Lab Guide

Advanced Event Management Features:

Correlation, Binding, and Impact

Benny Makovsky & Joe Steinfeld

Default Login / Password: Knowledge17

admin / Knowledge17

itil / Knowledge17

employee / Knowledge17

Lab Sponsored by Amazon Web Services



This
Page
Intentionally
Left
Blank

Lab Goal

Lab 1.0
Event Rules Designer

This lab explains how to use the new Event Rules Designer. Event rules allow transforming raw events into meaningful alerts by mapping the event fields into the alert fields and by setting additional conditions that control how and when events become alerts. Event Rules provide the following key capabilities:

1. Event filter: Defines the conditions in which the event rule applies to incoming events.
2. Alert field mapping and transform: Maps event fields into alerts fields and transforms event values using regular expressions.
3. Threshold: Defines a threshold condition in which the event rule applies to incoming events.
4. Binding: Defines the logic to bind an incoming event to CI in ServiceNow CMDB based on event values.

This lab focuses on event rule designer, which is a new UI-based tool that enables easier and more flexible ways to define event rules.

Event Rules Designer

The Event Rules Designer enables manual control of how events become alerts and establishes the logic and conditions for setting them. Use Event Rules Designer to define event rules using incoming events that match the filter. This lab explains how to build event rules that transform a raw incoming event into a meaningful alert using the event rule designer, which transforms the following events:

- **Simple** event: a standard event based on Event Management API format using JSON.
- **Free Format** event: a custom event in JSON format that is not aligned to EM API format.
- [Optional] **Threshold** event: a custom event transformation with the addition of threshold condition.
- [Optional] **Binding** event: a custom event transformation with the addition of CI binding logic.

The same custom event is used for multiple scenarios. You use the POSTMAN application (a Chrome extension) to simulate events. Make sure the target instance of events is the same instance you were assigned to.

Send Event Using Event Management API Format

Follow these steps to send a standard event based on Event Management API format using JSON:

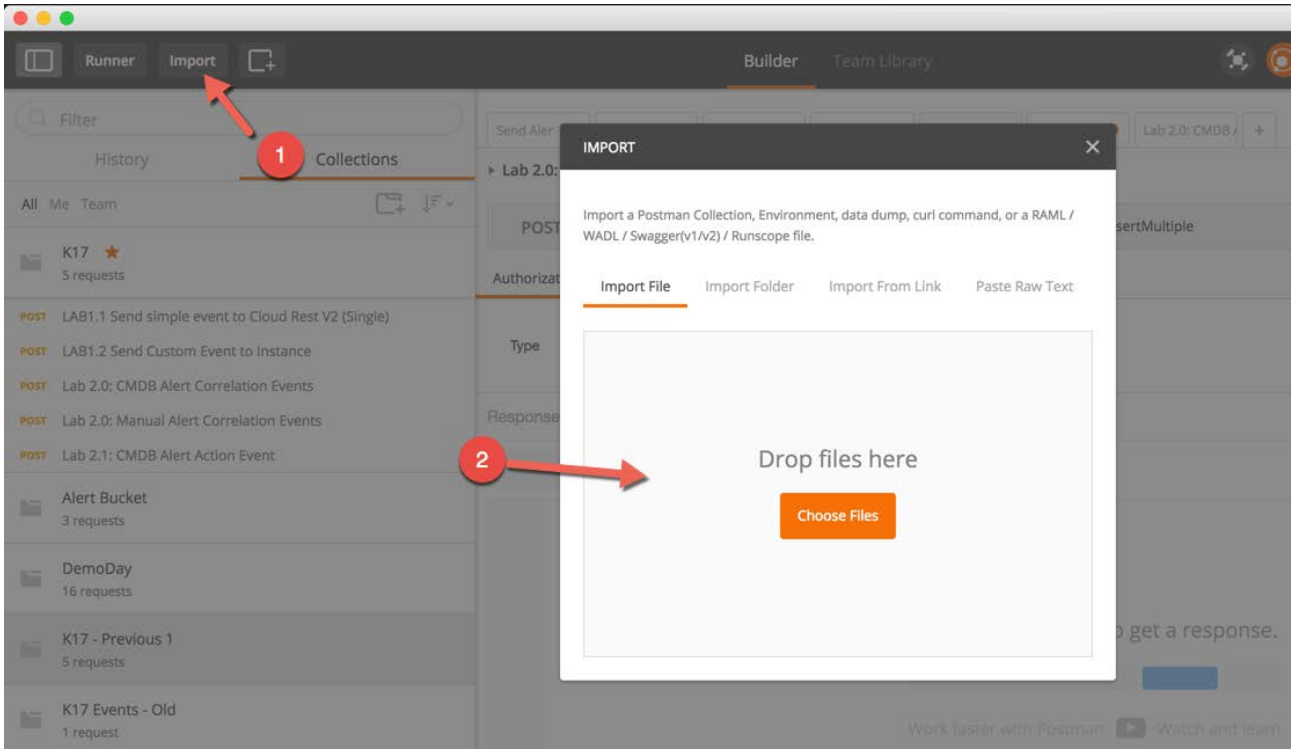
1. First off, we need to install the Postman by going to the following URL: https://www.getpostman.com/ in your Chrome Browser and clicking on "Chrome". This will kickoff install of Chrome App inside Chrome. Just take defaults

   (NOTE: If you don't have chrome you can install Postman on your OS by clicking on the appropriate OS link and taking the defaults)

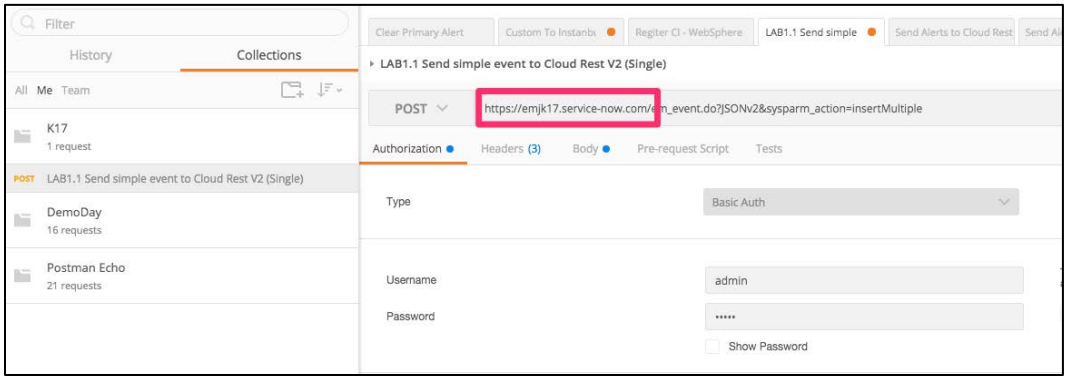
2. Import Postman collection by doing the following:

   - Download Postman collection from following URL: https://servicenow.app.box.com/s/7nht1s07zk0tmmp5b7z7dqzre30plyht/folder/20891838665 <Need to update URL with final URL for downloading collection>

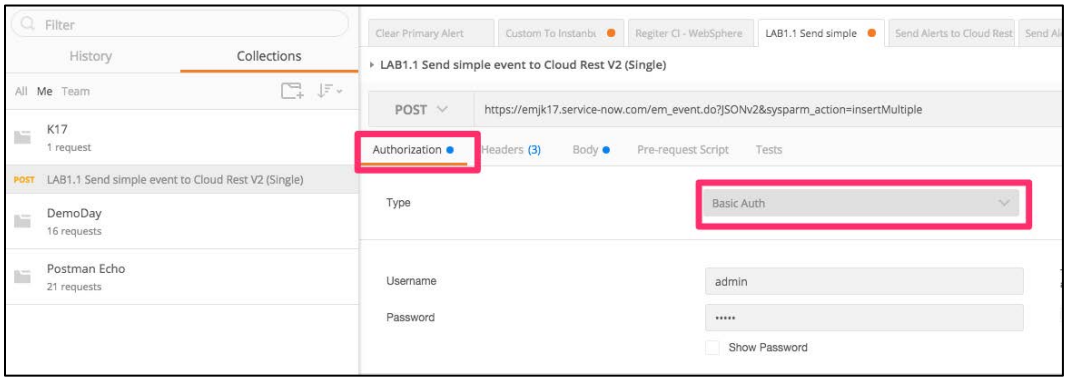
   - Open Postman and in the upper right click on **Import** and then either select or drag and drop the Postman collection you downloaded above



3. Open **Postman** and select **LAB1.1 Send simple event to Cloud Rest V2 (Single).** Make sure the **Post** is targeted to your instance.

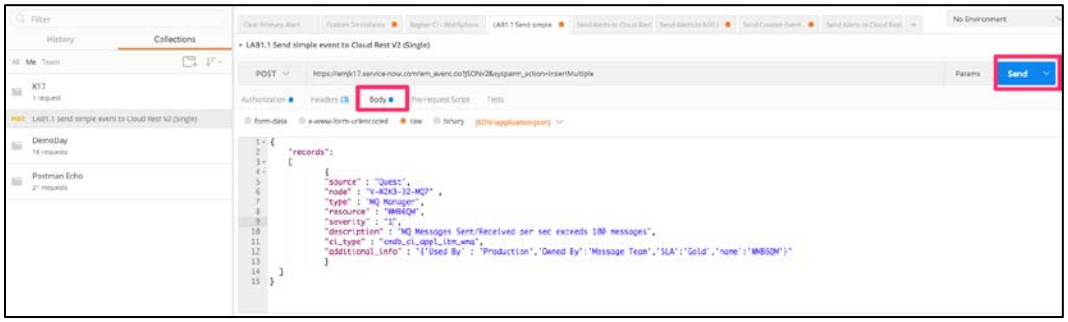


4. Make sure the authorization type is **Basic Auth** and user name and password are **admin/Knowledge17**.

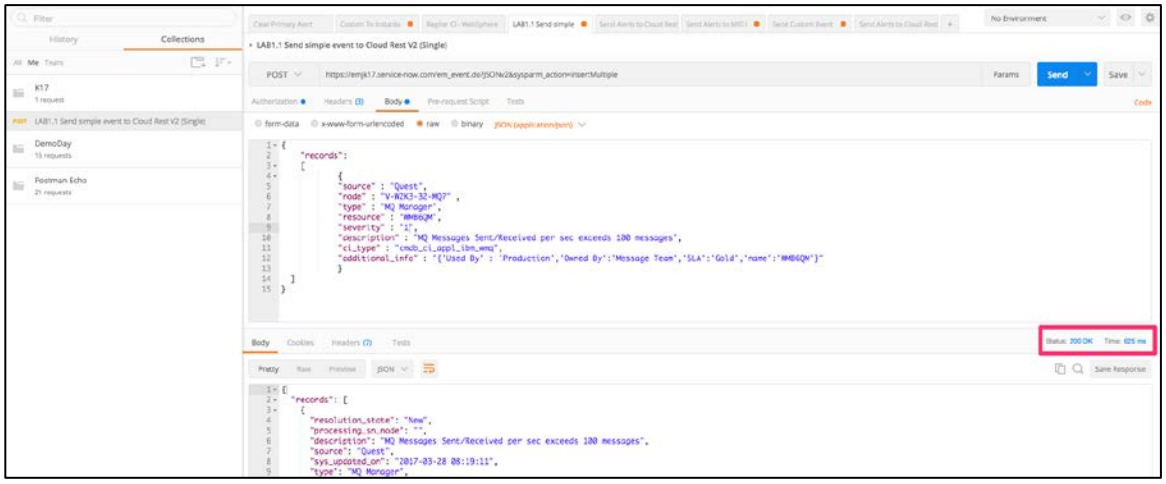


5. After you update the URL and Authentication, click **SAVE** in the upper right.

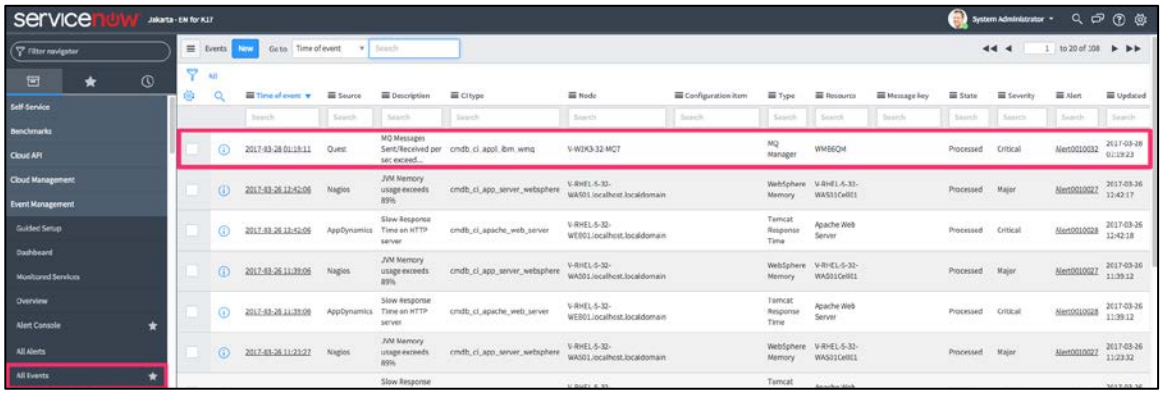
6. Click **Body** to identify the event and click **Send**.



