

# AWS Lambda Hands-on: How to Create Phone Call Notifications in a Serverless Way

In this blog I will walk you through the setup of creating a call notification system using Python and Serverless. Let's get started!

## Prerequisites:

Make sure the following software/modules are installed:

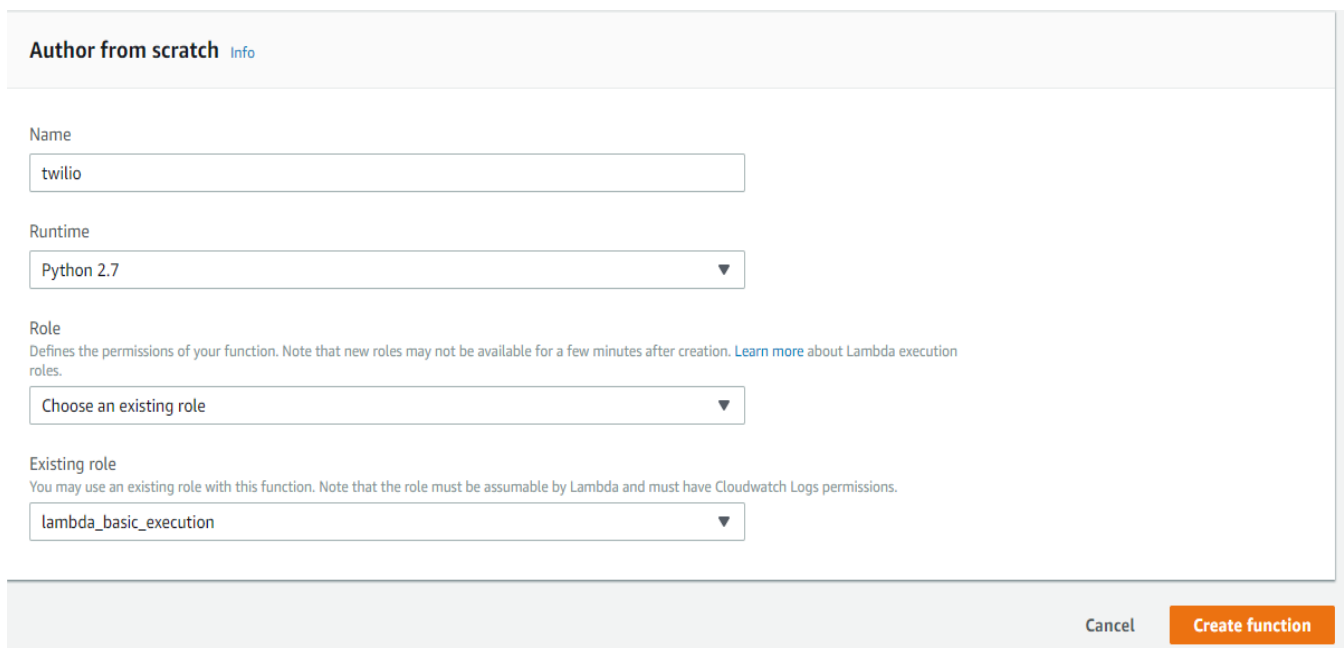
1. Python 2.7 (because we are going to write the core logic in Python)
2. Twilio module (Twilio is for making automated calls) - **pip install twilio**

## Step 1: Clone/download from the GitHub repo

You can clone/download the following GitHub repo: [https://github.com/SrushithR/twilio\\_automated\\_calls/](https://github.com/SrushithR/twilio_automated_calls/)

## Step 2: Create an AWS Lambda function

Sign In/Sign Up to your AWS account and navigate to Lambda (<https://console.aws.amazon.com/lambda/home>). Choose "Create function" button and fill the details as shown:



**Author from scratch** [Info](#)

Name

Runtime

Role  
Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. [Learn more](#) about Lambda execution roles.

Existing role  
You may use an existing role with this function. Note that the role must be assumable by Lambda and must have Cloudwatch Logs permissions.

