

Alibaba Cloud

Web应用防火墙 Monitoring and Alarm

Document Version: 20211221

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.
2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.
3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.
4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).
5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.
6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions









Style	Description	Example
 Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 Danger: Resetting will result in the loss of user configuration data.
 Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 Warning: Restarting will cause business interruption. About 10 minutes are required to restart an instance.
 Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	 Notice: If the weight is set to 0, the server no longer receives new requests.
 Note	A note indicates supplemental instructions, best practices, tips, and other content.	 Note: You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click Settings> Network> Set network type .
Bold	Bold formatting is used for buttons, menus, page names, and other UI elements.	Click OK .
Courier font	Courier font is used for commands	Run the <code>cd /d C:/window</code> command to enter the Windows system folder.
<i>Italic</i>	Italic formatting is used for parameters and variables.	<code>bae log list --instanceid</code> <i>Instance_ID</i>
[] or [a b]	This format is used for an optional value, where only one item can be selected.	<code>ipconfig [-all -t]</code>
{ } or {a b}	This format is used for a required value, where only one item can be selected.	<code>switch {active stand}</code>

Table of Contents

1.Configure WAF alerting	05
2.Supported monitoring types and service metrics	08
3.Use CloudMonitor to configure monitoring and alerting for WA... ..	12
4.Configure alarms in Log Service	18
4.1. Overview	18
4.2. Step 1: Create a WAF log analysis dashboard	19
4.3. Step 2: Configure log charts	20
4.4. Step 3: Configure log alerts	23
4.5. Examples of alert configurations based on WAF logs	30
4.6. Common monitoring metrics	56
4.7. Query statements	59

1. Configure WAF alerting

After you add your website to Web Application Firewall (WAF), you can use the alert settings feature to configure alerting. After you configure alerts in the alert settings, WAF sends alert notifications in real time when WAF detects attacks and unusual traffic. This helps you understand the security posture of your website in a timely manner.

Prerequisites

- Your website is added to WAF. For more information, see [Tutorial](#).
- (Optional) The Log Service for WAF feature is enabled for your WAF instance. The log collection feature is enabled for the domain name of your website. For more information, see [Enable Log Service for WAF](#) and [Step 2: Enable the log collection feature](#).

By default, WAF allows you to configure monitoring and alert rules by using Alibaba Cloud CloudMonitor. In the CloudMonitor console, you can configure monitoring and alert rules for the WAF metrics that are supported by CloudMonitor and the attack events that are detected by WAF. For more information about the WAF metrics, see [WAF metrics](#). If the WAF metrics that are supported by CloudMonitor do not meet your business requirements, you can use the Log Service for WAF feature to configure the alert settings for WAF.

If you want to use the Log Service for WAF feature to configure the alert settings, the preceding prerequisites must be met.

Procedure

1. Log on to the [Web Application Firewall console](#).
2. In the top navigation bar, select the resource group and region to which the WAF instance belongs. The region can be **Mainland China** or **International**.
3. In the left-side navigation pane, choose **System Management > Alarm Settings**.
4. On the **Alarm Settings** page, configure notification methods based on the type of attack events.

Alarm Settings		
WAF provides basic statistics of protected domains and attacks for CloudMonitor. You can customize alert rules based on domain data and configure alert contacts and methods for attacks. This makes it easy to perform maintenance operations. For more information about how to configure alert rules, click here . For more information about best practices, click here .		
Attack Alerts/Alert Settings <small>For more alert methods, see CloudMonitor notifications.</small>		
Event	Description	For more alert methods,
Web Attacks	Triggers alerts when WAF detects a large number of Web Attacks within a short period of time. Alert details include the attack time, the number of attack requests, the number of blocked attack requests, and attack traffic analysis.	CloudMonitor Notifications
HTTP Flood Attack Events	Triggers alerts when WAF detects HTTP flood attacks. Alert details include the attack time, number of attack requests, the number of blocked attack requests, and attack traffic analysis.	CloudMonitor Notifications
ACL-based Attacks	Triggers alerts when WAF detects a large number of ACL-based attacks within a short period of time. Alert details include the attack time, the number of attack requests, the number of blocked attack requests, and attack traffic analysis.	CloudMonitor Notifications
Scan Attacks	Triggers alerts when WAF detects scan attacks based on threat intelligence. Alert details include the attack time, the number of attack requests, the number of blocked attack requests, and attack traffic analysis.	CloudMonitor Notifications
Traffic Volume Monitoring	You can use metrics that monitor business traffic to customize business traffic monitoring settings based on your business requirements. This allows you to create custom events, and specify alert methods and recipients.	CloudMonitor Notifications Log Service Configurations
Abnormal Traffic Monitoring	You can use metrics that monitor abnormal business traffic to create custom events based on business requirements, and specify alert methods and recipients. This helps you collect data on abnormal business traffic.	CloudMonitor Notifications Log Service Configurations
Custom Attack Monitoring	You can use metrics that monitor business traffic volume to create custom events based on your actual needs, and specify alert methods and recipients. This helps you collect data on the traffic volume of blocked attacks.	CloudMonitor Notifications Log Service Configurations
Bandwidth Threshold Exceeded	Your service bandwidth exceeds the default bandwidth provided by WAF. While in this state, WAF cannot guarantee the SLA, and exceptions such as packet loss may occur. Purchase extra bandwidth at the earliest opportunity.	CloudMonitor Notifications View Alerts Generated During Last 30 Days
QPS Threshold Exceeded	Your business exceeds the maximum QPS supported by WAF. While in this state, WAF cannot guarantee the SLA, and exceptions such as packet loss may occur. Upgrade WAF specifications at the earliest opportunity.	CloudMonitor Notifications View Alerts Generated During Last 30 Days

