

# Lab 5: CS 524

In this assignment, you will learn use AWS Lambda and make microservice.

AWS Lambda is a compute service that lets you run code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service, all with zero administration. Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring and logging.

A microservice aims to deliver a different set of capabilities and focuses on a specific domain. For instance, you can have a component responsible for getting thumbnails and pictures from a database and returning them to the web or mobile application. microservice aims to deliver a different set of capabilities and focuses on a specific domain. For instance, you can have a component responsible for getting thumbnails and pictures from a database and returning them to the web or mobile application. microservice aims to deliver a different set of capabilities and focuses on a specific domain. For instance, you can have a component responsible for getting thumbnails and pictures from a database and returning them to the web or mobile application.

- a. Once registered for AWS, now go into AWS Console



## Sign in

### ☒ Root user

Account owner that performs tasks requiring unrestricted access. [Learn more](#)

### ☐ IAM user

User within an account that performs daily tasks. [Learn more](#)

Root user email address

kmodi5@stevens.edu

Next

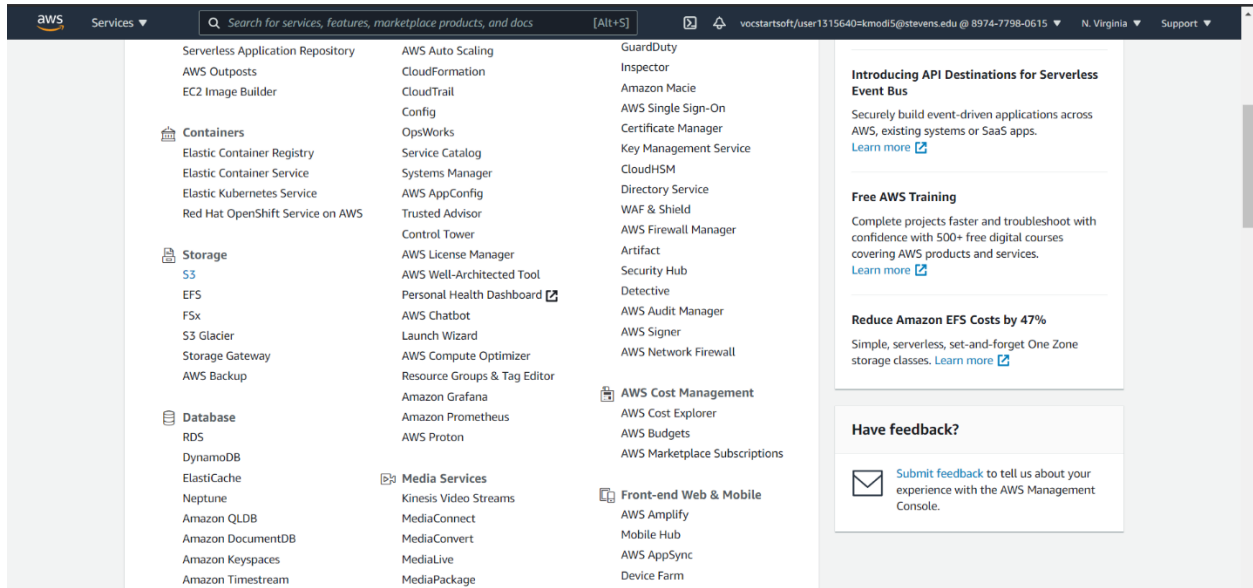
By continuing, you agree to the [AWS Customer Agreement](#) or other agreement for AWS services, and the [Privacy Notice](#). This site uses essential cookies. See our [Cookie Notice](#) for more information.

New to AWS?

