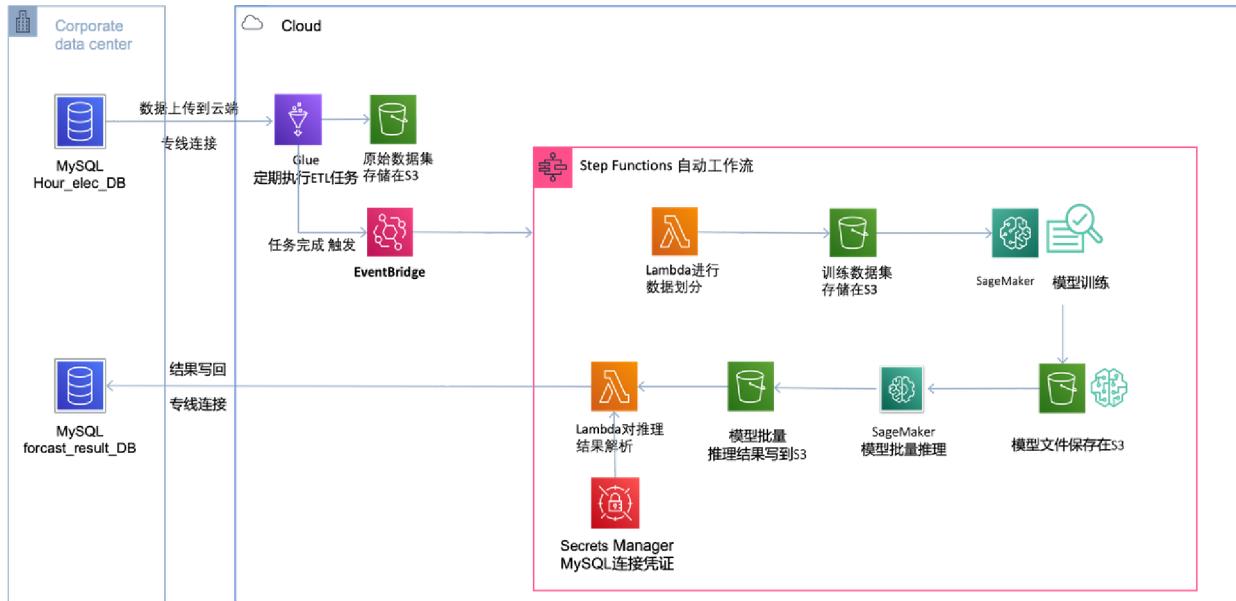


自动化ETL，训练，推理

架构图



1. Glue的设置

添加连接

添加JDBC MySQL数据库连接：进入Glue服务，选择连接-添加连接

The screenshot shows the AWS Glue 'Connections' page with the following details:

- Left Sidebar:** Shows navigation links for Data Catalog, ETL, and ML.
- Header:** '连接' (Connections) with sub-instruction: '连接包含连接到您的数据所需的属性。' (Connections contain attributes required to connect to your data.)
- Buttons:** '添加连接' (Add Connection), '测试连接' (Test Connection), and '操作' (Actions).
- Table:** Displays existing connections:

名称	类型	创建日期	上次更新时间
hour_elec_connection	JDBC	1 六月 2021 6:52 上午 UTC+8	1 六月 2021 7:11 上午 UTC+8
mysql	JDBC	31 五月 2021 2:47 下午 UTC+8	1 六月 2021 6:32 上午 UTC+8
redshift-cluster-1	JDBC	8 三月 2021 10:42 下午 UTC+8	8 三月 2021 10:42 下午 UTC+8

进入连接向导，设置连接属性：连接名称，连接类型选择JDBC

设置您的连接属性。

有关更多信息，请参阅[使用连接](#)。

连接名称

hour_elec_connection

连接类型

JDBC



需要 SSL 连接

When selected, connection fails if unable to connect over SSL.

描述（可选）

输入描述...

下一步

设置JDBC数据库连接访问属性

设置对您的数据存储的访问。

有关更多信息，请参阅[使用连接](#)。

JDBC URL

jdbc:mysql://172.31.17.239:3306/hour_elec

大多数数据库引擎的 JDBC 语法为 jdbc:protocol://host:port/databasename。

SQL 服务器语法为 jdbc:sqlserver://host:port;databaseName=db_name。Oracle 语法为 jdbc:oracle:thin://@host:port/service_name。有关更多差异，请参阅[使用连接](#)。

用户名

sean

密码

VPC

选择包含您的数据存储的 VPC 名称。

vpc-ba8fc4de



子网

选择 VPC 内的子网。

subnet-654dd801 | defaultsubnet_172.31.16.0



安全组

选择允许访问您的 VPC 中的数据存储的一个或多个安全组。AWS Glue 将这些安全组与您的子网所挂载的 ENI 关联。为了允许 AWS Glue 组件进行通信且同时阻止其他网络的访问，至少一个选定的安全组必须为所有 TCP 端口指定自引用入站规则。

组 ID

▼ 组名

sg-009e523fa677ece4e

mysql

预览设置，点击完成

连接属性

名称	hour_elec_connection
类型	JDBC
需要 SSL 连接	false
描述 (可选)	-

连接访问

JDBC URL	jdbc:mysql://172.31.17.239:3306/hour_elec
用户名	sean
VPC ID	vpc-ba8fc4de
子网	subnet-654dd801
安全组	sg-009e523fa677ece4e

