LAB 16a

WEB SITE HOSTING

What You Will Learn

- How to host a PHP web site using the Heroku Platform-as-a-Service
- How to host a PHP web site using the Google Cloud Platform App Engine
- How to host a PHP web site using the

Approximate Time

The exercises in this lab should take approximately 100 minutes to complete.

Fundamentals of Web Development, 3rd Ed

Randy Connolly and Ricardo Hoar

Textbook by Pearson
http://www.funwebdev.com

Date Last Revised: March 23, 2019

In this lab, you will be hosting a sample web application on a variety of different cloud-based hosting environments. In the first section, you will begin with Heroku, a popular Platform-as-a-Service (PaaS). In the second, you will host the same application on the Google Cloud Platform (GCP), a more complicated Infrastructure-as-a-Service environment.

PREPARING DIRECTORIES

1 This lab has additional content that you will need to copy/upload this folder into your eventual working folders. You will notice that in the starting folder, there are folders for Heroku, GCP, and AWS.

HOSTING USING HEROKU PLATFORM-AS-A-SERVICE

Heroku focuses on the developer experience: it abstracts away many of the complexities involved in setting up a virtual server and the necessary services it requires. Applications are hosted within Heroku dynos, which are virtualized Linux containers. In this example, you will make use of a free shared dyno. While quite limited it is free and will illustrate the ease of deployment ease of this PaaS.

Note: using Heroku will require you create an account on heroku.com and install software on your development computer.

Exercise 16a.1 — SETTING UP HEROKU

1 Navigate to `https://www.heroku.com` and create a free account.

2 You now need to install the Heroku CLI. Navigate to `https://devcenter.heroku.com/articles/heroku-cli` and choose the appropriate installer.

You will also need to have git installed on your computer.

3 Verify your installation by entering the following command from the terminal/command window/powershell/etc.

```
heroku -version
```

4 If heroku has been installed correctly, then you need to login into the CLI via:

```
heroku login
```

Depending on your installation, you will either login via the CLI or via the web browser.

You are now ready to begin creating applications hosted on Heroku. This typically involves the following starting process:

1. On development machine, `cd` to the root folder for the site.
2. Run: `git init`
3. Run: `heroku create`

This generates a random domain name for your site, and then links the local git repository with this new heroku domain.

Exercise 16a.2 — Creating a LAMP Stack Site

1 Examine the folder where you copied the Heroku starting files for this lab (recall you are provided with a Heroku version of the starting folder and a Google Cloud version of the starting folder). If you examine this folder, you will see it consists of a simple data-driven (from MySQL) PHP application.

2 Heroku uses the Composer program to manage PHP dependencies. Thus, every PHP application in Heroku must have a `composer.json` file.

Create an empty file in your site's root folder named `composer.json`.

3 Using the terminal, navigate to the folder mentioned in step 1.

4 Create a git repository for this app using the following command:

```
git init
```

5 Add the files in your folder to your git repo and commit them via the following:

```
git add *
git commit -m "first upload to heroku"
```

