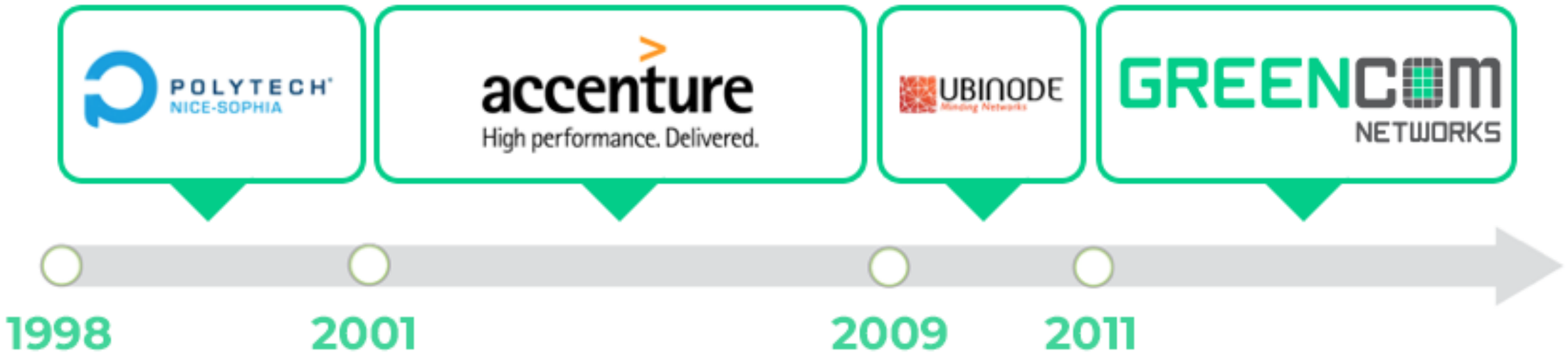
GreenCom Networks Platform Architecture

January 2019

Sebastien Alegret
CTO@ GreenCom Networks

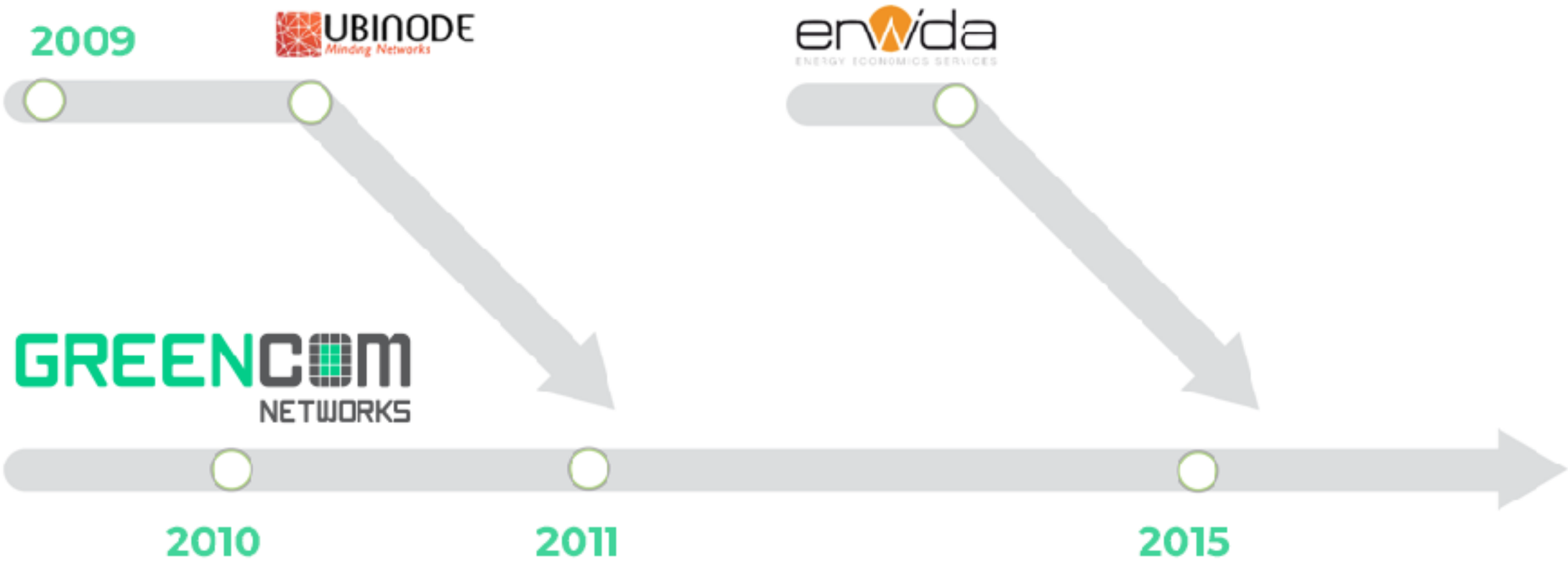
Mathias Chevalier
Infrastructure/Devops Team

A bit of (my) history

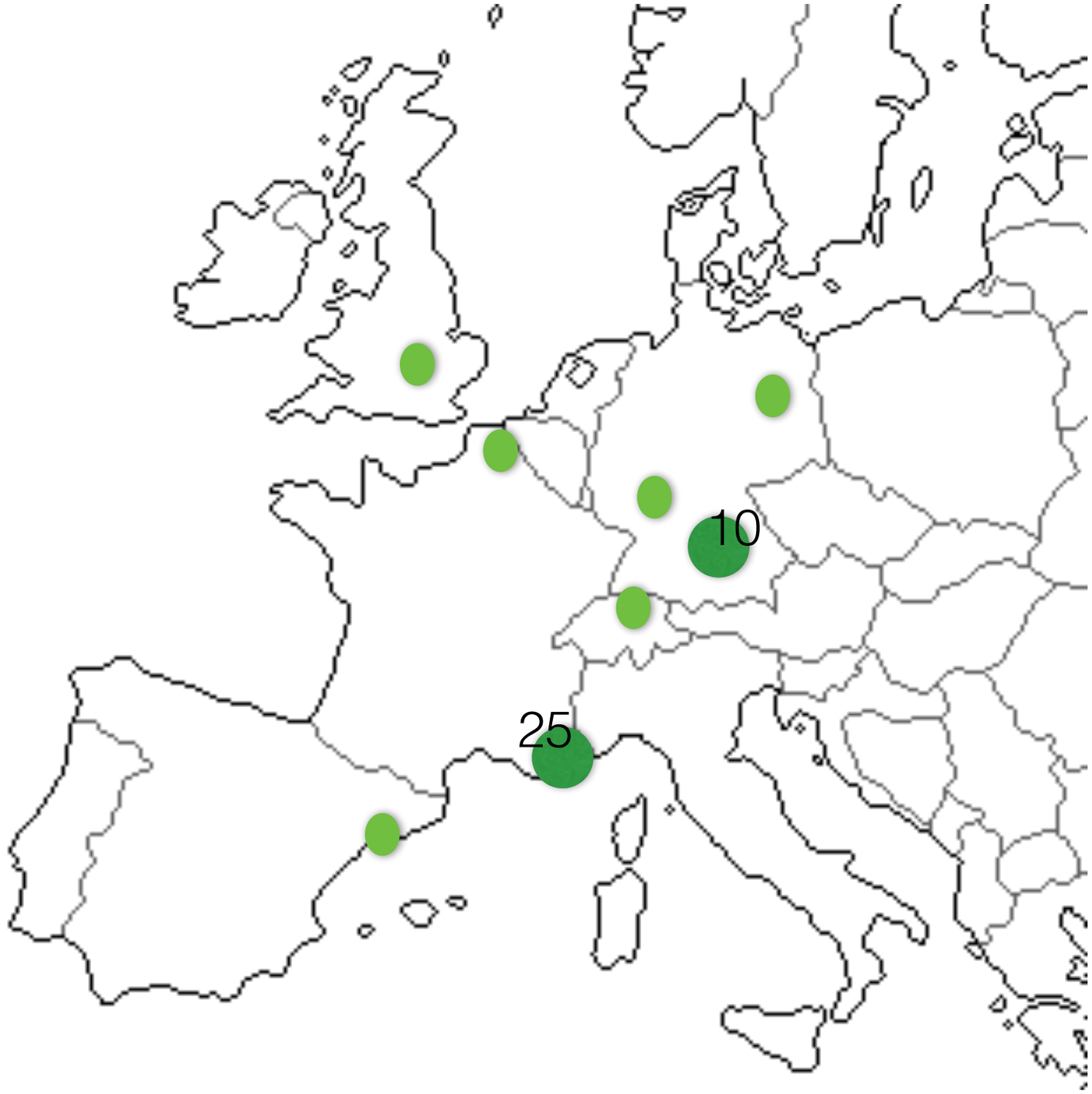


GreenCom Networks

A bit of GCN history



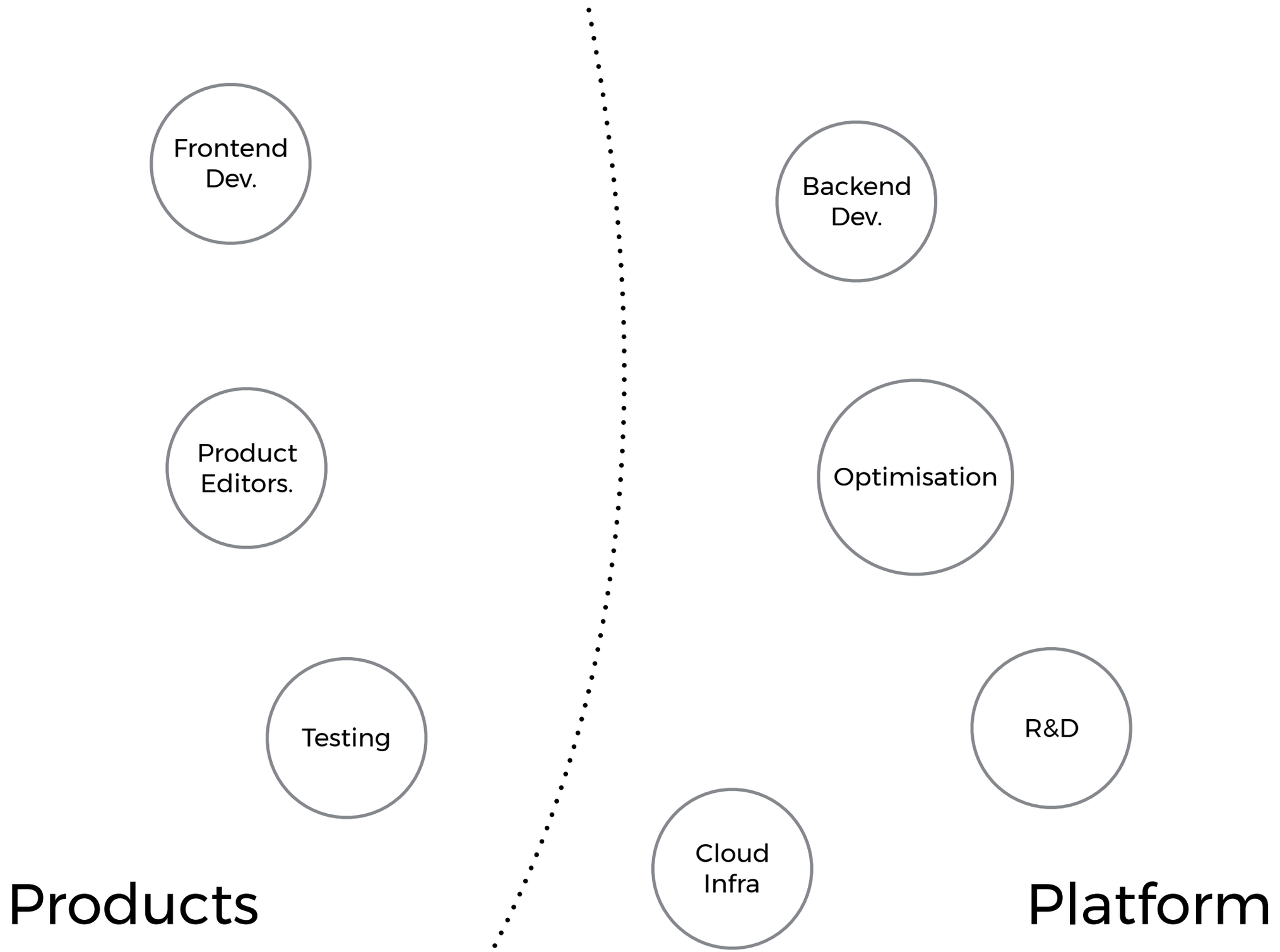
Our team



HQ in Munich
Offices in Munich / Sophia Antipolis
People in

- Berlin, Heidelberg (Germany)
- London
- Barcelona
- Winterthur (Switzerland)
- Lille
- Somewhere in a jet

Sophia Antipolis - Team Structure (originally)



Sophia Antipolis - Team Structure (today)

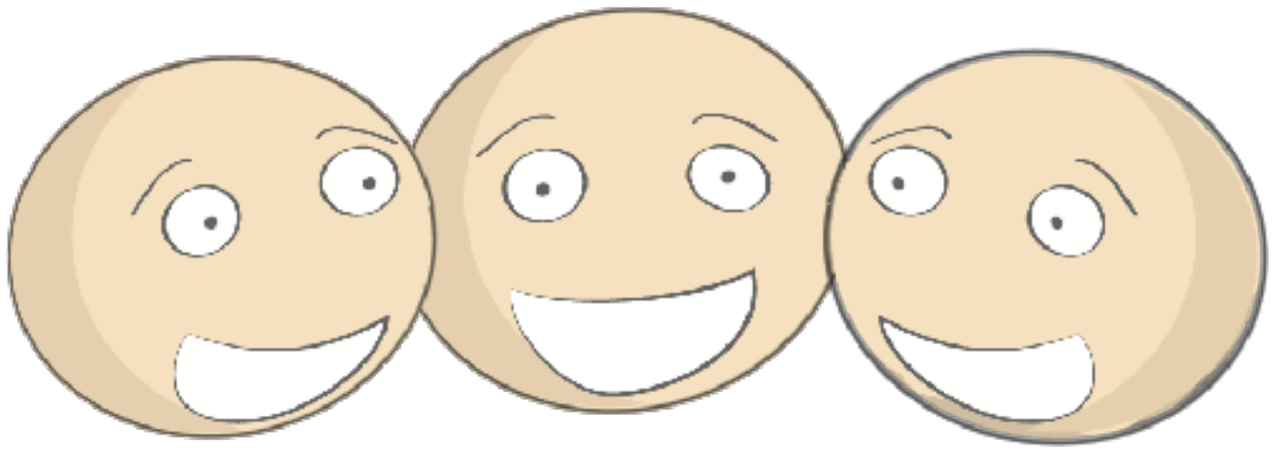
Things to do

People to do
them

Dev Methodology

WE THINK THAT

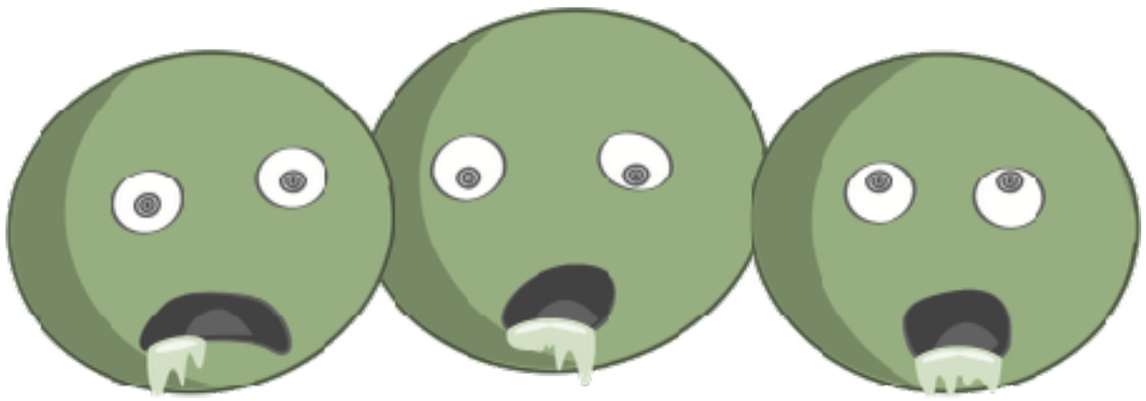
BLINDLY APPLYING METHODOLOGIES...



team of happy developers

... SLOWLY TURN THEM INTO ZOMBIES.

the walking devs

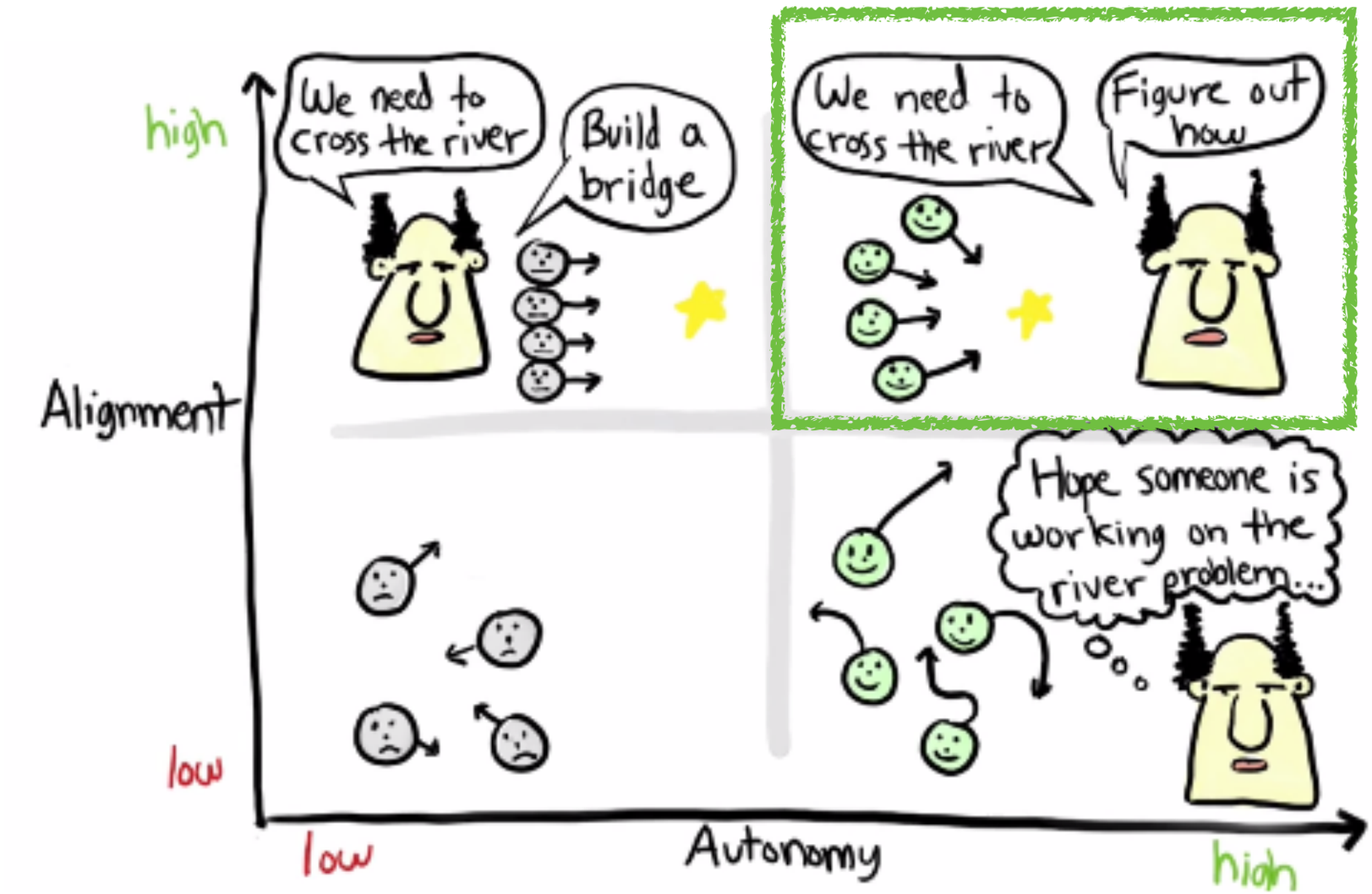


We’ve got **Things** to do and **People** to do them.

We do **Empower** the people (context, ownership).

We seek for **Autonomy** through **Alignement.**

Autonomy through **Alignement**



Spotify

Our world / Our business
Our products

Paradigm Shift

Old World

New World

Energy Supply

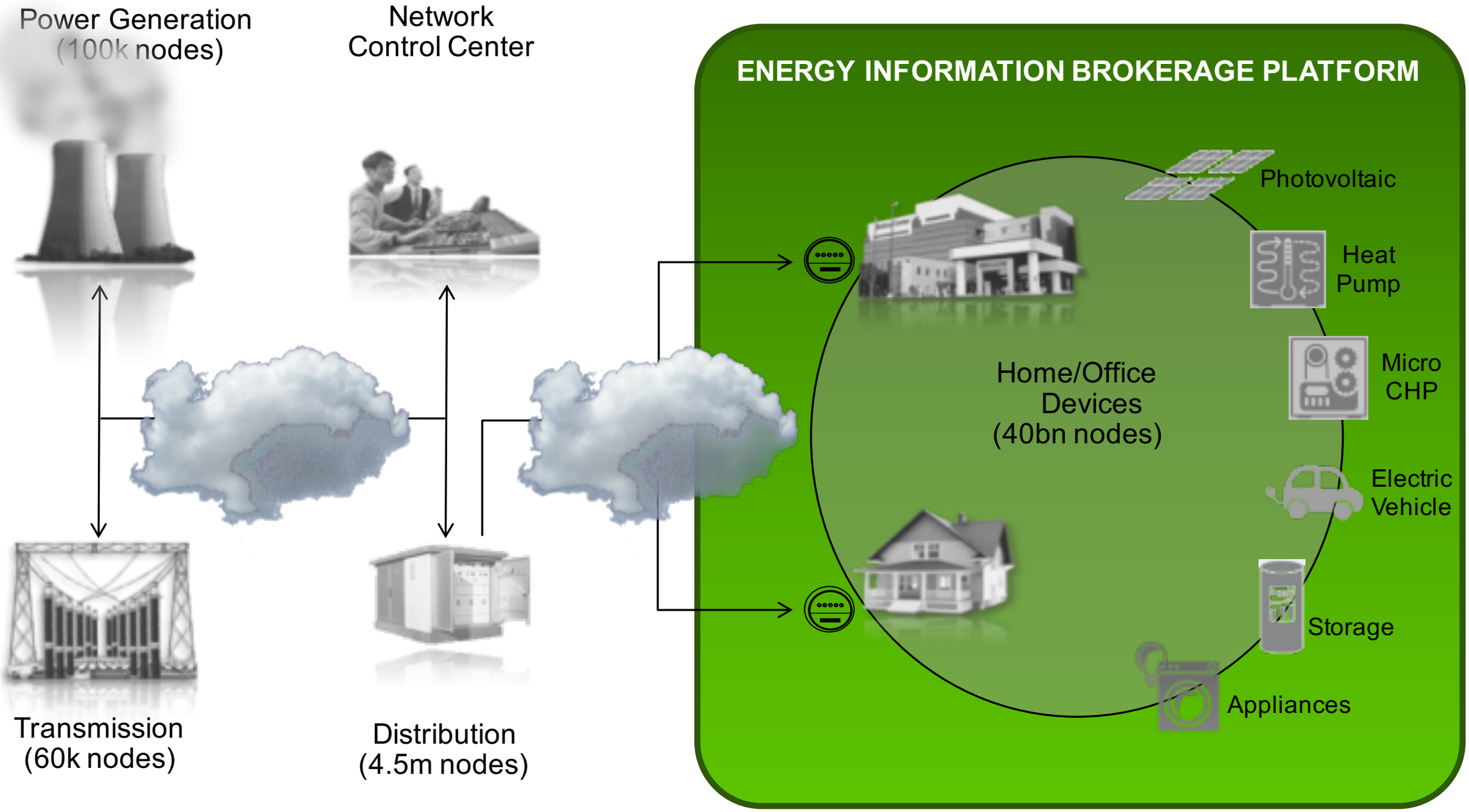
- Central convential power generation
- Central system management
- Consumers or End Points

Energy Management

- Central and distributed power generation based on conventional and renewable sources
- Central and distributed system management
- Customers and Prosumers

Paradigm Shift

Cleantech and Communication Technology leads to a Paradigm Shift in the resources being used

