

CS-524 Course Project

CS 524 A - Introduction to Cloud Computing

Spring 2022

By,

Omkar Sinha

CWID: 10468301

Dept. Computer Science

Stevens Institute of Technology

Under Guidance of,

Prof. Dr. Igor Faynberg

PART A: AWS

1) Firstly, we login to our AWS Account, as below:

The screenshot shows the AWS Management Console homepage. At the top, there's a banner about the new AWS Console Home. Below it, the title "AWS Management Console" is displayed. On the left, there's a sidebar titled "AWS services" with sections for "Recently visited services" (AWS Marketplace Subscriptions, EC2, CloudFormation, FSx, Storage Gateway, VPC, IAM, AWS Organizations, Simple Queue Service) and "All services". In the center, there's a "Build a solution" section with options like "Launch a virtual machine", "Build a web app", and "Build using virtual servers". On the right, there are two main sections: "New AWS Console Home" (describing the new interface) and "Stay connected to your AWS resources on-the-go" (describing the AWS Console Mobile App). The bottom of the page includes a feedback link, copyright information (© 2022, Amazon Web Services, Inc. or its affiliates.), and links for Privacy, Terms, and Cookie preferences.

2) We open the services and choose Lambda:

The screenshot shows the AWS search results for the term "Lambda". The search bar at the top contains "Lambda". The results are listed under the "Services" category. The first result is "Lambda" (Run Code without Thinking about Servers), which is highlighted. Other results include "CodeBuild" (Build and Test Code), "AWS Signer" (Ensuring trust and integrity of your code), and "Amazon Lex" (Build Voice and Text Chatbots). To the right of the search results, there's a banner for the new AWS Console Home and another for staying connected via the mobile app. The bottom of the page includes a feedback link, copyright information (© 2022, Amazon Web Services, Inc. or its affiliates.), and links for Privacy, Terms, and Cookie preferences.

3) We see the AWS Lambda page (currently no functions are created) :-

The screenshot shows the AWS Lambda Functions page. The left sidebar has sections for "Dashboard", "Applications", "Functions" (which is selected and highlighted in orange), "Additional resources", and "Related AWS resources". The main content area is titled "Functions (0)" and shows a table with columns: Function name, Description, Package type, Runtime, Code size, and Last modified. A message at the bottom of the table says "There is no data to display." The top of the page includes the AWS logo, a search bar, and navigation links for Services, N. Virginia, Omkar Sinha, and other standard AWS links.

4) We choose “create function” to create a new function “Avengers”:

(i) We put the function name as “Avengers”, keeping other details same:

The screenshot shows the 'Create function' wizard in the AWS Lambda console. The 'Basic information' step is active, with the 'Function name' field set to 'Avengers'. Other options like 'Author from scratch', 'Use a blueprint', 'Container image', and 'Browse serverless app repository' are shown but not selected. The 'Runtime' dropdown is set to 'Node.js 14.x'. The 'Architecture' dropdown is set to 'x86_64'. The 'Permissions' section indicates a default execution role will be created. The 'Advanced settings' section is collapsed. At the bottom right are 'Cancel' and 'Create function' buttons.

5) We see that “Avengers” Lambda function is created:

The screenshot shows the 'Function overview' page for the 'Avengers' Lambda function. A success message at the top states: 'Successfully created the function Avengers. You can now change its code and configuration. To invoke your function with a test event, choose "Test".' The main interface includes sections for 'Function overview' (with a thumbnail icon), 'Code source' (with an 'Upload from' button), and 'Actions' (Throttle, Copy ARN). Below these are tabs for 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The 'Code' tab is currently selected. At the bottom are 'Feedback', '© 2022, Amazon Web Services, Inc. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'.

We see one function listed in Lambda -> Functions:

The screenshot shows the AWS Lambda console. On the left, there's a sidebar with 'AWS Lambda' selected under 'Functions'. The main area is titled 'Lambda > Functions' and shows a table with one row. The row contains the function name 'Avengers', package type 'Zip', runtime 'Node.js 14.x', code size '304.0 byte', and last modified '1 minute ago'. There are buttons for 'Actions' and 'Create function' at the top right of the table.

6) Now we proceed to add our given code source to the “Avengers” function. We copy and paste the code into the source code window as follows:

The screenshot shows the AWS Lambda Code source editor for the 'Avengers' function. The title bar says 'Successfully created the function Avengers. You can now change its code and configuration. To invoke your function with a test event, choose "Test."'. The editor has tabs for 'Code source' and 'Info'. The 'Code source' tab is active, showing the 'index.js' file. The code is a Node.js script that defines a Lambda function with a specific service URL and a list of questions. The 'Deploy' button is visible at the top of the editor.

```

1 var json = { "service": 
2   "lambda": {
3     "function": "https://aws.amazon.com/lambda/avengers/", "questions": [
4       {"q": "What is the real name of the Scarlet Witch?", 
5        "a": "Wanda Maximoff"}, {
6         "q": "Which film did The Aether first appear in?", 
7         "a": "Thor: The Dark World"}, {
8           "q": "Which of the Infinity stones is hidden on Vormir?", 
9           "a": "Soul Stone"}, {
10          "q": "What is Captain America's shield made of?", 
11          "a": "Vibranium"}, {
12            "q": "Which country is Black Panther next in line to be king of?", 
13            "a": "Wakanda"}, {
14              "q": "What is the real name of Black Widow?", 
15              "a": "Natasha Romanoff"}, {
16                "q": "What is the name of the axe created for and then used by Thor in Avengers: Infinity War?", 
17                "a": "Stormbreaker"}, {
18                  "q": "What is Loki's title?", 
19                  "a": "God of Mischief"}, {
20                    "q": "What is the name of the organisation which is revealed to have infiltrated S.H.I.E.L.D. in Captain America: The Winter Soldier?", 
21                    "a": "Hydra"}, {
22                      "q": "What nickname does Captain America know the Winter Soldier as?", "a": "Bucky"}, {
23                      "q": "What food do the Avengers go to eat after the Battle of New York in the first Avengers film at Tony Stark's suggestion?", 
24                      "a": "Shawarma"}, {
25                        "q": "What type of radiation caused Bruce Banner to become the Hulk?", 
26                        "a": "Gamma rays"}, {
27                          "q": "What is the name of the treaty the Avengers are asked to sign which divide the Avengers, bringing us to Captain America: Civil War?", 
28                          "a": "The Sokovia Accords"}, {
29                            "q": "Who has directed the most MCU movies?", 
30                            "a": "The Russo Brothers"}, {
31                            "q": "What is the name of the Scarlet Witch?", 
32                            "a": "Wanda Maximoff"}, {
33                            "q": "What is the real name of the Scarlet Witch?", 
34                            "a": "Wanda Maximoff"}, {
35                            "q": "Which film did The Aether first appear in?", 
36                            "a": "Thor: The Dark World"}, {
37                            "q": "Which of the Infinity stones is hidden on Vormir?", 
38                            "a": "Soul Stone"}], "exports.handler = function(event, context) { var rand = Math.floor(Math.random() * json.questions.length); console.log("Quote selected: ", rand); var question = json.questions[rand]; context.succeed(question); }"; }

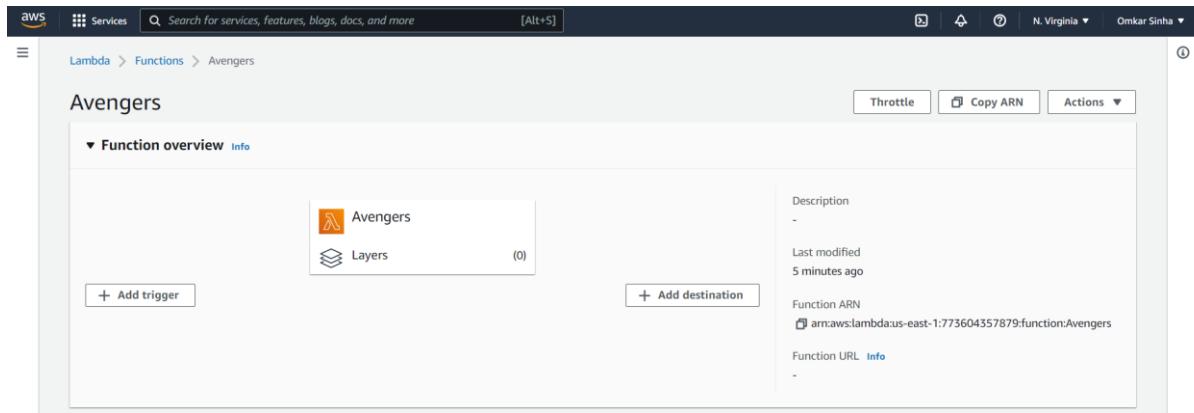
```

After adding the code and making sure that syntax is proper, we choose “Deploy”:

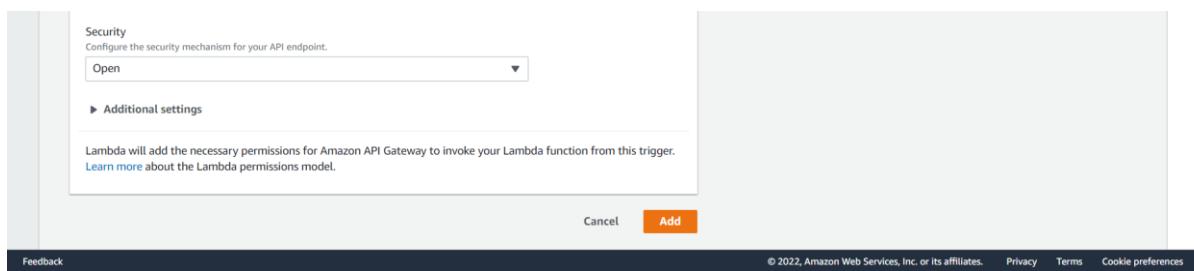
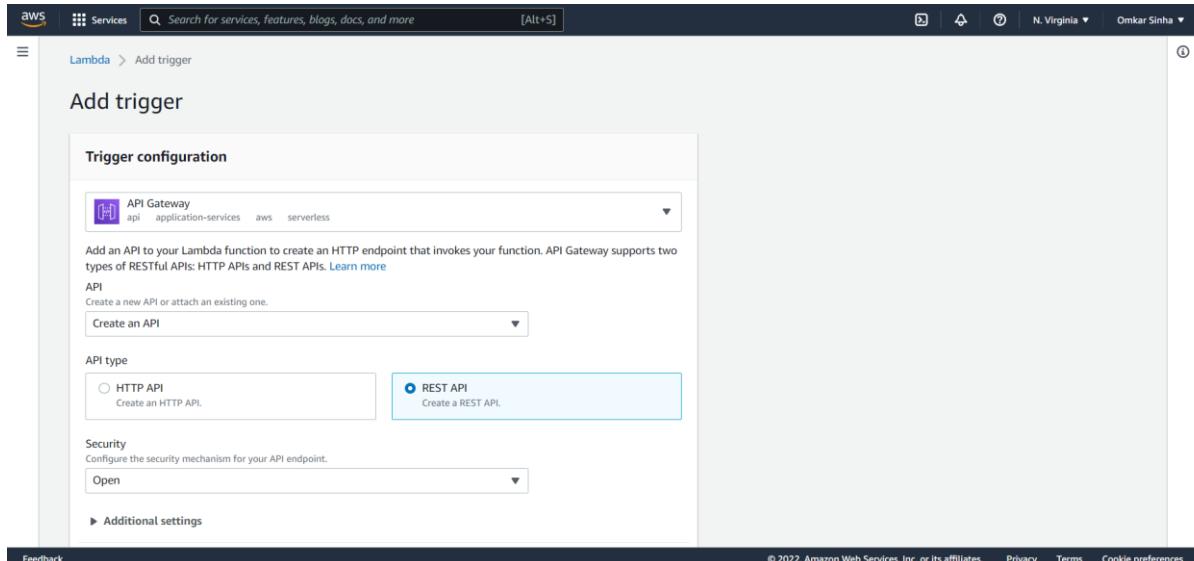
The screenshot shows the AWS Lambda Code source editor for the 'Avengers' function. The title bar says 'Successfully updated the function Avengers.'. The editor interface is identical to the previous screenshot, showing the 'Code source' tab with the 'index.js' file containing the same Node.js code. The 'Deploy' button is visible at the top of the editor.

