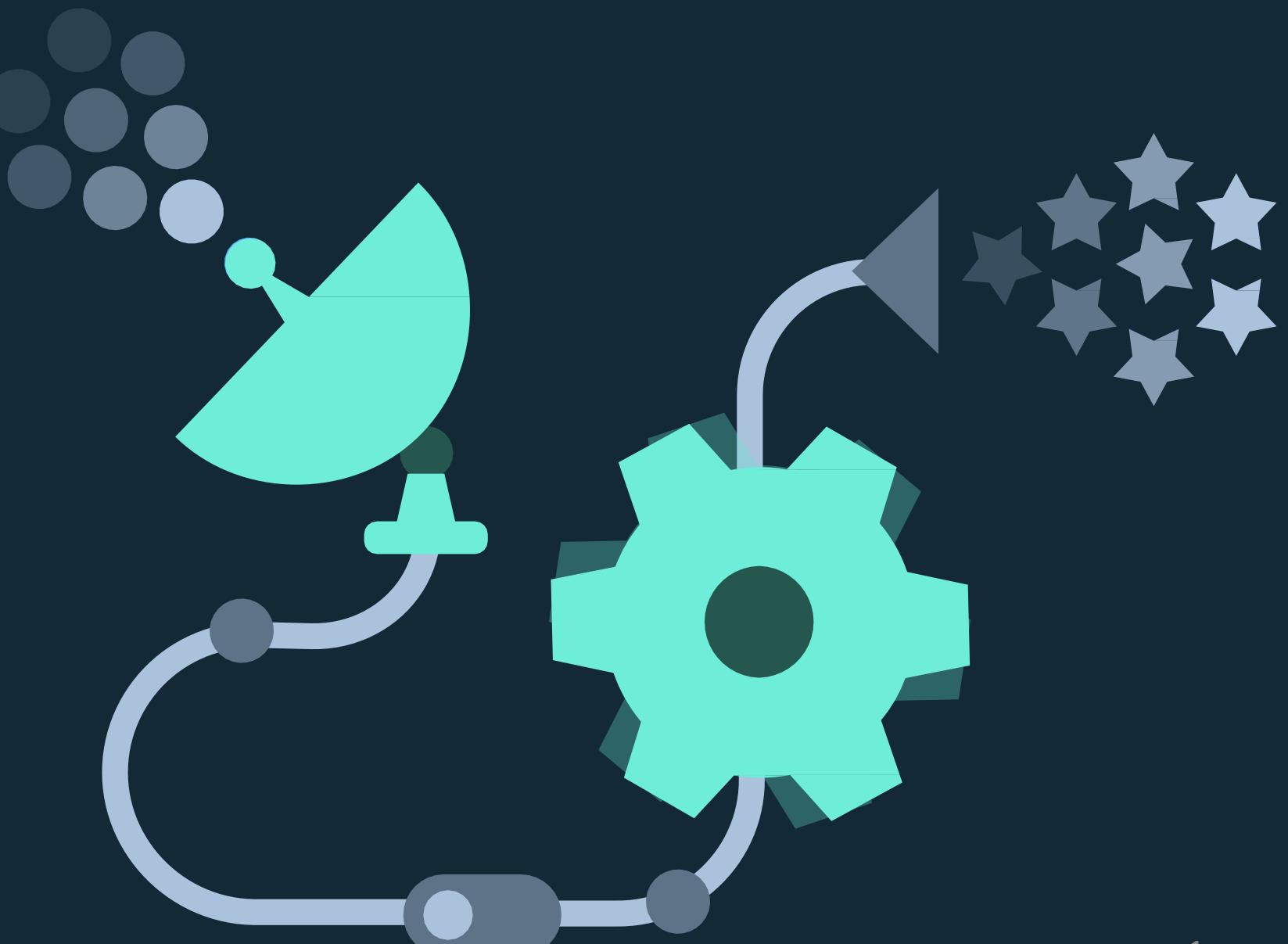


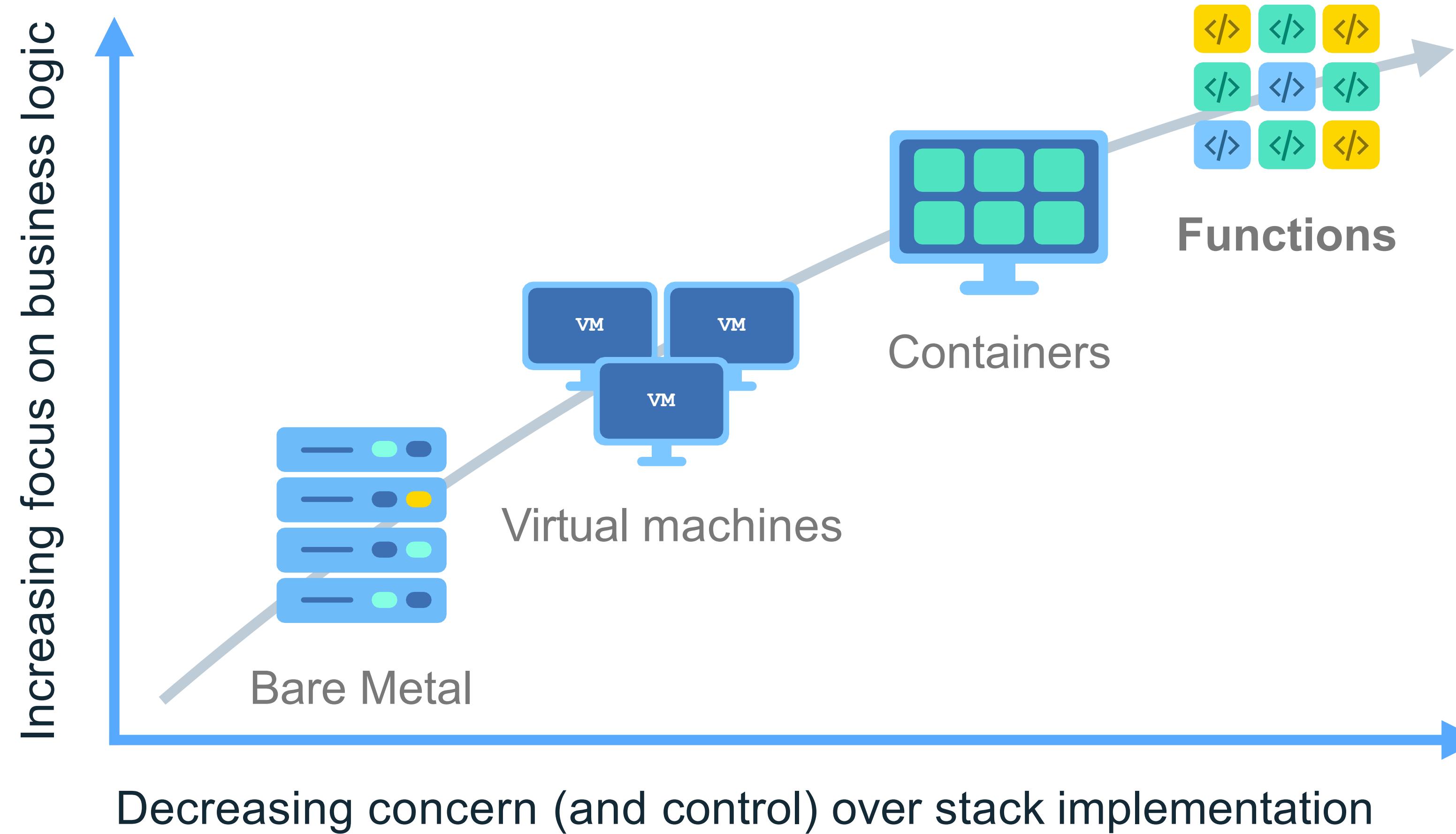
Building serverless applications with IBM Cloud Functions

Sudharshan Govindan
Developer Advocate

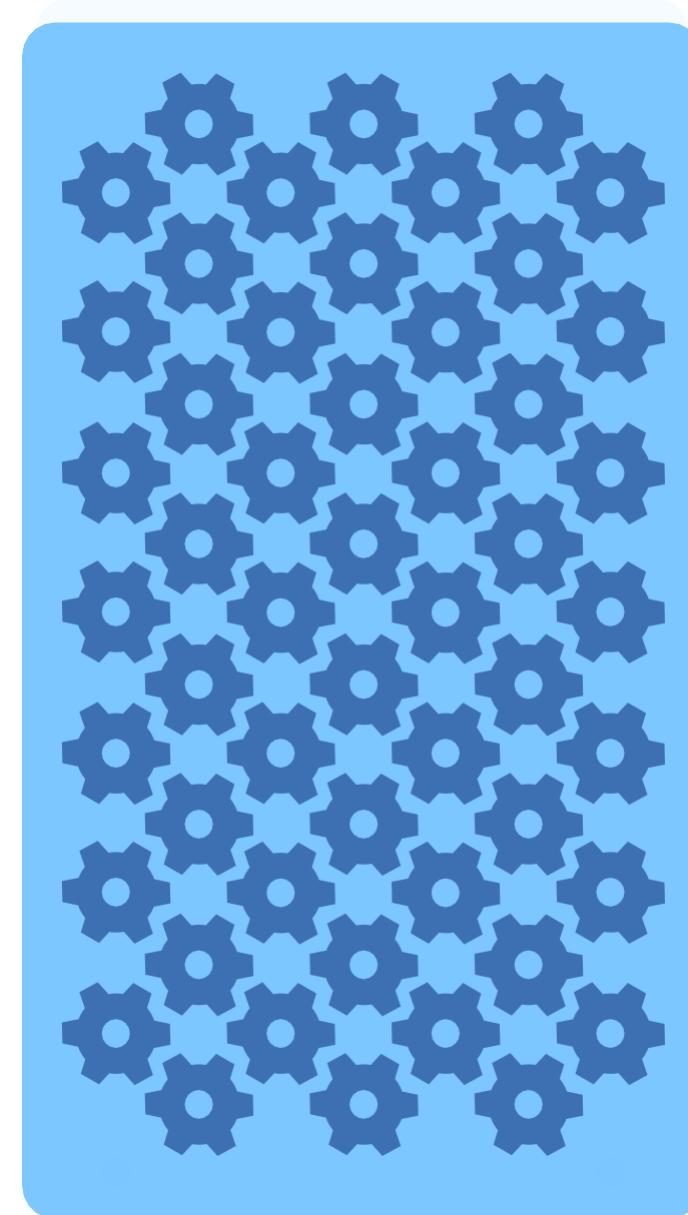
sudharshan.govindan@in.ibm.com
@sudhargovindan



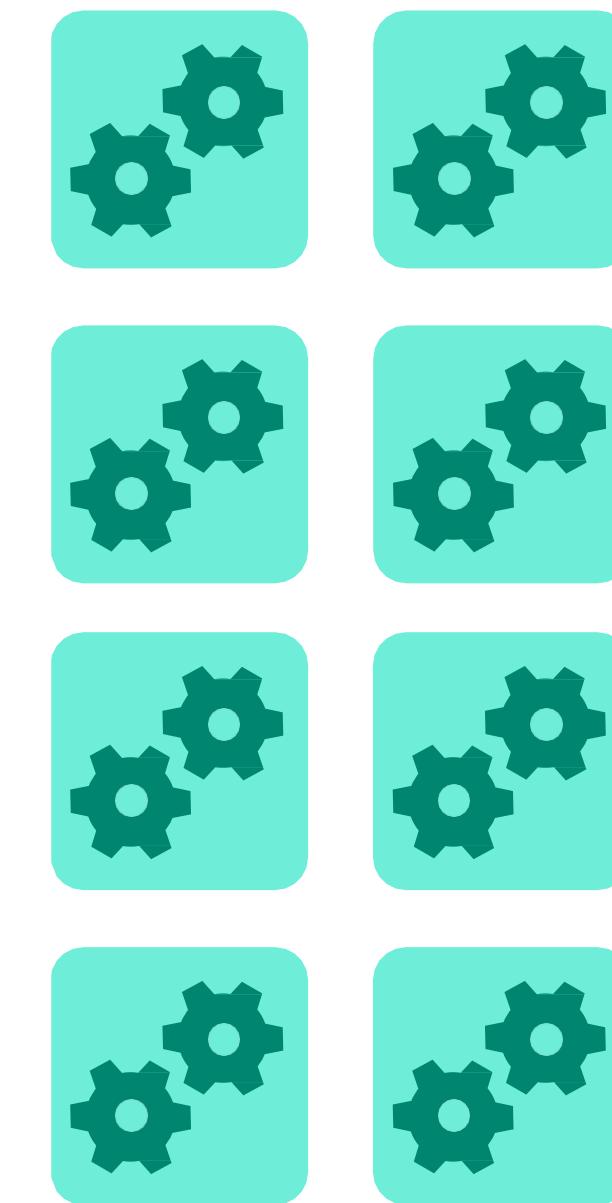
Serverless developers focus more on code, less on infrastructure



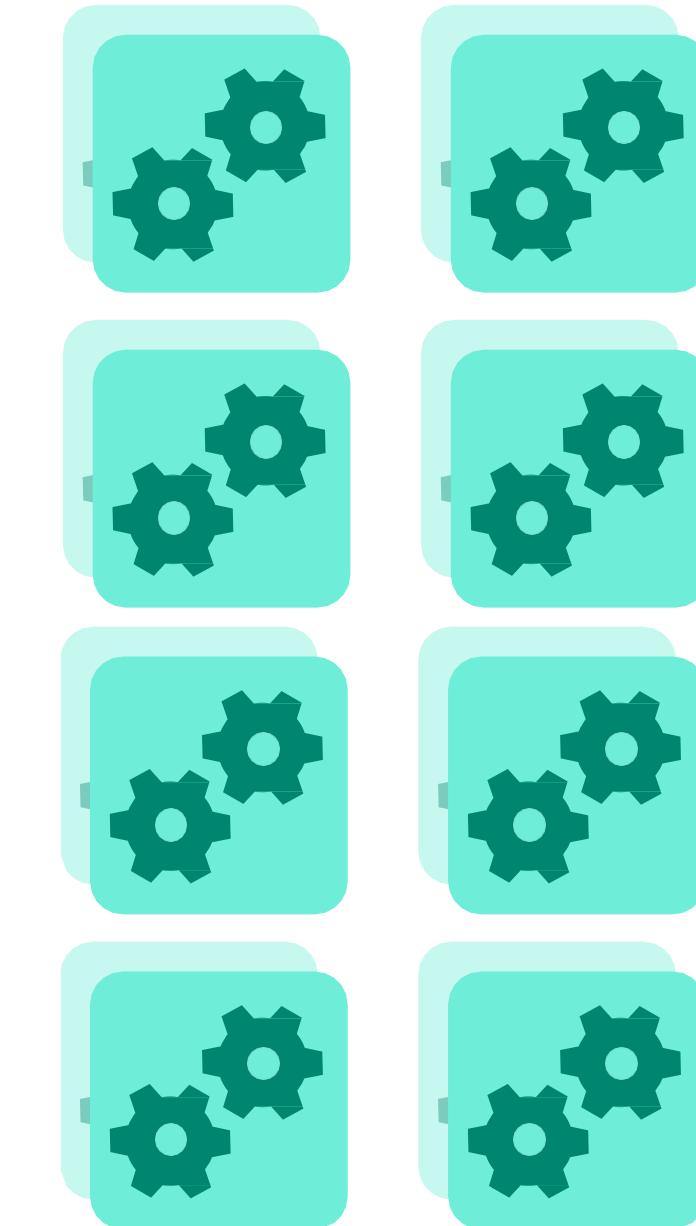
Microservices can be hard to manage at scale



