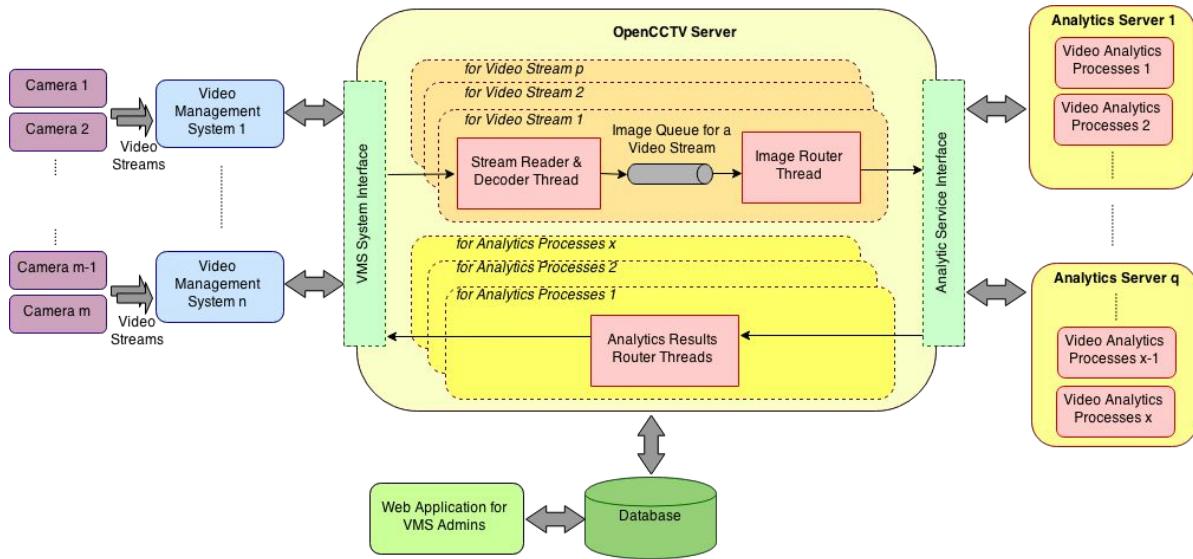


OpenCCTV Platform Installation Manual

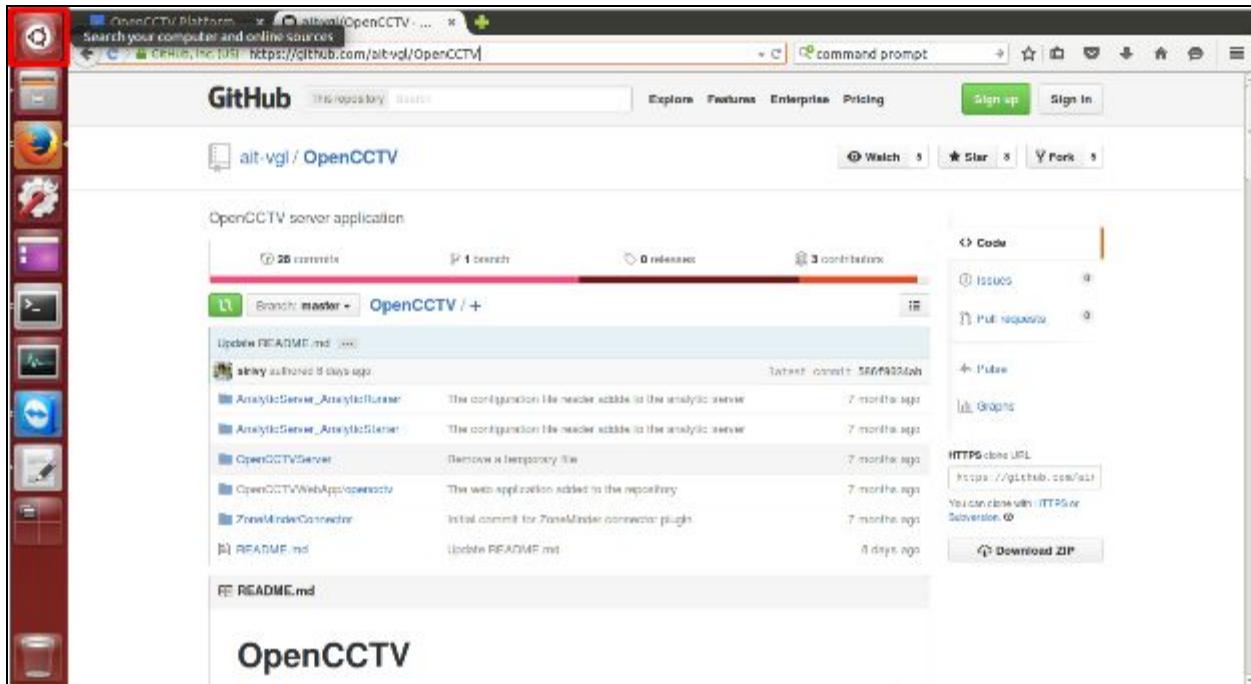
System diagram of OpenCCTV system



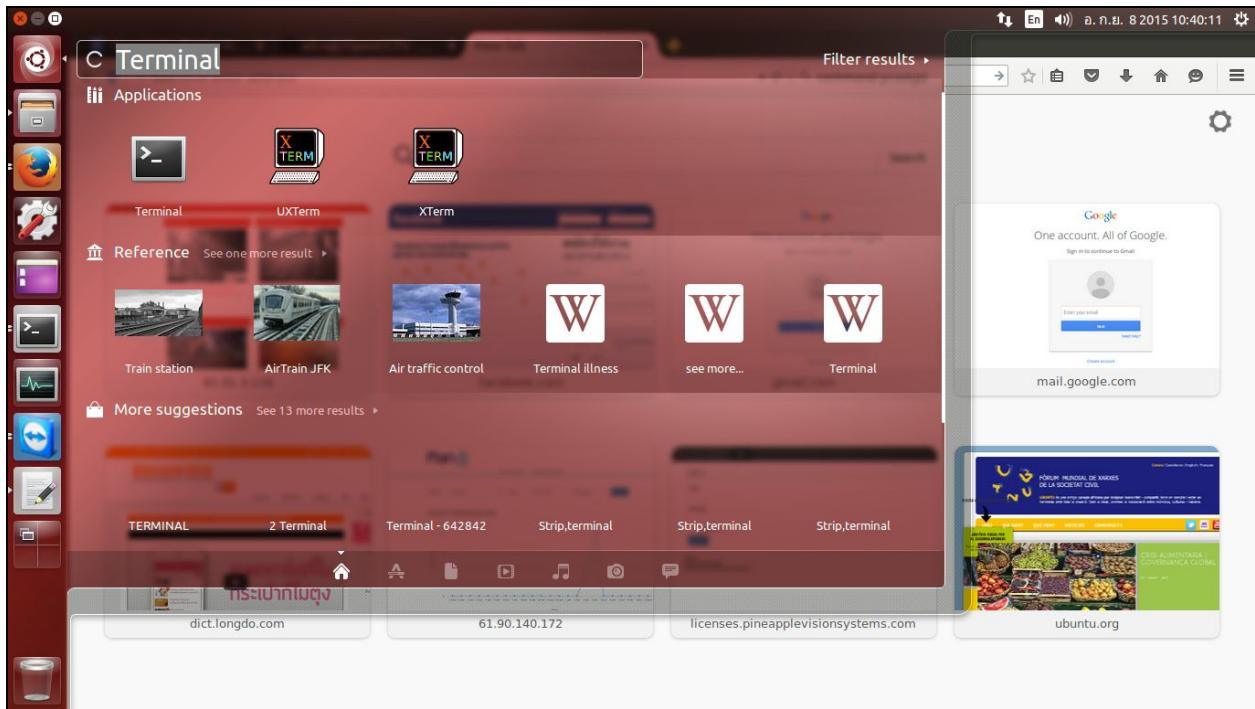
The OpenCCTV platform is developed on Ubuntu operating system. A web application is developed using Ruby on Rails architecture. The backend system is developed using C/C++. Ubuntu 14.04 is required for the OpenCCTV deployment. User can download Ubuntu ISO image from <http://www.ubuntu.com/download/desktop>. After user finishes Ubuntu installation, user can follow the following steps to deploy OpenCCTV platform.

Step 1: Clone OpenCCTV Platform repository

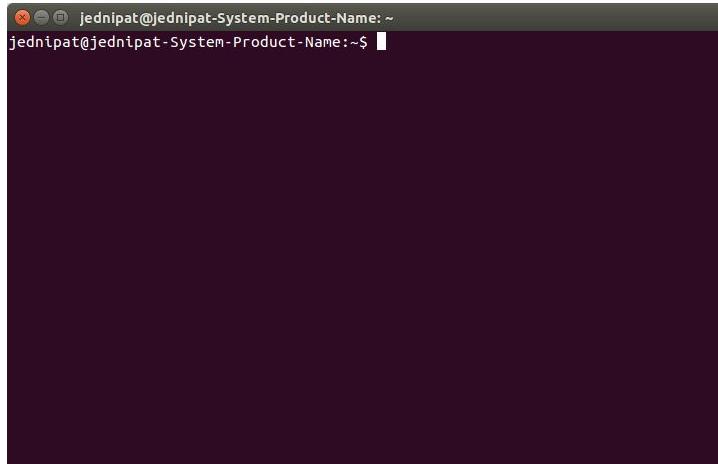
The OpenCCTV platform is an open source project. Source code is available on Github: <https://github.com/ait-vgl/OpenCCTV>. The first step of installation is cloning a repository to user's computer. User first needs to open a Terminal window, which is a command prompt tool in Ubuntu, by clicking at the icon on the top of toolbar on the left side of desktop screen.



Then, user has to type a word “Terminal” in a text field as shown in figure below. After that, user has to click on an icon “Terminal”.



A terminal window will be pop-up and waits for a command line from user. A default directory of terminal window is /home/your_username.



At this step, user has to clone the OpenCCTV repository to user's computer by executing the following command:

```
git clone https://github.com/ait-vgl/OpenCCTV.git
```

A result from executing this command is shown in figure below.

```
jednipat@jednipat-System-Product-Name: ~/works
jednipat@jednipat-System-Product-Name:~/works$ git clone https://github.com/ait-vgl/OpenCCTV.git
Cloning into 'OpenCCTV'...
remote: Counting objects: 540, done.
remote: Total 540 (delta 0), reused 0 (delta 0), pack-reused 540
Receiving objects: 100% (540/540), 1.82 MiB | 1.02 MiB/s, done.
Resolving deltas: 100% (126/126), done.
Checking connectivity... done.
jednipat@jednipat-System-Product-Name:~/works$
```

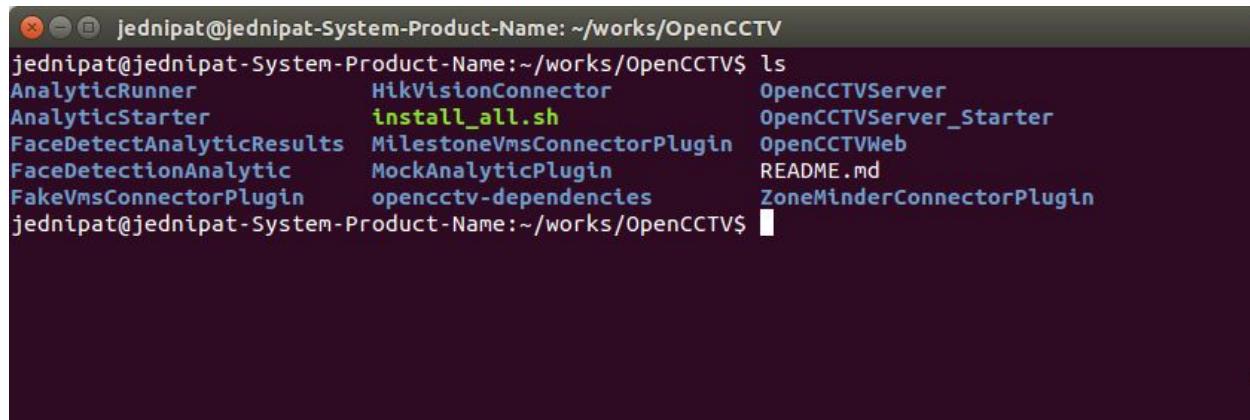
At this step, user will get a folder named “OpenCCTV”. To change directory, user has to type in a command “cd OpenCCTV” and press an enter button.

```
jednipat@jednipat-System-Product-Name: ~/works
jednipat@jednipat-System-Product-Name:~/works$ cd OpenCCTV
```

Now your current directory is changed to OpenCCTV directory. An OpenCCTV folder consists of 13 subfolders which are

- **AnalyticRunner** is a module which works with AnalyticStarter to start/stop analytic module.
- **AnalyticStarter**
- **FaceDetectAnalyticResults** is a simple Ruby on Rails web application for displaying the output result from FaceDetectionAnalytic.
- **FaceDetectionAnalytic** is a simple face detection software. The output of the software is stored in the database used by FaceDetectAnalyticResults.
- **FakeVmsConnectorPlugin** is an example of VMS connector.
- **HikVisionConnector** is a connector for getting a video stream from HikVision NVR.
- **MilestoneVmsConnectorPlugin** is a connector for getting a video stream from Milestone VMS.
- **MockAnalyticPlugin** is an example of analytic module. If user would like to use your own analytic, user has to follow the APIs in this example. User also can look into the FaceDetectionAnalytic as an example too.
- **opencctv-dependencies** contains libraries required for OpenCCTV platform.
- **OpenCCTVServer** is a module which works with OpenCCTVServer_Starter. This module is used to start/stop server.
- **OpenCCTVServer_Starter**

- **OpenCCTVWeb** is a main Ruby on Rails web application source code.
- **ZoneMinderConnectorPlugin** is a connector used for getting a video stream from ZoneMinder VMS.



```
jednipat@jednipat-System-Product-Name:~/works/OpenCCTV
jednipat@jednipat-System-Product-Name:~/works/OpenCCTV$ ls
AnalyticRunner          HikVisionConnector           OpenCCTVServer
AnalyticStarter          install_all.sh            OpenCCTVServer_Starter
FaceDetectAnalyticResults MilestoneVmsConnectorPlugin OpenCCTVWeb
FaceDetectionAnalytic   MockAnalyticPlugin        README.md
FakeVmsConnectorPlugin   opencctv-dependencies     ZoneMinderConnectorPlugin
jednipat@jednipat-System-Product-Name:~/works/OpenCCTV$
```

Step 2: Ruby on Rails installation

Reference: https://cis.ait.asia/course_offerings/224/rails-companion

To set up OpenCCTV Web application, user has to install Ruby on Rails framework. The instruction about Ruby on Rails installation was written by Dr. Matthew N. Dailey. User can visit https://cis.ait.asia/course_offerings/224/rails-companion and follow the instruction step-by-step. If user can not access the reference link, user can follow the instruction below, which is copied from the reference link.

