International Journal of Marketing and Technology
Vol. 8 Issue 3, March 2018,
ISSN: 2249-1058 Impact Factor: 6.559
Journal Homepage: http://www.ijmra.us, Email: editorijmie@gmail.com
Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gage as well as in Cabell's Directories of Publishing Opportunities, U.S.A

INSTALL AND CONFIGURE GITLAB SERVER IN MAC OS

Sukhendu Mukherjee*

Abstract

This article describes how we can Install and configure GitLab in macOS.GitLab is an online Git repository manager with a wiki, issue tracking, CI and CD. It is a great way to manage git repositories on a centralized server. GitLab gives you complete control over your repositories or projects and allows you to decide whether they are public or private for free.If we use GitLab.com for repository then source code will be hosted in GitLab server and code is not 100% secure.The best approach to configure in company server to make it secure.In this article you will see how we can install GitLab server in macOS and confire it with private domain.

Keywords:
GitLab;
macOS;
install;
server;
GitLab CI.

***CTO Tiny Planet Inc**

1. Introduction

GitLab is an online Git repository manager with a wiki, issue tracking, CI and CD. It is a great way to manage git repositories on a centralized server

It gives us following advantages.

GitLab is an online Git repository manager with a wiki, issue tracking, CI and CD. It is a great way to manage git repositories on a centralized server. GitLab gives you complete control over your repositories or projects and allows you to decide whether they are public or private for free.

1. GitLab.com

GitLab.com hosts your (private) software projects for free.

It offers free public and private repositories, issue-tracking and wikis.

It runs GitLab Enterprise Edition and GitLab CI.

No installation required, you can just sign up for a free account. Support Package:

Free subscribers can use the GitLab.com Support Tracker if they have questions.

GitLab.com Bronze Support will let you email support directly for timely, personal and private answers. This costs $9.99 per user per year for next-business-day response time and is available in packs of 20 users.

2. GitLab Community Edition (CE)

Free, self hosted application where you can get support from the Community

Feature rich: Git repository management, code reviews, issue tracking, activity feeds and wikis. It comes with GitLab CI for continuous integration and delivery.

Open Source: MIT licensed, community driven, 700+ contributors, inspect and modify the source, easy to integrate into your infrastructure.

Scalable: support 25,000 users on one server or a highly available active/active cluster.

Merge requests with line-by-line comments, CI and issue tracker integrations.

If we install GitLab in company/organization server then we can get all above benefit with secured domain.

2. Research Method

We took company installed mac server having macOSinstalled.Then we installed following component and execute following steps to achive the result.Finally able to land GitLab homepage successfully.

1. Packages and dependency need to install before installing git lab in MAC.

Following Command line tools need to install first.

xcode-select --install #xcode command line tools

Homebrew (missing package manager for Mac OS X)

ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

brew install icu4c gitlogrotate libxml2 cmakepkg-configopenssl brew link openssl --force

Make sure you have python 2.5+ (gitlab don't support python 3.x)

Confirm python 2.5 (or greater)

python --version

GitLab looks for python2

sudo ln -s /usr/bin/python /usr/bin/python2

Some more dependencies

sudoeasy_install pip sudo pip install pygments

Install docutils from source.

brew install docutils

2. Create system users for GIT Lab.

Need to run the following commands in order to create the group and user git.

LastUserID=$(dscl . -list /Users UniqueID | awk '{print $2}' | sort -n | tail -1)

NextUserID=$((LastUserID + 1))

sudodscl . create /Users/git

sudodscl . create /Users/gitRealName "GitLab"

sudodscl . create /Users/git hint "Password Hint"

sudodscl . create /Users/gitUniqueID $NextUserID

LastGroupID=$(dscl .readall /Groups | grep PrimaryGroupID | awk '{ print $2 }' | sort -n | tail -1)

