Serverless (...on a server!?)

Elton de Souza

IBM

eltondesouza

Battle Plan

What is serverless

When you should use serverless technology

Design patterns for serverless

Best practices for consuming serverless

Benefits of using IBM Z/LinuxONE as a backing platform for serverless

Live Demo!

Serverless is..

- A different **consumption** model
- A different **pricing** model
- Typically, used to run **ephemeral** tasks
- Typically, used to run **stateless** workloads

Serverless is not..

- A piece of **technology**
- The **panacea**
- The **default** pattern for green field applications
- The **goal** of modernization
- **Free** {of complexity}

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**

No servers

Just code

Traditional model

Worry about scaling

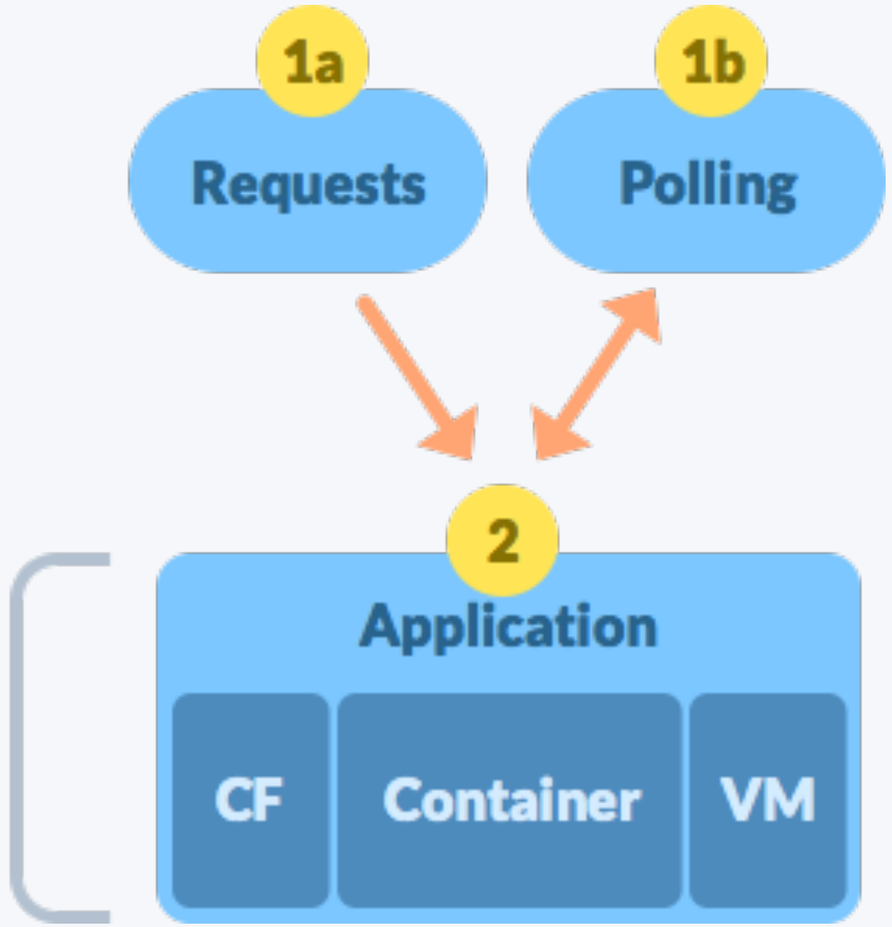
- When to scale? (mem-, cpu-, response time-, etc. driven?)
- How fast can you scale?

