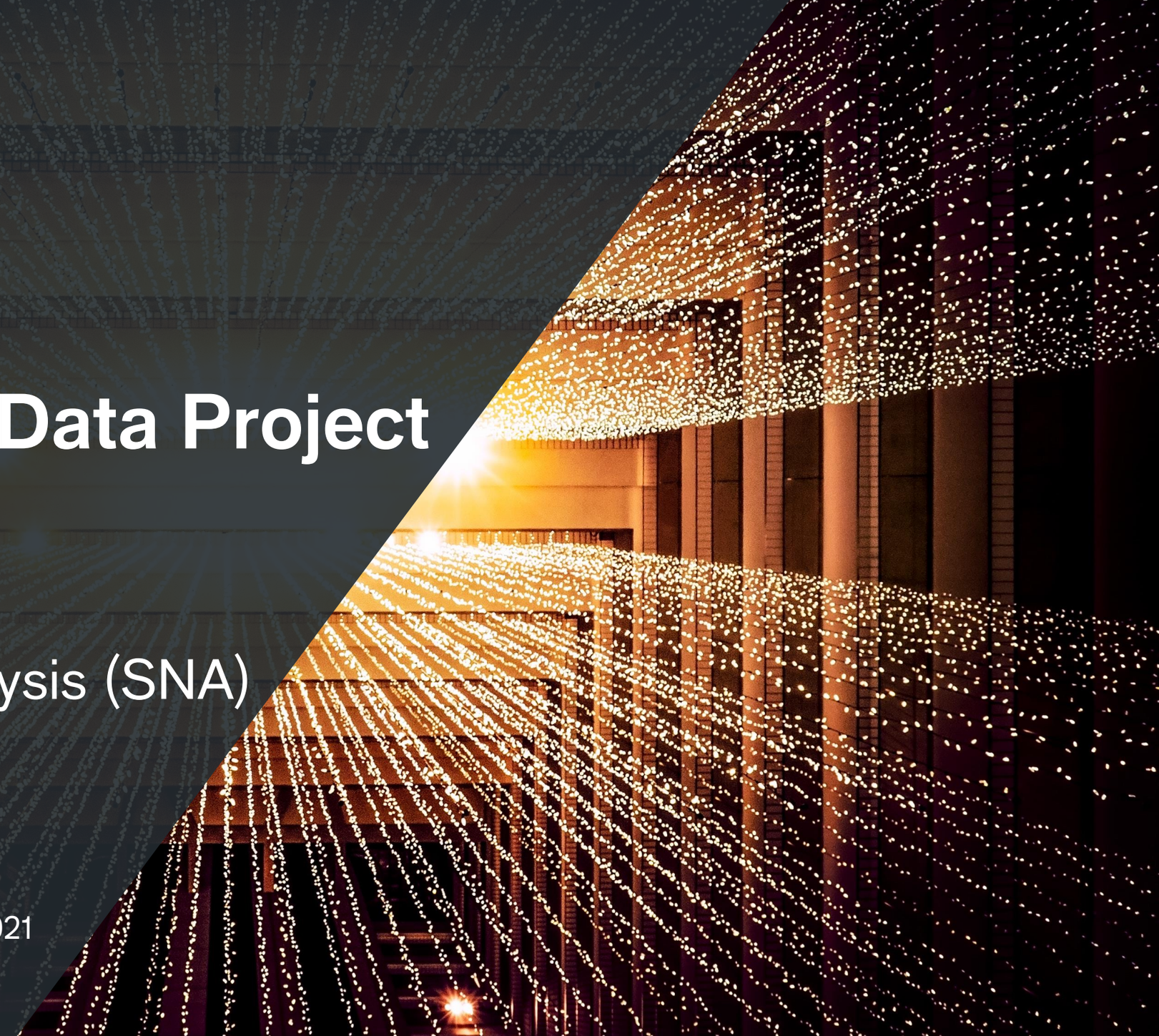
BitBang

Genesis of a Big Data Project in the “Cloud”

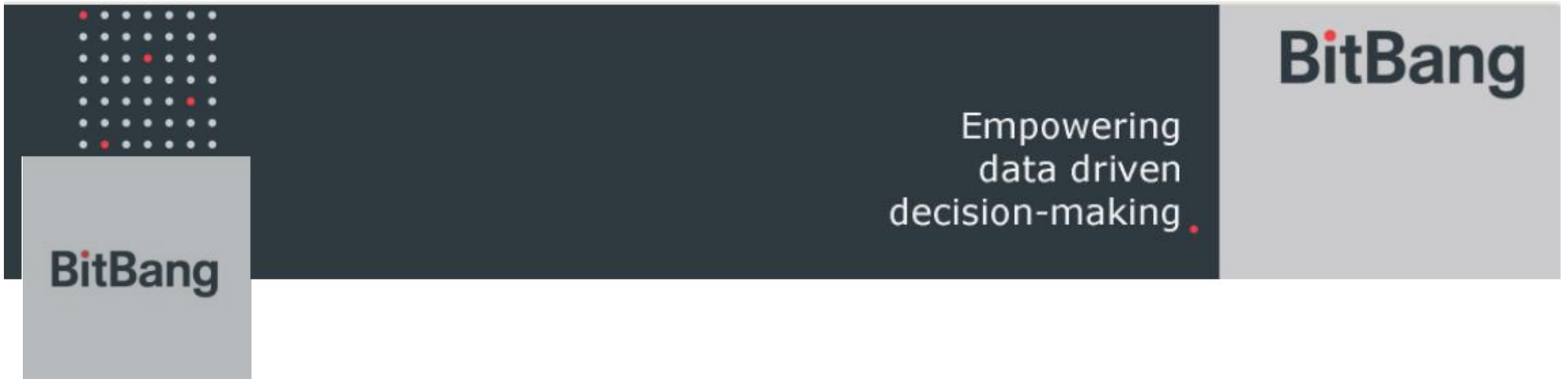
The Social Network Analysis (SNA) Case Study on AWS

Matteo Casadei

17.05.2021



My Company



https://www.linkedin.com/company/bitbang/

Who we are

BitBang was founded in 2003, with the vision that data is the most powerful asset of a business.

For over a decade, we have been providing data management consulting services and strategies to empower **Data Driven Decision-Making at Scale.**

Our services and solutions will help you transform your business through orchestrated executable data strategies that achieve desired business outcomes.

What we do

- Understand Your Goals
- Design Use Cases
- Collect Data
- Combine & Prepare
- Report & Analyze
- Model & Predict
- Manage by Objectives

Awards

2014

Forrester
Digital Analytics Agency in The New CI Services Lens

2015

CIO Review
100 Most Promising Big Data Solution Providers

2016

1IT Enterprise Magazine
25 Most Empowering Companies Big Data Special Ed.

2019

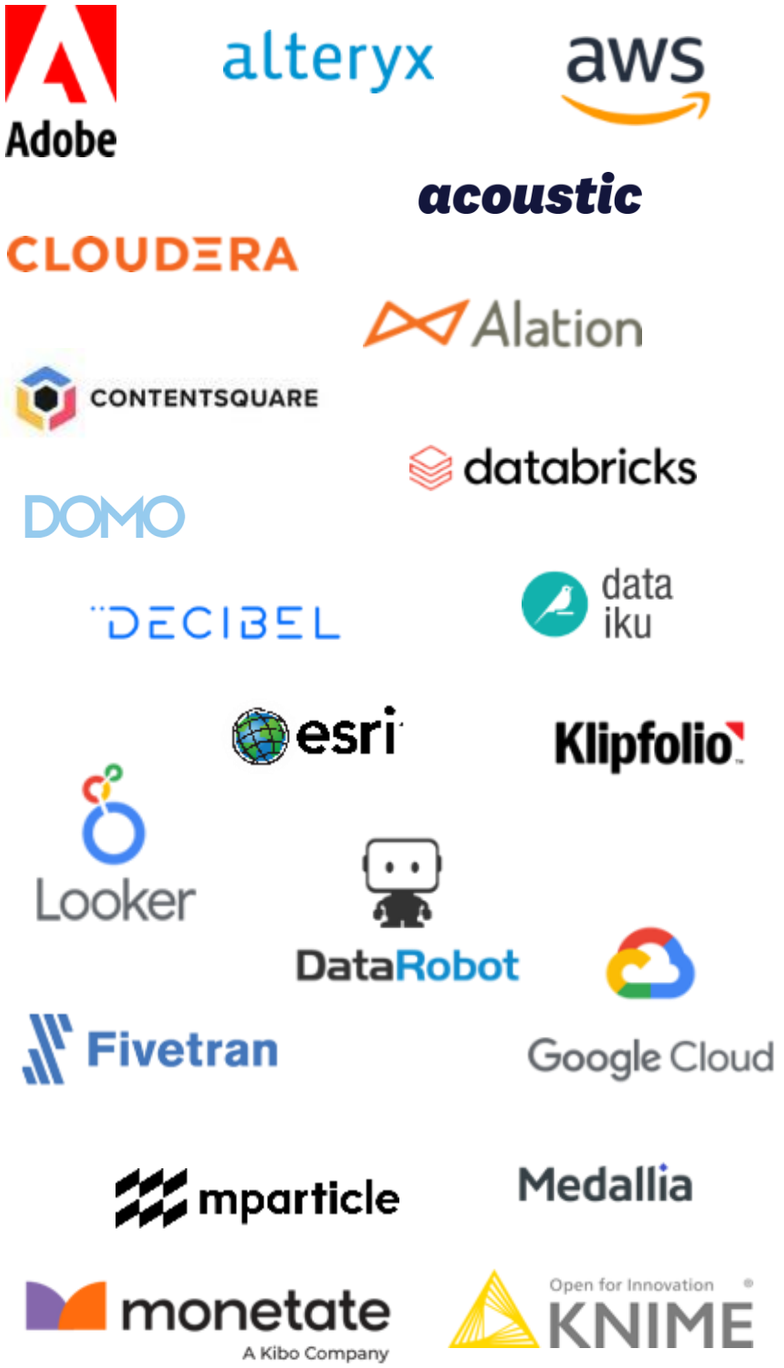
Enterprise Viewpoint
Top 20 Big Data Solution Providers

