How We Design Service APIs for Big Data Sources

Imaduddin Amin – Data Engineer at Pulse Lab Jakarta

United Nations Global Pulse

Vision	Big Data harnessed responsibly as a public good
Mission	Accelerate discovery and adoption of big data innovation for sustainable development and humanitarian action

More Information : https://unglobalpulse.org

Data and Infrastructure

Data

- We get our data from different partners with different regulations and policies
- Our data come in **high volume, high velocity** and **high variety**
- Data will be used in various dashboards, toolkits and research projects.

Infrastructure

- Local Server
- Cloud Based Infrastructure (Amazon Web Services)

Datasets and Platforms

Dataset

- Social Media Data
- CDR (Call Detail Record)
- Transactions data (financial, transport)
- World AIS dataset
- Climate and weather dataset
- Point of Interest
- Spatial data and Satellite Imagery
- Others

Platforms and Toolkits

- Social Media Toolkit
- Social media dashboard / Social Listener
- Haze Gazer
- Cyclone Monitoring
- VAMPIRE
- Others

Python at our Office

Why Python ?

- General purpose dynamic programming
- Comprehensive and large standard library
- Integration Feature
- Great support

How We Utilize Python

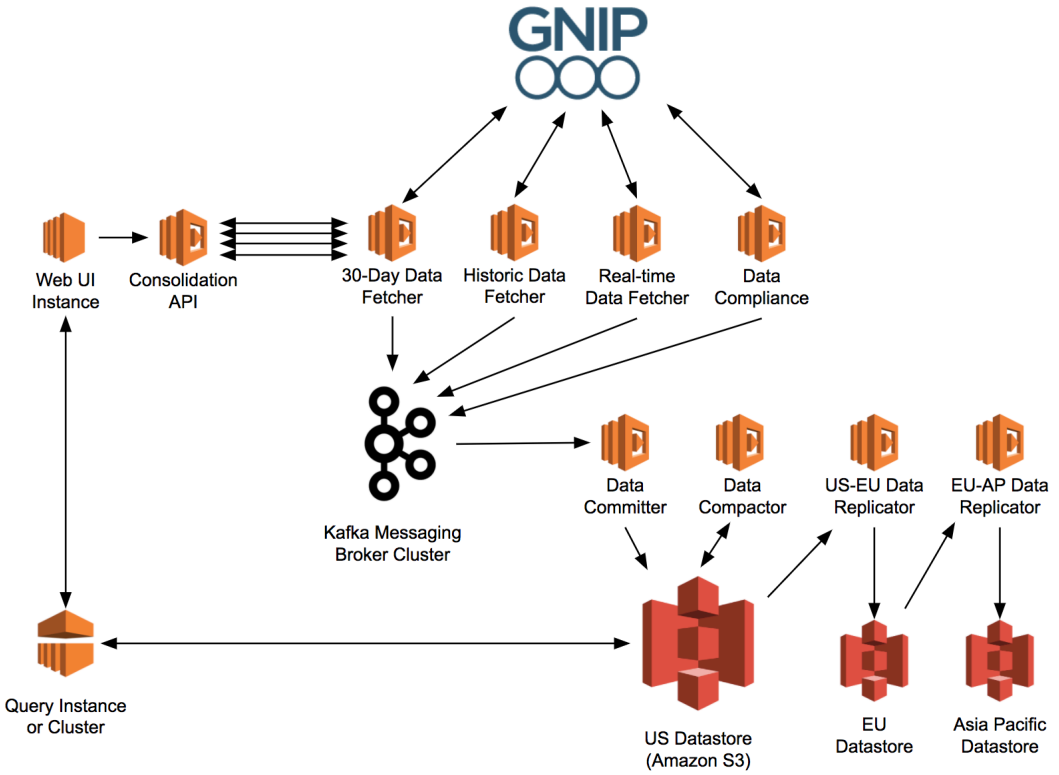
- Data Collecting
- Data Processing
- Data Analyzing
- Manage Data Services

Our data come in high volume, high velocity and **high variety**. And python is a one-stop shop. Python's library is extensive and cover our main needs.

