

Download the Code

Click the command line area in the Cloud Shell so you can type commands.

Fetch the code from Github and then change to the code folder:

```
git clone https://github.com/googlecode-labs/user-authentication-with-iap.git
content_copy
cd user-authentication-with-iap
content_copy
```

This folder contains one subfolder for each step of this lab. You will change to the correct folder to perform each step.

Deploy Application and Protect it with IAP

This is an App Engine Standard application written in Python that simply displays a "Hello, World" welcome page. We will deploy and test it, then restrict access to it using IAP.

Review the Application Code

Change from the main project folder to the 1-HelloWorld subfolder that contains code for this step.

```
cd 1-HelloWorld
content_copy
```

The application code is in the main.py file. It uses the [Flask](#) web framework to respond to web requests with the contents of a template. That template file is in templates/index.html, and for this step contains only plain HTML. A second template file contains a skeletal example privacy policy in templates/privacy.html.

There are two other files: `requirements.txt` lists all the non-default Python libraries the application uses, and `app.yaml` tells Google Cloud that this is a Python App Engine application.

You can list each file in the shell using the `cat` command, as in:

```
cat main.py
content_copy
```

Or you can launch the Cloud Shell code editor by clicking the Pencil icon at the top right-hand side of the Cloud Shell window, and examine the code that way.

You do not need to change any files for this step.

Deploy to App Engine

Deploy the app to the App Engine Standard environment for Python.

```
gcloud app deploy
content_copy
```

Select a region near to you that says it "supports standard".

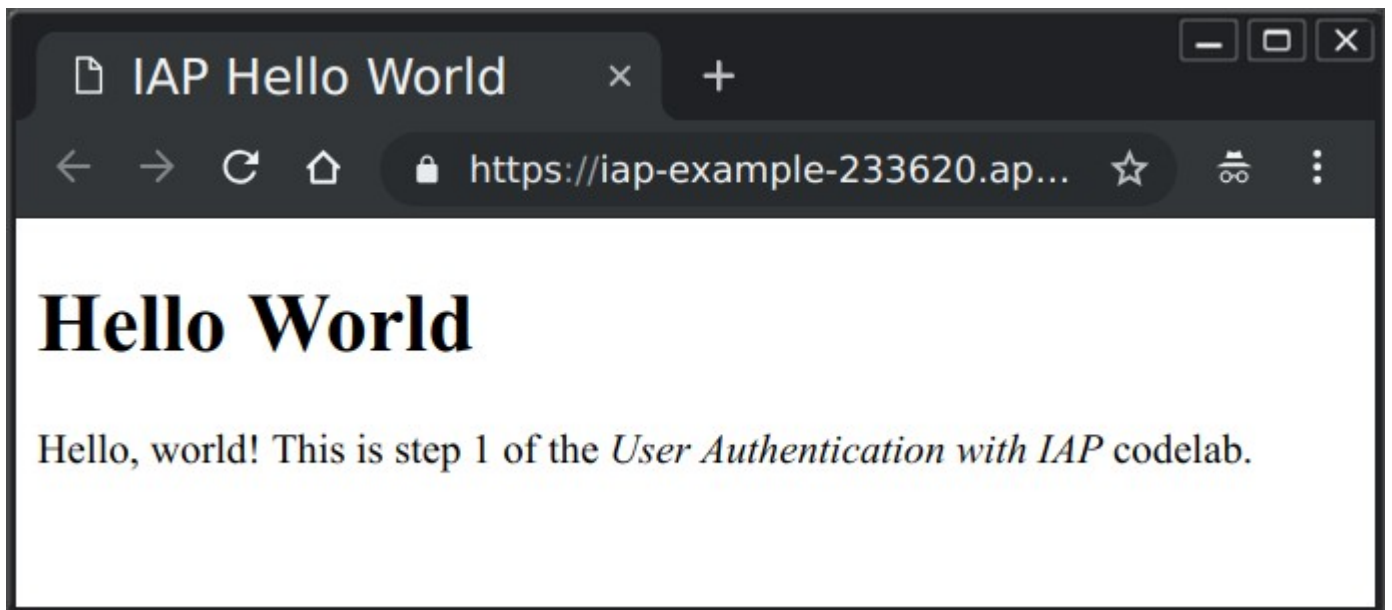
When you are asked if you want to continue, enter **Y** for yes.

In a few minutes the deploy completes. You will see a message that you can view your application with `gcloud app browse`.

Enter that command:

```
gcloud app browse
content_copy
```

Click the displayed link to open it in a new tab, or copy it to a manually opened new tab if necessary. Since this is the first time this app is run, it will take a few seconds to appear while a cloud instance is started, and you should see the following window.

