

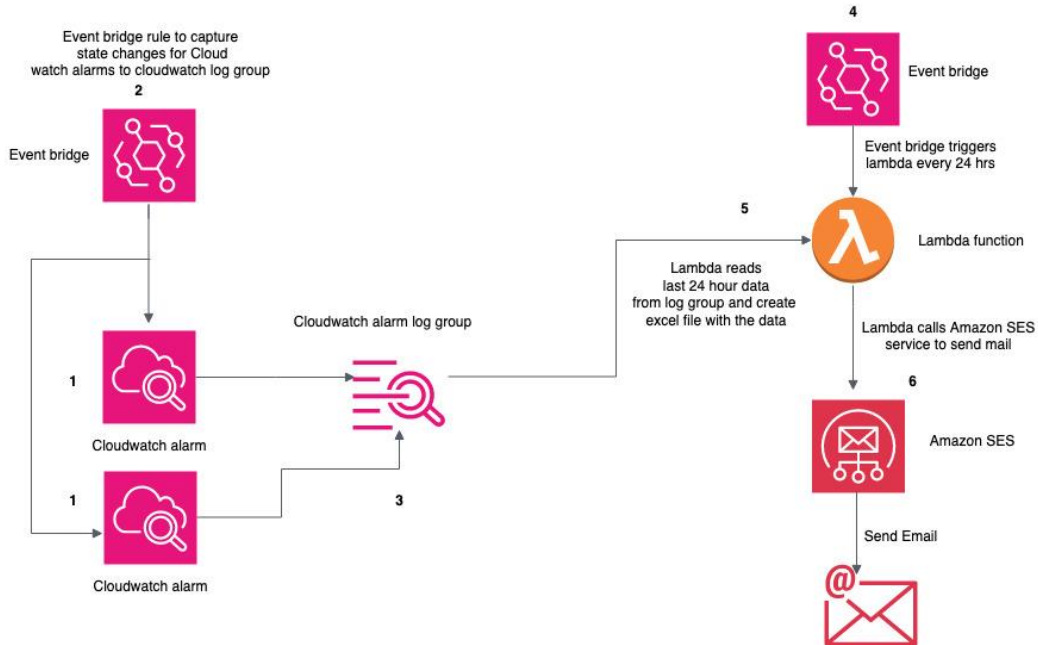
## Usecase:

Create a csv file which includes the following details of all the cloudwatch alarm in a region. This csv file will be created by AWS lambda with Amazon Event Bridge trigger, which triggers AWS lambda function for every 24 hours and store the created csv file in S3 and also sends email attachment with the help of Amazon SES service. The data of the files are for the last 24 hours.

### CSV file columns :

instance name (take from alarm description), instance id, Metric name, Alarm Name ,Alarm Time, Threshold value, Threshold breach value.

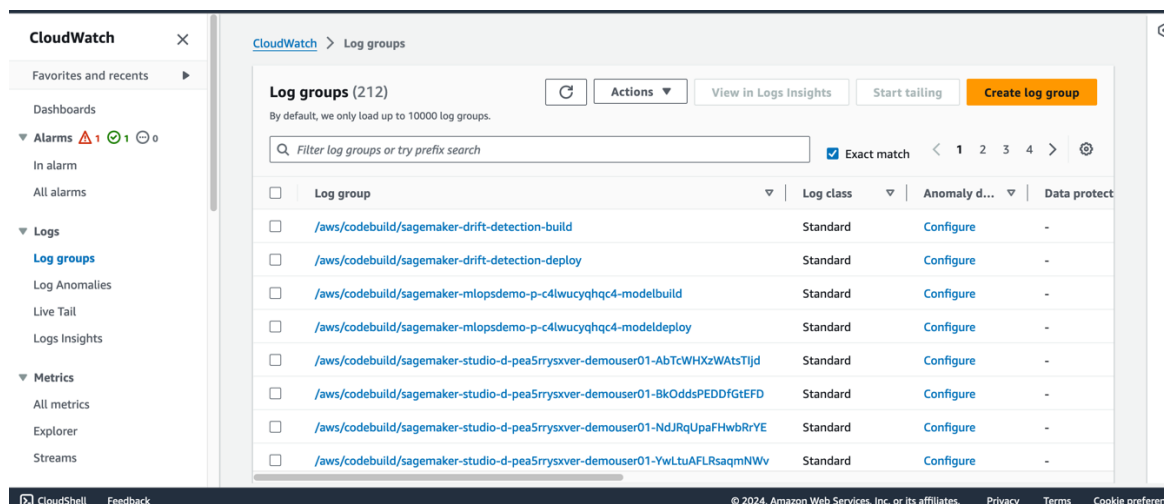
### Architecture Diagram:



The purpose of this code is to generate a report containing details of all Amazon CloudWatch alarms triggered in a specific AWS region within the last 24 hours. The report will be generated automatically by an AWS Lambda function, triggered by an Amazon EventBridge rule every 24 hours. The report will be in CSV format and will include columns such as instance name, instance ID, metric name, alarm name, alarm time, threshold value, and threshold breach value for each triggered alarm. The generated CSV file will be stored in an S3 bucket, and an email with the CSV file attached will be sent using Amazon SES.

### Step 1: Create a Log group in Amazon cloudwatch.

- Navigate to cloudwatch console and click 'Log groups' from the left panel and then Click "Create log group" button in right top corner.



- Enter any name in the “**Log group name**” field and click create button at the right bottom corner of the page.

## Step 2: Create an Amazon EventBridge rule to capture the cloudwatch alarm state changes into the previously created Log group in step 1.

- Navigate to Amazon Event bridge console and click ‘**Create rule**’ button in the Amazon Event bridge home page or from the left panel click “Rules” and then Click “**Create rule**” button in right top corner.

- Enter any name for this rule in the field “Name” and give the description of this rule in “Description” field and then click “Next” button.

- Choose **“AWS Events or event bridge partner events”** for event source radio button. Scroll down and choose **“Use pattern form”** for **“creation method”** section. In the **“Event pattern”** section choose AWS service as **“CloudWatch”** from dropdown and choose **“CloudWatch Alarm State Change”** from the drop down for **“Event type”**. Click **Next**.

Amazon EventBridge

Developer resources

- Learn **Now**
- Sandbox
- Quick starts

Buses

- Event buses
- Rules**
- Global endpoints
- Archives
- Replays

Pipes

- Pipes

Scheduler

- Schedules
- Schedule groups

Integration

- Partner event sources
- API destinations

Schema registry

- Schemas

Documentation

Amazon EventBridge

Developer resources

- Learn **Now**
- Sandbox
- Quick starts

Buses

- Event buses
- Rules**
- Global endpoints
- Archives
- Replays

Pipes

- Pipes

Scheduler

- Schedules
- Schedule groups

Integration

- Partner event sources
- API destinations

Schema registry

- Schemas

Documentation

Amazon EventBridge > Rules > Create rule

Step 1  
[Define rule detail](#)

Step 2  
**Build event pattern**

Step 3  
[Select target\(s\)](#)

Step 4 - optional  
[Configure tags](#)

Step 5  
[Review and create](#)

### Build event pattern Info

**Event source**

Select the event source from which events are sent.

- ☒ **AWS events or EventBridge partner events**  
Events sent from AWS services or EventBridge partners.
- ☐ **Other**  
Custom events or events sent from more than one source, e.g. events from AWS services and partners.
- ☐ **All events**  
All events sent to your account.

**Sample event - optional**

You don't have to select or enter a sample event, but it's recommended so you can reference it when writing and testing the event pattern, or filter criteria.

You can reference the sample event when you write the event pattern, or use the sample event to test if it matches the event pattern. Find a sample event, enter your own, or edit a sample event below. [Learn more about the required fields in a sample event.](#)

Sample event type

- ☒ **AWS events**
- ☐ EventBridge partner events
- ☐ Enter my own

Sample events

Filter by event source and type or by keyword.