NextGroupID=$(($LastGroupID + 1))

sudodscl . create /Groups/git

sudodscl . create /Groups/gitRealName "GitLab"

sudodscl . create /Groups/gitpasswd "*"

sudodscl . create /Groups/gitgid $NextGroupID

sudodscl . create /Users/gitPrimaryGroupID $NextGroupID

sudodscl . create /Users/gitUserShell $(which bash)

sudodscl . create /Users/gitNFSHomeDirectory /Users/git

sudocp -R /System/Library/User\ Template/English.lproj /Users/git

sudochown -R git:git /Users/git

Hide the git user from the login screen:

sudo defaults write /Library/Preferences/com.apple.loginwindowHiddenUsersList -array-add git

Unhide:

sudo defaults delete /Library/Preferences/com.apple.loginwindowHiddenUsersList

3. Need to install Ruby as On OS X we are forced to use non-system ruby and install it using version manager.

Install rbenv and ruby-build

brew install rbenv ruby-build

Make sure rbenv loads in the git user's shell

echo 'export PATH="/usr/local/bin:$PATH"' | sudo -u git tee -a /Users/git/.profile

echo 'if which rbenv> /dev/null; then eval "$(rbenvinit -)"; fi' | sudo -u git tee -a /Users/git/.profile

sudo -u gitcp /Users/git/.profile /Users/git/.bashrc

If you get the following error on OS X 10.8.5 or lower: ./bin/install:3: undefined methodrequire_relative' for main:Object (NoMethodError)` Do the following to update to the proper Ruby version

echo 'export PATH="/usr/local/bin:$PATH"' >> ~/.bash_profile

echo 'eval "$(rbenvinit - --no-rehash)"' >> ~/.bash_profile

. ~/.bash_profile

Install ruby for the git user

sudo -u git -H -i 'rbenv install 2.3.3'

sudo -u git -H -i 'rbenv global 2.3.3'

Install ruby for your user too (optional)

rbenv install 2.3.3

rbenv global 2.3.3

4. Go - Since GitLab 8.0, Git HTTP requests are handled by gitlab-git-http-server. This is a small daemon written in Go. To install gitlab-git-http-server we need a Go compiler.

brew install go

5. Database Configuration for GIT lab.

Gitlab recommends using a PostgreSQL database. But you can use MySQL too, see MySQL setup guide.

brew install postgresql

ln -sfv /usr/local/opt/postgresql/*.plist ~/Library/LaunchAgents

launchctl load ~/Library/LaunchAgents/homebrew.mxcl.postgresql.plist

Login to PostgreSQL

psql -d postgres

Create a user for GitLab.

CREATE USER git;

Create the GitLab production database & grant all privileges on database

CREATE DATABASE gitlabhq_production OWNER git;

Quit the database session

\q

Try connecting to the new database with the new user

sudo -u git -H psql -d gitlabhq_production

6. Redis is an open-source in-memory database project implementing a distributed, in-memory key-value store with optional durability.Need to install for GitLab.

brew install redis

ln -sfv /usr/local/opt/redis/*.plist ~/Library/LaunchAgents

Redisconfig is located in /usr/local/etc/redis.conf. Make a copy:

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

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

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

Edit file (vi /usr/local/etc/redis.conf) and uncomment/edit:

unixsocket /tmp/redis.sock

unixsocketperm 777

Start Redis

launchctl load ~/Library/LaunchAgents/homebrew.mxcl.redis.plist

7. Configure GitLab

cd /Users/git

Clone the Source

Clone GitLab repository

sudo -u git -H git clone https://gitlab.com/gitlab-org/gitlab-ce.git -b 9-2-stable gitlab

Note: You can change 9-2-stable to master if you want the bleeding edge version, but never install master on a production server!