Notification method	Description
CloudMonitor Notifications Create alert rules for different types of events by using the alerting feature provided by CloudMonitor.	<p>CloudMonitor is a service that monitors resources and Internet applications. For more information, see What is CloudMonitor. CloudMonitor provides the alerting feature to monitor events and metrics of cloud services. For more information about the feature, see Overview.</p> <p>If you use the CloudMonitor Notifications method, you can create alert rules for all types of events that are listed on the Alarm Settings page. The event types are Web Attacks, HTTP Flood Attack Events, ACL-based Attacks, Scan Attacks, Traffic Volume Monitoring, Abnormal Traffic Monitoring, Custom Attack Monitoring, Bandwidth Threshold Exceeded, and QPS Threshold Exceeded.</p> <p>When you click CloudMonitor Notifications, you are redirected to the Alert Rules tab of the CloudMonitor console. You can configure alert rules on the Alert Rules tab. For more information, see Use CloudMonitor to configure monitoring and alerting for WAF.</p>
Log Service Configurations Create alert rules for different types of events by using the alerting feature provided by Log Service for WAF.	<p>Log Service for WAF allows you to collect and store the logs for requests that are sent to the domain name of your website that is added to WAF. Then, you can query and analyze the logs. You can use query and analysis results to customize alert rules for WAF metrics based on your business requirements.</p> <p>If you use the Log Service Configurations method, you can create alert rules for different combinations of metrics. This method provides high flexibility and is suitable for business scenarios in which you want to customize alert rules. Compared with the previous method, this method is more complex to use.</p> <p>When you click Log Service Configurations, you are redirected to the Log Service page. You can query and analyze WAF logs and customize alert rules on the Log Service page. For more information about how to configure WAF log alerting, see Configure an alert in Log Service. For more information about the examples on how to configure WAF log alerting, see Overview.</p>

Notification method	Description												
<h3>View Alerts Generated During Last 30 Days</h3> <p>View details about alerts that are sent because the actual traffic exceeds the specified WAF service bandwidth or QPS threshold value.</p>	<p>If the actual traffic of your website exceeds the service bandwidth or QPS threshold value of your WAF instance, the system automatically displays an alert notification in the upper part of the WAF console. The system also sends the alert notification by email to the contact that you specify for your Alibaba Cloud account.</p> <div><div><div>Your bandwidth or QPS in the last 30 days has exceeded the specifications that you purchased. If this lasts for a long time, your service is throttled, and normal access is affected. We recommend that you use the bandwidth upgrade package or upgrade your instance edition to increase your bandwidth or QPS.</div><div>Upgrade Now View details</div></div><div><div>Overview</div><div>Meet Experts Version <div></div> Product Updates <div>1</div> Renew Auto Renew Upgrade</div></div><div>Overview</div></div> <p>On the Alarm Settings page, you can view the details about the alerts that are generated within the last 30 days. You can click View Alerts Generated During 30 Days for Bandwidth Threshold Exceeded or QPS Threshold Exceeded to view alert details.</p> <div><div>Details of Bandwidth Threshold Exceeded</div><div><table><tr><th>Date</th><th>Maximum Value</th><th>Current Specifications</th><th>Type</th></tr><tr><td>Nov 2, 2021, 00:00:00 - Nov 2, 2021, 23:59:00</td><td>53 Mbps</td><td>20 Mbps</td><td>Bandwidth</td></tr><tr><td>Nov 1, 2021, 00:00:00 - Nov 1, 2021, 23:59:00</td><td>53 Mbps</td><td>20 Mbps</td><td>Bandwidth</td></tr></table></div></div> <p>For more information, see WAF service bandwidth.</p>	Date	Maximum Value	Current Specifications	Type	Nov 2, 2021, 00:00:00 - Nov 2, 2021, 23:59:00	53 Mbps	20 Mbps	Bandwidth	Nov 1, 2021, 00:00:00 - Nov 1, 2021, 23:59:00	53 Mbps	20 Mbps	Bandwidth
Date	Maximum Value	Current Specifications	Type										
Nov 2, 2021, 00:00:00 - Nov 2, 2021, 23:59:00	53 Mbps	20 Mbps	Bandwidth										
Nov 1, 2021, 00:00:00 - Nov 1, 2021, 23:59:00	53 Mbps	20 Mbps	Bandwidth										

2.Supported monitoring types and service metrics

You can use CloudMonitor to configure monitoring and alerting for website services. This topic describes the site metrics, attack event types, and Web Application Firewall (WAF) service metrics that are supported by CloudMonitor.

Context

WAF integrates CloudMonitor. CloudMonitor allows you to configure monitoring and alerting for site metrics, attack events that occurred on domain names that are added to WAF, and WAF service metrics for the domain names.

CloudMonitor is a service that monitors Internet applications and Alibaba Cloud resources. CloudMonitor sends you notifications when alerts are triggered. You can configure alert rules. CloudMonitor sends alert notifications to specific contacts by using email or by using the alert callback feature when CloudMonitor detects system events. This way, you can be notified of critical events in real time after they are generated and can handle the events in an automated online O&M process.

Site metrics supported

CloudMonitor can simulate the detection requests of real users, monitor access to your service sites from all cities and provinces in China, and detect exceptions in real time.


The following table lists the metrics that site monitoring supports. We recommend that you configure all supported metrics when you use the site monitoring feature.