[Cancel](#) [Create function](#)

Select 'Choose an existing role' under 'Role' and choose the default role available. If you don't have this option, select 'Create new role from template(s)' and enter a role name – lambda\_role and click on 'Create function'. (AWS Lambda attaches the necessary permissions (permission to write logs to CloudWatch) automatically).

Note: You can go to the IAM console (<https://console.aws.amazon.com/iam/home#/roles>) and view/update/modify the automatically attached policies.

Runtime

Python 2.7 ▼

Role

Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. [Learn more](#) about Lambda execution roles.

Create new role from template(s) ▼

Lambda will automatically create a role with permissions from the selected policy templates. Note that basic Lambda permissions (logging to CloudWatch) will automatically be added. If your function accesses a VPC, the required permissions will also be added.

Role name

Enter a name for your new role.

lambda\_role

**i** This new role will be scoped to the current function. To use it with other functions, you can modify it in the IAM console.

Policy templates

Choose one or more policy templates. A role will be generated for you before your function is created. [Learn more](#) about the permissions that each policy template will add to your role.

▼

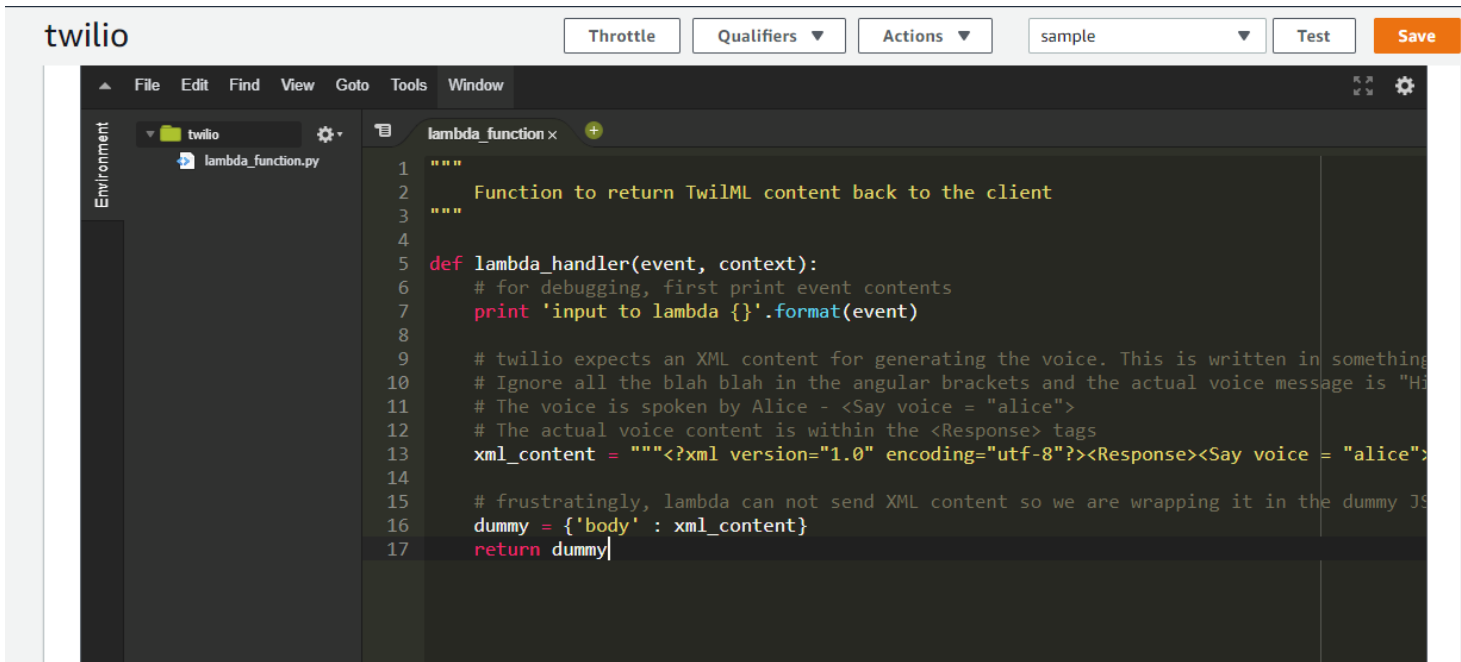
Cancel Create function

Paste the code from the "lambda\_function.py" file (or as shown below) into the online editor and save it.

```

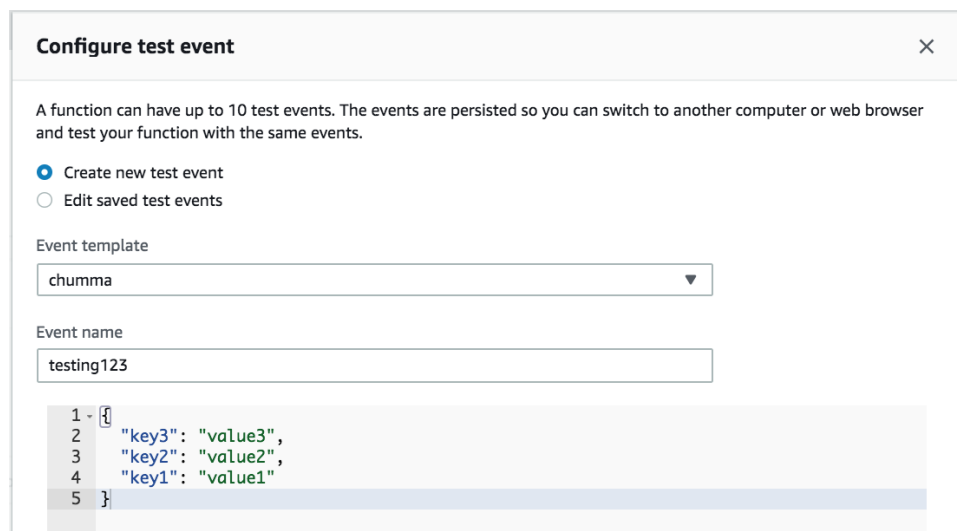
"""
Function to return TwiML content back to the client
"""
def lambda_handler(event, context):
