MAY 2022

Mobile Edge Computing

Realizing the benefits of 5G with MongoDB's Developer Data Platform and Verizon 5G Edge

MongoDB + verizon

Table of Contents

Introduction

As the world continues to grapple with COVID-19 and a rapidly evolving technological landscape, industries are experiencing change like never before. From delays and disruptions to an evolving workforce, enterprises are under tremendous pressure to innovate.

Enterprises are facing many challenges as they evolve their physical infrastructure, digital infrastructure, and workforce at large. As an increasingly large volume of devices begin to become connected, the heterogeneity of hardware types, legacy applications, and proprietary protocols grows with it. As a result, maintenance costs can quickly rise, the need for specialized labor further increases, and the underlying desire for higher reliability and ubiquitous connectivity remains far-fetched.

Beyond the devices themselves, enterprises are grappling with the right choice for connectivity. With the advent of high-speed 5G networks, enterprises are beginning to embrace private mobile networks and advanced networking solutions, such as SD-WAN and other cloud-network networking solutions. However, many lack the expertise to manage these networks, particularly in the face of strong SLOs/SLAs.

Lastly, beyond the digital footprint itself, the labor crunch is further accelerating the need for digital transformation. The precarious state of COVID-19 has left workers calling-in sick or in search of new opportunities altogether.

Across each of these challenges, one thing remains clear. 5G is beginning to unlock new opportunities for bold innovators in one sector after another. By using the power of 5G networks and pairing that with intelligent software, customers are beginning to embrace the next generation of industry.

In this white paper, learn how the speed, throughput, reliability and responsiveness of the Verizon network, paired with the sophistication of next generation MongoDB developer data platform, is poised to transform a variety of industries.

Introducing the Network Edge

As enterprises look to embrace the stringent requirements of modern applications, standalone 5G networks are not enough. While next-generation networks can introduce up to 100x improvements in speed and throughput, the underlying cloud computing environments, without any optimization, remain the primary bottleneck to delivering ultra-low latency applications.

To that end, a new cloud computing paradigm has recently emerged that combines the best experiences of hyperscaler compute and storage with the topological proximity of 5G networks. Mobile edge computing, or MEC, introduces a new mode of cloud deployments whereby enterprises can run applications – through virtual machines, containers, or Kubernetes clusters – within the network itself, across both public and private networks.

In **public MEC deployments**, operators leverage the access network across a variety of geographies to deliver cloud computing services at the edge of commercial networks, across both 4G and 5G. One such example, Verizon 5G Edge with AWS Wavelength, extends the Virtual Private Cloud (VPC) to the network edge to introduce subnets within the 5G network while maintaining a single-pane-of-glass for management and service continuity through best-in-class compute (e.g., Amazon Elastic Compute Cloud) and storage (e.g., Amazon Elastic Block Storage) services respectively.

In **private MEC deployments**, operators leverage the dedicated packet core of the enterprise's private network to co-locate hyperscaler compute and storage to enable a seamless extension of the parent region to the customer premises. In another example, Verizon 5G Edge with AWS Outposts

connects a dedicated, private 5G network with a 42U AWS Outpost that enables more advanced data analytics (e.g., EMR), storage (e.g., EFS), and compute services.

One of the most powerful benefits of Verizon 5G Edge, Verizon's mobile edge compute portfolio, is the ability to create a unified compute mesh across both of these environments – producing a seamless exchange of data and stateful workloads. This simultaneous deployment of both public and private MEC is best characterized as a **hybrid MEC**. Particularly in the case where both the Public MEC and Private MEC deployment are delivered by the same hyperscaler (e.g., AWS Wavelength and AWS Outposts), customers can create a single VPC to encapsulate their public MEC zones (e.g., Wavelength Zones) alongside their dedicated private MEC premises (e.g., Outposts). Moreover, in cases where customers are looking to connect a nationwide (or even global) network of branches, VPC peering and Transit Gateways can be used to simplify the management overhead of maintaining complex cloud networking environments.

Managing Multiple Edges

Beyond the underlying **hybrid MEC infrastructure**, edge computing introduces operational challenges that, at face value, appear insurmountable. Consider the following engineering questions:

- **Deployment:** Given a dynamic fleet of devices, in an environment with 20+ edge zones across both public and private MEC, to which edge(s) should the application be deployed?
- **Orchestration:** For Day 2 operations and beyond, what set of environmental changes, albeit on the cloud, network, or on the device(s), should trigger a change to my edge environment?
- **Edge Discovery:** Throughout the application lifecycle, for a given connected device, which edge(s) is the optimal endpoint for connection?

Verizon has developed a suite of network APIs tailored to answer these questions. From edge discovery and network performance to workload orchestration and network management, Verizon has drastically simplified the level of effort required to build resilient, highly available applications at the network edge without undifferentiated heavy lifting previously required to do so.

Prior to edge network APIs, imagine the brute-force approach to discovering the closest edge endpoint across a deployment of 20+ edge zones. To start, one could leverage a service registry, such as HashiCorp Consul or AWS CloudMap, to populate each of the carrier-facing endpoints. From there, a client wishing to connect to the optimal edge could authenticate to the service registry, and retrieve all of the edge endpoints: perhaps fully-qualified domain names (e.g., edgeapplication.mongodb.com) or Carrier IP addresses (e.g., 145.156.11.28).

Next, the client would have two options to select an endpoint. One naïve approach could be to arbitrarily select an edge endpoint; this would result in the optimal edge being selected roughly ~5% of the time. Another approach would be to conduct an ICMP echo request – colloquially referred to as a "ping" – to each of the endpoints to identify the lowest-latency endpoint. While this approach may yield the optimal result, the latency penalty would be far more significant than simply connecting to the next-best alternative in the cloud (i.e., parent region). As an illustrative example, if the average response time was 50ms, this second approach would incur a latency penalty of 1,000ms – a whole second of latency without any exchange of meaningful application traffic.

