IoT and Edge computing with Kubernetes

White paper

By Dejan Bosanac, Cindy Xing, Preston Holmes, …

Status: 15/03/2019 - Draft

Goals

Cloud native computing paradigm is the main focus of many development teams these days and there's a good reason for that. Concepts like microservices and devops, to name just a few, radically changed how teams approach building applications.

But there's another class of applications that on the first look don't belong to this world. IoT and Edge applications have a lot of distributed components that don't usually sit together within the same data center infrastructure. On the other hand developers of those applications would benefit greatly from the concepts, infrastructure and tools that are being developed in the cloud-native universe.

The goal of this document is to bridge the two worlds together and lay the foundations for future work. In order to do that, we need to start by defining the **vocabulary of common terms** and make sure we understand each other.

Next, we can start **defining basic concepts, use cases and architectures** used today in this space. Note that we will not try to go deeper into general IoT and Edge computing discussions as there are excellent papers that already cover these topics in detail and you can find them within the references.

Finally, we can focus on Kubernetes as a platform for developing IoT and Edge applications. We'll try to define **layout of Kubernetes-managed clusters** for some of the identified use cases. And also describe challenges that exist today for deploying some of the workload and use cases.

Hopefully, this will give us enough **foundation for the future work** in our overall goal to improve document and improve Kubernetes to be best possible platform for IoT and Edge computing. But also, start new projects where necessary to bring cloud-native and edge computing closer together.

What is Edge

When discussing IoT and Edge computing in various environments, you soon realize that these terms mean different things to different people. So let’s spend a bit of time defining the terms in the context we’re going to use them.

By **IoT** (Internet of Things) we’ll be talking mostly about **connecting devices to the cloud**. These devices are source of data allowing us to sense **the environment**. But they are also data receivers as they **control that same environment** by receiving commands from cloud (or local) applications. Typical examples of IoT solutions are next generation factory automation, usually called Industrial IoT (or IIoT), smart homes, smart cities, etc. We will describe all the concepts relevant to IoT in much more detail in the coming sections, but in general we’ll be focusing on how we can improve the deployment and operation of IoT systems using Kubernetes.

Edge computing is an even broader term and is in its most general context related to **bringing computer resources closer to the end users and devices**. For example, in the IoT solutions, devices are usually geographically dispersed, so it might be beneficial for many reasons (as we’ll discuss soon) to have data processed as close as possible to the exact location of the device. There are many other interesting non-IoT related Edge applications, like low-latency game servers and such. The main theme here, however, is that the Edge is a computer resource more geographically distributed from the main cloud datacenter. So the main challenge is how we can extend the resource management systems used in the cloud to manage edge resources as well. Our goal is to describe all the concepts relevant to Edge computing today in this context and see how Kubernetes can be used (and to what extent) to address anticipated challenges.

Why Edge

As we mentioned earlier, there are multiple reasons why one might want to run computing workloads on the edge. In this section we will attempt to list the key motivations. Depending on the use case, one or more of these motivations may apply at once, and they might be ranked in importance differently.

One of the crucial factors that makes Edge computing different than the pure Cloud based solutions is minimizing **network traffic over larger distances between devices (users) and the workloads.** This affects data processing in multiple ways and could be considered as an explicit benefit of the Edge computing.

First, we add **reliability** to the system as it can survive and operate even if the network is down. This is important for many use cases where edge devices are hard to connect using reliable network connections. In these scenarios, the network connectivity is usually also expensive, so having all data transferred from edge locations to the cloud is not always cost-effective or even achievable. So having data (pre)processed at the edge node can help us **deal with bandwidth issues and avoid network saturation**.

The value of **Edge data (pre)processing** is not only limited to bandwidth, but can also help to reduce overall **latency** of the system, which is one of the main drivers of Edge computing for many applications, like VR or industrial applications dealing with critical infrastructure for example. These use cases can greatly benefit from reducing latency by pushing computation closer to the devices and/or users.

Besides pure processing, **data locality** is also an important aspect to consider. With Edge computing architectures, we can enforce **policies** where sensitive data may never leave the physical environment. You can think of GDPR and other data protection and privacy policies as good candidates to be tackled in this manner for some applications. Through enforcing policies on the edge we should be able to provide a **better security** of the whole system as well, by adding a protection layer closer to the originating source of possible attacks (devices).

Finally, we can discuss how edge systems can help us **scale** beyond centralized datacenter. In some situations it might be easier (and more effective) to add more resources on the edge locations. We can also use hardware that is better suited to the particular needs of the solution and create systems that **integrate** better with on-premise systems. All these aspects can also evidently affect the **total cost** of deployed solution.

There’s no doubt that all these aspects can improve many solutions that exists today, but certainly we expect that the whole field will be evolving over time and that new aspects of Edge computing will be available in the future.

Edge Resource Categories

"Edge" is used to describe multiple shapes and sizes of resource capacity and location. This section is an attempt to disambiguate "what" edge is. We’ll go through different categories, explain the environment and types of nodes you can typically find in those environments.