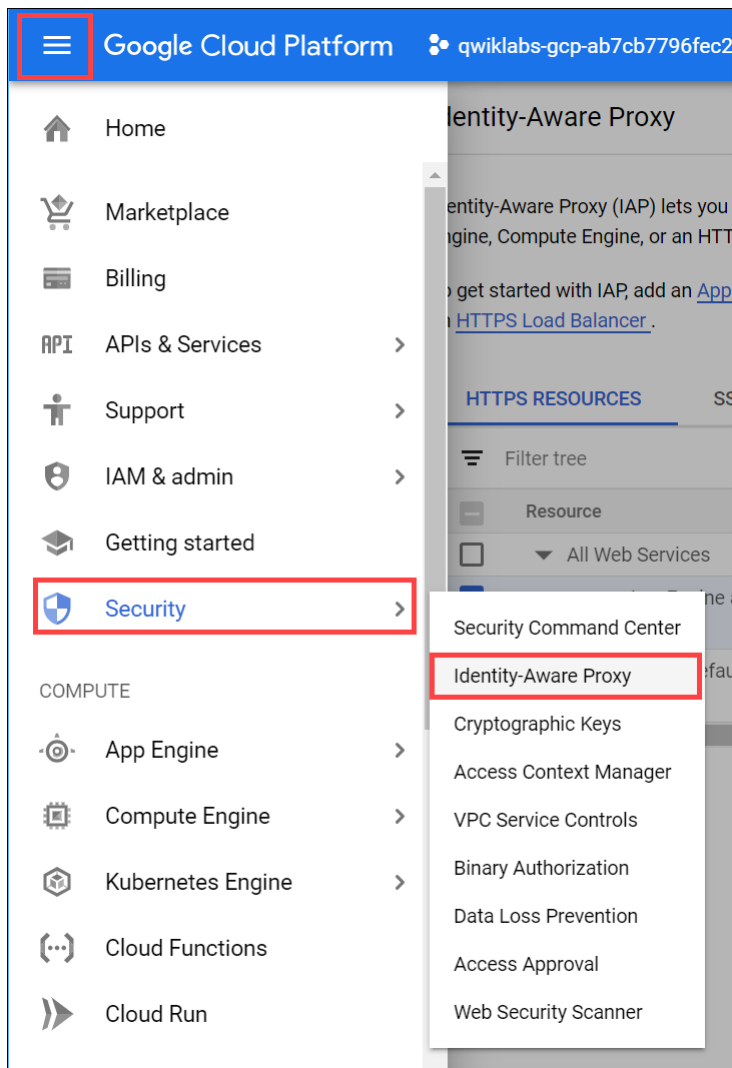


You can open that same URL from any computer connected to the Internet to see that web page. Access is not yet restricted.

Click Check my progress to verify the objective.

Restrict Access with IAP

1. In the cloud console window, click the **Navigation menu** > **Security** > **Identity-Aware Proxy**.



2. Click **ENABLE API**.

Identity-Aware Proxy

The Identity-Aware Proxy(Cloud IAP) controls access to your cloud applications and VMs running on Google Cloud Platform(GCP). [Learn more](#)



Identity-Aware Proxy API is not enabled.

[ENABLE API](#)

3.Click **GO TO IDENTITY-AWARE PROXY**.

Identity-Aware Proxy

The Identity-Aware Proxy(Cloud IAP) controls access to your cloud applications and VMs running on Google Cloud Platform(GCP). [Learn more](#)

[GO TO IDENTITY-AWARE PROXY](#)

4.Click **CONFIGURE CONSENT SCREEN**.



You may now put IAP in front of your on-prem/non-GCP resources using our IAP Connector. [Learn more](#).

DISMISS

HTTPS RESOURCES

SSH AND TCP RESOURCES



Before you can use IAP, you need to configure your OAuth consent screen.

CONFIGURE CONSENT SCREEN

5. Select **Internal** under User Type and click **Create**.

6. Fill in the required blanks with appropriate values:

| Field | Value |
|---------------------------------|--|
| App name | IAP Example |
| User support email | <i>Your email address. it may already be filled in for you.</i> |
| Application home page | <i>The URL you used to view your app</i> |
| Application privacy Policy link | <i>The privacy page link in the app, same as the homepage link with /privacy added to the end</i> |
| Authorized domains | <i>Click + ADD DOMAIN The hostname portion of the application's URL, e.g. iap-example-999999.appspot.com. You can see this in the address bar of the Hello World web page you previously opened. Do not include the starting https:// or trailing / from that URL.</i> <i>You must press Enter after filling in this value.</i> |
| Developer Contact Information | <i>Enter at least one email</i> |

7. Click **SAVE AND CONTINUE**. For **Scopes**, click **SAVE AND CONTINUE** and for **Summary** click **BACK TO DASHBOARD**. You will be prompted to create credentials. You do not need to create credentials for this lab, so you can simply close this browser tab.

