Building a WhatsApp GenAI Agent on OCI

article by Anshuman Panda & Mateusz Śliwiński

March 2025

Table of contents

Introduction

WhatsApp Cloud API allows businesses to send and receive messages using cloud hosted version of its WhatsApp Business API. By offering free, secure, and cloud-based hosting, businesses can scale their customer communication and reduce the go-live time.

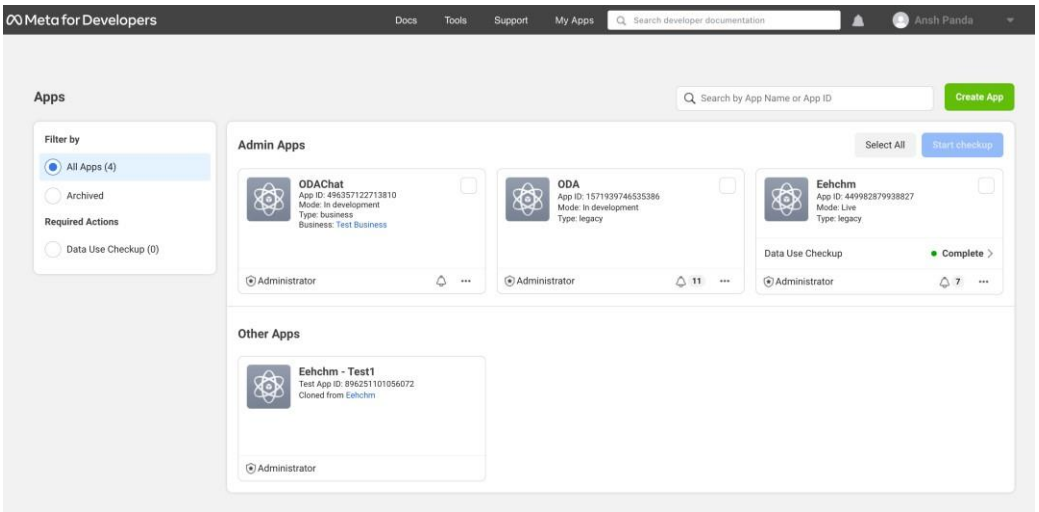
Sandbox Setup

Sandbox is for testing. It is easy to use. You need to have an app created in Meta for Developers my apps. Here are the steps to create a new app if not already created.

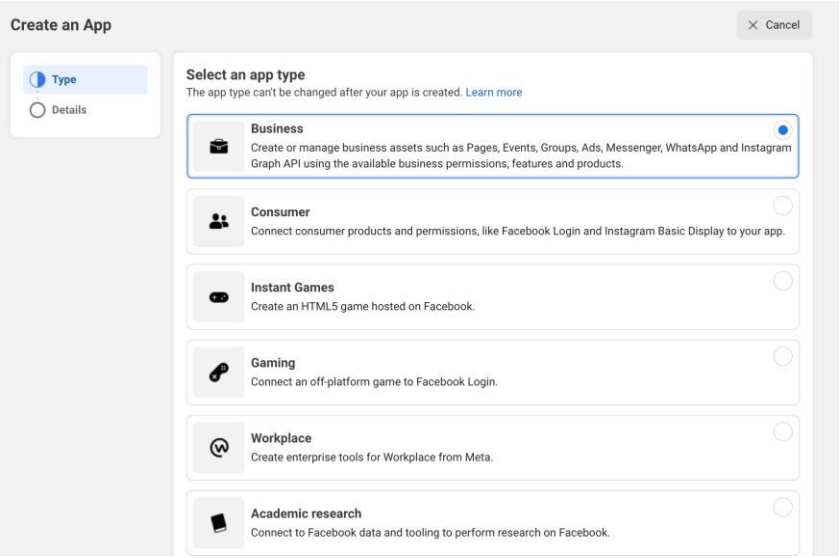
You can download all of the files needed from here. Please ensure the Knowledge Base and AI Agent have been created, with the GenAI Agent endpoint already saved in the configuration file (.env), prior to uploading files to the compute instance.

1. Go to Meta for Developers. If you are visiting for the first time, you will see get started button. Click on "Get Started". Else, go to step 2.