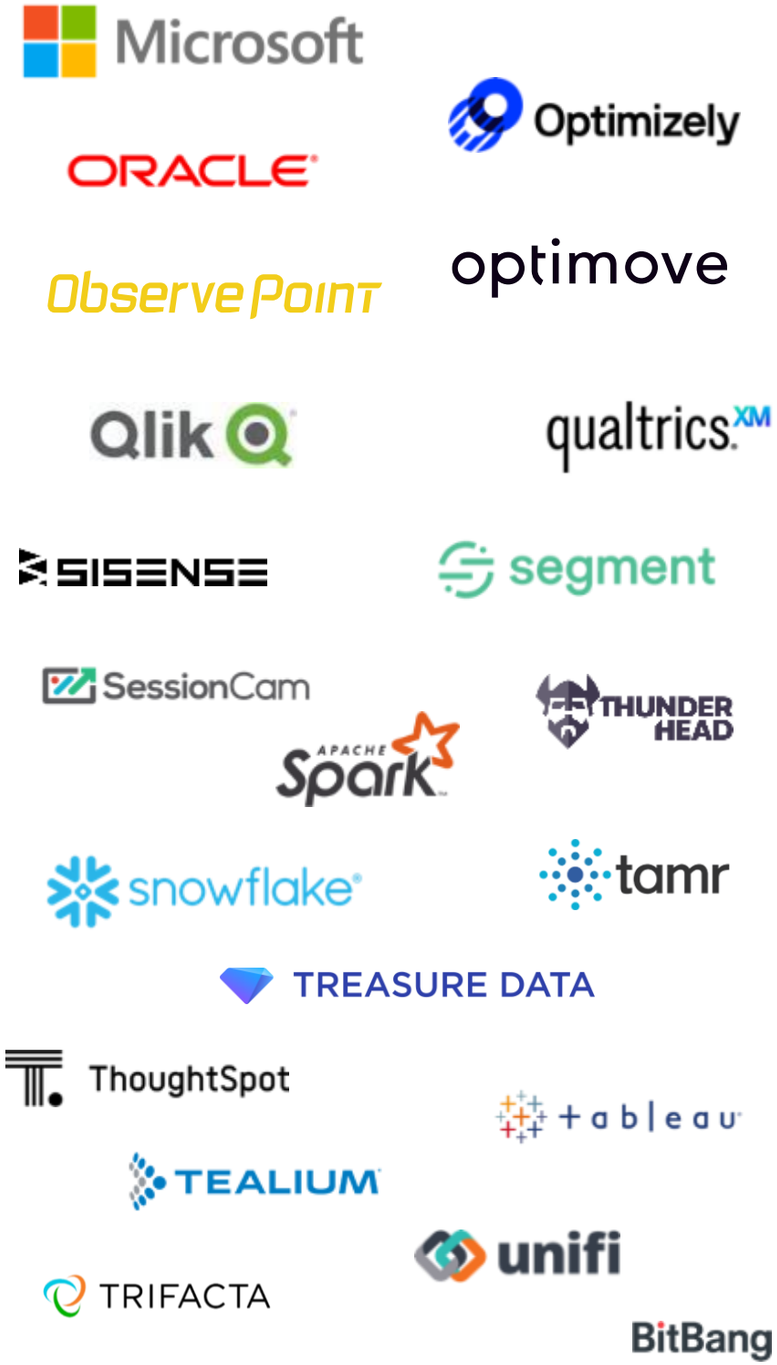
2020

CIO Applications
Top 10 Big Data Consulting/Services Companies

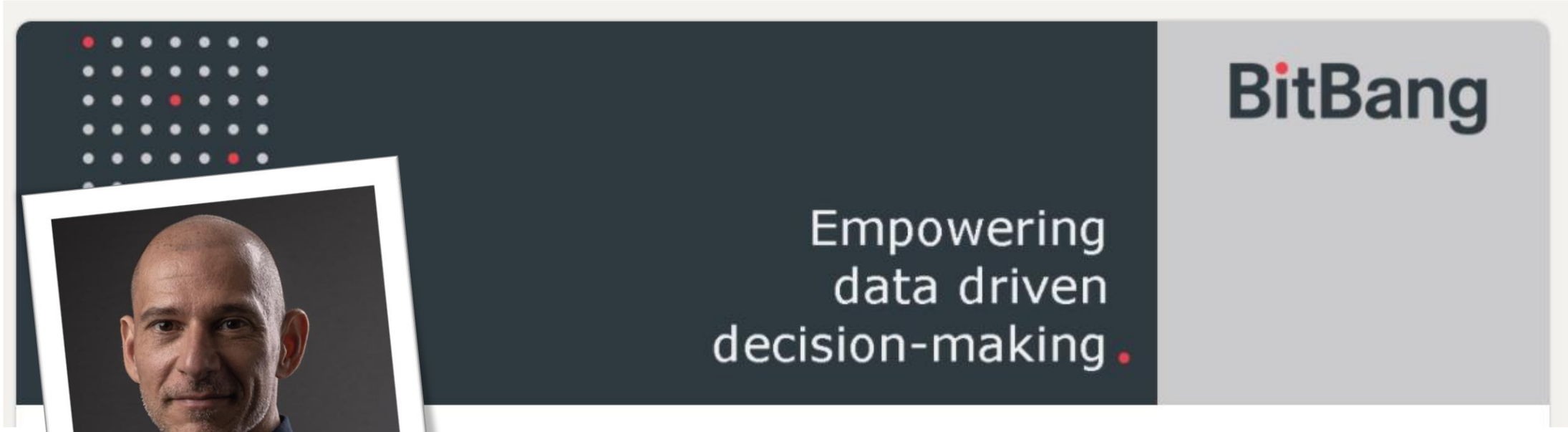
Corporate Excellence Awards
Leading CX Management & Analytics Consultancy

Partner Ecosystem





Myself



Matteo Casadei
Head of Data Science
BitBang

mcasadei@bitbang.com

The Agenda

- Start from business needs
- Translate into functional requirements
- The ‘‘typical’’ workflow of (almost) any big data project
- Choose the ‘‘best’’ technological stack: the relevance of cloud nowadays
- Social Network Analysis architecture: a solution based on AWS, relying on fully-managed components only
- Architectural improvements

Start from Business Needs

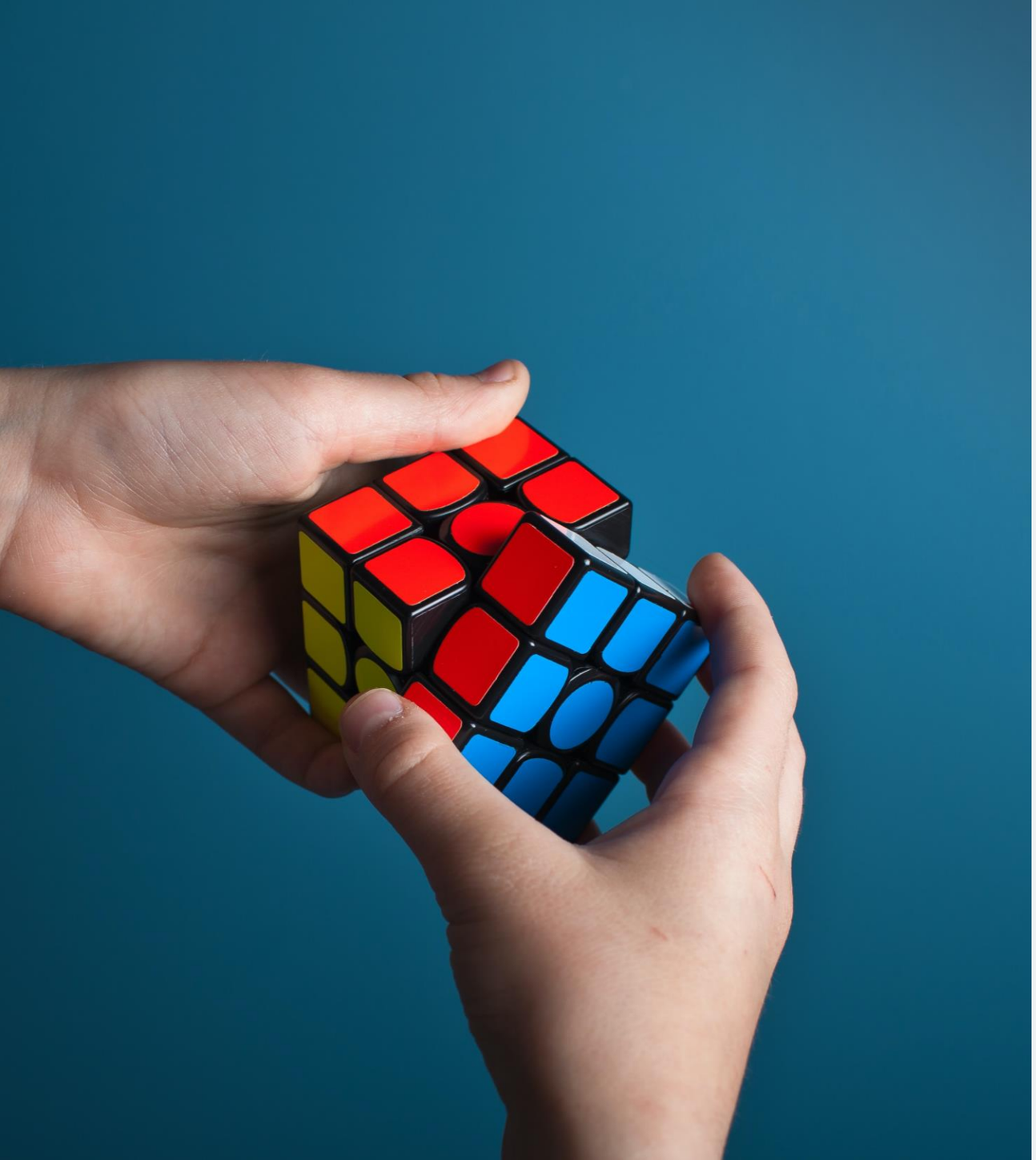
More often than not...

... Big Data Projects start from *unclear* business need!

Business needs

In other words, though business stakeholders have a pretty clear idea of what they need and want to achieve, e.g. in term of ROI, better efficiency etc.

- They do not know how to translate it *into well-stated* functional requirements
- Sometimes, they completely ignore technological issues and pitfalls



Our Job as Consultants

Translate business needs into functional requirements, of course, easy!

That means

- **Assessing** feasibility, suggesting the most appropriate approach, methodology, technological stack, so as to set the right set of expectations

- **Set-up** the whole project plan and take care of the corresponding end-to-end delivery

End-to-End Delivery?

- **Requirements and assessment**: identify *functional requirements* and the ‘‘best’’ trade-off *technological stack*
- **Design**: define *application architecture* relying on previously identified technology
- **Implementation**: well that’s exactly what you think, implementing the *big-data ‘‘solution’’*
- **Knowledge transfer**: making our client able to *manage* the application on its own
- **Maintenance**: *respond to specific issues* and take care of introducing new features

A Big-Data Project: Long Story Short

Preliminaries: Biz Requirements

Any project starts with collecting and analysing **client requirements**

A clear understanding of **business goals** lays the grounds to undertake the project with the necessary **confidence**

- Find **customers likely** to **buy a product or service** in the coming 6 months
- Detect **marketing channels** more effective than others in leading users to convert
- Highlight **influencers** in social media, e.g. Twitter
- Find **factors** with the great impact on my eCommerce's bounce rate
- Discover (usually implicit) **topics** often associated with opinions and contents about my brand

Preliminaries: Assessment

Once requirements are clearly stated, **assessment** is needed to build **awareness**

- What are the relevant biz processes?
- Which are the available data sources?
- What information in the data?

This usually done by relying on **interviews** both with **business and IT people**

- Do we have **enough data** to fulfill requirements?
- Is **data quality** in line with expected performance?
- Do we need **additional** data sources?

Hey, wait a sec, ain’t talked about technology...

Start with one *crucial* question...

What's the available technology?

Remember, not always the **"best-of-breed"** solution is also a viable one!

Which technological options?

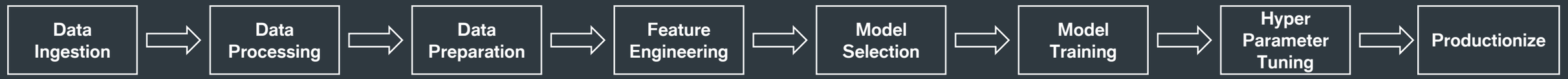
Cloud

- Customer-managed (e.g. VM instances)
- Fully-managed (e.g. AWS Lambda, etc.)

On-premise

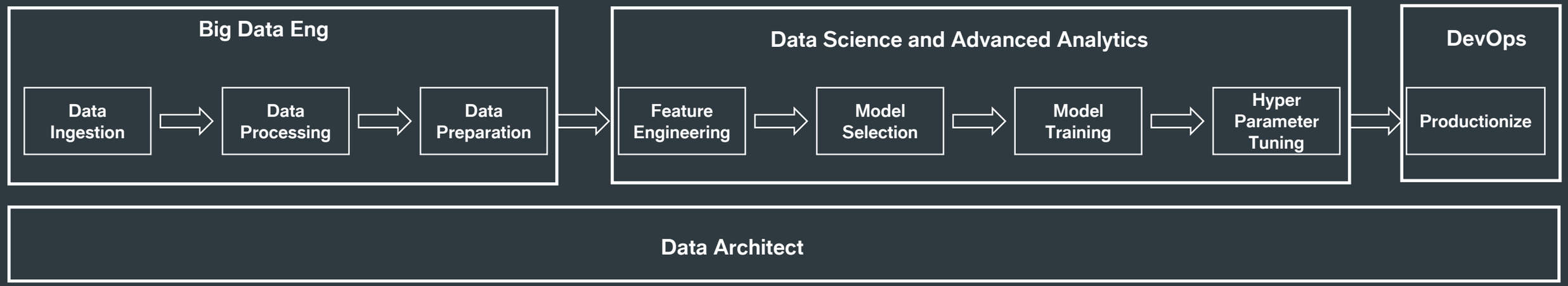
- Installed on your own infrastructure
  - That, of course, needs to be managed!
- Of course your infrastructure can be
  - Physical (bare metal)
  - Virtual (on physical or cloud VM instances, etc.)

A Big-Data Project Typical Workflow



A Big-Data Project: the Usual Workflow

The workflow typical of any big data project is the result of several **BitBang's teams** actively **collaborating** towards success



The Social Network Analysis (SNA) Project

Business Needs

Our client is a *global* company if the **energy sector**, active in plenty of countries.