Configure It

Go to GitLab installation folder (NOTE: you will need to remain in /Users/git/gitlb for all further steps!)

cd /Users/git/gitlab

Copy the example GitLabconfig

sudo -u git -H cpconfig/gitlab.yml.exampleconfig/gitlab.yml

sudo -u gitsed -i "" "s/\/usr\/bin\/git/\/usr\/local\/bin\/git/g" config/gitlab.yml

sudo -u gitsed -i "" "s/\/home/\/Users/g" config/gitlab.yml

Update GitLabconfig file, follow the directions at top of file

sudo -u git -H vi config/gitlab.yml

Copy the example secrets file

sudo -u git -H cpconfig/secrets.yml.exampleconfig/secrets.yml

sudo -u git -H chmod 0600 config/secrets.yml

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

sudochown -R git log/

sudochown -R gittmp/

sudochmod -R u+rwX,go-w log/

sudochmod -R u+rwXtmp/

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

sudochmod -R u+rwXtmp/pids/

sudochmod -R u+rwXtmp/sockets/

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

sudomkdir public/uploads

sudochmod 0700 public/uploads

Make sure GitLab can write to the repositories directory

sudomkdir /Users/git/repositories

sudochmod -R ug+rwX,o-rwx /Users/git/repositories/

sudochmod -R ug-s /Users/git/repositories/

sudo find /Users/git/repositories/ -type d -print0 | sudoxargs -0 chmodg+s

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

sudochmod -R u+rwX builds/

Copy the example Unicorn config

sudo -u git -H cpconfig/unicorn.rb.exampleconfig/unicorn.rb

sudo -u gitsed -i "" "s/\/home/\/Users/g" config/unicorn.rb

Find number of cores

sysctl -n hw.ncpu

Enable cluster mode if you expect to have a high load instance Ex. change amount of workers to 3 for 2GB RAM server Set the number of workers to at least the number of cores

sudo -u git -H vi config/unicorn.rb

Copy the example Rack attack config

sudo -u git -H cpconfig/initializers/rack_attack.rb.exampleconfig/initializers/rack_attack.rb

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

sudo -u git -H gitconfig --global core.autocrlf input

Disable gitgc --auto because GitLab runs gitgc for us already.

sudo -u git -H gitconfig --global gc.auto 0

Configure Git to generate packfile bitmaps (introduced in Git 2.0) on the GitLab server during gitgc.

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

Configure Redis connection settings

sudo -u git -H cpconfig/resque.yml.exampleconfig/resque.yml

Change the Redis socket path to /tmp/redis.sock:

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

Redis (single instance)

url: unix:/tmp/redis.sock

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

Note: If you want to use HTTPS, see Using HTTPS for the additional steps.

Configure GitLab DB Settings

PostgreSQL only:

sudo -u gitcpconfig/database.yml.postgresqlconfig/database.yml

MySQL only:

sudo -u gitcpconfig/database.yml.mysqlconfig/database.yml

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

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

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

sudo -u git -H chmod o-rwxconfig/database.yml

Install Gems

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

Preparation:

sudosugit

. ~/.profile

gem install bundler --no-ri --no-rdoc

rbenv rehash

cd ~/gitlab/

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

bundle install --deployment --without development test mysqlawskerberos

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

bundle install --deployment --without development test postgresawskerberos

Note: If you want to use Kerberos for user authentication, then omit kerberos in the --without option above.

Install GitLab Shell

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

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

sudosugit

. ~/.profile

cd ~/gitlab/

bundle exec rake gitlab:shell:install REDIS_URL=unix:/tmp/redis.sock RAILS_ENV=production

By default, the gitlab-shell config is generated from your main GitLabconfig. You can review (and modify) the gitlab-shell config as follows:

sudo -u git -Hvi /Users/git/gitlab-shell/config.yml

Note: If you want to use HTTPS, see Using HTTPS for the additional steps.

Note: Make sure your hostname can be resolved on the machine itself by either a proper DNS record or an additional line in /etc/hosts ("127.0.0.1 hostname"). This might be necessary for example if you set up gitlab behind a reverse proxy. If the hostname cannot be resolved, the final installation check will fail with "Check GitLab API access: FAILED. code: 401" and pushing commits will be rejected with "[remote rejected] master -> master (hook declined)".

