servicenow

Service Usage & Health Analytics (SUHA)

User Guide

VERSION 2.2

09/11/2018

TABLE OF CONTENTS

OVERVIEW

SUHA is a self-service framework that enables the user to measure their services with minimal configuration. It provides a consistent view of all the services, their health and other metrics like performance and usage.

TERMINOLOGY

Groups

- Groups are entities that correspond to the services or logical grouping of services (teams/organization).
- They are hierarchical in nature and help SUHA to capture the organizational hierarchy
- Group can be of type '**Group'** or type **'Service'**
- Group of type **'Group'** can contain (be a parent of) other '**Groups'** or '**Services'**
- Group of type **'Service'** cannot be a parent of any other group

Metrics

Metrics hold the description of the quantity that needs to be measured. For the same metric, different services can define different quantitative logic.

Group Metric Definitions

Holds the actual quantitative/evaluation logic that defines the metric for a given service and is the most critical piece. Also holds other defining attributes like collection frequency, expected metric range etc.

Data Source

Defines the table and the query that will be used to extract the data required to compute different group metrics. It is also possible to define aggregated queries. Data sources are used in group metric definition, where they are available in the evaluation script as a GlideRecord or a GlideAggregate.

Dashboards

Three dashboards are available which allows us to render the metrics for each service.

- **Health Dashboard** – A common place to display the current health of all the services within Service Engineering Organization that have been on boarded. This is also the entry level dashboard to all the other dashboards.
- **Service Level Dashboard** – Displays a few other top level important metrics for each service. Each service can configure different metrics to be seen for each service. One logical dashboard per service.
- **Group Level Dashboard** – Displays aggregated view of the metrics for the services which are under the group/team. One logical dashboard for each group/team.

Dashboard Configuration

Used to configure the metrics to be seen on the respective dashboard.

SUHA ONBOARDING

SUHA application modules can be accessed via filter navigator, by typing in **'SUHA'**. If you are not able to access/see some modules, you need to request for the right role.

In this section, we will go through the detailed steps on how to create SUHA components, that will be a part of the onboarding process.

To get a high level overview of the onboarding process, refer to the section standard onboarding flow.

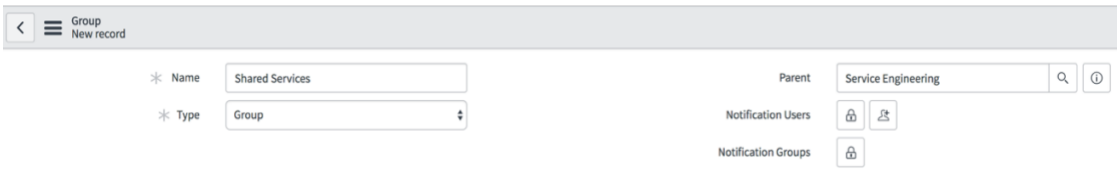
NOTE: All the configuration has to be created in **DCDev** and captured in update sets. This configuration will be moved to DCTest and DC via update sets.

For any issues related to permissions and configuration reach out to CloudAnalytics@servicenow.com

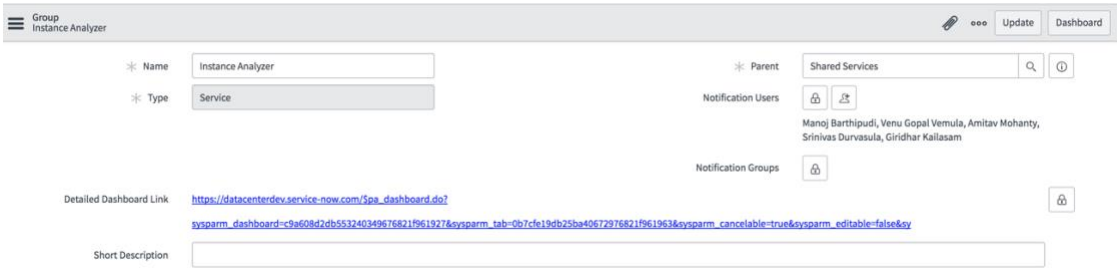
CREATING A GROUP/SERVICE

Example: Onboard a service **'Instance Analyzer'** within team **'Shared Services'**, which is within Service Engineering.

1. Browse to *SUHA > GROUPS*
2. **Service Engineering** group already exists
3. Click on New to create the **'Shared Services'** group
4. Fill in the form with details as shown below and save



5. Click on New to create the **'Instance Analyzer'** service
6. Fill in the form with details as shown below and save

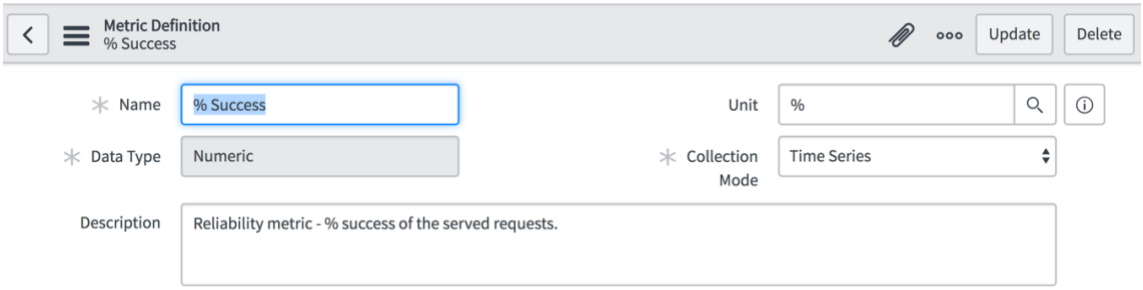


7. Notification Users/Groups define the set of users, who would be sent notifications related to metric definitions for the group/service, in case the metric threshold is breached
8. If your service has an existing detailed developer dashboard developed outside SUHA (even outside Datacenter Instance), provide the link here. This dashboard will be automatically linked to the SUHA service level dashboard
9. Provide a short description on what the service does

CREATING A METRIC

Example: Create a metric **'% Success'**

1. Browse to *SUHA > METRIC DEFINITIONS*
2. If a metric definition already exists which match the criteria of your metric, it can be reused
3. To create a new metric, fill the form with the details as shown below and save
4. If the metric needs to be measured more frequently than a day, choose '***Real Time'*** as the *Collection Mode.* Otherwise choose '***Time Series'***

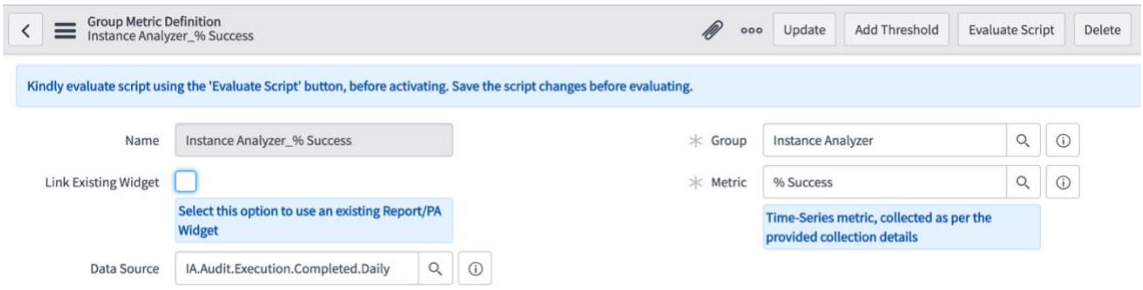


CREATING A GROUP METRIC DEFINITION

Group metric definition holds the actual evaluation logic for the metrics we want to measure for the service. It holds other configuration like the collection frequency. This is the most critical piece and needs most time spent in thinking and defining the correct logic.