Data Collecting

Python on

- Data Pipelining services from external API to AWS
- Data Service (PySpark) for data formatting and querying
- Inter Region data replication in AWS Lambda

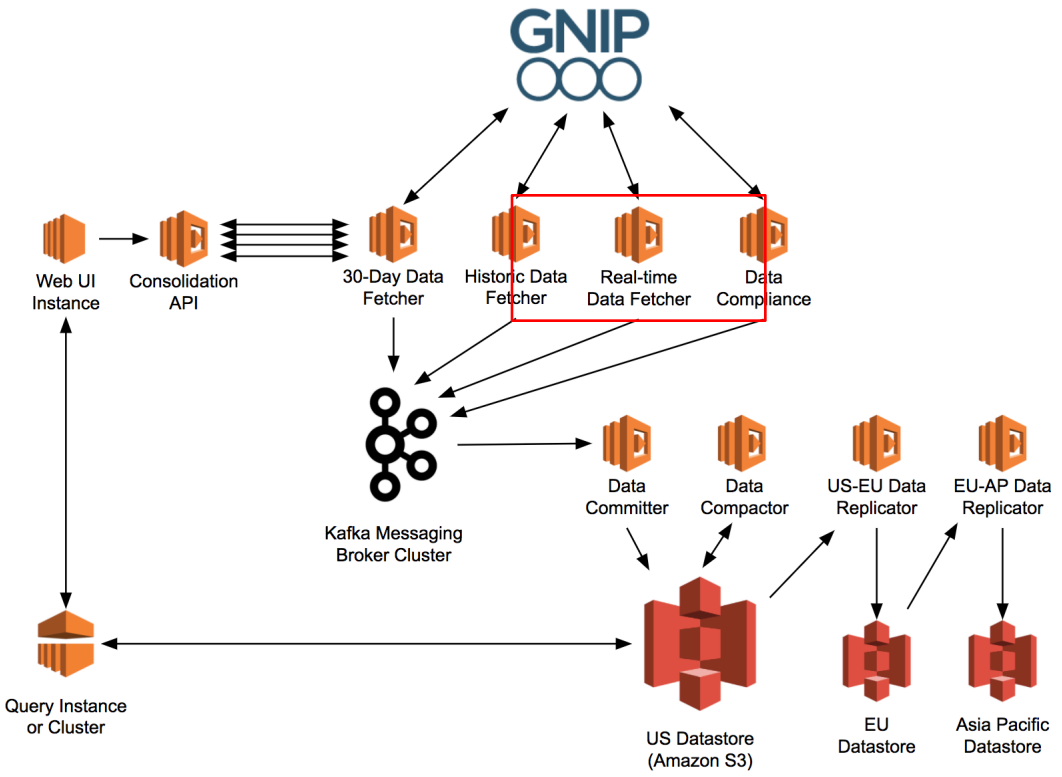


Data Collecting

Not Python

- Real-time data fetcher and data compliance. It needs a good Redis-backed bloom filter implementation (because of high-volume, high-velocity ingestion)

- Any suggestions?



Data Processing and Analyzing

Social Media Data

- NLTK for Text Mining project
- Parsing and querying using PyParsing
- GeoLocation using Python GeoCoder, PySHP, GeoText

Mobile Data

- Data transformation using Bandicoot
- Inferring pattern such as activity, mobility and others

Satellite Data

- Manipulating geospatial raster data using GDAL
- Data Transformation for example GRIB2JSON

Scientific Computation and Data Analysis

- NumPy, SciPy, Pandas (Structured data operations and manipulations)
- Scikit Learn (machine learning), Matplotlib, Searborn (Visualization)

Intermezzo : Bloom Filter

What is Bloom Filter

- A **Bloom filter** is a space-efficient probabilistic data structure, conceived by Burton Howard Bloom , that is used to test whether an element is a member of a set.

What is the difference to feature-hashing?

- Bloom filter offers a better space-efficient compare to other. It does not store the elements it self.
- But … Bloom filter does not guarantee 100 % accuracy. It returns False Positive
- query returns either “inside set (may be wrong)” or “definitely not in set”.