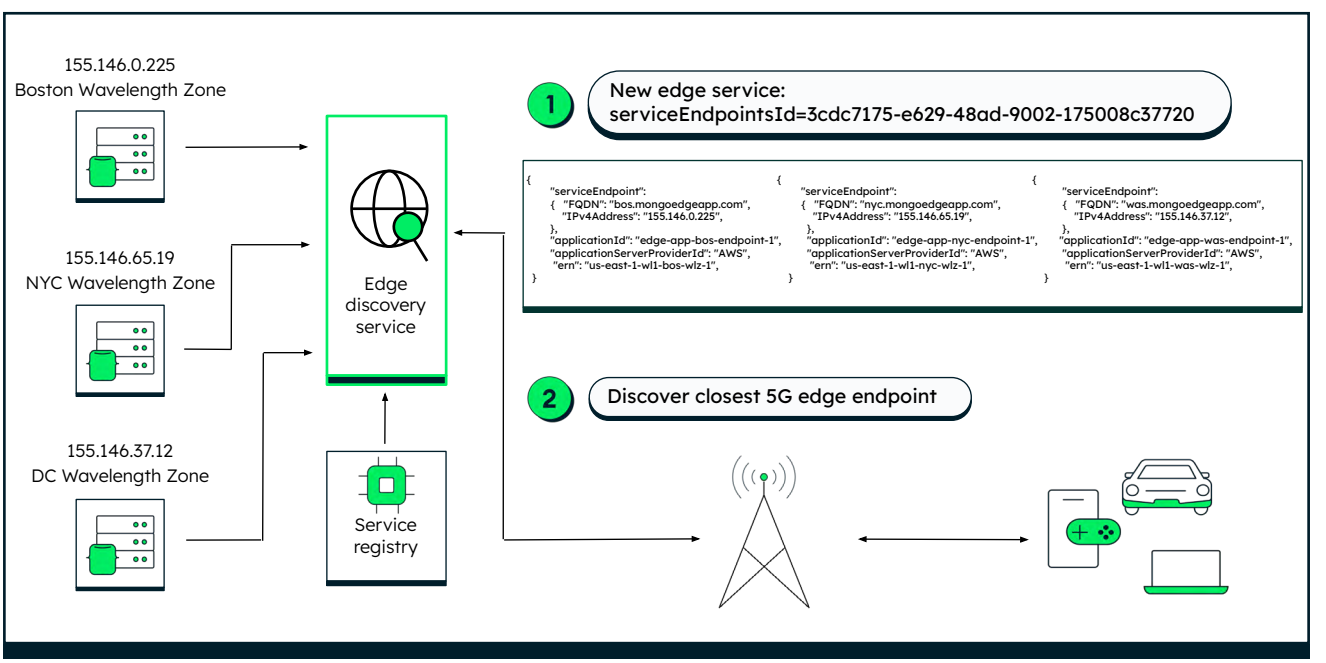
Verizon Edge Discovery Service

Using the Verizon Edge Discovery API, customers can let Verizon manage the complexity of maintaining the service registry as well as identifying the optimal endpoint for a given mobile device. Said differently, using the edge discovery API workflow in place of the self-implemented latency tests, a single request-response would be needed to identify the optimal endpoint.

More broadly, why might you need the operator to create this API?

Today, determining the optimal server across a cloud deployment typically occurs by leveraging well-maintained databases of <IP address, geolocation> to identify the coarse location of the user device, and then calculating the closest server from the user. While GeoIP lookups perform well for traditional CDNs and cloud datacenters, this method falls wildly short in a world of hyper-localized edge deployments.

To realize the true potential of edge computing, the Verizon Edge Discovery Service selects the optimal endpoint is from live network intelligence by dynamically evaluating the state of the network and making routing decisions based on latency, connectivity, load and other unique characteristics of the application profile (Source: "Introducing the Edge Discovery Service" by Robert Belson, Verizon 5G Edge Blog).



While this API addresses the challenge of service discovery, routing, and in some cases advanced deployment scenarios, there are additional challenges outside of the scope of the underlying network APIs. In the case of stateful workloads, how might you manage the underlying data generated from your device fleet? Should all of the data live at the edge, or should it be replicated to the cloud? What about replication to the other edge endpoints?

Using the suite of MongoDB services coupled with Verizon 5G Edge and its network APIs, we will demonstrate popular reference architectures for data across the hybrid edge.

MongoDB for the Edge and Cloud

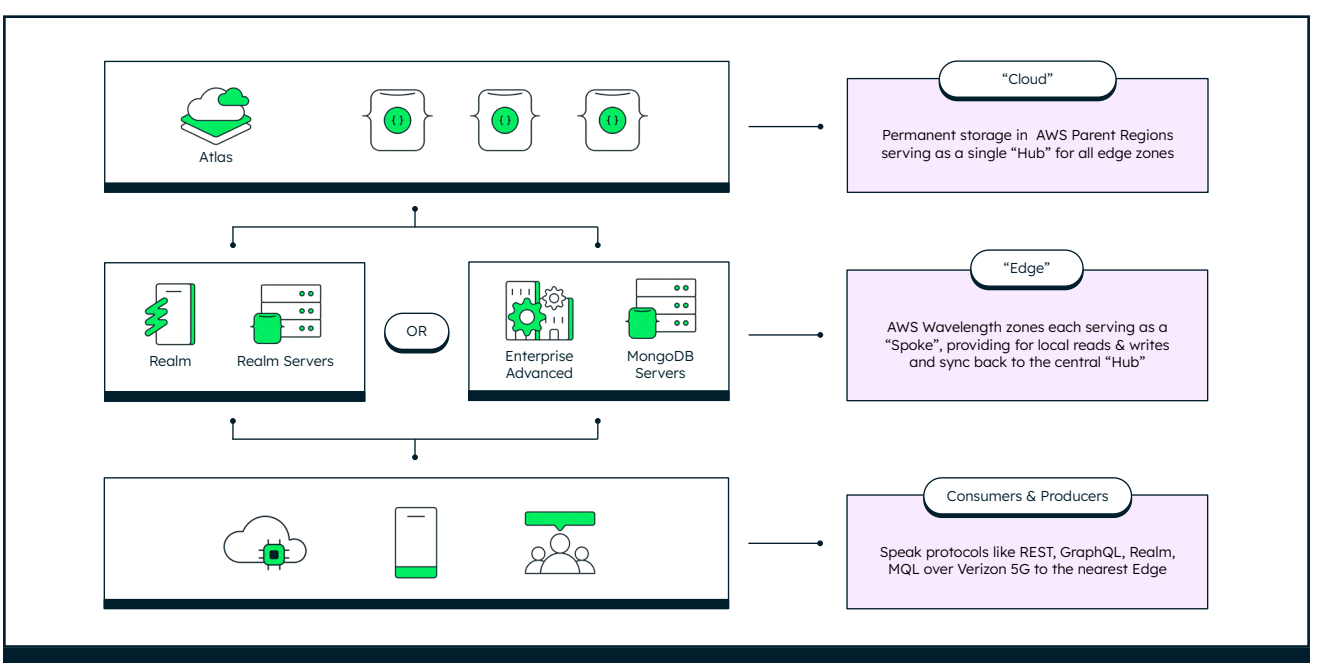
Through Verizon 5G Edge, developers can now deploy parts of their application that require low latency at the edge of 4G and 5G networks using the same APIs, tools, and functionality they use today, while seamlessly connecting back to the rest of their application and the full range of cloud services running in a cloud region.

However, for many of these use cases, a persistent storage layer is required that extends beyond the native storage and database capabilities of the hyperscalers at the edge.

