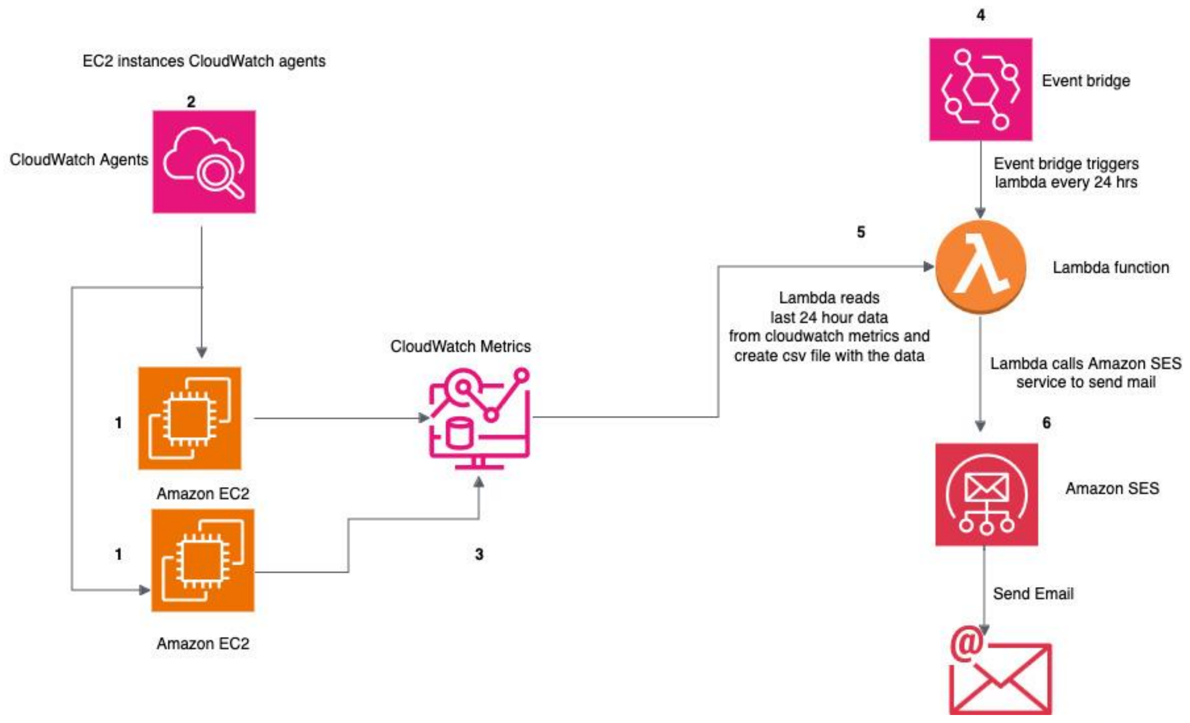
Usecase:

Create a csv file which includes the following details for all the EC2 instances. The csv file will be created by AWS lambda function which gets triggered by Amazon event bridge rule for every 24 hours and store the created file in S3 and also sends email attachment. The data of the files are for the last 24 hours.

CSV file columns :

Instance Name, Instance Id, Average CPU utilization Percent, Max CPU utilization Percent, Average Memory utilization Percent, Max Memory utilization Percent, Root disk total in GB, Root disk used in GB,Root disk free in GB,EBS1 disk total in GB,EBS1 disk used in GB,EBS1 disk free in GB,EBS2 disk total in GB,EBS2 disk used in GB,EBS2 disk free in GB.

Memory usage , root disk, EBS disk details are from the CloudWatch agents configured in each of the EC2 instances.

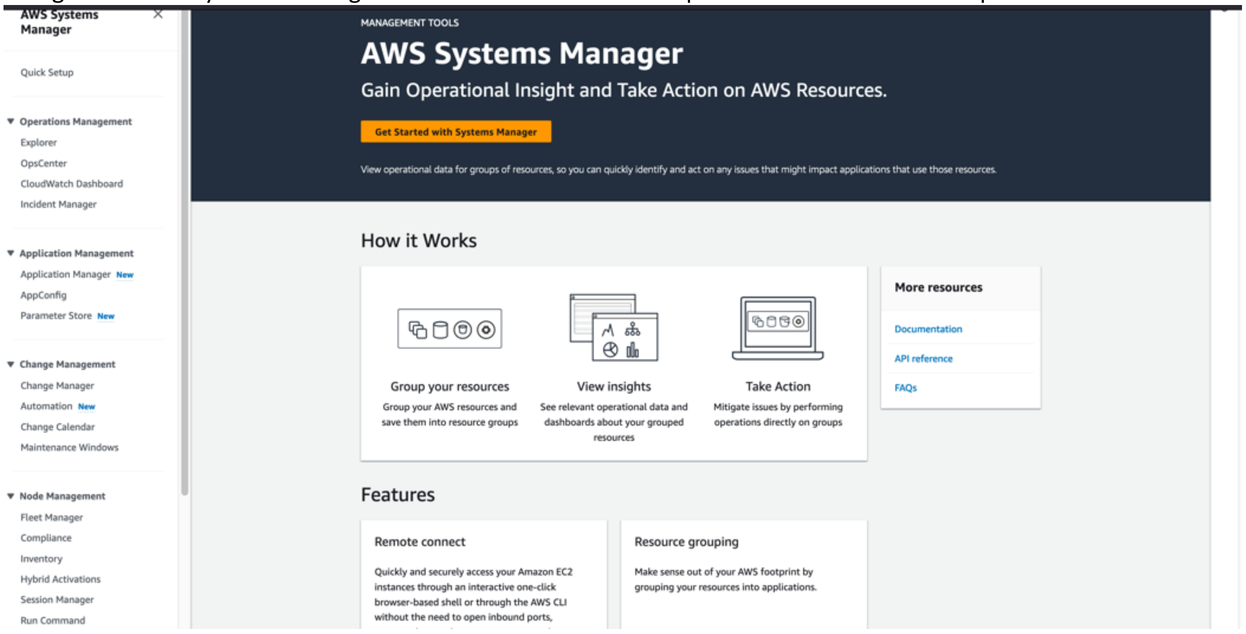

EC2 instances CloudWatch agents
2
CloudWatch Agents
4
Event bridge
Event bridge triggers lambda every 24 hrs
5
Lambda function
Lambda reads last 24 hour data from cloudwatch metrics and create csv file with the data
CloudWatch Metrics
Lambda calls Amazon SES service to send mail
1
Amazon EC2
6
Amazon SES
1
3
Amazon EC2
Send Email

The purpose of this code is to generate a report containing monitoring data from of all the EC2 instances, including CPU utilization, memory usage, and disk usage from Amazon CloudWatch agents. This includes the process where an AWS Lambda function is triggered daily by an Amazon EventBridge rule to gather monitoring data of all the EC2 instances collected by cloudwatch agents which is sent to Amazon CloudWatch. This data is then compiled into a CSV file, stored in an S3 bucket, and sent as an email attachment using Amazon SES, effectively automating the daily monitoring and reporting process for all EC2 instances.

