Advanced Platform Development with Kubernetes

Enabling Data Management, the Internet of Things, Blockchain, and Machine Learning

—

Craig Johnston

Apress®

Advanced Platform Development with Kubernetes

Enabling Data Management, the Internet of Things, Blockchain, and Machine Learning

Craig Johnston

Apress®

Advanced Platform Development with Kubernetes: Enabling Data Management, the Internet of Things, Blockchain, and Machine Learning

Craig Johnston
Los Angeles, CA, USA

ISBN-13 (pbk): 978-1-4842-5610-7 ISBN-13 (electronic): 978-1-4842-5611-4
https://doi.org/10.1007/978-1-4842-5611-4

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Natalie Pao
Development Editor: James Markham
Coordinating Editor: Jessica Vakili

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 NY Plaza, New York, NY 10004. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-5610-7. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

Table of Contents

About the Author

Craig Johnston currently holds the position of Chief Architect at Deasil Works, Inc. and has been developing software for over 25 years. Craig's expertise revolves around microservices, artificial intelligence, algorithms, machine learning, and blockchain technologies.

Craig has helped lead his team to significantly improved productivity and return on investment across many client projects, leveraging Kubernetes, Docker, Golang, Cassandra, Kafka, and Elastic, to name a few. The team and he are developing more productive, stable, clean, and faster applications than ever in the past, and the results are beautiful and innovative IoT management systems, IoT implementations, mobile applications, business intelligence, data management, and machine learning platforms.

As the former Director of R&D at Napster and later a handful of Universal and Sony subsidiaries, Craig has been fortunate to spend many of his early days on the bleeding edge, in the open green fields of new media and disruptive technology.

Craig is successfully operating multiple commercial Kubernetes platforms utilizing all the technology and concepts proposed in *Advanced Platform Development with Kubernetes*.

About the Technical Reviewer

David Gonzalez is a DevOps engineer who has written three books about DevOps and microservices. He works as a consultant, helping large companies to advance their systems development, by tweaking related software processes and tools. David is also a Google Developer Expert (https://developers.google.com/experts/people/davidgonzalez-gonzalez) in Kubernetes (Google Container Engine) and a member of the Node.js Foundation, working on security in third-party npm packages. In his free time, he enjoys cycling and walking with his dogs in the green fields of Ireland.

Acknowledgments

I want to start by thanking Kelsey Hightower, who inspired me and so many others with his passion and excitement for technologies that advance a developer's productivity. Kelsey's live demonstrations, talks, and tutorials convinced me that Kubernetes is a platform for developing platforms. Kelsey is also responsible for the popularity of my Kubernetes development utility kubefwd.

A big thank you to my friend, co-worker, author, and software developer, David Elsensohn. David poured over every draft to ensure my English syntax would compile in the readers' minds. Thanks to everyone at Deasil Works, especially Jeff Masud, for helping me carve out time to write a book in one of our busiest years (and for fixing the clusters I broke along the way).

Thanks to Apress editors Natalie Pao and Jessica Vakili for their patience and encouragement. Thanks again to Natalie Pao for having a vision for this book and encouraging me to write it. Thanks to my technical reviewer David González for correcting my mistakes, unintentional obfuscations, and providing valuable guidance for technical clarity.

Lastly, thanks to my family and friends (most of them having no idea what a "Kubernetes" is) who encouraged me to stay focused and motivated. Thank you!

CHAPTER 1

Software Platform and the API

On October 28, 2018, IBM announced a $34 billion deal to buy Red Hat,[1] the company behind Red Hat Enterprise Linux (RHEL), and more recently Red Hat OpenShift, an enterprise Docker/Kubernetes application platform. What we see is $34 billion of evidence that Cloud-native and open source technologies, centered on the Linux ecosystem and empowered by Kubernetes, are leading disruption in enterprise software application platforms.

