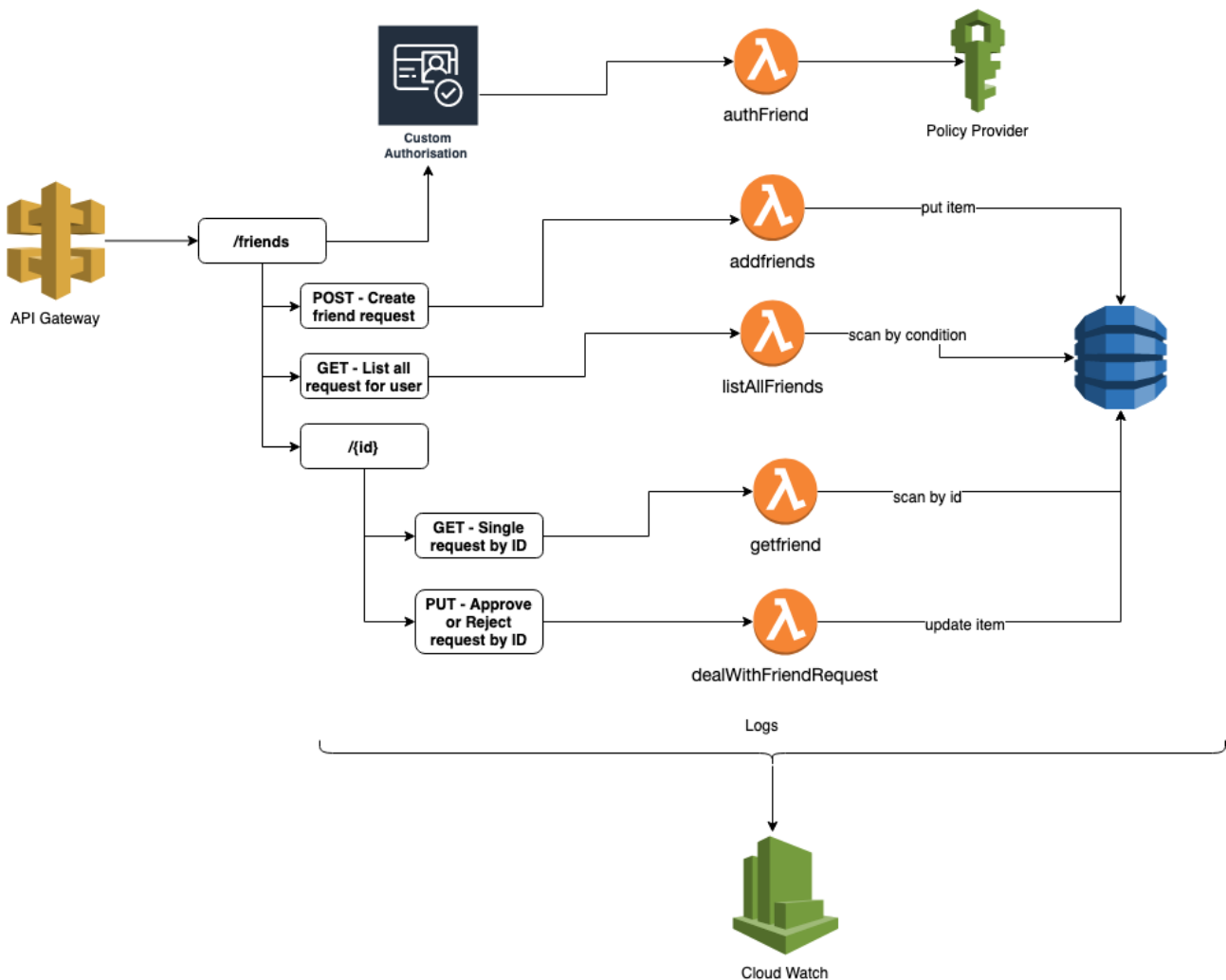


Project Name: DNA Friends App

This application expose rest endpoints for online friends app to create new friend request, allows user to list all friend request and either approve or reject request.

