

Proactive Incident Resolution

300-level live demo script



Introduction

In this demo, I'll show you how Cloud Pak for Watson AIOps helps SREs and IT Ops proactively identify, diagnose, and resolve incidents across mission-critical workloads.

You'll see how:

- Watson AIOps intelligently correlates multiple disparate sources of information such as logs, metrics, events, and topology
- All of this information is condensed and presented in actionable alerts instead of large quantities of unrelated alerts
- You can resolve a problem within seconds to minutes of being notified using Watson AIOps' automation capabilities

We will be using an application called Quote of the Day, which will serve as a proxy for any type of app. This is a content delivery app that serves up random quotations. The application is built on a microservices architecture, and the services are running on Kubernetes.

1 - Simulating a failure

1.1 - Navigate to the anomaly generator and input sources

Narration

To see how this all works, I'm going to generate an anomaly in our application.

Action 1.1.1

- From the **Sabine Cluster Details** page, click the **Quote of the Day Anomaly Generator** link to launch the anomaly generator web application.

Credentials

This environment is made up of a number of separate systems, each with its own set of systems and credentials. For the time being these are shared credentials.

System
Slack Workspace
Automation Hub
Event Manager (Netcool Operations Insight)
Prometheus
ELK (Kibana)
Quote of the Day Home Page
Quote of the Day Anomaly Generator
Instana
Grafana
OCP Console
QoD OCP/OKD Cluster

Action 1.1.2

- Choose **Rating service failure**.

Home | Manual | Manager

Quote of the Day App
Anomaly Generator

[Reset All Services to Factory Settings](#)

Rating service failure
This use case simulates a failure in the ratings service. It starts with the introduction of a new type of log entry. Then the service latency, CPU and memory usage are all increased.

Author and Image Service Slowage
New log entries are added to author and image services. They both experience sharp increases in response time of their primary services, near 2 seconds.

Rating, author, image service cascade failure
This use case simulates a failure in the ratings service. Minutes later the failures cascade to web, author and image services.

Action 1.1.3

- Choose **Start**.

Home | Source | Manual | Manager

Quote of the Day App
Anomaly Generator
Rating service failure

This use case simulates a failure in the ratings service. It starts with the introduction of a new type of log entry. Then the service latency, CPU and memory usage are all increased.

rating	Start new repeating log warning about memory checksum every 2 seconds
rating	Start log warning about requests for resource from IP address (author service). Repeats every 1.5 seconds.
rating	Increase memory usage
rating	Increase CPU usage
rating	Increase service delay (2s)

Narration

Without Watson AIOps, we would get all sorts of alerts and notifications from multiple sources when a problem occurs.

Prometheus would start firing alerts, but we would have to look at them manually to find out if they are related.

Same with ELK – we might see new types of logs or errors coming in, but again it's time-consuming to determine the relationship. Hundreds of logs are streaming in every minute, making it very difficult for a human to keep up with them.

As we'll see in a moment, with Watson AIOps, all of this information gets correlated and presented in one place. This includes recommendations for how to resolve the incident. We can take action directly from the notification and resolve the incident quickly.

Action 1.1.4

- The anomaly generator will take a minute or so to start showing alerts in Slack. While it is running, navigate to your **Prometheus** tab (you should already have this open from the demo preparation).

Prometheus Alerts Graph Status • Help Classic UI

Active (20) Pending (0) Firing (0)

Show annotation

/jetcprometheus2/_rules_rating.yml > rating

RatingVeryHighRequestLatency (0 active)

name: RatingVeryHighRequestLatency
expr: sum(rate(http_request_duration_seconds_sum[instance='qotd-rating-qotd.apps.sabine-0kd.coc-lba.con:80',method='GET',route='/ratings/*val*',status='2xx']) by()) /
for: 1m
labels:
annotations:
alertGroup: qotd-latency
tags:
modeName: qotd-rating
summary: Rating service response time over past minute is significantly high (> 1.8s)

> RatingVeryHighCPUUsage (0 active)

> RatingVeryHighMemoryUsage (0 active)