But before we dive deeper, let’s first focus on one component that is a crucial part of IoT applications, but still something we’re not considering as being a managed part of the Edge: a constrained device.

Constrained Device

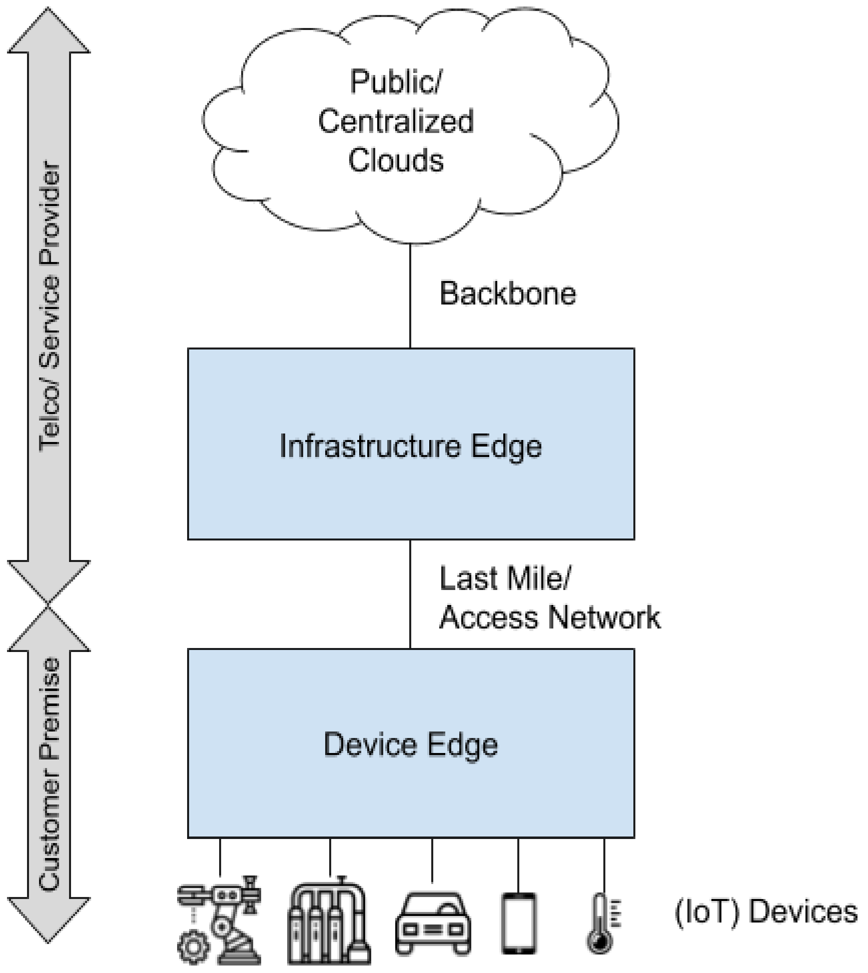
Elementary components of the IoT are the “things” or the devices. They serve two main purposes in the systems: sensing and actuating. In a **sensor** role, a device is sensing its environment (e.g. temperature sensor) and sending that data to the cloud (or other edge components as we’ll soon discuss). In an **actuator** role, it’s controlling an environment (controlling AC for example)according to the commands received from the other system components. Devices are usually both sensors and actuators at the same time with the corresponding controller logic.

Although the term device may vary (as we will see in a minute), the most commonly used “things” have some common technical properties. They are defined by their main properties: a need to be **deployable everywhere** and inexpensive for **massive scale**. This implies that they are **often battery powered,** which further dictates their other technical aspects. This usually implies that they are very **limited in the computing power**, not to mention lack of other resources like storage for example. Additionally, there are implications on the networking layer and patterns how these devices communicate with the rest of the system. Often, they are **not IP-enabled** and communicate only using the **near-range field protocols**, like BLE or ZigBee. They are often operating in the sleep mode, waking up occasionally to exchange messages with the cloud systems and perform necessary tasks.

From the above discussion, we can see why constrained devices are not considered a managed part of the edge, but more as endpoints connecting to the managed edge infrastructure. They are not suitable for containerization, not part of the IT (or even OT environments) and **cannot be controlled or managed by the clusters**.

Note, however, that there may be and usually are devices which have the same or similar functionality as constrained devices, but which more resource rich. We call these devices smart devices (see section below), and they can be part of the managed edge.

Now let’s go up one level and discuss different type of edge components. We divided these components into two broad categories: Device edge and Infrastructure edge. Both edge domains are on either side of the last mile and hence close to the IoT devices and/or end users. The biggest difference is that the infrastructure edge is usually owned and operated by an operator (telco or service provider), while the device edge is located on the customer premise. The IoT devices are usually part of the device edge (smart devices) or directly connected to it (constrained devices).



