Creating a Webex ChatBot

In this lab, you will set up a functional Webex chatbot, integrating it with Webex Teams and ensuring its responsiveness to user messages. Webex Chatbots streamline interactions within the Webex platform, harnessing Python, Flask, ngrok, and webhooks to create a responsive and interactive environment. Bots integrate smoothly with your business workflows, offering a wide range of functionalities that can be tailored to meet the specific needs of your operation. Some use cases of a Webex Chatbot include:

- Automated responses to frequently asked questions
- Scheduling and managing meetings directly from the chat interface
- Custom notifications and alerts based on user-defined triggers
- Gathering feedback and surveys within a chat or space
- Integrating with enterprise applications to pull in data and perform actions within the chat flow.
- Offering interactive tutorials or guidance through complex workflows

Activity Objective

Description

- Make a local copy of a chatbot repository
- Create a webhook and register with Webex API
- Configure a bot and confirm functionality

Task 1: Setup the Project

Activity Procedure

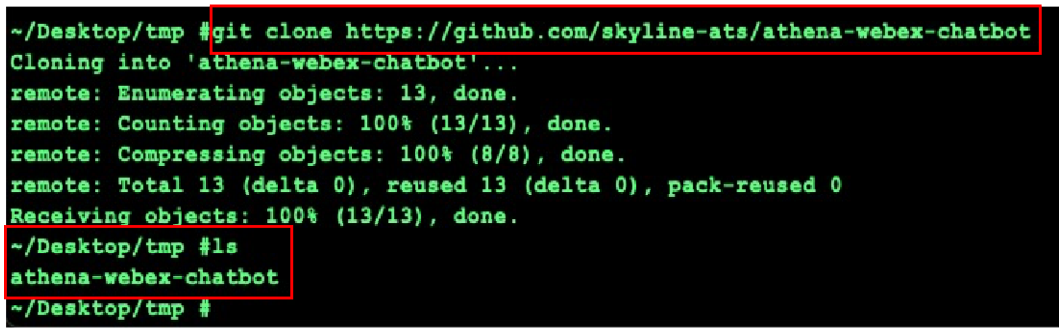
Complete these steps to retrieve your Webex developer sandbox API token:

Step 1: Open your preferred command-line interface (CLI).

Step 2: In a directory of your choice, enter the command below:

```
git clone https://github.com/skyline-ats/athena-webex-chatbot.git
```

This command will clone a GitHub repository, which is a process of creating a local copy of the code provided by the author. Cloning sets you up for an independent workspace where you can contribute or modify without affecting the original code. If Git isn't part of your setup, you can access the URL above in a web browser, download the repository as a zip file and extract it to a local directory on your computer.



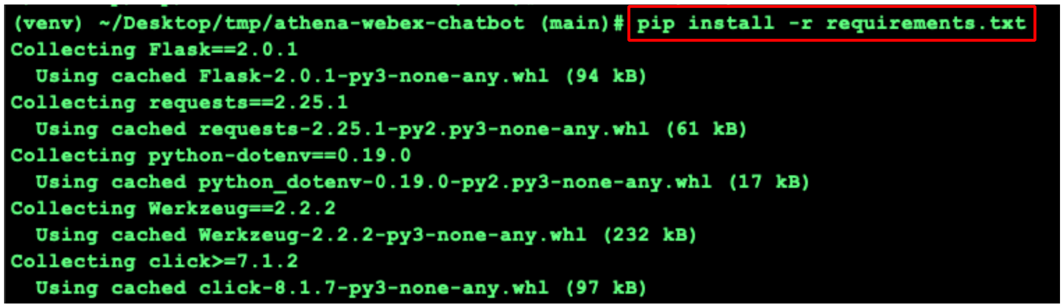
```
~/Desktop/tmp #git clone https://github.com/skyline-ats/athena-webex-chatbot
Cloning into 'athena-webex-chatbot'...
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 13 (delta 0), reused 13 (delta 0), pack-reused 0
Receiving objects: 100% (13/13), done.
~/Desktop/tmp #ls
athena-webex-chatbot
~/Desktop/tmp #
```

Step 3: Enter the command `cd athena-webex-chatbot` in the CLI to change your working directory to the project's root folder.

Step 4: Enter the command `python -m venv venv` to create a Python virtual environment. A virtual environment is a directory that contains a Python installation for a particular version of Python, plus additional packages, unique from the globally installed Python versions and packages. This keeps your project's dependencies isolated from those of other projects.

Step 5: Enter the command `source venv/bin/activate` on macOS/Linux or `venv\Scripts\activate` on Windows to activate the virtual environment.

Step 6: After activation, install the required packages with `pip install -r requirements.txt`, ensuring your bot has access to all the necessary Python modules.