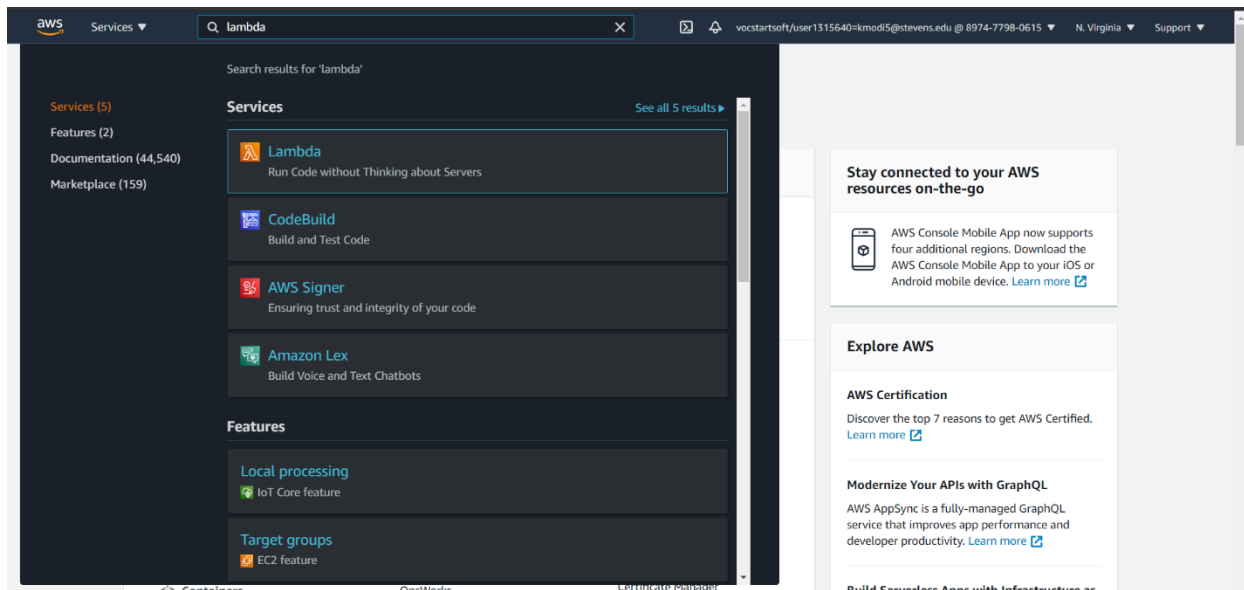
Create a new AWS account



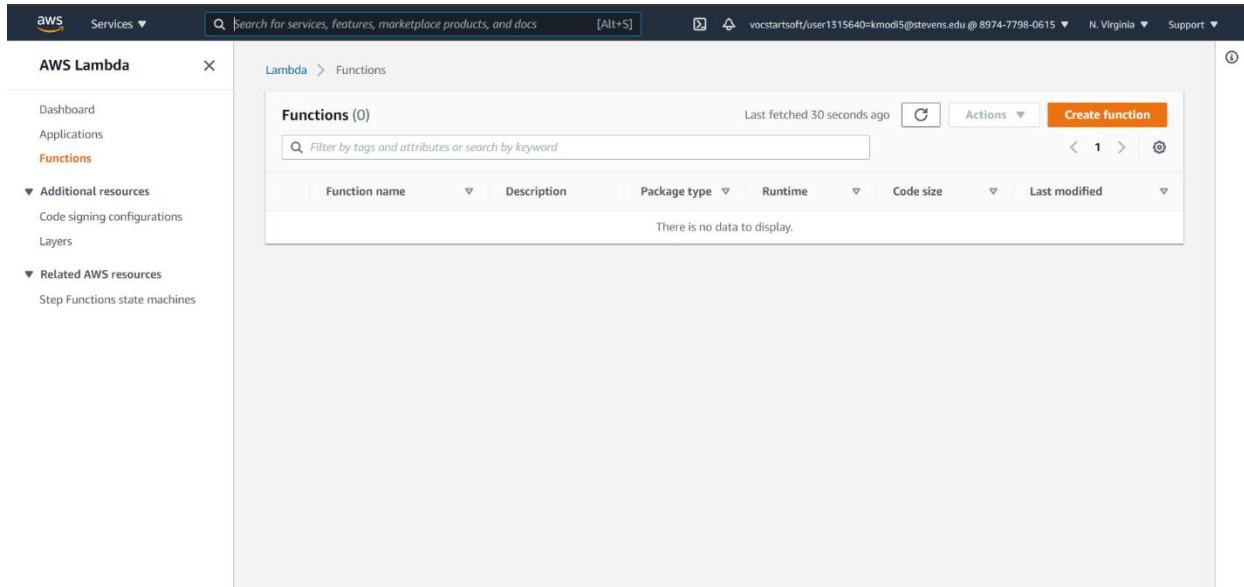
## b. Go into AWS Console and Launch a S3 Bucket



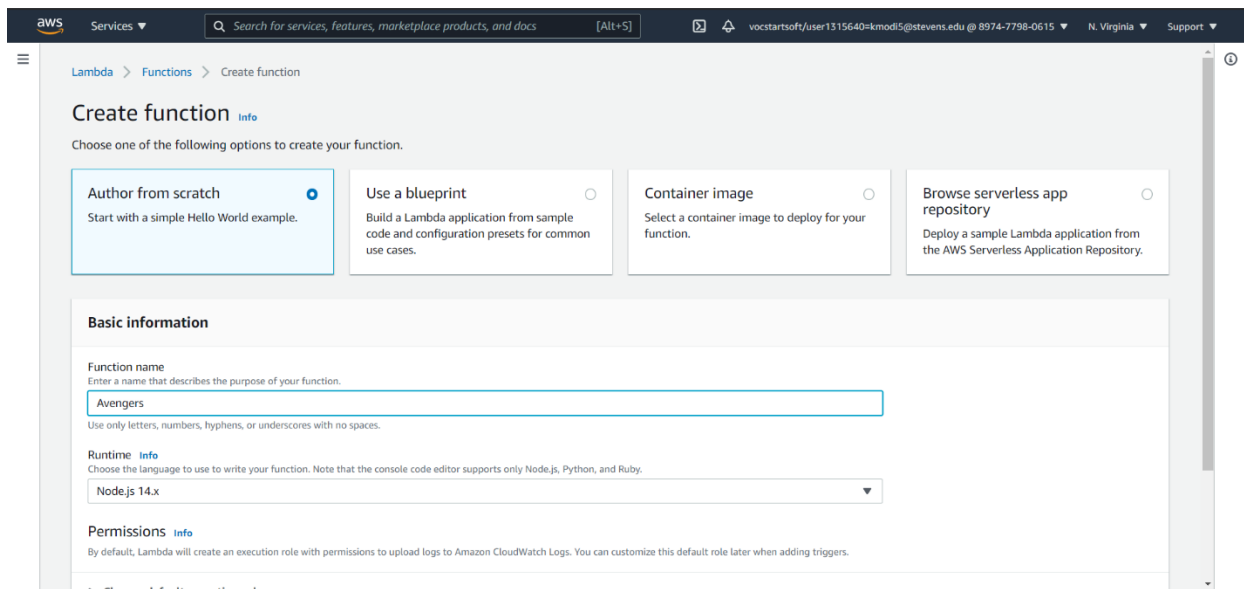
## c. Open Lambda



#### d. Create function



#### e. We are going to use Author from Scratch option( choice your preference)



## f. Review and create function

The screenshot shows the 'Create function' wizard in the AWS Lambda console. The 'Basic information' section is active, showing the function name 'Avengers' and the runtime 'Node.js 14.x'. The 'Permissions' section indicates that a default execution role will be created. The 'Advanced settings' section is collapsed. The 'Create function' button is visible at the bottom right.

**Author from scratch**  
Start with a simple Hello World example.

**Use a blueprint**  
Build a Lambda application from sample code and configuration presets for common use cases.

**Container image**  
Select a container image to deploy for your function.

**Browse serverless app repository**  
Deploy a sample Lambda application from the AWS Serverless Application Repository.

**Basic information**

**Function name**  
Enter a name that describes the purpose of your function.  
  
Use only letters, numbers, hyphens, or underscores with no spaces.

