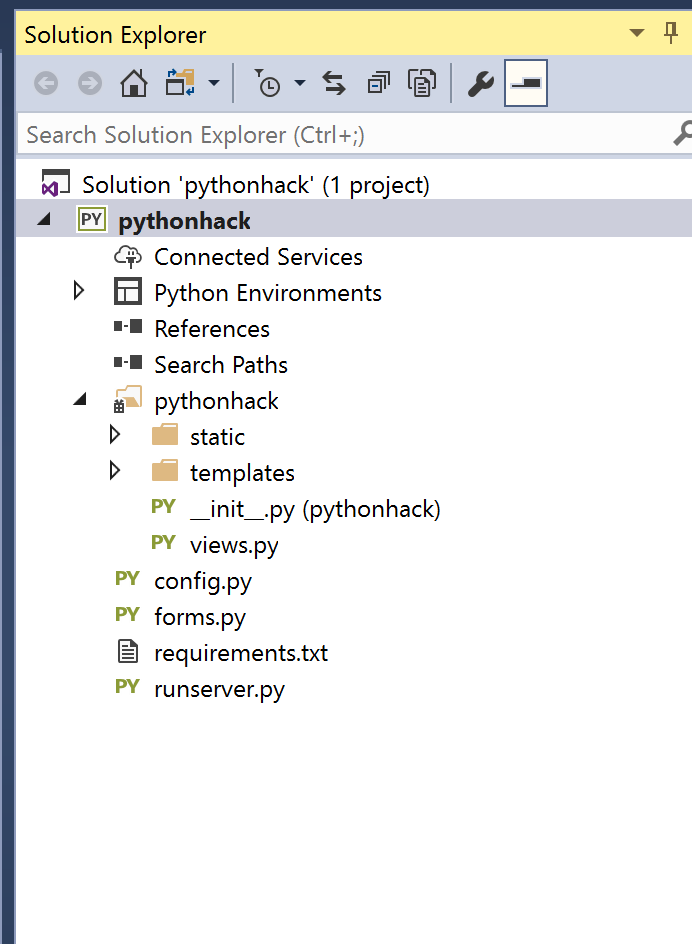
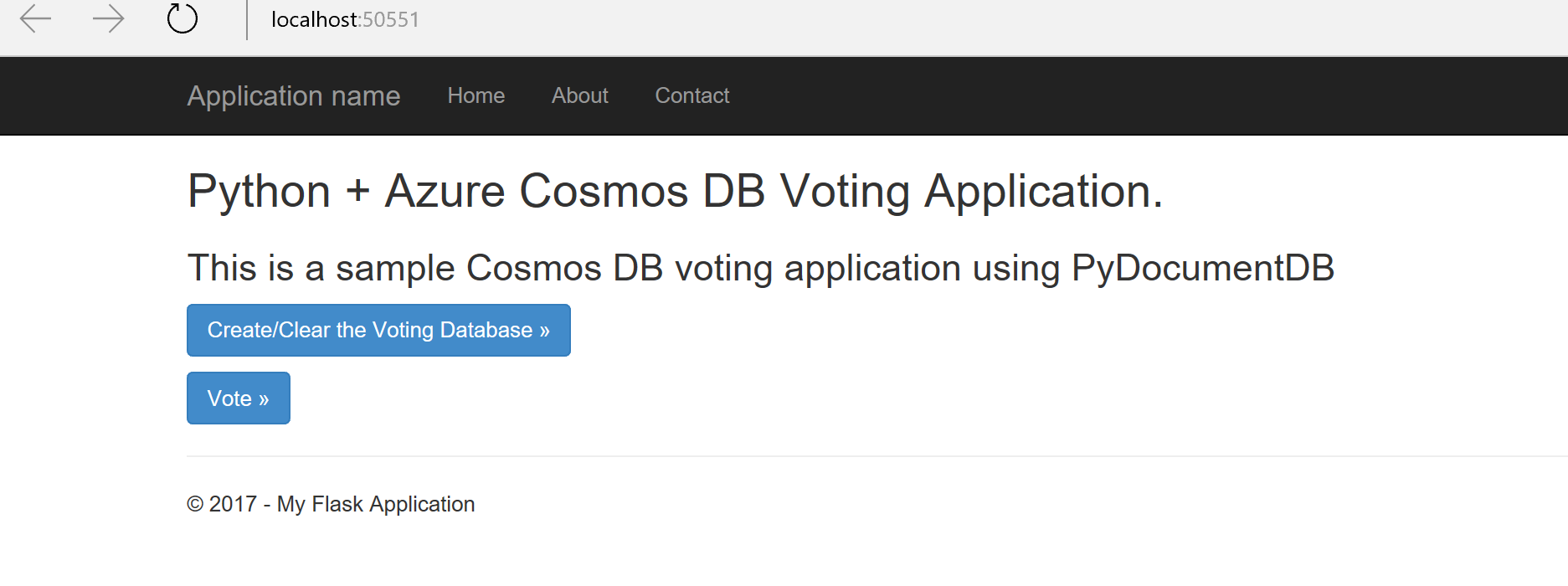
# Hosting Python Flask Web Application using Azure Linux App Service Plan

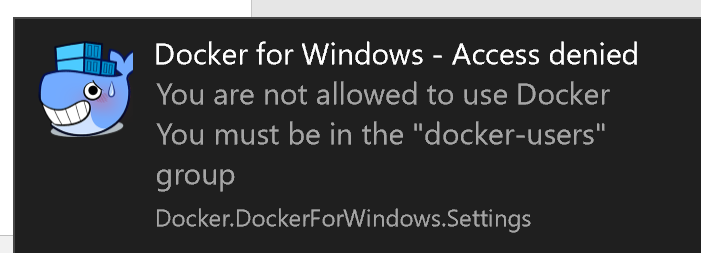
This hands on lab shows ow to create a Docker Python web app in Azure, you will connect the web app to Azure cosmos DB, when you are done you will have a Python Flask application running within a Docker container on Azure App Service Web Apps.

