Chapter 1: Introduction to Industrial IoT

**Cloud Services for IoT**

*Do you use, or plan to use, any of the following cloud service offerings for implementing your IoT solution?*

- Amazon AWS: 51.8%
- Microsoft Azure: 31.2%
- Private / on-prem cloud: 19.4%
- Google Cloud Platform: 18.8%
- Kubernetes: 17.3%
- IBM Bluemix: 14.2%
- OpenStack (on-prem): 10.3%
- Don't know: 10.0%
- Red Hat OpenShift: 9.1%
- None: 9.1%
- Cloud Foundry (on-prem): 8.8%
- Other: 6.1%
- GE Predix: 2.1%

Copyright © 2018, Eclipse Foundation, Inc. Made available under a Creative Commons Attribution 4.0 International License.
IoT DATABASES

Which of the following database technologies do you use in your IoT solution?
The Fourth Industrial Revolution

- First Industrial Revolution:
  - Introduction of mechanical production tools

- Second Industrial Revolution:
  - Labor organization, mass production, use of electricity

- Third Industrial Revolution:
  - First step regarding automation, with electronics and computer science entering companies

- Fourth Industrial Revolution:
  - Interconnected products and services thanks to the new digital technologies

1780: First loom powered by steam
1870: First assembly line
1970: First PLCs
2011: First appearance of "Industry 4.0"
Chapter 2: Understanding the Industrial Process and Devices

INDUSTRIAL PROCESS

Row Materials → Energy → Machines → Human Work → Product

Reject
Chapter 3: Industrial Data Flow and Devices
Industrial Ethernet: 52% (46)
Annual growth: 22% (22)

Fieldbus: 42% (48)
Annual growth: 6% (4)

Wireless: 6% (6)
Annual growth: 32% (32)
Chapter 4: Implementing the Industrial IoT Data Flow

- OPC API
- Application Layer: Distributed COM
- Session Layer: RPC
- Transport Layer: TCP or UDP
- Data Link Layer: Ethernet Framing
- Physical Layer: Ethernet
OpcUaClient

OpcUaServer

DiscoveryEndpoint

SessionEndpoint

AuthenticationService

ValidationAuthority

CreateSession Request

Validate Server Software Certificate

ActivateSession Request

Validate Client Software Certificate

Validate User Identity

Validation Result

Validate Result

ActivateSession Response
Chapter 5: Applying Cybersecurity
Chapter 6: Performing an Exercise Based on Industrial Protocols and Standards
<table>
<thead>
<tr>
<th>Signal Name / Nodeld</th>
<th>Visualize</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counter1</td>
<td></td>
<td>34</td>
</tr>
<tr>
<td>Random1</td>
<td></td>
<td>0.1965140296037563</td>
</tr>
<tr>
<td>IIOT_SIM_RND_01</td>
<td></td>
<td>0.3066766856710803</td>
</tr>
<tr>
<td>Sinusoid1</td>
<td></td>
<td>0.8134736271864131</td>
</tr>
<tr>
<td>Square1</td>
<td></td>
<td>2.0</td>
</tr>
</tbody>
</table>

**Create New Signal**

- **Signal Type**: Sinusoid
- **NodeId/Name**: MyDevice.Pump_01.Pressure
## Prosys OPC UA Simulation Server

**Signal Name / NodeId** | **Visualize** | **Value**
--- | --- | ---
Sawtooth1 | | 0.8000001907348633
Triangle1 | | 0.8000005182741307
IoT_Sim SIN_01 | | 4.755281431440678
Expression1 | | 3.975571353942806
MyDevice.Pump_01.P | | 1.1755711632080925

**Create New Signal**
- **Signal Type**: Sinusoid
- **NodeId/Name**

---

**Graph**
- **MyDevice.Pump_01.Pressure**
- **Show Legend**: Checked
- **Animated**: Checked
- **Symbols**: Checked
- **Show data for last (seconds)**: 60

**v3.0.0-157**
Connect to an endpoint like opc.tcp://host:port/UA/EndpointName.

Actions are:
- Read
- Write
- Browse
- Subscribe
- Unsubscribe
- Event
- Info

Inject your OPC UA address (NodeID) by the Topic of an Inject node or with the OpcUa-Item controlled by an Inject node.

To Read/Write inject the Topic for every operation.

The value to Write should be injected by an OpcUa-Item.