Specifically, we have been working with the dept. taking care of global digital communication and strategy.

Detect **influencers**, users having the *greatest impact* on opinion *amplification*

Find the most **peculiar topics**, i.e. those polarizing social opinion (on **Twitter**) on the company

Must be designed to play a fundamental role in **supervising** the overall content **strategy** definition.

Provided analyses should **deliver** relevant business **insights**, such as: detecting **unexpected** influential accounts as well as **communities** representing opportunities for ad-hoc **communication** activities, and assessing **trending** topics' compliance with Company's **vision.**

What we Did

SNA: a Dashboard for Social Reputation Analysis

Based on AWS

We leveraged SNA techniques and methodologies - with **business needs** in mind – to design and develop an **interactive dashboard for social analysis on Twitter.**

A *tailor-made tool* aimed at devising **interactive**, **exploratory analyses** on **Twitter data**, so as to get **insight** on how users **interact** when publishing **opinions** about the Company

A fundamental **support** for decision makers to address peculiar **business** issues **hardly uncoverable** with **traditional** social monitoring **tools** and **KPIs.**

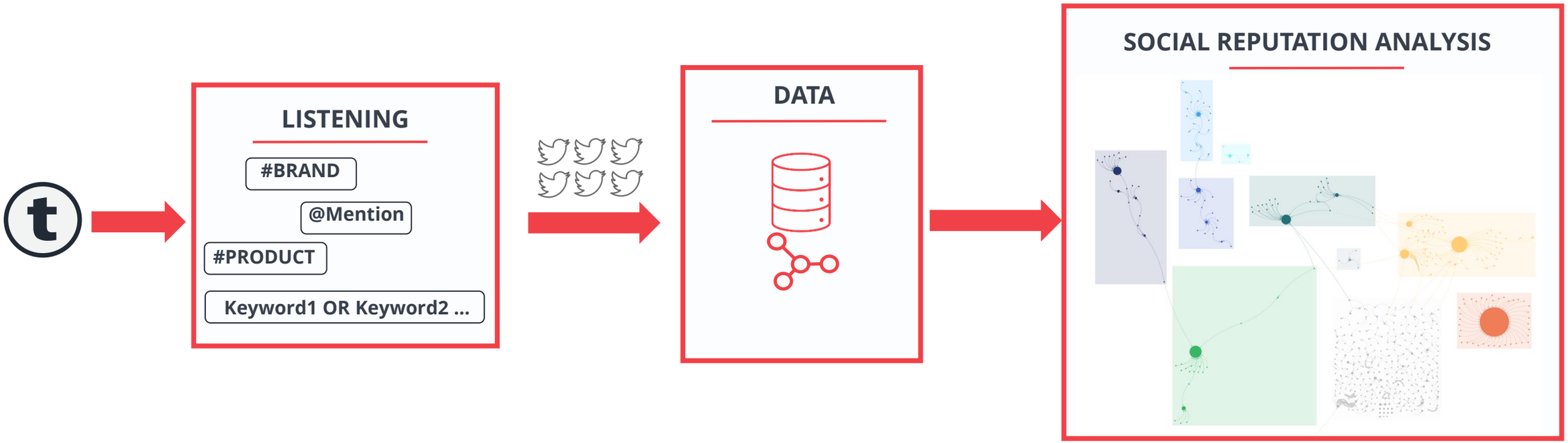
Our approach in a “Nutshell”: Social Reputation Analysis

Social Reputation Analysis

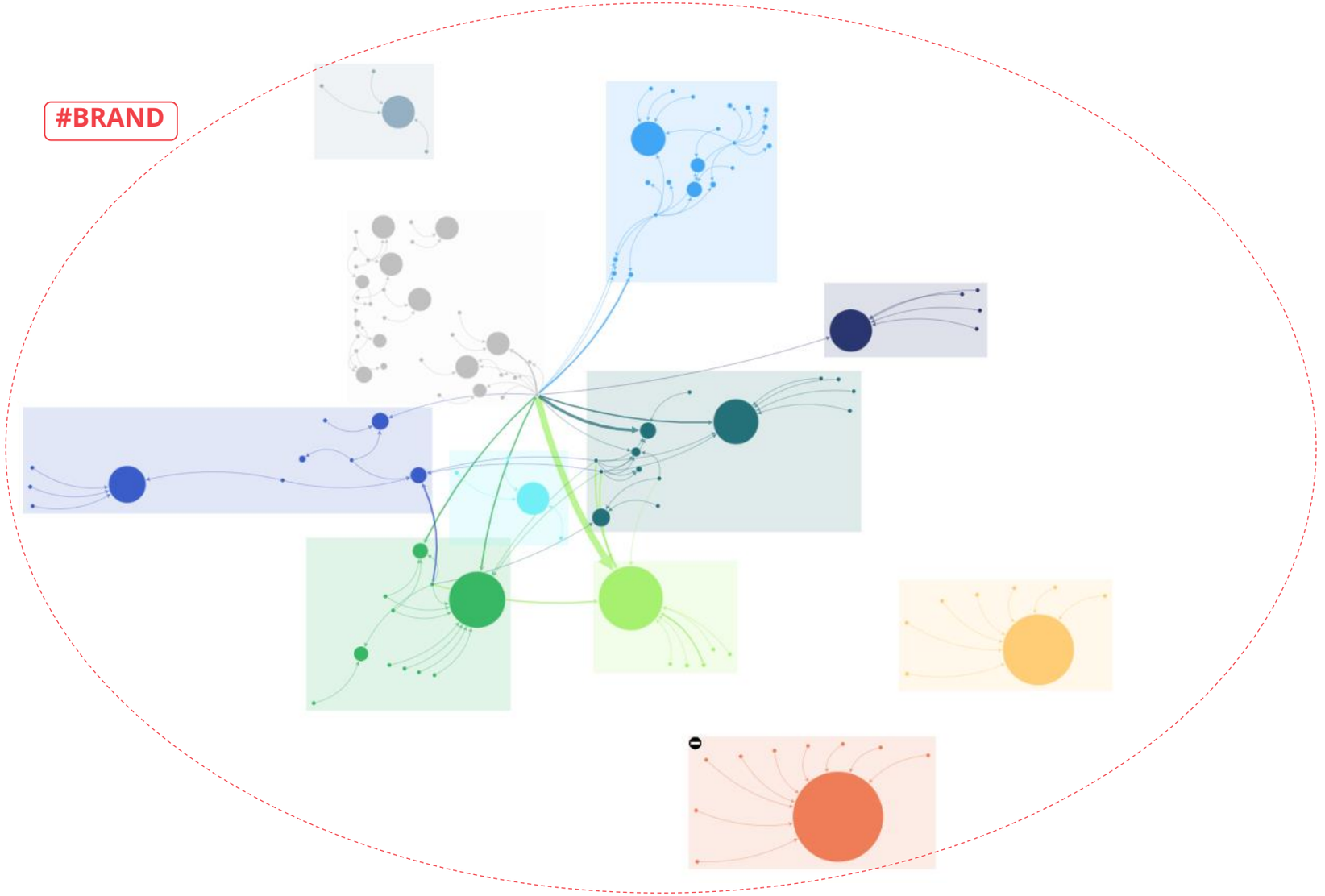
Social Reputation Analysis relies on data collected from social sources (e.g. Twitter)

Leveraging methods from **graph theory** and **social network analysis**, aims at detecting **brand influencer** and **relevant "topics"** related to Company's Brand

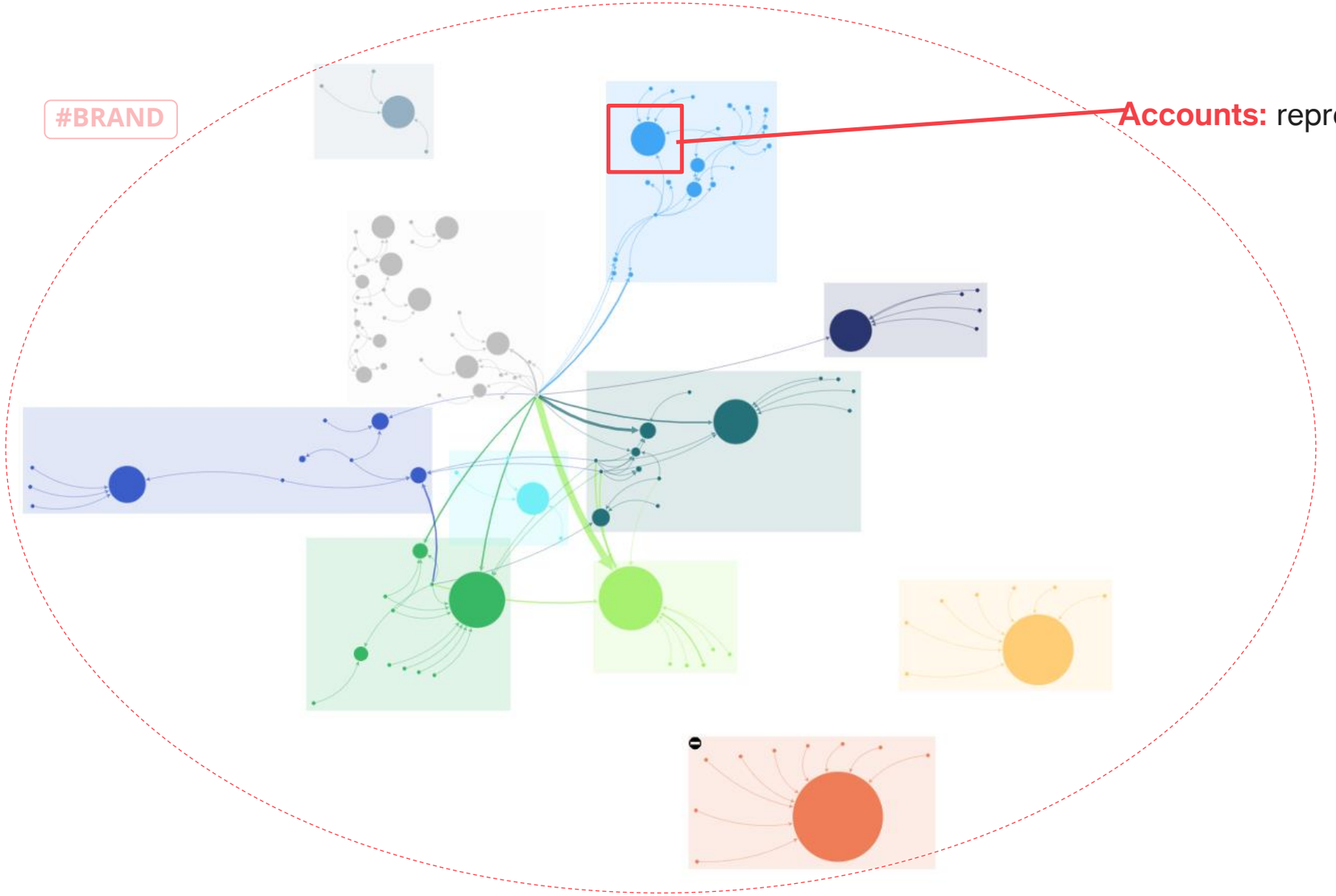
Social Reputation Analysis: Pipeline



Social Reputation Analysis “in action”