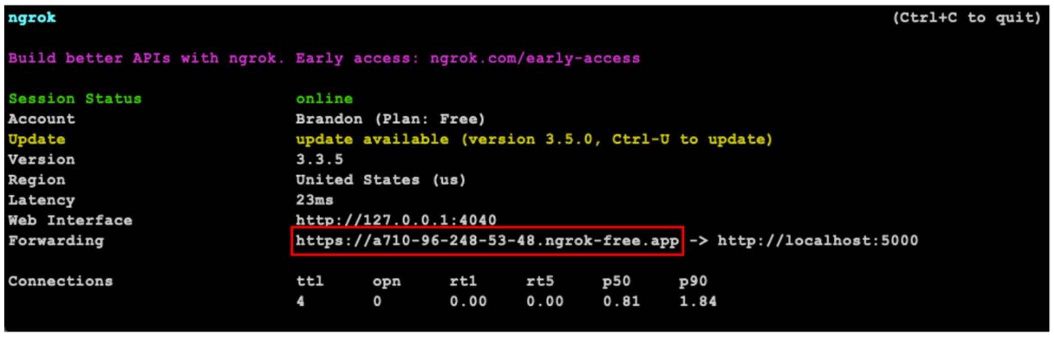


```
(venv) ~/Desktop/tmp/athena-webex-chatbot (main)# pip install -r requirements.txt
Collecting Flask==2.0.1
  Using cached Flask-2.0.1-py3-none-any.whl (94 kB)
Collecting requests==2.25.1
  Using cached requests-2.25.1-py2.py3-none-any.whl (61 kB)
Collecting python-dotenv==0.19.0
  Using cached python_dotenv-0.19.0-py2.py3-none-any.whl (17 kB)
Collecting Werkzeug==2.2.2
  Using cached Werkzeug-2.2.2-py3-none-any.whl (232 kB)
Collecting click>=7.1.2
  Using cached click-8.1.7-py3-none-any.whl (97 kB)
```

Step 7: The next step is to make your local development server accessible to the internet using Ngrok, which is a useful tool for creating a secure tunnel to your localhost. This is especially important when you're working with webhooks, as it allows external services like Webex to reach your server. After installing Ngrok, you can use the following command to expose your local port 5000 via an ngrok tunnel:

```
ngrok http 5000
```

Step 8: Note the Ngrok URL from the CLI output; this URL is the bridge between Webex and your local server, allowing them to communicate. Do not stop this service or close this terminal window.



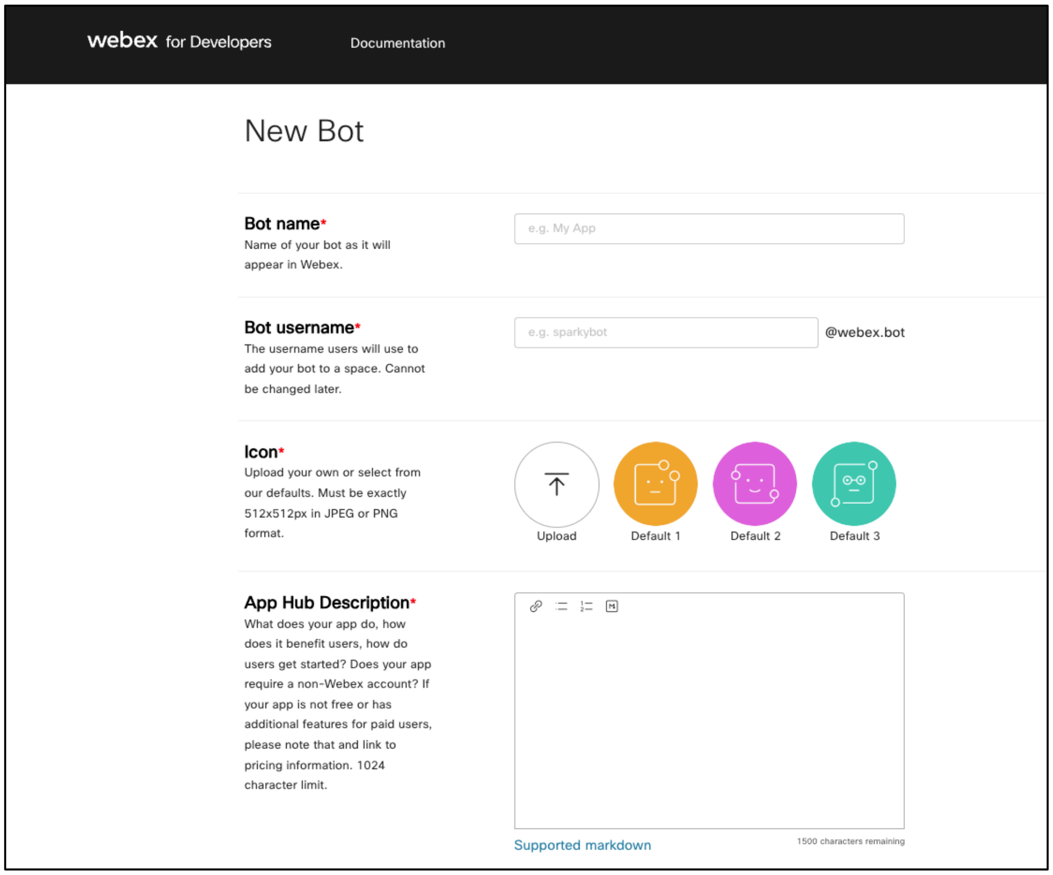
```
ngrok                                                                        (Ctrl+C to quit)

Build better APIs with ngrok. Early access: ngrok.com/early-access

Session Status                online
Account                       Brandon (Plan: Free)
Update                        update available (version 3.5.0, Ctrl-U to update)
Version                       3.3.5
Region                        United States (us)
Latency                       23ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://a710-96-248-53-48.ngrok-free.app -> http://localhost:5000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              4       0       0.00    0.00    0.81    1.84
```

