

Docker Deployment Document

****

[1. Objective 3](#_Toc54361282)

[1.1 About Rasa Chatbot 3](#_Toc54361283)

[2. Prerequisites 3](#_Toc54361284)

[3. How to get started 4](#_Toc54361285)

[3.1 Clone the Repository 4](#_Toc54361286)

[3.2 Actions after cloning the repository 6](#_Toc54361287)

[4. Verification/Validation 7](#_Toc54361288)

[4.1 Validate using Post API 7](#_Toc54361289)

[4.2 Validate using curl on console 7](#_Toc54361290)

[4.2 Verification from running containers 8](#_Toc54361291)

[5. Container actions 9](#_Toc54361292)

[5.1 Inside container 9](#_Toc54361293)

[5.2 chat history inside container 9](#_Toc54361294)

[5.3 stopping the containers 10](#_Toc54361295)

[6. Running docker process in background 10](#_Toc54361296)

[7. Rasa Training 10](#_Toc54361297)

[8. Re-Deploying in the case of new commits 11](#_Toc54361298)

# Objective

This document describes the Deployment Details of AI-powered Rasa Chatbot. It focuses on building the necessary Docker images, composing the network, and creating containers. This document is meant for the Software Developer and Maintenance Team members.

## About Rasa Chatbot

The AI-Powered Rasa Chatbot makes use of two services/servers, specifically Rasa Server and the Action Server. Docker Compose allows us to run both of the servers simultaneously while interacting with each other, maintain the chat history in the container, expose the host over API, and many other operations.

# Prerequisites

Docker (version 18 or above) and Docker Compose must be installed on the Server/VM.

* To check Docker version: **docker --version**
* To check docker-compose version: **docker-compose --version**

A picture containing text

Description automatically generated

# How to get started

This section covers the steps needed to create the Compose network, set up both servers, and set up the intercommunication between them.

## 3.1 Clone the Repository

Clone the repository from the GitHub: **git clone** <https://github.com/NSSAC/chatbot.git> .In order to clone you must have the access to the NSSAC/chatbot repository.

Text

Description automatically generated

The cloned repository will have Dockerfile and docker-compose.yml file which contains all the rasa image, rasa-dependencies, network and volumes information. Below is the snapshot of the Dockerfile:Text

Description automatically generated

docker-compose.yml file is responsible for the internal port mappings, creating a compose network for both action and rasa server, and the volumes information.

Below is the snapshot of the docker-compose.yml file:

Text

Description automatically generated

## 3.2 Actions after cloning the repository

* Go to the cloned directory ‘chatbot’ where Dockerfile and docker-compose.yml file exists
* Run **docker build -t rasa/rasa-action-server .** at the level where the Dockerfile exists. It will create a custom rasa action server image. The name of the image must be **rasa/rasa-action-server** or, if you want a custom image name, you also need to change the image name in the docker-compose.yml file. You can see the images by running the **docker images** command.
* As in the snapshot below, rasa/rasa-action-server is our image.

A picture containing text

Description automatically generated

* Now, run **docker-compose up** to start both rasa and action servers. This command will create an internal network, start both the servers simultaneously and interconnect them and you will be able to see the logs on the console. In order to run rasa in the background refer section 6.

