Patient Service Analysis Tool Deployment Manual

The following is a guide on every step needed to get the system up and running. This includes creating Power BI and Azure accounts, adding API keys to the config file, setting up the database and installing the required packages.

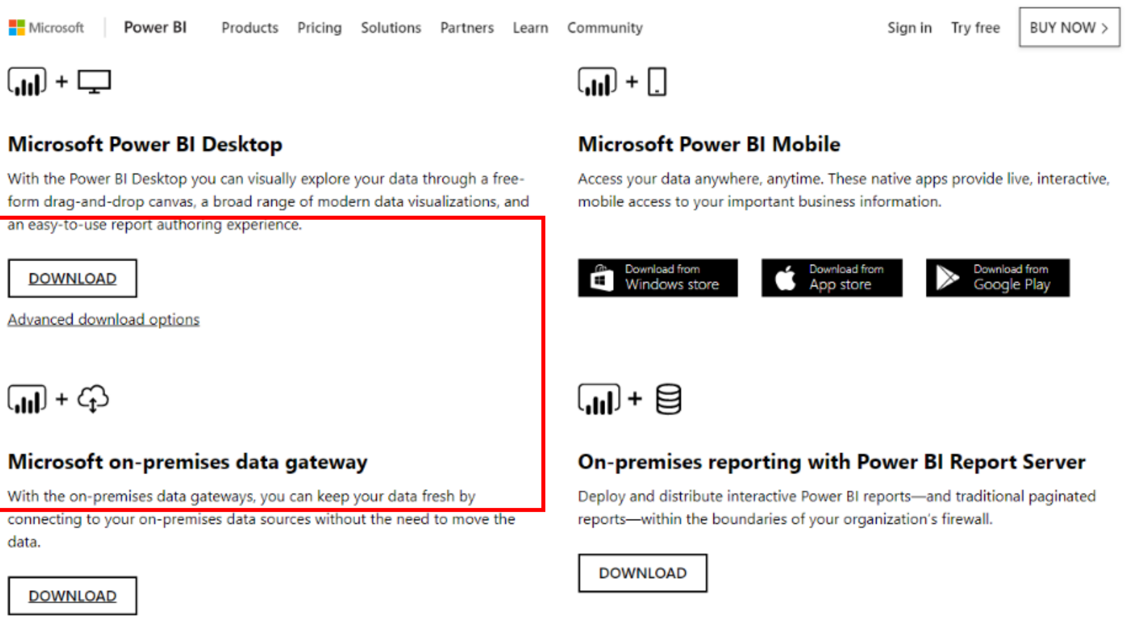
For information on how to refresh data, how to use Power BI, types of requests that can be made etc, please look at the User Manual. Before starting please be aware that in order to use Power BI, our visualisation tool of choice, you will need to be using windows, preferably windows 10.

The first steps in this manual are in regard to setting up Power BI and creating the required resources on Azure. The following sections are compulsory and must be completed in order for the system to function properly:

- Downloading and setting up Power BI
- Creating an Azure account
- Check for and create subscription if necessary
- Create a Resource Group
- Create MySQL database on Azure
- Creating Azure blob storage container
- Creating Azure Text Analytics API resource
- Installing Python and required python packages.
- Adding required details to config.json file
- Request.py
- Running tests and checking everything is working

Of the last 2 sections only one of them needs to be completed. You can choose to either set up the API to run on localhost or create a web app on Azure to host the API. The latter option comes with some restrictions that are further explained in the section on hosting the API on Azure.

<u>Downloading and setting up Power BI</u>



Microsoft Power BI Desktop

With the Power BI Desktop you can visually explore your data through a free-form drag-and-drop canvas, a broad range of modern data visualizations, and an easy-to-use report authoring experience.

DOWNLOAD

Advanced download options

Microsoft Power BI Mobile

Access your data anywhere, anytime. These native apps provide live, interactive, mobile access to your important business information.

Download from Windows store | Download from App store | Download from Google Play

Microsoft on-premises data gateway

With the on-premises data gateways, you can keep your data fresh by connecting to your on-premises data sources without the need to move the data.

