Open edX (Dogwood on AWS) Documentation

Release 1

Sachin Sable

June 25, 2016

Contents

CHAPTER 1

General Information

1.1 Implementation on Amazon Web Services

Author Sachin Sable

Contact ssable@umail.iu.edu

organization Data Science Program

date 24 May 2016

status This is a "work in progress"

Dedication For developers who want to install Open edX platform.

abstract This document is a demonstration of the how to install and configure open edX platform (dogwood release) on amazon web services.

1.2 About Open edx

1.2.1 edX

EdX is a nonprofit online initiative created by founding partners Harvard and MIT and composed of dozens of leading global institutions, the xConsortium. EdX offers interactive online courses and MOOCs from the world's best universities and institutions.

1.2.2 Open edX

The Open edX platform is a free–and open source–course management system (CMS) that was originally developed by edX. The Open edX platform is used all over the world to host Massive Open Online Courses (MOOCs) as well as smaller classes and training modules.

Visit Open edX website

CHAPTER 2

Installation Process

2.1 Virtual Machine Instance

Launch virtual machine instance in Amazon EC2

Follow steps

1. Click on ‘Launch Instance’
2. Chose ‘Community AMIs’
3. Search ‘ubuntu 12.04 precise server’
4. Select the AMI
5. Configure the machine according to following attributes in the table

Attribute	Value
OS	Ubuntu 12.04 precise server
Architecure	64 bit
Root device type	EBS
Architecure	64 bit
Virtualization type	paravirtual
Instance Type	Compute optimized (c3.large)
Instance Details	Use default settings
Storage	General Purpose SSD (GP2)
Size	50 GB

Note:

1. Do not forget to create security group allowing traffic from required ports such as SSH on port 22, LMS and CMS traffic on port 80 and 18010.
2. These are the suggested attributes, you can change the attributes except the operating system - ‘ubuntu 12.04’

2.2 Prerequisite

Update Ubuntu package sources

```
sudo apt-get update -y
sudo apt-get upgrade -y
sudo reboot
```

2.3 Automated installation

Follow following steps for automated installation

```
# Set environment variable to open edX release
export OPENEDX_RELEASE=named-release/dogwood.3
wget https://raw.githubusercontent.com/edx/configuration/master/util/install/ansible-bootstrap.sh -O
# Activate virtual environment
source /edx/app/edx_ansible/venvs/edx_ansible/bin/activate
wget https://raw.githubusercontent.com/edx/configuration/$OPENEDX_RELEASE/util/install/sandbox.sh -O
```

CHAPTER 3

Basic information and configuration

3.1 Basic Configuration

You should have running open edX system after performing automated installation. Next step is to configure open edX platform.

3.1.1 Domain Name

IP address of the instance running open edX can be mapped with domain name. It is good idea to have domain name as it will be easy to use. Also, it will be useful while integrating with other services, as many will require to configure DNS settings, such as DKIM signature.

3.1.2 Accessing open edX

lms is accessed using port 80, and cms is accessed using port 18010.

LMS	domain_name:80
LMS admin	domain_name:80/admin
CMS	domain_name:18010
CMS admin	domain_name:18010/admin

3.1.3 Manage Users

Default users and passwords:

Username	Password
honor	edx
audit	edx
verified	edx
staff	edx

Initially no user has superuser access, thus we need to go through command line and give users superuser access.

We can change the user permissions by manually modifiying 'auth_user' table inside 'edxapp' database. It has two important parameters. First is 'is_staff' indicates if user can access admin console or not. Second is 'is_superuser' it indicates if user can modify the setting using admin console.

Let's give superuser access to user - staff.

```
$ sudo -u www-data /edx/app/edxapp/venvs/edxapp/bin/python /edx/app/edxapp/edx-platform/manage.py lms
mysql> use edxapp;
mysql> update auth_user set is_superuser=1 where username="staff";
```

for security purposes

Note: To improve security, after giving superuser permissions to 'staff' user, login into admin console and change default password of 'staff'.

3.2 Installing Theme (Stanford theme)

1. Modify /edx/app/edx_ansible/server-vars.yml to contain following variables set to given values.

```
edxapp_use_custom_theme: true
edxapp_theme_name: 'stanford'
edxapp_theme_source_repo: 'git://github.com/Stanford-Online/edx-theme.git'
edxapp_theme_version: 'HEAD'
```

2. Now run the provisioning script

```
$ sudo /edx/bin/update edx-platform named-release/dogwood.3
```

3.3 References

1. Standford theme installation: http://www.dangtrinh.com/2014/03/edx-platform-using-standford-them-for.html

3.4 Fixing 'view course in studio' and 'view live' buttons

'View course in studio' button will take you from learning platform to CMS, so that you can modify the course. 'View live' button will take you from CMS to LMS platform.

Thus, these two buttons are used to navigate between LMS and CMS platform for a course.

Hence, LMS needs to know the DNS for CMS, to configure it, in file /edx/app/edxapp/lms.env.json change the value of "CMS_BASE" to DNS of CMS. Similarly, for CMS to know DNS of LMS, in file /edx/app/edxapp/cms.env.json change the value of "LMS_BASE" to DNS of LMS.

3.5 Deleting courses

To delete the course you need to know id of the course. use the first command to list ids of available courses and next two commands to delete the course.

```
sudo -u www-data /edx/bin/python.edxapp /edx/bin/manage.edxapp lms dump_course_ids --settings aws
sudo -u edxapp /edx/bin/python.edxapp ./manage.py cms --settings=aws delete_orphans <course id> --cor
sudo -u edxapp /edx/bin/python.edxapp ./manage.py cms --settings=aws delete_course <course id>
```

CHAPTER 4

Amazon AWS Integration

Add why to use Amazon SES instead of deploying own emaling server.

