IBSNOW: Azure Quickstart

Prerequisites

- Install IoT Bridge for Snowflake into your Azure account
  - Before being able to access the Virtual Machine you must have completed the installation process here.
- Install an MQTT Server configured with a real signed TLS certificate

> This quickstart guide uses the Chariot MQTT Server which can be installed as a free trial from the Azure Marketplace.
>
> Review the Chariot MQTT Server Configuration for details on how to upload the necessary certificates and keys for enabling SSL/TLS

> If you choose not to use Chariot MQTT Server, any Sparkplug compliant MQTT Server will work.
>
> > Azure IoT Hub is not Sparkplug compliant
>
> > AWS IoT Core has a message size limit of 128KB and will disconnect the client if it receives a message that exceeds this limit. If you have a large number of UDT definitions/instances and/or have very large UDTs, you will very likely hit this limit when sending your UDTs to AWS IoT Core. Review this document for ways to reduce the message size.

Summary

IoT Bridge for Snowflake (IBSNOW) is an application that connects to an MQTT Server (such as Chariot MQTT Server) and consumes MQTT Sparkplug messages from Edge devices.

When these messages are formatted as Sparkplug Templates, as defined in the Sparkplug Specification, the templates are used to create the data in Snowflake automatically with no additional coding or configuration.

> If the messages do not use templates, they will be stored in a database table as unprocessed messages and additional work will be required to use this data in Snowflake.

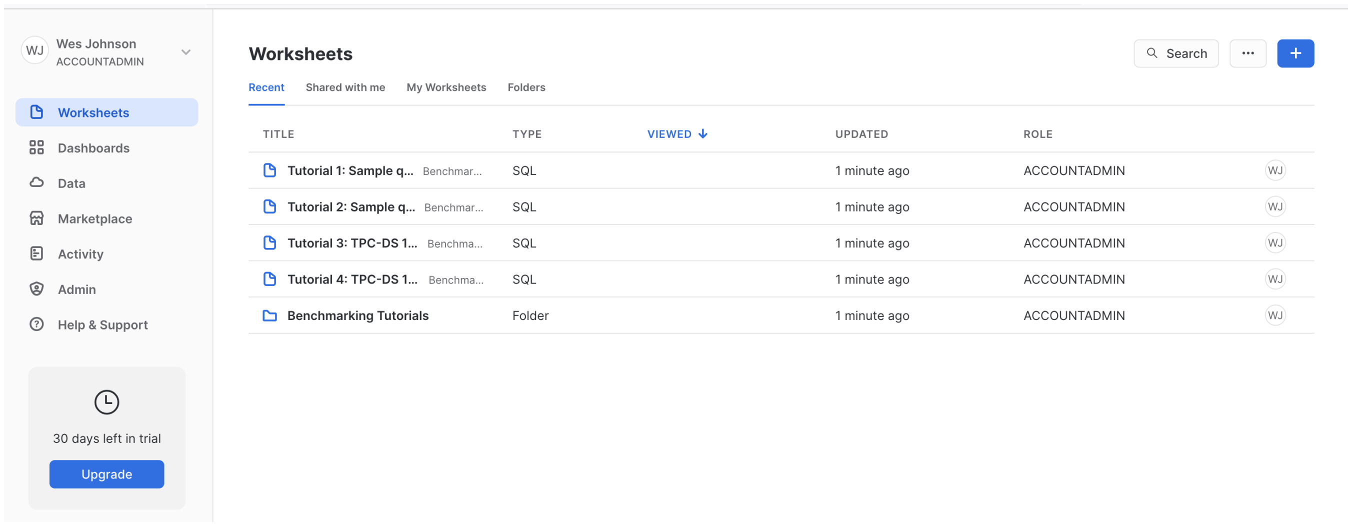
Then multiple instances of these Templates generate the Assets and start to populate with real time data sent on change only, thus significantly reducing the amount of data being sent to the cloud. For further details on Snowflake, refer to the documentation here. For further details on Eclipse Sparkplug, 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 data in Snowflake. 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 Snowflake components. This tutorial will use the Cirrus Link Chariot MQTT Server implementation. However, IBSNOW does work with any MQTT v3.1.1 compliant MQTT Server including Cirrus Link's MQTT Servers.

It is also important to note that 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.

Snowflake Setup

If you don't have a Snowflake account, open a Web Browser and go to https://www.snowflake.com. Follow the instructions there to start a free trial. After creating an account, log in to Snowflake via the Web Console. You should see something like what is shown below.

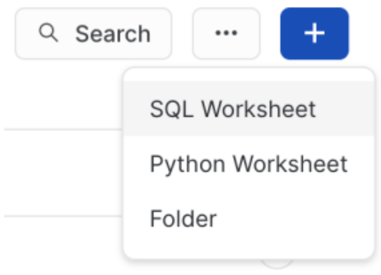


Worksheets

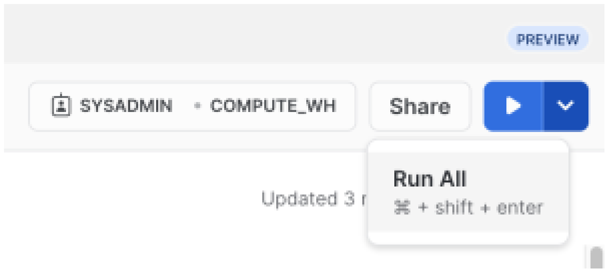
Recent | Shared with me | My Worksheets | Folders

TITLE	TYPE	VIEWED	UPDATED	ROLE	
Tutorial 1: Sample q... Benchmar...	SQL		1 minute ago	ACCOUNTADMIN	WJ
Tutorial 2: Sample q... Benchmar...	SQL		1 minute ago	ACCOUNTADMIN	WJ
Tutorial 3: TPC-DS 1... Benchma...	SQL		1 minute ago	ACCOUNTADMIN	WJ
Tutorial 4: TPC-DS 1... Benchma...	SQL		1 minute ago	ACCOUNTADMIN	WJ
Benchmarking Tutorials	Folder		1 minute ago	ACCOUNTADMIN	WJ

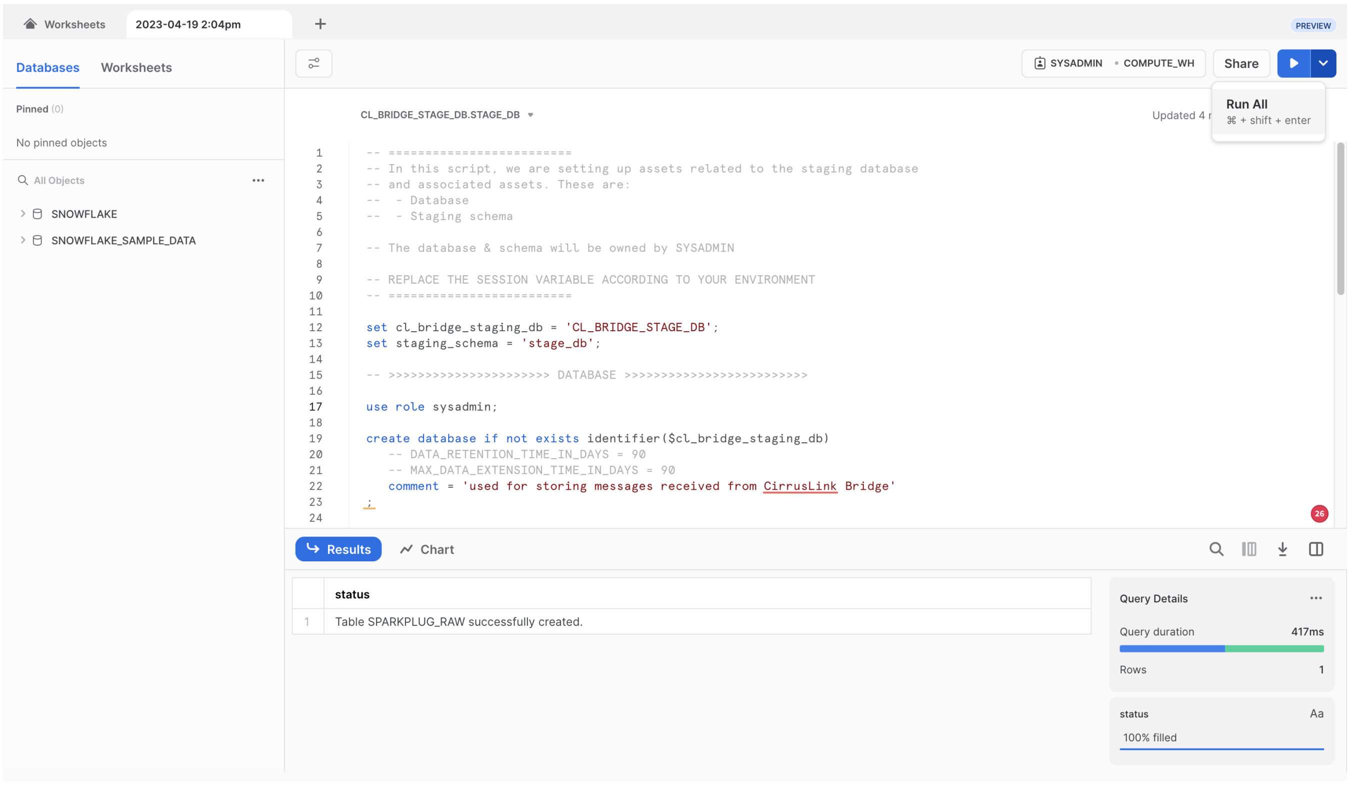
Create a new 'SQL Worksheet' by clicking the blue + button in the upper right hand corner of the window as shown below.



