Accelerating Edge Computing with Arm and Rancher k3s Lightweight Kubernetes

AUTHORS

Mark Abrams, Field Engineer, Rancher

Bhumik Patel - Director, Software Ecosystem Development, Arm

Accelerating Edge Computing with Arm and Rancher k3s Lightweight Kubernetes

Executive Summary

Edge computing has become a clear strategic goal for many enterprises, for some it is already an absolute necessity. As the demands of applications software and the capabilities of hardware continue to grow, containers and Kubernetes have become a compelling technology for edge computing. While there is no corollary to Moore's Law in the software space, any casual observer can see that the improvements in software grow in lock step with hardware capability. In just the past 5 years, the way that software is manufactured and maintained has changed dramatically with the introduction of Kubernetes container orchestration. The pace of adoption of Kubernetes has been very rapid and is accounted for by numerous factors. Two of the most significant are the ability to provide advanced application operations as system wide utilities, and the standard management and control interface Kubernetes delivers across any infrastructure. Simultaneously, we have seen an explosion of hardware to every corner of our daily lives in both private and public locations.

Kubernetes was contributed as an open source solution to the community by Google and is an offshoot of their in-house experience and success running container orchestration. In addition to having its roots in Google's data center success, Kubernetes delivers on the promise to have a standard interface across any infrastructure (aka hybrid cloud)[1] and now it is finding its way to the edge as well. Kubernetes also found immediate success in some well known companies like Spotify and Ancestry.com and the list of success stories continues to grow in every industry.[2]

1 Gupta, Anurag. "Why Is Kubernetes so Popular?" Opensource.com, Oct. 2017, opensource.com/article/17/10/why-kubernetes-so-popular.
2 "Case Studies." Kubernetes, 2019, kubernetes.io/case-studies/.

In the past 15 years the cloud has emerged as a centralized computing platform based originally on commodity hardware made available through APIs. It has become apparent, however, that the same cloud computing model does not apply to edge devices due to high data loads and latency. The edge is often disconnected from the internet and other external networks for long periods of time. Originally, edge devices were simply not built for data processing or what has become edge compute. It quickly became obvious that sending the data to the cloud or a remote data center has disadvantages over processing the data in place or as close to the place of creation and usage as possible.[3] With that understanding and more powerful small devices, edge devices have become a key part of modern computing ecosystems.

In the last year, we have seen enterprises from a wide array of industries begin to include Kubernetes in their strategic edge initiatives. This is due in part to the release of k3s, the lightweight, small footprint Kubernetes developed by Rancher Labs. The result is that edge devices can now take advantage of the same advanced tooling that was developed originally for cloud to deliver better, more robust and less costly experiences.[4] This whitepaper documents how k3s and Arm come together to be an ideal edge solution.

3 Hobbs, Andrew. "IoT: Understanding the Shift from Cloud to Edge Computing." Internet of Business, 23 Aug. 2018, internetofbusiness.com/shift-from-cloud-to-edge-computing/.
4 Dvoretskyi , Ihor, and Chris Aniszczyk. "Cncf/Foundation." GitHub, 2015, github.com/cncf/foundation/blob/master/charter.md.

Introduction

To start, we are going to establish what containers are and the value of container orchestration as a model for software development, delivery and distribution. With that, we will bring Kubernetes, the industry-standard container orchestrator[5] into the picture and describe how Kubernetes addresses edge constraints.

At first glance, Kubernetes may look to be too large and complex for edge devices which typically have a smaller resource footprint than we are accustomed to in the data center or cloud. This issue is solved by k3s, a lightweight Kubernetes distribution. We will discuss how k3s delivers the core functionality of Kubernetes in a lightweight package that is ready for the edge. The k3s distribution has been built for quite a few distinct device architectures as of the writing of this paper. It is available for both Armv7-A (AARch32) and Armv8-A (AArch64) architectures. Guidance on how to quickly install k3s is included.

It is important to understand both why containers are valuable to the edge and also how k3s on Arm can be used in practice. In closing, we will share three cluster architectures for using Kubernetes on the edge and some examples of tactical uses of container orchestration at the edge.

Rather than include a Glossary in these pages, we refer readers to the edge-glossary[6] under development by the LFEdge and a sub project of the Linux Foundation.

5 Barnard, Kaitlyn. "CNCF Survey: Use of Cloud Native Technologies in Production Has Grown Over 200%." https://www.cncf.io, 29 Aug. 2018, www.cncf.io/blog/2018/08/29/cncf-survey-use-of-cloud-native-technologies-in-production-has-grown-over-200-percent/.

6 https://github.com/lf-edge/glossary/blob/master/edge-glossary.md

Containers

Today we can identify and refer to an electronic component or a hardware component by it's standardized name and function such as an M4 nut and bolt or a 555 timer IC. Software has yet to have this degree of standardization and containers are a move in that direction.

Containers are often compared to virtual machines, however they differ in some very significant ways. The most significant and relevant difference for edge computing may be the native performance of containerized processes. Where virtual machines require a hypervisor and fully loaded operating system to run, containers simply use the resources of the host operating system. Containers give us process isolation without the overhead of a virtual machine. This is obviously significant when it comes to edge computing where resources are limited. Modern containers are more than just process isolation, they are modular components of reusable software. They are the building blocks of complex software solutions.