Example: Define **'% Success'** metric for the service '**Instance Analyzer'**

1. If the group metric definition already exists, edit it to make any changes
2. To create new definition browse to *SUHA > ONBOARD > CREATE GROUP METRIC*
3. Fill in the form details as shown below



4. Here the data source (which is already created) denotes the source table and query that would be used in the evaluation logic. It is not mandatory to have a data source, and you can write the entire custom Glide script within the evaluation logic
   NOTE: It is recommended to use Data Sources wherever possible

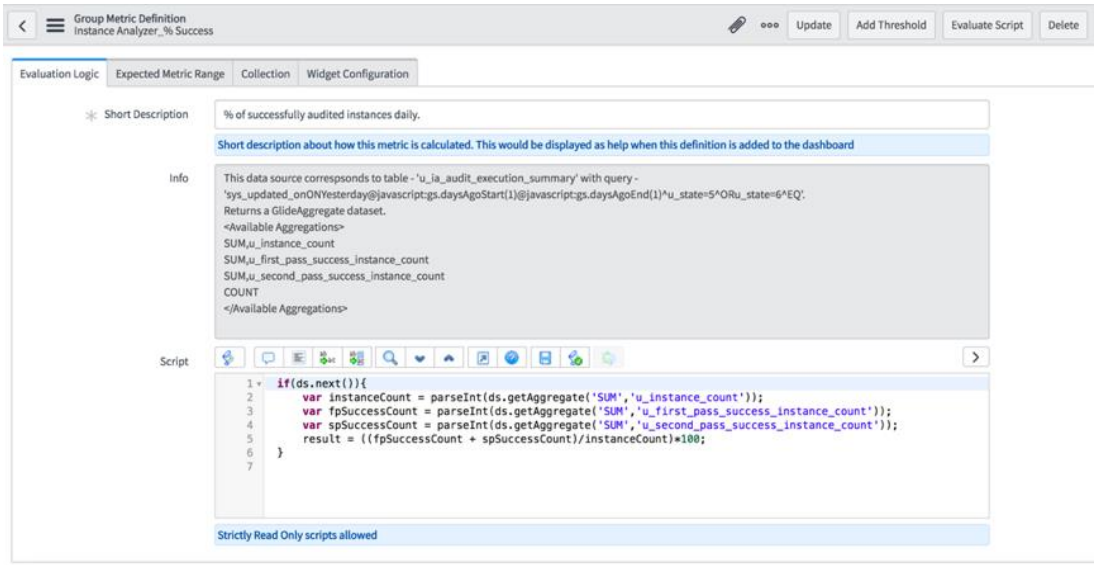
5. On populating the data source, info field is auto populated, and provides a handy description of the data source

6. Write the evaluation logic within 'Evaluation Logic' section tab

   - Evaluation script has access to three special variables **'ds', 'result', 'logs'**
   - **'ds'** – Selected data source (GlideRecord or GlideAggregate object depending on the data source configuration)
   - **'result'** – To be assigned the final evaluated metric value
   - **'logs'** – An **array** into which the you can push debug statements.

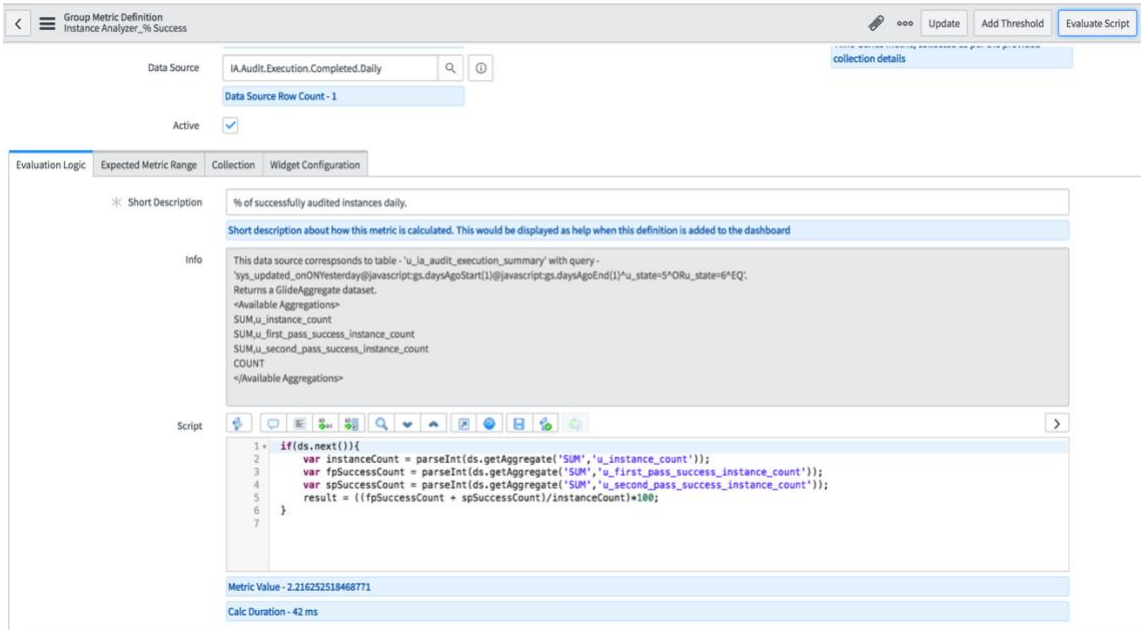
7. Short description field describes about the evaluation logic (how are we calculating the metric)

8. In the example shown below, the data source is a **GlideAggregate** data source



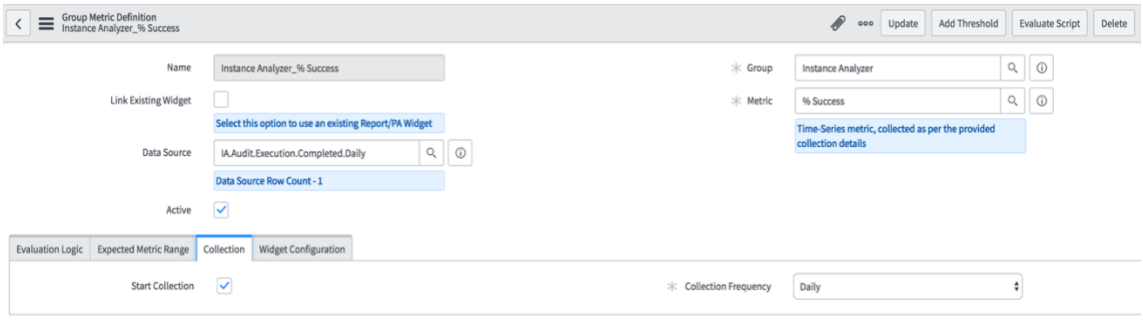
9. Once the evaluation logic is defined, save the group metric definition and click **'Evaluate Script'**

10. This would evaluate the script and display the calculated value, duration and data source row count next to the corresponding fields as shown in the image below



11. Make sure that the evaluation logic is defined correctly, and also computes in reasonable time (not more than 30 seconds)

12. Define the collection details in the **'Collection'** tab



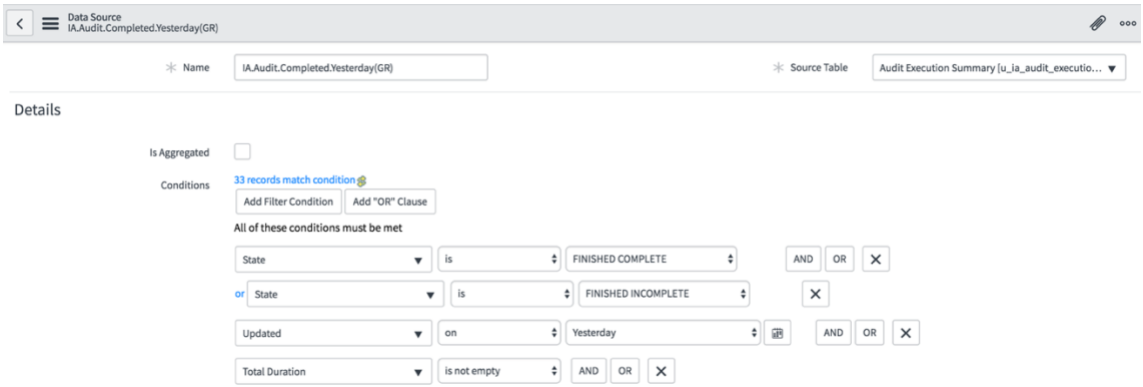
13. Set the Widget Title in the 'Widget Configuration' tab. If not defined, the metric name is used as the widget title, when the metric is configured to be displayed on the Service Level Dashboard

14. One the entire configuration is verified, check the 'Active' checkbox

CREATE A DATA SOURCE

Example: Define a data source for service 'Instance Analyzer', which would be used to calculate **'daily'** metrics. Data Source returns all the audits which completed yesterday (**STATE = (FINISHED COMPLETE || FINISHED INCOMPLETE)**)

1. Browse to ***SUHA > DATA SOURCES***
2. Select the **'Source Table'** and define the **'Conditions'**



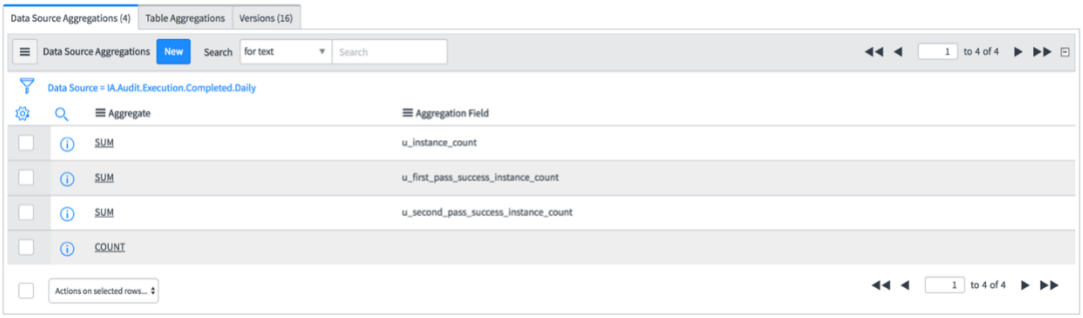
3. **NOTE:** In the conditions, we have a query **'updated on yesterday'**
    - This is critical for our group metric definition with **'daily'** collection frequency to function correctly
    - This part of the query can be replaced with any other query which makes sure that for calculating a time series metric value today, we get the intended data for yesterday
    - Every **'daily'** collected time series metric is evaluated at **0:00 PST** every day
4. This data source is a GlideRecord data source, since it is not aggregated

CREATE AN AGGREGATED DATA SOURCE

Aggregated data source provides with a GlideAggregate object, when used in a Group Metric Definition. Always prefer an **Aggregated Data Source** unless it does not suffice your metric calculation requirements.