Select

1.

Enter the event JSON

Copy

**Creation method**

Method

- ☐ Use schema  
Use an Amazon EventBridge schema to generate the event pattern.
- ☒ **Use pattern form**  
Use a template provided by EventBridge to create an event pattern.
- ☐ Custom pattern (JSON editor)  
Write an event pattern in JSON.

**Event pattern Info**

Event source

AWS service or EventBridge partner as source

AWS services

AWS service

The name of the AWS service as the event source

CloudWatch

Event type

The type of events as the source of the matching pattern

CloudWatch Alarm State Change

Event pattern

Event pattern, or filter to match the events

```
1 {
2   "source": ["aws.cloudwatch"],
3   "detail-type": ["CloudWatch Alarm State Change"]
4 }
```

Copy Test pattern Edit pattern

Cancel Previous **Next**

- Choose **“AWS service”** radio button from the Target 1 section and choose **“CloudWatch log group”** from the **“select a target”** drop down. In the **“Log group”** section drop down, choose the log group name created in the step 1. Click **Next** and then next and then **“create rule”** button in the review and create section.

Amazon EventBridge

Developer resources

- Learn **Now**
- Sandbox
- Quick starts

Buses

- Event buses
- Rules**
- Global endpoints
- Archives
- Replays

Pipes

- Pipes

Scheduler

- Schedules
- Schedule groups

Integration

- Partner event sources
- API destinations

Schema registry

- Schemas

Documentation

Amazon EventBridge

Developer resources

- Learn **Now**
- Sandbox
- Quick starts

Buses

- Event buses
- Rules**
- Global endpoints
- Archives
- Replays

Pipes

- Pipes

Scheduler

- Schedules
- Schedule groups

Integration

- Partner event sources
- API destinations

Schema registry

- Schemas

Documentation

Amazon EventBridge > Rules > Create rule

Step 1  
[Define rule detail](#)

Step 2  
[Build event pattern](#)

Step 3  
**Select target(s)**

Step 4 - optional  
[Configure tags](#)

Step 5  
[Review and create](#)

### Select target(s)

**Permissions**

Note: When using the EventBridge console, EventBridge will automatically configure the proper permissions for the selected targets. If you're using the AWS CLI, SDK, or CloudFormation, you'll need to configure the proper permissions.

**Target 1**

Target types

Select an EventBridge event bus, EventBridge API destination (SaaS partner), or another AWS service as a target.

- ☐ EventBridge event bus
- ☐ EventBridge API destination
- ☒ **AWS service**

Select a target Info

Select target(s) to invoke when an event matches your event pattern or when schedule is triggered (limit of 5 targets per rule)

CloudWatch log group

Log Group:

- ☐ /aws/events/
- ☒ **/aws/events/cloudwatchalarms**

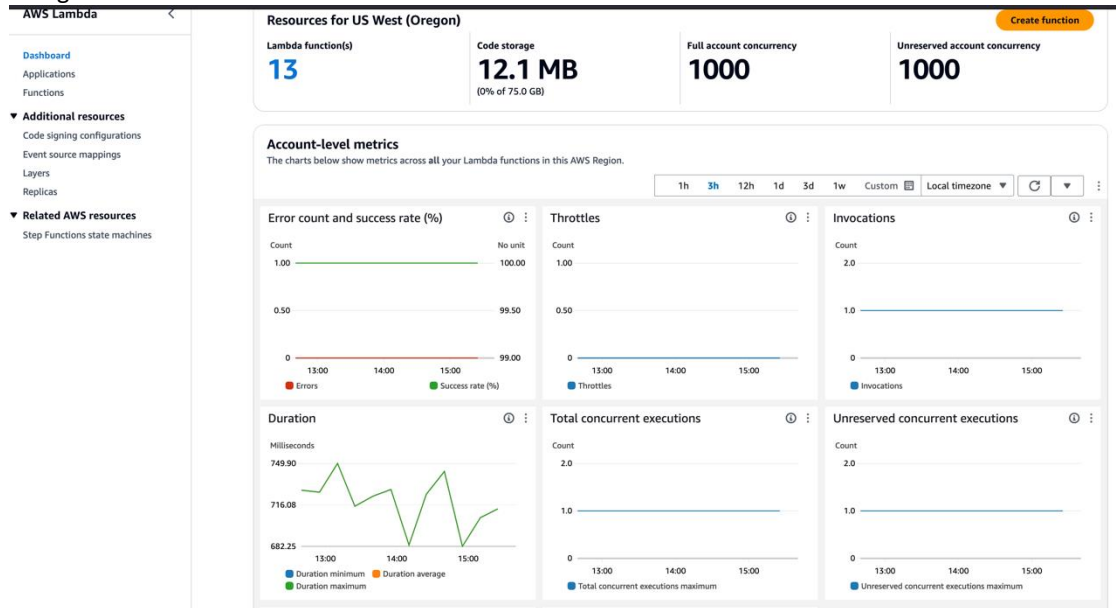
Additional settings

Add another target

Cancel Skip to Review and create Previous **Next**

### Step 3: Create a AWS lambda function with the attached sample code automated-cloudwatch-alarm-reporting-system.py

- Navigate to AWS Lambda dashboard and click create function button.



- Enter the function name and choose python as runtime and click create function button.

The screenshot shows the 'Create function' wizard in the AWS Lambda console. The 'Author from scratch' option is selected. The 'Basic information' section shows the function name 'cloudwatch\_alarm\_monitor\_aws\_lambda', the runtime 'Python 3.12', and the architecture 'x86\_64'. The 'Permissions' section shows the default execution role. The 'Advanced settings' section is collapsed. The 'Create function' button is visible at the bottom right.

- Click "Layers" from the function overview section and then click "Add a layer" button.

The screenshot shows the 'Function overview' section for the 'cloudwatch\_alarm\_monitor\_aws\_lambda' function. The 'Layers' tab is selected, showing a list of layers. The 'Add trigger' button is visible. The 'Description' section shows the function's last modified time, ARN, and URL. The 'Code source' section shows the function's code, which is a Python lambda handler.

1:1 Python Spaces: 4

Code properties [Info](#)

Package size

299.0 byte

SHA256 hash

HAPq9EReJVEC5gLvtc/gyd5vZtd9eiUGF932t0BkY=

Last modified

April 7, 2024 at 03:46 PM GMT+5:30

Runtime settings [Info](#)

Runtime

Python 3.12

Handler [Info](#)

lambda\_function.lambda\_handler

Architecture [Info](#)

x86\_64

▶ Runtime management configuration

Edit

Edit runtime management configuration

Layers [Info](#)

Edit

Add a layer

Merge order	Name	Layer version	Compatible runtimes	Compatible architectures	Version ARN
There is no data to display.					

- 
- Choose “AWSSDKPandas-Python312” as AWS layers and choose latest version from version drop down and then click “Add” button.

Lambda > Layers > Add layer

Add layer

Function runtime settings

Runtime

Python 3.12

Architecture

x86\_64

Choose a layer

Layer source [Info](#)

Choose from layers with a compatible runtime and instruction set architecture or specify the Amazon Resource Name (ARN) of a layer version. You can also [create a new layer](#).

☒ AWS layers  
Choose a layer from a list of layers provided by AWS.

☐ Custom layers  
Choose a layer from a list of layers created by your AWS account or organization.

☐ Specify an ARN  
Specify a layer by providing the ARN.

AWS layers

Layers provided by AWS that are compatible with your function's runtime.

