Getting Started: AWS Injector Quick Start

Prerequisites

- Knowledge of Ignition and Module installation process: Cloud and MQTT Module Installation.
- An existing AWS account with an active Kinesis Stream and/or DynamoDB Database.
    - Documentation on creating a Kinesis Stream can be found here.
    - 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: GROUP_EDGE_ID (String)
        - Primary sort key: TIMESTAMP (String)

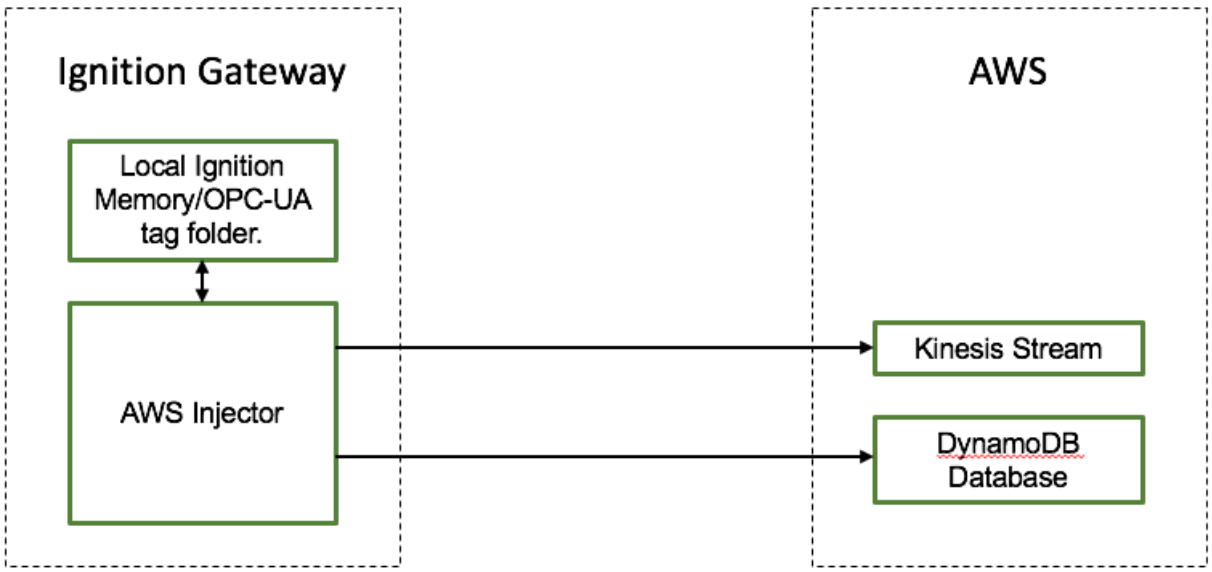
Summary

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

- Installing the Ignition Gateway Industrial Application Platform with the AWS Injector Module
- 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: Download and Install Ignition

Ignition is an Industrial Application Platform that can be used to create SCADA and HMI solutions. A fully functional Ignition system can be downloaded and run in trial mode. Using Ignition as a tool in this way, we can install the Sparkplug MQTT Modules and observe everything working.

Go to the Inductive Automation download page and download the desired version (select version from the 'Ignition Version' dropdown) of the Ignition installer for Windows, Linux or MacOS;
https://inductiveautomation.com/downloads/archive

Once the Ignition installer has been downloaded, follow the instructions provided by Inductive Automation to install and startup Ignition.

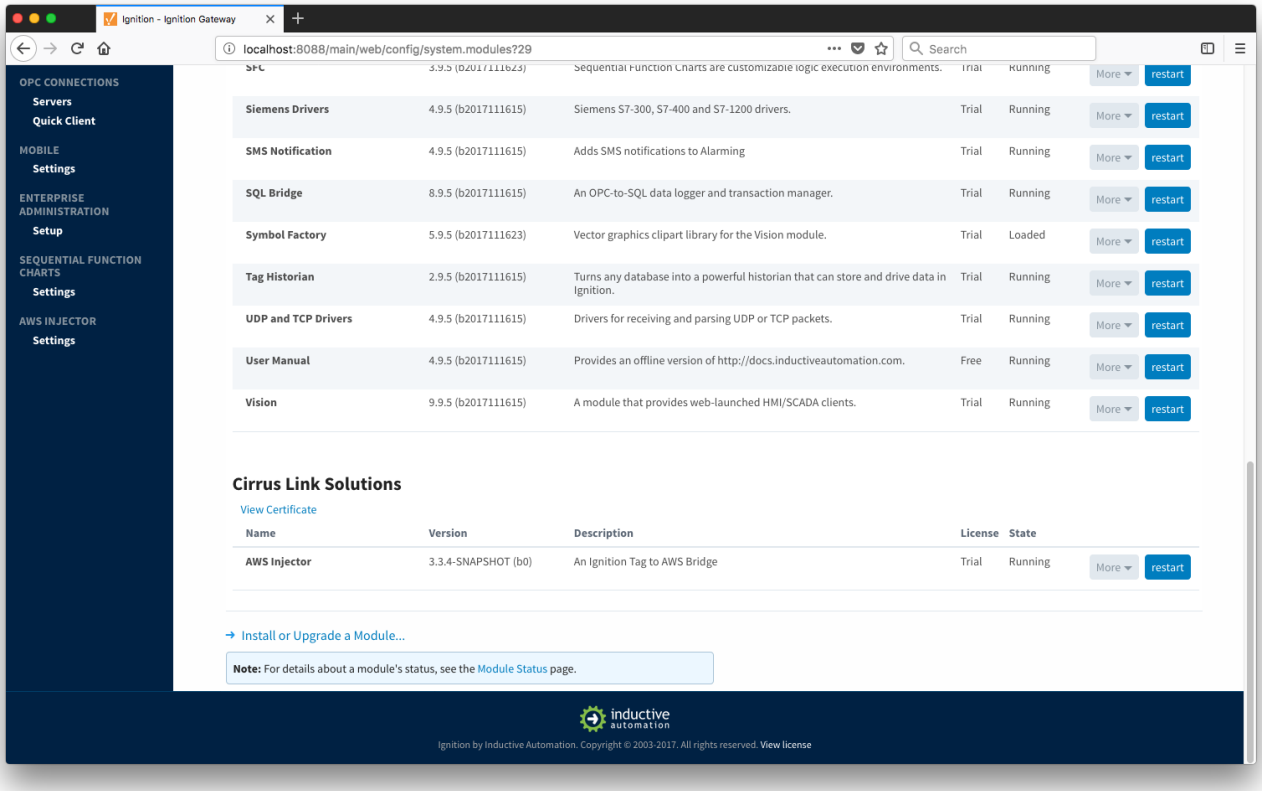
Step 2: Download and Install the Cirrus Link AWS Injector Module

Go to the Inductive Automation download page again and scroll down to the Third Party modules section. Find the Cirrus Link modules section and download the AWS Injector Module.
https://inductiveautomation.com/downloads/archive. The download links should look similar to what is shown below.

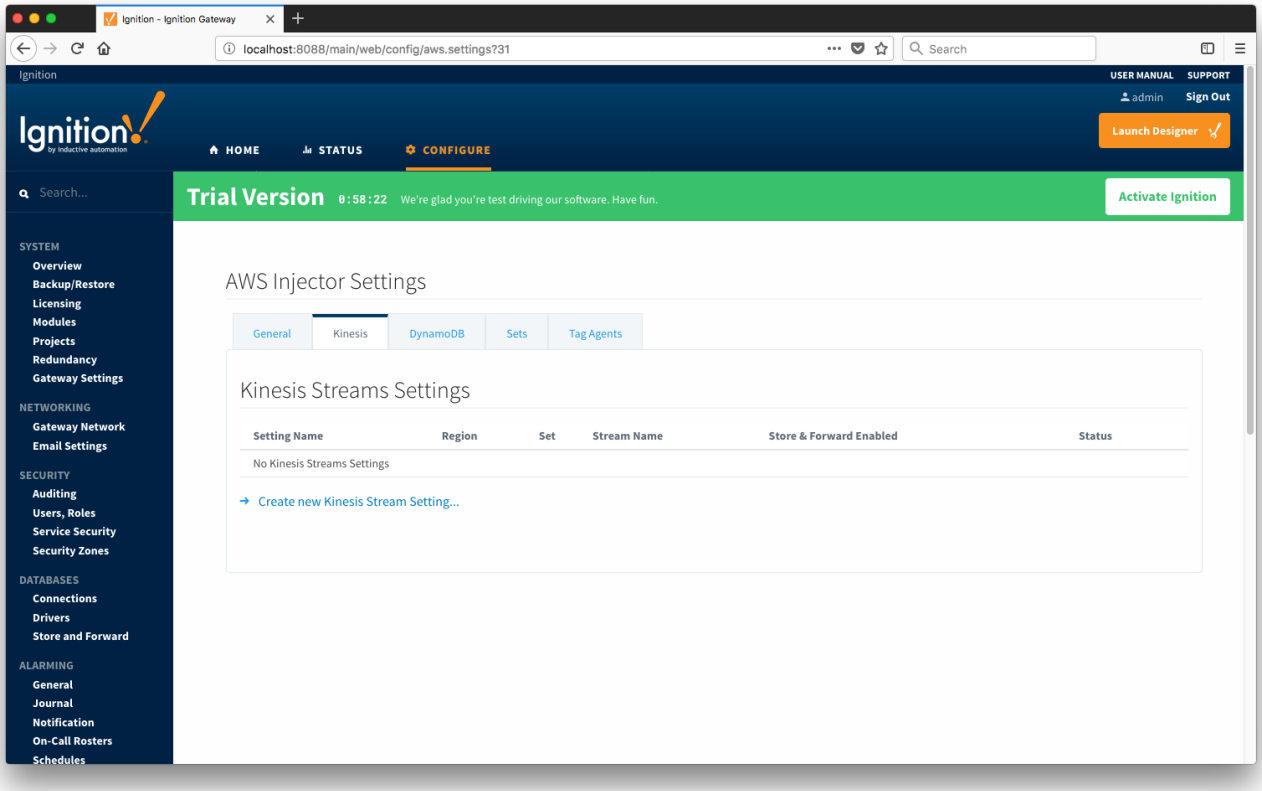
(placeholder for image)

