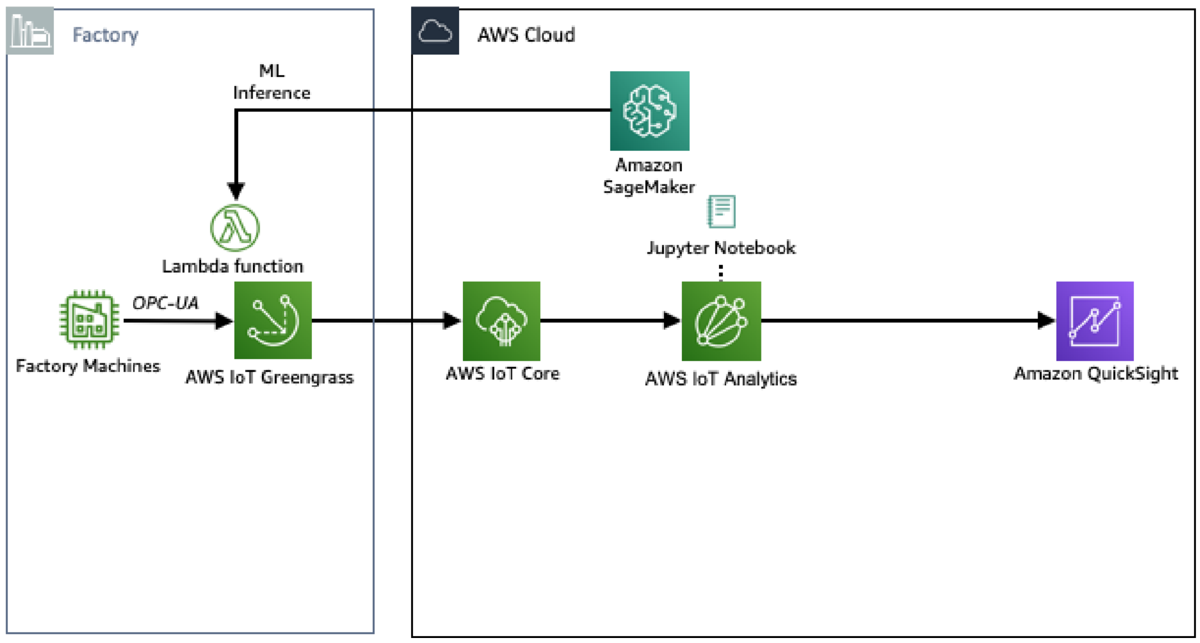
MFG401 - Builders Session - Predictive maintenance with AWS industrial IoT services

Introduction

In this builders session you will go through the fundamental steps to see how to build an predicative maintenance solution in-line with the below architecture.



For this session you will deploy a CloudFormation template that will create a Drill Machine simulator on a EC2 instance that will simulate the Factory Machine within the architecture.

Instructions

BEFORE YOU START

This builders session is set at level 400 and therefore provides the high-level steps to build the architecture with the AWS builder providing you guidance through out the session.

You will need a laptop that is running Linux, Windows or macOS with a AWS account to complete the session and AWS will provide you AWS credits to cover the cost of services used during the session.

STEP 1. DEPLOYING THE CLOUDFORMATION TEMPLATE

- Log into your AWS Console with a IAM user that has an administrator role.
- Launch one of the following CloudFormation templates per your region of choice (suggest Oregon).
    - Oregon

    - Ireland
    - N. Virginia
- Leave all the Parameters as default
- Click the tick box at the bottom of the page to acknowledge the IAM capabilities and create the stack.

> *Note: It will take about 5-10 minutes for the CloudFormation template to successfully deploy*

CONFIGURING GREENGRASS

- Via the AWS Console go into IoT Core > Greengrass and “Create a Group”
- If promted “Grant permission”
- Click “Use easy creation” and name the group for example “Drill”.
- Create the group and core and download the security resources and the current Greengrass Core software.

> *Note: For this session as Greengrass is running on a EC2 instance you will need the x86_64 distribution.*

- Click Finish within the AWS Console window.
- To make the session easier for yourself open a new browser window and go to Cloud9 within the AWS Console
- Click on Open IDE
- Select the folder “my-greengrass” and upload the downloaded zip files from the previous step
- Run the following commands within the terminal to extract and install the Greengrass software and the security certificates.

```
sudo tar -xzvf ~/environment/my-greengrass/greengrass-linux-x86-*.tar.gz -C /
```

```
sudo tar -xzvf ~/environment/my-greengrass/*-setup.tar.gz -C /greengrass
```

- Using yum install Java 8.

```
sudo yum install java-1.8.0-openjdk
```

- Switch /usr/bin/java to link to the java8 version.

```
sudo ln -sf /usr/bin/java8 /usr/bin/java
```

- Start Greengrass with the following command

```
sudo /greengrass/ggc/core/greengrassd start
```

- Switch back to the IoT Core window
- For the Greengrass group add a Role via the Settings menu.

