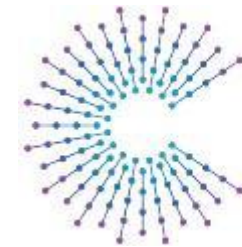


# BMW & MongoDB Architecture and Infrastructure Working Session VSS-based Data Middleware with Tiered Sync (Edge Server)

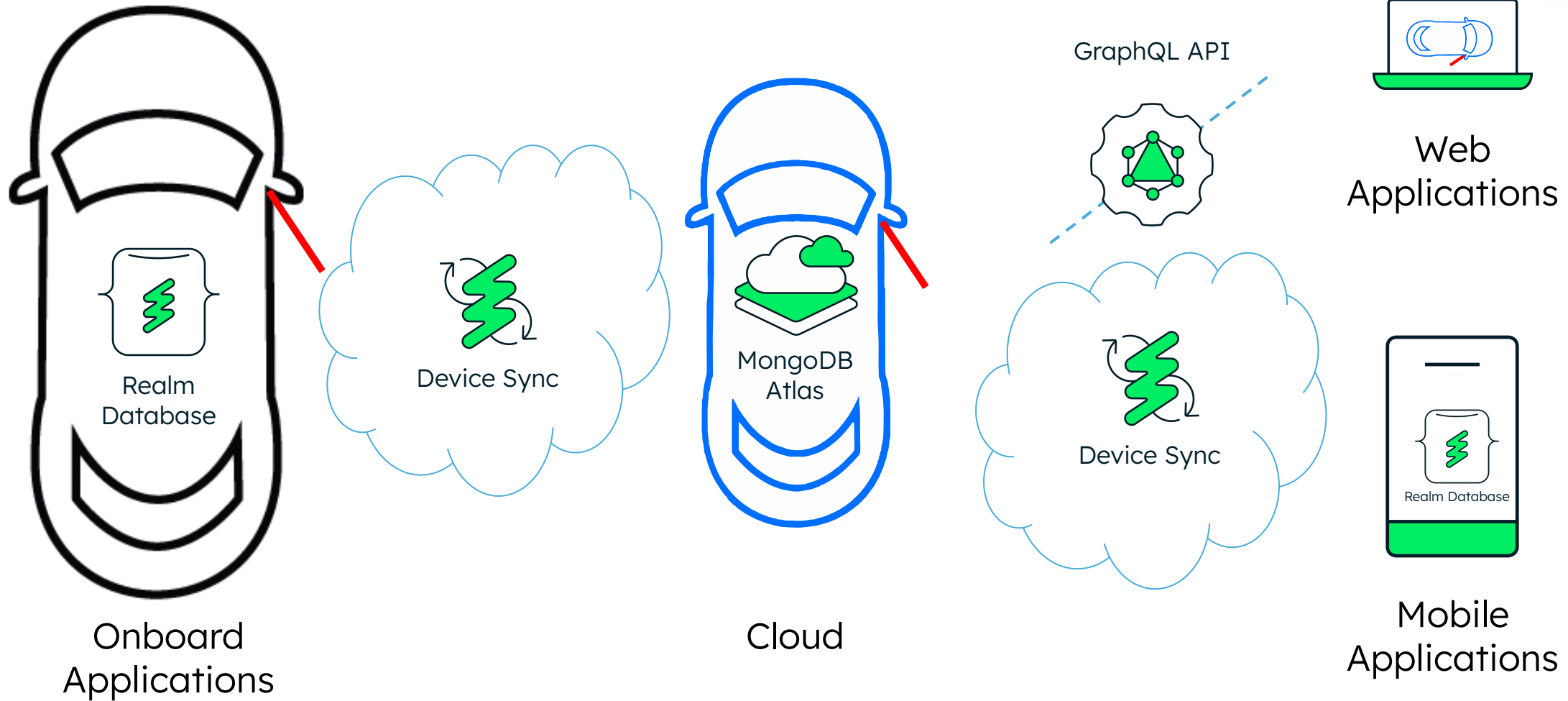
Arnaldo Vera, Industry Solutions MongoDB



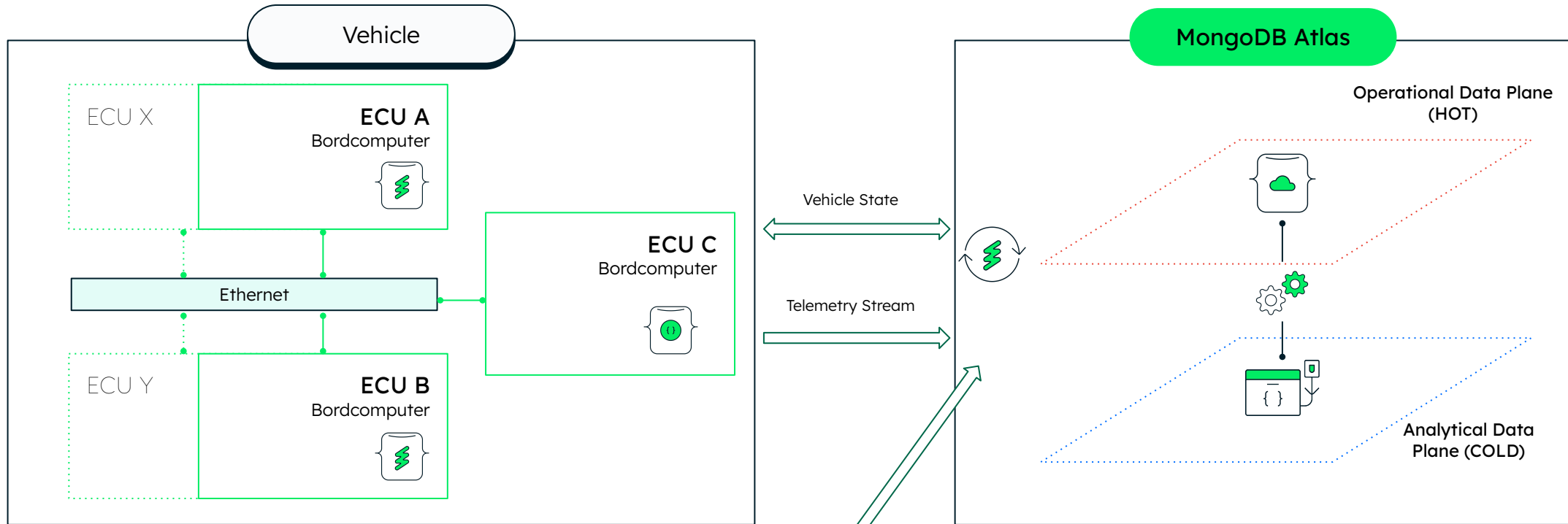
**COVESA**

Accelerating the future of connected vehicles




# The Concept

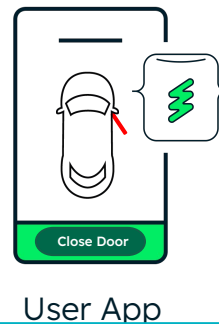


# Connected Car without Edge Server



## Legend

-  **Realm** - Object Oriented lightweight DB.
-  **Device Sync** - Automatic Synchronization engine
-  **MongoDB** - Document Model based DB

