

PyLab 2 : Raspberry Pi, Azure 物联网中心和 Docker 容器调试

在 Twitter [@dgllover](#) 上关注我



作者	微软云开发者倡导者 Dave Glover
平台	Linux, macOS, Windows, Raspbian Buster
服务	Azure 物联网中心
工具	Visual Studio 代码
硬件	Raspberry Pi , Raspberry Pi Sense HAT
语言	Python
日期	2019 年 9 月

PDF 实验指南

您可能会发现下载和遵循“[Raspberry Pi, Python, Azure IoT Central 和 Docker Container Debugging](#) 动手实验指南”的 PDF 版本更容易。

PyLab 内容

- [PyLab 1 : Raspberry Pi, 调试 Python 物联网应用程序](#)
- [PyLab 2 : Raspberry Pi, Azure 物联网中心和 Docker 容器调试](#)

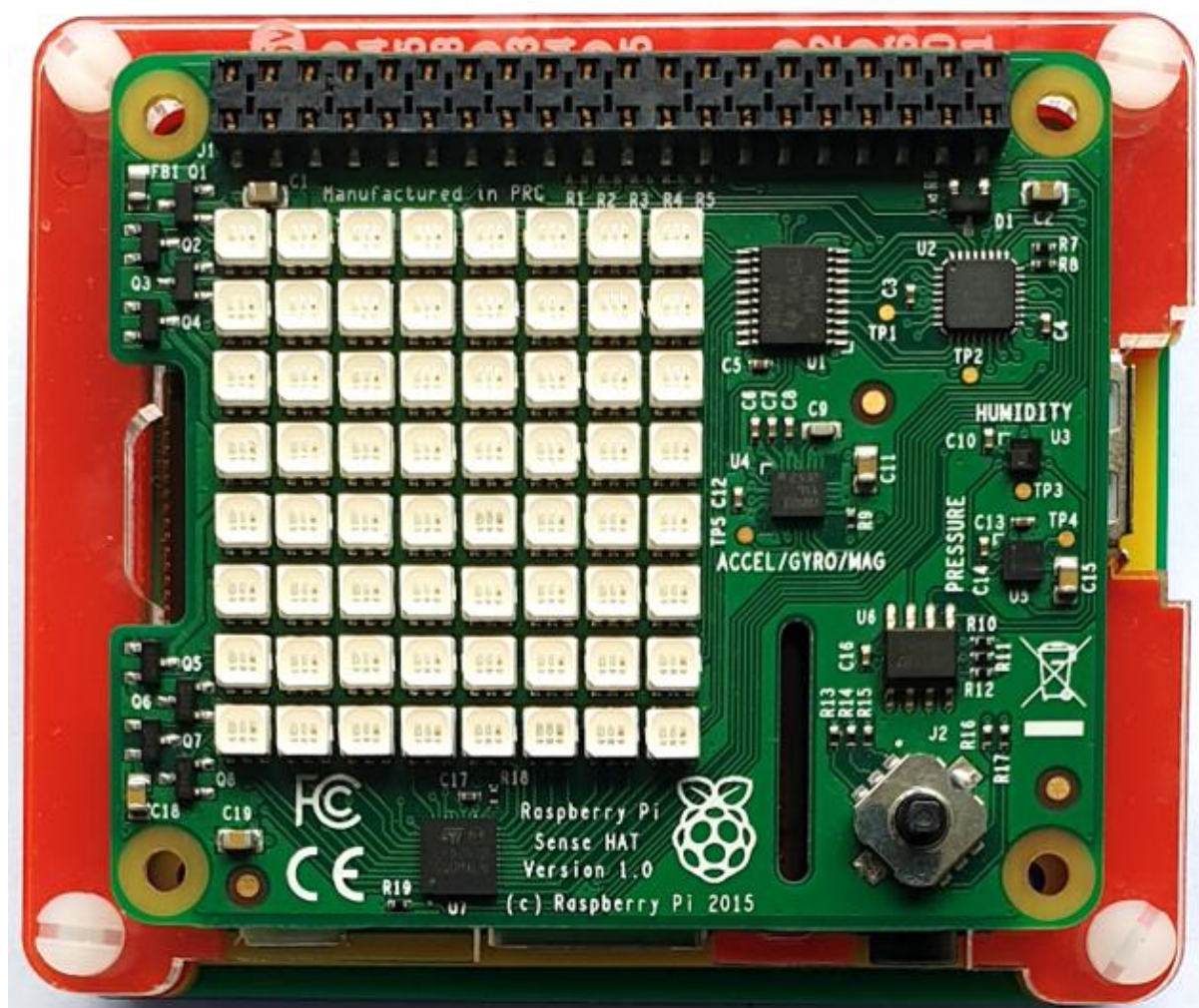
介绍

在本动手实验中，您将学习如何使用 [Visual Studio Code](#) 创建 Python 物联网（IoT）应用程序。在 Raspberry Pi 上的 Docker Container 中运行应用程序，从传感器读取温度，湿度和气压遥测，最后调试 Docker 容器中运行的应用程序。

PyLab 设置

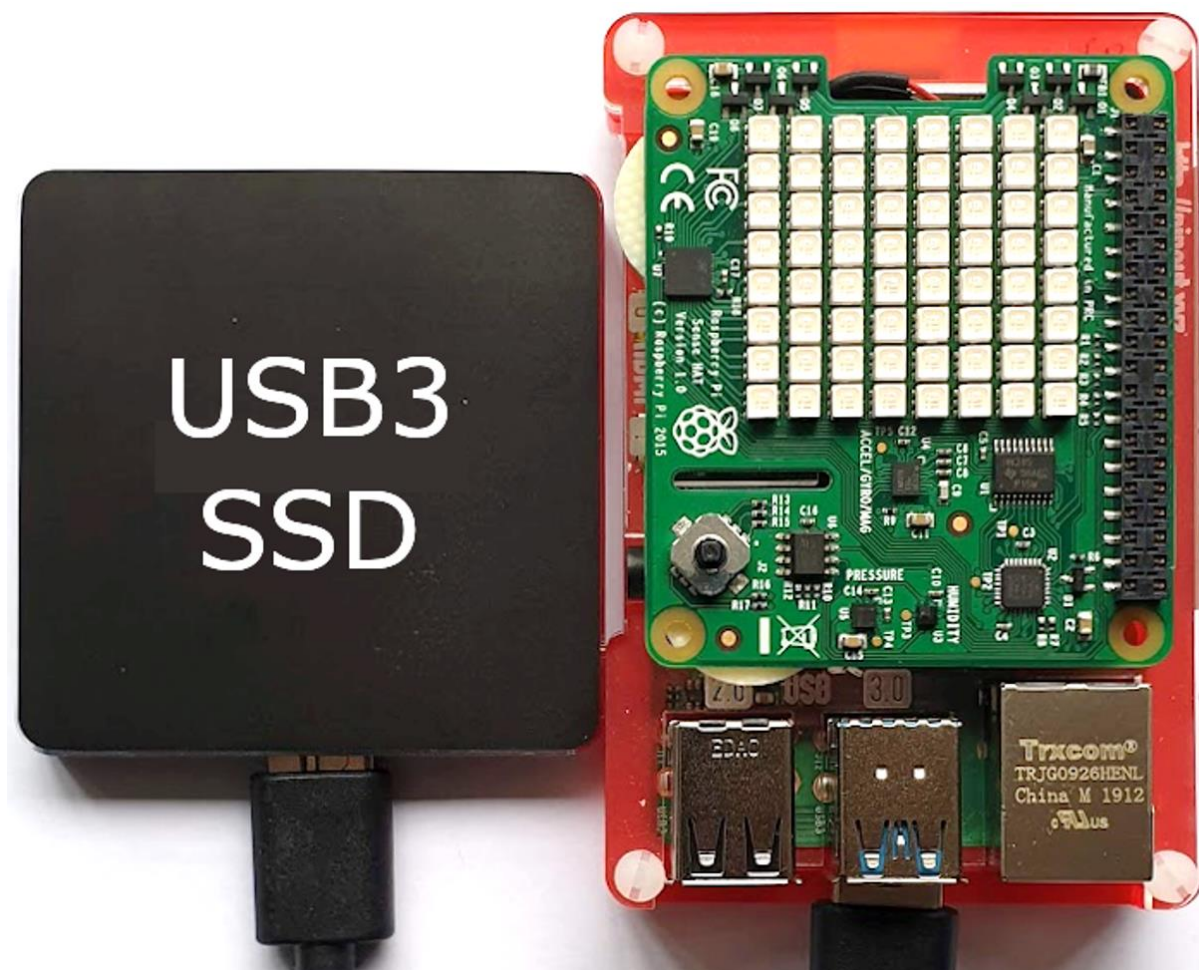
- [单用户设置](#)

此自动设置安装所需的库， Docker， 并构建实验室 docker 镜像。



- [多用户设置](#)

多用户设置允许每个 Raspberry Pi 4 4GB 最多 20 个用户/学生。需要 USB3 SSD 驱动器来支持此数量用户的磁盘 IO 要求。安装脚本安装实验室内容和 Docker。构建实验室 Docker 镜像，并设置所有用户。



软件安装



这个动手实验室使用 Visual Studio Code。Visual Studio Code 是一个代码编辑器，是 [GitHub 上](#) 最受欢迎的**开源**项目之一。它可以在 Linux, macOS 和 Windows 上运行。