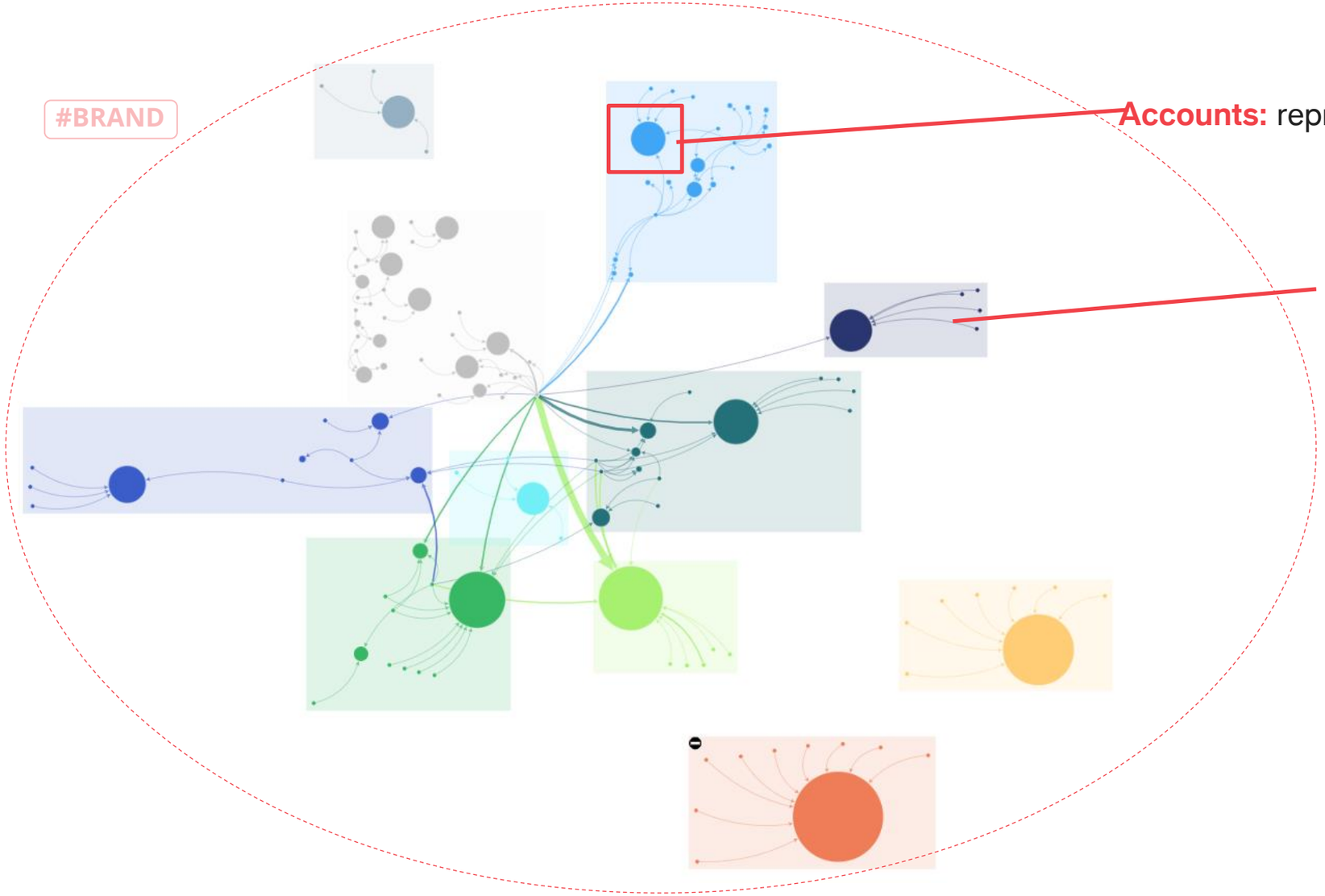


Social Reputation Analysis “in action”



Accounts: represented as nodes of the social network

Social Reputation Analysis “in action”

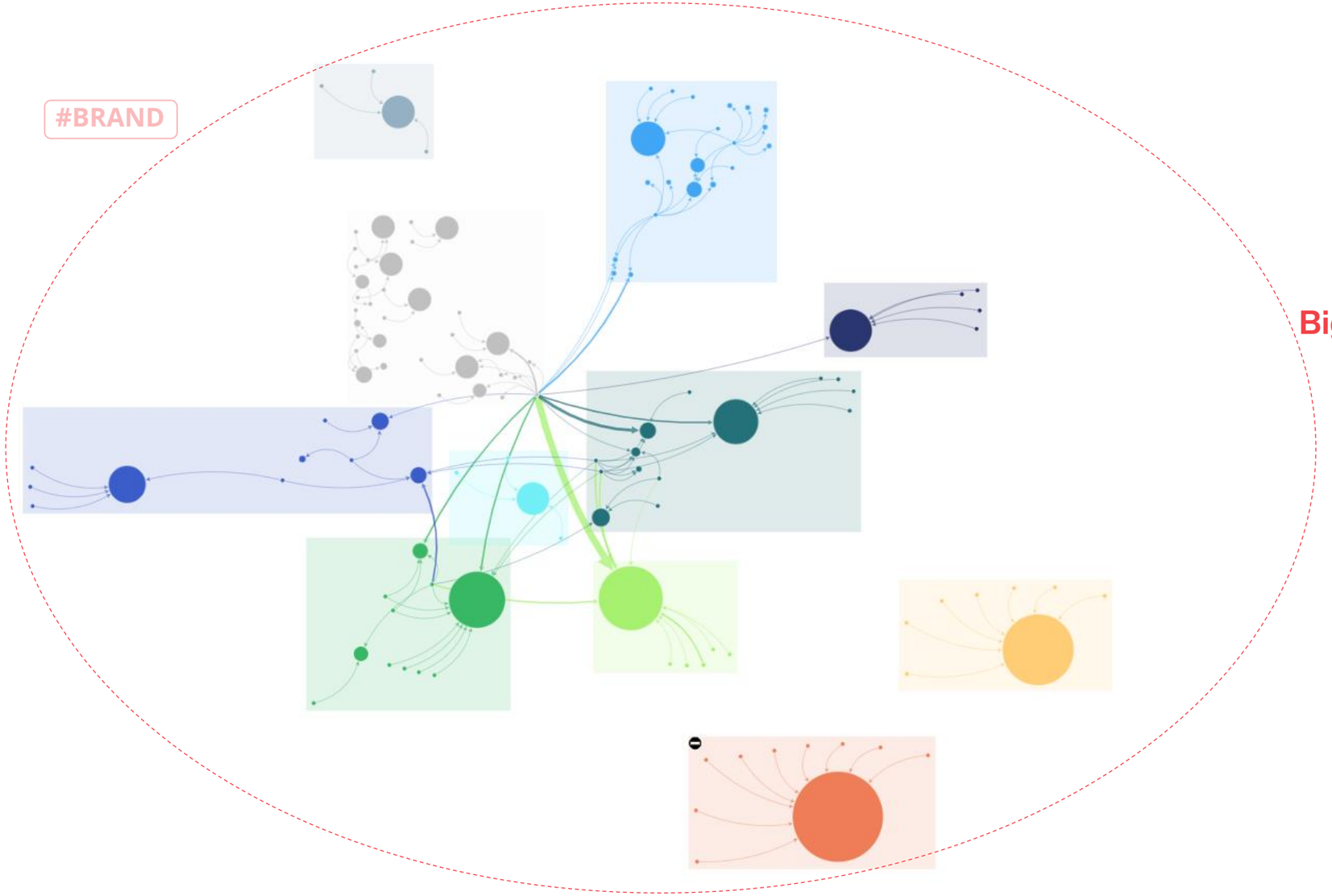


Accounts: represented as nodes of the social network

Retweet events : links between nodes/accounts

Links represent **amplification phenomena** among accounts

Social Reputation Analysis “in action”

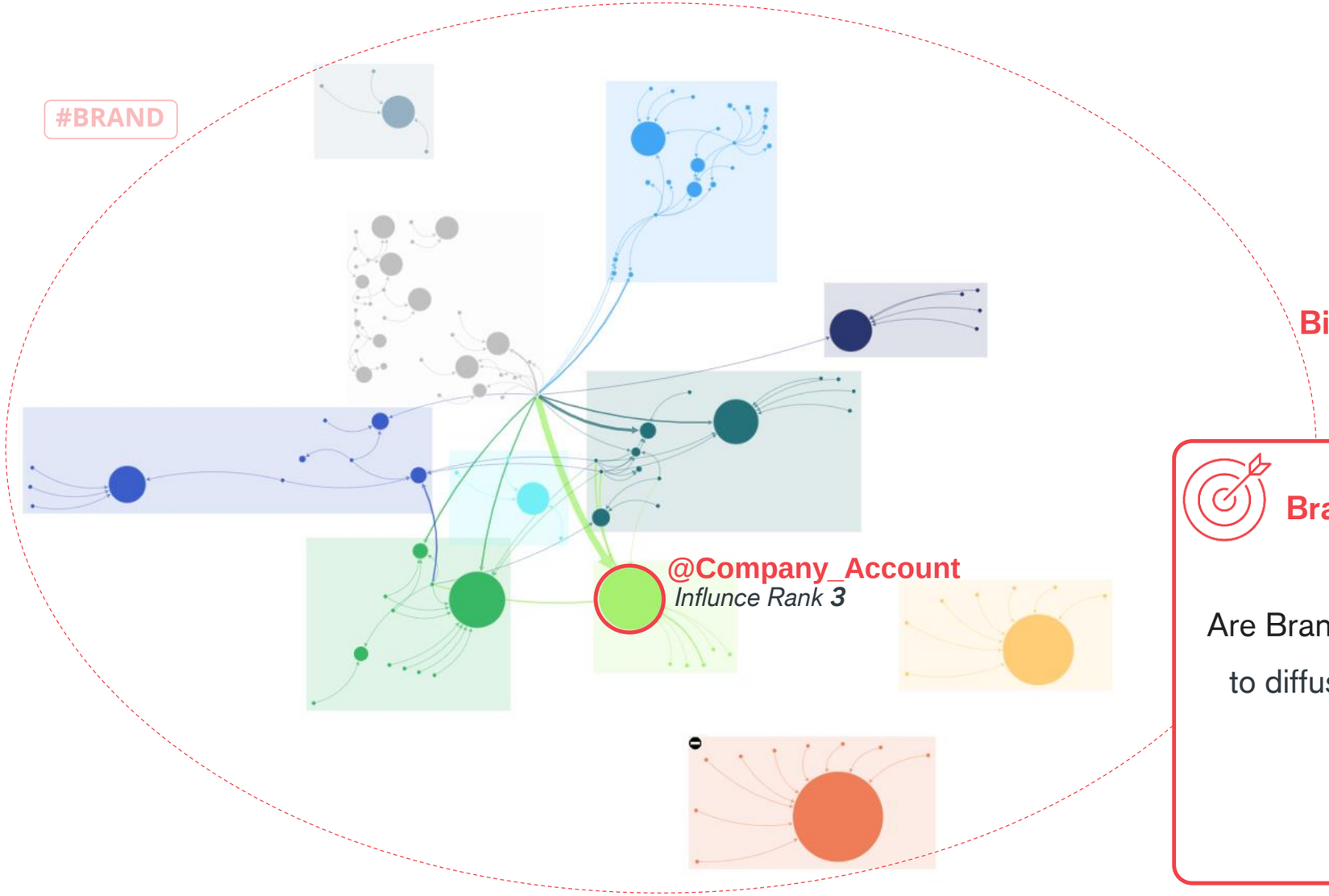


Artificial Intelligence

Centrality algorithms to **detect brand influencers**

Bigger nodes denote **“influential”** accounts

Social Reputation Analysis “in action”