Install gitlab-workhorse

sudosu - git

cd /Users/git/gitlab

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

Initialize Database and Activate Advanced Features

sudosugit

. ~/.profile

cd ~/gitlab/

bundle exec rake gitlab:setup RAILS_ENV=production

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

Note: You can set the Administrator/root password by supplying it in environmental variable GITLAB_ROOT_PASSWORD as seen below. If you don't set the password (and it is set to the default one) please wait with exposing GitLab to the public internet until the installation is done and you've logged into the server the first time. During the first login you'll be forced to change the default password.

bundle exec rake gitlab:setup RAILS_ENV=production GITLAB_ROOT_PASSWORD=yourpassword

Secure secrets.yml

The secrets.yml file stores encryption keys for sessions and secure variables. Backup secrets.yml someplace safe, but don't store it in the same place as your database backups. Otherwise your secrets are exposed if one of your backups is compromised.

Install Init Script

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

cd /Users/git/gitlab

sudomkdir -p /etc/init.d/

sudomkdir -p /etc/default/

sudocp lib/support/init.d/gitlab /etc/init.d/gitlab

Since you are installing to a folder other than default /home/users/git/gitlab, copy and edit the defaults file:

curl -O https://raw.githubusercontent.com/WebEntity/Installation-guide-for-GitLab-on-OS-X/master/gitlab.default.osx

sudocpgitlab.default.osx /etc/default/gitlab.default

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

Setup Logrotate

sudocp lib/support/logrotate/gitlab /usr/local/etc/logrotate.d/gitlab

sudosed -i "" "s/\/home/\/Users/g" /usr/local/etc/logrotate.d/gitlab

ln -sfv /usr/local/opt/logrotate/*.plist ~/Library/LaunchAgents

launchctl load ~/Library/LaunchAgents/homebrew.mxcl.logrotate.plist

Check Application Status

Check if GitLab and its environment are configured correctly:

sudosugit

. ~/.profile

cd ~/gitlab/

bundle exec rake gitlab:env:info RAILS_ENV=production

Compile Assets

sudosugit

. ~/.profile

cd ~/gitlab/

bundle exec rake assets:precompile RAILS_ENV=production

Start Your GitLab Instance

sudosh /etc/init.d/gitlab start

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

Nginx Installation ForGit Lab

brew install nginx

sudomkdir -p /var/log/nginx/

Site Configuration

Default nginx configuration has an example server on port 8080, same as Gitlab Unicorn instance, which will collide and Gitlab won't start. Edit nginx configuration and comment out whole example server block for it to work together:

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

Copy the example site config:

sudocp lib/support/nginx/gitlab /usr/local/etc/nginx/servers/gitlab

sudosed -i "" "s/\/home/\/Users/g" /usr/local/etc/nginx/servers/gitlab

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

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

sudo vi /usr/local/etc/nginx/servers/gitlab

Note: If you want to use HTTPS, replace the gitlab Nginx config with gitlab-ssl. See Using HTTPS for HTTPS configuration details.

Test Configuration

Validate your gitlab or gitlab-ssl Nginx config file with the following command:

sudonginx -t

You should receive syntax is okay and test is successful messages. If you receive errors check your gitlab or gitlab-ssl Nginx config file for typos, etc. as indicated in the error message given.

Start

sudonginx

Done!

Double-check Application Status

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

sudosugit

. ~/.profile

cd ~/gitlab/

bundle exec rake gitlab:check RAILS_ENV=production

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

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

Initial Login

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

root 5iveL!fe

Important Note: On login you'll be prompted to change the password.

