

AWS Lab on Lambda Functions (Building Server less web applications)

Python Server less Microframework for AWS

The python server less microframework for AWS allows you to quickly create and deploy applications that use Amazon API Gateway and AWS Lambda. It provides:

- A command line tool for creating, deploying, and managing your app
- A familiar and easy to use API for declaring views in python code
- Automatic IAM policy generation

Installing Chalice → `pip install chalice`

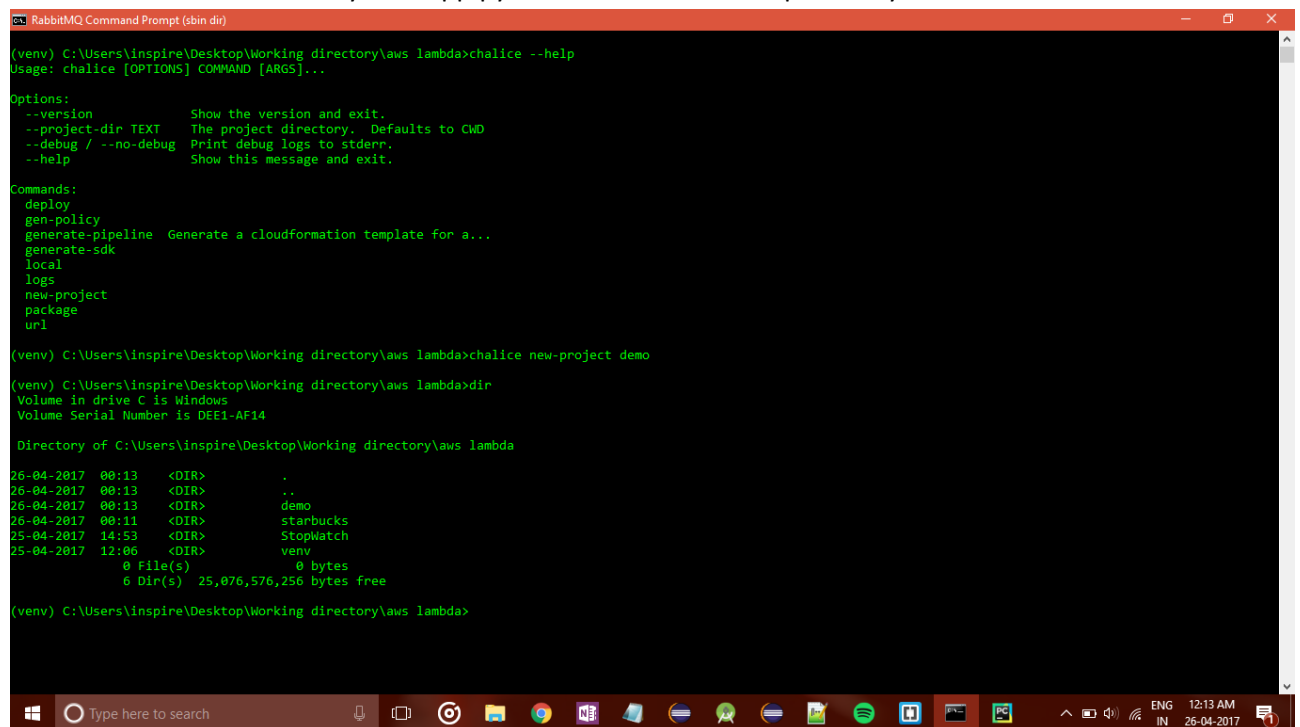
Steps to deploy a simple application with custom Response and hello world

Create a directory for chalice and make a virtual environment for python and run chalice inside it.

1. Create new project

`chalice new-project 'name '`

This will create new directory with app.py file and some other dependency resource files



```
RabbitMQ Command Prompt (sbin dir)

(venv) C:\Users\inspire\Desktop\Working directory\aws lambda>chalice --help
Usage: chalice [OPTIONS] COMMAND [ARGS]...

Options:
  --version            Show the version and exit.
  --project-dir TEXT  The project directory. Defaults to CWD
  --debug / --no-debug Print debug logs to stderr.
  --help              Show this message and exit.

Commands:
  deploy
  gen-policy
  generate-pipeline  Generate a cloudformation template for a...
  generate-sdk
  local
  logs
  new-project
  package
  url

(venv) C:\Users\inspire\Desktop\Working directory\aws lambda>chalice new-project demo

(venv) C:\Users\inspire\Desktop\Working directory\aws lambda>dir
Volume in drive C is Windows
Volume Serial Number is DEE1-AF14

Directory of C:\Users\inspire\Desktop\Working directory\aws lambda

26-04-2017  00:13  <DIR>      .
26-04-2017  00:13  <DIR>      ..
26-04-2017  00:13  <DIR>      demo
26-04-2017  00:11  <DIR>      starbucks
25-04-2017  14:53  <DIR>      StopWatch
25-04-2017  12:06  <DIR>      venv
               0 File(s)              0 bytes
               6 Dir(s)  25,076,576,256 bytes free

(venv) C:\Users\inspire\Desktop\Working directory\aws lambda>
```

