Microsoft Azure SQL Edge with Lenovo ThinkSystem SE350 Solution Guide

Last update: **01 December 2021**

Version 1.1

Highlights the benefits of ThinkSystem SE350 Edge Server

Presents a use case for SQL Server on Lenovo Edge devices

Provides deployment information and best practices

Includes list of all software package download URLs

In partnership with

David West

Vinay Kulkarni

Table of Contents

1 Introduction

The Internet of Things (IoT) is the name coined for the billions of physical devices around the world that are connected to the internet, all collecting and sharing data. With cheap computer chips and the spread of wireless networks it's possible to turn any small or large thing into an IoT device. By adding sensors to these objects and connecting them to the internet converts them into intelligence devices. Data generated by these edge devices can be collected and mined to make customer lives better or improve profits for businesses. Data generated needed to be uploaded to the public cloud for processing which compromised security and increased latency. Azure SQL Edge is a very small footprint, robust IoT database product from Microsoft that can run on these edge devices to process the data locally for increased security and reduced latency.

The combination of Microsoft Azure SQL Edge and Lenovo's innovative ThinkSystem SE350 Edge Server provides the ideal edge gateway solution for processing data aggregated from scores of IoT devices. Azure SQL Edge is a fully containerized solution compatible with most docker container engines. It can run on any Intel or ARM64 device and includes native support for data streaming (the same engine that powers Azure Stream Analytics). It has many security features like data encryption, classification, and access controls. Azure SQL Edge also has Time Series Processing and ML inferencing capabilities. ML Models can be trained on premises and deployed to the edge where inferencing can be done on data that is being collected.

This white paper walks you through the steps required to set up and run Azure SQL Edge on the SE350. It also provides some performance data with Azure SQL Edge running on the SE350.

This solution can also be implemented on ThinkAgile MX1020 Integrated system or ThinkAgile MX1021 certified nodes for high availability.

2 Business value

Microsoft Azure SQL Edge extends the performance and security of the very popular Microsoft SQL Server engine to the intelligent edge. It is optimized for IoT gateways and devices, is available in an easy to deploy modern container format and has a very small footprint of around 500MB. Azure SQL Edge provides the RDBMS features of a full version of Microsoft SQL Server along with real-time insights, built-in streaming, storage, and support for AI. Azure SQL Edge helps customers with their application modernization journey so they can develop their application once and deploy anywhere across the edge, their datacenter and Azure.

Azure SQL Edge has simplified pricing that is well suited for IoT deployments. Azure SQL Edge follows a per device pricing model as listed below in table number 1. Purchasing one unit of Azure SQL Edge provides usage rights to run Azure SQL Edge on one device.

Product	Pay as you go	1 Year Reserved	3 Year Reserved
Azure SQL Edge	$10/device/month	$100/device/year	$60/device/year

Table 1. Azure SQL Edge pricing

Billing starts when an Azure SQL Edge is deployed to devices, irrespective of whether the SQL process is running/failed/stopped.

3 Architectural overview

Azure SQL Edge is a compact, containerized version of SQL Server for Linux that is designed to run on smaller edge devices at remote locations. The system can run disconnected from Azure or a data center and then reconnect later to sync data.

Ideal use cases include using SQL Edge as a local database for IoT devices in retail or similar environments.

With the Azure connected model, an Azure IoT Hub is used to connect to an Azure defined IoT Edge device. Then the Azure SQL Edge module, which is a container, is deployed to the edge device.

The following diagram, complements of Microsoft, provides a high-level view of it.

