Elasticsearch

Presented By: Scott Harris and Noah Dunn

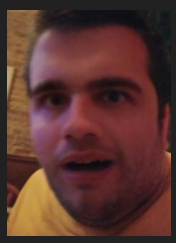
Introductory Information

- System Name: **Elasticsearch**
- Primary Database Model: **Search Engine Database**
- Host Company: **Elasticsearch B.V (AKA Elastic).**
- URL: Link to the Company

The Authors

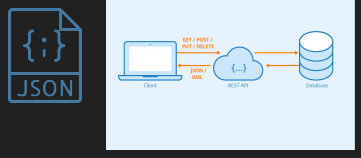
Scott Harris:

Noah Dunn:



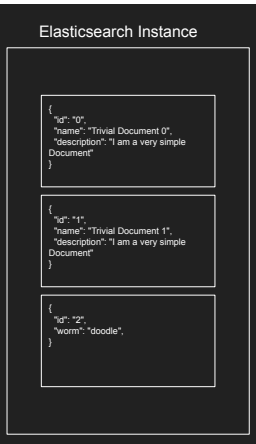
System Overview

- History
  - Based on Apache Lucene
  - First released in 2010
- Intended Usage
  - REST API
  - Retrieving Small Doses of Information
  - Website Searching
  - Focus on Read Operations
- Special Feature(s)
  - Fuzzy or Incomplete Querying
  - Optimized for Data Retrieval
  - Document Store



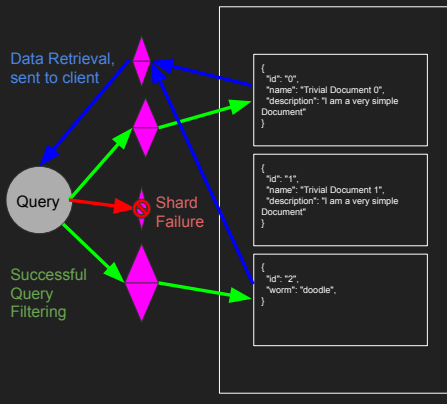
Document Store

- Elasticsearch makes use of JSON Document-Store format
- The ‘Database’ consists of the entire collection of all Documents
- Much like Mongo or other NoSQL solutions, these documents are unstructured and can contain a wide array of keys and values

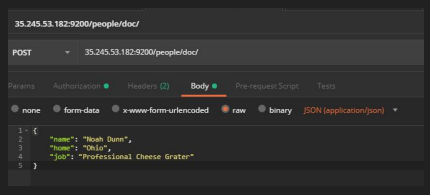


Sharding Visual

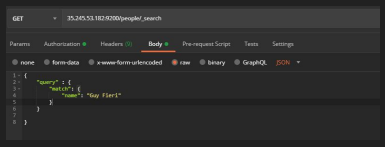
- All Database Processing in Elasticsearch is done via what are called ‘shards’
- Shards are processing units of various sizes that can handle retrieval for queries
- Shards can merge together or split for increased distribution or increased processing power
- Shards provide ‘fault-tolerance’. If one process fails, the other shards will pick up the slack



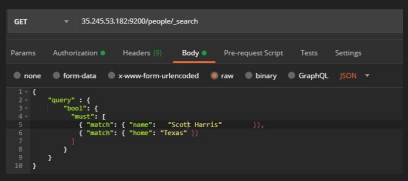
INSERT Query Example



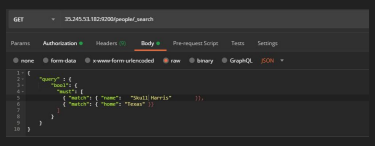
SELECT Query Example

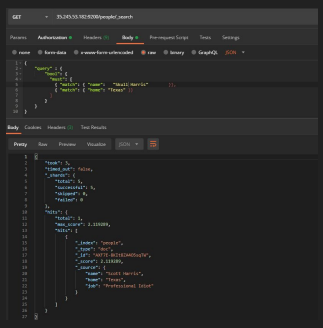


Multi-Conditional Querying



Fuzzy Matching Example





Links, Description, Relevant Documentation

- Official Website
  - https://www.elastic.co/
- Querying Documentation
  - https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html
- A More Complex Look at Sharding
  - https://buildingvts.com/elasticsearch-architectural-overview-a35d3910e515
- Guide: Installation to Querying
  - https://medium.com/@factoryhr/elasticsearch-introduction-implementation-and-example-17dd66c35c35
- Link to Where to Get Install Packages
  - https://www.elastic.co/guide/en/elasticsearch/reference/current/install-elasticsearch.html

Challenges, Issues, Things to keep in mind

- Things Elasticsearch doesn't do well or at all
  - Deletion (Finicky)
  - Joining different documents (Not possible from our research)
  - Bulk insertion
  - There is no custom client offering. Everything is done in REST calls, which can be a negative to those unfamiliar with HTTP or REST.
- Potential DevOPS Issues
  - Google Cloud consumes credit for an instance very quickly
  - Local Installs have to be manually started for use
  - Shard allocation can lead to consuming a large portion of CPU
- When Querying
  - There are at least two ways to write each query: POST body call, and URL modification
  - Do what you find easier

Video Presentation

https://www.youtube.com/watch?v=aSJ5yqS02qQ&feature=youtu.be

Elasticsearch Data Demands(

Created by: Scott Harris and Noah Dunn

Dataset Description:

The documents provided in the sample data take the form:

```
{
  "first_name": "Cesar",
  "last_name": "Yosselevitch",
  "fav_animal": "Hornbill, southern ground",
  "gender": "Male",
  "fav_application": "Sonair"
}
```

All data was pseudo-randomly generated using the website Mockaroo

Database Insertion Information

1. Download this JSON data file.
2. Execute this command in the same folder as the data file. Swapping in the correct values for the text in black.

   ```
   curl -u "[INSERT_USERNAME_HERE_FOR_SERVER]":"[INSERT_PASSWORD_HERE_FOR_SERVER]" -s -XPOST -H "Content-type: application/x-ndjson" [INSERT_SERVER_IP_HERE]/_bulk --data-binary "@MOCK_DATA_FILE.json"
   ```

3. This will generate 1000 documents inside the people index.
4. Validate the documents were successfully inserted using

   ```
   {
           "query": {
                   "match_all" : {}
           },
           "size": 1000
   }
   ```

Data Demand 1:
Determine all Females in the Database whose favorite animal is some type of Turtle (Hint: No regular expression needed) Using Bool, Must, and Match (3 results)

Data Demand 2:
Determine all people who either have a name similar to “alley” within a fuzziness of 2, or are male. Ensure the people found have at least a score of 2 to fit this criteria semi-decently. Make use of min_score, bool, should, match, and fuzziness. (15 results)

Data Demand 3:
Find the top 3 scored people whose first name, last name, or favorite animal is similar to “Kirk” with a fuzziness of 2 using multi_match with type ‘best score’ and a tie breaker of 0.3. Use multi_match, fuzziness, type, fields, tie_breaker, and size. (3 results)

Elasticsearch Data Demands (Teacher)

Created by: Scott Harris and Noah Dunn

Dataset Description:

The documents provided in the sample data take the form:

```
{
  "first_name": "Cesar",
  "last_name": "Yosselevitch",
  "fav_animal": "Hornbill, southern ground",
  "gender": "Male",
  "fav_application": "Sonair"
}
```

All data was pseudo-randomly generated using the website Mockaroo

Database Insertion Information

1. Download this JSON data file.
2. Execute this command in the same folder as the data file. Swapping in the correct values for the text in black.
   curl -u "[INSERT_USERNAME_HERE_FOR_SERVER]":"[INSERT_PASSWORD_HERE_FOR_SERVER]" -s -XPOST -H "Content-type: application/x-ndjson" [INSERT_SERVER_IP_HERE]/_bulk --data-binary "@MOCK_DATA_FILE.json"
3. This will generate 1000 documents inside the people index.
4. Validate the documents were successfully inserted using

```
{
        "query": {
                "match_all" : {}
        },
        "size": 1000
}
```

Data Demand 1:

Determine all Females in the Database whose favorite animal is some type of Turtle (Hint: No regular expression needed) Using Bool, Must, and Match (3 results)

Solution:

POST [IP_ADDRESS]/people/_search

Query:

```
{
    "query": {
        "bool": {
            "must": [
            {
                "match": {
                    "fav_animal": "Turtle"
                }
            },
            {
                "match":{
                    "gender" : "Female"
                }
            }
            ]

        }
    }
}
```

Result:


```
"hits": {
    "total": 3,
    "max_score": 5.5262794,
    "hits": [
        {
            "_index": "people",
            "_type": "descriptions",
            "_id": "820",
            "_score": 5.5262794,
            "_source": {
                "first_name": "Honey",
                "last_name": "Saturley",
                "fav_animal": "Turtle (unidentified)",
                "gender": "Female",
                "fav_application": "Redhold"
            }
        },
        {
            "_index": "people",
            "_type": "descriptions",
            "_id": "53",
            "_score": 4.198447,
            "_source": {
                "first_name": "Ardra",
                "last_name": "Adkins",
                "fav_animal": "Eastern box turtle",
                "gender": "Female",
                "fav_application": "Biodex"
            }
        },
        {
            "_index": "people",
            "_type": "descriptions",
            "_id": "297",
            "_score": 4.198447,
            "_source": {
                "first_name": "Janith",
                "last_name": "Rillatt",
                "fav_animal": "Long-necked turtle",
                "gender": "Female",
                "fav_application": "Fixflex"
            }
        }
    ]
}
```

Data Demand 2:

Determine all people who either have a name similar to “alley” within a fuzziness of 2, or are male. Ensure the people found have at least a score of 2 to fit this criteria semi-decently. Make use of min_score, bool, should, match, and fuzziness. (15 hits)

Solution:

POST [IP_ADDRESS]/people/_search

Query:

```
{
    "min_score": 2,

    "query": {
        "bool":{
            "should": [
                {"match": {"first_name": {"query": "alley", "fuzziness":2} } },
                {"match": {"gender": "Male"} }
                ]
        }
    }
}
```

Response:


```
{
    "took": 15,
    "timed_out": false,
    "_shards": {
        "total": 5,
        "successful": 5,
        "skipped": 0,
        "failed": 0
    },
    "hits": {
        "total": 15,
        "max_score": 4.5856805,
        "hits": [
            {
                "_index": "people",
                "_type": "descriptions",
                "_id": "118",
                "_score": 4.5856805,
                "_source": {
                    "first_name": "Arley",
                    "last_name": "Nussii",
                    "fav_animal": "Asiatic wild ass",
                    "gender": "Male",
                    "fav_application": "Aerified"
                }
            },
            {
                "_index": "people",
                "_type": "descriptions",
                "_id": "937",
                "_score": 3.5954564,
                "_source": {
                    "first_name": "Kelley",
                    "last_name": "Blowes",
                    "fav_animal": "Lemming, collared",
                    "gender": "Male",
                    "fav_application": "Quo Lux"
                }
            },
            {
                "_index": "people",
                "_type": "descriptions",
                "_id": "156",
                "_score": 3.5818043,
                "_source": {
```

Data Demand 3:

Find the top 3 scored people whose first name, last name, or favorite animal is similar to “Kirk” with a fuzziness of 2 using multi_match with type ‘best score’ and a tie breaker of 0.3. Use multi_match, fuzziness, type, fields, tie_breaker, and size. (3 Results)

Query:

Solution:

POST [IP_ADDRESS]/people/_search

Query:

```
{
  "query": {
    "multi_match" : {
      "query":      "Kirk",
      "fuzziness": 2,
      "type": "best_fields",
      "fields":     [ "first_name", "last_name", "fav_animal" ],
      "tie_breaker": 0.3
    }
  },
  "size": 3
}
```

