第五讲
ThingsBoard（II）
Lecture 5
ThingsBoard (II)

明玉瑞 Yurui Ming

yrming@gmail.com

声明
Disclaimer

- 本讲义在准备过程中由于时间所限，所用材料来源并未规范标示引用来源。所引材料仅用于教学所用，作者无意侵犯原著者之知识版权，所引材料之知识版权均归原著者所有；若原著者介意之，请联系作者更正及删除。

  The time limit during the preparation of these slides incurs the situation that not all the sources of the used materials (texts or images) are properly referenced or clearly manifested. However, all materials in these slides are solely for teaching and the author is with no intention to infringe the copyright bestowed on the original authors or manufacturers. All credits go to corresponding IP holders. Please address the author for any concern for remedy including deletion.

ThingsBoard相关概念
ThingsBoard Terminology

- 实体关系定义了属于同一租户的两个ThingsBoard实体之间的连接。关系具有任意类型：包含、管理、支持等。关系也是定向的，可以将ThingsBoard关系视为面向对象编程的Has-a关系。

  Entity relation defines connection between two ThingsBoard entities that belong to the same Tenant. The relation has an arbitrary type: Contains, Manages, Supports, etc. The relation is also directional. You may treat ThingsBoard relations as a Has-a relationship from object-oriented programming.

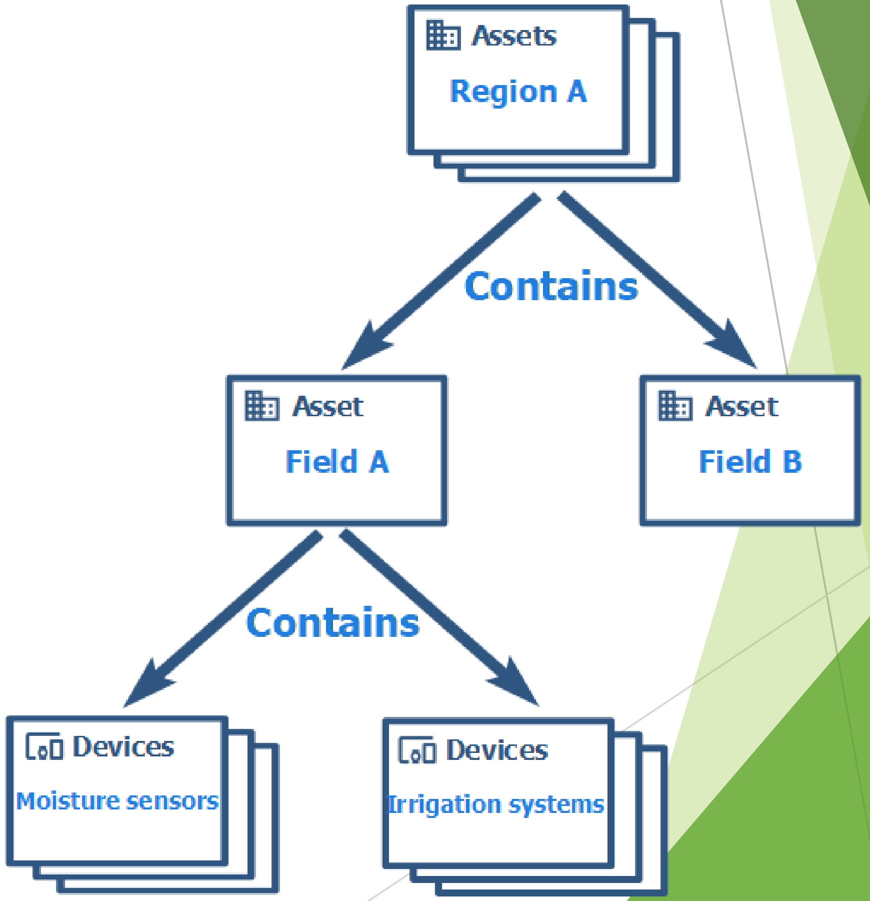
- 关系有助于在 ThingsBoard 中对物理世界对象进行建模。 理解它们的最简单方法是使用示例。

  Relations help to model physical world objects in ThingsBoard. The easiest way to understand them is using the example.

ThingsBoard相关概念
ThingsBoard Terminology

- 假设我们要构建一个应用程序，从土壤湿度和温度传感器收集数据，在仪表板上可视化这些数据，检测问题，发出警报并控制灌溉。我们还假设我们想要支持具有数百个传感器的多个地块。地块也可以分组到地理区域中。下图解释了这些实体是如何配置和存储在ThingsBoard中的：

  Let’s assume we want to build an application that collects data from soil moisture and temperature sensors, visualize this data on the dashboard, detect issues, raise alarms and control the irrigation. Let’s also assume we want to support multiple fields with hundreds of sensors. Fields may be also grouped into the Geo regions. The following diagram explains how those entities are configured and stored in ThingsBoard:



ThingsBoard相关概念
ThingsBoard Terminology

- ThingsBoard中的属性就是通常的键值对，尽管ThingsBoard预定义了一些属性，但其也支持自定义属性，并将其分配给用户的实体并允许用户管理这些属性。

  Attributes in ThingsBoard are key-value pairs. Although ThingsBoard predefines some attributes, it also provides the ability to assign custom attributes to your entities and manage these attributes.

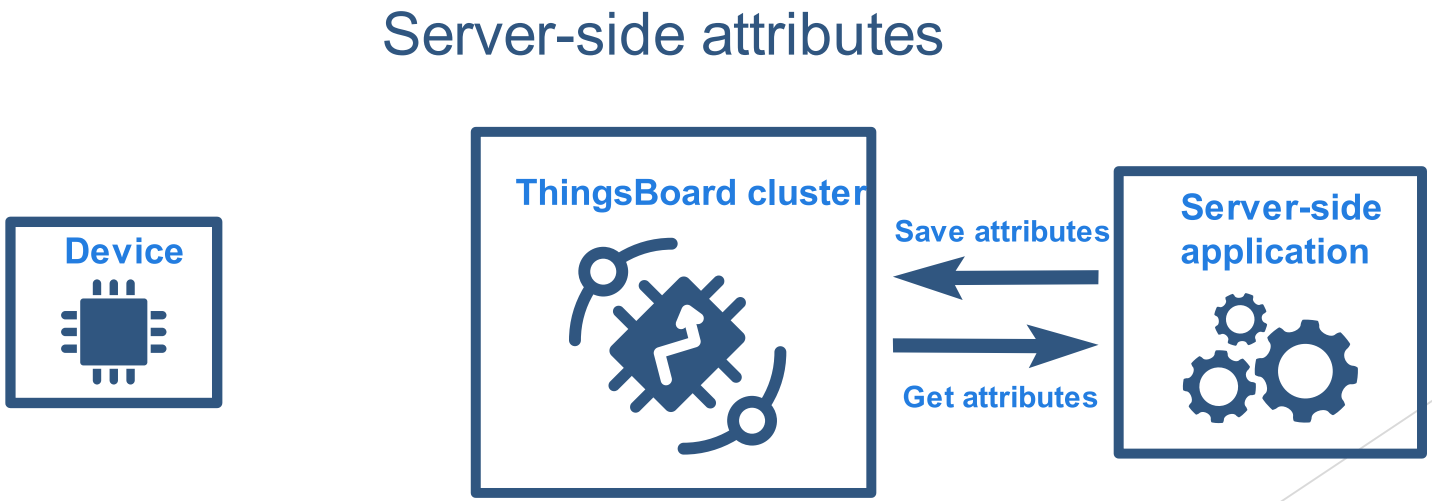
- 这些属性存储在数据库中，可用于数据可视化和数据处理。键值格式的灵活性和简单性允许与市场上几乎任何物联网设备轻松无缝集成。键始终是字符串，基本上是属性名称，而属性值可以是字符串、布尔值、双精度、整数或 JSON。尽管作为平台用户，可以定义任何属性名称。但是，系统建议使用camelCase，以使得编写用于数据处理和可视化的自定义 JS 函数变得容易。

  Those attributes are stored in the database and may be used for data visualization and data processing. Flexibility and simplicity of the key-value format allow easy and seamless integration with almost any IoT device on the market. Key is always a string and is basically an attribute name, while the attribute value can be either string, boolean, double, integer or JSON. Though as a platform user, you can define any attribute name. However, we recommend to use camelCase. This make it easy to write custom JS functions for data processing and visualization.

