



Deep Dive: Amazon CloudWatch – Logs and Events

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Deep Dive: Amazon CloudWatch – Logs and Events

What you will learn

At the core of the lesson

You will learn how to:

- Describe the features of Amazon CloudWatch Events
- Describe the features and benefits of Amazon CloudWatch Logs

Topics:

- Amazon CloudWatch Events
- Amazon CloudWatch Logs

Key terms:

- Amazon CloudWatch Events
- Amazon CloudWatch Logs

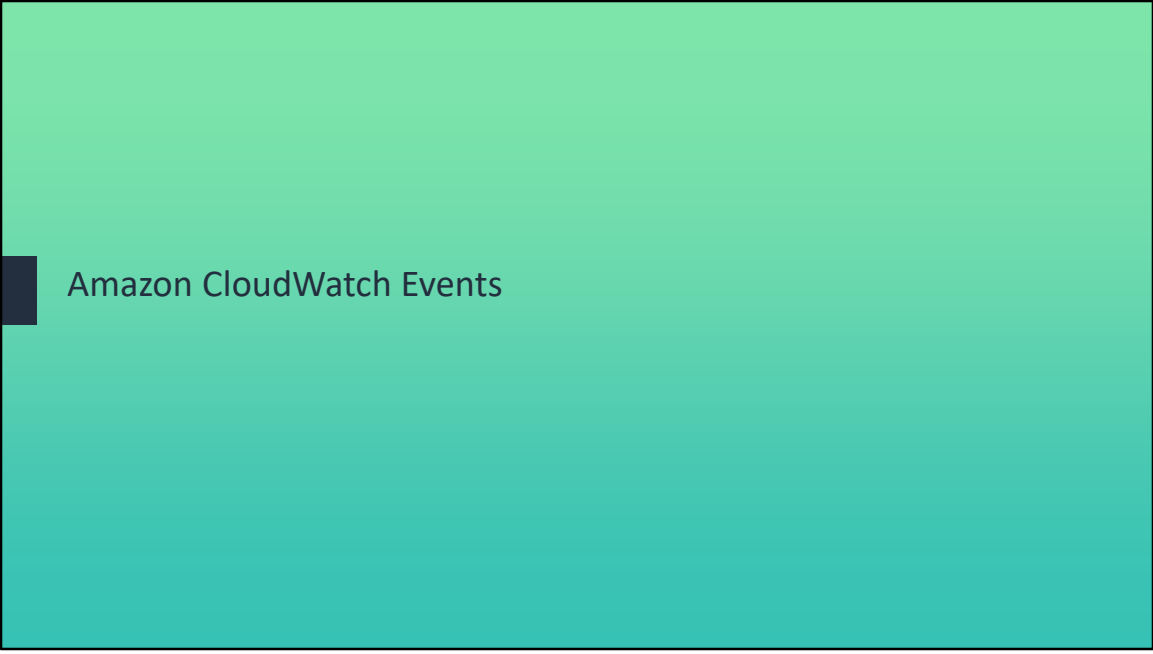
Project:

Troubleshooting Knowledge Base



At the end of this module, you will be able to:

- Describe the features of Amazon CloudWatch Events
- Describe the features and benefits of Amazon CloudWatch Logs



Amazon CloudWatch Events

Amazon CloudWatch Events

Step 1: Create rule

Create a CloudWatch Events rule

Create rules to invoke Targets based on Events happening in your AWS environment.

Event Source

Build or customize an Event Pattern

Event Pattern

Build event pattern to match

Service Name: Auto Scaling

Event Type: Launch and Terminate

Any instance event

Specific instance event(s)

EC2 Instance Launch Successful

Any group name

Specific group name(s)

Event Pattern Preview

```
{
  "source": [
    "aws.autoscaling"
  ]
}
```

Copy to clipboard Edit

Targets

Select Target to invoke when an event matches your Event Pattern or when schedule is triggered.

SSM Run Command

Document: AWS-RunRemoteScript (Windows, Linux)

Target key: <tag:Amazon EC2 tag>

Target value(s): <InstanceID>

A Run Command Target provides a way to specify which instances to invoke SSM Run Command on. Learn more

Configure parameter(s)

CloudWatch Events needs permission to call SSM Run Command on your EC2 Instance(s). By continuing, you are allowing us to do so.