6 Create the Heroku project via the following

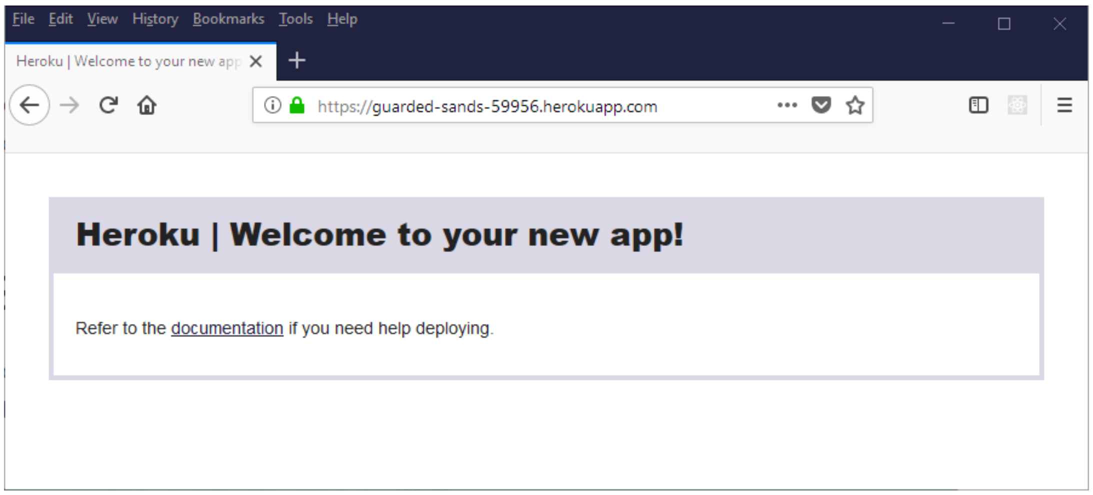
```
heroku create
```

You will see that this command generates a random url for this new site and a remote git repo for the site. For instance, when I ran this command, I saw the following message:

```
Creating app... done, guarded-sands-59956
https://guarded-sands-59956.herokuapp.com/ |
https://git.heroku.com/guarded-sands-59956.git
```

7 In the browser, go to the URL generated in step 6.

You should see something similar to that shown in Figure 16a-1.


File Edit View History Bookmarks Tools Help
Heroku | Welcome to your new app
https://guarded-sands-59956.herokuapp.com

Heroku | Welcome to your new app!

Refer to the documentation if you need help deploying.

Figure 16a.1 – After creating the heroku project

8 You now have to upload your local code to your application's Heroku site. Because of Heroku's integration with `git`, this simply requires a `git push`.

Using the terminal, enter the following:

```
git push heroku master
```

There will be a variety of messages displayed detailed the upload and configuration process.

9 When all done, try testing the same URL from step 7 in the browser.

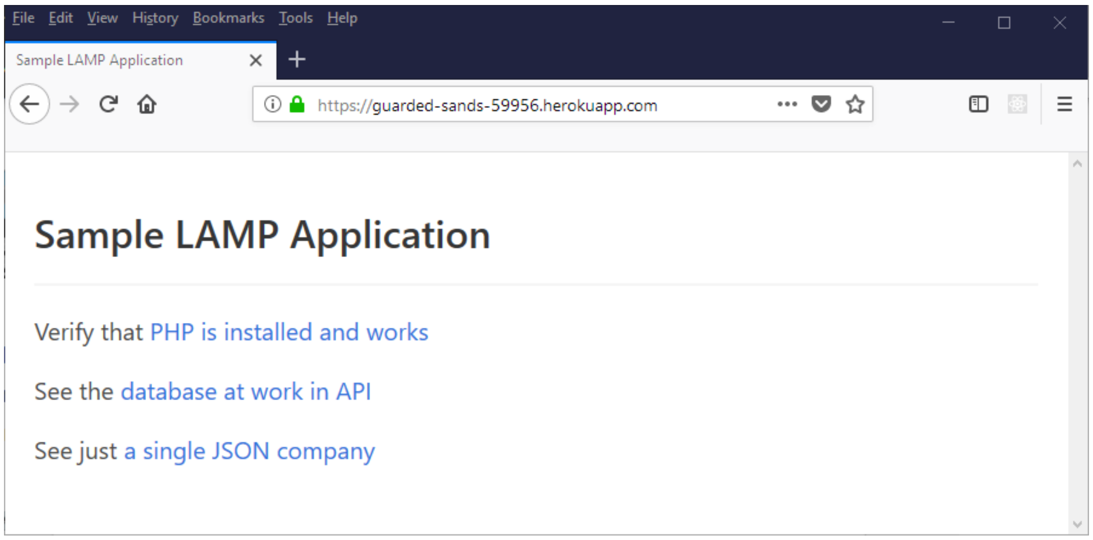
The result should look like that shown in Figure 16a.2.

10 Click on the first link to verify that PHP is installed on your site.

Everything should be working.

11 Go back and then click on the second link with the database.

This won't work. You need to provision a MySQL add-on and then modify the connection parameters to work with it.


File Edit View History Bookmarks Tools Help
Sample LAMP Application
https://guarded-sands-59956.herokuapp.com

Sample LAMP Application

Verify that PHP is installed and works

See the database at work in API

See just a single JSON company

Figure 16a.2 – After pushing source code to Heroku

Exercise 16a.3 — Examining the Heroku Dashboard

1 The Heroku web site provides an easy way to examine any of your sites. Visit `https://dashboard.heroku.com/apps` in a browser.

You will see the site you created in the previous exercise.

