Eros-Magento

CREDENTIALS FOR EROS BASTION SERVER :

PUBLIC IP : 15.184.164.205

PRIVATE IP : 10.0.1.39

SSH KEY TO LOG IN ON BASTION SERVER :

```
$ssh -i "ErosProdBahrain.pem" ubuntu@15.184.164.205
```

CREDENTIALS FOR EROS PRIVATE-PRODUCTION SERVER :

PRIVATE IP : 10.0.2.187

SSH KEY TO LOG IN ON BASTION SERVER :

```
$sudo ssh -i "ErosProdBahrain.pem" ubuntu@10.0.2.57
```

CONFIGURATION ON PROD SERVER :

Registration On Magento Online Portal & generate public/secrete keys in marketplace, As log in creadentials :

CONFIGURING PREREQUISITES FOR MAGENTO :

1>> LAMP server installation :

```
$ sudo apt-get install nginx -y
$ sudo apt-get install mysql -y
$ sudo apt-get install php -y
```

2>> INSTALL COMPOSER :

Composer is a popular dependency management tool for PHP, created mainly to facilitate installation and updates for project dependencies. It will check which other packages a specific project depends on and install them for you, using the appropriate versions according to the project requirements.

(reference : https://www.digitalocean.com/community/tutorials/how-to-install-composer-on-ubuntu-20-04-quickstart)

```
$sudo apt update
$sudo apt install php-cli unzip
$cd ~
```

```
$curl -sS https://getcomposer.org/installer -o /tmp/composer-setup.php
$HASH=`curl -sS https://composer.github.io/installer.sig`
$php -r "if (hash_file('SHA384', '/tmp/composer-setup.php') === '$HASH') { echo
'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); }
echo PHP_EOL;"
$sudo php /tmp/composer-setup.php --install-dir=/usr/local/bin --filename=composer
$composer
```

3>> Elastic Search installation :

Elasticsearch is an open-source platform for full-text search and analytics engines. It allows you to store, search, and analyze a large amount of data in real-time. It is a popular search engine designed for applications that have complex search requirements

video : https://www.youtube.com/watch?v=hVgrXi_9L6Y
https://www.youtube.com/watch?v=HsKXy8AnJTM

Document : https://www.elastic.co/guide/en/elasticsearch/reference/current/deb.html

ES Commands :

:For heap size -

```
$sudo su -
$cd /etc/elasticsearch/
$sudo vim jvm.options/
```

:For ES running status -

```
$sudo systemctl start elasticsearch
$sudo systemctl stop elasticsearch
$sudo systemctl enable elasticsearch
$curl localhost:9200
```

4>> INSTALLING PHP EXTENSIONS TO ADD REQUIRE EXTENSIONS FOR MAGENTO THROUGH CLI :

By using php extensions, you can avoid having to recreate the same code for numerous applications. To use an extension, you simply connect it to your application’s PHP binary. Because existing extensions don’t cover all the application domains, from time to time, you will need to develop a new extension.

: The OCI8.so file is located in the folder /usr/lib/php/modules with the correct privileges.

```
$sudo apt install php7.4-xml
$sudo apt-get install php7.4.3-soap php7.4.3-bcmath
$sudo apt-get install php-curl
$sudo apt-get install gd
$sudo apt-get install intl
$sudo apt-get install soap
$sudo apt-get install zip
```

5>> PHP OCI-8 CONNECT EXTENTION INSTALLATION:

The OCI8 extension lets you access Oracle Database. Use 'pecl install oci8' to install for PHP 8.1.
- { https://gist.github.com/eSkiSo/781269c79b4dd740e90fcc059c1985ae }

- Need to add following line maually at the end of nano /etc/php/7.4/fpm/php.ini
  - extension = oci8.so
  - systemctl restart php7.4-fpm

INSTALLATION STEPS FOR OCI8 ON UBUNTU 20.04 WITH PHP 7.4

Step 1: Download Oracle Instant Client and SDK from Oracle:

http://www.oracle.com/technetwork/topics/linuxx86-64soft-092277.html

Download files:

Basic Package (ZIP): instantclient-basic-linux.x64-21.1.0.0.0.zip
SDK Package (ZIP): instantclient-sdk-linux.x64-21.1.0.0.0.zip

Step 2: Create new folders for Oracle Instant Client on server

```
$sudo mkdir /usr/lib/oracle
$sudo mkdir /usr/lib/oracle/21.1
$sudo mkdir /usr/lib/oracle/21.1/client64
```

Step 3: Extract files

```
$sudo cp instantclient-basic-linux.x64-21.1.0.0.0.zip /usr/lib/oracle/21.1/client64
$sudo cp instantclient-sdk-linux.x64-21.1.0.0.0.zip /usr/lib/oracle/21.1/client64

$cd /usr/lib/oracle/21.1/client64

$sudo unzip instantclient-basic-linux.x64-21.1.0.0.0.zip
$sudo unzip instantclient-sdk-linux.x64-21.1.0.0.0.zip

$sudo mv instantclient_21_1 lib
```

Step 4: Create symbolic link to the new Instant Client files:

```
$cd /usr/lib/oracle/21.1/client64/lib/
$sudo ln -s libclntsh.so.21.1 libclntsh.so (It may already exist, continue)
$sudo ln -s libocci.so.21.1 libocci.so (It may already exist, continue)
```

Step 5: Edit/Create this file with the path to the lib (for LDCONFIG):

```
$sudo echo /usr/lib/oracle/21.1/client64/lib > /etc/ld.so.conf.d/oracle.conf
```

Step 6: Update Dynamic Linker

```
$ldconfig
```

Step 7 : Install php-dev php-pear build-essential and libaio1

$sudo apt-get install php-dev php-pear build-essential libaio1

Step 8 : Update PECL to install OCI8 from it

$sudo pecl channel-update pecl.php.net

Install OCI8 from PCEL (desired version, default seems not to find it for me, i used php7.)

$sudo pecl install oci8 (php 8)
$sudo pecl install oci8-2.2.0 (php 7.)

During install it will require the path to Instant Client, write this:

instantclient,/usr/lib/oracle/21.1/client64/lib

At the end something like this should appear:

Build process completed successfully
Installing '/usr/lib/php/20190902/oci8.so'
install ok: channel://pecl.php.net/oci8-2.2.0
configuration option "php_ini" is not set to php.ini location
You should add "extension=oci8.so" to php.ini

Step 9 : Load OCI8 into PHP

$sudo echo "extension=oci8.so" >> /etc/php/7.4/cli/php.ini

: Restart Apache

$sudo systemctl restart apache2

Step 10: Add to mods-available

$cd /etc/php/7.4/mods-available/
$sudo nano oci.ini

: *Add this to the file:*

extension = oci8.so

Step 11 : Create sym link to the created ini file

$cd /etc/php/7.4/apache2/conf.d
$sudo ln -s /etc/php/7.4/mods-available/oci.ini 20-oci.ini

: *Restart Apache*
$sudo systemctl restart apache2

Step 12 : (Testing) Check if OCI is loaded in PHP

$php -i | grep oci

: *Should output something like this:*

```
oci8
oci8.connection_class => no value => no value
oci8.default_prefetch => 100 => 100
oci8.events => Off => Off
oci8.max_persistent => -1 => -1
oci8.old_oci_close_semantics => Off => Off
oci8.persistent_timeout => -1 => -1
oci8.ping_interval => 60 => 60
oci8.privileged_connect => Off => Off
oci8.statement_cache_size => 20 => 20
```

6>> CONFIGURATION AND INSTALLATION OF PHPMYADMIN :

REFERENCE:

{video: https://www.youtube.com/watch?v=NboJeJTRzic&t=324s }
{docs : https://www.codewithharry.com/blogpost/install-phpmyadmin-ubuntu }

phpMyAdmin is a free software tool written in PHP, intended to handle the administration of MySQL over the Web. phpMyAdmin supports a wide range of operations on MySQL and MariaDB. Frequently used operations (managing databases, tables, columns, relations, indexes, users, permissions, etc) can be performed via the user interface, while you still have the ability to directly execute any SQL statement.

