Serverless IoT Applications

with Apache OpenWhisk

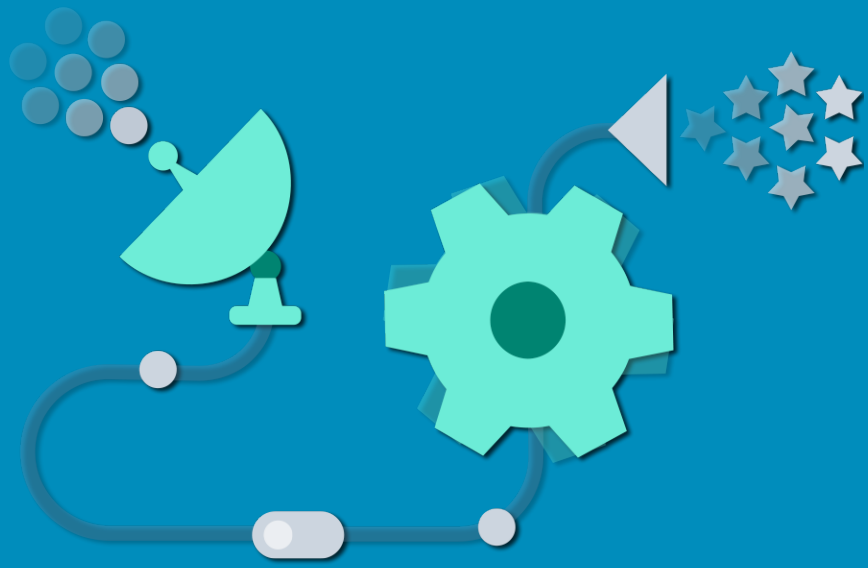
on IBM Bluemix

http://openwhisk.org/

#openwhisk

https://openwhisk-team.slack.com/

https://console.bluemix.net/openwhisk/



Alex Glikson, IBM Research

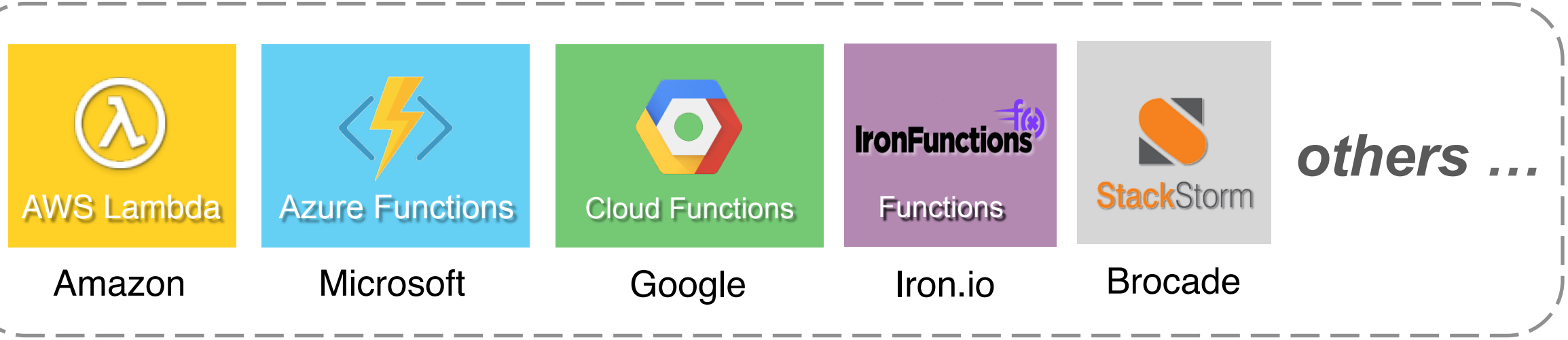
Session Topics

1. Overview of the Serverless Paradigm and Apache OpenWhisk
2. The Internet of Things (IoT) and IoT Applications
3. Serverless Application Patterns for the IoT

IBM

1. Overview of the Serverless Paradigm and Apache OpenWhisk

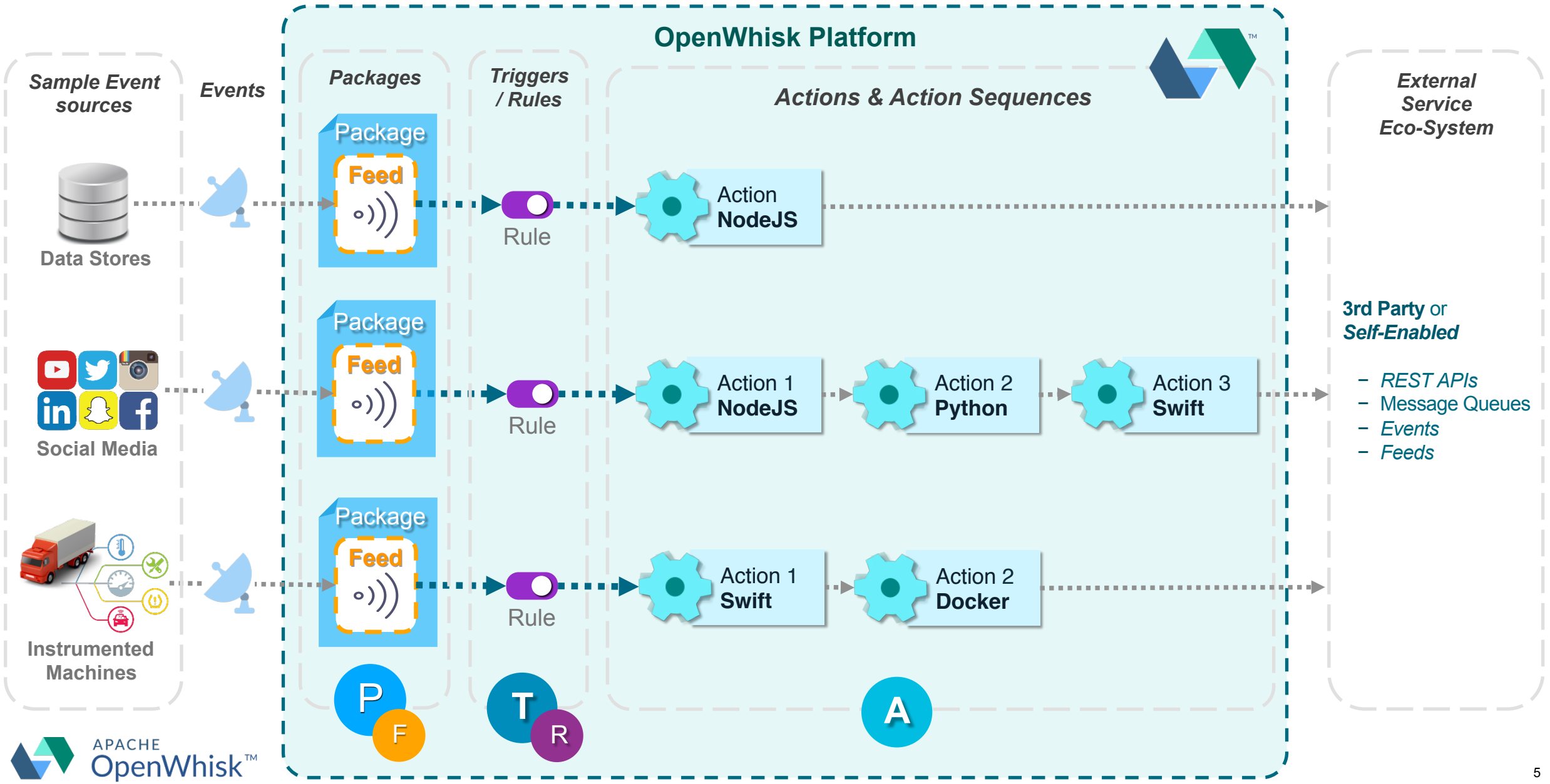
Many Cloud providers offer some form of Serverless framework