Step 9: Navigate to [the Cisco FedRAMP New Bot page (https://developer-usgov.webex.com/my-apps/new/bot)](https://developer-usgov.webex.com/my-apps/new/bot) to create a bot. Login with your Cisco ID if prompted.



New Bot

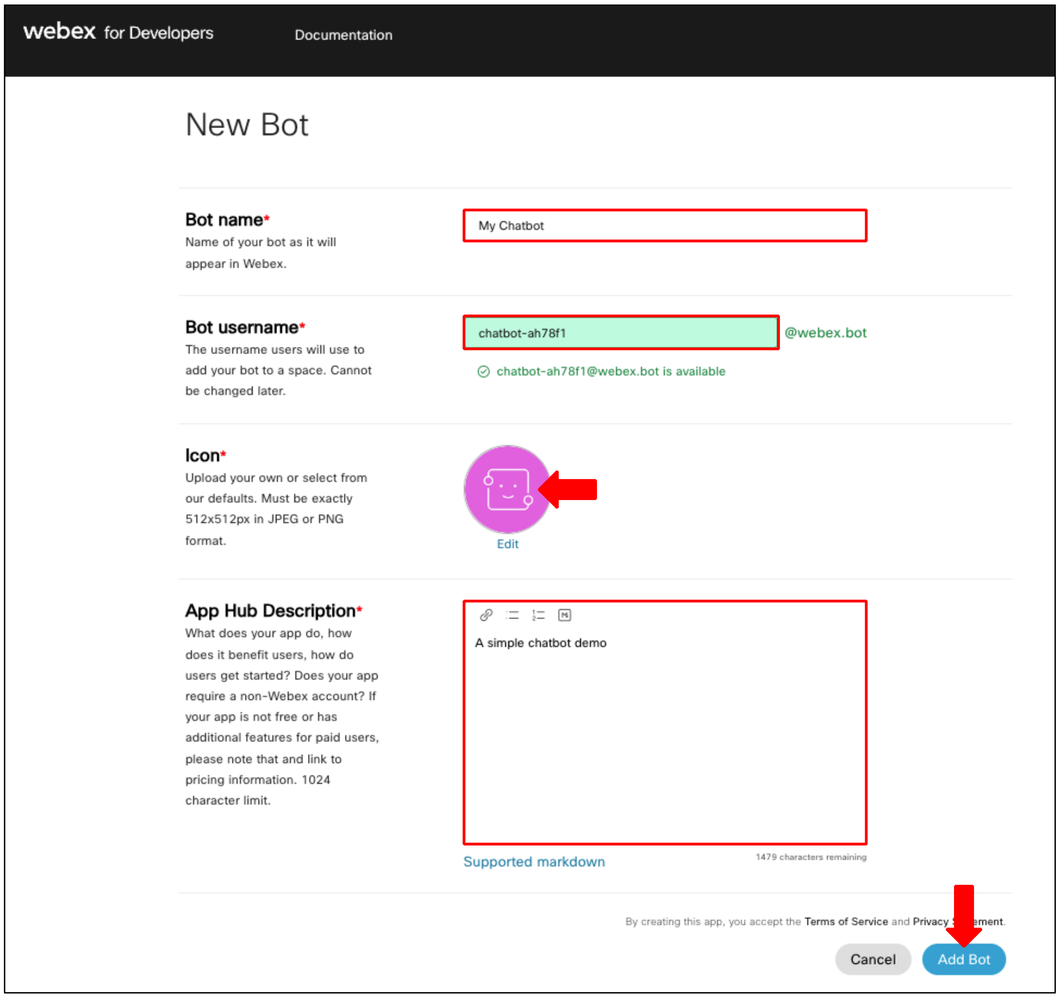
Bot name* Name of your bot as it will appear in Webex. — e.g. My App

Bot username* The username users will use to add your bot to a space. Cannot be changed later. — e.g. sparkybot @webex.bot

Icon* Upload your own or select from our defaults. Must be exactly 512x512px in JPEG or PNG format.

