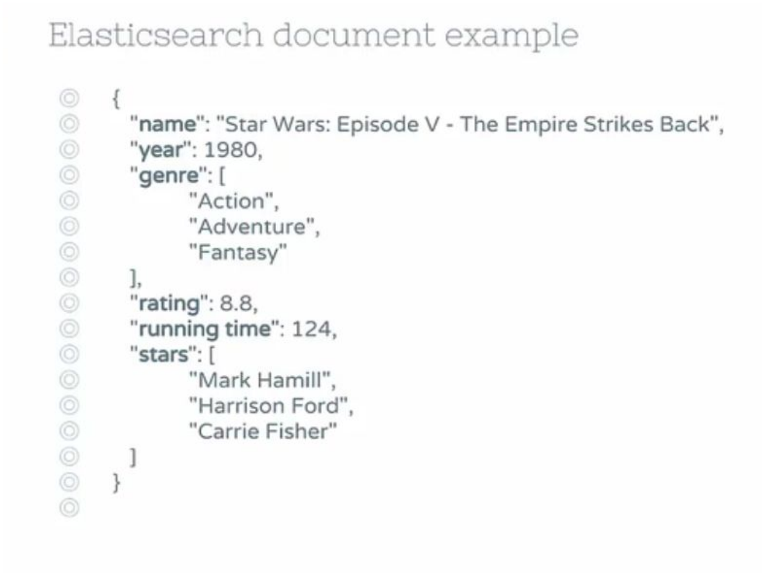
Elastic Search cheat sheet

What is Elastic Search?

- It's a real time distributed search and analytics system built on top of Apache Lucene. Also it's it is a distributed document store with real time analytics.
- Companies that use elasticsearch include the following: Mozilla, Quora, Foursquare, Github, Netflix etc.
- Elasticsearch is document printed which means it's not your typical relational database store. (It's similar to mongoDB)
- Elasticsearch also indexes the documents so that they're searchable
- You can communicate with elastic search via REST API using JSON.
- Elasticsearch allows for multi tenancy. Multitenancy is when a single piece of software can be accessed by multiple tenants(group of users).
- Example of a document:



Elasticsearch document example

```
{
  "name": "Star Wars: Episode V - The Empire Strikes Back",
  "year": 1980,
  "genre": [
        "Action",
        "Adventure",
        "Fantasy"
  ],
  "rating": 8.8,
  "running time": 124,
  "stars": [
        "Mark Hamill",
        "Harrison Ford",
        "Carrie Fisher"
  ]
}
```

-

Terminology

- In RDBMS
  - Database
    - Table
      - Tuple
- In elasticsearch
  - Index
    - Type
      - Document
- “Indexing a document” in elasticsearch is used as “inserting/updating a document”
- Verb and noun versions of index have different meanings.

Node, Cluster

- Node is a running instance of elasticsearch.
- One node per server
- When you start a node, it tries to join a cluster.
- Cluster is a group of nodes.
- Each cluster has a single master node which is chosen automatically

Shard, Replica

- A shard is a subset of data in db. Breaks up the database horizontally and putting each horizontal piece/shard in a different server. This helps to balance load. Default for elastic search is 5.
- Primary shard is the first place where your document is stored when you index it.
- Replica of your primary shard will get their copy as well.
- Replica shard is just a copy of the primary shard.
- Why would we want several copies of data?
- For a backup plan just in case the primary shard goes down.
- Also replica shards can enhance the performance of elastic search.
- Don't put the shard and the replica shard on the same nose or else if the node fails you will lose all your data.

How to run and install elasticsearch (On Mac OSX)

- We will use home brew, so first update home brew by typing “brew update”
- Then run “brew install elasticsearch”
- Copy the last line home brew have you and enter that command in terminal and now the node should be up and running.