# Build Python Web App with Flask

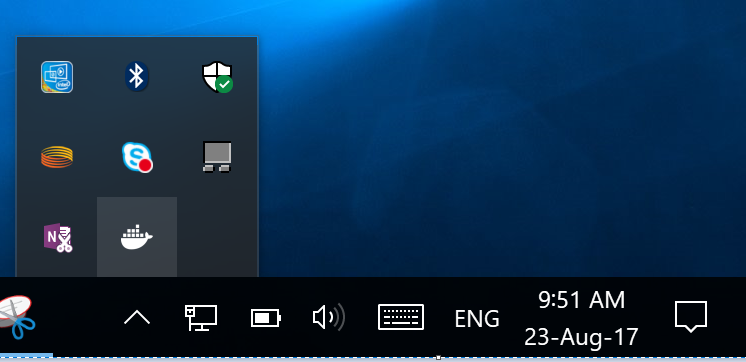
1. Follow the Microsoft Article in >>[LINK](https://docs.microsoft.com/en-us/azure/cosmos-db/documentdb-python-application)<< to build a Python Flask web application that uses Azure Cosmos DB to store data
2. Follow the HOL along in the above given link until step -4 (Step 4: Run your web application locally), do not continue to step -5, we will not be publishing directly to Azure in this HOL, instead we will be making Docker image and publishing to Azure APP Service as a Docker container
3. Your final solution structure should be looking like this
4. 
5. When you build and run the application, you should have the Python web application running locally 

## Setup Docker

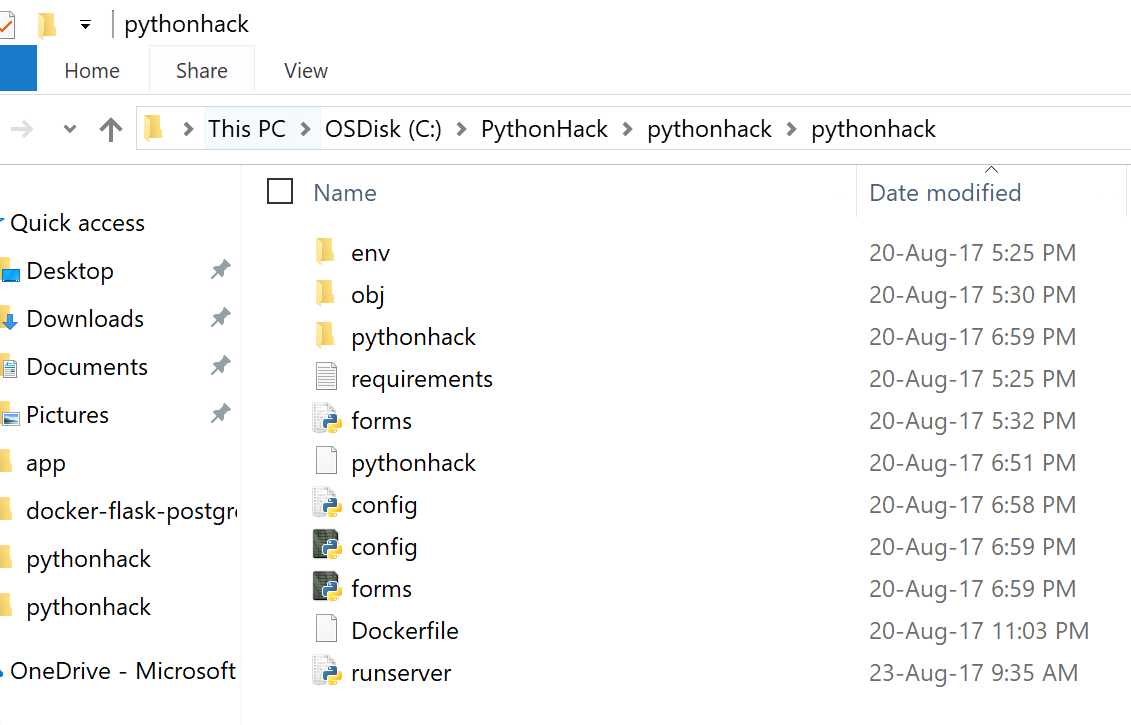
1. Download and install Docker CE for windows from - <https://www.docker.com/community-edition>
2. Try to launch “Docker for Windows” from your desktop, if you get an error as shown below, go to step 3, otherwise skip step 3



1. Go to control panel and access “Edit local users and group” and find a group called “docker-users”
2. After successful installation, you should have Docker up and running, verify the status by checking in status bar



## Running the application from a Docker Container

1. Make a note of the local folder where the Python web application’s folders are located
2. Open a notepad file and past the following content and save the file as a “Dockerfile” without any file extension, update the WORKDIR if needed (based on your naming convention). Save this file in the root folder of the python web application