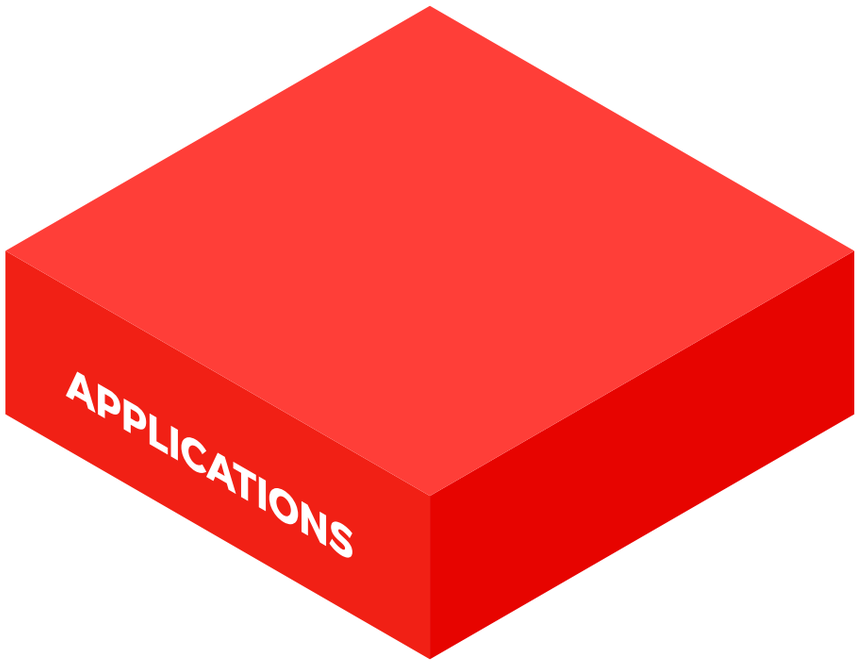


GreenCom Networks

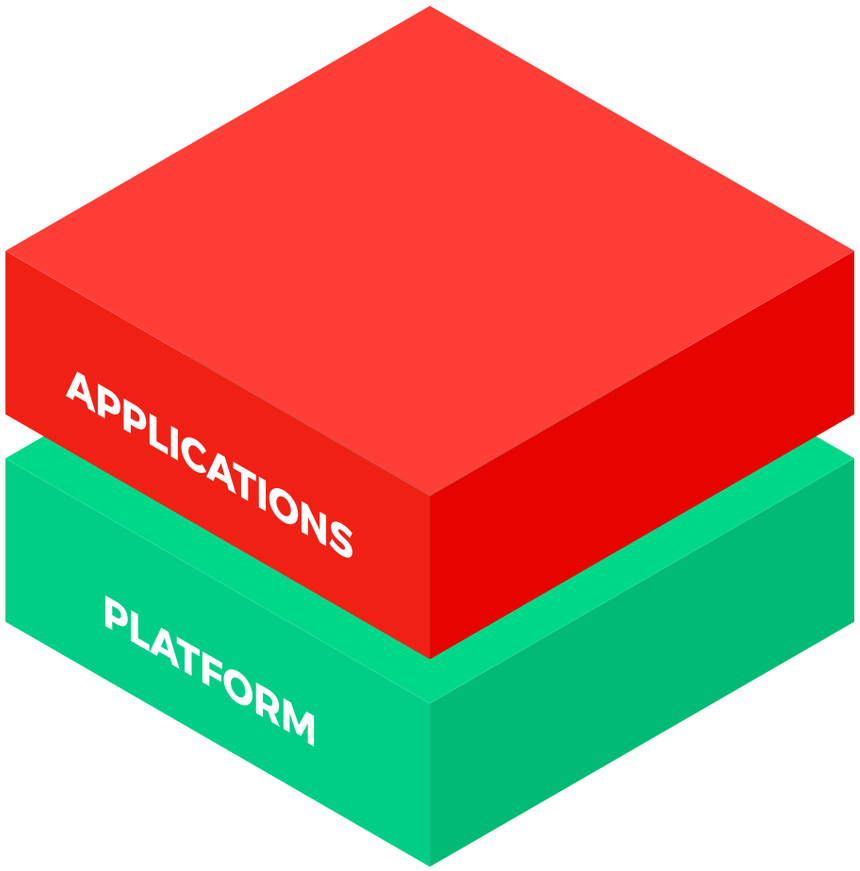
A software-as-a-service company that offers white-label solutions for the utility industry and energy service companies

GreenCom Networks

We offer energy related services based on our end-to-end IoT platform

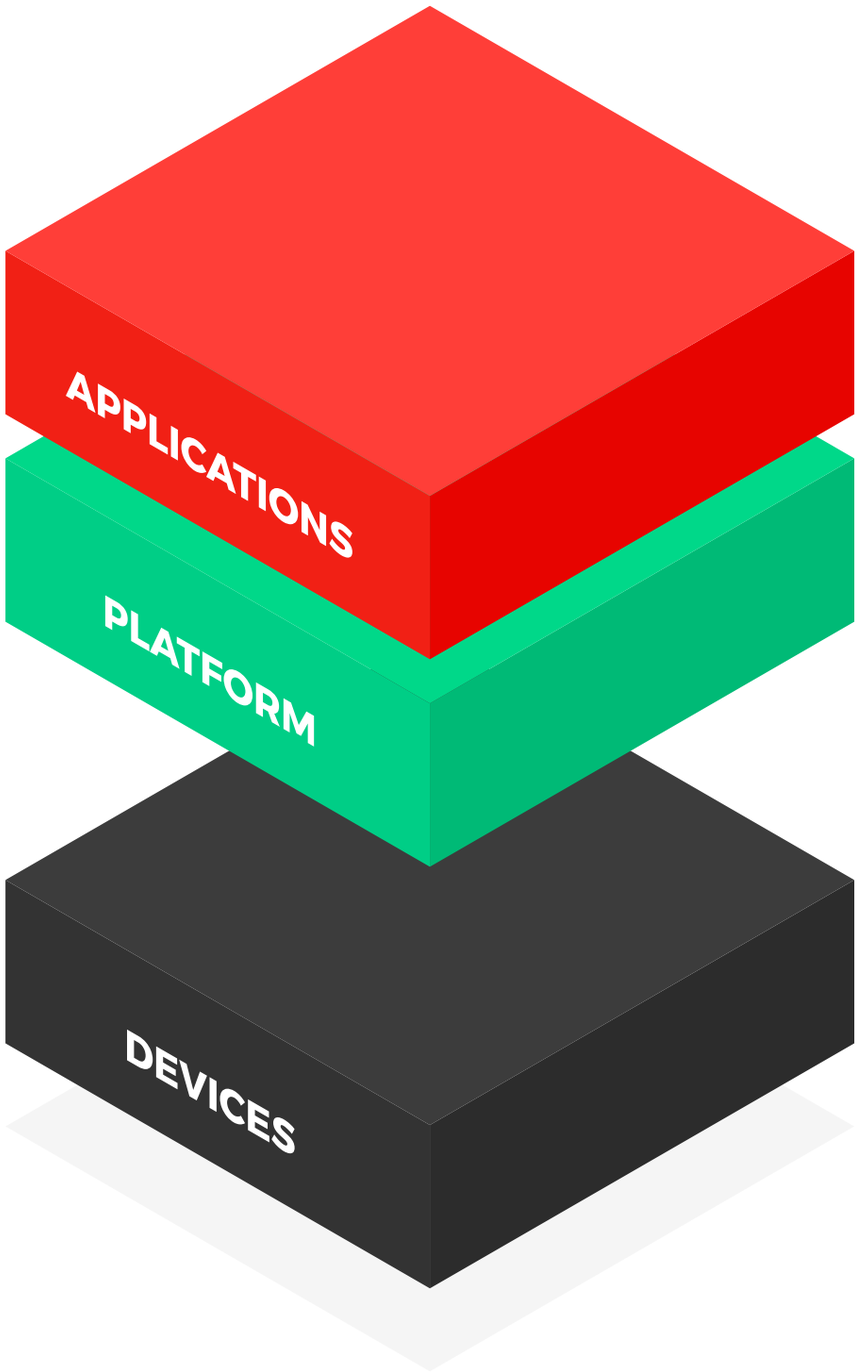


A typical IoT solution consists of a set of **applications**



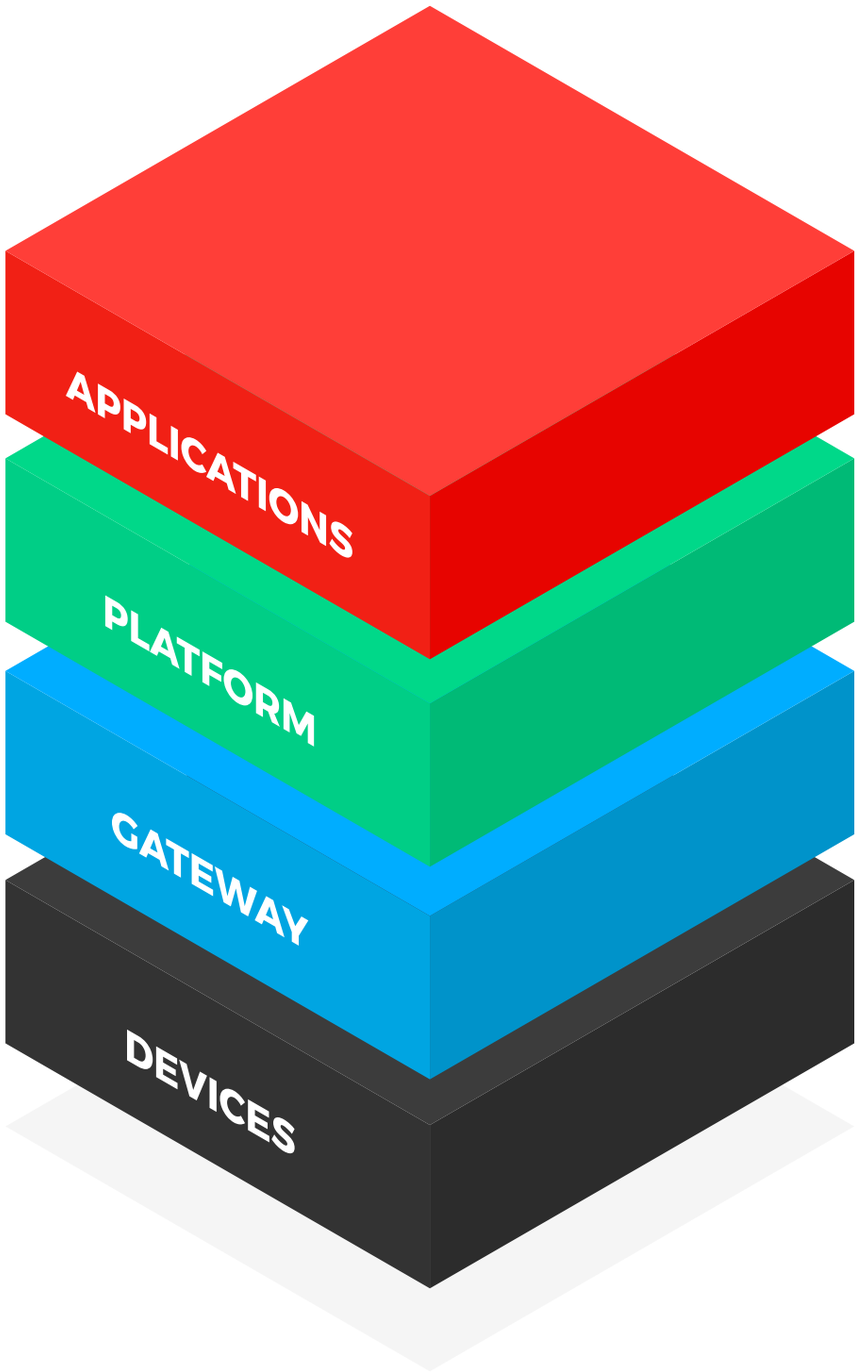
A typical IoT solution consists of a set of **applications** leveraging an IoT **platform**

GREENCOM NETWORKS



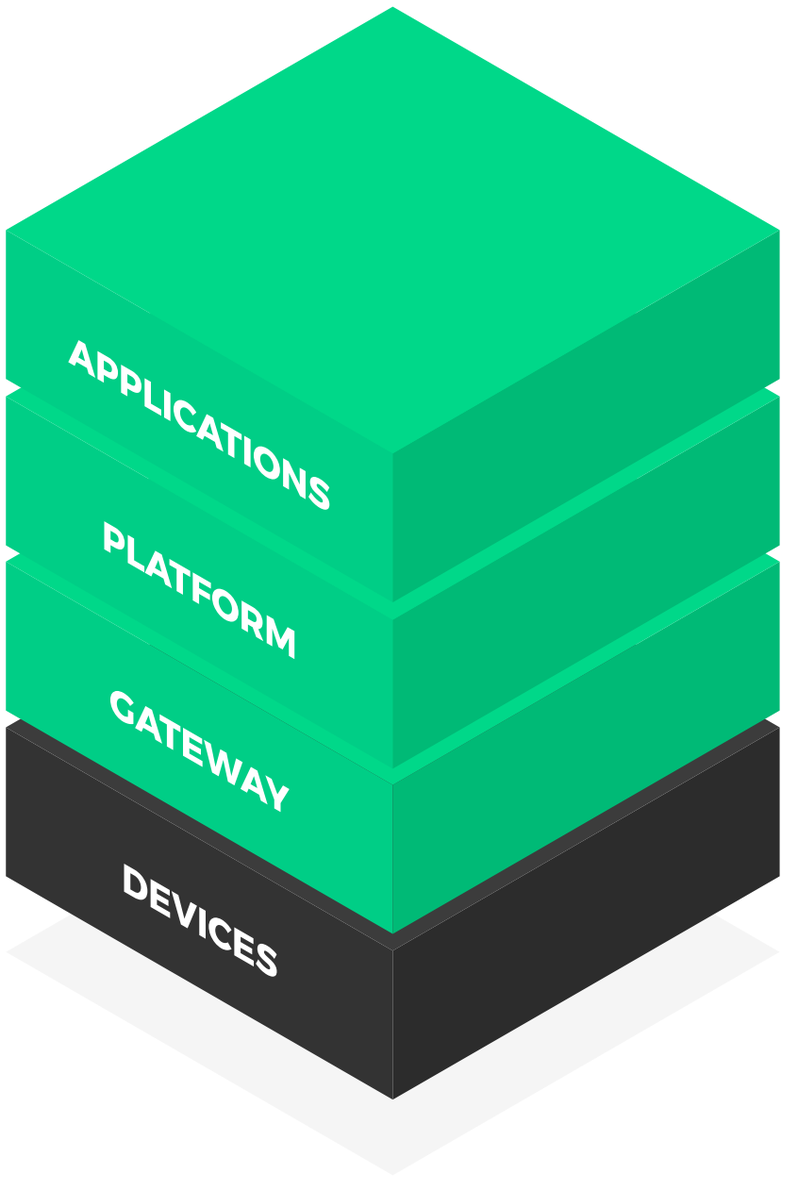
A typical IoT solution consists of a set of **applications** leveraging an IoT **platform** to connect to a large amount of **devices**

GREENCOM NETWORKS



A typical IoT solution consists of a set of **applications** leveraging an IoT **platform** to connect to a large amount of **devices** through some sort of **gateway**.

GREENCOM NETWORKS



GreenCom Networks through its various offerings is acting in the **application**, **platform** and **gateway** spaces.

The EIBP

Energy Information Brokerage Platform

- Energy related implementation of an IoE platform
- Built around a data collection engine and a high-throughput, low-latency message broker
- Offers ability to quickly create and deploy new energy related services
- secured
- multi tenant
- multi cloud provider
- scalable

A bit of history

EIBP Roadmap (past to present)

10+ years of experience
in Telco World

2009 — DATA COLLECTION PLATFORM

PLATFORM V1

(focus on energy data)

SQL Based

2012 — IoT PLATFORM

PLATFORM V2

NoSQL based

REST APIs

2015 — NEXT GEN PLATFORM

PLATFORM V3

Event driven,

streaming enabled,

containerized

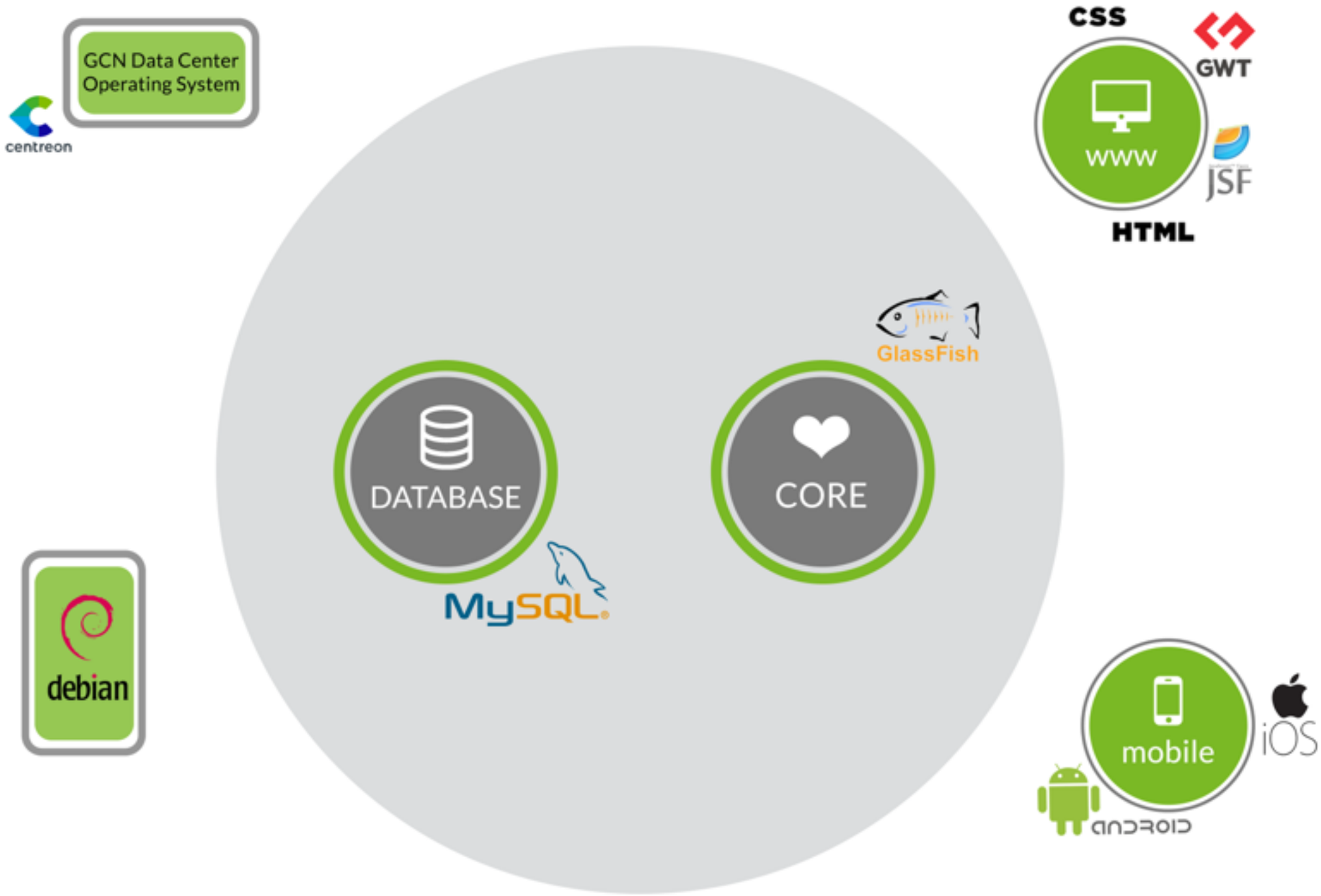
2018 — TO THE FUTURE

PLATFORM V3+

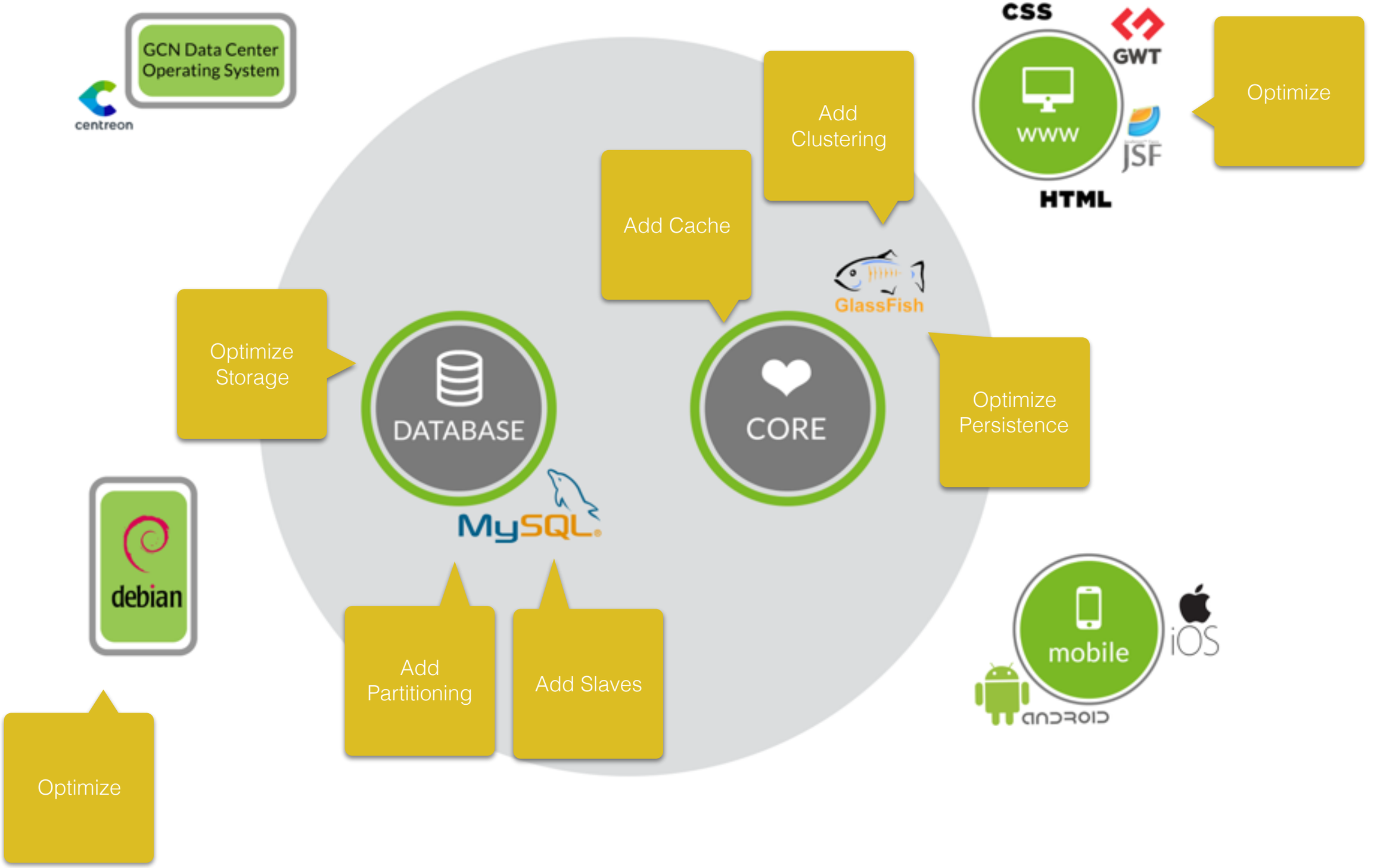
Multi cloud

Service mesh

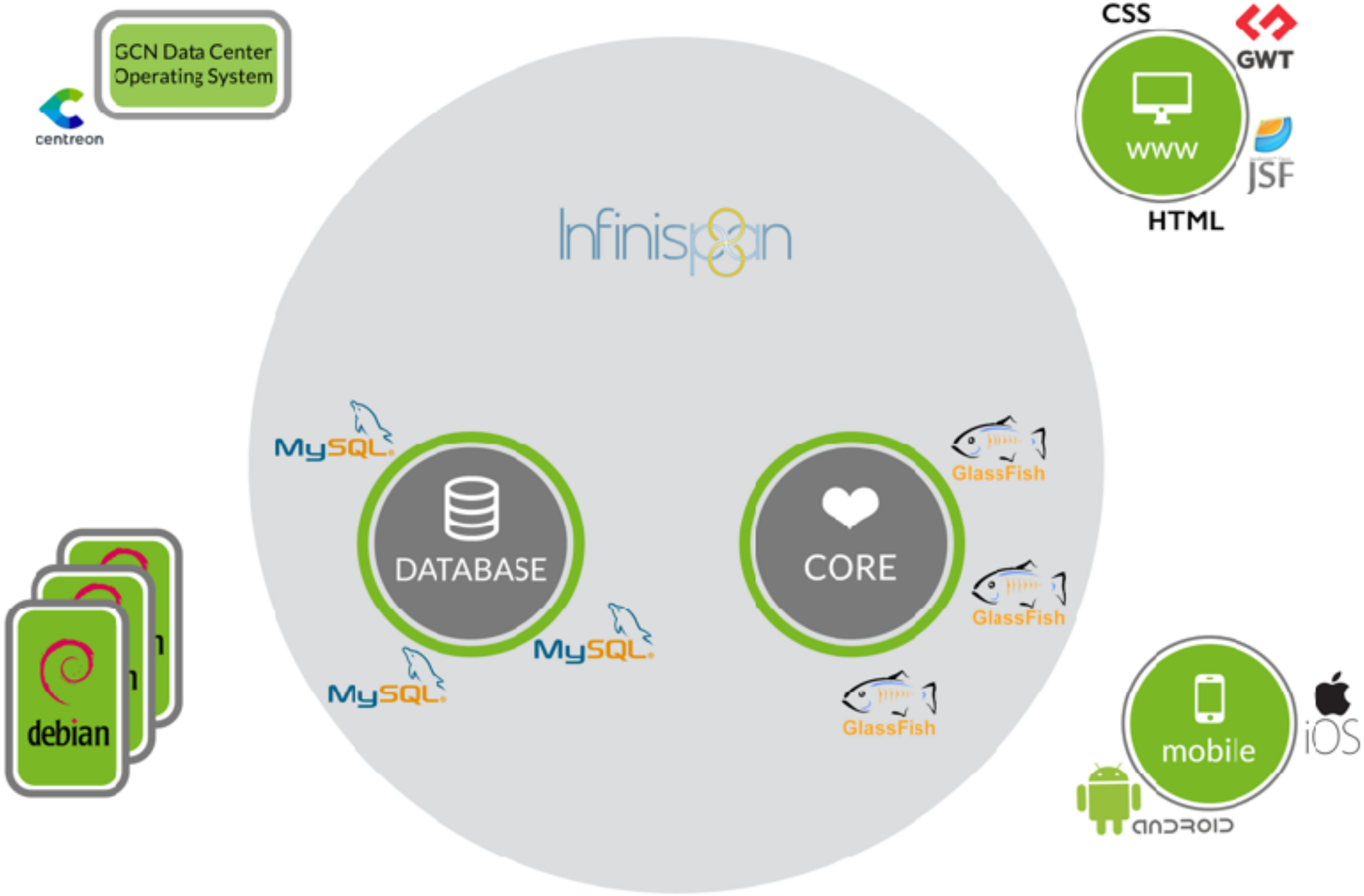
EIBP Platform V1



EIBP Platform V1 ... after a while



EIBP Platform V1 ... after a while



GREENCOM NETWORKS

When you start spending more time optimising your internals than creating value for your customers, it’s time to revise the architecture...