8. In Cloud Shell, run this command to disable the Flex API:

```
gcloud services disable appengineflex.googleapis.com
```

[content copy](#)

9. Return to the Identity-Aware Proxy page and refresh it. You should now see a list of resources you can protect.

Click the toggle button in the IAP column in the App Engine app row to turn **IAP** on.

HTTPS RESOURCES

SSH AND TCP RESOURCES

Filter tree

?

| <div></div> Resource <div>IAP ?</div> <div>Method</div> <div>Published ?</div> <div>Status ?</div> |
|--|
| <div></div> <div>▼ All Web Services</div> |
| <div> <div></div> <div>▼ App Engine app</div> <div> <div></div> <div></div> </div> <div>IAM</div> <div>https://qwiklabs-gcp-02-6302b5c11e7a.uc.r.appspot.com</div> <div> <div>! Error</div> <div>⋮</div> </div> </div> |
| <div> <div></div> <div>▶ default</div> <div>https://qwiklabs-gcp-02-6302b5c11e7a.uc.r.appspot.com</div> </div> |

10. The domain will be protected by IAP. Click **TURN ON**.

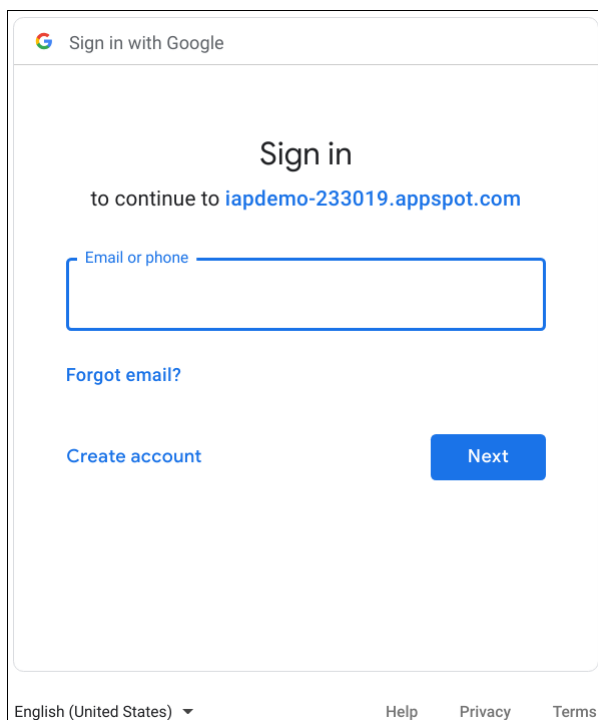
Turn on IAP

This will only allow access to App Engine app by members listed in the permission panel.

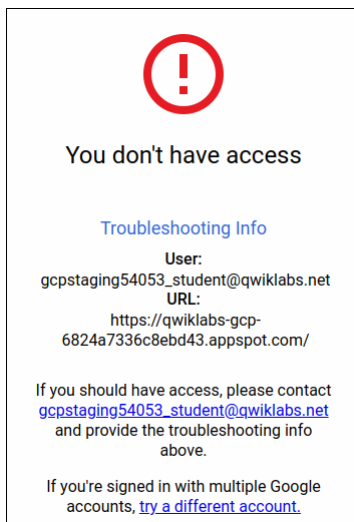
[CANCEL](#) [TURN ON](#)

Test that IAP is turned on

1. Open a browser tab and navigate to the URL for your app. A Sign in with Google screen opens and requires you to log in to access the app.

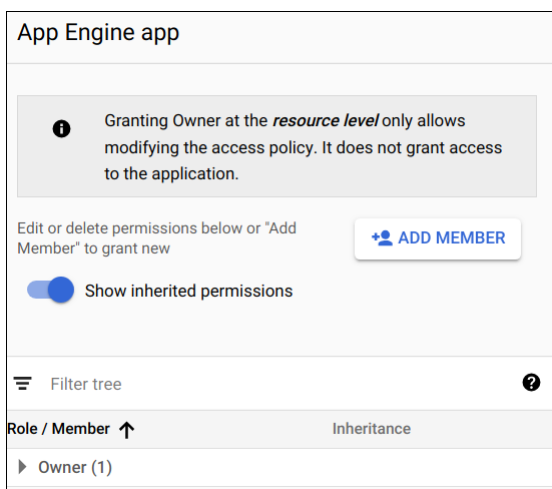
A screenshot of the 'Sign in with Google' authentication screen. The header shows the Google logo and the text 'Sign in with Google'. The main heading is 'Sign in', followed by the text 'to continue to iapdemo-233019.appspot.com'. Below this is a text input field with the placeholder 'Email or phone'. To the left of the input field is a link 'Forgot email?'. Below the input field is a link 'Create account' and a blue 'Next' button. At the bottom, there is a language selector 'English (United States)' with a dropdown arrow, and links for 'Help', 'Privacy', and 'Terms'.

