Getting Started: AWS Injector Quick Start

Prerequisites

- Ignition with Google Cloud Injector Module installed
    - Review the Cirrus Link Module Installation documentation for installation details.
- Ignition Designer installed
    - Review the Inductive Automation documentation for Launching Designer against the Ignition gateway
- An existing AWS account with an active Kinesis Stream and/or DynamoDB Database.
    - Documentation on creating a Kinesis Stream can be found here.
        - If using a Firehose stream, its source must be "Direct PUT or other sources"
    - Documentation on creating a DynamoDB Database table can be found here.
        - DynamoDB Database tables ideally should be created with the following settings:
            - Primary partition key: UUID (String)
            - Primary sort key: GROUP_EDGE_ID (String)

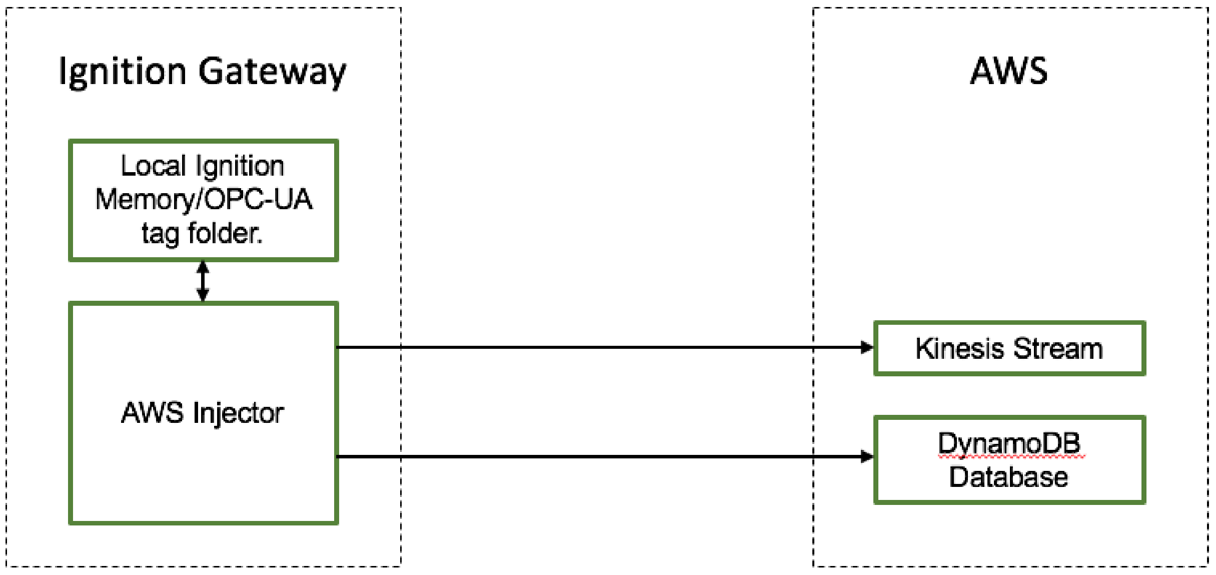
Summary

This tutorial will provide step-by-step instructions for the following:

- Configuring the AWS Injector Module to connect to a Kinesis Stream
- Configuring the AWS Injector Module to connect to a DynamoDB Database
- Publishing live Tag data and events to the connected Kinesis Stream and DynamoDB Database.

Upon completion of this module you will have an Ignition Gateway connected and publishing live Tag data to AWS.

Architecture



Tutorial

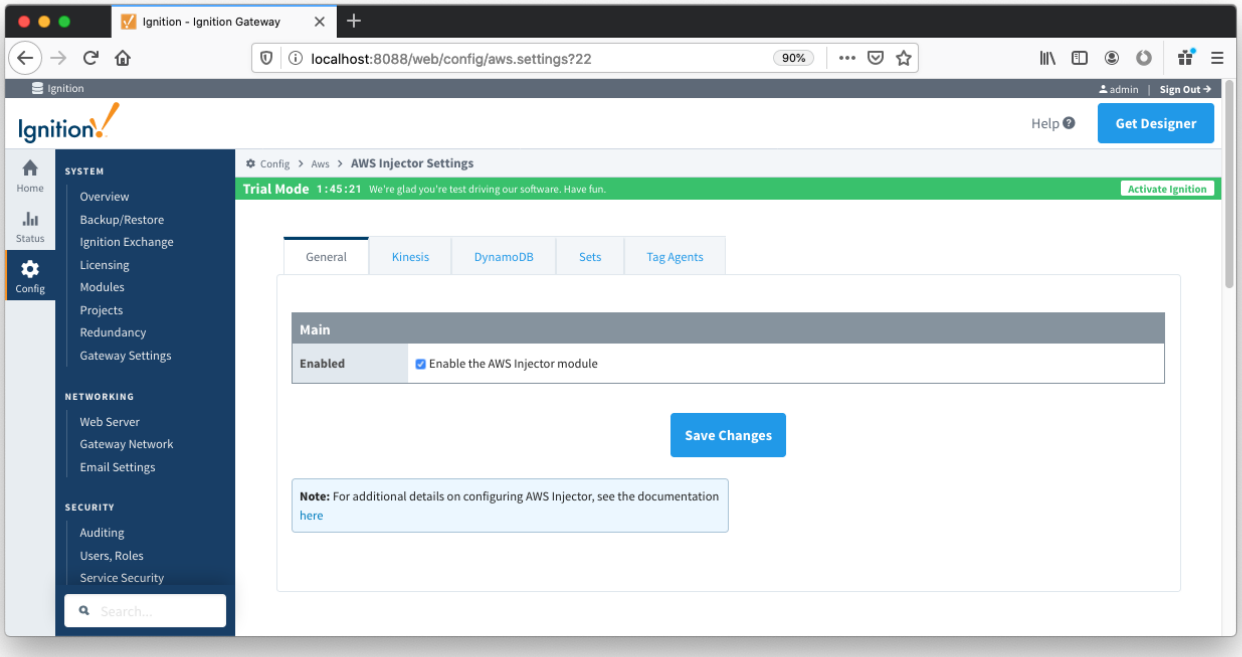
Step 1: Configure the AWS Injector Modules

Once you have Ignition and the AWS Injector Module installed and running we can setup the configuration to connect to AWS.

Review the AWS Injector Module configuration guide for more details on each tab

Navigate to the AWS Injector Module configuration section from the left side bar in the Ignition Gateway.

For this tutorial, we will be adding new Kinesis Stream and DynamoDB Database Settings. You may optionally choose to skip one of the following two sections if you do not wish to setup a Kinesis Stream endpoint or a DynamoDB Database.