7) Now we proceed to add trigger to our Lambda function.

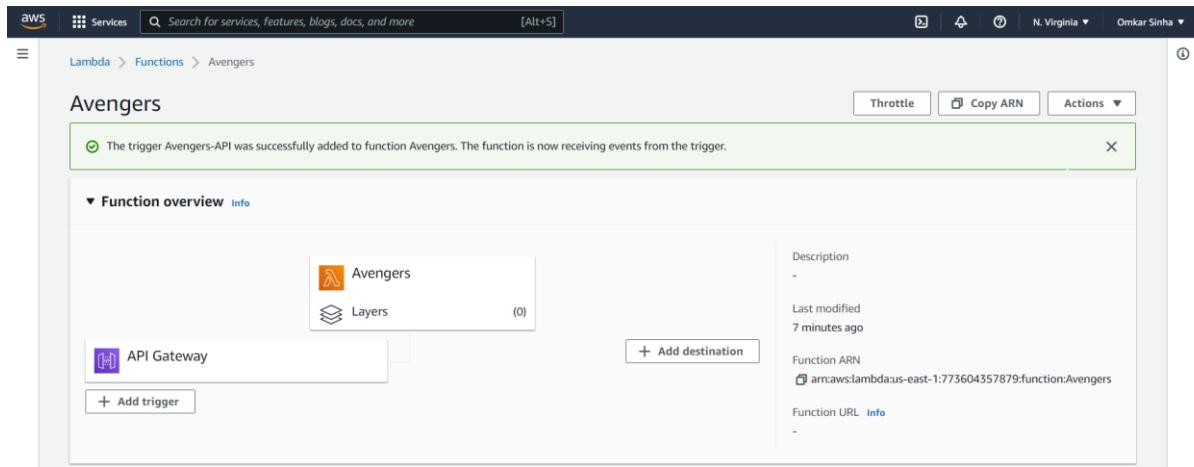
(i) We select “Add Trigger” from the Avengers function home-screen as shown below and choose “API Gateway” from the drop-down menu:



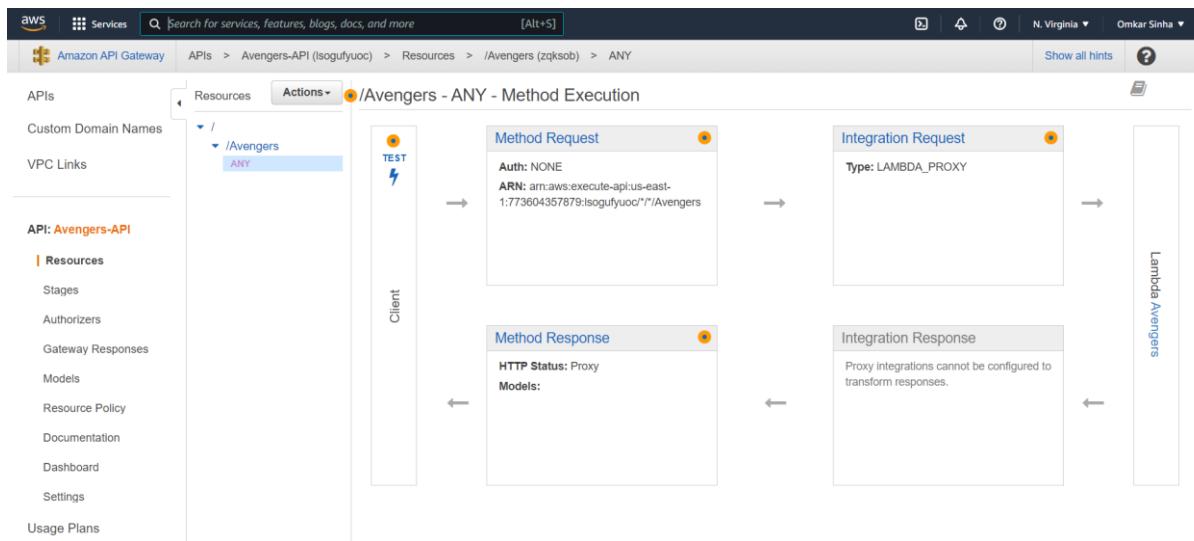
(ii) We choose “Rest API” as the API type and security as “Open”, leaving other details unchanged, and then select “Add”:



(iii) We see a prompt message declaring that the trigger is successfully created:



(iv) We can view the method execution for Avengers function in the API gateway as follows:



8) Now, we proceed to test our Lambda function by copying our API Gateway endpoint URL and pasting it into the browser:

(i) We firstly select the API Gateway from above Avengers function overview, and are presented with the following trigger details:

The screenshot shows the AWS Lambda configuration interface. On the left, a sidebar lists various settings: General configuration, Triggers (selected), Permissions, Destinations, Function URL - new, Environment variables, Tags, VPC, Monitoring and operations tools, Concurrency, Asynchronous invocation, and Code signing. The main panel is titled 'Triggers (1)' and shows a single trigger named 'Trigger'. It details an API Gateway trigger for the 'Avengers-API' with the API endpoint <https://lsogufyuoc.execute-api.us-east-1.amazonaws.com/default/Avengers>. The trigger's details include: API: api-gateway/lsogufyuoc/*/*Avengers, API name: Avengers-API, API type: rest, Authorization: NONE, Enable metrics and error logging: No, Method: ANY, Resource path: /Avengers, Security: NONE, and Stage: default.

(ii) We now test our Lambda function by copying the API endpoint URL from above and paste it into the browser. We see the following question and answer pair in JSON format from marvel series that we included in the code. This proves our function is working:

<https://lsogufyuoc.execute-api.us-east-1.amazonaws.com/default/Avengers>

```
{"q": "What is Captain America's shield made of?", "a": "Vibranium"}
```

(iii) We again run the same URL and get another random question and answer:

<https://n030yxp9j.execute-api.us-east-1.amazonaws.com/default/Avengers>

```
{"q": "What is the name of the organisation which is revealed to have infiltrated S.H.I.E.L.D. in Captain America: The Winter Soldier?", "a": "Hydra"}
```

9) We check CloudWatch logs now after running the URL first time:

(i) We choose “Monitor” from the panel as shown below. Then we check the Metrics toggle. Here we see the following metrics:

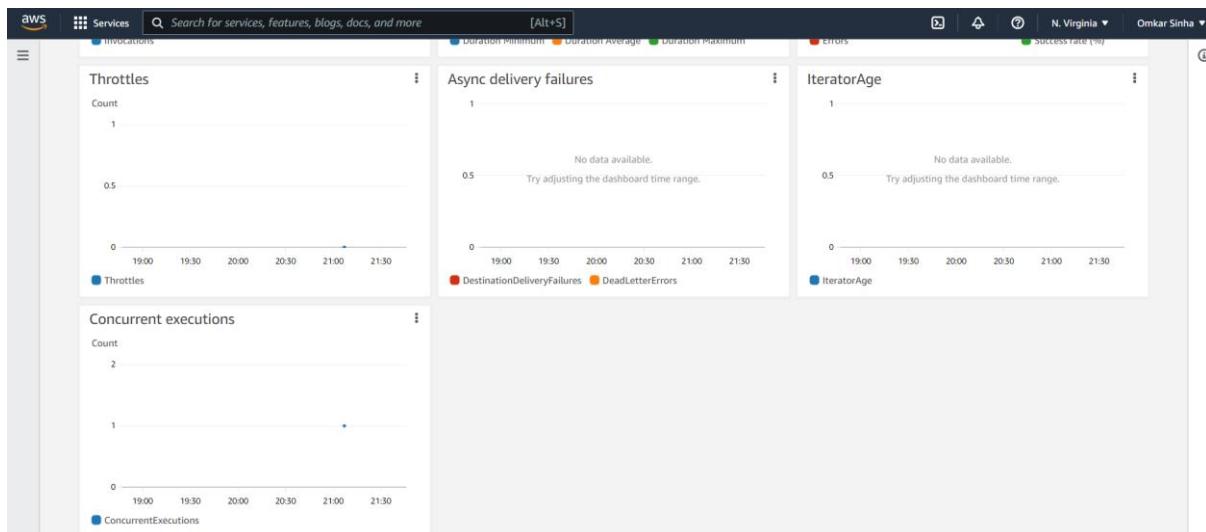
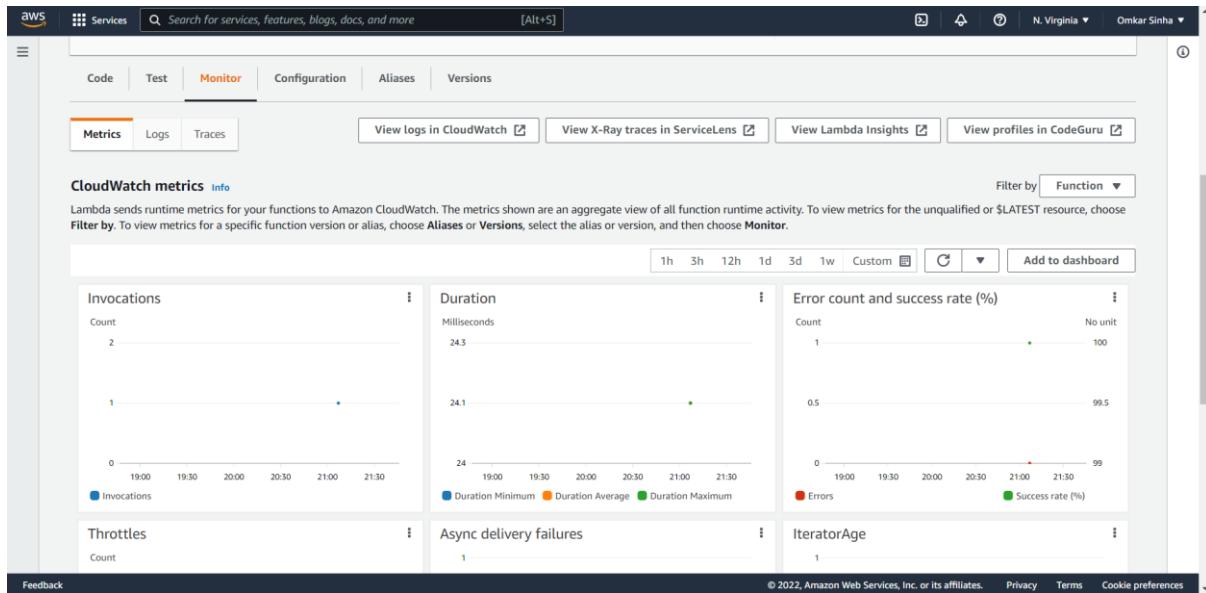
a) **Invocations:** We see the 1 invocation shortly after 9PM

b) **Duration:** In duration graph we find that our invocations are plotted against time. We notice that our invocation took 24.1 milliseconds.