Containerization

Models for isolating individual software processes and limiting them based on resource utilization have existed for at least 40 years. Process isolation via the Unix chroot system call was first introduced in 1979.[7] Eventually lxc on Linux and Windows Communication Framework (WCF) on Windows made their way into the software methodology but they still lacked a truly reusable component nature, especially outside of the stacks they were designed to operate within.

The Docker container model, plus container orchestration technologies (which we discuss in the next section) are helping shift the software development lifecycle toward a component model. Docker and eventually the Container Runtime Interface (CRI) delivered not only process isolated runtimes but the ability to compose and capture the component parts of runnable process as layers of a file system. With composition in this manner we are able to validate and verify each change in the component stack as well as maintain strong vulnerability checks throughout.

7 Osnat, Rani. "A Brief History of Containers: From the 1970s to 2017." https://blog.aquasec.com, 21 Mar. 2018, blog.aquasec.com/a-brief-history-of-containers-from-1970s-chroot-to-docker-2016.

Containerizing workloads is improving software by providing environmental and deployment consistency, security through process isolation, and by reducing the overall complexity of building software.

- **Environmental consistency** - Containers are built to include exactly what is required for the software to function. No more and no less. In this way, containers are guaranteed to run exactly the same regardless of where they run.
- **Process isolation** - Containerization is being used to create applications which are operated as a system of microservices. Each service may stand alone in a container. The containers are then isolated from other containers or permitted to interact with other containers. Isolation can be used as part of a security strategy.
- **Reusable components** - The idea of software reuse has been notably discussed for 50 years[8]. Container technology has moved us closer to the goal of not just software reuse but also componentization. Standards in the CRI make it easy for systems operators to guarantee the components will operate in the environment provided.
- **Reduced application complexity** - In computer hardware manufacturing componentization became a game changer for the industry. Software is similarly beginning to see the impact with the advent of containers and container orchestration. Complexity is reduced in Kubernetes because the environment is consistent for common operations on containerized workloads. A common environment for deployment reduces the development of applications to the business logic while leaving the complexities of common utility to the underlying system.
- **Deployment consistency** - With isolated and componentized software, containers provide us a means to deploy the exact same thing every time. While this is an assumed and obvious need in software it is being truly realized in organizations today which have adopted containerized workloads as the model for software solutions.

8 Kotovs, Vladimirs. "Forty Years of Software Reuse." Scientific Journal of Riga Technical University. Computer Sciences, vol. 38, no. 38, 2009, pp. 153–160., doi:10.2478/v10143-009-0013-y.

Container orchestration

As valuable as containers have become, they still have undesirable limitations when they are put into a runtime state. Regardless of what process the container houses, the limitations of availability, reliability and scalability are universal. Securing containers from one another and at the same time allowing them to communicate when necessary is required to build functional applications from a set of microservices.

By themselves, containers are not a sufficient building material. Just like we need standards to interface nuts to bolts, containers need a common and consistent interface in order to be resilient, secure and composable. Kubernetes provides these capabilities and more and it is quickly becoming the most widely adopted solution for container orchestration.[9]

Fundamentally, container orchestration is nothing more than a control loop operating on a set of instructions defining a desired state of the system and then taking the appropriate actions to make that the active state of the system. The emergent properties of this system go well beyond the same set of manual processes - creating an environment for simplified operations, robust computing, improved deployment outcomes and faster, more durable change management.

- **Simplified operations** - Container orchestration handles aspects of software applications that are critical to making systems highly available, reliable and scalable. Having this functionality as a subsystem instead of developing it on a per application basis has clear benefits to the development lifecycle. In addition, having a system to do this work means operations teams can provide the functionality as a service to development teams and similarly devops can take advantage of the consistency of these features.
- **Robust compute** - Kubernetes operates on the simple principle of control structures. The desired runtime state of an application is declared using a standard configuration format. The Kubernetes controller(s) check the declared state at timed intervals and then they will set into motion the operations to make the system stable by making the active state of the system equal to the desired state.

9 Barnard, Kaitlyn. "CNCF Survey: Use of Cloud Native Technologies in Production Has Grown Over 200%." https://www.cncf.io, 29 Aug. 2018, www.cncf.io/blog/2018/08/29/cncf-survey-use-of-cloud-native-technologies-in-production-has-grown-over-200-percent/.

- **A model for deployment** - This desired state model provides a "hands off" solution for edge computing where the hardware and software are often monitored remotely and may be hard to access or physically isolated in ways that are very different from the traditional data center. As we successfully move some of the typical hardware solutions up the stack to software, we can use the methods of digital and remote deployment to fix, upgrade, migrate and alter existing systems. This is a great solution for edge where it is significantly more challenging and costly to make hardware modifications, flash firmware or upgrade device interfaces.
- **Rapid, robust change management** - Kubernetes provides multiple benefits to the traditional challenges for updating software and even delivers significant value in updating hardware components. Kubernetes provides the facilities for rolling updates of software resulting in zero downtime when change is necessary. Perhaps more impressive than the software upgrade capability is that fact that hardware can be replaced in a multi-node cluster while the system continues to operate at full function. Kubernetes provides the foundational facilities for additional approaches to both software and hardware upgrades.

Kubernetes can be configured in several different ways as a container orchestration platform. A few of these various architectures will be further discussed in the cluster architecture section below.

A Lightweight Kubernetes Distribution

K3s

Rancher Labs created k3s[10] as a lightweight certified Kubernetes distribution designed for use in production environments. We will describe how k3s became a lightweight Kubernetes distribution below.

