Build API Gateway with different stages using stage variables

Level: **Advanced**

Amazon EC2 | AWS Lambda | Identity And Access Management | Amazon Web Services | Amazon API Gateway

Lab Duration 01:15:00

Average Start time Less than a minute

Start Guided Lab →

Overview | Steps

Cloud Developer, Cloud DevOps Engineer

Compute, Networking, Serverless

Lab Steps

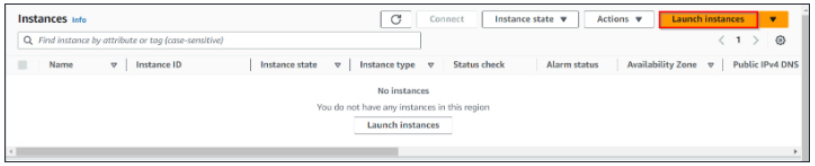
Task 1: Sign in to AWS Management Console

1. Click on the **Open Console** button, and you will get redirected to AWS Console in a new browser tab.
2. On the AWS sign-in page,
   - Leave the Account ID as default. Never edit/remove the 12-digit Account ID present in the AWS Console. Otherwise, you cannot proceed with the lab.
   - Now copy your **Username** and **Password** in the Lab Console to the **IAM Username and Password** in AWS Console and click on the **Sign-in** button.
3. Once Signed In to the AWS Management Console, make the default AWS Region as **US East (N. Virginia) us-east-1.**
4. Select Maybe later in New AWS Console Home page pop-up

Task 2: Launching an EC2 Instance

In this task, we are going to allow the user to create an EC2 instance, which will be used later in the lab to run CLI commands and perform actions related to the API Gateway and Lambda functions.

1. Make sure you are in the US East **(N. Virginia) us-east-1** Region.
2. Navigate to EC2 by clicking on the **Services** menu in the top, then click on **EC2** in the **Compute** section.
3. Navigate to **Instances** on the left panel and click on **Launch instances**



Need help?

How to use Hands on Lab

Troubleshooting Lab

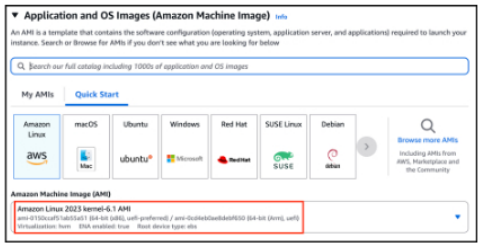
FAQs

Submit Feedback | Share

Help

4. Name : Enter **MyEC2CLIInstance**

5. **For Amazon Machine Image (AMI):** Search for **Amazon Linux 2023 AMI** in the search box and click on the **Select** button.

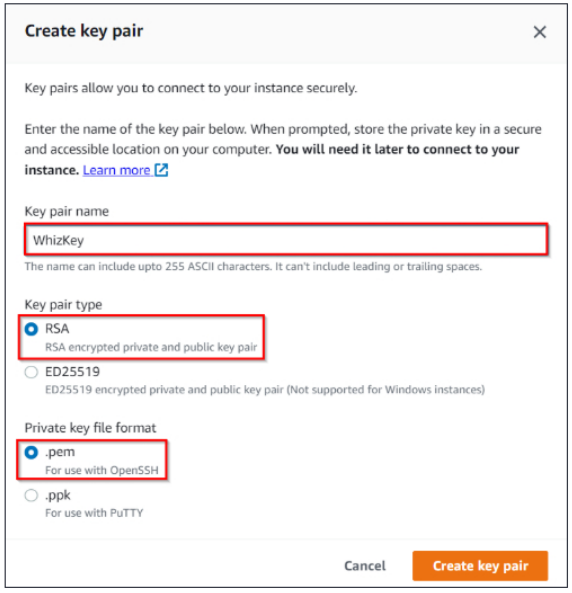


6. **For Instance Type:** select ***t2.micro***

7. **For Key pair:** Select **Create a new key pair** Button

- Key pair name: **WhizKey**
- Key pair type: **RSA**
- Private key file format: **.pem**

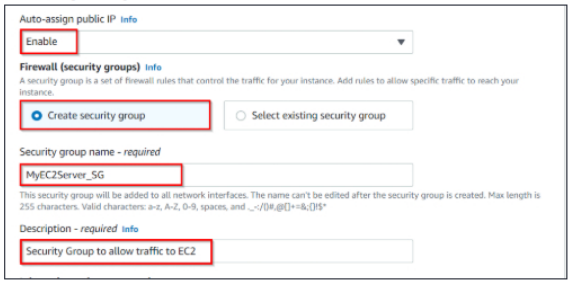
8. Select **Create key pair** Button.