GREENCOM NETWORKS

Let's

~~revise~~

~~rework~~

reboot

What do we need ?

Time Series Storage / Management
Metadata Storage / Management
Scalability
Replication / Data center awareness (data safety)
Performance
Security

Speed of development
Agility

Time Series Storage:

- It will be NoSQL !!!!

Metadata Storage:

- SQL or NoSQL ? —> NoSQL

Which NoSQL database ?

- Lack of maturity of TSDB at that time
- Need to store not only time series but also metadata
- CAP Theorem

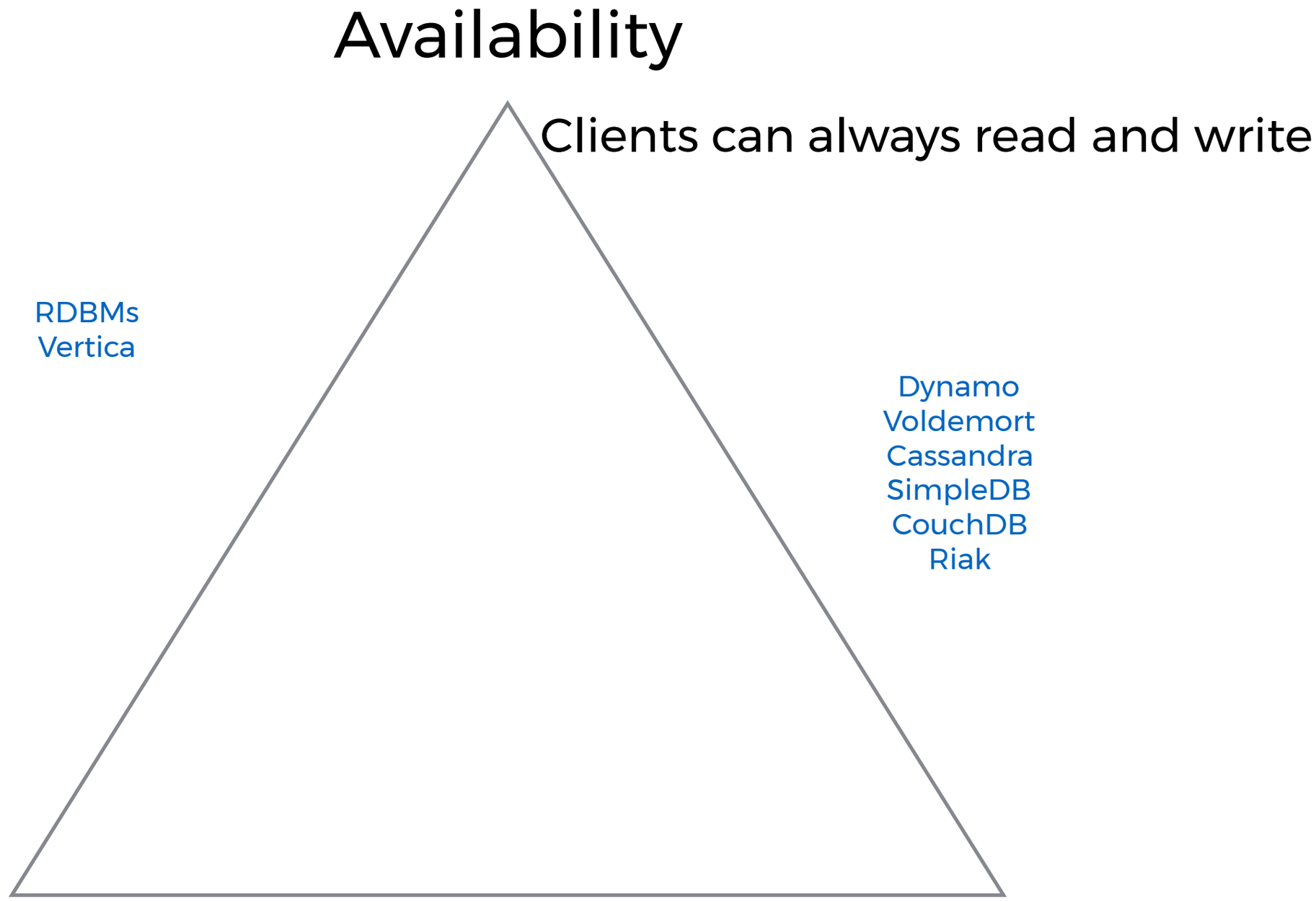
CAP theorem

In theoretical computer science, the CAP theorem, also known as Brewer's theorem, states that it is impossible for a distributed computer system to simultaneously provide all three of the following guarantees:

- Consistency (all nodes see the same data at the same time)
- Availability (a guarantee that every request receives a response about whether it succeeded or failed)
- Partition tolerance (the system continues to operate despite arbitrary partitioning due to network failures)

Wikipedia

CAP theorem



Cassandra

Apache Cassandra is an **open source, distributed** database management system designed to handle **large amounts of data** across many **commodity servers**, providing **high availability** with **no single point of failure**. Cassandra offers robust support for clusters spanning **multiple datacenters**, with asynchronous masterless replication allowing low latency operations for all clients.

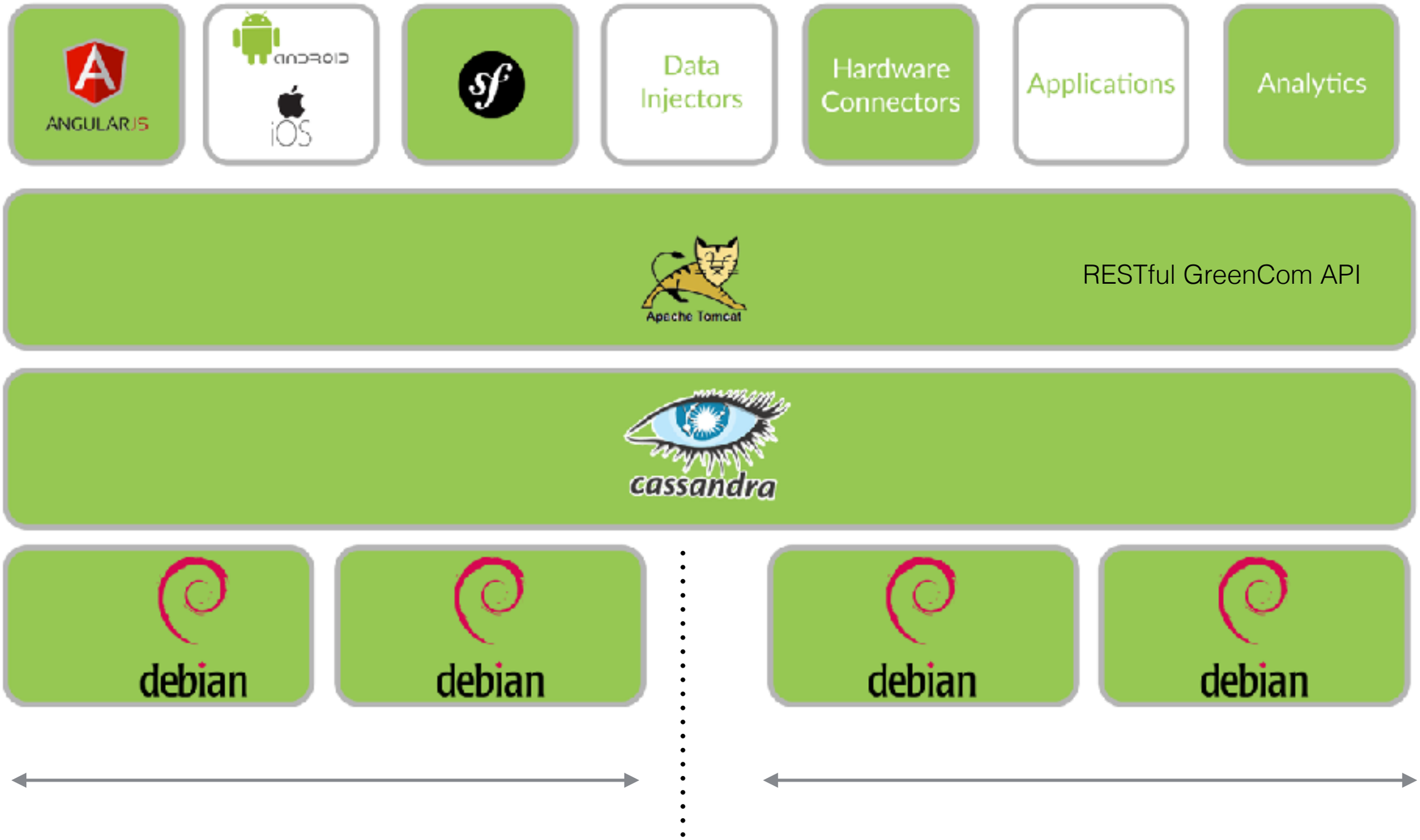
Cassandra also places a high value on **performance**.

Apache Cassandra was initially developed at Facebook to power their Inbox Search feature.
It was released as an open source project in 2008.

Wikipedia

EIBP Platform V2

GREENCOM NETWORKS



EIBP Platform V2

- Simple architecture
- Agile
- API Based
- Resilient
- Built to scale
- Easy to maintain
- Easy to deploy

What do we need ?

Time Series Storage / Management
Metadata Storage / Management
Scalability
Replication / Data center awareness (data safety)
Performance
Security

=> Cassandra

Speed of development
Agility

=> Micro Service Architecture

Unix Philosophy...

"Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new features."

"Expect the output of every program to become the input to another, as yet unknown, program."

Doug McIlroy,
inventor of Unix pipes and one of the founders of the Unix tradition

GREENCOM NETWORKS

Micro-services architecture:

- Simple: small code base, easy to create, understand and maintain.
- Fits in the head of one developer
- The services are small - fine-grained as a singular business purpose similar to the Unix philosophy of "Do one thing and do it well"
- The services are elastic, resilient, composable, minimal, and complete.
- Easy to test
- Easy to deploy (each component can de deployed/upgraded separately; can leverage containers)
- Easy to scale : only the required component can be scaled (!= monolithic architecture)
- Use the right language/stack for the right task
- trivial lifecycle management : in a fast moving environment, micro services can be updated/modified/thrown away easily.
- Resilience: if a service fails, impact on the platform is limited

How to create good micro services

GREENCOM NETWORKS

It’s not only about:
- splitting
- replacing function call with network calls

But also about:
- Loose coupling (changing a service must not impact the others)
- High cohesion (related behaviour sits together)
- Technology agnostic API
- KISS principle (Keep It Simple Stupid) to ease consumption
- Keep scaling in mind (external persistence)

micro services @GCN

GREENCOM NETWORKS

350+ services
1000+ instances
20 customers

- making heavy use of RESTful interfaces
- carrying JSON representation
- over HTTP / HTTPs

in a wide variety of languages and frameworks

Heterogeneity (language / frameworks)

Heterogeneity (language / frameworks)

A true language

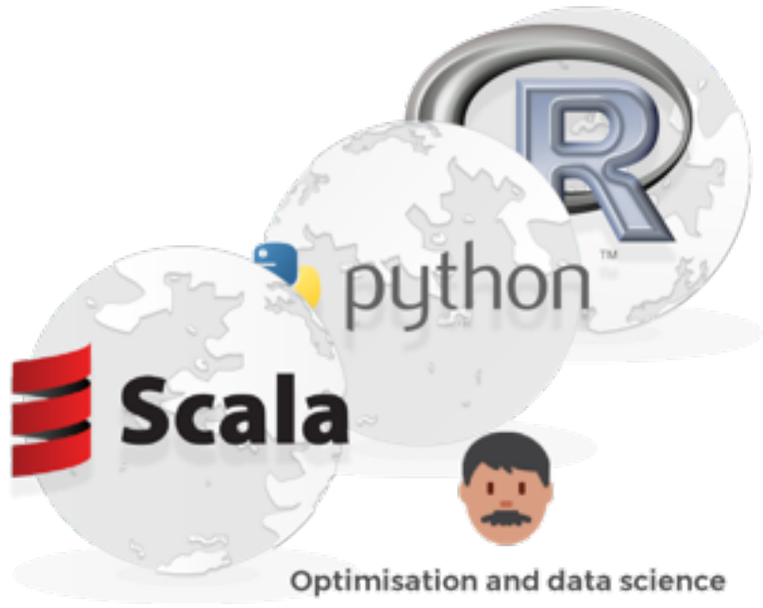
Fading

For old guys

UI

For young guys

Nice logo



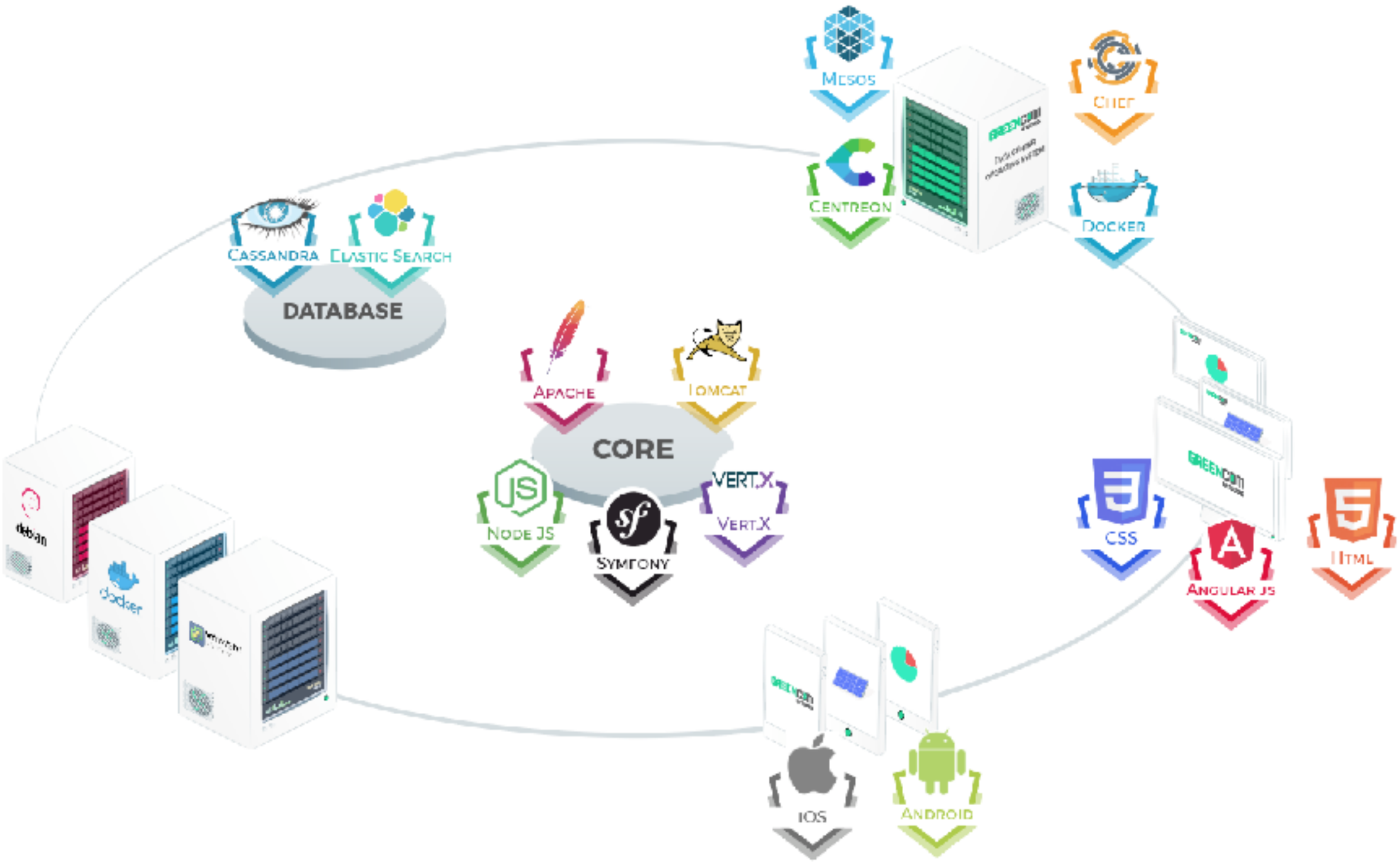
Optimisation and data science

GREENCOM NETWORKS

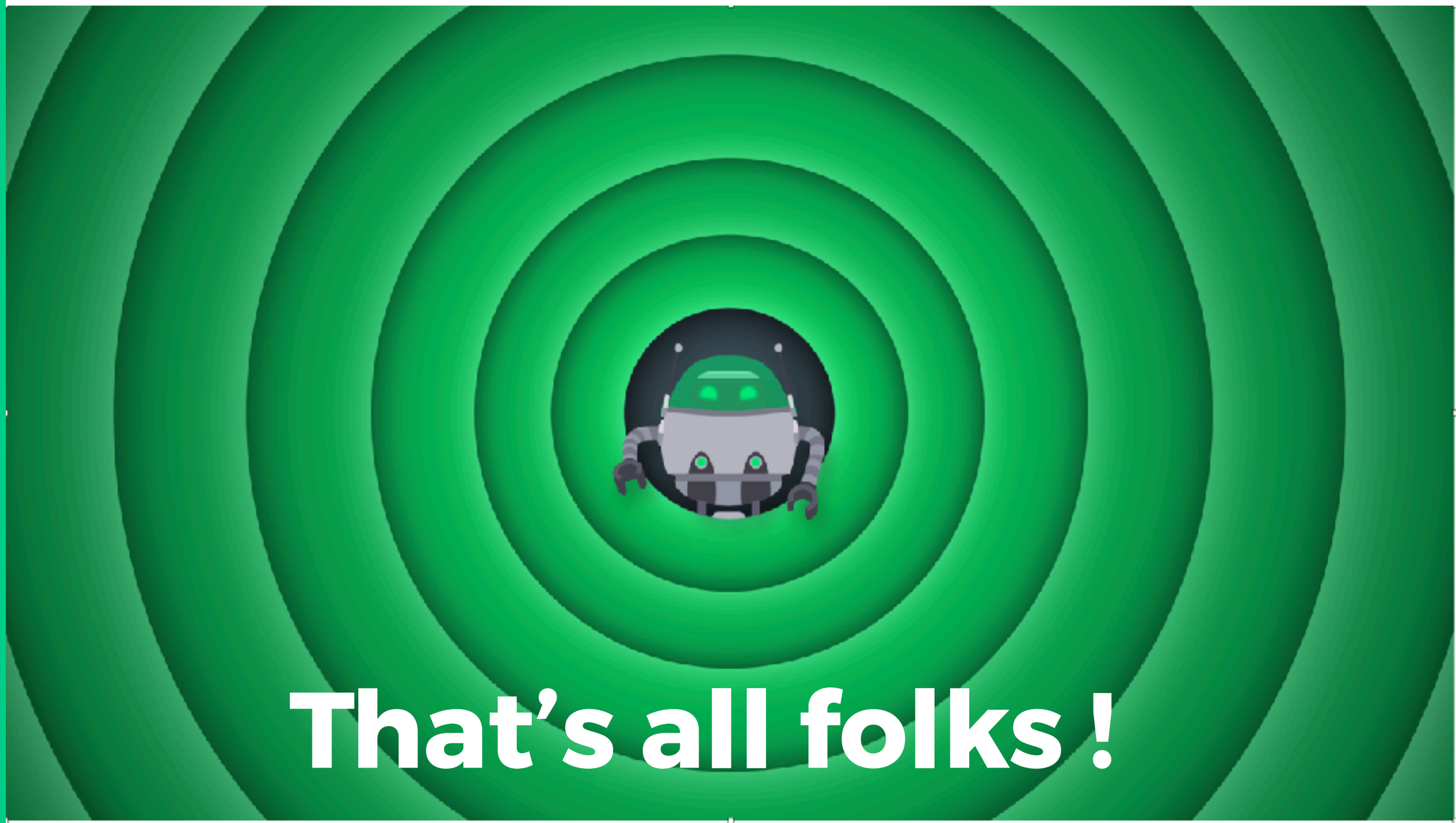
But… Not a silver bullet

- The organisation culture should embrace automation of deployment and testing.
- Adds complexity on the monitoring side
- Fault diagnostic can be complex
- The culture and design principles should embrace failure and faults
- Too early decomposition might jeopardise the entire system


EIBP Platform V2+



GREENCOM NETWORKS



That’s all folks !

Oh…
Wait….

What if I want
to
react in real time ?

Event vs Data

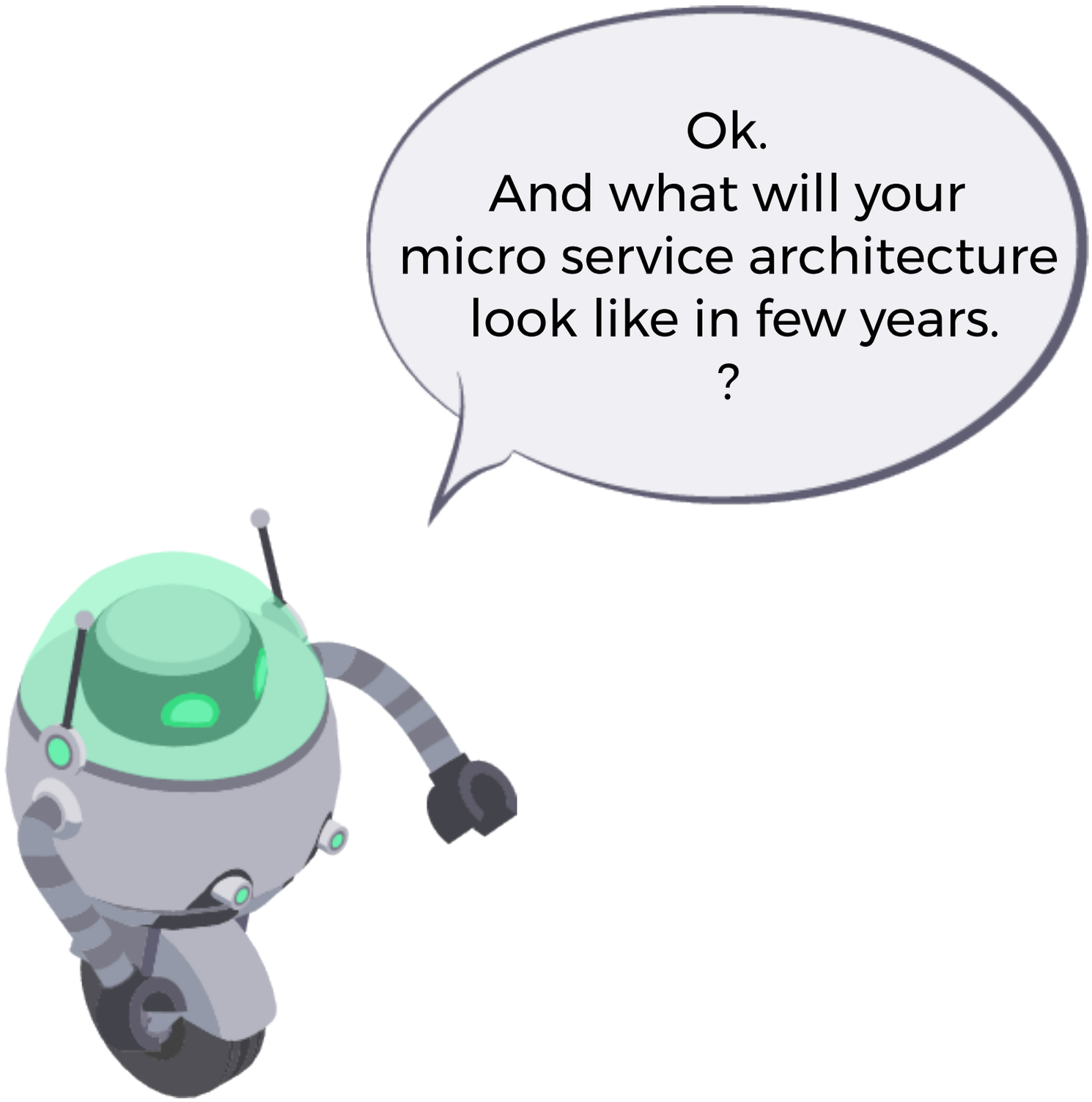
Everything is event

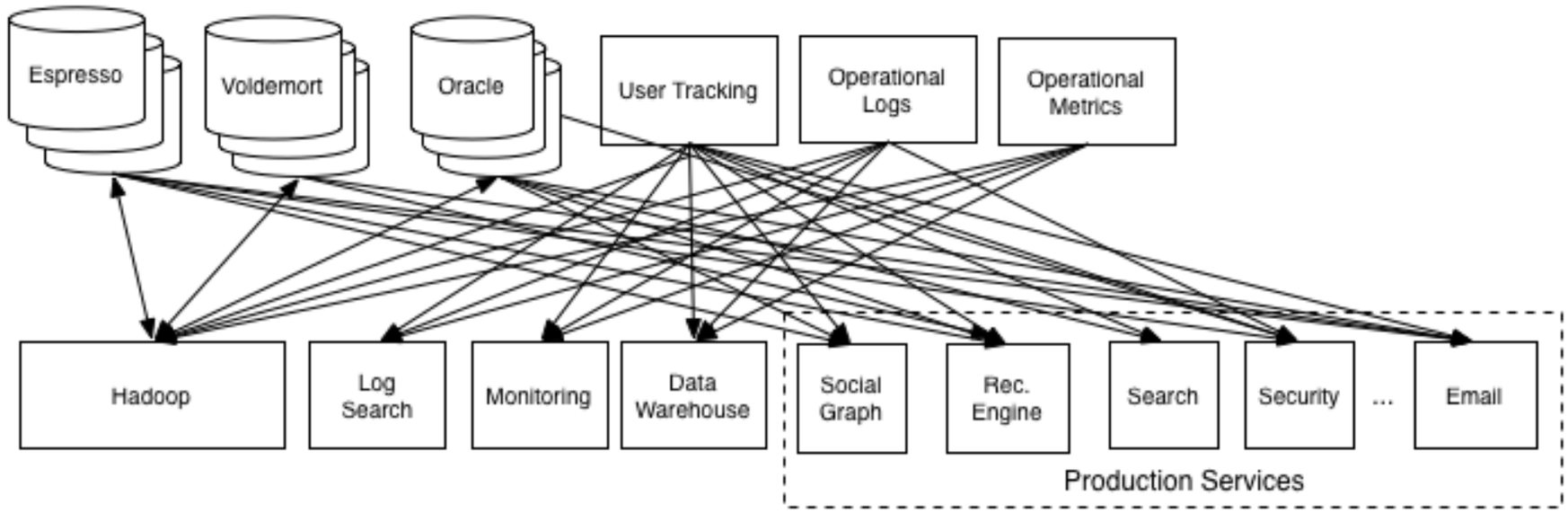
- Meter reading
- EV plugged
- Account Created