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



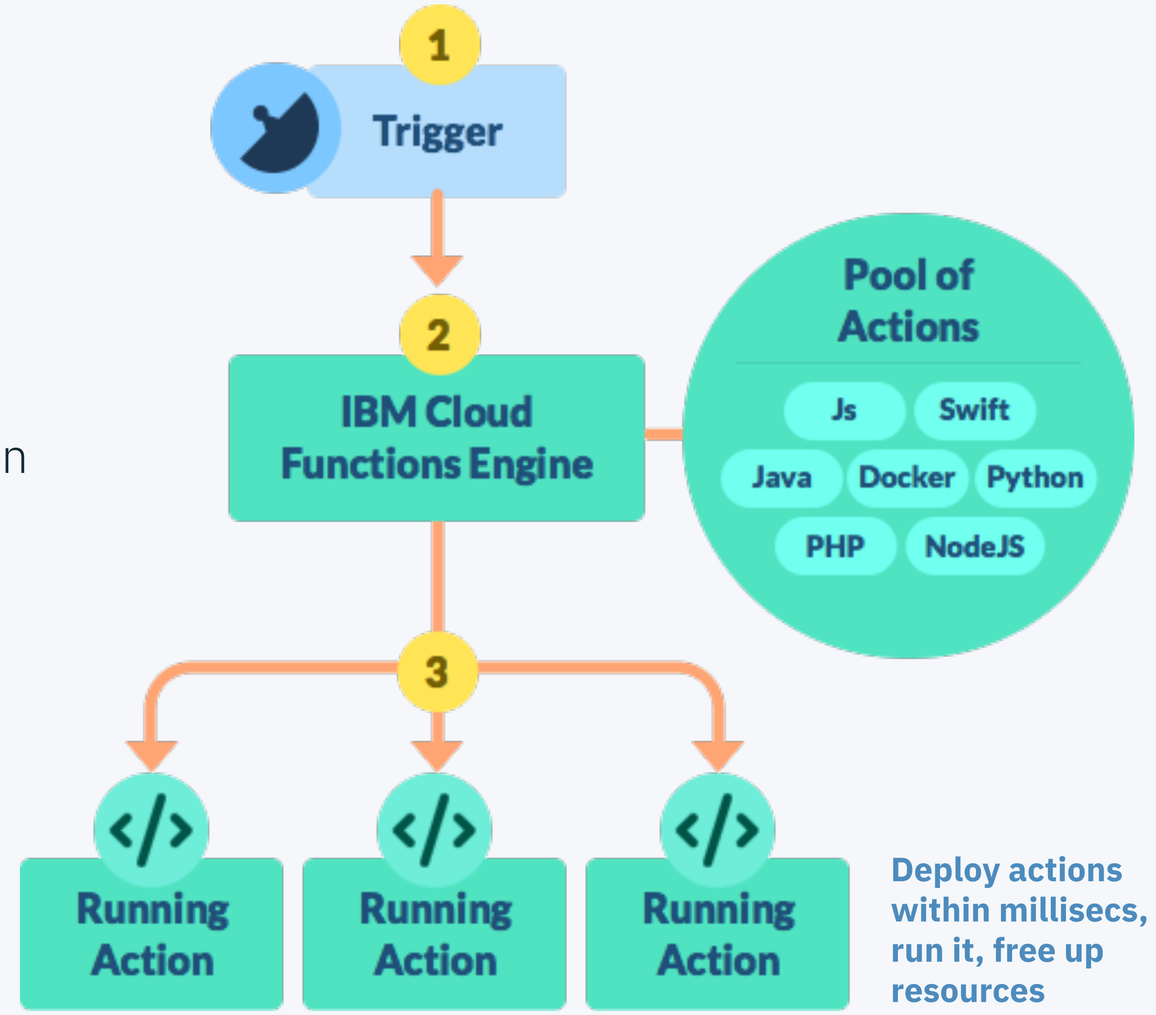
Serverless model

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

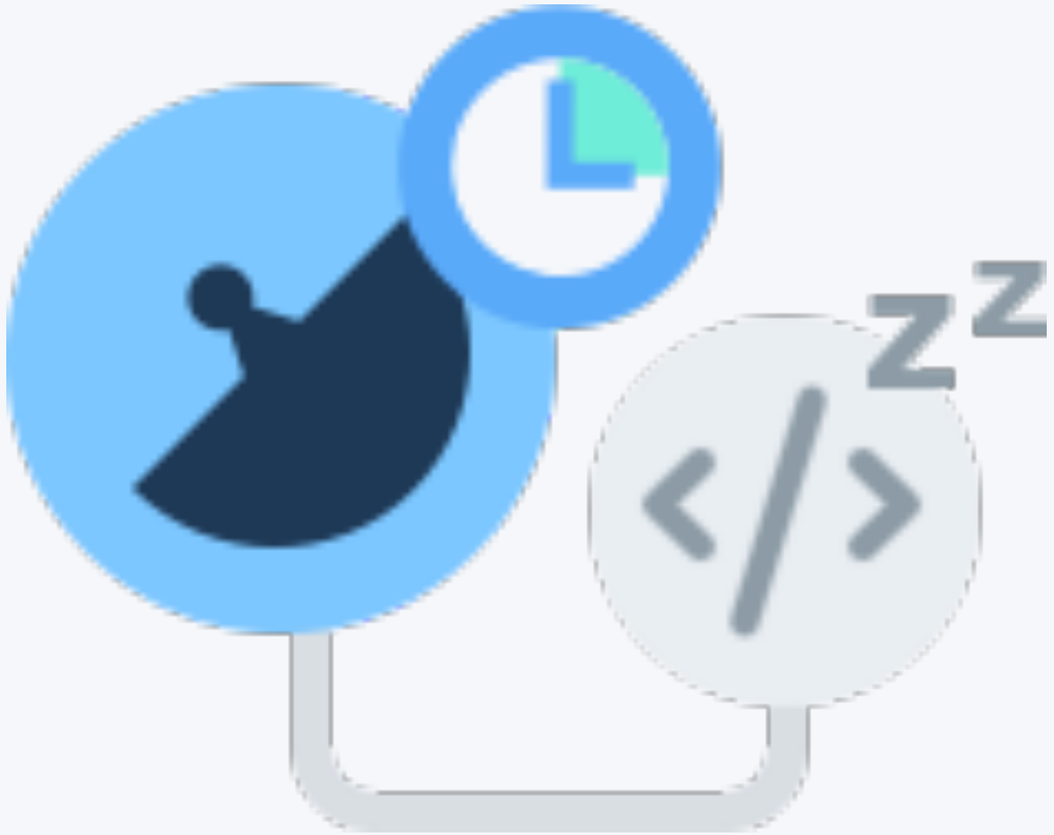
Charges only for what is used
- Only worry about code
  higher dev velocity, lower operational costs



