

CSEN 241- Cloud Computing

Homework 2

Your Own Serverless Infrastructure

Submitted by:

Aditya Shrivastava

W1648524

Github link for HW2:

https://github.com/AdityaShrivastava30/CSEN241_Cloud_Computing/tree/master/Homeworks/HW2

Step 8:

- 1. Provide a screenshot of invoking the figlet function (5 pts)**

[illegible]

2. Provide a screenshot of running the following command (5 pts)
- ```
sudo journalctl -u faasd --lines 40
```

```

ubuntu@faasd:~$ sudo journalctl -u faasd --lines 40
Feb 22 22:15:55 faasd faasd[4869]: Removing old container for: nats
Feb 22 22:15:55 faasd faasd[4869]: Removing old container for: prometheus
Feb 22 22:15:55 faasd faasd[4869]: 2024/02/22 22:15:55 Start-up order:
Feb 22 22:15:55 faasd faasd[4869]: 2024/02/22 22:15:55 - nats
Feb 22 22:15:55 faasd faasd[4869]: 2024/02/22 22:15:55 - prometheus
Feb 22 22:15:55 faasd faasd[4869]: 2024/02/22 22:15:55 - gateway
Feb 22 22:15:55 faasd faasd[4869]: 2024/02/22 22:15:55 - queue-worker
Feb 22 22:15:55 faasd faasd[4869]: Starting: nats
Feb 22 22:15:55 faasd faasd[4869]: Creating local directory: /var/lib/faasd/nats
Feb 22 22:15:55 faasd faasd[4869]: 2024/02/22 22:15:55 Running nats with user: "65534"
Feb 22 22:15:55 faasd faasd[4869]: 2024/02/22 22:15:55 Created container: nats
Feb 22 22:15:55 faasd faasd[4869]: 2024/02/22 22:15:55 nats has IP: 10.62.0.2
Feb 22 22:15:55 faasd faasd[4869]: 2024/02/22 22:15:55 Task: nats Container: nats
Feb 22 22:15:55 faasd faasd[4869]: Starting: prometheus
Feb 22 22:15:55 faasd faasd[4869]: Creating local directory: /var/lib/faasd/prometheus
Feb 22 22:15:55 faasd faasd[4869]: 2024/02/22 22:15:55 Running prometheus with user: "65534"
Feb 22 22:15:55 faasd faasd[4869]: 2024/02/22 22:15:55 Created container: prometheus
Feb 22 22:15:56 faasd faasd[4869]: 2024/02/22 22:15:56 prometheus has IP: 10.62.0.3
Feb 22 22:15:56 faasd faasd[4869]: 2024/02/22 22:15:56 Task: prometheus Container: prometheus
Feb 22 22:15:56 faasd faasd[4869]: Starting: gateway
Feb 22 22:15:56 faasd faasd[4869]: 2024/02/22 22:15:56 Created container: gateway
Feb 22 22:15:56 faasd faasd[4869]: 2024/02/22 22:15:56 gateway has IP: 10.62.0.4
Feb 22 22:15:56 faasd faasd[4869]: 2024/02/22 22:15:56 Task: gateway Container: gateway
Feb 22 22:15:56 faasd faasd[4869]: Starting: queue-worker
Feb 22 22:15:56 faasd faasd[4869]: 2024/02/22 22:15:56 Created container: queue-worker
Feb 22 22:15:56 faasd faasd[4869]: 2024/02/22 22:15:56 queue-worker has IP: 10.62.0.5
Feb 22 22:15:56 faasd faasd[4869]: 2024/02/22 22:15:56 Task: queue-worker Container: queue-worker
Feb 22 22:15:56 faasd faasd[4869]: 2024/02/22 22:15:56 Supervisor init done in: 12 seconds
Feb 22 22:15:56 faasd faasd[4869]: 2024/02/22 22:15:56 Looking up IP for: "prometheus"
Feb 22 22:15:56 faasd faasd[4869]: 2024/02/22 22:15:56 Resolver rebuilding map
Feb 22 22:15:56 faasd faasd[4869]: 2024/02/22 22:15:56 Resolver: "localhost"="127.0.0.1"
Feb 22 22:15:56 faasd faasd[4869]: 2024/02/22 22:15:56 Resolver: "faasd-provider"="10.62.0.1"
Feb 22 22:15:56 faasd faasd[4869]: 2024/02/22 22:15:56 Resolver: "nats"="10.62.0.2"
Feb 22 22:15:56 faasd faasd[4869]: 2024/02/22 22:15:56 Resolver: "prometheus"="10.62.0.3"
Feb 22 22:15:56 faasd faasd[4869]: 2024/02/22 22:15:56 Resolver: "gateway"="10.62.0.4"
Feb 22 22:15:56 faasd faasd[4869]: 2024/02/22 22:15:56 Resolver: "queue-worker"="10.62.0.5"
Feb 22 22:15:56 faasd faasd[4869]: 2024/02/22 22:15:56 Looking up IP for: "gateway"
Feb 22 22:15:56 faasd faasd[4869]: 2024/02/22 22:15:56 Proxy from: 0.0.0.0:8080, to: gateway:8080 (10.62.0.4)
Feb 22 22:15:56 faasd faasd[4869]: 2024/02/22 22:15:56 faasd: waiting for SIGTERM or SIGINT
Feb 22 22:15:56 faasd faasd[4869]: 2024/02/22 22:15:56 Proxy from: 127.0.0.1:9090, to: prometheus:9090 (10.62.0.3)
ubuntu@faasd:~$