> *The role will start with a name like IIoTWS-GGGatewayRole-**

- Add a local log type for Lamba and Greengrass System with a level of Debug.
- Deploy the Group via the Action menu, use “Automatic selection“

> *Note: If the deploy fails run the following commands in the terminal window.*

```
gg_service_arn=$(aws iam get-role --role-name Greengrass_ServiceRole --query Role.Arn)
aws greengrass associate-service-role-to-account –role-arn ${gg_service_arn//\"/}
```

VIEWING THE DRILL DATA

- Go back to the Cloud9 terminal window and run the following command.

```
opcua-commander -e opc.tcp://192.168.128.20:4840
```

- Using the arrow keys browse down the following tree

> `Objects-> Server-> PLC1-> MAIN-> Drill`.

- Select “Status” and press “M”, this will allow you to view the values being generated from the Drill.
- Press “Q” to quit out of the OPC-UA browser.
- Go back to the IoT window and within the Drill Greengrass group select “Lambdas” and “Add Lambda”
- Use and existing Lambda which starts with “IIoTWS-OPCUA”
- Select the “latest-version” and click “Finish”
- Edit the Lambda configuration and ensure the “Memory limit” is 160MB and the “Lambda lifecycle” is “long-lived”.
- From the Subscriptions menu add a new subscription with the source being the IIoTWS-OPCUA lambda and the target being the IoT Cloud.

> *Note: Set the topic filter to #*

- Now redeploy the Group via the “Actions” menu
- From the AWS IoT menu, select “Test”
- Subscribe to “#” in the topic.

You should see message with values similar to that from the OPC-UA Browser.

EXPOSING ALL DATA FIELDS

- Go back to the browser window with the Cloud9 terminal.
- Open the `index.js` file and add 3 more subscriptions for the **Spindle Speed, the Motor Speed and the Pressure**

> *Note: index.js is within the Lambda_Functions > IIoTWS-OPCUA folder.*

```
{
    name: 'Drill.DrillState',
    nodeId: 'ns=5;s=MAIN.Drill.DrillState',
},
{
    name: 'Drill.Pressure',
    nodeId: 'ns=5;s=MAIN.Drill.Pressure',
},
{
    name: 'Drill.SpindleSpeed',
    nodeId: 'ns=5;s=MAIN.Drill.SpindleSpeed',
},
{
    name: 'Drill.MotorSpeed',
    nodeId: 'ns=5;s=MAIN.Drill.MotorSpeed',
}
```

- Make sure you save the file changes
- Run the following commands below to update the functions version.

```
cd ~/environment/Lambda_Functions
./upload-publish-and-version-lambda-function.sh *OPCUA*
```

- In the IoT console window in the Greengrass group, deploy the group via the action menu.
- Via the "Test" menu and Subscribing to # you will be able to see all the data values.

PRE-PROCESSING THE DATA AT THE EDGE

The next section can be done within AWS IoT Events but for simplicity of the session we will do event processing via a Lambda function at the Edge.

- From IoT Core, add a new Lambda to the Greengrass Group
- Select the existing Lambda which name starts with IIoTWS-CombineEvents
- Select the alias "latest-version" and set the memory to 160MB and the Lambda to run as long lived.
- Add two new subscriptions to the Greengrass group as per the below.

Source	Target	Topic
IIoTWS-OPCUA	IIOTWS-CombineEvents	`opcua/server/node/#`
IIOTWS-CombineEvents	IoT Cloud	#

- Deploy the Greengrass group

> *Note: If you now subscribe to the topic* `data/aggregated you will see a combined data set.`

ANALYISING THE DATA

- In the AWS console, go to IoT Analytics.
- Use the Quick Create option, using the Resource prefix as "Drill" and the Topic as "`data/aggregated"`
- From the menu go to Data sets, select the "drill_dataset"
- Add a schedule of 15 mins and retain data set content as Indefinitely
- From the Action menu, run now

After a couple of seconds you should see a preview of the data in the Result preview menu.

INGEST CLASSIFIED DATA DIRECTLY INTO AWS IOT ANALYTICS

For supervised learning you need to have classified data in order to build a ML model. We have a classified data set that you can now ingest along with the data coming from the PLC.

