

# Creating Actionable Alerts to Maximize Resolution Speed



# Creating Actionable Alerts to Maximize Resolution Speed

When issues occur within your DevOps or IT stack, categorizing and correctly routing alerts is the first requirement needed to reduce mean time-to-resolve (MTTR). This is the reason that on-call and alerting solutions are being rapidly adopted by operations teams in all industries. Ensuring that the alerts are actionable, however, is not always common practice, yet has the ability to further accelerate response time-ensuring that issues are addressed before business is impacted.

This paper serves as a introductory guide to creating actionable alerts, ensuring that responders efficiently learn of incidents, but also are provided context and guidance to further reduce MTTR.



# What is an actionable alert?

In the world of incident response, an actionable alert is an alert which describes an issue, is routed to the right people at the right time, and communicates not only the urgency, but the issue's scope of impact. Preferably, the alert directs responders in the right direction to avoid or minimize damage.

## Actionable alerts:

- Notifies correct people at the right time
- Suggests how to start investigating the issue with contextual information
- Enables common tasks to be performed through automation

This screenshot shows a non-actionable alert in the OpsGenie interface. The alert is titled "#46 CPU > 90% for 5 minutes" and has a status of "x1". It is marked as "No owner" and was received on "Feb 13, 2018 6:57 PM". The interface includes a "Select All" checkbox, a date range filter, a "Sort by" dropdown set to "Created", and a settings gear icon. Action buttons for "Ack", "Close", and "Open" are visible on the right side of the alert card.

*Non Actionable alert example*

This screenshot shows an actionable alert in the OpsGenie interface. The alert is titled "#47 ALARM: \"SQS Queue alert\_automation\_commands\_queue messageCount Too High\" in US West (Oregon)" and has a status of "x1". It is marked as "No Owner" and was received on "Feb 13, 2018 8:31 PM". The alert card includes tags for "autoClose", "cloudwatch", "delay", and "us-west-2". A dropdown menu is open, showing various actions: "Assign", "Add Team", "Add Recipient", "Escalate to Next", "Snooze", "Delete", "Join video-call", and "Retrieve logs". The "Details" tab is selected, showing the source as "serhat+can@opsgenie.com", integration as "Default API (API)", and teams as "ops\_team". The description states: "Threshold Crossed: 1 datapoint [783.0 (12/02/18 13:56:00)] was greater than or equal to the threshold (100.0)". The priority is "P2 - High". The "runbook" section provides instructions: "SQS CloudWatch alerts mainly denotes that a queue is not being processed fast enough, the reason why the queue is being processed slow should be find out by checking que transactions on new relic , queue dashboard on New Relic [https://insights.newrelic.com/accounts/4444/dashboards/09999](\"https://insights.newrelic.com/accounts/4444/dashboards/09999\") ."

*Actionable alert example*

## Notifying the correct people at the right time

As stated in introduction, most operations teams embrace alerting and on-call management solutions for the ability to define on-call schedules and route alerts when incidents occur. With very little configuration, these applications can look at the alert details, apply some filtering to suppress noise, and send them to predesignated teams. More modern tools enable users to introduce complex logic to their routing workflows. Using specific details from the alert, combined with timing, users are better able to determine urgency and improve routing. Alert enrichment plays a role here. For example, if an error is reported from a specific server that affects a users' "Gold Level Customers", a corresponding property or tag can be added to the alert specifying this fact and therefore aid in routing it with elevated priority.

## Suggesting how to start investigating the issue with contextual information

Adding tags and properties to an alert not only aids in effective routing, but saves the responder time by providing context for the issue. Less time can be spent investigating and formulating a response plan when additional information is given within the alert. There is often no single source of information for determining urgency and impact of an incident. Information is scattered throughout logging, monitoring, ticketing, and configuration management tools. Avoid context switching by providing supplemental details during alert creation.

In addition, other information is helpful for a fast response. Preparing a runbook link is a good example of an enrichment done during alert creation. Runbooks list the procedures and operations for responders to follow. Having a runbook is solid proof of a planned effort and is key to reduce the MTTR.

Creating a runbook is a continuous process. Instructions must be crystal clear and everyone needs to understand the same meaning from a statement. After an incident, it is the responder's duty to reflect learnings and update the instructions.