```

### 3. Complete slack-request/handler.py (10 pts)

➔ Attached is the screenshot for the complete code for slack-request/handle.py

```

import json

def handle(req):
 data = {
 "text": "Serverless Message",
 "attachments": [
 {
 "title": "The Awesome world of Cloud Computing! COEN 241",
 "fields": [
 {
 "title": "Amazing Level",
 "value": "100",
 "short": True
 }
],
 "author_name": "Aditya Shrivastava",
 "author_icon": "https://github.com/AdityaShrivastava30.png",
 "image_url": "https://github.com/AdityaShrivastava30.png"
 },
 {
 "title": "About COEN 241",
 "text": "COEN 241 is the most awesome class ever!."
 },
 {
 "fallback": "Would you recommend COEN 241 to your friends?",
 "title": "Would you recommend COEN 241 to your friends?",
 "callback_id": "response123",
 "color": "#3AA3E3",
 "attachment_type": "default",
 "actions": [
 {
 "name": "recommend",
 "text": "Of Course!",
 "type": "button",
 "value": "recommend"
 },
 {
 "name": "definitely",
 "text": "Definitely",
 "type": "button",
 "value": "definitely"
 }
]
 }
]
 }
 return json.dumps(data)

```

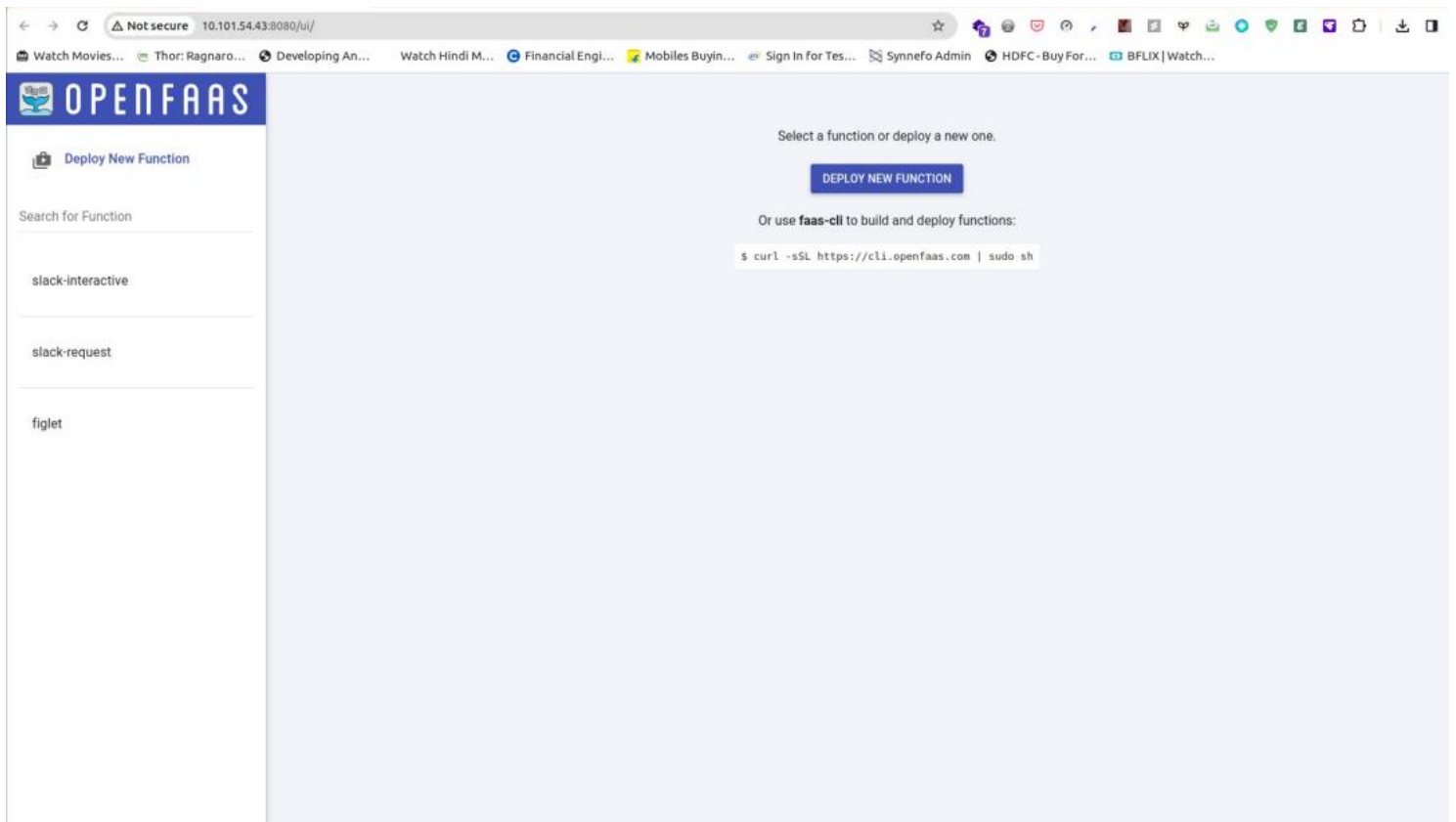
#### 4. Complete slack-interactive/handler.py (10 pts)

➔ Attached is the screenshot for the complete code for slack-interactive/handle.py

```
import json
import urllib.parse

def handle(req):