c) **Error Count and success rate:** We see that our success rate is 100% and error rate 0%, for the time interval shortly after 9PM – as represented by green and red dots, respectively.

d) **Concurrent execution:** Concurrent execution has a count of 1 because we had only 1 invocation.

Other metrics like **Throttles**, **Async delivery failures** and **IteratorAge** have no data.



(ii) Now, we choose the Log toggle and see the following details(after our second execution at 21:47):

- a) **Recent invocations:** We see details of our latest invocation including its timestamp, RequestID, log stream, duration, etc.
- b) **Most Expensive invocation:** This gives billing details for invocation starting with most expensive. Our 2nd invocation is the most expensive in this case and its timestamp, request ID, logstream, billed duration, memory used(in MB) and other details are given.

The screenshot shows the AWS CloudWatch Logs Insights interface. At the top, there are tabs for Metrics, Logs (which is selected), and Traces. Below the tabs are buttons for View logs in CloudWatch, View X-Ray traces in ServiceLens, View Lambda Insights, and View profiles in CodeGuru. The main area displays two tables of log data.

Recent invocations

#	:Timestamp	:RequestID	:LogStream	:DurationInMS	:BilledDurationInMS	:MemorySetInMB
1	2022-04-21T21:47:31.641Z	686880de-dfe1-4639-9703-edad7b74572f	2022/04/21/[LATEST]b48799c80573430289ff4223f068ae	57.31	58	128

Most expensive invocations in GB-seconds (memory assigned * billed duration)

#	:Timestamp	:RequestID	:LogStream	:BilledDurationInMS	:MemorySetInMB	:BilledDura
1	2022-04-21T21:47:31.641Z	686880de-dfe1-4639-9703-edad7b74572f	2022/04/21/[LATEST]b48799c80573430289ff4223f068ae	58	128	0.00725

10) Now, we invoke the Lambda function and view CloudWatch logs for this function.

(i) Firstly, we need to create a test event. We choose “test event” from Actions as follows:

The screenshot shows the AWS Lambda Functions page. A single function named "Avengers" is selected. The table shows the following details:

Function name	Description	Package type	Runtime	Code size
Avengers	-	Zip	Node.js 14.x	968.0 byte

(ii) We choose “Create new event” and name it “MyInvokeEvent”, while keeping rest of the details unchanged:

The screenshot shows the AWS Lambda Test event configuration page. The "Test event" section is active. The "Test event action" dropdown is set to "Create new event". The "Event name" field contains "MyInvokeEvent". The "Event sharing settings" section has "Private" selected. The "Template - optional" field contains "hello-world". The "Event JSON" section shows the following JSON code:

```

1 - [
2   {
3     "key1": "value1",
4     "key2": "value2",
5     "key3": "value3"
6   ]

```

(iii) Then we select “Test”. We see the following prompt announcing “Execution result: succeeded”.

The screenshot shows the AWS Lambda Test interface. At the top, there are tabs for Code, Test (which is selected), Monitor, Configuration, Aliases, and Versions. Below the tabs, a green box displays the message "Execution result: succeeded (logs)" with a link to "Details". At the bottom of the screen, there is a "Test event" section with "Save" and "Test" buttons.

(iv) We expand the details to see the following. The body of the log output consist of the question-answer pair related to the marvel series: (We execute the function twice)

The screenshot shows the expanded "Details" view of the execution log. It includes a summary of the request (Code SHA-256, Duration, Resources configured) and the log output itself. The log output shows two invocations of the function. The first invocation's log is as follows:

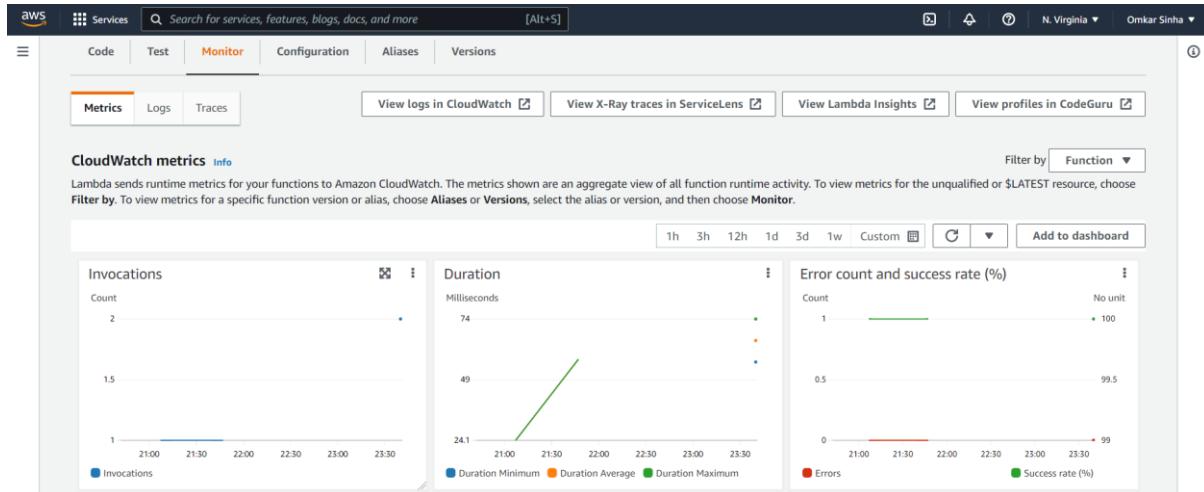
```

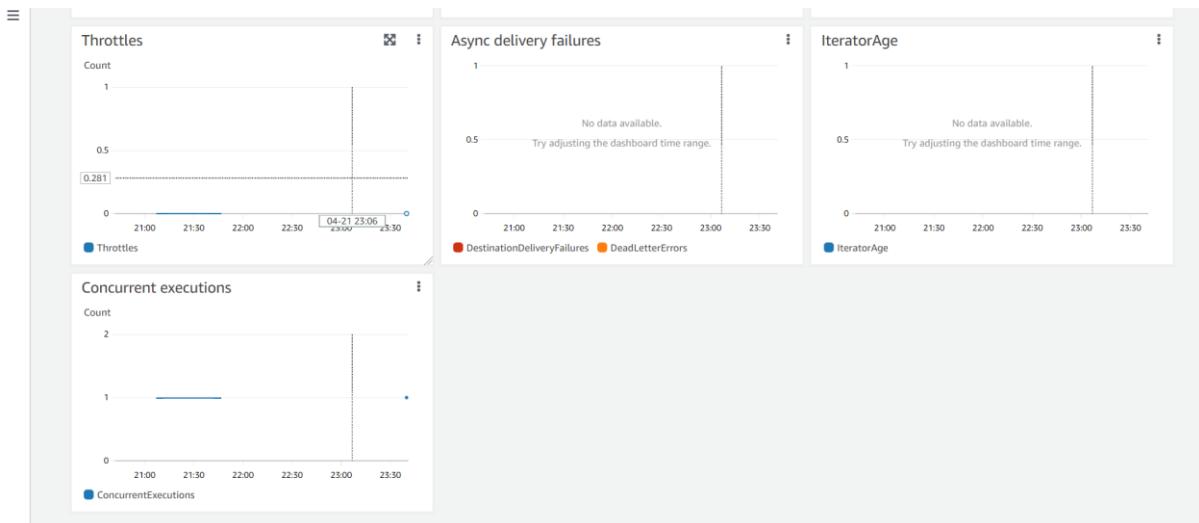
{
  "body": "{\"q\":\"what type of radiation caused Bruce Banner to become the Hulk?\",\"a\":\"Gamma rays\"}"
}
  
```

The second invocation's log is identical. The log output also includes the Request ID, Billed duration, and Max memory used for each invocation.

11) We view the cloudWatch details.

(i) We see the following metrics as follows for our invocations:





(ii) We also find the following logs

(iii) We open CloudWatch and find following log details(here local time-zones are shown):

(iv) We click on the log streams to view log events. We can see details of timestamps, ID, start and end time and all the information sent of the requests made using our API:

The screenshot shows the AWS CloudWatch Log Events interface. On the left, there's a sidebar with navigation links like 'CloudWatch', 'Favorites', 'Dashboards', 'Alarms', 'Logs' (which is selected), 'Log groups', 'Logs Insights', 'Metrics', 'X-Ray traces', 'Events', 'Application monitoring', and 'Insights'. The main area is titled 'Log events' and shows a list of log entries. The first entry is a 'START RequestId' followed by several 'INFO' entries detailing a query about Loki's title. The last entry is a 'REPORT RequestId' with details about the request duration and memory usage.

12) Conclusion of the report:

We can conclude from this exercise that the microservices architecture creates a neat division of responsibility between various services. The Lambda acts as serverless computing platform where we upload our code and attach an API gateway as the trigger. We create this API gateway as a REST API to both receive and serve up json inputs/outputs upon HTTP call. The output json follows the REST model. This is also a strong demonstration of AWS Lambda's serverless platform which provides a clear separation between the infrastructure and applications that run on the platform.

PART B: Salesforce

1) We login to our Salesforce account and go to setup:

The screenshot shows the Salesforce Setup interface. At the top, there's a banner with the text 'It's Better in Lightning' and 'Move to Lightning Experience and give your users a productivity boost.' Below the banner, there's a 'Getting Started' section with two main items: 'Build App' (with a 'Get Started' button) and 'Salesforce Lightning' (with a 'Get Started' button). To the right of these, there's a 'TRAILHEAD' section with a green background and the text 'The fun way to learn Salesforce' and a 'VIEW FREE TUTORIALS >' button. On the left side, there's a sidebar with sections for 'Lightning Experience Transition Assistant', 'Salesforce Mobile Quick Start', 'Home', and 'Administrator' (with sub-options like 'Release Updates', 'Manage Users', 'Manage Apps', 'Manage Territories', and 'Company Profile'). A 'Recent Items' table is also visible on the left.

2) Creating custom field on Account Object:

(i) Now, we will create custom field name: Field Update, on the Account object. We search “Accounts” in quick find bar and go to fields. We get all fields:

The screenshot shows the Salesforce Home page. On the left, there's a sidebar with navigation links like Home, Chatter, Libraries, Content, Subscriptions, Student Information, and a 'Build' section containing Campaigns, Leads, Accounts, and Fields. In the center, there's a 'Getting Started' section with a 'Build App' card and a 'Salesforce Lightning' card. Below that is a 'Recent Items' table with three entries: Omkar Sinha (User), SFDC_DevConsole (Debug Level), and Student Information (Custom Object Definition). On the right, there's a 'TRAILHEAD' banner with the text 'the fun way to learn Salesforce' and 'VIEW FREE TUTORIALS >'. At the bottom right, there are 'Browse Docs' and 'Explore documentation' links.

