

# FreeBSD Administration - Support #944

## Install GitLab 11.5 on FreeBSD

12/01/2018 06:30 PM - Daniel Curtis

<b>Status:</b>	Closed	<b>Start date:</b>	12/01/2018
<b>Priority:</b>	Normal	<b>Due date:</b>	
<b>Assignee:</b>	Daniel Curtis	<b>% Done:</b>	50%
<b>Category:</b>	Source Code Management	<b>Estimated time:</b>	2.50 hours
<b>Target version:</b>	FreeBSD 11	<b>Spent time:</b>	0.00 hour

### Description

This is a guide on installing GitLab 11 from source with continuous integration on FreeBSD 11.2-RELEASE.

## Prepare the Environment

- Make sure the system is up to date:

```
pkg update && pkg upgrade
```

- Install a few dependencies:

```
pkg install sudo bash icu cmake pkgconf git node ruby ruby24-gems logrotate postfix krb5 go wget rubygem-bundler rubygem-gpgme rubygem-rugged rubygem-debug_inspector portmaster postgresql96-server postgresql96-contrib redis gmake mercurial
```

- Add the GitLab user

```
pw add user -n git -m -s /usr/local/bin/bash -c "GitLab"
```

- Add git to the redis group:

```
pw usermod git -G redis
```

## Configure Postfix

- Enable postfix as the mail delivery agent, and disable sendmail:

```
sysrc postfix_enable="YES"
sysrc sendmail_enable="NO"
sysrc sendmail_submit_enable="NO"
sysrc sendmail_outbound_enable="NO"
sysrc sendmail_msp_queue_enable="NO"
```

## Configure PostgreSQL

- Initialize, start, and enable postgresql at boot:

```
sysrc postgresql_enable="YES"
service postgresql initdb
service postgresql start
```

- Log in to postgresql user account

```
su - postgres
```

- Connect to postgresql database

```
psql -d template1
```

- Create a user for GitLab:

```
CREATE USER git WITH PASSWORD 'SuperSecretPassword' CREATEDB;
```

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

```
CREATE DATABASE gitlabhq_production OWNER git encoding='UTF8';
```

- Quit the database session

```
\q  
exit
```

## Configure Redis

- Back up the original Redis config file:

```
cp /usr/local/etc/redis.conf /usr/local/etc/redis.conf.orig
```

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

```
sed -i '' -e 's/^port .*/port 0/' /usr/local/etc/redis.conf
```

- Enable Redis socket

```
echo 'unixsocket /usr/local/var/run/redis/redis.sock' >> /usr/local/etc/redis.conf
```

- Grant permission to the socket to all members of the redis group

```
echo 'unixsocketperm 770' >> /usr/local/etc/redis.conf
```

- Create the directory which contains the socket

```
mkdir -p /usr/local/var/run/redis  
chown redis:redis /usr/local/var/run/redis  
chmod 755 /var/run/redis
```

- Start and enable redis at boot:

```
sysrc redis_enable="YES"
```

```
service redis start
```

## Install GitLab 11.5

- Switch to the git user:

```
su - git
```

- Clone GitLab repository:

```
git clone https://gitlab.com/gitlab-org/gitlab-ce.git -b 11-5-stable gitlab
cd gitlab
```

### Edit lib/tasks/gitlab/setup.rake and remove line 4 (check\_gitaly\_connection)

- Copy the example GitLab config:

```
cp config/gitlab.yml.example config/gitlab.yml
```

- Update GitLab config file, follow the directions at top of file:

```
vi config/gitlab.yml
```

- And modify at least the following parameters:

```
gitlab:
  host: gitlab.example.com
  email_from: gitlab@example.com
```

```
git:
  bin_path: /usr/local/bin/git
```

- Copy the example secrets file:

```
cp config/secrets.yml.example config/secrets.yml
chmod 0600 config/secrets.yml
```

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

```
chown -R git log/
chown -R git tmp/
chmod -R u+rwx,go-w log/
chmod -R u+rwx tmp/
```

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

```
chmod -R u+rwx tmp/pids/
chmod -R u+rwx tmp/sockets/
```

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

```
mkdir public/uploads/  
chmod 0700 public/uploads
```

- Change the permissions of the directory where CI build traces are stored:

```
chmod -R u+rwx builds/
```

- Change the permissions of the directory where CI artifacts are stored:

```
chmod -R u+rwx shared/artifacts/
```