安装 Visual Studio Code

1. 安装 Visual Studio Code

Visual Studio Code 扩展

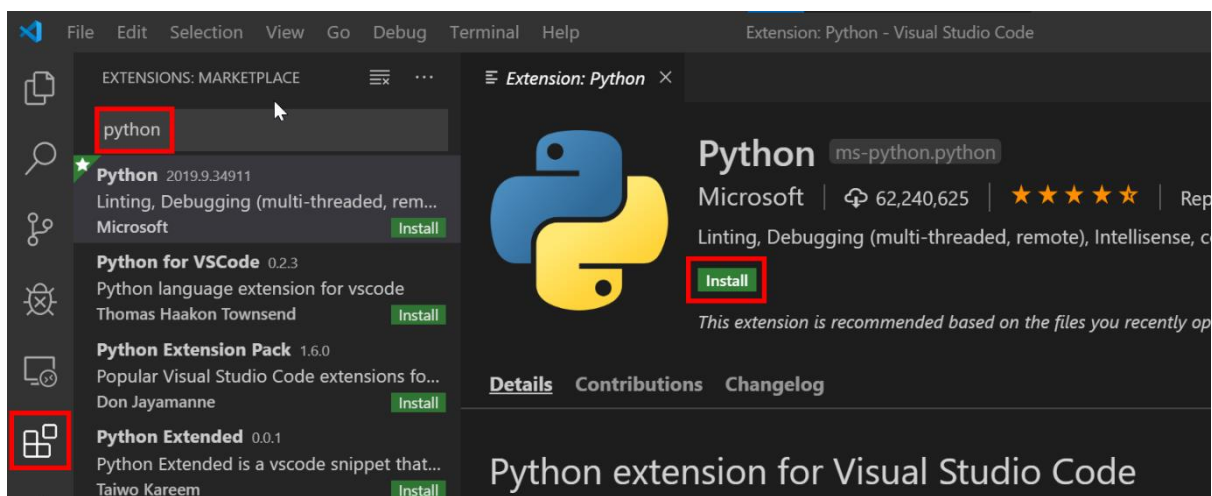
Visual Studio Code 包含开箱即用的功能只是一个开始。VS Code 扩展允许您为安装添加语言，调试器和工具，以支持您的开发工作流程。

浏览扩展程序

您可以从 Visual Studio Code 中搜索和安装扩展。从 Visual Studio Code 主菜单中打开 Extensions 视图，选择 **View > Extensions** 或单击 Visual Studio Code 一侧活动栏中的 Extensions 图标。



这将显示 [VS Code Marketplace](#) 上最受欢迎的 VS Code 扩展的列表。



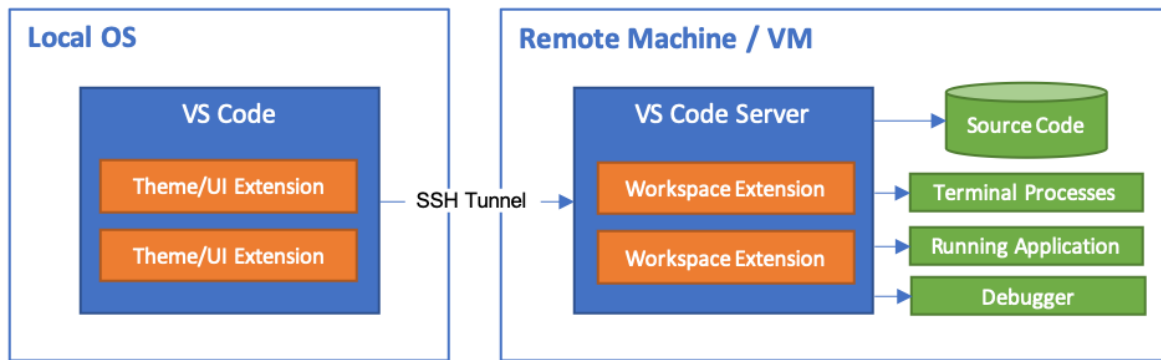
安装 Python, RemoteSSH 和 Docker Extensions

搜索并安装 Microsoft 发布的以下两个 Visual Studio Code 扩展。

1. Python
2. [Remote - SSH](#)
3. [Docker 扩展](#)

RemoteSSH 开发

Visual Studio Code Remote - SSH 扩展允许您打开任何远程计算机，虚拟机或容器，并充分利用 Visual Studio Code。



Raspberry Pi 硬件

您需要以下信息：

1. Raspberry Pi 的网络地址
2. 您的 Raspberry Pi 登录名和密码。

使用私钥/公钥进行 SSH 身份验证



为 [SSH](#) 身份验证设置公钥/私钥对是一种安全，快速的方法，可以从您的计算机进行身份验证到 Raspberry Pi。建议用于此动手实验。

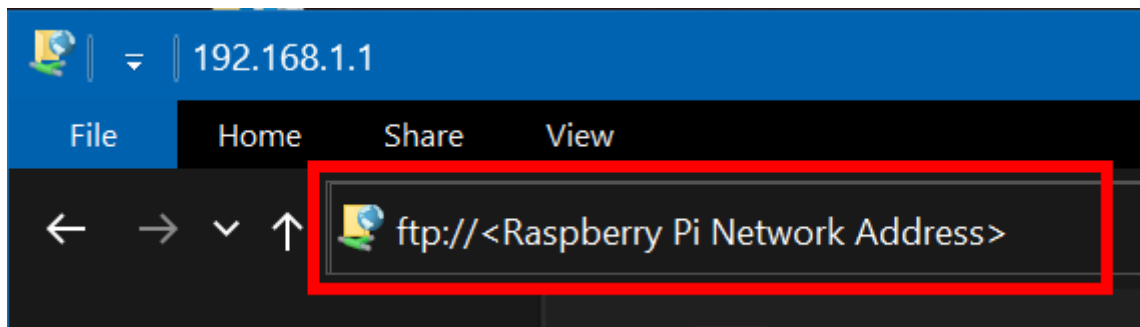
SSH 为 Windows 用户设置

SSH 实用程序将指导您完成为 Visual Studio Code 和 Raspberry Pi 设置安全 SSH 通道的过程。

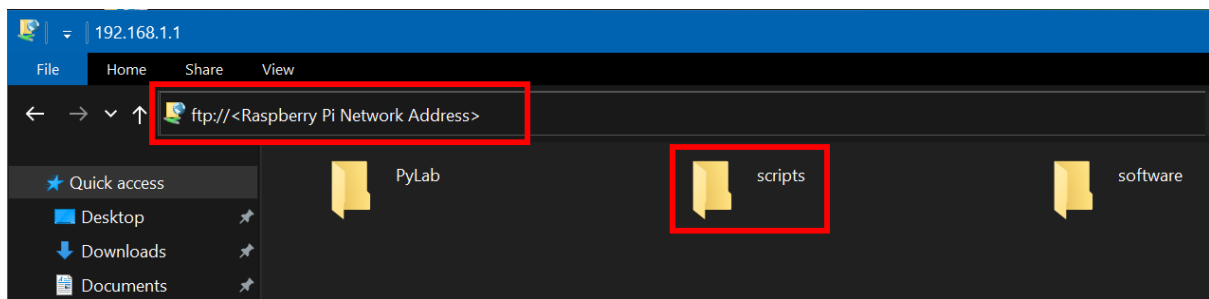
系统将提示您：

- Raspberry Pi 网络 IP 地址,
- Raspberry Pi 登录名和密码

1. 从 **Windows 文件资源管理器**中， 打开 **ftp://<Raspberry Pi Address>**



2. 将脚本目录复制到桌面



3. 打开复制到桌面的脚本文件夹

4. 双击 windows-setup-ssh.cmd

SSH 为 Linux 和 macOS 用户设置

SSH 实用程序将指导您完成成为 Visual Studio Code 和 Raspberry Pi 设置安全 SSH 通道的过程

系统将提示您：

- Raspberry Pi 网络 IP 地址,
- Raspberry Pi 登录名和密码

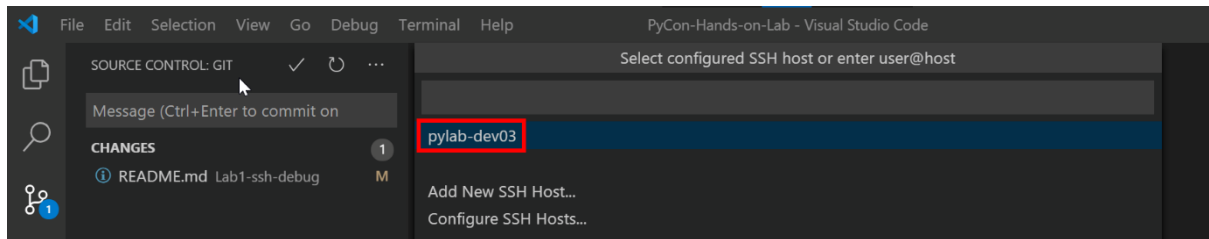
1. 打开终端窗口

2. 复制并粘贴以下命令，然后按 ENTER 键

```
read -p "Enter the Raspberry Pi Address: " pyurl && \  
curl ftp://$pyurl/scripts/ssh-setup.sh | bash
```

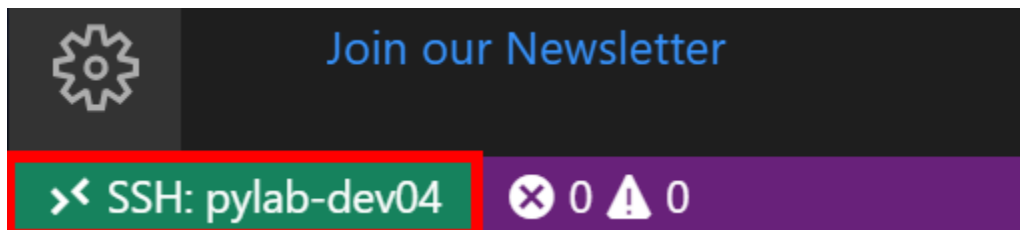
启动 RemoteSSH 连接

1. 启动 Visual Studio Code
2. 按 **F1** 打开命令选项板，键入 **ssh connect** 并选择 **Remote-SSH : Connect to Host**
3. 选择 **pylab-devnn** 配置