# for debugging, first print event contents
print 'input to lambda {}'.format(event)
# twilio expects an XML content for generating the voice. This is written in
something strange called TwiML
# Ignore all the blah blah in the angular brackets and the actual voice message is
"Hi! ....".
# The voice is spoken by Alice - <Say voice = "alice">
# The actual voice content is within the <Response> tags
xml_content = "<?xml version='1.0' encoding='utf-8'?><Response><Say voice =
"alice">Hi! Hope you are having a great time hacking code in the
meetup</Say></Response>"
# frustratingly, lambda can not send XML content so we are wrapping it in the dummy
JSON variable
dummy = {'body' : xml_content}
return dummy

```



Now, let's test the lambda function with a sample event. Serverless functions are event driven and work on the inputs provided. Even though our application code is independent of the input, we have to provide one (sample test event) for testing in the AWS Lambda console.

Click on the “Test” button and the following window would pop up. Provide an event name and click on create and click on “Test” button again.



Hurray! It succeeded - here is the test results:

Lambda > Functions > twilio ARN - arn:aws:lambda:us-east-1:431635030606:function:twilio

twilio Throttle Qualifiers ▼ Actions ▼ chumma ▼ Test Save

✓ Execution result: succeeded (logs) ✕

▼ Details

The area below shows the result returned by your function execution.

```
{
  "body": "<?xml version='1.0' encoding='utf-8'?'><Response><Say voice = 'alice'>Hi! Hope you are having a great time hacking code in the meetup</Say></Response>"
}
```

