

IBAZ: Quickstart

Summary

IoT Bridge for Azure (IBAZ) is an application that connects to an MQTT Server and consumes MQTT Sparkplug messages from Edge devices.

These messages must be formatted as Sparkplug Templates which are defined in the [Sparkplug Specification](#). The Templates are used to create the Models in Azure Digital Twins automatically with no additional coding or configuration. Then multiple instances of these Templates generate the Assets and start to populate with real time data sent on change only, thus significantly reducing the amount of data being sent to the cloud.

For further details on Azure Digital Twins terminology, refer to the [Azure Digital Twins documentation](#). For further details on Eclipse Sparkplug, refer to the [Eclipse Sparkplug resources](#).

This quickstart document covers how IoT Bridge can be used to consume MQTT Sparkplug data and create and update Azure Digital Twins Asset Models and Assets. This will show how to configure IoT Bridge as well as show how to use Inductive Automation's Ignition platform along with Cirrus Link's MQTT modules to publish device data to an MQTT Server. This data will ultimately be consumed by IoT Bridge to create and update the Azure Digital Twins components.



This tutorial will use the MQTT Distributor implementation however the IBAZ does work with any MQTT v3.1.1 compliant MQTT Server.



Ignition in conjunction with Cirrus Link's MQTT Transmission module converts Ignition User Defined Types (UDTs) to Sparkplug Templates. This is done automatically by the MQTT Transmission module. So, much of this document will refer to UDTs rather than Sparkplug Templates since that is what they are in Ignition.

More information on Inductive Automation's Ignition platform can be found [here](#). Additional information on Cirrus Link's MQTT Transmission module can be found [here](#).

Step 1: IBAZ Installation

- Complete the [IBAZ Installation](#) process
- This will install the Virtual machine, Disk, Network interface and Public IP address resources

Step 2: Create an Azure Digital Twins instance

- If needed create following the Microsoft [Set up an Azure Digital Twins instance and authentication \(portal\)](#) instructions
- Collect the Host name as this is used in the [IBAZ: Configuration](#)

Step 3: Configure the Azure Digital Twins instance

- [Assign the Azure Digital Twins Data Owner role](#) to your Azure Active Directory app registration identity and an additional Azure user(s)
- [Add a system-managed identity](#) to your instance

Step 4: Create an event Hubs Namespace

- If needed create following the Microsoft [Create an event hub using Azure portal](#) instructions

Step 5: Create an Azure Data Explorer Cluster

- If needed create following the Microsoft [Create an Azure Data Explorer cluster and database](#) instructions

Step 6: Configure the Azure Data Explorer Cluster

- Configure the [database permissions](#) to add your Azure Digital Twins Instance with a Database Admin role
- Add additional Viewer role(s) to be able to query data

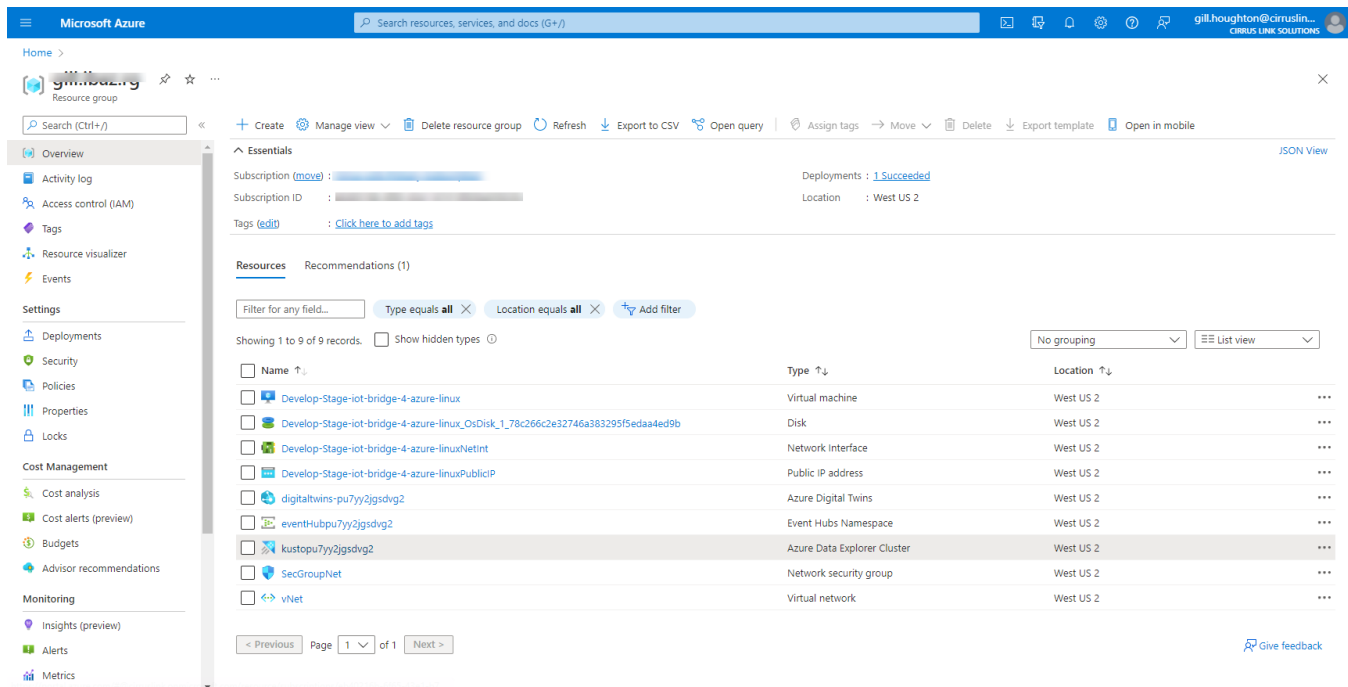
Step 7: Configure the Azure Digital Twins instance data history connection

