Install Tracks on Ubuntu 20.10 - protected

https://www.getontracks.org/

https://github.com/TracksApp/tracks/releases

It helps to keep track of many projects and the actions needed for each of them.

You can create contexts such as ''home'', ''tired'', ''work'', ''library'' or whatever so you know what actions you should do when you are on which context.

You can also have recurrent actions and if you think you can only do something tomorrow or next week, there's 2 handy buttons (+1 and +7) to have the action go away and come back later. There's also a mobile phone login page which should load faster.

Tracks will be served by Phusion Passenger through Apache in a subdir of its own.

MySQL will be used for the database.

it is not the best practice... but normally I do

> *>sudo -i*

at the beginning to avoid all the sudos during the installation

Install LAMP

LAMP = Linux, Apache, MySQL, PHP

We'll just install Apache and MySQL

Install Apache and MySQL

> *>apt install apache2*
> *>apt install mysql-server*
> *>apt install mysql-client*

> *>systemctl enable mysql*

> *>systemctl start mysql.service*

To enable or disable the automatic start of MySQL service when you boot your machine use

> *>systemctl enable mysql*
>
> *>systemctl disable mysql*

To start or stop manually the MySQL service use

> *>systemctl start mysql.service*
>
> *>systemctl stop mysql.service*

Install Ruby

Tracks requires Ruby 2.5 or greater. Most of the testing is done with 2.6.

> *>apt install unzip*
> *>apt install zlib1g-dev*
> *>apt install ruby*
> *>apt install rubygems*
> *# forces*
> *# >apt install ruby2.7*
> *# >apt install fonts-lato*
> *# >apt install libruby2.7*
> *# >apt install rake*
> *# >apt install ruby-minitest*
> *# >apt install ruby-net-telnet*
> *# >apt install ruby-power-assert*
> *# >apt install ruby-test-unit*
> *# >apt install ruby-xmlrpc*
> *# >apt install rubygems-integration*
> *# and suggests*
> *# >apt install ri*
> *# >apt install ruby-dev*
> *>apt install bundler*
> *# forces*
> *# >apt install libgmp-dev*
> *# >apt install libgmpxx4ldbl*
> *# >apt install ruby-bundler*
> *# >apt install ruby-connection-pool*
> *# >apt install ruby-dev*
> *# >apt install ruby-molinillo*
> *# >apt install ruby-net-http-persistent*
> *# >apt install ruby-thor*
> *# >apt install ruby2.7-dev*
> *# >apt install ruby2.7-doc*
> *# and suggests*

 # >apt install gmp-doc
 # >apt install libgmp10-doc
 # >apt install libmpfr-dev

>apt install thin
 # forces
 # >apt install ruby-daemons
 # >apt install ruby-eventmachine

>apt install build-essential
>apt install libapache2-mod-passenger
 # forces
 # >apt install passengerrake time:zones:local
 # >apt install ruby-rack
 # and suggests
 # >apt install nodejs
 # >apt install passenger-doc
 # >apt install rails

~~>apt install libopenssl-ruby~~
>apt install libssl-dev
>apt install libpq-dev
>apt install libxml2-dev
>apt install libxslt-dev
>apt install libsqlite3-dev

 # if you use MySQL database
~~>apt install libmysqld-dev~~
~~>apt install libdbd-mysql-ruby~~
>apt install libmysqlclient-dev

#if you use Maria database
>apt install libmariadbd-dev # for de server, d:demon
>apt install libmariadb-dev # for the client

Debian, Ubuntu with Thor

As explained here

https://bugs.launchpad.net/ubuntu/+source/ruby-thor/+bug/1885424

the following change on the files has broken their magic...

/usr/share/rubygems-integration/all/gems/bundler-2.1.4/lib/bundler/vendored_molinillo.rb

--- a/lib/bundler/vendored_molinillo.rb

+++ b/lib/bundler/vendored_molinillo.rb

@@ -1,4 +1,4 @@

 # frozen_string_literal: true

 module Bundler; end

-require_relative "vendor/molinillo/lib/molinillo"

+require "molinillo"

/usr/share/rubygems-integration/all/gems/bundler-2.1.4/lib/bundler/vendored_thor.rb

--- a/lib/bundler/vendored_thor.rb

+++ b/lib/bundler/vendored_thor.rb

@@ -2,7 +2,7 @@

```
 module Bundler
  def self.require_thor_actions
- require_relative "vendor/thor/lib/thor/actions"
+ require "thor/actions"
  end
 end
-require_relative "vendor/thor/lib/thor"
+require "thor"
```