Create a new role for this specific target

AWS_Events_Invoke_Run_Command

Use existing role

Learn more about CloudWatch Events Identity-Based Policies

Add target*

Each time Amazon EC2 Auto Scaling launches an Amazon Elastic Compute Cloud (Amazon EC2) instance

Run a specific AWS Systems Manager Run Command script on the instance

4



Amazon CloudWatch Events delivers a near-real-time stream of system events that describe changes in AWS resources. By using simple rules that you can configure, you can match events and route them to one or more target functions or streams. CloudWatch Events becomes aware of operational changes as they occur. It responds to these operational changes by sending messages, activating functions, making changes, and capturing state information.

You can use CloudWatch Events to schedule automated actions that self-trigger at certain times by using cron or rate expressions. For details about creating schedule expressions using Rate or Cron, see:

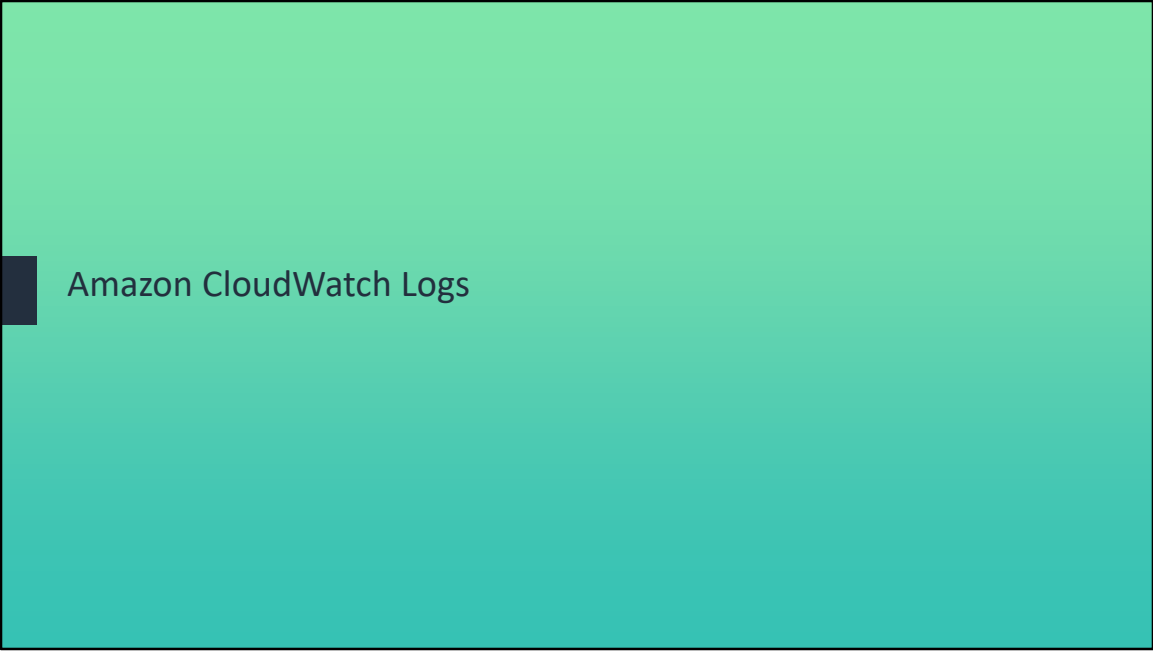
<https://docs.aws.amazon.com/lambda/latest/dg/tutorial-scheduled-events-schedule-expressions.html>.

Before you use CloudWatch Events, you should understand the following concepts:

- **Events** – An *event* indicates a change in your AWS environment. AWS resources can generate events when their state changes. For example, Amazon Elastic Compute Cloud (Amazon EC2) generates an event when the state of an EC2 instance changes from *pending* to *running*. You can generate custom application-level events and publish them to CloudWatch Events. You can also set up scheduled events that are generated on a periodic basis.

- **Targets** – A *target* processes events. Example targets include EC2 instances, AWS Lambda functions, Amazon Simple Notification Service (Amazon SNS) topics, and Amazon Simple Queue Service (Amazon SQS) queues.
- **Rules** – A *rule* matches incoming events and routes them to targets for processing. A single rule can route to multiple targets, all of which are processed in parallel. This enables different parts of an organization to look for and process the events that are of interest to them.