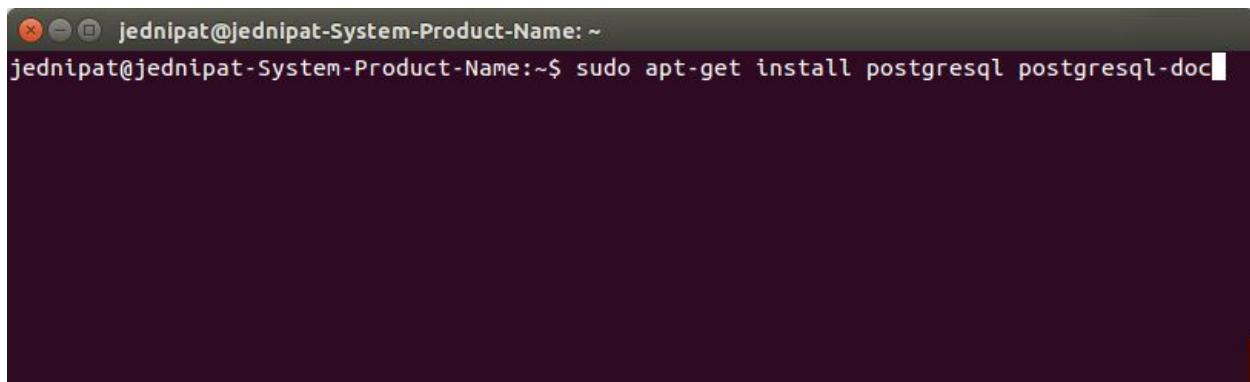
These instructions are oriented towards the **Ubuntu 14.04 (Trusty Tahr)** desktop installation. For other Linux distros, the instructions can vary.

Please note that this document provides quick-start information only. If you need more detailed information, look through the reference section or search the Web.

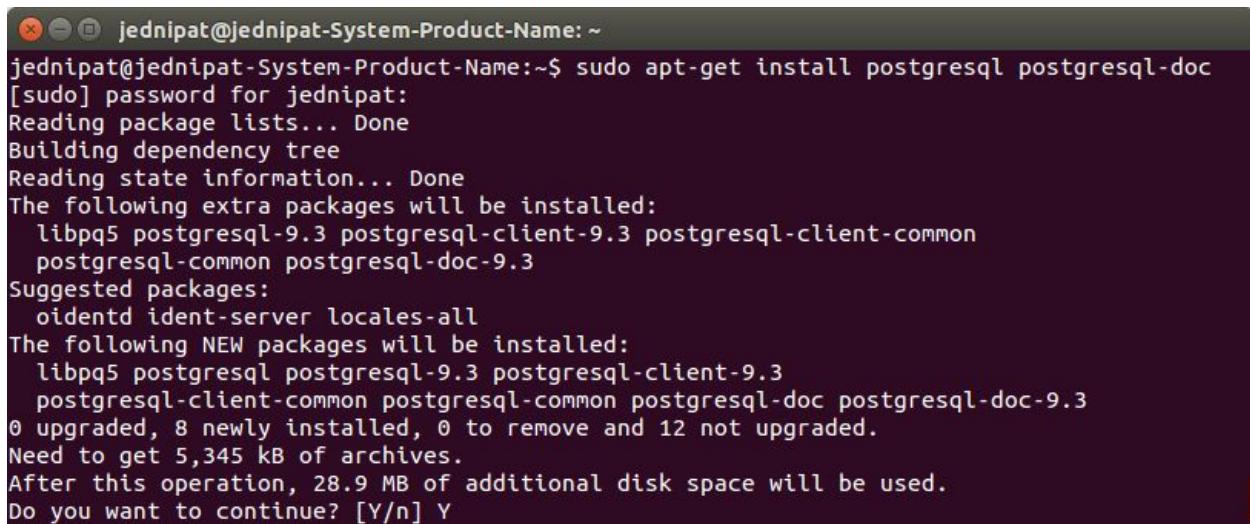
PostgreSQL Installation

Install PostgreSQL before you create your Rails projects. This is simple with Ubuntu: 14.04:

```
$ sudo apt-get install postgresql postgresql-doc
```



```
jednipat@jednipat-System-Product-Name: ~
jednipat@jednipat-System-Product-Name:~$ sudo apt-get install postgresql postgresql-doc
```



```
jednipat@jednipat-System-Product-Name: ~
jednipat@jednipat-System-Product-Name:~$ sudo apt-get install postgresql postgresql-doc
[sudo] password for jednipat:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libpq5 postgresql-9.3 postgresql-client-9.3 postgresql-client-common
  postgresql-common postgresql-doc-9.3
Suggested packages:
  identd ident-server locales-all
The following NEW packages will be installed:
  libpq5 postgresql postgresql-9.3 postgresql-client-9.3
  postgresql-client-common postgresql-common postgresql-doc postgresql-doc-9.3
0 upgraded, 8 newly installed, 0 to remove and 12 not upgraded.
Need to get 5,345 kB of archives.
After this operation, 28.9 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

Rails Installation

You can install Ruby, Ruby Gems, and other Ruby-related packages using apt-get or from source packages. But these approaches are outdated and lead to major headaches when you need to update to new version or work with older versions.

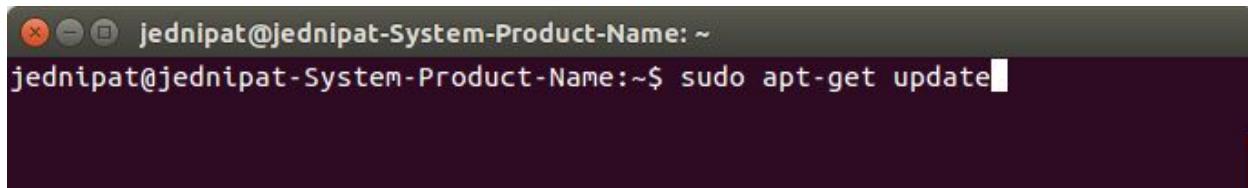
We will be using rbenv (a new lightweight Ruby version management tool) to install Ruby, Ruby Gems, Rails, and other Ruby-related packages. rbenv is a command-line tool allowing you to easily install, manage, and work with multiple Ruby environments including interpreters and sets of Ruby gems.

Ruby Gems is a package manager for the Ruby programming language that provides a standard way of distributing Ruby programs and libraries (in a self-contained format called a "gem"), a tool designed to easily manage the installation of gems, and a server for distributing them.

Housekeeping

First of all, run

```
$ sudo apt-get update
```

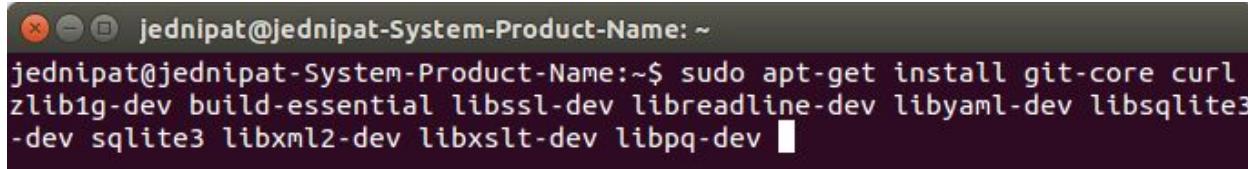


```
jednipat@jednipat-System-Product-Name: ~
jednipat@jednipat-System-Product-Name:~$ sudo apt-get update
```

so that we have the latest sources and update on our machine.

Next, we're going to install git-core and curl which are both required to install and use rbenv, and build-essential which is required to compile Ruby and some of the Ruby gems.

```
$ sudo apt-get install git-core curl zlib1g-dev build-essential libssl-dev libreadline-dev
libyaml-dev libsqlite3-dev sqlite3 libxml2-dev libxslt-dev libpq-dev
```



```
jednipat@jednipat-System-Product-Name: ~
jednipat@jednipat-System-Product-Name:~$ sudo apt-get install git-core curl
zlib1g-dev build-essential libssl-dev libreadline-dev libyaml-dev libsqlite3-
-dev sqlite3 libxml2-dev libxslt-dev libpq-dev
```

```
jednipat@jednipat-System-Product-Name: ~
jednipat@jednipat-System-Product-Name:~$ sudo apt-get install git-core curl
zlib1g-dev build-essential libssl-dev libreadline-dev libyaml-dev libsqlite3
-dev sqlite3 libxml2-dev libxslt-dev libpq-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'libxslt1-dev' instead of 'libxslt-dev'
build-essential is already the newest version.
zlib1g-dev is already the newest version.
zlib1g-dev set to manually installed.
libssl-dev is already the newest version.
libxml2-dev is already the newest version.
libxml2-dev set to manually installed.
The following extra packages will be installed:
comerr-dev krb5-multidev libgssrpc4 libkadm5clnt-mit9 libkadm5srv-mit9
libkdb5-7 libreadline6-dev libtinfo-dev libyaml-0-2
Suggested packages:
krb5-doc krb5-user sqlite3-doc
The following NEW packages will be installed:
comerr-dev curl git-core krb5-multidev libgssrpc4 libkadm5clnt-mit9
libkadm5srv-mit9 libkdb5-7 libpq-dev libreadline-dev libreadline6-dev
libsqlite3-dev libtinfo-dev libxslt1-dev libyaml-0-2 libyaml-dev sqlite3
0 upgraded, 17 newly installed, 0 to remove and 12 not upgraded.
Need to get 1,840 kB of archives.
After this operation, 8,069 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

Installing rbenv

Installing with rbenv is a simple two step process. First you install rbenv, then you install the ruby-build plugin for rbenv.

First install the latest version of rbenv:

```
$ cd
```

```
jednipat@jednipat-System-Product-Name: ~
jednipat@jednipat-System-Product-Name:~$ cd
```

```
$ git clone git://github.com/sstephenson/rbenv.git .rbenv
```

```
jednipat@jednipat-System-Product-Name: ~
jednipat@jednipat-System-Product-Name:~$ git clone git://github.com/sstephen
son/rbenv.git .rbenv
```

```
jednipat@jednipat-System-Product-Name: ~
jednipat@jednipat-System-Product-Name:~$ git clone git://github.com/sstephenson/rbenv.git .rbenv
Cloning into '.rbenv'...
remote: Counting objects: 2075, done.
remote: Total 2075 (delta 0), reused 0 (delta 0), pack-reused 2075
Receiving objects: 100% (2075/2075), 352.58 KiB | 119.00 KiB/s, done.
Resolving deltas: 100% (1265/1265), done.
Checking connectivity... done.
jednipat@jednipat-System-Product-Name:~$
```

```
$ echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc
```

```
jednipat@jednipat-System-Product-Name: ~
jednipat@jednipat-System-Product-Name:~$ echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc
```

```
$ echo 'eval "$(rbenv init -)"' >> ~/.bashrc
```

```
jednipat@jednipat-System-Product-Name: ~
jednipat@jednipat-System-Product-Name:~$ echo 'eval "$(rbenv init -)"' >> ~/.bashrc
```

```
$ exec $SHELL
```

```
jednipat@jednipat-System-Product-Name: ~
jednipat@jednipat-System-Product-Name:~$ exec $SHELL
```

Next install the ruby-build plugin for rbenv:

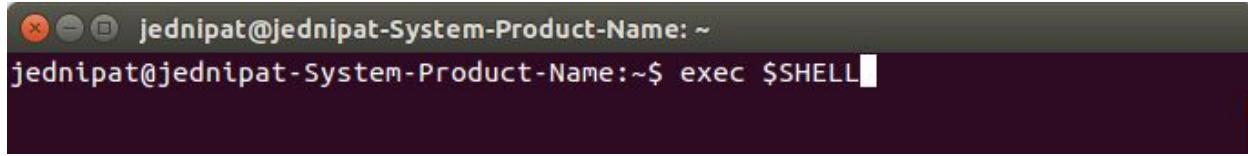
```
$ git clone git://github.com/sstephenson/ruby-build.git ~/.rbenv/plugins/ruby-build
```

```
jednipat@jednipat-System-Product-Name: ~
jednipat@jednipat-System-Product-Name:~$ git clone git://github.com/sstephenson/ruby-build.git ~/.rbenv/plugins/ruby-build
Cloning into '/home/jednipat/.rbenv/plugins/ruby-build'...
remote: Counting objects: 5021, done.
remote: Total 5021 (delta 0), reused 0 (delta 0), pack-reused 5021
Receiving objects: 100% (5021/5021), 927.94 KiB | 529.00 KiB/s, done.
Resolving deltas: 100% (2710/2710), done.
Checking connectivity... done.
jednipat@jednipat-System-Product-Name:~$
```

```
$ echo 'export PATH="$HOME/.rbenv/plugins/ruby-build/bin:$PATH"' >> ~/.bashrc
```

```
jednipat@jednipat-System-Product-Name: ~
jednipat@jednipat-System-Product-Name:~$ echo 'export PATH="$HOME/.rbenv/plugins/ruby-build/bin:$PATH"' >> ~/.bashrc
```

```
$ exec $SHELL
```