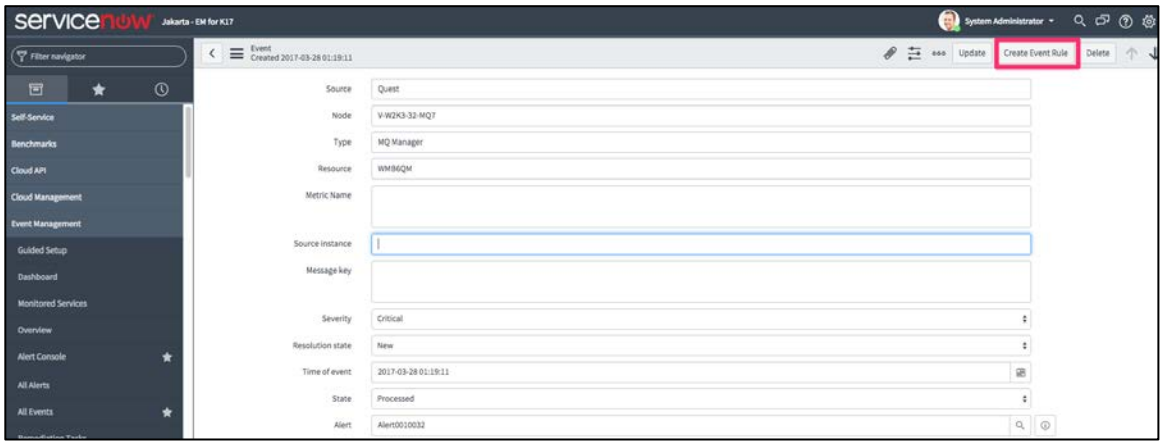
7. Make sure you get success response form the server **200**.



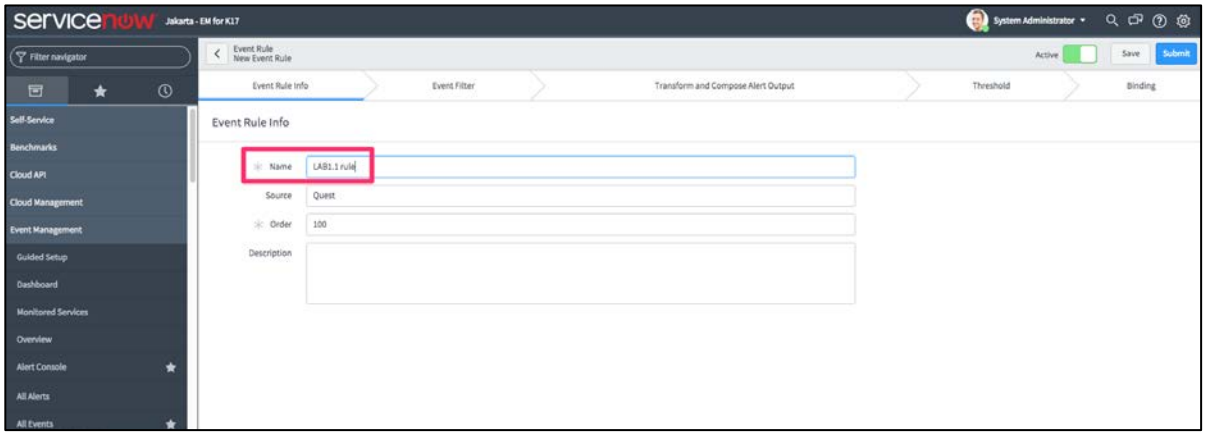
8. Navigate to **Event Management > All Events** and identify the event in the event table. You may meed to click on **Time of Event** to sort events so most recent is on top.



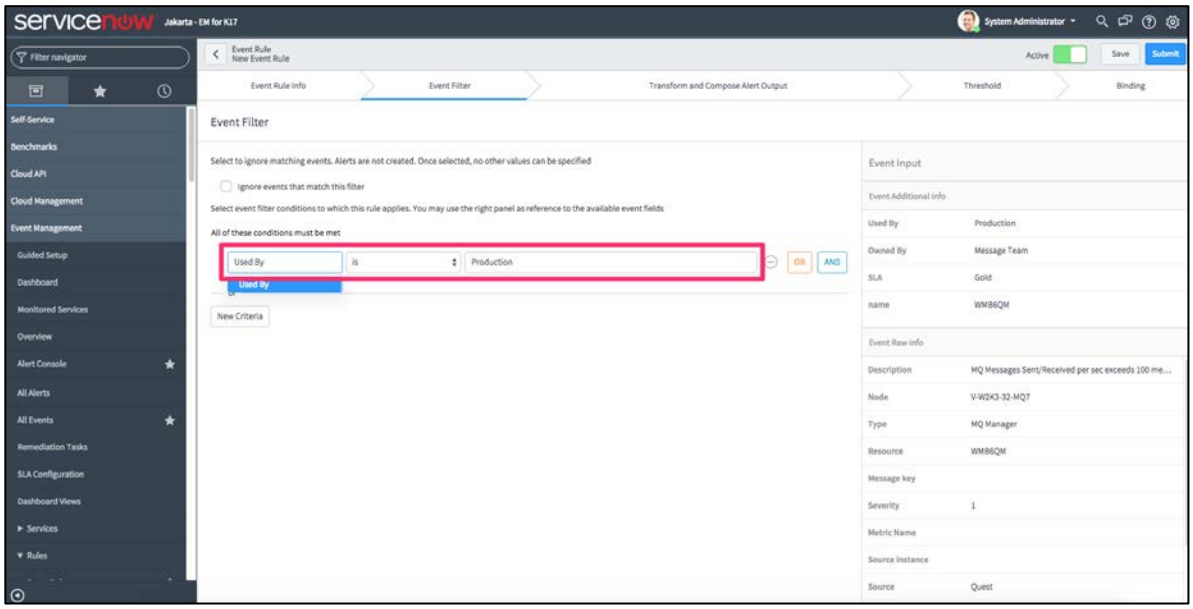
9. Click the event **Time of Event** and drill down to the event form. Then click **Create Event Rule** and wait for the designer to load.



10. The new event rule designer opens. In the Event Rule Info provide a **Name**. Click **Event Filter**.

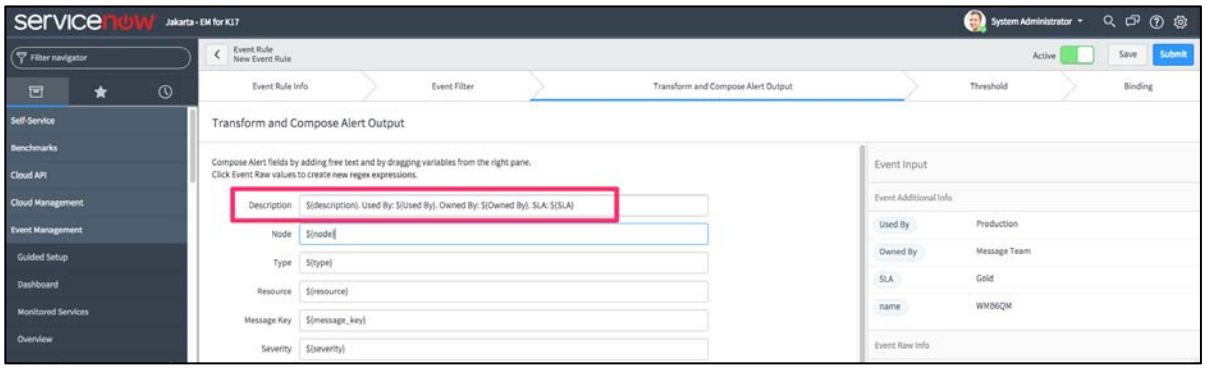


11. Enter the following filter conditions: **Used by is Production** using the drop-down selection. Click **Transform and Compose Alert Output**.

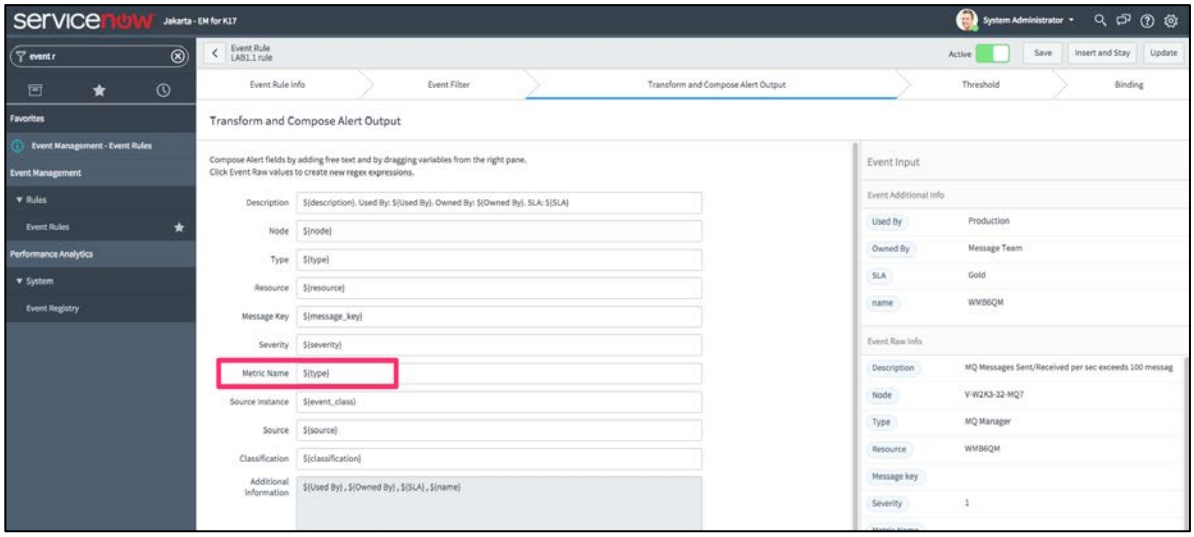


12. In the description field, complete the following values by dragging and dropping the fields from the right **Event Input** panel and by typing the relevant text.

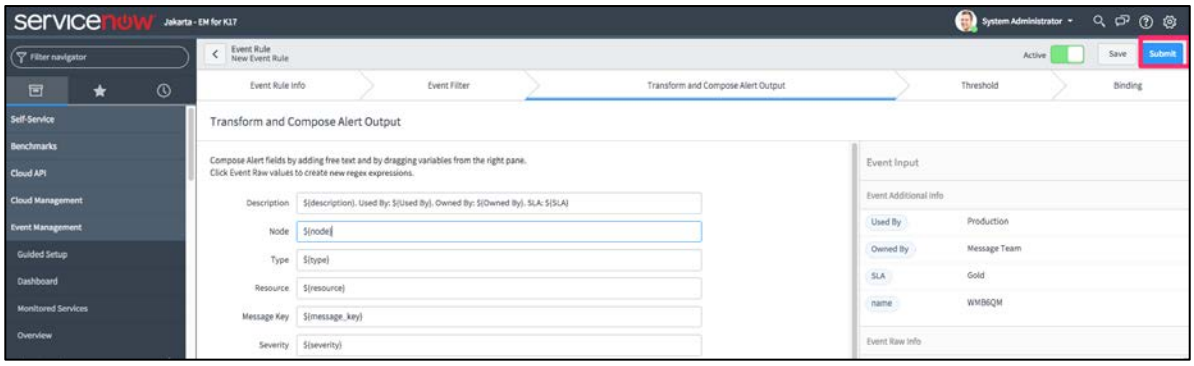
    Description: **${description}. Used By: ${Used By}. Owned By: ${Owned By}. SLA: ${SLA}**



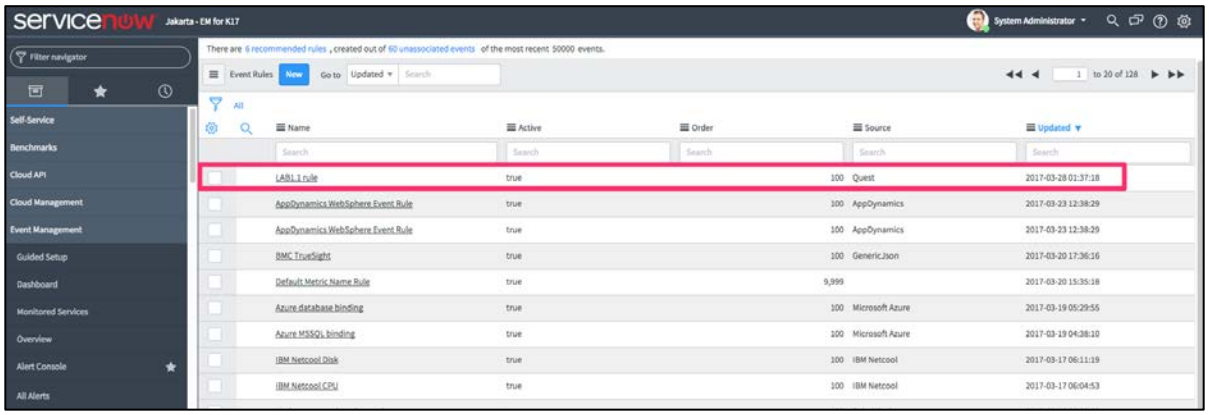
13. Update the Metric Name with **${type}**.