Step 3: Configure the MQTT Modules

Once you have Ignition installed and running, and the AWS Injector module downloaded, browse to the Ignition Gateway console (e.g. http://localhost:8088). Login using the default credentials of admin/password. Click on Configuration tab and then click on the Modules tab on the left side of the page. Scroll to the bottom of the Modules section and click on the Download/Upgrade modules button. When prompted, select the AWS Injector module from the file browser and install it. When complete, the Ignition Gateway Web UI module section should look similar to what is shown below:



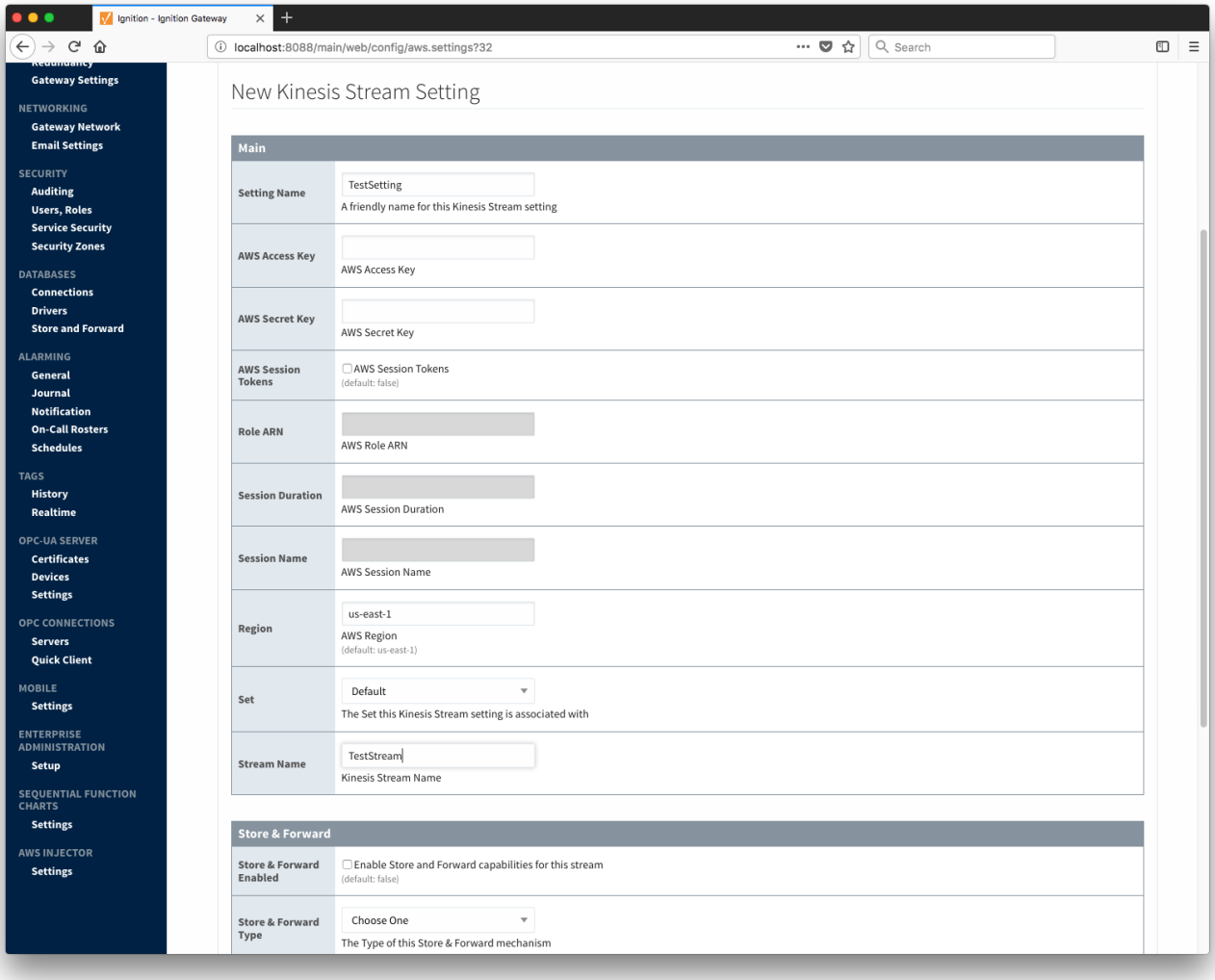
Select the "AWS INJECTOR" "Settings" link on the lower left of the page to navigate to the AWS Injector Module's configuration page.



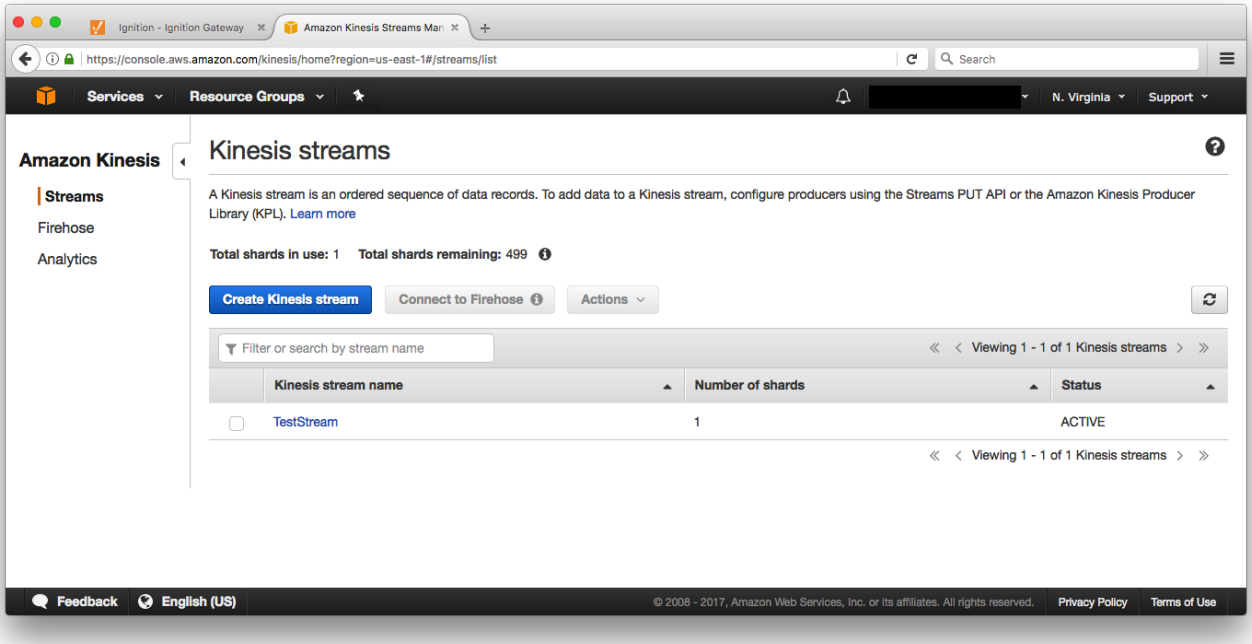
A detailed explanation of each configuration tab can be found here. For this tutorial, we will be adding new Kinesis Stream and DynamoDB Database Settings. You may optionally choose to skip one of the following two sections if you do not wish to setup a Kinesis Stream endpoint or a DynamoDB Database.

Create an AWS Kinesis Stream Setting

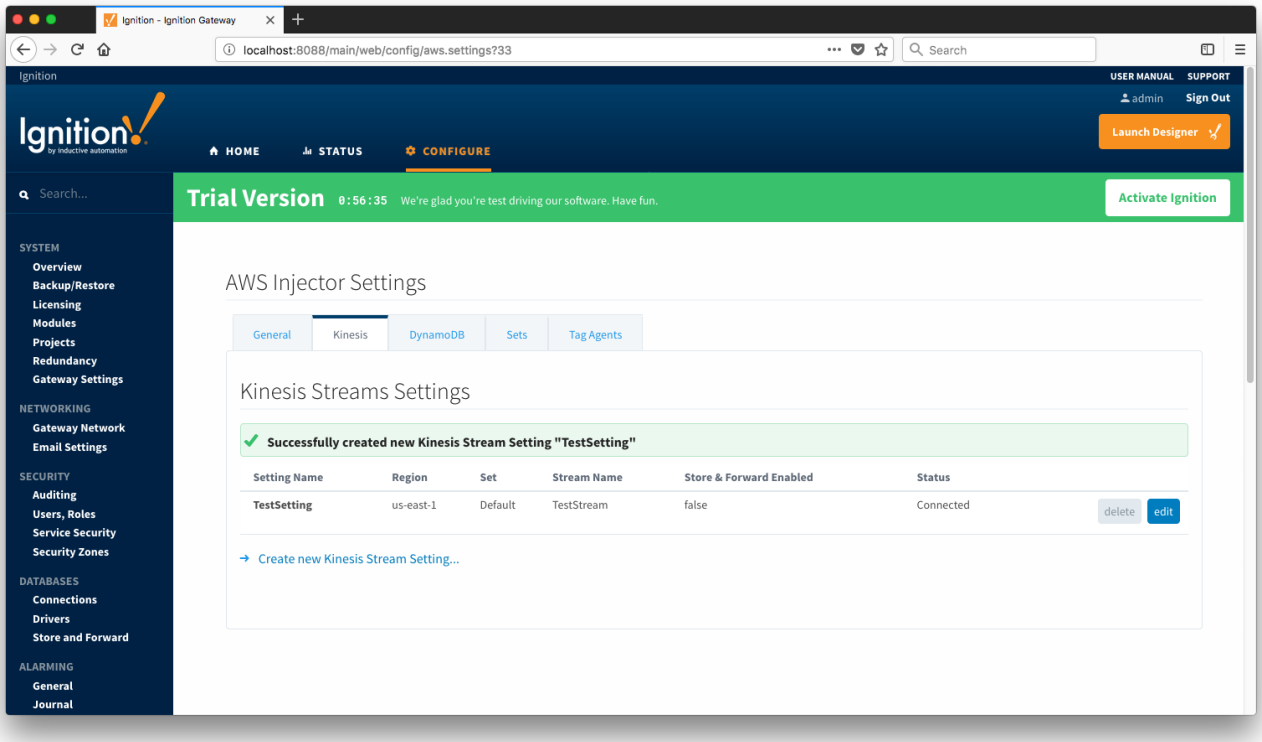
With the Kinesis tab selected, Click on the "Create new AWS Kinesis Stream Setting..." link to bring up the following configuration form:



For the Setting Name you can enter any unique identifier, we will use "TestSetting". For the AWS credentials, you will need to enter your Access Key and Secret Key used to authenticate to your AWS account as well as the region that your Kinesis Stream is in. The Set can be left as Default. Finally, the Stream Name will be the name of the Kinesis Stream that you wish to connect to. This is shown below in the AWS Kinesis Streams console.