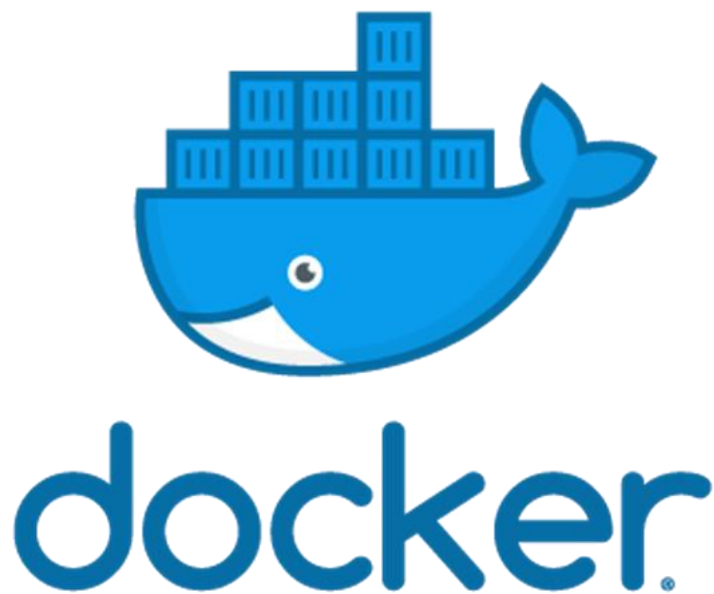
4. 检查 RemoteSSH 是否已连接。

连接需要一些时间，然后 Visual Studio Code 左下角的 SSH Status 将更改为 **> <SSH : pylab-devnn**。devnn 是你的 Raspberry Pi 登录名。



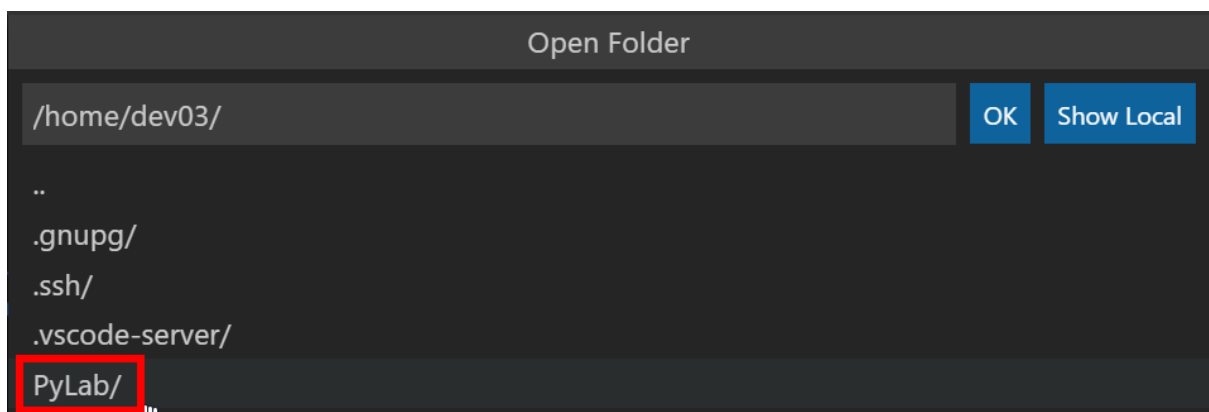
Docker 简介

[Jake Wright 的 Docker 12 分钟](#)是对 Docker 的精彩介绍。

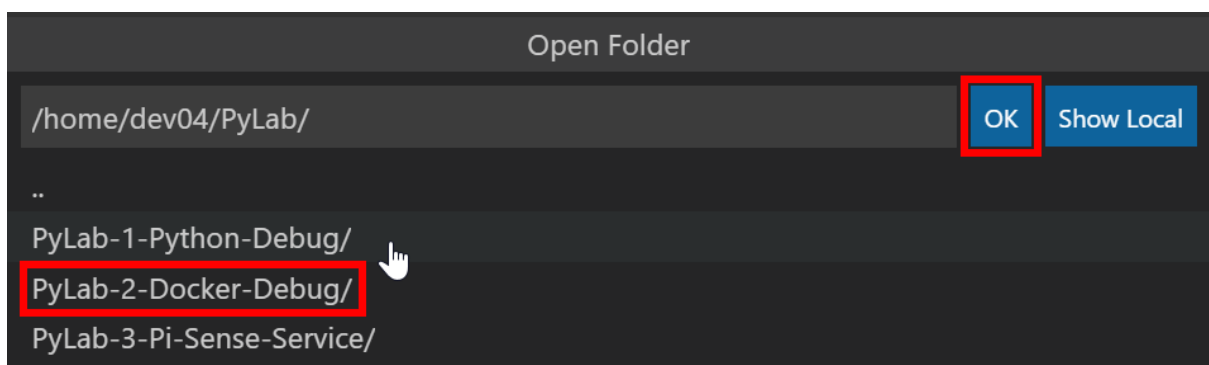


打开 PyLab 2 Docker 调试项目

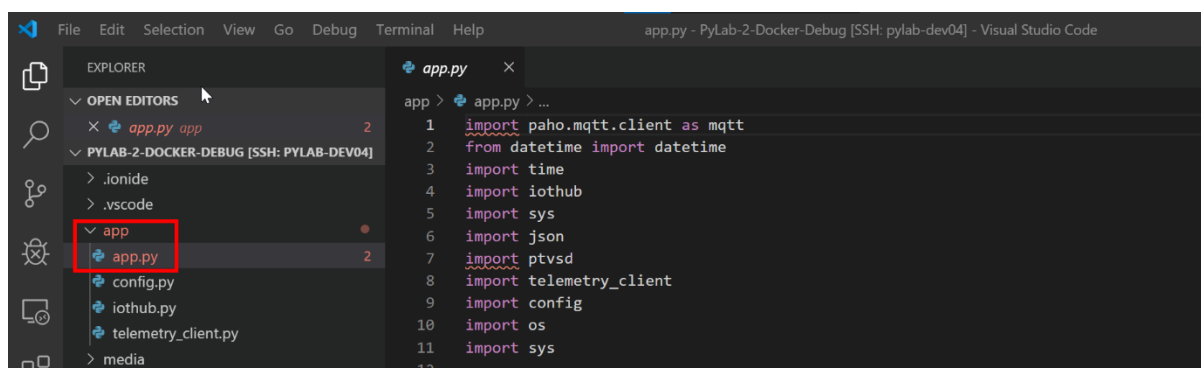
1. 从 Visual Studio Code 主菜单：文件 > 打开文件夹
2. 选择 **PyLab** 目录



3. 接下来选择 **PyLab-2-Docker-Debug** /目录



4. 单击“**确定**”以打开目录
5. 在资源管理器栏中，打开 **app** 文件夹，然后打开 **app.py** 文件，并查看内容



创建 Azure IoT 中央应用程序

我们将创建一个 Azure IoT Central 应用程序，然后创建一个设备，最后创建一个将在 Docker 容器中运行的应用程序所需的设备连接字符串。



创建新的 IoT 中央应用程序

1. 在新的浏览器选项卡中打开 [Azure IoT Central](#)，然后单击“**开始**”。
2. 接下来，您需要使用 **Microsoft Personal**，**Work** 或 **School** 帐户进行签名。如果您没有 Microsoft 帐户，则可以使用“**创建**”帐户免费创建一个帐户！



Sign in

to continue to Azure IoT Central

Email, phone, or Skype

[Can't access your account?](#)

No account? [Create one!](#)

Next

3. 创建一个新的 Azure IoT Central 应用程序，选择 **New Application**。这会将您带到“创建应用程序”页面。
4. 选择 **Trial, Custom application**，命名您的 IoT Central 应用程序并完成注册信息。

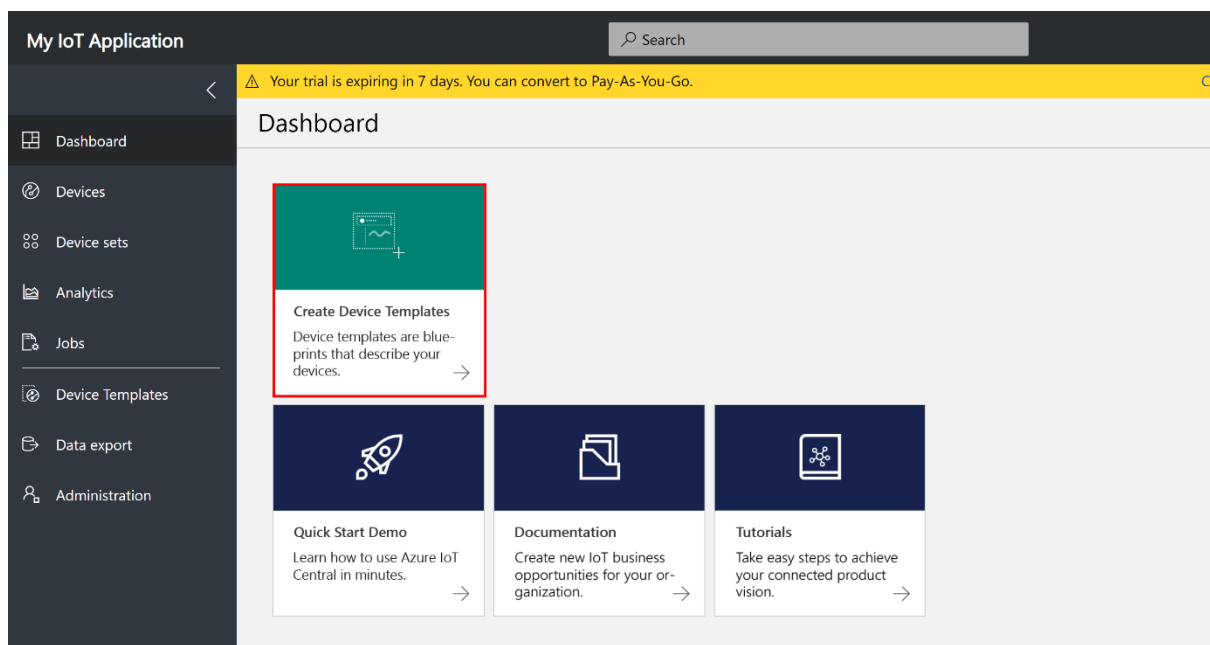
Choose a payment plan

<input checked="" type="radio"/> Trial	<input type="radio"/> Pay-As-You-Go
Free trial for 7 days. No subscription required.	Price is based on the number of devices you use. Free for the first 5 devices. Subscription required. Learn more

Select an application template

<input type="radio"/> Sample Contoso	<input type="radio"/> Sample Devkits	<input checked="" type="radio"/> Custom application
Get started with a predefined application for a connected device.	Want to connect a Raspberry PI or MXChip IoT DevKit? Start with this predefined app and get them connected in minutes.	Start with a blank template and define your application from scratch.

