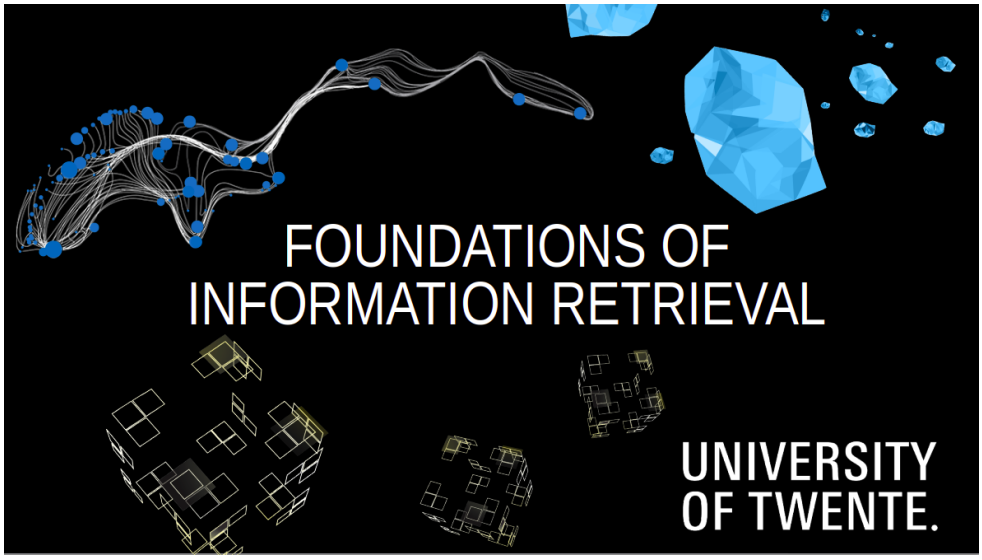
FOUNDATIONS OF
INFORMATION RETRIEVAL
UNIVERSITY
OF TWENTE.

UNIVERSITY OF TWENTE.

FIR Assignment 01, Getting Started

Foundations of Information Retrieval

by Djoerd Hiemstra, Dolf Trieschnigg, and Theo Huibers

Welcome to the first assignment of Foundations of Information Retrieval. In this assignment you will: 1) Prepare the practical work environment in docker 2) register for the practial work 3) submit your first `assignment`, and 4) get started with Elasticsearch.

> This tutorial contains background information in indented text blocks like this one. If you're doing fine and you understand the assignments, then it is safe to skip the text blocks.

1. Preparation

1.1 Install Docker

For the practical work, we will use Docker. Download and install Docker community edition which can be downloaded here: https://www.docker.com/products/docker-engine (https://www.docker.com/products/docker-engine)

> Docker allows us to all work in the same environment. With a single command you can run all the tools you need for the practical work. We will use a single docker machine (a virtual machine) in which 4 container are run. Each container provides an isolated environment for a program to run. The containers can communicate using their own > network, provided by Docker.
>
> If you are on **Linux**, you will have to separately install `docker-compose`, see: https://docs.docker.com/compose/install/ (https://docs.docker.com/compose/install/) . To run `docker-compose` without `sudo`, you have to add yourseld (`$USER`) to the docker group as follows: `sudo gpasswd -a $USER docker`. If Elasticsearch exits with the error "max virtual memory areas is too low", then increase the maximum virtual memory size with: `sudo sysctl -w vm.max_map_count=262144`

1.2 Download and run the Docker environment

Download the Docker environment for the practical work here: http://circus.ewi.utwente.nl/fir2018/docker.tgz (http://circus.ewi.utwente.nl/fir2018/docker.tgz)

Extract the archive to its own directory. Open a terminal (Linux/Mac) or command prompt (Windows) and navigate to the directory.

You can now start the docker environment with a single command:

```
docker-compose up -d
```

> -d tells docker to run the Docker containers in the background.

The first time you run this command this might take some time. Docker needs some time to download the required images. When the command (succesfully) completes, docker has three containers running:

- notebook, a Jupyter notebook environment.
- elasticsearch, an Elasticsearch instance.
- kibana, a Kibana instance.

