DD2482 AUTOMATED SOFTWARE TESTING AND DEVOPS

IoT fleet management with Thingsboard.io

in a DevOps perspective

Fabian Segatz
segatz@kth.se

May 22, 2022

1 Introduction

In their mobility report from November 2021, Ericsson estimates that by 2027, there will be 30.2 billion interconnected internet of things (IoT) devices. Meaning nearly every object around us: traffic lights, thermostats, electric meters, fitness trackers, water pumps, cars, elevators, even gym vests will be connected to the internet [Ericsson, 2021].

Large amounts of IoT devices - usually referred to as fleets - are often managed by one operator. I.e. Stockholm Exergi, a local energy company in Stockholm, operates a fleet of about 9000 energy monitoring devices [Andersson, 2018]. It is rather obvious, that smart solutions for monitoring and maintaining such number of devices are needed.

3 of the 5 most valuable companies in the world (Microsoft, Alphabet & Amazon) have realized the potential of this market and each offer an **IoT device management tool**, included in their cloud services. Other than that, there are also open source tools such as Thingsboard available, that enables companies to have a scalable management of their IoT fleet and allows a DevOps style development, without being dependant on an external cloud provider.

This essay targets to present the features and capabilities that Thingsboard offers, to help you improve in a DevOps perspective. Particularly, features that help to reduce the time-to-market of IoT products, accelerate the time-to-resolution in case of error and additionally gives you relevant data for advertising your product.

2 Thingsboard

Before discussing the advantages that arise form using Thingsboard, let's have a brief look at its architecture. Figure 1 is adapted from the official Thingsboard documentation and gives a good overview on the interconnected parts.

The left side represents the client side. There can be unlimited individual devices that are connected to one Thingsboard server individually or through a gateway. Each device or gateway gets its individual login credential in form of an access token that is needed for verifying the connection. Many different transport protocols can be used of which HTTP, MQTT and CoAP are most common with IoT devices.

The right side represents the server side. The **Thingsboard UI** can be used to administrate the server or to interface with devices through dashboards. It is important to mention, that interfaces must be specified on both client and server side. The **Thingsboard Core** is the main program, which is taking care of the communication with the devices, the UI, additional databases, as well as controlling the **Rule Engine**, which is an easy to use framework for building event-based workflows. It allows to generate server side

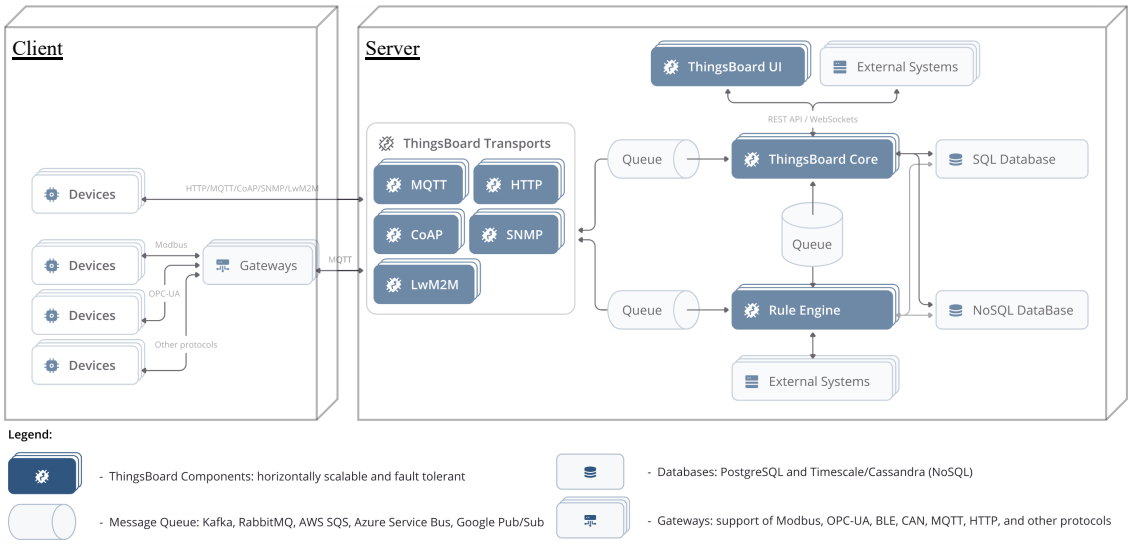

Figure 1: Architecture of Thingsboard [Thingsboard, 2022b]

applications, that interact with devices. Those rules can be triggered by messages from devices or other events.

3 Key features

Using Thingsboard or any other IoT device management platform should definitely be considered for any IoT project. There are many use cases, that can elevate the quality of your project by the means of DevOps. I found that these applications can be pooled in four categories (key features): Monitoring, Remote control, Server-side applications, Scalability. In the following sections, we'll look into each key feature and discuss in which ways they can help to improve your life as a developer and increase the satisfaction of your customer.