Step 1: Check if the EBS is mounted to EC2 instance and showing while running the below command in EC2 terminal.

- connect to the EC2 instance.
- Run the below command to check the free spaces.
    - df -H
- If the EBS volumes are not shown while running the above command, then follow the below remaining steps to list all the EBS devices and mount it to the EC2 instance one by one.
- Run the following command in Ec2 instance terminal to list the available EBS devices
    - lsblk
- Lets first create a directory to be used as the mount point and mount the EBS to Ec2 instance:
    - sudo mkdir -p /mnt/ebs_volume1
    - sudo mkfs -t ext4 /dev/sdb
    - sudo file -s /dev/sdb
    - sudo mount /dev/sdb /mnt/ebs_volume1

Step 2: For this demo configuration, We use "AWS System Manager" service parameter store feature to store our Cloudwatch agent config file.

- We do this to make the cloudwatch agent config file centralized in one place. In your production setup you may have already installed and configured Cloudwatch agents in each of the ec2 instances. In that scenario, we need to update the cloudwatch agent config file in each ec2 instances with the below configuration. Here we are collecting the memory metrics and disk metrics from EC2 and sends to CWAgent metric in Cloud Watch. https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/metrics-collected-by-CloudWatch-agent.html
  - { "metrics": { "append_dimensions": { "InstanceId": "${aws:InstanceId}" }, "metrics_collected": { "mem": { "measurement": ["mem_used_percent"], "metrics_collection_interval": 60 }, "disk": { "measurement": ["disk_used", "disk_total", "disk_free"], "metrics_collection_interval": 60 } } } }
- Navigate to AWS Systems Manager from aws console and click parameter store in the left pane.
- 

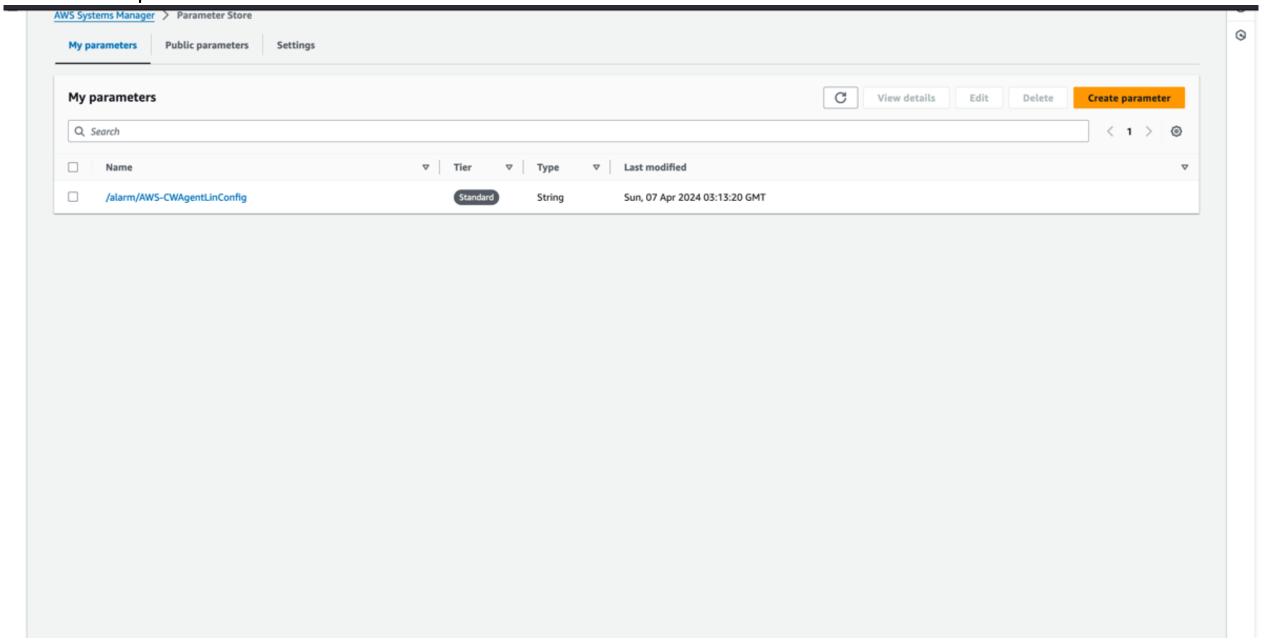
View operational data for groups of resources, so you can quickly identify and act on any issues that might impact applications that use those resources.

Group your AWS resources and save them into resource groups

See relevant operational data and dashboards about your grouped resources

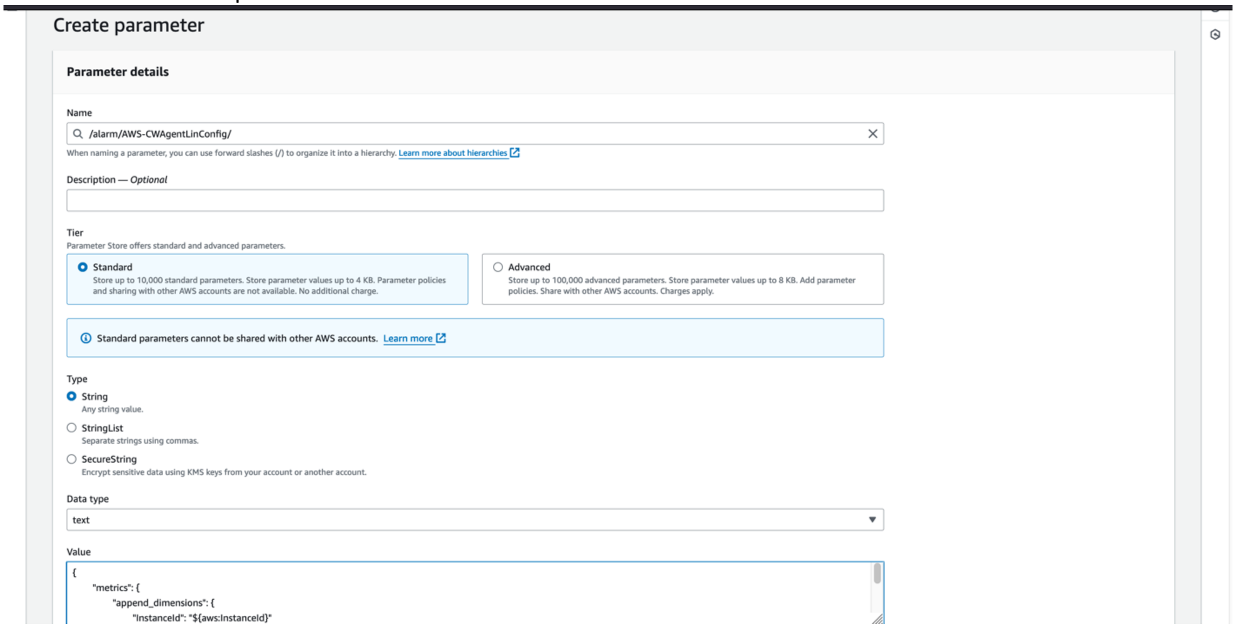
Mitigate issues by performing operations directly on groups