App Hub Description* What does your app do, how does it benefit users, how do users get started? Does your app require a non-Webex account? If your app is not free or has additional features for paid users, please note that and link to pricing information. 1024 character limit.

Step 10: Define your bot's name, username, icon, and description, which are essential for users to identify and interact with your bot. Name your bot **My Chatbot** and pick a unique **Bot username**, like `chatbot-[random hash]`. Select an **Icon**, write a brief **App Hub Description** for your bot, and click **Add Bot**. Upon creation of your bot, Webex will provide you with an access token, which is necessary for the bot's authentication process with the Webex APIs.



New Bot

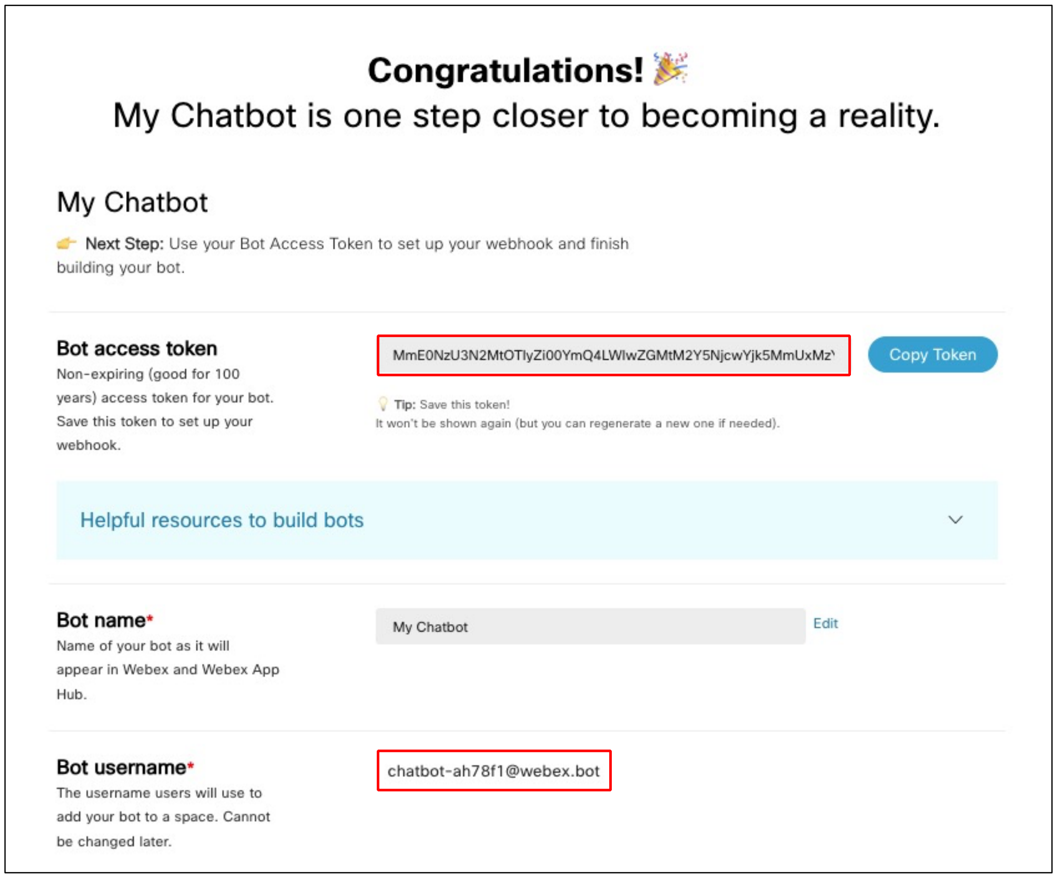
Bot name*
Name of your bot as it will appear in Webex.
My Chatbot

Bot username*
The username users will use to add your bot to a space. Cannot be changed later.
chatbot-ah78f1 @webex.bot
chatbot-ah78f1@webex.bot is available

Icon*
Upload your own or select from our defaults. Must be exactly 512x512px in JPEG or PNG format.
Edit

App Hub Description*
What does your app do, how does it benefit users, how do users get started? Does your app require a non-Webex account? If your app is not free or has additional features for paid users, please note that and link to pricing information. 1024 character limit.
A simple chatbot demo
Supported markdown
1479 characters remaining

By creating this app, you accept the Terms of Service and Privacy Statement.