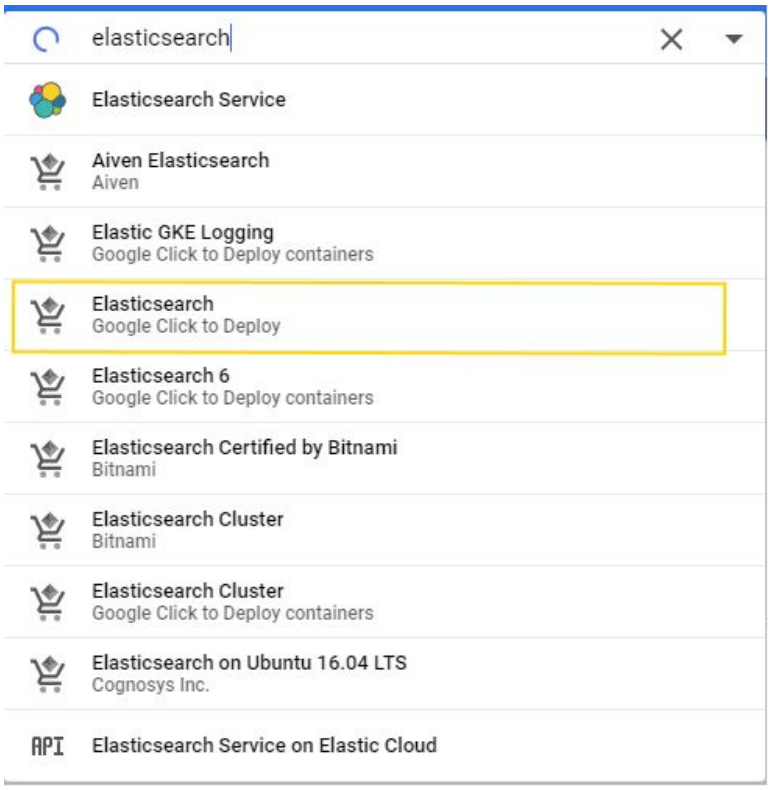
Response:


```
"hits": {
   "total": 31,
   "max_score": 3.6993575,
   "hits": [
      {
         "_index": "people",
         "_type": "descriptions",
         "_id": "951",
         "_score": 3.6993575,
         "_source": {
            "first_name": "Dominic",
            "last_name": "Girk",
            "fav_animal": "Carmine bee-eater",
            "gender": "Male",
            "fav_application": "Sonsing"
         }
      },
      {
         "_index": "people",
         "_type": "descriptions",
         "_id": "346",
         "_score": 3.67338,
         "_source": {
            "first_name": "Kerk",
            "last_name": "Playden",
            "fav_animal": "Bee-eater, nubian",
            "gender": "Male",
            "fav_application": "Overhold"
         }
      },
      {
         "_index": "people",
         "_type": "descriptions",
         "_id": "124",
         "_score": 3.169876,
         "_source": {
            "first_name": "Harald",
            "last_name": "Le Marquand",
            "fav_animal": "Kirk's dik dik",
            "gender": "Male",
            "fav_application": "Vagram"
         }
      }
   ]
}
```

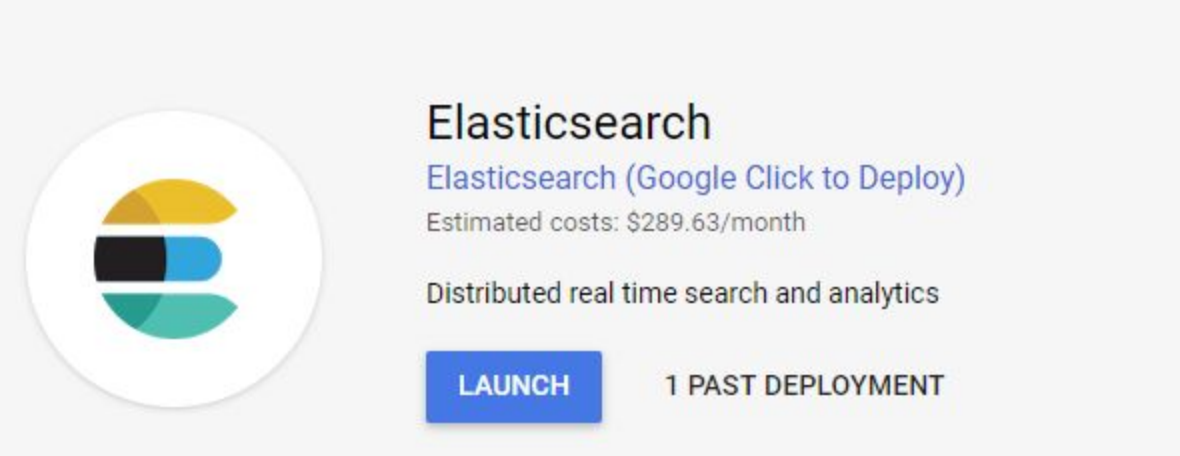
Elasticsearch Installation + Querying Guide

By: Noah Dunn and Scott Harris

1. Log into GCP
2. Select Elasticsearch from Google Click to Deploy in the search bar



3. Hit Launch



4. Configure the instance as follows, changing the deployment name

Deployment name

dunnnm2-test-deployment

Zone

us-east4-a

Cluster Name

Setting a cluster name prevents Elasticsearch nodes in one logical cluster from joining another.

elasticsearch-cluster

Install X-Pack in Elasticsearch

Xpack is monitoring and management software designed for Elasticsearch clusters.

Elasticsearch

Instance Count

1

Machine type

1 vCPU | 3.75 GB memory | Customize

Boot disk type

Standard Persistent Disk

Boot disk size in GB

20

Data disk type

Standard Persistent Disk

Data disk size in GB

150

Elasticsearch overview

Solution provided by Google Click to Deploy

Software

Operating System	Debian (9.9)
Software	Elasticsearch (5.6.8) Google-Fluentd (1.6.17) OpenJDK (1.8.0) Stackdriver-Agent (5.5.2)

Documentation

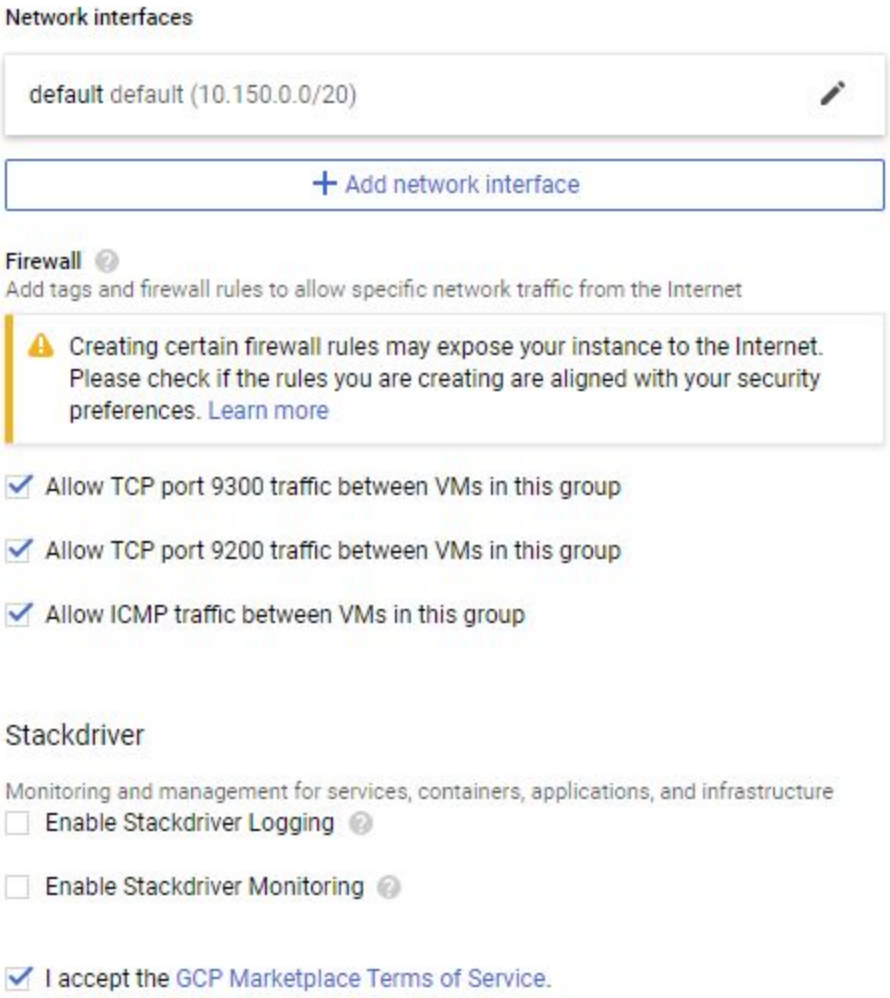
Get started with Elasticsearch

Terms of Service

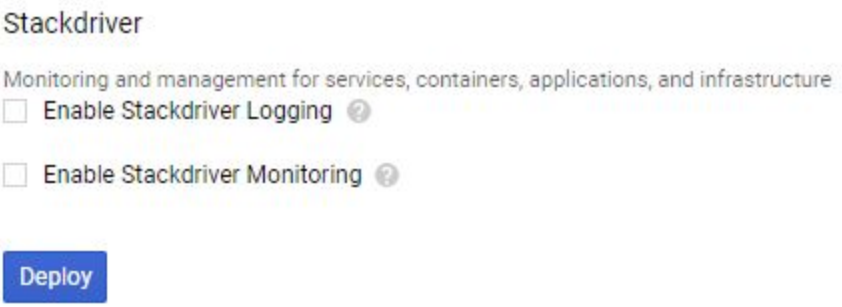
By deploying the software or accessing the service you are agreeing to comply with the GCP Marketplace terms of service and the terms of applicable open source software licenses bundled with the software or service. Please review these terms and licenses carefully for details about any obligations you may have related to the software or service. To the limited extent an open source software license related to the software or service expressly supersedes the GCP Marketplace Terms of Service, that open source software license governs your use of that software or service.

By using this product, you understand that certain account and usage information may be shared with Google Click to Deploy for the purposes of sales attribution, performance analysis, and support.

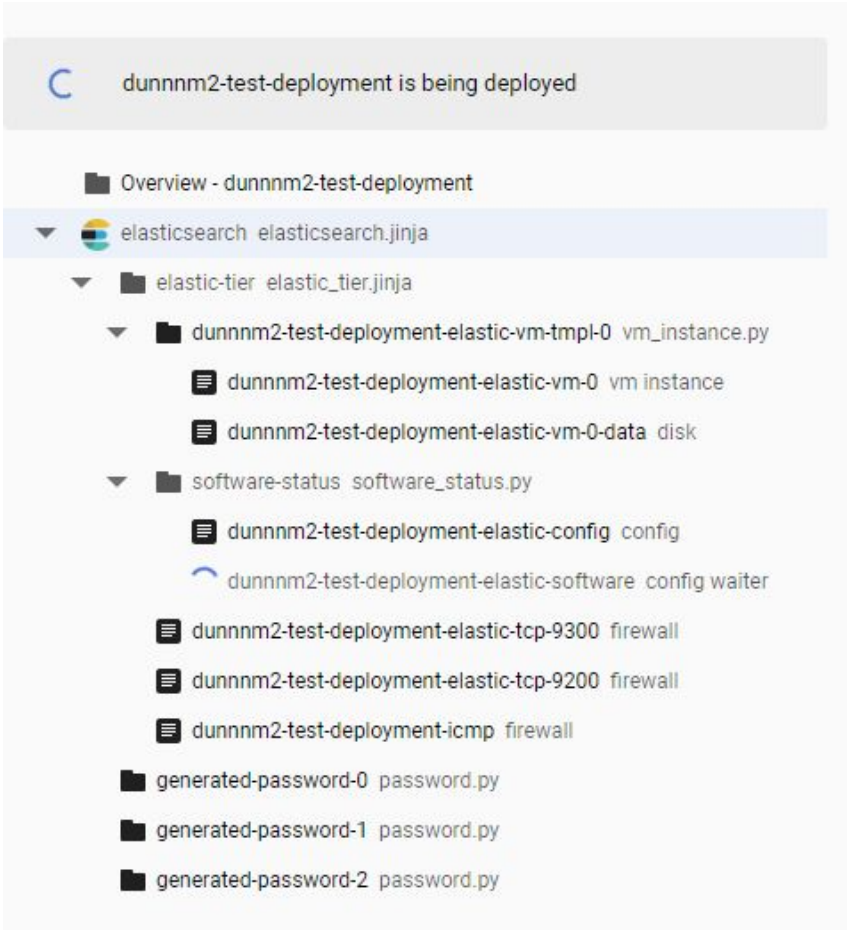
Google is providing this software or service "as-is" and will not perform any ongoing maintenance. Ongoing upgrades and maintenance are your responsibility.