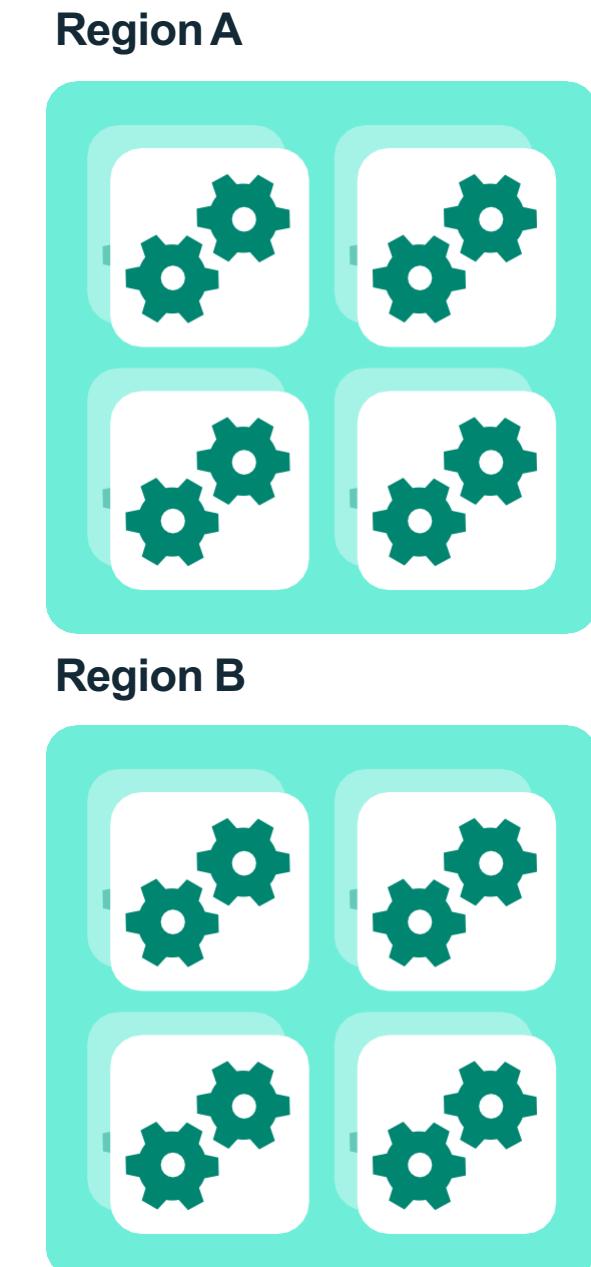
Monolithic Application



Break-down into microservices



Make each microservice HA



Region A

Region B

Protect against regional outage

Emerging workloads are a good fit for event-driven programming



Execute logic in response to database change



Perform analytics on sensor input messages



Provide cognitive computing via chatbots



Schedule tasks performed for a short time

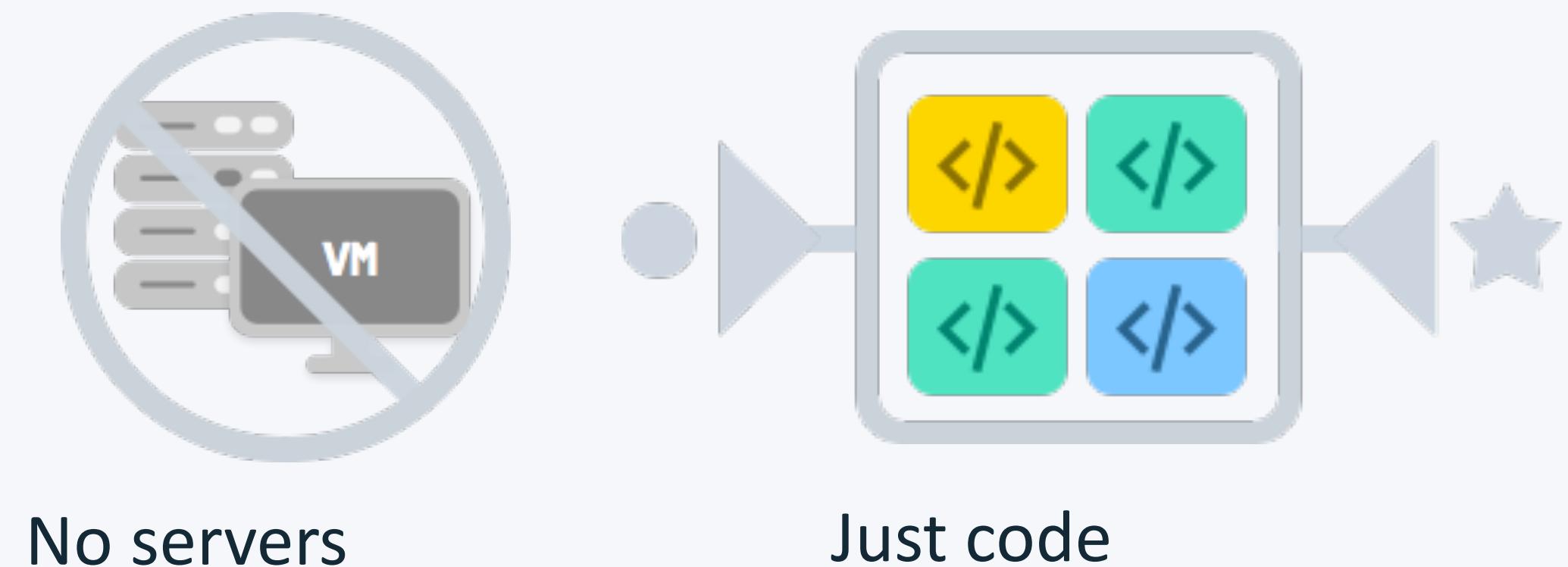


Invoke auto scaled APIs and mobile backends

Runs code **only** on-demand on a per-request basis

No management and operation of **infrastructures**

Focus on developing **value-adding code** and on driving **innovations**



Traditional model

Worry about scaling

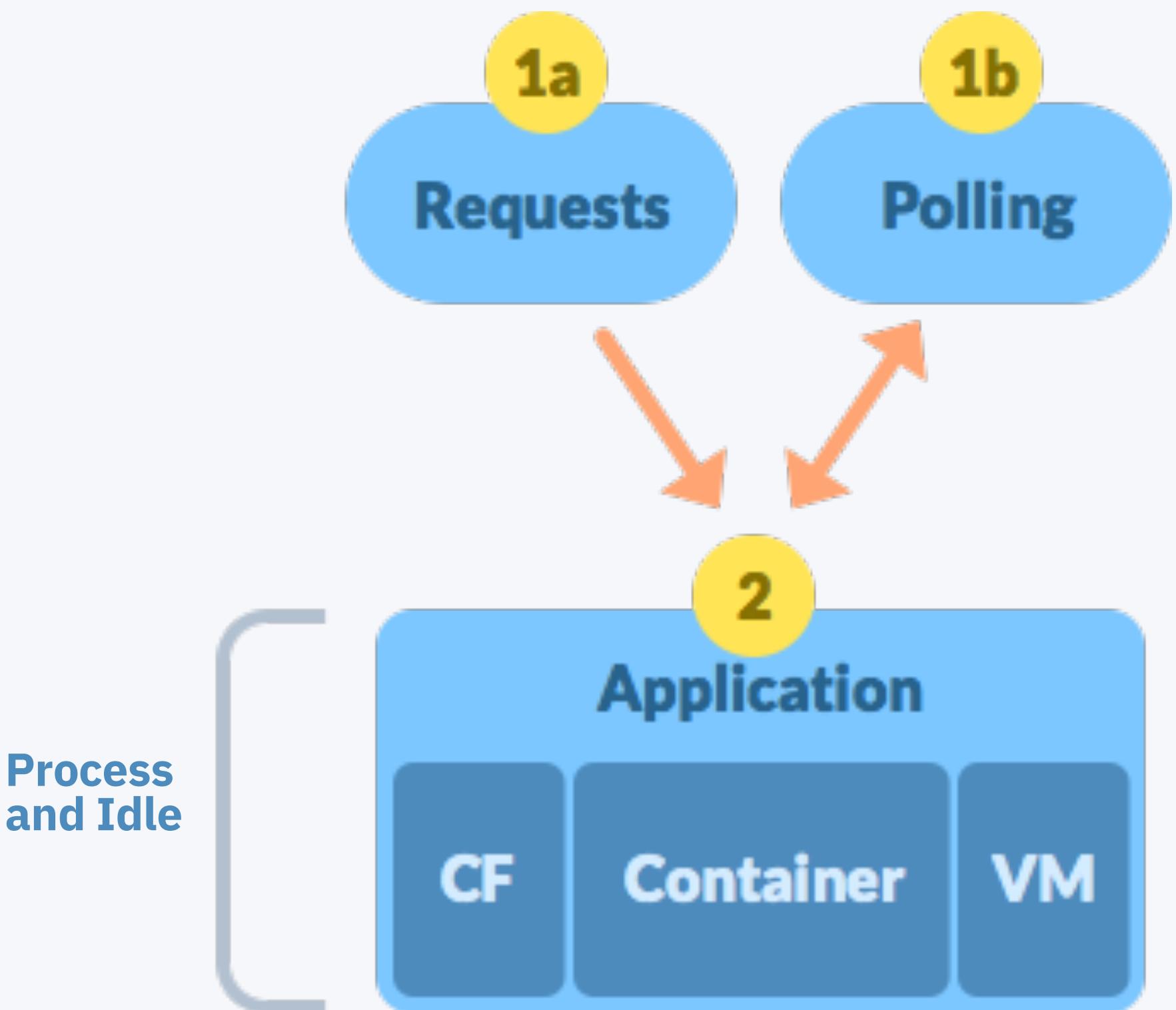
- When to scale? (mem-, cpu-, response time-, etc. driven?)
- How fast can you scale?

Worry about resiliency & cost

- At least 2 processes for HA
- Keep them running & healthy
- Deployment in multiple regions

Charged even when idling / not 100% utilized

Continuous polling due to missing event programming model



Serverless model

Scales inherently

- One process per request

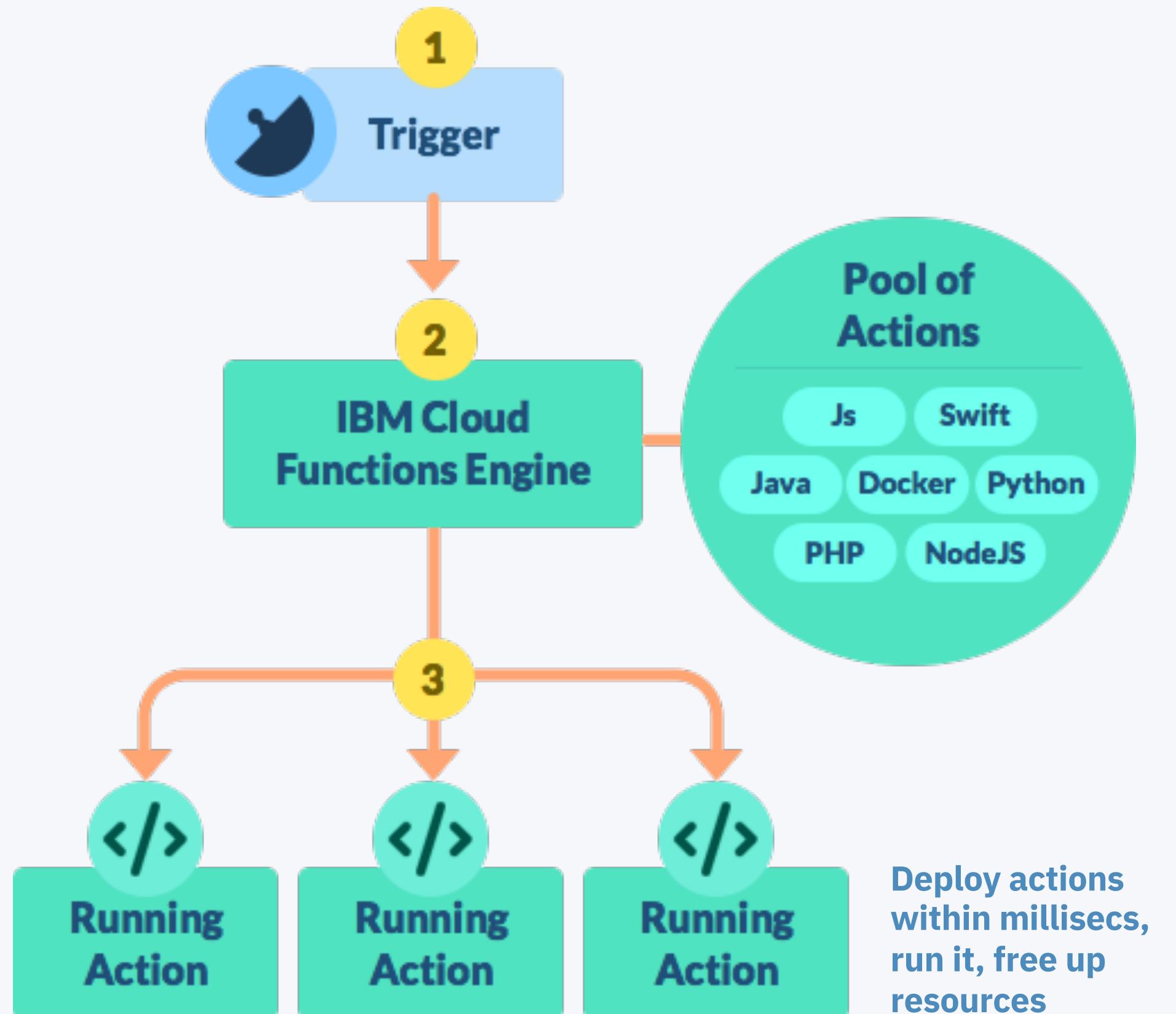
No cost overhead for resiliency

- No long running process to be made HA / multi-region

Introduces event programming model

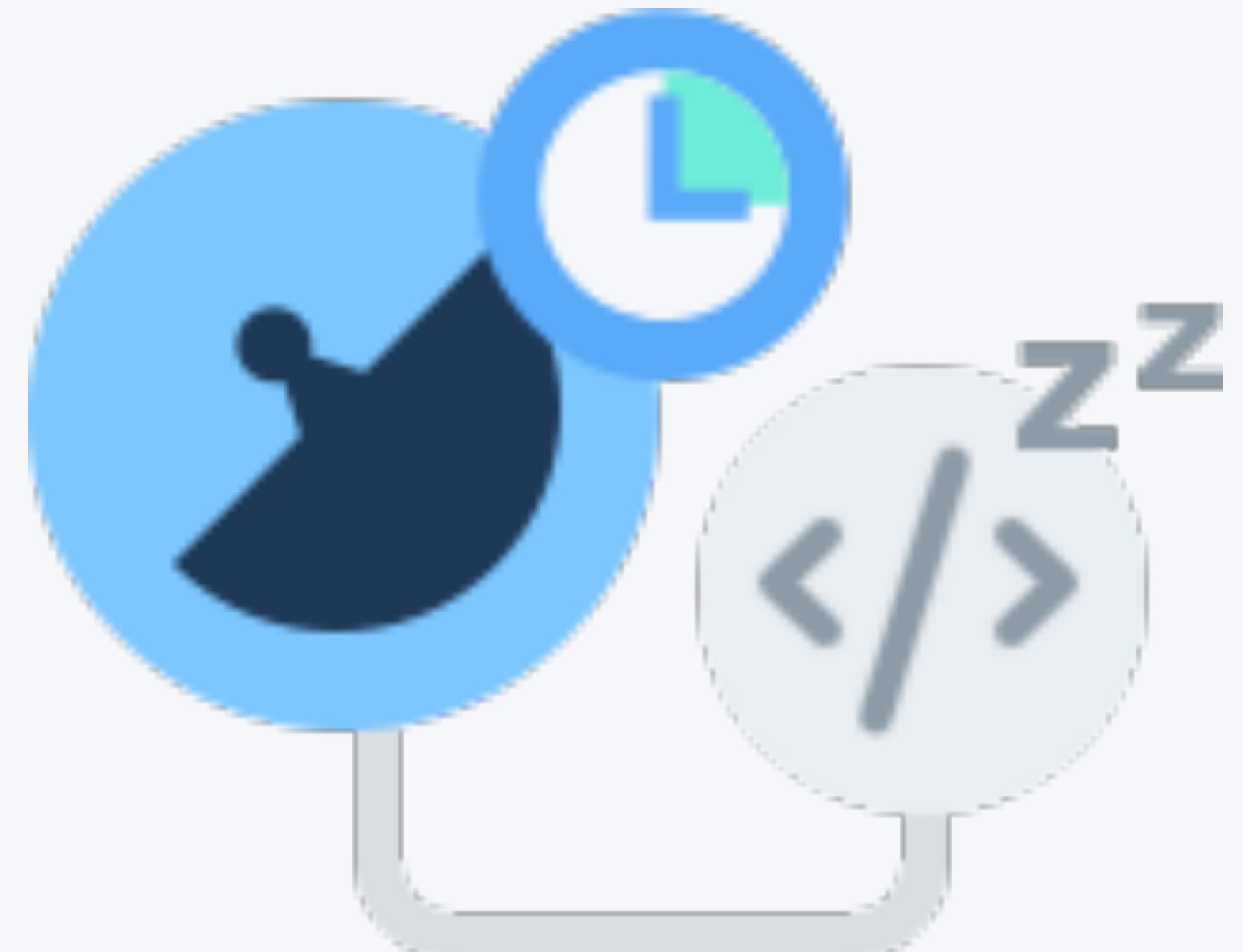
Charges only for what is used

- Only worry about code
- Higher dev velocity, lower operational costs



Runs code **only** on-demand on a per-request basis

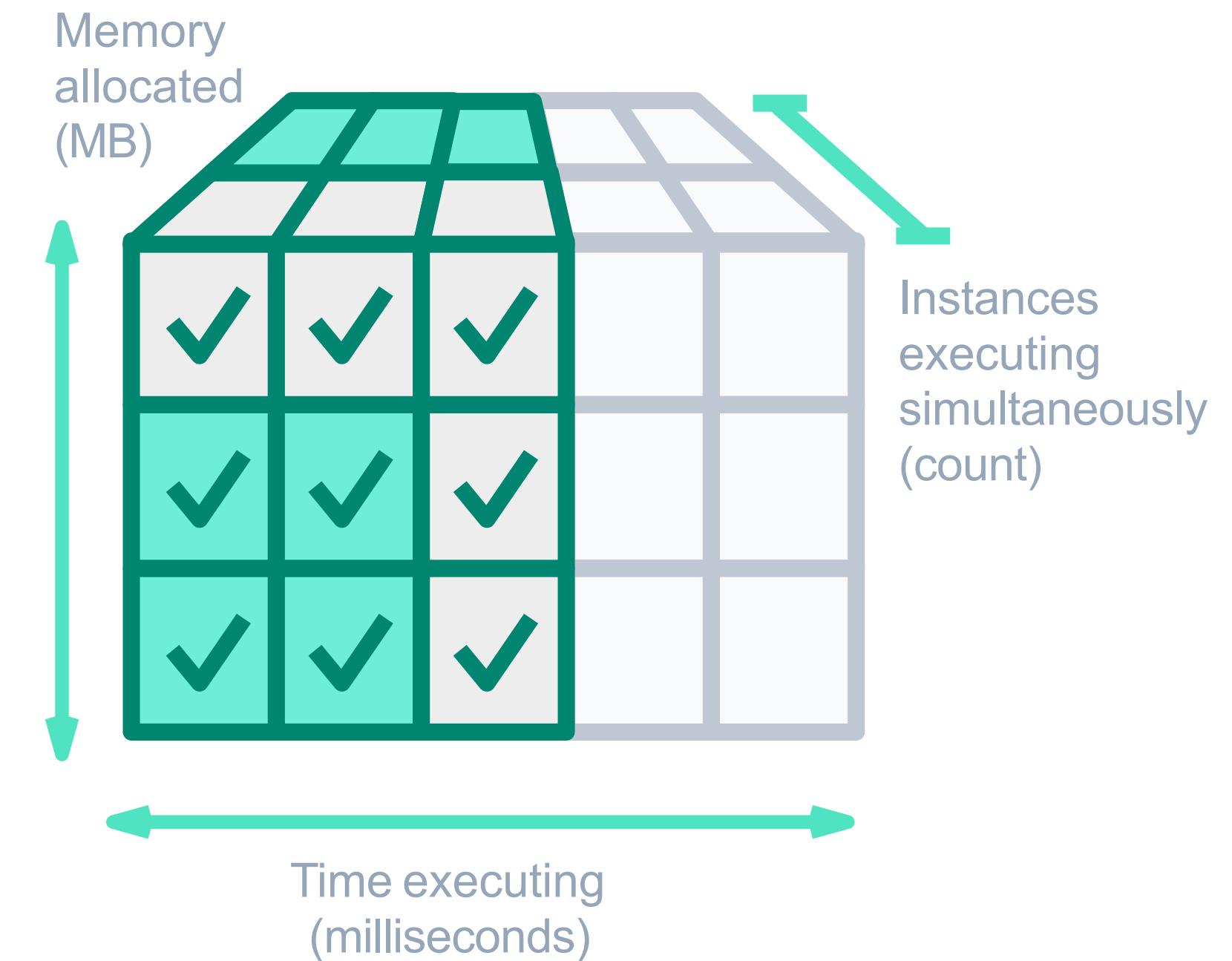
Only **pay** for resources being used, instead of resources idling around



New cost models more accurately charge for usage

Cloud resource cost
better matches
business value gained

*Not a silver bullet, but this can result in
substantial savings for many workloads*



OpenWhisk enables these serverless, event-driven workloads



OpenWhisk is an
opensource cloud
platform that
executes code in
response to events

- Provides serverless deployment and operations model
- Runs code only on-demand on a per-request basis
- Optimized utilization, fine-grained metering at any scale
- Flexible, extensible, polyglot programming model
- Open source and open ecosystem
- Ability to run in public, private and hybrid cloud models

Any language or binary is supported

Natively supported languages (performance-optimized)

JS/NodeJS

Java

PHP

Go

Swift

Python

Ruby

.NET

Executable binaries & any other application or language via Containers

Docker

C

bash

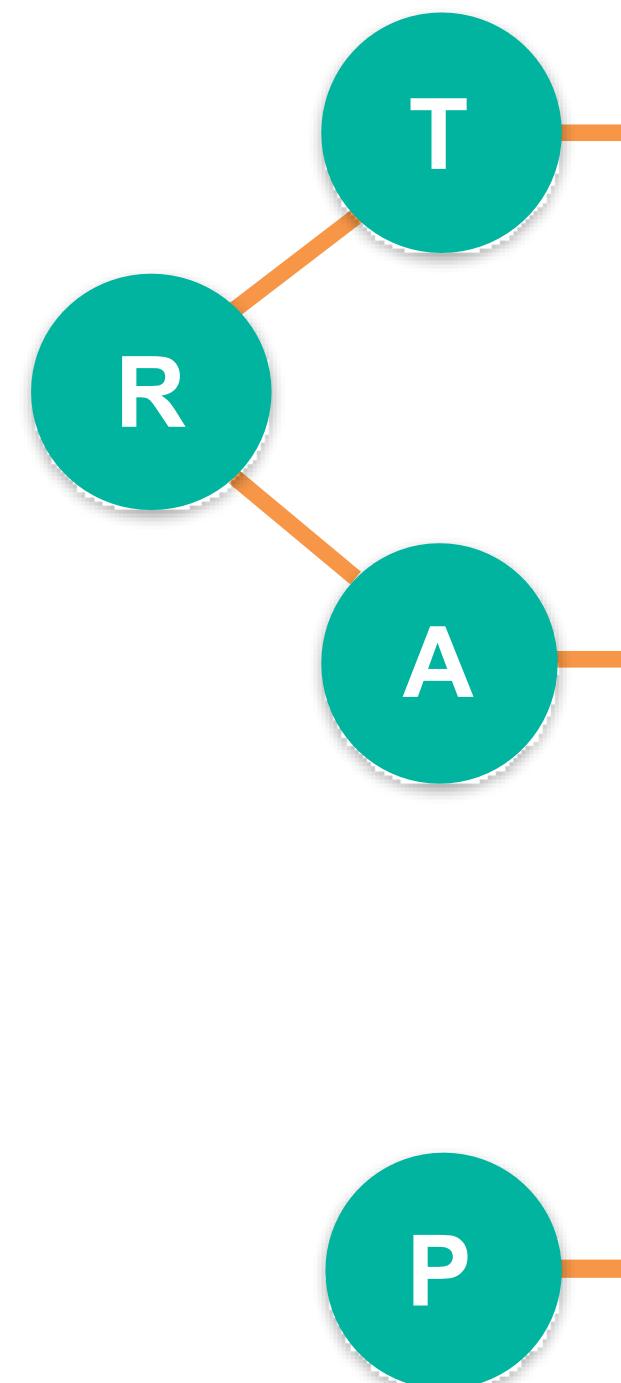
Rust

C++

...