ThingsBoard相关概念
ThingsBoard Terminology

- ThingsBoard中的属性有三种类型，第一种是服务器属性，几乎任何平台实体都支持这种类型的属性：设备、资产、客户、租户、用户等。服务器端属性可以通过管理UI或REST API配置的属性。设备固件无法访问服务器端属性。

  There are three types of attributes. The first is server-side attributes. This type of attribute is supported by almost any platform entity: Device, Asset, Customer, Tenant, User, etc. Server-side attributes are the ones that you may configure via Administration UI or REST API. The device firmware can’t access the server-side attribute.



ThingsBoard相关概念 ThingsBoard Terminology

- 第二种属性为共享属性，此类型的属性仅适用于“设备”。它类似于服务器端属性，但有一个重要的区别。设备固件/应用程序可以请求共享属性的值或订阅属性的更新。通过MQTT或其他双向通信协议进行通信的设备可以订阅属性更新并实时接收通知。通过HTTP或其他请求—响应通信协议进行通信的设备可以定期请求共享属性的值。

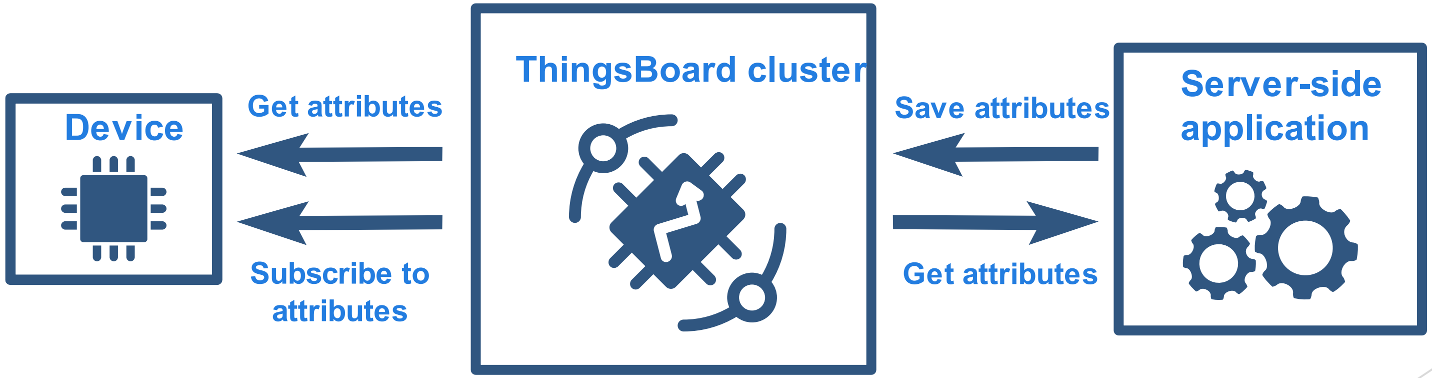
  The second type of attribute is shared attribute. This type of attributes is available only for Devices. It is similar to the Server-side attributes but has one important difference. The device firmware/application may request the value of the shared attribute(s) or subscribe to the updates of the attribute(s). The devices which communicate over MQTT or other bi-directional communication protocols may subscribe to attribute updates and receive notifications in real-time. The devices which communicate over HTTP or other request-response communication protocols may periodically request the value of shared attribute.

ThingsBoard相关概念
ThingsBoard Terminology

- 共享属性最常见的用例是存储设备设置。 用户可以通过 UI 更改属性。 脚本或其他服务器端应用程序可以通过 REST API 更改属性值。

  The most common use case of shared attributes is to store device settings. The user may change the attribute via UI. The script or other server-side application may change the attribute value via REST API.

Shared attributes



ThingsBoard相关概念
ThingsBoard Terminology

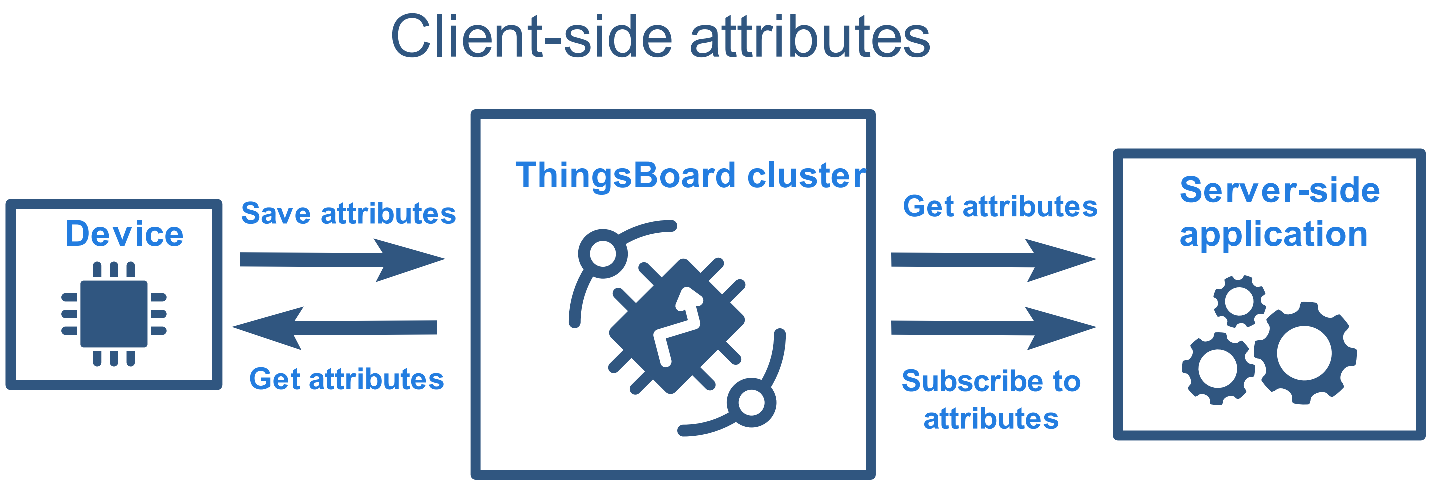
- 第三类属性为客户端属性，此类属性仅适用于设备。它用于将各种半静态数据从设备（客户端）上报到ThingsBoard（服务器）。它类似于共享属性，但有一个重要区别。设备固件/应用程序可以将属性值从设备发送到平台。

  The third type of attribute is client-side attribute. This type of attributes is available only for Devices. It is used to report various semi-static data from Device (Client) to ThingsBoard (Server). It is similar to shared attributes, but has one important difference. The device firmware/application may send the value of the attributes from device to the platform.