Congratulations!
My Chatbot is one step closer to becoming a reality.

My Chatbot

Next Step: Use your Bot Access Token to set up your webhook and finish building your bot.

Bot access token
Non-expiring (good for 100 years) access token for your bot. Save this token to set up your webhook.

MmE0NzU3N2MtOTIyZi00YmQ4LWIwZGMtM2Y5NjcwYjk5MmUxMz

Copy Token

Tip: Save this token!
It won't be shown again (but you can regenerate a new one if needed).

Helpful resources to build bots

Bot name*
Name of your bot as it will appear in Webex and Webex App Hub.

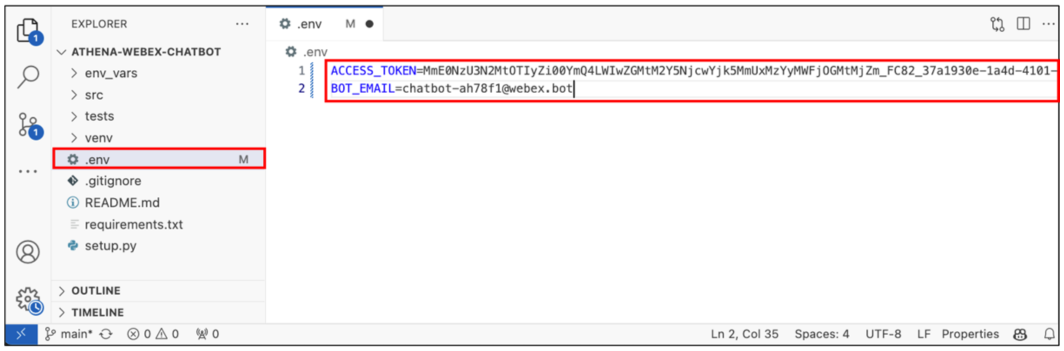
My Chatbot

Edit

Bot username*
The username users will use to add your bot to a space. Cannot be changed later.

chatbot-ah78f1@webex.bot

Step 11: Open Visual Studio Code, then open the Python chatbot project directory in VS Code. Once you have VS Code open, select **File > Open Folder**, navigate to, and select the Python project you cloned from GitHub. Open the file named `.env` and update the values of `ACCESS_TOKEN` and `BOT_EMAIL` with the values from the bot you created. This step is critical for your bot to operate within the Webex ecosystem. Save the file when you are done editing.

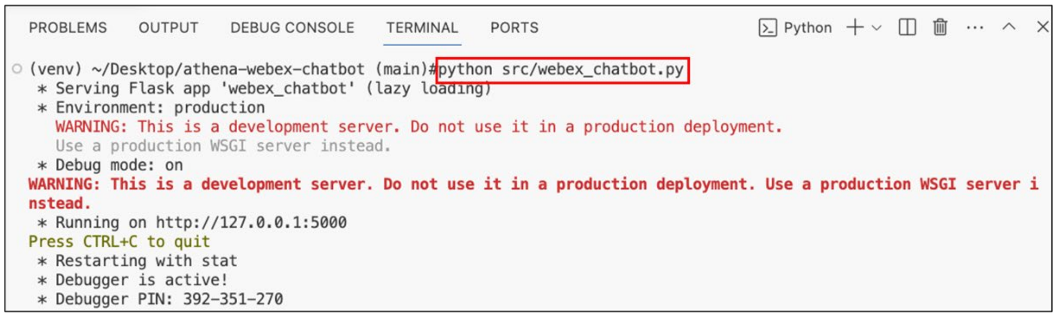


```
ACCESS_TOKEN=MmE0NzU3N2MtOTIyZi00YmQ4LWIwZGMtM2Y5NjcwYjk5MmUxMzYyMWFjOGMtMjZm_FC82_37a1930e-1a4d-4101-
BOT_EMAIL=chatbot-ah78f1@webex.bot
```

Step 12: With everything in place, it's time to start the Flask application. Return to the CLI you used to clone the git repo, or open the VS Code integrated terminal with the shortcut **ctrl +** `. From your project directory, execute the command below:

```
python -m src.webex_chatbot
```

This command fires up the Flask server, which will host your bot's functionalities. Ensuring no errors are thrown at this point is vital for the next steps to work correctly.