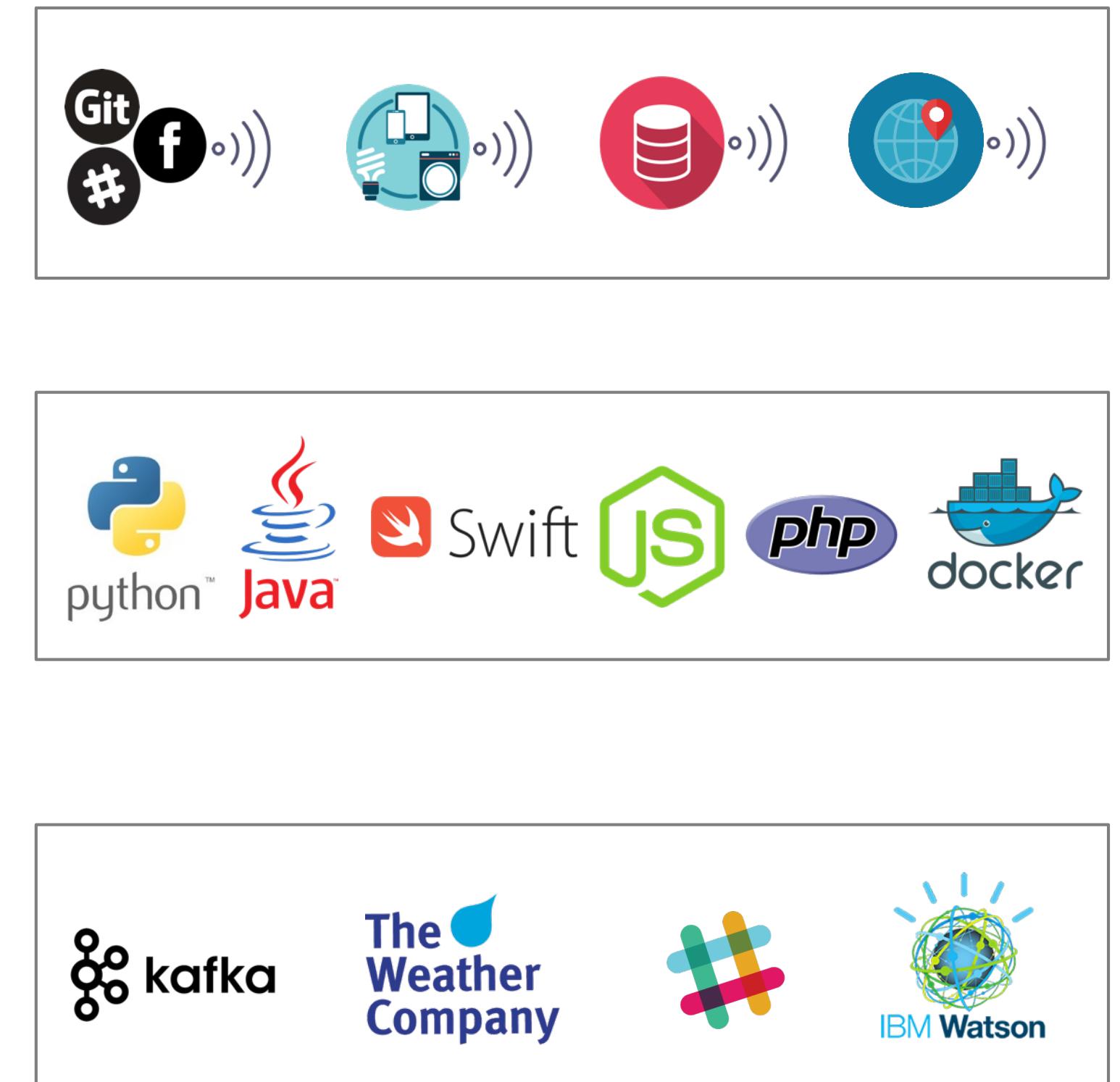
Developers work with triggers, actions, rules and packages

Data sources define events they emit as **Triggers**

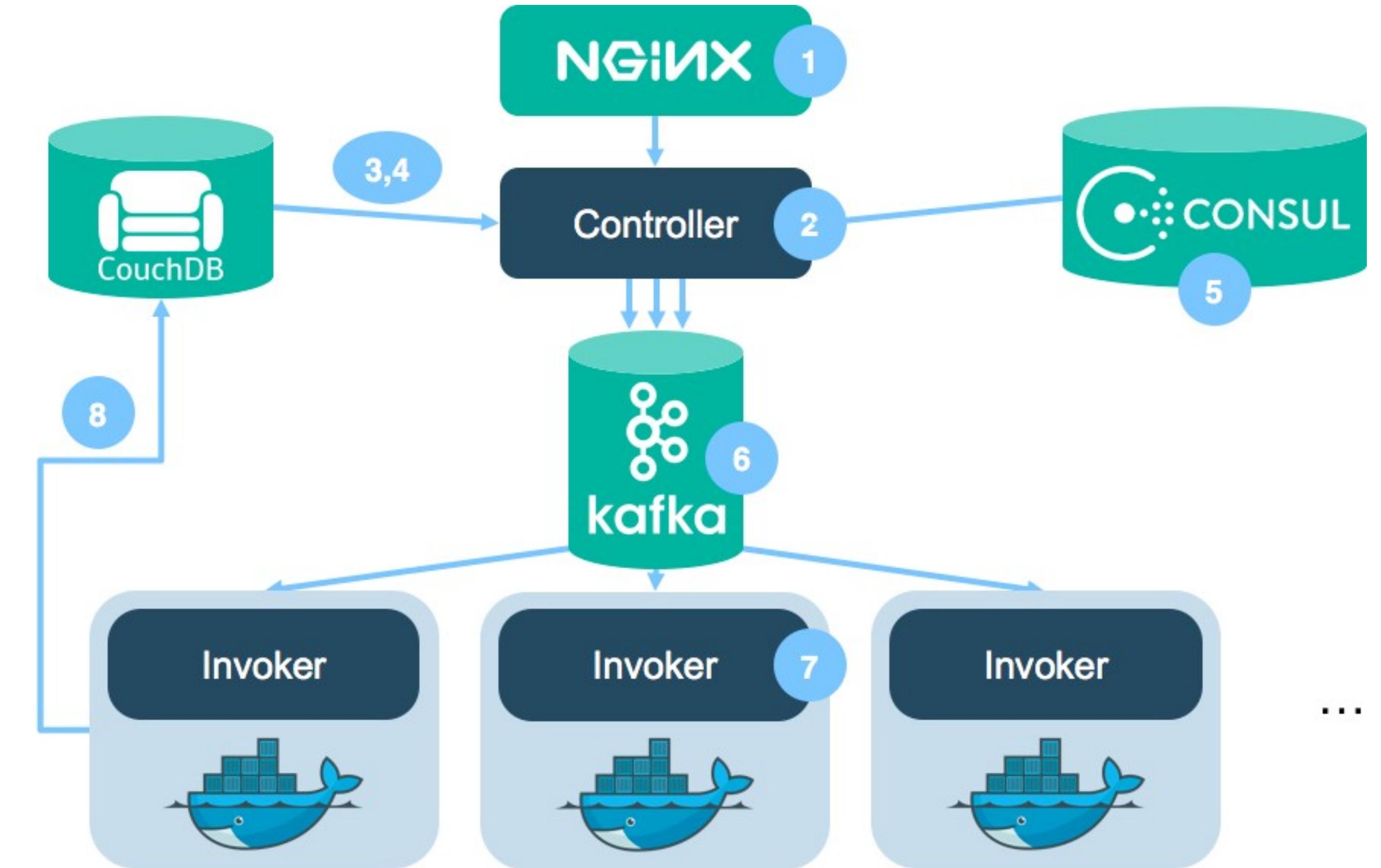


Developers map **Actions** to **Triggers** via **Rules**

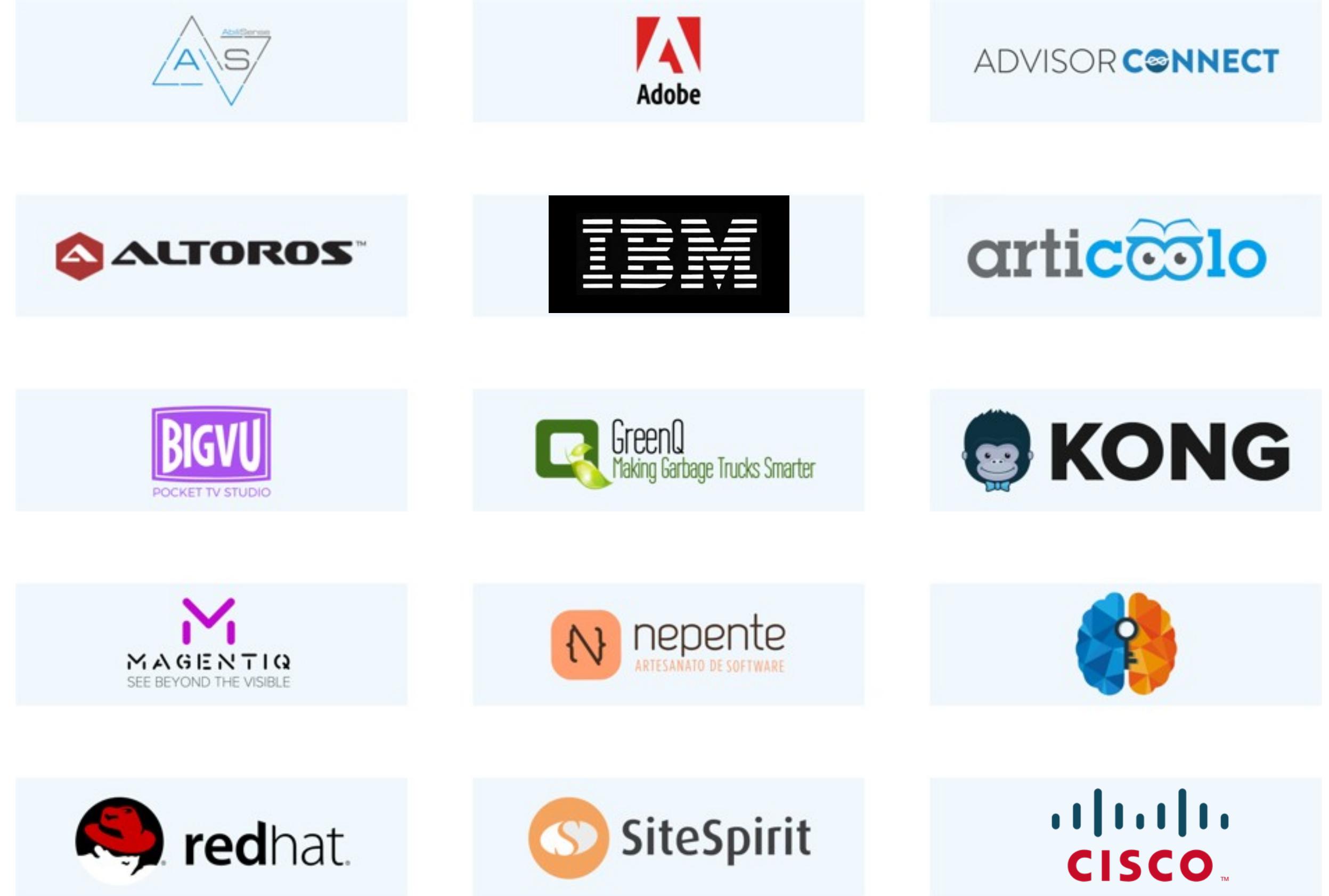
Packages provide integration with external services



OpenWhisk is built on solid open source foundations



The OpenWhisk ecosystem is growing



Serverless platform can address many of the 12 Factors

I	Codebase	Handled by developer (Manage versioning of functions themselves)
II	Dependencies	Handled by developer, facilitated by serverless platform (Runtimes and packages)
III	Config	Handled by platform (Environment variables or injected event parameters)
IV	Backing services	Handled by platform (Connection information injected as event parameters)
V	Build, release, run	Handled by platform (Deployed resources immutable and internally versioned)
VI	Processes	Handled by platform (Single stateless containers used)
VII	Port binding	Handled by platform (Actions or functions automatically discovered)
VIII	Concurrency	Handled by platform (Process model hidden and scales in response to demand)
IX	Disposability	Handled by platform (Lifecycle hidden from user, fast startup and elastic scale prioritized)
X	Dev/prod parity	Handled by developer (Developer is deployer. Scope of what differs narrower)
XI	Logs	Handled by platform (Developer writes to console.log, platform streams logs)
XII	Admin processes	Handled by developer (No distinction between one off processes and long running)

Common use-cases

- Microservices / RESTful APIs
- Batch / Periodic Processing
- Operations Research / Combinatorial Optimization
- Machine Learning Models
- Blockchain Front End
- ETL Pipelines
- Chatbots
- Event Stream Processing

Serverless Application Platform

While IBM Cloud Functions is the key anchor point for serverless, there is a **growing set of services** from other domains **also delivering serverless attributes**

This enables customers to **build application topologies which are entirely serverless**