Deploy actions within millisecs, run it, free up resources

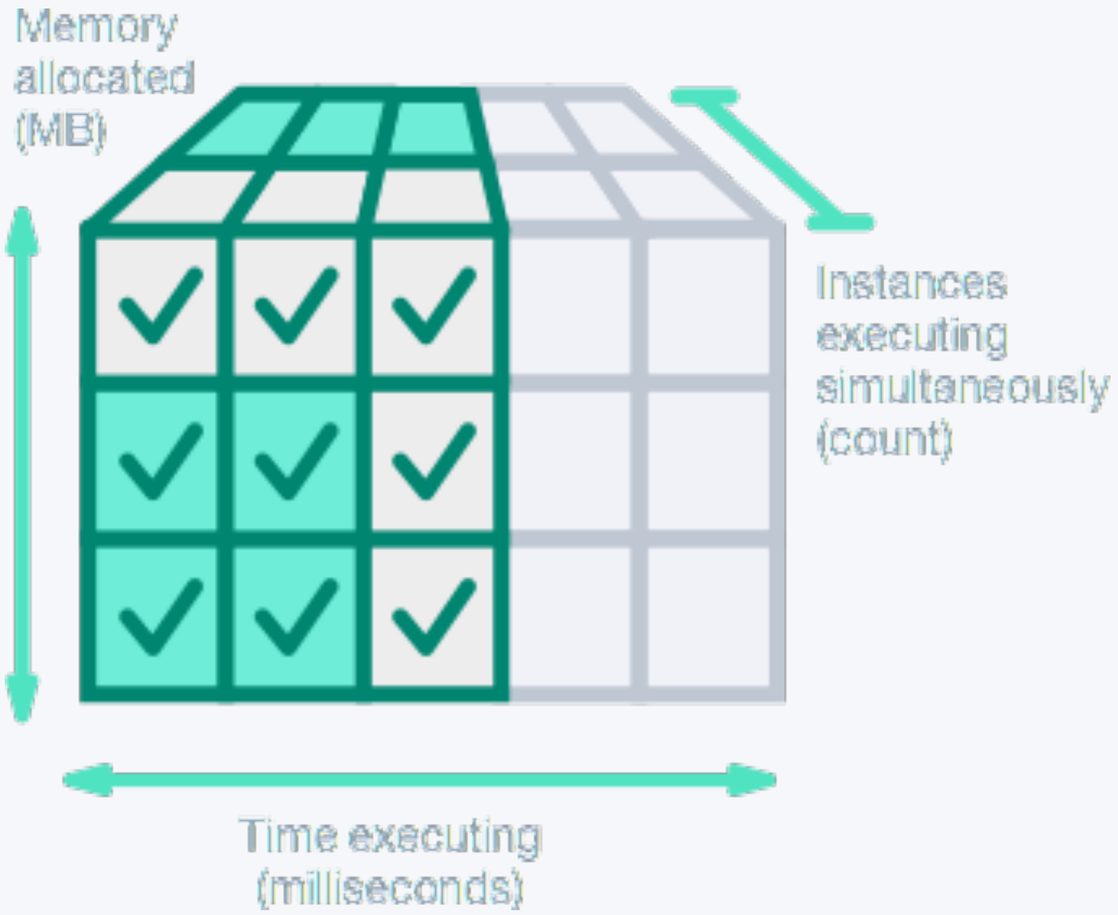
Runs code **only** on-demand on a per-request basis

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



Pricing model

Can vary depending on vendor..



Any language or binary is supported

Natively supported languages (performance-optimized)

- JS/NodeJS
- Swift
- Java
- Python
- PHP
- Ruby
- Go
- .NET

Executable binaries & any other application or language via Containers

- Docker
- Rust
- C
- C++
- bash
- ...

