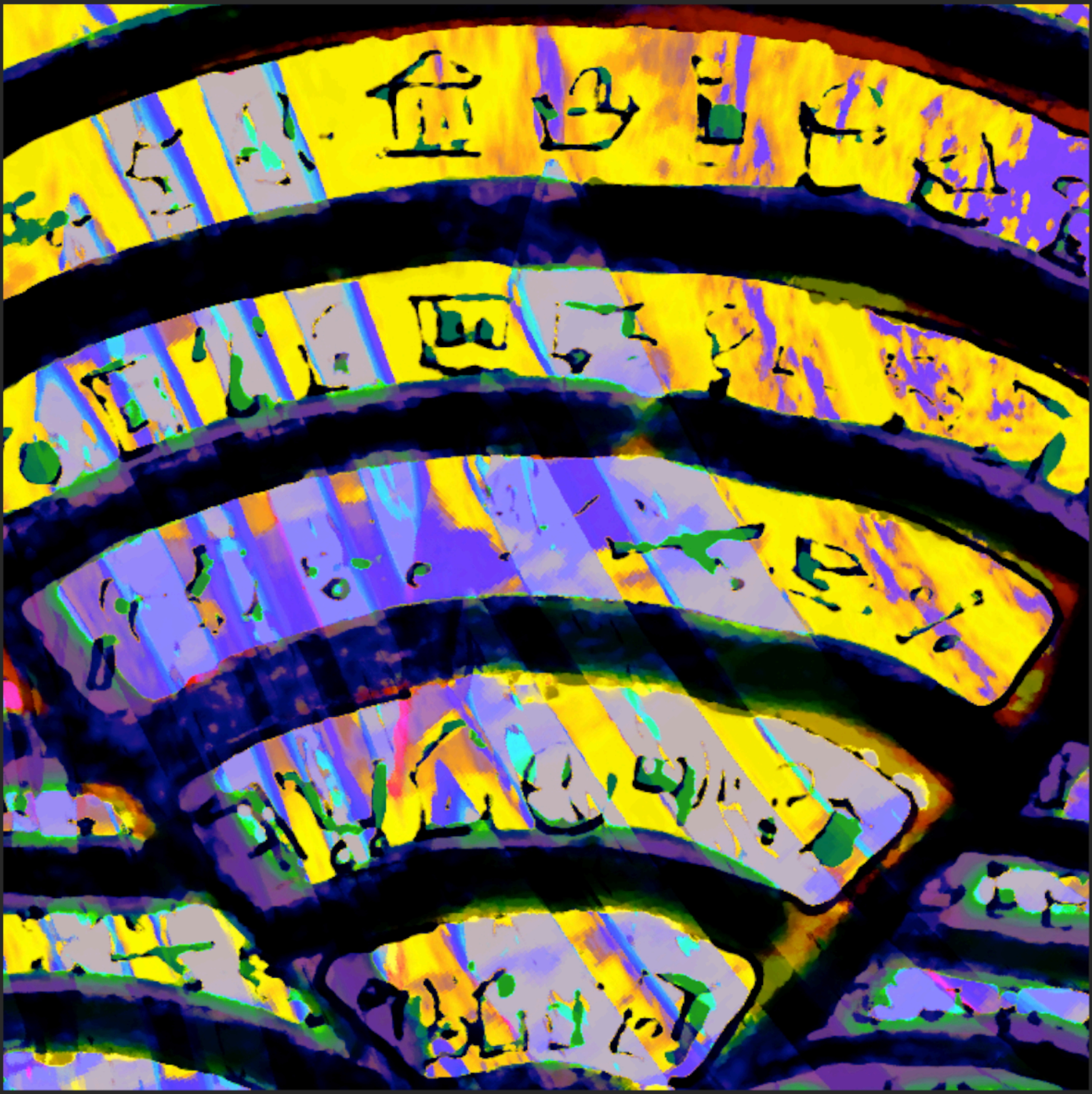


# Internet of Things using AWS Cloud

Lukasz Malinowski
Internet of Things Advisor and Trainer

# Internet of Things using AWS Cloud

part 1: Connectivity

**Lukasz Malinowski**
Internet of Things Advisor and Trainer

ThingRex.com

**Internet of Things using AWS Cloud - part 1: Connectivity**

Lukasz Malinowski

ISBN 978-83-969335-0-8

First published: September 2023
Updated: March 2024

Published by ThingRex.com Lukasz Malinowski

I dedicate this book to my wife, who inspires me to be a better person every single day. I love you!

P.S. It was her idea that I should write this publication.

# Table of Contents

## Introduction

The Internet of Things (IoT) is not a "single thing" but a blend of technologies and concepts such as networking, cybersecurity, software, and hardware design. Starting the IoT journey might be intimidating since there is no clear path to follow. Often technical documentation operates on ambiguous terms without explaining the basic principles and lacks real-life examples.I prepared this book as an introduction to the Internet of Things domain. My goal is to create solid foundations you can build on.

IoT is all about communication. For instance, devices gather data and send it to applications for processing and knowledge extraction. Then the applications send commands to devices to optimize their performance. This is why it is important to focus on various aspects of connectivity and device management. Establishing a secure connection between devices and backend applications is a crucial backbone of any IoT system. Through this book, you will learn how to manage devices while leveraging the Amazon Web Services (AWS) Cloud. This information will empower you to work with vast fleets of devices distributed around the globe.

What makes this book unique? I will share tools and techniques I used during commercial deployments of IoT solutions for multi-national enterprises. I reference real-life scenarios to help you understand the reasoning behind my recommendations and learn how to apply them. Commercial projects rarely use AWS Web Console to manage cloud resources. Manual configuration is error-prone and does not scale. I will use Python programming language to guide you through the world of IoT and manage relevant AWS infrastructure. You will learn how to automate tasks and create production-ready assets.This approach does not limit you to Python language. After practicing these concepts, you will be able to implement them using any language supported by AWS Software Development Kits (SDKs[1]).

No programming or AWS knowledge is required to start your IoT journey. I prepared a sample Python code with easy-to-understand comments and explanations. I described every AWS service mentioned in this book to assist readers without a cloud experience. You can leverage the interactive

[1] https://aws.amazon.com/what-is/sdk/

ThingRex IoT Lab environment[1] to execute code samples and experiment with your IoT infrastructure. I will show you how to simulate devices so there is no need to obtain dedicated hardware. The only prerequisite is to have access to an AWS account[2].

I covered the following topics in this book:

- ☑ The fundamental principles of the Internet of Things (IoT).
- ☑ How to represent devices in the AWS IoT Core device registry.
- ☑ How to establish trust in a distributed IoT environment.
- ☑ How to use the Private Key and X.509 Certificate as proof of identity for devices.
- ☑ The basics of the MQTT protocol, the various connectivity options it provides, and the best practices in designing the MQTT Topic structure.
- ☑ How to ensure end-to-end message delivery in distributed IoT environment.
- ☑ Important limits and quotas of AWS IoT services.

[1] https://www.thingrex.com/lab/

[2] https://aws.amazon.com/account/

## About me

My name is Lukasz Malinowski. I am the Internet of Things Advisor and Trainer. Throughout my career, I have gained extensive experience in IT projects spanning over 15 years. I have participated in various roles, including Project Manager, Team Leader, Solution Architect, Security Auditor, and Developer. That experience gave me a deep understanding of IoT technology and its applications. For the last four years, I worked at Amazon Web Services (AWS), where I helped the world's largest enterprises design, implement and secure global IoT solutions.

Currently, I am actively involved as an independent consultant and trainer, enabling companies of all sizes to achieve their business goals by leveraging modern technologies. I regularly post on my ThingRex blog[1] to share my experience with the IoT community and engage in public speaking to help people enter the fascinating Internet of Things domain.

Lukasz Malinowski, IoT Advisor and Trainer

[1] https://www.thingrex.com/

## The Internet of Things

According to Wikipedia:

*“The Internet of things (IoT) describes physical objects (or groups of such objects) with sensors, processing ability, software and other technologies that connect and exchange data with other devices and systems over the Internet or other communications networks.”* [1]

I prefer to avoid that perspective on IoT as it focuses on technology and neglects the business aspects. I saw Proof of Concept (PoC) initiatives failing for that exact reason. As a result of this perspective, project teams have been stuck in technical considerations deliberating various solutions and implementations. That dangerous trap is very tempting for IT professionals who can spend infinite time wandering in this space. From my own experience, I suggest using the desired business outcome as a finish line and avoiding solely technological contemplations.

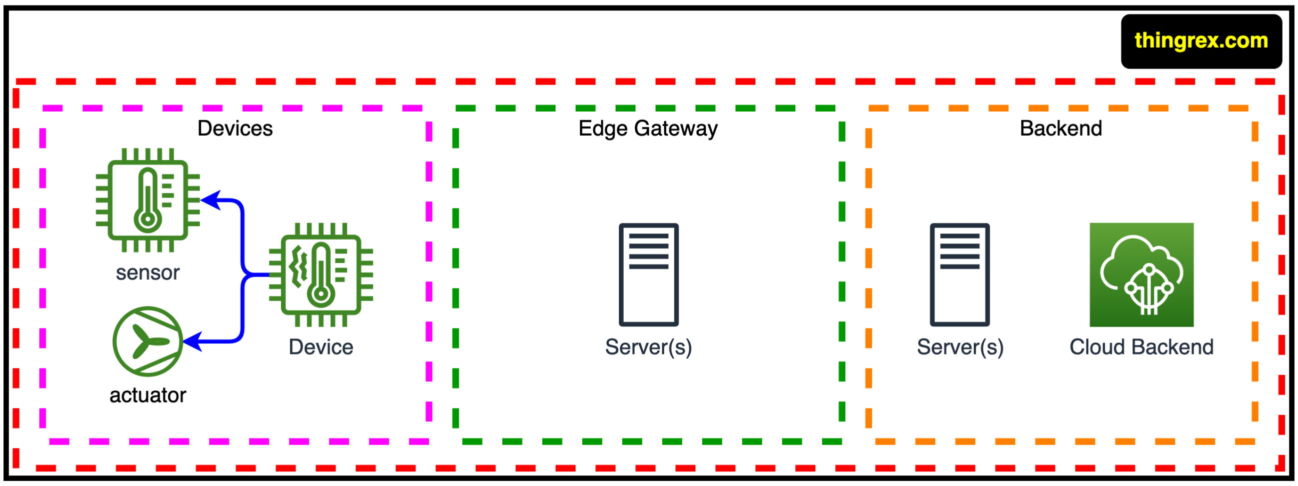
Having said that, I focus mainly on technical details in this book. I want to give you tools and explain how to use them to achieve desired business outcomes in a scalable and secure way. Technical knowledge is the foundation of IoT, but the goal is to apply it to deliver value for the end users (whomever they might be in a given case).

Before we begin, I want to clarify a common misconception. The "Internet" (in the Internet of Things phrase) refers to communication protocols devices use to exchange information, not to the "Public Internet" we use daily. Many systems use private networks and are still considered IoT solutions. While that is not super important, it might be a bit misleading, and I want to avoid any confusion.

[1] https://en.wikipedia.org/wiki/Internet_of_things

## Areas of IoT

I propose splitting the Internet of Things infrastructure into three areas. This will help organize our discussion and distinguish physical objects from logical entities.



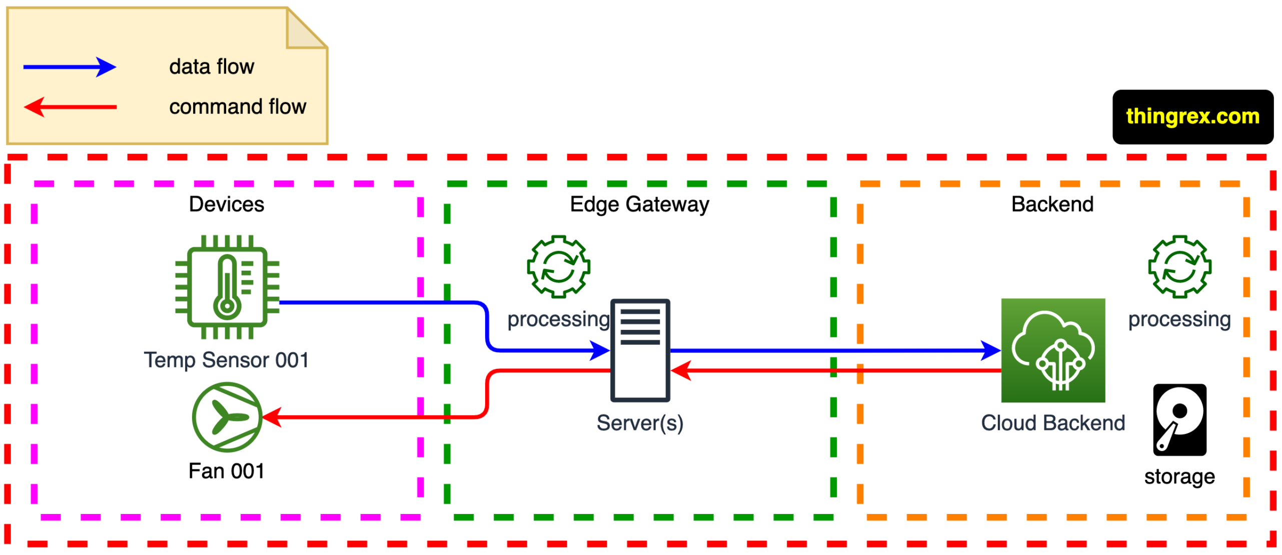
- **Devices** - Typically small, resource-constrained hardware equipped with sensors (such as thermometers) to collect data from the environment and actuators (such as fans) to interact with the surroundings.
- **Edge Gateway** - A larger device with ample resources, including Central Processing Unit (CPU), Random-Access Memory (RAM), and storage; capable of receiving data from multiple connected Devices, processing, and sending it to the Backend.
- **Backend** - The on-premise or cloud infrastructure utilized for storing and analyzing the vast amount of data the connected Devices collect.

Note: The **Edge Gateway** area is optional. In many use cases, **Devices** are connected directly to the **Backend** infrastructure.

Throughout this book, we will use simulated **Devices** and **AWS Cloud Backend** omitting the **Edge Gateway** infrastructure.

## Information flow

The typical information flow in the Internet of Things system displays as follows:



**Devices** collect data using sensors and send it to the **Edge Gateway** or **Backend**.
The **Edge Gateway** is an optional component that pre-processes data before sending it to the **Backend**. It is also capable of independently sending commands to connected **Devices** (for example, in case of network issues interrupting communication with the **Backend**).
The **Backend** analyzes data and sends commands to **Devices.** Potentially impacting their environment using actuators.

Note: Our simulated devices will communicate directly with the AWS Backend. To keep things simple, we will not implement the Edge Gateway.

From a security perspective, separating the data flow from the command flow is crucial. Hackers can leverage a compromised **Device** to gain access to the **Backend** and intercept control over other connected **Devices**. To mitigate that risk, restrict **Devices** from sending *commands* to other **Devices** or the **Backend**. Assuming that the business case does not require the above actions. To further improve the security posture of an IoT system, connected devices should not accept any incoming communication outside of the established session. That will prevent them from a wide range of remote attacks including distributed detail of service and unauthorized access.

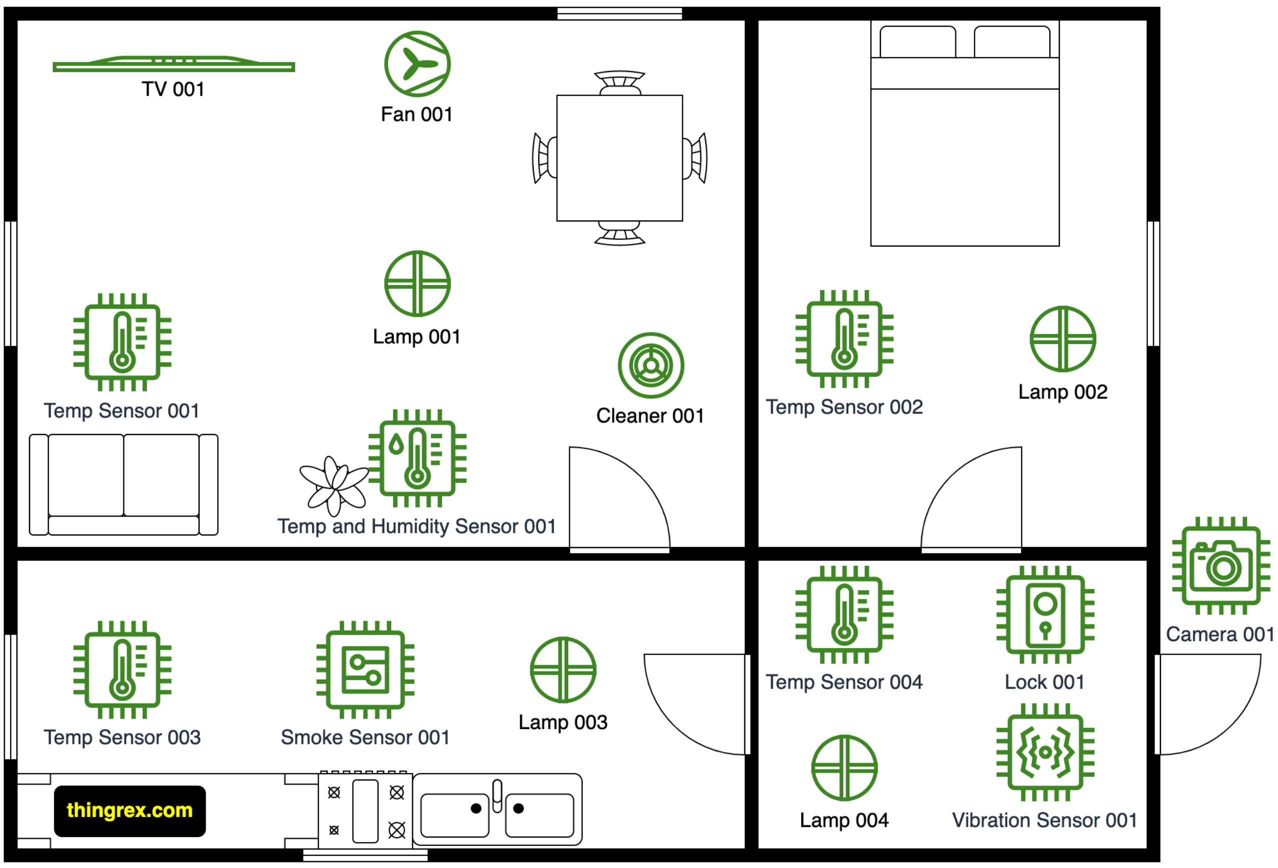
Physical and virtual equipment always act as active participants during the messages exchange process by following the sequence below:

- Verify the identity of the exposed Backend's interface.
- Establish an encrypted communication channel.
- Send and receive messages via that secure channel.

We will cover all those aspects later in this book and experiment with various implementations. I aim to keep this book as practical and hands-on as possible.

## Business Case and Environment Overview

The **Internet of Things** provides a *technical capability,* but it is not a *business goal* in and of itself. Let's start by defining the business case and the simulated environment we will use to achieve it. I suggest we use a **Smart Home** deployment as an example since it appeals to the imagination.



Our business goal is to create a framework enabling efficient management of connected devices. The designed solution will allow business cases to leverage the end-to-end communication flow:

- **Devices** send data directly to the **AWS Backend.**
- **AWS Backend** processes received telemetry data.
- If appropriate, **AWS Backend** sends commands to connected **Devices.**
- **Devices** receive commands from the **AWS Backend** within the established session and invoke local actions.

On top of the created framework, we will implement business solutions as the following points demonstrate:

- Receiving temperature readings from sensors located in various simulated rooms.
- Detecting a person standing in front of the hall door and automatically opening the lock for known users.

The technical requirement mandates efficient organization of our fleet.
The system's security is crucial. The solution should be resilient to “man-in-the-middle” and compromised device attacks. We must provide remote management functionality without exposing devices to the Internet.

Do not worry if you are not familiar with those terms. I will explain everything as we go. I listed those requirements to underline that we are going to learn how to build a production-ready solution, not a purely sandbox environment.

## IoT Lab Environment

Throughout this book, we will use the **ThingRex IoT Lab** to execute commands. That is a pre-configured learning environment designed to dive into the exciting world of the Internet of Things without the hassle of setting up local development tools. Learn more about the **ThingRex IoT Lab** and how to use it on ThingRex blog[1].

I will use the following convention to note the commands I execute and the *outputs* they produce:

```
command

output
```

In some cases, I will use *comments* (starting with the '#' symbol) to describe some aspects of invoked commands.

```
command # comment
```

Note: *Comments* do not impact the outputs of executed commands.

These are the versions of software used in this book:

Python[2] is easy to learn powerful programming language. I will not use any advanced Python functionalities; the explained concepts can be understood without knowing this language.

```
python -V

Python 3.10.13
```

[1] https://www.thingrex.com/lab/

[2] https://www.python.org/

Developers use Software Development Kits (SDKs[1]) to minimize the complexity of the source code. Instead of writing a solution from scratch, they leverage the high-level functionalities SDK exposes. AWS SDK for Python language is called Boto3[2]. It allows us to create, configure, and manage AWS services using easy-to-understand objects and functions. We will learn how to use that tool to effectively manage AWS infrastructure even without the developer's background. Please check my post to learn how to install it in the lab environment[3].

```
pip show boto3

Name: boto3
Version: 1.34.59
Summary: The AWS SDK for Python
Home-page: https://github.com/boto/boto3
```

Eclipse Mosquitto[4] is an open-source message broker that implements the MQTT protocol - one of the most popular protocols in the Internet of Things domain. We will leverage this tool to learn and experiment with MQTT. I frequently use Mosquitto in early prototyping phases during customer engagements and for demonstration purposes.

```
mosquitto -h

mosquitto version 2.0.18
```

[1] https://en.wikipedia.org/wiki/Software_development_kit

[2] https://boto3.amazonaws.com/v1/documentation/api/latest/index.html

[3] https://www.thingrex.com/lab/#your-first-notebook

[4] https://mosquitto.org/

## Cloud Infrastructure Management

We will use the AWS backend to receive data from simulated devices, analyze it, and take automated actions. You will learn how to represent tangible devices using logical assets of AWS IoT services. To efficiently manage complex AWS Cloud resources, we will use the **Infrastructure as Code** (IaC[1]) approach. It enables us to describe the IoT solution using source code instead of virtual and physical hardware configuration. Codifying AWS cloud resources and local assets provides various benefits as outlined below:

- Ease of reviewing the environment's configuration, because it is defined using the source code. There is no need to log into the AWS Web Console and manually browse cloud resources.
- Quickly reproduce the working setup by invoking our code again. Moreover, experiment with various configurations by modifying the parameters of our invocations.
- Version the source code to track changes introduced to the environment. This is crucial for debugging and auditory purposes.

I recommend to use the Software Development Kit (SDK) to manage the AWS infrastructure. This creates understanding of all properties and relations between utilized AWS services. The AWS Web Console sometimes assists users by executing some tasks "under the hood". While that helps to realize the user's intent, it is counter-productive during learning. The operator is unaware of the actions taken and does not fully understand every part of the received outcome. We want to consciously configure our cloud infrastructure, and SDK enables that.

Gained SDK skills are beneficial not only for training purposes. Internet of Things systems require advanced configuration. Infrastructure management services (like CloudFormation[2] or Terraform[3]) do not support operations specific to IoT deployments. I used SDK to create and manage business-critical IoT infrastructure for global companies. This book will explain how to manage the AWS resources as a code.

[1] https://en.wikipedia.org/wiki/Infrastructure_as_code

[2] https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html

[3] https://www.terraform.io/

The fundamental concept of AWS SDK is a **boto3 session**[1]. It establishes an encrypted connection between your local development environment and AWS Cloud. The AWS IoT infrastructure is regional. That means you must define the geographical area to host your AWS assets. In the examples, I use Ireland. I define the location of resources using the `eu-west-1` region. Feel free to choose a different region depending on your location (or, more precisely, the geographical location of the AWS resources you want to assemble). You can find the list of available regions in the AWS documentation[2].

To start the boto3 session, invoke the following code:

```
# Importing the boto3 package.
import boto3

# Variables store the AWS profile and region to use in further
invocations.
PROFILE = 'default'
REGION = 'eu-west-1'

# Starting the boto3 session.
session = boto3.Session(profile_name=PROFILE, region_name=REGION)
```

By leveraging that session, we can create specific clients for every AWS service we use and manage related cloud resources.

```
# Creating an IoT client, the main boto3 client we will use.
iot_c = session.client('iot')
```

Every AWS Account has a unique ID. Obtain it using the client for AWS Security Token Service (STS). The primary role of the AWS STS is to create credentials providing temporary access to AWS resources[3]. We will invoke

[1] https://boto3.amazonaws.com/v1/documentation/api/latest/guide/session.html#session

[2] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html#concepts-regions

[3] https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_temp.html

the `get_caller_identity`[1] call to receive details about your AWS account. Refer to AWS documentation[2] to learn about this service's full potential.

```
# STS client is required to obtain your unique AWS Account Id.
sts_c = session.client('sts')

# Obtaining the AWS Account Id and storing it into a variable for
later use.
ACCOUNT_ID = sts_c.get_caller_identity()['Account']
```

In the next chapter, we will start managing AWS infrastructure using created `iot client`.

[1] https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/sts/client/get_caller_identity.html

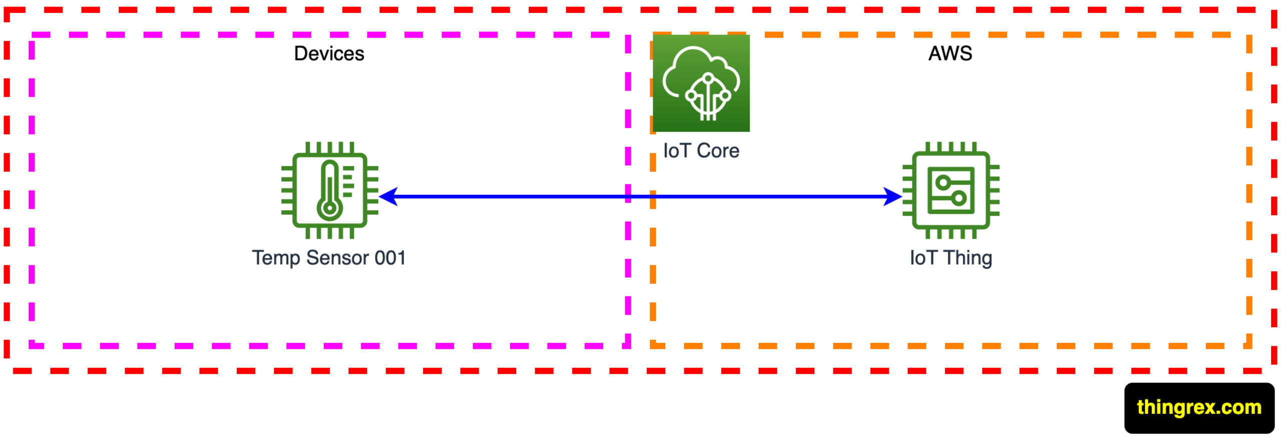
[2] https://docs.aws.amazon.com/STS/latest/APIReference/welcome.html

## Representation of Devices in the AWS Cloud

AWS IoT[1] services provide functionalities to connect and manage fleets of Devices. In this book, we will use various components of the AWS IoT suite. We will extensively use the **AWS IoT Core** as it securely connects **Devices** to **AWS IoT** and other non-IoT services offered by the AWS Cloud. **AWS IoT Core** includes a **message broker** that enables bi-directional communication with **Devices** and real-time message processing.

Let's begin with modeling our Devices in the AWS Cloud.

The **IoT Thing** is a *virtual representation* of a physical device or logical entity (example: an application) in AWS IoT.

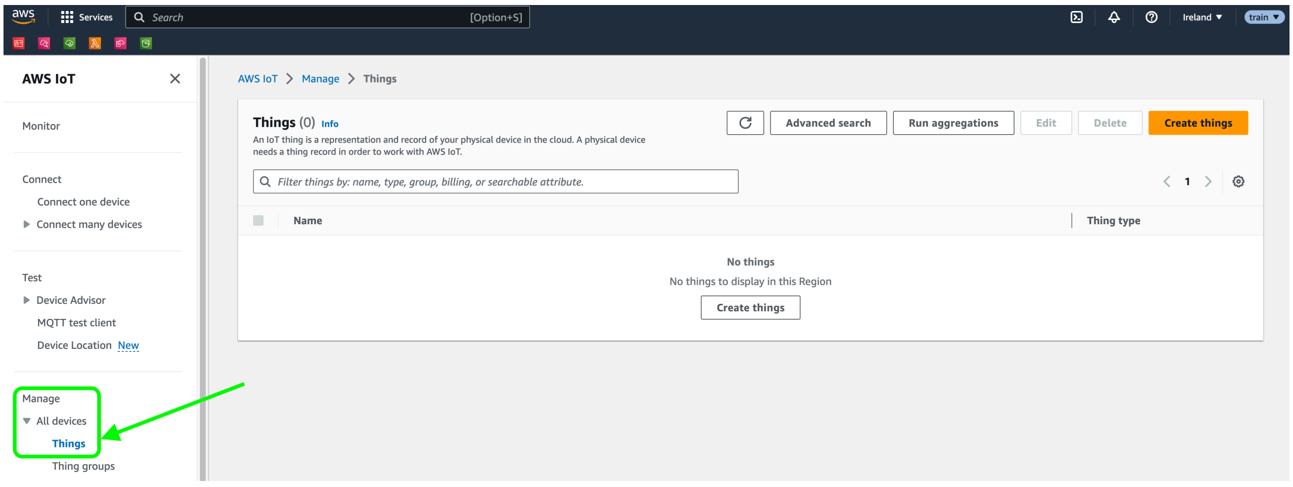


The **IoT Thing** has the following properties:

- Thing Name
- Thing Type
- Thing Attributes
- Thing Groups
- Billing Group
- Device Shadow

[1] https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html

The screenshot below displays where you can find **IoT Things** in the **AWS Console**:



An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.

<u>Note:</u> Amazon constantly updates its web console, therefore your console may look different. That is one of the reasons why I prefer to use SDK instead of a web console in most examples. AWS Console screenshots are for **reference purposes only**.

Below is a sample invocation of AWS API using boto3 SDK - notice the declaration of various attributes of the **IoT Thing**.

```
# DO NOT EXECUTE

iot_c.create_thing(
    thingName='string',
    thingTypeName='string',
    attributePayload={
        'attributes': {
            'string': 'string'
        }
    },
    billingGroupName='string'
)
```

We can invoke the following SDK call to list all **IoT Things** registered in the **AWS IoT Core**:

```
iot_c.list_things()

{'ResponseMetadata': {'RequestId': '8a8abbc7-e646-4024-ae93-
b1a5b3dbf121',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'date': 'Fri, 07 Jul 2023 08:35:10 GMT',
   'content-type': 'application/json',
   'content-length': '30',
   'connection': 'keep-alive',
   'x-amzn-requestid': '8a8abbc7-e646-4024-ae93-b1a5b3dbf121'},
  'RetryAttempts': 0},
 'things': []}
```

`'things': []` indicates that there are no **IoT Things** in the **AWS IoT Core Device Registry**.

Before we create our first **IoT Thing**, let's learn an essential aspect - the **cost allocation**.