

The background is a vibrant, multi-colored gradient. It features a diagonal split between a blue-purple gradient on the left and a yellow-orange gradient on the right. The text 'AWS re:Invent' is positioned on the left side, with 'AWS' in a smaller font above 're:Invent'.

AWS
re:Invent

WIN407-R

Performance tune SQL Server using AWS Machine Learning

Jugal Shah

Partner Solutions Architect
Amazon Web Services

Agenda

- Machine learning (ML) introduction
- ML stack
- ML process
- Demo
- Lab

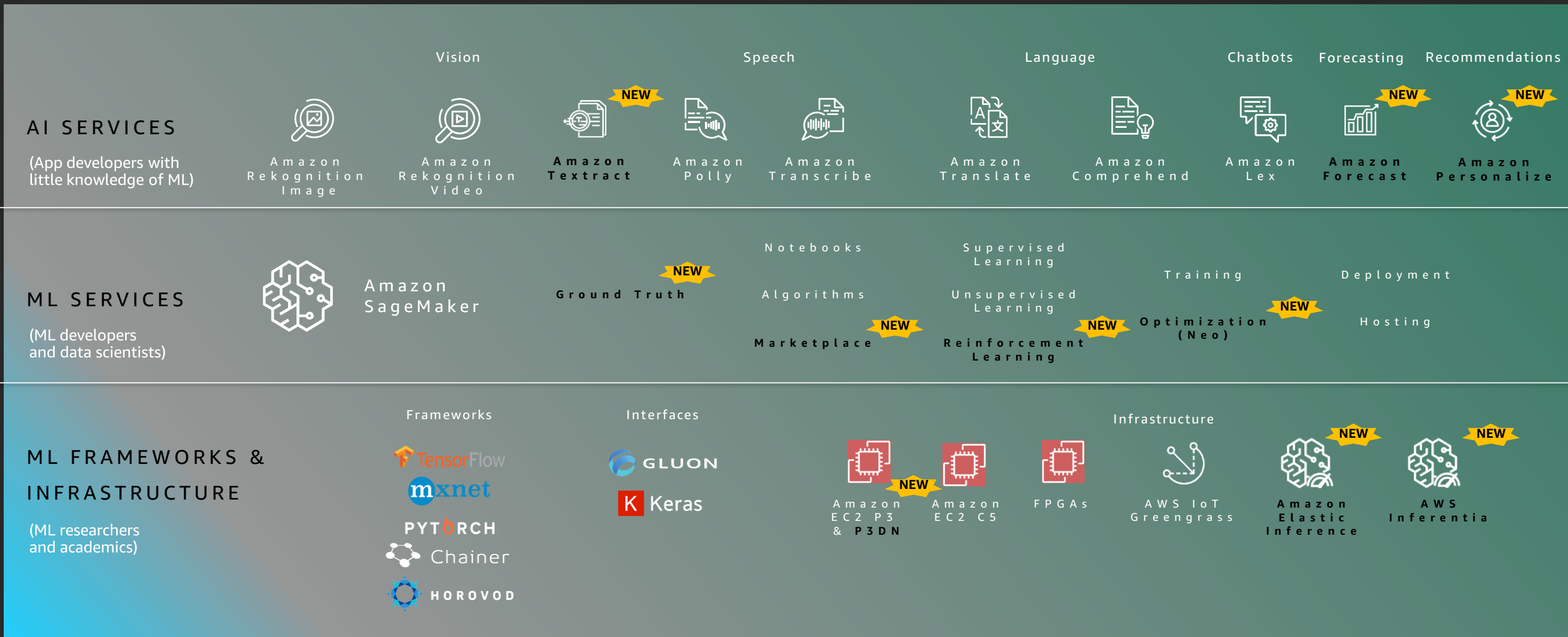
What is machine learning?