Artificial Intelligence

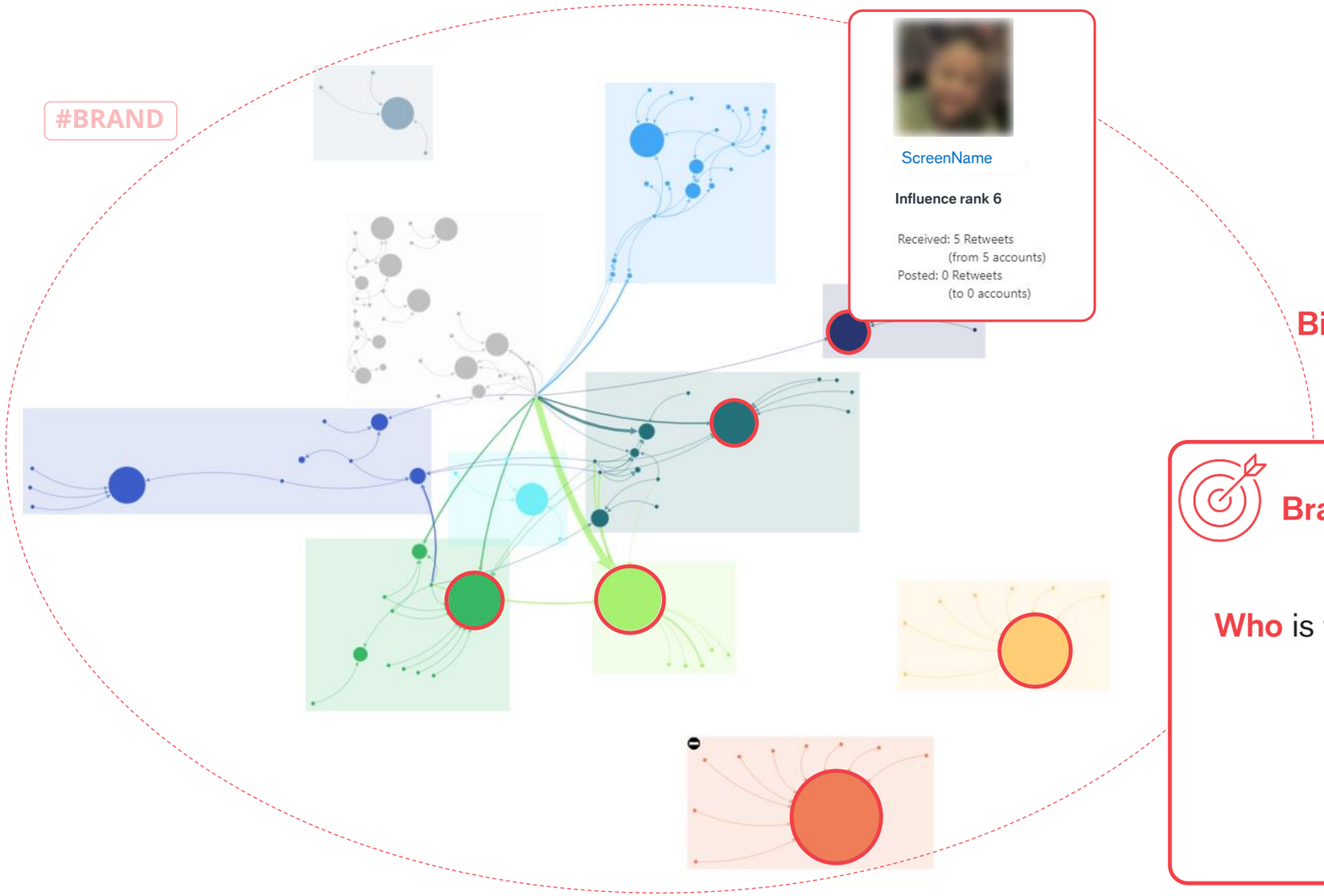
Centrality algorithms to **detect brand influencers**

Bigger nodes denote **“influential”** accounts

Brand Identity

Are Brand’s official accounts effective (**influent!**) to diffuse messages within the social network itself?

Social Reputation Analysis “in action”



Artificial Intelligence

Centrality algorithms to **detect brand influencers**

Bigger nodes denote **“influential”** accounts

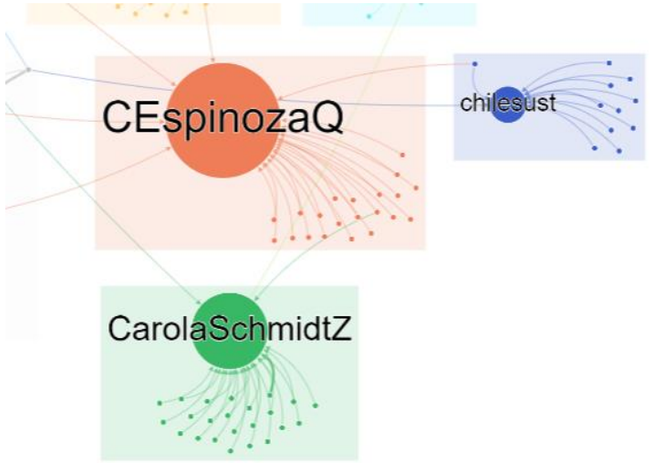
Brand Image

Who is talking about the Brand and who are the **peculiar influencers**?

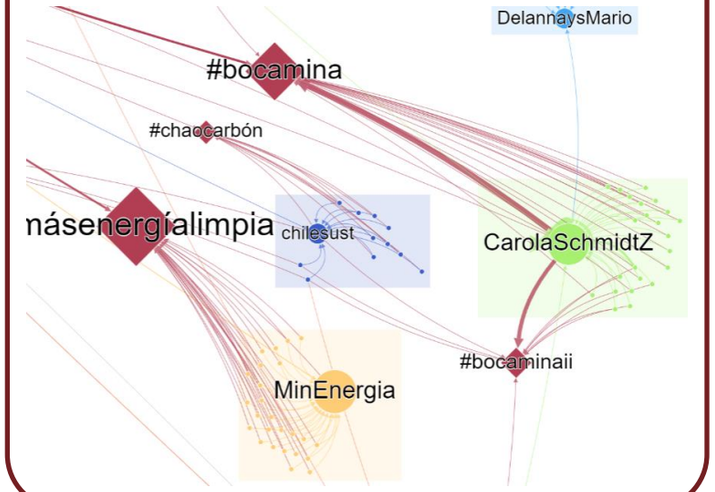
Example of SNA Analyses and Insights

Several **flavors** of self-service analyses targeting **different business needs**

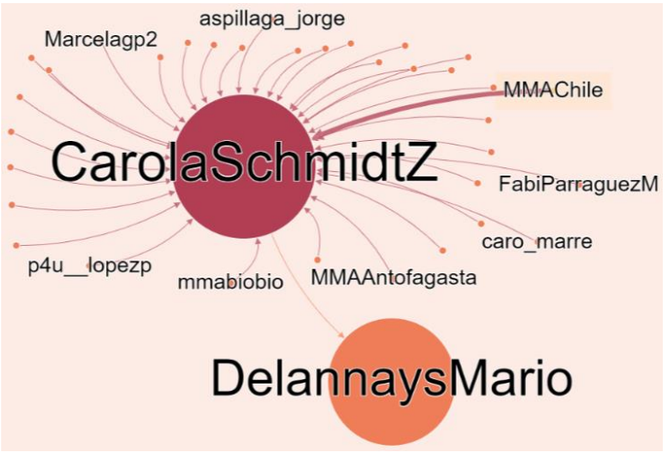
Influencers and Communities



Topics



EgoNets



Wanna Watch a Demo?

The Application Architecture: a Fully-Managed Solution on AWS

Project Roadmap

Stage 1

Devise an MVP (Minimum Viable Product) on a single scenario (social ‘‘monitoring domain’’)

Stage 2

Extend the app to enclose any relevant ‘‘monitoring domain’’

Productionalization

Stage 1

1. Data analysis and collection

2. Data ingestion/ETL

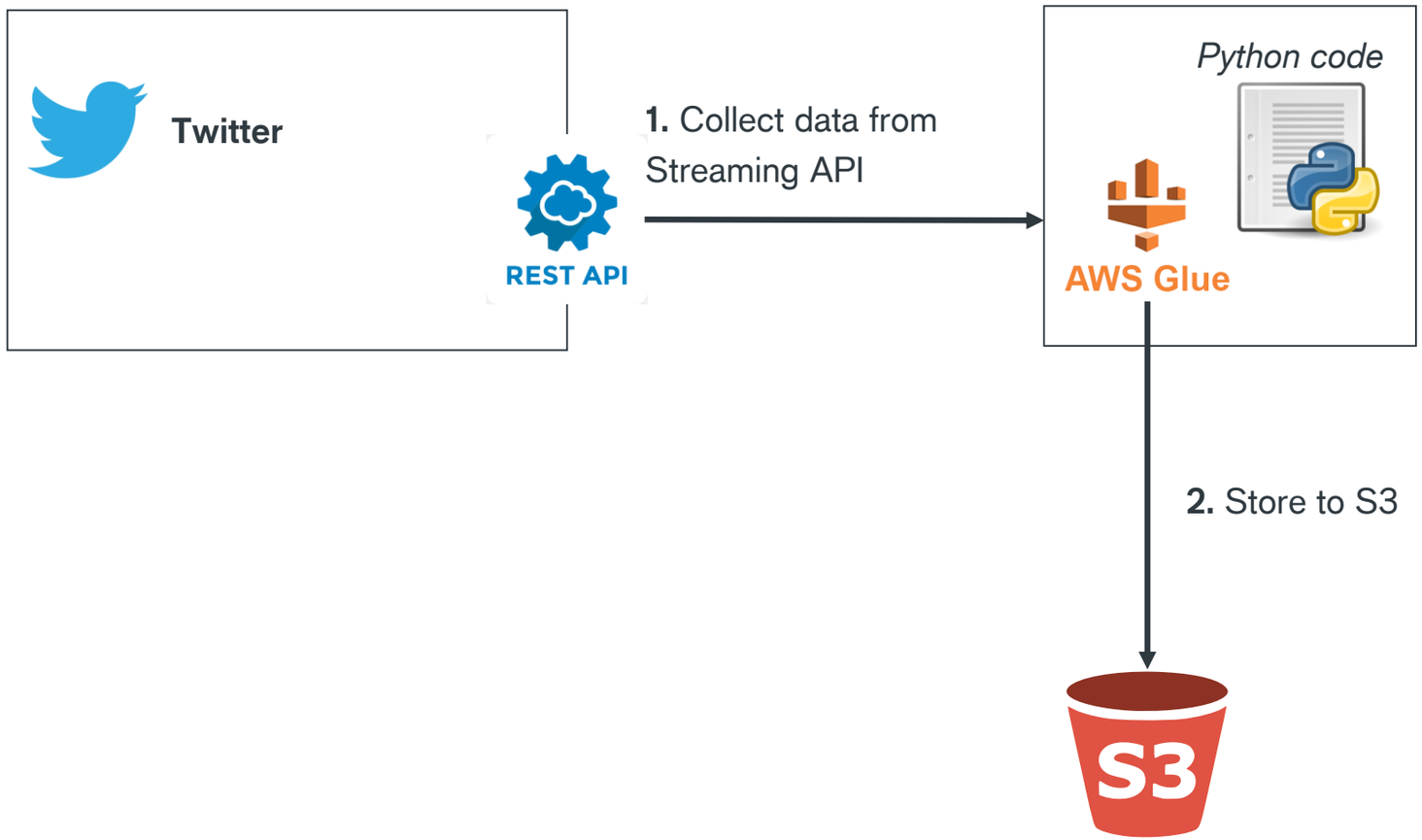
3. Data visualization (Front-end development)

Stage 2

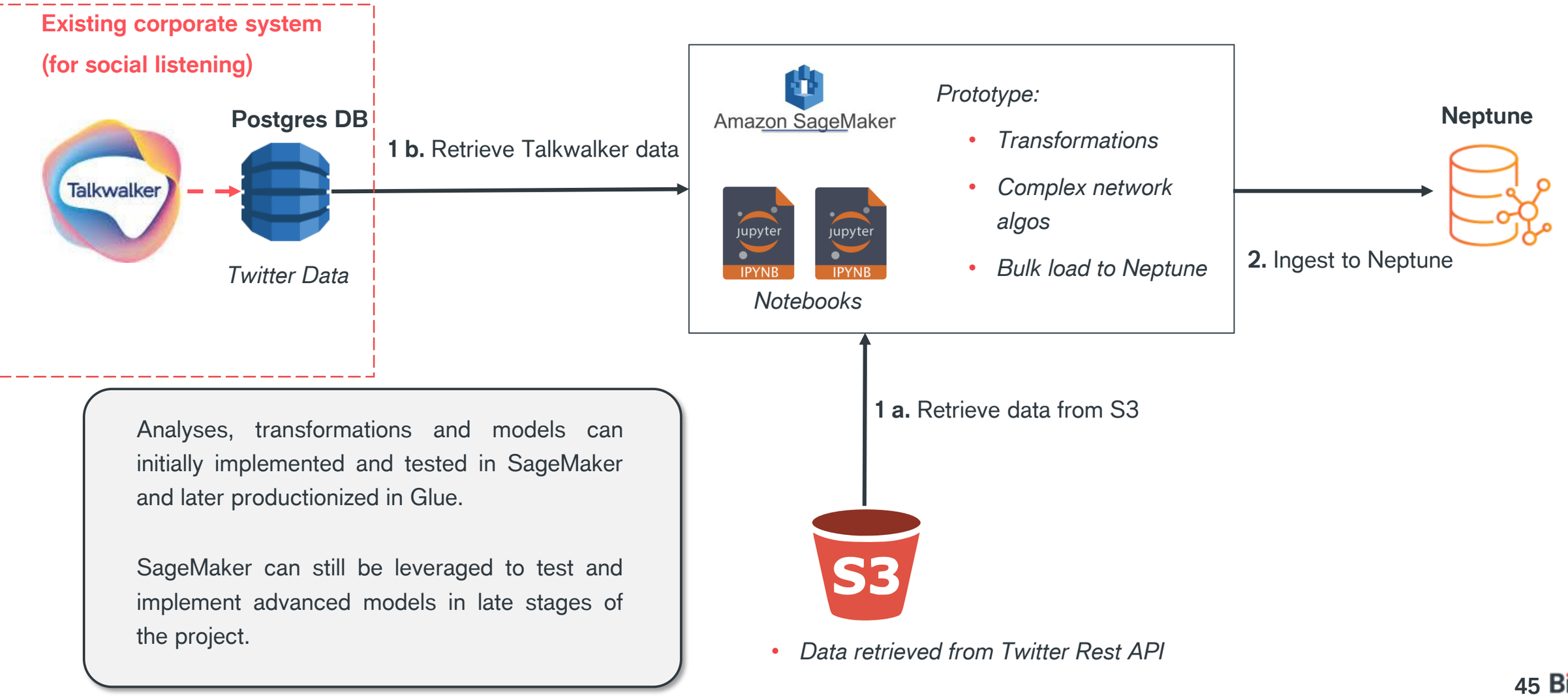
1. Generalize to a wider listening frame (more «key topics» ingested and analyzable from the dashboard)
2. Authentication and Authorization
3. Productionize
4. Integration with other systems