DOWNLOAD

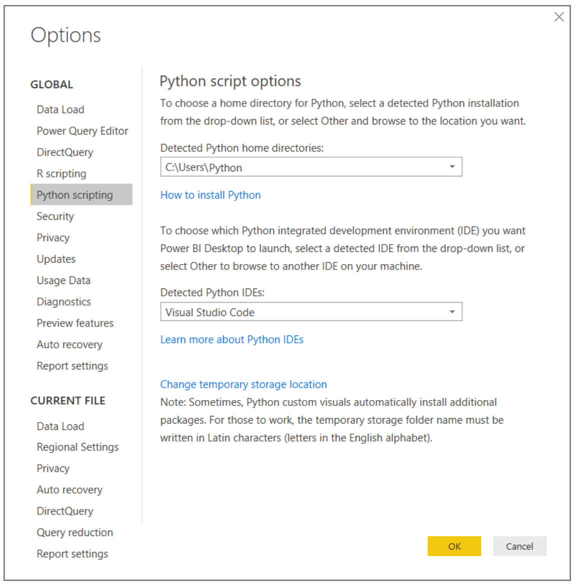
On-premises reporting with Power BI Report Server

Deploy and distribute interactive Power BI reports—and traditional paginated reports—within the boundaries of your organization's firewall.

DOWNLOAD

The first step needed is to download and install Power BI Desktop which is the free version of Power BI. You can use this link which will take you to a downloads page where you should then click on download Power BI desktop. This should then ask if you want to be redirected to the Microsoft store. Say yes and then download Power BI.

If you have a Power BI Pro subscription then by all means feel free to use it but it's not essential. If you don't already have a Power BI account you will need to create one. You will need to use a work or company email address to do this as you can't sign up using email addresses provided by consumer email services or telecommunication providers. This includes outlook.com, hotmail.com, gmail.com, and others. For detailed steps on how to create a Power BI account please see this link.



Python script options

To choose a home directory for Python, select a detected Python installation from the drop-down list, or select Other and browse to the location you want.

Detected Python home directories: C:\Users\Python

How to install Python

To choose which Python integrated development environment (IDE) you want Power BI Desktop to launch, select a detected IDE from the drop-down list, or select Other to browse to another IDE on your machine.

Detected Python IDEs: Visual Studio Code

Learn more about Python IDEs

Change temporary storage location

Note: Sometimes, Python custom visuals automatically install additional packages. For those to work, the temporary storage folder name must be written in Latin characters (letters in the English alphabet).

After this you will need to open up Power BI and enable python scripting. In Power BI Desktop, select File > Options and settings > Options > Python scripting. The Python script options page appears. If necessary, specify your local Python installation path in Detected Python home directories then click OK. If you do not already have python installed then continue with this deployment manual until you reach the section titled 'Installing Python and required python packages.' Complete this section then return back to Power BI to enable python scripting. You now have Power BI set up to be used later on.

Creating an Azure account

The next step needed is to create an Azure account. Azure is used in this project to host a MySQL database, provide access to the Text Analytics API for sentiment analysis, and provide a storage container to upload data to be analysed.

Visit this link and click 'Start for free' to create your own Azure account.

Sign in with your Microsoft or GitHub account or create a free Microsoft account.

On the About you page, select your correct country or region. Enter your first and last name, email address, and phone number. Depending on your country, you might see additional fields, such as a VAT number. Select Next to continue.

On the Identity verification by phone screen, select your country code, and type the number of a telephone to which you have immediate access.

You have the option of text or call back to obtain a verification code. Select the relevant button, type the code in the Verification code box, and select Verify code.

If the verification code is correct, you're asked to enter details of a valid credit card. You will not be charged anything for signing up to Azure as Azure operates on a pay as you go basis. Azure also provides you with £150 of free credit when you sign up for the first 30 days. Enter the card information and select Next.

The last step is to review the agreement and privacy statement then select Sign up.