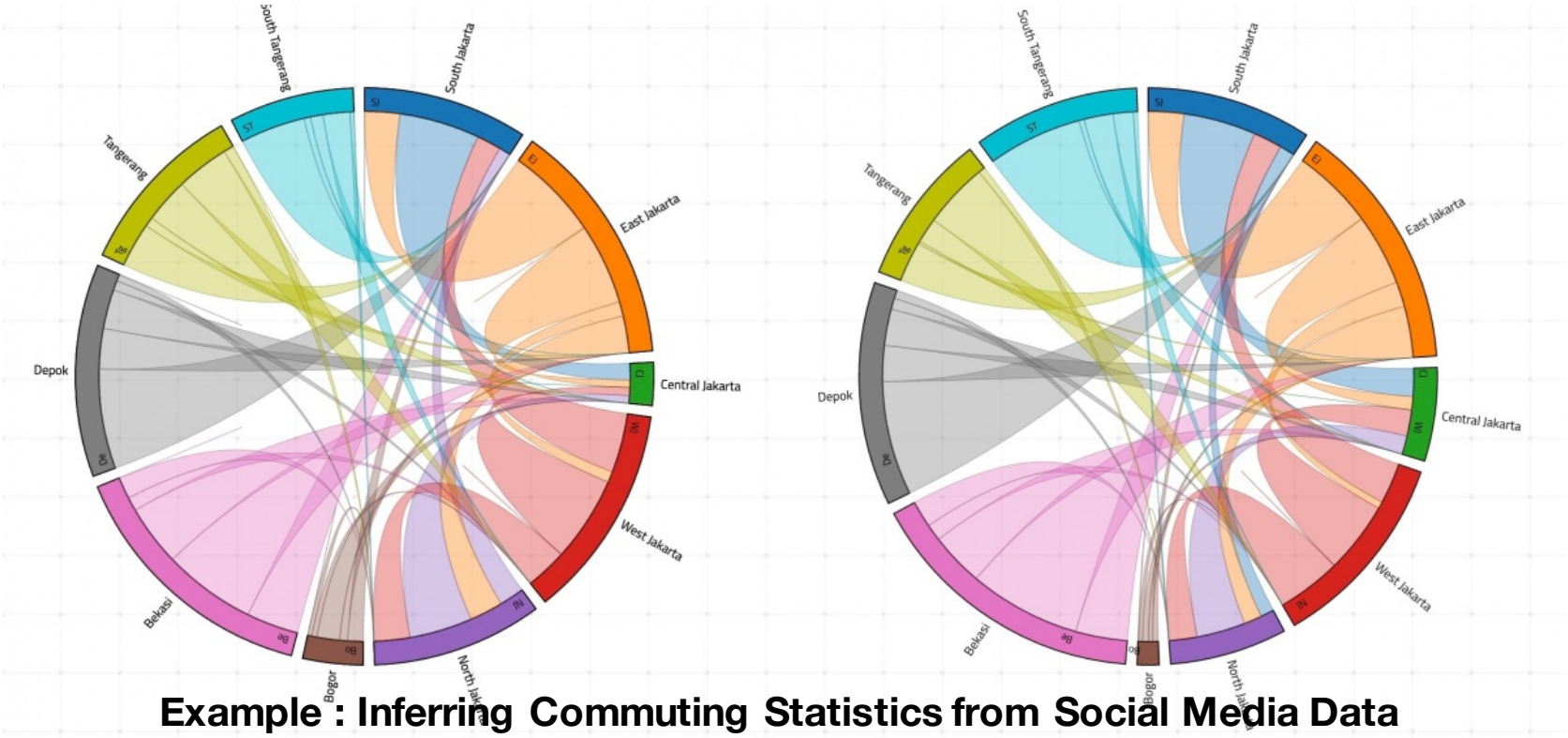
When to use bloom filter ?