Copy and paste SQL Script 01 from Snowflake Setup Scripts into the center pane of the SQL Worksheet, click the drop down arrow next to the blue play button in the upper right corner of the window and click 'Run All' as shown below.



After doing so, you should see a message in the 'Results' pane denoting the SPARKPLUG_RAW table was created successfully as shown below.



```
-- ========================
-- In this script, we are setting up assets related to the staging database
-- and associated assets. These are:
--   - Database
--   - Staging schema

-- The database & schema will be owned by SYSADMIN

-- REPLACE THE SESSION VARIABLE ACCORDING TO YOUR ENVIRONMENT
-- ========================

set cl_bridge_staging_db = 'CL_BRIDGE_STAGE_DB';
set staging_schema = 'stage_db';

-- >>>>>>>>>>>>>>>>>>>>>> DATABASE >>>>>>>>>>>>>>>>>>>>>>>>>

use role sysadmin;

create database if not exists identifier($cl_bridge_staging_db)
   -- DATA_RETENTION_TIME_IN_DAYS = 90
   -- MAX_DATA_EXTENSION_TIME_IN_DAYS = 90
   comment = 'used for storing messages received from CirrusLink Bridge'
;
```

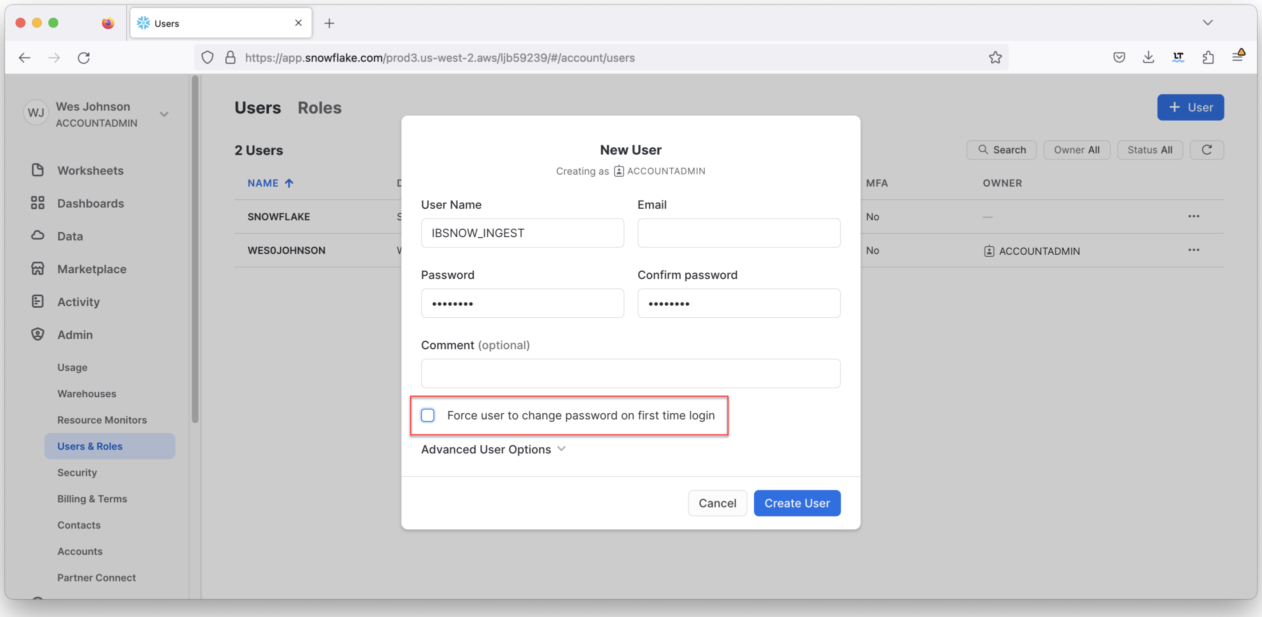
	status
1	Table SPARKPLUG_RAW successfully created.

Now, repeat the process for each of the following scripts in the Snowflake Setup Scripts in order. Each time, fully replace the contents of the SQL script with the new script and click the 'Run All' button after pasting each script. Make sure no errors are displayed in the Results window after running each script.

- SQL Script 02 Expected Result: Stream NBIRTH_STREAM successfully created.
- SQL Script 03 Expected Result: Function GENERATE_DEVICE_ASOF_VIEW_DDL successfully created.
- SQL Script 04 Expected Result: Function CREATE_EDGE_NODE_SCHEMA successfully created.
- SQL Script 05 Expected Result: Function CREATE_ALL_EDGE_NODE_SCHEMAS successfully created.
- SQL Script 06 Expected Result: Statement executed successfully.
- SQL Script 07 Expected Result: Statement executed successfully.
- SQL Script 08 Expected Result: Statement executed successfully.

After all of the scripts have successfully executed, create a new user in Snowflake. This user will be used by IoT Bridge for Snowflake to push data into Snowflake. In the Snowflake Web UI, go to Admin Users & Roles and then click '+ User' in the upper right hand corner. Give it a username of your choice and a secure password as shown below. For this example we're calling the user IBSNOW_INGEST so we know this user is for ingest purposes. See below for an example and then click 'Create User'.

Force user to change password on first time login must be set to False.



New User

Creating as ACCOUNTADMIN

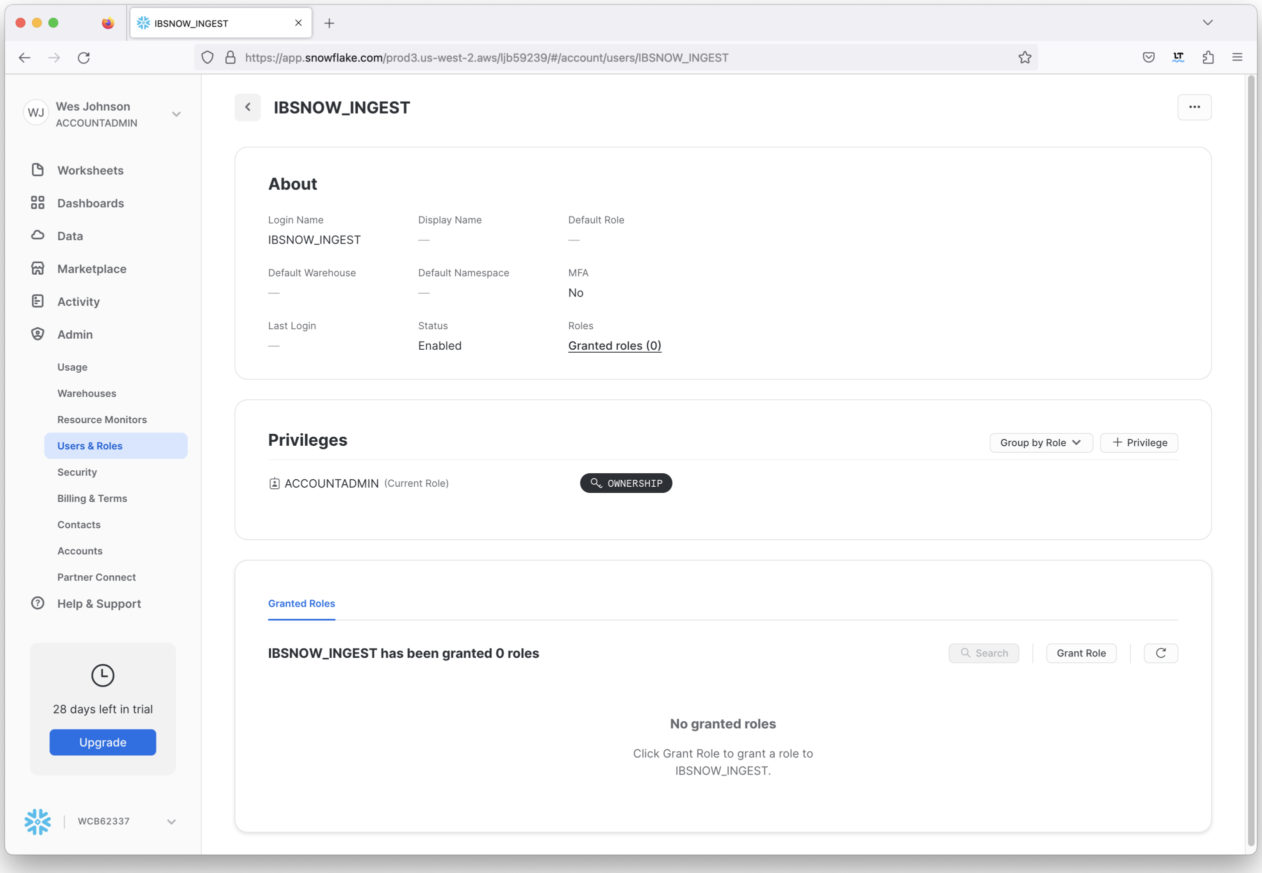
User Name	IBSNOW_INGEST
Email	
Password	••••••••
Confirm password	••••••••
Comment (optional)	

☐ Force user to change password on first time login

2 Users

NAME	MFA	OWNER
SNOWFLAKE	No	—
WESOJOHNSON	No	ACCOUNTADMIN

In addition, the user must have a specific role to be able to stream data into Snowflake. Click the newly created user to see the following.