*# Use an official Python runtime as a parent image*

*FROM python:2.7-slim*

*# Set the working directory to /app*

*WORKDIR /pythonhack*

*# Copy the current directory contents into the container at /app*

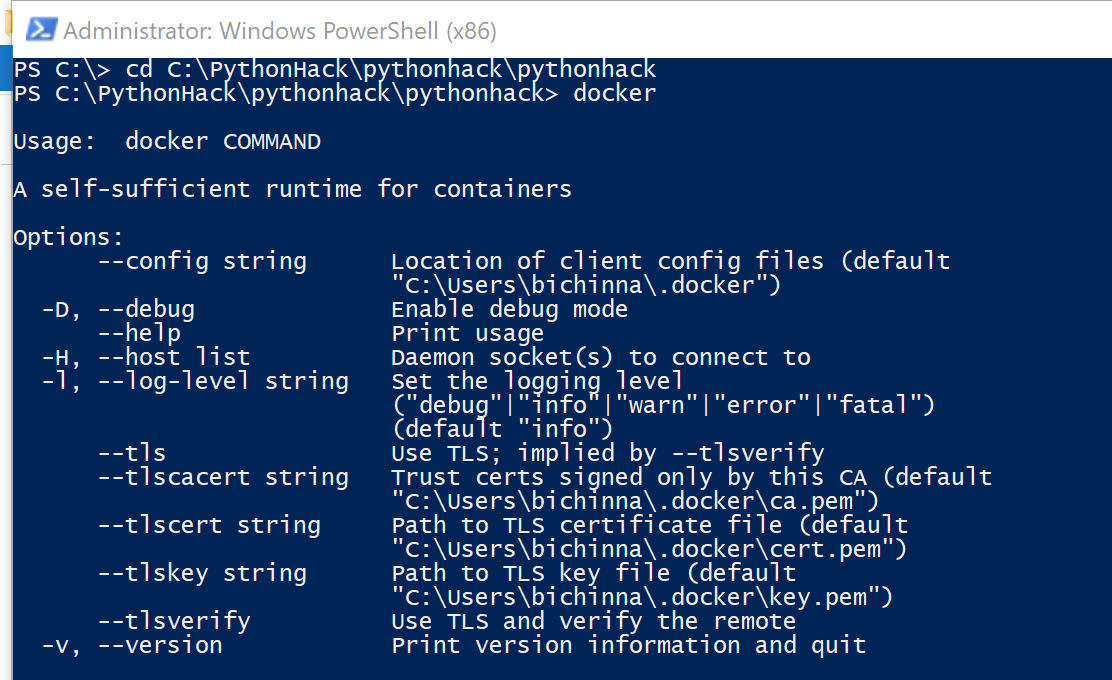
*ADD . /pythonhack*

*# Install any needed packages specified in requirements.txt*

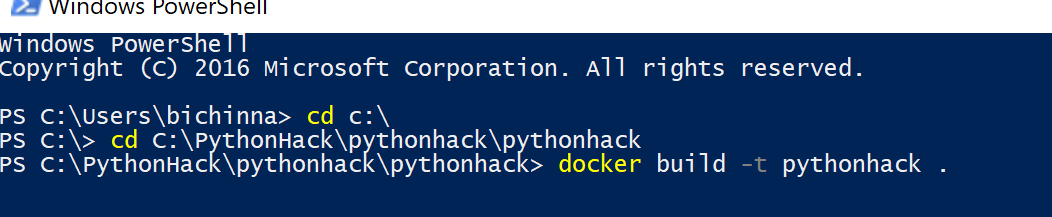
*RUN pip install -r requirements.txt*

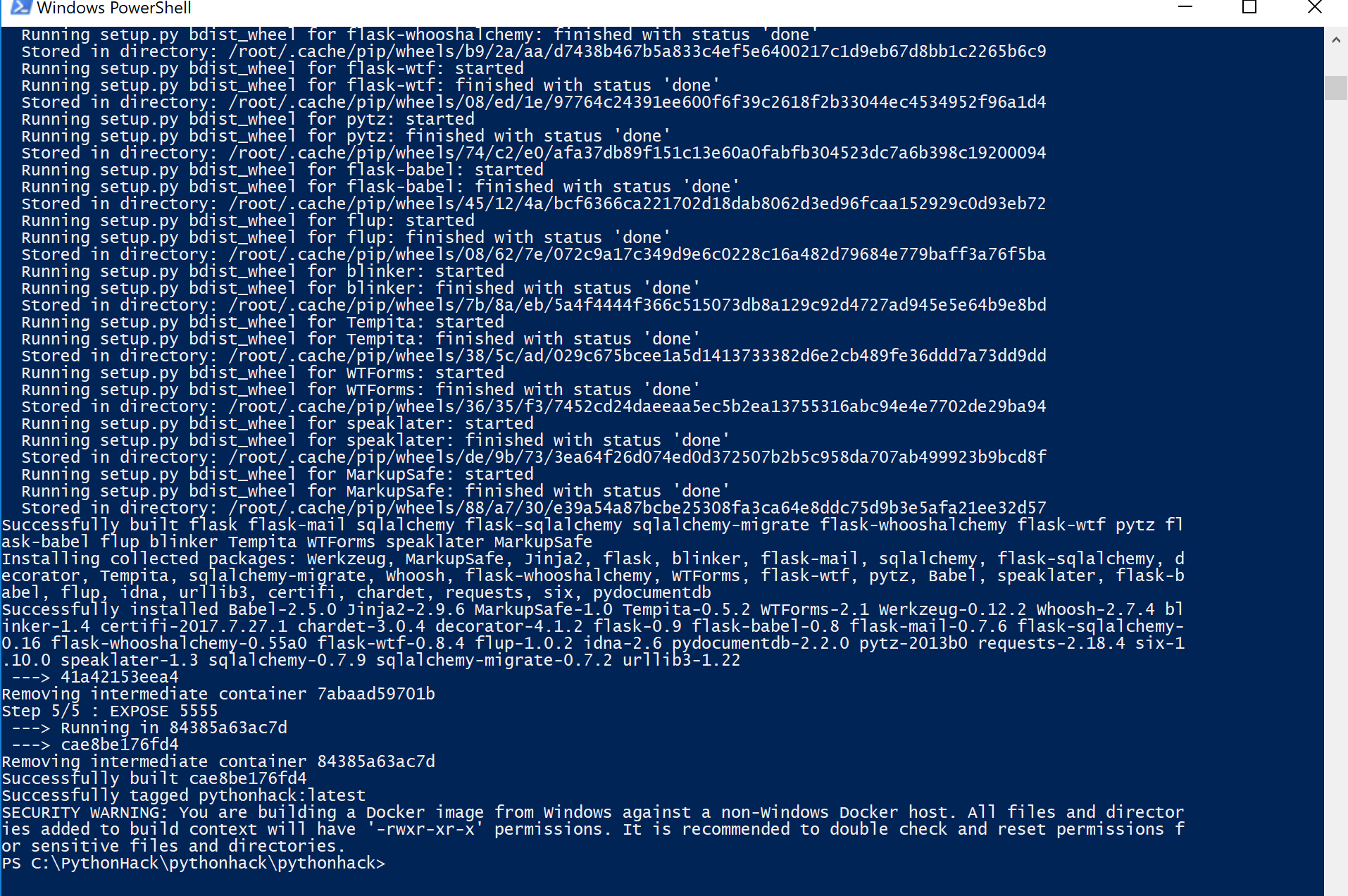
*# Make port 5555 available to the world outside this container*

