**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=XXXXXXXXXXXXXXXXXXXXXXXXXXXXX*

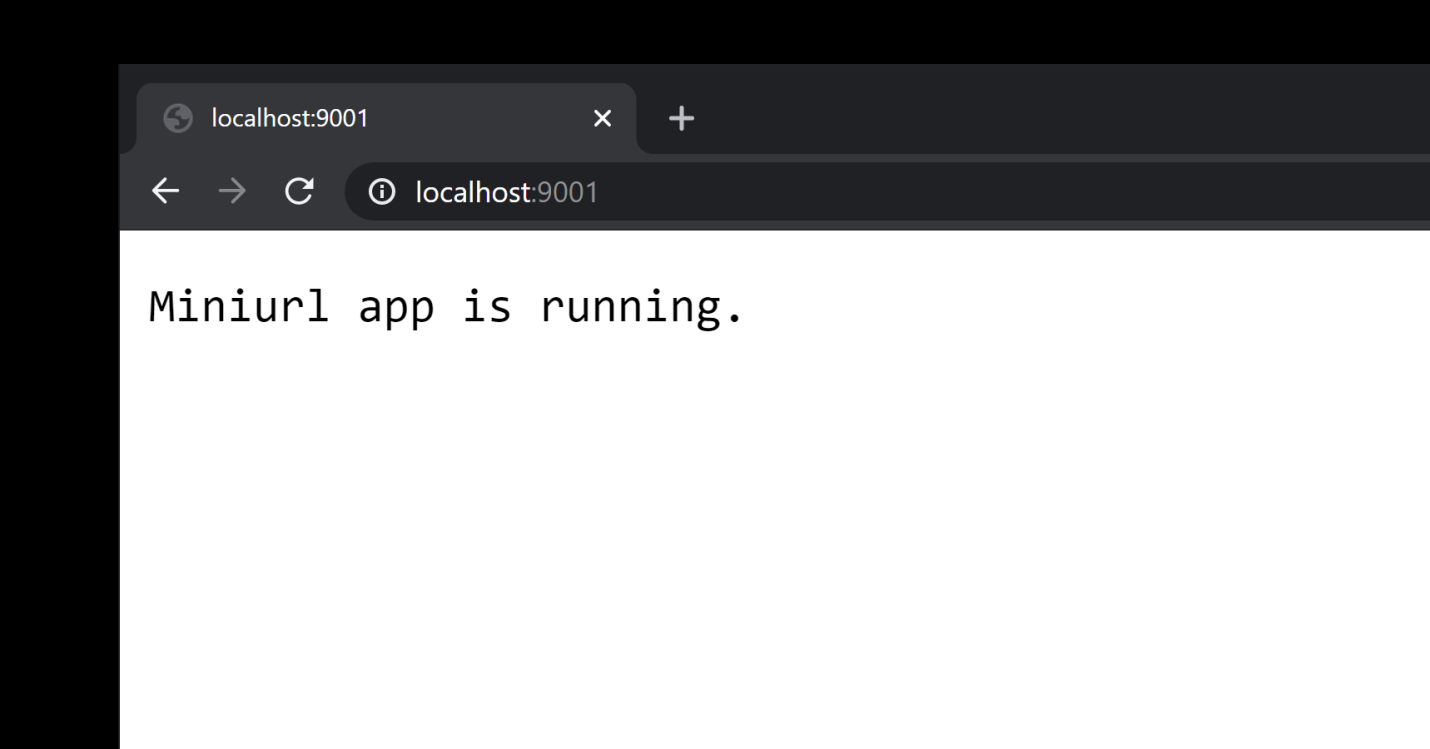
*REDIS\_URL=redis:6379*

*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*

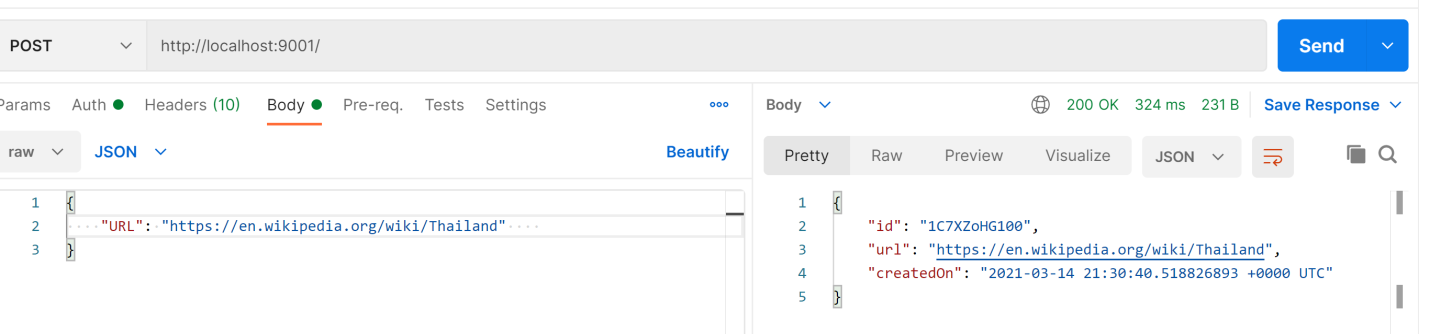
Local environment is using AWS dynamoDB as main db, redis docker as cache db. To connect with the dynamo db, we need the Access & Secret Keys.