Machine learning: AI machines that improve their predictions by learning from large amounts of input data

Deep learning: ML AIs that use complex “artificial neural networks” to greatly improve their predictions and decisions

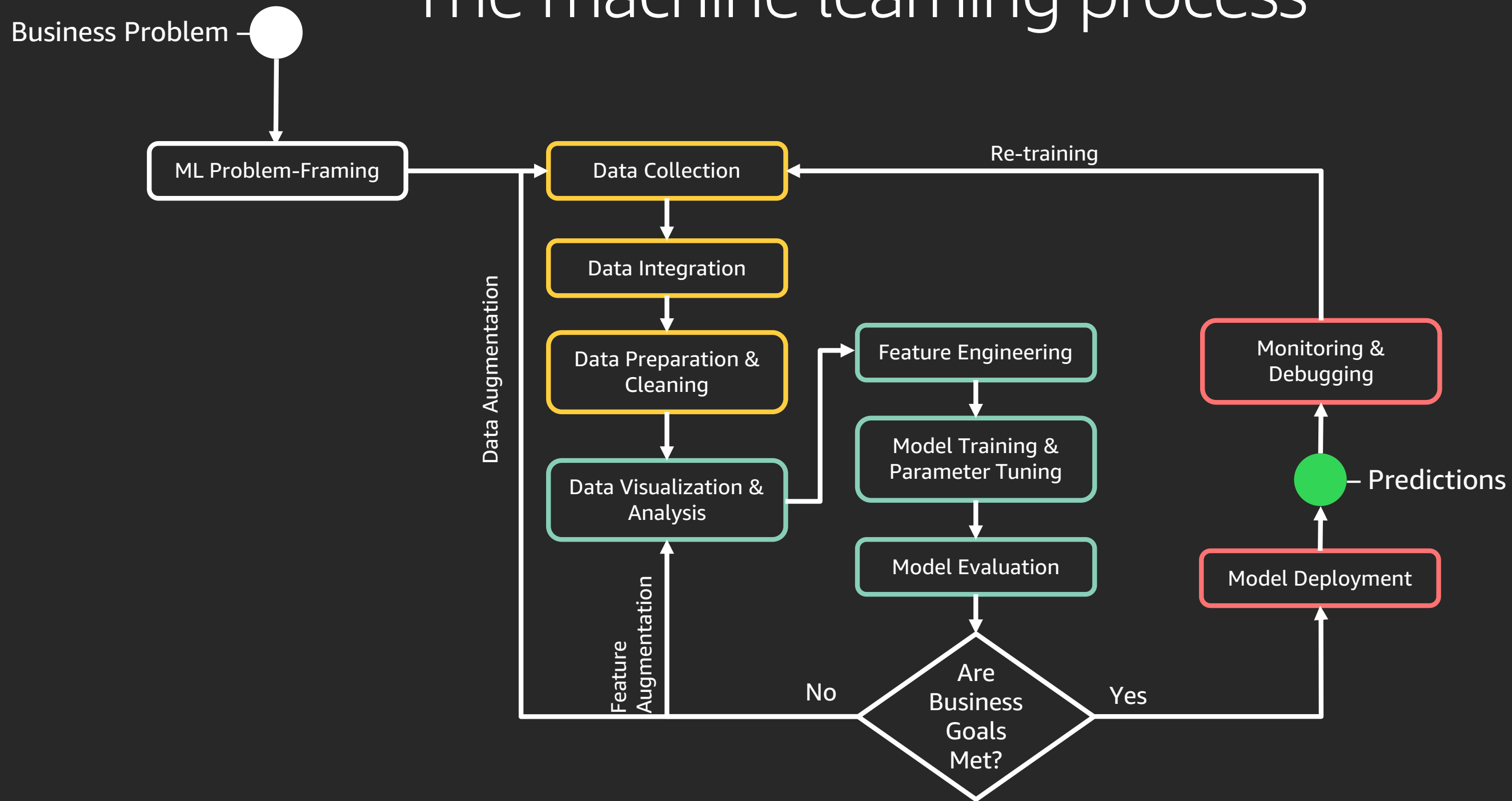
Artificial intelligence (AI): Machines that are programmed to perform the tasks that normally require humans

The Amazon Machine Learning stack



The machine learning process

The machine learning process



Demo

Problem:

Monitoring the multiple RDS metrics and identifying the problematic instance configuration is a time-consuming task. Manual tasks can also lead to human errors.