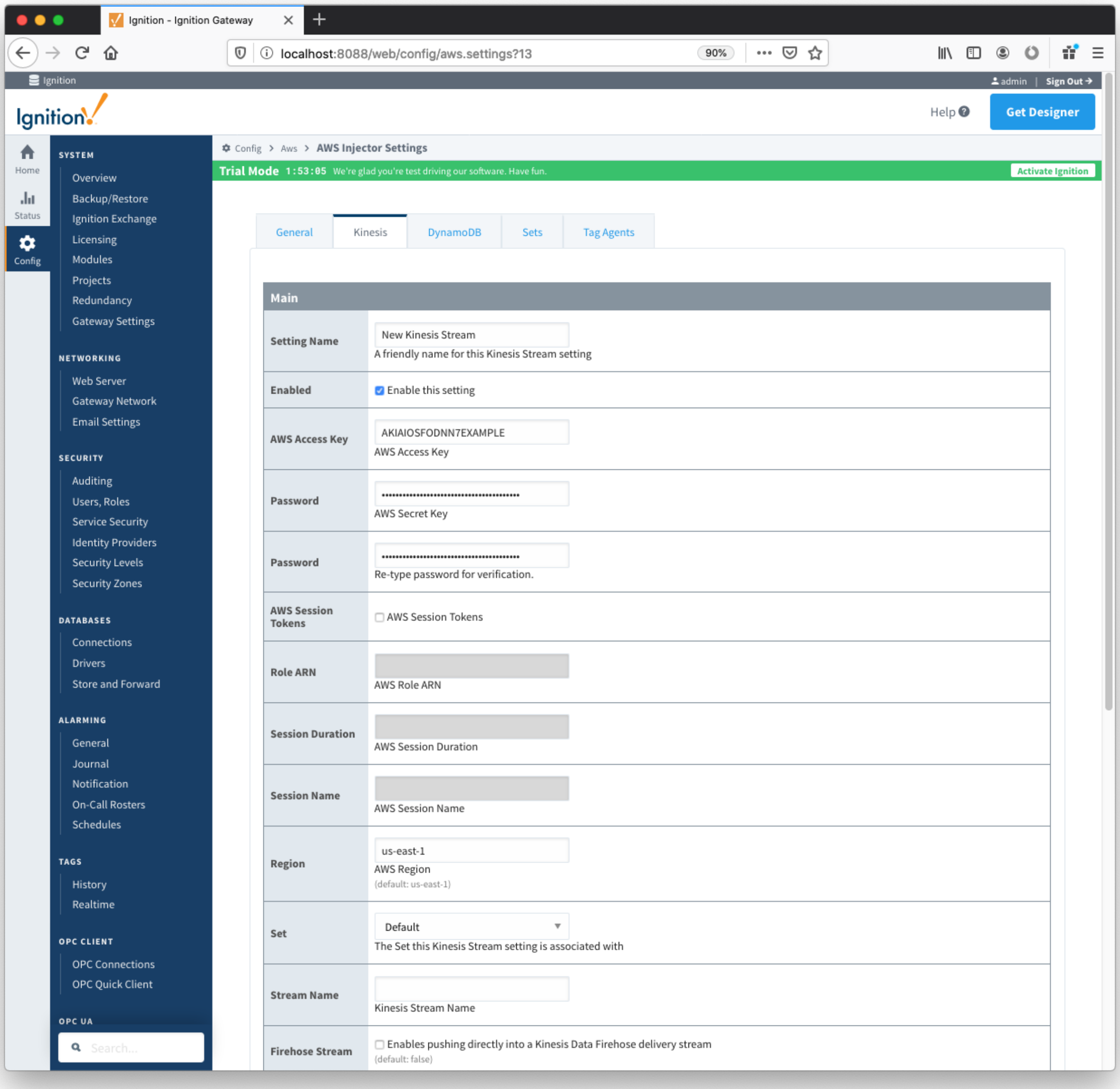


Main	
Enabled	✓ Enable the AWS Injector module

Note: For additional details on configuring AWS Injector, see the documentation here

Create an AWS Kinesis Stream Setting

From the Kinesis tab, click on the "Create new AWS Kinesis Stream Setting..." link to bring up the following configuration form:

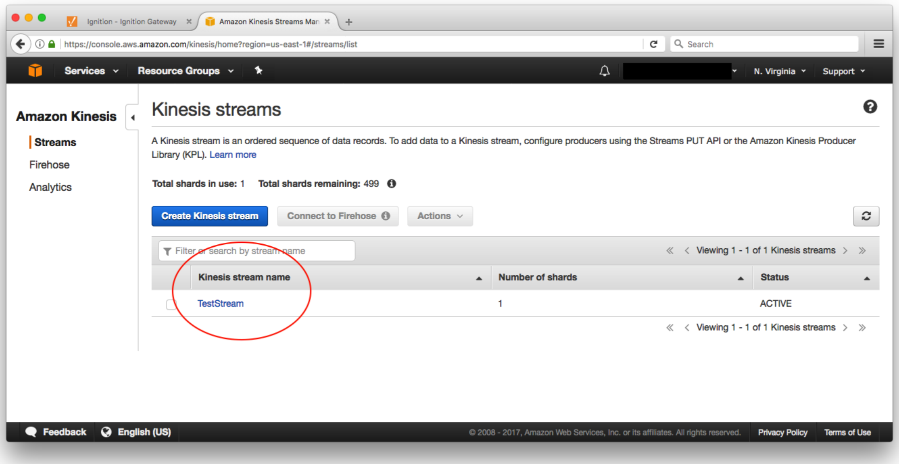


Setting Name	New Kinesis Stream A friendly name for this Kinesis Stream setting
Enabled	✓ Enable this setting
AWS Access Key	AKIAIOSFODNN7EXAMPLE AWS Access Key
Password	•• AWS Secret Key
Password	•• Re-type password for verification.
AWS Session Tokens	☐ AWS Session Tokens
Role ARN	AWS Role ARN
Session Duration	AWS Session Duration
Session Name	AWS Session Name
Region	us-east-1 AWS Region (default: us-east-1)
Set	Default The Set this Kinesis Stream setting is associated with
Stream Name	Kinesis Stream Name
Firehose Stream	☐ Enables pushing directly into a Kinesis Data Firehose delivery stream (default: false)

Set the following parameters:

- Setting Name
  - This can be any unique identifier. For this tutorial we will use "TestSetting".
- AWS Access Key
  - This is the Access Key used to authenticate to your AWS account
- Password
  - This is the Secret Key used to authenticate to your AWS account
- Region
  - This is the AWS Region that your Kinesis Stream is in
- Stream Name

- This is the name of the Kinesis Stream that you wish to connect to. This is shown below in the AWS Kinesis Streams console.


