Amazon Web Services: Getting started on the Raspberry Pi 5

Colophon

Release	2
Build date	21/11/2025
Build version	02feeb6684b8

Legal disclaimer notice

TECHNICAL AND RELIABILITY DATA FOR RASPBERRY PI PRODUCTS (INCLUDING DATASHEETS) AS MODIFIED FROM TIME TO TIME (“RESOURCES”) ARE PROVIDED BY RASPBERRY PI LTD (“RPL”) “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW IN NO EVENT SHALL RPL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THE RESOURCES, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

RPL reserves the right to make any enhancements, improvements, corrections or any other modifications to the RESOURCES or any products described in them at any time and without further notice.

The RESOURCES are intended for skilled users with suitable levels of design knowledge. Users are solely responsible for their selection and use of the RESOURCES and any application of the products described in them. User agrees to indemnify and hold RPL harmless against all liabilities, costs, damages or other losses arising out of their use of the RESOURCES.

RPL grants users permission to use the RESOURCES solely in conjunction with the Raspberry Pi products. All other use of the RESOURCES is prohibited. No licence is granted to any other RPL or other third party intellectual property right.

HIGH RISK ACTIVITIES. Raspberry Pi products are not designed, manufactured or intended for use in hazardous environments requiring fail safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, weapons systems or safety-critical applications (including life support systems and other medical devices), in which the failure of the products could lead directly to death, personal injury or severe physical or environmental damage (“High Risk Activities”). RPL specifically disclaims any express or implied warranty of fitness for High Risk Activities and accepts no liability for use or inclusions of Raspberry Pi products in High Risk Activities.

Raspberry Pi products are provided subject to RPL’s Standard Terms. RPL’s provision of the RESOURCES does not expand or otherwise modify RPL’s Standard Terms including but not limited to the disclaimers and warranties expressed in them.

Document version history

Release	Date	Description
1	20 Nov 2025	Initial release
2	21 Nov 2025	Update GitHub repository name

Scope of document

This document applies to the following Raspberry Pi products:

Single Board Computers / SBCs

Pi Zero			Pi Zero 2		Pi 1		Pi 2		Pi 3	Pi 4	Pi 5
-	W	H	W	H	A	B	A	B	B	-	-
											✓

Introduction

Applicable operating systems for this guide

This guide assume you are using Raspberry Pi OS, with a full desktop and browser. Other OS installations (e.g. Ubuntu) are also likely to work, but are not guaranteed. Some instructions require use of a browser for setting up Amazon Web Services. A full Raspberry Pi OS installation includes both Chromium and Firefox.

Overview

This document describes how to use the Raspberry Pi 5 to connect to an Amazon Web Services (AWS) instance. As the Raspberry Pi is a general-purpose computing device, rather than providing a specific AWS IoT application, this document describes the basics of obtaining certificates, adding them to the Raspberry Pi 5 and testing the device with AWS Device Compliance.

This example uses Python and the AWS Python bindings to provide proof of principle. It does not go in to detail on how to write AWS applications, nor does it cover the use of the AWS IoT C/C++ SDK.

Hardware Description

The Raspberry Pi 5 is a Small Board Computer (SBC) from Raspberry Pi Ltd. It is a general purpose computing device that can leverage AWS APIs and libraries to connect to AWS IoT.

Product Datasheet

Product information on the Raspberry Pi 5 can be found here.

Product Brief

https://pip-assets.raspberrypi.com/categories/892-raspberry-pi-5/documents/RP-008348-DS-1-raspberry-pi-5-product-brief.pdf

Standard kit contents

The Raspberry Pi 5 is usually sold as a standalone item or it can be purchased in a kit with cables, case, power supply and user guide.

https://www.raspberrypi.com/products/raspberry-pi-5/

User provided items

Unless supplied in a kit, the user will need to provide a power supply, HDMI cable, monitor, mouse, keyboard and SD card. Raspberry Pi Ltd sell all of these items through their official reseller network.

https://www.raspberrypi.com/resellers

3rd party purchasable items

None

Additional hardware references

None

Set up your development environment

Tools installation (IDEs, Toolchains, SDKs)

1 IDE based
   a Raspberry Pi Ltd recommend Microsoft Visual Studio Code (VSCode) for software development on Raspberry Pi Ltd devices. However, this is not mandatory, users can use whichever IDE or development system they prefer.
   b If you are installing on a Raspberry Pi device, use the following:

```
Code
sudo apt update
sudo apt install code
```

   c On Windows or other Linux device, please follow the standard VSCode installation instructions for installation. https://code.visualstudio.com/download
   d You may find it useful to install VSCode extensions for Python.
2 CLI based
   a The example code described here is written in Python and can be easily run from the command line.
3 Python 3 is required, along with the AWS Python bindings.

Creating a project folder and Python virtual environment

The latest versions of Python and Raspberry Pi OS require the use of virtual environments in which the Python packages needed for the applications are installed.