Given the number of different edge locations into which an application can potentially be deployed and to which consumers are able to connect, it is critical to ensure appropriate data is available at the edge. It is also important to note that where consumers are mobile (e.g. vehicles), the optimal edge location can vary. At the same time, having a complete copy of the entire dataset at every edge location in order to cater for this is neither desirable nor practical due to the potentially large volumes of data being managed and the associated multi-edge data synchronization challenges that would be introduced.

The solution revolves around having a complete master view of the dataset stored in the cloud and synchronizing only required data to dedicated edge data stores on-demand. For many cases such as digital twin, this synchronization needs to be bi-directional and potentially include conflict resolution logic.

MongoDB recommends the use of MongoDB Atlas for the central, cloud-based datastore. (Note that while it is also technically feasible to run MongoDB Enterprise for the central datastore, this deployment model is not discussed in detail in this paper). For the edge datastores, there are two options which are outlined below: MongoDB Enterprise, and MongoDB Realm.



It is important to note that the likely data processing requirements at the edge and the cloud often differ with edge-based workload being primarily operational and operating on a subset of the data, while central, cloud-based workloads can be more diverse requiring a broader array of data processing capabilities. These required capabilities will vary based on use case but can often include support for analytical workloads, search, and the efficient processing of much larger data volumes.

MongoDB Atlas in the cloud

MongoDB Atlas provides organizations with a fully managed, elastically scalable developer data platform upon which to build modern applications. MongoDB Atlas can be simultaneously deployed across any of the three major cloud providers (Amazon Web Service, Microsoft Azure, and Google Cloud Platform) and is a natural choice to act as the central data hub in an edge or multi-edge based architecture, as it enables diverse data to be ingested, persisted, and served in ways that support a growing variety of use cases.

Central to Atlas is the MongoDB database which combines a flexible document-based model with advanced querying and indexing capabilities. Atlas is however more than just the MongoDB database and includes many other components to power advanced applications with diverse data requirements.

Native search capabilities

One of the primary ways users engage with applications is through search, and they expect those search capabilities to be instant, intuitive, and comprehensive while behaving in a way that is analogous to Google. This functionality is not well suited to databases and as a result, organizations often resort to integrating specialized search technologies. Atlas Search eliminates the need for a separate search architecture by directly offering developers built-in, full text search capabilities, requiring no additional infrastructure management responsibilities.

Atlas's native search eliminates the challenges of setting up, maintaining, or scaling a separate search platform, meaning developers can develop search based use cases directly against Atlas, removing the need to replicate data elsewhere. Search capabilities, like visual editor, autocomplete, fast faceting, and highlighting facilitate rapid, reliable search.

Real-time analytics

Modern end user applications are often required to generate real-time insights based on data as it becomes available. In parallel, there is typically a need to execute more traditional analytical queries in order to support data science or to train/back-test AI models. MongoDB Atlas can support both of these different categories of use case simultaneously without needing to integrate additional technologies which introduce the need for ETL to move data between systems.

MongoDB's aggregation framework allows for arbitrarily complex pipeline based queries to be executed. Where these queries are long-running analytical queries, they can be directed to a dedicated node containing live data such that there is no contention between it and an operational workload. On the operational front, MongoDB supports change streams which allow for event based systems to be built which automatically react to changes in the underlying data.

Atlas Data Lake operates as a serverless, scalable query engine, delivering more valuable data insights, and a simpler, faster experience for developers. Data stored in AWS S3 in multiple formats – JSON, BSON, CSV, TSV, Avro, ORC, and Parquet – can be analyzed in place, making it possible to glean deeper value from data, while avoiding the constraints of siloed data.

Preserving the rich structure of data generated by modern connected devices is critical. Using Data Lake's federated query capabilities, developers can run a single query to analyze data across multiple MongoDB databases and AWS S3 together and in-place, resulting in the generation of real-time insights while at the same time controlling cost.

BI integration

MongoDB Atlas can be connected to industry-standard BI tooling via the provision of an SQL-based interface. This allows developers to use existing BI tooling to run analytical queries on live data without employing complex ETL processes to generate BI reports and dashboards. This ensures that Atlas integrates seamlessly with tools that teams already rely on, minimizing the disruption to current workloads.

Realm integration

Realm was originally developed as an open source, embedded, lightweight persistence layer for use on resource constrained devices such as mobile phones. Realm was acquired by MongoDB and is now fully integrated into the Atlas data platform. What was once primarily a mobile offering has been enhanced to also support deployment at the edge, which is discussed in greater detail in the following sections.

Core to the integration of Realm into Atlas is the ability to automate the synchronization of data between the Realm database(s) and a central Atlas database via Atlas Data Sync. This addresses challenges otherwise left to application developers such as variable network latency, unreliable connectivity, unpredictable application shut down, impacts on battery life, etc.

It is possible to define server-side functions which allow client applications to interact directly with the central data platform. These functions on Atlas can be called directly from the Realm SDK on the client application, allowing edge-based devices to interact with the cloud-based data-store where needed.

MongoDB Enterprise at the edge

Using MongoDB Enterprise at the edge, developers can leverage the underlying compute and storage available at the edge to deploy MongoDB clusters either a) as standalone clusters or b) highly available replica sets that can synchronize data seamlessly between members.

MongoDB Enterprise offers the same powerful core database used in MongoDB Atlas offering advanced data management capabilities as required by data-intensive edge-based applications.

Database management and orchestration

MongoDB Cloud Manager can automate, monitor, and back up MongoDB infrastructure, including infrastructure deployed at a network edge.

Cloud Manager Automation enables the configuration and maintenance of MongoDB nodes and clusters. This is achieved by the installation of MongoDB Agents running on each MongoDB host which communicates with Cloud Manager to maintain MongoDB deployments.