*EXPOSE 5555*

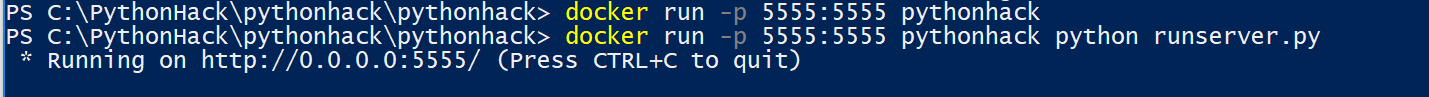
1. Open windows PowerShell and run as “Administrator”, and navigate to your working folder
2. Type command “docker” and you should see information about Docker displayed as shown below. This indicates we are good to go with further steps
3. Use the following Docker build command to create a Docker image, we are using -t to have a friendly name for the image (note the period at the end of the command)

***docker build -t pythonhack .***

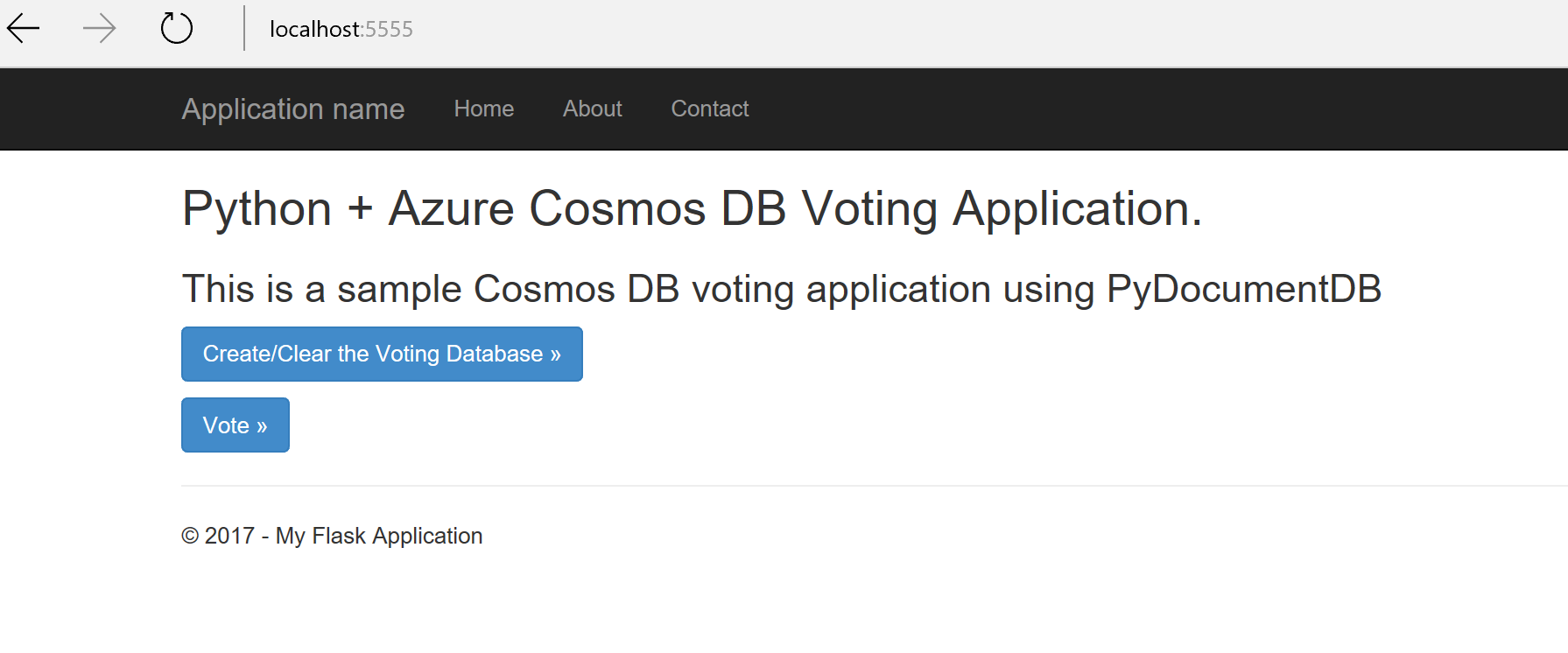


1. The command will take some time to build the image, when it is done, you will have the following output in the PowerShell window 
2. To double check everything in place, run “docker images” command, this should produce an output that should have the image we just created
3. Now we are ready to run the docker image which is have our Python web application. In order to run the Python Web application, we also need to start the Flask server, we already have a Python file called “runserver.py” to start the server. Go ahead and run the following command to start the flask server