A picture containing background pattern

Description automatically generated

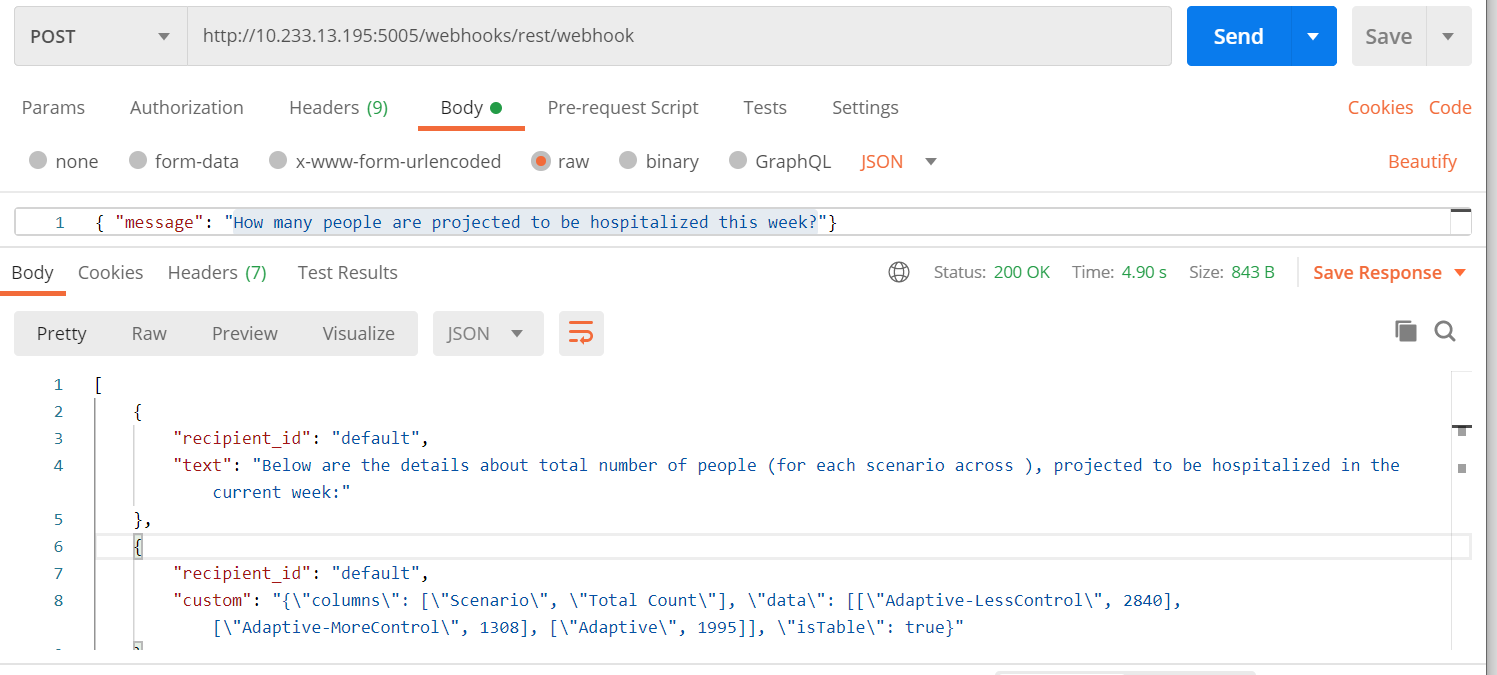
# Verification/Validation

## 4.1 Validate using Post API

Now that both of the servers are up and running, we can access the docker service by making a POST API request to the docker container using the details below:

* + http://<host-ip>:5005/webhooks/rest/webhook
  + Content-type: application/json
  + Body example:

{“message”: “How many people are projected to be hospitalized this week?”}



## 4.2 Validate using curl on console

We can also verify by sending the messages to the bot from the console as shown below:

curl --request POST **\**

--url http://localhost:5005/webhooks/rest/webhook **\**

--header 'content-type: application/json' **\**

--data '{

"message": "How many people are projected to be hospitalized this week?"

}'

Your chatbot should answer similar to this:

[{"recipient\_id":"default","text":"Below are the details about total number of people (for each scenario across ), projected to be hospitalized in the current week:"},

{"recipient\_id":"default","custom":"{\"columns\": [\"Scenario\", \"Total Count\"], \"data\": [[\"Adaptive-LessControl\", 2840], [\"Adaptive-MoreControl\", 1308], [\"Adaptive\", 1995]], \"isTable\": true}"},