Any exposure to enterprise software marketing presents a steady stream of platform services released almost daily by major cloud providers, including products like Google Cloud Machine Learning Engine, Microsoft's Azure Machine Learning service, Amazon Managed Blockchain, and IBM Watson IoT Platform, to name a few. Big providers like Amazon, Microsoft, IBM, and Google are not only responding to market demand for these technologies but creating a greater awareness of their accessibility for solving problems across a variety of industries. Large software vendors are rapidly responding to the demand for these capabilities and perpetuate their demand by refining and marketing products that demonstrate their value. These vendors are often merely

[1]IBM to Buy Red Hat, the Top Linux Distributor, for $34 Billion." The New York Times, October 28, 2018, sec. Business. https://www.nytimes.com/2018/10/28/business/ibm-red-hat-cloudcomputing.html

C. Johnston, *Advanced Platform Development with Kubernetes*,
https://doi.org/10.1007/978-1-4842-5611-4_1

service-wrapping the latest in open source software, adding polished user interfaces and proprietary middleware. Peek under the hood of these hyper-cloud services and you often find a mesh of cloud-native and even vendor-neutral technologies for machine learning (ML), like TensorFlow, Keras, and PyTorch, or Blockchain capabilities powered by Ethereum and Hyperledger, and high-performance IoT data collectors like Prometheus and Kafka. These vendors are not stealing this technology from the open source community; some of the most significant contributions in this ecosystem are the vendors themselves.

Developing an enterprise-grade platform from the ground up, with capabilities as diverse as Blockchain and Machine Learning, would have required an enormous effort only a few years ago. Your other option would have been a significant investment and long-term commitment to a commercial platform. Google disrupted the entire commercial platform business with Kubernetes, a free, open source, cloud-native, and vendor-neutral system for the rapid development of new platforms that can easily support almost any technology with enterprise-grade security, stability, and scale. Expect to see another significant wave of platform innovation, as Kubernetes matures and allows software and platform developers to focus more time on features, with less custom work needed on infrastructure, networking, scaling, monitoring, and even security.

This book aims to build a simple demonstration platform in a vendor-neutral approach using Kubernetes. With only minimal modifications, this new platform should run on any primary cloud provider able to run Kubernetes and offer a small number of widely available dependencies such as storage, memory, and CPU. Each existing, open source technology implemented in this platform has a specialized focus on a particular solution. Offering Machine Learning, Blockchain, or IoT-based services will not in themselves be a core differentiator for a platform. However, operating these technologies together within Kubernetes provides a foundation in which to build and offer novel solutions through their combined efforts, along with providing a template for future additions.

In the early 1990s, databases were often operated and accessed as independent applications. The combination of a database and a web server revolutionized the Internet with dynamic database-driven websites.

These combinations seem obvious now, and Kubernetes together with service mesh technologies like Istio and Linkerd is making connections between diverse applications, even with conflicting dependencies, not only possible but adding security and telemetry to the platform.

Software Applications vs. Software Platforms

You may be a software developer and have a solution to a problem in a specific industry vertical. With a specific mix of closed and open source software, you wish to combine these capabilities under an API and expose them in support of a specific application. Alternatively, you may be a value-added reseller and want to offer customers an application development platform that comes with a suite of prepackaged features such as Machine Learning, Blockchain, or IoT data ingestion. Software platforms like Kubernetes are the ideal environment for developing a singular focused application or a platform as a service (PaaS) offering customers an environment in which they can develop and extend their applications (Figure 1-1).

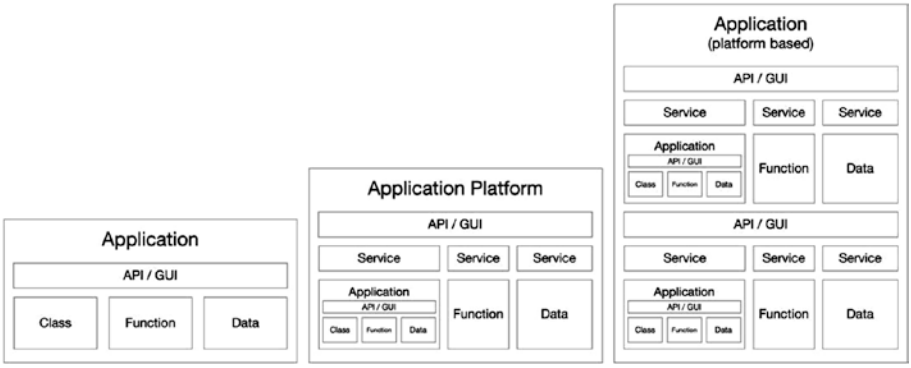

Figure 1-1. *A software application, a platform as a collection of applications, and a platform-based application*

Dependency Management and Encapsulation

Containerization has made running software applications more portable than ever by creating a single dependency, a container runtime. However, applications often need access to a sophisticated mix of resources, including external databases, GPUs (graphics processing units for machine learning), or persistent storage, and likely need to communicate with other applications for authentication, database access, and configuration services. Even a single containerized application typically needs some form of management over it and its access to external resources. The problem of managing connected containers is where Kubernetes comes in; Kubernetes orchestrates the containers of applications and manages their relationship to resources.