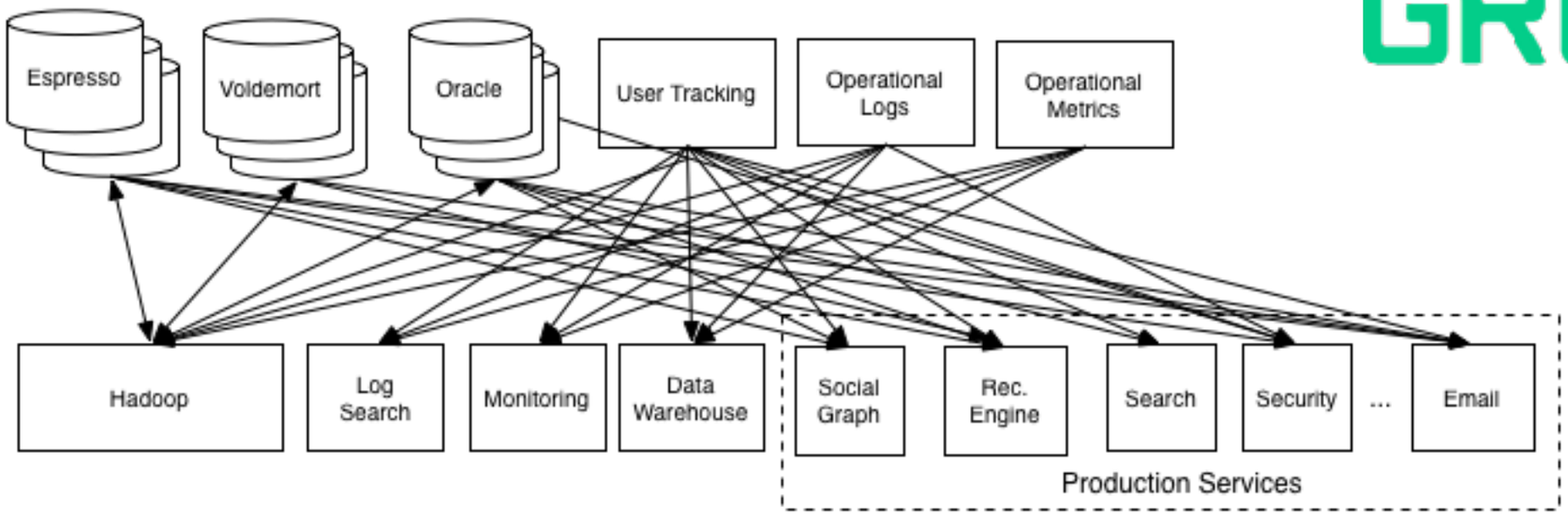
Storing it creates data

But what if we want to react in real time ?

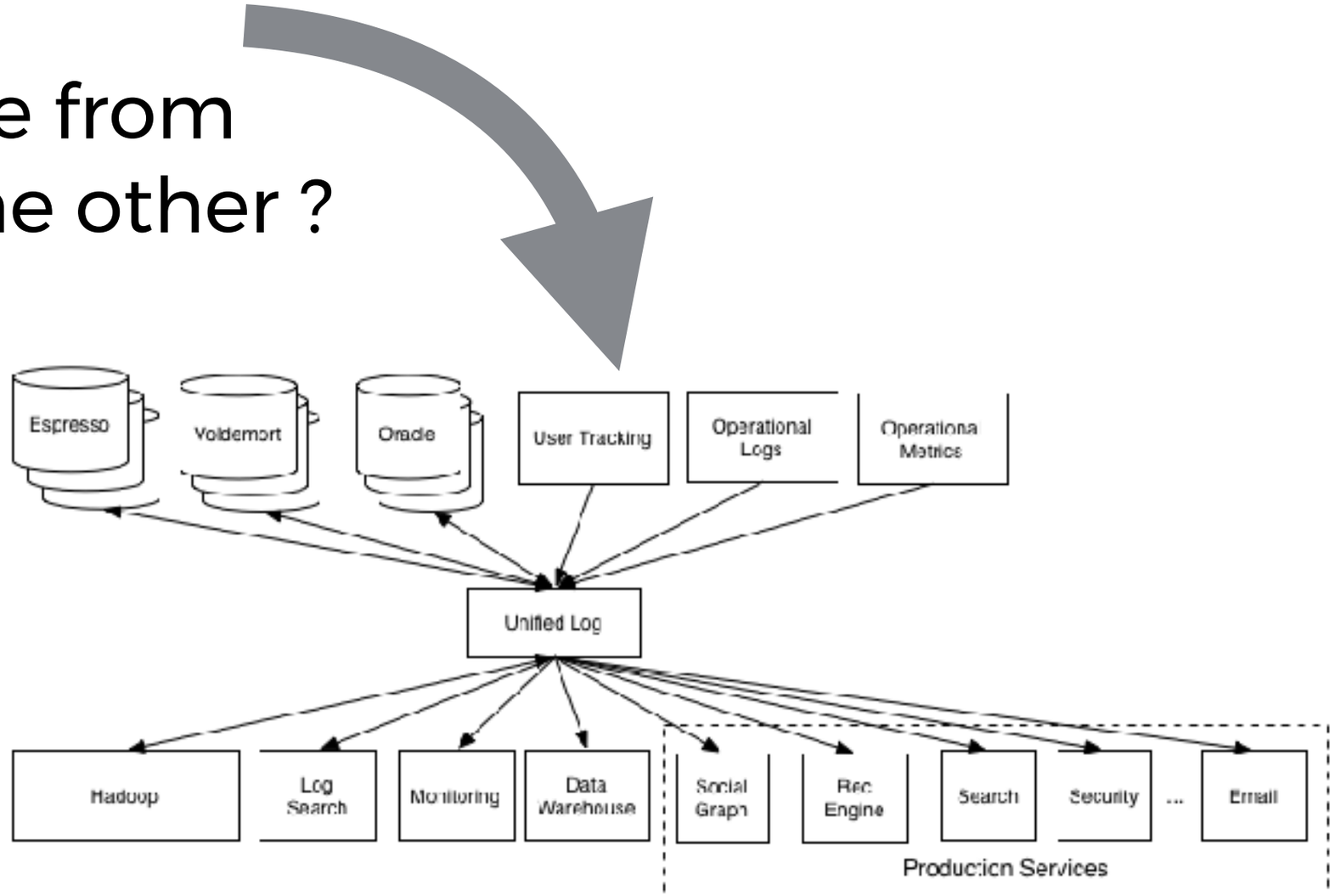
Event is lost...
Option 1 : process the event while receiving it, hardwiring use cases (not flexible)
Option 2 : recreate the event later (beurk)





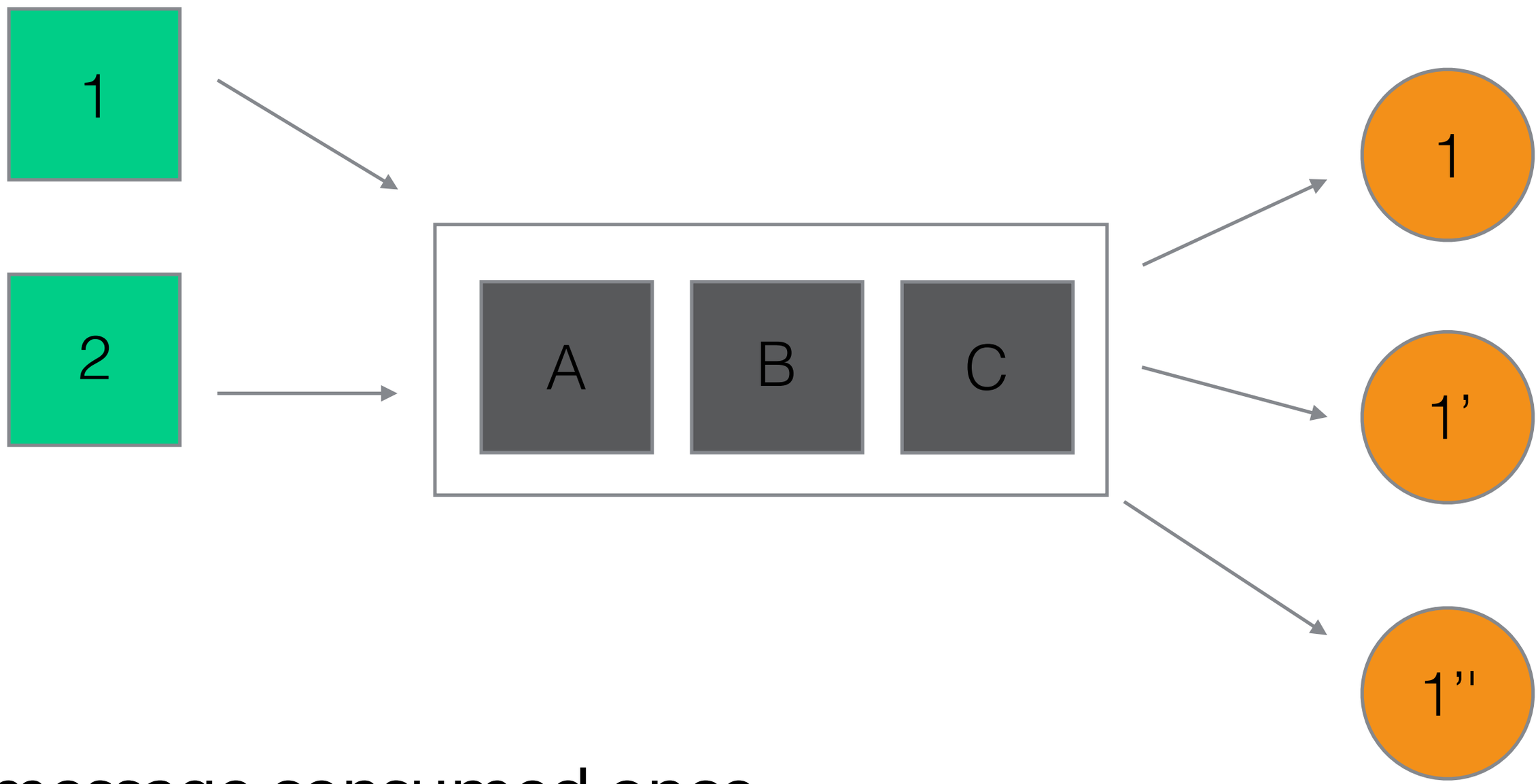


How do we move from 1 architecture to the other ?



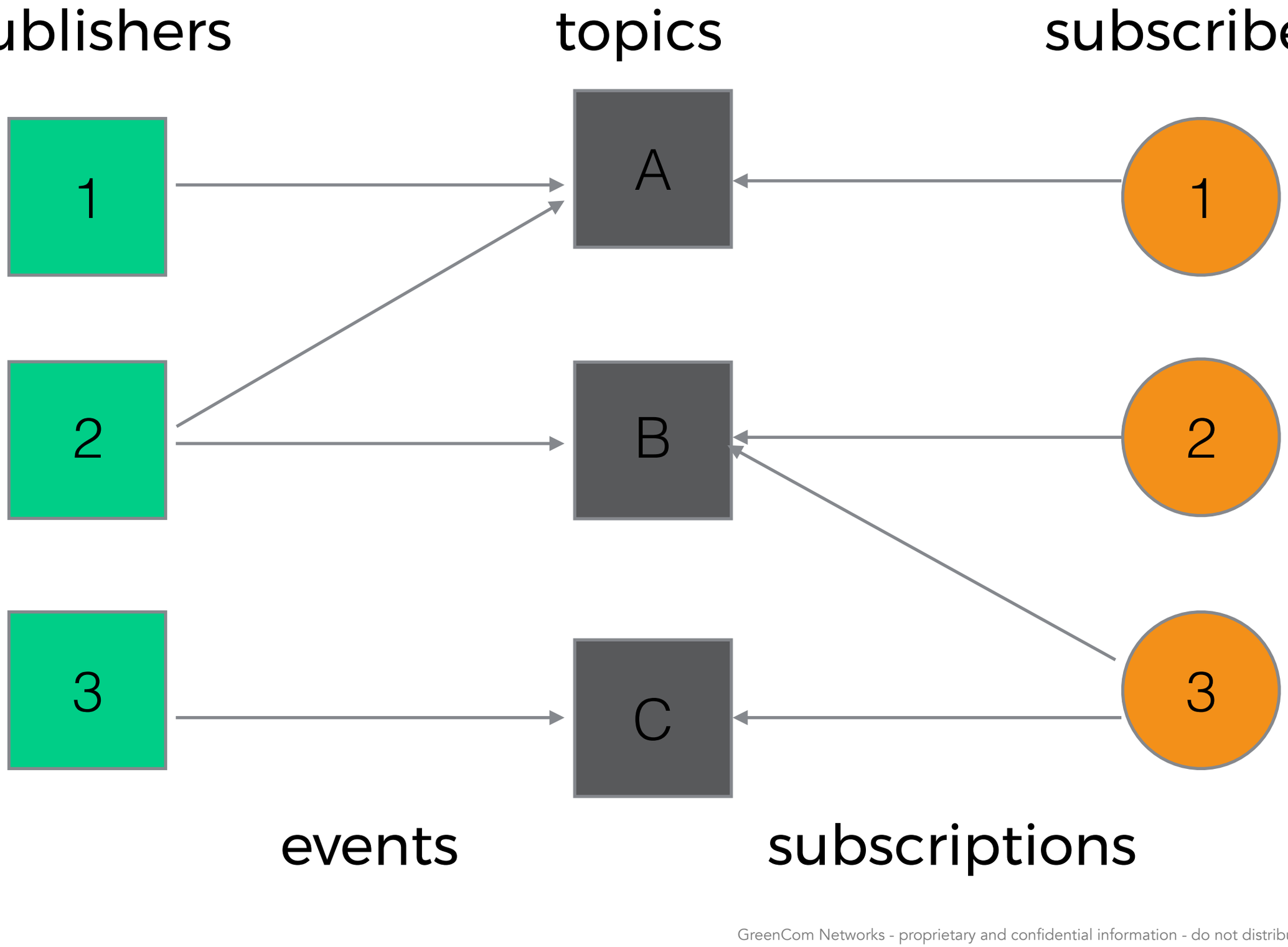
https://engineering.linkedin.com/distributed-systems/log-what-every-software-engineer-should-know-about-real-time-datas-unifying

Queues



message consumed once

Publish Subscribe



Ok, we need a messaging system then...

Name dropping

ActiveMQ
RabbitMQ
ZeroMQ*
HornetQ
...

Kafka
Kerstrel (RIP)
DistributedLog
...

SNS
SQS
MessageHub
EventHub
IoTHub

Drivers for choice

The target solution will have to be:

Fast (performant)
Reliable (no SPOF)
Agile (easy to deal with)
Scalable (horizontal scalability)
Cheap (operationally)
Secure

Apache Kafka

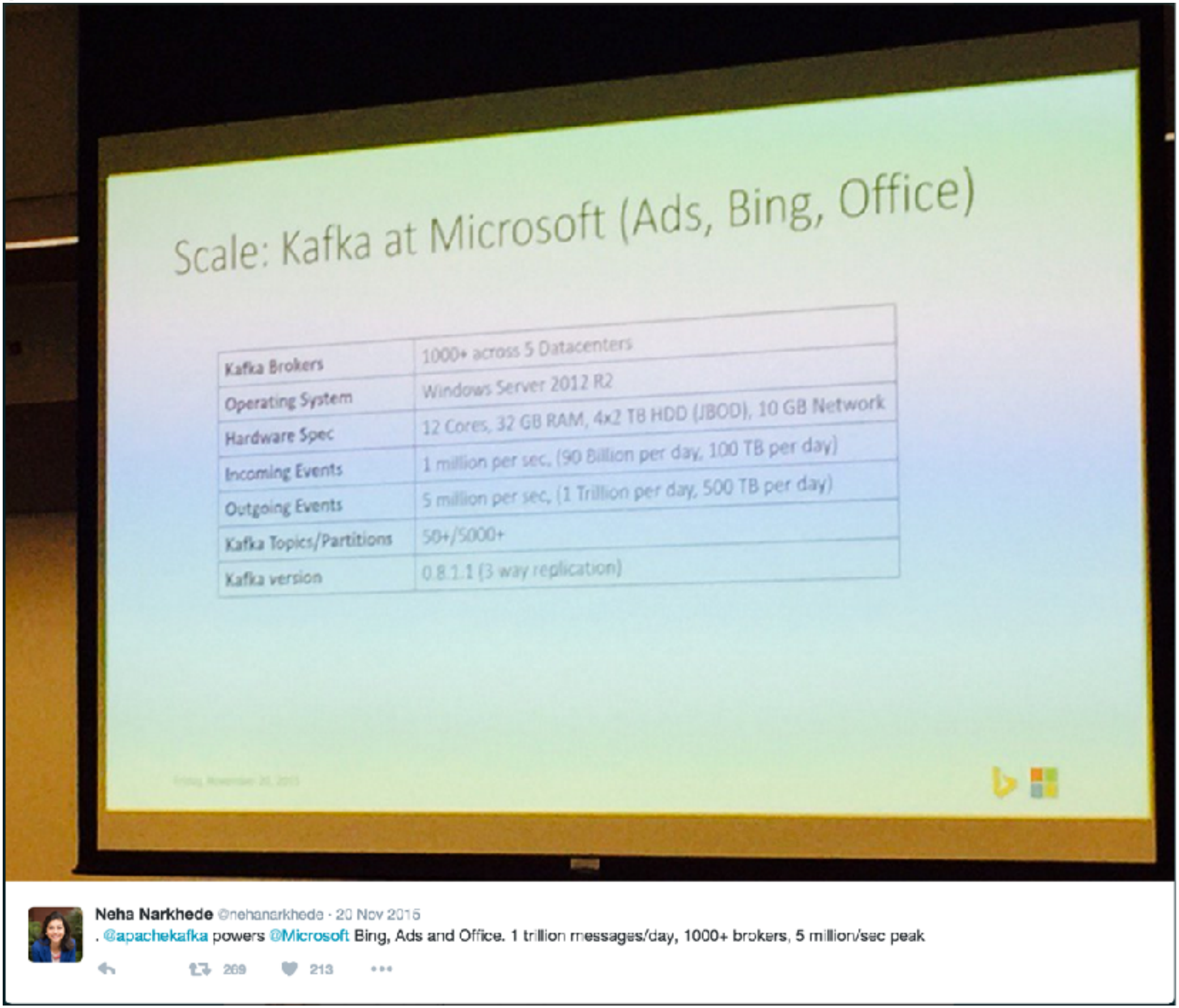
A high-throughput distributed messaging system.

Apache Kafka is publish-subscribe messaging rethought as a distributed commit log.

Fast
A single Kafka broker can handle hundreds of megabytes of reads and writes per second from thousands of clients.
Scalable
Kafka is designed to allow a single cluster to serve as the central data backbone for a large organization. It can be elastically and transparently expanded without downtime. Data streams are partitioned and spread over a cluster of machines to allow data streams larger than the capability of any single machine and to allow clusters of co-ordinated consumers
Durable
Messages are persisted on disk and replicated within the cluster to prevent data loss. Each broker can handle terabytes of messages without performance impact.
Distributed by Design
Kafka has a modern cluster-centric design that offers strong durability and fault-tolerance guarantees.

http://kafka.apache.org/

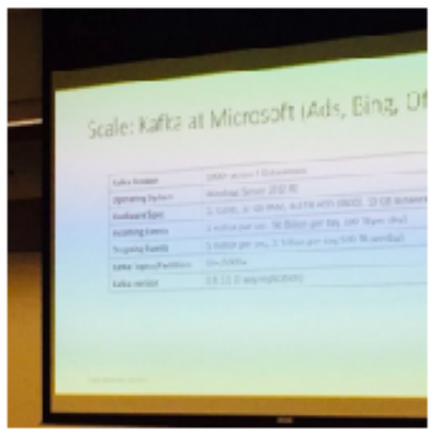
Kafka



Kafka

Octave Klaba / Oles @olesovhcom · 27 Nov 2015

> 5 millions per sec here .. with lot of BigHG 24x600Go SAS each .. :)



Neha Narkhede @nehanarkhede

. @apachekafka powers @Microsoft Bing, Ads and Office.
1 trillion messages/day, 1000+ brokers, 5 million/sec peak

4 7

@linkedin:
400 nodes,
18k topics,
220 billions msg/day (pic 3.2 millions msg/s)

May 2014

Kafka deployments

LinkedIn

400 nodes,
18k topics,
220 billions msg/day
peak 3.2 millions msg/s

NETFLIX

50 clusters
4,000 nodes
3 AWS regions
1 trillion msg/day

Microsoft

1000 nodes
5 datacenters
1 trillion msg/day
peak 5 millions msg/s

GREENCOM NETWORKS

Kafka deployments

400 nodes,
18k topics,
220 billions msg/day
peak 3.2 millions msg/s

50 clusters
4,000 nodes
3 AWS regions
1 trillion msg/day

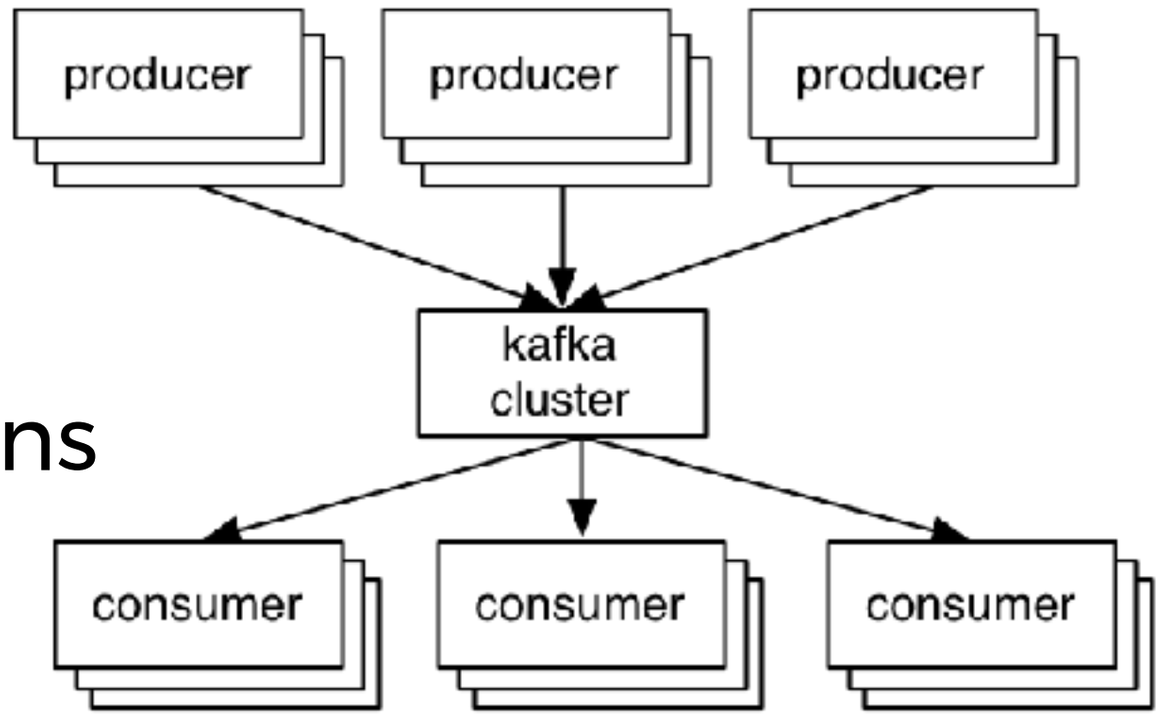
1000 nodes
5 datacenters
1 trillion msg/day
peak 5 millions msg/s