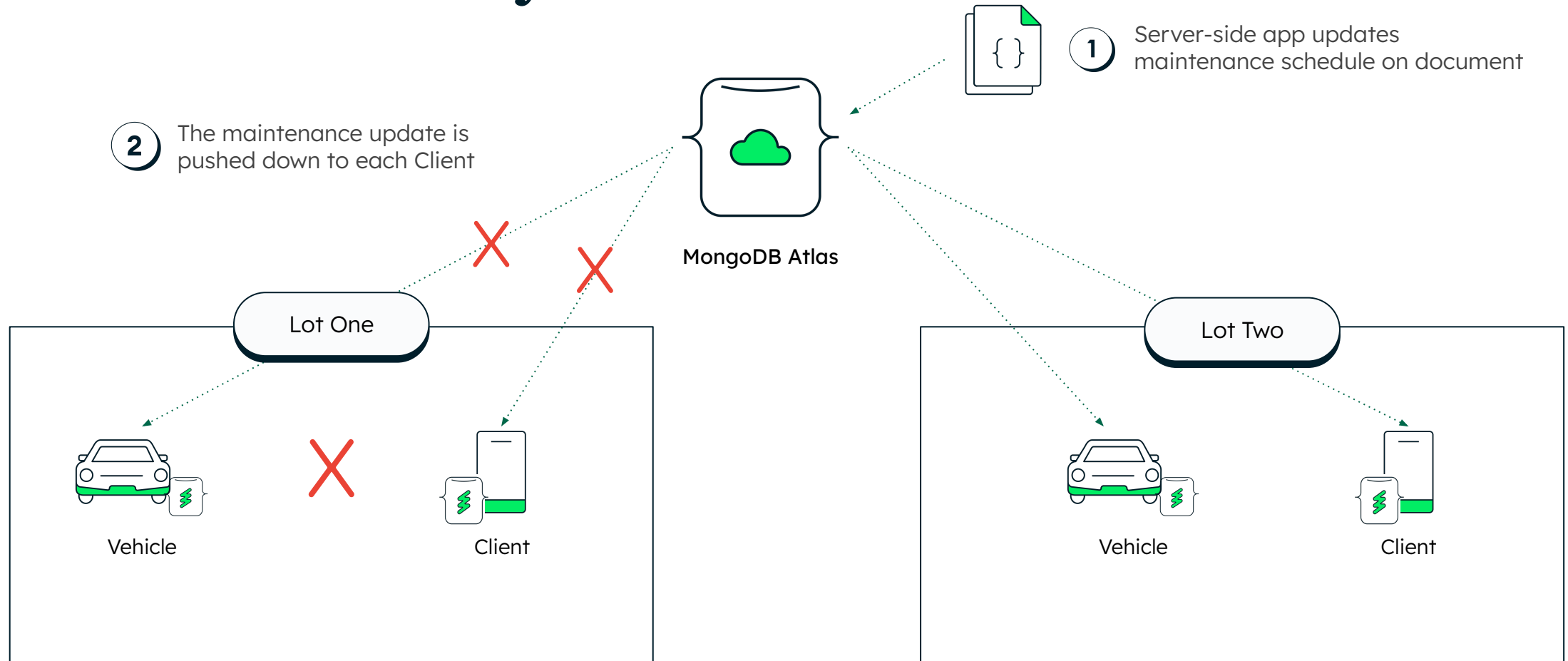




# Atlas Device Sync: Edge Server (Tiered Sync)



# Atlas Device Sync: Standard Architecture





# Edge: Lot Offline

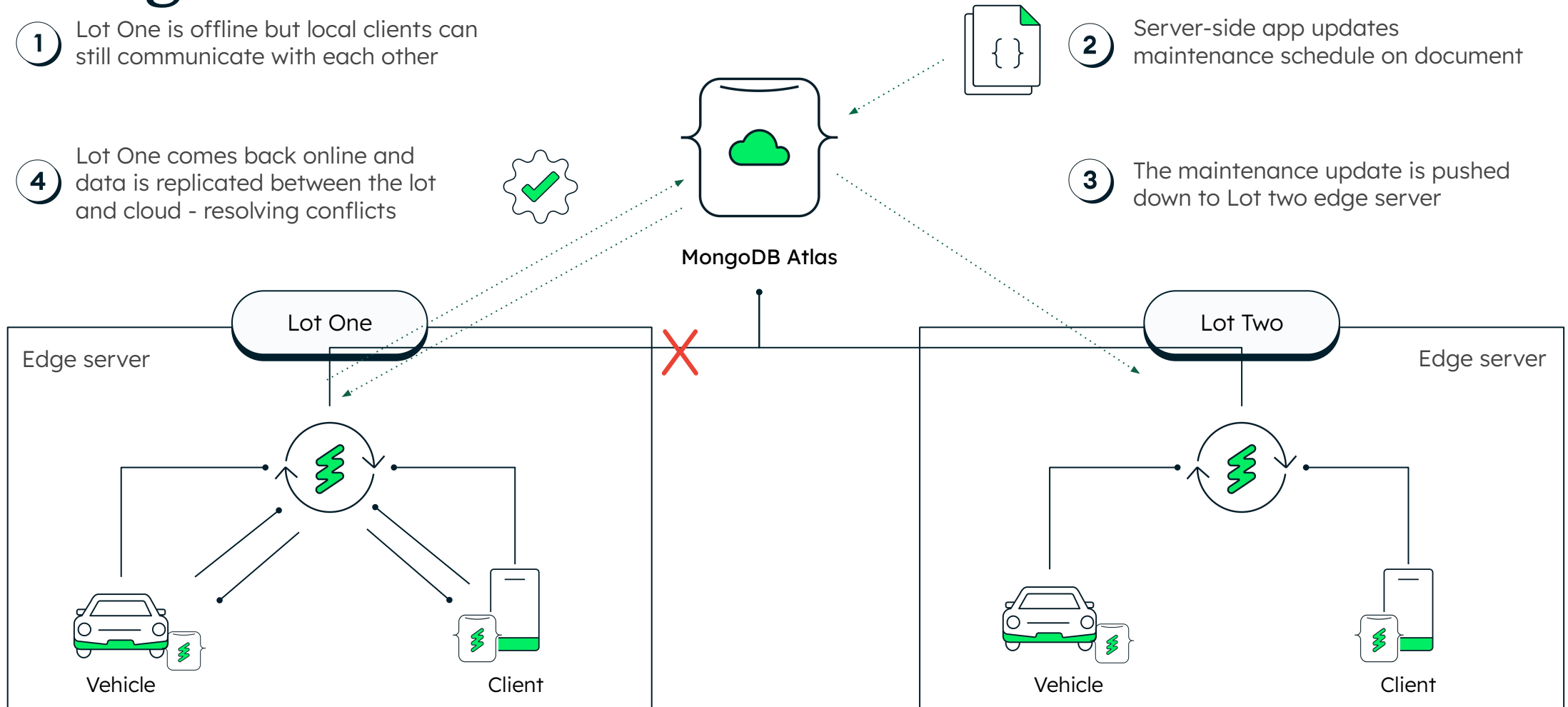
1 Lot One is offline but local clients can still communicate with each other