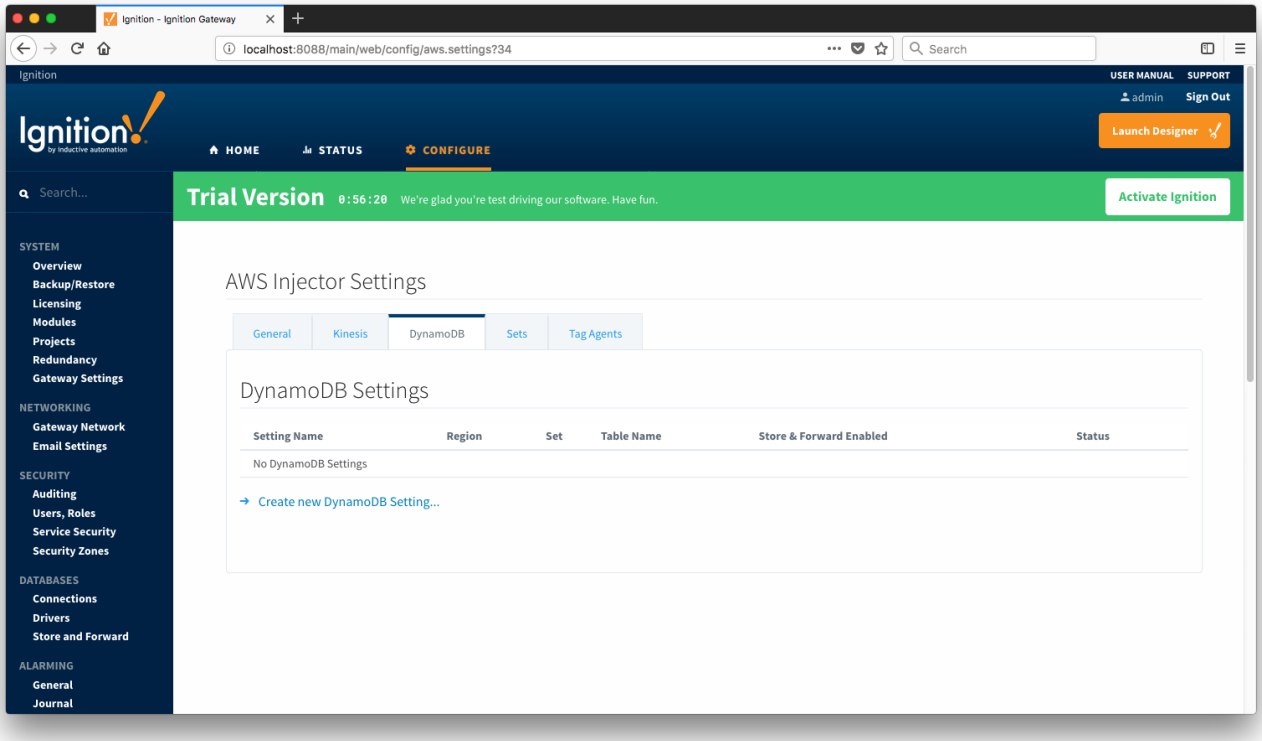


Click on "Create New Kinesis Stream Setting" to finish creating the new configuration setting.