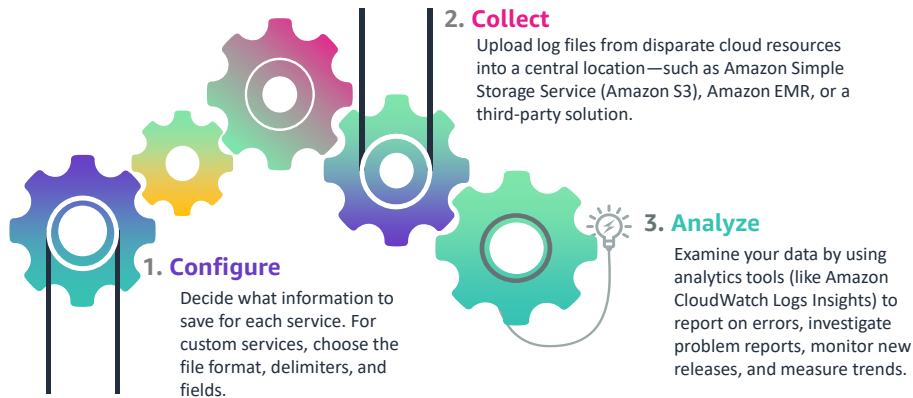
In the example, a CloudWatch Event *rule* is being created. Here, each time a new instance is created, an AWS Systems Manager Run Command script is run on the instance.



Amazon CloudWatch Logs

Amazon CloudWatch Logs

Typical log analysis process



6

aws re/start

You can use Amazon CloudWatch Logs to monitor, store, and access your log files from EC2 instances, AWS CloudTrail, Amazon Route 53, and other sources. You can then retrieve the associated log data from CloudWatch Logs. You can monitor your logs, in near-real time, for specific phrases, values, or patterns.

For example, you could set an alarm on the number of errors that occur in the system logs on one or more of your EC2 instances. You can also view graphs that visualize the latency of web requests from application logs on your EC2 instances. You can then view the original log data to see the source of the problem. Log data can be stored and accessed indefinitely—and stored externally to EC2 instances—so you do not need to worry about filling up hard drives.

You can think of the process of log analysis as having three distinct phases:

- 1. Configure** – Decide what information you need to capture in your logs, and where and how it will be stored.
- 2. Collect** – Instances are provisioned and removed in a cloud environment. You need a strategy for periodically uploading a server's log files so that this valuable

information is not lost when an instance is eventually terminated.

3. **Analyze** – After all the data is collected, it is time to analyze it. Using log data gives you greater visibility into the daily health of your systems. It can also provide information on upcoming trends in customer behavior, and insight into how customers currently use your system.

Amazon CloudWatch Logs functionality

CloudWatch Logs functionality includes:

- Automatically collecting logs—for example, from EC2 instances
- Aggregating data into *log groups*
- Being able to configure *metric filters* on a log group –
 - Look for specific string patterns
 - Have each match increment a custom CloudWatch metric
 - Use the metric to create CloudWatch alarms or send notifications
- Querying logs and creating visualizations with **CloudWatch Logs Insights**

CloudWatch Logs enables you to automatically collect logs from one of the supported services, such as from EC2 instances. You must install the new unified CloudWatch agent (or the older CloudWatch Logs agent) on any EC2 instance that you want to collect log data from.

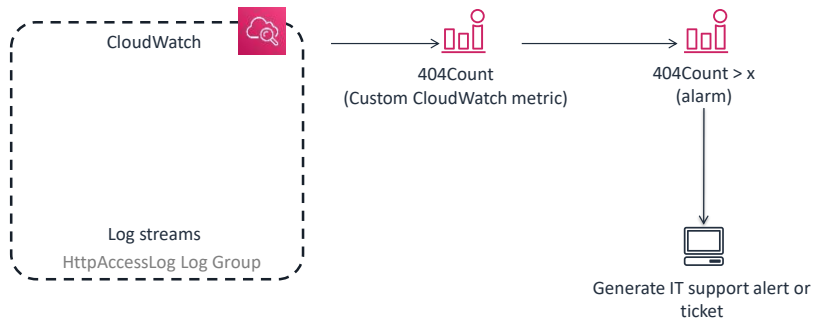
You can then aggregate data from several different EC2 instances into what are called *log groups*. Each log group should represent a specific type of log with a set format. The CloudWatch agent on each EC2 instance gathers data from the log that you specify (such as an application log) and sends it to the appropriate log group.

An administrator can then *create filters* on a log group to look for specific strings or string patterns. Each match is assigned a numeric value, which is then used to increment a custom CloudWatch *metric*. Administrators can then use that metric—like they would any other custom CloudWatch metric—to create CloudWatch alarms or send notifications.