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

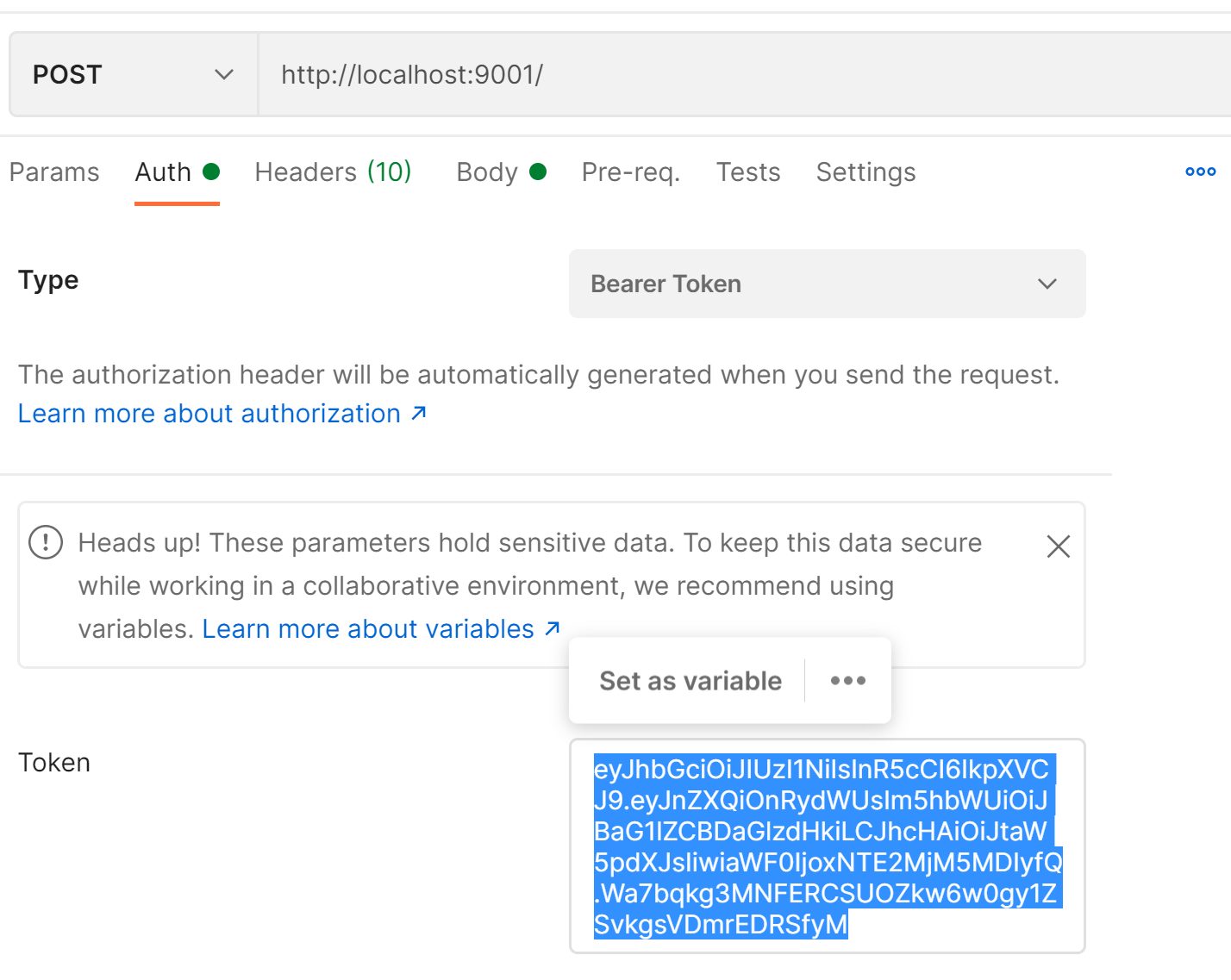


**POST Call/ Save URL:**

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



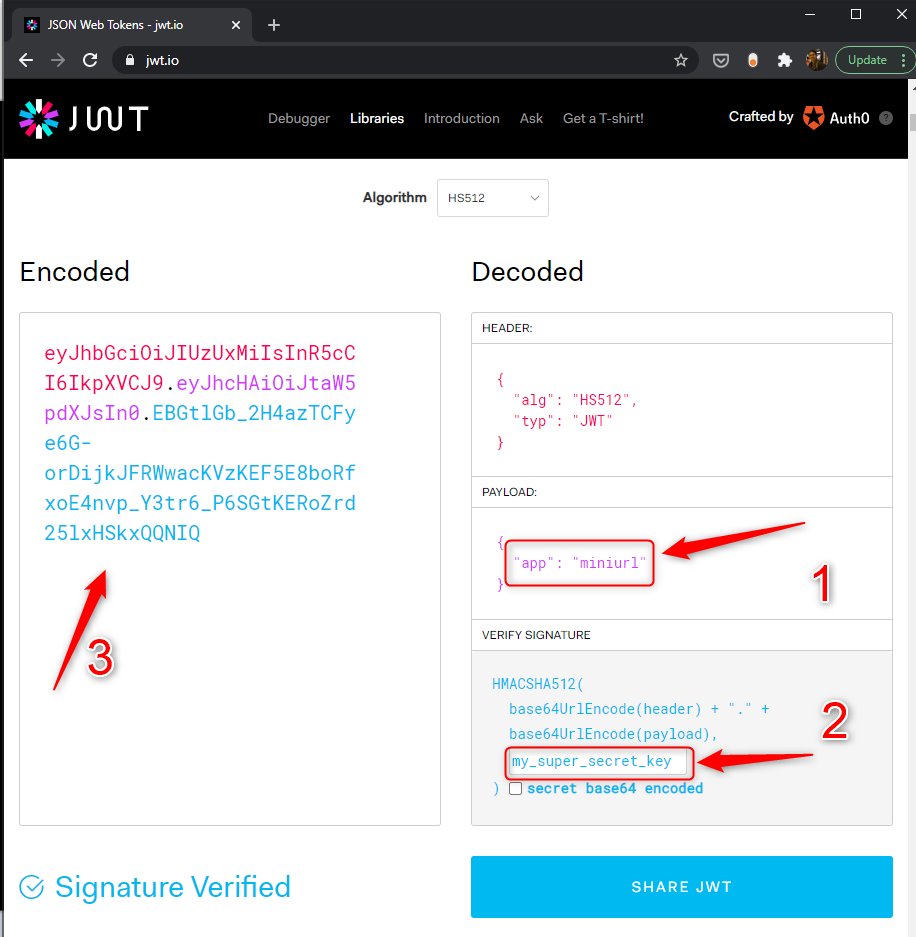
Before that we have to set our bearer token here:



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

***eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhcHAiOiJtaW5pdXJsIn0.lYWHpDPJYf5NjJQlYCLgzmiOeqfKLa7V5qEheyTmOUc***

