A Knowledge Graph Driven Approach for Edge Analytics

Authors:

Narendra Anand, Ph.D.
Accenture Labs
narendra.anand@accenture.com

Anuraag Chintalapally
Accenture Labs
a.chintalapally@accenture.com

Colin Puri, M.S.
Accenture Labs
colin.puri@accenture.com

Srinivas Yelisetty, M.S.
Accenture Labs
srinivas.yelisetty@accenture.com

Teresa Tung, Ph.D.
Accenture Labs
teresa.tung@accenture.com

Michael Giba
Accenture Labs
michael.t.giba@accenture.com

INTRODUCTION

The rapidly expanding Internet of Things (IoT) market introduces new challenges surrounding the processing and storage of large quantities of data produced by edge analytics systems and the management of applications deployed on varied array of devices that comprise these systems. To address these challenges, we present a novel, operationalized approach to the deployment of a practical, edge analytics framework that accomplishes two key goals: (1) Enable the simplified integration of heterogeneous hardware and software resources with existing applications (or models) and (2) Utilizes a knowledge graph to capture domain knowledge for reusability, relationship inferencing, maintainability, and data stream communication normalization for infrastructure instantiation. Inspired by the Web of Things (WoT), the approach has a "network layer" and "accessibility layer"; however, expands upon it conceptually to provide an abstraction for expertise modularity. Through addressing these goals the framework streamlines a division of roles, operations, and deployments while minimizing associated maintenance of IoT ecosystems. This is enabled by bootstrapping domain knowledge via a graph and instantiating device-specific information through an onboarding interface.

While companies without existing IoT deployments can deploy vertically-integrated edge analytics hardware and software solutions unencumbered by legacy issues, many companies have already invested in IoT infrastructure and software development resources. The result of which is a heterogeneous, piecemeal, or disparate deployments coupled with legacy applications where replacement of existing infrastructure and system refactoring is not economically nor technically feasible. To overcome the associated challenges of legacy and heterogeneous ecosystems, we utilize containerization techniques to integrate existing applications and enable the deployment of applications to any node within an existing heterogeneous deployment. The core of our framework tackles the challenges with a knowledge graph in the cloud, an orchestration layer that enables communications and container instances, and a distributed messaging backbone for resiliency and filtering.

In this discussion, we present the components and benefits of our system through two in-flight use cases (a large oil and gas client, and a large construction and mining client) along with an explanation of the problem space, framework walkthrough, and brief comparison to other approaches throughout.

MOTIVATION

By 2020, the Internet of Things (IoT) market will reach $267 billion with 50% of spending driven by industrial and commercial applications[1]. As IoT deployments become more numerous, scalable and easily maintainable, architectural solutions are paramount for meeting and sustaining the demand of large, expanding, and elastic device networks.

Current IoT deployments are extensions of the edge-to-cloud paradigm where control and analysis of signal data from edge devices is centralized with elastic compute in mind. While this methodology is the simplest in terms of initial deployment and scalable from a compute perspective, cloud management does not imply scalable maintainability with respect to a heterogeneous device environment with device types where there is a lack of protocol or implementation rigor. While vertically integrated solutions exist[2] [3] that aid in this type of deployment, they require a complete migration of existing applications into the new ecosystem (*e.g.,* hardware, OS, language, cloud tools, etc.). Additionally, the edge-to-cloud paradigm assumes stable, low latency and high bandwidth backhaul (uplink and downlink) connectivity. However, IoT deployments in remote locations may suffer from limited backhaul connectivity and thus render the timely processing and reaction to data impractical.

Additionally, knowledge of the edge networks relies on key personnel with intimate domain knowledge and site information details, namely: device expert, application expert and the field engineer. Collectively these engineers hold the domain knowledge required to start and fluidly maintain the network of edge devices at any given location.

As infrastructure communication is orchestrated and edge devices are brought online there is subtle knowledge of devices that must be documented to onboard new and similar devices. IoT applications are moving beyond simple sensor data storage and towards analytics applications where novel architectures are required to enable the horizontal scalability of device additions, reduce instantiation time, boot strap domain knowledge, enable system reuse and migrate compute and intelligence toward the edge. When moving from domain to domain, isolation of knowledge kept with the domain experts becomes a limiting factor of rapid deployment when working with a mature portfolio of edge framework configurations (varying hardware, varying device driver versions, language attributes, communication protocols, etc.).