**Summary**

<b>Code SHA-256</b>	8i2LVAESwHAdtnlaDI5+YqOrHYMeV542Im984/6bsXs=	<b>Request ID</b>	28711848-600a-11e8-b276-db9c71ef2555
<b>Duration</b>	19.32 ms	<b>Billed duration</b>	100 ms
<b>Resources configured</b>	128 MB	<b>Max memory used</b>	19 MB

**Log output**

The area below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. [Click here](#) to view the CloudWatch log group.

```
START RequestId: 28711848-600a-11e8-b276-db9c71ef2555 Version: $LATEST
input to lambda {u'key3': u'value3', u'key2': u'value2', u'key1': u'value1'}
END RequestId: 28711848-600a-11e8-b276-db9c71ef2555
REPORT RequestId: 28711848-600a-11e8-b276-db9c71ef2555 Duration: 19.32 ms Billed Duration: 100 ms Memory Size: 128 MB Max Memory Used: 19 MB
```

### Step 3: Create an API and resource on API Gateway

Navigate to API Gateway (<https://console.aws.amazon.com/apigateway/home>) in the services section and create an API as shown below:

APIs > Create

## Create new API

In Amazon API Gateway, an API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.

☒ New API
 ☐ Clone from existing API
 ☐ Import from Swagger
 ☐ Example API

### Settings

Choose a friendly name and description for your API.

<b>API name*</b>	<input type="text" value="NotificationService"/>
<b>Description</b>	<input type="text" value="API for notification services"/>
<b>Endpoint Type</b>	<input type="text" value="Regional"/>

Under the 'Actions' button, select 'Create Resource' and fill the details as shown and create a resource:

## New Child Resource

Use this page to create a new child resource for your resource. 

Configure as [proxy resource](#) ☐ 

Resource Name\*

twilio

Resource Path\*

/ twilio

You can add path parameters using brackets. For example, the resource path `{username}` represents a path parameter called 'username'. Configuring `/ {proxy+}` as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to `/foo`. To handle requests to `/`, add a new ANY method on the `/` resource.