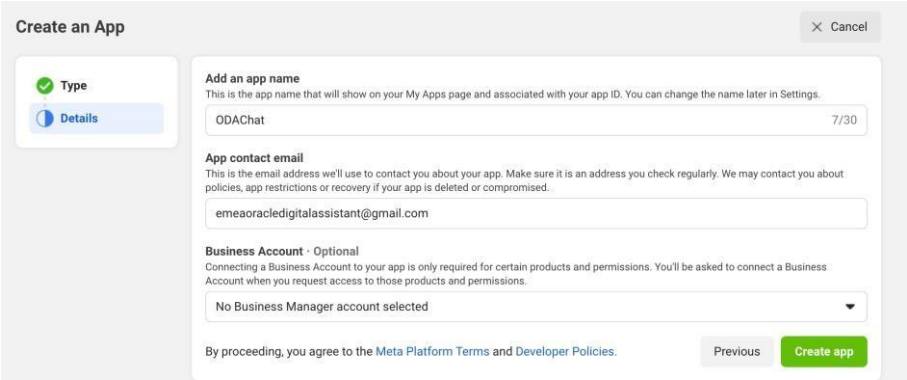
2. Navigate to MyApps buttons on the top right corner.



3. Click on create app and choose "Business" as app type and click on next.

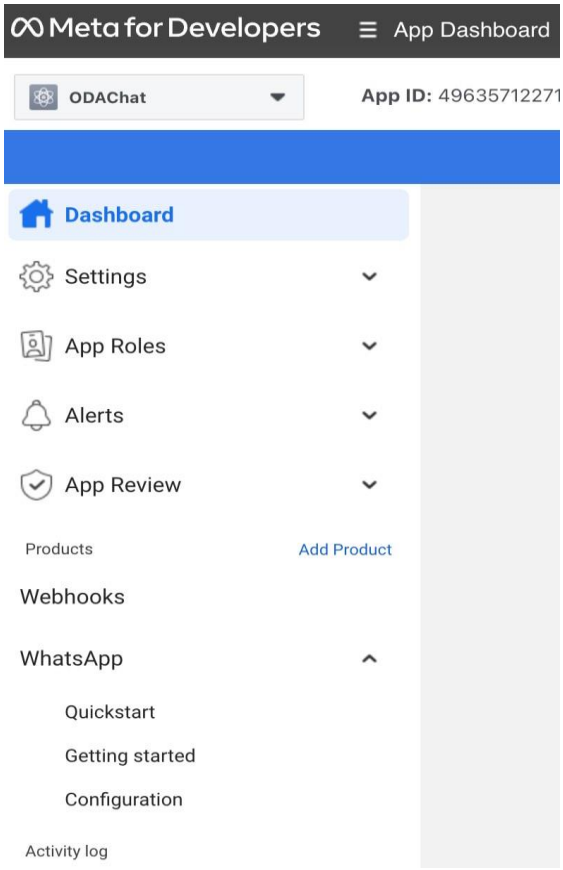


4. Enter the display name for the app.



5. If you have business account, then select it or leave it as it is and click on submit button.

6. Scroll down on the page and select WhatsApp for setup.

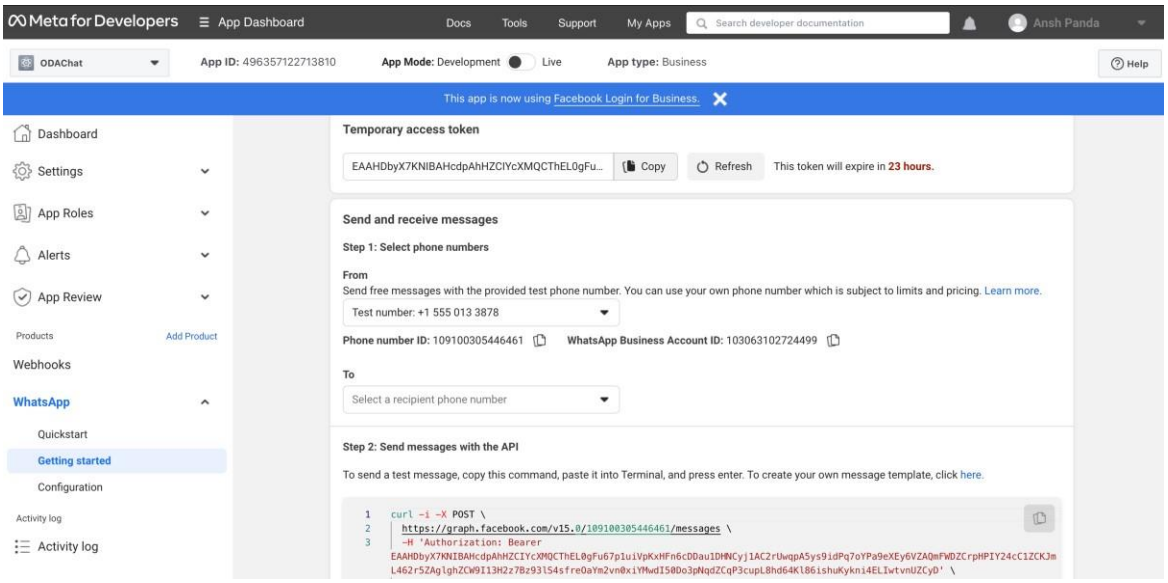


7. Now create a meta business account by selecting "Create a business account" and click on continue.

8. Now you will see the get started page where you can see the temporary access token. Copy this token.

9. Here a temporary **phone number(phone number id)** is generated for testing purpose. Copy this temporary phone number id.

10. Copy the **WhatsApp Business Account Id**

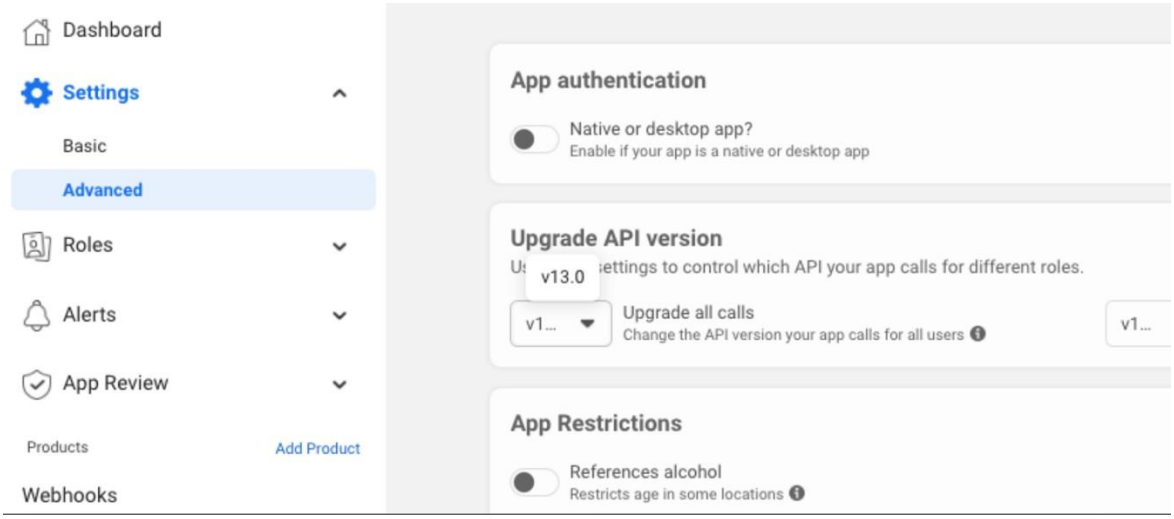


11. Add a recipient (To) phone number and verify it with the code which WhatsApp will send on the number.

12. Now click on test button, for testing and getting the messages on the "Phone Number" mentioned in the sandbox page. Test the flow with the API which is given and see how messages are flowing from test number to the recipient number.

13. After testing, you can add a phone number and verify it using a code which WhatsApp sends. This is actual business number which you want to use for messaging your customer. This number will get added to the from number on the get started page.

14. Check the API version from the configuration section from the left panel in the setting->advanced tab as shown below. You will need to enter it on the webhook at the time of integration.



Correctly setting up config file

Before starting with creating a webhook in compute instance, we should configure our .env file

```
ACCESS_TOKEN=""

APP_ID=""
APP_SECRET=""
RECIPIENT_WAID=""
VERSION="v22.0"
PHONE_NUMBER_ID=""

VERIFY_TOKEN=""

ENDPOINT = "https://inference.generativeai.eu-frankfurt-1.oci.oraclecloud.com" #
Replace with your service endpoint
COMPARTMENT_ID = "ocid1.compartment.oc1.." # Replace with your compartment OCID
AGENT_ENDPOINT_OCID = ""
```

