miniurl - URL shortener service for eatigo

Features

- Create unique short URL for each request URL/Link.
- Has cache support which will remain stored for 30 days. Cache will refresh for 30 days again if it is used before expiration.
- Basic authentication service implemented with jwt token.
- Unit test added.
- Has docker support for both local and prod environment with separate configuration.

Tool & Reason

- Primary database is Dyanmo DB
    - It is a faster & simple key-value store.
    - For this project it is a good option, since in my design i have simple model to save. I am generating unique short id(key) for each requested url and save them in dynamo. It will be fast to fetch them by the id since it is primary key.
    - I am generating unique id for each request because it will keep the implementation simple & we have enough space to accommodate more than ~100 years data in 100k write scale.
- Redis cache
    - docker for local, ElastiCache in AWS for PROD.
    - Redis support TTL (time to live) for cache which is handy.
- Snowfalke Id Generator
    - Snowflake is twitter's popular distributed id generator.
    - I have used this as a package in this project because in real scenario this id generator should be separate microservice.
- OAuth/JWT Token
    - I have created a dummy jwt token from jwt.io website. I have used HMAC algorithm & ***"my_super_secret_key"*** as a secret key to demonstrate that this app can parse jwt token from bearer token and authenticate request.
    - In real scenario, token /refresh token generation will be handled by separate microservice.
- docker-compose file
    - docker-compose.dev.yml file is to support local development whereas docker-compose.prod.yml will support prod deployment.
    - I have listed all necessary environment variables in docker-compose file which should be saved in PROD env variable or CI/CD secret in real scenario.

How to Run

Local Environment

Running below command in console will fetch docker images and run miniurl app on port 9000 & 9001 in local environment:

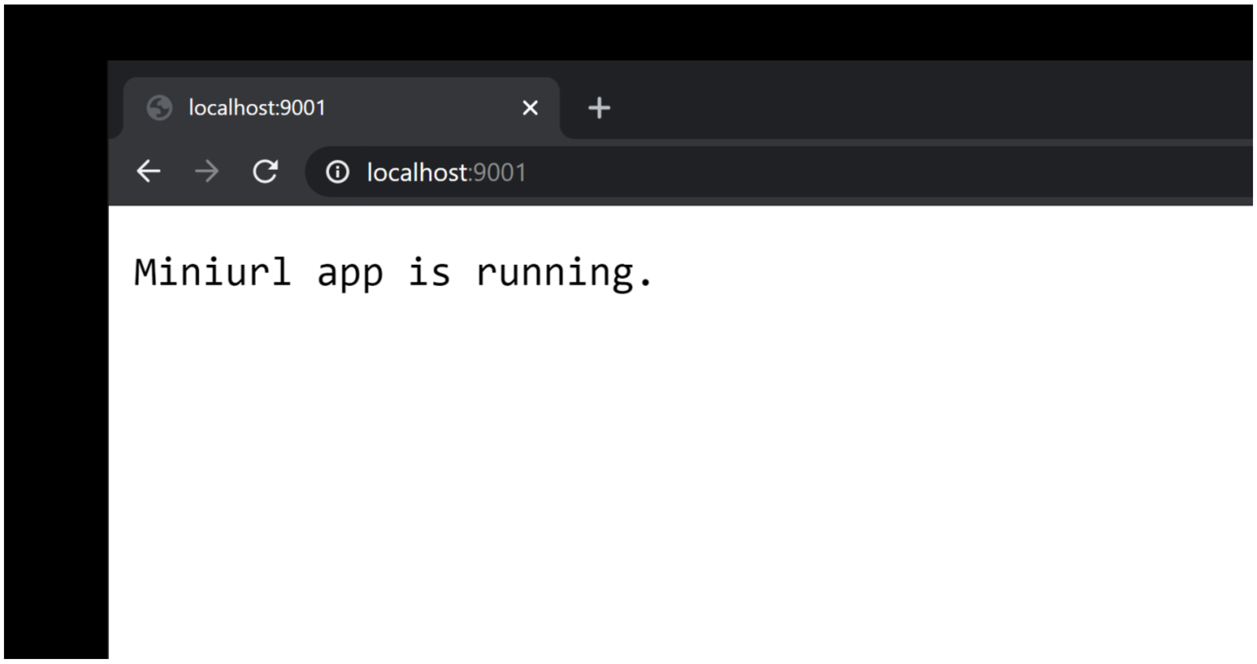
docker-compose -f docker-compose.dev.yml up

Please notice that environment variables are mandatory in docker-compose file:

AWS_REGION=ap-southeast-1
AWS_ACCESS_KEY=XXXXXXXXXX
AWS_SECRET_KEY=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
REDIS_URL=redis:6379
REDIS_TTL=10 //TTL is saved in second(s). locally it is 10 second.
JWT_SECRET=my_super_secret_key //secret key for jwt token

Local environment is using AWS dynamoDB as main db, redis docker as cache db. To connect with the dynamo db, we need the Access & Secret Keys.