- When false positive is acceptable at some degree (e.g 1%)
- When you have space issue on your environment
- https://en.wikipedia.org/wiki/Bloom_filter#Examples

http://llimllib.github.io/bloomfilter-tutorial/

Intermezzo : Social Media Data

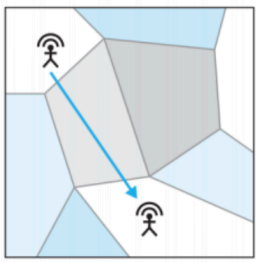
Social media data not only contains **text information** but also includes **actor data** (who posted the content and its attributes), **generator** (utility used to post the content), **location** (geo location, city, country level), **entities** (mentions, hashtags, media), **posted time**

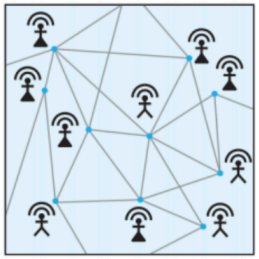


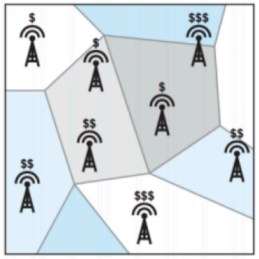
Example : Inferring Commuting Statistics from Social Media Data

