

*To send messages with WhatsApp in production, you have to wait for WhatsApp to formally approve your account. But, that doesn't mean you have to wait to start building.* ***Twilio Sandbox for WhatsApp*** *lets you test your app in a developer environment.*

**1.**[**Make account on Twilio**](https://www.twilio.com/try-twilio)

**2.**[**Create new project**](https://www.twilio.com/console/projects/create)

* Select Products
* Choose Programmable SMS
* Click Continue
* Give your project a name
* Click Continue (and skip remaining steps if you want)

**3. Open**[**Programmable SMS Dashboard**](https://www.twilio.com/console/sms/dashboard)**on project console and select WhatsApp *Beta*.**

**4. Learn features of Twilio Sandbox for WhatsApp.**

* Link your WhatsApp phone number to your Sandbox.  
  Shortcut link: <https://api.whatsapp.com/send?phone=+14155238886&text=join%20horn-metal>
* Send a One-Way WhatsApp Message. (Notice that **outbound messages** have a predefined format)
* Test Two-Way Messaging

**5. Configuring the Sandbox (in next part)**

* Setup **webhook** for *incoming messages*
* Optional: Setup a *Status Callback URL*



**1. Setup a Python Virtual Environment**

A virtual environment is a tool that helps to keep dependencies required by different projects separate by creating isolated python virtual environments for them.

* Create a Project Folder.
* Run following command to create a new virtual environment inside your project folder:

python -m venv myvenv

After running above command, a folder named myvenv will get created in your project folder.

* Activate the virtual environment by running following command:
  + For ubuntu and mac users:

source myvenv/bin/activate

* + For windows users:

myvenv\Scripts\activate

**2. Install required Python Packages:**

* [flask](https://github.com/pallets/flask)

pip install flask

* [twilio](https://github.com/twilio/twilio-python)

pip install twilio

**3. Create a Flask App (app.py)**

**from** **flask** **import** Flask, request

**from** **twilio.twiml.messaging\_response** **import** MessagingResponse

app = Flask(\_\_name\_\_)

@app.route("/")

**def** hello():

**return** "Hello, World!"

@app.route("/sms", methods=['POST'])

**def** sms\_reply():

*"""Respond to incoming calls with a simple text message."""*

*# Fetch the message*

msg = request.form.get('Body')

*# Create reply*

resp = MessagingResponse()

resp.message("You said: **{}**".format(msg))

**return** str(resp)

**if** \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=**True**)

**4. Run the flask app**

python app.py

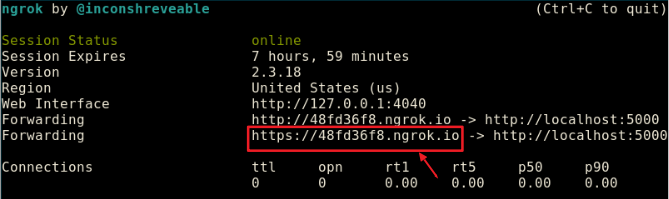
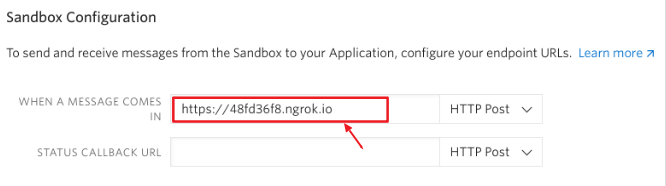
**5. For local testing: Generate Public URL for Webhook using ngrok.io**



*ngrok is a free tool that allows us to tunnel from a public URL to our application running locally.*

* Download [ngrok](https://ngrok.com/download).
* Unzip it.
* Run ngrok from command line (from the location where executable is stored)

./ngrok http 5000

* Copy the **HTTPS Forwarding URL**
* Paste it as the **webhook URL for incoming messages** in your sandbox configuration.





Heroku is a platform as a service (PaaS) that enables developers to build, run, and operate applications entirely in the cloud.