To avoid refactoring, increased man-hours and to promote rapid onboarding and enable knowledge retention, we developed a solution that ensures that the instantiation

[1] L. Columbus, "Internet Of Things Market To Reach $267B By 2020," Forbes, January 2017. [Online]. Available: http://www.forbes.com/sites/louiscolumbus/2017/01/29/internet-of-things-market-to-reach-267b-by-2020/#2343106e609b

[2] "GE Predix IoT Platform," 2017. [Online]. Available: https://www.ge.com/digital/predix

[3] "Thingworkx IoT Platform," 2017. [Online]. Available: https://www.thingworx.com/

of the first edge device is as simple as the instantiation and addition of the 1000th edge device. We accomplish this by employing a unique knowledge graph approach, standardized communication enabled with a distributed message queue and containerization for consistent compute and driver deployment.

PROBLEM SPACE

Many of the challenges facing the design of edge analytics frameworks are well understood[4]. However, our goal is not to design a vertically integrated edge analytics solution anew. Instead, we aim to create an edge analytics framework that tackles the following challenges: (1) Seamless and rapid integration with a client's existing heterogeneous environment (varied application portfolio and varied hardware infrastructure) and (2) Provide an abstraction layer that enables domain-specific experts to develop edge analytics components that are decoupled from and agnostic to the implementation details of the underlying connected components.

What follows is an explanation of a generalized use case, two in-flight domain specific use cases demonstrating the cross-domain applicability of our Edge Framework, their core and distilled challenges and resolutions.

Common Deployment Scenario

Consider the following common client site deployment scenario: (1) A device expert wants to onboard a new type of sensor for use in an edge analytics deployment, (2) An application expert wants to develop or refactor an analytics application that consumes the data generated by that sensor and (3) A field engineer will deploy several of these new sensors when steps (1) and (2) are complete.

An ad-hoc solution allows for deployment on heterogeneous hardware and software infrastructure. However, it requires all three roles in the scenario to have intimate knowledge of the entire edge framework and requires close coordination of their work. Tight integration and coupling of work efforts runs counter to a scalable or maintainable solution because it doesn't allow for separation of component design, development and workflow modularity. Additionally, tight coupling of work tasks may require numerous task handoffs for quality control or modifications that are difficult to streamline and minimize. Vertically-integrated solutions[5] attempt to address this problem by controlling every aspect of the edge framework from the hardware to the cloud. While an IoT framework may be partially decoupled, system engineers still require knowledge of and experience in that solution's complex development environment for each of the three components of the scenario.

[4] M. Patel and B. Naughton, "Mobile-edge computing introductory technical white paper," *White Paper, Mobile-edge Computing (MEC) industry initiative,* September 2014.

[5] “Foghorn Lightning," 2017. [Online]. Available: https://foghorn.io

Additionally, in a situation where the modification or removal of an edge device is required, that change must be communicated throughout the system due to the inability of a vertically integrated solution to handle disparate heterogeneous deployments because there isn't a filtering, normalization and routing, of messages based on ecosystem topology. Even robust horizontally-integrated systems such as the Cisco® IOx platform require some degree of hardware standardization (networking infrastructure). The knowledge graph driven approach requires solely software level standardization, namely the ability to run containers.

Client Use Case Description

Following the generalized use case are two on-going specific use case examples, in the Oil and Gas (O&G) and Mining sectors, from clients for whom we are currently developing our edge analytics framework.

Large O&G Services Company

Consider the use case of a large multinational O&G field services provider with a complex organizational and operational structure. This client has been digitizing and networking their industrial assets in on-shore and off-shore oil/gas fields for decades and has garnered a mature understanding of and experience with Industrial IoT technologies and tools. In the process of doing so the client has accumulated technological "debt" over the years ranging from proprietary solutions, heterogeneous approaches and dated hardware involving several business units in pursuit of gathering data and running analytics on the edge. Thus, the client is dependent on and manages legacy technology with redundancies in hardware capabilities and software functionality across the stack. This leads to an opportunity to optimize on operational costs, increase efficiency by standardizing methodologies and adopt a modernized enterprise-wide Edge Analytics platform.