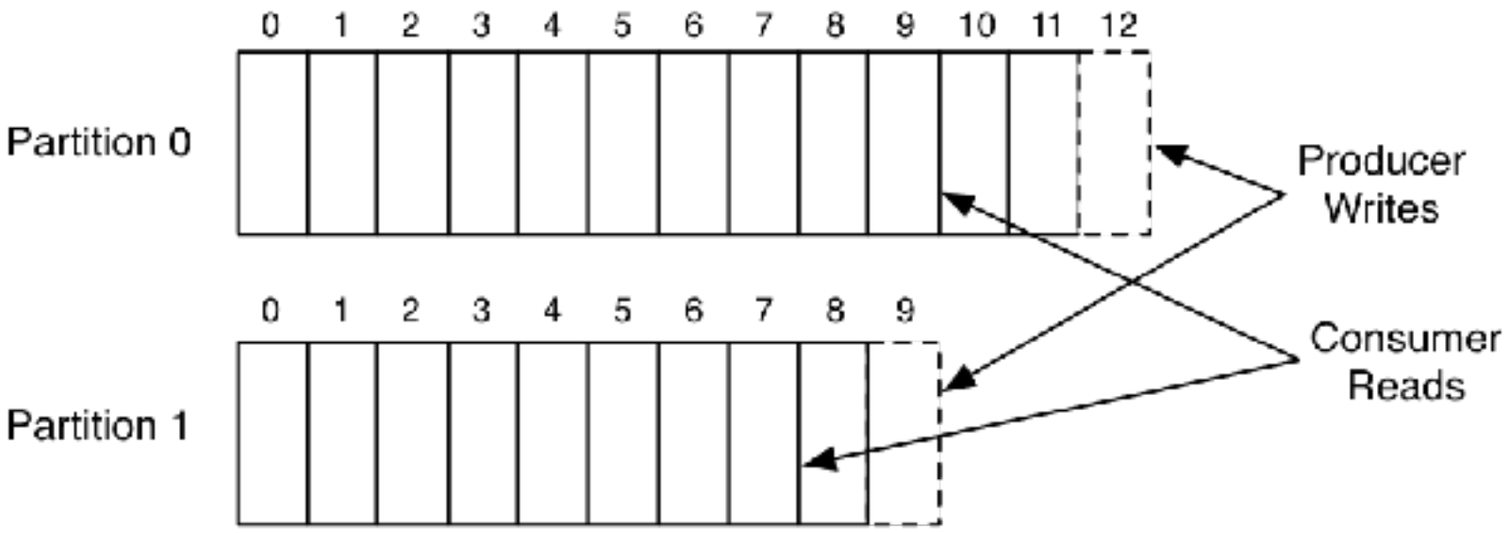
4 clusters
8 nodes in production,
1.8k topics,
500 millions msg/day
peak 5000 msg/s

Kafka concepts

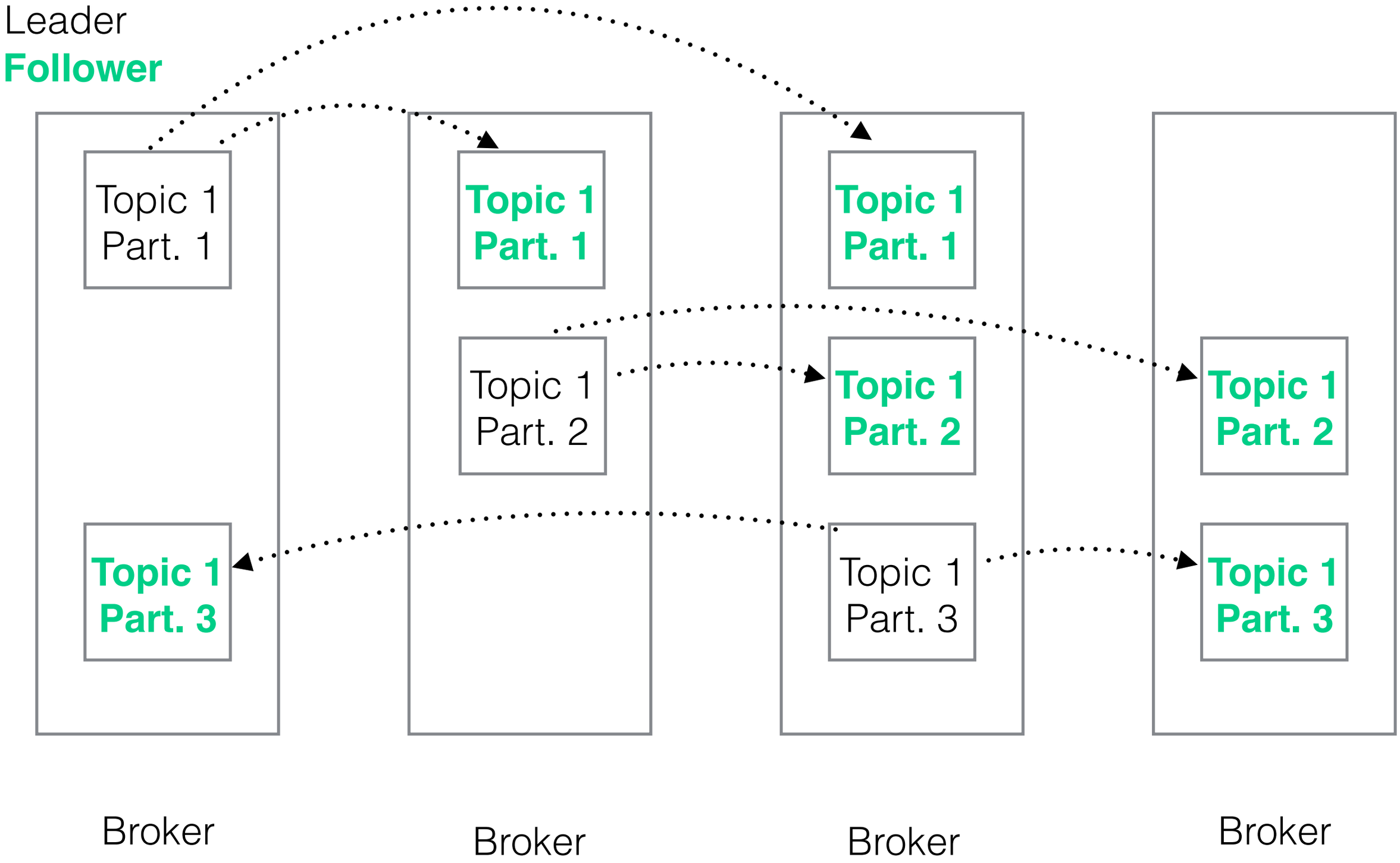
producers produce
consumers consume :-o

- data are stored in topics
- topics are split in partitions
- partitions are replicated

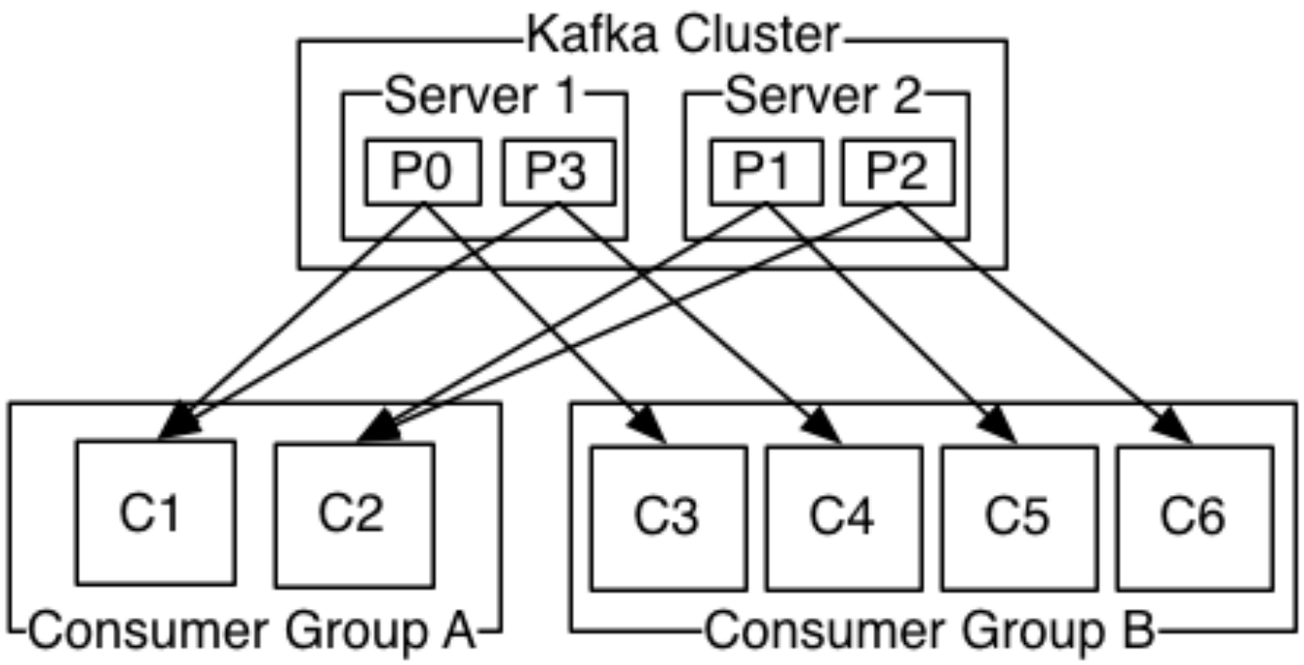




Kafka concepts



Kafka concepts



Number of partitions for a topic fixes the maximum number of consumers for a consumer group.

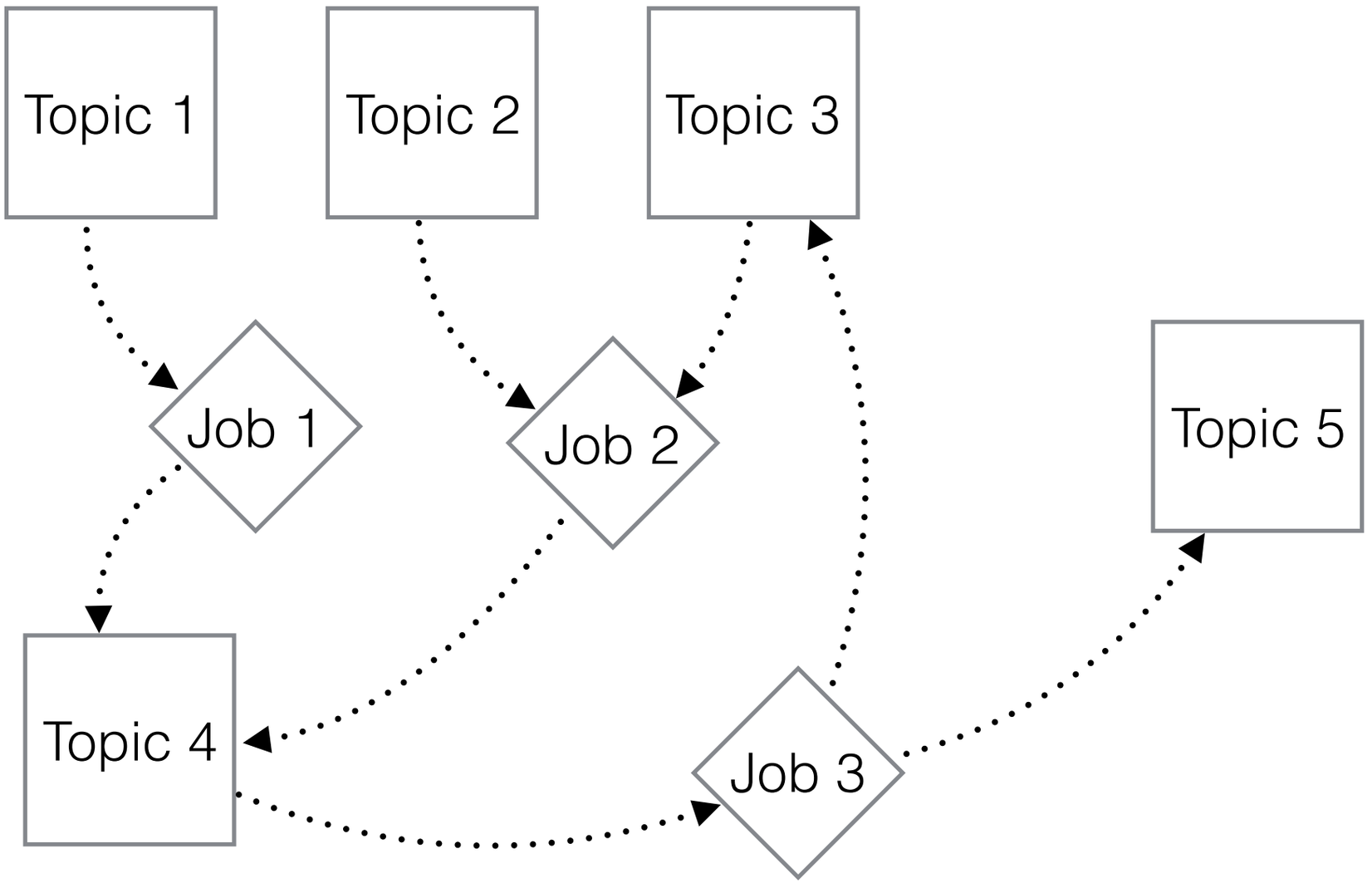
Kafka concepts

Seb, don’t forget to say few words regarding:

- how to scale (cluster & app)
- ordering
- keying
- retention

And now...

Now that we have a way of producing/consuming messages we can create complex event processing pipelines



Event processing

Producing to Kafka or consuming from Kafka is easy.
Client library exist for most of the languages

```
> bin/kafka-console-producer.sh --broker-list localhost:9092 --topic test
This is a message
This is another message
```

```
> bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic test --from-beginning
This is a message
This is another message
```

But for some high throughput complex task. we might want to rely on existing framework / applications

STORM

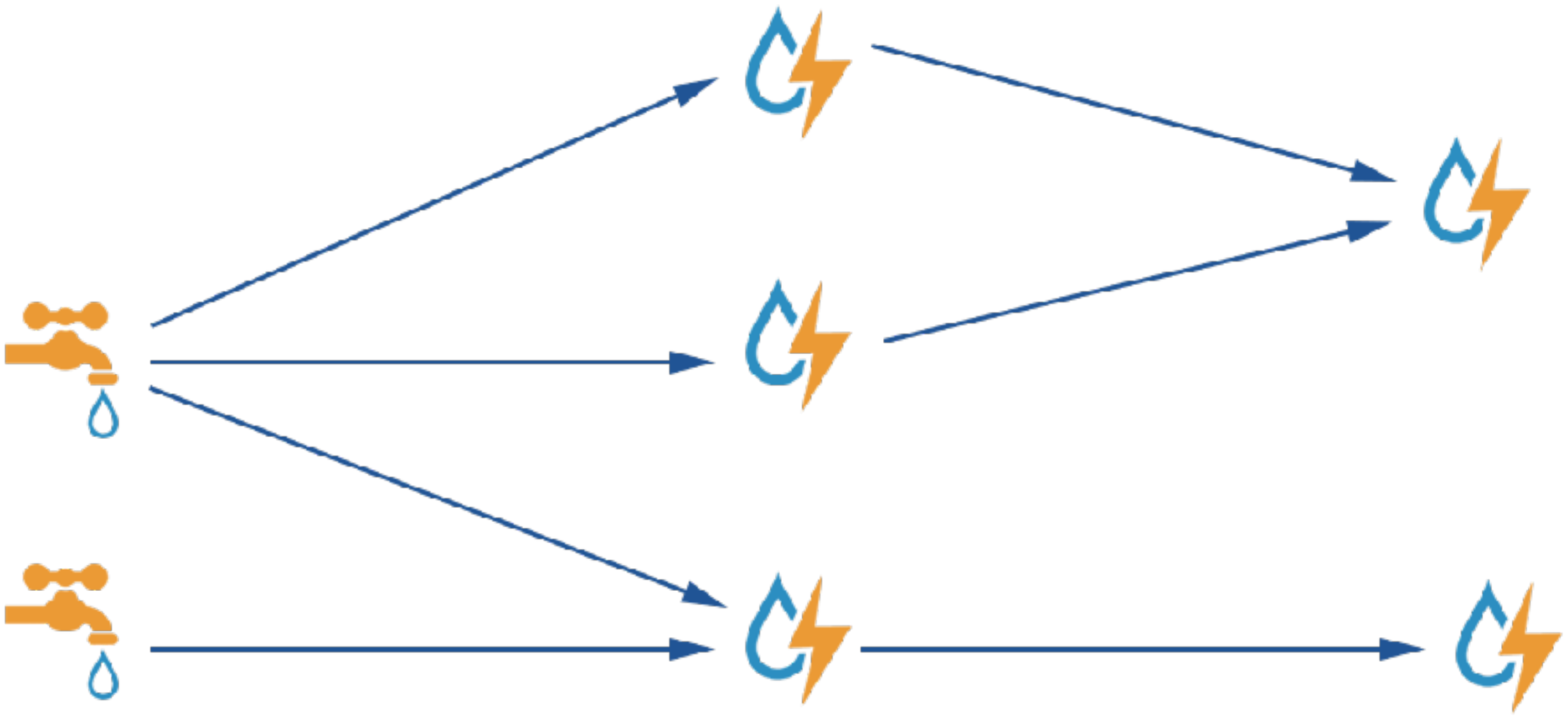
Apache Storm is a free and **open source distributed realtime computation system**. Storm makes it easy to reliably process unbounded streams of data, doing for realtime processing what Hadoop did for batch processing. Storm is simple, can be used with any programming language, **and is a lot of fun to use!**

Storm has many use cases: realtime analytics, online machine learning, continuous computation, distributed RPC, ETL, and more. Storm is **fast**: a benchmark clocked it at **over a million tuples processed per second per node**. It is **scalable**, **fault-tolerant,** guarantees your data will be processed, and is easy to set up and operate.

Storm integrates with the queueing and database technologies you already use. A Storm topology consumes streams of data and processes those streams in arbitrarily complex ways, repartitioning the streams between each stage of the computation however needed. Read more in the tutorial.

http://storm.apache.org/

Storm concepts



http://storm.apache.org/

Storm architecture

ZK Quorum

Nimbus
(master)

Storm UI

Supervisor
(slave)

workers

Supervisor
(slave)

workers

Some figures

STORM

Apache Storm

1M+ messages/second on a 10 nodes cluster

mid-2017 @gcn:

3 clusters

170 topologies

Do we really need that ?

100 000 households
100 endpoints/ site
15 minutes interval
datapoints

960 000 000 direct raw events / day
> 10 000 events / second

Only talking about DIRECT events.
final figures will be more around
50 000 events / second

but our reality is closer to:

100 000 households
100 endpoints/ site
30 seconds interval
datapoints

28 800 000 000 direct raw events / day
> 350K events / second

Still wanna use MS Access ? :-)

some other frameworks exist / can be used

GREENCOM NETWORKS

Apache Spark is a fast and general engine for large-scale data processing.

Akka is an open-source toolkit and runtime simplifying the construction of concurrent and distributed applications on the JVM.

Heron is a distributed streaming processing engine developed at Twitter. According to the creators at Twitter, the scale and diversity of Twitter data has increased, and Heron is a real-time analytics platform to process streaming.

Apache Samza is a distributed stream processing framework. It uses Apache Kafka for messaging, and Apache Hadoop YARN to provide fault tolerance, processor isolation, security, and resource management.

Some figures

Apache Storm

1M+ messages/second on a 10 nodes cluster

mid-2017 @gcn:

3 clusters

170 topologies

Some figures

STORM

Apache Storm

1M+ messages/second on a 10 nodes cluster

end 2017 @gcn:

0 clusters

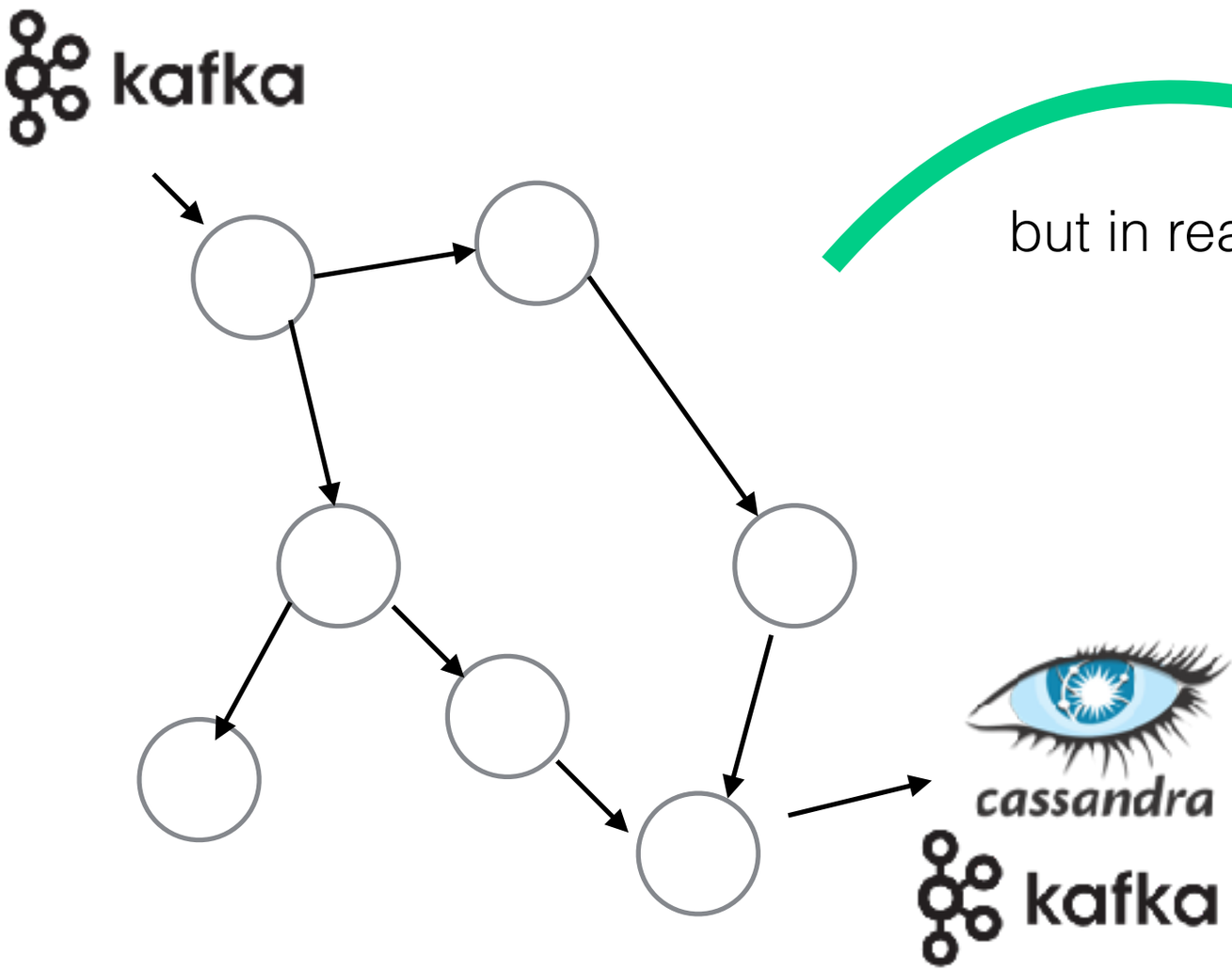
0 topologies

oups...
but why ?

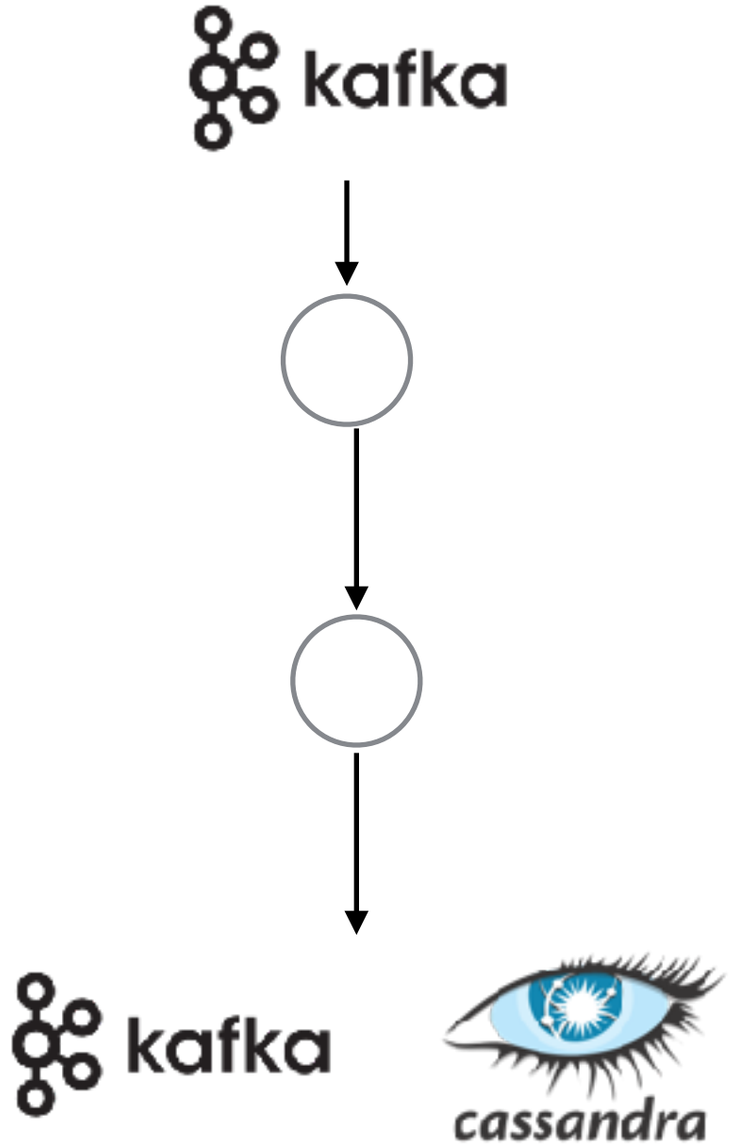
Why ?

