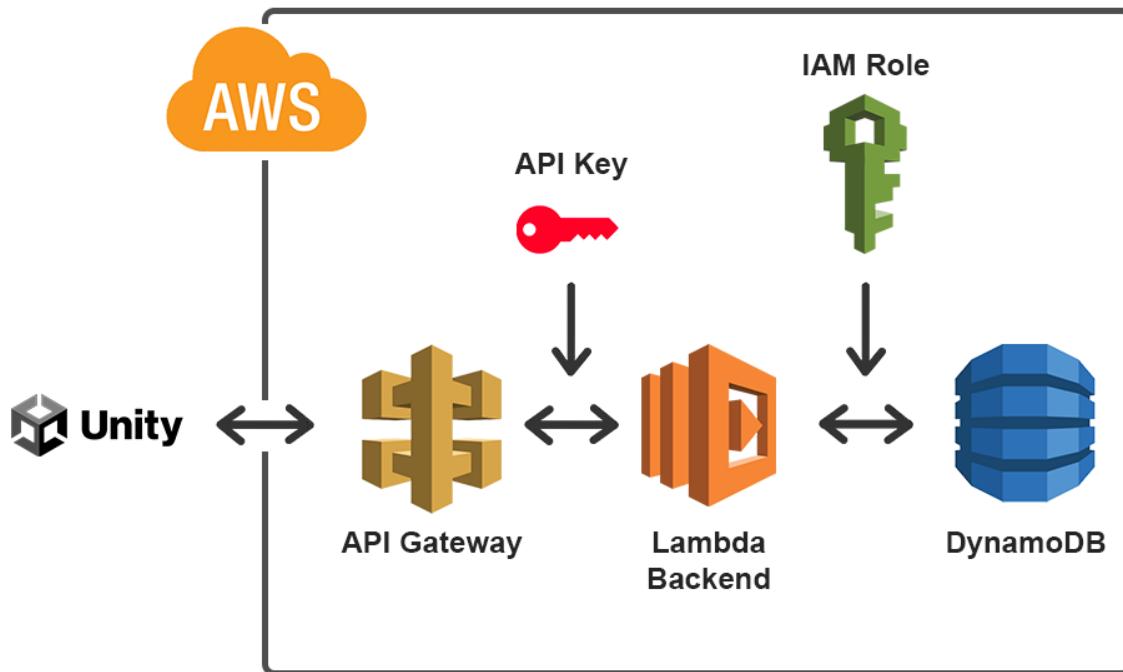


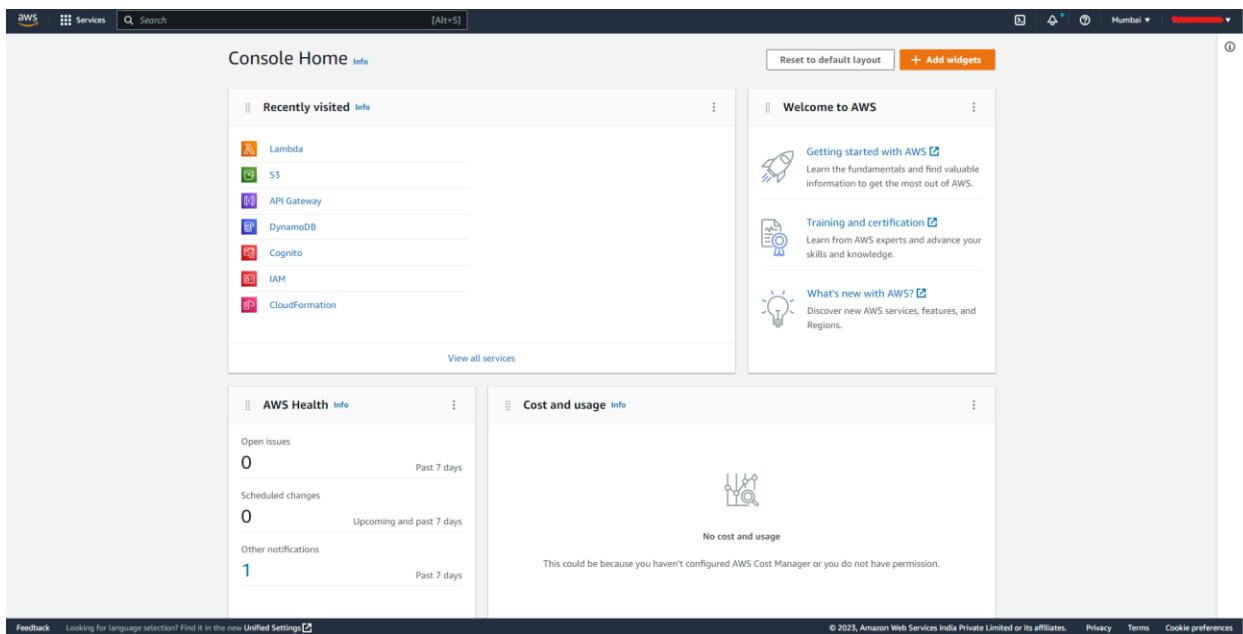
AWS DynamoDB and Server-less Backend Integration with Unity



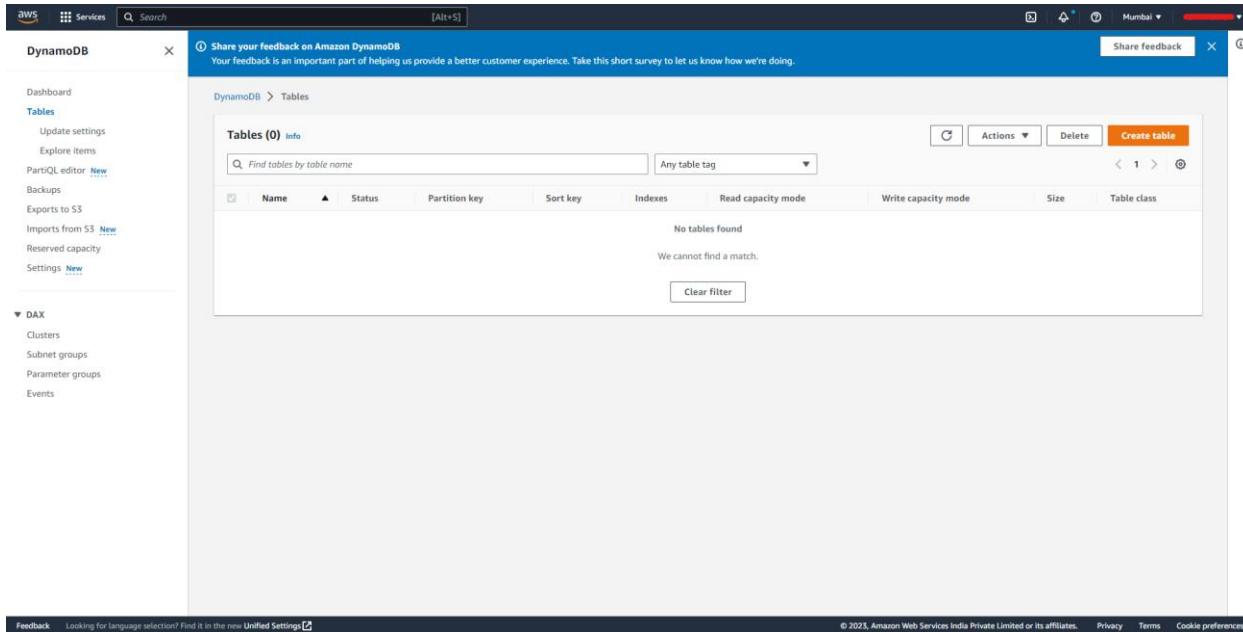
Process Blueprint

1. Unity sends web request to one of the endpoints in API Gateway
2. The web request must have the API Key as one of the headers
3. API Gateway then directs the request to Lambda Backend
4. Lambda Backend needs a Default Execution Role to function
5. Lambda then does the necessary operation (register, login, verify etc.) by accessing the DynamoDB
6. The data is retrieved / updated and the status is returned as JSON object
7. JSON object is parsed in Unity

Login into AWS Console



Creating DynamoDB Table



Go to *DynamoDB service > Tables and then Create Table*

Share your feedback on Amazon DynamoDB
Your feedback is an important part of helping us provide a better customer experience. Take this short survey to let us know how we're doing.

