



Build Your Own IoT Platform

Develop a Fully Flexible and
Scalable Internet of Things
Platform in 24 Hours

Anand Tamboli

Apress®

Build Your Own IoT Platform

**Develop a Fully Flexible and
Scalable Internet of Things
Platform in 24 Hours**

Anand Tamboli

Apress®

Build Your Own IoT Platform

Anand Tamboli
Sydney, NSW, Australia

ISBN-13 (pbk): 978-1-4842-4497-5
<https://doi.org/10.1007/978-1-4842-4498-2>

ISBN-13 (electronic): 978-1-4842-4498-2

Copyright © 2019 by Anand Tamboli

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director, Apress Media LLC: Welmoed Spahr

Acquisitions Editor: Nikhil Karkal

Development Editor: Siddhi Chavan

Coordinating Editor: Divya Modi

Cover designed by eStudioCalamar

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit <http://www.apress.com/rights-permissions>.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at <http://www.apress.com/bulk-sales>.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-4497-5. For more detailed information, please visit <http://www.apress.com/source-code>.

Printed on acid-free paper

Table of Contents

About the Author	ix
About the Technical Reviewer	xi
Acknowledgments	xiii
Introduction	xv
 Chapter 1: So... You Want to Build Your Own!.....	 1
The Background of IoT and Our Focus.....	1
How Many Platforms Are Out There?	3
Platforms Supporting Network Servicing	3
Platforms Sitting Between Networks and Applications	3
Application-Layer Development Platforms	4
What Should a Good IoT Platform Have?	4
Why Should You Build Your Own IoT Platform?	8
Summary.....	9
 Chapter 2: The Building Blocks of an IoT Solution	 11
The Functional Blocks of an IoT Solution	12
The Detailed Block Diagram of an IoT Platform.....	14
Is Everything from this Block Architecture Mandatory?	21
What Is the Proposed Approach?	22
Summary.....	22

TABLE OF CONTENTS

Chapter 3: The Essentials for Building Your Own Platform	23
Deciding Cloud Instance Specifics.....	23
Additional Specifications	25
Where Do We Get this Cloud Instance?	25
What About Our Own Machine?	26
Expanding on the IoT Platform Block Diagram.....	27
Edge Interface, Message Broker, and Message Bus.....	27
Message Router and Communications Management.....	28
Time-Series Storage and Data Management.....	28
REST API Interface	28
Microservices	28
Rule Engine	28
Device Manager and Application Manager.....	29
Our Own IoT Platform Block Diagram	29
Summary.....	30
Chapter 4: Let's Create Our Platform Wish List	31
Connecting with the Platform in Real Time.....	31
Using MQTT as the Message Broker.....	32
How Do We Want to Store the Data?	33
Data Storage Schema.....	34
Accessing Platform Resources Through APIs.....	36
Data Accessing APIs.....	37
Elementary Microservices and Utilities.....	39
Routing and Filtering Data and Messages	41
Updated Block Diagram of Our IoT Platform	42
Summary.....	42

Chapter 5: Here We Go!	43
Initializing the Cloud Instance	43
Register and Create	44
Choosing an Operating System Image	45
Choosing the Size	46
Choosing a Datacenter Region	47
Finalizing and Creating the Instance	49
Connecting to Our Cloud Instance	50
Installing Basic Software Stacks	52
Installing Apache	54
Installing MySQL	56
Installing PHP	59
Securing the Instance and Software	62
It's Easier with a Domain Name	66
Add Virtual Hosts to Our Web Server	69
Installing SSL Certificates	71
Installing Node.js and Node-RED	75
Modifying Node-RED Settings	77
Securing our Node-RED Editor	80
Summary	83
Chapter 6: The Message Broker	85
What Is MQTT?	85
Publish and Subscribe Paradigm	86
Other Features of a Message Broker and MQTT	88
Quality of Service	89
Keep Alive Period	90
Last Will and Testament	91
The Retained Message	92