How to install head and marvel plugins (Mac OSX)

- **If you are using elastic search version 2 or later then don't install the marvel plugins.**
- head is a web prompt and used for browsing and interacting with an elastic cluster.
- Marvel is used for monitoring your cluster. Marvel contains a sas plugin that we will use throughout our course.
- Go to the bin folder in the elasticsearch folder in terminal.
- Elasticsearch’s files is located in `cd /usr/local/Cellar/elasticsearch/` (http://stackoverflow.com/questions/11954677/how-do-i-find-where-elasticsearch-is-installing-my-plugins)

To download plugins you need to run the plugin program with the name of the plugin as an argument:

- In version 1.6 the plugin program is in the folder `/usr/local/Cellar/elasticsearch/`bin
- In our version, 2.3.2, it is inc
  - To install head run: ./plugin install mobz/elasticsearch-head

Installing Kibana and Sense

For version 2, you must download kibana to work with sense. Kibana is a data visualization platform which allows you to create graphs with your data.

To download to kibana:

https://www.elastic.co/downloads/kibana

Click mac , and you will have a zip file. Unzip this file

To download sense plugin of kibana:

Go to kibana folder, and cd into bin

Run in terminal: ./kibana plugin --install elastic/sense

Next, you must link kibana plugin to your elasticsearch:

Cd to kibana folder, and then into the config folder

Open *kibana.yml* in the editor

Scroll down to elastic search url and uncomment it out

To tests if this works:

Launch elasticsearch at first.

cd `/usr/local/Cellar/elasticsearch/`2.3.2/libexec/bin

Run ./elasticsearch

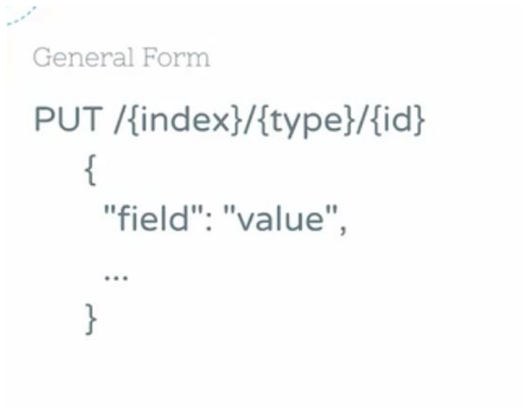
Launch kibana.

Cd into the kibana folder and then its bin.

Then run ./kibana

Next, go to your local browser, and go to localhost:5601/app/sense

Creating Documents in ElasticSearch (PUT)


General Form

```
PUT /{index}/{type}/{id}
    {
     "field": "value",
     ...
    }
```

Index is like a database, Type is like a table, ID is the unique ID. After that we put our document as a JSON

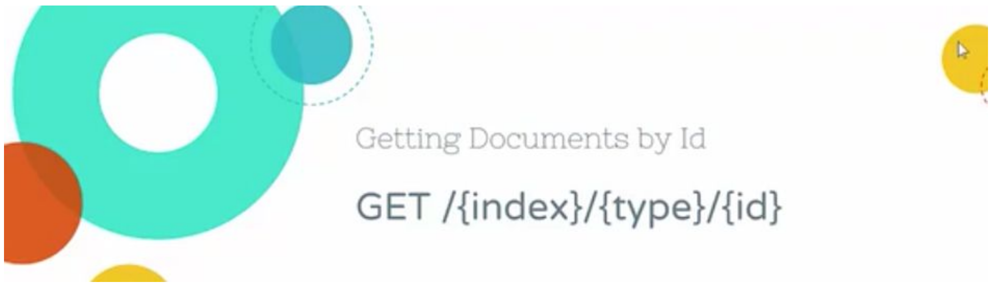
1. Open sense plugin by running elasticsearch, and sense.
2. Put is the command to add things in it

   ```
   PUT /sad music/ drake /1

   {

     "title": "marvin room",

     "artist" :"Drake",

     "album" : "Views",

     "year" : 2016,

   }
   ```