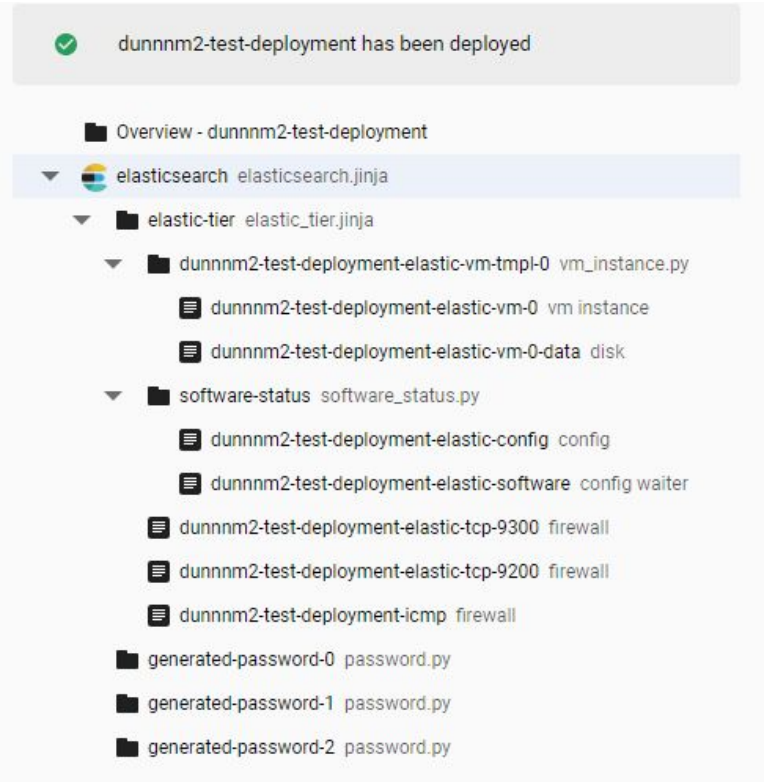
5. Hit Deploy



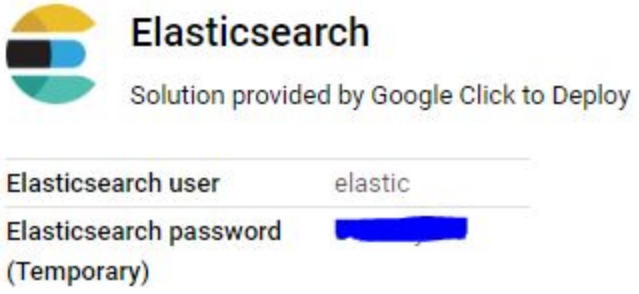
6. Allow Deployment to Process



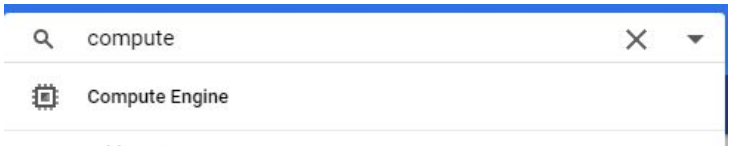
7. Deployment finishes



8. Make note of the user and password listed to the right of the deployment finish. If you forget this later, check the Deployment Manager in GCP.

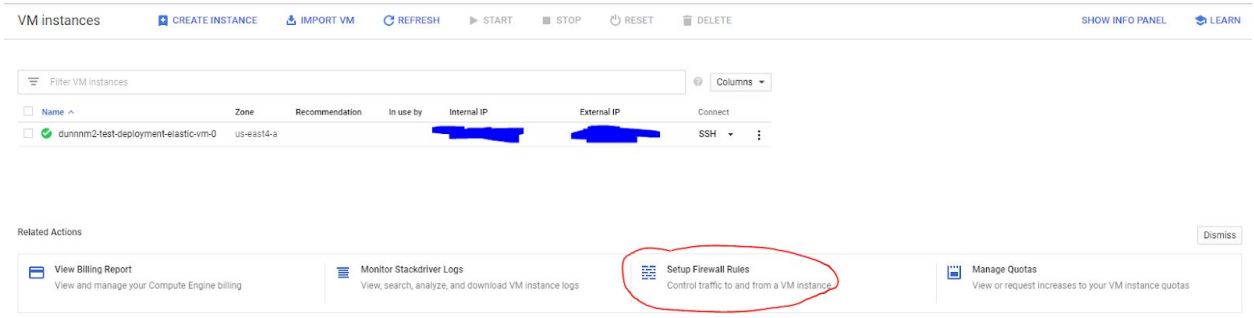


9. Navigate to the Compute Engine section of your GCP account



10. Make note of the ExternalIP here

11. Select Firewall Settings options



12. Select Create Firewall Rule

13. Configure a firewall rule to enable HTTP + HTTPS

Create a firewall rule

previous page

Firewall rules control incoming or outgoing traffic to an instance. By default, incoming traffic from outside your network is blocked. Learn more

Name *
httprule
Lowercase letters, numbers, hyphens allowed

Description

Logs

Turning on firewall logs can generate a large number of logs which can increase costs in Stackdriver. Learn more

- On
- Off

Network *
default

Priority *
1000
Priority can be 0 - 65535 Check priority of other firewall rules

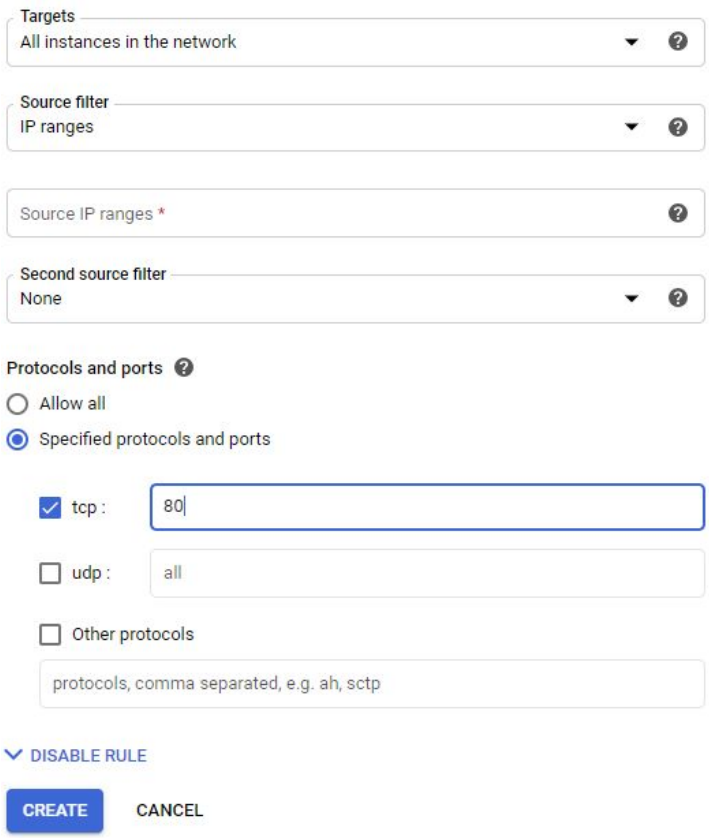
Direction of traffic

- Ingress
- Egress

Action on match

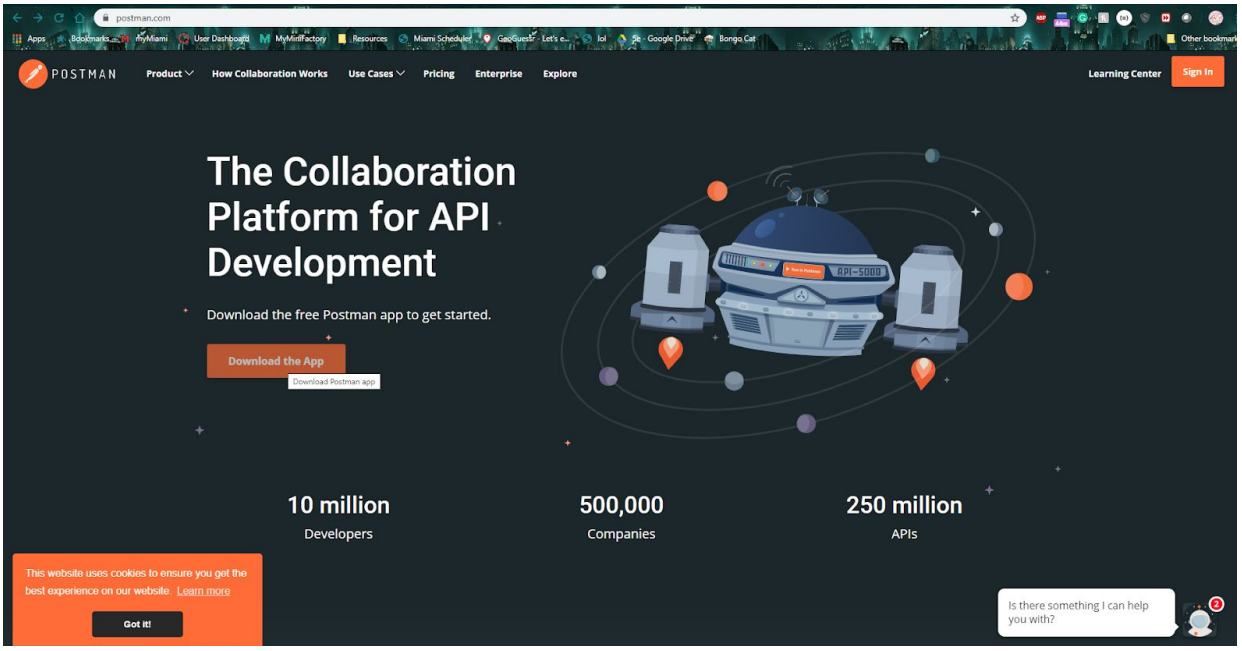
- Allow
- Deny

Targets
All instances in the network

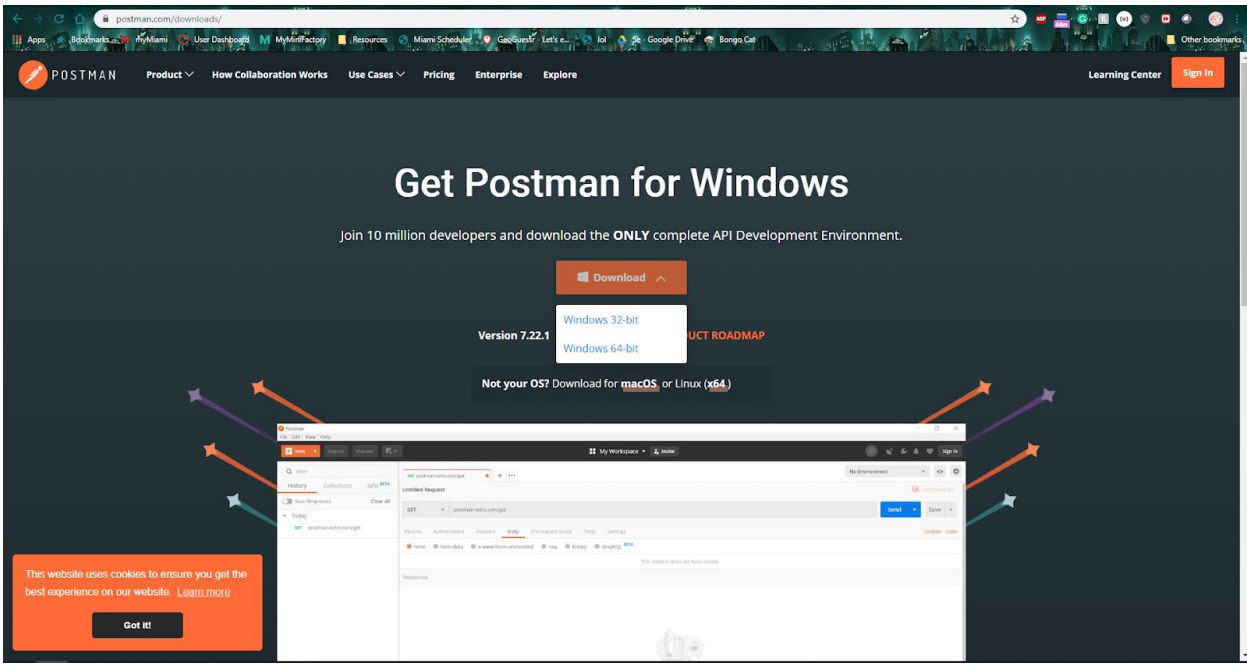


14. Install Postman

a. Go to www.postman.com and click “Download the App”

