**Installation Guide**Native Open edX Installation on Ubuntu 16.04

Detailed explanation of the script - native.sh

|  |  |
| --- | --- |
|  | if [[ ! $OPENEDX\_RELEASE ]]; then |
|  | echo "You must define OPENEDX\_RELEASE" |
|  | exit |
|  | fi |

- Checks whether the openedx release tag is defined or not.

|  |
| --- |
| if [[ `lsb\_release -rs` != "16.04" ]]; then |
| echo "This script is only known to work on Ubuntu 16.04, exiting..." |
| exit |
| fi |

- Check the version of ubuntu, it must be 16.04 for installing this Open edX platform.

|  |  |
| --- | --- |
|  | if [[ ! -f config.yml ]]; then |
|  | echo 'You must create a config.yml file specifying the hostnames (and if' |
|  | echo 'needed, ports) of your LMS and Studio hosts.' |
|  | echo 'For example:' |
|  | echo ' EDXAPP\_LMS\_BASE: "11.22.33.44"' |
|  | echo ' EDXAPP\_CMS\_BASE: "11.22.33.44:18010"' |
|  | exit |
|  | fi |

-YAML (YAML Ain't Markup Language).

It is a data serialization language that matches user’s expectations about data.

In config.yml file we will specify host names of the LMS and CMS

LMS (Learning Management System) - It is an application for managing educational courses and learning programs.

CMS (Configuration Management System) - It is a collection of tools which is used for managing, updating and analyzing all configuration items.

|  |  |
| --- | --- |
|  | grep -Fq EDXAPP\_LMS\_BASE config.yml |
|  | GREP\_LMS=$? |
|  |  |
|  | grep -Fq EDXAPP\_CMS\_BASE config.yml |
|  | GREP\_CMS=$? |

- Next, the script uses grep to search for the hostnames of the Learning Management System (LMS) and the Content Management System (CMS). After every grep command, the script assigns the corresponding exit status to the variables in the succeeding command. (Hostname for LMS is assigned to GREP\_LMS, for example).

|  |  |
| --- | --- |
|  | if [[ $GREP\_LMS == 1 ]] || [[ $GREP\_CMS == 1 ]]; then |
|  | echo 'Your config.yml file must specify the hostnames (and if' |
|  | echo 'needed, ports) of your LMS and Studio hosts.' |
|  | echo 'For example:' |
|  | echo ' EDXAPP\_LMS\_BASE: "11.22.33.44"' |
|  | echo ' EDXAPP\_CMS\_BASE: "11.22.33.44:18010"' |
|  | exit |
|  | fi |

- In case grep was unable to match the required patterns with the config.yml input file, the values assigned to GREP\_LMS and GREP\_CMS would be 1 (since, the exit status of grep would be 1). Subsequently, a message informing the same to the user is displayed on the screen, and the script ends.

grep -Fq edx. config.yml

GREP\_BAD\_DOMAIN=$?

if [[ $GREP\_BAD\_DOMAIN == 0 ]]; then

echo '\*\*\* NOTE: Open edX and edX are registered trademarks.'

echo 'You may not use "openedx." or "edx." as subdomains when naming your site.'

echo 'For more details, see the edX Trademark Policy: https://edx.org/trademarks'

echo ''

echo 'Here are some examples of unacceptable domain names:'

echo ' openedx.yourdomain.org'

echo ' edx.yourdomain.org'

echo ' openedxyourdomain.org'

echo ' yourdomain-edx.com'

echo ''

echo 'Please choose different domain names.'

exit

fi

- The official Open edX wiki asks us not to use “edx.” subdomain. If the config.yml file doesn’t follow this, the script ends.

mkdir -p logs

log\_file=logs/install-$(date +%Y%m%d-%H%M%S).log

exec > >(tee $log\_file) 2>&1

echo "Capturing output to $log\_file"

echo "Installation started at $(date '+%Y-%m-%d %H:%M:%S')"

- Here a parent directory named logs is created and the output is captured to a logfile.

tee--read from standard input and write to standard output and files

2>&1 taking both standard output and standard error into log file.

function finish {

echo "Installation finished at $(date '+%Y-%m-%d %H:%M:%S')"