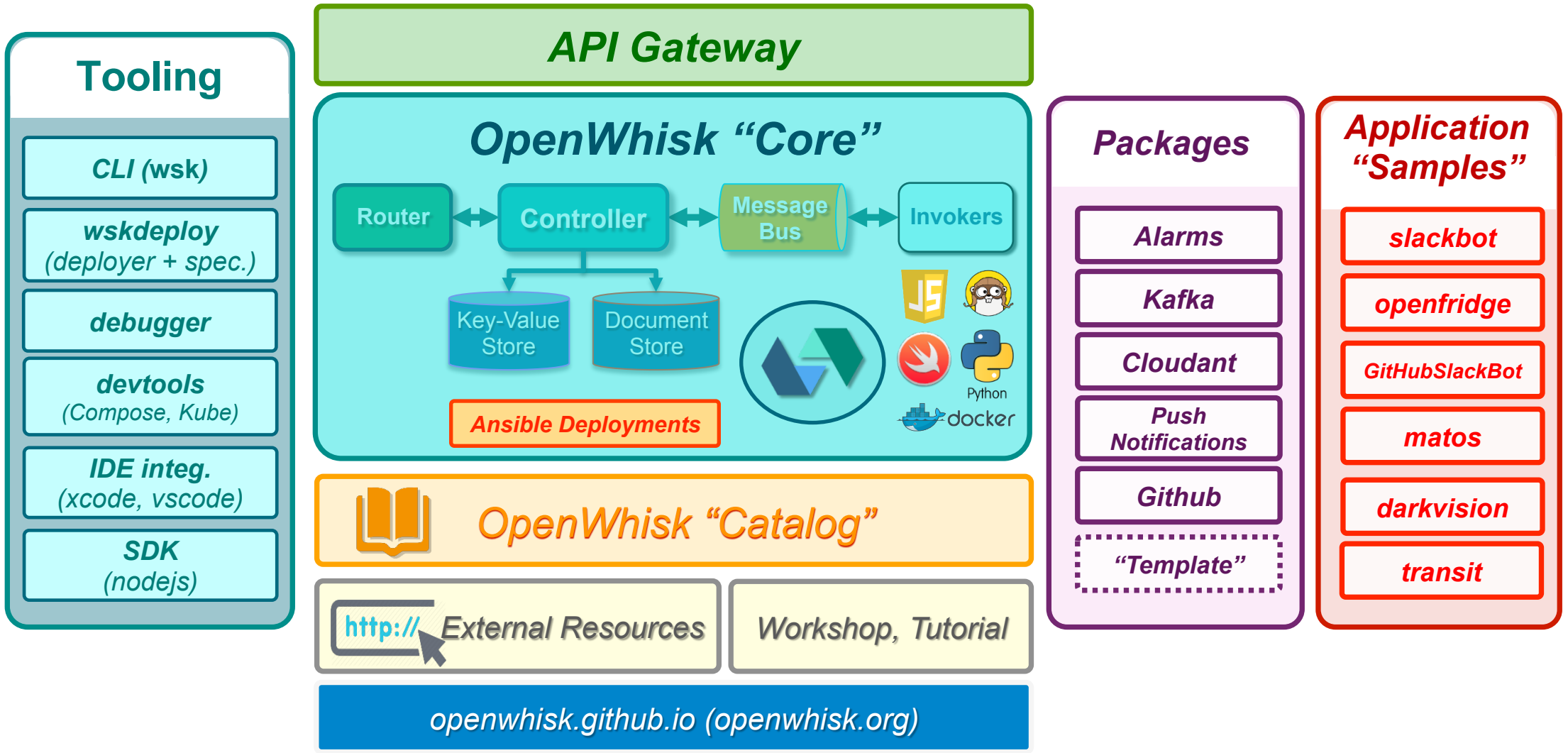
Apache OpenWhisk *offers:*

- **Apache Software Foundation** (ASF)
  - *True community-driven open source* *(Apache 2 License)*
- **Proven on IBM's Bluemix Cloud Platform**
  - *Exact same code in open source*

Apache OpenWhisk – Event-Trigger-Rule-Action Processing



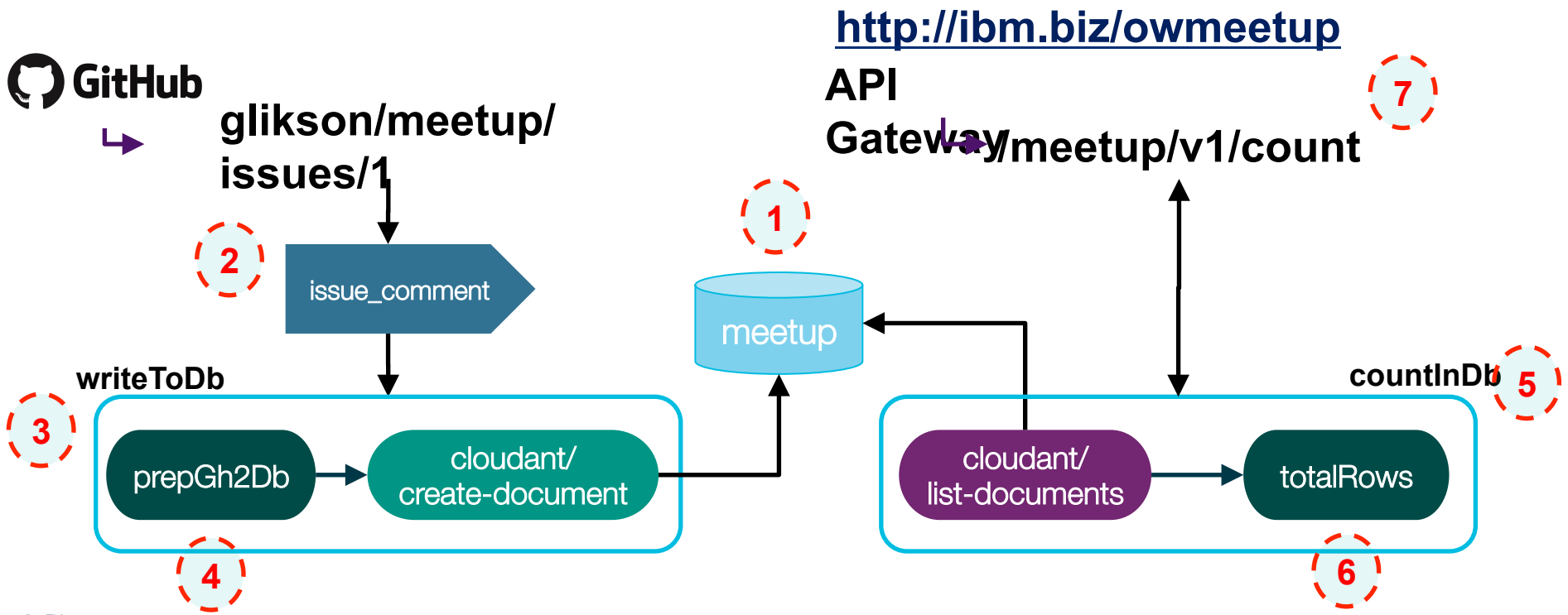
Apache OpenWhisk “Eco-System” Overview



APACHE OpenWhisk™

Example: Monitoring Github comments

- **Goals:**
  1. Watch for new comments in a given github.com repository
  2. Save github.com events in a DB (Cloudant)
  3. Provide REST API to access data in the DB (e.g., aggregate stats)