Solution:

In this session, we create the solution using machine learning to analyze the workload.

RDS: Amazon CloudWatch metrics

CloudWatch (14)



Add instance to compare

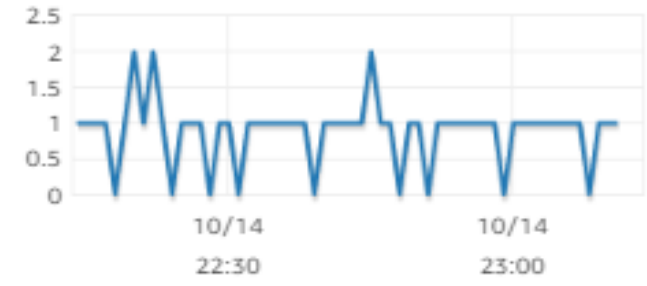
Monitoring

Last Hour

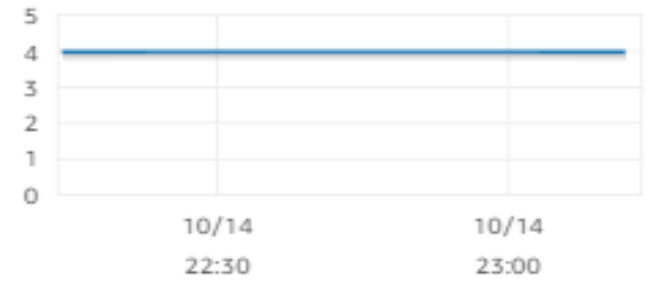
Legend: database-3



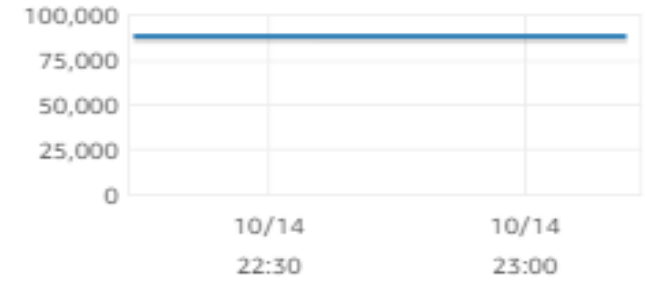
CPU Utilization (Percent)



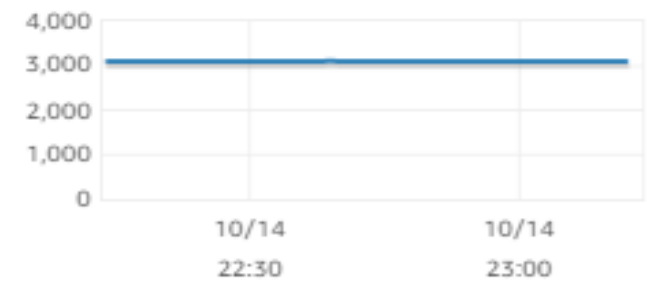
DB Connections (Count)



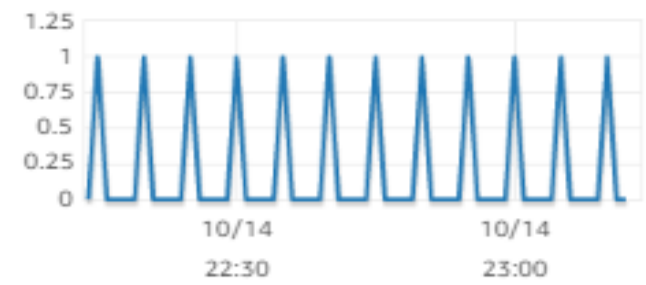
Free Storage Space (MB)



Freeable Memory (MB)