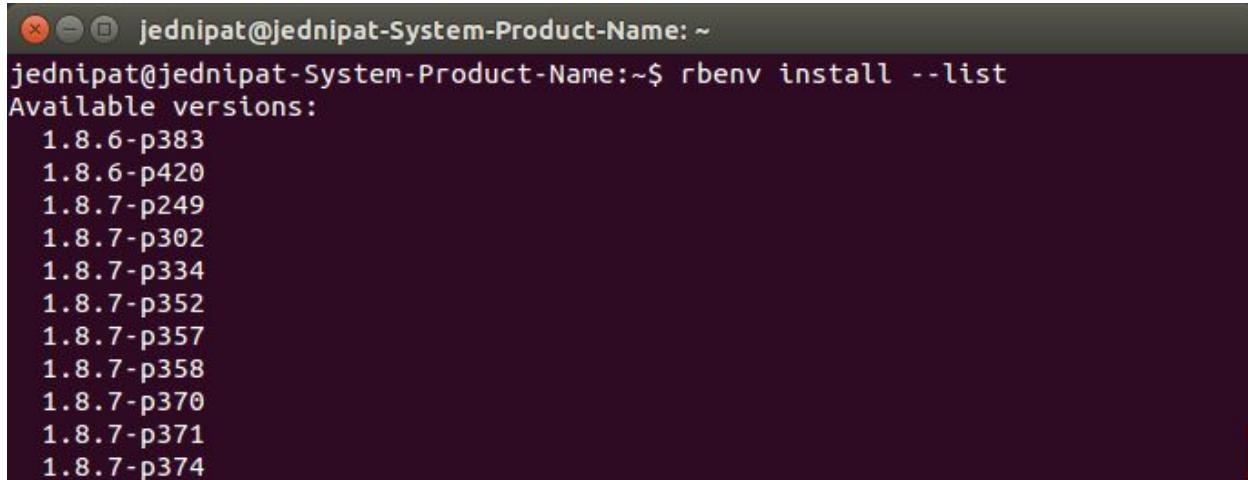


```
jednipat@jednipat-System-Product-Name: ~
jednipat@jednipat-System-Product-Name:~$ exec $SHELL
```

Installing Ruby

Now we can install any version of Ruby we like. To check what releases of Ruby are available, you could run

```
$ rbenv install --list
```



```
jednipat@jednipat-System-Product-Name: ~
jednipat@jednipat-System-Product-Name:~$ rbenv install --list
Available versions:
 1.8.6-p383
 1.8.6-p420
 1.8.7-p249
 1.8.7-p302
 1.8.7-p334
 1.8.7-p352
 1.8.7-p357
 1.8.7-p358
 1.8.7-p370
 1.8.7-p371
 1.8.7-p374
```

and see that (at the time I'm writing) the latest release is 2.1.2-p95. Install it:

```
$ rbenv install 2.1.2
```

```
$ rbenv global 2.1.2
```

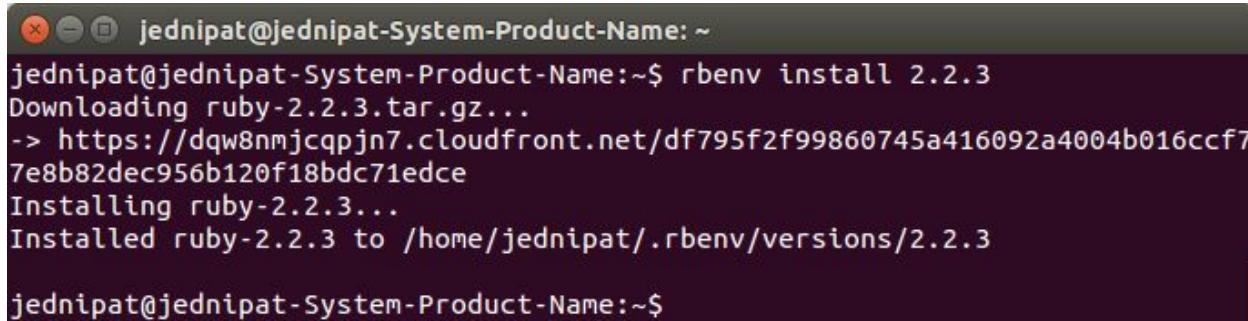
```
$ ruby -v
```

You should see output similar to this:

```
ruby 2.1.2p95 (2014-05-08 revision 45877) [x86_64-linux]
```

NOTE: On 8 September 2015, the latest release is 2.2.3. User can install the latest release by executing the following commands:

```
$ rbenv install 2.2.3
```



```
jednipat@jednipat-System-Product-Name: ~
jednipat@jednipat-System-Product-Name:~$ rbenv install 2.2.3
Downloading ruby-2.2.3.tar.gz...
-> https://dqw8nmjcqpjn7.cloudfront.net/df795f2f99860745a416092a4004b016ccf7
7e8b82dec956b120f18bdc71edce
Installing ruby-2.2.3...
Installed ruby-2.2.3 to /home/jednipat/.rbenv/versions/2.2.3
jednipat@jednipat-System-Product-Name:~$
```

NOTE: This step may take some time.

```
$ rbenv global 2.2.3
```

```
$ ruby -v
```

You should see this output:

```
jednipat@jednipat-System-Product-Name: ~  
jednipat@jednipat-System-Product-Name:~$ rbenv global 2.2.3  
jednipat@jednipat-System-Product-Name:~$ ruby -v  
ruby 2.2.3p173 (2015-08-18 revision 51636) [i686-linux]  
jednipat@jednipat-System-Product-Name:~$ █
```

Ruby Gems

The following step is to tell Rubygems not to install the documentation for each package locally. you can simply update your `~/.gemrc` file.

```
$ echo "gem: --no-ri --no-rdoc" > ~/.gemrc
```

Next, check your Ruby Gems version:

```
$ gem -v
```

```
jednipat@jednipat-System-Product-Name: ~  
jednipat@jednipat-System-Product-Name:~$ echo "gem: --no-ri --no-rdoc" > ~/.  
gemrc  
jednipat@jednipat-System-Product-Name:~$ gem -v  
2.4.5.1  
jednipat@jednipat-System-Product-Name:~$ █
```

It should report version 2.2.2. Run the following command to update to the latest version:

```
$ gem update --system
```

```
jednipat@jednipat-System-Product-Name: ~
jednipat@jednipat-System-Product-Name:~$ gem update --system
Updating rubygems-update
Fetching: rubygems-update-2.4.8.gem (100%)
Successfully installed rubygems-update-2.4.8
Installing RubyGems 2.4.8
RubyGems 2.4.8 installed

== 2.4.8 / 2015-06-08

Bug fixes:

* Tightened API endpoint checks for CVE-2015-3900

== 2.4.7 / 2015-05-14

Bug fixes:
```

```
$ gem -v
```

```
jednipat@jednipat-System-Product-Name: ~
RubyGems system software updated
jednipat@jednipat-System-Product-Name:~$ gem -v
2.4.8
jednipat@jednipat-System-Product-Name:~$
```

Installing Rails

Next let's install the latest version of Rails (4.1.5 at the time of writing). To install:

To install NodeJS, you can add it using a PPA repository:

```
$ sudo add-apt-repository ppa:chris-lea/node.js
```

```
jednipat@jednipat-System-Product-Name: ~
jednipat@jednipat-System-Product-Name:~$ sudo add-apt-repository ppa:chris-lea/node.js
```

```
jednipat@jednipat-System-Product-Name: ~
jednipat@jednipat-System-Product-Name:~$ sudo add-apt-repository ppa:chris-lea/node.js
[sudo] password for jednipat:
  Evented I/O for V8 javascript. Node's goal is to provide an easy way to build scalable network programs
  More info: https://launchpad.net/~chris-lea/+archive/ubuntu/node.js
Press [ENTER] to continue or ctrl-c to cancel adding it
```

Press an enter button to proceed to next step

```
jednipat@jednipat-System-Product-Name: ~
jednipat@jednipat-System-Product-Name:~$ sudo add-apt-repository ppa:chris-lea/node.js
[sudo] password for jednipat:
Evented I/O for V8 javascript. Node's goal is to provide an easy way to build scalable network programs
More info: https://launchpad.net/~chris-lea/+archive/ubuntu/node.js
Press [ENTER] to continue or ctrl-c to cancel adding it

gpg: keyring `/tmp/tmpsg_zsksk/secring.gpg' created
gpg: keyring `/tmp/tmpsg_zsksk/pubring.gpg' created
gpg: requesting key C7917B12 from hkp server keyserver.ubuntu.com
gpg: /tmp/tmpsg_zsksk/trustdb.gpg: trustdb created
gpg: key C7917B12: public key "Launchpad chrislea" imported
gpg: Total number processed: 1
gpg:                      imported: 1  (RSA: 1)
OK
jednipat@jednipat-System-Product-Name:~$
```

\$ sudo apt-get update

```
jednipat@jednipat-System-Product-Name: ~
jednipat@jednipat-System-Product-Name:~$ sudo apt-get update
```

\$ sudo apt-get install nodejs

```
jednipat@jednipat-System-Product-Name: ~
jednipat@jednipat-System-Product-Name:~$ sudo apt-get install nodejs
```

\$ gem install rails

```
jednipat@jednipat-System-Product-Name: ~
jednipat@jednipat-System-Product-Name:~$ gem install rails
```

```
jednipat@jednipat-System-Product-Name:~$ gem install rails
Fetching: thread_safe-0.3.5.gem (100%)
Successfully installed thread_safe-0.3.5
Fetching: tzinfo-1.2.2.gem (100%)
Successfully installed tzinfo-1.2.2
Fetching: i18n-0.7.0.gem (100%)
Successfully installed i18n-0.7.0
Fetching: activesupport-4.2.4.gem (100%)
Successfully installed activesupport-4.2.4
Fetching: rails-deprecated_sanitizer-1.0.3.gem (100%)
Successfully installed rails-deprecated_sanitizer-1.0.3
Fetching: mini_portile-0.6.2.gem (100%)
Successfully installed mini_portile-0.6.2
Fetching: nokogiri-1.6.6.2.gem (100%)
Building native extensions. This could take a while...
```

You'll need to run the following command to make the rails executable available:

```
$ rbenv rehash
```

```
jednipat@jednipat-System-Product-Name:~$ rbenv rehash
```

Now that you've installed Rails. You can check the version as follows:

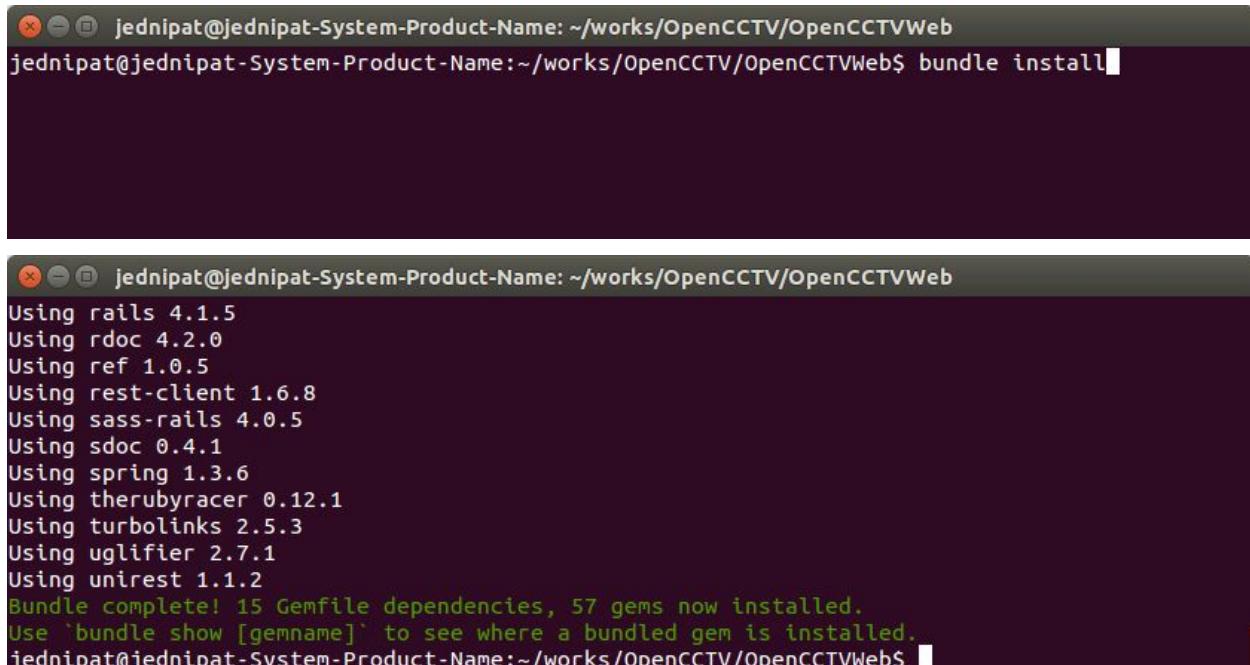
```
$ rails -v
```

```
jednipat@jednipat-System-Product-Name:~$ rails -v
Rails 4.2.4
jednipat@jednipat-System-Product-Name:~$
```

Step 3: OpenCCTV Web application configurations

User has to execute the following commands:

- 1) cd OpenCCTVWeb/
- 2) bundle install



```
jednipat@jednipat-System-Product-Name: ~/works/OpenCCTV/OpenCCTVWeb
jednipat@jednipat-System-Product-Name:~/works/OpenCCTV/OpenCCTVWeb$ bundle install
```



```
jednipat@jednipat-System-Product-Name: ~/works/OpenCCTV/OpenCCTVWeb
Using rails 4.1.5
Using rdoc 4.2.0
Using ref 1.0.5
Using rest-client 1.6.8
Using sass-rails 4.0.5
Using sdoc 0.4.1
Using spring 1.3.6
Using therubyracer 0.12.1
Using turbolinks 2.5.3
Using uglifier 2.7.1
Using unirest 1.1.2
Bundle complete! 15 Gemfile dependencies, 57 gems now installed.
Use `bundle show [gemname]` to see where a bundled gem is installed.
jednipat@jednipat-System-Product-Name:~/works/OpenCCTV/OpenCCTVWeb$
```

- 3) If your computer doesn't have MySQL, you have to install MySQL

```
sudo apt-get install mysql-server
```

While installing MySQL, you will be asked to set username and password for administrator (root).