**Runtime** [Info](#)  
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

**Permissions** [Info](#)  
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

[Change default execution role](#)

[Advanced settings](#)

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

## g. Now add a trigger

The screenshot shows the 'Avengers' function overview page in the AWS Lambda console. The 'Function overview' section displays the function name 'Avengers', the number of layers '(0)', and the function ARN. The 'Code source' section shows the 'Test' button and the 'Deploy' button. The 'Test' button is highlighted, and the 'Deploy' button is disabled. The 'Changes deployed' status is shown at the bottom right.

**Successfully created the function Avengers.** You can now change its code and configuration. To invoke your function with a test event, choose "Test".

**Avengers** [Throttle](#) [Copy ARN](#) [Actions](#)

**Function overview** [Info](#)

**Avengers** [Layers](#) (0)

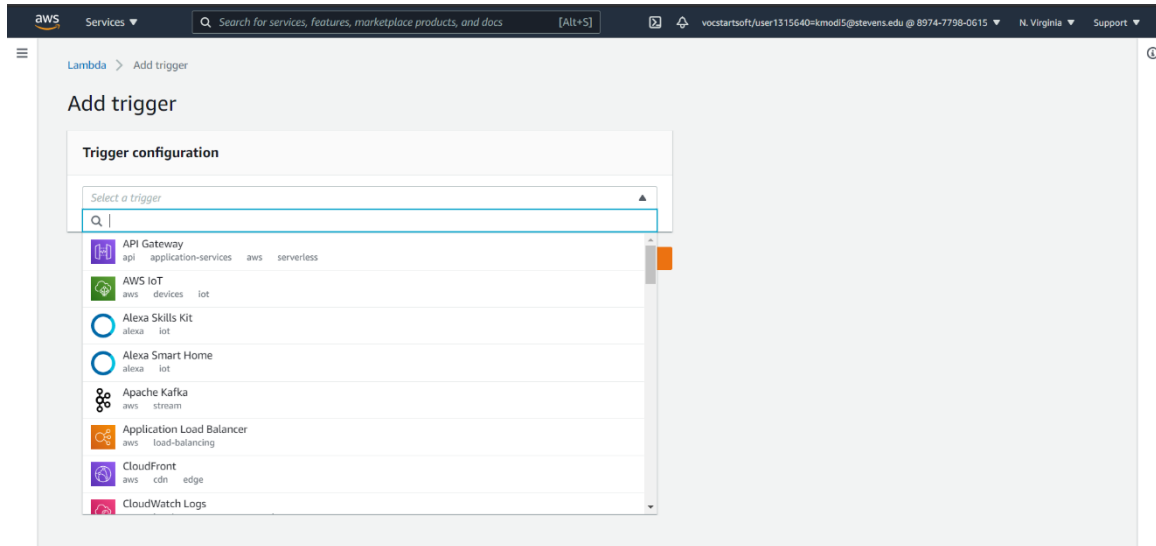
[Add trigger](#) [Add destination](#)

**Code source** [Info](#) [Upload from](#)

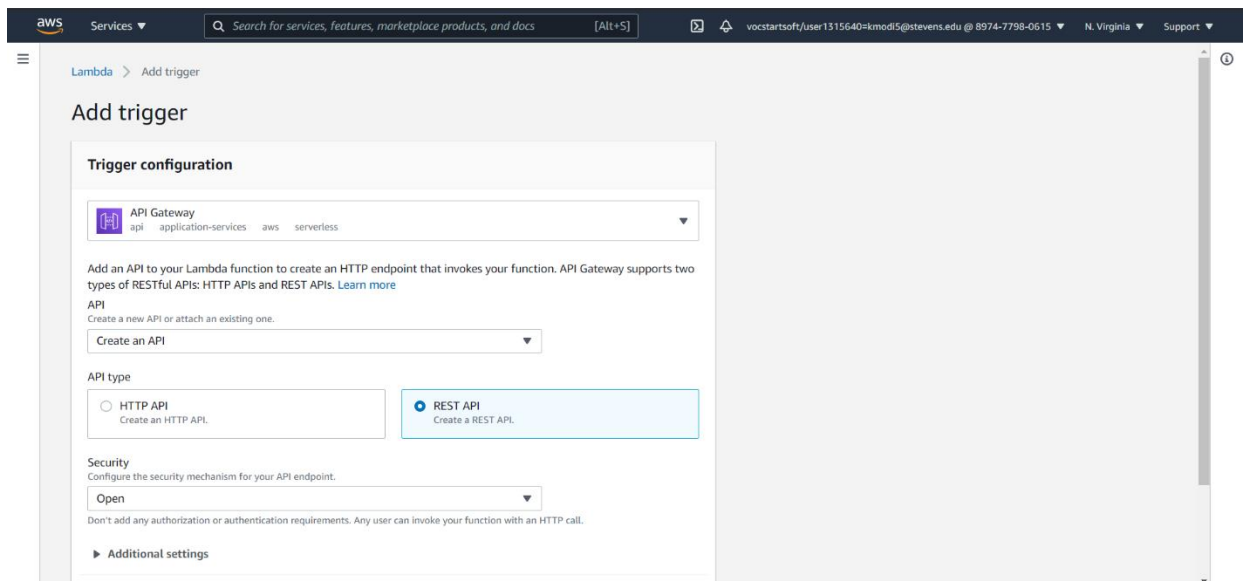
[File](#) [Edit](#) [Find](#) [View](#) [Go](#) [Tools](#) [Window](#) [Test](#) [Deploy](#) [Changes deployed](#)

[Go to Anything \(Ctrl-P\)](#) [Help](#) [Settings](#)

