Classroom Connect

Developer's Documentation

Kyle Wright, Anthony Chan, Jacob Wyner, Joseph Cao

1. Introduction

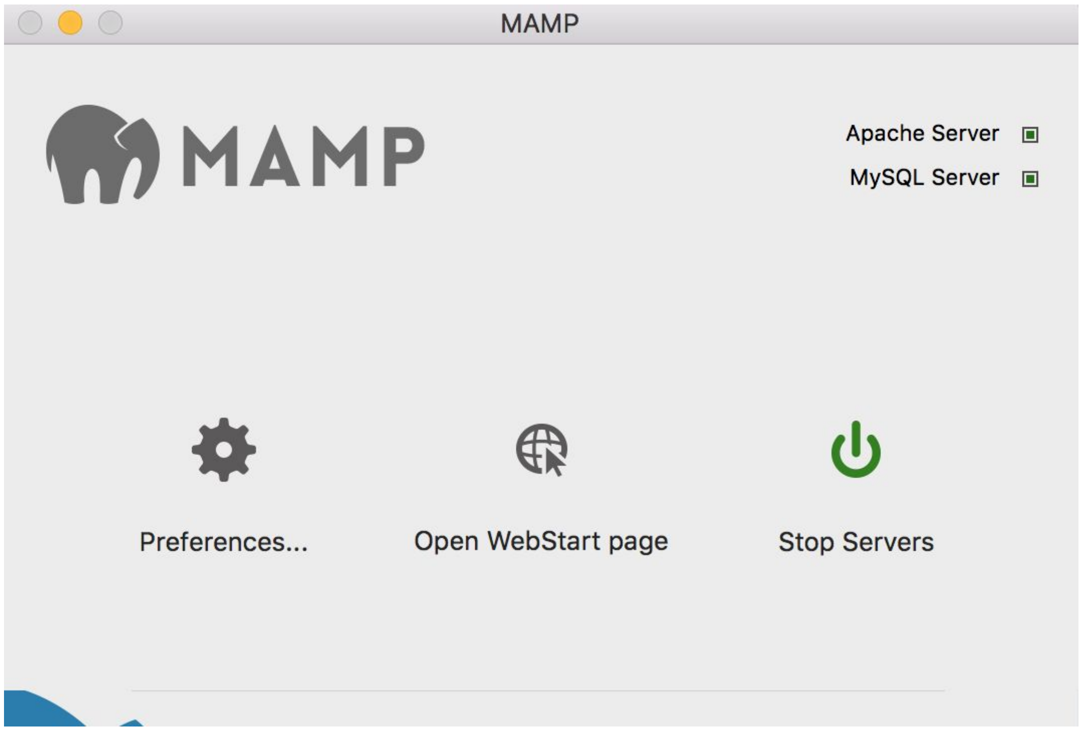
Classroom Connect is a product designed to help facilitate communication between students and professors in a lecture setting. Often times, many students are confused by the same topics during lecture. Classroom Connect software aims to help students communicate their confusion to the professor anonymously. Classroom Connect software also provides a platform for a professor to poll students about specific questions and receive results in real time, assisting both parties in the overall goal of creating a productive learning environment. This document provides an in depth walk through in how to compile, install, and run Classroom Connect. For the sake of simplicity, this document will rely on usage on the LocalHost server on a Mac, however there will be comments regarding deployment to the Google Cloud Services.

2. Getting Started

Classroom Connect is comprised of several major components: the 'Front End' web framework with PHP Networking, the MySQL backend server, and a PHP client websocket server. For the purposes of local testing & development, it is best to view these components together via a downloadable web stack - MAMP (for Macs), or WAMP (for Windows), or LAMP (for Linux). This code was developed on MAMP - *Mac, Apache, MySQL, PHPMyAdmin* - and to download the stack, follow the link below:

https://www.mamp.info/en/

The link will provide steps regarding setting up the MAMP environment on the local system. It will be as easy as downloading the necessary .dmg file, and 'turning on' the Apache server from the MAMP GUI.



