

# Use GeoLite2 To Create A Searchable IP and GeoLocation Database

Practical Exam



# Goal

Maxmind.com provides regulary exports of worldwide IP and Geolocation data:

- <https://dev.maxmind.com/geoiip/geolite2-free-geolocation-data>

```
curl -s http://ifconfig.me  
88.130.59.75
```

1

```
network,geoname_id,registered_country_geoname_id,represented_country_geoname_id,is_anonymous_proxy,is_satellite_provider,postal_code,latitude,longitude,accuracy_radius  
88.130.59.0/24,2939623,2921044,,0,0,85221,48.2600,11.4340,50  
[...]
```

2

```
geoname_id,locale_code,continent_code,continent_name,country_iso_code,country_name,subdivision_1_iso_code,subdivision_1_name,subdivision_2_iso_code,subdivision_2_name,city_name,metro_code,time_zone,is_in_european_union  
3205335,de,EU,Europa,DE,Deutschland,BY,Bayern,,,Höhenkirchen-Siegertsbrunn,,Europe/Berlin,1  
2939623,de,EU,Europa,DE,Deutschland,BY,Bayern,,,Dachau,,Europe/Berlin,1  
3207410,de,EU,Europa,DE,Deutschland,BY,Bayern,,,Rödentel,,Europe/Berlin,1  
3207412,de,EU,Europa,DE,Deutschland,BY,Bayern,,,Röslau,,Europe/Berlin,1  
3208324,de,EU,Europa,DE,Deutschland,BY,Bayern,,,Asbach-Bäumenheim,,Europe/Berlin,1  
[...]
```

*GeoLite2-City-Blocks-IPv4.csv*

*GeoLite2-City-Locations-[XX].csv*



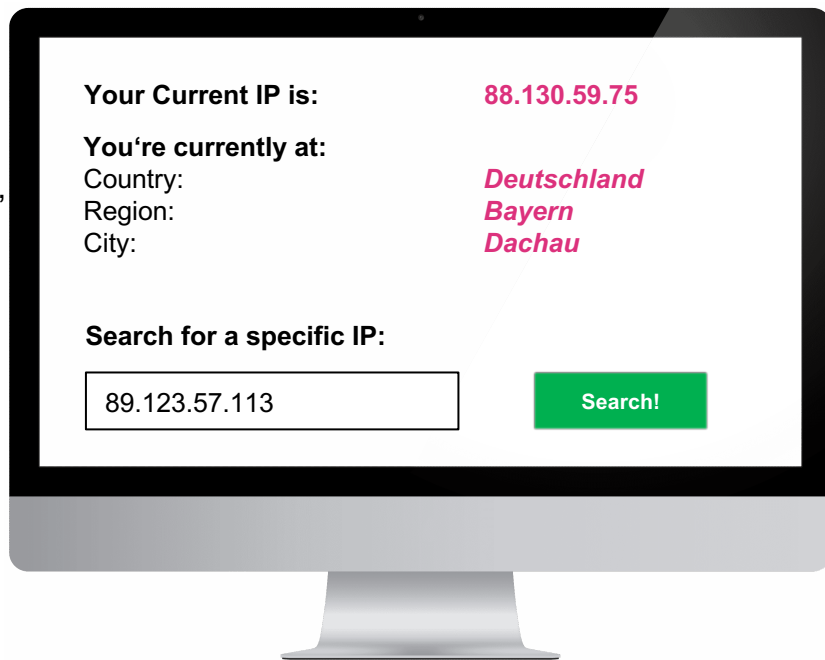
[www.marcel-mittelstaedt.com](http://www.marcel-mittelstaedt.com)

# Goal

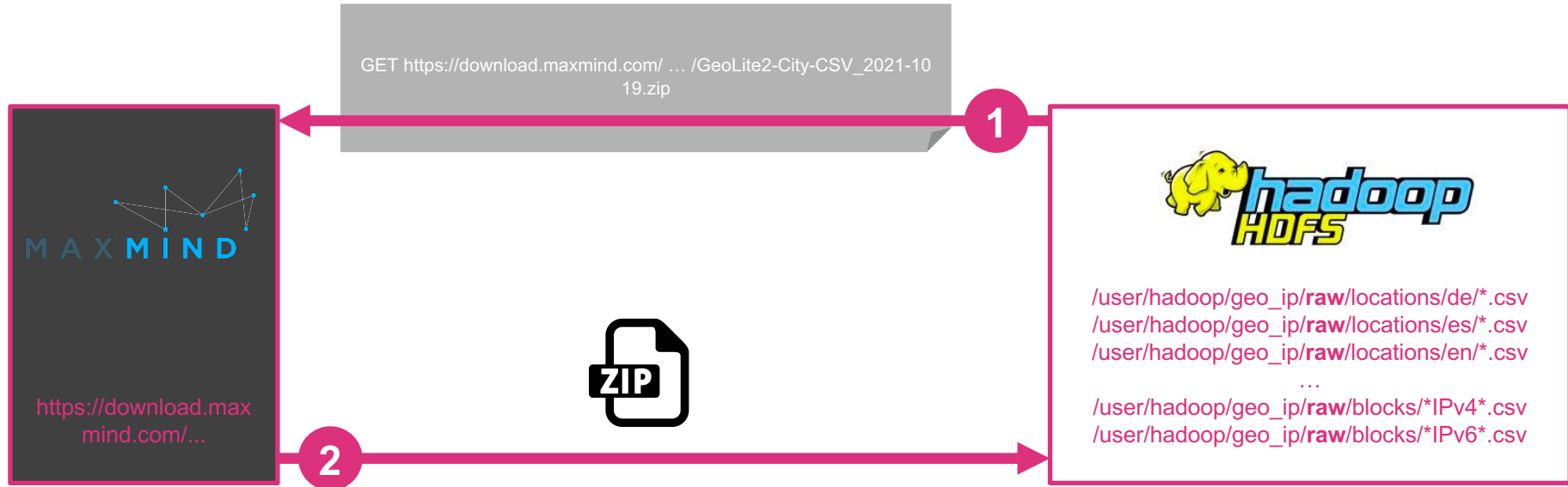
We want to make use of this data to build a real time IP-Geolocation resolution as well as a searchable database for Ips and related Geolocations.