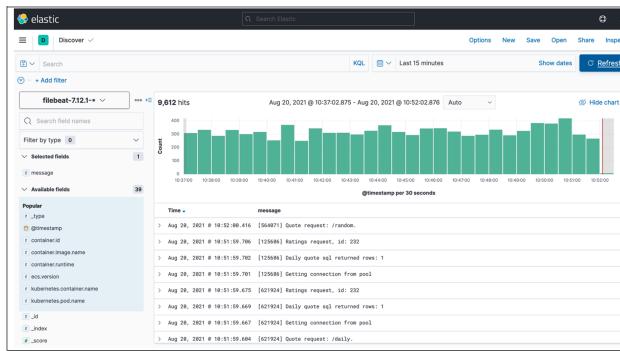
> RatingServiceDown (0 active)

/jetcprometheus2/_rules_web.yml > web

WebVeryHighRequestLatencyRandomQuote (0 active)

Action 1.1.5

- Then, navigate to the **ELK** tab (you should already have this open from the demo preparation).



2 - Getting notified of an emerging problem

2.1 - AIOps formats the notification as a story and adds affected services to it

Narration

Notifications are now appearing in Slack. We're using Slack in this demo, but Watson AIOps also integrates with Microsoft Teams.

Watson AIOps formats the notifications into a "story" using AI to correlate events, metrics, alerts, and logs. Each story brings together the various notifications for all the affected services by the same underlying issue. Imagine if each piece of data presented in the story was a separate notification – we'd quickly be inundated with alerts.

The story is like a home base for action when a problem arises. Instead of manually correlating things across multiple different tools, it's all right here immediately when the notification is received.

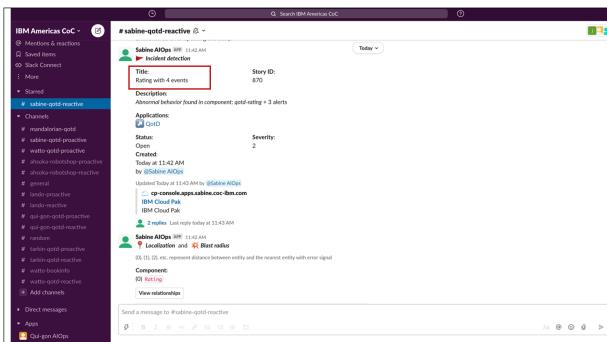
In addition to providing a highly contextualized view of the incident, it enables us to jump to other tools to explore further details. This eliminates tool silos and helps us restore service faster.

This story is telling us there's a problem with the Rating service, which is one of the microservices in our Quote of the Day application.

In the background, the AI has done the work for us. It shows which services are affected and presents us with a curated view of relevant information: the events and alerts that are indicative of the symptoms of this problem, anomalies that Watson AIOps has found in the log files, and similar incidents that have occurred in the past so we can see how they were successfully resolved. We'll explore each of these components in more detail.

Action 2.1.1

- The story starts showing up in Slack.



Narration

When this story first appeared, the only affected service was the Rating service.

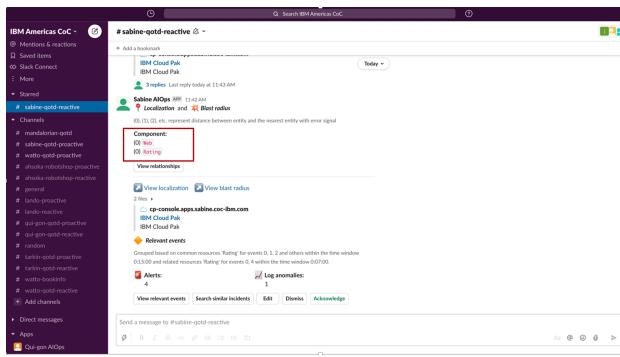
Watson AIOps updates the stories in real time as more information comes in and gets correlated to this story.

So now we can see that in addition to the Rating service, the Web service is also affected. This is now even more critical since the Web service is the customer-facing front end of the application, and we need to ensure that users can still access the app.