Enable API Gateway CORS ☐ 

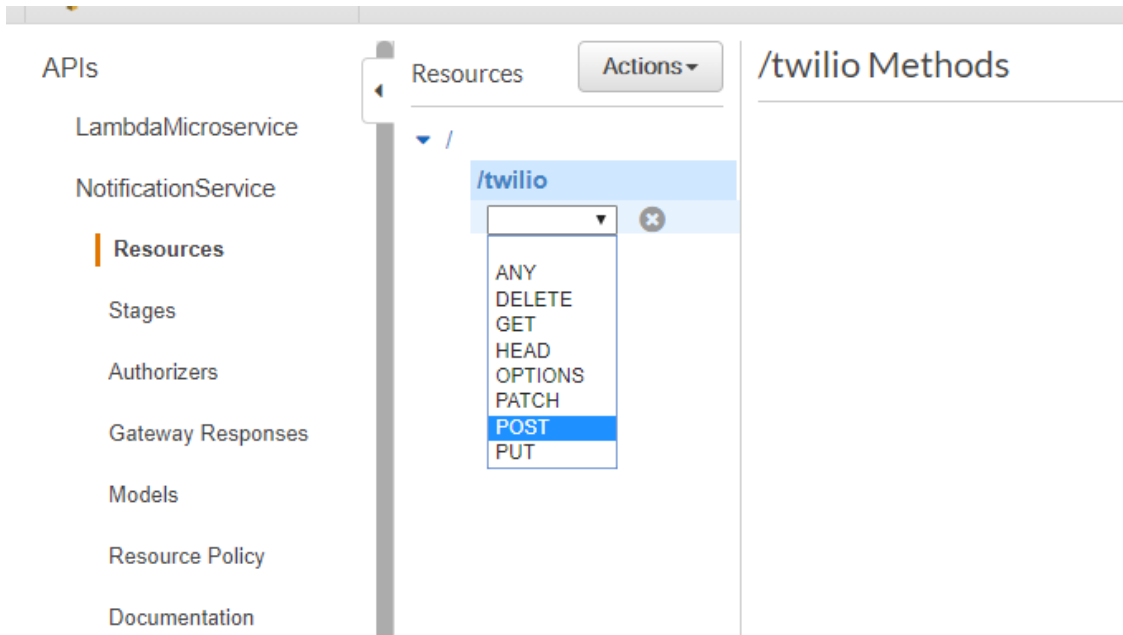
\* Required

Cancel

Create Resource

### Step 4: Create a method

Create a POST method under the resource (that we created in the previous step - twilio)



The screenshot shows the AWS API Gateway console. On the left, a sidebar lists navigation options: APIs, LambdaMicroservice, NotificationService, Resources (selected), Stages, Authorizers, Gateway Responses, Models, Resource Policy, and Documentation. The main area is titled 'Resources' and shows a tree view with a root '/' and a child resource '/twilio'. The '/twilio' resource is selected, and a dropdown menu is open below it, listing available HTTP methods: ANY, DELETE, GET, HEAD, OPTIONS, PATCH, POST (highlighted), and PUT. To the right of the resource tree, the title '/twilio Methods' is visible, indicating the next step in the configuration process.

Select the lambda function that was created in step 2

## /twilio - POST - Setup



Choose the integration point for your new method.