}

trap finish EXIT

- A function named finish is declared. This function is trapped when the script exits (using trap finish EXIT) and a message is displayed notifying that installation is finished along with the date and time.

sudo apt-get install -y python-software-properties

sudo add-apt-repository -y ppa:ubuntu-toolchain-r/test

- Software-properties-common provides some useful scripts to add or remove PPAs (used in our case). Hence, the script executes apt-get install -y software-properties-common command. Python-software-properties is an alternative for software-properties-common and is used for ubuntu releases older than Xenial Xerus (12.04). Therefore, the script executes the apt-get install -y python-software-properties command (since we’ve used Ubuntu Xenial Xerus). Now the script uses the sudo add-apt-repository -y ppa:ubuntu-toolchain-r/test command to add the Ubuntu Toolchain PPA. This will be used later to install gcc/g++.

sudo apt-get install -y build-essential software-properties-common curl git-core libxml2-dev libxslt1-dev python-pip libmysqlclient-dev python-apt python-dev libxmlsec1-dev libfreetype6-dev swig gcc g++

sudo pip install --upgrade pip==9.0.3

sudo pip install --upgrade setuptools==39.0.1

sudo -H pip install --upgrade virtualenv==15.2.0

- The script installs the system pre-requisites (build-essential is installed, for example, which is a list of all packages which are required to compile a debian package). Along with that, the script installs setuptools (used to package python projects) and virtualenv (a tool to create isolated python environments) with the pre-defined versions.

VERSION\_VARS=(

edx\_platform\_version

certs\_version

forum\_version

xqueue\_version

configuration\_version

demo\_version

NOTIFIER\_VERSION

INSIGHTS\_VERSION

ANALYTICS\_API\_VERSION

ECOMMERCE\_VERSION

ECOMMERCE\_WORKER\_VERSION

DISCOVERY\_VERSION

THEMES\_VERSION

)

for var in ${VERSION\_VARS[@]}; do

# Each variable can be overridden by a similarly-named environment variable,

# or OPENEDX\_RELEASE, if provided.

ENV\_VAR=$(echo $var | tr '[:lower:]' '[:upper:]')

eval override=\${$ENV\_VAR-\$OPENEDX\_RELEASE}

if [ -n "$override" ]; then

EXTRA\_VARS="-e $var=$override $EXTRA\_VARS"

fi

done

--There is a for loop looping through all the elements in an array VERSION\_VARS.

Firstly, the script translates the name of the variables in the VERSION\_VARS array to uppercase and hence, overrides the corresponding variables with the associated environment variables. Next, the script uses EXTRA\_VARS to maintain a list of variables which are to be passed to the ansible playbook later on.

[The command :

eval override=\${$ENV\_VAR-\$OPENEDX\_RELEASE}

stores the value of ENV\_VAR into override if ENV\_VAR is already set to some value. Otherwise it will store the value of OPENEDX\_RELEASE into override.

The next step checks if the value of override is not empty. As override contains a non-empty string, the condition is satisfied and a variable EXTRA\_VARS is set to “-e $var=$override $EXTRA\_VARS”.]

f [[ -f my-passwords.yml ]]; then

EXTRA\_VARS="-e@$(pwd)/my-passwords.yml $EXTRA\_VARS"

fi

EXTRA\_VARS="-e@$(pwd)/config.yml $EXTRA\_VARS"

- If the file my-passwords.yml is present then its path is prepended to the original value of EXTRA\_VARS and is stored in EXTRA\_VARS. Similarly the path for config.yml is also prepended to the value of EXTRA\_VARS.

cd /var/tmp

git clone https://github.com/edx/configuration

cd configuration

git checkout $CONFIGURATION\_VERSION

git pull

cd /var/tmp/configuration

sudo -H pip install -r requirements.txt

- The script clones the configuration repository to the /var/tmp directory. Later on, it uses the git checkout command to switch to the desired local branch. The script then install the required python packages with their corresponding versions listed in the requirements.txt file.

To trace all the roles, go to edx/configuration/playbooks/roles directory. Within each role, look for the dependencies inside the meta folder’s main.yml file.

Submitted by:

Mridul Mahajan, Harshit Mogalapalli and Soumyakant Mohakul