Bitnami GitLab

IMPORTANT: It is necessary to enable SSH server to be able to pull your code into the application. The SSH server is disabled by default in the Bitnami GitLab Virtual Machine. Check this guide to know how to enable it.

gitlab.png

GitLab allows you to keep your code secure on your own server, manage repositories, users and access permissions, communicate through issues, line-commens, wiki pages and perform code reviw with merge requests. It is powered by Ruby on Rails and completely free and open source (MIT license).

Please, take a look to the Quick Start Guide to know the basic use of this Stack.

How to log in the GitLab application?

The default credentials for the GitLab application are:

```
user mail: user@example.com
password: bitnami1 (bitnami, for versions of Bitnami GitLab > 6.4.x)
```

It is advisable to use a **Hostname** instead of and IP address. In the Virtual Machines and AMIs exist a tool for changing it automatically. Do not use "git." in your hostname, this cause problems if you want to access through http. You only have to run the following command to change it:

```
$ sudo /opt/bitnami/apps/gitlab/bnconfig --machine_hostname example.com
```

If you have configured GitLab to use an static domain name, you **should remove or rename** the "/opt/bitnami/apps/gitlab/bnconfig" to disable it. You can also add the domain in the /etc/hosts file so gitlab-shell will use 127.0.0.1 to push the changes in the repository:

```
127.0.0.1    example.com
```

If you want, you can configure it manually. These are the configuration options that you should modify in the "/opt/bitnami/apps/gitlab/htdocs/config/gitlab.yml" file:

```
host: YOUR_DOMAIN
```

In previous versions (GitLab 4.x) it is also necessary to change the following parameter:

```
ssh_host: YOUR_DOMAIN
```

Then restart Apache server:

```
$ installdir/ctlscript.sh restart apache
```

How to push your changes to the GitLab application?

Once you have uploaded your private key to Gitlab, you can upload your repository to the application. These are the basic steps:

```
git config --global user.name "Your full name"
git config --global user.email "user@example.com"

git checkout master
git remote add origin git@YOUR_HOSTNAME:test/test.git
git push -u origin master
```

Virtual Machines have SSH server disabled by default. That it is necessary to sync your changes with GitLab. You can see how to enable it at /Virtual_Appliances_Quick_Start_Guide#How_to_enable_sshd.3f

How to start/stop the servers?

You can use the "ctlscript.sh" utility from the command line. This script is in the installation directory.

```
$ cd installdir
$ ./ctlscript.sh start
```

This command start alls the required servers: Apache, MySQL, Redis and the GitLab Sidekiq server.

How to change the default URL to the root?

How to change the default URL?

This steps are for Gitlab versions newer than 6.0.0

This approach is based on the Bitnami Configuration Tool (bnconfig).

Bitnami Cloud Hosting

The best way to change your URL in BCH is to go to your application tab and modify it there. In the Bitnami Cloud Hosting console, select Servers, choose your server, Manage and go to the Applications tab. Press there the pencil next to the application which URL you want to modify and choose Use Custom Domain.

Refer to this guide for more information.

Cloud Images and Virtual Machines

Moving the application to /

If your application is running in "/gitlab" you can remove the prefix from the URL executing the following command:

```
$ sudo /opt/bitnami/apps/gitlab/bnconfig --appurl /
```

(use --help to check if that option is available for your application)

Now you will be able to access to the application at http://YOUR_DOMAIN instead of http://YOUR_DOMAIN/gitlab.

Updating the IP or hostname

Some applications require to update the IP/domain if the machine IP/domain changes. The bnconfig tool also has an option which updates the IP automatically during boot, called machine_hostname (use --help to check if that option is available for your application). Note that this tool changes the URL to http://NEW_DOMAIN/gitlab.

```
sudo /opt/bitnami/apps/gitlab/bnconfig --machine_hostname NEW_DOMAIN
```

If you already moved your application to the root URL you should include both options at the same time.

```
sudo /opt/bitnami/apps/gitlab/bnconfig --appurl / --machine_hostname NEW
```

If you have configured your machine to use an static domain name or IP, you **should rename or remove** the "/opt/bitnami/apps/gitlab/bnconfig" file.

```
sudo mv /opt/bitnami/apps/gitlab/bnconfig /opt/bitnami/apps/gitlab/bncon
```

Native Installer

Remember to use your actual installation directory instead of installdir.