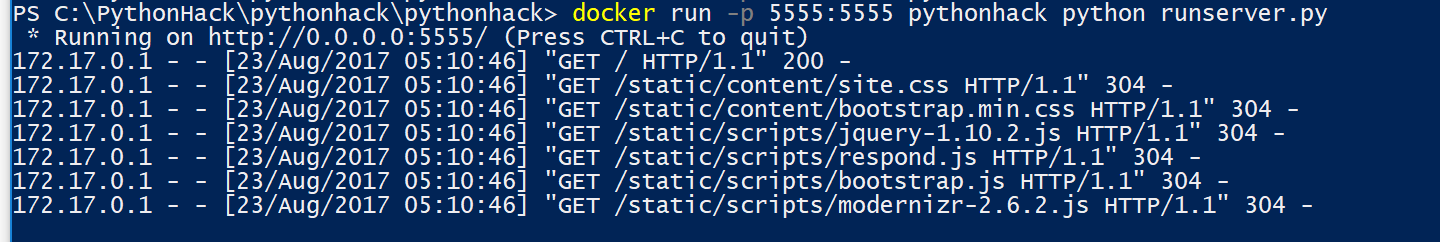
***docker run -p 5555:5555 pythonhack python runserver.py***



1. Now go to your browser and type <http://localhost:5555/> and you should be able to access the web site running in Docker container. Port 0.0.0.0 in docker image is exposed (mapped) to localhost



1. You should now also see the following output in the PowerShell window



1. [Optional]Alternatively, you can update the “Dockerfile” to run the runserver.py file automatically when Docker runs, append the following two lines to “Dockerfile” and re-create the image

ENTRYPOINT [ "python" ]  
CMD ["runserver.py"]

1. Now run the following command to run python flask server automatically when Docker runs the container

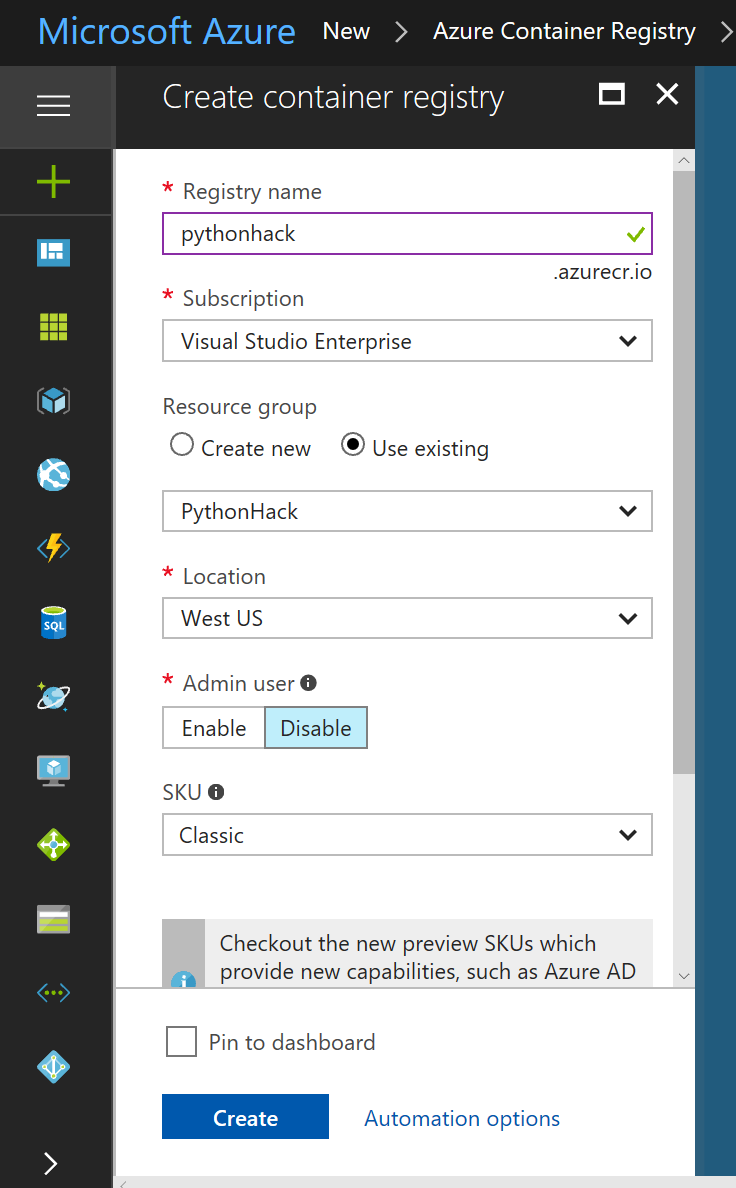


1. To stop Flask server at any time, type Ctrl+C in the PowerShell terminal

## Upload the Docker Container to a container registry

In this step, we will upload the Docker container to a container registry, you will use Azure container registry, but you could also use other popular ones such as Docker Hub