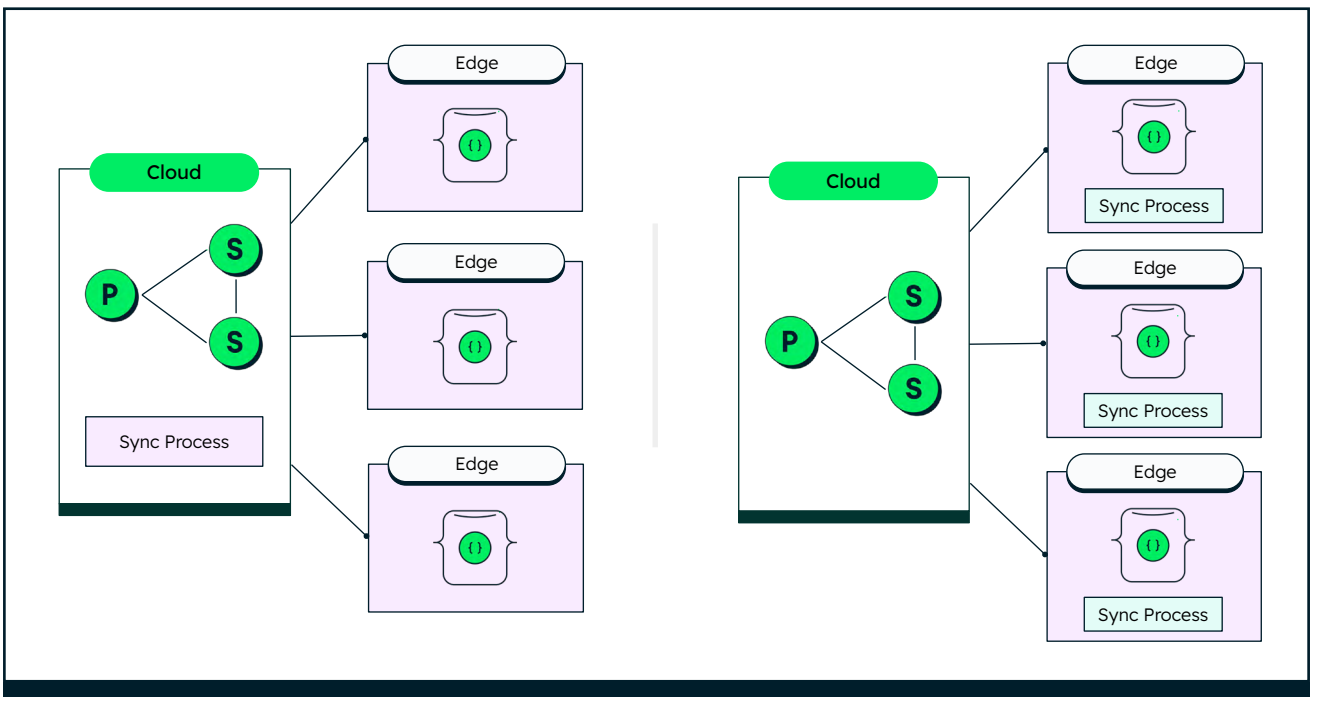
With Cloud Manager, it is possible to deploy and manage both standalone instances, as well as highly available MongoDB replica sets onto edge-based compute and storage resources.

Data synchronization

As previously mentioned, for most use cases there will be a requirement to synchronize data from the edge to the cloud. While it is technically possible to utilize MongoDB's native replication capabilities (if using MongoDB Enterprise as the central datastore), the single master nature of a MongoDB replica set will mean that local writes could only happen either at the edge or in the cloud.

Similarly, in multi-edge deployments, if MongoDB native replication is used for synchronization between edge locations, similar challenges would exist with local writes only being available in a single location. Extending this pattern to a hybrid-edge deployment is however discouraged and should be considered as an anti-pattern due to the latency that could be introduced.

A more flexible approach will be to implement a custom solution to facilitate the synchronization of data subsets between the edge and the cloud and vice versa based on the needs of the individual application.



Such a solution can be built using MongoDB's native change stream feature which allows a consumer process to register for real-time notifications of events occurring in a cluster (e.g. data conforming to a query is inserted, updated, etc). The consumer assumes responsibility for processing these events, potentially including conflict resolution and persisting to a target cluster. As indicated above, the consumer in question can be deployed either centrally or on edge nodes, supporting either push or pull models of data distribution as needed.

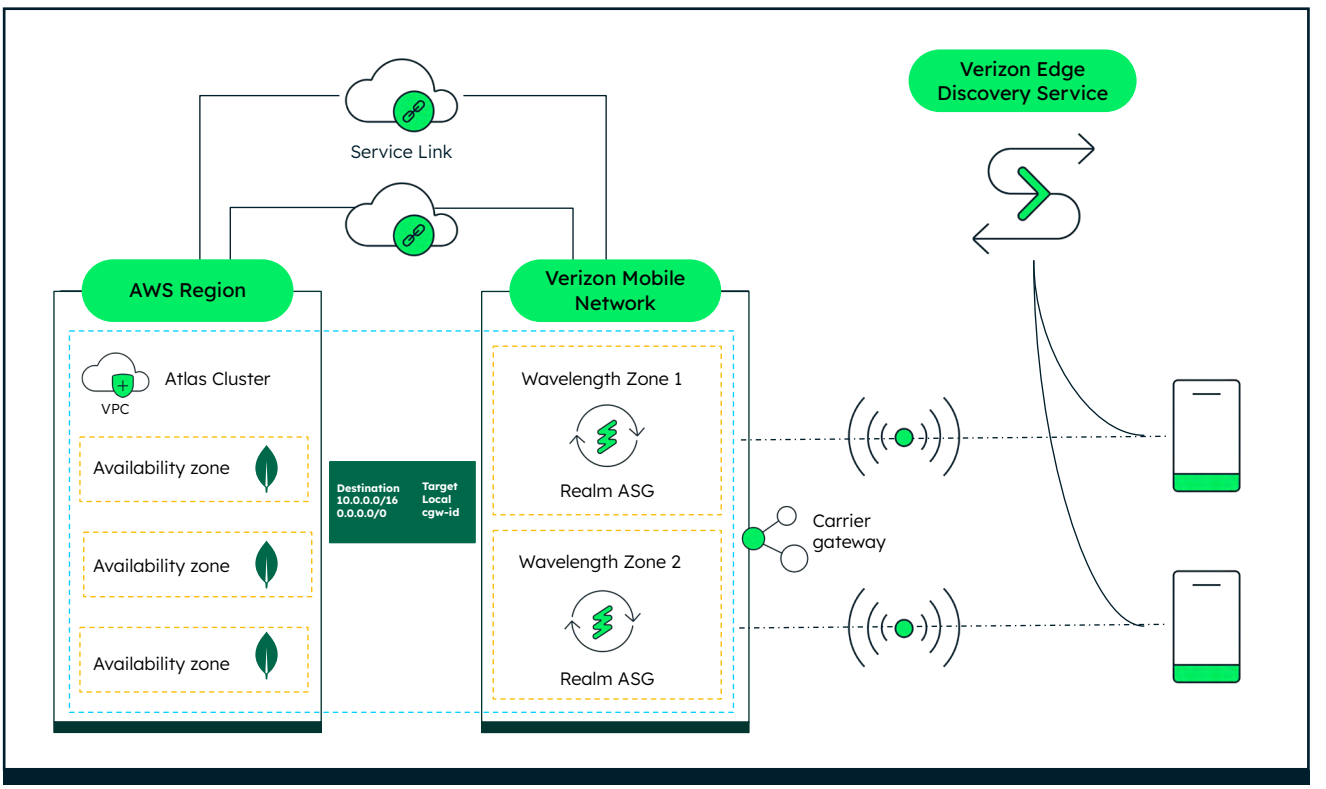
Realm at the edge

While MongoDB Enterprise and Cloud Manager are supported at the network edge, there may be drawbacks of deploying this pattern. In cases of existing MongoDB customers, the management of additional clusters and inter-cluster replication may not be desired. Moreover, as a cluster typically requires three or more nodes, the compute and storage resources required may be "overkill" for the use case in question.