9. In Network Settings, Click on **Edit** Button:

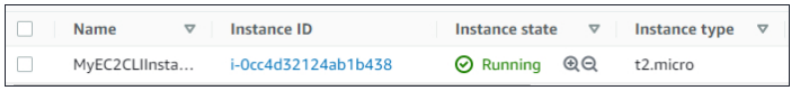
- Auto-assign public IP: **Enable**
- Select **Create new Security group**
- Security group name : Enter **MyEC2Server_SG**

- Description : Enter **Security Group to allow traffic to EC2**



- Check Allow SSH from and Select Anywhere from dropdown
- To add **SSH**,
    - Choose Type: **SSH**
    - Source: Select **Anywhere**

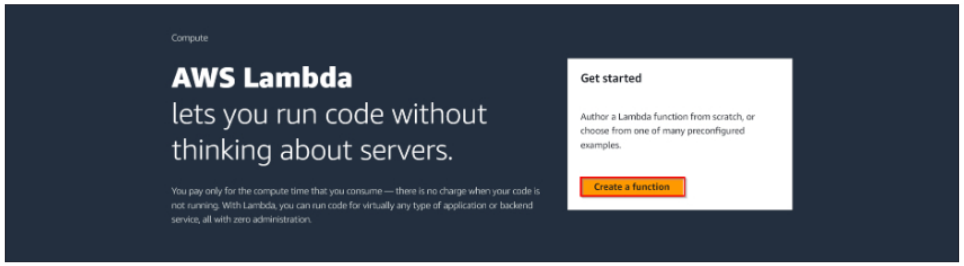
10. Under **Advanced Settings,** choose IAM Instance profile as **EC2_Role_profile<RANDOM_NUMBER>**.
11. Keep rest thing Default and Click on **Launch Instance** Button.
12. Select **View all Instances** to View the Instance you created.
13. **Launch Status:** Your instances are now launching, Navigate to **Instances** page from left menu and wait the status of the EC2 Instance changes to running and health check status changes to **2/2 checks passed.**



Task 3: Create two Lambda Functions

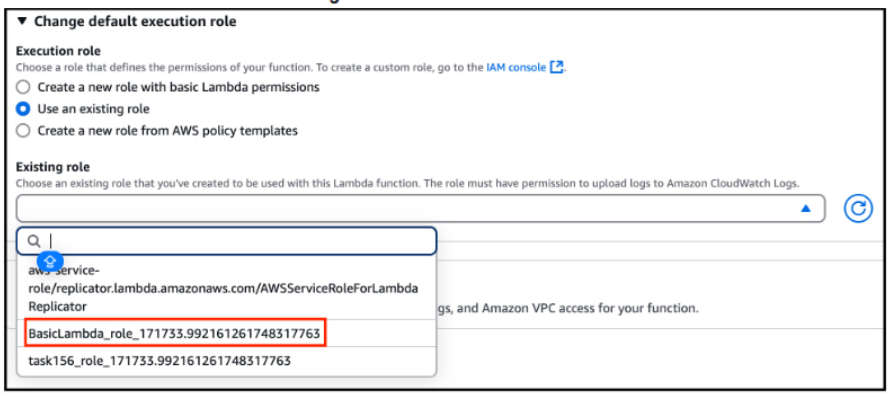
In this task, we are going to create two Lambda functions, namely ProductionLambda and TestingLambda. These functions will be integrated with the API Gateway in subsequent tasks.

