

**MAI4CAREU**

Master programmes in Artificial  
Intelligence 4 Careers in Europe

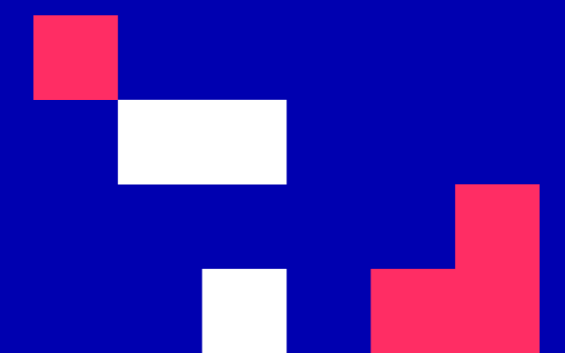


University of Cyprus

# MAI650 Internet of Things

**Vasos Vassiliou**

September - December 2023





## CS6xx Internet of Things (8 ECTS)

**Course purpose and objectives:** The purpose of the course is to provide an overview on IoT tools and applications and to introduce to students hands-on IoT communication concepts through lab exercises.

**Learning outcomes:** Upon completion of this course, students will be able to explain the definition and usage of the term “Internet of Things” in different contexts. More specifically, the students will know how to apply the knowledge and skills acquired during the course to build and test a complete, working IoT system involving prototyping, programming and data analysis

**Teaching methodology:** interactive face-to-face lectures, group activities and discussions, in class/lab activities, student presentations and guest lectures or significant recorded public lectures

**Assessment:** Final exam (50%), midterm exam (20%) and assignments/project (30%).

### Main text:

Rajkumar Buyya, Amir Vahid Dastjerdi, Internet of Things Principles and Paradigms, Morgan Kaufmann; 1st edition, 2016

J. Biron and J. Follett, "Foundational Elements of an IoT Solution", O'Reilly Media, 2016.

### Other reading:

Jamil Y. Khan and Mehmet R. Yuce, Internet of Things (IoT) Systems and Applications, 2019, ISBN 9789814800297

David Hanes, Gonzalo Salgueiro, Patrick Grossetete, Robert Barton, and Jerome Henry, IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things, 2016, Cisco Press.



**INTRODUCTION**

# Architectural Design and Applications in IoT- Advanced

**CONTENTS**

1. IoT Solution Architecture Subsystems
2. Architecture Subsystems Details
3. Implementation considerations of IoT architecture

## INTENDED LEARNING OUTCOMES

Upon completion of this introductory unit, students will be:

1. familiar with the IoT Solution Architecture Subsystems
2. familiar with the Architecture Subsystems Details
3. familiar with the Implementation considerations of IoT architecture

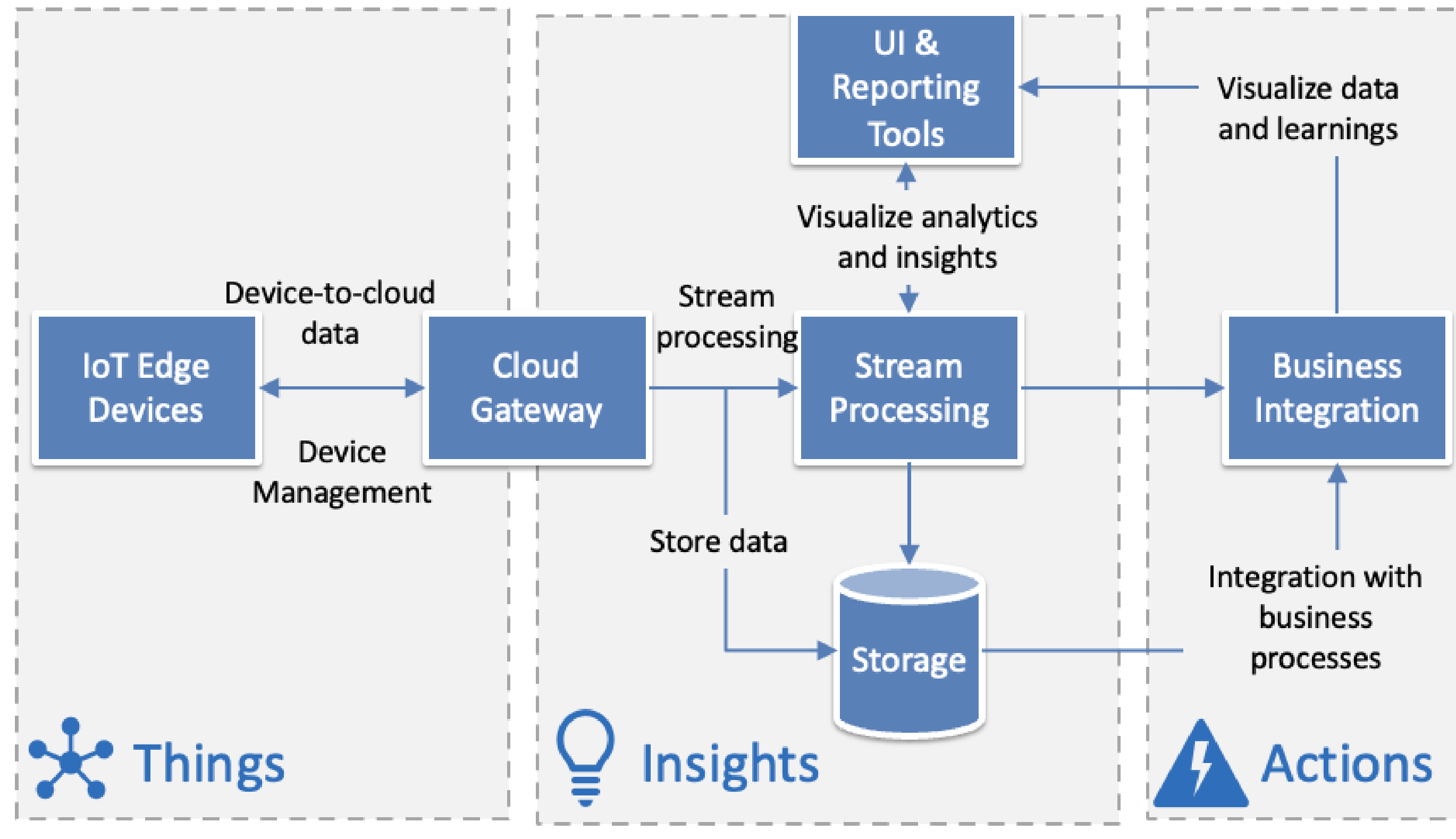
# IoT Solution Architecture Subsystems

## IoT Solution Subsystems

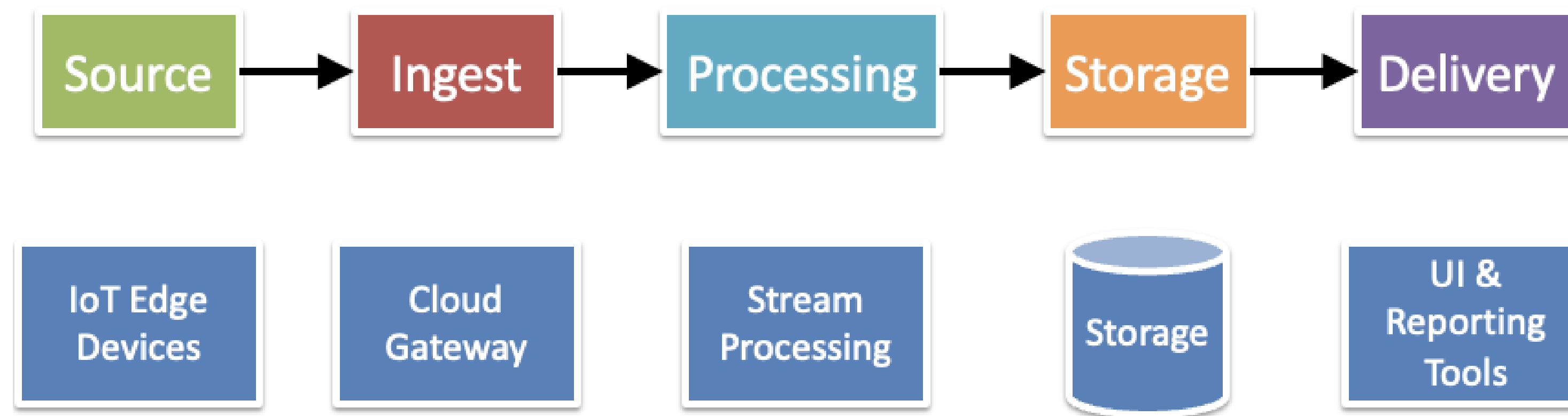
- Built as discrete services allowing:
  - independent deployment
  - ability to scale independently
  - ability to choose appropriate technology on a per subsystem basis
  - ability to monitor subsystems individually
- Cloud compute strategy: for batch data processing
- Edge compute strategy: light data processing on premise
- Service orchestrator for horizontal subsystem scalability



## Core Subsystems



## Analytics Data Pipeline

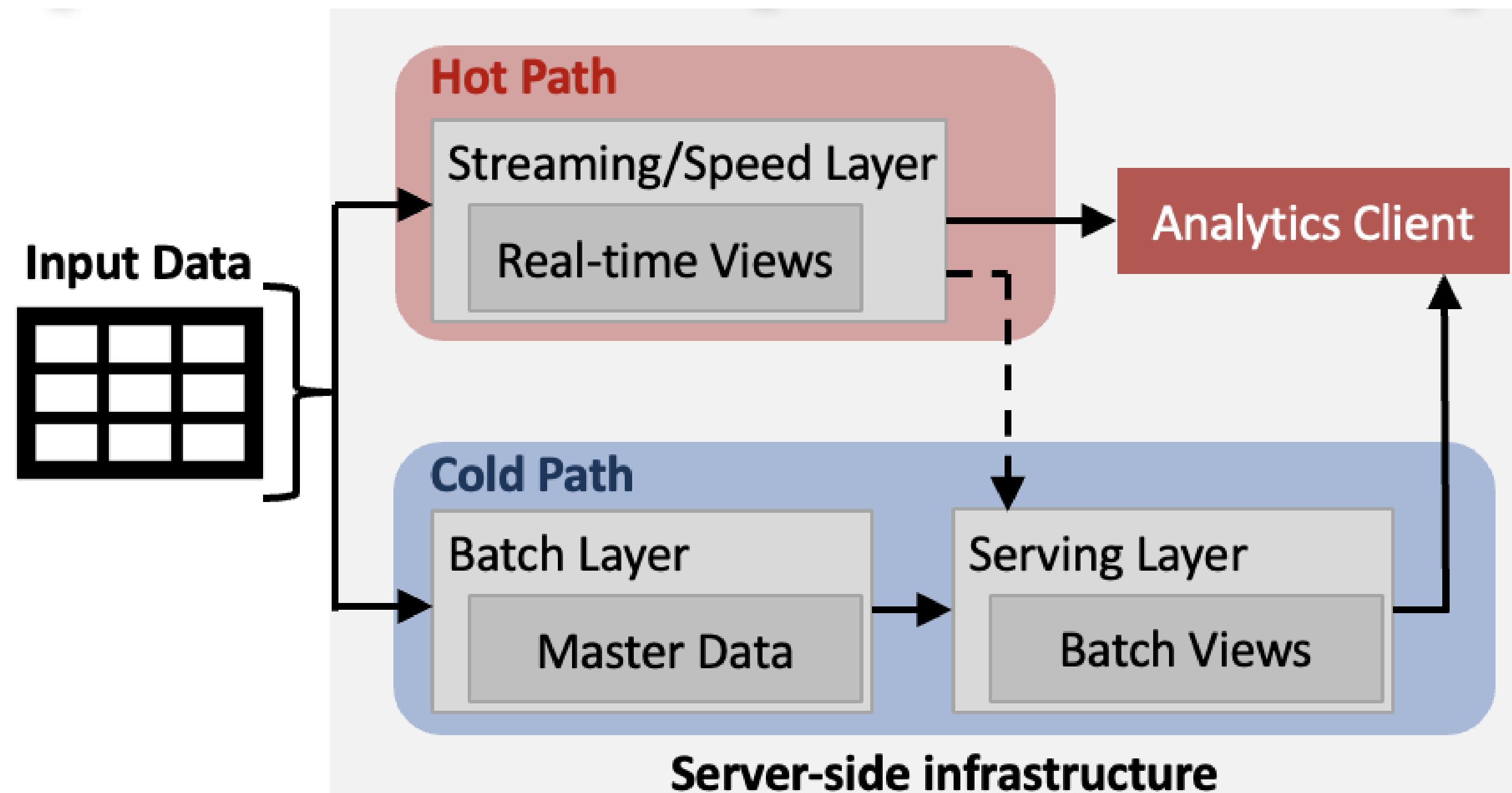




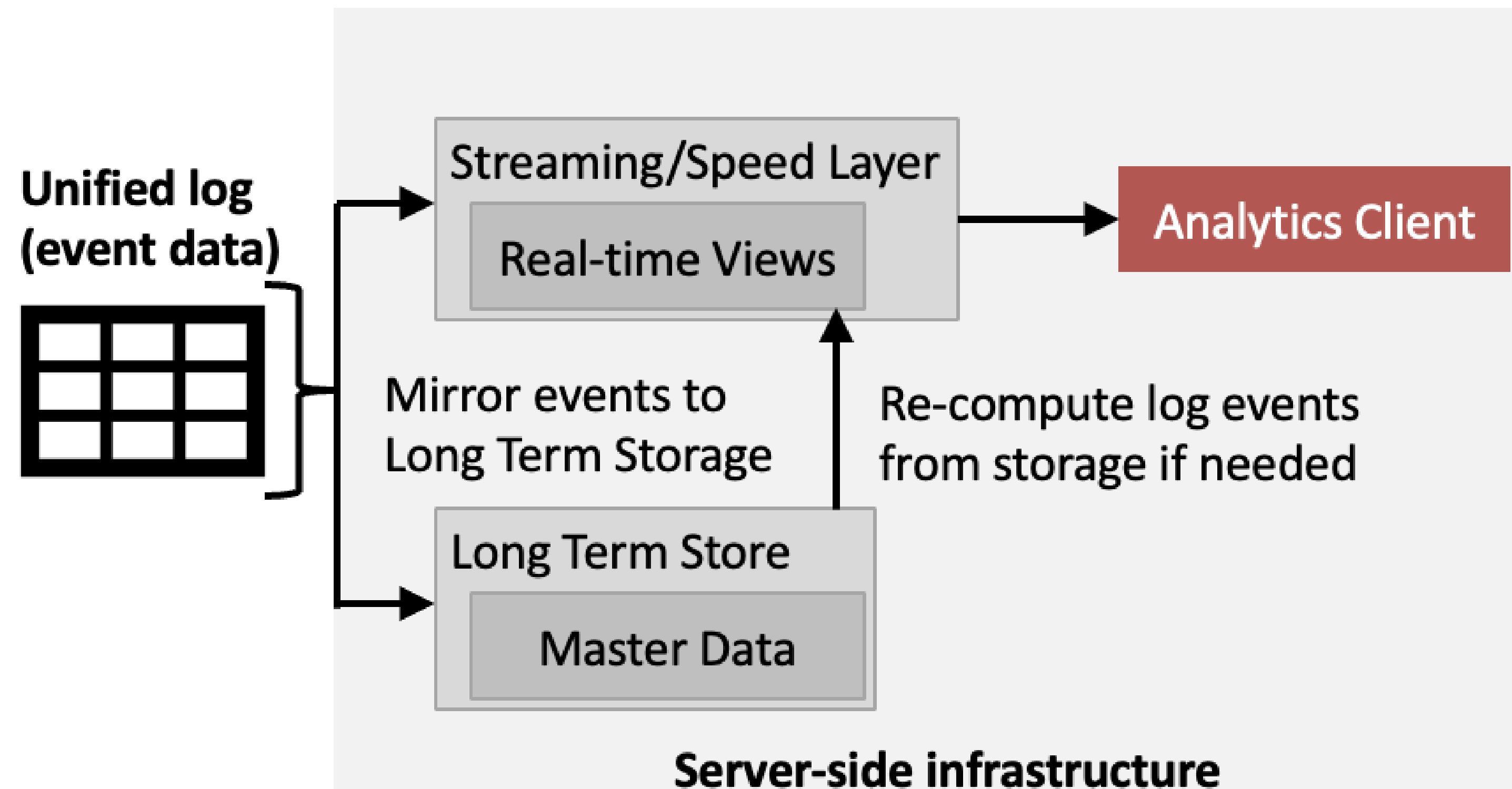
## Data Lakes

- Storage + Processing
- Storage: scalable, fault-tolerant, storage repository for massive volumes of data (big data)
- Processing: engine to operate on big data
- Architectures to act on data: Lambda ( $\lambda$ ) & Kappa ( $\kappa$ )

## Lambda Architecture



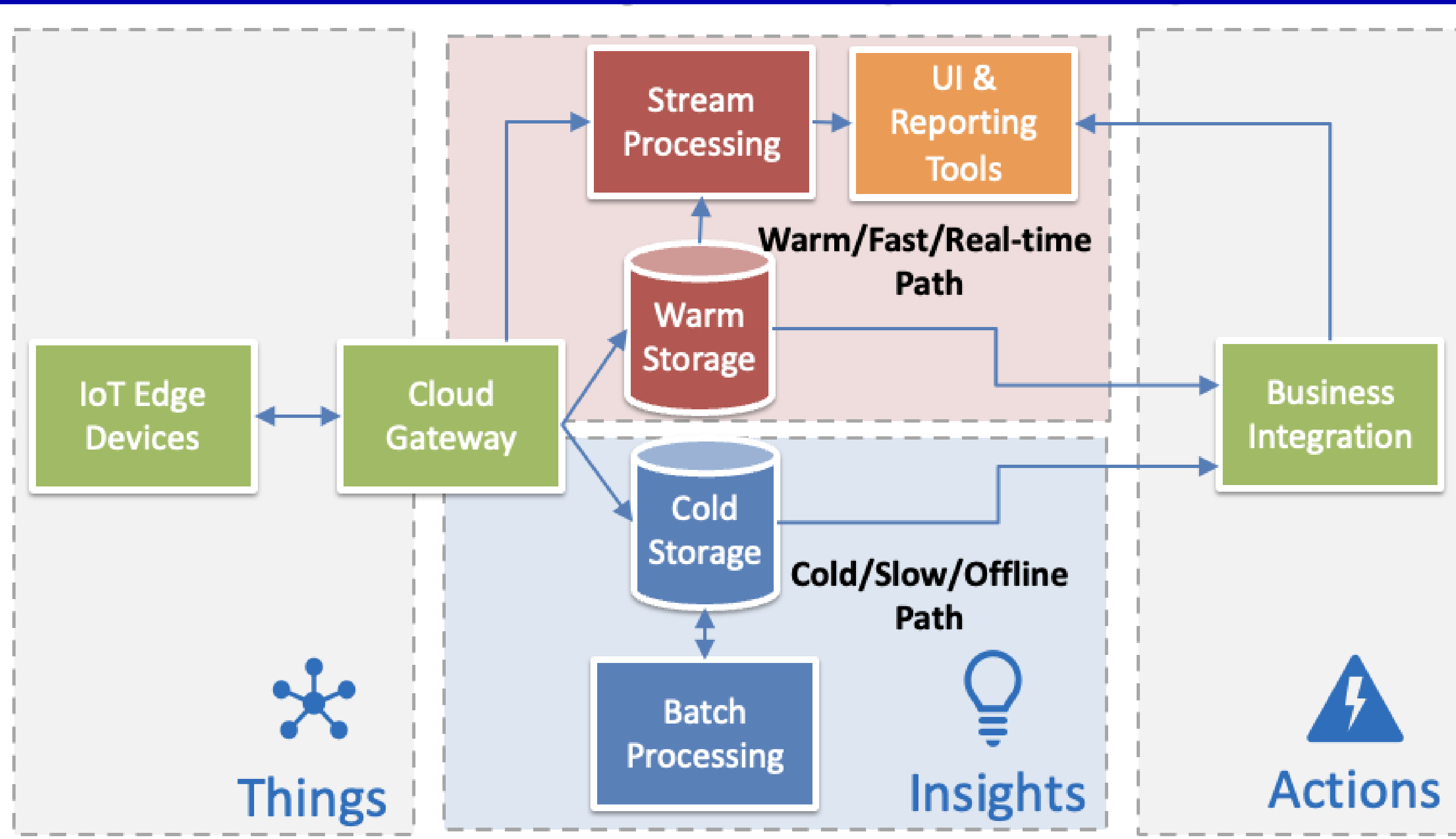
## Kappa Architecture



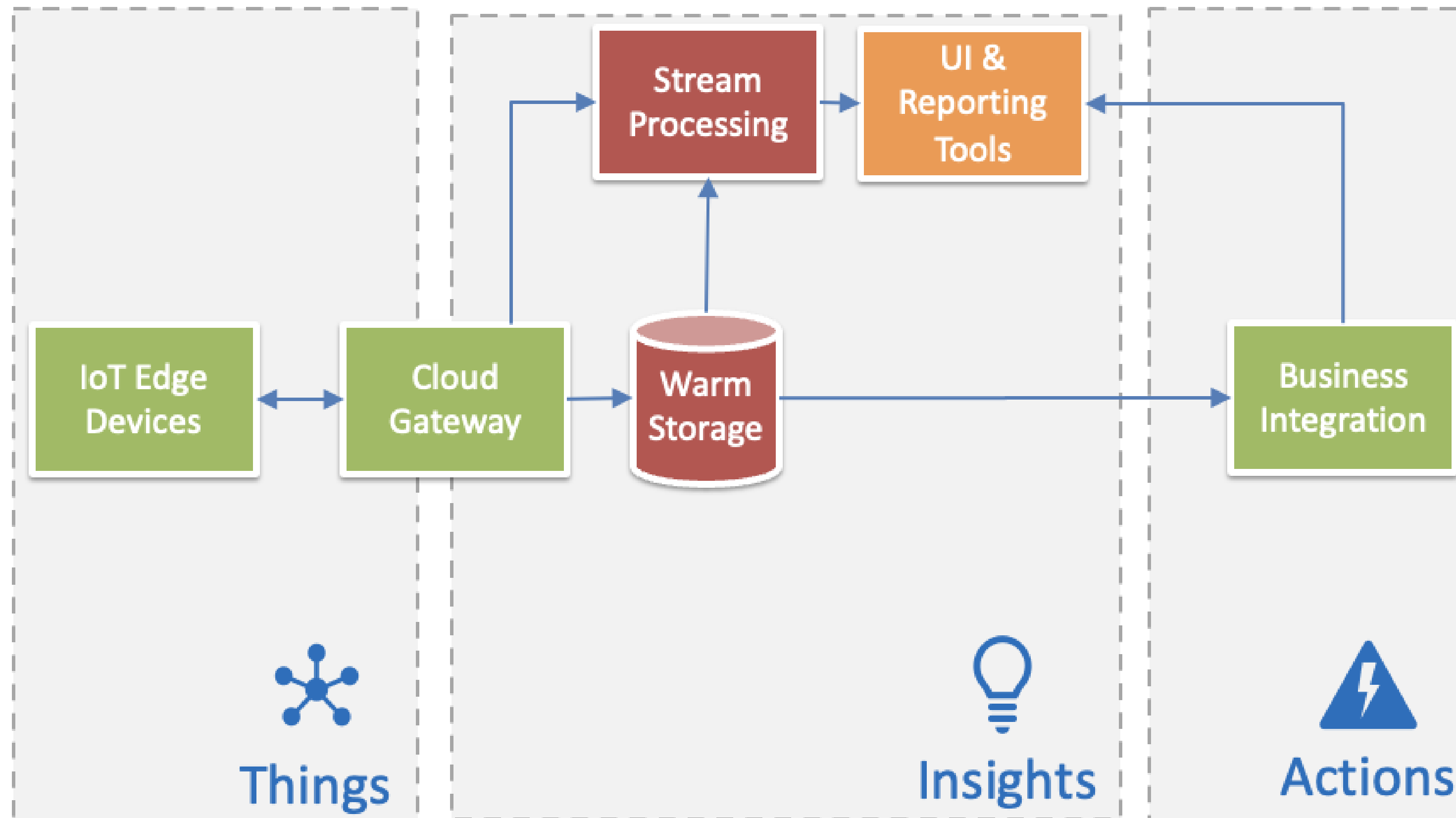
## Lambda vs Kappa

- No single optimal solution for all problems
- Is the analysis and processing that we are going to carry out in the batch and streaming layers the same? -> Kappa architecture
- Do batch and streaming algorithms generate very different results? -> Lambda architecture

## Core Subsystems (Lambda)

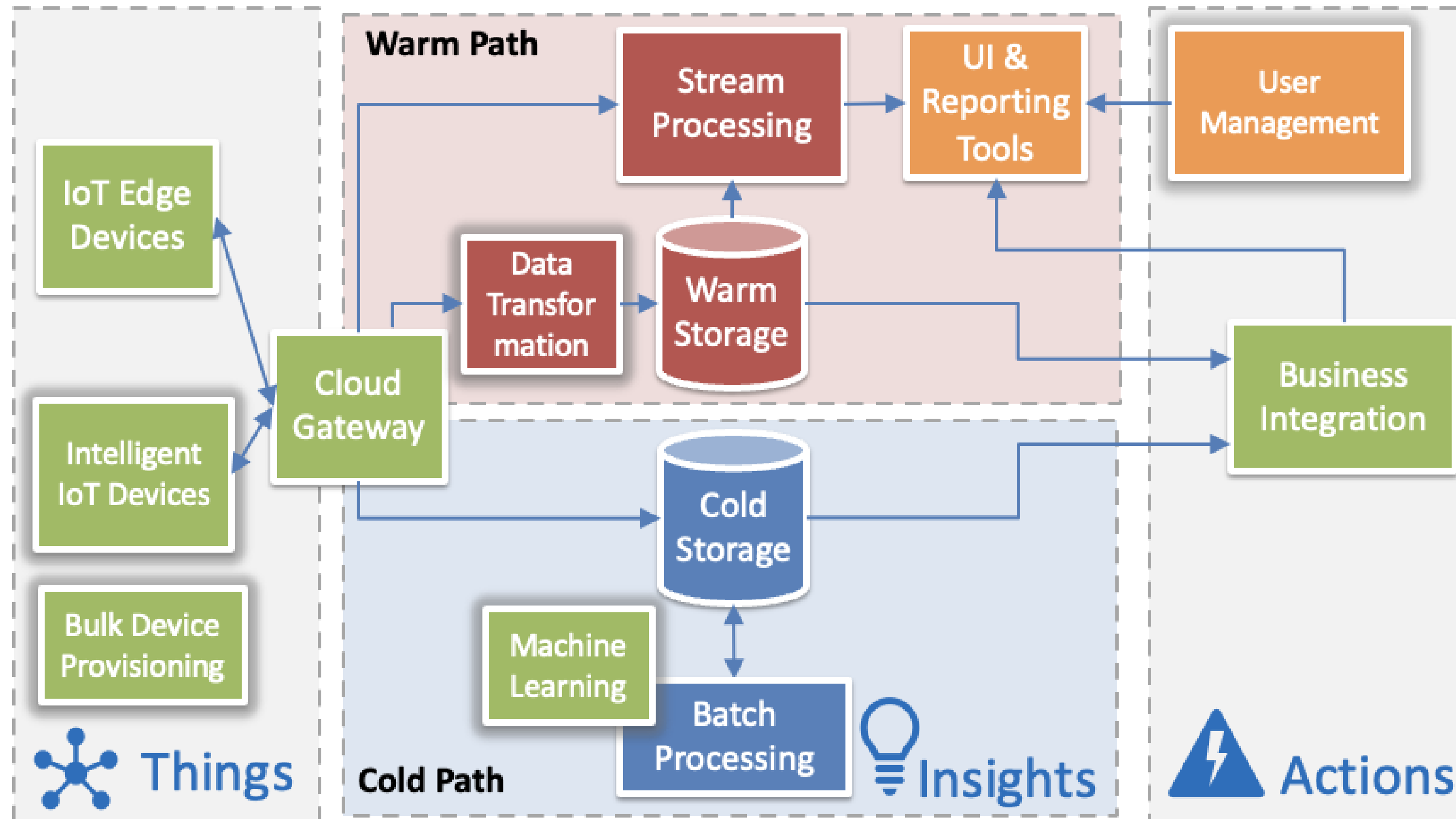


## Core Subsystems (Kappa)

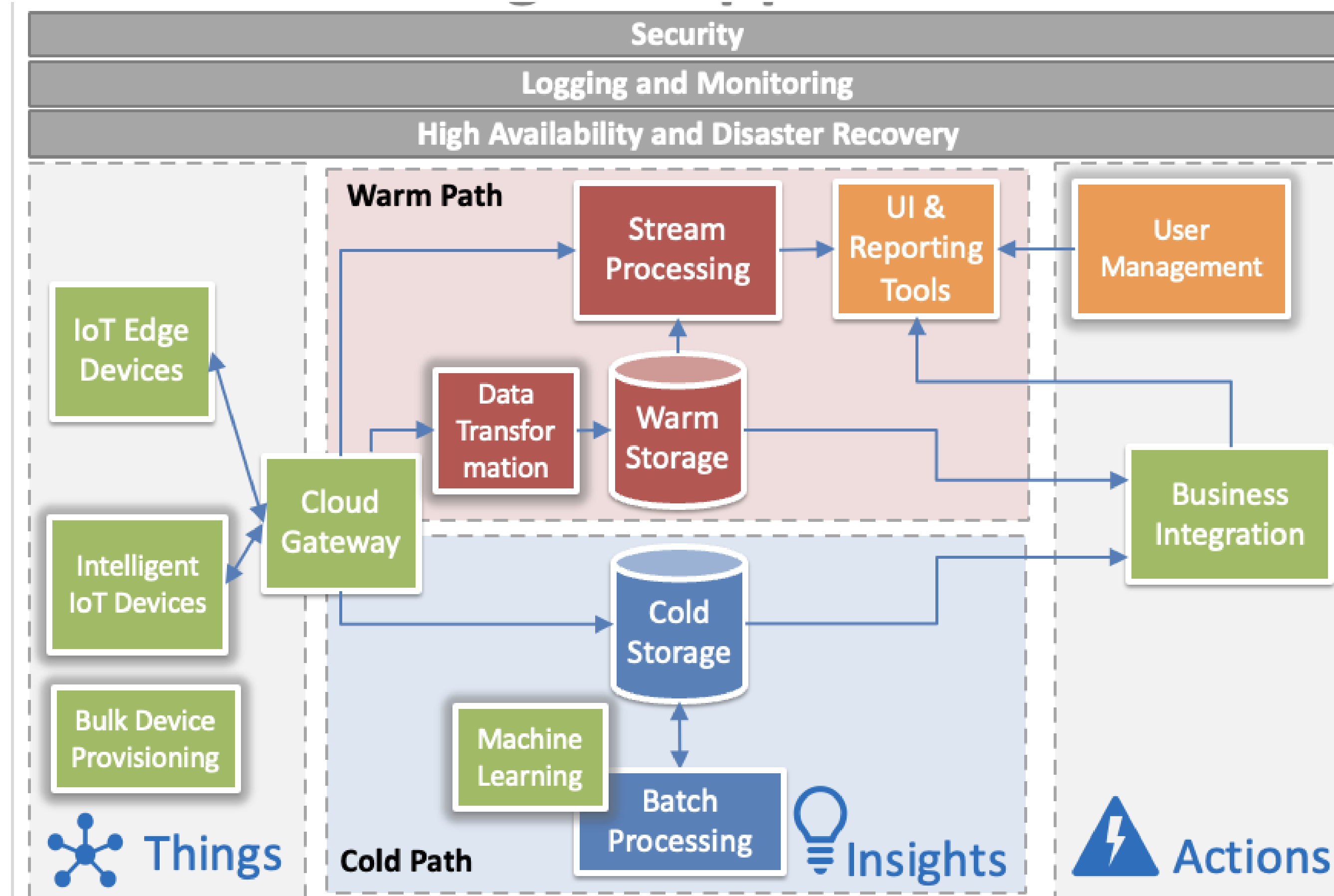




## All Subsystems (Lambda)

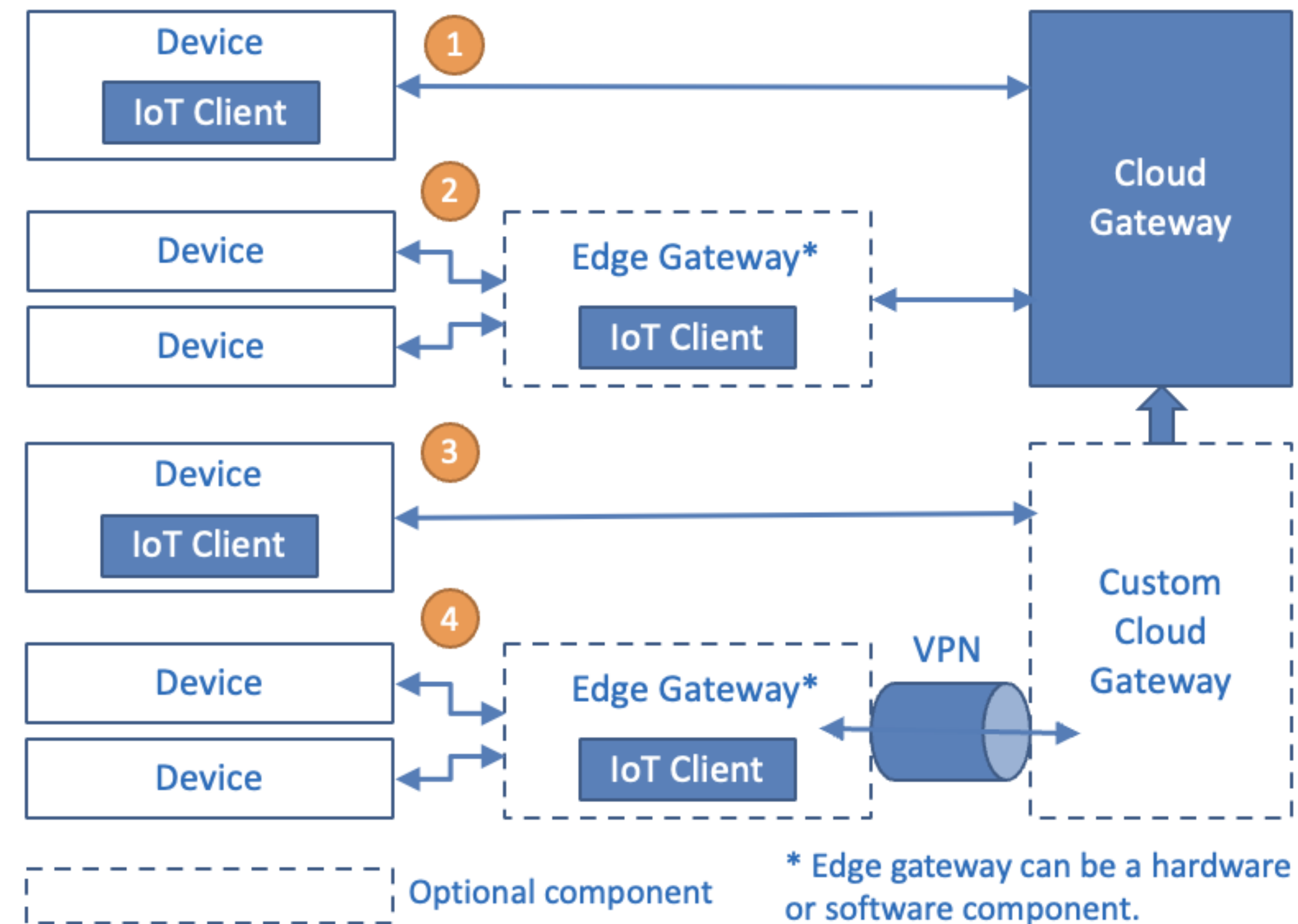


## Cross-Cutting IoT Application needs



# Architecture Subsystems Details

## Device Connectivity



## Devices: Technology options

- Edge Gateways
  - MS Azure IoT Edge
  - AWS IoT Greengrass
  - Google Cloud IoT Edge
- Cloud Gateways
  - Azure IoT Hub, Azure Event Hubs
  - AWS IoT Core
  - Google Cloud IoT Core
- IoT Client
  - Azure IoT Device SDKs
  - AWS IoT Device SDKs
  - Google Cloud IoT Device SDKs

## Device Entity Store

- Basic authority for device identity information
- Stores / allows for validation of cryptographic secrets for device client authentication
- Contained in cloud gateway or as an external service
- Used by device provisioning to create ids for new devices or to remove devices from system



## Device Entity Store: Technology options

- Relational databases: MySQL, MSSQL
  - identity store -> table
  - each device -> row
  - system-level device id -> primary key column, indexed
  - complex data as json if needed
- No SQL databases: MongoDB, Cassandra
  - Identity store -> collection
  - Each device -> row
- Proprietary cloud-based databases: Azure Cosmos DB, Amazon Dynamo DB

## Topology and entity store

- Database containing:
  - application entities (products, assets, machines)
  - relationships among the entities
  - device metadata (e.g. geo-position, model)
- Technology options:
  - SQL (relational) databases
  - No SQL databases
  - Proprietary cloud-based databases

## Storage

- Edge devices generate significant amounts of data
- Need to be stored for visualization/reporting/processing
- Data stores categories:
  - Warm: recent data (e.g. last day, week or month)
    - Small amount of data for fast access (near-zero delay), easily queried
  - Cold: old historical data
    - Huge amount of data, slow access (seconds to minutes)

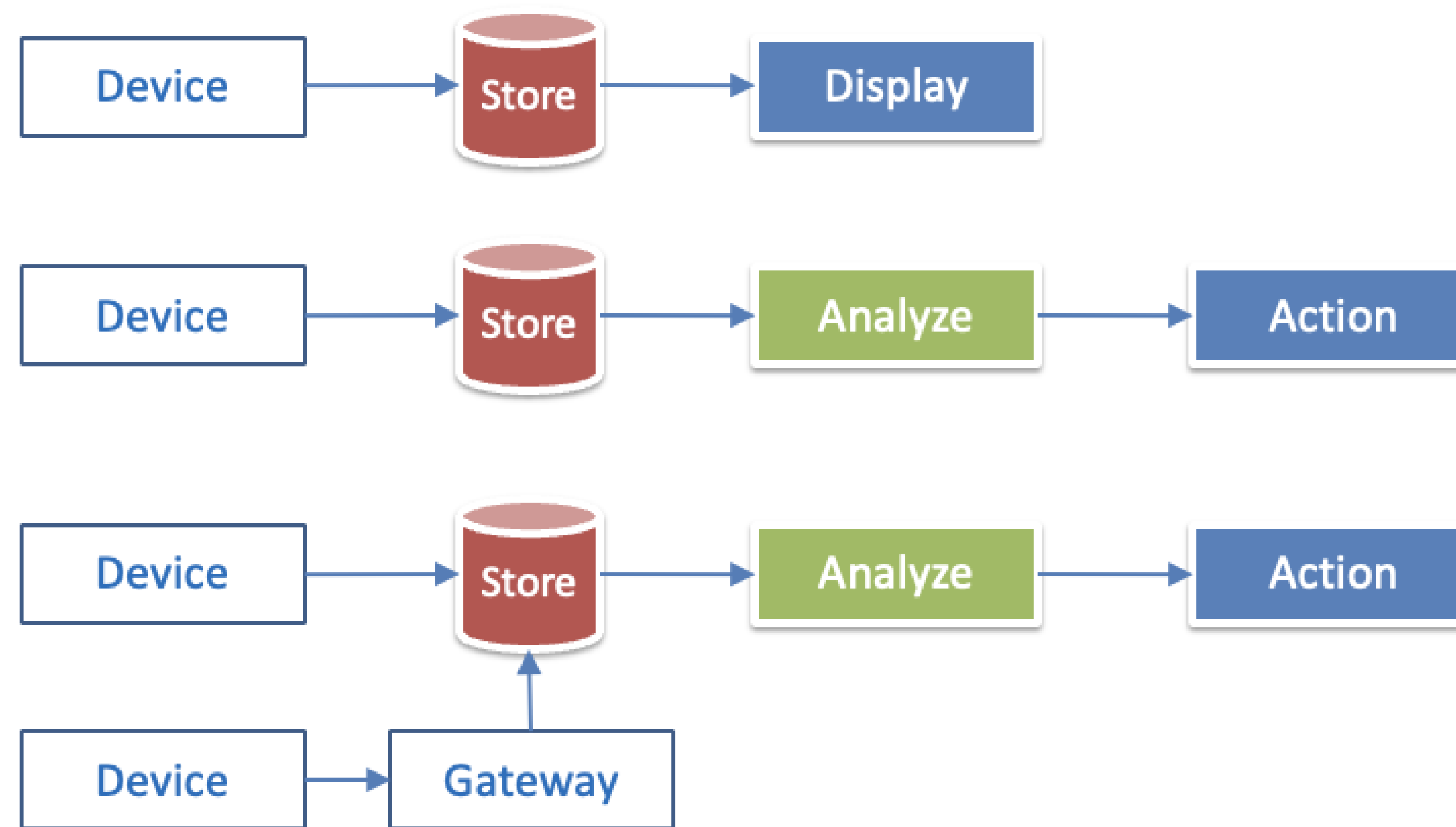
## Storage: Technology options

- Warm storage:
  - In-memory databases
  - SSDs
- Cold storage:
  - On-disk databases
  - Hard-disk drives
- Managed storage services
- Self-managed storage services

## Data Flow and Stream Processing

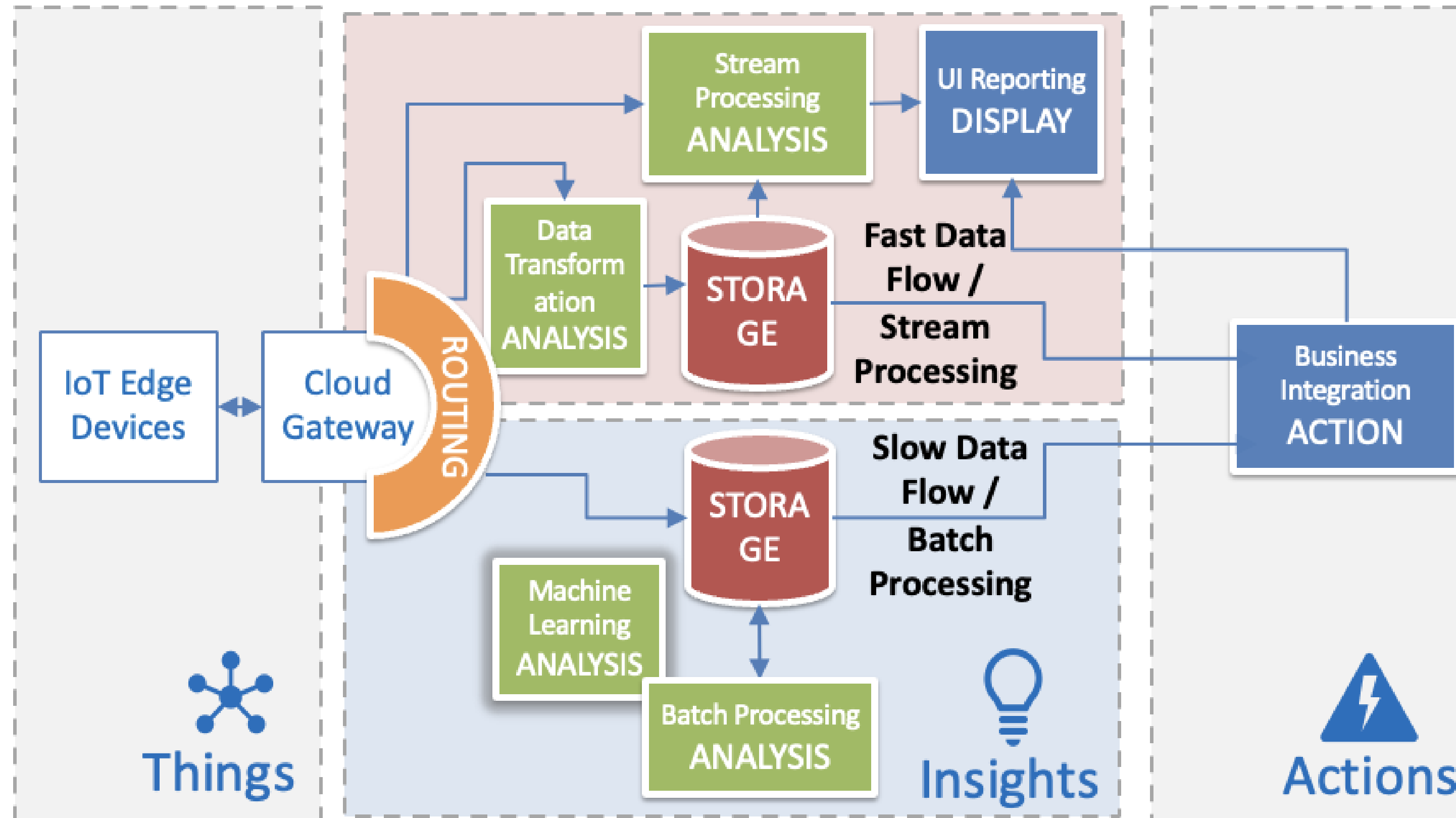
- **Storage:** in-memory caches, temporary queues, permanent archives
- **Routing:** dispatching of data records to one or more storage endpoints, analysis processes, and actions
- **Analysis:** run input data records through a set of conditions & produce different output data records
- **Action/Display:** Original input data records and analysis output records are typically stored and available to display, and may trigger actions such as emails, instant messages, incident tickets, device commands, etc.

## Data Flow and Stream Processing










## Lambda Architecture Data Flows



## Stream processing

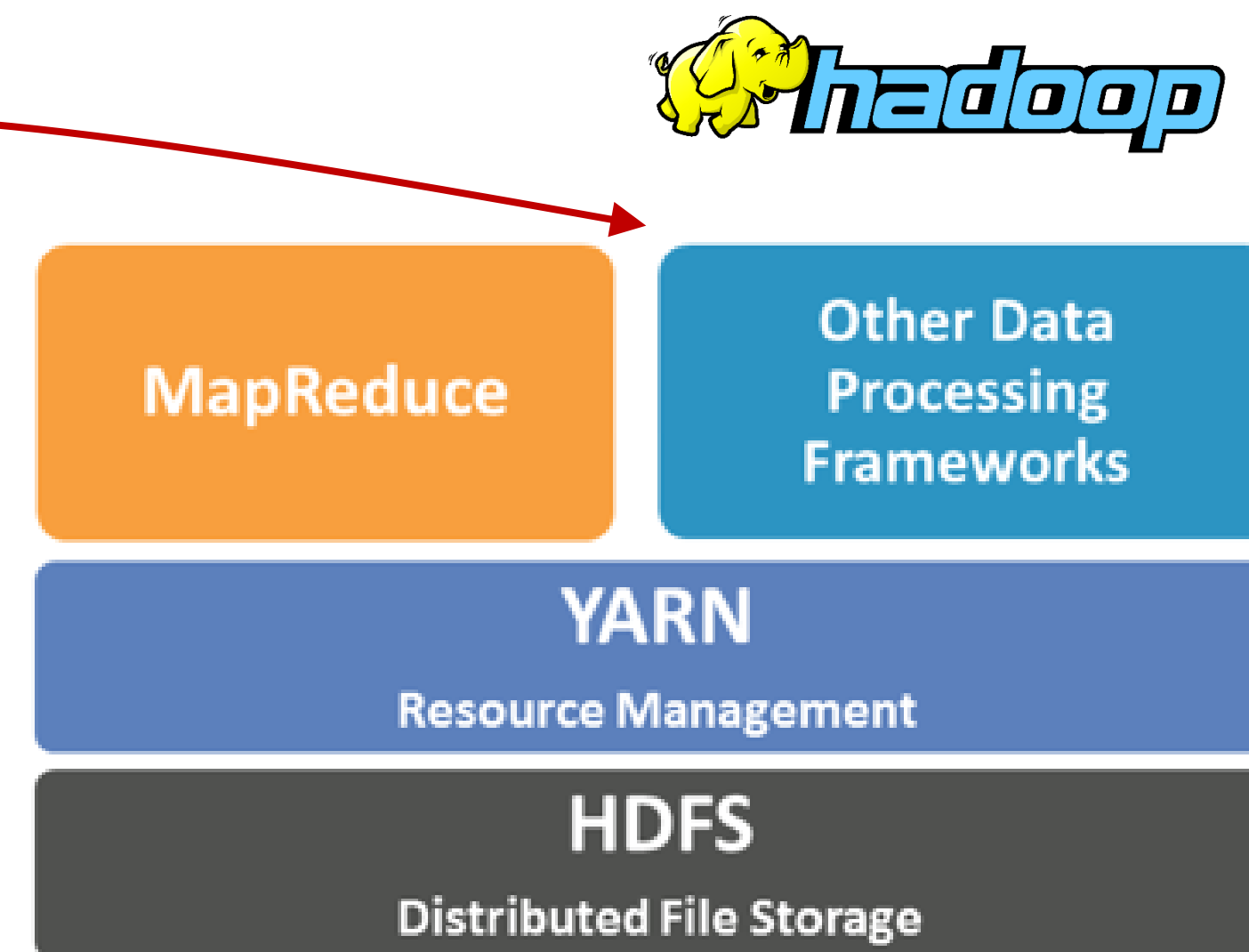
- Processing of data in motion, as it is received
- Important building block of real-time applications
- State-less stream processing: no memory on event handling, easy to scale
- Stateful stream processing: “state” shared between events, difficult to scale

## Stream processing: Technology options

- Apache Spark 
  - contains Spark Streaming
  - in-memory data processing of data in motion or at rest
  - analytics, machine learning, SQL processing, graph processing
- Apache Kafka 
  - messaging and stream processing platform
  - collect & route data to stream processors
  - perform stream processing natively on top of Kafka Stream API
- Apache Storm 
- Apache Samza 
- Apache Flink 

## Batch processing

- Designed to process big data at rest efficiently
- Requires a large set of data collected over time
- Allows for high accuracy computation
- Technology options:
  - Apache Hadoop:
    - Map-Reduce + HDFS + YARN
  - Apache Spark
    - Runs on top of existing Hadoop clusters for enhanced (100x faster) and additional functionality (machine learning)



## User Interface

- Visualization of device data
- Analysis results
- Notifications and alerts
- Device discovery
- Command and control capabilities

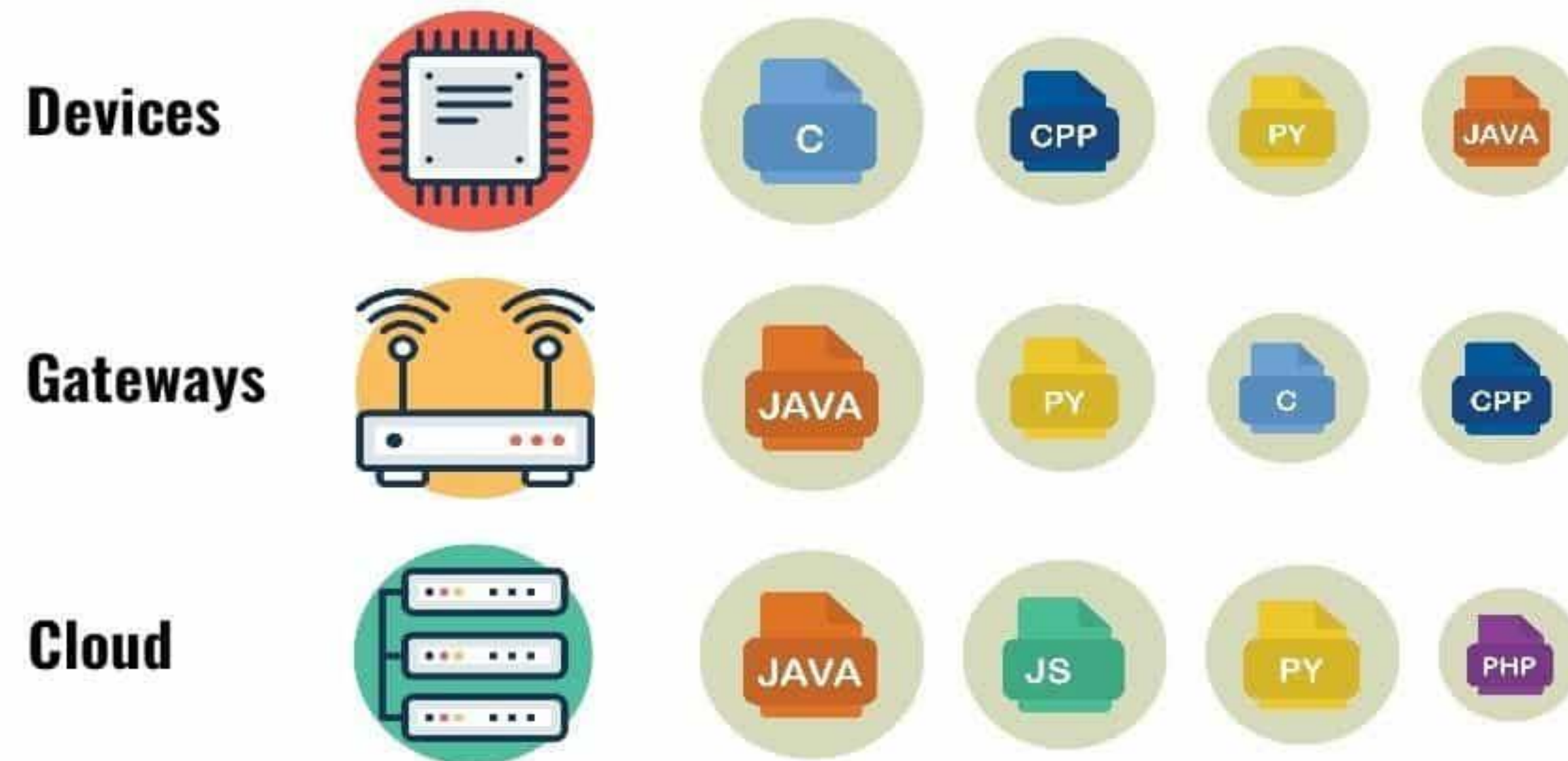


- Interactive dashboards when manipulating large population of devices



## User Interface: Technology options

### TOP IoT PROGRAMMING LANGUAGES



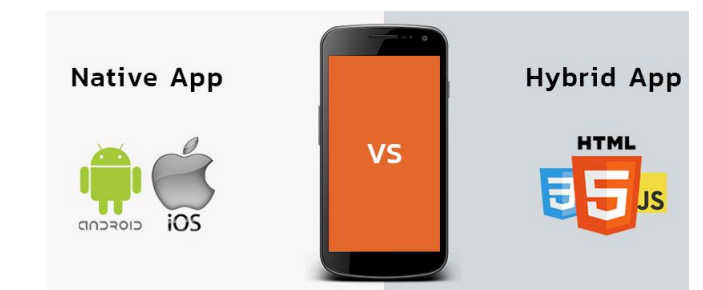
Copyright (c) 2019, Eclipse Foundation, Inc. | Made available under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

9



## User Interface : Technology options

- Native apps
  - Coded in a specific programming lang. such as Java for Android
  - Greater access to phone's capabilities
  - Time-consuming development (Java □ Android, iOS □ Swift)
  - No compatibility between mobile platforms
- Web apps
  - Built mainly with HTML/CSS/Javascript/PHP
  - Accessed by phone's web browser by going to a certain url
  - Compatible with all platforms
- Hybrid apps
  - Built mainly with web technologies or general-purpose language such as C# - translated to native instructions at runtime
  - Need to be downloaded
  - Slower execution than native apps
  - Good for creating MVPs - Fast development



## Hybrid App Development: Frameworks and languages

- Flutter is mobile app SDK by Google used for developing high-quality native interfaces on Android and iOS
- React Native is a framework by Facebook in which developers build products using JS and React
- Xamarin is a set of tools (framework) in which developers code using C# and .NET
- NativeScript is a framework in which developers build code in JavaScript and Angular
- Cordova/PhoneGap/Ionic are frameworks in which developers use HTML, CSS and JavaScript



## Monitoring and Logging

- System performance and timeline visualization tools
  - to monitor the system and for basic troubleshooting.
- Buffered data ingestion
  - to buffer log data
- Persistence store
  - to store log data.
- Search and query capabilities
  - to view log data for use in detailed troubleshooting.

## Monitoring

- Metrics for health, security, stability, performance:
  - Physical devices, edge devices, and infrastructure components
    - configuration changes patches, services, users
    - power consumption, CPU, memory, disk usage
  - Applications
    - configuration changes, security audit logs, request rates, response times, error rates, garbage collection statistics
  - Databases, persistence stores and caches
    - query / write performance, schema changes, security audit log, locks or deadlocks, index performance, CPU, memory, disk usage
  - Managed services (IaaS, PaaS and SaaS)
    - health metrics and configuration changes



## Logging

- Understanding actions/activities performed, failures occurred, error conditions
- Models for logging info:
  - Text-based: unstructured info

```
2017-12-18T08:15:30 [INFO] - ClientAuthed, dev/opcpub, 35f634d7
2017-12-18T08:15:30 [INFO] - New device connection for device dev/opcpub
2017-12-18T08:15:30 [INFO] - Bind device proxy for device dev/opcpub
```

- Structured-based: fixed schema to enhance search and query capabilities

```
{"@t":"2017-12-18T08:15:30","@l":"info","deviceId":"opcpub","@m":" ClientAuthed, 35f634d7."}
{"@t":"2017-12-18T08:15:30","@l":"info","deviceId":"opcpub","@m":" New device connection."}
{"@t":"2017-12-18T08:15:30","@l":"info","deviceId":"opcpub","@m":" Bind device proxy."}
```

## Technology options

- Monitoring tools
    - Microsoft Operations Management Suite
    - Amazon CloudWatch
    - Splunk\*
    - Elastic Stack (Elasticsearch, Logstash, Kibana)
    - Zabbix
    - Nagios
  - Logging tools
    - Serilog
    - log4J
- } **Commercial**  
**(\* there is a free version)**
- } **Free**

## Business System Integration

- IoT applications can be integrated with existing business systems across an organization:
  - Customer Relationship Management (CRM)
  - Enterprise Resource Planning (ERP)
  - Line-of-business applications (LOB)
- Business integration layer is responsible for integrating IoT environment into business systems
- IoT applications tie to business systems mainly through APIs, using predefined content types



## Machine Learning

- Towards smart, autonomous IoT applications
- Machine Learning algorithms used to:
  - extract meaningful information and identify patterns out of the enormous amount of telemetry data
  - provide predictions
- Real-time ML analysis in stream processing
- Offline ML analysis in batch processing
- ML outcomes can be used for decision making aids such as alerts and recommendations

## Technology options

- Weka
  - regression, clustering, classification
  - Java API
- Apache Mahout
  - Scalable ML library
  - Follows Map-Reduce paradigm on top of Hadoop
  - item recommendation, clustering, and classification
- Apache Spark Mlib
  - Java, Scala API
  - regression, clustering, classification, collaborative filtering
- Tensor Flow
  - ML and deep learning framework for high-performance numerical computations created by Google

# Implementation considerations of IoT architecture

## Device intelligence

- Intelligent devices vs intelligent system
- More intelligence on edge => more software updates of edge components at higher frequencies
- More intelligence on system's backend => maintenance in a centralized fashion & higher security levels
- Changes gradually easier when moving from device/edge software to cloud backend
  - start planning from devices

## Device telemetry

- Type and frequency of telemetry data should be driven by business requirements
- Problems of collecting too much data:
  - no guarantee that right business questions can be answered
  - difficult to differentiate useful info from noise
  - high storage and processing costs
- Good strategies:
  - program devices to emit different granularity of telemetry data
  - different categories of data can be treated differently



## Communication protocols

- TCP/IP are the defacto network/transport protocols
- Physical/link layer protocols impact QoS & dictate the frequency and communication patterns of edge devices:
- radio networks (GSM, 3/4G) suffer from packet corruption and loss as a result of interference
- battery-operated devices optimize send/receive patterns for lower power consumption => not reachable at all times
- roaming leads to frequent connection / IP address switching



## Communication protocols

- Mitigation of communication issues:
  - Design communication patterns on the basis of the underlying (physical/link layer) technologies:
    - Communication model switching based on connectivity type to avoid battery drain and high data transfer charges
    - Service-assisted communication using out-of-band channels (e.g. SMS) to “wake up” battery-constrained sleeping devices
  - Enable VPN technology for integrating and isolating the network involving edge devices & cloud gateway
    - single & stable address space
    - secure join and participation of edge devices

## Messaging protocols

- Hypertext Transfer Protocol (HTTP)
  - Request/response communication, unidirectional
  - High overhead: text-based messages with headers
  - Ideal when devices send data to cloud occasionally & low-latency and bi-directional communication is not required
- Advanced Message Queueing Protocol (AMQP)
  - Bi-directional communication with compact data encoding
  - Ideal for long-lived connections transferring large data amount
- WebSocket protocol
  - Bi-directional layer over TCP; negotiation over HTTP/HTTPS
  - Allows tunnelling other protocols (e.g. AMQP) through HTTP/HTTPS infrastructure and ports for firewall traversal

## Messaging protocols

- Message Queue Telemetry Transport (MQTT)
  - Lightweight client-server protocol with small footprint
  - No support for message metadata
- Constrained Application Protocol (CoAP)
  - Datagram-based protocol over UDP or SMS
  - Compact reformulation of principles & methods of HTTP
  - Complicated when security is added

## Security

- Information send/received from/to edge device must be trustworthy
  - Verifiable origin, unaltered, timely
- IoT devices usually trade security for cost savings
- Recommended security-oriented device capabilities:
  - Symmetric-key data encryption e.g. AES with 128 bit key
  - Symmetric-key signature algorithm e.g. SHA-2 with 128 bit key
  - TLS 1.2 for TCP or DTLS 1.2
  - Updateable key-store and per-device keys
  - Updatable software to repair security vulnerabilities



## Physical tamper proofing and safety

- IoT devices in public areas are vulnerable to physical attacks
- Digitally trustworthy devices may be tricked into reporting misleading data by dismounting, relocating or misleading physical conditions in device proximity
- Measures to improve the security of the physical device:
  - Choose microcontrollers or auxiliary hardware providing secure storage and use of cryptographic key material
  - Use sensors to detect intrusion attempts and attempts to manipulate the device environment

## Data Encoding

- XML
  - Popular data exchange format, large file size footprint
  - Broad library support
- JSON
  - Lightweight textual format compared to XML, faster parsing
  - Broad library support
- CSV
  - Structurally constrained to row / columns; enough for timeseries
- BSON / MessagePack
  - Binary encoding, very small file size footprint, lack of arrays (BSON)
  - Requires own libraries
- Protobuf / Apache Thrift
  - Very small encoding size; require distribution of schema
- Apache Avro
  - Binary encoding; schema embedded as preamble
  - Higher payload compared to BSON



## Data Layout

- Defines structure (schema) of data
- All IoT application subsystems must rely on common schema
- Of equal importance as data encoding
  - Major impact on encoded data size

```
[ {"id":3, "name":"tom"}, {"id":4, "name":"john"} ]  
{"id": [3, 4], "name": ["tom", "john"]} `
```

## Edge Processing

- Raw sensor readings pose significant:
  - costs when sent via metered networks
  - load burden to cloud system when many unprocessed data streams handled in parallel
- Timeseries data can be aggregated by grouping to a single record or by providing average/median
- Signal data streams like video & audio should be preprocessed and compressed using broadly accepted industry standards
- Pre-analysis, aggregation and compression can be done on the basis of underlying technologies or network conditions

## Device Management

- Heterogeneous IoT landscape drives the need for a device management protocol enabling:
  - Device provisioning and discovery
  - Device access management
  - Remote control
  - Remote administration and monitoring
  - Remote configuration
  - Remote firmware and software update

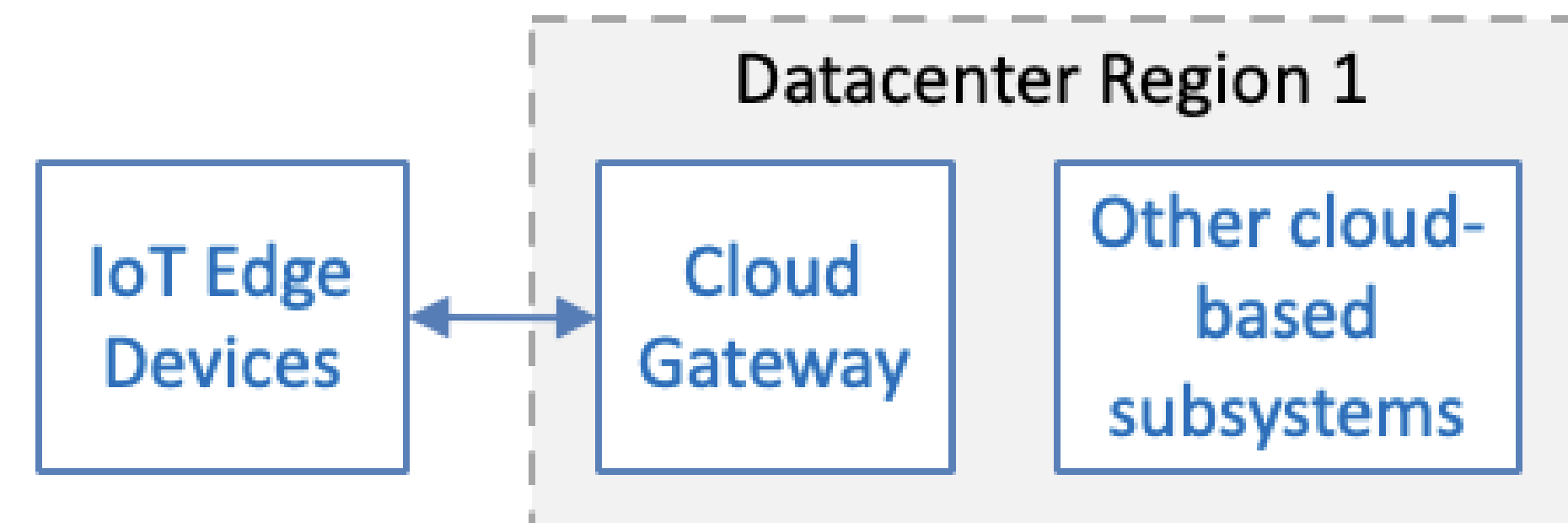
## Device Provisioning and Discovery

- Device deployment location not known at manufacturing
  - Set bootstrap service to provide cloud gateway at device initialization phase
- Device is described to cloud using:
  - Self-defined device model
  - Pre-defined device model
  - Pre-defined master model
- Remote control
  - Interactive connection (connection needed via SSH/RDP)
  - Device command (via existing connection between device & cloud gateway)
- Remote administration and monitoring
  - state or configuration updates: via device commands
  - health status: via received data monitoring
- Remote firmware and software updates
  - firmware & software updates defined through device model
  - updates are initiated at IoT backend; device informed via command
  - Device downloads & deploys update, verifies and identifies good state or initiates rollback

## High Availability and Disaster Recovery

# Single-site cloud backend topology

- cloud gateway & data stores in a single data center

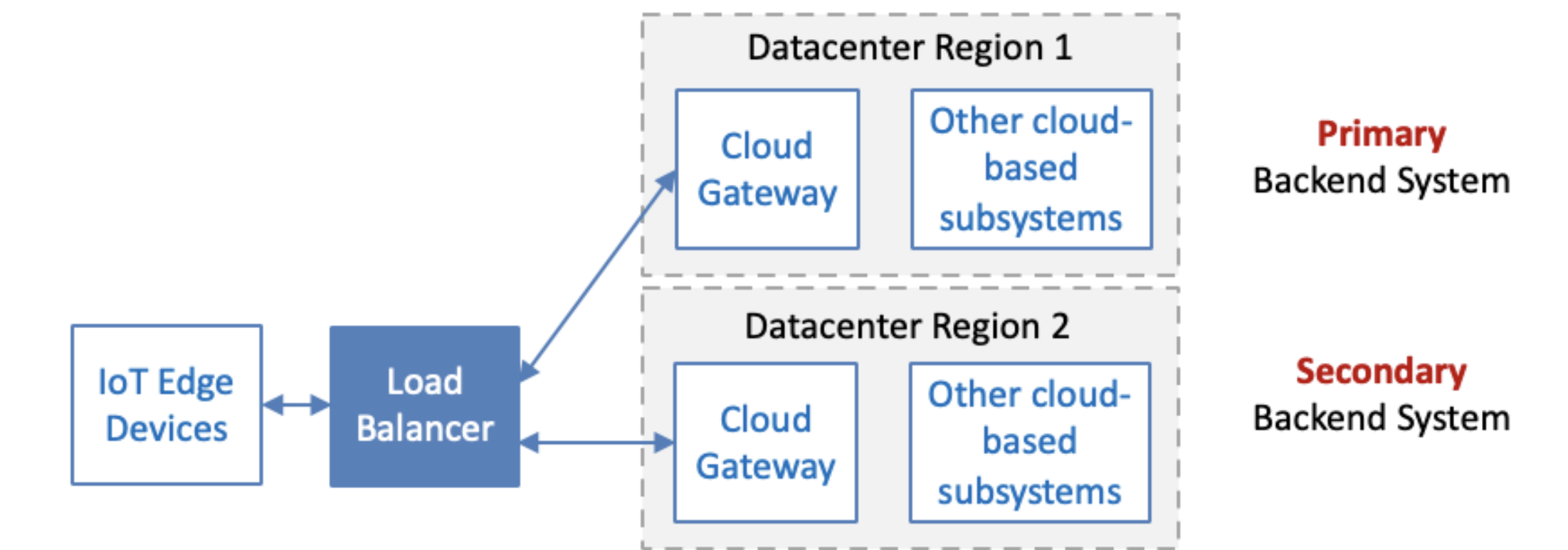




**High Availability and Disaster Recovery**

# Regional failover cloud backend topology

- primary backend system deployed in one datacenter region
- secondary backend system deployed in additional datacenter
- Load balancer (LB) manages traffic from edge devices
- LB health checks help isolating non responsive cloud gateways

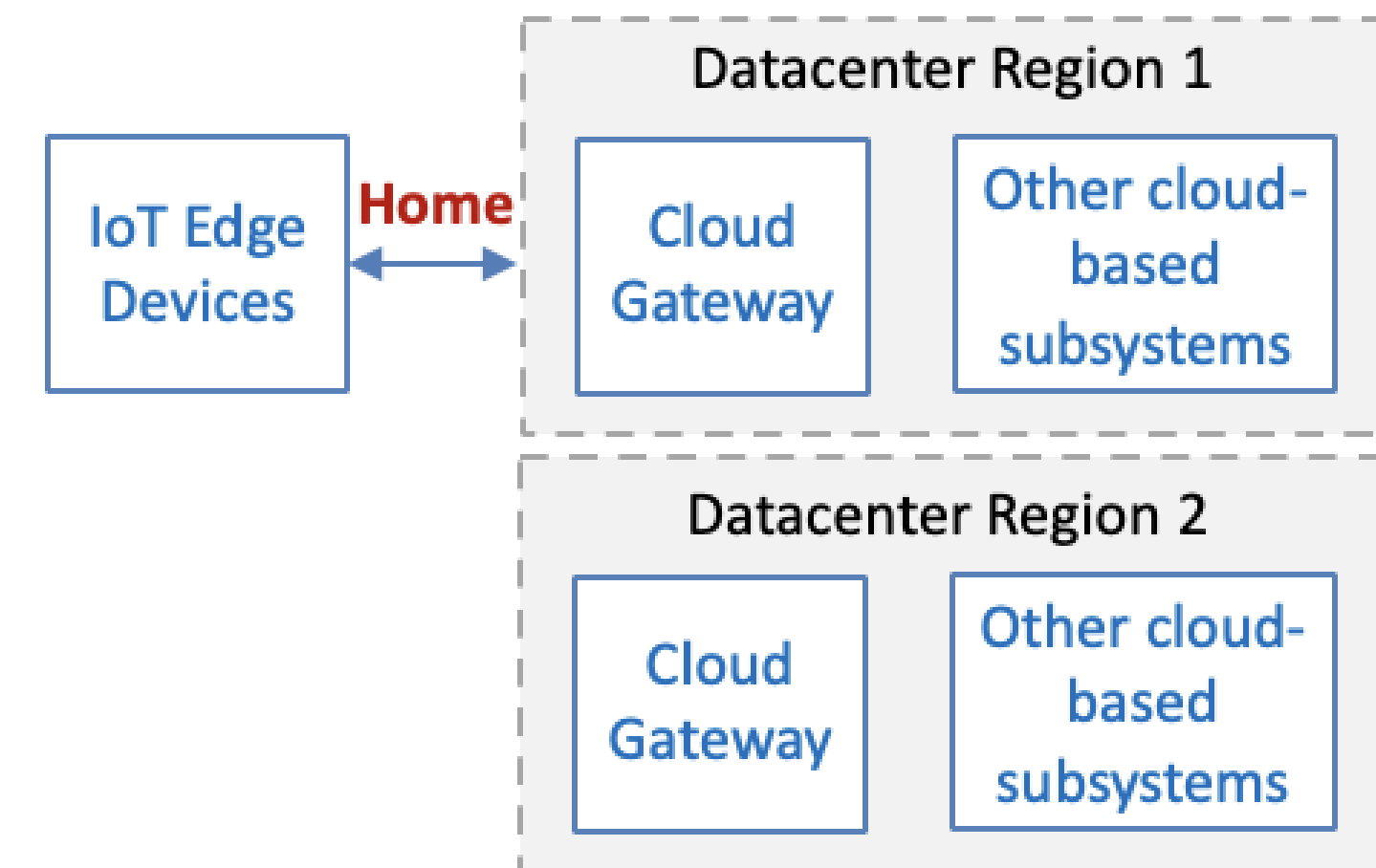




**High Availability and Disaster Recovery**

# Multi-site cloud backend topology

- Backend system runs concurrently in multiple independent sites
- Sites can be collocated in the same datacenter region or across different data center regions
- Devices registered (homed) in one of those sites
  - Based on proximity



## High Availability and Disaster Recovery

- Multi-site with roaming
  - devices on move connect every time to the closest datacenter based on proximity estimation
  - Information routed to home site
- Multi-site/multi-home
  - device may roam across sites; home site changes
  - captured data is stored in new home site

## Data protection and privacy

- Data collection and remote-control scenarios are being subject to increased regulations
- Solution builders must anticipate regionally differing regulations
  - Owners/equipment operators can opt-out for data collection, erase data for past periods
  - Owners/equipment operators may have full control over usage rights & retention duration of their data
  - Opt-in and opt-out scenarios to occur retroactively long after the data has been collected

# Summary

- ❑ **IoT Solution Architecture Subsystems**
  - ❑ **Core subsystems**
  - ❑ **Optional subsystems**
- ❑ **Architecture Subsystems Details:**
  - ❑ **Purpose of each subsystem**
  - ❑ **Technology options for subsystem implementation**
- ❑ **Implementation considerations of IoT architecture**