DynamoDB > Tables > Create table

Create table

Table details [Info](#)
DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
This will be used to identify your table.
 User_Data
Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.)

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

<input type="text" value="username"/> username	String
--	--------

1 to 255 characters and case sensitive.

Sort key - optional
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

<input type="text" value="Enter the sort key name"/> Enter the sort key name	String
--	--------

1 to 255 characters and case sensitive.

Table settings

Default settings
The fastest way to create your table. You can modify these settings now or after your table has been created.

Customize settings
Use these advanced features to make DynamoDB work better for your needs.

Feedback Looking for language selection? Find it in the new [Unified Settings](#). © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

Creating IAM Role

New! Securely access AWS services from your data center with [IAM Roles Anywhere](#) [Learn more](#)

Identity and Access Management (IAM)

Access management

- User groups
- Users
- Roles**
- Policies
- Identity providers
- Account settings

Access reports

- Access analyzer
- Archive rules
- Analyzers
- Settings

Credential report

Organization activity

Service control policies (SCPs)

Related consoles

IAM Identity Center [New](#)

Roles (7) Info
An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

<input type="checkbox"/> Role name	Trusted entities	Last activity
<input type="checkbox"/> AWSServiceRoleForAPIGateway	AWS Service: ops.apigateway (Service-Linked Role)	-
<input type="checkbox"/> AWSServiceRoleForApplicationAutoScaling_DynamoDBTable	AWS Service: dynamodb.application-autoscaling (Service-Linked Role)	36 minutes ago
<input type="checkbox"/> AWSServiceRoleForSupport	AWS Service: support (Service-Linked Role)	-
<input type="checkbox"/> AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linked Role)	-
<input type="checkbox"/> Cognito_SafetyVRDemoAuth_Role	Identity Provider: cognito-identity.amazonaws.com	-
<input type="checkbox"/> Cognito_SafetyVRDemoUnauth_Role	Identity Provider: cognito-identity.amazonaws.com	-
<input type="checkbox"/> lambda-11-16-22-role-5jn204su	AWS Service: lambda	-

Roles Anywhere [Info](#)
Authenticate your non AWS workloads and securely provide access to AWS services.

Access AWS from your non AWS workloads
Operate your non AWS workloads using the same authentication and authorization strategy that you use within AWS.

X.509 Standard
Use your own existing PKI infrastructure or use [AWS Certificate Manager](#) [Private Certificate Authority](#) to authenticate identities.

Temporary credentials
Use temporary credentials with ease and benefit from the enhanced security they provide.

Feedback Looking for language selection? Find it in the new [Unified Settings](#). © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

Introducing the new IAM roles experience
We've redesigned the IAM roles experience to make it easier to use. [Let us know what you think.](#)

IAM > Roles > Create role

Step 1 Select trusted entity

Step 2 Add permissions

Step 3 Name, review, and create

Select trusted entity Info

Trusted entity type

- AWS service Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation Allows users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy Create a custom trust policy to enable others to perform actions in this account.

Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Common use cases

- EC2 Allows EC2 instances to call AWS services on your behalf.
- Lambda Allows Lambda functions to call AWS services on your behalf.

Use cases for other AWS services:

Choose a service to view use case

[Cancel](#) [Next](#)

[Feedback](#) Looking for language selection? Find it in the new [Unified Settings](#).

© 2023, Amazon Web Services India Private Limited or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Introducing the new IAM roles experience
We've redesigned the IAM roles experience to make it easier to use. [Let us know what you think.](#)

IAM > Roles > Create role

Step 1 Select trusted entity

Step 2 Add permissions

Step 3 Name, review, and create

Add permissions Info

Permissions policies (806) Info
Choose one or more policies to attach to your new role.

Filter policies by property or policy name and press enter:

<input type="checkbox"/>	Policy name <small>Info</small>	Type	Description
<input type="checkbox"/>	AWSDirectConnectReadOnlyAccess	AWS m...	Provides read only access to AWS Direct Connect via the AWS Management Console.
<input type="checkbox"/>	AmazonGlacierReadOnlyAccess	AWS m...	Provides read only access to Amazon Glacier via the AWS Management Console.
<input type="checkbox"/>	AWSMarketplaceFullAccess	AWS m...	Provides the ability to subscribe and unsubscribe to AWS Marketplace software, allows users to manage Marketplace software instances from...
<input type="checkbox"/>	AWSSSOIdentityAdministrator	AWS m...	Administrator access for SSO Directory
<input type="checkbox"/>	AWSIoT1ClickReadOnlyAccess	AWS m...	Provides read only access to AWS IoT 1-Click.
<input type="checkbox"/>	AutoScalingConsoleReadOnlyAccess	AWS m...	Provides read-only access to Auto Scaling via the AWS Management Console.
<input type="checkbox"/>	AmazonDMSRedshiftS3Role	AWS m...	Provides access to manage S3 settings for Redshift endpoints for DMS.
<input type="checkbox"/>	AWSQuickSightListIAM	AWS m...	Allow QuickSight to list IAM entities
<input type="checkbox"/>	AWSHealthFullAccess	AWS m...	Allows full access to the AWS Health APIs and Notifications and the Personal Health Dashboard
<input type="checkbox"/>	AlexaForBusinessGatewayExecution	AWS m...	Provide gateway execution access to AlexaForBusiness services
<input type="checkbox"/>	AmazonElasticTranscoder_ReadOnlyAcc...	AWS m...	Grants users read-only access to Elastic Transcoder and list access to related services.
<input type="checkbox"/>	AmazonRDSFullAccess	AWS m...	Provides full access to Amazon RDS via the AWS Management Console.
<input type="checkbox"/>	SupportUser	AWS m...	This policy grants permissions to troubleshoot and resolve issues in an AWS account. This policy also enables the user to contact AWS suppo...

[Feedback](#) Looking for language selection? Find it in the new [Unified Settings](#).

© 2023, Amazon Web Services India Private Limited or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Add Policies

1. *CloudWatchLogsFullAccess*
2. *AmazonDynamoDBFullAccess*

The screenshot shows the AWS IAM Roles creation interface. At the top, a blue banner says "Introducing the new IAM roles experience. We've redesigned the IAM roles experience to make it easier to use. Let us know what you think." Below this, the navigation path is IAM > Roles > Create role. The current step is Step 3: Name, review, and create. The sub-step is Select trusted entity. The main area is titled "Name, review, and create". Under "Role details", the "Role name" is set to "login-backend-role". The "Description" field contains "Allows Lambda functions to call AWS services on your behalf.". In the "Step 1: Select trusted entities" section, there is a code editor containing a JSON-based policy document:

```
1: [ { 2: "Version": "2012-10-17", 3: "Statement": [ 4: { 5: "Effect": "Allow", 6: "Action": [ 7: "sts:AssumeRole" 8: ], 9: "Principal": { 10: "Service": [ 11: "lambda.amazonaws.com" 12: ] 13: } 14: } 15: ] 16: }]
```

At the bottom of the page, there are links for Feedback, Unified Settings, and a footer with copyright information.

Add Role Name and then create role

Creating Lambda function

The screenshot shows the AWS Lambda Functions management interface. The left sidebar has sections for Dashboard, Applications, Functions, Additional resources (Code signing configurations, Layers, Replicas), and Related AWS resources (Step Functions state machines). The main area is titled "Lambda > Functions" and shows a table for managing functions. The table header includes columns for Function name, Description, Package type, Runtime, Last modified, and Actions. A message at the bottom of the table says "There is no data to display." There is a prominent orange "Create function" button at the top right of the table area. The top bar includes the AWS logo, Services, a search bar, and a "Mumbai" region selector. The footer contains links for Feedback, Unified Settings, and a footer with copyright information.

aws Services Search [Alt+S] Mumbai ⓘ

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.