Network of Applications

Not all software applications need sophisticated platform architecture. Most software applications can be developed and merely run on a computer that meets their operational dependencies. Platforms come into play when you wish to operate multiple applications together and form an interconnected network of services, or when multiple applications can benefit from shared functionality, configuration, or resource management (Figure 1-2).

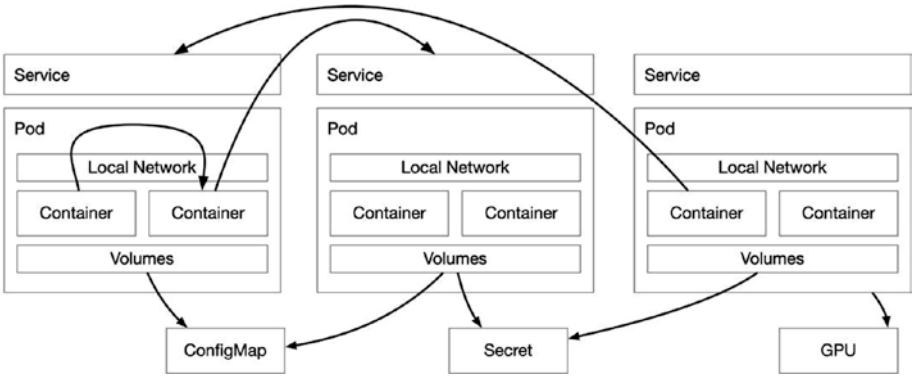

Figure 1-2. *Network of containerized applications*

Application Platform

Even if your goal is to develop a single-purpose online application, there are several reasons to embark on developing a software platform in Kubernetes. Large and small, complex and straightforward, enterprise and small-scale applications benefit when implemented in the context of a software platform. Software platforms provide an architecture to solve common problems and reduce the need for custom development in several areas, including communication, storage, scaling, security, and availability.

Architecting an application as a platform means that from the ground up the software is intended to be extended beyond its fundamental requirements, with the ability to upgrade and deploy new components independently. A proper platform welcomes the addition of the latest trends in open source, and when innovations arise, and open source products are released, it is successful software platforms that wrap and leverage their functionality to stay current. A proper software platform should never assume the label legacy; it should remain in a constant, iterative cycle of improvement.

The next section goes more in-depth into how this is accomplished with Kubernetes as the central component. Kubernetes solves the problems that traditional enterprise solutions like the service-oriented architecture (SOA) have attempted to solve for decades, only Kubernetes does this with protocols and methodologies that power the global Internet, like DNS, TCP, and HTTP, and wraps them in an elegant and robust API, accessible through those very same protocols. The platform is architected around Kubernetes's concept of a Service and its relationship to containerized applications (Figure 1-3).

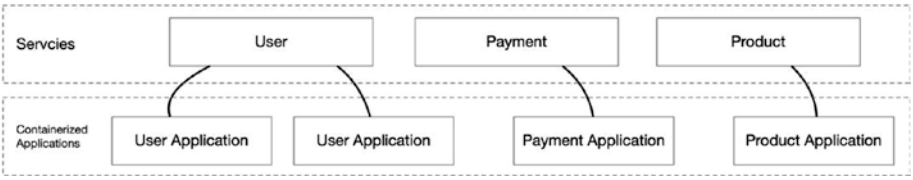

Figure 1-3. *The relationship between services and application*

Platform Requirements

This book focuses on implementing a foundational data-driven, Data Science, and Machine Learning platform, primarily but not limited to IoT data, and providing opportunities for interconnection with Blockchain technology. If this sounds like a lot of hype, it is, and as the hype fades, it's time to get to work. As these technologies leave the lab, they begin to fade into the background, and over the next decade, they will begin to silently provide their solutions behind new and innovative products.

If you are familiar with the "Gartner Hype Cycle for Emerging Technologies" (Figure 1-4) in 2018,[2] you would have seen deep neural networks (deep learning), IoT platforms, and Blockchain still on the "peak of inflated expectations" and rolling toward the "trough of disillusionment." Disillusionment sounds dire, but Gartner marks the following phase for these technologies as the "slope of enlightenment" and a later plateau in the next 5–10 years. Much innovation happens before these technologies plateau, and a flexible architecture built from a collection of connected containers, managed by Kubernetes, should easily keep you relevant for the next decade or more.

