2FA OTP WEB APPLICATION

IS690D FALL 2020
PROFESSOR NASSAR

INSTALLATION GUIDE

BY JASMINE WASHINGTON, ALEXANDRA IFILL, AND SKANDHAN NETHI

CONTENTS

Getting Started

Welcome to the installation guide for 2FA OTP, a PHP, HTML, CSS and JavaScript web application! After setting up the proper environment on your machine, you will be able to run the 2FA OTP web application both locally and on the cloud!

Download XAMPP

XAMPP is the most popular PHP development environment. XAMPP is a completely free, easy to install Apache distribution containing MariaDB, PHP, and Perl. The XAMPP open source package has been set up to be incredibly easy to install and to use [Apache Friends]. Please download the latest version of XAMPP for your OS here: https://www.apachefriends.org/

*Note: For Windows users, you need to change the default installation path from "C:/Program Files" to ANY other path. You will receive many errors during installation about UAC or your XAMPP application may not run properly.

Once XAMPP is installed, you should see something similar as shown in Figure 1 below:

Figure 1 Mac OS X XAMPP Application

You can start the required services by going to the Manage Servers tab as shown in Figure 2:

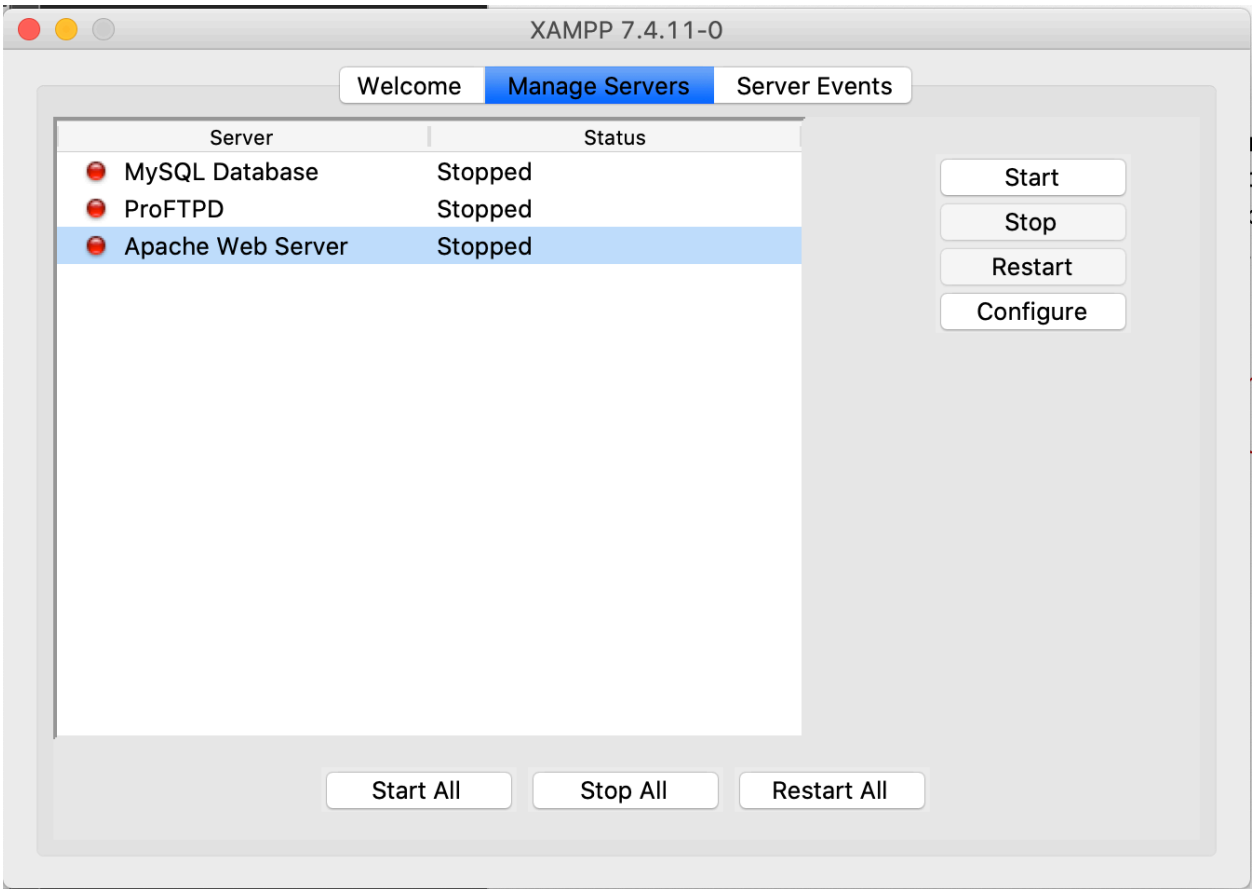


Figure 2 Manage Servers MAC OSX

For our 2FA OTP application, we only need MySQL Database and Apache Web Server running. Please turn on these servers now. Visit http://localhost to see a message saying the installation was successful as shown in Figure 3:

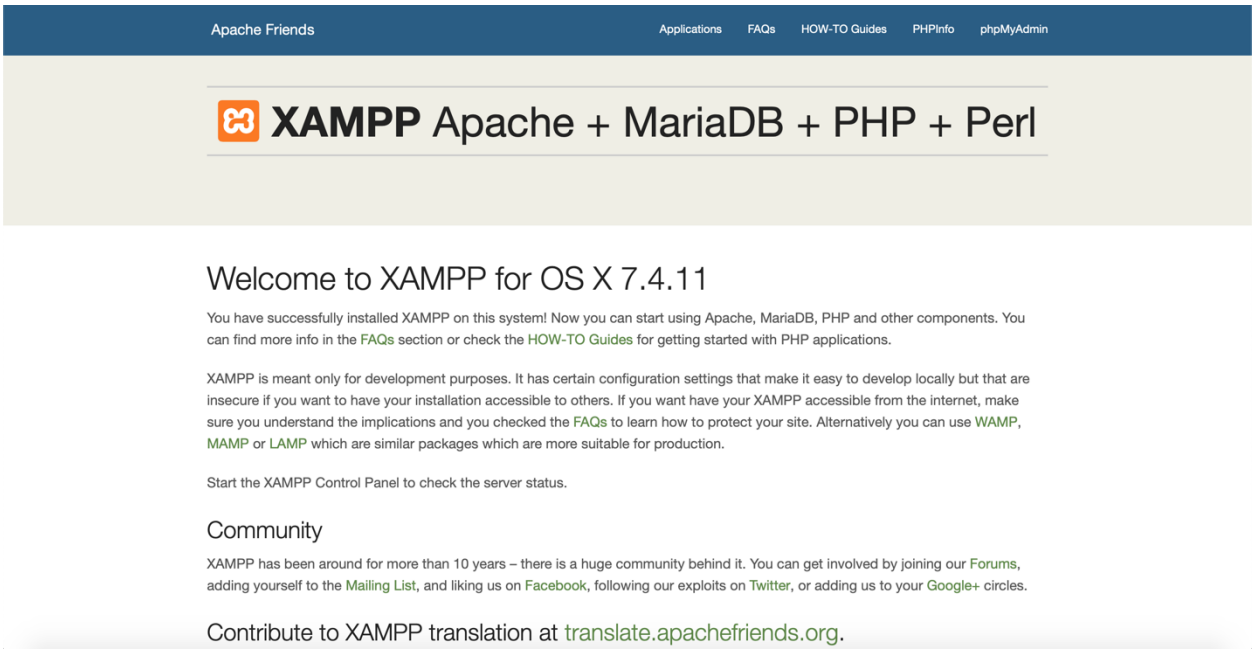


Figure 3 Successful XAMPP Installation localhost page

Download Visual Studio Code

Visual Studio Code is a free, open source, code editor with many functions such as a built-in git, run and debug capabilities, and IntelliSense. Please download the latest version here: https://code.visualstudio.com

*Note: Downloading VS Code is optional. Any code editor is fine to use for our 2FA OTP application!

Once Visual Studio Code is installed, you should see something similar as shown in Figure 4:

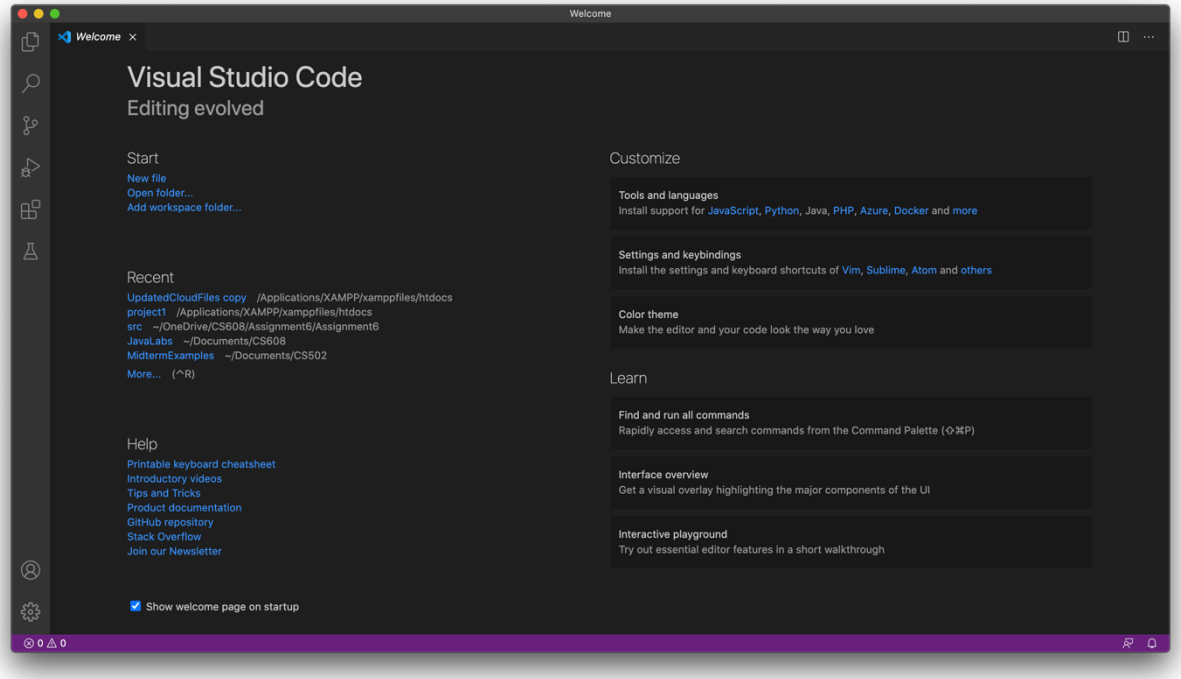


Figure 4 Visual Studio Code MAC OSX

Download FileZilla

FileZilla is a free, open source, FTP solution. It also supports FTP over TLS (FTPS) and SFTP. We will need SFTP when we are ready to deploy on the cloud! Please download the latest version of FileZilla from here: https://filezilla-project.org

After installing FileZilla, you should see something similar as shown in Figure 5:

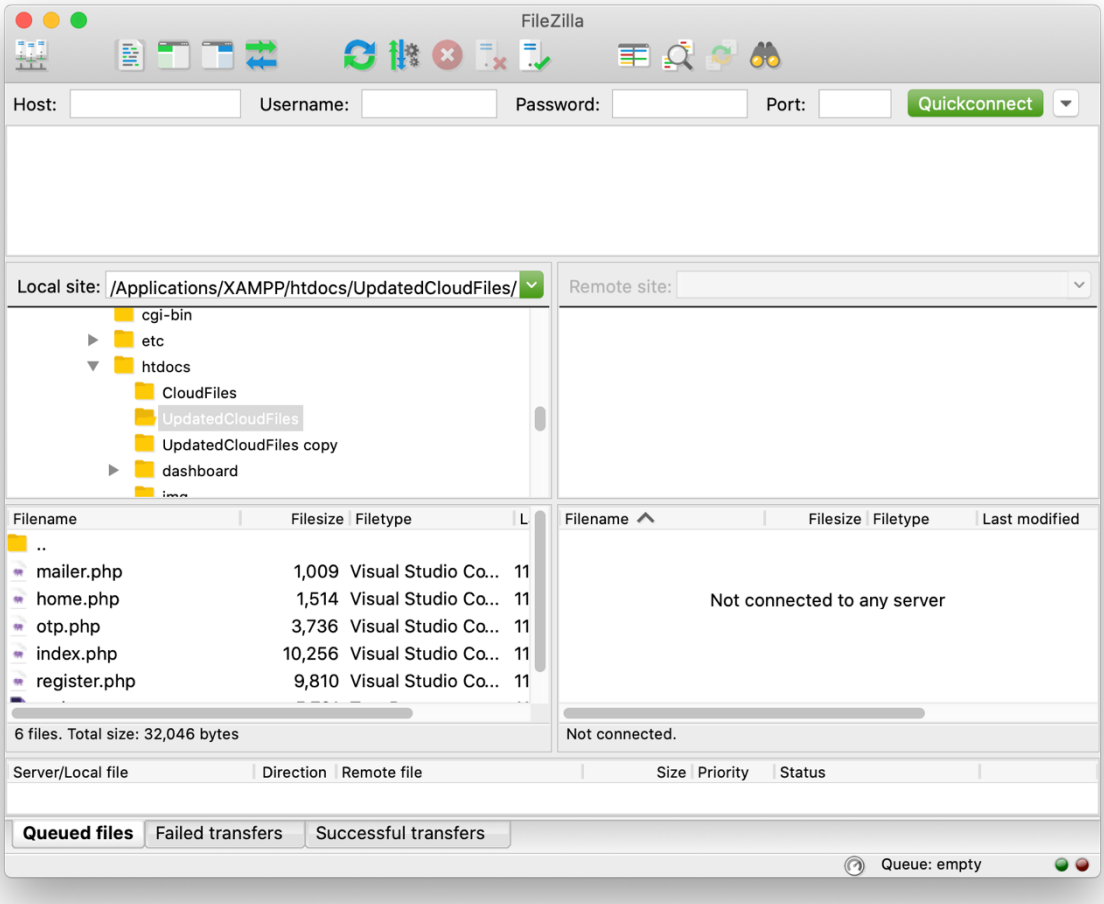


Figure 5 FileZilla on MAC OSX

Download Composer

Composer is a dependency manager for PHP. It allows you to declare the libraries your project depends on and it will manage (install/update) them for you. Please download the latest version here: https://getcomposer.org/download/

*Note: Downloading Composer is only required for local deployment. Composer will already be installed in the cloud environment!

After installing composer,

1. Run the command (as sudo if needed)
   mv composer.phar /usr/local/bin/composer chmod +x
   /usr/local/bin/composer

You should be able to run “composer” or “composer.phar” and see the following output along with other messages in any directory:

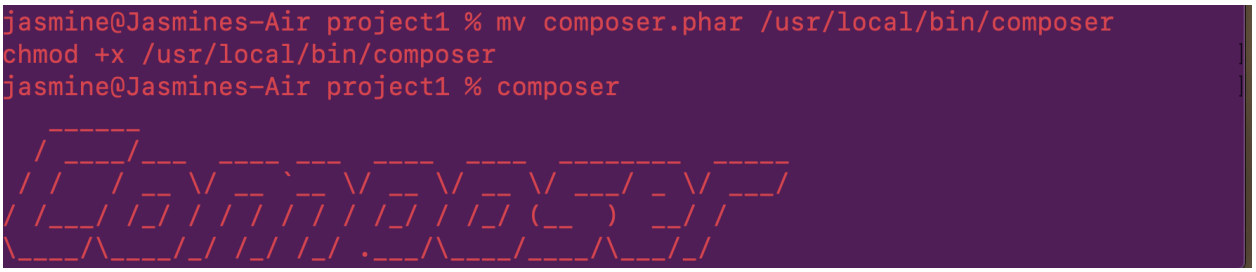


Figure 6 Composer MAC OSX terminal

Download 2FA OTP code from GitHub

Now we need the 2FA OTP files. You can download them from here: https://github.com/jazzymaya/2FAOTP.git

You can use Git, download via zip, or however you are familiar with getting files from GitHub.

Please read the README.md file to understand which file does what for the application.

Set Up phpMyAdmin Database

If it is not open already, please open your browser to http://localhost

On the top right corner, you should see a link that says, “phpMyAdmin” as shown in Figure 7:

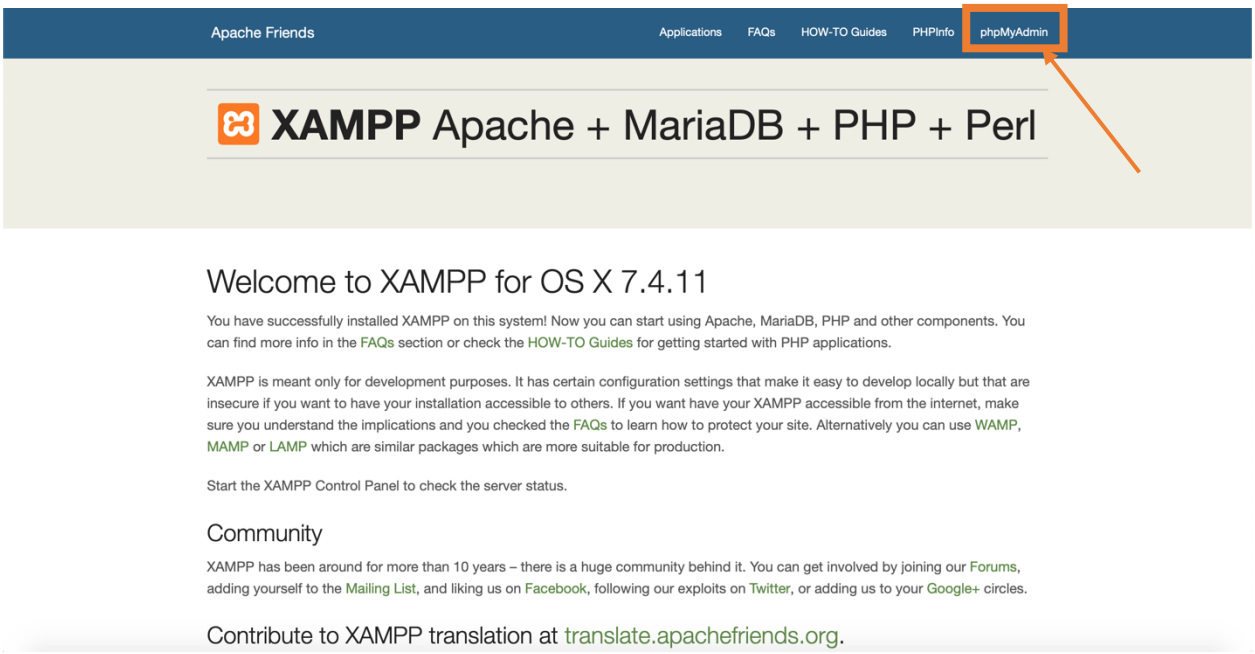


Figure 7 phpMyAdmin link

Click on this link and you should see the following page:

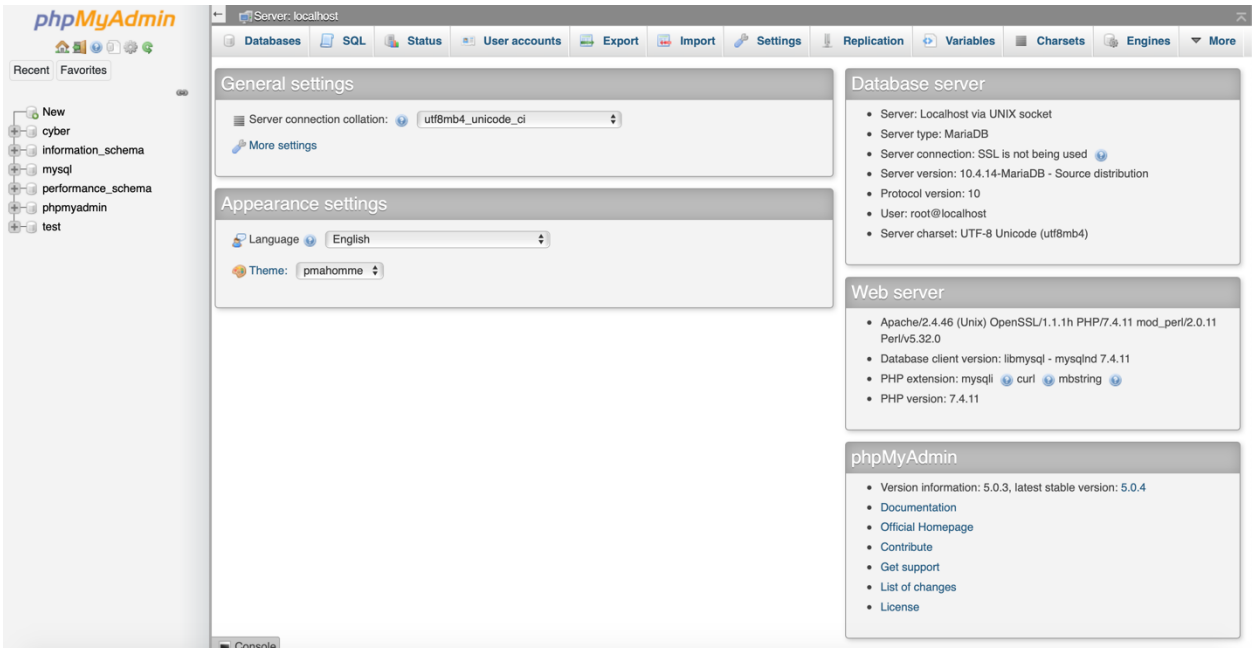


Figure 8 phpMyAdmin homepage

Click on the “New” button located at the top left, indicated in the Figure below:

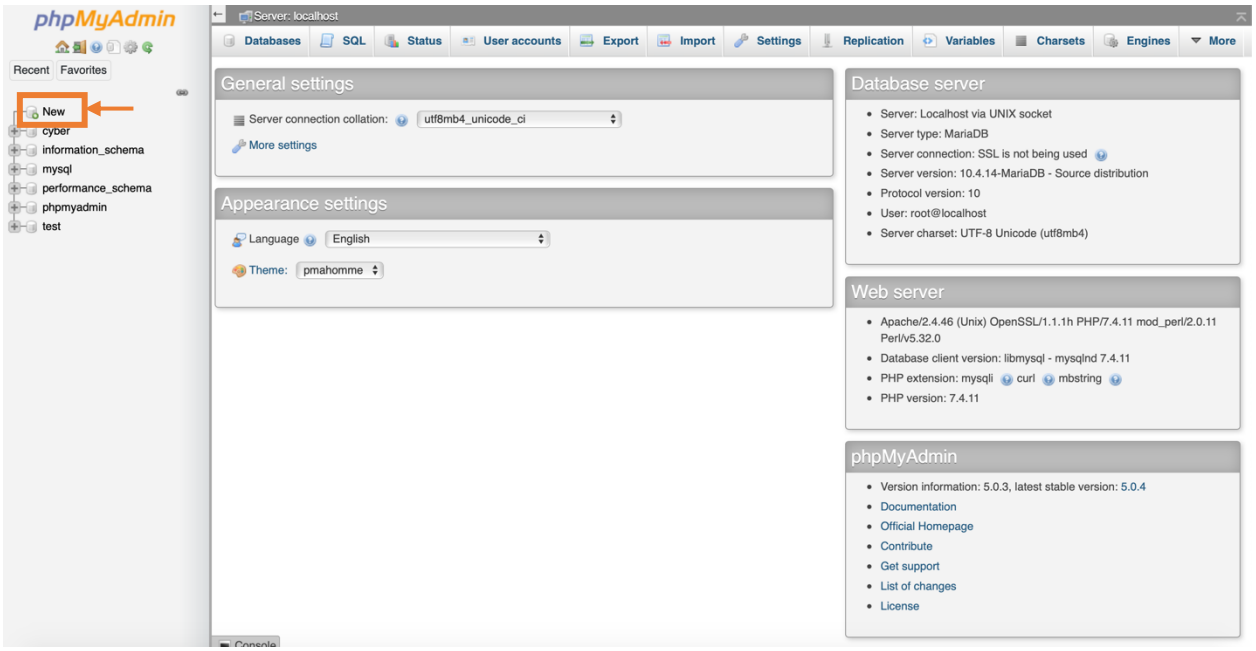


Figure 9 New button on phpMyAdmin homepage

You will then see the Figure below. Please create a database name. You can name it whatever you like. For this exercise, I named it “example”.

*Note: Remember the name of your newly created database. We need it for section 2.3!