Cloud
Functions



Cloudant



Object
Storage



Event
Streams



API Gateway



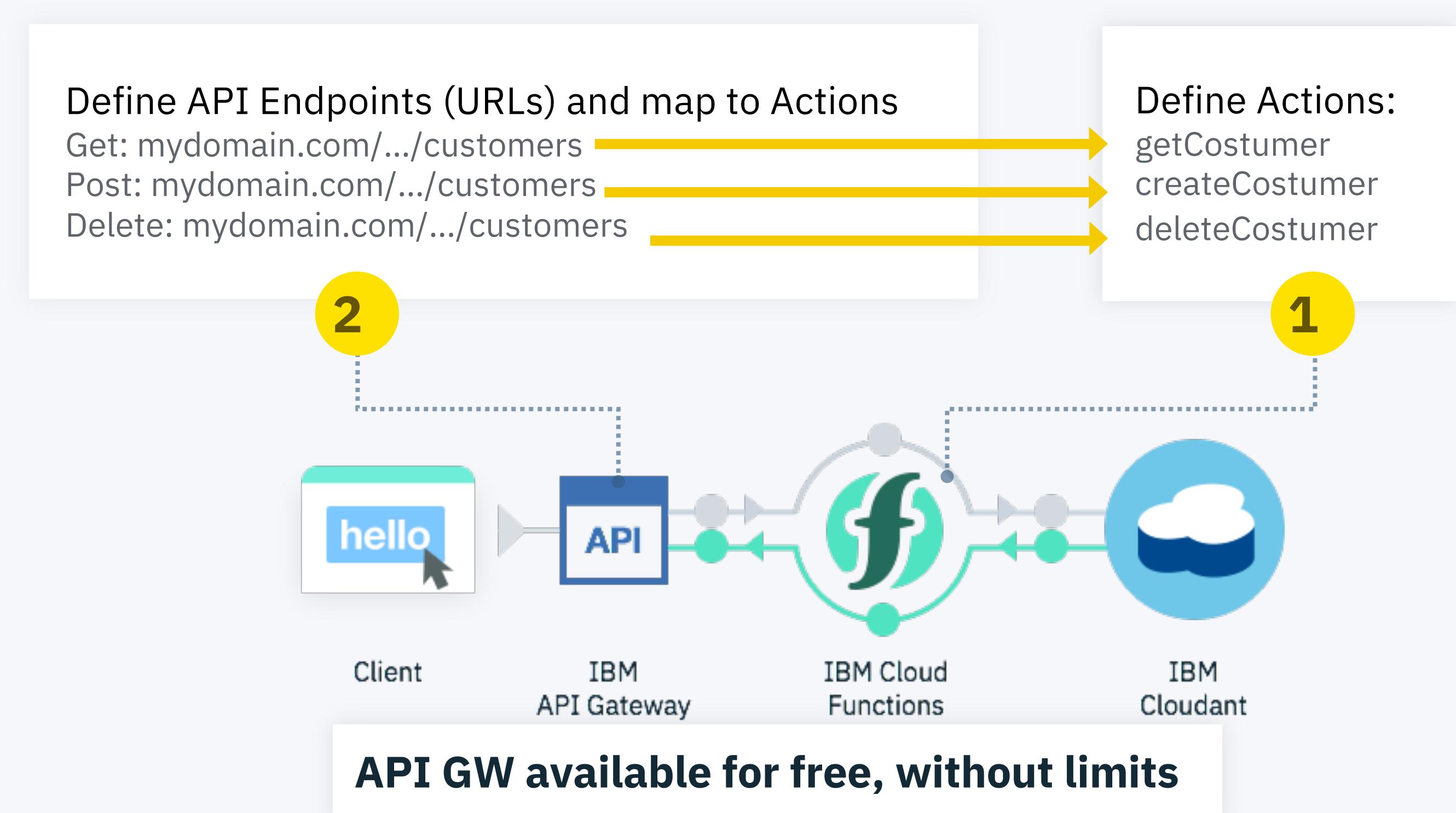
Watson
services



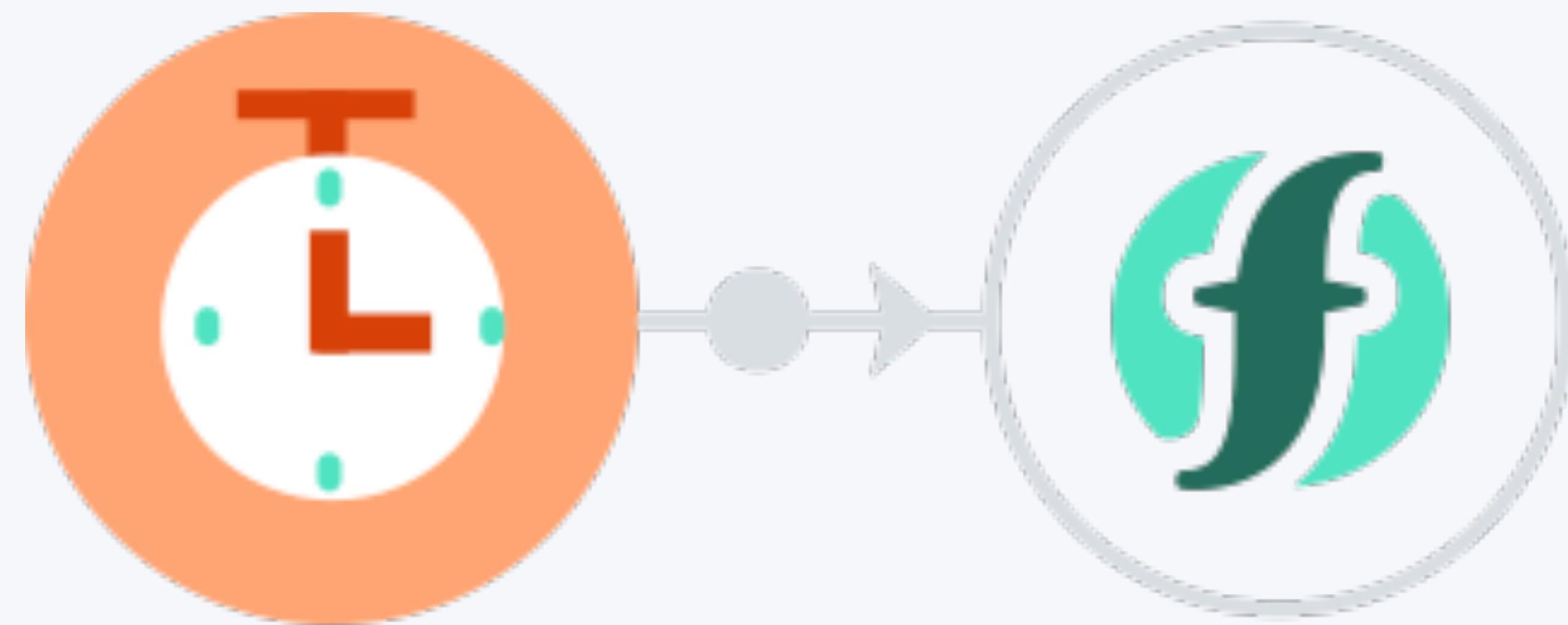
SQL
service

Microservices/ API Backends

Allows to map API
endpoints
to IBM Cloud
Functions actions



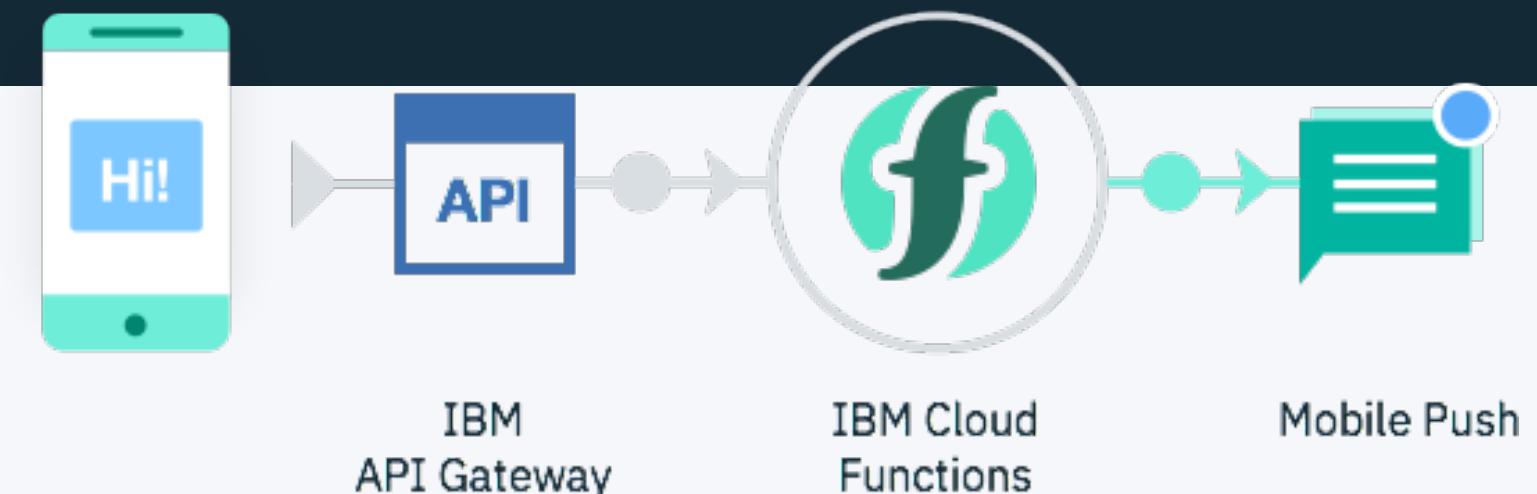
Batch / Scheduled tasks



Periodic
Alarm

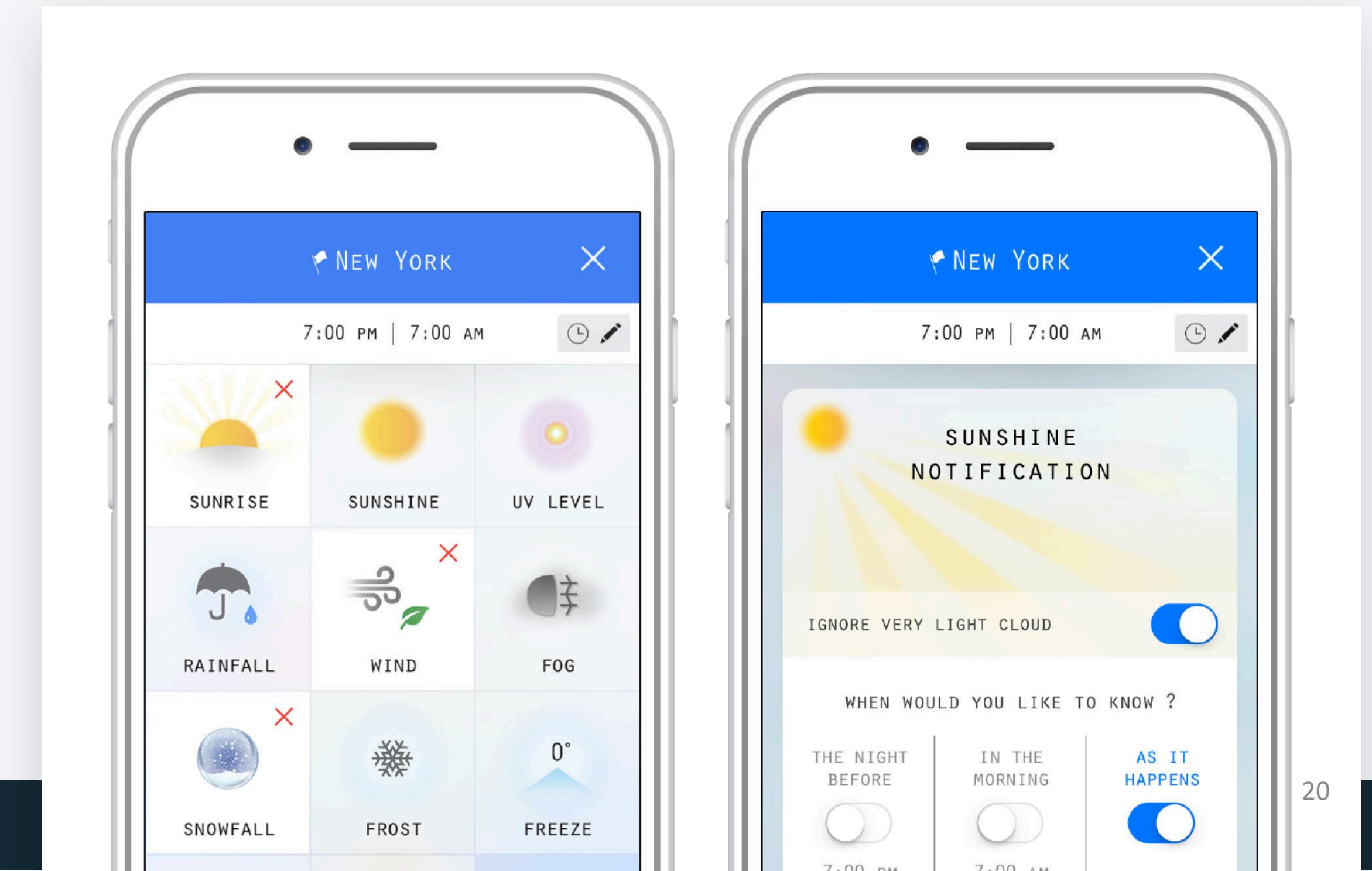
IBM Cloud
Functions

Mobile backend



The Weather Gods

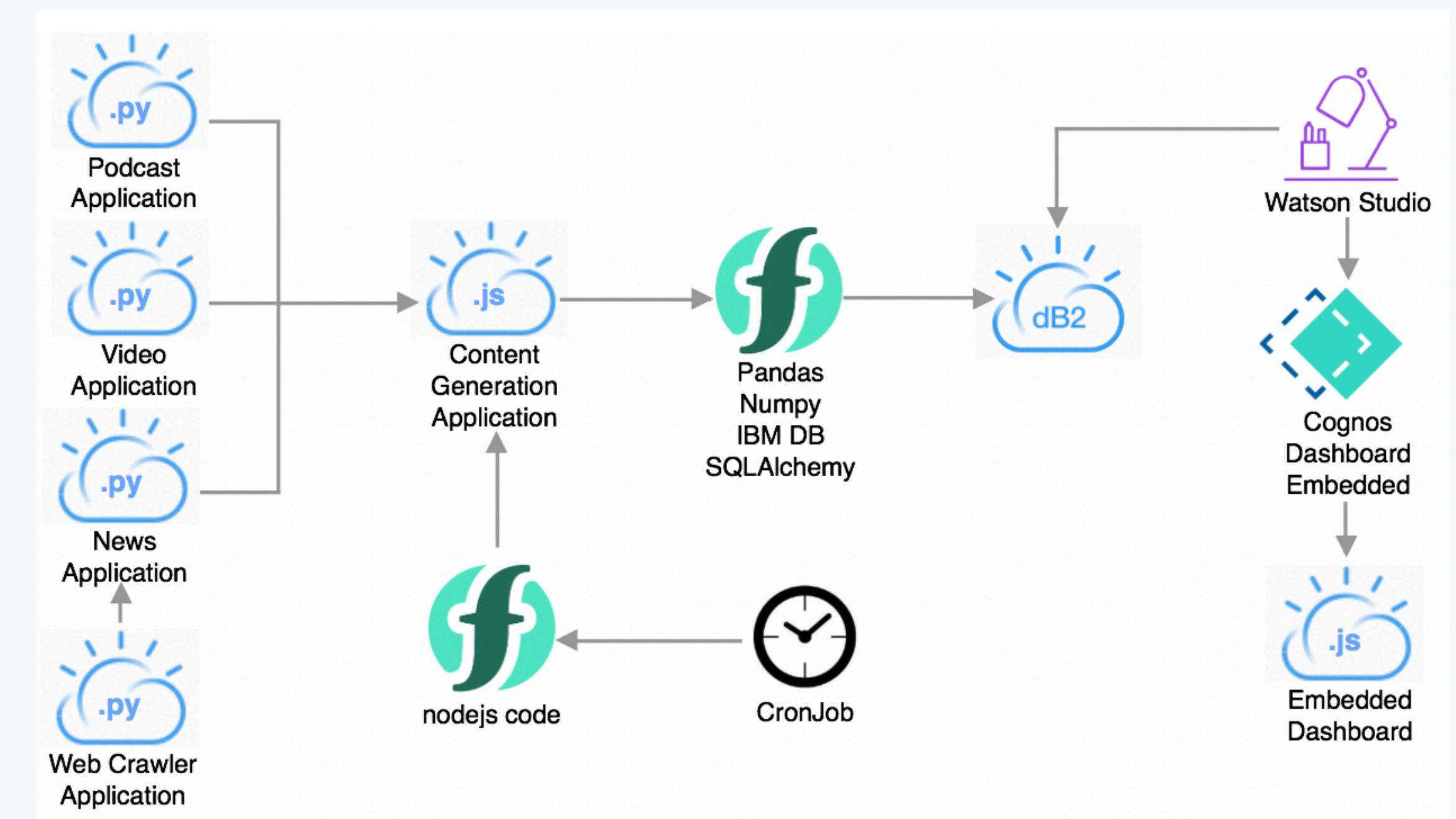
<https://apps.apple.com/us/app/weather-gods/id1041512978>



ESPN Fantasy Football

Cloud Functions is used to **compute the content** of user dashboards

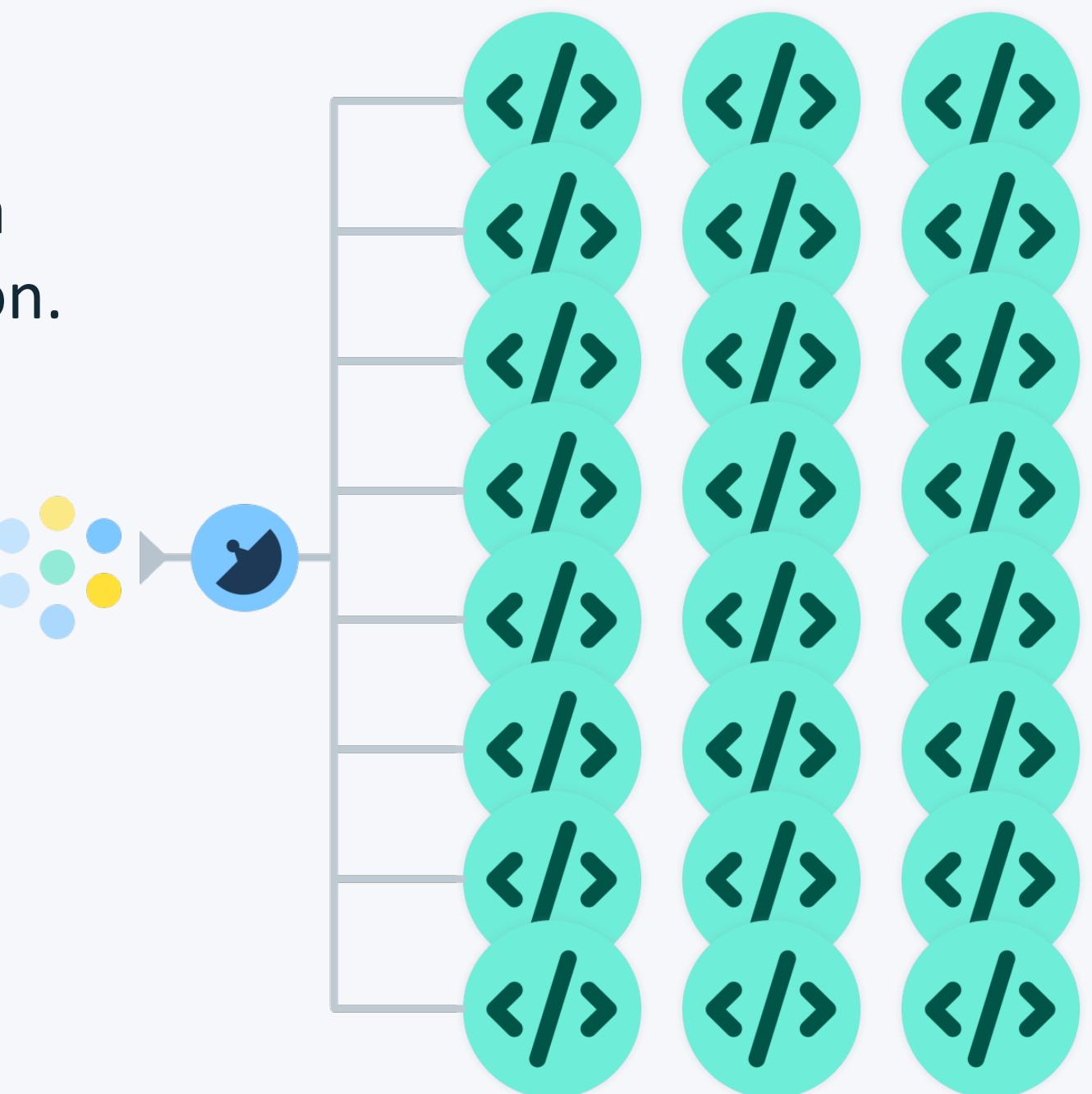
<https://developer.ibm.com/articles/watson-behind-the-code-fantasy-football-2018-part7>



Operations Research / Combinatorial Optimization

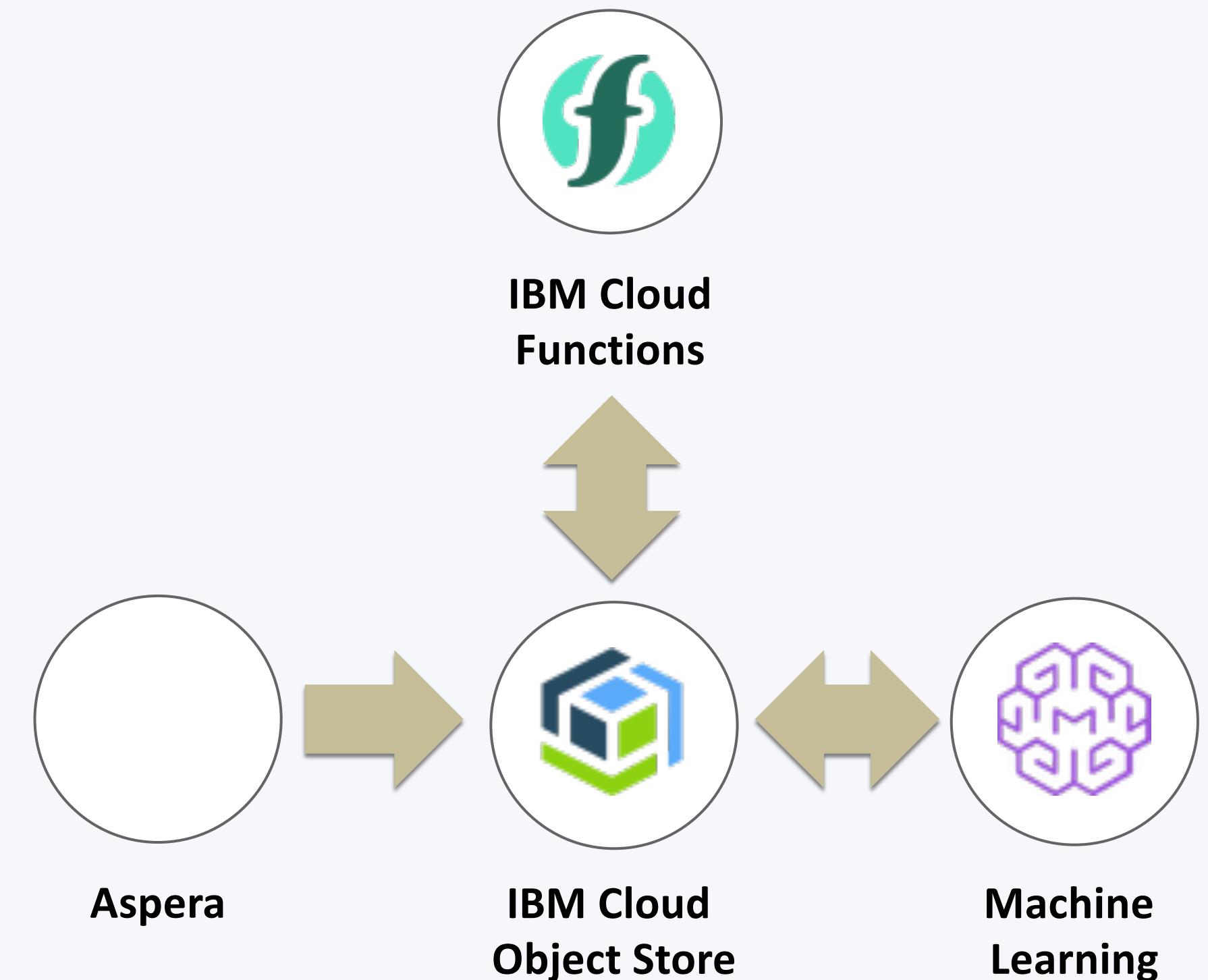
Any kind of **embarrassingly parallel task** is well-suited to be run on a serverless runtime. Each parallelizable task results in one action invocation.