Source: Icons from https://www.flaticon.com/free-icon

Device Edge

A key feature about device edge scenarios is that they are in a physical environment that is not typically managed and maintained as an IT focused space. Data centers and Urban interconnect/POP locations have extensive investment in things like physical security, power redundancy and backup, multiple network links, and high quality network. IoT environments are those which may have **never been seen as compute infrastructure before**, or have been **managed by OT divisions**, not IT and may be less able to provide the same environmental standards. Examples of device edge environments are factories, oil fields, trains or trucks and similar.

One of the important aspects of this environment is **network connectivity**. Network connections are many times **limited** in availability and/or bandwidth. Often, they come with **premium costs**. Nodes running there should at least be able to **work offline, sometimes even for a considerable amount of time**.

Now let's take a look at the type of nodes found in the device edge environments.

Smart devices

Smart devices are more traditional computing resources, like mobile phones or tablets for example. They are, of course, capable of running all kind of **complex applications**. They are capable of running different kind of workloads and **connecting** to the rest of the infrastructure **directly**. These are well understood devices widely used, so there's no need to explain them in much more details.

IoT Gateway

One more interesting "specialized" node found on the device edge is IoT gateway. It's main purpose is to be an **aggregation point for the constrained devices**. What does that mean? First and foremost, the job of the gateway is to do an **IP-onboarding** of the sensors and actuators. As we said earlier, constrained devices are often using near-field protocols and gateways bridge the communication between them and the cloud platform. They use some of the "cloud protocols" to do that and we'll cover them in more details in a bit. Additionally they can run some of the **workloads to transform, normalize or aggregate data** sent to the cloud. As gateways run in the device edge environment all aspects of the connectivity applies here and that affects both workloads running on them and how they are managed by the cloud platforms.

Gateways are usually accompanied by the **IoT cloud platform**, which **manages** the gateway itself and devices connecting to it. There's a growing interest in **containerizing** IoT gateways and managing them as cluster nodes. One of the main challenges in doing so is the aforementioned network connectivity. We'll discuss this issue more throughout the document.

Edge node

- Alternative name(s): Edge gateway, IoT Edge System
- More powerful gateways that can do data storing and processing
- Not enough capacity to run a full Kubernetes control plane, but enough to run several containers
- More heavy applications are moved from cloud to the device edge nodes, like
  - Data storage
  - Stream analytics
  - Business process management
- Benefits
  - Reduce traffic between gateway and cloud (send only important events)
  - Make local decisions
  - More functionality available in offline mode

Edge "Cluster"

- Enough compute capacity to run a full Kubernetes control plane
- exists in the IoT or OT environment
- may consist of heterogeneous node types
  - mixed x86 and ARM architecture
  - diverse HW capabilities (GPU, SSD, etc)
  - different classes of physical security
  - nodes may also be running other workloads

Infrastructure Edge

Network Edge

- Alternative name(s): Edge cloudlets
- Reliable and low latency connection to cloud platforms or datacenter
- limited level of compute compared to datacenter-class infrastructure
- focus is on latency improvements as network edges tend to be distributed with populations
- Infrastructure may already exist for POP, or CDN point
- Typically an IT managed physical environment
- May perform workloads that have no direct connection to the physical world and are therefore not "IoT" (eg low latency game server)

	Reliability	Bandwidth	Policy	Latency
Device	-	-	-	-
Gateway	Light/Medium	Light/Medium	-	-
Edge node	-	Light/Medium	Medium/Heavy	-
Network edge	-	-	Medium/Heavy	Heavy

Edge Workloads

Traffic Shaping

What is it?

Traffic shaping includes tasks of managing bandwidth usage for ingress and egress between pods and between the cluster and other networks. For IoT edge environments the network link may be more limited (think of SMB asymmetric cable modem, or cellular connectivity). The kubernetes cluster may share that link with other machines or processes, and so may need to be allocated only a portion. Within that portion of network bandwidth, different workloads running in Kubernetes may be more critical and so may be considered as running at different priorities.

Where is Kubernetes involved?

Need a cluster-wide or site-wide setting for traffic management

- policy is more about access and rights than resource usage
    - https://kubernetes.io/docs/concepts/services-networking/network-policies/
- sounds like in kubelet
    - https://github.com/kubernetes/kubernetes/issues/2856
    - https://godoc.org/k8s.io/kubernetes/pkg/util/bandwidth
- https://github.com/kubernetes/kubernetes/issues/11965

Protocol Conversion

What is it?

Edge environments are full of a great many diverse protocols, often having been developed from niche industries, or legacy and proprietary contexts. "Protocol" may apply to the content and shape of data, or to the wire-level format. Many industrial protocols are conveyed on non-IP serial busses, and other modern RF protocols may also be non-IP (e.g. Bluetooth). Strictly speaking this is a preamble or subset of "data processing" but is common enough to highlight it as a specific workload.

Where is Kubernetes involved?

For data-content conversions, this is just logic that can run in containers, and Kubernetes is not concerned. For network level protocol conversion, this often requires privileged containers (root) to access hardware interfaces. Node taints and tolerances may be used to represent the presence of certain interfaces. Conventions may be established around exposing these HW interfaces through conventions of certain service annotations to be used by unprivileged containers.

