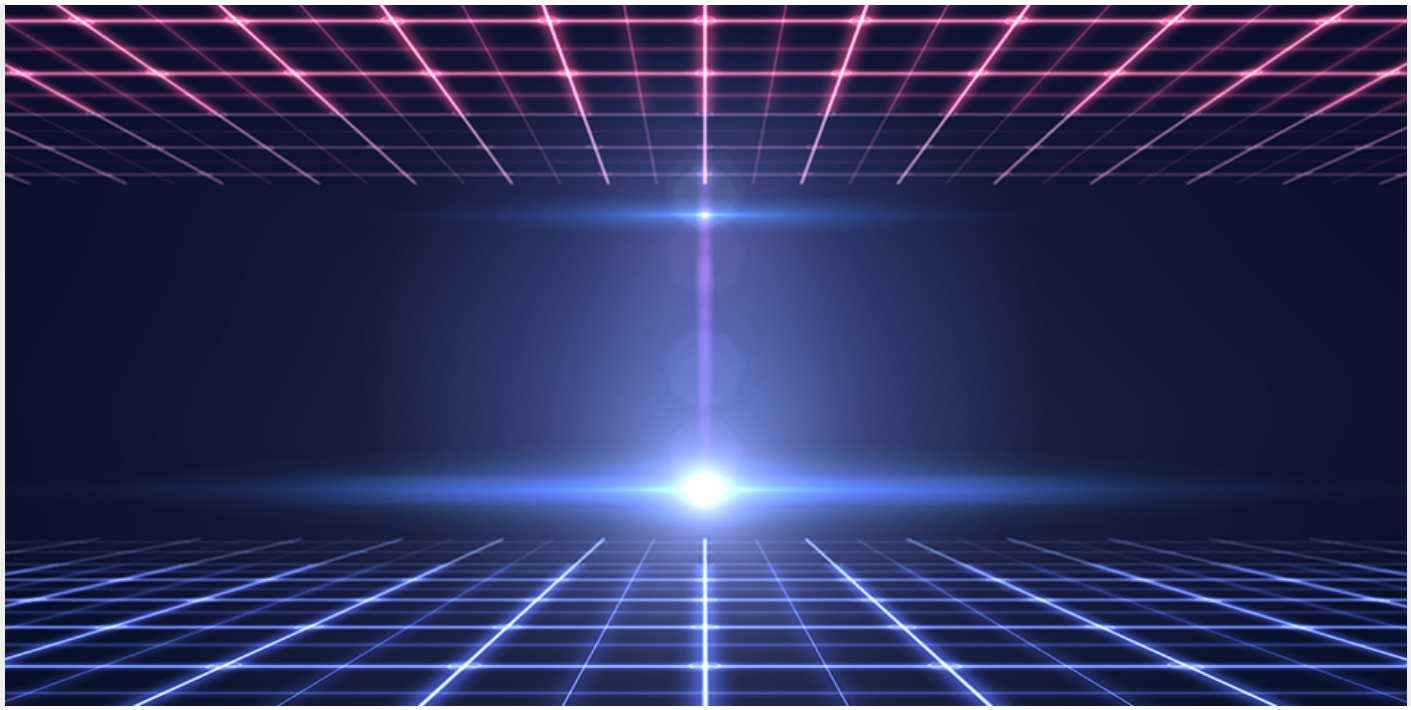
# EDGE COMPUTING FOR REAL-TIME ANOMALY DETECTION OF IOT DATA



An anomaly, defined as any change in usual behavior, can provide early warning of a problem. For example, anomalies in an Internet of Things (IoT) sensor's timeseries data can indicate a failure in a manufacturing unit. However, detecting anomalies in real time is becoming more and more challenging. Traditional anomaly detection methods such as creating visualizations and dashboards can't keep pace with the extreme data volumes and velocity of today's IoT. In this blog, we'll look at other methods.

**Static threshold-based alerts** can be configured to detect anomalies. As a more automated approach, this is an improvement over traditional methods, but it still has several disadvantages, such as:

- Admins need domain knowledge to set the right thresholds
- Thresholds are not adaptive to data changes
- Static alerts are difficult to maintain in ever-changing environments

**Supervised or unsupervised machine learning (ML)-based methods** bring intelligence to automated anomaly detection. By continuously learning timeseries data behavior, the system becomes more adaptive to data changes and can better handle changing environments.

However, the volume of data generated by IoT devices is increasing with time, driving related increases in network bandwidth, storage volume, and compute. As a result, the use of centralized cloud resources for real-time IoT data processing is becoming more and more expensive and lengthening the latency in data processing.

**Edge computing** provides a more effective way to leverage ML-based anomaly detection. By shifting critical data processing workloads closer to the data source (IoT devices), you can reduce workloads on the cloud; ensure zero-latency data processing; improve response time; and decrease network load and cloud costs.

Edge computing can be also used to monitor machine health in real time to detect anomalies that might indicate a failure in a system. Zero-latency data processing makes it possible to report major incidents in near-real time and prevent system failures.