Basic information

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

Routine Info
Choose the language to use when writing your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Node.js 18.x

Architecture Info
Choose the instruction set architecture you want for your function code.
x86_64 **arm64**

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

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console [IAM console](#).
 Create a new role with basic Lambda permissions
 Use an existing role
 Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
login-backend-role [View the login-backend-role role](#) [on the IAM console](#).

Feedback Looking for language selection? Find it in the new [Unified Settings](#).

© 2023, Amazon Web Services India Private Limited or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

aws Services Search [Alt+S] Mumbai ⓘ

Lambda > Functions > login-backend

login-backend

Function overview [Info](#)

Throttle	Copy ARN	Actions ▾
 login-backend	Description -	
 Layers (0)	Last modified 15 seconds ago	
+ Add trigger	Function ARN arn:aws:lambda:ap-south-1:583891986352:function:login-backend	
	Function URL Info	

Code [Test](#) [Monitor](#) [Configuration](#) [Aliases](#) [Versions](#)

Code source [Info](#)

[Upload from](#)

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

```
index.js
1 export const handler = async(event) => {
2   // TODO: Implement
3   const response = {
4     statusCode: 200,
5     body: JSON.stringify('Hello from Lambda!'),
6   };
7   return response;
8 }
9
```

Feedback Looking for language selection? Find it in the new [Unified Settings](#).

© 2023, Amazon Web Services India Private Limited or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

The screenshot shows the AWS Lambda function configuration page for a function named 'login-backend'. The 'Configuration' tab is selected. On the left, a sidebar lists various configuration options: Triggers, Permissions, Destinations, Function URL, Environment variables, Tags, VPC, Monitoring and operations tools, Concurrency, Asynchronous invocation, Code signing, Database proxies, File systems, and State machines. The main panel displays the 'General configuration' section with the following details:

Description	Memory	Ephemeral storage
-	128 MB	512 MB

Below this, there is a 'Timeout' setting of '0 min 3 sec'. An 'Edit' button is located in the top right corner of the configuration panel.

Edit General Configuration (Optional)

500 MB Memory

5 Sec Timeout

The screenshot shows the 'Edit basic settings' dialog box for the 'login-backend' function. The 'Basic settings' tab is selected. The configuration fields are as follows:

- Description**: An optional text input field containing the placeholder 'Description - optional'.
- Memory**: A text input field set to '500 MB'. A tooltip indicates that memory is allocated CPU proportional to the memory configured.
- Ephemeral storage**: A text input field set to '512 MB'. A tooltip indicates that you can configure up to 10 GB of ephemeral storage (/tmp) for your function.
- Timeout**: A numeric input field set to '0 min 3 sec'.
- Execution role**: A radio button group where 'Use an existing role' is selected, and 'Create a new role from AWS policy templates' is unselected.
- Existing role**: A dropdown menu set to 'login-backend-role'. A tooltip indicates that the role must have permission to upload logs to Amazon CloudWatch Logs.

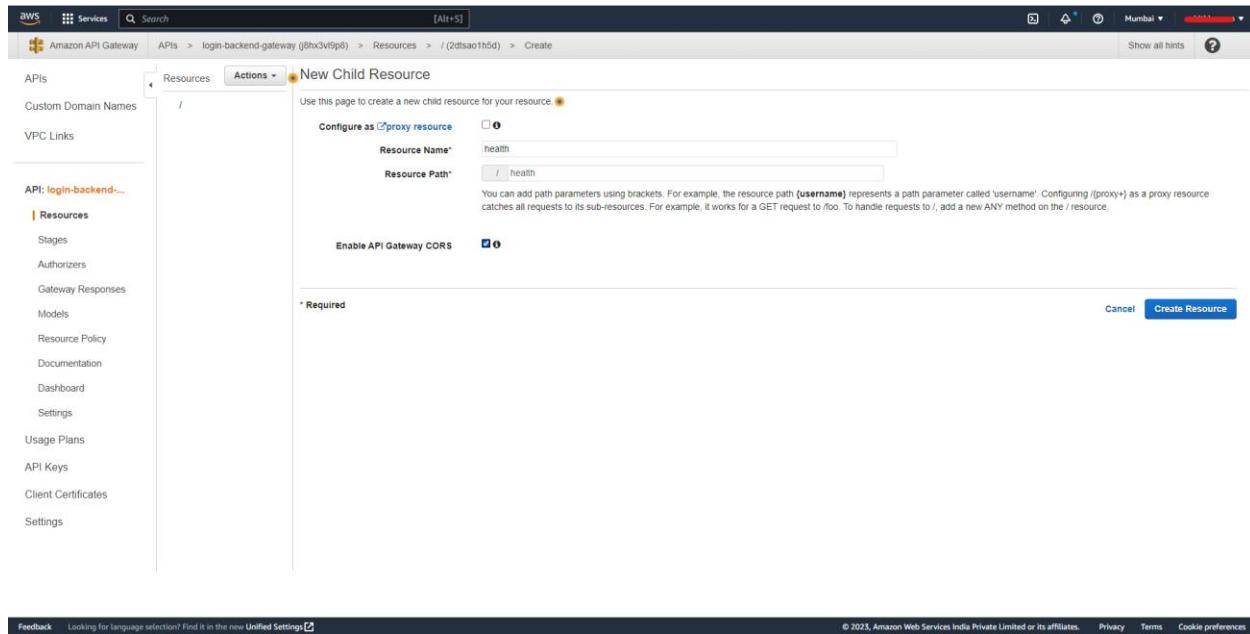
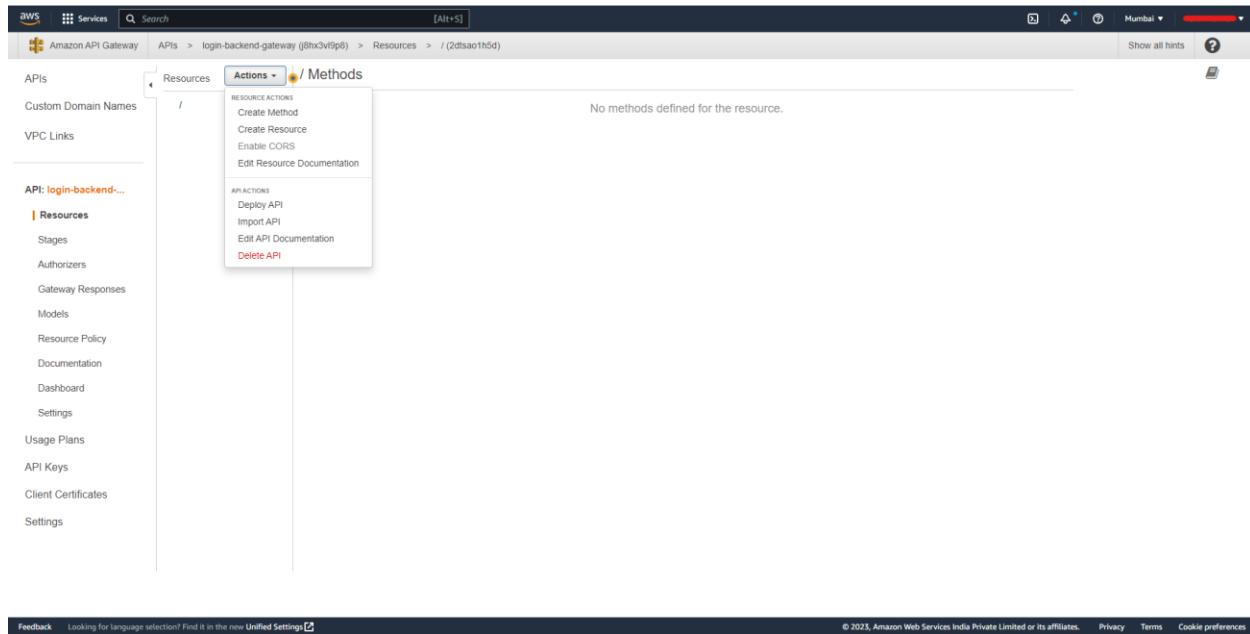
At the bottom of the dialog are 'Cancel' and 'Save' buttons. The footer includes standard AWS navigation links: Feedback, Unified Settings, Privacy, Terms, and Cookie preferences.