- 客户端属性最常见的用例是报告设备状态。用户端和服务器端应用程序可以通过UI/REST API浏览客户端属性，但它们无法更改它们。基本上，对于UI/REST API，客户端属性的值是只读的。

  The most common use case of client attributes is to report device state. The user and server-side applications may browser the client-side attributes via UI/REST API but they are not able to change them. Basically, the value of the client-side attribute is read-only for the UI/REST API.

ThingsBoard相关概念
ThingsBoard Terminology



- ThingsBoard将属性的最新值和上次修改时间存储在 SQL 数据库中，这样就可以在仪表板中使用实体筛选器进行可视化。对用户启动的属性所做的更改将记录在审核日志中。

  ThingsBoard stores latest value of the attribute and last modification time in the SQL database. This enables use of entity filters in the dashboards for data visualization. Changes to the attributes initiated by the user are recorded in the audit logs.

ThingsBoard相关概念
ThingsBoard Terminology

- ThingsBoard中另外一个重要概念是规则引擎，规则引擎负责处理各种传入数据和事件。下面是在规则引擎中使用属性的最常用方案：
  - 对服务器端属性的更改做出反应，在传入的客户端属性存储在数据库中之前对其进行修改。
  - 根据针对属性值的逻辑表达式生成警报，例如，使用报警规则通过 UI 配置最常见的报警条件，或使用过滤器节点通过自定义 JS 函数配置更具体的用例。
  - 使用消息类型切换规则节点筛选包含“发布属性”请求或“属性已更新”通知等消息。然后，使用转换规则节点修改特定消息，或对传入事件做出反应。
  - 提取属性值以分析来自设备的传入遥测数据，例如，使用扩充规则节点，使用设备、相关资产、客户或租户的属性来丰富传入的遥测消息。这是一种非常强大的技术，允许根据存储在属性中的设置修改处理逻辑和参数。

ThingsBoard相关概念
ThingsBoard Terminology

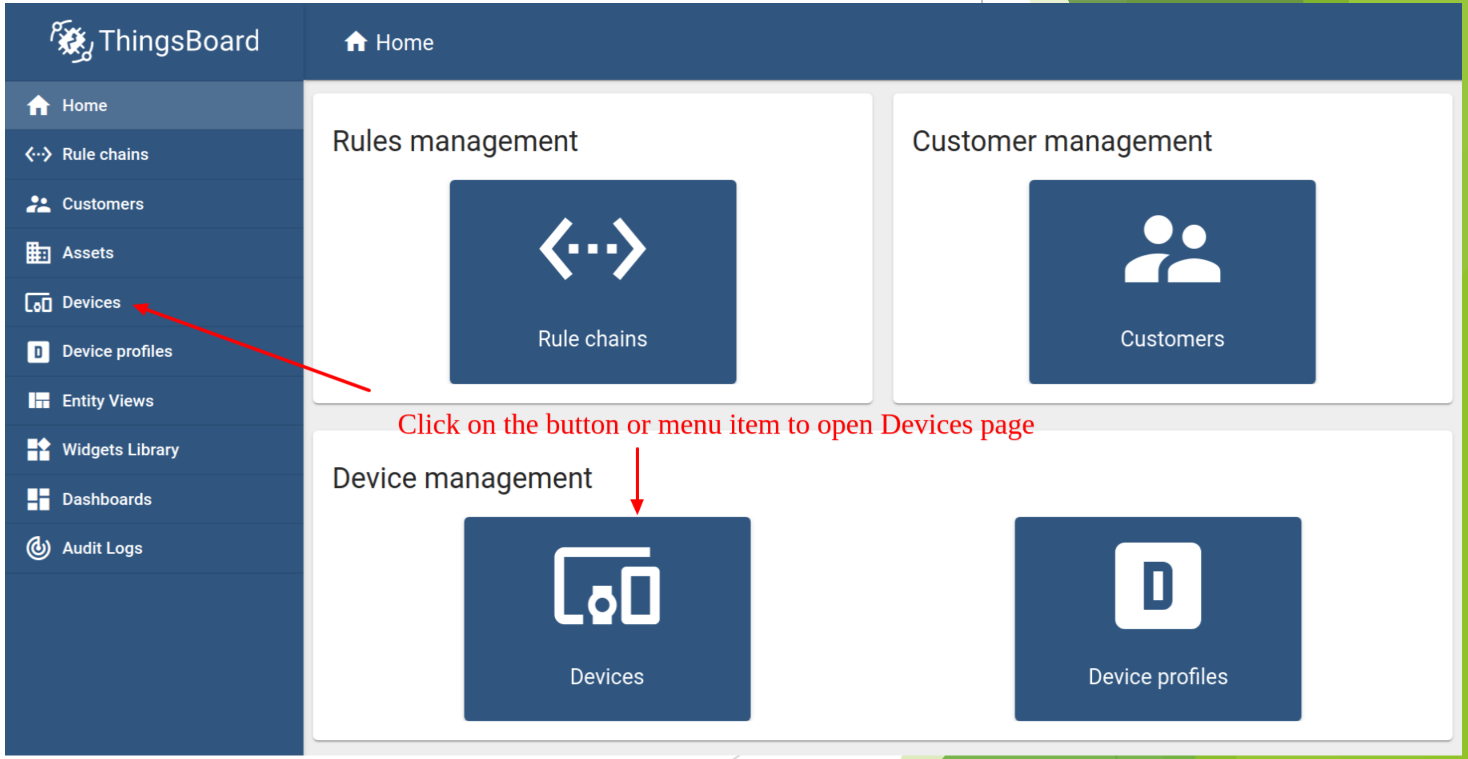
- Another import concept in ThingsBoard is Rule engine. The Rule Engine is responsible for processing all sorts of incoming data and event. You may find most popular scenarios of using attributes within rule engine below:
  - React on the change of server-side attribute; Modify incoming client-side attributes before they are stored in the database
  - Generate alarms based on the logical expressions against attribute values, for example, use alarm rules to configure most common alarm conditions via UI or use filter nodes to configure more specific use cases via custom JS functions.
  - Use message type switch rule node to filter messages that contain “Post attributes” request, or “Attributes Updated” notification. Then, use transformation rule nodes to modify a particular message, or use action or external to react on the incoming event.
  - Fetch attribute values to analyze incoming telemetry from device. For example, use enrichment rule nodes to enrich incoming telemetry message with attributes of the device, related asset, customer or tenant. This is extremely powerful technique that allows to modify processing logic and parameters based on settings stored in the attributes.

ThingsBoard基本使用
ThingsBoard Basic Usage

- 下面我们通过一个简单的虚拟温度感测器的示例，讲一下ThingsBoard的基本使用:
  - 将设备连接到 ThingsBoard;
  - 将数据从设备推送到 ThingsBoard;
  - 构建实时最终用户仪表板;
  - 定义阈值并触发警报;
  - 通过电子邮件、短信或其他系统推送有关新警报的通知。
- In the following by utilizing a virtual temperature sensor, we introduce the basic usage of ThingsBoard:
  - Connect devices to ThingsBoard;
  - Push data from devices to ThingsBoard;
  - Build real-time end-user dashboards;
  - Define thresholds and trigger alarms;
  - Push notifications about new alarms over email, sms or other systems.