- [Setup a data history connection](#)

Step 8: Create an Azure Active Directory App Registration

- If needed create following the Microsoft [Create an app registration to use with Azure Digital Twins](#) instructions
- Collect the client ID, tenant ID and Client secret Value as these are used in the [IBAZ: Configuration](#)

In our example below we have all the resources in the same Resource group:



The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes the Microsoft Azure logo, a search bar, and user information. The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, Resource visualizer, Events, Settings, Deployments, Security, Policies, Properties, Locks, Cost Management, Cost analysis, Cost alerts (preview), Budgets, Advisor recommendations, Monitoring, Insights (preview), Alerts, and Metrics. The main content area displays the 'Essentials' section for a resource group, showing subscription details, deployment status (1 Succeeded), and a list of resources. The resources table includes columns for Name, Type, and Location. The resources listed are: Develop-Stage-iot-bridge-4-azure-linux (Virtual machine), Develop-Stage-iot-bridge-4-azure-linux_OsDisk_1_78c266c2e32746a383295f5edaa4ed9b (Disk), Develop-Stage-iot-bridge-4-azure-linuxNetint (Network Interface), Develop-Stage-iot-bridge-4-azure-linuxPublicIP (Public IP address), digitaltwins-pu7yy2jgsdvg2 (Azure Digital Twins), eventHubpu7yy2jgsdvg2 (Event Hubs Namespace), kustopu7yy2jgsdvg2 (Azure Data Explorer Cluster), SecGroupNet (Network security group), and vNet (Virtual network). The 'kustopu7yy2jgsdvg2' resource is highlighted.

Step 9: Install Ignition and MQTT Modules

For this setup, we'll install Inductive Automation's Ignition platform along with the MQTT Distributor (MQTT Server) and MQTT Transmission (Edge device) modules from Cirrus Link.

Installation of Ignition is very straightforward following the instructions in the [Installing and Upgrading Ignition](#) guide.

With Ignition installed, the Cirrus Link modules must be installed as a plugin to Ignition. Follow the instructions in our [Module Installation](#) guide.

Step 10: IBAZ Configuration

Update your configuration properties in the IBAZ configuration file located on the VM at `/opt/ibaz/conf/ibaz.properties`. Instructions for accessing the VM can be found [here](#).

Follow our [IBAZ configuration](#) guide for details.



With the default MQTT Distributor configuration use the following properties:

```
mqtt_server_url = tcp://youripaddress of installed Ignition instance:1883
```

```
mqtt_username = admin
```

```
mqtt_password = changeme
```

Now the IBAZ service can be restarted to pick up the new configuration. Do so by running the following command.

```
sudo systemctl restart ibaz
```

At this point, IBAZ should connect to MQTT Distributor and be ready to receive MQTT Sparkplug messages. Verify by running the following command.

```
tail -f /opt/ibaz/log/wrapper.log
```

After doing so, you should see something similar to what is shown below.