Inject the Topic only once on Subscribe or Event for subscription and you got the changing value on every Interval. Every inject subscribes Dragging a node onto a wire will splice it into the link.
Hold down \[\text{Ctrl}\] when you [click] on a node port to enable quick-wiring.
Edit OpcUa-Item node

- Item: ns=5;s=MyDevice.Pump_01.Pressure
- Type: Double

[Done button circled]
Chapter 7: Developing Industrial IoT and Architecture
<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
<th>TYPE</th>
<th>QUALITY</th>
<th>ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT001.TEMPERATURE</td>
<td>25</td>
<td>Number</td>
<td>GOOD</td>
<td></td>
</tr>
<tr>
<td>CT002.EVT.ANOMALY</td>
<td>Anomaly detected</td>
<td>Text</td>
<td>GOOD</td>
<td>“cause: low temp, ..”</td>
</tr>
</tbody>
</table>

......
Symbolic Name: Temperature
ID: 0435
Alias 1: TP
Type: Number
Attributes:
Attribute 1: ....
Attribute 2: ....
Chapter 8: Implementing a Custom Industrial IoT Platform
device0.my.measure.temp, 1529514834, 11, GOOD

hash(device0..., ...) = 58978

device5.measure.flow, 1529514834, 0.1, GOOD

hash(device5..., ...) = 32990
A screen capture of a graph generated by KairosDB. The graph shows a time series of temperature measurements, with a trend line indicating an increase over time. The graph is labeled 'device01.my.measure.temperature' and has a time range from 2016-08-01 06:10:00.712 pm to another unspecified time. The graph includes options for 'Graph', 'Show Query', 'Save', and 'Delete Data'.
<table>
<thead>
<tr>
<th>Conn Id</th>
<th>http_kairosdb</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conn Type</td>
<td>HTTP</td>
</tr>
<tr>
<td>Host</td>
<td><a href="http://localhost:8080">http://localhost:8080</a></td>
</tr>
<tr>
<td>Schema</td>
<td></td>
</tr>
<tr>
<td>Login</td>
<td></td>
</tr>
<tr>
<td>Password</td>
<td></td>
</tr>
<tr>
<td>Port</td>
<td></td>
</tr>
<tr>
<td>Extra</td>
<td></td>
</tr>
</tbody>
</table>

**Save** | **Save and Add Another** | **Save and Continue Editing** | **Cancel**

### DAGs

**Search:** lot-book

<table>
<thead>
<tr>
<th>DAG</th>
<th>Schedule</th>
<th>Owner</th>
<th>Recent Tasks</th>
<th>Last Run</th>
<th>DAG Runs</th>
<th>Links</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Status:** On
- **Schedule:** **'****'**
- **Owner:** lot-book
- **Last Run:** 2018-06-17 23:58
- **DAG Runs:**
- **Links:**

Showing 1 to 1 of 1 entries
Symbolic Name: Temperature
ID: 0435
Alias 1: TP
Type: Number
Attributes:
Attribute 1: ....
Attribute 2: ....

CREATE (TEMP01:Measure {name:'CT001.TEMPERATURE01', alias:'TEMP01', type:'TEMPERATURE', uom:'DEG'})
CREATE (FLOW01:Measure {name:'CT001.FLOW01', alias:'FLOW01', type:'FLOW', uom:'sm3/sec'})
CREATE (TEMP01)-[:MEASURE_OF]->(CT001),
FLOW01)-[:MEASURE_OF]->(CT001)
MATCH (:Section)<-[BELONGING_OF]-(EQ)<-[MEASURE_OF]-(M)
WHERE M.type='TEMPERATURE'
RETURN EQ.name, M.name, M.uom

MATCH (:Section)<-[BELONGING_OF]-(EQ)<-[MEASURE_OF]-(M)
WHERE M.type='TEMPERATURE'
RETURN EQ.name, M.name, M.uom

<table>
<thead>
<tr>
<th>EQ.name</th>
<th>M.name</th>
<th>M.uom</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;CT001&quot;</td>
<td>&quot;CT001.TEMPERATURE01&quot;</td>
<td>&quot;DEG&quot;</td>
</tr>
</tbody>
</table>

Started streaming 1 records after 1 ms and completed after 2 ms.
This utility will walk you through creating a package.json file. It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and save it as a dependency in the package.json file.

Press ^C at any time to quit.

package name: neo4j
version: 1.0.0
description:
git repository:
keywords:
author:
license: (ISC)

About to write to /Users/giacomoveneri/Documents/workspace-iiot/neo4j/package.json:

```json
{
  "name": "neo4j",
  "version": "1.0.0",
  "main": "ask_for_measure.js",
  "dependencies": {
    "neo4j-driver": "^1.6.1"
  },
  "devDependencies": {},
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "description": ""
}
```

Is this OK? (yes)

```bash
npm WARN neo4j@1.0.0 No description
npm WARN neo4j@1.0.0 No repository field.
```

+ neo4j-driver@1.6.1
updated 1 package and audited 6 packages in 3.33s
found 0 vulnerabilities

Record {
  keys: [ 'EQ.name', 'M.name', 'M.uom' ],
  length: 3,
  _fields: [ 'CT001', 'CT001.TEMPERATURE@1', 'DEG' ],
  _fieldlookup: { 'EQ.name': 0, 'M.name': 1, 'M.uom': 2 } }

```bash
```
Chapter 9: Understanding Industrial OEM Platforms
CATALOG
Browse unique services and analytics by category, function, and utility, and combine them to build custom apps.