As a result, for environments utilizing MongoDB Atlas in the Cloud, Realm at the edge is a powerful alternative. Since its initial design as a mobile-only database, Realm has evolved to support deployment as an edge-based data store. Realm persists data on-disk and uses an object-oriented data model that saves developers from writing thousands of lines of non-value-add code.

Given that Realm is an embedded database, there is no associated management overhead or dedicated resource requirements. It is deployed along with the edge-based application or alternatively as a dedicated, independently scalable, edge-based persistence layer.

Additionally, Atlas Device Sync supports seamless connections back to MongoDB Atlas (see below), alongside fully integrated application development services like functions, triggers, and authentication. Moreover, the underlying resources supporting the Device Sync-based applications can be fully elastic. Any number of instances can be spun up within seconds – via auto scaling – within a given edge zone or across multiple edge zones, as seen in the diagram below.



Data synchronization

One of the most powerful capabilities of Realm is the built-in synchronization capabilities. Atlas Device Sync allows data to be automatically bi-directionally synchronized between a Realm instance and MongoDB Atlas. This sync includes full, deterministic conflict resolution which is critical, especially for digital twin use cases.

In complex, multi-edge deployments, the Verizon Edge Discovery Service can be used to identify the closest Realm endpoint and to retrieve data from MongoDB Atlas at the start of a mobile session, allowing the architecture to unlock a multi-tiered design between the region, mobile edge, and device itself.

Key to determining which data should be synchronized to which edge location is the definition of the Realm object schema. An object schema defines the properties and relationships of a Realm object using JSON schema notation. Using this object schema, an application is able to use familiar MongoDB query syntax to define which data should be synchronized. This vastly simplifies development of applications requiring multi-edge architectures.

In the example below which uses the Flexible Sync version of Atlas Device Sync, a subscription is initiated for device data in which a field 'ern' (Edge Resource Name) is equal to the value 'us-east-1-wl1-bos-wlz-1'. This will result in all documents in MongoDB Atlas matching the specified criteria being automatically synchronized to the edge location on which the code is executing.

```
SyncConfiguration config = new SyncConfiguration.Builder(app.currentUser())
  .initialSubscriptions(new SyncConfiguration.InitialFlexibleSyncSubscriptions() {
   @Override
   public void configure(Realm realm, MutableSubscriptionSet subscriptions) {
    // add a subscription with a name
    subscriptions.add(Subscription.create("deviceSubscription",
      realm.where(device.class)
       .equalTo("ern", "us-east-1-wl1-bos-wlz-1")));
   }
  })
  .build();

Realm.getInstanceAsync(config, new Realm.Callback() {
 @Override
 public void onSuccess(Realm realm) {
  Log.v("EXAMPLE", "Successfully opened a realm.");
  // later, you can look up this subscription by name
  Subscription subscription = realm.getSubscriptions().find("deviceSubscription");
 }
});
```

Through this Verizon and MongoDB-native architecture, the overarching complexity of edge data management is dramatically simplified both from an application development perspective and from a network operations perspective. From the complexity of the network (e.g., carrier-grade network address translation, edge discovery) to the complexity of the data (e.g., edge-to-cloud replication, conflict resolution), the development time to create stateful applications is drastically reduced.

AWS Wavelength Integration

As one such example of mobile edge computing on Verizon 5G Edge, AWS Wavelength brings the best of AWS compute and storage to the edge of the Verizon mobile network. In fact, the vast majority of AWS Wavelength's infrastructure concepts are the exact same as what would be expected in the parent region today.

In the parent region, a Virtual Private Cloud (VPC) consists of a series of subnets corresponding to an availability zone (e.g., us-east-1a). With Wavelength Zones, subnet creation is exactly the same and Verizon Wavelength Zones are denoted with the following nomenclature:

[Region] [Carrier][City Code][Edge Logical ID]	us-east-1-wl1-nyc-wlz-1

Today, Verizon 5G Edge with AWS Wavelength is available in 17 cities across both us-east-1 and us-west-2 regions. The complete list of cities can be found below:

Boston: us-east-1-wl1-bos-wlz-1

Atlanta: us-east-1-wl1-atl-wlz-1

Washington DC: us-east-1-wl1-was-wlz-1

New York City: us-east-1-wl1-nyc-wlz-1

Miami: us-east-1-wl1-mia-wlz-1

Dallas: us-east-1-wl1-dfw-wlz-1

Houston: us-east-1-wl1-iah-wlz-1

Chicago: us-east-1-wl1-chi-wlz-1

Charlotte: us-east-1-wl1-clt-wlz-1

Detroit: us-east-1-wl1-dtw-wlz-1

Minneapolis: us-east-1-wl1-msp-wlz-1

San Francisco Bay Area: us-west-2-wl1-sfo-wlz-1

Las Vegas: us-west-2-wl1-las-wlz-1

Denver: us-west-2-wl1-den-wlz-1

Seattle: us-west-2-wl1-sea-wlz-1

Phoenix: us-west-2-wl1-phx-wlz-1

Los Angeles: us-west-2-wl1-lax-wlz-1

After creation of a subnet in a Wavelength Zone, a carrier gateway is attached to the VPC to connect the logically-isolated network to that of the carrier. The carrier gateway specifically takes care of the Network Address Translation (NAT) between private IP addresses within the VPC and the Carrier IP addresses allocated to each Wavelength Zone, otherwise referred to as a *network border group* within AWS parlance.

To ensure that the carrier gateway is participating in the routing selection for Wavelength Zone subnets, a carrier route table must be created. In this route table, the default route will point directly to the Amazon Resource Name (ARN) of the carrier gateway. As a result, any resources with traffic directed to the internet or a Verizon-connected mobile device will be routed through the carrier gateway to the Verizon carrier network.

While internet-bound traffic initiated from the Wavelength Zone is supported, the reverse is not supported. This is due to the overarching security posture of the mobile edge computing design.

Across all traffic flows, 4G and 5G-connected devices are supported, irrespective of the underlying spectral resources consumed. (To learn more about Verizon 5G Nationwide or Verizon Ultra Wideband, visit the Verizon FAQ page.)

Data Transfer

Device to edge

Transferring data from device to edge supports a number of protocols. As one example, using a MongoDB Realm based architecture at the edge it is possible to support both MQTT and RESTful interfaces. However, the choice of when to use MQTT over REST depends on the use case. REST was designed as a request-response model over HTTP and supports a small set of predefined operations (e.g., GET, POST, PUT). MQTT, on the other hand, is a publish-subscribe model over TCP/IP sockets that is faster and more flexible.