Challenges

The Brownfield nature of this client's operational environment means any edge framework must be flexible enough to support existing operations and at the same time provide a seamless path forward toward ecosystem modernization, technology migration and device deployment. The diversity of needs for this organization means it must support multiple edge hardware devices, operating systems (OS's), variety of data-processing and analytical requirements, data storage and data format needs.

In-flight Implementation Requirements

The O&G client stipulated the following key solution requirements:

1) Heterogeneous Software (SW) Support: support for applications capabilities ranging from Complex Event Processing (CEP) and rules engine for data processing, as well as employment of Artificial Intelligence (AI) open source library frameworks to reduce licensing costs.
2) Extensibility and Agnostic Support: expansion capability for vendor and other 3rd party application integration, i.e., support ecosystem of players

3) Deployment Modularity: support for multiple operating systems (e.g., Windows and Linux)
4) Messaging Support: support data persistence, replication, and querying (with support of time series databases e.g., Cassandra, as well as document stores, e.g., MongoDB®)
5) Connectivity Robustness: robust to signal noise and data connectivity limitations
6) Heterogeneous Hardware (HW) Support: support multiple device versions and deployments

One of the more crucial use case requirements is the robustness regarding adverse site location issues such as lack of connectivity. This client has several locations where broadband or steady network connectivity is a challenge and data connectivity must fall back upon cellular or satellite coverage at significant costs. That is, high bandwidth streams may be cost prohibitive to maintain beyond a few Megabits/second. Therefore, data that is originally of high value must be filtered and down-sampled to lower rates when possible and high-fidelity data streams must be offloaded to physical storage and transported (e.g., via helicopter) from the remote locations. Thus, some analytics are not streamed dynamically and require post-processing.

Additionally, the client currently has ongoing requirements that the edge framework must support; a wide range of devices including laptops, edge gateways and mini server racks. These requirements are for providing three tiers of compute capacity (processing, memory and storage) and mobility in field deployments and maintenance.

Large Industrial Construction and Mining Manufacturing Company

The second use case is that of a large construction equipment manufacturer, building earth moving equipment, mining vehicles, etc., and has the need of interconnected and mobile sensors in large open areas. The client has been deploying machinery capable of sending communication and status feeds but in an ad hoc manner with many iterations on device firmware. To complicate the situation the environment in which deployment resides is prone to high mobility, equipment breakdown from heavy usage and noisy signal environment. As a result, the client works within a high-maintenance/high-touch environment prone to equipment failure where vehicles need constant attention. The client has been enabling equipment over time but is tasked with an extremely heterogeneous environment where the approach is of "if it isn't already broken, then don't fix it." While construction and mining equipment has been updated and maintained at a priority as it impacts the core business model, the needs of sensors and IoT devices is deprioritized in favor of reliability and stability of signal data.

Challenges

As in the previous use case description, this is an existing environment. This means that the overlay of any solution must be extensible to changing needs and support current operations. Furthermore, in this specific case the challenge is for a solution to be highly resilient and reduce the need for highly intermingled device expertise and support functional modularity. The challenge is to reduce the high-touch environment of sensors so that field operations can focus on their core business competencies, free from distractions of IoT deployment, maintenance and systems upgrading.

In-flight Implementation Requirements:

1) Decoupling of HW/SW stack: modularity of the hardware capabilities from specific applications and their versions
2) Simplified Deployment at Scale: capability to deploy thousands of sensors and provide a gated, pipelined and streamlined onboarding approach needing only minimal interaction of experts
3) Data message and protocol management: capability to queue messages, filter information and focus only on relevant information

This client has a critical need to reduce high maintenance touch from the device expert and modularize the deployment approach to avoid confusion when problems arise from installation or maintenance. When a device goes down, equipment is sidelined and revenue may be lost from unwarranted prolonged maintenance periods. There is also a deluge of events that stream from these devices that may or may not be important. Proper filtering and handling of said data is paramount to avoid operator attrition when parsing data feeds.