User Account and Authentication
Use this service for a full-featured OAuth 2.0 server.

Time Series
Quickly and efficiently manage, distribute, ingest, store, and analyze time series data.
New subscription

* Required fields

Region*: US West

Org*: 

Space*: dev

Service instance name*: myname-uaa

Service plan*: Free

Admin client secret*: mypassword

Subdomain: Subdomain
Space: dev

Service instances

- myname-UAA
  - Free, predix-uaa, 3 apps
- myname-TS
  - Free, predix-timeseries, 3 apps

At-A-Glance

- 2 Clients
  - Manage Clients
- 1 Users
  - Manage Users
- 24 Groups
  - Manage Groups
- 1 Identity Providers
  - Manage IdPs

Your UAA

https://predix-uaa.run.aws-usw02-pr.ice.predix.io
Choose Request: Latest Datapoints Request
POST https://time-series-store-predix.run.aws-usw02-pr.ice.predix.io/v1/datapoints/latest

Request Headers
- predix-zone-id
- authorization: Bearer
- content-type: application/json

Request Body
```
{
  "tags": [
    {
      "name": "IOT-BOOK:CompressionRatio"
    }
  ]
}
```
Sample Application

Putting it together

It’s super easy to create an application by combining components and design modules from the Predix Design System. Coded versions of most of the layout examples provided in the Design Starter Kit are available here:

Get the Layouts

View the Sample App | Get the Code

We’ve also created an holistic example application for you that combines branding, navigation, routing, asset selection, data visualization, and other components to illustrate several example layouts that are possible with the Predix components.

Use with other frameworks

Versions of the sample app have also been created to demonstrate the interoperability of Predix components with the following popular JavaScript frameworks:

Angular | React
Customize Predix Machine

Point your device at a custom Predix Time Series and a custom Application URL. Later, you may use the Reset Device link to point everything back at the shared cloud app.

Learn more about setting up a Predix Time Series instance.

wss://gateway-predix-data-services.run.aws-usw02-pr.ice.predix.io/v1/stream/messages

Enter Timeseries Secure Websocket Endpoint URL for Data Ingestion

Enter Instance ID of your Timeseries Service

https://[redacted].predix-uaa.run.aws-usw02-pr.ice.predix.io/oauth/token

Enter Issuer ID URL of the UAA service, ending with /oauth/token

myclient

Enter Client ID. Client ID is the UAA account with privileges to Time Series

*********

Enter Secret. Secret is the password for the Client Id

https://kit-cloud-app.run.aws-usw02-pr.ice.predix.io

Enter the url of your application and we will replace the url for the View in Cloud button

Cancel

Submit
Make a few basic queries against your time series instance.

UAA URL: https://predix-uaa.run.aws-usw02-pr.ice.predix.io

Choose Request: Get Tags

Request Headers
- predix zone-id
- authorization: Bearer

HTTP Response Code: 200 OK

```json
{
  "result": [
    "light",
    "rotaryangle",
    "sound",
    "temperature",
    "Congestion-2017:CompressionRatio",
    "IOT BOOK:CompressionRatio",
    "leftBTR",
    "rightBTR",
    "MN:temp1",
    "MN:IDP-IPF1:light",
    "MN:IDP-IPF1:rotaryangle",
    "MN:IDP-IPF1:sound",
    "MN:IDP-IPF1:temperature"
  ]
}
```
Chapter 10: Implementing a Cloud Industrial IoT Solution with AWS
AWS Accounts Include 12 Months of Free Tier Access

Including use of Amazon EC2, Amazon S3, and Amazon DynamoDB

Visit [aws.amazon.com/free](http://aws.amazon.com/free) for full offer terms
Success
You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: https://i.xxxxxxxxxxxxxx.signin.aws.amazon.com/console

<table>
<thead>
<tr>
<th>User</th>
<th>Access key ID</th>
<th>Secret access key</th>
</tr>
</thead>
<tbody>
<tr>
<td>myadmin</td>
<td>XXXXXXXXXXXXX</td>
<td>********* Show</td>
</tr>
<tr>
<td>Category</td>
<td>Services</td>
<td></td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Database</td>
<td>RDS, DynamoDB, ElastiCache, Neptune, Amazon Redshift</td>
<td></td>
</tr>
<tr>
<td>Migration</td>
<td>AWS Migration Hub, Application Discovery Service, Database Migration Service, Server Migration Service, Snowball</td>
<td></td>
</tr>
<tr>
<td>Networking &amp; Content Delivery</td>
<td>VPC, CloudFront, Route 53, API Gateway, Direct Connect</td>
<td></td>
</tr>
<tr>
<td>Developer Tools</td>
<td>CodeStar</td>
<td></td>
</tr>
<tr>
<td>Machine Learning</td>
<td>Amazon SageMaker, Amazon Comprehend, AWS DeepLens, Amazon Lex, Machine Learning, Amazon Polly, Rekognition, Amazon Transcribe, Amazon Translate</td>
<td></td>
</tr>
<tr>
<td>Customer Engagement</td>
<td>Amazon Connect, Pinpoint, Simple Email Service</td>
<td></td>
</tr>
<tr>
<td>Business Productivity</td>
<td>Alexa for Business, Amazon Chime, WorkDocs, WorkMail</td>
<td></td>
</tr>
<tr>
<td>Desktop &amp; App Streaming</td>
<td>WorkSpaces, AppStream 2.0</td>
<td></td>
</tr>
<tr>
<td>Internet of Things</td>
<td>IoT Core, IoT 1-Click, IoT Device Management, IoT Analytics, Greengrass, Amazon FreeRTOS</td>
<td></td>
</tr>
</tbody>
</table>
You don't have any policies yet