Then, you have to create MySQL account for using with Ruby On Rails Web application by the following commands:

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

After executing this command, you will be asked to enter a password. You have to fill the same password which is set during MySQL installation. Next, you have to enter the following commands for creating a new username and password, and setting privileges.

```
CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';
```

```
GRANT ALL PRIVILEGES ON *.* TO 'username'@'localhost';
```

```
FLUSH PRIVILEGES;
```

NOTE: Please do not copy these commands and paste into terminal because a single quote character (') will create a problem. You have to type in a command by yourself.

```
jednipat@jednipat-K401LB:~/works/OpenCCTV/OpenCCTVWebApp/opencctv$ mysql -u root  
-p  
Enter password:  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 50  
Server version: 5.5.44-0ubuntu0.14.04.1 (Ubuntu)  
  
Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
mysql> CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';  
Query OK, 0 rows affected (0.41 sec)  
  
mysql> GRANT ALL PRIVILEGES ON *.* TO 'username'@'localhost';  
Query OK, 0 rows affected (0.02 sec)  
  
mysql> FLUSH PRIVILEGES;  
Query OK, 0 rows affected (0.06 sec)  
  
mysql> █
```

If you can execute all three commands without error messages, you have to press Ctrl+d (hold a Ctrl button and press ‘d’ button) to exit mysql command line. In this example, a new user called “username” is created for “localhost”. A *username*’s password is “password”. After that, you have to restart mysql service by executing this command:

```
sudo service mysql restart
```

Before making changes in database.yml, you need to run a command

```
gem install mysql2
```

at opencctv folder.

At this step, user needs to understand database configuration. If user opens a database.yml file in OpenCCTV/OpenCCTVWeb/config, the following lines will be shown up:

```
# MySQL. Versions 5.0+ are recommended.  
#  
# Install the MYSQL driver  
# gem install mysql2  
#  
# Ensure the MySQL gem is defined in your Gemfile
```

```

# gem 'mysql2'
#
# And be sure to use new-style password hashing:
# http://dev.mysql.com/doc/refman/5.0/en/old-client.html
#
default: &default
  adapter: mysql2
  encoding: utf8
  pool: 5
  username: username
  password: password
  host: host
  socket: /var/run/mysqld/mysqld.sock

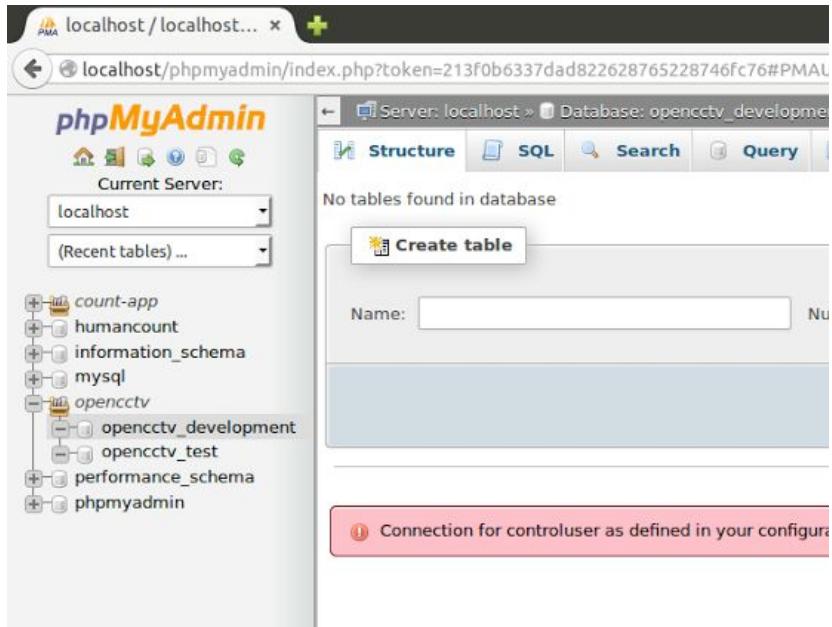
```

Since a user called “username” is created for localhost, you have to change host name from “host” to “localhost”. If you would like to change Web application configurations, please make sure that you create a new username and password, and assign to the same hostname used in database.yml.

If everything is fine and there is no error message, we are now ready for creating a database which will be used by Ruby on Rails Web application. To create a database for Ruby on Rails, at opencctv folder, you have to execute a command

```
rake db:create
```

After executing this command, you should get database as shown in figure below.



NOTE: This Web GUI is PHPMyAdmin. User can install it by executing a

command: sudo apt-get install phpmyadmin

The rake db:create will just create only a database named “opencctv” and two tables: opencctv_development, opencctv_test.

Next, user has to run a command

rake db:migrate

to create all tables required for Ruby on Rails Web application as shown below.

localhost / localhost... x

localhost/phpmyadmin/index.php?token=213f0b6337dad822628765228746fc76#PMAURL-10:db_structure.php?db=opencctv_development&table=&server=1&target=&t=

Search

phpMyAdmin

Current Server: localhost

Recent tables: (Recent tables) ...

Structure SQL Search Query Export Import Operations Privileges Routines Events

Table Action Rows Type Collation Size Overhead

analytics Browse Structure Search Insert Empty Drop 0 InnoDB utf8_unicode_ci 16 Kib

analytic_input_streams Browse Structure Search Insert Empty Drop 0 InnoDB utf8_unicode_ci 32 Kib

analytic_instances Browse Structure Search Insert Empty Drop 0 InnoDB utf8_unicode_ci 32 Kib

analytic_instance_streams Browse Structure Search Insert Empty Drop 0 InnoDB utf8_unicode_ci 64 Kib

cameras Browse Structure Search Insert Empty Drop 0 InnoDB utf8_unicode_ci 32 Kib

schema_migrations Browse Structure Search Insert Empty Drop 14 InnoDB utf8_unicode_ci 16 Kib

servers Browse Structure Search Insert Empty Drop 0 InnoDB utf8_unicode_ci 16 Kib

streams Browse Structure Search Insert Empty Drop 0 InnoDB utf8_unicode_ci 32 Kib

users Browse Structure Search Insert Empty Drop 0 InnoDB utf8_unicode_ci 48 Kib

vmses Browse Structure Search Insert Empty Drop 0 InnoDB utf8_unicode_ci 16 Kib

10 tables Sum 14 InnoDB utf8_unicode_ci 384 Kib 0 B

Check All With selected:

Print view Data Dictionary

Create table

Name: Number of columns:

Step 4: OpenCCTV installation

We are almost ready for OpenCCTV installation. Please make sure that you have installed the required libraries as follows:

- [Boost C++ Libraries](#)
 - Installation guide to Boost:
<http://particlephysicsandcode.com/2013/03/11/installing-boost-1-52-ubuntu-12-04-fedora/>
- OpenCV
 - Installation guide to OpenCV:
http://docs.opencv.org/trunk/doc/tutorials/introduction/linux_install/linux_install.html
- libVLC
 - sudo apt-get install x264 libvlccore-dev libvlc-dev
- libsodium
 - Download libsodium source code from
<https://download.libsodium.org/libsodium/releases/>
 - Extract tar ball file
 - Execute the following command
 - ./configure
 - make
 - sudo make install
- Google Protocol Buffer
 - <https://developers.google.com/protocol-buffers/?hl=en>
- libzmq
 - Download libzeromq source code from <http://zeromq.org/area:download>
 - Extract tar ball file
 - Execute the following command
 - ./configure
 - make
 - sudo make install
- zmq header file
 - Download zmq.hpp from <https://github.com/zeromq/cppzmq>
 - Copy zmq.hpp to /usr/local/include/
- libmysqlconn-dev
 - sudo apt-get install libmysqlcppconn-dev
- Update recently installed libraries by executing a command
 - sudo ldconfig

NOTE: User can check about commands for installing required libraries in opencctv-dependencies folder.

There is one more step before we can compile the source code and install the OpenCCTV. User has to compile two .proto files (image.proto and analytic_result.proto) by using the command below.

```
protoc -I=/home/jednipat/works/OpenCCTV/OpenCCTVServer/src/opencctv/util/serialization/gpb  
--cpp_out=/home/jednipat/works/OpenCCTV/OpenCCTVServer/src/opencctv/util/serialization/gpb  
/home/jednipat/works/OpenCCTV/OpenCCTVServer/src/opencctv/util/serialization/gpb/image.proto
```

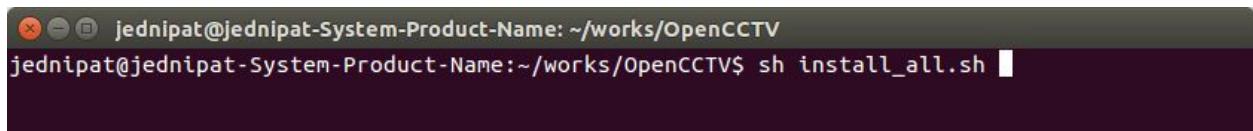
and

```
protoc -I=/home/jednipat/works/OpenCCTV/OpenCCTVServer/src/opencctv/util/serialization/gpb  
--cpp_out=/home/jednipat/works/OpenCCTV/OpenCCTVServer/src/opencctv/util/serialization/gpb  
/home/jednipat/works/OpenCCTV/OpenCCTVServer/src/opencctv/util/serialization/gpb/analytic_result.proto
```

User has to change a path to folder and file to match with your system environment. If there is no error, we can now proceed to final step of installation.

To install the OpenCCTV, user has to execute a command

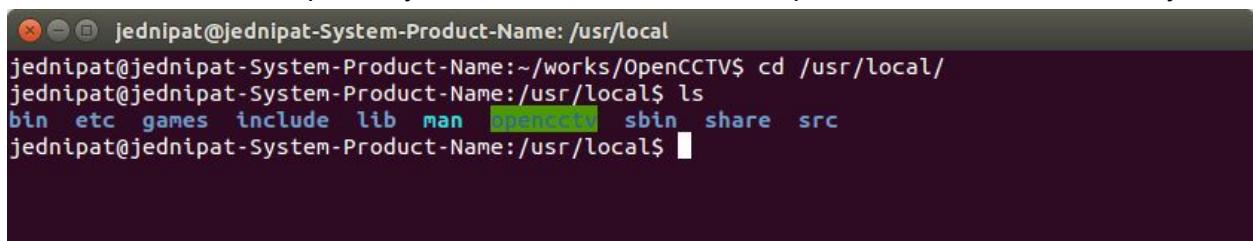
```
sh install_all.sh
```



```
jednipat@jednipat-System-Product-Name: ~/works/OpenCCTV  
jednipat@jednipat-System-Product-Name:~/works/OpenCCTV$ sh install_all.sh
```

Executing this shell script will ask you to enter root's password. After that, you have to read the messages in the terminal carefully to make sure that all modules are compiled successfully without any error messages.