4 Lot One comes back online and data is replicated between the lot and cloud - resolving conflicts

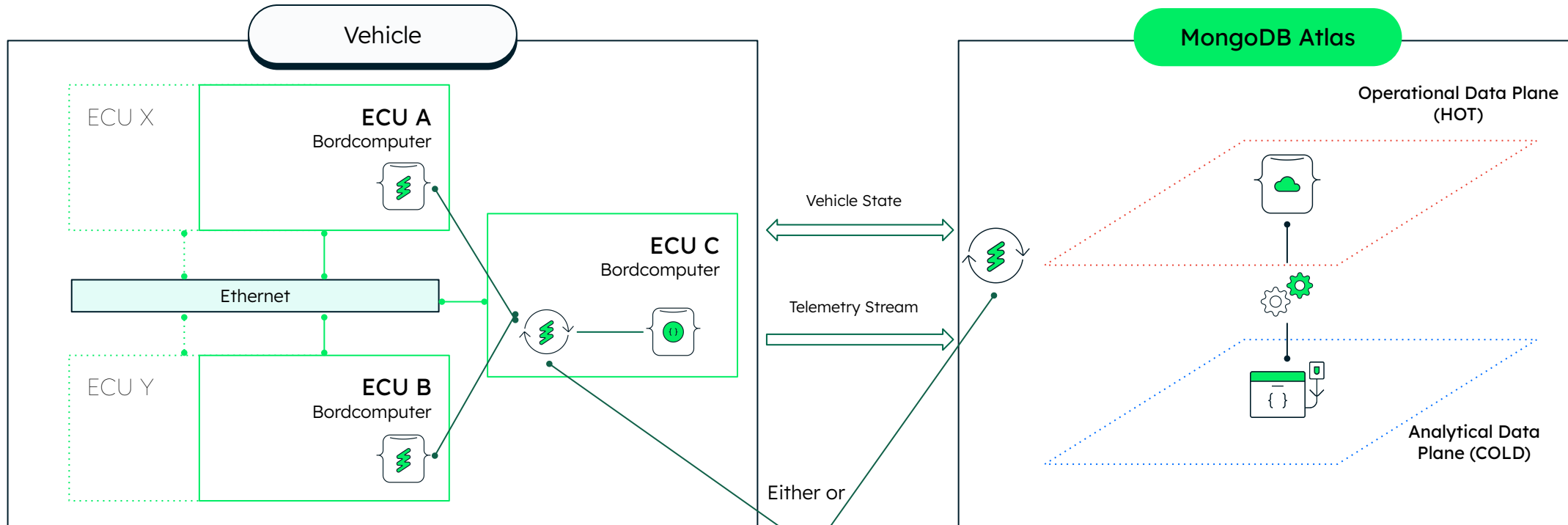


2 Server-side app updates maintenance schedule on document




3 The maintenance update is pushed down to Lot two edge server

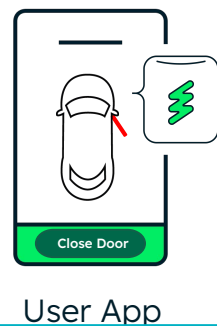


# Connected Car with Edge Server

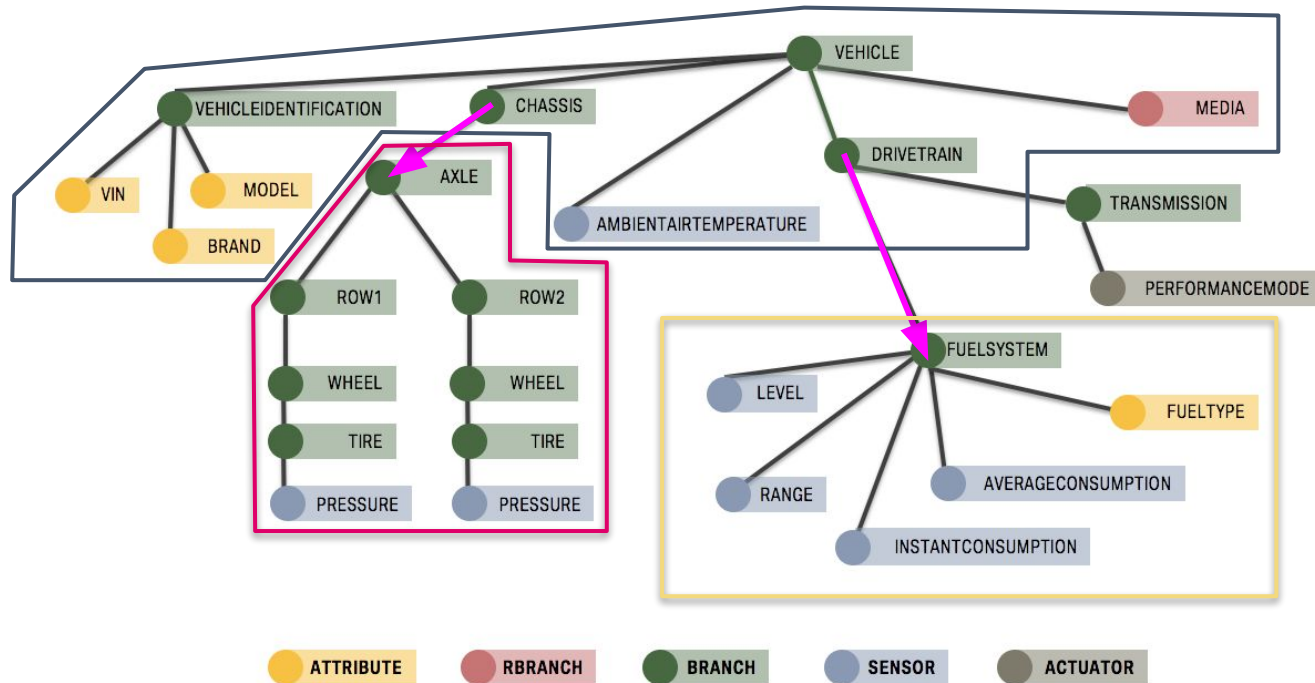
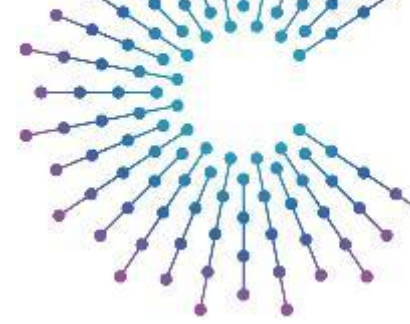


## Legend

-  **Realm** - Object Oriented lightweight DB.
-  **Device Sync** - Automatic Synchronization engine
-  **MongoDB** - Document Model based DB



# Tree of Objects



- Device (Vehicle) Class
- Component A Class
- Component B Class
- Relationships / References





# Realm Data Model & VSS in the vehicle

The VSS data model maps perfectly to Realm's object based data model

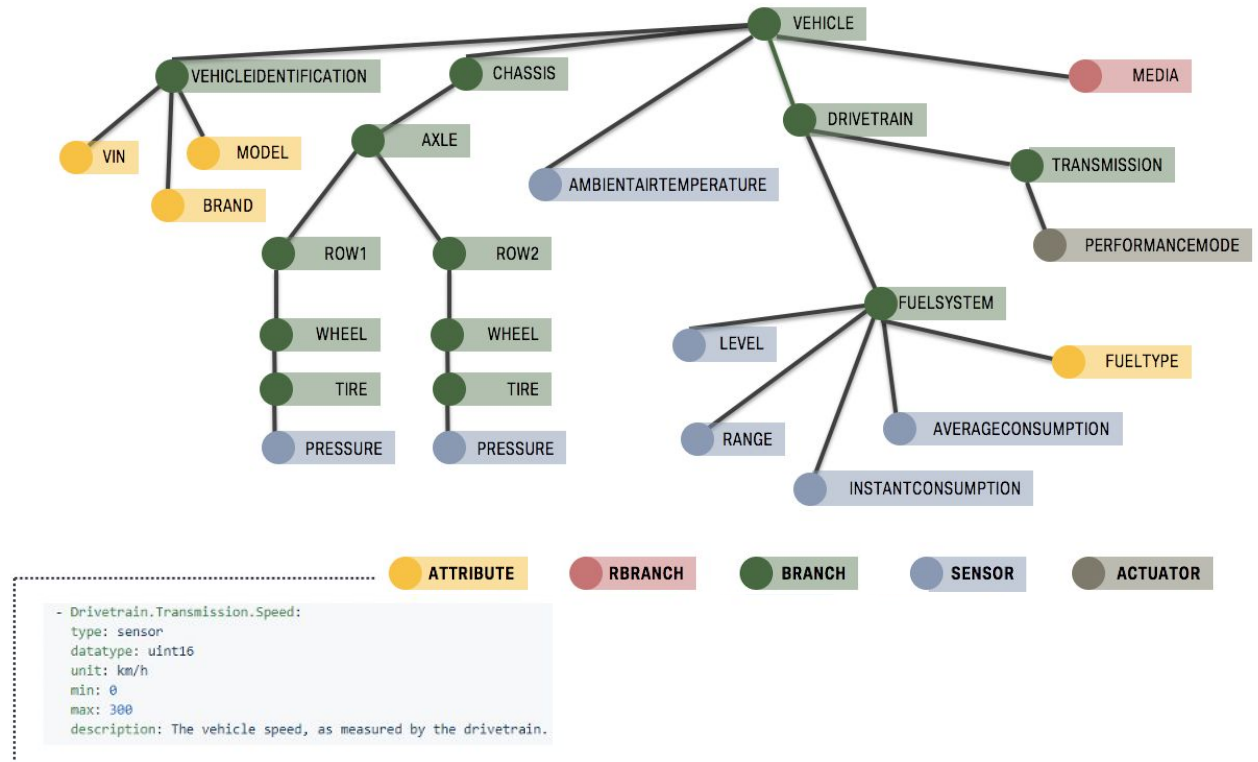
## Vehicle Model in Realm

```

Class Vehicle = {
  vehicleIdentification: {
    vin: "string",
    model: "string",
    brand: "string",
  },
  ambientAirTemperature: "int",
  driveTrain: "DriveTrain?"
  ...
}