AWS IoT policies give things permission to access AWS IoT resources (like other things, MQTT topics, or thing shadow).

Learn more  Create a policy
You don't have any things yet
A thing is the representation of a device in the cloud.

Learn more  Register a thing
CREATE A THING
Add your device to the thing registry

This step creates an entry in the thing registry and a thing shadow for your device.

Name
my-iot-book-device

Apply a type to this thing
Using a thing type simplifies device management by providing consistent registry data for things that share a type. Types provide things with a common set of attributes, which describe the identity and capabilities of your device, and a description.

Thing Type
simplestest

Create a type

Add this thing to a group
Adding your thing to a group allows you to manage devices remotely using jobs.

Thing Group
Groups / my-iot-book-group /
Certificate created!

Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page.

In order to connect a device, you need to download the following:

<table>
<thead>
<tr>
<th>Item</th>
<th>File Name</th>
<th>Download</th>
</tr>
</thead>
<tbody>
<tr>
<td>A certificate for this thing</td>
<td>58b5ac2b10.cert.pem</td>
<td>Download</td>
</tr>
<tr>
<td>A public key</td>
<td>58b5ac2b10.public.key</td>
<td>Download</td>
</tr>
<tr>
<td>A private key</td>
<td>58b5ac2b10.private.key</td>
<td>Download</td>
</tr>
</tbody>
</table>

You also need to download a root CA for AWS IoT:
A root CA for AWS IoT [Download]

Deactivate

Cancel

Done

Attach a policy

CREATE A THING
Add a policy for your thing

Select a policy to attach to this certificate:

- [ ] iiot-book-policy-4-mqtt

1 policy selected

Register Thing
**Manage**
- Things
- Types
- Groups

**Secure**
- Act
- Test

**Settings**

**Custom endpoint**
This is your custom endpoint that allows you to connect to AWS IoT. Each of your Things has a REST API available at this endpoint. This is also an important property to insert when using an MQTT client or the AWS IoT Device SDK.
Your endpoint is provisioned and ready to use. You can now start to publish and subscribe to topics.

**Logs**
You can enable AWS IoT to log helpful information to CloudWatch Logs. As messages from your devices pass through the message broker and the rules engine, AWS IoT logs process events which can be helpful in troubleshooting.

**Role**
Level of verbosity
DISABLED

**Event-based messages**
AWS IoT can send event-based messages to pre-determined MQTT topics when specific service events occur.
Create DynamoDB table

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

**Table name**: current_signals

**Primary key**: Partition key
- **ts_date**: String
- Check box: Add sort key
- **ts_time**: String

**Table settings**

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

- **Use default settings**
  - No secondary indexes.
  - Provisioned capacity set to 5 reads and 5 writes.
  - Basic alarms with 80% upper threshold using SNS topic "dynamodb".
  - On-Demand Backup and Restore Enabled

**You do not have the required role to enable Auto Scaling by default.**

Please refer to documentation.

Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console.
You don't have any rules yet

Rules give your things the ability to interact with AWS and other web services. Rules are analyzed and actions are performed based on the messages sent by your things.

Learn more
Create a rule
Message source
Indicate the source of the messages you want to process with this rule.
Using SQL version
2016-03-23

Rule query statement
```
SELECT *, topic() as topic FROM 'signals/#' 
```

Attribute
```
*, topic() as topic 
```

Topic filter
```
signals/# 
```
Author from scratch
Start with a simple "hello world" example.

Name
myFunctionName

Runtime
Node.js 6.10

Role
Create new role from template(s)
Lambda will automatically create a role with permissions from the selected policy templates. Note that basic Lambda permissions (logging to CloudWatch) automatically be added. If your function accesses a VPC, the required permissions will also be added.

Role name
myRoleName
IoT type
Configure a custom IoT rule, or set up an IoT button.

- Custom IoT rule
- IoT Button

Rule
Pick an existing rule, or create a new one.