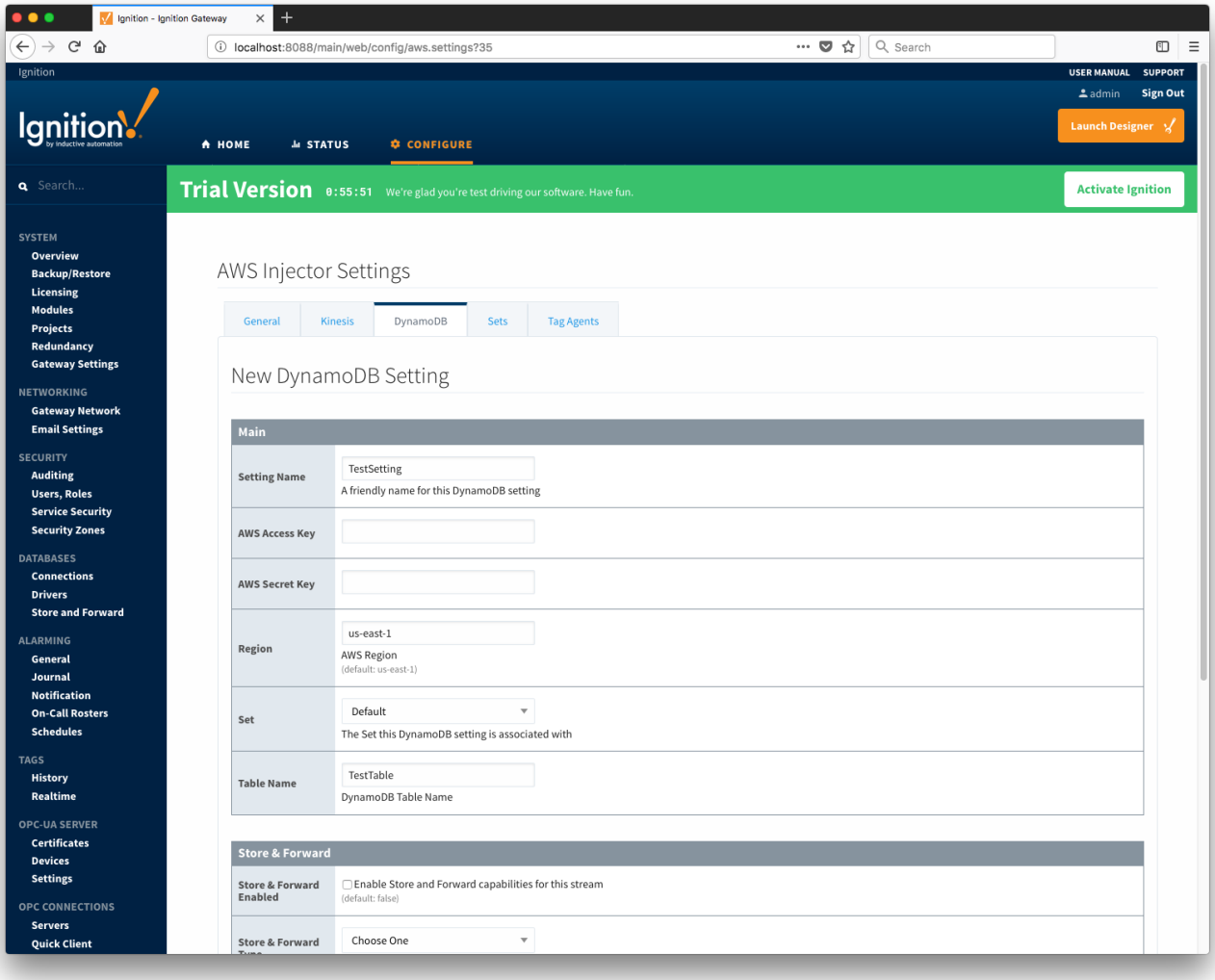


Create an AWS DynamoDB Database Setting

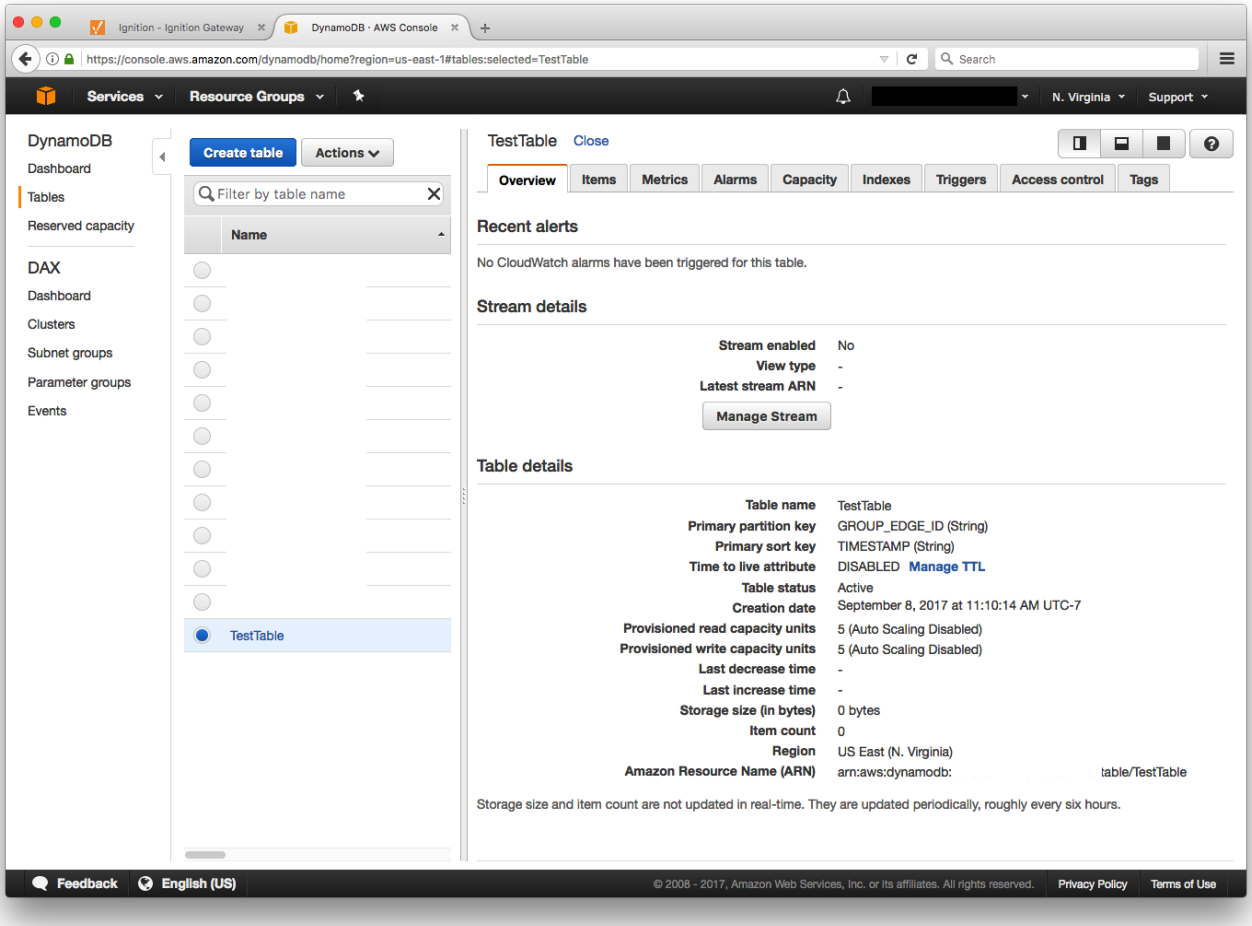
Select the DynamoDB tab to bring up the DynamoDB Database Settings page.



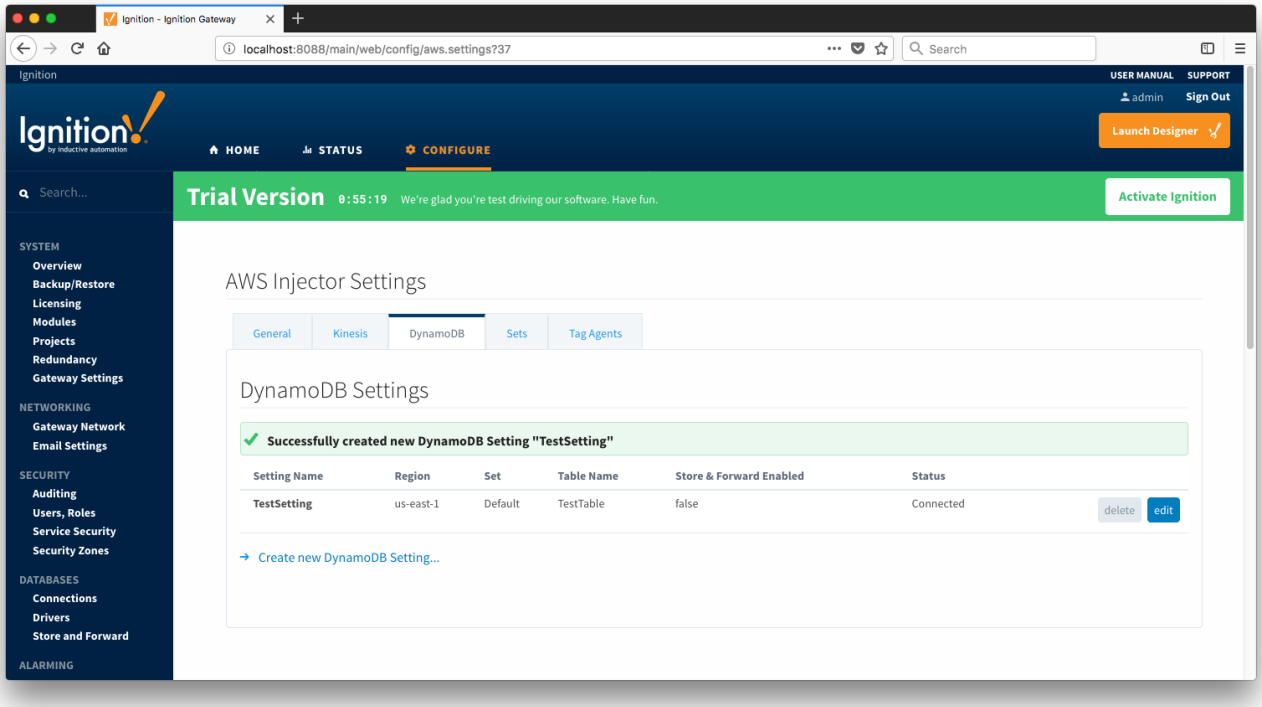
Click on the "Create new DynamoDB Setting..." link to bring up the following configuration form:



For the Setting Name you can enter any unique identifier. We will again use "TestSetting" as it must only be unique among DynamoDB Settings. For the AWS credentials, you will need to enter your Access Key and Secret Key used to authenticate to your AWS account as well as the region that your DynamoDB Database is in. The Set can be left as Default. Finally, the Table Name will be the name of the DynamoDB Table that you wish to connect to. For example, this is shown below in the AWS DynamoDB console.

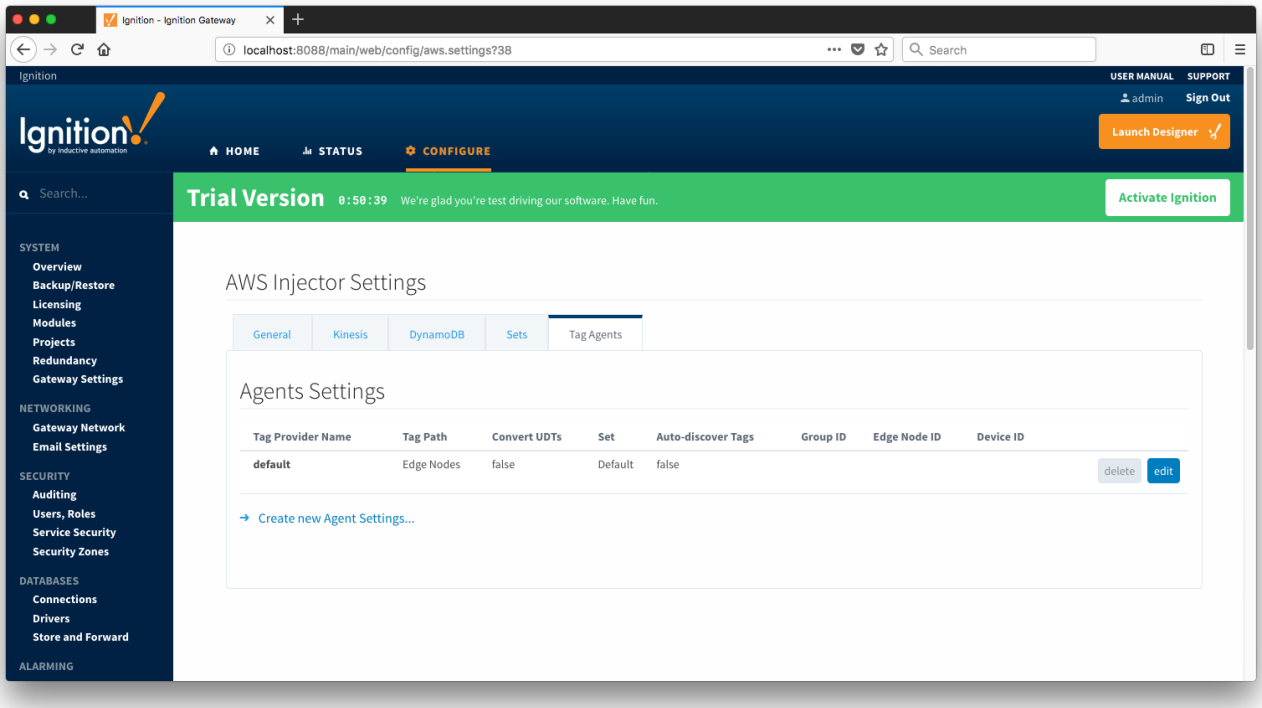


Click on "Create New DynamoDB Setting" to finish creating the new configuration setting.