The Proposed Architecture

Collect Data from External API



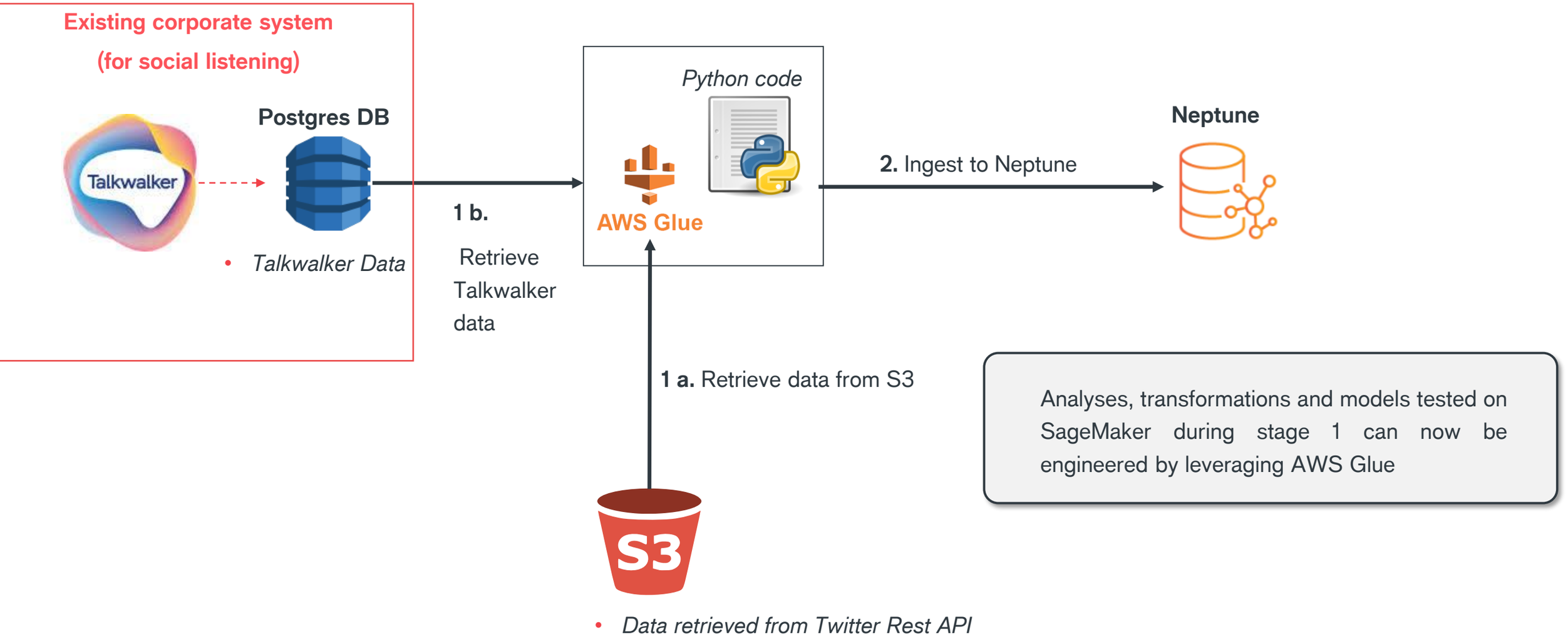
Data Ingestion – Prototype Transformation and Algorithms (Stage 1)



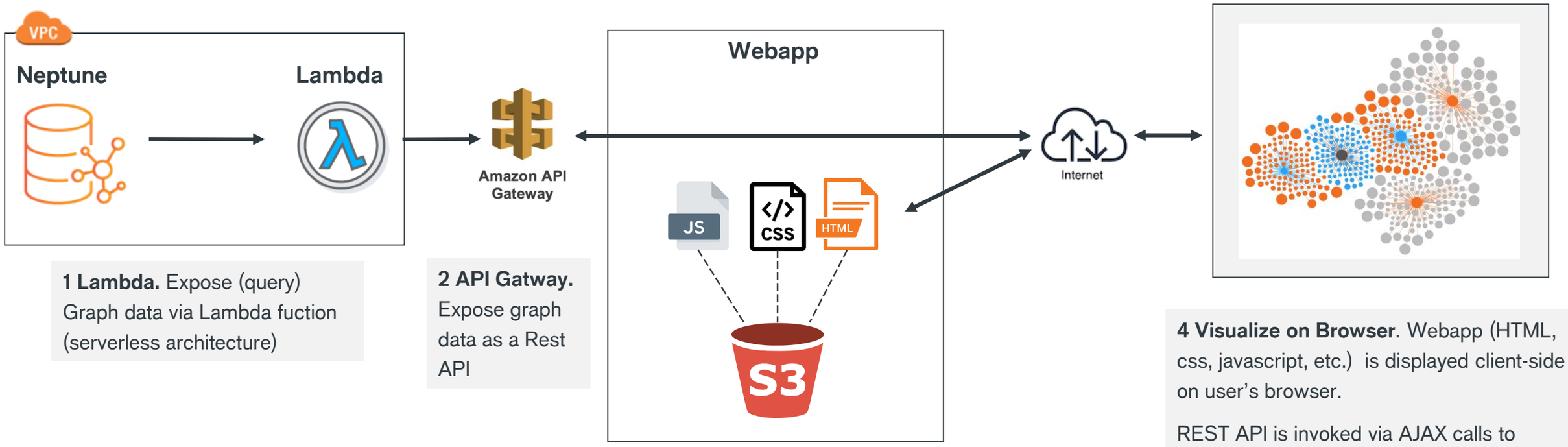
Analyses, transformations and models can initially implemented and tested in SageMaker and later productionized in Glue.

SageMaker can still be leveraged to test and implement advanced models in late stages of the project.

Data Ingestion – Productionize (Stage 2)



Front-end (Visualization) Application Architecture



1 Lambda. Expose (query) Graph data via Lambda fuction (serverless architecture)

2 API Gatway. Expose graph data as a Rest API

3 Webapp. Actual Web application is hosted in a S3 bucket, containing all the necessary static resources:

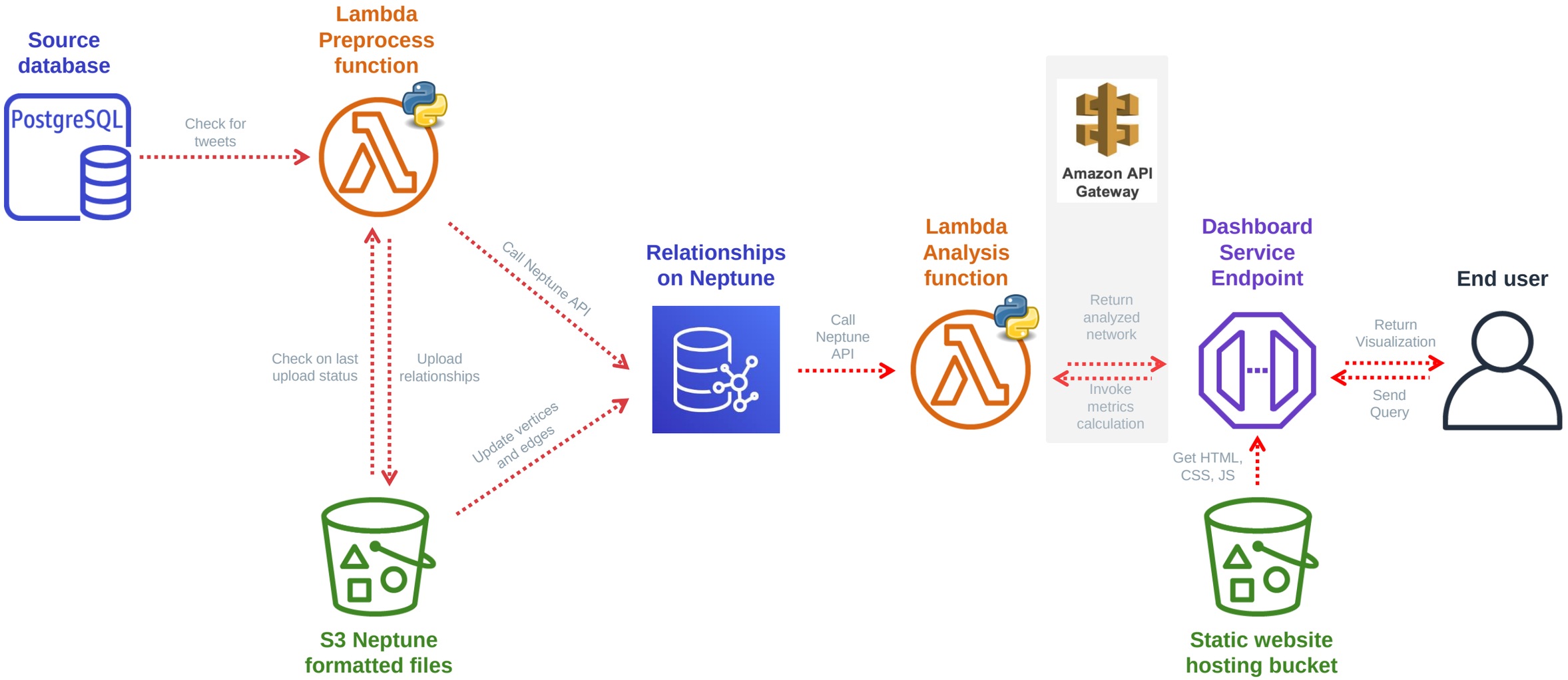
- HTML;
- CSS;
- Javascript libraries.