While k3s can be run in any Linux OS, Rancher Labs also released k3os a container operating system with k3s pre-installed and ready to operate.[11] A container operating system is a specialized OS used for container orchestration and is beyond the scope of this whitepaper. Several articles are available for more information.[12]

The following sections provide insight into how and why Kubernetes was simplified into the lightweight k3s distribution.

Design principles

As an open source community driven project, the Kubernetes code tree is vast and many extra features and functionality exist out of the box or can be enabled at runtime. In some cases, these extras are similar to Linux modules[13] which are not installed in the kernel by default but, when enabled interact and integrate with the kernel seamlessly. In other cases, these extras are implemented to provide tight integration with infrastructure providers or storage providers. In practice, each deployment of Kubernetes is hosting a massive amount of unused code paths. How massive? The k3s project removes over 1 million lines of code.

Given all this extra, mostly unused functionality, the architects of k3s asked themselves this basic question: "What if we could have just the parts of Kubernetes we need"?

10 https://k3s.io
11 Rancher. "Rancher/k3os." GitHub, 26 June 2019, github.com/rancher/k3os.
12 "Comparison of Container Operating Systems." Rancher Labs, 16 Apr. 2019, rancher.com/blog/2019/comparison-of-container-operating-systems/.
13 Debian. "Chapter 3. The System Initialization." Chapter 3. The System Initialization, 2019, www.debian.org/doc/manuals/debian-reference/ch03.en.html#_the_kernel_module_initialization.

Implementation

The removal of extra and unused functionality is unquestionably a significant part of the value that k3s brings to container orchestration on the edge. In addition to the millions of lines of code that were removed, the architects of k3s sought to overcome the burden of setup, installation and maintenance of Kubernetes. This is achieved in k3s by a) removing extra features, b) eliminating external dependencies, c) reducing the number of binaries required at runtime, and d) reducing the complexity of installation. The result is a lightweight, production capable, container orchestration system which meets all the requirements of a certified Kubernetes distribution.

Features removed

- Legacy, alpha, non-default features
- In-tree plugins (cloud providers and storage plugins) which can be replaced with out of tree add-ons

External dependencies

- Remove the dependency on etcd
    - Add sqlite3 as the default storage mechanism. etcd3 is still available, but not the default.
- Minimal to no OS dependencies (just a sane kernel and cgroup mounts needed).
- Required dependencies built into the k3s releases
    - containerd
    - Flannel
    - CoreDNS
    - CNI
    - Host utilities (iptables, socat, etc)

Reducing Kubernetes complexity

In Kelsey Hightower's canonical article "Kubernetes the Hard Way"[14] he provides instructions for installing Kubernetes including the underlying toolset and dependencies as well as at least seven binaries which must be installed to operate the cluster. Compare this with k3s which has composed all the required parts, including dependencies, into a single binary. While there are tools and distributions to help install the Kubernetes specific components of a cluster, the value of a single binary for edge use cases is clear. K3s simplifies both the ease of installation, runtime operations, and maintenance of kubernetes.

Put simply, k3s is Kubernetes wrapped in simple launcher that handles a lot of the complexity of TLS and options for the embedded binaries. Kubernetes is a complex system and operations teams should fully understand how it works, however this does not mean container orchestration should be complex to install, run or operate.

14 Hightower, Kelsey. "Kelseyhightower/Kubernetes-the-Hard-Way." GitHub, 30 Sept. 2018, github.com/kelseyhightower/kubernetes-the-hard-way.

K3s Addresses Edge Constraints

Every enterprise is realizing cloud in a way that is most suitable for their organizational, process, and business model. During the research, development and/or adoption process enterprises have also realized that cloud computing has limitations depending on the workloads, systems, end users, or processes that are accessing the cloud. Many of those constraints are solvable using systems that already sit between the cloud and edge including the edge itself.

A warehouse or a factory may have 1000s of devices. Each of these devices is an independent unit which can only operate on it's own data save for the occasional mesh network in which they transport data for one another. Add Kubernetes to those devices and they become a cluster of compute, capable of not only communicating across devices, but also processing data in place or on another device within the cluster. In addition many other edge constraints are alleviated or completely overcome by adding cluster orchestration as a subsystem to a single node or across co-located devices.

The edge is composed of massive numbers of resource constrained devices producing substantial amounts of data which is unrealistic to transmit over the wire for processing in the cloud because of bandwidth constraints, near real time processing / response requirements, network latency and limited or no connectivity with external networks. K3s addresses these concerns in the following ways:

- **Remote device locations** - The definition of edge computing is said to be doing compute processes close to the data that is being produced or consumed by systems or users.[15] K3s is a production grade Kubernetes distribution that can be easily installed and operated in remote locations without the overhead of a standard Kubernetes installation and operational process.
- **Bandwidth** - One of the main issues about cloud computing for edge devices is the use of massive amounts of network bandwidth to have large data sets transferred to centralized cloud data centers where the data is processed and results are sent back to the edge. One of the keys to solving this issue is simply to do as much compute at the edge as possible. K3s provides the ability to compute at the edge with the same robust, scalable and reliable capabilities that are available in the cloud.

15 Gill, Bob, and David Smith. "The Edge Completes the Cloud: A Gartner Trend Insight Report." Gartner, 14 Sept. 2018, www.gartner.com/document/3889058.