Creating API Gateway

The screenshot shows the AWS API Gateway landing page. It features a sidebar with 'APIs', 'Custom domain names', and 'VPC links'. Four main options are listed: 'API Gateway', 'WebSocket API', 'REST API', and 'REST API Private'. Each option has a brief description and an 'Import' and 'Build' button. A red arrow points to the 'Build' button for the 'REST API' entry.

The screenshot shows the 'Create new API' wizard. Step 1: Choose the protocol. It asks if you want to create a REST API or a WebSocket API. 'REST' is selected. Below it, it says 'In Amazon API Gateway, a REST API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.' There are three options: 'New API' (selected), 'Import from Swagger or Open API 3', and 'Example API'. A red arrow points to the 'Create API' button at the bottom right.

The screenshot shows the 'Create new API' wizard. Step 2: Settings. It asks for a friendly name and description. The 'API name*' field contains 'login-backend-gateway'. Other fields include 'Description' (empty) and 'Endpoint Type' (set to 'Regional'). A red arrow points to the 'Create API' button at the bottom right.



Create New Resource “health”

Enable API Gateway CORS

The screenshot shows the AWS API Gateway console. On the left, the navigation sidebar is visible with options like APIs, Custom Domain Names, VPC Links, and API: login-backend-... (selected). Under Resources, there are links for Stages, Authorizers, Gateway Responses, Models, Resource Policy, Documentation, Dashboard, Settings, Usage Plans, API Keys, Client Certificates, and Settings. The main content area shows the path: APIs > login-backend-gateway (jBhx3vI9p0) > Resources > /health (9y4t5). A modal window titled 'OPTIONS' is open for the /health resource, showing method details: GET, OPTIONS, Mock Endpoint, Authorization: None, and API Key: Not required.

Feedback Looking for language selection? Find it in the new Unified Settings [\[?\]](#)

© 2023, Amazon Web Services India Private Limited or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Select the “health” resource

Actions > Create Method

Set Method to “GET”

The screenshot shows the AWS API Gateway console. The left sidebar is identical to the previous screenshot. The main content area shows the path: APIs > login-backend-gateway (jBhx3vI9p0) > Resources > /health (9y4t5) > GET. A modal window titled '/health - GET - Setup' is open, prompting the user to choose an integration point. The 'Integration type' section has 'Lambda Function' selected. Below it, 'Use Lambda Proxy integration' is checked. The 'Lambda Region' dropdown is set to 'ap-south-1' and the 'Lambda Function' input field contains 'login-backend'. A 'Save' button is at the bottom right of the modal.

Feedback Looking for language selection? Find it in the new Unified Settings [\[?\]](#)

© 2023, Amazon Web Services India Private Limited or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Now, repeat the above steps for creating Resources “register”, “login”, “verify”

With method as “POST”

Select / (root) and then Actions > Create Resource

The screenshot shows the AWS API Gateway console. The left sidebar is open with the 'APIs' tab selected. Under 'APIs', the 'Resources' section is highlighted. In the main content area, a tree view shows a root node '/'. Below it is a node '/health' with 'GET' and 'OPTIONS' methods listed. A new node '/register' has been created and is currently selected. The right side of the screen displays a message: 'No methods defined for the resource.' At the bottom, there is a feedback bar and a footer with copyright information.

The screenshot shows the AWS API Gateway console with the '/register' resource selected. The right panel displays the configuration for the 'OPTIONS' method. It includes a 'Mock Endpoint' section and fields for 'Authorization' (set to 'None') and 'API Key' (set to 'Not required'). The left sidebar remains the same as the previous screenshot, showing the 'APIs' tab and the 'Resources' section.

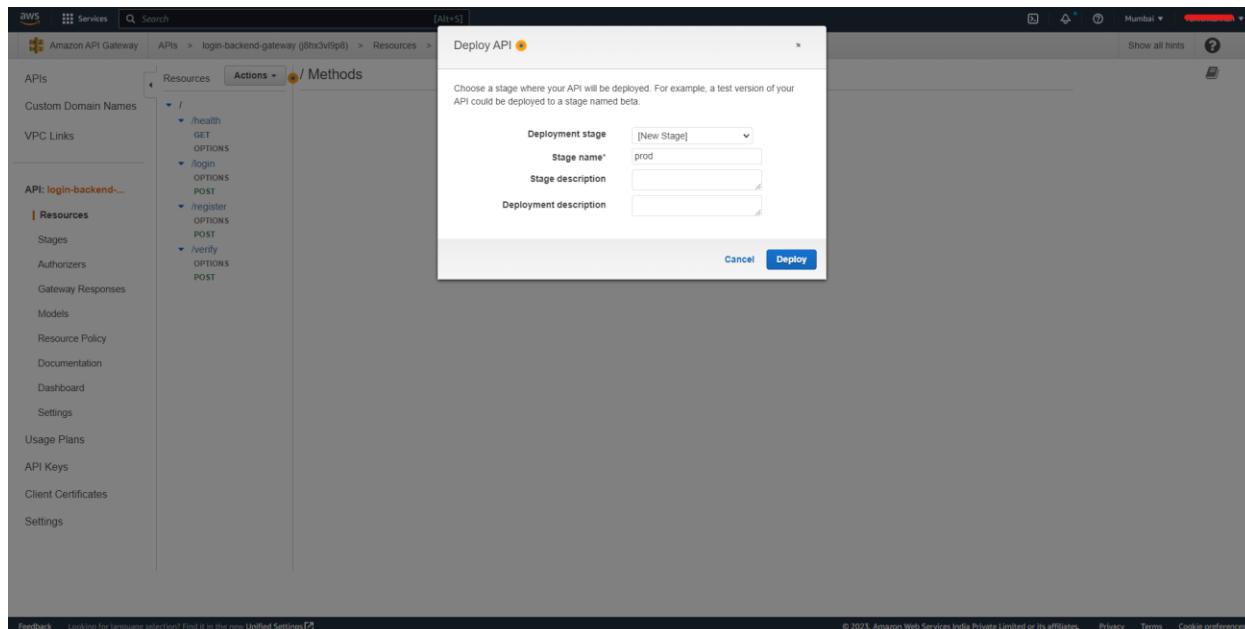
Select **/register** and then Actions > Create Resource

The screenshot shows the AWS API Gateway console. The left sidebar shows the navigation menu with 'APIs' selected. Under 'APIs', the 'login-backend-' API is selected. In the main content area, the path 'APIs > login-backend-gateway (j8hx3vtp0) > Resources > /register (ae5buc) > POST' is shown. A modal window titled '/register - POST - Setup' is open, prompting the user to choose an integration point. The 'Integration type' dropdown is set to 'Lambda Function'. The 'Lambda Region' is set to 'ap-south-1' and the 'Lambda Function' is set to 'login-backend'. A 'Save' button is visible at the bottom right of the modal.