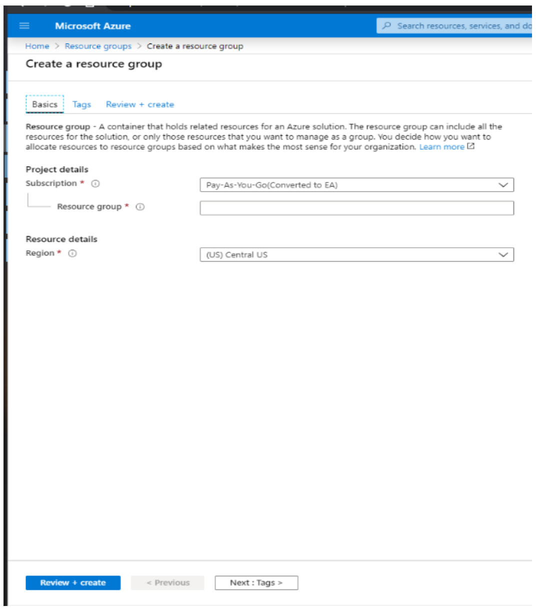
You have now successfully set up a free account on Azure and should be on the Azure portal home page.

<u>Check for and create subscription if necessary.</u>

On the Azure portal home click on ‘Subscriptions’. Check to see if you have a subscription already, it may have been created when you set up your azure account. If the subscription is the trial account that lasts 30 days then that’s fine as well. Before the trail is over Azure will prompt you to upgrade to a pay as you go subscription to continue using any services that use the subscription. If there is no subscription then click on ‘Add’ at the top left and follow the instructions to create an Azure subscription.

<u>Create a Resource Group</u>

Now you have an Azure account the next step is to set up a resource group. A resource group is a collection of resources that share the same lifecycle, permissions, and policies. To create a resource group first click on ‘Resource groups’ on the Azure portal home page. Then click on ‘Create resource group’. Choose a subscription to use and enter the name you want to call your resource group into the field called ‘Resource group’. Then for the field ‘Resource details Region’, select the appropriate region for you from the drop-down list. Then click ‘Review and create’, followed by ‘Create’.



Resource group - A container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want to allocate resources to resource groups based on what makes the most sense for your organization. Learn more

Project details

Subscription * : Pay-As-You-Go(Converted to EA)

Resource group * :

Resource details

Region * : (US) Central US

<u>Create MySQL database on Azure</u>

<u>This link</u> contains a detailed guide on how to set up a MySQL database on Azure. You should complete the section titled ‘Create an Azure Database for MySQL server’. This will walk you through all the steps needed to set up the MySQL server.

When deciding on what pricing tier to choose <u>this link</u> provides information as to what each tier provides in terms of processing power and storage as well as price per month. Choose the plan that you think will be enough for your needs and the amount of data you plan on analysing and storing. For most cases the Basic tier should be sufficient.

You should then complete the section titled ‘Configure a server-level firewall rule’. This creates a firewall rule for specific IP addresses so only registered IP addresses are able to make a connection to the database. Be sure to register all IP addresses that plan on using this database by following the instructions provided in that section.

You should then complete the section titled ‘Connect to the server using the MySQL Workbench GUI tool’. This section walks you through how to connect to the MySQL server using MySQL Workbench. Once you have done this and have connected to the Azure MySQL server, click on File then New query tab. From here you should enter CREATE DATABASE <your database name> and execute the query by clicking on the lightning symbol. You should name the database yourself and will need to remember the name as it’s needed later on. At this stage you should not create a table in your database.

The final thing you need to do is go back to Azure, click on your MySQL server resource that you just created, then in Overview change SSL enforce status to DISABLED.

You now have a MySQL server set up on Azure and have created a database on it. In order to use it though you will need to register your IP address with the database. Only registered IP addresses can make successful connections to the database, this is to add a layer of security. To do this go to your MySQL Server resource and on the left under Settings you'll find Connection security. Click on this then click on '+ client IP'. This will automatically fill in your IP address so all you need to do is add it.

Creating Azure blob storage container

This link contains instructions on how to set up a storage account on Azure but there a few changes that need to be done. In the section 'Create a storage account', on step 7, for the field 'Account kind' don't leave it as 'StorageV2 (general-purpose v2)'. Instead change this to be BlobStorage. Step 8 can be skipped.

Other than the above changes, follow the steps in the link to create your storage account. Once you have done this go to the Overview page in your storage account resource. Click on 'Containers'. You then need to create a new container by clicking on the '+ Container' option. Name the container what you what but be sure to remember it as it will be needed later on.

In regard to how much the storage costs you can see this link. It costs around £0.015 per GB of data uploaded.

You now have a storage account set up on Azure. The container you just made is where you will upload your Excel files containing patient feedback data to.