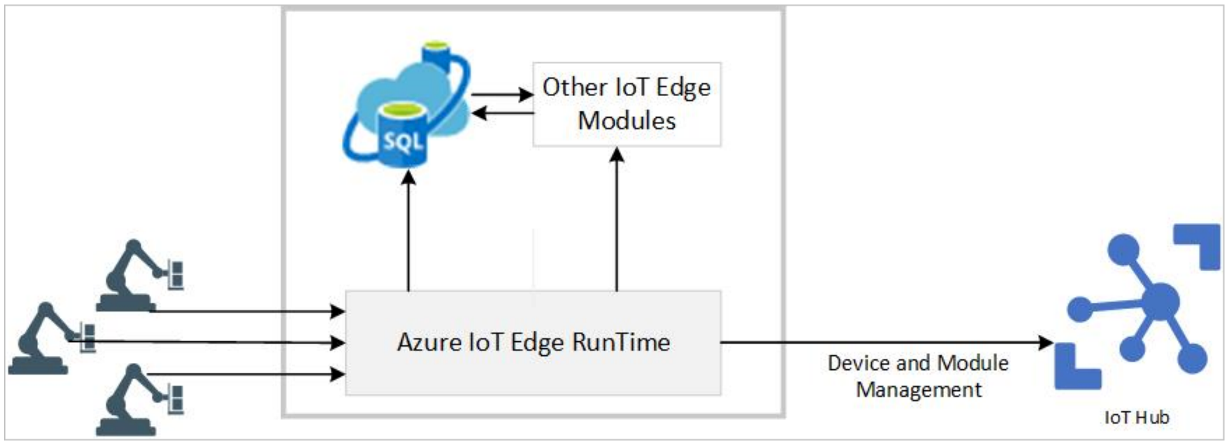


Figure 1 Overview of Azure SQL Edge architecture. Drawing credit, Microsoft

More information on the SQL Edge architecture can be found here:
https://docs.microsoft.com/en-us/azure/azure-sql-edge/overview

4 Deployment prerequisites

There are several concepts and pre-requisites to be aware of before deploying the solution. There are two deployment options. The more complex one is the Azure connected method, while the disconnected Docker container approach is rather straight forward.

For the connected option there is a requirement for an IoT Hub in Azure, along with a corresponding connected IoT Edge device which is where the SQL Edge container is deployed to. The IoT Edge device is a Linux system, which can be any size system or even a virtual machine running on an edge system.

Microsoft's documentation and packages are based on Ubuntu or Raspberry PI platforms. These 2 are the only Tier 1 supported platforms for the IoT Edge device (see the link below). There are several Tier 2 systems which should work fine but are not fully tested, supported, or as well documented by Microsoft. Lenovo's preference for this project was to use Red Hat Enterprise Linux (RHEL) for the IoT Edge device OS and the only Red Hat version on the Tier 2 list is RHEL 7.x. As a result, the deployment tested and detailed in this guide is based on RHEL 7.9 with specific RPM package links and dependency information for this OS. Setting this up on a Tier 2 platform requires more effort to find the right combination of supported packages and dependencies, which we have already worked through for you and provide in the sections below.

More information on the supported platforms for the IoT Edge device is available at:
https://docs.microsoft.com/en-us/azure/iot-edge/support?view=iotedge-2020-11#operating-systems

Specific package pre-requisites are covered in each component section that follows. Since most of the packages are not included in the usual Red Hat repositories, this guide includes the URLs to download the RPMs via wget and the commands to run the installations.

The deployment steps below assume access to an Azure account and an edge device has been provisioned running RHEL 7.x, with internet connectivity.

5 IoT Edge device setup

The IoT Edge devices are based on the Moby container engine, which is essentially open-source Docker. This is the container engine Microsoft requires. The packages below are the most recent at the time of writing. As time goes by, these may need updating, by looking in the same packages.microsoft location.

For each of the packages below, use the wget command to download the RPM packages.

Example: Sudo wget https://url_to_package.rpm

5.1 Create an IoT Hub in Azure

Follow the steps in this link to setup an IoT Hub in Azure, it is a simple wizard driven setup. The hub will be used later to register, connect, and manage the IoT Edge device.

https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-create-through-portal

5.2 Install Moby Engine prerequisites on edge device

There are 3 packages that must be installed as Moby engine dependencies. They must be installed **in the order provided below** to satisfy the dependencies. Use sudo wget <url> to download these.

Download links for the packages:

1. Container-selinux
   http://mirror.centos.org/centos/7/extras/x86_64/Packages/container-selinux-2.119.2-1.911c772.el7_8.noarch.rpm
2. Moby-runc
   https://packages.microsoft.com/centos/7/prod/moby-runc-1.0.0~rc95%2Bazure-1.x86_64.rpm
3. Moby-containerd
   https://packages.microsoft.com/centos/7/prod/moby-containerd-1.4.8%2Bazure-1.el7.x86_64.rpm

Install each of these one at a time, in order, as there is a y/n prompt on each one. Change to the directory where the packages were downloaded and install each one with the following yum command.