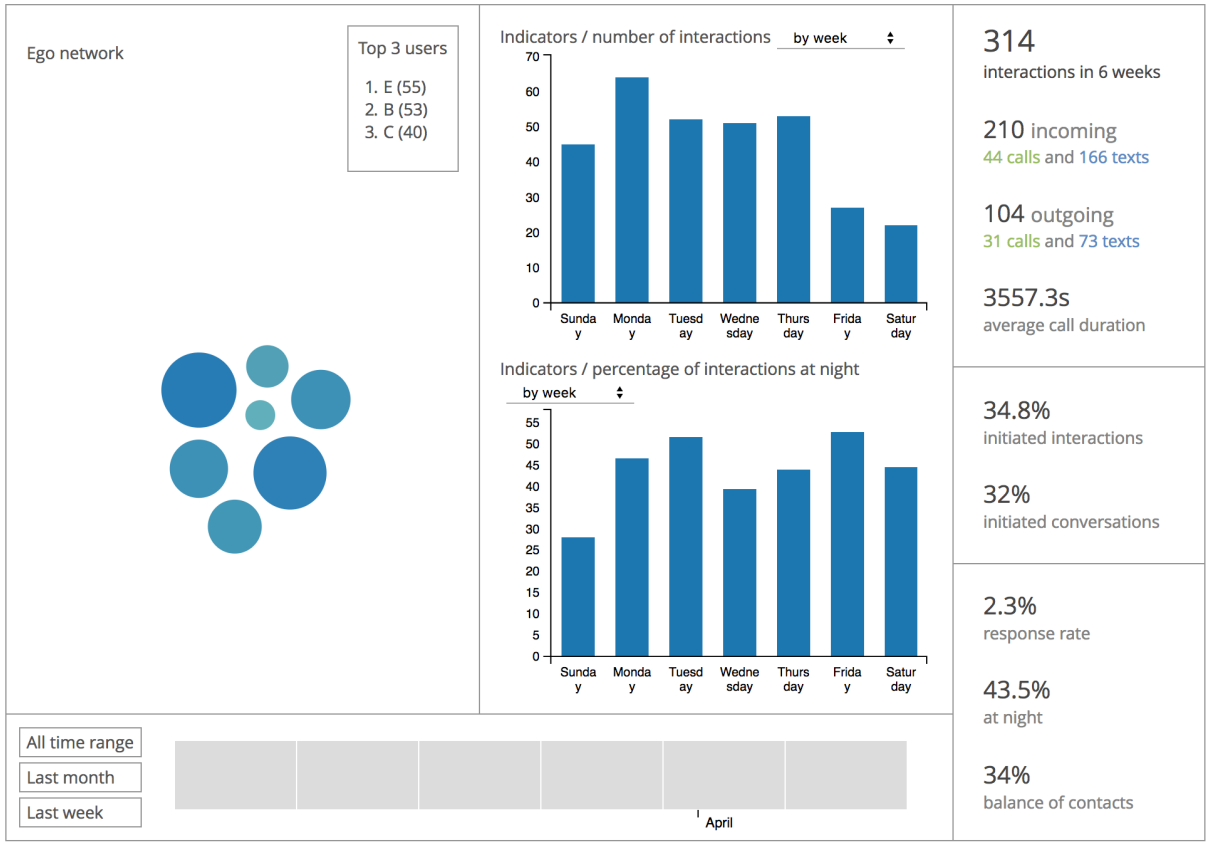
Intermezzo : Mobile Data

Whenever a mobile phone call or transaction is made, a **Call Detail Record (CDR)** is **automatically generated** by the mobile network operator.







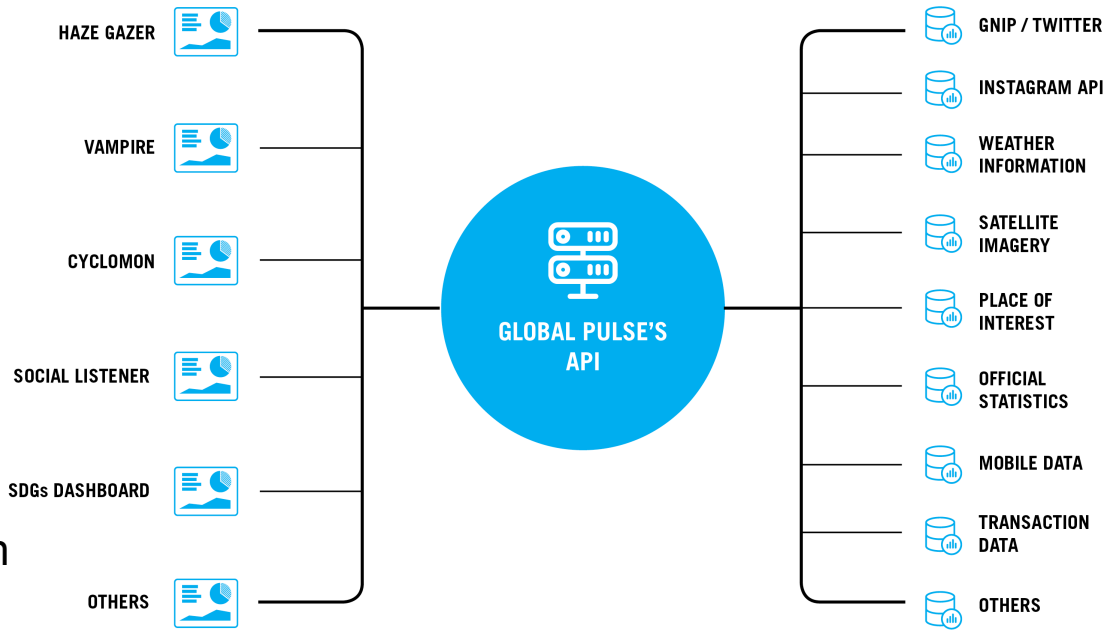


Try dataset : http://nodobo.com/release.html and Library : http://bandicoot.mit.edu/

Data Service (APIs)

Goals

- Clear separation between platform and data provider.
- Should be easy to use by multiple platforms in different servers or regions.
- Easy to manage and High Scalable, e.g. in adding new data source.



Python Web Framework

Flask

- “Micro framework” primarily aimed at small applications with simpler requirements use by multiple platforms in different servers or regions

Django

- Aimed at larger applications, include all the batteries a web application will need so developers need only open the box and start working