Quickly and securely access your Amazon EC2 instances through an interactive one-click browser-based shell or through the AWS CLI without the need to open inbound ports,

Make sense out of your AWS footprint by grouping your resources into applications.
- Click Create parameter button.
- 

My parameters

Name	Tier	Type	Last modified
/alarm/AWS-CWAgentLinConfig	Standard	String	Sun, 07 Apr 2024 03:13:20 GMT

- Enter the parameter name and paste the configuration which we mentioned in the Step 2 in the Value field and click Create parameter button.
- 

Create parameter

Parameter details

Name
/alarm/AWS-CWAgentLinConfig/
When naming a parameter, you can use forward slashes (/) to organize it into a hierarchy. Learn more about hierarchies

Description — *Optional*

Tier
Parameter Store offers standard and advanced parameters.
Standard
Store up to 10,000 standard parameters. Store parameter values up to 4 KB. Parameter policies and sharing with other AWS accounts are not available. No additional charge.
Advanced
Store up to 100,000 advanced parameters. Store parameter values up to 8 KB. Add parameter policies. Share with other AWS accounts. Charges apply.

Standard parameters cannot be shared with other AWS accounts. Learn more

Type
String
Any string value.
StringList
Separate strings using commas.
SecureString
Encrypt sensitive data using KMS keys from your account or another account.

Data type
text

Value
```
{
    "metrics": {
        "append_dimensions": {
            "InstanceId": "${aws:InstanceId}"
```

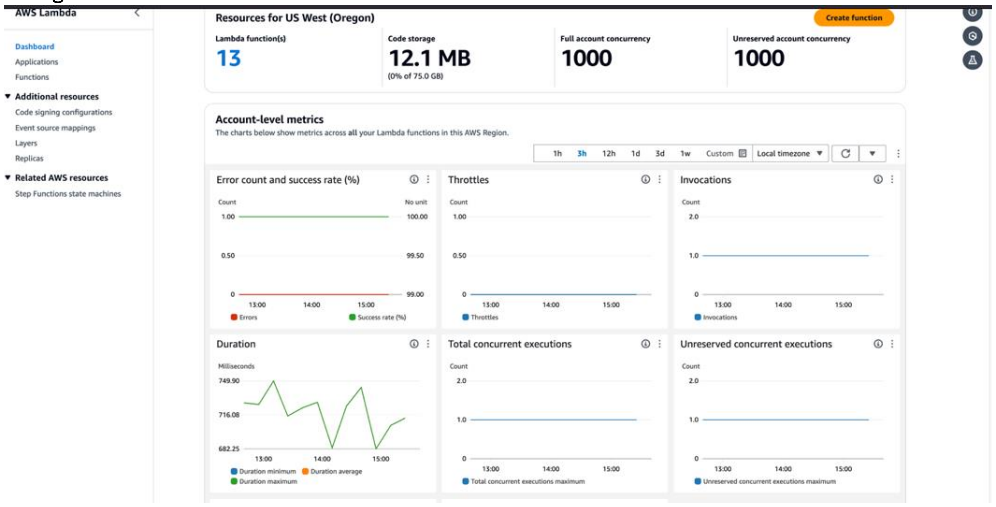
Step 3: This step is to download and install Cloudwatch agent in the EC2 machine and update the config file in EC2 from AWS systems Manager. Skip this step if you have already installed Cloudwatch agent in Ec2 and updated Cloudwatch agent config file.

- wget https://s3.amazonaws.com/amazoncloudwatch-agent/linux/amd64/latest/AmazonCloudWatchAgent.zip
- unzip AmazonCloudWatchAgent.zip
- sudo ./install.sh
- sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -c ssm:/alarm/AWS-CWAgentLinConfig -s

Step 4: Create and upload the configuration json file to S3. This file will be used by AWS Lambda. This file contains the instance name, id and the metric metadata details which the AWS lambda has to use to get the required details from the Cloudwatch Agent Metrics stored in Cloudwatch.

- This Json File "CPU_mem_disk_conf_json.json" will be uploaded to a S3 bucket in the same region where lambda function runs.

Step 5: Create a AWS lambda function with the attached sample code Automated Daily EC2 Instance Monitoring and Reporting.py

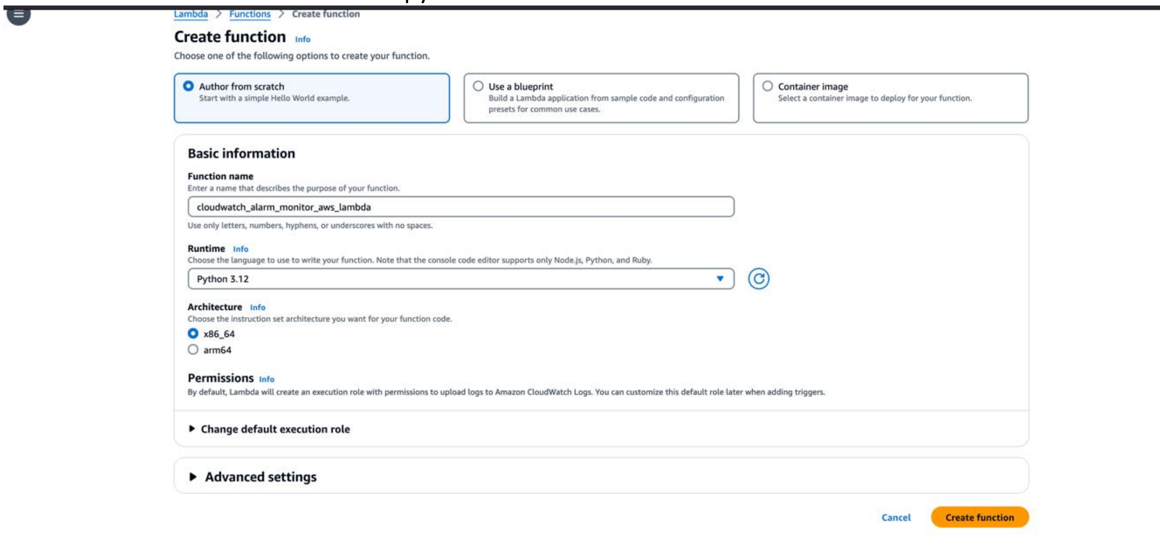
- Navigate to AWS Lambda dashboard and click create function button.
- 

Resources for US West (Oregon)

Lambda function(s)	Code storage	Full account concurrency	Unreserved account concurrency
13	12.1 MB (0% of 75.0 GB)	1000	1000

Account-level metrics
The charts below show metrics across all your Lambda functions in this AWS Region.

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



Lambda > Functions > Create function

Create function Info

Choose one of the following options to create your function.

[✓ Author from scratch — Start with a simple Hello World example.] Use a blueprint — Build a Lambda application from sample code and configuration presets for common use cases. Container image — Select a container image to deploy for your function.