Metric	Level	Description	Configuration method
Elastic Compute Service (ECS) performance monitoring	Major	Monitor the CPU utilization, memory usage, disk space usage, and bandwidth usage of ECS instances.	Configure alerts for an ECS instance
Server Load Balancer (SLB) performance monitoring	Major	Monitor the number of connections, bandwidth usage, and packets per second (PPS) of SLB instances.	Configure alert rules for SLB instances
Object Storage Service (OSS) sandbox status monitoring	Major	Monitor the OSS sandbox to view the status of the OSS service.	Overview

Metric	Level	Description	Configuration method
HTTP/HTTPS	Major	Send HTTP or HTTPS requests to a specific URL or IP address to monitor the URL or IP address.	<p>Site monitoring is provided by CloudMonitor. The site monitoring feature does not involve WAF-related operations. You need only to log on to the CloudMonitor console by using your Alibaba Cloud account and perform the following operations:</p> <ul style="list-style-type: none"> • Create a site monitoring task • Modify a site monitoring task • View data related to a site monitoring task
PING	Major	Run Internet Control Message Protocol (ICMP) ping command for a specific URL or IP address to monitor the URL or IP address.	
TCP	Major	Send Transmission Control Protocol (TCP) requests to a specific port to monitor the port.	
UDP	Optional	Sends User Datagram Protocol (UDP) requests to a specific port to monitor the port.	
DNS	Optional	Send domain name system (DNS) requests to a specific domain to monitor the domain name.	
POP3	Optional	Send Post Office Protocol version 3 (POP3) requests to a specific URL or IP address to monitor the URL or IP address.	
SMTP	Optional	Send Simple Mail Transfer Protocol (SMTP) requests to a specific URL or IP address to monitor the URL or IP address.	
FTP	Optional	Send File Transfer Protocol (FTP) requests to a specific URL or IP address to monitor the URL or IP address.	

Attack events supported

CloudMonitor allows you to configure monitoring and alerting for web attacks, HTTP flood attacks, scan attacks, and unauthorized access control events on domain names that are added to WAF. You can select a notification method by which you want to receive alerts based on the severity level of events. The notification method includes text messages, emails, DingTalk, or the alert callback feature. For more information about how to configure monitoring and alerting for attack events, see [Configure monitoring and alerting for attack events](#).


 **Notice** Event monitoring takes effect only for domain names that are added to WAF. Before you can configure alert rules for a domain name, make sure that the domain name is added to WAF. For more information about how to add a domain name, see [Add a website](#).

The following table lists the supported attack events.

Event name	Description	Type	Status value	Event level
waf_event_aclattack	An access control event occurred.	acl	start and end	CRITICAL
waf_event_ccattack	An HTTP flood attack occurred.	cc	start and end	CRITICAL
waf_event_webattack	A web attack occurred.	web	start and end	CRITICAL
waf_event_webscan	A web scan attack occurred.	webscan	start and end	CRITICAL

WAF service metrics supported

CloudMonitor allows you to configure monitoring and alerting for WAF service metrics on domain names that are added to WAF. You can specify the method to identify exceptions on the service metrics and select a notification method by which you want to receive alerts, such as by using text messages, emails, DingTalk, or the alert callback feature. For more information about how to configure monitoring and alerting for the service metrics, see [Configure monitoring and alerting for metrics](#).

 **Notice** Service metric monitoring takes effect only for domain names that are added to WAF. Before you can configure alert rules for a domain name, make sure that the domain name is added to WAF. For more information about how to add a domain name, see [Add a website](#).

The following table lists the supported service metrics.

Metric	Dimension	Unit	Description	Remarks
4XX_ratio	Domain name	%	The percentage of the HTTP 4xx status codes per minute (405 excluded).	The value is displayed as a decimal number.
5XX_ratio	Domain name	%	The percentage of the HTTP 5xx status codes per minute.	The value is displayed as a decimal number.

Metric	Dimension	Unit	Description	Remarks
acl_blocks_5m	Domain name	Pieces (PCS)	The number of requests blocked by access control within the last five minutes.	None.
acl_rate_5m	Domain name	%	The percentage of requests blocked by access control within the last five minutes.	The value is displayed as a decimal number.
cc_blocks_5m	Domain name	PCS	The number of requests blocked by HTTP flood protection within the last five minutes.	None.
cc_rate_5m	Domain name	%	The percentage of requests blocked by HTTP flood protection within the last five minutes.	The value is displayed as a decimal number.
waf_blocks_5m	Domain name	PCS	The number of requests blocked by web intrusion prevention within the last five minutes.	None.
waf_rate_5m	Domain name	%	The percentage of requests blocked by web attack protection within the last five minutes.	The value is displayed as a decimal number.
QPS	Domain name		The queries per second.	None.
qps_ratio	Domain name	%	The minute-on-minute growth rate of QPS.	The value is displayed in percentage.
qps_ratio_down	Domain name	%	The minute-on-minute decrease rate of QPS.	The value is displayed in percentage.

3. Use CloudMonitor to configure monitoring and alerting for WAF

WAF is integrated with CloudMonitor. This allows you to configure alert notification rules for metrics supported by Web Application Firewall (WAF) and attack events detected by WAF in the CloudMonitor console. This topic describes how to use CloudMonitor to configure monitoring and alerting for WAF.

Prerequisites

The domain name of your website is added to WAF. For more information, see [Add a website](#).


Supported metrics and attack events

For more information about the WAF-related metrics and attack events that can be monitored by CloudMonitor, see [Supported monitoring types and service metrics](#).

Configure alert contacts

After you configure an alert contact, CloudMonitor sends notifications for the alerts that you configure to the contact. The alert contact must check the alert notifications in time and handle the alerts at the earliest opportunity.


- 1.
2. Create an alert contact.
 - i. In the left-side navigation pane, choose **Alerts > Alert Contacts**.
 - ii. On the **Alert Contacts** tab, click **Create Alert Contact**.
 - iii. In the **Set Alert Contact** panel, enter the contact information, drag the slider to complete verification, and then click **OK**.

 **Note** You must retain the default value **Automatic** of **Alert Notification Information Language**. **Automatic** indicates that CloudMonitor automatically determines the language for alert notifications based on the language that you use to create your Alibaba Cloud account.

- iv. Optional. Activate the email address and mobile phone number of the alert contact.

By default, the email address and mobile phone number of the alert contact are in the **Pending Activation** state. After the alert contact receives an email or a text message that contains the activation link, the alert contact must activate the email address and mobile phone number within 24 hours. Otherwise, the alert contact cannot receive alert notifications. After the email address and mobile phone number are activated, you can view the email address and mobile phone number in the alert contact list.