### Create some new files for Heroku deployment

#### 1. Procfile

A Procfile is a mechanism for declaring what commands are run by your application's dynos on the Heroku platform.

web gunicorn app:app

Also, install gunicorn in your virtual environment:

pip install gunicorn

#### 2. runtime.txt

To specify a particular version of Python via your app's runtime.txt

python-3.7.2

#### 3. requirements.txt

Contains all 3rd party libraries required by your app.

Simply do:

pip freeze > requirements.txt

to generate a **requirements.txt** file.

#### 4. .gitignore

.gitignore file specifies patterns which are used to exclude certain files in your working directory from your Git history.

myvenv/

\*.pyc

### Now, its time to create a Heroku app!

#### 1. Setup Git repository ([Download](https://git-scm.com/downloads))

* Initialize a new git repository in your project folder.

git init

* Add all untracked files to git repository by:

git add .

* Commit the changes to git repository by:

git commit -m "YOUR\_COMMIT\_MESSAGE\_HERE"

#### 2. Create a new [heroku account](https://signup.heroku.com/)

#### 3. Download [Heroku CLI](https://devcenter.heroku.com/articles/heroku-cli#download-and-install).

#### 4. Create a new Heroku app.

heroku create <your-app-name>

#### 5. Finally, you are ready to deploy your app by pushing your local git repository to the remote heroku app's git repository by:

git push heroku master

#### 6. To check the logs of your heroku app:

heroku logs

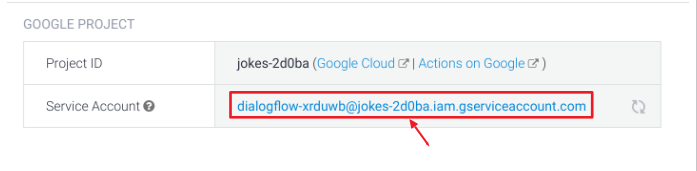
### Now, reset the webhook URL of your sandbox with the URL of your heroku app.[¶](https://render.githubusercontent.com/view/ipynb?color_mode=light&commit=a6ce335542f24103a9fdcb3aff16b7715b8061fc&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f6e696b68696c6b756d617273696e67682f5768617473417070426f745475742f613663653333353534326632343130336139666463623361666631366237373135623830363166632f30332e2532304465706c6f79696e67253230466c61736b2532304170702532306f6e2532304865726f6b752e6970796e62&nwo=nikhilkumarsingh%2FWhatsAppBotTut&path=03.+Deploying+Flask+App+on+Heroku.ipynb&repository_id=176266416&repository_type=Repository#Now,-reset-the-webhook-URL-of-your-sandbox-with-the-URL-of-your-heroku-app.)



**1. Login into [dialogflow console](https://console.dialogflow.com/api-client/" \l "/login)**

**2. Create a new agent or import a pre-built agent**

**3. From settings page of agent, open the service account of your project in Google Cloud Console**



**4. Create a new service account for your project. Download private key for the service account in a JSON file**

**5. Install Python Client for Dialogflow**

* [dialogflow-python-client](https://github.com/googleapis/dialogflow-python-client-v2)

pip install dialogflow

**import** **os**

os.environ["GOOGLE\_APPLICATION\_CREDENTIALS"] = "nikubuntu-1-5d7d63a5b8ca.json"

**import** **dialogflow\_v2** **as** **dialogflow**

dialogflow\_session\_client = dialogflow.SessionsClient()

PROJECT\_ID = "nikubuntu-1"

**def** detect\_intent\_from\_text(text, session\_id, language\_code='en'):

session = dialogflow\_session\_client.session\_path(PROJECT\_ID, session\_id)

text\_input = dialogflow.types.TextInput(text=text, language\_code=language\_code)

query\_input = dialogflow.types.QueryInput(text=text\_input)

response = dialogflow\_session\_client.detect\_intent(session=session, query\_input=query\_input)

**return** response.query\_result

response = detect\_intent\_from\_text("say joke", 12314)

response.fulfillment\_text

response.intent.display\_name

response.intent\_detection\_confidence