```

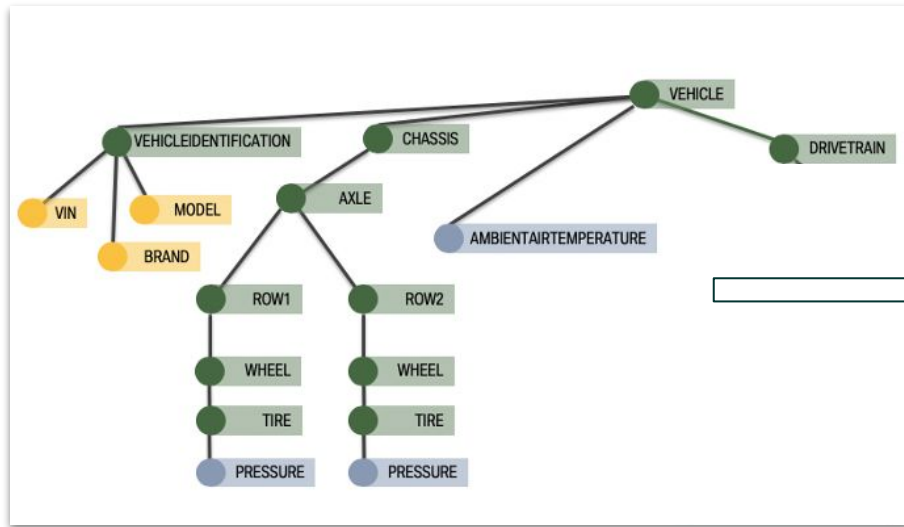
## Vehicle Model in VSS





# Realm Delta Change Notification

## Vehicle Signal Specification "VSS"



## Realm Schema & Change Notifications

```
// Realm Schema Definition
const Vehicle = {
  name: "Vehicle",
  properties: {
    vehicleIdentification: {
      vin: "string",
      brand: "string",
      model: "string",
    },
    chassis: {
      axles: "Axle[]",
    },
    ambientAirTemperature: "int",
    driveTrain: "DriveTrain?",
  },
};
```

Create delta notifications **containing only the changed attributes**

Console Output:

**1** properties have been changed.  
 Modified Attributes: {  
**ambientairtemperature: "21.0"**  
 ...  
 }

Out of the box change notifications ->



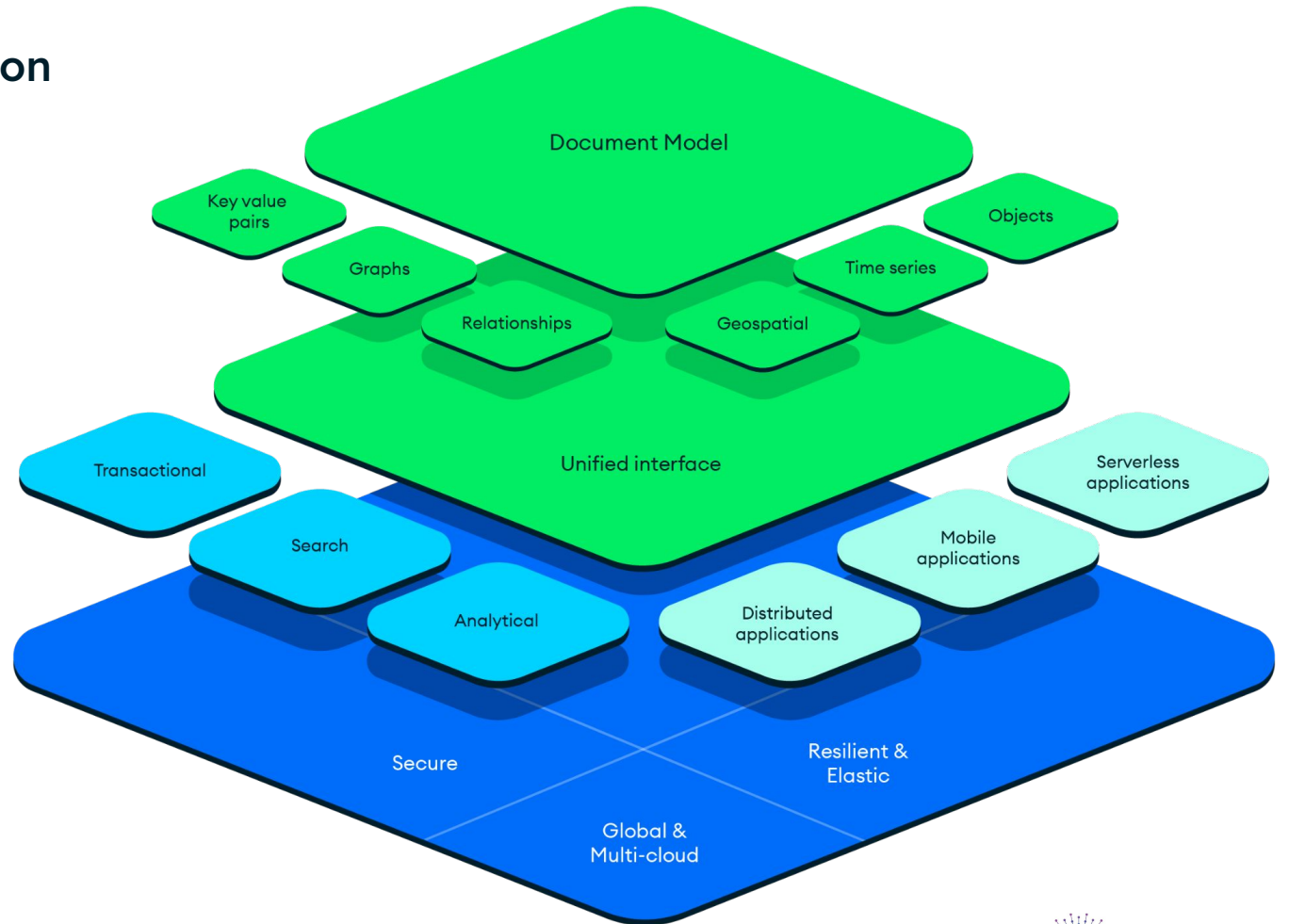
```
// Realm Object or Collection Change Listener
function listener(vehicle, changes) {
  console.log(
    `${changes.changedProperties.length} properties have been changed.`
  );
  changes.changedProperties.forEach((prop) => {
    console.log("Modified Attributes: " + `${prop}: ${vehicle[prop]}`);
  });
}
```