It should be noted that in a Realm architecture, an *MQTT-to-Realm* proxy service is needed on each edge – particularly in the case of IoT applications – to connect the underlying MQTT message from the IoT device to the Realm database. More information on the MQTT implementation can be found in this MongoDB Blog: Take Advantage of Low-Latency Innovation with MongoDB Atlas, Realm, and AWS Wavelength.

Edge to cloud

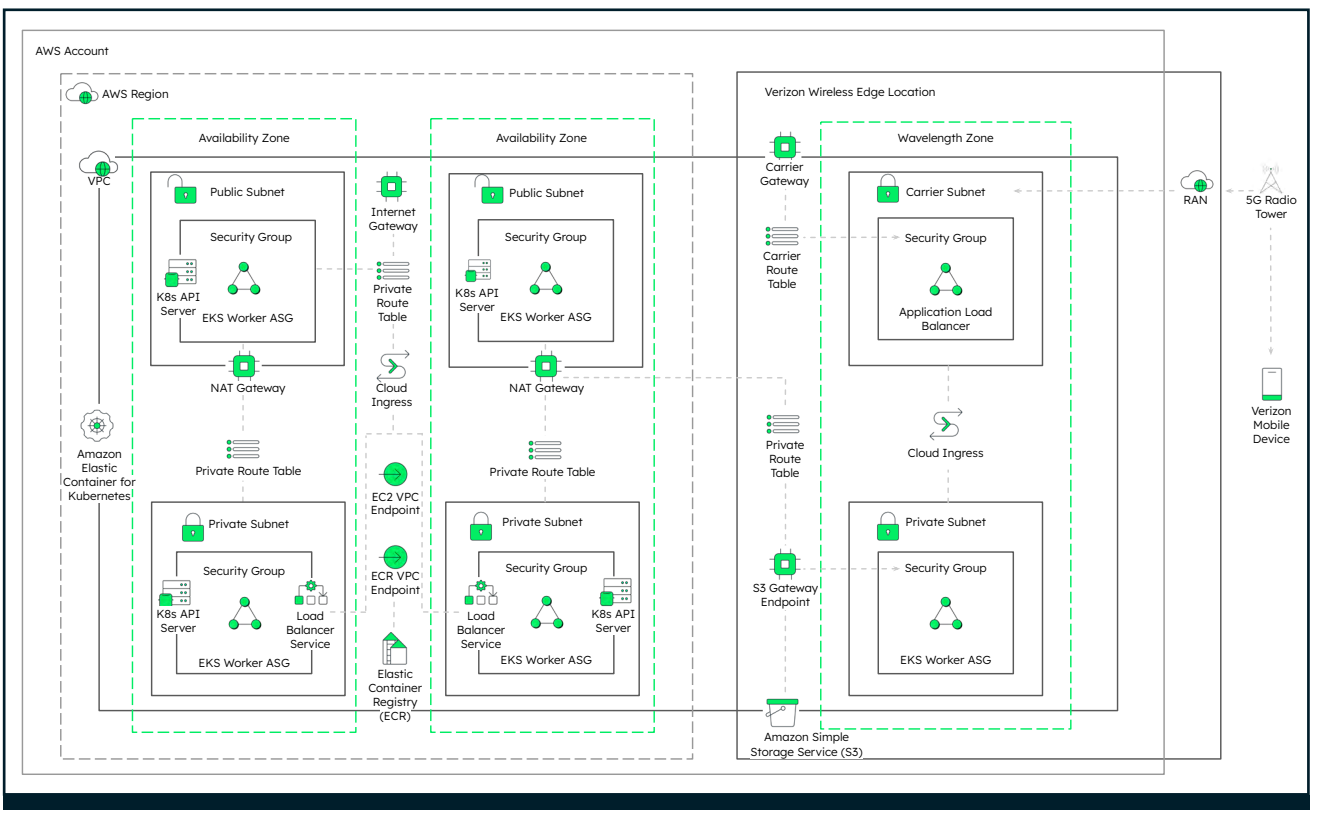
As discussed at length above, seamless data transfer between edge(s) and the cloud is a critical component of an edge-based architecture using Verizon 5G Edge and MongoDB. Both of the outlined data persistence options for edge deployment (Realm + Atlas Device Sync and Enterprise) are fully supported both for public and private MEC.

Edge to edge

Verizon 5G Edge with AWS Wavelength was designed with a hub-and-spoke architectural model. For each Wavelength Zone deployed, a service link that connects the edge back to the parent region was constructed for ultra-high throughput and reliability. As a result, direct edge-to-edge connectivity within a VPC is not supported (in the absence of a region-based proxy), while edge-to-edge connectivity using Carrier IP addresses over the Verizon backbone is supported. However, direct edge-to-edge data replication is currently considered an anti-pattern.

Containers at the Edge

While the entirety of the reference data architectures above have been focused on virtual machines, containers are deeply supported at the edge. In fact, Verizon 5G Edge with AWS Wavelength supports both Amazon Elastic Container Service (ECS) and Amazon Elastic Kubernetes Service (EKS). Consider the following reference architecture for Amazon EKS at the edge:



When deploying EKS, the control plane is instantiated in subnets in the parent region and control plane in the Wavelength Zones is not supported. After deploying the control plane, self-managed node groups can be launched in one or many Wavelength Zones. However, AWS Fargate and managed node groups are not supported in AWS Wavelength.

For your node groups to communicate with the control plane, there are two primary modes of EKS cluster endpoint access.

- **Public access:** In this access mode, a route over the public internet must exist from your carrier node groups to the EKS control plane. To do this, a Carrier IP must be assigned to each node and the EKS control plane security group should permit inbound traffic from the node.
- **Private access:** In this access mode, the connection from your node to the control plane is forged over the AWS backbone. To do this, you must create three VPC Interface endpoints: an EC2, ECR (DKR), and ECR (API) endpoint. This, in turn, allows the traffic from your Wavelength node to be routed through the interface endpoints to the cluster endpoint.

If MongoDB Enterprise is used at the edge, after configuring the cluster, MongoDB can be deployed as a StatefulSet within the Kubernetes cluster.