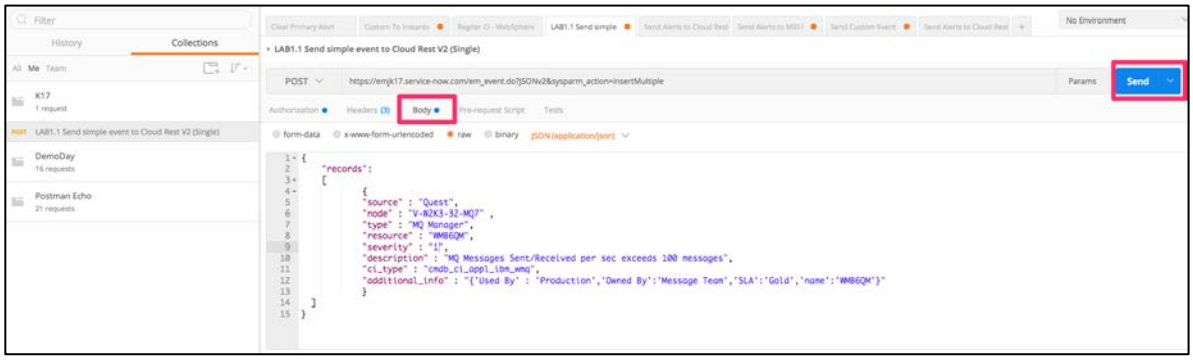
14. After completion, click **Submit** and wait for operation to be completed.



15. After completion, the screen returns to the rule list. You may need to click on **Updated** column to see most recent rule on top or you can always search for it.

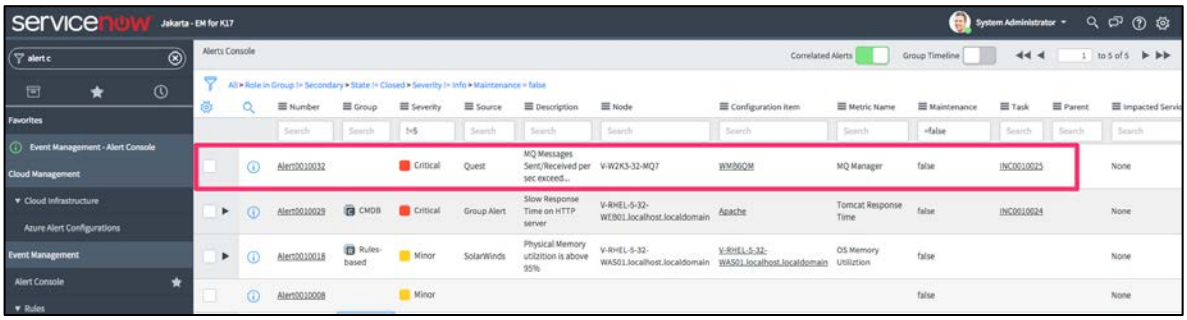


16. Now regenerate the event and see the event rule in action. Open **Postman** and select **LAB1.1 Send simple event to Cloud Rest V2 (Single)** and click **Send.** This sends the same event to the instance but now the event is transformed by the event rule just created.

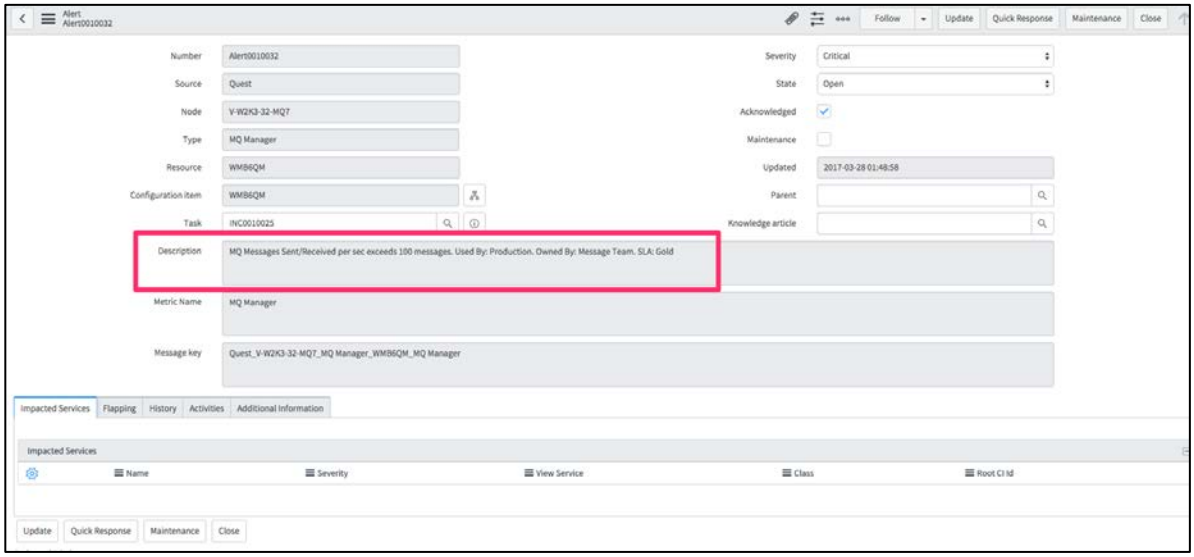


17. Navigate to **Event Management > Alert Console** and add **Updated** column to the list view if it is not already there. Sort by **Updated** so that latest Alerts are at the top of the page.

18. You should have a critical alert at the top of the page from Quest for **MQ Messages Sent/Received per sec exceed…**.



19. Open the alert by clicking on its number and look for the description field. Notice the new values were populated as defined in the event rule.



Send Free Format Event

Follow the steps to send a custom “free format” event based on JSON format that is not aligned to Event Management API format. This lab demonstrates how nonstandard events are not transformed by default and explains how to use event rule designer to easily transform these events into meaningful alerts.

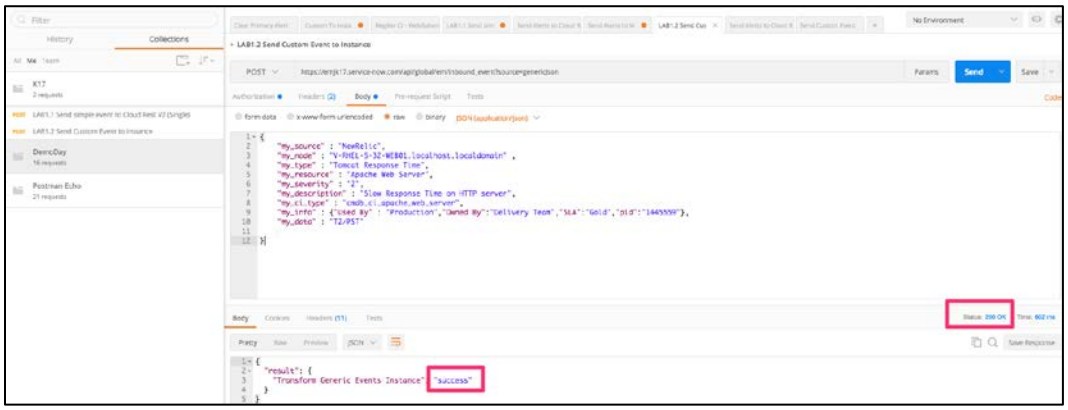
1. First, send a custom event using the Postman application. Open **Postman** and select **LAB1.2 Send Custom event to Instance.** Make sure the POST is targeted to your instance. Notice the target URL is different from previous LAB and targets a different URL. (NOTE: You will need to update the URL and authentication for you instance and don't forget to save it.)



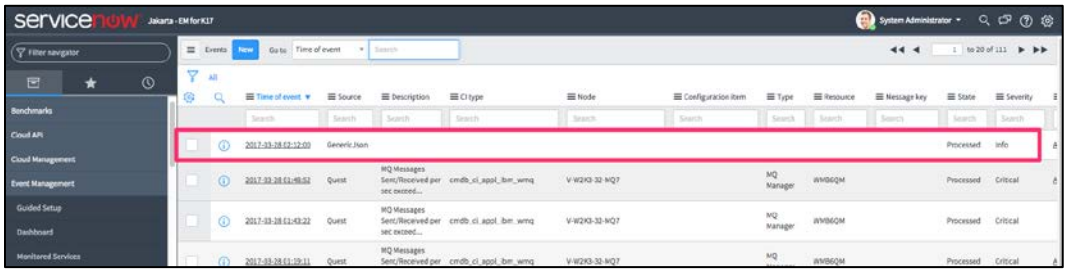
2. Also notice that the event body includes custom only fields (**my_field**), which means these are not standard Event management API fields.



3. Once verified, click **Send** and confirm a successful return code **200**.

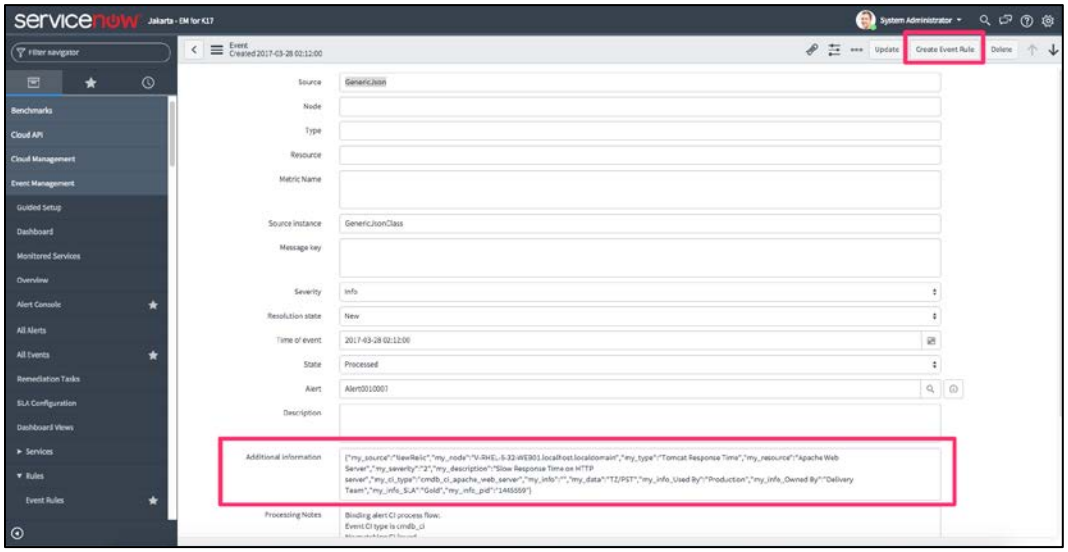


4. Navigate to **Event Management > All Events** and identify the event in the event table.

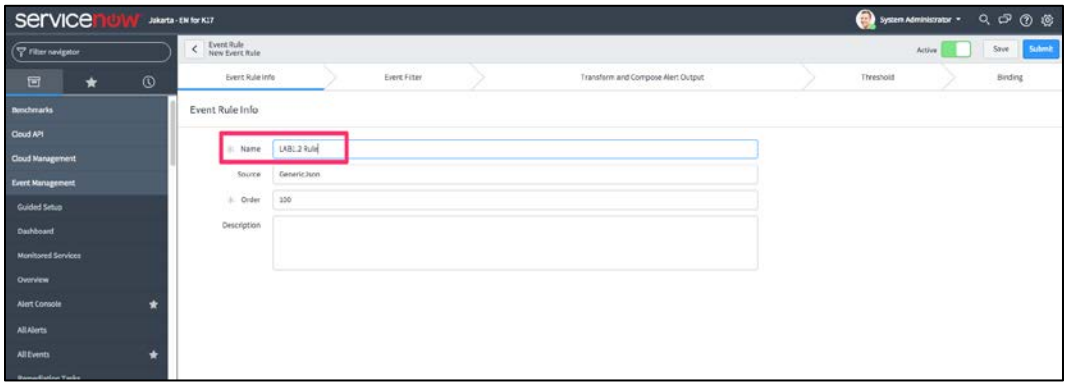


Notice that none of the fields are populated.