- Map(-Reduce) operations
- Monte-Carlo Simulations
- Genetic Algorithms
- Hyperparameter tuning
- Web scraping
- Genome processing

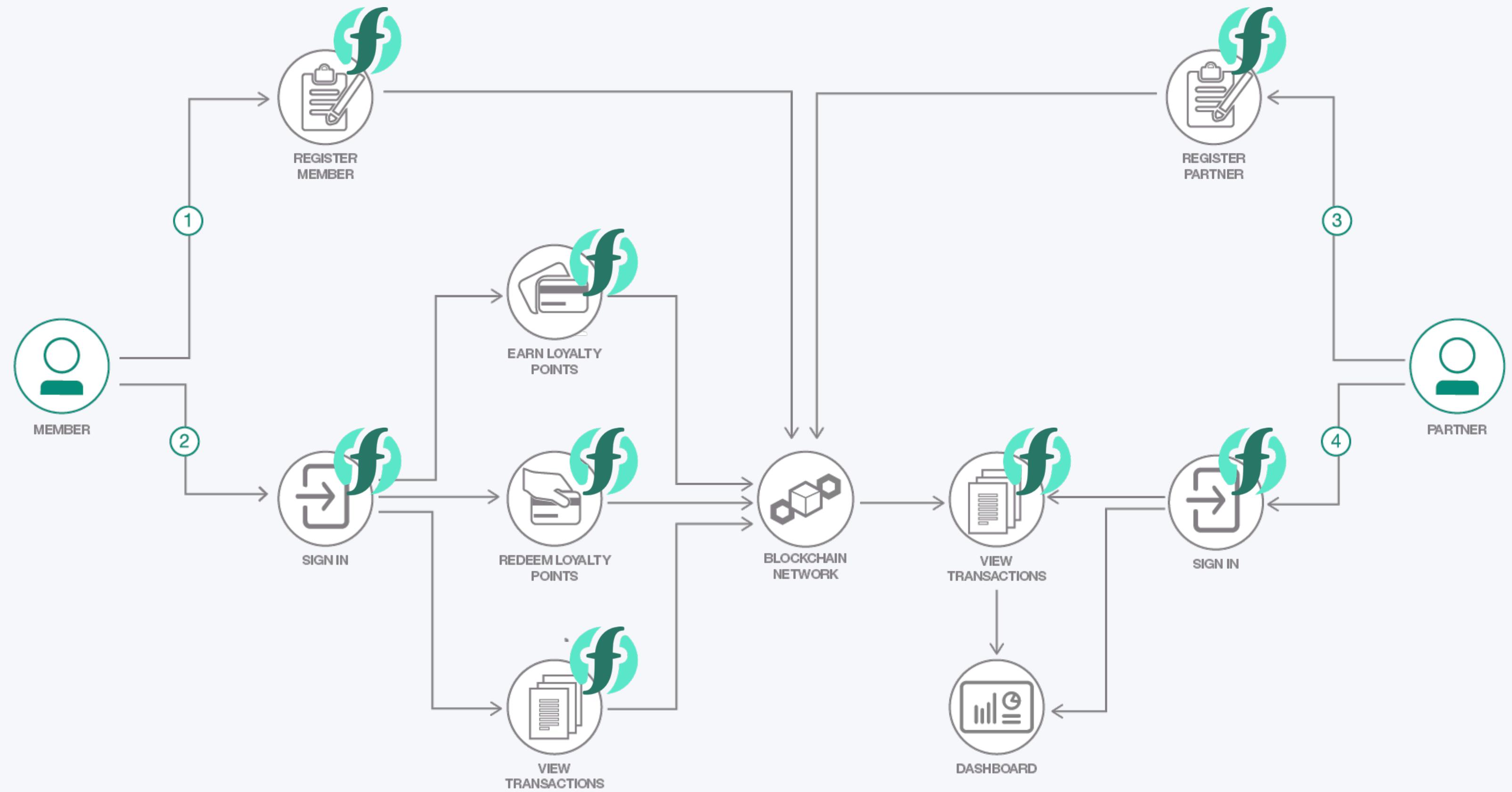


Example: Serverless AI Pipeline

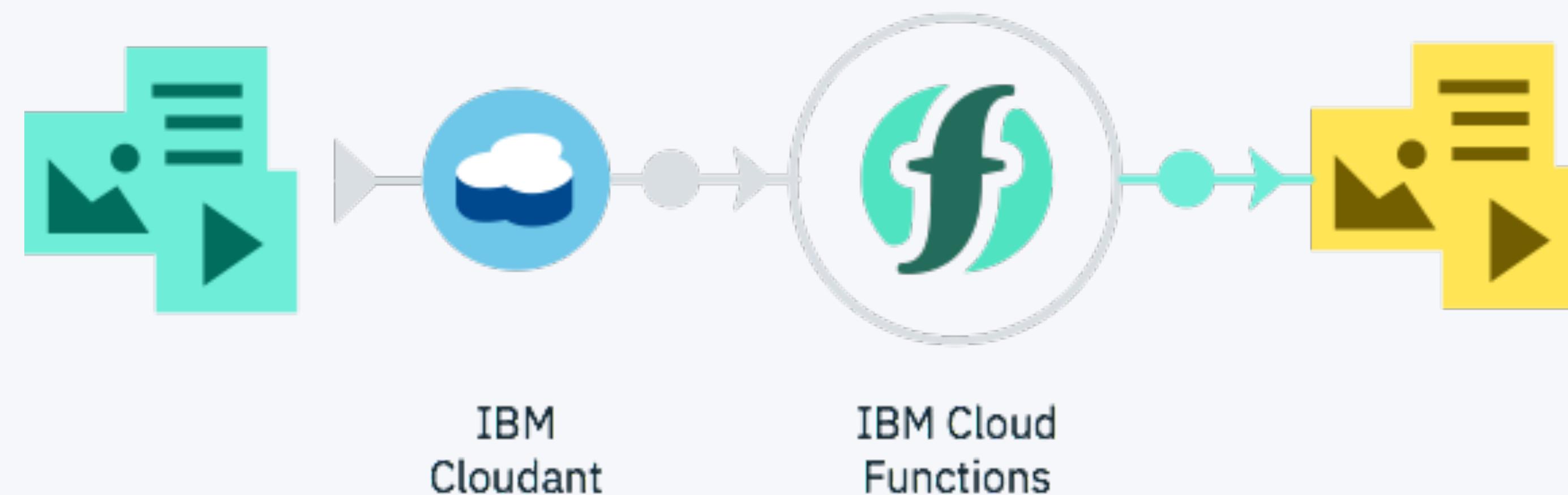
1. Serverless upload with Aspera
2. Serverless storage with COS
3. Serverless data preparation with Cloud Functions and PyWren
4. Machine learning with Watson Machine Learning



Blockchain APIs



Data-at-rest processing & ETL pipelines



Event stream processing via Message Hub



Ideally suited for working with all sorts of data stream ingestions (for validation, cleansing, enrichment, transformation, ...)

Business data streams (from other data sources)

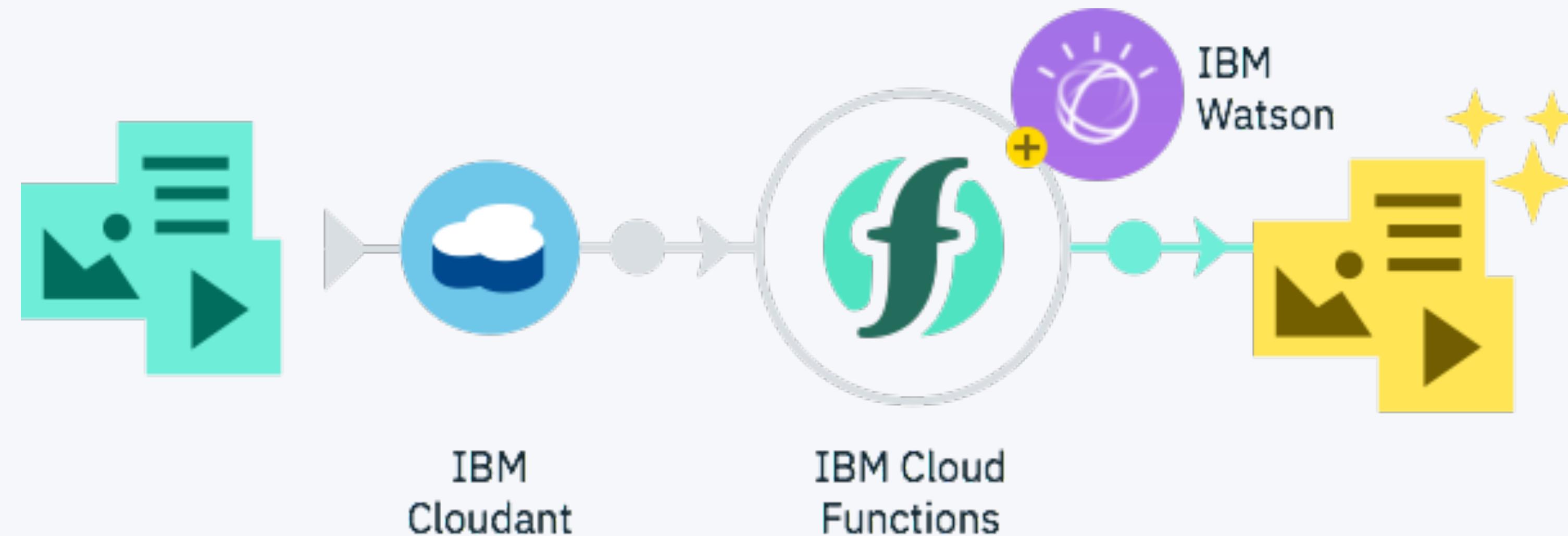
IoT sensor data

Log data

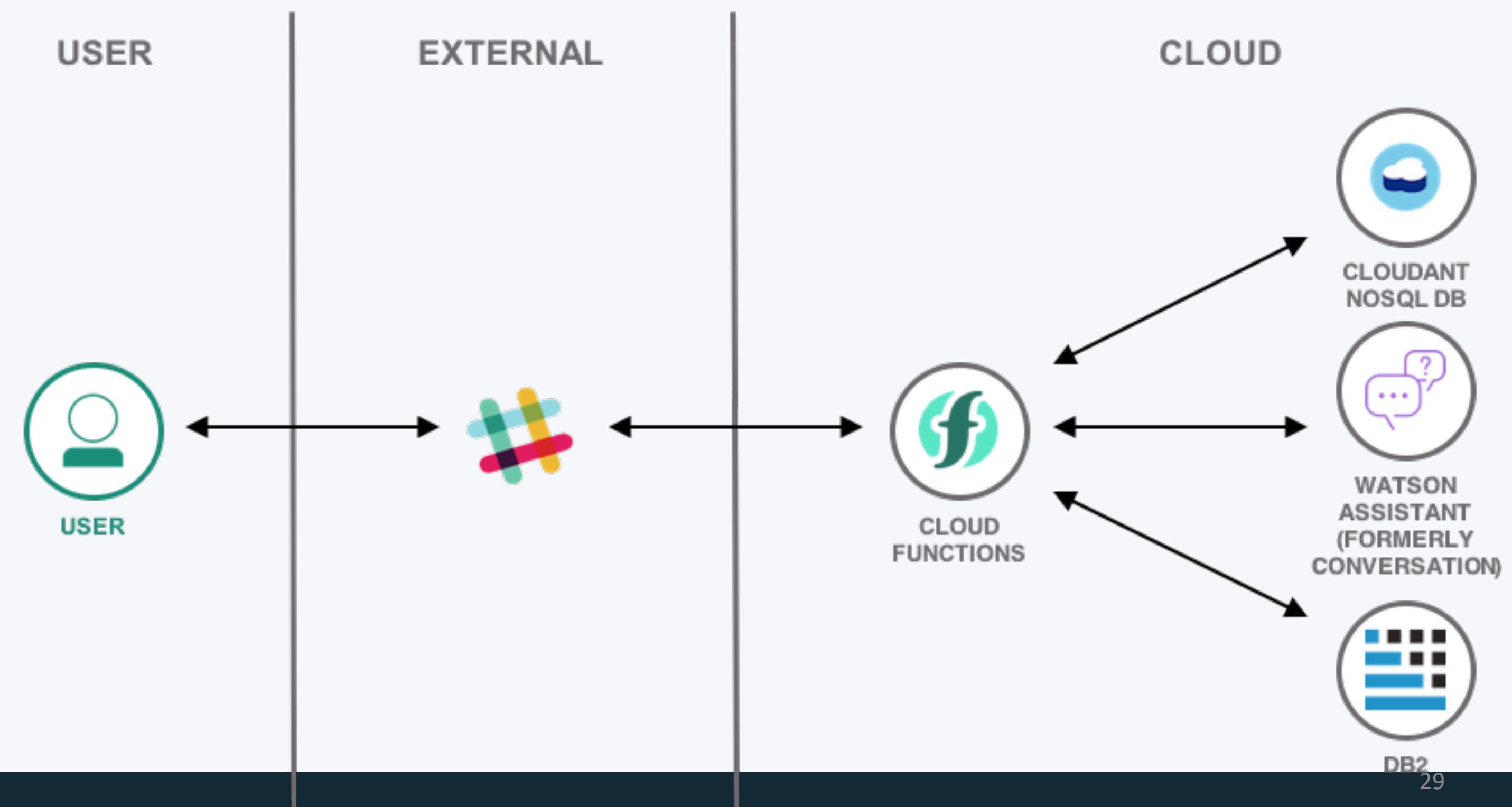
Financial (market) data

...