Moving the application to /

If your application is running in "/gitlab" you can remove the prefix from the URL executing the following command:

On Linux,

```
installdir/apps/gitlab/bnconfig --appurl /
```

On Mac OS X,

```
installdir/apps/gitlab/bnconfig.app/Contents/MacOS/installbuilder.sh --a
```

On Windows,

```
installdir/apps/gitlab/bnconfig.exe --appurl /
```

(use --help to check if that option is available for your application)

Now you will be able to access to the application at http://YOUR_DOMAIN instead of http://YOUR_DOMAIN/gitlab.

Updating the IP or hostname

Some applications require to update the IP/domain if the machine IP/ domain changes. The bnconfig tool also has an option which updates the IP , called machine_hostname (use --help to check if that option is available for your application). Note that this tool changes the URL to http://NEW_DOMAIN/gitlab.

```
installdir/apps/gitlab/bnconfig --machine_hostname NEW_DOMAIN
```

If you already moved your application to the root URL you should include both options at the same time.

```
installdir/apps/gitlab/bnconfig --appurl / --machine_hostname NEW_DOMAIN
```

Tabs end

How to create a full backup of GitLab?

Backup

Bitnami stacks are self-contained and the simplest option for performing a backup is to copy or compress the Bitnami stack installation directory. To do so in a safe manner, you will need to stop all servers, so this method may not be appropriate if you have people accessing the application continuously.

Cloud Server

Follow these steps:

- Change to the directory in which you wish to save your backup.

  ```
  cd /your/directory
  ```

- Stop all servers.

  ```
  $ sudo /opt/bitnami/ctlscript.sh stop
  ```

- Create a compressed file with the stack contents.

  ```
  $ sudo tar -pczvf application-backup.tar.gz /opt/bitnami
  ```

- Restart all servers.

  ```
  $ sudo /opt/bitnami/ctlscript.sh start
  ```

You should now download or transfer the *application-backup.tar.gz* file to a safe location.

Virtual Machine

Follow these steps:

- Change to the directory in which you wish to save your backup.

  ```
  cd /your/directory
  ```

- Stop all servers.

  ```
  $ sudo /opt/bitnami/ctlscript.sh stop
  ```

- Create a compressed file with the stack contents.

  ```
  $ sudo tar -pczvf application-backup.tar.gz /opt/bitnami
  ```

- Restart all servers.

  ```
  $ sudo /opt/bitnami/ctlscript.sh start
  ```

You should now download or transfer the *application-backup.tar.gz* file to a safe location.

Native Installer (Windows)

Follow these steps:

- Stop all servers using the shortcuts in the Start Menu or the graphical manager tool.

- Create a compressed file with the stack contents. You can use a graphical tool like 7-Zip or WinZip.
- Stop all servers using the shortcuts in the Start Menu or the graphical manager tool.

You should now download or transfer the *application-backup.zip* file to a safe location.

Native Installer (Linux and Mac OS X)

Follow these steps:

- Change to the directory in which you wish to save your backup.

  ```
  cd /your/directory
  ```

- Stop all servers.

  ```
  $ sudo installdir/ctlscript.sh stop
  ```

- Create a compressed file with the stack contents.

  ```
  $ sudo tar -pczvf application-backup.tar.gz installdir
  ```

- Restart all servers.

  ```
  $ sudo installdir/ctlscript.sh start
  ```

You should now download or transfer the *application-backup.tar.gz* file to a safe location.

Tabs end

Restore

Bitnami stacks are self-contained, so to restore a stack, you only need to uncompress the backup file in the same location. It is important to use the same path that was used when the stack was originally installed.