Ignition - Ignition Gateway
Amazon Kinesis Streams Man
https://console.aws.amazon.com/kinesis/home?region=us-east-1#/streams/list
Search
Services
Resource Groups
[redacted]
N. Virginia
Support
Amazon Kinesis
Streams
Firehose
Analytics

Kinesis streams

A Kinesis stream is an ordered sequence of data records. To add data to a Kinesis stream, configure producers using the Streams PUT API or the Amazon Kinesis Producer Library (KPL). Learn more

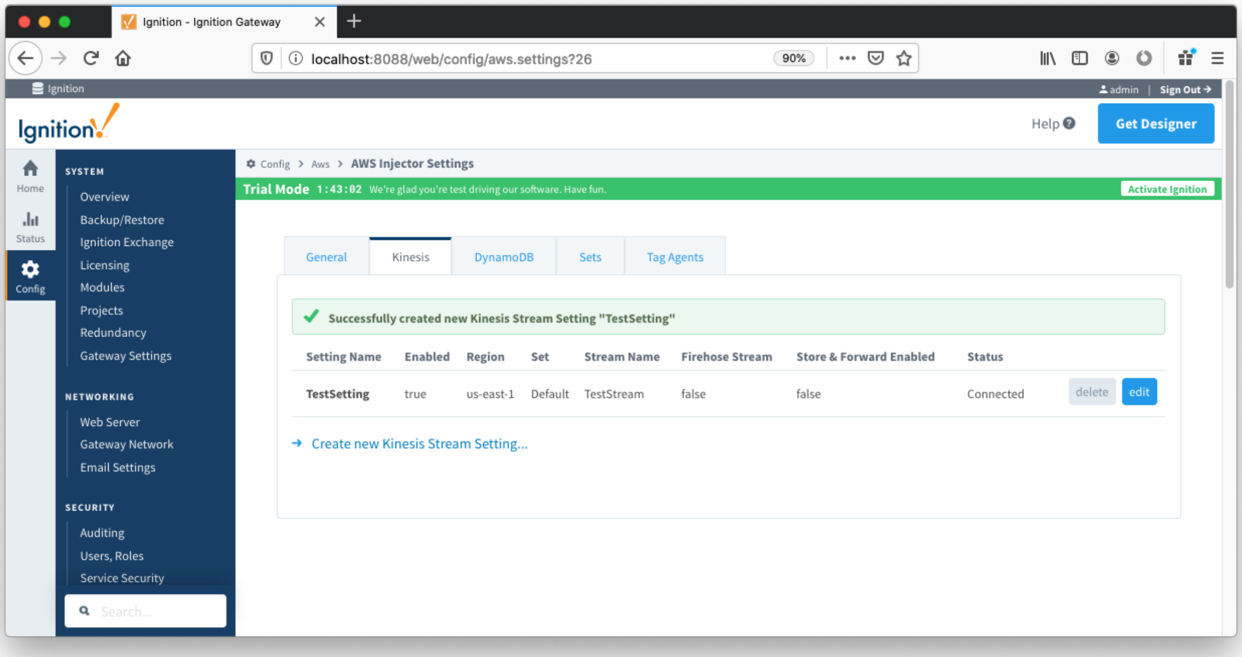
Total shards in use: 1   Total shards remaining: 499

Create Kinesis stream
Connect to Firehose
Actions
Filter or search by stream name
Viewing 1 - 1 of 1 Kinesis streams

Kinesis stream name	Number of shards	Status
TestStream	1	ACTIVE

Viewing 1 - 1 of 1 Kinesis streams
Feedback
English (US)
© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

Click on "Create New Kinesis Stream Setting" to finish creating the new configuration setting.



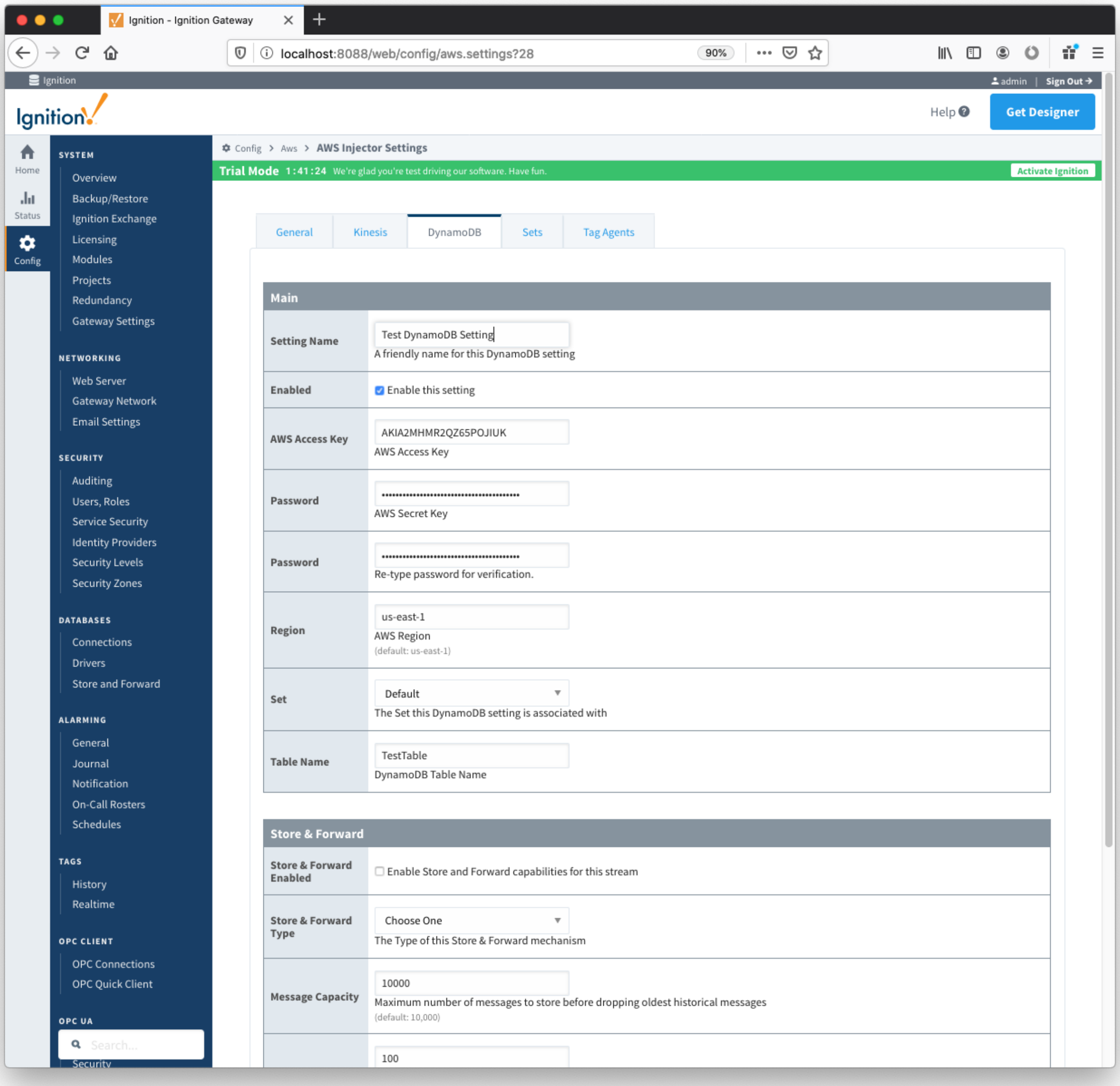
Successfully created new Kinesis Stream Setting "TestSetting"

Setting Name	Enabled	Region	Set	Stream Name	Firehose Stream	Store & Forward Enabled	Status	
TestSetting	true	us-east-1	Default	TestStream	false	false	Connected	delete edit

→ Create new Kinesis Stream Setting...