Note the last line is 'MQTT Client connected to ...'. and this indicates that we have successfully configured IBAZ and properly provisioned MQTT Distributor.

```
ubuntu@ibaz: ~$
Aug 25, 2022 11:09:43 PM org.rzo.yajsw.os.posix.PosixService getPid
INFO: wrapper pid file: /run/wrapper.ibaz.pid
Service ibaz started
ubuntu@ibaz:~$ tail -f /opt/ibaz/log/wrapper.log
INFO|28265/0|[22-08-25 23:09:47|23:09:47.275 [main] INFO c.c.i.a.model.DigitalTwinInfoModel - Creating 1 EdgeNode Info Digital Twin Models
INFO|28265/0|[22-08-25 23:09:47|23:09:47.307 [main] INFO c.c.i.common.info.Diagnostics - Creating or updating Bridge Model instances
INFO|28265/0|[22-08-25 23:09:47|23:09:47.307 [main] INFO c.c.i.a.model.DigitalTwinInfoModel - Creating IBAZ Info Digital Twin: IBAZ Metrics IBAZ_Info_Gill_IBAZ_Instance
INFO|28265/0|[22-08-25 23:09:47|23:09:47.365 [main] INFO c.c.i.common.info.Diagnostics - Creating or updating MQTT Client Model instances
INFO|28265/0|[22-08-25 23:09:47|23:09:47.365 [main] INFO c.c.i.a.model.MqttClientInfoModel - Getting MQTT Client Info instance
INFO|28265/0|[22-08-25 23:09:47|23:09:47.365 [main] INFO c.c.i.a.model.DigitalTwinInfoModel - Creating MQTTClient Info Digital Twin: IBAZ Metrics MQTT_Client_Info_Gill_IBAZ_Instance
INFO|28265/0|[22-08-25 23:09:47|23:09:47.430 [main] INFO c.c.i.common.AbstractSparkplugClient - MQTT Client IBAZ-b410439b-dfd6-4c9d successfully started
INFO|28265/0|[22-08-25 23:09:47|main method terminated
INFO|28265/0|[22-08-25 23:09:47|exit on main terminate -1
INFO|28265/0|[22-08-25 23:09:47|23:09:47.431 [Thread-9] INFO org.eclipse.tahu.mqtt.TahuClient - IBAZ-b410439b-dfd6-4c9d: Creating the MQTT Client to tcp://52.91.136.214:1883 on thread Thread-9
INFO|28265/0|[22-08-25 23:09:47|23:09:47.921 [MQTT Call: IBAZ-b410439b-dfd6-4c9d] INFO org.eclipse.tahu.mqtt.TahuClient - IBAZ-b410439b-dfd6-4c9d: connect with retry succeeded
INFO|28265/0|[22-08-25 23:09:47|23:09:47.921 [MQTT Call: IBAZ-b410439b-dfd6-4c9d] INFO org.eclipse.tahu.mqtt.TahuClient - IBAZ-b410439b-dfd6-4c9d: Connected to tcp://52.91.136.214:1883
INFO|28265/0|[22-08-25 23:09:47|23:09:47.921 [MQTT Call: IBAZ-b410439b-dfd6-4c9d] INFO c.c.i.common.AbstractMessageCallback - Connection complete to Local MQTT Server
INFO|28265/0|[22-08-25 23:09:48|23:09:48.446 [Thread-9] INFO org.eclipse.tahu.mqtt.TahuClient - IBAZ-b410439b-dfd6-4c9d: MQTT Client connected to tcp://52.91.136.214:1883 on thread Thread-9
```

Now we can configure our Ignition MQTT Transmission module to connect to the MQTT Distributor server in order to publish data into Azure. We will also import some UDT Definitions and UDT Instances that will be mapped to Azure Models and Digital Twins.

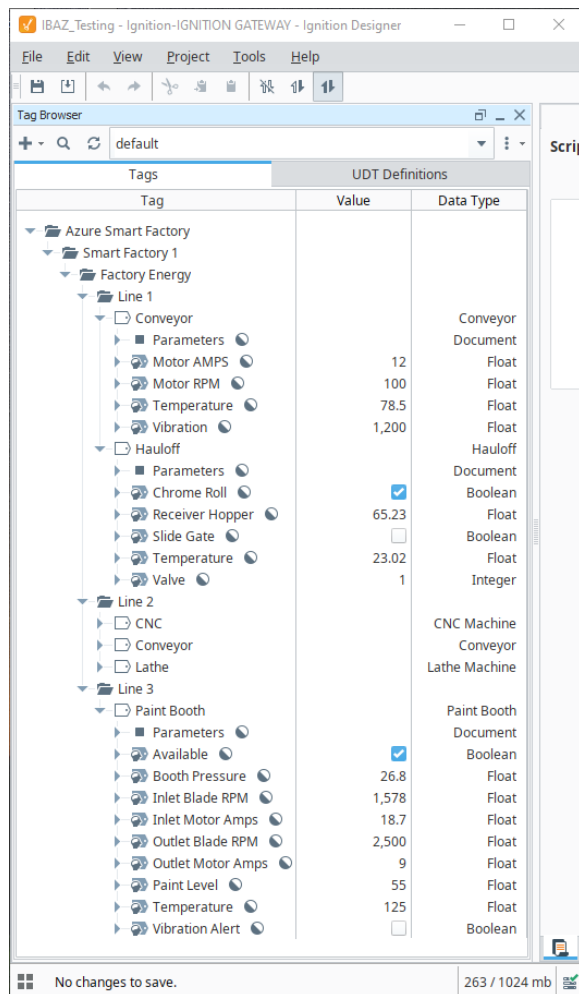
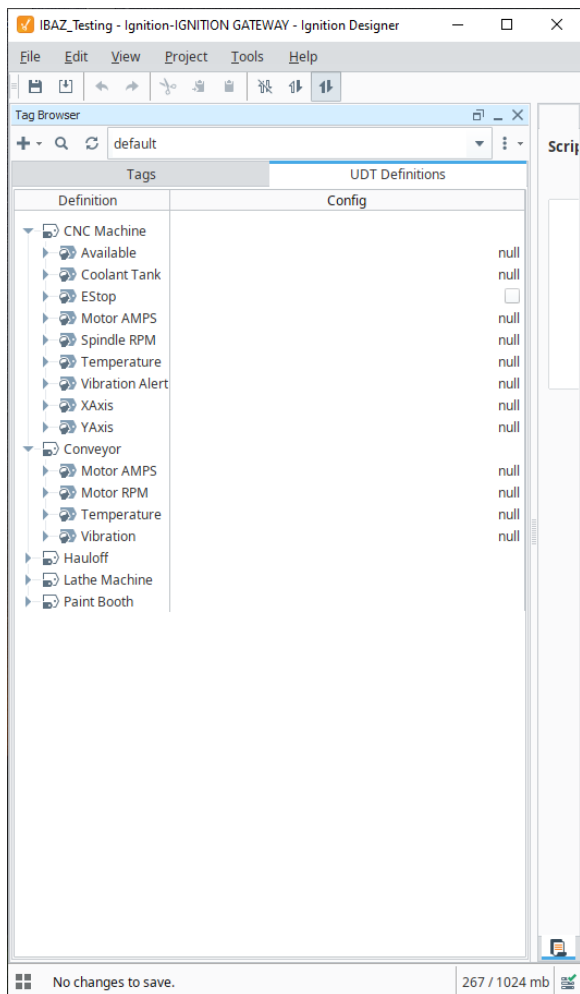
Step 11: Import UDTs and tags