IBSNOW_INGEST

About

Login Name	Display Name	Default Role
IBSNOW_INGEST	—	—
Default Warehouse	Default Namespace	MFA
—	—	No
Last Login	Status	Roles
—	Enabled	Granted roles (0)

Privileges

ACCOUNTADMIN (Current Role) OWNERSHIP

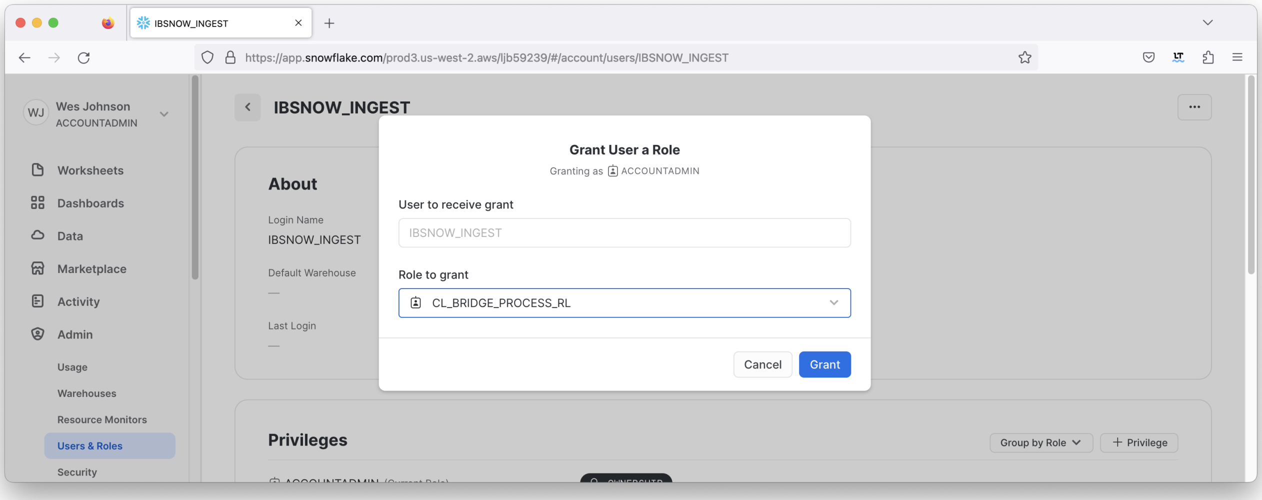
Granted Roles

IBSNOW_INGEST has been granted 0 roles

No granted roles

Click Grant Role to grant a role to IBSNOW_INGEST.

In the bottom of the center 'Granted Roles' pane you will see this user has no roles. Click 'Grant Role' to set up a new role. Then, select the 'CL_BRIDGE_PROCESS_RL' role and click 'Grant' as shown below.



Grant User a Role

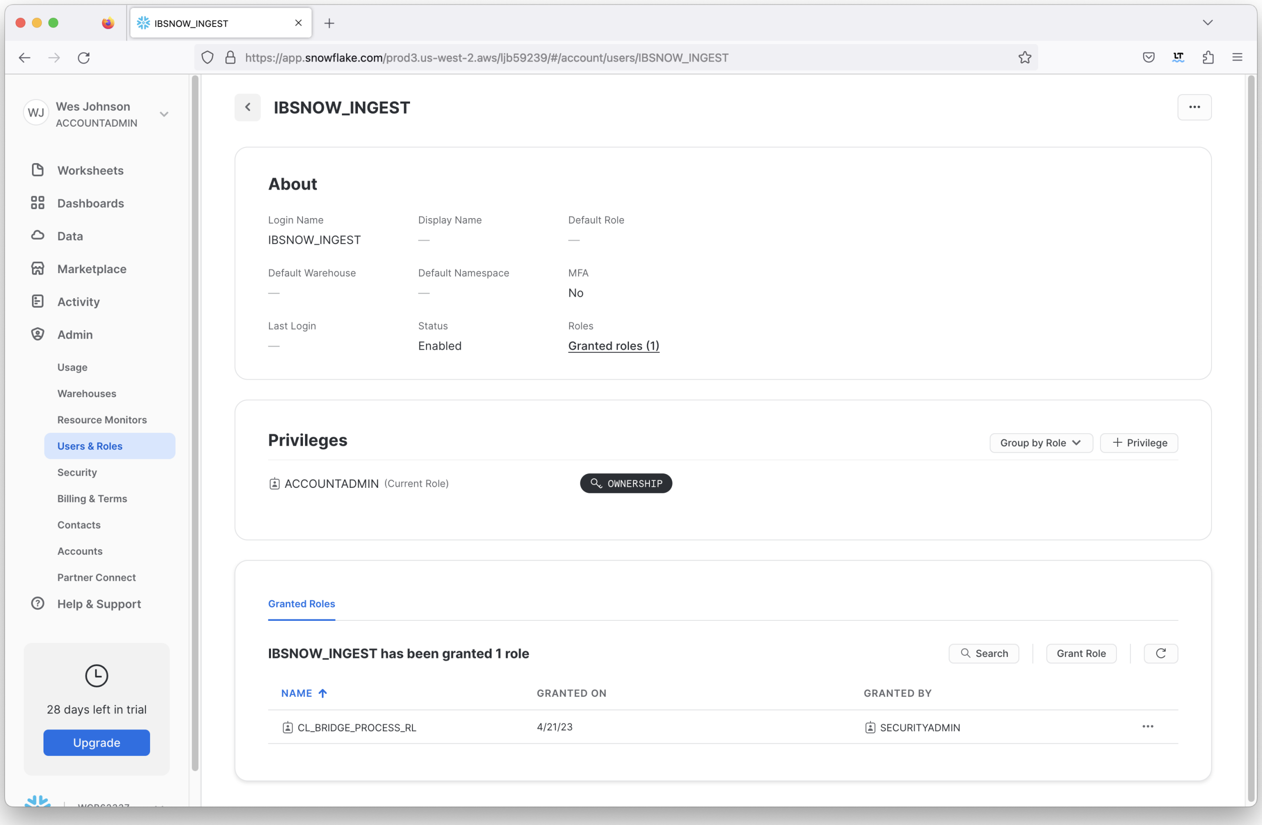
Granting as ACCOUNTADMIN

User to receive grant: IBSNOW_INGEST

Role to grant: CL_BRIDGE_PROCESS_RL

Cancel | Grant

After this has been done successfully you will see the role now associated with the new user as shown below.



IBSNOW_INGEST

About

Login Name IBSNOW_INGEST	Display Name —	Default Role —
Default Warehouse —	Default Namespace —	MFA No
Last Login —	Status Enabled	Roles Granted roles (1)

Privileges

ACCOUNTADMIN (Current Role) — OWNERSHIP

Granted Roles

IBSNOW_INGEST has been granted 1 role

NAME	GRANTED ON	GRANTED BY
CL_BRIDGE_PROCESS_RL	4/21/23	SECURITYADMIN

Now an unencrypted key pair must be generated and uploaded to Snowflake. This will be used for authentication by the IoT Bridge for Snowflake application to push data to Snowflake via the Snowflake Streaming API.

Attach the generated unencrypted public key to the IBSNOW_INGEST user that we just created for Snowflake ingest purposes.

See this document for details on how to generate this unencrypted key and assign this to a user in your snowflake account: https://docs.snowflake.com/en/user-guide/key-pair-auth.

Note: The step "Configuring the Snowflake Client to User Key Pair Authentication" in the linked tutorial can be skipped.

