Track server errors with custom metrics and alerts

Published: 2018-11-09

While ExtraHop offers over 4,500 built-in metrics, there are many situations in which it is more effective to track network issues with a custom metric. For example, while built-in metrics show you issues with HTTP responses and requests, a custom metric can identify 500-level server errors. These types of errors can indicate gateway issues, an overloaded server, or configuration issues.

In this walkthrough, you will learn how to write a trigger to collect server error data as custom metrics and how to create an alert that only sends an email notification when those specific errors occur. Then, you will be able to answer the following types of questions about server errors on your network:

- Are my customers receiving 500-level server errors?
- What error codes are seen?
- When did the errors occur?
- What URI was the customer attempting to access?
- What is the IP address of the client and server affected in the transaction?

Prerequisites

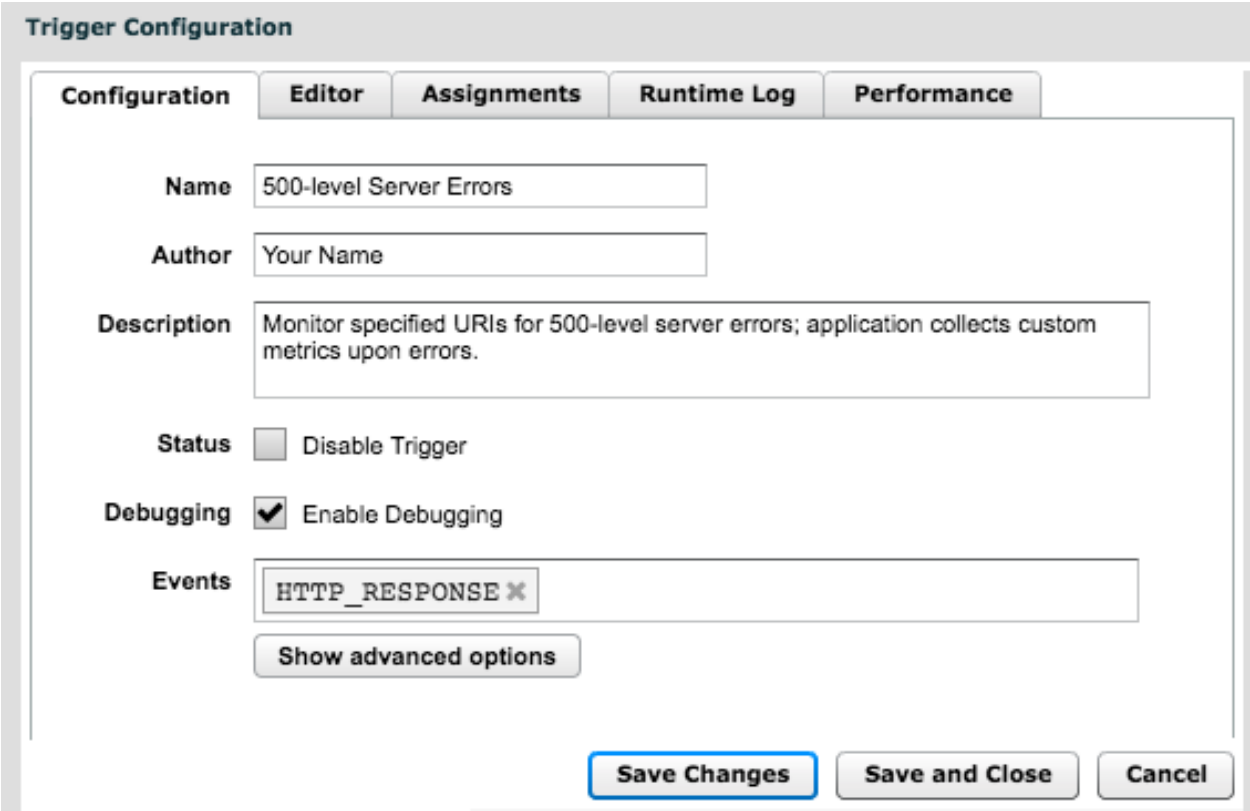
- You must have access to an ExtraHop Discover appliance with a user account that has unlimited privileges.
- Your Discover appliance must have network data with web server traffic.
- Your Discover appliance must be configured to send email notifications before you can send alert emails.
- Familiarize yourself with the concepts in this walkthrough by reading Triggers and Alerts.
- Familiarize yourself with the processes of creating triggers by reading Build a trigger.
- It is helpful to have basic JavaScript knowledge.