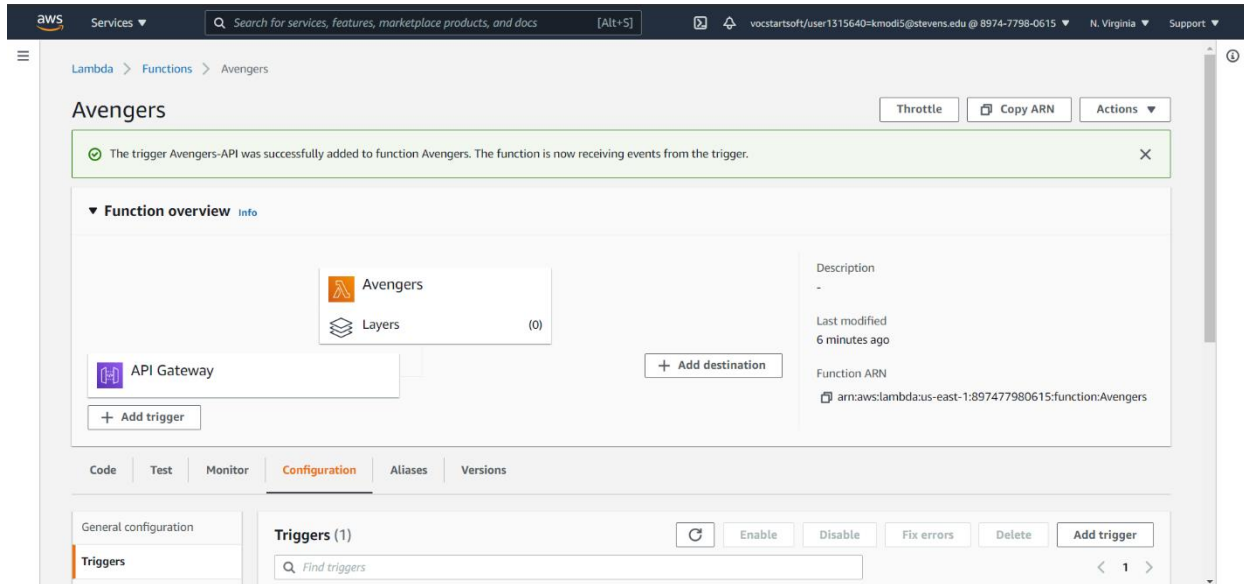
h. Now we will use API Gateway to expose the URL



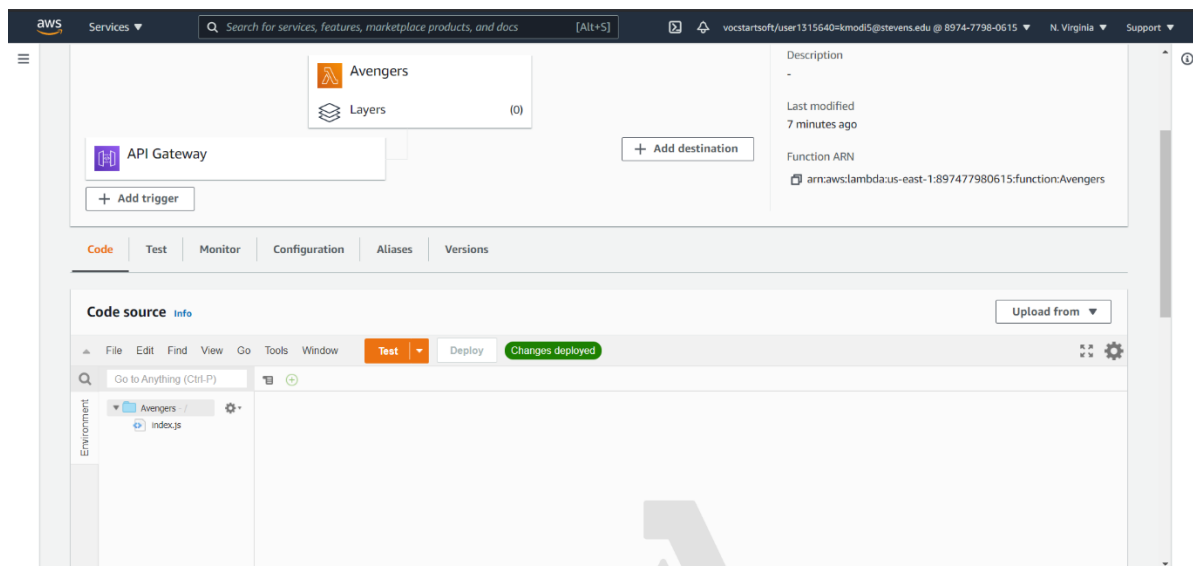
i. Select REST API and security for our API