返回完成

添加爬网程序

AWS Glue

爬网程序 爬网程序与数据存储连接，按照优先级列表调用分类器以确定数据架构，然后在数据目录中创建元数据表。

正在显示: 1 - 13 < > ⌂ 用户首页

操作	添加爬网程序	运行爬网程序	操作
	glue-lab-crawler	Ready	日志 40 秒 40 秒 0 15
	hour_elec_crawler	Ready	日志 2 分钟 2 分钟 1 0
	meter-data-anomaly-crawler-cn-north-1	Ready	0 秒 0 秒 0 0
	meter-data-business-aggregated-daily	Ready	日志 2 分钟 2 分钟 1 0
	meter-data-business-aggregated-monthly	Ready	日志 49 秒 49 秒 1 0
	meter-data-business-daily-crawler-cn-north-1	Ready	日志 1 分钟 1 分钟 1 0
	meter-data-clean-crawler-cn-north-1	Ready	日志 43 秒 43 秒 0 0
	meter-data-forecast-crawler-cn-north-1	Ready	日志 44 秒 44 秒 0 1
	meter-data-raw-crawler-cn-north-1	Ready	日志 50 秒 50 秒 1 1
	mysql	Ready	日志 48 秒 48 秒 0 8
	tianrun-crawler	Ready	日志 42 秒 42 秒 1 0
	tianrun-parquet-crawler	Ready	日志 44 秒 44 秒 0 1

添加有关您的爬网程序的信息

爬网程序名称

hour_elec_crawler

- ▶ 标签、描述、安全配置和分类器（可选）

下一步

Specify crawler source type

Choose Existing catalog tables to specify catalog tables as the crawler source. The selected tables specify the data stores to crawl. This option doesn't support JDBC data stores.

Crawler source type

- Data stores
- Existing catalog tables

Repeat crawls of S3 data stores

- Crawl all folders

Crawl all folders again with every subsequent crawl.

- Crawl new folders only

Only Amazon S3 folders that were added since the last crawl will be crawled. If the schemas are compatible, new partitions will be added to existing tables.

返回

下一步

添加数据存储，选择JDBC，和上一步创建好的连接

添加数据存储

选择数据存储

JDBC

连接

hour_elec_connection

[添加连接](#)

包含路径

hour_elec/hour_elec_table

您可以替换架构或表的百分号 (%) 字符。对于支持架构的数据库，键入 MyDatabase/MySchema/% 以将 MySchema 中的所有表与 MyDatabase 匹配。Oracle 和 MySQL 不支持路径中的架构，应键入类型 MyDatabase/%。有关哪些 JDBC 数据存储支持架构的信息，请参阅 [使用爬网程序编录表](#)。

▶ 排除模式（可选）

[返回](#)

[下一步](#)

添加另一个数据存储

是
 否

[返回](#)

[下一步](#)

选择一个 IAM 角色

IAM 角色允许爬网程序运行和访问您的 Amazon S3 数据存储。[了解更多](#)

IAM 角色 ⓘ

AWSGlueServiceRole-Default



此角色必须提供类似于 AWS 托管策略 **AWSGlueServiceRole** 的权限以及对您的数据存储的访问权限。

您还可以在 [IAM 控制台上](#) 创建 IAM 角色。

[返回](#)

[下一步](#)

为此爬网程序创建计划

频率

按需运行

返回

下一步

添加数据库

hour_elec_db

► 描述和位置（可选）

Resource link name

Enter resource link name

Shared database suggestions

Choose a database to autofill form

Shared database

Enter database to link to

Shared database owner account ID

Enter an AWS account ID

创建

配置爬网程序的输出

数据库 1

hour_elec_db

添加数据库

已添加到表的前缀 (可选)

键入已添加到表名称的前缀

► S3 数据的分组行为 (可选)

► 配置选项 (可选)

返回
下一步

设置完成后运行爬网程序：

AWS Glue

爬网程序 爬网程序与数据存储连接，按照优先级列表调用分类器以确定数据架构，然后在数据目录中创建元数据表。

名称	计划	状态	日志	上次运行时间	平均运行时间	表已更新	表已添加
glue-lab-crawler		Ready	日志	40 秒	40 秒	0	15
hour_elec_crawler		Ready	日志	2 分钟	2 分钟	1	0
meter-data-anomaly-crawler-cn-north-1		Ready		0 秒	0 秒	0	0
meter-data-business-aggregated-daily		Ready	日志	2 分钟	2 分钟	1	0
meter-data-business-aggregated-monthly		Ready	日志	49 秒	49 秒	1	0

查看数据库和表

正常执行完爬网程序后，在表，可以看到爬的表

AWS Glue

表 表是表示数据 (包括架构) 的元数据定义。表可以用作作业定义中的源或目标。

名称	数据库	位置	分类	上次更新时间	已弃用
hour_elec_hour_elec_table	hour_elec_db	hour_elec.hour_elec_table	mysql	1 六月 2021 5:16 下午 UTC+8	

点击这个表，编辑架构，修改数据类型

表 > hour_elec_hour_elec_table

上次更新时间 16月 2021 09:16 上午 表 版本 (当前版本) ▾

[编辑表](#) [删除表](#)

名称: hour_elec_hour_elec_table
 描述:
 数据库: hour_elec_db
 分类: mysql
 位置: hour_elec.hour_elec_table
 连接: hour_elec_connection
 已弃用: 否
 上次更新时间: Tue Jun 01 09:16:06 GMT+800 2021
 Serde 参数: -

表属性: [UPDATED_BY_CRAWLER](#) [hour_elec_crawler](#) CrawlerSchemaSerializerVersion: 1.0 CrawlerSchemaDeserializerVersion: 1.0 compressionType: none typeOfData: table

架构

正在显示: 1 - 2 个 (共 2 个) < >

列名称	数据类型	分区键	评论
1 datetime	date		
2 elec_value	float		

编辑架构 :

表 > hour_elec_hour_elec_table

上次更新时间 16月 2021 09:16 上午 表 版本 (当前版本) ▾

[添加列](#) 编辑架构 [取消](#) [保存](#)