ThingsBoard基本使用
ThingsBoard Basic Usage

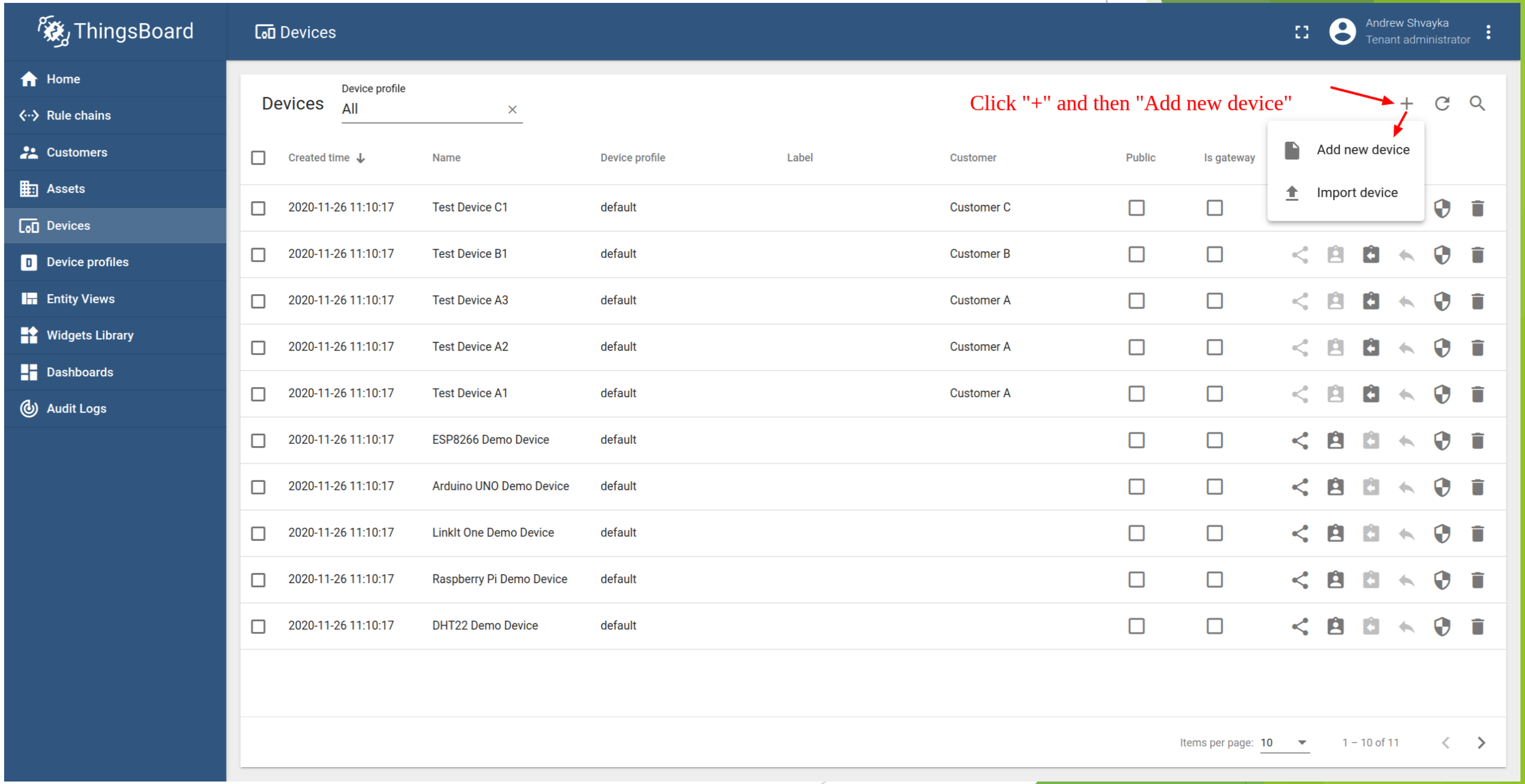
- 第一步，添加设备：
  - 启动ThingsBoard服务后，通过http://localhost:8080访问用户登录界面，可用创建或系统默认的凭据，即用户为名tenant@thingsboard.org，密码为tenant的凭据登录系统；登录进ThingsBoard实例后，打开设备页面；
- The first step, device provision:
  - Start the ThingsBoard service, access the user login UI via http://localhost:8080; You may use your created credential or use the default one, such as username: tenant@thingsboard.org and password: tenant, to log into the system; Login to your ThingsBoard instance and open the Devices page;



Click on the button or menu item to open Devices page

ThingsBoard基本使用
ThingsBoard Basic Usage

- 第一步，添加设备：
  - 单击表格右上角的“+”图标，然后选择“添加新设备”。
- The first step, device provision:
  - Click on the "+" icon in the top right corner of the table and then select "Add new device".



Devices

Device profile: All

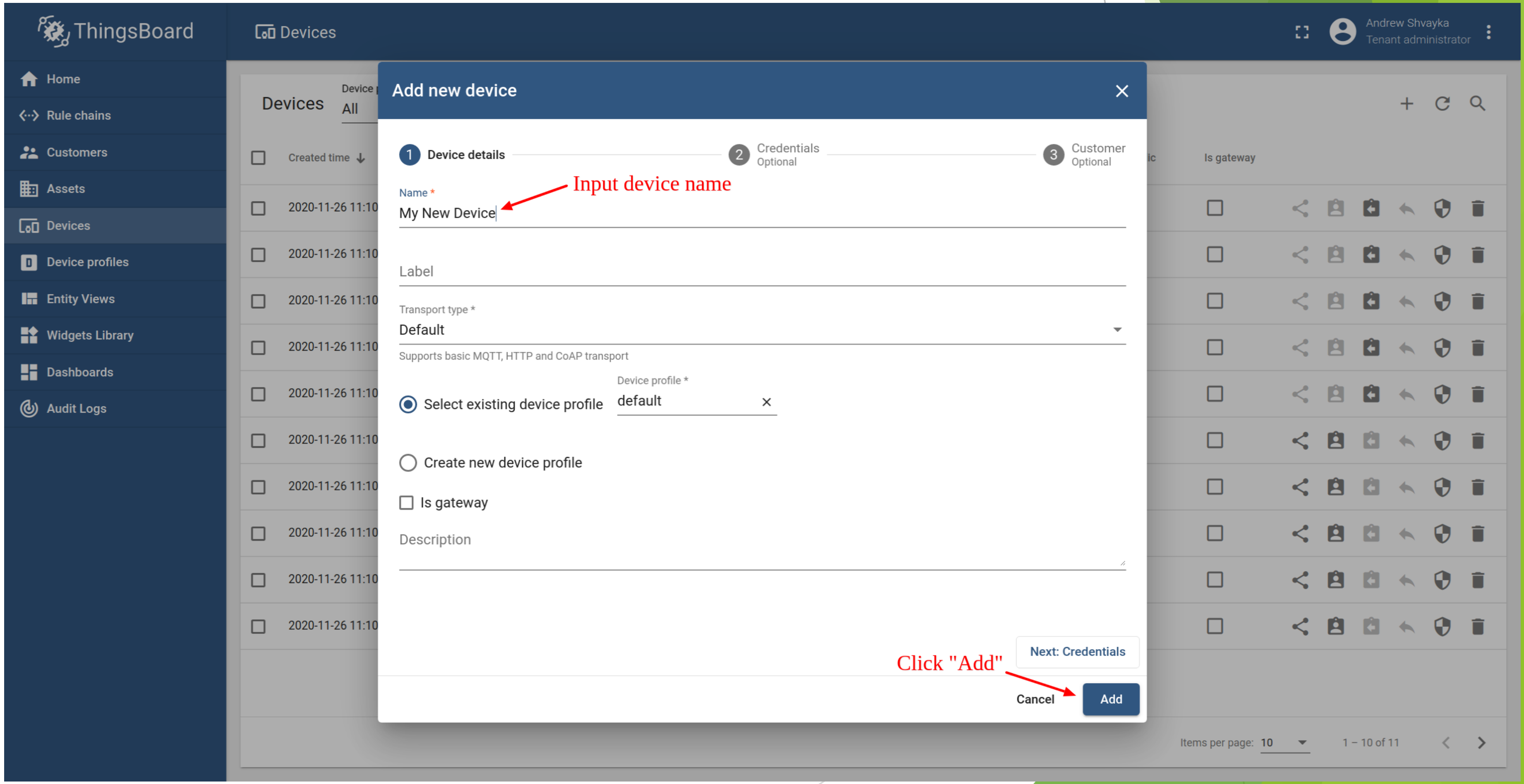
Click "+" and then "Add new device"