Create an AWS DynamoDB Database Setting

From the DynamoDB tab, click on the "Create new DynamoDB Setting..." link to bring up the following configuration form:



Main

Setting Name	Test DynamoDB Setting — A friendly name for this DynamoDB setting
Enabled	✓ Enable this setting
AWS Access Key	AKIA2MHMR2QZ65POJIUK — AWS Access Key
Password	•• — AWS Secret Key
Password	•• — Re-type password for verification.
Region	us-east-1 — AWS Region (default: us-east-1)
Set	Default — The Set this DynamoDB setting is associated with
Table Name	TestTable — DynamoDB Table Name

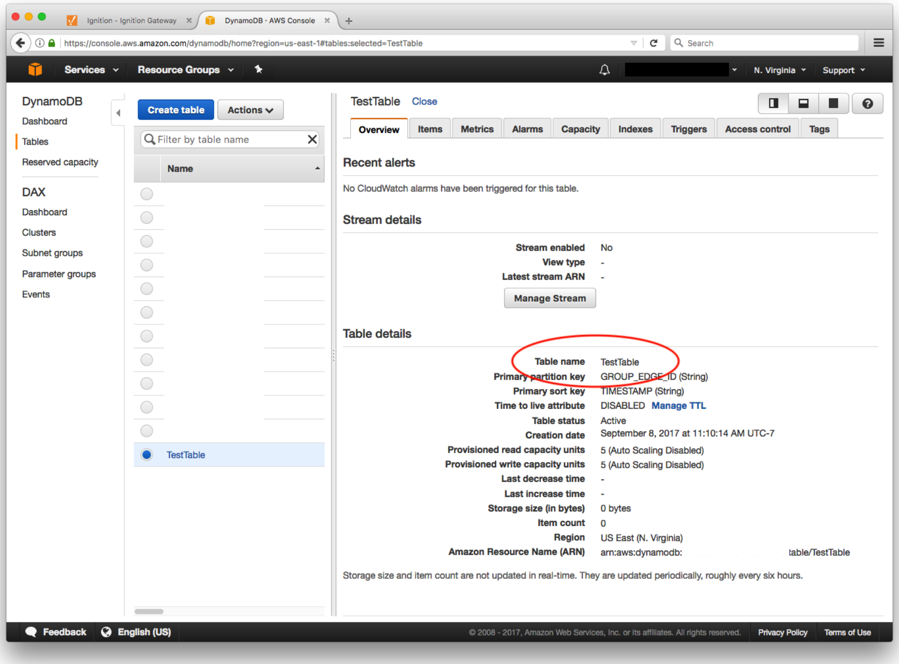
Store & Forward

Store & Forward Enabled	☐ Enable Store and Forward capabilities for this stream
Store & Forward Type	Choose One — The Type of this Store & Forward mechanism
Message Capacity	10000 — Maximum number of messages to store before dropping oldest historical messages (default: 10,000)
	100

Set the following parameters:

- Setting Name
  - This can be any unique identifier. For this tutorial we will use "TestSetting".
  - Note that this only has to be unique among DynamoDB Settings so we can configure with the same used for the Kinesis Stream.
- AWS Access Key
  - This is the Access Key used to authenticate to your AWS account
- Password
  - This is the Secret Key used to authenticate to your AWS account
- Region
  - This is the AWS Region that your DynamoDB Database is in
- Table Name

- This is the name of the DynamoDB Database Table to connect to. This is shown below in the AWS DynamoDB console.



TestTable

Recent alerts

No CloudWatch alarms have been triggered for this table.

Stream details

Stream enabled	No
View type	-
Latest stream ARN	-

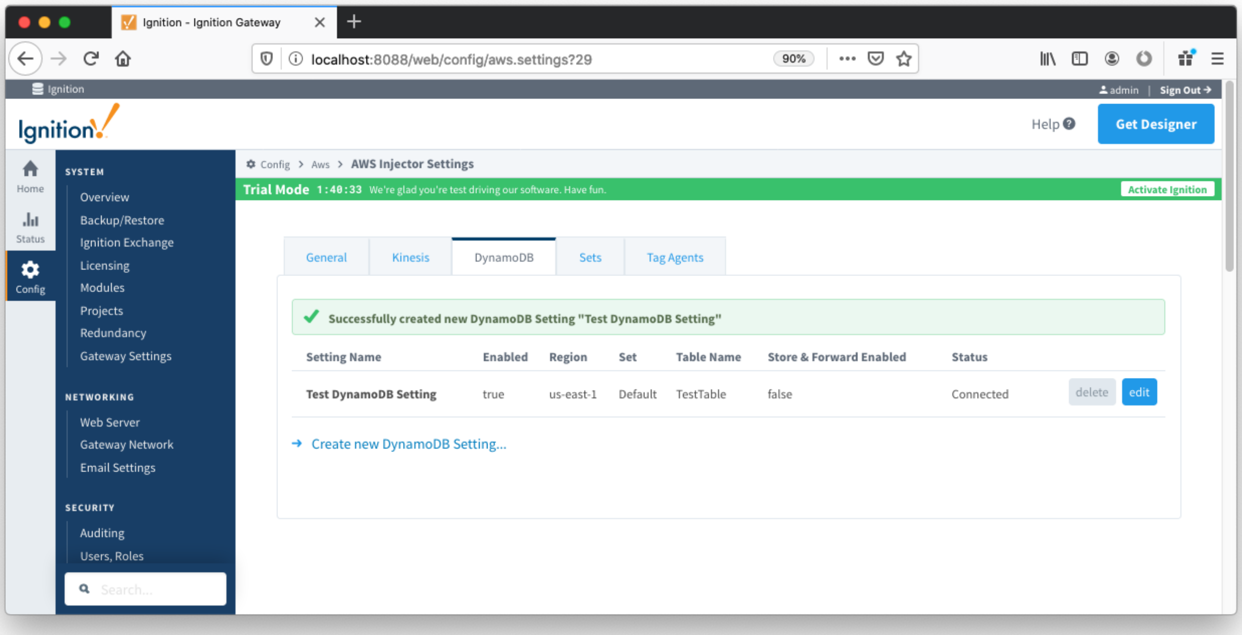
Table details

Table name	TestTable
Primary partition key	GROUP_EDGE_ID (String)
Primary sort key	TIMESTAMP (String)
Time to live attribute	DISABLED Manage TTL
Table status	Active
Creation date	September 8, 2017 at 11:10:14 AM UTC-7
Provisioned read capacity units	5 (Auto Scaling Disabled)
Provisioned write capacity units	5 (Auto Scaling Disabled)
Last decrease time	-
Last increase time	-
Storage size (in bytes)	0 bytes
Item count	0
Region	US East (N. Virginia)
Amazon Resource Name (ARN)	arn:aws:dynamodb: table/TestTable

Storage size and item count are not updated in real-time. They are updated periodically, roughly every six hours.

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Click on "Create New DynamoDB Setting" to finish creating the new configuration setting.



Successfully created new DynamoDB Setting "Test DynamoDB Setting"

Setting Name	Enabled	Region	Set	Table Name	Store & Forward Enabled	Status
Test DynamoDB Setting	true	us-east-1	Default	TestTable	false	Connected

Now the AWS Injector module is connected to a Kinesis Stream and/or a DynamoDB Database, we have to determine if there are are changes needed to the Tag Agent tab to be able to push data.

If you already have Ignition tags defined, for example from the Ignition OPC UA Server, then depending on the depth of your tag tree you may need to configure the Sparkplug Settings.