1. Navigate to the **Services** menu at the top, then click on **Lambda** under the **Compute** section.
2. Click on **Create a function**

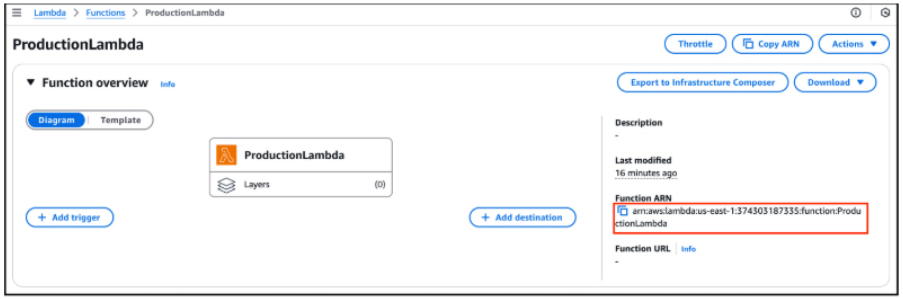


- Select **Author from Scratch**
- Function Name : Enter **ProductionLambda**

- Runtime : Select **Python 3.12**
- Expand Change default execution role
- Execution Role : Click on **Use an existing role** and choose **BasicLambdaRole**



3. Click on the **Create function**
4. Once the Lambda Function is created successfully **copy the Function ARN.**



5. Function code : Replace the existing code with the below and then click on **Deploy**.

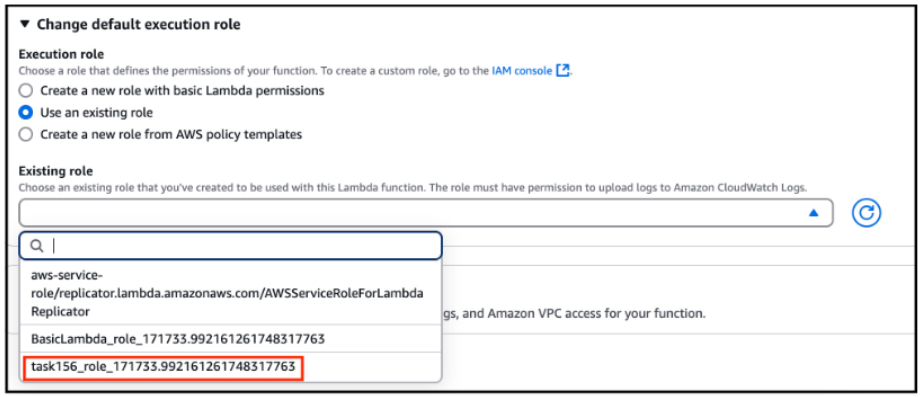
```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Message from production Lambda.')
    }
```

6. Repeat the step 2 and 3 to create one more Lambda function with,

- Function Name : Enter **TestingLambda**
- Runtime : Select **Python 3.12**
- Execution Role : Click on **Use an existing role**

- Existing role : Select **Task156_Role**



- Once the Lambda Function is created successfully **copy the Function ARN.**

- Function code : Replace the existing code with the below and then click on **Deploy** button.

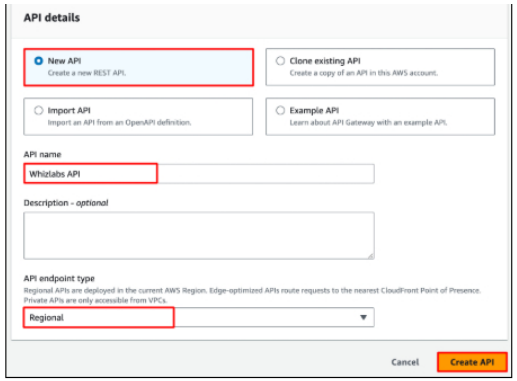
```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Message from testing Lambda')
    }
```

Task 4: Create an API

In this task, we are going to enable the user to create a new API in API Gateway called WhizlabAPI. The API will serve as the entry point for accessing the Lambda functions.

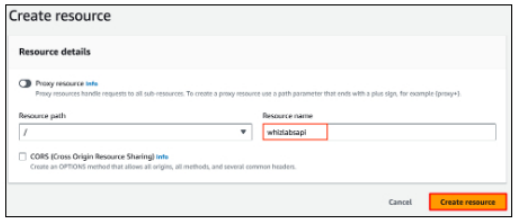
1. Navigate to the **Services** menu at the top, then click on **API Gateway** in the **Networking and Content Delivery** section.
2. Click on Build under **REST API**
3. Choose the protocol: Select **REST**
4. Choose create new API as **New API,** Under settings, choose the API name **Whizlabs API.** Leave other options as default and click on **Create API.**



Task 5: Creating a Resource

In this task, we are going to assist the user in creating a resource within the API. In this case, the resource will be used to define the endpoint for accessing the Lambda functions.