We need to find the root cause and fix this problem as quickly as possible.

Action 2.1.2

- Additional affected services are added to the story.



3 - Determining which service caused the problem

3.1 - Review the notification to learn about the issue

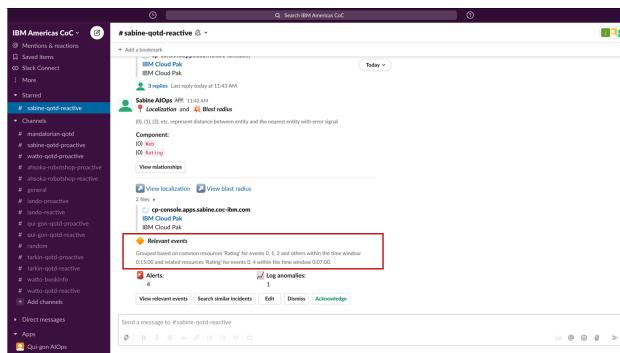
Narration

We need to find out where the issue began, so we can prevent it from causing cascading failures across the components of the application.

Instead of having to go to Prometheus or another tool to look at the alerts, we can see them right here from the notification. Watson AIOps has determined that these events are related, and it provides an explanation for how it determined the relationships. We can see that there are two groups of events based on related resources and the timing of the events.

Action 3.1.1

- Scroll to the **Relevant events** section of the story.



Narration

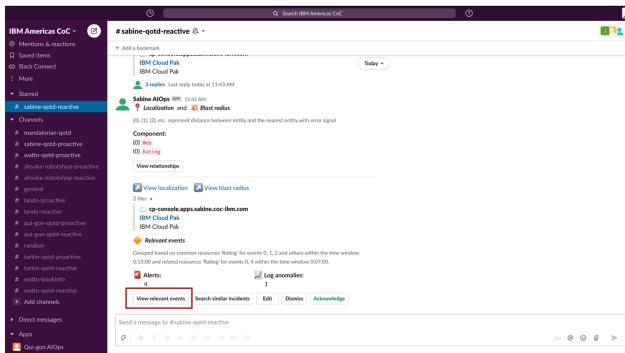
We can inspect the grouped events right here, without searching for them in another tool.

It looks like the memory and CPU on the Rating service increased significantly. This is causing a significant slowdown in the response times on both the Rating and Web services.

Based on this information, it seems that the Rating service is the source of the issue. But let's get a bit more detail – this time from the log files.

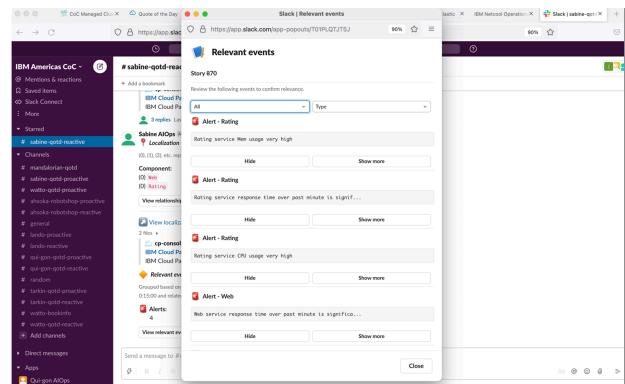
Action 3.1.2

- Click the **View relevant events** button at the bottom of the notification.



Action 3.1.3

- A pop-up will appear with the grouped events.

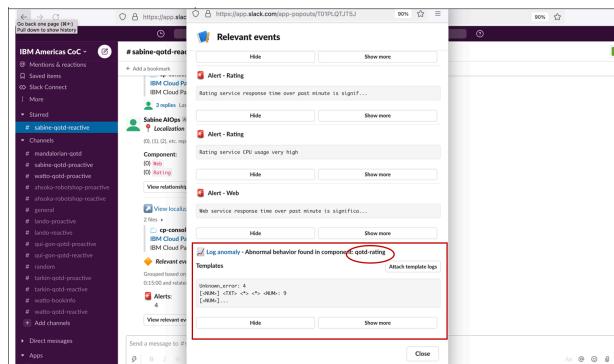


Narration

Instead of needing to go to Kibana and manually sort through the hundreds or thousands of log entries that come in every minute, Watson AIOps has found several anomalies in the log files and presented them here. It trains on the log files of the application when it's operating normally, and it continually monitors for deviations from that baseline. We can see that the anomalies are occurring on the Rating service, which fits with what we saw in the alerts.

