This lab sheet guides you through the steps to deploy a Flask application to Microsoft Azure Cloud. When the Flask application is successfully deployed, your Dialogflow agent can use it as a web service to enable fulfillment.

Before you proceed with this lab, please ensure that you have met the following pre-requisites.

- Eliminate bugs and errors in the Flask application
- Able to deploy the Flask application within your local environment
- The Dialogflow agent can call the Flask application

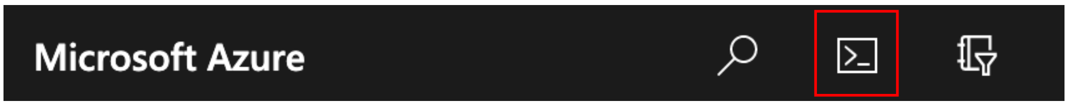
1. In your local development environment, make a copy of the existing Python program (e.g. **index.py**) and name it as **application.py**.
   Note: the naming is a MS Axzure convention.

2. Prepare a zip file **(webhook.zip)** that contains the required files needed to deploy your Flask application successfully. The compressed file should contain the following items:
   - .env
   - application.py
   - <keyfile>.json (application key file from Google)
   - requirements.txt

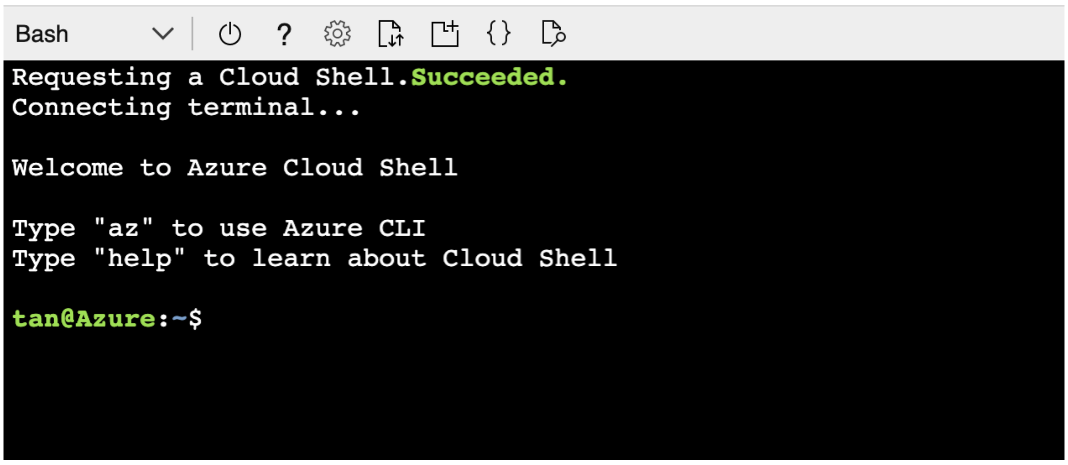
3. Create a free Microsoft Azure account at www.azure.microsoft.com.
   Note: *Please make sure that you are fully aware of charges that you may incur when using this service. The author of this lab sheet cannot take responsibility for any bills that are sent to you.*

4. After your account is created, go to the Azure portal at https://portal.azure.com .

5. Open the **Azure Cloud Shell** by selecting the **Cloud Shell button** on the menu found at the top of the Azure portal.



Within the browser, you should see the bash terminal window, similar to the example below.



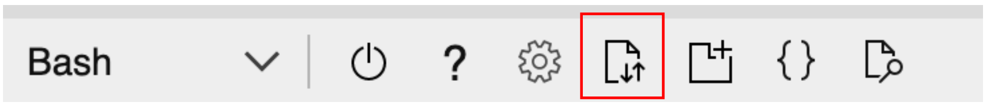
```
Requesting a Cloud Shell.Succeeded.
Connecting terminal...

Welcome to Azure Cloud Shell

Type "az" to use Azure CLI
Type "help" to learn about Cloud Shell

tan@Azure:~$
```

6. In the Cloud Shell, choose the **Upload button**.



Select the zip file that was created in a prior step.
Check that the file is successfully uploaded.

```
$ ls –l
```