Write a trigger to collect error data from custom metrics

First, let's create a trigger that monitors certain URIs for 500-level server errors. When errors occur, the trigger collects data such as error codes and server and client IP addresses and commits that data as custom metrics to an application.

1. Log into the Web UI of a Discover appliance.
2. Click the System Settings icon, and then click **Triggers**.
3. Click **New** to open the Trigger Configuration window.
4. In the **Name** field, type a name for the trigger. For this walkthrough, type `500-level Server Errors`.
5. In the **Description** field, type information about the trigger. For this walkthrough, type `Monitor specified URIs for 500-level server errors; application collects custom metrics upon errors`.
6. Click **Enable Debugging**.
7. Click in the **Events** field and select **HTTP_RESPONSE** from the list.

   The following figure displays the trigger settings we configured above:



8. Click the **Editor** tab.
9. In the Trigger Script editor, copy and paste the following code:

```
// VARIABLES TO EDIT //

// Edit this array with host, URI, or host/URI combinations to monitor
var CHECK_URI_LIST = [
'Check.Host.Only.com',
'/Check/URI/Only',
'Check.Host.com/And/Check/Uri',
];

// Edit this array with client IP addresses to ignore
var IGNORE_CLIENT_IP = [
'180.57.175.147',
'172.16.156.130'
];

// END VARIABLES TO EDIT //

// DO NOT EDIT REMAINDER OF SCRIPT //

// If client IP is in array above, end process
if (IGNORE_CLIENT_IP.indexOf(Flow.client.ipaddr.toString()) > -1){
//debug ('Ignoring client IP: ' + Flow.client.ipaddr);
return}

// If URI is empty, end process
var uri = HTTP.uri || HTTP.host;
if (uri === null){//debug ('No URI Found, ending');
return}
```

```
// If URI/hostname does not match array above, end process
var matches = CHECK_URI_LIST.filter(condition => uri.indexOf(condition) >
 -1);
if (matches.length === 0) {
//debug ('URI or host did not match: ' + uri);
return}

var app = "HTTP Server Errors",
code = HTTP.statusCode,
client = Flow.client.ipaddr.toString(),
server = Flow.server.ipaddr.toString(),
detail = 'Code: ' + code ' || Server: ' + server + ' || Client: ' +
 client + ' || URI: '
          + uri;

if (code < 500 || code > 600) {return}

var code = HTTP.statusCode.toString();

// If URI matches and is a 5xx error, commit custom metrics
// to the application, which is created upon the initial event
Application(app).metricAddCount('HTTP_error', 1);
Application(app).metricAddDetailCount('HTTP_error_statusCode', code, 1);
Application(app).metricAddDetailCount('HTTP_error_uri', uri, 1);
Application(app).metricAddDetailCount('HTTP_error_serverIP', server, 1);
Application(app).metricAddDetailCount('HTTP_error_clientIP', client, 1);
Application(app).metricAddDetailCount('HTTP_error_allDetail', detail, 1);
debug ('Detail: ' + detail);
```

10. At the `CHECK_URI_LIST` variable in the script, replace the URI examples with the hosts, URIs, and host-URI combinations that you want the trigger to monitor.
11. Replace 180.57.175.147 and 172.16.156.130 with the IP addresses you want the trigger to ignore in the `IGNORE_CLIENT_IP` variable.
12. Click **Save and Close**.
    The trigger editor validates the syntax of your script. Invalid actions, syntax errors, or deprecated elements will prevent you from saving the trigger until you fix the code or disable syntax validation.

Assign the trigger to a device

Before the trigger can run, it must be assigned to at least one device. In this step, we will assign the trigger to a device group named HTTP Servers.

By creating a device group that contains only the devices that support traffic to the URIs monitored by the trigger, we eliminate unnecessary performance impact on the system. Triggers should only be assigned to specific, relevant devices.