To create a project folder and a Python Virtual Environment, do the following:

```
Code
mkdir Pi5AWSExample
cd Pi5AWSExample

# Create the virtual environment
python3 -m venv venv

# activate it
source venv/bin/activate

# Update everything
pip install --upgrade pip

# Install AWS specific libraries
pip install AWSIoTPythonSDK
```

Installing the example application

The example application is available from the Raspberry Pi Ltd GitHub repo. Use the following command to clone the code to the Raspberry Pi 5 into a pre-existing folder named `Pi5AWSExample` .

```
Code
cd Pi5AWSExample
git clone https://github.com/raspberrypi/rpi-aws-examples.git
```

In a later section, we will be editing the example Python code to customise it with the users thing name, security keys and topic names.

Additional software references

Technical support for Raspberry Pi Ltd devices can be found on the Raspberry Pi Ltd forums at https://www.raspberrypi.com/forums

Set up device hardware

Full instructions for setting up a Raspberry Pi 5 device can be found on the Raspberry Pi Ltd website.

https://www.raspberrypi.com/documentation/computers/getting-started.html

Setup your AWS account and permissions

If you do not have an existing AWS account and user, refer to the online AWS documentation at Set up your AWS Account To get started, follow the steps outlined in the sections below:

- Sign up for an AWS account
- Create an administrative user
- Open the AWS IoT console

Pay special attention to the Notes.

Some explanation on AWS terminology

Getting to grips with Amazon Web Services (AWS) terminology and permissions is vital to ensure everything works as expected. The following section explains the various concepts and naming schemes involved.

ARNs, Permissions and Roles

ARN

An AWS ARN (Amazon Resource Name) is a unique identifier for an AWS resource, allowing for unambiguous reference across the entire AWS ecosystem. It's a string that specifies a resource, such as an EC2 instance, S3 bucket, or IAM user, enabling you to manage and control access to these resources within AWS services, like IAM policies, API calls, etc.

Policies

> In AWS, a policy is a document/object that defines permissions for an entity (user, group, or role) or resource. It dictates what actions an entity can perform on which AWS resources. These policies are crucial for managing access to AWS services and ensuring security within the AWS environment.
>
> — Amazon

How it works

Policies control access to AWS resources by specifying which actions are allowed or denied.

Policies are typically written in JSON format and consist of statements that define the effect (allow or deny), actions, resources, and conditions for permissions.

There are different types of policies, including:

- Identity-based policies: Attached to users, groups, or roles, controlling what those identities can do.
- Resource-based policies: Attached to a specific resource, like an S3 bucket, controlling who can access that resource.

Permissions

> In AWS, a permission defines what actions a user, role, or service can perform on specific AWS resources. It's a grant of access, allowing or denying specific operations based on defined rules.
>
> — Amazon

How it works

Permissions determine the specific actions that can be performed on AWS resources. They are defined using IAM policies, which specify which actions are allowed or denied. IAM (Identity and Access Management) is the AWS service responsible for managing permissions.

Roles

> An AWS IAM role is an AWS identity that you create to grant permissions to trusted entities to access AWS resources, without needing to create separate credentials for each entity. It's like a set of permissions that can be assumed by users, applications, or services, to temporarily access specific resources within your AWS environment.

— Amazon

How it works

Instead of assigning long-term credentials to users or applications, you create a role with specific permissions and allow trusted entities to “assume” that role temporarily. Unlike Users, Roles do not have passwords or access keys associated with them. Instead, roles provide temporary security credentials to whoever has the ability to assume that role.

Things

An AWS Thing is a device or a virtual entity that is registered and managed within the IoT platform. It’s a way to model and track physical devices, such as sensors or actuators, or even abstract entities, within the AWS ecosystem. Each Thing has associated certificates and policies that define its interactions with AWS IoT services.

So a Raspberry Pi SBC (or Pico) could be an AWS IoT Thing. You can use the AWS console to give a Thing a name and generate a unique certificate for that Thing. These certificates need to be downloaded onto the Thing as they uniquely identify it.

Note

When you generate a certificate set on the console you must download it straight away - you cannot go back and download them later

When the certificate is generated, the AWS console also provides a set of 4 keys. Private, public and two Amazon Root certificates which are Certificate Authority blocks. See https://docs.aws.amazon.com/privateca/latest/userguide/PcaTerms.html

During the creation of a Thing, you also assign a Policy to it, which is actually attached to the certificate. This policy defines what the Thing is able to do (see above).

Create resources in AWS IoT

Creating a Thing object

Note

All the following operations can be carried out on the Raspberry Pi 5 itself

Rather than outline how to do this here, please refer to the AWS documentation on creating a Thing object. https://docs.aws.amazon.com/iot/latest/developerguide/create-iot-resources.html#create-aws-thing.

In short, create a new thing with the name `Pi5AWSExample` , select no shadow, select auto-generate a new certificate for the device. You do not need to attach a policy.