正在显示: 1 - 2 个 (共 2 个) < >

列名称	数据类型	密钥	评论
1 datetime	date		
2 elec_value	float		

添加作业

AWS Glue

作业 作业是执行提取、转换和加载 (ETL) 工作所需的业务逻辑。作业运行由可通过事件安排或驱动的触发器启动。

添加作业 操作 ▾ 按标签和属性筛选 正在显示: 1 - 10 < >

名称	Type	ETL 语言	脚本位置	上次修改时间	作业书签
business_aggregate_daily-cn-north-1	Spark	python	s3://utilityplatform-...	8 三月 2021 10:43 下午 UTC+8	
business_aggregate_monthly-cn-north-1	Spark	python	s3://utilityplatform-...	8 三月 2021 10:43 下午 UTC+8	
business_aggregate_weekly-cn-north-1	Spark	python	s3://utilityplatform-...	8 三月 2021 10:43 下午 UTC+8	
business_daily_to_redshift-cn-north-1	Spark	python	s3://utilityplatform-...	8 三月 2021 10:43 下午 UTC+8	启用
hour_elec_job	Spark	python	s3://aws-glue-script-...	1 六月 2021 7:22 上午 UTC+8	禁用
import_demo_data_to_redshift-cn-north-1	Spark	python	s3://utilityplatform-...	8 三月 2021 10:43 下午 UTC+8	启用
mysql-s3	Spark	python	s3://aws-glue-script-...	31 五月 2021 3:44 下午 UTC+8	禁用
tiannrun-csc-parque	Spark	python	s3://aws-glue-script-...	28 五月 2021 3:36 下午 UTC+8	禁用
transform_clean_to_business_partition-cn-north-1	Spark	python	s3://utilityplatform-...	8 三月 2021 10:43 下午 UTC+8	启用
transform_raw_to_clean-cn-north-1	Spark	python	s3://utilityplatform-...	8 三月 2021 10:43 下午 UTC+8	启用

配置作业属性

名称

IAM 角色

 确保该角色有权访问您的 Amazon S3 源、目标、临时目录、脚本及作业所用的任何库。创建 IAM 角色。

Type

 此作业运行

Glue version

- AWS Glue 生成的拟定脚本 创建脚本
- 您提供的现有脚本
- 您将编写新脚本

脚本文件名

存储脚本所在的 S3 路径

 编辑

临时目录

 编辑

选择一个数据源

正在显示: 1 / 1 < >

名称	数据库	位置	分类
<input checked="" type="radio"/> hour_elec_hour_elec_table	hour_elec_db	hour_elec.hour_elec_table	mysql

选择转换类型

Machine learning transforms are currently not supported for Glue 2.0.

更改架构

更改源数据的架构并创建新的目标数据集

查找匹配记录

使用机器学习在您的源数据中查找匹配记录

返回

下一步

选择一个数据目标：在数据目标中创建表，数据存储选择Amazon S3，格式CSV，设定目标路径为一个S3的路径

选择一个数据目标

- 在数据目标中创建表
 使用数据目录中的表并更新数据目标

数据存储

Amazon S3

格式

CSV

压缩类型

None

连接

- 选择一个 -

添加连接

目标路径

s3://sagemaker-cn-north-1-456370280007/sagemaker/go

返回

下一步

数据架构定义，根据情况修改

Output Schema Definition

验证 AWS Glue 所创建的映射。通过选择映射到目标的其他列来更改映射。您可以清除所有映射并重置为默认的 AWS Glue 映射。AWS Glue 使用定义的映射生成您的脚本。

源			目标		
列名称	数据类型	映射到目标	列名称	数据类型	
datetime	date	datetime	datetime	date	<input type="button" value="x"/> <input type="button" value="▼"/> <input type="button" value="▲"/>
elec_value	float	elec_value	elec_value	float	<input type="button" value="x"/> <input type="button" value="▼"/> <input type="button" value="▲"/>

添加触发器：

AWS Glue

触发器 触发器被激发时会启动作业。

<input type="checkbox"/> 触发器名称	触发器类型
--------------------------------	-------

设置触发器属性，按计划，注意时间是UTC时间，本地时间为UTC+8小时

设置您的触发器属性

名称
hour_evel_trigger

标签 (可选)

触发器类型
 计划 作业事件 按需
选择 **Schedule** 可根据计时器激发触发器，选择 **Job events** 可在作业事件匹配您的监视列表时激发触发器，而选择 **On-demand** 可在启动时立即激发触发器。

频率
每日

Start Hour (UTC) 01 : **Start Minute** 00

选择要触发的作业：



2. 设置Step Functions

创建状态机：

定义状态机：选择使用代码段创作，标准类型

The visual diagram shows a 'Start' state leading to a 'Generate dataset' state, which then leads to a 'Train Step' state.

按照如下json代码，进行定义：

```
{
  "StartAt": "Generate dataset",
  "States": {
    "Generate dataset": {
      "Resource": "arn:aws-cn:lambda:cn-north-1:456370280007:function:SageMaker-DeepAR-Train-Step",
      "Type": "Task",
      "Next": "Train Step",
      "Catch": [
        {
          "ErrorEquals": [
            "States.TaskFailed"
          ],
          "Next": "ML Workflow failed"
        }
      ]
    },
    "Train Step": {
      "Resource": "arn:aws-cn:states:::sagemaker:createTrainingJob.sync",
      "Parameters": {
        "AlgorithmSpecification": {
          "TrainingImage": "390948362332.dkr.ecr.cn-north-1.amazonaws.com.cn/forecasting-deepar:1.0",
          "TrainingInputMode": "File"
        },
        "OutputDataConfig": {
          "S3OutputPath.$": "$.body.output"
        },
        "StoppingCondition": {
          "MaxRuntimeInSeconds": 86400
        },
        "ResourceConfig": {
          "InstanceCount": 1,
          "InstanceType": "ml.c5.2xlarge",
          "VolumeSizeInGB": 30
        },
        "RoleArn": "arn:aws-cn:iam::456370280007:role/service-role/AmazonSageMaker-Executor-Role"
      },
      "InputDataConfig": [
        {
          "DataSource": {
            "S3DataSource": {
              "S3DataType": "S3Prefix",
              "S3Uri.$": "$.body.train",
              "S3DataDistributionType": "FullyReplicated"
            }
          },
          "ChannelName": "train"
        },
        {
          "DataSource": {
            "S3DataSource": {
              "S3DataType": "S3Prefix",
              "S3Uri.$": "$.body.test",
              "S3DataDistributionType": "FullyReplicated"
            }
          }
        }
      ]
    }
  }
}
```