7. Create a new **webhook** directory. Then, unzip the file to the new directory.

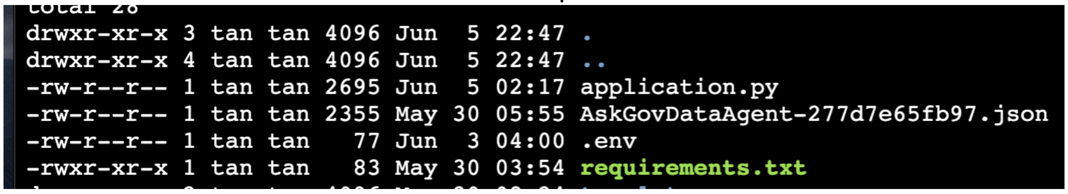
```
$ mkdir webhook
$ unzip -d webhook webhook.zip
```

Note: webhook is the destination directory name.

8. Change to the webhook directory. List the contents of the directory.

```
$ cd webhook
$ ls –la
```

It should contain items similar to the example below.

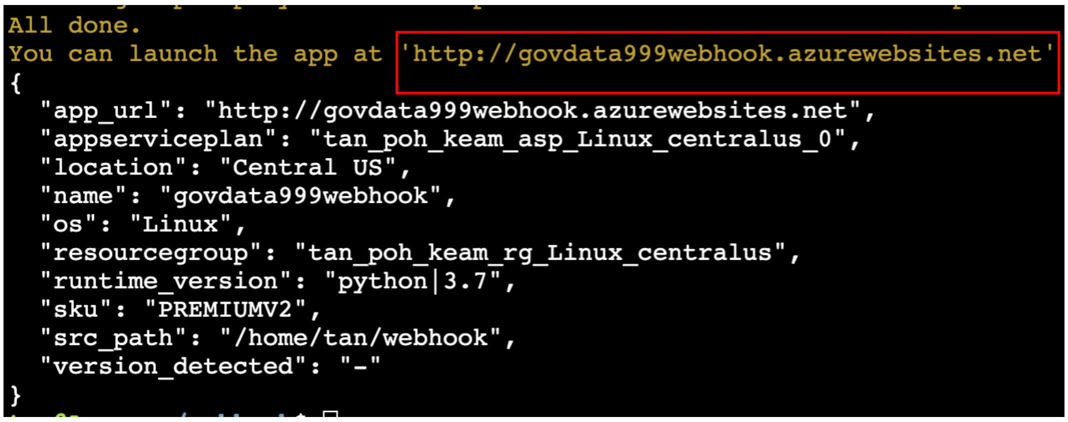


```
total 28
drwxr-xr-x 3 tan tan 4096 Jun  5 22:47 .
drwxr-xr-x 4 tan tan 4096 Jun  5 22:47 ..
-rw-r--r-- 1 tan tan 2695 Jun  5 02:17 application.py
-rw-r--r-- 1 tan tan 2355 May 30 05:55 AskGovDataAgent-277d7e65fb97.json
-rw-r--r-- 1 tan tan   77 Jun  3 04:00 .env
-rwxr-xr-x 1 tan tan   83 May 30 03:54 requirements.txt
```

9. Within the webhook directory, run the **az webapp up** command. Replace <app-name> with a unique app name. This command may take a few minutes to run.

```
$ az webapp up -n <app-name>
```

When completed, it should display information similar to the following example:



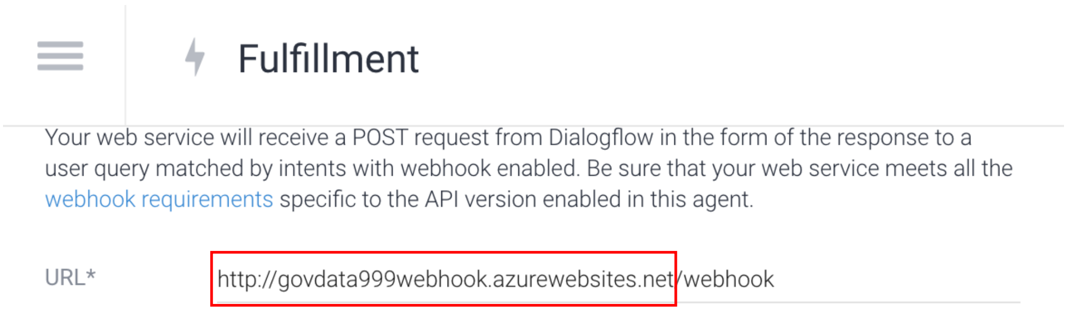
```
All done.
You can launch the app at 'http://govdata999webhook.azurewebsites.net'
{
  "app_url": "http://govdata999webhook.azurewebsites.net",
  "appserviceplan": "tan_poh_keam_asp_Linux_centralus_0",
  "location": "Central US",
  "name": "govdata999webhook",
  "os": "Linux",
  "resourcegroup": "tan_poh_keam_rg_Linux_centralus",
  "runtime_version": "python|3.7",
  "sku": "PREMIUMV2",
  "src_path": "/home/tan/webhook",
  "version_detected": "-"
}
```

The URL that points to your Flask Application is highlighted.