# MongoDB Atlas Developer Data Platform



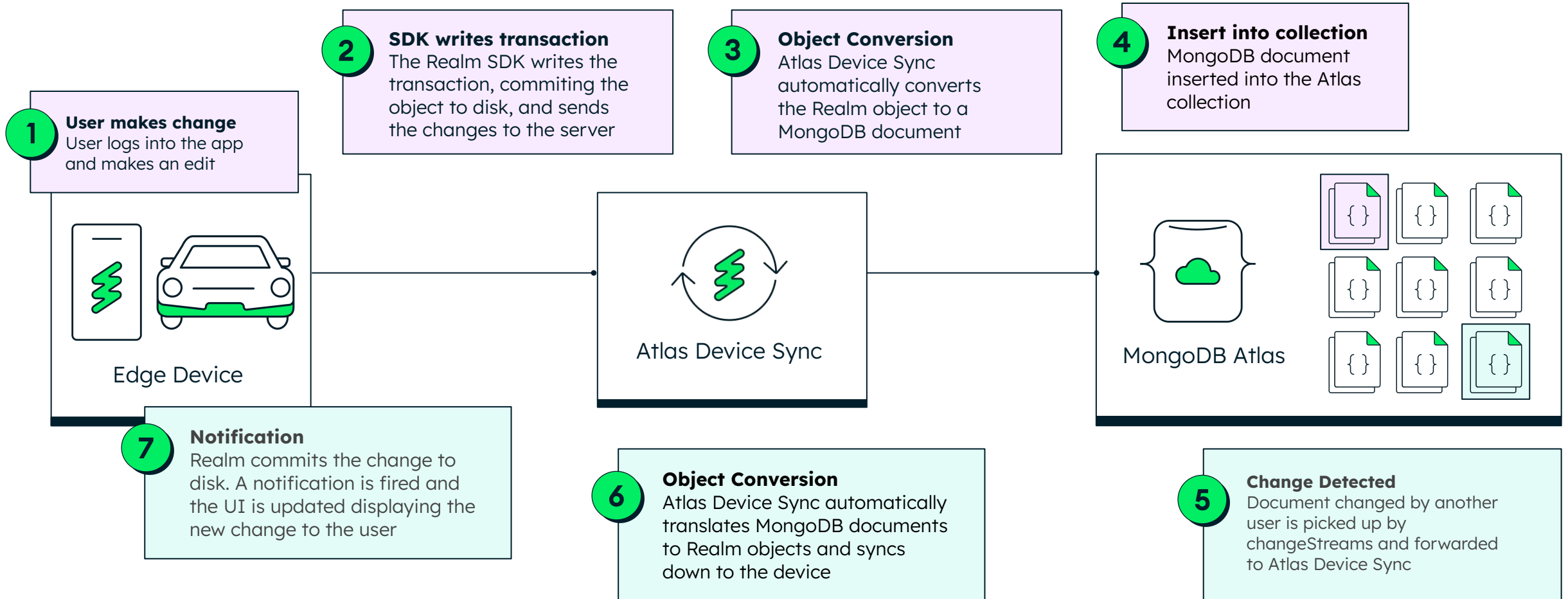
## MongoDB Atlas address a range of application use cases and removing complexity

- Flexible data model that maps to how developers think/code
- Strongly consistent, support for ACID transactions
- Able to support full-text search functionality for delivering a great user experience
- Able to support data on mobile and embedded devices w/o having to manually sync data
- Able to deliver real-time analytics on live data w/o having to move data back & forth