Use Default Tag Agent Settings

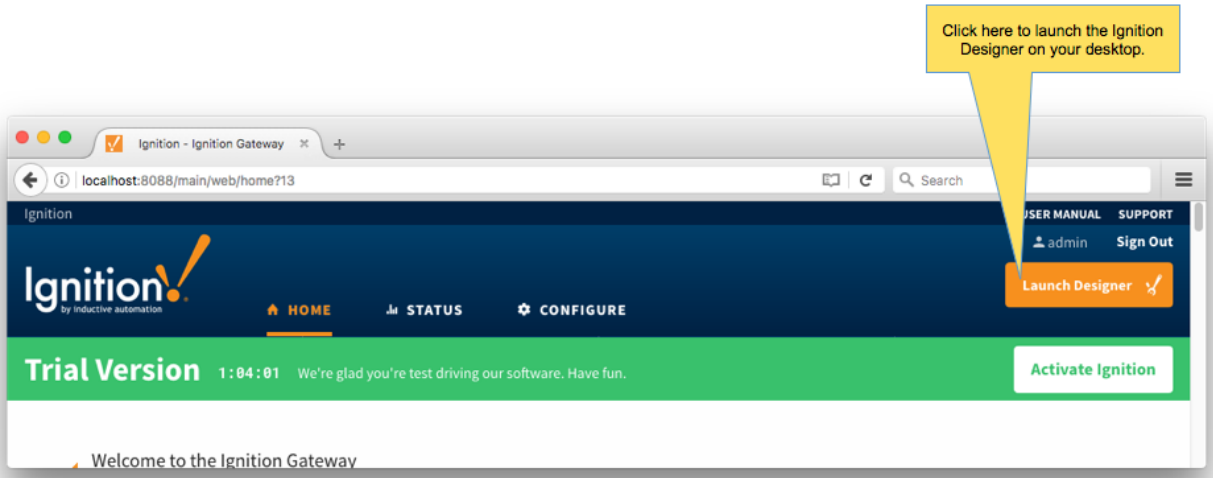
Now the AWS Injector module is connected to a Kinesis Stream and/or a DynamoDB Database and ready to push Tag data you can click on the "Tag Agents" tab you will see that out-of-the-box the AWS Injector module will have one default Tag Agent defined. For this tutorial we will not need to make any configuration changes to the Tag Agents.



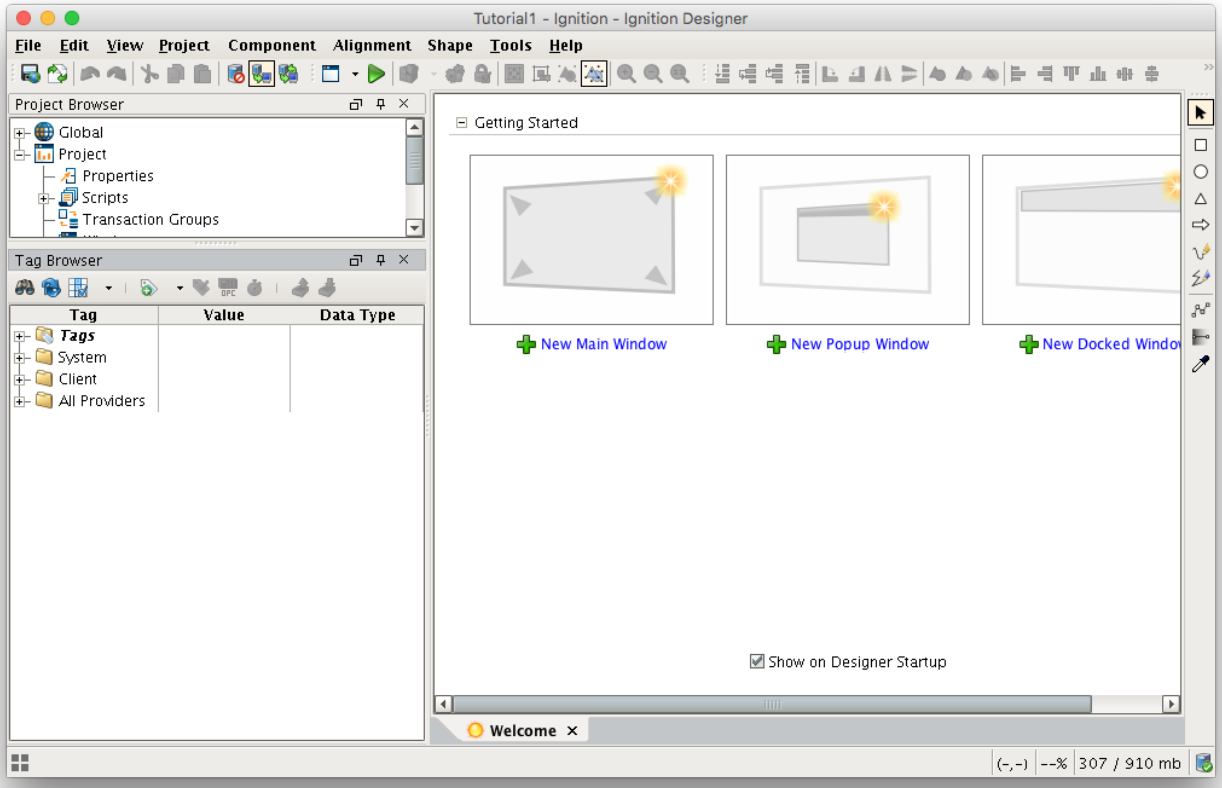
The Default Tag Agent will monitor tags that are in the "Edge Nodes" folder of the "default" Tag Provider. In the next step we go into more detail about the tags in this folder.

Step 4: Use Ignition Designer to Examine the Initial Tag Structure

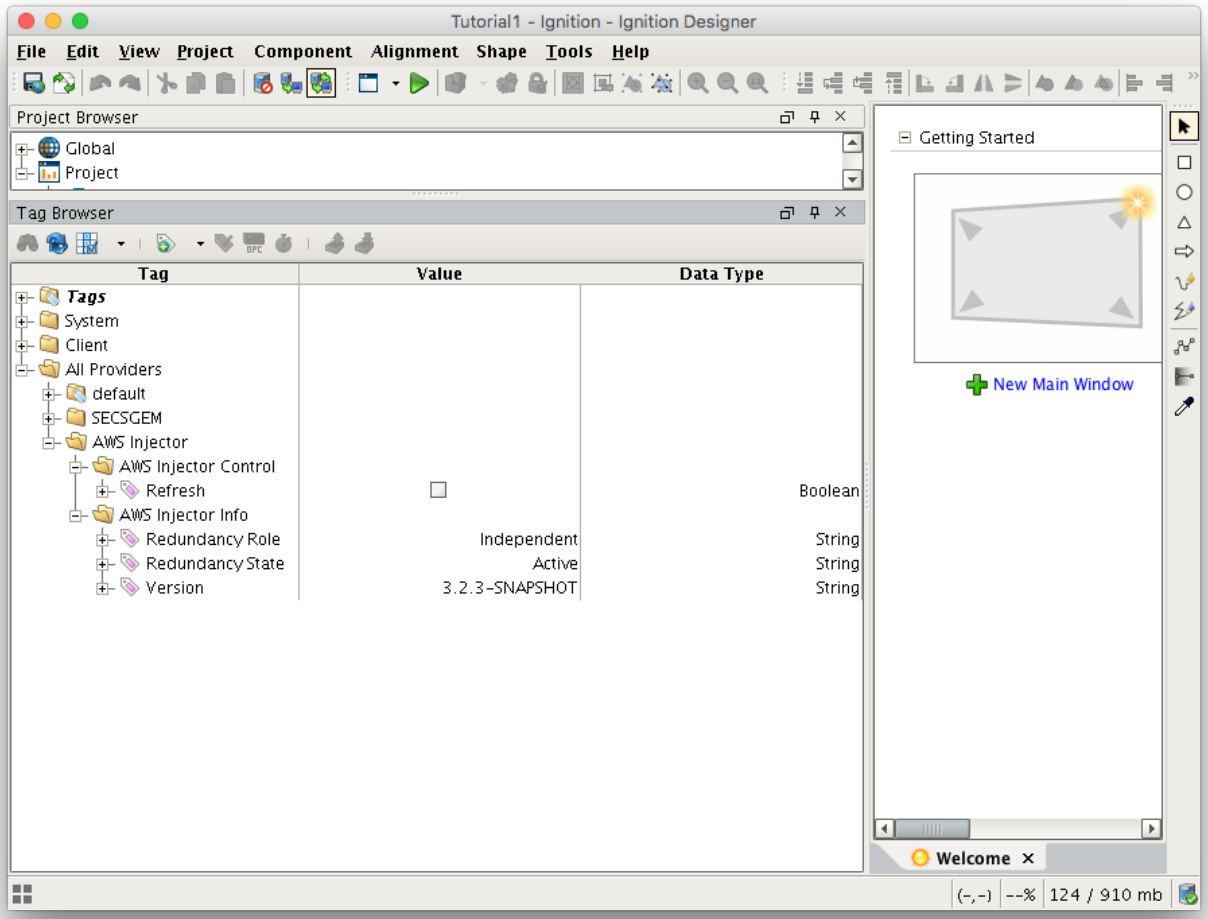
With Ignition running and the AWS Injector module loaded and configured we can now open the Ignition Designer to create/observe the initial Tag structure. Regardless of the OS Ignition is running on, there is a “Launch Designer” button on the Ignition Gateway Console. From here you can launch your Designer on any machine. This is shown below. The default credentials for the designer are the same as the Gateway Console, admin /password. Once you have logged into the Designer enter a new project name and open the project. The project name that we used for this tutorial is simply called “Tutorial1”.



After Designer opens, you will see the default Designer screen as shown below.