[2]Walker, Mike. "Hype Cycle for Emerging Technologies, 2018." Gartner. https://www.gartner.com/en/documents/3885468/hype-cycle-for-emerging-technologies-2018.

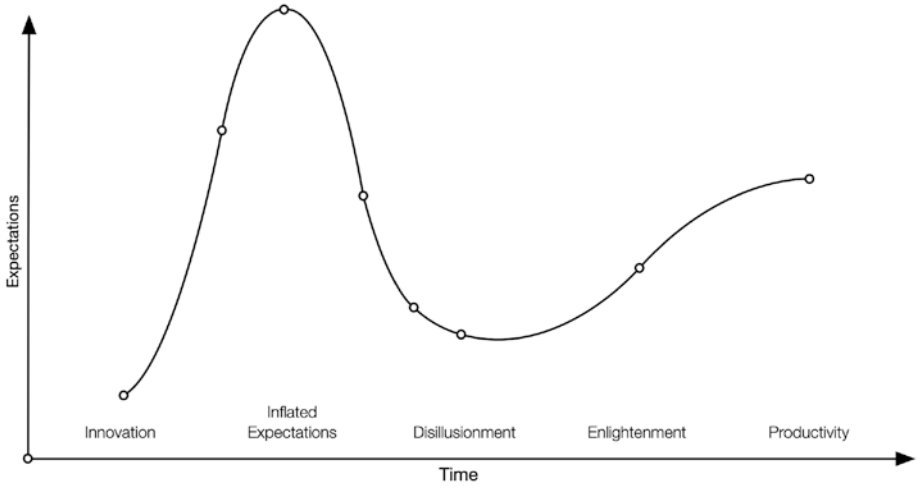

Figure 1-4. *Gartner's Hype Cycle for Emerging Technologies, 2018*[3]

While individual components may come and go as trends peak and plateau, data is here to stay; the platform needs to store it, transform it, and provide access to it by the latest innovations that produce value from it. If there is a central requirement for Advanced Platform Development with Kubernetes, it would be accessing the value of data, continuously, through the latest innovative technologies in IoT, Machine Learning, Blockchain, and whatever comes next.

A final requirement of Advanced Platform Development with Kubernetes is to stay open source, Cloud native, and vendor neutral. A platform with these principles can leverage open source to harness the global community of contributing software developers looking to solve the same problems we are. Remaining Cloud native and vendor neutral means not being tied to or constrained by a specific vendor and is just as functional in a private data center, as it can on AWS, GKE, Azure, or all of them combined as the concept of "hybrid cloud" grows in popularity.

[3]Walker, Mike. "Hype Cycle for Emerging Technologies, 2018." Gartner. https://www.gartner.com/en/documents/3885468/hype-cycle-for-emerging-technologies-2018.

Platform Architecture

With Kubernetes it is common to build software platforms from a collection of specialized components, written in a variety of languages and having vastly different and even conflicting dependencies. A good platform can encapsulate different components and abstract their interfaces into a standard API or set of APIs.

Object-oriented software concepts are a great reference tool for overall platform architecture. Trends in microservice architectures encourage the development of several, minimal applications, often taking the form of an object Class, providing a limited number of operations in a specific problem domain, letting the larger platform take care of aggregate business logic. To implement this approach, take the concept of an Object and apply it to the Kubernetes implementation of a Service (Figure 1-5). Like software interfaces, Kubernetes services represent one or more entry points to an application. The object-oriented software principles of abstraction, encapsulation, inheritance, and polymorphism can express every layer of the platform architecture.

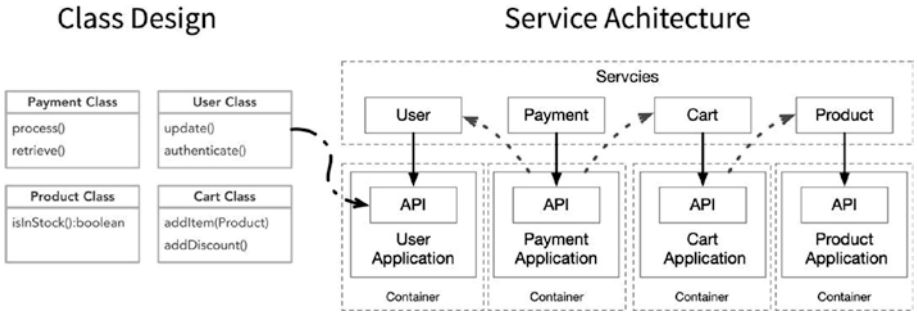

Figure 1-5. *Class design and service architecture*

Kubernetes is well suited for platform development and may be overkill for any lesser task. I believe, as I hope you discover in this book, that there is not much to debate on Kubernetes fitness for platform development. Containers solved many of the problems with dependency management by isolating and encapsulating components; Kubernetes manages these containers and in doing so forms the framework for a software platform.