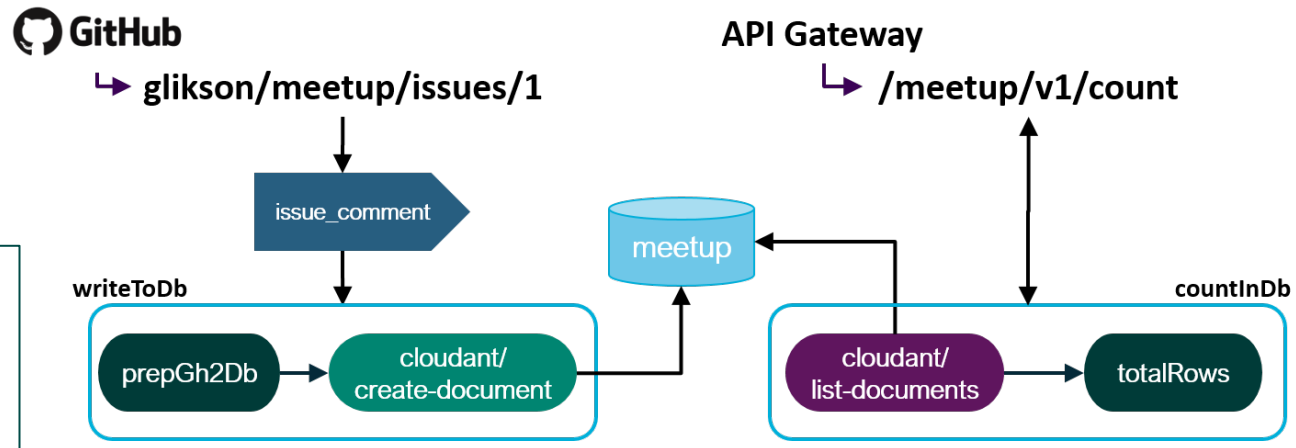
APACHE OpenWhisk™

Example: monitoring Github comments

```
// prepGh2Db.js : converts Github event payload
// to be written to Cloudant DB
// using /whisk.system/cloudant/create-document
function main(hgPayload) {
    return {
        "dbname": hgPayload.repository.name,
        "doc": hgPayload
    };
}
```

```
// totalRows.js : returns total count
// of rows in a Cloudant DB.
// should be invoked following
// cloudant/list-documents
function main(docs) {
    return { "count": docs.total_rows};
}
```

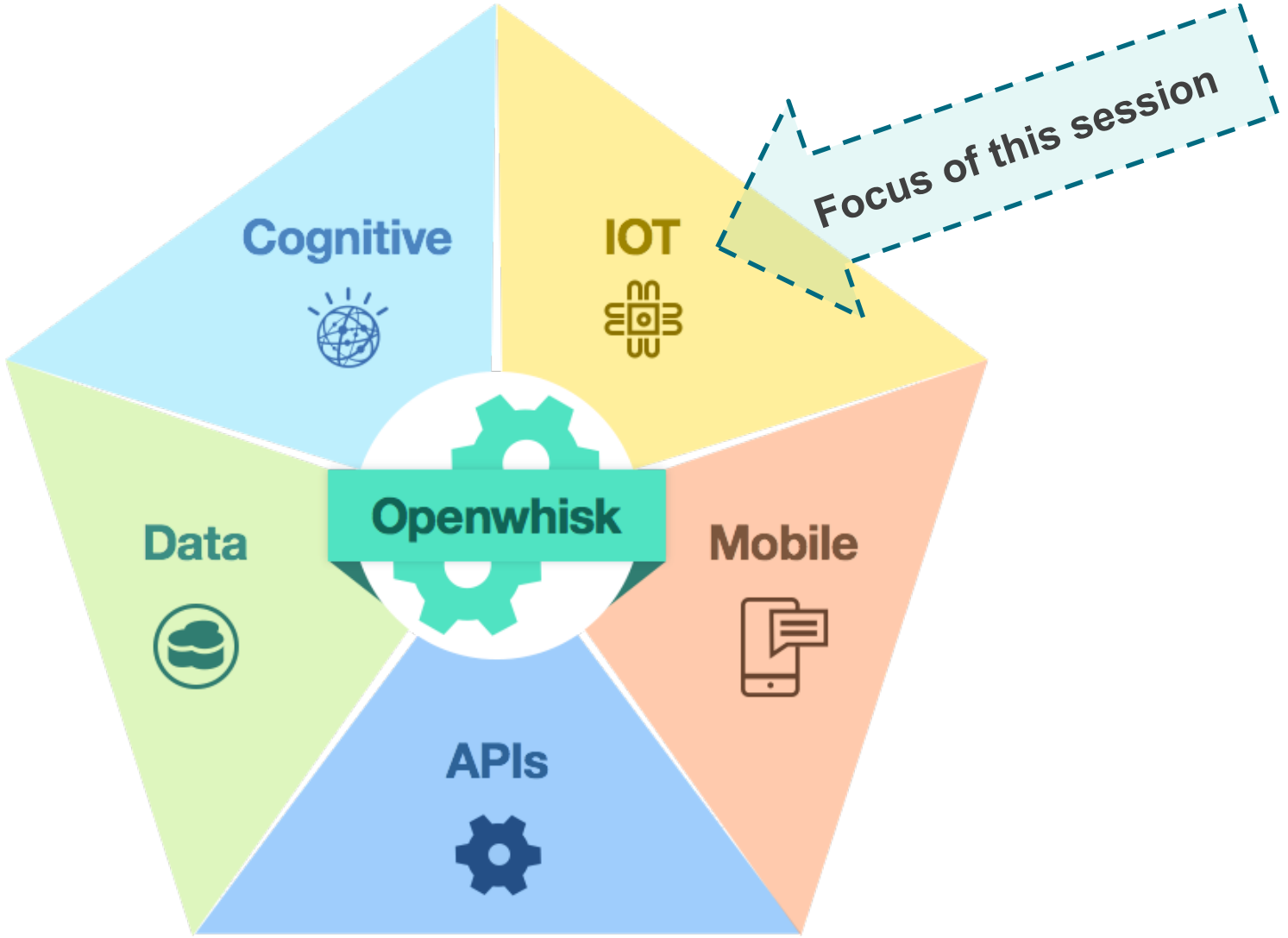
http://ibm.biz/owmeetup




```
[1] $ wsk action create --kind nodejs:6 prepGh2Db prepGh2Db.js
[2] $ wsk action create --kind nodejs:6 totalRows totalRows.js
[3] $ wsk package refresh
[4] $ wsk action create --sequence writeToDb prepGh2Db,Bluemix_db_meetup/create-document
[5] $ wsk action create --sequence countInDb Bluemix_db_meetup/list-documents,totalRows
[6] $ wsk trigger create github-meetup-issues --feed /whish.system/github/webhook -p username \
glikson -p repository meetup -p accessToken xxx -p events issue_comment
[7] $ wsk rule create gh2db github-meetup-issues writeToDb
[8] $ wsk api-experimental create /meetup/v1 /count get countInDb
[9] $ curl https://ffac6dba-9f5c-4444-gws.api-gw.mybluemix.net/meetup/v1/count?dbname=meetup
{
  "count": 9
}
```

APACHE OpenWhisk™

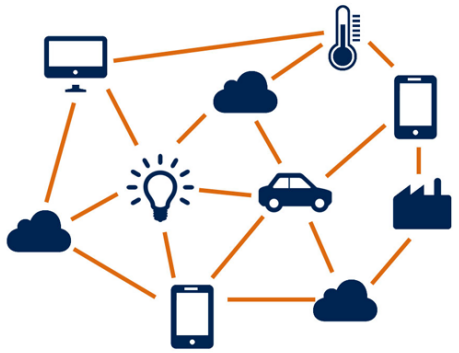
Serverless (and OpenWhisk) is applicable in many domains

