**NOTES**

When you see ":~$" (no quotes), you should type in the linux command on the same line

Example

:~$ sudo apt-get update

means you should type:

sudo apt-get update

----

When you see "=#" (no quotes), you should type in the postgresql command on the same line

Example

=# \list

means you should type:

\list

**01. Create an EC2 Instance:**

- Linux

- Security Group

- Confirm correct IP address(es)

- Make a note of your IP address

- This example uses 52.6.154.11 ... Your IP address will be different

**02. Log into EC2 using Putty [if you're using a Windows computer to connect to your Linux EC2]:**

Session :: Host Name (or IP address)

ubuntu@ec2-52-6-154-11.compute-1.amazonaws.com

Session :: Port

22

Connection >> SSH >> Auth :: Private key file for authentication:

C:\Users\bryan\Downloads\bjones-key-pair-east.ppk

/\* BEG: Delete \*/

My New IP Address:

24.178.251.37/32

My Old IP Address:

24.178.249.218/32

My EC2 Public IP Address:

52.6.154.11

/\* END: Delete \*/

--------

Once you're logged into your Linux instance:

**01. Look for nginx (should not be available)**

:~$ find / -name "nginx\*" 2> /dev/null

***xx. FAILED #1 (Connect to website IP address)***

http://52.6.154.11

**02. Update Ubuntu Packages**

:~$ sudo apt-get update

**03. Install nginx**

:~$ sudo apt-get install nginx

*Key Files:*

/usr/share/nginx/html/index.html (our homepage)

/etc/nginx/sites-available/default

/etc/init.d/nginx (to check the running status of nginx)

***xx. SUCCESS #1 (Connect to website IP address)***

http://52.6.154.11 in your browser (your IP address will be different)

**04. Update the index.html w/ "FullStacker" text (This is your homepage... But we'll change that later)**

:~$ sudo nano /usr/share/nginx/html/index.html

[ Change <h1>Welcome to Nginx!</h1> ... to ... <h1>Welcome, FullStackers!</h1> ]

***xx. Refresh homepage in browser to see the changes***

http://52.6.154.11 in your browser

[ You should see your updated text ]

**05. Add a profile.html page**

:~$ sudo cp /usr/share/nginx/html/index.html /usr/share/nginx/html/profile.html

[ This will create a profile.html page that is identical to index.html ]

**06. Update the profile.html page w/ "Profile" text**

:~$ sudo nano /usr/share/nginx/html/profile.html

[ Change <h1>Welcome to Nginx!</h1> ... to ... <h1>Your Profile Page</h1> ]

***xx. Go to profile.html page via your browser to see the new page***

http://52.6.154.11/profile.html in your browser

[ You should see your profile page ]

--------

Back on your Linux EC2 instance:

**01. Look for php ( this should only return a few (or no) files )**

:~$ find / -name "php\*" 2> /dev/null

**02. Install php and php/pg components**

:~$ sudo apt-get install php5-fpm php5-common php5-json

*Key Files:*

/etc/php5/fpm/php.ini (php configuration settings)

/etc/php5/fpm/pool.d/www.conf (so php5-fpm listens on the socket)

**03. Edit etc/php5/fpm/php.ini file**

:~$ sudo nano /etc/php5/fpm/php.ini

UPDATE the following line in the file:

;cgi.fix\_pathinfo = 0;

**04. Edit etc/php5/fpm/pool.d/www.conf file**

:~$ sudo nano /etc/php5/fpm/pool.d/www.conf

CONFIRM the following line is in the file (uncommented):

listen = /var/run/php5-fpm.sock

-------

**01. Edit etc/nginx/sites-available/default file**

:~$ sudo nano /etc/nginx/sites-available/default

UPDATE the 2nd line (immediately below) ... insert "index.php" (no quotes) right after "index":

root /usr/share/nginx/html;

index **index.php**  index.html index.htm

Uncomment (delete the #) the following lines:

location ~ \.php$ {

# fastcgi\_split\_path\_info ^(.+\.php)(/.+)$;

# fastcgi\_pass 127.0.0.1:9000;

fastcgi\_pass unix:/var/run/php5-fpm.sock;

fastcgi\_index index.php;

include fastcgi\_params;

}

--------

**01. Create a index.php file**

:~$ sudo nano /usr/share/nginx/html/index.php

INCLUDE the lines:

<?php

echo phpinfo();

?>

***xx. FAILED #2 (Connect to website PHP homepage)***

http://52.6.154.11

**02. Restart php5-fpm AND THEN Restart nginx to see index.php as homepage**

:~$ sudo service php5-fpm restart

:~$ sudo service nginx restart

***xx. SUCCESS #2 (Connect to website PHP homepage)***

http://52.6.154.11 in your browser

[ You should see a bunch of php configuration stuff ]

--------

**01. Look for postgresql ( this should only return a few (or no) files )**

:~$ find / -name "postg\*" 2> /dev/null

**02. Create repository file**

:~$ sudo nano /etc/apt/sources.list.d/pgdg.list

Add the line to the (blank) file:

deb http://apt.postgresql.org/pub/repos/apt trusty-pgdg main

**03. Import the repository signing key**

USE THIS FIRST ONE!!!

:~$ wget --quiet -O - http://apt.postgresql.org/pub/repos/apt/ACCC4CF8.asc | sudo apt-key add -

[ This is all one line! ]

----------------------------------

:~$ wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc

:~$ sudo apt-key add ACCC4CF8.asc

----------------------------------

:~$ wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -

[ This is all one line! ]

**04. Update the package lists**

:~$ sudo apt-get update

**05. Install Postgresql 9.4**

:~$ sudo apt-get install postgresql-9.4

**06. Confirm the correct version of Postgresql is installed (9.4)**

:~$ psql --version

--------

**01. Get into Postgresql for the first time**

:~$ sudo -u postgres psql postgres

You should now be connected to your Postgresql database (9.4)

**02. List all the original databases**

=# \list

You should see 3 databases listed

**03. Create a new database**

=# CREATE DATABASE fullstacker;

**04. See what database you're currently in**

=# SELECT current\_database();

**05. Connect to the newly created database**

=# \c fullstacker

**06. Create a new table in the newly created database**

=# CREATE TABLE testuser (userid INTEGER, username VARCHAR(100));

**07. Insert records in the new table**

=# INSERT INTO testuser (userid, username) VALUES (1, 'UserOne'), (2, 'UserTwo');

**xx. Confirm your insert was successful**

=# SELECT userid, username FROM testeruser;

**08. To exit Postgresql**

=# \q

--------

@@@@@@@@

--------

**01. Install php5-pgsql**

:~$ sudo apt-get install php5-pgsql

**02. Create a PHP/Postgresql page for testing**

:~$ sudo nano /usr/share/nginx/html/test.php

**03. Paste in PHP code into this new page (test.php)**

<?php

// Connect to your database

$db = pg\_connect("host=localhost dbname=fullstacker user=postgres")

or die('Could not connect: ' . pg\_last\_error());

$sql = "SELECT userid, username FROM testuser";

$result = pg\_query($db, $sql);

if ( !$result )

{

die("Error in SQL query: " . pg\_last\_error());

}

while ( $row = pg\_fetch\_assoc($result) )

{

echo "UserID: " . $row['userid'] . "<br />";

echo "User Name: " . $row['username'] . "<br />";

}

pg\_free\_result($result);

pg\_close($db);

?>

***xx. FAILED #3 (Connect to PHP/Postgresql page)***

Go to http://52.6.154.11/test.php in your browser

-------

Prep php and postgresql to talk to each other

**04. Update php.ini file**

:~$ sudo nano /etc/php5/fpm/php.ini

Add this line at the bottom of the page:

extension=pgsql.so

**05. Restart nginx (your web server)**

:~$ sudo service nginx restart

**06. Update postgresql.conf file**

:~$ sudo nano /etc/postgresql/9.4/main/postgresql.conf

(a) Search for 'localhost' in the file

Ctrl-W >> localhost

(b) Find the line:

**listen\_addresses = 'localhost'**

(c) Remove the "#" at the beginning of the line

(d) Replace 'localhost' w/ '\*' (ignore the apostrophes)

THE **FINISHED LINE** SHOULD LOOK LIKE THIS: **listen\_addresses = '\*'**

**07. Modify the pg\_hba.conf file**

:~$ sudo nano /etc/postgresql/9.4/main/ pg\_hba.conf

CHANGE THE FOLLOWING METHODS TO trust:

TYPE DATABASE USER ADDRESS METHOD

local all all trust (peer)

host all all 127.0.0.1/32 trust (md5)

host all all ::1/128 trust (md5)