To schedule a workload to a particular Wavelength Zone, node selectors can be used to match the workload to a specific Availability Zone ID for a given set of nodes. Through the native StatefulSets object in Kubernetes, a MongoDB ReplicaSet can be deployed. Using a StatefulSet, each MongoDB replica is given an ordinal identifier, as seen here:

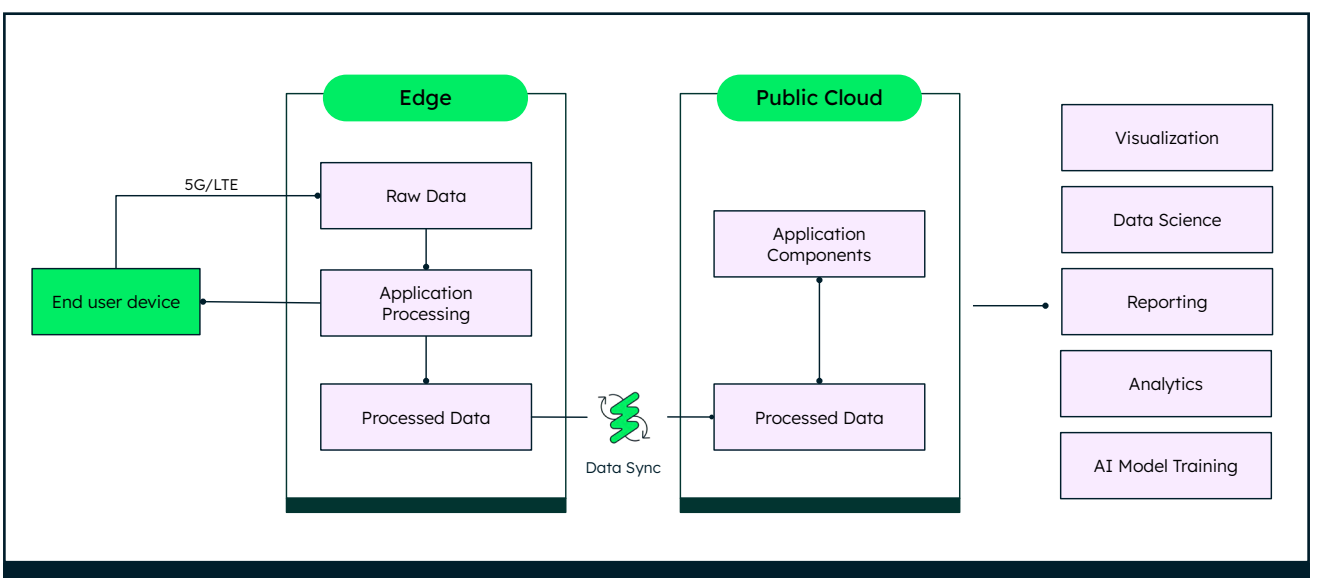
```
kubectl get pods

NAME READY STATUS RESTARTS AGE
mongo-0 2/2 Running 0  3m
mongo-1 2/2 Running 0  3m
mongo-2 2/2 Running 0  3m
```

To learn more about MongoDB and StatefulSets, visit the Kubernetes documentation.

Data Processing

In an edge or multi-edge based application, there are multiple layers in which data can be processed and the objective of the data processing at each layer can differ based on the use case in question.



Processing on the device

End devices will typically be responsible for the accumulation of large amounts of locally generated data. In the example of a manufacturing IOT platform or an automotive use case, this might be in the form of sensor readings. While in some circumstances, local processing may be required, in many cases, the devices will be mainly responsible for the aggregation of the numerous local data sources and the transmission of high-fidelity, high-frequency data to the edge for processing.

While there can be many formats for data to be handled within a device, the transmission of the data to the edge will typically use industry standard protocols such as MQTT, OPC, or AMQP.

Processing at the edge

By executing data processing at the edge, the time taken to react to any given situation is minimized by enabling application logic to be executed with minimal latency. Additionally by processing and aggregating data at the edge, it is possible to control the data that is sent upstream to the cloud both in terms of content and volume.

Data arriving at an edge can first be pre-processed to verify the fidelity and completeness of the data before it is passed on for application-specific processing. Application-based processing at the edge is able to utilize not only the incoming data but also locally stored data via MongoDB or Realm. This data can for example include either corresponding data from other devices attached to the same edge and/or short-term history which allows correlation and regression-based algorithms to be utilized.

The output from the data processing could be that no action is needed and the data can simply be persisted for later use. It could alternatively result in the generation of an error/warning which should be sent to the operators. In other circumstances however, it could result in an automated decision to take action. Where action is required, command messages can be automatically sent to the end device.

Another responsibility of edge-based processing is data aggregation. Data received and processed from devices will often be high frequency and represent a raw data stream. While this raw data stream may be required for edge-based processing, the same granularity of data is often not required in central, cloud-based storage. Aggregating data at the edge allows for available bandwidth to be optimized by only sending the data upstream that is required for the supported use cases. For example, if the cloud-based data is used for reporting and visualization, instead of sending every reading to the cloud, it might be possible to periodically calculate an average reading.

Processing in the cloud

Cloud-based processing typically supports use cases such as analytics, visualization (digital twin), and data science as well as general application services. In order to do this, having a complete view of the data accumulated from all edges is critical. As mentioned above however, the granularity of data required to support these use cases is often lower than that required for edge-based automated decisioning.

In the example of a digital twin use case, human operators will often interact with a device based on the state presented in the cloud view with any actuation commands (e.g. turn on/off) being translated and sent to the device via the edge where high-level user-based comments are translated into messages sent to underlying devices.

Real-World Use Case Applications

The techniques and technologies described in this paper have far reaching implications for real-world use cases across many different industries. Any organization that has a requirement for low-latency data processing can benefit from edge based architectures. Some examples are outlined below.

Agriculture

One of the most powerful uses of data analytics at-scale has been in the agriculture sector. For decades, researchers have grappled with challenges such as optimal plant breeding and seed design which, to date, has been a largely manual process.

Through purpose-built drones and ground robotics, new ways to conduct in-field inspection using computer vision have been used to collect height, biomass, early vigor, and anomaly detection. However, these robots are often purpose-built with large data systems on-device, requiring manual labor to upload the data to the cloud for post-processing.

Using the edge, this entire workflow can be completely optimized; starting with the ground robotics fleet, the device can be retrofitted with a 5G modem to disintermediate much of the persistent data collection. Instead, the device can collect data locally, extract relevant metadata, and immediately push data to the edge for real-time analytics and anomaly detection.

In this manner, field operators can collect insights about the entirety of their operations – across a given crop field or nationwide – without waiting patiently until the completion of a given mission.

Automotive

Modern vehicles are far more connected than ever before with almost all models being produced today containing embedded SIM cards enabling even more connected experiences. Additionally, parallel advances are being made to enable roadside infrastructure connectivity. Together these advances will power not just increased data sharing between vehicles but also between vehicles and the surrounding environment (V2V2X).

In the shorter term, edge-based data processing has the potential to yield many benefits both to road users and vehicle manufacturers.

Data quality and bandwidth optimization

Modern vehicles have the ability to transmit large amounts of data both in terms of telemetry relating to the status of the vehicle but also the observed status of the roads. If a vehicle detects that it is in a traffic jam, for example, then it might relay this information so that updates can be made available to other vehicles in the area to alert drivers or replan programmed routes. While a useful feature, there can be many vehicles reporting the same information. By default all of this information will be relayed to the cloud for processing resulting in large amounts of redundant data. Instead, through edge-based processing:

1. Data is shared more quickly between vehicles in a given area using only local resources.
2. Costs relating to cloud-based data transfer are better controlled.
3. Network bandwidth usage is optimized.