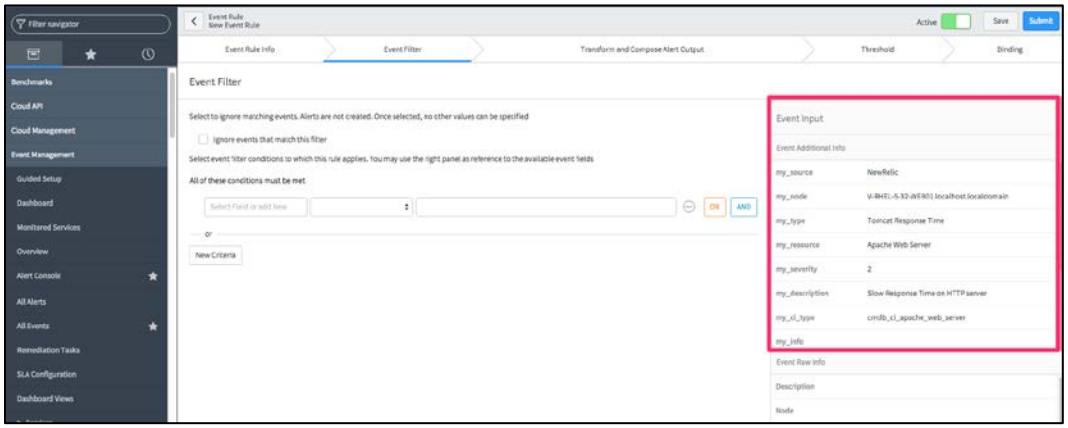
5. Click the event **Time of Event** and drill down to the event form. Notice that all the fields are grouped under the **additional information** field. After verifying, click **Create Event Rule** and wait for the designer to load.



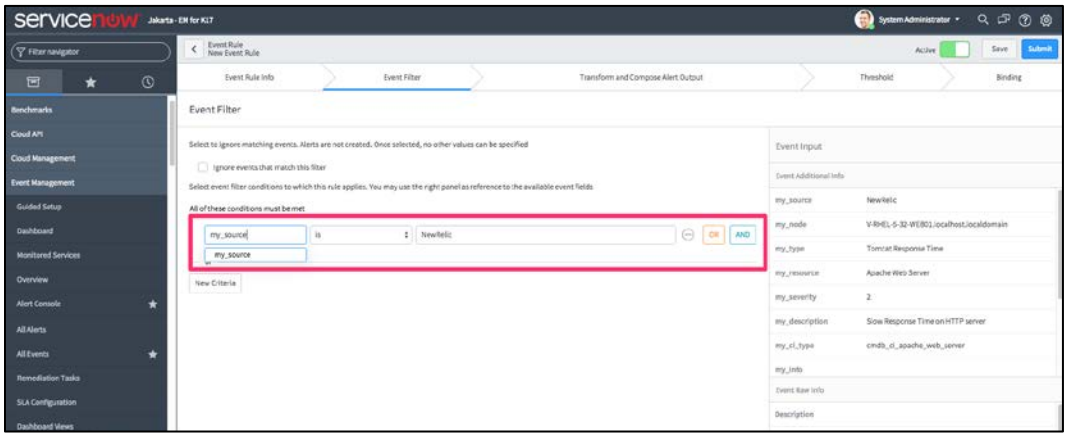
6. The new event rule designer opens. In the Event Rule Info provide a **Name**. Click **Event Filter**.



7. Notice that all the event fields are automatically parsed into the **Event Input** right panel and have separate values that can be used for event filter and alert composition.

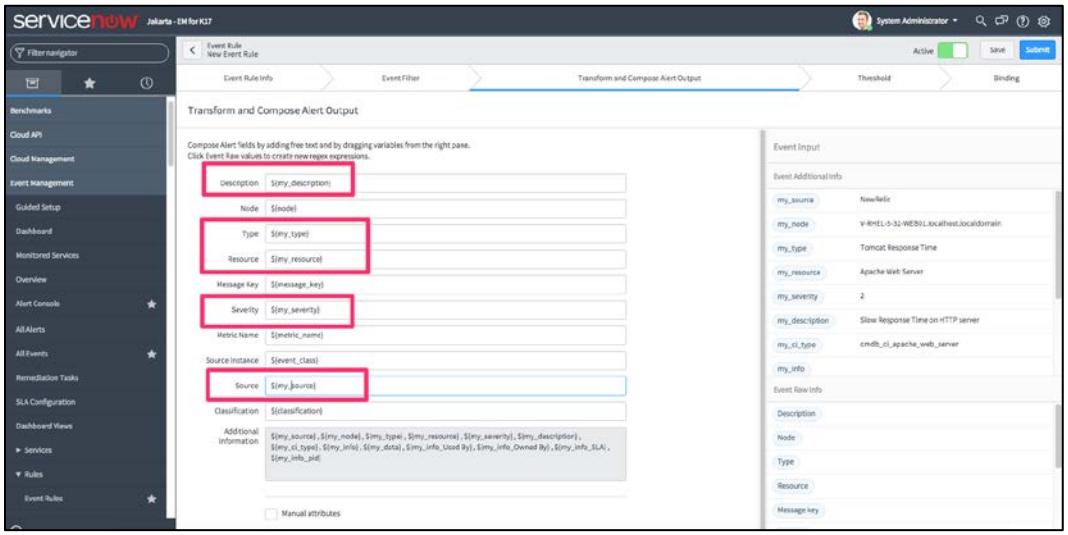


8. First, define a generic filter to capture similar events by **my_source** field. Fill in the following filter conditions: **my_source is NewRelic** using the drop-down selection. Click **Transform and Compose Alert Output**.



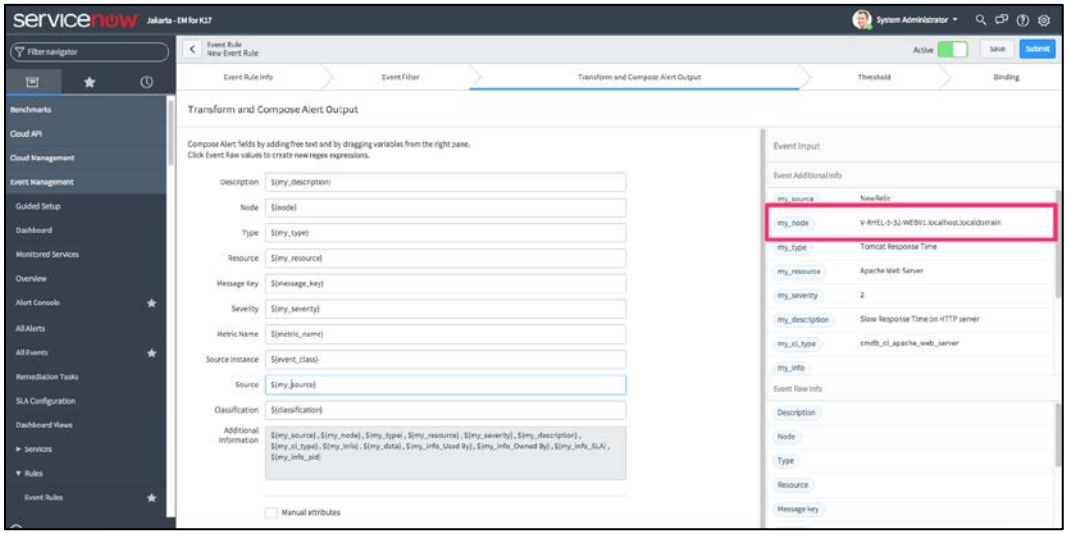
9. Define the following transformation logic on the event to map into alert fields. Drag each of the fields form the right panel into the main panel so eventually all the alert fields are mapped to the new fields as follows:

   Description: **${my_description}**
   Type: **${my_type}**
   Resource: **${my_resource}**
   Severity: **${my_severity}**
   Source: **${my_source}**

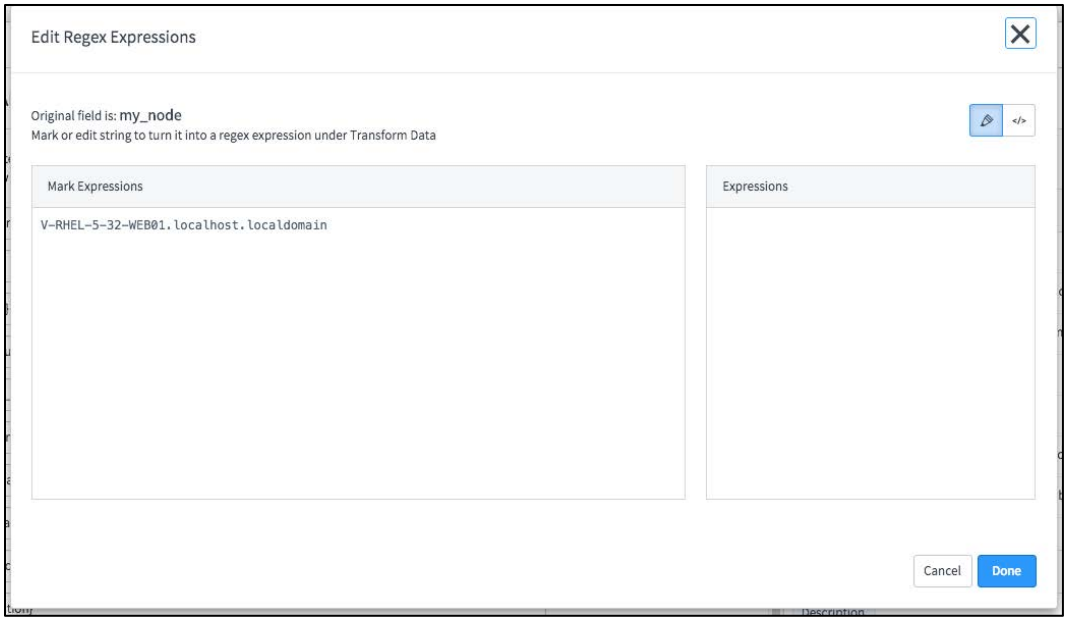


   Notice that some fields were skipped (e.g., **Node** and **metric name**). You will address them later.

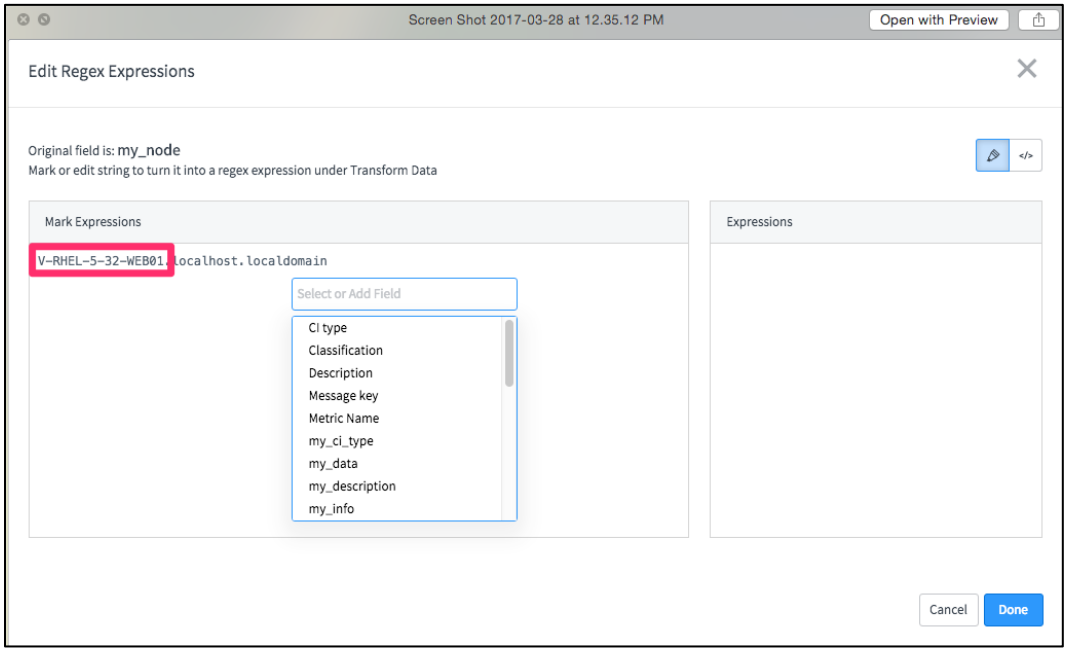
10. Now transform the node field from FQDN into short name. Click the **my_node** field in the right panel.