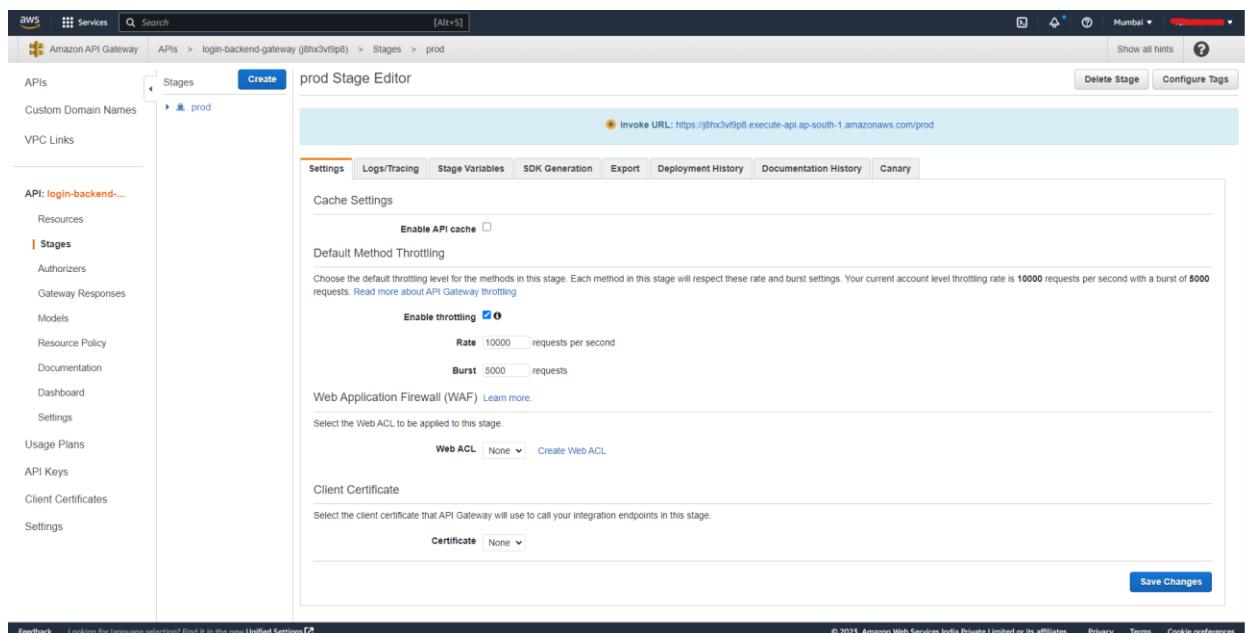
The screenshot shows the AWS API Gateway console. The left sidebar shows the navigation menu with 'APIs' selected. Under 'APIs', the 'login-backend-' API is selected. In the main content area, the path 'APIs > login-backend-gateway (j8hx3vtp0) > Resources > /verify (xfbmqu)' is shown. A modal window titled '/verify Methods' is open, showing two tabs: 'OPTIONS' and 'POST'. The 'OPTIONS' tab shows 'Mock Endpoint' settings with 'Authorization: None' and 'API Key: Not required'. The 'POST' tab shows similar settings. A 'Save' button is visible at the bottom right of the modal.

All four resources with methods have been created

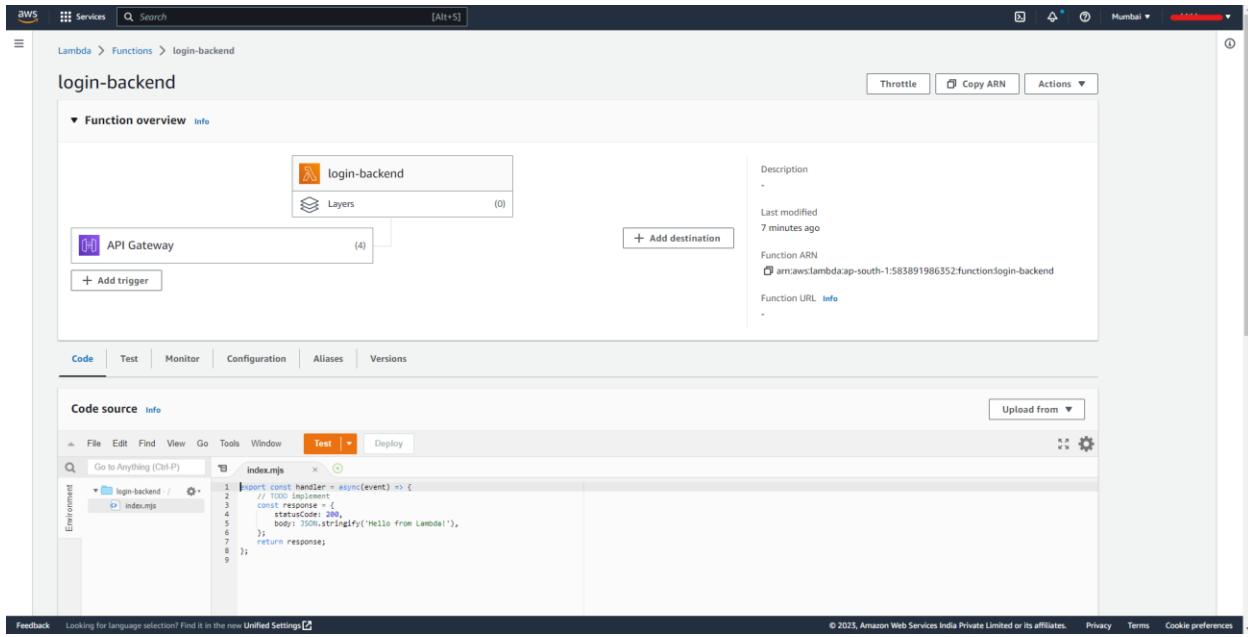
Select “ / ” (root) and then Actions > Deploy API



API Deployed



Testing the API



Modify index.js with the below program

```
const healthPath = '/health';
const registerPath = '/register';
const loginPath = '/login';
const verifyPath = '/verify';

export const handler = async(event) => {
    console.log('Request Event: ', event);
    let response;
    switch(true){
        case event.httpMethod === 'GET' && event.path === healthPath:
            response = buildResponse(200);
            break;
        case event.httpMethod === 'POST' && event.path === registerPath:
            response = buildResponse(200);
            break;
        case event.httpMethod === 'POST' && event.path === loginPath:
            response = buildResponse(200);
            break;
        case event.httpMethod === 'POST' && event.path === verifyPath:
            response = buildResponse(200);
            break;
        default:
            response = buildResponse(404, '404 Not Found');
    }
    return response;
};

function buildResponse(statusCode, body) {
    return {
        statusCode : statusCode,
        headers : {
            'Access-Control-Allow-Origin' : '*',
            'Content-Type' : 'application/json',
        },
        body : JSON.stringify(body)
    }
}
```

Deploy changes

Successfully updated the function **login-backend**.

login-backend

Description

Last modified 1 second ago

Function ARN arn:aws:lambda:ap-south-1:583891986352:function:login-backend

Function URL [Info](#)

Code Test Monitor Configuration Aliases Versions

Code source [Info](#)

File Edit Find View Go Tools Window Test Deploy

index.js

```
const healthPath = '/health';
const registerPath = '/register';
const loginPath = '/login';
const verifyPath = '/verify';

export const handler = async(event) => {
  if (event.requestType === 'POST') {
    let response;
    if (event.path === healthPath) {
      response = buildResponse(200);
    } else if (event.path === registerPath) {
      response = buildResponse(201);
    } else if (event.path === loginPath) {
      response = buildResponse(200);
    } else if (event.path === verifyPath) {
      response = buildResponse(200);
    } else {
      response = buildResponse(404);
    }
    return response;
  }
};
```

Feedback Looking for language selection? Find it in the new [Unified Settings](#)

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

Go back to API Gateway

Copy the API invoke URL

Successfully updated the stage **prod**.

prod Stage Editor

Invoke URL: <https://j0nx3vlp8.execute-api.ap-south-1.amazonaws.com/prod>

Settings Logs/Tracing Stage Variables SDK Generation Export Deployment History Documentation History Canary

Cache Settings

Enable API cache

Default Method Throttling

Choose the default throttling level for the methods in this stage. Each method in this stage will respect these rate and burst settings. Your current account level throttling rate is **10000** requests per second with a burst of **5000** requests. [Read more about API Gateway throttling](#)

Enable throttling Rate 10000 requests per second Burst 5000 requests

Web Application Firewall (WAF) [Learn more](#)

Select the Web ACL to be applied to this stage.

Web ACL None [Create Web ACL](#)

Client Certificate

Select the client certificate that API Gateway will use to call your integration endpoints in this stage.

Certificate None

Save Changes

Feedback Looking for language selection? Find it in the new [Unified Settings](#)

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

Open Postman and click on Send a request

The screenshot shows the Postman homepage. At the top, there's a search bar and a navigation bar with links for Home, Workspaces, API Network, and Explore. A 'Search Postman' input field is also present.

Good afternoon, [REDACTED]!