GREENCOM NETWORKS

Un-needed complexity due to poor DAG



but in reality



Why ?

Logging

Where are my logs ?

Not so dynamic scalability (in our case)

topology.yaml

Why ?

GREENCOM NETWORKS

Yet an additional cluster

kafka

STORM

MESOS

APACHE Spark™

cassandra

elasticsearch

So now?

Kafka Streams API

Elastic, highly scalable, fault-tolerant
Deploy to containers, VMs, bare metal, cloud
Equally viable for small, medium, & large use cases
Write standard Java applications
No separate processing cluster required

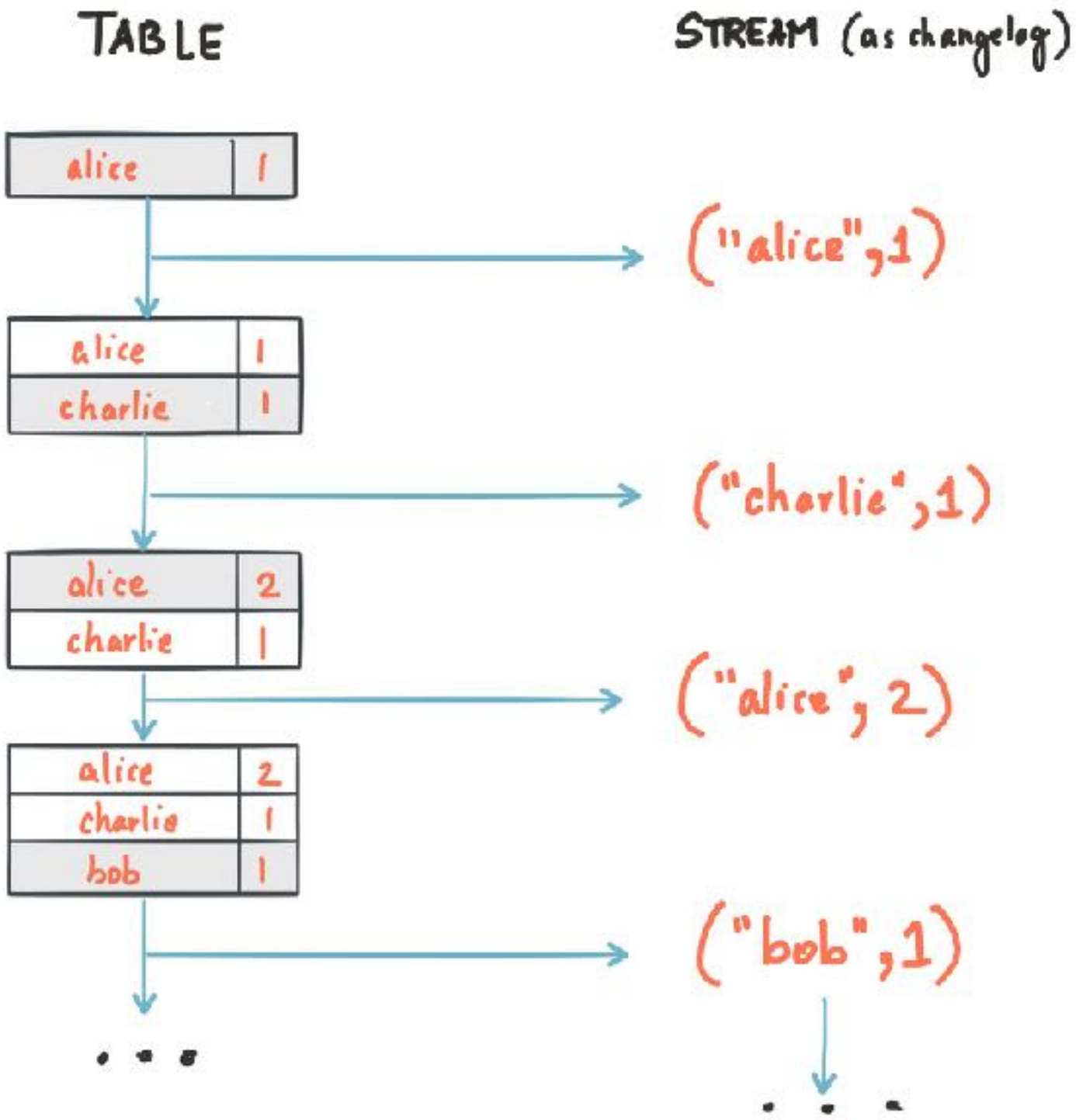
Code Sample

```
1.      public static void main(final String[] args) throws Exception {
2.              Properties config = new Properties();
3.              config.put(StreamsConfig.APPLICATION_ID_CONFIG, "simple-application");
4.              config.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "10.46.2.91:9092,10.46.2.92:9092,10.46.2.93:9092");
5.              config.put(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass());
6.              config.put(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass());
7.              config.put(StreamsConfig.DEFAULT_TIMESTAMP_EXTRACTOR_CLASS_CONFIG, WallclockTimestampExtractor.class);
8.
9.
10.             KStreamBuilder builder = new KStreamBuilder();
11.             KStream<String, String> messageStream = builder.stream("testcorp.datapoints");
12.             messageStream.filter(new Predicate<String, String>() {
13.
14.                     @Override
15.                     public boolean test(String key, String value) {
16.                             return (key != null && key.contains("Power"));
17.                     }
18.             }).foreach(new ForeachAction<String, String>() {
19.
20.                     @Override
21.                     public void apply(String key, String value) {
22.                             System.out.println("Key: " + key);
23.                             System.out.println("Value: " + value);
24.                     }
25.
26.             });
27.
28.             KafkaStreams streams = new KafkaStreams(builder, config);
29.             streams.start();
30.
31.             Runtime.getRuntime().addShutdownHook(new Thread(streams::close));
32.     }
```

Code Sample

```
1.      public static void main(final String[] args) throws Exception {
2.              Properties config = new Properties();
3.              config.put(StreamsConfig.APPLICATION_ID_CONFIG, "simple-application");
4.              config.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "10.46.2.91:9092,10.46.2.92:9092,10.46.2.93:9092");
5.              config.put(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass());
6.              config.put(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass());
7.              config.put(StreamsConfig.DEFAULT_TIMESTAMP_EXTRACTOR_CLASS_CONFIG, WallclockTimestampExtractor.class);
8.
9.              KStreamBuilder builder = new KStreamBuilder();
10.             KStream<String, String> messageStream = builder.strean("testcorp.datapoints");
11.             messageStream.filter(new Predicate<String, String>() {
12.
13.                     @Override
14.                     public boolean test(String key, String value) {
15.                             return (key != null && key.contains("Power"));
16.                     }
17.             }).to("aNewTopic");
18.
19.             KafkaStreams streams = new KafkaStreams(builder, config);
20.             streams.start();
21.
22.             Runtime.getRuntime().addShutdownHook(new Thread(streams::close));
23.     }
```

Stream & Tables



from confluent.io

Stream & Tables

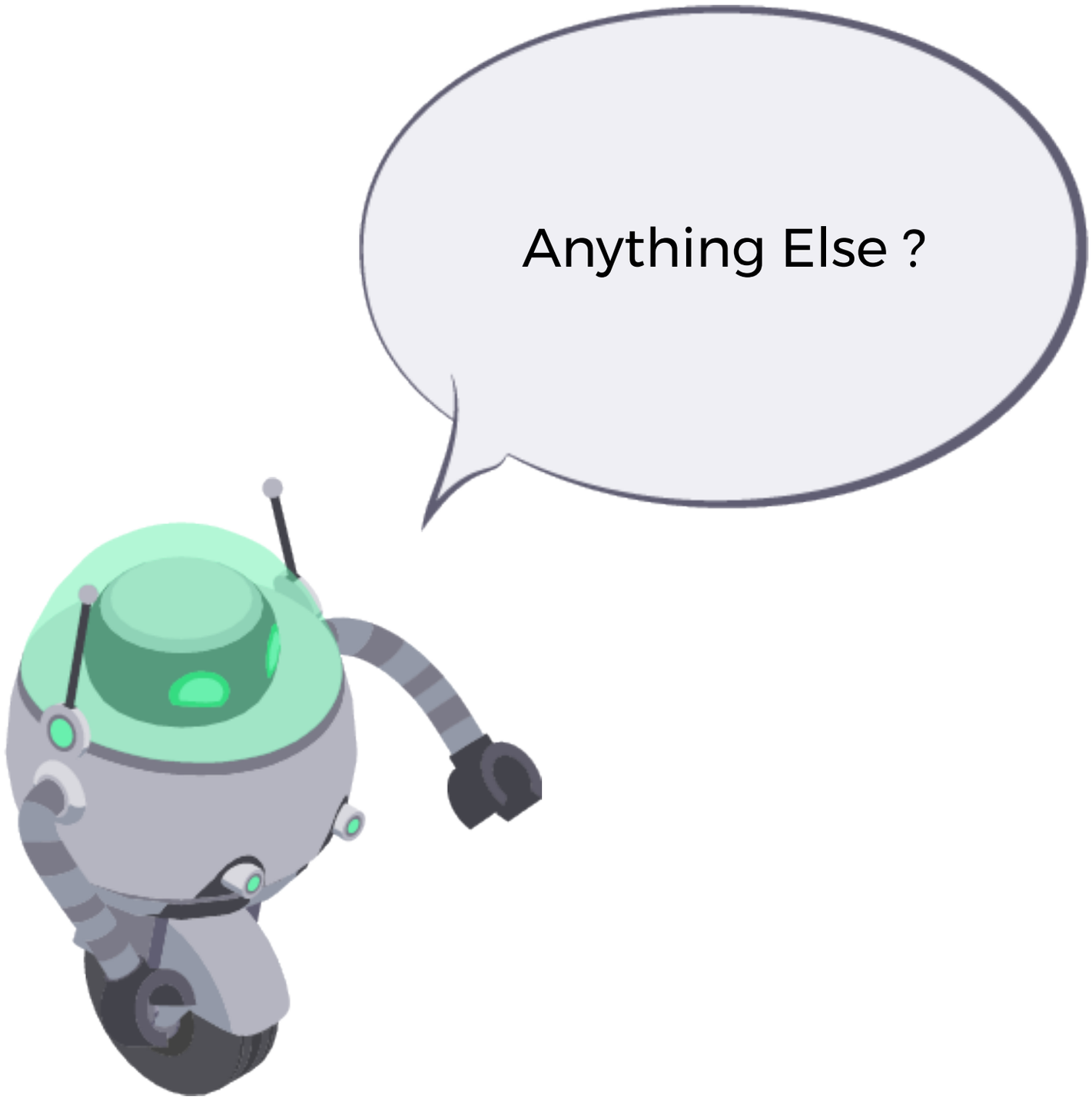
```
1.      public static void main(final String[] args) throws Exception {
2.              Properties config = new Properties();
3.              config.put(StreamsConfig.APPLICATION_ID_CONFIG, "simple-application");
4.              config.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "10.46.2.91:9092,10.46.2.92:9092,10.46.2.93:9092");
5.              config.put(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass());
6.              config.put(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass());
7.              config.put(StreamsConfig.DEFAULT_TIMESTAMP_EXTRACTOR_CLASS_CONFIG, WallclockTimestampExtractor.class);
8.
9.              KStreamBuilder builder = new KStreamBuilder();
10.             KStream<String, String> messageStream = builder.stream("testcorp.datapoints");
11.             KTable<String, String> ktable = builder.table("sensor-location");
12.
13.             messageStream.leftJoin(ktable, new ValueJoiner<String, String, String>() {
14.
15.                     @Override
16.                     public String apply(String value1, String value2) {
17.                             return value1 + " measured at " + value2;
18.                     }
19.             }).to("aNewTopic");
20.
21.             KafkaStreams streams = new KafkaStreams(builder, config);
22.             streams.start();
23.
24.             Runtime.getRuntime().addShutdownHook(new Thread(streams::close));
25.     }
```

from confluent.io

Stream & Tables

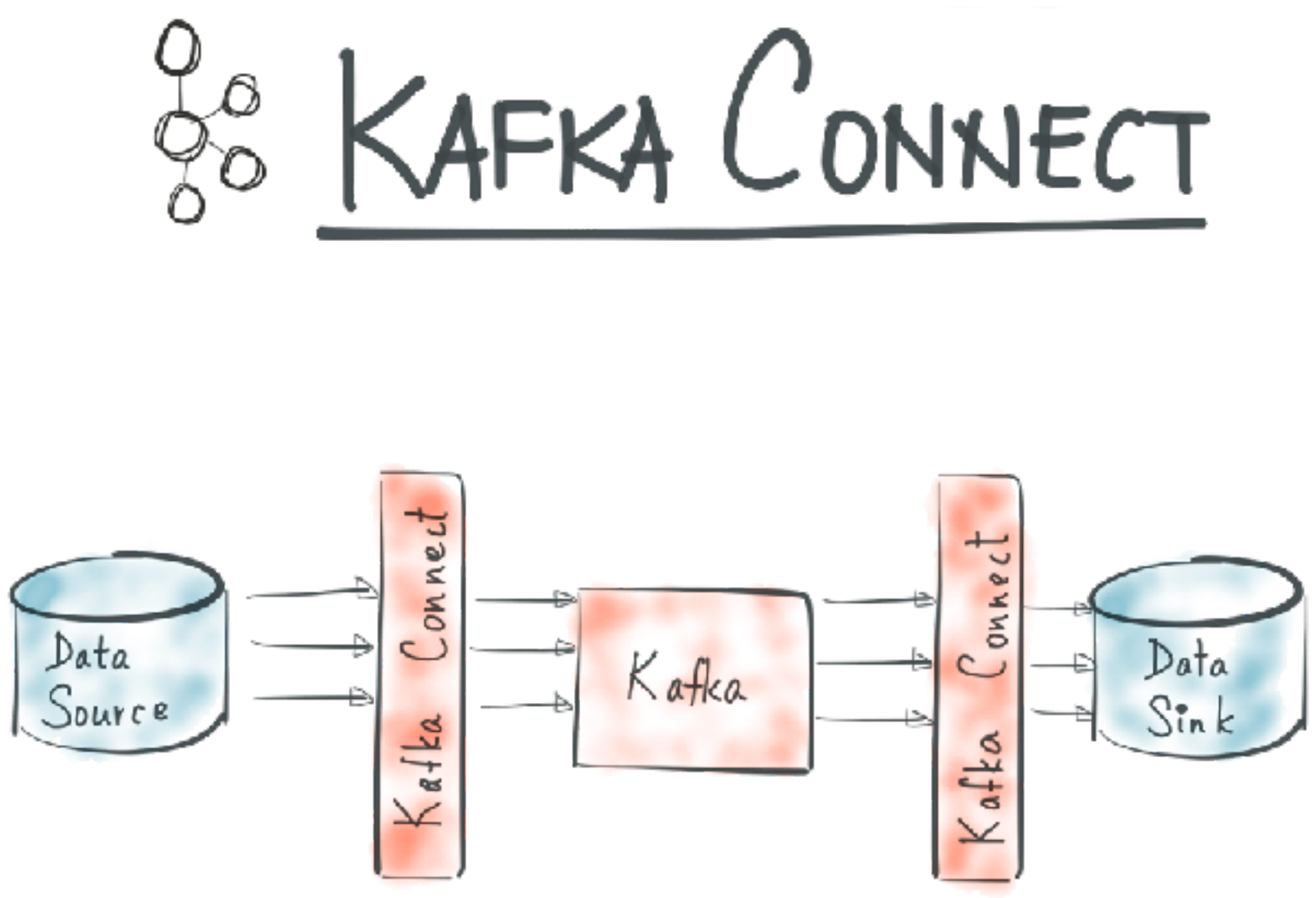
GREENCOM NETWORKS

```
1.      @Override
2.    protected KStreamBuilder kStreamBuilder( ) {
3.        final KStreamBuilder builder = new KStreamBuilder( );
4.        builder.globalTable( getCustomerTopic( "endpoints" ), STATE_STORE_NAME );
5.        return builder;
6.    }
7.
8.    private Endpoint retrieveEndpointFromStore( KafkaStreams stateStore, String uri ) {
9.        if( stateStore == null )
10.           return null;
11.       try {
12.           try {
13.               Object jsonEdp = stateStore
14.                       .store( EntitiesStateStore.STATE_STORE_NAME, QueryableStoreTypes.keyValueStore( ) ).get( uri );
15.               return gson.fromJson( jsonEdp.toString( ), Endpoint.class );
16.           } catch( NullPointerException npe ) {
17.               populateGlobalTable( eibp, EndpointLink.fromURI( uri ) );
18.           }
19.       } catch( Exception ex ) {
20.           log.warn( ExceptionUtils.getStackTrace( ex ) );
21.       }
22.       return null;
23.   }
```



Connect

- s3
- Elasticsearch
- HDFS
- JDBC
- SAP HANA
- Cassandra
- CoAP
- FTP
- Github
- InfluxDB
- MongoDB
- MQTT
- Twitter
- …
-



https://www.confluent.io/product/connectors/

...

Exactly once semantic
Message headers

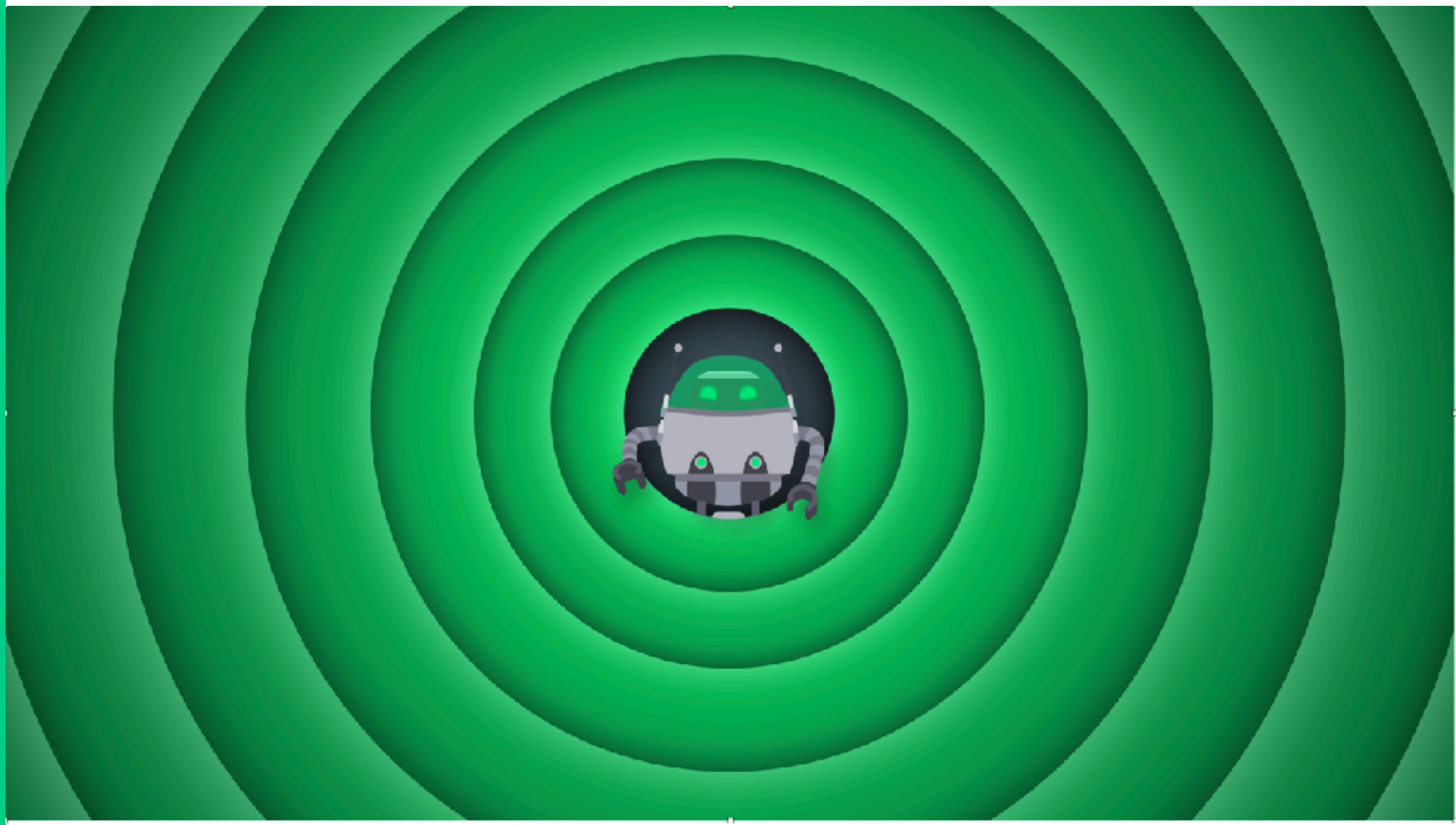
KSQL

KSQL is an open source streaming SQL engine for Apache Kafka.

```
CREATE TABLE possible_fraud AS
SELECT card_number, count(*)
FROM authorization_attempts
WINDOW TUMBLING (SIZE 5 SECONDS)
GROUP BY card_number
HAVING count(*) > 3;
```

```
CREATE STREAM vip_actions AS
SELECT userid, page, action
FROM clickstream c
LEFT JOIN users u ON c.userid = u.user_id
WHERE u.level = 'Platinum';
```

GREENCOM NETWORKS



Oh… Wait…

How does all of this scale/survive ?

From pet to cattle

✓ scales

✓ survives

✗ scales

✗ survives

✓ scales

✓ survives

✓ scales

✓ survives

✓ scales

✓ survives

Misc Stuff

✗ scales

✗ survives

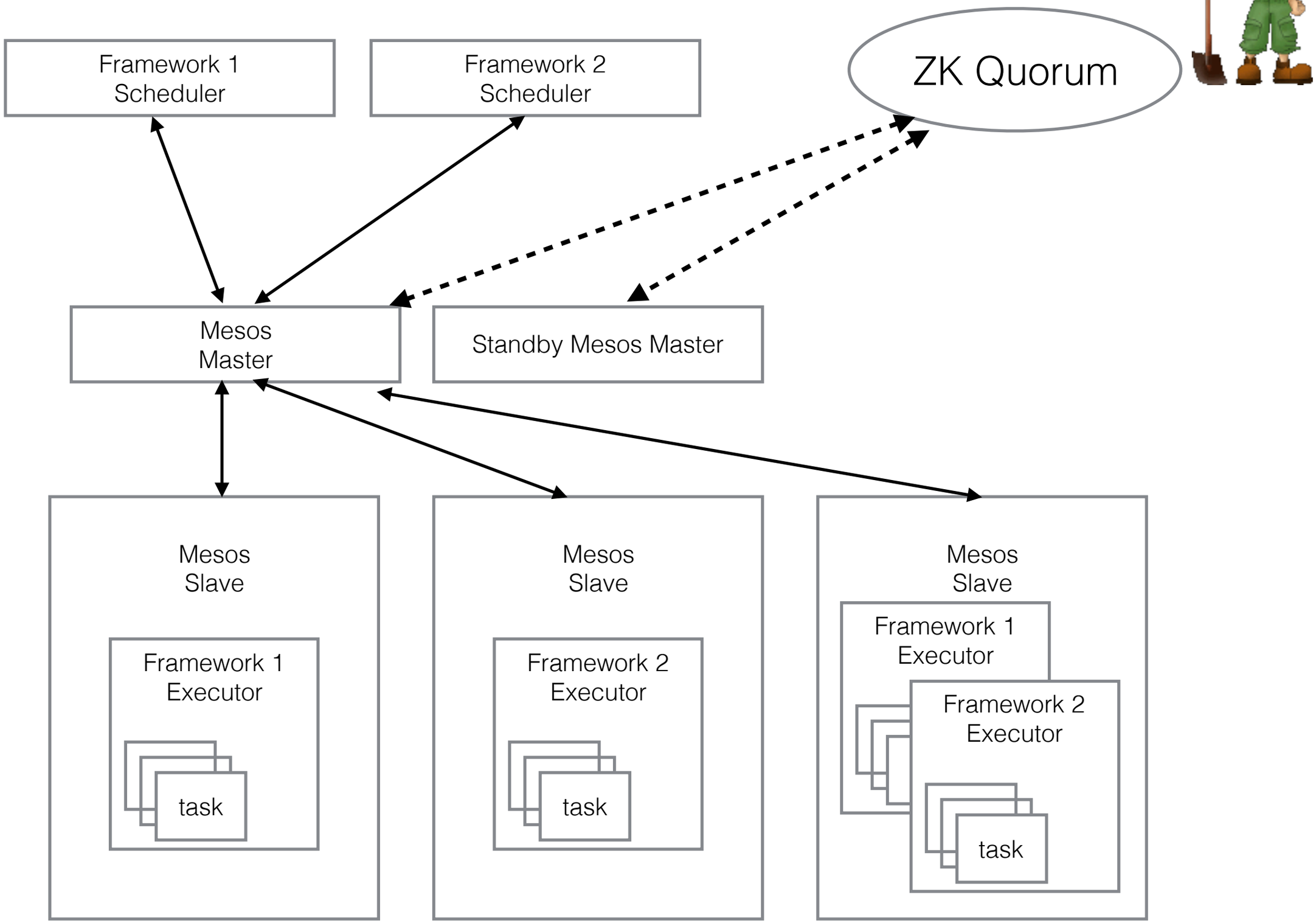
✓ scales

✓ survives

MESOS

Apache Mesos is an open-source cluster manager. It "provides efficient resource isolation and sharing across distributed applications, or frameworks". The software enables resource sharing in a fine-grained manner, improving cluster utilization.