- **Edge device updates/upgrades** - As a Kubernetes distribution, k3s can be deployed as a single binary onto new or existing hardware. This means systems that are already on the edge can be quickly adapted to run containers at the current scale of the edge devices. Devices can operate independently or be joined into a cluster providing zero downtime upgrade capability.
- **Compute resource limits** - Standard upstream Kubernetes is a not a small system to operate, out of the box needing about 8GB of memory to host the data plane (etcd)[16]. Compare this to k3s which can operate in 512MB- of memory leaving about half of that capacity for containers to function in a single node cluster.
- **Limited or no network connectivity** - Kubernetes can be fully operated in air-gapped and closed networks with no loss of functionality. Edge clusters and the containers they orchestrate can be intentionally designed to take advantage of local compute and only push or pull from/to external networks on-demand or when the network becomes available.

16 "Etcd-Io/Etcd." GitHub, github.com/etcd-io/etcd/blob/master/Documentation/op-guide/hardware.md#example-hardware-configurations.

Arm is Ideal for Kubernetes and Edge

Evolving IoT & Edge Landscape

Arm experts have predicted more than one trillion intelligent connected devices by 2035. These increasingly connected devices are driving the next generation of IoT & edge architectures. Let's look at these in detail.

IoT Devices

For the "things" within IoT, the majority of devices are based on the Arm architecture. These devices are increasingly connected to the network and providing valuable data that can be analyzed at various edge and cloud locations based on cost, network bandwidth, and latency sensitivity requirements. In addition, there is a large embedded gateway ecosystem powered by Arm based SoCs that are adopting container technologies to meet the scalability and security requirements necessary to manage these devices efficiently. For IoT developers, containers provide benefits of abstraction, efficiency, scale and automation. These devices are primarily based on the Arm Cortex-M processor family. The Arm Cortex-M processor family is optimized for cost and power-efficient microcontrollers. These processors are found in a variety of applications, including IoT, industrial and everyday consumer devices.[17]

IoT Gateways

There is a great deal of heterogeneity within the IoT space with a vast and evolving gateway ecosystem. IoT gateways are typically the first aggregation point for these increasingly connected endpoints where any kind of collation or processing of data takes place. Gateways provide the crucial intersection between the Operations Technology (OT) and IT teams in terms of end-to-end connectivity, data processing, device management and security.

Historically, the primary role of an IoT gateway was to enable multi-protocol connectivity between various types of endpoints and the cloud/datacenters. These are traditionally called M2M gateways and are used for simple data filtering and transport and perform a limited variety of functions. These devices do not host any on-device applications.

17 Arm Ltd. "Cortex-M." ArmDeveloper, 2019, developer.arm.com/ip-products/processors/cortex-m.

Due to the explosion of data from IoT devices that are increasingly connected and the increase in importance of the relevancy of such data, a new class of IoT gateways has emerged, known as Intelligent Gateways. Intelligent Gateways are general purpose in nature and can be programmed to host applications and perform localized, near-real time data processing and analytics. This is the gateway class that features a software architecture environment. In addition, there is a growing ecosystem performing Machine Learning (ML) at the edge that is driving the intelligent gateway deployments.

These gateways are traditionally based on the Arm Cortex-A, a series of applications processors and provide a range of solutions for devices undertaking complex compute tasks, such as hosting a rich Operating System (OS) platform, and supporting multiple software applications.[18]

Infrastructure Services & Edge Cloud

In order to meet the growing data processing needs within IoT there are multiple different types of edge computing architectures evolving such as:

- **Compute Edge** - The compute edge includes specialized compute processing systems located at non–data center and non–cloud locations—such as retail stores, factories, oil rigs, electricity substations—to support data processing at the edge.
- **Network Edge** - Edge and IoT use cases require connectivity from the device edge and compute edge back and forth to the cloud and datacenter environments. The network edge connects critical functions – containerization, compute, IoT, cloud, data center - at the edge of the enterprise where these processes reside.
- **Telecommunication Edge** - With the advent of 5G, telecommunication operators are driving edge architectures to meet latency and performance requirements.

18 Arm Ltd. "Cortex-A." Arm ArmDeveloper, 2019, developer.arm.com/ip-products/processors/cortex-a.

To further fuel the innovations in the infrastructure space, Arm announced in October 2018 a dedicated infrastructure roadmap with Arm Neoverse powered products enabling a diverse set of high-performance, secure, and scalable solutions required for the infrastructure foundation in a world of a trillion intelligent devices. Arm Neoverse based solutions are made possible by a broad ecosystem and span a wide range of design points from high-performance servers to power efficient edge compute platforms to gateways and WAN routers.

These different edge architectures are driving multiple industry use cases within Industrial IoT, Automotive, and Smart Cities.

Cloud Computing

Cloud computing allows the data processed from the IoT devices increased scalability, availability, performance & flexible cost-economics. The following cloud platforms provide Arm infrastructure in the cloud to ease access for IoT developers:

- AWS EC2 A1 - Amazon EC2 A1 instances are the first EC2 instances powered by AWS Graviton Processors that feature 64-bit Arm Neoverse cores and custom silicon designed by AWS. These instances deliver significant cost savings for scale-out and Arm-based applications such as web servers, containerized microservices, caching fleets, and distributed data stores that are supported by the extensive Arm ecosystem. These instances will also appeal to developers, enthusiasts, and educators across the Arm developer community. Most architecture-agnostic applications that can run on Arm cores could also benefit from A1 instances. Customers can achieve the following benefits by using these instances: For more information, please visit https://aws.amazon.com/ec2/instance-types/a1/.
- Packet - Packet has partnered with Arm to drive data center use cases by providing CI/CD infrastructure to a wide variety of software projects. Learn more here.