1. Click **Metrics** from the top menu.
2. Click **Device Groups**, and then select the **HTTP Servers** checkbox.
3. Click **Assign Trigger** from the top of the page to open a list of triggers that are eligible for assignment.
4. Select the **500-level Server Errors** trigger we created in the previous section, and then click **Assign Triggers**.

Configure an alert to track a custom metric

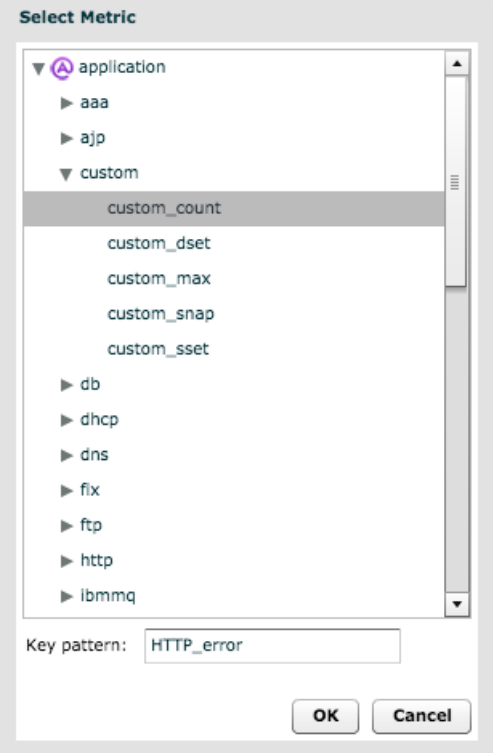
Next, let's configure settings for an alert that will track the custom `HTTP_error` metric we created with the trigger. We'll also configure email notifications for each error occurrence.

Before you begin

The Discover appliance must be configured for email notifications.

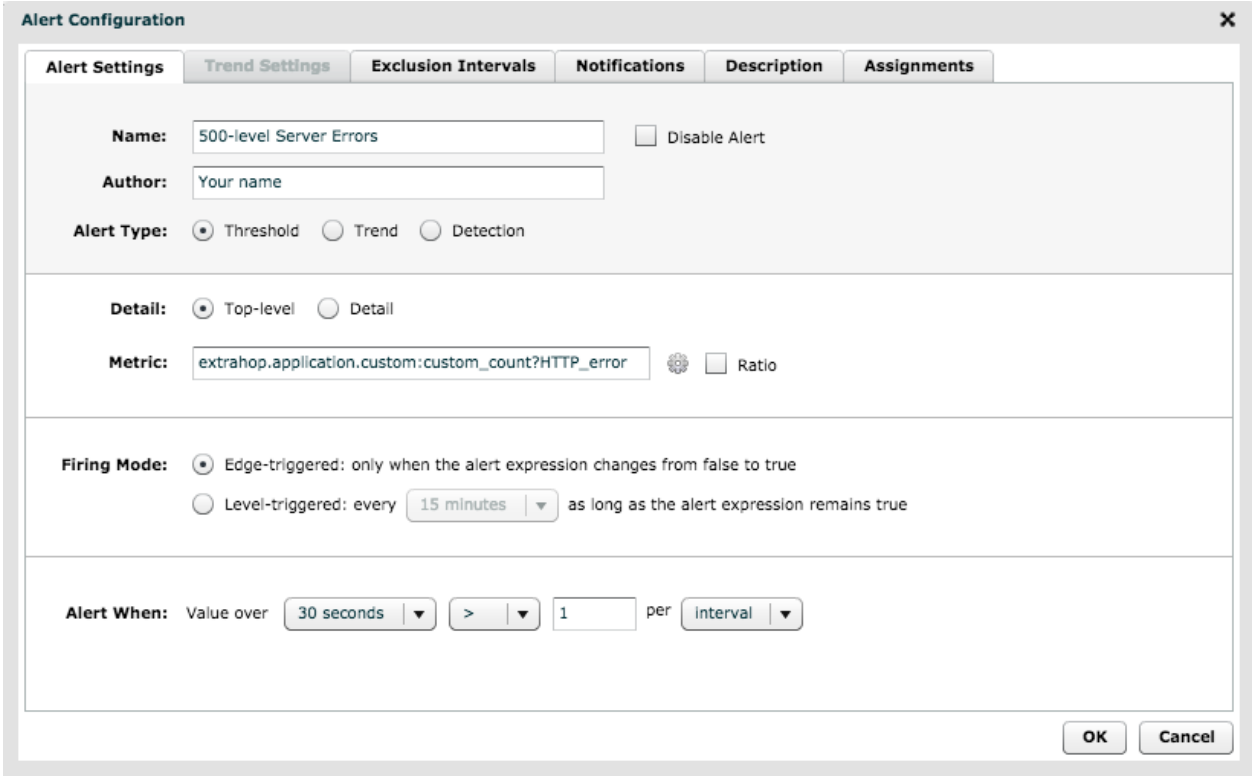
1. Click the System Settings icon and then click **Alerts**.
2. Click **New** and then configure the following alert settings:
   a) In the **Name** field, type a name for the alert. In this walkthrough, type `500-level Server Errors`.
   b) In the Alert Type section, select **Threshold** to issue an alert when the tracked metric event occurs.
   c) In the Detail section, select **Top-level** and then click the Select metric icon.
   d) In the Select Metric window, click **application** > **custom** > **custom_count**.
   e) In the **Key pattern** field, type `HTTP_error` and then click **OK**.