Cloud Server

Follow these steps:

- Change to the directory containing your backup.

  ```
  cd /your/directory
  ```

- Stop all servers.

  ```
  $ sudo /opt/bitnami/ctlscript.sh stop
  ```

- Rename the current directory to save it.

  ```
  $ sudo mv /opt/bitnami /opt/bitnamiBackup
  ```

- Uncompress the backup file to the original directory.

  ```
  $ sudo tar -pxzvf application-backup.tar.gz -C /
  ```

- Start all servers.

  ```
  $ sudo /opt/bitnami/ctlscript.sh start
  ```

Virtual Machine

Follow these steps:

- Change to the directory containing your backup.

  ```
  cd /your/directory
  ```

- Stop all servers.

  ```
  $ sudo /opt/bitnami/ctlscript.sh stop
  ```

- Rename the current directory to save it.

  ```
  $ sudo mv /opt/bitnami /opt/bitnamiBackup
  ```

- Uncompress the backup file to the original directory.

  ```
  $ sudo tar -pxzvf application-backup.tar.gz -C /
  ```

- Start all servers.

  ```
  $ sudo /opt/bitnami/ctlscript.sh start
  ```

Native Installer (Windows)

Follow these steps:

- Uncompress the backup file to the original directory.
- Install services by launching a new command prompt and executing the following commands. Administrator privileges are required.

  ```
  $ cd installdir
  $ serviceinstall.bat INSTALL
  ```

You can now start or stop servers using the graphical manager tool.

Native Installer (Linux and Mac OS X)

Follow these steps:

- Change to the directory containing your backup.

  ```
  cd /your/directory
  ```

- Stop all servers.

  ```
  $ sudo /opt/bitnami/ctlscript.sh stop
  ```

- Rename the current directory to save it.

  ```
  $ sudo mv installdir installdirBackup
  ```

- Uncompress the backup file to the original directory.

  ```
  $ sudo tar -pxzvf application-backup.tar.gz -C /
  ```

- Start all servers.

  ```
  $ sudo installdir/ctlscript.sh start
  ```

Tabs end

IMPORTANT: When restoring, remember to maintain the original permissions for the files and folders. For example, if you originally installed the stack as 'root', make sure that the restored files are owned by 'root'.

If you want to create only a database backup, refer to these instructions for MySQL and PostgreSQL.

You should also create a backup of the /home/git folder where are stored the repositories.

How to run rake commands?

To run rake commands you have to use the rake script located inside *installdir*/apps/gitlab/htdocs/bin.

First load the Bitnami environment

```
sudo installdir/use_gitlab
```

Then go to htdocs folder:

```
cd installdir/apps/gitlab/htdocs
```

and run the command like this:

```
ruby bin/rake rake_command
```

How to configure the email settings of GitLab?

You can configure the SMTP settings during the installation process. If you are using the Virtual Machine or AMI, you can configure it manually. For example, these are the options to configure it using a GMail account:

/opt/bitnami/apps/gitlab/htdocs/config/environments/production.rb

```
config.action_mailer.raise_delivery_errors = true
config.action_mailer.delivery_method = :smtp
config.action_mailer.perform_deliveries = true
config.action_mailer.smtp_settings = {
:address => "smtp.gmail.com",
:port => 587,
:domain => "gmail.com",
:authentication => :plain,
:user_name => "your_account@gmail.com",
```

```
:password => "your_password",
:enable_starttls_auto => true
}
```

If you want to use your office365 account to send emails, then you have to configure it like this:

/opt/bitnami/apps/gitlab/htdocs/config/environments/production.rb

```
config.action_mailer.raise_delivery_errors = true
config.action_mailer.delivery_method = :smtp
config.action_mailer.perform_deliveries = true
config.action_mailer.smtp_settings = {
:address => "smtp.office365.com",
:port => 587,
:domain => "office365.com",
:authentication => :login,
:user_name => "your_account@hotmail.com",
:password => "your_password",
:enable_starttls_auto => true
}
```

Take into account that user_name should be an email linked to your office365 account. Typycally is a hotmail/outlook mail account.

Then restart Sidekiq and Apache servers:

```
$ sudo /opt/bitnami/ctlscript.sh restart gitlab_sidekiq
$ sudo /opt/bitnami/ctlscript.sh restart apache
```

How to upgrade Gitlab?

General procedure to upgrade the stack

Remember that all the paths in this guide are for Virtual Machines and Cloud Images. If you are using Gitlab stack, you have to change the path /opt/bitnami for your installation path.