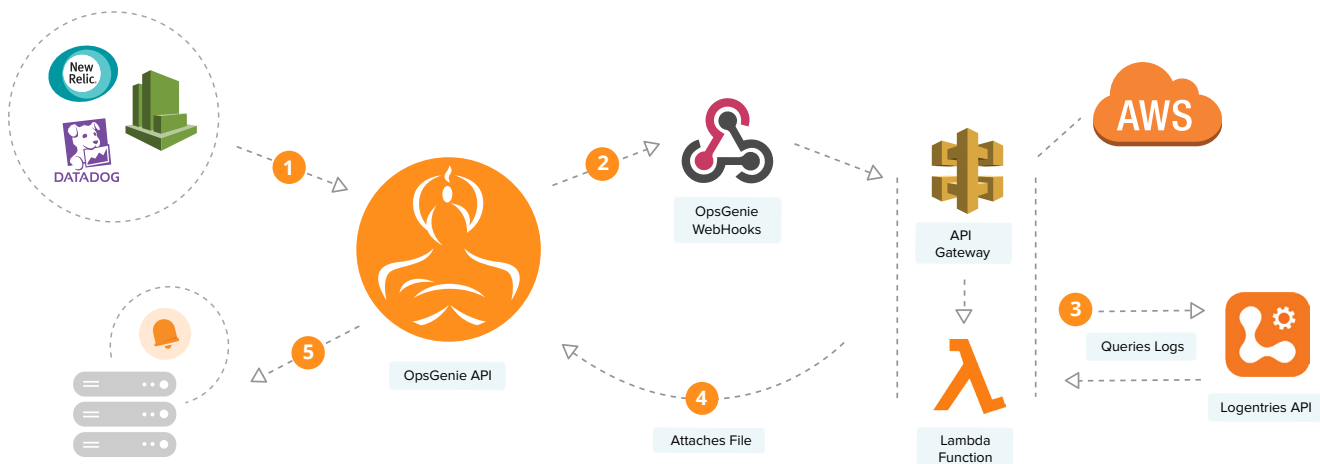
## Enabling common tasks to be performed through automation

Once an alert is enhanced with contextual information, more value can be realized by automating the retrieval of supporting information and providing one-click actions for responders. Tasks that can be automated include gathering investigative information, taking remedial actions, and automatically initiating communication and collaboration.

- **Gather more investigative information**

Investigative actions automate the retrieval of any relevant information. Time consuming jobs such as manually logging into a service and running a query can be automated using action buttons attached to the alert.

Retrieving application logs, attaching graphs, or just pinging a server saves significant amount of time during the investigation of the issue. Moreover, automating the retrieval of these data using predefined queries and actions reduces the chances of human error.



Use case from a joint webinar by OpsGenie and Logentries:

<https://www.opsgenie.com/webinars/logentries-2-use-cases-logs-incident-response>

- **Take remedial actions**

Having one-click remedial actions saves a significant amount of time by automating complex workflows. If you have a repetitive task which can resolve the incident, make it an action on the alert. Restarting a process or a server is a common remedial action example. Another example is to add new servers to the load balancer to handle a spike in the request count. Automation makes sense for most repetitive tasks, but the key is providing flexibility so that people can still decide when to take a critical action, especially when stakes are high.

- **Initiate communication and collaboration**

In case of an high-priority incident, incident response team members need to talk to each other. There has to be a reliable, unified tool that is easily accessed. A button on the alert which allows team members to join a call or video conference bridge is a great example of alert enrichment. Enriched alerts like these help your team resolve the incident faster by bringing the responders and stakeholders together instead of just being a messenger. OpsGenie's Incident Command Center (ICC) is another unified way of bringing the incident response team together. Teams gain access to the alerts that are being worked on and can exchange information from an enriched view.

Enriching alerts can drastically impact MTTR. Alert enrichment is a continuous process that lends itself to responders building off of previous experiences to streamline their workflows and response procedures. System administrators or operation engineers in NOCs make use of these guidelines to automate as much as possible, leaving only critical tasks that may need human judgement during high-impact incidents.

## Real life scenario using OpsGenie

OpsGenie gives you the complete toolset to create actionable alerts. Capabilities include:

- Custom actions
- Filtering
- Deduplication and grouping
- On-demand data retrieval using webhooks
- Referencing and creating runbooks
- Access to call and conference bridges

In this part, a reference use case will be used to show how alert enrichment done in OpsGenie.