10. Browse to the deployed application using your web browser. If it is successfully deployed, you should see a web page.

11. From the Dialogflow Agent, navigate to the Fulfillment configuration. Replace the web service with the URL that is given by Azure. Retain the name of the route (e.g. /webhook).

    In the example below, the highlighted URL is provided by Azure.



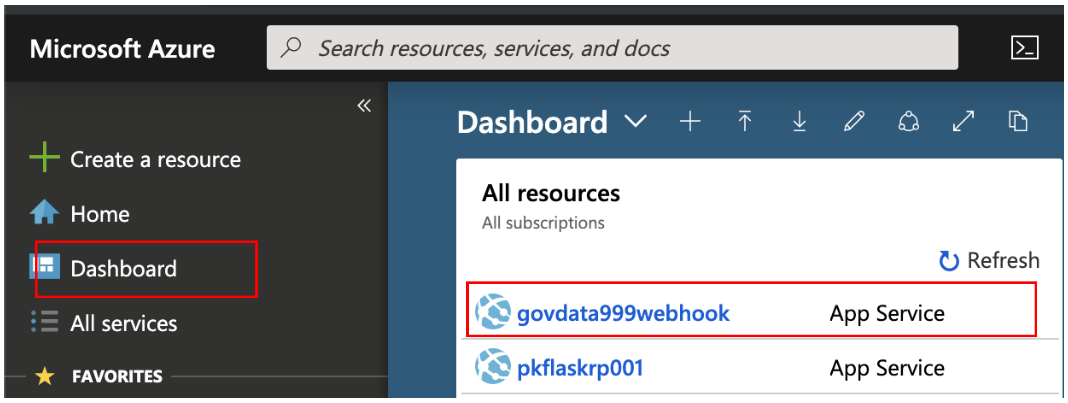
Fulfillment

Your web service will receive a POST request from Dialogflow in the form of the response to a user query matched by intents with webhook enabled. Be sure that your web service meets all the webhook requirements specific to the API version enabled in this agent.

URL* http://govdata999webhook.azurewebsites.net/webhook

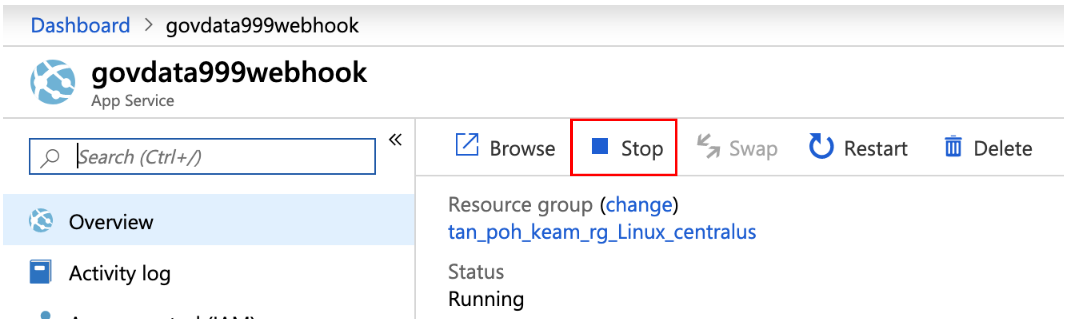
At this point, you should continue to test the Dialogflow agent in the usual way.

12. Note that leaving the app running in Azure may incur charges. Please monitor it using the Azure Dashboard and stop the app service when it is not required. The author of this lab sheet cannot take responsibility for any bills that are sent to you.



All resources
All subscriptions

govdata999webhook	App Service
pkflaskrp001	App Service



govdata999webhook
App Service

Resource group (change)
tan_poh_keam_rg_Linux_centralus

Status
Running