**08. Restart Postgresql**

:~$ sudo service postgresql restart (works)

OR

:~$ /etc/init.d/postgresql restart (didn't work for me)

**xx. SUCCESS #3 (Connect to PHP/Postgresql page)**

Go to http://52.6.154.11/test.php

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

***CONGRATULATIONS!!!*** You now have your components talking to each other:

(1) Web server

(2) PHP code

(3) Postgresql database

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**01. Request an Elastic IP**

- Go to your AWS console

- Select 'EC2'

- Choose 'Elastic IPs' under the 'Network & Security' group (left panel)

- Click 'Allocate New Address' button (near the top of the page)

- Select '**VPC**' [**not 'EC2'**] (In the pop-up window... **'EIP used in :'**)

- Click the 'Yes Allocate' button

- Close the pop-up window

- Select the new Elastic IP

- Click the 'Associate Address' button

- Select your EC2 instance in the 'Instance' textbox

- Click the 'Associate' button

**02. Set up Route 53 Domain Name System ---> Only for domains purchased more than 60 days ago!**

\* Like a phone book (resolves domain name to IP address)

\* Costs around $1.20 / month

-----------------------

- In Your AWS Console:

-----------------------

- Select 'Route 53'

- Click the 'Create Hosted Zone' button

- Type in your domain name (no http://www.)

- Make sure the Type is 'Public Hosted Zone' (if you want your site to be public)

- Click the 'Create' button

- You should see 4 AWS name servers:

For example:

ns-1570.awsdns-04.co.uk.

ns-487.awsdns-60.com.

ns-1301.awsdns-34.org.

ns-672.awsdns-20.net.

- Go back to your domain provider (GoDaddy, in this example)

Here you will tell your provider to use the AWS name servers for your domain

**Go to GoDaddy (or whoever is your domain provider):**

- Sign in to your GoDaddy account

- Click on 'Domains'

- Select your domain name (That you own)

- Click the green 'Manage' button

- Select your domain name (again...)

- Click the checkbox in front of the name

- On the 'Nameservers' dropdown:

- Select 'Set Nameservers' option

- Click the 'Custom' radio button

- Click the 'ADD NAMESERVER' button

- Paste in the first of your AWS nameserver names (w/o the ending period)

- Continue adding the other AWS nameservers (see jpg image)

- Click the 'Save' button

**Back in AWS:**

- Go to your Route53 page

- Click on the 'Hosted Zones' circle

- Select your domain's hosted zone

- Click the 'Go to Record Sets' button

- Click the 'Create Record Set' button

- Leave the **'Name:'** textbox blank!!!

- Keep the **'Type:'** as 'A-IPv4 address'

- In the Value box, enter your Elastic IP address (e.g. 52.7.154.23)

- Click the 'Create' button

- Click the 'Create Record Set' button

- In the 'Name:' textbox enter "www" (no quotes)

- Keep the 'Type:' as 'A-IPv4 address'

- In the Value box, enter your Elastic IP address (e.g. 52.7.154.23)

- Click the 'Create' button

- It may take a few minutes for the change to go through

- **REMEMBER:** Amazon states that the domain must have been purchased over 60 days ago!

--------

**Prepare a location to receive image uploads from your site members:**

On your Linux EC2 instance:

**01. Create a 'profile' directory in your root folder (/usr/share/nginx/html)**

:~$ sudo mkdir profiles

**02. Change the permissions on this directory**

:~$ sudo chmod 777 profiles (if you're already in: /usr/share/nginx/html)

------------------------------------------------------------

Upload scripts to build the login application. All scripts should go to your root folder in Linux

**/usr/share/nginx/html**

|  |  |
| --- | --- |
| index.php | Your homepage |
| signup.php | Form for new member sign ups |
| signup-process.php | The actual code that processes member sign ups |
| login.php | Form for returning member logins |
| login-process.php | The actual code that validates member logins |
| logout.php | The actual code that logs out members (no form) |
| profile.php | Profile page for each logged-in member |
| upload-pic.php | Form on the profile page that allows image uploads |

This location (which you created earlier) is where members can upload their profile pics.

upload-pic.php is the script that actually transfers the files to this location.

**/usr/share/nginx/html/profiles**

*Please let me know if you have any questions!*

*Good Luck!*

*Bryan*