1. Create a Data Source as described in the previous section
2. Check the *'Is Aggregated'* checkbox
3. Save the group metric definition
4. This would enable a ***'Data Source Aggregations'*** related list

5. Add new aggregations to the data source from the related list

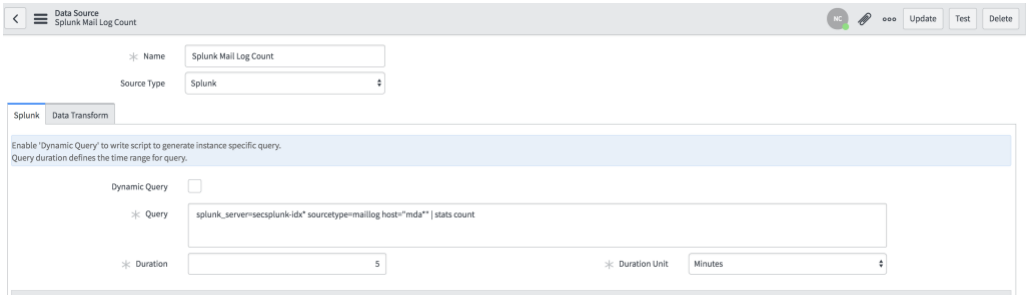


6. This would provide with a ***GlideAggregate*** object when selected in a Group Metric Definition

SPLUNK AS DATA SOURCE

Static Query:

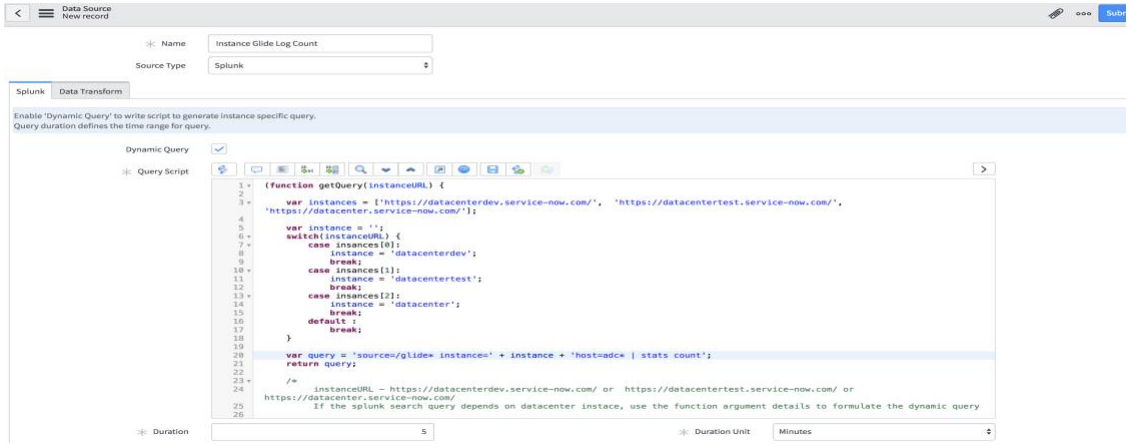
1. Browse to *SUHA > DATA SOURCES*
2. Provide 'Name' to Data Source, select 'Source Type' as 'Splunk'
3. In Splunk section
    a. Provide valid Splunk query in 'Query' field
    b. In 'Duration' field, fill in time duration that would be the range for querying logs in that time range.
    c. In 'Duration Unit', select the duration unit.
    Example: For the Data Source shown in the screen shot, Splunk logs queried are in the last 5 minutes range from the time of execution.
    d. Screen shot below shows the sample query configured



Dynamic Query:

If the query varies from instance to instance, use 'Dynamic Query' option to write a script and generate the run time query.

1. Enable ‘Dynamic Query’ to formulate run time query.
2. Write script in ‘Query Script’ field that would return valid final query.
    a. Script arguments
        i. *instanceURL:* URL of the instance.
    b. Return valid Splunk query as string
    c. Below example adds ‘instance’ dynamically based on the instance from which data source is being executed from.

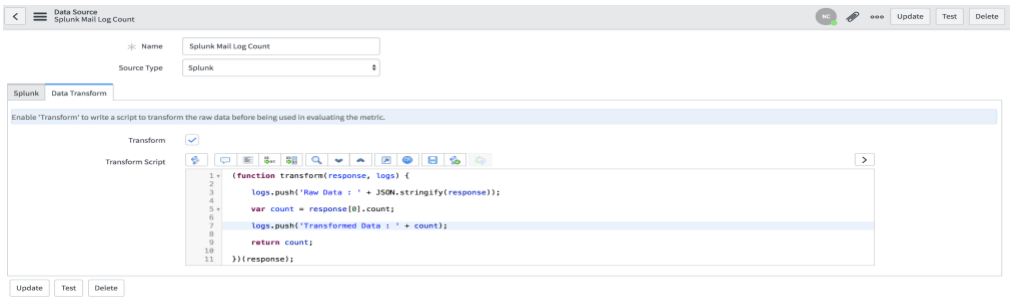


3. Submit to save the record

Transform Data:

Transform the raw data retrieved from Splunk into a user intended data structure.

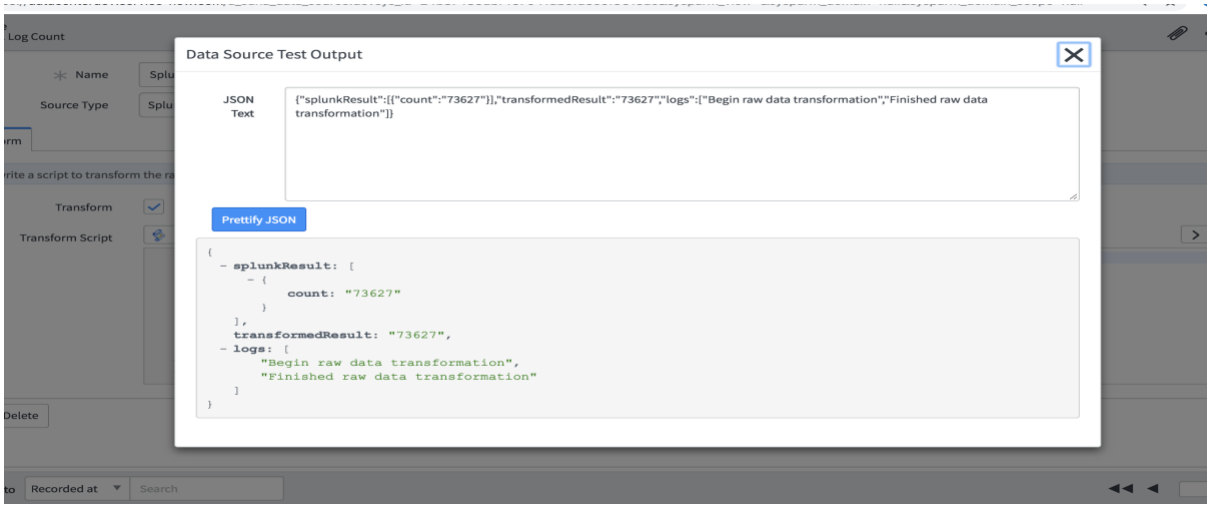
1. In ‘Data Transform’ section, enable ‘Transform’
2. Write script in ‘Transform Script’ to process and return user intended data structure.
    a. Script arguments
        i. *response:* It has the raw Splunk data
        ii. *logs:* Empty array. Transform script logs can be pushed to this array for testing and debugging purpose.
    b. Return the data in the user intended format.
3. Screen shot below for reference