# EDGE COMPUTING ARCHITECTURE

There are several types of edge computing architectural models used in the industry, including 100 percent edge computing and hybrid models that combine edge and cloud computing. A generic architecture of this type is shown in Figure 1.

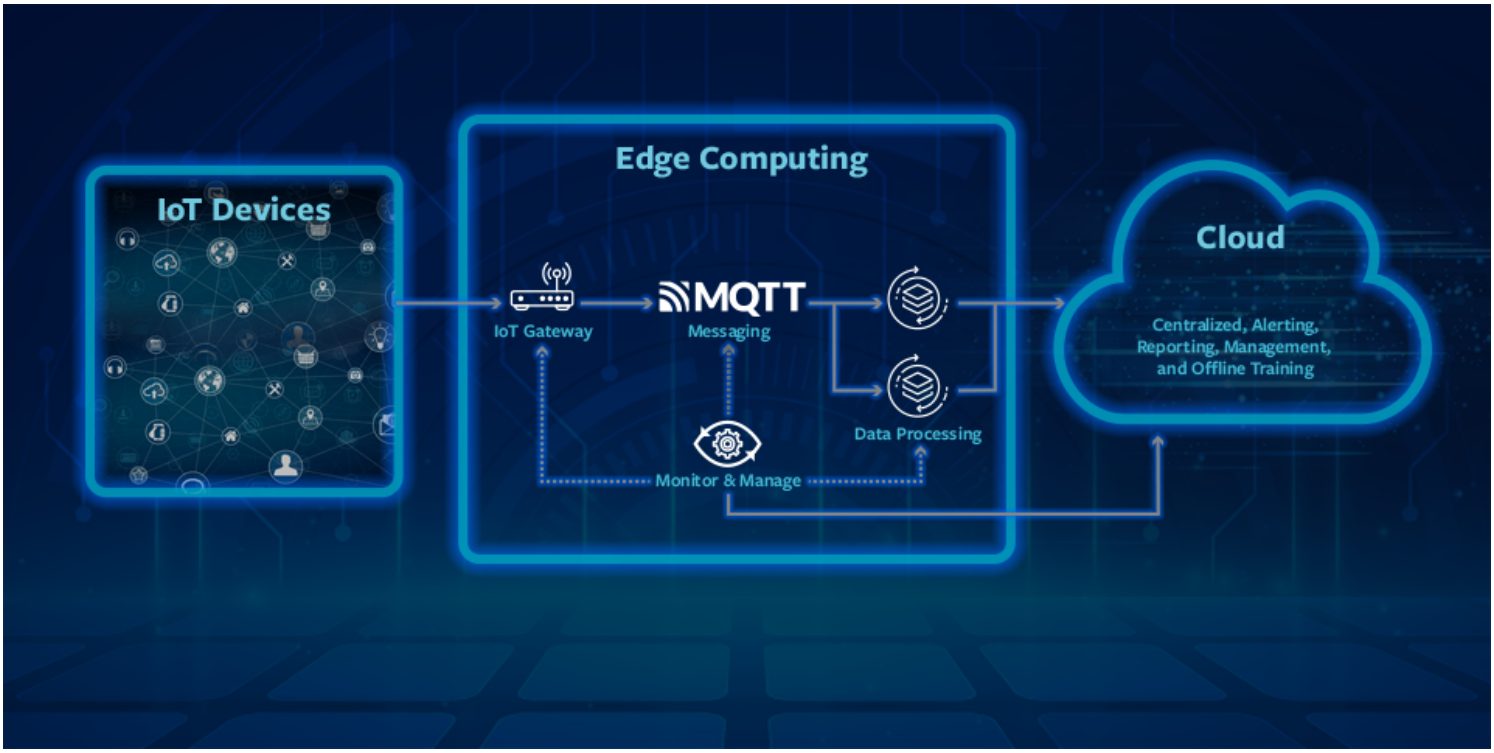



*Figure 1: A generic architecture including edge computing and cloud computing*

In this model, the **IoT Gateway** collects and receives data from **IoT Devices** using various IoT data protocols and pushes data to a **Messaging** layer. **Data Processing** components at the edge pull data from the **Messaging** layer, process it, and send insights to the **Cloud** for centralized alerting and reporting. A **Monitor & Manage** element reports operational issues to the **Cloud** and ensures that components are in synch with the configuration data centrally managed from the **Cloud**.

# EDGE COMPUTING DEPLOYMENT

The deployment of edge computing components, which can be based on scalability requirements, can be deployed together in a single board computer like Raspberry Pi, or across multiple devices in the same data center for greater scalability.

# REAL-TIME IOT DATA ANOMALY DETECTION

Figure 2 shows an architecture to implement edge computing for anomaly detection. This model is similar to the generic architecture described in Figure 1 with added intelligence.