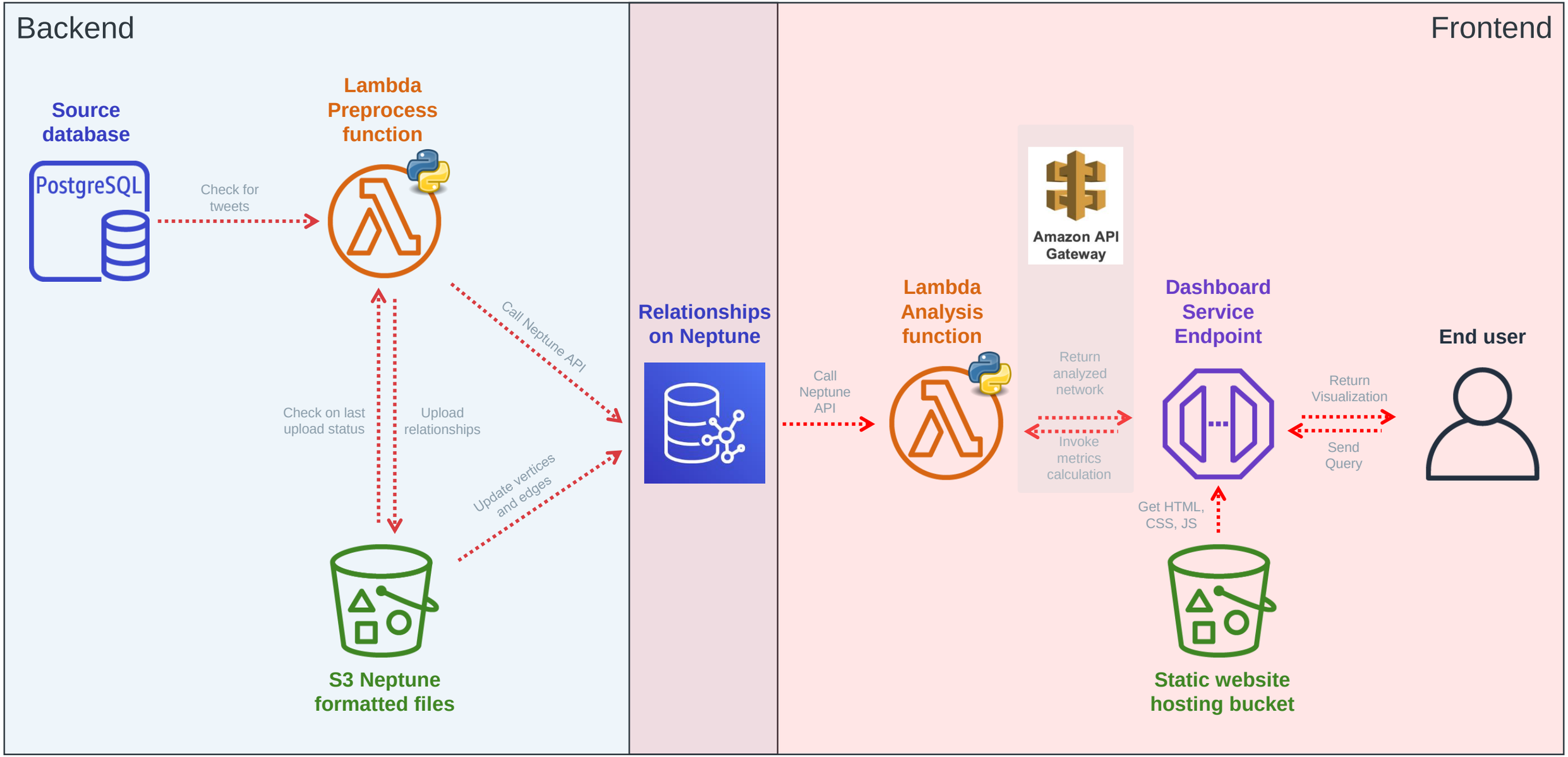
S3 bucket will accessible via a public URL

4 Visualize on Browser. Webapp (HTML, css, javascript, etc.) is displayed client-side on user's browser.

REST API is invoked via AJAX calls to retrieve data needed to render Twitter-based social networks

The Eventual Architecture


Source database
PostgreSQL
Check for tweets
Lambda Preprocess function
Call Neptune API
Check on last upload status
Upload relationships
Update vertices and edges
S3 Neptune formatted files
Relationships on Neptune
Call Neptune API
Lambda Analysis function
Amazon API Gateway
Return analyzed network
Invoke metrics calculation
Dashboard Service Endpoint
Return Visualization
Send Query
End user
Get HTML, CSS, JS
Static website hosting bucket


Backend
Frontend
Source database
PostgreSQL
Check for tweets
Lambda Preprocess function
Call Neptune API
Check on last upload status
Upload relationships
Update vertices and edges
S3 Neptune formatted files
Relationships on Neptune
Call Neptune API
Lambda Analysis function
Amazon API Gateway
Return analyzed network
Invoke metrics calculation
Dashboard Service Endpoint
Return Visualization
Send Query
End user
Get HTML, CSS, JS
Static website hosting bucket

Fully-Managed Architectural Components

You should have notice by now, that one of the **requirements** was to use **AWS** as the cloud environment for implementing the solution!

In general we are always in favor to adopt of fully-managed components. In addition, this time, such a choice was made not because we believe that customer-managed components are worse but simply because **we had a challenging project deadline.**

Thus, we had to choose components

- **Easy** to be interconneted with one another in AWS
- With no need to **explicitly manage** the underlying computational resources

Despite this we had to convince our client (especially their IT Division) on some of our picks

From “Proposed” to “Actual”: the “Selection” Process

Software Selection Process

Our client had a strict set of guidelines and **software tool map.**

Despite how good and effecient your application architecture might be, **you need to convince them**...

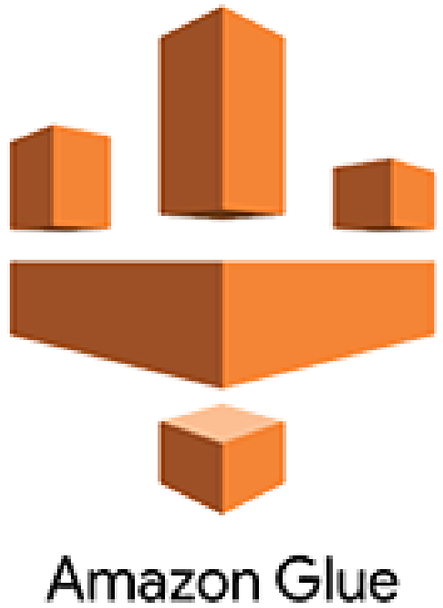
We underwent this process mainly in **three areas**

1. ETL
2. Storage
3. Computation engine

ETL

ETL: AWS Glue

Our pick was AWS Glue, the **fully-managed ETL solution by AWS**



Fully-Managed

- **Severless**, no need to manage any computation infrastructure
- Full **horizontal scalability** and distribution (basically, it is Apache Spark!)

Integration with Storage

- **Seamless integration** with AWS native storage (e.g. S3)

Sagemaker

- Easy to work with **prototyping ETL scripts** in Sagemaker (Jupyter notebooks)

Security

- Access is managed straight by AWS itself, i.e. by **IAM** (AWS's Identity and Access Management)

Cost

- Perhaps not the most important factor, but it is ‘‘pay-per-use’’ based on the resource dynamically allocated **(no licensing cost)**

AWS Glue: Contenders

They proposed us to use one between **Talend** and **Informatica.**

Features

- **Proprietary,** usually **on-prem**
- Typically a pricing model based on **license**

Downsides

- A more difficult integration with other selected AWS components
- We did not have so much experience with those
- Maintenance of the ETL pipelines and scaling on increasing volume of data
- In one word: **missing project deadline**
- Access control? Guys, no idea...

Benefits

- Already available and (almost) ready to be used

AWS Glue: Long Story Short

Well guess you guys succeeded...

AWS Glue: Long Story Short

NOT AT ALL

Our client's IT Division did not agree to introduce yet another ETL tool in their ''ecosystem''.

Since we just could not ''afford'' to resort to Informatica/Talend, we propose to adopt **AWS Lambda** (more on that later) for scheduled ETL jobs on source Twitter data.

We had a few issues in setting up the right resource quota, but it worked darn well in the end.

Storage: AKA Graph Database

AWS Neptune: our pick

Our pick was AWS Neptune, the **fully-managed Graph-based DB solution by AWS**

Spoiler

We **succeeded** in convincing our client with respect to other solutions!

Amazon Neptune

Fully-Managed

- **Severless**, no need to manage any computation infrastructure (scaling to increasing data volumes and query time)
- *Automatic* management of software **updates** and **backups**
- *Automatic* management of **failover** (in short amount of time)

Security

- Access is managed straight by AWS itself, i.e. by **IAM** (AWS's Identity and Access Management)

Graph models

- Support also **Semantic Web** standards (RDF/RDFS SPARQL) not just property graphs

AWS Neptune: Contenders

They asked us to adopt one between **Neo4j** and **OrientDB**

Features

- Both available in the cloud (e.g. AWS marketplace)
- Now Neo4j available also as a managed-cloud service (SaaS): **Neo4j Aura**, though at the time of the project it had not been released yet

Downsides

Not a lot in this case

- A more difficult integration with other selected AWS components
- In one word: **missing project deadline**
- Security model to be integrated into AWS IAM

Benefits

- Already available and (almost) ready to be exploited

AWS Neptune: Yes...

...this time we succeeded!

AWS Neptune: but...

...we need to ‘‘convince’’ the IT Division!

That was done by organizing a **PoC** for them with the help of AWS professionals

The PoC consisted of a Neptune instance running for 6 months in the Client’s AWS subscription (on a dev env)

In the meanwhile, we were developing our project...

Computation layer

AWS Lambda

In order to productionize the complex-network alogorithms prototyped in Sagemaker, we decided to resort to AWS Lambda, the **serverless computing platform by AWS**.

No servers to manage
AWS Lambda automatically runs your code without requiring you to provision or manage infrastructure. Just write the code and upload it to Lambda either as a ZIP file or container image.

Continuous scaling
AWS Lambda automatically scales your application by running code in response to each event. Your code runs in parallel and processes each trigger individually, scaling precisely with the size of the workload, from a few requests per day, to hundreds of thousands per second.

Cost optimized with millisecond metering
With AWS Lambda, you only pay for the compute time you consume, so you're never paying for over-provisioned infrastructure. You are charged for *every millisecond your code executes* and the *number of times* your code is triggered. With Compute Savings Plan, you can additionally save up to 17%.

Consistent performance at any scale
With AWS Lambda, you can optimize your code execution time by choosing the right memory size for your function. You can also keep your functions initialized and hyper-ready to respond within double digit milliseconds by enabling Provisioned Concurrency.

This one was easy...

They were already adopting Lambda, though they had a clear inclination for containers!

But, again, we had **very strict deadlines**...

Example of Cost Calculation

Lambda Function - Include Free Tier
Lambda counts a request each time it starts executing in response to an event notification or invoke call, including test invokes from the console. You are charged for the total number of requests across all your functions. The price depends on the amount of memory you allocate to your function. The Lambda free tier includes 1M free requests per month and 400,000 GB-seconds of compute time per month.

Lambda Function - Without Free Tier
Lambda counts a request each time it starts executing in response to an event notification or invoke call, including test invokes from the console. You are charged for the total number of requests across all your functions. The price depends on the amount of memory you allocate to your function. The Lambda free tier discounts are excluded

Service settings Info

The calculations below exclude free tier discounts.

Number of requests
50000

Duration of each request (in ms)
Duration is calculated from the time your code begins executing until it returns or otherwise terminates.
120000

Amount of memory allocated
Enter the amount between 128 MB and 10 GB
8 GB

▼ Show calculations

50,000 requests x 120,000 ms x 0.001 ms to sec conversion factor = 6,000,000.00 total compute (seconds)
8 GB x 6,000,000.00 seconds = 48,000,000.00 total compute (GB-s)
48,000,000.00 GB-s x 0.0000166667 USD = 800.00 USD (monthly compute charges)
50,000 requests x 0.0000002 USD = 0.01 USD (monthly request charges)
800.00 USD + 0.01 USD = 800.01 USD
Lambda costs - Without Free Tier (monthly): 800.01 USD

Lambda Function - Include Free Tier
Lambda counts a request each time it starts executing in response to an event notification or invoke call, including test invokes from the console. You are charged for the total number of requests across all your functions. The price depends on the amount of memory you allocate to your function. The Lambda free tier includes 1M free requests per month and 400,000 GB-seconds of compute time per month.