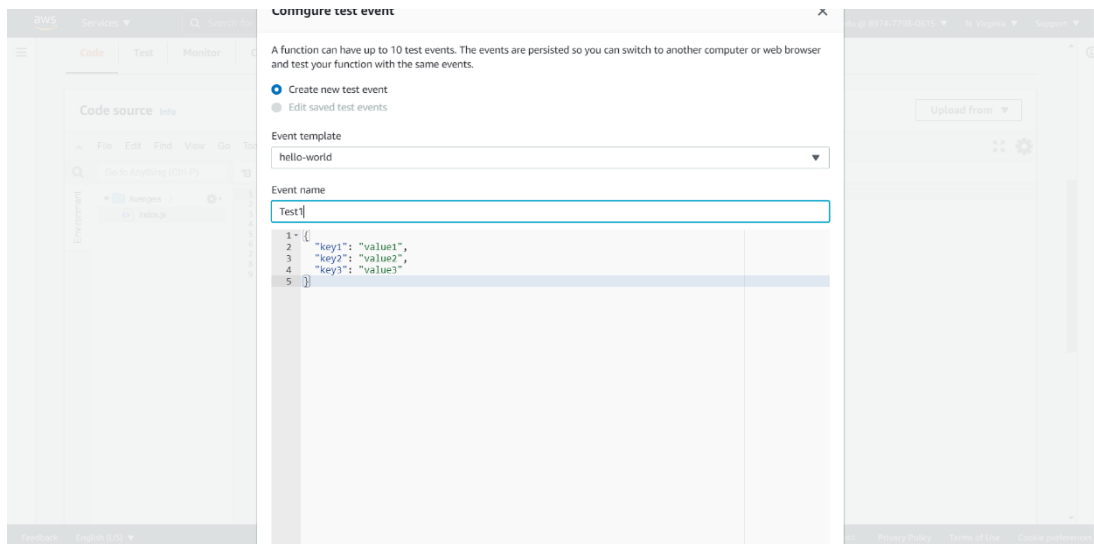
j. API has been created



k. Add your code or upload a ZIP



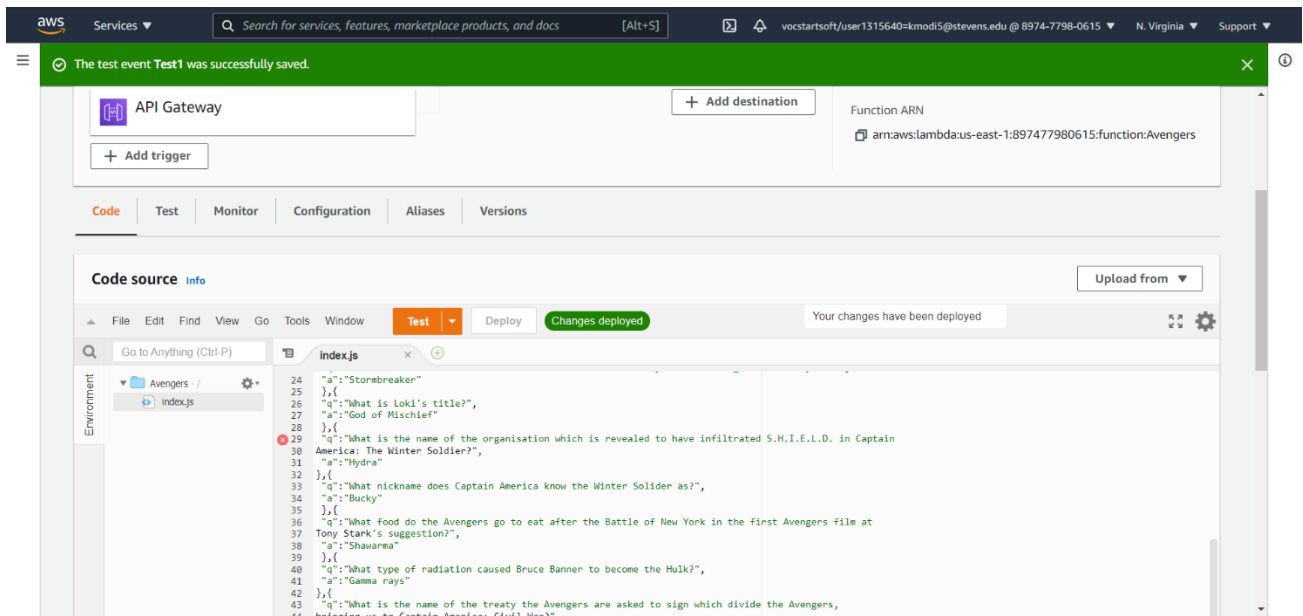
## I. Testing the service



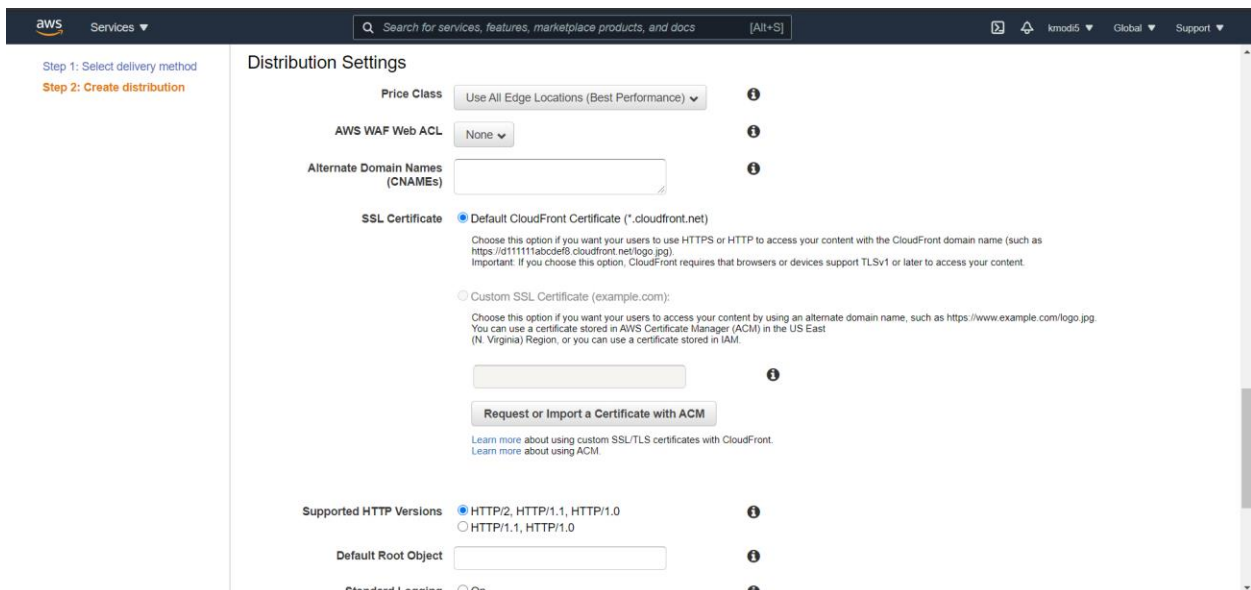
## m. The Lambda works



n. Change the code and Deploy it

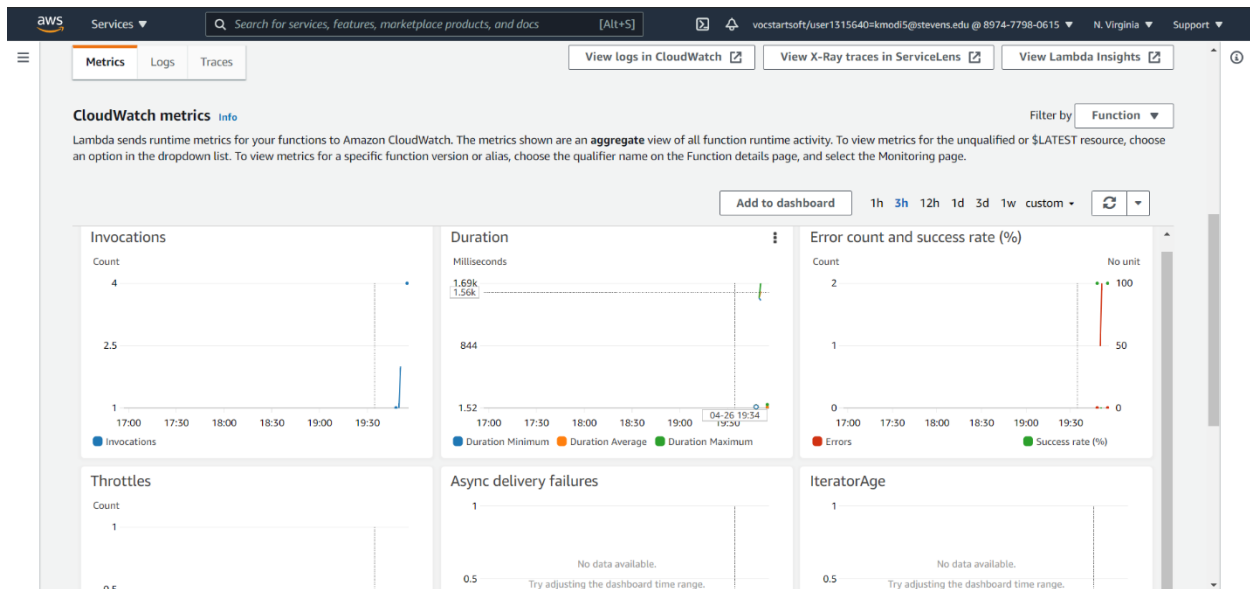


o. Now the lambda function will randomly cycle through the questions and answer from our Marvel Encyclopedia!





p. Use functions page and go to monitor to access Cloudwatch dashboard and logs



The screenshot shows the AWS CloudWatch Logs Insights dashboard. The top navigation bar includes the AWS logo, a search bar, and user information. The main header has tabs for Code, Test, Monitor, Configuration, Aliases, and Versions. Below the header, there are links to 'View logs in CloudWatch', 'View X-Ray traces in ServiceLens', and 'View Lambda Insights'. The dashboard title is 'CloudWatch Logs Insights' with an 'info' link. A filter dropdown is set to 'Function'. The dashboard contains two tables: 'Recent invocations' and 'Most expensive invocations in GB-seconds (memory assigned \* billed duration)'. The 'Recent invocations' table lists the most recent and most expensive function invocations across all function activity. The 'Most expensive invocations in GB-seconds (memory assigned \* billed duration)' table lists the most expensive function invocations across all function activity.