Test Data Source:

Verify the correctness of the Splunk query with 'Test' button available

1. Save the Data Source record before clicking 'Test'
2. Result of the 'Test' action is as follows
    a. *splunkResult:* This holds the actual result from Splunk
    b. *transformedResult:* This holds the output returned from the transform script (if any)
    c. *logs:* This holds the array of logs pushed in the transform script(if any).
    d. *error:* This holds the error messages that resulted in testing the data source. Error could at Splunk api level or in transform data script etc
    e. Sample result of test action is as shown in screen shot



THRESHOLDS AND NOTIFICATIONS

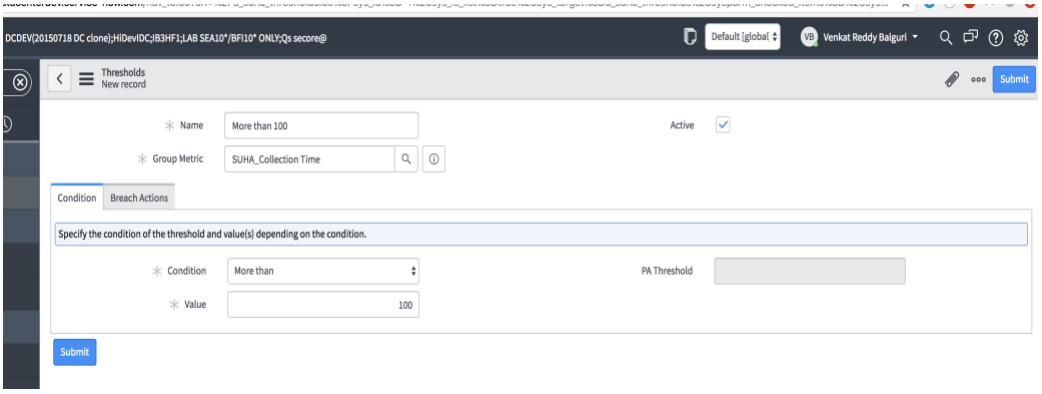
Every group metric has an expected range, defined the service owner. User can configure to get a notification if this range is breached.

1. Define the expected metric range, while defining the group metric definition
2. Go to the '*Notifications'* section/tab and add the notification users and groups to be notified
3. If notification users and groups have been configured at the service level and they need to be notified, check the *'Notify Group Notification Members'* checkbox
4. Under the *'Expected Metric Range'* section, check the *'Range Breach Notification'* checkbox

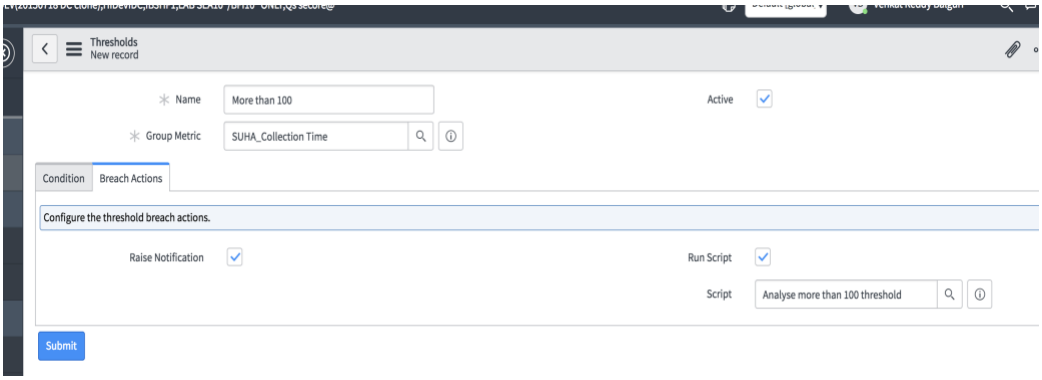
Apart from the expected metric range, the user can define multiple thresholds on breach of which breach actions are executed as configured. Configure multiple thresholds for a Group Metric by following the steps below

1. In the Group Metric Definition form, click on the *'Add Threshold'* button and fill in 'Name'
    OR

Browse to SUHA -> Configuration -> Thresholds and click ‘New’ to create a new threshold. Fill in ‘Name’ and ‘Group Metric’ as shown below



2. Toggle ‘Active’ check box to activate/deactivate the threshold
3. Fill in the values for threshold condition and the threshold value in ‘Condition’ and ‘Value’ respectively
4. For condition as ‘outside’, configure the range of values - ‘Lower Value’ and ‘Upper Value’
5. In the ‘Breach Actions’ section, configure the actions needed to be performed on the breach of this particular threshold
6. Enable ‘Raise Notification’ to receive email alerts on breach of the threshold
7. ‘Run Script’ and select a script to execute this script on the threshold breach as shown below.
   Go to ‘*Diagnostic Script*’ section for detailed notes about script run on threshold breach



8. Submit to save the record

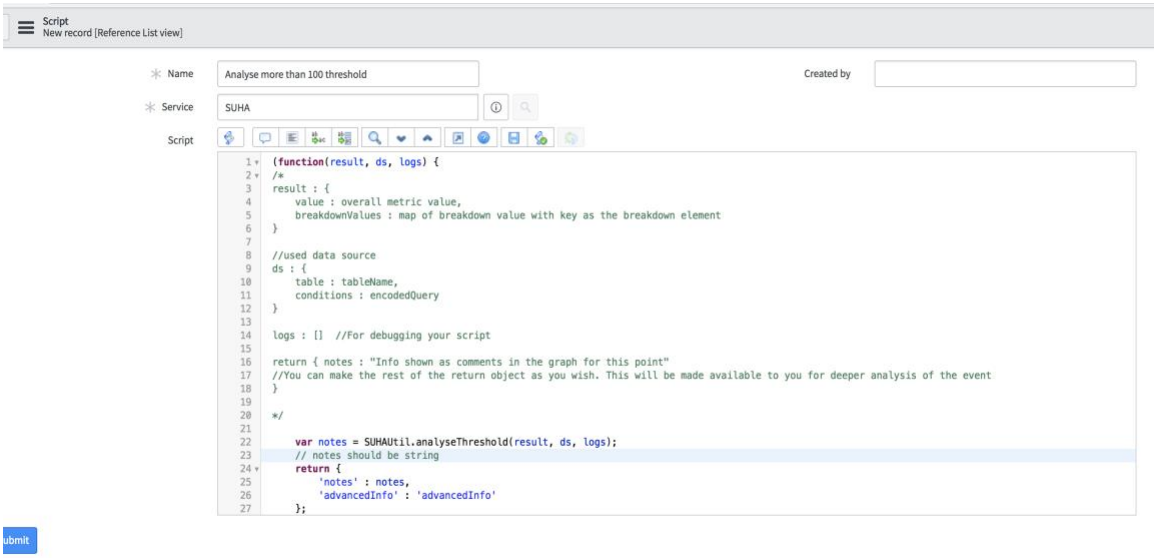
DIAGNOSTIC SCRIPT

On breach of the threshold, run Diagnostic Script to analyze the root cause of the breach. The result notes of the script will be shown on the widget at that particular breach point.

The script evaluates on breach of every threshold. If the same threshold breach event occurs continuously, script run happens only after the 'Min Notification Gap (Hours)' configured at the Group Metric level.