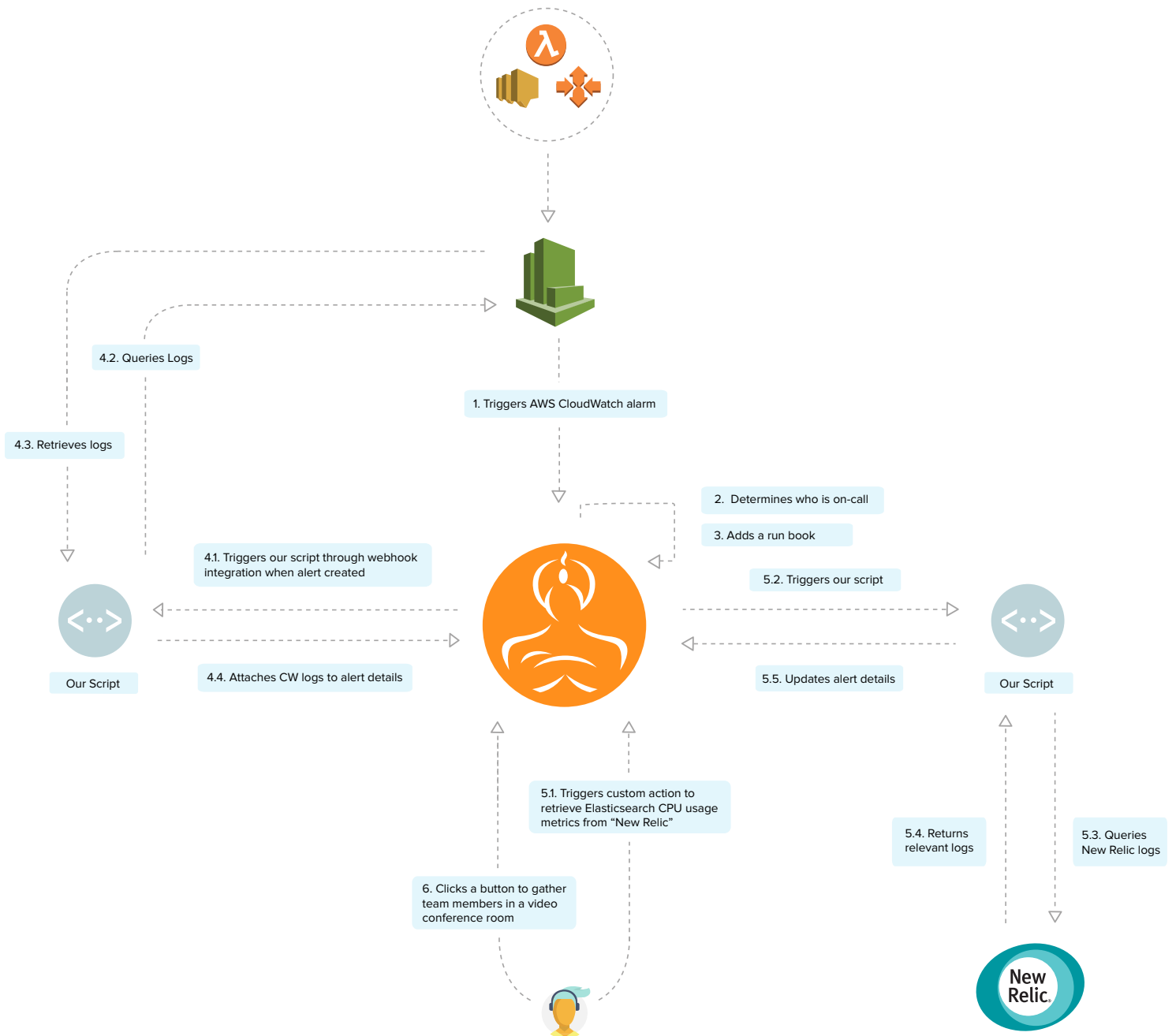
The use case consists of several steps designed around the lifecycle of the incident. Starting with crossing the threshold in AWS CloudWatch, OpsGenie will receive the alert, and enrich it to attach a runbook, take investigative actions, and collaborate using video bridge all within the same interface.

In this use case, AWS CloudWatch triggers an alarm when AWS Lambda function fails. AWS Lambda is Amazon's serverless computing service. Our function accesses DynamoDB and ElasticSearch to save and query data.

Step by step flow is like this:

1. AWS CloudWatch alarm is triggered
2. OpsGenie receives the alert and determines who is on-call
3. Runbook is added to the alert during the alert creation
4. A script retrieves CW logs and attaches it to the alert
5. Custom action is executed to retrieve Elasticsearch CPU usage metrics from NewRelic
6. Team joins a video bridge with one click

## Real life scenario using OpsGenie





## Step 1: AWS CloudWatch alarm is triggered

Amazon CloudWatch provides monitoring for Amazon Web Services (AWS) and the applications that run on AWS. It triggers user-defined alarms by watching metrics. OpsGenie acts as a dispatcher for these alarms.