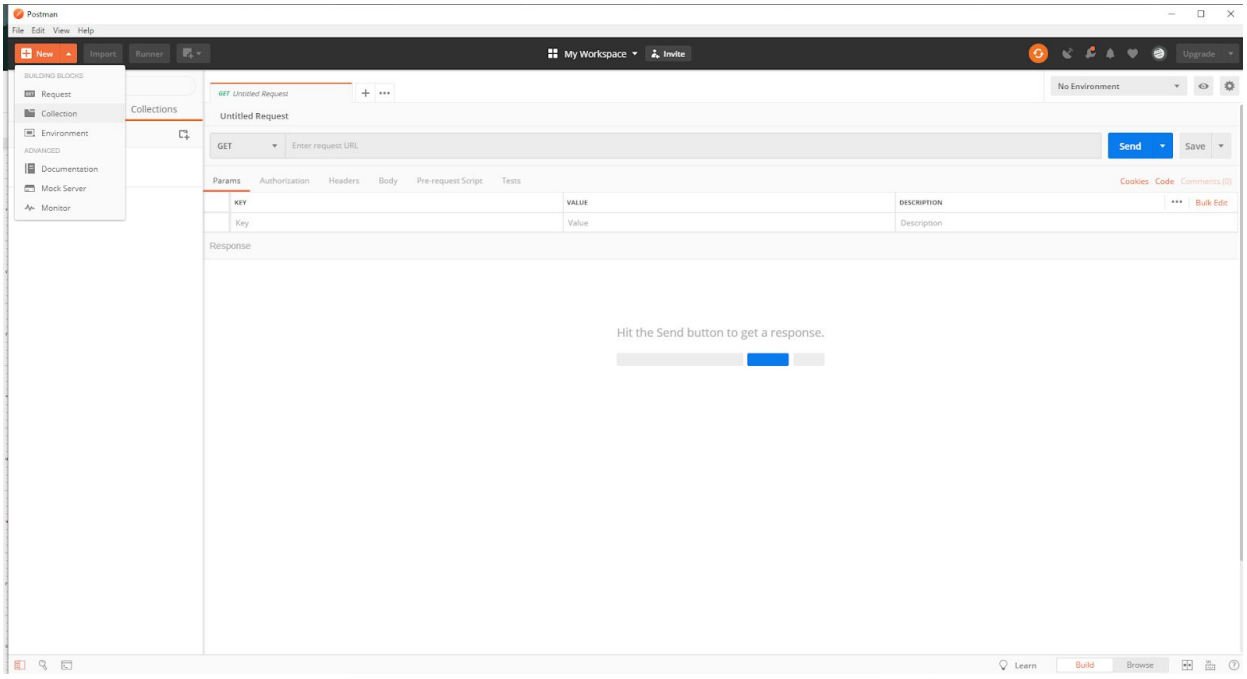


b. Select your OS and download/install Postman

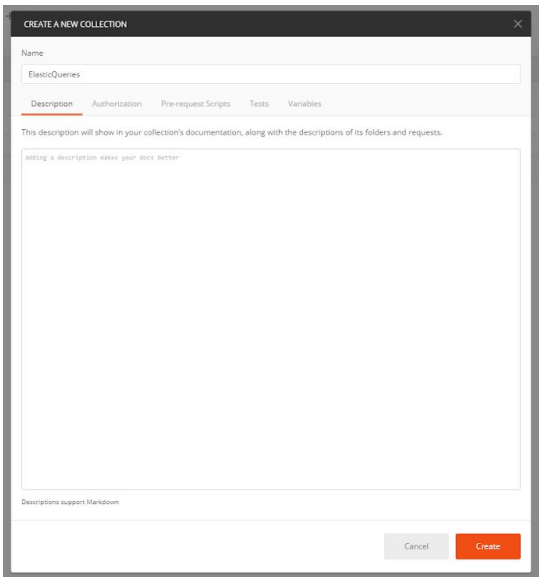


15. Using Postman

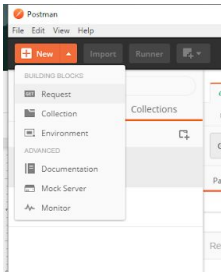
a. Create a collection to store your requests using the “New” dropdown in the upper lefthand corner



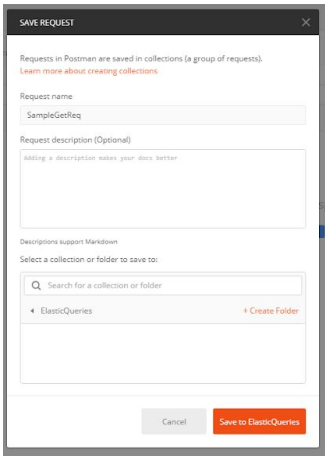
b. Assign a name to the collection and click Create



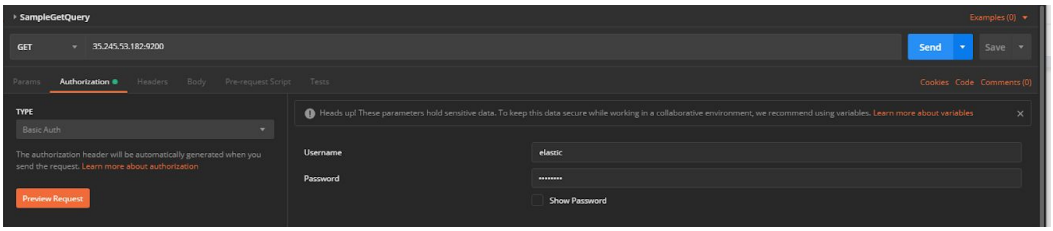
c. Using the New dropdown menu select Request



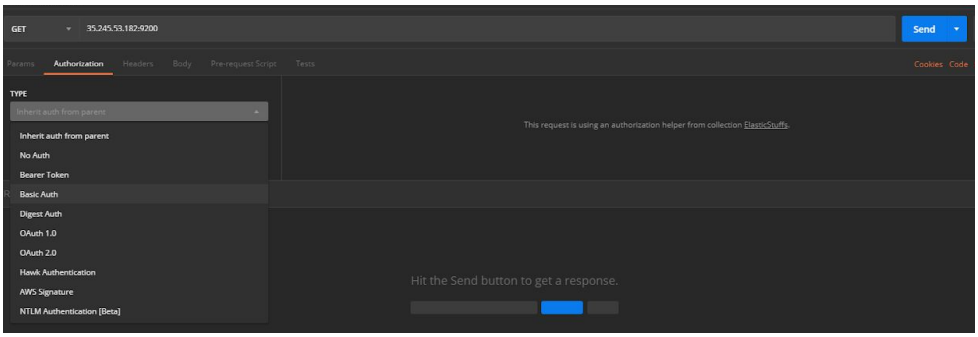
d. Give it a good name and make sure the previously created collection is selected in the lower area.



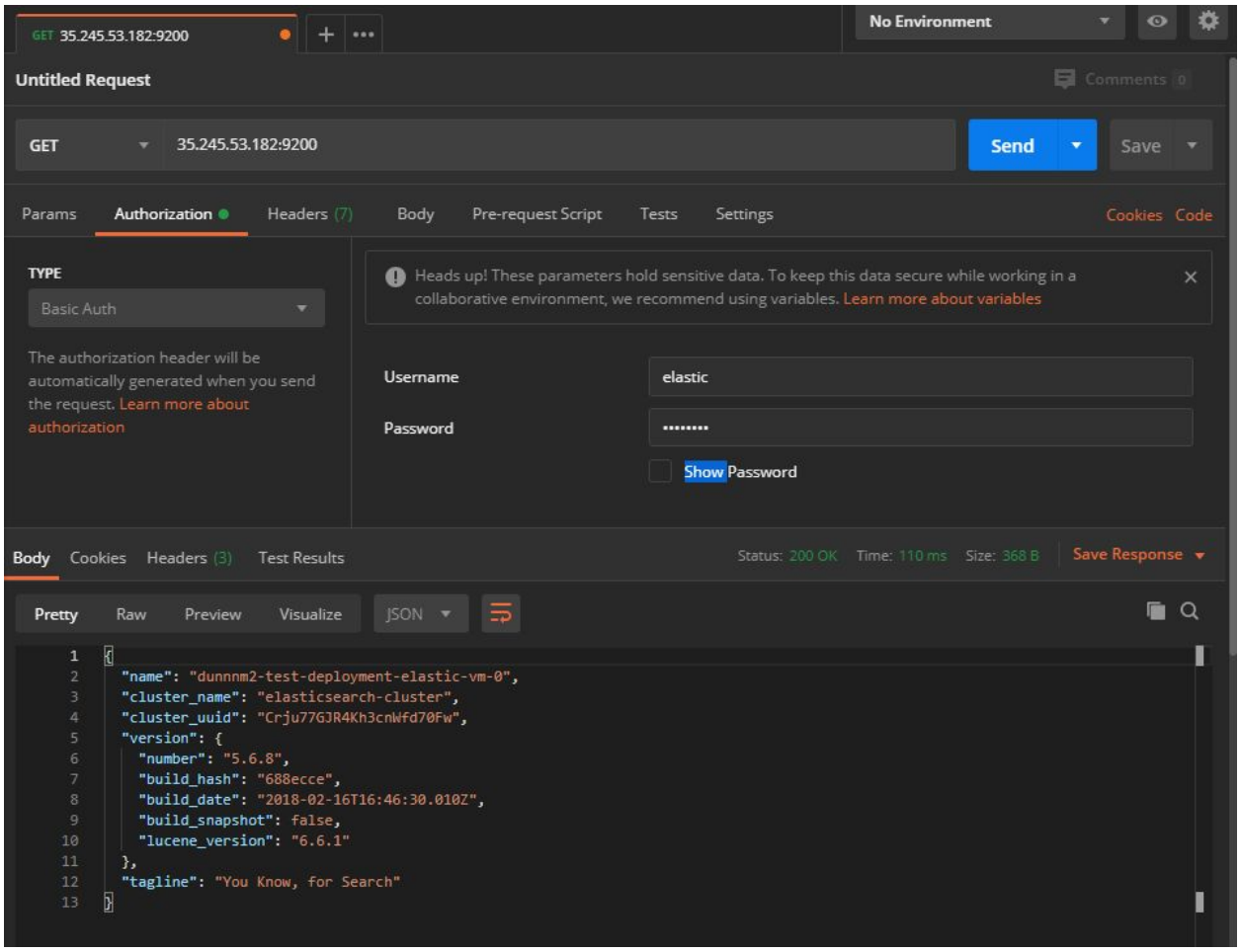
e. Type in the External IP shown in GCP into the request bar followed by :9200 which indicates the port.