Workload prioritization

What is it?

Edge clusters are more likely to be composed of a limited number of nodes, and without any benefit of cluster auto-scaling. Edge environments also may be more likely to have a great range in workload priority with several critical workloads and other much lower priority. This condition is not unique to edge-clusters, but will have a higher probability of being a concern given the typically smaller cluster footprint.

Where is Kubernetes involved?

For workloads there could be interaction of:

- priority: ~high
- qos: best effort
- tolerance: PreferNoSchedule

See also WG on oversubscription

Buffer and Batch (aka store and forward)

What is it?

An edge cluster may serve as a gateway sitting between data generating local devices, and cloud data sinks. The connectivity between the cluster and the cloud may be unreliable and intermittent. Devices generating data may have limited logic and memory as far as tracking eventual delivery of their payloads with retry-on-fail.

Where is Kubernetes involved?

<workload template>

What is it?

Where is Kubernetes involved?

<workload template>

What is it?

Where is Kubernetes involved?

<workload template>

What is it?

Where is Kubernetes involved?

<workload template>

What is it?

Where is Kubernetes involved?

Workload	Aspects for Kubernetes
Protocol conversion ● Data ○ industrial protocol -> general structured message ● Network ○ non-IP protocol (modbus) to TCP/IP based protocol	● Requirement to run privileged containers to access HW peripherals ● Convention around exposing these to less privileged containers via broker or service
Data Processing (Data streaming) ● converting	● largely relate to data structure standards - outside of Kubernetes scope

• packing • validating • combining • enhancing	• Projects like LWM2M, vorto, etc
Traffic shaping • bandwidth management • Network QOS or prioritization	Need a cluster-wide or site-wide setting for traffic management • policy is more about access and rights than resource usage ○ https://kubernetes.io/docs/concepts/services-networking/network-policies/ • sounds like in kubelet ○ https://github.com/kubernetes/kubernetes/issues/2856 ○ https://godoc.org/k8s.io/kubernetes/pkg/util/bandwidth • https://github.com/kubernetes/kubernetes/issues/11965
Workload prioritization	For workloads there could be interaction of: • priority: ~high • qos: best effort • tolerance: PreferNoSchedule See also WG on oversubscription
Logical device composition • sensor fusion • combine multiple sources of data from edge into a single semantic model	
• **Identity Mapping** ○ Broker local only identifier to global identity	• Role of Kubernetes service accounts?
• **Security upgrade** ○ take no or low security local communications and provide enhanced security (identity and transport) to remote/cloud destination.	

● **Caching (e.g., CDN/UDN)** ● **Local caching of cloud/remote state** ○ different edge nodes can maintain caches of data for local consumers ○ manage sync between cloud and edge, or between edges	● potentially istio sidecar to handle some caching?
Buffer and Batch (aka store and forward)	● Edge Kubernetes less likely to have access to extensive storage services
Local reactive functions ● Functions run locally in response to events ● events may be schedules or sensor based ● see: https://github.com/cncf/wg-serverless/tree/master/proposals/open-events	https://github.com/knative/
Machine Learning on the edge ● inference on the edge with cloud trained models ● training on edge for specific adaptation to the local environment, or for privacy preservation	● How would an event system link to a generic ML runtime ● Potential to use Taints and Tolerances to ensure ML workloads run on more capable node (one with GPU)
Compute offload ● For example, real-time server side rendering of AR/VR visuals.	● potentially similar to ML - taints to ensure GPU runs those workloads
Application deployment and management	● Application can run as container or function ● Do we want to support process when the edge node is resource constraint

Light

Cloud platforms

IoT Cloud platform

- Enable connectivity of large number of devices, gateways and edge nodes
- Ingest large amount of telemetry data and normalize it for cloud application usage
- Enable the addressing of devices for sending commands back to devices
- Provide device registry for identifying and securing devices
- Provide device management for provisioning and managing gateways
- Provide services for enabling of IoT application development (data storage, event management, analytics, API management, ...)
- Enable application, configuration deployment & orchestration from cloud to edge

TODO define Edge cloud in more details?

Communication protocols

- Wide variety of protocols in use for connecting to cloud platforms
- Two main categories
  - messaging-based protocols
  - Request-reply based protocols
- TLS is often used but not supported in all use-cases

Request-reply protocols

- Used by more constrained devices
- Short-lived connections
- Usually device is in the sleep mode, wakes up, establish the connection, do the work and go back to sleep
- A lot of usage of UDP and other “compact binary HTTP” alternatives
- Most used
  - CoAP
  - OPC-UA
  - HTTP
  - HTTP/2

Messaging protocols

- Used by more powerful devices and edge nodes that are usually not battery powered and gateways
- Maintain permanent connection to the cloud

- Provide traditional messaging communication patterns like, publish-subscribe
- Most used
    - MQTT
    - AMQP 1.0
    - OPC-UA