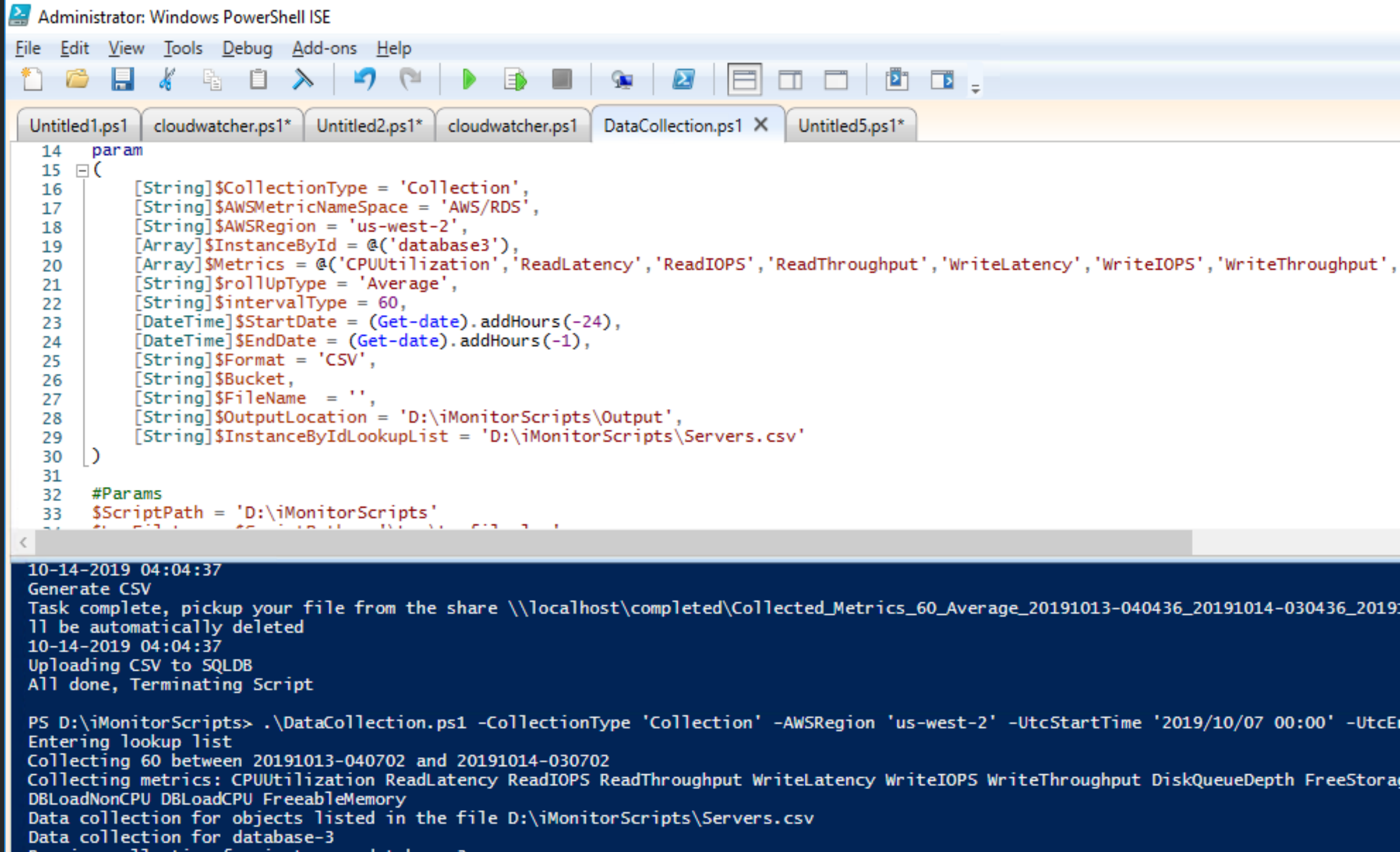
Write IOPS (Count/Second)



Read IOPS (Count/Second)