CONFIGURING DIAGNOSTIC SCRIPT

1. Browse to *SUHA -> Configuration -> Scripts*
2. Click 'New' to create a new script
3. Fill in 'Name' and 'Service'
4. Write script in the script column and submit as shown below



5. Choose the script as 'Run Script' on any threshold

Script arguments:

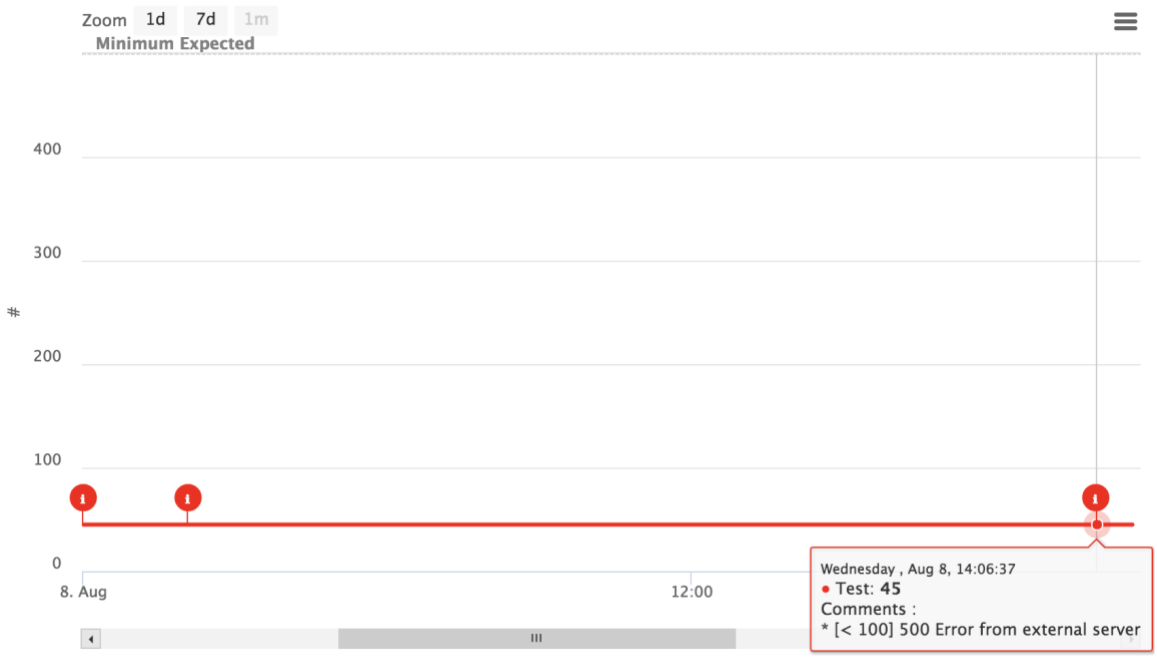
- **result**: JSON contains the result object with '*value*' holding metric value and '*breakdownValues*' holding the metric breakdowns (if any)
- **ds**: JSON with '*table*' holding the data source table name and '*conditions*' holding the encoded query of the data source conditions
- **logs:** Holds array of logs from the metric evaluation logic

Script return format:

- Script should return a JSON with 'notes' having the outcome of the Diagnostic Script action
- Additional key-values can be present in the result JSON as shown in the above script screen shot

View Diagnostic Script notes:

- Diagnostic Script Run notes is displayed on the widget on hover of the threshold breached point under '*Comments*' section as shown below.
- Since multiple thresholds can be configured for single metric, if multiple threshold breaches happen and different scripts are configured to run as diagnostic scripts, each script's comments are separated by '*' followed by threshold condition as *[< 100].
- *[< 100] infers metric value is below the threshold of 100 and the script notes is specific to this threshold breach.



CONFIGURING NOTIFICATION USERS AND GROUPS

Notifications users and groups can be configured at two levels.

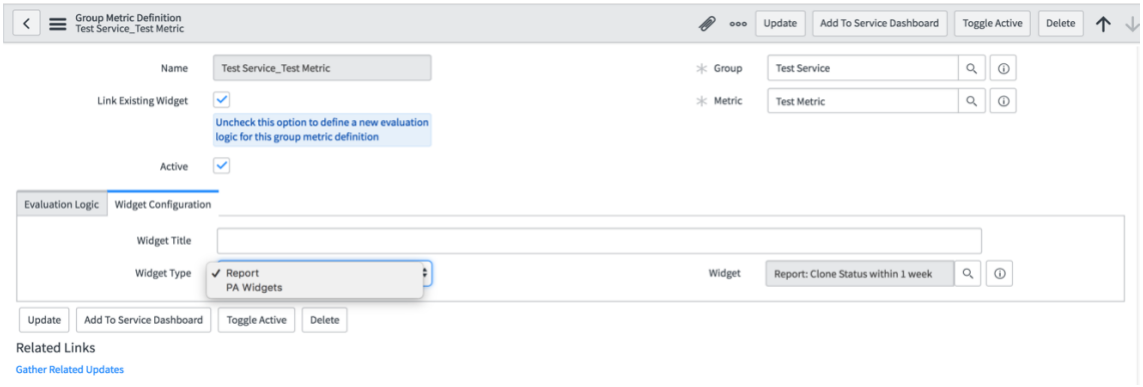
- Service/Group
  - Configure the members here so you do not need to configure the same every time at the group metric definition level
- Group Metric Definition
  - Configure the specific notification members who need to be notified for this group metric definition in addition to the members configured at the group level.

- Members configured at the group level can be notified by checking the *'Notify Group Notification Members'* checkbox under the *'Notifications'* section/tab

LINKING AN EXISTING WIDGET TO A GROUP METRIC DEFINITION

If we have already configured a **report** or a **PA Widget** that we wish to display as a group metric definition. This implies that we do not want to configure **SUHA** to evaluate/gather that metric, but just wish to render it on the Service Dashboard for a group metric.

1. Create a new group metric definition. If the group metric definition already exists, open it for editing
2. Check the checkbox which reads '*Link Existing Widget*'
3. Once the checkbox is checked, it would hide the configuration options to gather/evaluate the metric
4. In the *'Widget Configuration'* tab select the *'Widget Type'* as '*Report / PA Widgets'* and the corresponding *'Widget'*



NOTE: This implies that the group metric definition is just a link to the configured report/pa widget for rendering purposes on the dashboard and does not control its evaluation in any manner.

CREATE GROUP METRIC DEFINITIONS WITH BREAKDOWNS

SUHA allows a breakdown / drilldown by some criteria on a metric. For example, if we are measuring **Incoming Requests'** as a daily metric for a service, we might be interested in a breakdown by **'Request Type'.** So here, **'Request Type'** is the **'Breakdown source'** and each type of request is a **'Breakdown Element'.**

If the **breakdown source** is an existing column in the **data source** table selected as the data source, we call this a **'Standard Breakdown'** where all we need to do is to select the column which would be treated as the breakdown

source, here the evaluation script logic needs no modification, and is same as you would define for a non-breakdown group metric definition.

Other scenarios could be as follows:

- Breakdown Source exists as a column in some table but not the table selected in the data source
- Group metric definition does not have a data source
- Breakdown Source is not stored in any column, but is driven by some custom logic

SUHA supports breakdown in all these scenarios as well, we call this as a **'Custom Breakdown'.** For configuring a custom breakdown, we need to change the way we write our evaluation script such that its result is an object which contains a metric value corresponding each breakdown element, which is the key. We will be now discussing each scenario going forward.

STANDARD BREAKDOWN

This is the simplest way to define a metric with breakdowns. To leverage this functionality, you must work with a data source. The idea is to define the evaluation logic as if you were not trying to breakdown and just measuring the overall metric and assigning the evaluated metric to the ***'result'*** variable. SUHA internally splits the dataset into multiple filtered data sets, based on the possible values in the breakdown field and applies the same evaluation logic to each of those datasets to collect the breakdown metric values.

For example, if you want to define **'Incoming Requests'** for your service and have a breakdown on the **type** of request (where **'type'** is a column in the selected data source table). You simply define the evaluation logic as:

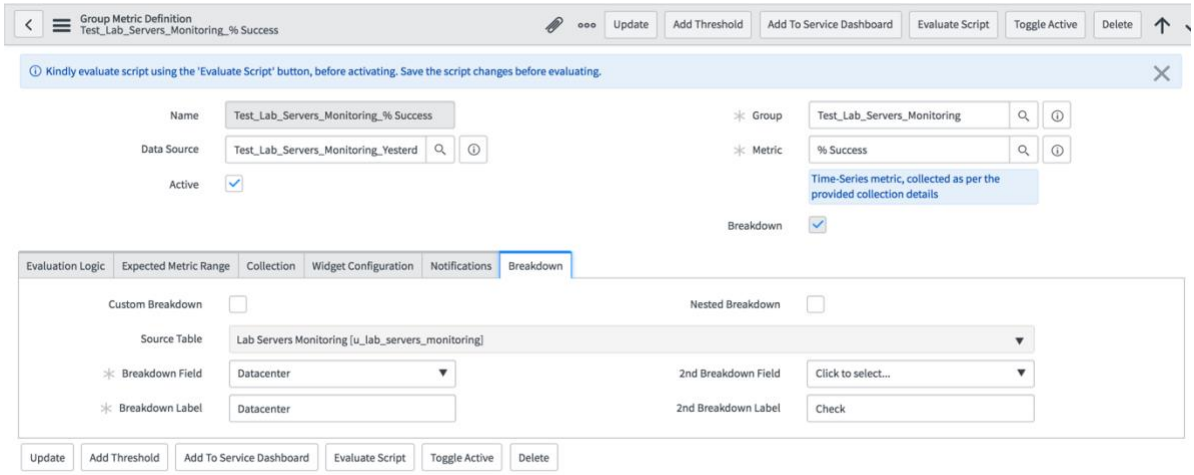
```
var count = 0
//Data source is 'aggregate' type and is an instance of GlideAggregate
if(ds.next()) {
   count = parseInt(ds.getAggregate('COUNT'));
}
result = count;
```