 urlstring = urllib.parse.unquote(req).strip('payload=')
 response = json.loads(urlstring)
 data = {
 "attachments": [
 {
 "replace_original": True,
 "response_type": "ephemeral",
 "fallback": "Required plain-text summary of the attachment.",
 "color": "#36a64f",
 "pretext": "Ahh yeah! Great choice, COEN 241 is absolutely amazing!",
 "author_name": "",
 "title": "COEN 241",
 "title_link": "https://www.scu.edu/engineering/academic-programs/department-of-computer-engineering/graduate/course-descriptions/",
 "text": "Head over to COEN 241",
 "image_url": "https://www.scu.edu/media/offices/umc/scu-brand-guidelines/visual-identity-and-photography/visual-identity-toolkit/logos-and-seals/Mission-Dont3.png",
 "thumb_url": "https://www.scu.edu/engineering/academic-programs/department-of-computer-engineering/graduate/course-descriptions/",
 "footer": "Slack Apps built on OpenFaas",
 "footer_icon": "https://a.slack-edge.com/45901/marketing/img/_rebrand/meta/slack_hash_256.png",
 "ts": 123456789
 }
]
 }
 return json.dumps(data)
```

#### 5. Provide a screenshot of your OpenFaaS gateway AFTER deploying figlet, slack-handler and slack-interactive functions (5 pts)



6. Provide a screenshot of invoking slack-request and slack-interactive functions (5 pts) Invoking Slack-Request:

Invoking Slack-Request:

Response body

```
{
 "text": "Serverless Message",
 "attachments": [
 {
 "fields": [
 {
 "short": true,
 "value": "100",
 "title": "Amazing Level"
 }
],
 "author_icon": "https://github.com/AdityaShrivastava30.png",
 "image_url": "https://github.com/AdityaShrivastava30.png",
 "author_name": "Aditya Shrivastava",
 "title": "The Awesome world of Cloud Computing! COEN 241"
 },
 {
 "text": "COEN 241 is the most awesome class ever!.",
 "title": "About COEN 241"
 },
 {
 "title": "Would you recommend COEN 241 to your friends?",
 "color": "#3AA3E3",
 "actions": [
 {
 "text": "Of Course!",
 "type": "button",
 "name": "recommend",
 "value": "recommend"
 },
 {
 "text": "Most Definitely!",
 "type": "button",
 "name": "definitely",
 "value": "definitely"
 }
],
 "callback_id": "response123",
 "fallback": "Would you recommend COEN 241 to your friends?",
 "attachment_type": "default"
 }
]
}
```

## Invoking Slack-Interactive:

The screenshot displays the OpenFaas web interface. On the left sidebar, the 'slack-interactive' function is selected. The main panel shows the 'INVOKE' button and the 'Text' radio button selected. The 'Request body' is set to '"500"'. Below this, the 'Response status' is 200 and the 'Round-trip (s)' is 0.066. The 'Response body' is a JSON object representing a Slack message attachment.

```
{
 "attachments": [
 {
 "footer": "Slack Apps built on OpenFaas",
 "author_link": "https://github.com/AdityaShrivastava30",
 "color": "#36a64f",
 "text": "Head over to COEN 241",
 "title": "COEN 241",
 "ts": 123456789,
 "author_name": "",
 "title_link": "https://www.scu.edu/engineering/academic-programs/department-of-computer-engineering/graduate/course-descriptions/",
 "image_url": "https://www.scu.edu/media/offices/umc/scu-brand-guidelines/visual-identity-amp-photography/visual-identity-toolkit/logos-amp-seals/Mission-Dont3.png",
 "response_type": "ephemeral",
 "replace_original": true,
 "footer_icon": "https://a.slack-edge.com/45901/marketing/img/_rebrand/meta/slack_hash_256.png",
 "pretext": "Ahh yeah! Great choice, COEN 241 is absolutely amazing!",
 "fallback": "Required plain-text summary of the attachment.",
 "thumb_url": "https://www.scu.edu/engineering/academic-programs/department-of-computer-engineering/graduate/course-descriptions/",
 "author_icon": "https://github.com/AdityaShrivastava30.png"
 }
]
}
```

## 7. Complete the chatbot with a yml file (25pt)

Chatbot/handler.py:



```

import datetime
from pyfiglet import Figlet
import sys

def handle(req):
 """Process incoming requests based on the input text"""
 if "name" in req.lower() or "what is your name" in req.lower():
 # Respond with the bot's name in 3 different ways
 responses = [
 "My name is Cloud.",
 "I'm called Cloud CC.",
 "You can call me Cloud 241."
]
 return "\n".join(responses)
 elif "current time" in req.lower() or "current date" in req.lower():
 # Respond with the current date and time in 3 different ways
 now = datetime.datetime.now()
 responses = [
 now.strftime("The current time is %H:%M on %B %d, %Y."),
 now.strftime("It's now %H:%M on %d/%m/%Y."),
 now.strftime("Today is %B %d, %Y, and the time is %H:%M.")
]
 return "\n".join(responses)
 elif req.lower().startswith("generate a figlet for"):
 # Extract the text to generate figlet
 text = req[len("generate a figlet for"):].strip("\n ")
 # For the purpose of this example, we'll simulate figlet output using PyFiglet
 f = Figlet(font='slant')
 return f.renderText(text)
 else:
 return "I'm not sure how to process that request."

if __name__ == "__main__":
 # For local testing, input can be sent directly through command line
 req = sys.argv[1] if len(sys.argv) > 1 else ""
 print(handle(req))

```

## Chatbot.yml:

```

version: 1.0
provider:
 name: openfaas
 gateway: http://127.0.0.1:8080
functions:
 slack-request:
 lang: python
 handler: ./slack-request
 image: aditya300998/slack-request:latest
 environment:
 content_type: application/json

```

8. Provide a screenshot of invoking three different cases of the chatbot (5 pts)

Case 1:



sleep

chatbot

slack-interactive

slack-request

### Invoke function

**INVOKE**

☒ Text ☐ JSON ☐ Download

Request body


`*name*`

| Response status | Round trip (s) |
|-----------------|----------------|
| 200             | 0.296          |

Response body

My name is Cloud.  
I'm called Cloud CC.  
You can call me Cloud 241.