## Workflow:

- **Gather data** from maxmind.com
- **Save raw data** (CSV files) to HDFS (partitioned by country code, e.g. *de*, *es*, *en*...)
- **Optimize, reduce** and **clean raw data** and save it to **final** directory on HDFS
- **Export** Geolite2 data to **end-user database** (e.g. MySQL, MongoDB...)
- Provide a simple **HTML Frontend** which is able to:
  - **determine** a user's IP address, **lookup** and **show Geolocation**
  - process user input (IP...) and check against end-user database
  - Display result Geolocation
- The whole data workflow **must be implemented** within an ETL **workflow tool** (e.g. Pentaho Data Integration or Airflow) and **run automatically**



# Dataflow: 1. Get Geolite2 Data



# Dataflow: 2. Raw To Final Transfer



/user/hadoop/geo\_ip/**raw**/locations/de/\*.csv  
/user/hadoop/geo\_ip/**raw**/locations/es/\*.csv  
/user/hadoop/geo\_ip/**raw**/locations/en/\*.csv  
...  
/user/hadoop/geo\_ip/**raw**/blocks/\*IPv4\*.csv  
/user/hadoop/geo\_ip/**raw**/blocks/\*IPv6\*.csv



1

- move data from *raw* to *final* directory
- **optimize** and **reduce** data structure for later query purposes if necessary
- remove duplicates if necessary
- ...



/user/hadoop/geo\_ip/**final**/locations/de  
/user/hadoop/geo\_ip/**final**/locations/es/  
/user/hadoop/geo\_ip/**final**/locations/en/  
...  
/user/hadoop/geo\_ip/**final**/blocks/\*IPv4\*  
/user/hadoop/geo\_ip/**final**/blocks/\*IPv6\*

# Dataflow: 3. Enhance Data And Save Results



```
/user/hadoop/geo_ip/final/locations/de  
/user/hadoop/geo_ip/final/locations/es/  
/user/hadoop/geo_ip/final/locations/en/  
...  
/user/hadoop/geo_ip/final/blocks/*IPv4*  
/user/hadoop/geo_ip/final/blocks/*IPv6*
```



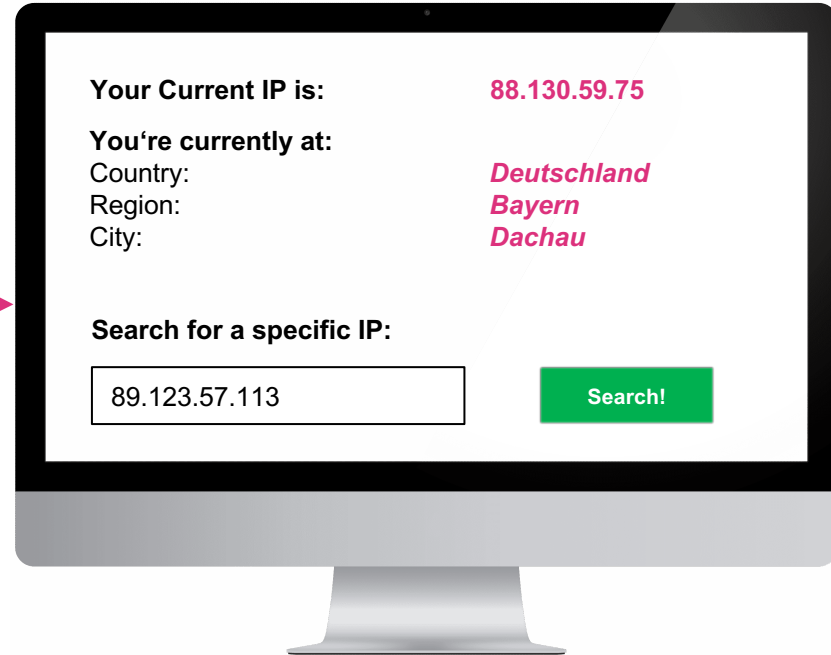
1

- enhance data (e.g. for later querying)
- use *Hive*, *Python*, *Spark* or *PySpark*
- save everything to a end-user database (e.g. *MySQL*, *MongoDB*)





# Dataflow: 4. Provide Simple Web Interface



- Provide a simple **HTML Frontend** which is able to:
  - **determine** a user's IP address, **lookup** and **show Geolocation**
  - process user input (IP...) and check against end-user database
  - Display result Geolocation