AI / Cognitive



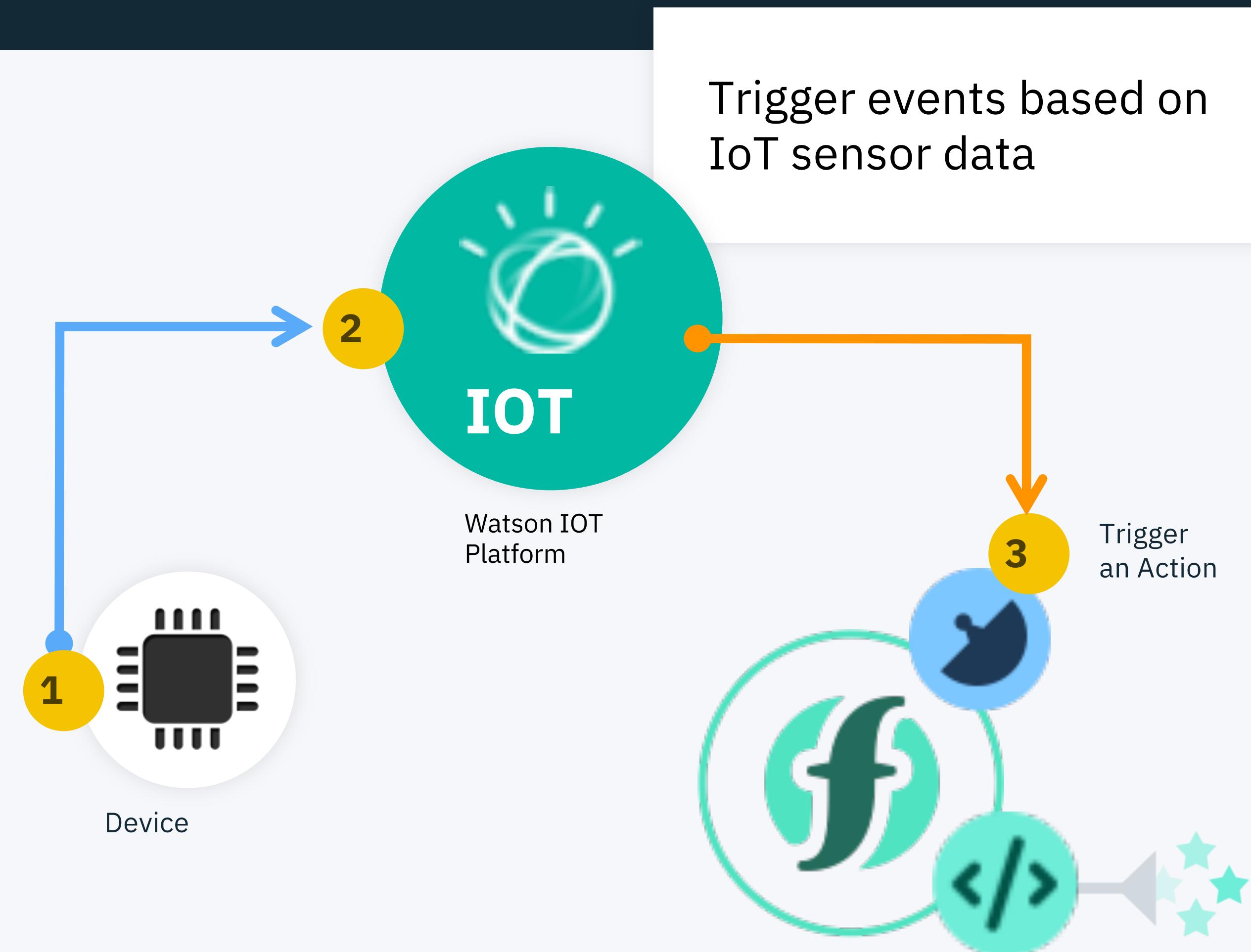
Conversational applications



IoT

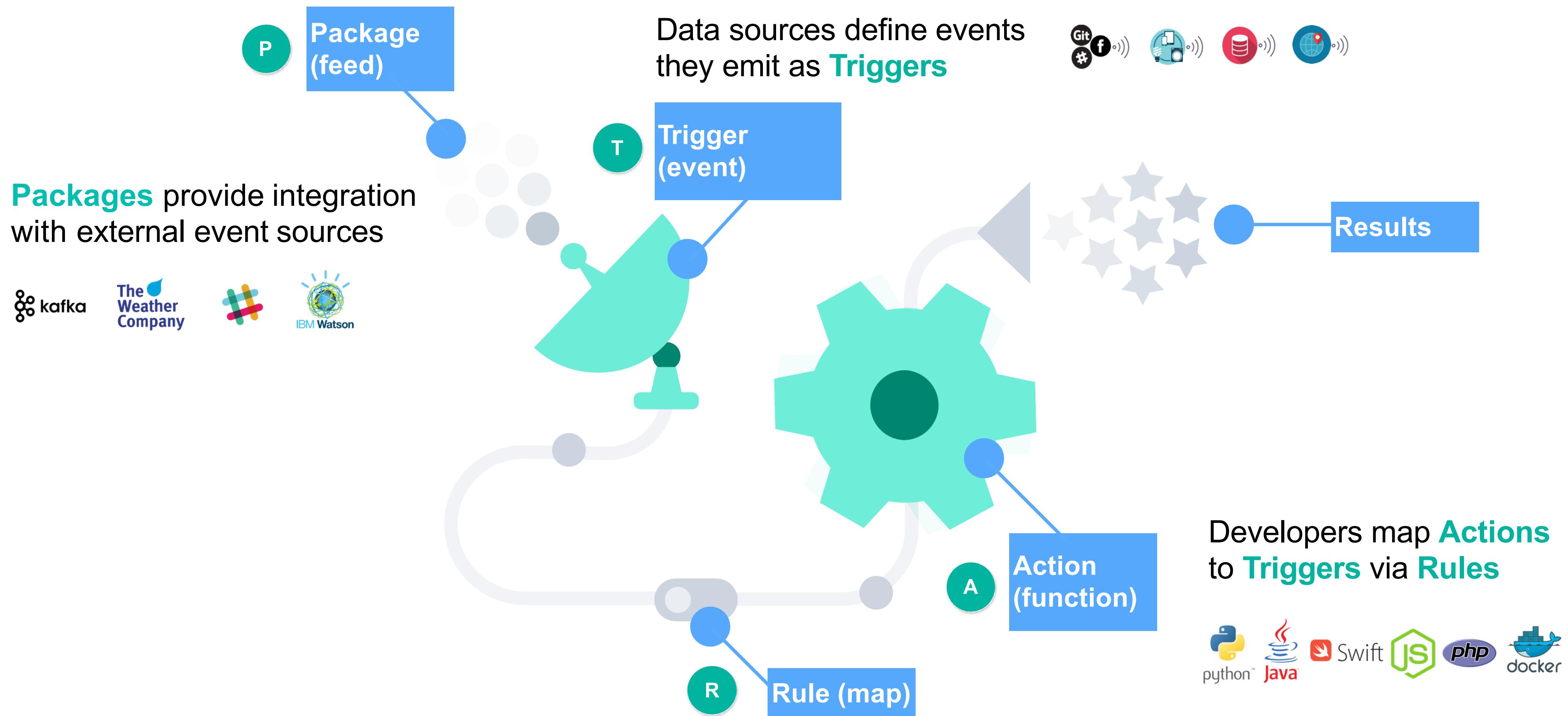
Example, KONE

- Eliminate or reduce the time that KONE equipment goes out of service
- Sensor data from the equipment can reveal potential issues and risks before human eyes and ears can detect them
- Event-driven architecture that allocates the compute resources required to handle each incoming stream of data, automatically scaling as needed



Apache OpenWhisk programming model

Developers work with packages, triggers, actions, and rules



Triggers

T

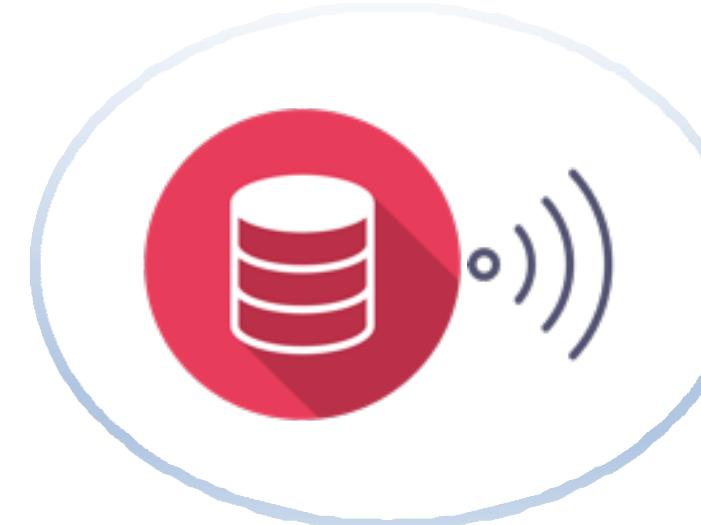
A class of events that can occur



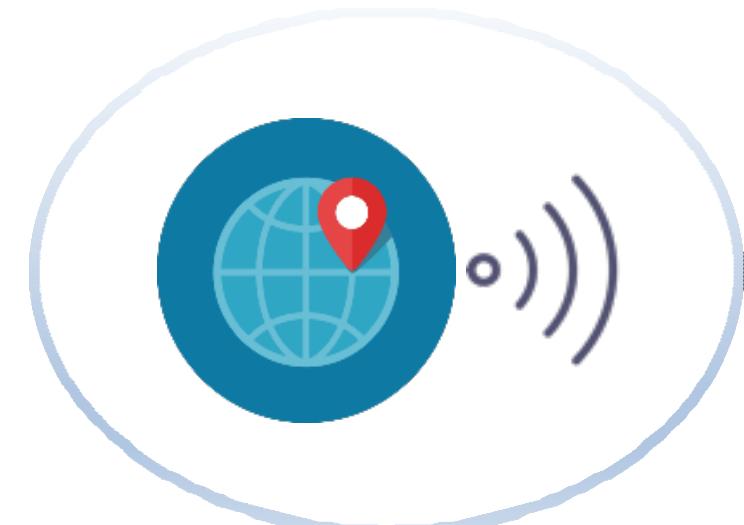
Social events



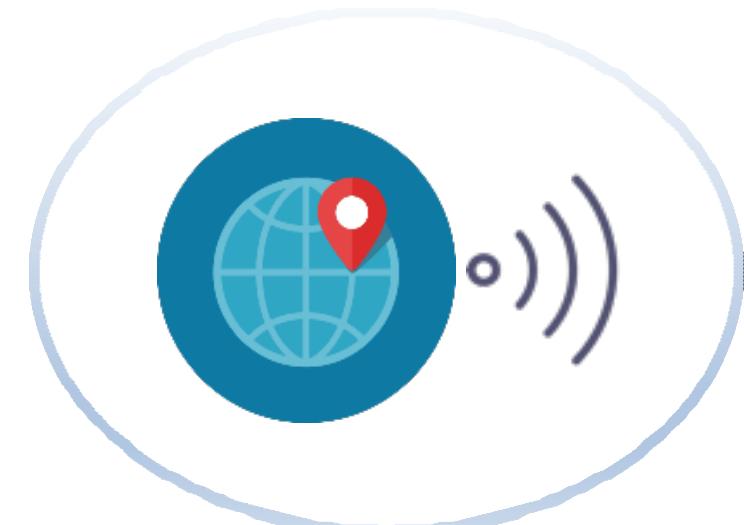
Data changes



Device readings



User input



Location updates

Actions

A *Code that runs in response to an event
(that is, an event handler)*



Actions

A

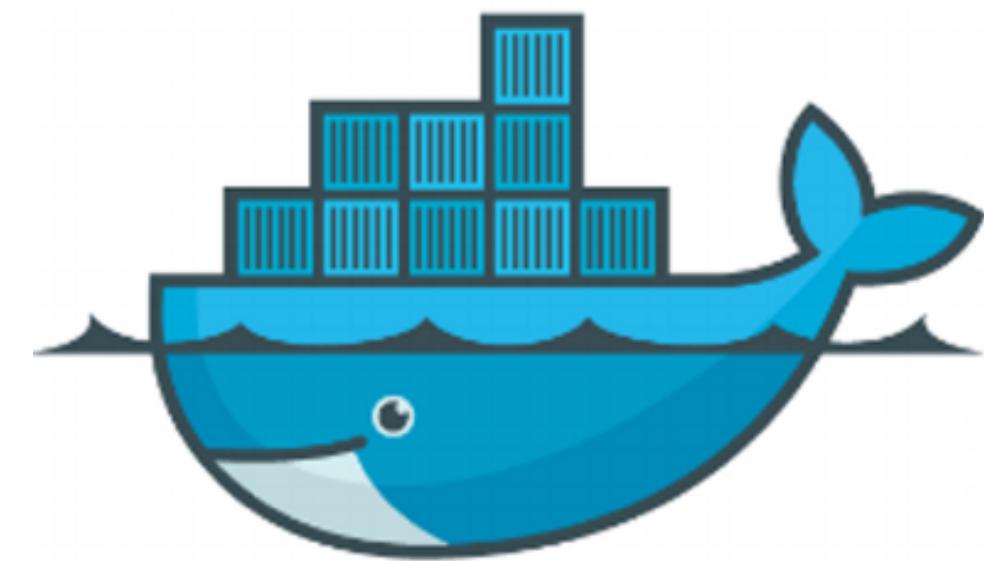
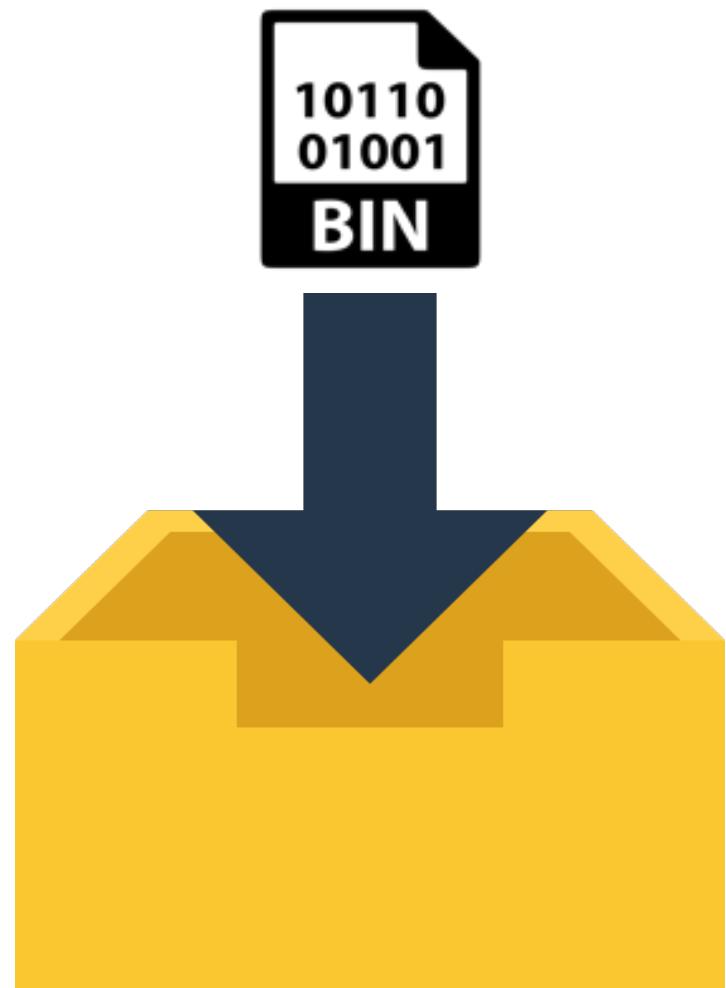
Can be written in a variety of languages, such as
JavaScript, Python, Java, PHP, and Swift

```
function main(params) {  
    return { message: 'Hello, ' + params.name + ' from ' + params.place };  
};
```



Actions

- A Or any other language by packaging with Docker



Actions

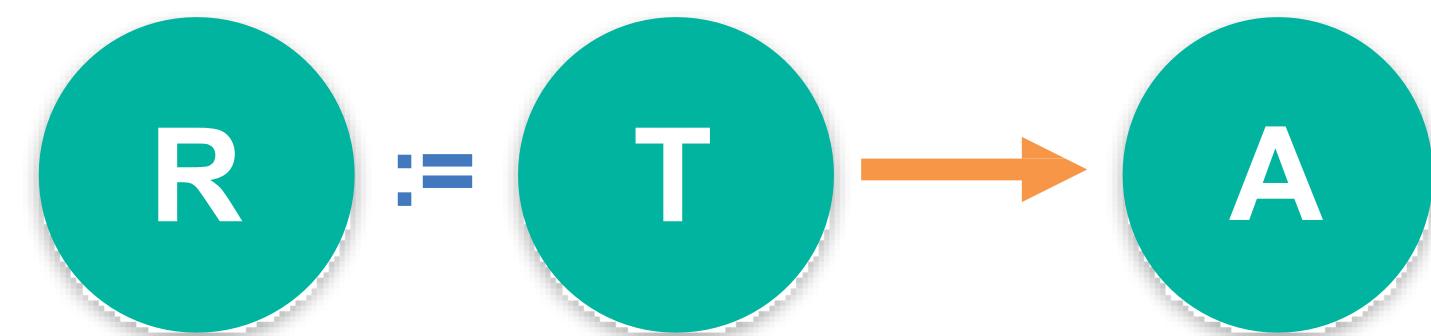
- A Can be composed to create sequences that increase flexibility and foster reuse

$$\begin{aligned} A_A &:= A_1 + A_2 + A_3 \\ A_B &:= A_2 + A_1 + A_3 \\ A_C &:= A_3 + A_1 + A_2 \end{aligned}$$