We scroll down to “Account Custom Fields & Relationships” and click on “New”:

The screenshot shows the 'Account Custom Fields & Relationships' list view. The left sidebar has sections for Activities, Campaigns, Leads, and Accounts. Under 'Fields', it lists Related Lookup Filters, Validation Rules, Triggers, Partner Roles, Contact Roles on Accounts, Page Layouts, Field Sets, Compact Layouts, Search Layouts, Buttons, Links, and Actions. The main table lists custom fields for the Account object, such as Website (Type: URL(255)), YearStarted (Type: Text(4)), Active (Type: Picklist), CustomerPriority (Type: Picklist), NumberofLocations (Type: Number(3, 0)), SLA (Type: Picklist), SLAExpirationDate (Type: Date), SLASerialNumber (Type: Text(10)), and UpsellOpportunity (Type: Picklist). Each row includes 'Edit | Del | Replace' buttons and a 'Modified By' column showing 'Omkar.Sinha' at 8:18 AM on 4/13/2022.

(ii) We find dialog screen for new custom field details:

The screenshot shows the 'New Custom Field' dialog for Step 1: Choose the field type. It has a 'Data Type' section with a 'None Selected' radio button and a note to 'Select one of the data types below.' Below are four options: 'Auto Number' (described as a system-generated sequence number), 'Formula' (described as a read-only field derived from a formula expression), 'Roll-Up Summary' (described as a read-only field displaying the sum, minimum, or maximum value of a field in a related list), and 'Lookup Relationship' (described as creating a relationship between objects). There are 'Next' and 'Cancel' buttons at the top right.

First is data type. We choose checkbox, and go to next:

Step 1. Choose the field type

Specify the type of information that the custom field will contain.

Data Type

None Selected	Select one of the data types below.
<input type="radio"/> Auto Number	A system-generated sequence number that uses a display format you define. The number is automatically incremented for each new record.
<input type="radio"/> Formula	A read-only field that derives its value from a formula expression you define. The formula field is updated when any of the source fields change.
<input type="radio"/> Roll-up Summary	A read-only field that displays the sum, minimum, or maximum value of a field in a related list or the record count of all records listed in a related list.
<input type="radio"/> Lookup Relationship	Creates a relationship that links this object to another object. The relationship field allows users to click on a lookup icon to select a value from a popup list. The other object is the source of the values in the list.
<input type="radio"/> External Lookup Relationship	Creates a relationship that links this object to an external object whose data is stored outside the Salesforce org.
<input checked="" type="radio"/> Checkbox	Allows users to select a True (checked) or False (unchecked) value.
<input type="radio"/> Currency	Allows users to enter a dollar or other currency amount and automatically formats the field as a currency amount. This can be useful if you export data to Excel or another spreadsheet.
<input type="radio"/> Date	Allows users to enter a date or pick a date from a popup calendar.
<input type="radio"/> Date/Time	Allows users to enter a date and time, or pick a date from a popup calendar. When users click a date in the pop-up, that date and the current time are entered into the Date/Time field.
<input type="radio"/> Email	Allows users to enter an email address, which is validated to ensure proper format. If this field is specified for a contact or lead, users can choose the address when clicking Send an Email. Note that custom email addresses cannot be used for mass emails.

(iii) We enter the other details for the field, ad go to next:

Step 2. Enter the details

Field Label

Default Value Checked Unchecked

Field Name

Description

Help Text

Auto add to custom report type Add this field to existing custom report types that contain this entity

(iv) We check “visibility” and go to next:

Step 3. Establish field-level security

Field-Level Security for Profile	Visible	Read-Only
Analytics Cloud Integration User	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Analytics Cloud Security User	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Authenticated Website	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Authenticated Website	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Contract Manager	<input checked="" type="checkbox"/>	<input type="checkbox"/>

(v) In add to page layout, we select all the checkboxes and then select Save:

(vi) We see “Field Update” added to “Account Custom Fields & Relationships”:

(vii) We select “Field Update” and look at its details as follows:

3) Creating Apex trigger on Contact Object:

(i) Now, we will create trigger on the Account object. We search “Accounts” in quick find bar and go to triggers:

The screenshot shows the Salesforce Lightning Experience interface. At the top, there's a banner with the text "It's Better in Lightning" and "Move to Lightning Experience and give your users a productivity boost." Below the banner, the page title is "Account Custom Field Field Update". The left sidebar under "Build" has "Customize" expanded, with "Triggers" selected. The main content area shows the "Custom Field Definition Detail" for "Field Update". The "Field Information" section includes fields like "Field Label" (Field Update), "Field Name" (Field_Update), "API Name" (Field_Update__c), "Description" (Help Text), and "Data Owner". To the right, "Object Name" is set to "Account" and "Data Type" is set to "Checkbox". There are buttons for "Edit", "Set Field-Level Security", "View Field Accessibility", and "Where is this used?".

(ii) We choose “New” to create new trigger:

The screenshot shows the Salesforce Lightning Experience interface. At the top, there's a banner with the text "It's Better in Lightning" and "Move to Lightning Experience and give your users a productivity boost." Below the banner, the page title is "Contact Triggers". The left sidebar has "Lightning Experience Transition Assistant" expanded. The main content area shows the "Triggers" section with a "New" button. A message says "No triggers defined".