3.1 Monitoring

In modern software engineering it is understood that **software testing** is its own discipline and not something that is only applied ad hoc when software is not working. Testing needs a lot of expertise, planning and mostly goes side-by-side with every step of the software engineering process. In DevOps, where software life cycles usually goes beyond the initial delivery, it is of great benefit, when testing can be continued even after deployment.

Thingsboard can be used as a central point to collect test results, general monitoring data to evaluate the system's performance, as well as sensor data measured by the embedded IoT device. This data can then simply be store in a database server-sided and be used for **long-term analysis**. This is valuable i.e. to the marketing department, which can use the data for advertisement. Other than that, also the work of business analysts is dependent on such data.

Just as relevant as long term data collection is the feature of **real-time monitoring and visualization**. Thingsboard offers an advanced tool to create dashboards as part of its Thingsboard UI component. Figure 2 shows the dashboard from a smart metering demo example. Dashboards can not only be used internally, but also made public in case live telemetry needs accessible by end-users.

Ultimately, Thingsboard also allows to feed log data to an **automated pipeline process**, which is executed by the rules engine. This is useful, when data needs to be processed automatically. This will elaborate on further in section 3.3, where server-side applications are discussed.

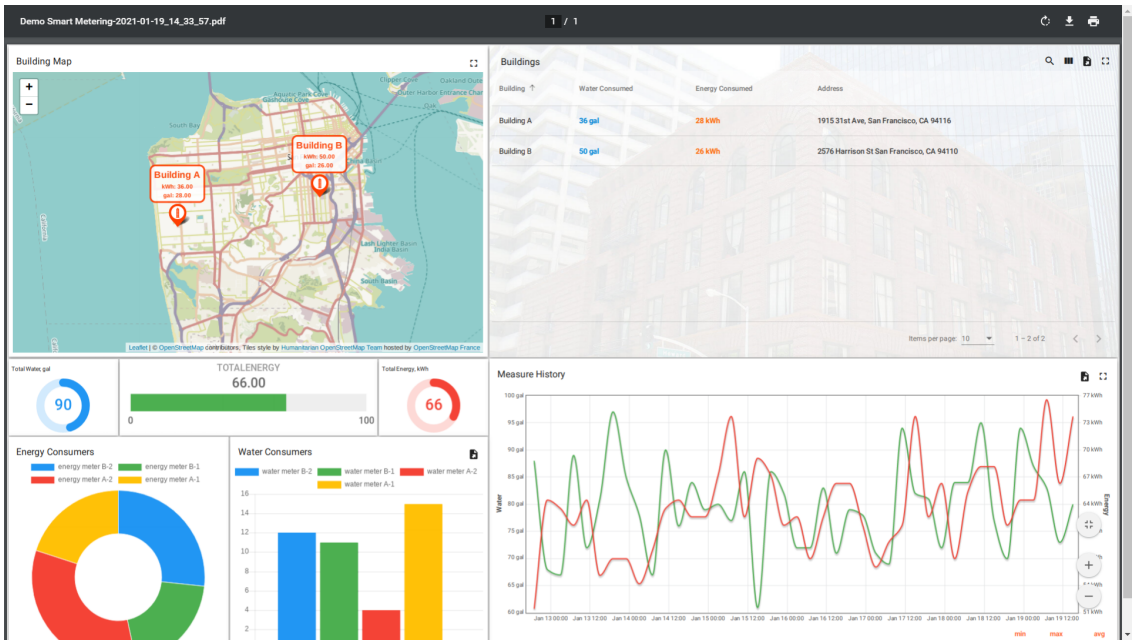

Figure 2: Smart metering demo dashboard [Thingsboard, 2022c]

3.2 Remote control

Dashboards can not only be used for data visualisation, but also to remotely control IoT devices. This allows i.e. to easily set up a control panel for **feature flags**.

This remote control capability can even get used for software and firmware **deployment** on connected devices. This is very important to ensure a short meantime-to-resolution. Software bugs can easily be fixed remotely, which is effectively reducing downtime, as well as travel- or transport cost. Moreover, it's relevant from a **security** perspective. With the rise in popularity and the increase of IoT usage, there is also a growing need to develop new methods to cope with security concerns.

The DevOps philosophy answers the security topic in a novel way: Since there is no possibility to know every vulnerability from a beginning and create a perfect secure system, it is rather important to resolve security issues with minimal downtime to maximize customer satisfaction. Remote deployment allows to patch weak points in the fastest way possible.

One main difficulty when developing embedded IoT products is, that they are mostly used in a very specific context. Thus, it is not possible to just create a product and then sell it to multiple clients. Instead, there is often a baseline product that must be adjusted to the individual client needs. With Thingsboard, it is possible to reduce the time-to-deployment, because products can be deployed with their core functionality immediately and then later be adjusted through remote deployment.