#	Timestamp	RequestID	LogStream	DurationInMS	BilledDurationInMS	MemorySetInMB
1	2021-04-26T19:55:23.116Z	b078ff91-9283-4db0-9881-0fbdf913cc1	2021/04/26/[SLATEST]f7ea76fcccba346f28f5eb8071ff57f85	1.52	2	128
2	2021-04-26T19:55:22.487Z	8c249e13-1149-49ee-9893-df534deafe7b	2021/04/26/[SLATEST]f7ea76fcccba346f28f5eb8071ff57f85	2.08	3	128
3	2021-04-26T19:55:21.576Z	2a5e79dd-728f-425b-aa4c-db2004877c15	2021/04/26/[SLATEST]f7ea76fcccba346f28f5eb8071ff57f85	2.08	3	128
4	2021-04-26T19:55:19.067Z	7a3ab283-16e5-4b0e-a895-8c883d23556d	2021/04/26/[SLATEST]f7ea76fcccba346f28f5eb8071ff57f85	46.39	47	128
5	2021-04-26T19:51:40.491Z	52f33c94-99c2-4765-89df-312c6e9527ff	2021/04/26/[SLATEST]57d4a34852c948948d1af2ffd336c281	1456.99	1457	128
6	2021-04-26T19:51:07.351Z	7cc48d48-d10a-4cd7-b3b4-e563a139b4a9	2021/04/26/[SLATEST]57d4a34852c948948d1af2ffd336c281	1686.66	1687	128
7	2021-04-26T19:50:06.727Z	8dc16865-accd-45c5-8f5c-6f2dd9ec67b	2021/04/26/[SLATEST]72f8dad1fb084849bab89636447a606f	1491.04	1492	128

#	Timestamp	RequestID	LogStream	BilledDurationInMS	MemorySetInMB	BilledDuration
---	-----------	-----------	-----------	--------------------	---------------	----------------