Create a new rule

Rule name
Enter a name to uniquely identify your IoT rule.

my_iot_lambda_threshold_rule

Rule description
Provide an optional description for your rule.

Rule query statement
Create a SQL statement for this rule. For example, to set up your first dash button: SELECT * FROM 'iotbutton/+'.

SELECT * FROM 'signals/#'

Lambda will add the necessary permissions for AWS IoT to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.

Enable trigger
Enable the trigger now, or create it in a disabled state for testing (recommended).
The Greengrass Group is a cloud-configured managed collection of local devices and Lambda functions that can be programmed to communicate with each other through a Core device. Groups can contain up to 200 local devices.

Group Name

my-iot-book-group-greengrass
Connect your Core device

The final steps are to load the Greengrass software and then connect your Core device to the cloud. You can defer connecting your device at this time, but you must download your public and private keys now as these cannot be retrieved later.

Download and store your Core’s security resources

<table>
<thead>
<tr>
<th>Resource</th>
<th>Filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>A certificate for this Core</td>
<td>91bc2408e.cert.pem</td>
</tr>
<tr>
<td>A public key</td>
<td>91bc2408e.public.key</td>
</tr>
<tr>
<td>A private key</td>
<td>91bc2408e.private.key</td>
</tr>
<tr>
<td>Core-specific config file</td>
<td>config.json</td>
</tr>
</tbody>
</table>

Download the current Greengrass Core software

To install Greengrass on your Core download the package and follow Getting Started Guide.

<table>
<thead>
<tr>
<th>Software configurations</th>
<th>Show architectures</th>
<th>Show all distributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture - Distribution - OS</td>
<td>Download</td>
<td></td>
</tr>
<tr>
<td>x86_64</td>
<td>Amazon Linux</td>
<td>Linux</td>
</tr>
<tr>
<td>ARMv8 (AArch64)</td>
<td>Ubuntu 14.04 - 16.04</td>
<td>Linux</td>
</tr>
<tr>
<td>ARMv7L</td>
<td>Ubuntu 14.04 - 16.04</td>
<td>Linux</td>
</tr>
<tr>
<td>x86_64</td>
<td>Ubuntu 14.04 - 16.04</td>
<td>Linux</td>
</tr>
</tbody>
</table>

By downloading this software you agree to the License Agreement.
SET UP YOUR GREENGRASS GROUP

Run a scripted easy Group creation

In order to speed up and simplify Group creation AWS Greengrass will handle the following processes and use default settings. By proceeding to the next step, you are giving permission for us to complete the following steps.

AWS Greengrass will take these actions on your behalf using default settings:

- Create a new Greengrass Group in the cloud
- Provision a new Core in the IoT Registry and add to the Group
- Generate public and private key set for your Core
- Generate a new security certificate for the Core using the keys
- Attach a default security policy to the certificate

[AWS Greengrass Core SDK]

About this Software Developer Kit

The AWS Greengrass Core SDK enables Lambda functions to interact with the Greengrass Core on which they run. This allows them to publish messages and interact with shadow data or invoke Lambda functions within the Greengrass Core.

- Version 1.2.0
- `aws-greengrass-core-sdk-js-1.2.0.tar.gz`

Download Greengrass Core SDK

Cancel
Add a Lambda to your Greengrass Group

Local Lambdas are hosted on your Greengrass Core and connected to each other and devices by Subscriptions, but they can also be deployed individually to your Group.

Create a new Lambda function
You will be taken to the AWS Lambda Console and can author a new Lambda function.

Create new Lambda

Use an existing Lambda function
You will choose from a list of existing Lambda functions.

Use existing Lambda

Create function

Author from scratch
Start with a simple 'Hello world' example.

my-test-book-opcua-lambda

Runtime
Node.js 6.10

Role
Choose an existing role
Create a new role
Create a custom role

Cancel
Create function
AWS Lambda requires access to your resources

AWS Lambda uses an IAM role that grants your custom code permissions to access AWS resources it needs.

- Hide Details

Role Summary

Role Description: Lambda execution role permissions

IAM Role: Create a new IAM Role

Role Name: lambda_basic_execution

View Policy Document

my-iot-book-opcua-lambda

Configuration | Monitoring

Designer

Add triggers: Choose a trigger from the list below to add it to your function.

API Gateway
AWS IoT
CloudWatch Events
CloudWatch Logs
CodeCommit
Cognito Sync
Trigger

Amazon CloudWatch Logs
Resources that the function's role has access to appear here

Function code

The deployment package of your Lambda function "my-iot-book-opcua-lambda" is too large to enable inline code editing. However, you can still invoke your function.

Code entry type:
- Upload a zip file
- Create Jenkins Pipeline
- Uploaded

Runtime: Node.js 6.10
Handler: index.handler
Save
Group-specific Lambda configuration

my-iiot-book-opcua-lambda

View function in AWS Lambda

Version 1  Remove version

Memory limit

| 16 | MB |

Timeout

| 3 | Second |

Lambda lifecycle

- On-demand function
- Make this function long-lived and keep it running indefinitely

Read access to /sys directory

- Disable
- Enable

Input payload data type

- JSON
- Binary
Collect device messages
You haven't created a channel just yet, but you can get started now.