It is strongly recommended to create a backup before starting the update process. If you have important data, create and try to restore a backup to ensure that everything works properly.

There are two different ways to upgrade your application.

- You can upgrade the application and all stack components, such as PHP, Ruby, MySQL and Apache.
    - Follow these instructions.
- You can upgrade the application only without modifying any other stack components.
    - Use the links provided in the application page on the wiki.

Keeping in sync with the GitLab repository at GitHub

It is important to create a backup of the full server before upgrading GitLab. If you are using Bitnami Cloud Hosting you are able to create a backup easily.

It is necessary to add some packages because they required by GitLab for new versions:

```
sudo apt-get install zlib1g-dev libkrb5-dev cmake
```

Then stop all servers and start only PostgreSQL (or MySQL for GitLab < 7.11.4) and Redis servers:

```
sudo /opt/bitnami/ctlscript.sh stop
sudo /opt/bitnami/ctlscript.sh start postgresql
sudo /opt/bitnami/ctlscript.sh start redis
```

Use the "git" user for running all the commands from now on:

```
sudo su git
```

Update GitLab shell:

```
cd /opt/bitnami/apps/gitlab/gitlabshell
git stash
git fetch https://github.com/gitlabhq/gitlab-shell
git pull
git checkout v2.6.3  [replace it with the latest version]
git stash apply
```

Create a backup of GitLab

```
cd /opt/bitnami/apps/gitlab/htdocs
bundle exec rake gitlab:backup:create RAILS_ENV=production

Dumping database ...
```

```
Dumping MySQL database bitnami_gitlab ... [DONE]
done
Dumping repositories ...
* user/test ... [DONE]
* user/test.wiki ...  [SKIPPED]
done
Dumping uploads ...
done
Creating backup archive: 1434969584_gitlab_backup.tar ... done
Uploading backup archive to remote storage  ... skipped
Deleting tmp directories ... done
done
done
done
Deleting old backups ... skipping
```

Check the latest version stable of GitLab at https://github.com/gitlabhq/gitlabhq/releases and run the upgrade process replacing 'v7.11.4' with the latest stable version:

```
ruby -Ilib -e 'require "gitlab/upgrader"' -e 'class Gitlab::Upgrader' -e

GitLab 7 upgrade tool
Your version is 7.10.5
Latest available version for GitLab 7 is 7.11.4
Newer GitLab version is available
Do you want to upgrade (yes/no)? yes
Stash changed files
 -> git stash
Saved working directory and index state WIP on (no branch): 489b413 Versi
HEAD is now at 489b413 Version 7.10.5
 -> OK
Get latest code
 -> git fetch
 -> OK
Switch to new version
 -> git checkout v7.11.4
Previous HEAD position was 489b413... Version 7.10.5
HEAD is now at b725318... Version 7.11.4
 -> OK
Install gems
 -> bundle
Fetching source index from https://rubygems.org/
Using rake 10.4.2
...
```

```
Your bundle is complete!
Gems in the groups development, test, sqlite, test and sqlite were not i
It was installed into ./vendor/bundle
Post-install message from httparty:
When you HTTParty, you must party hard!
 -> OK
Migrate DB
 -> bundle exec rake db:migrate
...
 -> OK
Recompile assets
 -> bundle exec rake assets:clean assets:precompile
I, [2015-06-22T11:42:34.973207 #3133]  INFO -- : Writing /opt/bitnami/ap
I, [2015-06-22T11:42:49.065350 #3133]  INFO -- : Writing /opt/bitnami/ap
 -> OK
Clear cache
 -> bundle exec rake cache:clear
 -> OK
Done
```

Then log out as git user and start the servers again:

```
sudo /opt/bitnami/ctlscript.sh start
```

You should see the new version in your GitLab admin panel

Screen Shot 2015-06-22 at 13.48.02.png

Migration process for GitLab with MySQL (GitLab < 7.11.4)

In the Gitlab case, these are the steps to **migrate the database** from an old version to a new one:

- First of all, backup the gitlab, gitlab_ci databases, the .ssh file in git and gitlab_ci directories and the uploads folder.

```
cd /opt/bitnami
sudo ./use_gitlab
/opt/bitnami/mysql/bin/mysqldump -u root -p bitnami_gitlab > bitnami_
/opt/bitnami/mysql/bin/mysqldump -u root -p bitnami_gitlabci > bitnam
cp -r /home/git/.ssh PATH_TO_BACKUP/git/.ssh
cp -r /home/gitlab_ci/.ssh PATH_TO_BACKUP/gitlab_ci/.ssh
cp -r /opt/bitnami/apps/gitlab/htdocs/public/uploads PATH_TO_BACKUP/u
```

Important: If you are using Gitlab stack in the same machine, you have now to stop the old version and remove the gitlab users.

```
/opt/gitlab_old_version/ctlscript.sh stop
deluser git
rm -r /home/git
deluser gitlab_ci
rm -r /home/gitlab_ci
```

- Start the new Gitlab version.
- Copy the repositories and .ssh backups to the new version server.
- Stop all servers and start only MySQL. Note that the installation directory could be different.

```
sudo /opt/bitnami/ctlscript.sh stop
sudo /opt/bitnami/ctlscript.sh start mysql
```

- Now remove the databases and add the old ones.

```
sudo /opt/bitnami/use_gitlab
mysql -u root -p
INSERT YOUR PASSWORD HERE
drop database bitnami_gitlab;
create database bitnami_gitlab;
grant all privileges on bitnami_gitlab.* to 'bitnami'@'localhost' ide
grant all privileges on bitnami_gitlab.* to 'gitlab'@'localhost' iden
flush privileges;
drop database bitnami_gitlabci;
create database bitnami_gitlabci;
grant all privileges on bitnami_gitlabci.* to 'bitnami'@'localhost' i
grant all privileges on bitnami_gitlab.* to 'gitlab'@'localhost' iden
flush privileges;
```

```
\q

mysql -u root -p bitnami_gitlab < PATH_TO_BACKUP/bitnami_gitlab_bk.s
mysql -u root -p bitnami_gitlabci < PATH_TO_BACKUP/bitnami_gitlabci_
```

- Update configuration files in the new version so that they correspond to the old version ones. At least you have to update database.yml files, so open them with a text editor and modify the password fields with the password provided to the database in the previous step. There will be two database.yml files.

```
/opt/bitnami/apps/gitlab/htdocs/config/database.yml
/opt/bitnami/apps/gitlabci/htdocs/config/database.yml
```

- Copy the .ssh backup files and the to the new server

```
cp -r PATH_TO_BACKUP/git/.ssh /home/git/.ssh
cp -r PATH_TO_BACKUP/gitlab_ci/.ssh /home/gitlab_ci/.ssh
cp -r PATH_TO_BACKUP/uploads /opt/bitnami/apps/gitlab/htdocs/public/
```

- Now run the following command:

```
cd /opt/bitnami/apps/gitlab/htdocs
bundle install --deployment --without development test sqlite postgr
```

- Log in as "git" user and update the database. It should be necessary to update any change in the database (you will maybe need to do more changes. Please go to the gitlab documentation to check it https://github.com/gitlabhq/gitlabhq...ter/doc/update):

```
sudo su git
ruby bin/rake db:migrate RAILS_ENV=production
exit
```

- Now fix the dashboard

```
cd /opt/gitlab/apps/gitlab/htdocs
ruby bin/rake migrate_iids RAILS_ENV=production
```

- You can check the the process running the following command

```
bundle exec rake gitlab:check RAILS_ENV=production
```

- Then restart services:

```
/opt/bitnami/ctlscript.sh restart
```

Troubleshooting

If you get a "Satellite doesn't exist" message during the upgrade, please run the following commands:

```
sudo su git
bundle exec rake gitlab:satellites:create RAILS_ENV=production
exit
```

Upgrade Gitlab-shell

To upgrade Gitlab-shell to the specific version of Gitlab you have to check the version of Gitlab-shell that your Gitlab needs. You will find a file called GITLAB_SHELL_VERSION in the repository of the specific version of Gitlab that contains the version of Gitlab-shell.
After that, you will need to update Gitlab-shell, please use the Bitnami console if you aren't using a cloud image.