Basic information

Function name: Enter a name that describes the purpose of your function.
cloudwatch_alarm_monitor_aws_lambda
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info: Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.12

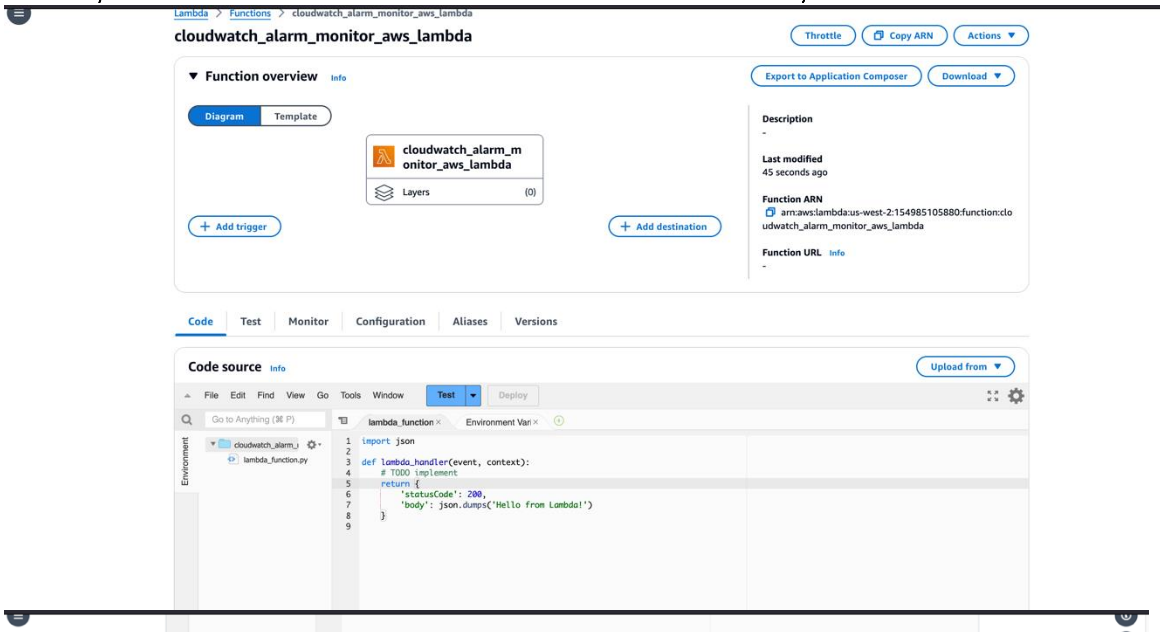
Architecture Info: Choose the instruction set architecture you want for your function code.
[✓ x86_64] arm64

Permissions Info: By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▸ Change default execution role

▸ Advanced settings

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



Lambda > Functions > cloudwatch_alarm_monitor_aws_lambda

cloudwatch_alarm_monitor_aws_lambda

Description: -

Last modified: 45 seconds ago

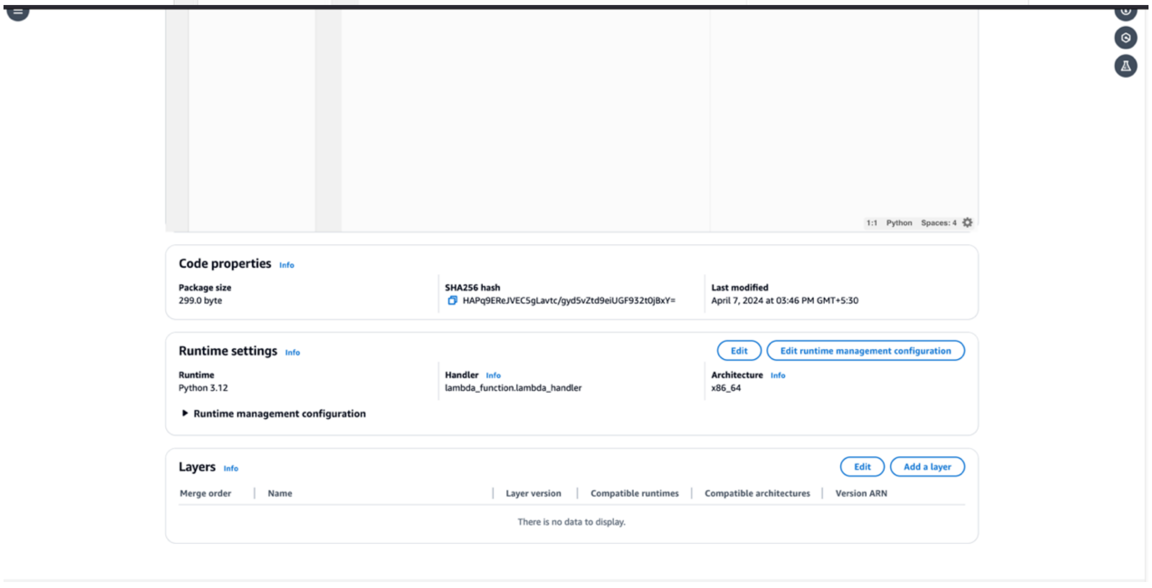
Function ARN: arn:aws:lambda:us-west-2:154985105880:function:cloudwatch_alarm_monitor_aws_lambda

Function URL Info: -

```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
```

-



Code properties Info

Package size	SHA256 hash	Last modified
299.0 byte	HAPq9EReJVEC5gLavtc/gyd5vZtd9eiUGF932t0jBxY=	April 7, 2024 at 03:46 PM GMT+5:30

Runtime settings Info

Runtime	Handler Info	Architecture Info
Python 3.12	lambda_function.lambda_handler	x86_64

▸ Runtime management configuration

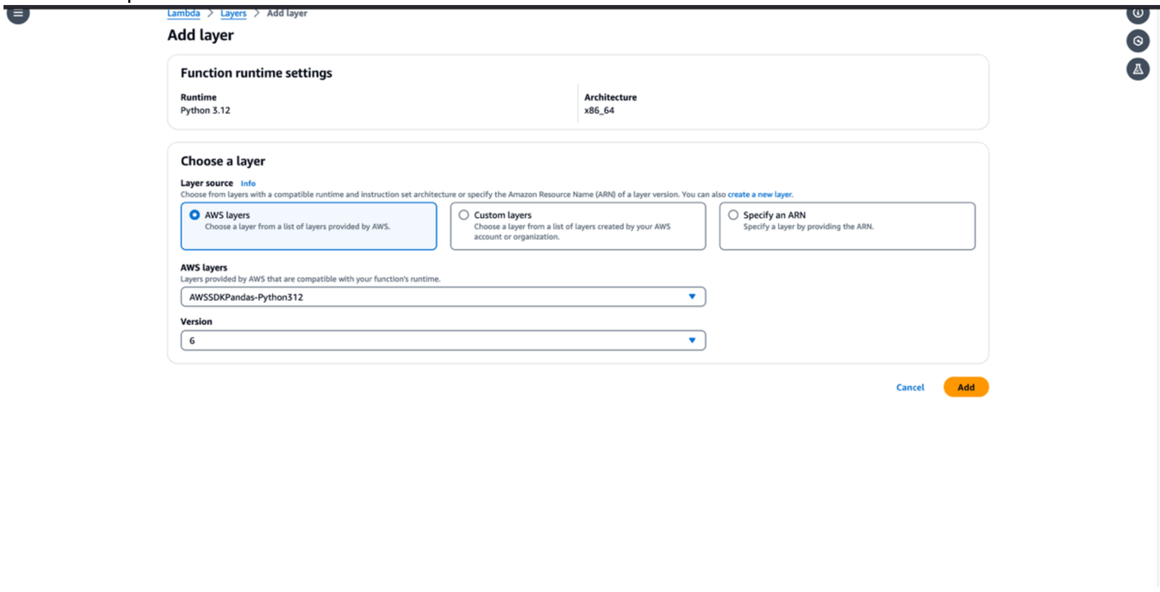
Layers Info

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.