Security

One of the critical areas of focus for the "IoT Edge" is security against attacks such as physical attacks, communication attacks, and attacks in the software stack. For example, an IoT device needs to be secured throughout its lifecycle.

Arm Platform Security Architecture (PSA) provides a security framework that allows security to be consistently designed into a device at both the hardware and firmware level. PSA is a four-stage process, with a set of holistic deliverables to guide you through each stage. These deliverables include a set of sample threat models, security analyses, hardware and firmware architecture specifications, and an open source firmware reference implementation.[19] The fourth and final stage is PSA Certified, which currently offers certification for constrained IoT via an independent body. This allows the ecosystem to recognize whether a device is secure, without worrying about the different design patterns or implementation specifics.[20]

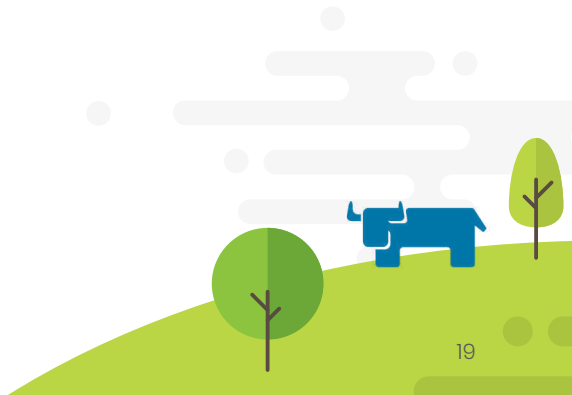
19 "TF-M DashboardActivePublicInstall Dashboard." TF-M Dashboard, 2019, developer.trustedfirmware.org/dashboard/view/5/.
20 "Arm Platform Security Architecture Overview." Arm, 2019, pages.arm.com/PSA-Building-a-secure-IoT.html.

Cluster Architectures on the Edge

The general promise of Kubernetes, regardless of whether it is in the cloud or on the edge, is to provide a common setting and consistent implementation of scalability, reliability and availability to software. The very nature of shifting these functionalities from application specific implementations to system wide utility impacts and simplifies the maintainability by reducing the complexity of an application's critical tasks - the business logic. The following chart outlines the three fundamental system architectures of Kubernetes on the edge for a high level view of cluster configurations. Each configuration type has relevance to different tactical enterprise solutions. In the next section we will provide example use cases demonstrating how each architecture is used in practice.

Cluster Architectures

	System Availability	System Reliability	System Scalability	Application Availability	Application Reliability	Application Scalability
Single Node Cluster	Low	Failover can be accomplished by restoring to another device	N/A	Low	High - System can restart the app components if necessary	Vertical
Multi Node Cluster - Single Master	Medium	Failover can be accomplished by restoring to another device in the cluster	Medium	High	High	Vertical and Horizontal
Multi Node Cluster - Multi Master	High	Failover built in	High	High	High	Vertical and Horizontal



Single Node Use Case – File Server

Scenario Description

A retailer has 500 stores. Currently, each store has a POS system which relies on large datasets that are delivered periodically to a file server sitting within each store. The server in store is the file server to the POS system and can be accessed from the retail data center as well. Datasets are updated by pushing from the datacenter to the server in store. The file server operates about a half dozen executable programs like ftp, the file server, an ETL component and a few others. The functionality provided by the server is considered non critical and short term outages are acceptable.

Goals

The current issues the retailer would like to overcome include replacing the aging hardware in store, improving the costly and time consuming process required to update software components, deliver more frequent data updates to the POS and add reliability to the applications running on the file server component. In addition, the retailer would like to enable turnkey operation of the new devices for the stores because the current process involves deploying a skilled service expert onsite to manually setup the server.

Solution

Replace the current hardware with edge servers based on 64-bit Arm SOCs such as NXP Layerscape LS2160a or NVIDIA EGX. Example devices based on these SOCs are from SolidRun or Adlink. Preload the system with k3os and a generic manifest for store setup. The store setup container has a small web server and hosts a website which the store owner is instructed to log into upon startup of the server. A few pieces of information are requested from user including the store identity. Upon filling out the form information, the server contacts the retail data center and begins configuration of the system. All of the required executable programs are deployed via Kubernetes and a few tests are run to insure proper operation. At this point, the POS system can be cutover to use the new edge device.

Analysis

Turnkey operation is achieved by providing a configuration portal available at first boot. The overall system can be monitored, managed and maintained from the data center. If at any time additional information is necessary to be collected via the initial portal, it can be updated remotely as well. There is no need for a skilled services expert for software installation, updates or management. This single node k3s cluster adds high availability for all of the runtime processes of the file server. If any of those processes should fail, k3s will take care of bringing them back up. If the hardware fails, a new device will be shipped.