AWSSDKPandas-Python312

▼

Version

6

▼

Cancel

Add

- 
- Click “configuration” tab and then click “permissions” tab. Click on the Role name and it will take us to IAM and then attach policies for this Lambda function to have access to Amazon Cloudwatch, S3 bucket and Amazon SES service.

cloudwatch\_alarm\_monitor\_aws\_lambda

Throttle

Copy ARN

Actions

Function overview [Info](#)

Diagram

Template

cloudwatch\_alarm\_monitor\_aws\_lambda

Layers (1)

+ Add trigger

+ Add destination

Description

-

Last modified

12 minutes ago

Function ARN

arn:aws:lambda:us-west-2:154985105880:function:cloudwatch\_alarm\_monitor\_aws\_lambda

Function URL [Info](#)

-

Code

Test

Monitor

Configuration

Aliases

Versions

General configuration

Triggers

Permissions

Destinations

Function URL

Environment variables

Tags

VPC

RDS databases

Monitoring and operations tools

Execution role

Role name

cloudwatch\_alarm\_monitor\_aws\_lambda-role-jun0714

Resource summary

To view the resources and actions that your function has permission to access, choose a service.

Amazon CloudWatch Logs

3 actions, 2 resources

By action

By resource

Resource	Actions
arn:aws:logs:us-west-2:154985105880:*	Allow: logs:CreateLogGroup

- Click “configuration” tab and then click “general configuration” tab. Click Edit button to update the Timeout and then click save button.

The screenshot shows the AWS Lambda console for the function `cloudwatch_alarm_monitor_aws_lambda`. The **Configuration** tab is selected, and the **General configuration** sub-tab is active. The **Edit** button is visible in the top right corner of the configuration panel. The configuration details shown are:

- Description:** -
- Timeout:** 0 min 3 sec
- Memory:** 128 MB
- Ephemeral storage:** 512 MB
- SnapStart:** None

On the left, a sidebar lists various configuration options: Triggers, Permissions, Destinations, Function URL, Environment variables, Tags, VPC, and RDS databases.

•

The screenshot shows the **Edit basic settings** page for the function `cloudwatch_alarm_monitor_aws_lambda`. The **Basic settings** section is expanded, showing the following configuration options:

- Description - optional:** (empty text field)
- Memory:** 128 MB (with a note: "Your function is allocated CPU proportional to the memory configured. Set memory to between 128 MB and 10240 MB.")
- Ephemeral storage:** 512 MB (with a note: "You can configure up to 10 GB of ephemeral storage (/tmp) for your function. View pricing")
- SnapStart:** None (with a note: "Reduce startup time by having Lambda cache a snapshot of your function after the function has initialized. To evaluate whether your function code is resilient to snapshot operations, review the SnapStart compatibility considerations")
- Timeout:** 5 min 0 sec
- Execution role:** Use an existing role (selected) or Create a new role from AWS policy templates
- Existing role:** service-role/cloudwatch\_alarm\_monitor\_aws\_lambda-role-jux0714 (selected)

At the bottom right, there are **Cancel** and **Save** buttons.

•

- Copy the sample code from “Cloudwatch\_alarm\_dataCSV\_to\_mail\_s3.py” file and paste it in “code” section into `lambda_function.py` file and then click Deploy button.

The screenshot shows the **Code source** page for the function `cloudwatch_alarm_monitor_aws_lambda`. The **Code source** tab is selected, and the **Upload from** button is visible in the top right corner. The code editor shows the following Python code:

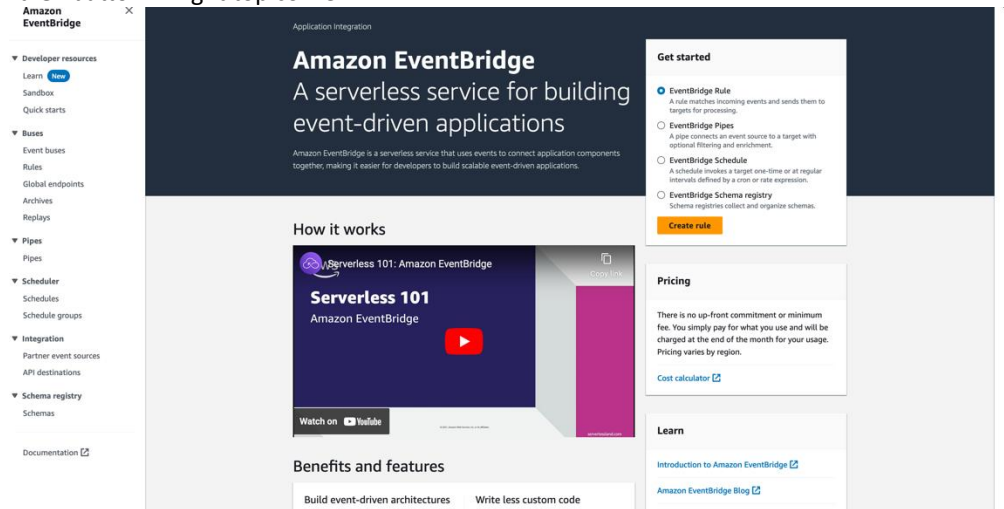
```
1 import json
2 import boto3
3 import datetime
4 from datetime import datetime
5 import time, traceback
6 import pandas as pd
7 import os.path
8 import email
9 from email.mime.multipart import MIMEMultipart
10 from email.mime.text import MIMEText
11 from email.mime.application import MIMEApplication
12
13 def lambda_handler(event, context):
14     try:
15         def convertToMili(value):
16             dt_obj = datetime.strptime(str(value), '%Y-%m-%d %H:%M:%S')
17             result = int(dt_obj.timestamp() * 1000)
18             return result
19     except Exception as e:
20         print(e)
```

The code editor has a menu bar with **File**, **Edit**, **Find**, **View**, **Go**, **Tools**, and **Window**. The **Test** and **Deploy** buttons are visible at the top of the editor.

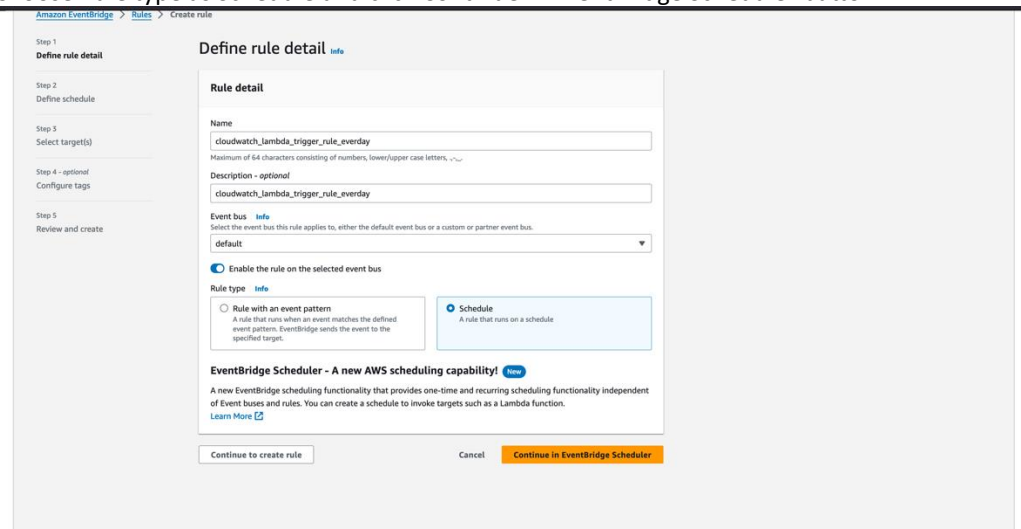
•

**Step 4: Create a Amazon Event Bridge rule to trigger the AWS Lambda function created in step:2, everyday at a specific time of a day.**