Rules

R

*An association of a trigger to an action
in a many to many mapping.*



Packages

P

A shared collection of triggers and actions



IBM Cloudant®



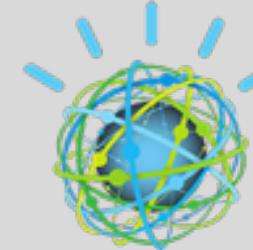
read



write



changes



IBM Watson



translate



A forecast

Open
Source



A post
T topic

Third
Party



T commit

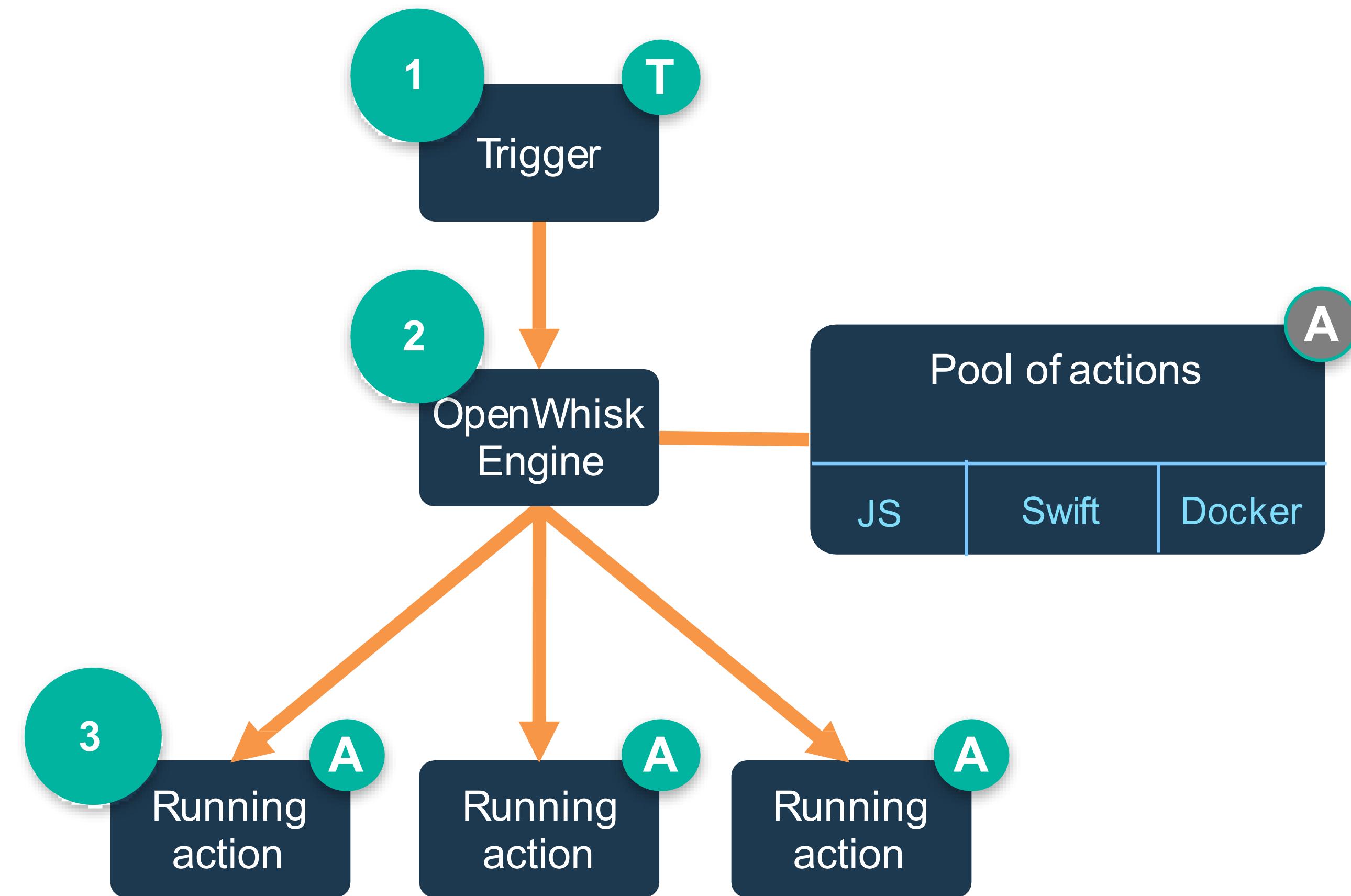
Yours



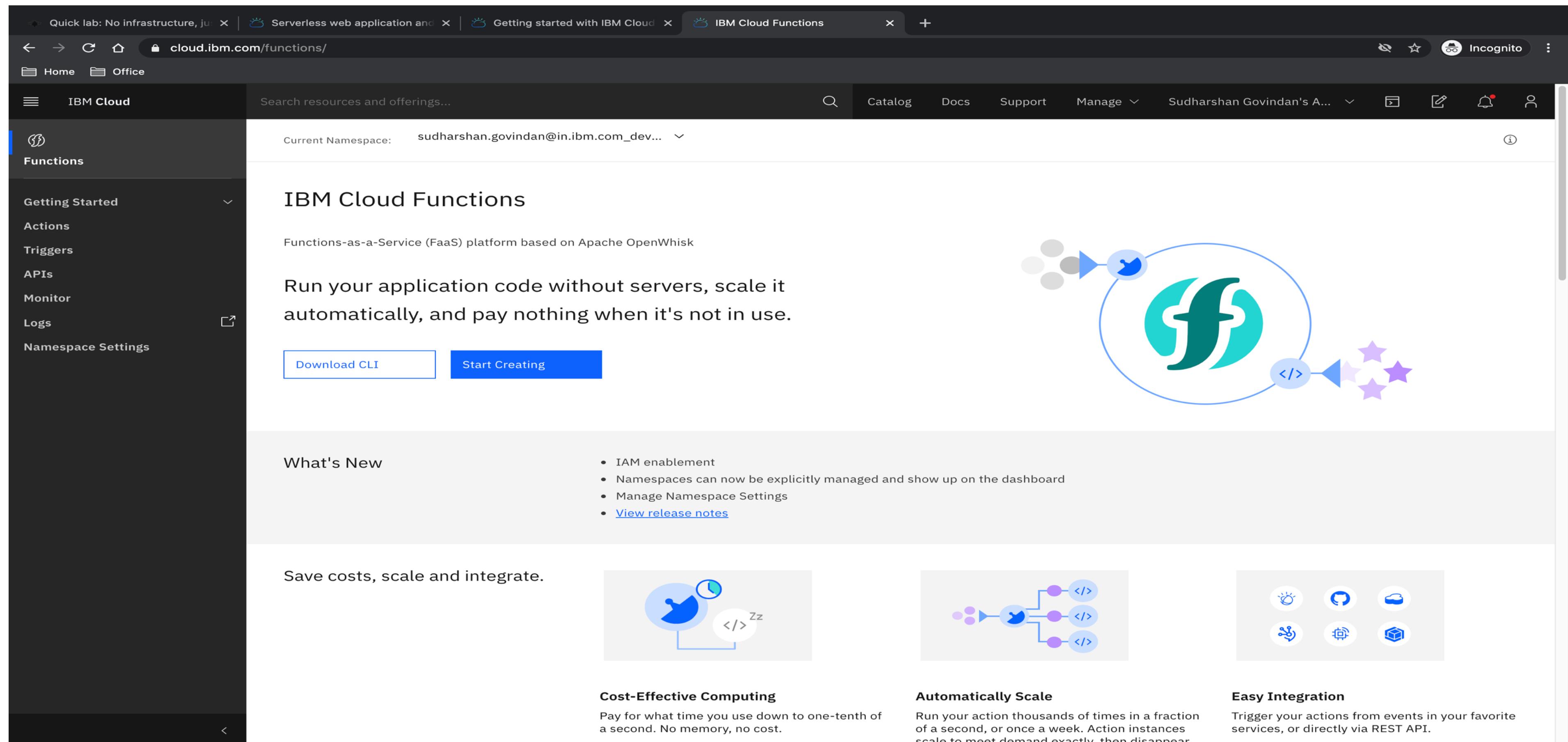
A myAction
T myFeed

Apache OpenWhisk in action

OpenWhisk awaits events, fetches mapped code, runs it in a container



Use Apache OpenWhisk via IBM Cloud Functions: <https://cloud.ibm.com/functions/>



The screenshot shows the IBM Cloud Functions dashboard. The top navigation bar includes tabs for "Quick lab: No infrastructure, ju...", "Serverless web application and...", "Getting started with IBM Cloud", and "IBM Cloud Functions". The main content area features the "IBM Cloud Functions" logo (a stylized green 'ff' inside a blue circle) surrounded by a network of nodes and arrows. Below the logo, the text "Run your application code without servers, scale it automatically, and pay nothing when it's not in use." is displayed. A "Download CLI" button and a "Start Creating" button are present. The left sidebar contains sections for "Getting Started", "Actions", "Triggers", "APIs", "Monitor", "Logs", and "Namespace Settings". The "Namespace Settings" section is currently selected. The "What's New" section lists recent changes: "IAM enablement", "Namespaces can now be explicitly managed and show up on the dashboard", "Manage Namespace Settings", and a link to "View release notes". The "Save costs, scale and integrate." section includes icons for a timer and a sleep icon, a branching flowchart, and a grid of integration icons (IFTTT, GitHub, Slack, etc.). The "Cost-Effective Computing" section states "Pay for what time you use down to one-tenth of a second. No memory, no cost." The "Automatically Scale" section states "Run your action thousands of times in a fraction of a second, or once a week. Action instances scale to meet demand exactly, then disappear." The "Easy Integration" section states "Trigger your actions from events in your favorite services, or directly via REST API."

Code Editor

IBM Cloud Catalog Docs Support Manage Search for resource... 1609129 - DAVID TRO...

Functions / Actions / exec-query-find Region: US South Org: davetropeano@us.ibm.com Space: dev

Bluemix_tododb_tododb_key/exec-qu... Read-only

Code Parameters Runtime Endpoints Connected Triggers Enclosing Sequences Logs

Code Node.js 6 Change Input Invoke

```
1 /**
2  * Query using a Cloudant Query index:
3  * https://docs.cloudant.com/cloudant_query.html#finding-documents-using-an-index
4  */
5
6 function main(message) {
7   var cloudantOrError = getCloudantAccount(message);
8   if (typeof cloudantOrError !== 'object') {
9     return Promise.reject(cloudantOrError);
10  }
11  var cloudant = cloudantOrError;
12  var dbName = message dbname;
13  var query = message.query;
14
15  if(!dbName) {
16    return Promise.reject('dbname is required.');
17  }
18  if(!query) {
19    return Promise.reject('query field is required.');
20  }
21  var cloudantDb = cloudant.use(dbName);
22
23  if (typeof message.query === 'object') {
24    query = message.query;
25  } else if (typeof message.query === 'string') {
26    try {
27      query = JSON.parse(message.query);
28    } catch (e) {
29      return Promise.reject('query field cannot be parsed. Ensure it is valid JSON.');
30    }
31  } else {
32    return Promise.reject('query field is ' + (typeof query) + ' and should be an object or a JSON string.');
33  }
34
35  return queryIndex(cloudantDb, query);
36}
37
38
39 function queryIndex(cloudantDb, query) {
40  return new Promise(function(resolve, reject) {
```

IBM Cloud Functions provides management, tooling, and monitoring

The screenshot shows the IBM Bluemix OpenWhisk Invocation Console interface. The top navigation bar includes 'Docs', '101 Trial Days Remaining', 'Daniel Krook's Account | US South : krook@us.ibm.com : dev', and account management links. The main menu has tabs for 'Getting Started', 'Manage', 'Develop' (selected), 'Monitor', and 'APIs'. On the left, a sidebar lists 'MY ACTIONS' (Hello World, Hello World With Params, alert-customer-event, analyze-service-event, check-warranty-renewal, create-order-event), 'MY SEQUENCES' (order-sequence, service-sequence), 'MY RULES' (check-warranty-rule, openfridge-feed-rule, order-rule, service-rule), and 'MY TRIGGERS'. The central 'Invocation Console' area displays three recent invocations of the 'Hello World' action:

- Invocation 1:** Invoked at 1:32:26 PM, completed in 813ms, billed for 51ms. Response: { "message": "hello world" }
- Invocation 2:** Invoked at 1:32:30 PM, completed in 721ms, billed for 3ms. Response: { "message": "hello world" }
- Invocation 3:** Invoked at 1:32:33 PM, completed in [partially visible]. Response: { "message": "hello world" }

Buttons at the bottom right include 'Clear Console', 'Run Again', and 'Close'.

Action / Trigger Monitoring

IBM Cloud

Getting Started Actions Triggers Monitor APIs

REGION: US South CLOUD FOUNDRY ORG: eweiter@us.ibm.com CLOUD FOUNDRY SPACE: dev

Activity Summary Invocation counts per action or rule

Action	Activations	Avg Time
1234123412341234	1	43.62ms
2missMachine	13	43.62ms
anActionwithaverylongname...	1	56ms on average
aVulfW	3	3.33ms on average
aVulfW_1234123412341234	1	67ms on average
bubbleGum	3	2ms on average
bubbleGum_1234123412341...	1	63ms on average
buffalos	3	3.67ms on average
buffalos_1234123412341234	1	64ms

Activity Log

Action	Time	Duration
changes	1/19/2018 7:44:13 PM	896ms
2missMachine	1/19/2018 11:18:49 AM	82ms
2missMachine	1/19/2018 10:20:46 AM	63ms
2missMachine	1/19/2018 9:54:39 AM	64ms

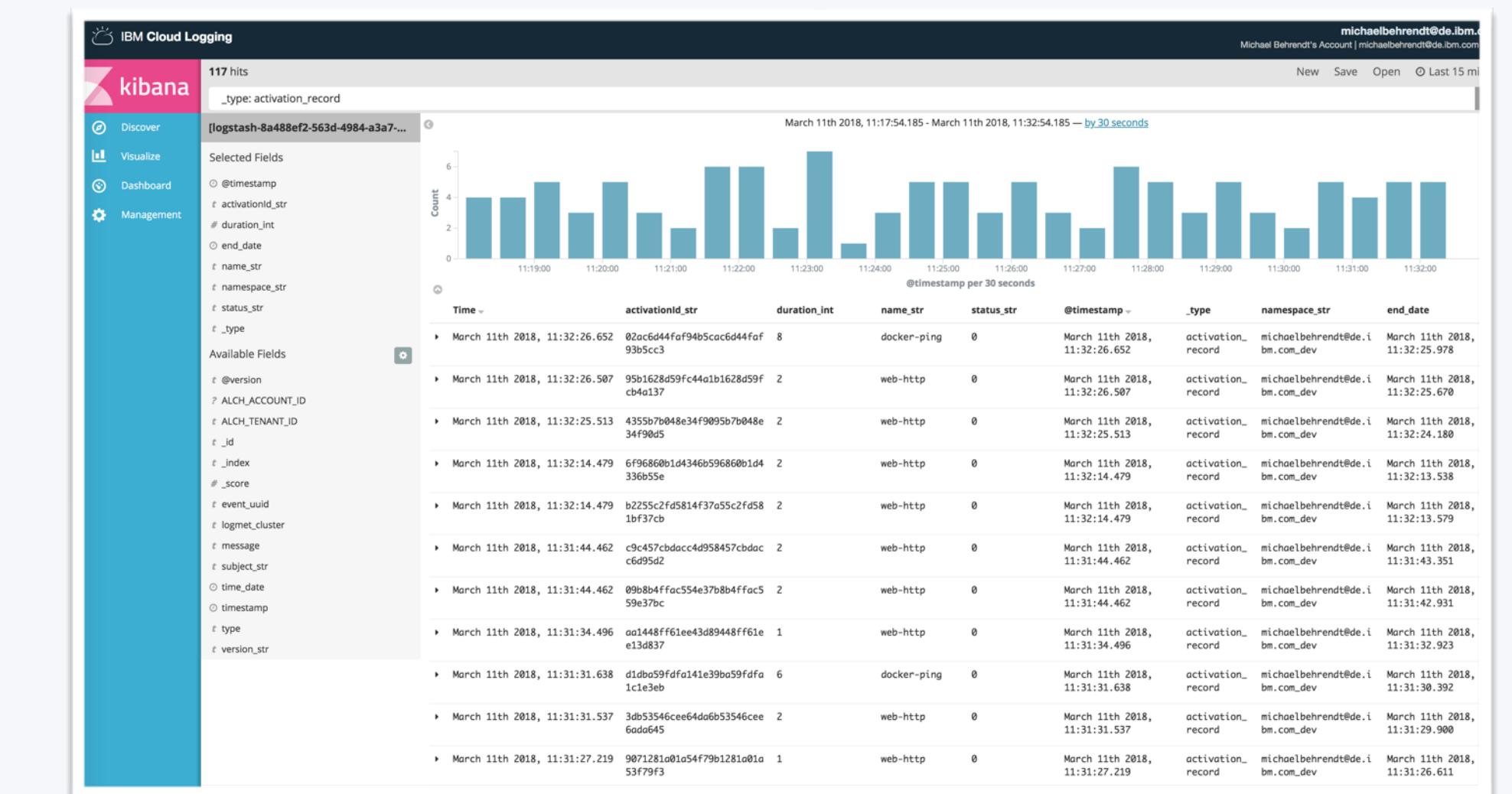
Activity Timeline Invocation count over time

Filtering Options

Time Frame: most recent 50 Limit to: All Actions Exclude triggers from the views?

Integration with IBM Logging Service

- All logs and activation records automatically available via IBM Logging Service
- Provides rich Kibana-based experience for searching logs in any dimension
- Critical for problem determination at scale
- Includes visualization capabilities which are critical for quickly detecting trends and anomalies



IBM Cloud Functions also provides included API gateway integration

The screenshot shows the IBM Bluemix Functions API definition page for a 'Cats API'. The top navigation bar includes 'Docs', '323 Trial Days Remaining', 'Daniel Krook's Account | US South : krook@us.ibm.com : functions17', and 'Catalog Support Manage'. On the left, a sidebar lists 'All Cloud Functions APIs', 'Summary', 'Definition' (which is selected), 'Sharing', and 'API Explorer'. The main content area is titled 'Cats API' with a network icon. It features an 'Expose Managed API' toggle switch (set to off) and a help icon. The 'Definition' tab contains sections for 'API definition file' (with a dropdown menu), 'API name *' (set to 'Cats API'), 'Base path for API *' (set to '/v1'), and 'Operations *' (with a 'Create operation +' button). A table lists two operations: a POST to '/cat' using the 'create-cat' action and a GET to '/cat' using the 'fetch-cat' action.

PATH	VERB	PACKAGE	ACTION
/cat	POST		create-cat
/cat	GET		fetch-cat

Connecting to on prem-resources

Common **requirements**:

- Receive events from on-premise resources (e.g. Kafka)
- Writing data into on-premise data stores from IBM Cloud Functions
- Call on-prem APIs

Solution:

- Leverage **IBM Secure Gateway for IBM Cloud Functions** to communicate with your on-prem data center



Compliance standards

Certified for...

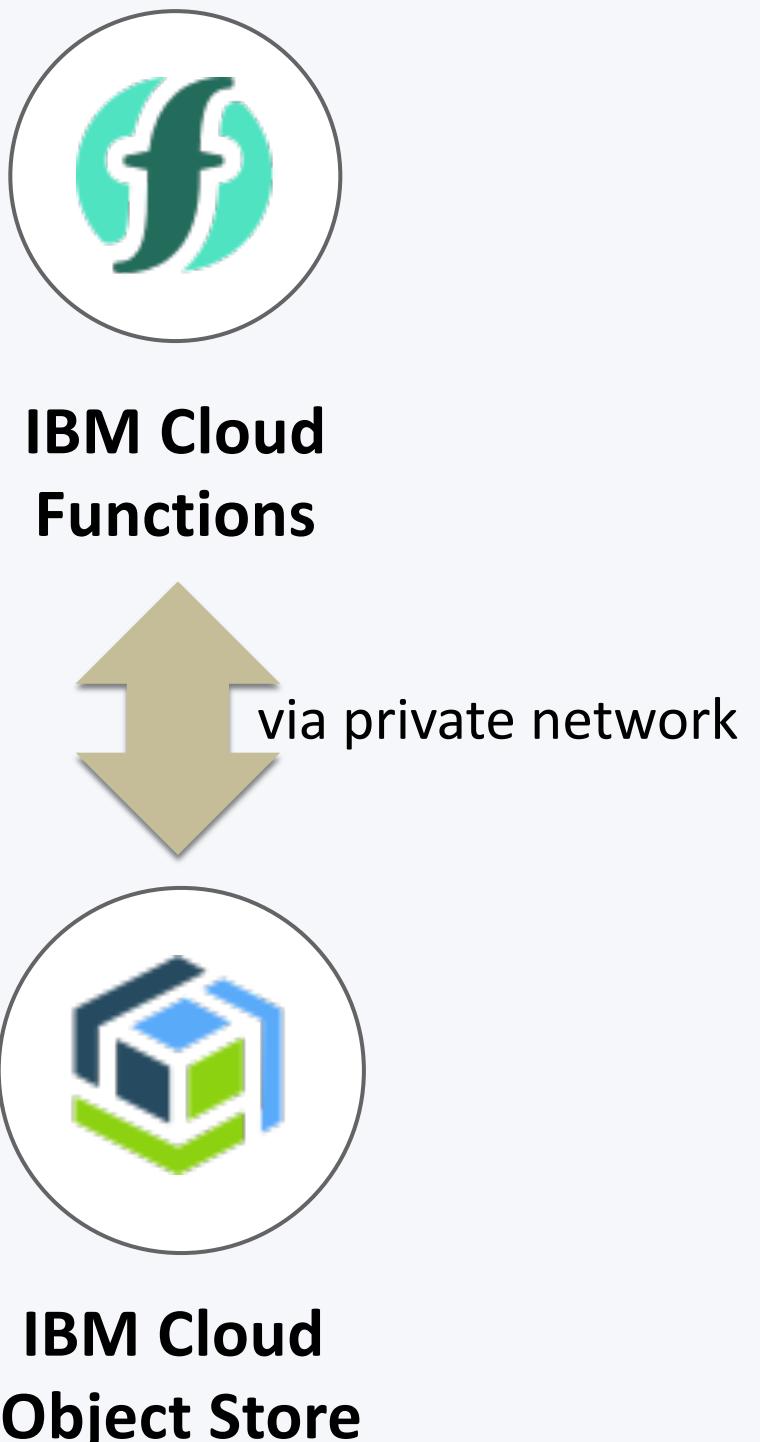
- **ISO 27001**
 - specifies a management system that is intended to bring information security under management control and gives specific requirements
- **ISO 27017**
 - gives guidelines for information security controls applicable to the provision and use of cloud services
- **ISO 27018**
 - provides guidance aimed at ensuring that cloud service providers offer suitable information security controls to protect the privacy of their customers' clients by securing PII
- **GDPR**
- **HIPAA**
- **SOC2**