The generated key MUST NOT be encrypted

IoT Bridge Setup

First you will need access to the Azure Virtual Machine via SSH. See this document for information on how to access the VM.

Configuring the Snowflake properties

Modify the file **/opt/ibsnow/conf/ibsnow.properties** file. Set the following:

- mqtt_server_url
    - This is the MQTT Server endpoint URL. It must be of the form: ssl://ENDPOINT_URL:8883
- mqtt_server_name
    - Give it a meaningful name if desired
- mqtt_username
    - The username for the MQTT connection if required
    - If using Chariot MQTT Server, the default username is 'admin'
- mqtt_password
    - The password for the MQTT connection if required
    - If using Chariot MQTT Server, the default password is 'changeme'
- primary_host_id
    - Set it to a text string such as 'IamHost'
- snowflake_streaming_client_name
    - Some text string such as 'MY_CLIENT'
- snowflake_streaming_table_name
    - This is the staged_sparkplug_raw_table created by the Snowflake setup in SQL Script 02
    - If the default Snowflake setup scripts were used, this is 'SPARKPLUG_RAW'
- snowflake_notify_db_name
    - This is the cl_bridge_node_db created by the Snowflake setup in SQL Script 06
    - If the default Snowflake setup scripts were used, this is 'cl_bridge_node_db'
- snowflake_notify_schema_name
    - This is the stage_db created by the Snowflake setup in SQL Script 06
    - If the default Snowflake setup scripts were used, this is 'stage_db'
- snowflake_notify_warehouse_name
    - This is the cl_bridge_ingest_wh created by the Snowflake setup in SQL Script 07
    - If the default Snowflake setup scripts were used, this is 'cl_bridge_ingest_wh'

When complete, it should look similar to what is shown below.

If you are using self-signed certificates rather than a real signed certificate, you will need to copy the CA certificate chain file uploaded to your MQTT Server to the bridge instance and set

- mqtt_ca_cert_chain_path.1
    - This is the filepath to the TLS Certificate Authority certificate chain

Chariot MQTT Server using a real signed TLS certificate

```
# The IBSNOW instance friendly name. If ommitted, it will become 'IBSNOW-ec2-instance-id'
#ibsnow_instance_name =

# The Cloud region the IoT Bridge for Snowflake instance is in
# ibsnow_cloud_region = us-east-1

# MQTT Server definitions. IoT Bridge for Snowflake supports multiple MQTT Servers. Each definition must
include and 'index' as shown
# below represented by 'X'. The first should begin with 1 and each additional server definition should have an
index of 1 greater
# than the previous.
# mqtt_server_url.X                     # The MQTT Server URL
# mqtt_server_name.X                    # The MQTT Server name
# mqtt_username.X                       # The MQTT username (if required by the MQTT Server)
# mqtt_password.X                       # The MQTT password (if required by the MQTT Server)
# mqtt_keepalive_timeout.X              # The MQTT keep-alive timeout in seconds
# mqtt_ca_cert_chain_path.X             # The filepath to the TLS Certificate Authority certificate chain
# mqtt_client_cert_path.X               # The filepath to the TLS certificate
# mqtt_client_private_key_path.X        # The filepath to the TLS private key
# mqtt_client_private_key_password.X    # The TLS private key password
# mqtt_verify_hostname.X                # Whether or not to verify the hostname against the server certificate
# mqtt_client_id.X                      # The Client ID of the MQTT Client
# mqtt_sparkplug_subscriptions.X        # The Sparkplug subscriptions to issue when connecting to the MQTT
Server.
                    # By default this is spBv1.0/# but can be scoped more narrowly (e.g. spBv1.0/Group1/#)
                                        # It can also be a comma separated list (e.g. spBv1.0/Group1/#,spBv1.0
/Group2/#)

mqtt_server_url.1 = ssl://chariot.mycompany.com:8883
mqtt_server_name.1 = Chariot MQTT Server
mqtt_sparkplug_subscriptions.1 = spBv1.0/#
#mqtt_keepalive_timeout.1 = 30
#mqtt_verify_hostname.1 = true
mqtt_username.1 = admin
mqtt_password.1 = changeme
#mqtt_ca_cert_chain_path.1 = /opt/ibsnow/conf/certs/myCACert.epm
#mqtt_client_cert_path.1 = /opt/ibsnow/conf/certs/myClientcert.pem
#mqtt_client_private_key_path.1 = /opt/ibsnow/conf/certs/myPrivateKey.pem
#mqtt_client_private_key_password.1 =
#mqtt_client_id.1 =

# The Sparkplug sequence reordering timeout in milliseconds
sequence_reordering_timeout = 5000

# Whether or not to block auto-rebirth requests
#block_auto_rebirth = false

# The primary host ID if this is the acting primary host
primary_host_id = IamHost

# Snowflake streaming connection properties - A custom client name for the connection (e.g. MyClient)
snowflake_streaming_client_name = IBSNOWClient

# Snowflake streaming connection properties - The scheme to use for channels and their names
# This MUST be one of the following: STATIC, GROUP_ID, EDGE_ID
# STATIC - means to use a single channel. If using this mode, the snowflake_streaming_channel_name
# GROUP_ID - means to use the Sparkplug Group ID for the channel name on incoming data
# EDGE_ID - means to use the Sparkplug Group ID and the Edge Node ID for the channel name on incoming data
# DEVICE_ID - means to use the Sparkplug Group ID, Edge Node ID, and Device ID for the channel name on incoming
data
snowflake_streaming_channel_scheme = EDGE_ID

# Snowflake streaming connection properties - A custom channel name for the connection (e.g. MyChannel)
# If this is left blank/empty, Channel names of the Sparkplug Group ID will be used instead of a single channel
# snowflake_streaming_channel_name =

# Snowflake streaming connection properties - The Table name associated with the Database and Schema already
provisioned in the Snowflake account (e.g. MyTable)
snowflake_streaming_table_name = SPARKPLUG_RAW

# Snowflake notify connection properties - The Database name associated with the connection that is already
provisioned in the Snowflake account (e.g. MyDb)
snowflake_notify_db_name = cl_bridge_node_db

# Snowflake notify connection properties - The Schema name associated with the Database already provisioned in
the Snowflake account (e.g. PUBLIC)
snowflake_notify_schema_name = stage_db

# Snowflake notify connection properties - The Warehouse name associated with the notifications already
provisioned in the Snowflake account (e.g. PUBLIC)
snowflake_notify_warehouse_name = cl_bridge_ingest_wh

# Whether or not to create and update IBSNOW infomational tracking metrics
# ibsnow_metrics_enabled = true

# The Sparkplug Group ID to use for IBSNOW asset names
ibsnow_metrics_sparkplug_group_id = IBSNOW

# The 'Bridge Info' Sparkplug Edge Node ID to use for IBSNOW assets
ibsnow_metrics_bridge_info_sparkplug_edge_node_id = Bridge Info

# The 'Edge Node Info' Sparkplug Edge Node ID to use for IBSNOW assets
ibsnow_metrics_edge_node_info_sparkplug_edge_node_id = Edge Node Info

# The 'MQTT Client Info' Sparkplug Edge Node ID to use for IBSNOW assets
ibsnow_metrics_mqtt_client_info_sparkplug_edge_node_id = MQTT Client Info

# Whether or not to send notification tasks to Snowflake based on incoming Sparkplug events
snowflake_notify_task_enabled = true

# The number of threads to use for BIRTH handling in Snowflake
# snowflake_notify_task_birth_thread_count = 100

# The number of milliseconds to delay after receiving an NBIRTH before notifying Snowflake over the event
(requires snowflake_notify_task_enabled is true)
snowflake_notify_nbirth_task_delay = 10000

# The number of milliseconds to delay after receiving a DBIRTH or DATA message before notifying Snowflake over
the event (requires snowflake_notify_task_enabled is true)
snowflake_notify_data_task_delay = 5000
```

Configuring the snowflake streaming profile

Now, modify the file **/opt/ibsnow/conf/snowflake_streaming_profile.json** file as described in Setting snowflake_streaming_profile configuration

When complete, it should look similar to what is shown below.

```
{
  "user": "IBSNOW_INGEST",
  "url": "https://ueb12345.snowflakecomputing.com",
  "account": ueb12345",
  "private_key_file": "/opt/ibsnow/conf/certs/rsa_key.p8",
  "port": 443,
  "host": "ueb12345.snowflakecomputing.com",
  "schema": "stage_db",
  "scheme": "https",
  "database": "cl_bridge_stage_db",
  "connect_string": "jdbc:snowflake://ueb12345.snowflakecomputing.com:443",
  "ssl": "on",
  "warehouse": "cl_bridge_ingest_wh",
  "role": "cl_bridge_process_rl"
}
```

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

```
sudo systemctl restart ibsnow
```

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

```
tail -f /opt/ibsnow/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 ...'. That denotes we have successfully configured IBSNOW and properly provisioned MQTT Server.