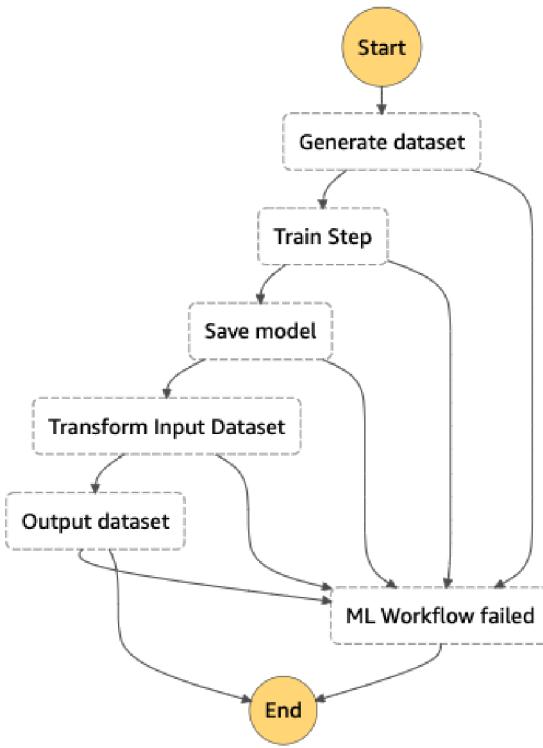
```

        },
        "ChannelName": "test"
    }
],
"HyperParameters": {
    "time_freq": "1H",
    "epochs": "4",
    "early_stopping_patience": "40",
    "mini_batch_size": "256",
    "learning_rate": "1E-3",
    "context_length": "672",
    "prediction_length": "168"
},
"TrainingJobName.$": "$.body.jobname"
},
>Type": "Task",
"ResultPath": "$.taskresult",
"Next": "Save model",
"Catch": [
{
    "ErrorEquals": [
        "States.TaskFailed"
    ],
    "Next": "ML Workflow failed"
}
]
},
"Save model": {
    "Parameters": {
        "ExecutionRoleArn": "arn:aws-cn:iam::456370280007:role/service-role/AmazonSageMaker-ExecutionRole-1JLWVQDZ",
        "ModelName.$": "$.body.modelname",
        "PrimaryContainer": {
            "Environment": {},
            "Image": "390948362332.dkr.ecr.cn-north-1.amazonaws.com.cn/forecasting-deepai:1.0",
            "ModelDataURL.$": "$['taskresult']['ModelArtifacts']['S3ModelArtifacts']"
        }
    },
    "Resource": "arn:aws-cn:states:::sagemaker:createModel",
    "Type": "Task",
    "ResultPath": "$.taskresult",
    "Next": "Transform Input Dataset",
    "Catch": [
{
    "ErrorEquals": [
        "States.TaskFailed"
    ],
    "Next": "ML Workflow failed"
}
]
},
"Transform Input Dataset": {
    "Resource": "arn:aws-cn:states:::sagemaker:createTransformJob.sync",
    "Parameters": {
        "TransformJobName.$": "$.body.jobname",
        "ModelName.$": "$.body.modelname",

```

```
    "TransformInput": {
        "DataSource": {
            "S3DataSource": {
                "S3DataType": "S3Prefix",
                "S3Uri.$": "$.body.test"
            }
        },
        "SplitType": "Line"
    },
    "TransformOutput": {
        "S3OutputPath.$": "$.body.output",
        "AssembleWith": "Line"
    },
    "TransformResources": {
        "InstanceCount": 1,
        "InstanceType": "ml.m5.xlarge"
    },
    "BatchStrategy": "SingleRecord"
},
"Type": "Task",
"Next": "Output dataset",
"Catch": [
    {
        "ErrorEquals": [
            "States.TaskFailed"
        ],
        "Next": "ML Workflow failed"
    }
]
},
"Output dataset": {
    "Resource": "arn:aws-cn:lambda:cn-north-1:456370280007:function:SageMaker-DeepAF",
    "Type": "Task",
    "End": true,
    "Catch": [
        {
            "ErrorEquals": [
                "States.TaskFailed"
            ],
            "Next": "ML Workflow failed"
        }
    ]
},
"ML Workflow failed": {
    "Cause": "SageMakerProcessingJobFailed",
    "Type": "Fail"
}
}
```

自动生成流程图：



指定详细信息：

指定详细信息

名称	
状态机名称 <input type="text" value="MyStateMachine-DeepAR"/>	
必须介于 1 到 80 个字符之间。可使用字母数字字符、短划线或下划线。	
权限	
执行角色 <small>用于定义状态机在执行过程中有权访问的资源的 IAM 角色。要创建自定义角色，请转到IAM 控制台</small>	
<input type="radio"/> 创建新的角色 <small>让 Step Functions 根据您的状态机定义和配置详细信息为您创建新角色。</small>	
<input checked="" type="radio"/> 选择现有角色 	
<input type="radio"/> 输入角色 ARN 	
现有角色 <div style="border: 1px solid #ccc; padding: 2px; width: 100%;">StepFunctions-MyStateMachine-DeepAR-role-c18b4dbe</div>	

按照如下策略创建角色：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:MyLambdaFunction"
    }
  ]
}
```