You can type `chalice --help` to get list of available commands

Creating new project with chalice

```
RabbitMQ Command Prompt (sbin dir)
gen-policy
generate-pipeline Generate a cloudformation template for a...
generate-sdk
local
logs
new-project
package
url

(venv) C:\Users\inspire\Desktop\Working directory\aws lambda>chalice new-project demo

(venv) C:\Users\inspire\Desktop\Working directory\aws lambda>dir
Volume in drive C is Windows
Volume Serial Number is DEE1-AF14

Directory of C:\Users\inspire\Desktop\Working directory\aws lambda

26-04-2017  00:13    <DIR>          .
26-04-2017  00:13    <DIR>          ..
26-04-2017  00:13    <DIR>          demo
26-04-2017  00:11    <DIR>          starbucks
25-04-2017  14:53    <DIR>          StopWatch
25-04-2017  12:06    <DIR>          venv
                0 File(s)            0 bytes
                6 Dir(s)  25,076,576,256 bytes free

(venv) C:\Users\inspire\Desktop\Working directory\aws lambda>cd demo

(venv) C:\Users\inspire\Desktop\Working directory\aws lambda>demo>dir
Volume in drive C is Windows
Volume Serial Number is DEE1-AF14

Directory of C:\Users\inspire\Desktop\Working directory\aws lambda>demo

26-04-2017  00:13    <DIR>          .
26-04-2017  00:13    <DIR>          ..
26-04-2017  00:13    <DIR>          .chalice
26-04-2017  00:13    39 .gitignore
26-04-2017  00:13    816 app.py
26-04-2017  00:13    0 requirements.txt
                3 File(s)            855 bytes
                3 Dir(s)  25,076,436,992 bytes free

(venv) C:\Users\inspire\Desktop\Working directory\aws lambda>demo>
```

2. Writing app.py

By default, app.py will look something like this.

```
app.py (-\Desktop\Working directory\aws lambda\demo) - VIM
from chalice import Chalice

app = Chalice(app_name='demo')

@app.route('/')
def index():
    return {'hello': 'world'}

# The view function above will return {"hello": "world"}
# whenever you make an HTTP GET request to '/'.
#
# Here are a few more examples:
#
# @app.route('/hello/{name}')
# def hello_name(name):
#     # '/hello/james' -> {"hello": "james"}
#     return {'hello': name}
#
# @app.route('/users', methods=['POST'])
# def create_user():
#     # This is the JSON body the user sent in their POST request.
#     user_as_json = app.json_body
#     # Suppose we had some 'db' object that we used to
#     # read/write from our database.
#     # user_id = db.create_user(user_as_json)
#     return {'user_id': user_id}
#
# See the README documentation for more examples.
```

We are going to write a route that returns custom response as follows:

```
body='hello world!',
status_code=200,
headers={'Content-Type': 'text/plain'})
```

Here is a app.py file with route which will return custom response as specified.

```
app.py (-\Desktop\Working directory\aws lambda\demo) - VIM
from chalice import Chalice, Response

app = Chalice(app_name='demo')

app.route('/')
def main():
    return {"Hello": "World"}

app.route('/customResponse')
def index():
    return Response(body='hello world!',
                    status_code=200,
                    headers={'Content-Type': 'text/plain'})
```