```
INFO|7263/0||23-06-29 20:19:32|20:19:32.932 [Thread-2] INFO  org.eclipse.tahu.mqtt.TahuClient - IBSNOW-8bc00095-
9265-41: Creating the MQTT Client to ssl://54.236.16.39:8883 on thread Thread-2
INFO|7263/0||23-06-29 20:19:33|20:19:33.275 [MQTT Call: IBSNOW-8bc00095-9265-41] INFO  org.eclipse.tahu.mqtt.
TahuClient - IBSNOW-8bc00095-9265-41: connect with retry succeeded
INFO|7263/0||23-06-29 20:19:33|20:19:33.280 [MQTT Call: IBSNOW-8bc00095-9265-41] INFO  org.eclipse.tahu.mqtt.
TahuClient - IBSNOW-8bc00095-9265-41: Connected to ssl://54.236.16.39:8883
INFO|7263/0||23-06-29 20:19:33|20:19:33.294 [MQTT Call: IBSNOW-8bc00095-9265-41] INFO  o.eclipse.tahu.host.
TahuHostCallback - This is a offline STATE message from IamHost - correcting with new online STATE message
FINEST|7263/0||23-06-29 20:19:33|20:19:33.297 [MQTT Call: IBSNOW-8bc00095-9265-41] INFO  o.eclipse.tahu.host.
TahuHostCallback - This is a offline STATE message from IamHost - correcting with new online STATE message
FINEST|7263/0||23-06-29 20:19:33|20:19:33.957 [Thread-2] INFO  org.eclipse.tahu.mqtt.TahuClient - IBSNOW-
8bc00095-9265-41: MQTT Client connected to ssl://54.236.16.39:8883 on thread Thread-2
```

Edge Setup with Ignition and MQTT Transmission

Install Ignition and MQTT Transmission module

At this point IoT Bridge is configured and ready to receive data. To get data flowing into IBSNOW we'll set up Inductive Automation's Ignition platform along with the MQTT Transmission module from Cirrus Link.

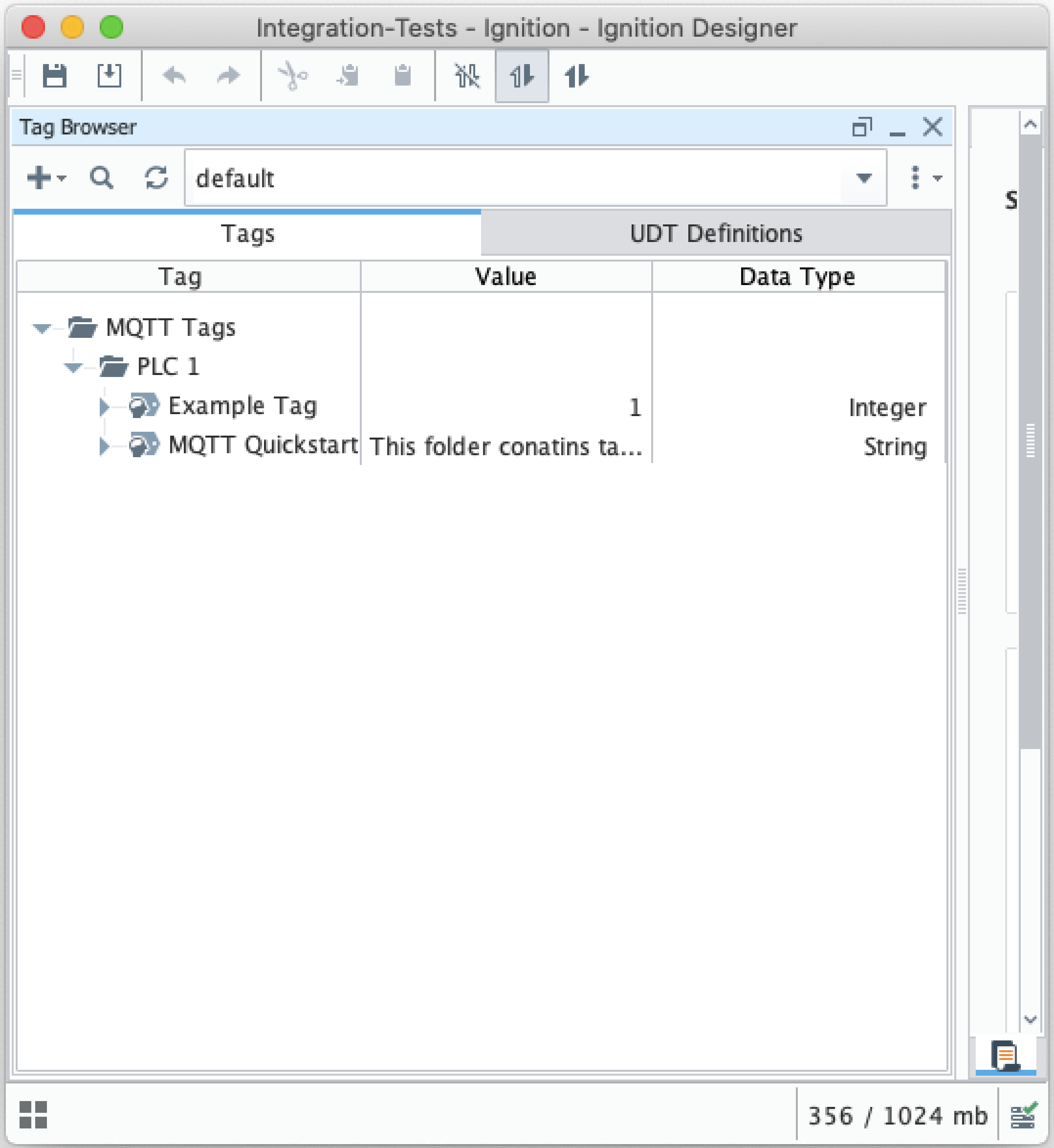
Installation of Ignition is very straightforward following the instructions in the Installing and Upgrading Ignition guide.

With Ignition installed, the Cirrus Link MQTT Transmission module must be installed as a plugin to Ignition. Follow the instructions in our Module Installation guide

Import UDTs and tags

Launch the Ignition Designer to connect to your Ignition instance.

Once it is launched, navigate to the 'default' tag provider in the Tag Browser, expand the tag tree to see the automatically created tags as shown below and delete tags Example Tag and MQTT Quickstart.