□	Created time ↓	Name	Device profile	Label	Customer	Public	Is gateway
□	2020-11-26 11:10:17	Test Device C1	default		Customer C	□	□
□	2020-11-26 11:10:17	Test Device B1	default		Customer B	□	□
□	2020-11-26 11:10:17	Test Device A3	default		Customer A	□	□
□	2020-11-26 11:10:17	Test Device A2	default		Customer A	□	□
□	2020-11-26 11:10:17	Test Device A1	default		Customer A	□	□
□	2020-11-26 11:10:17	ESP8266 Demo Device	default			□	□
□	2020-11-26 11:10:17	Arduino UNO Demo Device	default			□	□
□	2020-11-26 11:10:17	LinkIt One Demo Device	default			□	□
□	2020-11-26 11:10:17	Raspberry Pi Demo Device	default			□	□
□	2020-11-26 11:10:17	DHT22 Demo Device	default			□	□

Items per page: 10 1 – 10 of 11

ThingsBoard基本使用
ThingsBoard Basic Usage

- 第一步，添加设备：
  - 输入设备名称。例如，“虚拟温度感测器”。目前不需要其他更改。单击“添加”以添加设备。
- The first step, device provision:
  - Input device name. For example, “Virtual Temperature Sensor". No other changes required at this time. Click "Add" to add the device.



Add new device

1 Device details — 2 Credentials Optional — 3 Customer Optional

Name *: My New Device

Input device name

Label

Transport type *: Default

Supports basic MQTT, HTTP and CoAP transport

● Select existing device profile — Device profile *: default

○ Create new device profile

□ Is gateway

Description

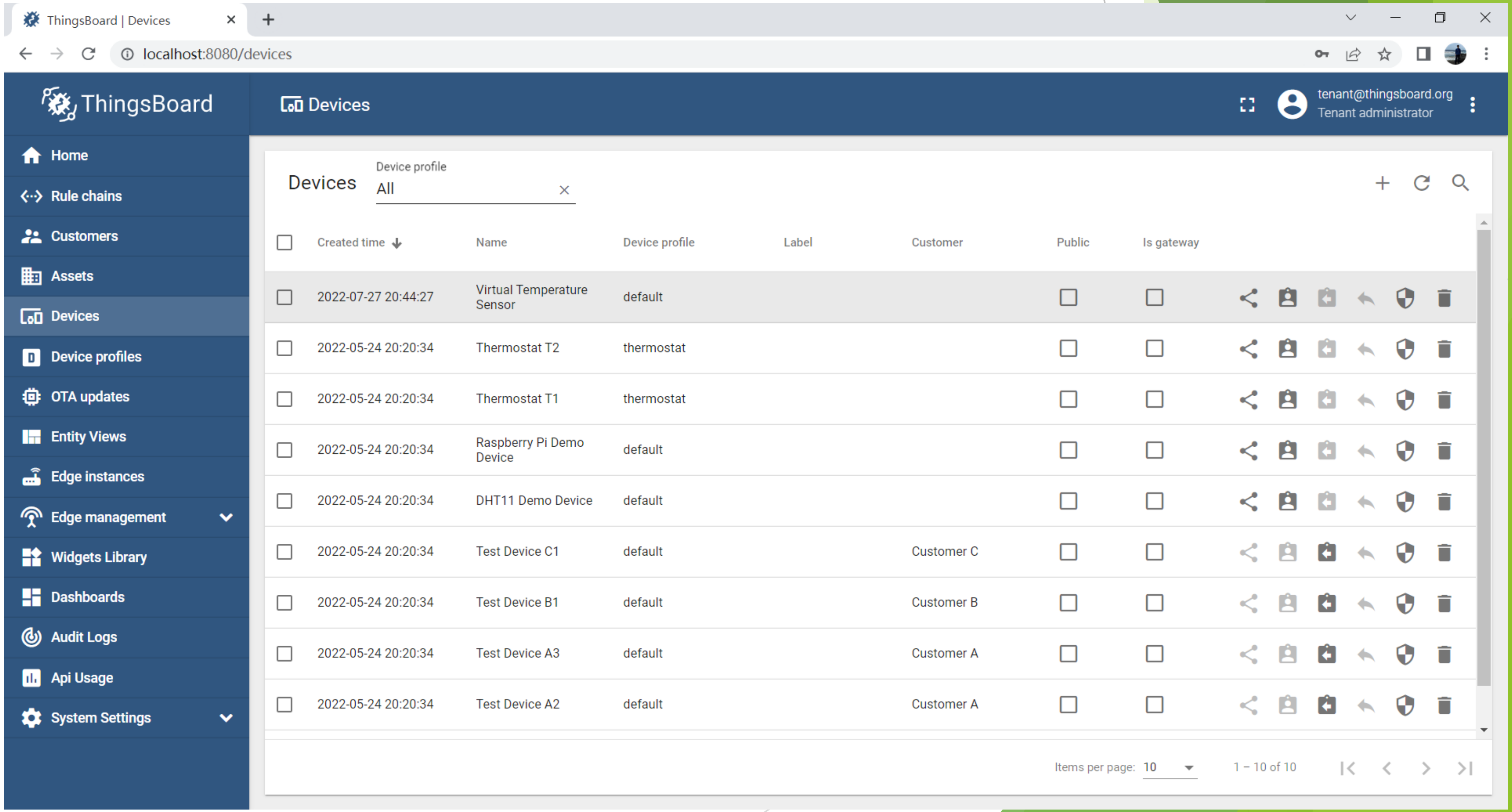
Next: Credentials

Click "Add"

Cancel | Add

ThingsBoard基本使用
ThingsBoard Basic Usage

- 第一步，添加设备：
  - 现在，我们创建的设备应该首先列出，因为默认情况下，表使用创建时间对设备进行排序。
- The first step, device provision:
  - Now your device should be listed first, since the table sort devices using the time of the creation by default.