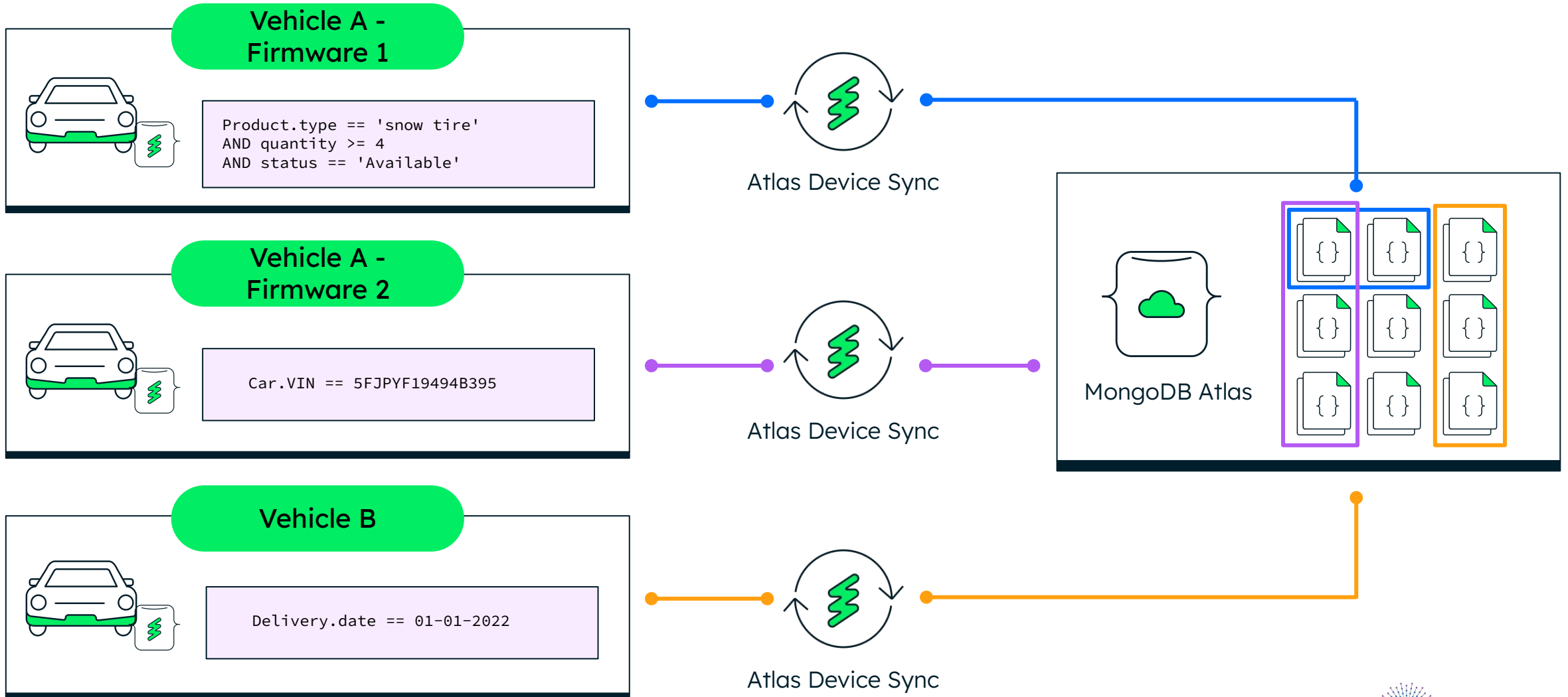


# Sync mechanism between device & cloud



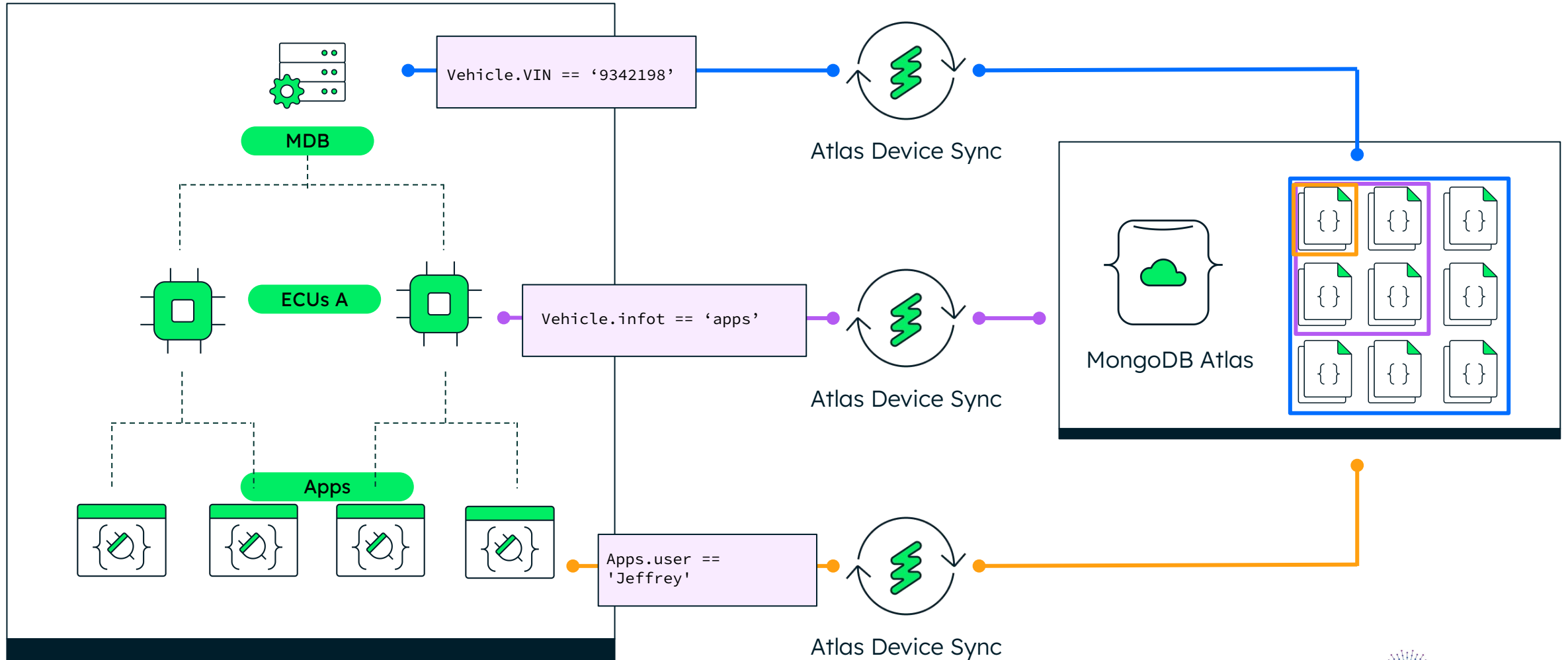


# How Device Sync works: dynamic queries



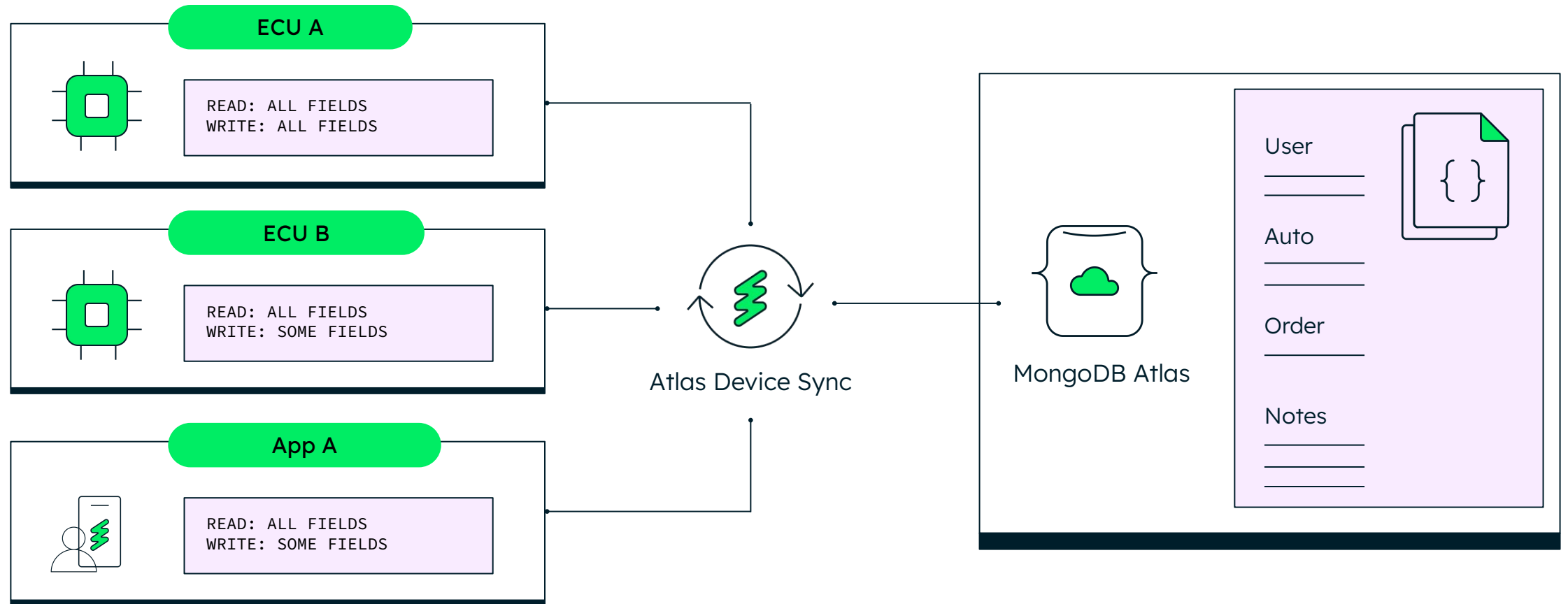


# How Device Sync works: hierarchical permissions





# How Device Sync works: document-level permissions





# How Device Sync works: field-level permissions

## Customer/Driver

```
{
  ID      "_id": {
    "$oid": "60e51a64d0ff67f998bc1732"
  },
  Address "name": "Jane Leaf",
          "address": {
            "street": "Bedford Ave"
            "city": "New York"
          }
  Billing  "billing": {
            "creditCard": "1234123412341234"
            "balance": "300"
          }
  Booking "booking": [{
            "orderDate": "May62022",
            "auto": "sedan",
            "days": "5",
            "returned": false,
          }
  Rental Notes "rentalNotes": [{
            "childSeat": true
            "description": "for my pug"
          }
}
```

## Billing

```
{
  "_id": {
    "$oid": "60e51a64d0ff67f998bc1732"
  },
  "name": "Jane Leaf",
  "address": {
    "street": "Bedford Ave"
    "city": "New York"
  }
  "billing": {
    "creditCard": "1234123412341234"
    "balance": "300"
  }
}
```