```
        "Effect": "Allow",
        "Action": [
            "batch:DescribeJobs",
            "batch:SubmitJob",
            "batch:TerminateJob",
            "dynamodb>DeleteItem",
            "dynamodb:GetItem",
            "dynamodb:PutItem",
            "dynamodb:UpdateItem",
            "ecs:DescribeTasks",
            "ecs:RunTask",
            "ecs:StopTask",
            "glue:BatchStopJobRun",
            "glue:GetJobRun",
            "glue:GetJobRuns",
            "glue:StartJobRun",
            "lambda:InvokeFunction",
            "sagemaker>CreateEndpoint",
            "sagemaker>CreateEndpointConfig",
            "sagemaker>CreateHyperParameterTuningJob",
            "sagemaker>CreateModel",
            "sagemaker>CreateProcessingJob",
            "sagemaker>CreateTrainingJob",
            "sagemaker>CreateTransformJob",
            "sagemaker>DeleteEndpoint",
            "sagemaker>DeleteEndpointConfig",
            "sagemaker>DescribeHyperParameterTuningJob",
            "sagemaker>DescribeProcessingJob",
            "sagemaker>DescribeTrainingJob",
            "sagemaker>DescribeTransformJob",
            "sagemaker>ListProcessingJobs",
            "sagemaker>ListTags",
            "sagemaker>StopHyperParameterTuningJob",
            "sagemaker>StopProcessingJob",
            "sagemaker>StopTrainingJob",
            "sagemaker>StopTransformJob",
            "sagemaker>UpdateEndpoint",
            "sns:Publish",
            "sns:SendMessage"
        ],
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "iam:PassRole"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "iam:PassedToService": "sagemaker.amazonaws.com"
            }
        }
    },
    {
```

```

        "Effect": "Allow",
        "Action": [
            "events:)"
        ],
        "Resource": [
            "arn:aws-cn:events:*::rule/*"
        ]
    }
}

```

3.设置EventBridge规则

The screenshot shows the 'Amazon EventBridge > 事件 > 个规则' (Create Rule) page. On the left, there's a sidebar with 'Getting started', '事件' (Events), '事件总线' (Event Bus), and '个规则' (Create Rule). The main area has a heading '个规则' (Create Rule) and a sub-section '选择事件总线' (Select Event Bus) with a dropdown set to 'default'. Below is a table titled '个规则 (10/10)' containing three rows of existing rules, each with columns for '名称' (Name), '状态' (Status), '类型' (Type), and '描述' (Description). A red '创建规则' (Create Rule) button is at the top right of the table.

This screenshot shows the configuration dialog for a new rule named 'hour_elec_event_rule'. It's divided into two tabs: '名称和描述' (Name and Description) and '定义模式' (Definition Mode).

- 名称和描述** tab:
 - 名称**: hour_elec_event_rule (highlighted in blue)
 - 描述 - 可选**: 输入描述
- 定义模式** tab:
 - 事件模式** (Event Mode):
 - 信息** (Information): 构建模式以匹配事件 (Selected)
 - 计划** (Schedule): 按计划时间调用您的目标
 - 事件匹配模式** (Event Matching Mode):
 - 您可以使用服务提供的预定义模式或创建自定义模式
 - 服务提供的预定义模式**
 - 自定义模式** (Selected)
 - 事件模式** (Event Mode):
 - 保存 (Save) and 取消 (Cancel) buttons
 - Code preview area showing a JSON template for a custom event matching mode:

定义模式：选择事件模式，事件匹配模式选择自定义模式，并且编辑事件模式：