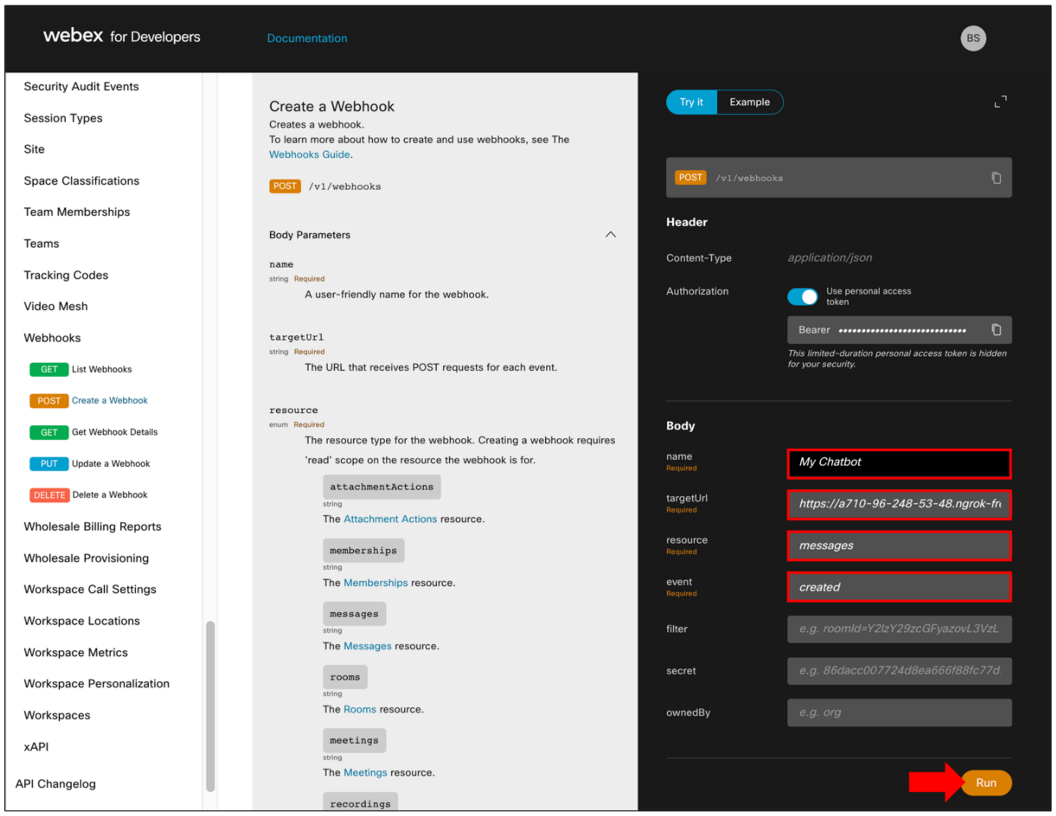


```
○ (venv) ~/Desktop/athena-webex-chatbot (main)#python src/webex_chatbot.py
  * Serving Flask app 'webex_chatbot' (lazy loading)
  * Environment: production
    WARNING: This is a development server. Do not use it in a production deployment.
    Use a production WSGI server instead.
  * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server i
nstead.
  * Running on http://127.0.0.1:5000
Press CTRL+C to quit
  * Restarting with stat
  * Debugger is active!
  * Debugger PIN: 392-351-270
```

Step 13: Webhooks are user-defined HTTP callbacks, which are triggered by specific events. When that event occurs, the source site makes an HTTP request to the URL configured for the webhook. Navigate to the Webex for Developers - Create Webhook Page to create a new webhook, which will notify your server whenever a new message is sent to your bot. You will need to provide your Ngrok URL as the target and specify the types of events you want your webhook to listen for. This connection is essential for your bot to receive messages from users and respond accordingly. Use the following values in the associated input fields, then click the **Run** button:

- name: A name for your webhook (e.g., "My Chatbot Webhook").
- targetUrl: Your ngrok URL (e.g., http://12345.ngrok.io).
- resource: messages.
- event: created.
- secret: A secret phrase for added security (optional, leave empty).
- filter: Leave empty to receive all messages.

On submission you should see a JSON response appear below the **Run** button, which will contain information about the webhook you just created.


webex for Developers   Documentation   BS

Security Audit Events
Session Types
Site
Space Classifications
Team Memberships
Teams
Tracking Codes
Video Mesh
Webhooks
GET List Webhooks
POST Create a Webhook
GET Get Webhook Details
PUT Update a Webhook
DELETE Delete a Webhook
Wholesale Billing Reports
Wholesale Provisioning
Workspace Call Settings
Workspace Locations
Workspace Metrics
Workspace Personalization
Workspaces
xAPI
API Changelog

Create a Webhook
Creates a webhook.
To learn more about how to create and use webhooks, see The Webhooks Guide.