Sudo yum install <package name.rpm>

5.3 Install Moby Engine

After the 3 dependency packages are installed, proceed with downloading and installing the Moby container CLI and engine. These are two separate packages to install.

Download links for Moby Engine and CLI:

Moby-cli
https://packages.microsoft.com/centos/7/prod/moby-cli-20.10.7%2Bazure-1.x86_64.rpm

Moby-engine
https://packages.microsoft.com/centos/7/prod/moby-engine-20.10.7%2Bazure-1.el7.x86_64.rpm

Install each of these, one at a time, with the following command.

Sudo yum install <package name.rpm>

5.4 Install the IoT Edge components

After the Moby engine is installed, the two IoT Edge components can be installed. These must be installed **in the order listed below**, to meet dependencies.

Download links for the two IoT Edge files:

https://github.com/Azure/azure-iotedge/releases/download/1.1.4/libiothsm-std_1.1.4-1.el7.x86_64.rpm

https://github.com/Azure/azure-iotedge/releases/download/1.1.4/iotedge-1.1.4-1.el7.x86_64.rpm

Install each of these, one at a time, with the following command.

Note: These use the rpm command below, not the yum installer as previously used.

Sudo rpm -Uhv <package name.rpm>

5.5 Register the device with Azure IoTHub

The Edge device needs to be registered with an IoTHub to enable it as a connected Edge device. Follow the steps in the below link to register the device. For test or proof of concept configurations, the symmetrical key method is the simplest approach.

https://docs.microsoft.com/en-us/azure/iot-edge/how-to-register-device?view=iotedge-2020-11&tabs=azure-portal

After the device is registered, open the object and copy the connection strings. These are used to configure the IoTEdge connection in the next step.

5.6 Configure IoT Hub connection

The IoT Edge device needs connection strings configured to allow it to connect with the IoT Hub. Open the configuration file located at:

/etc/iotedge/config.yaml

Using vi or a similar editor, find the below section and add the connection string where indicated, then save the file. Note that this is a read-only file, so to save it in vi editor, use :wq! or the equivalent if using another editor.

```
# Manual provisioning with an IoT Hub connection string (SharedAccessKey authentication only)
provisioning:
    source: "manual"
    device_connection_string: "<ADD DEVICE CONNECTION STRING HERE>"
    dynamic_reprovisioning: false
```

Complete the remaining steps below to apply changes and verify the install.

1. To apply the new configuration, restart the IoTEdge service with the following command.

   ```
   Sudo systemctl restart iotedge
   ```

2. Use this command to verify that the service is running.

   ```
   Sudo systemctl status iotedge
   ```
   , shows the status of the service

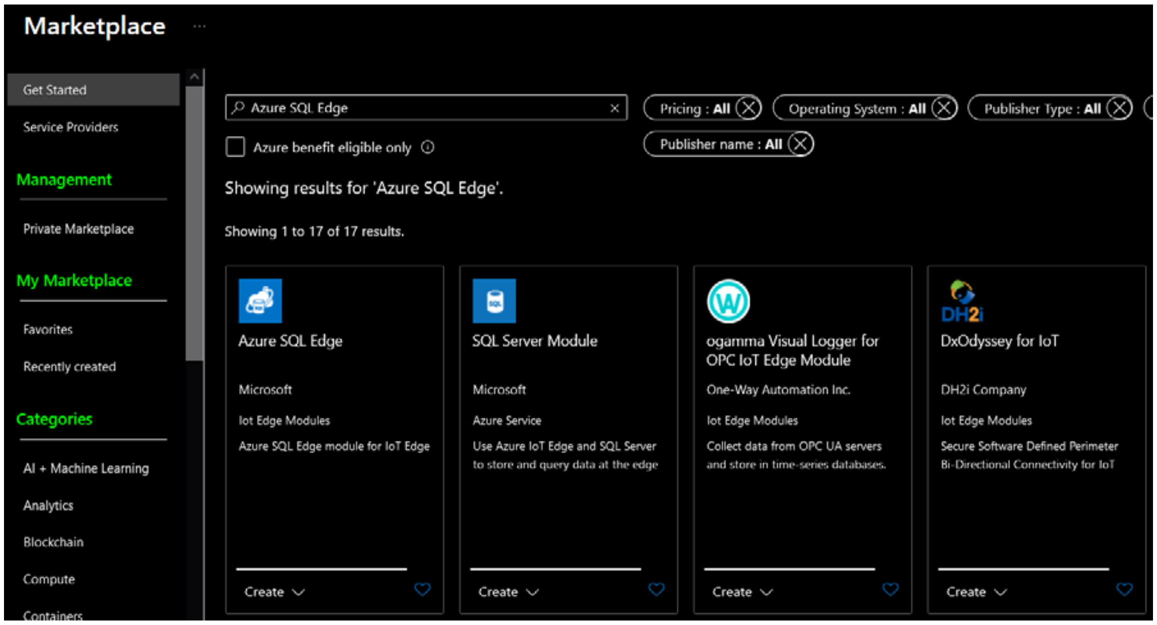
It should show a status of active (running)