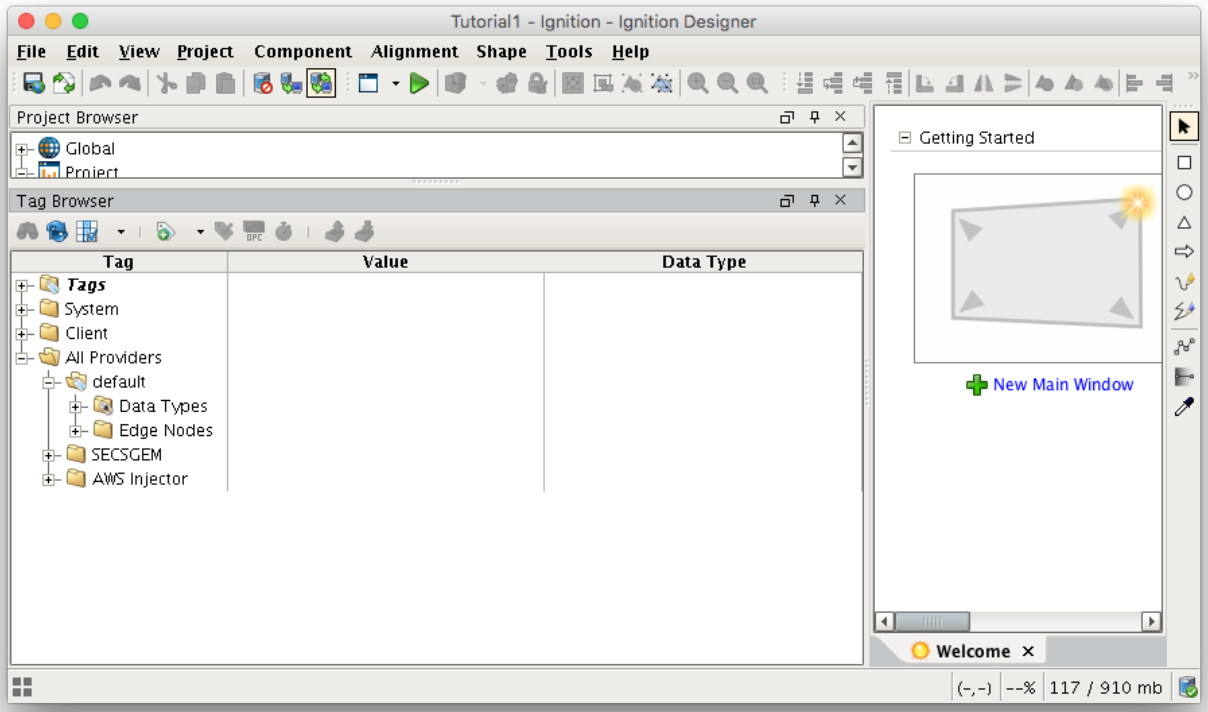


With the AWS Injector module installed in Ignition, a new folder is created under the "All Providers" folder and is called “AWS Injector”. This folder will contain both information tags about the module's version and state, as well as control tags for refreshing the module and it's Tag Agents.



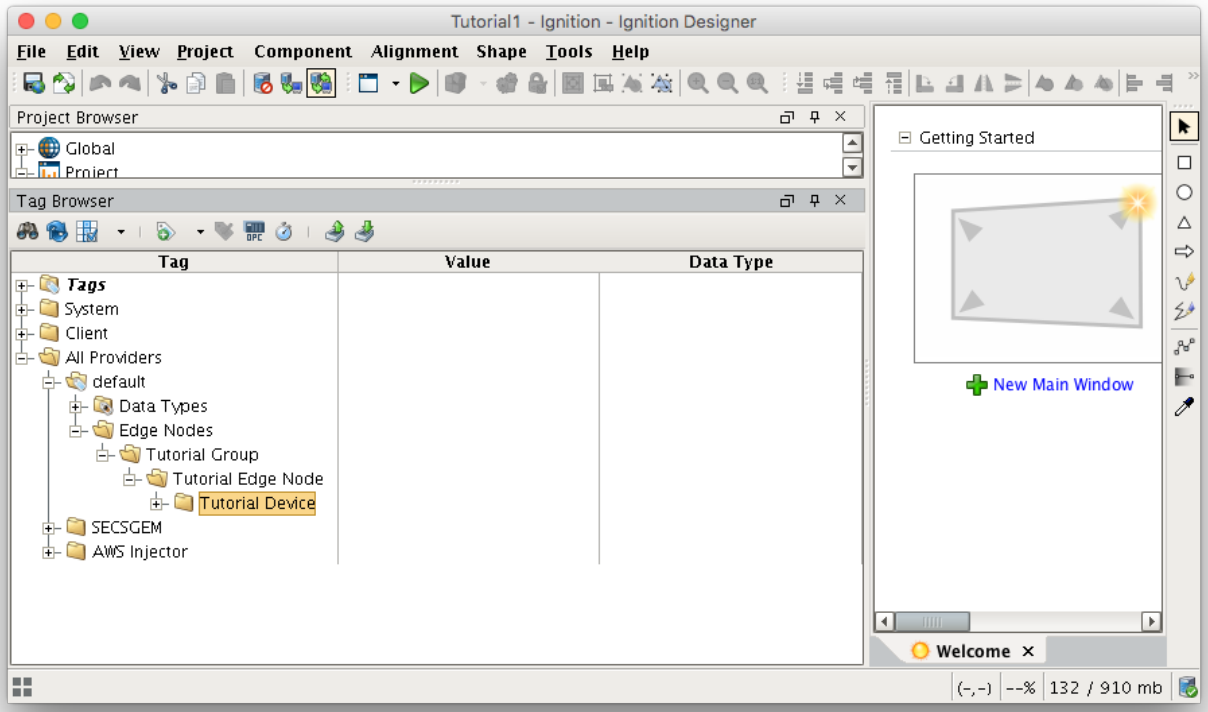
Next, we need to create a folder structure where we will create a virtual Edge device and some tags to be sent by the AWS Injector module. When the AWS Injector module is installed in Ignition, a folder is automatically created in the Ignition tag structure with the following path:

- All Providers/default/Edge Nodes

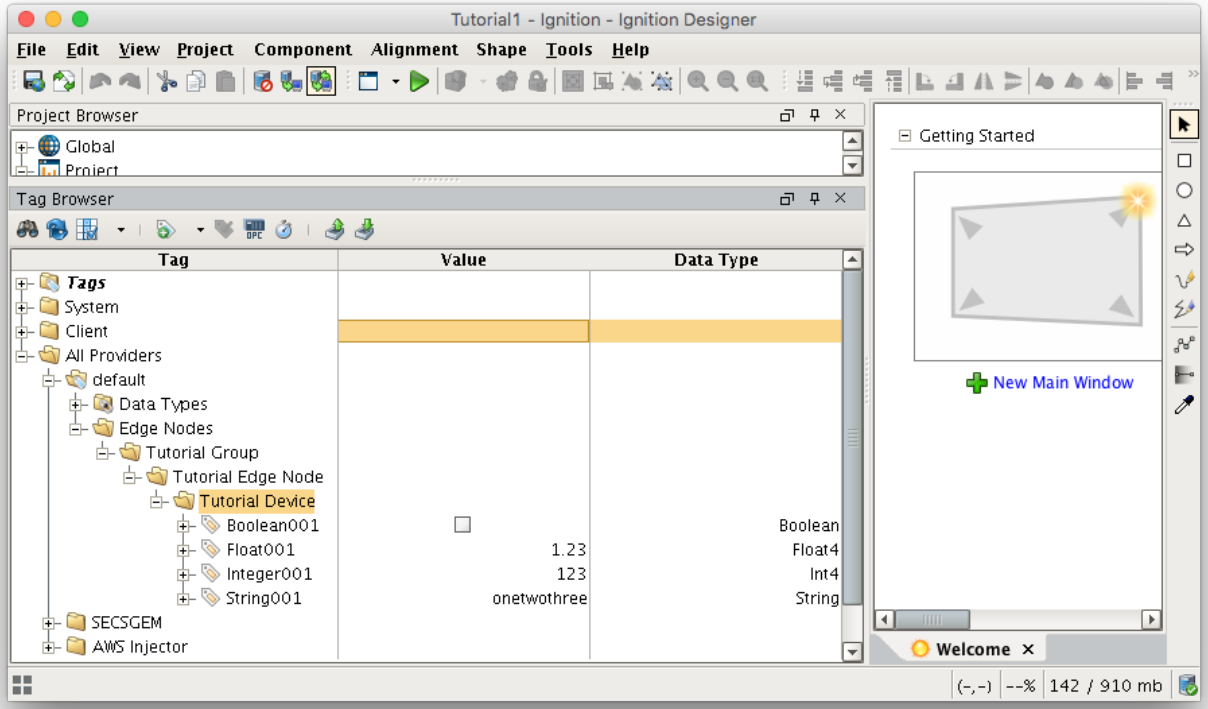


Step 5: Use Ignition Designer to Create New Tags

For this tutorial, right click on the Edge Nodes folder and create a new folder called Tutorial Group. Then right click on the Tutorial Group folder and create another new folder called Tutorial Edge Node. Finally, right click on the Tutorial Edge Node folder and create another new folder called Tutorial Device. This folder structure creates the same hierarchy that is described in the Sparkplug B specification of Group ID, Edge ID, and Device ID.