You can check the status of the containers with the following command

```
docker-compose ps
```

> `ps` stands for process status

This will give a listing like this:

```
        Name                              Command               State
           Ports
-------------------------------------------------------------------------
-----------------------------
irpractice_elasticsearch_1   /usr/local/bin/docker-entr ...   Up      0.
0.0.0:9200->9200/tcp, 9300/tcp
irpractice_kibana_1          /bin/bash /usr/local/bin/k ...   Up      0.
0.0.0:5601->5601/tcp
irpractice_notebook_1        tini -- start-notebook.sh  ...   Up      0.
0.0.0:8888->8888/tcp
```

The listing shows a number of ports are forwarded to your own machine:

- Elasticsearch forwards ports 9200. On this port Elasticsearch accepts REST api calls. Open http://localhost:9200 (http://localhost:9200) to see Elasticsearch running.
- Kibana forwards port 5601. Kibana provides a web interface to inspect and manage Elasticsearch servers. Open http://localhost:5601 (http://localhost:5601) to open the Kibana interface.
- Notebook forwards port 8888. On this port you can access this Jupyter notebook. Open http://localhost:8888 (http://localhost:8888) to open Jupyter notebook. Use `genomics4all` as the password.

> When you are finished with your practical work, you can shut down the environment with:
> `docker-compose down`

The directory with docker files you will find a number of folders:

- `esdata`, this stores the Elasticsearch indices you will create during the practical work.
- `notebooks`, containing the Jupyter notebooks you have to complete for the practical work.
- `notebooks/data`, will be used for storing the document collection used for the practical work.

1.3 Read Elasticsearch getting started

Read "Elasticsearch: Getting Started (https://www.elastic.co/guide/en/elasticsearch/reference/current/getting-started.html)"

1.4 Open the Jupyter notebook

Open http://localhost:8888 (http://localhost:8888) in your browser (use the password `genomics4all`) and open this notebook called `FIR01.ipynb`.

2. Register for the practical work

Use the following link to register yourself for the practical work.

Register here (http://circus.ewi.utwente.nl:8880/register)

You will receive an e-mail with your `username` and `token` you need for submitting assignments. Open `myusername.py` (/edit/myusername.py) (use "File" -> "Open" from this Jupyter notebook) and set the username and token you received by e-mail.

3. Submit your first assignment

Execute the following code to check your registration works.

In [ ]:

```
import firutils

firutils.say_hi('hello!')  # say hi to our server!
```

> Our notebooks will use the programming language Python. If you do not know Python at the moment, don't worry, we will get to that in a later assignment.

4. Getting started with Elasticsearch

We base our first assignment on the "Elasticsearch, reference guide (https://www.elastic.co/guide/en/elasticsearch/reference/current/getting-started.html)". You can skip the section on Installation (https://www.elastic.co/guide/en/elasticsearch/reference/current/_installation.html), because Elasticsearch is already started using Docker.

> If you feel adventurous (we do **not** recommend this!), you can follow the Installation (https://www.elastic.co/guide/en/elasticsearch/reference/current/_installation.html) to run Elasticsearch on your laptop without Docker. But beware, your system will now be different from the ones of your colleagues and they might not be able to help you if you have problems that are specific to your system, your operating system, or your Elasticsearch version.

4.1 The REST API

Elasticsearch runs its own server that can be accessed by a regular web browser as the client, for instance by opening this link in your browser: http://localhost:9200 (http://localhost:9200).

> Services on the internet usually follow the client-server model (https://en.wikipedia.org/wiki/Client%E2%80%93server_model), where providers of a service are called *servers*, and requesters of a service are called *clients*. Web services are identified a by link called the Unified Resource Locator (URL). In the above URL `localhost` is the name of your local machine (the guest machine), and 9200 is the port number of the Elasticsearch service. Port numbers are used because servers might provide multiple types of services; they might for instance also send email or show normal web pages (for which the standard port numbers are respectively 25 and 80).