TABLE OF CONTENTS

The Best Part: WebSocket.....93

Are We Using the Best Message Broker Option?94

When to Utilize a Message Broker and When Not To95

Installing a Message Broker96

 Securing a Message Broker98

Summary..... 103

Chapter 7: Building the Critical Components.....105

 Creating a Time-Series Core Database 105

 Installing Required Nodes in Node-RED 108

 Creating First Flow for Our Platform 109

 Adding MQTT Publish Capability..... 110

 REST API Message Publisher 112

 Creating the Database Listener..... 115

 REST API Message Retriever..... 118

 Verifying that Everything Is Working as Expected..... 121

 Running Node-RED in the Background Continuously..... 122

 Summary..... 123

Chapter 8: Configuring the Message Broker.....125

 The Difference Between WebSocket and Normal MQTT 125

 Why Is WebSocket Important? 126

 Adding WebSocket to Our MQTT Configuration 126

 Testing WebSocket 128

 Let’s Add User Access Controls..... 129

 Let’s Check If This Is Working..... 132

 Using the Forever Tool with the Message Broker..... 134

 Summary..... 135

Chapter 9: Creating a REST Interface	137
Data Access APIs.....	137
Adding Time-Based Filters	141
Data Deletion APIs.....	145
Removing Data Records Completely	150
Adding Microservices to the Platform.....	152
Getting the Current Timestamp	153
Random Code Generator.....	154
Adding New Modules to Node-RED	155
UUID Generator	156
Email and Text Message Microservice APIs	158
Configuration of Nodes.....	158
SMS Sending Utility	159
Email-Sending Utility	161
Summary.....	163
Chapter 10: Rule Engine and Authentication	165
Start with the Rule Engine Logic.....	165
Creating a Database	166
Building the Flow Sequence	168
Testing the Rule Engine	170
Rule Management APIs	171
Enable and Disable a Specific Rule	172
Enable and Disable All Rules	173
Create a New Rule	174
Building Another Rule Engine with Node-RED	176
Adding Authentication to the Data API	178
What Are Our Options?	178
What Is the Plan?.....	180

TABLE OF CONTENTS

Adding Authentication Middleware.....	181
Enable and Test Authentication	185
Our Core Platform Is Ready Now.....	186
Summary.....	187
Chapter 11: Documentation and Testing.....	189
Preparing a Valid OpenAPI Specification Document.....	190
Platform API Specification File Explained	191
Preparing Distribution Package for Final Upload.....	194
Upload API Docs and Make It Live	195
Authorize and Test API.....	196
Summary.....	198
Chapter 12: What We Built and the Takeaways	199
Increasing Security for the Cloud Instance	200
What About SQL Injection Through APIs?	202
Should We Have Used MongoDB Instead of MySQL?	203
Some Experts Might Still Try to Talk You Out of This	204
How Is Our Platform Different from AWS, Google, and Azure?	205
There Is a New Version of MQTT	207
My Platform Is Ready. Now What?	208
The Next Big Thing	208
If You Need to Find More Resources	209
Finally... ..	209
Glossary.....	211
References.....	213
Index.....	215

About the Author



Anand Tamboli has loved solving problems in smarter ways since his childhood. As life progressed, he started to do that at scale with his entrepreneurial mindset.

Anand is a versatile professional, a seasoned entrepreneur, and creator of many innovative products & services. With his cross-domain and multi-industry experiential knowledge, Anand sees things with a uniquely different lens.

With his profound understanding of disruption and business transformation, Anand concentrates on solving customer problems while utilizing the latest technology advancements. However, he strongly believes that technology is not a panacea; it rather takes the right combination of people, process, technology, and timing to achieve the best results.

With numerous innovative products & services deployed in last 20+ years, Anand has garnered deep expertise in business improvement, business transformation, data science, IoT, cognitive systems, machine learning, artificial intelligence, etc.

Anand helps businesses to improve their key metrics by finding and solving meaningful problems in innovative ways. He constantly evangelizes and inspires people for the emerging future and encourages non-linear thinking. Sane and sensible adoption of technology remains his area of focus. Reach him at <https://www.anandtamboli.com/linkedin?9781484244975>.

About the Technical Reviewer

Name: Contributing Editor

Just another geek playing with digital and analog software and hardware toys since the late 1960s.

Acknowledgments

I am grateful to my entire family: my son, daughter, and wife, who are my daily sources of inspiration and energy. My humble regards to my father, who nurtured my love of books since childhood; my mother, who supported my quests; and my sister and all my in-laws, who have encouraged and supported me for all my endeavors.

My sincere thanks to the Apress team—Nikhil, Divya, and Matthew—for their continued support and helping to bring this book to life.

Special thanks to Frank and Siddhi for their careful review and helpful feedback.

Thanks to my friends who helped in the initial review of a few chapters with their valuable feedback—Peter Vinogradoff, Prof. Pradnya Kulkarni, and Prof. Dipalee Rane. Their input helped with shaping and making this book more suitable to the target audience.