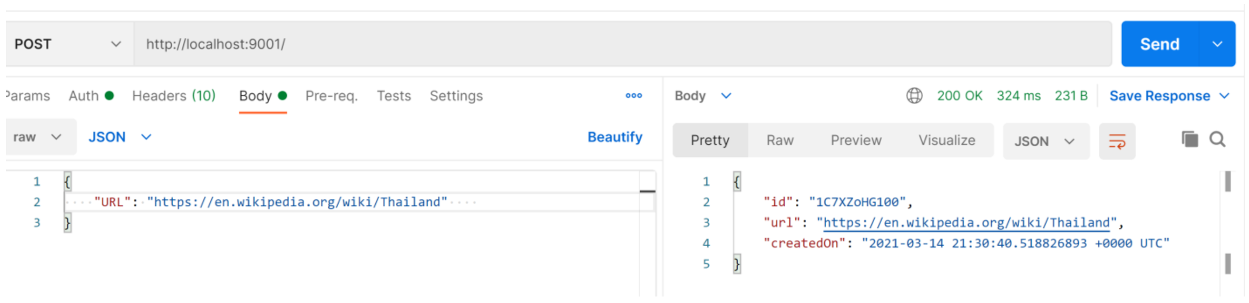
After executing docker-compose, go to localhost: 9001 or 9002.



```
Miniurl app is running.
```

POST Call / Save URL:

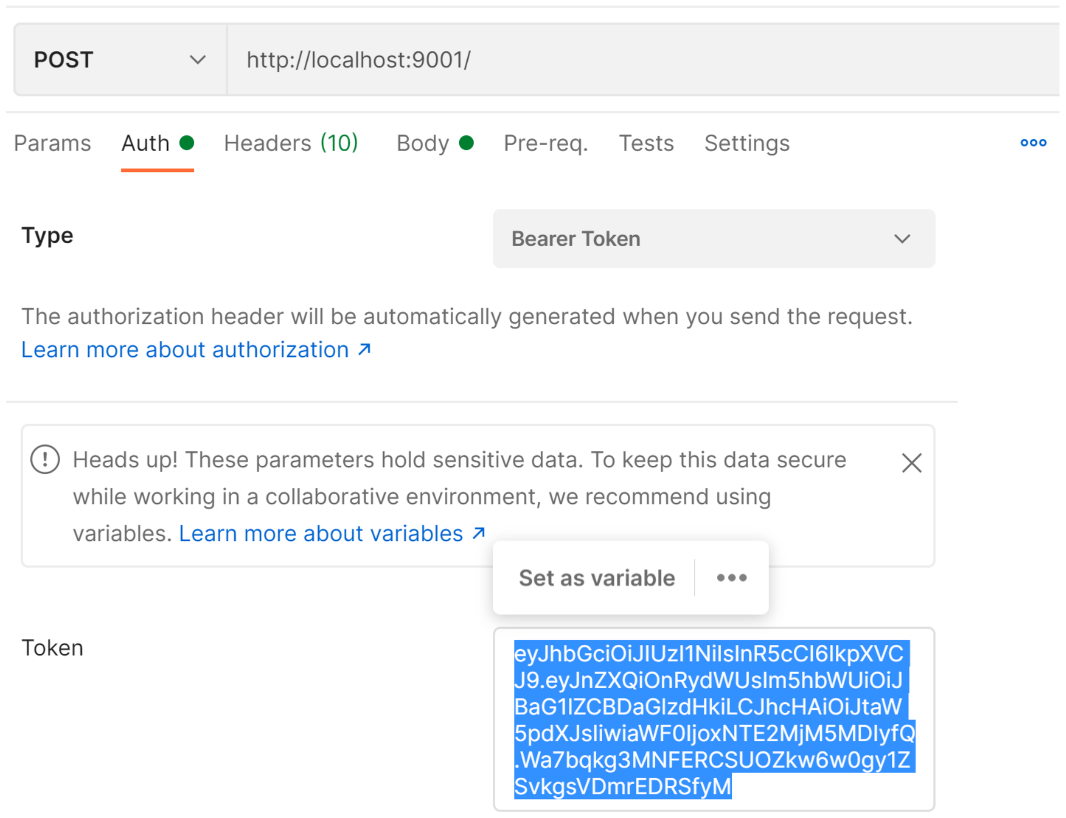
We can now save data using Post call to our app. Using Postman, it will be like below:


POST
http://localhost:9001/
Send
Params Auth Headers (10) Body Pre-req. Tests Settings
raw JSON Beautify
Body 200 OK 324 ms 231 B Save Response
Pretty Raw Preview Visualize JSON

```
{
    "URL": "https://en.wikipedia.org/wiki/Thailand"
}
```

```
{
    "id": "1C7XZoHG100",
    "url": "https://en.wikipedia.org/wiki/Thailand",
    "createdOn": "2021-03-14 21:30:40.518826893 +0000 UTC"
}
```

Before that we have to set our bearer token here:



Type: Bearer Token

The authorization header will be automatically generated when you send the request.
Learn more about authorization ↗

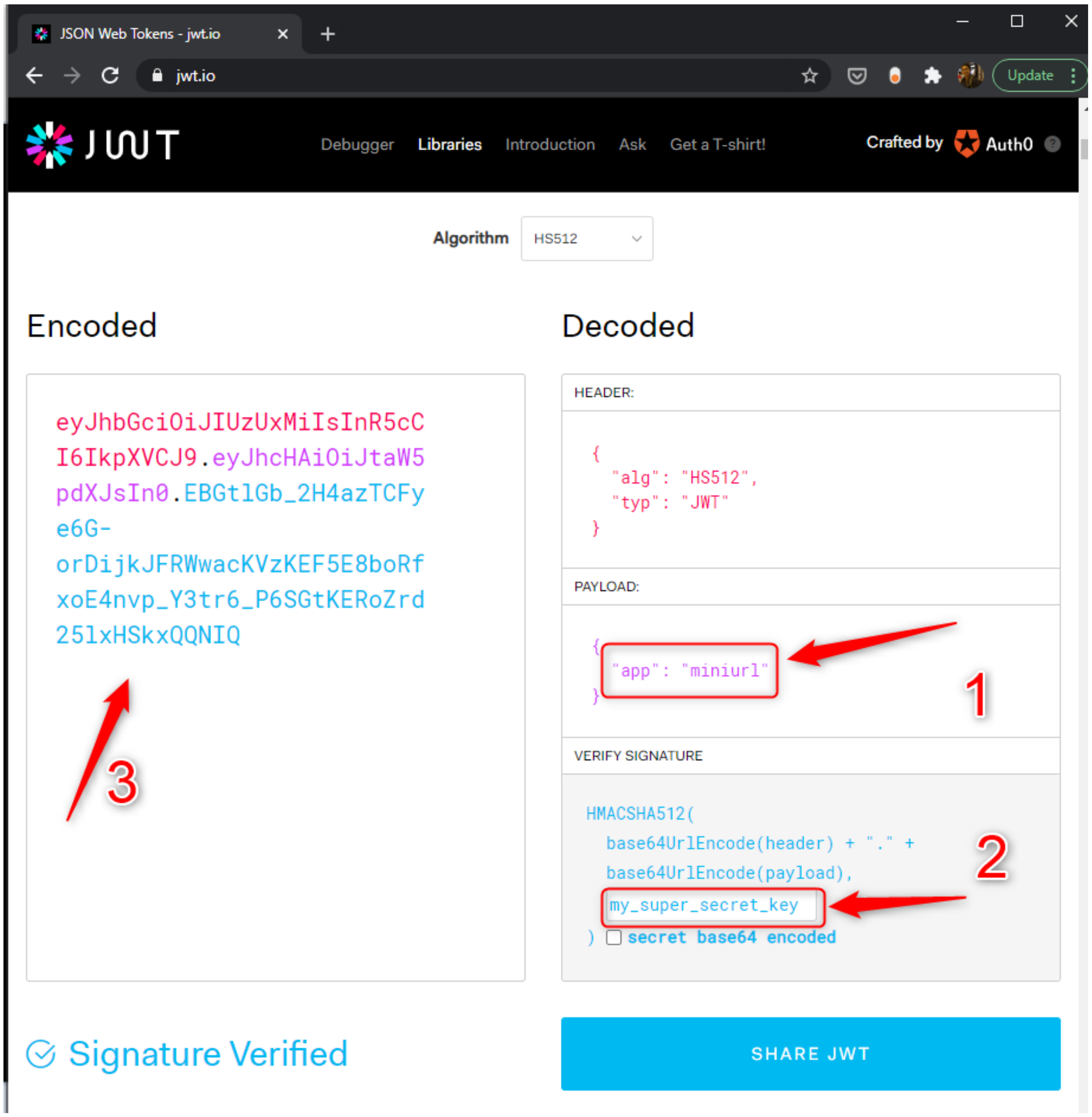
Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables ↗