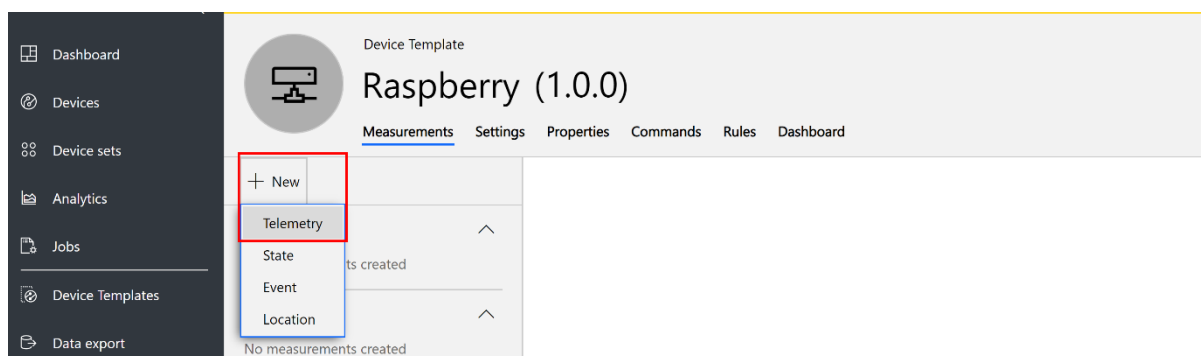
1. 单击“创建设备模板”，然后选择“自定义”模板，为模板命名，例如，**Raspberry**。然后单击 Create



2. 编辑模板，添加测量的温度，湿度和压力遥测。

测量值来自您的设备。您可以向设备模板添加多个测量值，以匹配设备的功能。

- **遥测**测量是设备随时间收集的数字数据点。它们被表示为连续的流。一个例子是温度。
- **事件**测量是时间点数据，代表设备上的重要事项。严重性级别表示事件的重要性。一个例子是风扇电机错误。
- **状态**测量表示设备或其组件在一段时间内的状态。例如，可以将扇形模式定义为具有两种可能状态的“**操作**”和“**已停止**”。
- **位置**测量值是设备在一段时间内的经度和纬度坐标。例如，风扇可以从一个位置移动到另一个位置。

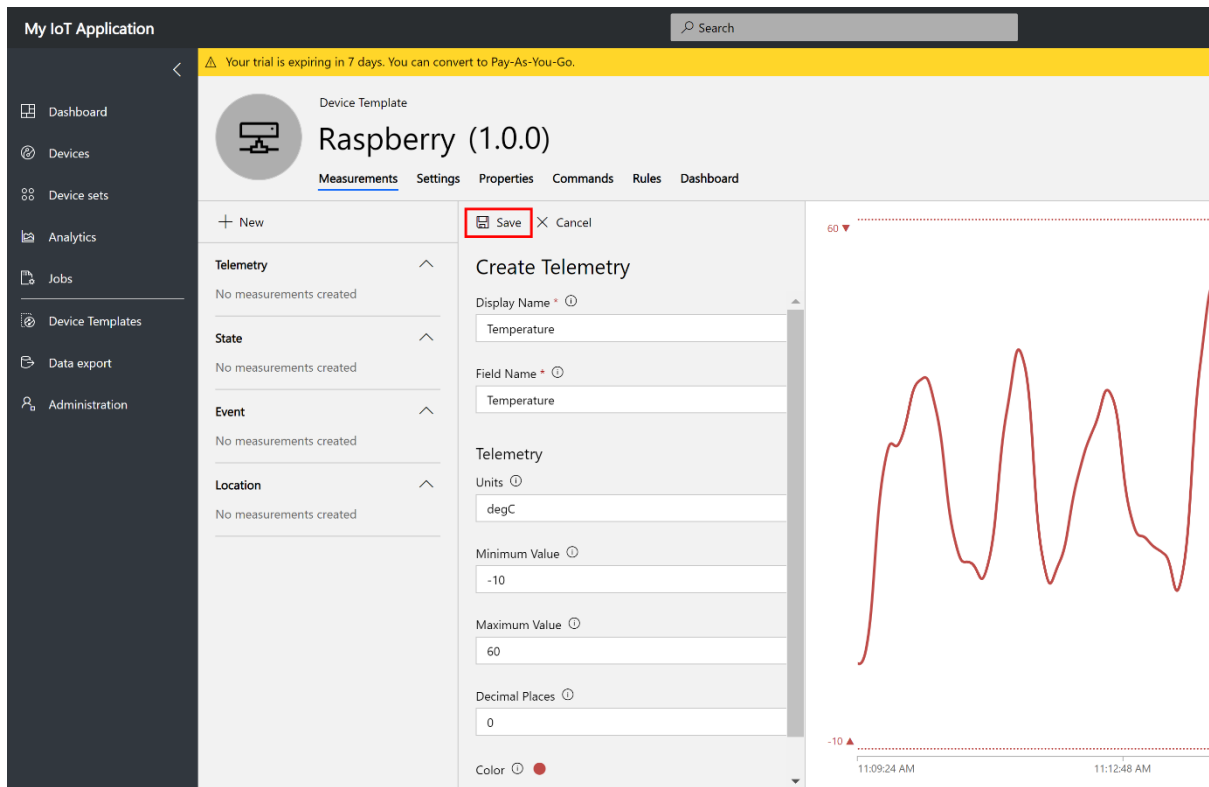


使用下表中的信息设置三个遥测测量。字段名称区分大小写。

你必须点击**保存**每次测量后确定。

显示名称	字段名称	单位	最小值	最大值	小数点
湿度	湿度	%	0	100	0
温度	温度	摄氏度	-10	60	0
压力	压力	百帕	800	1260	0

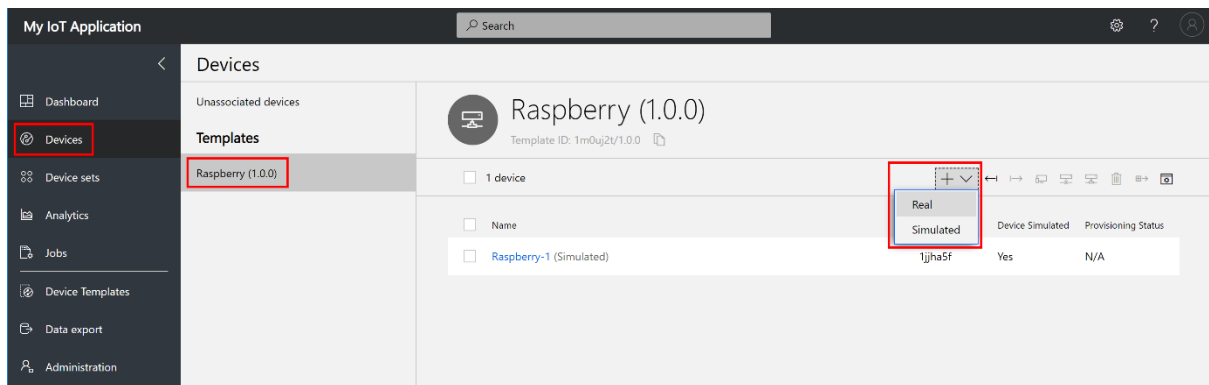
以下是设置**温度**遥测测量的示例。



3. 单击侧栏菜单上的“**Device**”，选择您创建的 **Raspberry** 模板。

物联网中心支持真实设备，例如本实验中使用的 Raspberry Pi，以及生成对系统测试有用的随机数据的模拟设备。

4. 选择 **Real**。



为您的 DeviceID 命名，以便在 IoT Central 门户中轻松识别设备，然后单击“**Create**”。

Create New Device

Device ID * ⓘ

raspberry-pi-01

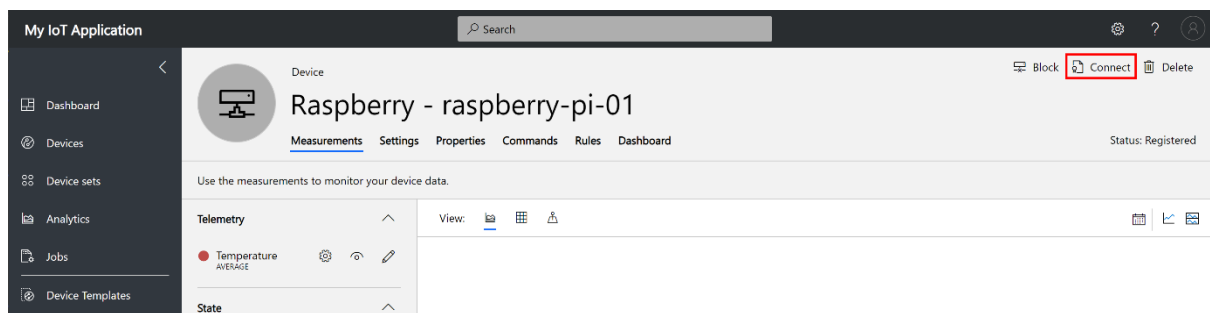
Device Name ⓘ

Raspberry - raspberry-pi-01

Create

Cancel

5. 创建真实设备后，单击屏幕右上角的“**Connect**”按钮以显示设备凭据。



保持此页面处于打开状态，因为您将需要此连接信息以用于动手实验的下一步骤。

Device Connection

Scope ID ⓘ
0n1 F116

Device ID ⓘ
raspberrypi-01

Credentials

Shared Access Signature (SAS) Certificates (X.509)

SAS security tokens are an attestation mechanism for devices to connect to IoT Central. The group SAS keys for this device are shown below. Use them to register your device with IoT Central. [Click to learn more.](#)

Primary Key ⓘ
qflrRG+jA23...PeMWEQUP35Q=

Secondary Key ⓘ
EQ3iQjiy...14kvKV3GdJx5a/HgbNQ=

Close

生成 Azure IoT 中心连接字符串

1. 按住控制键，然后单击以下链接“[Connection String Generator](#)”以在新选项卡中打开。

将“ScopeID”，“DeviceID”和“Primary Key”从“Azure IoT 中央设备连接”面板复制并粘贴到“连接字符串生成器”页面，然后单击“Get Connection String”。