Devices

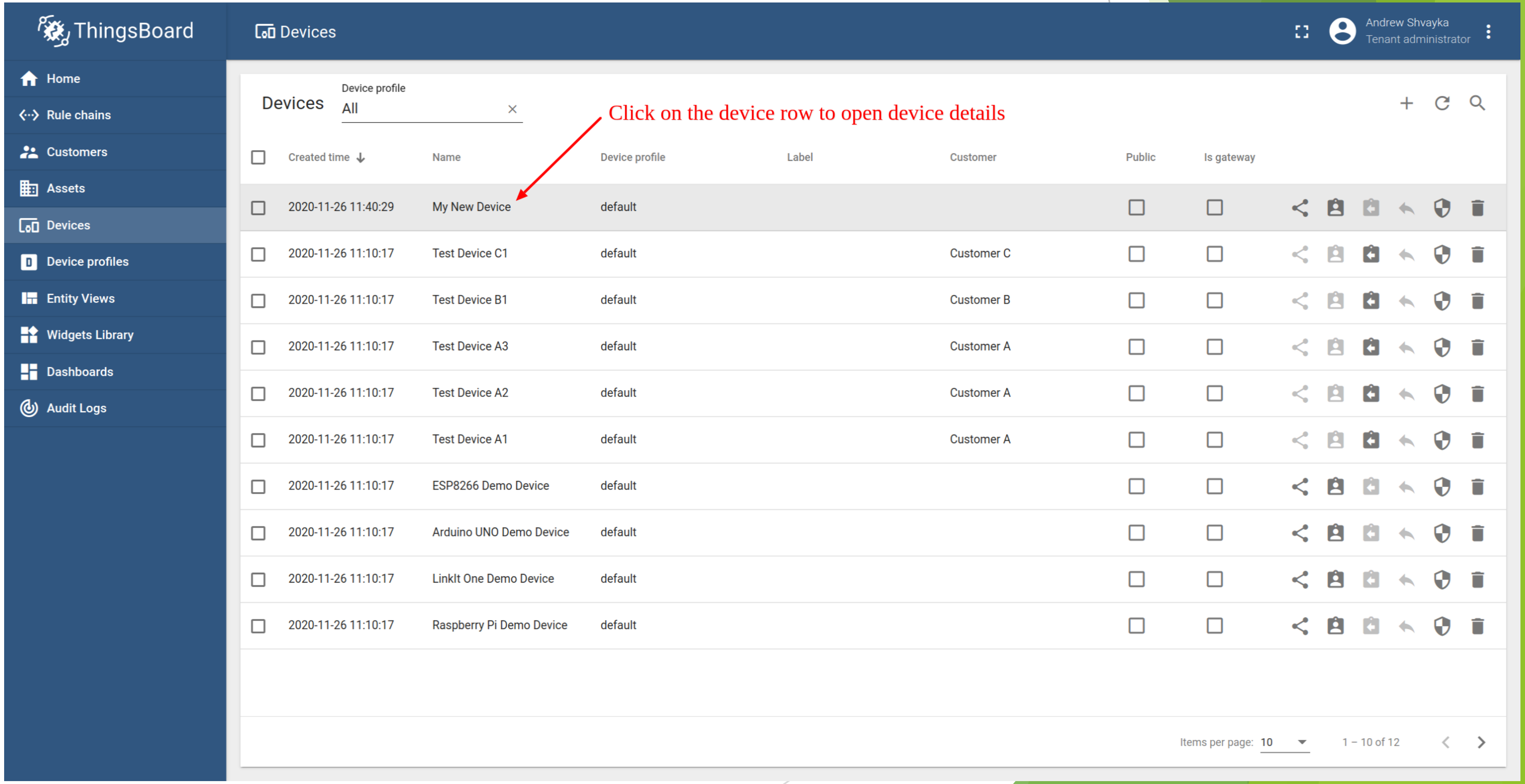
Device profile: All

☐	Created time ↓	Name	Device profile	Label	Customer	Public	Is gateway
☐	2022-07-27 20:44:27	Virtual Temperature Sensor	default			☐	☐
☐	2022-05-24 20:20:34	Thermostat T2	thermostat			☐	☐
☐	2022-05-24 20:20:34	Thermostat T1	thermostat			☐	☐
☐	2022-05-24 20:20:34	Raspberry Pi Demo Device	default			☐	☐
☐	2022-05-24 20:20:34	DHT11 Demo Device	default			☐	☐
☐	2022-05-24 20:20:34	Test Device C1	default		Customer C	☐	☐
☐	2022-05-24 20:20:34	Test Device B1	default		Customer B	☐	☐
☐	2022-05-24 20:20:34	Test Device A3	default		Customer A	☐	☐
☐	2022-05-24 20:20:34	Test Device A2	default		Customer A	☐	☐

Items per page: 10　1 – 10 of 10

ThingsBoard基本使用
ThingsBoard Basic Usage

- 第二步，连接设备：
  - 若要连接设备，需要先获取设备凭据。ThingsBoard支持各种设备凭据，我们这里使用默认的自动生成的凭据。
- The second step, device connection:
  - To connect the device you need to get the device credentials first. ThingsBoard supports various device credentials. We use the default auto-generated credentials which is access token here.



Devices

Device profile: All

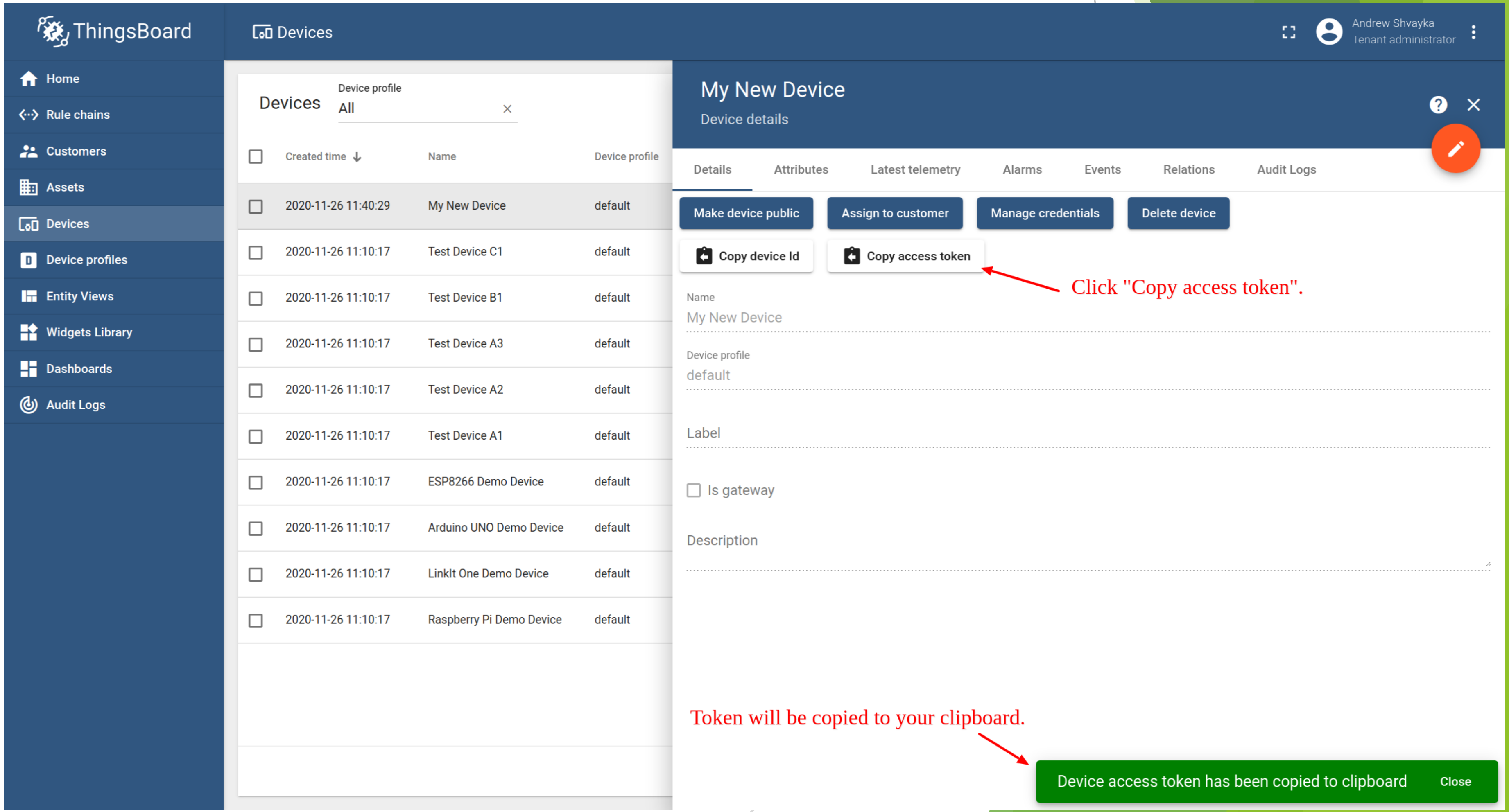
Click on the device row to open device details