Lambda Function - Without Free Tier
Lambda counts a request each time it starts executing in response to an event notification or invoke call, including test invokes from the console. You are charged for the total number of requests across all your functions. The price depends on the amount of memory you allocate to your function. The Lambda free tier discounts are excluded

Service settings Info

Number of requests
50000

Duration of each request (in ms)
Duration is calculated from the time your code begins executing until it returns or otherwise terminates.
120000

Amount of memory allocated
Enter the amount between 128 MB and 10 GB
8 GB

▼ Show calculations

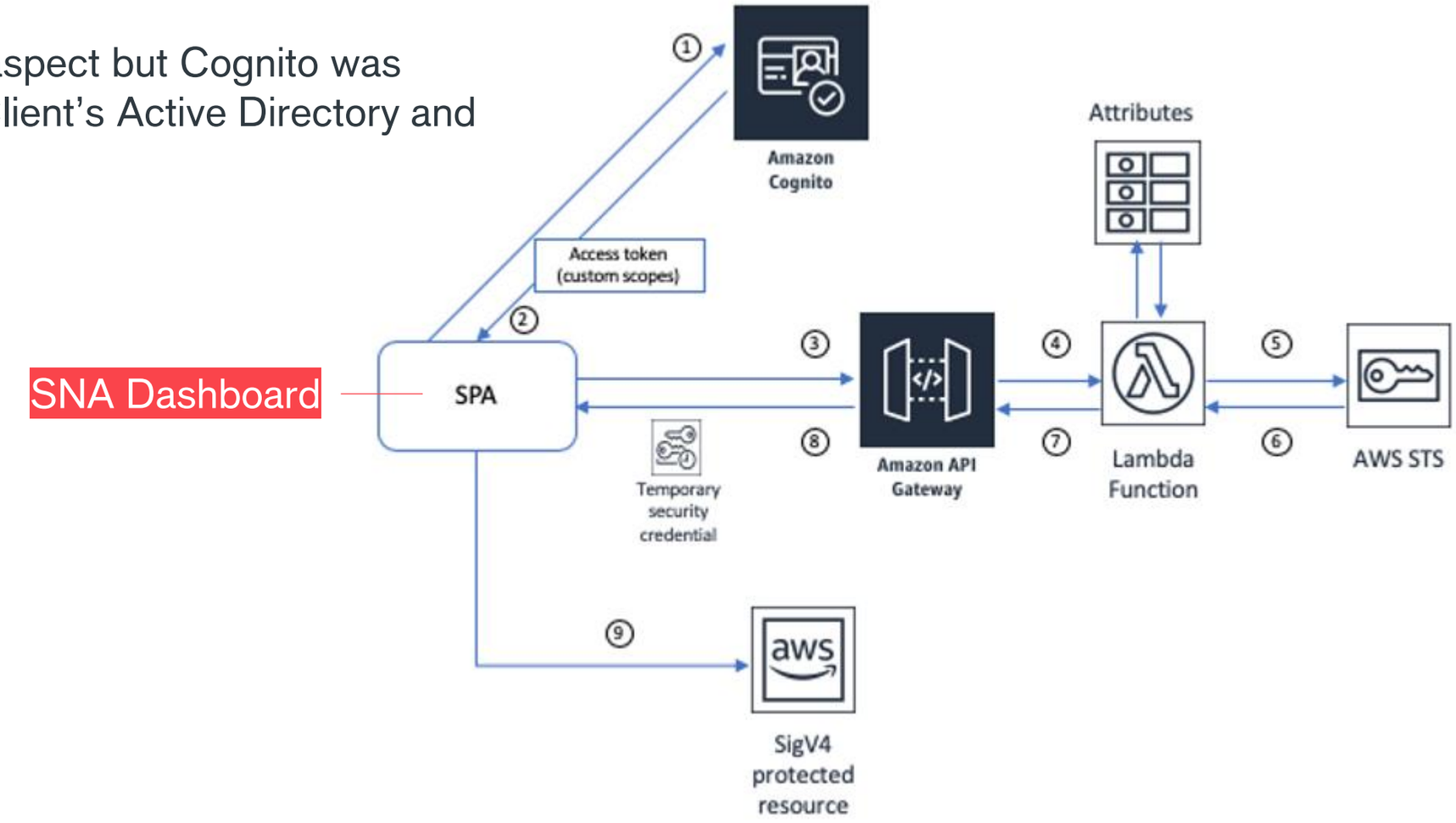
50,000 requests x 120,000 ms x 0.001 ms to sec conversion factor = 6,000,000.00 total compute (seconds)
8 GB x 6,000,000.00 seconds = 48,000,000.00 total compute (GB-s)
48,000,000.00 GB-s - 400000 free tier GB-s = 47,600,000.00 GB-s
Max (47600000.00 GB-s, 0) = 47,600,000.00 total billable GB-s
47,600,000.00 GB-s x 0.0000166667 USD = 793.33 USD (monthly compute charges)
50,000 requests - 1000000 free tier requests = -950,000 monthly billable requests
Max (-950000 monthly billable requests, 0) = 0.00 total monthly billable requests
Lambda costs - With Free Tier (monthly): 793.33 USD

What about “Security”

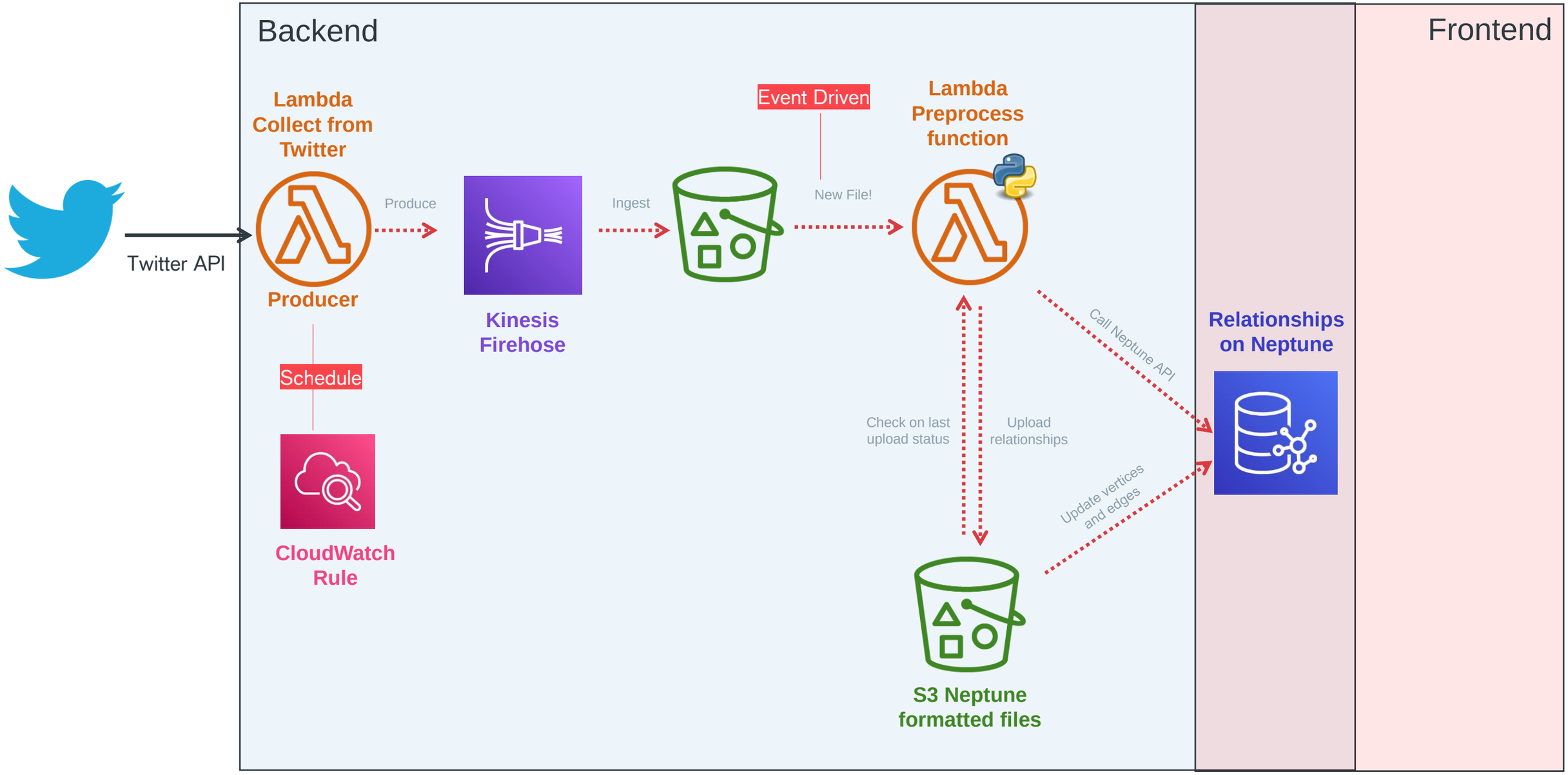
Cognito

AWS Cognito was used to implement the access and authorization layer.

We were not the owner of this aspect but Cognito was adopted via a Federation with Client's Active Directory and single sign-on (SSO) system.



Other Back-End “Alternatives”


Backend
Frontend
Lambda
Collect from
Twitter
Event Driven
Lambda
Preprocess
function
Twitter API
Produce
Ingest
New File!
Producer
Kinesis
Firehose
Schedule
Call Neptune API
Relationships
on Neptune
Check on last
upload status
Upload
relationships
Update vertices
and edges
CloudWatch
Rule
S3 Neptune
formatted files

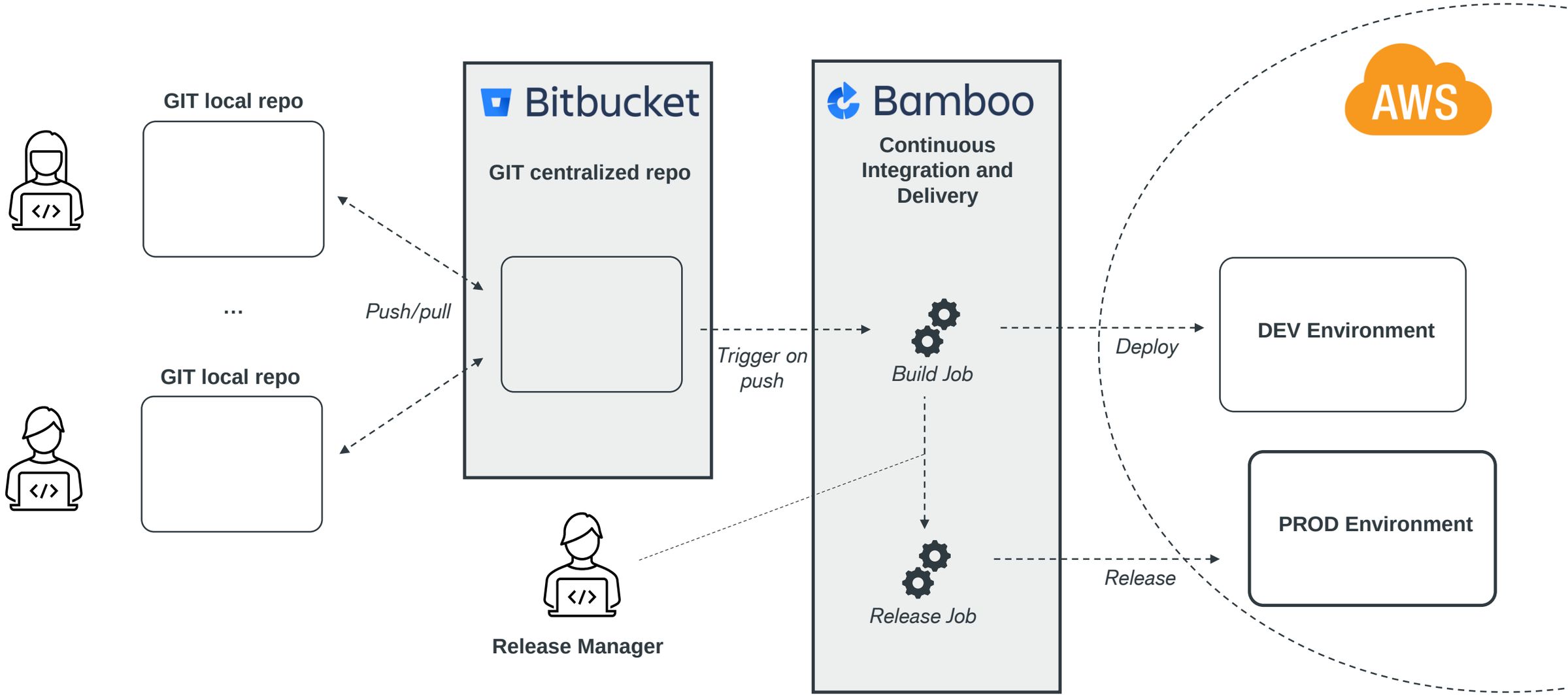
Kinesis Firehose

Fully Managed Service, no administration required!

- **Near Real Time** (60 seconds latency minimum for non full batches)
- Data Ingestion into Redshift / Amazon S3 / ElasticSearch / Splunk
- Automatic scaling
- Supports many data formats
- Data Conversions from CSV / JSON to Parquet / ORC (only for S3)
- Data Transformation through AWS Lambda (ex: CSV => JSON)
- Supports **compression** when target is Amazon S3 (GZIP, ZIP, and SNAPPY)
- Pay for the amount of data going through Firehose

Software Development and Delivery

Sofware Development and Continuous Integration



BitBang

Thank you!

Info@bitbang.com

Bologna
Via E. Mattei 102,
40138, Bologna, BO
+39 051 58 75 314

Milano
Via S. Marco 21,
20121, Milano, MI
+39 02 12 41 23 352

P.IVA 02329121202
Capitale Soc: € 60.000
R.E.A. BO 431512

Questions?