If the installation is completed, you will see a folder named "opencctv" in /usr/local directory.



```
jednipat@jednipat-System-Product-Name: /usr/local  
jednipat@jednipat-System-Product-Name:~/works/OpenCCTV$ cd /usr/local/  
jednipat@jednipat-System-Product-Name:/usr/local$ ls  
bin etc games include lib man opencctv sbin share src  
jednipat@jednipat-System-Product-Name:/usr/local$
```

The opencctv in /usr/local contains the following sub-folders.

```
jednipat@jednipat-System-Product-Name: /usr/local/opencctv
jednipat@jednipat-System-Product-Name:/usr/local/opencctv$ ls
AnalyticRunner  AnalyticStarter  OpenCCTVServerStarter  vms_connectors
analytics        OpenCCTVServer    OpenCCTVWeb
jednipat@jednipat-System-Product-Name:/usr/local/opencctv$ █
```

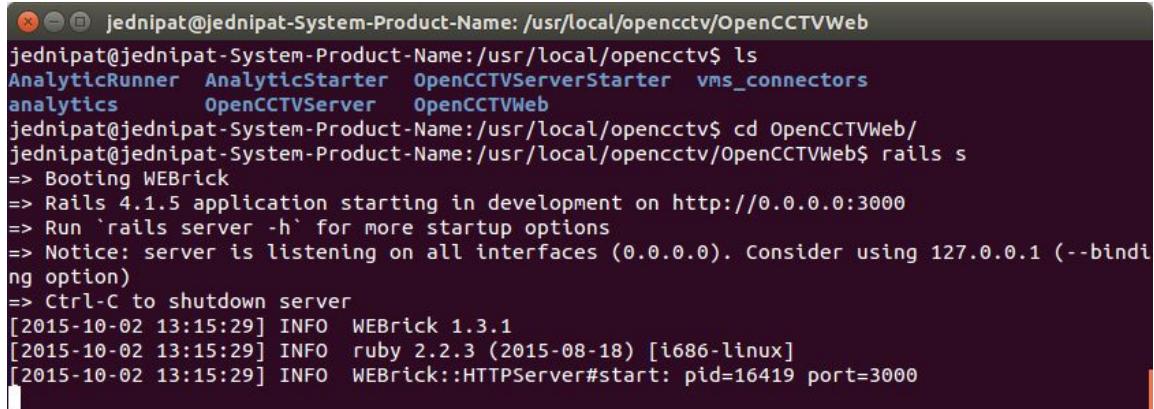
Step 5: Start OpenCCTV services

Once the installation is completed, user will get opencctv folder in /usr/local. We need to start three services which are

1) OpenCCTVWeb

As we explain earlier, the OpenCCTVWeb is a main web application which is used for controlling the whole system. To start this module, we have to execute the following command:

```
rails s
```



```
jednipat@jednipat-System-Product-Name: /usr/local/opencctv/OpenCCTVWeb
jednipat@jednipat-System-Product-Name:/usr/local/opencctv$ ls
AnalyticRunner  AnalyticStarter  OpenCCTVServerStarter  vms_connectors
analytics        OpenCCTVServer   OpenCCTVWeb
jednipat@jednipat-System-Product-Name:/usr/local/opencctv$ cd OpenCCTVWeb/
jednipat@jednipat-System-Product-Name:/usr/local/opencctv/OpenCCTVWeb$ rails s
=> Booting WEBrick
=> Rails 4.1.5 application starting in development on http://0.0.0.0:3000
=> Run `rails server -h` for more startup options
=> Notice: server is listening on all interfaces (0.0.0.0). Consider using 127.0.0.1 (--binding option)
=> Ctrl-C to shutdown server
[2015-10-02 13:15:29] INFO  WEBrick 1.3.1
[2015-10-02 13:15:29] INFO  ruby 2.2.3 (2015-08-18) [i686-linux]
[2015-10-02 13:15:29] INFO  WEBrick::HTTPServer#start: pid=16419 port=3000
```

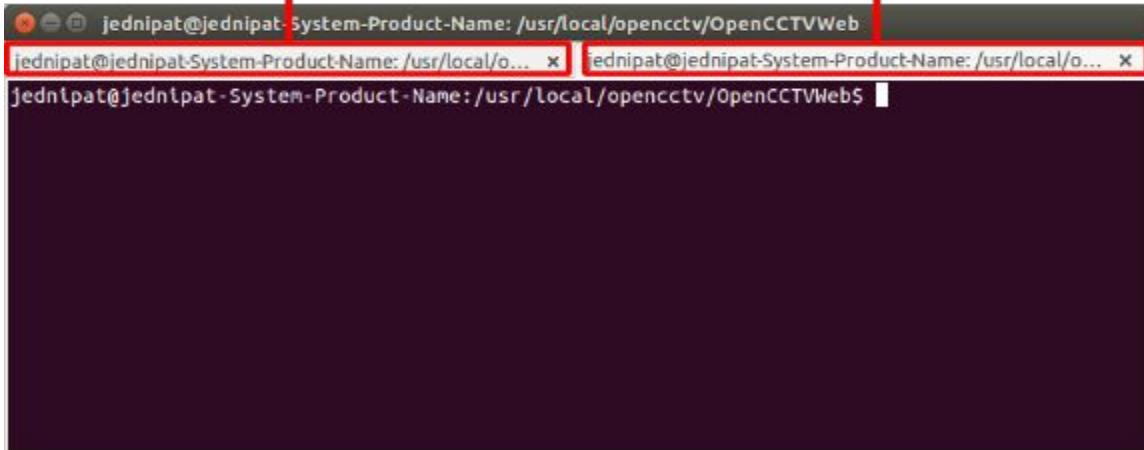
User cannot close this terminal window or kill this process. Otherwise, the web application will stop working. If you want to stop web application, you can hold a control button and press 'c' button on your keyboard (Ctrl+C). This is a shortcut key to kill process via terminal in Ubuntu OS.

2) AnalyticStarter

AnalyticStarter is a module to handle analytic plugin for retrieving video stream from OpenCCTV server and passing a video frame to analytic module. To open a new terminal window without stopping the running process, user can press a hold ctrl and shift button, and the press 't' button on your keyboard (Ctrl+Shift+T). This shortcut key will open a new tab of terminal window within the same window. You will see a terminal window like this.

Currently running a web application service

This terminal will be used for running the AnalyticStarter module.

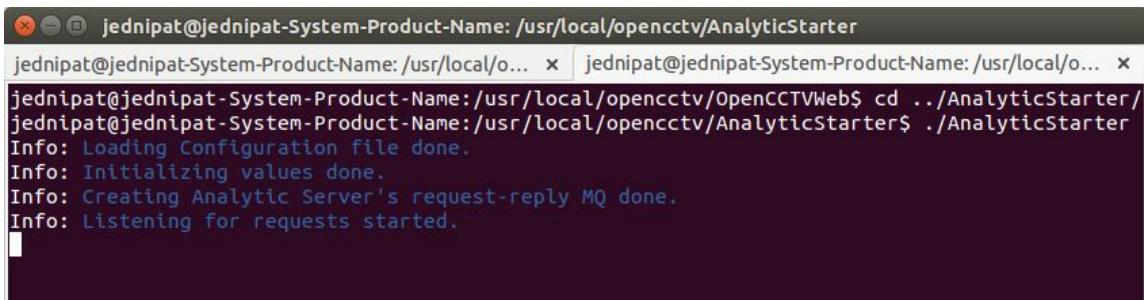


Once we open a new terminal under the same window, we have to change our current directory to AnalyticStarter directory.

```
cd ..../AnalyticStarter
```

Then, user has to execute the AnalyticStarter using a command

```
./AnalyticStarter
```



*As previously mentioned in OpenCCTVWeb, we **cannot close this terminal or kill this process.***

3) OpenCCTVServerStarter

OpenCCTVServerStarter is used to start/stop OpenCCTVServer. To start this service, we will do the same process as we did for AnalyticStarter. User has to open a new terminal, go to OpenCCTVServerStarter directory, and then execute the following command:

```
./OpenCCTVServer_Starter
```

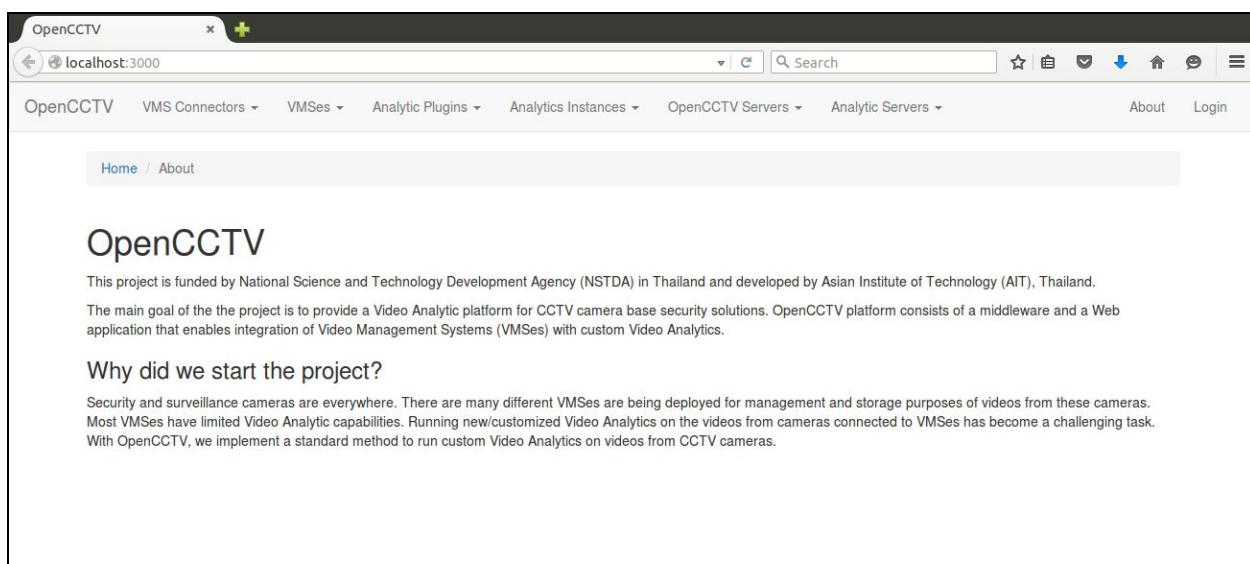
```

jednipat@jednipat-System-Product-Name:/usr/local/openccctv/OpenCCTVServerStarter
jednipat@jednipat-System-Product-Name: ~ jednipat@jednipat-System-Product-Name: ~ jednipat@jednipat-System-Product-Name: ~
jednipat@jednipat-System-Product-Name:/usr/local/openccctv/AnalyticStarter$ cd .. /OpenCCTVServerStarter/
jednipat@jednipat-System-Product-Name:/usr/local/openccctv/OpenCCTVServerStarter$ ./OpenCCTVServer_Starter
Info: OpenCCTV Server Manager started.

```

*As previously mentioned in OpenCCTVWeb, we **cannot close this terminal or kill this process.***

Now, we can check our web application on web browser. User has to enter <http://localhost:3000> in web browser and you should see the main web page of OpenCCTV.

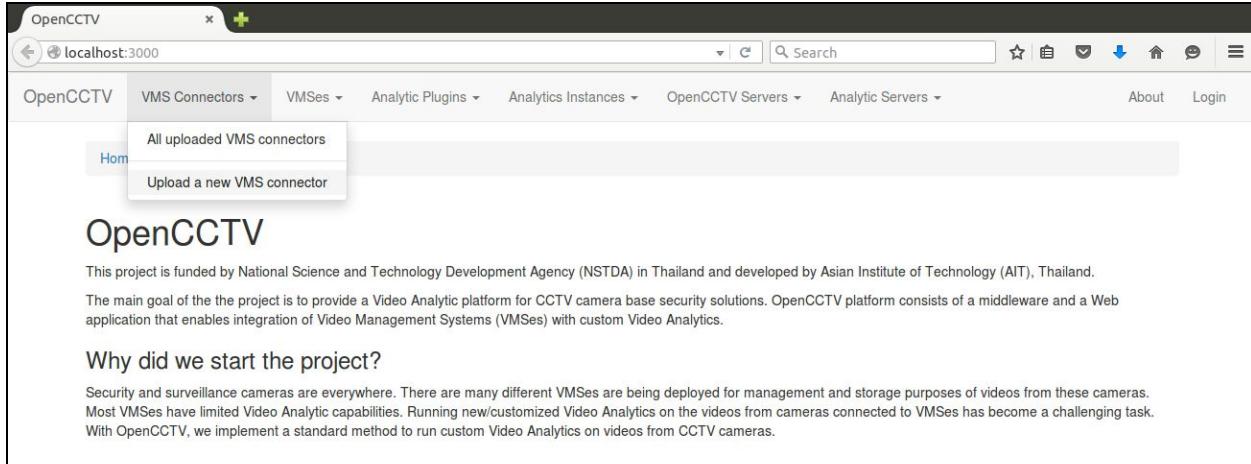


If you see this main web page, it means that your OpenCCTV is ready.

OpenCCTV Platform User Manual

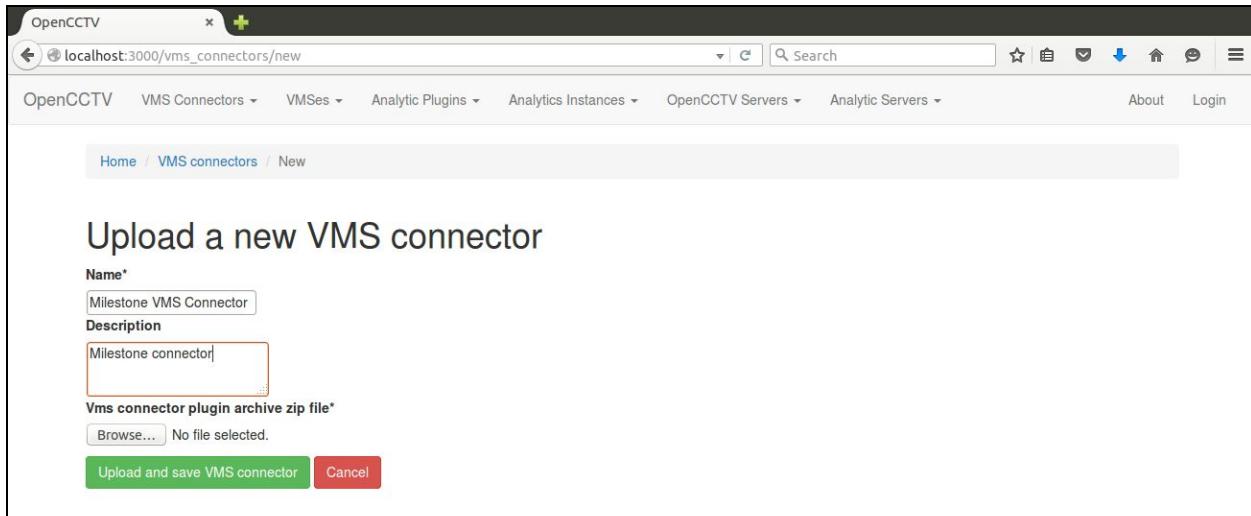
1. Upload VMS Connector

1.1) Click at VMS Connectors->Upload a new VMS Connector



The screenshot shows the OpenCCTV platform interface. At the top, there is a navigation bar with links for 'OpenCCTV', 'VMS Connectors', 'VMSSes', 'Analytic Plugins', 'Analytics Instances', 'OpenCCTV Servers', 'Analytic Servers', 'About', and 'Login'. The 'VMS Connectors' link is currently selected, and a dropdown menu appears below it with two options: 'All uploaded VMS connectors' and 'Upload a new VMS connector'. The main content area is titled 'OpenCCTV' and contains a brief introduction about the project being funded by NSTDA and developed by AIT, Thailand. It also mentions the main goal of providing a Video Analytic platform for CCTV camera base security solutions. Below this, a section titled 'Why did we start the project?' explains the challenge of running custom video analytics on videos from CCTV cameras.