3. Create an alert group.

 **Notice** An alert contact must belong to an alert group. You can add one or more alert contacts to an alert group.

- i. On the **Alert Contact Group** tab, click **Create Alert Contact Group**.


- ii. In the **Create Alert Contact Group** panel, enter the group name, select alert contacts from the **Existing Contacts** section, and then add the alert contacts to the **Selected Contacts** section.
- iii. Click **Confirm**.


Configure monitoring and alerting for attack events

After you configure monitoring and alerting for attack events, CloudMonitor sends alert notifications based on the rules you configure when WAF detects attacks such as web and HTTP flood attacks. The rules cover the severities of attack events and the methods to receive alert notifications. For more information about the attack events that can be monitored by CloudMonitor, see [Attack events supported](#).

- 1.
2. Create an alert rule for attack events.
 - i. In the left-side navigation pane, choose **Alerts > Alert Rules**.
 - ii. Click the **Event Alert** tab.
 - iii. On the **Event Alert** tab, click **Create Event Alert**.
 - iv. In the **Create / Modify Event Alert** panel, configure the following parameters.

The following table describes the parameters that are used to create an alert rule for attack events.

Parameter	Description
Alert Rule Name	Enter the name of the alert rule. The name can be up to 30 characters in length and can contain letters, digits, and underscores (_).
Event Type	Select System Event .
Product Type	Select WAF from the drop-down list.
Event Type	Select All Types .
Event Level	Select the level of the attack event for which you want to receive alert notifications. Valid values: CRITICAL , WARN , and INFO .  Notice You can select multiple levels. If you select multiple levels, you must select CRITICAL .

Parameter	Description
Event Name	Select the type of the attack event for which you want to receive alert notifications. Valid values: <ul style="list-style-type: none">■ waf_event_aclattack■ waf_event_ccattack■ waf_event_webattack■ waf_event_webscan <div> Notice The event level for each of these event types is CRITICAL.</div>
Resource Range	Select All Resources .
Alert Type	Select Alert Notification and configure Contact Group and Notification Method. <ul style="list-style-type: none">■ Contact Group: Select an existing alert group. All contacts in the alert group can receive alert notifications.■ Notification Method: Select Info (Email ID+ DingTalk Robot). You can click Add to add more alert groups and notification methods. You can also configure other alert types, such as MNS queue , Function service , URL callback , and Log Service . For more information, see Create an event-triggered alert rule .

v. Click **OK**.

After you configure the alert rule for attack events, the contacts in the alert rule can receive alert notifications when specific attacks are detected on the domain names added to WAF.

You can also query recent attack events detected by WAF in the CloudMonitor console.


Configure monitoring and alerting for metrics

After you configure monitoring and alerting for metrics, CloudMonitor sends alert notifications to the contacts in the alert rules that you configure. Alerts are triggered when WAF detects exceptions in the metrics of domain names that are added to WAF. The exceptions include minute-to-minute decrease in queries per second (QPS) and surges in error codes and blocked attacks. For more information about the metrics that can be monitored by CloudMonitor, see [WAF service metrics supported](#).

- 1.
2. Create an alert rule for metrics.
 - i. In the left-side navigation pane, choose **Alerts > Alert Rules**.
 - ii. On the **Threshold Value Alert** tab, click **Create Alert Rule**.
 - iii. On the **Create Alert Rule** page, configure the following parameters.

Parameter	Description
-----------	-------------

Parameter	Description
Product	The service that you want to monitor. Select WAF from the drop-down list.
Resource Range	The scope of the domain names that you want to monitor. Valid values: <ul style="list-style-type: none">■ All Resources: monitors all the domain names that are added to WAF. If the alert rule is triggered for one of the domain names, an alert notification is sent.■ Instance: monitors specific domain names. An alert notification is sent only when the alert rule is triggered for all the selected domain names.
Region	The region where the WAF instance resides. This parameter is required only if you select Instance from the Resource Range drop-down list. Valid values: <ul style="list-style-type: none">■ China East 1 (Hangzhou): specifies a WAF instance in mainland China.■ Asia Pacific SE 1 (Singapore): specifies a WAF instance outside mainland China.
Instance	The ID of the WAF instance. This parameter is required only if you select Instance from the Resource Range drop-down list. After you configure Region , the ID of the WAF instance in the selected region automatically appears. You do not need to modify this parameter. If no WAF instances are purchased in the selected region, No Data appears.
domain	The domain name that you want to monitor. This parameter is required only if you select Instance from the Resource Range drop-down list. Select the domain name that you want to monitor from the domain names that are added to the WAF instance. You can select multiple domain names.
Alert Rule	The name of the alert rule.

Parameter	Description
Rule Description	<p>The content of the alert rule. This parameter defines the condition that triggers the alert rule.</p> <div>  Note We recommend that you configure alert thresholds for different metrics based on your business requirements. For more information about the metrics, see WAF service metrics supported. A low threshold may trigger frequent alerts and affect user experience. A high threshold may not provide you with sufficient time to respond to exceptions detected in metrics. </div> <p>Example of an alert rule:</p> <p>If you configure the following rule, CloudMonitor determines whether to trigger an alert based on the QPS detected in three consecutive cycles. CloudMonitor reports a data point at an interval of 60s. A total of 15 data points are reported in the three consecutive cycles. If the maximum QPS among the reported data points is greater than 200, an alert is triggered.</p> <div> <div>Alarm Rule: <input type="text" value="doc-test"/></div> <div> Rule Description: <input type="text" value="qps"/> <input type="text" value="5Minute cycle"/> <input type="text" value="Continue for 3"/> <input type="text" value="Max. Value"/> <input type="text" value=">="/> <input type="text" value="200"/> <input type="text" value="counts"/> </div> </div> <p>You can click Add Alert Rule to create more rules. You must configure Alert Rule and Rule Description for each rule.</p>
Mute for	The interval of re-sending the notification for an alert before the alert is cleared. The minimum value is 5 minutes, and the maximum value is 24 hours.
Effective Period	The time period during which the alert rule remains effective. CloudMonitor sends alert notifications within the effective period and only records alerts beyond the effective period.
Notification Contact	The alert group that receives alert notifications.
Notification Methods	The method that is used to send alert notifications. Different levels of alerts are sent by using different methods. Alert levels are Critical, Warning, and Info. Valid values: Email + DingTalk (Info).
Auto Scaling	If you select Auto Scaling, no additional configurations are required. After you select a scaling rule, it is triggered when an alert is generated.
Log Service	<p>If you select Log Service, the alert information is written to Log Service when an alert is generated. You must also configure Region, Project, and Logstore.</p> <p>For more information about how to create a project and a Logstore, see Quick start.</p>