: To activate database user in phpmyadmib

steps:

> cd /etc/phpmyadmin/

> sudo nano config-db.php

```
<?php
$dbuser='phpmyadmin';
$dbpass='erosnowphpmyadmin';    ( CHANGE PASWWORD HERE )
$basepath='';
$dbname='phpmyadmin';                ( CHANGE DB NAME HERE )
$dbserver='eros-prod-rds-mysql.cuywhj9qtja2.me-south-1.rds.amazonaws.com';  ( CHANGE HOSTNAME HERE < localhost to endpoint > )
$dbport='3306';
$dbtype='mysql';
```

> EXIT

7>> INSTALLATION ON MAGENTO APP

Download & Install magento

Video : (https://www.youtube.com/watch?v=36eWE0K1_0w)

Document :

https://docs.google.com/document/d/1O43-SwvQPmyTNdHQ5ySE1KyohwMHKljDJQEJFPkHcq4/edit

check magento version :

$sudo cd /var/www/erosnow
$sudo bin/magento --version

8>> ATTACHING AWS-RDS DB TO MAGENTO INSTANCE :

Path to configure the datails for attaching the RDS database instance using endpoint.

$cd/var/www/erosdigital/app/etc/env.php

9>> CONFIGURATION AND INSTALLTION OF REDISS ON THE SERVER.

Redis, which stands for Remote Dictionary Server, is a fast, open source, in-memory, key-value data store. Redis is an open-source, highly replicated, performant, non-relational kind of database and caching server. It works by mapping keys to values with a sort of predefined data model. Its benefits include: Mapped key-value-based caching system, almost comparable to memcached.

$apt update
$apt install net-tools
$apt update
$systemctl status redis.service
$apt install redis-server
$systemctl status redis.service
$redis-cli
$sudo nano /etc/redis/redis.conf
$systemctl restart redis.service
$systemctl status redis.service
$redis-cli -h 10.0.2.204 ping

Made changes in sudo nano /etc/redis/redis.conf

- set "protected mode no" from "yes"
- set "supervised systemd" from "no"

Then,

- added IP Address "bind 10.0.2.204" to acces it using host ip

To Start redis-cli ::

-redis-cli -h 10.0.2.204

To check redis.service after every modification:

$systemctl restart redis-service
$systemctl status redis-service

To Stop Service.

$systemctl stop redis-service

10 : VIRTUAL HOST CONFIHURATION FOR EROS .AE

create a vhost for the erosnow and created one "eros" folder /var/html/ and in that create "pub" folder and point that vhost to that pub folder

$cd /etc/nginx/sites-available
$sudo vim eros.ae.conf

Added server name and domain. make chnages accordingly.
$server_name eros.ae www.eros.ae;

NGINIX FILE CONTENT :

```
server {
        listen 80;
#       listen [::]:80 default_server;


        root /var/www/erosdigital/pub;

        # Add index.php to the list if you are using PHP
        index index.php index.html index.htm index.nginx-debian.html;

        server_name eros.ae www.eros.ae;

        location / {
                # try_files $uri $uri/ =404;
                try_files $uri $uri/ /index.php$is_args$args;
        }

      access_log /var/log/nginx/eros.ae_access.log;
      error_log /var/log/nginx/eros.ae_error.log;


        # pass PHP scripts to FastCGI server
        #
        location ~ \.php$ {
                include snippets/fastcgi-php.conf;
        #
        #       # With php-fpm (or other unix sockets):
                fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
        #       # With php-cgi (or other tcp sockets):
        #       fastcgi_pass 127.0.0.1:9000;
#               fastcgi_buffers 16 16k;
                fastcgi_buffers 16 128k;
#       fastcgi_buffer_size 32k;
        fastcgi_buffer_size 256k;
```

```
        fastcgi_read_timeout 1h;
        fastcgi_send_timeout 600;
        fastcgi_connect_timeout 600;

        client_max_body_size 2G;


}

# deny access to .htaccess files, if Apache's document root
# concurs with nginx's one
#
location ~ /\.ht {
        deny all;
}
```