

Usecase 2:

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.

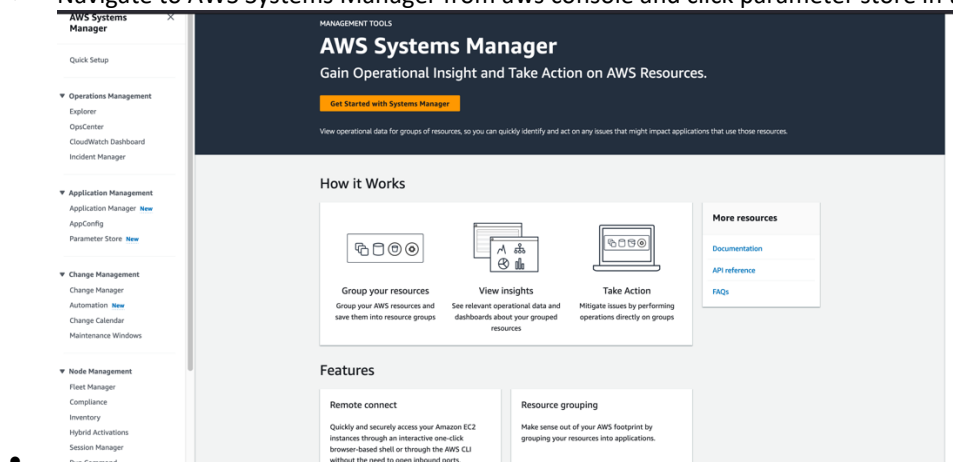
Memory usage , root disk, EBS disk details are from the CloudWatch agents configured in each of the EC2 instances.

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

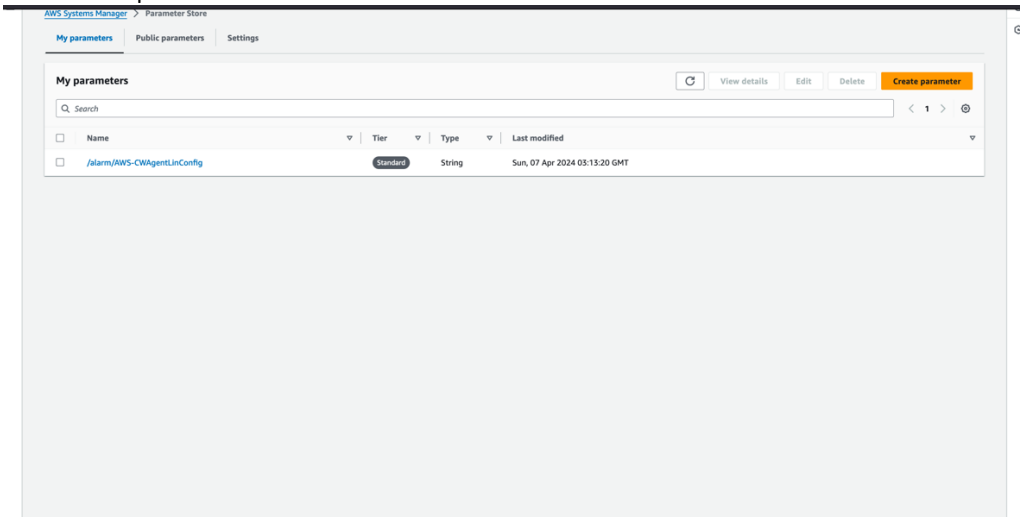
- connect to the EC2 instance.
- Run the below command to check the free spaces.
 - `df -H`
- If the EBS volumes are not shown while running the above command, then follow the below remaining steps to list all the EBS devices and mount it to the EC2 instance one by one.
- Run the following command in 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.