```
{  
  "detail-type": [  
    "Glue Job State Change"  
  ],  
  "source": [  
    "aws.glue"  
  ],  
  "detail": {  
    "jobName": [  
      "hour_elec_job"  
    ],  
    "state": [  
      "SUCCEEDED"  
    ]  
  }  
}
```

选择目标：Step Function状态机，然后点击创建

选择目标

选择在事件与您的事件模式匹配时或计划时间被触发时要调用的目标 (每个规则最多 5 个目标)。

目标

选择在事件与您的事件模式匹配时或计划时间被触发时要调用的目标 (每个规则最多 5 个目标)。

Step Functions 状态机

删除

状态机

MyStateMachine-DeepAR

▶ 配置输入

为此特定资源创建新角色

Amazon_EventBridge_Invoke_Step_Functions_763021589

使用现有角色

[了解更多有关基于 EventBridge 身份的策略的信息。](#)

添加目标

标签 - 可选

密钥 值

输入键 输入值 删除标签

添加标签

取消 创建

4. 设置Amazon Secrets Manager，存储MySQL的密匙信息：

Amazon Secrets Manager

在整个生命周期中轻松轮换、管理和检索密钥

Amazon Secrets Manager 可帮助您限制对您的应用程序、服务和 IT 资源的访问。您可以在数据库凭证、API 密钥和其他密钥的整个生命周期中轻松轮换、管理和检索它们。

入门

您可以存储数据库凭证或任何其他类型的密钥

[存储新的密钥](#)

工作原理

使用 Secrets Manager 来存储、轮换、监控和控制对数据库凭证、API 密钥和 OAuth 令牌等密钥的访问。使用内置集成对 Amazon RDS 上的 MySQL、PostgreSQL 和 Amazon Aurora 启用密钥轮换。您还可以使用 Amazon Lambda 功能启用任意密钥的轮换。要检索密钥，您只需要通过调用 Secrets Manager API 来更换应用程序中的硬编码密钥，无需暴露明文密钥。[了解更多](#)

优势和功能

安全地轮换密钥

轻松轮换密钥并使用户和应用程序能够在不部署代码的情况下检索最新的密钥。

集中保障密钥安全和对其进行审计

使用 Amazon CloudTrail 和 Amazon CloudWatch 等亚马逊云科技 日志记录和监控服务轻松审计和监控密钥。

使用精细权限管理访问

按照 亚马逊云科技 Identity and Access Management 策略使用 精细权限控制对密钥的访问。

按需付费

为您存储在 Amazon Secrets Manager 中的密钥及对这些密钥的使用付费。

相关服务

定价

每个密钥每月 ¥2.752
每 10000 个 API 调用 ¥0.344

30 天免费试用

[了解更多](#)

其他信息

[文档](#)

[教程](#)

[身份验证和访问控制](#)

[最佳实践](#)

存储新的密钥

选择密钥类型

[信息](#)

RDS 数据库的凭证

Redshift 集群的凭证

其他数据库的凭证

其他类型的密钥
(如 API 密钥)

指定要存储在此密钥中的用户名和密码 [信息](#)

用户名

sean

密码

password

显示密码

选择加密密钥

[信息](#)

选择 Amazon KMS 密钥来用于加密您的密钥信息。您可以使用 Amazon Secrets Manager 代表您创建的默认服务加密密钥或您存储在 Amazon KMS 中的客户主密钥(CMK)进行加密。

DefaultEncryptionKey



[添加新密钥](#)

选择该密钥将访问的数据库

[信息](#)



服务器地址

172.31.17.239

数据库名称

hour_elec

端口

3306

密钥名称和描述 信息

密钥名称

为密钥指定一个使您能够轻松找到并管理它的名称。

密钥名称必须仅包含字母数字字符和字符 /_+=. @_

描述 - 可选

最多 250 个字符

标签 - 可选

键 值 - 可选

资源权限 (可选) 信息

添加新资源策略或编辑资源策略以安全地跨 亚马逊云科技 账户访问密钥

复制密钥- 可选

存储新的密钥

i 如果您启用自动轮换，当您存储此密钥时便会立即进行第一次轮换。如果此密钥已使用，您必须将您的应用程序更新为从 Amazon Secrets Manager 中检索密钥。阅读 [轮换入门](#) 指南。

配置自动轮换 - 可选 信息

将 Amazon Secrets Manager 配置为自动轮转此密钥。阅读 [轮换入门](#) 指南。

禁用自动轮换

建议在您的应用程序使用该密钥且尚未更新为使用 Amazon Secrets Manager 时使用。

启用自动轮换

建议在您的应用程序尚未使用此密钥时使用。

选择轮换间隔 信息

此密钥将根据您确定的计划轮换。

30 天

值必须介于 1 到 365 天之间

选择 Amazon Lambda 函数 信息

选择一个有权轮换此密钥的 Amazon Lambda 函数。

创建函数 

取消

上一步

下一个 

示例代码

查看说明如何在您的应用程序中检索密钥的代码示例。

Java | JavaV2 | JavaScript | C# | **Python3** | Ruby | Go

```

1 # You can copy snippets in your app.
2 # If you need more information about configurations or implementing the sample code, visit the AWS docs:
3 # https://aws.amazon.com/developers/getting-started/python/
4
5 import boto3
6 import base64
7 from botocore.exceptions import ClientError
8
9
10 def get_secret():
11     secret_name = "hour_elec_secret"
12     region_name = "cn-north-1"
13
14     # Create a Secrets Manager client
15     session = boto3.Session()
16     client = session.client(
17         service_name='secretsmanager',
18
19

```

 下载适用于 Python 的 [亚马逊云科技 开发工具包](#)

取消

上一步

存储 

5. 创建Lambda函数

生成Lambda的层的软件依赖包