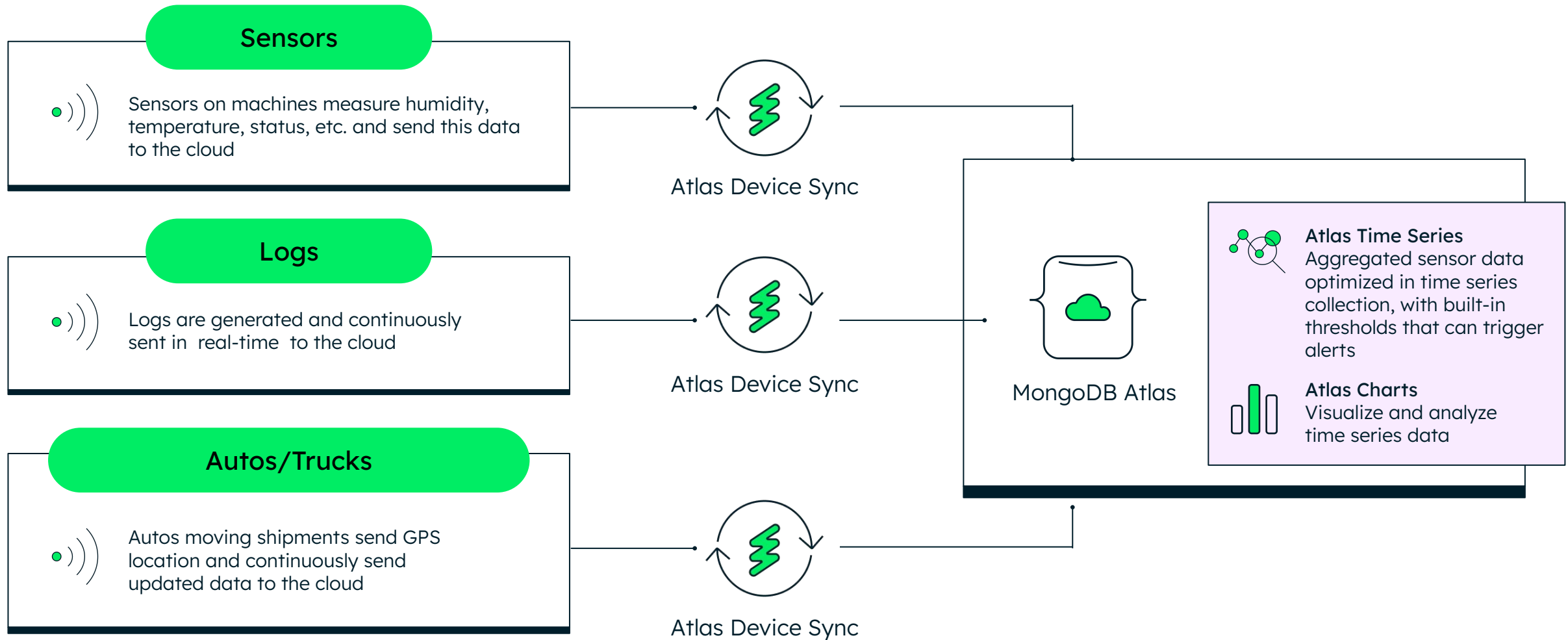
## Agent

```
{
  "_id": {
    "$oid": "60e51a64d0ff67f998bc1732"
  },
  "name": "Jane Leaf",
  "address": {
    "street": "Bedford Ave"
    "city": "New York"
  }
  "orders": [{
    "orderDate": "May62022",
    "auto": "sedan",
    "days": "5",
    "returned": false,
  }
  "rentalNotes": [{
    "childSeat": true
    "description": "for my pug"
  }
}
```

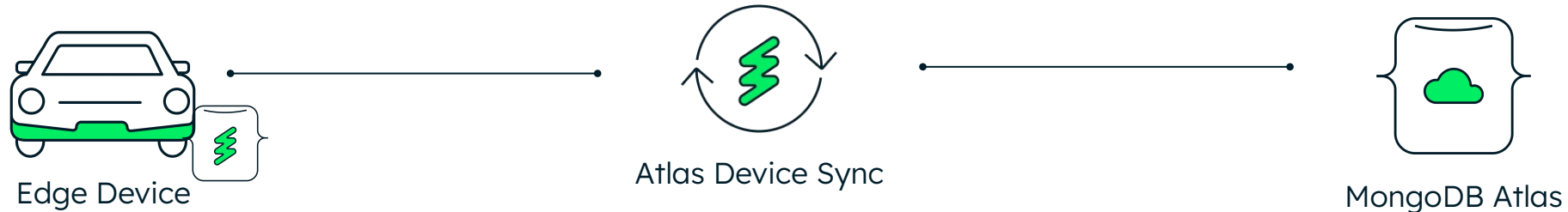




# How Device Sync works: Data Ingest



# End-to-end, security features



## On-device

**On-device encryption:** Easily encrypt database files with the encryption APIs

**Authorization:** Rich authorization methods including anonymous, email/password, API key, Custom Function, Custom JWT, and also Facebook, Google, and Apple.

## In-flight

**In-flight encryption:** All network traffic is encrypted using Transport Layer Security (TLS).

**Granular permissions:** Document and field-level access permissions are used to dynamically determine the data synced to the device.