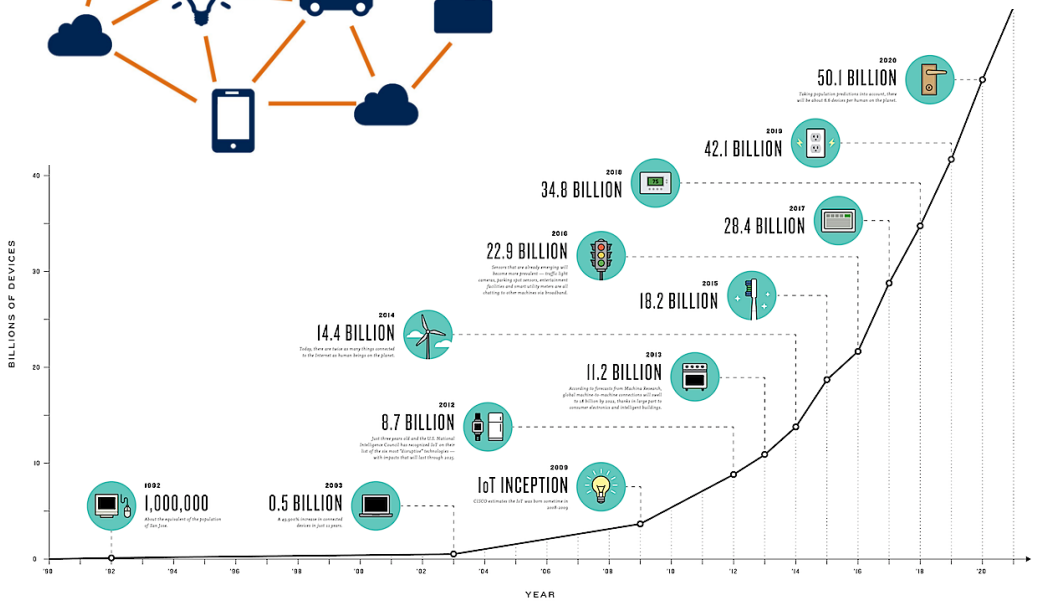


IBM

2. The Internet of Things (IoT) and IoT Applications

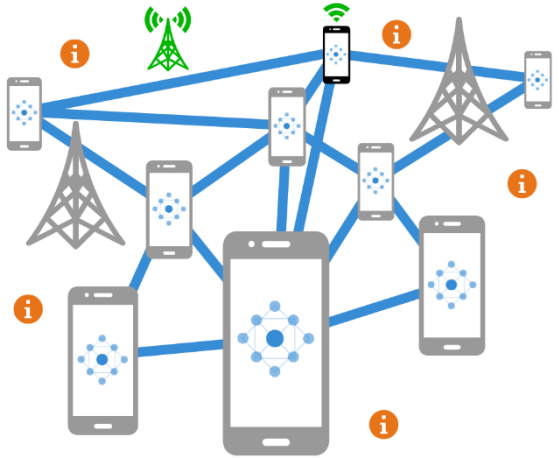
What is Internet of Things (IoT)?





- **Internet of Things (IoT) applications:**
  - Rapidly emerging application development paradigm, taking advantage of the ubiquitous **connectivity** and **programmability** of physical **devices**:
    1. Environment sensors (e.g., cities, private weather stations)
    2. Consumer electronics (e.g., wearables, home appliances)
    3. Industry (e.g., factories, agriculture, buildings)
    4. Mobility (e.g., personal navigation, connected cars)

  - Synergetic with the phenomenal adoption of **smartphones** and the rapid growth of the enabling communication infrastructure, providing **commodity** (low-cost, ubiquitous) connectivity
    – 3G, 4G, 5G …