- Go to your gitlab-shell folder:

```
installdir/apps/gitlab/gitlab-shell
```

- Change the user

```
su git
```

- Run the following commads, please change the version for the version that you need:

```
git stash
git fetch https://github.com/gitlabhq/gitlab-shell
git pull
git checkout v2.6.3
git stash apply
git stash drop
```

- Change the directory to installdir/apps/gitlab/htdocs and run the following commands:

```
cd installdir/apps/gitlab/htdocs
bundle install --deployment --without development test sqlite --binstubs
ruby bin/rake db:migrate RAILS_ENV=production
```

```
ruby bin/rake migrate_iids RAILS_ENV=production
bundle exec rake gitlab:satellites:create RAILS_ENV=production
```

- Now run the rewrite-hooks script

```
cd installdir/apps/gitlab/gitlab-shell/support
./rewrite-hooks.sh /opt/gitlab/apps/gitlab/repositories
```

- Finally, change the directory to installdir/apps/gitlab/htdocs and run the following commands:

```
cd installdir/apps/gitlab/htdocs
bundle exec bin/rake gitlab:check RAILS_ENV=production
```

You will have Gitlab-shell upgraded.

Keeping in sync with the GitLab repository at GitHub

Since the GitLab application is changing very fast, Bitnami GitLab stack is including the .git files necessaries to be synched with the repository.

This is an **advanced feature** that should be used only by someone that knows the application and what is happening at every step of the process described below.

A backup of the application directory should be done before moving forward.

As some configuration parameters are adjusted during the installation, there will be some differences between the installed version and the repository even if the installer was built recently.

To synch GitLab with its repository at GitHub, the steps below must be followed.

```
cd /opt/bitnami
./ctlscript.sh stop gitlab_sidekiq
cd apps/gitlab/htdocs
su git
```

Bitnami modifies both Gemfile and Gemfile.lock files file in order to allow an offline installation. You should check them out and run bundle install after the "git checkout":

```
git checkout Gemfile*
git fetch
git checkout <brach> (where <branch> is, for instance, master or 5-2-sta
```

Now log in as root user or use "sudo" to run the following command:

```
bundle install --deployment --without development test sqlite postgres -
```

Again, log in as "git" user and update the database. It should be necessary to update any change in the database:

```
su git
ruby bin/rake db:migrate RAILS_ENV=production
```

Then restart services:

```
/opt/bitnami/ctlscript.sh start gitlab_sidekiq
/opt/bitnami/ctlscript.sh restart apache
```

Now, the GitLab application shipped at the Bitnami stack, AMI or Virtual Machine is up to date.

Troubleshooting

Charlock_holmes gem requires a modification in the Ruby config.h file. If you find problems compiling it, add the following entry in the "/opt/bitnami/ruby/includes/ruby-1.9.1/x86_64-linux/ruby/config.h" file:

```
#define HAVE_EACCESS 1
```

How to edit and commit files from the GitLab application?

GitLab web application allows you edit a file and commit the changes into the repository. To enable this feature it is necessary to create the "satellites" repositories. If you check the production.log file you can see an error similar to this: "RuntimeError: Satellite doesn't exist"

To create them you can run the following command **after creating a project**:

```
$ sudo su gitlab
$ cd installdir/apps/gitlab/htdocs
$ bundle exec bin/rake gitlab:satellites:create RAILS_ENV=production
```

Now you can edit & commit changes into the repository from the GitLab application itself.

How to debug GitLab errors?

The GitLab log files are saved in /opt/bitnami/apps/gitlab/htdocs/logs folder. The main log file is production.log.

Cloud Server

Once Apache starts, it will create two log files at */opt/bitnami/apache2/logs/access_log* and */opt/bitnami/apache2/logs/error_log* respectively. On Amazon Linux and Red Hat Enterprise cloud images, the log files are created at */var/log/httpd/access_log* and */var/log/httpd/error_log* instead.