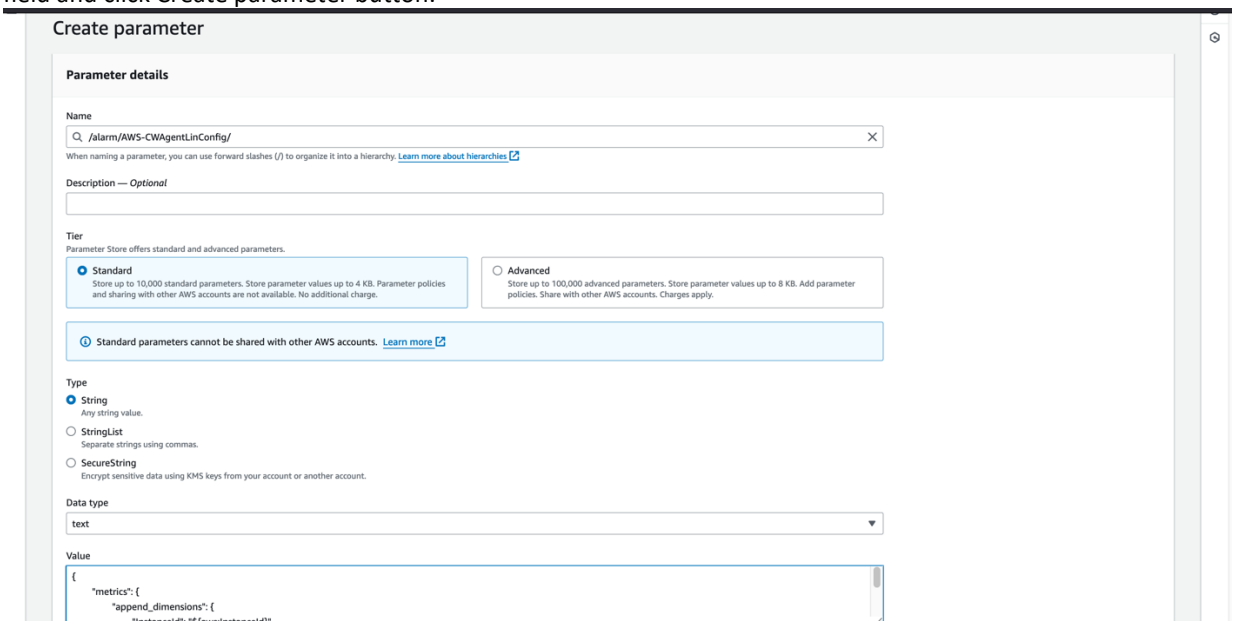
- We do this to make the cloudwatch agent config file centralized in one place. In your production setup you may have already installed and configured Cloudwatch agents in each of the ec2 instances. In that scenario, we need to update the cloudwatch agent config file in each ec2 instances with the below 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.



- Click Create parameter button.



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



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.

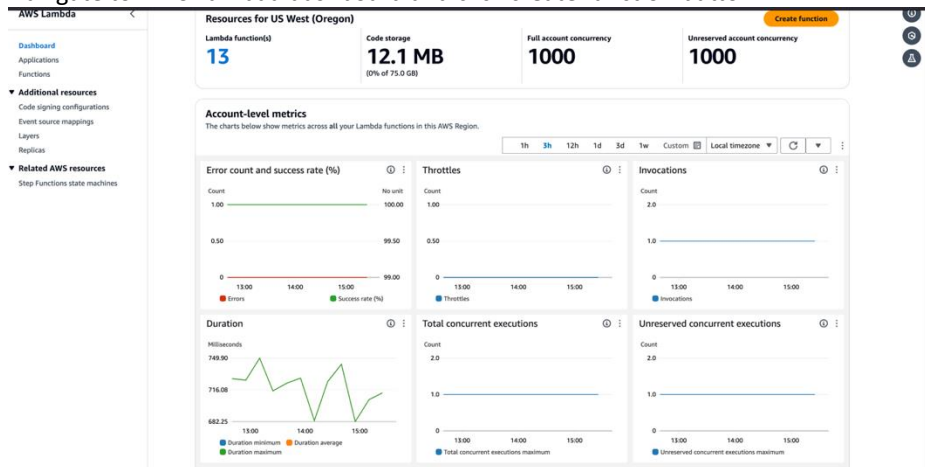
- wget <https://s3.amazonaws.com/amazoncloudwatch-agent/linux/amd64/latest/AmazonCloudWatchAgent.zip>
- unzip AmazonCloudWatchAgent.zip
- sudo ./install.sh
- sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -c ssm:/alarm/AWS-CWAgentLinConfig -s

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 Cloudwatch_metric_data_ec2instances_csv_lambda.py