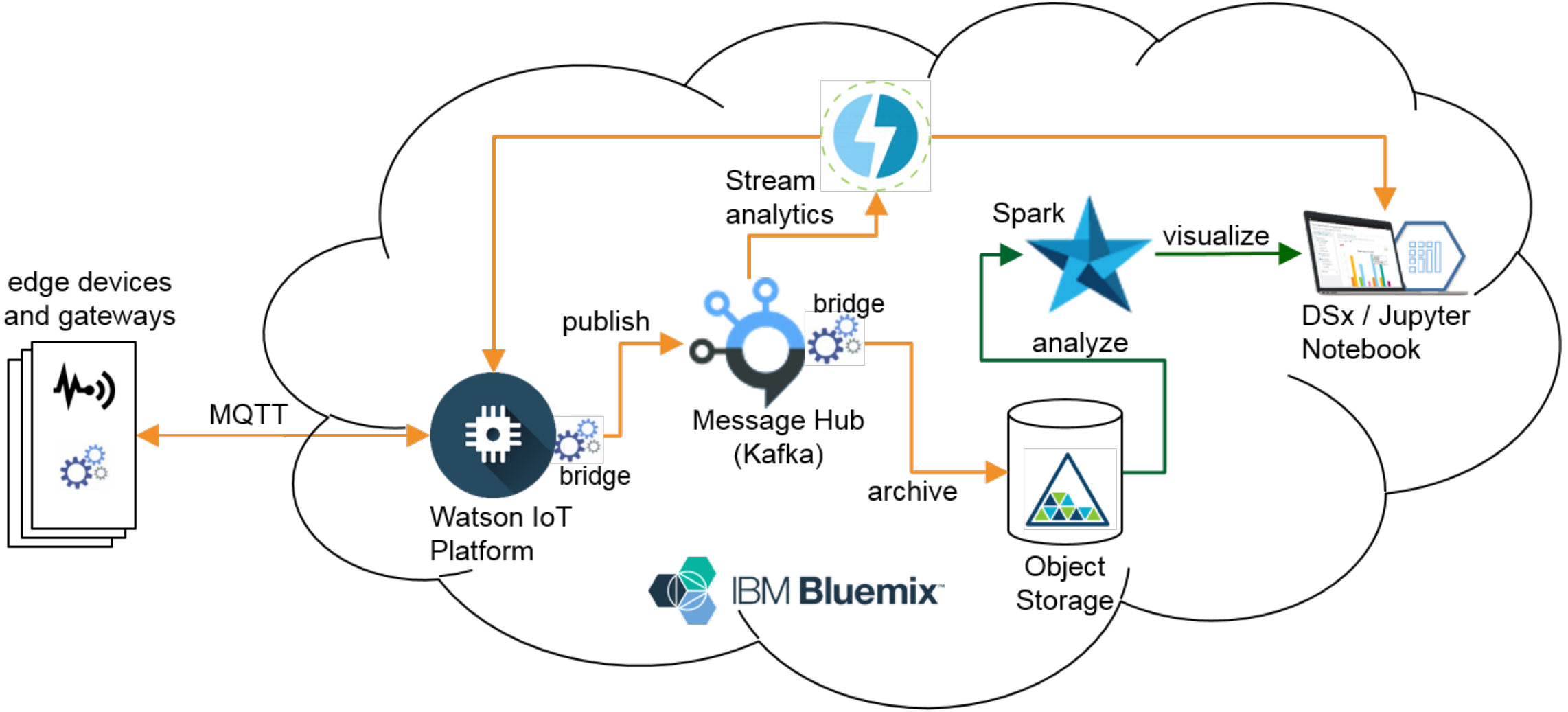


APACHE OpenWhisk™

Characteristics of a Typical IoT Application

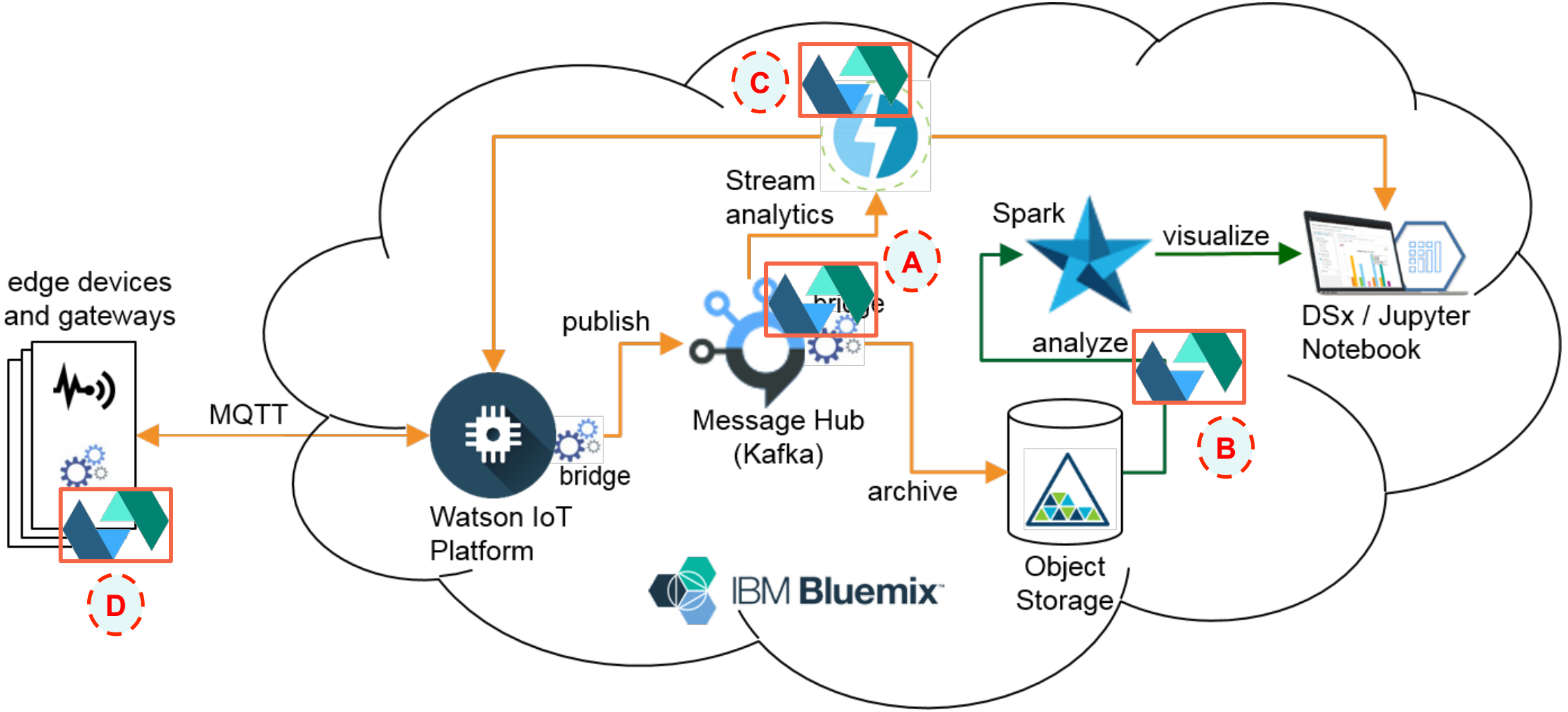
1. **Big Data: large volume (e.g., # devices) and variety (e.g., message formats)**
   - Sensors (e.g., location, temperature, audio, motion)
   - Actuators (e.g., valve, alarm, motor)
2. **Devices are typically connected via Gateways**
   - Point of aggregation and control for devices in physical proximity
   - Device-to-gateway communicating is often not TCP/IP-based (e.g., Bluetooth Low Energy)
3. **Data often undergoes initial processing at the Gateway**
   - Motivated by latency, bandwidth or privacy constraints
   - Often comprises simple real-time processing (e.g., property-based filtering, time-based aggregation)
     - May trigger local actuation (e.g., based on pre-defined rules)
4. **Data is shipped to a centralized cloud for storage and analytics**
   - Both (near) real-time and batch
   - May trigger a feedback loop to the gateways
     - E.g., actions triggered by a detected anomaly, updates of filtering thresholds