2 In the dashboard, click on the domain for your new site. You will now see the details for your site. It should be using free dynos and, at present, have no add-ons installed.

Notice that you can rename your subdomain as well.

3 In a separate browser tab, visit `https://elements.heroku.com/addons`.

As you can see, there are multitudes of add-ons that can be added to any Heroku site. Search for JawsDB MySQL from the list on this page and click on it. Scroll down to the Plans and Pricing section.

As you can see, these add-ons can cost a lot of money. Compare what is provided with the different pricing levels. You can see that a MySQL install that runs on a solid-state drive, allows 1000s of connections, and dozens of GB of memory space will cost 1000s of dollars per month!

We will make use of the Kitefin Shared plan which is free. Examine its capabilities. Our database will be limited to 5 MB and only allow for 10 connections. Despite these limitations, this will be enough for our database (and indeed, almost all the databases used in this book).

Notice the command needed to install this add-on (it appears within a black box that looks like a terminal/command window). Copy that snippet onto the clipboard.

Exercise 16a.4 — Provisioning a MySQL Database

1 Heroku doesn't provide a MySQL database. Instead, it (actually you) relies on third-party add-ons for databases, logging, caching, and other services needed by most web applications. You can perform these actions via the heroku web site or via the CLI.

In the terminal/command window, enter the following command (or paste from the clipboard):

```
heroku addons:create jawsdb:kitefin
```

This will provision the free MYSQL add-on in your application.

2 Enter the following command:

```
heroku config:get JAWSDB_URL
```

This will show you the connection string needed to connect to this database. It is in the format: mysql://username:password@hostname:port/dbname

3 Enter the following command:

```
heroku config -s | grep JAWSDB_URL >> .env
```

This will store the config information inside a hidden file named .env.

4 To import data into this new database, you will need access to the mysql shell. If you have it installed on your local development machine, you can run the following command from the terminal/command line (but replace NEWHOST, NEWUSER, NEWPASS, and NEWDATABASE with the values you saw in step 2. Note: there is no space between the -p and the password value.

```
mysql -h NEWHOST -u NEWUSER -pNEWPASS NEWDATABASE < companies.sql
```

If you don't have mysql locally, but have access from another environment (for instance, something like cloud9), you should be able to run the same command in that environment's terminal.

If this works, you have successfully imported data into the JawsDB MySQL add-on.

5 Edit `stock-config.inc.php` in the `includes` folder as follows:

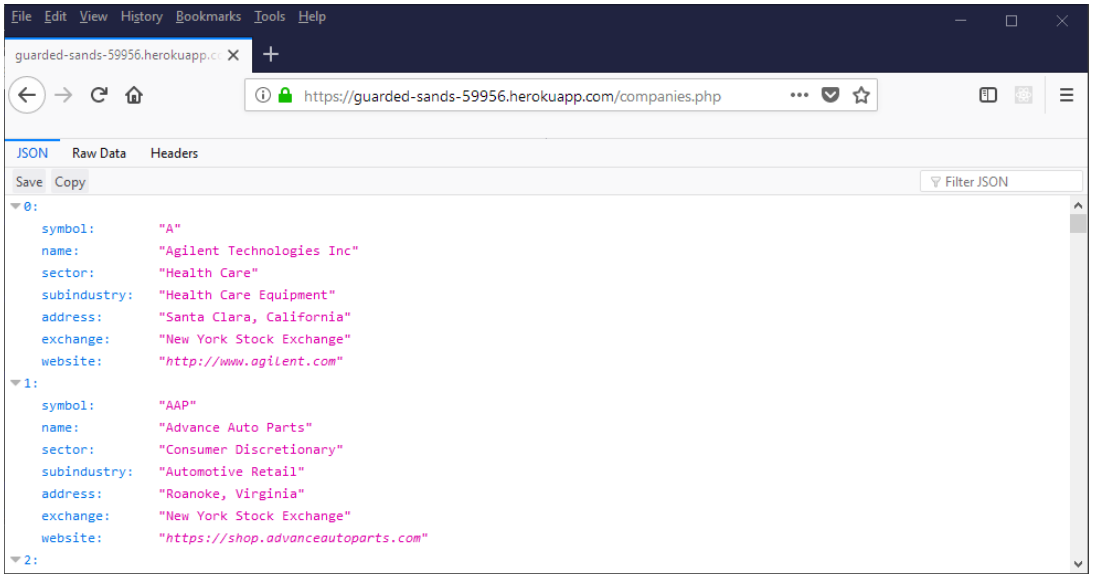
```
// you may need to change these for your own environment
$url = getenv('JAWSDB_URL');
$dbparts = parse_url($url);

$hostname = $dbparts['host'];
$username = $dbparts['user'];
$password = $dbparts['pass'];
$database = ltrim($dbparts['path'],'/');

define('DBCONNECTION', "mysql:host=$hostname;dbname=$database");
define('DBUSER', $username);
define('DBPASS', $password);
```