Azure IoT Central Connection String Generator

Scope

Device Id

Device Key

Get Connection String

0n1 F116

my-device

UzZtjO...gaszy5RAn/j+bSEfEjBW7w3V145lp/c=

2. 将生成的连接字符串复制到剪贴板，因为下一步需要它。

配置 Python 应用程序

1. 切换回 Visual Studio Code。打开 **env 文件**（环境文件）。此文件包含将传递给 Docker 容器的环境变量。
2. 将您在上一步中复制的连接字符串粘贴到同一行的 env 文件中，并加在 **CONNECTION_STRING =**之后。

例如：

```
CONNECTION_STRING=HostName=saas-iothub-8135cd3b....
```

3. 保存 env file 文件（Ctrl + S）

构建并运行 Docker 镜像

按 **F5** 开始调试 Python 应用程序。该过程将首先构建然后启动 Docker Container。当 Docker Container 启动时，Visual Studio Code Debugger 将附加到正在运行的应用程序。

在.vscode 文件夹中有两个配置文件，负责运行和调试 Python 应用程序。您可以在 [Debugger Configuration](#) 附录中找到更多详细信息。

设置 Visual Studio 调试程序断点

1. 从 Visual Studio Code 活动栏上的资源管理器中，打开 **app.py** 文件
2. 在发布函数中的第 66 行设置断点，**温度，压力，湿度，timestamp = mysensor.measure（）**。

。您可以通过执行以下任一操作来设置断点：

- 将光标放在该行上，按 **F9**，或者，
- 将光标放在该行上，选择 **Debug> Toggle Breakpoint** 菜单命令，或者直接单击行号左边的边距（悬停在那里时会出现一个褪色的红点）。断点在左边距中显示为红点：

```

62 def publish():
63     msgId = 1
64     while True:
65         try:
66             temperature, pressure, humidity, timestamp, cpu_temperature = mysensor.measure()
67         except:
68             pass
69         data = {
70             "Geo": 'Sydney, AU',
71             "Humidity": humidity,
72             "Pressure": pressure,
73             "Temperature": temperature,
74             "CpuTemperature": cpu_temperature,
75             "Epoch": timestamp,
76             "Id": msgId
77         }

```

调试操作

调试会话启动后，Debug toolbar 将显示在编辑器窗口的顶部。

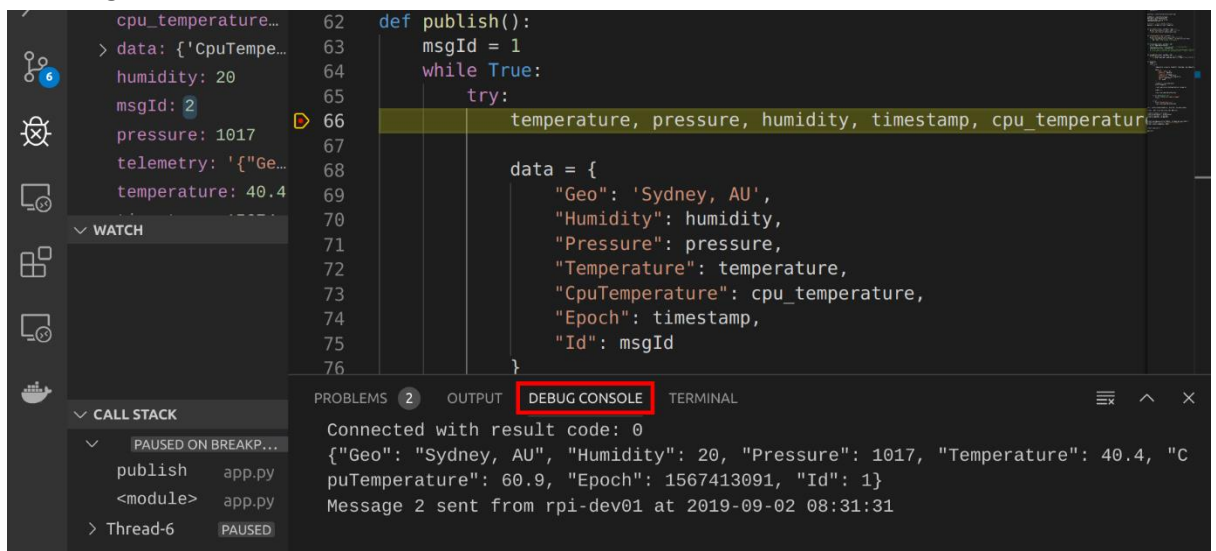


调试器工具栏（如上所示）将出现在 Visual Studio Code 中。它有以下选项：

1. Pause（或继续，F5），
2. Step Over（F10）
3. Step Into（F11），
4. Step Out（Shift + F11），
5. Restart（Ctrl + Shift + F5），
6. And Stop（Shift + F5）。

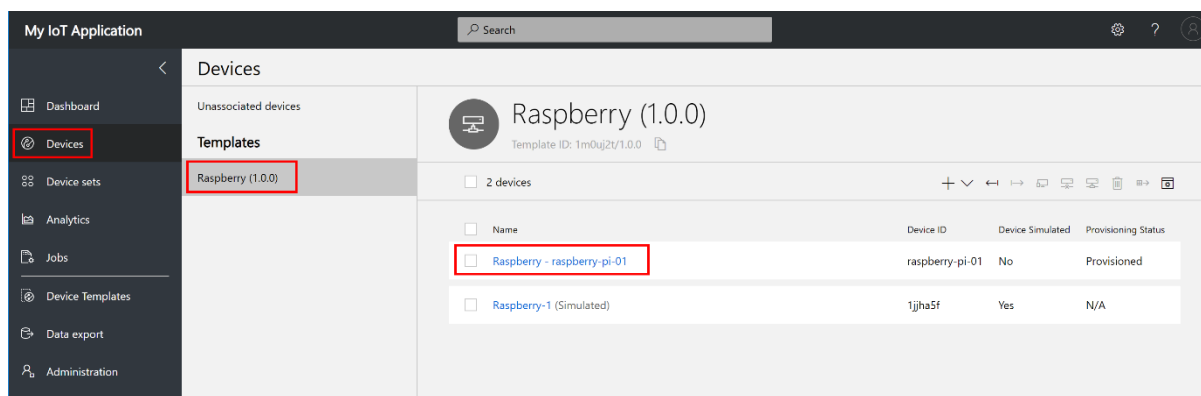
逐步完成 Python 代码

1. 按 **F10**，或从调试器工具栏中单击“Step Over”，直到超过 `print(telemetry)` 代码行。
2. 浏览 **Variable Window**（Ctrl + Shift + Y）。尝试更改变量值。
3. 浏览调 **Debug Console**。您将看到传感器遥测以及将遥测发送到 Azure IoT Central 的结果。

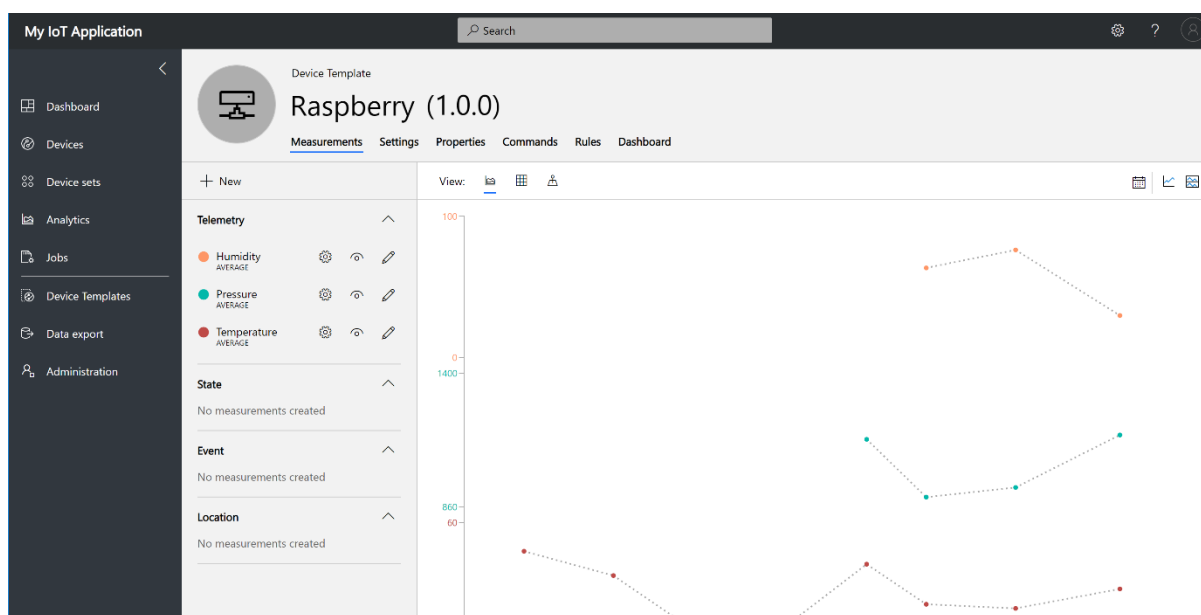


4. 从 **Debug** 菜单 -> **Disable All Breakpoints**
5. 按 **F5** 或从调试器工具栏中单击 **Continue**，以便运行 Python 应用程序并将遥测流式传输到 **Azure IoT Central**。

1. 探索 **Azure IoT Central** 中的 **Device Telemetry** 使用 **Device** 导航到您添加的真实 Raspberry Pi 设备的“**Measurements**”页面：