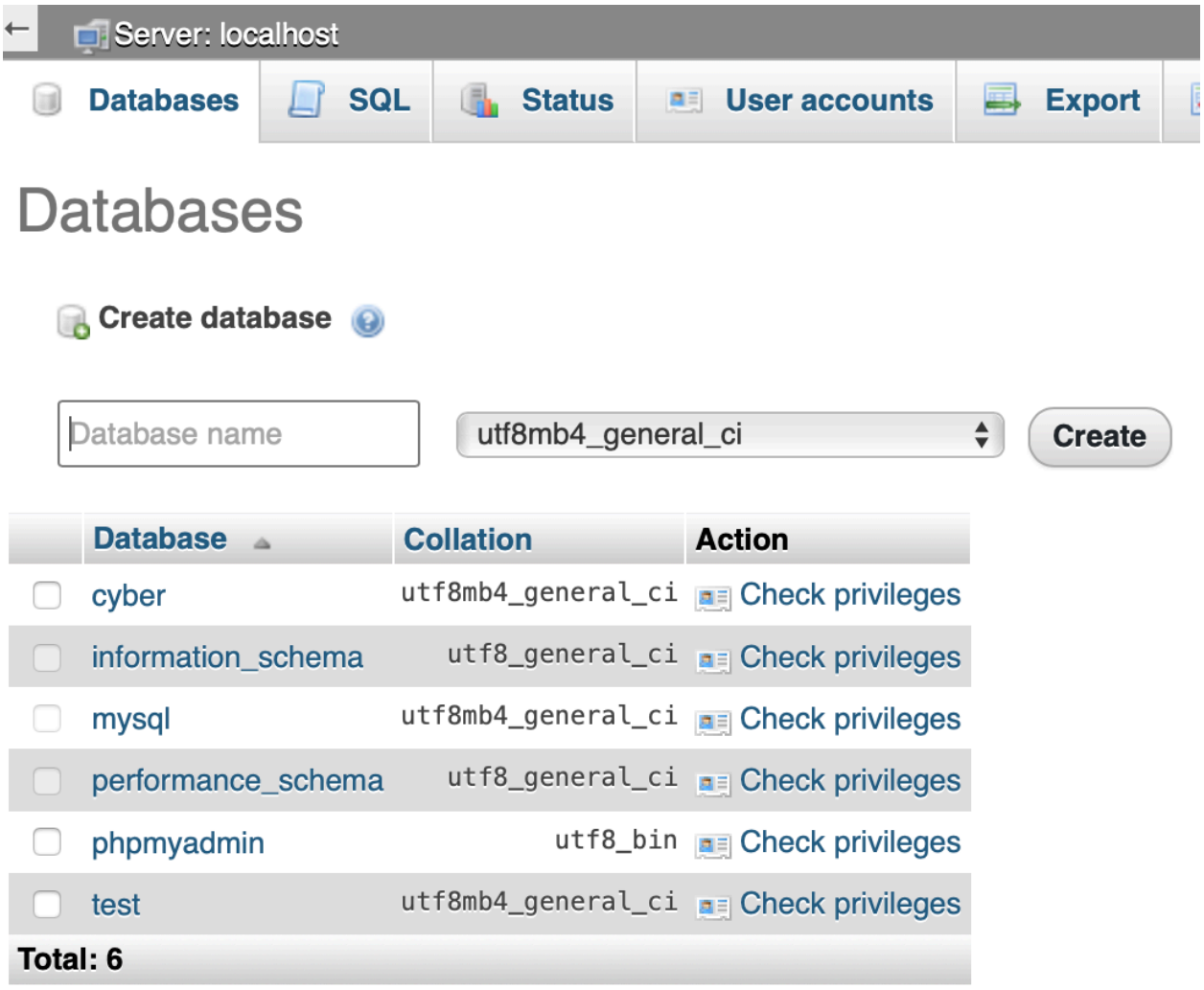


Figure 10 Database Name creation

After creating you new database, you will be redirected to this screen:

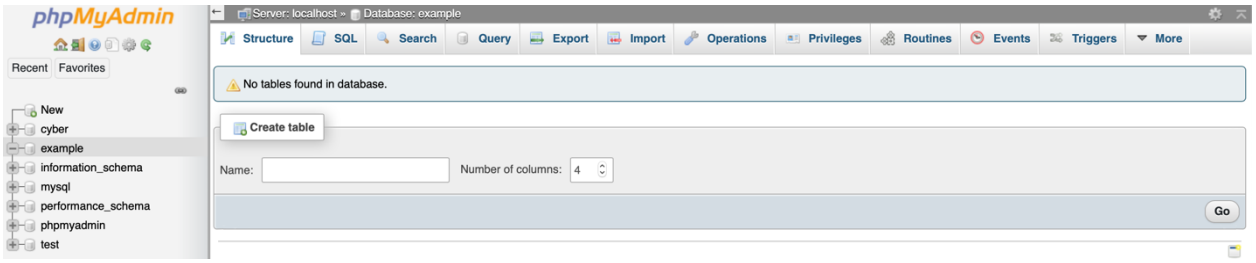


Figure 11 After Database Creation phpMyAdmin

Please click on the “import” tab located on the top as shown in Figure 12:

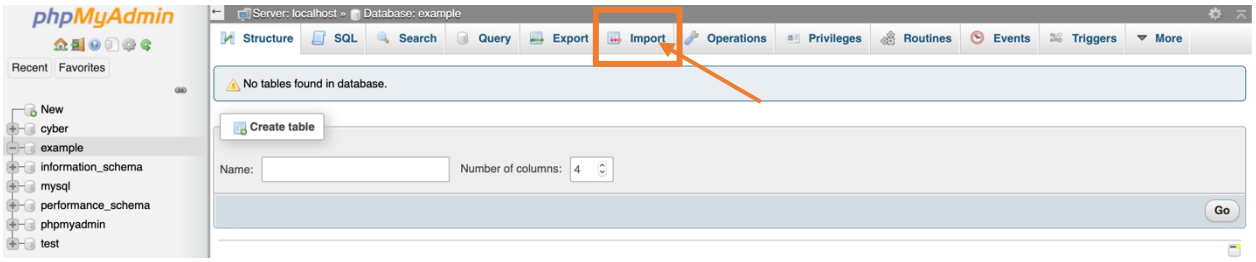


Figure 12 Import tab phpMyAdmin

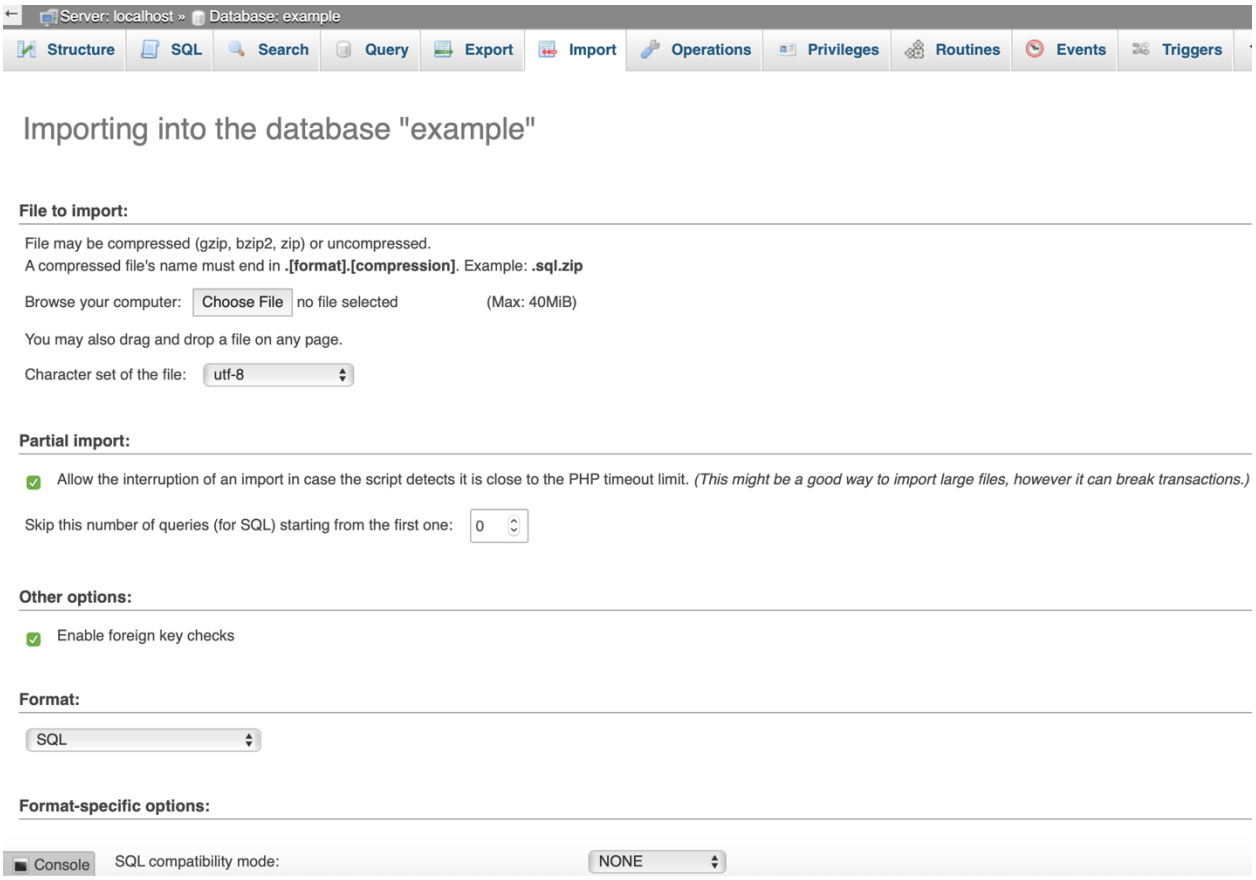


Figure 13 Import page phpMyAdmin

In Figure 13, we see a “Choose File” button. Please click on this button and navigate to the “accounts.sql” file that was downloaded from the 2FAOTP GitHub repo. Leave everything as its default setting and scroll down to click the button, “Go” for submission. Figure 14 shows a successful page of each SQL statement included in the accounts.sql file:

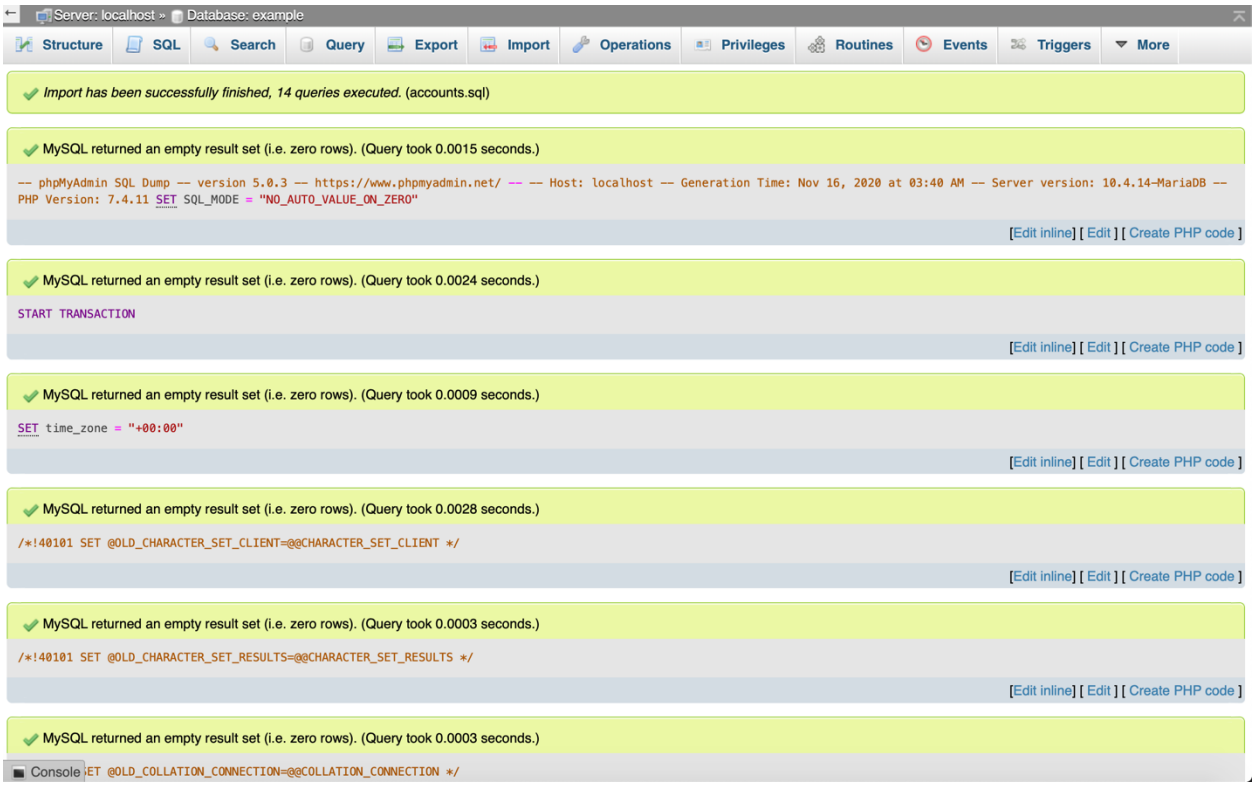


Figure 14 Successful SQL Import

Now, navigate back to the "Structure" tab located on the top left of Figure 14.

You will see the new table "accounts" as shown in Figure 15:

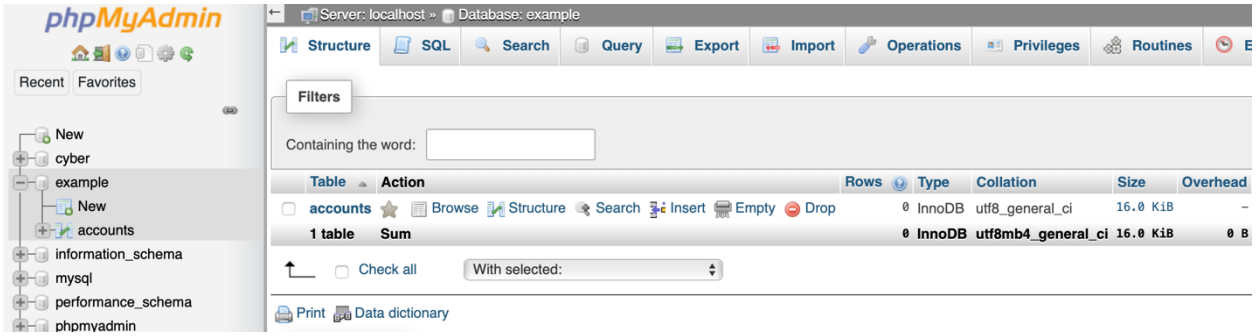


Figure 15 Accounts table in phpMyAdmin

Right now, this table is empty. We will populate it section 2.3!

2FA OTP Web App (Local)

Now that we have most of the required applications downloaded, we can start running our 2FA Web App!

Set up XAMPP File Structure

Where ever you installed XAMPP, you should see the file structure as shown in Figure 8:

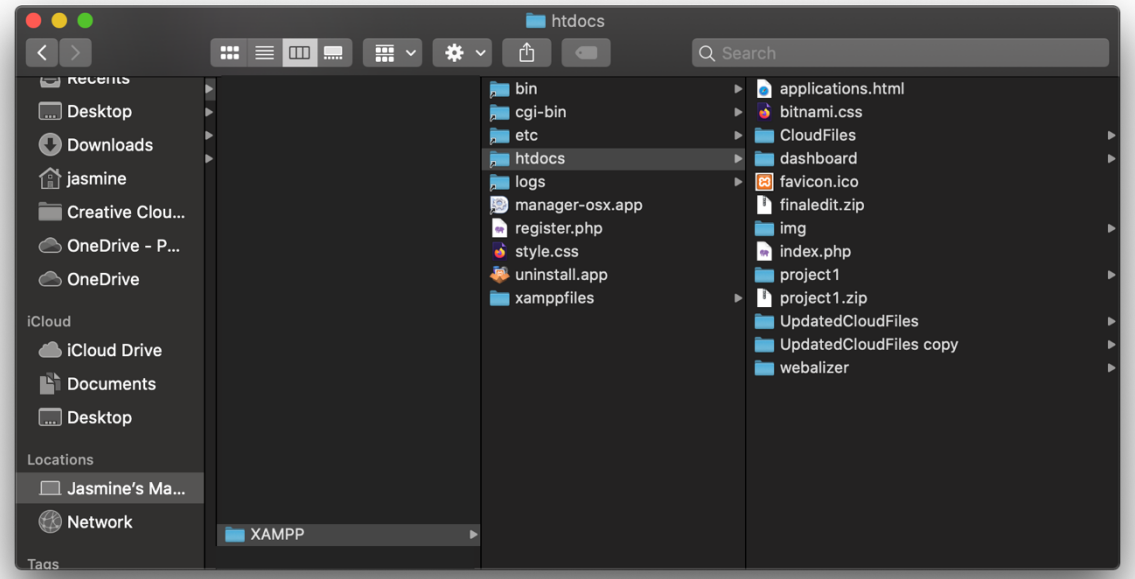


Figure 16 XAMPP File Structure MAC OSX

Do not worry if the inside of your htdocs does not look exactly like the image provided.

1. Create a new folder within the htdocs folder. (You can name this folder anything you want. In my example, it is named project1)
2. Move the 2FA OTP code you downloaded from GitHub into this newly created folder. The contents of the folder should look like Figure 9:

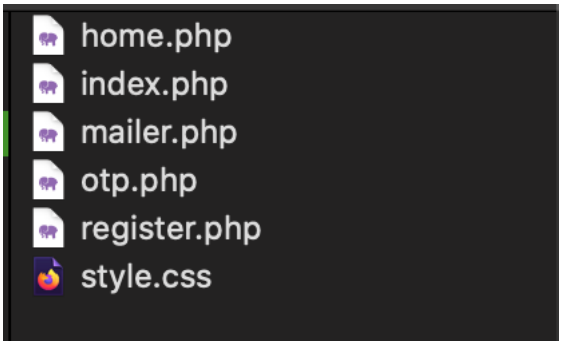


Figure 17 Files within the new project1 folder

Download PHPMailer

PHPMailer is a full-featured email creation and transfer class for PHP. It allows our application to be able to send users the OTP email for 2FA! You can download PHPMailer by using composer!

1. Open a terminal in the following path:
   your-XAMPP-folder-path/htdocs/project1

*Note: "your-XAMPP-folder-path is wherever you have XAMPP installed. For Windows users this should not be in C:/Program Files as stated in the XAMPP section. Also, remember project1 is a folder we created in Step 1 of "Set Up XAMPP File Structure".

2. Run the following command to install PHPMailer in this directory:
   composer require phpmailer/phpmailer

You should see something similar to the following output within your terminal:

```
jasmine@Jasmines-Air [redacted] % composer require phpmailer/phpmailer
Using version ^6.1 for phpmailer/phpmailer
./composer.json has been created
Running composer update phpmailer/phpmailer
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
  - Locking phpmailer/phpmailer (v6.1.8)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
  - Installing phpmailer/phpmailer (v6.1.8): Extracting archive
5 package suggestions were added by new dependencies, use `composer suggest` to
see details.
Generating autoload files
1 package you are using is looking for funding.
Use the `composer fund` command to find out more!
```

Figure 18 Composer Install PHPMailer

Notice the change in the project1 directory:

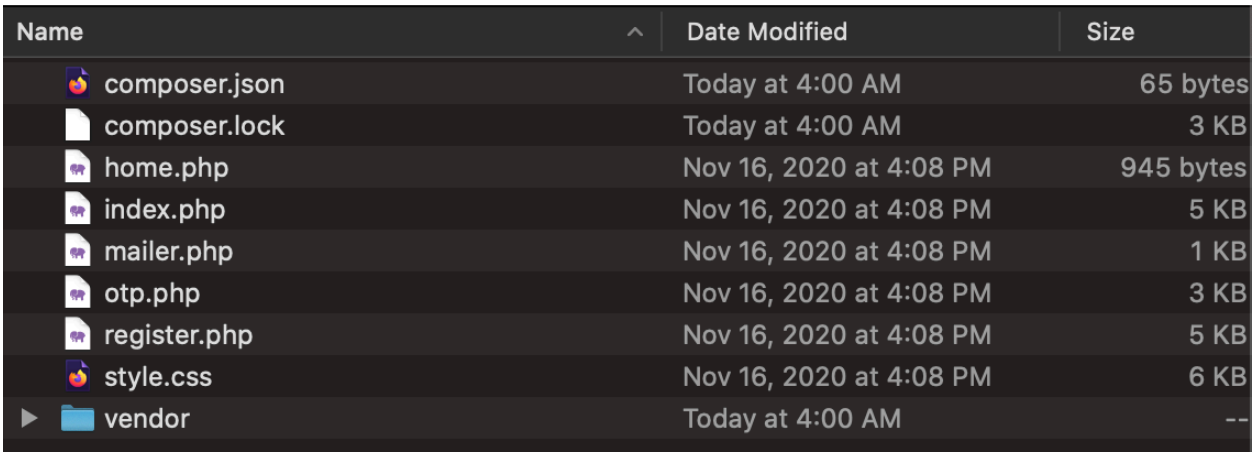


Figure 19 Changed project1 directory

Edit code credentials

Now, we need to use Visual Studio Code (or your preferred text editor) to edit the code's credentials. Right now, the code does not have valid database or email credentials. This is why the code is not working if you tried to run it before this section!

1. Open Visual Studio Code and click on "Open Folder" under the Start menu:

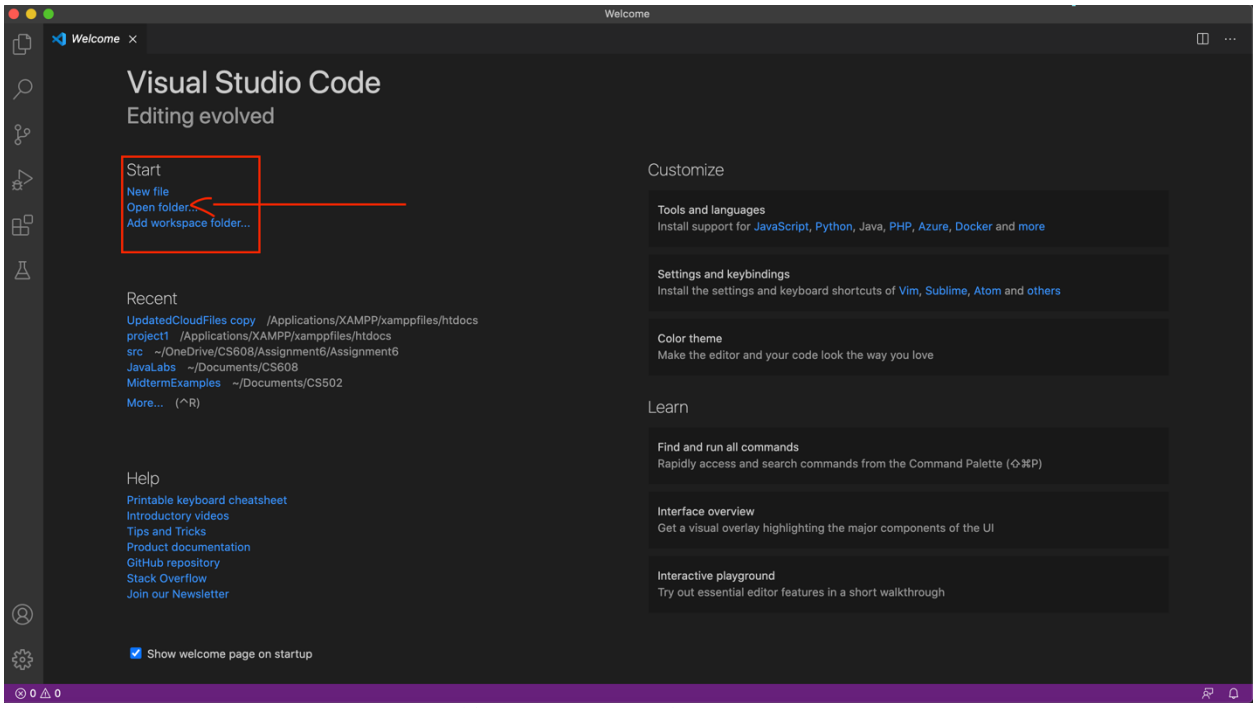


Figure 20 Open Folder Option in VS Code MAC OSX

2. Open to the path of our "project1" folder (where the 2FAOTP code resides)
3. Open the following files:
    a. index.php
    b. register.php
    c. otp.php
    d. mailer.php
4. In the file "index.php", change line 27, "$DATABASE_NAME = '';" to have your newly created databse name from section 1.7. In my example, it would be:

```
$DATABASE_NAME = 'example';
```

5. In the file "register.php", change line 22 to the same database name as you did in index.php
6. In the file "otp.php", change line 16 to the same database name as you did in index.php

Now, you must decide what email address you want to use for this application. We recommend using Gmail because…

1. You will have to create a Google account for the Cloud deployment anyways

2. We successfully ran this code using a gmail account

Each email provider may have different settings, outlook, yahoo, msn, school domains, etc. We do not have a guide for these other providers. Please lookup the proper settings for your email provider.

If you decided to use Gmail for your application deployment, please make sure you do the following:

1. Create a Google account. https://accounts.google.com/SignUp
2. Go to https://myaccount.google.com/security
3. Make sure 2-Step Verification and Use your phone to sign in options are DISABLED/OFF
4. Make sure Less Secure App Access is ENABLED/ON
5. Visit https://accounts.google.com/b/0/DisplayUnlockCaptcha and allow access to your Google account.

After performing the Google steps, open the "index.php" page within Visual Studio Code. Enter the following information:

1. Go to line 81, enter "smtp.gmail.com" in-between the quotation marks.
2. Go to line 83, enter your new google account email.
3. Go to line 85, enter you new gmail account's password.
4. Go to line 92, enter your gmail address in the empty "" and write an alternate name that you want users to see you by.
5. Go to line 94, enter your gmail address in the empty "" and write an alternate name for the reply-to field.

The file "mailer.php" is very similar to the "index.php" fields. Please fill in the required portions so that you can run "mailer.php" for debugging purposes. Fields required are lines 19, 21, 23, 26, and 27 respectively.

Run the application

You are now ready to run the 2FA OTP application via localhost! We recommend running the script "mailer.php" before attempting the 2FA OTP. "mailer.php" will show you if there are errors in sending emails from your gmail account. You can run this by go to the following URL:

http://localhost/project1/mailer.php

*Note: The folder “project1” may be named something else if you did not follow my example structure.

If your “mailer.php” respone was successful or if you skipped that step, you can run the 2FA OTP application by navigating to this URL: http://localhost/project1/index.php

See section 4 for screenshots of a running application!

2FA OTP Web App (Cloud)

Deploying to the cloud is easy! The following guide below is from Vikram Vaswani: https://www.apachefriends.org/docs/hosting-xampp-on-google.html
We will include his original guide along with our notes on certain steps in red as previous notes have been format.

Quick Links

Introduction

If you're a PHP developer building a public-facing Web application, there are a number of good reasons why the cloud should be on your radar. It's highly scalable, allowing you to quickly scale up if you application turns out to be a hit. It's cost-efficient, because you only pay for the resources - bandwidth, CPU cycles, memory - you use. And it's secure, because cloud providers have invested a great deal of time and thought into ring- fencing applications and user data.

However, if you're new to the cloud or do most of your development locally, getting your PHP application from your local XAMPP box to the cloud can be a bit challenging. That's where this tutorial comes in. Over the next few pages, I'll walk you, step by step, through the process of deploying a PHP/MySQL application running on your local XAMPP server, to a cloud server running Bitnami's LAMP Stack. Keep reading!

What You Will Need

Before we begin, a few quick assumptions. This tutorial assumes that you have a XAMPP installation with a working PHP/MySQL application. It also assumes that you're familiar with the MySQL command-line client and that you have a working knowledge of transferring files between servers using FTP.

> If you don't have a custom PHP/MySQL application at hand, use the example application included with this tutorial: it's a simple to-do list, created with Twitter Bootstrap and PHP. You can download it from here.

Note: We already have a working PHP/MySQL application (2FAOTP) so you don't need the Twitter Bootstrap example.

Now, if you're new to the cloud, you might be wondering what Google Cloud Platform and Bitnami are. Very briefly, Google Cloud Platform is a cloud service offering, which allows you to easily create Windows and Linux virtual servers online. Bitnami provides pre-packaged server images for these cloud servers, so that you can become productive with them the moment they come online. In short,

Google provides the cloud infrastructure, and Bitnami provides the server images and software. And since Google Cloud Platform currently offers a 60- day free trial, you can easily experiment with it without worrying about being billed for usage.

For this tutorial, I'll be using the Bitnami LAMP Stack, which is Linux-based and bundles PHP, MySQL and Apache, together with key applications and components like phpMyAdmin, SQLite, Memcache, OpenSSL, APC and cURL. The LAMP stack also includes a number of common PHP frameworks, including the Zend Framework, Symfony, CodeIgniter, CakePHP, Smarty and Laravel.

To deploy your application to the Google cloud with the Bitnami LAMP Stack, here are the steps you'll follow:

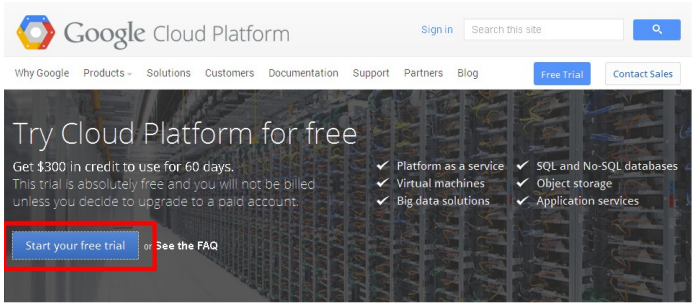
- Register with Google Cloud Platform
- Register with Bitnami
- Connect your Google Cloud Platform and Bitnami accounts
- Provision a cloud server with the Bitnami LAMP Stack
- Validate the cloud server
- Deploy and test your application on the cloud server.