POST /v1/webhooks

Body Parameters

name
string Required
A user-friendly name for the webhook.

targetUrl
string Required
The URL that receives POST requests for each event.

resource
enum Required
The resource type for the webhook. Creating a webhook requires 'read' scope on the resource the webhook is for.

attachmentActions
string
The Attachment Actions resource.

memberships
string
The Memberships resource.

messages
string
The Messages resource.

rooms
string
The Rooms resource.

meetings
string
The Meetings resource.

recordings

Try it   Example

POST /v1/webhooks

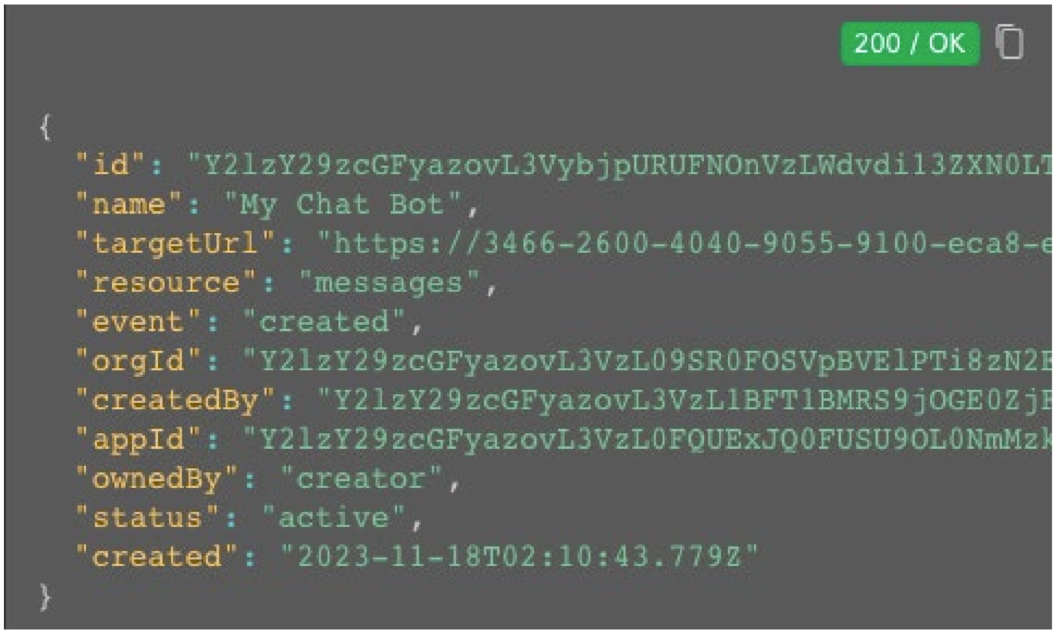
Header

Content-Type   application/json
Authorization   Use personal access token
Bearer ••••••••••••••••••••••••••••
This limited-duration personal access token is hidden for your security.

Body

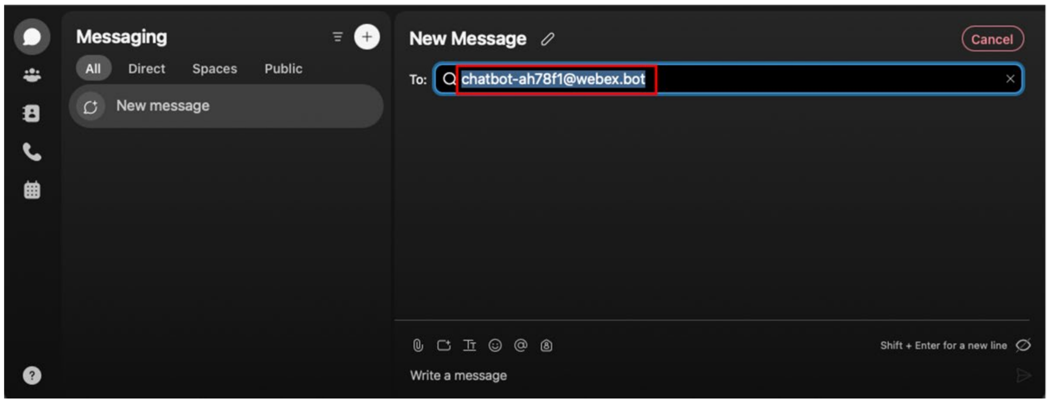
name Required	My Chatbot
targetUrl Required	https://a710-96-248-53-48.ngrok-fr
resource Required	messages
event Required	created
filter	e.g. roomId=Y2lzY29zcGFyazovL3VzL
secret	e.g. 86dacc007724d8ea666f88fc77d
ownedBy	e.g. org