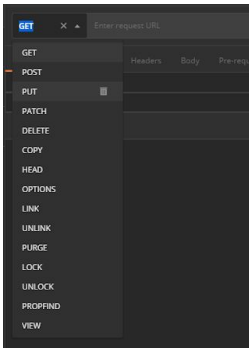
f. Go to Authorization and select Basic Auth, filling in the Username and Password from GCP



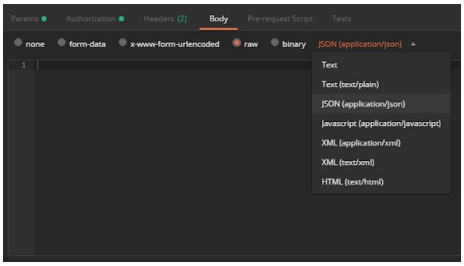
g. Pressing send should give you information regarding your instance



h. Make a new Request as you did before but this time change the request type to PUT rather than GET and ensure that your authorization settings are correct

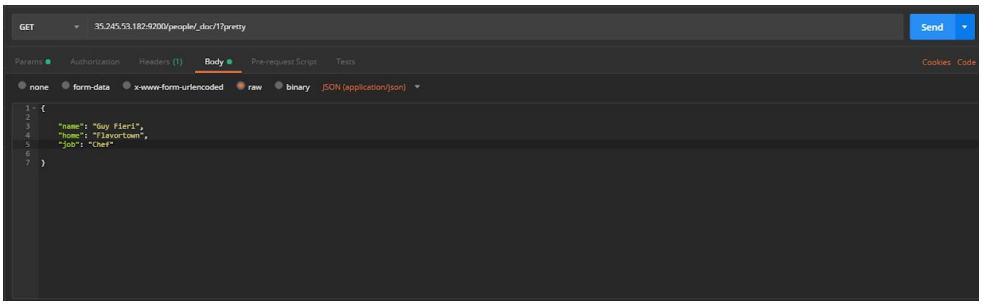


i. Go to the Body tab and change the setting to raw and select JSON from the dropdown list



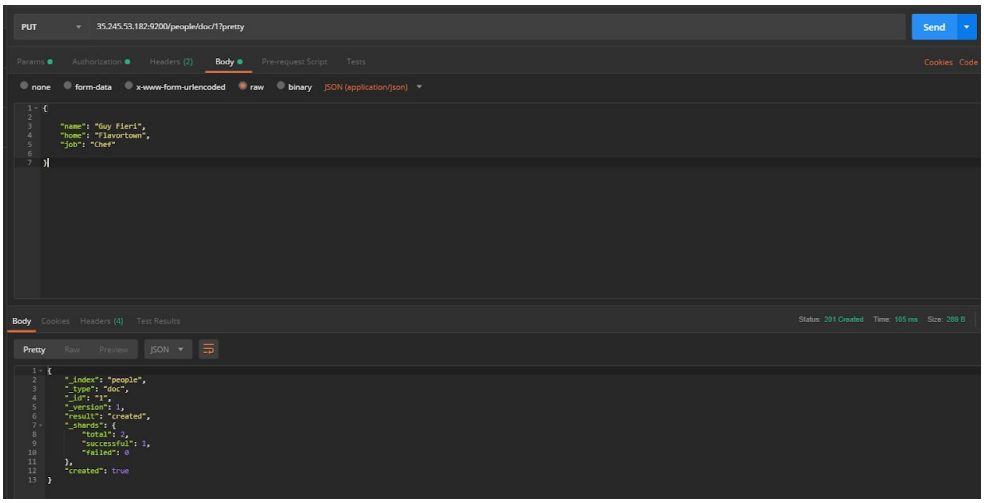
j. Modify the request URL by adding a document name followed by /doc to declare which document to insert into and follow that with /1?pretty

k. Add data to insert into the document.



(This should be a PUT request rather than a GET request)

l. Pressing send will produce the following and add the entry to the document



m. Add a lot more data

35.245.53.182:9200/people/doc/

POST 35.245.53.182:9200/people/doc/

Params Authorization Headers (2) Body Pre-request Script Tests

none form-data x-www-form-urlencoded raw binary JSON (application/json)

```
{
    "name": "Scott Harris",
    "home": "Texas",
    "job": "Professional Idiot"
}
```

35.245.53.182:9200/people/doc/

POST 35.245.53.182:9200/people/doc/

Params Authorization Headers (2) Body Pre-request Script Tests

none form-data x-www-form-urlencoded raw binary JSON (application/json)

```
{
    "name": "Dr. Davis",
    "home": "Ohio",
    "job": "Database Queen"
}
```

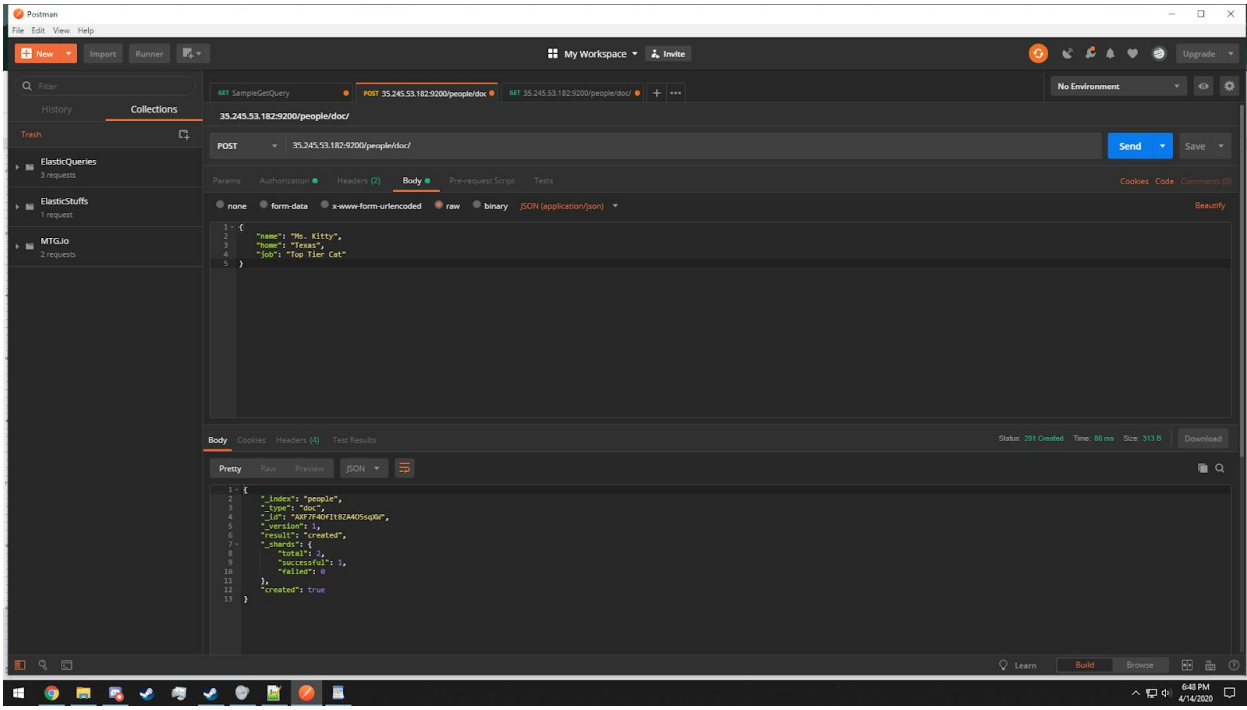
35.245.53.182:9200/people/doc/

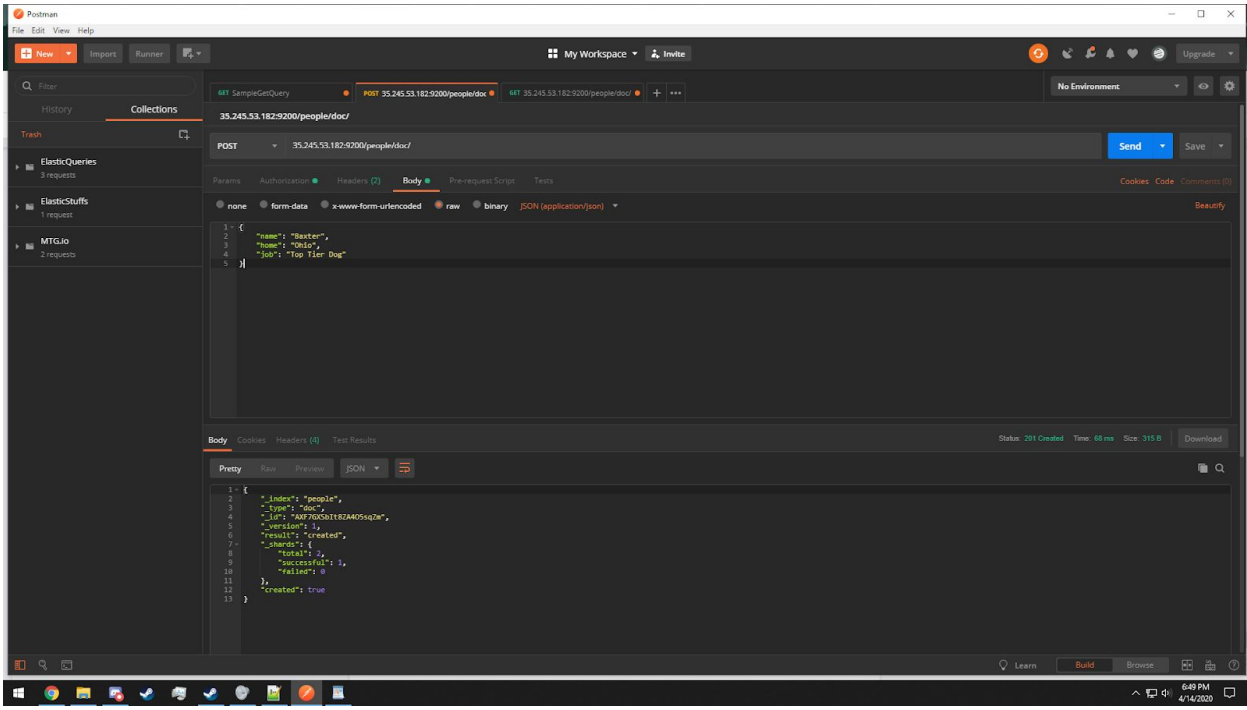
POST 35.245.53.182:9200/people/doc/

Params Authorization Headers (2) Body Pre-request Script Tests

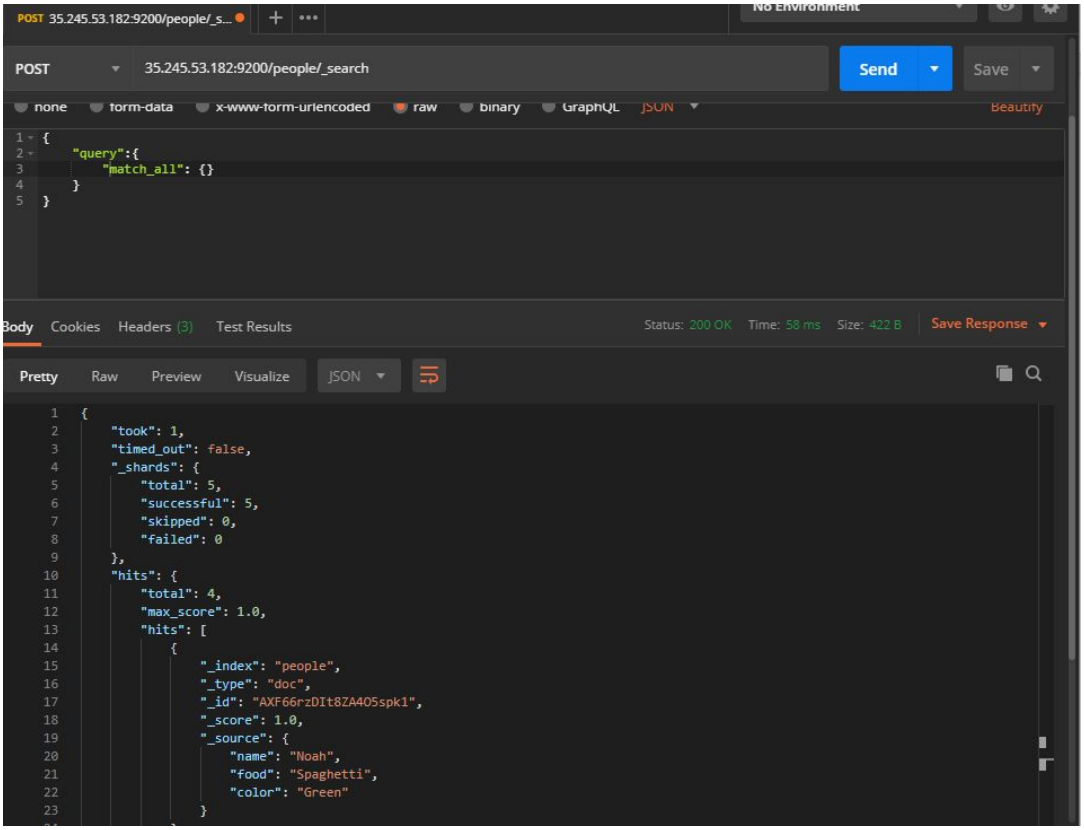
none form-data x-www-form-urlencoded raw binary JSON (application/json)

```
{
    "name": "Noah Dunn",
    "home": "Ohio",
    "job": "Professional Cheese Grater"
}
```