While improving control of network usage is clearly beneficial, arguably a more compelling use of edge-based processing in the automotive industry relates to aggregating data received from many vehicles to improve the quality of data sent to the

cloud-based data store. Staying with the example of the traffic jam, all of the vehicles transmitting information about the road conditions will be doing so based on their understanding, gained through GPS as well as internal sensors. Some vehicles will be sending more complete or accurate data than others, and therefore by aggregating the many different data feeds at the edge, resulting in a more accurate, complete representation of the situation.

Smart manufacturing

Modern industrial manufacturing processes are making greater use of connected devices to optimize production while controlling costs. Connected devices exist throughout the process, from sensors on manufacturing equipment to mobile devices used by employees on the factory floor to connected vehicles transporting goods – all generating large amounts of data.

To realize the benefits of this data, it is critical that it be processed and analyzed in real time to enable rapid action. As described earlier in this paper, moving this data from the devices to the cloud for processing introduces unnecessary latency and data transmission that can be avoided by processing at the edge.

Process optimization

Through real-time processing of telemetry data it is possible to make automated, near-instantaneous changes to the configuration of industrial machinery in response to data relayed from a production line. There are numerous potential outcomes of such a process such as improved product quality, increased yield, optimization of raw material use, and the tracking of standard KPIs such as OEE (overall equipment efficiency).

Preventative maintenance

Similar to process optimization, real-time processing of telemetry data can enable the identification of potential impending machinery malfunctions before they occur and result in production downtime. More critically however, should a situation be detected which has the potential to either damage equipment or pose a danger to those working in the immediate vicinity, being able to shut it down automatically as soon as the condition is detected is vital.

Conclusion

The future of intelligent machines and data is all about taking individual components and having them operate together as one. Through data generated by connected devices, we can begin to orchestrate a new set of applications that will drive efficiencies across businesses and improve user experiences. Through 5G networks and edge computing, we can begin to develop end-to-end value chains that perform as a cohesive, self-improving, and self-healing service.

About the Authors

Robert Belson is a Principal Engineer at Verizon and currently leads their Developer Relations & Experience efforts for Verizon's edge computing portfolio. In this capacity, he serves as Editor-in-Chief of the 5G Edge Blog, designs developer tutorials & training modules, and works with large enterprise customers to solve their business challenges using automation, hybrid networking, and the edge cloud.

Steve Dalby is a Director in the MongoDB Industry Solutions team where he focuses on how MongoDB technology can be leveraged to solve challenges faced by organizations working in the telecommunications industry. Prior to this role, Steve held numerous leadership roles with MongoDB's Professional Services team in EMEA.

Resources

For more information, please visit mongodb.com or contact us at sales@mongodb.com.

Case Studies

Presentation

Free Online Training

Webinars and Events

Documentation

MongoDB Atlas database as a service for MongoDB

MongoDB Enterprise Download

MongoDB Realm

Legal notice (MongoDB)

This document includes certain "forward-looking statements" within the meaning of Section 27A of the Securities Act of 1933, as amended, or the Securities Act, and Section 21E of the Securities Exchange Act of 1934, as amended, including statements concerning our financial guidance for the first fiscal quarter and full year fiscal 2021; the anticipated impact of the coronavirus disease (COVID-19) outbreak on our future results of operations, our future growth and the potential of MongoDB Atlas; and our ability to transform the global database industry and to capitalize on our market opportunity. These forward-looking statements include, but are not limited to, plans, objectives, expectations and intentions and other statements contained in this press release that are not historical facts and statements identified by words such as "anticipate," "believe," "continue," "could," "estimate," "expect," "intend," "may," "plan," "project," "will," "would" or the negative or plural of these words or similar expressions or variations. These forward-looking statements reflect our current views about our plans, intentions, expectations, strategies and prospects, which are based on the information currently available to us and on assumptions we have made. Although we believe that our plans, intentions, expectations, strategies and prospects as reflected in or suggested by those forward-looking statements are reasonable, we can give no assurance that the plans, intentions, expectations or strategies will be attained or achieved. Furthermore, actual results may differ materially from those described in the forward-looking statements and are subject to a variety of assumptions, uncertainties, risks and factors that are beyond our control including, without limitation: our limited operating history; our history of losses; failure of our database platform to satisfy customer demands; the effects of increased competition; our investments in new products and our ability to introduce new features, services or enhancements; our ability to effectively expand our sales and marketing organization; our ability to continue to build and maintain credibility with the developer community; our ability to add new customers or increase sales to our existing customers; our ability to maintain, protect, enforce and enhance our intellectual property; the growth and expansion of the market for database products and our ability to penetrate that market; our ability to integrate acquired businesses and technologies successfully or achieve the expected benefits of such acquisitions; our ability to maintain the security of our software and adequately address privacy concerns; our ability to manage our growth effectively and successfully recruit and retain additional highly-qualified personnel; the price volatility of our common stock; the financial impacts of the coronavirus disease (COVID-19) outbreak on our customers, our potential customers, the global financial markets and our business and future results of operations; the impact that the precautions we have taken in our business relative to the coronavirus disease (COVID-19) outbreak may have on our business and those risks detailed from time-to-time under the caption "Risk Factors" and elsewhere in our Securities and Exchange Commission ("SEC") filings and reports, including our Quarterly Report on Form 10-Q filed on December 10, 2019, as well as future filings and reports by us. Except as required by law, we undertake no duty or obligation to update any forward-looking statements contained in this release as a result of new information, future events, changes in expectations or otherwise.

Legal notice (Verizon)

Further, this document includes statements based on Verizon's current assumptions and expectations about its future performance, including statements regarding operational, technological and business strategy, plans, initiatives and objectives. These statements typically include words such as "will," "aim," "anticipate," "believe," "drive," "estimate," "expect," "intend," "may," "plan," "project," "strategy," "target," and "goal" or similar terms. For those statements, we claim the protection of the safe harbor for forward-looking statements contained in the Private Securities Litigation Reform Act of 1995. Our actual future results, including the achievement of our strategic, operational and business plans, initiatives and objectives, could differ materially from our projected results as the result of changes in circumstances, assumptions not being realized, or other risks, uncertainties and factors. For information on certain factors that could cause actual events or results to differ materially from our expectations, please see our filings with the Securities and Exchange Commission, including our most recent annual report on Form 10-K and subsequent reports on Forms 10-Q and 8-K. Investors are cautioned not to place undue reliance on any such forward-looking statements, which speak only as of the date they are made. Verizon undertakes no obligation to update any forward-looking statements, whether as a result of new information, future events or otherwise.