3. Deploy

Run chalice deploy

This will package your application and respective dependencies and deploy it by using lambda and API gateway.

```
RabbitMQ Command Prompt (sbin dir)
Initial creation of lambda function.
Creating role
Creating deployment package.
Initiating first time deployment...
Deploying to: dev
https://y6d8eb816k.execute-api.us-west-1.amazonaws.com/dev/

(venv) C:\Users\inspire\Desktop\Working directory\aws lambda\demo>vim app.py

(venv) C:\Users\inspire\Desktop\Working directory\aws lambda\demo>chalice deploy
Updating IAM policy.
Updating lambda function...
Regen deployment package...
Sending changes to lambda.
API Gateway rest API already found.
Deploying to: dev
https://y6d8eb816k.execute-api.us-west-1.amazonaws.com/dev/

(venv) C:\Users\inspire\Desktop\Working directory\aws lambda\demo>vim app.py

(venv) C:\Users\inspire\Desktop\Working directory\aws lambda\demo>vim app.py

(venv) C:\Users\inspire\Desktop\Working directory\aws lambda\demo>chalice deploy
Unable to import your app.py file:
File "C:\Users\inspire\Desktop\Working directory\aws lambda\demo\app.py", line 1
    from chalice import Chalice, Response
SyntaxError: invalid syntax

(venv) C:\Users\inspire\Desktop\Working directory\aws lambda\demo>vim app.py

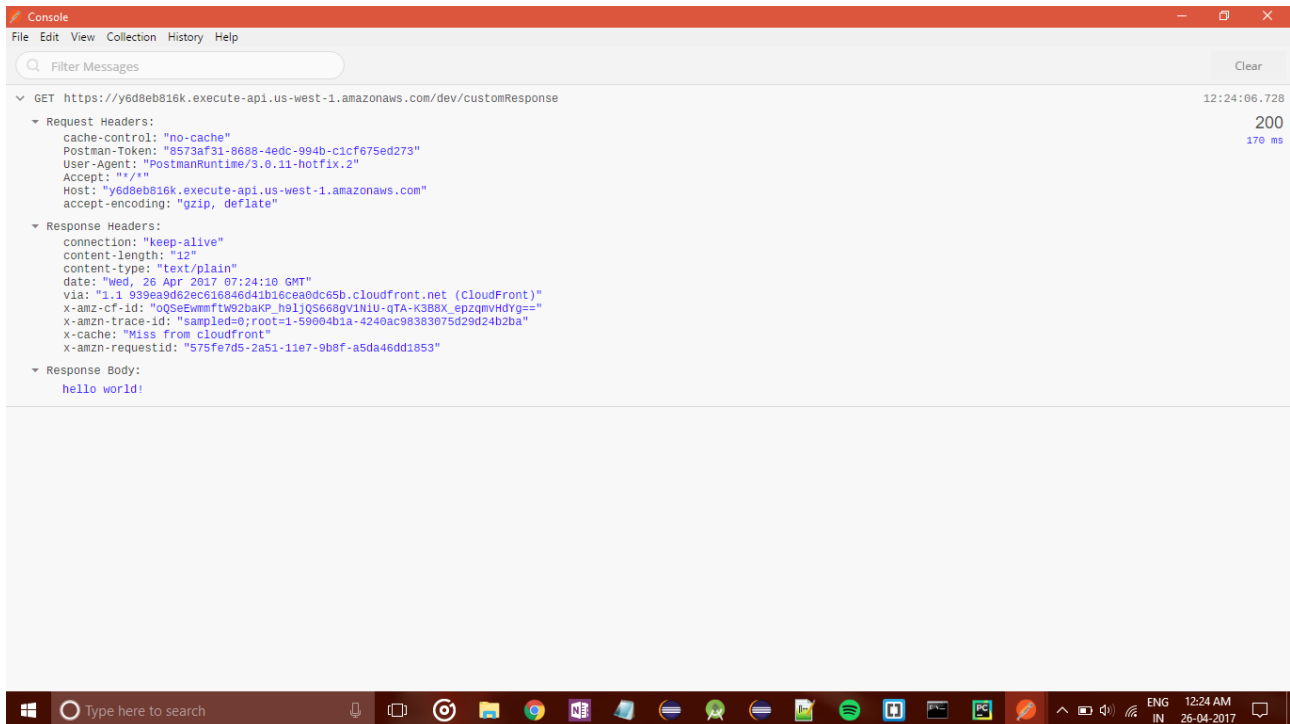
(venv) C:\Users\inspire\Desktop\Working directory\aws lambda\demo>vim app.py

(venv) C:\Users\inspire\Desktop\Working directory\aws lambda\demo>chalice deploy
Updating IAM policy.
Updating lambda function...
Regen deployment package...
Sending changes to lambda.
API Gateway rest API already found.
Deploying to: dev
https://y6d8eb816k.execute-api.us-west-1.amazonaws.com/dev/

(venv) C:\Users\inspire\Desktop\Working directory\aws lambda\demo>
```

Final Output

Hello World



Customized http response.

- Specify the status code to return
- Specify custom headers to add to the response
- Specify response bodies that are not application/json