Example of IoT Application in IBM Bluemix



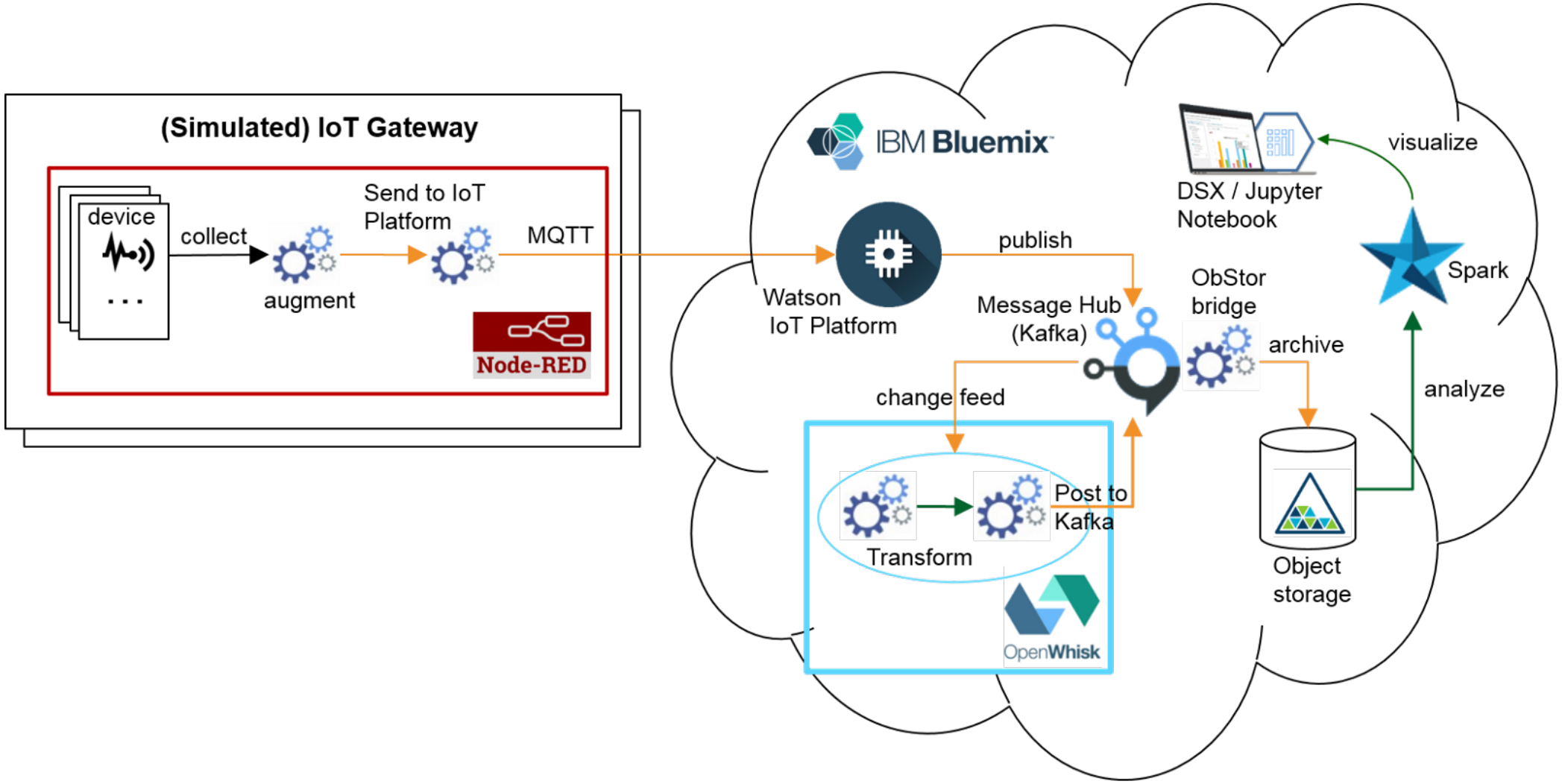
APACHE OpenWhisk™

Opportunities for Serverless/FaaS in IoT



APACHE OpenWhisk™

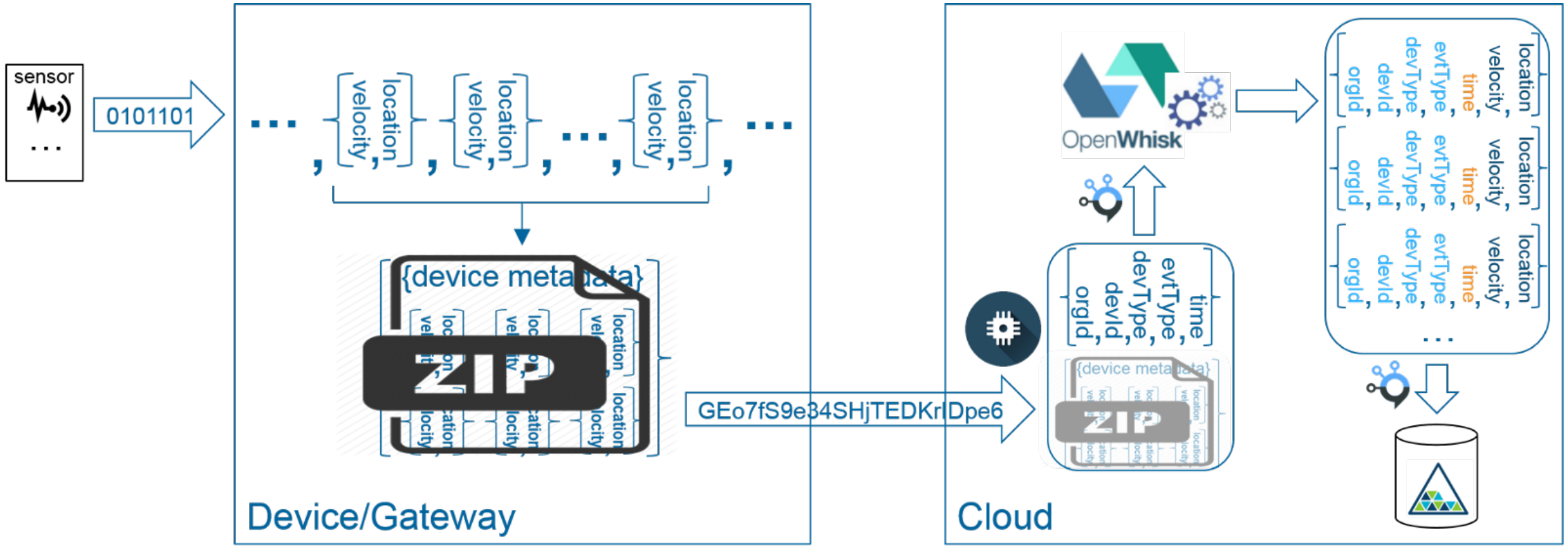
TRANSIT: Serverless Transformation of IoT Data-in-Motion



https://medium.com/openwhisk/serverless-transformation-of-iot-data-in-motion-with-openwhisk-272e36117d6c

APACHE OpenWhisk™

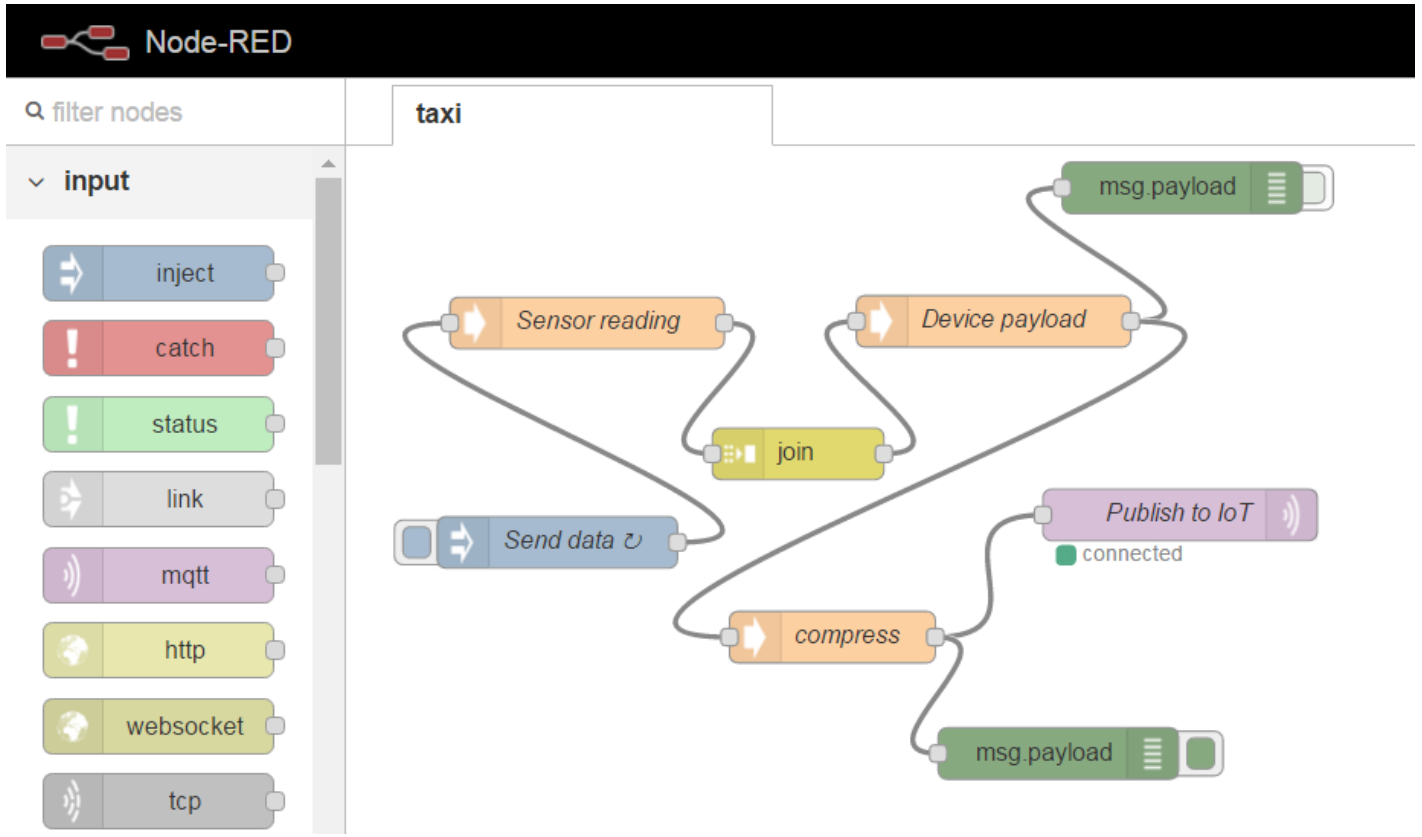
Schematic end-to-end data flow



APACHE OpenWhisk™

End-to-end data flow – 1 (Node-RED)

- **Node-RED (simulated device):**
  - The ‘inject node’ triggers the flow at given intervals (e.g., every second).
  - A sample event JSON is generated (by a custom ‘function node’).
  - It is then aggregated using the ‘join’ node,
  - compressed (using a custom function node), and
  - sent to the Watson IoT Platform (using MQTT with properly configured orgId, device type, event type, device id and security token).



End-to-end data flow – 2 (Watson IoT Platform)

- **Watson IoT Platform (with historical data storage extension to Message Hub):**
  - Receives MQTT events from devices (after proper auth validation),
  - Augments them (e.g., adding timestamp and context metadata) and
  - Publishes in small batches to a Kafka topic (Message Hub).

End-to-end data flow – 3 (OpenWhisk)