Case 2:



sleep

chatbot

slack-interactive

slack-request

### Invoke function

**INVOKE**

☒ Text ☐ JSON ☐ Download

Request body

`*current time*`

| Response status | Round trip (s) |
|-----------------|----------------|
| 200             | 0.182          |

Response body

The current time is 23:27 on February 26, 2024.  
It's now 23:27 on 26/02/2024.  
Today is February 26, 2024, and the time is 23:27.

### Case 3:

```
ubuntu@faasd: $ echo "generate a figlet for Aditya Shrivastava" | faas-cli invoke chatbot
```



```
Aditya Shrivastava
```

```
ubuntu@faasd: $
```

### Step 9:

#### 1. What is the command to invoke the slack-request function (2 pts)?

- ➔ Via faas-cli = Command: `echo "Aditya Shrivastava" | faas-cli invoke slack-request`
- ➔ Via curl = `echo "Aditya Shrivastava" | curl -d @- http://10.101.54.43:8080/function/slack-request`

#### 2. What is the output you see when you invoke the slack-request function? (2 pts)

```
ubuntu@faasd: ~/functions$ echo "Aditya Shrivastava" | faas-cli invoke slack-request
```

```
{
 "text": "Serverless Message",
 "attachments": [
 {
 "fields": [
 {
 "short": true,
 "value": "100",
 "title": "Amazing Level"
 }
],
 "author_icon": "https://github.com/AdityaShrivastava30.png",
 "image_url": "https://github.com/AdityaShrivastava30.png",
 "author_name": "Aditya Shrivastava",
 "title": "The Awesome world of Cloud Computing! COEN 241",
 "text": "COEN 241 is the most awesome class ever!",
 "title": "About COEN 241",
 "text": "Would you recommend COEN 241 to your friends?",
 "color": "#3AA3E3",
 "actions": [
 {
 "text": "Of Course!",
 "type": "button",
 "name": "recommend",
 "value": "recommend"
 },
 {
 "text": "Most Definitely!",
 "type": "button",
 "name": "definitely",
 "value": "definitely"
 }
],
 "callback_id": "response123",
 "fallback": "Would you recommend COEN 241 to your friends?",
 "attachment_type": "default"
 }
]
}
```

```
ubuntu@faasd: ~/functions$
```

#### 3. What is the command to invoke the slack-interactive function? (2 pts)

##### a. Via curl:

- ➔ `curl -d '{"Nityanand":"COEN 241"}' http://10.101.54.43:8080/function/slack-interactive`