Introduction

If you search for “IoT platform” on Google, it will return about 190 million results within a second. This is the level of proliferation that IoT has achieved (especially IoT platforms) in recent years. Every solution that is related to the IoT needs a platform.

Whether you develop a custom platform or buy it off the shelf means a lot to your final product. Moreover, the term *IoT platform* has many connotations, and vendors have overused it to a point where it does not convey anything meaningful.

As businesses and working scenarios are evolving, I am seeing many smaller companies delving into IoT. However, not having your own IoT platform is one of the impediments for such an evolution. The easy/lazy answer, as many would suggest, is to use freemium or free-trial platforms.

What lies ahead is a greater challenge when things scale and costs skyrocket exponentially. When the trial expires or freemium is not enough, users find themselves locked-in, and switching over is neither simpler nor easier.

Additionally, buying off-the-shelf solution often means that you subordinate requirements or retrofit things to suit what is available. You might end up building a subpar solution, if not an outright bad one. If having full flexibility and control means something to you, this book is for you.

I chose to write this book as I saw many of my customers struggling to understand the IoT platform landscape. State of the play has not been balanced with many vendors convoluting the offering to make it look like the greatest thing ever built. For short-term gains, they have raised artificial constraints and showed superficial problems that only their offering can solve. I believe in empowering customers, and this book is a humble attempt to do it.

INTRODUCTION

The book is not about building a full-blown enterprise-grade system. It is about being agile in a true sense and reducing time to the market without breaking the bank. It is about designing something that you can scale incrementally without having to do a lot of rework or disrupting your current state of the work.

If you are a small to medium-sized company, or part of the development team at a non-IT company, you will find this book quite useful. If you are an independent developer, researcher, or learner, you will see the usefulness of the content for your endeavors too. Whether you are new to the programming world or have basic to intermediate programming skills, you will find this hands-on book equally useful.

The book supports the idea of being frugal at the start, and then invests only when and where necessary. It would help you to tap into technology advancements without bank-breaking budgets, and get off the ground quickly, contrary to the longer times required to adapt to the off the shelf or freemium platforms. More importantly, you will be in full control of what you are developing throughout the process.

Throughout 12 chapters of this book, I guide you through the step-by-step process of building your own IoT platform. There are must-haves and there are nice-to-haves; I will distinguish between the two and focus on how to build the must-haves. You will not only save heaps but also enjoy a control-wielding and satisfying learning experience.

In the first chapter, I discuss the necessary and sufficient qualities that any IoT platform must have and why. I also elaborate on the key question of why you should build your own.

Building your own means understanding at the ecosystem level is important; we do that in Chapter 2, where block diagram-level details of the IoT platform are discussed.

Better planning is a key to success that reduces confusion and agony later on. So, I cover a platform wish list, and the technical and general requirements for the building of our platform in Chapters 3 and 4.

The rubber actually hits the road in Chapter 5, where we initialize the cloud instance, install the required software stack, and apply security. If you are eager to jump into the “how” of building things, this is where you might want to start (and read about the “why” later).

One of the core elements of the platform is a two-way messaging system bus, which is explained in Chapter 6 along with the installation of broker software and securing it.

Building critical components of the platform, and the message broker extension with additional functionality, are covered in Chapter 7. Additional configurations and testing the core built to that point are covered in Chapter 8.

In Chapter 9, additional microservices and data access APIs are covered, along with the foundation for the rule engine. Then we build a full rule engine and authentication mechanism in Chapter 10.

In Chapter 11, we add documentation and provide the testing facility for developers with interactive API documentation.

Finally, in Chapter 15, I address a few commonly asked questions in various forums and discuss a few advancements that are in progress, which you might want to add to the platform when you build it. As I conclude, I leave you with a few possibilities to experiment.

Remember that all the code and configuration files discussed in this book are available on GitHub at <https://github.com/knewron-technologies/in24hrs>. Feel free to star, fork, or download them as you wish, and if you have more to add or suggest, I will be happy to hear from you.

I wish you all the best on this interesting journey and sincerely hope that you will enjoy the book as much as I enjoyed writing it!

CHAPTER 1

So... You Want to Build Your Own!

It's good that you are keen on building your own IoT platform, or at least you are interested about knowing what it takes to build one. For either reason, it is important to understand what an IoT platform essentially means in the general sense. First, let's look at what *IoT* means.

In this chapter, I briefly touch upon IoT's background and building our own platform in this book. I discuss the following:

- The types of IoT platforms
- The characteristics of a good IoT platform
- Why you should build your own IoT platform

The Background of IoT and Our Focus

The Internet of Things, a.k.a. IoT, is the network of physical devices, such as appliances, smartphones, vehicles, street lights, infrastructure elements, industrial machines, and so forth, which are also known as *things* (the T in IoT).

While working for The Procter & Gamble Company, Kevin Ashton coined the term *Internet of Things* (although he preferred the phrase Internet *for* Things).

At the outset, it was merely an exercise to identify physical objects, or things, with the help of RFID tags, and then using that information in software systems. Things have evolved since then. Several changes and ideas contributed to shaping the scope of IoT into something larger.

Today, IoT is a combination of physical objects that have some sort of computing power, some level of intelligence built into the object itself, media through which the object can be connected to the Internet ecosystem, and then the whole computing machinery of the Internet—going all the way to user devices and computers.

From the IoT platform perspective, our focus will be on where physical objects first meet the Internet and the magic that happens before software and applications take control.

These platforms are often termed as *middleware software* because they sit right in the middle of two heterogeneous things: physical objects and digital systems. Middleware is usually a mix of a high and a low level of logic, also incorporating the mixture of high- and low-level languages to accomplish the task.

You should be mindful of the fact that we are not going to build a full-blown, enterprise-grade IoT platform with all the bells and whistles. Instead, we will be agile and focus on reducing the time to market without breaking the bank. We will aim to design something that we can scale incrementally without having to do a lot of rework and potentially disrupting the current state of the work.

While there is no strict definition for what we can call an IoT platform, there is a general expectation that the platform will help high-level software, applications, and systems interact with lower-level protocols, methods of communication, and heterogeneous objects overall. This type of broad definition or understanding often means that there are far too many things that could fit in this criterion.

The IoT platform is one of the vital parts of an entire IoT ecosystem, and the term has become quite confusing due to marketing gimmicks and vendor proliferation.

How Many Platforms Are Out There?

Today, there are more than 500 platforms of various shapes and sizes, and this number will only keep growing. It is interesting to note that many of the platforms are losing their charm, so they are shutting down or merging with others. At the same time, a few platforms are morphing into more futuristic and powerful ones. In short, changes are happening in both directions.

An overview of these platforms shows that we can categorize all of them in three core types.

Platforms Supporting Network Servicing

These platforms support network servicing parts, such as MAC layer communication decoders and converters. These platforms essentially control and coordinate the telecommunications part of things. A good example is a network server for LoRaWAN communications. These platforms convert radio-level communication into raw data information and pass it on to upstream platforms or applications for further processing.

In addition to network service platforms, there are a few other parts, such as identity and key management services, and combinations of these.

Platforms Sitting Between Networks and Applications

These platforms support processing post network and pre-application, such as system-level protocol decoding, converting, decrypting, and so forth. These platforms can control and coordinate protocols and overall communication orchestration. They also support driver-level logic and the underlying architecture of the overall system that depends on them. We can treat them as the core plumbing of the system, which is what we will be building throughout this book.

Application-Layer Development Platforms

There are platforms that support high-level developments on the cloud. Most of these platforms help in the integration of multiple middleware platforms, other systems—such as ERPs and CRMs, and similar applications. The difference between this type of platform and the other two (network and middleware) is if the high-level platform fails, the other two will still function and may support parts of the high-level platform that are still working. On the contrary, if the network or middleware platform fails, there can be downtime for the overall system and solution.

Given that we have so many types of platforms and too many options available in the market, it is very important that we define what a good IoT middleware platform should have in it.

What Should a Good IoT Platform Have?

For every product, there are functions and features that are must-have or are nice to have. When we distinguish between the two, the answer is relatively simple. Building your own IoT platform makes much more sense. For any middleware platform to be worthy of being part of the Internet of Things, it is imperative that it has the following functionalities and capabilities.

- *Scalability.* Just like any new application or product, things start small and then grow later. Therefore, if the middleware platform must be at the core of the solution, it must be able to scale in the same proportion. It should not be a one-click change, which is okay; however, it should be reasonably easy to scale the platform without breaking existing functionalities and without disrupting existing production setup.