The next sections will walk you through these steps in detail

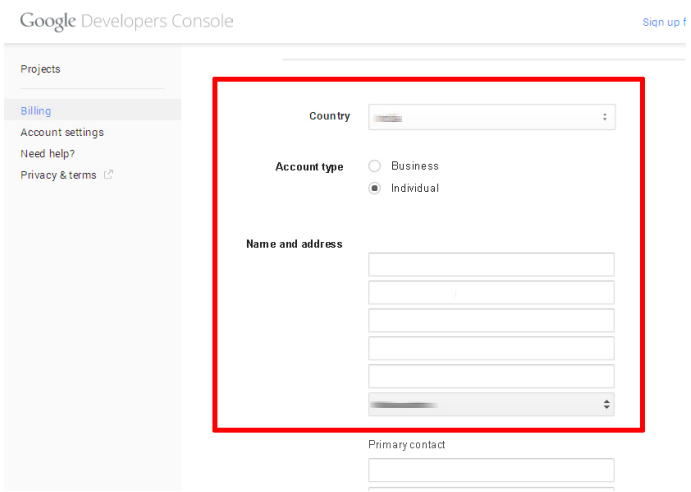
Step 1: Register with Google Cloud Platform

At the end of this step, you will have signed up for the Google Cloud Platform free trial.

Begin by creating a Google Cloud Platform account, by browsing to https://cloud.google.com/ and choosing the "Start your free trial" option. You will need an existing Google account to log in and sign up for the free trial; if you don't have one, you can create one here (remember to keep track of your account username and password, because you'll need them in the next step).



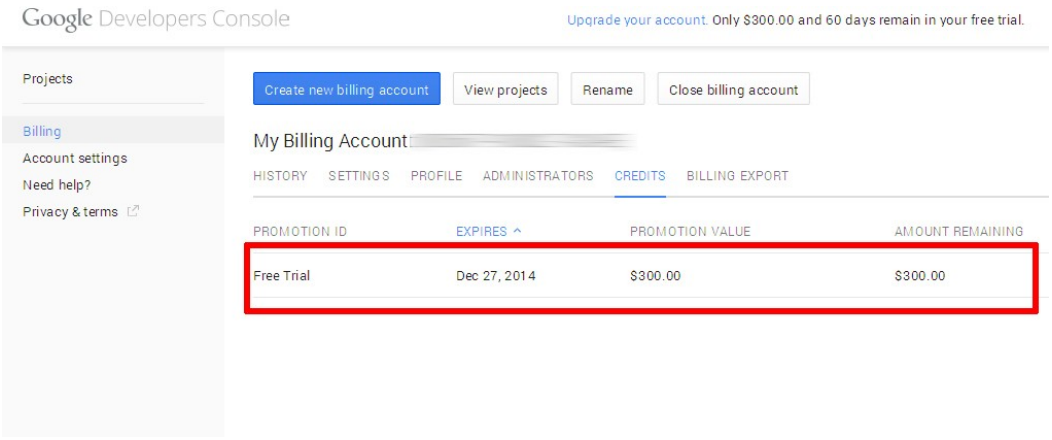
Once you've signed in, provide Google with some personal information and your credit card details.



It's important to note that you're signing up for a free trial and your credit card won't be billed unless you migrate to a paid account. The trial includes $300 of free credit, valid for 60 days. Once the trial ends, your account will automatically be paused and you'll only be charged if you explicitly choose to upgrade to a paid account.

The Google elves will go away for a minute or so to verify your information and

if all is well, you will be redirected to the Google Developers Console, which allows you to manage your billing account, create new projects and get support. You should see that your free trial is now active in the billing credits section.

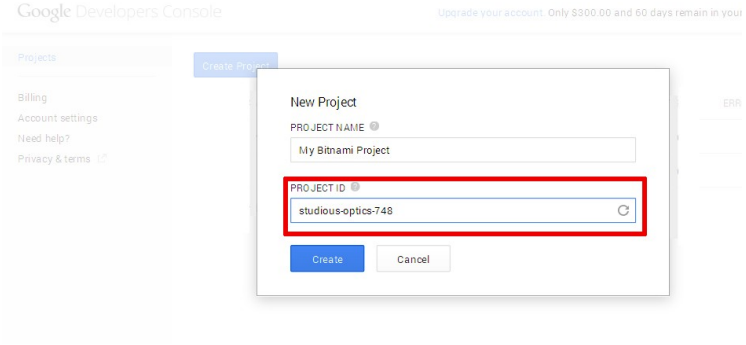


You should also receive an account confirmation email, which tells you that your account is good to go.

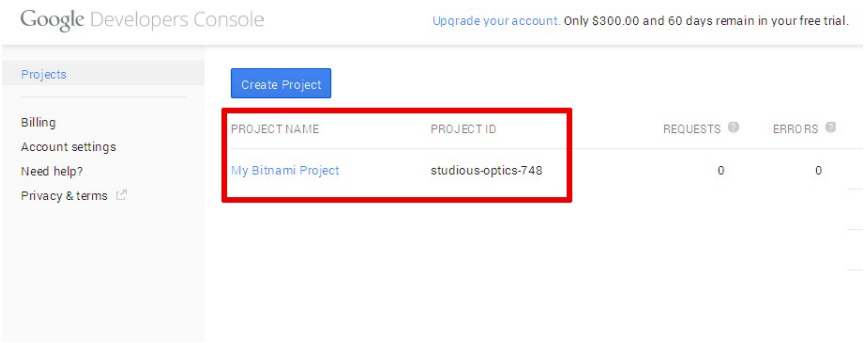
Bitnami uses the Google Compute Engine API in order to manage and launch cloud servers, so this is a good time to enable the API. Plus, new cloud servers always launch within a project, so you'll also need to create a project. Both these tasks are easy to do from the Google Developers Console. Follow these steps:

- While you're logged in to the Google Developers Console, select the "Projects" menu item and click the "Create Project" button.

- Enter a name for the project (such as "My Bitnami Project") and make a note of the auto-generated project ID.



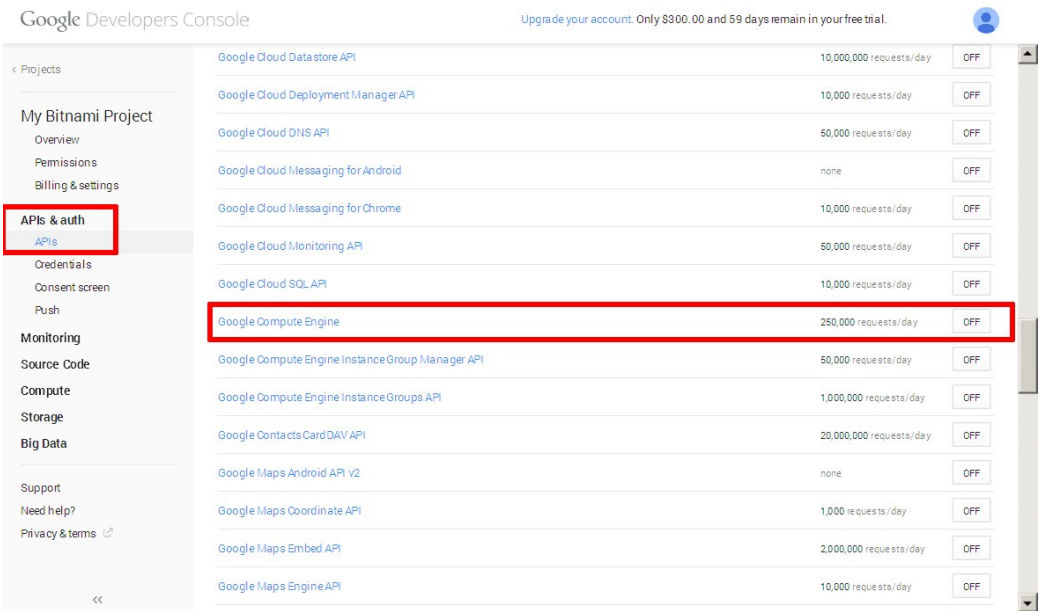
- Click "Create" to create and activate the project. You should now see it in the project listing.



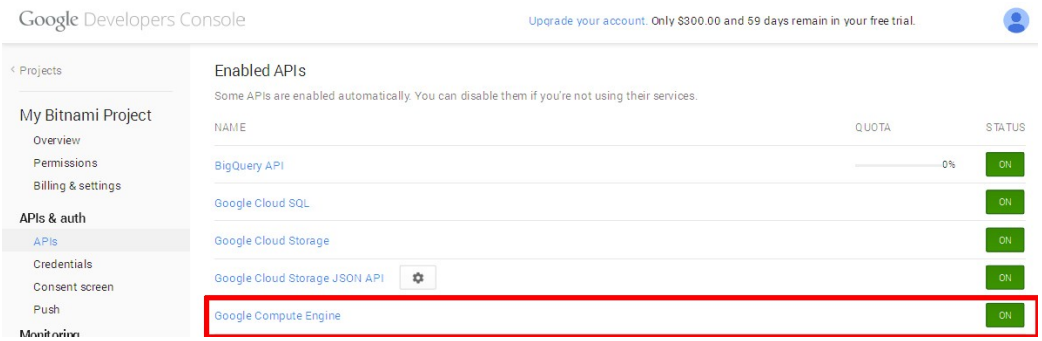
- Select the new project name in the project listing, and you'll be transferred to the project information page.

- Select the "APIs" menu item in the left navigation bar.

- Look through the list of APIs, or use the API search box to search for the term "compute engine". Find and turn on the Google Compute Engine API.



- Verify that the Google Compute Engine API now appears in the list of enabled APIs.



Step 2: Register with Bitnami

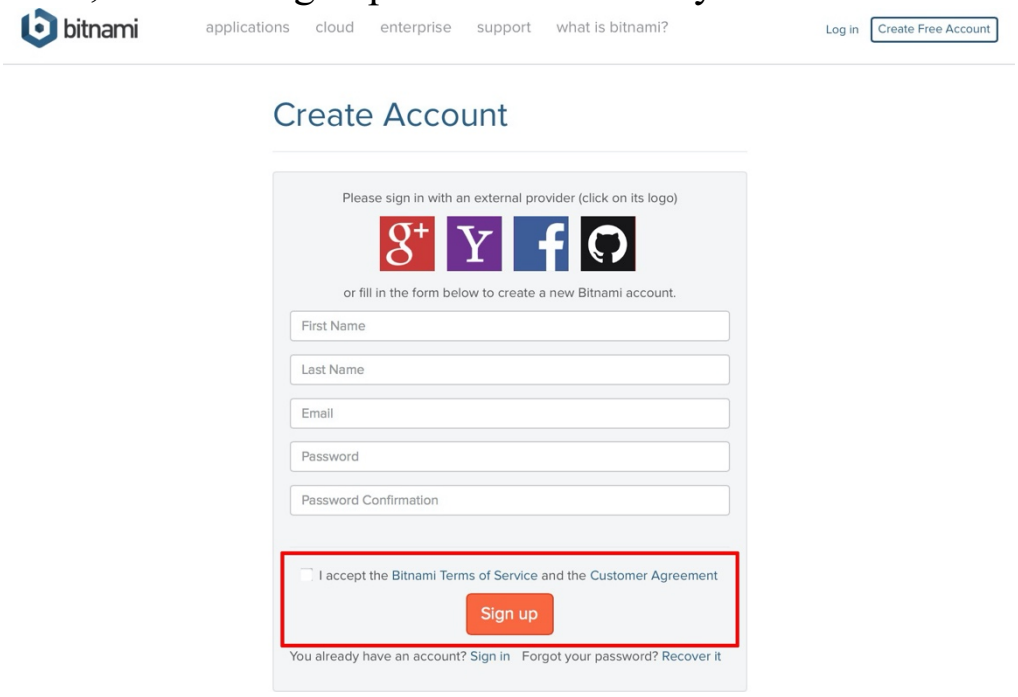
At the end of this step, you will have created a Bitnami account.

The next step is to create a Bitnami account, so that you can launch a cloud server with Bitnami's LAMP Stack image. If you have a Google, Facebook, Yahoo! or Github account, you can use your credentials from those services with OpenID to create your Bitnami account.

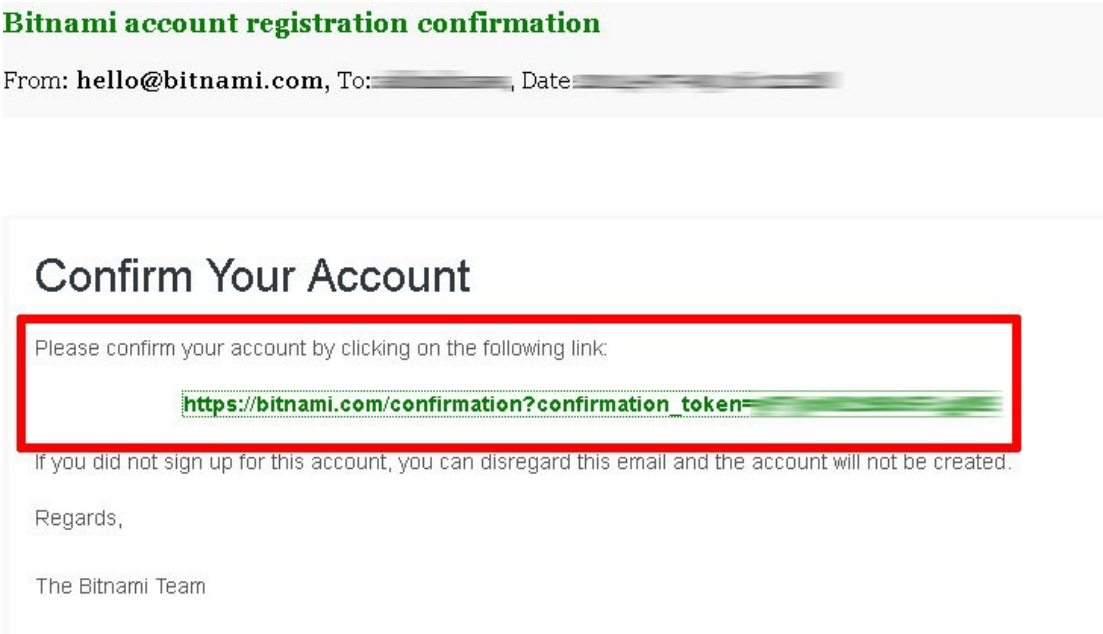
If you don't have accounts with those services (or you don't want to use them), you can use your email address and password to create a Bitnami account, as described below:

- Head to the Bitnami sign-up page.
- Enter your name and email address.
- Choose a password.
- Review and agree to the Bitnami terms of service.

Then, use the "Sign up" button to create your account.



Bitnami will send you an email with a verification link which you'll need to click or browse to, to activate your account. This will also sign you in to your Bitnami account.



Step 3: Connect your Google Cloud Platform and Bitnami Accounts

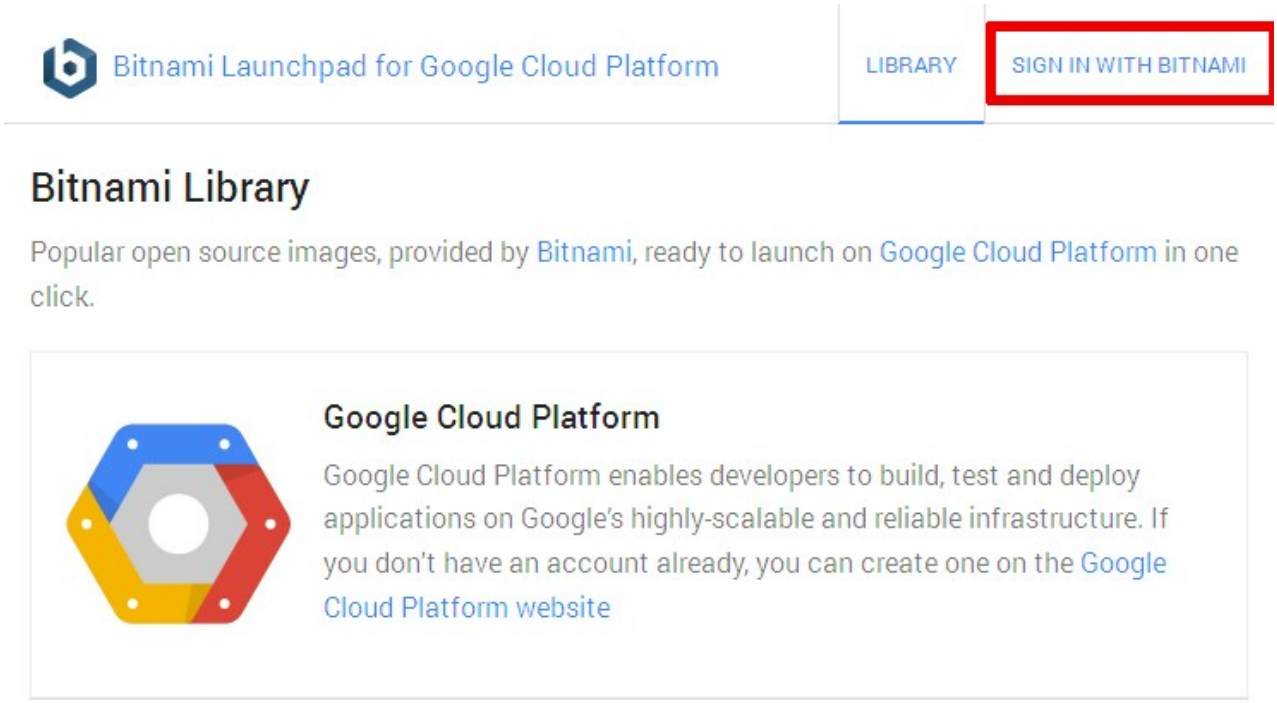
At the end of this step, your Bitnami Launchpad for Google Cloud Platform will be configured and you will be ready to provision a cloud server.

The easiest way to set up your Google cloud server with Bitnami's LAMP Stack is via the Bitnami Launchpad for Google Cloud Platform, which gives you a simple control panel to provision, start, stop, connect to and check status of your cloud servers. However, to use it, you must first connect your Google Cloud Platform and Bitnami accounts.

To do this:

- Log in to your Bitnami account if you're not already logged in.
- Browse to https://google.bitnami.com/.

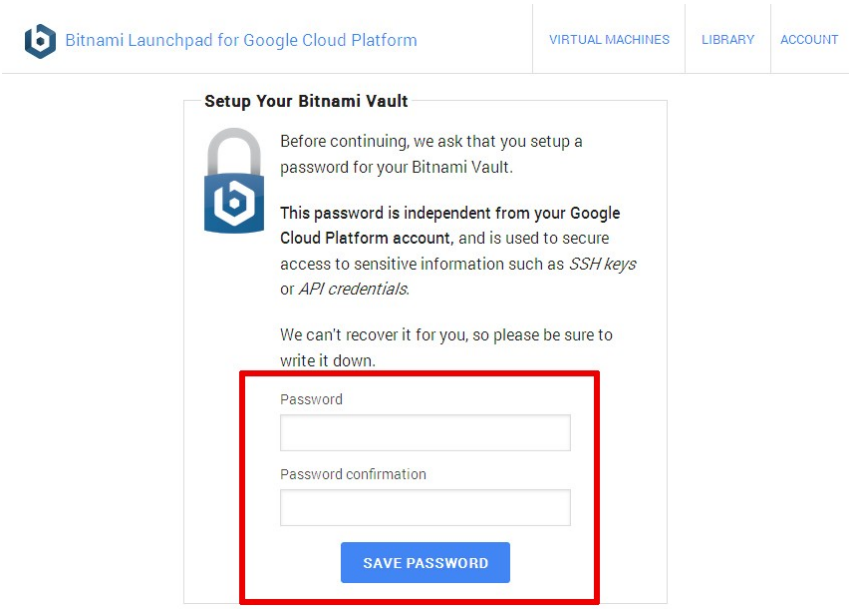
- Select the "Sign in with Bitnami" link in the top right corner.



The Launchpad will recognize your Bitnami credentials and automatically sign you in.

The next step is to set up an administrative password and connect your Google Cloud Platform account with your Bitnami account. To do this:

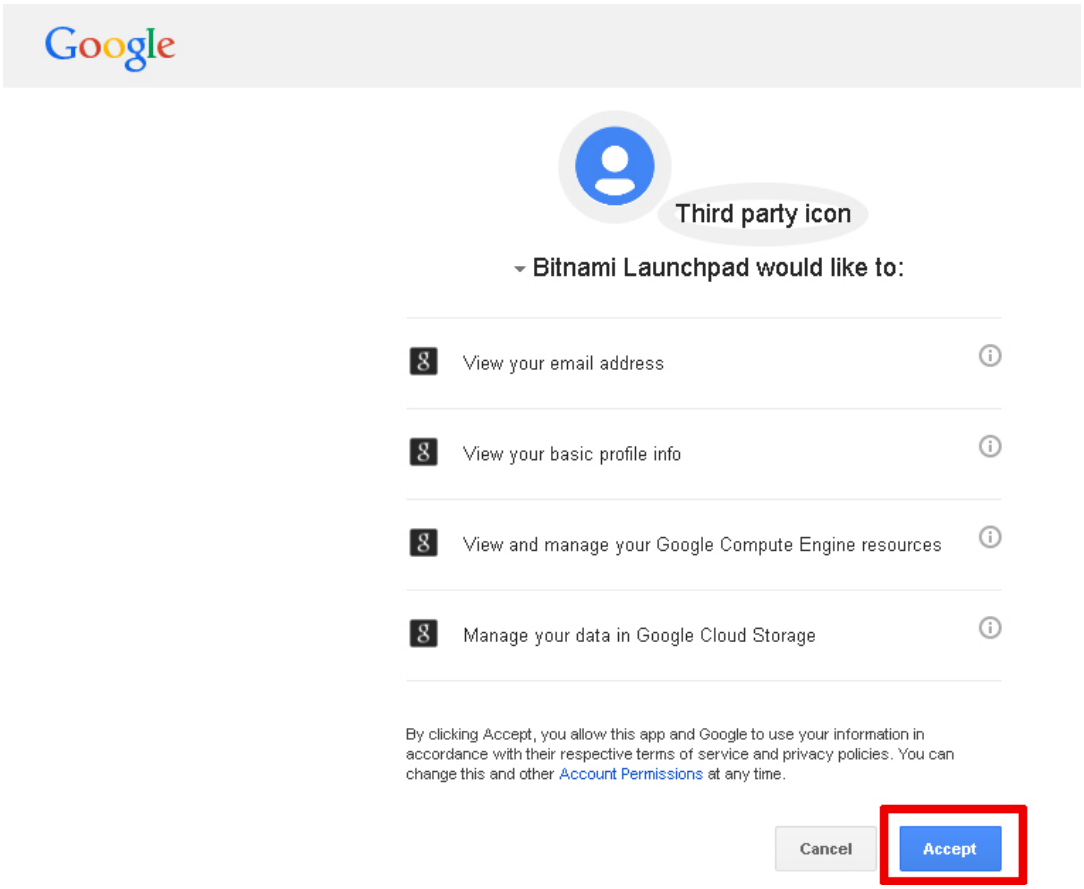
- Select "Virtual Machines" in the Launchpad menu.
- Since this is your first time, you'll be prompted to set up your Bitnami password vault by entering an administrative password. Enter a hard-to-guess password.



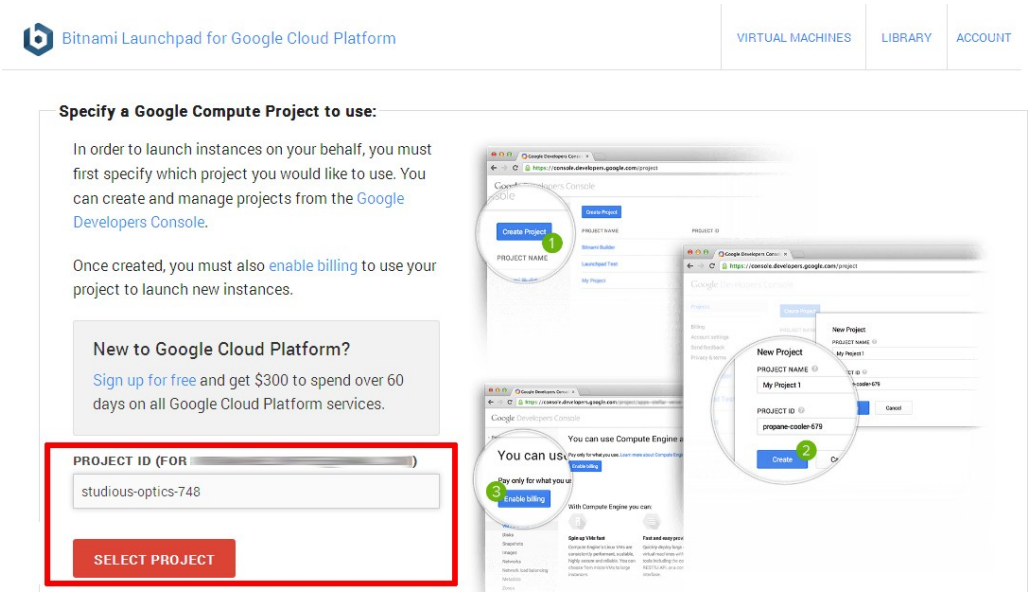
The Bitnami Vault password offers an additional level of protection against misuse: you'll need to enter it when performing certain operations, such as creating new cloud servers. Again, make sure you note it down for future reference.

IMPORTANT Your Bitnami Vault password is different from your Google Account password.

- Once your password has been accepted, you'll be redirected back to the Launchpad page. Select "Virtual Machines" again in the Launchpad menu.
- You'll be transferred to an authorization page, where you can confirm that the Bitnami Launchpad is authorized to connect to your Google Cloud Platform account. Click the "Accept" button on the page to proceed.



- You'll now be redirected back to the Bitnami Launchpad, and asked to select a project within which to launch new cloud servers. Enter the project ID you noted in Step 1.