1. Once the API is created, select the API.
2. Select **Create Resource**.
    - Resource Name: Enter **whizlabsapi**
3. Enter the resource name and click on **Create Resource** button

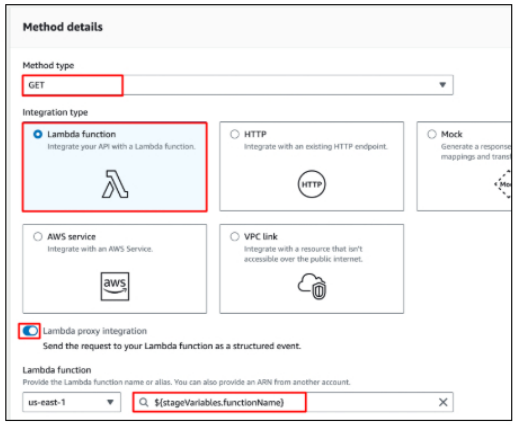


Task 6: Creating a Method

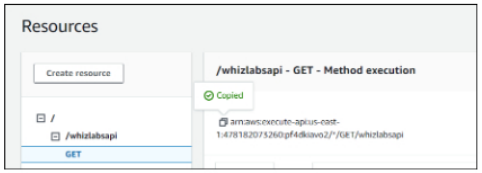
In this task, we are going to guide the user in creating a method for the resource. The method, specifically a GET method, will be integrated with the Lambda functions using the Lambda Proxy integration.

1. Once you create a resource **whizlabsapi**, click on **Create method**. Select **Get** in the drop-down list.
2. Select the Integration Type as **Lambda Function**
3. Use Lambda Proxy integration: Check the checkbox
4. Lambda Region: Select **us-east-1**
5. Lambda Function : Enter ***${stageVariables.functionName}***

Note: Don't select the lambda function name, only enter **${stageVariables.functionName}**



6. Click on the **Save** and Copy the **Source-ARN**



Task 7: Run the CLI command to give lambda permissions to the API using the EC2 instance

?In this task, we are going to run the CLI command to give lambda permissions to the API using the EC2 instance.

1. Navigate to the EC2 page and make sure you're in the **us-east-1 (N.Virginia)** region.

> **Note: Don't close the API Gateway page**

2. Click on **Instances** from the left side menu and select the EC2 instance which you created in the above step.

3. Please follow the steps in SSH into EC2 Instance.

4. Configure the **AWS CLI** on the EC2 instance by running the following command:

```
aws configure
```

AWS Access Key ID [None] : Enter the **access key** which is available below the lab credentials

AWS Secret Access Key [None] : Enter the **secret access key** which is available below the lab credentials

Default region name [None] : Enter **us-east-1**

Default output format [None] : Click the **Enter** button.

```
[ec2-user@ip-172-31-91-254 ~]$ aws configure
AWS Access Key ID [None]: AKIAVZ7S6AUGN4MCD3I7
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]:
[ec2-user@ip-172-31-91-254 ~]$
```

5. Copy the CLI commands for API gateway. To generate a UUID in Bash follow below command:

```
echo STATEMENT_ID=$(uuidgen)
```

6. To combine certain information to ensure uniqueness follow below command:

```
STATEMENT_ID="my_statement_id_$(date +%s)" # This appends current timestamp to the statement ID
```

7. Now, run the below command. Ensure you replace the **<productionlambda-function-arn>** with the function arn of **production lambda** and **<api-gateway-arn>** with the arn of the **api gateway** you have copied earlier.

```
aws lambda add-permission --function-name "<productionlambda-function-arn>" \
--source-arn "<api-gateway-arn>" \
--principal apigateway.amazonaws.com \
--statement-id "$STATEMENT_ID" \
--action lambda:InvokeFunction --region us-east-1
```

```
[ec2-user@ip-172-31-82-3 ~]$ aws lambda add-permission --function-name "arn:aws:lambda:us-east-1:374303187335:function:ProductionLambda" \
--source-arn "arn:aws:execute-api:us-east-1:374303187335:g8n8c901sh/*/GET/whizlabsapi" \
--principal apigateway.amazonaws.com \
--statement-id "$STATEMENT_ID" \
--action lambda:InvokeFunction --region us-east-1
```