**b. Via faas-cli:**

➔ `sudo faas-cli invoke slack-interactive`

**4. What is the output you see when you invoke the slack-interactive function? (2 pts)**

```
ubuntu@faasd:~/functions$ curl -d '{"Aditya":"COEN 241"}' http://10.101.54.43:8080/function/slack-interactive
{"attachments": [{"footer": "Slack Apps built on OpenFaas", "author_link": "https://github.com/AdityaShrivastava30", "color": "#36a64f", "text": "Head over to COEN 241", "title": "COEN 241", "ts": 123456789, "author_name": "", "title_link": "https://www.scu.edu/engineering/academic-programs/department-of-computer-engineering/graduate/course-descriptions/", "image_url": "https://www.scu.edu/media/offices/unc/scu-brand-guidelines/visual-identity-and-photography/visual-identity-toolkit/logos-and-seals/Mission-Dont3.png", "response_type": "ephemeral", "replace_original": true, "footer_icon": "https://a.slack-edge.com/45901/marketing/img/rebrand/meta/slack_hash_256.png", "pretext": "Ahh yeah! Great choice. COEN 241 is absolutely amazing!", "fallback": "Required plain-text summary of the attachment.", "thumb_url": "https://www.scu.edu/engineering/academic-programs/department-of-computer-engineering/graduate/course-descriptions/", "author_icon": "https://github.com/AdityaShrivastava.png"}]}
Reading from STDIN - hit (Control + D) to stop.
"AR"
{"attachments": [{"footer": "Slack Apps built on OpenFaas", "author_link": "https://github.com/AdityaShrivastava30", "color": "#36a64f", "text": "Head over to COEN 241", "title": "COEN 241", "ts": 123456789, "author_name": "", "title_link": "https://www.scu.edu/engineering/academic-programs/department-of-computer-engineering/graduate/course-descriptions/", "image_url": "https://www.scu.edu/media/offices/unc/scu-brand-guidelines/visual-identity-and-photography/visual-identity-toolkit/logos-and-seals/Mission-Dont3.png", "response_type": "ephemeral", "replace_original": true, "footer_icon": "https://a.slack-edge.com/45901/marketing/img/rebrand/meta/slack_hash_256.png", "pretext": "Ahh yeah! Great choice, COEN 241 is absolutely amazing!", "fallback": "Required plain-text summary of the attachment.", "thumb_url": "https://www.scu.edu/engineering/academic-programs/department-of-computer-engineering/graduate/course-descriptions/", "author_icon": "https://github.com/AdityaShrivastava.png"}]}}
```

**5. How would you pass different arguments to the functions? (3 pts)**

Here are two ways of passing arguments to functions:

1. Using faas-cli: We can invoke the figlet function from the command line using faas-cli and pipe in the argument "Hello, FaaS World" like this: `echo "Hello, FaaS World" | faas-cli invoke figlet`  
The text "Hello, FaaS World" gets passed as an argument to the figlet function.
2. Using curl: We can send arguments in a POST request to an HTTP server. For example: `curl -d '{"text":"Hello COEN 241"}' http://example.com/function/slack-request` This curl command sends a JSON payload `{"text":"Hello COEN 241"}` as data in a POST request to the `/function/slack-request` endpoint. The text "Hello COEN 241" gets passed as an argument to the function.

**6. How would you change the slack-interactive function to react to different inputs? (3 pts)**

➔ Changes to the slack-interactive function's code are required to enable it to handle different inputs. To do this, code would need to be added that determines the type of input it has received and takes the appropriate action. Assume you would like to enable JSON inputs for the function. If that's the case, you can modify the code by adding a step that uses the `json.loads()` function to understand the incoming JSON. This stage would

transform the JSON input into a Python dictionary so that the function could process it appropriately.

You may need to make changes to the OpenFaaS gateway user interface or command line used to trigger the function, including sending the JSON input as an argument, in order to enable the slack-interactive function to handle different inputs. This would allow the input to be passed into the function as a parameter, allowing it to be processed appropriately.

In conclusion, altering the slack-interactive function's code and making sure the input is handled appropriately inside the function are necessary to adapt it to accept various input kinds.