Edge Framework

Our edge framework tackles the challenges from the generalized use case and specific use cases as follows: (1) To decouple the different components of our solution, we employ a *knowledge graph* (graph that bridges concepts together) that serves as a centralized abstraction layer between each of the components that enable the domain-knowledge expert to focus on their relevant components while promoting knowledge retention for future infrastructure modifications and deployments. (2) To enable seamless scalability, simple maintainability and application reusability in heterogeneous infrastructure deployments, we use the knowledge graph to manage and employ a *containerized* approach that allows for development in whichever OS or language the developer chooses (or to reuse applications that were previously developed for a differing OS in another language). This is done by creating relationship links in a schema to act as a reference to various containers that can be deployed to an edge gateway instance given a set of features, capabilities, etc. We can then perform a search and infer which equipment can handle which containers after querying an actual graph instance.

By analyzing our client's deployment scenarios, we developed the following core framework to enable scalable and maintainable edge analytics that seamlessly integrate with heterogeneous hardware and software infrastructures:

Knowledge Graph

A centralized ontology containing schema information describing each edge device type, application type and the capabilities and data sources which connect them, lays the foundation for the knowledge graph. The instances associated with these application and device types are also represented in the graph to describe the current ecosystem within the represented network. The knowledge graph approach utilizes a resource description framework (RDF) graph paradigm to represent relationships and connections between two nodes in a knowledge graph instance as triples with a subject, predicate and object (e.g. [Node_1, "has Component", Node_2] with the relationship expressed as "type Node_1 has a component consisting of type Node_2"). The entire knowledge graph instance demonstrates a series of "triples" from the schema. A triple is a schema linkage between a pair of nodes where the predicate is effectively a type of edge and the subject and object pair are the equivalent of node types in the graph. This structural representation allows for searching and inferencing of information within a graph to understand and reason upon implications. Edge applications and the edge orchestrator, a layer in the architecture housing several services for facilitating communication back and forth with the up-stream knowledge graph, use this ontology to determine how to interact with each edge device and what supporting container infrastructure must be launched.

Messaging Backbone

An inter-component communication interface through which all edge devices and applications communicate. While a messaging protocol is chosen and fixed during framework onboarding, the schema of the data being transmitted is stored in the knowledge graph. This backbone combined with the information in the knowledge graph allows for the decoupling of the different components.

Containerization

All components on the edge appliance (from the edge applications to the orchestrator and the messaging backbone) are containerized[6]. This enables the deployment of the core components and the custom applications on any hardware infrastructure or OS. Additionally, it enables applications to be dynamically deployed and moved, and their resource utilization to be monitored and controlled.

Edge Framework Domain Knowledge Roles

Revisiting our previous enumerated scenarios, our edge analytics framework would enable each domain-knowledge expert to onboard, develop applications for, and deploy a new type of sensor in the roles as follows:

Device Expert

To onboard a new type of sensor, the device expert is responsible for updating the knowledge graph with the relevant information regarding a new sensor type, namely, the upstream data message schema streamed out of the sensor, the upstream control message schema generated by the sensor (e.g., heartbeat or status messages),

[6] "What is a Container", 2017. [Online]. Available: https://www.docker.com/what-container

the downstream control message schema to modify the operation of the sensor or actuator, relevant protocol information, and any run-time information and parameters (e.g., container references, etc.).

To enable the edge devices and analytics applications to communicate with one another, we deploy a messaging backbone within the edge appliance gateway. Since the messaging protocol employed by the sensor may not be the same as chosen for our messaging backbone, the device expert would need to create a translation container specific to a device type that would consume messages from the edge device's protocol and push them to the messaging backbone's protocol (and vice versa). While a container only handles a single edge device instance, a containerized approach allows for additional instances of the same container to be deployed whenever a new sensor is onboarded.

Application Expert

The application expert can query all necessary information for development from the knowledge graph. The analytics container developed by the application expert pulls the data and control message schema from the knowledge graph allowing the application to consume data from the edge device. This information, along with any known hardware dependencies, is documented in the knowledge graph when metadata for model sensor types is onboarded into the RDF ontology. The application container communicates with the edge device (or another application container) through the centralized messaging backbone to seamlessly produce and consume data.