6 Save then in terminal, enter the following commands:

```
git add *
git commit -m "configured database"
git push heroku master
```

In order to get code changes uploaded to the server, you have to first add them to your local repo, commit them, and then git push them to heroku.

7 Test your site in the browser. Clicking on the Database at Work link should display the company information in JSON format (which has been retrieved from your newly-configured MySQL database.


File Edit View History Bookmarks Tools Help
guarded-sands-59956.herokuapp.c
https://guarded-sands-59956.herokuapp.com/companies.php
JSON Raw Data Headers
Save Copy
Filter JSON

```
▾0:
    symbol:        "A"
    name:          "Agilent Technologies Inc"
    sector:        "Health Care"
    subindustry:   "Health Care Equipment"
    address:       "Santa Clara, California"
    exchange:      "New York Stock Exchange"
    website:       "http://www.agilent.com"
▾1:
    symbol:        "AAP"
    name:          "Advance Auto Parts"
    sector:        "Consumer Discretionary"
    subindustry:   "Automotive Retail"
    address:       "Roanoke, Virginia"
    exchange:      "New York Stock Exchange"
    website:       "https://shop.advanceautoparts.com"
▾2:
```

Figure 16a.3 – With the database configured and working

PaaS Hosting using the Google App Engine

The Google Cloud Platform (GCP), like its competitors Amazon Web Services (AWS) and Microsoft Azure are full-featured cloud platforms. In this section you will use the GCP App Engine, which is somewhat equivalent to Heroku in that it is a simplified PaaS that is optimised for developers.

Exercise 16a.5 — Setting Up GCP

1. If you don't already have a Google account, you will need to create one. If you have been using gmail, then you already have one.

2. Visit `https://cloud.google.com`. Sign-in with your Google account. Ideally, you already have access to free credits supplied by your instructor or institution.

   You can work with GCP via the web-based GCP Console or via the gcloud CLI. You will begin with the web-based console.

3. All GCP resources and services you use must belong to a project.

   To begin, create a new GCP project for this lab. At the upper-left of the console, there is a drop-down list labelled **Select a project**. Choose this option. From the resulting dialog, click the New Project button.

   Give the project a meaningful name (I called mine `Lab16a-March2019`) and click Create.

   It might take some time (5-10 minutes) for GCP to create the project.

4. Use the Project selector at the top of the GCP console to select your project you just created.

   This will take you to the Dashboard for the project.

5. You can also interact with GCP via the Google Cloud SDK, which will install the gcloud CLI.

   You now need to install the GCP CLI. Navigate to `https://cloud.google.com/sdk` and choose the appropriate installer.

6. The installer should automatically run the **`gcloud init`** command (if it hasn't then enter it manually from the terminal/command window).

   It will then ask you to log in. This login will then happen within a web browser. As part of the Google login, you will be asked about sharing information with Google. After the login, control will return to the terminal, where you will be asked to specify a project (if you have more than one). If everything works, configuration will be completed and control will return to the terminal prompt.

7. Verify your gcloud is working correctly via the following command:

   ```
   gcloud --version
   ```

8 Using the terminal, navigate to the folder mentioned in step 1.

9 In the web GCP console, use the hamburger menu in top left corner and choose the **App Engine** option. Click the Create Application button.

10 Choose the appropriate region for your site. Click the **Create App** button.

11 From the Language drop-down, select PHP and Standard environment. Click Next.

12 The application will then be ready for you to deploy the source code. Let's skip this step for now and simply choose the **I'll do it later** option.

Exercise 16a.6 — Setting Up MySQL on GCP

1 Now let's setup the MySQL database. Click on the hamburger menu and choose **SQL** option.

Choose the **Create instance** button. Choose **MySQL**.

