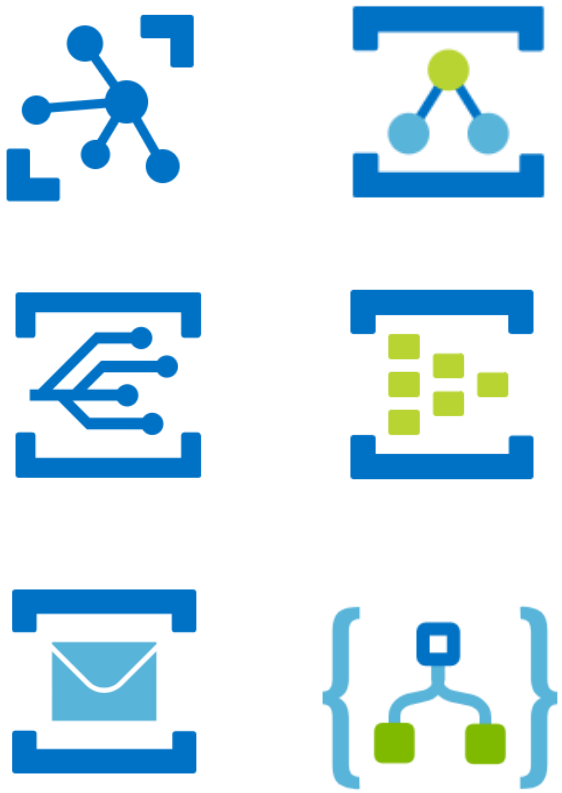
Microsoft

# Eventing, Serverless, and the Extensible Enterprise

Clemens Vasters
Product Architect – Azure Messaging, Microsoft
@clemensv

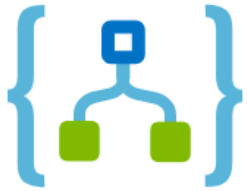
OASIS AMQP Technical Committee Chair
CNCF CloudEvents Spec Owner
OPC UA PubSub Co-Architect



Microsoft Azure
Messaging

**Notification Hubs**

Mobile push notifications



**Logic Apps**

Workflow and LOB Integration

**Service Bus & Azure Queues**

Cloud messaging

**Event Hubs**

Telemetry stream ingestion

Kafka/AMQP

**Event Grid**

Event distribution

**IoT Hub**

IoT messaging and manage-ment

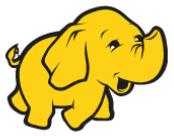
**Relay**

Discovery, Firewall/NAT Traversal

**Storage Blobs**

**Data Lake**

**Stream Analytics**



**Storm Spark**

**Azure Functions**

Microsoft Azure

# Event Driven Apps

# Event-Driven Photos

News Website Editorial Office (edit/publish)

New! Import?

Lightroom

Photoshop

Newsroom App

edit

republish

Check & add tags

update

Mobile (out shooting)

**Cloud**

photoReindexed

photoIngested

Uploads RAW and/or JPG

Aircraft in Trouble!!!

publishDone

sizedImageCreated

import

Index Database

aircraft, indoors, blue, white, UK, Cosford, Canon, 6D

update

blobCreated

aircraft, indoors, blue, white, UK, Cosford, Canon, 6D

Update published assets

write

blobCreated

Normalize format, auto-classify, tag, move

read/delete

blobCreated
blobDeleted

Blob Container "sizedImages"

Create sized images

read

Blob Container "inventory"

Blob Container "uploads"

# Sensor-Event-Driven Building Management: Two Perspectives

- Climate
- Energy
- Lighting
- Water
- Gas
- Space Utilization
- Access Control
- Movement Tracking
- Elevator/Escalator
- Refrigeration
- Networking
- Fire/Smoke/Gas
- Biohazards
- Flooding/Wind
- Earthquake

...

## Building Management

## Device Management

- Power
- Battery
- Temperature
- Vibration
- Component Health
- Firmware
- TPM
- Hypervisor
- OS
- Container
- App
- Config
- Credentials
- Access Control
- Traces
- Logs
- Data

...

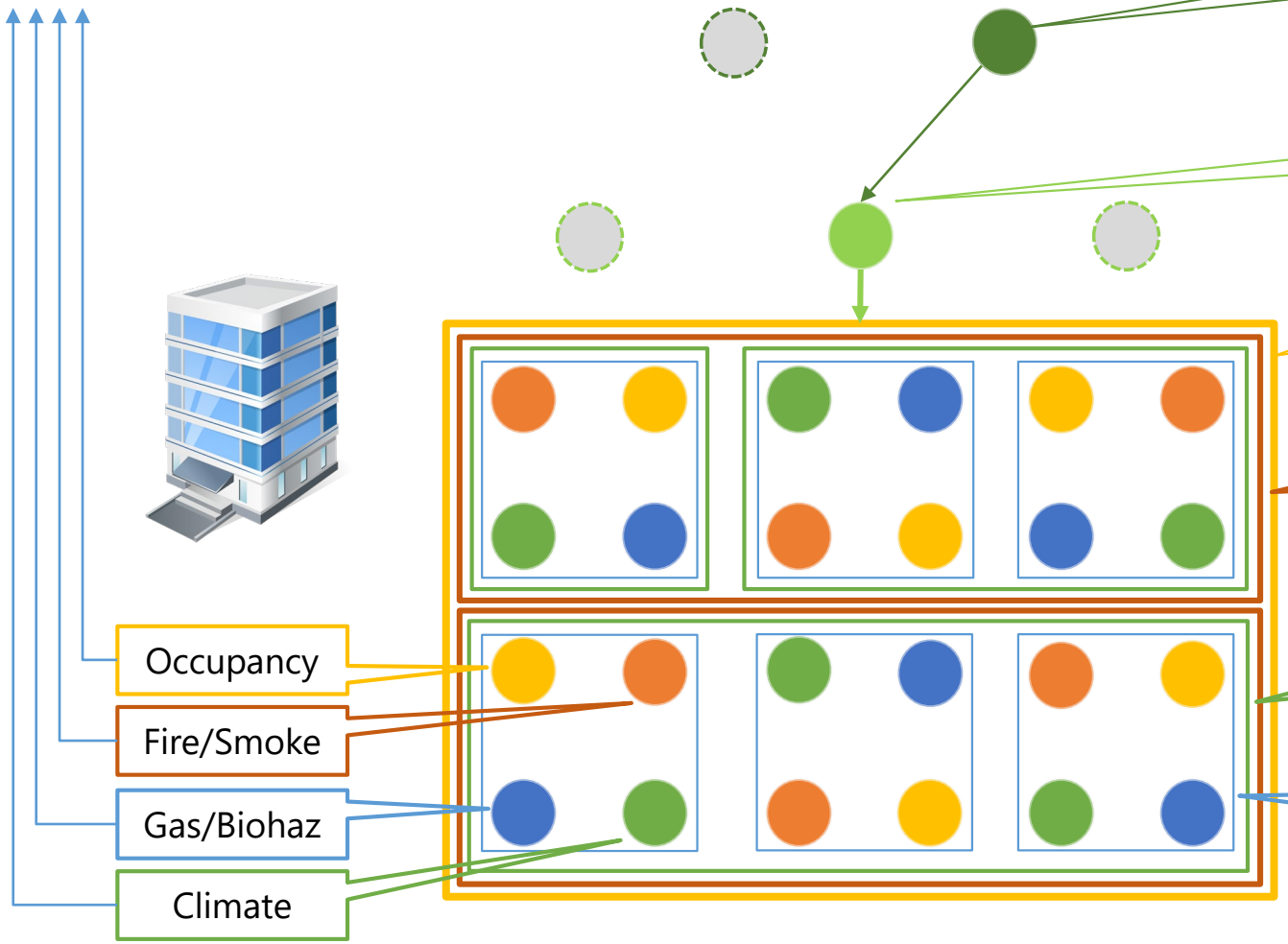
/org/contoso/site/berlin/building/41/floor/3/unit/j/room/4/sensors/occupancy

/org/contoso/site/berlin/building/41/floor/3/unit/j/room/4/sensors/fire

/org/contoso/site/berlin/building/41/floor/3/unit/j/room/4/sensors/gasbio

/org/contoso/site/berlin/building/41/floor/3/unit/j/room/4/sensors/climate

# Building Management




Org

/org/contoso/

Site

/org/contoso/site/berlin/

Building

/org/contoso/site/berlin/building/41

Floor

/org/contoso/site/berlin/building/41/floor/3

Unit

/org/contoso/site/berlin/building/41/floor/3/unit/j

Room

/org/contoso/site/berlin/building/41/floor/3/unit/j/room/4

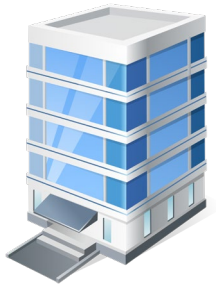
# Building Management

/org/contoso/site/berlin/building/41/floor/3/unit/j/room/4/sensors/occupancy

/org/contoso/site/berlin/building/41/floor/3/unit/j/room/4/sensors/fire

/org/contoso/site/berlin/building/41/floor/3/unit/j/room/4/sensors/gasbio

/org/contoso/site/berlin/building/41/floor/3/unit/j/room/4/sensors/climate



Occupancy

Fire/Smoke

Gas/Biohaz

Climate

*Some Analytics Questions:*

- Are there people in room 41 3J/4?
- What room is unoccupied in building 41?
- Is there a fire alarm at the site?
- How is the air quality on the lab floor?
- What's the temp in tenant unit 41 3J?

*Some Reactive Actions:*

- Signal evacuation and alert the Fire Department if any fire or gas/biohaz sensor on site goes into an alert state.
- Adjust floor HVAC when average temp on any building floor deviates by +/- 2C from 20C.
- Alert Security when unexpected occupancy is detected in Unit 41 3J.

Org

/org/contoso/

Site

/org/contoso/site/berlin/

Building

/org/contoso/site/berlin/building/41

Floor

/org/contoso/site/berlin/building/41/floor/3

Unit

/org/contoso/site/berlin/building/41/floor/3/unit/j

Room

/org/contoso/site/berlin/building/41/floor/3/unit/j/room/4

/org/contoso/site/berlin/build

/org/contoso/site/berlin/build

/org/contoso/site/berlin/build

/org/contoso/site/berlin/build

nagement

Occupancy

Fire/Smoke

Gas/Biohaz

Climate

## "Are there people in room 41 3J/4?"

- SUBSCRIBE /org/contoso/site/berlin/building/41/floor/3/unit/j/room/4/sensors/occupancy
- EVENT
  - **Topic**: /org/contoso/site/berlin/building/41/floor/3/unit/j/room/4/sensors/occupancy
  - **Data** { sensor: xyz, bldg: 41, room: "3J/4", persons : 0 }

## What room is unoccupied in building 41?

- SUBSCRIBE /org/contoso/site/berlin/building/41
- EVENT
  - **Topic**: /org/contoso/site/berlin/building/41
  - **Filter on subject:** /floor/*/unit/*/room/*/sensors/occupancy == 0
  - **Data** { sensor: xyz, bldg: 41, room: "3J/4", persons : 0 }

## Is there a fire alarm at the site?

- SUBSCRIBE /org/contoso/site/berlin
- EVENT
  - **Topic**: /org/contoso/site/berlin
  - **Filter on subject:** /building/*/floor/*/unit/*/room/*/sensors/fire (type== 'alarm')

## How is the air quality on the lab floor?

- SUBSCRIBE /org/contoso/site/berlin/building/41/floor/1
- EVENT
  - **Topic**: /org/contoso/site/berlin/building/41/floor/1
  - **Filter on subject:** /unit/*/room/*/sensors/climate

## What's the temp in tenant unit 41 3J?

- SUBSCRIBE /org/contoso/site/berlin/building/41/floor/3/unit/j
- EVENT
  - **Topic:** /org/contoso/site/berlin/building/41/floor/3/unit/j
  - **Filter on**: /room/*/sensors/climate

unit/j

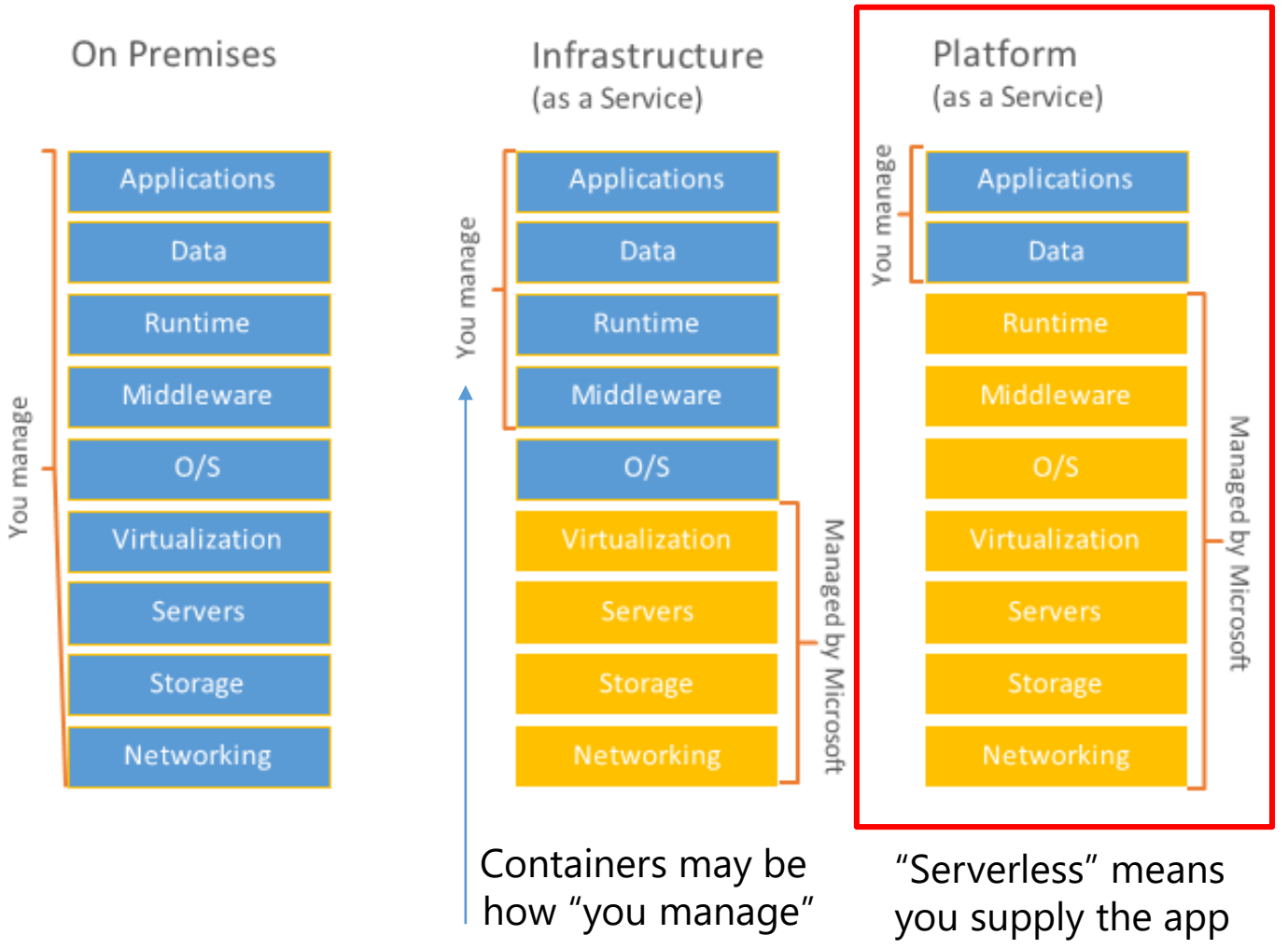
unit/j/room/4

# Serverless and Microservices

(Spoiler: Of course there are servers)
(Spoiler: This isn't new)

# "Serverless" = Platform as a Service (PaaS)

- "Serverless" is PaaS
  - The promise of "Platform as a Service" is to liberate you from managing low-level infrastructure and focus on apps and data
- Virtualization is IaaS
  - Containers and container orchestrators are a very flexible way to share, configure, and reuse portions of virtualized infrastructure.
  - Container platforms may form a complete virtualization layer, including software-defined networking.

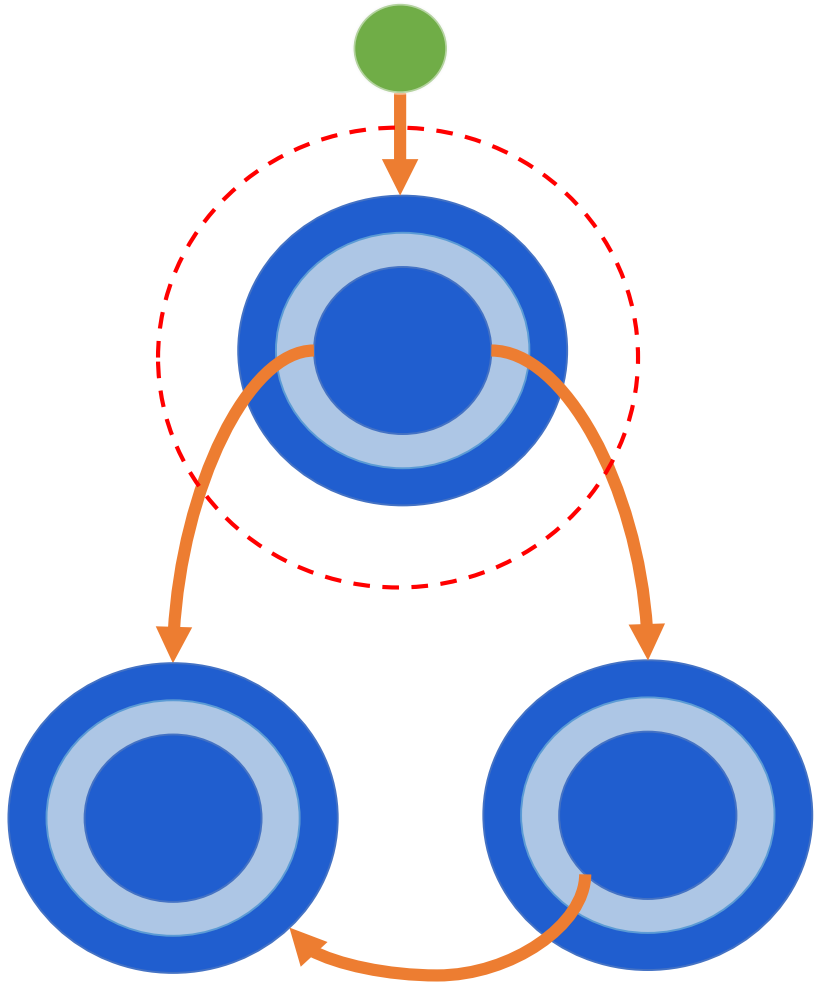


# Degrees of Serverlessness

- Managed Cluster
  - You supply applications that are deployed on a cluster that allows for placement, replication, ownership consensus and management of stateful resources
- Managed Middleware
  - You supply applications that are deployed on sets of independent, “stateless” middleware servers, like web servers or pure compute hosts
  - Applications may be always-on or start on demand; typically maintain shared cached state and resources
- Managed Functions
  - You supply (small groups of) function implementations that are run for you when triggered by a configured condition.

# A “Service” is software that …

- … is responsible for holding, processing, and/or distributing particular kinds of information within the scope of a system
- … can be built, deployed, and run independently, meeting defined operational objectives
- … communicates with consumers and other services, presenting information using conventions and/or contract assurances
- … protects itself against unwanted access, and its information against loss
- … handles failure conditions such that failures cannot lead to information corruption



# Services: Autonomous Entities

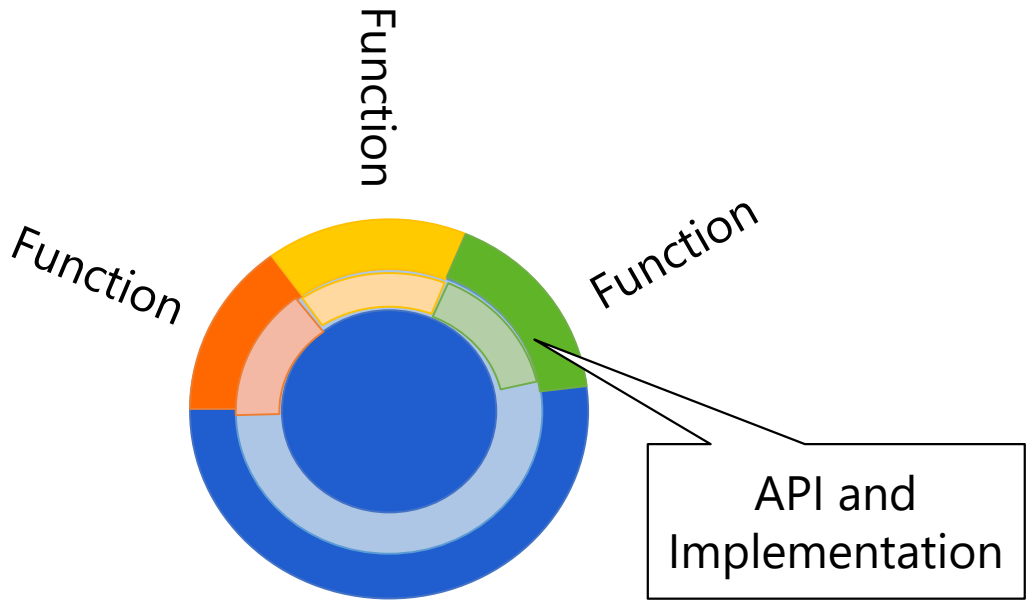
- Defining property of services is that they're *Autonomous*
  - A service owns all of the state it immediately depends on and manages
  - A service owns its communication contract
  - A service can be changed, redeployed, and/or completely replaced
  - A service has a well-known set of communication paths
- Services shall have no shared state with others
  - Don't depend on or assume any common data store
  - Don't depend on any shared in-memory state
- No sideline communications between services
  - No opaque side-effects
  - All communication is explicit
- **Autonomy is about agility and cross-org collaboration**

The modern notion of "Service" is not about code artifact counts or sizes or technology choices.

It's about ownership.

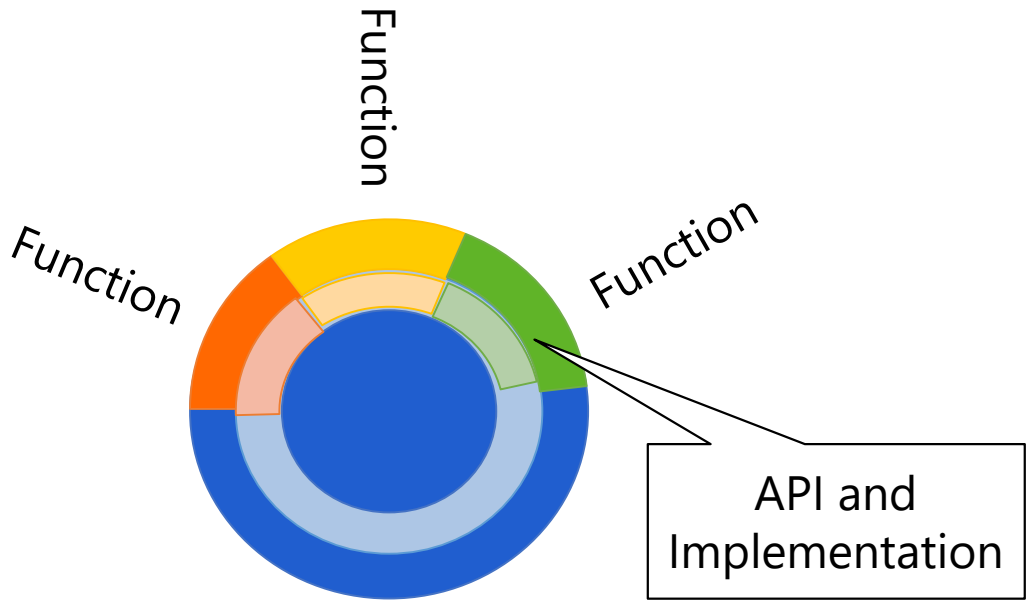
# Microservices and Functions-as-a-Service?

- A service can be made up of a **fleet of** independently deployed **functions** that jointly operate on a shared set of resources
- The service interface is made up from the union of the function interfaces
- The function interfaces may be a mix of RPC-style call interfaces and event driven ones



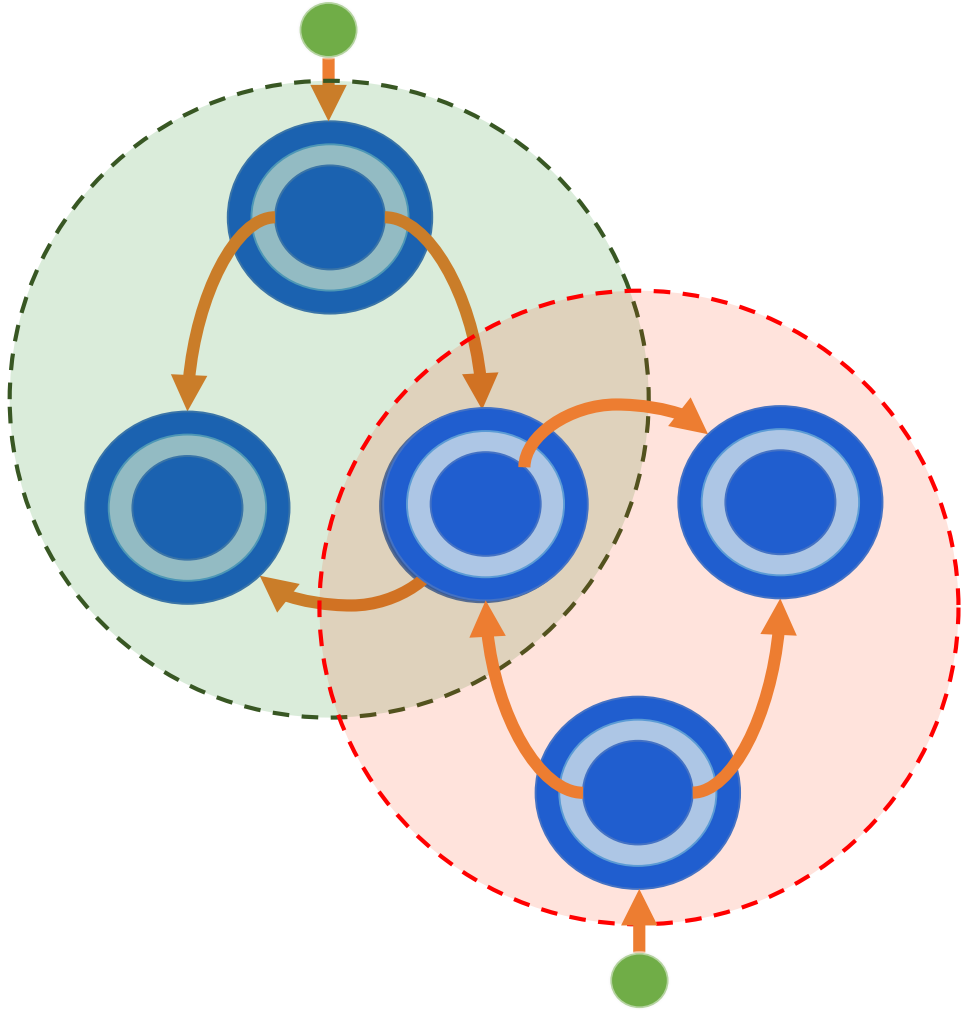
# Upsides of Functions?

- Independently deployable and versionable
- Independently scalable
- Can have independent communication paths and invocation triggers
  - Webhook
  - Queue
  - DB Transaction
  - Etc.



# System

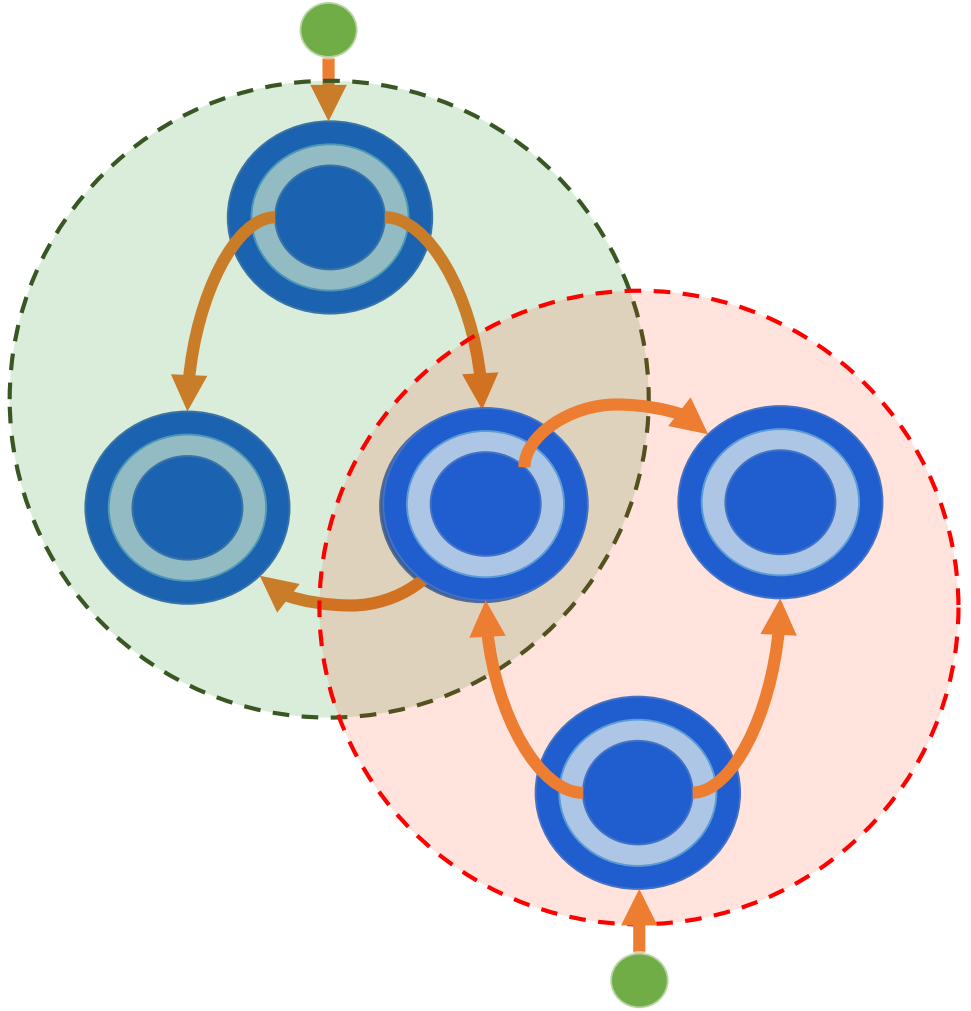
- A system is a federation of services and systems, aiming to provide a composite solution for a well-defined scope.
- The solution scope may be motivated by business, technology, policy, law, culture, or other criteria
- A system may appear and act as a service towards other parties.
- Systems may share services
- Consumers may interact with multiple systems



# Eventing and Messaging

# Service Communication

- “REST” is great for interactively accumulating and acting on state from multiple sources.
  - Let’s not pretend all clients are like that – there’s a lot more
- HTTP and RPC are great to obtain immediate answers.
  - The longer it takes to generate the answer, the more brittle the model becomes



Command

Report

Transfer

Notification

Query

Measurement

Assignment

Job

Handover

Update

Request

Trace

Command

Transfer

Query

Handover

Job

Assignment

Update

Request

Report

Notification

Measurement

Trace

Intents

Facts

Messaging

Intents

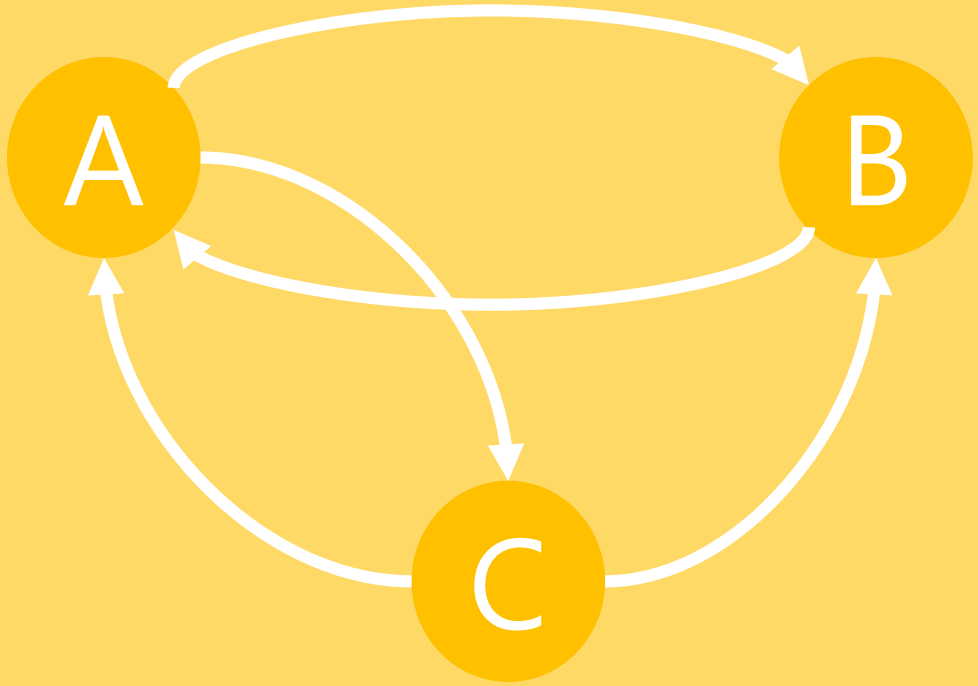
Expectations
Conversations
Contracts
Control Transfer
Value Transfer

Eventing

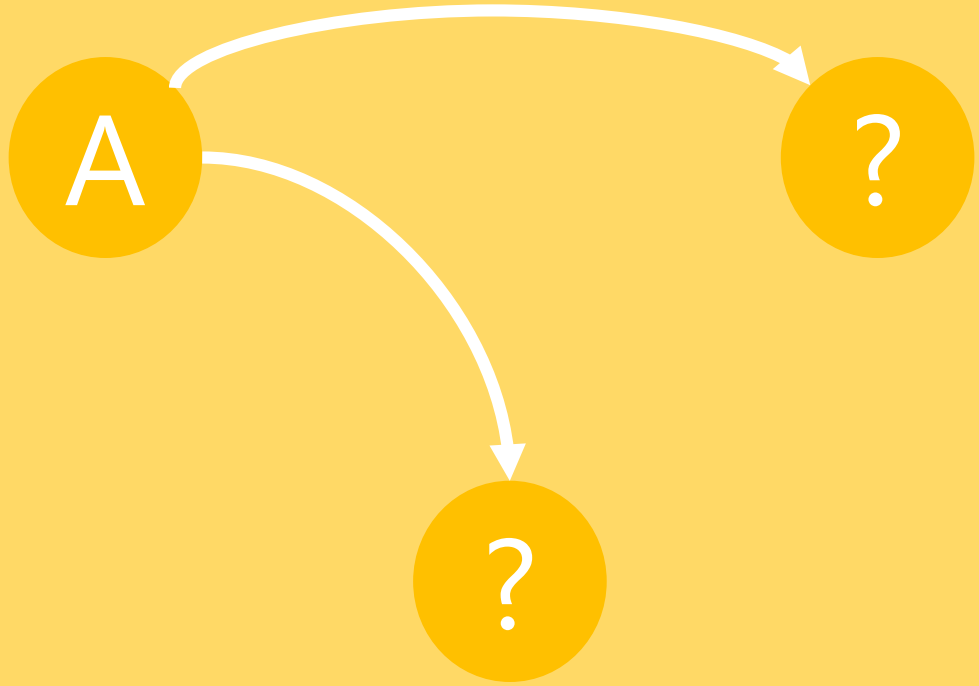
Facts

History
Context
Order
Schema

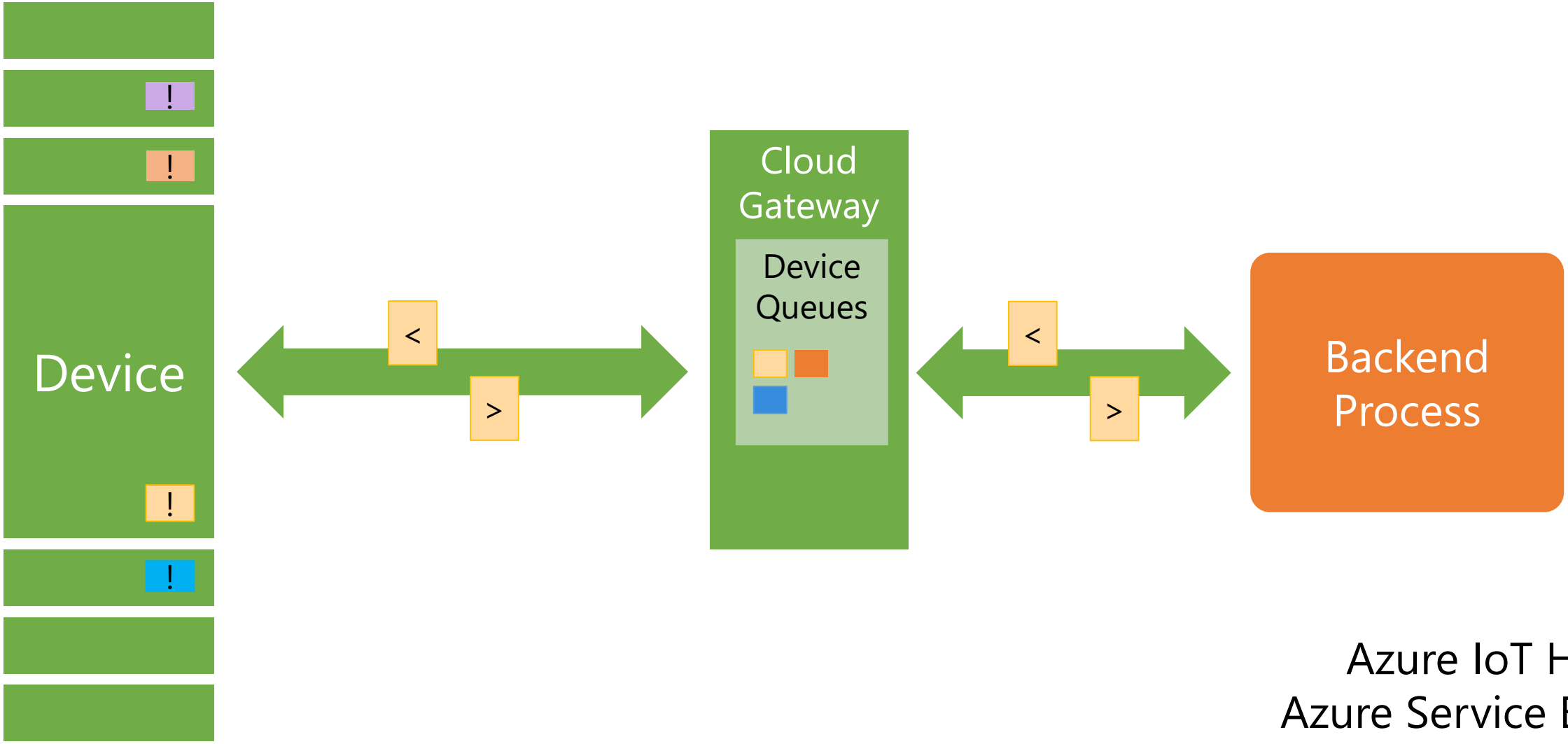
# Messaging




# Eventing

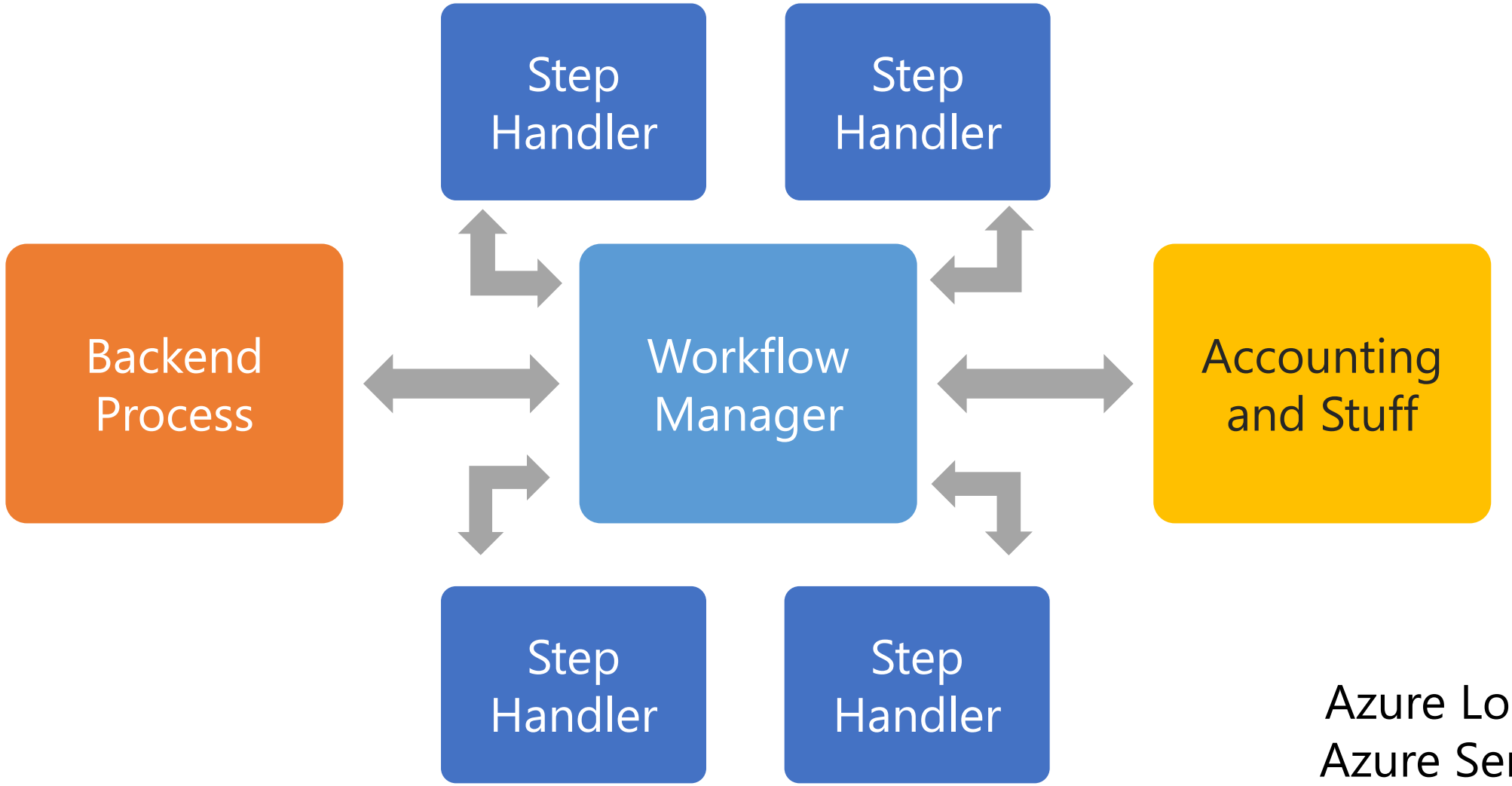


# Messaging – Example: Device Command




Azure IoT Hub
Azure Service Bus

# Messaging – Example: Workflow Execution




Azure Logic Apps
Azure Service Bus

# Events

## Discrete

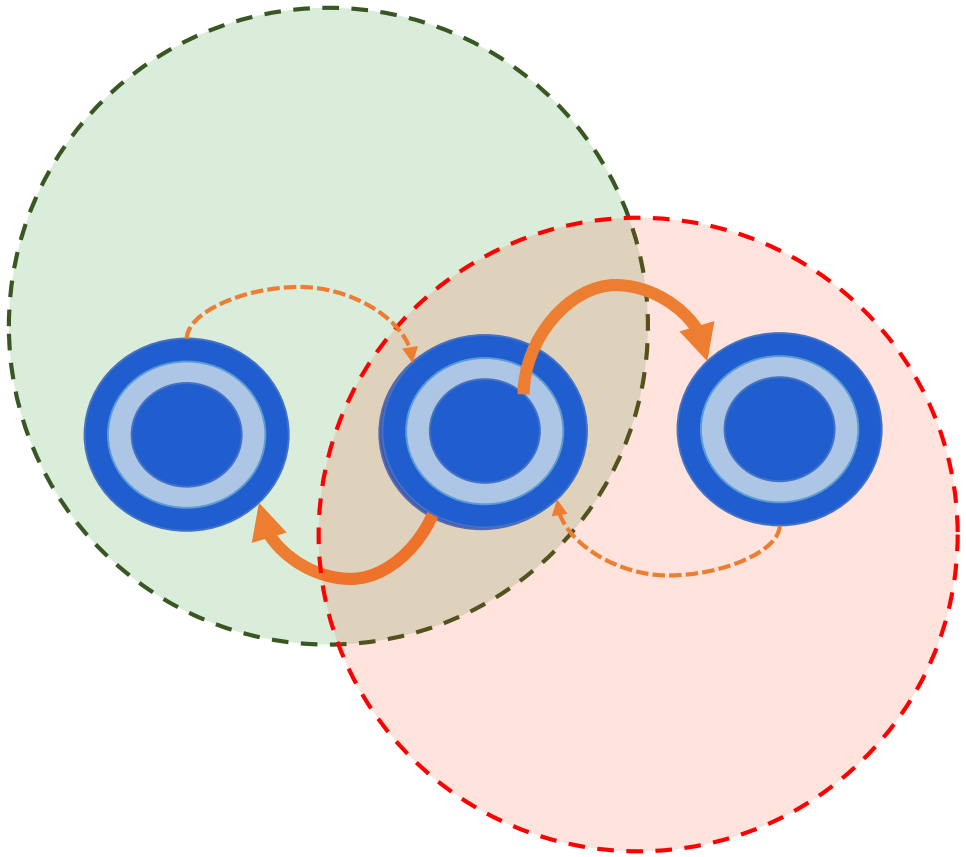
Independent
Report State Change
Actionable

## Series

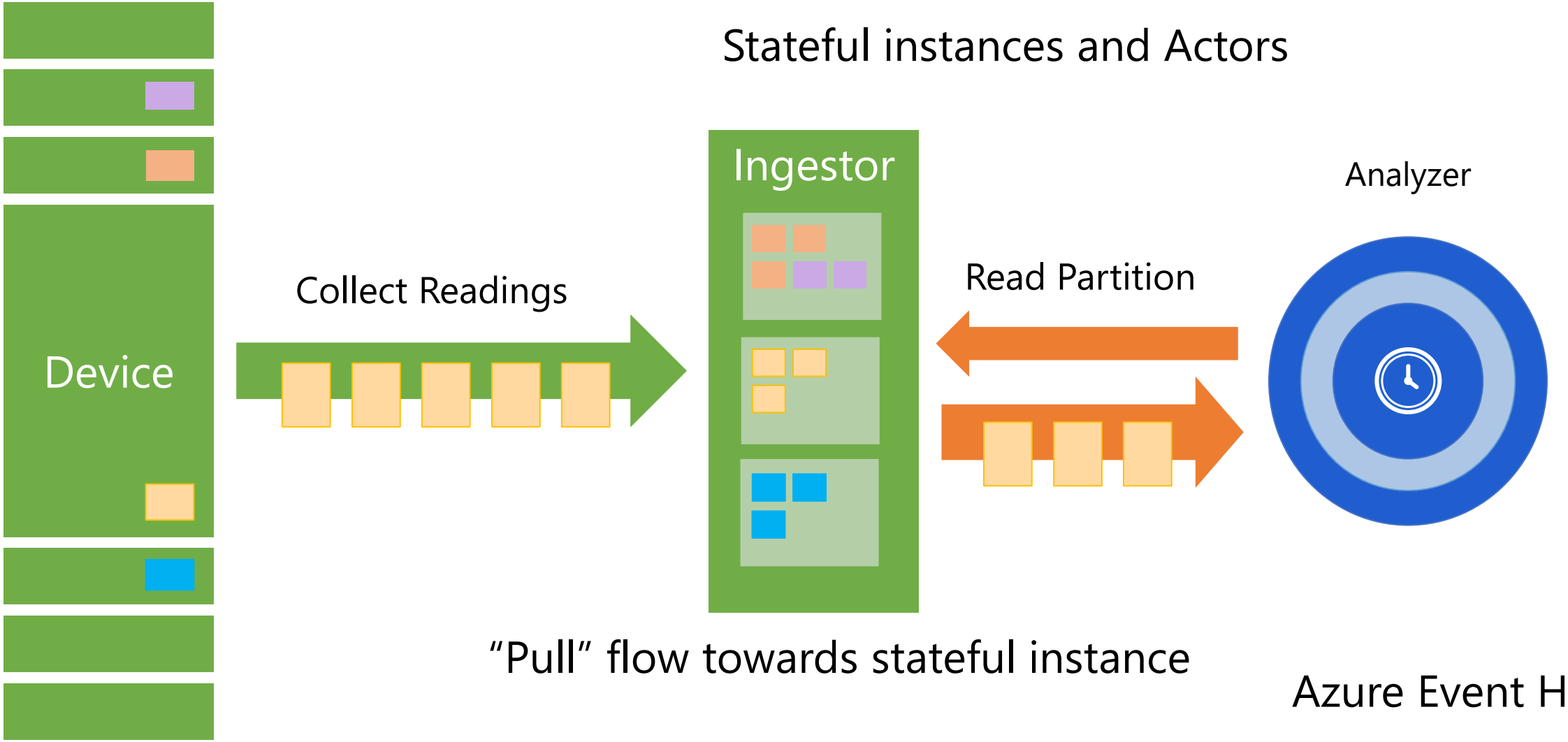
Time Ordered
Context Partitioned
Report Condition
Analyzable

# Discrete Events are an Extensibility Enabler

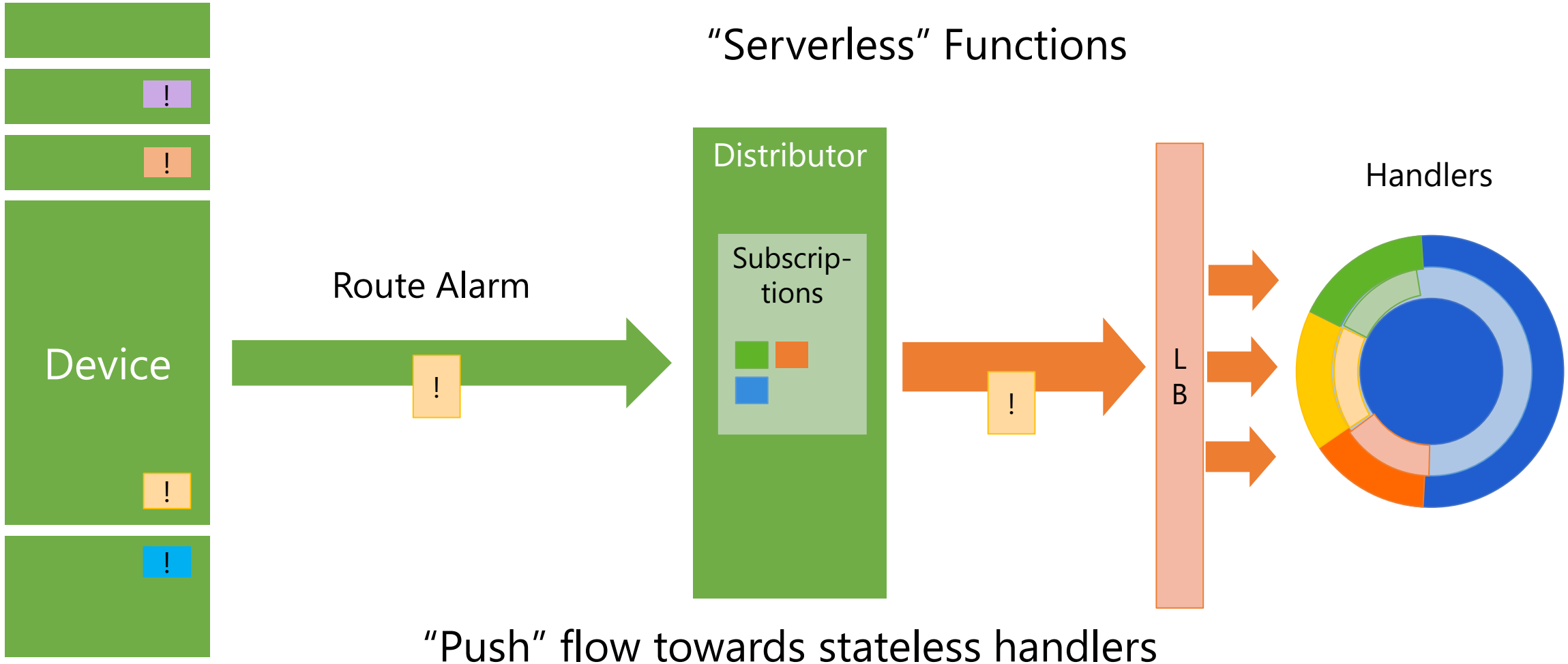
- Report independent, actionable state changes to authorized subscribers
  - “Blob created”
  - “Sales lead created”
  - “Order confirmed”
- Allows simple, noninvasive, reactive extension of the functionality of a service



# Example: Data Series Processing



# Example: Discrete Event Handling –Alarms

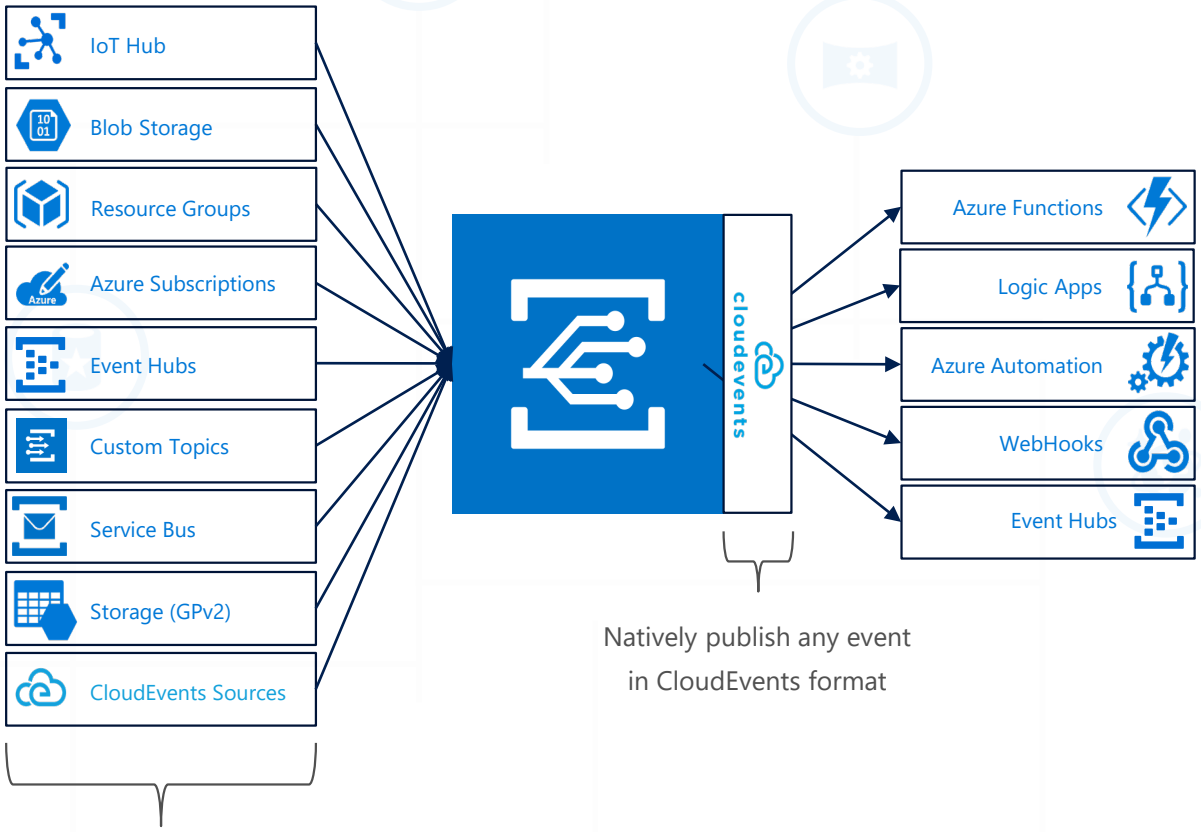



Azure Event Grid

# CNCF CloudEvents?

- Event Protocol Suite developed in CNCF Serverless WG
- Common metadata attributes for events
- Flexibility to innovate on event semantics
- Simple abstract type system mappable to different encodings
- Transport options
- HTTP(S) 1.1 Webhooks, also HTTP/2 (v0.1)
- MQTT 3.1.1 and 5.0 (draft)
- AMQP 1.0 (draft)
- Encoding options
- JSON (v0.1, required for all implementations)
- Extensible for binary encodings: Avro, MessagePack, AMQP, etc.

# First-class support for CloudEvents on Azure Event Grid

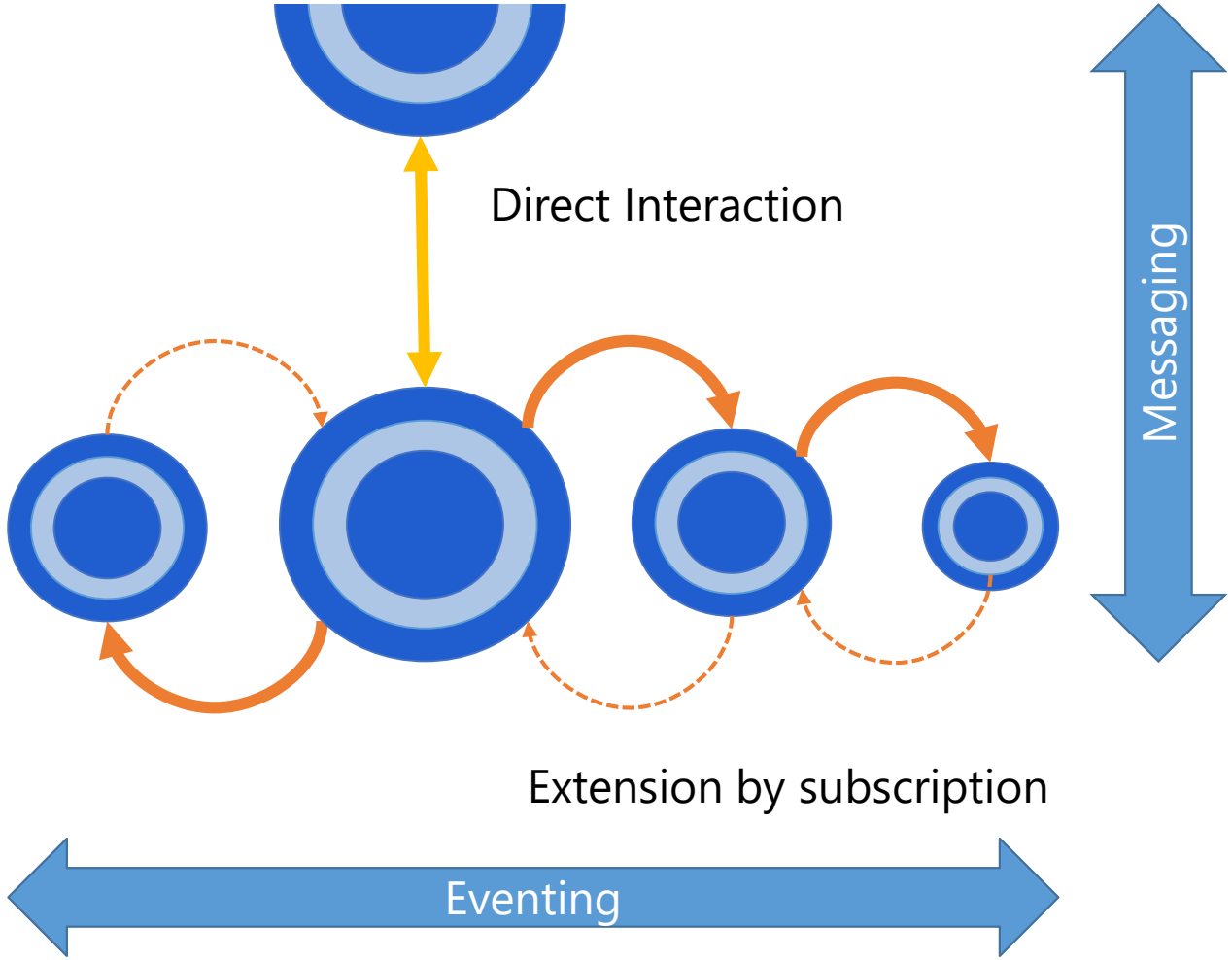



- CloudEvents v0.1 support for **all** Azure platform events, available in production
- CloudEvents v0.1 support for Event Grid custom topics (structured JSON encoding)
- Docs: **https://aka.ms/egcncfdocs**

Azure

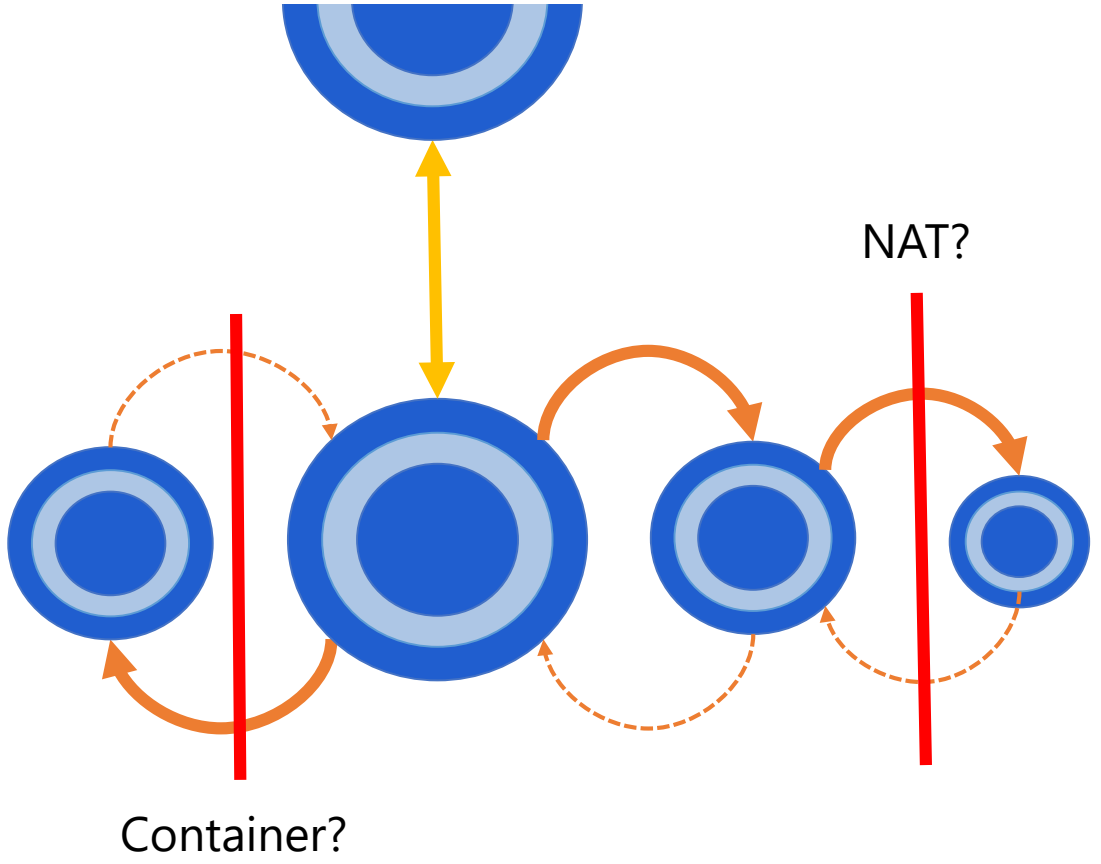
# Event-Driven: Cores + Extensions

- "Core" functions of services allow for direct interaction:
  - Commands, Requests, etc.
- Extensions react to events emitted by services based on core activities.
  - Might turn to the emitting service to ask for details or perform actions.



# Extend to the Edge? Extend into containers?

- Event handlers may need to be implemented by apps that reside out of easy network reach
  - Behind NATs
  - Inside of Containers w/o public endpoints
- A: Route via queues
- B: Use a relay



# Azure Relay HTTP

- Regular Node.js Listener

```
var http = require('http‘);
var port = process.env.PORT || 1337;

http.createServer(function (req, res) {
     res.writeHead(200, {
      'Content-Type': 'text/plain'
     });
     res.end('Hello World\n');
}).listen(port);
```

- Relayed Node.js Listener

```
var http = require('hyco-https‘);

http.createRelayedServer(
     cxnString, function (req, res){
     res.writeHead(200, {
        'Content-Type': 'text/plain‘
      });
      res.end('Hello World\n‘);
}).listen();
```

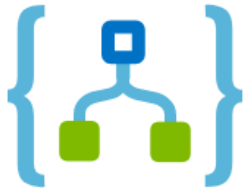
http://aka.ms/vxa

# Summary

- Services: Autonomous software entities grouped around resource ownership and team ownership scopes
- Platform-as-a-Service: Hosting options for services with tailored degrees of control for certain scenarios.
  - Managed clusters: Complex, stateful, high reliability, always-on.
  - Managed servers: Stateless/shared-nothing, on-demand start.
  - Managed functions: Event-driven, short-lived
- Messaging and Eventing: Communication backplane for services, eventing as extensibility enabler.

**Notification Hubs**

Mobile push notifications



**Logic Apps**

Workflow and LOB Integration

**Service Bus & Azure Queues**

Cloud messaging

**Event Hubs**

Telemetry stream ingestion

**Event Grid**

Event distribution

**IoT Hub**

IoT messaging and manage-ment

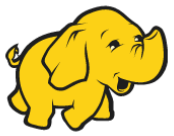
**Relay**

Discovery, Firewall/NAT Traversal

**Storage Blobs**

**Data Lake**

**Stream Analytics**



**Storm Spark**

**Azure Functions**

Microsoft Azure