**7. How long does it take for the chat response to come back? (10pts)**

- a) For the first request that does not call figlet
- b) For the second request that does not call figlet
- c) Average over 10 requests that do not call figlet
- d) For the first request that calls figlet
- e) For the second request that calls figlet
- f) For the second request that calls figlet that follows the first request that does not call figlet
- g) Average over 10 requests that do call figlet

```

import requests
import time

Endpoint URL for the chatbot service
BOT_SERVICE_URL = "http://10.101.54.43:8080/function/chatbot"

def calculate_avg_latency(payload, iterations=1):
 """Calculates the latency of a request or the average latency over several iterations."""
 cumulative_time = 0
 for _ in range(iterations):
 start = time.time()
 response = requests.post(BOT_SERVICE_URL, data=payload)
 finish = time.time()
 cumulative_time += (finish - start)
 if iterations == 1: # Directly return the latency for a single iteration
 return finish - start
 return cumulative_time / iterations # Return the average latency over multiple iterations

def main():
 # Evaluating response times under different conditions
 # a. Initial query without invoking figlet
 latency_without_figlet_initial = calculate_avg_latency("How are you?")
 print(f"a. Response time for the first request (no figlet): {latency_without_figlet_initial:.4f} seconds")

 # b. Follow-up query without invoking figlet
 latency_without_figlet_followup = calculate_avg_latency("How are you?")
 print(f"b. Response time for the second request (no figlet): {latency_without_figlet_followup:.4f} seconds")

 # c. Average latency over multiple queries not invoking figlet
 average_latency_without_figlet = calculate_avg_latency("How are you?", iterations=10)
 print(f"c. Average response time over 10 requests (no figlet): {average_latency_without_figlet:.4f} seconds")

 # d. Initial query invoking figlet
 latency_with_figlet_initial = calculate_avg_latency("Create a figlet for Hello")
 print(f"d. Response time for the first request (with figlet): {latency_with_figlet_initial:.4f} seconds")

 # e. Follow-up query invoking figlet
 latency_with_figlet_followup = calculate_avg_latency("Create a figlet for Hello")
 print(f"e. Response time for the second request (with figlet): {latency_with_figlet_followup:.4f} seconds")

 # f. Follow-up query invoking figlet after an initial query without it
 # Execute initial query without figlet
 calculate_avg_latency("How are you?")
 # Then measure latency for follow-up query with figlet
 latency_with_figlet_after_without = calculate_avg_latency("Create a figlet for Hello")
 print(f"f. Response time for the second request (with figlet, after no figlet): {latency_with_figlet_after_without:.4f} seconds")

 # g. Average latency for multiple queries invoking figlet
 average_latency_with_figlet = calculate_avg_latency("Create a figlet for Hello", iterations=10)
 print(f"g. Average response time over 10 requests (with figlet): {average_latency_with_figlet:.4f} seconds")

if __name__ == "__main__":
 main()

```

```

ubuntu@faasd:~$ python3 chatbot_avg.py
a. Response time for the first request (no figlet): 0.1464 seconds
b. Response time for the second request (no figlet): 0.1449 seconds
c. Average response time over 10 requests (no figlet): 0.1483 seconds
d. Response time for the first request (with figlet): 0.1524 seconds
e. Response time for the second request (with figlet): 0.1528 seconds
f. Response time for the second request (with figlet, after no figlet): 0.1526 seconds
g. Average response time over 10 requests (with figlet): 0.1527 seconds
ubuntu@faasd:~$

```

8. Now try sending a series of requests to the chatbot in parallel. At what queries per second does OpenFaaS add a new instance of the function? (6 pts)

```

import concurrent.futures
import requests
import time

Define the URL where the chatbot service is hosted
SERVICE_ENDPOINT = "http://10.101.54.43:8080/function/chatbot-service"

def send_request_to_service(data_payload):
 """Function to send data to the chatbot service endpoint."""
 try:
 response = requests.post(SERVICE_ENDPOINT, data=data_payload)
 return response.status_code
 except Exception as e:
 return str(e)

def perform_concurrent_requests(req_per_sec, duration_sec=10):
 """Function to send concurrent requests to the service."""
 with concurrent.futures.ThreadPoolExecutor() as executor:
 future_tasks = []
 time_start = time.time()

 while time.time() - time_start < duration_sec:
 for _ in range(req_per_sec):
 future = executor.submit(send_request_to_service, "What's the weather like?")
 future_tasks.append(future)
 time.sleep(1) # Wait before sending the next batch of requests

 results = [future.result() for future in future_tasks]

 successful_results = [result for result in results if result == 200]
 print(f"Total requests sent: {len(results)}")
 print(f"Successful responses: {len(successful_results)}")
 print(f"Success Rate: {(len(successful_results) / len(results)) * 100:.2f}%")

Example of calling the function:
request_rate = 5 # You can adjust this value to test different load scenarios
perform_concurrent_requests(request_rate)

```

```

ubuntu@faasd:~$ python3 chatbot_parallel.py
Total requests: 50
Successful responses: 50
Success rate: 100.00%