2 Provide an Instance ID of **`testsql`** and a root password. Be sure to remember these. Click Create.

There will be a delay.

3 Once completed, click on the MySQL instance. From the Instance details screen, click on the Databases tab.

Click on the Create database button. Name the database **`tester`**.

Click on the Import button

4 We need to import the SQL data into this table. To do so, we first need to upload the `companies.sql` file to Cloud Storage.

Click on the hamburger menu and Choose the Storage option. Select the bucket corresponding to your project. Click the Upload button and select the `companies.sql` file.

5 Return to the MySQL instance (click on SQL from hamburger menu then click on your instance).

Click on the Import button at the top of the page. Choose the `tester` database. Use the Cloud Storage file browser to select the `companies.sql` file you just uploaded. Make sure the import format is SQL. Click Import button.

6 After the import, you can now access this database using the `mysql` shell. If you have it (mysql) installed on your development machine, try the following command from the terminal.

```
gcloud sql connect testsql
```

If you do **not** have mysql on your development machine, you can still access it via gcloud terminal available within the GCP web console. In the upper right part of the page, you will see the Activate Cloud Icon >_. Click on that and you will be able to enter the above gcloud command.

7 In the `mysql` shell, enter the following commands:

```
show databases;
use tester;
show tables;
select * from companies;
exit
```

Hopefully everything works!

Exercise 16a.7 — Configuring Source Code for GCP

1 We have to make a few file changes so our PHP runs on the GCP.

Recall you needed both a Heroku version of the starting folder and a Google Cloud Platform (GCP) version of the starting folder. In your gcp folder, create an empty file named `composer.json`.

2 In the root folder, create a file named `app.yaml` with the following content.

```
runtime: php55
api_version: 1

handlers:
# Serve images as static resources
- url: /(.+\.(gif|png|jpg))$
  static_files: \1
  upload: .+\.(gif|png|jpg)$


# Serve images as static resources
- url: /(.+\.(htm|html))$
  static_files: \1
  upload: .+\.(htm|html)$


# Serve php scripts
- url: /(.+\.php)$
  script: \1

- url: /.*
  script: index.php
```

This file is used to configure the App Engine settings. This file specifies how URL paths correspond to request handlers and static files. The app.yaml file also contains information about your app's code, such as the runtime and the latest version identifier.

3 Using the terminal, navigate to the folder mentioned in step 1.

4 Our application is ready for deployment to the server. Enter the following command in the terminal.

```
gcloud init
```

Choose the default options.

5 Enter the following command.

```
gcloud app deploy
```

This command uploads the files to the virtual GCP server. This will take a few minutes.

6 To view content in browser, enter the following command:

```
gcloud app browse
```

In your browser, you should be able to see something similar to that shown in Figure 16a.4. The database options won't work yet.

Exercise 16a.8 — Using Your Database

1 We have to make a few file changes so our PHP can access the database.

Add the following to your app.yaml file.

```
runtime: php55
api_version: 1

env_variables:
   MYSQL_USER: "root"
   MYSQL_PASSWORD: "<password>"
   MYSQL_DSN: mysql:dbname=tester;unix_socket=/cloudsql/<instance>

beta_settings:
    cloud_sql_instances: "<instance>"
```

You will need to replace `<password>` with your password created in Step 2 of Exercise16a.6. A better approach would be to create a separate MySQL user account for this site and use that instead of the root user. This is left as a future independent exercise for the reader.

You will also need to replace `<instance>` with the Instance connection name, which can be found in the Overview tab for your SQL instance (see Figure 16a.4)

2 Edit `stock-config.inc.php` in the `includes` folder as follows.

```
// you may need to change these for your own environment
define('DBCONNECTION', getenv('MYSQL_DSN') );
define('DBUSER', getenv('MYSQL_USER') );
define('DBPASS', getenv('MYSQL_PASSWORD') );
```