2. 在 **Measurements** 页面上，您可以看到 Raspberry Pi 设备的 **telemetry streaming**：



成品



附录

Debugger Configuration

在.vscode 文件夹中有两个文件（ launch.json 和 tasks.json），负责运行和调试应用程序。

启动配置

创建启动配置文件非常有用，因为它允许您配置和保存调试设置详细信息。

launch.json

```
{
  // Use IntelliSense to learn about possible attributes.
  // Hover to view descriptions of existing attributes.
  // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Attach Debugger",
      "preLaunchTask": "start-docker",
      "postDebugTask": "stop-docker",
      "type": "python",
      "request": "attach",
      "pathMappings": [
        {
          "localRoot": "${workspaceRoot}/app",
          "remoteRoot": "/app"
        }
      ],
      "port": "${env:LAB_PORT}",
      "host": "localhost"
    },
    {
      "name": "Stop Container",
      "preLaunchTask": "stop-docker",
      "type": "python",
      "request": "launch"
    }
  ]
}
```

```
}  
}
```

Tasks Configuration

任务集成了外部工具以自动化构建周期。

tasks.json

```
{  
  // See https://go.microsoft.com/fwlink/?LinkId=733558  
  // for the documentation about the tasks.json format  
  "version": "2.0.0",  
  "tasks": [  
    {  
      "label": "start-docker",  
      "type": "shell",  
      "command": "sh",  
      "args": [  
        "-c",  
        "\"docker build -t $USER:latest . ;docker run -d -p $LAB_PORT:3000 -e  
TELEMETRY_HOST=$LAB_HOST --env-file env-file --name $USER --rm $USER:latest; sleep 1 \""  
        // -d Run container in background and print container ID,  
        // -p maps the $LAB_PORT to port 3000 in the container, this port is used for  
debugging,  
        // -e Environment Variable. The IP Address of the telemetry service.  
        // --env-file reads from a file and sets Environment Variables in the Docker  
Container,  
        // --name names the Docker Container  
        // --rm removes the container when you stop it  
        // Docker run reference https://docs.docker.com/engine/reference/run/  
      ],  
    },  
    {  
      "label": "stop-docker",  
      "type": "shell",  
      "command": "sh",  
      "args": [  
        "-c",  
        "\"docker stop $USER\""br/>      ]  
    }  
  ]  
}
```

Azure IoT Central (Azure 物联网中心)

浏览 Azure IoT Central UI

本文向您介绍 Microsoft Azure IoT 中央 UI。您可以使用 UI 来创建、管理和使用 Azure IoT Central 解决方案及其连接的设备。

作为 *构建者*，您可以使用 Azure IoT Central UI 来定义 Azure IoT Central 解决方案。您可以使用 UI 来：

- 定义连接到解决方案的设备类型。

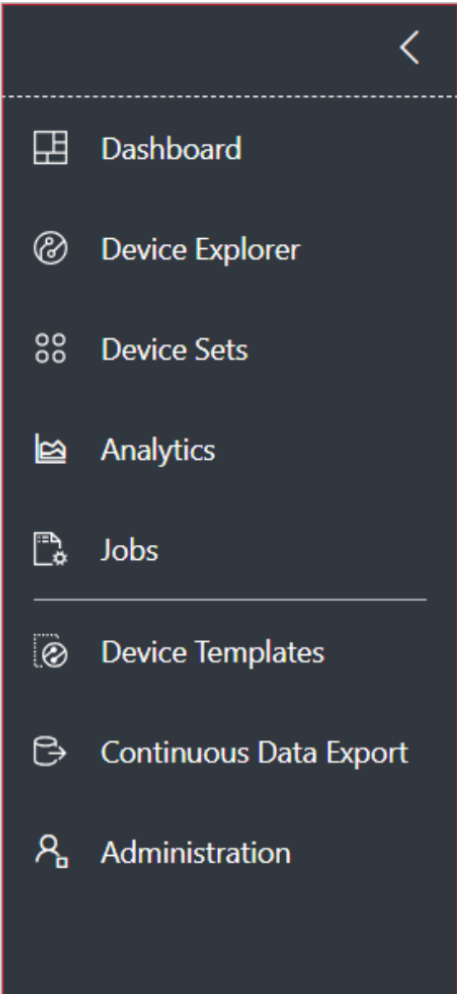
- 配置设备的规则和操作。
- 为使用您的解决方案的 **操作员** 自定义 UI 。

作为 **操作者**，您可以使用 Azure IoT Central UI 来管理 Azure IoT Central 解决方案。您可以使用 UI 来：

- 监控您的设备。
- 配置您的设备。
- 排除故障并解决设备问题。
- 提供新设备。

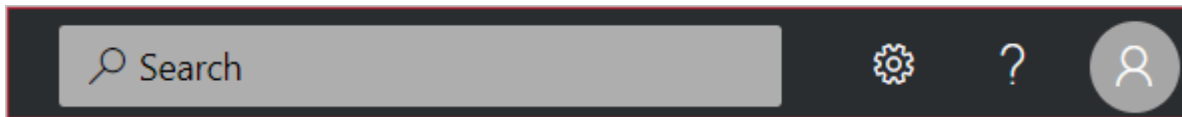
使用左侧导航菜单

使用左侧导航菜单访问应用程序的不同区域。您可以通过选择 **<或>** 来展开或折叠导航栏：

Menu	Description
	<ul style="list-style-type: none"> • The Dashboard button displays your application dashboard. As a builder, you can customize the dashboard for your operators. Users can also create their own dashboards. • The Device Explorer button lists the simulated and real devices associated with each device template in the application. As an operator, you use the Device Explorer to manage your connected devices. • The Device Sets button enables you to view and create device sets. As an operator, you can create device sets as a logical collection of devices specified by a query. • The Analytics button shows analytics derived from device telemetry for devices and device sets. As an operator, you can create custom views on top of device data to derive insights from your application. • The Jobs button enables bulk device management by having you create and run jobs to perform updates at scale. • The Device Templates button shows the tools a builder uses to create and manage device templates. • The Continuous Data Export button an administrator to configure a continuous export to other Azure services such as storage and queues. • The Administration button shows the application administration pages where an administrator can manage application settings, users, and roles.

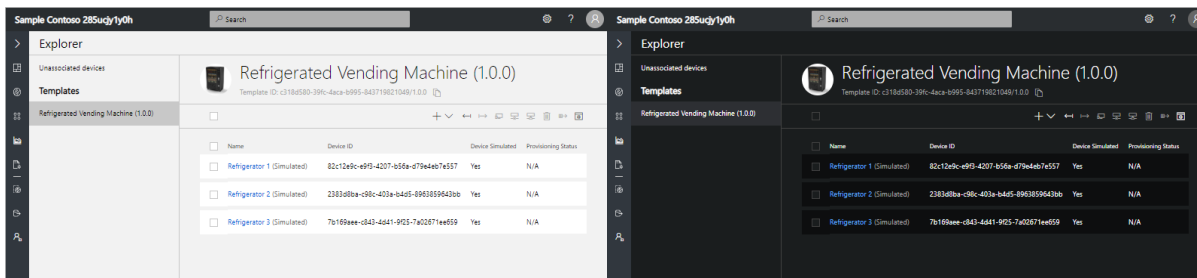
搜索，帮助和支持

顶部菜单显示在每个页面上：

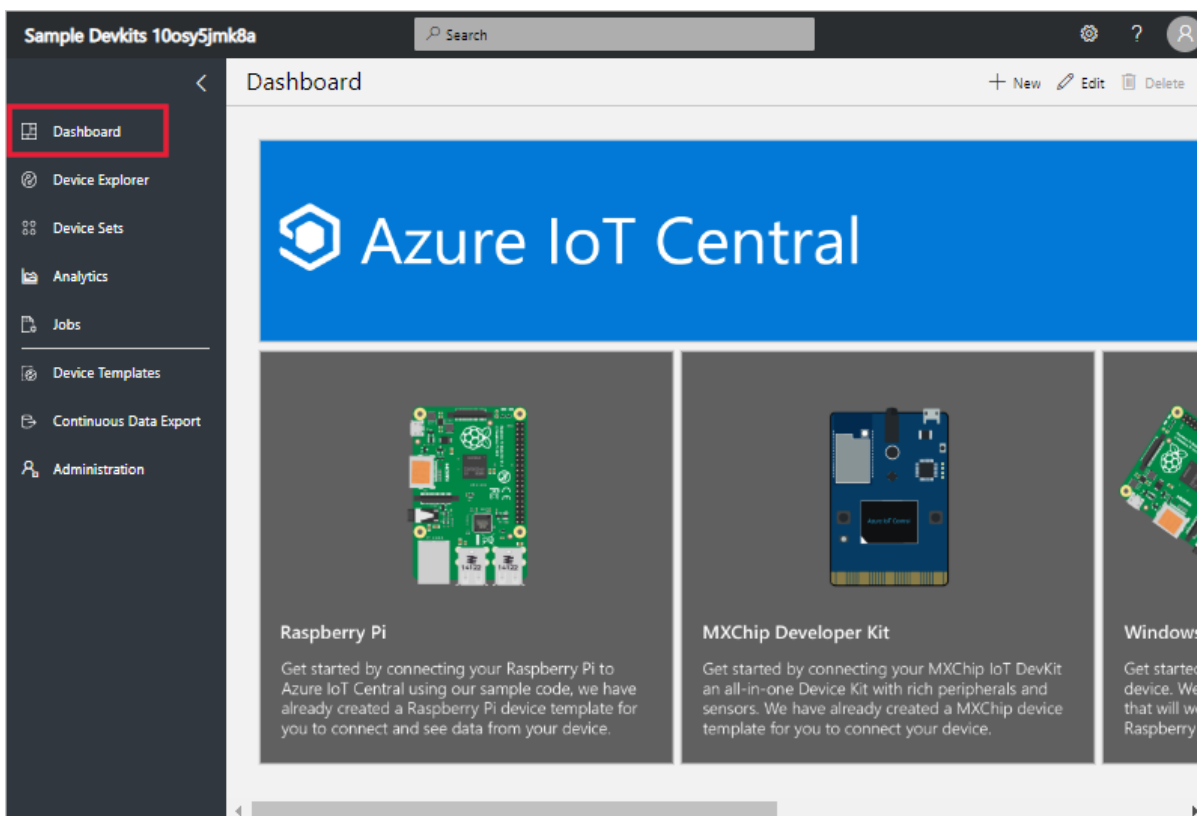


- 要搜索设备模板和设备，请输入**搜索值**。
- 要更改 UI 语言或主题，请选择“**设置**”图标。
- 要退出应用程序，请选择“**帐户**”图标。
- 要获得帮助和支持，请选择“**帮助**”下拉列表以获取资源列表。在试用版应用程序中，支持资源包括访问 [live chat](#)。