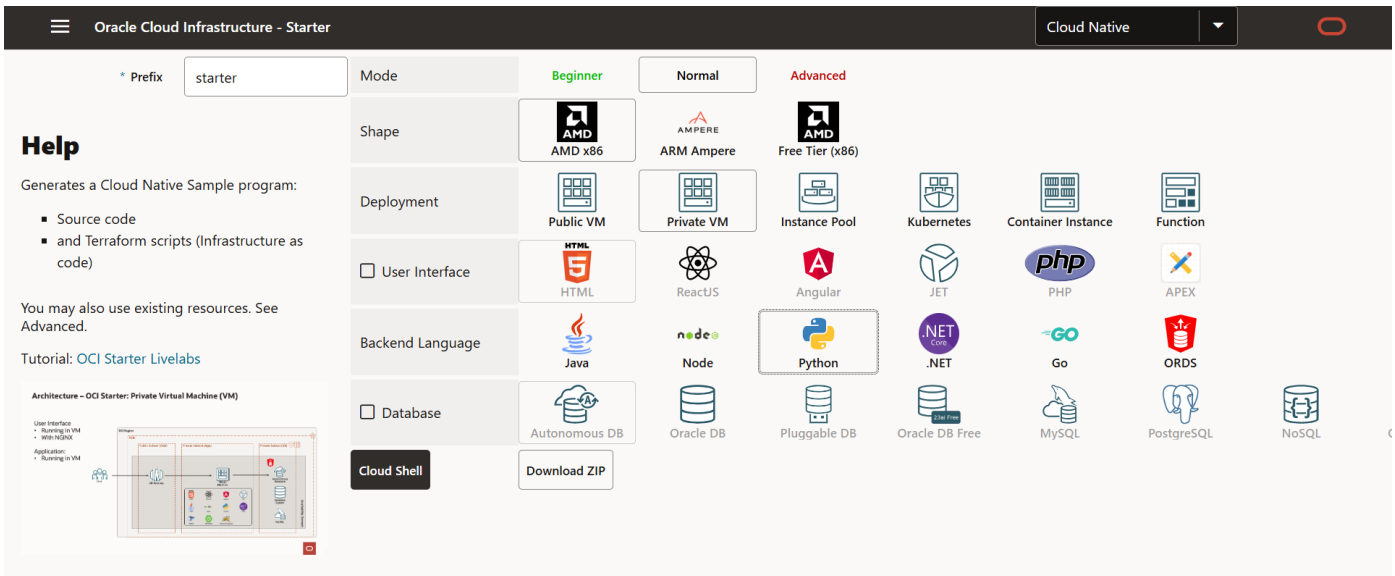
ACCESS_TOKEN -> Generated from developers.facebook.com webpage in API Setup of your application
APP_SECRET -> In your app dashboard in Settings > Basic on developers.facebook.com
RECIPIENT_WAIT -> Your WhatsApp number with country code (e.g., +31612345678)
VERSION -> Facebook API version that you are using
PHONE_NUMBER_ID -> Phone number ID from API Setup of your application on facebook developer
VERIFY_TOKEN -> Token that you create in configuration of your app on developers.facebook

ENDPOINT -> https://inference.generativeai.<region_identifier>.oci.oraclecloud.com, your correct region identifier you can find here.
COMPARTMENT_ID -> OCID of your compartment
AGENT_ENDPOINT_OCID -> Your GenAI Agent endpoint that you have created in OCI