Integration type ☒ Lambda Function ⓘ

☐ HTTP ⓘ

☐ Mock ⓘ

☐ AWS Service ⓘ

☐ VPC Link ⓘ

Use Lambda Proxy integration ☐ ⓘ

Lambda Region

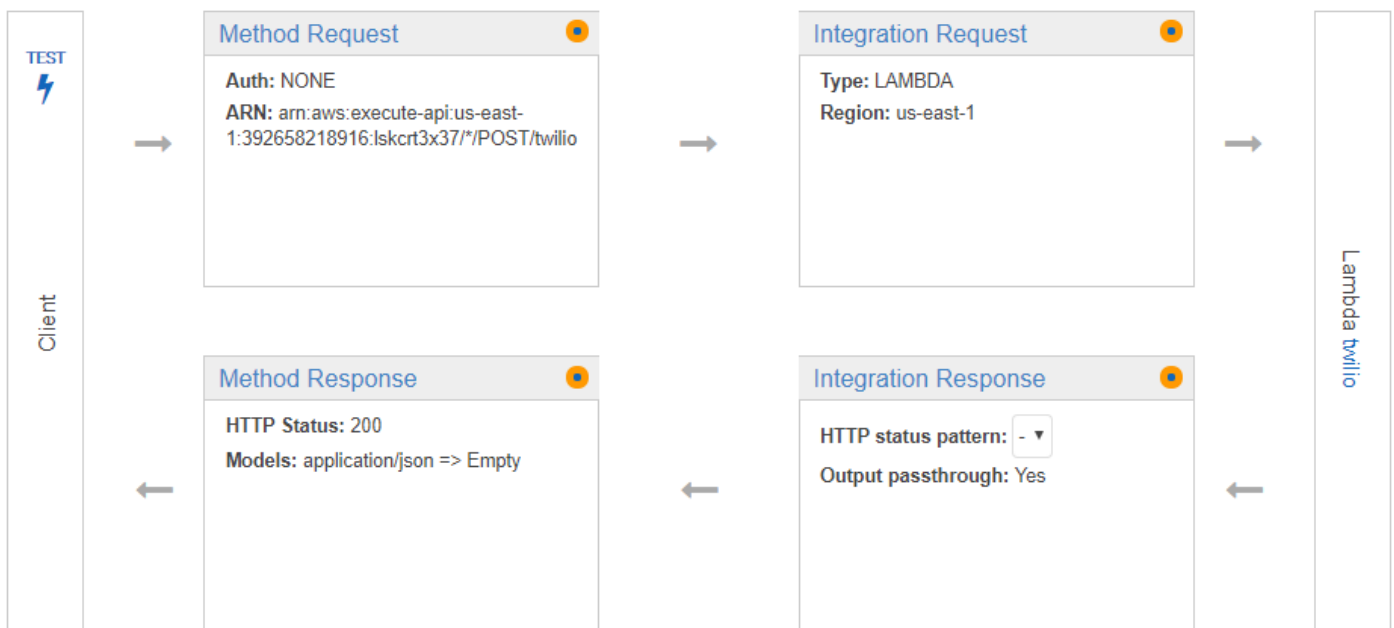
Lambda Function  ⓘ

Use Default Timeout ☒ ⓘ

Save

And once saved, the “Method Execution” page would like this:

## /twilio - POST - Method Execution



Follow these steps for configuring up the API Gateway:

4a. Click in “Integration Request” and scroll down to “Body Mapping Templates” and add “**application/x-www-form-urlencoded**” as the mapping template and provide the value as follows:

```
{
  "reqbody": "$input.path('$')
}
```

Content-Type

application/x-www-form-urlencoded

+ Add mapping template

application/x-www-form-urlencoded

Generate template: ▼

```

1  |
2  | "reqbody":"${input.path('$')}"
3  |

```

Save and navigate back to the “Method Execution” (click on **"use current settings"** when a message is popped up)

4b. Click the “Integration Response” and move to the “Body Mapping Templates” by clicking the triangular arrow. Edit the “application/json” Content-Type to “application/xml” with a template as follows and save it:

`${input.path('$').body}`

Body Mapping Templates

Content-Type

application/xml

+ Add mapping template

Generate template: ▼

```

1  | ${input.path('$').body}

```

Cancel Save

4c. Move back to the “Method Execution” and in the “Method Response”, edit “application/json” to “application/xml” in the “Response Body for 200” section and select “Empty” under the Models as shown below:

← Method Execution /twilio - POST - Method Response

Provide information about this method's response types, their headers and content types.

HTTP Status
<div> <div>▼</div> <div>200</div> <div> </div> </div>

Response Headers for 200

Name
No headers

Add Header

Response Body for 200

Content type	Models
application/xml	Empty

Add Response Model

Add Response

Hit the 'Test' button in the “Method Execution” page and test the method (Leave the “Request Body” empty)The response would look like this:

Request: /twilio

Status: 200

Latency: 43 ms

Response Body

```
<?xml version="1.0" encoding="utf-8"?><Response><Say voice = "alice">Hi! Hope you are having a great time hacking code in the meetup</Say></Response>
```