8. Now, you will be able to see a JSON success output.

```
{
    "Statement": "{\"Sid\":\"my_statement_id_1748321592\",\"Effect\":\"Allow\",\"Principal\":{\"Service\":\"apigateway.amazonaws.com\"},\"Action\":\"lambda:InvokeFunction\",\"Resource\":\"arn:aws:lambda:us-east-1:374303187335:function:ProductionLambda\",\"Condition\":{\"ArnLike\":{\"AWS:SourceArn\":\"arn:aws:execute-api:us-east-1:374303187335:g8n8c901sh/*/GET/whizlabsapi\"}}}"
}
```

9. Similarly, run the same command but change the function name to **TestingLambda** (Second lambda function you created). Ensure you replace the **<testinglambda-function-arn>** with the function arn of **testing lambda** and **<api-gateway-arn>** with the arn of the **api gateway** you have copied earlier.

```
aws lambda add-permission --function-name "<testinglambda-function-arn>" \
--source-arn "<api-gateway-arn>" \
--principal apigateway.amazonaws.com \
--statement-id "$STATEMENT_ID" \
--action lambda:InvokeFunction --region us-east-1
```

```
[ec2-user@ip-172-31-82-3 ~]$ aws lambda add-permission --function-name "arn:aws:lambda:us-east-1:374303187335:function:TestingLambda" \
--source-arn "arn:aws:execute-api:us-east-1:374303187335:g8n8c901sh/*/GET/whizlabsapi" \
--principal apigateway.amazonaws.com \
--statement-id "$STATEMENT_ID" \
--action lambda:InvokeFunction --region us-east-1
```

10. Now, you will be able to see a JSON success output.


```
{
    "Statement": "{\"Sid\":\"my_statement_id_1748321592\",\"Effect\":\"Allow\",\"Principal\":{\"Service\":\"apigateway.amazonaws.com\"},\"Action\":\"lambda:InvokeFunction\",\"Resource\":\"arn:aws:lambda:us-east-1:374303187335:function:TestingLambda\",\"Condition\":{\"ArnLike\":{\"AWS:SourceArn\":\"arn:aws:execute-api:us-east-1:374303187335:g8n8c901sh/*/GET/whizlabsapi\"}}}"
}
```

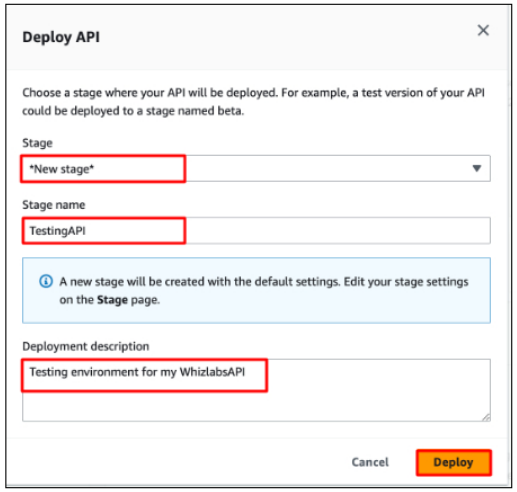

?

Task 8: Deploy API Gateway with two different stages

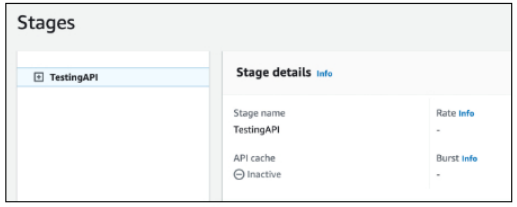
In this task, we are going to assist the user in deploying the API Gateway with two different stages: TestingAPI and ProductionAPI. These stages provide separate environments for testing and production.