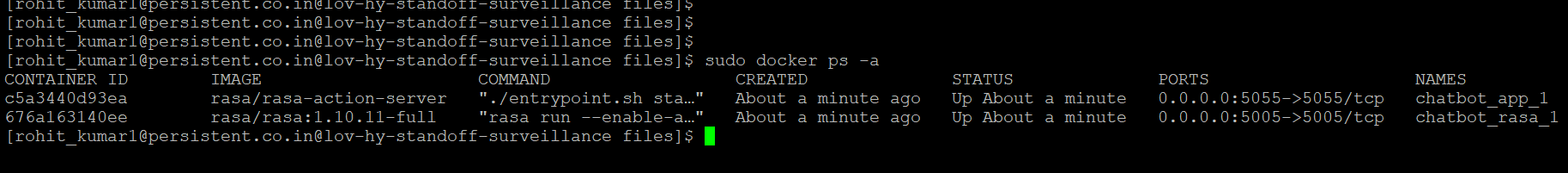
{"recipient\_id":"default","text":"What else do you want to check about?",

"buttons":[{"payload":"\/hospitalization","title":"Hospitalization"},

{"payload":"\/occupied\_beds","title":"Occupied Beds"},{"payload":"\/about\_mrdd","title":"About MRDD"}]

## 4.2 Verification from running containers

To list out all of the running containers, run the **docker ps –a** command.In the snapshot below, both the rasa-action-server and rasa server containers can be seen up and running



# Container actions

## 5.1 Inside container

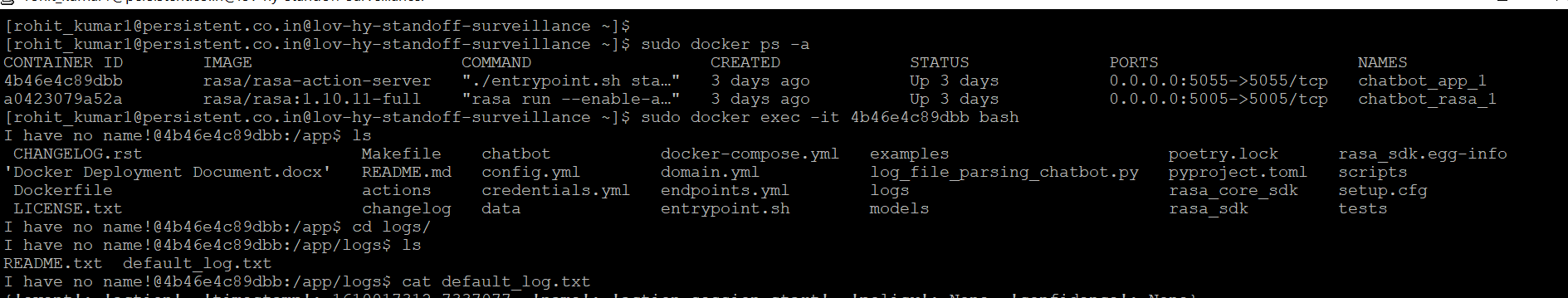
To work inside the running container, run **docker exec -it <rasa-action-server-container-id> bash**