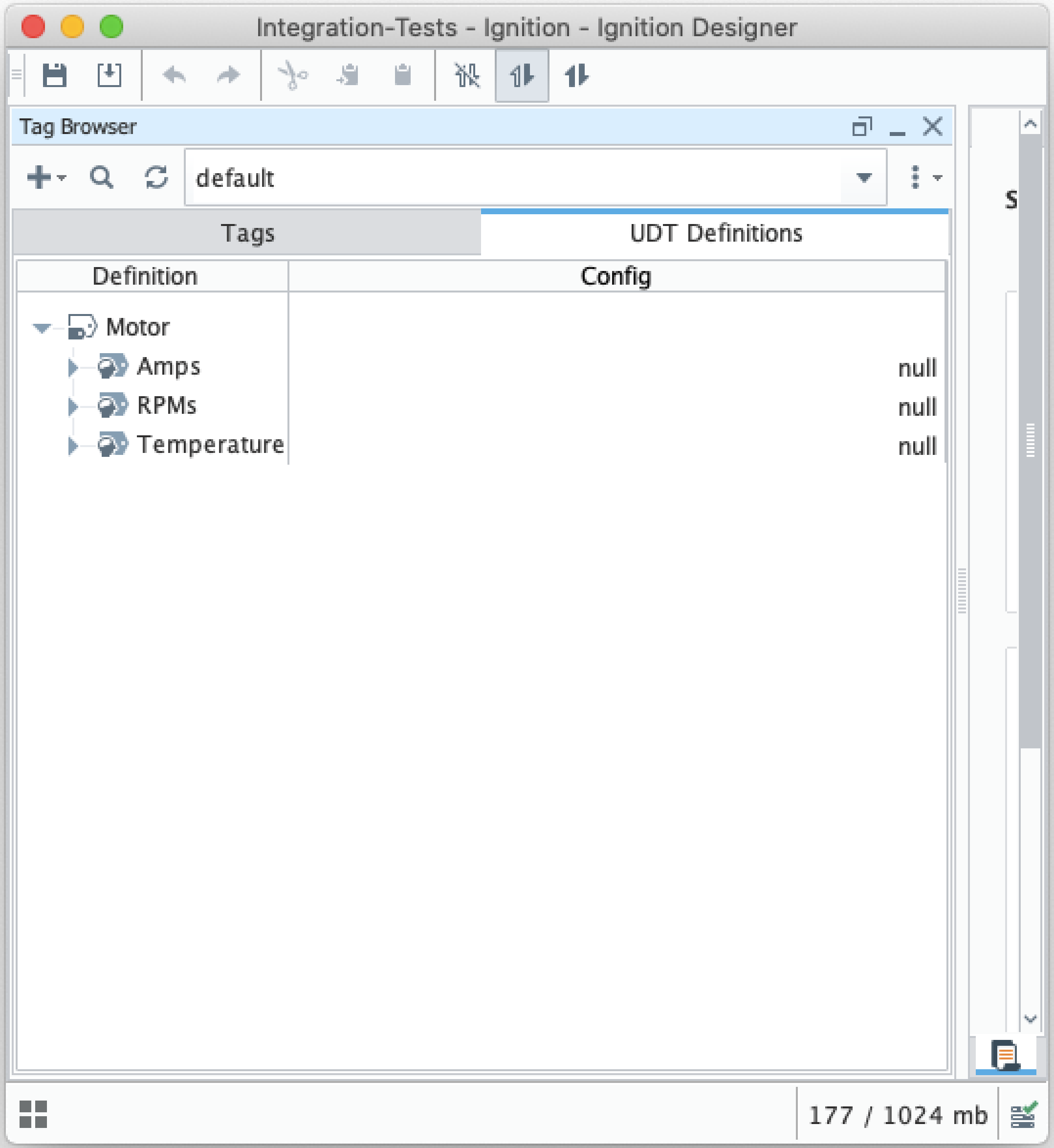


Tag	Value	Data Type
MQTT Tags		
PLC 1		
Example Tag	1	Integer
MQTT Quickstart	This folder conatins ta...	String

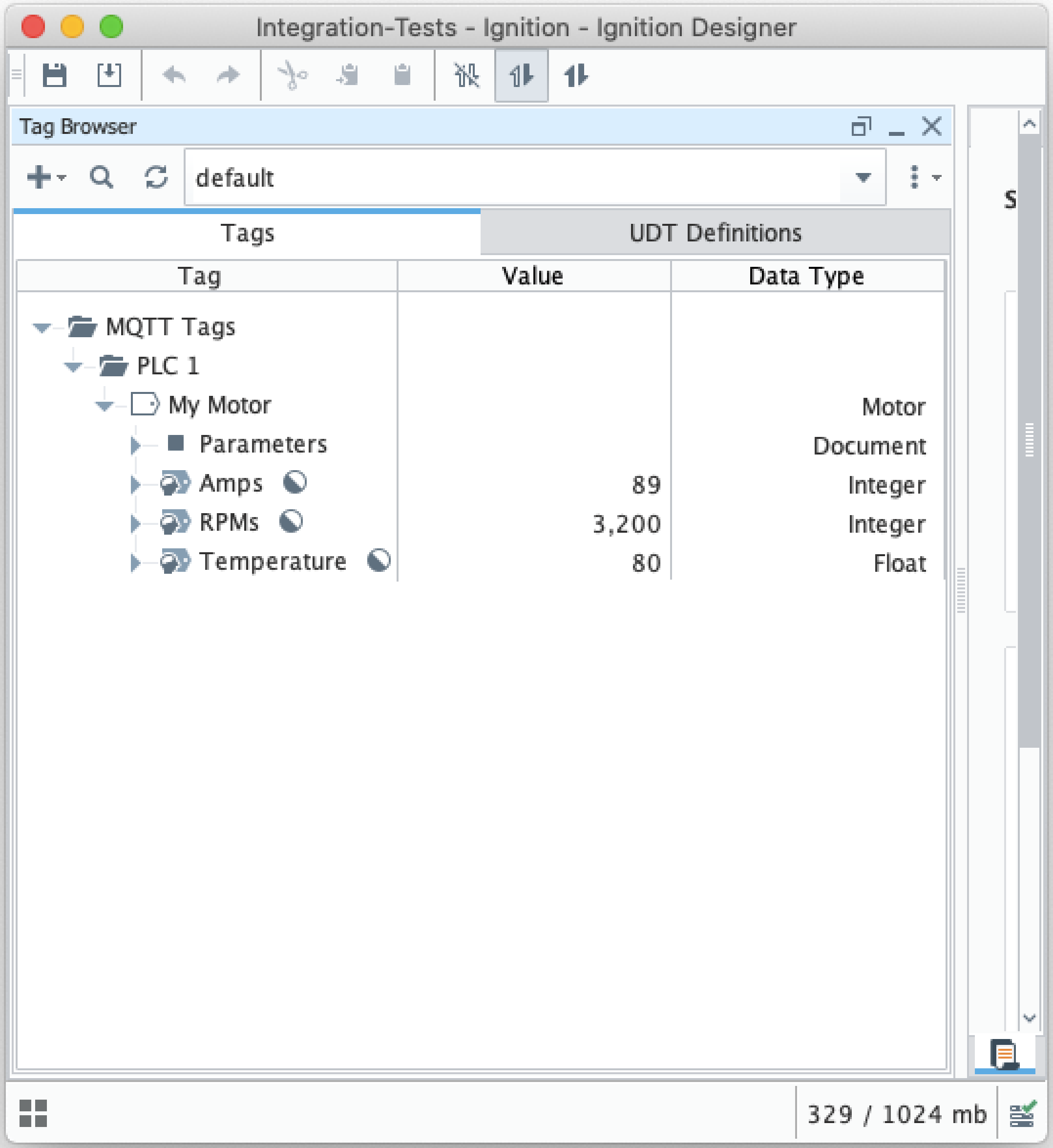
From the Designer import these tags IBSNOW_Quickstart_tags.json to MQTT Tags > PLC 1 to create a UDT Definition and instance.

Review the Ignition Exporting and Importing Tags document if needed

You can view the imported UDT Definition and instance in the tag browser:



Definition	Config
Motor	
Amps	null
RPMs	null
Temperature	null


Integration-Tests - Ignition - Ignition Designer

Tag Browser

default

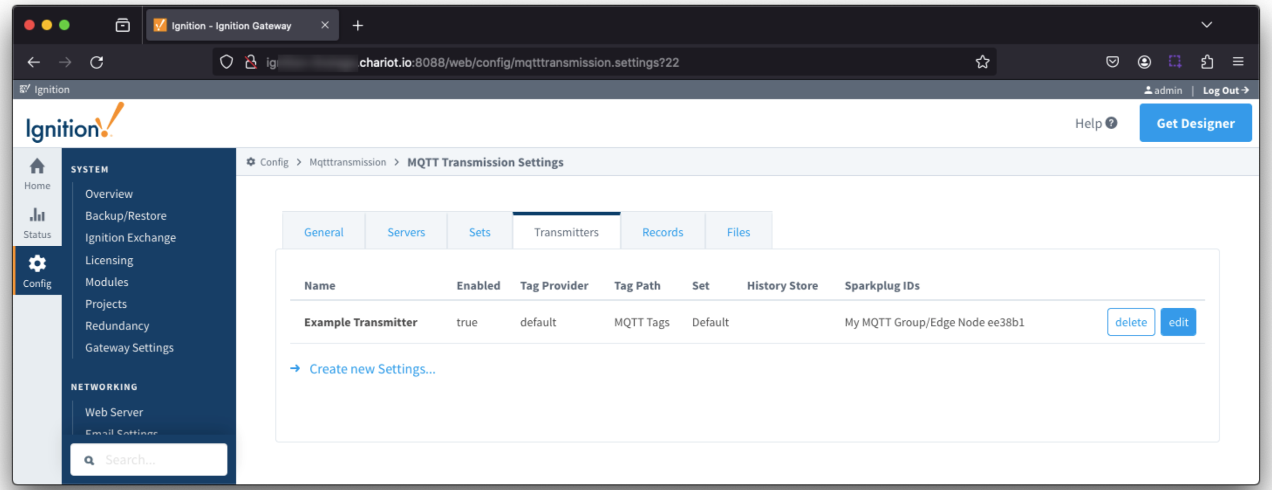
Tag	Value	Data Type
MQTT Tags		
PLC 1		
My Motor		Motor
Parameters		Document
Amps	89	Integer
RPMs	3,200	Integer
Temperature	80	Float

329 / 1024 mb

At this point, our tags are configured. A UDT definition will map to a model in Snowflake and UDT instances in Ignition will map to Snowflake.

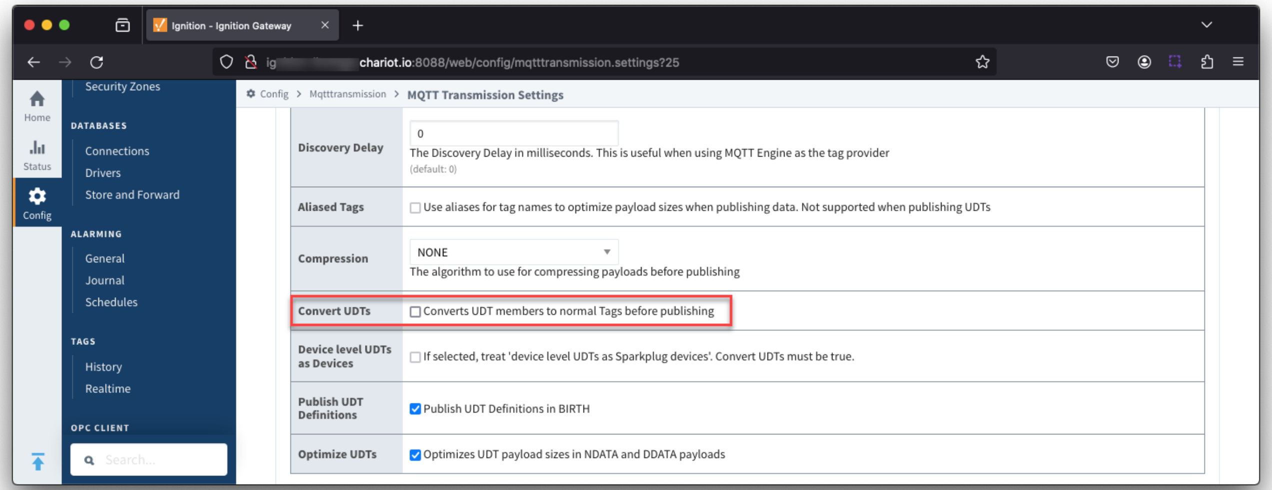
But, before this will happen we need to point MQTT Transmission to the Chariot MQTT Server. To do so, browse back to the Ignition Gateway Web UI and select MQTT Transmission Settings from the left navigation panel.

Select the 'Transmitters' tab as shown below.



Name	Enabled	Tag Provider	Tag Path	Set	History Store	Sparkplug IDs
Example Transmitter	true	default	MQTT Tags	Default		My MQTT Group/Edge Node ee38b1

Now click the 'edit' button to the right of the 'Example Transmitter'. Scroll down to the 'Convert UDTs' option and uncheck it as shown below. This will also un-grey the 'Publish UDT Definitions' option. Leave it selected as shown below.

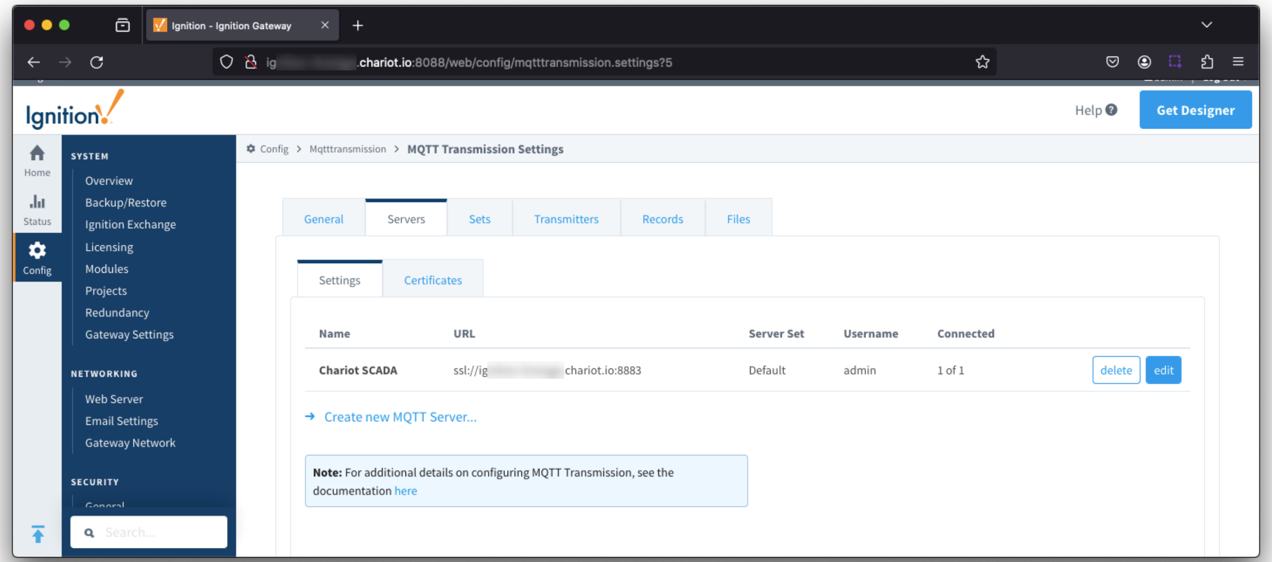

Ignition - Ignition Gateway
ig[redacted]chariot.io:8088/web/config/mqtttransmission.settings?25
Security Zones
Home
Status
Config
DATABASES
Connections
Drivers
Store and Forward
ALARMING
General
Journal
Schedules
TAGS
History
Realtime
OPC CLIENT
Search...
Config > Mqtttransmission > MQTT Transmission Settings