3. Setting up the Environment

For viewing of Classroom Connect on your own system, there are certain environmental alterations necessary to ensure that Classroom Connect can work 'out of the box' from github.

Firstly, after downloading or cloning the github repository:

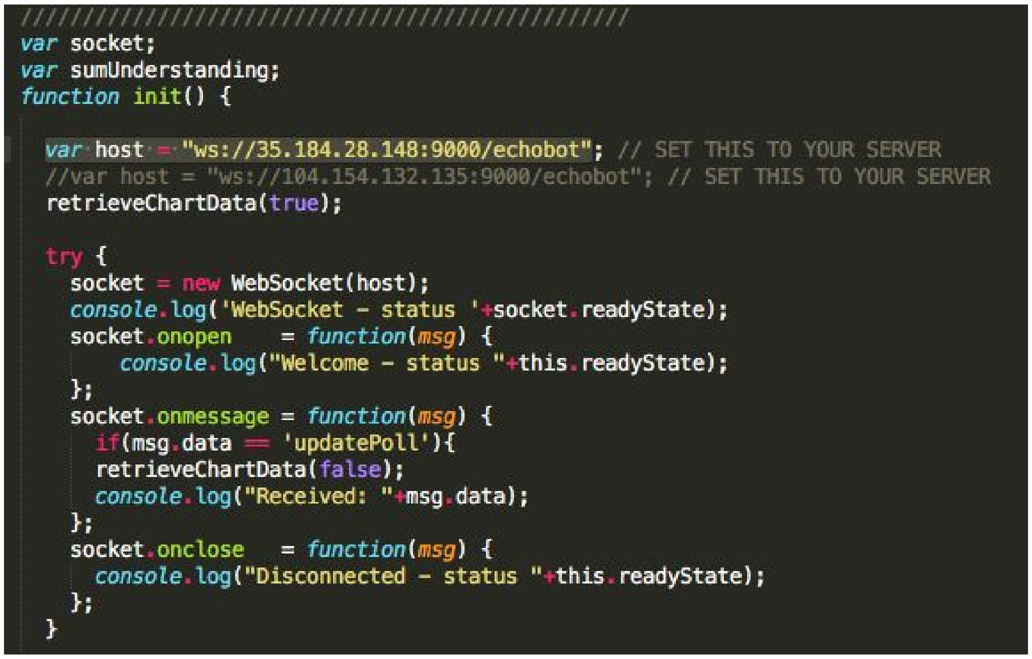
https://github.com/ECE373-Fall16/classroom-connect

Find the files listed below in the Classroom Connect directory:

- create_class.html
- host.php
- join_class.html
- student.html
- pre_existing.html

In each of these files, search for a javascript function, *init*(), and change the host link to the one below:

```
var host = "ws://127.0.0.1:9000/echobot";
```


```
////////////////////////////////////////////////////
var socket;
var sumUnderstanding;
function init() {

  var host = "ws://35.184.28.148:9000/echobot"; // SET THIS TO YOUR SERVER
  //var host = "ws://104.154.132.135:9000/echobot"; // SET THIS TO YOUR SERVER
  retrieveChartData(true);

  try {
    socket = new WebSocket(host);
    console.log('WebSocket - status '+socket.readyState);
    socket.onopen    = function(msg) {
        console.log("Welcome - status "+this.readyState);
    };
    socket.onmessage = function(msg) {
      if(msg.data == 'updatePoll'){
      retrieveChartData(false);
      console.log("Received: "+msg.data);
    };
    socket.onclose   = function(msg) {
      console.log("Disconnected - status "+this.readyState);
    };
  }
```

The link which is already in the code refers to a previous IP of the Google Cloud Service Computer Engine, for the ‘online’ version of Classroom Connect.