Action 3.1.4

- Scroll down past the alerts to show **Log anomaly**.



Narration

Watson AIOps gives us additional context on the anomaly. In this case, the ‘Unknown_error’ anomaly is telling us that Watson AIOps has never seen this type of log before (hence the “unknown”) and that the log message indicates there is some type of error. Watson AIOps is not only looking at the statistical frequency of the type of log, but it is also using Natural Language Processing to analyze the content of the log message to give additional context (in this case that there’s likely an error).

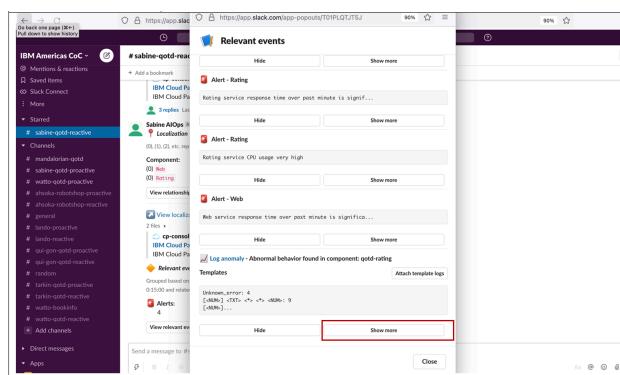
Watson AIOps also explains why the anomaly was flagged. It expected to see zero (0) of this type of log, but it actually saw four (4).

Now we know that there is an unfamiliar log coming from the Rating service, and it’s indicating an error.

This further reinforces what we saw with the alerts - it looks like the Rating service is likely the root cause of this problem.

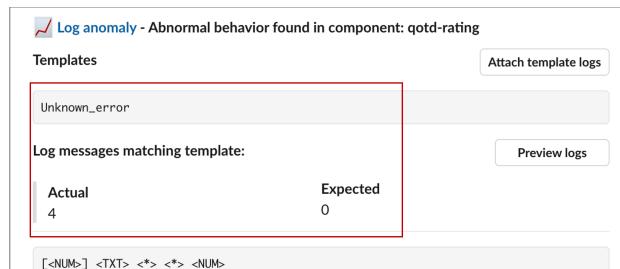
Action 3.1.5

- Click **Show more**.



Action 3.1.6

- Notice that there are four log messages matching the template.



Narration

We can preview the log messages that caused Watson AIOps to find the anomaly.

Action 3.1.7

- For the **Unknown_error** anomaly, click **Preview logs**.

The screenshot shows the 'Log anomaly' interface. At the top, there's a red exclamation mark icon followed by the text 'Log anomaly - Abnormal behavior found in component: qotd-rating'. Below this, there's a 'Templates' section with a 'Unknown_error' button. A 'Log messages matching template:' section follows, containing two columns: 'Actual' (with value '4') and 'Expected' (with value '0'). To the right of these columns is a 'Preview logs' button, which is highlighted with a red box. At the bottom, there's a text area containing log entries: 'WARNING - memory checksum doesn't match eligendi', 'DANGER Will Robinson. Unexpected request for http://ahmad.net from source: 55.249.8.48', 'DANGER Will Robinson. Unexpected request for http://dax.net from source: 121.98.150.77', and 'WARNING - memory checksum doesn't match iure'.

Action 3.1.8

- The **Log preview** pop-up will appear.

The screenshot shows a 'Log preview' pop-up window. It has a title bar with a blue icon and the text 'Log preview'. Below the title bar, it says 'Template: Unknown_error'. The main content area displays the same log messages as the previous screenshot: 'WARNING - memory checksum doesn't match eligendi', 'DANGER Will Robinson. Unexpected request for http://ahmad.net from source: 55.249.8.48', 'DANGER Will Robinson. Unexpected request for http://dax.net from source: 121.98.150.77', and 'WARNING - memory checksum doesn't match iure'.