Parameter	Description
Email Subject	The subject of the alert notification email. By default, the email subject is in the format of service name + metric name + instance ID.
Email Remark	The additional information that you want to include in the alert notification email.
HTTP CallBack	The URL to which CloudMonitor sends alert notifications by using HTTP POST. You can enter only an HTTP URL.

iv. Click **Confirm**. The alert rule is created.

If WAF-related metrics meet the conditions described in the alert rule, alert notifications are sent to the specified alert group.

4.Configure alarms in Log Service

4.1. Overview

In this practice, the alert feature of Alibaba Cloud Log Service is used to configure custom monitoring charts and alerts for domain names that are added to WAF and have Log Service enabled. Enterprise users and individual users can refer to this practice to monitor the traffic and security status of their workloads and configure alerts.

Procedure

This practice contains the following steps.

Step	Description
Step 1: Create a WAF log analysis dashboard	After you use Log Service in WAF to initiate log query and analysis, you can create a dashboard based on the SQL statement. By default, the dashboard contains the charts generated based on the SQL statements.
Step 2: Configure log charts	After you create a log analysis dashboard, you can edit or delete log charts on the dashboard or create a new log chart by copying an existing chart.
Step 3: Configure log alerts	After you create a log analysis dashboard, you can configure log alert on the dashboard. You must associate an alert with an existing log chart and set the alert trigger conditions based on the parameters in the associated chart. You can customize the alert message template.

Configuration examples

This practice provides 13 examples of log charts and alert configurations, including alerts on an abnormal percentage of 4xx status codes (blocked requests excluded), alerts on an abnormal percentage of 5xx status codes, alerts on an abnormal query rate, alerts on an abrupt increase in query rate, alerts on an abrupt decrease in query rate, alerts on requests blocked by HTTP ACL policy in the last five minutes, alerts on requests blocked by web application protection in the last five minutes, alerts on requests blocked by HTTP flood protection in the last five minutes, alerts on requests blocked by anti-scan rules in the last five minutes, alerts on the number of attacks from a single source IP address in the last five minutes, alerts on the number of domains attacked by a single IP address in the last five minutes, alerts on average delay in the last five minutes, and alerts on an abrupt decrease in query rate from a user.

We recommend that you learn how to configure a log chart (step 2), configure an alert rule (step 3), and then create chart and configure an alert rule. For more information, see [Examples of alert configurations based on WAF logs](#).

For more information about the metrics used in alert configuration and the recommended thresholds for the metrics, see [Common monitoring metrics](#).

For more information about the SQL statements used to query and analyze logs, see [Query statements](#).

4.2. Step 1: Create a WAF log analysis dashboard

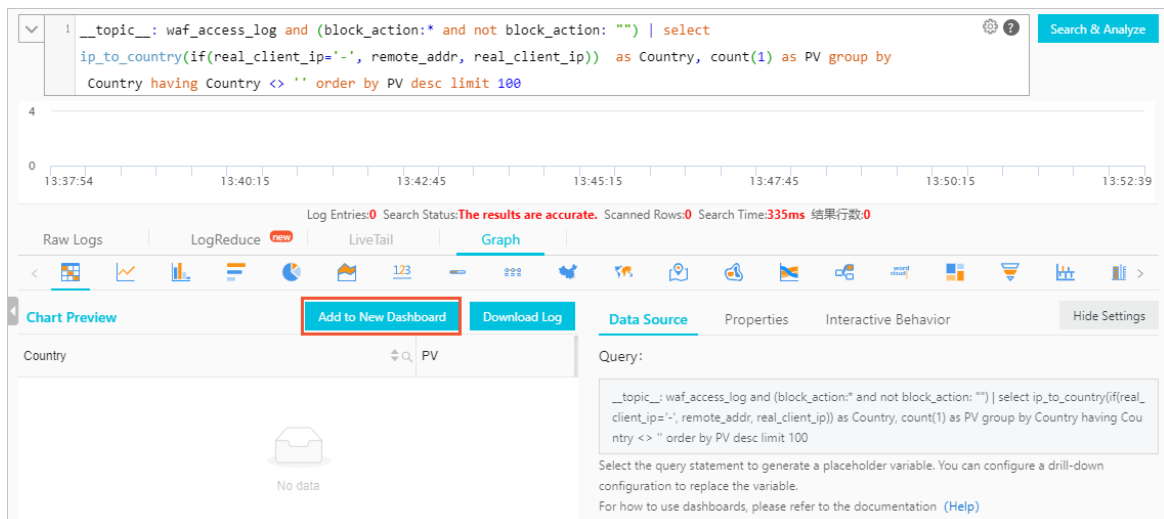
After you perform log query and analysis by using the Log Service for WAF feature, you can create a dashboard based on the SQL statement. By default, the dashboard contains the chart that is generated based on the SQL statement.

Prerequisites

- Your domain name is added to WAF for protection. For more information, see [Change a DNS record](#).
- The Log Service for WAF feature is enabled for your domain name. For more information, see [Enable Log Service for WAF](#).