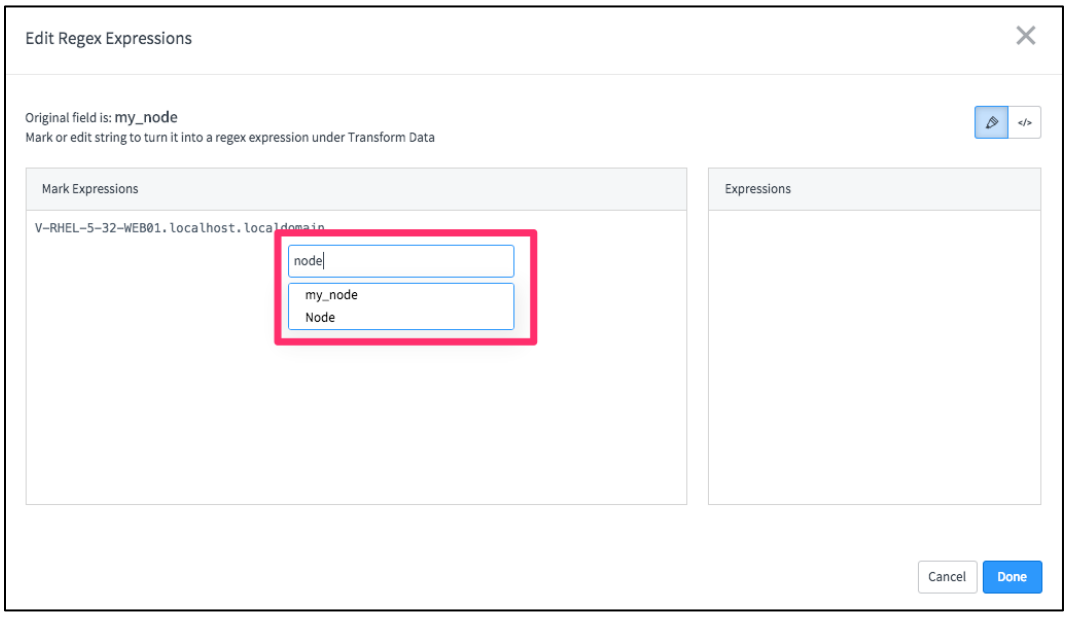
11. The **Edit RegEx Expressions** window appears.



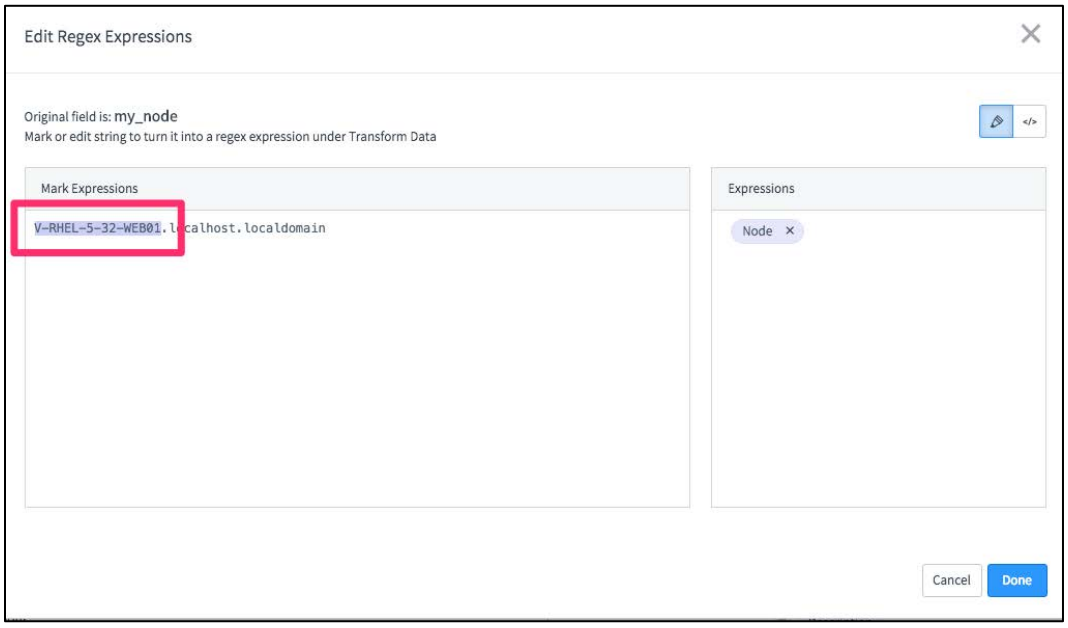
12. Using the mouse, mark the prefix of the node name (excluding the node suffix). After completion, the following popup menu opens.



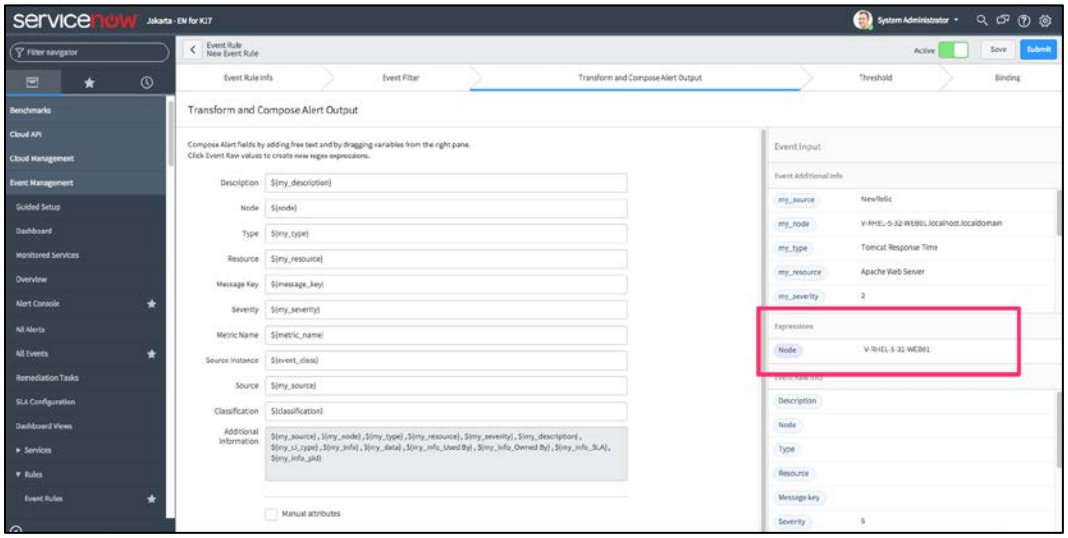
13. In the search box, type **Node** and select the **Node** from the drop-down menu.



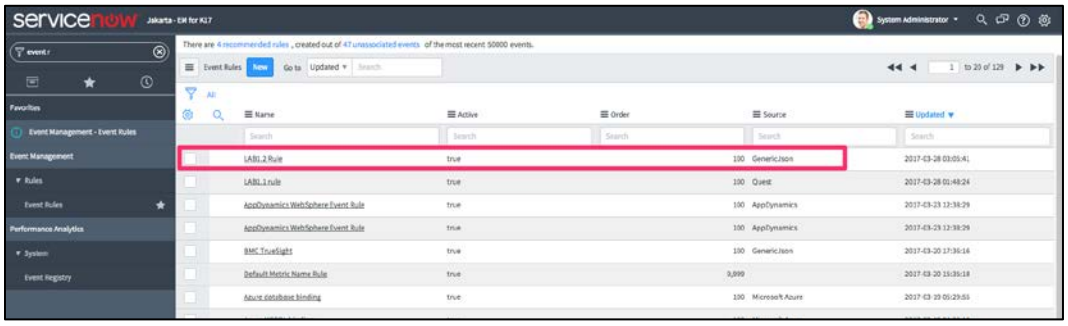
14. Notice the new field is populated in the expression right panel, and the relevant text is marked.



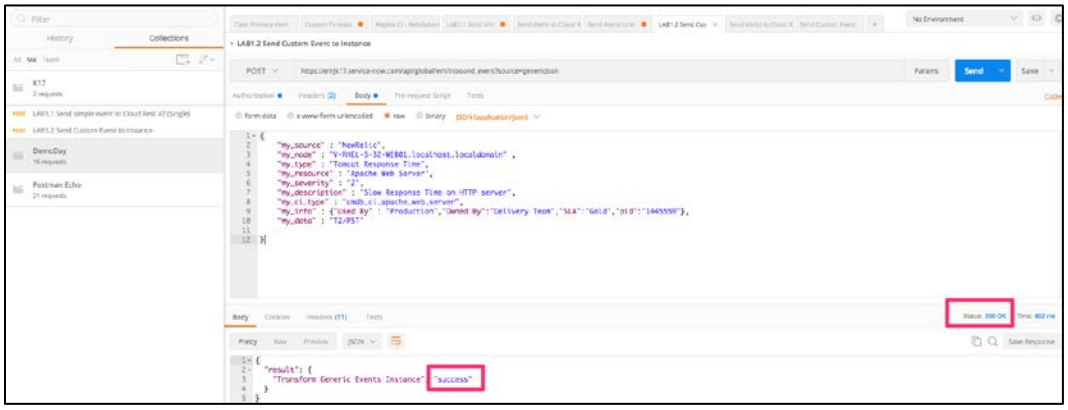
15. Click **Done** and notice the new expression appears in the event rule right panel.



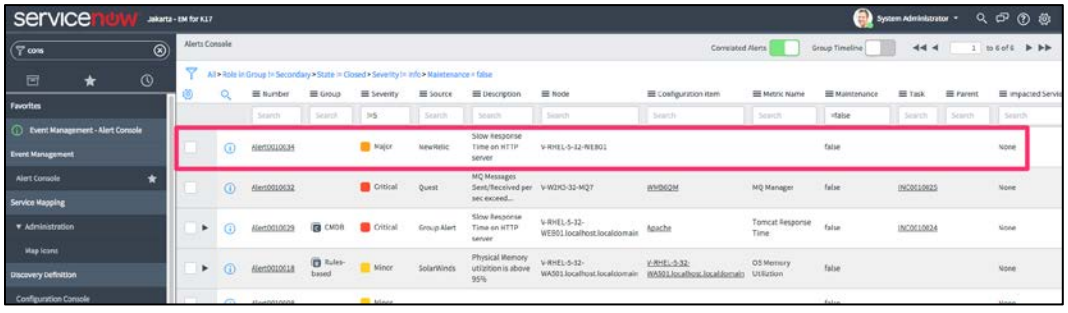
16. Once verified, click **Submit** to save the event rule and to be redirected to the event rule list.



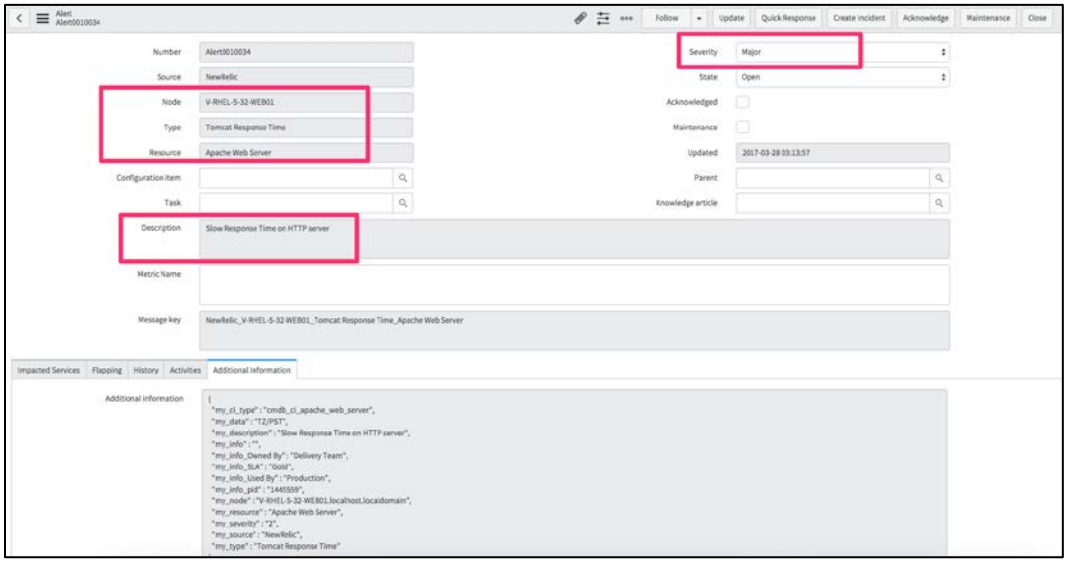
17. Now, resend the event and see the transformation in action. Open **Postman** and select **LAB1.2 Send Custom event to Instance.** Then select **Send**, verify a successful return code **200**.



18. Navigate to **Event Management > Alert Console** and add **Updated** column to the list view if it is not already there. Sort by **Updated** so that latest Alerts are at the top of the page.



19. Drill down to the alert and verify the fields were mapped correctly. [optionally other additional info fields can be mapped to compose more meaningful alert]



1. [Optional] **Threshold Event**: a custom event transformation with the addition of threshold condition. If you finish the lab before the given time you may proceed and try to create your own event rule that uses threshed logic.
2. [Optional] **Binding Event**: a custom event transformation with the addition of CI binding logic. If you finish the lab before the given time you may proceed and try to create your own event rule that uses binding logic.