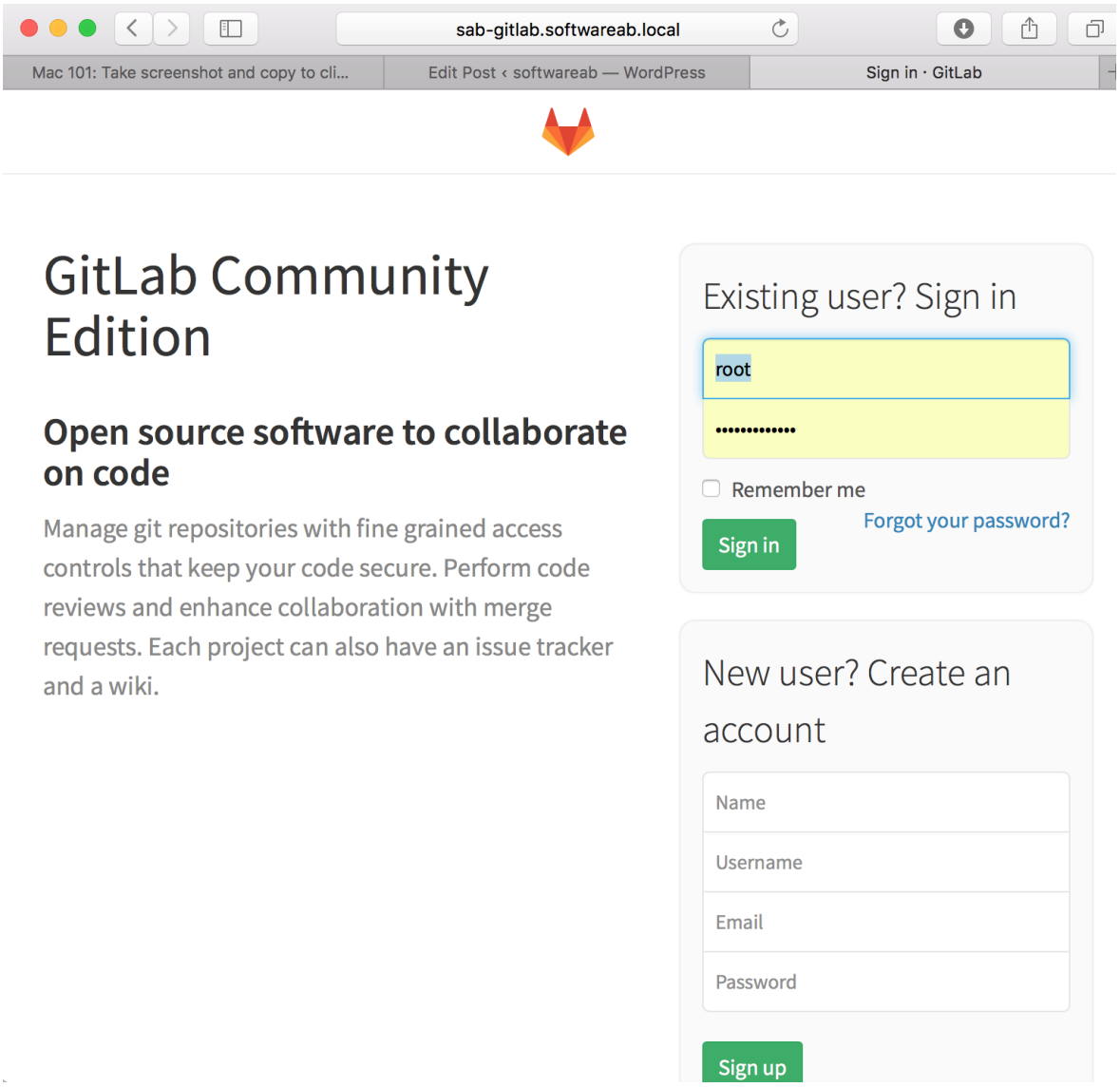

Figure 1.*GitLab Home page in macOS*

3. Results and Analysis (10pt)

As a result we are able to install following software in macOS

- Ruby
- Go
- Node
- System User
- Database
- Redis

- GitLab
- Nginx

Able to land GitLabloginpage from where we can create/manage group/repositories and configure CI(Continuous Integration)

4. Conclusion

Following the all steps mentioned above we can land GitLabloginpage from where we can create/manage group/repositories and configure CI(Continuous Integration).