After altering the lines of code specified above, please open the MAMP Console, and change the pathway of the Web Server Document Root by going to Preferences > Web Server, and selecting the cloned repository’s ‘front_end’ as Document Root. If you turn on the MAMP Apache server and go to the specified localhost port (by default, MAMP uses port 8888 and 8889 for the web hosting & databasing, respectively)

After changing the websocket address and the Document Root, Classroom Connect should be ready for local testing. Please refer to Section 8: Final Notes & Comments, for troubleshooting.

4. Setting up the Database

This section summarizes how to import the Classroom Connect database into the MySQL server using PHPMyAdmin, for local testing. Open the MAMP Control Panel, turn on the servers, and click on the ‘Open WebStart Page’ in the center of the console. The console will open up your browser. Please scroll down and click on the blue ‘phpMyAdmin’, as seen below:

MySQL can be administered with phpMyAdmin.

This will open up the phpMyAdmin control panel. Please download & zip the file ‘CLASSROOMCONNECT.sql’ from front_end > back_end, and follow the steps below closely to correctly set up the database. If just using the MySQL terminal, just copy and paste each query from the .sql file and execute:

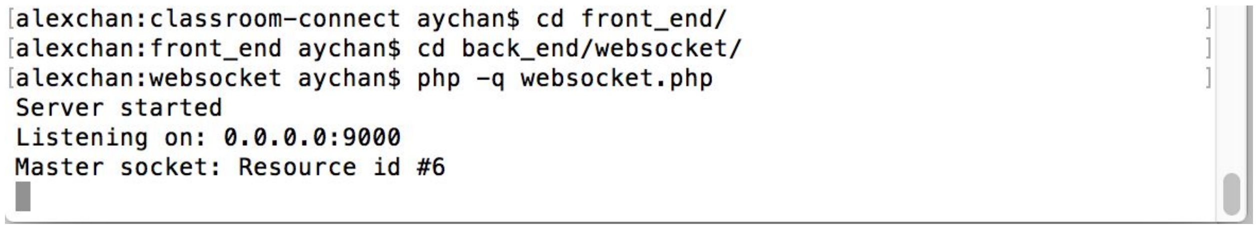
1. On the teal side nav-bar on the left, click on ‘New’, and insert the name ‘CLASSROOMCONNECT’ as the new database name. Click ‘create’.
2. phpMyAdmin will create the new database, and will be visible in the same side nav-bar. Click on the database name.
3. You will by default be viewing the ‘Structure’ tab, which should show an empty database. Please find on the same horizontal nav-bar the ‘Import’ tab, and click on it.
4. Under ‘File to import’, please choose the zipped CLASSROOMCONNECT.sql file.
5. Scroll down the page, and click ‘Go’. This should run through the queries within the file, and will populate the database with empty tables with the correct schema.

5. Setting up the Websocket

This section summarizes the steps required to turn on & turn off the PHP Websocket used within Classroom Connect. Follow the steps below closely to correctly turn on the Websocket Server. Note: these steps assume use of a Mac OS. Steps for alternate operating systems, however, should not be that far off from the ones listed below:

1. Open Terminal from the Applications list.
2. Change directory until in the websocket folder, following the path below:
   cd [PATH TO CLASSROOM-CONNECT]/front_end/back_end/websocket
3. Run the command below to run the php script websocket.php:
   php -q websocket.php

Notice that if your terminal looks like such below, the websocket is correctly running on port 9000 of your system. If you open a browser and go to one of the pages from Section 3 of this Document, you will notice that users will connect with an id:



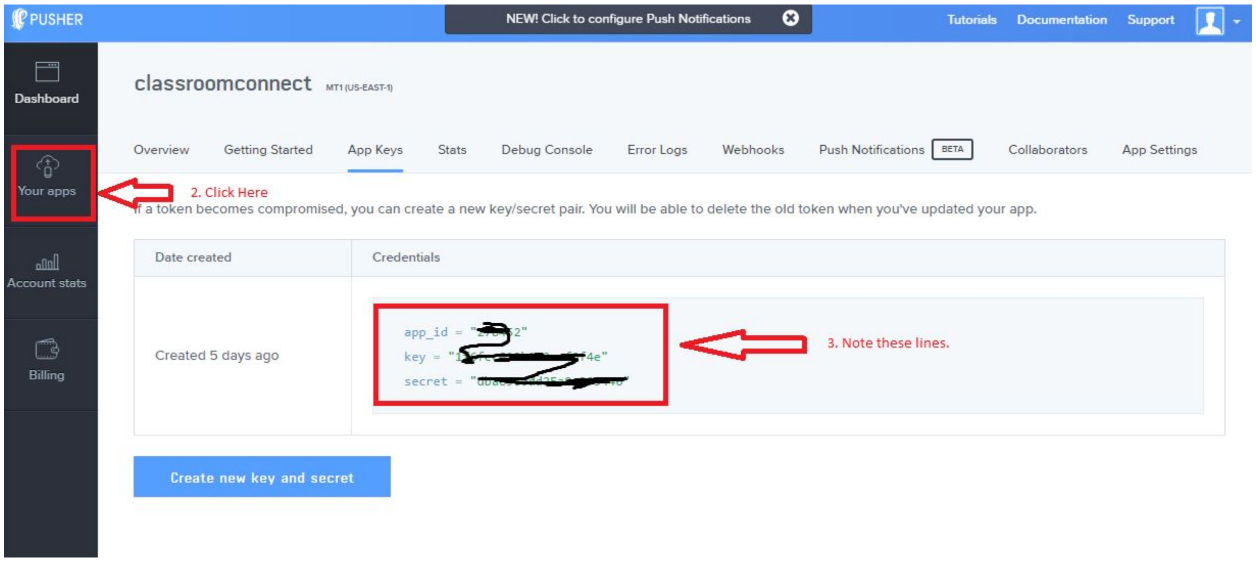
```
alexchan:classroom-connect aychan$ cd front_end/
alexchan:front_end aychan$ cd back_end/websocket/
alexchan:websocket aychan$ php -q websocket.php
Server started
Listening on: 0.0.0.0:9000
Master socket: Resource id #6
```

If there are issues running this command or starting the Websocket server, please refer to Section 8: Final Notes & Comments, for troubleshooting.

6. Setting up the Chat Service

Our implementation of the chat as shown during the demo is not fully stable. To enable it, there are several code changes to our websocket and php pages that are necessary. During our testing, this caused crashes to our other services under some circumstances, such when too many users connected to the chat at once, or if a user refreshed the chat page and then disconnected . As a result, we created a clone of our main Compute Engine instance, and redirected a subdomain of our website to this address. This allowed us to separate the chat functionality from the main functions of the website. This allowed us to use a separate websocket ensuring that if the chat crashed, the main site would still be able to perform adequately. Since the combined implementation was not fully stable, it is not reflected under our v1.0 code. However, in order to add chat functionality the following steps should be followed.

1. Register for an account at http://www.pusher.com
2. Login into the newly created account and click on “Your Apps” under the left sidebar.
3. Record the “app_id”, “key”, and “secret” keypairs



If a token becomes compromised, you can create a new key/secret pair. You will be able to delete the old token when you've updated your app.

Date created	Credentials

Created 5 days ago	app_id = "[redacted]"
	key = "[redacted]"
	secret = "[redacted]"