Platform Capabilities

The purpose of the platform outlined in this book is to demonstrate how Kubernetes gives developers the ability to assemble a diverse range of technologies, wire them together, and manage them with the Kubernetes API. Developing platforms with Kubernetes reduces the risk and expense of adopting the latest trends. Kubernetes not only enables rapid development but can easily support parallel efforts. We develop a software platform with as little programming as necessary. We use declarative configurations to tell Kubernetes what we want. We use open source applications to build a base software platform, providing IoT data collection, Machine Learning capabilities, and the ability to interact with a private managed Blockchain.

Starting with the ingestion, storage, and retrieval of data, a core capability of the platform is a robust data layer (Figure 1-6). The platform must be able to ingest large amounts of data from IoT devices and other external sources including a private managed Blockchain. Applications such as Elasticsearch, Kafka, and Prometheus manage data indexing, message queueing, and metrics aggregation. Specific services capture Blockchain transactions from applications such as Ethereum Geth nodes and send them to Apache Kafka for queueing and Elasticsearch for indexing.

Above the data layer sits an application layer (Figure 1-6), providing capabilities utilizing this data, such as Machine Learning automation. Platform services wire together and expose data sources that export and serve persistent and streaming data usable for Machine Learning experiments, production AI inference, and business analytics.

The Platform naturally supports the expansion of features through the management of containers by Kubernetes. Serverless technologies including OpenFaaS provide higher-level expansion of features. Serverless support allows the rapid development and deployment of real-time data processors, operations that run at specific intervals, and new API endpoints, allowing specialized access to data, performing AI operations, or modifying the state of the platform itself.

The platform envisioned in this book forms a data-driven foundation for working with trending technologies, specializing in Machine Learning, Blockchain, and IoT. Components for the ingestion, storage, indexing, and queueing of data are brought together and allow efficient access to data between the specialized technologies. The platform provides data scientists the access to data and tools needed to perform Machine Learning experimentation and the development of production-ready neural network models for deployment by way of Serverless functions able to make predictions, perform classification, and detect anomalies from existing and inbound data. Blockchain technology is used to demonstrate how third-party ledger transactions and smart contract executions can seamlessly inner-connect to the data processing pipeline.

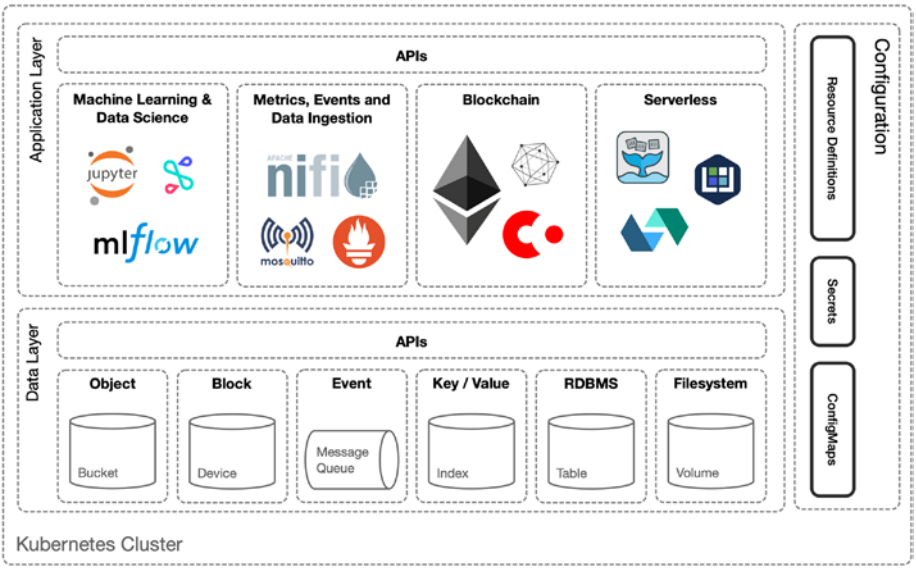

Figure 1-6. *Platform application and data layers*

The platform, developed iteratively, eventually consists of a large number of services, ranging in size and complexity, mixing giant monoliths mixed with small serverless functions. Some services consist of a cluster of Java applications, while some services only execute a few lines of Python. If this sounds like a nightmare, it is not. Fortunately, containerization has helped us isolate an application's operation and dependencies, exposing what is needed to configure, control, and communicate with the application. However, containerization only gives us limited options for visibility and control over our collection of services. Kubernetes gives us great configuration access controls over infrastructure resources, security, and networking, but leaves platform application–level concerns like encrypted communication between services, telemetry, observability, and tracing, to the applications themselves or higher-level specialized systems like Istio or Linkerd. The platform developed in this book is a collection of services that can operate with or without Istio or Linkerd. Istio and Linkerd are still young, and best practices for implementing them are still maturing.