Our framework's containerized approach allows the application expert to reuse or to develop an application in whatever environment they choose so long as containerization is supported for that operating system (currently supported by all major OS’s). For clients that seek to reuse applications from their existing application portfolio, the only required addition is communication with the messaging backbone. Additionally, because the production and consumption of data is abstracted through the message bus, this application container may be deployed on any edge appliance that has access or subscribes to the edge device's message topics. Messages are labeled as belonging to a topic category for subscription models and filtering of data messages, allowing for simplified vertical mobility of analytics containers as hardware resource constraints or uplink backhaul quality requires.

Field Engineer

The field engineer needs only to concentrate on the physical deployment of the sensor itself. Once the sensor is installed, it will begin to generate identifying heartbeat messages, which in turn will be consumed by an auto-discovery agent resulting in the registration of the new unique sensor instance into the knowledge graph. The registration of a new instance of an edge device will trigger the deployment of the container instances required to communicate with the edge device (e.g., message protocol translator) by our orchestrator. Every new sensor that is onboarded will inherit a triggered deployment of unique instances of the same required containers.

Solution Architecture

Knowledge Graph

as inherit properties from other device types.

The strength of the knowledge graph is in the capturing of relationships such as an "`instance_of`" link between a device type and a device instance when a field engineer

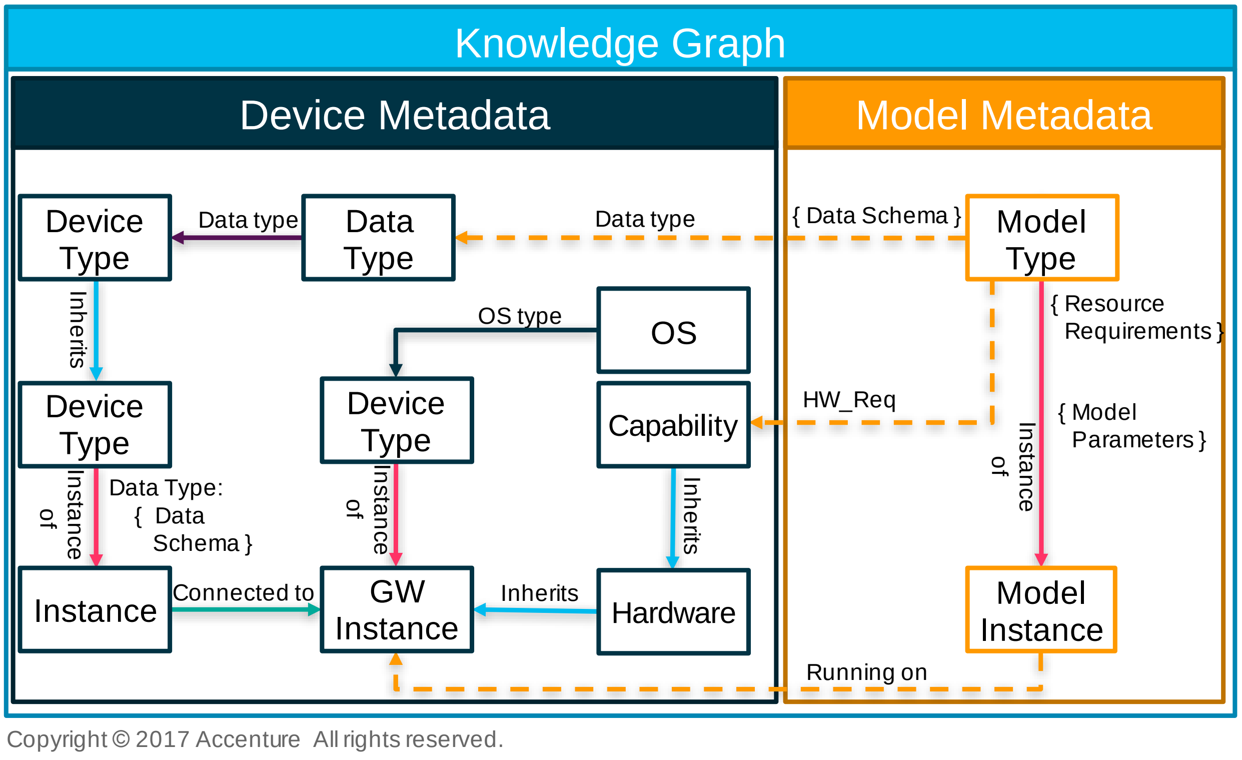

Figure 1: Knowledge Graph

The knowledge graph (simplification shown in Figure 1) portion is implemented as a graph in the cloud. It stores all relationships for devices, sensors, models, and data types. When onboarding a new device, the device engineer creates a new device type which in turn is associated to data schemas (documenting the semantics and format of the data produced by a device), associated hardware (e.g., GPU or other unique property specifying which types of applications can run on the device), as well