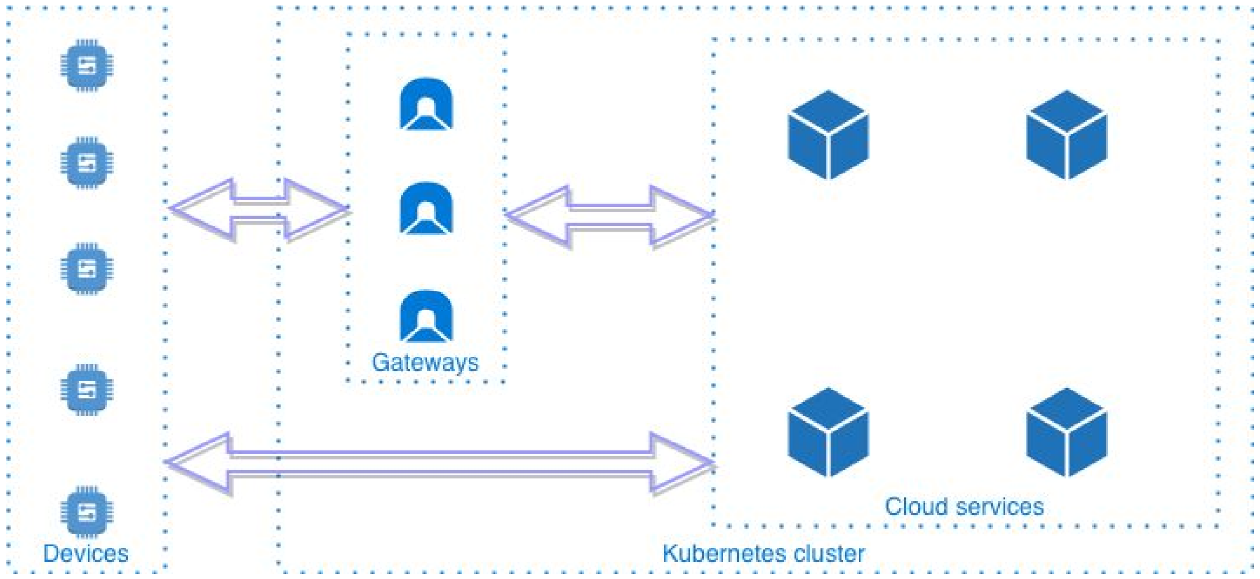
IoT and Edge architectures with Kubernetes

Now that we defined basic concepts and components of IoT and Edge solutions, we can talk about different architectures and how (if at all) we can deploy them in the Kubernetes environment. We will not be able to cover all use cases here, but just to give some examples for the prominent examples. Through this exercise we will just open topics for discussion and dig deeper into individual use cases in other working group efforts.

Industrial IoT

In industrial IoT use cases we have a requirement for an IoT platform to support a very large number of devices connecting to it. These devices can connect **directly** or via **IoT gateways**. Of course, in such an environment not all devices and gateway belong to a single tenant, but still the platform as a whole needs to be able to handle **thousands to millions devices connected** to it.

The typical example of these deployments are geographically hard to reach locations, like oil fields.



Constrained devices that are IP-enabled often connects directly to the cloud using some of the request-reply protocols. The main challenge for Kubernetes based clouds in this scenario is to provide an **ingress layer** that is well suited for the cloud protocols and communication patterns described in the previous sections. We need to be able to route connections to the services at scale for starters. Additionally, Kubernetes ingress layer is today primarily HTTP TLS centric. In order to support wide range of devices, it needs to have first-class **support for industrial IoT protocols based on both TCP and UDP** underlying network protocols.