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



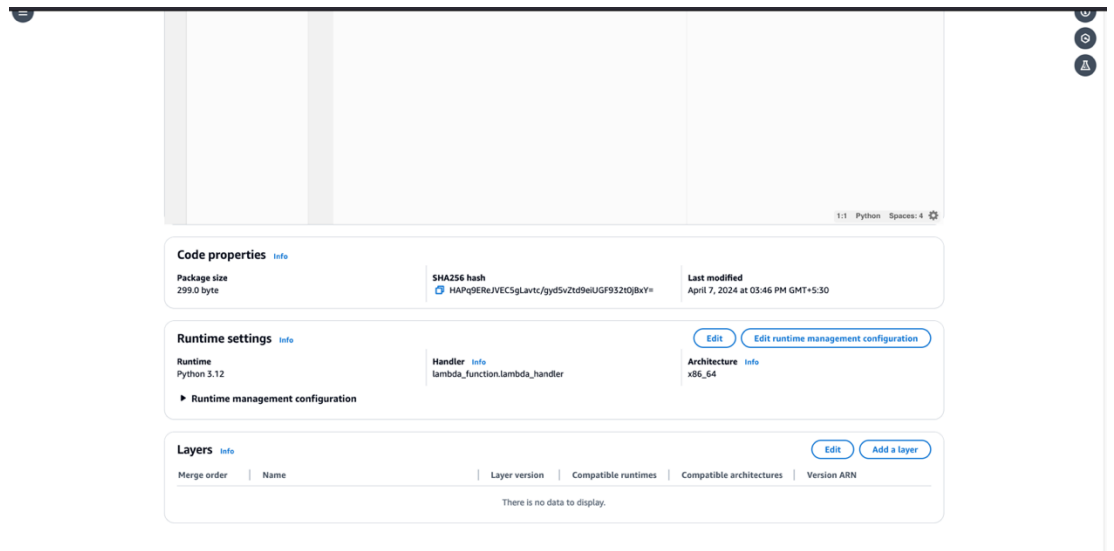
- Enter the function name and choose python as runtime and click create function button.

The screenshot shows the 'Create function' page in the AWS Lambda console. The 'Basic information' section is expanded, showing 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.

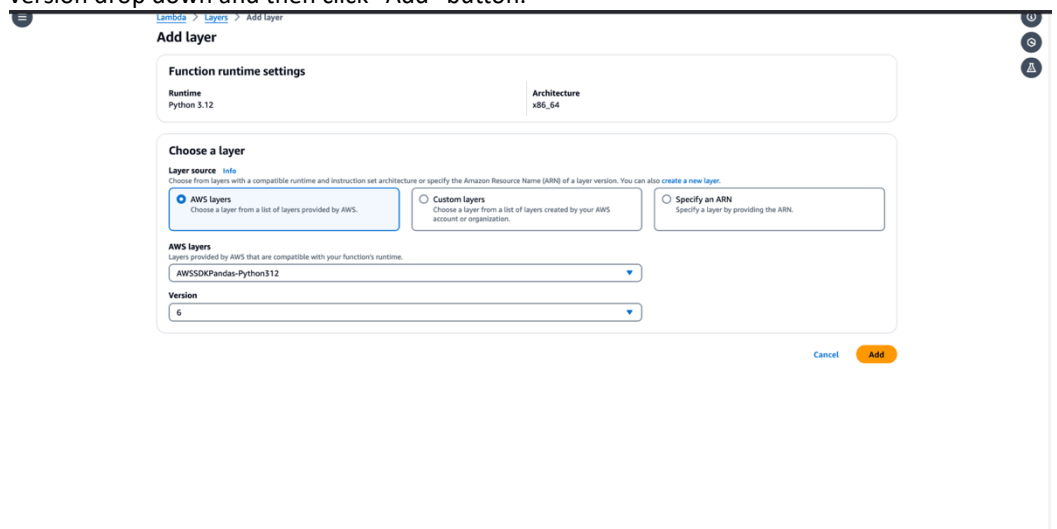
- Click “Layers” from the function overview section and then click “Add a layer” button.