Log streams	Metric filters	Subscription filters	Contributor Insights	Tags
<div> <b>Log streams (10)</b> <span>🔄</span> <span>Delete</span> <span>Create log stream</span> <span>Search all</span> </div> <div> <input type="text" value="Filter log streams or try prefix search"/> <span>&lt; 1 &gt;</span> <span>⚙️</span> </div>				
<input type="checkbox"/>	Log stream	▼	Last event time	▼
<input type="checkbox"/>	2021/04/27/[\$LATEST]b3d7ac7b4cf240269bcc72236ad584bd		2021-04-27 14:39:01 (UTC-04:00)	
<input type="checkbox"/>	2021/04/26/[\$LATEST]afe072ba40f44a258568d82549260612		2021-04-26 16:57:37 (UTC-04:00)	
<input type="checkbox"/>	2021/04/26/[\$LATEST]47cd33e5f0424d6781028f8f19bfde60		2021-04-26 16:57:17 (UTC-04:00)	
<input type="checkbox"/>	2021/04/26/[\$LATEST]cfe19c34e8414fb4b35c3d4945a4677c		2021-04-26 16:56:52 (UTC-04:00)	
<input type="checkbox"/>	2021/04/26/[\$LATEST]22b483996a7648a3a9819cd5b5858c9d5		2021-04-26 16:53:39 (UTC-04:00)	
<input type="checkbox"/>	2021/04/26/[\$LATEST]ed80be1001b34bf0a74c72a2ec8a7b93		2021-04-26 16:53:20 (UTC-04:00)	
<input type="checkbox"/>	2021/04/26/[\$LATEST]fea76fecbba346f28f5eb8071ff57f85		2021-04-26 15:55:23 (UTC-04:00)	
<input type="checkbox"/>	2021/04/26/[\$LATEST]57d4a34852c948948d1af2ffd336c281		2021-04-26 15:51:40 (UTC-04:00)	
<input type="checkbox"/>	2021/04/26/[\$LATEST]72f8dad1fb084849bab89636447a606f		2021-04-26 15:50:07 (UTC-04:00)	
<input type="checkbox"/>	2021/04/26/[\$LATEST]969ca108db914b039367f0b751ab7351		2021-04-26 15:48:31 (UTC-04:00)	

Now we have deployed our new code we can view the logs, the requests and any exceptions to our function. Note since this is a microservice, so if it is a part of a website only compute used for microservice is counted towards the use.

Now we will query some results using Logs insight:

CloudWatch > CloudWatch Logs > Logs Insights

Select log group(s)

/aws/lambda/Avengers
Clear

1 fields @message  
2 .....