The next few sections define the platform's three main requirements: IoT, Blockchain, and Machine Learning in more detail (Figure 1-7).

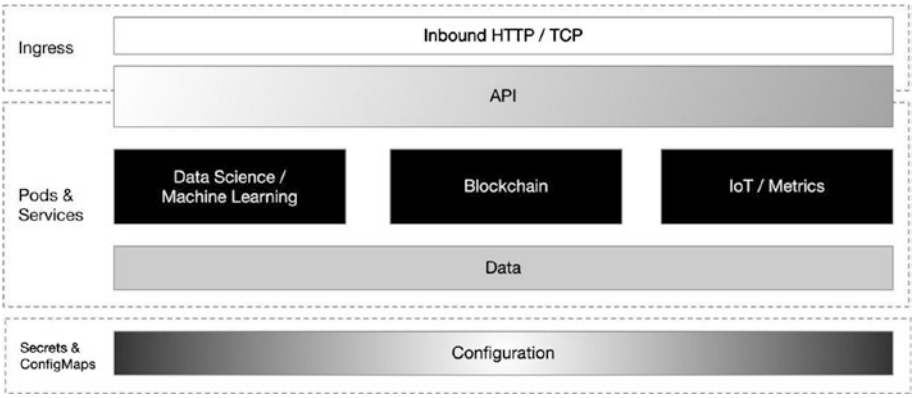

Figure 1-7. *IoT, Blockchain, and Machine Learning in Kubernetes*

IoT

The Internet of Things (IoT) and the newer Industrial Internet of Things (IIoT) are technologies that have matured past the hype phase. The physical devices of an industry are not only expected to be connected and controlled over the Internet but have a closer relationship to their larger data platforms. Kubernetes is capable of managing both the data and control plane in every aspect of IoT. This book focuses on three main uses for Kubernetes in the IoT domain, including the ingestion of data, as an edge gateway, and even an operating system (Figure 1-8).

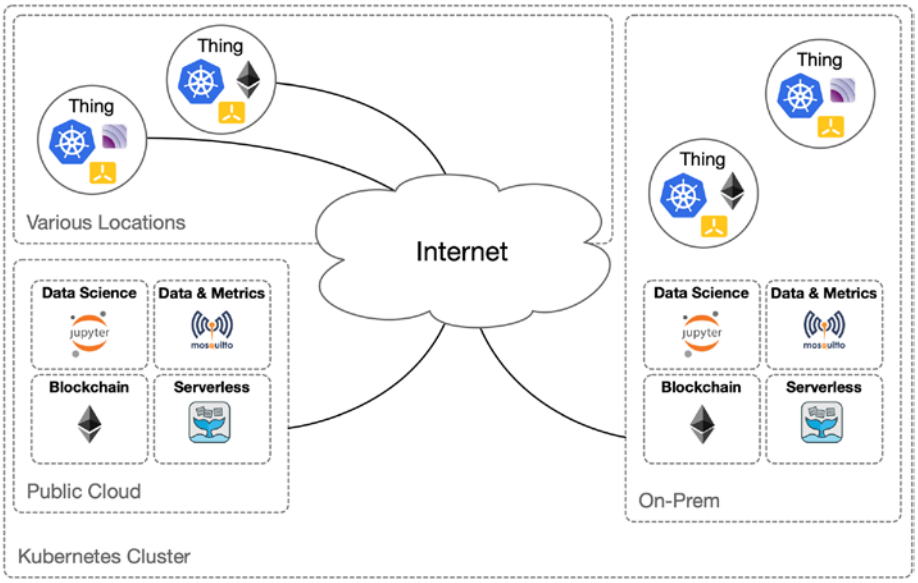

Figure 1-8. *Three uses of Kubernetes platforms in IoT*

Ingestion of Data

The first and most obvious use of Kubernetes is to orchestrate a data ingestion platform. IoT devices have the potential of producing a large volume of metrics. Gathering metrics is only one part of the problem. Gathering, transforming, and processing metrics into valuable data and performing actions on that data requires a sophisticated data pipeline. IoT devices utilize a wide range of communication protocols, with varying quality of support from various software products built to specific devices and protocols. To effectively support data from a range of IoT and IIoT devices, the platform needs to speak in protocols like AMQP (Advanced Message Queuing Protocol), MQTT (Message Queue Telemetry Transport), CoAP (Constrained Application Protocol), raw TCP, and HTTP, to name a few.