Procedure

1. Log on to the [WAF console](#).
2. Go to the advanced management page of Log Service for WAF.
 - i. In the top navigation bar, select **Mainland China** or **International**.
 - ii. In the left-side navigation pane, choose **Log Management** > **Log Service**.
 - iii. In the upper-right corner of the **Log Service** page, click **Advanced Settings**.
 - iv. In the dialog box that appears, click **OK**.
3. In the project list, find the log project that you want to manage, and click the project name.
4. Enter an SQL statement and click **Search & Analyze**.
5. After the query is complete, click **Add to New Dashboard** on the **Graph** tab.



6. In the **Add to New Dashboard** dialog box, configure the following parameters and click **OK**.

Add to New Dashboard

* Operation:

Create Dashboard

* Dashboard Name:

waf-alert

* Chart Name:

c1

Cancel

OK

Parameter	Description
Operation	Select Create Dashboard .
Dashboard Name	Enter a dashboard name.
Chart Name	Enter a name for the chart generated based on the SQL statements.

Result

After the dashboard is created, you are redirected to the new dashboard. By default, the dashboard contains the chart that is generated based on the SQL statement entered in Step 4. You can edit the chart or create more charts on the dashboard.

What's next

[Step 2: Configure log charts](#)

4.3. Step 2: Configure log charts

After you create a log analysis dashboard, you can edit or delete log charts on the dashboard. In addition, you can create a new log chart by copying an existing chart.

Prerequisites

A log analysis dashboard is created. For more information, see [Step 1: Create a WAF log analysis dashboard](#).

Context

The practices provide 13 default chart configuration examples. For more information, see [Examples of alert configurations based on WAF logs](#).

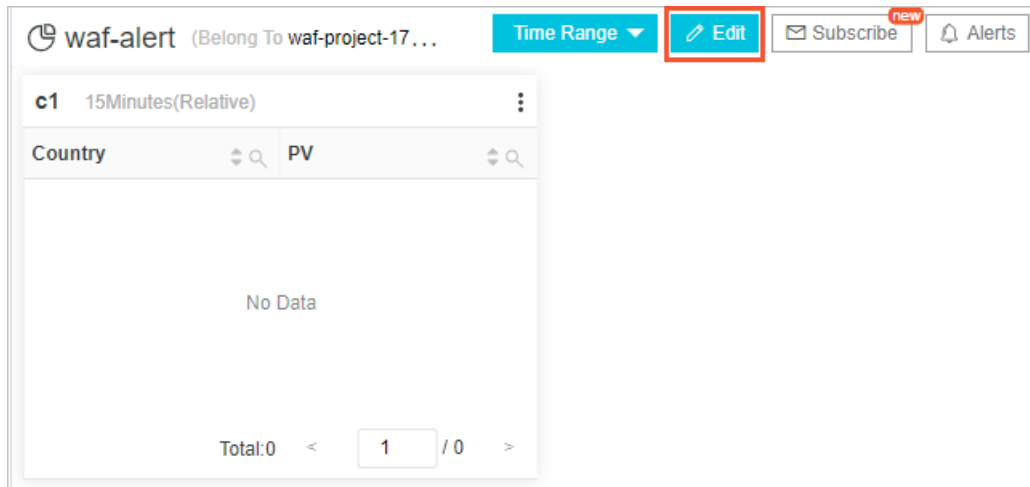
Before you create a chart based on the examples and configure alerts during the chart creation process, we recommend that you familiarize yourself with how to configure alerts. For more information, see [Step 3: Configure log alerts](#).

Procedure

1. Enter the customized WAF log analysis dashboard.

For more information, see [Step 1: Create a WAF log analysis dashboard](#).


2. In the upper-right corner of the dashboard, click **Edit**.



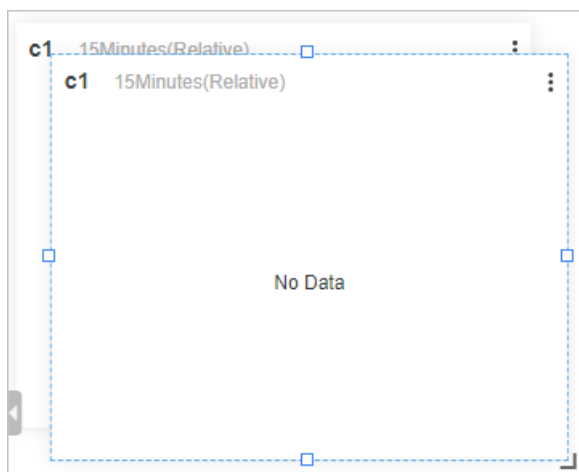
The dashboard enters the edit mode.

3. In this mode, you can edit or delete the charts on the dashboard. In addition, you can create a new chart by copying a chart.


Note You can copy a chart to create a new chart. Then, you can edit the new chart. You can add more than one chart to a dashboard. This allows you to display data and configure alerts in various ways.

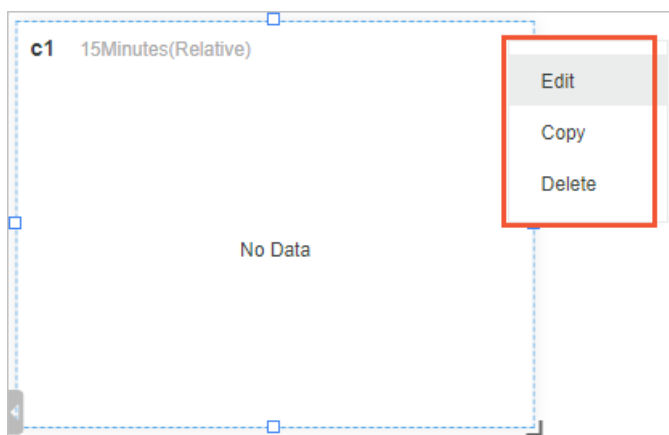
- Copy a chart to create a new chart.
 - a. Find the chart that you want to copy. Move the pointer over the  icon in the upper-right corner of the chart, and click **Copy**.

After you copy a chart, an identical chart appears.



