FreeBSD Administration - Support #596

Install GitLab CI on FreeBSD

04/11/2015 08:49 PM - Daniel Curtis

Status:	Rejected	**Start date:**	04/11/2015
Priority:	Normal	**Due date:**	
Assignee:	Daniel Curtis	**% Done:**	80%
Category:	Source Code Management	**Estimated time:**	4.00 hours
Target version:	FreeBSD 9	**Spent time:**	3.00 hours

Description

This is a guide for installing GitLab CI on FreeBSD 9.3.

NOTE: This issue has been rejected due to Continuous Integration being introduced in GitLab 8

Setting up the Environment

- Start by making sure everything is up to date:

```
pkg update && pkg upgrade
portsnap fetch extract
```

- Install portmaster:

```
cd /usr/ports/ports-mgmt/portmaster
make install clean
pkg2ng
```

- Install a few dependencies:

```
portmaster sysutils/rubygem-bundler ftp/wget ftp/curl textproc/libxml2 textproc/libxslt databa
ses/redis devel/icu devel/readline devel/git textproc/libyaml
```

- Install the libv8 gem system-wide:

```
gem install libv8
```

- Add the GitLab CI user

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

Install MariaDB 5.5

- This environment will be setup with MariaDB 5.5 for its MySQL server:

```
portmaster databases/mariadb55-server databases/mariadb55-client
```

  - Ensure you have MySQL version 5.5.14 or later

```
mysql --version
```

- Start and enable MySQL at boot:

```
echo 'mysql_enable="YES"' >> /etc/rc.conf
service mysql-server start
```

- Secure your installation:

```
mysql_secure_installation
```

- Login to MySQL

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

  - Create a user for GitLab CI

```
CREATE USER 'gitlab_ci'@'localhost' IDENTIFIED BY 'SuperSecretPassword';
```

    NOTE: Change SuperSecretPassword to what ever password desired
  - Ensure you can use the InnoDB engine which is necessary to support long indexes.

```
SET storage_engine=INNODB;
```

    NOTE: If this fails, check your MySQL config files (e.g. `/etc/mysql/*.cnf`, `/etc/mysql/conf.d/*`) for the setting "innodb = off"
  - Create the GitLab CI production database

```
CREATE DATABASE IF NOT EXISTS `gitlab_ci_production` DEFAULT CHARACTER SET `utf8` COLLATE
`utf8_unicode_ci`;
```

  - Grant the GitLab user necessary permissions on the table.

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

  - Quit the database session

```
quit
```

- Try connecting to the new database with the new user

```
mysql -h localhost -u gitlab_ci -p -D gitlab_ci_production
```

- You should now see a mysql> prompt, quit the database session:

```
quit
```

Install GitLab CI

- Switch to the GitLab CI user:

```
su - gitlab_ci
```

- Download GitLab CI:

```
git clone https://github.com/gitlabhq/gitlab-ci.git
```

- Checkout the latest stable 2.2 version

```
cd gitlab-ci
git checkout 2-2-stable
```

- Create a tmp and sockets directory inside application

```
mkdir -p tmp/pids
mkdir tmp/sockets
```

- Install dependencies

```
bundle --without development test postgresql --path vendor/bundle
```

  - **NOTE**: Installation with bundle will most likely fail the first time through, just run the following to fix:

```
bundle update
```

- Create the mysql db config

```
cp config/database.yml.mysql config/database.yml
```

- Edit the database config file:

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

  - And modify the database parameters:

```
production:
  adapter: mysql2
  encoding: utf8
  reconnect: false
  database: gitlab_ci_production
  pool: 5
  username: gitlab_ci
  password: "SuperSecretPassword"
  host: localhost
```

- Setup the database

```
bundle exec rake db:setup RAILS_ENV=production
```

- During the database setup a secret key will be generates, add this to the devise initializer :

```
vi config/initializers/devise.rb
```

- and add the following before the **end** statement at the bottom of the file

```
config.secret_key = 'dd3036798c2c74a3c41ea5b39b2593d417be8dfd62252219c158101cefdecea4d40a9
d28f46df0a664328ceb7965207046520c18197489aea1f79b53dd6a4418'
```

- And restart the database setup

```
bundle exec rake db:setup RAILS_ENV=production
```