In the sample use case, AWS CloudWatch works in conjunction with AWS Lambda. It is the default and easiest way of logging somethings and searching afterwords. Users create AWS CloudWatch Alarms to get notified when certain conditions like unhandled exceptions, time-outs, or out of memory exceptions happen.

When a function fails, [OpsGenie CloudWatch integration](#) receives a payload like this:

```
AlarmName: us-east-2 sre-dynamodb-autoscale-put-config Lambda execution failed
functionName: sre-dynamodb-autoscale-put-config
logGroupName: /aws/lambda/sre-dynamodb-autoscale-put-config
NewStateReason: Threshold Crossed: 1 datapoint [1.0 (18/01/18 02:08:00)] was greater than the threshold (0.0).
NewStateValue: ALARM
OldStateValue: INSUFFICIENT_DATA
Region: us-east-2
StateChangeTime: 2018-01-18T02:09:25.285+0000
Trigger: {Dimensions=[{name=FunctionName, value=sre-dynamodb-autoscale-put-config}]}
```

The configuration in OpsGenie looks like this. It maps the received payload on to the alert with the format we want.



The screenshot shows the OpsGenie configuration page for an integration named 'sre-ops-cloudwatch-alarms (AWS CloudWatch)'. The interface is divided into several sections:

- Actions:** A sidebar on the left with a list of actions: 'Ignore', 'Create Alert', 'Lab AWS Create' (highlighted), 'Create Alert', 'Close Alert', and 'Close Alert'. Each action has a gear icon for configuration.
- Settings:** A section at the top right with a dropdown arrow.
- Filter:** A section below Settings with a dropdown menu set to 'Match all conditions below'. It contains two filter rules:
  - Rule 1: 'NewStateValue' equals 'ALARM'.
  - Rule 2: 'TopicArn' contains '495179308371'.
- Alert Fields:** A section for mapping incoming fields to OpsGenie alert fields.
  - Message:** Set to `{{Subject}}`.
  - Alias:** Set to `{{Region}} - {{AlarmName}}`.
  - Priority:** Set to `{{priority}}`.
  - Entity:** Empty.
  - Source:** Set to 'CloudWatch'.
  - Tags:** Set to 'lab'.
  - Actions:** Empty.
  - Description:** Set to `{{AlarmDescription}} {{NewStateReason}}`.
  - Extra Properties:** A list of properties:
    - `Trigger: {{Trigger}}`
    - `AlarmName: {{AlarmName}}`
- Field Picker:** A panel on the right with a list of available fields: 'AlarmDescription', 'NewStateReason', 'NewStateValue', 'Subject', 'AlarmName', 'Region', 'StateChangeTime', 'OldStateValue', 'Trigger', and 'TopicArn'. A handwritten note 'Drag fields into form' with an arrow points to this panel.

## Step 2: OpsGenie receives the alert and determines who is on-call

In this step, predefined on-call schedules, routing rules, and escalations are used to determine who is on-call and should be notified. Related information is attached to the alert automatically.

In complex systems or NOC's with hundreds of teams and services, an external service is often used to find out who is responsible for what. This is also an alert enrichment.

 Update Escalation 

Name


sre-ops\_escalation


Assign To Team


sre-ops


Description

Escalation Rules

 If the Alert is not acknowledged, **immediately** notify on call users in **sre-ops\_schedule**

 If the Alert is not acknowledged, **5 min** after creation, notify next user in **sre-ops\_schedule**

 If the Alert is not acknowledged, **10 min** after creation, notify all members of **sre-ops**

 Add Rule

☐ Repeat

Cancel

Update

OpsGenie

## Step 3: Runbook is added to the alert during the alert creation

Runbooks are guides that responders depend on when taking actions during war time. They can easily be added to the alert description or details from the integration's advanced settings options. Adding a link to a runbook here is another option.