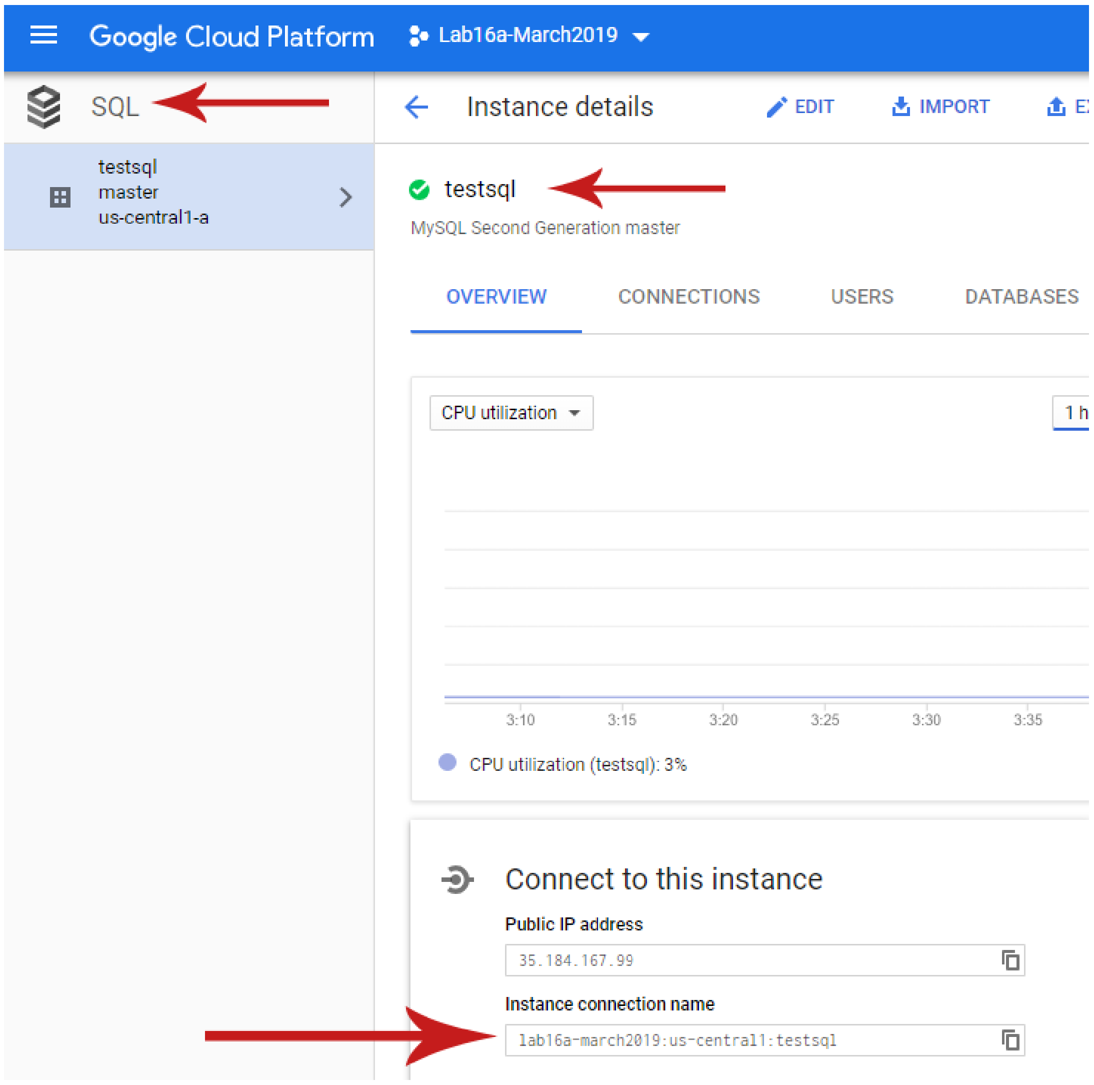
3 In the terminal, enter the following commands:

```
gcloud app deploy
gcloud app browse
```

Anytime you make a change to your code, you will have to upload it to the GCP server via `gcloud app deploy`.

4 Test your files in the browser.

The database pages should work correctly.



testsql

MySQL Second Generation master

CPU utilization (testsql): 3%

Connect to this instance

Public IP address

35.184.167.99

Instance connection name

lab16a-march2019:us-central1:testsql

Figure 16a.4 – Finding the Instance connection name

Exercise 16a.9 — Removing Your GCP Instances

1 SQL and App Engine instances will use up GCP credits over time, even if they aren't being used/requested. When you are satisfied with your progress (and you don't need them for future course assignments or exercises), you may want to delete both to prevent credit depletion over time.