onboards a sensor. All capabilities of the instance are documented via the device type through the knowledge graph. However, instance-specific information, such as what sensors are connected, which applications are running, location, current resource usage, etc. are associated to that instance alone. The instance and device type relationships together enable deployment and management teams to query for subsets of devices based on capabilities and properties when determining which applications to deploy where. Furthermore, these relationships allow for more complex

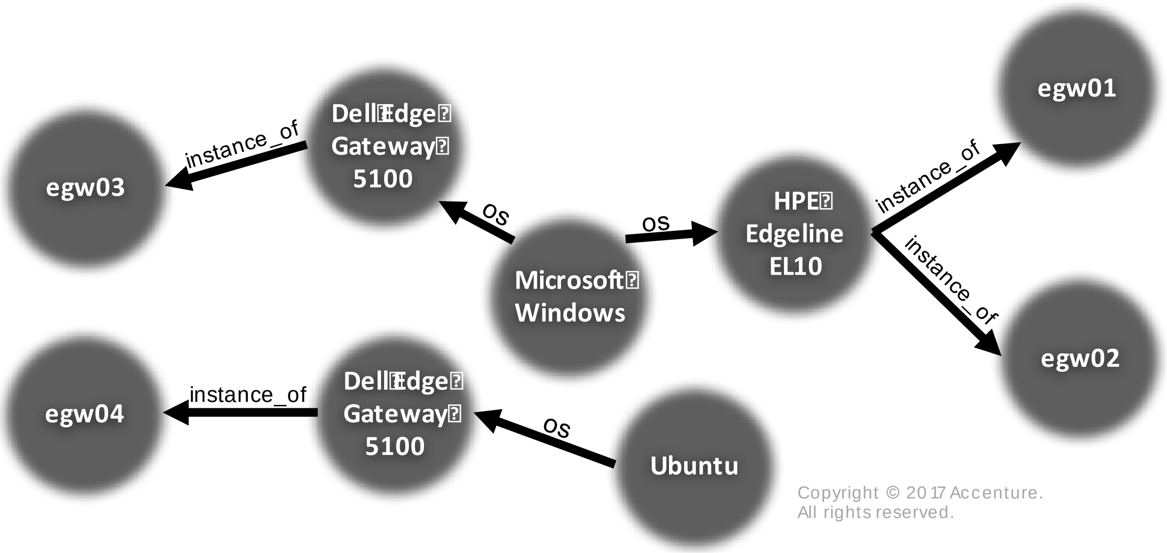

Figure 2: Small section of a client knowledge graph showing HW, OS, and physical gateway instance. Depicts four edge gateways (egw01-egw04) and their HW and OS types.

inferencing from both the device and application centric perspectives. The device domain expert can run queries to determine which devices can connect to one another as well as which theoretical network topology could host a given application.

The model types are the complementary component to the devices in the knowledge graph. When an application expert onboards a model type, the knowledge graph maintains the data schema as well as other dependencies which are required for that model to run. When a model is deployed to an instance, a "`model_instance`" relationship is automatically associated with the device instance to denote that a model is now running on a device (see Figure 2 for a small example segment taken from the O&G client graph as an illustrative example).

The last component within the knowledge graph is the data schema property reference. The data schema reference allows us to follow and extend upon the OPC UA format, a standardized machine to machine protocol for communication, in determining a standard for how data is streamed to devices. This allows the application expert to develop models without intimate knowledge of the devices or sensors and only need to know what data messaging protocol and handshaking is needed.