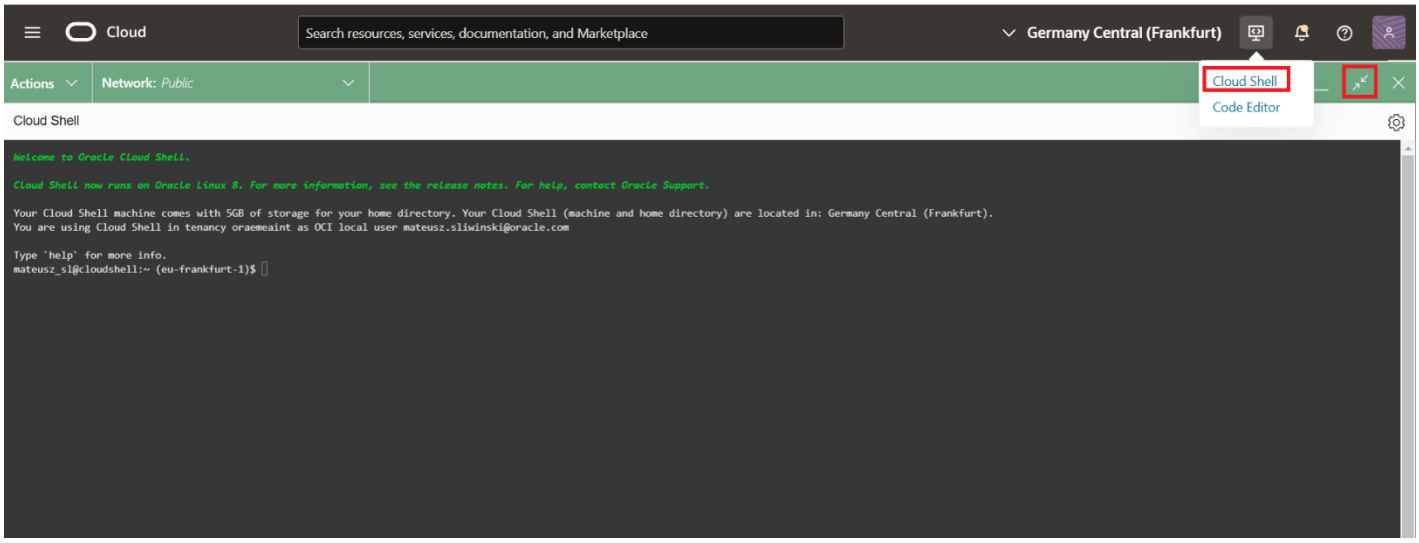
Creating a webhook in compute instance

On a compute instance, we can configure our server to run continuously without interruptions.