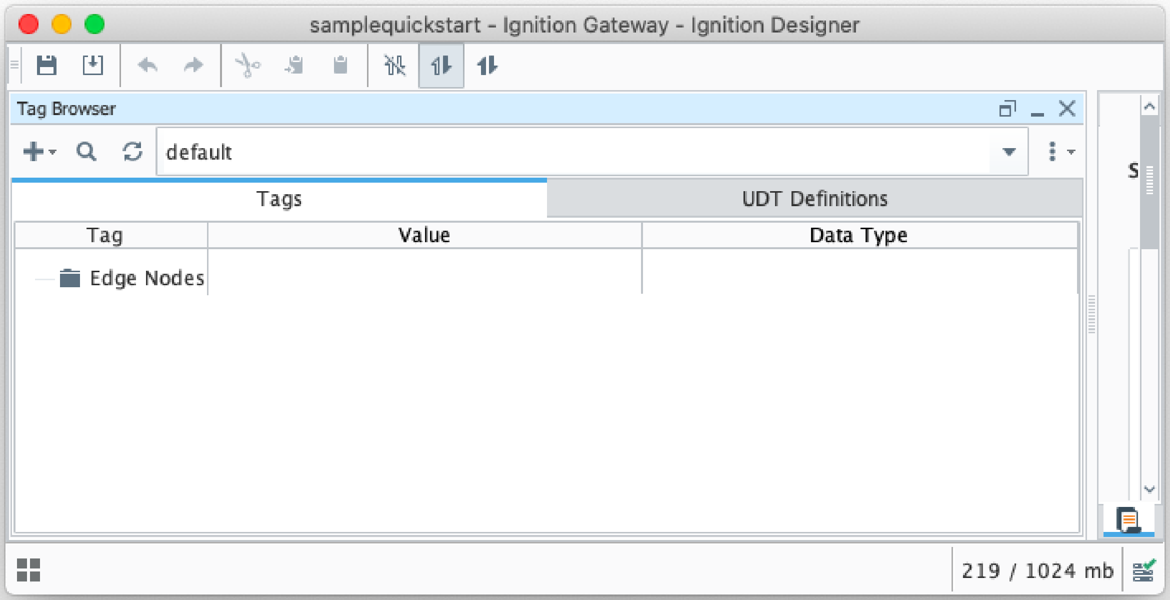
Review the Cloud Injector Tag Agents and Tag Trees document which describes how Cloud Injector Agent configurations interact with Ignition tag trees to push messages and tag change events to the cloud service. It explains how tags get identified to be pushed as well as what specific 'topics' will be included with the messages. It also goes over some example configurations to show how the system will behave in different scenarios.

Once the Tag Agent is setup as needed, you can jump to Step 3: Publishing data.

If you do not have Ignition tags defined we will do that in the next step with a tag tree depth that requires no additional Sparkplug settings.

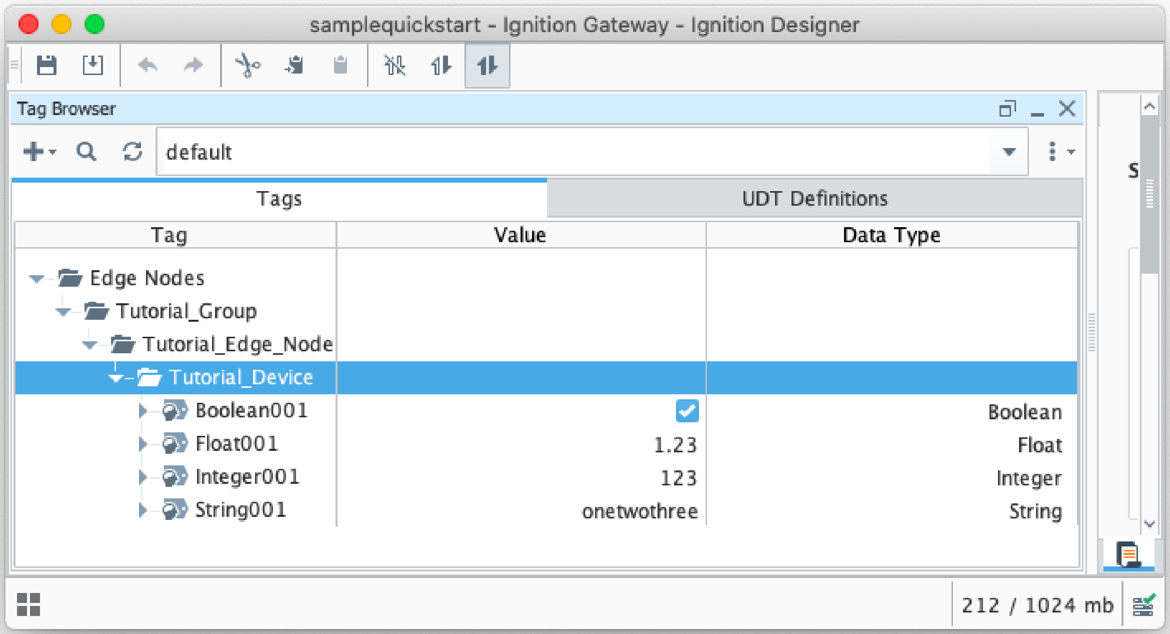
Step 2: Create tags to be published in Designer

When the AWS Injector module is installed in Ignition, an Edge Node folder is automatically created in the 'default' Ignition tag provider.



Create a tree structure under this folder as shown below with some memory tags - this folder structure creates the same hierarchy that is described in the Sparkplug B specification of Group ID, Edge ID, and Device ID.

Refer to the Ignition Tag Browser and Creating Tags documentation for assistance in configuring Ignition tags



samplequickstart - Ignition Gateway - Ignition Designer

Tag Browser

default

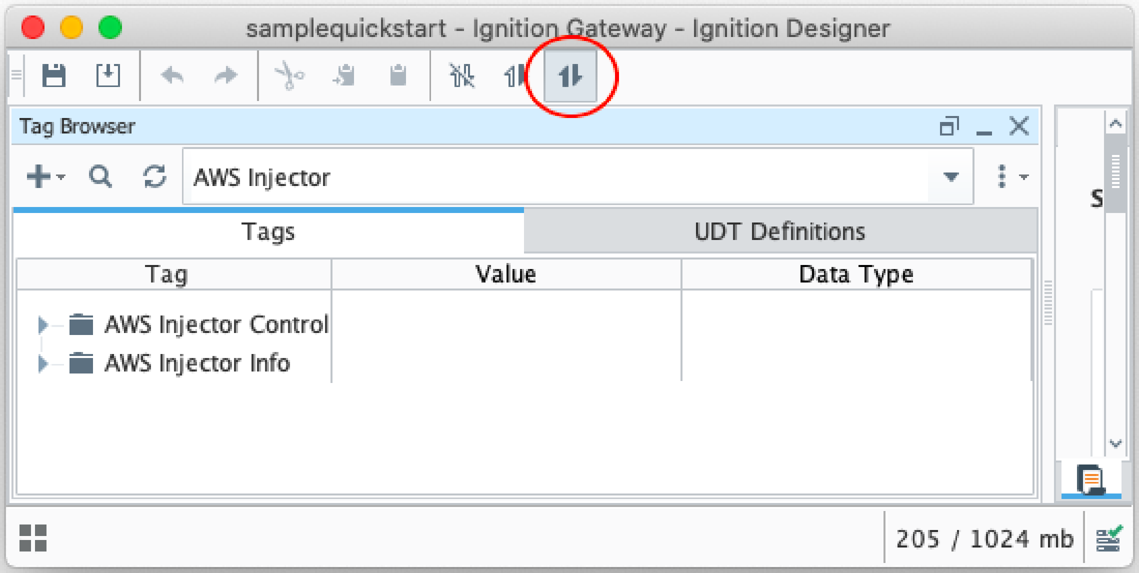
Tag	Value	Data Type
Edge Nodes		
Tutorial_Group		
Tutorial_Edge_Node		
Tutorial_Device		
Boolean001	✓	Boolean
Float001	1.23	Float
Integer001	123	Integer
String001	onetwothree	String

212 / 1024 mb

Step 3: Publishing data

When the AWS Injector module is installed in Ignition, an AWS Injector tag provider is automatically created. This folder will contain both information tags about the module's version and state, as well as control tags for refreshing the module and Tag Agents.