- The *access_log* file is used to track client requests. When a client requests a document from the server, Apache records several parameters associated with the request in this file, such as: the IP address of the client, the document requested, the HTTP status code, and the current time.
- The *error_log* file is used to record important events. This file includes error messages, startup messages, and any other significant events in the life cycle of the server. This is the first place to look when you run into a problem when using Apache. If no error is found, you will see a message similar to:

```
Syntax OK
/installdir/ctlscript.sh : httpd started
```

Virtual Machine

Once Apache starts, it will create two log files at */opt/bitnami/apache2/logs/access_log* and */opt/bitnami/apache2/logs/error_log* respectively.

- The *access_log* file is used to track client requests. When a client requests a document from the server, Apache records several parameters associated with the request in this file, such as: the IP address of the client, the document requested, the HTTP status code, and the current time.
- The *error_log* file is used to record important events. This file includes error messages, startup messages, and any other significant events in the life cycle of the server. This is the first place to look when you run into a problem when using Apache. If no error is found, you will see a message similar to:

```
Syntax OK
/installdir/ctlscript.sh : httpd started
```

Native Installer

Once Apache starts, it will create two log files at *installdir/apache2/logs/access_log* and *installdir/apache2/logs/error_log* respectively.

- The *access_log* file is used to track client requests. When a client requests a document from the server, Apache records several parameters associated with the request in this file, such as: the IP address of the client, the document requested, the HTTP status code, and the current time.
- The *error_log* file is used to record important events. This file includes error messages, startup messages, and any other significant events in the life cycle of the server. This is the first place to look when you run into a problem when using Apache. If no error is found, you will see a message similar to:

```
Syntax OK
/installdir/ctlscript.sh : httpd started
```

Cloud Server

The main MySQL log file is created at */opt/bitnami/mysql/data/mysqld.log*.

Virtual Machine

The main MySQL log file is created at */opt/bitnami/mysql/data/mysqld.log*.

Native Installer

The main MySQL log file is created at *installdir/mysql/data/mysqld.log*.

GitLab servers write the log files into the "/*installdir*/apps/gitlab/htdocs/logs" folder. Check these log files if Sidekiq server can not be started.

How to use GitLab CI integrated with GitLab (> 5.4 with GitLab CI 3.0.0)?

IMPORTANT: For this integration, the domain name of the machine is set in the configuration files of the repositories that are cloned internally. Because of that, if the domain of the machine change, GitLab and GitLab CI will continue working automatically in Bitnami AMI and Virtual Machines but the internal clone of the repositories will fail. Also, the necessary information for the integration of each project will be out of date too. To fix it, these steps must be repeated for all existing projects.

Bitnami GitLab stack also ships GitLab CI. GitLab CI can be selected during the installation of Bitnami GitLab stack and it is installed by default on Bitnami GitLab cloud images and virtual machines.

- GitLab will be accessible at "http://<*hostname_gitlab_server*>:<*apache_port*>**/gitlab**"
- GitLab CI will be accessible at "http://<*hostname_gitlab_server*>:<*apache_port*>**/gitlabci**"

GitLab and GitLab CI have been designed to work jointly, so, when a commit is done on a GitLab project, for instance, GitLab CI is able to detect it and run any defined task. An example of this configuration is described below.

1. Add a project on GitLab.

projectongitlab.png

2. Add the same project on GitLab CI. To do so, go to GitLab CI (*http://example.com/gitlabci*) and log in. Then press on *GitLab Projects* and click on the *Add* button of the proper project.

gitlabprojects-on-gitlabci.png

3. Add a runner to the new project. Go to the *Runners* menu inside the project created and click on *Add* the existing runner to the current project.

gitlabci-runner-to-project-2.png

4. Enable the service "*GitLab CI*" on the project at GitLab. On GitLab CI, go to the menu *Integration* of the current project and copy the token and the url. Now go to GitLab. On the same project, go to services. Add the copied information, tick to activate the service and press both *Save* and *Test settings*. A green circle will appear if everything is correct.

gitlabci-enabled-at-gitlab.png

5. The integration is completed! Also a first build will start automatically at GitLab CI.

gitlabci-working-with-gitlab.png

How to use GitLab CI integrated with GitLab (GitLab < 5.4 with GitLab CI 2.2.0)?