Now assume that the **'type'** field has 3 possible values. The evaluation logic would be run 4 times:

- Once for the overall metric value on the unfiltered data source
- Thrice on filtered data sets, for evaluating the metric value for each **'type'.**

The steps to configure a group metric with standard breakdown are as follows:

1. The group metric definition **must be defined using a data source** to leverage this feature
2. While creating a group metric definition, select the *'Breakdown'* checkbox
3. In the *'Breakdown'* tab the source table should be auto-populated from the data source
4. Select the *'Breakdown Field'* and provide a *'Breakdown Label*'. Breakdown label is a short name for the field you are selecting as the breakdown source, this label would appear in the widget for selecting this breakdown



5. Make sure *'Custom Breakdown'* is **unchecked**
6. If we want to a second breakdown as well, fill in the similar details for the *'2nd Breakdown'*
7. If we want the 2nd breakdown to be nested within the first breakdown, i.e. we want a two-level drill down, check the *'Nested Breakdown'* checkbox. Otherwise the two breakdowns act as independent breakdowns

NOTE: Breakdown field that can be selected can only be of type 'String, Reference, GUID, Choice'

CUSTOM BREAKDOWN

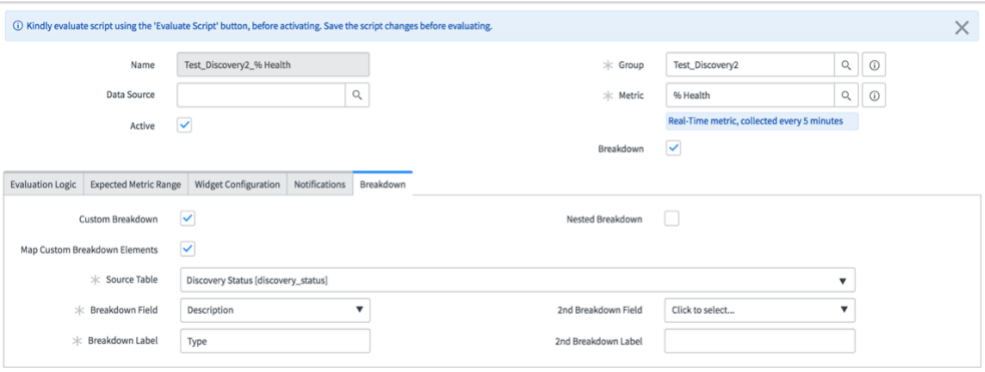
Users would need a custom breakdown, when the breakdown elements are not a column in the data source, but in some other table. Or when the breakdown elements are not present in any table column, but evaluated by some custom logic. We will be considering both the scenarios in this section.

We need to modify our evaluation logic to explicitly return the overall metric value and the breakdown values. The **'result'** object structure is as shown below

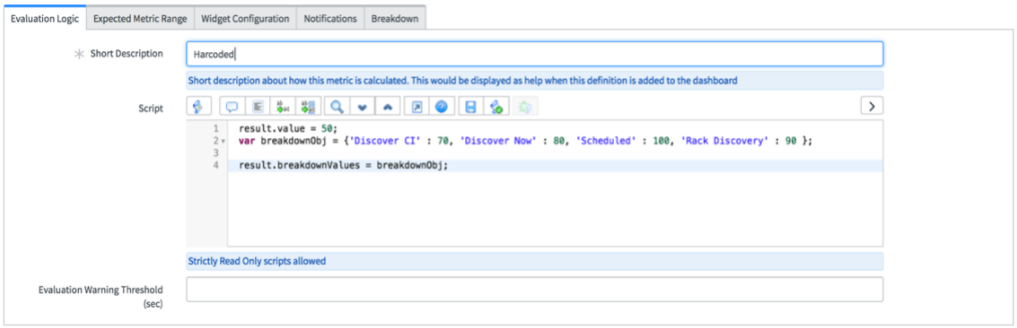
```
result : {
   value : val,  //overall metric value
   breakdownValues : {key1: val1, key2 : val2}  //keys are the breakdown elements
}
```

While defining the group metric definition check the *'Custom Breakdown'* checkbox.

- **Breakdown Source exists as a column in a table**



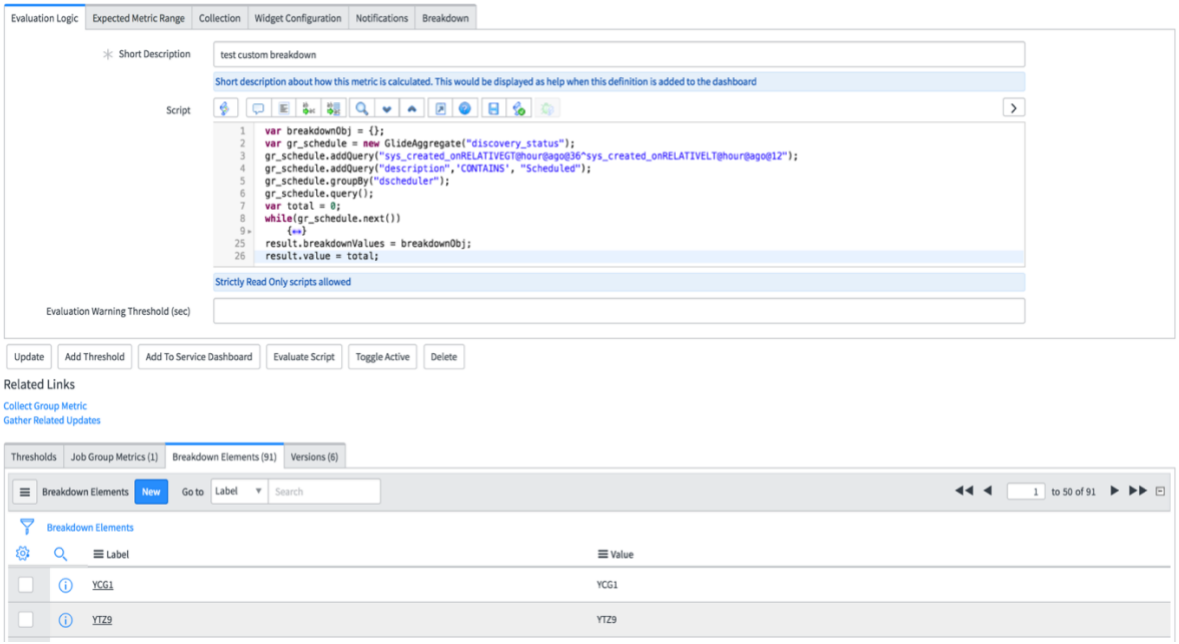
  - Check the *'Map Custom Breakdown Elements'* checkbox
  - Select the *'Source Table',* populate the *'Breakdown Field'* and *'Breakdown Label'* **(This table can be different than the data source table***)*
  - The **result** of the evaluation logic needs to be an Object with the above defined structure.



NOTE: The keys of the breakdownValues object are the same as the possible values of the selected breakdown field. In the shown example *"Discover CI", "Discovery Now"* are the values of the "*Description*" field of the table "*Discovery Status*"

- **Breakdown Source and elements do not exist in a table column**

- **Uncheck** the *'Map Custom Breakdown Elements'* checkbox
- Provide the 'Breakdown Label'
- The keys of the *result.breakdownValues* object defined in the evaluation script are treated as the *Breakdown Elements.* These breakdown elements appear in the 'Breakdown Elements' related list when the group metric is evaluated for the first time, as per the defined *Collection* schedule
- Label of the breakdown elements is what appears in the widget, and is **editable**.



WRITING THE EVALUATION SCRIPT

The evaluation script block has access to two variables *'ds'* and *'result',* where *ds* is a queried **GlideRecord** or a **GlideAggregate** object corresponding to the selected *data source* and *result* is the expected metric value calculated by the script logic.

Depending on the group metric configuration, the expected metric *result* data structure might vary.

1. **No Breakdown or Standard Breakdown**

   When the group metric definition has no breakdown, or has a standard breakdown, *result* is of decimal type. Script should evaluate the metric value(decimal) and assign that to the *result* variable.