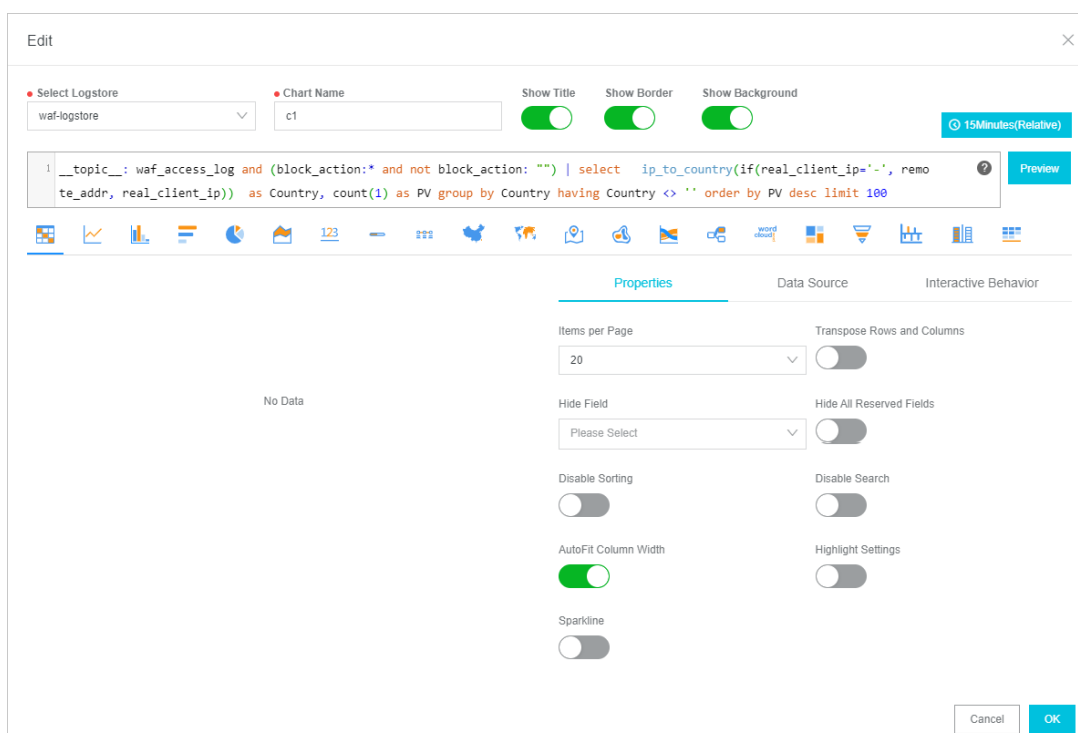
- b. Drag the new chart and drop it at an appropriate position on the dashboard.
- Edit a chart.

- a. Find the chart that you want to edit, move the pointer over the  icon in the upper-right corner of the chart, and click **Edit**.




- b. On the **Edit** page, modify the chart configurations, such as **Chart Name**, SQL statements, relative data collection period, and chart type. Then, click **OK**.

Note If you have modified the SQL statements, you must click **Preview** before you click **OK**. This operation triggers a check of the statement validity. If the SQL statements are invalid, an error message appears, and the **OK** button becomes unavailable. You can click **OK** only after you make sure the statements are valid.



- o Delete a chart.

Find the chart that you want to delete. Move the pointer over the  icon in the upper-right corner of the chart, and click **Delete**.

What's next

Step 3: Configure log alerts

4.4. Step 3: Configure log alerts

After you create a log analysis dashboard, you can configure log alerts on the dashboard. You must associate the alerts with existing log charts and set the alert trigger conditions based on the parameters in the charts. You can customize an alert message template.

Prerequisites

A log analysis dashboard is created. For more information, see [Step 1: Create a WAF log analysis dashboard](#).

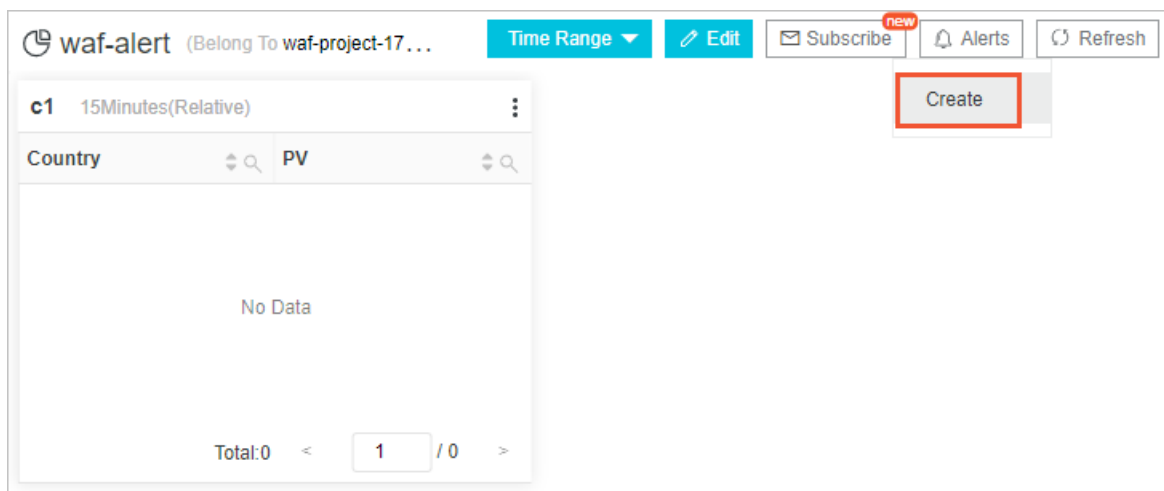
Context

The practices provide 13 default alert configuration examples. For more information, see [Examples of alert configurations based on WAF logs](#).

We recommend that you learn how to configure charts before you create a chart based on the examples. In addition, you can configure alerts and notification methods during chart creation. For more information, see [Step 2: Configure log charts](#).

Procedure

1. Enter the customized WAF log analysis dashboard.
For more information, see [Step 1: Create a WAF log analysis dashboard](#).
2. In the upper-right corner of the dashboard, choose **Alerts > Create**.