Use Launchpad to start something new, pick up where you left off, or explore some resources to help you master Postman.

Product Updates

- Postman Academy is live!** New developer and administrator learning paths are now available! [Start Now](#)
- Intergalactic: Collaboration and API Governance for Teams** Ensure the security, scalability and documentation of your APIs in this January 25th Postman learning session. Learn about Git integration, collaboration and more. [Register Now](#)

Activity Feed

Your team's activity will show up here. Get started by inviting people to your team. [Create Team](#)

Get started with Postman

- Start with something new**: Create a new request, collection, or API in a workspace. [Create New →](#)
- Import an existing file**: Import any API schema file from your local drive or GitHub. [Import file →](#)

Explore popular APIs [Explore all →](#)

Paste the invoke url

The screenshot shows the Postman workspace interface. On the left, there's a sidebar with categories like Collections, Environments, Mock Servers, Monitors, Flows, and History. The History section is currently selected, showing a list of recent requests.

In the main area, there's a collection named "Yesterday" with several requests listed under it. One specific request is highlighted:

GET https://b5n2d4ypa9.execute-api.ap-south-1.amazonaws.com/prod

The request details are shown in the editor:

- Method**: GET
- URL**: https://b5n2d4ypa9.execute-api.ap-south-1.amazonaws.com/prod
- Params**: Authorization, Headers (5), Body, Pre-request Script, Tests, Settings
- Query Params**: KEY, VALUE, DESCRIPTION, Bulk Edit

The response pane below shows a small illustration of a character pointing at a screen with the text "Click Send to get a response".

Start working with APIs [67%](#)
Next: Save requests. [Show me](#)

The screenshot shows the Postman application interface. On the left, there's a sidebar with sections like 'My Workspace', 'Collections', 'APIs', 'Environments', 'Mock Servers', 'Monitors', 'Flows', and 'History'. The 'History' section is currently selected and highlighted in orange. In the main workspace, a request is being made to the URL `https://j8hx3v9pb.execute-api.ap-south-1.amazonaws.com/prod/health`. The method is set to 'GET'. Below the URL, there are tabs for 'Params', 'Authorization', 'Headers (5)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. Under 'Headers (5)', there is a single entry: 'Key' (with value 'Key') and 'Description' (with value 'Description'). The 'Body' tab is selected, showing a simple JSON response: `1`. At the bottom right of the response area, there's a red arrow pointing to the status bar which displays 'Status: 200 OK'.

Add “**/health**” to the end of the url and click Send

Response received with Status 200 Ok

Try the same for all endpoints “**/register**” “**/login**” “**/verify**”

with respective methods “**POST**”

This API gateway allows requests by anyone with the url

Create an API Key to send authorized requests

We'll send the API Key as one of the headers “**x-api-key**”

The screenshot shows the AWS Amazon API Gateway interface. On the left, there's a navigation sidebar for the 'login-backend' API, which includes sections for Resources, Stages, Authorizers, Gateway Responses, Models, Resource Policy, Documentation, Dashboard, Settings, Usage Plans, and API Keys. The 'API Keys' section is currently selected. The main content area displays a list of API keys under the heading 'Select an API key'. At the top of this list, there's a 'Actions' dropdown with options 'Create API key' and 'Import API keys'. A search bar labeled 'Search...' is also visible.

This screenshot shows the details of a specific API key named 'unity-client'. The key has the following attributes: ID: jrb2v4d58l, Name: unity-client, API key (redacted), Description: No description, and Enabled status. In the 'Associated Usage Plans' section, there is a button to 'Add to Usage Plan' and a table showing 'No associated Usage Plans'. There are also 'Delete API Key' and 'Configure Tags' buttons at the top right of the key's detail view.

Feedback Looking for language selection? Find it in the new Unified Settings [\[Feedback\]](#) © 2023, Amazon Web Services India Private Limited or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Enter Customer Name and Create API with Auto-Generate on

Click on show next to API key to view the key

Go back to API Gateway > Usage Plans

The screenshot shows the 'Create Usage Plan' interface. On the left, a sidebar lists various API management options like APIs, Custom Domain Names, VPC Links, and the selected 'API: login-backend-...'. Under 'Usage Plans', 'Create' is highlighted. The main form has fields for 'Name' (test-plan) and 'Description'. The 'Throttling' section is expanded, showing 'Enable throttling' checked, 'Rate' set to 1000 requests per second, and 'Burst' set to 500 requests. The 'Quota' section is also expanded, showing 'Enable quota' checked, with a dropdown set to '1000000 requests per Month'. A note at the bottom says '* Required'. At the bottom right are 'Next' and 'Back' buttons.

Add API Stage

The screenshot shows the 'Associated API Stages' interface. The left sidebar is identical to the previous screenshot. The main area displays a table titled 'Add API Stage' with one row. The row contains 'login-backend-gateway' under 'API', 'prod' under 'Stage', and 'No Methods Configured' under 'Method Throttling'. A 'Configure Method Throttling' button is next to it. At the bottom are 'Back' and 'Next' buttons.

Add API Key to Usage Plan

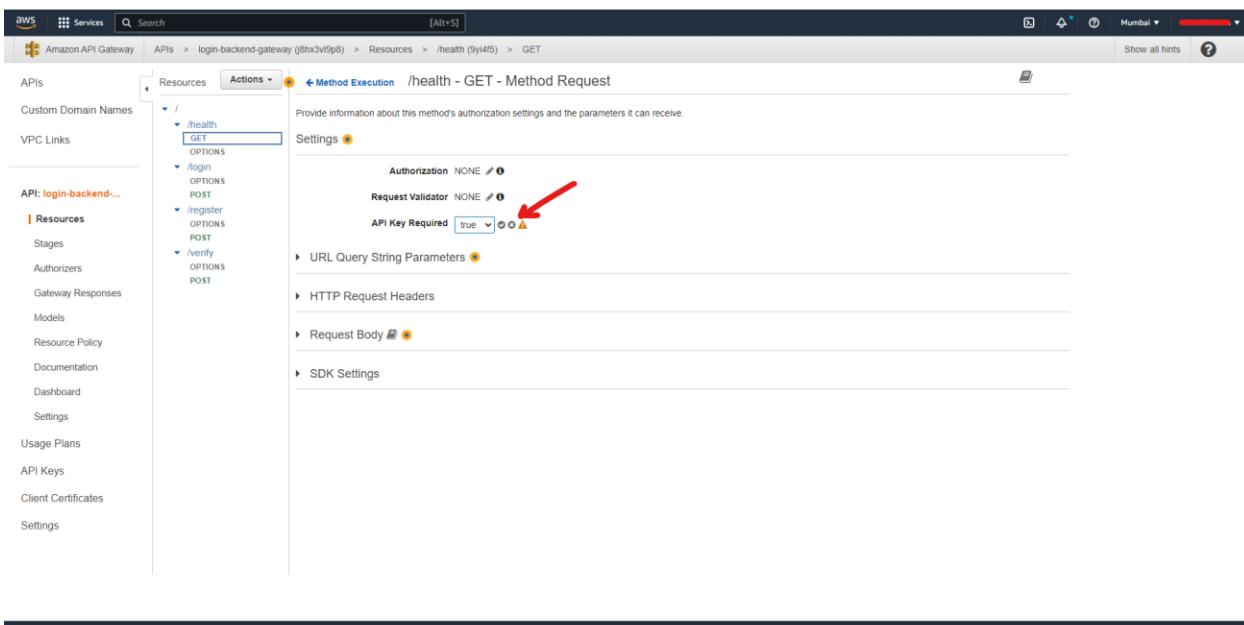
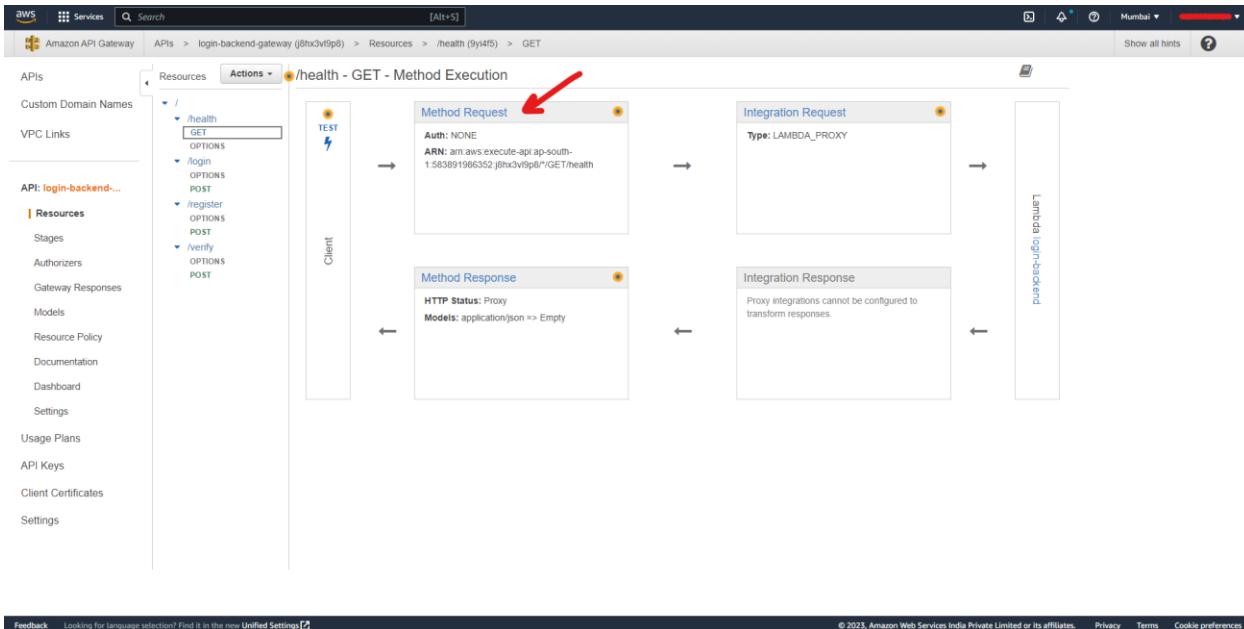
The screenshot shows the AWS Amazon API Gateway Usage Plans API Keys page. On the left, a sidebar for the 'login-backend' API lists various options like APIs, Stages, and Settings. The main area is titled 'Usage Plan API Keys' and contains a search bar and two buttons: 'Add API Key to Usage Plan' and 'Create API Key and add to Usage Plan'. A table displays a single row with the name 'unity-client'. At the bottom right are 'Back' and 'Done' buttons.