Make sure that the Ignition Designer has read/write communications turned on by selecting the Project/Comm Read/Write button highlighted in the image below.



samplequickstart - Ignition Gateway - Ignition Designer

Tag Browser

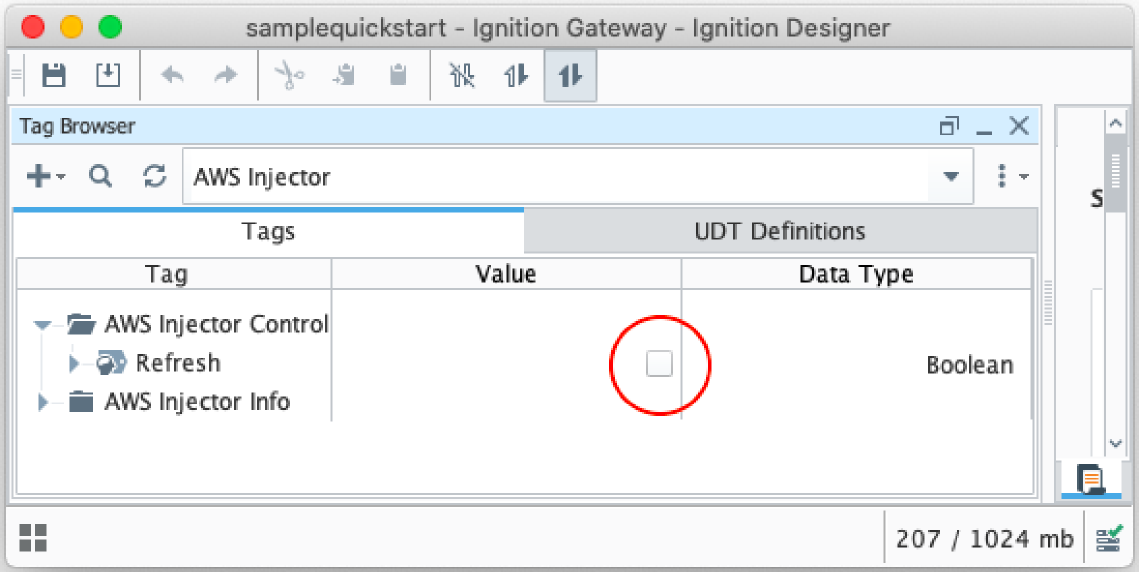
AWS Injector

Tag	Value	Data Type
AWS Injector Control		
AWS Injector Info		

205 / 1024 mb

Review the Inductive Automation Designer documentation for additional assistance on setting the project communication mode

To refresh the default Tag Agent, open the folder "AWS Injector Control" and click on the Refresh Boolean. When this happens, the Tag Agent will scan the "Edge Nodes" folder and find the new Memory Tags that we have created, construct messages representing those tags with their current values and send the messages to the Kinesis Stream and/or DynamoDB Database that we have configured.



Tag	Value	Data Type
AWS Injector Control		
Refresh	☐	Boolean
AWS Injector Info		

The Boolean tag will not change to true. This is really a one-shot and as a result, the tag will not change to true.

Sending tag data to the Kinesis Stream

The AWS Injector Tag Agent will push two JSON messages to the Kinesis Stream. The format of these messages closely follows the Sparkplug B Specification's payload structure.

The first message represents the Edge Node and will contain the following:

- The Sparkplug elements: Namespace, Group ID, Edge Node ID. They will be grouped under "topic".
- A "timestamp" for when the payload was constructed.
- A "bdSeq" sequence number to track the "session" of the Tag Agent.
- Any Edge Node tags defined in the "Tutorial Edge Node" folder (in our example we have none).

It will look something like this:

First Payload

```
{
  "topic": {
    "namespace": "spBv1.0",
    "groupId": "Tutorial Group",
    "edgeNodeId": "Tutorial Edge Node"
  },
  "payload": {
    "timestamp": 1504739061495,
    "metrics": [
      {
        "name": "bdSeq",
        "timestamp": 1504739061495,
        "dataType": "Int64",
        "value": 0
      }
    ],
    "seq": 0
  }
}
```

The second message represents the Device and will contain the following:

- The Sparkplug elements: Namespace, Group ID, Edge Node ID, Device ID. They will be grouped under "topic".
- A "timestamp" for when the payload was constructed.
- A "bdSeq" sequence number to track the "session" of the Tag Agent.
- Any Device tags defined in the "Tutorial Device" folder. It will look something like this:

It will look something like this:

Second Payload

```
{
  "topic": {
    "namespace": "spBv1.0",
    "groupId": "Tutorial Group",
    "edgeNodeId": "Tutorial Edge Node",
    "deviceId": "Tutorial Device"
  },
  "payload": {
    "timestamp": 1504739061501,
    "metrics": [
      {
        "name": "Boolean001",
        "timestamp": 1504739061546,
        "dataType": "Boolean",
        "properties": {
          "Quality": {
            "type": "Int32",
            "value": 192
          }
        },
        "value": true
      },
      {
        "name": "String001",
        "timestamp": 1504739061546,
        "dataType": "String",
        "properties": {
          "Quality": {
            "type": "Int32",
            "value": 192
          }
        },
        "value": "onetwothree"
      },
      {
        "name": "Integer001",
        "timestamp": 1504739061546,
        "dataType": "Int32",
        "properties": {
          "Quality": {
            "type": "Int32",
            "value": 192
          }
        },
        "value": 123
      },
      {
        "name": "Float001",
        "timestamp": 1504739061546,
        "dataType": "Float",
        "properties": {
          "Quality": {
            "type": "Int32",
            "value": 192
          }
        },
        "value": 1.23
      }
    ],
    "seq": 1
  }
}
```