Data collection



```
Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled1.ps1 cloudwatcher.ps1* Untitled2.ps1* cloudwatcher.ps1 DataCollection.ps1 X Untitled5.ps1*
14 param
15 (
16     [String]$CollectionType = 'Collection',
17     [String]$AWSMetricNameSpace = 'AWS/RDS',
18     [String]$AWSRegion = 'us-west-2',
19     [Array]$InstanceById = @('database3'),
20     [Array]$Metrics = @('CPUUtilization', 'ReadLatency', 'ReadIOPS', 'ReadThroughput', 'WriteLatency', 'WriteIOPS', 'WriteThroughput',
21     [String]$rollUpType = 'Average',
22     [String]$intervalType = 60,
23     [DateTime]$StartDate = (Get-date).addHours(-24),
24     [DateTime]$EndDate = (Get-date).addHours(-1),
25     [String]$Format = 'CSV',
26     [String]$Bucket,
27     [String]$FileName = '',
28     [String]$OutputLocation = 'D:\iMonitorScripts\Output',
29     [String]$InstanceByIdLookupList = 'D:\iMonitorScripts\Servers.csv'
30 )
31
32 #Params
33 $ScriptPath = 'D:\iMonitorScripts'
```

```
10-14-2019 04:04:37
Generate CSV
Task complete, pickup your file from the share \\localhost\completed\Collected_Metrics_60_Average_20191013-040436_20191014-030436_2019:
ll be automatically deleted
10-14-2019 04:04:37
Uploading CSV to SQLDB
All done, Terminating Script

PS D:\iMonitorScripts> .\DataCollection.ps1 -CollectionType 'Collection' -AWSRegion 'us-west-2' -UtcStartTime '2019/10/07 00:00' -UtcEn
Entering lookup list
Collecting 60 between 20191013-040702 and 20191014-030702
Collecting metrics: CPUUtilization ReadLatency ReadIOPS ReadThroughput WriteLatency WriteIOPS WriteThroughput DiskQueueDepth FreeStorage
DBLoadNonCPU DBLoadCPU FreeableMemory
Data collection for objects listed in the file D:\iMonitorScripts\Servers.csv
Data collection for database-3
```

CloudWatch metrics: Collected by PowerShell in CSV format

	A	B	C	D
1	Timestamp	InstanceId	Metric	value
2	10/13/2019 4:07	database-3	ReadThroughput	0.00
3	10/13/2019 4:07	database-3	CPUUtilization	1.00
4	10/13/2019 4:07	database-3	FreeableMemory	13927755776.00
5	10/13/2019 4:07	database-3	WriteThroughput	2303.00
6	10/13/2019 4:07	database-3	DiskQueueDepth	0.00
7	10/13/2019 4:07	database-3	WriteLatency	0.00
8	10/13/2019 4:07	database-3	FreeStorageSpace	107032346624.00
9	10/13/2019 4:07	database-3	ReadLatency	0.00
10	10/13/2019 4:07	database-3	ReadIOPS	0.00
11	10/13/2019 4:07	database-3	WriteIOPS	0.00
12	10/13/2019 4:07	database-3	DatabaseConnections	0.00
13	10/13/2019 4:08	database-3	ReadLatency	0.00
14	10/13/2019 4:08	database-3	FreeStorageSpace	107032346624.00
15	10/13/2019 4:08	database-3	ReadThroughput	0.00
16	10/13/2019 4:08	database-3	FreeableMemory	13927292928.00
17	10/13/2019 4:08	database-3	ReadIOPS	0.00
18	10/13/2019 4:08	database-3	WriteLatency	0.00
19	10/13/2019 4:08	database-3	DatabaseConnections	0.00
20	10/13/2019 4:08	database-3	DiskQueueDepth	0.00
21	10/13/2019 4:08	database-3	WriteThroughput	2756.00
22	10/13/2019 4:08	database-3	CPUUtilization	1.00

