

AuburnHacks

Introduction to AWS



Cloud Background



- **Resilient:** redundancy and failover systems
- **On-demand:** resources whenever you need them
- **Elastic:** able to scale resources on demand

What is AWS?

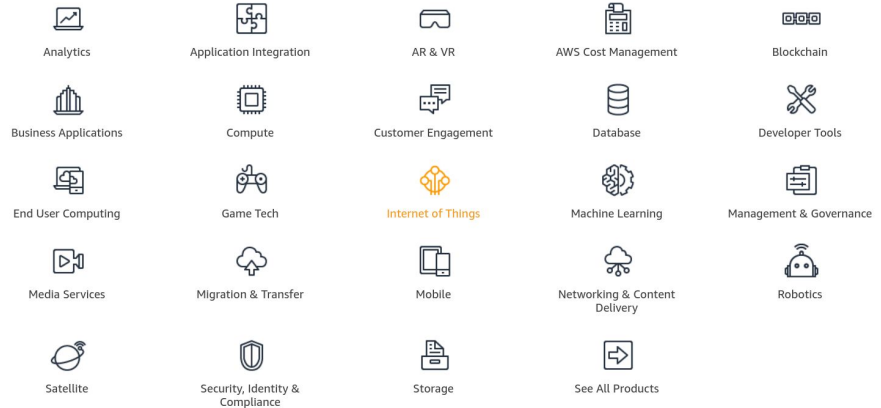


Amazon Web Services (AWS) is one of the largest cloud providers and offer many services for your cloud solutions

There is a lot of documentation and numerous tutorials for you to get started on something today.

Some of their clients include big names like Netflix, Airbnb, and Adobe

Explore Our Products





Getting started

1. Sign up for a new account:

<https://portal.aws.amazon.com/billing/signup#/start>

Make sure to write down or save your Access key ID and secret access key

2. Download the code that is being used in this tutorial:

https://github.com/auburnhacks/hacker-resources/tree/master/aws_techtalk

DynamoDB



1. Click 'Create table'
2. Give your table a descriptive name
3. Name the Partition key 'id'
4. Name the sort key 'runtime'
5. Make sure to make both of these Numbers

Create DynamoDB table

DynamoDB is a schema-less database that only requires a table name and primary key. The table's two attributes that uniquely identify items, partition the data, and sort data within each partition.

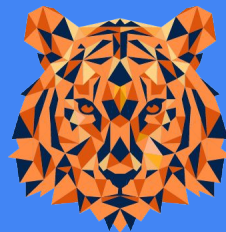
Table name*	<input type="text" value="descriptive-name"/>	
Primary key*	Partition key	
	<input type="text" value="id"/>	<input type="text" value="Number"/>
	<input checked="" type="checkbox"/> Add sort key	
	<input type="text" value="runtime"/>	<input type="text" value="Number"/>



Adding items to DynamoDB

1. You can either add an item via the console or through the command line
2. First install the AWS CLI:
`sudo apt-get install awscli` (or with your other installer if not on linux)
3. Configure:
`aws configure`
Then enter your access key ID, secret, region name, and output format
4. Write the items to the table
`aws dynamodb batch-write-item --request-items file://items.json`

Lambda



1. Click 'Create function'
2. Give your function a descriptive name
3. Select language
4. Select 'Create a custom role'. A new window will pop up taking you to the IAM console

Author from scratch [Info](#)

Name

Runtime
You can select a supported AWS Lambda runtime or provide your own runtime as part of the function deployment package or Lambda layer after creating the function.

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 can use an existing role with this function. Lambda must be able to assume this role, and the role must have Amazon CloudWatch Logs permissions.

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



Create new execution role

1. Give your role a descriptive name
2. Edit the policy document
3. Add this in the 'Statement' array:

```
{  
    "Effect": "Allow",  
    "Action": [  
        "dynamodb:Scan"  
    ],  
    "Resource": "your-arn-here"  
}
```

(You can find your resource arn in the overview section of your table)

Role Lambda execution role permissions

Description

IAM Role

Role Name

▼ Hide Policy Document [Edit](#)

```
{  
    "Effect": "Allow",  
    "Action": [  
        "dynamodb:Scan"  
    ],  
    "Resource": "your-arn-here"  
}
```


Lambda



1. Select APIGateway from the left
2. For 'Security' just select Open for now
3. Give your API a descriptive name
4. The rest can stay default for now
5. Click 'Add', then click 'Save' at the top
6. Click on the blue link to open up your new API in APIGateway

Configure triggers

We'll set up an API Gateway endpoint with a [proxy integration type](#) (learn more about the [input](#) and [output](#) format). Any method (GET, POST, etc) the [Amazon API Gateway console](#).

API

Pick an existing API, or create a new one.

Create a new API

Security

Configure the security mechanism for your API endpoint.

Open

Warning: Your API endpoint will be publicly available and can be invoked by all users.

▼ Additional settings

API name

Enter a name to uniquely identify your API.

descriptive-name

Deployment stage

The name of your API's deployment stage.

default

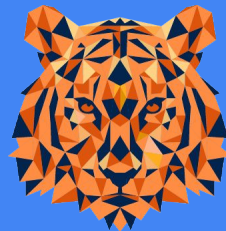
CORS

To enable cross-origin resource sharing (CORS) for a proxy integration, you must add Access-Control-Allow-Origin: <domain_name> to the output headers. <dc

☐ Enable metrics and error logging

Emit latency/error metrics and logs at the ERROR level. Metrics and logs are charged at the standard CloudWatch rates. Go to the [API Gateway console](#) to configure more advanced features such as request/response traces and custom access logging.