4.1 Basic Amazon SES Integrating

Before integrating SES we need to install postfix and configure it to send emails. Postfix should be configured properly and it should be able to send emails. After this step we need to verify email addresses from which emails will be sent. Emails can be varified from Amazon web service -> SES -> Email Addresses (Identity Management).

Resources for installing postfix and configuring Amazon SES

1. Postfix installation guide from ubuntu community
2. Integrating Amazon SES with Postfix

4.2 DKIM settings

DomainKeys Identified Mail (DKIM) lets an organization take responsibility for a message that is in transit. The organization is a handler of the message, either as its originator or as an intermediary. Their reputation is the basis for evaluating whether to trust the message for further handling, such as delivery. Technically DKIM provides a method for validating a domain name identity that is associated with a message through cryptographic authentication. DKIM attaches a new domain name identifier to a message and uses cryptographic techniques to validate authorization for its presence. The identifier is independent of any other identifier in the message, such in the author's From: field.

It is strongly advised to send DKIM signed emails. As it authenticated the sender and guarantees that message was not modified while in transit. Thus, it's spam score will be reduced and it will have higher chances of ending in inbox rather than spam folder.

Please follow Easy DKIM in Amazon SES to implement DKIM signature in AWS.

4.2.1 references

1. DKIM.org

4.3 Configuring verification link

After a student registers on edX platform, student will need to verify the email address he/she provided. edX will send a verification link to email id provided by student. By default this verfication link will be addressed to localhost. We need to replace localhost by DNS of edX platform. It requires to modify environmental variable in /edx/app/edxapp/lms.env.json, change variable 'SITE_NAME' to DNS of edX platform. Now whenever student registers, he/she will get verification link pointing ot edX platform and not localhost.

CHAPTER 5

Shibboleth configuration

First follow the steps outlined here - http://edx.readthedocs.io/projects/edx-installing-configuring-and-running/en/latest/configuration/tpa/

After 4.16.3 section, go to /edx/app/edxapp/lms.env.json and add the following lines to the end just before the closing ‘]’ -

```
"THIRD_PARTY_AUTH_BACKENDS": [
    "third_party_auth.saml.SAMLAuthBackend"
]
```

Next, in Django admin for LMS, under third_party_auth section, in Provider Configuration (SAML IdP), add SAML Provider configuration and fill it with the data available in IU KB here - https://kb.iu.edu/d/bdgs

The metadata column in the above page should me marked done after a couple of minutes.

Only for integrating with Indiana University:

1. follow the steps outlined in the IU KB - https://kb.iu.edu/d/bdag
2. After this setup, you should see the login with IU button.

CHAPTER 6

Amazon snapshot bug

While working on AWS EC2 it was observed that if you create instance using snapshot, it causes rabbitmqctl to malfunction and celery workers cannot connect to amqp.

If you check log file /edx/var/log/cms/edx.log, you will see following error

```
error: [Errno 104] Connection reset by peer
```

if you check error log for workers (available in /edx/var/log/supervisor/) you will find similar

```
[2015-11-03 14:07:56,018: ERROR/MainProcess] consumer: Cannot connect to amqp://celery:**@127.0.0.1:5
Trying again in 4.00 seconds...

[2015-11-03 14:08:00,036: ERROR/MainProcess] consumer: Cannot connect to amqp://celery:**@127.0.0.1:5
Trying again in 6.00 seconds...
```

It means that all celery workers are unable to connect to amqp broker.

We need to have celery user is created. We can check celery user is defined in Open edX config *.auth.json. Below is default value.

```
"CELERY_BROKER_PASSWORD": "celery",
"CELERY_BROKER_USER": "celery",
```

Solve the problem using following steps

```
# List rabbitnq users
$ sudo rabbitmqctl list_users
Listing users ...
guest    [administrator]
# create user
$ sudo rabbitmqctl add_user celery celery
Creating user "celery" ...
# set permissions for celery user
$ sudo rabbitmqctl set_permissions celery ".*" ".*" ".*"
Setting permissions for user "celery" in vhost "/" ...
# restart rabbitmq-server
$ sudo service rabbitmq-server restart
 * Restarting message broker rabbitmq-server     [OK]
# we need to restart the apps
$ sudo /edx/bin/supervisorctl restart all
```

6.1 reference

1. blogs.infinitesquares.net

from docutils import nodes from docutils.parsers.rst import directives

CODE = """<object type="application/x-shockwave-flash"

width="%(width)s" height="%(height)s" class="youtube-embed" data="http://www.youtube.com/v/%(yid)s">

<param name="movie" value="http://www.youtube.com/v/%(yid)s"></param> <param name="wmode" value="transparent"></param>%(extra)s

</object> """

PARAM = """n <param name="%s" value="%s"></param>""""

def youtube(name, args, options, content, lineno,

contentOffset, blockText, state, stateMachine):

""" Restructured text extension for inserting youtube embedded videos """ if len(content) == 0:

return

string_vars = { 'yid': content[0], 'width': 425, 'height': 344, 'extra': '' }

extra_args = content[1:] # Because content[0] is ID extra_args = [ea.strip().split("=") for ea in extra_args] # key=value extra_args = [ea for ea in extra_args if len(ea) == 2] # drop bad lines extra_args = dict(extra_args) if 'width' in extra_args:

string_vars['width'] = extra_args.pop('width')

if 'height' in extra_args: string_vars['height'] = extra_args.pop('height')

if extra_args: params = [PARAM % (key, extra_args[key]) for key in extra_args] string_vars['extra'] = "".join(params)

return [nodes.raw('', CODE % (string_vars), format='html')]

youtube.content = True directives.register_directive('youtube', youtube)

CHAPTER 7

Documentation