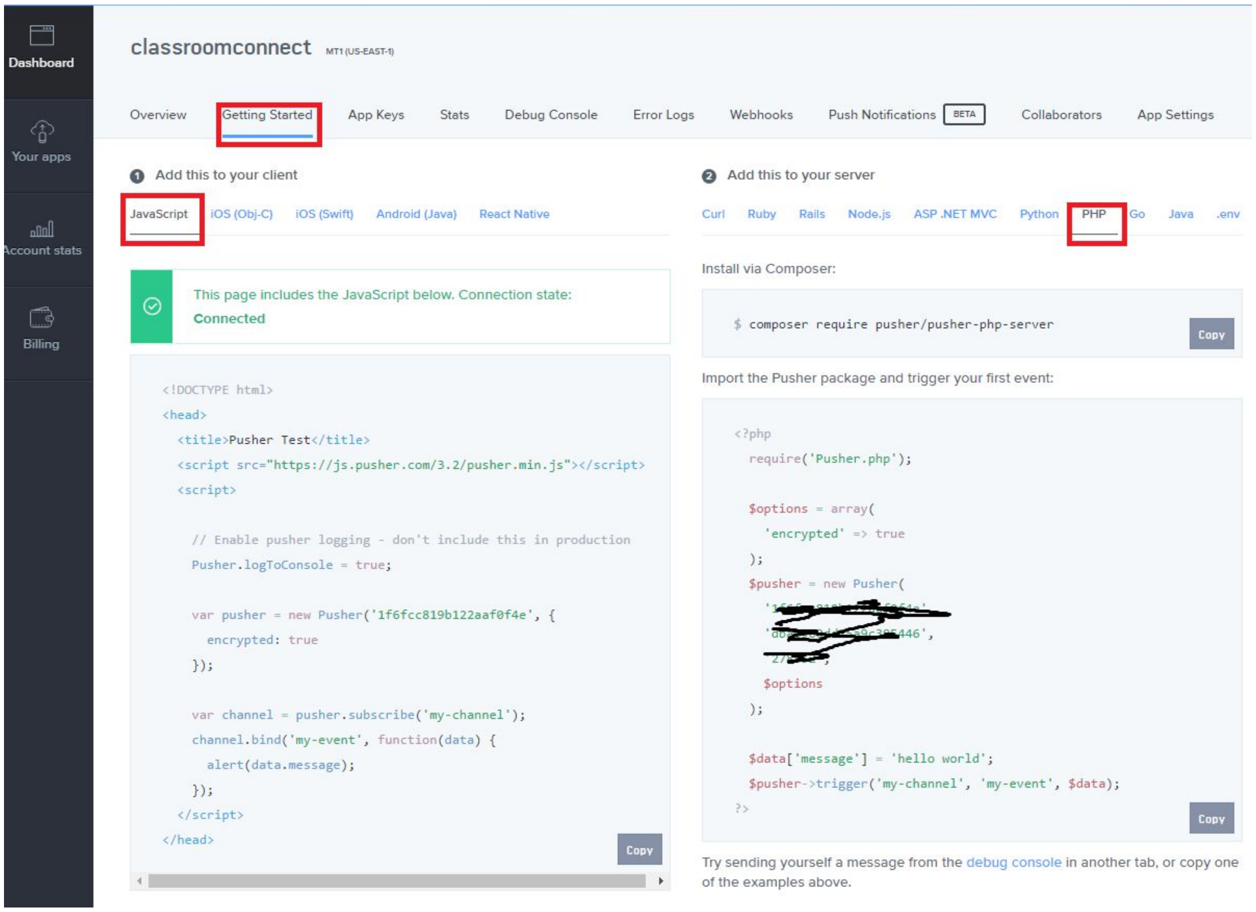
4. Next navigate to the Getting Started panel.
5. This section will differ depending on your environment, in this section we will specifically be focusing on Debian GNU/Linux 8 (jessie), the default OS for Google Cloud Compute Engine VM instances.
    a. Install composer using the following command

    ```
    curl -sS https://getcomposer.org/installer | sudo php -- --install-dir=/usr/local/bin --filename=composer
    ```

    b. After composer is installed run the following command where your php backend root directory is.

    ```
    composer require pusher/pusher-php-server
    ```

6. Next, navigate back to the Getting Started panel and click on JavaScript under “Add this to your client” and click on PHP under “Add this to your server”.