2. Sign in with the account you used to log into the console. You will see a screen denying you access.



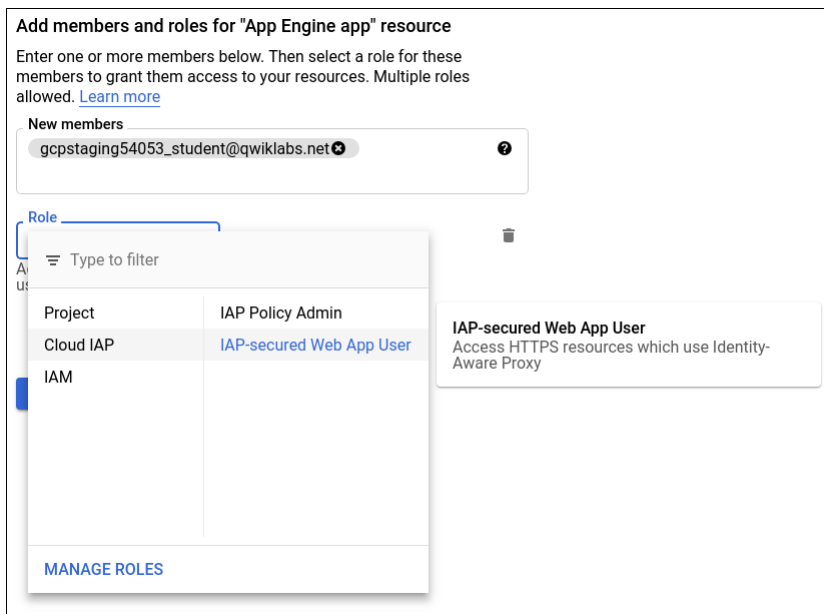
You have successfully protected your app with IAP, but you have not yet told IAP which accounts to allow through.

3. Return to the Identity-Aware Proxy page of the console, select the checkbox next to App Engine app, and see the sidebar at the right of the page.



Each email address (or Google Group address, or GSuite domain name) that should be allowed access needs to be added as a Member.

4. Click **ADD MEMBER**. Enter your email address, then pick the **Cloud IAP/IAP-Secured Web App User** role to assign to that address. You may enter more addresses or GSuite domains in the same way.



5. Click **SAVE**.

The message "Policy Updated" will appear at the bottom of the window.

Click Check my progress to verify the objective.

Test access

Navigate back to your app and reload the page. You should now see your web app, since you already logged in with a user you authorized.

If you still see the "You don't have access" page, IAP did not recheck your authorization. In that case, do the following steps:

1. Open your web browser to the home page address with `/_gcp_iap/clear_login_cookie` added to the end of the URL, as in `https://iap-example-999999.appspot.com/_gcp_iap/clear_login_cookie`.
2. You will see a new Sign in with Google screen, with your account already showing. Do not click the account. Instead, click Use another account, and re-enter your credentials.

These steps cause IAP to recheck your access and you should now see your application's home screen.

If you have access to another browser or can use Incognito Mode in your browser, and have another valid GMail or GSuite account, you can use that browser to navigate to your app page and log in with the other account. Since that account has not been authorized, it will see the "You Don't Have Access" screen instead of your app.

Access User Identity Information

Once an app is protected with IAP, it can use the identity information that IAP provides in the web request headers it passes through. In this step, the application will get the logged-in user's email address and a persistent unique user ID assigned by the Google Identity Service to that user. That data will be displayed to the user in the welcome page.

This is the last step ended with your Cloud Shell open in the `iap-codelab/1-HelloWorld` folder. Change to the folder for this step:

```
cd ~/user-authentication-with-iap/2-HelloUser
content_copy
```

Deploy to App Engine

Since deployment takes a few minutes, start by deploying the app to the App Engine Standard environment for Python:

```
gcloud app deploy
content_copy
```

When you are asked if you want to continue, enter **Y** for yes.

In a few minutes the deploy should complete. While you are waiting you can examine the application files as described below.

Click [Check my progress](#) to verify the objective.

Examine the Application Files

This folder contains the same set of files as seen in 1-HelloWorld, but two of the files have been changed: `main.py` and `templates/index.html`. The program has been changed to retrieve the user information that IAP provides in request headers, and the template now displays that data.

There are two lines in `main.py` that get the IAP-provided identity data:

```
user_email = request.headers.get('X-Goog-Authenticated-User-Email')
user_id = request.headers.get('X-Goog-Authenticated-User-ID')
content_copy
```

The **X-Goog-Authenticated-User-** headers are provided by IAP, and the names are case-insensitive, so they could be given in all lower or all upper case if preferred. The `render_template` statement now includes those values so they can be displayed:

```
page = render_template('index.html', email=user_email, id=user_id)
content_copy
```

The `index.html` template can display those values by enclosing the names in doubled curly braces:

```
Hello, {{ email }}! Your persistent ID is {{ id }}.
content_copy
```

As you can see, the provided data is prefixed with `accounts.google.com`, showing where the information came from. Your application can remove everything up to and including the colon to get the raw values if desired.