Application build around **serverless** architecture by using **AWS** service such as **API Gateway** (Rest interface), **Lambda** (NodeJS) and **DynamoDB** (storage).

All rest endpoints are secured using **custom authorisation** module which generate appropriate policies for valid users.



Service Description

Create Friend Request	POST https://hoeezamirc.execute-api.eu-west-2.amazonaws.com/development/friends	<p>Header:</p> <p>Content-Type: application/json</p> <p>Authorization: allow</p> <p>Notes: Authorization is a simulation, you can change allow to deny to simulate unauthorization scenario.</p>	<p>Request Body (Sample):</p> <pre>{ "sender" : { "id" : "ABC999", "name" : "Ashely" }, "receiver" : { "id" : "ABC34343", "name" : "Mick" } }</pre> <p>Response: 200 OK with</p> <pre>{ "statusCode": 200, "body": "{\\"location\\": \\"friendsdna_ofvftp95gmo0\\",\\"sender\\": \\"Ashely\\",\\"status\\": \\"Pending\\"}" }</pre>
List all friends by user	GET https://hoeezamirc.execute-api.eu-west-2.amazonaws.com/development/friends/	<p>Content-Type: application/json</p> <p>Authorization: allow</p> <p>auth-user: <receiver user name used in previous request>, this is to show that the user is extracted from header. in real scenario user will be coming from authorisation token.</p>	<p>Response:</p> <pre>[{ "requestId": "dna_ofvftp95gmo0", "sender": { "id": "ABC999", "name": "Ashely" }, "receiver": { "id": "ABC34343", "name": "Mick" }, "status": "Pending" }]</pre> <p>Note: more down request id, this will be used in future request when we approved or reject friend requests.</p>

Get single friend request by id	https://hooezamirc.execute-api.eu-west-2.amazonaws.com/development/friends/dna_ofvftp95gmo0 Note: replace dna_id with the id from previous response.	Content-Type: application/json Authorization: allow	Response: <pre>{ "requestId": "dna_ofvftp95gmo0", "sender": { "id": "ABC999", "name": "Ashely" }, "receiver": { "id": "ABC34343", "name": "Mick" }, "status": "Pending" }</pre>
Approved or Reject friend request	https://hooezamirc.execute-api.eu-west-2.amazonaws.com/development/friends/dna_ofvftp95gmo0 Note: replace dna_id with the id from previous response.	Content-Type: application/json Authorization: allow	Request Body: <pre>{ "status" : "Approved" }</pre> or <pre>{ "status" : "Rejected" }</pre> <p>This is also validated against model which is configured in API Gateway and has only 2 value “Approved/ Rejected”, you can test by giving any other value and you should see 400 bad request error.</p> <p>Response for approval:</p> <pre>"{"message\":"Request has been Approved"}"</pre> <p>Response for rejection:</p> <pre>"{"message\":"Request has been Rejected"}"</pre>

Postman configuration:

Create friend request

POST [/development/friends](#) HTTP/1.1
Host: hoeezamirc.execute-api.eu-west-2.amazonaws.com
Content-Type: application/json
Authorization: allow
Cache-Control: no-cache
Postman-Token: aa5ce4ad-04e1-4a7a-8dd8-d996448110af

```
{
  "sender" : {
    "id" : "ABC9991",
    "name" : "Ashely"
  },
  "receiver" : {
    "id" : "ABC343431",
    "name" : "Adrian"
  }
}
```

List all request for the user

GET [/development/friends/](#) HTTP/1.1
Host: hoeezamirc.execute-api.eu-west-2.amazonaws.com
Content-Type: application/json
Authorization: allow
auth-user: Adrian
Cache-Control: no-cache
Postman-Token: c9a5686b-ef67-45e6-b7e4-19f7d95b1a96