1.2) Fill a connector name and description



The screenshot shows the 'Upload a new VMS connector' form. The URL in the browser is 'localhost:3000/vms_connectors/new'. The form has fields for 'Name*' (containing 'Milestone VMS Connector'), 'Description' (containing 'Milestone connector'), and 'Vms connector plugin archive zip file*'. The 'Browse...' button shows 'No file selected'. There are two buttons at the bottom: 'Upload and save VMS connector' (green) and 'Cancel' (red).

1.3) Compile and upload Milestone VMS connector

1.3.1) Compile Milestone VMS connector

To compile Milestone VMS connector, user has to go to a folder which is cloned from OpenCCTV Github repository and follow the steps below:

```
cd path_to_your_OpenCCTV
cd MilestoneVmsConnectorPlugin
cd Release
make all
```

```
jednipat@jednipat-System-Product-Name: ~/works/OpenCCTV/MilestoneVmsConnector$ ls
AnalyticRunner          MockAnalyticPlugin
AnalyticStarter          opencctv-dependencies
FaceDetectAnalyticResults OpenCCTVServer
FaceDetectionAnalytic   OpenCCTVServer_Starter
FakeVmsConnectorPlugin   OpenCCTVWeb
HikVisionConnector       README.md
install_all.sh           ZoneMinderConnectorPlugin
MilestoneVmsConnectorPlugin
jednipat@jednipat-System-Product-Name:~/works/OpenCCTV$ cd MilestoneVmsConnectorPlugin/
jednipat@jednipat-System-Product-Name:~/works/OpenCCTV/MilestoneVmsConnectorPlugin$ ls
Release  src
jednipat@jednipat-System-Product-Name:~/works/OpenCCTV/MilestoneVmsConnectorPlugin$ cd Release/
jednipat@jednipat-System-Product-Name:~/works/OpenCCTV/MilestoneVmsConnectorPlugin$ make all
```

If the source code is compiled successfully, user will get **libMilestoneVmsConnectorPlugin.so** file.

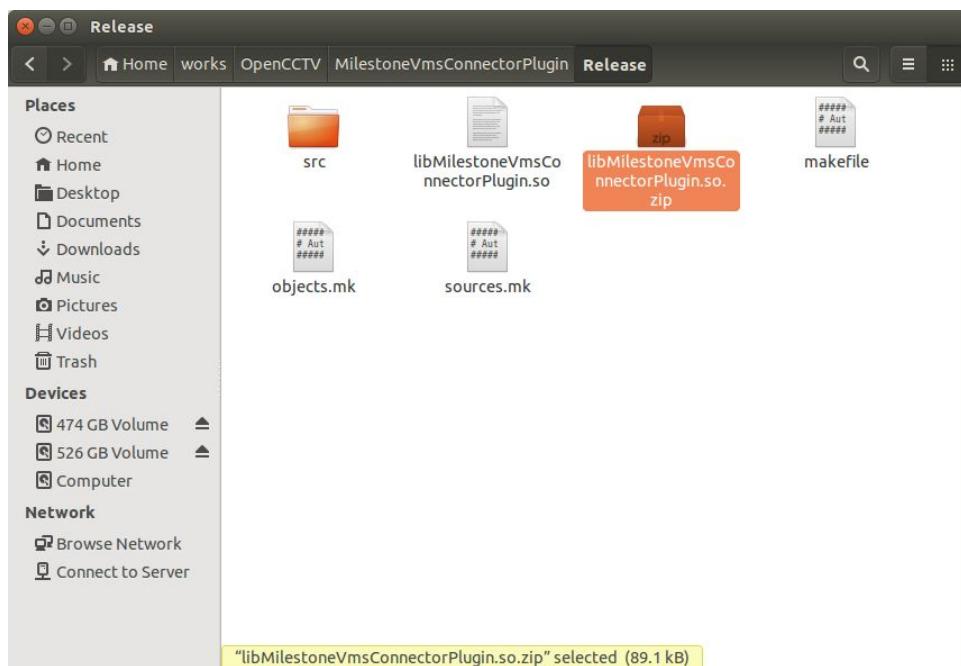
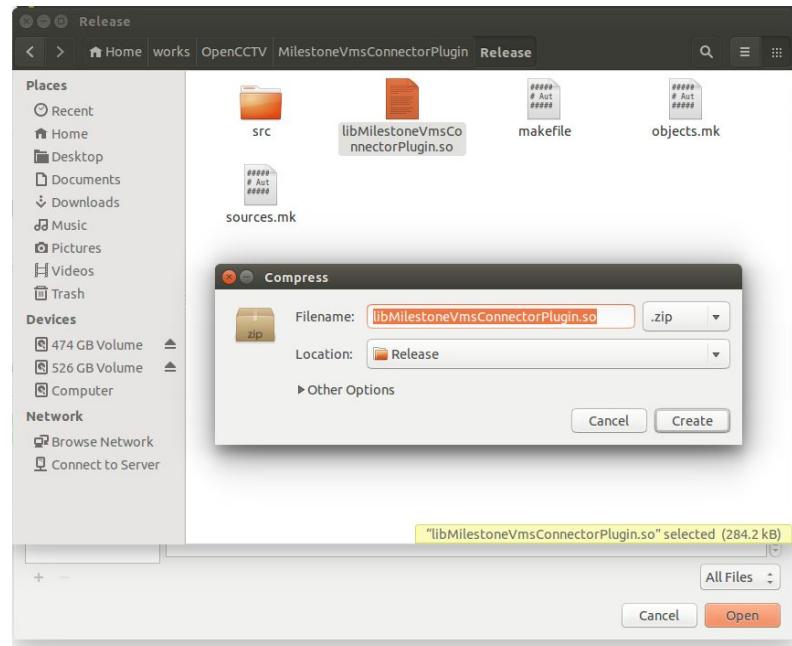
```
jednipat@jednipat-System-Product-Name: ~/works/OpenCCTV/MilestoneVmsConnector$ g++ -O3 -Wall -c -fmessage-length=0 -fPIC -MMD -MP -MF"src/MilestoneVmsConnector.d" -MT"src/MilestoneVmsConnector.d" -o "src/MilestoneVmsConnector.o" "../src/MilestoneVmsConnector.cpp"
Finished building: ../src/MilestoneVmsConnector.cpp

Building target: libMilestoneVmsConnectorPlugin.so
Invoking: GCC C++ Linker
g++ -L/usr/local/lib -shared -o "libMilestoneVmsConnectorPlugin.so" ./src/tcpsocket/Tcpconnector.o ./src/tcpsocket/Tcpstream.o ./src/opencctv/Image.o ./src/milestone/MilestoneMessage.o ./src/milestone/TcpMpegDecoder.o ./src/MilestoneVmsConnector.o -lboost_system
Finished building target: libMilestoneVmsConnectorPlugin.so

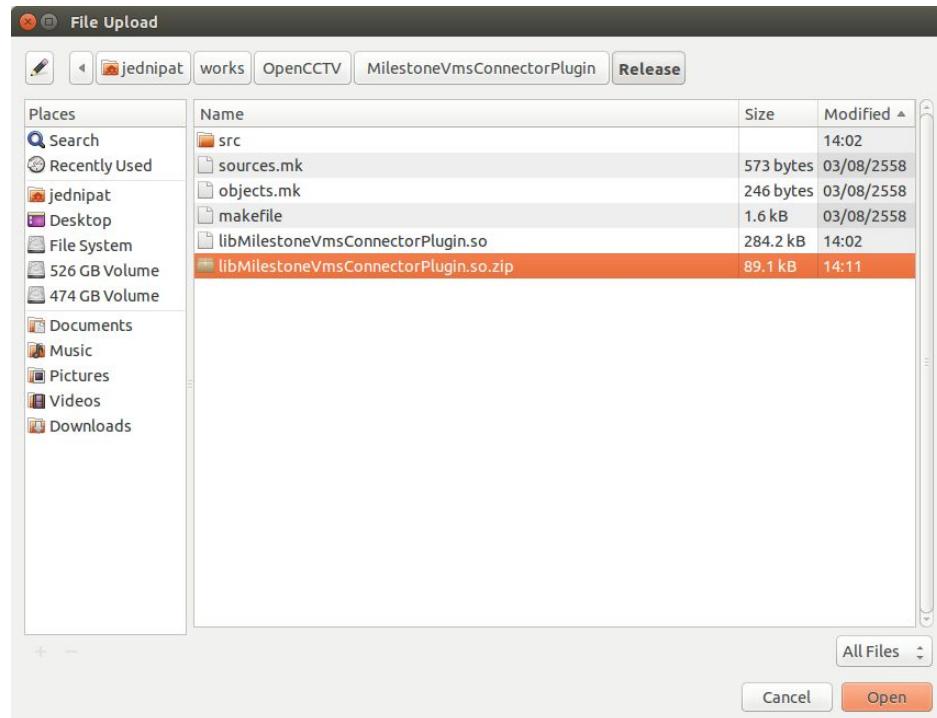
jednipat@jednipat-System-Product-Name:~/works/OpenCCTV/MilestoneVmsConnectorPlugin$ ls
libMilestoneVmsConnectorPlugin.so  makefile  objects.mk  sources.mk  src
jednipat@jednipat-System-Product-Name:~/works/OpenCCTV/MilestoneVmsConnectorPlugin$
```

1.3.2) Create a zip file of VMS connector

Once we get **libMilestoneVmsConnectorPlugin.so**, user has to create a zip file of this file for uploading.



1.3.3) Select a file for uploading

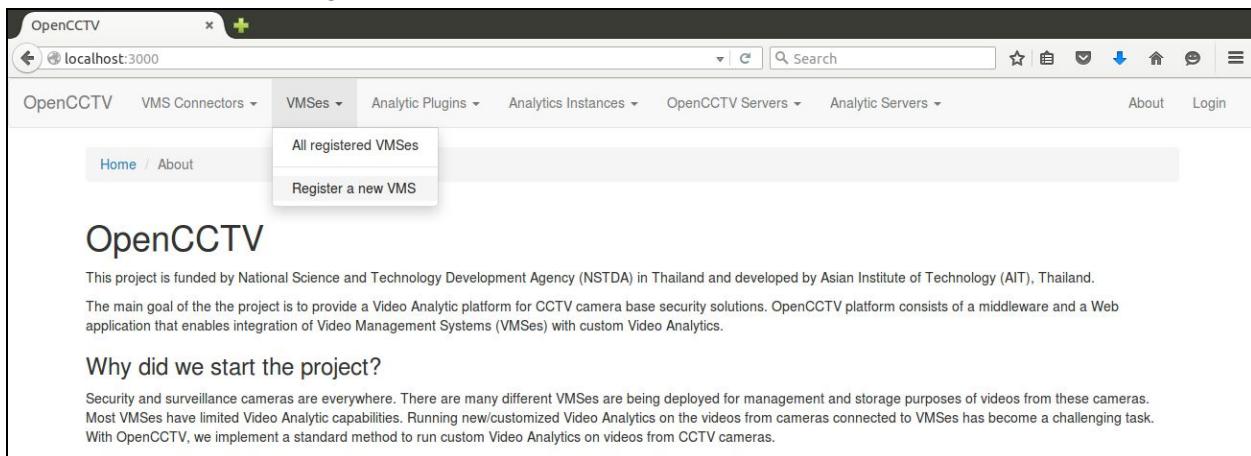


1.3.4) Upload and save VMS connector to server

After uploading, user should see a verification report. If a VMS connector is implemented correctly, the system will show a status “Verified”. This means that the uploaded VMS connector is ready for using.