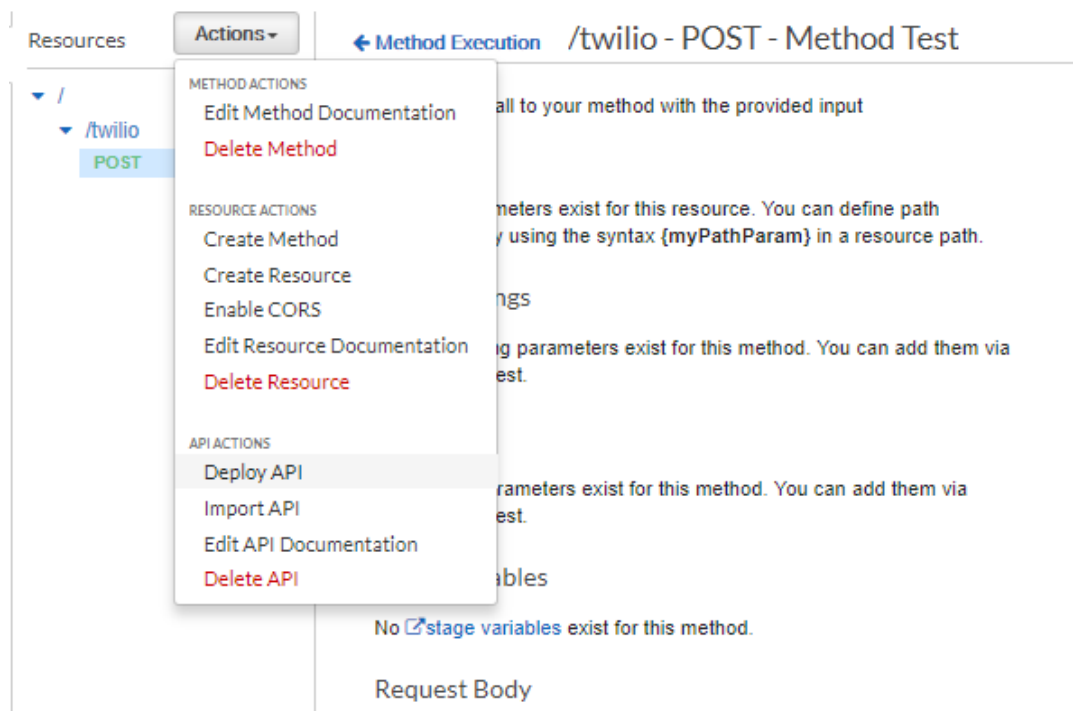
Response Headers

```
{"X-Amzn-Trace-Id": "Root=1-5b07ba73-24fe4c9844c7deb0ad82e592", "Content-Type": "application/xml"}
```

## Step 5: Deploy the API

After a successful test, it's time you deploy the API. Select the POST method and click on “Deploy API” under the Actions tab





Add the following details in the deployment pop-up and deploy:

Deploy API

Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.

Deployment stage

[New Stage]

Stage name\*

dev

Stage description

Development stage

Deployment description

Deploying Twilio API

Cancel

Deploy

Once deployed, navigate to your method under the 'dev' section and copy the 'Invoke URL'

Stages

Create

dev

/

/twilio

POST

dev - POST - /twilio

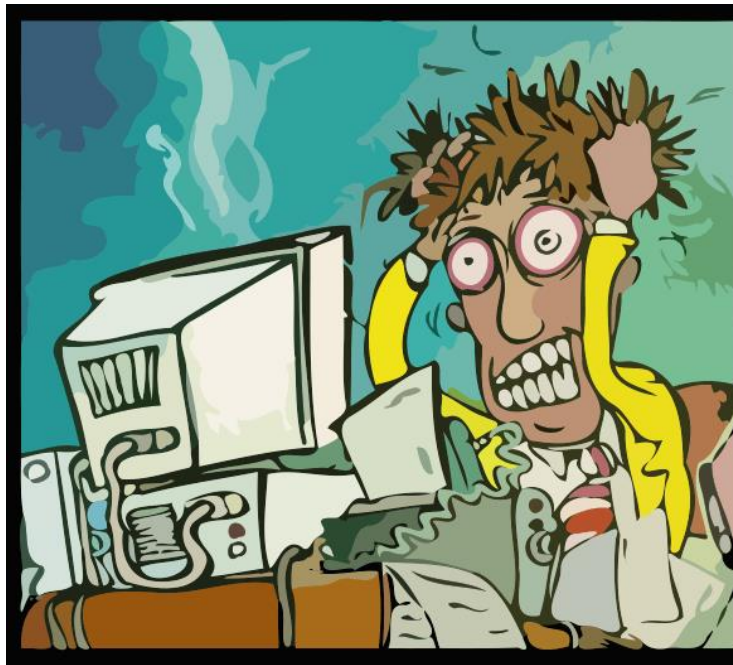
Invoke URL: <https://lskrt3x37.execute-api.us-east-1.amazonaws.com/dev/twilio>

Use this page to override the [dev stage](#) settings for the POST to /twilio method.