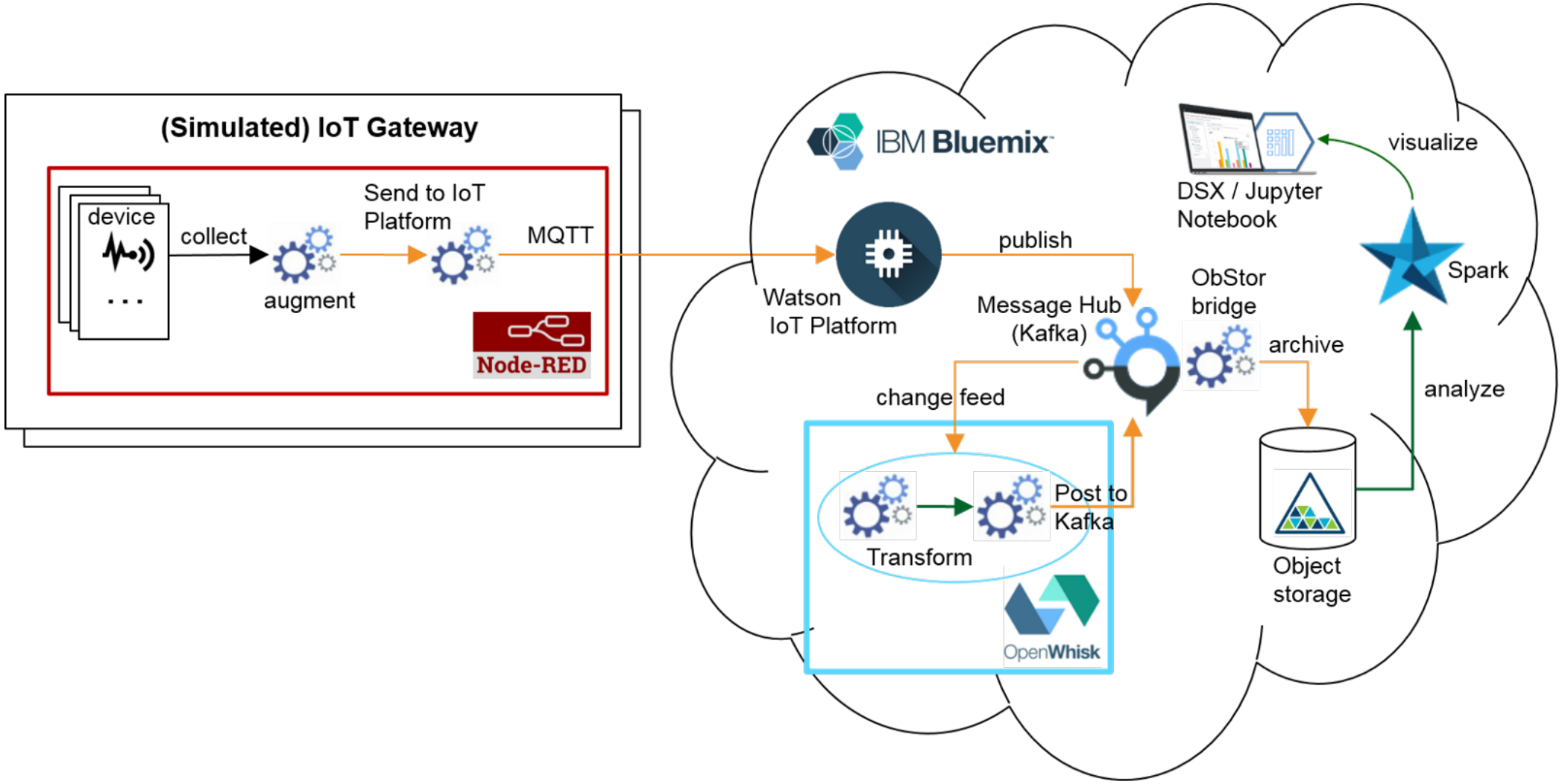
- **OpenWhisk (with custom transformation action):**
  - The Message Hub feed provider in OpenWhisk subscribes to messages in the *iotp* Kafka topic,
  - Aggregates small batches of messages and
  - Sends them as a payload to the OpenWhisk trigger, which in turn
  - Triggers a sequence of actions
  - Performing the message transformation and
  - Publishing the results to the *transformed* Kafka topic

```
function main(params) {
    zlib = require('zlib');
    return new Promise(function(resolve, reject) {
        if (!params.messages || !params.messages[0] || !params.messages[0].key || !params.messages[0].value) {
            reject("Invalid arguments. Must include 'messages' JSON array with 'key' and 'value' fields");
        }
        var msgs = params.messages;
        var out = [];
        for (var i=0; i<msgs.length; i++){
            var msg = msgs[i];
            console.log ("Processing MSG=" + JSON.stringify(msg));
            var key = JSON.parse(msg.key);
            var sensorReadings = JSON.parse(zlib.gunzipSync(msg.value))["d"].sensor_data;
            console.log ("sensor data: " + sensorReadings);
            for (var j=0; j<sensorReadings.length; j++) {
                newmsg = {
                    orgId: key.orgId,
                    deviceType: key.deviceType,
                    deviceId: key.deviceId,
                    eventType: key.eventType,
                    timestamp: key.timestamp.substr(0,19),
                    sensorData: JSON.parse(sensorReadings[j])
                    };
                out.push(newmsg);
                console.log("result: " + JSON.stringify(newmsg));
            }
        }
        resolve({ "messages": out });
    });
}
```

End-to-end data flow – 4 (Message Hub)

- **Message Hub (bridge to Object Storage):**
  - Aggregates messages published to the transformed topic according to a specified policy (e.g., files of up to 1 hour or 1MB of data), and
  - Uploads the files to Object Storage according to the specified layout (e.g., applying date-based partitioning).

TRANSIT: Serverless Transformation of IoT Data-in-Motion

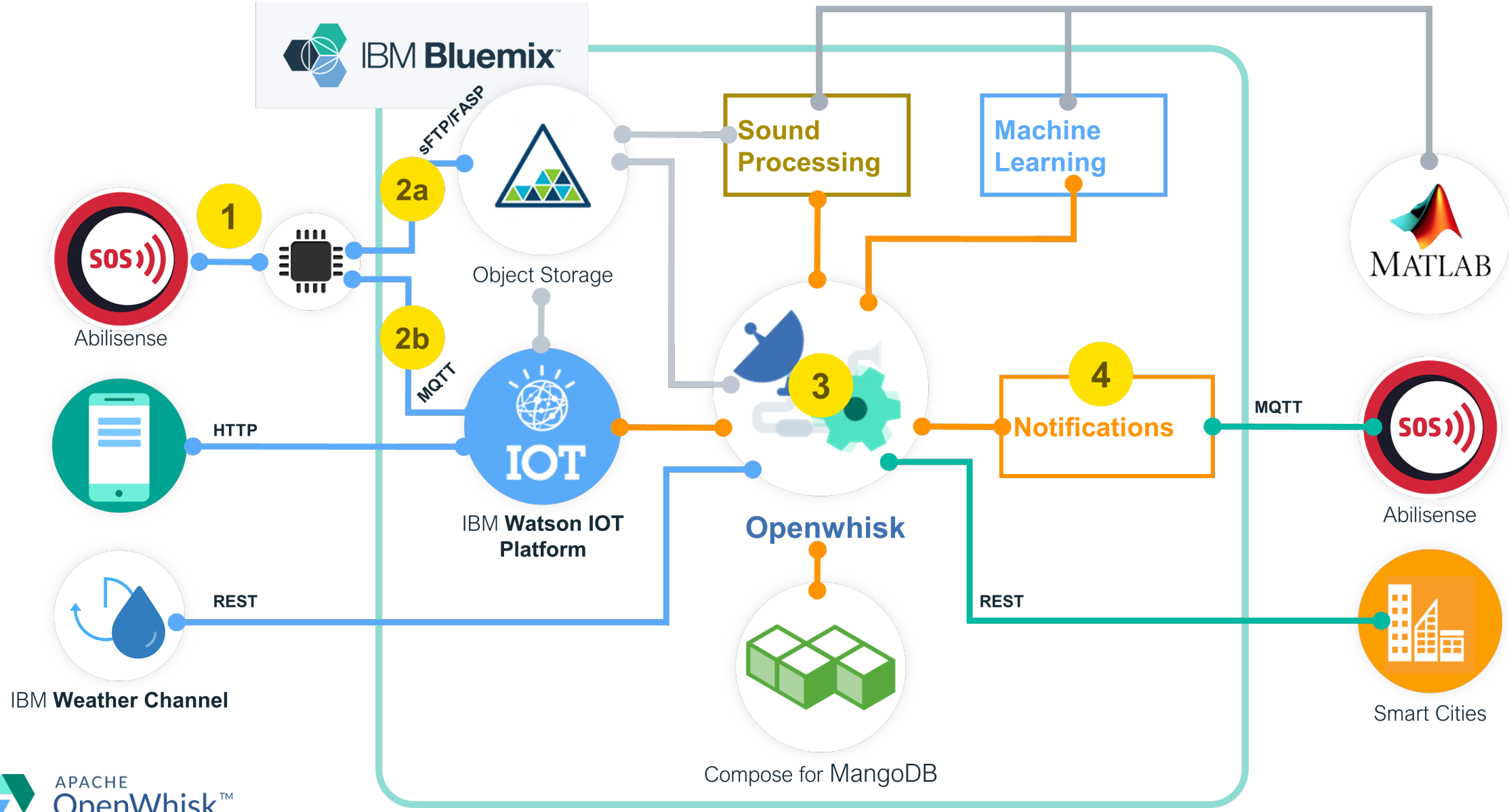


https://medium.com/openwhisk/serverless-transformation-of-iot-data-in-motion-with-openwhisk-272e36117d6c

