**Amazon Connect Integration Deployment**

**V0.60**

**1/7/2022**

Document History

Version:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version No.** | **Version Date** | **Revised By** | **Reviewed By** | **Approver** |
| 0.60 | 1-Jul-2022 | Robin Lian | Matthew Foster |  |

Change:

| **Version**  | **Description of Change** |
| --- | --- |
| 0.12 | dotnet lambda updates |
| 0.21 | Grafana update, vpc policy added |
| 0.50 | Cloudformation and lambda updates |
| 0.51 | Add [AgentStatusHistory] feature in 5.2, UserGroups |
| 0.53 | Add DELAY var to OnboardObservations and UserGroups |
| 0.55 | Add STORAGE var for all. Timezone fix or onboardCherwell |
| 0.59 | Added ONboardS3 info |
| 0.60 | Runtime upgrade. Java Deprecation |

**Distribution List:**

| **Name** | **Role** | **Department** | **Email** | **Phone** |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |

**CONTENTS**

[1 Scope of Requirements 1](#_Toc110679612)

[1.1 Inclusions 1](#_Toc110679613)

[1.2 Assumptions 1](#_Toc110679614)

[1.3 Prerequisites 1](#_Toc110679615)

[1.4 Constraints 1](#_Toc110679616)

[2 Amazon RDS 2](#_Toc110679617)

[2.1 Database Setup 2](#_Toc110679618)

[2.2 AWS Key Management Service (KMS) 2](#_Toc110679619)

[3 AWS Identity and Access Management (IAM) 3](#_Toc110679620)

[3.1 Roles Setup 3](#_Toc110679621)

[4 AWS Kinesis 4](#_Toc110679622)

[4.1 Kinesis for Connect 4](#_Toc110679623)

[4.2 Optional Automation 5](#_Toc110679624)

[5 AWS Lambda 6](#_Toc110679625)

[5.1 Process AgentEvents 7](#_Toc110679626)

[5.2 Onboard Observations 7](#_Toc110679627)

[5.3 Onboard Metrics 8](#_Toc110679628)

[5.4 Process Contact Trace Records 8](#_Toc110679629)

[5.5 Onboard User Groups 8](#_Toc110679630)

[5.6 Configuration Updates 9](#_Toc110679631)

[6 AWS Cloudwatch 10](#_Toc110679632)

[6.1 Cloudwatch Events 10](#_Toc110679633)

[6.2 Schedule Expressions 11](#_Toc110679634)

1. Scope of Requirements
	1. Inclusions

This document includes the technical implementation steps required to deploy the reporting functionalities onto the AWS cloud.

* 1. Assumptions
* Necessary bare-minimum credentials to deploy the various cloud services in AWS are provided
* Default region used in AWS is Sydney or ap-southeast-2.
* Reference links to official AWS documentations are inserted where more detailed information can be given.
	1. Prerequisites
* Cloudformation templates can be used for deployment.
* Replace the {accountid} variable with the 12-digit amazon account id for actual commands to be executed.
* Dotnet runtime v3.1.26 can be installed for local client testing.

<https://dotnet.microsoft.com/en-us/download/dotnet/thank-you/runtime-desktop-3.1.26-windows-x64-installer>

* AWS Command Line Interface (CLI) can be used to automate deployment efficiently with minimal web UI/console usage. The latest stable version 2.7.12 can be downloaded at

<https://awscli.amazonaws.com/AWSCLIV2.msi>

* 1. Constraints
* There is currently no known business, operational or regulatory constraints observed.
1. Amazon RDS
	1. Database Setup

Create a DB instance of family type General Purposes within the same region (for minimal latency) of disk size 20Gb. m5.large should be sufficient for initial deployment.

<https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_GettingStarted.CreatingConnecting.SQLServer.html>

A database called “AmazonConnect” and a user with read/write permissions to the database needs to be created.

The DB configuration can be upgraded later if further analysis shows cases of throttling, etc.

Do take note of the connection string to be used later. Standard database maintenance tasks needs to be set up for index management.

* 1. AWS Key Management Service (KMS)

AWS KMS is a managed service that is used to create encrypted data such as the connection string to the MsSQL database. This provides a safe way to encrypt sensitive data without involving saving environment variables inside the lambda functions.

<https://aws.amazon.com/blogs/mt/the-right-way-to-store-secrets-using-parameter-store/>

<https://docs.aws.amazon.com/kms/latest/developerguide/create-keys.html#create-keys-console>

aws ssm put-parameter --name "amazonconnect-rdsaccess-nonprod" --value "Server=server\_ip,2383;Database=AmazonConnect;User Id=admin;Password=XXX;MultipleActiveResultSets=True" --type String --overwrite

[https://ap-southeast-2.console.aws.amazon.com/systems-manager/parameters?region=ap-southeast-2#](https://ap-southeast-2.console.aws.amazon.com/systems-manager/parameters?region=ap-southeast-2)

The name and value can also be manually inserted.