Our ontological approach to these components decouples the different domains and allow the experts to work independently removing any interaction roadblock that may hinder a deployment while the platform takes on the complex task of orchestration (which requires knowledge from all fields of expertise). Given a deployment template, the orchestration agents will query the knowledge graph to identify deployment targets based on the device capabilities, device attributes, and application dependencies.

Messaging Backbone and Application Services

The messaging backbone (shown in Figure 3) allows for applications and data streams to independently interact with each other without having to previously determine which applications and sensor streams are present on each device.

We utilize NSQ[7] as the default message broker because it is fault-tolerant, dynamically scalable, has simplified configuration, no single points of failure and is primarily in-memory (bounded volatile memory with disk overflow) so that the applications have a lower latency to access sensor data.

Integral to the operations of the messaging backbone are the protocol converters which support it. These translators are dynamically instantiated to convert device specific data protocols (MQTT, Serial, etc.) to the consuming NSQ format applications. These converters or containers reside on the edge appliance. They are automatically created by the orchestration layer when the knowledge graph reflects that a new device has been onboarded and connected to the platform.

The application services component provides a software development kit (SDK) for the application engineer to parameterize components of the application which vary across device instances. For example, the sensor stream topic names, IP addresses of the message brokers, etc. will all vary. The application engineer needs only to define a parameterized query and the application services will fill in the specifics (e.g., an accelerometer stream from the right front wheel well of a vehicle).

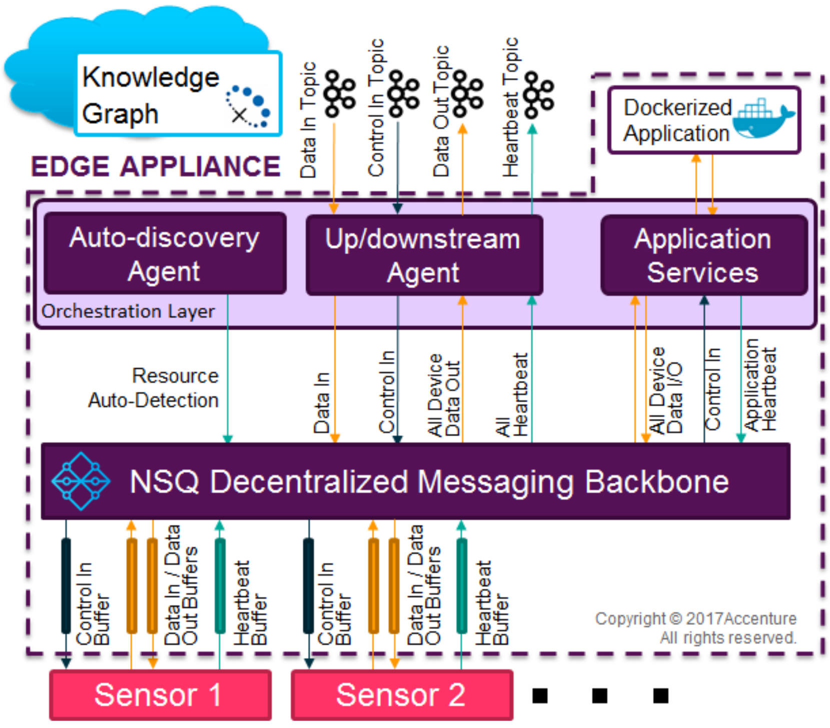

Figure 3: Edge Appliance Architecture

Containerization

Containerization, providing near native runtime specs[8], allows for the application engineer to encapsulate nearly all software dependencies of the target device. This allows the platform to orchestrate the

[7] "A realtime distributed messaging platform", 2017. [Online]. Available: http://nsq.io/

[8] W. Felter and F. Alexandre, "An updated performance comparison of virtual machines and Linux containers," in *IEEE Performance Analysis of Systems and Software (ISPASS)*, 2015.

deployment of applications across any device if it supports Docker containerization, an extensively used industry standard technology in the cloud which eases the skills transition for the application engineer.

Our platform provides an interaction layer on top of the containers for setting configurable runtime parameters. For example, we can declare that some applications require access to specialized hardware such as a GPU. After querying the knowledge graph, the orchestrator layer enables launching of the container on the specified device target while exposing the required hardware components via Docker runtime parameters.

