Root Cause Explorer

Use Root Cause Explorer for AWS to troubleshoot incidents with apps and microservices running on AWS, and isolate root causes.

Availability

<table>
<thead>
<tr>
<th>Account Type</th>
<th>Account Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud Flex</td>
<td>Trial, Enterprise</td>
</tr>
<tr>
<td>Credits</td>
<td>Trial, Enterprise Operations, Enterprise Suite</td>
</tr>
</tbody>
</table>

Root Cause Explorer is an add-on to the AWS Observability Solution. It relies on AWS CloudWatch metrics.

Overview

Root Cause Explorer helps on-call staff, DevOps, and infrastructure engineers accelerate troubleshooting and root cause isolation for incidents in their apps and microservices running on AWS. Root Cause Explorer helps you correlate unusual spikes, referred to as Events of Interest (EOIs), in AWS CloudWatch metrics, using the context associated with the incident. Such incident context includes timeline, AWS account, AWS region, AWS namespaces, resource identifiers, AWS tags, metric type, metric name and more. Given an alert, for instance, a microservice in AWS us-west-2 experiencing unusual user response times, an on-call user can use Root Cause Explorer to correlate EOIs on over 500 AWS CloudWatch metrics over 11 AWS service namespaces (such as EC2, RDS, and so on) to isolate the probable cause to a specific set of EC2 instances, serving the given microservice in AWS us-west-2 that may be overloaded.

Root Cause Explorer supports the following AWS namespaces by processing CloudWatch metrics data and computing EOIs:

- AWS/EC2
- AWS/EBS
- AWS/ELB
- AWS/ApplicationELB
- AWS/NetworkELB
- AWS/Lambda
- AWS/RDS
- AWS/Dynamodb
• AWS/API Gateway
• AWS/ECS
• AWS/Elasticache

Root Cause Explorer can also work with EC2 and EBS metrics collected by [Host Metrics Sources](#) configured on installed collectors that run on your EC2 hosts. In addition, Root Cause Explorer can leverage AWS X-Ray to correlate spikes in service metrics to AWS infrastructure metric spikes.

The screenshot below shows the Root Cause Explorer UI.

![Root Cause Explorer UI](image)

### Troubleshooting concepts

Root Cause Explorer is built to enable six concepts that accelerate troubleshooting and issue resolution. These concepts should be familiar to on-call staff, DevOps, and infrastructure engineers.

**Concept 1: Abnormal spikes are symptoms of an underlying problem**

A spike in a metric on a resource is a sign of an underlying problem. Larger spikes compared to the expected baseline and longer-lasting spikes require closer attention than other spikes.

An abnormal spike in a metric is a statistical anomaly. Root Cause Explorer leverages spikes and adds additional context to them to compute Events of Interest (EOIs). EOIs are constructed based on modeling the periodicity of the underlying AWS Cloudwatch metric on each resource in your AWS account to create resource-specific baselines. The periodicity of a metric can be daily, weekly, or none. EOIs also leverage proprietary noise reduction rules curated by subject matter experts. One example of a rule is how long the system watches an anomalous metric before detecting an EOI. Similarly, EOIs on metrics that have an upper bound (for example, CPU utilization cannot exceed 100%) are subject to additional curation rules.
Concept 2: Context and correlation of spikes are essential strategies for root cause exploration

In a complex system, many resources may behave anomalously within the short time range of an incident. In the figure below, an application that is experiencing throttling at the DynamoDB level is likely to exhibit symptoms in the same time range at the EC2, ELB Target Group, and ELB levels. Root Cause Explorer leverages this insight to correlate EOIs based on the following dimensions:

- AWS account (account id)
- Time range
- AWS region
- AWS Namespace
- Entity (resource identifier)
  - If an AWS X-Ray source is configured, services show as entities in the entity dimension.
- AWS tags
- Golden signals: error, latency, throughput, bottleneck
- Metric name
  - If an AWS X-Ray source is configured, throughput, load, latency, and error metrics corresponding to service entities are shown in this dimension.
- Advanced filters
  - Metric periodicity
  - Metric stability
  - Intensity—the extent of drift from baseline
  - Duration of EOI
  - Positive or negative drift. Negative drifts can lead to an incident. Positive drifts typically relate to metrics that have bounced back from an abnormal level, indicating recovery. However, not all positive drifts are good: for example, a down-spike in CPU utilization on an EC2 instance may be the result of a breakage in connected upstream resources.