1. [Azure portal](http://portal.azure.com/) and create “Azure Container Registry” – Chose your Azure resource group and also “enable admin user” in the options

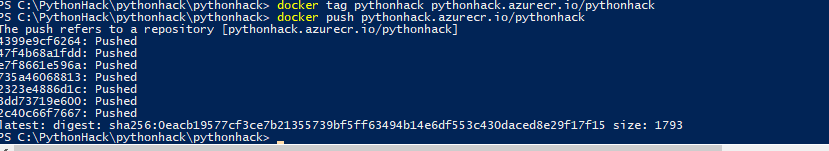


1. Go to the created container registry ->Access keys and make note of the user name and the password (first password)

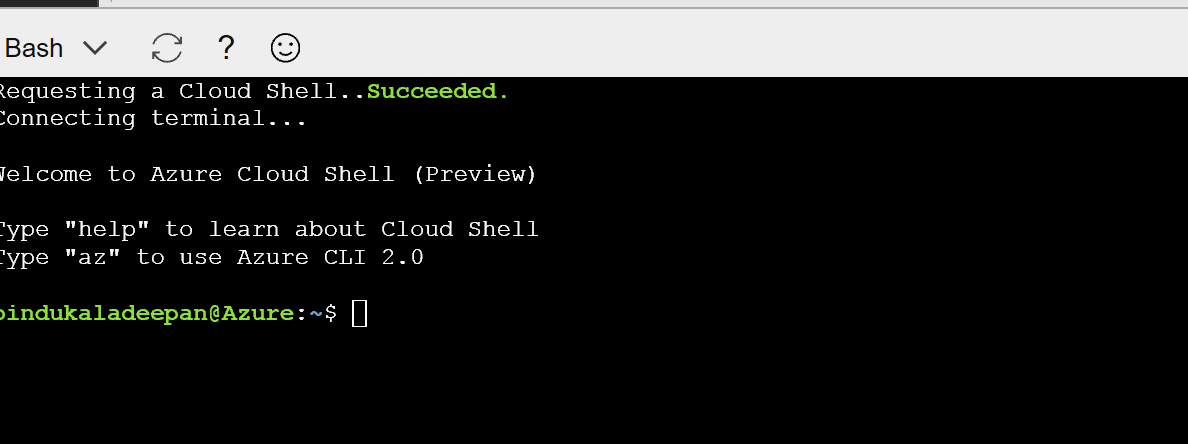


1. Upload the docker container to Azure container registry

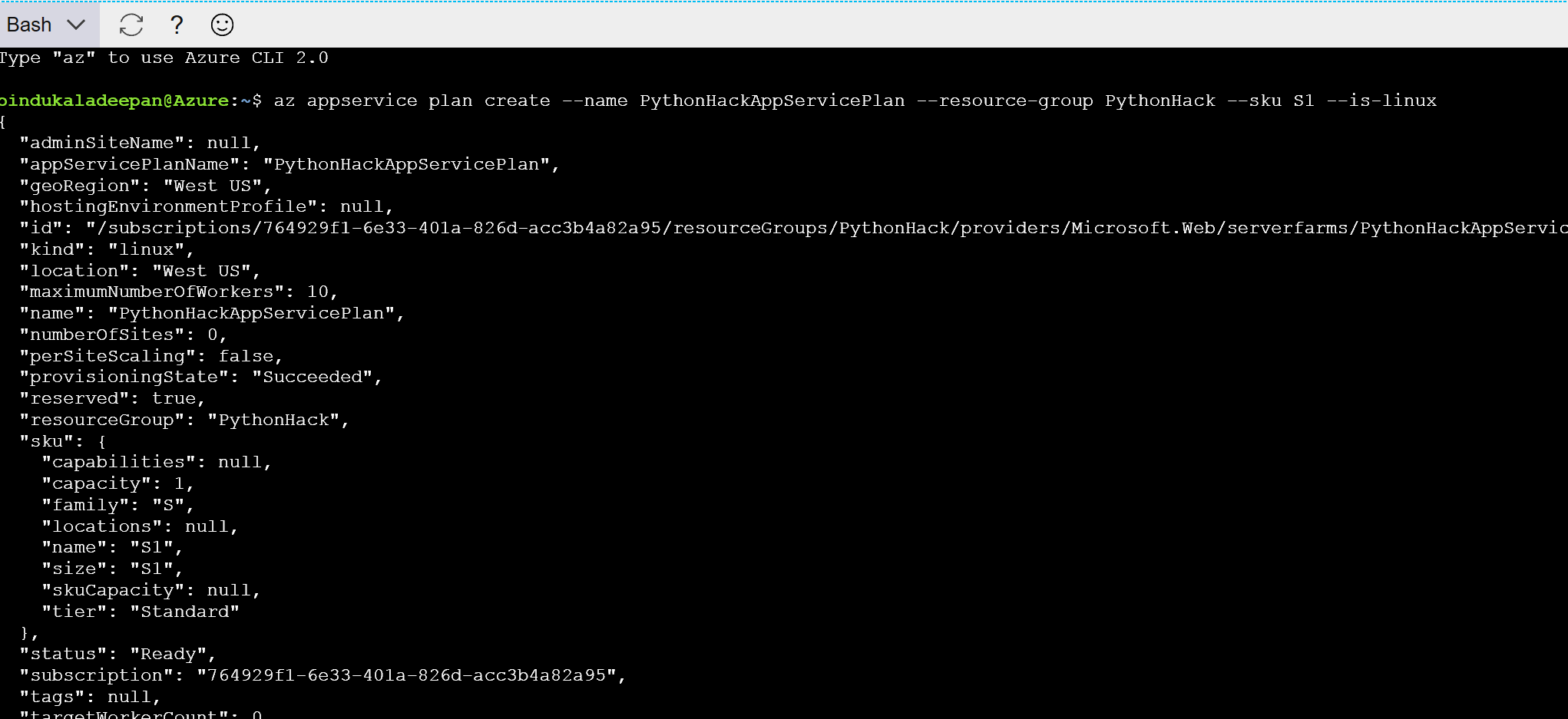
*docker login <registry\_name>.azurecr.io -u <registry\_name> -p "<registry\_password>"  
docker tag pythonhack <registry\_name>.azurecr.io/pythonhack  
docker push <registry\_name>.azurecr.io/pythonhack*