Create a channel

Create a pipeline from this channel
Create Pipeline
Set attributes of your messages

List the attributes expected in incoming messages to make the AWS IoT Analytics experience smarter and faster. Upload a JSON document of attributes or enter attributes manually.

Attributes

- Attribute name
- temperature

Back
Next
CREATE DATA SET
Set ID and source

Provide an ID and select a data store as a source for this data set. This cannot be changed later.

ID
signals_ds

Select data store source
signals_datastore

Tags

Add Tag

Cancel

Next

DATA SET
signals_dataset
SUCCEEDED

Details

Data set ARN

A data set Amazon Resource Name (ARN) uniquely identifies this data set.

arn:aws:iotanalytics:eu-west-1:602032247067:dataset/signals_dataset

Details

SQL query

SELECT * FROM signals_datastore

Result preview

<table>
<thead>
<tr>
<th>temperature</th>
<th>_dt</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0</td>
<td>2018-07-24 00:00:00.000</td>
</tr>
</tbody>
</table>

Actions

Run query now
Delete
QuickSight

Edition

QuickSight account name

Enter a unique QuickSight account name
You will need this for you and others to sign in.

Notification email address

Enter account notification email address
For QuickSight to send important notifications.

QuickSight capacity region

EU (Ireland)

☐ Enable autodiscovery of data and users in your Amazon Redshift, Amazon RDS and AWS IAM services.

☐ Amazon Athena
Enables QuickSight access to Amazon Athena databases

☐ Amazon S3
Enables QuickSight to auto-discover your Amazon S3 buckets

☐ Amazon S3 Storage Analytics
Enables QuickSight to visualize your S3 Storage Analytics data

☑ Amazon IoT Analytics
Enable QuickSight to visualize your IoT Analytics data

Finish

QuickSight

Data sets

Snowflake

AWS IoT Analytics
Chapter 11: Implementing a Cloud Industrial IoT Solution with Google Cloud
New Project

Project Name *
iiot-book

Project ID: iiot-book. It cannot be changed later. EDIT

Location *

No organisation

Parent organisation or folder

CREATE CANCEL
APIs
Requests (requests/sec)

Google Cloud Platform status
All services normal
→ Go to Cloud status dashboard

Billing
Estimated charges
For the billing period 1–5 Jul 2018
→ View detailed charges

Error Reporting
No sign of any errors. Have you set up Error Reporting?

IoT Core
API Disabled

Reliable real-time messaging
The Cloud IoT API must be enabled before you can view Cloud IoT in the console.

Enable API
IoT Core

Device registries

A device registry allows you to group devices and set properties that they all share, such as connection protocol, data storage location and Cloud Pub/Sub topics. To start connecting devices to Cloud IoT, first create a device registry to place them in. Learn more

Create a device registry
Google Cloud Platform

IoT Core  Registries  CREATE A REGISTRY

Filter registries

<table>
<thead>
<tr>
<th>Region</th>
<th>Protocol</th>
<th>Telemetry Pub/Sub topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Europe-west1</td>
<td>M...</td>
<td>projects/iot-book/topics/signals</td>
</tr>
</tbody>
</table>

No registries selected

Permissions

Please select at least one resource.
Add a device to registry [back-registry].

Device ID

my-iot-device

Device communication:
- Allow
- Block

Authentication (Optional)

Input method:
- Enter manually
- Upload

Public key format:
- RS256
- ES256
- RS256_X509
- ES256_X509

Public key value

-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA7664rQMSiSzNThYioYHJtMnuHlyl7N6sJnNqQ2rYr9qQp86rrsSeksv4zi8gYEsFQ4w4ULRbYzu3v+YWwMDgUa6eBH
C6oNFqG0wT13lYpp Yöcv0L2z+Ky65sw9/7Ino3hxfhggGcaBbVtvE5JXuyieBzm
g0JwkJH5yWohzKQV+8aqzohyqkJAL28L3zzzErVBmYu3i9HAdROXzurtUY7nva1aHY
2Vnhjj+J&3yRa4rJmDazRLFvjBwnuJmQZrrdAigeCbcb1TJM3ulVyeMehk693cki
t7JHBqYQgg43g0L16p0zm3qqa55kYkKnpw1VFHIDz2Fw4o8gL7/7A/jewog8OKH
hwIDAQAB
-----END PUBLIC KEY-----

Public key expiry date (Optional)

Expires on:

11/07/2019, 14:06 CEST

Device metadata (Optional)

Key:
Value:

+ Add attribute

Add
Cancel
Google Cloud Platform

IoT Core

Registry details

Registry ID: iiot-book-registry
Region: europe-west1
Protocol: MQTT, HTTP

Pub/Sub topics for telemetry and device state

View in Stackdriver

Devices
Certificates

Add device

Search: Enter exact device IDs separated by commas

Device ID: my-iot-device
Communication: Allowed
Last seen: 11 Jul 2018, 15:12:05

Cloud IoT Core documentation

mqtt_example -- projectId=iiot-book --cloudRegion=europe-west1 --registryId=iiot-book-registry --deviceName=my-iot-device --privateKeyFile=./certificates/device_private.pem --algorithm=RS256

Google Cloud IoT Core MQTT example.

connect
Publishing message: my-iot-device,signal1,1,1531566749577,GOOD
message received:
Publishing message: my-iot-device,signal1,2,1531566750882,GOOD
Publishing message: my-iot-device,signal1,3,1531566751584,GOOD
Publishing message: my-iot-device,signal1,4,1531566752591,GOOD
Publishing message: my-iot-device,signal1,5,1531566753594,GOOD
Publishing message: my-iot-device,signal1,6,1531566754398,GOOD
Publishing message: my-iot-device,signal1,7,1531566755398,GOOD
Publishing message: my-iot-device,signal1,8,1531566756587,GOOD
Publishing message: my-iot-device,signal1,9,1531566757611,GOOD
Publishing message: my-iot-device,signal1,10,1531566758614,GOOD
Publishing message: my-iot-device,signal1,11,1531566759617,GOOD
Publishing message: my-iot-device,signal1,12,1531566760622,GOOD
Publishing message: my-iot-device,signal1,13,1531566761628,GOOD
Cloud Bigtable

Cloud Bigtable instances

Cloud Bigtable is a fully managed NoSQL database that supports the popular open-source Apache HBase 1.0 API. You can provision Cloud Bigtable instances for your workload, then use the Bigtable HBase client to develop applications using the standard open-source Big Data tools you’re familiar with.

Create instance or Learn more
**Instance name**
For display purposes only

```plaintext
iot-book-storage
```

**Instance ID**
ID is permanent

```plaintext
iot-book-storage
```

**Instance type**
- **Production (recommended)**
- **Development**
  Low-cost instance for development and testing. Does not provide high availability or replication. Can upgrade to Production later.

**Storage type**
Choice is permanent. Applies to all clusters. Affects cost.
- **SSD**
  Lower latency and higher read QPS. Typically used for real-time serving use cases, such as ad serving and mobile app recommendations.
- **HDD**
  Higher latency for random reads. Good performance on scans and typically used for batch analytics, such as machine learning or data mining.

**Clusters**

**Edit item**

**Cluster ID**
ID is permanent.

```plaintext
iot-book-storage-c1
```

**Zone**
Choice is permanent. Determines where cluster data is stored. To reduce latency and increase throughput, store your data near the services that need it.

```plaintext
europe-west2-b
```
Google Cloud Functions

Google Cloud Functions is a lightweight, event-based, asynchronous compute solution that allows you to create small, single-purpose functions which respond to cloud events, without the need to manage a server or a runtime environment.

⚠️ Cloud Functions API not enabled

Enable API
<table>
<thead>
<tr>
<th>Name</th>
<th>Region</th>
<th>Trigger</th>
<th>Memory allocated</th>
<th>Executed function</th>
</tr>
</thead>
<tbody>
<tr>
<td>iot-book-function-1</td>
<td>us-central1</td>
<td>topic: signals</td>
<td>128 MB</td>
<td>helloPubSub</td>
</tr>
</tbody>
</table>

Select a function

**LABELS**

Labels help to organise your resources (e.g. cost, centre: sales or env: prod).

- No functions selected.
const digitalTwin = {};
digitalTwin['signall'] = {upperLimit: 40, lowerLimit: 10};
exports.helloPubSub = (event, callback) => {
  const pubsubMessage = event.data;
  const str = Buffer.from(pubsubMessage.data, 'base64').toString();
  console.log(str);
  const data = str.split(',');
  console.log(
    'device': data[0],
    'tag': data[1],
    'value': data[2],
    'ts': data[3],
    'quality': data[4]
  );
  callback(null, digitalTwin['signall']);
};

const deviceId=data[0];
const tag=data[1];
const value=data[2];
const timestamp = data[3];
Job name
Must be unique among running jobs. Use lowercase letters, numbers and hyphens (-).

projects/iiot-book/topics/signals

Cloud PubSub to GCS Text
A pipeline that reads from a Pub/Sub topic and writes messages text files stored in GCS.
Note that this pipeline assumes no newlines in the body of the Pub/Sub message and thus each message becomes a single line in the output file.