Mesos architecture



Full abstraction of physical hardware or virtual machine.
Data center is view as a pool of resources

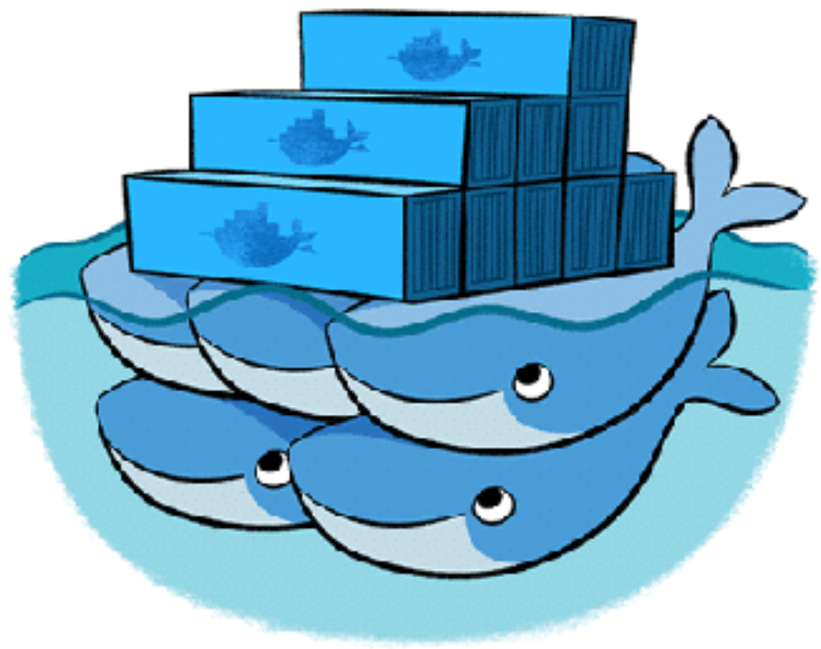
Resources

	CPUs	GPUs	Mem	Disk
Total	120	0	552.3 GB	32.8 TB
Allocated	94.7	0	413.9 GB	323.4 GB
Offered	0	0	0 B	0 B
Idle	25.3	0	138.4 GB	32.5 TB

Frameworks negotiate with mesos masters to get access to resources

Competition

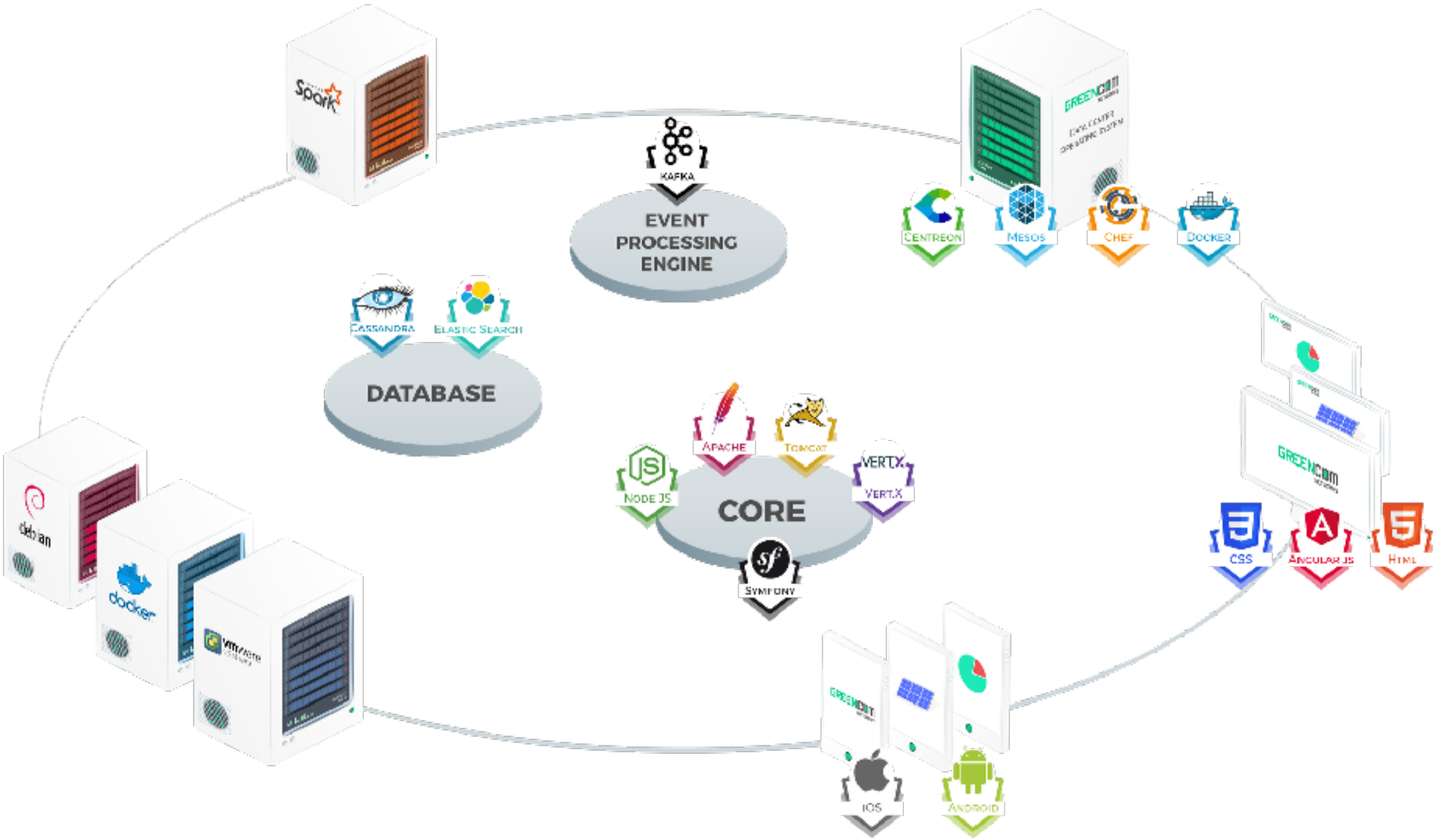
kubernetes



Apache AURORA™

nextGen Platform
aka
EIBP Platform V3

EIBP Platform V3



Our stack

GREENCOM NETWORKS

Few of the main components

- **Cassandra**: open source NoSQL distributed database management system (brings high availability with no single point of failure; scales horizontally). *Originally developed by Facebook.*

- **Kafka**: open source message broker (brings a unified, high-throughput, low-latency platform for handling real-time data feeds). *Originally developed by LinkedIn*

- **Mesos**: open source cluster manager (brings resource sharing capabilities in a highly distributed environment) .*Originally developed at the University of California, Berkeley. Used by Twitter, Airbnb, Apple*

- **Spark**: open source cluster computing framework (brings high performance in memory data processing capabilities). *Originally developed at UC Berkeley*

Some figures:

Production:

Cassandra:

- 16 nodes
- 50 millions calls/day on low level API
- ~ 40 millions datapoints/day

Kafka:

- 8 brokers
- 1700 topics / 7000 partitions
- 3500 messages/second (300 millions / day)

GREENCOM NETWORKS

micro services @GCN

GREENCOM NETWORKS

350+ services
1000+ instances
20 customers

100+ daily deployments
120 servers
2-5 minutes from commit to prod
3.5 cloud engineers

Open points/questions

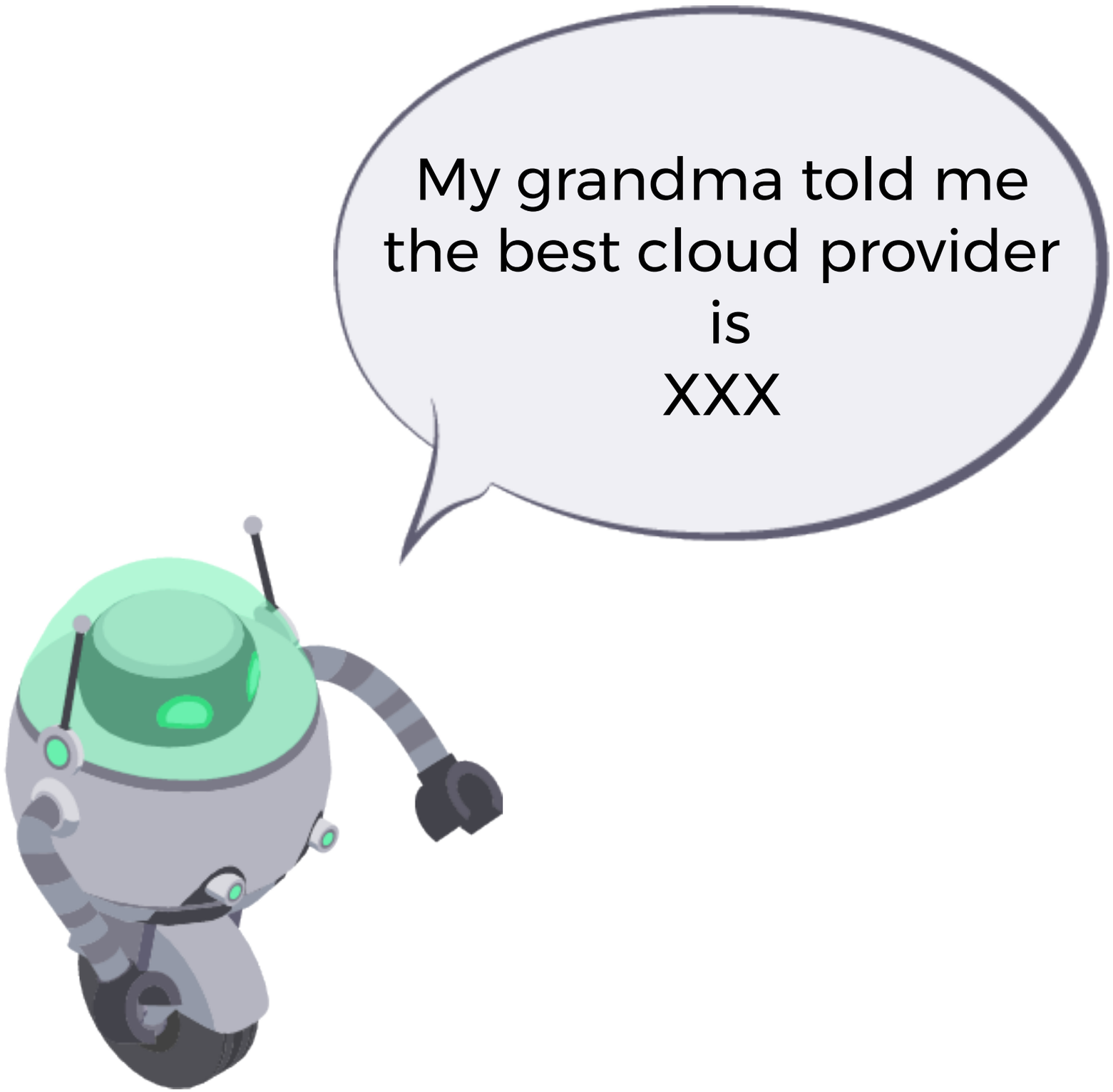
- **Monitoring**: Hard to monitor so many micro services in an highly distributed environment
- **Deployement**: Can't afford manual operations. Everything as to be automated
- **Testing/diagnostic**: micro services + loose coupling creates complexity
- **Number of clusters/split brain**: Minimum number of server is already high. Many different schedulers/cluster managers. High complexity.
- **Skillset**: Not so easy te recruit people with the right skill set

Roadmap

EIBP v3+ : What we are working on:

- Multi Cloud
- Mesos to Kubernetes migration
- Better service mesh management

GREENCOM NETWORKS



Going multi-cloud

Going multi-cloud

Going multi-cloud

Kubernetes

GREENCOM NETWORKS

Service mesh management

· Zero Downtime deployment	✓
· Dark Launch	✓
· Logging	✓
· A/B Testing	✓
· API Management	✓
· Tracing	✓
· Canary Release	✗
· Blue/Green Deployment	✗

GREENCOM NETWORKS

kubernetes

Container orchestrator

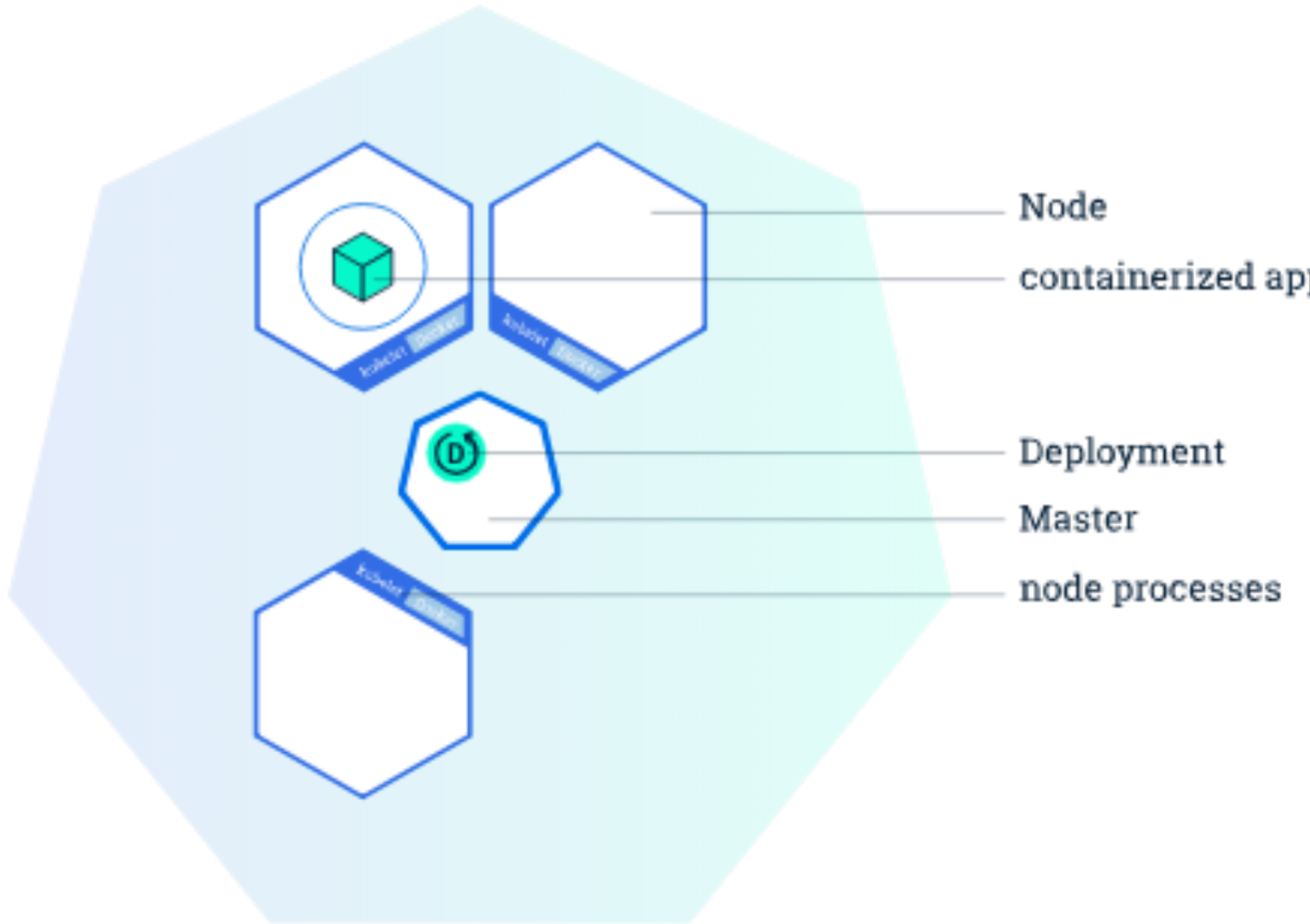
CLOUD NATIVE COMPUTING FOUNDATION

Open Source
Industry choice
Next big thing
Container centric
Inspired by Google's Borg
Pretty damn cool

GREENCOM NETWORKS

KUBERNETES 101.

kubernetes

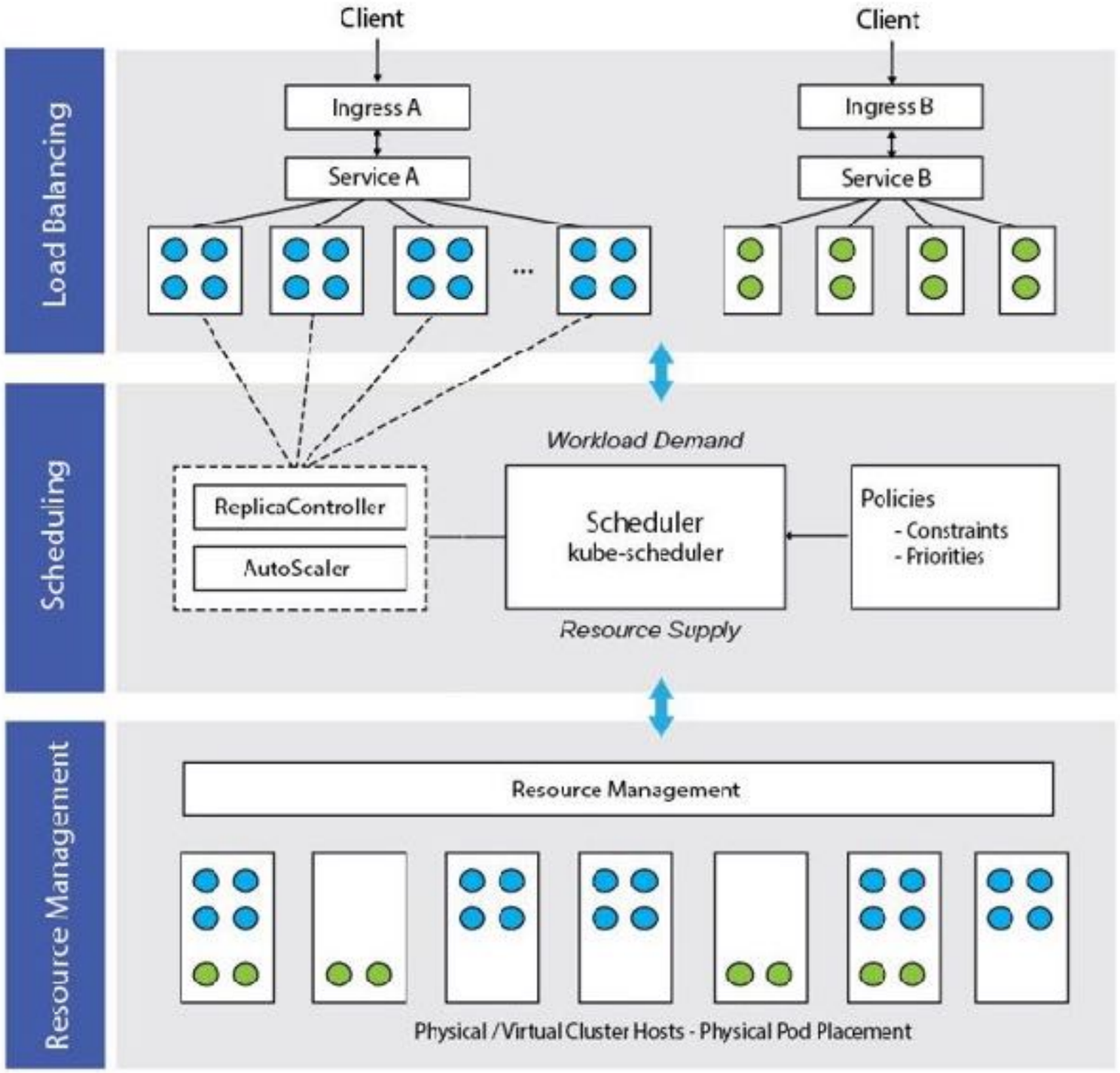


Kubernetes Cluster

GREENCOM NETWORKS

KUBERNETES 101.

kubernetes



- **Availability - Networking**
- **Lifecycles handling**
- **Computation resources**

GREENCOM NETWORKS

KUBERNETES 101.

kubernetes

```yaml
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: python-basic-api
  labels:
    app: pba
spec:
  replicas: 2
  selector:
    matchLabels:
      app: pba
  template:
    metadata:
      labels:
        app: pba
    spec:
      containers:
        - name: python-basic-container
          image: murrau/pba:ready
          ports:
            - containerPort: 8000
          imagePullPolicy: Always

---
apiVersion: v1
kind: Service
metadata:
  name: pba-service
  labels:
    app: pba
spec:
  selector:
    app: pba
  type: ClusterIP
  ports:
    - protocol: TCP
      name: http
      port: 80
      targetPort: 8000
```