This page includes the JavaScript below. Connection state: Connected

```
<!DOCTYPE html>
<head>
  <title>Pusher Test</title>
  <script src="https://js.pusher.com/3.2/pusher.min.js"></script>
  <script>

    // Enable pusher logging - don't include this in production
    Pusher.logToConsole = true;

    var pusher = new Pusher('1f6fcc819b122aaf0f4e', {
      encrypted: true
    });

    var channel = pusher.subscribe('my-channel');
    channel.bind('my-event', function(data) {
      alert(data.message);
    });
  </script>
</head>
```

Install via Composer:

```
$ composer require pusher/pusher-php-server
```

Import the Pusher package and trigger your first event:

```
<?php
  require('Pusher.php');

  $options = array(
    'encrypted' => true
  );
  $pusher = new Pusher(
    [redacted]
    $options
  );

  $data['message'] = 'hello world';
  $pusher->trigger('my-channel', 'my-event', $data);
?>
```

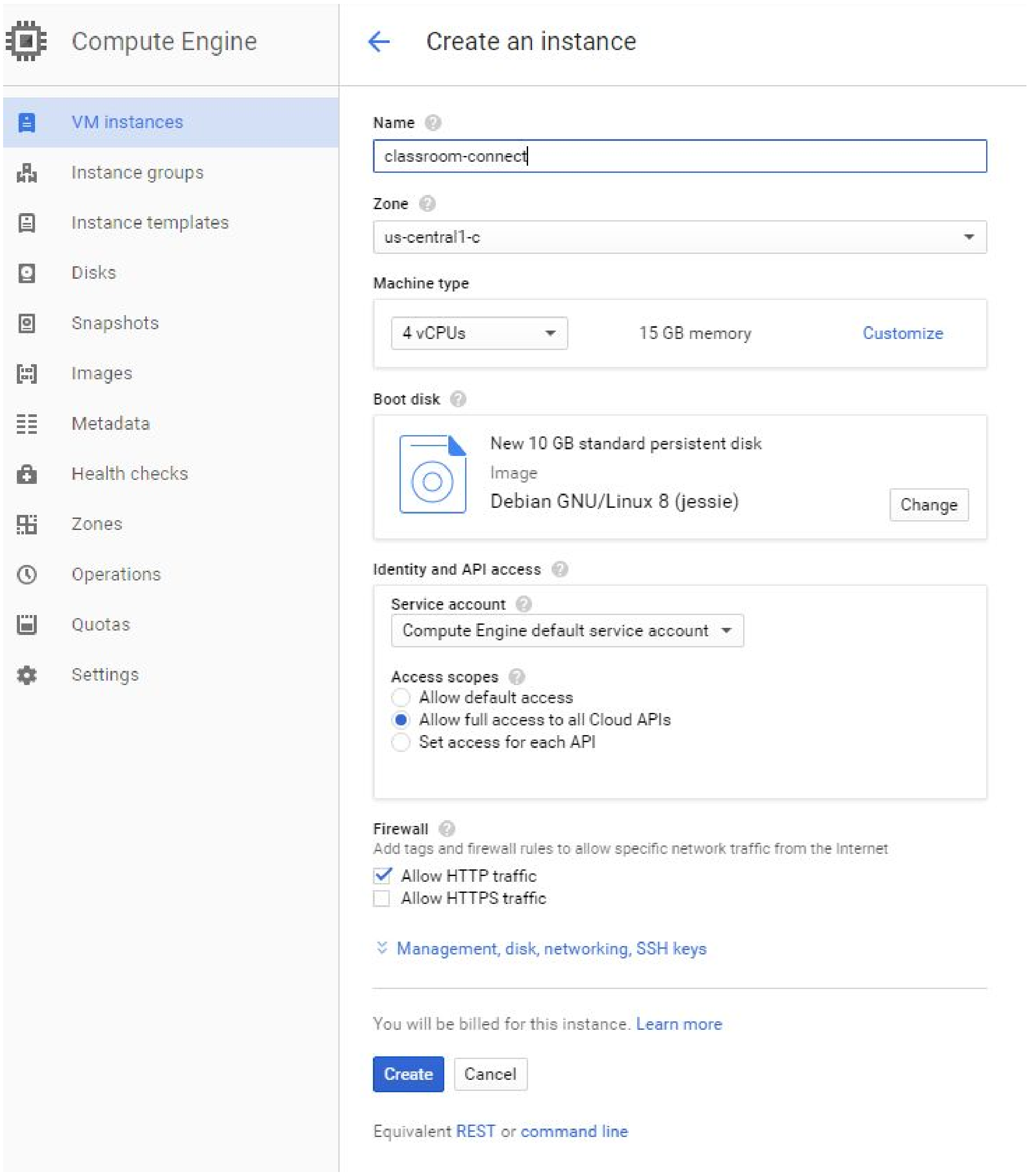
Try sending yourself a message from the debug console in another tab, or copy one of the examples above.