您可以为 UI 选择浅色主题或黑暗主题：

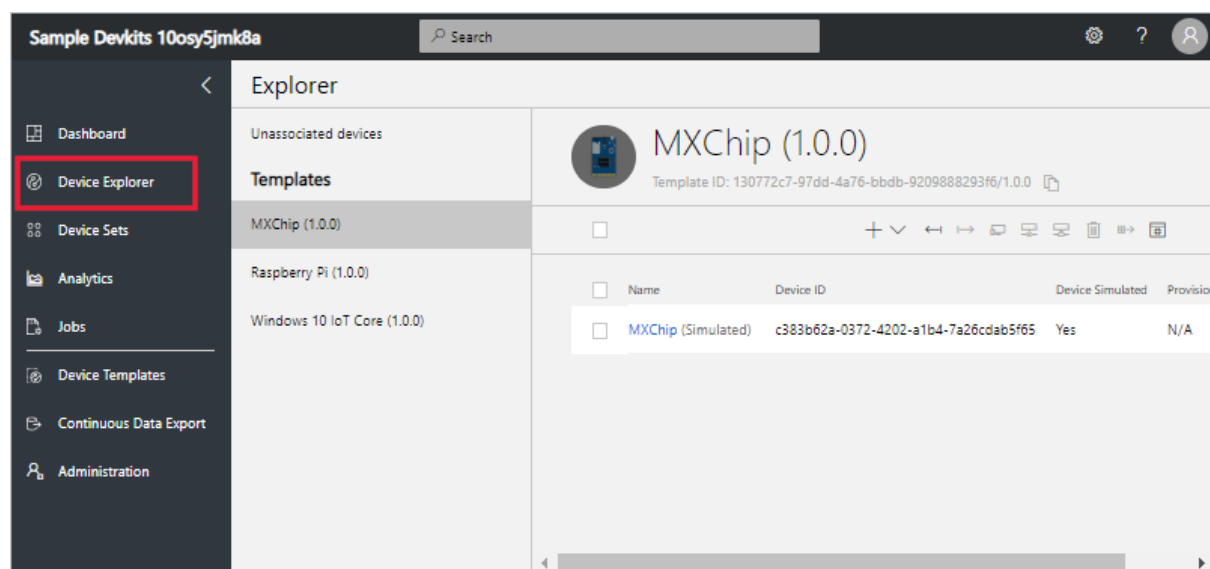


Dashboard (仪表板)



仪表板是您登录 Azure IoT Central 应用程序时看到的第一个页面。作为构建者，您可以通过添加切片为其他用户自定义应用程序仪表板。有关详细信息，请参阅 [Customize the Azure IoT Central operator's view](#) 教程。用户还可以 [create their own personal dashboards](#)。

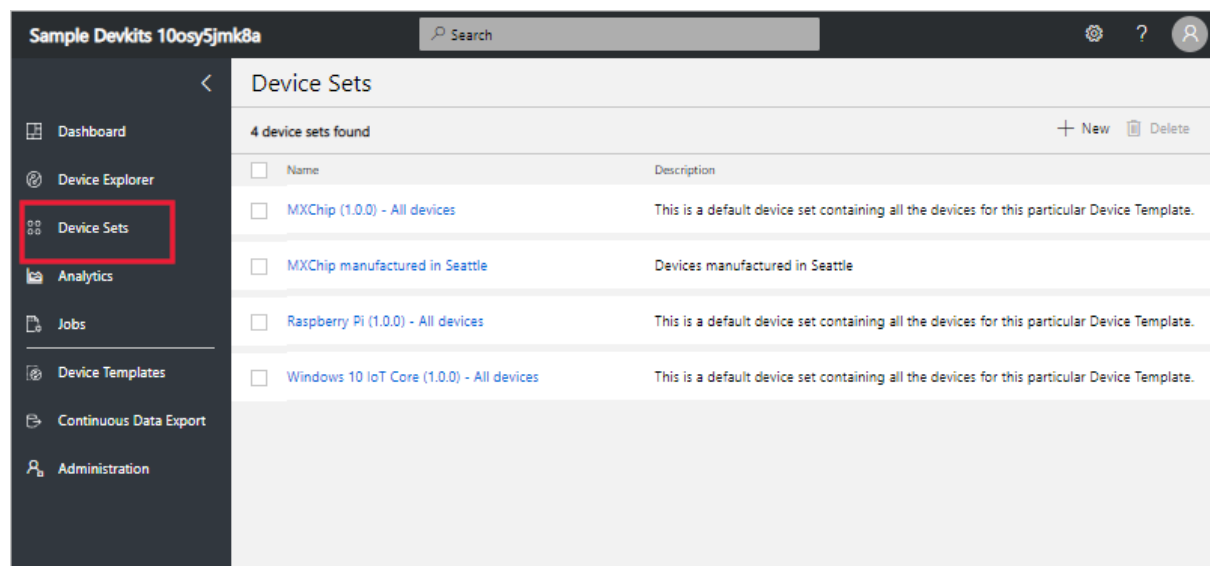
Device Explorer (设备资源管理器)



探险页面显示设备中通过分组的 Azure 的物联网应用中心 设备模板。

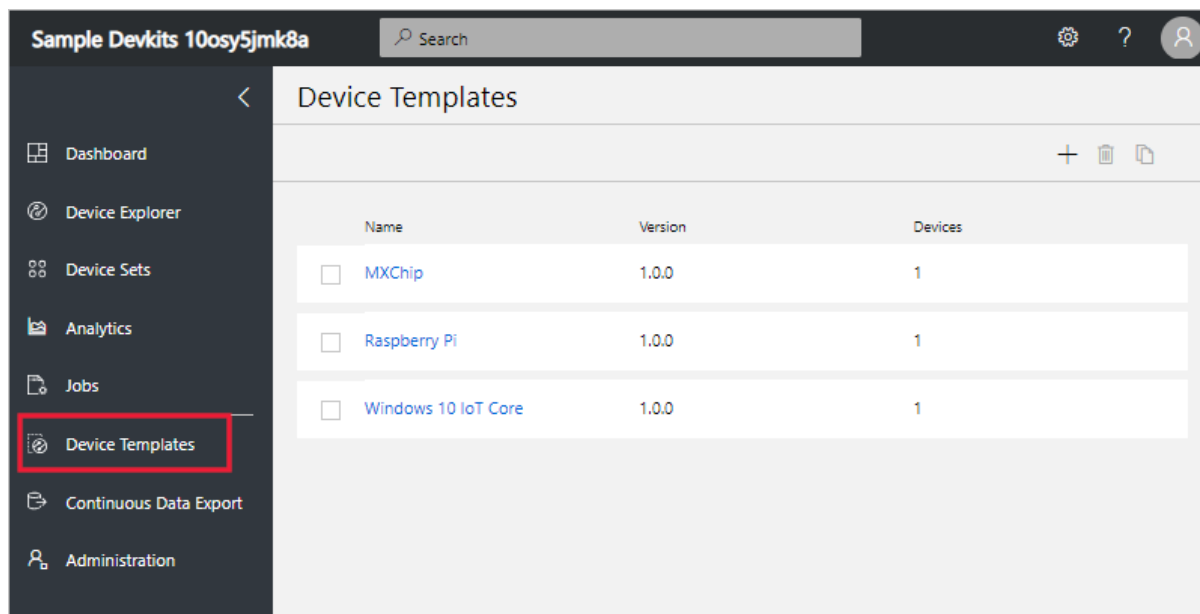
- 设备模板定义了可以连接到您的应用程序的设备类型。有关详细信息，请参阅 [Define a new device type in your Azure IoT Central application](#).
- 设备表示应用程序中的真实或模拟设备。有关详细信息，请参阅 [Add a new device to your Azure IoT Central application](#)。

Device sets (设备集)



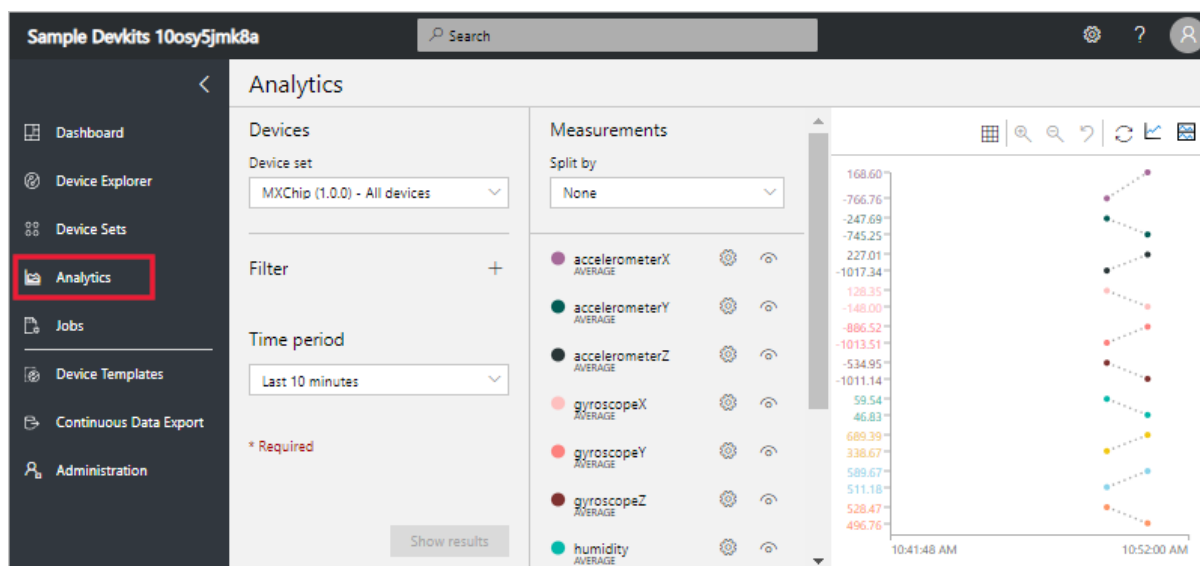
Device Set 页面显示 **Builder** 创建设备组。设备集是相关设备的集合。构建者定义查询以标识设备集中包含的设备。在应用程序中自定义分析时，可以使用设备集。有关详细信息，请参阅 [Use device sets in your Azure IoT Central application](#)。