Customers


GreenQ
Making Garbage Trucks Smarter


Wakefern
FOOD CORP.®


fileshadow™
cloud file assurance


ESPN


Santander


DOW®


KONE


SiteSpirit


WEATHER GODS


articoolo


croosing
Taking Browsing Forward


code
zinger


The
Weather
Company


NeuroApplied


Panorays


BIGVU
POCKET TV STUDIO

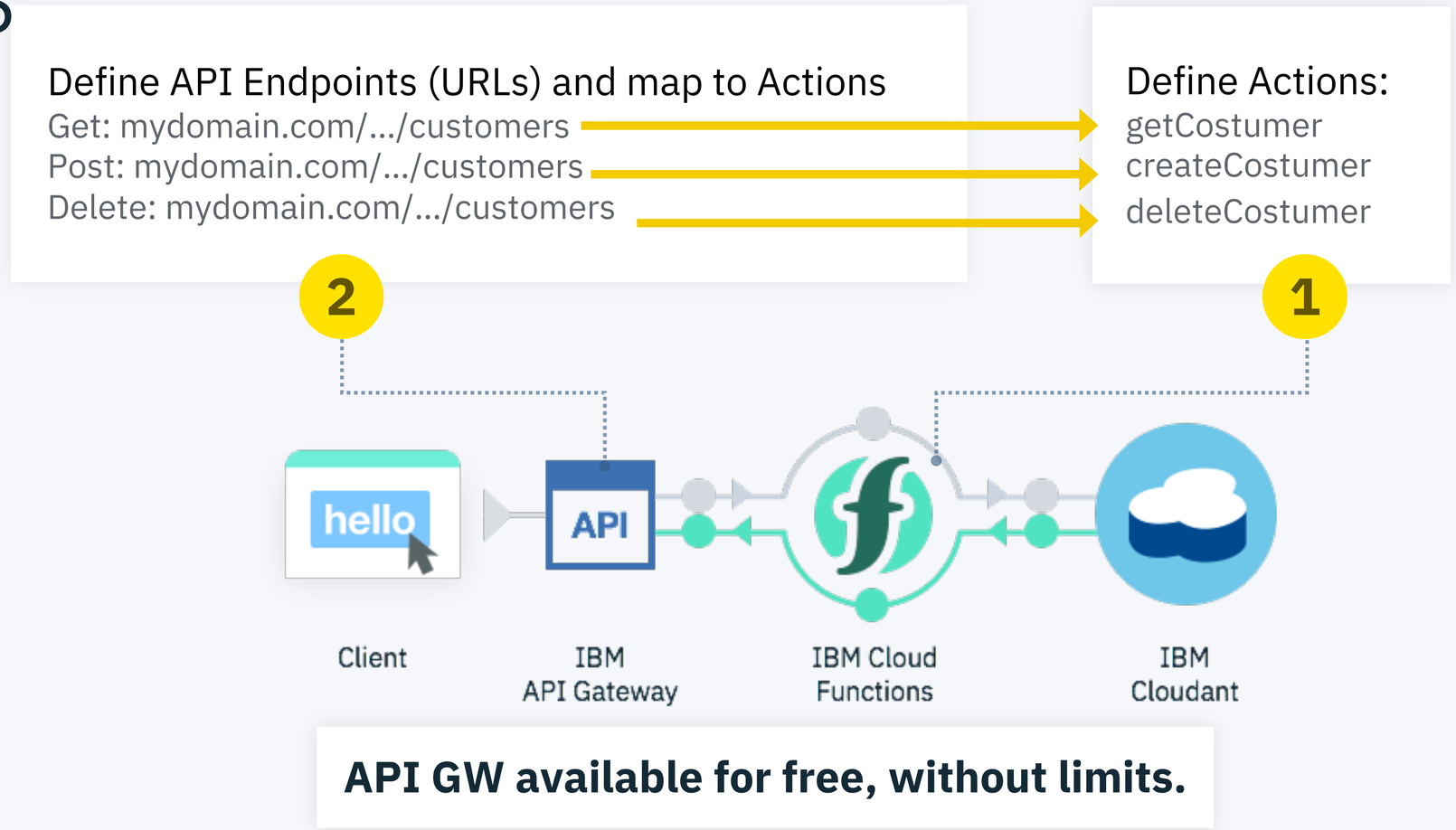

AbiliSense
A S

Common use-cases

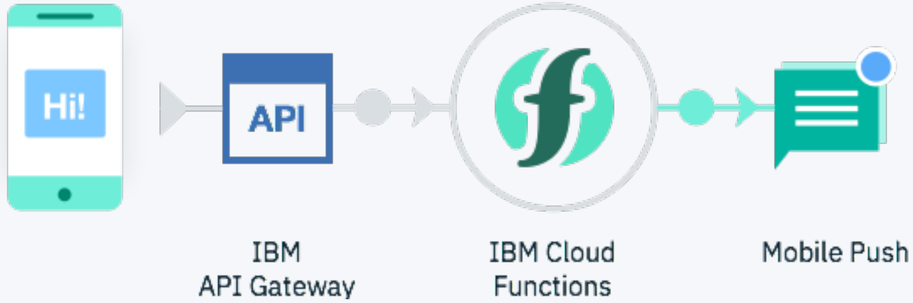
- Microservices / RESTful APIs
- Batch / Periodic Processing
- Operations Research / Combinatorial Optimization
- Machine Learning Models
- Blockchain Front End
- ETL Pipelines
- Chatbots
- Event Stream Processing

Microservices/ API Backends

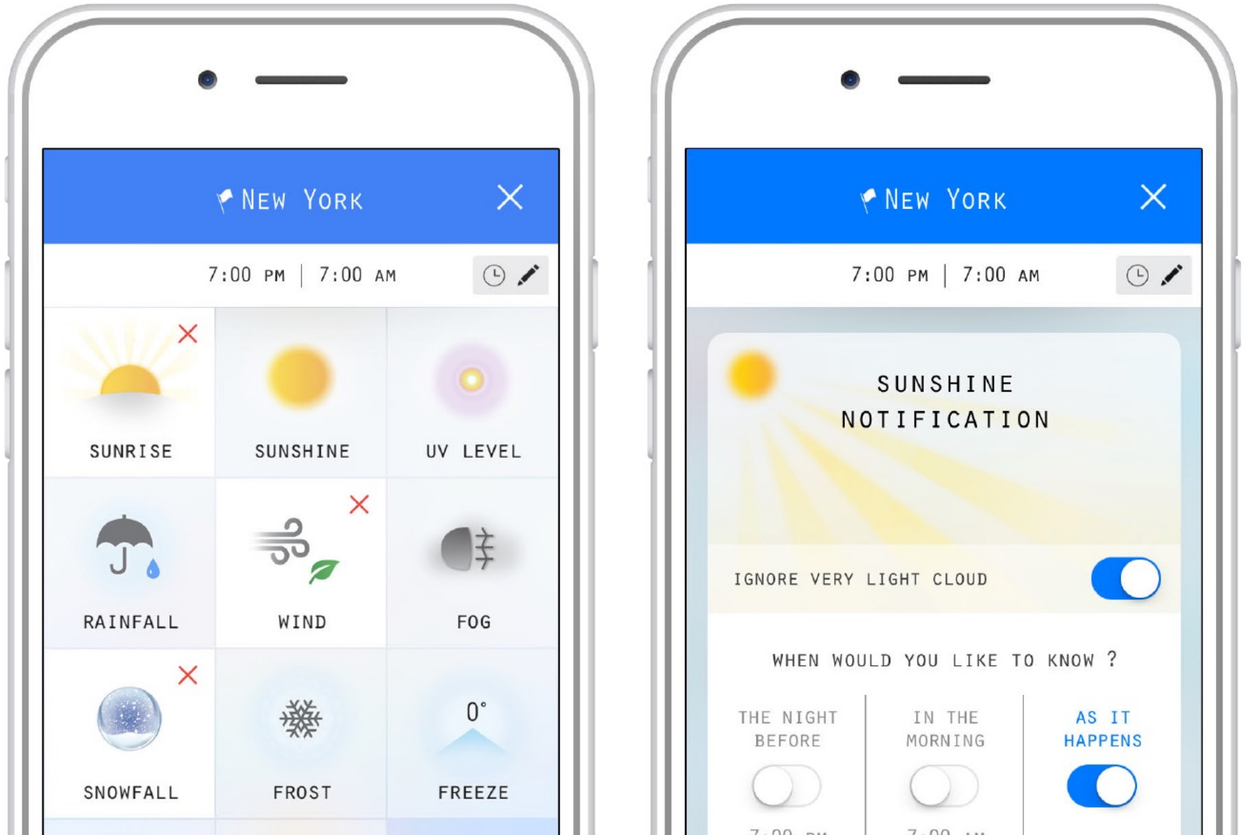
Allows to map API endpoints to IBM Cloud Functions actions