For this quick start, we have created a number of sample UDTs and associated tags that can be easily imported. Using the Ignition Designer, import these UDTs and tags into the default Tag Browser.

[IBAZ_quickstart_UDTs.json](#)

[IBAZ_quickstart_tags.json](#)

Once imported, you can browse into each tag to see the structure of the UDT and UDT Instances.



Refer to Inductive Automations [Exporting and Importing Tags](#) documentation if you are unfamiliar with importing tags.

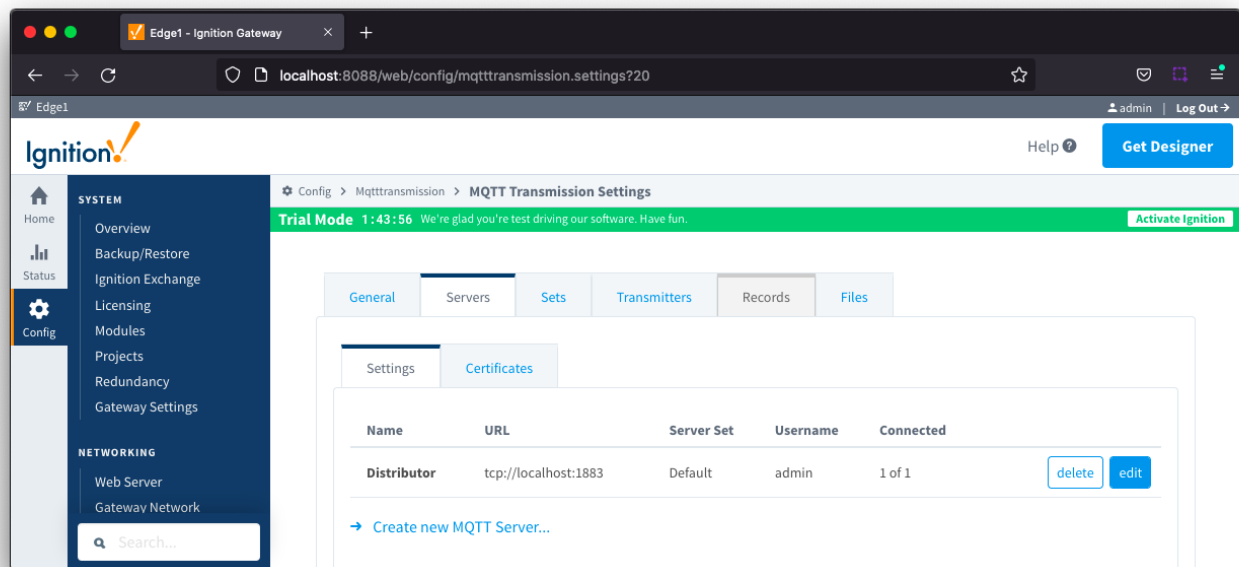
Step 12: Edge Device Configuration

Now we can configure the MQTT Transmission module to connect to MQTT Distributor. From the left hand menu bar in Ignition, select MQTT Transmission > Settings.

Select the Transmitters tab and edit the Default Transmitter by setting the following properties:

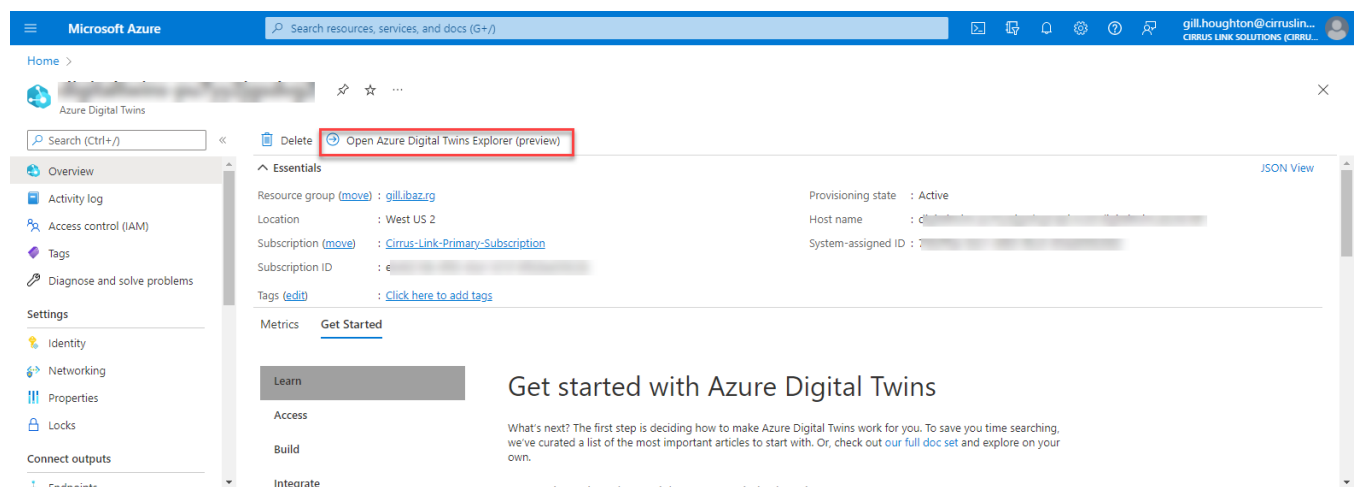
- Tag Path = Azure Smart Factory
- Convert UDTs = Unchecked
- Publish UDT Definitions = Checked
- Optimize UDTs = Checked

Once this configuration is saved, you can confirm you are connected by switching to the Servers tab and verifying that the Connected status shows 1 of 1

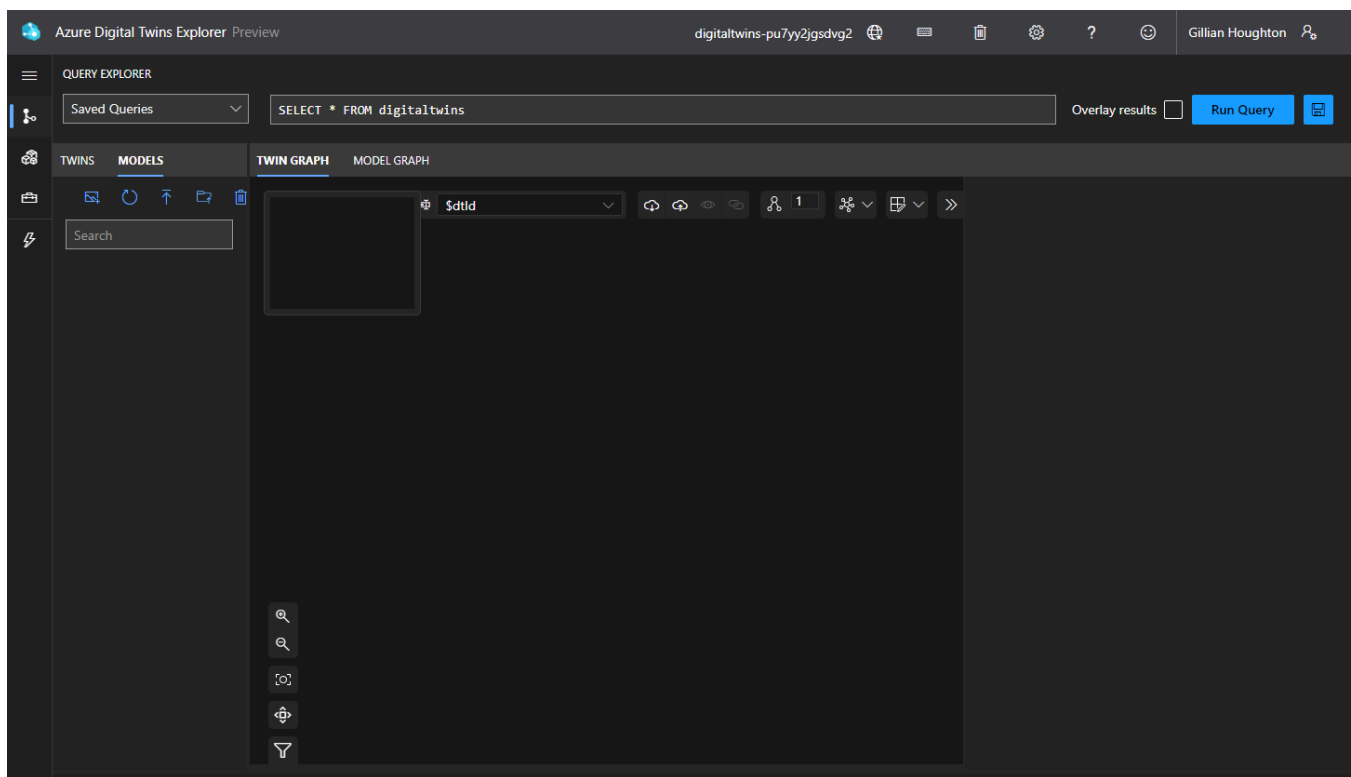


Step 13: Confirm connection with the Azure Digital Twins Explorer

Navigate to your Azure Digital Twins Explorer by clicking on the Overview for your Azure Digital twins and selecting the Open Azure Digital Twins Explorer (preview) link at the top