Action 3.1.9

- Close the **Log preview** pop-up.

The screenshot shows the 'Log preview' pop-up window again. The content area contains a large amount of log data, including multiple entries for 'announceCycle' and 'agentHostLookup' modules. At the bottom right of the content area is a 'Close' button, which is highlighted with a red box. The window has a standard OS X style with a close button in the top right corner.

4 - Getting resolution recommendations

4.1 - Watson AIOps searches similar incidents for recommendations

Narration

Now that we understand a bit more about what's going on, we need to focus on our main goal: resolving the incident as quickly as possible.

Watson AIOps also brings in similar incidents and information regarding how they were resolved. This enhances operational efficiency by leveraging institutional knowledge that may be time-consuming to find otherwise.

Since the alerts seem to point to the Rating service as having increases in CPU and memory, we'll search for incidents that happened with that service.

Action 4.1.1

- Click the **Search similar incidents** button.

The screenshot shows the IBM America CoC interface. On the left, there's a sidebar with various navigation items like 'Monitor & react', 'Saved Items', 'Block Connect', 'More', 'Starred', and 'Search'. Under 'Search', there's a section for '# sabine-qpid-reactive'. In the center, there's a search bar with the placeholder 'Search IBM America CoC...'. Below the search bar, there's a list of entities: 'IBM Cloud Pak', 'IBM Cloud Pak' (with a note 'Last reply today at 11:43 AM'), 'Sabine AIOps', and 'Blat radius'. There are buttons for 'View relationships', 'View location', 'View Max radius', and 'View recent events'. At the bottom of the page, there's a red box highlighting the 'Search similar incidents' button. The URL in the address bar is 'ibm-emea-coc.sabine.coc.ibm.com'.

Action 4.1.2

- Type **rating** into the search box.

The screenshot shows a modal dialog titled 'Search Similar Incidents'. At the top, it says 'Story 870'. Below that, there's a text input field with the placeholder 'Tell us about the problem' and the word 'rating' typed into it. To the right of the input field, it says '194'. At the bottom of the dialog, there are two buttons: 'Close' and 'Search' (which is highlighted in green). The URL in the address bar is 'ibm-emea-coc.sabine.coc.ibm.com'.

Narration

Watson AIOps found two similar incidents. The first one seems like what we're experiencing: 'Rating service overheating and slowing.' Using Natural Language Processing, Watson AIOps went through the comments on the incident and highlighted key relevant information for us (in this case that the incident was resolved by running a runbook). If we can use the same runbook to resolve the current incident, it will make our job that much easier and restore service even faster!

Action 4.1.3

- Review the **Search results** window.

The screenshot shows a 'Search results' window titled 'Story 870'. It displays two incidents found based on the search term 'rating'.
1. Incident 1: Rating service overheating and slowing. Summary: Run book created to **reset fix rating service.** Incident created Jun 24.
2. Incident 2: Cascading service failure starting with Rating service. Summary: ..service **fixes** rating and web service, however reset of author and **image** is also.. Incident created Jun 24.
Buttons at the bottom include 'Back' and 'Done'.

Narration

The description for this incident sounds a lot like what we are observing right now. The Rating service is stressed, with increases in CPU, memory, and latency. It's also affecting the Web service. The comment says there is a runbook that resolved this issue. This is very useful information that may have taken a long time to find out just by searching or asking colleagues. Watson AIOps helped us find this info very quickly.

Action 4.1.4

- Click the link for the first similar incident **Rating service overheating and slowing.** It will open a window in GitLab. If needed, sort comments **newest first**.