Verification:

aws ssm describe-parameters --filters "Key=Name,Values=amazonconnect-rdsaccess-nonprod"

1. AWS Identity and Access Management (IAM)
	1. Roles Setup

“**ConnectLambda**” is the name of the new role to be created, along with a customized role policy for the lambda service. Actual files are attached within this section.

aws iam create-role --role-name ConnectLambda --assume-role-policy-document file://ConnectLambdaTrust.json --description "Access to AWS Connect"

1. {
2. "Version": "2012-10-17",
3. "Statement": {
4. "Effect": "Allow",
5. "Principal": {"Service": ["lambda.amazonaws.com", "events.amazonaws.com"] },
6. "Action": "sts:AssumeRole"
7. }
8. }

aws iam put-role-policy --role-name ConnectLambda --policy-name ConnectLambdaPolicy --policy-document file://ConnectLambdaPolicy.json

This implements the below policy

{

 "Version": "2012-10-17",

 "Statement": [

 {

 "Effect": "Allow",

 "Action": [

 "connect:GetCurrentMetricData",

 "connect:GetMetricData",

 "connect:ListQueues",

 "connect:DescribeUserHierarchyStructure",

 "connect:DescribeUserHierarchyGroup",

 "connect:DescribeUser",

 "connect:ListUserHierarchyGroups",

 "connect:ListUsers",

 "connect:ListRoutingProfiles",

 "ec2:CreateNetworkInterface",

 "ec2:DescribeNetworkInterfaces",

 "ec2:DeleteNetworkInterface",

 "ec2:DescribeSecurityGroups",

 "ec2:DescribeSubnets",

 "ec2:DescribeVpcs",

 "lambda:InvokeFunction",

 "lambda:InvokeAsync",

 "kinesis:DescribeStream",

 "kinesis:ListStreams",

 "kinesis:SubscribeToShard",

 "kinesis:GetShardIterator",

 "kinesis:GetRecords",

 "logs:CreateLogGroup",

 "logs:CreateLogStream",

 "logs:PutLogEvents",

 "ssm:GetParameter",

 "ssm:GetParameters",

 "ssm:GetParametersByPath",

 "ec2:AssignPrivateIpAddresses",

 "ec2:UnassignPrivateIpAddresses"

 ],

 "Resource": "\*"

 }

 ]

}



**Explanations**:

Provides rights to invoke Lambda functions (from AWSLambdaRole template)

Actions: "lambda:InvokeFunction"

Provides read access to Kinesis streams and write permissions to CloudWatch Logs (from AWSLambdaKinesisExecutionRole template)

Actions: "kinesis:SubscribeToShard", " kinesis:GetRecords", "logs:CreateLogGroup", "logs:CreateLogStream", "logs:PutLogEvents"

Provides read permission to encrypted parameter like database connection string (from AmazonSSMReadOnlyAccess template)

Actions: “ssm:getParameter”

The EC2 permissions allow the Lambda functions to access Amazon RDS in a VPC and come from the AWSLambdaVPCAccessExecutionRole

1. AWS Kinesis
	1. Kinesis for Connect

Processing must be deployed at same AWS account as Amazon Connect. Cross-account is not supported via event source mapping.

Use the aws cli to create a kinesis stream called AgentEvents

aws kinesis create-stream --stream-name AgentEvents --shard-count 1

aws kinesis create-stream --stream-name ContactTraceRecords --shard-count 1

Inside the Connect web console under Data streaming tab, assign the stream to “Agent Events”



For “Contact Trace Records”, select “Kinesis Stream” as the output. Then select the correct name for export to the stream.

* 1. Optional Automation

Alternatively, the kinesis streams can be manually switched off outside business hours and re-activated again every morning during weekdays.

<https://docs.aws.amazon.com/en_pv/lambda/latest/dg/with-kinesis.html>

Deletion of streams:

aws kinesis delete-stream --stream-name AgentEvents

aws kinesis delete-stream --stream-name ContactTraceRecords

Note:

Create the event source mapping for the stream by executing the following command, replace the {accountid} variable with the 12-digit amazon account id.

**One time only**

aws lambda create-event-source-mapping --event-source-arn arn:aws:kinesis:ap-southeast-2:{accountid}:stream/AgentEvents --function-name ProcessAgentEvents --batch-size 500 --enabled --starting-position LATEST

take note of the uuid of the event source mapping after creation

To enable:

aws lambda update-event-source-mapping --uuid {uuid} --enabled

To disable kinesis:

aws kinesis delete-stream --stream-name AgentEvents

aws lambda update-event-source-mapping --uuid {uuid} --no-enabled

Verification:

aws lambda list-event-source-mappings

aws kinesis describe-stream-summary --stream-name AgentEvents

1. AWS Lambda

There are various lambda functions to be deployed. All the runtimes are using “.Net Core 3.1” including kinesis streams integration for optimal performance.

To manually invoke the scheduled lambda, a “Amazon CloudWatch” test event can be added in the lambda UI.