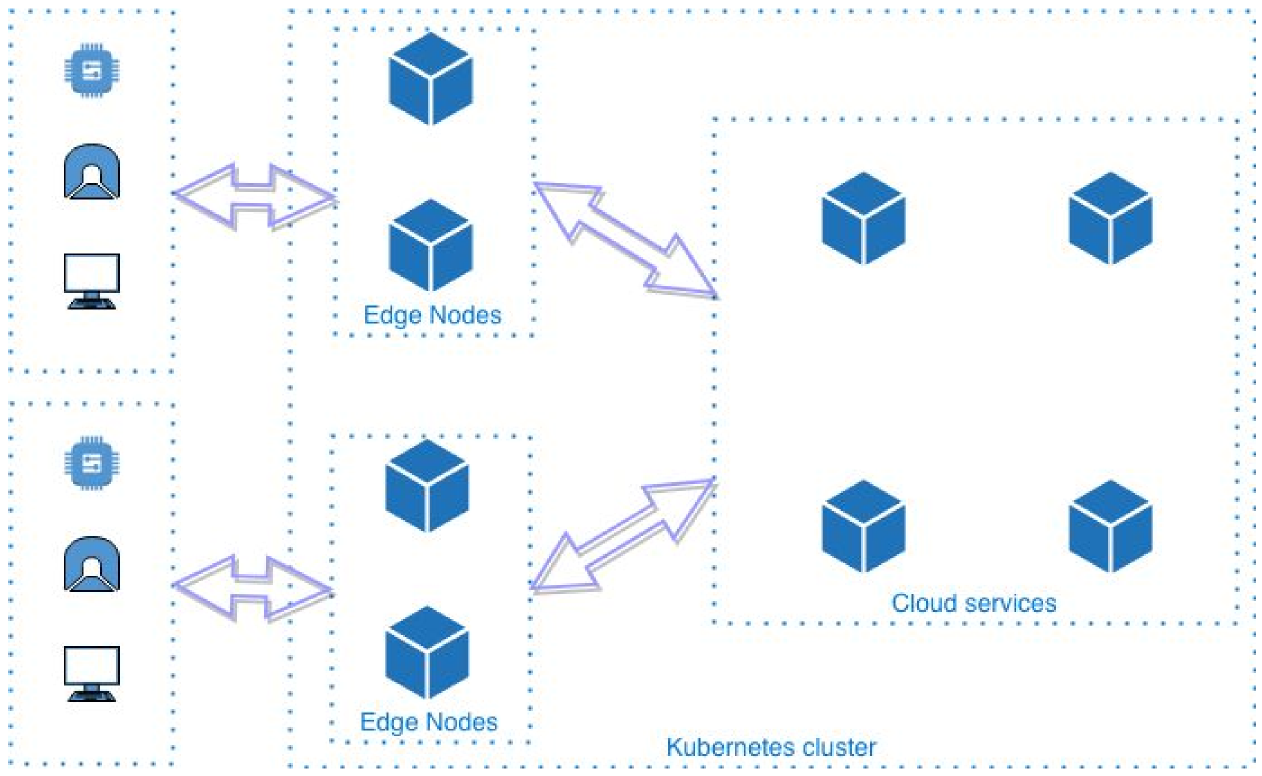
Constrained devices that are not IP-enabled are connected to the IoT gateways, which provides IP-onboarding and try to keep permanent connection with the cloud and using messaging based protocols to exchange data with the cloud. Even if this is the case, the **connection** between gateways and cloud is often **unreliable and expensive**. The same challenges for ingress remain; a better support for message messaging oriented protocols and **scaling to millions of connections** are prerequisite for successful deployments.

Besides connectivity, IoT gateways can run **workloads** to improve data exchanged with the cloud. They are usually limited to **simpler tasks**, like protocol conversion, data preprocessing, identity mapping, security upgrade and such.

There's a **growing interest in containerizing IoT gateways** and managing them using Kubernetes cluster control plane. However, **limited resources and unreliable constrained networks provide a challenge** for doing this with the current set of tools. This is the subject of research currently using technologies such as Virtual Kubelets and it is expected to be further researched in the future.

Edge applications

As we mentioned earlier, Edge nodes brings **more resources (compute and storage) to the edge** than IoT gateways. Also, they are typically deployed to locations with better infrastructure **more reliable network**. One such example could be servers at retail locations.



These conditions allow us to start deploying applications closer to the users (and devices) and start providing benefits of Edge architectures, like reduced latency or bandwidth. One of the example could be deploying **data streaming** applications to the Edge nodes in order to reduce traffic and save bandwidth between devices and the central cloud. Or going even further, deploying a **serverless framework** for using local functions that can be triggered as a response to certain events (without communication with the cloud).