1. Change the permissions of the directory where GitLab Pages are stored:

```
chmod -R ug+rwx shared/pages/
```

- Copy the example Unicorn config:

```
cp config/unicorn.rb.example config/unicorn.rb
```

- Copy the example Rack attack config:

```
cp config/initializers/rack_attack.rb.example config/initializers/rack_attack.rb
```

- Configure Git global settings for git user, used when editing via web editor:

```
git config --global core.autocrlf input
```

- Disable 'git gc --auto' because GitLab already runs 'git gc' when needed:

```
git config --global gc.auto 0
```

- Enable packfile bitmaps:

```
git config --global repack.writeBitmaps true
```

- Enable push options:

```
git config --global receive.advertisePushOptions true
```

- Configure resque connection settings

```
cp config/resque.yml.example config/resque.yml
```

- Change the resque socket path if you are not using the default Debian / Ubuntu configuration

```
vi config/resque.yml
```

- And modify the following parameter:

```
production: unix:/usr/local/var/run/redis/redis.sock
```

- Configure GitLab DB Settings:

```
cp config/database.yml.postgresql config/database.yml
```

- Change the database credentials if necessary:

```
vi config/database.yml
```

- Make config/database.yml readable to git only

```
chmod o-rwx config/database.yml
```

- Configure the gpgme, rugged and re2 rubygem to use the installed library (to prevent hanging up during install):

```
bundle config build.gpgme "--use-system-libraries"
bundle config build.rugged "--use-system-libraries"
bundle config build.re2 "--use-system-libraries"
bundle config build.charlock_holmes "--use-system-libraries"
```

- Install the gems for PostgreSQL:

```
bundle install --deployment --without development test mysql aws kerberos
```

## Install GitLab Shell

- Run the installation task for gitlab-shell:

```
bundle exec rake gitlab:shell:install RAILS_URL=unix:/usr/local/var/run/redis/redis.sock RAILS_ENV=production SKIP_STORAGE_VALIDATION=true
```

- By default, the gitlab-shell config is generated from your main GitLab config, double check the settings are correct:

```
vi /home/git/gitlab-shell/config.yml
```

## Install GitLab Workhorse

- Install the gitlab workhorse:

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

## Install gitlab-pages

- This step is optional and only needed if you wish to host static sites from within GitLab:

```
cd /home/git
git clone https://gitlab.com/gitlab-org/gitlab-pages.git
cd gitlab-pages
git checkout v$(</home/git/gitlab/GITLAB_PAGES_VERSION)
gmake
```

## Install Gitaly

- ~~Fetch Gitaly source with Git and compile with Go~~

```
bundle exec rake "gitlab:gitaly:install[/home/git/gitaly,/home/git/repositories]" RAILS_ENV=production
```

- Install the binary package for gitaly (I had issues during the build from source):

```
exit
pkg install gitaly
chown -R git:git /usr/local/share/gitaly/
su - git
```

- Create the gitaly config:

```
vi /usr/local/share/gitaly/config.toml
```

- And add the following:

```
bin_dir = "/home/git/gitaly/bin"
socket_path = "/home/git/gitlab/tmp/sockets/private/gitaly.socket"
[gitaly-ruby]
dir = "/home/git/gitaly/ruby"
[gitlab-shell]
dir = "/home/git/gitlab-shell"
[[storage]]
name = "default"
path = "/home/git/repositories"
```

- Edit the gitlab.yml config:

```
vi /home/git/gitlab/config/gitlab.yml
```

- And change the gitaly path:

```
gitaly:
  client_path: /usr/local/share/gitaly/bin
```

- Restrict Gitaly socket access

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

## Initialize Database

- Initialize the database:

```
cd /home/git/gitlab  
bundle exec rake gitlab:setup RAILS_ENV=production
```

- Exit out of the git user, back into root:

```
exit
```

## Install Init Script

- Download the init script:

```
wget -O /usr/local/etc/rc.d/gitlab https://gitlab.com/gitlab-org/gitlab-recipes/raw/master/init/freebsd/gitlab-unicorn
```

- Fix the shell environment for the init script:

```
sed -i '' -e 's/\#\!\\ \ /bin\\ /sh/\#\!\\ \ /usr\\ /local\\ /bin\\ /bash' /usr/local/etc/rc.d/gitlab
```

- Make the init script executable:

```
chmod +x /usr/local/etc/rc.d/gitlab
```

## Check Configuration and Compile Assets

- Check the configuration:

```
cd /home/git/gitlab  
bundle exec rake gitlab:env:info RAILS_ENV=production
```