<https://docs.aws.amazon.com/lambda/latest/dg/configuration-vpc.html>

The **STORAGE** variable can be set to “pgsql” or “postgresql” to use PostgreSql database, instead of the default Ms Sql Server.

For all functions the ConnectLambda role should be used as the default execution role and architecture should be x86-64. After each function is created upload the zip and edit the handler as defined below, and add the environment variables in the configuration tab.

Since the lambda functions require internet access and SQL access at the same time, the VPC needs private and public subnets. The private subnets require a NAT gateway and a internet gateway must be present in the VPC. The lambda functions should set to run within this VPC, selecting the private subnets for use.

* 1. Process AgentEvents

Create function called ProcessAgentEvents with “dotnet 3.1” runtime. The main purpose is to process kinesis stream events from Amazon Connect and store the results into the database.

Handler: ConnectLambda::ConnectLambda.ProcessAgentEvents::Handler

Trigger : kinesis type, select the AgentEvents Stream

Environment Variables

SQL\_PARAM amazonconnect-rdsaccess-nonprod

* 1. Onboard Observations

Create function called OnboardObservations with “dotnet 3.1” runtime. The main purpose is to poll real-time metrics using the Connect SDK into the database.

Handler: ConnectLambda::ConnectLambda.OnboardObservations::Handler

Environment Variables

INSTANCE\_IDd3089d84-d218-47df-a0bf-41793c9e4141

SQL\_PARAM amazonconnect-rdsaccess-nonprod

DELAY 1 (delay in seconds between runs)

This function needs a higher timeout configuration setting of 1 minute, since it continuously runs during the scheduled time.

* 1. Onboard Metrics

create function called OnboardMetrics with “.Net Core 3.1” runtime. The main purpose is to poll historical metrics using the Connect SDK into the database.

Handler: ConnectLambda::ConnectLambda.OnboardMetrics::Handler

<https://docs.aws.amazon.com/connect/latest/adminguide/historical-metrics.html>

Environment Variables

INSTANCE\_ID<amazonconnectinstanceid>

SQL\_PARAM amazonconnect-rdsaccess-nonprod

* 1. Process Contact Trace Records

create function called ProcessCTR with “.Net Core 3.1” runtime. The main purpose is to process CTR kinesis stream events from Amazon Connect.

Handler: ConnectLambda::ConnectLambda.ProcessCTR::Handler

Environment Variable

SQL\_PARAM amazonconnect-rdsaccess-nonprod

Trigger : kinesis type, select the ContactTraceRecords Stream

* 1. Onboard User Groups

create function called OnboardUserGroups with “.Net Core 3.1” runtime.

Handler: ConnectLambda::ConnectLambda.OnboardUserGroups::Handler

Environment Variables

INSTANCE\_ID<amazonconnectinstanceid>

SQL\_PARAM amazonconnect-rdsaccess-nonprod

DELAY 1 (delay in seconds to prevent rate limit exceeding)

* 1. Configuration Updates

Some parameters of the lambda functions may be updated to improve performance, or lower cost, depending on the data throughput required.

Update sample:

aws lambda update-function-configuration --function-name ProcessAgentEvents --memory-size 256 --timeout 300

1. AWS Cloudwatch
	1. Cloudwatch Events

These events would act as triggers for some lambda functions to poll historical data periodically. A cloudwatch event can be created and a target put in place. Then the permission must be added to allow the lambda to be invoked by the cloudwatch event.

aws events put-rule --name scheduleObs --schedule-expression "rate(2 minutes)" --role-arn "arn:aws:iam::{accountid}:role/ConnectLambda" --state ENABLED

aws events put-targets --rule scheduleObs --targets "Id"="1","Arn"="arn:aws:lambda:ap-southeast-2:{accountid}:function:OnboardObservations"

aws events put-rule --name scheduleHist --schedule-expression "rate(5 minutes)" --role-arn "arn:aws:iam::{accountid}:role/ConnectLambda" --state ENABLED

aws events put-targets --rule scheduleHist --targets "Id"="1","Arn"="arn:aws:lambda:ap-southeast-2:{accountid}:function:OnboardMetrics"

Verification:

aws events list-rules

aws events describe-rule --name scheduleObs

aws events list-targets-by-rule --rule scheduleObs

* 1. Schedule Expressions

Custom periods can be realized via schedule expression, depending on the frequency of data update required.

Cron expression for daily working hours in UTC time

cron(Minutes Hours Day-of-month Month Day-of-week Year)

Australia UTC+11

cron(0/1 21-9 ? \* SUN-FRI \*) OnboardObservations 1min 9am to 7pm

cron(0/3 21-9 ? \* SUN-FRI \*) OnboardMetrics 3min 9am to 7pm

UK GMT

cron(0/1 8-20 ? \* MON-FRI \*) OnboardObservations 1min 9am to 7pm

cron(0/3 8-20 ? \* MON-FRI \*) OnboardMetrics 3min 9am to 7pm

<https://docs.aws.amazon.com/lambda/latest/dg/tutorial-scheduled-events-schedule-expressions.html>