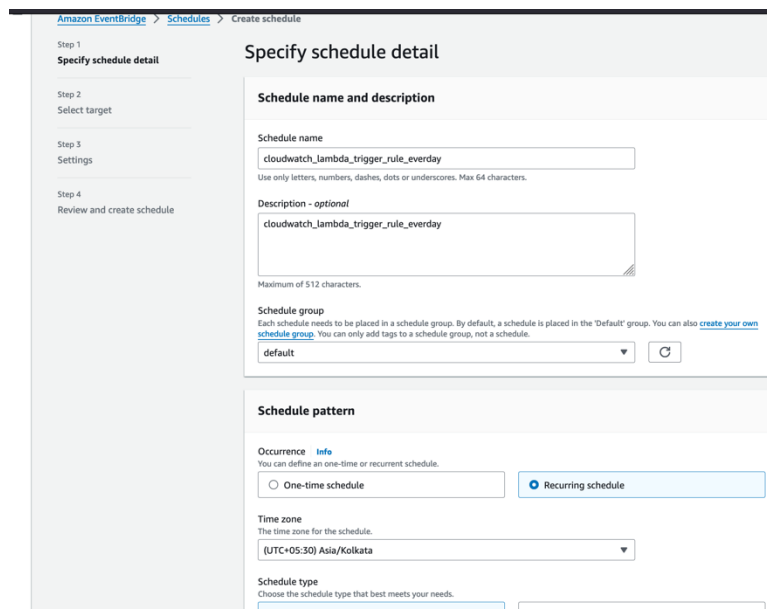
- [https://docs.aws.amazon.com/scheduler/latest/UserGuide/schedule-types.html?icmpid=docs\\_console\\_unmapped#cron-based](https://docs.aws.amazon.com/scheduler/latest/UserGuide/schedule-types.html?icmpid=docs_console_unmapped#cron-based)
- Navigate to Amazon Event bridge console and click **'Create rule'** button in the Amazon Event bridge home page or from the left panel click "Rules" and then Click **"Create rule"** button in right top corner.



- 
- Choose Rule type as Schedule and click Continue in EventBridge Scheduler button.



- 
- Enter the schedule name, description and choose radio button Recurring Schedule.





- Under Schedule type, choose Cron-based schedule. Enter the Cron expression, The below screenshot shows cron expression to run this schedule at everyday 2.30 a.m. Choose Flexible time window as 5 minutes and click Next.

**Cron-based schedule**  
A schedule set using a cron expression that runs at a specific time, such as 8:00 a.m. PST on the first Monday of every month.

**Rate-based schedule**  
A schedule that runs at a regular rate, such as every 10 minutes.

**Cron expression** Info  
Define the cron expression for the schedule

**cron** (   \* \* ? \* )  
Minutes Hours Day of month Month Day of the week Year

**Next 10 trigger dates**  
Date and time are displayed in your current time zone in UTC format, e.g. "Wed, Nov 9, 2022 09:00 [UTC -08:00]" for Pacific time

Mon, 08 Apr 2024 02:30:00 (UTC+05:30)  
Tue, 09 Apr 2024 02:30:00 (UTC+05:30)  
Wed, 10 Apr 2024 02:30:00 (UTC+05:30)  
Thu, 11 Apr 2024 02:30:00 (UTC+05:30)  
Fri, 12 Apr 2024 02:30:00 (UTC+05:30)  
Sat, 13 Apr 2024 02:30:00 (UTC+05:30)  
Sun, 14 Apr 2024 02:30:00 (UTC+05:30)  
Mon, 15 Apr 2024 02:30:00 (UTC+05:30)  
Tue, 16 Apr 2024 02:30:00 (UTC+05:30)  
Wed, 17 Apr 2024 02:30:00 (UTC+05:30)

**Flexible time window**  
If you choose a flexible time window, Scheduler invokes your schedule within the time window you specify. For example, if you choose 15 minutes, your schedule runs within 15 minutes after the schedule start time.

**Timeframe**

**Daylight saving time**  
Amazon EventBridge Scheduler automatically adjusts your schedule for daylight saving time. When time shifts forward in the Spring, if a cron expression falls on a non-existent date, your schedule invocation is skipped. When time shifts backwards in the Fall, your schedule runs only once and does not repeat its invocation. The following invocations occur normally at the specified date and time.