Token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJnZXQiOnRydWUsIm5hbWUiOiJBaG1lZCBDaGlzdHkiLCJhcHAiOiJtaW5pdXJsIiwiaWF0IjoxNTE2MjM5MDIyfQ.Wa7bqkg3MNFERCSUOZkw6w0gy1ZSvkgsVDmrEDRSfyM

For now we can use the defined token below as bearer token

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhcHAiOiJtaW5pdXJsIn0.lYWHpDPJYf5NjJQlYCLgzmiOeqfKLa7V5qEheyTmOUc

JWT token can be generated from jwt.io webpage. *miniurl* app only checks if it has any claim with ***id: "app" & value: "miniurl"***. Currently it does not check any expiration data. Notice, the secret key is defined as ***"my_super_secret_key"***



Algorithm HS512

Encoded

```
eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJhcHAiOiJtaW5pdXJsIn0.EBGtlGb_2H4azTCFye6G-orDijkJFRWwacKVzKEF5E8boRfxoE4nvp_Y3tr6_P6SGtKERoZrd25lxHSkxQQNIQ
```

Decoded

HEADER:

```
{
  "alg": "HS512",
  "typ": "JWT"
}
```

PAYLOAD:

```
{
  "app": "miniurl"
}
```

VERIFY SIGNATURE

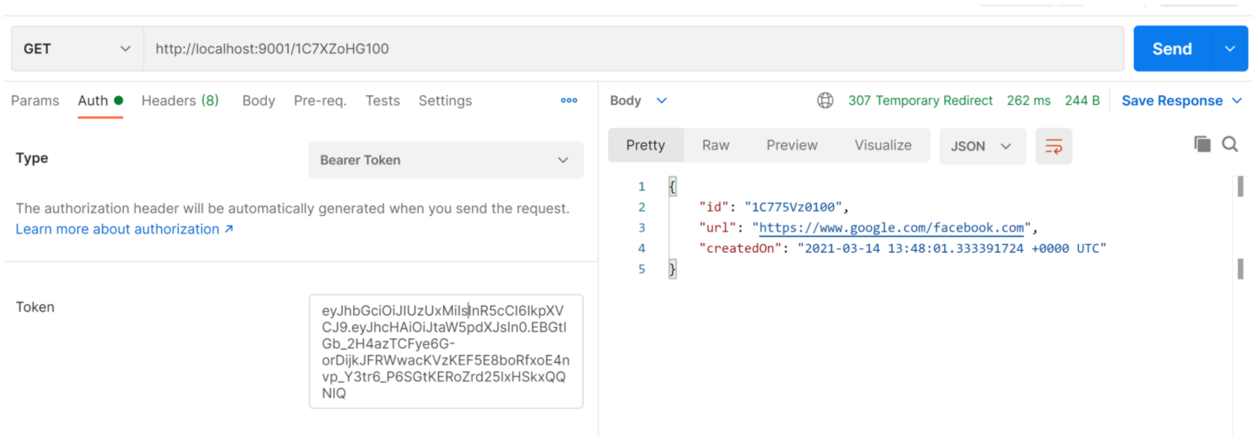
```
HMACSHA512(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  my_super_secret_key
) ☐ secret base64 encoded
```

Signature Verified

Copying the token from Encoded section (3) and putting it as bearer token will be sufficient for now. Both ***GET & POST*** is validating our basic JWT token.