2. **Custom Single Breakdown with breakdown element mapping**

When the group metric definition has a **single custom** breakdown and breakdown element mapping is configured as the *Source Table* and *Breakdown Field,* the expected *result* is an object with two properties – '*value'* and *'breakdownValues'.* Here *value* is the value for the overall metric and *breakdownValues* is an object map where each key corresponds to a **breakdown element** and its value corresponds to the metric value for that breakdown element. **This key should map to some value in the selected breakdown field.** For example, if the selected breakdown field is of type reference or sys_id, the key should either be a reference or sys_id. If the selected breakdown field is of type choice, the key should be one of the choice values**.**

3. **Custom Single Breakdown without breakdown element mapping**

   The expected result structure is same as for case 2. Here the key for the *breakdownValues* object can be any ***String*****.**

4. **Custom Nested Breakdown**

   The expected *result* is an object with two properties – '*value'* and *'breakdownValues'.* Here *value* is the value for the overall metric and *breakdownValues* is an object map of maps where each key of the **outer map** corresponds to a **breakdown element of the first breakdown** (key maps to some value of the breakdown field, if breakdown element mapping exists) and key of inner maps corresponds to a **breakdown element of the second breakdown** (key maps to some value of the 2nd breakdown field, if breakdown element mapping exists)**.** The value of the inner map corresponds to the metric value for each pair of nested breakdown elements (outer map key and inner map key).

ONBOARDER UTILITIES

To ease the onboarding process, SUHA comes with some handy utility features.

MOVE SUHA CONFIGURATION IN AN UPDATE SET

SUHA automatically creates a lot of configuration, which the user does not interact directly with. In order to move the changes to production all this configuration needs to be captured in an update set. SUHA automatically captures all the necessary updates in the **current** update set. To move the SUHA configuration, just move the user update set to the target instance. However, things might not be that straight forward most of the times. Here are a few scenarios

- The user switched update sets while working and updates are now distributed across multiple update sets, hard for the user to manually gather those updates
- The user created a lot of configuration, let's say five group metric definitions and a lot of sample configuration during ramping up on SUHA on development environment. Now he just wishes to take a part of that configuration to production. It would not be possible to manually separate out the updates corresponding to the partial configuration that needs to be pushed to production.

To handle such scenarios, Group Metric Definition have a defined UI action called ***'Gather Related Updates'.*** This UI action is available from the list view (actions) and the form view (context menu and related link) of the **group metric definition**. This action gathers all the updates related to the selected group metric definitions (e.g. group, metric, dashboard configuration) from all incomplete update sets and moves them to the **current** update set.

*NOTE: It is advised to always use the **'Gather Related Updates'** to ensure that your update set has captured everything that is needed for the configuration to be complete*

ON DEMAND GROUP METRIC COLLECTION

In the development phase of group metric, we are already aware of the **'Evaluate Script'** action, that is used to test our evaluation script logic. However, evaluate script just evaluates the script and displays the script output, it does not actually store that data for it to reflect in the widgets on the dashboard. In case of metrics collected daily or weekly, the user might need to wait for a day or a week before they could see some data in the chart and make sure if that's exactly what is needed. To handle this scenario, group metric definitions have a defined action called ***'Collect Group Metric'*** which occur as a related link in the UI form view. This action collects the group metric and updates the time series data for it to start reflecting in the widgets on the dashboard.

VALIDATING THE GROUP METRIC EVALUATION RESULTS

Under the 'Evaluation Logic' section/tab in the group metric definition form, we can see the metric evaluation results related list. The source of the evaluation results defines the evaluation trigger.

- Scheduled – Triggered as per the schedule defined under the *'Collection'* tab
- On Demand – Triggered manually using the *'Collect Group Metric'* link UI Action
- Test – Triggered manually using the *'Evaluate Script'* button UI Action

The output is a journal field that has a **Prettify JSON** link to view the output in a more readable format.

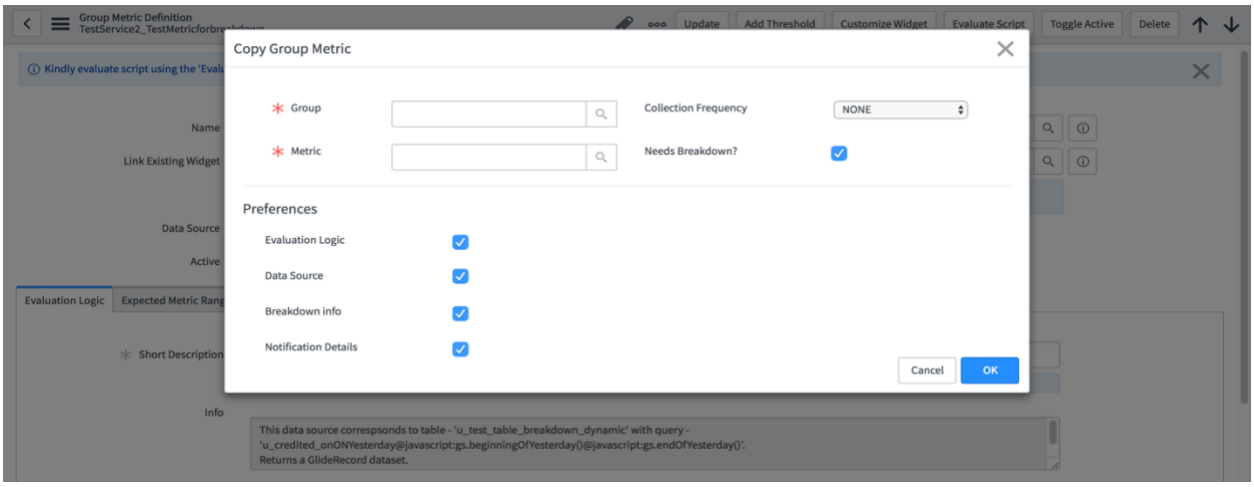
DEBUGGING THE GROUP METRIC DEFINITION EVALUATION LOGIC

In order to debug the logic, we can push log messages to the ***'logs'*** array variable that is available in the script. These messages can be viewed in the output field of the metric evaluation results

COPYING A GROUP METRIC DEFINITION

To create a new Group Metric Definition with similarities with existing one we can use the ***Copy Group Metric*** context menu UI action.

1. Open the Group Metric Definition which you want to copy from.
2. Open the context menu on the top-left corner and click on Copy Group Metric.
3. Copy Group Metric Popup opens up.

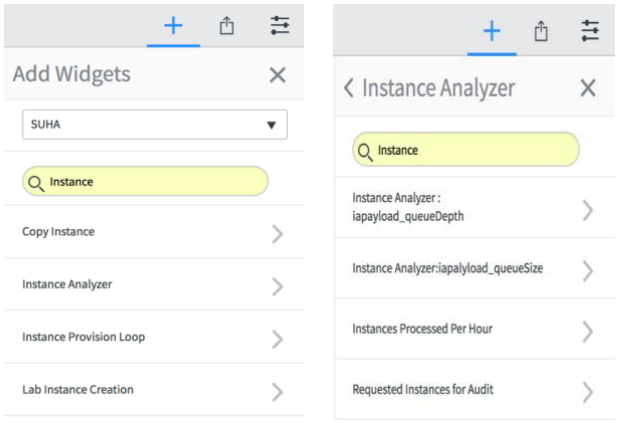


4. Select the '*Group*' and '*Metric*'. Make sure this group metric does not already exist in the system
5. Select the required *'collection frequency'* from the dropdown
6. Check the *'Needs Breakdown'* checkbox is the new Group Metric needs breakdowns
7. Copy Preferences allows you to select the information you want to copy
8. *'Data Source'* and *'Breakdown* Info' checkboxes are only enabled if the source Group Metric has a Data Source and breakdown respectively.
   NOTE : If *'Needs Breakdown*' is not checked, checking '*Breakdown Info'* preference has no effect
9. Click on *'OK'* to create a new group metric definition with data copied from the existing group metric definition with the selected preferences.
10. If there are no errors, page is redirected to the newly created group metric definition for further editing.