Test the updated IAP

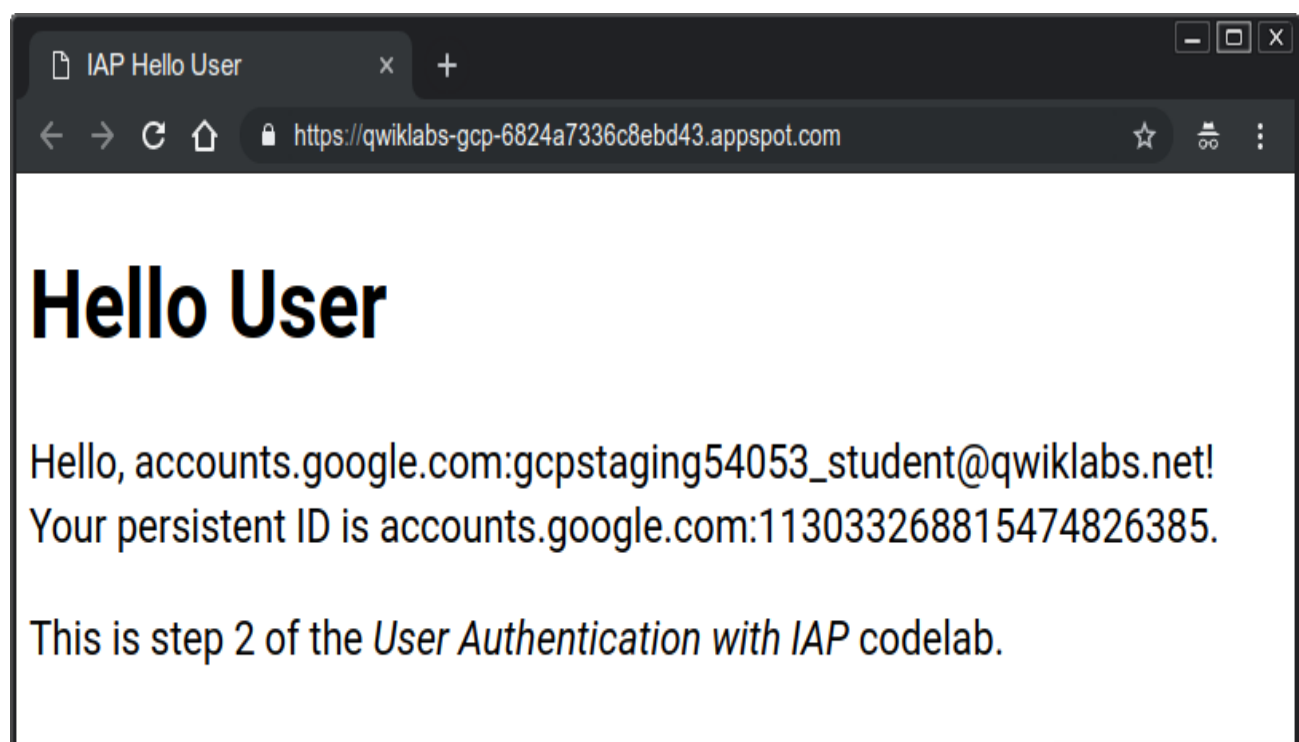
Going back to the deployment, when it is ready, you will see a message that you can view your application with gcloud app browse.

1. Enter that command.

```
gcloud app browse
```

```
content_copy
```

2. If a new tab does not open on your browser, copy the displayed link and open it in a new tab normally. You should see a page similar to the following:



You may need to wait a few minutes for the new version of your application to replace the prior version. Refresh the page if needed to see a page similar to the above.

Turn off IAP

What happens to this app if IAP is disabled, or somehow bypassed (such as by other applications running in your same cloud project)? Turn off IAP to see.

In the cloud console window, click **Navigation menu > Security > Identity-Aware Proxy**. Click the **IAP** toggle switch next to App Engine app to turn **IAP** off.