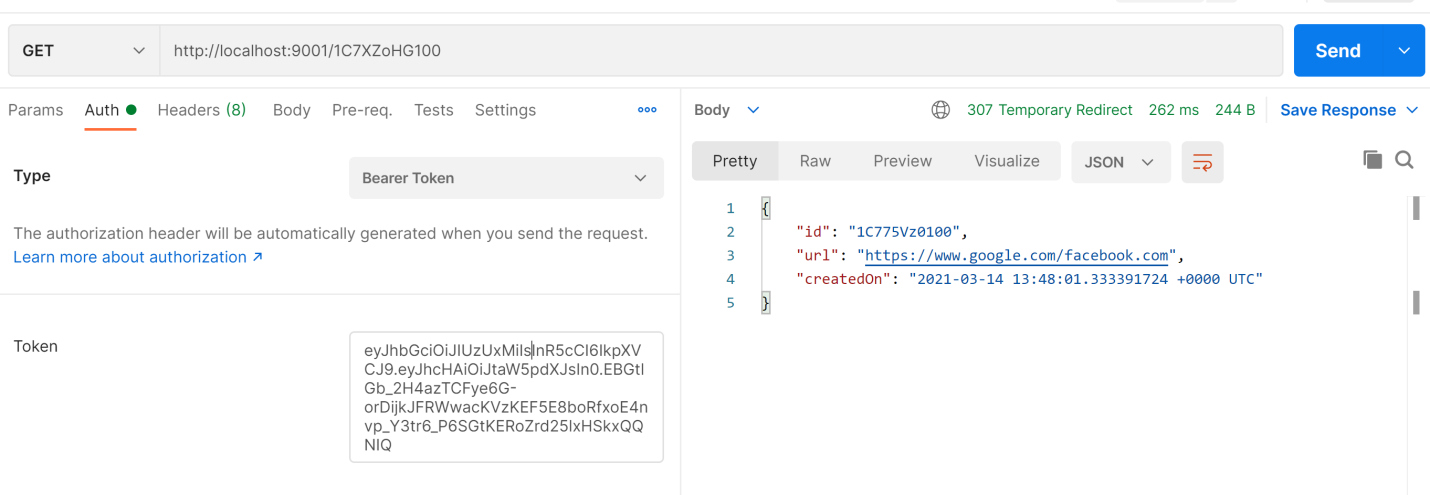
JWT token can be generated from jwt.io webpage. *miniurl* app only checks if it has any claim with ***id: “app” & value: “miniurl”***. Currently it does not check any expiration data. Notice, the secret key is defined as ***“my\_super\_secret\_key”***