The screenshot shows the 'Function overview' page for the function 'cloudwatch_alarm_monitor_aws_lambda'. The 'Layers' section is expanded, showing the 'Add a layer' button. The 'Code source' section shows the function code:

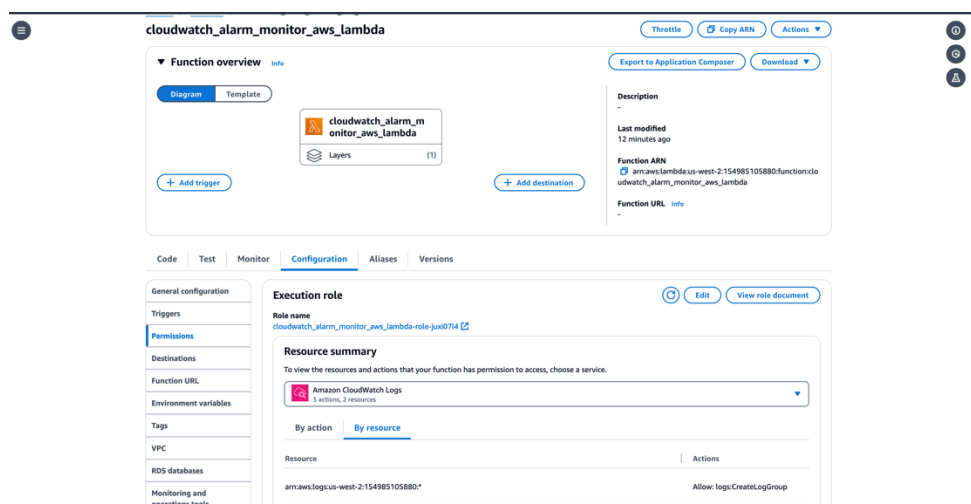
```
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     return {
6         'statusCode': 200,
7         'body': json.dumps('Hello from Lambda!')}
8
9
```



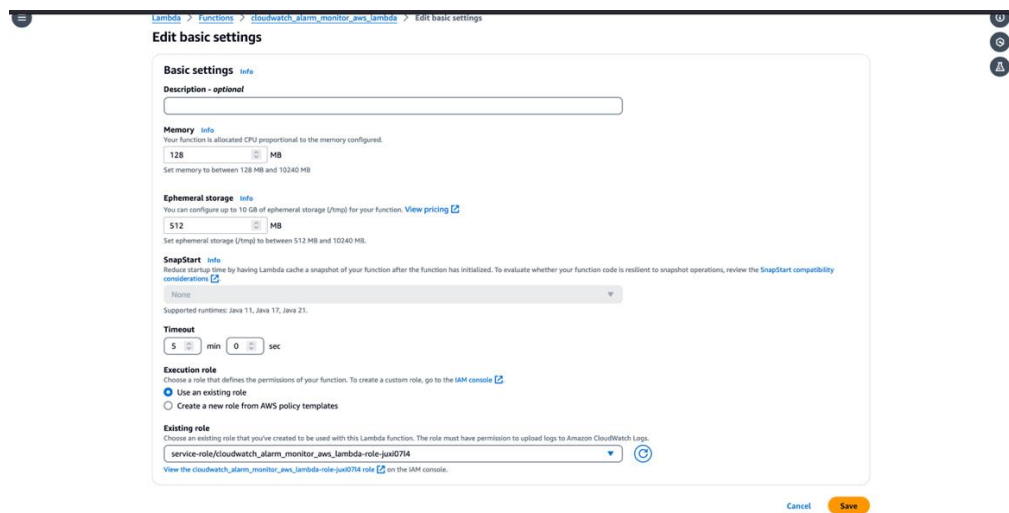
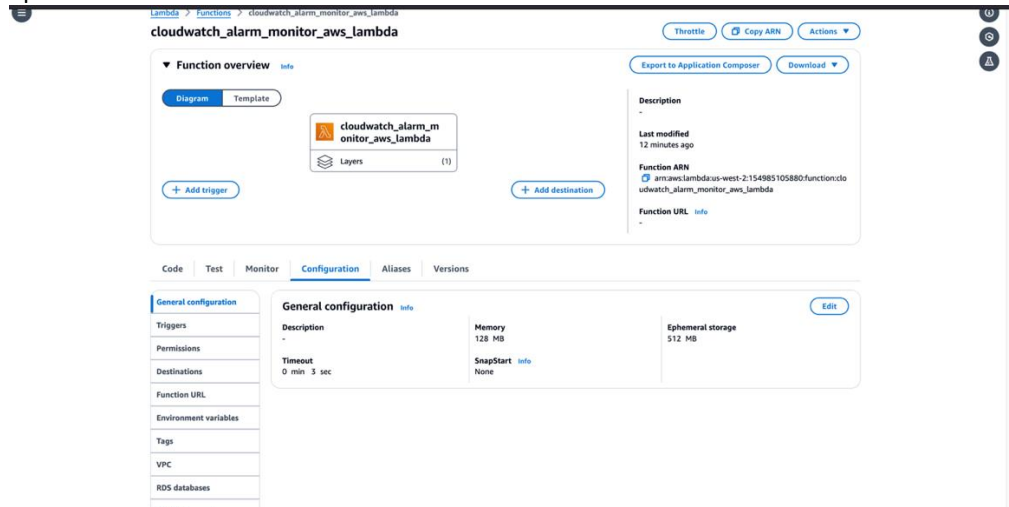
-
- Choose “AWSSDKPandas-Python312” as AWS layers and choose latest version from version drop down and then click “Add” button.