Mobile backend

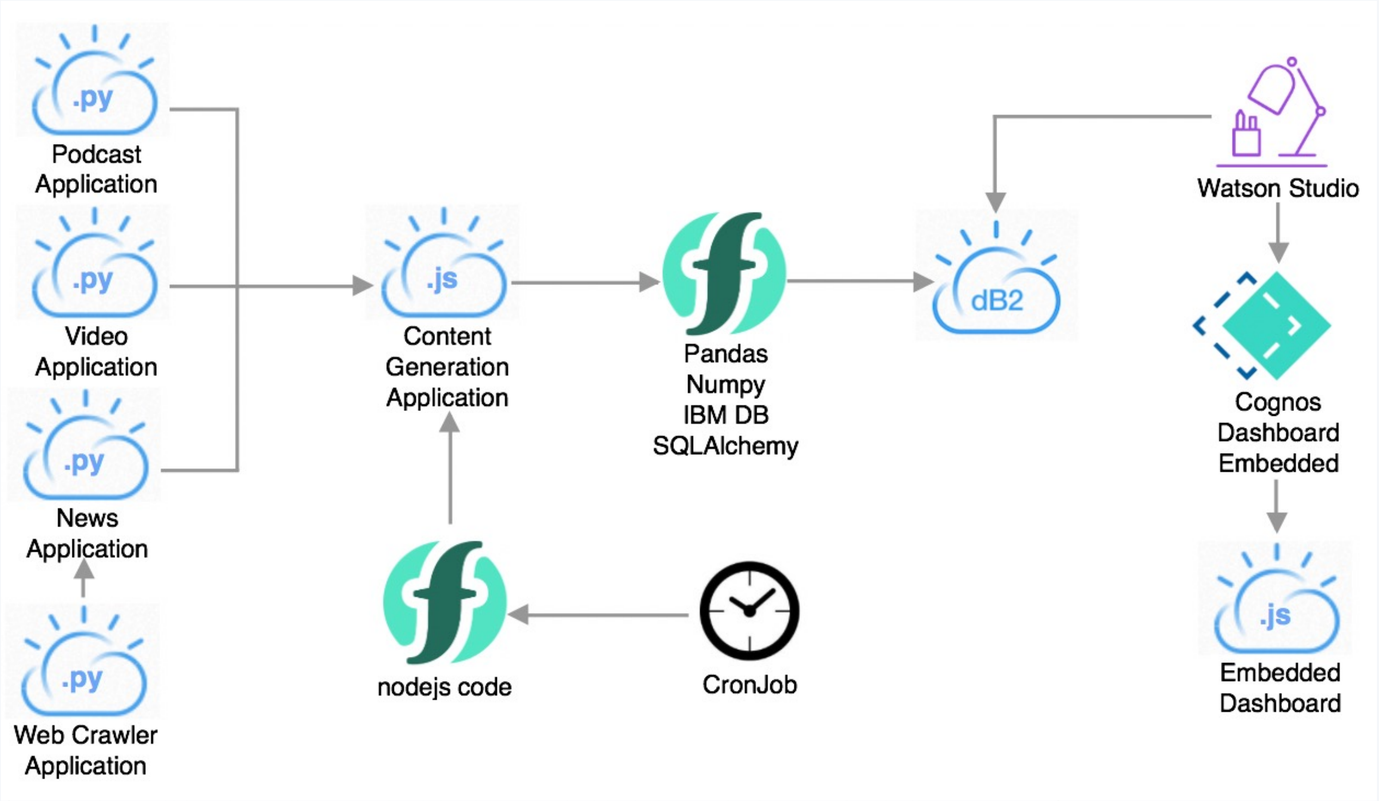


The Weather Gods
https://itunes.apple.com/us/app/weather-gods/id1041512978?mt=8



ESPN Fantasy Football

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



(Source: https://developer.ibm.com/articles/watson-behind-the-code-fantasy-football-2018-part7)

Demo time 🤓

Personas

- Consumer (developer) – most of you
- Serverless provider – some of you

Useful links from demo:

- https://knative.dev/docs/serving/
- https://docs.openshift.com/container-platform/4.7/serverless/knative_serving/serverless-applications.html
- https://knative.dev/docs/serving/samples/hello-world/helloworld-java-spring/ (Java) or https://knative.dev/docs/serving/samples/ (complete list)
- https://developer.ibm.com/components/ibm-linuxone/tutorials/red-hat-openshift-container-platform-linuxone-community-cloud-web-server/ (Red Hat OpenShift Serverless on IBM LinuxONE)
- https://developer.ibm.com/depmodels/serverless/series/ibm-cloud-code-engine-managed-serverless-platform/ (IBM’s managed serverless platform)

When to use serverless..