Elasticsearch will respond with something like:

```
{
  "name" : "epRATWu",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "KsOTBsyeTmy6fJCcZ64d_A",
  "version" : {
    "number" : "6.2.4",
    "build_hash" : "ccec39f",
    "build_date" : "2018-04-12T20:37:28.497551Z",
    "build_snapshot" : false,
    "lucene_version" : "7.2.1",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

If you see this, then your Elasticsearch node is up and running. The RESTful API uses simple text or JSON over HTTP.

REST, API, JSON, HTTP, that's a lot of abbreviations! It is good to be familiar with the terminology. Let us explain: The Elasticsearch response is not (only) intended for humans. It is supposed to be used by applications that run on the client machines, and therefore the interface is called an Application Programming Interface (API). The API uses a format called JSON (JavaScript Object Notation), which can be easily read by machines (and humans). The API sends its JSON response using the same method as your web browser displays web pages. This method is called HTTP (Hyper Text Transfer Protocol), and it is the reason you can inspect the response in a normal web browser. APIs that use HTTP are called RESTful interfaces. REST stands for REpresentational State Transfer, arguably one of the simplest ways to define an API.

4.2 Kibana, cURL, and more cURL

A bit of background on hostnames in docker: When running multiple Docker containers using docker-compose, each container has its own hostname in a local network. The docker-compose file started three containers:

1. notebook, running a jupyter notebook at port 8888
2. elaticsearch, running an elasticsearch server instance at port 9200
3. kibana, running a kibana server instance at port 5601 The ports are all forwarded to your local machine, that's why you can access this notebook from http://localhost:8888 (http://localhost:8888) in your browser, and you can access your elaticsearch instance on http://localhost:9200 (http://localhost:9200). However, if you want to access elasticsearch from within the notebook container, you have to use `elasticsearch` as the hostname. So if you start an additional terminal container you can access elasticsearch using the hostname `elasticsearch` (and not `localhost`!)

You can interact with your Elasticsearch service in different ways. In this first assignment we will describe three ways. Later during the practical work we will use the Python Elasticsearch client.

1. Using the Kibana Console
2. Using cURL
3. Using cURL from a Jupyter notebook (not recommended)

Kibana

Kibana provides a web interface to interact with your Elasticsearch service. It's available from http://localhost:5601 (http://localhost:5601). You can use Kibana to create interactive dashboards visualizing data in your Elasticsearch indices. It also provides a console to execute Elasticsearch commands. Its available from http://localhost:5601/app/kibana#/dev_tools (http://localhost:5601/app/kibana#/dev_tools)

Many examples from the Elasticsearch user guide can be directly executed in Kibana by clicking the `VIEW IN CONSOLE` button.

cURL

CURL (https://en.wikipedia.org/wiki/CURL) is a software tool that enables you to execute HTTP method requests from the commandline. The name originally stood for "see URL".

Curl is already installed in the docker notebook image. Let's open a bash terminal in the notebook container by executing the following docker-compose command:

```
docker-compose exec notebook bash
```

This should give you a bash shell, looking something like this:

```
jovyan@ccd6fdce539a:~$
```

You can exit the shell by executing `exit` (and start it again executing the docker-compose command). You can execute curl commands on this prompt, for instance retrieving the Elasticsearch state. Note you have to use `elasticsearch` as the hostname:

```
jovyan@ccd6fdce539a:~$ curl elasticsearch:9200
{
  "name" : "epRATWu",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "KsOTBsyeTmy6fJCcZ64d_A",
  "version" : {
    "number" : "6.2.4",
    "build_hash" : "ccec39f",
    "build_date" : "2018-04-12T20:37:28.497551Z",
    "build_snapshot" : false,
    "lucene_version" : "7.2.1",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

The `COPY AS CURL` button found in the Elasticsearch user guide gives you the curl command. Make sure you replace `localhost` with the hostname `elasticsearch` in these commands!

cURL from this notebook

Alternatively, jupyter notebooks allow you to directly execute cURL commands (or other shell commands), by starting a line of code with an exclamation mark (see example below). Plase be warned: when executing commands which result in long output (for instance when indexing a large number of documents), stick to the terminal to execute curl commands. Jupyter might freeze when handling long output from the shell.

In [ ]:

```
! curl http://elasticsearch:9200  # note the ! before curl
```

4.3 Getting started with Elastic Search

From here, follow the Elasticsearch getting started guide until the "Conclusion":

1. Exploring Your Cluster
2. Modifying Your Data
3. Exploring Your Data
4. Conclusion

If you carried out all the assignments, the following questions should be easy to answer. Answer the following questions about the `bank` index you created.

Q1. What is the account_number of the account with the largest balance?

In [ ]:

```
import firutils
firutils.submit_largest_balance(123)  # change to the id you found
```

Q2. How many accounts have a balance between 25000 and 45000 (including 25000 and 45000) and are from the state `"IL"`?

In [ ]:

```
import firutils
firutils.submit_number_of_accounts(123)  # change to the number of accounts you
 found
```

Q3. How can you make Elasticsearch only return the `state` attribute of the returned hits (so `_source` only includes the `state`)?

In [ ]:

```
# how should you alter the query to only return the `state` of the retrieved hit
s?
query = {
  "query": { "match_all": {} },
# BEGIN ANSWER
# END ANSWER
}
import firutils
firutils.submit_query_state_only(query)
```