3.3 Server-side applications

Modern software products are often delivered as **software as a service (SaaS)**. This means that clients don't need to operate the software themselves. Instead they pay a usage subscription fee to the operator, which is often the developer of the software. The operator needs to make sure, that the product is working at anytime and adjust it according to the customers individual needs. Thingsboard is the base point for extending a companies product line to SaaS products. An example use case would be online image classification. Embedded computers usually don't have the computational power to do even slightly complicated calculations in real-time. Often they are optimized on energy consumption. Luckily, Thingsboard's rule engine allows to set up server-sided pipelines that can be triggered by incoming telemetry messages from

devices. Thus, one could create a pipeline, that gets triggered by incoming image data. The image is then processed by a server-sided classification algorithm before the result gets sent back to the device.

A typical pipeline created with the rules engine is sketched in figure 3. There are many predefined nodes that can be combined via an easy graphical user interface. Special functionality can be coded in JavaScript into script nodes (see node B and D).

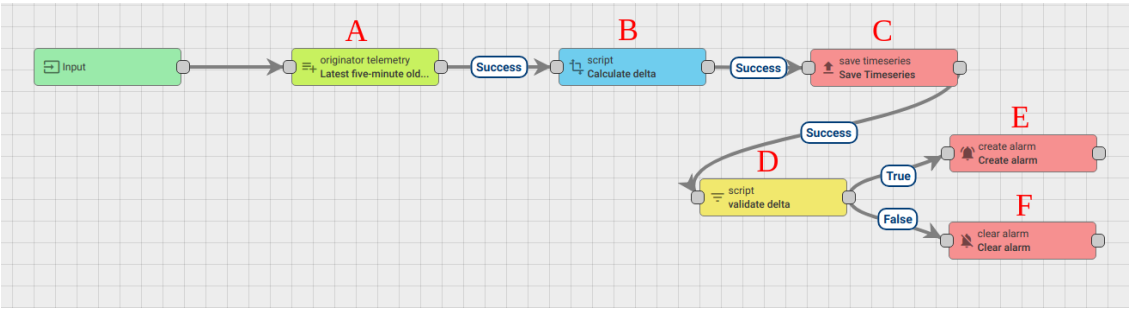

Figure 3: Example pipeline created with Thingsboard's rule engine [Thingsboard, 2022a]

Server-sided applications can be interesting to companies specialised in IoT for multiple reasons. Again security concerns can be a driving force behind storing and **running business logic server-sided**. Especially with mainstream products, that are sold in large amounts, measurements need to be taken to prevent reverse engineering and copying of the products by malicious agents.

Other than that it can also help to make production cost for the devices cheaper. Instead of integrating storage-hardware on the IoT device itself, Thingsboard allows to use **online storage** on servers instead.

Relevant is also the use-case for **automatic anomaly detection**. Monitoring data and analysing it manually is great, but sometimes it just has to go faster. "Machine learning techniques enable the development of anomaly detection algorithms that are non-parametric, adaptive to changes in the characteristics of normal behaviour" [Omar et al., 2013]. Such as previously stated, are embedded IoT devices not so well suited for running machine learning applications, thus running them server-sided is essential for running them at all. They allow to automatically take predefined counter measures, like disabling the attacked device or just inform the operator to decide on individual actions.

In a way this can be described as a type of **software bot**, which is a classic DevOps topic. Software bots can also be used in other ways. IoT devices are often used to monitor a process. The Thingsboard rule engine can be used to easily notify operators or end users when anything relevant happens.

Even other devices could be notified, which can than i.e. be triggered to start a new process automatically themselves. Thingsboard allows operate and maintain **relationships** between interconnect fleets of IoT devices easily. I.e. when two devices are dependent on each other and one of them fails, the rules engine allows to easily modify the relationship to replace the broken device with a new one, without the need to change firmware on the still working device - again effectively reducing the meantime-to-resolution.

3.4 Scalability

4 Conclusion

References

[Andersson, 2018] Andersson, A. (2018). Stockholm exergi och telia i samarbete för effektivare energisystem i smarta fastigheter - stockholm exergi. https://www.stockholmexergi.se/nyheter/stockholm-exergi-och-telia-i-samarbete-for-effektivare-energisystem-i-smarta-fastigheter/. Accessed: May 17, 2022.

[Ericsson, 2021] Ericsson (2021). Broadband iot still rising as 2g and 3g continue to decline. *Ericsson Mobility Report November 2021*, page 18.

[Omar et al., 2013] Omar, S., Ngadi, M., Jebur, H., and Benqdara, S. (2013). Machine learning techniques for anomaly detection: An overview. *International Journal of Computer Applications*, 79.

[Thingsboard, 2022a] Thingsboard (2022a). Telemetry data validation. https://thingsboard.io/docs/user-guide/rule-engine-2-0/tutorials/telemetry-delta-validation/. Accessed: May 19, 2022.

[Thingsboard, 2022b] Thingsboard (2022b). What is thingsboard? https://thingsboard.io/docs/getting-started-guides/what-is-thingsboard/. Accessed: May 17, 2022.

[Thingsboard, 2022c] Thingsboard (2022c). Working with iot dashboards. https://thingsboard.io/docs/user-guide/dashboards/. Accessed: May 19, 2022.