---

you can revert the changes or the files from the master branch of github https://github.com/rubygems/rubygems

also you need to copy from the master branch the directories

vendor/thor and vendor/molinillo

into the directory /usr/share/rubygems-integration/all/gems/bundler-2.1.4/lib/bundler/vendor

Install Tracks

you can download a selected releases from https://github.com/TracksApp/tracks/releases/tag/v2.5.1

/var/www is where Apache2 expects to have the websites (in Ubuntu, Debian flavors)

> *>cd /var/www*
>
> *>git clone https://github.com/TracksApp/tracks.git*

this will create the directory **tracks** and download the files from github

Configure the parameter files

create the configuration file database.yml from the file database.yml.tmpl

create the configuration file site.yml from the file site.yml.tmpl

from inside the **tracks** directory

> *>cp ./config/database.yml.tmpl ./config/database.yml*
> *>cp ./config/site.yml.tmpl ./config/site.yml*

database.yml

you can use the nano text editor

> *>nano database.yml*

Normally you edit only the "production" section

the words in red are the names you need to choose for the database, the user and password. This must match what you later use when manually creates the database.

> *production:*
> *adapter: mysql2*
> *database: tempe*
> *encoding : utf8*
> *host: localhost*
> *username: daniel*
> *password: DANIEL*

if you use sqlite

> *production:*
> *adapter: sqlite3*
> *database: db/tracks-20-blank.sqlite3.db*
> *pool: 5*
> *timeout: 5000*

site.yml

you can use the nano text editor

> *>nano site.yml*

Choose a new secret token and set the correct time zone

> *secret_token: ''change-me''*
>
> *time_zone: ''UTC''*
>
> *admin_email : my.email@domain.com*

For the time zone setting you can use the command

> ~~*>rake --rakefile=/var/www/tracks/Rakefile time:zones:local*~~
>
> *>bundle exec rake time:zones:local*

to see all available timezones on your machine

If you intend to use Tracks behind a web server or reverse proxy with https enabled, ensure to set force_ssl option to true.

If you are using Windows, you may need to check the 'shebang' lines (#!/usr/bin/env ruby) of the /public/dispatch.* files and all the files in the /script directory.

They are set to #!/usr/bin/env ruby by default.

This should work for all Unix based setups (Linux or Mac OS X), but Windows users will probably have to change it to something like #c:/ruby/bin/ruby to point to the Ruby binary on your system.

Adjust owner and owner group for the downloaded files

set the owner and the owner group of all the files to www-data

> *>chown -R www-data:www-data **tracks***

then you need to set the access rights, the simplest way is at /var/www execute

> *>chmod -R 777 tracks*

later once you have all running you can correctly set

> *> find /var/www/tracks -type d -exec chmod 700 '{}' \;*
> *> find /var/www/tracks -type f -exec chmod 600 '{}' \;*
> *> find /var/www/tracks/script -type f -exec chmod 700 '{}' \;*

The ''find'' and ''chmod'' commands will set permissions as ''700'' for all directories and files inside the ''script'' directory.

All other files will have permissions set to ''600''.

All the files will be owned by the user ''www-data'' which is the user that runs Apache

MySQL - change the default of default_authentication_plugin

since Mysql version 8.0 the default values has been changed from mysql_native_password to caching_sha2_password

this needs to be reverted in the cnf file

for my installation /etc/mysql/my.cnf is a symbolic link to /etc/alternatives/my.cnf which is a symbolic link to /etc/mysql/mysql.cnf

so we need to edit the file /etc/mysql/mysql.cnf

add the following at the end of the file

> *[mysqld]*
>
> *default_authentication_plugin=mysql_native_password*

the group [mysqld] is for the server, for the client the group is [mysql]

(info http://dev.mysql.com/doc/mysql/en/server-system-variables.html

https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html)

possible places of the cnf file

- /etc/my.cnf
- /etc/mysql/my.cnf
- $MYSQL_HOME/my.cnf
- [datadir]/my.cnf
- ~/.my.cnf

Manually create the database

Tracks is tested on MySQL and SQLite, but PostgreSQL can also be used.

Of the three, SQLite requires the least configuration but is also the least performant and may make it difficult to operate in the future.

We recommend either MySQL or PostgreSQL.