In the folder called Excel is another folder called MockData. This folder contains a set of excel files containing mock data that you can upload to the storage container you just made. This is so that you can test out the rest of the system and see the visualisations using the mock data. The data in total is under 1000 records so won't take too long to analyse (around 3 minutes). If you don't already have data to upload we recommend you upload these mock data files to your container for use in testing out the system and visualisations as it will give you a better idea of how it all looks.

Creating Azure Text Analytics API resource

This section will provide instructions on how to create a resource for connecting to the Text Analytics API which is used to carry out sentiment analysis on the comments provided by patients.

From your Azure portal home page, go to add resource and search for Text Analytics. Click on Create and enter a name of your choice for the resource. Select a subscription to use and choose a location. For the pricing tier you can choose the free tier. This provides 5000 free transactions per month. If you do choose the free tier and exceed the amount you won't be charged. Instead, your limit will be reached, and you will no longer be able to make successful requests to the API. If you will be analysing more than 5000 new records of data a month then you should choose to use the S-tier which is a pay as you go pricing tier. For more detailed information on pricing for the Text Analytics API please see this link. Choose a resource group to add this resource to and then click on Create.

Installing Python and required python packages.

If not already installed please visit this link to download python 3. You can check if python is installed by opening up command prompt and typing 'python --version'. If it gives you the version number of python then it is already installed, otherwise you will need to visit the previous link and install it yourself. You will also need to have PiP installed to download the python packages required. This should come by default with the python installation but can be checked similarly by running 'pip --version' on the command prompt. We have supplied a requirements.txt file with our code meaning that all you have to do to install the packages needed is to run 'pip install -r requirements.txt'. Before doing this though it may be best to set up a virtual environment so that the packages installed do not interfere with other versions of the same packages installed already.

To do this open a command prompt in the directory where the code is. In the command prompt type:

```
pip install virtualenv
```

This installs the module that creates virtual environments. Once this is installed type:

```
python -m venv venv
```

This creates a virtual environment called venv. To activate the virtual environment type:

```
.\venv\Scripts\activate
```

This activates your virtual environment. If you install packages now they will be localised to the virtual environment and won't interfere with other projects outside of it. To install the required modules for the code run the following command:

```
pip install -r requirements.txt
```

This installs all the modules listed in the requirements.txt file which should be in the same directory as the code and the tests.

Another thing you need to do is install the modules required for Power BI to use python and make requests to the API. These modules are matplotlib, pandas and requests. For this you should install the modules outside of your virtual environment so that Power BI has access to the modules when needed. If you go to the folder called PowerBI you will find inside of it is another requirements.txt file. Open a command prompt and navigate to the PowerBI folder. Then once again run:

```
pip install -r requirements.txt
```

making sure to run it outside of a virtual environment. You should now have installed the required modules for Power BI to make requests to the API and store the data it receives.

Adding required details to config.json file

In the folder containing the code and tests there is a file called config.json. This file is used to store all of your Azure details such as API keys, database credentials etc. You will need to fill in this file with your Azure details. When filling in your details make sure they are still surrounded by double quotes.

storage_account_name is the name given to you Azure storage account resource.

storage_account_key is a key that needs to be provided to access and use the Azure storage services. Go to the storage resource and you can find the key under Access keys in settings, down the left side of the page.

storage_container_name is the name you gave to the container that you will upload your excel files to.

text_analytics_key is a key that needs to be provided in order to use the Text Analytics API. Go to the Text Analytics resource and down the left, under Resource Management is a section called Keys and Endpoints. Use key1 as your key.

text_analytics_endpoint can be found in the same place as the text analytics key.

database_username and **database_host** can be found by opening MySQL Workbench, right clicking the connection you made earlier and selecting edit connection. Once you have these details you can then click close and exit Workbench.

Alternatively, you can go to the MySQL Server resource you made earlier in Azure portal and in the Overview section you can find the value for host under 'Server name' and the value for user under 'Server admin login name'.

database_password is the password you used when you created the resource.

database_name is the name you used earlier on when creating the database on MySQL Workbench.

API_username and API_password are 2 things you need to choose that will need to be supplied every time a request is made to the API to get data. This is to add a layer of security to prevent unauthorised access. This is in addition to the security measure where in order to access the database, the IP of the source of the request must be registered with the Azure MySQL server resource under Connection Security.