APIGateway



1. Click on 'Actions' then 'Create Resource'
2. Name your resource 'score'
3. Make the 'Resource Path' '{score}'
4. Click 'Create Resource'

Resources Actions **New Child Resource**

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

Configure as [proxy resource](#)

Resource Name*

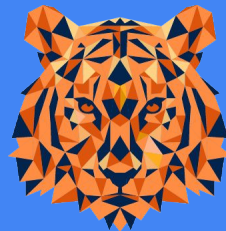
Resource Path*

You can add path parameters using brackets. For example, the resource catches all requests to its sub-resources. For example, it works for /hackathonTest resource.

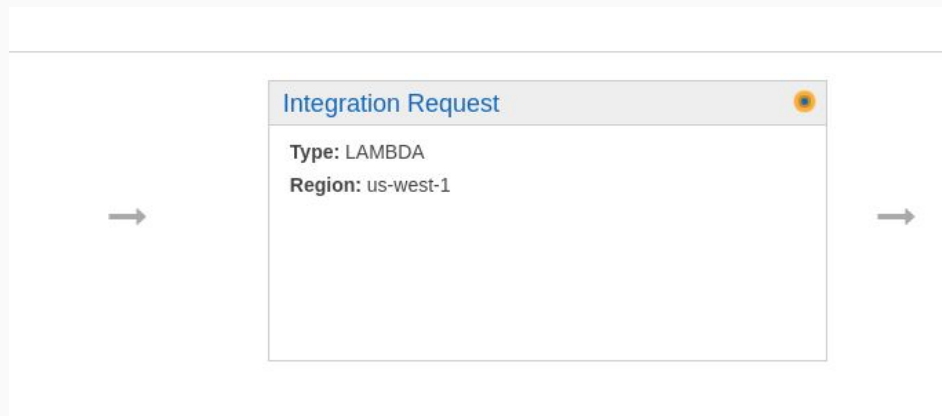
Enable API Gateway CORS

* Required

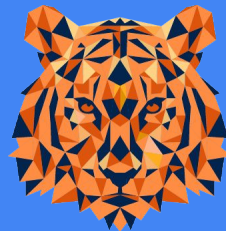
APIGateway



1. Click on 'Actions' then 'Create Method' and select 'GET'
2. In 'Lambda Function' type the name of your Lambda function
3. Click Save and then OK
4. Next, click 'Integration Request'



APIGateway



1. Click on 'URL Path Parameters'
2. Type 'score' under Name
3. Type the path that you want to capture the path parameter from
'method.request.path.score'
4. Click the check mark

Mock ⓘ
AWS Service ⓘ
VPC Link ⓘ

Use Lambda Proxy integration ☐ ⓘ

Lambda Region us-west-1 ✎

Lambda Function hackathonTest ✎

Execution role ✎

Invoke with caller credentials ☐ ⓘ

Credentials cache Do not add caller credentials to cache key ✎

Use Default Timeout ☒ ⓘ

▼ URL Path Parameters

Name	Mapped from ⓘ
score	method.request.path.score

APIGateway



1. Click on 'Mapping Templates'
2. Click 'Add mapping template'
3. Type 'application/json' and click the check mark
4. Click 'Yes, secure this integration'
5. Next, click on 'Generate template:' and select 'Method Request passthrough'
6. Click Save

▼ Mapping Templates

- Request body passthrough**
- ☒ When no template matches the request Content-Type header ⓘ
 - ☐ When there are no templates defined (recommended) ⓘ
 - ☐ Never ⓘ

Content-Type

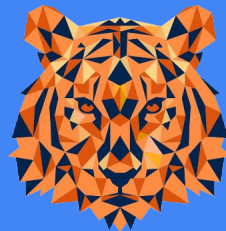
application/json

Add mapping template

Generate template: **Method Request passthrough ▼**

```
1  ## See http://docs.aws.amazon.com/apigateway/latest/developerguide/api-g
2  ## This template will pass through all parameters including path, query:
   to the integration endpoint via the body/payload
3  #set($allParams = $input.params())
4  {
5    "body-json" : $input.json('$'),
6    "params" : {
7      #foreach($type in $allParams.keySet())
8        #set($params = $allParams.get($type))
9        "$type" : {
10         #foreach($paramName in $params.keySet())
11           "$paramName" : "$paramName"
12         }
13       }
14     }
15  }
```

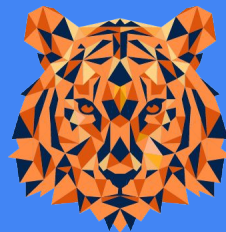
APIGateway



1. Click on 'Actions' then click on 'Deploy API'
2. Choose 'default' as the deployment stage or create a new one for testing or something
3. Click Deploy
4. Now you can click on your blue invoke URL

A screenshot of the 'Deploy API' dialog box in AWS API Gateway. The dialog has a title bar 'Deploy API' with a close button. The main content area contains a text instruction: '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.' Below this, there are two fields: 'Deployment stage' with a dropdown menu showing 'default', and 'Deployment description' with an empty text input field. At the bottom right, there are two buttons: 'Cancel' and 'Deploy'.

Code



Remember that the code for this project can be found here:

https://github.com/auburnhacks/hacker-resources/tree/master/aws_techtalk

Thanks for attending!