The screenshot shows the GitLab issue page for the incident 'Rating service overheating and slowing'.
Summary: The rating service is experiencing significant increases in CPU, memory and latency in primary APIs. Latency at nearly 2 seconds per call. This is affecting the web service.
Comments:

- IBM Hybrid Cloud Center of Competency (@ibmhccoc) closed 1 month ago: Resetting Rating service fixes the problem. Run book created to reset (fix) rating service.
- IBM Hybrid Cloud Center of Competency (@ibmhccoc) 1 month ago: Instana reports rating and web services under stress. The web service calls (asynchronously) the rating service.

Sort dropdown: Newest first.

5 - Fixing the problem and restoring the service

5.1 - View Event Manager to determine the root cause and fix the problem

Narration

This is the event management component of Watson AIOps. Here's the event group that we're working on. There are four events grouped together – these are the same ones we previewed earlier from the Slack notification. There are additional details here, including the likelihood of each event being the root cause of the overall issue and if there is a runbook associated with the event.

There are three events that occurred on the Rating service. They're all rated as 99% likely to be the root cause, meaning that the Rating service is the source of the problem.

It also shows us that there's a runbook associated with these events, which is what was mentioned in the GitLab incident comment.

Action 5.1.1

- Navigate to the **Event Manager** tab (you should already have this open from the demo preparation). Click **Events** on the sidebar menu. The top event should show a red circle (**critical**). Click the **Down** arrow to expand it.

The screenshot shows the IBM Netcool Operations Insight interface. The top navigation bar includes 'IBM Netcool Operations Insight' and 'IBM Cloud Analytics'. Below the navigation is a search bar and a 'Data sources' dropdown set to 'Example_IBM_CloudAnalytics'. The main area has tabs for 'Events' (selected), 'Logs', 'Metrics', and 'Dashboards'. Under 'Events', there are sections for 'Data sources' (with a dropdown for 'Default') and 'Summary'. The summary table lists events from '5:10@Database1e51511aaad...' with columns for 'Severity' (No), 'Ack' (No), 'Probable Cause' (qnd-reting), 'Rumbook' (●), 'Seasonal' (●), 'Topology' (●), 'Incident' (●), 'Node' (●), and 'Summary' (GROUP: (4 event(s): Rating service CPU usage very high). Rating service Mem usage very high, Rating service response time over past minute is significantly high (> 1.0s), Rating service CPU usage very high, Web service response time over past minute is significantly high (> 1.0s)). Below this is a detailed log table with columns: 'Pod status' (Status of pod: Pending 2 Running 147 Succeeded 43) No operator last updated 2023-09-19T10:45:00Z), 'Event count (alerts.status): 119', 'Event count (alerts.state): 119', and 'Event count (alerts.state): 119'. The log table lists entries for various pods, such as 'etcdmanager-recognition-0' and 'etcdmanager-recognition-0' under 'GATEWAY', and 'etcdmanager-recognition-0' under 'GATEWAY', all showing 'false' status.

Narration

Now we'll execute the runbook that should resolve the underlying failure of the Rating service. That should fix the problem.

Action 5.1.2

- Click the second row to select it (**Summary column: Rating service Mem usage very high**). Click **Execute runbook**.

The screenshot shows the IBM Netcool Operations Insight interface. On the left, there's a sidebar with various icons. The main area is titled 'Events' and contains a table with columns: 'Sev', 'Ack', 'Probable Cause', 'Runbook', 'Seasonal', 'Topology', 'Incident', 'Node', and 'Summary'. The 'Summary' column for the second row displays the message 'Rating service Mem usage very high'. A context menu is open over this row, with a red box highlighting the 'Execute runbook' option at the bottom.

Action 5.1.3

- Click **Start runbook**.

This screenshot shows a 'Runbook' dialog box. At the top, it says 'Runbook' and 'Fix Ratings Service Failure'. Below that, there are two sections: 'Step 1' (Automated step, 'Fix Ratings Service Failure') and 'Provide feedback' (with a 5-star rating). On the right, there are tabs for 'Details' and 'Information', and a section for 'Automation'. At the bottom right of the dialog, there is a red box around the 'Start runbook' button.

Action 5.1.4

- Click **Run**.

This screenshot shows the same 'Runbook' dialog box as the previous one, but with a red box highlighting the 'Run' button at the bottom right instead of the 'Start runbook' button.