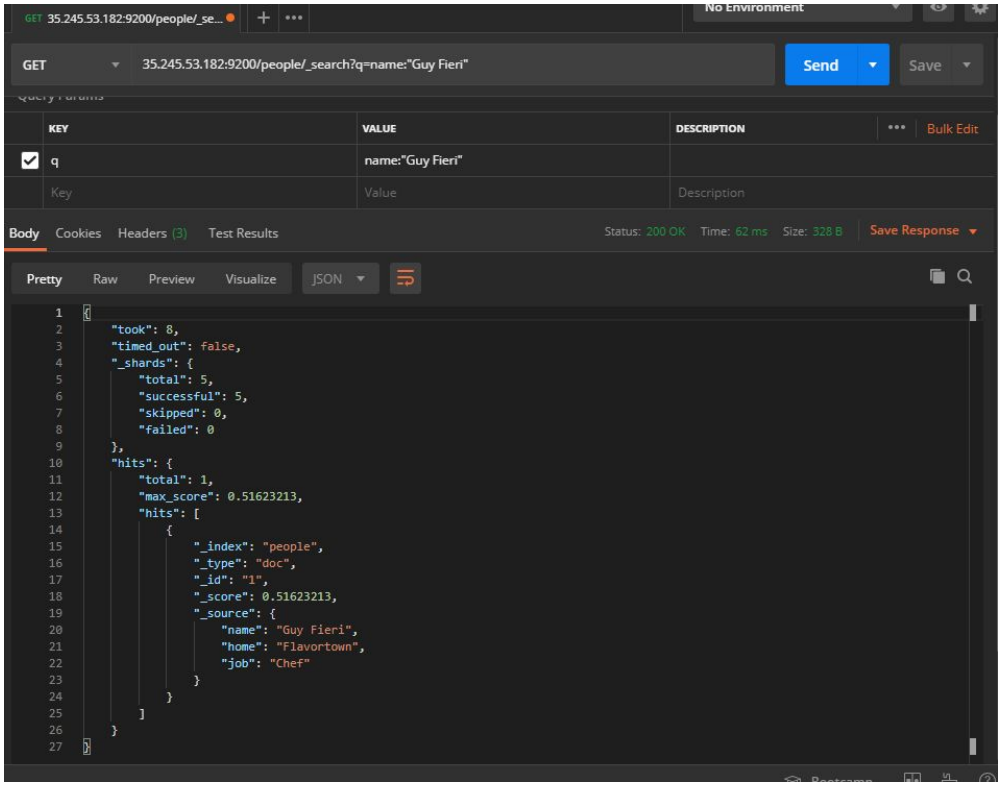


n. Creating a GET request to see what is in the people document should be as follows

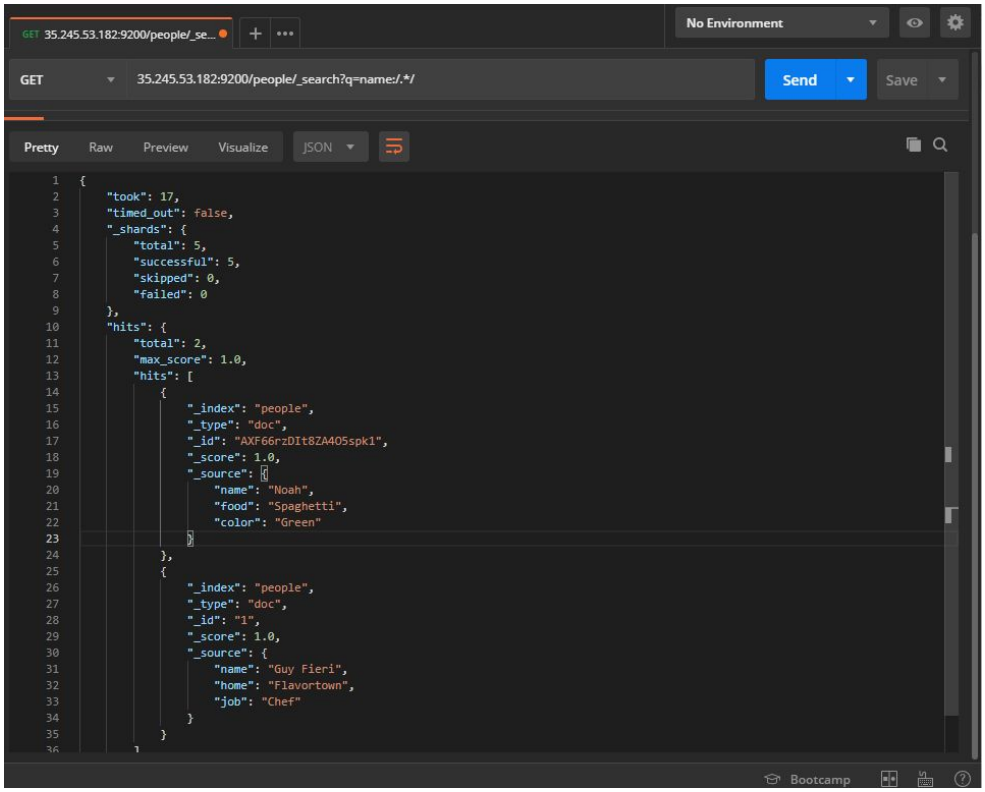


16. There are two methods of querying:
    a. URI construction
    b. DSL Language

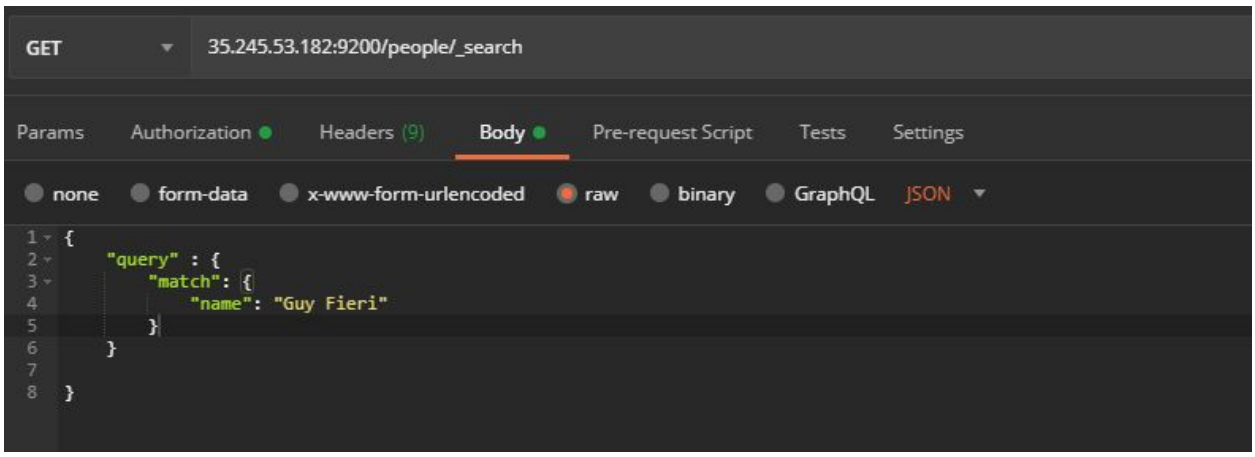
    URI construction usually makes use of GET calls, DSL usually uses POST requests
    The following uses a URI query that searches for the name of ‘Guy Fieri’



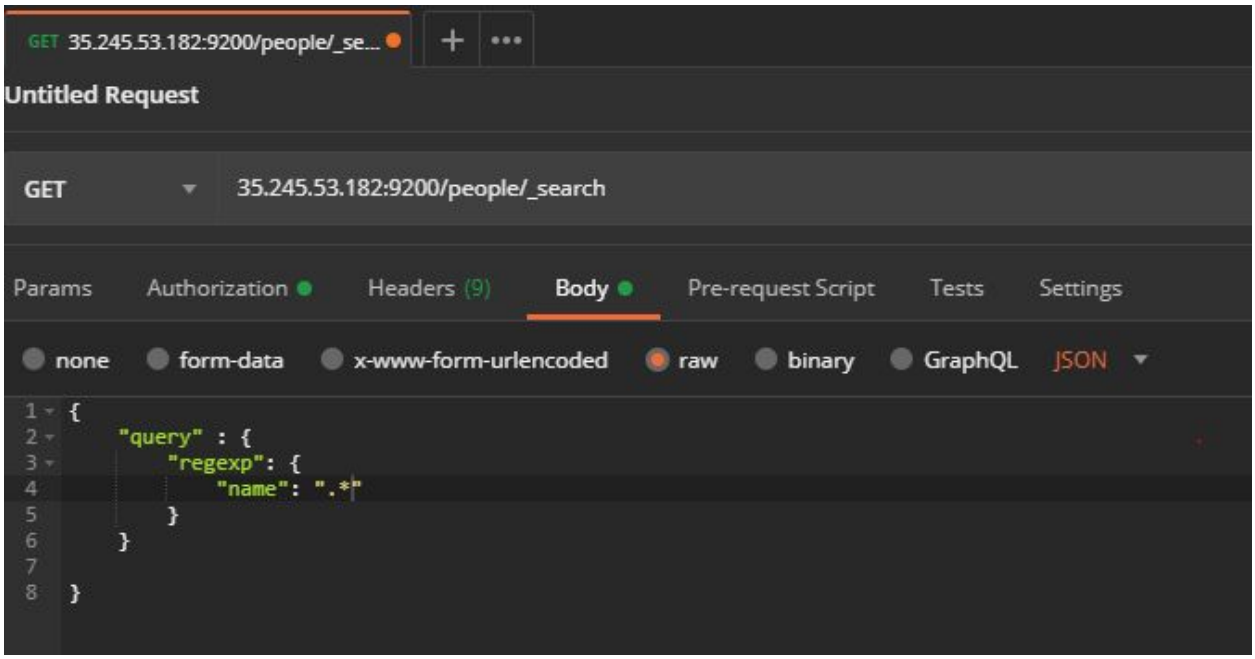
17. You can also use regular expressions to check certain fields against a matching regex.