Multi Node Use Case - Warehouse Management

Scenario Description

A warehouse is configured with several hundred sensors and alerting devices to manage everything from the pick and pack process to vehicle traffic and security inside and out. Sensors in place include humidity, camera, motion, temperature. Alerting devices are a variety of lights and audible forms. At this time, all the cameras are security cameras and the streams are both recorded and monitored in real time by a small team of personnel on a 24-hour rotation. Motion sensors are used as traffic alerts as well as to trigger safety alerts in high risk areas such as loading docks. Currently there is a 1-1 relationship between a sensor and an alerting device. Not all sensors cause alerts through lights or sound. Some sensors send data through a gateway for data collection and analysis offsite. The amount of sensor data transmitted is limited.

Goals

The warehouse believes they are operating efficiently within the limitations of their current systems. They also recognize that they can improve many areas by observing data in aggregate from across all areas of the warehouse. Some of the anticipated improvements include:

- Resolve traffic congestion in real time and redesign traffic flows over time.
- Manage and schedule events to reduce congestion during multiple high activity events in the warehouse.
- Improve inbound and outbound loading dock operations
- Increase utilization of bin locations, placements and vacancies.
- Enhance safety and employee satisfaction within the environment.

Solution

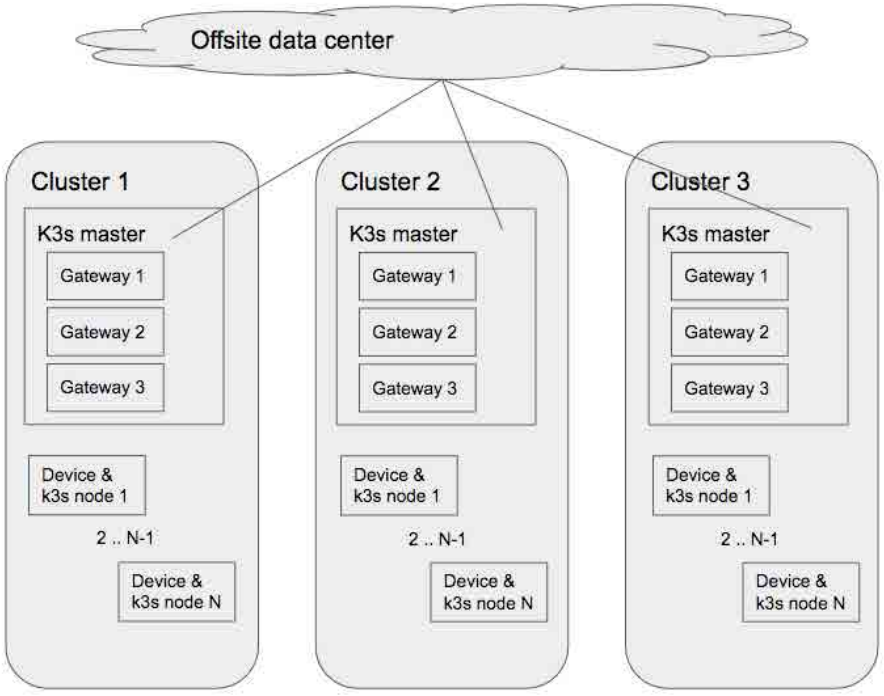
Replace current gateway devices with Hivecell. Nine gateway devices in all will be installed across 3 zones in the warehouse and serve as the masters for container orchestration. This provides one highly available (HA) Kubernetes cluster per zone. Device sensors and alerting devices will be replaced with an NXP NHS3152 NTAG SmartSensor.

In some known critical safety areas duplicate sensor devices are installed for analog redundancy. Duplicate sensors are also installed at zone boundaries. All of the sensor devices will be joined to the cluster in the proximal zone. Due to the use of the NTAG SmartSensor each sensor device is capable of performing compute without impacting the sensor operation. The three clusters are connected to a Rancher server providing visibility and management for the entire system. Most of the data analysis can now be done onsite. The gateways still interact with a remote data center, however the volume of throughput and costs associated are significantly reduced as only relevant data needs to be transmitted outside the

warehouse.

Analysis

Overall, the combined computing power of all the devices together providea foundation for onsite data analysis and processing. The software to do the analysis can be scaled horizontally and vertically across the devices in the cluster which provides a cloud like environment available to the warehouse operations team. Warehouse operations now has the ability to monitor and act on activity based on the feedback from multiple devices simultaneously. Live feedback devices can be operated in tandem based on information from multiple sensors, not just one. The entire system is highly redundant and provides an in place data center for real time processing of the information collected. The relevant aggregated and post processed data can be sent to the offsite data center for further analysis where heatmaps of the entire warehouse operation can be generated and analyzed by humans and AI. Each zone is highly available and devices can be managed without impacting any of the other operations. Physical devices can be removed and replaced or new devices may be added and the system will continue to function. Similarly, the software and drivers for each device can be updated and managed remotely. In the future devices can easily be added to the system providing new capabilities such as machine learning.



Tactical Considerations Based on Cluster Configurations

	Distributed compute at the edge	Remote sites which must be highly reliable	Simplified hardware and installation	Remote software management	Function disconnected from other networks	Utilize excess capacity on existing hardware
Single Node Cluster	No	No	Yes	Yes	Yes	Yes
Multi Node Cluster - Single Master	Yes	No	Yes	Yes	Yes	Yes
Multi Node Cluster - Multi Master	Yes	Yes	Yes	Yes	Yes	Yes

Conclusion

K3s is Kubernetes for the Edge

Containers and container orchestration are emerging as a key enabler of delivering modern applications in the most successful enterprises. Worldwide, Kubernetes is being adopted as the container orchestration tool for digital transformation, regardless of industry.[21] Kubernetes fundamentally solves many of the problems and addresses the technical needs and operational concerns at the edge. However, Kubernetes in it's upstream form is too heavy weight and operationally challenging for the edge case. Fortunately, k3s solves these issues and delivers all the value of Kubernetes in a lightweight, easy to operate package which is ideal for accelerating and delivering solutions at the edge.