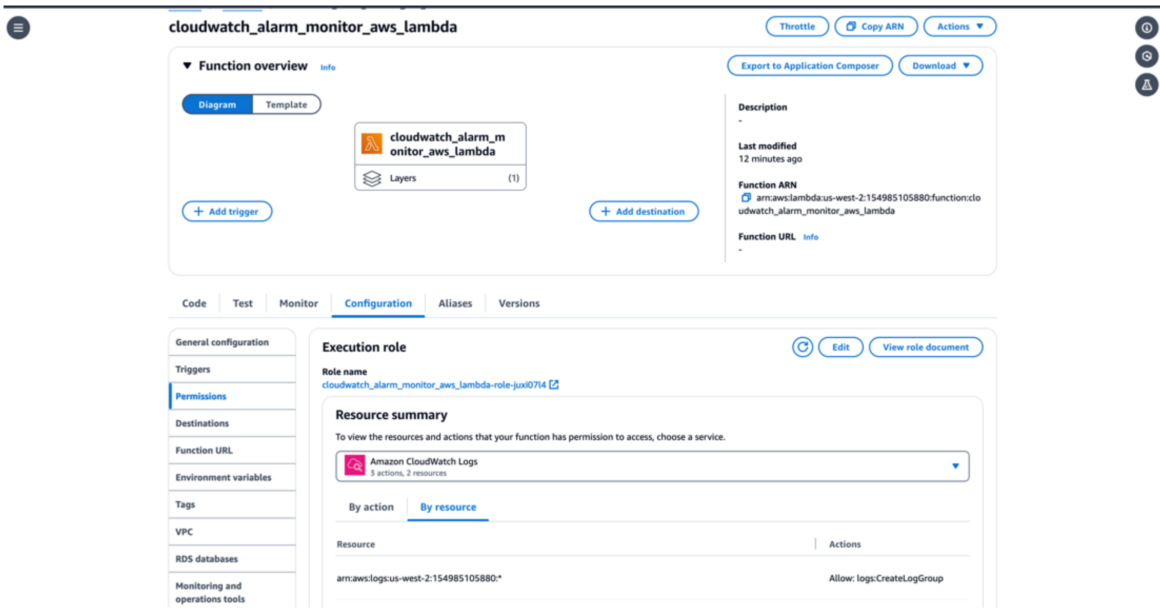
AWSSDKPandas-Python312

Version

6

-

- 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

Description: -

Last modified: 12 minutes ago

Function ARN: arn:aws:lambda:us-west-2:154985105880:function:cloudwatch_alarm_monitor_aws_lambda

Function URL Info: -

Execution role

Role name: cloudwatch_alarm_monitor_aws_lambda-role-juxi07l4

Resource summary

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

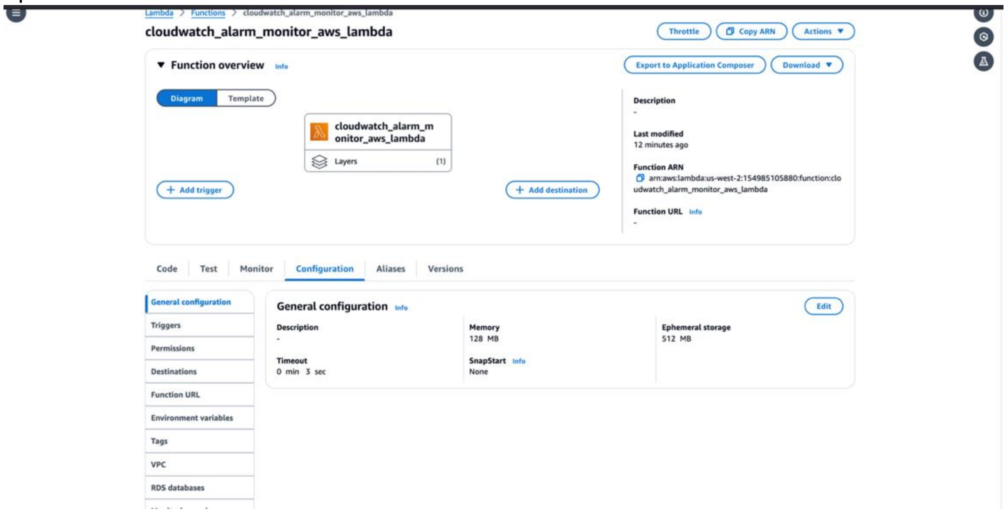
Amazon CloudWatch Logs
3 actions, 2 resources

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.



Lambda > Functions > cloudwatch_alarm_monitor_aws_lambda

cloudwatch_alarm_monitor_aws_lambda

Function overview

Description
-

Last modified
12 minutes ago

Function ARN
arn:aws:lambda:us-west-2:154985105880:function:cloudwatch_alarm_monitor_aws_lambda

Function URL
-

General configuration

Description
-

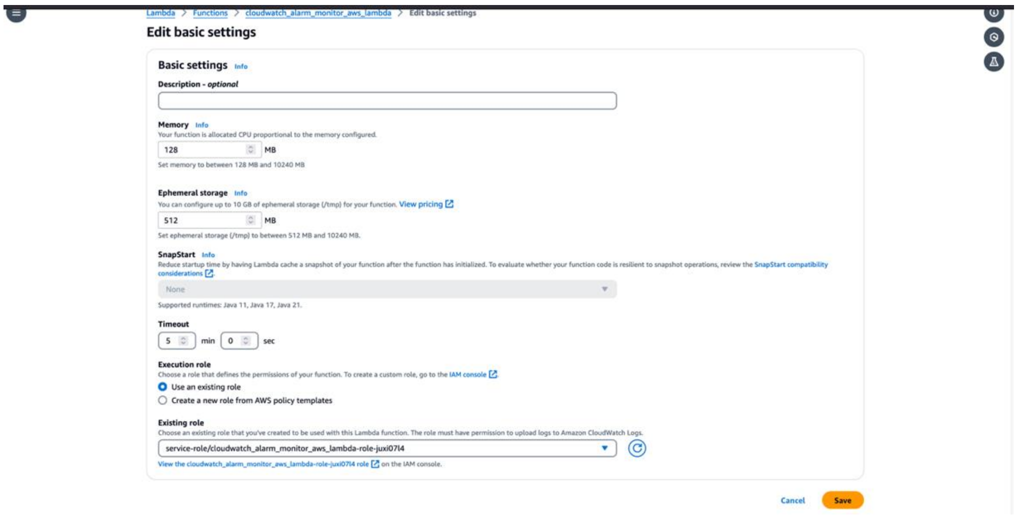
Memory
128 MB

Ephemeral storage
512 MB

Timeout
0 min 3 sec

SnapStart
None

-



Lambda > Functions > cloudwatch_alarm_monitor_aws_lambda > Edit basic settings

Edit basic settings

Basic settings

Description - *optional*

Memory
Your function is allocated CPU proportional to the memory configured.
128 MB
Set memory to between 128 MB and 10240 MB