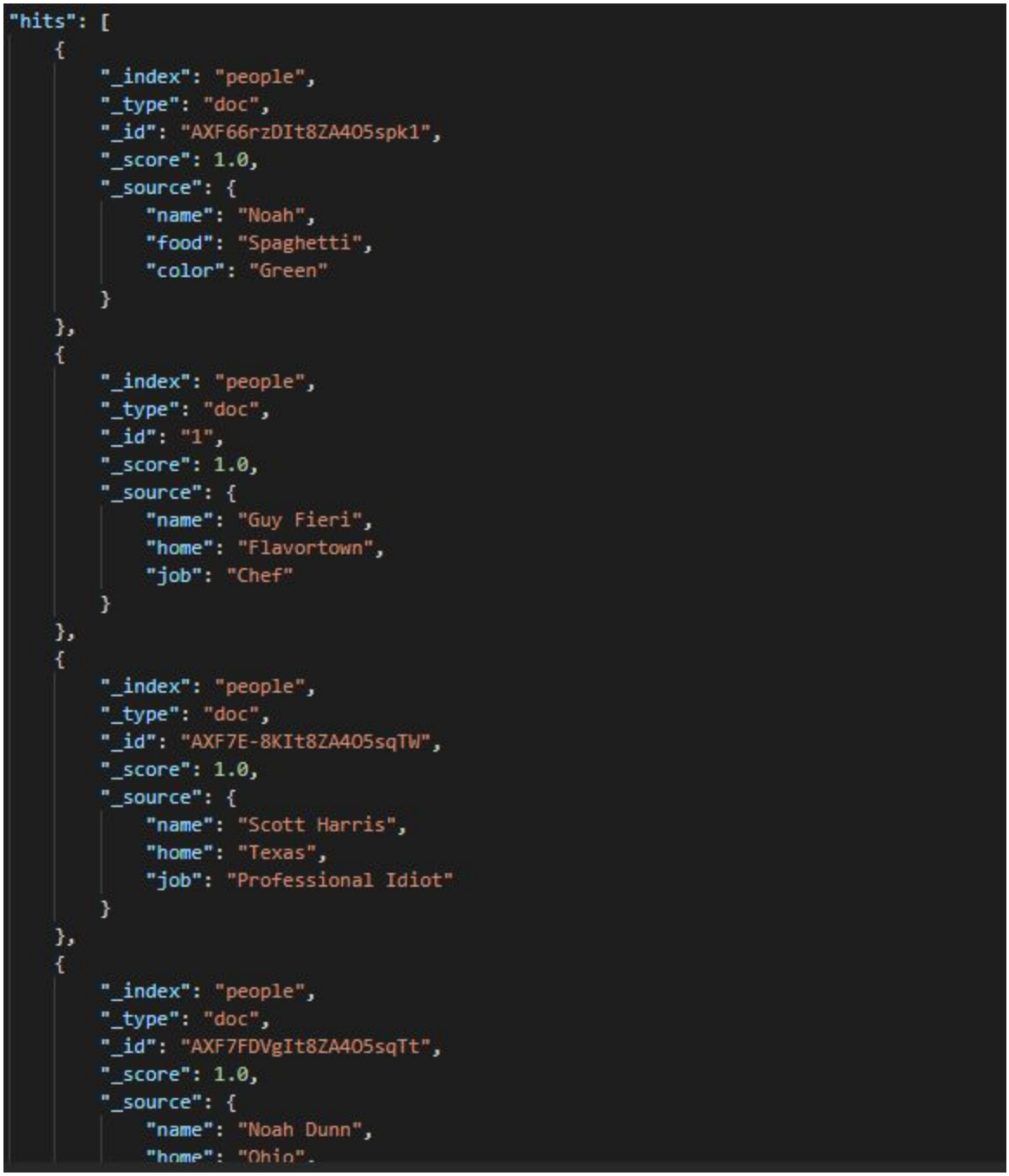


18. Here are the equivalent queries in DSL

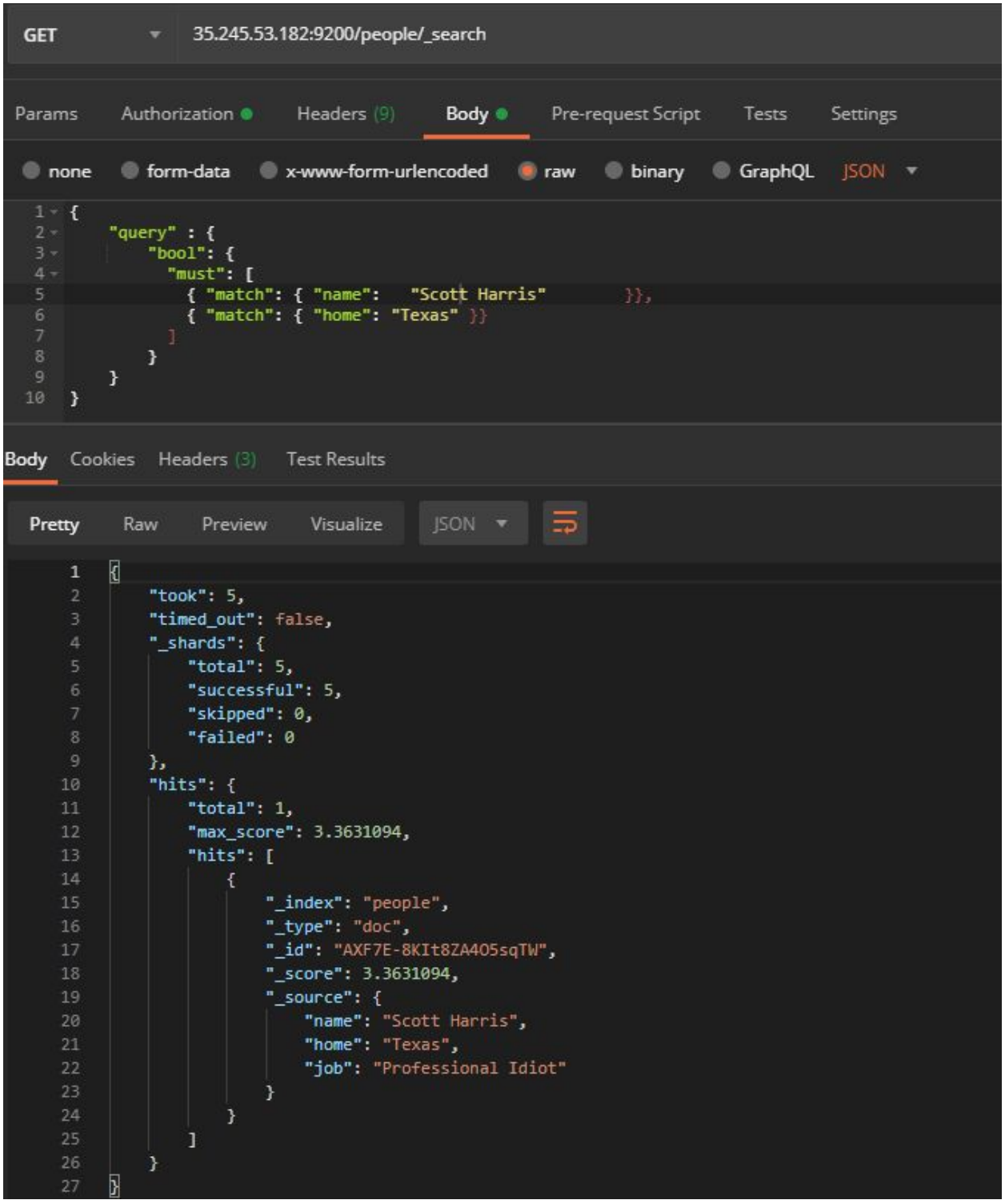


```
{
    "took": 6,
    "timed_out": false,
    "_shards": {
        "total": 5,
        "successful": 5,
        "skipped": 0,
        "failed": 0
    },
    "hits": {
        "total": 1,
        "max_score": 2.160473,
        "hits": [
            {
                "_index": "people",
                "_type": "doc",
                "_id": "1",
                "_score": 2.160473,
                "_source": {
                    "name": "Guy Fieri",
                    "home": "Flavortown",
                    "job": "Chef"
                }
            }
        ]
    }
}
```

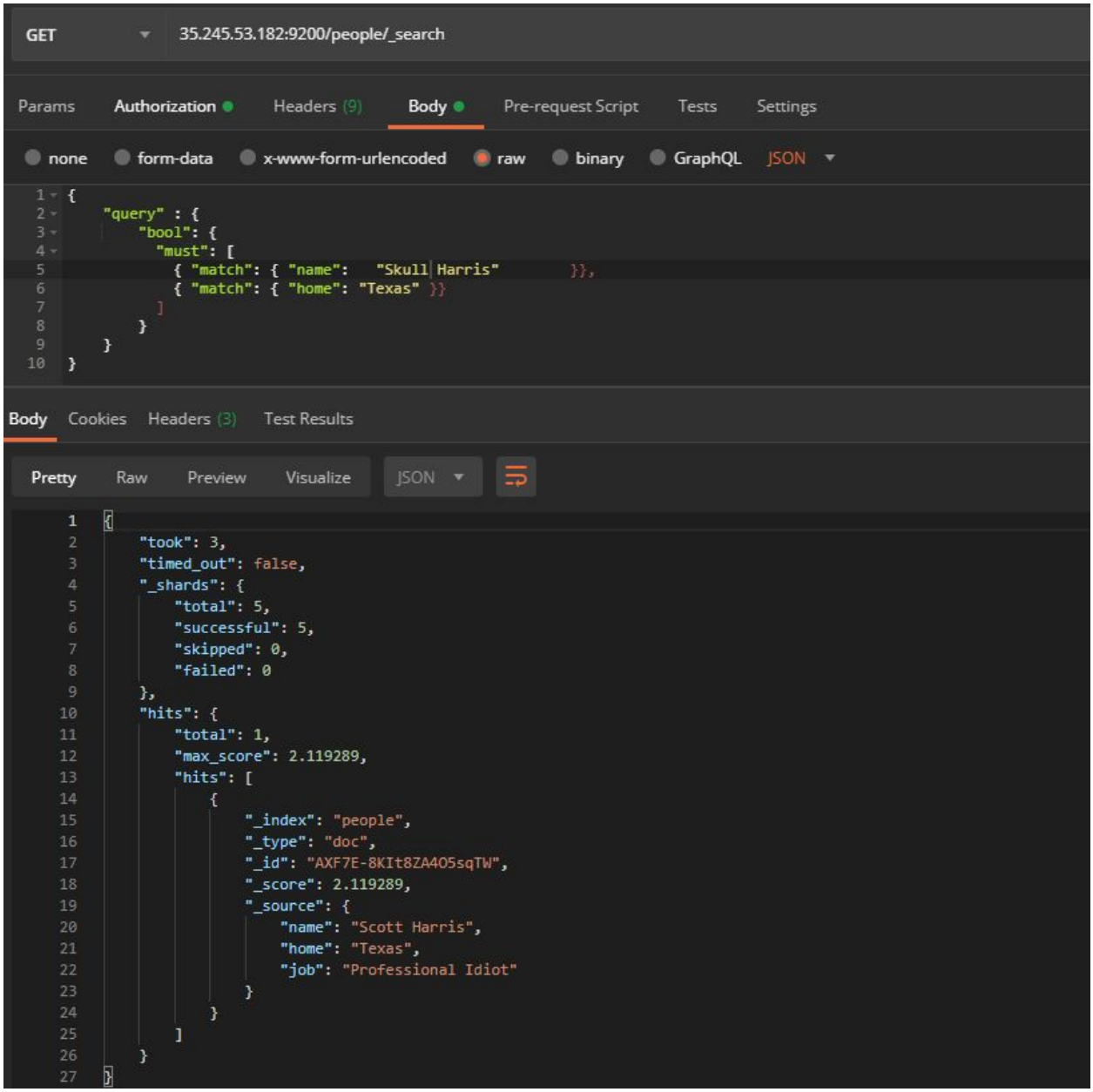





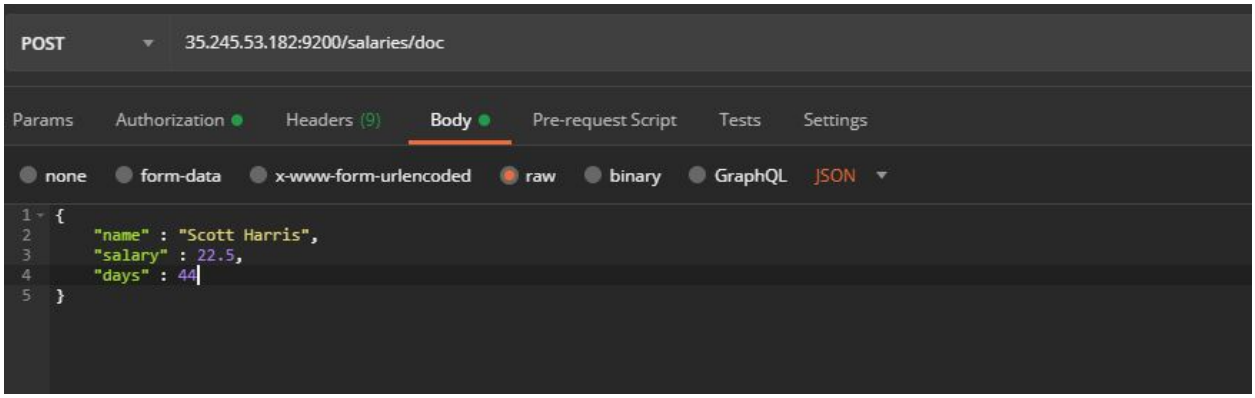
19. One of Elasticsearch's primary query features is its use of relevance scoring for queries that are almost correct. Try running a normal query in the following format.