Lab Goal

Lab 2.0 CMDB Alert Groups

This lab explains how to set up and use Correlation Rules. Correlation rules allow alerts that are related to be logically grouped together reduce the alarm noise in the environment and concentrate on what is most important. Alert Correlation Rules come in three types:

- Manual Alert Correlation Rules: These rules manually group like alerts together.
- Service Analytics - CMDB Alert Groups: These rules are automatically derived by relationships in the CMDB and group alerts based on these relationships.
- Service Analytics – Temporal Analysis: These rules are automatically derived by using machine learning techniques where the system learns over time what alerts are related to each other to provide predictive outage avoidance.

Service Analytics automatically groups alerts by using different methods of correlation. For CIs without historical data, Service Analytics correlates alerts based on CI relationships in the CMDB. CMDB alert groups are displayed in the alert console and in the Event Management dashboard.

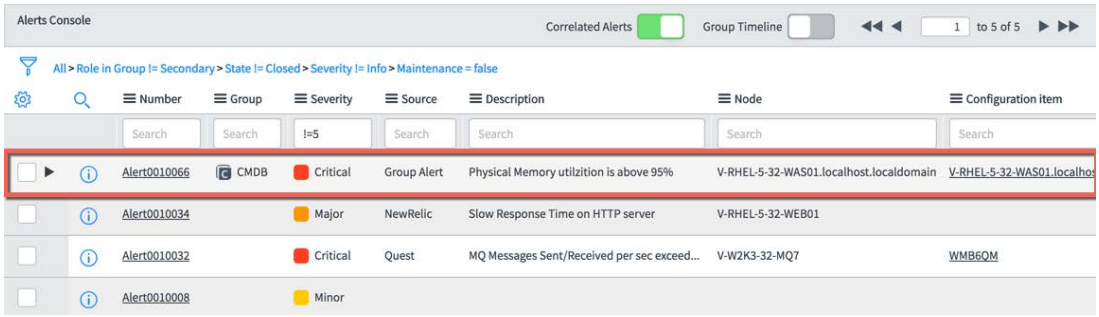
This lab simulates WebSphere Response Time issue event coming from AppDynamics and a OS Memory utilization issue coming from SolarWinds. Since WebSphere runs on the Linux host the CMDB relationship (Runs on::Runs) is used to automatically group these two alerts together.

Add Events and See Results

Follow these steps to introduce CMDB-related events into the system and see the results in the Alert Console.

1. Open Postman and click **Lab 2.0: CMDB Alert Correlation Events**. Click **Send**.

2. Return to the **Alert Console** to see the alerts being created from the events that were just sent. You may notice that the alerts have not yet been correlated. Give it a moment, and you should see the events automatically grouped using CMDB Grouping.



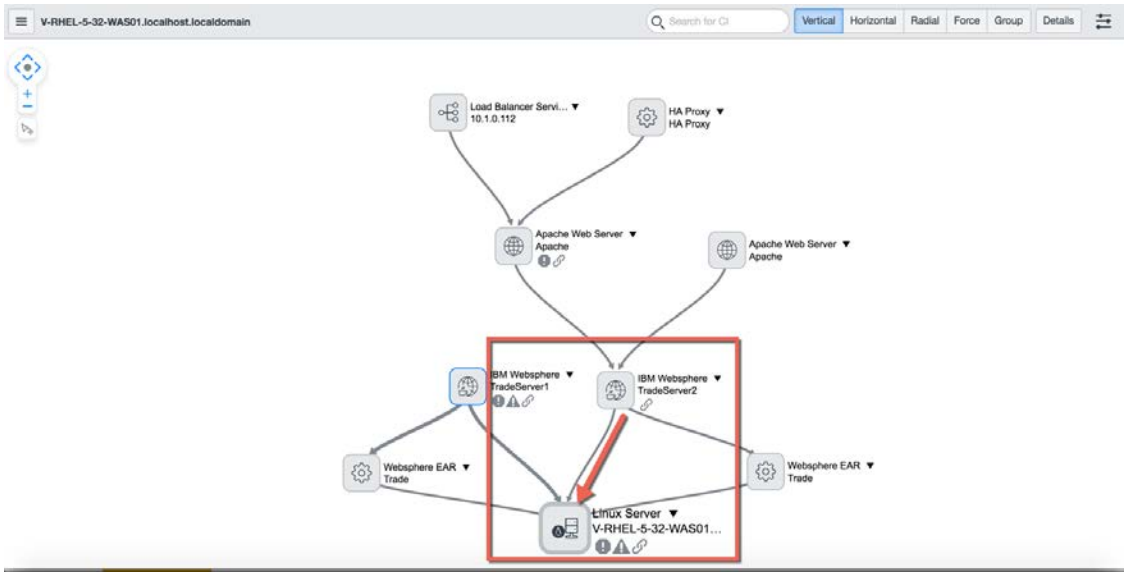
3. Click the **Primary Alert** and investigate the details there. **Scroll** to the bottom and notice that there are two alerts at correlated. What happens for CMDB correlation is that once 2 alerts have been correlated based on relationship a new virtual event is created and the two received events are set as secondary to this alert. CMDB correlation uses the most critical alert as basis for the new virtual alert.

4. **Scroll** to the bottom to see the **Impacted Services** and the Alerts that have been grouped into this primary alert.

5. Scroll to the top and click the **dependency map icon** to the right of the **Configuration Item** dialog box.



6. Notice that **TradeServer2 WebSphere Server** has a dependency on **V-RHEL-5-32-WAS01 Linux Server.** It is this relationship within the CMDB that was used to automatically correlate the alerts that happened between these two systems and group them automatically. In this case, it was Service Mapping that detected these relationships and populated the CMDB. It is this automatic correlation based on CMDB relationships that can help further reduce the noise in our monitored environment to get to root cause much faster.



Lab Goal

Lab 2.1 Manual Alert Correlation Rules

This lab explains how to set up and use Manual Alert Correlation Rules. Manual rules are used to allow companies to manually create correlation rules between components either because system does not allow for automatic correlation using CMDB relationships or Machine learning Temporal techniques or because user wants to override this automatic grouping mechanisms.

Manual Alert Correlation Rule

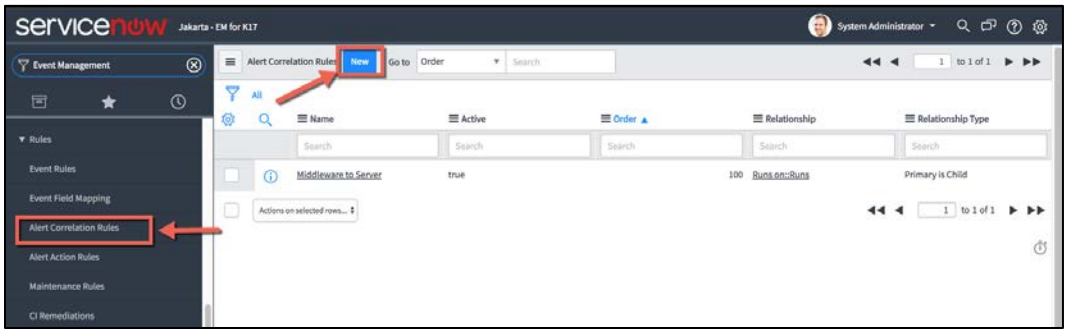
Alert correlation rules enable you to group alerts that are related. This lab explains how to build a correlation rule that groups two alerts together in a primary/secondary relationship. In this relationship, the **Primary** Alert identifies the probable cause of the issue; the **Secondary** alert is a causal alert that is caused or is related but is less important than the **Primary**. This lab explains how to build a rule that relates the following Alerts together:

- Primary Alert: Nagios sends an event that WebSphere JVM Memory Usage exceeds 89% utilization.
- Secondary Alert: AppDynamics sends an event that Apache Response Time is taking longer than 30 seconds.

While it may seem that the response time is a more critical alert, it is instead a symptom of the fact that memory usage on WebSphere has exceeded 89%, which is the issue that needs to be worked.

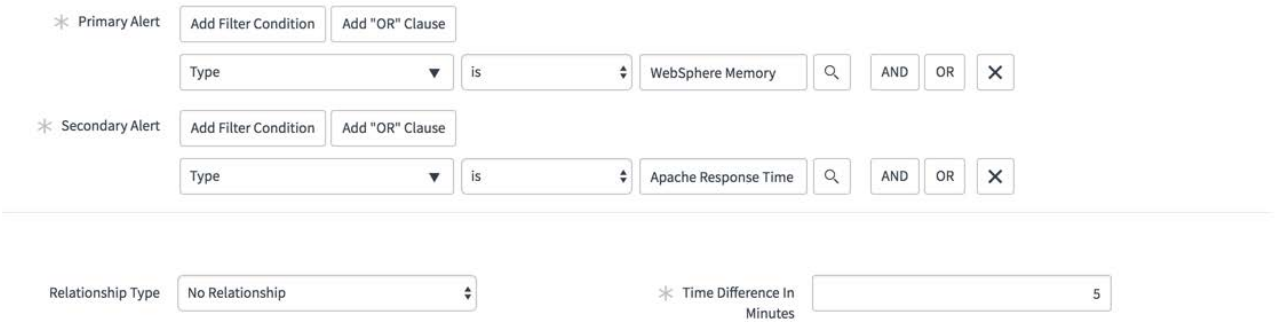
Follow the steps to build the correlation rule:

1. Navigate to **Event Management > Rules > Alert Correlation Rules** and click **New**.



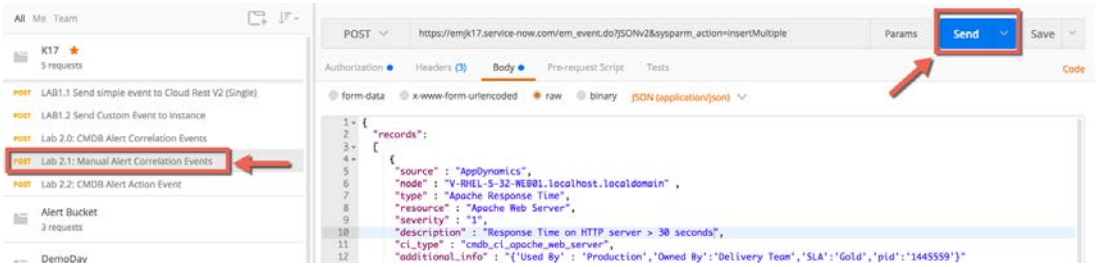
Fill in the following information in the Alert Correlation Rule:

Name: **Apache to WebSphere Rule**
Order: **100**
Active: **Checked**
Primary Alert Filter: **Type is "WebSphere Memory"**
Secondary Alert Filter: **Type is "Apache Response Time"**
Relationship Type: **No Relationship**
Time difference in minutes: **5**



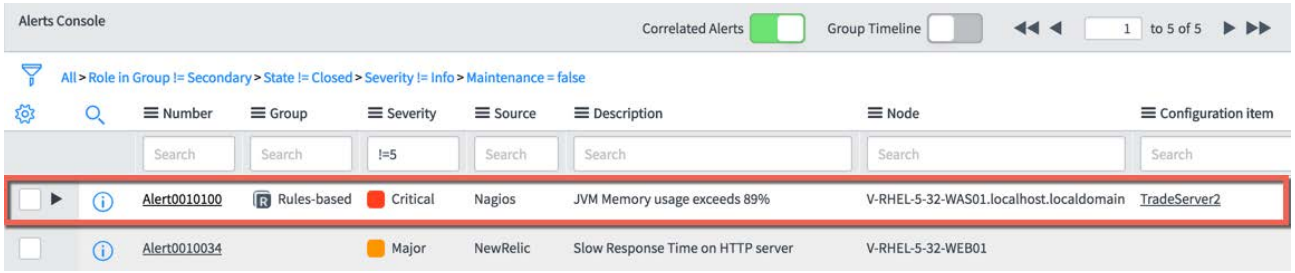
2. Click **Submit** to save the rule.

3. Now, generate some events and see the correlation rule in action. Open **Postman** and select **Lab 2.1: Manual Alert Correlation Events**. Click **Send.** This sends two alerts via Rest call simulating an Apache Response Time event from AppDynamics and a WebSphere Memory Usage Event from Nagios. (NOTE: You will need to update the URL and Authentication and save it)



4. Navigate to **Event Management > Alert Console** and add **Updated** column to the list view if it is not already there. Sort by **Updated** so that latest Alerts are at the top of the page.