Data preparation & cleaning

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the server 'EC2AMAZ-PNHBLKS (SQL Server 14.0.3048.4 - EC2AM)'. The 'Databases' folder is expanded, showing 'System Databases', 'Database Snapshots', and several user databases: 'AdventureWorks2017', 'partners', and 'iMonitor'. Under 'iMonitor', the 'Tables' folder is expanded, listing tables such as 'dbo.mydatetime', 'dbo.tblCSVDataProcessing', 'dbo.tblCWatchMetrics', 'dbo.tblDecisionMakingValues', 'dbo.tblInstanceConfig', and 'dbo.tblserverlist'. The 'Results' window on the right shows a SQL query with three UPDATE statements. The first statement updates 'FreeableMemory' based on 'tblCSVDataProcessing' data. The second statement updates 'DiskQueueDepth' to 5. The third statement updates 'DiskQueueDepth' to 20. Below the query, a table of results is displayed with columns: MyTimestamp, InstanceId, BurstBalance, CPUUtilization, DatabaseConnections, and DBLoad. The table contains 10 rows of data.

```
--select * from tblInstanceConfig

UPDATE A
SET A.FreeableMemory = round(((b.MyValue/1048576)/1024*100)/@memory,2)
FROM tblDecisionMakingValues A, tblCSVDataProcessing B
WHERE A.mytimestamp = B.mytimestamp and B.Metric = 'FreeableMemory'

declare @dq as int
set @dq = @iops/200

UPDATE A
SET A.DiskQueueDepth = 5
FROM tblDecisionMakingValues A, tblCSVDataProcessing B
WHERE A.mytimestamp = B.mytimestamp and B.Metric = 'DiskQueueDepth' and (b.MyValue + 5

UPDATE A
SET A.DiskQueueDepth = 20
```

	MyTimestamp	InstanceId	BurstBalance	CPUUtilization	DatabaseConnections	DBLoad
1	1900-01-01 00:00:00.000	database-3	0	100	0	0
2	2019-10-13 07:10:00.000	database-3	0	67	206	200.016393442623
3	2019-10-13 14:36:00.000	database-3	0	70	206	200
4	2019-10-13 15:05:00.000	database-3	0	68	206	200.033333333333
5	2019-10-14 01:16:00.000	database-3	0	72	206	200
6	2019-10-14 01:45:00.000	database-3	0	64	206	200.050847457627
7	2019-10-13 07:23:00.000	database-3	0	50	206	200
8	2019-10-13 08:38:00.000	database-3	0	48	206	200
9	2019-10-13 11:44:00.000	database-3	0	52	206	0
10	2019-10-13 12:06:00.000	database-3	0	71	206	200

Output file to train the ML model

	A	B	C	D	E	F	G	H
1	CPUUtilization	DiskQueueDepth	FreeableMemory	FreeStorageSpace	Latency	IOPSPercent	ThroughputPercent	Decision
2	67	5	74.19	20	0.7340323	1	32.99	Disk,Low Storage
3	70	6	19.34	20	1.33792073	1	33.78	Memory,Disk,Low Storage,High IO Latency
4	68	7	19.28	86.83	0.92694825	1	32.65	Memory,Disk
5	72	8	18.81	86.81	2	1	32.71	Memory,Disk,High IO Latency
6	64	8	18.77	86.8	0.95284423	1	33.12	Memory,Disk
7	50	9	72.14	95.23	3	1	35.61	Disk,High IO Latency
8	48	10	58.63	92.97	4	1	27.88	Disk,High IO Latency
9	52	11	23.8	87.53	0.50071423	1	29.15	Disk
10	71	12	19.43	86.84	5	1	34.61	Memory,Disk,High IO Latency
11	41	13	19.22	86.83	6	1	31.45	Memory,Disk,High IO Latency
12	71	15	19.03	86.82	7	1	32.32	Memory,Disk,High IO Latency
13	68	15	18.97	86.82	8	1	32.38	Memory,Disk,High IO Latency
14	67	16	18.96	86.82	0.88515428	1	32.54	Memory,Disk
15	72	18	18.81	86.81	9	1	32.51	Memory,Disk,High IO Latency
16	58	19	18.97	86.82	10	1	31.77	Memory,Disk,High IO Latency
17	42	5	18.97	86.81	11	1	33.7	Memory,Disk,High IO Latency
18	84	5	50.91	91.79	0.68257465	1	20.42	Disk
19	53	5	45.63	90.91	12	1	33.88	Disk,High IO Latency
20	66	5	45.38	90.91	0.67843437	1	20.63	Disk

Jupyter Notebook: Build, train & deploy the model

```
jupyter My-Dry-Run-Demo1-Copy1 (autosaved) Python
```