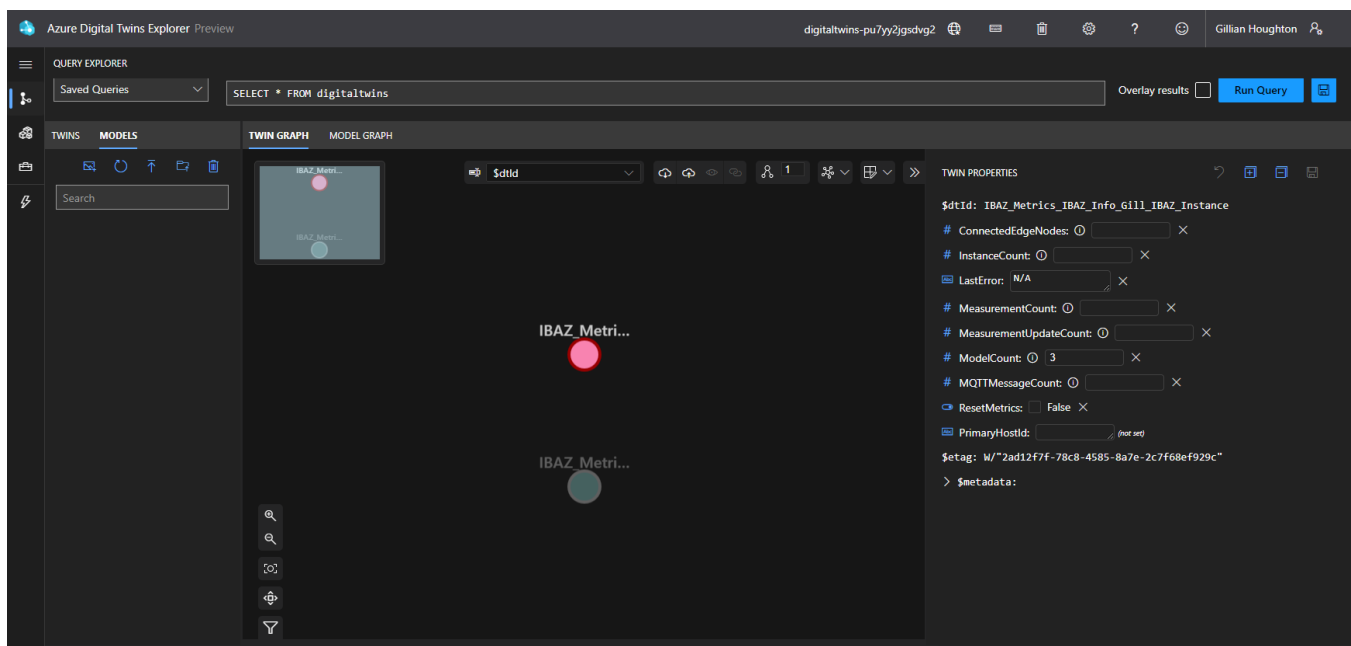


This will display the Azure Digital Twins Explorer view:



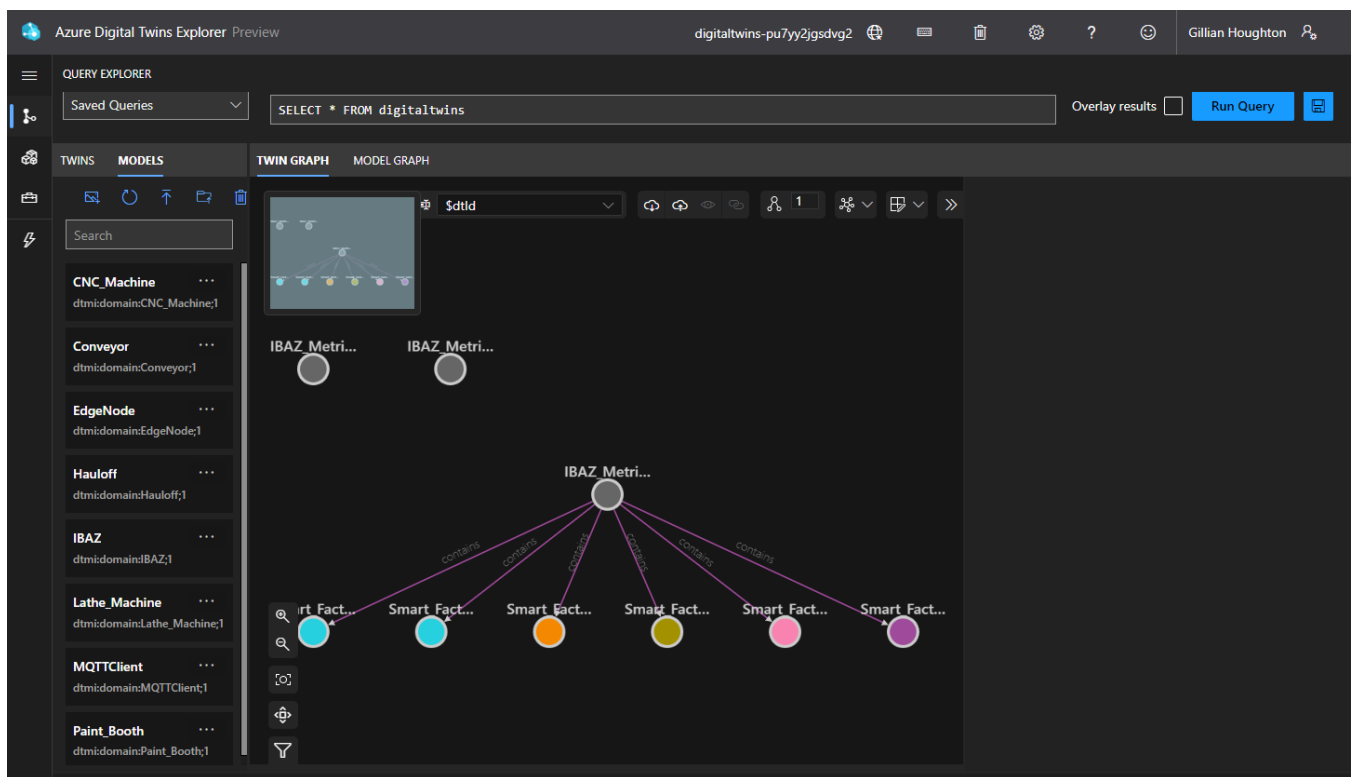
Run Query as `SELECT * FROM digitaltwins` and you will see two digital twins created; one for the IBAZ instance and one for the connected MQTT Client