Whatever your choice, the appropriate database software must be installed.

Create MySQL database and grant access to user

You need to create a database and database-user to use with Tracks, in this guide the name of the database is **tempe**, we create a user **daniel** with a password **DANIEL** with all priviliges to access the database

> *>mysql -u root -p*
>
> *mysql> CREATE DATABASE **tempe** CHARACTER SET UTF8;*
>
> *mysql> CREATE USER **daniel**@localhost IDENTIFIED WITH mysql_native_password BY '**DANIEL**';*
>
> *mysql> GRANT USAGE ON *.* TO daniel@localhost;*
>
> *mysql> GRANT ALL PRIVILEGES ON **tempe** .* TO daniel@localhost;*
>
> *mysql> GRANT GRANT OPTION ON **tempe** .* TO daniel@localhost; # Enables you to grant to or revoke from other users those privileges that you yourself possess.*
>
> *mysql>quit;*

Note: for the dump of the database you need to add the **global** PROCESS privilege to the user running the command

> *mysql>GRANT PROCESS ON *.* TO daniel@localhost;*

you can use the following to verify that the database and the user were created

> *mysql> SHOW DATABASES;*
>
> *mysql> SELECT User FROM mysql.user;*
>
> *mysql> quit*

if needed you can restart the MySQL service

> *>service mysql restart*

you can use the following that MySQL is running ok

> *>service mysql status*

to explicitly change an already existent user to use mysql_native_password you can use these commands

> *mysql> ALTER USER 'daniel'@'localhost' IDENTIFIED WITH mysql_native_password BY 'daniel';*

> *mysql> FLUSH PRIVILEGES;*

Install the required Ruby libraries known as ‘gems’

The Bundler tool makes it easy to install all the gems that Tracks needs, and ensures that they are all the correct versions.

go to the directory where you installed Tracks and run

> *>cd /var/www/tracks*
>
> *>bundle config set without 'development test sqlite'*
>
> *>bundle install*

This can take some time depending on the speed of your internet connection and the speed of the system you are installing Tracks on.

later in time you can update with

> *>bundle update*

Populate database

This will set up your database with the required structure to hold Tracks’ data.

go to the directory where you installed Tracks and run

> *>cd /var/www/tracks*
>
> ~~*>rake --rakefile=/var/www/tracks/Rakefile db:migrate RAILS_ENV=production*~~
>
> *>bundle exec rake db:migrate RAILS_ENV=production*

Precompile assets

Static assets (images, stylesheets, and javascript) need to be compiled in order for them to work correctly with the new asset pipeline feature in Rails.

go to the directory where you installed Tracks and run

> *>cd /var/www/tracks*
>
> ~~*>rake --rakefile=/var/www/tracks/Rakefile assets:precompile RAILS_ENV=production*~~
>
> *>bundle exec rake assets:precompile RAILS_ENV=production*

ToDo: test this

Optional; if you don't have SSL certificate

generate certificates, e.g. in /home/ubuntu/certificates

>openssl req -newkey rsa:2048 -nodes -keyout key.pem -out req.pem
>openssl x509 -req -days 365 -in req.pem -signkey key.pem -out cert.pem

Run tracks with thin + ssl

(in tracks-2.3.0 folder)

>bundle exec thin start -e production --ssl --ssl-verify --ssl-key-file /home/ubuntu/certificates/key.pem --ssl-cert-file /home/ubuntu/certificates/cert.pem

Run Apache

Enable module rewrite

> *>a2enmod rewrite*

Activate the new configuration

> *>systemctl restart apache2*

you can check the status, to see that up to here everything is ok, with

> *>systemctl status apache2.service*
>
> *>journalctl -xe*

if apache was installed correctly and is running go with your browser to http://localhost you should see the Apache webserver page

To enable or disable the automatic start of Apache2 service when you boot your machine use

> *>systemctl enable apache2.service*
>
> *>systemctl disable apache2.service*

To start or stop manually the Apache2 service use

> *>systemctl start apache2.service*
>
> *>systemctl stop apache2.service*

Setup Apache to run your website

at /etc/apache2/sites-available/ create the file tracks.conf

you can make a copy from 000-default.conf

IMPORTANT

The default/standard port for http is 80.