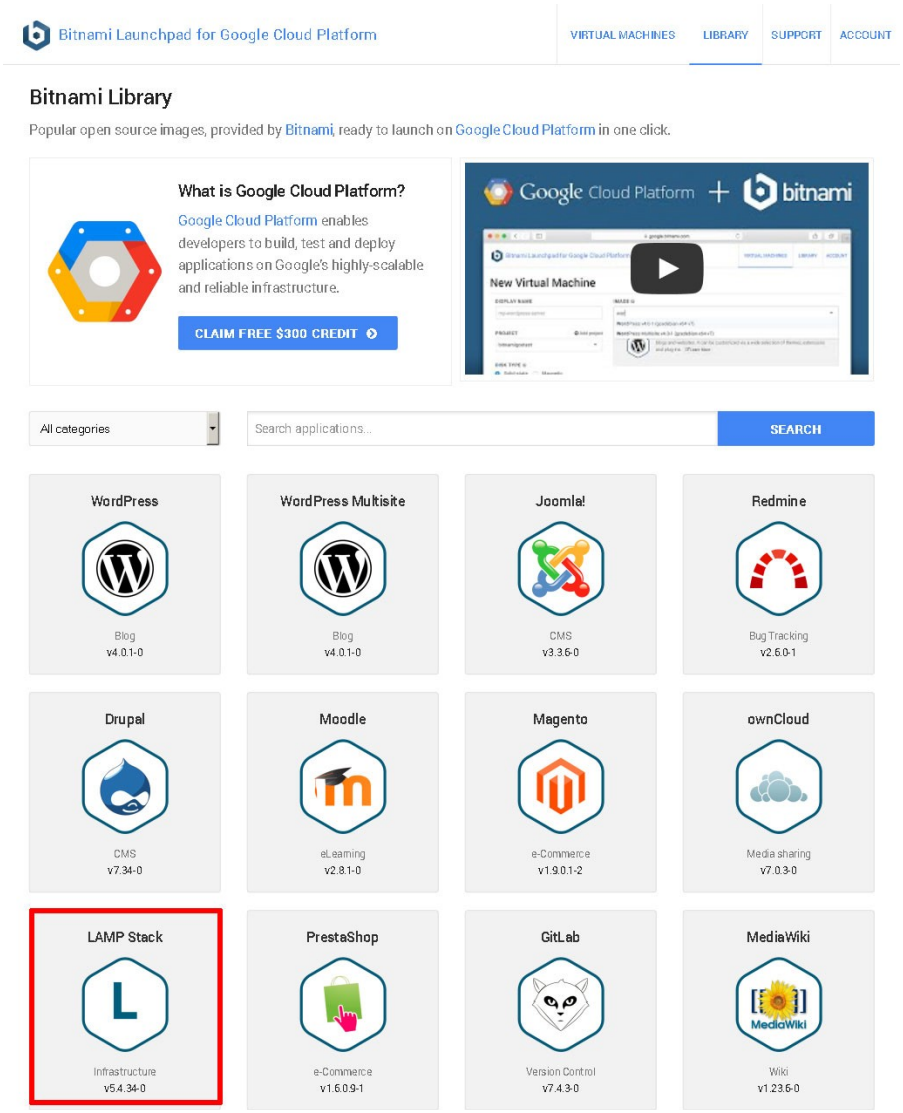
Your Google Cloud Platform and Bitnami accounts are now connected, and you can launch new cloud servers with Bitnami application stacks.

Step 4: Provision a Google Cloud Platform Server

At the end of this step, your Google Cloud Platform server will be running and you will be able to access it through your Web browser.

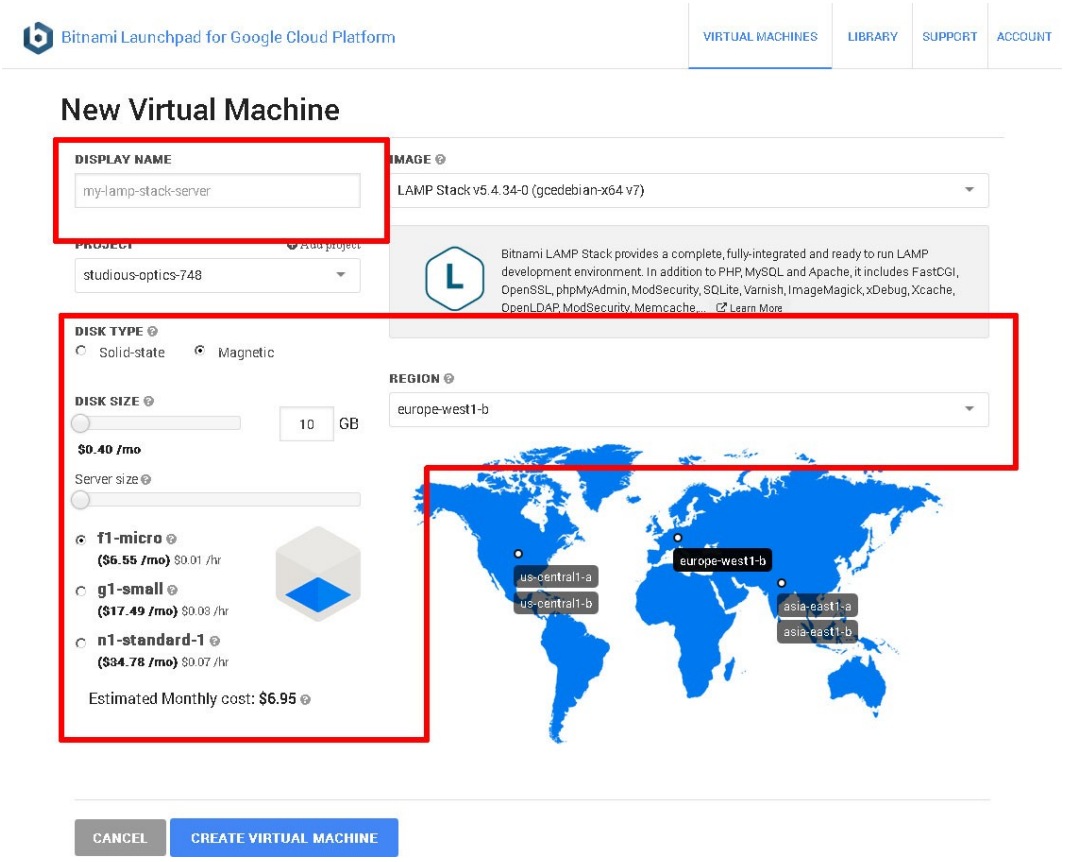
To provision your Google Cloud Platform server:

- Select "Library" in the Launchpad menu.
- Look through the list of applications available in Bitnami until you find the LAMP Stack. Select it and click "Launch". If required, enter your administrative password.



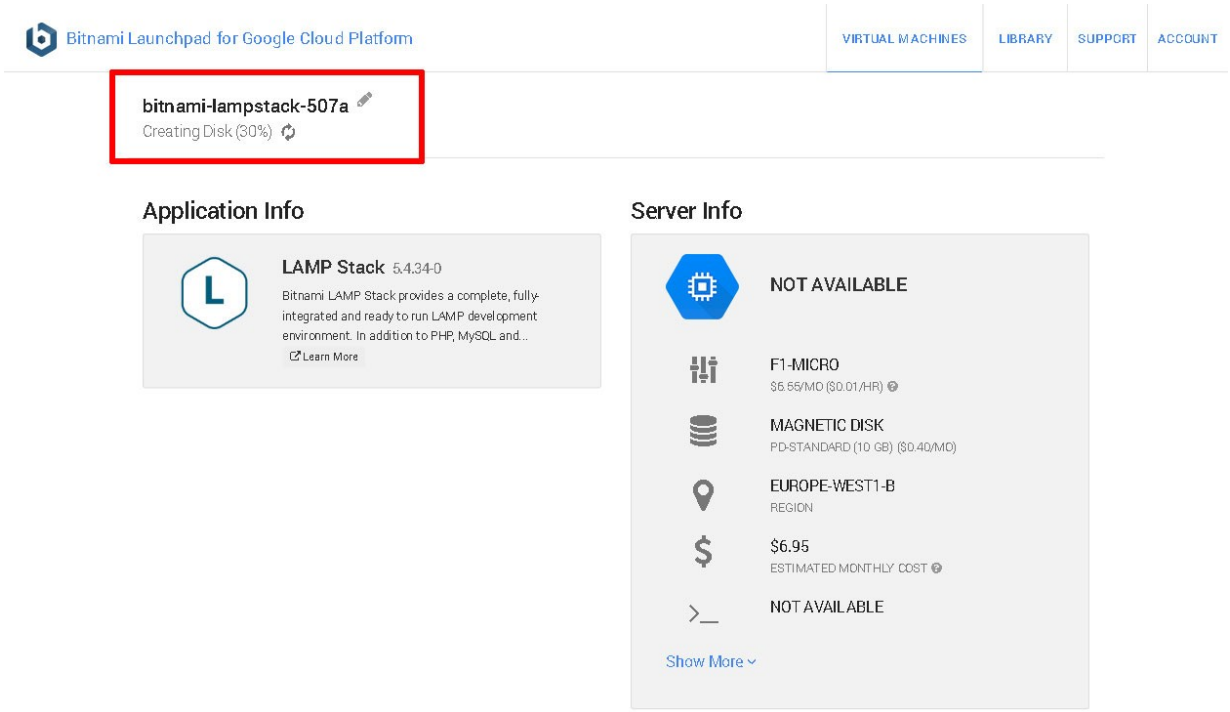
- Define a name, size and region for your cloud server. You can choose from a "micro" server, which uses a shared CPU to a "high CPU" server, which has 16 dedicated virtual cores. For more information, refer to the Google Compute Engine pricing sheet.

TIP A "micro" server will work just fine for most PHP application development tasks.

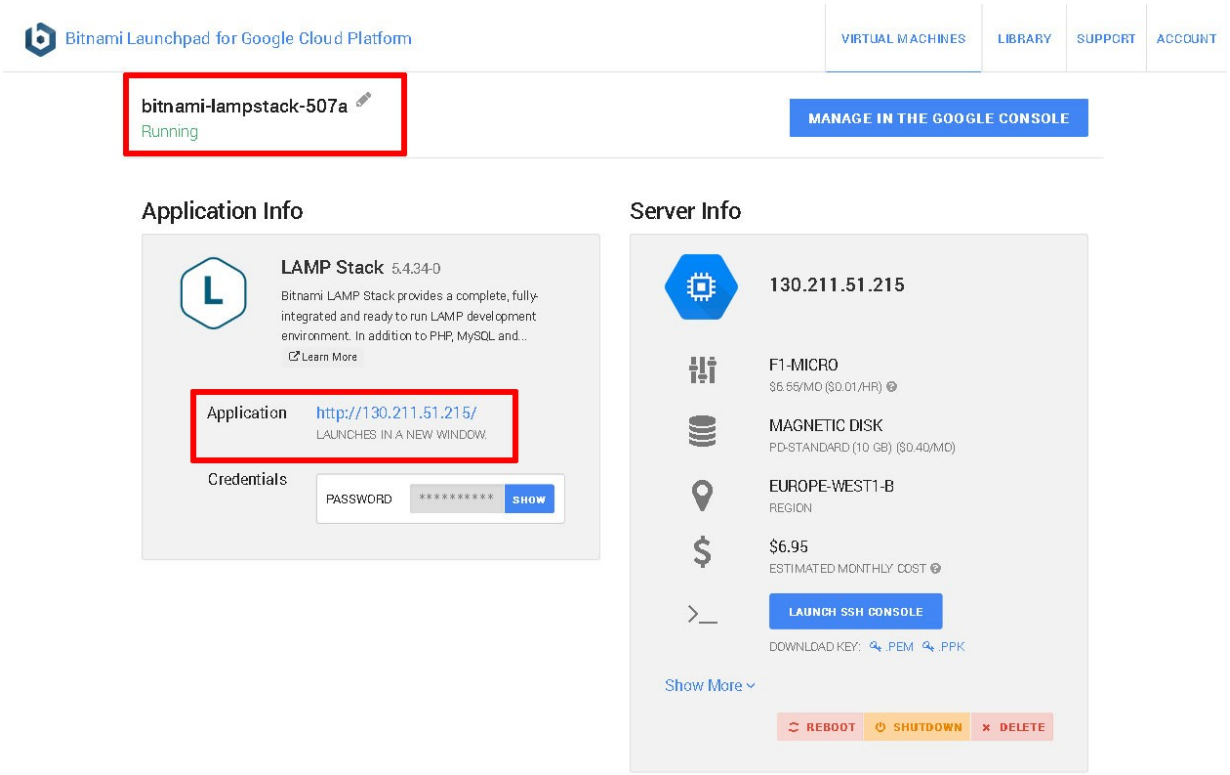


- Confirm your selection by hitting the "Create Virtual Machine" button at the end of the page.

The Bitnami Launchpad will now begin spinning up the new server. The process usually takes a few minutes: a status indicator on the page provides a progress update.

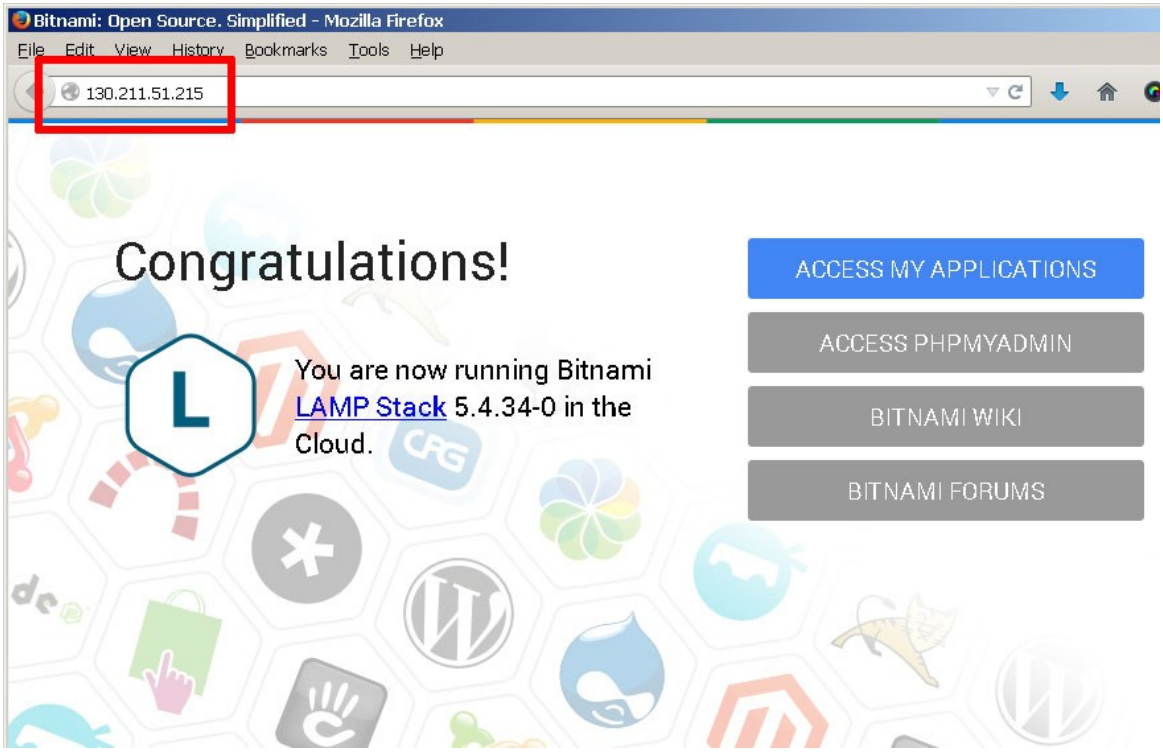


Once the cloud server has been provisioned, the status indicator will show that it's "running", and the Bitnami Launchpad page will display the server details, including its IP address.



At this point, you should be able to browse to the cloud server, either by clicking the link in the Bitnami Launchpad (a new browser tab will open) or entering the

cloud server IP address directly into your browser's address bar. You should see a welcome page like the one below (just so you know, it's served up by Apache, which is part of the Bitnami LAMP Stack).

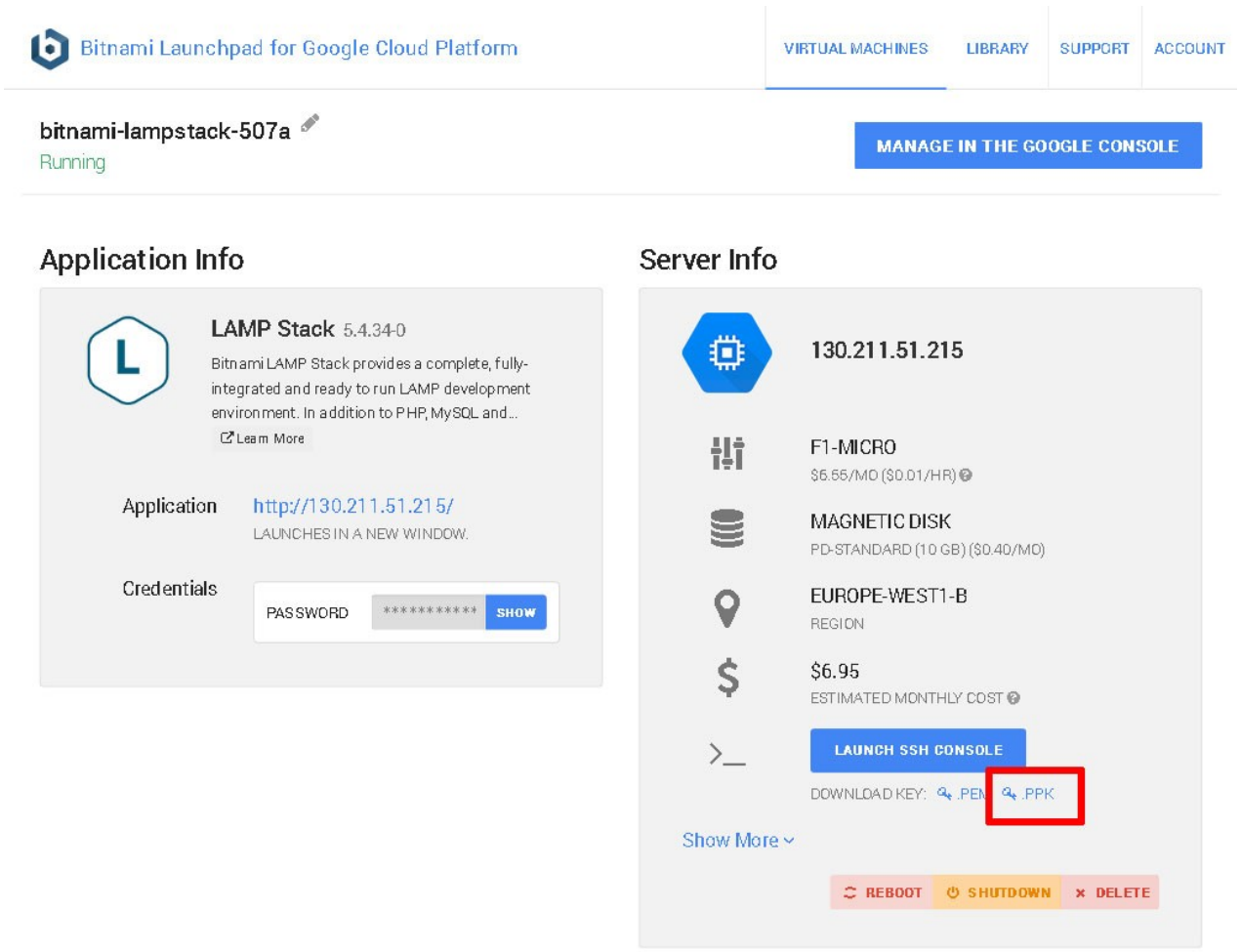


Once the server is provisioned, you need to gather the security credentials you will need to begin using it. To do this:

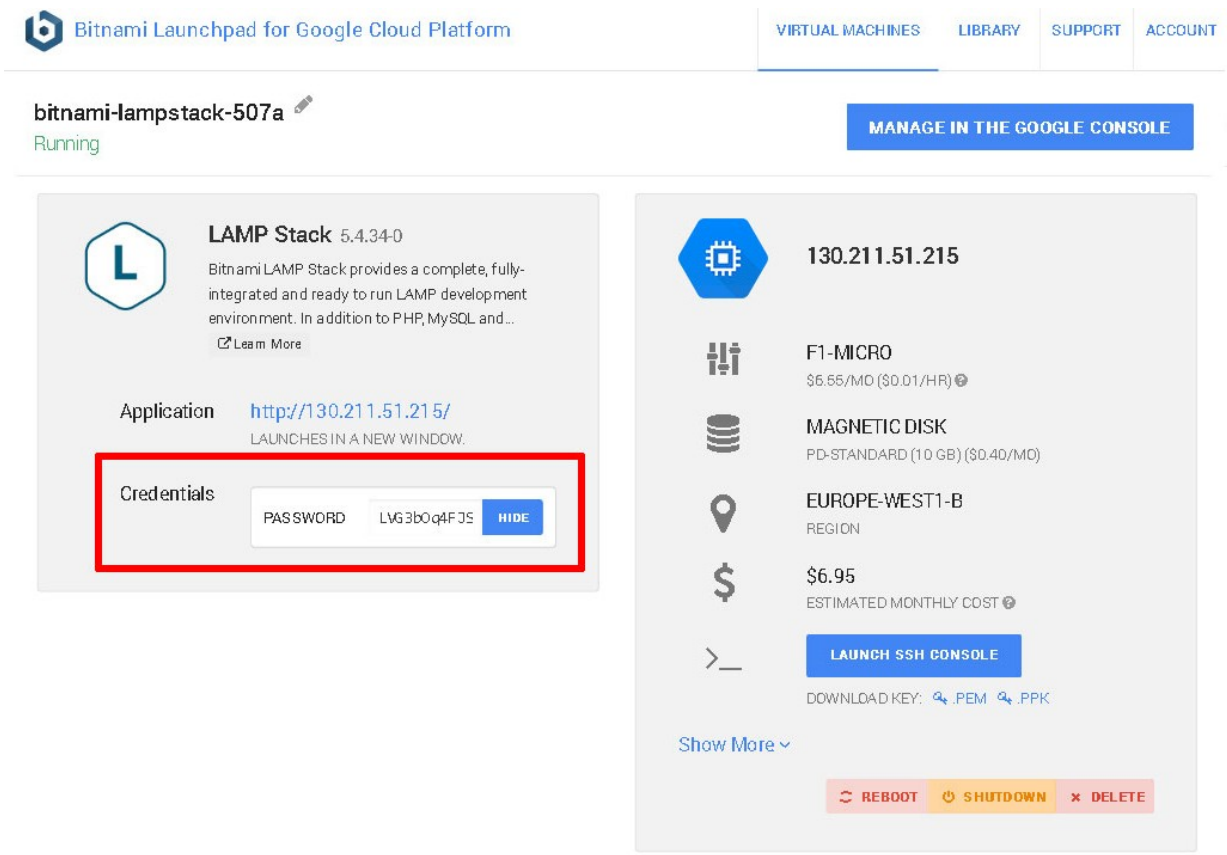
- Go back to the Bitnami Launchpad for Google Cloud Platform page and in the "Virtual Machines" section,

select the running server. This will launch the server information page.

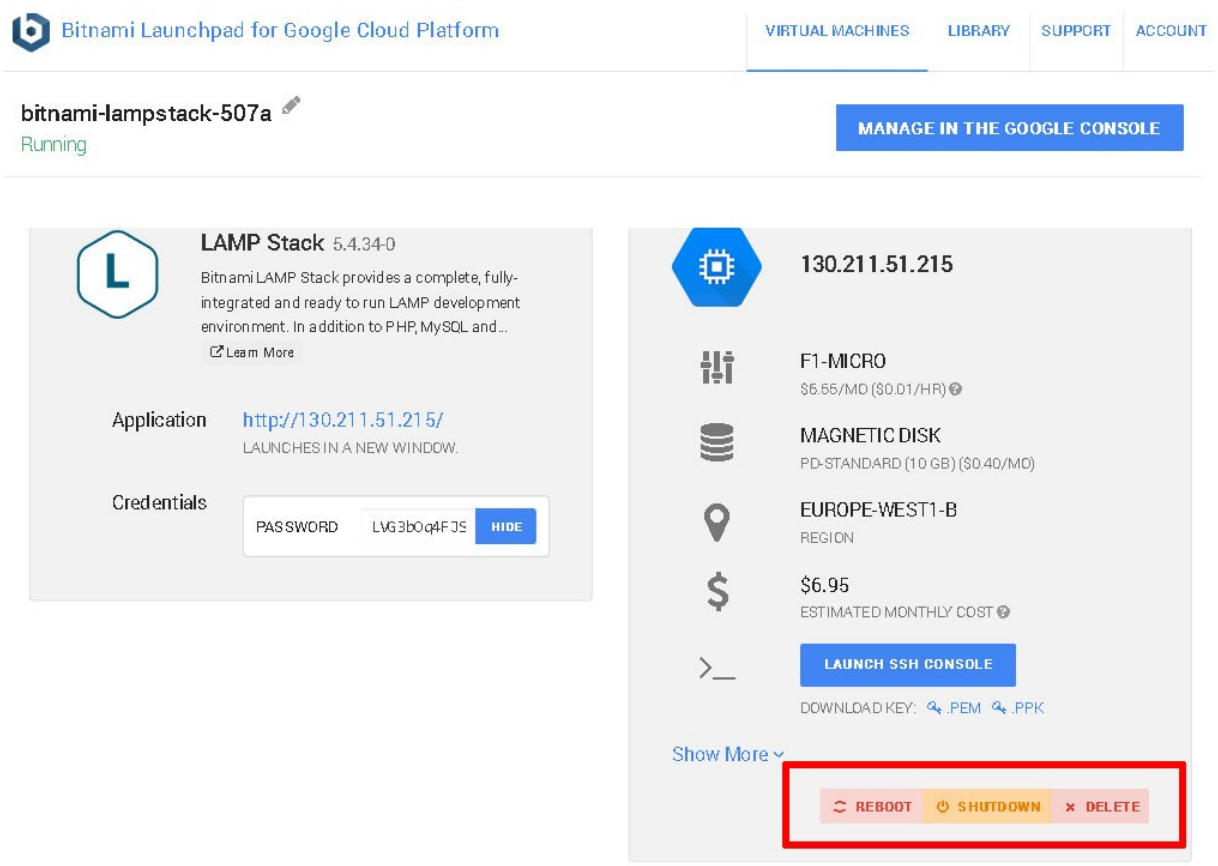
- From the server information page, download the *.ppk* file which contains the SSH access credentials you will need to connect to the server. Typically, this file is named using the format *bitnami-[google-project]- [nn].ppk*. If you're using Mac OS X or Linux, you should instead download the corresponding *.pem* file.



- By default, Bitnami Cloud Hosting creates a user account named 'user' and an auto-generated password when a new server is provisioned. You will need this password when accessing Bitnami-supplied applications (including MySQL). Go back to the server information screen, look in the "Credentials" section of the "Application Info" panel, and display and make a note of the application password.



The Launchpad page also includes controls to reboot, shut down or delete the server.



Step 5: Test PHP and MySQL

At the end of this step, you will have logged in to your cloud server and verified that PHP, MySQL and phpMyAdmin are working correctly.

You can now connect to the cloud server and test PHP to make sure it's working correctly and has all the extensions you need. The easiest way to do this is with PuTTY, a free SSH client for Windows and UNIX platforms.

