

```
apt update
apt-cache policy gitlab-ce | head -3
```