Warning

During the creation process you will be asked to download the certificate and keys. This is the only time this option is available so make sure all the keys are downloaded to the Raspberry Pi 5. You will need to add these keys to the example application

Provision the device with credentials

As this is simple example software, there are a few things you need to manually do to get it up and running. The main thing is to add the credentials (keys) that were downloaded in the previous section to the example code. These items are hardcoded in to the example so need to be updated to the user specific keys.

1 Copy all the keys downloaded to the location where the example application was cloned.
These can have rather long filenames consisting of seemingly random letters.

2 Edit the program, and fill in the following entries with the updated information.

Entry	Content
THING_NAME	Pi5AWSExample
PRIVATE_KEY_PATH	xxxxx-private.pem.key
CERTIFICATE_PATH	xxxxx-certificate.pem.crt
IOT_ENDPOINT	The endpoint is found in Test Suite data (See Run the example code section)

xxxxx represents the lengthy hexadecimal filename of the downloaded keys.

Create a device advisor test suite

From the AWS IoT console, select Device Advisor/Test suites, then select Create test suite. Choose AWS IoT Core qualification test suite for the test suite type, and MQTT 3.1.1 as the Protocol, and click next. You will now be presented with a page Create test suite, which contains the tests needed for the Device Advisor qualification. Click Test suite properties and give the test suite an appropriate name, for our example, Device Advisor Suite. Click next. You now need to select a role; this is how you give permissions to AWS to use your certificates etc during the Device Advisor testing. For our example we will create a new role, so select Create new role.

Create a role

We need to specify the access available to Device Advisor.

For the Connect option, enter the Thing name we specified when we created the Raspberry Pi 5 Thing, for our example, this is `Pi5AWSExample` .

For Publish, Subscribe, Receive and RetainPublish, we will enter a wildcard topic, `Pi5AWSExample*` , which allows any topics starting with `Pi5AWSExample` to be valid.

Build the example code

As the software is Python, there is no need for any compilation.

Run the example code

To run, we must start the Test Suite running and then run the example code on the Raspberry Pi 5, which then communicates with the Test Suite.

From the AWS IoT console, select Device Advisor/Test Suites, then select your newly created Test Suite by clicking on its name. On the subsequent screen, select Actions and Run test suite.

Select the Thing you want to test, in our case Pi5AWSExample. In the Test endpoint section, select Account-level endpoint.

Important

You need to copy the endpoint information from the Test endpoint into the Python example application, see the IOT_ENDPOINT configuration item in the example. This tells the example application where to find the Device Advisor tests. As this endpoint is fixed to the Test suite, it does not change, so you only need to do this once.

Once the example code is updated with the endpoint, you can run the Test Suite by clicking Run test. This will start up the Devise Advisor test suite. You now need to start the example code to communicate with it.

Code

```
# cd to the location of your cloned example repository.
cd Pi5AWSExample/rpi-aws-examples
# run the example
python3 Pi5AWSExample
```

If all is working correctly, after a few minutes the Devisor Advisor test suite will display the results of the test.

Verify messages in AWS IoT Core

You can examine the logging done during the test by selecting the Test case log links next to each part of the test suite. These are on the qualification results page displayed while the test is running. These logs display all the messages that were exchanged between the Raspberry Pi 5 and AWS, and can be very useful if you find that some tests are not passing.

Debugging

As the Raspberry Pi 5 with Raspberry Pi OS is a full Linux-based system, rather than a dedicated and targeted device, debugging is considerably easier. You have all the features available and easily accessible, simply by working with the Raspberry Pi OS desktop.

Device console output is simply viewed in the terminal windows used to start the example software, and it is easy to modify the example code in place (for example, to add debug print statements) to help with debugging. All the standard Linux logging is available, and you can use, for example, VSCodes Python debugger, to debug the Python example code.

Applications developed in C/C++ (not shown here) can also use IDE's like VSCode to do development and debugging on device.

Troubleshooting

Incorrect permissions are almost always the reason why programs may not work as intended or at all. Double check roles, permissions, polices and device names.

Please use the Raspberry Pi Ltd forums https://forums.raspberrypi.com for community technical support, this is usually the fastest way to get help. The Raspberry Pi Ltd applications team are also available on applications@raspberrypi.com.

Conclusions

This document describes how to get a simple AWS application up and running to pass the Device Advisor certification tests. It shows that Raspberry Pi Ltd SBCs are a great way of providing IoT compute power that can easily leverage AWS to provide cloud-based back-end services.

The example application is extremely basic, and is there simply to show the principles of connection, publishing and subscribing. The user will certainly need to develop their own applications targeted to their specific use case. Although the example applications is Python, AWS also provide a C/C++ SDK and other language bindings for application development, and the user should consider which is the most appropriate.

Although the testing was carried out on the Raspberry Pi 5, there is no reason that earlier models running the same system would not work in the same way, as they all run the same operating system, Raspberry Pi OS. End users should choose the device most appropriate to their application.

Contact Details for more information

Please contact applications@raspberrypi.com if you have any queries about this whitepaper.

Web: www.raspberrypi.com

Raspberry Pi

Raspberry Pi Ltd