ADDING SUHA WIDGETS TO YOUR PERSONAL DASHBOARD

Widgets corresponding to group metric definitions can be added to any dashboard, just like any other PA widget or report

1. Open your dashboard and click on the "+" option to add widgets
2. Browse for **'SUHA'**
3. The widgets are grouped by your service and then the metric
4. Select the widget corresponding to the required group metric



5. Click on Add to add the widget to your dashboard.

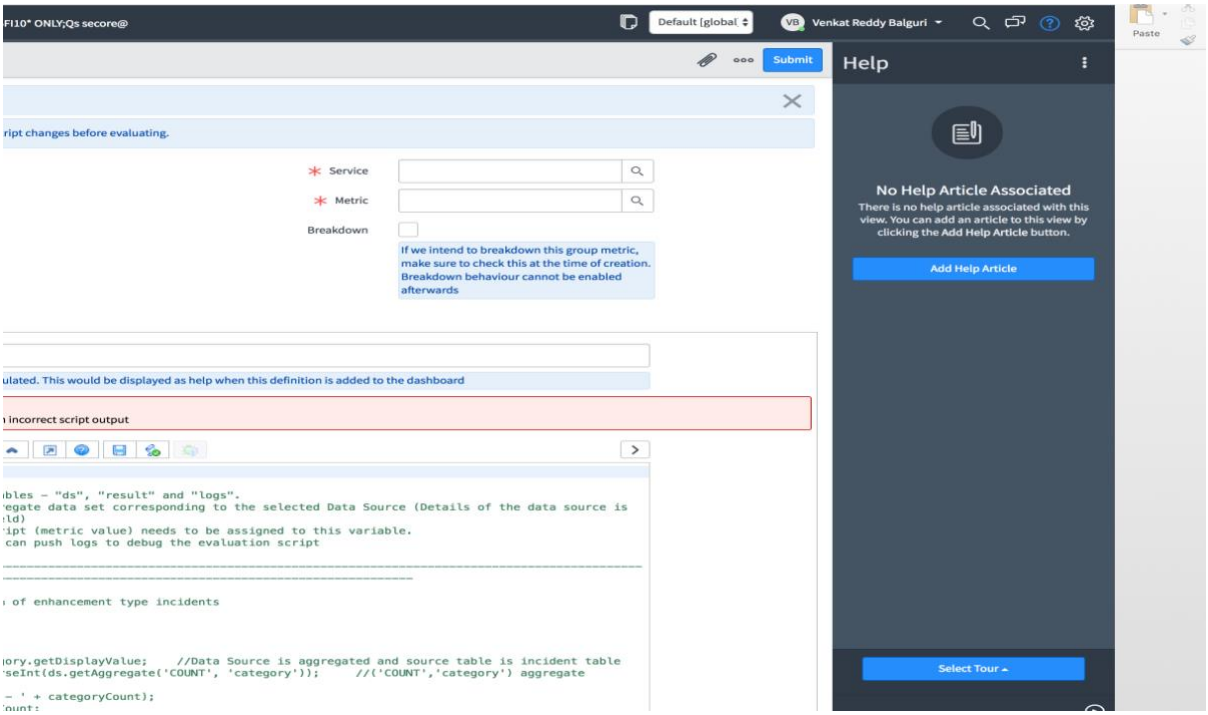
GUIDED TOURS

Guided tours help train and onboard users. Each tour contains a series of interactive steps that help users complete tasks.

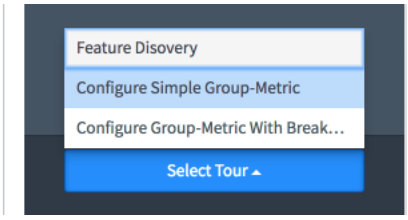
GUIDED TOURS AT GROUP METRIC LEVEL

Guided Tours at Group Metric level help users onboard easily and get trained on the features available.

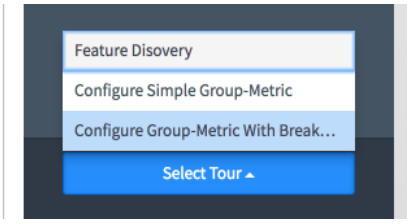
Guided Tours can be launched from 'Toggle Help Sidebar' -> 'Select Tour' as shown below.



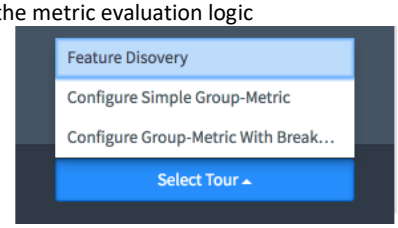
- On create of Group Metric
  - *Configure Simple Group Metric:* Tour guides through steps need to configure simple Group Metric without breakdown



  - *Configure Group Metric With Breakdowns:* Tour guides through steps needed to configure Group Metric with breakdowns.

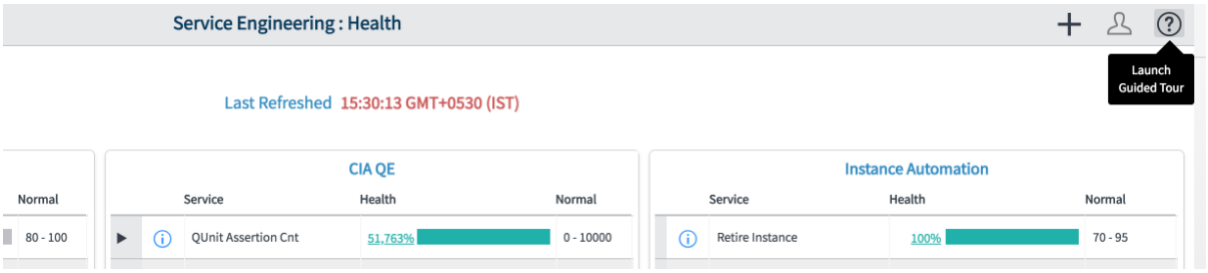


- On edit of Group Metric
  - *Feature Discovery: :* Tour helps to get trained on available features at Group Metric level to validate and test the metric evaluation logic



GUIDED TOURS ON HEALTH DASHBOARD

- Guided Tours on Health Dashboard help users get trained on the features available on Dashboard.
- Launch guided tour from the help icon available on the dashboard as shown below.



STANDARD ONBOARDING FLOW

1. Create your group/ service
   - Browse to ***SUHA > GROUPS***
   - If your Group/Service is not defined, create a new group
   - Make sure we assign the group/service in the right hierarchy (parent)
2. Define the Health Group Metric Definition (If not already defined)

- Browse to *SUHA > ONBOARD > CREATE GROUP METRIC*
- Select the service and **'% Health'** metric as the metric
- Define the health metric logic as any other group metric definition and set it active.
- To add your service to the health dashboard, click on the form link ***'Add To Health Dashboard'.***
- Go to *SUHA > DASHBOARD > SERVICE ENGINEERING*. This opens up the Health Dashboard
- Verify that we see a list widget for the parent group, and a row for the service we just on boarded (If not visible immediately, wait for 5 minutes)
- This widget on the health dashboard, is going to be the entry point to the **Service Level & Group Level Dashboards**

3. Define other metrics for your service
    - Check if any of the existing metric definitions suit our needs, if not create a new one
    - Define a Group Metric Definition for all the metrics
4. Define the Service Level Dashboard Configuration, to add this metric to your service dashboard. You can skip this step, if you do not wish to add this metric to the service level dashboard
    - Click on the UI action button ***'Add to Service Dashboard'***, this will pop up a Dashboard configuration form and add this metric to the metrics list. Validate and submit the form. Other way is to manually edit/create a Dashboard Configuration record, follow the steps below to do the same. **Using the UI action is the recommended way.**
    - Browse to *SUHA > ONBOARD > DASHBOARD CONFIGURATION*
    - Check if a configuration exists for the **'service'** and dashboard **'SE.Service Level View'**, if not create a new configuration
    - Select the metrics we want to see on the Service Level Dashboard for that service in the dashboard configuration
    - Browse to the service entry on the Health Dashboard, click on the **'i'** next to the service name. This takes us to the Service Level Dashboard
    - Verify if all the configured metrics exist on the dashboard
5. Define the Group Level Dashboard Configuration
    - Exactly similar to defining the Service Level Dashboard Configuration, but there is no UI action to perform this task. The only change is to select the dashboard **'SE.Group Level View'**

6. Create a new update set and set it as your current update set
7. Browse to *SUHA > CONFIGURATION > MY GROUP METRIC DEFINITIONS*
8. Select the group metric definitions you have just created/updated and wish to move to production
9. From the Actions list select, 'Gather Related Updates'
10. Move this update set from DCDev to DCTest to DCProd

GUIDELINES

SUHA is a framework and allows a lot of flexibility and leaves it to the discretion of the user to do the right configuration. Since, we are dealing with metrics, it is extremely important to configure it correctly. Here is a set of few guidelines, that might be of help.

- '% Health' is a real-time metric, the evaluation logic should be such that it describes the current health of the service at any given point in time
- Make sure that the '% Health' evaluation logic is robust enough to not return null in case your data source does not find any records. Or try to modify the data source such that it returns some records in most of the cases
- Data Sources which are used for Daily group metrics, should query for data for yesterday
- Daily group metrics are collected every day at 00:00 PST
- Data sources should query the minimal data required for your need. Think about what the optimal and accurate logic would be to define the group metric and query the data accordingly
- Use Aggregate data sources whenever possible.

For any feedbacks or assistance feel free to reach out at CloudAnalytics@servicenow.com