Description:



{{AlarmDescription}} {{NewStateReason}}

Runbook:

- Check the reduplication count to determine the urgency
- Check attached logs to see if there is an exception
- If logs indicate that there is something wrong with Elasticsearch, retrieve Elasticsearch CPU usage from NewRelic using custom action on the alert to investigate further

## Step 4: A script retrieves AWS CloudWatch logs and attaches it to the alert

OpsGenie can send an HTTP POST request to a web-accessible URL endpoint (what's often referred as webhooks) and pass alert activity data. URL endpoint can be any platform, web server, etc. as long as the URL is accessible from the Web. Webhook data includes the alert activity (create, acknowledge, etc.) as well as a subset of the alert fields (alertId, username, alias, entity, userId) as part of the HTTP request payload (JSON). Also, users can define custom headers to add the webhook call.

For OpsGenie alerts:

OpsGenie to Webhook

Post to Webhook URL for OpsGenie Alerts: ☒

If alert is created

in OpsGenie,

post to url

in Webhook

-

If alert is acknowledged

in OpsGenie,

post to url

in Webhook

-

If a note is added to the alert

in OpsGenie,

post to url

in Webhook

-

If alert is closed

in OpsGenie,

post to url

in Webhook

-

If alert is unacknowledged

in OpsGenie,

post to url

in Webhook

-

If alert's priority is updated

in OpsGenie,

post to url

in Webhook

-

If a custom action is executed on alert

in OpsGenie,

post to url

for this custom action:

leave empty for any

-

+ Add New Action

Alert Filter: Match all alerts

Webhook URL:

Creating your own web server and handling the events on your own is always an option with the webhook integration. Yet, OpsGenie recommends two different ways to ease the development and maintenance.

### **Serverless architectures such as AWS Lambda**

Users often choose serverless services such as AWS Lambda or Azure Functions to integrate with the webhook integration because maintenance and resource utilization isn't an issue in these platforms. Serverless services are also known with their automated scalability so in case of an incident which generates a lot of alerts, the system can easily scale without manual human interaction.

### **Marid Integration**

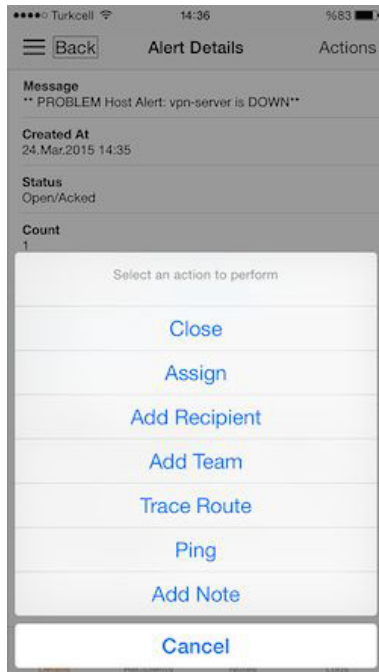
Marid is an integration server, designed to resolve challenges faced in the integration of internal and external systems. Using OpsGenie Marid integration, you can manage a two-way bridge between your system and OpsGenie; forwarding OpsGenie alert activity to your system, and updating the alerts on events from your system. Marid makes it easy to create scripts using preloaded - easy to use - packages.

*If creating an internet facing endpoint is not an option, Marid should be used as a proxy server. In this case, Marid is the way to do it.*

Focusing on the use case, OpsGenie will make a request to the http endpoint using the webhook integration. Once the code has the necessary alert details, it will query the CW logs and attach the data back to the alert. Pseudo code is like this:

- Received HTTP payload with alert details
- Make a request to AWS CloudWatch service by specifying the resource id and time interval
- Make a request to OpsGenie to add the received logs by specifying the alert id.

## Step 5: Custom action is executed to retrieve Elasticsearch CPU usage metrics from NewRelic



Custom Alert Actions are another way OpsGenie allows users to quickly and effectively react to Alerts. Paired with OpsGenie's integration capabilities, Custom Actions include anything relevant such as Ping, Trace Route, or Create Ticket. For a single alert you can define up to 10 custom actions. Executing a custom action is an opportunity for an initial, collaborative investigation into the issue.

Applications often depend on external services to store and retrieve data. In this case, lambda function accesses DynamoDB, and ElasticSearch. In case of a failure alert received from AWS CloudWatch, after investigating the logs, the on-call can decide to check ElasticSearch cluster to ensure that the load increase doesn't affect the quality of the service.

Custom action called "Get Elasticsearch CPU usage from NewRelic" will retrieve the related from NewRelic Insights using its API.

Actions:



Get Elasticsearch CPU usage from NewRelic



OpsGenie can call an HTTP endpoint (webhook) and pass the information that a user executed an action on an Alert when a custom action is executed. OpsGenie also provides the Marid utility as described in Step 4.