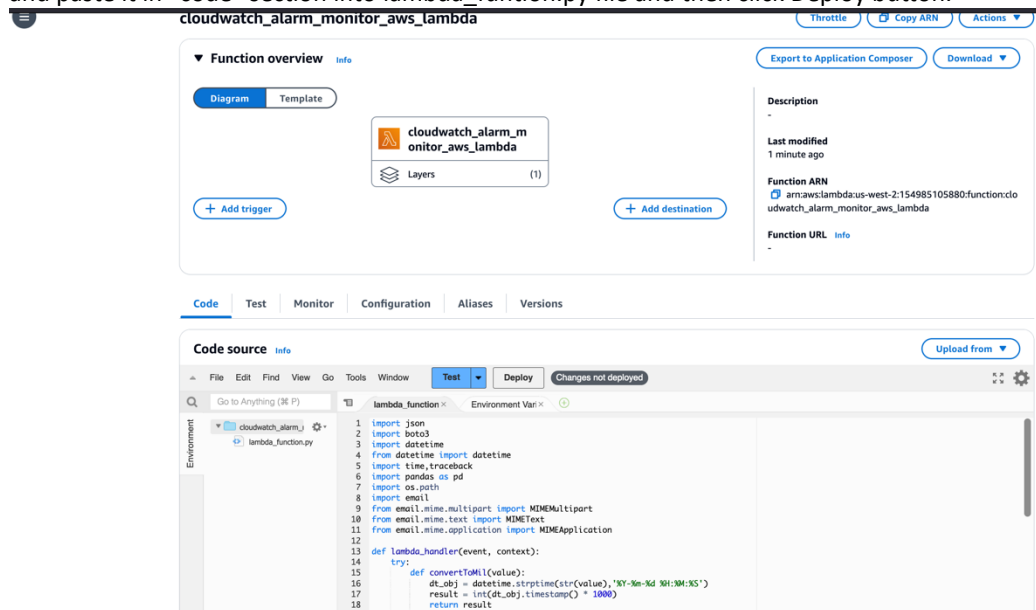
-
- Click “configuration” tab and then click “permissions” tab. Click on the Role name and it will take us to IAM and then attach policies for this Lambda function to have access to Amazon Cloudwatch, S3 bucket and Amazon SES service.



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

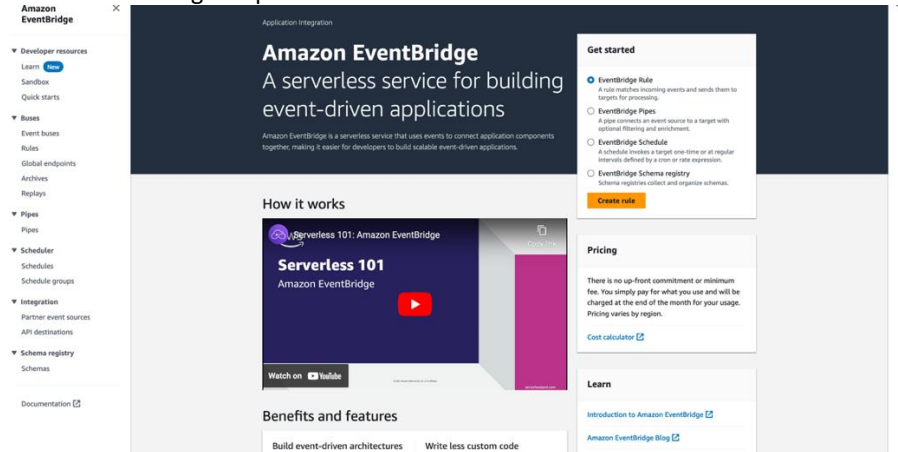


- Copy the sample code from “Cloudwatch_metric_data_ec2instances_csv_lambda.py” file and paste it in “code” section into lambda_function.py file and then click Deploy button.