☐	Created time ↓	Name	Device profile	Label	Customer	Public	Is gateway
☐	2020-11-26 11:40:29	My New Device	default			☐	☐
☐	2020-11-26 11:10:17	Test Device C1	default		Customer C	☐	☐
☐	2020-11-26 11:10:17	Test Device B1	default		Customer B	☐	☐
☐	2020-11-26 11:10:17	Test Device A3	default		Customer A	☐	☐
☐	2020-11-26 11:10:17	Test Device A2	default		Customer A	☐	☐
☐	2020-11-26 11:10:17	Test Device A1	default		Customer A	☐	☐
☐	2020-11-26 11:10:17	ESP8266 Demo Device	default			☐	☐
☐	2020-11-26 11:10:17	Arduino UNO Demo Device	default			☐	☐
☐	2020-11-26 11:10:17	LinkIt One Demo Device	default			☐	☐
☐	2020-11-26 11:10:17	Raspberry Pi Demo Device	default			☐	☐

Items per page: 10 1 – 10 of 12

ThingsBoard基本使用
ThingsBoard Basic Usage

- 第二步，连接设备：
  - 单击表中的设备行以打开设备详细信息，然后单击“复制访问令牌”。令牌将被复制到剪贴板，可将其保存到安全的地方以备后用。
- The second step, device connection:
  - Click on the device row in the table to open device details, then click "Copy access token". Token will be copied to your clipboard. Save it to a safe place for later use.



□	Created time ↓	Name	Device profile
□	2020-11-26 11:40:29	My New Device	default
□	2020-11-26 11:10:17	Test Device C1	default
□	2020-11-26 11:10:17	Test Device B1	default
□	2020-11-26 11:10:17	Test Device A3	default
□	2020-11-26 11:10:17	Test Device A2	default
□	2020-11-26 11:10:17	Test Device A1	default
□	2020-11-26 11:10:17	ESP8266 Demo Device	default
□	2020-11-26 11:10:17	Arduino UNO Demo Device	default
□	2020-11-26 11:10:17	LinkIt One Demo Device	default
□	2020-11-26 11:10:17	Raspberry Pi Demo Device	default

Name
My New Device

Device profile
default

Label

□ Is gateway

Description

Click "Copy access token".

Token will be copied to your clipboard.

Device access token has been copied to clipboard

ThingsBoard基本使用
ThingsBoard Basic Usage

- 第二步，连接设备：
  - 至此，代表设备发布遥测数据的准备工作已经完成。在本例中，我们将使用MQTT 发布数据。

The second step, device connection:

  - Now you are ready to publish telemetry data on behalf of your device. We will publish data over MQTT in this example.

- 我们以Paho库为例，编写代码实现相应虚拟设备。

Exemplified by using Paho library, we write the script to implement the virtual device.

```
import paho.mqtt.client as mqtt

from random import randrange, uniform

import time, signal
```

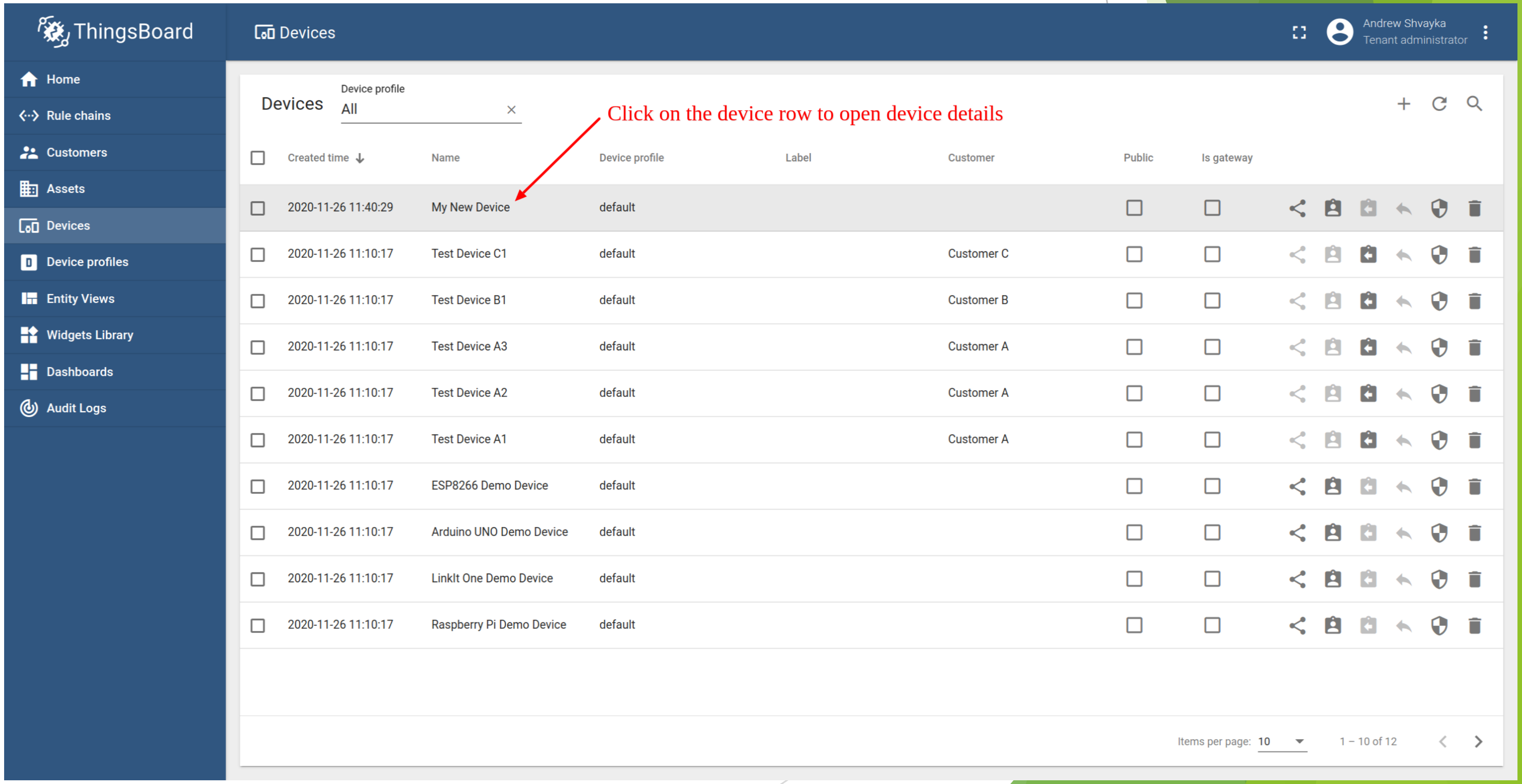
```
try:
    client = mqtt.Client("Virtual Temperature Sensor")
    client.username_pw_set("ryQnV5idaHhAPWT7lBOw", password=None)
    client.connect("localhost")
except Exception as e:
    print(e)
    exit(0)

def handler(signum, frame):
    global publishing, client
    publishing = False
    client.disconnect()
signal.signal(signal.SIGINT, handler)

publishing = True
while publishing:
    randNumber = int(uniform(30.0, 40.0))
    client.publish("v1/devices/me/telemetry", f"{{temperature:{randNumber}}}", qos=1)
    print("Just published " + str(randNumber) + " to topic telemetry")
    time.sleep(5)
```