```



## Extra credit:

### Invite to workspace:

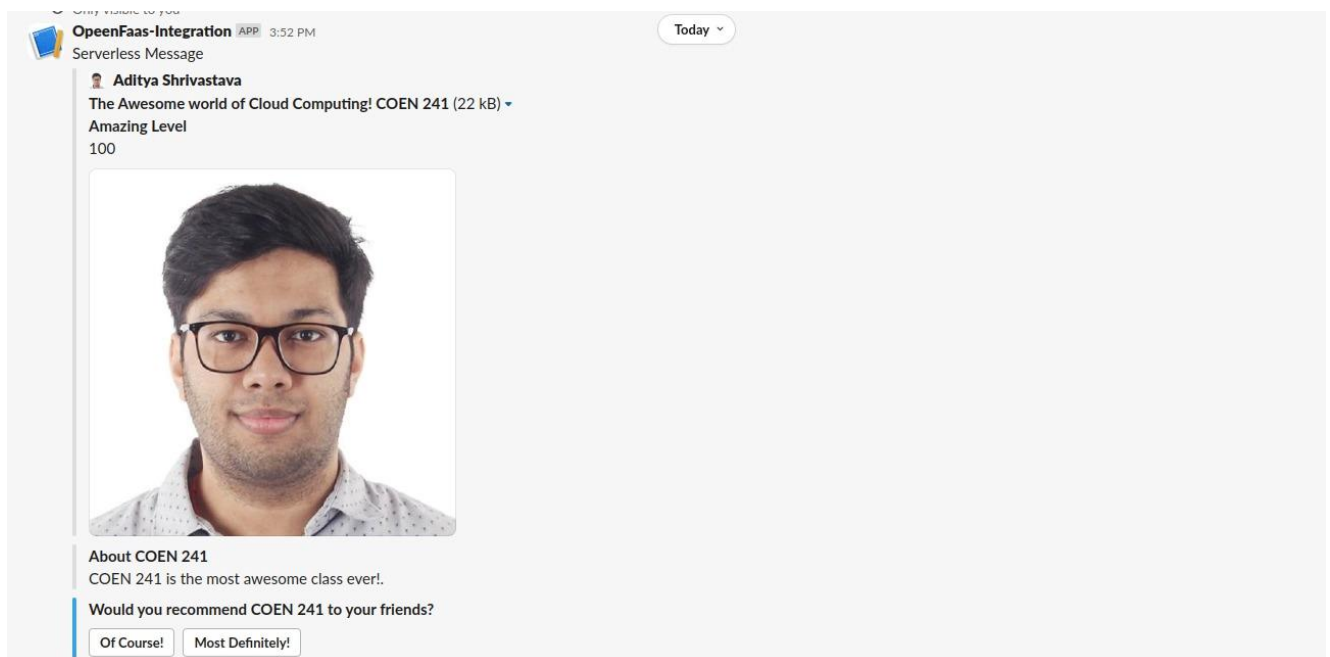
[https://join.slack.com/t/csen241cloudc-e4n4464/shared\\_invite/zt-2dr1qdfcx-NKWw5kmpbgo4obeyiIU3GQ](https://join.slack.com/t/csen241cloudc-e4n4464/shared_invite/zt-2dr1qdfcx-NKWw5kmpbgo4obeyiIU3GQ)

### Link to application:

<https://app.slack.com/client/T06LT54CLMQ/C06LVJMU684>

### Slash-command url:

<https://2d1c-24-23-244-181.ngrok-free.app/function/slack-request>





Only visible to you


**OpeenFaas-Integration** APP 3:52 PM

Ahh yeah! Great choice, COEN 241 is absolutely amazing!

**COEN 241**

Head over to COEN 241

Slack Apps built on OpenFaas | Nov 29th, 1973 (16 kB) ▾



```
Version 3.6.0
Region United States (California) (us-cal-1)
Latency 19ms
Web Interface http://127.0.0.1:4040
Forwarding https://95c0-24-23-244-181.ngrok-free.app -> http://localhost:8080

Connections ttl opn rt1 rt5 p50 p90
 1 1 0.01 0.00 60.05 60.05

HTTP Requests

POST /function/slack-interactive 200 OK
POST /function/slack-request 200 OK
```