Run query
Save
History

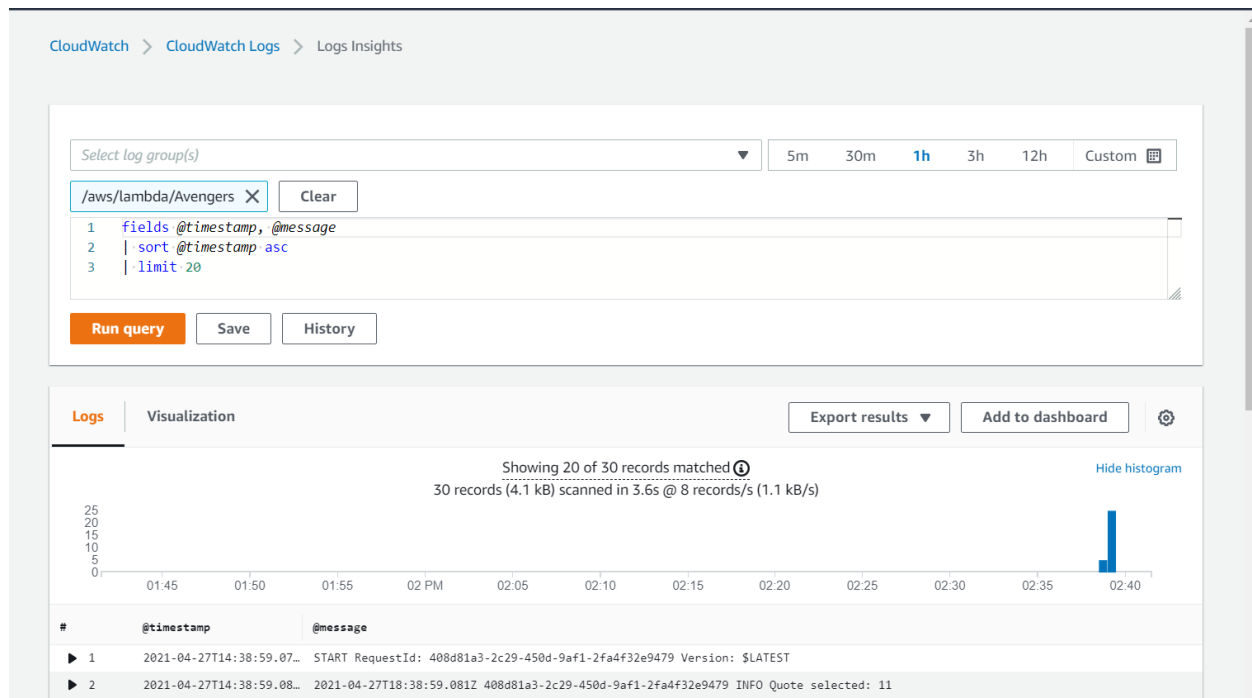
Logs
Visualization

Export results
Add to dashboard
⚙️

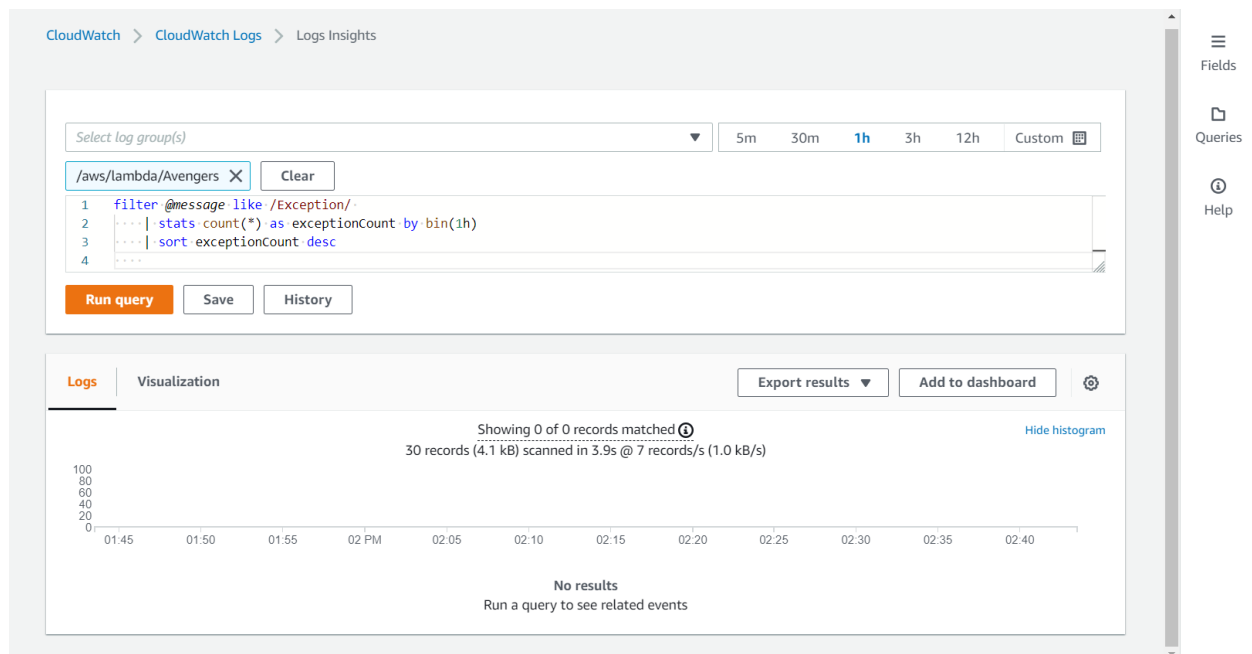
Showing 30 of 30 records matched ⓘ  
30 records (4.1 kB) scanned in 3.8s @ 7 records/s (1.1 kB/s)

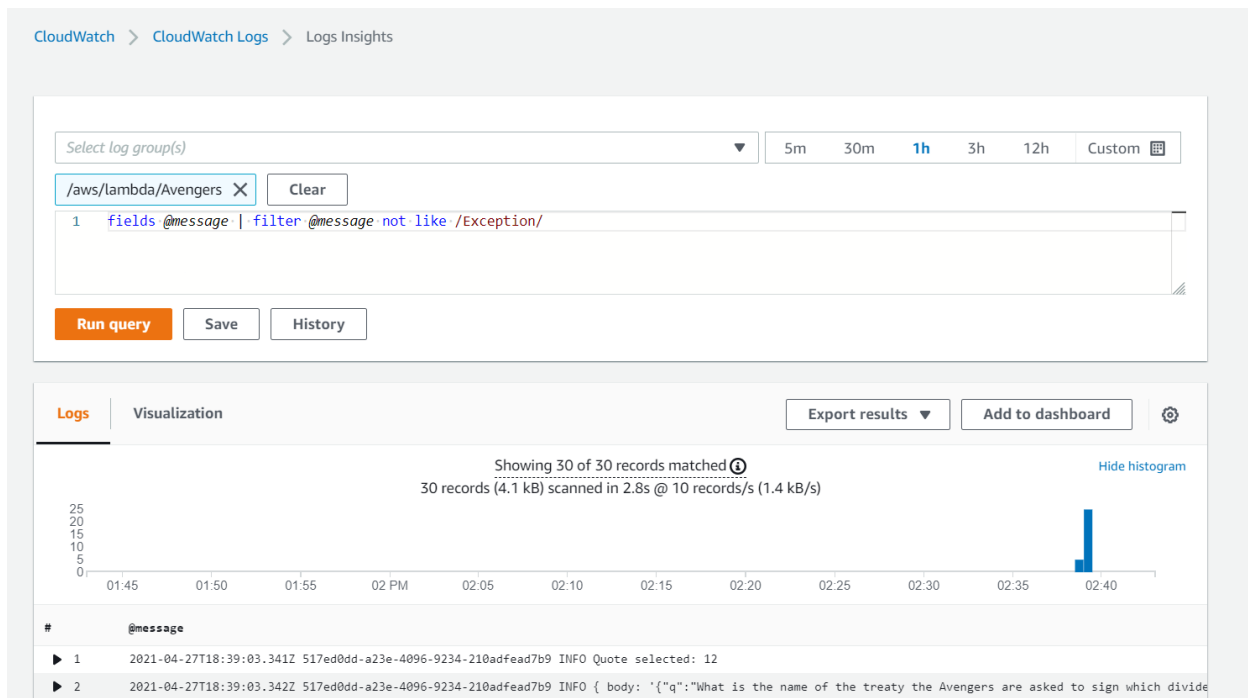
Hide histogram

# @message  
▶ 1 2021-04-27T18:39:03.341Z 517ed0dd-a23e-4096-9234-210adfead7b9 INFO Quote selected: 12  
▶ 2 2021-04-27T18:39:03.342Z 517ed0dd-a23e-4096-9234-210adfead7b9 INFO { body: '{"q":"What is the name of the treaty the Avengers are asked to sign which divide the Avengers,bringing u...  
▶ 3 END RequestId: 517ed0dd-a23e-4096-9234-210adfead7b9  
▶ 4 REPORT RequestId: 517ed0dd-a23e-4096-9234-210adfead7b9 Duration: 1.57 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 64 MB  
▶ 5 START RequestId: 517ed0dd-a23e-4096-9234-210adfead7b9 Version: \$LATEST  
▶ 6 2021-04-27T18:39:02.657Z 025e0e23-2b60-4620-8277-8afc3136cb61 INFO Quote selected: 9



So luckily we had no exceptions here:





By using the microservices we have learned that different code sections can be easily deployed and individually managed. Thus, we can conclude that moving from monolithic to microservice architecture it allows:

- Flexibility
- Scalability
- Isolated functions
- Faster implementation
- Efficient use of resources
- Lower deployment time
- Security and Reliability (Since it is modular)
- Monitoring (Metrics and Logs)

Some particular features like logs, queries also allows very granular control and metrics and makes implementation and deployment a joy.