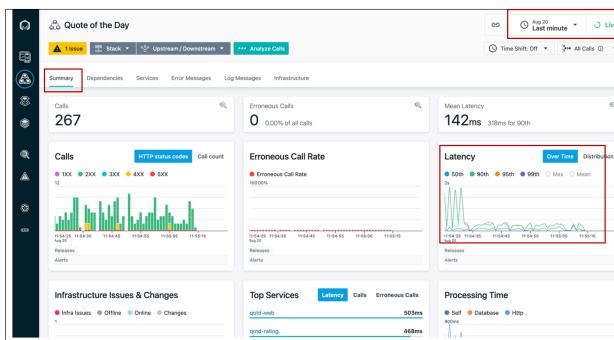
Narration

To check that the runbook is fixing the problem, we can look at the health metrics in Instana. We'll set the time period so that we can see data from when the incident began until now.

We can see that the latency of the application started increasing when the problem began. Now that we've run the runbook, latency is going back down. This confirms that the runbook was successful.

Action 5.1.5

- Navigate to **Instana**. Click the **Summary** tab. Set the time period to **last 15 minutes**, and click the **Live** button (you will be showing the latency before, during, and after the incident).



6 - Closing the incident

6.1 - Provide feedback on the runbook and close the incident

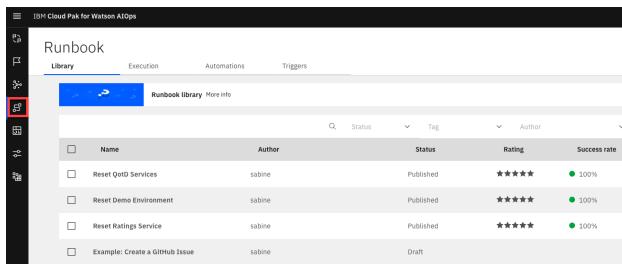
Narration

We can rate the runbook to preserve the information that it was successful. This will be helpful for the next person who comes along to look at this runbook and provide feedback to the author of the runbook.

This enables a collaborative approach to organically improve incident resolution over time. As the system matures, it can evolve into an increasingly automated self-healing system.

Action 6.1.1

- Go back to the **Runbook** screen on the **Event Manager** tab.
- **Note:** If you navigated away from it, go back with these steps:
Automations -> Runbooks -> Click the three dots at the far right for the **Reset Rating Service** runbook -> Choose **View history** -> Click **Resume on most recent**.

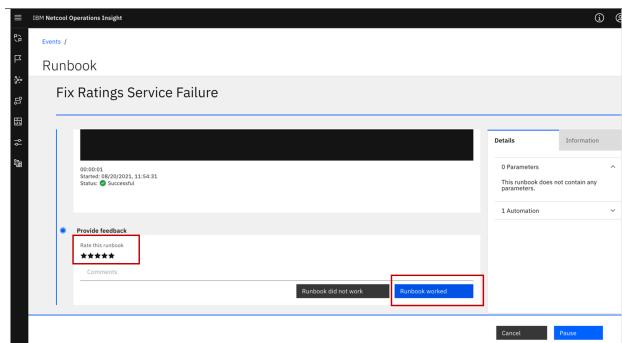


The screenshot shows the 'Runbook' library page. At the top, there are tabs for 'Library', 'Execution', 'Automations', and 'Triggers'. Below the tabs is a search bar and a 'Runbook library' button. A red box highlights the three-dot menu icon on the right side of the screen. The main area is a table with columns: Name, Author, Status, Rating, and Success rate. The table contains five rows:

Name	Author	Status	Rating	Success rate
Reset QoD Services	sabine	Published	★★★★★	100%
Reset Demo Environment	sabine	Published	★★★★★	100%
Reset Ratings Service	sabine	Published	★★★★★	100%
Example: Create a GitHub Issue	sabine	Draft		

Action 6.1.2

- Scroll down to the bottom where there are stars and ratings.