Arm is Ideal for k3s Container Orchestration on the Edge

The combination of edge optimized hardware running Arm processors, and container orchestration delivers a strong value proposition that should not be overlooked when developing the current and next generation edge. Together this combination of hardware and software delivers robust, scalable, reliable and highly available solutions. Software is decoupled from the device and simultaneously has the ability to take advantage of each hardware component as if it were purpose built for the task. This combined hardware / software strategy is also beneficial for long term usage. Containers provide isolation and security in a purpose built package and the underlying container orchestration gives physical devices fungibility to handle scenarios on the edge which have not yet been conceived.

Arm plus k3s is a clear strategy for success spanning needs from edge to cloud. Arm plus k3s has the ability to handle today's needs and the unknown requirements of the future.

21 Asay, Matt. "How Hot Is Kubernetes? Even Traditional Banks Are Transforming to Embrace It." TechRepublic, TechRepublic, 7 Aug. 2019, www.techrepublic.com/article/how-hot-is-kubernetes-even-traditional-banks-are-transforming-to-embrace-it/.+

Appendix A – Installation of k3s

Appendix A - Installation of k3s

K3s can be installed on any Arm v7 or v8 system runningLinux. Installation of k3s is as easy as running a script or downloading a single binary. K3s is open sourced under the Apache 2 license. Detailed install and configuration options, including air gap instruction, multi node clusters, cluster configuration options and more are available on the k3s github page.[22]

In addition to operating in the Linux OS on Arm, k3s also comes preinstalled on Rancher's container operating system k3os.[23] k3os can also be run on Arm devices.

Init script - quick install

Installation via the script at the get.k3s.io url will install the binary and setup an init.d script so that k3s can be started and stopped using the systemctl utility. The script can be executed and installed via a single command line entry. The following command will probe for the system architecture and install the latest k3s version.

```
curl -sfL https://get.k3s.io | sh -
```

A single node Kubernetes cluster will be automatically started when the script is finished.

Binary install

Installation of the binary requires prior knowledge of the underlying architecture and the version of k3s which is to be installed. As an example, the following line will download v0.9.1 of k3s for the Arm64 architecture.

```
curl -L -o k3s https://github.com/rancher/k3s/releases/
download/ v0.9.1/k3s-arm64
```

Starting the cluster operations is as simple as running the executable

```
./k3s-arm64
```

22 "Rancher/k3s." GitHub, 21 Jun. 2019, github.com/rancher/k3s.
23 "Rancher/k3os." GitHub, 26 June 2019, github.com/rancher/k3os.

Additional information

K3s docs site - https://rancher.com/docs/k3s/latest/en/
K3s github page (source code) - https://github.com/rancher/k3s
K3s landing page - https://k3s.io/

References

"Etcd-Io/Etcd." GitHub, github.com/etcd-io/etcd/blob/master/Documentation/op-guide/hardware.md#example-hardware-configurations.

"Rancher/k3s." GitHub, 21 Jun. 2019, github.com/rancher/k3s.

"Rancher/k3os." GitHub, 26 June 2019, github.com/rancher/k3os.

"Production-Grade Container Orchestration." Kubernetes, 2019, kubernetes.io/.

Debian. "Chapter 3. The System Initialization." Chapter 3. The System Initialization, 2019, www.debian.org/doc/manuals/debian-reference/ch03.en.html#_the_kernel_module_initialization.

Arm Ltd. "Cortex-A." Arm Developer, 2019, developer.arm.com/ip-products/processors/cortex-a.

Arm Ltd. "Cortex-M." Arm Developer, 2019, developer.arm.com/ip-products/processors/cortex-m.

Arm Ltd. "Neoverse N1." Arm, 2019, www.arm.com/products/silicon-ip-cpu/neoverse/neoverse-n1.

"Arm Platform Security Architecture Overview." Arm, 2019, pages.arm.com/PSA-Building-a-secure-IoT.html.

Asay, Matt. "How Hot Is Kubernetes? Even Traditional Banks Are Transforming to Embrace It." TechRepublic, TechRepublic, 7 Aug. 2019, www.techrepublic.com/article/how-hot-is-kubernetes-even-traditional-banks-are-transforming-to-embrace-it/.

Barnard, Kaitlyn. "CNCF Survey: Use of Cloud Native Technologies in Production Has Grown Over 200%." https://www.cncf.io, 29 Aug. 2018, www.cncf.io/blog/2018/08/29/cncf-survey-use-of-cloud-native-technologies-in-production-has-grown-over-200-percent/.

Bauer, Roderick. "What's the Diff: VMs vs Containers." https://www.backblaze.com, 28 June 2018, www.backblaze.com/blog/vm-vs-containers/.

"Case Studies." Kubernetes, 2019, kubernetes.io/case-studies/.

"Code Reuse." Wikipedia, Wikimedia Foundation, 22 Feb. 2019, en.wikipedia.org/wiki/Code_reuse.

"Comparison of Container Operating Systems." Rancher Labs, 16 Apr. 2019, rancher.com/blog/2019/comparison-of-container-operating-systems/.