In large deployments, thousands of AWS CloudWatch metrics may be anomalous over the course of an outage or incident, making it impossible for an on-call user to deduce which resource(s) may be related to the root cause. With the ability to correlate EOIs based on context, Root Cause Explorer can significantly accelerate incident triage and root cause isolation.
Concept 3: Connections between resources and services help pinpoint root cause

In a complex system, knowing the connections between resources and the services they serve can help a user trace problems from top-level symptoms to upstream root causes. In the figure below, an application that is experiencing throttling at the DynamoDB level will likely exhibit symptoms, in the form of abnormal spikes, at connected EC2 instances, ELB Target Group, and ELB levels. Root Cause Explorer discovers the topology of your AWS infrastructure using its AWS inventory source. This topology helps Root Cause Explorer group anomalous metrics, for example:

- A single abnormal spike on a single resource, like an unusual CPU spike on an EC2 instance.
- A disparate group of abnormal spikes on a single resource, like an unusual "Network In" spike and an unusual "Network Out" traffic spike on a single EC2 instance.
- Spikes are also grouped based on statistics for a given metric on a single entity. For example, if there are anomalies for Average, Max and Sum on a certain metric (provided they occur in the same time range) on an EC2 instance, they are grouped together.
- A group of similar unusual spikes on a collection of resources that are members of an EC2 autoscaling group or ELB target group.

For resources like API Gateway and Application Load Balancers, special notation and logic is used to drive grouping of EOs, given that these are parent entities that enclose other layers in an AWS stack. For API Gateway, Events of Interest are computed for the following combinations:

- API Name
- API Name and Stage
- API Name, Stage, Resource, and API Method

So, an EOI grouped on an API Gateway entity might consist of EOs on entities derived from any of the following entity hierarchies:

API Name only, for example OrderManagement
API Name::stage, for example OrderManagement::Prod
API Name::stage::resource::method, for example, OrderManagement::Prod::/:POST

In such a case, the three EOs would be grouped together, in conjunction with the entity/topology derived grouping.
Concept 4: Earlier spikes are closer to root cause

In a complex system, resources or services that break at the early stages of an incident are closer to the probable cause than resources that experience spikes later. Root Cause Explorer exploits this insight to display spikes on a timeline.

Concept 5: Real root cause requires log exploration

Root Cause Explorer helps triage the first level of root cause which can then drive quick recovery. However, it is also important to understand what caused the system to get into the state that caused an incident. This often requires exploring logs associated with an application or microservice. In the example in the figure below, the real root cause for DynamoDB throttling spikes is a change in the Provisioned IOPS setting on a table. Lowering this setting, while lowering AWS costs, can also lead to throttling. Such a configuration change might be evident in AWS CloudTrail logs associated with DynamoDB.
Concept 6: Golden signals help organize root cause exploration

If you’ve read the Google SRE handbook, you’ll be familiar with the golden signals of load, latency, bottleneck and errors. In a nutshell, errors and latency are signals that most affect users because your service is either inaccessible or slow. Bottleneck and load signals are likely early symptoms (and probable root causes) that may lead to latency and errors. Root Cause Explorer classifies each AWS CloudWatch metric into one of the golden signals to help users navigate spikes using golden signals and arrive at the root cause.