- Compile all of the assets for GitLab:

```
bundle exec rake assets:precompile RAILS_ENV=production
```

- Start and enable gitlab at boot:

```
echo 'gitlab_enable="YES"' >> /etc/rc.conf  
service gitlab start
```

## Install Nginx

- Install nginx:

```
pkg install nginx
```

- Start and enable nginx at boot:

```
echo 'nginx_enable="YES"' >> /etc/rc.conf
service nginx start
```

- Create a configuration directory to make managing individual server blocks easier

```
mkdir /usr/local/etc/nginx/conf.d
```

- Edit the main nginx config file:

```
vi /usr/local/etc/nginx/nginx.conf
```

- And strip down the config file and add the include statement at the end to make it easier to handle various server blocks:

```
worker_processes 1;
error_log /var/log/nginx-error.log;

events {
    worker_connections 1024;
}

http {
    include mime.types;
    default_type application/octet-stream;
    sendfile on;
    keepalive_timeout 65;

    # Load config files from the /etc/nginx/conf.d directory
    include /usr/local/etc/nginx/conf.d/*.conf;
}
```

- Create the gitlab nginx config:

```
vi /usr/local/etc/nginx/conf.d/gitlab.example.com.conf
```

- And add the following:

```
upstream gitlab-workhorse {
    server unix:/home/git/gitlab/tmp/sockets/gitlab-workhorse.socket fail_timeout=0;
}

server {
    listen 80;
    server_name gitlab.example.com;
    server_tokens off;
    root /home/git/gitlab/public;
    access_log /var/log/gitlab.example.com-access.log;
    error_log /var/log/gitlab.example.com-error.log;

    location / {
        client_max_body_size 0;
        gzip off;

        proxy_read_timeout 300;
        proxy_connect_timeout 300;
        proxy_redirect off;
    }
}
```

```

proxy_http_version 1.1;

proxy_set_header Host $http_host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto $scheme;

proxy_pass http://gitlab-workhorse;
}
}

```

## Install GitLab Runner

- Download the binary for 64-bit systems:

```
wget -O /usr/local/bin/gitlab-ci-multi-runner https://gitlab-ci-multi-runner-downloads.s3.amazonaws.com/latest/binaries/gitlab-ci-multi-runner-freebsd-amd64
```

- **NOTE:** If the host architecture is 32-bit download the 386 version of the gitlab runner:

```
wget -O /usr/local/bin/gitlab-ci-multi-runner https://gitlab-ci-multi-runner-downloads.s3.amazonaws.com/latest/binaries/gitlab-ci-multi-runner-freebsd-386
```

- Give it permissions to execute:

```
chmod +x /usr/local/bin/gitlab-ci-multi-runner
```

- Open a web browser and go to <http://gitlab.example.com>, then log in as the admin user:
  - Username **root**, password 5iveL!fe
- Next navigate to <http://gitlab.example.com/ci/admin/runners> to get the token
- Then switch to the git user
- Finally register the gitlab runner instance:

```
gitlab-ci-multi-runner register --non-interactive --url "http://gitlab.example.com/ci" --registration-token "5777041dc9651d08ff77" --description "gitlab-ce-ruby-2.1" --executor "shell" builds_dir = "" shell = "bash"
```

## Resources

- <http://doc.gitlab.com/ce/install/installation.html>
- <https://github.com/gitlabhq/gitlab-recipes/blob/master/install/freebsd/freebsd-10.md>
- <https://gitlab.com/gitlab-org/gitlab-ci-multi-runner/blob/master/docs/development/README.md>
- <https://gitlab.com/gitlab-org/gitlab-ce/issues/47483>

## History

---

### #1 - 12/09/2018 09:55 PM - Daniel Curtis

- *Description updated*
- *Status changed from New to In Progress*
- *% Done changed from 0 to 30*

### #2 - 01/01/2019 09:33 PM - Daniel Curtis

- *Description updated*

- % Done changed from 30 to 50

**#3 - 01/01/2019 10:06 PM - Daniel Curtis**

- Description updated

**#4 - 01/01/2019 11:09 PM - Daniel Curtis**

- Description updated

**#5 - 01/01/2019 11:29 PM - Daniel Curtis**

- Description updated

**#6 - 01/04/2019 04:27 PM - Daniel Curtis**

- Description updated

**#7 - 01/04/2019 04:45 PM - Daniel Curtis**

- Description updated

**#8 - 02/18/2022 08:06 PM - Daniel Curtis**

- Status changed from In Progress to Closed