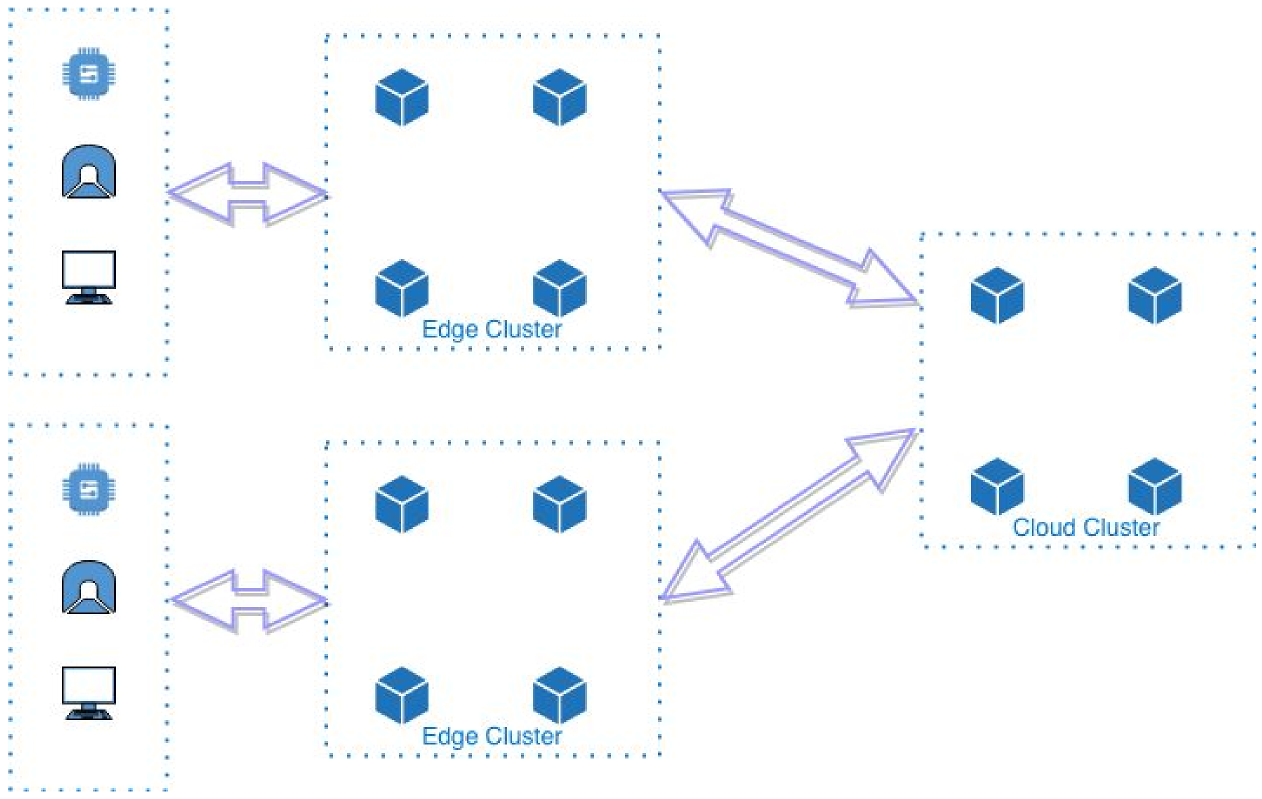
These type of Edge nodes could be used also to cache some of the centralized state and store and forward data produced locally. In this manner, it should be able to provide **uninterrupted functioning** of the edge location in the events of **network failures**.

As Edge nodes are resourceful, there's no issues in making them a part of the Kubernetes cluster in that sense. However, in these cases not all nodes are equal so there needs to be a special consideration in deploying services to proper nodes. Concepts like **affinities, taints and tolerances** could be very useful in this context. This is the area to that could be researched further.

For this kind of use case, KubeEdge (http://kubeedge.io) is an example architecture. It allows customers to enroll edge nodes to a cluster managed at cloud side; and then deploy, run and life cycle management of these edge applications locally.

Edge cluster applications

Expanding further on the Edge applications, deploying hardware capable of **running clusters locally**, opens door to many new use cases. These clusters can be part of the device or infrastructure Edge as it was explained previously.



With enough computing power and storage, it can be possible to do **machine learning** and **compute offload** on the Edge. This means that we can enjoy the benefits of Edge architecture even for demanding applications like VR/AR or gaming.

In these cases, we should investigate if the current **Kubernetes Federation** feature is suitable to support these use cases or is there any further improvements or concepts need to be developed to empower engineers developing these types of applications.

Mixed and hybrid architectures

As already discussed, this is not an attempt to cover and discuss in details all possible cases that are considered “IoT” or “Edge”. Instead we’re trying to showcase a couple of use cases we can use for later work. Of course, real world tends to be complicated so the solutions will in most cases use mixed architectures. Also, in some cases we can have nodes that are not uniformed in terms of hardware architectures. These are all considerations we need take care of in order to create a really useful platform for developing IoT and Edge solutions.

Lifecycle

TODO / Optional

- lifecycle of cluster
- lifecycle of individual workloads
- operational concerns

Conclusion and future steps

TODO

References

- Cloud Edge Computing: Beyond the datacenter - https://www.openstack.org/assets/edge/OpenStack-EdgeWhitepaper-v3-online.pdf
- The Three Software Stacks Required for IoT Architectures - https://iot.eclipse.org/white-paper-iot-architectures
- AT&T Edge Cloud - http://about.att.com/content/dam/innovationdocs/Edge_Compute_White_Paper%20FINAL2.pdf
- State of the Edge Report - https://www.stateoftheedge.com/
- Open Glossary of Edge Computing - https://www.stateoftheedge.com/glossary
- Related K8s SIGS
  - https://github.com/kubernetes/community/tree/master/sig-multicluster
  - https://github.com/kubernetes/community/tree/master/sig-cluster-lifecycle
  - https://github.com/kubernetes/community/tree/master/sig-scheduling
    - for resource oversubscription

Interested Parties

- Volterra Edge Services (VES), Marco Rodrigues <marco@ves.io>
- Volterra Edge Services (VES), Pranav Dharwadkar <pranav@ves.io>
- Volterra Edge Services (VES), Jakub Pavlik <jpavlik@ves.io>
- Rafay Systems: Robbie Gill <robbie@rafay.co>; John Dilley <jad@rafay.co>; Haseeb Budhani <haseeb@rafay.co>; Sarat Chandra <sarat@rafay.co>Red Hat, Dejan Bosanac <dbosanac@redhat.com>
- Red Hat, Miki Kenneth <mkenneth@redhat.com>
- Red Hat, Frank Zdarsky <fzdarsky@redhat.com>
- Red Hat, Pasi Vaananen <pvaanane@redhat.com>
- Red Hat, David Bericat <david.bericat@redhat.com>
- Eclipse Foundation, Benjamin Cabé <benjamin.cabe@eclipse-foundation.org>