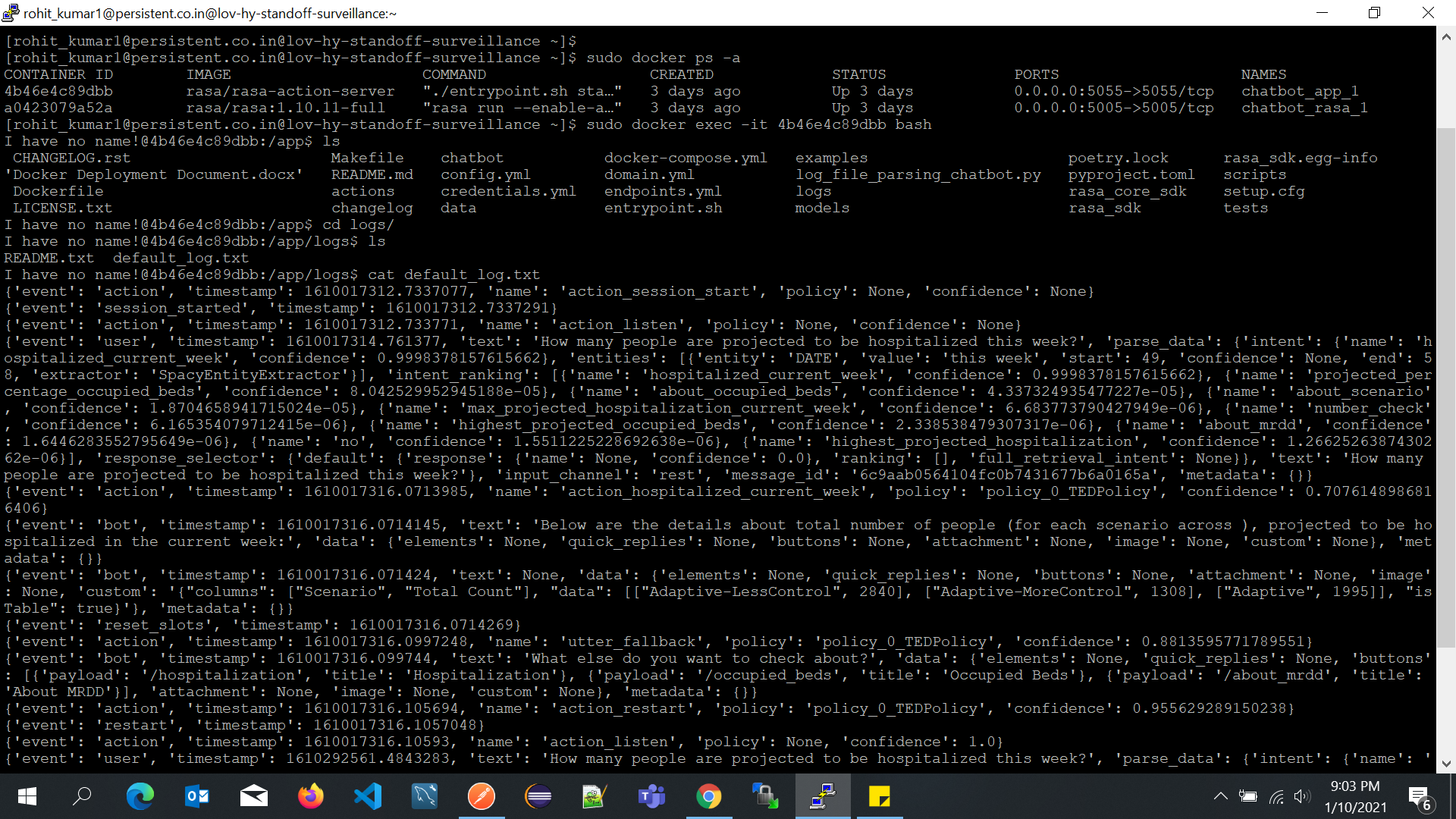
**A screenshot of a computer screen

Description automatically generated**

## 5.2 chat history inside container

Chat history with chatbot is maintained in the directory named logs inside in the rasa action container, and to see the contents of the chat history we need to go to that directory and seeasshown in the snapshot below.





## 5.3 stopping the containers

* To stop the compose network/servers, simply run **ctrl+c**; to remove all volume information, run **docker-compose down** command.
* And to stop the containers, run: **docker stop <container-id>**

# Running docker process in background

To run the docker compose network in the background run **docker-compose up -d**

**A screenshot of a computer screen

Description automatically generated**

* To stop this demon process or running rasa in the background run **docker-compose down**
* As it is running in the background, logs of the rasa servers will not appear on the console. If you still want to see the logs of the containers run **docker container logs <container-id>**

# Rasa Training

To train the Rasa for the newly added intents, or if you don’t have any model available to you, then run

**docker run --user 1000 -it -v $(pwd):/app rasa/rasa:1.10.11-full train**

# Re-Deploying in the case of new commits

* If there is a new commit GitHub, then follow the instructions below:
* Stop the currently running compose network: **ctrl+c**
* Delete the old rasa-action-server image: **docker rmi rasa/rasa-action-server**
* Run **docker build -t rasa/rasa-action-server .**
* Then run **docker-compose up**
* **docker system prune** command removes all the dangling images and stopped containers.
* To inspect or learn more information about a container like ipaddress,port,volume etc.

**run docker inspect <container-id>**

* Point to remember: Always run build and compose commands at the level where the Dockerfile and docker-compose.yml file is present.