2. Register a VMS

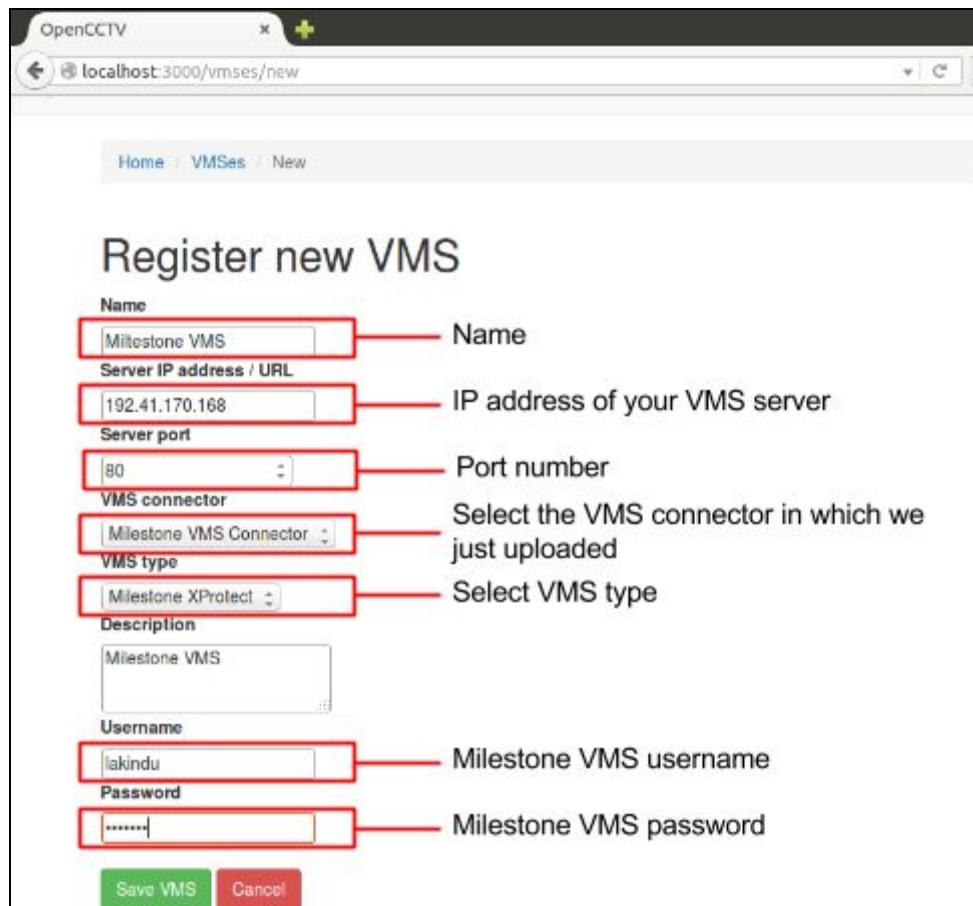
2.1) Click at VMSSes->Register a new VMS



The screenshot shows the OpenCCTV web application interface. At the top, there is a navigation bar with links for 'OpenCCTV', 'VMS Connectors', 'VMSSes', 'Analytic Plugins', 'Analytics Instances', 'OpenCCTV Servers', 'Analytic Servers', 'About', and 'Login'. Below the navigation bar, there is a breadcrumb trail: 'Home / About'. A dropdown menu for 'VMSSes' is open, showing options: 'All registered VMSSes' (which is selected and highlighted in blue) and 'Register a new VMS'. The main content area has a heading 'OpenCCTV' and a brief description about the project being funded by NSTD and developed by AIT. It also mentions the main goal of providing a Video Analytic platform for CCTV cameras. A section titled 'Why did we start the project?' explains that security cameras are everywhere and that running custom video analytics on them has become challenging. It notes that OpenCCTV provides a standard method to run custom video analytics on videos from CCTV cameras.

2.2) Register a new VMS

Before starting this process, please make sure that you have information about your VMS. In this example, we will use Milestone VMS as an example because we just uploaded the Milestone VMS connector in the previous section.



The screenshot shows the 'Register new VMS' form. The form fields are as follows:

- Name:** Milestone VMS
- Server IP address / URL:** 192.41.170.168
- Server port:** 80
- VMS connector:** Milestone VMS Connector
- VMS type:** Milestone XProtect
- Description:** Milestone VMS
- Username:** lakindu
- Password:** (redacted)

At the bottom of the form are two buttons: 'Save VMS' (green) and 'Cancel' (red).

2.3) Check the VMS connector information

OpenCCTV x +

localhost:3000/vmses/18

Successfully connected to the VMS.

Home / VMSSes / VMS

VMS connector information

Name	Description
Milestone VMS Connector	Milestone connector

Verified

VMS information

Name	Server name	Server port	Description	Username	Password
Milestone VMS	192.41.170.168	80	Milestone VMS	lakindu	*****

Verified

All registered Cameras

+ Add new Camera manually

Name	Camera ID	Description
Camera VGL	8725B441-2DED-4D87-BA9E-B2E046D0ADB6	Show Streams Edit Delete

As shown in the figure above, after user registered a new VMS, the system will show the result of VMS connection and all registered cameras on the VMS. In this example, we connect only one IP camera to the Milestone VMS. Therefore, there is only one camera in the registered cameras list. User can click at "Show streams" to view the image from the camera.

OpenCCTV x +

localhost:3000/vmses/18/cameras/13

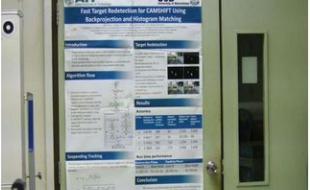
VMS information

Name	Server name	Server port	Description	Username	Password
Milestone VMS	192.41.170.168	80	Milestone VMS	lakindu	*****

Verified

Camera information

Name	Camera ID	Description
Camera VGL	8725B441-2DED-4D87-BA9E-B2E046D0ADB6	Verified



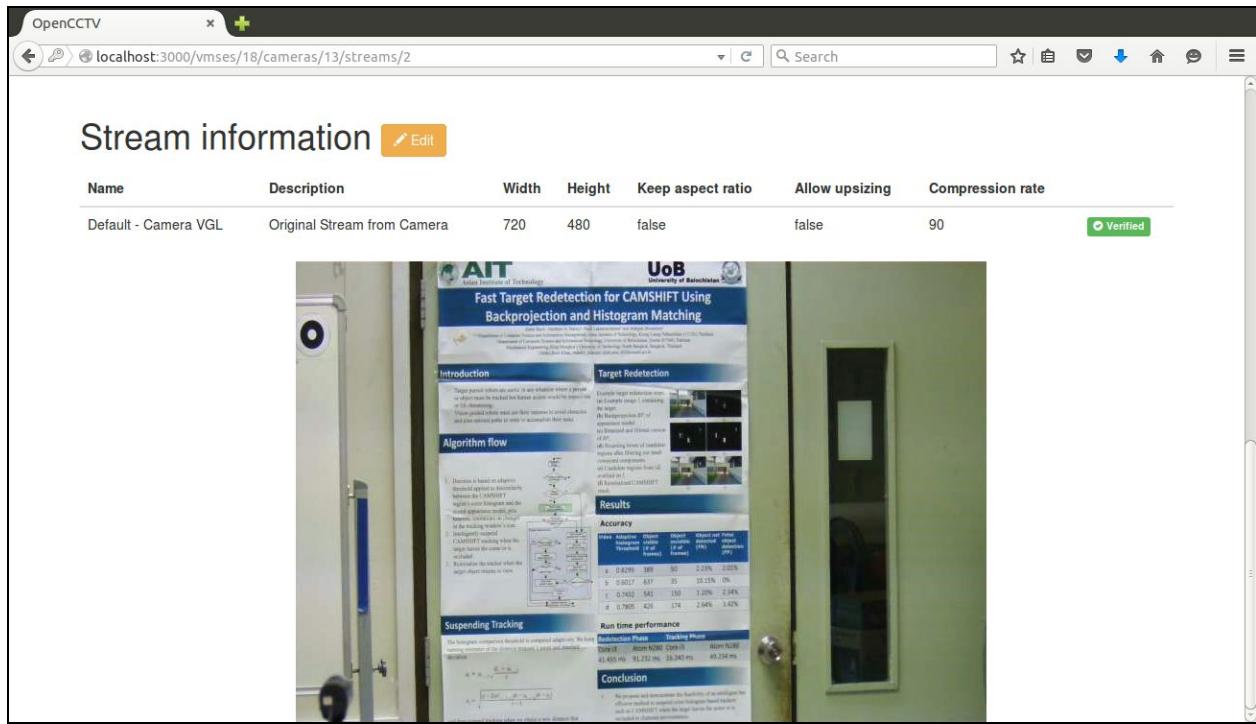
All registered Streams

+ Create new Stream

Name	Description	Width	Height	Keep aspect ratio	Allow upsizing	Compression rate
Default - Camera VGL	Original Stream from Camera	720	480	false	false	90

Show Edit Delete

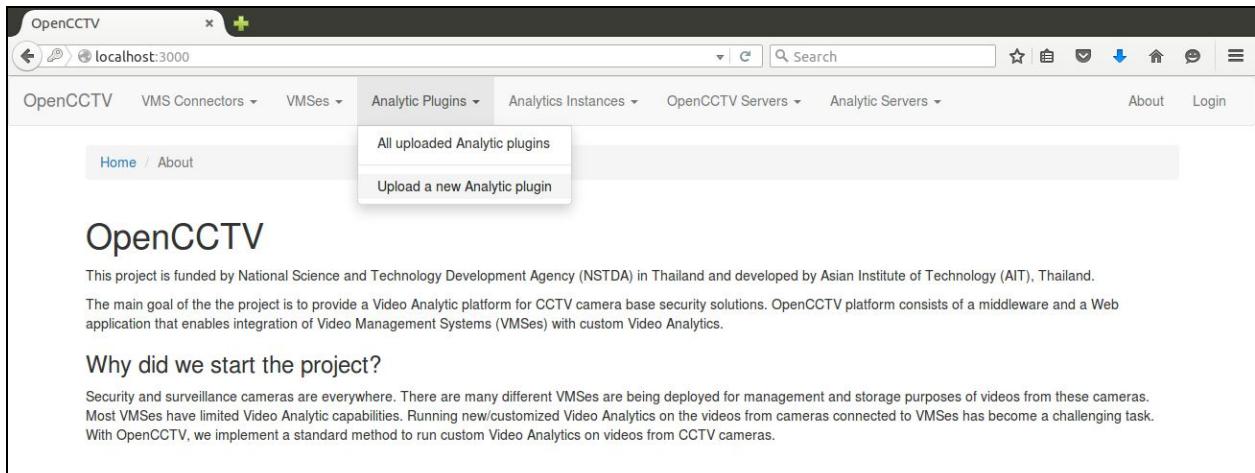
If the video stream is accessible from OpenCCTV, when user clicks at “Show” button in “All registered Streams”, user should see a single frame from the video stream as shown in figure below.



If you can see a single frame in stream information, you can proceed to next step.

3) Upload Analytic plugin

3.1) Click at Analytic Plugins->Upload a new Analytic plugin



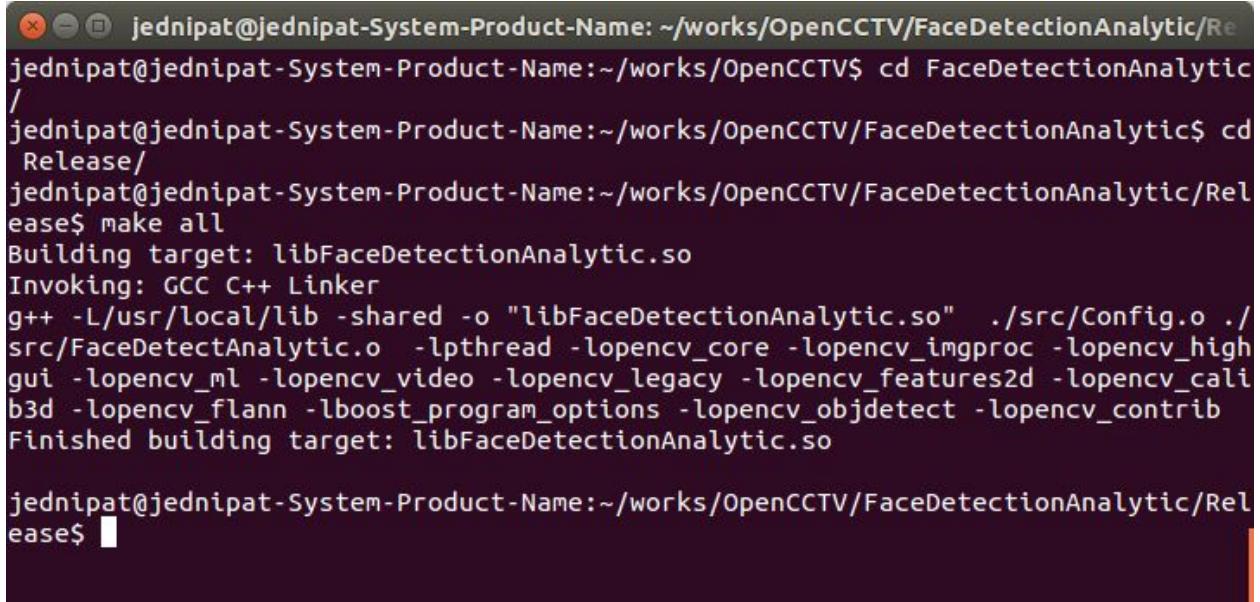
The screenshot shows the OpenCCTV web application running on localhost:3000. The main navigation bar includes links for OpenCCTV, VMS Connectors, VMsEs, Analytic Plugins, Analytics Instances, OpenCCTV Servers, Analytic Servers, About, and Login. Below the navigation, there's a breadcrumb trail: Home / About. A secondary navigation bar within the content area has two items: 'All uploaded Analytic plugins' (which is highlighted) and 'Upload a new Analytic plugin'. The main content area is titled 'OpenCCTV' and contains a brief project overview: 'This project is funded by National Science and Technology Development Agency (NSTDA) in Thailand and developed by Asian Institute of Technology (AIT), Thailand. The main goal of the the project is to provide a Video Analytic platform for CCTV camera base security solutions. OpenCCTV platform consists of a middleware and a Web application that enables integration of Video Management Systems (VMsEs) with custom Video Analytics.' Below this, a section titled 'Why did we start the project?' explains the motivation: 'Security and surveillance cameras are everywhere. There are many different VMsEs are being deployed for management and storage purposes of videos from these cameras. Most VMsEs have limited Video Analytic capabilities. Running new/customized Video Analytics on the videos from cameras connected to VMsEs has become a challenging task. With OpenCCTV, we implement a standard method to run custom Video Analytics on videos from CCTV cameras.'