- Setup schedules

```
bundle exec whenever -w RAILS_ENV=production
```

- Now exit from gitlab_ci user

```
exit
```

GitLab CI rc.d script

- Create the gitlab-ci rc script:

```
vi /usr/local/etc/rc.d/gitlab-ci
```

- And add the following:

```
#! /usr/local/bin/bash

# GITLAB CI
# Maintainer: @randx
# App Version: 2.2

### BEGIN INIT INFO
# Provides:          gitlab-ci
# Required-Start:    $local_fs $remote_fs $network $syslog redis-server
# Required-Stop:     $local_fs $remote_fs $network $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: GitLab CI
# Description:       GitLab CI
### END INIT INFO

APP_ROOT="/usr/home/gitlab_ci/gitlab-ci"
DAEMON_OPTS="-C $APP_ROOT/config/puma.rb -e production"
PID_PATH="$APP_ROOT/tmp/pids"
WEB_SERVER_PID="$PID_PATH/puma.pid"
SIDEKIQ_PID="$PID_PATH/sidekiq.pid"
STOP_SIDEKIQ="RAILS_ENV=production bundle exec rake sidekiq:stop"
START_SIDEKIQ="RAILS_ENV=production bundle exec rake sidekiq:start"
NAME="GitLab CI"
DESC="Gitlab CI service"

check_pid(){
  if [ -f $WEB_SERVER_PID ]; then
    PID=`cat $WEB_SERVER_PID`
    SPID=`cat $SIDEKIQ_PID`
    STATUS=`ps aux | grep $PID | grep -v grep | wc -l`
  else
    STATUS=0
```

```
    PID=0
  fi
}

start() {
  cd $APP_ROOT
  check_pid
  if [ "$PID" -ne 0 -a "$STATUS" -ne 0 ]; then
    # Program is running, exit with error code 1.
    echo "Error! $DESC is currently running!"
    exit 1
  else
    if [ `whoami` = root ]; then
      /usr/local/bin/sudo -u gitlab_ci -H /usr/local/bin/bash -l -c "RAILS_ENV=production
bundle exec puma $DAEMON_OPTS"
      /usr/local/bin/sudo -u gitlab_ci -H /usr/local/bin/bash -l -c "mkdir -p $PID_PATH &&
 $START_SIDEKIQ  > /dev/null  2>&1 &"
      echo "$DESC started"
    fi
  fi
}

stop() {
  cd $APP_ROOT
  check_pid
  if [ "$PID" -ne 0 -a "$STATUS" -ne 0 ]; then
    ## Program is running, stop it.
    kill -QUIT `cat $WEB_SERVER_PID`
    sudo -u gitlab_ci -H bash -l -c "mkdir -p $PID_PATH && $STOP_SIDEKIQ  > /dev/null  2>&
1 &"
    rm "$WEB_SERVER_PID" >> /dev/null
    echo "$DESC stopped"
  else
    ## Program is not running, exit with error.
    echo "Error! $DESC not started!"
    exit 1
  fi
}

restart() {
  cd $APP_ROOT
  check_pid
  if [ "$PID" -ne 0 -a "$STATUS" -ne 0 ]; then
    echo "Restarting $DESC..."
    kill -USR2 `cat $WEB_SERVER_PID`
    sudo -u gitlab_ci -H bash -l -c "mkdir -p $PID_PATH && $STOP_SIDEKIQ  > /dev/null  2>&
1 &"
    if [ `whoami` = root ]; then
      sudo -u gitlab_ci -H bash -l -c "mkdir -p $PID_PATH && $START_SIDEKIQ  > /dev/null
2>&1 &"
    fi
    echo "$DESC restarted."
  else
    echo "Error, $NAME not running!"
    exit 1
  fi
}

status() {
  cd $APP_ROOT
  check_pid
  if [ "$PID" -ne 0 -a "$STATUS" -ne 0 ]; then
    echo "$DESC / Unicorn with PID $PID is running."
    echo "$DESC / Sidekiq with PID $SPID is running."
  else
    echo "$DESC is not running."
    exit 1
```

```
  fi
}

## Check to see if we are running as root first.
## Found at http://www.cyberciti.biz/tips/shell-root-user-check-script.html
if [ "$(id -u)" != "0" ]; then
    echo "This script must be run as root"
    exit 1
fi

case "$1" in
  start)
        start
        ;;
  stop)
        stop
        ;;
  restart)
        restart
        ;;
  reload|force-reload)
        echo -n "Reloading $NAME configuration: "
        kill -HUP `cat $PID`
        echo "done."
        ;;
  status)
        status
        ;;
  *)
        echo "Usage: sudo service gitlab {start|stop|restart|reload}" >&2
        exit 1
        ;;
esac

exit 0
```

- Make the script executable:

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

- Start and enable GitLab CI at boot:

```
echo 'gitlab_ci_enable="YES"' >> /etc/rc.conf
service gitlab-ci start
```

Nginx

- Install nginx:

```
portmaster www/nginx
```

- Start and enable nginx at boot:

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

- Edit the nginx configuration file:

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

- And add the following inside the http block:

```
upstream gitlab_ci {
  server unix:/home/gitlab_ci/gitlab-ci/tmp/sockets/gitlab-ci.socket;
}

server {
  listen 80 default_server;
  server_name ci.example.com;
  root /home/gitlab_ci/gitlab-ci/public;

  access_log  /var/log/nginx/gitlab_ci_access.log;
  error_log   /var/log/nginx/gitlab_ci_error.log;

  location / {
    try_files $uri $uri/index.html $uri.html @gitlab_ci;
  }

  location @gitlab_ci {
    proxy_read_timeout 300;
    proxy_connect_timeout 300;
    proxy_redirect     off;

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

    proxy_pass http://gitlab_ci;
  }
}
```

Login

Visit http://ci.example.com for your first GitLab CI login. The setup has created an admin account for you. You can use it to log in:

- admin@local.host
- 5iveL!fe

Resources

- https://github.com/Phlzen/gitlab-ci-recipes/blob/master/freebsd/install.md
- https://github.com/gitlabhq/gitlab-ci/blob/2-2-stable/doc/installation.md
- https://raw.githubusercontent.com/gitlabhq/gitlab-ci/2-2-stable/lib/support/nginx/gitlab_ci
- https://github.com/Phlzen/gitlab-ci-recipes/blob/master/freebsd/rc.d-2.2.sh

History

#1 - 04/11/2015 08:57 PM - Daniel Curtis

- Description updated

#2 - 04/11/2015 09:22 PM - Daniel Curtis

- Description updated

#3 - 04/11/2015 10:20 PM - Daniel Curtis

- Description updated

- Status changed from New to In Progress

- % Done changed from 0 to 50

#4 - 04/13/2015 09:59 AM - Daniel Curtis

- Description updated

#5 - 04/13/2015 02:18 PM - Daniel Curtis

- Description updated

#6 - 04/13/2015 02:32 PM - Daniel Curtis

- Description updated

#7 - 04/13/2015 04:25 PM - Daniel Curtis

- Description updated

- % Done changed from 50 to 80

#8 - 10/03/2015 02:05 PM - Daniel Curtis

- Status changed from In Progress to Rejected

#9 - 10/03/2015 02:06 PM - Daniel Curtis

- Description updated