Ephemeral storage
You can configure up to 10 GB of ephemeral storage (/tmp) for your function. View pricing
512 MB
Set ephemeral storage (/tmp) to between 512 MB and 10240 MB.

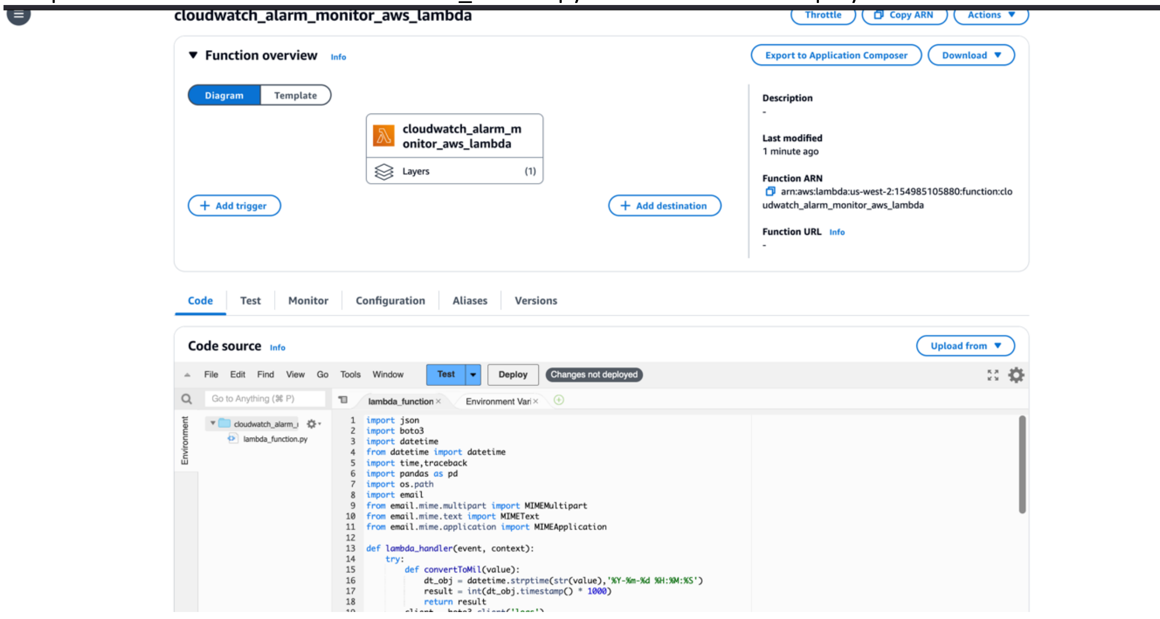
SnapStart
Reduce startup time by having Lambda cache a snapshot of your function after the function has initialized. To evaluate whether your function code is resilient to snapshot operations, review the SnapStart compatibility considerations.
None
Supported runtimes: Java 11, Java 17, Java 21.

Timeout
5 min 0 sec

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
[✓ Use an existing role]
Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
service-role/cloudwatch_alarm_monitor_aws_lambda-role-juxi07l4
View the cloudwatch_alarm_monitor_aws_lambda-role-juxi07l4 role on the IAM console.

-
- Copy the sample code from "Cloudwatch_metric_data_ec2instances_csv_lambda.py" file and paste it in "code" section into lambda_funtion.py file and then click Deploy button.



cloudwatch_alarm_monitor_aws_lambda

Function overview

Description
-

Last modified
1 minute ago

Function ARN
arn:aws:lambda:us-west-2:154985105880:function:cloudwatch_alarm_monitor_aws_lambda

Function URL
-

Code source

Changes not deployed

lambda_function.py

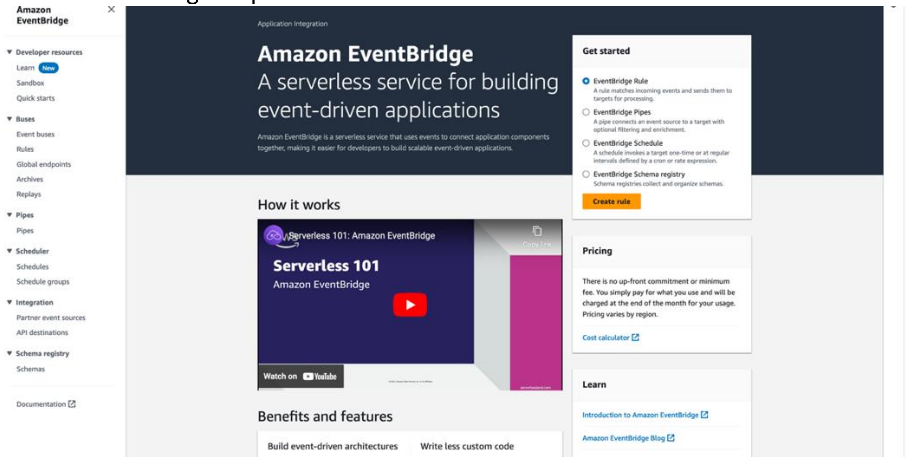
```
import json
import boto3
import datetime
from datetime import datetime
import time,traceback
import pandas as pd
import os.path
import email
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.application import MIMEApplication

def lambda_handler(event, context):
    try:
        def convertToMil(value):
            dt_obj = datetime.strptime(str(value),'%Y-%m-%d %H:%M:%S')
            result = int(dt_obj.timestamp() * 1000)
            return result
```

-

Step 6: Create a Amazon Event Bridge rule to trigger the AWS Lambda function created in step:5, 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
- 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.



Amazon EventBridge

▼ Developer resources
Learn New
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

Application Integration

Amazon EventBridge
A serverless service for building event-driven applications

Amazon EventBridge is a serverless service that uses events to connect application components together, making it easier for developers to build scalable event-driven applications.

Get started

- [✓ EventBridge Rule] A rule matches incoming events and sends them to targets for processing.
- EventBridge Pipes A pipe connects an event source to a target with optional filtering and enrichment.
- EventBridge Schedule A schedule invokes a target one-time or at regular intervals defined by a cron or rate expression.
- EventBridge Schema registry Schema registries collect and organize schemas.