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



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

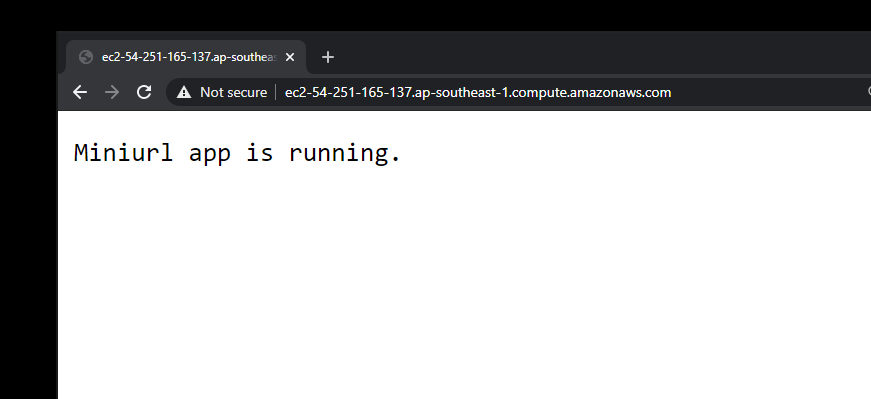
**PROD Environment**

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

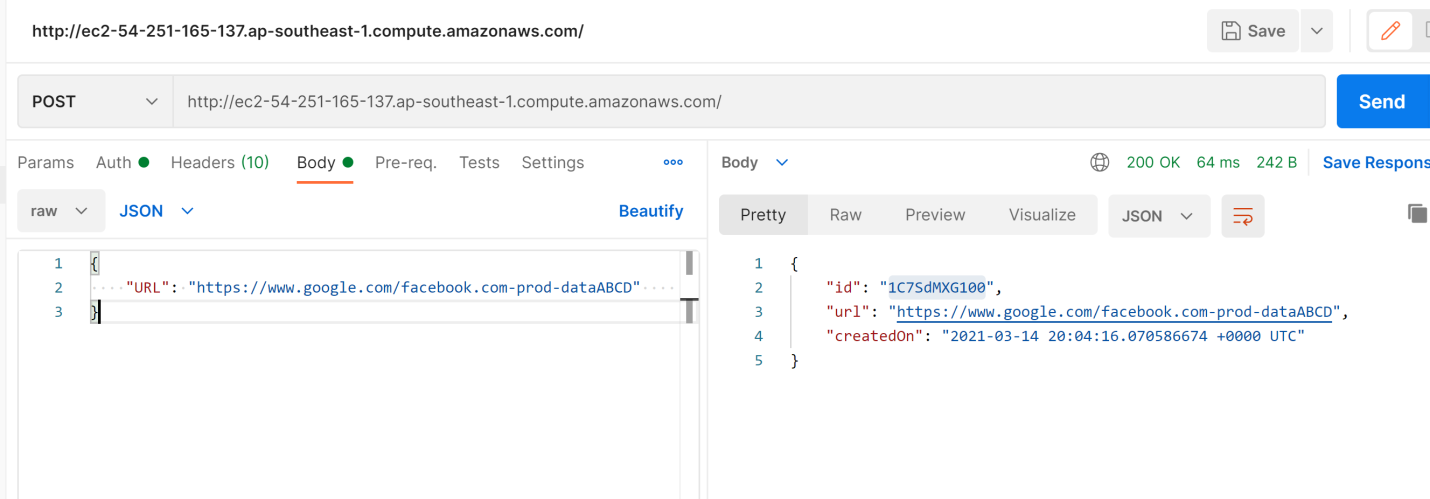
* Deployed on a docker container inside free ec2 instance.
* PROD and Local app using same dynamo db table.
* PROD using AWS ElastiCache (redis) free version.

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

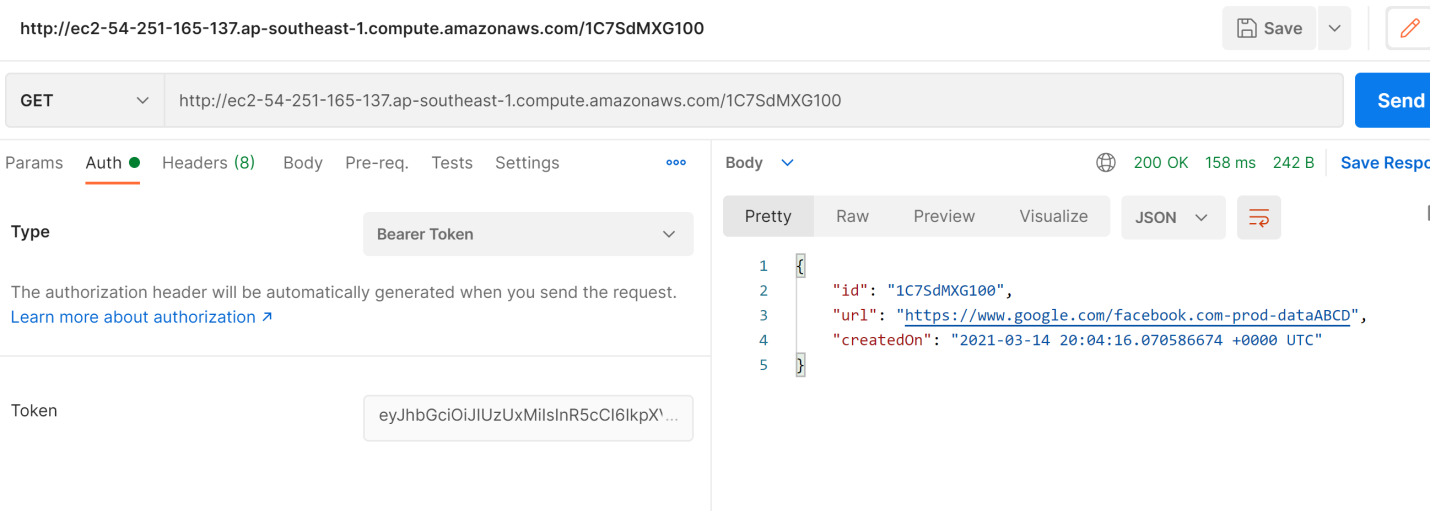
[***http://ec2-54-251-165-137.ap-southeast-1.compute.amazonaws.com/***](http://ec2-54-251-165-137.ap-southeast-1.compute.amazonaws.com/)



POST data in PROD:



GET data in PROD:



Please feel free to knock me for further details.

Regards

Z Ahmed Chisty