7. Add the HTML Javascript shown on the left panel to the “student.html” page in your Webroot directory. The look and feel of the chat widget can all be changed using basic HTML.
8. Next add the PHP code to “student.html” as well as any other page where you would like to implement a chat.
    a. Note you may need to configure your Apache server to allow parsing of html pages as php in order to enable this functionality, if running on MAMP, this should be enabled by default
    b. Refer to the troubleshooting section of this document for more details

7. Deploying to Google Cloud Services

This section concerns the deployment of ClassroomConnect on Google Cloud Services. The following process is for a specific configuration but may be used as a basis for other configurations.

1. First create a new VM instance as shown below. Take care to check the “allow HTTP traffic” as this will allow us access to the correct ports without having to change the firewall settings. If a custom port is necessary then click on the Edit tab under your VM instance and click on the “default” network and add the correct ports as needed. If you choose to run the full stack Apache, MySQL, and PHP as well as the chat service on one instance then it is recommended to choose a 4 vCPU VM. In our testing a 1 vCPU VM was inadequate when 5+ users are concurrently actively use both the core site and the chat.



Name: classroom-connect

Zone: us-central1-c

Machine type: 4 vCPUs — 15 GB memory — Customize

Boot disk: New 10 GB standard persistent disk
Image: Debian GNU/Linux 8 (jessie)

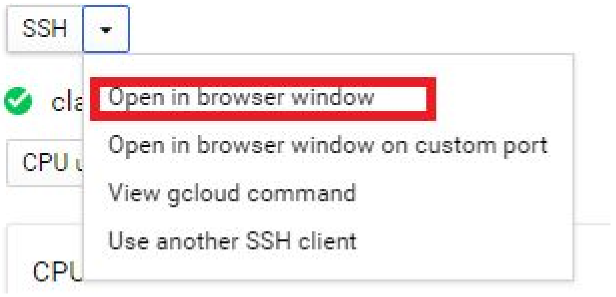
Identity and API access
Service account: Compute Engine default service account
Access scopes:
- ○ Allow default access
- ● Allow full access to all Cloud APIs
- ○ Set access for each API

Firewall
Add tags and firewall rules to allow specific network traffic from the Internet
- ☑ Allow HTTP traffic
- ☐ Allow HTTPS traffic

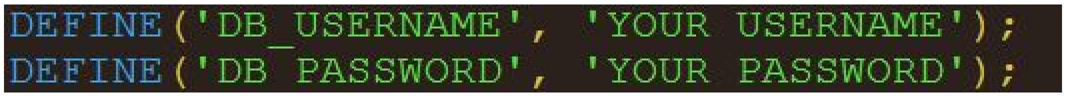
You will be billed for this instance. Learn more

Equivalent REST or command line

2. Next ssh to your newly created VM. There are many ways to do this depending on your development environment and tools you have available (if you wish to connect using a SSH client such as PuTTY, WinSCP, etc then it will be necessary to create an SSH key pair to access your VM see here), but for the sake of meeting core functionality and ease of use, it is recommended that the built in browser SSH terminal is used for the following steps.