DeMuro, Jonas. "What Is Fog Computing?" TechRadar, TechRadar pro IT Insights for Business, 13 Aug. 2018, www.techradar.com/news/what-is-fog-computing.

Dvoretskyi , Ihor, and Chris Aniszczyk. “Cncf/Foundation.” GitHub, 2015, github.com/cncf/foundation/blob/master/charter.md.

Foote, Keith D. “A Brief History of Cloud Computing.” DATAVERSITY, 22 June 2017, www.dataversity.net/brief-history-cloud-computing/.

Garrison, Justin. “No SDN Kubernetes.” Medium, Medium, 26 Sept. 2016, medium.com/@rothgar/no-sdn-kubernetes-5a0cb32070dd.

Gill, Bob, and David Smith. “The Edge Completes the Cloud: A Gartner Trend Insight Report.” Gartner, 14 Sept. 2018, www.gartner.com/document/3889058.

Goddard, William. “The Evolution of Cloud Computing – Where's It Going Next?” ITChronicles, 18 Dec. 2018, www.itchronicles.com/cloud/the-evolution-of-cloud-computing-wheres-it-going-next/.

Greenfield, David. “Fog Computing vs. Edge Computing: What's the Difference?” Fog Computing vs. Edge Computing: What's the Difference? | Automation World, 2 Aug. 2016, www.automationworld.com/fog-computing-vs-edge-computing-whats-difference.

Gupta, Anurag. “Why Is Kubernetes so Popular?” Opensource.com, Oct. 2017, opensource.com/article/17/10/why-kubernetes-so-popular.

Hightower, Kelsey. “Kelseyhightower/Kubernetes-the-Hard-Way.” GitHub, 30 Sept. 2018, github.com/kelseyhightower/kubernetes-the-hard-way.

Hobbs, Andrew. “IoT: Understanding the Shift from Cloud to Edge Computing.” Internet of Business, 23 Aug. 2018, internetofbusiness.com/shift-from-cloud-to-edge-computing/.

Johnson, Brad. “https://blog.swim.ai.” https://blog.swim.ai, 1 Dec. 2017, blog.swim.ai/2017/differentiate-edge-computing-part1.

Kotovs, Vladimirs. “Forty Years of Software Reuse.” Scientific Journal of Riga Technical University. Computer Sciences, vol. 38, no. 38, 2009, pp. 153–160., doi:10.2478/v10143-009-0013-y.

Luan, Tom H, et al. “Fog Computing: Focusing on Mobile Users at the Edge.” ArXiv.org, 30 Mar. 2016, arxiv.org/abs/1502.01815.

Osnat, Rani. “A Brief History of Containers: From the 1970s to 2017.” https://blog.aquasec.com, 21 Mar. 2018, blog.aquasec.com/a-brief-history-of-containers-from-1970s-chroot-to-docker-2016.

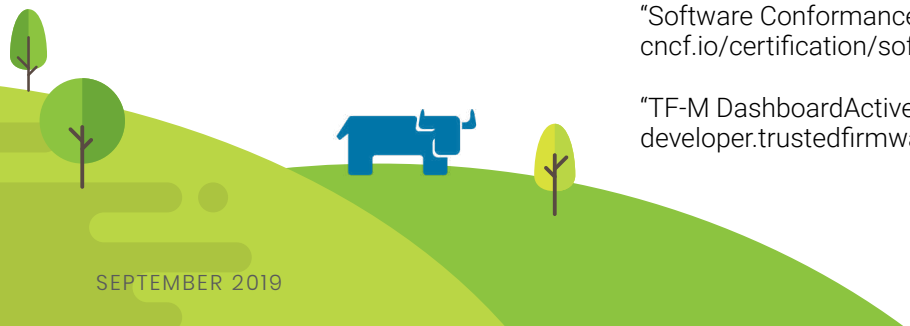
“PSA Certified.” PSA Certified, 2019, psacertified.org/.

Ready, Jeff. “How Edge Computing Is Impacting Manufacturing.” Manufacturing.net, 4 Oct. 2018, www.manufacturing.net/article/2018/10/how-edge-computing-impacting-manufacturing.

Seshachala, Sudhi. “Docker vs VMs.” https://devops.com, 24 Nov. 2014, devops.com/docker-vs-vms/.

“Software Conformance.” Cloud Native Computing Foundation, 2019, www.cncf.io/certification/software-conformance/.

“TF-M DashboardActivePublicInstall Dashboard.” TF-M Dashboard, 2019, developer.trustedfirmware.org/dashboard/view/5/.



About Rancher

Rancher Labs delivers open source software that enables organizations to deploy and manage Kubernetes at scale, on any infrastructure across the data center, cloud, branch offices, and the network edge. With 27,000 active users and greater than 100M downloads, their flagship Rancher product is the industry's most widely adopted Kubernetes management platform. For additional information, visit www.rancher.com and follow @rancher-bot_Labs on twitter.

For more information about ARM or other public reference customers in production with Rancher, please contact us at **sales@rancher.com.**

About Arm

Arm is world's leading semiconductor IP company with Arm technology reaching 70% of the global population. Arm defines the pervasive computing that's shaping today's connected world. Realized in 125+ billion silicon chips, our device architectures orchestrate the performance of the technology that's transforming our lives — from smartphones to supercomputers, from medical instruments to agricultural sensors, and from base stations to servers. For additional information, visit https://www.arm.com/company.