Get single request by request id

GET [/development/friends/dna_ofvftp95gmo0](#) HTTP/1.1
Host: hoeezamirc.execute-api.eu-west-2.amazonaws.com
Content-Type: application/json
Authorization: allow
Cache-Control: no-cache
Postman-Token: 52af52fd-4b39-4a9f-b275-6887849560f5

Approve or reject friend request

PUT [/development/friends/dna_ofvftp95gmo0](#) HTTP/1.1
Host: hoeezamirc.execute-api.eu-west-2.amazonaws.com
Content-Type: application/json
Authorization: allow
Cache-Control: no-cache
Postman-Token: b6d77f34-51b9-49a0-b249-b972c80d0384

```
{  
    "status" : "Approved"  
}
```

Artefact In github: <https://github.com/amitmahajan78/friend-request-service>

API Documentation:

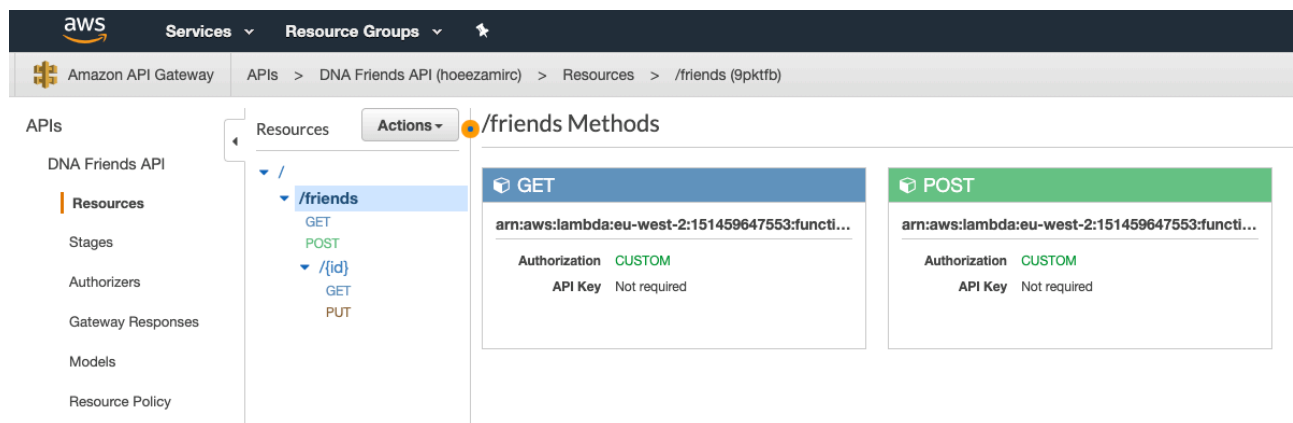
1. DNA Friends API-development-oas30-postman.json
2. DNA Friends API-development-oas30.json

Lambda Function export

1. addfriends.yaml
2. authFriend.yaml
3. dealWithFriendRequest.yaml
4. getfriend.yaml
5. listAllFriends.yaml

Screenshots

API Gateway



API Gateway Authorization module

The screenshot shows the AWS API Gateway console. The left sidebar contains a navigation menu with the following items: APIs, DNA Friends API, Resources, Stages, **Authorizers** (highlighted), Gateway Responses, Models, Resource Policy, Documentation, Dashboard, Settings, Usage Plans, API Keys, Custom Domain Names, and Client Certificates. The main content area is titled 'Authorizers' and includes a sub-header: 'Authorizers enable you to control access to your APIs using Amazon Cognito User Pools or a Lambda function.' Below this is a blue button labeled '+ Create New Authorizer'. A card for an existing authorizer, 'dna-auth-service' (ID: eyd1bo), is displayed. The card details include: 'Lambda Function' (authFriend (eu-west-2)), 'Lambda Event Payload' (Token), 'Token Source' (Authorization), 'Token Validation' (none), and 'Authorization Caching' (Authorization cached for 50 minutes). At the bottom of the card are 'Edit' and 'Test' links.