APACHE OpenWhisk™

Example: Citizen Safety

https://www.abilisense.com/



APACHE OpenWhisk™

IBM

3. Serverless Application Patterns for the IoT

4 Serverless/FaaS Patterns for IoT Applications

Pattern #1: FaaS-based micro-services (decoupled, event-driven)

a) Business logic is decoupled into self-contained micro-services, implemented as stateless Functions
b) The business logic is triggered by IoT events (e.g., ingested via MQTT)
c) Functions communicate via events or state changes in dedicated persistent services (e.g., data stores)

Pattern #2: FaaS as a 'glue' integrating services in an event- or data-driven flow

a) The application is heavily relying on existing services (e.g., cognitive, data, analytics)
b) The 'orchestration' involves lightweight custom code, triggered by events or data
c) Each handler interacting with external services is encapsulated as a stateless short-living Function

Pattern #3: Acting upon outstanding conditions in stream processing

a) IoT events are ingested via a high-throughput channel (e.g. Message Bus)
b) Stateful stream processing is applied on incoming events (e.g., anomaly detection)
c) Upon detection of pre-defined conditions, a Function is triggered to apply custom business logic

Pattern #4: FaaS-enabled edge analytics

a) Business logic deployed on individual devices or gateways (e.g., due to latency, bandwidth, privacy)
b) Functions are portable across cloud and edge, enabling streamlined app development and operation

Summary

- **FaaS platforms enable rapid development of loosely-coupled event-driven applications**
  - Increased agility, manageability, elasticity

- **OpenWhisk is a leading open source FaaS project**
  - Incubating under Apache, and available commercially by several vendors (including IBM)

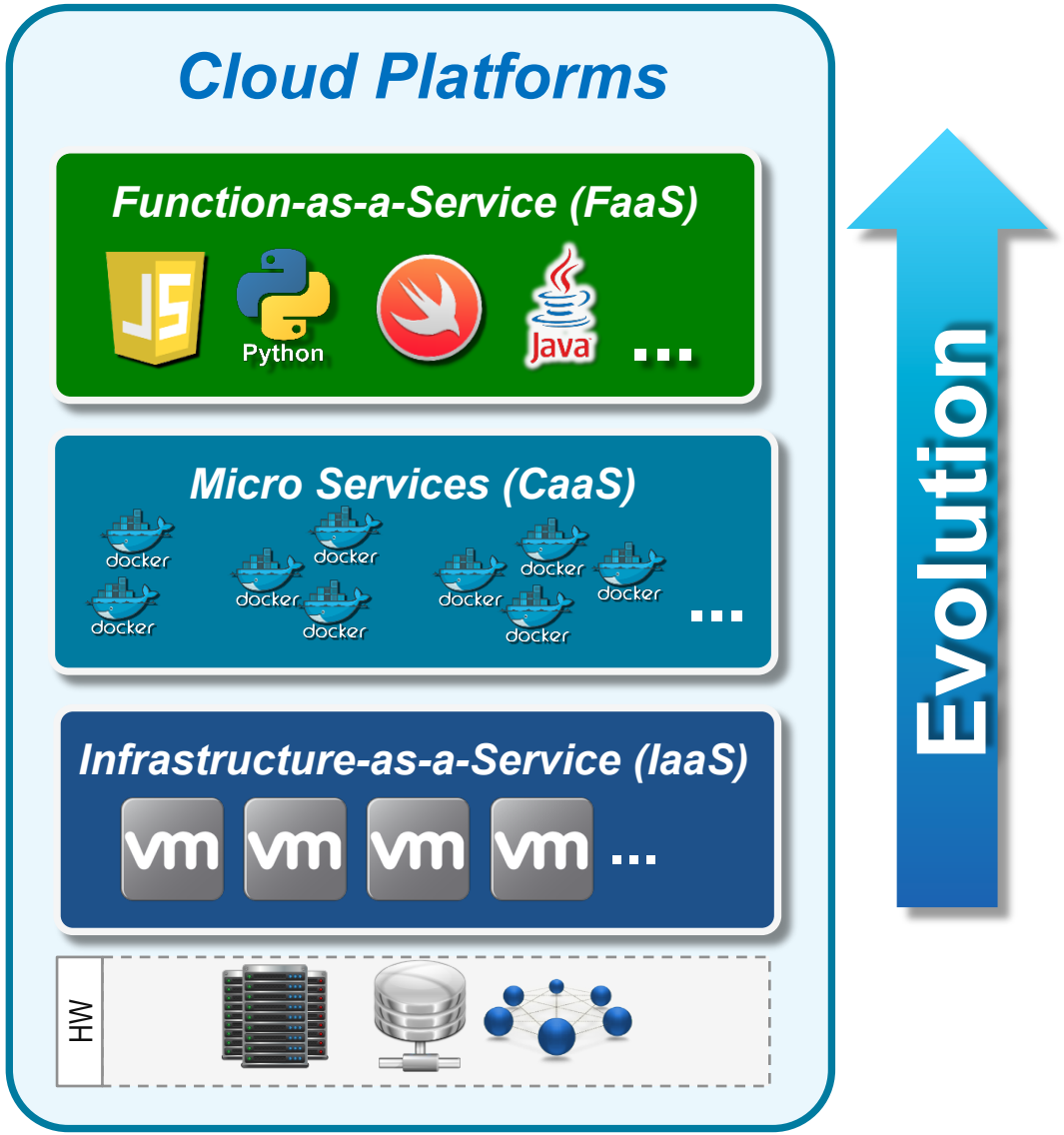
- **Serverless design patterns are still emerging, but there are many examples of leveraging FaaS in IoT applications**
  - Pattern #1: FaaS-based micro-services (decoupled, event-driven)
  - Pattern #2: FaaS as a ‘glue’ integrating services in an event- or data-driven flow
  - Pattern #3: Acting upon outstanding conditions in stream processing
  - Pattern #4: FaaS-enabled Edge Analytics (**work in progress**)

IBM

Questions?

Alex Glikson
https://medium.com/@glikson
http://slideshare.net/AlexGlikson/
@glikson

The Evolution of Cloud-Native Applications



Serverless ("3.0"): Functions-as-a-Service (FaaS)

- **Architecture**: event-driven flow of single-tasked functions
- **Unit of deployment**: Functions, collection of functions
- **Connectivity**: de-coupled (e.g., via state changes/events)
- **Scaling**: per-request (with optimization of warm-up times)

Micro Services ("2.0"): Container-as-a-Service (CaaS)

- **Architecture**: composition of small, self-contained services, calling each other via a REST API
- **Unit of deployment:** group of containers comprising a service
- **Connectivity**: run-time service discovery and routing
- **Scaling**: container-based auto-scaling of each micro-service

Scale-out ("1.0"): Infrastructure-as-a-Service (IaaS)

- **Architecture:** layered into few application tiers (often 3-tier)
- **Unit of deployment:** Virtual Machine (VM), collection of VMs
- **Connectivity**: configuration-time routing between tiers
- **Scaling**: VM-based (auto-) scaling within each tier

APACHE OpenWhisk™

Applying the Serverless Paradigm to IoT Applications

- **The Serverless/FaaS paradigm has great potential for IoT**
  - Event-driven programming model: a natural fit for many IoT applications
  - Rapid programmability: lack of standards results in high heterogeneity and the need for customization
  - Low entry barrier (pay-per-use, low ops overhead): attractive for the 'long tail' of DIY-style applications

- **Challenges**
  - New paradigm, lack of best practices ('patterns')
  - Lack of mature developer tools
  - Need more powerful programming model (e.g., for stateful stream processing)
  - Need a holistic approach that spans the cloud and the edge