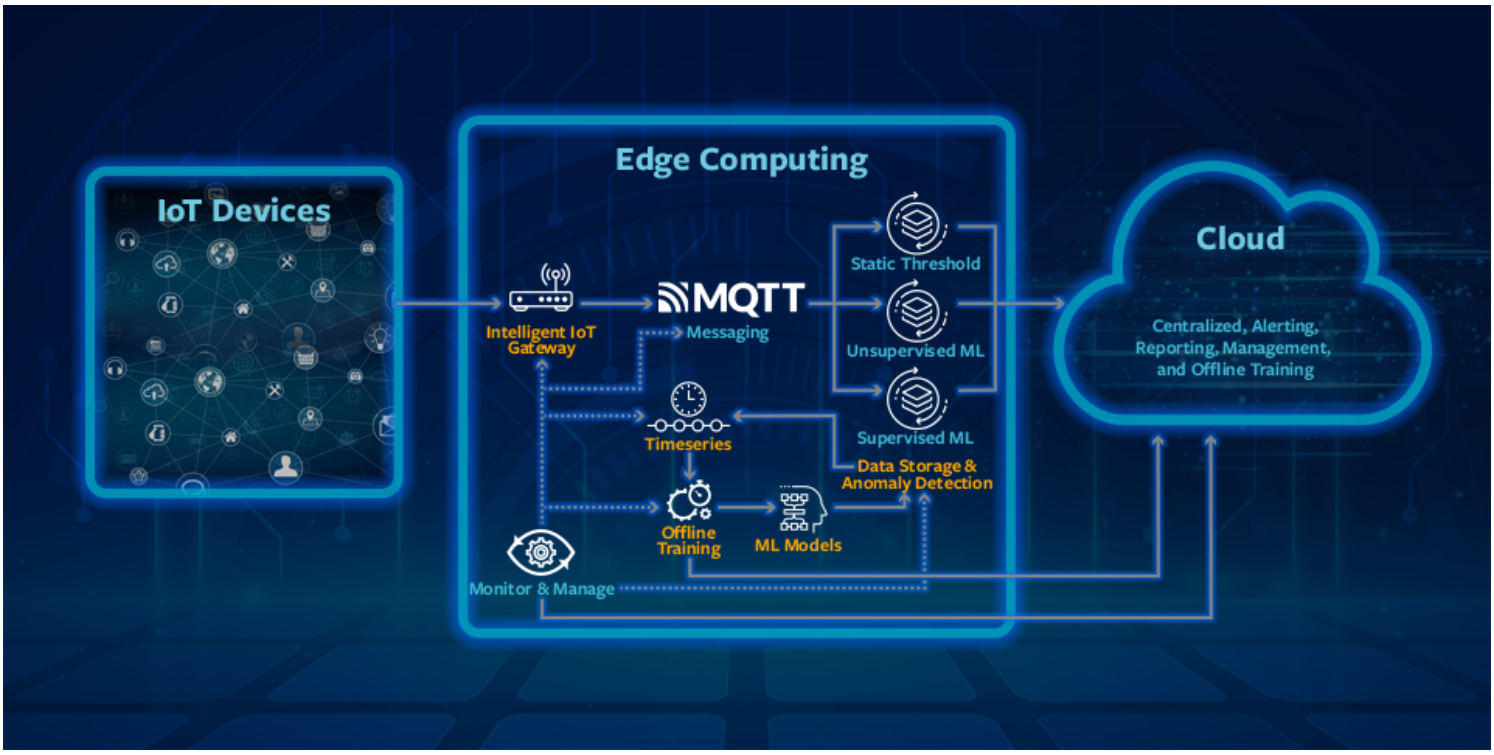



*Figure 2: Edge computing for real-time anomaly detection*

In this model, an **Intelligent IoT Gateway** acts as a timeseries data router. The **Messaging** layer has separate queues for each anomaly detection method. Based on the anomaly detection methods being applied to specific timeseries, the **Intelligent IoT Gateway** makes routing decisions and puts data into the respective queues. Dynamic routing decisions are driven by the configuration. Configuration changes are made at the **Cloud** and applied on the **Intelligent IoT Gateway** by the **Monitor & Manage** element.

**Anomaly Detection** processors fetch data from the respective queues, detect anomalies, and report them to the **Cloud**. These processors also store timeseries into a **Timeseries DB**.

An **Offline Training** scheduled job periodically fetches bulk data from the timeseries DB, compresses data, and sends it to the **Cloud** to train ML models for supervised anomaly detection. The job also fetches ML models from the **Cloud** and stores them in an **ML Models DB**. A **Supervised ML Anomaly Detector** fetches ML models from the **ML Models DB**.

The **Monitor & Manage** element reports operational issues to the **Cloud** and ensures that components are in synch with the configuration data centrally managed from the **Cloud**.

# CONCLUSION

The data volumes and velocity of a modern IoT implementation call for scalable, efficient, real-time anomaly detection. Edge computing makes it possible to leverage machine learning for IoT anomaly detection while avoiding high cloud costs and processing latency. By using the architecture described above, you can detect and resolve IoT failures quickly to ensure optimal service for your organization.