Testing Lambda Stage

1. Once the resource and the method have been created successfully, you can deploy the API.
2. Click on **Deploy API.**
3. Select the Deployment Stage in the drop-down as **New Stage**.
4. Stage Name: Enter **TestingAPI** and Stage description as ***Testing environment for my WhizlabsAPI***.
5. Click on **Deploy**



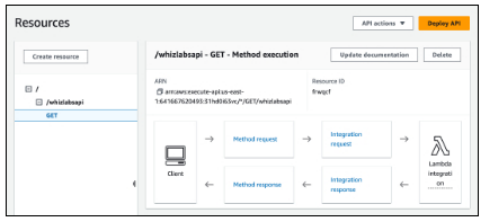
6. Once the API has been deployed, navigate to **Stages.**



Production Lambda stage

1. Click on the **Resources** from the left side menu.

2. Select the GET method which you have created.

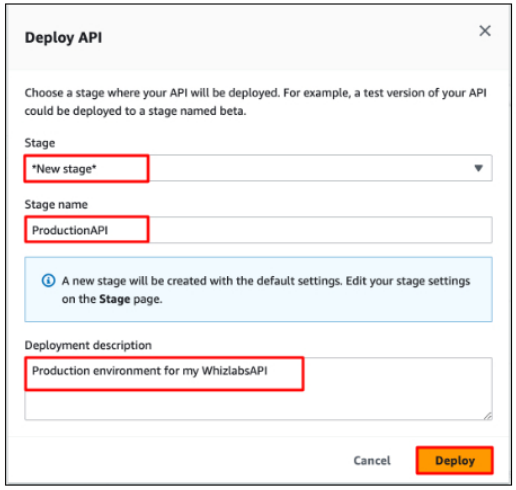


3. Select **Deploy API.**

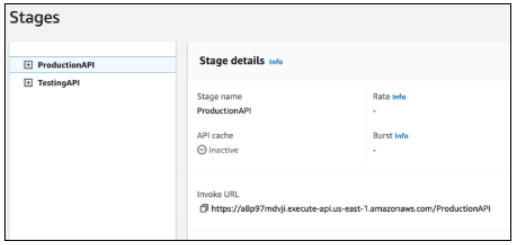
4. Select the Deployment Stage in the drop-down as **New Stage**.

5. Stage Name: Enter **ProductionAPI** and Stage description as ***Production environment for my WhizlabsAPI***.

6. Click on **Deploy**



7. Once the API has been deployed, navigate to **Stages.**

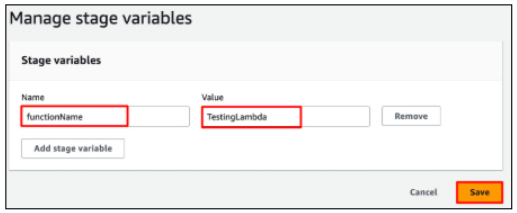


Task 9: Add stage variables to both stages

In this task, we are going to add stage variables to the deployed stages. The stage variables will be used to define the specific Lambda functions associated with each stage.

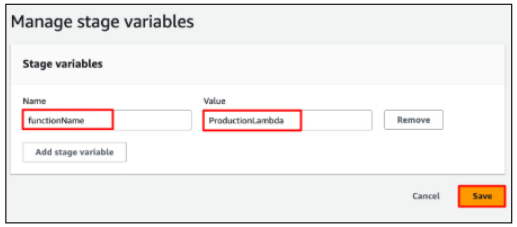
1. Now click on the **TestingAPI** stage in Stages.
2. Select the **Stage Variables** tab in the right side page and click on **Edit** and then **Add Stage Variable**

   - Name: Enter **functionName**
   - Value: Enter **TestingLambda**
   - Note: In the value field, enter the Testing lambda function name.
   - Now click on the **Save** button.



3. Now click on the **ProductionAPI** stage in Stages.
4. Select the **Stage Variables** tab in right side page and click on **Add Stage Variable**

   - Name: Enter **functionName**
   - Value : Enter **ProductionLambda**
   - Note: In the value field, enter the production lambda function name.



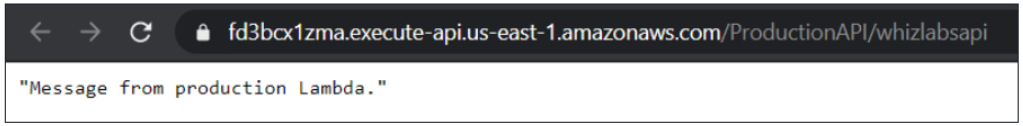
Task 10: Test the API Gateway

In this task, we are going to test the API Gateway by making GET requests to the endpoints associated with the deployed stages. This ensures that the API Gateway is properly integrated with the Lambda functions and functioning as expected.