JSON (JavaScript Object Notation) over HTTP is the most popular and supported messaging protocol on the Internet. Every significant programming language supports JSON. JSON drives nearly all public cloud APIs in one way or another. Kubernetes's own API is JSON-based, and YAML, a superset of JSON, is the preferred method of declaring the desired state.

JSON may not be as efficient as binary messages or as descriptive as XML; however, converting all inbound messages to JSON allows the platform to unify data ingestion on the most flexible and portable standard available today. The platform consists of custom microservices implementing a variety of protocols, parsing inbound message or querying and scraping remote sources, and transforming these messages to JSON. An HTTP collection service accepts JSON transformed data to buffer and batch. This architecture (Figure 1-9) allows unlimited horizontal scaling, accommodating large volumes of data.

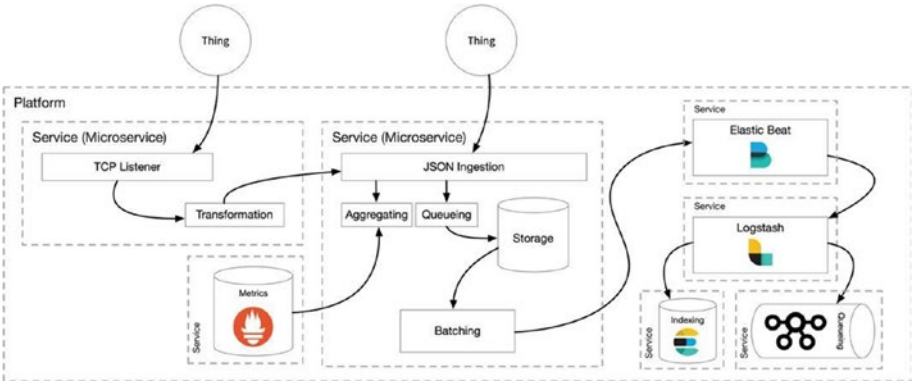

Figure 1-9. *IoT data ingestion*

The chapter *"Pipeline"* covers the implementation of the ingestion and transformation services: Apache NiFi, Prometheus, Logstash, Elasticsearch, and Kafka.

Edge Gateway

Kubernetes in the IoT space is beginning to include on-premises, edge deployments. These are mini-clusters that often include as little as a single node. On-premises clusters often operate a scaled-down version of the larger platform and are typically responsible for communicating with IoT devices on the local area network, or the nodes themselves are attached to proprietary hardware and protocols, legacy control systems, or lower-level, serial communication interfaces. Industrial use cases for the collection of data can often include sub-second sampling of device sensors or merely a volume of data only useful for classification, anomaly detection, or aggregation.

An on-premises platform (Figure 1-10) can handle the initial gathering and processing of metrics and communicate results back to a larger data processing platform. New Kubernetes distributions such as Minikube, Microk8s, k3s, and KubeEdge specialize in small or single-node implementations on commodity hardware.