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: basic-apis-ingress
  annotations:
    kubernetes.io/ingress.class: "nginx"
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: rnd.gcn-eibp.com
    http:
      paths:
      - path: /nba
        backend:
          serviceName: nba-service
          servicePort: 80
      - path: /pba
        backend:
          serviceName: pba-service
          servicePort: 80
```


GREENCOM NETWORKS

kubernetes

Am I writing

~300apps*15customers*4yamls ?

(PS: = 18000 files)

GREENCOM NETWORKS

kubernetes

Am I writing ~300apps*15customers*4yamls ?

(PS: = 18000 files)

Also....

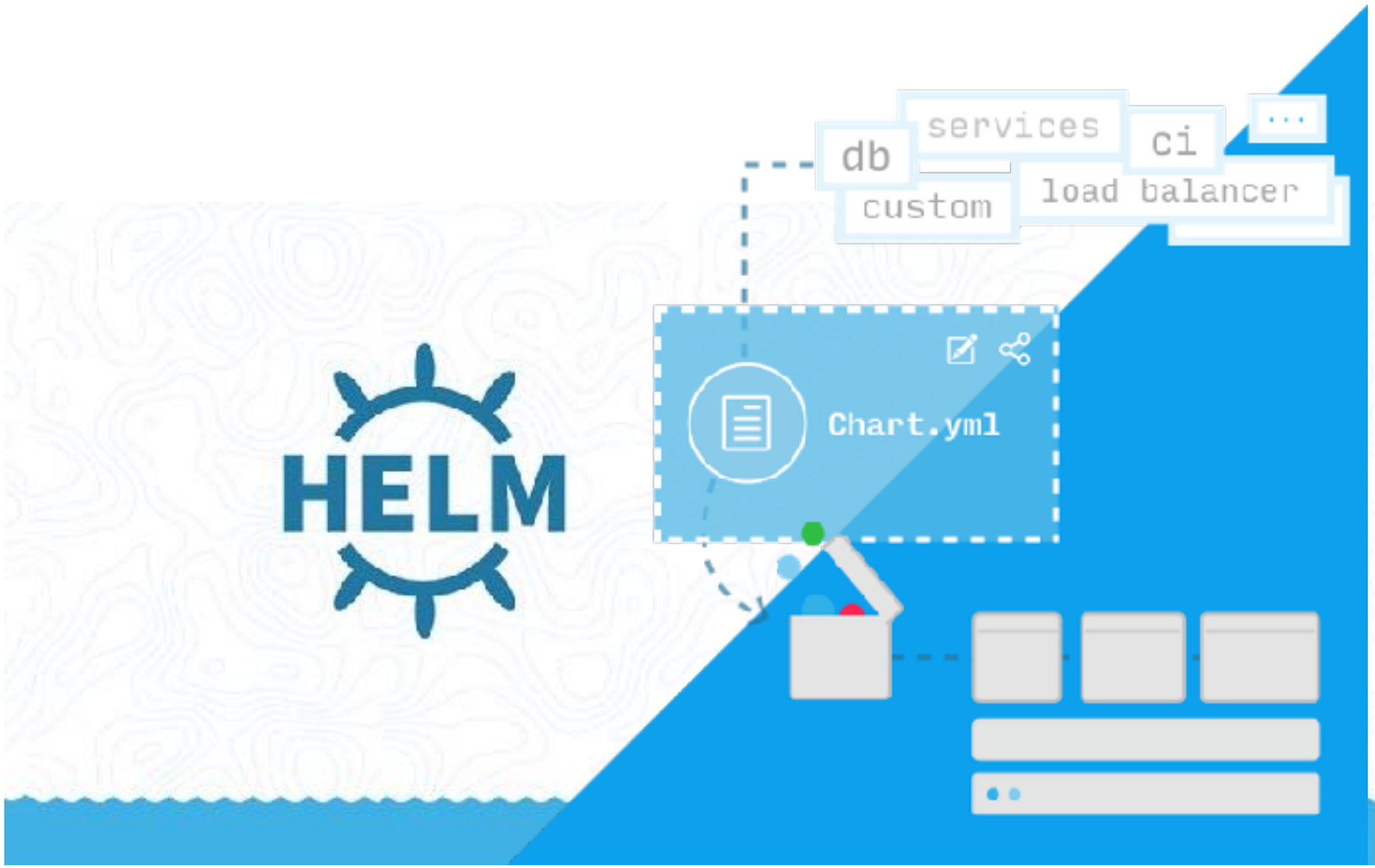
Less code

↓

Less mistakes, less problems, more fun

GREENCOM NETWORKS

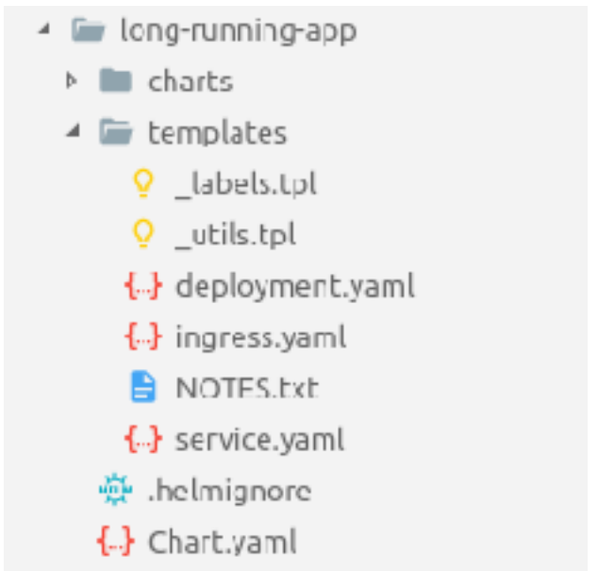
KUBERNETES 101.



GREENCOM NETWORKS

KUBERNETES 101.

kubernetes



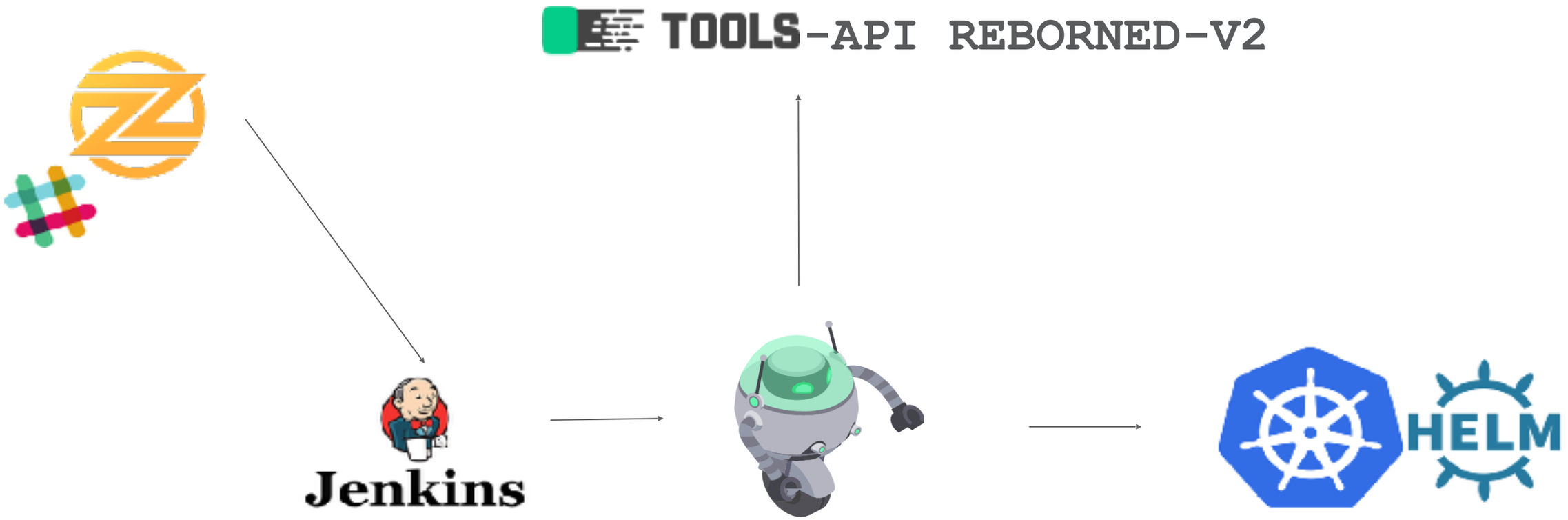

```yaml
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: {{ .Values.app }}
  labels: {{ include "labels.full" . | indent 2 }}
spec:
  replicas: {{ .Values.replicaCount }}
  selector:
    matchLabels: {{ include "labels.short" . | indent 4 }}
  template:
    metadata:
      labels: {{ include "labels.short" . | indent 6 }}
    spec:
      containers:
        - name: {{ .Values.app }}
          image: {{ .Values.registry.host }}/{{ .Values.registry.repository }}/{{ .Values.app }}:{{ .Values.tag }}

          {{- if .Values.exposed }}
          ports:
            - name: http
              containerPort: {{ .Values.service.targetPort }}
              protocol: TCP
          {{- end }}

          {{- if .Values.resources }}
          resources:
            {{- if .Values.resources.limits }}
            limits:
              cpu: {{ .Values.resources.limits.cpu }}
              memory: {{ .Values.resources.limits.memory }}
            {{- end }}
          {{- end }}
```

ONE CHART TO RULE THEM ALL

GREENCOM NETWORKS



UPDATED (simplified) DEPLOYMENT PIPELINE

WE MADE IT.

kubernetes

PS: At that point, we haven't really moved forward though

GREENCOM NETWORKS

TIME FOR BENEFITS.

Clean pipelines

Kubernetes is *almost* all-in-one

Load Balancing + Networking = almost free

On-the-shelf solutions

One place monitoring

Plenty of other cool stuff

GREENCOM NETWORKS

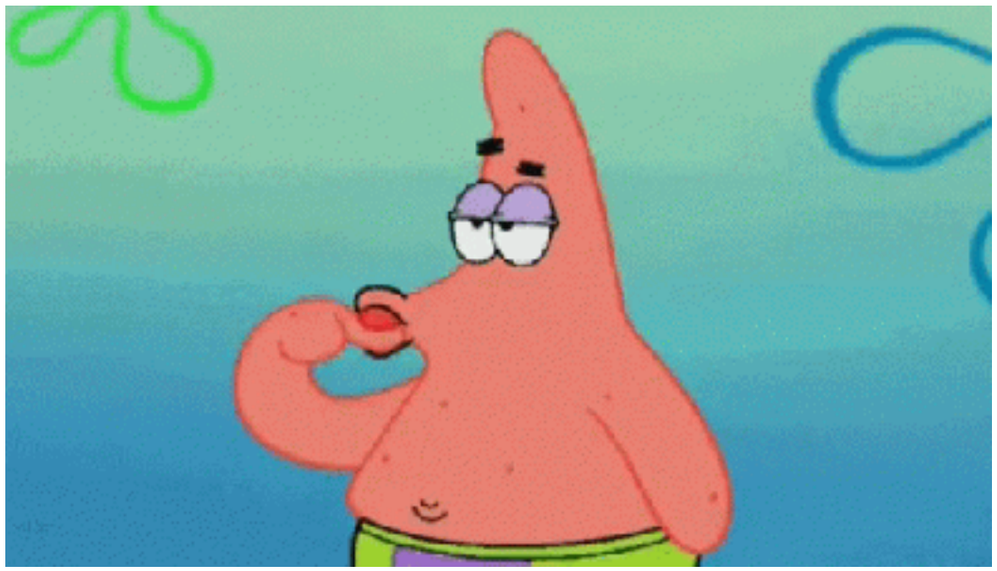
AND FOR DOWNSIDES…

***Too much* cool stuff**

Victims of industry choices

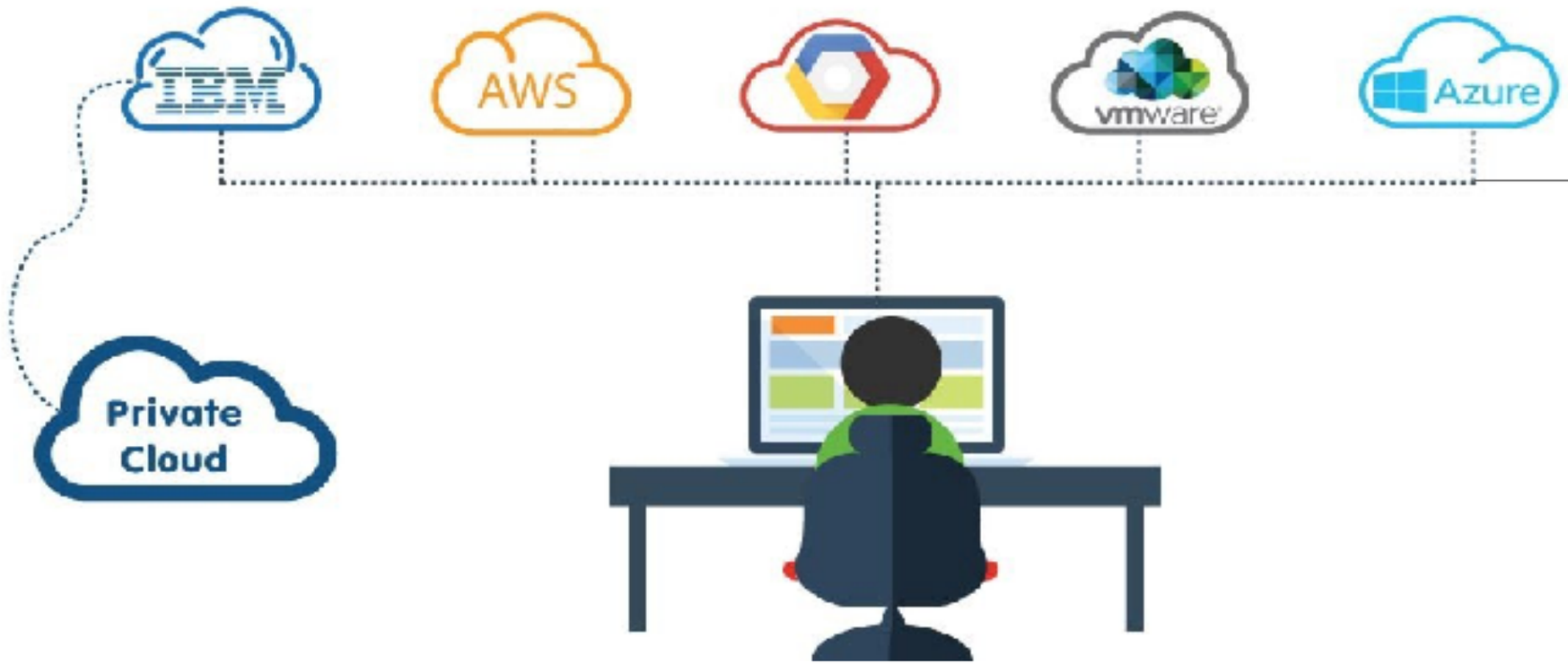
Learning curve, might be impressive

Many, many, many possibilities



WHERE IS THE CLOUD GONE?

Cloud providers were implementing their wrappers around Kubernetes.



Kubernetes integrations on-the-shelf

And let's deal with what's remaining

LET'S GET IN THE CLOUD.

Networking between components

Providing computing resources

On-demand installation and scalability

GREENCOM NETWORKS

LET'S GET IN THE CLOUD.

AVOID VENDOR LOCKING
RE-USE AND MIRROR OUR ON-PREMS INSTALL
MANAGE EVERYTHING THE SAME WAY

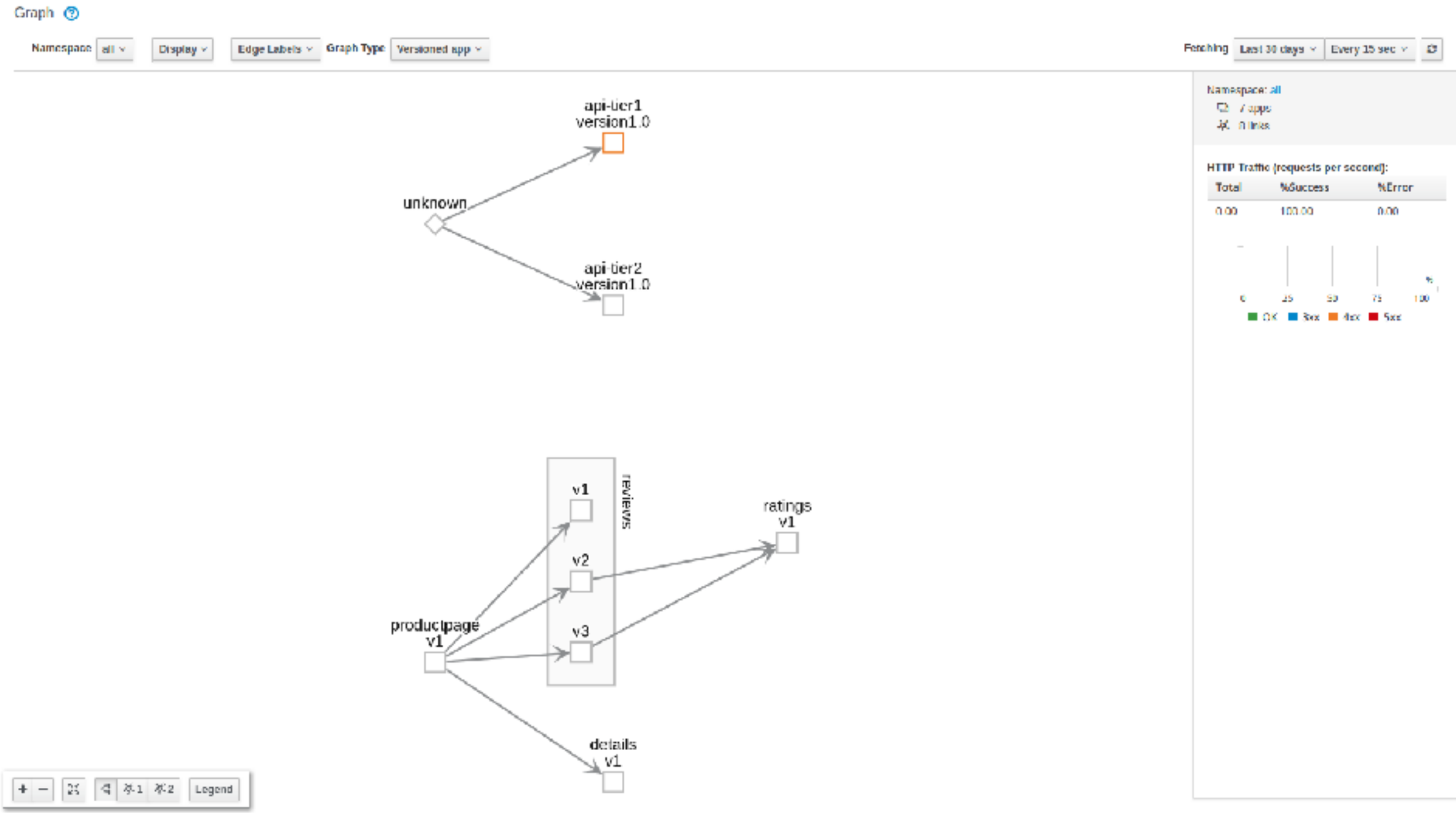
GREENCOM NETWORKS

STILL A LONG WAY TO GO.

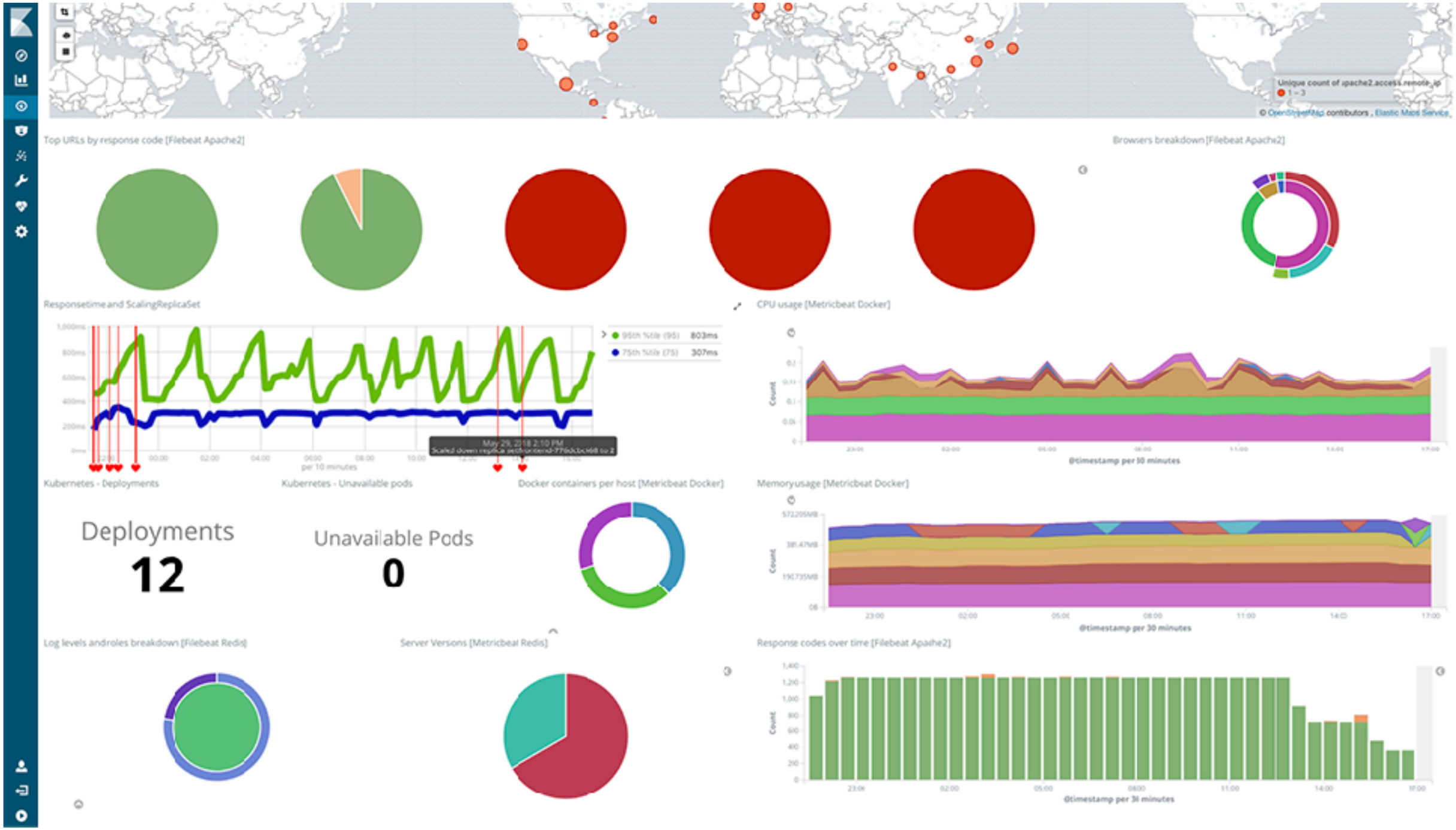
But that's the easy part

GREENCOM NETWORKS

SERVICE MESH LIVE STATUS.



COST ANALYTICS.



GREENCOM NETWORKS

AND SO ON.

Automatic Blue-Green deployment

On-demand scalable storage

Circuits breakers

Self-recovering workload

...

GREENCOM NETWORKS

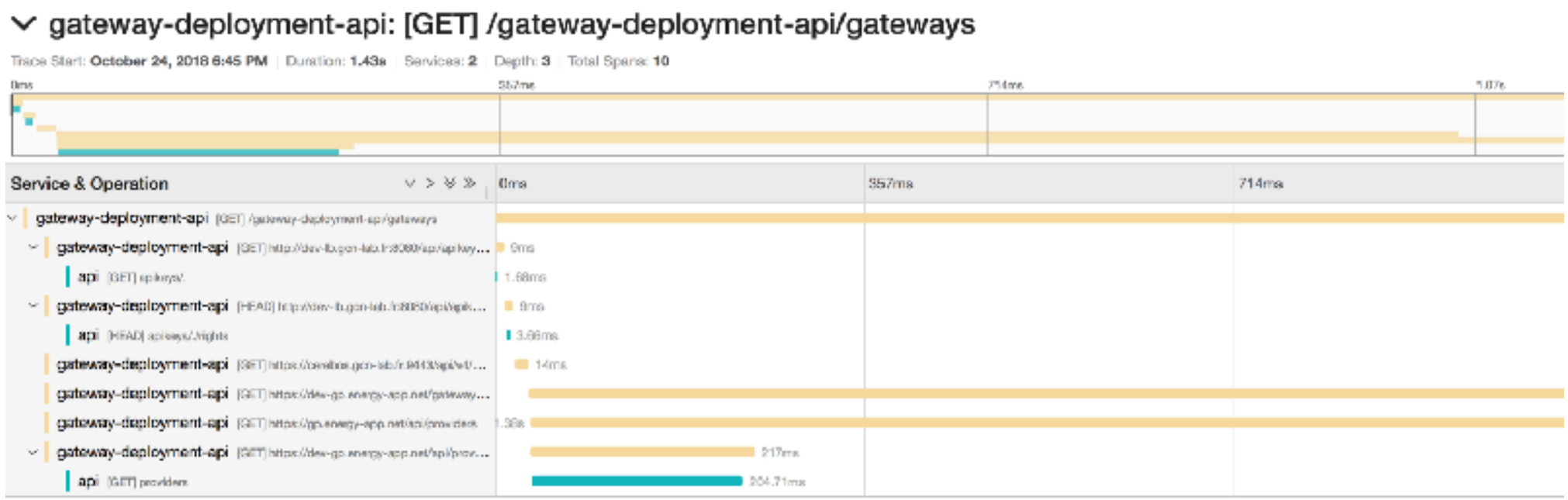
Service mesh management

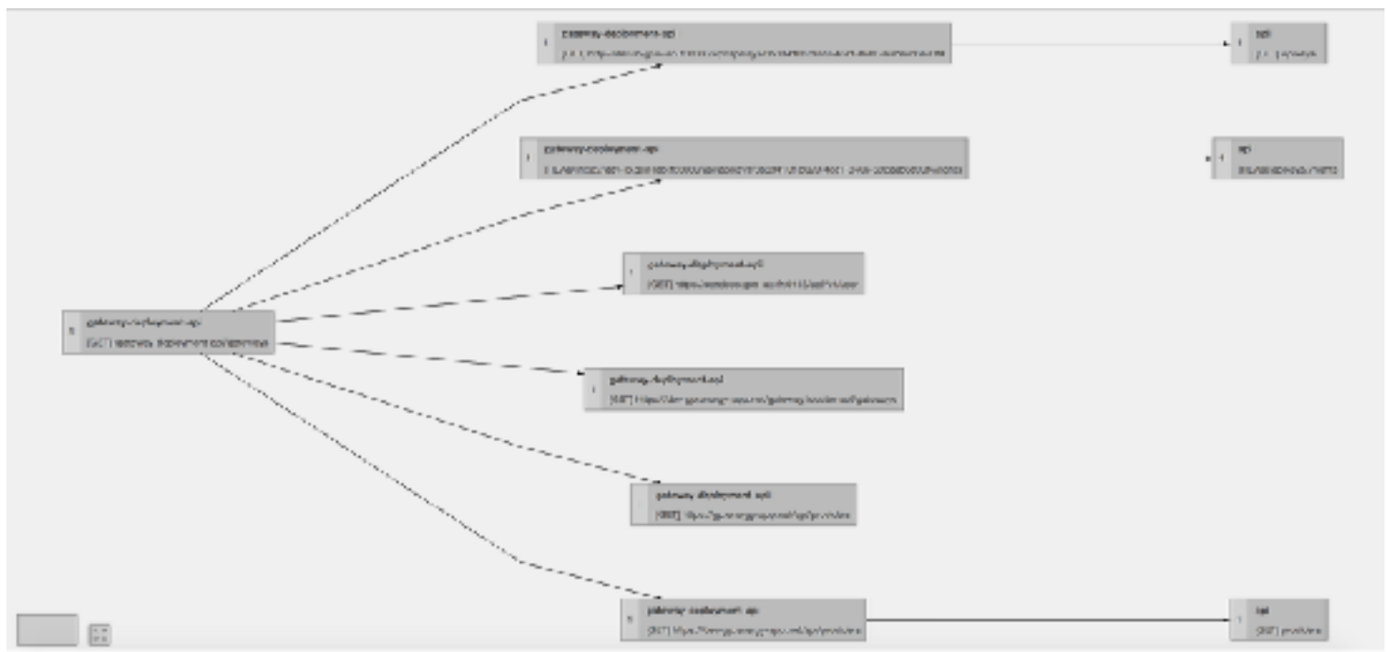
· Zero Downtime deployment	✓
· Dark Launch	✓
· Logging	✓
· A/B Testing	✓
· API Management	✓
· Tracing	✓
· Canary Release	✓
· Blue/Green Deployment	✓

GREENCOM NETWORKS

Tracing

JAEGER





GREENCOM NETWORKS

You have to stop now...
They all are hungry, some of them are sleeping for the last 20 minutes...

Wrap Up

Platform Design Principles:

- Have a look to companies who are facing today volumes we will have to handle tomorrow
- Keep performance, scalability in mind
- No Single Point Of Failure
- Keep it simple / No over-engineering
- Agility is Key
- No choice if definitive

GREENCOM NETWORKS

Continuous improvement is better than delayed perfection.

Mark Twain

THAT'S IT.