Usage Plan created

The screenshot shows the AWS Amazon API Gateway Usage Plans Details page for the 'test-plan' usage plan. The sidebar on the left shows the 'test-plan' usage plan selected. The main area displays the usage plan details: ID (yv0q5r), Name (test-plan), and Description (No description). It also shows rate limits (Rate: 1,000 requests per second, Burst: 500 requests) and a quota of 1,000,000 requests per month starting on the 1st day. Below this, the 'Associated API Stages' section shows a table with one entry: 'login-backend-gateway' in the API column, 'prod' in the Stage column, and 'No Methods Configured' in the Method Throttling column. A 'Configure Method Throttling' button is also present. At the bottom right are 'Edit' and 'Actions' buttons.

[Go back to API Gateway](#)

And for **GET** method, set **API Key required** to **True**

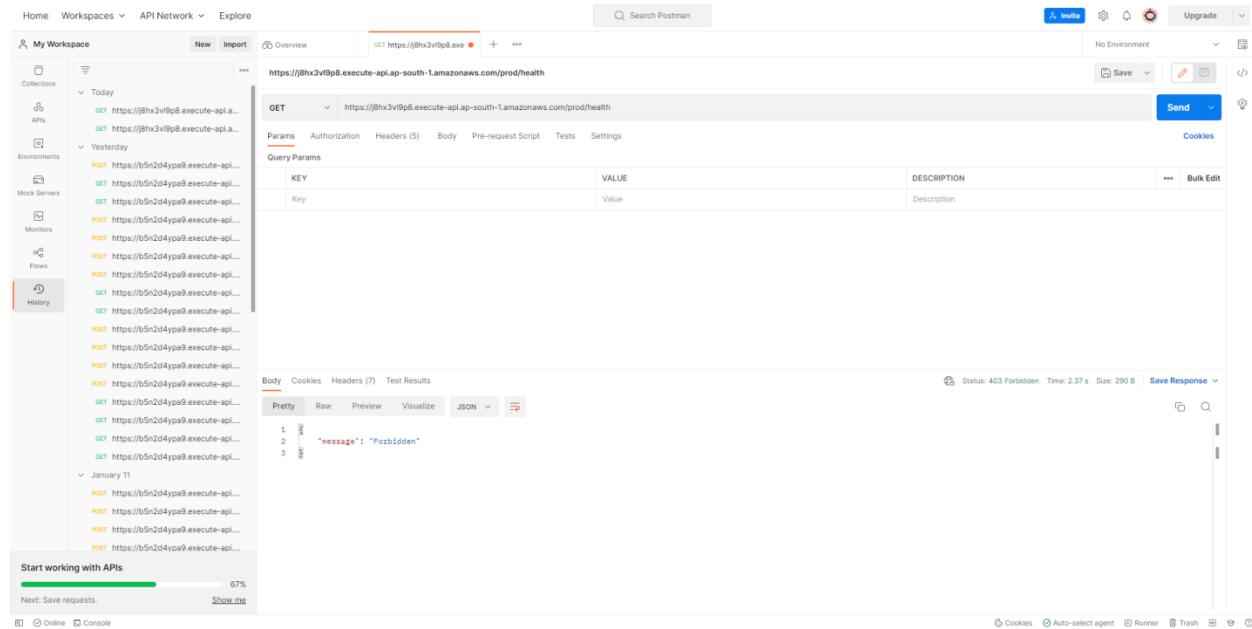


Repeat this for each all the **POST** methods

Select “/” (root) and then Actions > Deploy API

Go back to Postman

Sending *POST* and *GET* requests to the endpoints will return 403 Forbidden

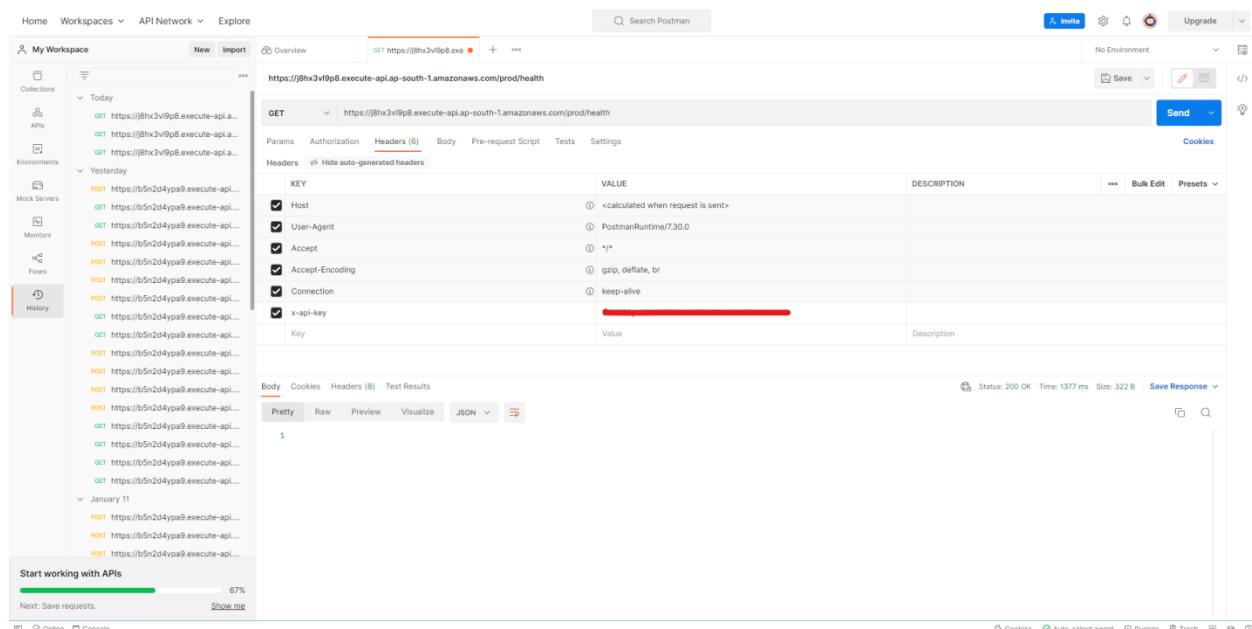


The screenshot shows the Postman interface with a failed GET request to `https://b8nx3v9p8.execute-api.ap-south-1.amazonaws.com/prod/health`. The response status is 403 Forbidden, and the response body contains the message "message": "Forbidden".