## In the cloud

**In-cloud security:** Features that integrate with your existing protocols and compliance standards.

- Encrypted storage volumes
- Network isolation
- Role-based access management

ISO

SOC

PCI

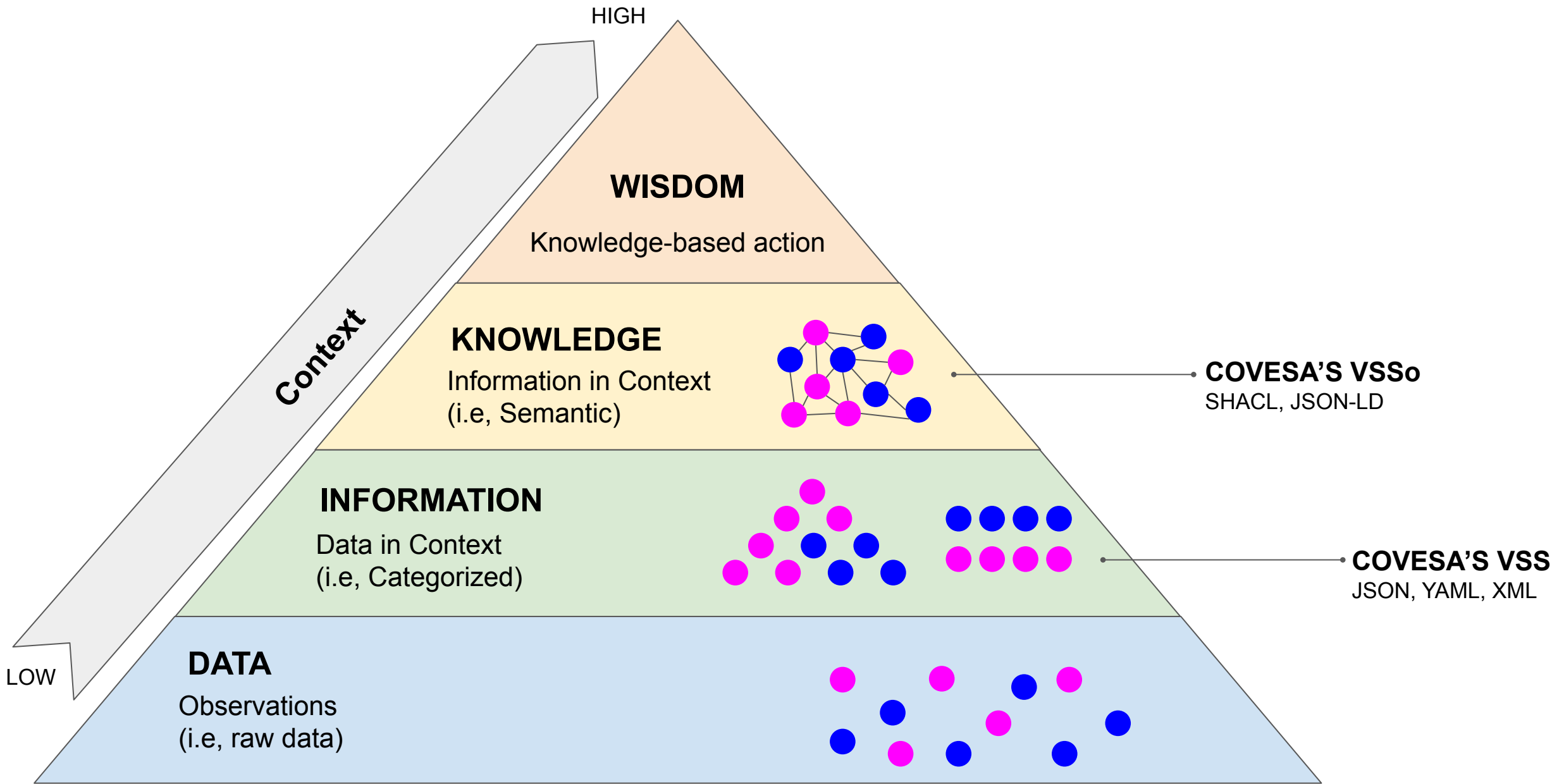
HIPAA

HITRUST

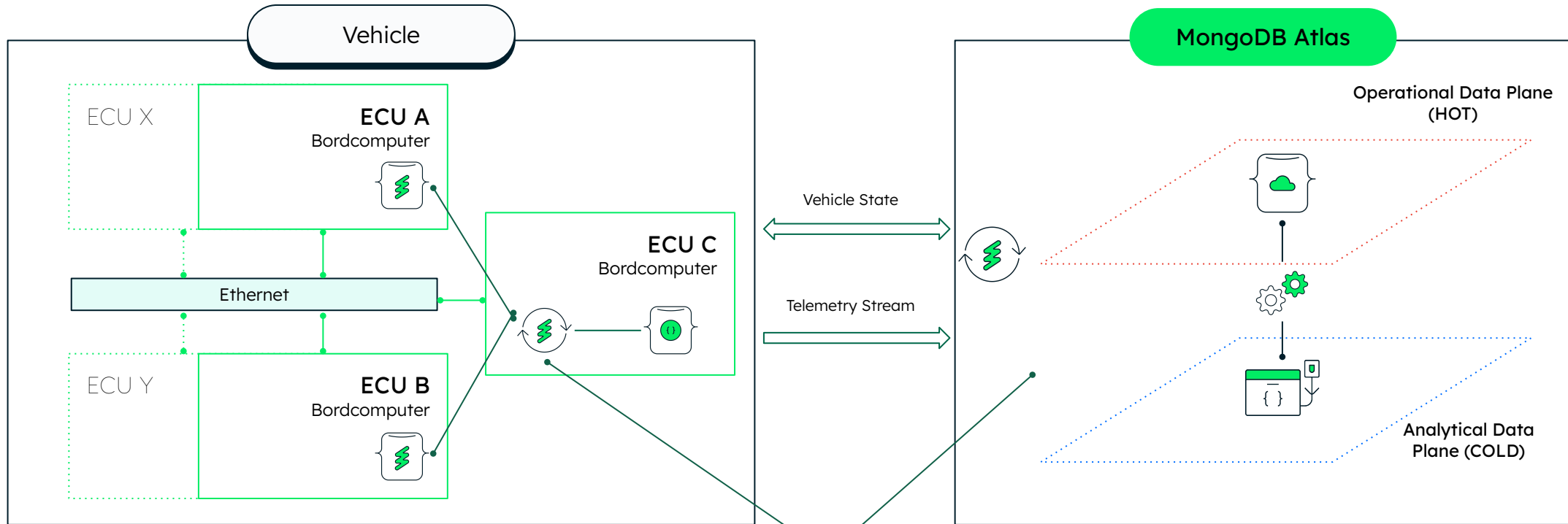
VPAT

GDPR




CSA

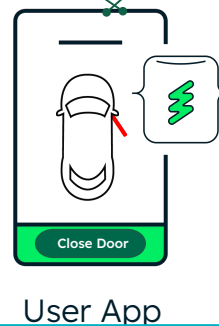


# Connected Car with Edge Server

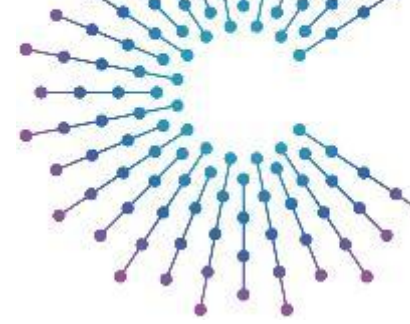


## Legend

-  **Realm** - Object Oriented lightweight DB.
-  **Device Sync** - Automatic Synchronization engine
-  **MongoDB** - Document Model based DB



# Demo time!



The image displays a multi-screen environment for a digital twin demo. On the left, a terminal window shows the execution of a Node.js application. The terminal output includes:

```
arnaldo.vera@M-WVL6CV0KKF device-ts % npm start
> device@1.0.0 start
> node build/server.js

app opened successfully
app is trying to log in
Digital-Twin app listening on port 3000
app logged in successfully
Connection[1]: Session[1]: client_reset_config = false, Realm exists = true, client reset = false
Connected to endpoint ':::1:80' (from ':::1:60339')
realm is open
subscriptions are added
Connection[1]: Session[1]: Integrated 1 changesets from pending bootstrap for query version 11, producing client version 170. 0 change sets remaining in bootstrap
{"command": "Reset Battery", "status": "submitted", "ts": "2023-10-03T15:55:54.711Z"}
{"command": "Reset Battery", "status": "inProgress", "ts": "2023-10-03T15:55:54.711Z"}
{"command": "Reset Battery", "status": "completed", "ts": "2023-10-03T15:55:54.711Z"}
ACShutdown initiated!
Realm cleaned up!
arnaldo.vera@M-WVL6CV0KKF device-ts %
```

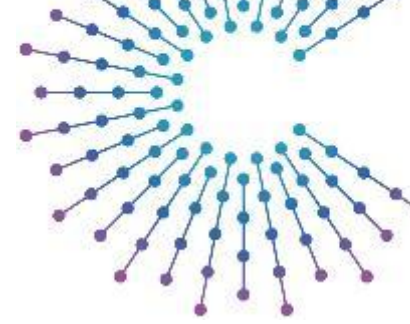
The top center window shows a web browser at localhost:3000 displaying a car simulator titled "My Car". The car has a battery icon, a voltage gauge (90), a current gauge (100), and a VIN: SUXFEB578L342684. A "Component Name" input field and an "Add Component" button are visible.

The bottom center window shows another view of the simulator with gauges for "Power Sys" (Amps: 0-100), "Braking Sys" (Volt: 0-100), "Track Velocity", and a "Bucket: 12/20" indicator.

On the right, a smartphone simulator (iPhone 14, iOS 16.2) displays a home screen with icons for Watch, Contacts, Files, Shortcuts, Freeform, and Controller.

Video Link: <https://www.youtube.com/watch?v=6qodPDPNoSg>

# Open Source



mongodb-industry-solutions / Connected-Products

Type to search

Code Issues 1 Pull requests Actions Projects Wiki Security Insights Settings

Connected-Products Public

Edit Pins Watch 1 Fork 4 Star 6

main 5 branches 0 tags

Go to file Add file Code


humza-akhtar Update README.md	929b053 on Sep 4	234 commits
atlas-backend	updated schema modification -> vin attribute removed and type of _id...	last year
aws-sagemaker	Update README.md	last year
device-cpp	test	last year
device-ts	Update config.ts	last month
media	Update EndToEndArchitecture.png	last year
mobile-swift	Update Config.xcconfig	last month
.gitignore	Update .gitignore	last year
README.md	Update README.md	last month

## README.md

### MongoDB & AWS Connected Vehicle End to End Demo Repository

Code samples and demos around using the Realm database in combination with MongoDB Atlas, Device Sync and AWS Sagemaker.

[Demo Video](#)



## About

Code samples and demos around using the Realm database in combination with MongoDB Atlas and device sync.

- Readme
- Activity
- 6 stars
- 1 watching
- 4 forks

Report repository

## Releases

No releases published

[Create a new release](#)

## Packages

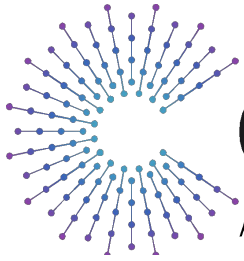
No packages published

[Publish your first package](#)

## Contributors 4

-  **felixreichenbach** Felix Reichenbach
-  **ainhoamugica**
-  **humza-akhtar**
-  **93arnaldo** Arnaldo Vera





# COVESA

Accelerating the future of connected vehicles

## Curious? -> Reach out

[arnaldo.vera@mongodb.com](mailto:arnaldo.vera@mongodb.com)

[humza.akhtar@mongodb.com](mailto:humza.akhtar@mongodb.com)