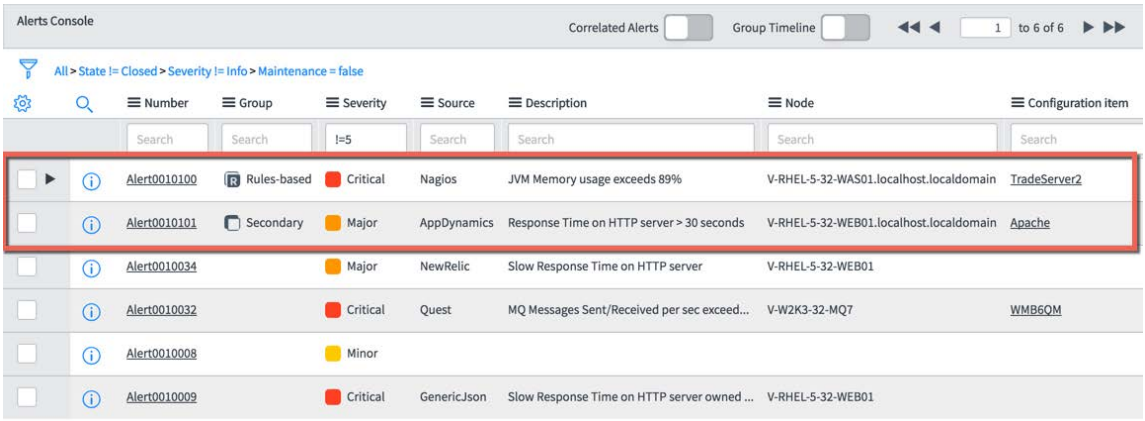
5. You should have a **Critical** alert at the top of the page from Nagios for JVM Memory usage exceeds 89%.



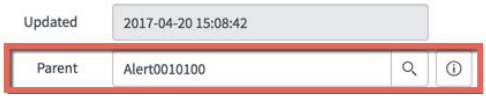
Notice that in the **Group** column there is a 

entry for this Alert. This signifies that an Alert Correlation Rule has matched and that alerts have been correlated and logically grouped.

6. At the top of the Alert Console there is a **Correlated Alerts slider**. When this is turned on, all alerts are grouped together. Turning the slider off ungroups alerts and displays them in the console. Turn off the **Correlated Alerts slider** and notice the change.



7. Look at the details of how these Alerts are linked. Click the **Secondary alert** from AppDynamics. Notice that since this alert is a Secondary Alert that the Parent field is populated with the Primary Alert Number.



8. Further investigate the Alert Form for the Secondary and Primary Alert and see how you could use this information to further prioritize this alert to your organization. (Hint: Investigate the **Impacted Services** tab.)

Lab Goal

Lab 2.2 Alert Action Rules

This lab explains how to setup and use Alert Action Rules (formerly Alert Rules). Alert Action Rules define what action should be performed when certain alerts arrive. These actions can include:

- Automatically open any task record in the system such as an Incident ticket and pre-fill fields based on template
- Run a remediation: A Remediation is an Orchestration workflow that can perform any automation such as collect logs and attach them to the ticket automatically or take auto-corrective actions such as restarting a program that has stopped or clear out filesystem that has filled up. Remediations can either to be automatically executed or run manually by an operator
- Launch an Application: Allows for a web-based application or console to be opened automatically. The URL parameters can be replaced dynamically with fields from the Alert.

This lab explains how to set up an Alert action rule to automatically open an incident record and assign it to the correct group for action.

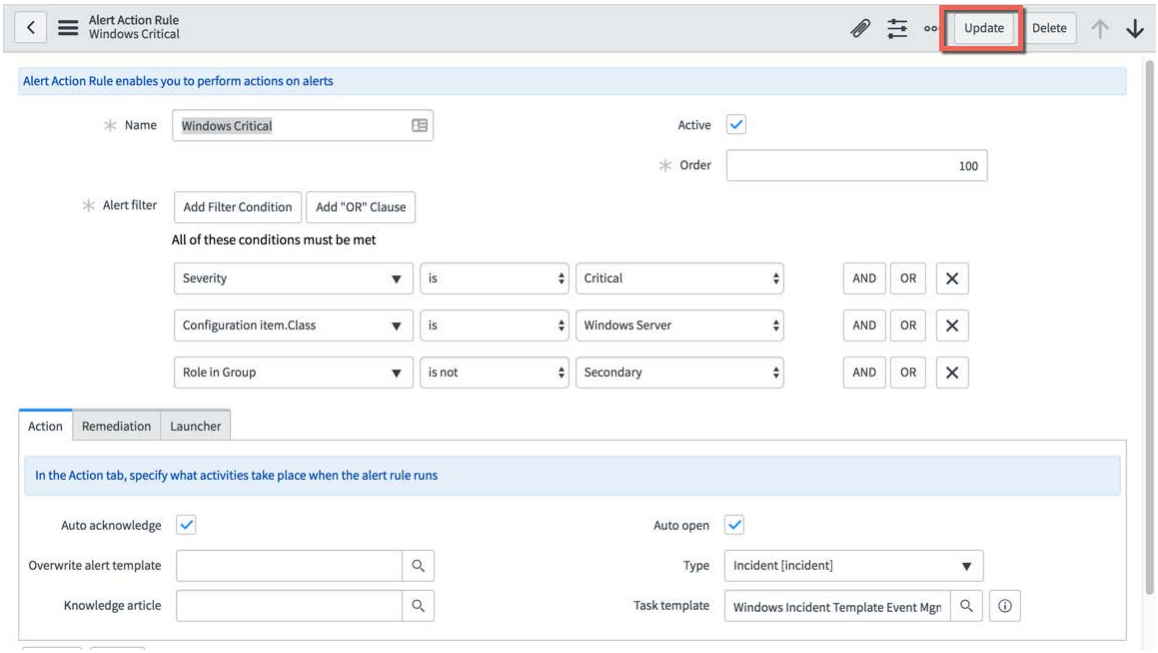
Create Alert Action Rule

1. Navigate to **Event Management > Rules > Alert Action Rules** and click **New**.

   Fill in the following information in the Alert Action Rule:

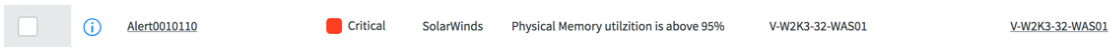
   Name: **Windows Critical**
   Alert Filter:
   Severity is Critical <AND>
   Configuration Item.Class "is" Windows Server <AND>
   Role in Group "is not" Secondary (Can you explain why this would be important?)
   Action Tab Fields:
   Auto Open: **Checked**
   Auto Acknowledge: **Checked**
   Type: **Incident**
   Incident Template: **Windows Incident Template Event Mgmt**
   (It is in the incident template that you assign it to the **Windows Server Team** to work the issue.)

2. Click **Update** to save the record.

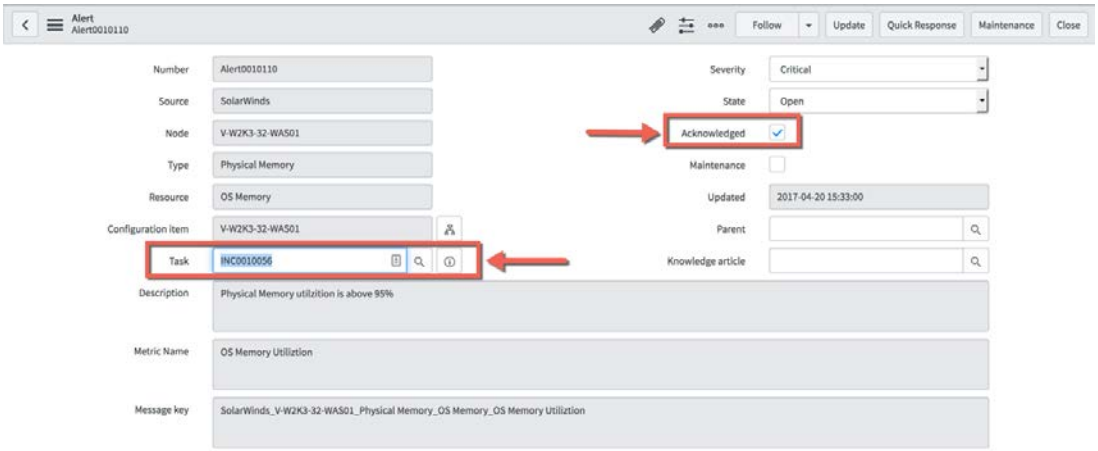


3. Now, generate an event to test the Alert Action Rule. Open **Postman** and select **Lab 2.2: CMDB Alert Action Event.** Click **Send.** This sends a Critical event from SolarWinds for Memory exceeding 95%.

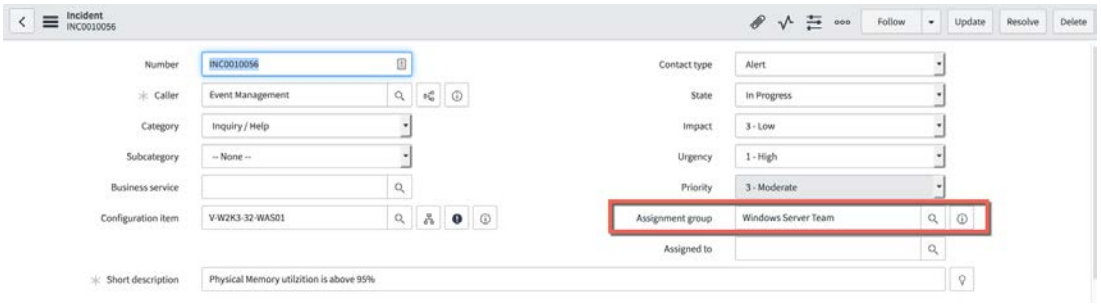
4. In ServiceNow, navigate to **Event Management > Alert Console**. Click the alert that was generated.



5. **Click** to open the Alert and notice that the Alert record has been updated based on the Alert Action. An incident record has been opened and the Alert has been Acknowledged.



6. In the Alert record, click to open the **Incident** record and verify that the incident ticket has been assigned to the **Windows Server Team**.



Lab Goal

This lab explores a scenario of triaging a service outage to show impact of an event on a critical business application. This shows the value of implementing Service Mapping along with Event Management, which allows a company to prioritize what issues need to be worked on based on impact to the business.

Lab 3.0 Triage a Service Outage

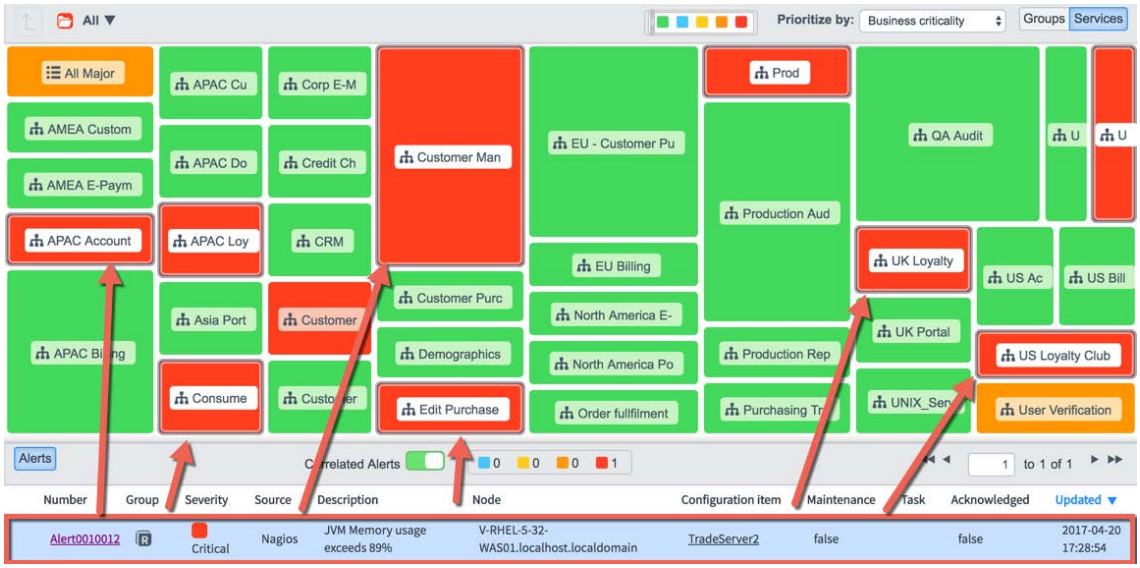
Detect and Prioritize

This scenario uses the **Lab 2.1: Manual Alert Correlation Events** that were sent in earlier. If you closed the alerts that were generated, do the following steps to open the alerts again. If you are unsure, go ahead and run these steps as any additional events are automatically correlated with the already existing alerts.