CloudWatch Logs Insights is a part of the CloudWatch service that provides a purpose-built query language with a few simple, but powerful, commands. It provides sample queries, command descriptions, automatic query completion, and log field discovery to help you get started quickly. Sample queries are included for several types of AWS service logs.

Create CloudWatch alarms on log filter metrics

Use CloudWatch alarms to report on out-of-bound conditions discovered in log files.



In the diagram, the following processes occur:

1. A CloudWatch log group that is named *HttpAccessLog* contains aggregated log data. The log data is collected by the CloudWatch agents that are installed on one or more EC2 instances.
2. An administrator then creates a custom CloudWatch filter on the log group. The filter searches the log data for *404 page not found* HTTP error messages in the log.
3. Each match increments a custom CloudWatch metric for the *404Count* metric.
4. The administrator then uses that metric to trigger a CloudWatch alarm when the *404Count* metric exceeds a specified value (which is shown as *x*).
5. When the CloudWatch alarm is triggered, a notification is sent—in this case, the notification goes to the IT support team.

Typical log formats

Example:

Apache httpd logs are configured by using a substitution string in httpd.conf

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
```

Result:

A space-delimited string contains information on each HTTP(S) request

```
127.0.0.1 - marcia [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif  
HTTP/1.0" 200 2326
```

Application logs typically generate data in a standardized format. To be able to parse relevant data out of the log files successfully, you must understand the log format.

The example shows an access log file that is generated by the Apache web server application. This log file is typically located at **/var/log/httpd/access_log** on the EC2 instance where the web server is installed. Each time the web server receives an HTTP request, a new line is created in the access log.

In this example, each log entry contains the following:

- **%h** is the IP address of the host machine that made an HTTP(S) request.
- **%l** is the identity of the client machine. This information is typically not available, which is indicated by the dash character (-) output in the example result.
- **%u** is the user ID of the person who requests the information. If the website did not require a login, this information will not be known. In the example, a user named *marcia* made the request.

- `%t` is the time that the request was received.
- `%r` is the request line. It includes the type of HTTP(S) request that was received, such as GET. It also includes what resource was requested, such as an HTML page, a graphic file, or a stylesheet file.
- `%S` is the HTTP status code that the web server replied with. In the example result, the status code is `200`, which indicates success, but you might find other codes, such as `404` (resource not found) error codes.
- `%b`, which is the last part, records the size of the object that was returned to the requesting client.

Amazon CloudWatch filter patterns

Filter patterns

Case-sensitive

- Multiple terms are allowed in a metrics filter pattern (however, all terms must appear in the log event to be a match)
- Example:
Create a metric for all results with the string *html* anywhere in the request, and any HTTP 400-series (client) error

```
[ip, user, username, timestamp, request = *html*, status_code = 4*, bytes]
```

You must define a filter pattern to create a *metric filter*. To do this in the AWS Management Console:

1. Open the [CloudWatch console](#).
2. In the navigation pane, choose **Logs**.
3. In the contents pane, select a log group, and then choose **Create Metric Filter**.
You can now define a *filter pattern*.

To create filter patterns, you must comply with the established filter pattern syntax. Be aware that filter patterns are case-sensitive.

Multiple terms are allowed in a metrics filter pattern. However, all terms must appear in an individual log event for there to be a match. For example, if you have a filter pattern of *ERROR Exception*, it will match only the log entries that contain both *ERROR* and *Exception*.

To learn more about filter syntax, refer to [Filter and Pattern Syntax](#).

Project time

Troubleshooting Knowledge Base

- Have your copy of the Troubleshooting Knowledge Base template open for editing
- Create new entries as you encounter troubleshooting situations
- Categories of focus for this module:
 - *Foundational IT*
 - *Monitoring & Reporting*



11

Be ready to create some entries based on the topics that will be covered in this module. These topics include the *Foundational IT* and the *Monitoring & Reporting* categories.

Key takeaways



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

12

- Amazon CloudWatch Events can trigger services, such as AWS Lambda, based on near-real-time events.
- Amazon CloudWatch Logs can collect application logs by event-based filtering metric data points.

aws re/start

In summary:

- Amazon CloudWatch Events can trigger services, such as AWS Lambda, based on near-real-time events.
- Amazon CloudWatch Logs can collect application logs by filtering metric data points for events.