The mental health chatbot was developed using RASA framework and acts as the backend REST API for the application. In order to deploy this API, Docker and Azure Cloud were used.

To Dockerize the application, a Dockerfile and a .dockerignore was built inside the folder where the application files are located. These files should be able to create a customized Docker image that runs the developed chatbot.

Shown below is the Dockerfile which does the following:

* Gets the official open source rasa image from docker hub as a base image (<https://hub.docker.com/r/rasa/rasa>)
  + A base image acts like a template which can be used as a starting point in building custom images
* Copies the chatbot files from local machine to the Docker image
* Sets the user which runs the application (any user would do as long as it is not as root)
* Runs the RASA application using native rasa command:
  + rasa run –enable-api –port 8080
  + this command runs RASA as an API and exposes port 8080 which we can use to access it.

FROM rasa/rasa:latest

# set workdir and copy data files from disk

WORKDIR /app

ENV HOME=/app

COPY . .

# set the user to run, don't run as root

USER 1001

# set entrypoint for interactive shells

ENTRYPOINT ["rasa"]

# command to run when container is called to run

CMD ["run", "--enable-api", "--port", "8080"]

As for the .dockerignore, it only consists of paths and files that should be excluded while copying to the docker image.

|  |
| --- |
| tests/\*  actions/\*  \*\*/\*.md  venv |

Once both files are set up, it is now time to build the image. Make sure that Docker is installed and running in your local machine. Then, inside the application folder, open the Terminal and run:

* docker build -t ncmdd/omdena-joybot **.**

The output should be similar to the one shown below.

Text

Description automatically generated with medium confidence

This should build the image successfully. Next is to push the image to an existing Dockerhub repository in order for the custom image to be publicly available. To do this, use the following commands:

* docker login – required to login to a docker hub before pushing the custom image
* docker push ncmdd/omdena-joybot – pushes the image to Dockerhub

Graphical user interface, application

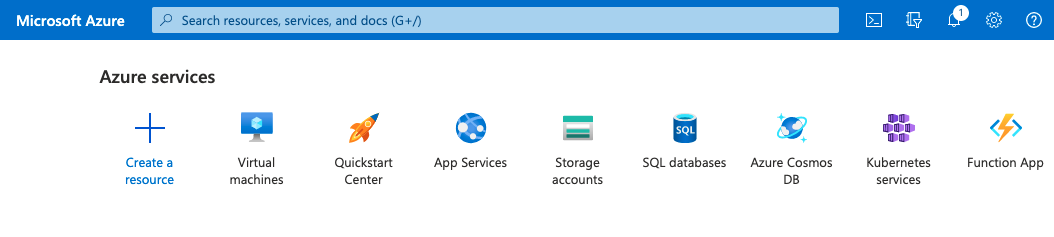
Description automatically generated

When the custom image is now publicly available, the application can now be deployed to Azure. For the cloud deployment, a virtual machine is launched in Azure Cloud. The specifications of the server is as follows:

* Operating System/ Image: Ubuntu Server 20.04 LTS – Gen2
* Instance Size: Standard\_D2s\_v3 (2 vCPUs, 8 GiB memory)
* OS disk type: Premium SSD
* Network: Allows port 22 and 8080

The server was launched with the following steps:

* Go to [Azure Portal](http://portal.azure.com/) then select Virtual Machines



* In the Virtual Machine Console, select Create > Virtual Machine

Graphical user interface, text, application

Description automatically generated

* On the Basics tab, fill in the following details and leave the others on default
  + Resource Group – create new then input the preferred name of the new group
  + Virtual Machine Name – any name would do
  + Region – any region would do depending on the requirements. It is recommended to deploy nearest to your target users
  + Image: Ubuntu Server 20.04 LTS – Gen2
  + Size: Standard\_D2s\_v3 (2 vCPUs, 8 GiB memory)
  + Key pair name – any name would do. This will be the filename for the Keypair to be used in accessing the Virtual Machine
  + Public inbound ports: Allow selected ports
  + Select inbound ports: SSH (22)
* Leave the other tabs on default settings then proceed to Review + create. Select **Create** to launch the virtual machine. It would take around a few minutes for it to fully be launched
* A window will show up asking to download the keypair. This must be downloaded ot be used in the next steps.
* After launch the machine successfully, go to the instance then select **Networking**.

Graphical user interface, text, application

Description automatically generated

* Select **Add inbound port rule** then allow port 8080

Graphical user interface, text, application, email

Description automatically generated

Once the virtual machine is running and has been configured, it needs to be configured from the inside to run the RASA application.

* SSH into the instance using the keypair downloaded upon launch.
* Once inside the instance, install docker using the commands:

sudo apt-get install \  
 apt-transport-https \  
 ca-certificates \  
 curl \  
 gnupg-agent \  
 software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key

add –

sudo apt-key fingerprint 0EBFCD88

sudo add-apt-repository \  
 "deb [arch=amd64] https://download.docker.com/linux/ubuntu \  
 $(lsb\_release -cs) \  
 stable"

sudo apt-get install docker-ce docker-ce-cli containerd.io

* When Docker is installed and running, it is time to deploy the Docker image using the commands:
  + docker pull ncmdd/omdena-joybot
  + docker run -it -p 8080:8080 ncmdd/omdena-joybot **run** --enable-api --port 8080

When successfully deployed, the API can be access using the following links:

* <public IP of Azure VM>:8080/model/parse – to check the model accuracy
* <public IP of Azure VM>:8080/webhooks/rest/webhook – to be able to interact with the chatbot’s responses

Note:

* The public IP of the Azure VM can be found in the Azure console.
* If you want to run the RASA shell instead, use the command
  + docker run -it -p 8080:8080 ncmdd/omdena-joybot **shell** --enable-api --port 8080