View the IoT Edge device in the Azure portal within the IoT Hub device. It should show running, however until the SQL Edge module is deployed there may be some errors showing. This is normal, proceed with the SQL Edge deployment below, and afterwards it will show connected and online.

6 Azure SQL Edge - Azure IoT Edge method

Now that the IoT Edge device is configured and connected to the Azure portal, the next step is to deploy SQL Edge to it from the Azure Marketplace. The Marketplace is an online listing of applications and services that are designed and optimized for Azure. Follow the steps below to deploy the SQL Edge container module.

1. Search within Azure Marketplace for the Azure SQL Edge offering.



Showing results for 'Azure SQL Edge'.

Showing 1 to 17 of 17 results.

Azure SQL Edge — Microsoft — Iot Edge Modules — Azure SQL Edge module for IoT Edge

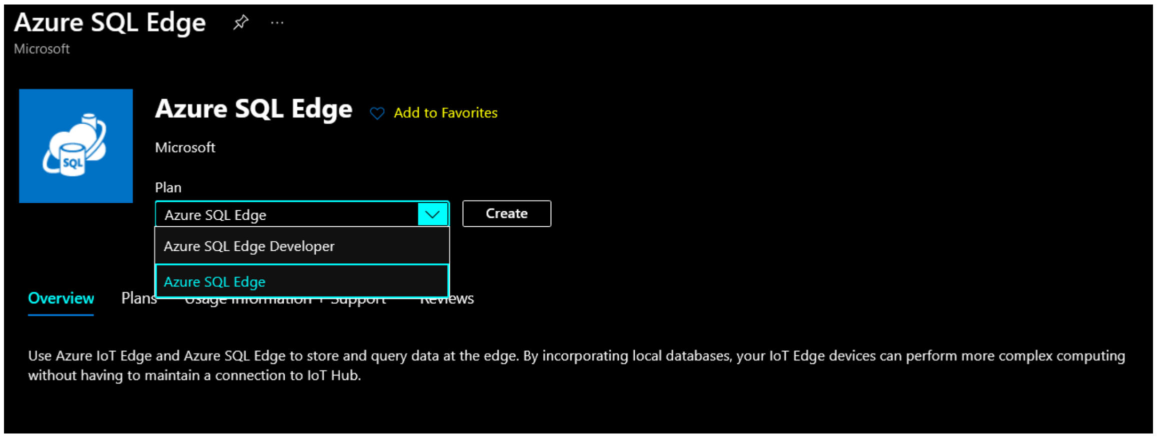
SQL Server Module — Microsoft — Azure Service — Use Azure IoT Edge and SQL Server to store and query data at the edge

ogamma Visual Logger for OPC IoT Edge Module — One-Way Automation Inc. — Iot Edge Modules — Collect data from OPC UA servers and store in time-series databases.