Request.py

This file doesn't make up part of the API code. It is provided to show examples of how to make requests to the API using python and how to make requests to the various endpoints provided.

In the files are multiple places with the default values your_chosen_username and your_chosen_password. Go through and replace them all with the username and password you chose for the API (the values you used for API_username and API_password). This file contains 4 examples of requests so there should be 4 places where you need to replace the default values with your values for the username and password.

Running tests and checking everything is working

The project folder includes a set of test files that can be run to check that everything is working as expected. These tests ensure that connections can be made to all the Azure resources, the keys, passwords and endpoints were entered correctly, and that authentication is set up for the API.

To run the tests, open up a command prompt in the same directory as where the code is. You should then type the following command into the terminal:

```
python -m unittest discover
```

This will then run all the unit tests and they should hopefully all pass.

Common reasons for test failures

Some common reasons that tests may fail include:

- Incorrect details for API key or API endpoint, double check that the API keys and endpoints you are using in the code are the exact same as the ones shown on Azure. Also check your details for the database connection is correct and everything is spelt correctly.
- Not surrounding your Azure details such as keys and endpoints in quotes.
- Your IP address is not registered on Azure. Go to your MySQL Server resource and on the left under Settings you'll find Connection security. Click on this then click on '+ client IP'. This will automatically fill in your IP address so all you need to do is add it. Allow it to update then try to run the tests again.
- SSL enforce status set to ENABLED. Go to your MySQL server resource and in overview check to see what your SSL enforce status is. It should be DISABLED. If it's not, click on it to change it to DISABLED.
- Mismatching passwords and usernames. Make sure you have spelt your username and passwords correctly in all places where they are used.
- Missing libraries. Make sure you have run pip install -r requirements.txt in the correct directory to install all the required packages and make sure you're in the same virtual environment you installed the packages in.

- Running unittest command in wrong directory, make sure it's run in the same directory as the project code files.
- You may be analysing large amounts of data causing the database connection that's open to time out. To increase the wait time of the database connection, go to the database resource you have on Azure. Down the left, under settings, select 'Server parameter'. At the bottom change the value for 'wait_timeout' to be a large enough value for the amount of data you are analysing.

<u>Adding python request script to Power BI with API running on localhost</u>

Assuming the tests ran successfully, you have now finished setting up the code. The last step in the setup is to configure Power BI to make a request to the API in order to refresh its data. The first step is to get the API running on localhost so that when Power BI makes a request to it, it is able to provide a response.

To run the API on localhost first open up a command prompt in the same directory where the project code is stored. Then enter your virtual environment by using .\venv\Scripts\activate.

Then type the following:

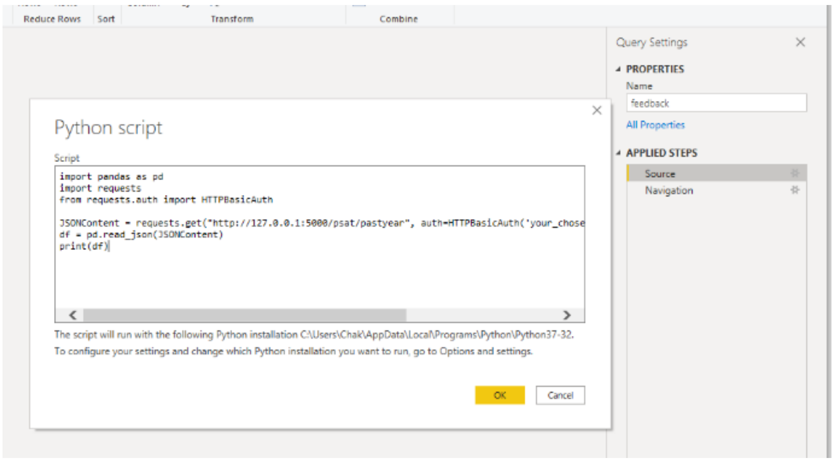
set FLASK_APP=application.py

press enter, then type:

flask run

Press enter and this should start up the API on your localhost on port 5000 and it is now ready to receive requests from Power BI and provide responses containing the data to be visualised.