      This key pattern is the name of the custom metric in the trigger script that counts each 500-level error.



   f) In the Firing Mode section, select **Edge-Triggered** to generate an alert only once when the tracked metric event occurs.
   g) In the Alert When section, specify the following expression to generate an alert if the tracked metric event occurs more than once in a 30-second time period:`Value over 30 seconds > 1`

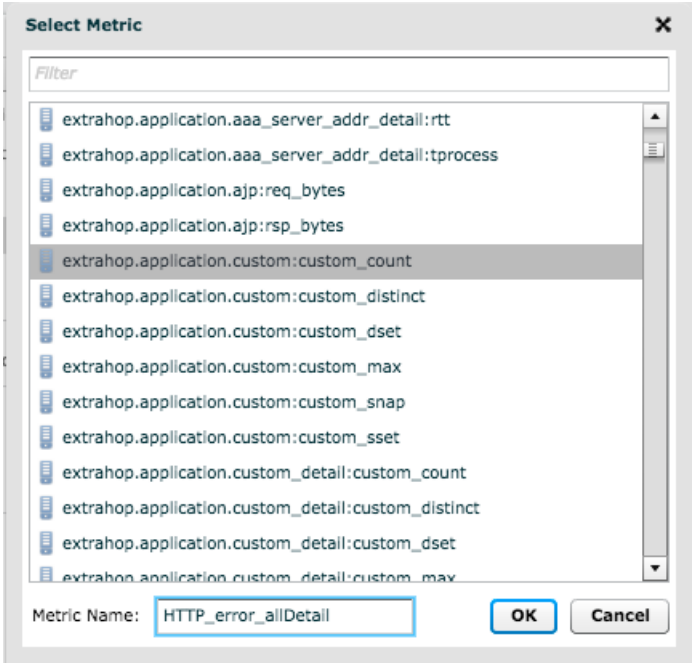
The following figure displays the alert email notification settings we configured above:



3. Click the **Description** tab and then type a description of the alert. For this walkthrough, type `Alert issued when a 500-level server error occurs on monitored URIs.`
4. Click the **Notifications** tab and then configure the following email settings:
   a) From the **Severity** drop-down menu, select **Error**.
   b) In the Additional email addresses section, type an email address that should receive alert notifications.

      Tip: The Email notification groups section displays all email groups configured in the Admin UI. You can select one or more groups that should receive alert notifications.