Bitnami GitLab stack also ships GitLab CI. GitLab CI can be selected during the installation of Bitnami GitLab stack and it is installed by default on Bitnami GitLab cloud images and virtual machines.

gitlabplusgitlabci.png

- GitLab will be accessible at "http://<*hostname_gitlab_server*>:<*apache_port*>**/gitlab**"
- GitLab CI will be accessible at "http://<*hostname_gitlab_server*>:<*apache_port*>**/gitlabci**"

GitLab and GitLab CI have been designed to work jointly, so, when a commit is done on a GitLab project, for instance, GitLab CI is able to detect it and run any defined task. An example of this configuration is described below.

1. Add a project on GitLab.

projectongitlab.png

2. Create a ssh key for gitlab_ci user in this machine without a password. This key will allow GitLab CI to have read access to any chosen repository:

```
sudo su gitlab_ci -c "ssh-keygen -t rsa"
```

3. Add this key in "Deploy Keys" on the GitLab project. On GitLab, go to Projects -> Bitnami sample project -> Settings -> Deploy Keys -> Add deploy key, paste the key generated before and save it with any name.

gitlabdeploykey2.png

4. On your client (if it is different from the server you are using to host the GitLab server), start the repository as it is described at GitLab. This step is not related to the integration with GitLab CI but the repository should be started to complete the integration. Add your personal key in My profile -> SSH Keys and run the following commands for a new repository:

```
mkdir bitnami-sample-project
cd bitnami-sample-project
git init
touch README
git add README
git commit -m 'first commit'
git remote add origin git@<hostname_gitlab_server>:bitnami-sample-projec
git push -u origin master
```

5. Clone the repository with the gitlab_ci user on the GitLab server.

```
sudo su gitlab_ci -c "mkdir /opt/bitnami/apps/gitlabci/repositories"
sudo su gitlab_ci -c "/opt/bitnami/git/bin/git config --global user.name
sudo su gitlab_ci -c "/opt/bitnami/git/bin/git config --global user.emai
```

```
sudo su gitlab_ci -c "cd /opt/bitnami/apps/gitlabci/repositories/ && /opt
Cloning into 'bitnami-sample-project'...
remote: Counting objects: 3, done.
Receiving objects: 100% (3/3), 201 bytes, done.
remote: Total 3 (delta 0), reused 0 (delta 0)
```

6. Create the project on GitLab CI by pressing 'Add project' on the GitLab CI application. Add the parameters described below and save it.

gitlabciproject.png

i. Name: *bitnami-sample-project*
ii. Token: *(blank)*
iii. Path: */opt/bitnami/apps/gitlabci/repositories/bitnami-sample-project*
iv. Follow branches: *master*
v. Scripts: *ls*

7. Once created, press on "Details". "*Project URL*" and "*Project Token*" are required for the next step.

importantvaluesgitlabci.png

8. Enable GitLab CI on the repository created on GitLab. To do so, go to GitLab -> Projects -> Bitnami sample project -> Settings -> Services -> GitLab CI. Select "Active" and fill "Project URL" and "Project Token" with the values at the previous step. Finally, press "Save".

placeimportantvaluesfromgitlabci.png

9. GitLab integration with GitLab CI is complete! It can be tested by commiting a new file (for instance) from your client (if it is different from the server you are using to host the GitLab server). Then, GitLab CI will show the following:

successfullyintegrated2.png

How to use your pem file to pull or push

You can use your server pem file instead of create a new key pair.

First you need to get your pem file public key. You can do so running the following command:

```
ssh-keygen -y -f /path/to/pem_file
```

Add this key to Gitlab using the application API.

Then edit (or create if not exists) ~/.ssh/config and add the following lines:

```
Host <your-gitlab-application-domain>
IdentityFile /path/to/pem_file
```

Troubleshooting

GitLab error:

```
http.rb:763:in `initialize': getaddrinfo: Temporary failure in name reso
```

Check the domain name returns the correct IP address. You can also add the domain name in the /etc/hosts file so gitlab-shell will resolve the domain to 127.0.0.1.

```
127.0.0.1     example.com
```