(iii) We paste following code for apex trigger:

```
trigger updateAccount on Contact (after insert, after update) {
    Set <String> accID = New Set <String> ();
    For (Contact con: Trigger.new) {
        if (con.AccountId != Null ) {
            accID.add (con.AccountId);
        }
    }
    if (accID.size ()> 0) {
        List <Account> upAccList = new List <Account> ();
        For (Account ac:
            [SELECT Id, Field_Update__c
             FROM Account
             WHERE id in: AccID
             AND Field_Update__c != True] {

```

```

ac.Field_Update__c = true;
UpAccList.add (ac);
}
if (upAccList.size ()> 0)
update upAccList;
}
}

```

The screenshot shows the Apex Trigger Edit screen. The trigger code is as follows:

```

trigger updateAccount on Contact (after insert, after update) {
    Set <String> accID = New Set <String> ();
    For (Contact con: Trigger.new) {
        if (con.AccountId != Null ) {
            accID.add (con.AccountId);
        }
    }
    if (accID.size ()> 0) {
        List <Account> upAccList = new List <Account> ();
        For (Account ac:
                [SELECT Id, Field_Update__c
                 FROM Account
                 WHERE id in: AccID
                 AND Field_Update__c != True]) {
            ac.Field_Update__c = true;
            UpAccList.add (ac);
        }
        if (upAccList.size ()> 0)
            update upAccList;
    }
}

```

We check that “Is Active” is selected and then save it. We see trigger is created:

The screenshot shows the Contact Triggers page. A single trigger is listed:

Action	Name	Api Version	Status	Size Without Comments	Last Modified By
Edit Del	updateAccount	54.0	Active	741	Omkar Sinha, 4/22/2022, 10:58 AM

3) Creating a Contact record:

Now we will be creating Contact records from the Account object.

(i) Firstly, we click on the Accounts object:

The screenshot shows the Accounts page in the Salesforce interface. The navigation bar at the top has 'Accounts' selected.

The screenshot shows the Salesforce interface with the 'Content' tab selected. The 'Recent Items' section lists 'Omkar Sinha'. Below it are links for 'Recycle Bin', 'Accounts', 'Individuals', and 'Invoices'. A message at the top says 'Use the links below to quickly navigate to a tab. Alternatively, you can add a tab to your display to better suit the way you work.' A 'View' dropdown is set to 'All Tabs'. Buttons for 'Add Tabs to Your Default Display' and 'Customize My Tabs' are visible.

(ii) We see our pre-existing record for Account. We notice that the “Contacts” object has no records.

The screenshot shows the 'Account Detail' page for 'Omkar Sinha'. The page includes fields for Account Owner (Omkar Sinha), Account Name (Omkar Sinha), Parent Account, Account Number, Account Site, Type (Technology Partner), Industry (Technology), Annual Revenue (\$100,000), Field Update, Billing Address, Customer Priority, SLA Expiration Date, Number of Locations, Active status, Created By (Omkar Sinha), Description, and Custom Links (Billing). The 'Sharing' and 'Include Offline' buttons are at the top right. Below the main table, there is a 'Contacts' section with 'New Contact' and 'Merge Contacts' buttons, and a message stating 'No records to display'.

(iii) We click on “New Contact”, to create a new contact. We fill out the details of a particular contact “Zeus Martin” in this case and select save:

The screenshot shows the 'Contact Edit' page for a new contact. The 'Contact Information' section includes fields for Contact Owner (Omkar Sinha), Salutation (Mr.), First Name (Zeus), Last Name (Martin), Account Name (Omkar Sinha), Title, Department, Birthdate, Reports To, Phone (332) 201-9462, Home Phone (332) 201-9462, Mobile (332) 201-9462, Other Phone, Fax, Email (zmartin@stevens.edu), Assistant, and Asst. Phone. The 'Save & New' button is highlighted in blue. A note at the top says 'Contacts not associated with accounts are private and cannot be viewed by other users or included in reports.' A legend indicates that red asterisks (*) denote required information.

Address Information

Mailing Street: 10 Bleeker Street, Apt-2

Mailing City: Jersey City

Mailing State/Province: New Jersey

Mailing Zip/Postal Code: 07307

Mailing Country: United States

Other Street: [Empty]

Other City: [Empty]

Other State/Province: [Empty]

Other Zip/Postal Code: [Empty]

Other Country: [Empty]

Additional Information

Languages: [Empty]

Level: --None--

Description Information

Description: [Empty]

Buttons: Save, Save & New, Cancel

(iv) We now again go to our original “Omkar Sinha” account. We see that “Field Update” field on the Account Object is Checked and “Zeus Martin” record is added in the Contacts as a new record:

Account Detail		Edit Delete Sharing Include Offline	
Account Owner	Omkar Sinha [Change]	Rating	Warm
Account Name	Omkar Sinha [View Hierarchy]	Phone	
Parent Account		Fax	
Account Number		Website	
Account Site		Ticker Symbol	
Type	Technology Partner	Ownership	
Industry	Technology	Employees	400
Annual Revenue	\$100,000	SIC Code	
Field Update	<input checked="" type="checkbox"/>	Shipping Address	
Billing Address		SLA	
Customer Priority		SLA Serial Number	
SLA Expiration Date		Upsell Opportunity	
Number of Locations		Last Modified By	Omkar Sinha, 4/22/2022, 12:44 PM
Active			
Created By	Omkar Sinha, 4/22/2022, 12:27 PM		
Description			
Custom Links	Billing		
Edit Delete Sharing Include Offline			

Contacts		New Contact Merge Contacts		
Action	Contact Name	Title	Email	Phone
Edit Del	Zeus Martin		zmartin@stevens.edu	(332) 201-9462