USE-CASE IMPLEMENTATION

We can leverage a combination of these key solution-architecture components (knowledge graph, messaging backbone and containerization) to address the various challenges we have previously identified. While all components are used in each use case, highlighted are the key contributions of our framework to resolving some of the major requirements for each use case.

Large Oil and Gas (O&G) Services Company

In the case of the O&G service provider the most impactful contribution to addressing the key requirements and important constructs from our edge framework for this client is as follows:

- ***Knowledge graph – Extensibility and Agnostic Support/*Heterogeneous HW Support*:*** The knowledge graph enables the client to extend support for any vendor or 3rd party provided applications on the edge devices. The knowledge-graph contains the metadata about each application's software dependencies and hardware requirements. It also contains metadata about the capabilities provided by each edge device instance and the various sensors attached to it from which the client can query and infer all the edge devices where an application could be deployed. The graph provides querying and inferencing capabilities of new and old devices alike along with their applications by capturing associated capability metadata. Furthermore, the knowledge graph promotes a structured environment in which onboarding heterogeneous edge-hardware is simplified for technology roadmaps and migration plans.
- ***Containerization* - Heterogeneous SW Support/Deployment Modularity*:*** Applying containerization decouples the edge-applications from one another and provides an abstraction over the underlying sensor hardware. When combined with the application-metadata captured in the knowledge-graph, it provides the capability to encapsulate, isolate dependencies and utilize a wide-variety of available CEP and database products or AI frameworks (proprietary or open source) without worry of conflicts between to co-located application containers.

Large Industrial Construction and Mining Manufacturing Company

Similar to that of the O&G use case, the edge framework components address all the key challenges and requirements in the heterogeneous environment construction and mining company as follows:

- ***Knowledge graph – Decoupling of HW/SW stack:*** The knowledge-graph captures the data subscription requirements for each of the edge applications, along with the sensor-configuration and data available on each edge device. By referencing this information from the graph, we can automate the build and deployment of applications specific to each mining equipment in the field. This simplifies the deployment of edge devices and edge applications while enabling scalability.
- ***Messaging backbone - Data message and protocol management:*** The abstraction and standard interface provided by the messaging backbone on each of the edge gateways enables the client to handle multiple message protocols on the edge device (MQTT, MODBUS, etc.). This means the client can support data-ingestion from a wide variety of sensors with all the services that a messaging broker provides (queuing, publish, subscribe, caching, persistence, filtering, etc.). Applications can then consume and process only contextually relevant high-fidelity data on the gateway itself and thus reduce the volume of data transmitted over the network.

CONCLUSION

Our edge framework analytics platform addresses key challenges that clients with IoT typically face: agnostic support and deployment, heterogeneous support of and decoupling of HW and SW, as well as message management. Our knowledge-driven approach simplifies industrial and commercial entities requiring IoT deployments to easily develop, reuse and seamlessly deploy applications to existing, heterogeneous collections of edge devices through a manageable and scalable framework. Our framework provides an abstraction layer that enables collaboration between the different IoT stakeholders (Device Expert, Application Expert and Field Engineer) and allows each role to focus on their area of domain expertise. This abstraction layer provided by the knowledge graph seamlessly tracks and manages the multitude of hardware and software parameters present in existing heterogeneous deployments and, coupled with containerization techniques, enables the development and deployment of applications anywhere in the edge deployment topology.

Enterprises struggle with technology adoption and this is especially true with IoT implementations given the high speed with which IoT HW and SW has been evolving in the recent past. As a result, there is a need for an extensible framework that allows enterprises to efficiently utilize experts by enabling easy integration and migration to these new technologies. Our evolving edge framework addresses these needs and works toward future proofing IoT and edge analytics infrastructures while preserving precious domain knowledge needed for supporting growing edge platforms.

➢ Return to IIC Journal of Innovation landing page for more articles and past editions.

The views expressed in the *IIC Journal of Innovation* are the contributing authors' views and do not necessarily represent the views of their respective employers nor those of the Industrial Internet Consortium.

© 2017 The Industrial Internet Consortium logo is a registered trademark of Object Management Group®. Other logos, products and company names referenced in this publication are property of their respective companies.