Discovery Delay	0 The Discovery Delay in milliseconds. This is useful when using MQTT Engine as the tag provider (default: 0)
Aliased Tags	☐ Use aliases for tag names to optimize payload sizes when publishing data. Not supported when publishing UDTs
Compression	NONE The algorithm to use for compressing payloads before publishing
Convert UDTs	☐ Converts UDT members to normal Tags before publishing
Device level UDTs as Devices	☐ If selected, treat 'device level UDTs as Sparkplug devices'. Convert UDTs must be true.
Publish UDT Definitions	✓ Publish UDT Definitions in BIRTH
Optimize UDTs	✓ Optimizes UDT payload sizes in NDATA and DDATA payloads

Now switch to the 'Servers' and 'Settings' tab. Delete the existing 'Chariot SCADA' pre-seeded MQTT Server Definition. Then create a new one with the following configuration.

- Name
  - Chariot MQTT Server
- URL
  - Your MQTT Server Endpoint URL of the form ssl://ENDPOINT_URL:8883
- Username
  - Your username for the Chariot MQTT Server connection
  - If using Chariot MQTT Server, the default username is 'admin'
- Password
  - Your password for the Chariot MQTT Server connection
  - If using Chariot MQTT Server, the default password is 'changeme'

When complete, you should see something similar to the following. **However, the 'Connected' state should show '1 of 1' if everything was configured properly.**



Name	URL	Server Set	Username	Connected
Chariot SCADA	ssl://ig[redacted]chariot.io:8883	Default	admin	1 of 1

Note: For additional details on configuring MQTT Transmission, see the documentation here

At this point, data should be flowing into Snowflake.

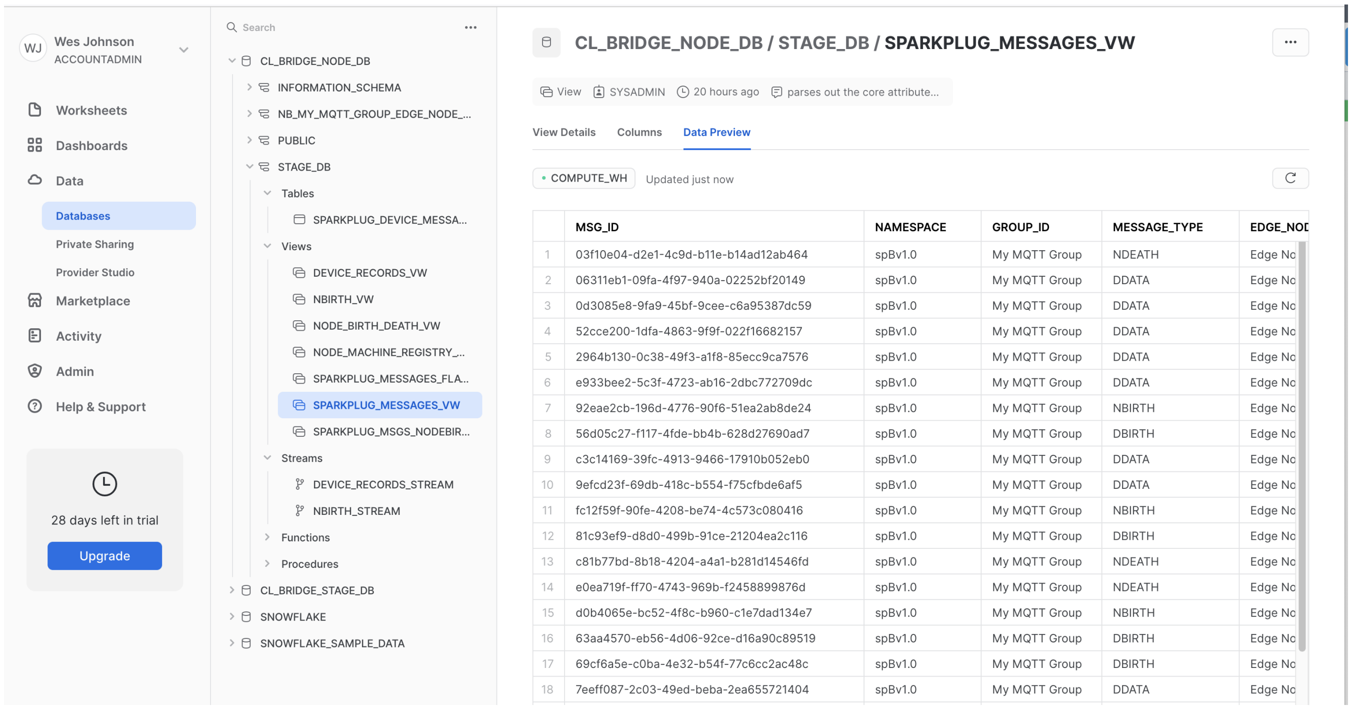
By tailing the log in IBSNOW you should see something similar to what is shown below which shows IBSNOW receiving the messages published from Ignition/MQTT Transmission.

When IBSNOW receives the Sparkplug MQTT messages, it creates and updates asset models and assets in Snowflake. The log below is also a useful debugging tool if things don't appear to work as they should.

Successful Insert