Create rule

How it works

Pricing

There is no up-front commitment or minimum fee. You simply pay for what you use and will be charged at the end of the month for your usage. Pricing varies by region.

Cost calculator

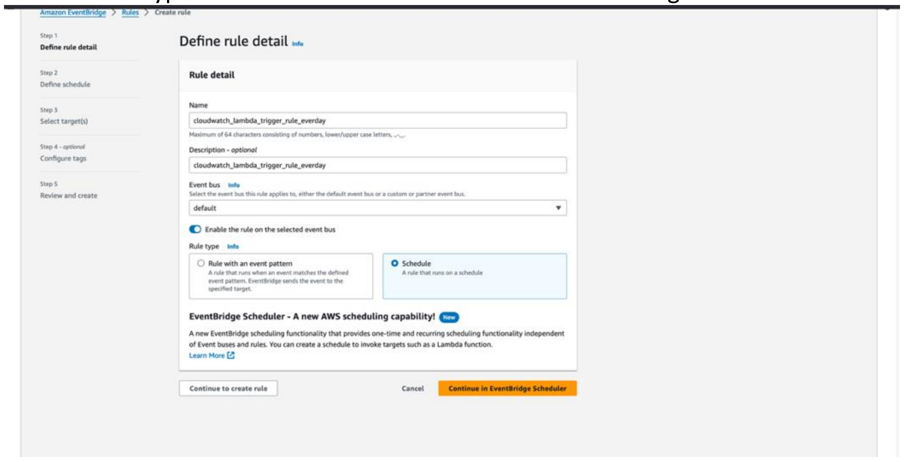
Learn

Introduction to Amazon EventBridge
Amazon EventBridge Blog

Benefits and features

Build event-driven architectures
Write less custom code

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



Amazon EventBridge > Rules > Create rule

Step 1
Define rule detail
Step 2
Define schedule
Step 3
Select target(s)
Step 4 - optional
Configure tags
Step 5
Review and create

Define rule detail Info

Rule detail

Name
cloudwatch_lambda_trigger_rule_everday
Maximum of 64 characters consisting of numbers, lower/upper case letters, .,-,_.

Description - *optional*
cloudwatch_lambda_trigger_rule_everday

Event bus Info
Select the event bus this rule applies to, either the default event bus or a custom or partner event bus.
default

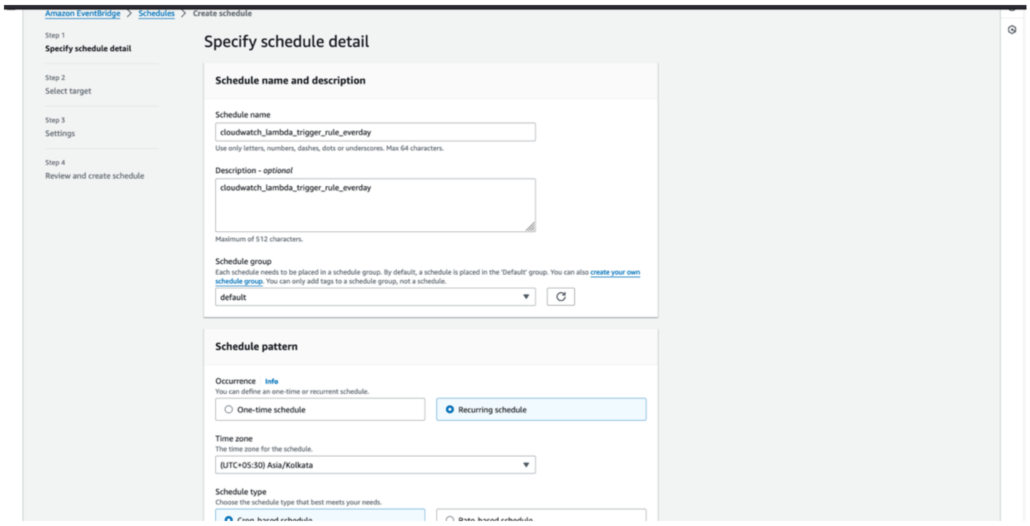
Enable the rule on the selected event bus

Rule type Info
- Rule with an event pattern A rule that runs when an event matches the defined event pattern. EventBridge sends the event to the specified target.
- [✓ Schedule] A rule that runs on a schedule

EventBridge Scheduler - A new AWS scheduling capability! New
A new EventBridge scheduling functionality that provides one-time and recurring scheduling functionality independent of Event buses and rules. You can create a schedule to invoke targets such as a Lambda function.
Learn More

Continue to create rule | Cancel | Continue in EventBridge Scheduler

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



Amazon EventBridge > Schedules > Create schedule

Step 1
Specify schedule detail
Step 2
Select target
Step 3
Settings
Step 4
Review and create schedule

Specify schedule detail

Schedule name and description

Schedule name
cloudwatch_lambda_trigger_rule_everday
Use only letters, numbers, dashes, dots or underscores. Max 64 characters.

Description - *optional*
cloudwatch_lambda_trigger_rule_everday
Maximum of 512 characters.

Schedule group
Each schedule needs to be placed in a schedule group. By default, a schedule is placed in the 'Default' group. You can also create your own schedule group. You can only add tags to a schedule group, not a schedule.
default

Schedule pattern

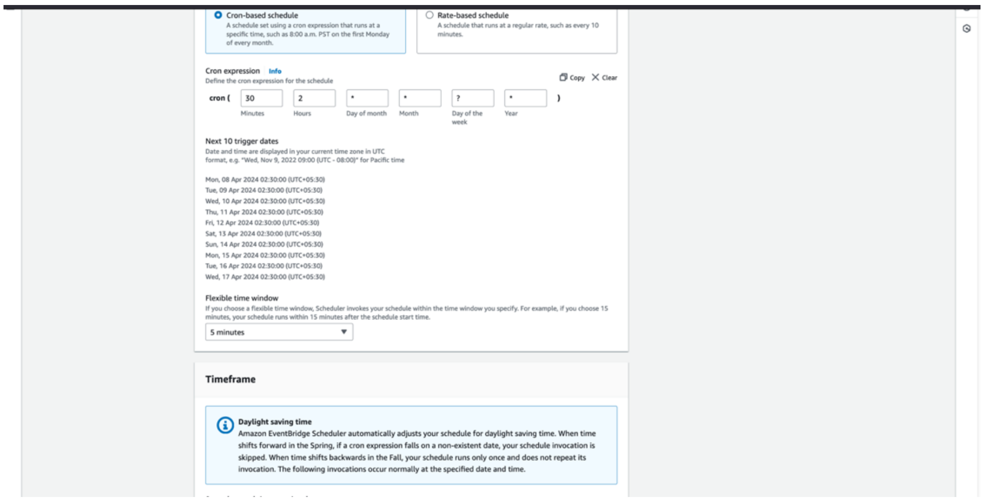
Occurrence Info
You can define an one-time or recurrent schedule.
One-time schedule [✓ Recurring schedule]

Time zone
The time zone for the schedule.
(UTC+05:30) Asia/Kolkata

Schedule type
Choose the schedule type that best meets your needs.