1. Go to the webpage https://www.ocistarter.com/
2. Select same as below

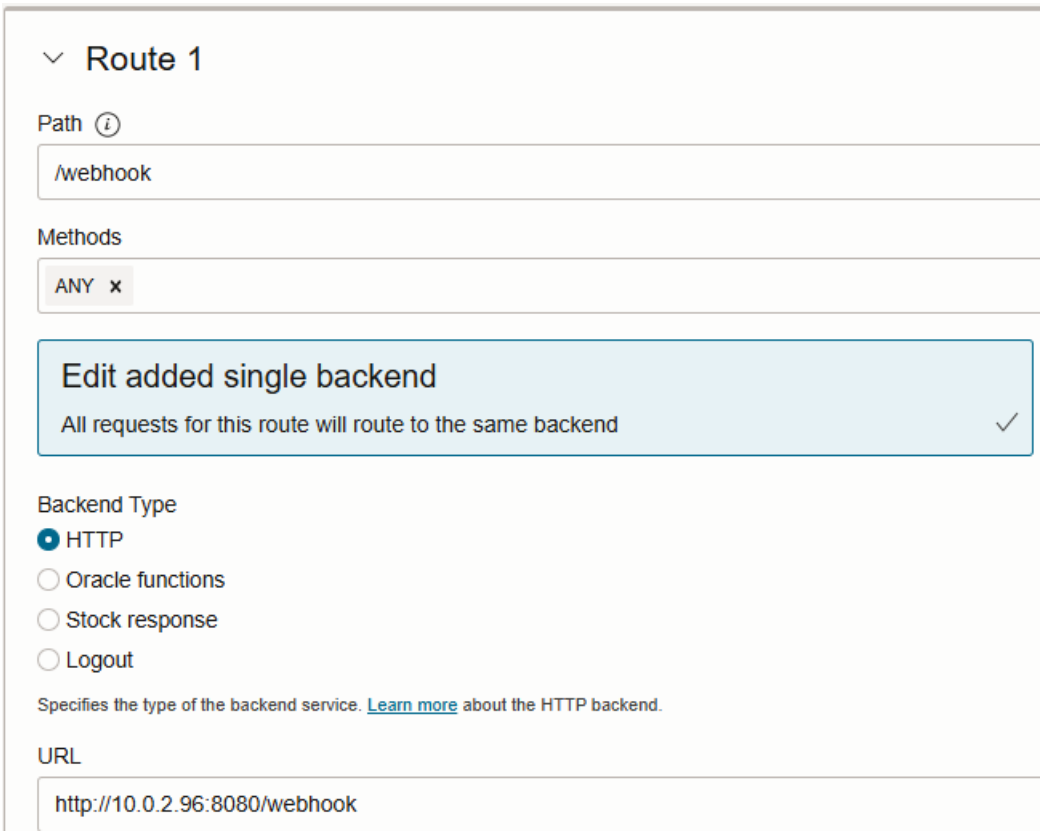


3. Cloud Shell download, copy output
4. Go to Oracle Cloud and open Cloud Shell, maximize the window.

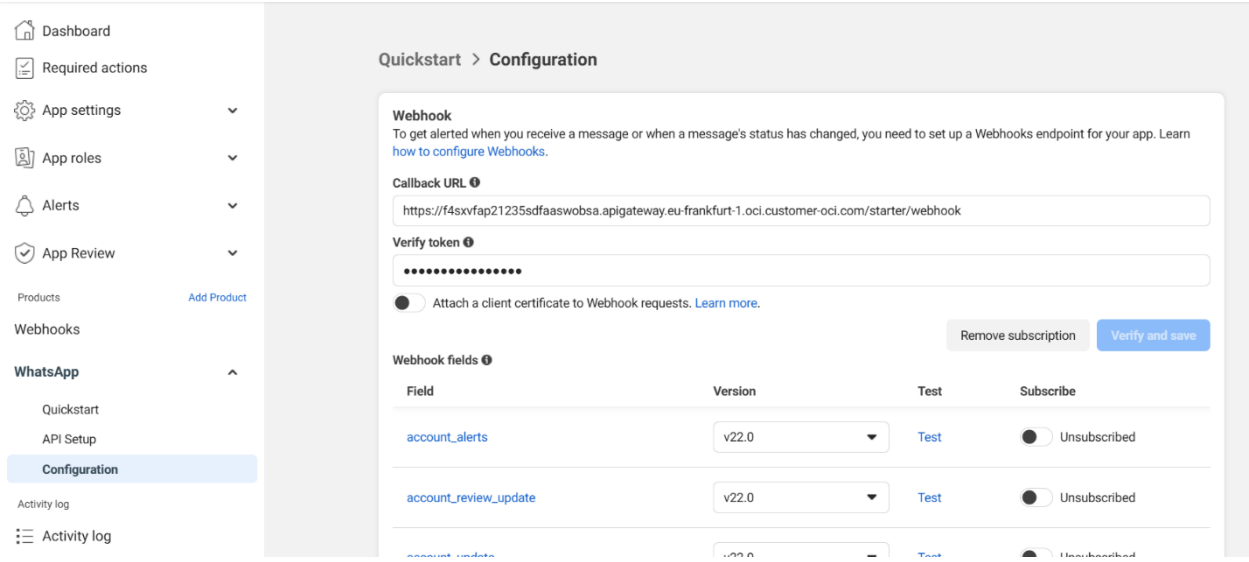


5. create directory by typing in *mkdir name_of_your_directory* e.g. *mkdir oci-whatsapp*
6. Type in cd *name_of_your_directory*
7. Paste in copied commands from **Step 3**
8. Type in ./starter.sh -> Build -> Create new compartment if needed.
9. Meanwhile we can proceed to creating
10. Go to Developer Services -> Gateways, there new Gateway has been created

11. Open your new Gateway -> Deployments -> Open your one deployment.
12. Copy and save the end point.
13. Press on Edit
14. Go to Routes and we will be editing Route 1, set Path to */webhook*, methods *ANY* backend type as *HTTP*, URL is your private compute instance IP with port and /webhook e.g. *http://10.0.2.96:8080/webhook*



15. Save, go back to your Deployments and copy your endpoint address, save it in safe place and add ***/webhook*** at the end. It should look like *https://f4sxvfap21235sdfaaswobsa.apigateway.eu-frankfurt-1.oci.customer-oci.com/starter/webhook*
16. Go to the Facebook developers page, and adjust your webhook. Verify token needs to be the same as you have in your ".env" file. Press on Verify and save.