Source : https://www.airpair.com/python/posts/django-flask-pyramid

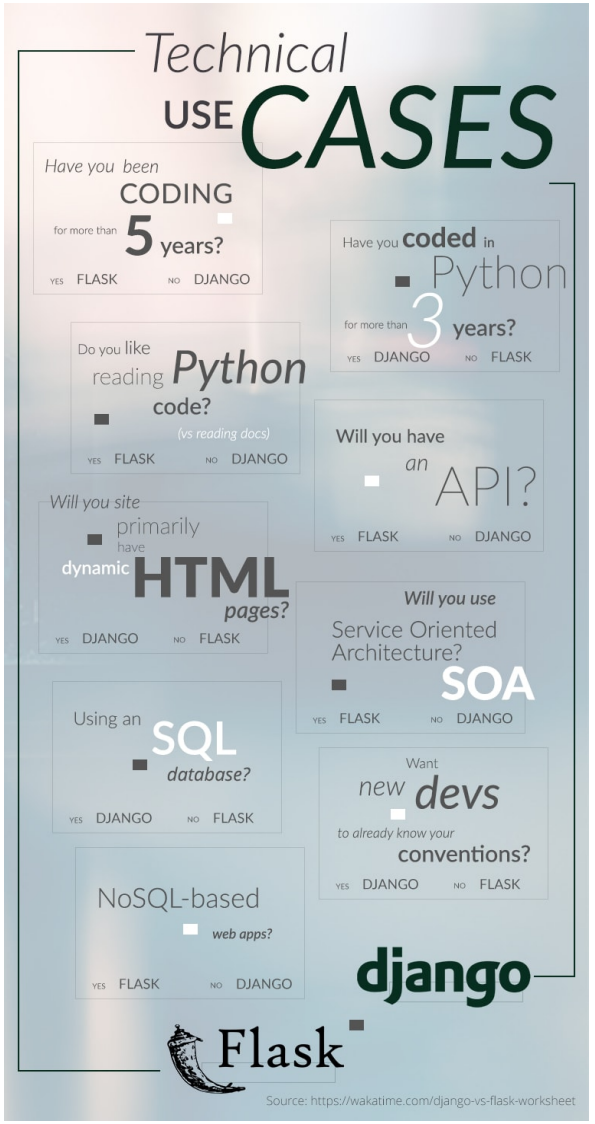
Which is Better for You ?

In our case, Flask is a better option

- Learning curves is easy
- The documentation is superb : comprehensive, full of examples, well structured
- Provides simplicity (small core) and flexibility (easily extensible)
- After all, what we need is only a web service APIs

It maybe different for you

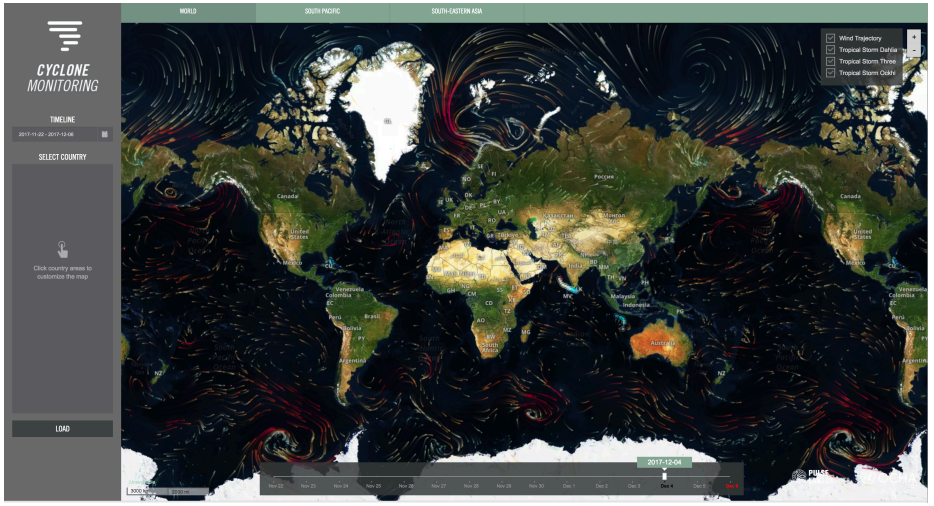
- You may need to use Django or others. Depend on your needs!
- Try : https://wakatime.com/django-vs-flask-worksheet