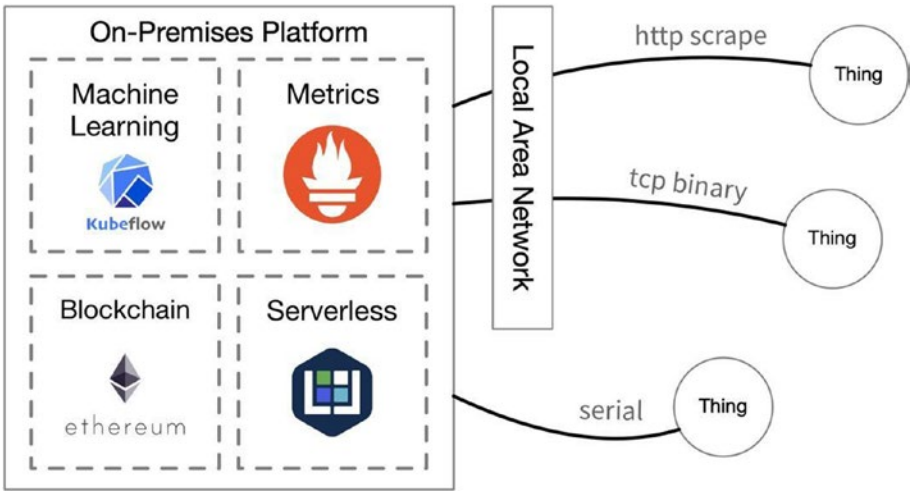

Figure 1-10. *On-premises Kubernetes platform*

Running a scaled-down platform on-premises solves many security and compliance issues with data handling. In scenarios where data must remain on-premises by strict compliance rules, on-premises clusters can process data, whose resulting metadata, inference, and metrics aggregation can transmit to a remote platform for further processing, analysis, or action.

IoT OS

The third use of Kubernetes for IoT addressed in this book is just starting to take root, that is, Kubernetes as an IoT operating system (Figure 1-11). ARM processors are cheap and energy efficient. Products like the Raspberry Pi have made them incredibly popular for hobbyists, education, and commercial prototyping. Container support for ARM-based systems has been around for a few years now, and running containerized applications on IoT devices has nearly all the advantages as does running them on more powerful and sophisticated hardware. IoT devices running containers orchestrated by Kubernetes can take advantage of features like rolling updates to eliminate downtime when upgrading applications. Running a small collection of containers in Kubernetes on an IoT device lets you take advantage of microservices application architecture, resource allocation, monitoring, and self-healing. The development of software for small, low-power devices once required using a proprietary operating system and writing much of the code to support activities like firmware updates, crash reporting, and resource allocation. IoT devices supporting scaled-down versions of Kubernetes are still new and poised for growth as more developers begin to see the potential for many of the common challenges with IoT software solved with platforms like Kubernetes.

Slimmed down distributions, like the 40mb k3s, are making Kubernetes an excellent choice for small, resource-limited devices like the Raspberry Pi and the large family of SOC boards on the market today.

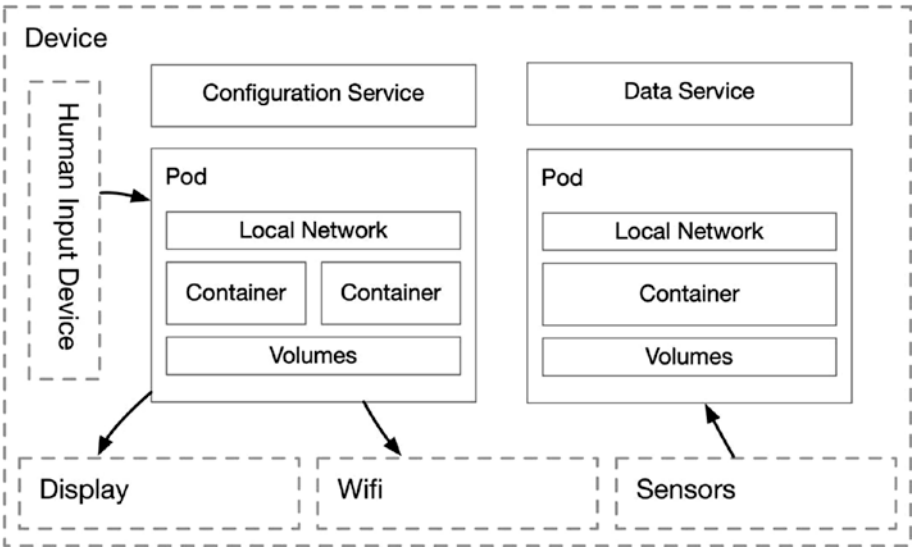

Figure 1-11. *Kubernetes platform on an IoT device*

Blockchain

With the maturity of Smart Contracts,[4] Blockchain technology is now a type of platform[5] itself. Smart contracts allow the storage and execution of code within the distributed, immutable ledger of the Blockchain (see Chapters 9 and 10). The inclusion of Blockchain technology provides the platform a capability for transactional communication with untrusted participants. Untrusted in this context means no personal or legal contractional trust is needed to transmit value expressed as data. Blockchain provides a permanent record of a transaction, verified in a shared ledger. The external parties only need to operate Blockchain

[4] https://en.wikipedia.org/wiki/Smart_contract

[5] Blockgeeks. "Smart Contract Platforms [A Deep Dive Investigation]," May 11, 2018. https://blockgeeks.com/guides/different-smart-contract-platforms/.