17. Go back to OCI Cloud Shell and wait for it to finish
18. Type in ./starter.sh -> Advanced -> Key, copy and save it
19. Type in ./starter.sh -> Advanced -> Compute to connect to your Compute instance. You can create folder for
20. **If you have downloaded whole package from this instruction, you can skip this step.**
    Now we need to adjust whatsapp code to work with GenAI Agent

Below functions need to be adjusted as below. File ***/app/utils/whatsapp_utils.py***

```
import logging
```

```python
from flask import current_app, jsonify
import json
import requests
import re
from dotenv import load_dotenv


from langchain_community.chat_models import ChatOCIGenAI
from langchain.schema import HumanMessage, SystemMessage, AIMessage
import oci

import os
```

```python
SESSION_STORE = {}
def generate_response(prompt: str, user_id: str) -> str:
    """
    Calls OCI Generative AI Agent Runtime to create (if needed) or reuse
    a session for this user, then sends the user prompt. Returns the answer.
    """
    try:
        # Check if we already have a session ID for this user
        session_id = SESSION_STORE.get(user_id)

        # If no session for this user, create a new one
        if not session_id:
            resp = GENAI_AGENT_RUNTIME_CLIENT.create_session(
                agent_endpoint_id=AGENT_ENDPOINT_OCID,
                create_session_details=oci.generative_ai_agent_runtime.models.Create
SessionDetails(
                    description='session',
                    display_name='session'
                )
            )
            session_id = resp.data.id
            # Store the session ID in our global dictionary
            SESSION_STORE[user_id] = session_id

        # Now always call chat with the existing or newly created session ID
        resp_chat = GENAI_AGENT_RUNTIME_CLIENT.chat(
            agent_endpoint_id=AGENT_ENDPOINT_OCID,
            chat_details=oci.generative_ai_agent_runtime.models.ChatDetails(
                session_id=session_id,
                user_message=prompt
            )
        )

        # The agent's response text:
        assistant_answer = resp_chat.data.message.content.text
        citations = resp_chat.data.message.content.citations

        # Build a simple string of citations
```

```
        citations_text = ""
        for i, c in enumerate(citations, start=1):
            t = getattr(c, "title", "N/A")
            p = getattr(c, "page_numbers", "N/A")

            citations_text += (
                f"Citation {i}:\n"
                f"  Title: {t}\n"
                f"  Page Numbers: {p}\n"
            )

        # Combine the answer with the citations
        return f"{assistant_answer}\n\n\n{citations_text}"

    except Exception as e:
        logging.error(f"Error generating response via OCI Agent: {e}")
        return "I'm sorry, but I couldn't process your request at this time."
```

```
def process_whatsapp_message(body):
    wa_id = body["entry"][0]["changes"][0]["value"]["contacts"][0]["wa_id"]
    name = body["entry"][0]["changes"][0]["value"]["contacts"][0]["profile"]["name"]

    message = body["entry"][0]["changes"][0]["value"]["messages"][0]
    message_body = message["text"]["body"]

    response = generate_response(message_body, user_id=wa_id)

    data = get_text_message_input(current_app.config["RECIPIENT_WAID"], response)
    send_message(data)
```

On the top of the file you can add below for easy access to the variables, or create config file

```
config = oci.config.from_file()
load_dotenv()
GENERATE_MODEL = os.getenv("GENERATE_MODEL")
ENDPOINT = os.getenv("ENDPOINT")
AGENT_ENDPOINT_OCID = os.getenv("AGENT_ENDPOINT_OCID")
ACCESS_TOKEN = os.getenv("ACCESS_TOKEN")


GENAI_AGENT_RUNTIME_CLIENT =
oci.generative_ai_agent_runtime.GenerativeAiAgentRuntimeClient(
    config=config,
    service_endpoint=ENDPOINT
)
```

Your main file to start the flash ***run.py*** should be running on the same port as we set it in Gateway Routes, here we set to 8080

```
import logging
```

```
from app import create_app

app = create_app()

if __name__ == "__main__":
    logging.info("Flask app started")
    app.run(host="0.0.0.0", port=8080)
```

21. On Windows PC open CMD or Shell, we will be uploading files from your PC to Compute Instance, we will be using commands *ssh -i <private_key_file> <username>@<ip_address>* and *scp -i <private_key_path> <file_path> opc@<ip_address>:.*

    First we will be checking connecting to bastion, you can check Bastion public IP by checking your compute instaces. Type the command *ssh -i /path/to/your/key opc@Bastion_IP* e.g. *ssh -i key.key opc@130.61.33.198*. If you encounter an error you need to **delete** "*_starter*" from the key you copied and add one or two empty lines at the end.

```
Load key "private_key.key": invalid format
opc@130.61.33.198: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
```

```
-----BEGIN PRIVATE KEY-----_starter
```