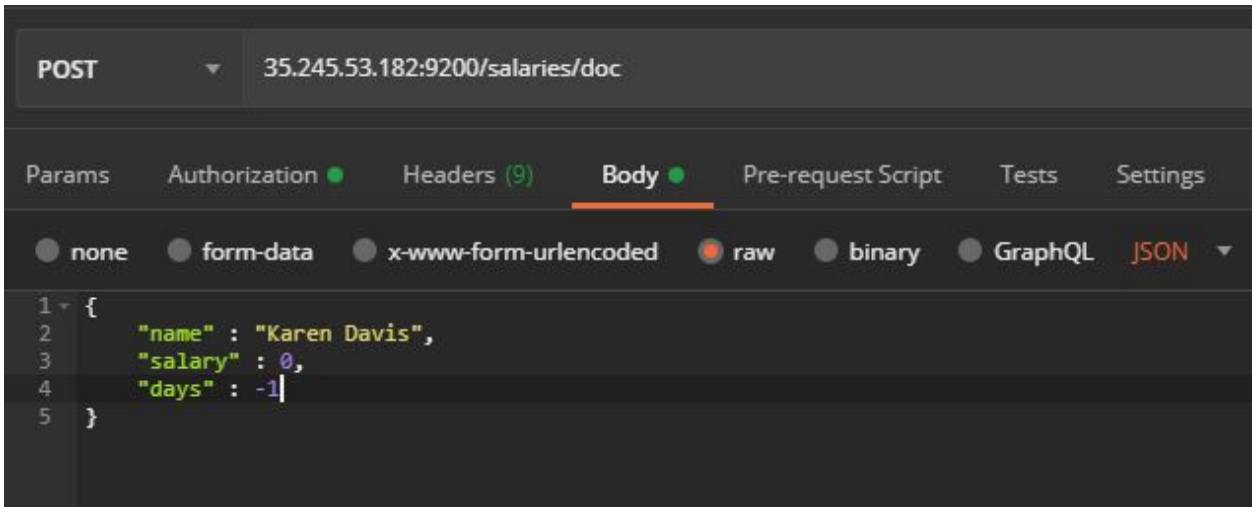


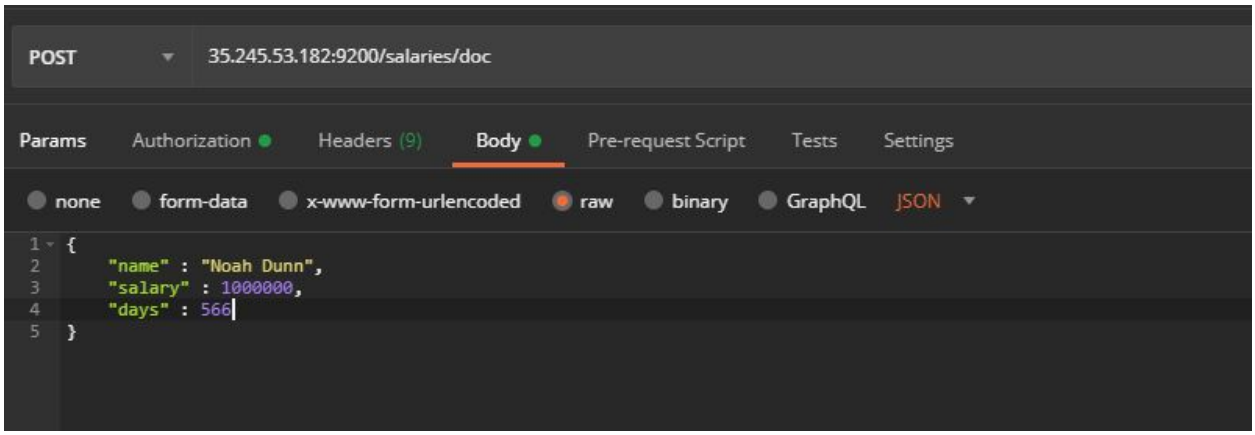
20. Now mess up some of the input on purpose. The relevance score (3.3631) will drop, but we will still be able to retrieve the same entry from the document.



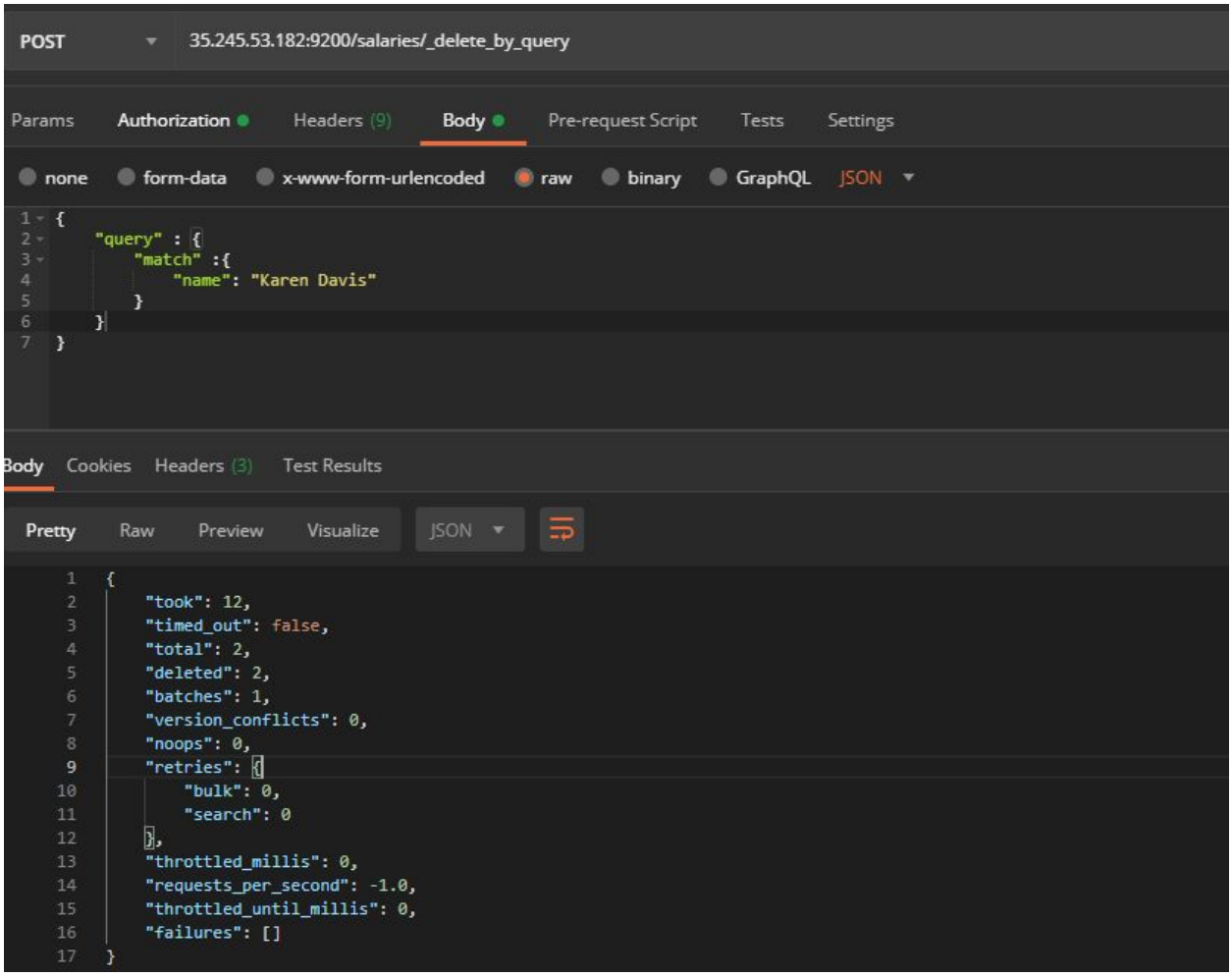
21. Note: In even more complex queries, the “fuzzy” key can be used to specify exactly how much difference between two words we want to allow. https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-fuzzy-query.html
22. Add some new entries to a new document called salaries



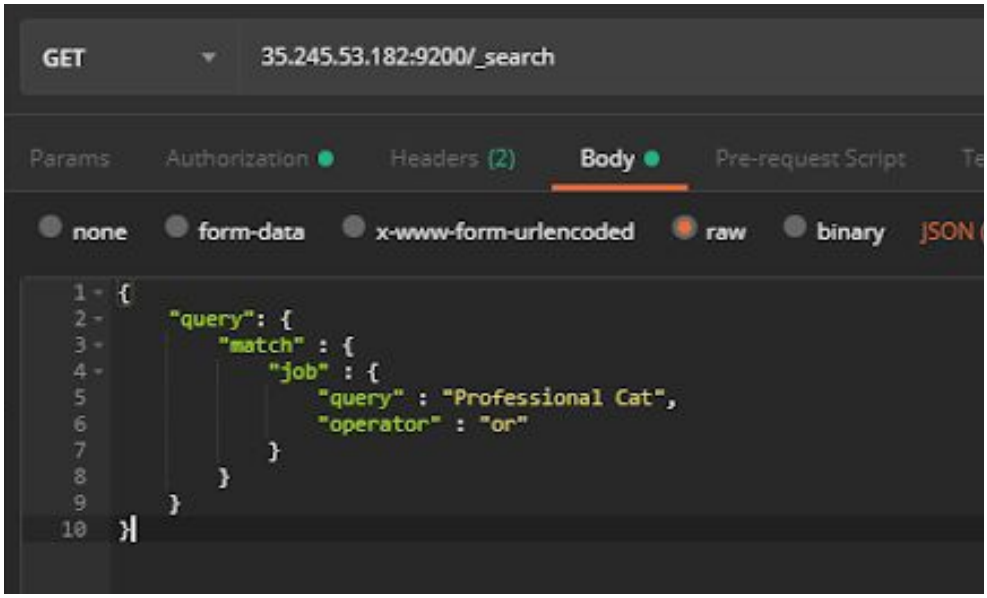




23. If you happen to double add an entry in this or any document, you can use the /_delete_by_query clause to remove all entries



24. Use the OR operator to search for keywords in a specified field across documents



This concludes a comprehensive introduction to the installation, running, and querying of the Elasticsearch database system.

names	date	start time	end time	duration	activity description	tasks for next time
Noah Dunn	4/14/2020	9:30 AM (EST)	9:55 AM(EST)	0.25	Finished reading over into to project description, texted Scott to see if he was okay with Elasticsearch	Talk to Dr. Davis about System choice
Noah Dunn	4/14/2020	10:00 AM(EST)	11:00 AM(EST	1.00	Talked with Dr. Davis, local installed ES, requested more GCP credit, ran sample queries	Get Elasticsearch loaded up on GCP
Scott Harris	4/14/2020	9:17 (CST)	9:52 AM (CST)	0.50	Worked on installing Elasticsearch	Get Elasticsearch loaded up on GCP
Noah Dunn, Scott Harris	4/14/2020	~4:40 PM (EST)	~8:00 PM (EST)	3.25	Got Elasticsearch loaded up on GCP, figured out and resolved all network issues, installed POSTMAN, wrote a significant portion of the installation guide	Finish the document and clean it up
Noah Dunn, Scott Harris	4/14/2020	9:40 PM (EST)	11:30 PM (EST)	1.75	Finished up the doucment, added finalized queries, cleaned and polished the remainder	Begin work on final presentation
Noah Dunn, Scott Harris	4/21/2020	12:00 PM (CST)	2:00 PM(CST)	2.00	Started & Finished working on presentation slides	Will Work on Data Demand and Solution
Noah Dunn	4/23/2020	4:20PM (EST)	5:00:00 PM (EST)	0.50	Started and Finished Writing Python script to organize input data file for ProtoBulk Insert	Working on Creating Data Demands
Scott Harris	4/23/2020	4:20PM (EST)	5:00PM (EST)	0.50	Began looking into types of queries to build	Working on Creating Data Demands
Noah Dunn and Scott Harris	4/23	5:00PM (EST)	6:30PM (EST)	1.50	Completed writing data demands 1-3 and providing documentation for teacher and student	Will Work on producing presentation video for submission
Noah Dunn & Scott Harris	4/28/2020	1:00 PM (CST)	2:16PM(CST)	1.50	Video clips recorded, edited, spliced together, and uploaded to Drive and Youtube	Will Work on the Peer Reviews once designated
Noah Dunn	5/1/2020	9:50 AM (EST)	10:20 AM (EST)	0.50	Finished first peer review, emailed Dr. Davis to get some help on the assignment	Finish Peer Review 2 and 3
Noah Dunn	5/1/2020	10:40 AM (EST)	11:10 AM (EST)	0.50	Finished second and third peer review	Finish Final Submisison for Project
Scott Harris	5/3/2020	12:00 PM (CST)	12:45 PM CST	0.75	Started & finished peer reviews	Work with Noah to finish up our final submission
Noah Dunn, Scott Harris	5/6/2020	12:30 PM (EST)	1:00 PM (EST)	0.50	Combined all files into the project report	This is the final. Nothing left to be done

	Noah Dunn	Scott Harris
Part 1: Selection of System	1.25	0.50
Part 2: Installation/Usage	5.00	5.00
Part 3: Slides	2.00	2.00
Part 4: Data Demands	1.50	2.00
Part 5: Video	1.50	1.50
Part 6: Peer Review	1.00	0.75
Part 7: Final Report	0.50	0.50
total:	12.75	12.25