3.2) Compile and upload analytic plugin

3.2.1) Compile analytic plugin

In this example, we will use a face detection analytic as an example. User has to go to a folder which is cloned from OpenCCTV Github repository and follow the steps below:

```
cd path_to_your_face_detection_analytic_plugin  
cd Release  
make all
```



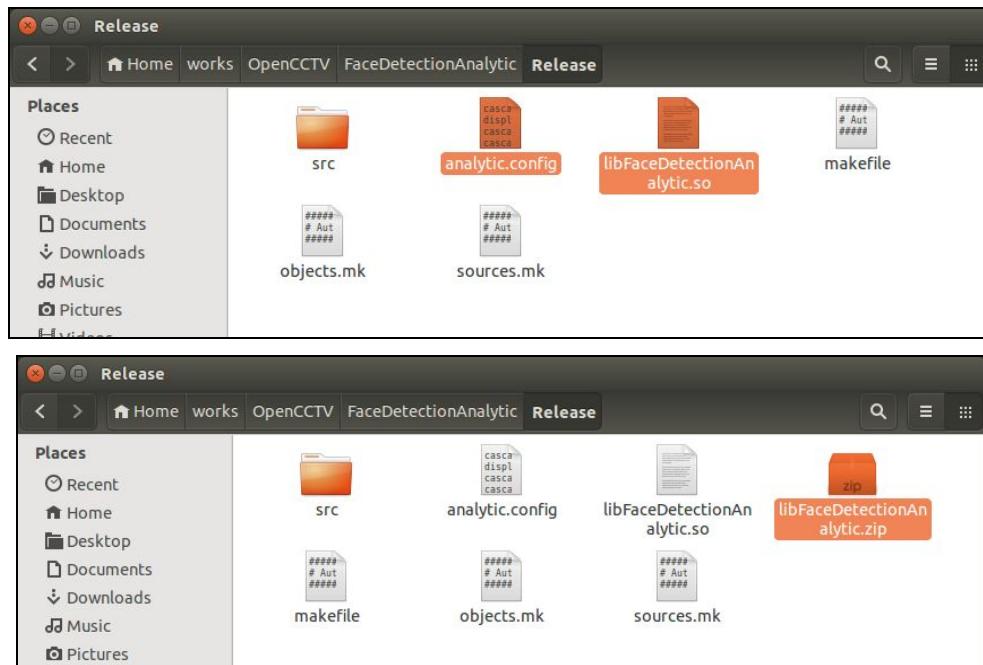
```
jednipat@jednipat-System-Product-Name: ~/works/OpenCCTV/FaceDetectionAnalytic/Re  
jednipat@jednipat-System-Product-Name:~/works/OpenCCTV$ cd FaceDetectionAnalytic/  
jednipat@jednipat-System-Product-Name:~/works/OpenCCTV/FaceDetectionAnalytic$ cd  
Release/  
jednipat@jednipat-System-Product-Name:~/works/OpenCCTV/FaceDetectionAnalytic/Rel  
ease$ make all  
Building target: libFaceDetectionAnalytic.so  
Invoking: GCC C++ Linker  
g++ -L/usr/local/lib -shared -o "libFaceDetectionAnalytic.so" ./src/Config.o ./  
src/FaceDetectAnalytic.o -lpthread -lopencv_core -lopencv_imgproc -lopencv_high  
gui -lopencv_ml -lopencv_video -lopencv_legacy -lopencv_features2d -lopencv_cali  
b3d -lopencv_flann -lboost_program_options -lopencv_objdetect -lopencv_contrib  
Finished building target: libFaceDetectionAnalytic.so  
  
jednipat@jednipat-System-Product-Name:~/works/OpenCCTV/FaceDetectionAnalytic/Rel  
ease$ █
```

After compiling face detection analytic plugin, you will get libFaceDetectionAnalytic.so.

```
jednipat@jednipat-System-Product-Name: ~/works/OpenCCTV/FaceDetectionAnalytic/Release$ ls
analytic.config      makefile      sources.mk
libFaceDetectionAnalytic.so  objects.mk  src
jednipat@jednipat-System-Product-Name:~/works/OpenCCTV/FaceDetectionAnalytic/Release$
```

3.2.2) Create a zip file of libFaceDetectionAnalytic.so and upload to server

To create a zip file for face detection analytic, we have to include analytic.config into the zip file too. This analytic.config contains configuration data of the face detection. Therefore, we have to select analytic.config and libFaceDetectionAnalytic.so for creating a zip file.



The libFaceDetectionAnalytic.zip will be used for uploading to the server.

OpenCCTV

localhost:3000/analytics/new

OpenCCTV VMS Connectors VMSSes Analytic Plugins Analytics Instances

Home / Analytic plugins / New

Upload a new analytic plugin

Name*

Description

Analytic plugin archive zip file*

libFaceDetectionAnalytic.zip

3.3) Check the status of analytic plugin

OpenCCTV

localhost:3000/analytics/3

OpenCCTV VMS Connectors VMSSes Analytic Plugins Analytics Instances OpenCCTV Servers Analytic Servers About Login

Uploaded the new analytic and verified.

Home / Analytic plugins / Analytic plugin

Analytic plugin information

Name: Face detection

Description: A simple face detection plugin for OpenCCTV

Verification report: **Verified**

```
All the files
zip
analytic.config
libFaceDetectionAnalytic.so

Shared Library found
libFaceDetectionAnalytic.so

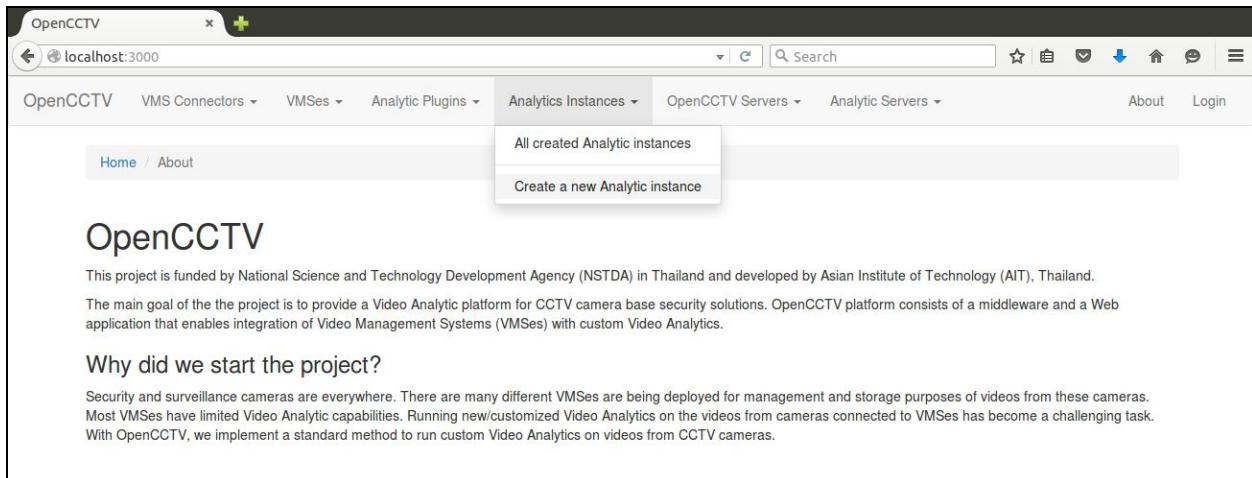
Analytic plugin verified!
```

If the verification report status is “verified”, your analytic plugin is ready to use. User can proceed to next step.

4) Create a new analytic instance

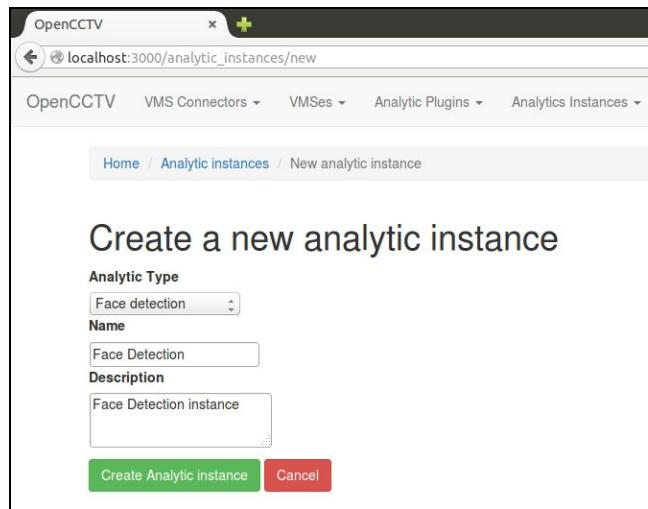
In the previous steps, we uploaded a Milestone VMS connector and a simple face detection analytic plugin. Now, we will create a new analytic instance.

4.1) Click at Analytics Instances->Create a new Analytic instance



The screenshot shows the OpenCCTV web application interface. The title bar says "OpenCCTV" and the address bar shows "localhost:3000". The navigation menu includes "OpenCCTV", "VMS Connectors", "VMSSes", "Analytic Plugins", "Analytics Instances", "OpenCCTV Servers", "Analytic Servers", "About", and "Login". Below the menu, there are two buttons: "All created Analytic instances" and "Create a new Analytic instance". The main content area has a heading "OpenCCTV" and a paragraph about the project being funded by NSTDA and developed by AIT. It also mentions the main goal of providing a Video Analytic platform for CCTV camera base security solutions. A section titled "Why did we start the project?" explains that security cameras are everywhere and running custom video analytics on them is challenging. The "Create a new Analytic instance" button is highlighted.

4.2) Select analytic plugin and create an analytic instance



The screenshot shows a "Create a new analytic instance" form. The title bar says "OpenCCTV" and the address bar shows "localhost:3000/analytic_instances/new". The navigation menu is identical to the previous screenshot. The main content area has a heading "Create a new analytic instance". It contains fields for "Analytic Type" (set to "Face detection"), "Name" (set to "Face Detection"), and "Description" (set to "Face Detection instance"). At the bottom are two buttons: "Create Analytic instance" (green) and "Cancel" (red).

4.3) Add input stream for a new analytic instance by clicking at "Add Input Stream"

Analytic instance information

Analytic ID: 3

Name: Face Detection

Analytic Type: Face detection

Description: Face Detection instance

+ Add Input Stream

Currently no input streams are registered with the analytic instance.

User has to select analytic input stream and input stream, then click at “Create Analytic instance stream” button.

New analytic instance input stream

Analytic Input Stream

default

Stream

Default - Camera VGL

Create Analytic instance stream Cancel

If you see the figure below, it means that we can create a new analytic instance and bind it to the specified video stream already. You can proceed to next step.

Analytic Instance Input Streams			
#	Analytic input stream	Stream	
1	default	Default - Camera VGL	Show Edit Destroy

5) Create OpenCCTV Server

5.1) Click at OpenCCTV Servers->OpenCCTV server Connection

The screenshot shows the OpenCCTV web application running on localhost:3000. The main navigation bar includes links for OpenCCTV, VMS Connectors, VMSSes, Analytic Plugins, Analytics Instances, OpenCCTV Servers (which is currently selected), and Analytic Servers. Below the navigation is a breadcrumb trail: Home / About. A prominent button labeled "OpenCCTV server Connection" is highlighted. The main content area features a heading "OpenCCTV" and a paragraph about the project's funding by NSTDA and development by AIT, Thailand. It also describes the main goal of providing a Video Analytic platform for CCTV cameras. A section titled "Why did we start the project?" explains the challenges of running custom video analytics on VMSSes. A note at the bottom states that security cameras are everywhere and discusses the limitations of existing solutions.

5.2) Click at “Register a Connection to OpenCCTVServer”

The screenshot shows the "OpenCCTV Server" page within the OpenCCTV application. The URL in the address bar is localhost:3000/open_cctv_servers. The navigation bar includes OpenCCTV, VMS Connectors, VMSSes, Analytic Plugins, Analytics Instances, OpenCCTV Servers, and Analytic Servers. The breadcrumb trail shows Home / OpenCCTV Server. A green button labeled "+Register a Connection to OpenCCTV Server" is visible. A yellow message box at the bottom states "Currently no OpenCCTV Server is registered with the system."

5.3) Fill the information about connection

User can use default value for creating this server connection, and then click at “Save Connection Details”.

The screenshot shows the "New OpenCCTV Server connection" form. The URL in the address bar is localhost:3000/open_cctv_servers/new. The form fields are as follows:

- Name:** OpenCCTV Server
- Host:** 127.0.0.1
- Port:** 4444

At the bottom of the form are two buttons: "Save Connection Details" (green) and "Cancel" (red).

5.4) Start OpenCCTV server by clicking at “Start Server”.

The screenshot shows a web-based interface for managing an OpenCCTV server. At the top, there's a navigation bar with links for 'OpenCCTV', 'VMS Connectors', 'VMSSes', 'Analytic Plugins', 'Analytics Instances', and 'Op'. Below the navigation is a breadcrumb trail: 'Home / OpenCCTV Server'. The main content area is titled 'OpenCCTV Server Information'. It displays the following connection details:

- Connection Name:** OpenCCTV Server
- Host:** 127.0.0.1
- Port:** 4444
- Status:** Stopped
- PID:** 0

Below these details are two buttons: a yellow 'Edit' button and a red 'Delete' button. At the bottom of the main content area is a green 'Start Server' button.

Once user clicks on a “Start Server” button, user should see one window showing a video stream from camera. This video stream window is displayed from our face detection analytic. In a real deployment, we can disable this video stream window. This video stream is fetched from Milestone VMS and then our analytic performs face detection on this video stream. The result (output) is written to the database. User may need another web application for displaying the output.

This screenshot shows the same 'OpenCCTV Server Information' page as above, but with a video stream overlay. The video stream is a screenshot of a tracking algorithm's results, specifically 'Fast Target Redetection for CAMSHIFT Using Backprojection and Histogram Matching'. The overlay includes a camera feed of a hallway, a histogram matching interface, and a table of tracking accuracy results. The tracking algorithm results table is as follows:

Frame	Algorithm	Target	Object	Object Area	Target Area	Overlap (%)	Accuracy (%)
A	3.0239	389	30	2.03%	2.03%		
B	3.0237	837	35	20.55%	97%		
C	3.0238	541	350	1.25%	2.34%		
D	3.0238	426	235	2.64%	3.42%		

At the bottom of the main content area, there are two buttons: a red 'Stop Server' button and a green 'Check Server Status' button.