HTTPS RESOURCES

SSH AND TCP RESOURCES

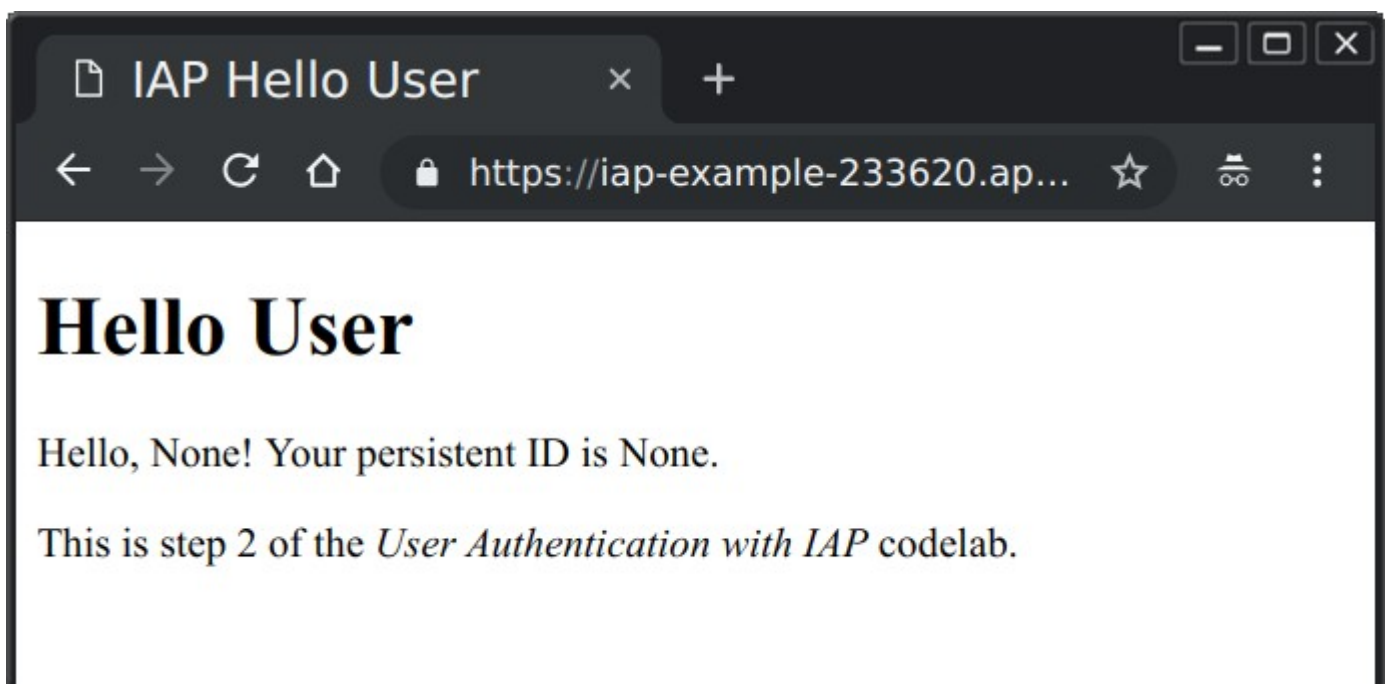
Filter tree

?

| <input type="checkbox"/> | Resource | IAP ? | Published ? | Configuration ? |
|--------------------------|--------------------|--------------------------|------------------------------------|--------------------------------|
| <input type="checkbox"/> | ▼ All Web Services | | | |
| <input type="checkbox"/> | ▼ App Engine app | <input type="checkbox"/> | https://iapdemo-233019.appspot.com | OK ⋮ |
| <input type="checkbox"/> | ▶ default | | https://iapdemo-233019.appspot.com | |

You will be warned that this will allow all users to access the app.

Refresh the application web page. You should see the same page, but without any user information:



Since the application is now unprotected, a user could send a web request that appeared to have passed through IAP. For example, you can run the following curl command from the Cloud Shell to do that (replace <your-url-here> with the correct URL for your app):

```
curl -X GET <your-url-here> -H "X-Goog-Authenticated-User-Email: totally fake email"
content_copy
```

The web page will be displayed on the command line, and look like the following (do not copy):

```
<!doctype html>
<html>
<head>
  <title>IAP Hello User</title>
</head>
<body>
  <h1>Hello World</h1>

  <p>
    Hello, totally fake email! Your persistent ID is None.
  </p>
  <p>
    This is step 2 of the <em>User Authentication with IAP</em>
    codelab.
  </p>
</body>
</html>content_copy
```

There is no way for the application to know that IAP has been disabled or bypassed. For cases where that is a potential risk, Cryptographic Verification shows a solution.

Use Cryptographic Verification

If there is a risk of IAP being turned off or bypassed, your app can check to make sure the identity information it receives is valid. This uses a third web request header added by IAP, called X-Goog-IAP-JWT-Assertion. The value of the header is a cryptographically signed object that also contains the user identity data. Your application can verify the digital signature and use the data provided in this object to be certain that it was provided by IAP without alteration.

Digital signature verification requires several extra steps, such as retrieving the latest set of Google public keys. You can decide whether your application needs these extra steps based on the risk that someone might be able to turn off or bypass IAP, and the sensitivity of the application.

The last step ended with your Cloud Shell open in the `iap-codelab/2-HelloUser` folder. Change to the folder for this step:

```
cd ~/user-authentication-with-iap/3-HelloVerifiedUser
content_copy
```

Deploy to App Engine

Deploy the app to the App Engine Standard environment for Python:

```
gcloud app deploy
content_copy
```

When you are asked if you want to continue, enter **Y** for yes. In a few minutes the deploy should complete. While you are waiting you can examine the application files as described below.