After deleting "*_starter" we should connect*

```
C:\Users\Mateusz\Downloads\new-python-whatsapp-bot-main>ssh -i private_key.pem opc@130.61.33.198
Activate the web console with: systemctl enable --now cockpit.socket

[opc@starter-bastion ~]$ exit
logout
Connection to 130.61.33.198 closed.
```

We can exit connection and upload our files using command:
scp -i <private_key_path> <file_path> opc@<bastion_public_ip>:.
It is important to have colon and dot ':.' at the end
Type in this command twice for Key and Zip

```
C:\Users\Mateusz\Downloads\new-python-whatsapp-bot-main>scp -i private_key.pem whatsapp.zip opc@130.61.33.198:.
whatsapp.zip                                                                  100%   23KB 319.6KB/s   00:00
```

22. We have uploaded our files from our PC to bastion, after ssh connection to our bastion we can type in command ls to see our files

```
C:\Users\Mateusz\Downloads\new-python-whatsapp-bot-main>ssh -i private_key.pem opc@130.61.33.198
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Fri Feb 21 12:13:15 2025 from 193.28.84.145
l[opc@starter-bastion ~]$ ls -h
private_key.pem  whatsapp.zip
```

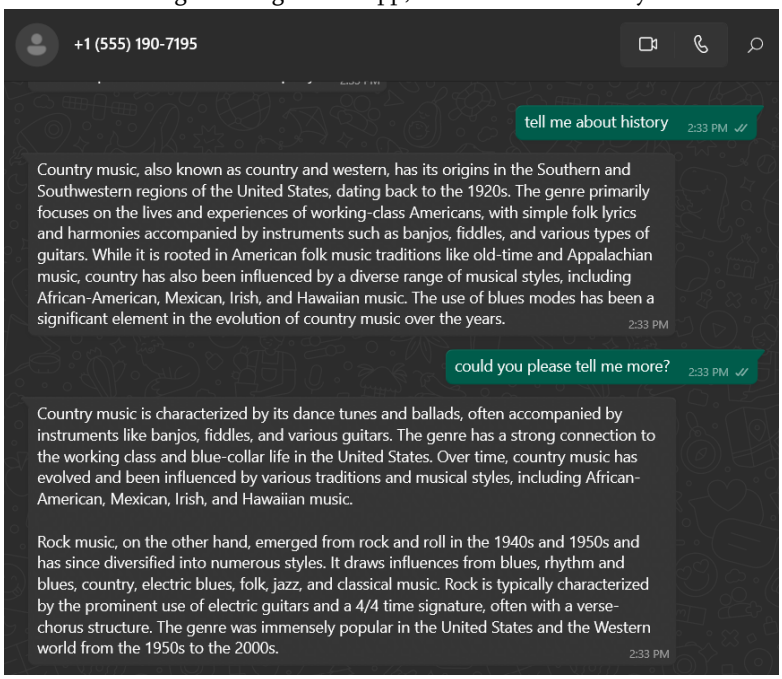
23. Now we can upload files to our compute instance, we need to secure our key with this command "*Chmod 600 <your_private_key>*", otherwise we will get this error message

```
[opc@starter-bastion ~]$ scp -i private_key.pem whatsapp.zip opc@10.0.2.96:.
The authenticity of host '10.0.2.96 (10.0.2.96)' can't be established.
ECDSA key fingerprint is SHA256:BM4W2i5xraeYreECKwfOge2vf/9x9E57lzMswiaeI5M.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.96' (ECDSA) to the list of known hosts.
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0664 for 'private_key.pem' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "private_key.pem": bad permissions
opc@10.0.2.96: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
lost connection
```

24. After securing our key we can upload files to our compute instance
    scp -i <private_key_path> <file_path> opc@<compute_private_ip>:.

```
[opc@starter-bastion ~]$ chmod 600 private_key.pem
[opc@starter-bastion ~]$ scp -i private_key.pem whatsapp.zip opc@10.0.2.96:.
whatsapp.zip                                                  100%   23KB  18.5MB/s   00:00
```

25. You can create folder with mkdir and unzip you zip file there
26. "Sudo systemclt stop app" to stop existing process running on port 8080
27. In the folder you unzipped your files there is requirements file, we need to install python libraries
    pip3 install -r requirements.txt
28. You can now run your application and test connection with Facebook developer webhook
    python3 run.py
    And chat with agent using WhatsApp, it will use data from your knowledge base to provide answers



Note: If you encounter any errors with installing Python libraries please update Python and check if your *python3 –version* is at least 3.12.