-

- 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 ( 30       2      *              *       ?                *     )
       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

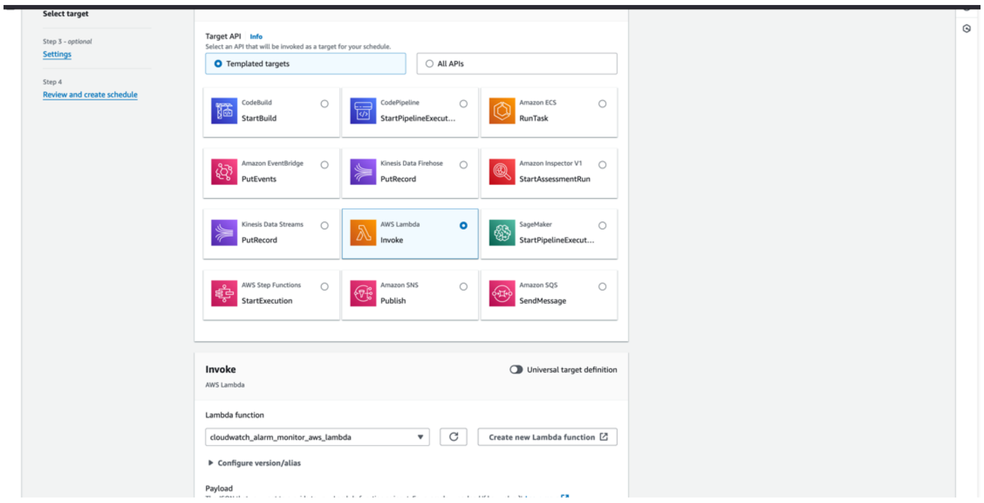
```
Mon, 08 Apr 2024 02:30:00 (UTC+05:30)
Tue, 09 Apr 2024 02:30:00 (UTC+05:30)
Wed, 10 Apr 2024 02:30:00 (UTC+05:30)
Thu, 11 Apr 2024 02:30:00 (UTC+05:30)
Fri, 12 Apr 2024 02:30:00 (UTC+05:30)
Sat, 13 Apr 2024 02:30:00 (UTC+05:30)
Sun, 14 Apr 2024 02:30:00 (UTC+05:30)
Mon, 15 Apr 2024 02:30:00 (UTC+05:30)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

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 5 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

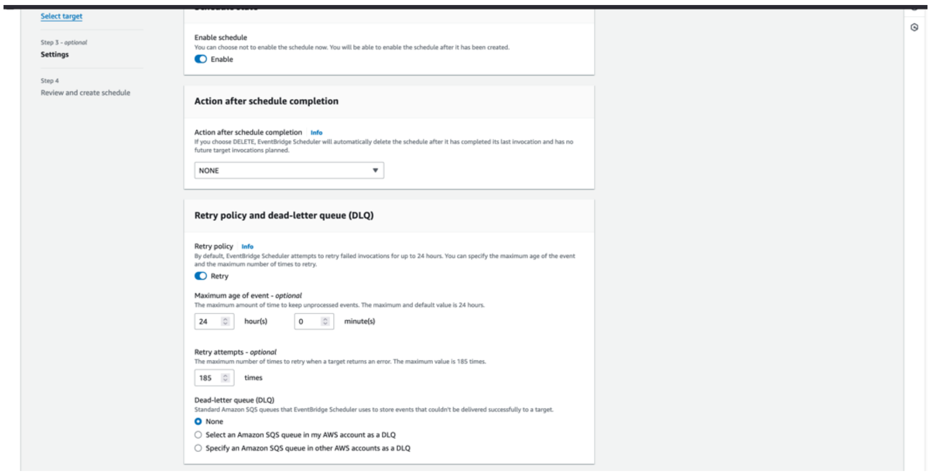
CodeBuild StartBuild	CodePipeline StartPipelineExecut...	Amazon ECS RunTask
Amazon EventBridge PutEvents	Kinesis Data Firehose PutRecord	Amazon Inspector V1 StartAssessmentRun
Kinesis Data Streams PutRecord	[✓ AWS Lambda Invoke]	SageMaker StartPipelineExecut...
AWS Step Functions StartExecution	Amazon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

Invoke
AWS Lambda

Lambda function
cloudwatch_alarm_monitor_aws_lambda

Payload

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



Select target
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

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.
NONE

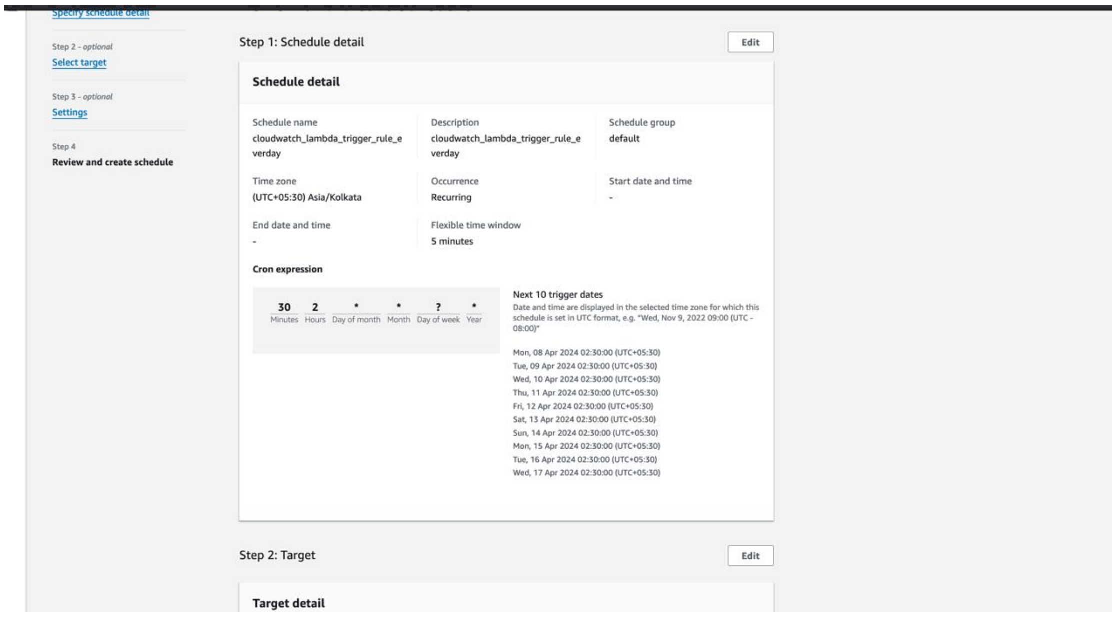
Retry policy and dead-letter queue (DLQ)

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.
24 hour(s) 0 minute(s)

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

- 

Specify schedule detail

Step 2 - *optional*
Select target

Step 3 - *optional*
Settings

Step 4
Review and create schedule

Step 1: Schedule detail

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

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

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

Target detail