COS private endpoint

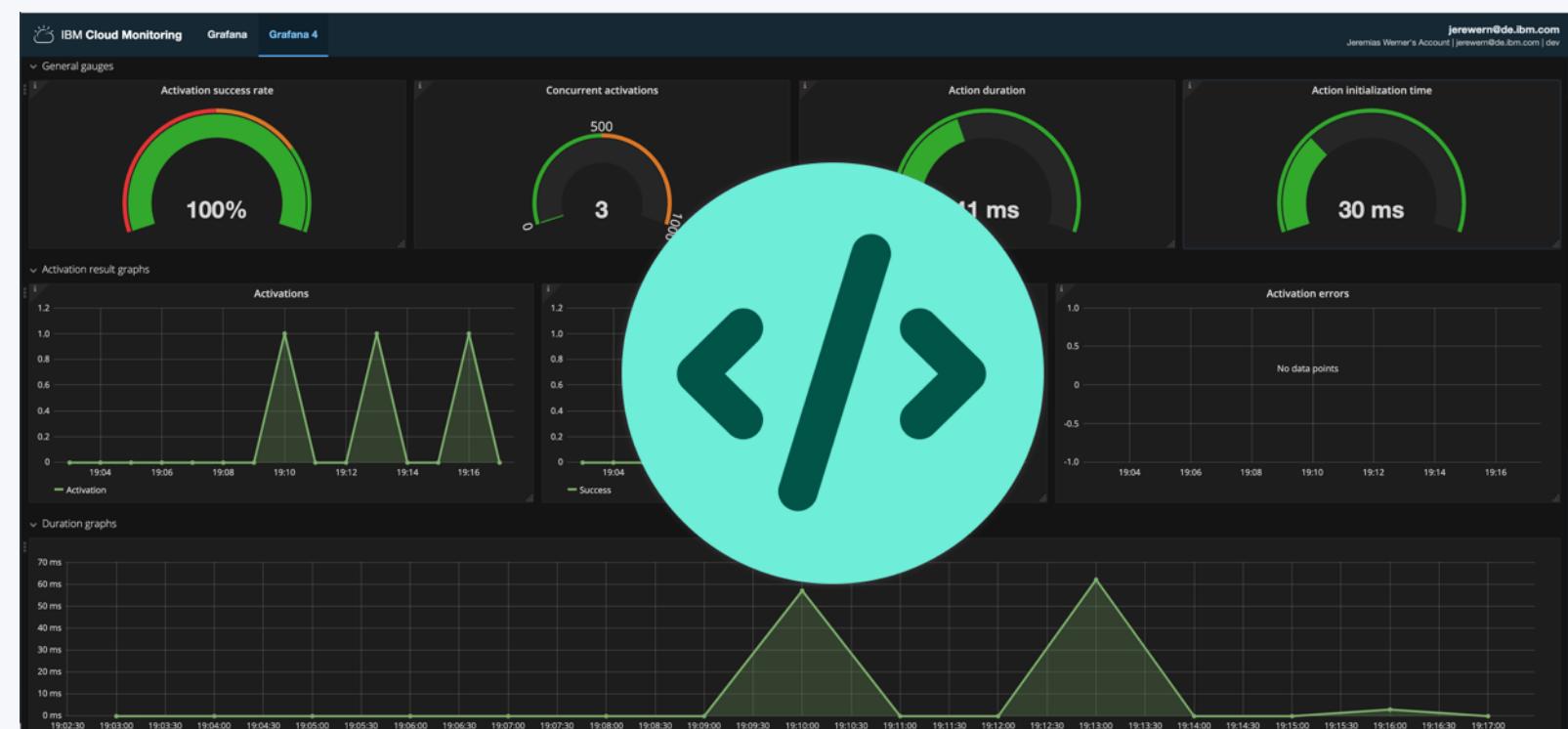
Connect via the private endpoints which:

- **allows higher throughput** and
- **do not incur charges for in/outgoing bandwidth**



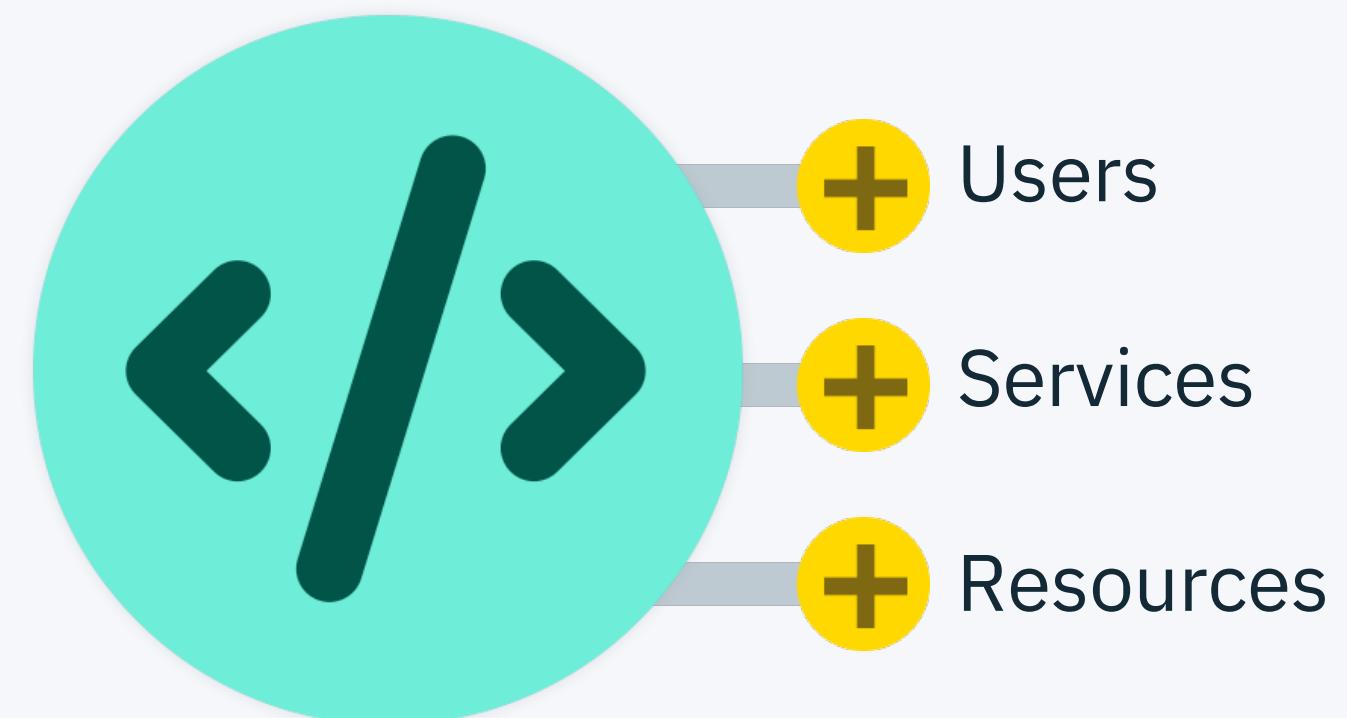
Integration with Monitoring Service

Empower your DevOps team for **insight** into how your functions are performing. Quickly **identify trends**, detect and **diagnose problems**.



IAM support

Authenticate users and **control access to resources** consistently across IBM Cloud

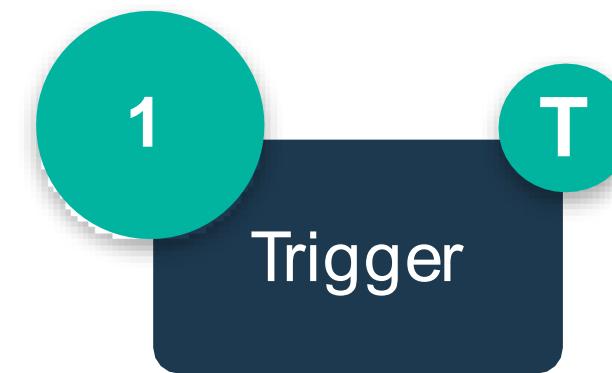


Use case 1: Your first trigger, action, and rule

```
function main(params) {  
  var date = new Date();  
  console.log("Invoked at: " + date.toLocaleString());  
  return { message: "Invoked at: " + date.toLocaleString() };  
}
```

```
$ ibmcloud fn activation poll
```

```
$ ibmcloud fn action create handler handler.js
```



1. Cron syntax alarm

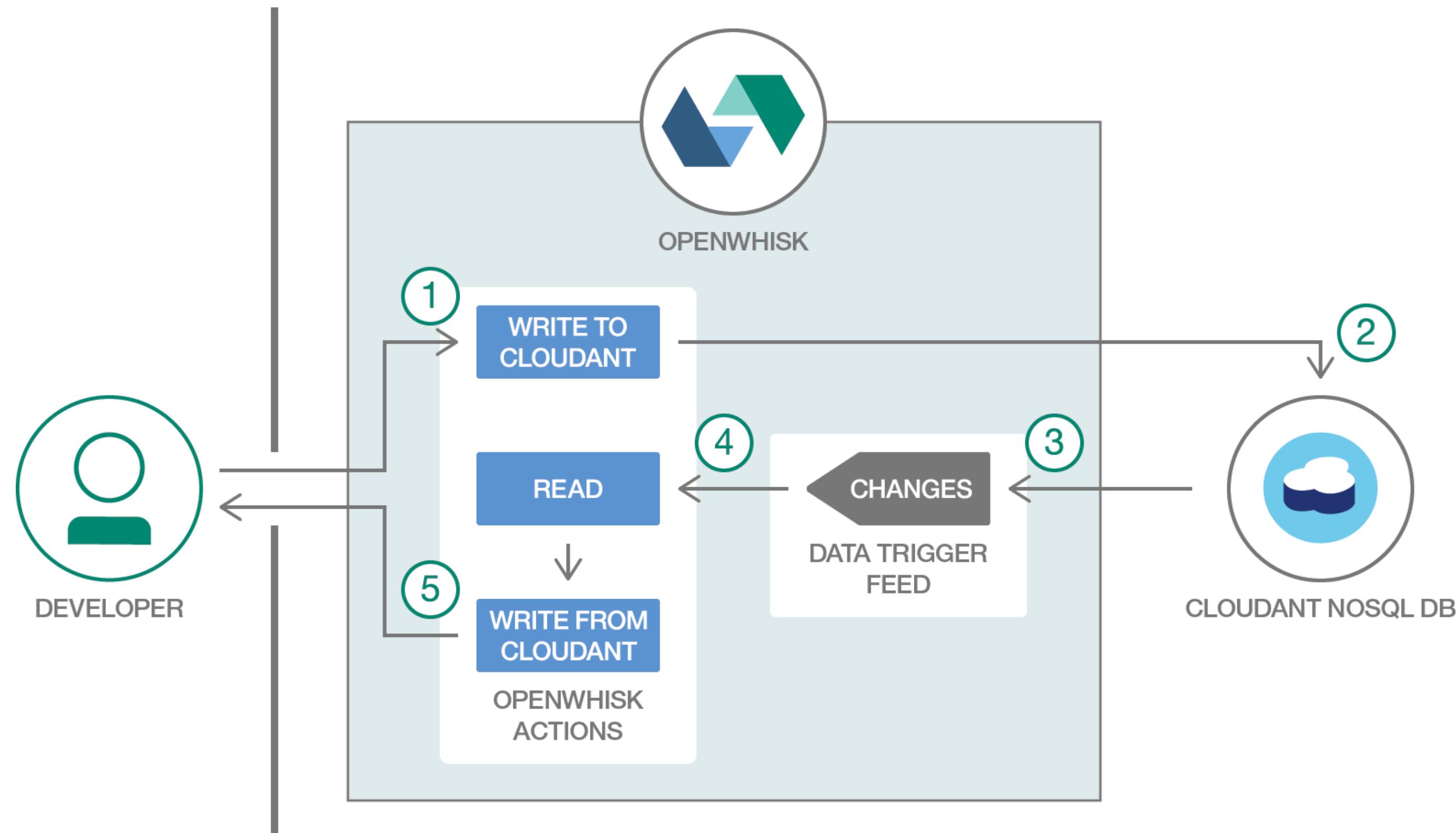
```
$ ibmcloud fn action invoke --blocking handler
```

```
$ ibmcloud fn trigger create every-20-second \  
  --feed /whisk.system/alarms/alarm \  
  --param cron "*/20 * * * *" \  
  --param maxTriggers 15
```

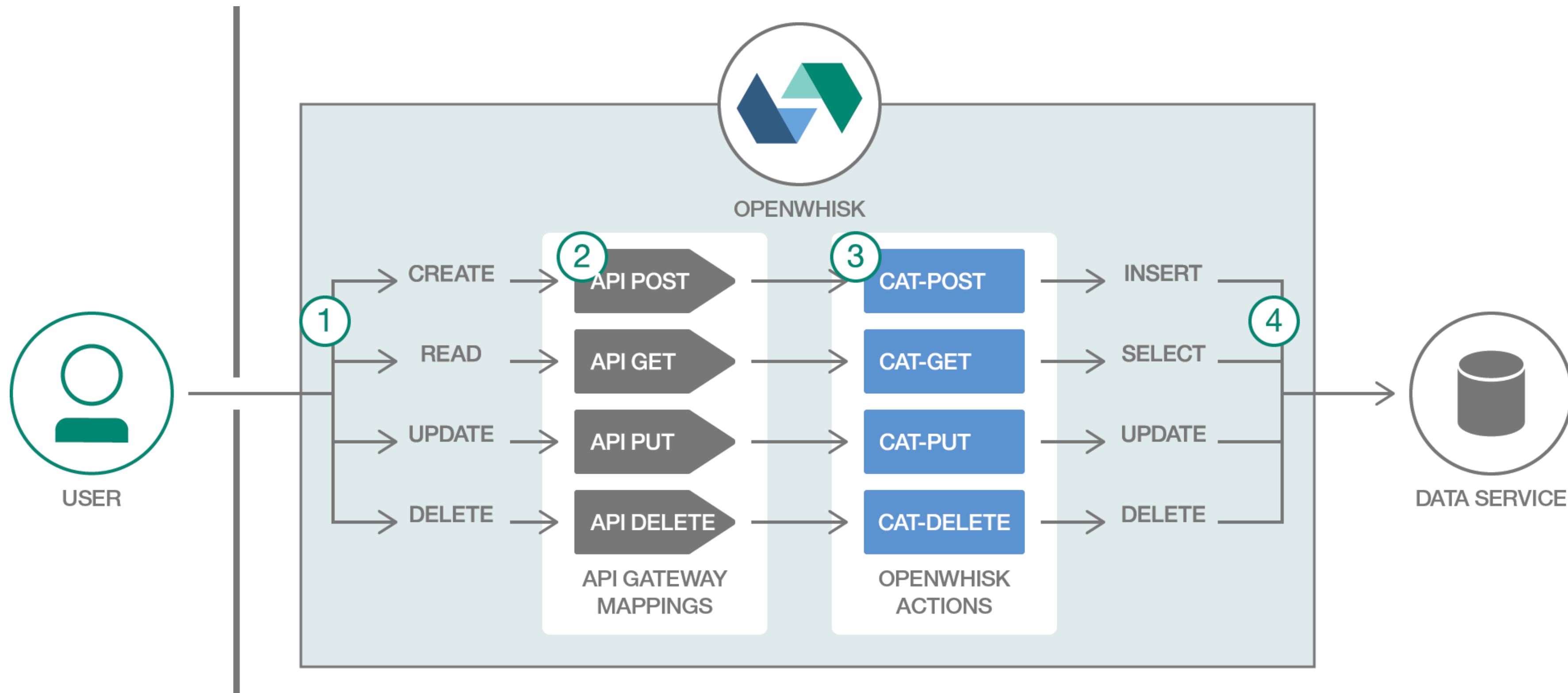
2. Log the current time

```
$ ibmcloud fn rule create \  
  invoke-periodically \  
  every-20-seconds \  
  handler
```

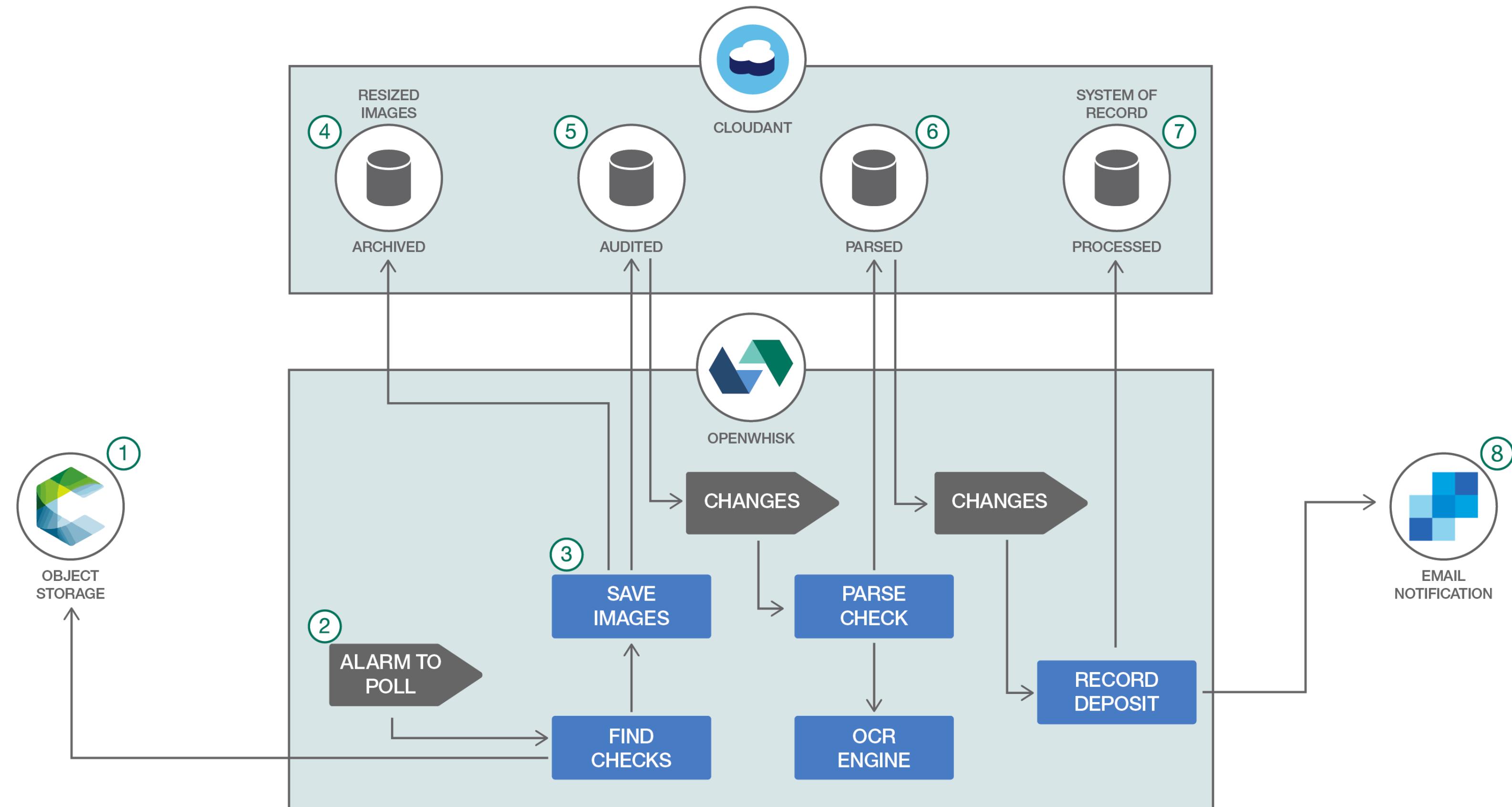
Use case 2: Database change triggered action



Use case 3: HTTP API request triggered action



Other simple apps, working solutions, more complex uses

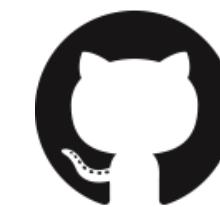


Get started with Cloud Functions, or explore the open source project

Managed OpenWhisk with
IBM Cloud Functions

cloud.ibm.com/functions

Delivered as
Open source via Apache
openwhisk.org



github.com/openwhisk



slack.openwhisk.org



twitter.com/openwhisk



medium.com/openwhisk

IBM Cloud Functions

Thank You!