Device Template 设备模板



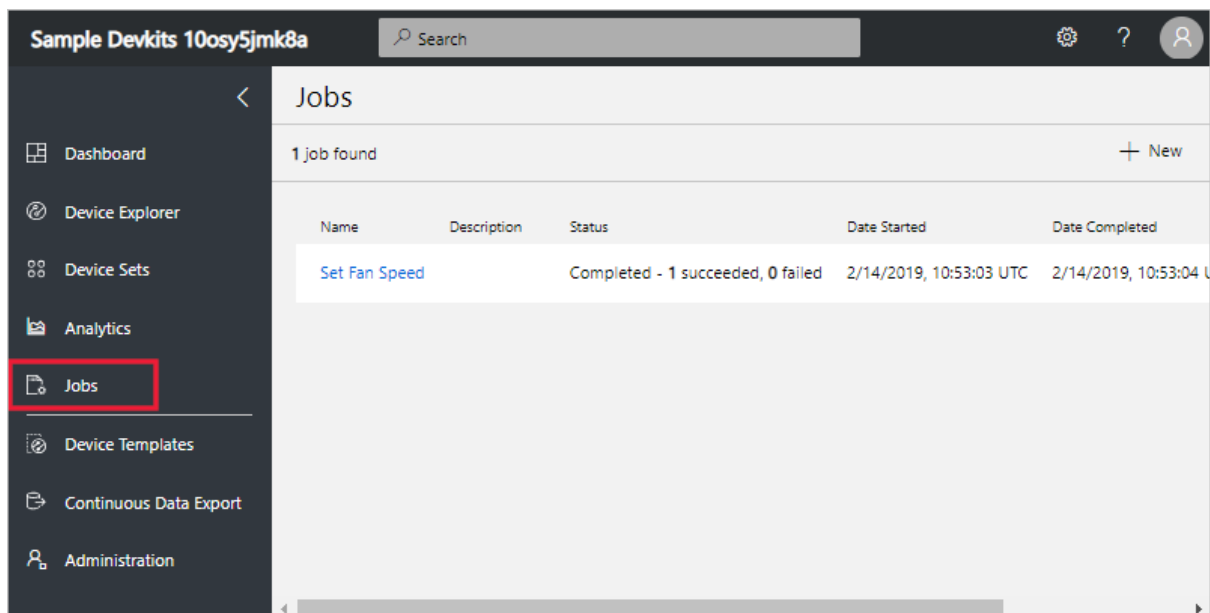
设备模板页面是构建者在应用程序中创建和管理设备模板的位置。有关详细信息，请参阅 [“Define a new device type in your Azure IoT Central application”](#) 教程。

Analytics（分析）



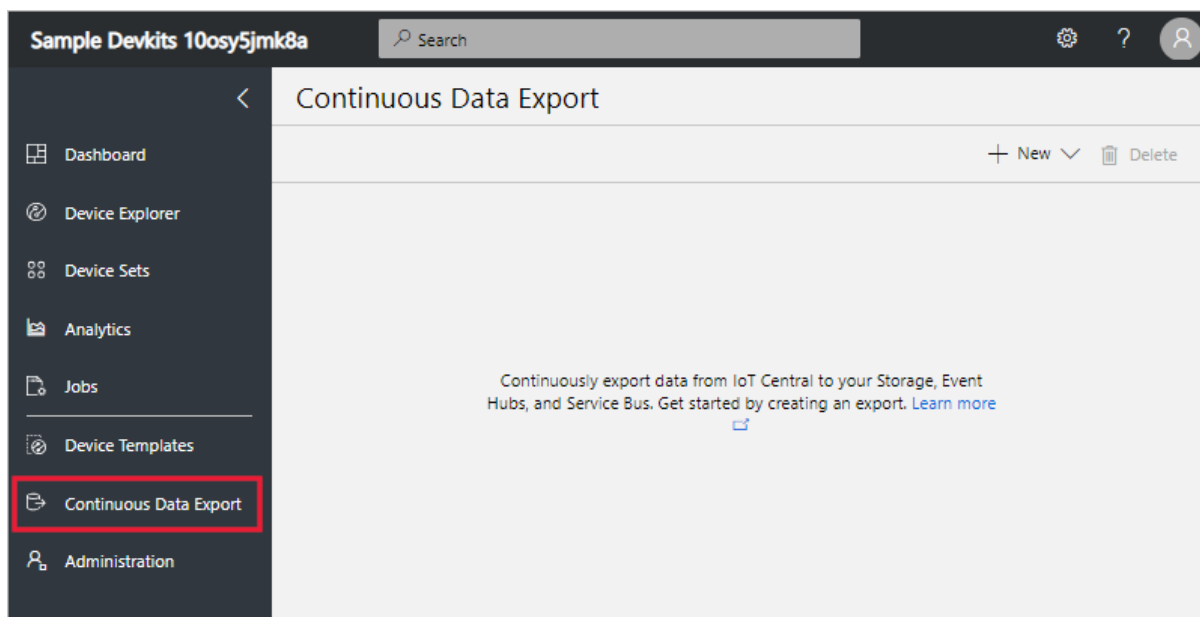
分析页面显示的图表可帮助您了解连接到应用程序的设备的行为方式。操作员使用此页面来监控和调查连接设备的问题。构建器可以定义此页面上显示的图表。有关详细信息，请参阅 [Create custom analytics for your Azure IoT Central application](#) 文章。

Jobs（作业）



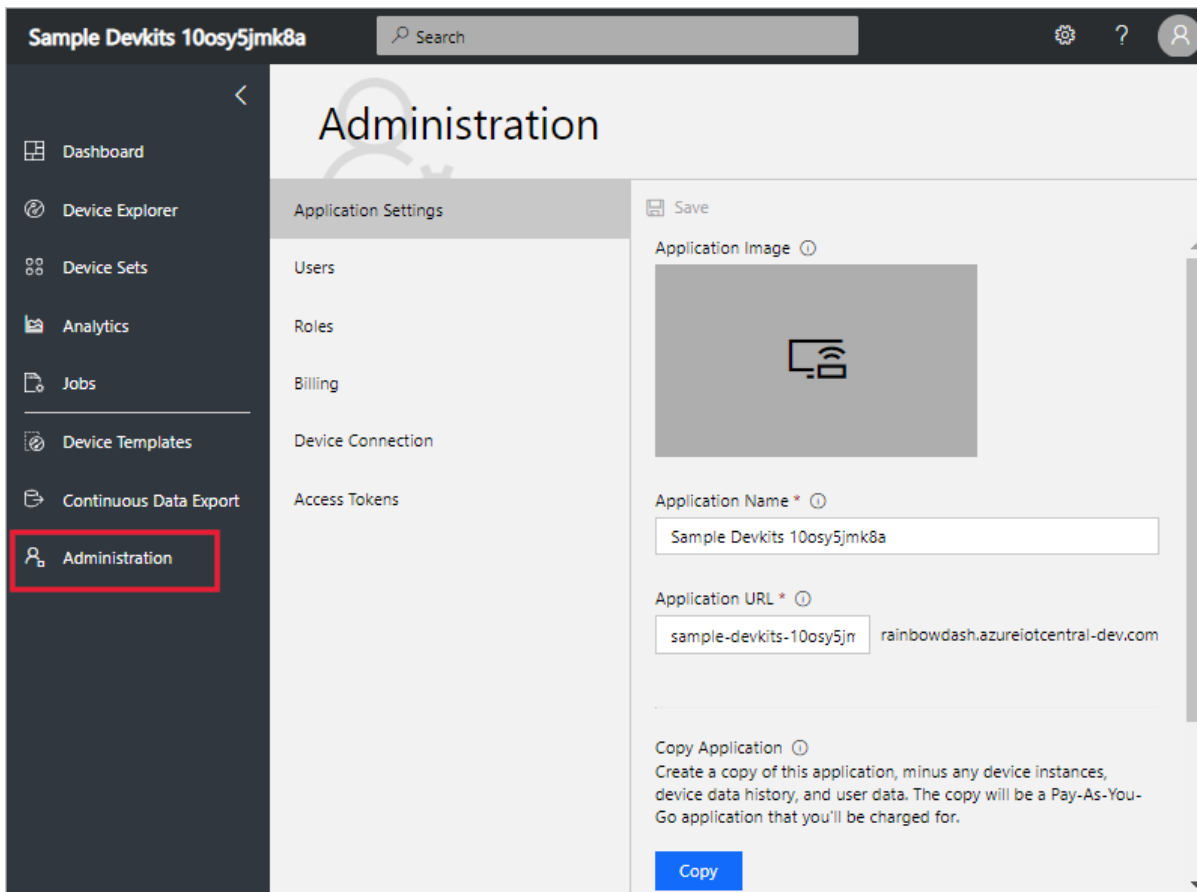
“Jobs”页面允许您在设备上执行批量设备管理操作。构建器使用此页面更新设备属性，设置和命令。要了解更多信息，请参阅 [Run a job](#) 文章。

Continuous Data Export



连续数据导出页面是管理员定义如何从应用程序导出数据（如遥测）的位置。其他服务可以存储导出的数据或将其用于分析。有关详细信息，请参阅 [Export your data in Azure IoT Central](#) 文。

Administration



管理页面包含管理员使用的工具的链接，例如在应用程序中定义用户和角色。有关详细信息，请参阅 [Administer your Azure IoT Central application](#) 文章。

参考

- [Visual Studio Code](#)
- [Azure IoT Central](#)
- [Installing Docker on Raspberry Pi Buster](#)
- [Understanding Docker in 12 Minutes](#)

[PyLab-2-Python-Azure-IoT-Central-and-Docker-Container-Debugging](#) 由 Dave Glover 维护。