File Edit View Insert Cell Kernel Widgets Help Trusted | conda_python3

```
import pandas as pd
import numpy as np
from datetime import datetime
import io
import sagemaker.amazon.common as smac

import boto3
from sagemaker import get_execution_role
import sagemaker

import matplotlib.pyplot as plt
import seaborn as sns
```

In [18]:

```
role = get_execution_role()
bucket='dryrun-2'
sub_folder = 'Input'
data_key = 'jd-cwdata.csv'
data_location = 's3://{}/{}{}'.format(bucket, sub_folder, data_key)
```

Amazon SageMaker > Models

Models Create endpoint Create endpoint configuration Actions Create model

Search models

Name	ARN	Creation time
cwm991-linear-learner-job-20191014212441	arn:aws:sagemaker:us-west-2:712820468924:model/cwm991-linear-learner-job-20191014212441	Oct 14, 2019 21:29 UTC

Amazon SageMaker > Endpoints

Search endpoints

Name	ARN
cwm991-linear-learner-job-20191014212441	arn:aws:sagemaker:us-west-2:712820468924:endpoint/cwm991-linear-learner-job-20191014212441

Lambda & API Gateway

```
invoke-sagemaker-endpoint
Throttle  Qualifiers ▼
Environment
  invoke-sagemaker-...
  lambda_function.py
  lambda_function X
1 import os
2 import io
3 import boto3
4 import json
5 import csv
6
7 # grab environment variables
8 ENDPOINT_NAME = os.environ['ENDPOINT_NAME']
9 runtime= boto3.client('runtime.sagemaker')
10
11 def lambda_handler(event, context):
12     print("Received event: " + json.dumps(event, indent=2))
13
14     data = json.loads(json.dumps(event))
15     payload = data['data']
16     print(payload)
17
18     response = runtime.invoke_endpoint(EndpointName=ENDPOINT_NAME,
19                                     ContentType='text/csv',
20                                     Body=payload)
21     print(response)
22     result = json.loads(response['Body'].read().decode())
23     print(result)
24     pred = int(result['predictions'][0]['predicted_label'])
25
26     if(pred == 0):
27         return 'Alert : Busy Disk and Low Free Storage'
28     if(pred == 1):
29         return 'Alert : Low free Memory, Busy Disk, Low Free Storage and High IO Latency'
30     if(pred == 2):
```

Method Execution / - POST - Method Test

Make a test call to your method with the provided input

Path
No path parameters exist for this resource. You can define path parameters by using the syntax {myPathParam} in a resource path.

Query Strings
No query string parameters exist for this method. You can add them via Method Request.

Headers
No header parameters exist for this method. You can add them via Method Request.

Stage Variables
No stage variables exist for this method.

Request Body

```
1 {
2   "data": "67,5,74.19,20,0.73,1,32.99"
3 }
```

Request: /
Status: 200
Latency: 1004 ms

Response Body

```
"Alert : Busy Disk"
```

Response Headers

```
{"X-Amzn-Trace-Id": "Root=1-5da557be-f128541d8dff8095b67148a4;Sampled=0", "Content-Type": "application/json"}
```

Logs

```
Execution log for request 8ebaaaa1-a759-4753-bee4-0837f2c589ea
Tue Oct 15 05:23:10 UTC 2019 : Starting execution for request: 8ebaaaa1-a759-4753-bee4-0837f2c589ea
Tue Oct 15 05:23:10 UTC 2019 : HTTP Method: POST, Resource Path: /
Tue Oct 15 05:23:10 UTC 2019 : Method request path: {}
Tue Oct 15 05:23:10 UTC 2019 : Method request query string: {}
Tue Oct 15 05:23:10 UTC 2019 : Method request headers: {}
Tue Oct 15 05:23:10 UTC 2019 : Method request body before transformations: {
  "data": "67,5,74.19,20,0.73,1,32.99"
}
Tue Oct 15 05:23:10 UTC 2019 : Endpoint request URI: https://lambda.us-west-2.amazonaws.com
```


Lab

Thank you!

Jugal Shah

jugashah@amazon.com



Please complete the session survey in the mobile app.