- Choose AWS Lambda radio button and select the lambda function which we created in step 3 and click next.

**Select target**

Step 3 - optional  
[Settings](#)

Step 4  
[Review and create schedule](#)

**Target API** Info  
Select an API that will be invoked as a target for your schedule.

☒ **Templated targets** ☐ All APIs

CodeBuild StartBuild  
Amazon EventBridge PutEvents  
Kinesis Data Streams PutRecord  
AWS Step Functions StartExecution  
CodePipeline StartPipelineExecution  
Kinesis Data Firehose PutRecord  
AWS Lambda Invoke  
Amazon SNS Publish  
Amazon ECS RunTask  
Amazon Inspector V1 StartAssessmentRun  
SageMaker StartPipelineExecution  
Amazon SQS SendMessage

**Invoke** ☒ Universal target definition  
AWS Lambda

**Lambda function**

**Configure version/alias**

**Payload**

- Choose NONE from Action after schedule completion dropdown and click Next and in Review and create schedule page click the review and create.

**Review and create schedule**

Step 3 - optional  
[Settings](#)

Step 4  
[Review and create schedule](#)

**Enable schedule**  
You can choose not to enable the schedule now. You will be able to enable the schedule after it has been created.  
☒ **Enable**

**Action after schedule completion** Info  
If you choose DELETE, EventBridge Scheduler will automatically delete the schedule after it has completed its last invocation and has no future target invocations planned.

**Retry policy and dead-letter queue (DLQ)**

**Retry policy** Info  
By default, EventBridge Scheduler attempts to retry failed invocations for up to 24 hours. You can specify the maximum age of the event and the maximum number of times to retry.

☒ **Retry**

**Maximum age of event - optional**  
The maximum amount of time to keep unprocessed events. The maximum and default value is 24 hours.

hours  minutes

**Retry attempts - optional**  
The maximum number of times to retry when a target returns an error. The maximum value is 185 times.

times

**Dead-letter queue (DLQ)**  
Standard Amazon SQS queues that EventBridge Scheduler uses to store events that couldn't be delivered successfully to a target.

☒ **None**  
☐ Select an Amazon SQS queue in my AWS account as a DLQ  
☐ Specify an Amazon SQS queue in other AWS accounts as a DLQ



Step 2 - optional

[Select target](#)

Step 3 - optional

[Settings](#)

Step 4

**Review and create schedule**

Step 1: Schedule detail

Edit

Schedule detail

Schedule name cloudwatch_lambda_trigger_rule_e verday	Description cloudwatch_lambda_trigger_rule_e verday	Schedule group default
Time zone (UTC+05:30) Asia/Kolkata	Occurrence Recurring	Start date and time -
End date and time -	Flexible time window 5 minutes	

Cron expression

30	2	*	*	?	*
Minutes	Hours	Day of month	Month	Day of week	Year

Next 10 trigger dates

Date and time are displayed in the selected time zone for which this schedule is set in UTC format, e.g. "Wed, Nov 9, 2022 09:00 (UTC - 08:00)"

Mon, 08 Apr 2024 02:30:00 (UTC+05:30)  
Tue, 09 Apr 2024 02:30:00 (UTC+05:30)  
Wed, 10 Apr 2024 02:30:00 (UTC+05:30)  
Thu, 11 Apr 2024 02:30:00 (UTC+05:30)  
Fri, 12 Apr 2024 02:30:00 (UTC+05:30)  
Sat, 13 Apr 2024 02:30:00 (UTC+05:30)  
Sun, 14 Apr 2024 02:30:00 (UTC+05:30)  
Mon, 15 Apr 2024 02:30:00 (UTC+05:30)  
Tue, 16 Apr 2024 02:30:00 (UTC+05:30)  
Wed, 17 Apr 2024 02:30:00 (UTC+05:30)

Step 2: Target

Edit

Target detail