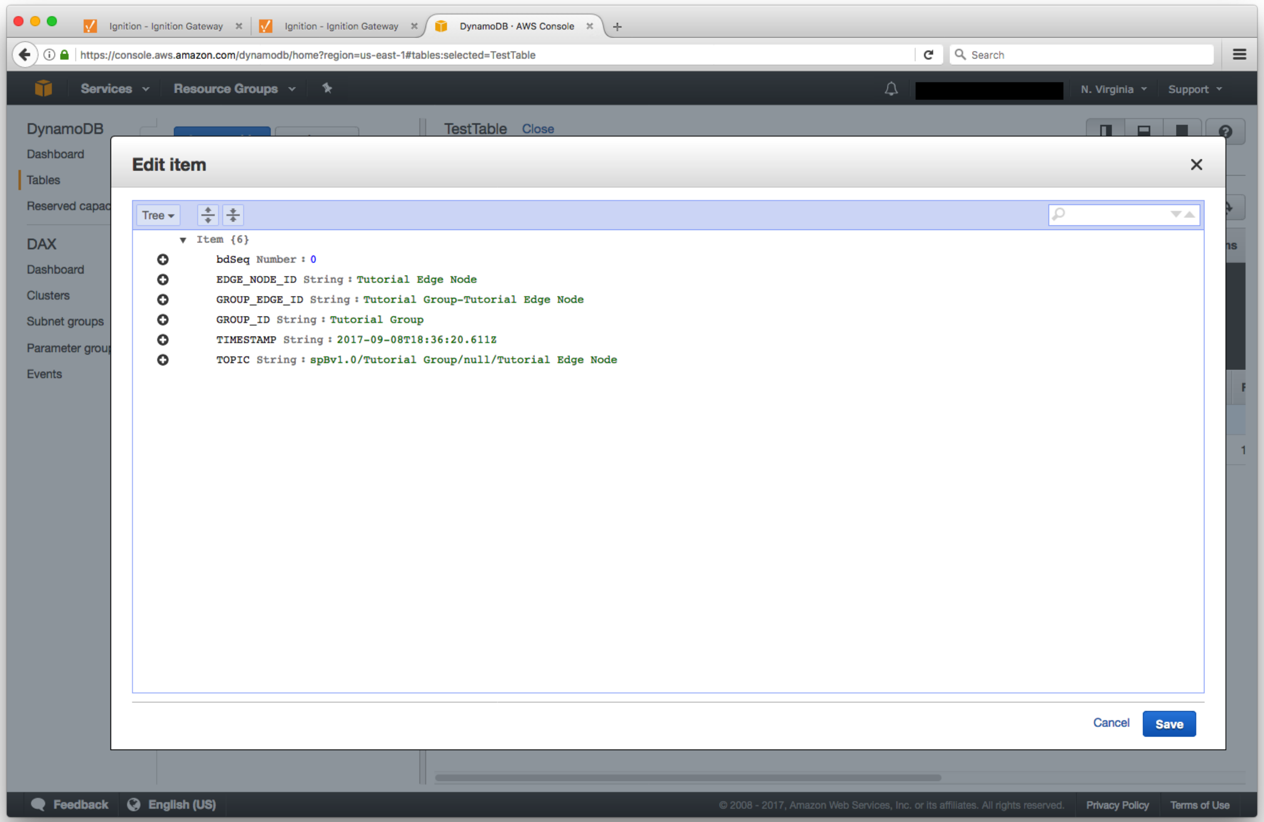
Sending tag data to the DynamoDB Table

The AWS Injector Tag Agent will insert two items into the DynamoDB Database table. The columns of these items represent different elements of the Sparkplug B Specification's payload structure.

The first item represents the Edge Node and will contain the following:

- EDGE_NODE_ID
  - The Sparkplug Edge Node ID.
- GROUP_ID
  - The Sparkplug Group ID.
- GROUP_EDGE_ID
  - A combination of the Group ID and Edge Node ID, (used as the Primary Partition Key).
- TIMESTAMP
  - A "timestamp" for when the payload was constructed
- TOPIC
  - The Sparkplug topic.
- bdSeq
  - A sequence number to track the "session" of the Tag Agent.
- Any Edge Node tags defined in the "Tutorial Edge Node" folder (in our example we have none).

It will look something like this:

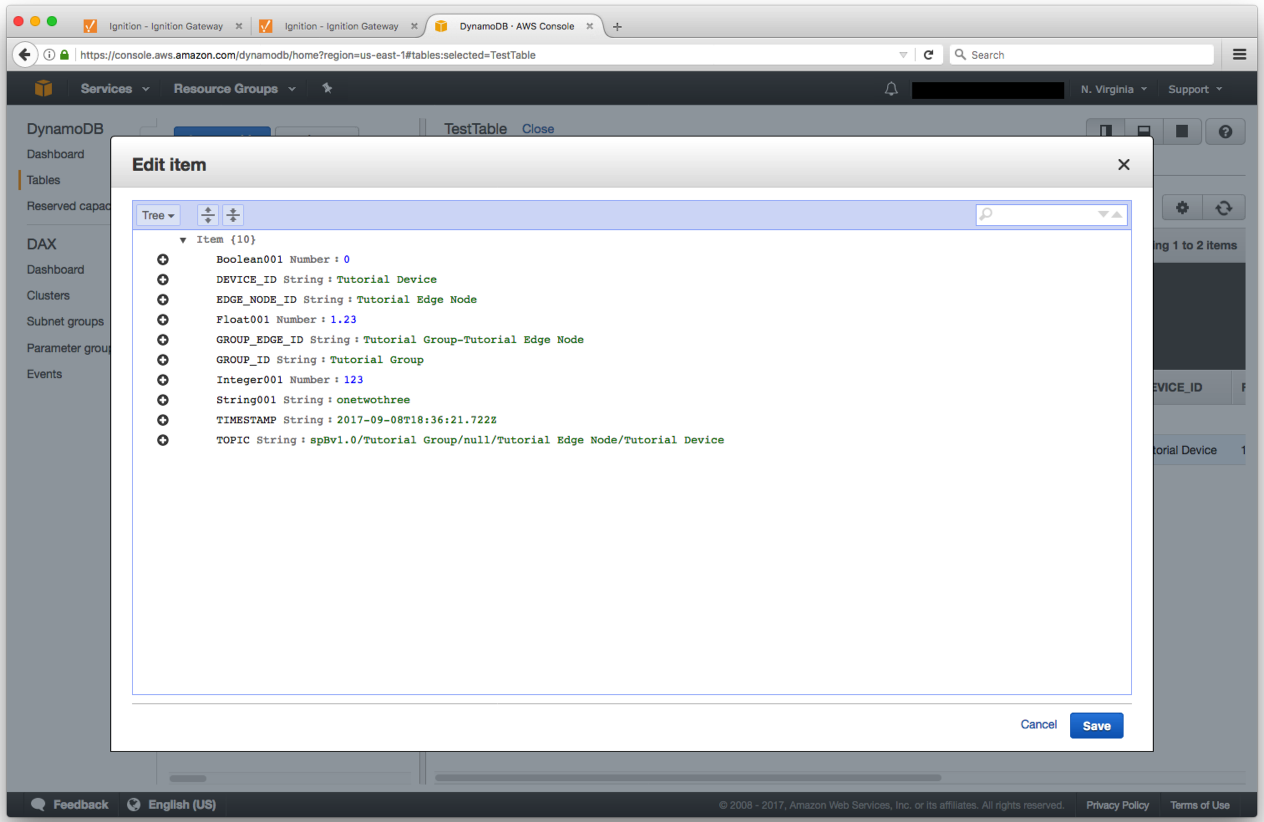


```
▼ Item {6}
    bdSeq Number : 0
    EDGE_NODE_ID String : Tutorial Edge Node
    GROUP_EDGE_ID String : Tutorial Group-Tutorial Edge Node
    GROUP_ID String : Tutorial Group
    TIMESTAMP String : 2017-09-08T18:36:20.611Z
    TOPIC String : spBv1.0/Tutorial Group/null/Tutorial Edge Node
```