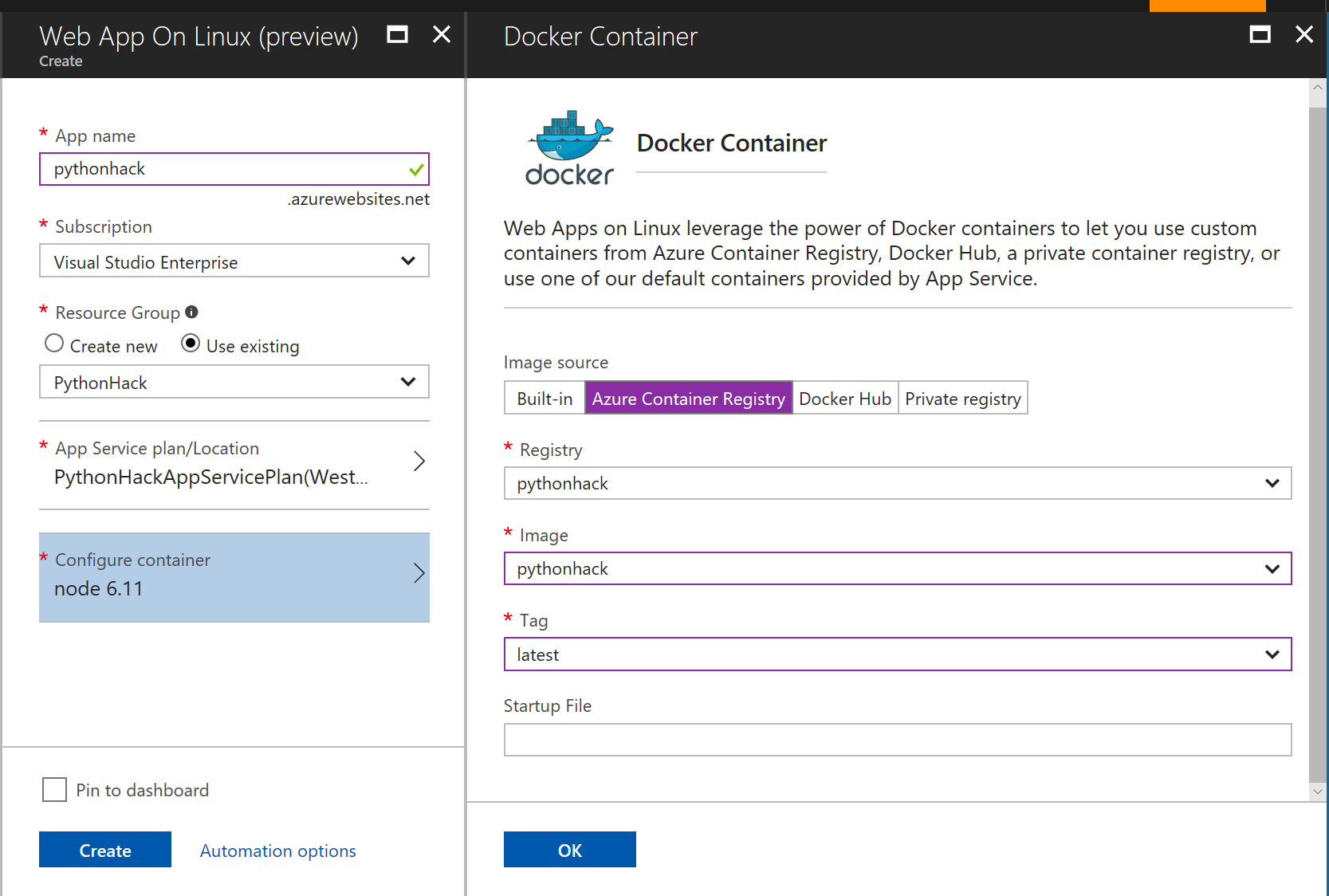
Deploy the Docker Python Flask application to Azure

1. Go to Azure portal and access Azure cloud Shell 
2. Run the following command, The following example creates a Linux-based App Service plan using the S1 pricing tier (You coud also do this manually in the portal)

az appservice plan create --name <<AppServicePlanName>> --resource-group <<ResourceGroup>> --sku S1 --is-linux