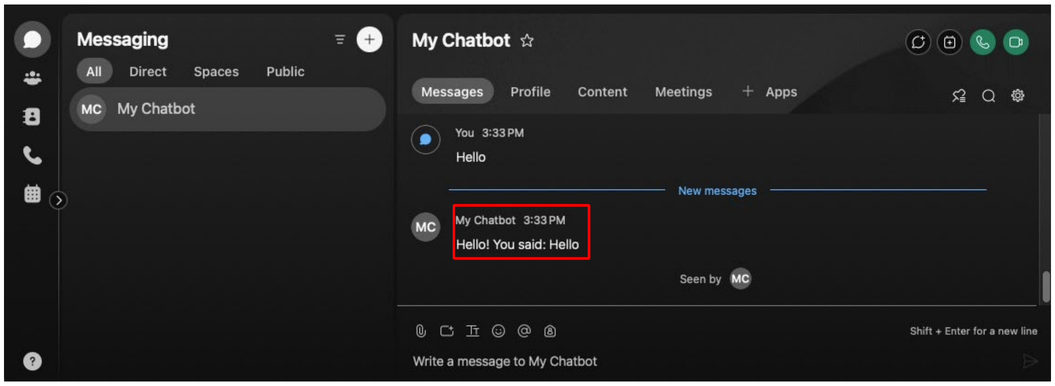
Run


200 / OK

```
{
  "id": "Y2lzY29zcGFyazovL3VybjpURUFNOnVzLWdvdi13ZXN0LT
  "name": "My Chat Bot",
  "targetUrl": "https://3466-2600-4040-9055-9100-eca8-e
  "resource": "messages",
  "event": "created",
  "orgId": "Y2lzY29zcGFyazovL3VzL09SR0FOSVpBVElPTi8zN2E
  "createdBy": "Y2lzY29zcGFyazovL3VzL1BFT1BMRS9jOGE0ZjE
  "appId": "Y2lzY29zcGFyazovL3VzL0FQUExJQ0FUSU9OL0NmMzk
  "ownedBy": "creator",
  "status": "active",
  "created": "2023-11-18T02:10:43.779Z"
}
```

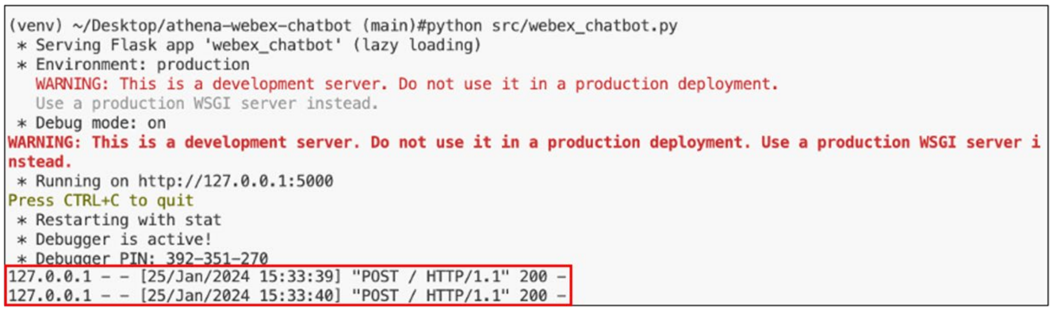
Step 14: It's now time to test the bot in the Webex Teams environment. After opening Webex Teams, initiate a conversation with your bot using its username (email address). When you send a message to your bot, the webhook you set up will notify your Flask server, and you should see the bot's response ("Hello! You said: [your last message]") in the chat window. Keep an eye on the Flask server's terminal output for incoming requests or errors; this will help you monitor the bot's interaction with users in real-time.




Messaging
All Direct Spaces Public
MC My Chatbot
My Chatbot
Messages Profile Content Meetings + Apps
New messages
Seen by MC
Shift + Enter for a new line
Write a message to My Chatbot

You 3:33 PM
Hello

My Chatbot 3:33 PM
Hello! You said: Hello



```
(venv) ~/Desktop/athena-webex-chatbot (main)#python src/webex_chatbot.py
 * Serving Flask app 'webex_chatbot' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server i
nstead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 392-351-270
127.0.0.1 - - [25/Jan/2024 15:33:39] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [25/Jan/2024 15:33:40] "POST / HTTP/1.1" 200 -
```

Activity Verification

You have completed this task when you attain these results:

- You message your bot in Webex chat and receive a response from the bot

Debugging

If your bot doesn't respond as expected, it's time to debug. First, check the Flask application's terminal output for any error messages that could indicate what went wrong. Confirm that your Ngrok session is still running and that the webhook's `targetUrl` matches the Ngrok URL in your terminal window. Finally, verify the `.env` file contains the correct access token and bot email. These checks ensure that your bot is correctly configured to communicate with the Webex servers.