The second item represents the Device and will contain the following:

- DEVICE_ID
  - The Sparkplug Device ID.
- EDGE_NODE_ID
  - The Sparkplug Edge Node ID.
- GROUP_ID
  - The Sparkplug Group ID.
- GROUP_EDGE_ID
  - A combination of the Group ID and Edge Node ID, (used as the Primary Partition Key).
- TIMESTAMP
  - A "timestamp" for when the payload was constructed
- TOPIC
  - The Sparkplug topic.
- Any Device tags defined in the "Tutorial Device" folder
  - Boolean001
  - Float001
  - Integer001
  - String001

It will look something like this:



[redacted]

Edit item

```
▼ Item {10}
    Boolean001 Number : 0
    DEVICE_ID String : Tutorial Device
    EDGE_NODE_ID String : Tutorial Edge Node
    Float001 Number : 1.23
    GROUP_EDGE_ID String : Tutorial Group-Tutorial Edge Node
    GROUP_ID String : Tutorial Group
    Integer001 Number : 123
    String001 String : onetwothree
    TIMESTAMP String : 2017-09-08T18:36:21.722Z
    TOPIC String : spBv1.0/Tutorial Group/null/Tutorial Edge Node/Tutorial Device
```

Step 4: Use Ignition Designer to Publish Tag Data (Live Tag Values Changes)

Now we can change the values of the new Memory Tags and generate messages that contain the Tag change events. Click on the value of the "Boolean001" Memory Tag to change it's value.

Sending Live Tag Data to the Kinesis Stream

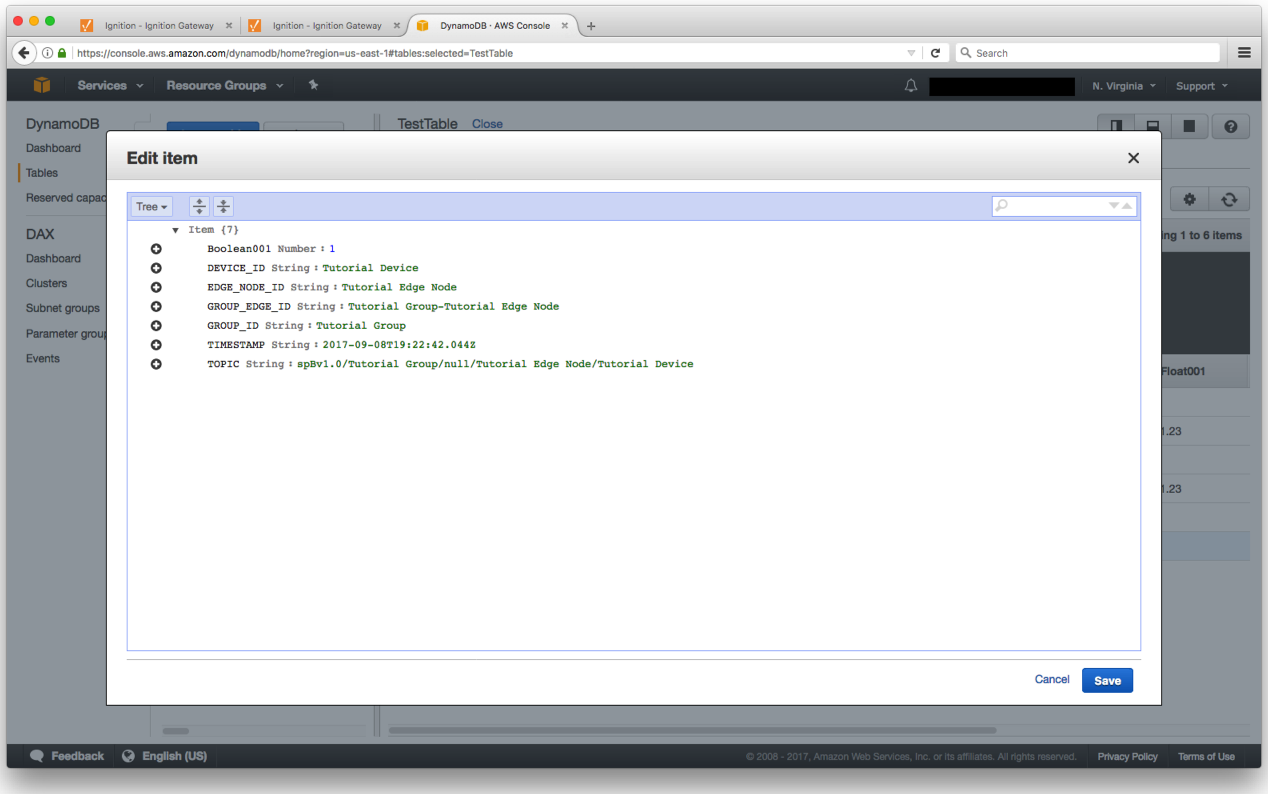
The following message will be constructed to represent this Tag change event and pushed to the Kinesis Stream:

Change Event Payload

```
{
  "topic": {
    "namespace": "spBv1.0",
    "groupId": "Tutorial Group",
    "edgeNodeId": "Tutorial Edge Node",
    "deviceId": "Tutorial Device"
  },
  "payload": {
    "timestamp": 1504740884529,
    "metrics": [
      {
        "name": "Boolean001",
        "timestamp": 1504740883526,
        "dataType": "Boolean",
        "value": false
      }
    ],
    "seq": 2
  }
}
```

Sending Live Tag Data to the DynamoDB Database

The following item will be constructed to represent this Tag change event and inserted into the DynamoDB Database table



```
▼ Item {7}
    Boolean001 Number : 1
    DEVICE_ID String : Tutorial Device
    EDGE_NODE_ID String : Tutorial Edge Node
    GROUP_EDGE_ID String : Tutorial Group-Tutorial Edge Node
    GROUP_ID String : Tutorial Group
    TIMESTAMP String : 2017-09-08T19:22:42.044Z
    TOPIC String : spBv1.0/Tutorial Group/null/Tutorial Edge Node/Tutorial Device
```

Step 5: AWS Applications

It is beyond the scope of this tutorial to show how to design an application in AWS to handle the data flowing into the Kinesis Stream and/or DynamoDB Database. For additional information on developing applications to consume this data see https://aws.amazon.com/.

Additional Resources

- Inductive Automation's Ignition download with free trial
  - Current Ignition Release
- Cirrus Link Solutions Modules for Ignition
  - Ignition Strategic Partner Modules
- Support questions
  - Check out the Cirrus Link Forum: https://forum.cirrus-link.com/
  - Contact support: support@cirrus-link.com
- Sales questions
  - Email: sales@cirrus-link.com
  - Phone: +1 (844) 924-7787
- About Cirrus Link
  - https://www.cirrus-link.com/about-us/