Challenge

Product management just came back from a security conference and saw people using free threat intel sources to do detections. Using DNS requests, you can send an IP address in reverse notation to determine whether it is on a block list or not.

Naturally, they want us to be able to query these servers and provide answers to our customers on whether or not an IP is malicious.

Considerations

Given an IP address of 1.2.3.4 we can look that up at Spamhause by issuing the following commands (just an example):

```
host 4.3.2.1.zen.spamhaus.org
```

Or

```
dig 4.3.2.1.zen.spamhaus.org
```

Response Codes: (https://www.spamhaus.org/faq/section/DNSBL%20Usage#200)

For now, we only care about IPv4 Addresses.

You can find example IPs from the Spamhaus ROKSO list (example: https://www.spamhaus.org/rokso/evidence/ROK11315/counterfeit-products-spam-gang/main-info)

Requirements

- Your API must be protected with basic authentication
  - Username: secureworks
  - Password: supersecret
- You should have a GraphQL API that is available via a `/graphql` endpoint
  - We suggest using https://gqlgen.com/
- Mutations
  - `enqueue(ip: ["ip1", "ip2"])`
    - This should kick off a background job to do the DNS lookup and store it in the database for each IP passed in for future lookups
    - If the lookup has already happened, this should queue it up again and update the `response` and `updated_at` fields
- Queries
  - `getIPDetails(ip: "ip address here")`
    - It should look up the IP in the database
    - If it's not there, it should respond with the appropriate response code

        - This should have:
            - `uuid` ID
            - `created_at` time
            - `updated_at` time
            - `response_code` string
            - `ip_address` string
- You should use SQLite as your database to make this portable
- Dependencies should be handled using `go mod`
- Tests should be written for the key components of the system
- A README is required and should explain how to develop and run the project as if it were a new team member working on it
- The application should be packaged as a Dockerfile, and should accept a PORT environment variable to know which port to run on
- You can use any external libraries you want, **but you must document and explain why you're using them**