Click [Check my progress](#) to verify the objective.

Examine the Application Files

This folder contains the same set of files as seen in `2-HelloUser`, with two files altered and one new file. The new file is `auth.py`, which provides a `user()` method to retrieve and verify the cryptographically signed identity information. The changed files are `main.py` and `templates/index.html`, which now use the results of that method. The unverified headers as found in the last deployment are also shown for comparison.

The new functionality is primarily in the `user()` function:

```
def user():
    assertion = request.headers.get('X-Goog-IAP-JWT-Assertion')
```



```
if assertion is None:
    return None, None

info = jwt.decode(
    assertion,
    keys(),
    algorithms=['ES256'],
    audience=audience()
)

return info['email'], info['sub']
```

[content copy](#)

The assertion is the cryptographically signed data provided in the specified request header. The code uses a library to validate and decode that data. Validation uses the public keys that Google provides for checking data it signs, and knowing the audience that the data was prepared for (essentially, the Google Cloud project that is being protected). Helper functions `keys()` and `audience()` gather and return those values.

The signed object has two pieces of data we need: the verified email address, and the unique ID value (provided in the `sub`, for subscriber, standard field).

This completes Step 3.

Test the Cryptographic Verification

When the deployment is ready you will see a message that you can view your application with `gcloud app browse`.

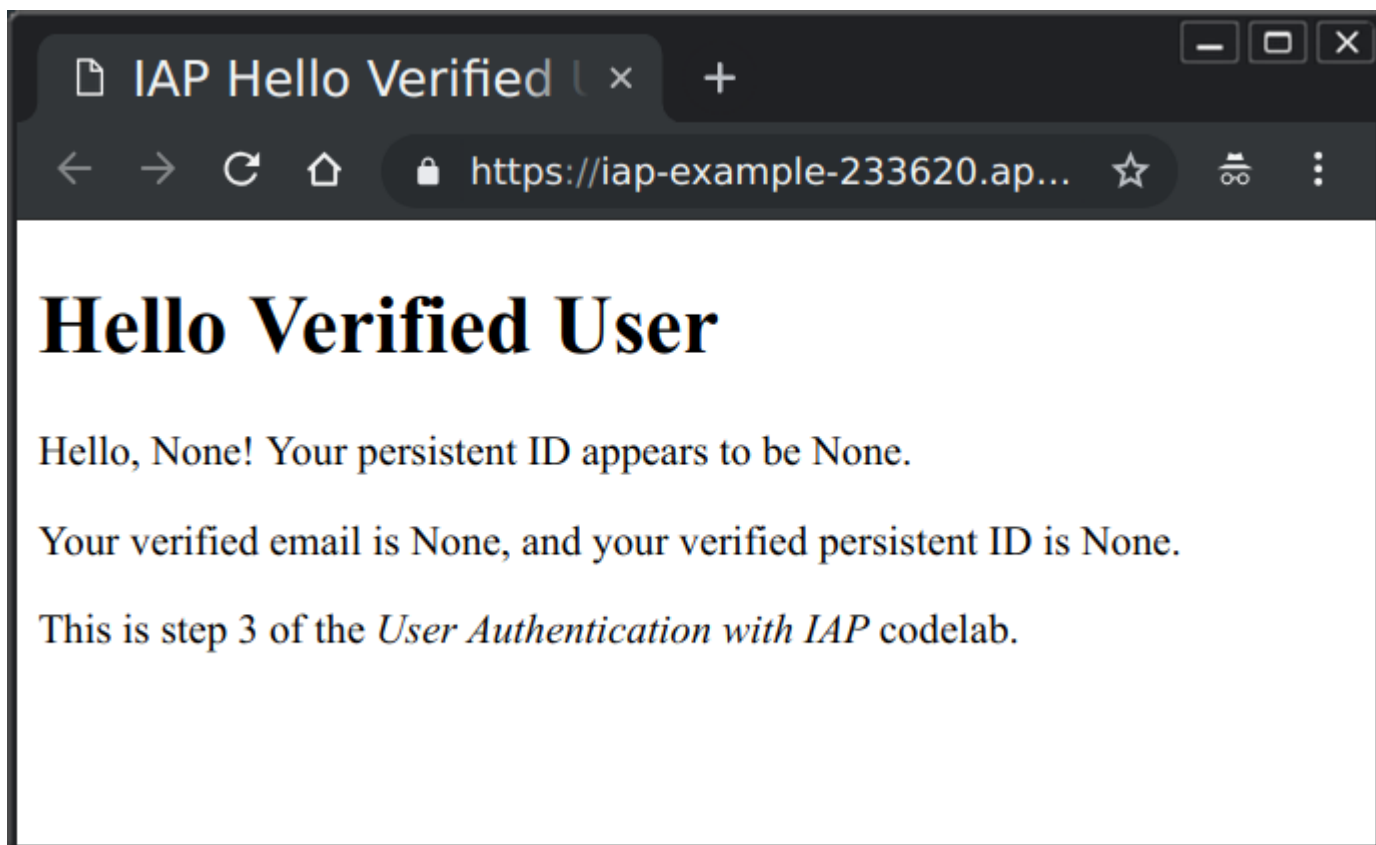
Enter that command:

```
gcloud app browse
```

[content copy](#)

If a new tab does not open on your browser, copy the displayed link and open it in a new tab normally.