The screenshot shows a runbook details page for 'Fix Ratings Service Failure'. At the top, there are tabs for 'Events /' and 'Runbook'. The main area displays the runbook's title and a summary box. Below the summary is a 'Provide feedback' section with a star rating field (set to 5 stars) and a 'Comments' input field. A red box highlights the 'Provide feedback' button and the star rating field. At the bottom of the page, there are 'Cancel' and 'Save' buttons.

Narration

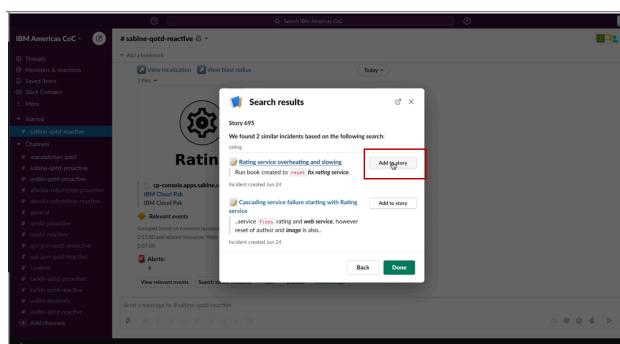
Now we are ready to close the incident.

We found the runbook because it was mentioned in the similar incident, so we want to preserve that information. To do so, we can click ‘Add to story.’ Watson AIOps adds a comment with the information about the similar incident.

This keeps all the information in one place so it’s easy to find later or if another similar incident occurs in the future.

Action 6.1.3

- Navigate to **Slack**. Add details about the resolution by clicking the **Add to story** button.

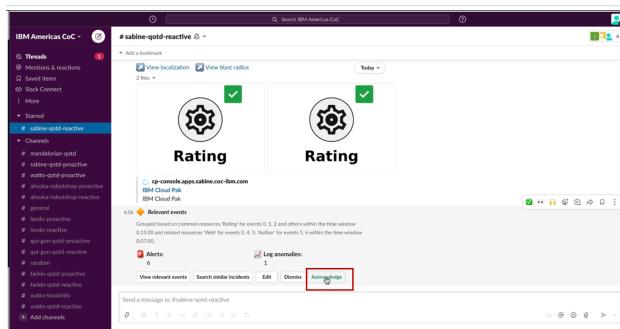


Narration

Now we will acknowledge the incident and mark it as closed. Watson AIOps archives all the information from the incident in the comments of the original message, so we can go back to it whenever needed to see exactly what happened and how it was fixed.

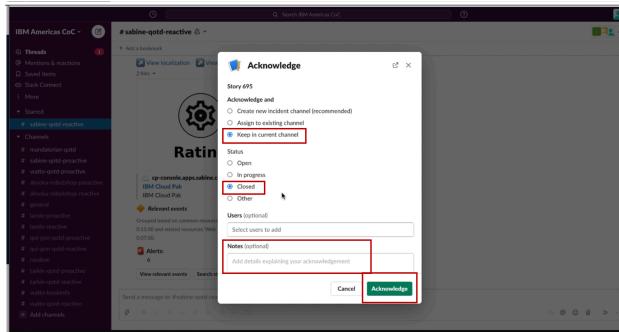
Action 6.1.4

- At the bottom of the story notification, click **Acknowledge**.



Action 6.1.5

- Choose **Keep in current channel** and **Closed**. Optionally add a comment, such as '**Executed runbook to fix issue**', and then click **Acknowledge**.



Summary

In this demo, we have demonstrated how Watson AIOps enables you to avoid business-impacting outages by applying AI to data gathered from your existing disparate tools. It helps you quickly determine the root causes of a failure and proactively alleviates outages, therefore minimizing the business impact of these IT issues on revenue or client experiences.

The anomaly detection capabilities alert you early to potential issues, enabling the SREs to quickly take remedial actions. The intelligent event analytics examine logs, metrics, tickets, and topology and provide a useful correlation that would otherwise be very challenging. Lastly, the AI-driven remediation, based on recommendations of previous incidents, accelerates problem resolution and significantly reduces the mean-time-to-repair.