```
vi requirements.txt
pandas
openpyxl
fsspec
s3fs
sagemaker
pymysql

pip3 install -r requirements.txt --target python -i https://opentuna.cn/pypi/web/simple
zip -q -r Pandas.zip python
aws s3 cp Pandas.zip s3://sagemaker-cn-north-1-456370280007/sagemaker/goldwind/
```

创建层

The screenshot shows the AWS Lambda Layers interface. On the left, there's a sidebar with options like '控制面板', '应用程序', '函数', '其他资源' (which is expanded to show '层'), and '相关亚马逊云科技资源'. The main area is titled 'Lambda > 层' and shows a table of layers. The table has columns for '名称' (Name), '版本' (Version), '运行时' (Runtime), and '版本 ARN' (Layer ARN). There are four entries:

名称	版本	运行时	版本 ARN
api-dependencies-cn-north-1	2	python3.7	arn:aws:lambda:cn-north-1:456370280007:layer:api-dependencies-cn-north-1:2
common-app-dependencies-cn-north-1	2	python3.7	arn:aws:lambda:cn-north-1:456370280007:layer:common-app-dependencies-cn-north-1:2
ml-app-dependencies-cn-north-1	2	python3.7	arn:aws:lambda:cn-north-1:456370280007:layer:ml-app-dependencies-cn-north-1:2
stepfunction_ml	3	python3.7	arn:aws:lambda:cn-north-1:456370280007:layer:stepfunction_ml:3

配置层，填写从S3上传的路径，和运行时：

Lambda > 层 > 创建层

创建层

层配置

名称: stepfunction-ml

描述 - 可选

上传 .zip 文件 (未选择)

从 Amazon S3 上传文件 (已选择)

Amazon S3 链接 URL
将 S3 链接 URL 贴到您的函数代码 .zip。
s3://sagemaker-cn-north-1-456370280007/sagemaker/goldwind/Pandas.zip

兼容运行时 - 可选 [信息](#)
选择最多 15 个运行时。

运行时: Python 3.8

许可证 - 可选 [信息](#)

取消 **创建**

创建两个Lambda函数 : SageMaker-DeepAR-Generate-Dataset和SageMaker-DeepAR-Output-Dataset

创建函数 [信息](#)

选择下列选项之一来创建函数。

从头开始创作
从一个简单的 Hello World 示例开始。

使用蓝图
从常用案例的示例代码和配置预设中构建 Lambda 应用程序。

浏览无服务器应用程序存储库
从 Amazon Serverless Application Repository 部署示例 Lambda 应用程序。

基本信息

函数名称: data-generation
输入描述函数用途的名称。
仅使用不带空格的字母、数字、连字符或下划线。

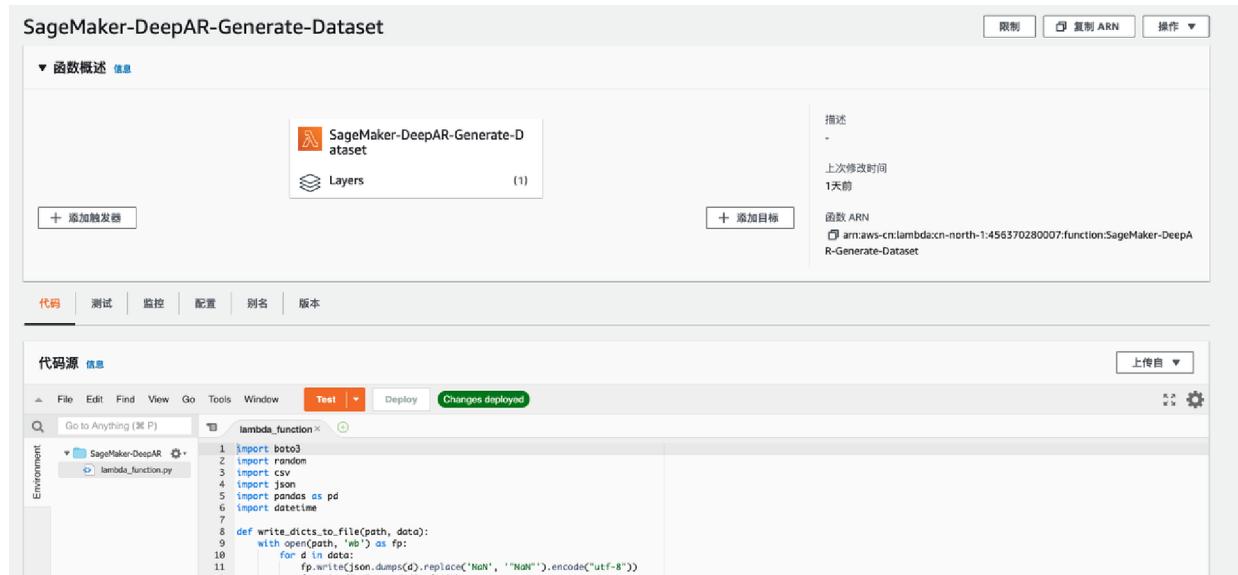
运行时 [信息](#)
选择用来编写函数的语言。请注意，控制台代码编辑器仅支持 Node.js、Python 和 Ruby。
Python 3.8

权限 [信息](#)
默认情况下，Lambda 将创建一个具有将日志上传到 Amazon CloudWatch Logs 权限的执行角色。之后添加触发器时，您可以再对此默认角色进行自定义。

▼ 更改默认执行角色
执行角色
选择定义您的函数权限的角色。要创建自定义角色，请转至 [IAM 控制台](#)。
 创建具有基本 Lambda 权限的新角色
 使用现有角色
 从亚马逊云科技策略模板创建新角色

现有角色
选择您创建的现有角色以与此 Lambda 函数结合使用。此角色必须有权将日志上传到 Amazon CloudWatch Logs。
service-role/SageMaker-DeepAR-Generate-Dataset-role-easgugl7
在 IAM 控制台上查看 [SageMaker-DeepAR-Generate-Dataset-role-easgugl7 角色](#)。

设置Lambda函数：



粘贴如下代码到SageMaker-DeepAR-Generate-Dataset 函数

```

import boto3
import random
import csv
import json
import pandas as pd
import datetime

def write_dicts_to_file(path, data):
    with open(path, 'wb') as fp:
        for d in data:
            fp.write(json.dumps(d).replace('NaN', '"NaN"').encode("utf-8"))
            fp.write("\n".encode('utf-8'))

s3 = boto3.resource('s3')
def copy_to_s3(local_file, s3_path, override=False):
    assert s3_path.startswith('s3://')
    split = s3_path.split('/')
    bucket = split[2]
    path = '/'.join(split[3:])
    buk = s3.Bucket(bucket)

    if len(list(buk.objects.filter(Prefix=path))) > 0:
        if not override:
            print('File s3://{}:{} already exists.\nSet override to upload anyway.\n'.format(s3_path, path))
            return
        else:
            print('Overwriting existing file')
    with open(local_file, 'rb') as data:
        print('Uploading file to {}'.format(s3_path))
        buk.put_object(Key=path, Body=data)
def lambda_handler(event, context):

    bucket_name = 'sagemaker-cn-north-1-456370280007' #改为自己的存储桶

```

```

base_key = 'sagemaker/goldwind/step-function/{}/'.format(datetime.datetime.now().strftime("%Y-%m-%d-%H"))
s3_data_path = "s3://{}{}".format(bucket_name, base_key)

#df = pd.read_excel('s3://'+bucket_name+'/'+ base_key+'hour_elec.xlsx', index_col=0)
#df.to_csv('s3://'+bucket_name+'/'+ base_key+'hour_elec.csv')

dfcsv = pd.read_csv('s3://'+bucket_name+'/'+ base_key+'hour_elec.csv')
df = dfcsv.sort_values(by = 'datetime', ascending = True)
df = df.set_index('datetime')

freq = '1H'
prediction_length = 24*7
training_data = [{ 'start': str(df.index[0]), 'target': list(df.iloc[:-prediction_length].values)}, { 'start': str(df.index[0]), 'target': list(df.iloc[:, 0].values)}]
test_data = [{ 'start': str(df.index[0]), 'target': list(df.iloc[:, 0].values)}]