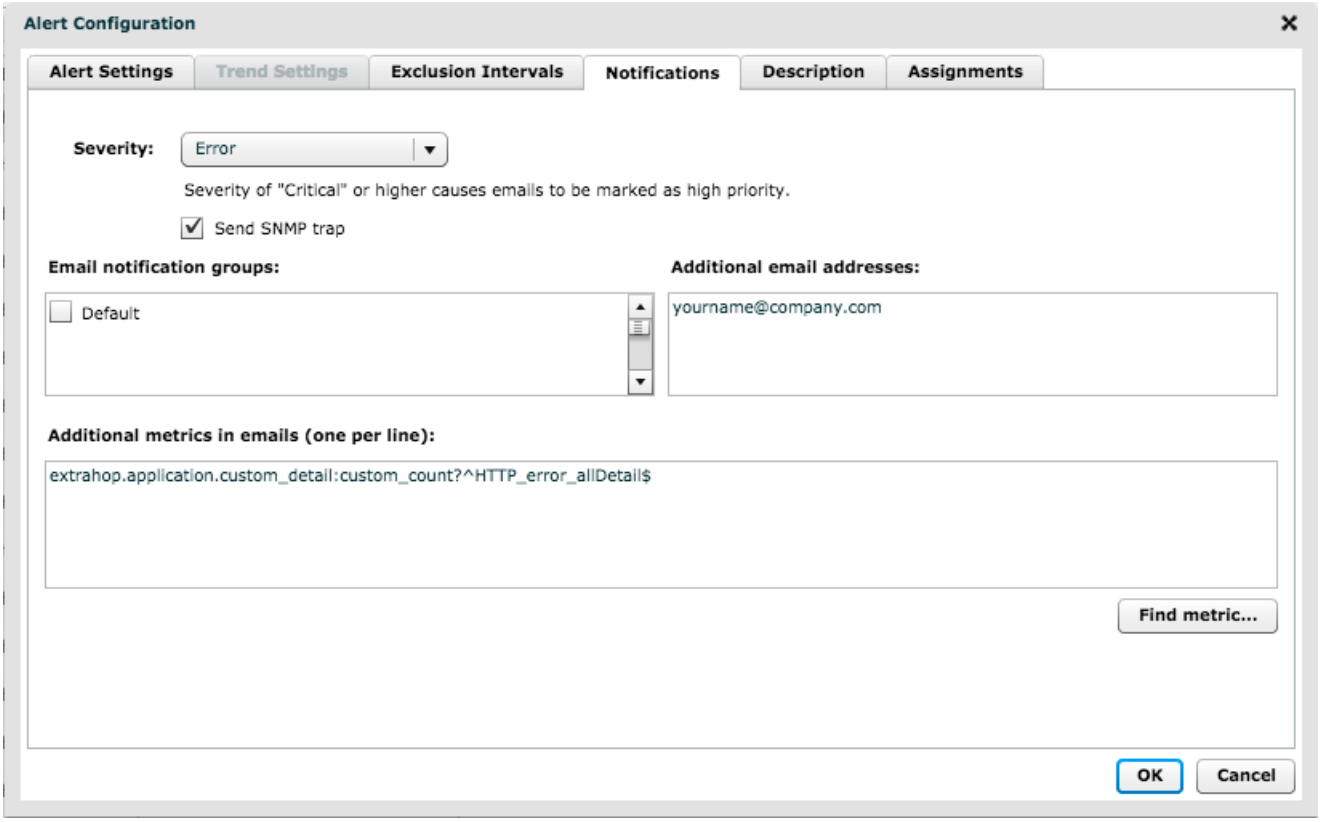
   c) In the Additional metrics in emails section, click **Find metric**.
   d) Scroll down the page and select **extrahop.application.custom_detail:custom_count**.



e) In the **Metric Name** field, type `HTTP_error_allDetail`, and then click **OK**.

This metric name is the name of the custom metric in the trigger script that collects all of the details about the error event, including the code number, the URI, and the server client IP addresses on which the error occurred. The value of this metric is included in the alert email for each event that meets the alert expression.

The following figure displays the alert email notification settings we configured above:



5. Click **OK**.

Assign the alert configuration to a source

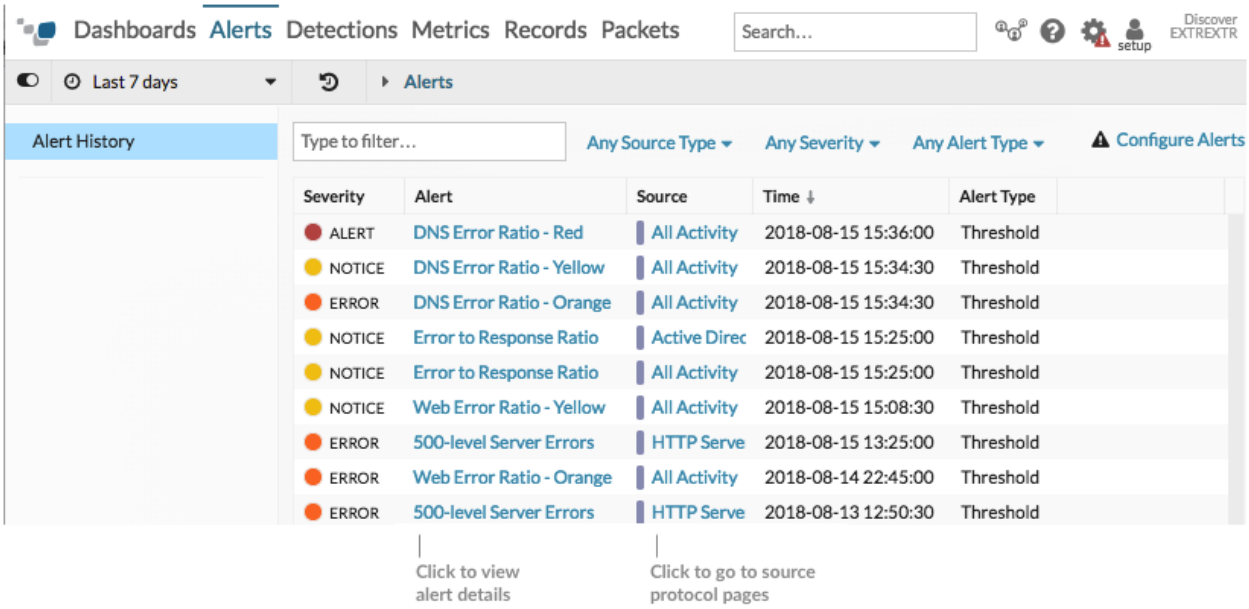
Similar to triggers, the system does not generate alerts until the alert configuration is assigned to at least one source. In this step, we will assign the alert to the application we created with the trigger script. The application collects the custom metrics we want the alert to track and is named HTTP Server Errors.

1. Click **Metrics** from the top menu.
2. Click **Applications**, and then select the **HTTP Server Errors** checkbox.
3. Click **Assign Alert** from the top of the page to open a list of alert configurations that are eligible for assignment.
4. Select the **500-level Server Errors** alert, and then click **Assign Alerts**.