ThingsBoard基本使用
ThingsBoard Basic Usage

- 第二步，连接设备：
  - 成功发布“温度”读数后，应立即在“设备遥测”选项卡中看到它们，单击表中的设备行以打开设备详细信息：
- The second step, device connection:
  - Once you have successfully published the “temperature” readings, you should immediately see them in the Device Telemetry Tab. Click on the device row in the table to open device details.



Devices

Device profile: All

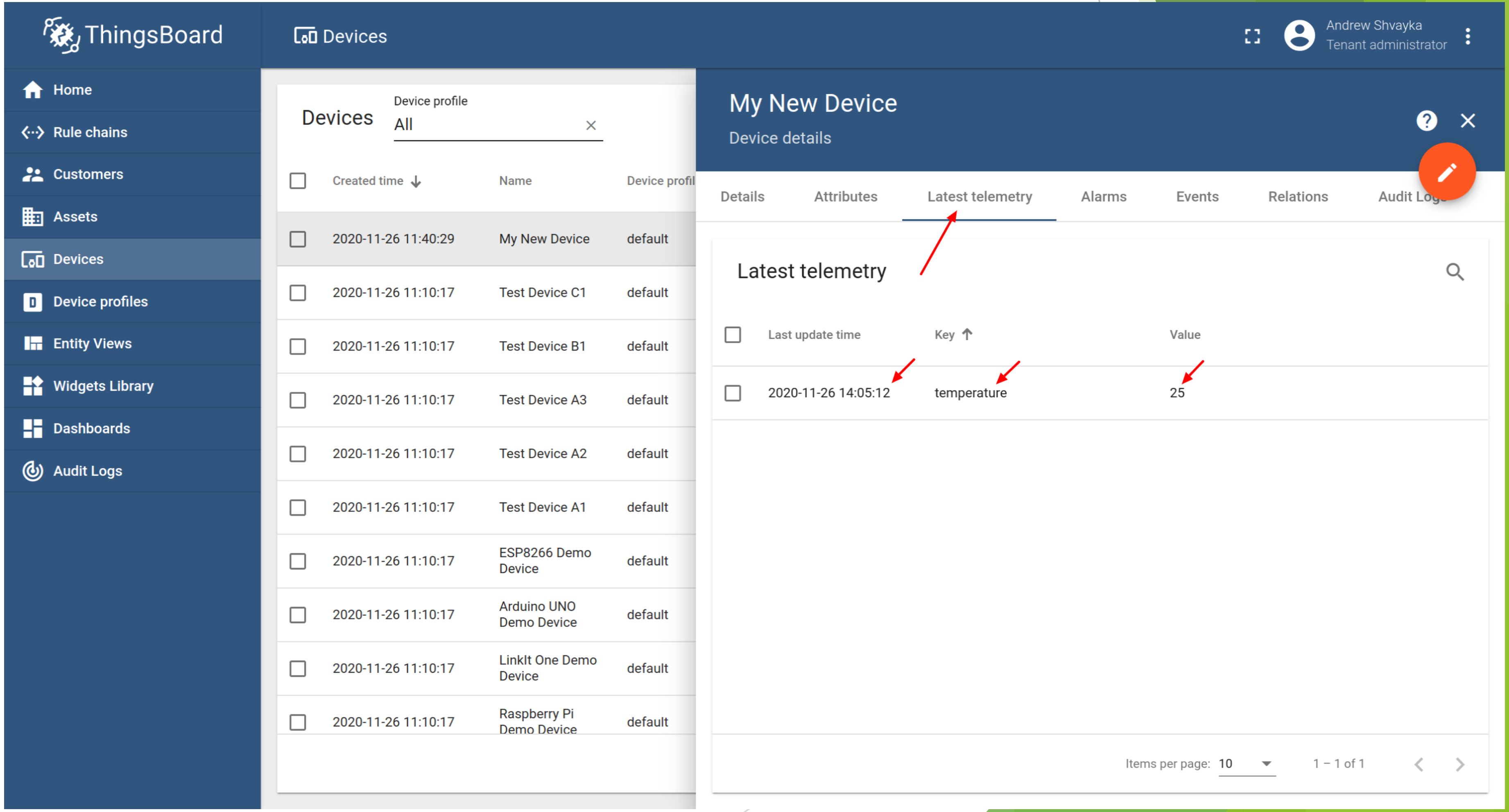
Click on the device row to open device details

□	Created time ↓	Name	Device profile	Label	Customer	Public	Is gateway
□	2020-11-26 11:40:29	My New Device	default			□	□
□	2020-11-26 11:10:17	Test Device C1	default		Customer C	□	□
□	2020-11-26 11:10:17	Test Device B1	default		Customer B	□	□
□	2020-11-26 11:10:17	Test Device A3	default		Customer A	□	□
□	2020-11-26 11:10:17	Test Device A2	default		Customer A	□	□
□	2020-11-26 11:10:17	Test Device A1	default		Customer A	□	□
□	2020-11-26 11:10:17	ESP8266 Demo Device	default			□	□
□	2020-11-26 11:10:17	Arduino UNO Demo Device	default			□	□
□	2020-11-26 11:10:17	LinkIt One Demo Device	default			□	□
□	2020-11-26 11:10:17	Raspberry Pi Demo Device	default			□	□

Items per page: 10　1 – 10 of 12

ThingsBoard基本使用
ThingsBoard Basic Usage

- 第二步，连接设备：
  - 导航到“遥测”选项卡：
- The second step, device connection:
  - Navigate to the telemetry tab:



☐	Created time ↓	Name	Device profil
☐	2020-11-26 11:40:29	My New Device	default
☐	2020-11-26 11:10:17	Test Device C1	default
☐	2020-11-26 11:10:17	Test Device B1	default
☐	2020-11-26 11:10:17	Test Device A3	default
☐	2020-11-26 11:10:17	Test Device A2	default
☐	2020-11-26 11:10:17	Test Device A1	default
☐	2020-11-26 11:10:17	ESP8266 Demo Device	default
☐	2020-11-26 11:10:17	Arduino UNO Demo Device	default
☐	2020-11-26 11:10:17	LinkIt One Demo Device	default
☐	2020-11-26 11:10:17	Raspberry Pi Demo Device	default

☐	Last update time	Key ↑	Value
☐	2020-11-26 14:05:12	temperature	25

Items per page: 10 1 – 1 of 1