DxOdyssey for IoT — DH2i Company — Iot Edge Modules — Secure Software Defined Perimeter Bi-Directional Connectivity for IoT

Figure 2 Azure Marketplace

2. From the list on the Azure SQL Edge page, choose developer or standard, and click Create

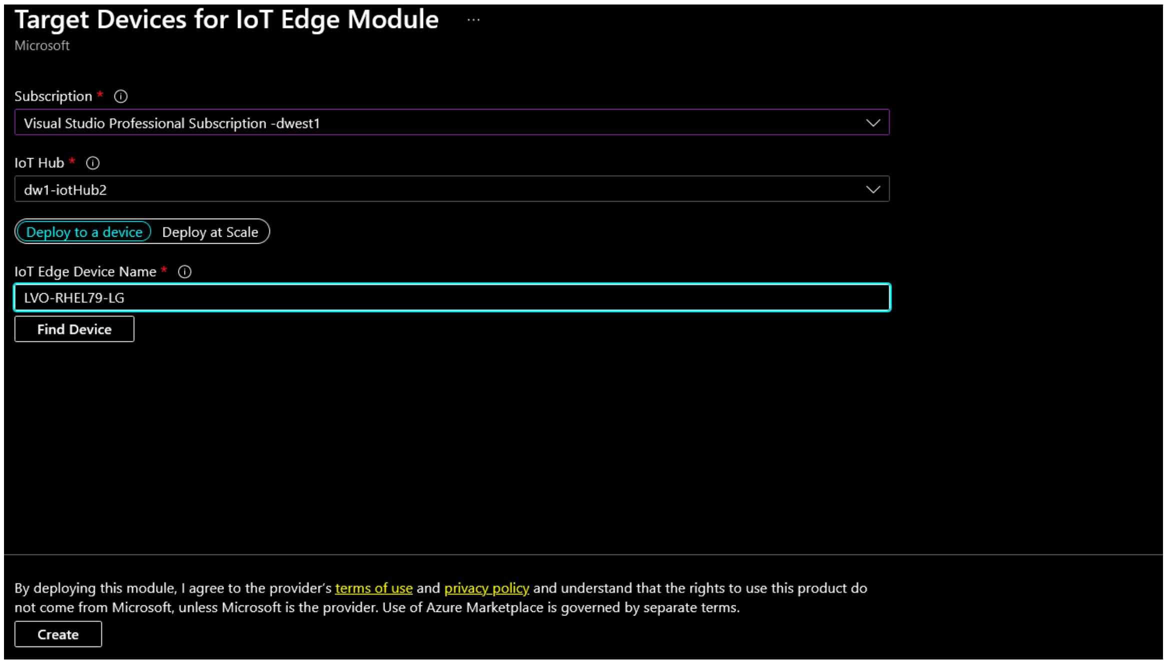


Use Azure IoT Edge and Azure SQL Edge to store and query data at the edge. By incorporating local databases, your IoT Edge devices can perform more complex computing without having to maintain a connection to IoT Hub.

Figure 3 Create the SQL Edge module

3. On the Target Devices page, provide the following information where prompted:
   a. Azure subscription number
   b. IoT Hub name
   c. IoT Edge device name

Then click Create, at the bottom of the page.