Note: This section. walks you through PuTTY to ssh into our new server. However, you can also use your built in SSH terminal if you are a Mac OS X user. After downloading the key, navigate to that directory and type the following command:

chmod og-r <the-name-of-your-bitnami-key>.pem

Because SSH keys should be handled securely, you must remove permissions from the group and others. See the example output below:

```
jasmine@Jasmines-Air project % ls -al
total 24
drwxr-xr-x@  4 jasmine  staff   128 Nov 13 16:42 .
drwx------@ 34 jasmine  staff  1088 Nov 13 16:42 ..
-rw-r--r--@  1 jasmine  staff  6148 Nov 13 16:36 .DS_Store
-rw-r--r--@  1 jasmine  staff  1675 Nov 13 16:39 bitnami-google-inspiring-team-2
95204.pem
jasmine@Jasmines-Air project % chmod og-r bitnami-google-inspiring-team-295204.p
em
jasmine@Jasmines-Air project % ls -al
total 24
drwxr-xr-x@  4 jasmine  staff   128 Nov 13 16:42 .
drwx------@ 34 jasmine  staff  1088 Nov 13 16:42 ..
-rw-r--r--@  1 jasmine  staff  6148 Nov 13 16:36 .DS_Store
-rw-------@  1 jasmine  staff  1675 Nov 13 16:39 bitnami-google-inspiring-team-2
95204.pem
```

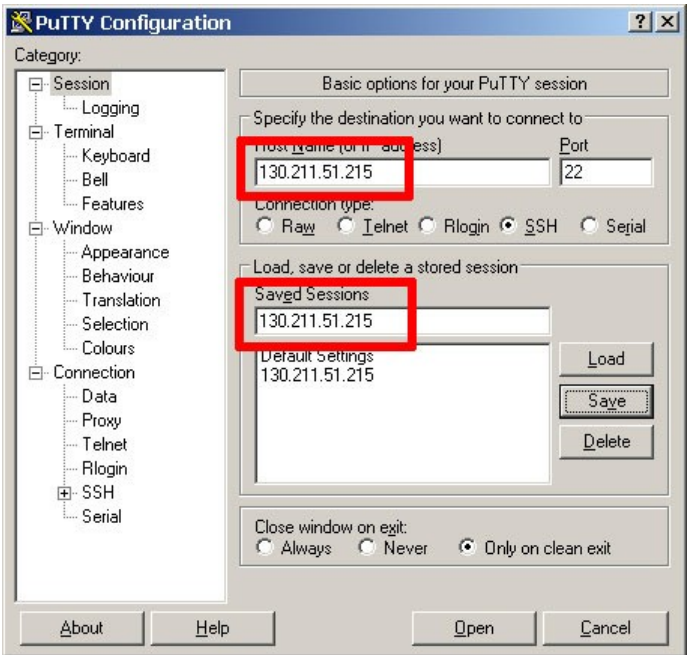
Now you will be able to SSH into your server. Type the following command to start a secure SSH session between your machine and the Bitnami server. You need to start an SSH session in order to navigate to your phpMyAdmin page:

ssh -N -L 8888:127.0.0.1:443 -i /<Your-Full-Path-To-Key>/<bitnami-key.pem>bitnami@<Bitnami's IP address>

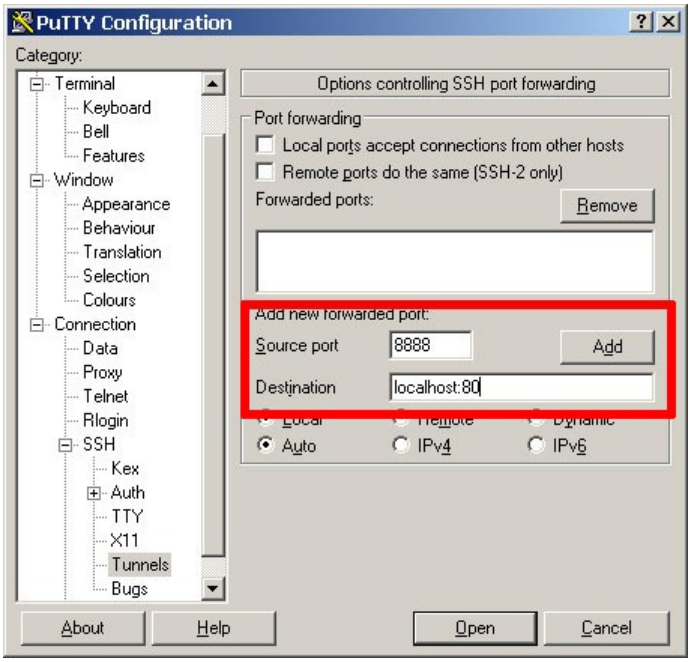
```
Last login: Fri Nov 20 14:25:43 on ttys000
jasmine@Jasmines-Air ~ % ssh -N -L 8888:127.0.0.1:443 -i /Users/jasmine/Documents/project/bitnami-google-inspiring-team-295204.pem
 bitnami@34.86.22.26_
```

When you first run this command, you may receive a warning about the server's credentials. Enter "Y" or "yes" to add these credentials as a trusted SSH connection. Your terminal will not output anything if the SSH connection is successful. You can now go to https://127.0.0.1:8888/phpmyadmin
The username is "root" and your password is the one provided on your Bitnami's instance main page where it says "Credentials". Continue this section if you would like to use PuTTY.

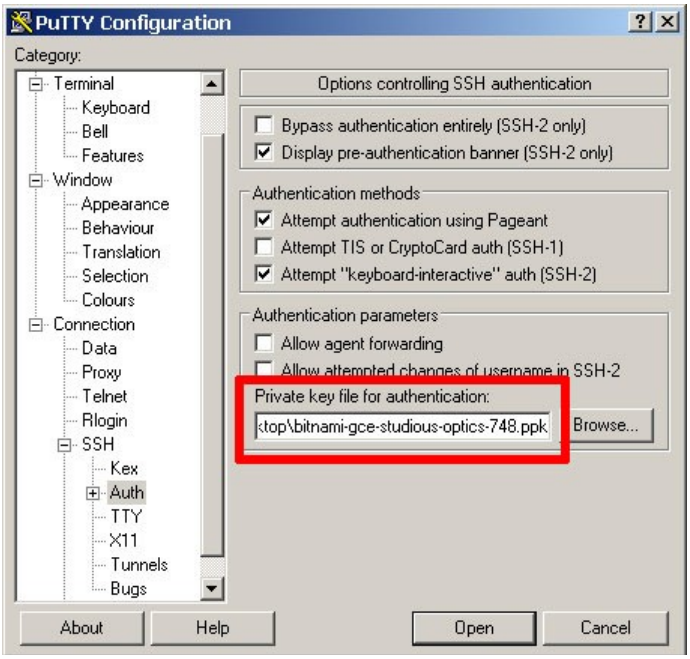
- Download the PuTTY ZIP archive from its website.
- Extract the contents to a folder on your desktop.
- Double-click the *putty.exe* file to bring up the PuTTY configuration window.
- Enter the host name of your cloud server into the "Host Name (or IP address)" field, as well as into the "Saved Sessions" field.
- Click "Save" to save the new session so you can reuse it later.



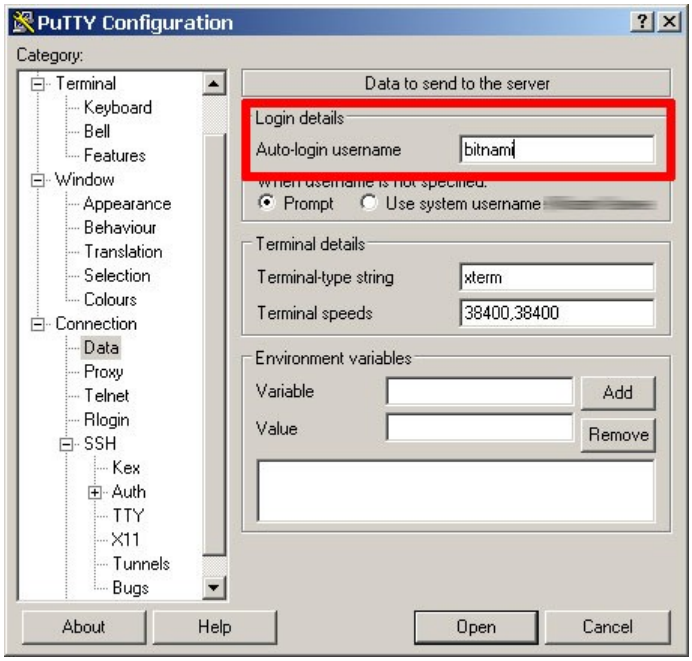
- In the "Connection _ SSH _ Tunnels" section, create a secure tunnel for the phpMyAdmin application by forwarding source port "8888" to destination port "localhost:80".
- Click the "Add" button to add the secure tunnel configuration to the session.



- In the "Connection _ SSH _ Auth" section, select the private key file (*.*ppk*) you saved in the previous step.



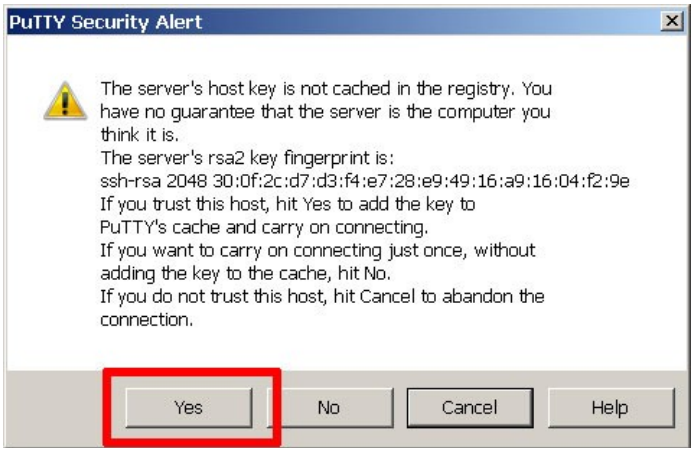
- In the "Connection _ Data" section, enter the username 'bitnami' into the "Auto-login username" field.



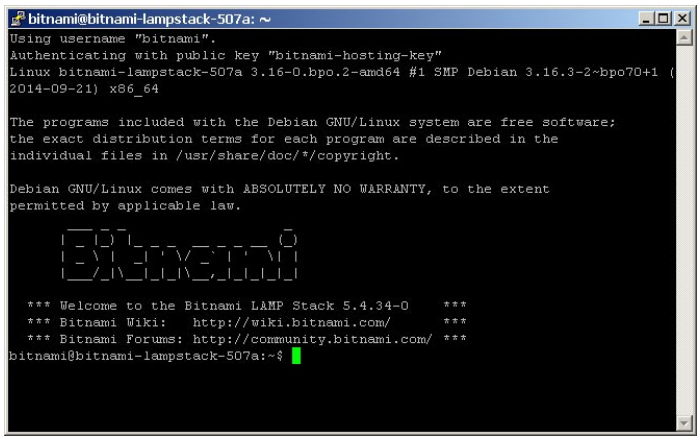
- Go back to the "Session" section and save your changes by clicking the "Save" button.

- Click the "Open" button to open an SSH session to the server.

- PuTTY will first ask you to confirm the server's host key and add it to the cache. Go ahead and click "Yes" to this request.



You should now be logged in to your cloud server.

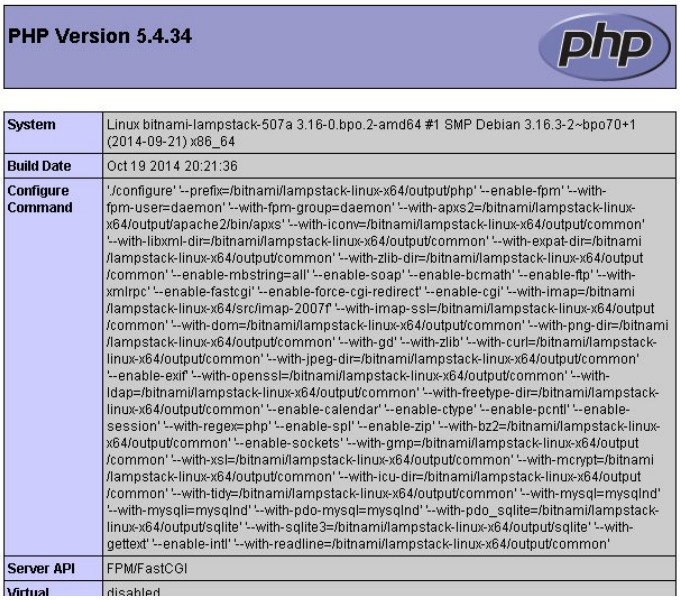


By default, the Bitnami LAMP stack includes running Apache and MySQL servers, and all the packages that come with the stack are located in the */opt/bitnami* directory. Your first step should be to create a *phpinfo.php* file in the Apache web server root at */opt/bitnami/apache2/htdocs* directory to verify PHP's capabilities.

Note: *You can skip these steps

```
shell> cd /opt/bitnami/apache2/htdocs
shell> echo "<?php phpinfo(); ?>" > phpinfo.php
```

*Once the file has been copied, browse to *http://[your-cloud-server-hostname]/phpinfo.php* and you should see the output of the *phpinfo()* command.



PHP Version 5.4.34

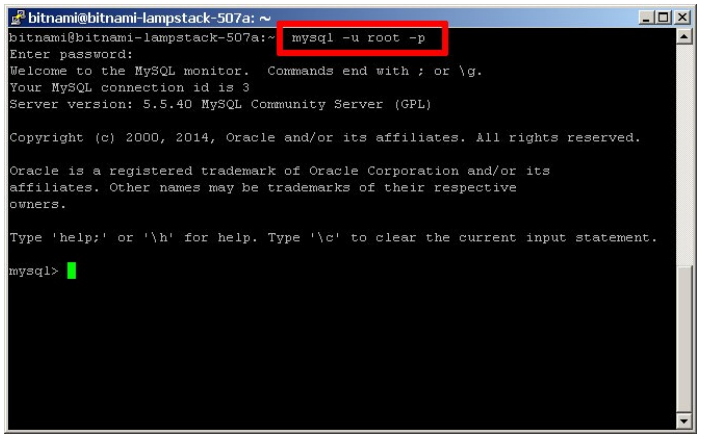
System	Linux bitnami-lampstack-507a 3.16-0.bpo.2-amd64 #1 SMP Debian 3.16.3-2~bpo70+1 (2014-09-21) x86_64
Build Date	Oct 19 2014 20:21:36
Configure Command	'./configure' '--prefix=/bitnami/lampstack-linux-x64/output/php' '--enable-fpm' '--with-fpm-user=daemon' '--with-fpm-group=daemon' '--with-apxs2=/bitnami/lampstack-linux-x64/output/apache2/bin/apxs' '--with-iconv=/bitnami/lampstack-linux-x64/output/common' '--with-libxml-dir=/bitnami/lampstack-linux-x64/output/common' '--with-expat-dir=/bitnami/lampstack-linux-x64/output/common' '--with-zlib-dir=/bitnami/lampstack-linux-x64/output/common' '--enable-mbstring=all' '--enable-soap' '--enable-bcmath' '--enable-ftp' '--with-xmlrpc' '--enable-fastcgi' '--enable-force-cgi-redirect' '--enable-cgi' '--with-imap=/bitnami/lampstack-linux-x64/src/imap-2007f' '--with-imap-ssl=/bitnami/lampstack-linux-x64/output/common' '--with-dom=/bitnami/lampstack-linux-x64/output/common' '--with-png-dir=/bitnami/lampstack-linux-x64/output/common' '--with-gd' '--with-zlib' '--with-curl=/bitnami/lampstack-linux-x64/output/common' '--with-jpeg-dir=/bitnami/lampstack-linux-x64/output/common' '--enable-exif' '--with-openssl=/bitnami/lampstack-linux-x64/output/common' '--with-ldap=/bitnami/lampstack-linux-x64/output/common' '--with-freetype-dir=/bitnami/lampstack-linux-x64/output/common' '--enable-calendar' '--enable-ctype' '--enable-pcntl' '--enable-session' '--with-regex=php' '--enable-spl' '--enable-zip' '--with-bz2=/bitnami/lampstack-linux-x64/output/common' '--enable-sockets' '--with-gmp=/bitnami/lampstack-linux-x64/output/common' '--with-xsl=/bitnami/lampstack-linux-x64/output/common' '--with-mcrypt=/bitnami/lampstack-linux-x64/output/common' '--with-icu-dir=/bitnami/lampstack-linux-x64/output/common' '--with-tidy=/bitnami/lampstack-linux-x64/output/common' '--with-mysql=mysqlnd' '--with-mysqli=mysqlnd' '--with-pdo-mysql=mysqlnd' '--with-pdo_sqlite=/bitnami/lampstack-linux-x64/output/sqlite' '--with-sqlite3=/bitnami/lampstack-linux-x64/output/sqlite' '--with-gettext' '--enable-intl' '--with-readline=/bitnami/lampstack-linux-x64/output/common'
Server API	FPM/FastCGI
Virtual	disabled

With this, you know that your PHP installation is configured and working correctly.

*You can also check that MySQL is working by launching the MySQL command-line client at the shell prompt.

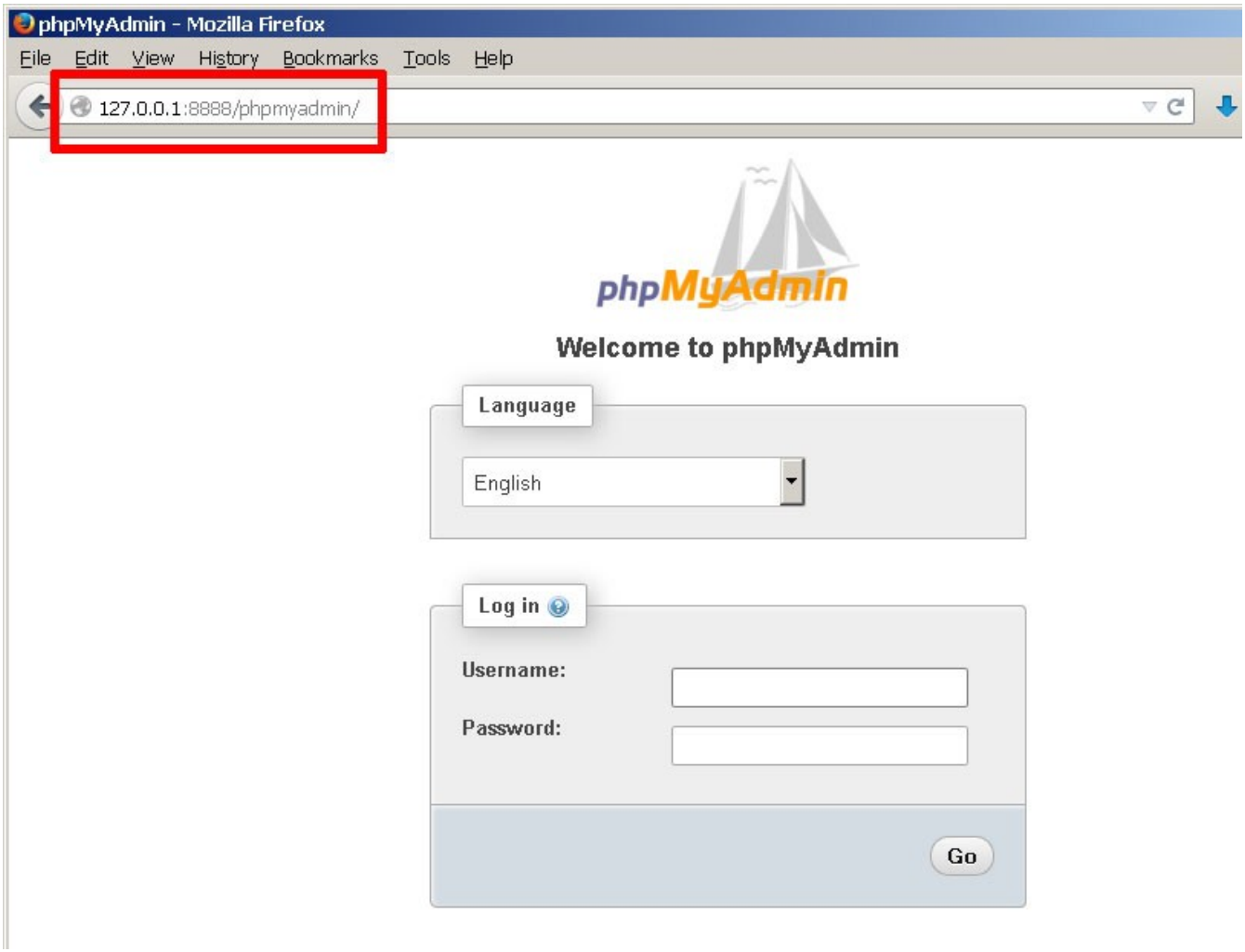
```
shell> mysql -u root -p
```

When prompted, enter the application password retrieved in the previous step. The client should start up and connect to the local MySQL server, displaying a welcome message as shown below.

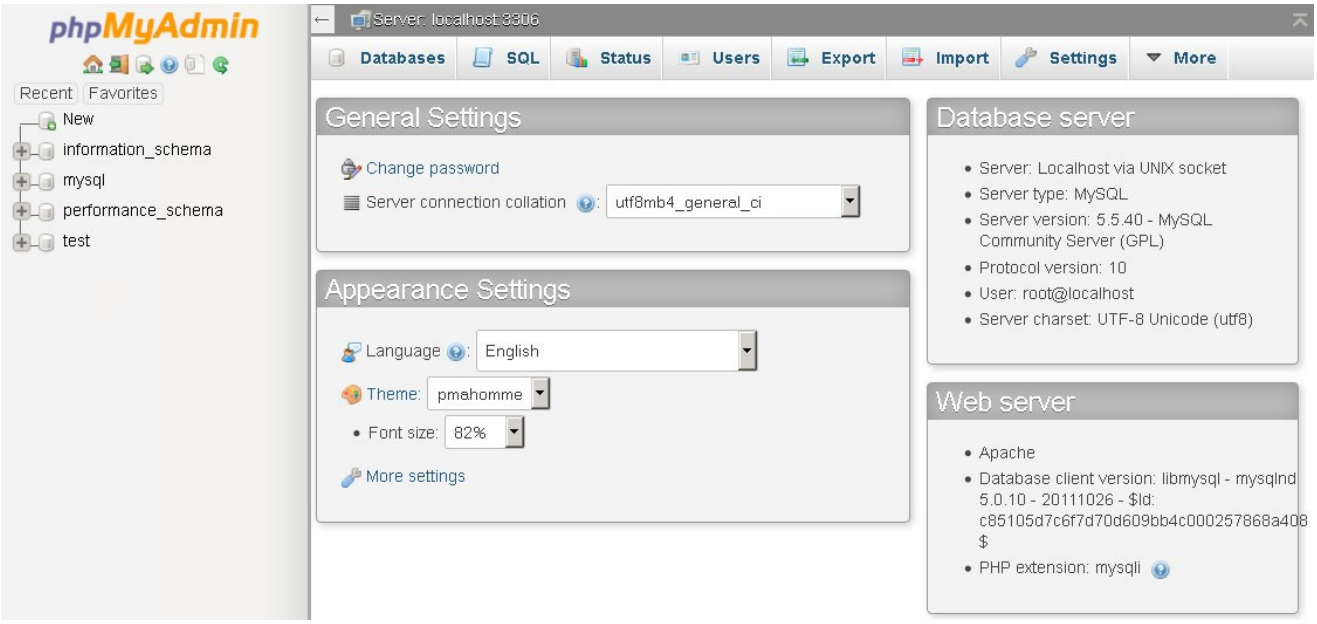


You should also be able to access phpMyAdmin through the secure SSH tunnel you created, by browsing to https://127.0.0.1:8888/phpmyadmin.

Note: The original guide has http://127.0.0.1:8888/phpmyadmin however, this will tell you it is a bad request. Please navigate to the secure version: https://127.0.0.1:8888/phpmyadmin



To log in, use username 'root' with the application password from the previous step.



In case you'd like to troubleshoot errors or modify the configuration for Apache, PHP or MySQL - for example, adjusting the maximum upload file size in PHP or changing the path to the MySQL data directory - here are the locations for key configuration and log files in the Bitnami LAMP Stack:

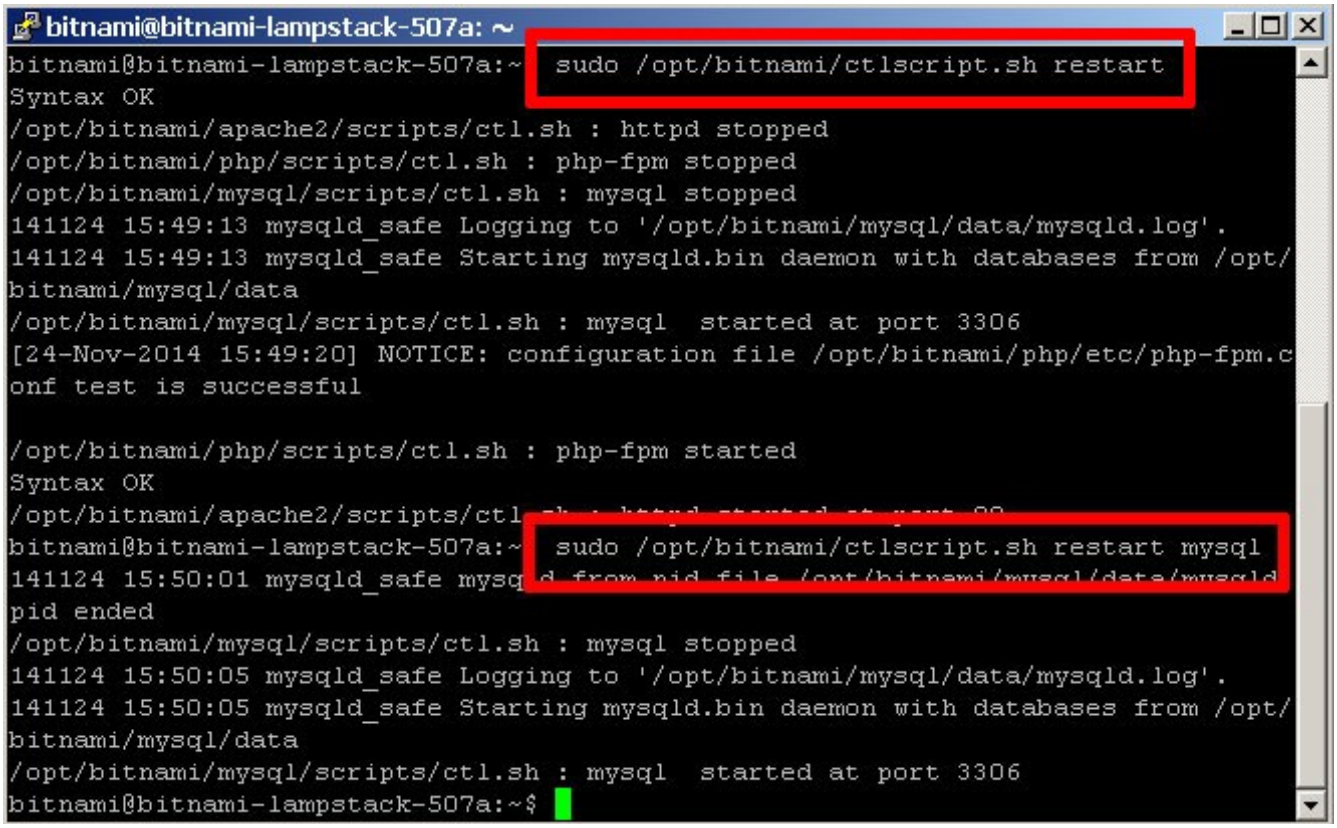
	Configuration file(s)	Log file(s)
Apache	*/opt/bitnami/apache2/conf/httpd.co nf*	*/opt/bitnami/apache2/logs/error_lo g*
PHP	*/opt/bitnami/php/etc/php.ini*	-
MySQL	*/opt/bitnami/mysql/my.cnf*	*/opt/bitnami/mysql/data/mysqld.log*

Usually, you'll need to restart your server(s) for your changes to take effect. The Bitnami LAMP Stack includes a control script that lets you easily stop, start and restart Apache, MySQL and PHP. The script is located at */opt/bitnami/ctlscript.sh*. Call it without any arguments to restart all services:

```
shell> sudo /opt/bitnami/ctlscript.sh restart
```

Or use it to restart a specific service only by passing the service name as argument - for example 'mysql':

```
shell> sudo /opt/bitnami/ctlscript.sh restart mysql
```



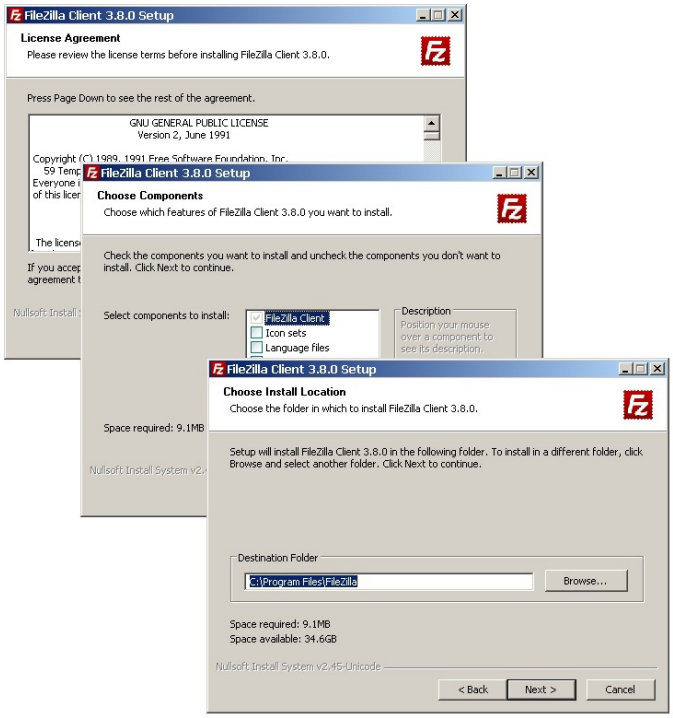
Step 6: Deploy the XAMPP Application to the Cloud Server

At the end of this step, your PHP/MySQL application will be running in the cloud.