Refer to the [IBAZ: Mappings and Constraints](#) document for details on digital twin properties

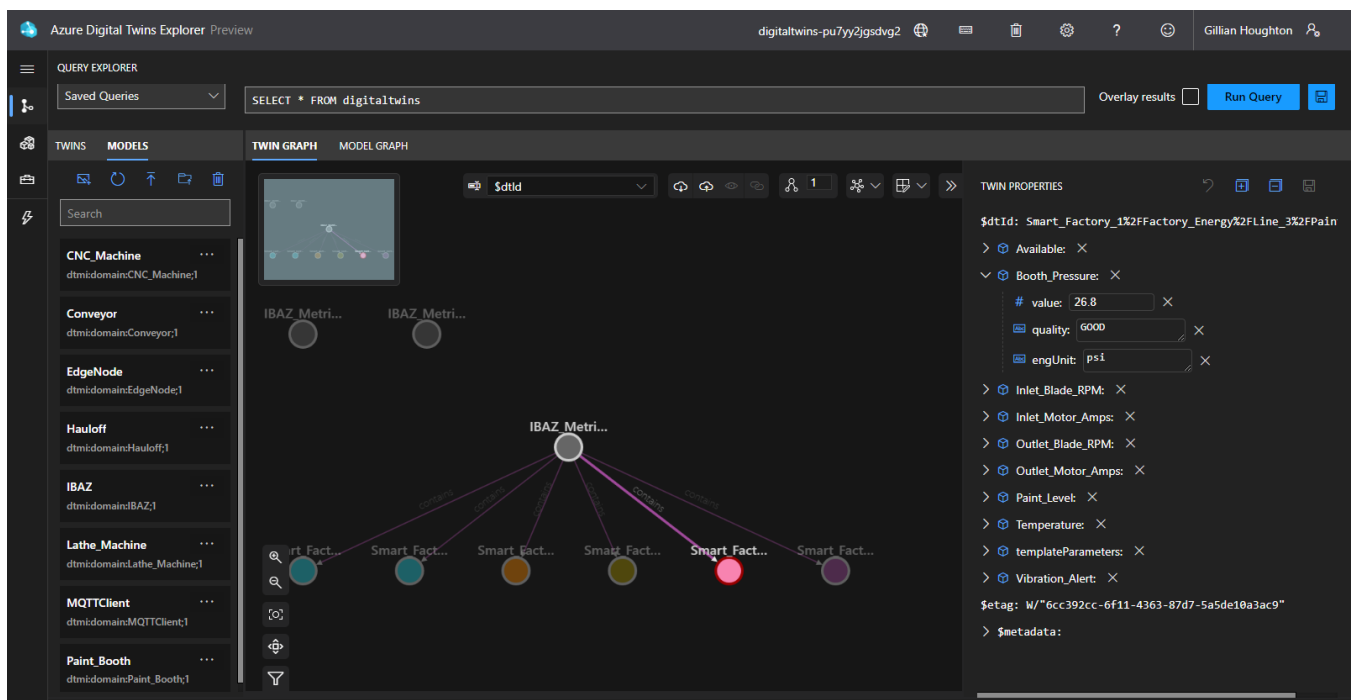


From Ignition Designer, [refresh Transmission](#) to publish the edge node tags to Azure

Run Query as `SELECT * FROM digitaltwins` and also refresh your MODELS. Now you will see that that Digital Twins have been created for each of your UDT Instances and an addition digital twin containing metrics for the edge node.



You can drill into each Digital Twin by selecting the twin and verifying the associated data by checking the Twin Properties. From the Edge Device, change the tag values and see that they are updated.





You will need to Run Query each time to see refreshed data through the Azure Digital Twins Explorer UI.

If you looking to query data on a single digital twin, you can modify the query using the WHERE statement for example: `SELECT * FROM digitaltwins WHERE $dtid = 'Smart_Factory_1%2FFactory_Energy%2FLine_3%2FPaint_Booth'`

Step 14: Confirm data in database

Navigate to your Azure Data Cluster Explorer and select Databases from the left hand menu bar.