3. On the right screen you will get a response. I will comment out an explanation next to each response with a //

```
{
  "_index": "my_deezer",   //
  "_type": "song",    //
  "_id": "1", //first thing in the database
  "_version": 1, //first version of this file. Deleting it, updating adds 1 to the version
  "_shards": { // idk
    "total": 2,// idk
    "successful": 1, //idk
    "failed": 0 //idk
  },
  "created": true //idk
}
```

How to read documents from Elasticsearch (GET)



Getting Documents by Id

GET /{index}/{type}/{id}

To retrieve what we got before

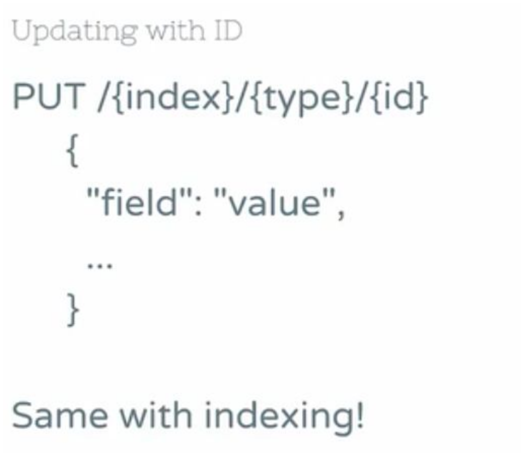
    GET my_deezer/song/1

To get the album/artist of this album

GET my_deezer/song/1?_source=album,artist

Updating with ID

https://www.youtube.com/watch?v=_Nua3Cjdik0&list=RDTmxIRKxZm6o&index=4



Updating with ID

```
PUT /{index}/{type}/{id}
    {
      "field": "value",
      ...
    }
```

Same with indexing!

Were going to update our first song and change the song to controlla

To update, you must include all the old fields, even if you are not updating it. Otherwise they will be deleted

```
PUT my_deezer/song/1
{
"title" : "controlla",
        "artist" :"Drake",
         "album" : "Views",
         "year" : 2016
}
```

To see if it came out good run GET my_deezer/song/1

Deleting

The general form of this is delete index/type/id

Found field, from the response, lets you know if the file is found.

Using Django with ElasticSearch:

https://pixabay.com/en/blog/posts/django-search-with-elasticsearch-47/

For search functionality, django will need only three operations.

1. **Create elasticsearch index, 2. Feed index with data, 3. Retrieve search results**

-Before we start, we must understand Django's request API and how it allows us to communicate with the elasticsearch server. To use this API, we first download it, by running `pip install requests`

General Syntax: requests.**request**(*method*, *url*, ***kwargs*)

arguments

-request is a database command (Get, Post, Delete, Head, Put, Patch)

-Method Url is where in the elastic search engine you want to go. For examp

to search everything in elastic search, we put http://127.0.0.1:9200/subject/_search

-kwargs is the json we pass to the elastic search

Return Value

-requests returns a response object. The response is in response.text

Example of how to translate a sense command to a django command(color coded)

Sense:

```
        PUT /my_deezer/song/1
        {
          "title": "9",
          "artist" :"Drake",
          "album" : "Views"
        }
```

Django:

```
data =
                {
                          "settings": { #these are the settings for an index
                            "number_of_shards": 4, #default number is 5
                            "number_of_replicas": 1 #default is 1
                 },
                "mappings": {
                            "song": { #our type
                              "properties": { #the properties our ids will take
                                "title": { "type": "string", "boost": 2 },#second most important
                                "album": { "type": "string", "boost": 1 } #most important
                              }
                            }
                          }
                        }
response = requests.put('http://127.0.0.1:9200/my_deezer/',data=json.dumps(data))
```

Now lets do 3 Basic Commands as examples.

1. Create Index

You must create an index . Reminder, ElasticSearch is organized as : index/type/id .