Recall that you previously disabled IAP, so the application provides no IAP data. You should see a page similar to the following:



As before, you may need to wait a few minutes for the newest version to be live to see the new version of the page.

Since IAP is disabled, no user information is available. Now turn IAP back on.

1. In the cloud console window, click the **Navigation menu > Security > Identity-Aware Proxy**.

2. Click the **IAP** toggle switch next to App Engine app to turn IAP on again.

HTTPS RESOURCES

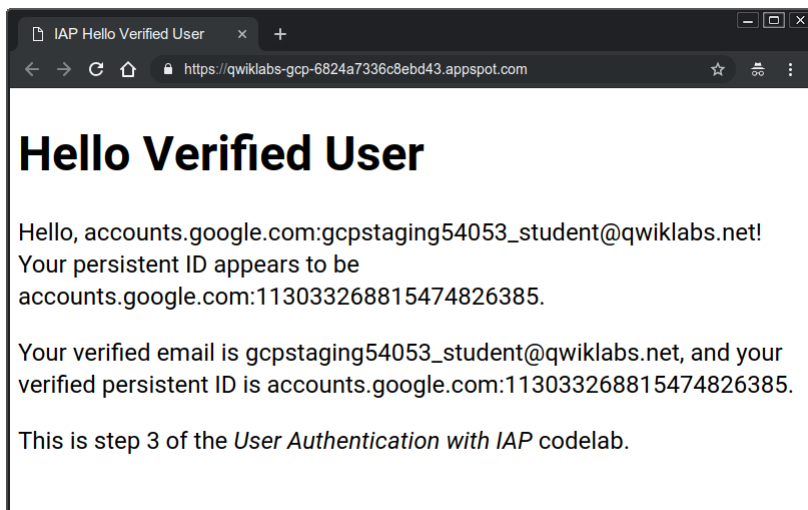
SSH AND TCP RESOURCES

Filter tree

?

| <input type="checkbox"/> | Resource | IAP ? | Published ? | Configuration ? |
|--------------------------|--------------------|--------------------------|------------------------------------|--|
| <input type="checkbox"/> | ▼ All Web Services | | | |
| <input type="checkbox"/> | ▼ App Engine app | <input type="checkbox"/> | https://iapdemo-233019.appspot.com | <input checked="" type="checkbox"/> OK |
| <input type="checkbox"/> | ▶ default | | https://iapdemo-233019.appspot.com | |

3. Refresh the page. The page should look like the following:



Notice that the email address provided by the verified method does not have the accounts.google.com: prefix.

If IAP is turned off or bypassed, the verified data would either be missing, or invalid, since it cannot have a valid signature unless it was created by the holder of Google's private keys.

Congratulations!

You deployed an App Engine web application. First, you restricted access to the application to only users you chose. Then you retrieved and displayed the identity of users that IAP allowed access to your application, and saw how that information might be spoofed if IAP were disabled or bypassed. Lastly, you verified cryptographically signed assertions of the user's identity, which cannot be spoofed.



Finish your Quest

This self-paced lab is part of the Qwiklabs Quest [Security & Identity Fundamentals](#). A Quest is a series of related labs that form a learning path. Completing this Quest earns you the badge above, to recognize your achievement. You can make your badge (or badges) public and link to them in your online resume or social media account. [Enroll](#) in this Quest and get immediate completion credit if you've taken this lab. [See other available Qwiklabs Quests](#).

Take your next lab

Continue your Quest with [Getting Started with Cloud KMS](#), or try one of these suggestions:

- [User Authentication with App Dev-Adding User Authentication to your Application - Python](#).
- [Securing Applications on Kubernetes Engine - Three Examples](#).

Next steps / learn more

- Get more detail about [Cloud Identity-Aware Proxy](#).
- Check out other [Google Security products](#).

License

This work is licensed under a Creative Commons Attribution 2.0 Generic License.

Google Cloud Training & Certification

...helps you make the most of Google Cloud technologies. [Our classes](#) include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. [Certifications](#) help you validate and prove your skill and expertise in Google Cloud technologies.

Manual Last Updated August 09, 2021

Lab Last Tested August 09, 2021

Copyright 2021 Google LLC All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated.