# Fish Watch

*SEAMLESS FARMING: MONITORING, ALERTS, AND INSIGHTS FOR FISH FARMS TO DETERMINE OPTIMAL HARVEST*

*PRESENTED BY : TEAM EVANGELIST*

*LEAD : KULDEEP SHRIVASTAV*

*DEV: AKHIL NAYTA , TARUN MITTAL*

Delivering Transformation. Together.

sopra steria

## KEY REQUIREMENTS

- **Scalability**: The system should be able to handle varying scales of customers, from those with a single farm to those with hundreds, each containing potentially millions of fish.
- **Real-time Data Processing**: The architecture must support real-time processing of data from water monitors(Including PH, temperature, salinity, oxygen levels and other factors), underwater cameras (fish health , Size , activity,  parasite detection), and fish recognition systems (to monitor health and life cycle of fish),   to provide timely insights to farmers .
- **Customizable Dashboards**: Farmers should be able to customize their dashboards to view relevant information about their farms, including water quality, fish health, and alerts.
- **Alerting Mechanism**: The system needs to trigger alerts based on customizable thresholds set by farmers, including both simple parameters like pH levels , more complex events like adverse weather conditions and water quality.
- **Data Modeling for Harvest Optimization**: The architecture should support the collection and analysis of data to build AI models that predict factors contributing to successful harvests.

## KEY REQUIREMENT( CONT..)

- **Cross-Farm Insights**: Large customers should be able to derive insights across multiple farms, enabling centralized management and decision-making.
- **Device Accessibility**: The system should be accessible from various devices ( mobile , web application), including rugged industrial devices used at sea during harvests.
- **Connectivity Challenges**: Considering that fish farms are often in remote areas with poor cellular signal, the architecture should address connectivity challenges including offline support.
- **Multi Geo Location Support** : The architecture should support the devices deployed across various farms present in multiple geographical location.
- **Extensibility for Future Enhancements**: The architecture should be flexible and extensible to accommodate future enhancements, such as adding capabilities for other livestock or aquarium management.

## ASSUMPTIONS

- **Nature of Devices**: The devices deployed in farms are categorized as IoT devices.
- **Device Capabilities**: Devices can store essential identifiers like enclosure, farm, and geographical location.
- **Gateway Installation**: A physical gateway device is installed at each farm location to support offline operations.
- **Preference for Azure Services**: The organization prefers using Microsoft Azure services for affordability and effectiveness.
- **Utilization of Azure Managed Services**: The organization is inclined to use various managed services offered by Azure wherever suitable.
- **Self-Storage Capability**: Devices, along with gateways, possess self-storage capacity. (Using  lightweight, embedded databases (e.g., SQLite, Berkeley DB) that can run directly on the IoT device to store data temporarily until an internet connection is available)
- **Gateway Communication**: Gateways can communicate with devices using wired or wireless protocols (e.g., Bluetooth) without reliance on internet connectivity

## MAJOR ABILITIES SUPPORTED IN THE ARCHITECTURE

### Scalability

**Azure Kubernetes Service (AKS):** Utilize AKS to deploy and manage containerized applications, providing automatic scaling capabilities based on resource utilization.

**Azure Cosmos DB**: Use Cosmos DB as a globally distributed, multi-model database service that can scale horizontally to handle increasing data volumes and user loads.

### Reliability

**Azure Availability Zones**: Deploy resources across multiple availability zones within a region to ensure high availability and fault tolerance.

### Security

- **Managed security service**
- Access control via Network Security policies and firewall
- **Data encryption in transit and at rest**
- **Azure Active Directory (AAD)**: Leverage AAD for identity and access management, enabling centralized authentication, authorization, and user management.

### Performance

- **Managed database service** (Cosmos DB)
- Auto scale capability
- **Azure Cache for Redis**: Use Redis Cache to improve application performance by caching frequently accessed data and reducing latency for read-heavy workloads

### Offline Capability

1. **Azure IoT Edge**: Utilize IoT Edge to enable local data processing and analytics on gateway devices, allowing the system to continue operating offline and synchronize data with the cloud once connectivity is restored

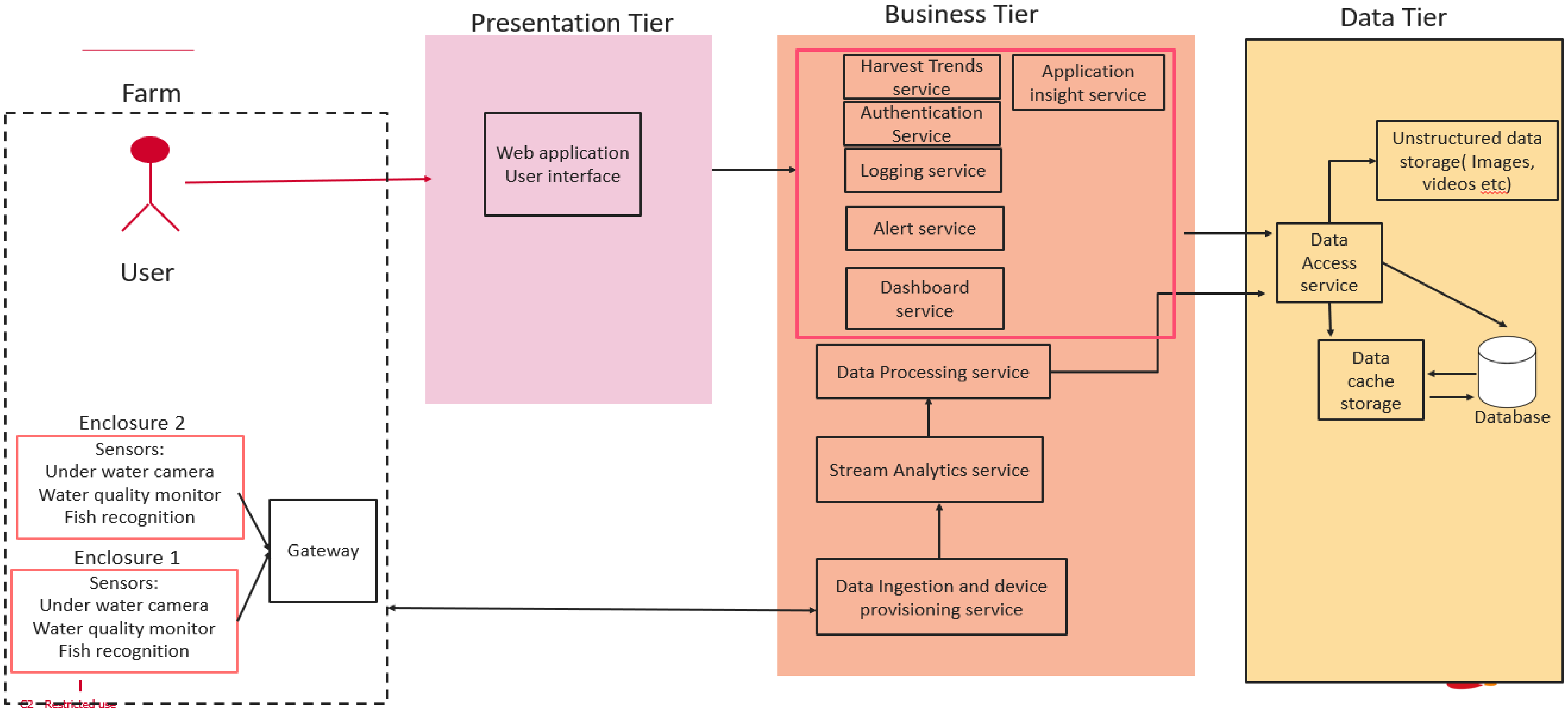
### Monitoring And Logging

**Azure application insights** : Azure Application Insights is a service that helps you monitor and diagnose the performance and usage of your web applications

## ARCHITECTURE DIAGRAMS COVERED IN THIS SLIDES

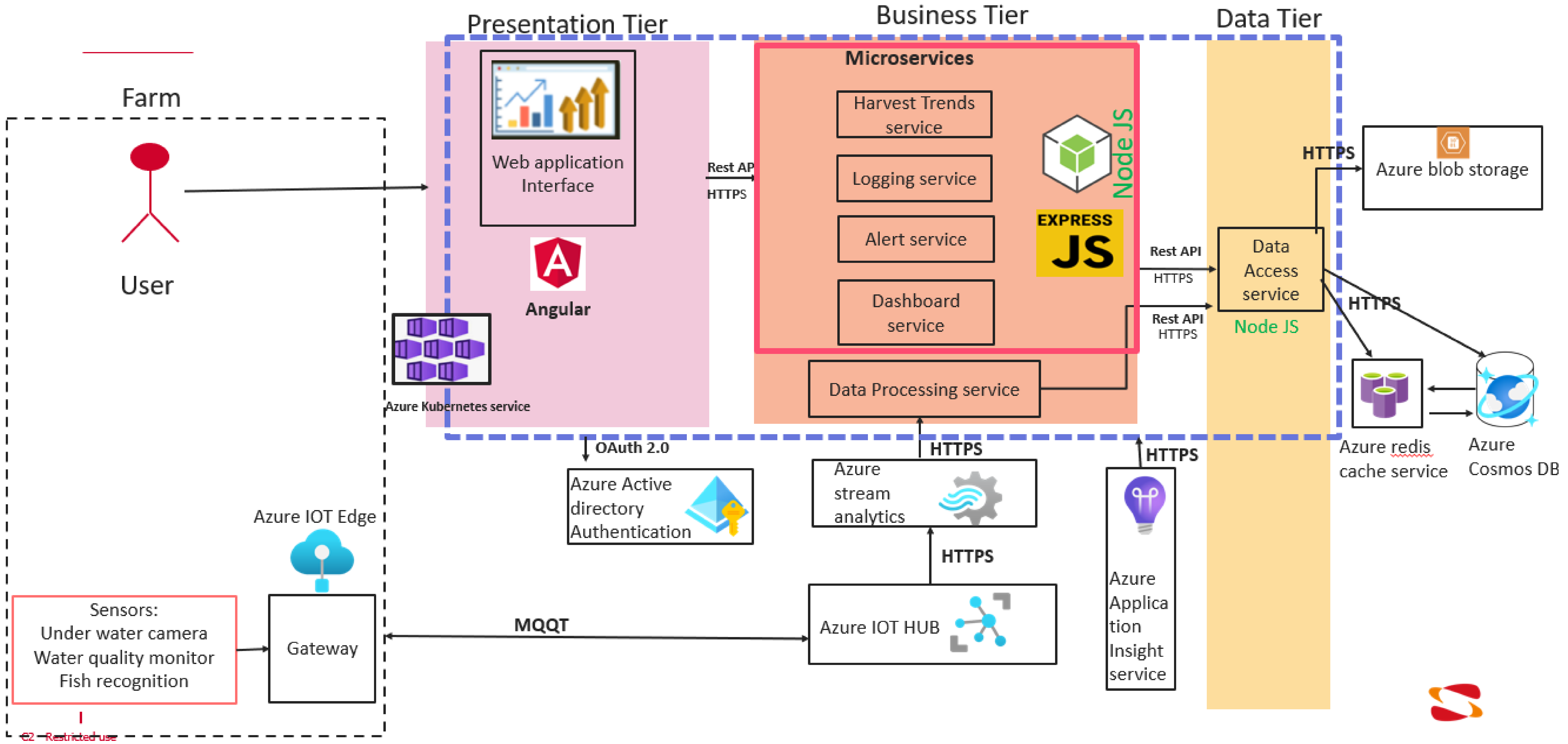
- **Functional Architecture:**
- Illustrates the functional components and their interactions within the system.
- Provides a high-level overview of the system's functionality without diving into technical details.
- Focuses on the logical organization of components and their relationships.
- Helps stakeholders understand the system's behavior and flow of data or processes.
- **Application Architecture:**
- Builds upon the functional architecture by incorporating the technical stack and communication protocols.
- Details the specific technologies, tools, and frameworks used in the system.
- Describes how the functional components are implemented using the chosen technical stack.
- Specifies the communication protocols between different components and services.
- **Deployment Architecture / Technical Architecture:**
- Specifies how the system is deployed across various environments.
- Describes the infrastructure components such as servers, containers, and networking configurations.
- Provides insights into scalability, reliability, and resilience aspects of the system's deployment.

# FUNCTIONAL ARCHITECTURE /COMPONENT ARCHITECTURE



Presentation Tier
Business Tier
Data Tier
Farm
User
Web application User interface
Harvest Trends service
Application insight service
Authentication Service
Logging service
Alert service
Dashboard service
Data Processing service
Stream Analytics service
Data Ingestion and device provisioning service
Unstructured data storage( Images, videos etc)
Data Access service
Data cache storage
Database
Enclosure 2
Sensors: Under water camera Water quality monitor Fish recognition
Enclosure 1
Sensors: Under water camera Water quality monitor Fish recognition
Gateway
~~C2 - Restricted use~~

## APPLICATION ARCHITECTURE



## TECHNICAL STACK

**Azure IoT Edge on Gateway:**

Justification: Azure IoT Edge provides edge computing capabilities, allowing for data preprocessing and filtering at the gateway level. This helps reduce bandwidth usage and latency by processing data closer to the source, enhancing the overall efficiency of the IoT solution.

**Azure IoT Hub:**

Justification: Azure IoT Hub serves as the central message hub for bi-directional communication between IoT devices and the cloud. It provides scalable device management, secure communication, and seamless integration with other Azure services, making it an ideal choice for managing and monitoring IoT device connectivity.

**Azure Stream Analytics Service (SAS):**

Justification: Azure Stream Analytics enables real-time data processing and analytics on streaming data from IoT devices. It offers a SQL-like query language for easy data transformation and analysis, allowing for the extraction of valuable insights from the continuous stream of data generated by the FishWatch system.

**Node.js with Express Framework:**

Justification: Node.js is a lightweight, event-driven JavaScript runtime, well-suited for building scalable, real-time applications. The Express framework provides a minimalist, yet powerful web application framework for Node.js, enabling rapid development of RESTful APIs and web services required for the microservices architecture of the FishWatch system

## TECHNICAL STACK ( CONTD.)

**Microservices (Data Processing, Dashboard, Alert, Logging, Harvest Trend, Data Access)**:

Justification: Microservices architecture promotes modularity, scalability, and flexibility by breaking down the application into smaller, independently deployable services. Each microservice focuses on a specific business functionality, allowing for better maintainability, fault isolation, and scalability of the Fish Watch system.

**Azure Redis Cache**:

Justification: Azure Redis Cache provides an in-memory data store for caching frequently accessed data, improving the performance and scalability of the Fish Watch system. By caching data closer to the application, Redis Cache reduces latency and offloads the load on backend data stores, enhancing overall system responsiveness.

**Azure Blob Storage**:

Justification: Azure Blob Storage offers scalable, cost-effective storage for unstructured data such as images, videos, and documents generated by the Fish Watch system. It provides high availability, durability, and seamless integration with other Azure services, making it an ideal choice for storing and managing large volumes of data.

**Azure Cosmos DB**:

Justification: Azure Cosmos DB is a globally distributed, multi-model database service designed for building highly responsive and scalable applications. It offers flexible data models, automatic scaling, and low-latency access to data, making it suitable for storing and querying diverse data types generated by the Fish Watch system while ensuring high availability and fault tolerance.

## TECHNICAL STACK (CONTD..)

**Angular for front end development**

Offers a robust framework for building dynamic, single-page applications.

Provides a rich set of features, including data binding, dependency injection, and routing.

Enables efficient development, testing, and maintenance of complex user interfaces.

Ensures a seamless user experience with responsive design and cross-platform compatibility

**Azure Kubernetes Service (AKS)**

selected for orchestration: Simplifies container management and scaling.

Ensures high availability, fault tolerance, and resource optimization

**Azure Application Insights**

It is utilized in the architecture to enable comprehensive monitoring and diagnostics of microservices. It offers real-time insights into application performance, proactive issue detection, customizable alerts, and powerful diagnostic tools, empowering continuous optimization and ensuring high reliability of the application.

## BUSINESS CONTINUITY PLAN AND AVAILABILITY

The below are built-in features that are typically available by default and do not incur additional costs in the mentioned Azure managed services which support business continuity:

**Utilizing Inbuilt Features of Managed Services:**

- Managed Database Service ( Azure Cosmos DB): Backup, replication, and failover mechanisms ensure data redundancy and automatic failover.

- Managed Kubernetes Service: Replication and failover capabilities ensure uninterrupted application availability.

**- Ensuring Uninterrupted Business Operations:**

- Regular backups and restore mechanisms for application and databases.

- Replication and failover minimize downtime and ensure continuous service delivery.

**- Business Continuity in Incidents:**

- Swift disaster recovery through replica promotion or backup restoration.

- Automatic failover and replication minimize downtime, ensuring business continuity.

## MONITORING AND LOGGING WITH AZURE APPLICATION INSIGHTS

Azure Application Insights offers comprehensive capabilities for real-time performance monitoring, error tracking, usage analytics, and custom logging. Integrated with application microservices, it enables proactive issue resolution and optimization, ultimately delivering exceptional user experiences

**Real-time Performance Monitoring**: Track response times, request rates, and resource utilization to optimize application performance.

**Error Tracking**: Automatically detect and log exceptions and errors, enabling proactive issue identification and resolution.

**Usage Analytics**: Collect telemetry data on user interactions to understand user behavior and prioritize feature development.

**Custom Logging**: Integrate custom logging and tracing for capturing tailored diagnostic information.

**Alerting and Notifications**: Set up customizable alerts based on predefined metrics thresholds for proactive monitoring and timely response.

**Integration with Azure Monitor**: Seamlessly integrate with Azure Monitor for centralized monitoring and correlation of telemetry data across services.

**Microservice Integration**: Integrate Application Insights with application microservices for deep insights into service health and performance, facilitating proactive optimization and exceptional user experiences

## SECURITY MEASURES

Security Measures Overview:

**1. Incoming Traffic:**

Azure IOT hub being the entry point for device traffic , below security measures are implemented :

Azure IoT Hub provides built-in security features to protect incoming traffic, including:

**Device Authentication**: IoT devices must authenticate with Azure IoT Hub using security tokens or X.509 certificates, ensuring that only authorized devices can connect to the hub.

**Device-to-Cloud Communication Encryption**: Data transmitted from devices to Azure IoT Hub is encrypted using industry-standard encryption protocols (TLS/SSL), preventing eavesdropping and data interception during transmission.

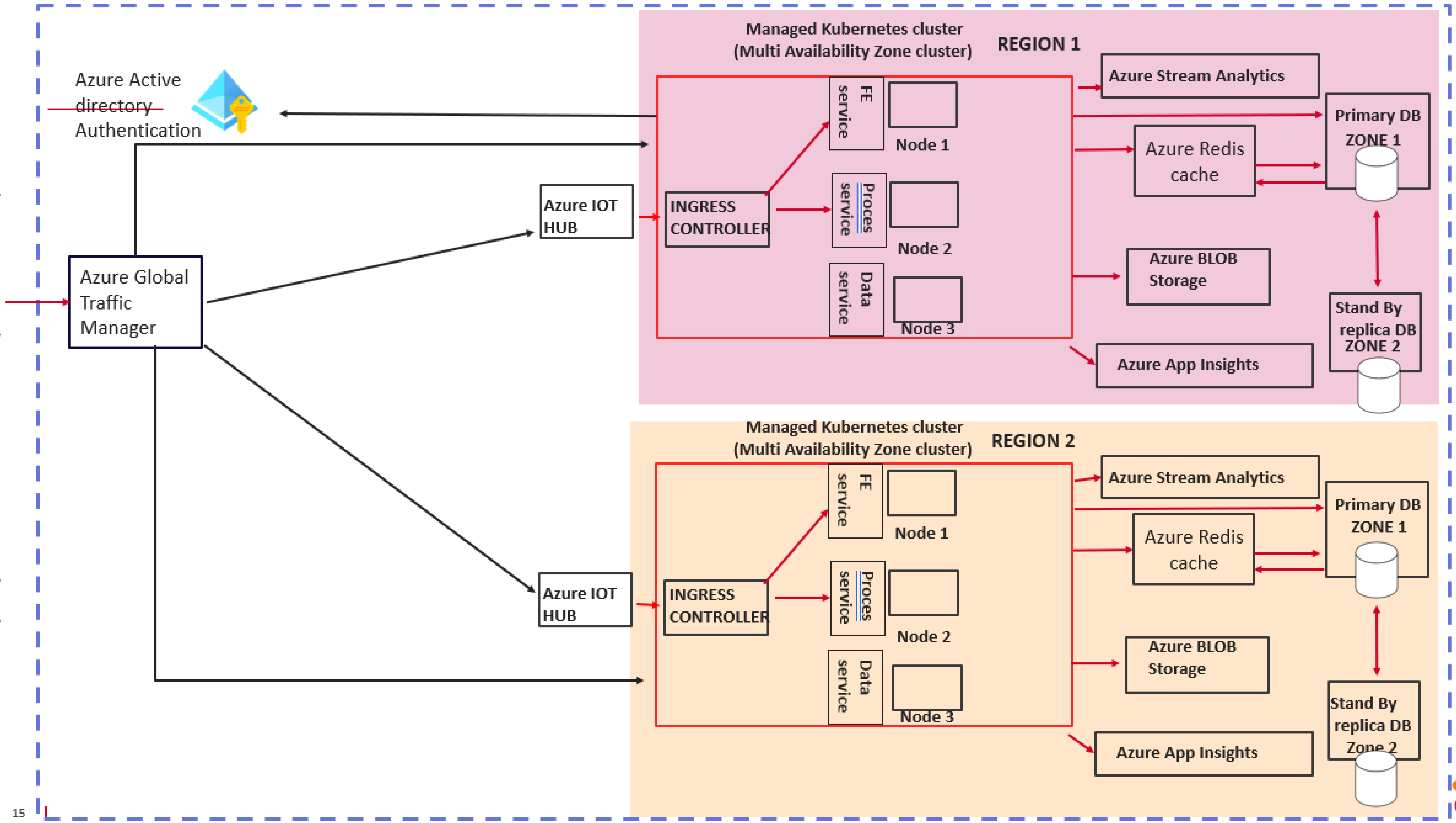
**2. Component Security:**

- Managed Kubernetes Service: Secure configurations and access controls.
- Managed Database Service: Encryption, access control, and network or firewall policies.
- Managed Blob Storage: Data encryption, access controls and network or firewall policies.

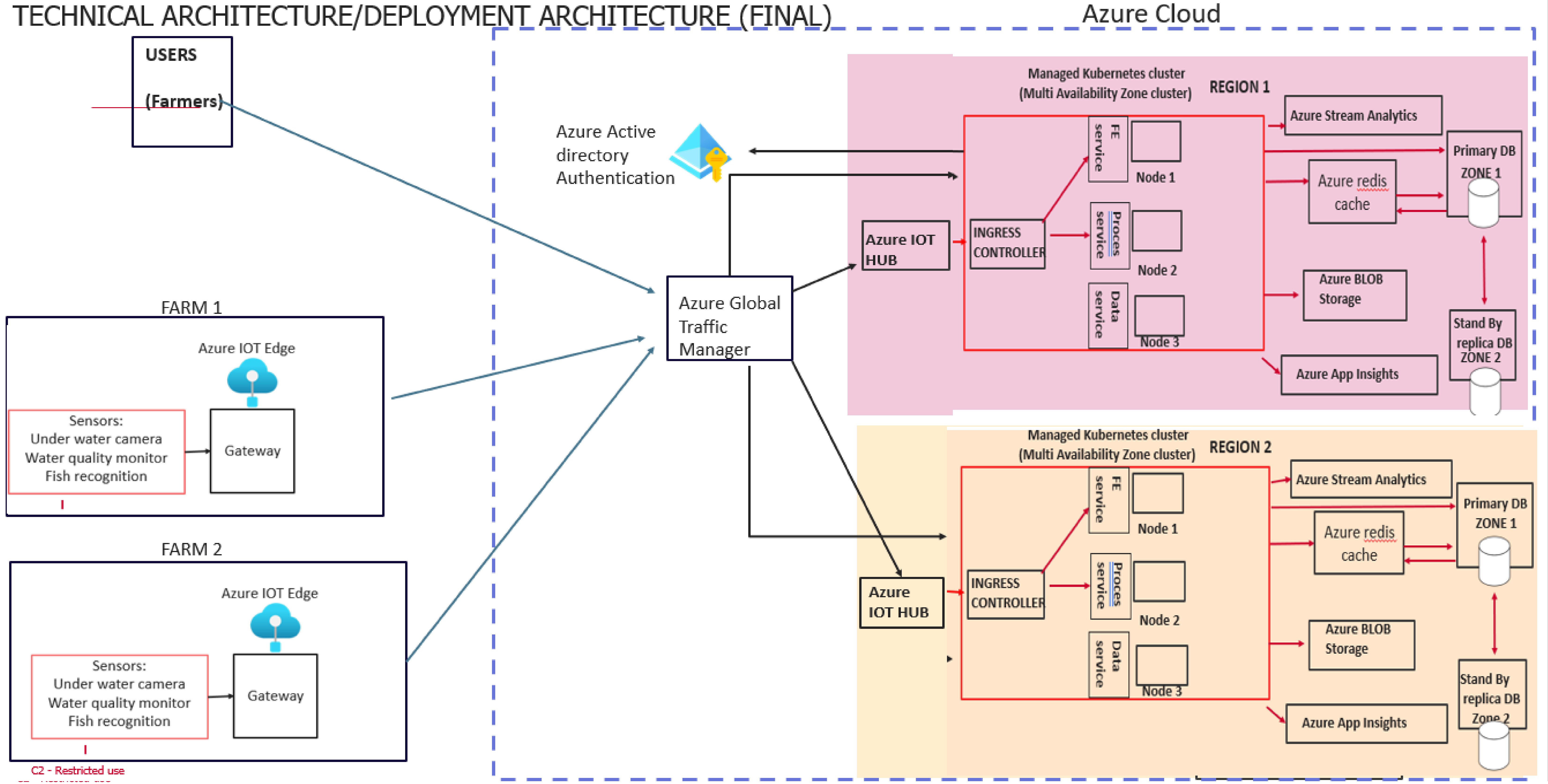
**3. Communication between various services** - Secure Communication Channels: Encrypted data in transit and at rest , access control using network security policies , firewall

**4.Authentication and Authorization**: Using Azure Active directory for authentication , following Industry standard protocol OAuth 2.0

Deployment Architecture (Zoom view)



## TECHNICAL ARCHITECTURE/DEPLOYMENT ARCHITECTURE (FINAL)



]

## CONCLUSION

- - Robust and scalable architecture
- - Utilization of managed services for scalability and optimal performance
- - Security measures with managed security services, firewalls, and secure communication channels
- - Integration of Azure ADD for secure authentication and authorization
- - Reliable data management with managed database and blob storage services
- - Business continuity ensured through inbuilt features of managed services for backup, replication, and failover
- - Effective monitoring and logging with a managed service for proactive issue detection
- - Solid foundation for future growth and success of Fish watch
- Offline connectivity support using gateway and IOT edge