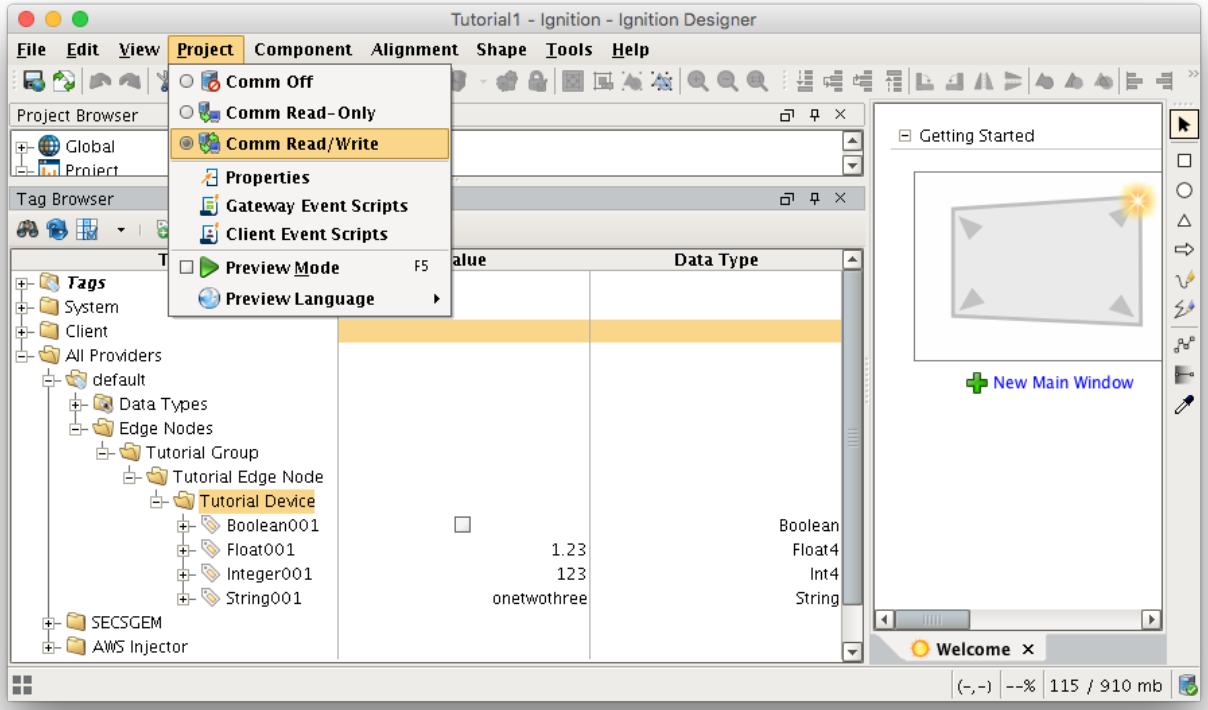


With this folder structure in place, now we can create some memory tags of various data types to publish. Right click on the Tutorial Device folder and select 'New Tag'/'Memory Tag'. In the tag editor change the Name of the tag to "Boolean001", and change the Data Type to Boolean. Follow this same procedure for new memory tags called "Integer001" of type Integer, "Float001" of type Float, and "String001" of type String. The resulting folder structure should look as follows.

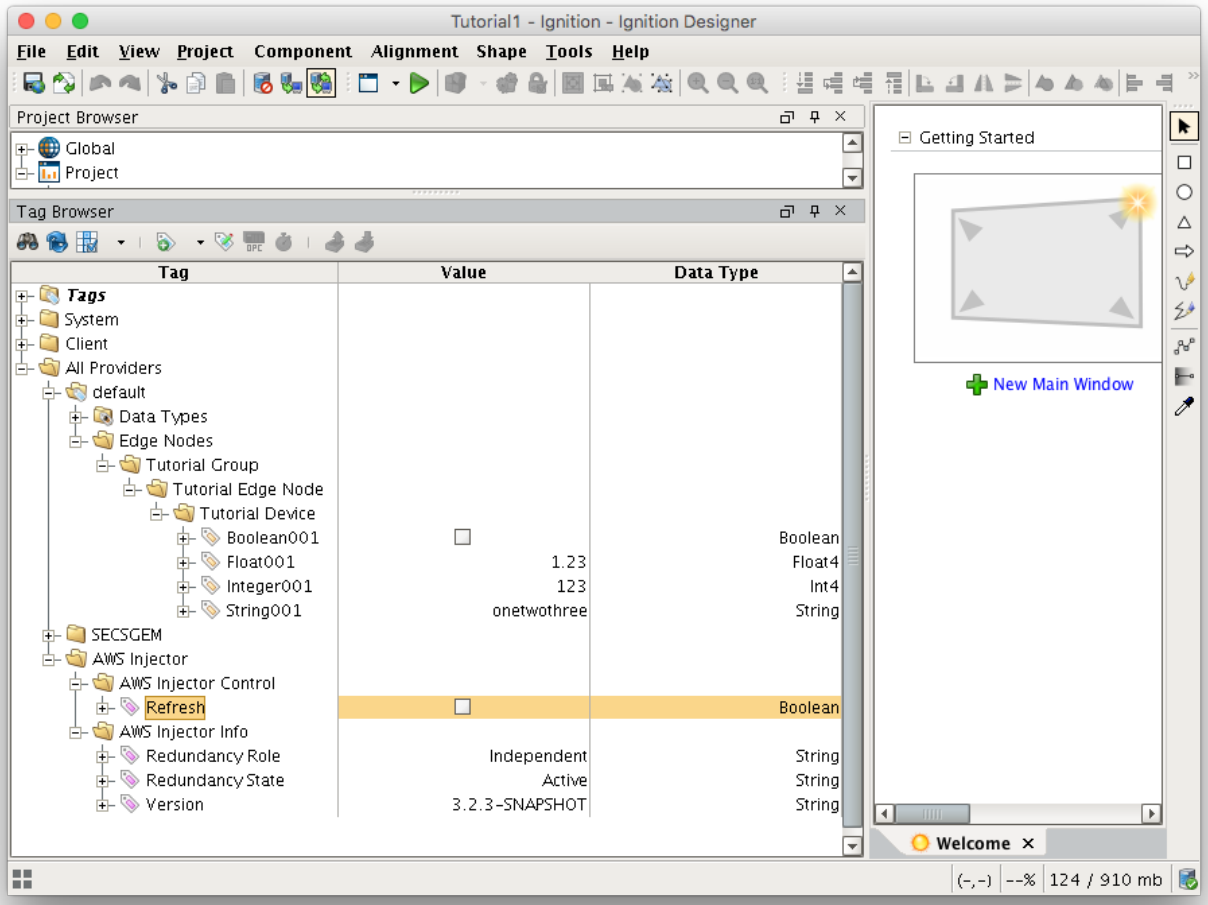


Step 6: Use Ignition Designer to Send Tag Data (Current Tag Values)

Now that we have a folder structure with some tags we can refresh the AWS Injector module. Make sure that the Ignition Designer has read/write communications turned on by selecting Project/Comm Read/Write.



To refresh the default Tag Agent with the folder structure we've created, open the folder "All Providers/AWS Injector/AWS Injector Control" and click on the Refresh Boolean. Note 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.



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.