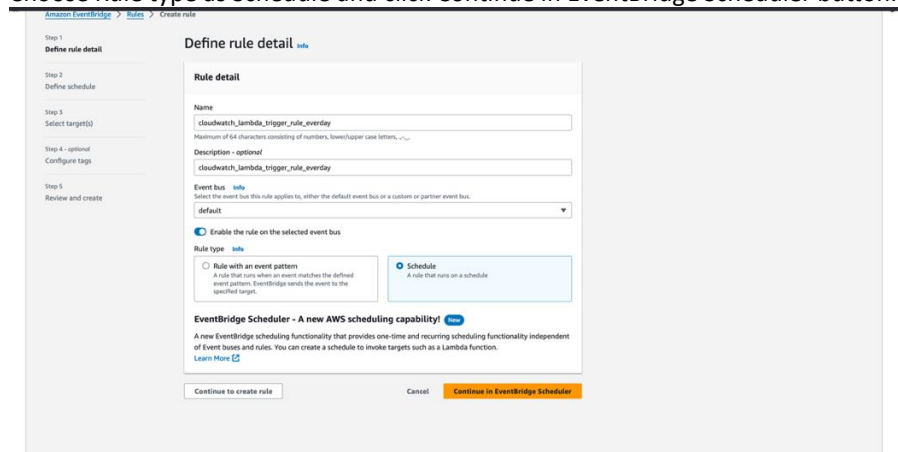


Step 6: Create a Amazon Event Bridge rule to trigger the AWS Lambda function created in step:5, everyday at a specific time of a day.

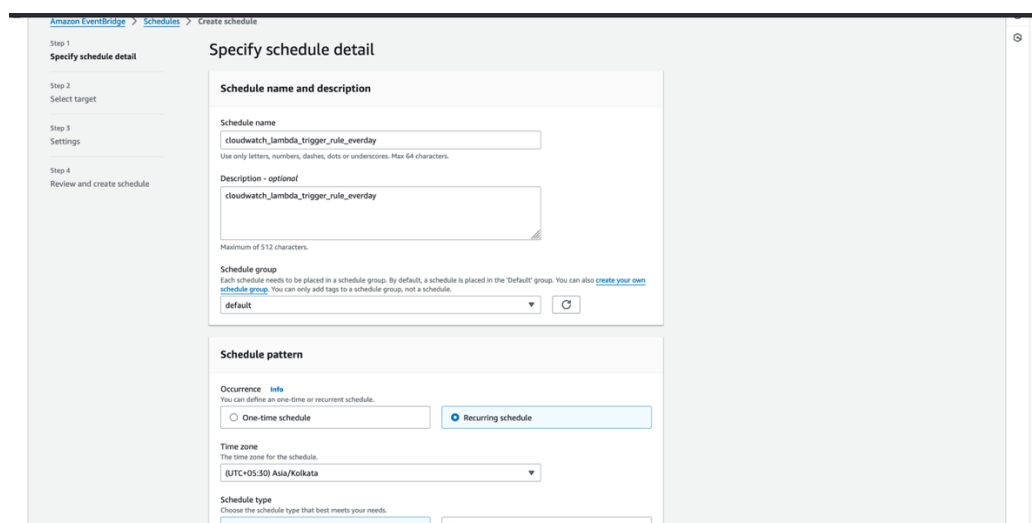
- https://docs.aws.amazon.com/scheduler/latest/UserGuide/schedule-types.html?icmpid=docs_console_unmapped#cron-based
- Navigate to Amazon Event bridge console and click **'Create rule'** button in the Amazon Event bridge home page or from the left panel click "Rules" and then Click **"Create rule"**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



- 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.

Cron expression info
Define the cron expression for the schedule
cron (30 2 * * ? *) Copy Clear
Minutes Hours Day of month Month Day of the week Year

Next 10 trigger dates
Data and time are displayed in your current time zone in UTC.
Format: e.g. "Mon, 08 Apr 2024 02:30:00 UTC+05:30"
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

StartBuild	StartPipelineExecut...	RunTask
PutEvents	PutRecord	StartAssessmentRun
PutRecord	Invoke	StartPipelineExecut...
StartExecution	Publish	SendMessage

Invoke info
AWS Lambda

Lambda function
cloudwatch_alarm_monitor_aws_lambda ↻ Create new 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.
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 hours 0 minutes

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

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