Required parameters
Regional endpoint
Choose where to deploy Cloud Dataflow workers and store metadata for the job.
europe-west1

Input Cloud Pub/Sub topic
Cloud Pub/Sub topic to read the input from. The topic name should be in the format of
projects/<project-id>/topics/<topic-name>

Output Cloud Storage directory
Path and filename prefix for writing output files (ex: gs://bucket-name/path/). This value must end in a slash.
gs://iiot-book/copy/

Output file prefix
The prefix to place on each windowed file (per output-).
raw-

Temporary Location
Path and filename prefix for writing temporary files. e.g.: gs://MyBucket/tmp
gs://iiot-book/tmp

Optional parameters

Run job  Cancel
Chapter 12: Performing a Practical Industrial IoT Solution with Azure
No IoT hub to display.
Try changing your filters if you don’t see what you’re looking for.

Create IoT hub
Create an IoT Hub to help you connect, monitor, and manage billions of your IoT assets. [Learn More]

**PROJECT DETAILS**

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

- **Subscription**: Free Trial
- **Resource Group**: iiot-book-resources
- **Region**: North Europe
- **IoT Hub Name**: iiot-book-hub

[Review + create]  [Next: Size and scale]  [Automation options]
Create stream analytics job

my-iot-job

IoT Hub

my-iot-job - Inputs

Save
// Sample UDF which returns sum of two values.
function main(temperature, flow) {
    return (flow - 60)/temperature;
}
Your query could be put in logs that are in a potentially different geography. Missing some language constructs? Let us know! (Powered by UserVoice - Privacy Policy)

Download results

<table>
<thead>
<tr>
<th>DEVICE</th>
<th>MAX</th>
<th>TEMPERATURE</th>
<th>FLOW</th>
<th>EFFICIENCY</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;my-iot-device&quot;</td>
<td>&quot;2018-07-30T09:23:50Z&quot;</td>
<td>20.20573942559547</td>
<td>76.05785742159547</td>
<td>0.792472570985967</td>
</tr>
<tr>
<td>&quot;my-iot-device&quot;</td>
<td>&quot;2018-07-30T09:23:50Z&quot;</td>
<td>22.8256749725008307</td>
<td>70.03991982548449</td>
<td>0.4448527695628635</td>
</tr>
<tr>
<td>&quot;my-iot-device&quot;</td>
<td>&quot;2018-07-30T09:23:50Z&quot;</td>
<td>23.34216663857835</td>
<td>77.93898349866656</td>
<td>0.7680190603947662</td>
</tr>
<tr>
<td>&quot;my-iot-device&quot;</td>
<td>&quot;2018-07-30T09:23:50Z&quot;</td>
<td>21.361508134146885</td>
<td>62.88626873649436</td>
<td>0.13175974443320923</td>
</tr>
</tbody>
</table>
Chapter 13: Understanding Diagnostics, Maintenance, and Predictive Analytics

- **Contextual anomalies**
- **Threshold**
- **Point anomaly**
- **Collective anomalies**
The diagram illustrates the concept of Remaining Useful Life (RUL) with respect to degradation over cycles. The Y-axis represents degradation, and the X-axis represents cycle time. The threshold is marked, and the prediction at different points (t and t + m) is shown. The uncertainty is indicated with different distributions: large, small, and no uncertainty.

The text box below the diagram categorizes maintenance into Preventive, Predetermined, and Corrective types. Preventive and Predetermined are further classified as Condition Based with options: On request and Scheduled. Corrective maintenance is divided into Immediate and Deferred.
outlier is an anomaly
Chapter 14: Implementing a Digital Twin – Advanced Analytics
Chapter 15: Deploying Analytics on an IoT Platform

- **Development**
  - Preferred IDE: Jupyter, Visual Studio Code
  - Azure Notebook

- **Computation**
  - Azure ML workspace service

- **Deployment**
  - Azure Container
  - Azure Event Hub
  - Azure IoT Hub

**Steps**
- Preparing the data and developing the model
- Training and Testing the model
- Registering, Building an image with the model and Deploying as a web service
Create notebook instance

Amazon SageMaker provides pre-built fully managed notebook instances that run Jupyter notebooks. The notebook instances include example code for common model training and hosting exercises. Learn more

**Notebook Instance settings**

- **Notebook instance name**: ilot-book-notebook
- **Notebook instance type**: ml.t2.medium
- **IAM role**: Choose a role or let us create a role with the AmazonSageMakerFullAccess IAM policy attached.
- **VPC - optional**: No VPC
- **Lifecycle configuration - optional**: No configuration
- **Encryption key - optional**: No Custom Encryption
- **Volume Size In GB - optional**: 5

**Tags - optional**

[Create notebook instance]
Local test (train and predict)
AWS deploy (train and predict)

local_test (dir)
- train_local.sh
- serve_local.sh
- predict_local.sh
- test_dir (dir)
during local test mounted to

rul (dir)
- serve
- nginx.cfg
- wsgi.py → predict.py

Container
Keras

http/https
test.csv

train
train.csv
- input
- data
- training
- config
- output
- model

container/Dockerfile based on Docker Tensorflow Python3