You will then need to open up the Power BI file called PatientServiceAnalysisTool.pbix in the folder called PowerBI. On the left find the Fields tab and right click on the table called 'feedback' then select 'edit query'.



```
import pandas as pd
import requests
from requests.auth import HTTPBasicAuth

JSONContent = requests.get("http://127.0.0.1:5000/psat/pastyear", auth=HTTPBasicAuth('your_chose
df = pd.read_json(JSONContent)
print(df)
```

The script will run with the following Python installation C:\Users\Chak\AppData\Local\Programs\Python\Python37-32.
To configure your settings and change which Python installation you want to run, go to Options and settings.

In the dialogue box that pops up, double click on 'Source' and a box called 'Python script' should pop up. The box will have some python code in it already, you should get rid of this code as you will be replacing it. In this box you should add the code shown below. This code will be run whenever you refresh data on Power BI, manually or automaitcally. The script makes a request to the API on localhost for the most recent years worth of data. If later on you publicly host the API somewhere else then be sure to change the host from local host to whatever the hostname of the platform you hosted the API on is. You can also use different endpoints to get different amounts of data to analyse. Endpoints available are explained further in the User Manual.

Code to copy into 'Python Script' box:

Please copy everything between (but not including) the dashed lines into the 'Python script' box. Also be sure to replace your_chosen_username and your_chosen_password with your username and password that you chose for your API (the values you used for API_username and API_password).

```
import pandas as pd

import requests

from requests.auth import HTTPBasicAuth


JSONContent = requests.get("http://127.0.0.1:5000/psat/pastyear",
auth=HTTPBasicAuth('your_chosen_username', 'your_chosen_password')).json()

feedback = pd.read_json(JSONContent)
```

You should then click on OK and then close the box using the red X. It will ask you if you want to apply the changes to your query. You should select yes. This will then run the script to make a request to the API running on localhost. The API will return with a response containing all the analysed data which Power BI will then use to populate a set of tables. These tables will then act as data sources for the visualisations.

From now on when you add data to your Azure storage account and want to update Power BI to see the visualisations, you must run the API on localhost if you are not publicly hosting it elsewhere, and then click Refresh at the top, under the home tab. This will re-run the script and process the data and visualise it. More information on making requests and the time taken for requests is included in the User Manual.

You have now set up all the resources necessary to use the system. For details on how the system works and how to use it please refer to the User Manual.

Adding python request script to Power BI with API running on Azure as WebApp

If you want to host the API so that more than one device can make a request to it at a time, you won't have to change too much. By hosting the API on Azure, you won't need to run the API on localhost every time you want to refresh the data. You already set up all the resources on Azure so there is no need to repeat that. There are however a few restrictions when using Azure to host the API.

Azure web apps time out requests after 4 minutes. This means when making a request to the API to analyse data, if it does not analyse the data, write it to the database and return a response within 4 minutes, then the response is timed out and a timeout error is returned. For small private clinics or GP practices that receive under 1000 patient feedback forms a month, this is perfectly fine and that amount of data can be analysed within the 4-minute time limit. For larger hospitals and organisations that receive large amounts of patient feedback each month, hosting on Azure likely isn't a viable option. Azure also doesn't let you run the analysis as a background task or allow you to extend the timeout duration so there isn't a way round it. For larger organisations looking to host the API somewhere we recommend choosing a hosting service that allows you to either run the analysis in the background or lets you choose the timeout limit.

For smaller organisations and clinics wanting to use the system, you may initially have a large amount of data you want to upload and analyse. For the initial batch upload and analyses of data we recommend using localhost as to not have issues with timeouts using Azure. From then onwards, when you want to analyse the new months' worth of data that's coming in you can use Azure.

Deploying API on Azure.

The first step you need to do is install the Azure CLI from this link. Once that's downloaded run the installer to set it up. You should then create a new folder which will contain the code you will deploy to Azure. To this folder copy the following files from the Code folder from the main project folder:

- application.py
- AzureBlobStorage.py
- Database.py
- ProcessData.py
- TextAnalyticsAPI.py
- requirements.txt
- config.json

Open a command prompt and navigate to the folder you just created and copied the files into. In this folder set up a virtual environment and activate it by doing:

python -m venv venv

followed by:

.\venv\Scripts\activate.