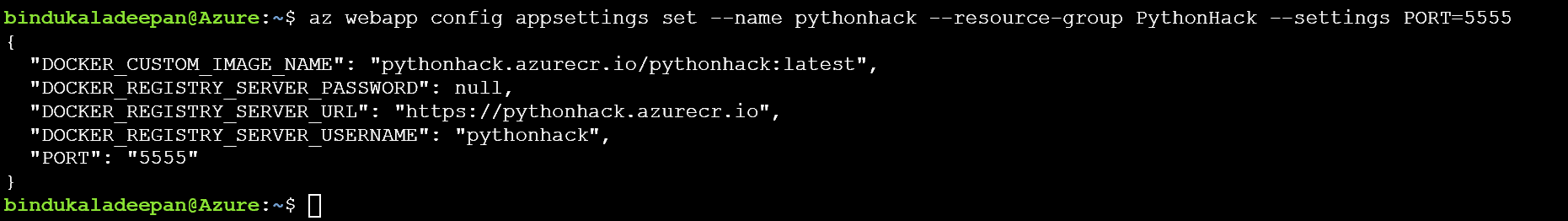
When the App Service plan is created, the Azure CLI shows information similar to the following screen 

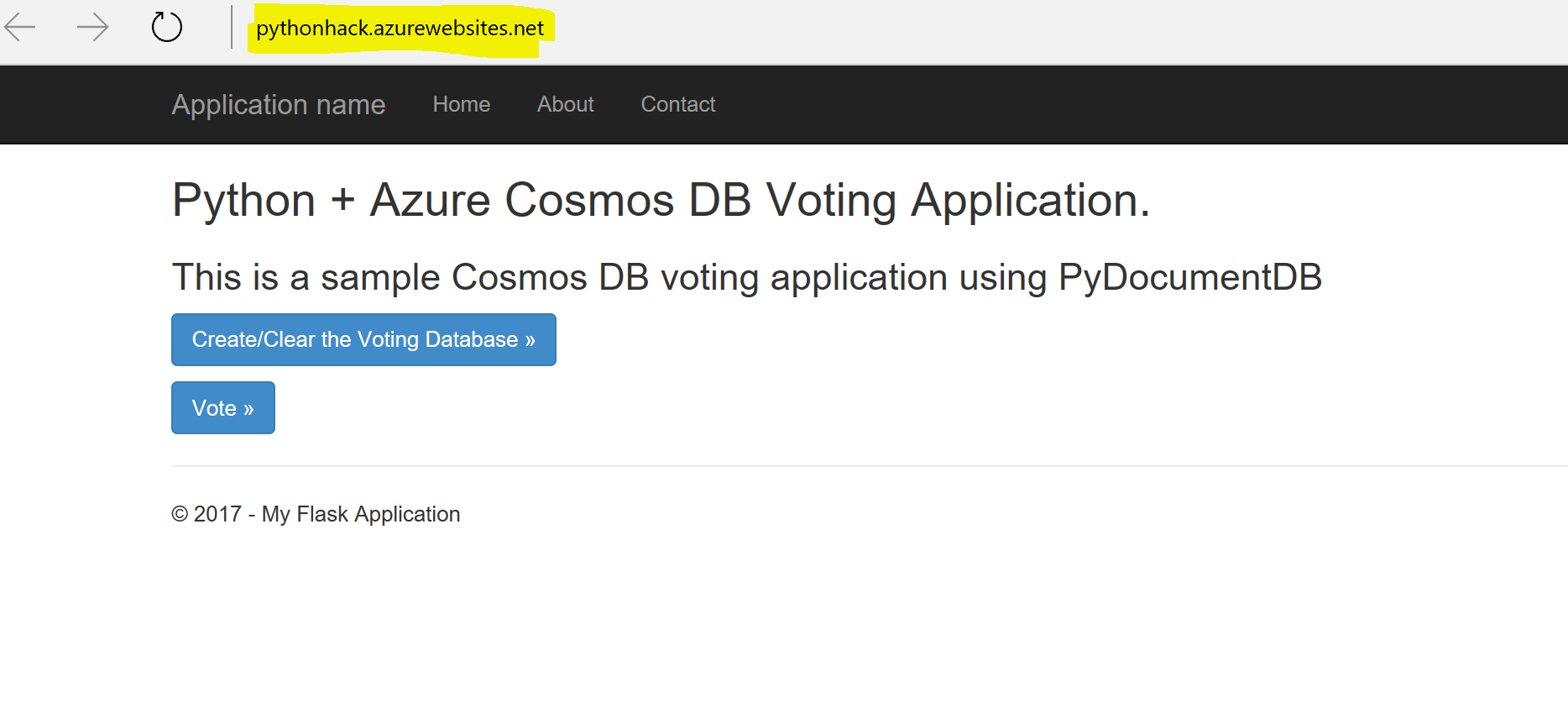
1. Create Web APP using “Web App on Linux (preview) as shown below, use the Azure container registry that we have used in the earlier step



1. Run the following command in Azure Cloud Shell , Change the port number, web app name and resource group name as per your configuration

*az webapp config appsettings set --name pythonhack --resource-group PythonHack --settings PORT=55555*



1. Re-start the WebAPP and try to access the URL, you should now able to access Python Flask Web Application running in Azure App service Linux Plan 

### References

This exercise borrows and adapts content from the following sources:

1. [The official Docker documentation](https://docs.docker.com/)
2. [Building a Docker Python and PostgreSQL web app in Azure](https://docs.microsoft.com/en-us/azure/app-service-web/app-service-web-tutorial-docker-python-postgresql-app)
3. [Building a Python Flask web application using Azure Cosmos DB](https://docs.microsoft.com/en-us/azure/cosmos-db/documentdb-python-application)