As I already have a web site already running I select the port 8079

(the port 8080 is another usual choice, that in my case is also already in use for VNC)

NOTE: if you go for the standard port 80, you need to remove the symbolic link "000-default.conf" in /etc/apache2/sites-enabled to have only your Tracks web "enabled" and running for Port 80 (also you need to use 80 where I refer to 8079)

(the use of libapache2-mod-passenger makes Passenger in Apache to auto-detect the rails app)

edit your website configuration file

> *>nano /etc/apache2/sites-available/tracks.conf*

```
<VirtualHost *:8079>
  ServerName tracks.local

  DocumentRoot /var/www/tracks/public
  AcceptPathInfo on

  ErrorDocument 403 "<h1>Site update in progress. Check back in a few minutes.</h1>"

  <Directory /var/www/tracks/public>
    Require all granted
    AllowOverride All
  </Directory>

  ## Logging
  ErrorLog "/var/log/apache2/tracks_error.log"
  ServerSignature Off
  CustomLog "/var/log/apache2/tracks_access.log" combined
</VirtualHost>
```

now edit the file /etc/apache2/ports.conf and add the Listen for port 8079

> *>nano /etc/apache2/ports.conf*

```
Listen 80
Listen 8079

<ifmodule ssl_module="">
  Listen 443
</ifmodule>

<ifmodule mod_gnutls.c="">
  Listen 443
</ifmodule> create link
```

finally create a symbolic link so Apache services it

> *>cd /etc/apache2/sites-enabled*
> *>ln -s ../sites-available/tracks.conf ./*

restart Apache

> *>systemctl restart apache2*

By default, the Passenger log file is the global (not the per-vhost) Apache error log file.

This is typically located in /var/log/apache2/error_log.

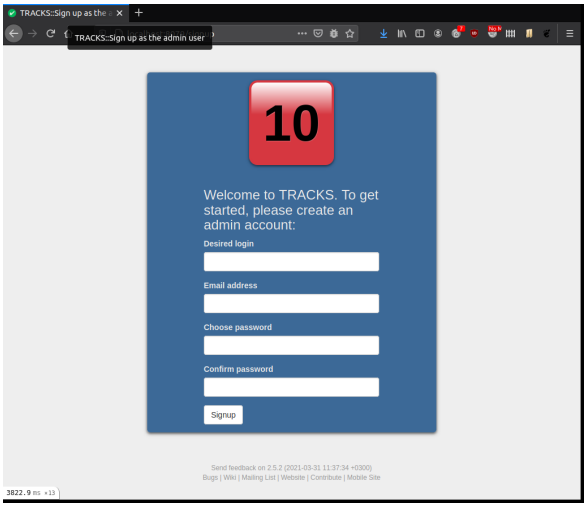
You can find out the exact location of the error log by running

>passenger-config --detect-apache2.

You'll need to create and admin account first.

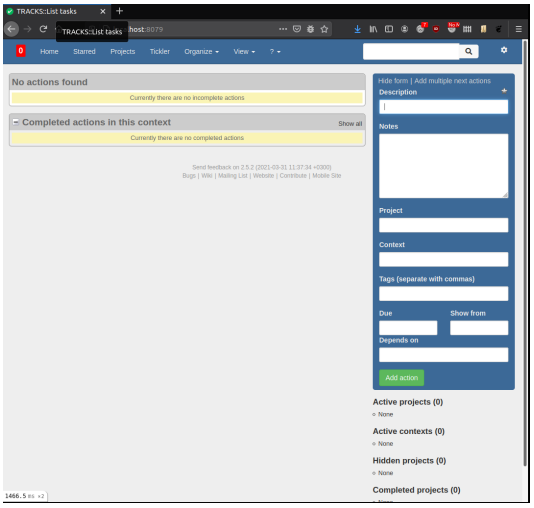
Open the URL http://localhost:8079/signup to start tracks for the first time

From the admin account you can create as many users as you want.



user:daniel

pass:daniel



http://localhost:8079/tracks

If you need to access Tracks from a mobile/cellular phone browser, visit http://yourdomain.com/mobile/.

This mobile version is a special, lightweight version of Tracks, designed to use on a mobile browser.

Customise Tracks

Once logged in, add some Contexts and Projects, and then go ahead and add your actions. You might also want to visit the Preferences page to edit various settings to your liking.

next time just connect using your internet browser to

http://localhost:8079/

http://{ip of the machine with apache}:8079/

e.g. http://192.168.11.15:8079/

if you used port 80 you can directly do

http://localhost/

http://{ip of the machine with apache}/

e.g. http://192.168.11.15/