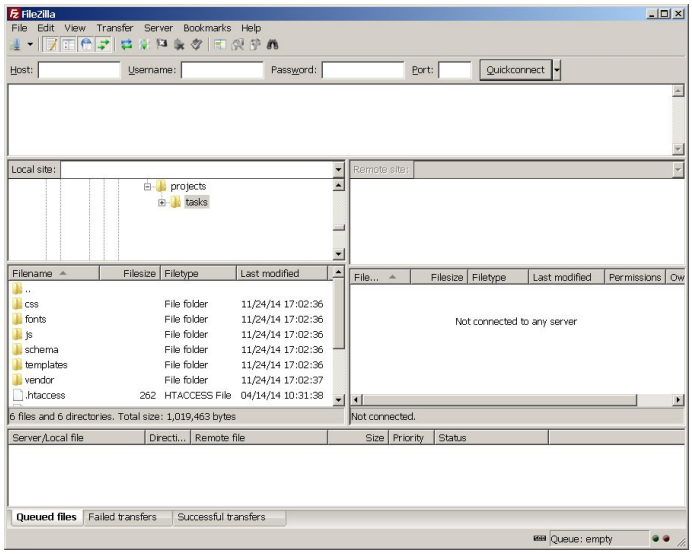
Your cloud server is now provisioned, secured and has a functional PHP/MySQL environment. All that's left is for you to transfer your application code from your local XAMPP environment to your cloud server and set up the database.

The easiest way to transfer files to the server is with FTP or SFTP. Although you can use any FTP/SFTP client, I like FileZilla, a cross-platform, open source and feature-rich client. Download it from the FileZilla website and install it using the automated installer - it's a quick process, only requiring you to agree to the license, choose the components (the default selection is usually fine) and specify the installation directory.

Note: You should already have FileZilla downloaded from section 1!


FileZilla Client 3.8.0 Setup
License Agreement
Please review the license terms before installing FileZilla Client 3.8.0.
Press Page Down to see the rest of the agreement.
GNU GENERAL PUBLIC LICENSE
Version 2, June 1991
FileZilla Client 3.8.0 Setup
Choose Components
Choose which features of FileZilla Client 3.8.0 you want to install.
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.
Select components to install:
FileZilla Client
Icon sets
Language files
Description
Position your mouse over a component to see its description.
Space required: 9.1MB
FileZilla Client 3.8.0 Setup
Choose Install Location
Choose the folder in which to install FileZilla Client 3.8.0.
Setup will install FileZilla Client 3.8.0 in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.
Destination Folder
C:\Program Files\FileZilla
Browse...
Space required: 9.1MB
Space available: 34.6GB
Nullsoft Install System v2.45-Unicode
< Back
Next >
Cancel

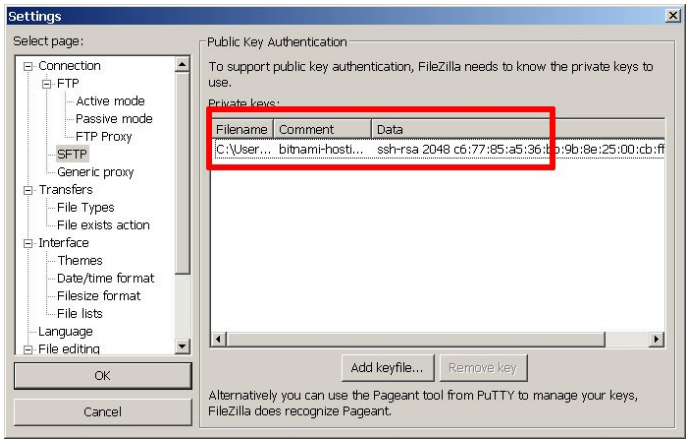
Once FileZilla is installed, launch it and you'll arrive at the main split-screen interface, one side for your local directories and the other for remote directories.



To connect to the cloud server and deploy your application, follow these steps:

- Use the "Edit _ Settings" command to bring up FileZilla's configuration settings.

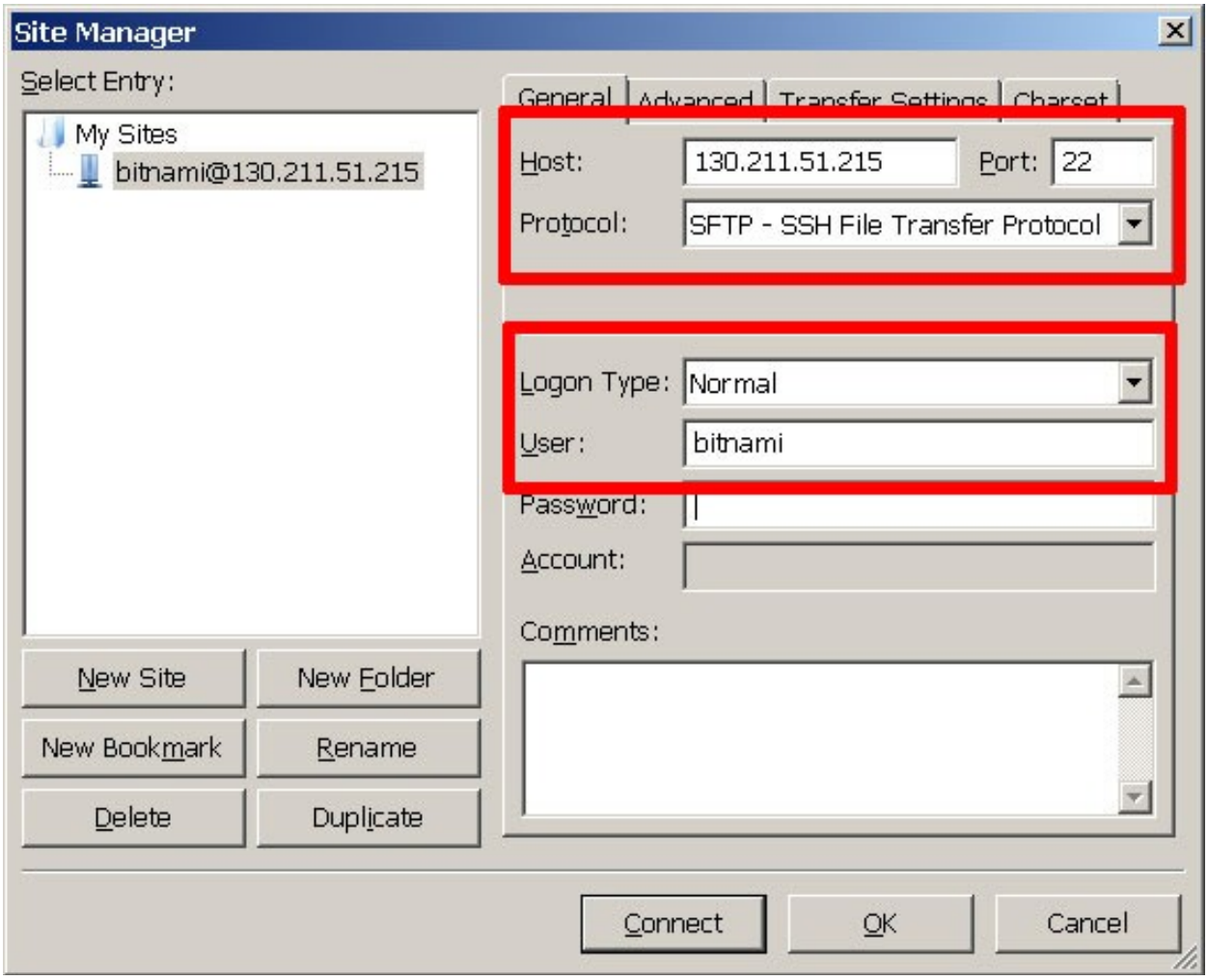
- Within the "Connection _ SFTP" section, use the "Add keyfile" command to select the private key file for your server. FileZilla will use this private key to log in to the cloud server.



- Use the "File _ Site Manager _ New Site" command to bring up the FileZilla Site Manager, where you can set up a connection to your cloud server.

- Enter your server host name or IP address and user name.

- Select "SFTP" as the protocol and "Normal" as the logon type.



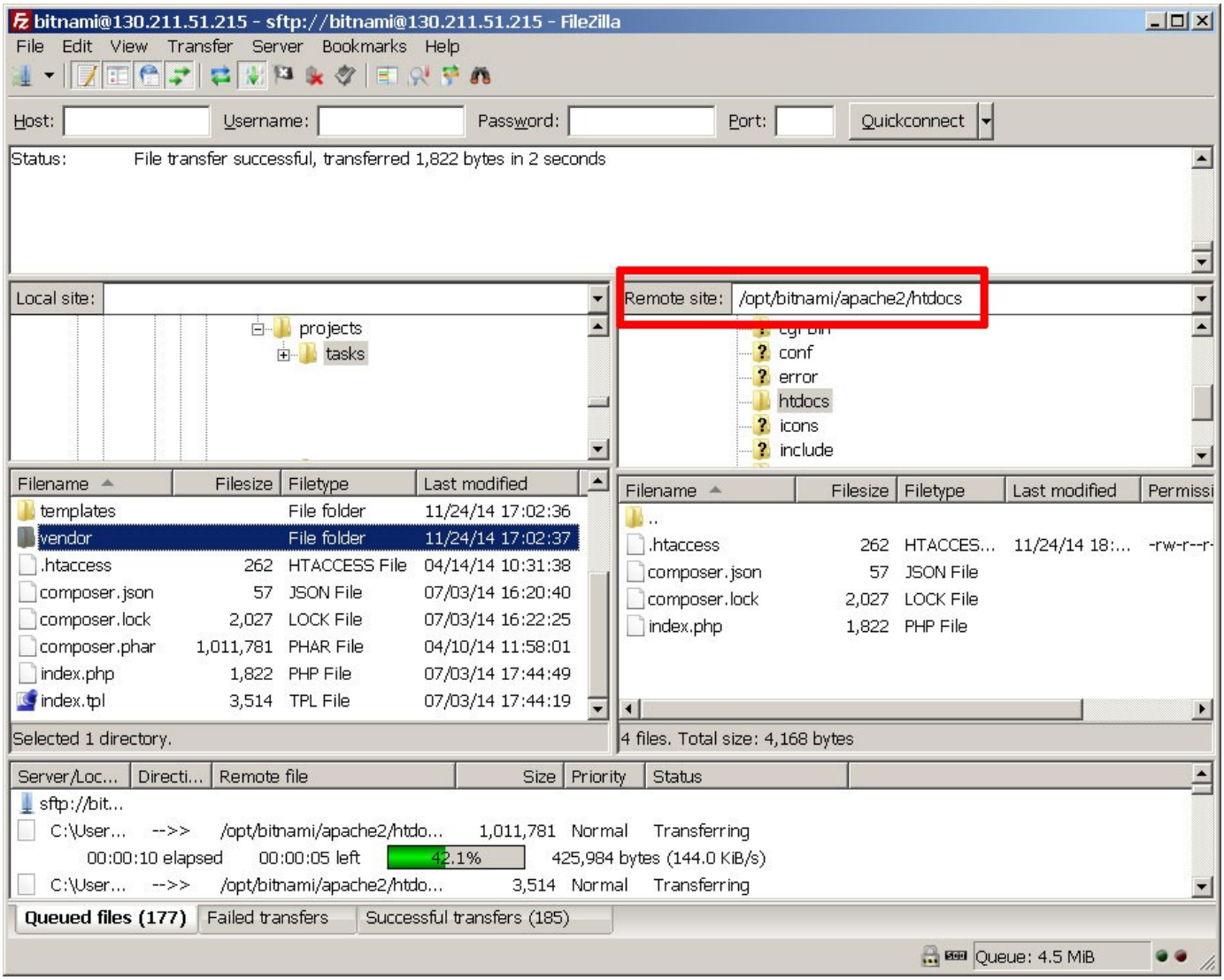
- Use the "Connect" button to connect to the cloud server and begin an SFTP session.

- On the remote server side of the window, change to the */opt/bitnami/apache2/htdocs* directory

- On the local server side of the window, change to the directory containing your application code.

Note: Make sure you update your 2FA OTP files to make sure the database host and password are configured to your Bitnami's credentials. The database name can still be whatever you name it but the password must be the one provided from Bitnami that was used to access phpMyAdmin!

- Upload your XAMPP application code to the remote directory by dragging and dropping the files from the local server to the cloud server (you can back up the original contents of the directory if you wish, by downloading them first).



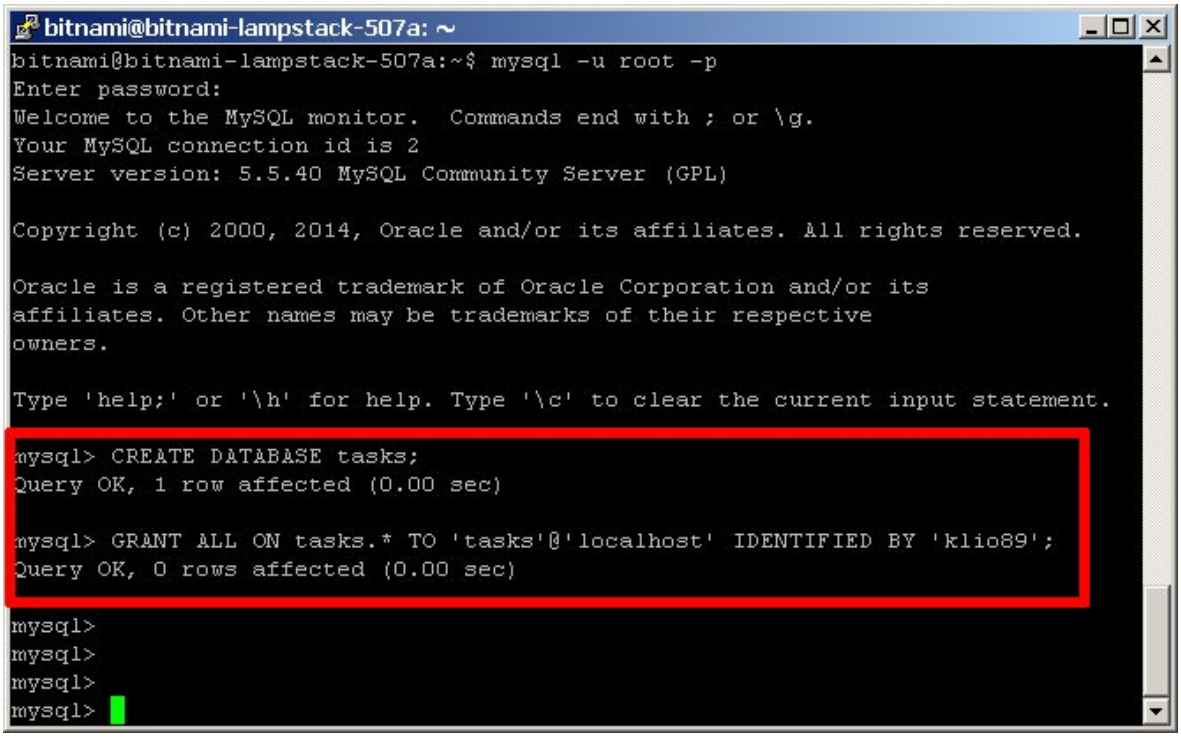
- Once the files are transferred, log in to the server console using PuTTY.
- Create a database for the application using the MySQL command-line client (you can use phpMyAdmin if you prefer a graphical interface). For example, since the application is a to-do list, let's call the database 'tasks'.

Note: Instead of using the MySQL command-line client, you can import the "accounts.sql" file as we did earlier in section 1.7. This guide talks about this method on page 37.

```
mysql> CREATE DATABASE tasks;
```

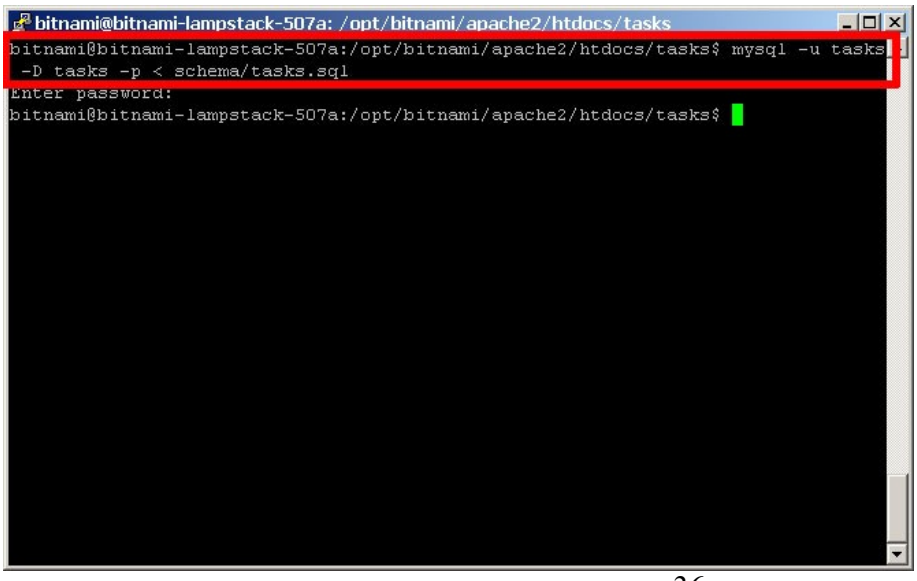
- Follow best practices and create a separate MySQL user with privileges to access only this database.

```
mysql> GRANT ALL ON tasks.* TO 'tasks'@'localhost' IDENTIFIED BY 'klio89';
```

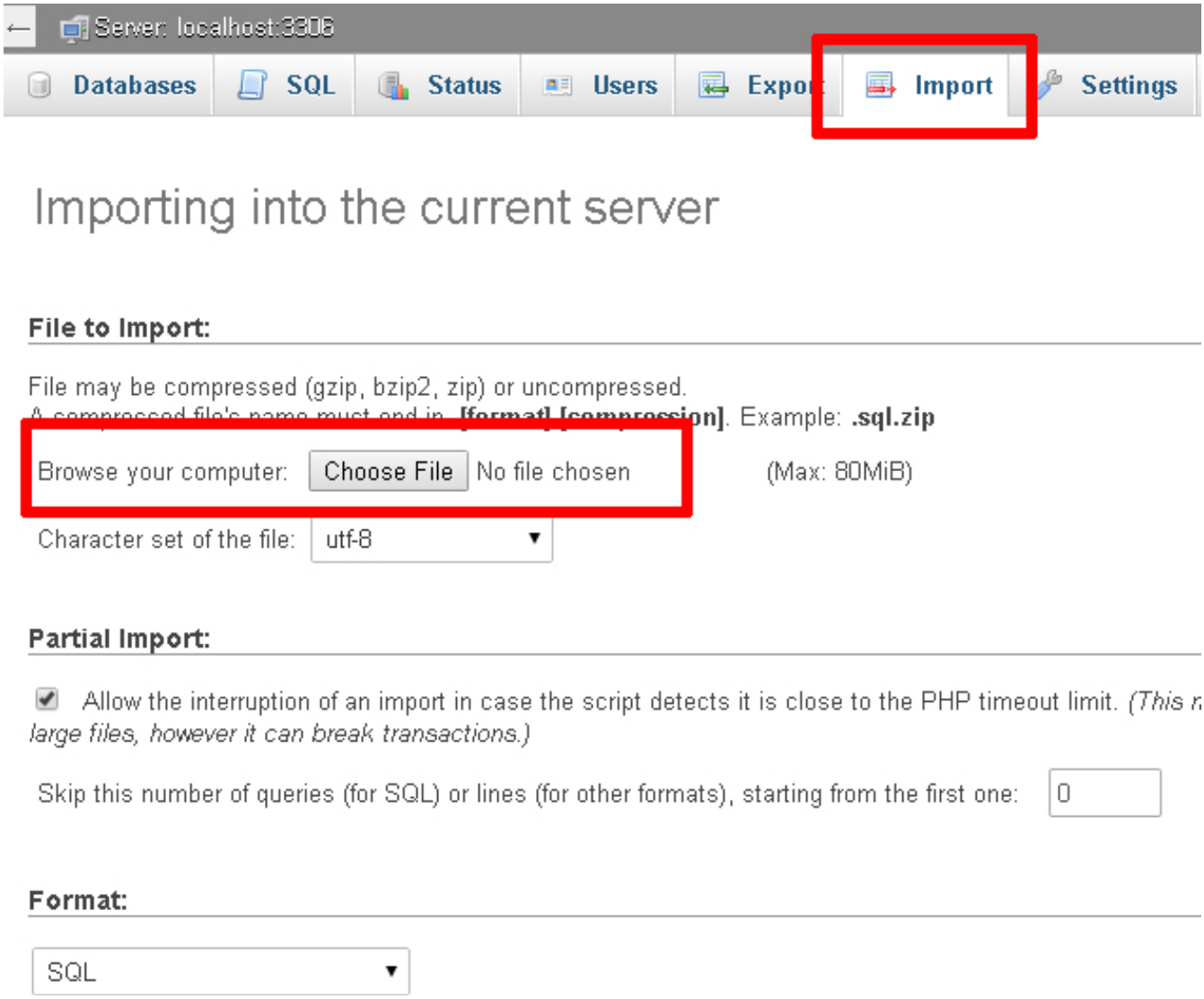


- If required, update database credentials in your application. Then, install the application schema in the new database (assuming you already uploaded it with the application code). For example, you can use the following command with the MySQL command-line client:

```
shell> mysql -u tasks -D tasks -p < schema/tasks.sql
```

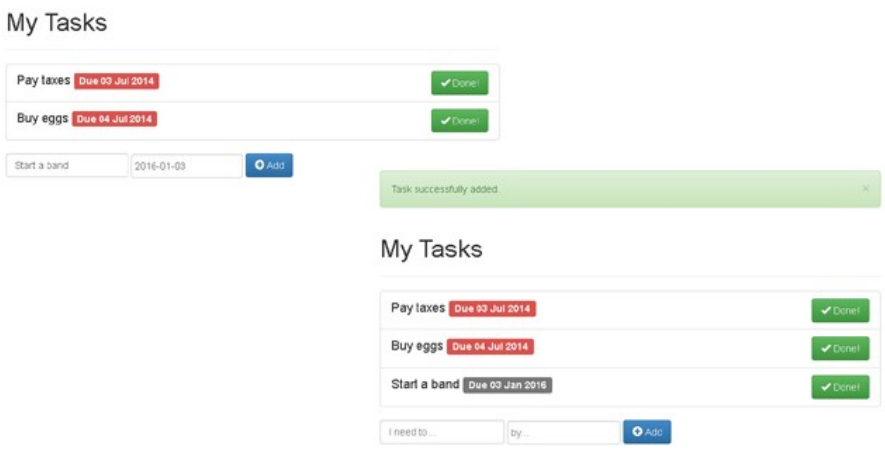


If you’re logged in to phpMyAdmin, you can also import the database schema from your local XAMPP system. To do this, select the "Import" tab of the phpMyAdmin dashboard, select the file containing the schema, and click "Go" to have the tables created in your selected database.



You can also learn more about using phpMyAdmin to back up and restore databases.

Browse to your cloud server's host name and your application should be active. Here are a few screenshots of the example to-do list application running on the cloud server.



Congratulations! You've successfully deployed your XAMPP application in the cloud.

Improve Application Performance

Web application performance problems are hard to debug at the best of times, and more so when your server is in the cloud and running a pre-packaged stack. The responsiveness of your application at any given moment depends on numerous factors: server type, network bandwidth, cloud provider load, database load, caching system in use, application code structure, query structure and various other variables.

IMPORTANT

The Bitnami LAMP Stack already uses the Apache Event MPM and PHP-FPM for reduced memory usage and an increase in the number of simultaneous requests that the server can handle (more information). It also comes with the mod_pagespeed Apache module activated to rewrite pages on the fly and improve latency.