```
FINEST|199857/0||23-04-21 15:46:22|15:46:22.951 [TahuHostCallback--3deac7a5] INFO  o.e.tahu.host.
TahuPayloadHandler - Handling NBIRTH from My MQTT Group/Edge Node ee38b1
FINEST|199857/0||23-04-21 15:46:22|15:46:22.953 [TahuHostCallback--3deac7a5] INFO  o.e.t.host.manager.
SparkplugEdgeNode - Edge Node My MQTT Group/Edge Node ee38b1 set online at Fri Apr 21 15:46:22 UTC 2023
FINEST|199857/0||23-04-21 15:46:23|15:46:23.072 [TahuHostCallback--3deac7a5] INFO  o.e.tahu.host.
TahuPayloadHandler - Handling DBIRTH from My MQTT Group/Edge Node ee38b1/PLC 1
FINEST|199857/0||23-04-21 15:46:23|15:46:23.075 [TahuHostCallback--3deac7a5] INFO  o.e.t.host.manager.
SparkplugDevice - Device My MQTT Group/Edge Node ee38b1/PLC 1 set online at Fri Apr 21 15:46:22 UTC 2023
FINEST|199857/0||23-04-21 15:46:23|15:46:23.759 [ingest-flush-thread] INFO  n.s.i.s.internal.FlushService -
[SF_INGEST] buildAndUpload task added for client=MY_CLIENT, blob=2023/4/21/15/46
/rth2hb_eSKU3AAtxudYKnPFztPjrokzP29ZXzv5JFbbj0YUnqUUCC_1049_48_1.bdec, buildUploadWorkers stats=java.util.
concurrent.ThreadPoolExecutor@32321763[Running, pool size = 2, active threads = 1, queued tasks = 0, completed
tasks = 1]
FINEST|199857/0||23-04-21 15:46:23|15:46:23.774 [ingest-build-upload-thread-1] INFO  n.s.i.i.a.h.io.compress.
CodecPool - Got brand-new compressor [.gz]
FINEST|199857/0||23-04-21 15:46:23|15:46:23.822 [ingest-build-upload-thread-1] INFO  n.s.i.streaming.internal.
BlobBuilder - [SF_INGEST] Finish building chunk in blob=2023/4/21/15/46
/rth2hb_eSKU3AAtxudYKnPFztPjrokzP29ZXzv5JFbbj0YUnqUUCC_1049_48_1.bdec, table=CL_BRIDGE_STAGE_DB.STAGE_DB.
SPARKPLUG_RAW, rowCount=2, startOffset=0, uncompressedSize=5888, compressedChunkLength=5872,
encryptedCompressedSize=5888, bdecVersion=THREE
FINEST|199857/0||23-04-21 15:46:23|15:46:23.839 [ingest-build-upload-thread-1] INFO  n.s.i.s.internal.
FlushService - [SF_INGEST] Start uploading file=2023/4/21/15/46
/rth2hb_eSKU3AAtxudYKnPFztPjrokzP29ZXzv5JFbbj0YUnqUUCC_1049_48_1.bdec, size=5888
FINEST|199857/0||23-04-21 15:46:24|15:46:24.132 [ingest-build-upload-thread-1] INFO  n.s.i.s.internal.
FlushService - [SF_INGEST] Finish uploading file=2023/4/21/15/46
/rth2hb_eSKU3AAtxudYKnPFztPjrokzP29ZXzv5JFbbj0YUnqUUCC_1049_48_1.bdec, size=5888, timeInMillis=292
FINEST|199857/0||23-04-21 15:46:24|15:46:24.148 [ingest-register-thread] INFO  n.s.i.s.internal.RegisterService
- [SF_INGEST] Start registering blobs in client=MY_CLIENT, totalBlobListSize=1, currentBlobListSize=1, idx=1
FINEST|199857/0||23-04-21 15:46:24|15:46:24.148 [ingest-register-thread] INFO  n.s.i.s.i.
SnowflakeStreamingIngestClientInternal - [SF_INGEST] Register blob request preparing for blob=[2023/4/21/15/46
/rth2hb_eSKU3AAtxudYKnPFztPjrokzP29ZXzv5JFbbj0YUnqUUCC_1049_48_1.bdec], client=MY_CLIENT, executionCount=0
FINEST|199857/0||23-04-21 15:46:24|15:46:24.301 [ingest-register-thread] INFO  n.s.i.s.i.
SnowflakeStreamingIngestClientInternal - [SF_INGEST] Register blob request returned for blob=[2023/4/21/15/46
/rth2hb_eSKU3AAtxudYKnPFztPjrokzP29ZXzv5JFbbj0YUnqUUCC_1049_48_1.bdec], client=MY_CLIENT, executionCount=0
```

Data will also be visible in Snowflake at this point. See below for an example. By changing data values in the UDT tags in Ignition DDATA Sparkplug messages will be produced. Every time the Edge Node connects, it will produce NBIRTH and DBIRTH messages. All of these will now appear in Snowflake with their values, timestamps, and qualities



	MSG_ID	NAMESPACE	GROUP_ID	MESSAGE_TYPE	EDGE_NOD
1	03f10e04-d2e1-4c9d-b11e-b14ad12ab464	spBv1.0	My MQTT Group	NDEATH	Edge No
2	06311eb1-09fa-4f97-940a-02252bf20149	spBv1.0	My MQTT Group	DDATA	Edge No
3	0d3085e8-9fa9-45bf-9cee-c6a95387dc59	spBv1.0	My MQTT Group	DDATA	Edge No
4	52cce200-1dfa-4863-9f9f-022f16682157	spBv1.0	My MQTT Group	DDATA	Edge No
5	2964b130-0c38-49f3-a1f8-85ecc9ca7576	spBv1.0	My MQTT Group	DDATA	Edge No
6	e933bee2-5c3f-4723-ab16-2dbc772709dc	spBv1.0	My MQTT Group	DDATA	Edge No
7	92eae2cb-196d-4776-90f6-51ea2ab8de24	spBv1.0	My MQTT Group	NBIRTH	Edge No
8	56d05c27-f117-4fde-bb4b-628d27690ad7	spBv1.0	My MQTT Group	DBIRTH	Edge No
9	c3c14169-39fc-4913-9466-17910b052eb0	spBv1.0	My MQTT Group	DDATA	Edge No
10	9efcd23f-69db-418c-b554-f75cfbde6af5	spBv1.0	My MQTT Group	DDATA	Edge No
11	fc12f59f-90fe-4208-be74-4c573c080416	spBv1.0	My MQTT Group	NBIRTH	Edge No
12	81c93ef9-d8d0-499b-91ce-21204ea2c116	spBv1.0	My MQTT Group	DBIRTH	Edge No
13	c81b77bd-8b18-4204-a4a1-b281d14546fd	spBv1.0	My MQTT Group	NDEATH	Edge No
14	e0ea719f-ff70-4743-969b-f2458899876d	spBv1.0	My MQTT Group	NDEATH	Edge No
15	d0b4065e-bc52-4f8c-b960-c1e7dad134e7	spBv1.0	My MQTT Group	NBIRTH	Edge No
16	63aa4570-eb56-4d06-92ce-d16a90c89519	spBv1.0	My MQTT Group	DBIRTH	Edge No
17	69cf6a5e-c0ba-4e32-b54f-77c6cc2ac48c	spBv1.0	My MQTT Group	DBIRTH	Edge No
18	7eeff087-2c03-49ed-beba-2ea655721404	spBv1.0	My MQTT Group	DDATA	Edge No

Additional Resources

- Snowflake
- IoT Bridge for Snowflake configuration
- Cirrus Link MQTT Servers
- IoT Bridge for Snowflake FAQ
- Ignition
- MQTT Modules for Ignition