# print('training_data:', training_data)
# print('test_data:', test_data)

write_dicts_to_file('/tmp/train_'+freq+'.json', training_data)
write_dicts_to_file('/tmp/test_'+freq+'.json', test_data)

copy_to_s3("/tmp/train_"+freq+".json", s3_data_path + "train/train_"+freq+".json",
copy_to_s3("/tmp/test_"+freq+".json", s3_data_path + "test/test_"+freq+".json", overWrite=True)

return {
    'statusCode': 200,
    'body':{
        'jobname':'goldwind-deepar-{}'.format(datetime.datetime.now().strftime('%Y-%m-%d-%H')),
        'modelname':'goldwind-deepar-model-{}'.format(datetime.datetime.now().strftime('%Y-%m-%d-%H')),
        'output': 's3://'+ bucket_name +'/' + base_key + 'output/',
        'train': 's3://'+ bucket_name +'/' + base_key + 'train/train_'+freq+'.json',
        'test': 's3://'+ bucket_name +'/' + base_key + 'test/test_'+freq+'.json'
    }
}

```

粘贴如下代码到SageMaker-DeepAR-Output-Dataset函数

```

import json
import boto3
import datetime
import pymysql

def lambda_handler(event, context):
    s3 = boto3.resource('s3')
    bucket_name = 'sagemaker-cn-north-1-456370280007'
    base_key = 'sagemaker/goldwind/step-function/{}/'.format(datetime.datetime.now().strftime("%Y-%m-%d-%H"))
    output_file = base_key+'output/test_1H.json.out'

    content_object = s3.Object(bucket_name, output_file)
    file_content = content_object.get()['Body'].read().decode('utf-8')
    json_content = json.loads(file_content)
    result=json_content['quantiles']['0.5']

    # connect to MySQL

```

```

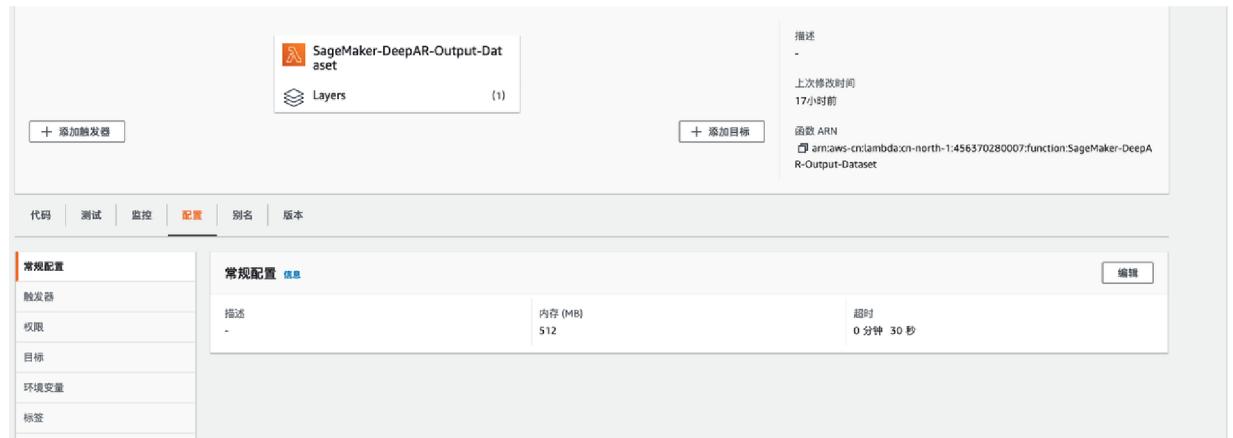
secret_name = "hour_elec_secret"
region_name = "cn-north-1"

# Create a Secrets Manager client
session = boto3.session.Session()
client = session.client(
    service_name='secretsmanager',
    region_name=region_name
)
secret = client.get_secret_value(
    SecretId=secret_name
)
secret_dict = json.loads(secret['SecretString'])

username = secret_dict['username']
password = secret_dict['password']
host = secret_dict['host']
con = pymysql.connect(host = host, user = username, passwd = password, db = 'forca')
cursor = con.cursor()
for i in result:
    cursor.execute("INSERT INTO forcast_result_table (forcast_result) VALUES (%s)")
con.commit()
con.close()
return {
    'statusCode': 200,
    'body': json_content['quantiles']['0.5']
}

```

设置内存和超时：



权限：Lambda执行角色需要：添加EC2 NetworkInterface相关权限，SecretsManager读权限和基本Lamda基本执行权限

The screenshot shows the 'Permissions' tab of a Lambda function configuration. On the left sidebar, '权限' (Permissions) is selected. The main area displays the '执行角色' (Execution Role) section, which includes a role name 'SageMaker-DeepAR-Output-Dataset-role-yfp07ojc' and a '查看角色文档' (View Role Document) button. A '资源摘要' (Resource Summary) section is also present.

权限设定参考

The screenshot shows the 'Permissions policies' section of the IAM Policies page. It lists three attached policies: 'AmazonEC2FullAccess', 'SecretsManagerReadWrite', and 'AWSLambdaBasicExecutionRole'. Each policy has a '策略类型' (Policy Type) column indicating it is an 'AWS 托管策略' (AWS-managed policy). A '附加策略' (Attached Policy) button is at the top left, and a '添加内联策略' (Add inline policy) button is at the top right.

VPC设置：

The screenshot shows the 'VPC' tab of a Lambda function configuration. On the left sidebar, 'VPC' is selected. The main area displays the 'VPC 信息' (VPC Information) section, which includes a VPC ID 'vpc-ba8fc4de', a subnet 'subnet-654dd801', and a security group 'sg-009e523fa677ece4e'. Below this, the '入站规则' (Inbound Rules) section shows a single rule for port All from security group 'sg-009e523fa677ece4e' to port All. A table summarizes the rule details.

安全组 ID	协议	端口	源
sg-009e523fa677ece4e	All	All	sg-009e523fa677ece4e, sg-0275a823d35f5c200