If you're finding that your PHP/MySQL application's performance is not up to scratch, here are a few general

tips you can consider:

- The Bitnami LAMP Stack includes APC, a popular PHP bytecode cache. Usually, when a PHP script is executed, the PHP compiler converts the script to opcodes and then executes the opcodes. APC provides a framework for opcode caching, thereby speeding up PHP applications without needing any code changes. Make sure your APC cache has enough memory and a long TTL. Read more about APC and how to use APC with PHP and Bitnami.
- The Bitnami LAMP Stack also includes the PHP memcache extension. Memcache is a high-performance, distributed memory object caching system. Consider using memcache to store frequently-accessed fragments of data in memory as arrays, thereby reducing the load on your MySQL database server. Read more about memcache in PHP and how to use memcache with PHP and Bitnami.
- Turn on MySQL's slow query log and set MySQL's 'long_query_time' variable to a low number. This lets you track which of your queries are performing inefficiently and adjust them, either structurally or by applying table indexes as needed, to improve performance. You can use tools like mysqldumpslow or mysql-slow-query-log-visualizer to parse and analyze the slow query logs generated.
- If your application is database-heavy, you'll gain performance by giving the MySQL server more memory. Use a tool like MySQLTuner to identify which server parameters need tuning, and incrementally make changes to your server's cache and buffers to improve performance. If your tables are all MyISAM, disable InnoDB in your *my.cnf* file to save further memory.

- Unload Apache modules which you don't need to save memory, and adjust the log level to errors only.
- Minify your JavaScript code, and consider using a CDN for static content like images.

Good luck, and happy coding!

Useful Links

- Google Cloud Platform
- Bitnami Launchpad for Google Cloud Platform
- Bitnami LAMP Stack
- Bitnami AMP Stack documentation
- PuTTY
- FileZilla
- Example Project (.zip)

About the author

Vikram Vaswani is the founder of Melonfire, an open source software consultancy firm, and the author of seven books on PHP, MySQL and XML development. Read more about him at http://vikram-vaswani.in/.