When there is a need for:

Scalable, stateless, short running/ephemeral code where scale-to-zero is preferred.

Design Patterns

- Event Driven
- CQRS
- Function Chaining
- Function Chaining with Rollback (transaction)
- async HTTP (HTTP 202)
- Fanout (Parallel)
- Aggregation
- Fanout + Fan-in
- Strangler
- Long Running Function
- Long Running Function with Timeout
- Manual interaction with Timeout
- State Machines

This is a mix of patterns/use-cases..

Choosing the right **backing** platform..

Serverless needs to run on...a server..😳

Software stack:

- *supports easy deployment*
- *easy management*
- *integration with devOps (or noOps!)*
- *provides good problem determination capabilities*

Infrastructure stack:

- *multi-dimensional scalability*
- *highest performance (throughput/latency)*
- *highest function density*
- *lowest TCO*

Red Hat OpenShift Serverless on IBM LinuxONE exceeds on *all* of these

Thank You!

Elton de Souza

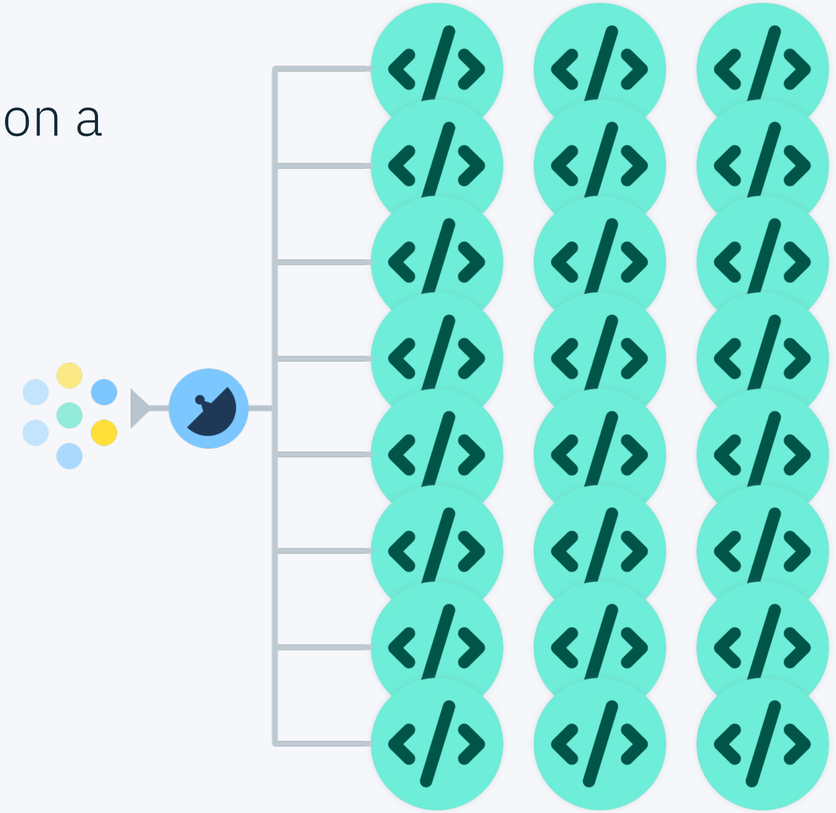
IBM

eltondesouza

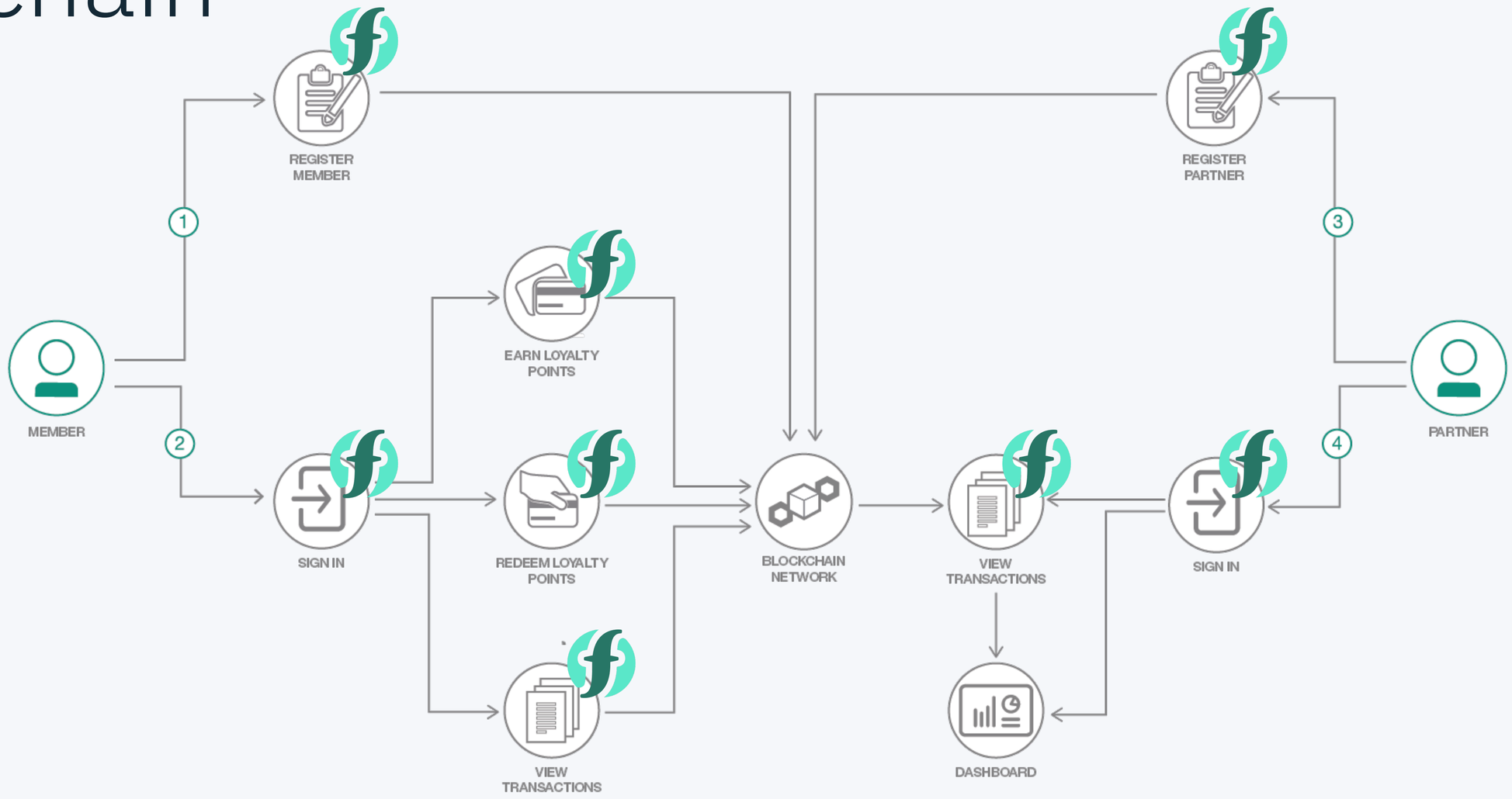
Operations Research / Combinatorial Optimization

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

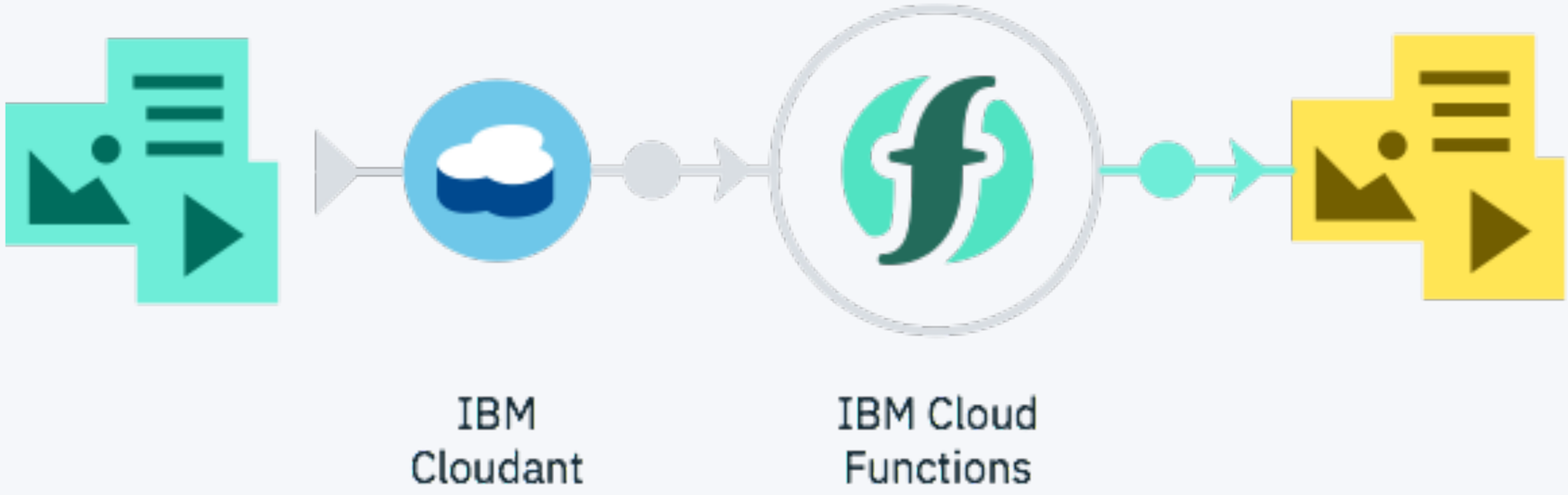
- Map(-Reduce) operations
- Monte-Carlo Simulations
- Genetic Algorithms
- Hyperparameter tuning
- Web scraping
- Genome processing