Settings ☒ Inherit from stage ☐ Override for this method

Save Changes

**Step 6: Take a deep breath! We are done with the AWS part now....**  
**Take another deep breath, we are going to start with Twilio**



### Step 7: Sign up on Twilio

Twilio is a cloud communications platform as a service company and offers an array of services. We will be using the Twilio programmable voice service.

*Step 1a:* Sign up for free on Twilio at <https://www.twilio.com/try-twilio>.

*Step 1b:* Create a new project by just giving a project name

*Step 1c:* Get started with the Twilio Programmable Voice

### Step 2: Get your first Twilio Number

Click on "Get a Number"

miketesting trial Voice / Learn & Build / Build / UPGRADE Go to...

**Programmable Voice**

Build With Programmable Voice

Choose a use case to build a production ready solution.

Basics

[You have a Trial Account >](#) [Show API Credentials >](#)

- 1. Get a Number**  
In order to make calls or send messages through the Twilio API, you need to get a Twilio phone number.  
[Get your first Twilio number](#)
- 2. Confirm Your Programming Language**
- 3. Make and receive voice calls to your Twilio phone number**  
Use the diagrams below to understand basic inbound and outbound call architectures, then complete the quick start guide to make and receive your first phone calls.

And click on "Choose this Number" (only if you like the number ;)). You can also search for a different number

**Your first Twilio Phone Number**

(254) 237-5652 [Don't like this one? Search for a different number](#)

This United States phone number has the following capabilities:

- Voice:** This number can receive incoming calls and make outgoing calls.
- SMS:** This number can send and receive text messages to and from mobile numbers.
- MMS:** This number can send and receive multi media messages to and from mobile numbers.

[Cancel](#) [Choose this Number](#)

Note: Each number costs \$1 per month. You don't have to pay anything for using it in this demo though! Congrats! Now you have a number to play with.

### Step 3: Make an outbound call

Before you can make an outbound call, you need the Twilio credentials for authorization in your python code. They can be found at the Twilio console dashboard (<https://www.twilio.com/user/account>) - note down the account SID and the Auth token.

"make\_phone\_call.py"

([https://raw.githubusercontent.com/SrushithR/twilio\\_automated\\_calls/master/make\\_phone\\_call.py](https://raw.githubusercontent.com/SrushithR/twilio_automated_calls/master/make_phone_call.py)) is the python code to make an outgoing call. Here is the same code:

```
# Download the Python helper library from twilio.com/docs/python/install
from twilio.rest import Client

# Your Account Sid and Auth Token from twilio.com/user/account
account_sid = "AC5*****811d8e4d445a"
auth_token = "ef65*****aeb3"
client = Client(account_sid, auth_token)

call = client.calls.create(
    # the verified number to which you wanna call
    to="+918686519259",
    # the number that you just purchased on Twilio
    from_="+18034087781",
    # the invoke URL from API Gateway
    url="https://lskcr****.execute-api.us-east-1.amazonaws.com/dev/twilio"
)
# An SID is generated for every call. It is useful for debugging
print(call.sid)
```

Before you run the above snippet,

- Replace the account\_sid and auth\_token with your credentials
- to - The number you want to call
- from - The number you just bought. Since, we are still in the trial account (can be upgraded by adding credit/debit card information), you must verify the 'to' number under the 'Verified Caller IDs' (<https://www.twilio.com/console/phone-numbers/verified>). Give your number for testing purposes
- URL - A URL that returns TwiML (Twilio Markup Language) with instructions on what should happen when the user picks up the call. In our case it is the 'Invoke URL' from the previous step

Once all the details are updated in the code, run it:

```
python make_phone_call.py
```

The above file will output a SID, which you can use for debugging in the Twilio console (<https://www.twilio.com/console/runtime/debugger>)

#### Reference URLs:

1. Twilio Docs: <https://www.twilio.com/docs/voice/tutorials/how-to-make-outbound-phone-calls-python>
2. <https://www.twilio.com/blog/2015/09/build-your-own-ivr-with-aws-lambda-amazon-api-gateway-and-twilio.html>