Application Results

index.php

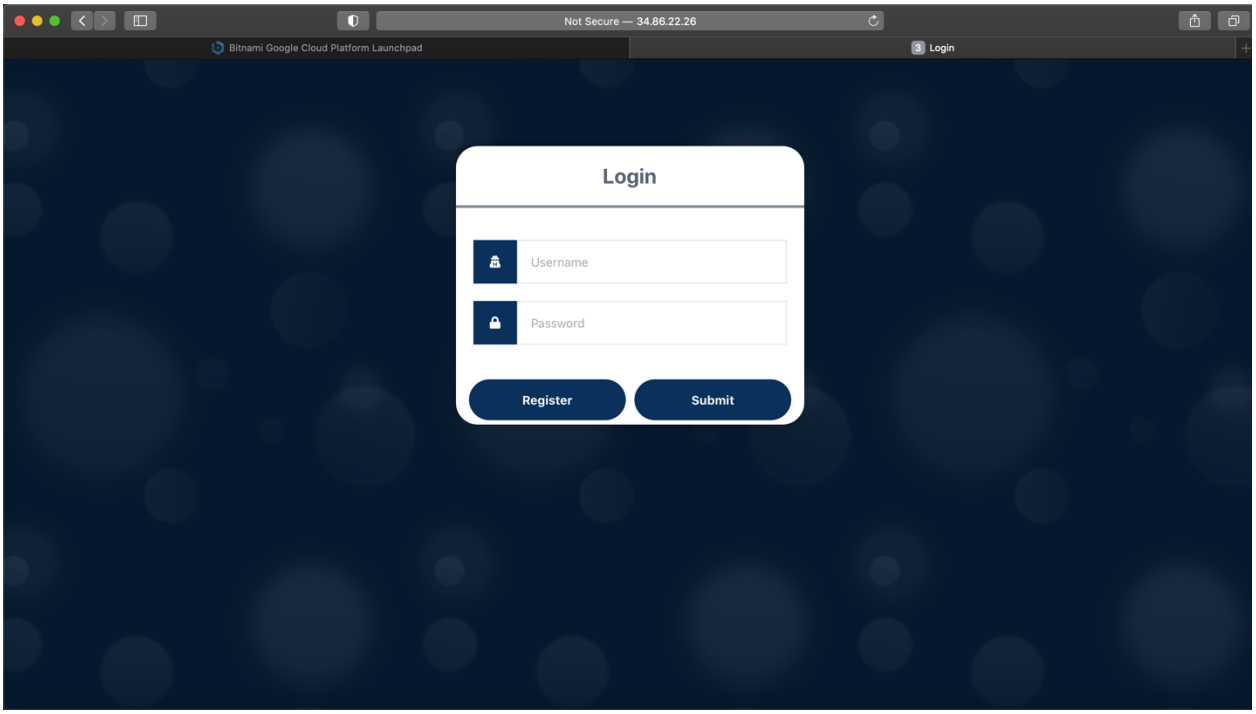


Figure 21 Login Page

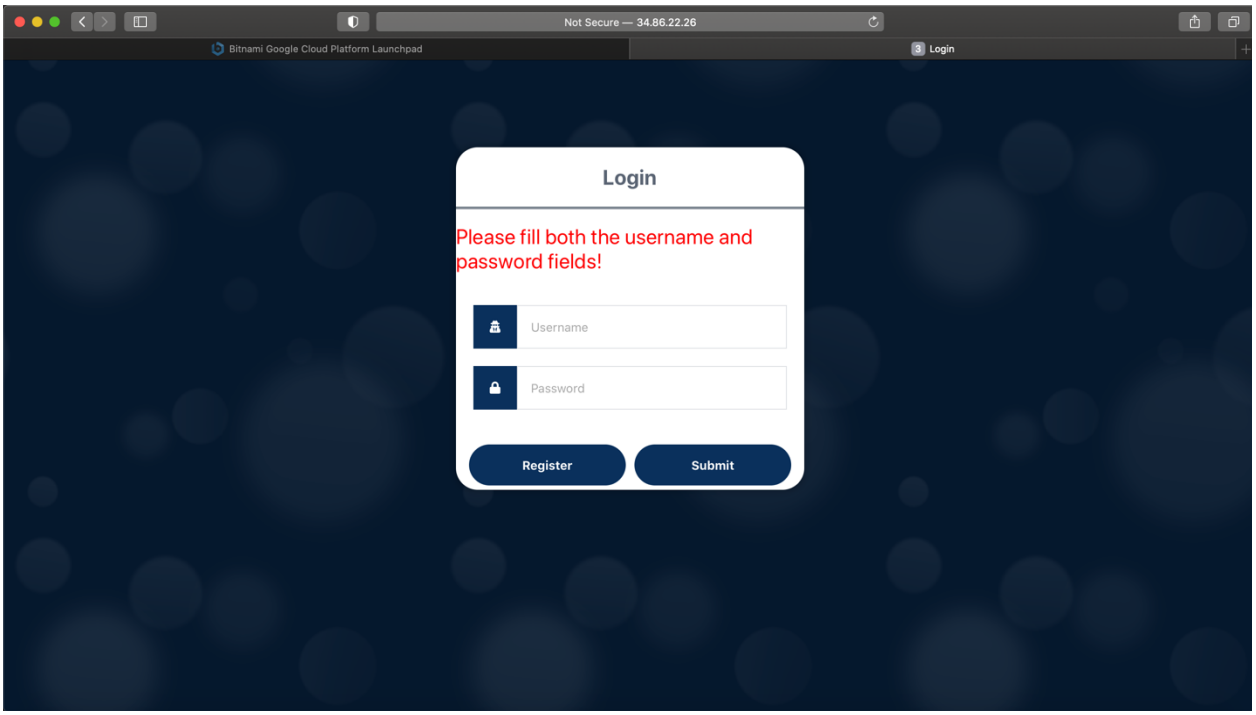


Figure 22 Empty submission

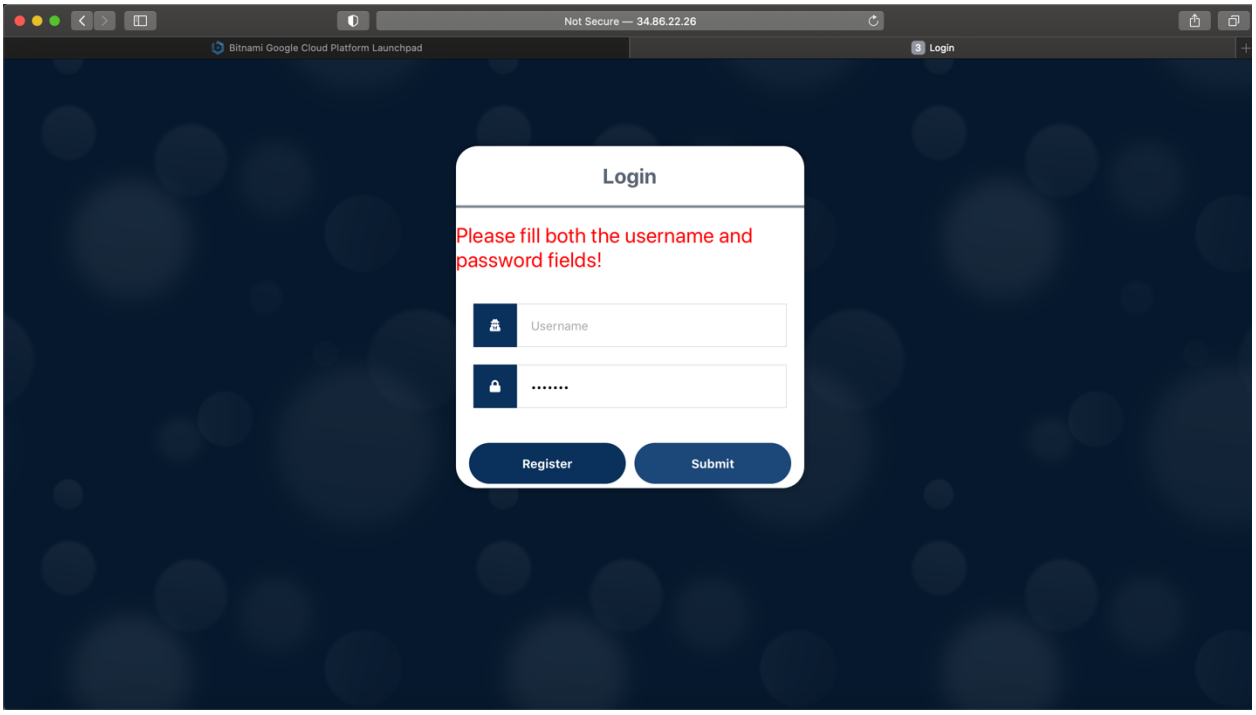


Figure 23 Password filled

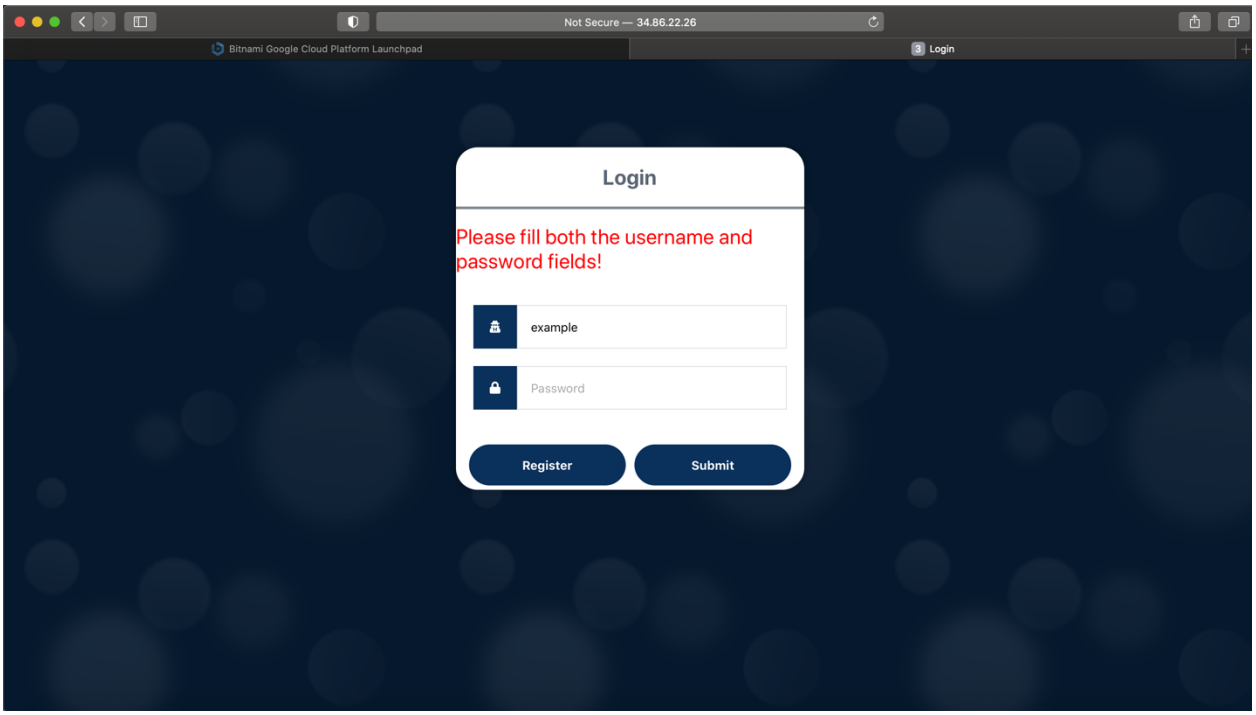


Figure 24 Username filled

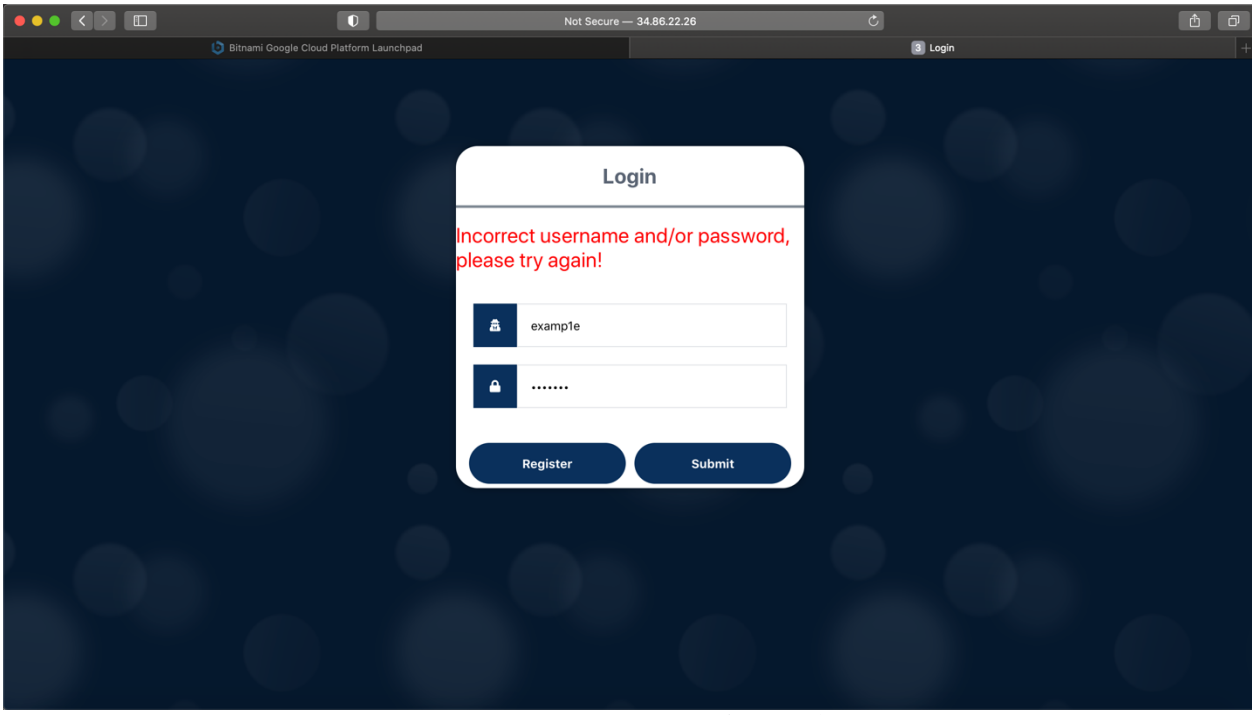


Figure 25 Incorrect username/password

register.php

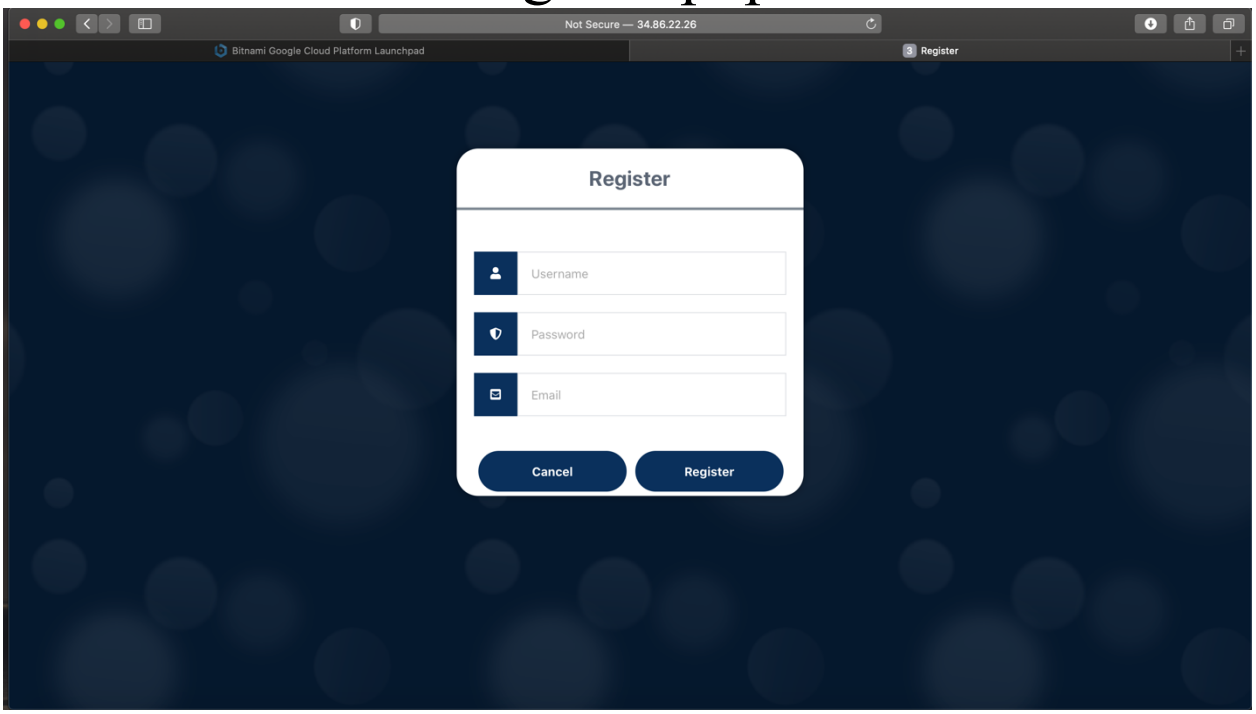


Figure 26 Register page

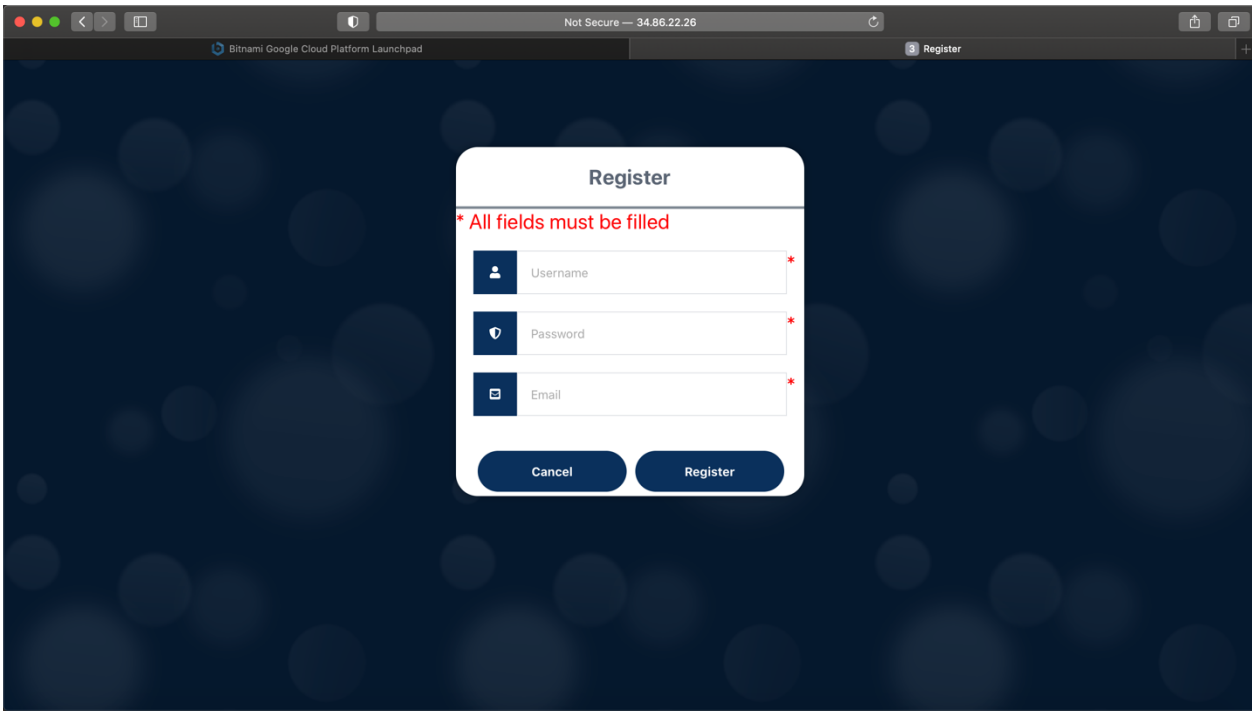


Figure 27 Empty submission

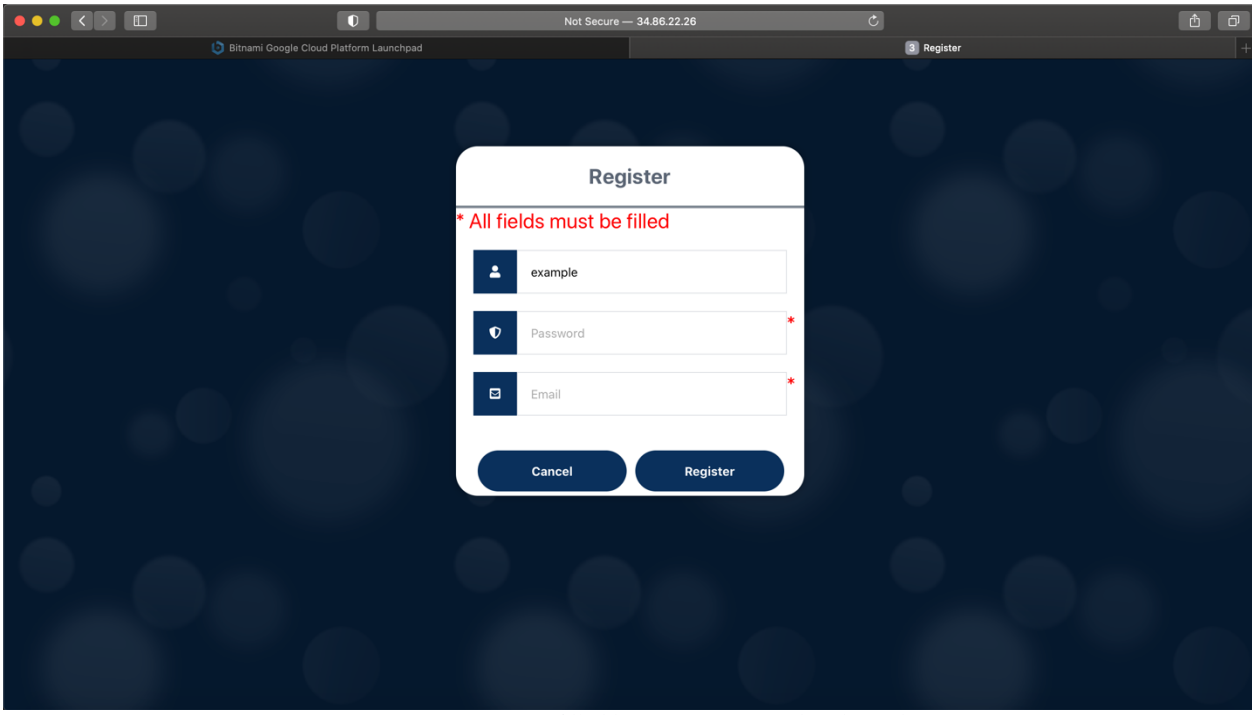


Figure 28 Username filled but empty password and email

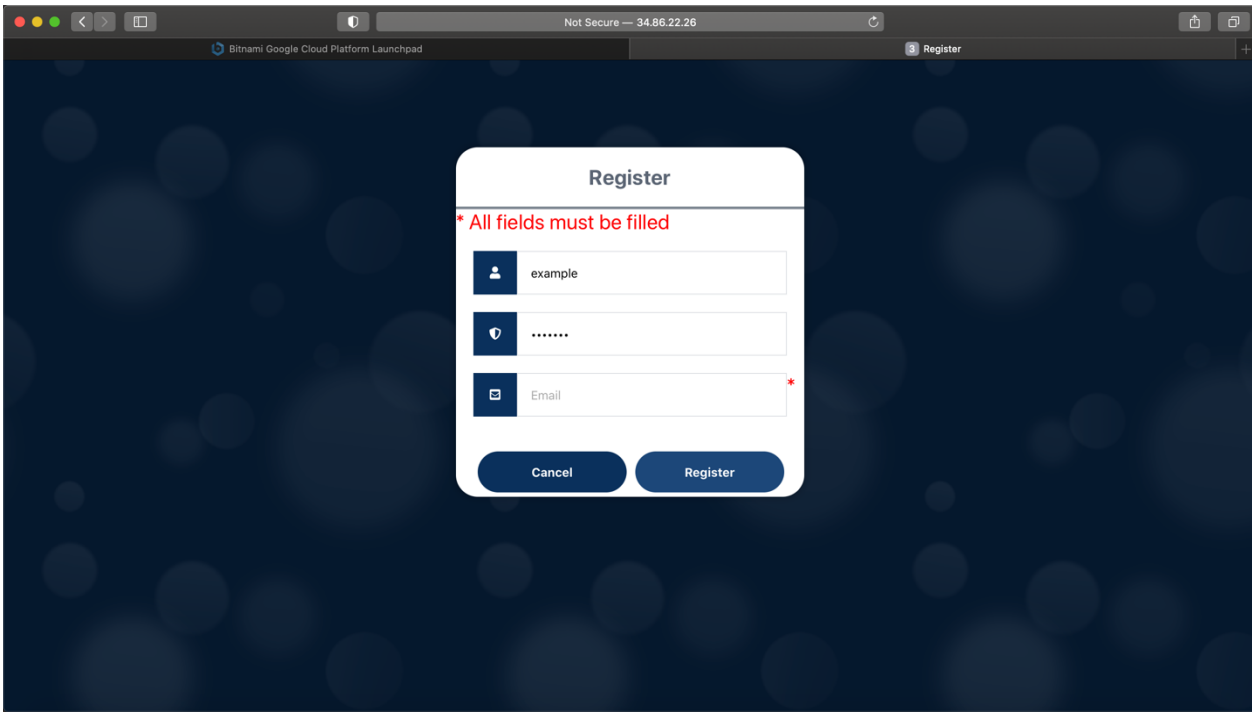


Figure 29 Username and password filled but email is empty

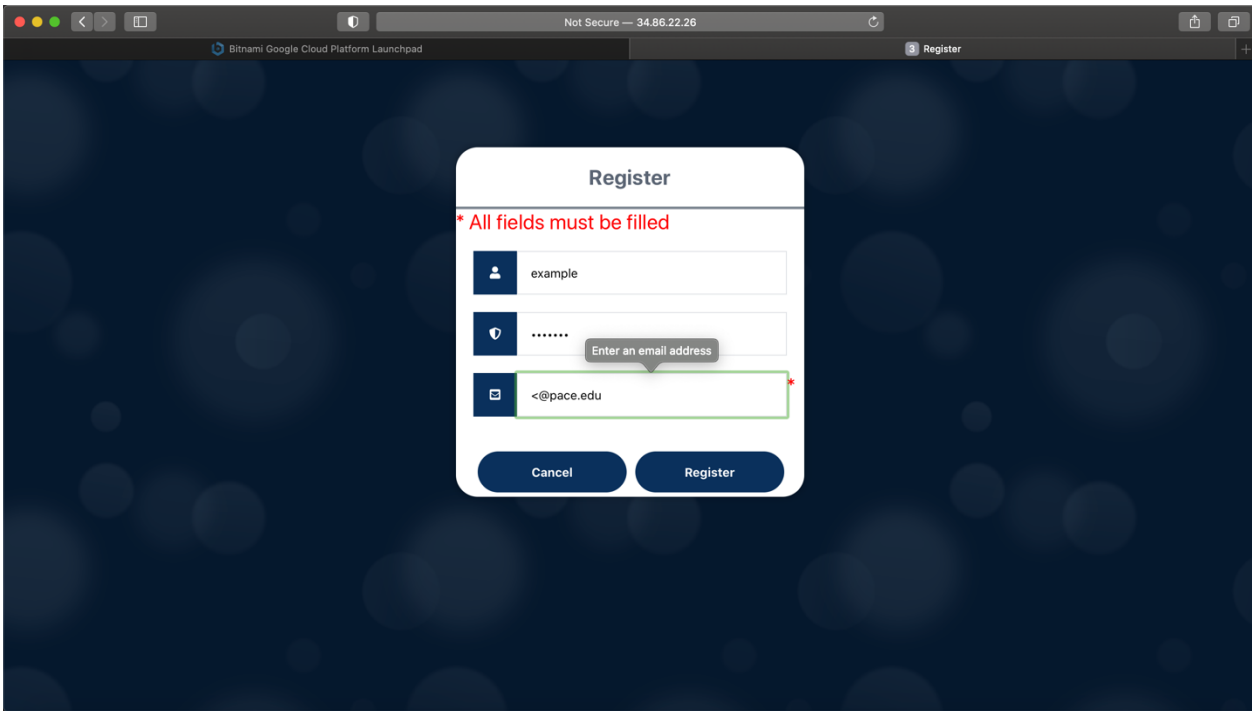


Figure 30 Email validation

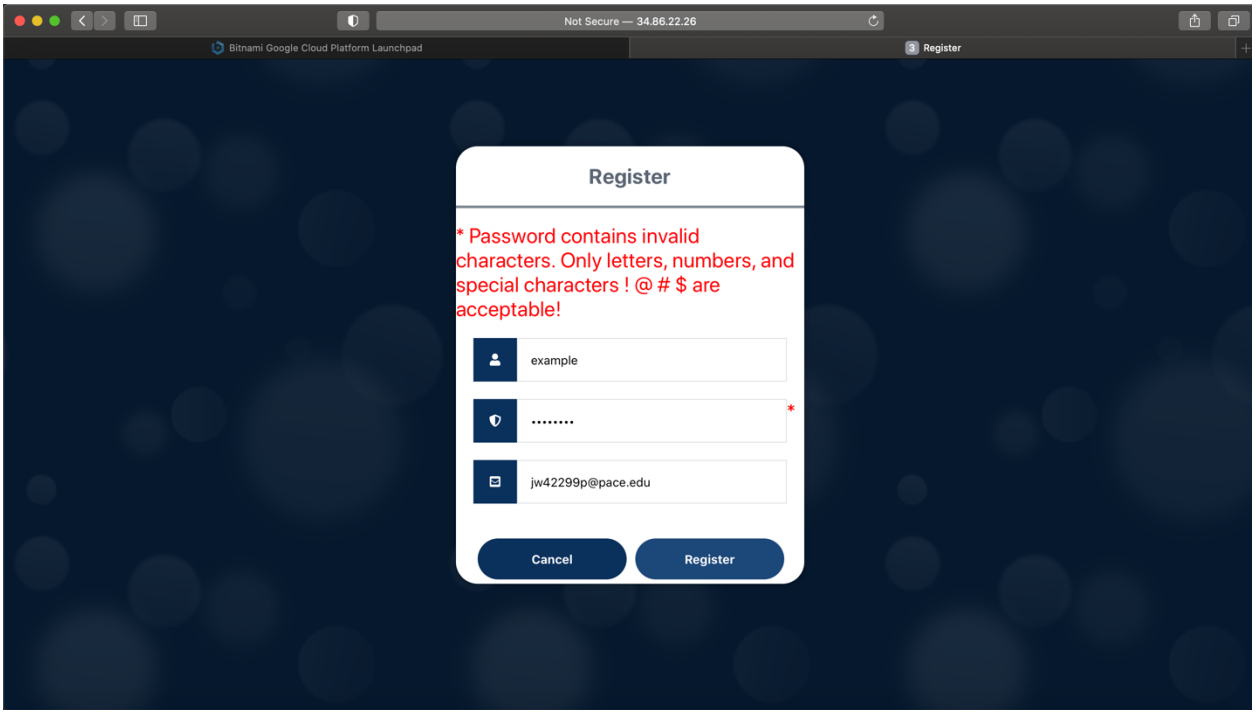


Figure 31 Custom password validation

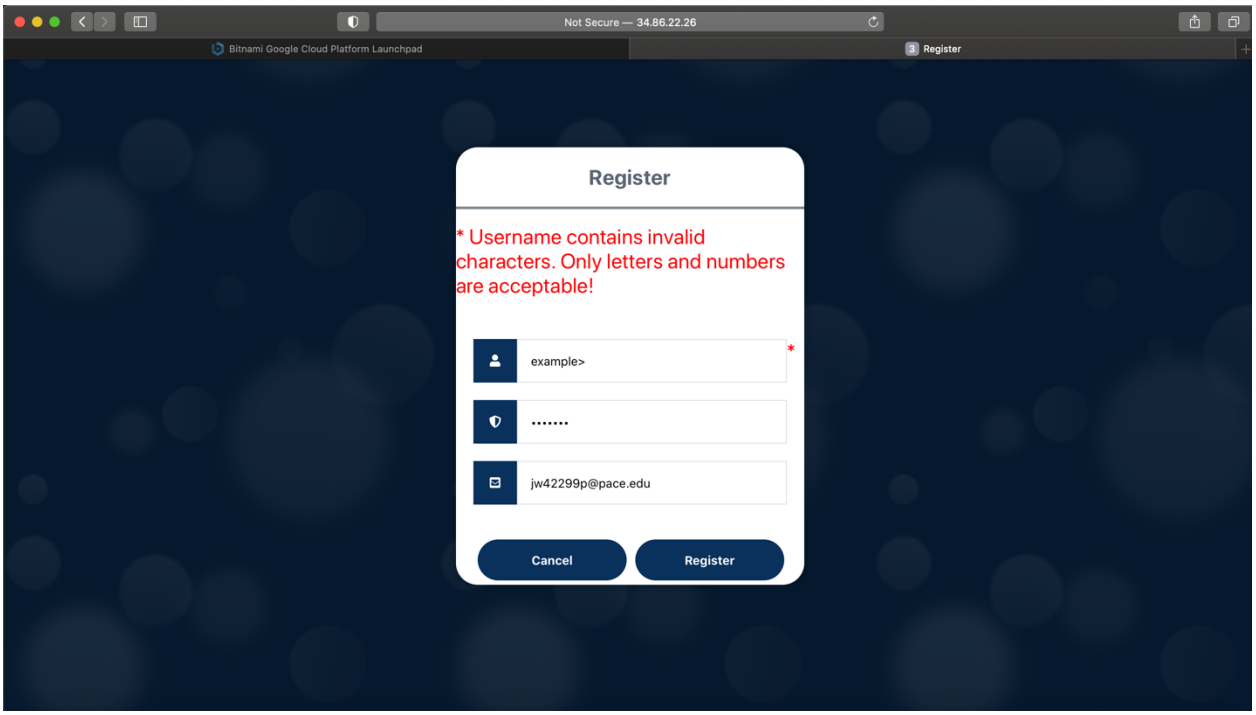


Figure 32 Custom username validation

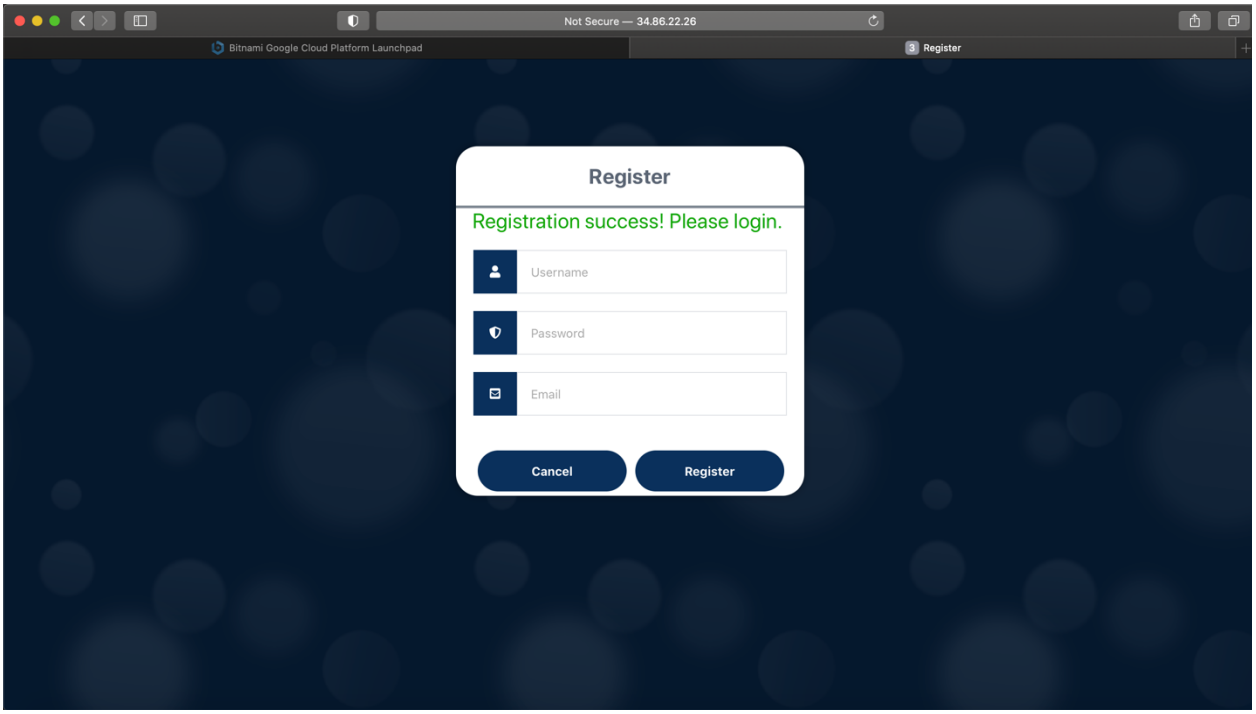


Figure 33 Registration success

otp.php

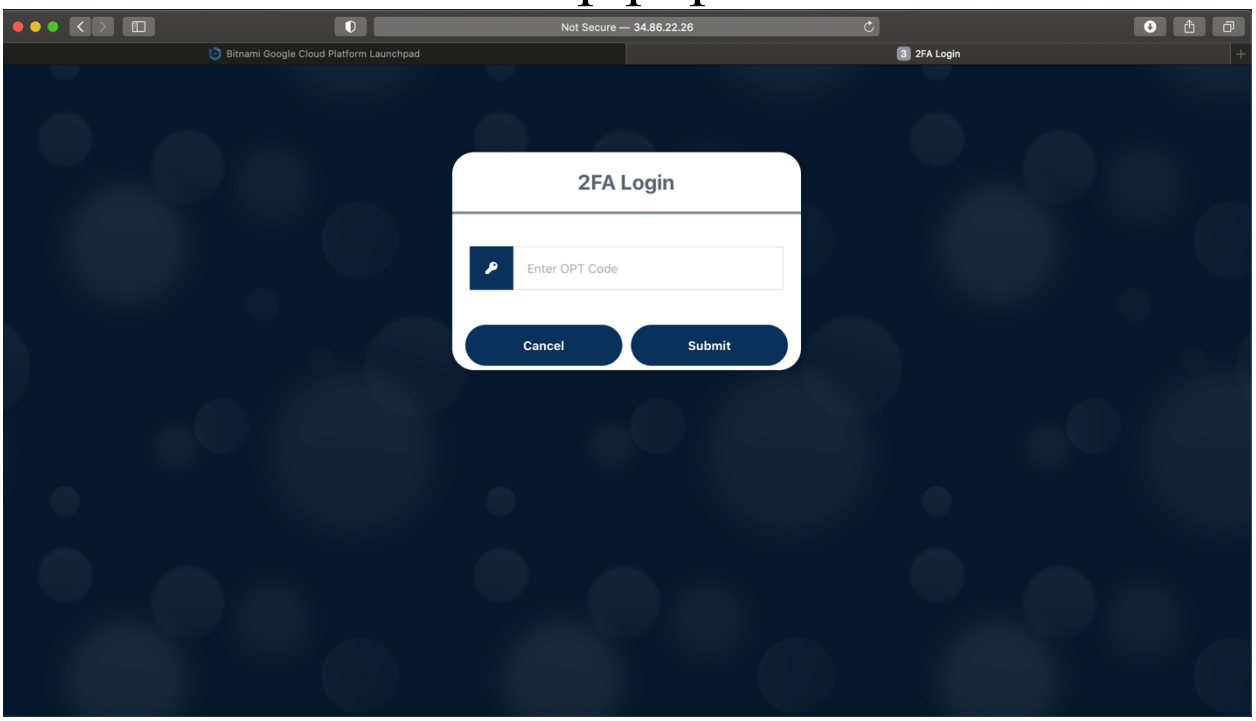


Figure 34 2FA Login page

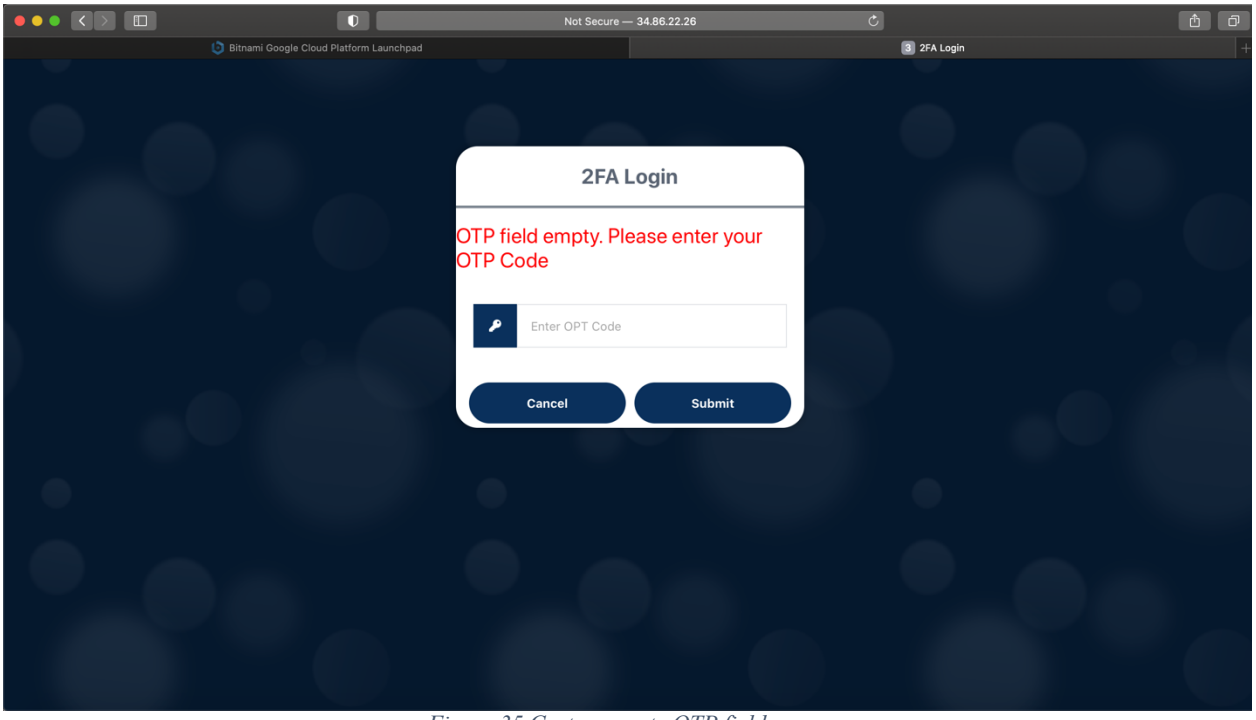


Figure 35 Custom empty OTP field error

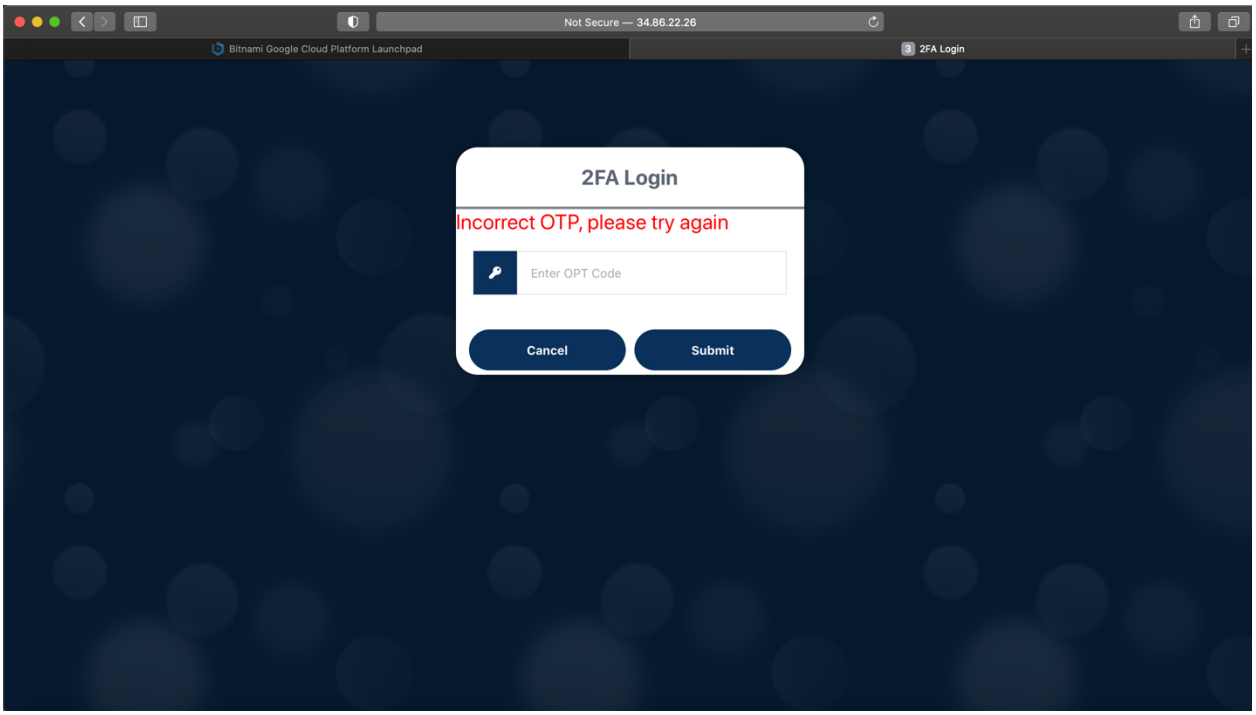


Figure 36 Incorrect OTP submission error

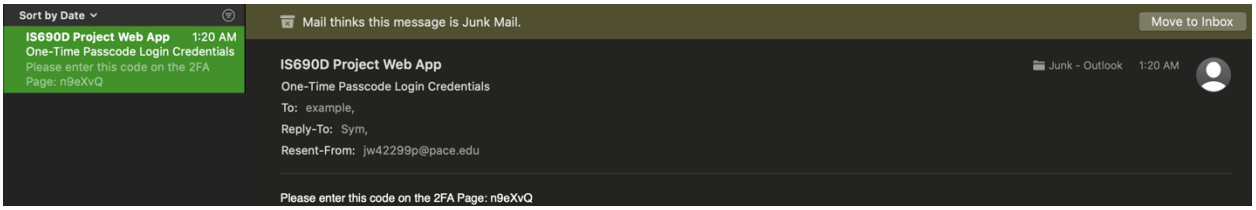


Figure 37 Sample email received from application

home.php

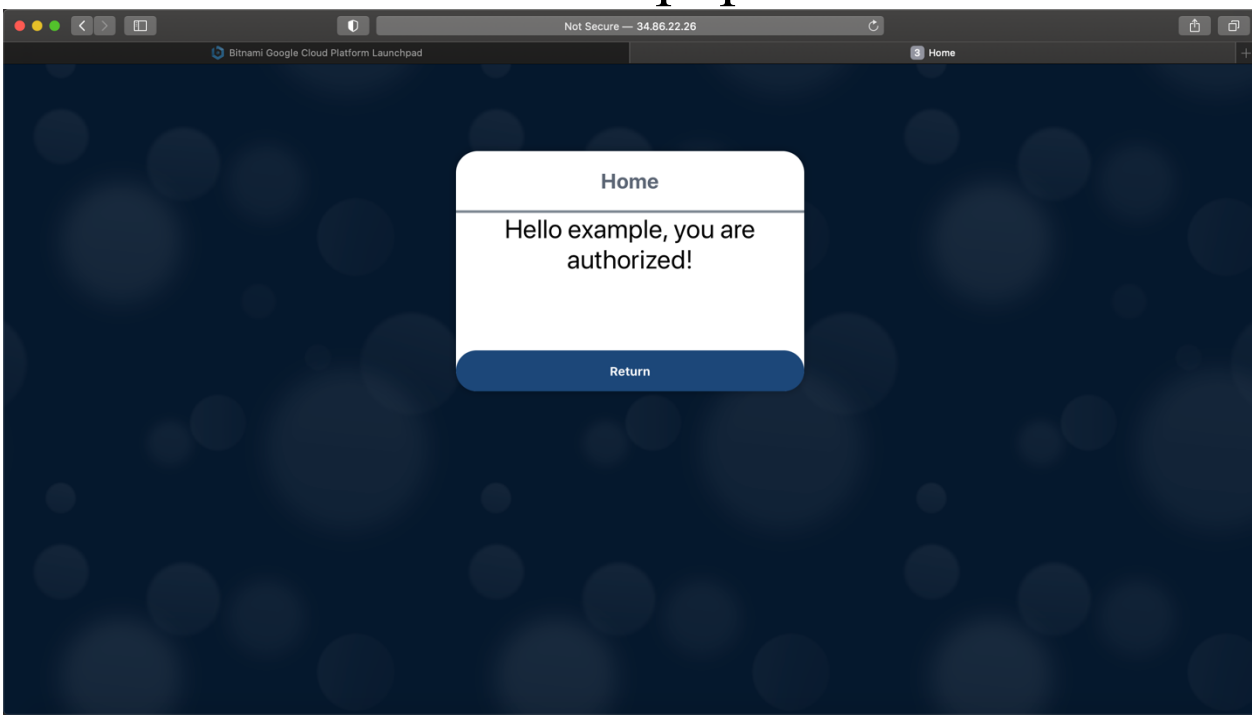


Figure 38 Home Page successful authorization

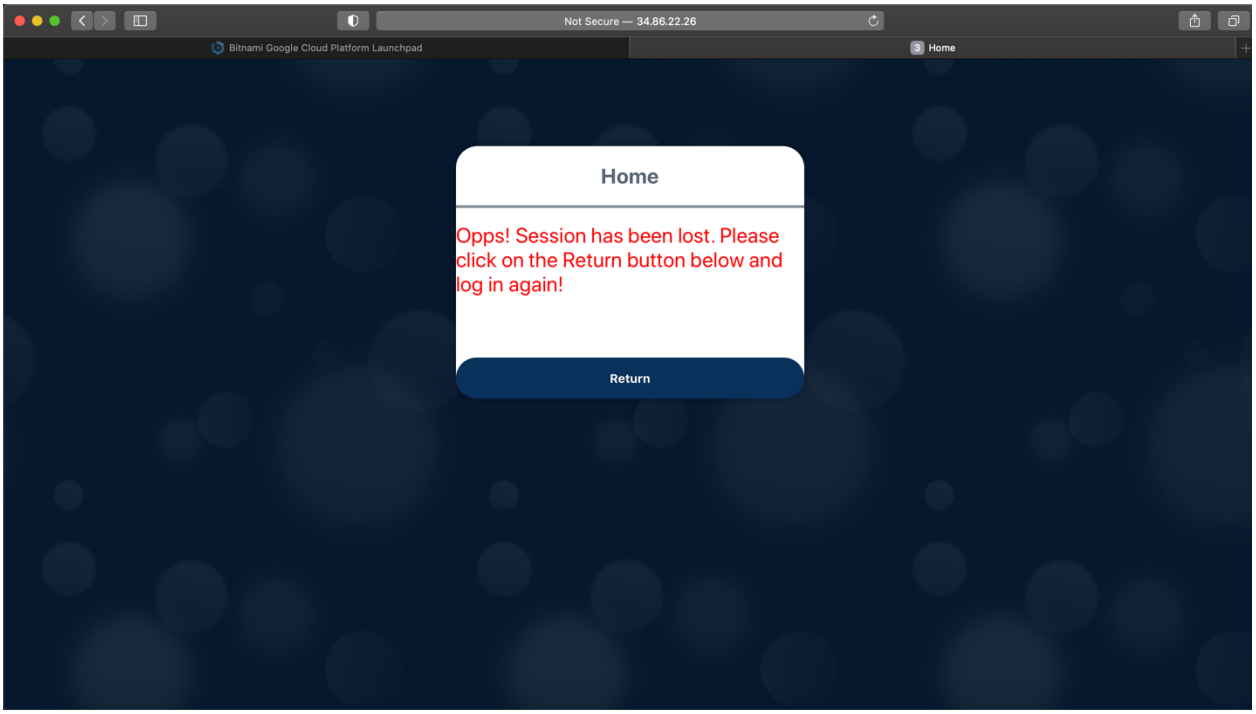


Figure 39 Custom user error, session lost