Target Devices for IoT Edge Module
Microsoft

Subscription * : Visual Studio Professional Subscription -dwest1

IoT Hub * : dw1-iotHub2

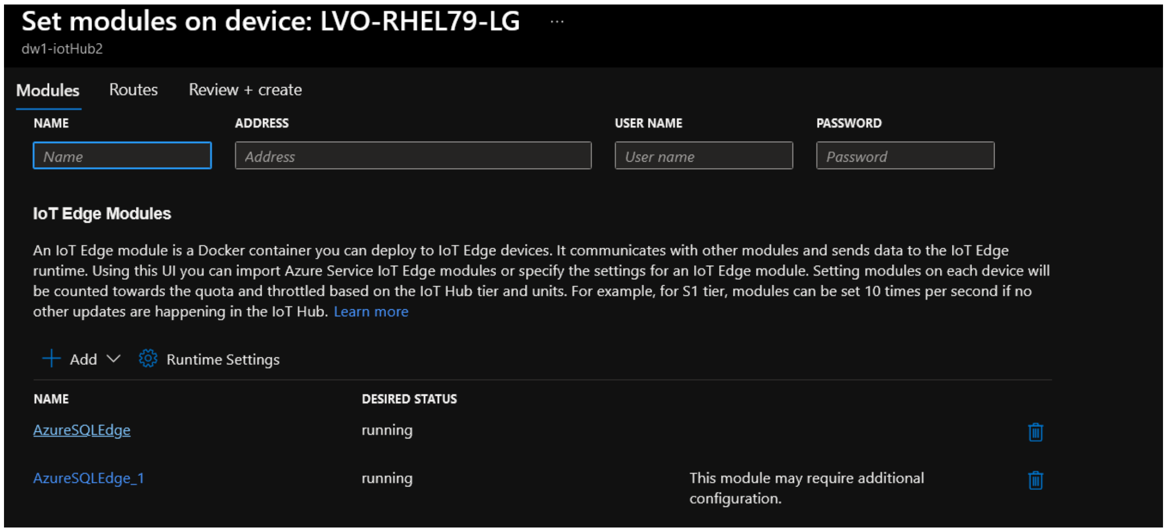
Deploy to a device | Deploy at Scale

IoT Edge Device Name * : LVO-RHEL79-LG

By deploying this module, I agree to the provider's terms of use and privacy policy and understand that the rights to use this product do not come from Microsoft, unless Microsoft is the provider. Use of Azure Marketplace is governed by separate terms.

Figure 4 Specify the target device

4. On the Set Modules page, click on the Azure SQL Edge module. The default name is set to AzureSQLEdge, which is fine, but it can also be changed here.



Set modules on device: LVO-RHEL79-LG
dw1-iotHub2

NAME	ADDRESS	USER NAME	PASSWORD
Name	Address	User name	Password

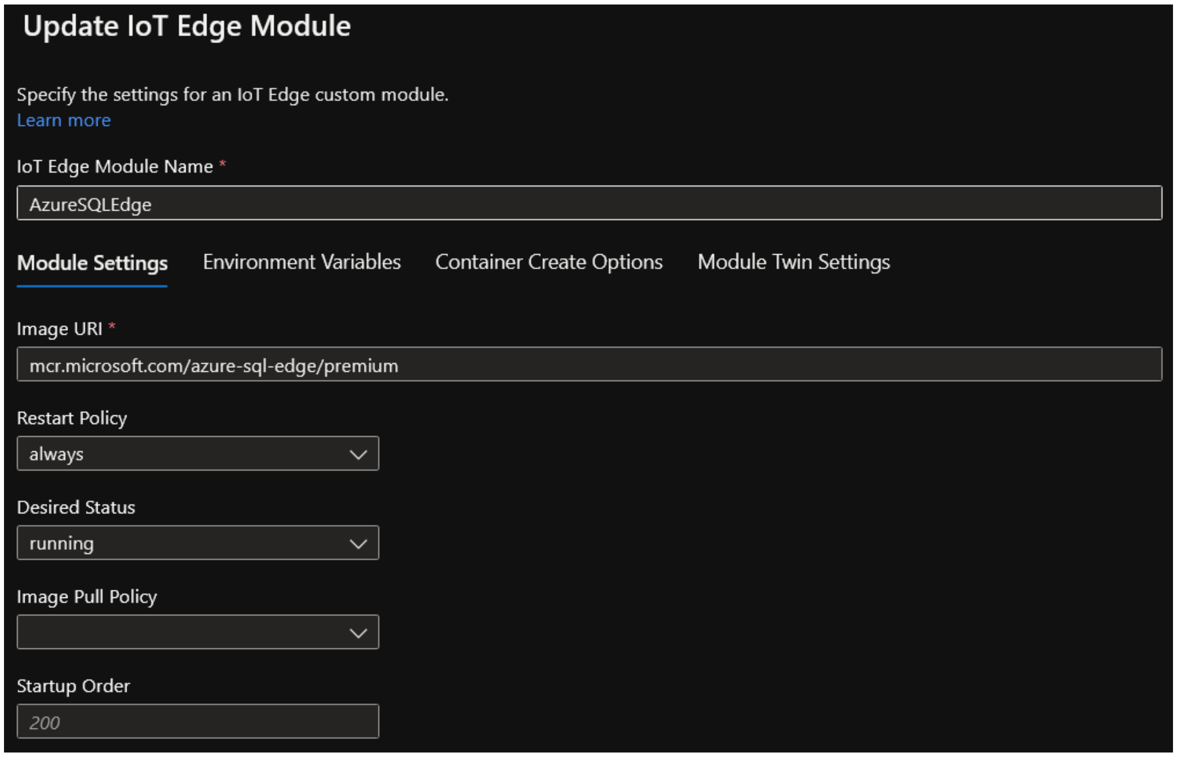
IoT Edge Modules

An IoT Edge module is a Docker container you can deploy to IoT Edge devices. It communicates with other modules and sends data to the IoT Edge runtime. Using this UI you can import Azure Service IoT Edge modules or specify the settings for an IoT Edge module. Setting modules on each device will be counted towards the quota and throttled based on the IoT Hub tier and units. For example, for S1 tier, modules can be set 10 times per second if no other updates are happening in the IoT Hub. Learn more