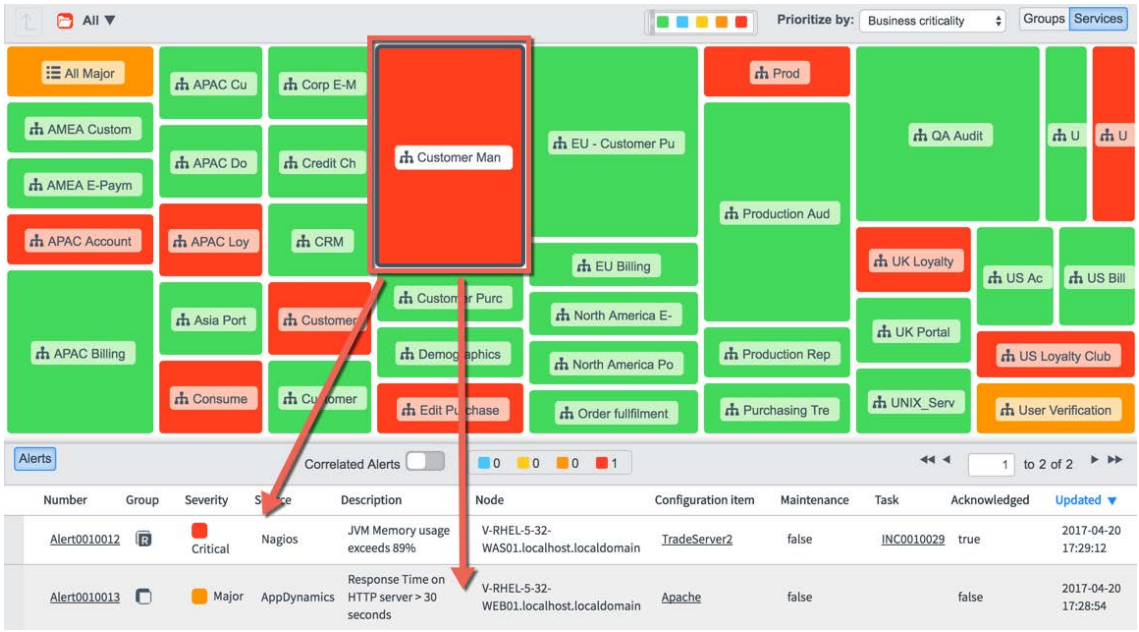
1. Open **Postman** and select **Lab 2.1: Manual Alert Correlation Events**. Click **Send**.

2. In ServiceNow, navigate to **Event Management > Dashboard.** The Event Dashboard shows status of all Business Applications at the top and all active alerts at the bottom. This gives a view of health of all applications in the environment.

3. Click the **JVM Memory usage exceeds 89%** Grouped alert in the dashboard and notice how all affected services are highlighted. This allows you to see the impact of alerts on the applications the organization depends on and helps you prioritize what issues are worked on first.



4. Now, click the **Customer Management** application box in the top. This filters alerts that are affecting infrastructure that are supporting that application.

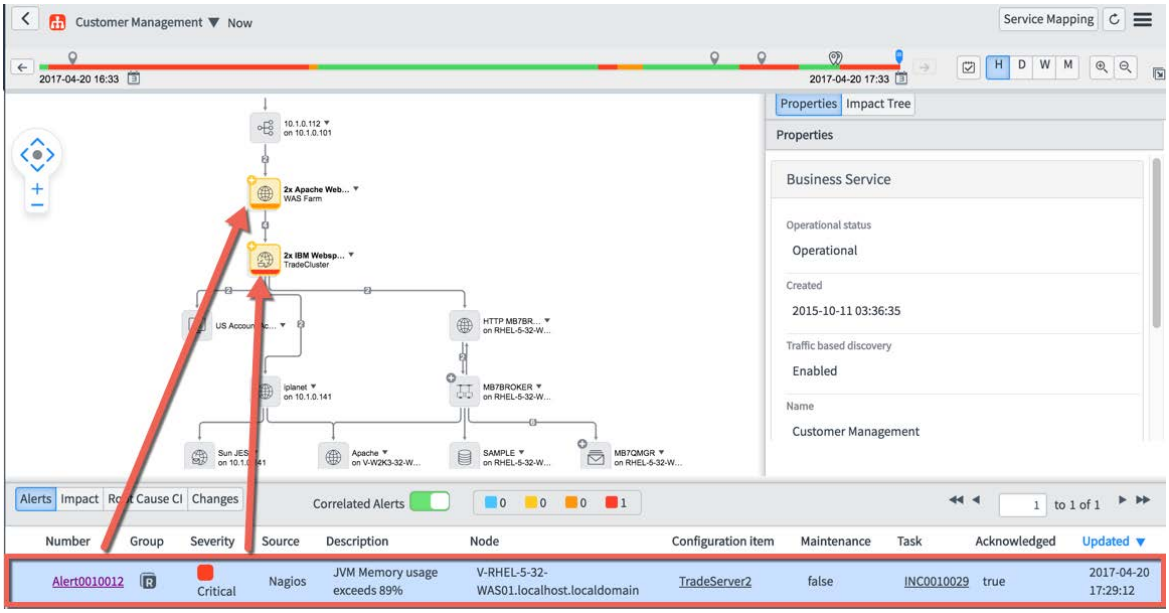


Analyze

Now analyze the issue affecting **Customer Management** application.

1. Double-click the **Customer Management** application square to bring up the application health map.

2. Select the **JVM Memory usage exceeds 89%.** This highlights **2x Apache Web** and **2x IBM WebSphere – TradeCluster**, which are the configuration items that are affected by the CMDB Grouped Alert.



3. Expand the **2x Apache Web Server – WAS Farm** and **2x IBM WebSphere – TradeCluster** groupings to drill down on the affected items in each group. Drilling in, note the affected components are **Apache on V-RHEL-5-32-WEB01.localhost.localdomain** and **TradeServer2 on V-RHEL-5-32-WAS01.localhost.localdomain.**

4. One of the first things to do when troubleshooting an issue is to determine if something changed in the environment. Service Mapping rediscovers applications on a scheduled basis and automatically detects changes to applications.

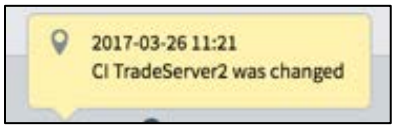
5. Just above the map window on the right side of the **History Bar,** click the **arrow icon** to get more detail. The display should look like the following.



6. This expanded view shows what has changed between two points in time in the business service. This can be useful when troubleshooting issues to see what has changed in the environment.

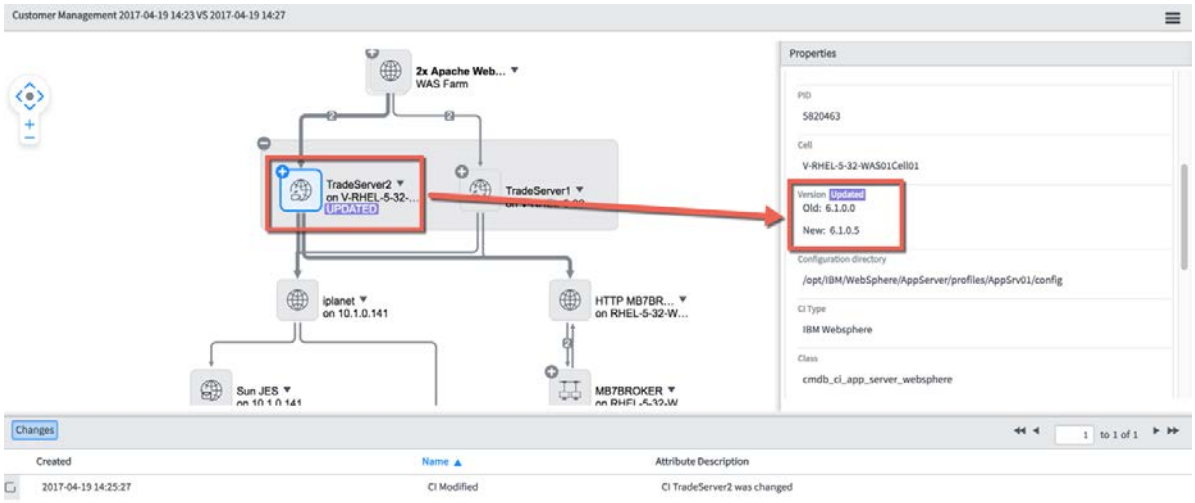
7. Change the view to hourly by click on **H** for hourly view 

. You should see a change bubble  on the timeline indicating a change. Hover over it with the cursor to see details about what has changed and when it was changed.



8. Using **Point #1** and **Point #2** at the bottom left of the menu, drag **Point #**1 to <u>before</u> the change bubble and draft **Point #**2 to <u>after</u> the change bubble. Click **Compare**. You should now see a new tab created with the selected date range.

9. Expand the **TradeCluster** and select **TradeServer2**, since this has been changed. In the right **Properties** view, note that this server has been upgraded recently. This may be the cause. Roll back the change to see if this resolves the issue.



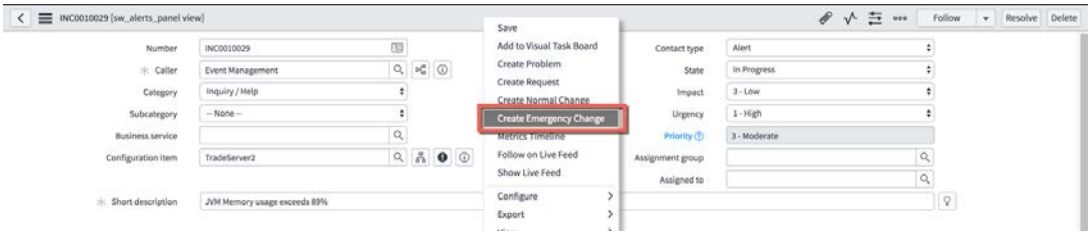
Respond

Create an Emergency Change against this issue to roll back the change.

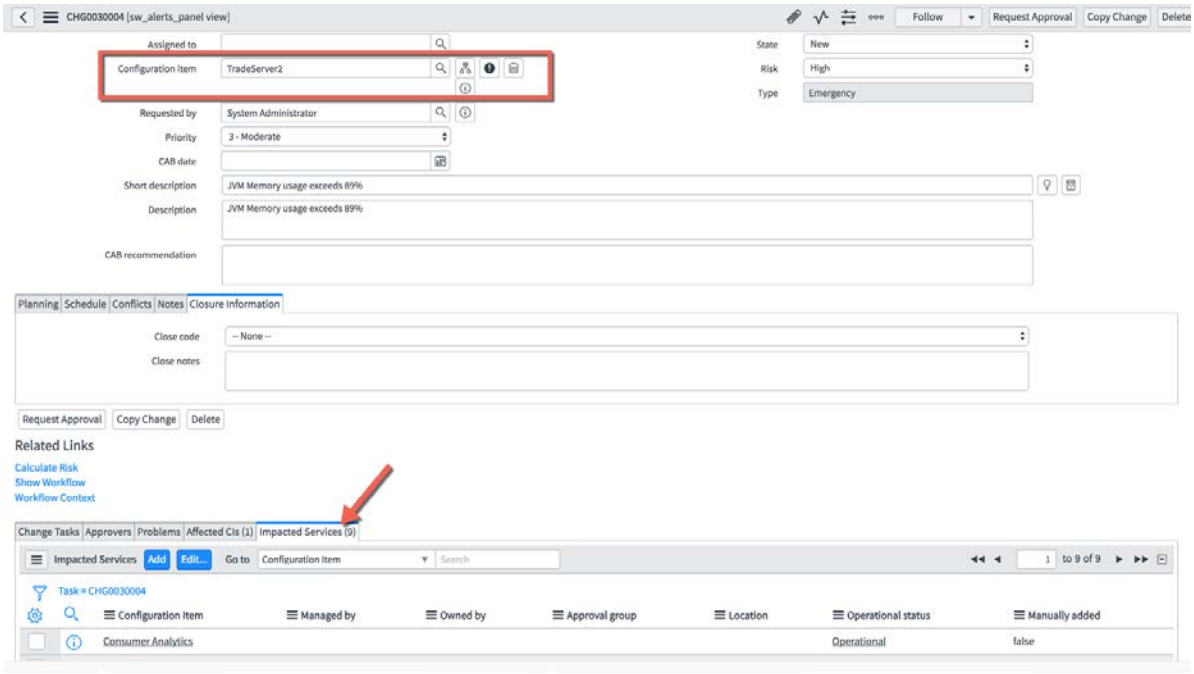
1. Return to **Dashboard View** and click in the **Incident** associated with this alert.



2. When the incident record appears, right-click in the gray header and select **Create Emergency Change**.



3. An Emergency Change record is automatically created in which incident properties are carried forward like the **Configuration Item** that is affected. On the **Impacted Services** tab, you can see all the other impacted services that also use **TradeServer2** component.



4. Lastly, to make sure appropriate risk mitigation is put into place there is an out-of-the-box risk condition rule that raises the **Risk** to **High** and **Impact** to **High** if a critical service is impacted**.** This can be seen by clicking on **Calculate Risk** link in the middle of the change record.