1. Click on the **ProductionLambda** stage, open the stage
2. Now click on the GET method.

3. Copy and Paste the **Invoke URL** (followed by the resource name) in the new tab to make a GET request

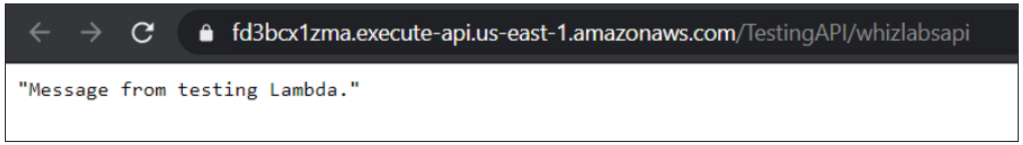
4. You will receive the GET request from the API. Here's an example:



> **Note**: If you make any mistake and after correcting it, still the API URL is showing the same error, please force refresh or clear cache.

5. Similarly, click on the **TestingLambda** stage, open the stage
6. Now click on the GET method.

7. Copy and Paste the **Invoke URL** (followed by the resource name) in the new tab to make a GET request.
8. You will receive the GET request from the API. Here's an example:

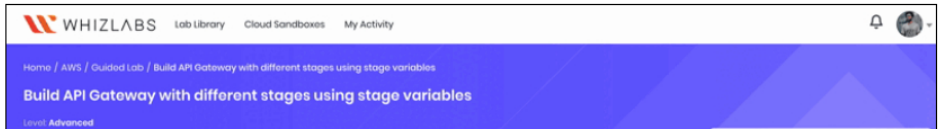


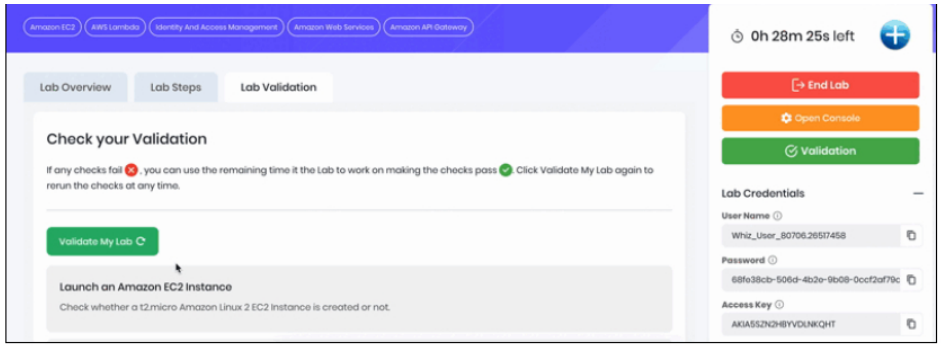
Do you Know?

API Gateway is designed to handle massive scale and can support up to hundreds of thousands of concurrent API requests. This means that even if your application experiences sudden spikes in traffic, API Gateway can handle the increased load seamlessly without any additional configuration or scaling effort on your part.

Task 11: Validation Test

1. Once the lab steps are completed, please click on the **Validation** button on the left side panel.
2. This will validate the resources in the AWS account and shows you whether you have completed this lab successfully or not.
3. Sample output :





Completion and Conclusion

1. You have successfully created an EC2 instance.
2. You have successfully created two Lambda functions.
3. You have successfully created the API.
4. You have successfully created the API Resource and Method.
5. You have successfully created two stages for API.
6. You have successfully added stage variables for both stages.
7. You have successfully tested the API

End Lab

1. Sign out of AWS Account.
2. You have successfully completed the lab.
3. Once you have completed the steps, click on **End Lab** from your whizlabs lab console and wait till the process gets completed.

DevOps

Cyber Security

Microsoft Power Platform

Microsoft 365 Certifications

Java Certifications

Certification

Google Cloud Certified Associate Cloud Engineer

Microsoft Power Platform Fundamentals (PL-900)

HashiCorp Certified Terraform Associate Certification

Snowflake SnowPro Advanced Architect Certification

Docker Certified Associate

Need help? You can or **+91 6364678444**