- *Reliability.* In general, it is an obvious expectation that anything that forms the core of a solution or product should be reliable. The level of redundancy built into the middleware slightly varies, depending on the end application, product, or industry vertical. For example, if the IoT platform is for medical devices, financial services, or security systems, the level of reliability expected is relatively high when compared to one for home appliances like coffee machine or similar others.
- *Customization.* Since we are building our own platform, it can be 100% customized; however, even if you were looking to buy off the shelf, customization without breaking the bank should be possible. If you cannot customize the middleware, then you have to modify your product or service to be fit for the platform, which is essentially working in the reverse direction.
- *Supported protocols and interfaces.* By fundamental definition, an IoT middleware platform sits between two heterogeneous systems: physical devices and cloud software (and there are umpteen numbers of device types and software). The platform should be able to coordinate with all of them, orchestrate things in unison, and speak all of the languages or protocols. Additionally, it needs the ability to create the required plugin and fill the gap whenever required, such that the middleware platform remains accommodating, for a very long time, before needing an overhaul.

- *Hardware agnostic.* The Internet of Things is essentially a group of heterogeneous connected things, hardware devices, computer systems, and software. This makes the requirement of being hardware-agnostic almost obvious. The reason why it still needs to be explicitly stated is due to a slightly skewed view. Many people think of hardware as an electronics circuit for a sensor, and for that view, we say that an IoT platform should be agnostic of whatever electronics you are using in your circuit. Whether it is an open source hardware design, proprietary circuit, or a mix, the platform should be able to support it.
- *Cloud agnostic.* Similar to being hardware agnostic, the platform also needs to be cloud agnostic. There are several cloud service providers—including Google, Microsoft, and Amazon Web Services (AWS)—but the platform should have no dependency on the cloud. Whether its your own service or a third-party cloud running behind a NAS (network-attached storage), the platform should be able to work. A good test of compliance is an answer to the question of whether the platform works on bare-metal servers. That is, if you get a virtual private server instance and install the platform, will it work? The answer should be a simple yes, which means the IoT platform is cloud agnostic.
- *Architecture and technology stack.* A well-defined architecture and the appropriate combination of the technology stack is a key thing that differentiates a good IoT platform from the others. The platform may be built on a rather weird combination of technologies that are not known for working together nicely. Maybe

the technology used is going to be deprecated in next few years, especially during the operational timespan of your usage. If this is the case, you should stay away from it. The same goes for the architecture, or the so-called “plumbing” of the middleware. If the architecture is not flexible enough for future changes, that is a red flag. A completely fluid architecture is not a good fit either. You need a good combination of a fluid and a rigid architecture backed by a solid, efficient technology stack.

- *Security.* Over the last several years, the Internet of Things has become a laughing stock, mainly due to poorly managed security aspects in far too many applications and IoT solutions. The saying, “The S in IoT stands for security,” has become commonplace and is a strong indication that security in a middleware platform is as important as it is in other aspects of the IoT ecosystem. Security becomes a vital consideration factor if you choose a multitenant platform. The multitenant aspect makes the system more vulnerable, because your own application may be just fine but another application using the same platform (a co-tenant of your application) can create security problems for every other tenant; the risk is always present.
- *Cost.* The budget set for an IoT platform has a relatively larger influence on cost factors; however, overall, if the cost of the platform (whether it was built in-house or bought off the shelf) does not justify the functionality and features, then it must be reviewed. In short, the platform should add enough value to justify its cost.

- *Support.* As much as ongoing support for platform management is essential, there is also support required for solution integration purposes. And as a mandatory requirement, the middleware platform should have strong support in the design, development, deployment, and management of the solution on an ongoing basis.

Why Should You Build Your Own IoT Platform?

As businesses and working scenarios evolve, we see many smaller companies delving into the IoT. However, not having your own IoT platform is one of the impediments or roadblocks for such an evolution.

Why not use a freemium or free trial platform? What lies ahead is a greater challenge when things scale and costs skyrocket exponentially. When the trial expires or a freemium is not enough, users find themselves locked in. This is challenging for many small players. Having your own IoT platform is a much better solution.

Buying off the shelf or a freemium might seem like better choices at the outset, however, there is a trade-off. IoT platforms that save you time may cost more in the end, depending on how vendors price them. This is mainly because the charges are either use-based or device-based. In addition, a subscription fee can add up over time. Yet, you get the benefit of significantly lower up-front costs, which means no capital expenditures; however, it also depends on how you plan to charge the end users or customers who buy your IoT product or solution.

IoT platforms that are initially inexpensive will likely cost you in time. This comes back to the same point: the less you spend, the more work you have to do on your own to integrate and nurse the platforms. If you must spend a significant amount of time, it would be better spent on building your own, don't you think?