Go to Headers

Create a new header with Key "**x-api-key**" and set Value to the [API key](#)

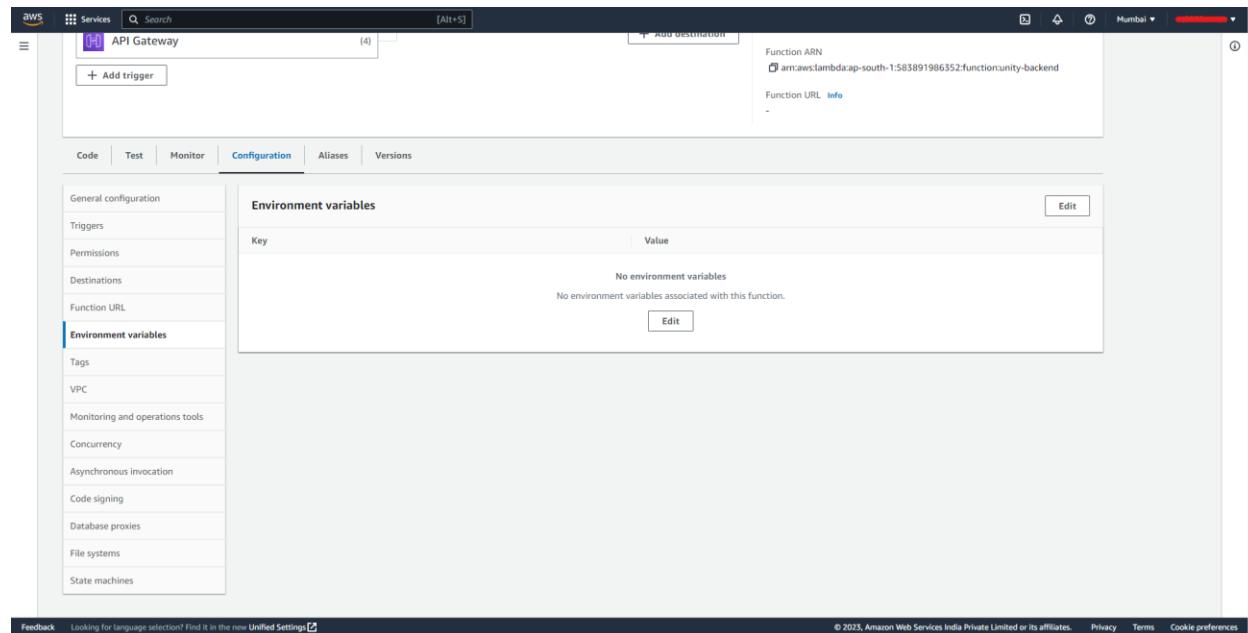
Send request to get response 200 OK



The screenshot shows the Postman interface with a successful GET request to `https://b8nx3v9p8.execute-api.ap-south-1.amazonaws.com/prod/health` after adding the "x-api-key" header. The response status is 200 OK.

*Next, we update the code for Lambda function.
Upload the Backend folder as a zip file to Lambda*

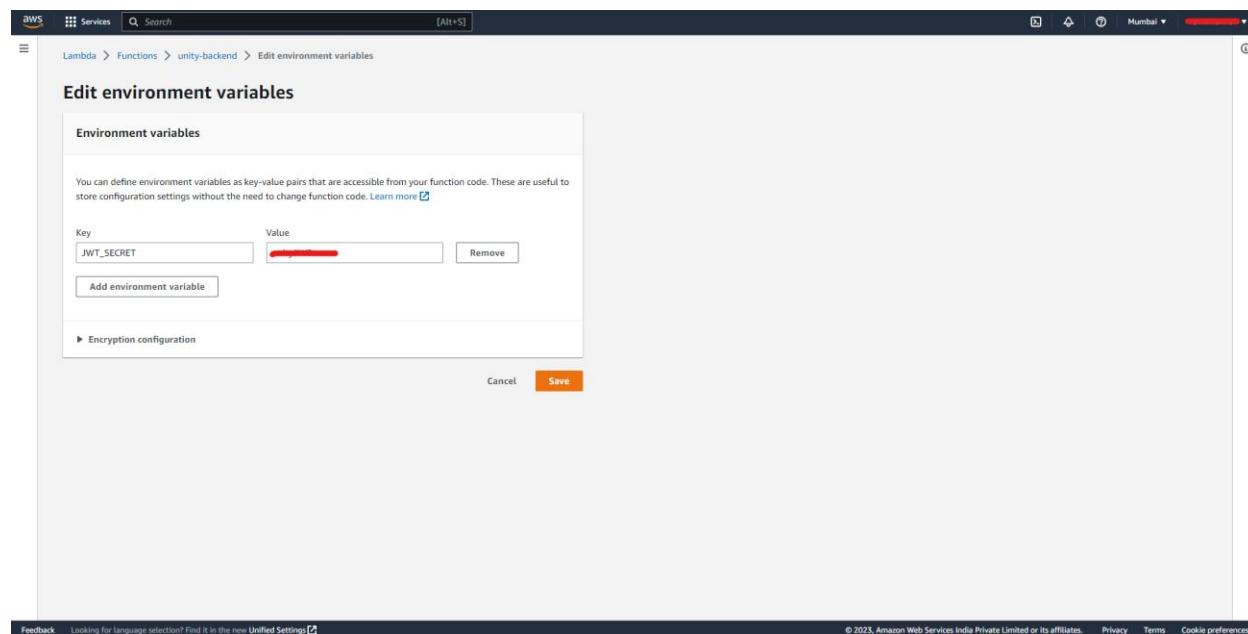
Setting Environment Variable



The screenshot shows the AWS API Gateway configuration interface. The left sidebar lists various options like General configuration, Triggers, Permissions, Destinations, Function URL, Environment variables (which is selected), Tags, VPC, Monitoring and operations tools, Concurrency, Asynchronous invocation, Code signing, Database proxies, File systems, and State machines. The main panel is titled 'Environment variables' and contains a table with columns 'Key' and 'Value'. A message at the bottom states 'No environment variables' and 'No environment variables associated with this function.' There is an 'Edit' button at the bottom right of the table.

Go to Lambda > Functions > current Lambda Function

Under Configuration > Environment Variables > Edit



The screenshot shows the AWS Lambda function configuration interface. The left sidebar shows the path 'Lambda > Functions > unity-backend > Edit environment variables'. The main panel is titled 'Edit environment variables' and contains a section for 'Environment variables'. It includes a note: 'You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. Learn more' with a link. Below this is a table with columns 'Key' and 'Value'. A row is shown with 'JWT_SECRET' in the Key column and a redacted value in the Value column. There are 'Add environment variable' and 'Remove' buttons. At the bottom, there is an 'Encryption configuration' section and a 'Save' button.

Enter a new Environment Variable and its value and then Save.

Test the register, login and verify endpoints on postman by giving the appropriate http body

```
POST      /register
{
  "name" : "somename",
  "email" : "email@gmail.com",
  "username" : "someusername",
  "password" : "somepassword"
}

POST      /login
{
  "username" : "someusername",
  "password" : "somepassword"
}

POST      /verify
{
  "user" : {
    "username" : "someusername",
    "name" : "somename"
  },
  "token" : "sometoken"
}
```