APIs

- DNA Friends API
 - Resources
 - Stages
 - Authorizers**
 - Gateway Responses
 - Models
 - Resource Policy
 - Documentation
 - Dashboard
 - Settings
- Usage Plans
- API Keys
- Custom Domain Names
- Client Certificates

Authorizers

Authorizers enable you to control access to your APIs using Amazon Cognito User Pools or a Lambda function.

[+ Create New Authorizer](#)

dna-auth-service
Authorizer ID: eyd1bo

Lambda Function
authFriend (eu-west-2)

Lambda Event Payload
Token

Token Source
Authorization

Token Validation
none

Authorization Caching
Authorization cached for 50 minutes

[Edit](#) [Test](#)

Lambda

The screenshot shows the AWS Lambda console 'Functions' page. At the top, there's a search bar and a 'Create function' button. Below is a table listing five functions. The table has columns for Function name, Description, Runtime, Code size, and Last modified. The functions listed are dealWithFriendRequest, addfriends, authFriend, getfriend, and listAllFriends, all using Node.js 10.x runtime.

Functions (5)

Filter by tags and attributes or search by keyword

Function name	Description	Runtime	Code size	Last modified
dealWithFriendRequest		Node.js 10.x	643 bytes	2 hours ago
addfriends		Node.js 10.x	655 bytes	yesterday
authFriend		Node.js 10.x	513 bytes	13 hours ago
getfriend		Node.js 10.x	472 bytes	16 hours ago
listAllFriends		Node.js 10.x	558 bytes	3 hours ago

DynamoDB

DynamoDB

Dashboard

Tables

Backups

Reserved capacity

Preferences

Create tableDelete table

Filter by table name

Choose a table ...Actions

Name

my-friends

my-friendsClose

OverviewItemsMetricsAlarmsCapacityIndexesGlobal TablesBackupsTriggersAccess controlTags

Create itemActions

Scan: [Table] my-friends: Id

Scan[Table] my-friends: IdAdd filterStart search

	Id	FriendId	FriendName	Status	UserId	UserName
	d_0.05018946943334135	1002	tom2	Pending	1002	tom2
	d_0.391030540469546	1004	tom2	Pending	1004	tom2
	d_0.6826403441389153	1005	abc1	Pending	1004	tom2
	dna_1lsrom4qztuo	ABC1001	John	adadas	XCV009	PaulH
	dna_2cfauydfbvfo	1001	abc1	Pending	1002	tom2
	dna_5w3ooeko2mo0	1005	Paul	Pending	1004	tom2
	dna_7ay2f9bf5jk0	1001	abc1	Pending	1002	tom1
	dna_acc3ratnxu00	1005	abc1	Pending	1004	tom2
	dna_b2f93hrze4c0	1001	abc1	Approved	1002	tom2
	dna_fokw7ulvnhk0	1001	abc1	Pending	1002	tom2

CloudWatch

CloudWatch

Dashboards

Alarms

ALARM

INSUFFICIENT

OK

Billing

Events

Rules

Event Buses

Logs

Insights

Metrics

Settings

Favorites

CloudWatch > Log Groups

Create Metric FilterActions

Filter: Log Group Name Prefix

Log Groups	Insights	Expire Events After	Metric Filters	Subscriptions
<input type="radio"/> /aws/lambda/addFriends	Explore	Never Expire	0 filters	None
<input type="radio"/> /aws/lambda/authFriend	Explore	Never Expire	0 filters	None
<input type="radio"/> /aws/lambda/dealWithFriendRequest	Explore	Never Expire	0 filters	None
<input type="radio"/> /aws/lambda/getfriend	Explore	Never Expire	0 filters	None
<input type="radio"/> /aws/lambda/listAllFriends	Explore	Never Expire	0 filters	None