- In the Cloud9 IDE, open the script `batch-upload-to-iot-analytics-channel.py` in the folder **"iot-analytics"**

- Check that the configured **channel name** matches the name of the AWS IoT Analytics channel you created earlier
- Open a **new terminal** in the Cloud9 IDE, and execute the following commands:
  `cd iot-analytics`
  `python batch-upload-to-iot-analytics-channel.py`
- The data in the csv file will be uploaded in batches directly into the channel

ADD A NEW DATA SET TO QUERY THE CLASSIFIED DATA

- Open the IoT Analytics Console
- Create a new SQL Data Set (**“Data sets”** → **“Create”**) and call it **“classified_data”**
- As SQL query please insert:

```
SELECT * FROM drilldata WHERE classified = 1
```

- Use default values for the Data selection window and the Frequency
- Use an infinite retention period
- Now manually trigger a run (**“Actions”** → **“Run now”**)

> *Note: You now see also the classification columns which indicates if an error occurred (**hasanomaly**) and which one (e.g. **spindlehigh**). The flag hasanomaly is the observation that we will eventually try to predict using a machine learning approach.*

Build a Machine Learning Model

OPEN THE JUPYTER NOTEBOOK

The CloudFormation template already created a Jupyter Notebook for post processing the data from IoT Analytics and create a ML model

- In AWS IoT Analytics go to “Analyze” → “Notebooks”
- Open arrows on the “DrillingAnalytics” notebook and click **“Open this in Jupyter”**

Now use the documentation in the Notebook to go through and execute every step, clicking on the “**Run**” button inside every cell. You can also use the keys combination `SHIFT+ENTER` on the keyboard.

> ***Please ensure you have modified the parameter for the AWS IoT Analytics dataset name if you chose an other name for the classified data set than `classified_data`.***

In the end you will have a ML model stored in the Amazon S3 bucket to be used on AWS Greengrass ML Inference

Inference at the Edge

INCLUDE THE INFERENCE LAMBDA FUNCTION IN THE AWS GREENGRASS GROUP

We will now add the ML Model as well as a Lambda function which will do the prediction based on that model. To do this, please return to the AWS Greengrass tab open in the browser, open the Lambdas menu and click “Add Lambda”:

- Add the Lambda function with the name “**PredictAnomalies**” to the Greengrass group.
- Use the “latest version” alias.
- Ensure to configure enough memory (>200MB), otherwise the function will not start

- This lambda will run on demand and therefore the timeout needs to be set to a proper value. In this case, 10 seconds should be sufficient. We will configure the subscriptions in such a way that any output coming from the CombineEvents Lambda is passed to the prediction Lambda. The model we generated above will then make a prediction and indicate whether the drilling process was normal or not.

To actually use the model, we need to define it as a resource the lambda may access. This is done as follows:

- Go to **“Resources”**
- Click **“Add Resource”**
- Click **“Add machine learning resource”** to connect the AWS Greengrass group with our learned model.

Since our model was stored in S3, we have to locate it and connect the file containing our model with the Greengrass resource:

- Provide a name for the resource
- Select “**Locate or upload a model in S3**”
- Below “Model from S3” select the bucket that starts with “iiotws-iiotwsgreengrass”
- Select the model “folder”
- This folder should have a file called “DrillingPrediction.model.tar.gz” (this file was created by the last step of the Jupyter Notebook)
- Select this file
- As “Local path” put in **“/trained_model”**
- Since our lambda only reads this model file, it is sufficient to have “read-only” access to this folder.

FUNNEL THE AGGREGATED EVENTS TO THE PREDICTION FUNCTION

The prediction lambda needs to be triggered whenever a new output by the “CombineEvents” lambda is generated. Hence, we need to configure the following subscriptions:

- The PredictAnomalies Lambda has to subscribe to the the CombineEvents lambda. CombineEvents publishes the relevant messages to `data/aggregated`
- IoT Core needs to be subscribed to the output of the PredictAnomalies
- Finally, deploy the new AWS Greengrass group.

On the AWS IoT Core Test Console you should see data from the prediction lambda. If you open the code of the prediction lambda in Cloud9, you will see that it will make the prediction by loading the model and the publishing the result along with the sensor data to MQTT topic `data/prediction`.

- Subscribe to the topic “**data/prediction**” in order to be able to filter just for those events.

You should see the prediction result within the Prediction attribute.

In a real scenario with the prediction result you could just signal this to the worker or even control the drill → this is up to you. In the example above, the ML model has detected a potential anomaly. If the drilling process has been completed normally, the model would indicate this by returning “OK” instead of “POSSIBLE_FAILURE”.

Conclusion

Congratulations you are now able to ingest data from an industrial device, store it into a time series data store and build and deploy a machine learning model onto an industrial gateway to detect possible failures.