Then install the requirements by doing:

pip install -r requirements.txt

The next thing you need to do is type 'az login' and press enter. This will open up a page in your browser and ask you to provide your azure credentials. Once you have done this you will be logged in and able to use the Azure CLI. You should then type into the command prompt the following:

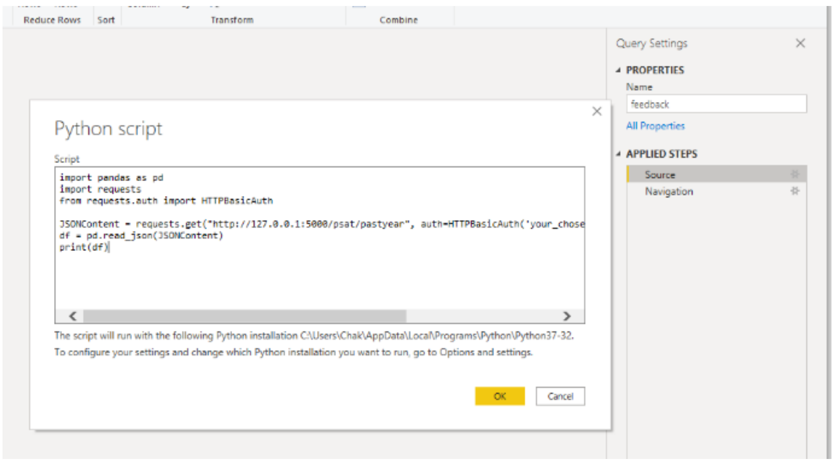
az webapp up --sku F1 -n <app-name> -l <location-name> -g <resource-group-name>

replace <app-name> with whatever you want to call your web app but it must be unique. Replace <location-name> with an Azure region such as centralus, eastasia, westeurope, koreasouth, uksouth, centralindia, and so on. Replace <resource-group-name> with the name of the resource group you made earlier.

Hit enter and it will take a few minutes to run. After it has finished running you will have the API deployed on Azure as a web app. If you encountered any issues please have a look at this tutorial from Azure showing how to deploy a web app.

Code to copy into 'Python Script' box:

You will now need to open up the Power BI file called PatientServiceAnalysisTool.pbix in the folder called Power BI. On the left find the Fields tab and right click on the table called 'feedback' then select 'edit query'.



```
import pandas as pd
import requests
from requests.auth import HTTPBasicAuth

JSONContent = requests.get("http://127.0.0.1:5000/psat/pastyear", auth=HTTPBasicAuth('your_chose
df = pd.read_json(JSONContent)
print(df)
```

The script will run with the following Python installation C:\Users\Chak\AppData\Local\Programs\Python\Python37-32.
To configure your settings and change which Python installation you want to run, go to Options and settings.

In the dialogue box that pops up, double click on 'Source' and a box called 'Python script' should pop up. The box will have some python code in it already, you should get rid of this code as you will be replacing it. In this box you should add the code shown below. This code will be run whenever you refresh data on Power BI, manually or automaitcally. The script makes a request to the API being hosted on Azure for the required data to populate the visualisations. You can also use different endpoints to get different amounts of data to analyse. Endpoints avaialable are explained further in the User Manual.

Please copy everything between (but not including) the dashed lines into the 'Python script' box. Also be sure to replace your_chosen_username and your_chosen_password with your username and password that you chose for your API (the values you used for API_username and API_password). You also need to replace <your-app-name> with the name you gave your web app when running the Azure CLI command.

----

import pandas as pd

import requests

from requests.auth import HTTPBasicAuth

JSONContent = requests.get("https://<your-app-name>.azurewebsites.net/psat/pastyear/", auth=HTTPBasicAuth('your_chosen_username', 'your_chosen_password')).json()

feedback = pd.read_json(JSONContent)

You should then click on OK and then close the box using the red X. It will ask you if you want to apply the changes to your query. You should select yes. This will then run the script to make a request to the API running on Azure. The API will return with a response containing all the analysed data which Power BI will then use to populate a set of tables. These tables will then act as data sources for the visualisations.

From now on when you add data to your Azure storage account and want to update Power BI to see the visualisations, you just need to click Refresh at the top, under the home tab. This will re-run the script and process the data and visualise it. More information on making requests and the time taken for requests is included in the User Manual.

You have now set up all the resources necessary to use the system. For details on how the system works and how to use it please refer to the User Manual.