The screenshot shows the Azure Data Explorer Cluster Overview page. The left-hand navigation menu has 'Databases' highlighted with a red box. The main content area shows cluster details like Resource group, Location, and State (Running). Below this is a 'Summary' section with cards for Resource health, Advisor recommendations, Scale out, and Insights. At the bottom, there's a 'Monitoring summary' section.

Click on the link for your database and select Query from the left hand menu bar:

The screenshot shows the Azure Data Explorer Cluster Databases page. The left-hand navigation menu has 'Query' highlighted. The main content area shows a table of databases with columns: Database, Size, Retention period, Cache period, Database kind, Shared with others, and Relationship. The table lists a database named 'kustodb' with a size of 5.8 MB and unlimited retention and cache periods.

Database	Size	Retention period	Cache period	Database kind	Shared with others	Relationship
kustodb	5.8 MB	unlimited	unlimited	Read-Write	No	None

You can now [query the data](#) using the Azure Data Explorer tooling to view the data from each of the digital twins.



Data received by Azure Digital Twins Explorer can take up to 5 minutes to be available in Azure Data Explorer

Timestamp	SourceTimeStamp	ModelId	Key	Value
> 2022-08-29 19:00:20.1550		dtmi:domain:Conveyor;1	Temperature	["engUnit":"degF","value":12.5,"quality":"STALE"]
> 2022-08-29 19:00:20.0190		dtmi:domain:Paint_Booth;1	Outlet_Motor_Amps	["engUnit":"amps","value":9,"quality":"STALE"]
> 2022-08-29 19:00:19.8160		dtmi:domain:Paint_Booth;1	Temperature	["engUnit":"degF","value":125,"quality":"STALE"]
> 2022-08-29 19:00:19.7780		dtmi:domain:Paint_Booth;1	Inlet_Blade_RPM	["engUnit":"rpm","value":1578,"quality":"STALE"]
> 2022-08-29 19:00:19.6870		dtmi:domain:Conveyor;1	Motor_RPM	["engUnit":"rpm","value":114,"quality":"STALE"]
> 2022-08-29 19:00:19.6450		dtmi:domain:Conveyor;1	Motor_AMPS	["engUnit":"amps","value":12,"quality":"STALE"]
> 2022-08-29 19:00:19.5930		dtmi:domain:Conveyor;1	Vibration	["engUnit":"hz","value":1300,"quality":"STALE"]
> 2022-08-29 19:00:19.5490		dtmi:domain:Conveyor;1	Temperature	["engUnit":"degF","value":89,"quality":"STALE"]
> 2022-08-29 19:00:19.4840		dtmi:domain:Hauloff;1	Slide_Gate	["engUnit":"status","value":false,"quality":"STALE"]
> 2022-08-29 19:00:19.4380		dtmi:domain:Hauloff;1	Receiver_Hopper	["engUnit":"cnt","value":65.23,"quality":"STALE"]
> 2022-08-29 19:00:19.3960		dtmi:domain:Hauloff;1	Valve	["engUnit":"status","value":1,"quality":"STALE"]
> 2022-08-29 19:00:19.3370		dtmi:domain:Hauloff;1	Chrome_Roll	["engUnit":"status","value":true,"quality":"STALE"]
> 2022-08-29 19:00:19.2980		dtmi:domain:Hauloff;1	Temperature	["engUnit":"degF","value":23.023,"quality":"STALE"]
> 2022-08-29 19:00:09.9930		dtmi:domain:EdgeNode;1	Connected	True
> 2022-08-29 19:00:09.9930		dtmi:domain:EdgeNode;1	ConnectDateTime	2022-08-29T18:19:04.3710000Z
> 2022-08-29 19:00:09.9930		dtmi:domain:EdgeNode;1	DisconnectDateTime	1970-01-01T00:00:00.0000000Z
> 2022-08-29 19:00:09.9930		dtmi:domain:EdgeNode;1	BirthCount	1
> 2022-08-29 19:00:09.9930		dtmi:domain:EdgeNode;1	DeathCount	0

Now the data is in the cloud, there are existing Microsoft documentation useful for processing this data for managing anomaly detection, active monitoring and trending/reporting. The reference links are include below:

- Anomaly detection
 - [Multivariate Anomaly Detection in Azure Data Explorer](#)
 - ADX contains native support for detecting anomalies over multiple time series by using the function [series_decompose_anomalies\(\)](#).
 - [Time series anomaly detection & forecasting](#)
 - This article details time series anomaly detection and forecasting capabilities of KQL.
- Active monitoring
 - [Azure Data Explorer connector for Power Automate](#)
 - Send notifications and alerts based on query results, such as when thresholds exceed certain limits.
- Trending and reporting

- [Azure Data Explorer connector for Power Automate](#)
 - Send regular, such as daily or weekly, reports containing tables and charts.
- [Azure Data Explorer data visualization](#)
 - Data visualization and reporting is a critical step in the data analytics process. Azure Data Explorer supports many BI services so you can use the one that best fits your scenario and budget.