Set up Root Cause Explorer

Before you begin, ensure that your organization is entitled to the appropriate features. The account types and levels that support Root Cause Explorer are listed in Availability, above. The AWS Observability Solution is a prerequisite.

You set up Root Cause Explorer using an AWS CloudFormation template. The template installs the AWS Inventory Source and optionally, the AWS X-Ray source, in your Sumo Logic account. The AWS Inventory Source collects metadata and topology relationships for resources belonging to the namespaces listed below:

- AWS/EC2
- AWS/EBS
- AWS/ELB
- AWS/ApplicationELB
- AWS/NetworkELB
- AWS/Lambda
- AWS/RDS
- AWS/Dynamodb
- AWS/API Gateway
- AWS/ECS
- AWS/Elasticache
- AWS/Autoscaling. Note that Auto Scaling data is used only for topology inference. CloudWatch metrics related to Auto Scaling groups are not supported at this time.

If you don’t already have the Sumo Logic CloudWatch Source for Metrics configured, the template will install the source to collect AWS CloudWatch metrics from the account permissioned by the credential provided in the template. The CloudFormation template gives you the option to configure an AWS X-Ray source, if required.

The CloudFormation template relies on the IAM role policies listed in the Appendix below.

Root Cause Explorer features

Root Cause Explorer adds dashboards at the following levels in the AWS Observability hierarchy:

- Account
- Region
• Namespace
• Entity

Each AWS Observability dashboard shows EOIs filtered for that level in the hierarchy. Each dashboard renders a scatter plot of EOIs at the appropriate level in the Explore hierarchy. The x-axis shows the duration based on start and end time of the EOI. The y-axis renders the intensity, measured by the percent of drift from the expected value.

From the Events of Interest panel at the account, region, namespace, or entity levels, launch the Root Cause Explorer tab to begin troubleshooting. Note that the context of the EOI is carried over from the Explore view. For example, if the Events of Interest view is launched from the region-level Events of Interest dashboard, the region filter will be pre-filled in the Events of Interest tab.

Next, change the time line to match the context—for example, if you know that an incident happened in the last 60 minutes, pick that duration in the duration picker. If you are concerned about errors, pick the Error legend in the EOI panel to filter EOIs by metric type. Click an error EOI to view its details.

In the screenshot below, an EOI on an EBS volume is shown. Click the EOI bubble to view its details and the details of the underlying time series in the right-hand panel. Next, click the namespace filter and view the list of impacted namespaces with their count of EOIs. Pick the top namespaces based on EOI count—these represent the prime suspects with respect to the incident. The metrics tab shows the time series underlying each EOI as shown in the screenshot. The Related tab presents the logs and dashboards that are most related to the entity on which the EOI is depicted.
Among the search filters in Root Cause Explorer, the Advanced Filters provide five dimensions you can use to narrow down EOIs, as shown below. Each dimension indicates the number of associated EOIs. The dimensions are:

- **Impact.** The EOI is positive (for example, a decrease in latency, errors, bottleneck metrics) or negative (for example, an increase in latency, errors, bottleneck metrics). Note that a positive impact is not necessarily a good thing: a CPU metric that has dropped significantly may imply problems in microservices that are upstream of the node experiencing the drop in CPU utilization.

- **Intensity.** The extent of drift from the expected value of a metric—classified as High, Medium or Low. Other things being equal, high intensity EOIs require more attention than others.

- **Duration.** How long a metric has been anomalous.

- **Seasonality.** Seasonality of the metric, on a 0 (low) to 100 (high) scale. This adds context and eliminates false positives in time series data that may otherwise look anomalous due to the presence of periodicity.

- **Stability.** Stability of the metric, based on a proprietary algorithm, on a 0 (low) to 100 (high) scale. EOIs on metrics that are usually stable are probably more indicative of a root cause than other metrics.
Root Cause Explorer UI tips

This section has tips for working with the Root Cause Explorer UI.