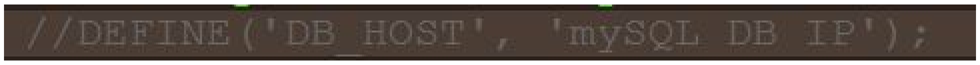


3. Run the following commands to install the necessary software
   sudo apt-get install apache2
   sudo apt-get install mysql-server libapache2-mod-auth-mysql php5-mysql
   sudo mysql_install_db
   sudo /usr/bin/mysql_secure_installation
4. After running the last command, there will be a prompt asking you to set your MySQL username and password. If you wish to use something other than ‘root’ and ‘root’ for the username and password (it is recommended to change both) navigate to the following directory: /front_end/back_end/ and change the following lines (8 and 9) in the “sqlConnectionGET.php” and “sqlConnection.php” files



```
DEFINE('DB_USERNAME', 'YOUR USERNAME');
DEFINE('DB_PASSWORD', 'YOUR PASSWORD');
```

   a. Aside, if you are running the database remotely on a separate instance, i.e. Google Cloud SQL instances, then be sure to allow remote connections on that mySQL instance and uncomment the following line in the previous files as well



```
//DEFINE('DB_HOST', 'mySQL DB IP');
```

5. Now under the VM Instance tab note the “external IP” of your VM instance. Change corresponding IPs in the code to this IP, see section 3. For details on how to do this.
6. Finally, run the websocket following the same process as in section 5.
   a. Note, since the websocket process will close by default if the SSH connection is terminated, it may be convenient to “daemonize” the processes and allow it to persist in the background after the SSH connection is closed. A sample script to “daemonize” the process is in webroot/front_end/back_end/daemonize.sh.

8. Understanding the Code

As a web service, Classroom Connect relies heavily on the conversation between the front end's AJAX commands & the PHP scripts which grab information from the mySQL database. That interaction, along with the php websocket, make Classroom Connect what it is. In each of the html & php pages listed in Section 3 of the documentation, the page will load the connection to the php websocket, and will initialize any php SESSION variables, or will run a php script to obtain values from the mySQL database through an AJAX call to said php script. Most all user interactions, from student.html or host.php, are linked to AJAX calls as well, which will run a MySQL query, and will alert the websocket to tell all other clients to either refresh or run certain commands themselves.

9. Final Notes & Comments

Overall, if all steps are followed, to run Classroom Connect locally, please follow these last few steps to confirm & run the web service:

1. Open the MAMP service from the Applications folder, and turn on the Apache server.
2. Follow the steps from Section 5: Setting up the Websocket, to turn on the php server.
3. Open Localhost at the specified port, and enjoy!

Potential errors:

The websocket is not seeing any client connections on client-loading of the necessary pages.

- This issue may be due to an error in the websocket link typed in. Please go to each file specified in Section 3, and ensure that the host link is currently written.
- Please ensure that port 9000 is free during testing & development
- Please ensure that the websocket connects to port 9000 of localhost, or whichever port you would like to use.

phpMyAdmin is throwing errors with the CLASSROOMCONNECT.sql.zip file when attempting to import the database schema

- Please ensure that the database created is 'CLASSROOMCONNECT'. Take care to ensure no spaces, and all upper-case.
- If the database is correct, try to copy, paste & execute bits of the queries within CLASSROOMCONNECT.sql in the SQL tab, right of the 'Structure' tab in phpMyAdmin.

Poll Chart and/or User Understanding does not update correctly

- Make sure to change and update all necessary IP addresses and ports for your configuration

- If you are on linux or an environment that does not support a preconfigured Apache, MySQL, and PHP stack (such as MAMP, WAMP, etc) then it might be necessary to modify each component individually to function correctly together
- In general, the following functionality is required:
    - Apache parsing of html as php
    - System must be able to access MySQL database with the correct schema configuration (for example if using a separate optimized instance for SQL such as Google SQL Instances ensure that appropriate permissions are set on both ends)
    - Change the Apache configuration file on the line starting with “DocumentRoot” in `/etc/httpd/conf/httpd.conf` to point to the correct directory
    - Ensure the DocumentRoot contains the previously specified folder structure
- If deploying on a Google Cloud VM, make sure to update IPs on each reboot if the instance has an ephemeral IP assignment