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
  }
}
```

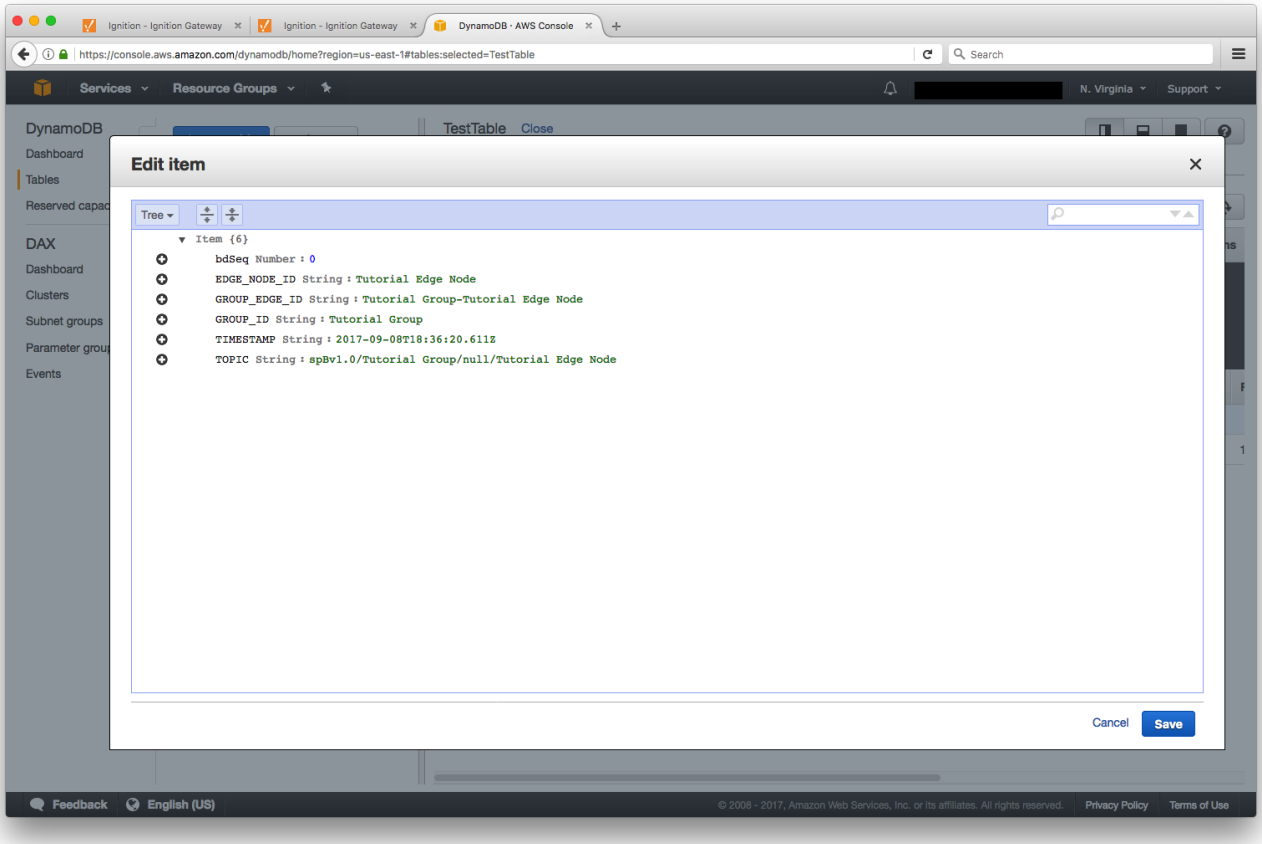

Sending Tag Data to the DynamoDB Table

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:

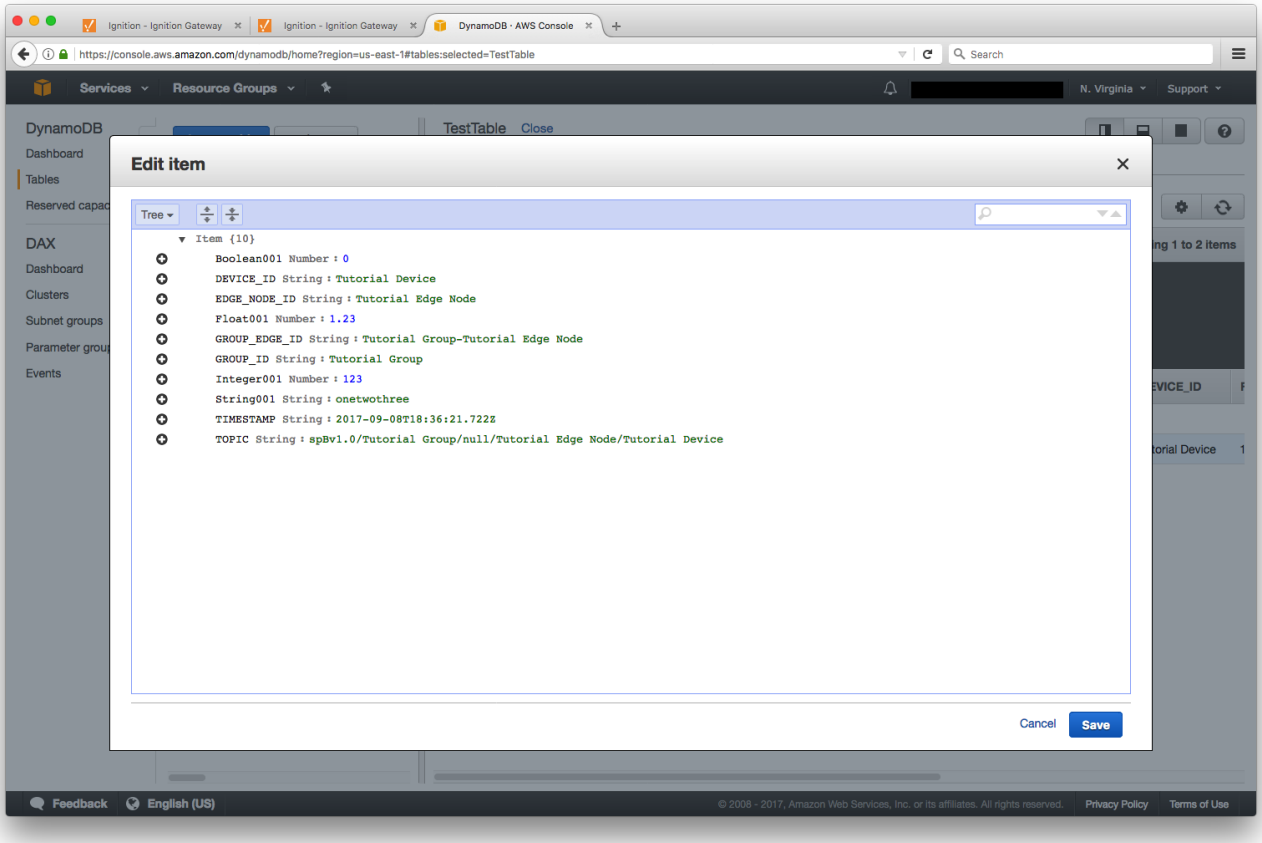


The second item represents the Device and will contain the following:

- 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

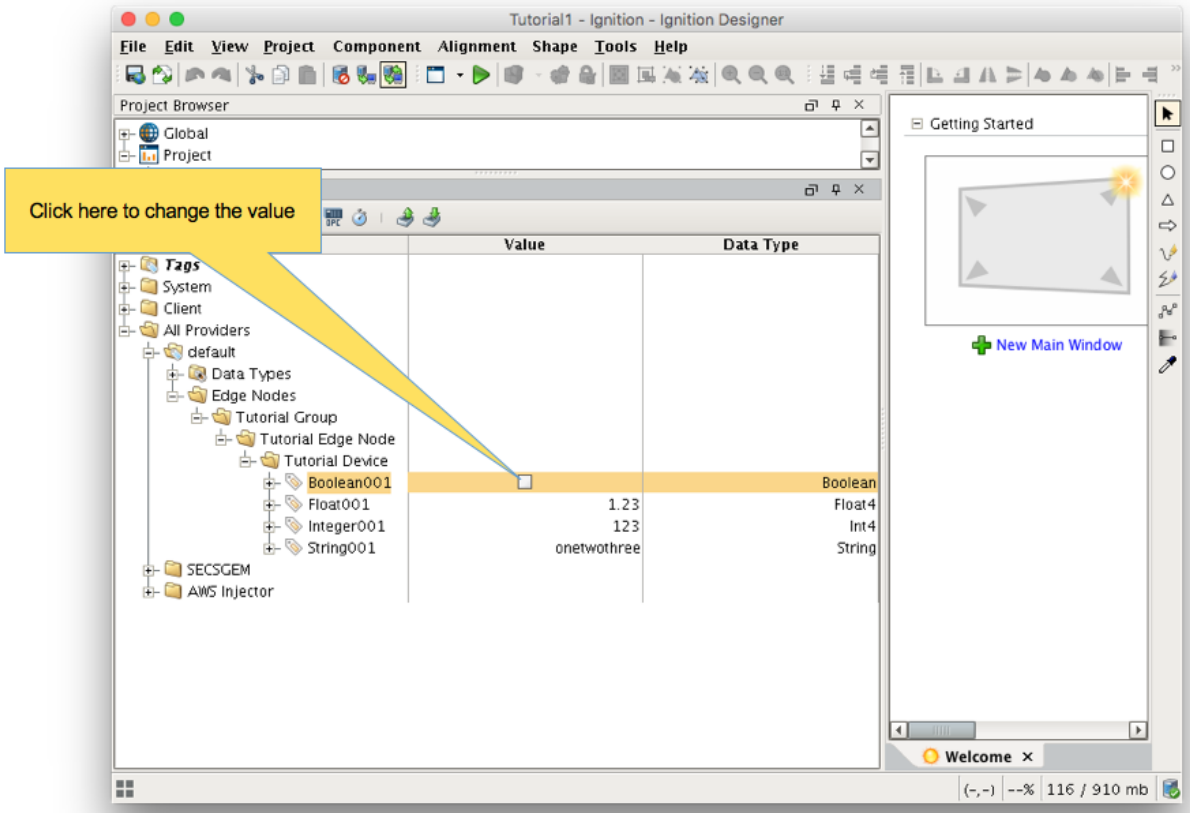
It will look something like this:



Step 7: 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

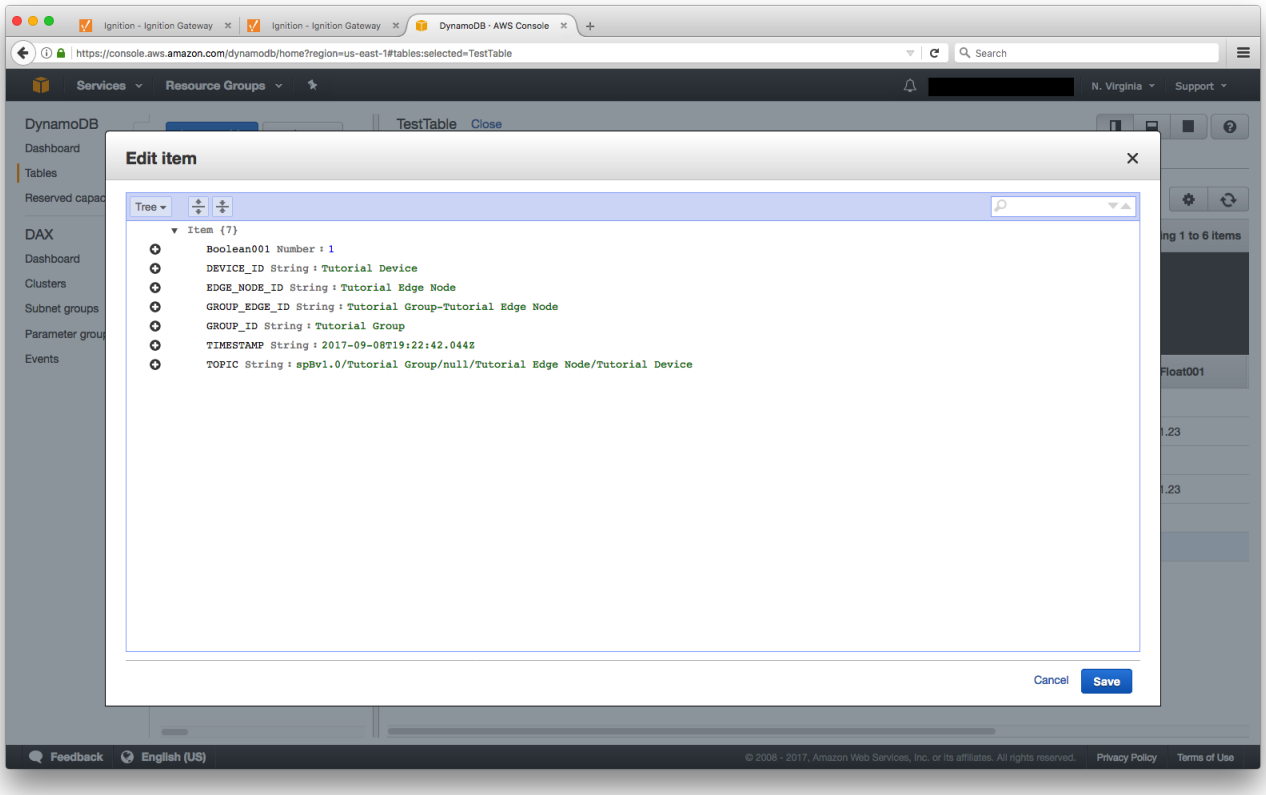
The following message will be constructed to represent this Tag change event and pushed to the Kinesis Stream:

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



Step 8: 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/.