Instead of executing the script right after the alert creation like in the previous step, custom actions are executed on demand. It can be executed from web or mobile interface, API, or even integrations like Slack.

The flow of the script is like this:

- Receive HTTP payload with alert id
- Make a request to NewRelic Insights API to retrieve the latest metrics related with the CPU usage of Elasticsearch cluster
- Update alert details by attaching the metrics retrieved

Different monitoring solutions have different capabilities. For example, with Nagios or DataDog, users can attach an image of the resource usage instead of a plain text. This can be very useful to gain fast insights as often anomalies or differences are easy to spot on charts.

## Step 6: Team joins a video bridge with one click

OpsGenie Conference Bridge provides the ability to launch a conference bridge from within the OpsGenie application. It allows you to collaborate with key individuals to resolve incidents using your preferred web conferencing provider. With this capability, after an incident arises, it is easier to organize your team and join the conference bridge. With access to the preset conference bridge details instantly within the incident notification, responders are able to join the call with just one-click.



## Conclusion

Creating actionable alerts is key to reducing MTTR. Accessing contextual insights, taking investigative and remedial actions, and bringing the team together with a click are common ideas for creating them. OpsGenie empowers its users to realize these either by providing native flexible solutions or giving access to the right resources.

### *Additional Resources:*

<https://docs.opsgenie.com/v1.0/docs/callbacks>

<https://www.opsgenie.com/blog/2017/01/monitoring-aws-lambda-functions>

<https://docs.opsgenie.com/v1.0/docs/marid-integration-server-for-opsgenie>

<https://docs.opsgenie.com/docs/aws-cloudwatch-integration>

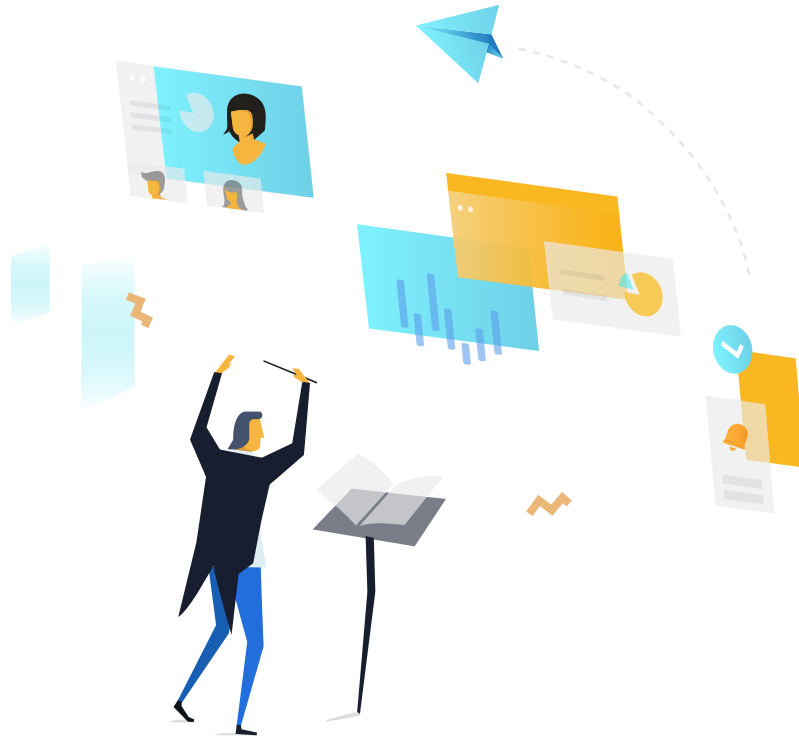
<https://docs.opsgenie.com/v1.0/docs/conference-bridge>

<https://docs.opsgenie.com/docs/custom-actions>

<https://docs.opsgenie.com/v1.0/docs/webhook-integration>

*March 2018*





## About OpsGenie

Founded in 2012, OpsGenie is an alerting and on-call management solution for dev & ops teams. We provide the tools needed to design actionable alerts, manage on-call schedules & escalations, and ensure that the right people are notified at the right time, using multiple notification methods.

### About the author

---



#### *Serhat Can*

*Serhat is a Technical Evangelist for OpsGenie. He contributed to different parts of OpsGenie as an engineer and now tries to spread the word by coding, writing and talking about DevOps. He is still a proud member of the on-call schedules.*

Sign up for a free  
trial of OpsGenie

Sign up today