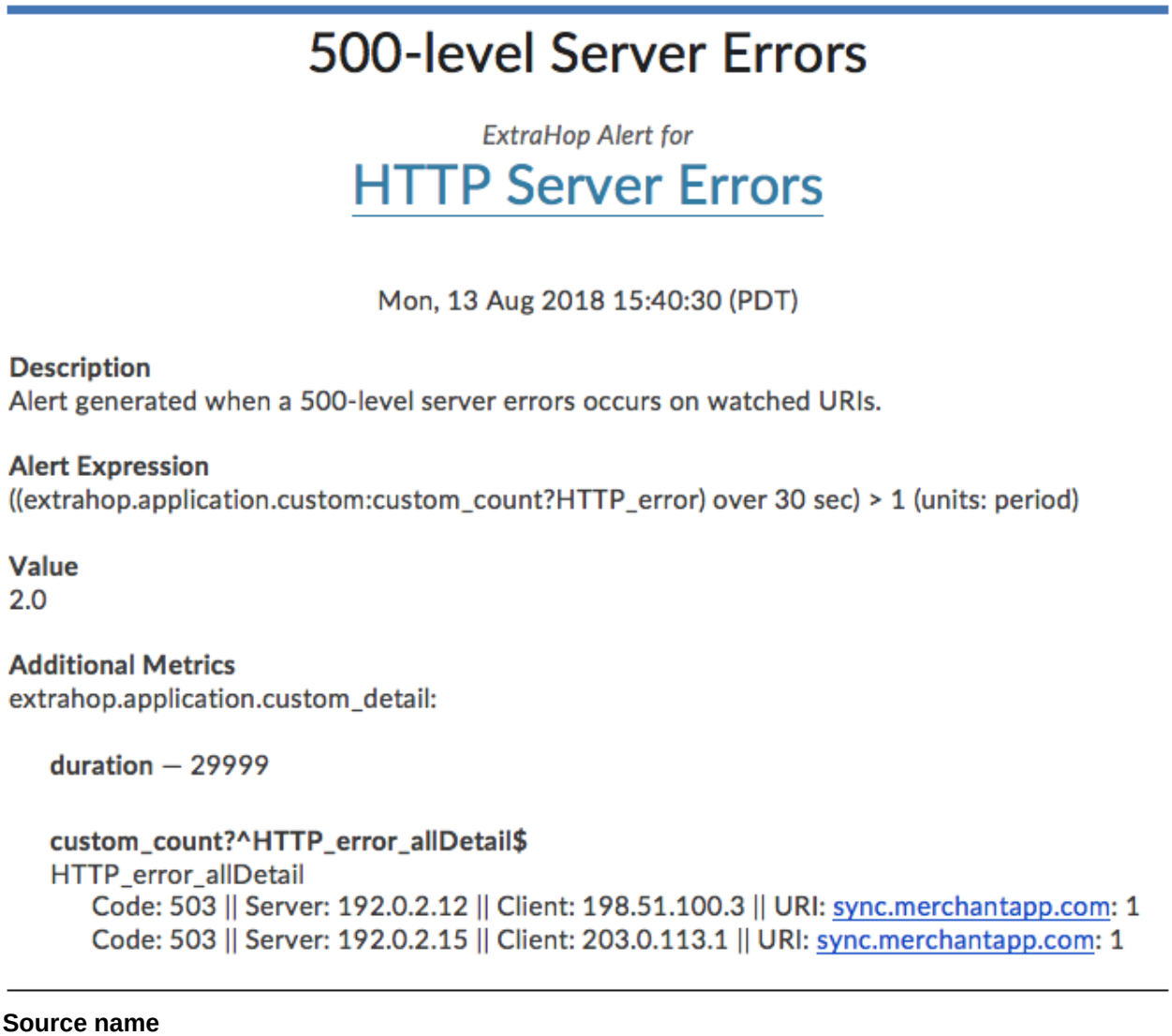
Check the Alert History and view email notifications

Now that we've configured the alert and assigned it to a source, we can check if the alert has issued any entries.

Click **Alerts** from the top menu to see the Alert History. All alerts that were issued during the selected time period are listed, similar to the following figure:



When an alert is generated, a notification is sent to the specified email recipients The following email example shows that two HTTP error events occurred that met the conditions we set in the alert expression and provides additional information that helps us investigate the source of the errors:



500-level Server Errors

ExtraHop Alert for

HTTP Server Errors

Mon, 13 Aug 2018 15:40:30 (PDT)

Description
Alert generated when a 500-level server errors occurs on watched URIs.

Alert Expression
((extrahop.application.custom:custom_count?HTTP_error) over 30 sec) > 1 (units: period)

Value
2.0

Additional Metrics
extrahop.application.custom_detail:

duration — 29999

custom_count?^HTTP_error_allDetail$
HTTP_error_allDetail
Code: 503 || Server: 192.0.2.12 || Client: 198.51.100.3 || URI: sync.merchantapp.com: 1
Code: 503 || Server: 192.0.2.15 || Client: 203.0.113.1 || URI: sync.merchantapp.com: 1

Source name

Displays the name of the source that the alert is assigned to, which is the HTTP Server Errors application in this example. Click the source name to go to the HTTP protocol page for the application.

Value

Provides the number of times the metric tracked by the alert met the conditions specified by the alert expression.

Additional Metrics

Displays the name of the additional tracked metric and provides the following details about the two HTTP errors that occurred:

- Error code number
- IP address of the server on which the error occurred
- IP address of the client that made the request
- URI that was requested and returned the error

In our example, we see that there were two 503 errors returned for the same URI across two different server IP addresses. A 503 status code might indicate an overloaded server that requires more CPU or memory resources to handle requests. By knowing the affected IP addresses you can immediately investigate potential problems on the listed servers.

Next steps

- Create charts to monitor your custom metrics on a dashboard or protocol page.
- Configure a trend alert to only issue alerts when server errors trend instead of for each error occurrence.
- Add an exclusion interval to your alert to suppress alerts during times when errors are expected.