- Siemens: Harald Mueller <h.mueller@siemens.com>
- InfluxData, David Simmons <davidgs@influxdata.com>
- SAP, Erich Clauer <erich.clauer@sap.com>
- VMware, Tiejun Chen <tiejunc@vmware.com>; Steven Wong <wongsteven@vmware.com>; Fangyuan Li<fangyuanl@vmware.com>; Xiaoran Zhan<lzhan@vmware.com>; Hanlin Shi<hanlins@vmware.com>;Yixing Jia<yixingj@vmwar.com> ; Malini Bhandaru <mbhandaru@vmware.com> ; Jed Salazar <jsalazar@vmware.com>; Peter Grant <pegrant@vmware.com> Tunde Olu-Isa < oluisat@vmware.com>
- Christopher M Luciano <cmluciano@cruznet.org>
- Vapor IO: Matt Trifiro <mtrifiro@vapor.io>
- Packet: Jacob Smith <jacob@packet.net>
- Rapyuta Robotics, Dhananjay Sathe <dhananjay.sathe@rapyuta-robotics.com >
- Microsoft: Nikhil Manchanda <nikman@microsoft.com>
- Microsoft: Andrey Moor <amoor@microsoft.com>
- Microsoft: Cindy Xing <cindy.xing@microsoft.com>
- Nokia: Gergely Csatari <gergely.csatari@nokia.com>
- Cisco: Matt Caulfield <mcaulfie@cisco.com>
- Docker: Justin Cormack <justin.cormack@docker.com>
- Google: Preston Holmes <ptone@google.com>
- Huawei: Wenjing Chu <wenjing.chu@huawei.com>
- Palo Alto Networks: Apoorva Jain <ajain@paloaltonetworks.com>
- gridX: Joel Hermanns <j.hermanns@gridx.ai>
- Chick-fil-A: Brian Chambers <brian.chambers@cfacorp.com>
- Network Architecture 2020: Alex Marcham <alex@networkarchitecture2020.com>
- Polar Squad: Erno Aapa <erno@polarsquad.com>
- China Mobile: Xuan Jia <jiaxuan@chinamobile.com>
- MatMaCorp: William L. Dye <wdye@matmacorp.com>
- Mirantis: Satish Salagame <ssalagame@mirantis.com>
- Mirantis: Marcin Bednarz <mbednarz@mirantis.com >
- IBM: Brent Taylor <brentct@us.ibm.com>; Duy Nguyen <dnguyenv@us.ibm.com>
- Ant Financial: Kailun Qin <kailun.qkl@alibaba-inc.com>
- Platform9: Roopak Parikh <rparikh@platform9.com>
- ARC Advisory Group: Harry Forbes <hforbes@arcweb.com>
- FBK: Daniele Santoro <dsantoro@fbk.eu>
- Atılım University : Ziya Karakaya <ziya.karakaya@atilim.edu.tr
- Red Hat: Hyde Sugiyama <hyde@redhat.com >
- NTT: Hiroki Ito <ito.hiroki@lab.ntt.co.jp>
- Vopak: Mario Pereira <mario.pereira@vopak.com>
- Agriness Edge: Matheus Mota <matheus@agrinessedge.com>
- Transwarp: Hao Xie <hao.xie@transwarp.io>
- Umeå university: Lars Larsson <larsson.work@gmail.com>
- Midokura: Ryu Ishimoto <ryu@midokura.com>

- Midokura: YAMAMOTO Takashi <yamamoto@midokura.com>
  Midokura: Fernando Moreno <fernando@midokura.com>
- Edgeworx: Kilton Hopkins <kilton@edgeworx.io>
- Open Factory: Bharat Khatri <khatri@openfactory.xyz>
- Arundo Analytics: Sindre Gulseth <sindre.gulseth@arundo.com>
- TenxCloud: Lei Wang <wanglei@tenxcloud.com>
- TenxCloud: WeiWei <weiwei@tenxcloud.com>
- Huawei: Helloway He<helloway.wewe@gmail.com>
- Siemens: Sebastian Hänisch <sebastian.haenisch@siemens.com>
- NetApp: Girish Kumar B K <girishkb@netapp.com>
- EdgeSec : Sm@edgesec.io
- CDNetworks : Wayne Rowe : wayne.rowe@cdnetworks.com
- Baidu: Ti Zhou <zhouti@baidu.com>
- Manabu Tsukada <tsukada@hongo.wide.ad.jp>
- EDF: Nick Kampe <nicholas.kampe@edf-re.com>