Case Study : Cyclone Monitoring

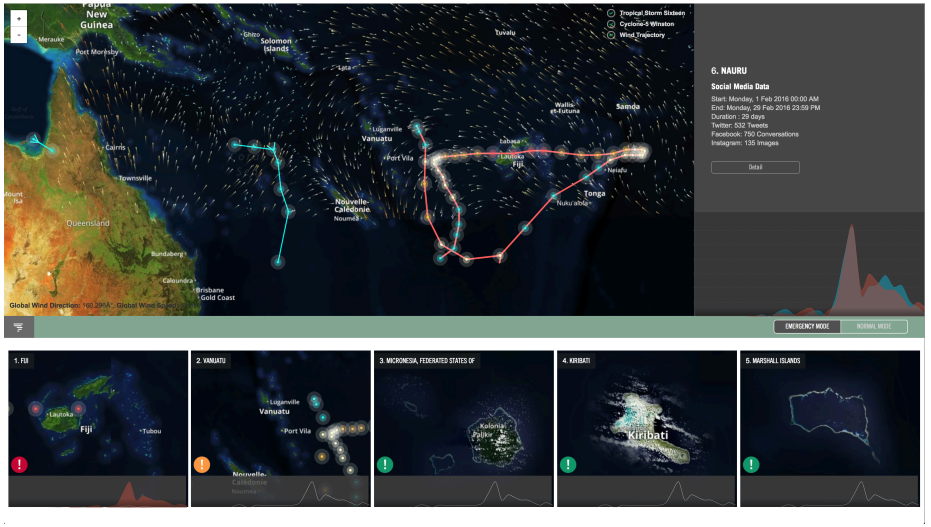
What is Cyclomon?

- A real-time analysis and visualisation platform that generates insights on behaviour patterns before, during and after cyclones using social media big data.



What is available on Cyclomon?

- It provides information from multiple data sources such as Social Media data, weather information, place of interest, citizen generated report, official statistics on population and others.



Case Study : Cyclone Monitoring

Separation between Application and Data Sources

- We use PHP for frontend application.
- Why? We need to *handover* this platform to our partner. Considering the PHP popularity, we think it is the best option for now.

Managing Data Source

- Except static information and frontend configuration, all the data from this platform connects to Data Services (API)
- This scheme allows us for easier modification both on Frontend side and backend / data provider side
- We built a corresponding APIs for each new dataset. For example :
  /cyclone/api/v0.1/cyclone_track to define cyclone trajectory
  /cyclone/api/v0.1/social_media_conversation to define social media data an so on

Case Study : Cyclone Monitoring

```python
from flask import Flask
app = Flask(__name__)

@app.route('/')
def index():
    return "Hello from Flask"

if __name__ == '__main__':
    app.run(debug=True)
```

- Flask is very light and simple.
- What if our application get larger? We can generate a “section” of our application. It makes your application more modular.
- We manage corresponding APIs for each dataset

```python
from flask import Flask

#app = Flask(__name__, static_url_path = "")
app = Flask(__name__)
# start import from modules
from cyclone.cyclone_route import cyclone_api
app.register_blueprint(cyclone_api)
from twitter.twitter_route import twitter_api
app.register_blueprint(twitter_api)

@app.errorhandler(400)
def not_found(error):
    return make_response(jsonify( { 'error': 'Bad request' } ), 400)

@app.errorhandler(404)
def not_found(error):
    return make_response(jsonify( { 'error': 'Not found' } ), 404)

@app.errorhandler(405)
def not_found(error):
    return make_response(jsonify( { 'error': 'Method Not Allowed' } ), 405)

@app.errorhandler(500)
def not_found(error):
    return make_response(jsonify( { 'error': 'Internal Server Error' } ), 500)

@app.route('/files/<path:path>')
def static_proxy(path):
  # send_static_file will guess the correct MIME type
  return app.send_static_file(path)

if __name__ == '__main__':
    app.run(host='0.0.0.0',port=5000,debug = True)
```

Things to do

- Use Versioning
- Use HTTP status codes correctly
- Use standardized data format
- Do not forget about authentication
- Rate limiting
- Keep it simple and intuitive
- …..
- .....

- and **Have a good documentation !**

We are happy to collaborate with you!

More Info : imaduddin.amin@un.or.id