**Zoom in**

You can zoom in on a particular time range by dragging to select that range. To zoom out, click the magnifying glass icon in the upper right corner of the visualization.

**About EOI stats**

When you click an EOI, a popup appears that displays key information about the event. The Stats line shows the latest average, maximum, and minimum values of the median of the metric over each 10m segment during the Event of Interest. For example, suppose an EOI lasts 30 minutes. In each 10 minute segment, the median value of the metric underlying the EOI is 6, 8, 4. The stats maximum is 8, the minimum is 4 and the average is 6.

The suffix for a stat indicates units of measure:

- m indicates a thousandth, or $10^{-3}$
- k indicates thousands, or $10^3$
- M indicates millions, or $10^6$

<table>
<thead>
<tr>
<th>Type</th>
<th>ELB App</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity</td>
<td>nite-events</td>
<td></td>
</tr>
<tr>
<td>At</td>
<td>us-west-1, 246946804217</td>
<td></td>
</tr>
<tr>
<td>Metric</td>
<td>TargetResponseTime</td>
<td>Average</td>
</tr>
<tr>
<td>Stats</td>
<td>Avg 110m, Max 110m, Min 110m</td>
<td></td>
</tr>
<tr>
<td>Drift</td>
<td>675% for 10 minutes</td>
<td></td>
</tr>
<tr>
<td>Start</td>
<td>12/08, 04:10 pm</td>
<td></td>
</tr>
<tr>
<td>End</td>
<td>12/08, 04:20 pm</td>
<td></td>
</tr>
</tbody>
</table>
Filtering EOs

Events of Interest can be grouped because of causal or noise reduction features. For example, CPU spikes on 10 EC2 instances belonging to the same autoscaling group are grouped into a single EOI in the scatter plot. In this case, there is one parent EOI and 10 child EOIs.

Filter counts behave as follows:

- For account and region filters, the parent EOI count is shown. In the example above, a count of 1 is shown in the account and region filters.
- For namespace (EC2) and metric filters, the child plus parent EOI counts are shown, 11 in the example above.
- For the entity filter, the parent EOI count is shown on the parent entity while the child EOI counts are shown on each child entity. A count of 1 for each parent and child entity is shown in the filter in this example.

### Four-step troubleshooting methodology with example

There are four steps to root cause exploration:

**STEP 1:** Find a time range of interest based on the incident timeline.

**STEP 2:** For a given AWS region, view root cause guidance, using the Top Contributing Entities panel.

**STEP 3:** Toggle context filters (for example, metrics, golden signals, AWS tags, Advanced Filters) to further isolate the root cause.

**STEP 4:** View time series and logs to analyze the true root cause.

In the figure below, imagine an on-call user supporting a mission-critical application on AWS. The application uses three AWS services:

- ELB (Application Load Balancer)
In this scenario, a developer has reconfigured DynamoDB to use lower-provisioned IOPS (Input/Output Operations Per Second). As AWS charges for DynamoDB based on provisioned Read/Write Capacity Units, cost optimization could be a motivation for the developer to make the change. As load on the application increases, this change could result in spikes in the following CloudWatch metrics:

- ELB errors: HTTP 5xx Count
- ELB errors: HTTP 4xx Count
- ELB: UnHealthyHostCount
- EC2: CPU Utilization, Network
- DynamoDB: ThrottledRequests, ReadThrottleEvents, WriteThrottleEvents

Of course, in a real situation, only the top level symptom, in this case, either the HTTP 5xx Count or HTTP 4xx Count might be apparent to the on-call user. The troubleshooting challenge is to rapidly isolate the causal chain as illustrated in the flow using Root Cause Explorer. To set the stage for root cause exploration, we assume you perform the following steps before launching Root Cause Explorer:

1. View an alert indicating that "ELB 5xx (unhealthy targets) has spiked in AWS account = 1234". This could be an alert on CloudWatch metrics triggered by a Sumo Logic metric monitor.
2. In AWS Observability, navigate to the AWS account = 1234.
3. In AWS Observability, launch the Root Cause Explorer tab from AWS account = 1234 node.