The first thing we must do is create the index, or subject and the arguments for it in a JSON format.

```
data = {
    "settings": { #these are the settings for an index
        "number_of_shards": 4, #default number is 5
        "number_of_replicas": 1 #default is 1
    },
    "mappings": {
        "student ID": { #our type
            "properties": { #the properties our ids will take
                "student Name": { "type": "string", "boost": 2 },#second most
important
                "nationality": { "type": "string", "boost": 3 }, #most important
                "id update": { "type": "integer", “boost”: 1 }  #least important
            }
        }
    }
}


import json, requests
requests.put('http://127.0.0.1:9200//',data=json.dumps(data))
print response.text
```

More references for this step :

- https://www.elastic.co/guide/en/elasticsearch/reference/current/indices-create-index.html
- https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping.html

Mapping is basically declaring a document's fields. You can have multiple mappings for one type.

Data Types Are: objects, string, date long, double, boolean, ip, geo_point, geo_shape, completion

Boost gives the importance of a field when it comes to search (https://www.elastic.co/guide/en/elasticsearch/guide/master/query-time-boosting.html). A field with Boost 2 is twice as important as a boost wih 1

You can add new fields by using the PUT API, but you cannot edit old mappings. If you need to reindex the data into a new mapping and delete the old document.

2. Feed index with data

Now we need to take every fileupload object and add them to the elasticsearch database. To do this, we must first understand queries in django:

Take the following documentation:
https://docs.djangoproject.com/en/1.9/topics/db/queries/

For each model you create in django, we have an api that allows use to CRUD with db.

To create a database object, all we do is .save() on a model. This does a behind the scene insert SQL statement. We do this already so we gucci.

To Retrieve: you need to create a queryset(collection of objects from the database). To do this run MODEL.objects.filter(parameters =) or MODEL.objects.all() . The second statement returns all objects the database . If you want to get a specific object you can use its pk, or primary key. MODEL.objects.get(pk=).

```
import json, requests
from webapp.uploads.models import Roster
```

```
data = ''"
for p in Roster.objects.all():
    data += '{"index": {"_id": "%s"}}\n' % p.pk #unique id of note objects
    data += json.dumps({ #JSON encoder method
        "first name": p.first_name,
        "last name": p.last_name,
        "updated": p.updated
    })+'\n'
response = requests.put('http://127.0.0.1:9200/first_name/last_name/_bulk',
data=data) #_bulk is an api that allows multiple tasks to happen at once to
search engine
print response.text
```

https://www.elastic.co/guide/en/elasticsearch/reference/current/docs.html

3. Retrieving Search Results from the Index

```
data = {
"query": {
        "query_string": { "query": "hello world" }
    }
}
response = requests.post('http://127.0.0.1:9200/subject/_search',
data=json.dumps(data))
print response.json()
```

https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html

Three types of queries. Term , Range, and Match.

Term is for exact

Range is for range

Math

In conclusion:

Elasticsearch is a different database that is faster for search. You must send arguments to it in JSON format using requests API.The argument is the same thing you would normally do in sense with elasticsearch. Different arguements are as follows (http://docs.python-requests.org/en/master/api/) .

To set up elasticsearch we must

- Create index and type
- Populate it with everything in the database . we do this in bulk with _bulk api
- Every Time we add a new file, we need to add it to elasticsearch
- Everytime we update a file we need to update in elasticsearch

References

http://joelabrahamsson.com/elasticsearch-101/

http://docs.python-requests.org/en/master/api/

ElasticSearch Commands

Return Everything in database

```
GET _search
{
   "query" : {
       "match_all" : {}
   }
}
```

Search Everything in index

```
GET /index/_search
{
   "query" : {
       "match_all" : {}
   }
}
```

Delete Everything In Database

```
DELETE /_all
```

Delete Everything from an Index

```
DELETE index/_all
```

Search a string

https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-query-string-query.html

https://gist.github.com/acreeger/1027916

Security

https://www.elastic.co/downloads/shield