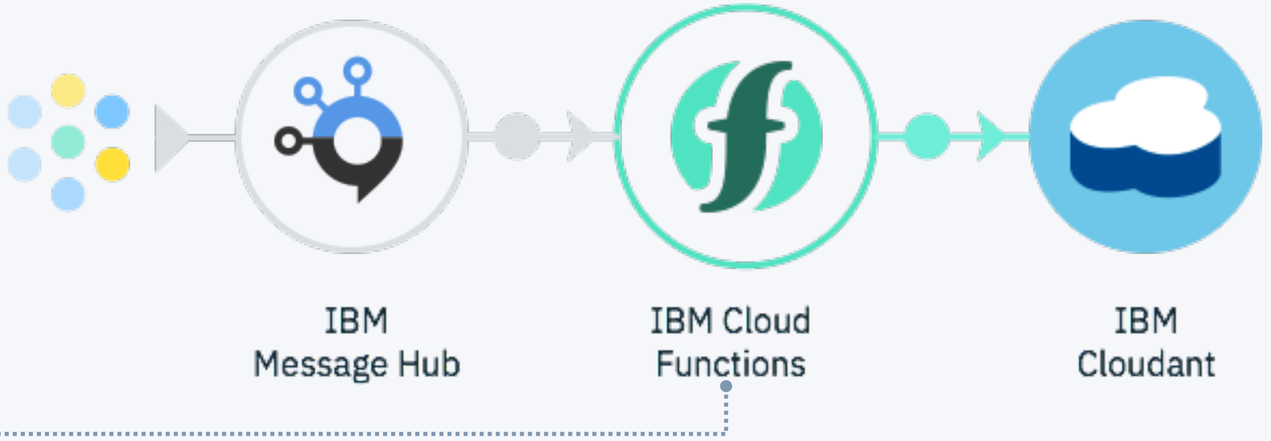
Blockchain APIs



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)

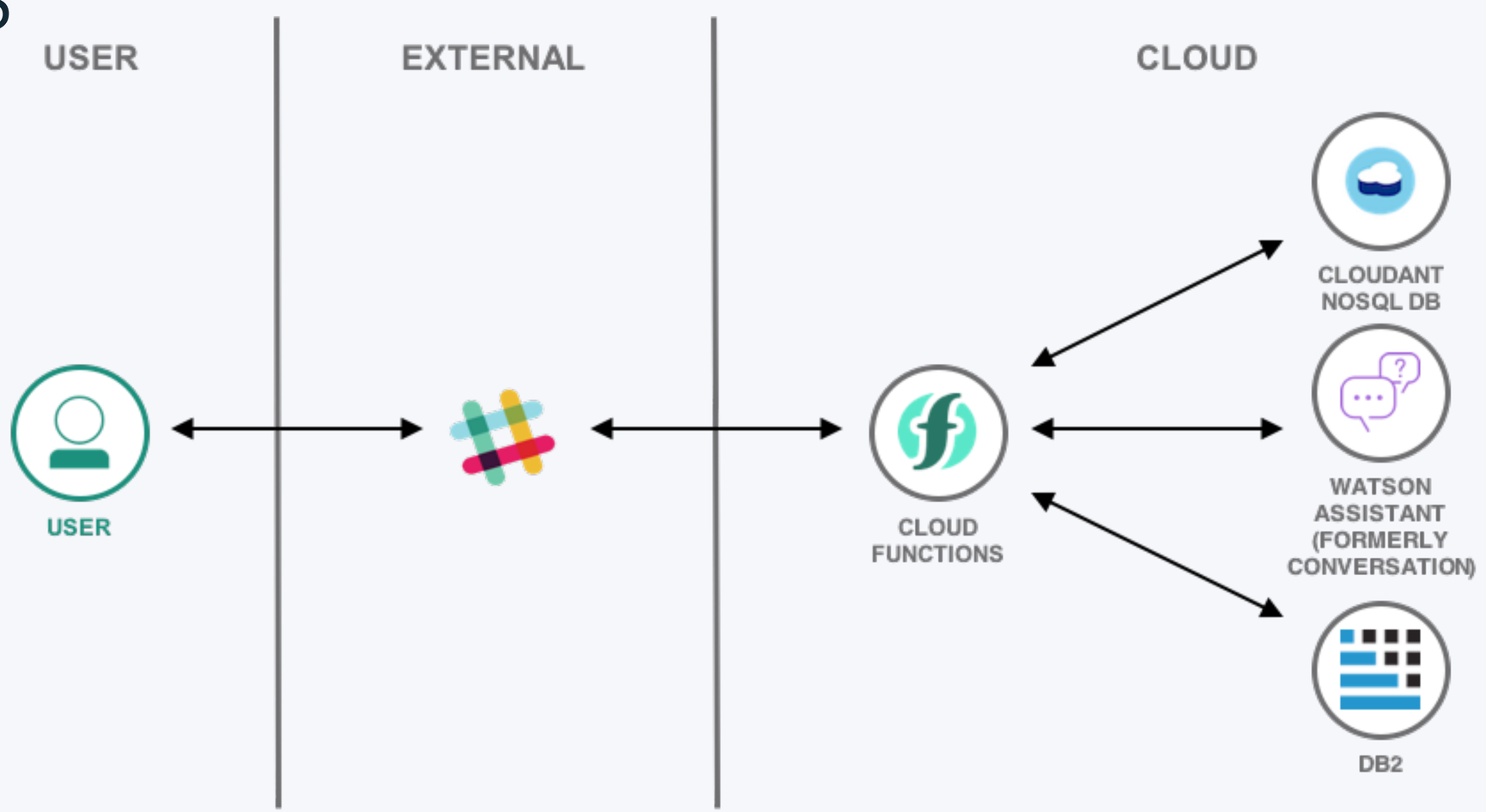
IoT sensor data

Log data

Financial (market) data

...

Conversational applications



Batch / Scheduled tasks