In Root Cause Explorer, perform the following steps:

1. Narrow the time range and region based on the alert context or the AWS Observability dashboards. Use the zoom in feature on the x-axis of either the scatter plot or the histogram.
2. Review the root cause guidance under the Top Contributing Entities panel. This panel incorporates the start time of spikes, duration, and the number of spikes and golden signals to compute a list of entities that are most related to the root cause. Click Apply to filter EOIs based on this guidance. Often, this might be adequate to diagnose the first-level root cause—if so, skip to step 4. Otherwise, proceed to step 3.
3. If required, toggle the metric name filter and Advanced Filters or AWS tags filter count to further analyze EOIs that coincide with the incident timeline.
4. Use related Dashboards in Root Cause Explorer to go to the "AWS-DynamoDB events" Dashboard to view the dashboard for DynamoDB table = 1234.

Then, in AWS Observability (AWS DynamoDB - Events dashboard, All Table Events panel), look for the Update Table event, note that user = Joe has updated the table’s provisioned throughput. This is the root cause.

Appendix

Amazon CloudWatch Source Metrics Source

For information about Sumo Logic's CloudWatch source, see Amazon CloudWatch Source for Metrics.

AWS Inventory Source

The AWS Inventory Source collects the inventory of AWS resources in your AWS account, such as EC2 and RDS instances, including all metadata and tags applied to those resources. We use this data to construct a topology of resources, such as which resource talks to or depends upon which other resources, and so on. The CloudFormation template configures the source with the read permissions listed below. However, data is only collected for the namespace provided to the CloudFormation template.

Permissions

```json
{
   "Statement": [{
      "Action": [
       "apigateway:GET",
       "autoscaling:Describe*",
       "cloudfront:GetDistributionConfig",
       "cloudfront:ListDistributions",
       "cloudtrail:DescribeTrails",
       "cloudtrail:GetTrailStatus",
       "cloudwatch:Describe*",
       "cloudwatch:Get*",
       "cloudwatch:List*",
       "codedeploy:List*",
       "codedeploy:BatchGet*",
       "dynamodb:Describe*",
       "dynamodb:List*",
       "ec2:Describe*",
       "ecs:Describe*",
       "ecs:List*",
       "elasticache:Describe*",
       "elasticache:List*"
      ]
   }
}
```
"elasticfilesystem:Describe*",
"elasticloadbalancing:Describe*",
"elasticmapreduce:Describe*",
"elasticmapreduce:List*",
"es:Describe*",
"es:List*",
"health:Describe*",
"kinesis:Describe*",
"kinesis:List*",
"lambda:List*",
"mq:Describe*",
"mq:List*",
"rds:Describe*",
"rds:List*",
"redshift:Describe*",
"route53:List*",
"sqs:ListQueues",
"ses:GetSendStatistics",
"ses:GetSendQuota",
"sns:List*",
"tag:GetResources",
"tag:GetTagKeys",
"tag:GetTagValues",
"xray:BatchGetTraces",
"xray:Get*",
"pi:DescribeDimensionKeys",
"pi:GetResourceMetrics"
],
"Effect": "Allow",
"Resource": "*
}
],
"Version": "2012-10-17"
}

**AWS X-Ray Source**

The AWS X-Ray source collects the AWS X-Ray service graph, as well as service-level metrics such as latency, throughput, and error rate. The service graph allows us to figure out which service depends on which other service(s).
Known issues

• When you create or update an AWS X-Ray source or an AWS Inventory source, it is possible to save the source without a name. The solution is to delete the source and re-create it.

• Some ad blockers prevent Root Cause Explorer from rendering as it contains “analytics” in its URL. The solution is to add Root Cause Explorer to the allowed list in your ad blocker.