GET Call/ Fetch URL:

Setting the jwt token & putting **{id}** in the route will fetch the stored long URL



http://localhost:9001/1C7XZoHG100

307 Temporary Redirect 262 ms 244 B

Type: Bearer Token

The authorization header will be automatically generated when you send the request.
Learn more about authorization ↗

Token: eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJhcHAiOiJtaW5pdXJsIn0.EBGtIGb_2H4azTCFye6G-orDijkJFRWwacKVzKEF5E8boRfxoE4nvp_Y3tr6_P6SGtKERoZrd25IxHSkxQQNIQ

```
{
    "id": "1C775Vz0100",
    "url": "https://www.google.com/facebook.com",
    "createdOn": "2021-03-14 13:48:01.333391724 +0000 UTC"
}
```

For now, app is returning 307 (temporary redirect) with full response.

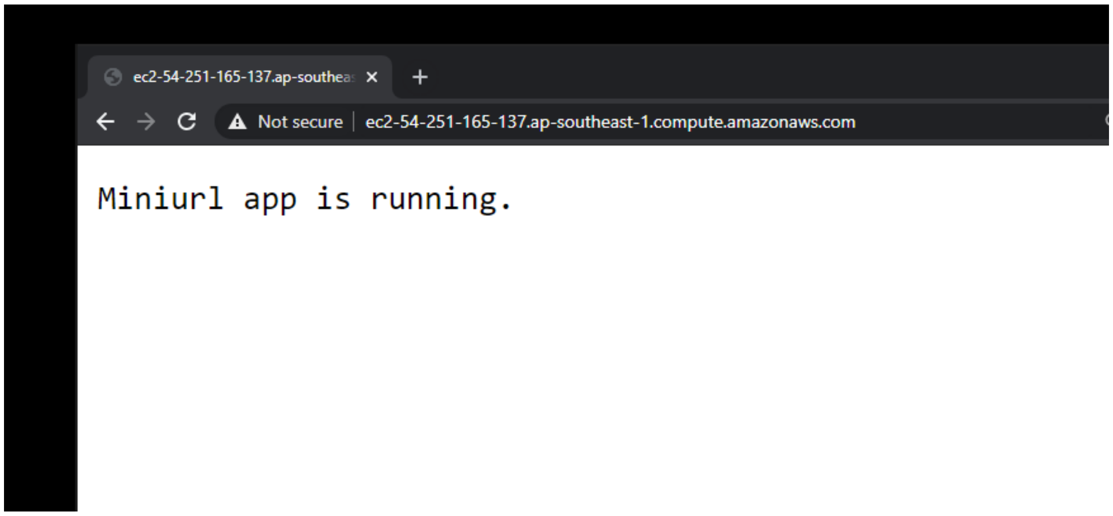
PROD Environment

In production, miniurl is running on AWS free tier only.

- Deployed on a docker container inside free ec2 instance.
- PROD and Local app using same dynamo db table.
- PROD using AWS ElastiCache (redis) free version.

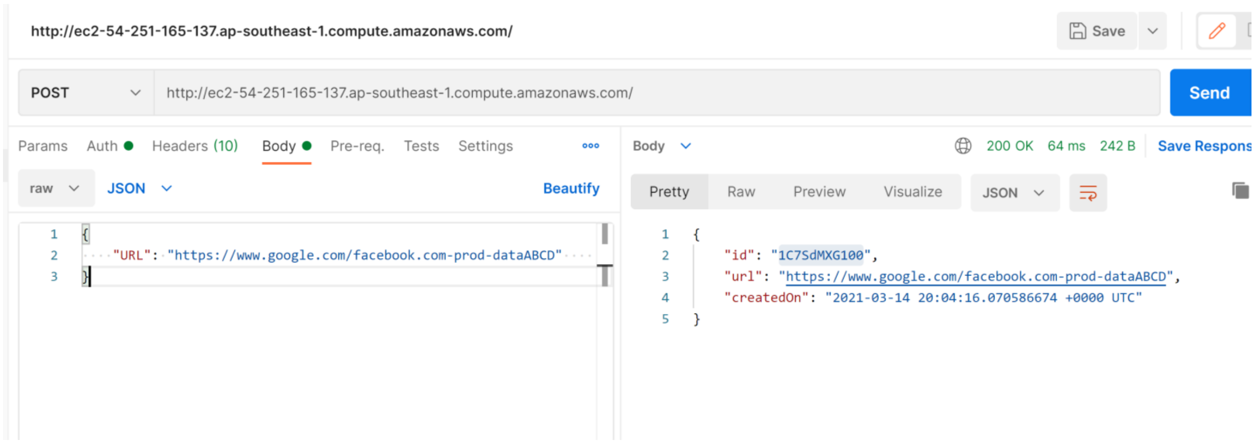
Testing PROD instance is same like local instance. Both needs same bearer token now. PROD web URL:

http://ec2-54-251-165-137.ap-southeast-1.compute.amazonaws.com/


ec2-54-251-165-137.ap-southea
Not secure | ec2-54-251-165-137.ap-southeast-1.compute.amazonaws.com

```
Miniurl app is running.
```

POST data in PROD:



http://ec2-54-251-165-137.ap-southeast-1.compute.amazonaws.com/

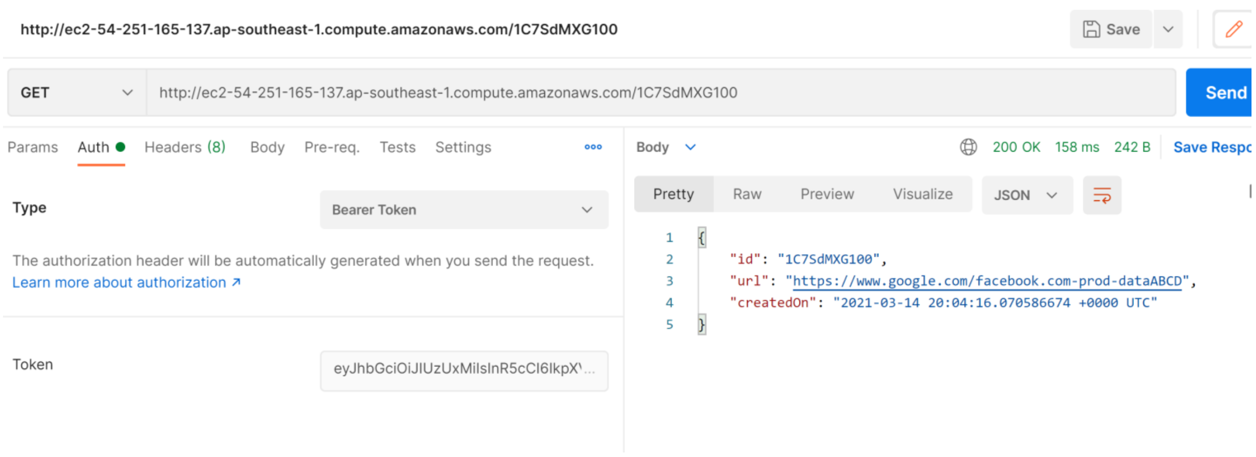
POST http://ec2-54-251-165-137.ap-southeast-1.compute.amazonaws.com/

Save
Send
Params Auth Headers (10) Body Pre-req. Tests Settings
raw JSON Beautify
Body 200 OK 64 ms 242 B Save Respons
Pretty Raw Preview Visualize JSON

```
{
    "URL": "https://www.google.com/facebook.com-prod-dataABCD"
}
```

```
{
    "id": "1C7SdMXG100",
    "url": "https://www.google.com/facebook.com-prod-dataABCD",
    "createdOn": "2021-03-14 20:04:16.070586674 +0000 UTC"
}
```

GET data in PROD:



http://ec2-54-251-165-137.ap-southeast-1.compute.amazonaws.com/1C7SdMXG100

GET http://ec2-54-251-165-137.ap-southeast-1.compute.amazonaws.com/1C7SdMXG100

Save
Send
Params Auth Headers (8) Body Pre-req. Tests Settings
Body 200 OK 158 ms 242 B Save Respo
Pretty Raw Preview Visualize JSON

Type: Bearer Token

The authorization header will be automatically generated when you send the request.
Learn more about authorization ↗

Token: eyJhbGciOiJIUzUxMiIsInR5cCI6IkpX\...

```
{
    "id": "1C7SdMXG100",
    "url": "https://www.google.com/facebook.com-prod-dataABCD",
    "createdOn": "2021-03-14 20:04:16.070586674 +0000 UTC"
}
```

Please feel free to knock me for further details.

Regards

Z Ahmed Chisty