NAME	DESIRED STATUS	
AzureSQLEdge	running	
AzureSQLEdge_1	running	This module may require additional configuration.

Figure 5 Select module to deploy

5. This opens the Module Settings tab of the Update IoT Edge Module page. Set the values for Module Name, Restart Policy and Desired Status per your requirements.


Update IoT Edge Module

Specify the settings for an IoT Edge custom module.
Learn more

IoT Edge Module Name *
AzureSQLEdge

Image URI *
mcr.microsoft.com/azure-sql-edge/premium

Restart Policy
always

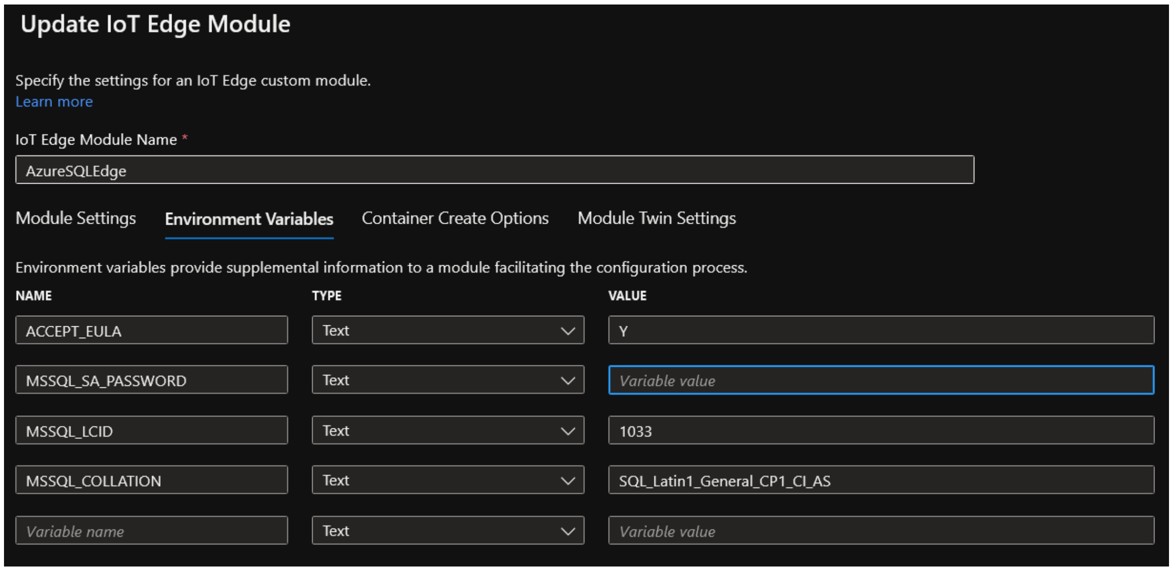
Desired Status
running

Image Pull Policy

Startup Order
200

Figure 6 Set module parameters and policy

6. On the Environment Variables section, specify the SQL SA password, language code (leave default 1033 for English) and collation options.


Update IoT Edge Module

Specify the settings for an IoT Edge custom module.
Learn more

IoT Edge Module Name *
AzureSQLEdge

Environment variables provide supplemental information to a module facilitating the configuration process.

NAME	TYPE	VALUE
ACCEPT_EULA	Text	Y
MSSQL_SA_PASSWORD	Text	Variable value
MSSQL_LCID	Text	1033
MSSQL_COLLATION	Text	SQL_Latin1_General_CP1_CI_AS
Variable name	Text	Variable value

Figure 7 Set SQL password and language

7. For the Container Create Options section, there is a JSON file displayed. Here the module's volume bindings and SQL port number can be edited. Recommend leaving it at the default of 1433.