3. In the **Create Alert** pane, set the parameters in **Alert Configuration**, and click **Next**.

Create Alert

Alert Configuration

Notifications

* Alert Name

alert-test

10/64

* Associated Chart

0

Chart Name

c1

Query

```
__topic__: waf_access_log and (block_action:*
and not block_action: "") | select ip_to_country
(if(real_client_ip='-', remote_addr, real_client_ip)) as Country, count(1) as PV group by Country having Country <> " order by PV desc limit 100
```

Search Period

15Minutes(Relative)

1

Add

* Frequency

Fixed Interval

15

Minutes

* Trigger Condition

Country!=china

Five basic operators are supported: plus (+), minus (-), multiplication (*), division (/), and modulo (%). Eight comparison operators are supported: greater than (>), greater than or equal to (>=), less than (<), less than or equal to (<=), equal to (==), not equal to (!=), regex match (=~), and negated regex match (!~). [Documentation](#)

Advanced

* Notification Trigger Threshold

1



* Notification Interval


5 Minutes


Next

Cancel

Parameter	Description
Alert Name	The name of the alert. The name must be 1 to 64 characters in length.

Parameter	Description
Associated Chart	<p>The charts with which an alert is associated.</p> <p>The Search Period parameter specifies the time range of log data that WAF reads when you query data. You can select a relative time or a time frame. For example, if you set Search Period to 15 minutes (relative) and start the query at 14:30:06, WAF reads the log data that was written from 14:15:06 to 14:30:06. If you set Search Period to 15 minutes (time frame) and start the query at 14:30:06, WAF reads the log data that was written from 14:15:00 to 14:30:00.</p> <p>To associate the alert with more than one chart, click Add and configure the new charts. You can add up to three charts. The number before the chart name is the sequence number of the chart in alert configuration. You can use the sequence number to associate a chart with a conditional expression in the trigger condition.</p>
Frequency	<p>The time interval at which the server checks log data according to the alert configuration.</p> <div>  Note Currently, the server samples and checks only the first 100 data entries each time the specified time interval arrives. </div>
Trigger Condition	<p>The conditional expression that determines whether the alert is triggered. When the condition is met, the system sends an alert notification based on the specified Frequency and Notification Interval.</p> <p>By default, the charts are numbered from 0. In a trigger condition, you can use <code>\$0</code> to indicate the first chart. For example, you can set a trigger condition to <code>\$0.domainnum>=10</code>, and this indicates that an alert is triggered if the <code>domainnum</code> parameter in the first chart is greater than or equal to 10.</p> <p>If two conditions are jointed with two consecutive ampersands (<code>&&</code>), both the conditions must be met to trigger the alert. If two conditions are jointed with two consecutive vertical bars (<code> </code>), either of the condition can trigger the alert.</p> <div>  Note For more syntaxes of conditional expressions, see Syntax of trigger conditions in alert rules. </div>
Advanced	

Parameter	Description
Notification Trigger Threshold	<p>The threshold for sending an alert notification based on the specified notification interval when the cumulative number of times that the trigger condition is met exceeds this threshold. If the trigger condition is not met, the overall count does not change.</p> <p>The default value of Notification Trigger Threshold is 1. That is, each time the specified trigger condition is met, the server checks whether the specified notification interval arrives.</p> <p>You can also specify this parameter to enable the server to send an alert notification after the trigger condition is met multiple times. For example, if you set this parameter to 100, the server checks whether the specified notification interval arrives only after the trigger condition is met 100 times. If the specified notification trigger threshold is reached and the specified notification interval arrives, the server sends an alert notification. Then, the overall count is reset. If the server fails to check log data due to exceptions such as a network failure, the overall count does not change.</p>
Notification Interval	<p>The time interval at which the server sends an alert notification.</p> <p>If the trigger condition is met several times that exceed the specified notification trigger threshold and the specified notification interval arrives, the server sends an alert notification. If you set this parameter to 5 minutes, you can receive up to one alert notification every 5 minutes for the alert. The default value is No Interval.</p> <div><p> Note By setting Notification Trigger Threshold and Notification Interval, you can control the number of alert notifications that you receive.</p></div>

 **Note** After you specify **Notification Trigger Threshold**, **Notification Interval**, and **Frequency**, the system checks whether the trigger conditions are met at the specified frequency and sends notifications if **Notification Trigger Threshold** is exceeded within a **Notification Interval**.

4. In the **Create Alert** pane, complete the settings for **Notifications**, and click **Submit**.

Create Alert

Alert Configuration Notifications

Notifications

WebHook-DingTalk Bot x

SMS x Voice x

Email x

WebHook-DingTalk Bot

* Request URL

Title [Log Service Alert] test

Recipients None All Sp

* Content

- [Uid] \${aliuid}

- [Project] \${project}

(https://sls.console.aliyun.com/#/project/\${project}/categoryList)

- [Trigger] \${AlertDisplayName}

Supported template variables: \${Project}, \${Condition}, \${AlertName}, \${AlertID}, \${Dashboard}, \${FireTime}, \${Results} [View all variables](#)

Previous Submit Cancel

Multiple common alert notification methods are supported, such as **SMS**, **Voice**, **Email**, and **WebHook-DingTalk Bot**. You must select a notification method on the right of **Notifications** and complete the configuration. You can select and configure multiple notification methods.

- **SMS**: Set **Phone Number** to receive alerts and the **Content** of the notification. You can specify variables to be included in the notification. Click **View all variables** to view the description of each variable.

The screenshot shows the 'Alert Configuration' interface with the 'Notifications' tab selected. A dropdown menu shows 'SMS' is chosen. Below it, a modal window for 'SMS' configuration is open. It contains a 'Phone Number' field with a note 'Use commas (,) to separate multiple mobile phone numbers.' and a 'Content' text area. At the bottom, it lists supported template variables: \${Project}, \${Condition}, \${AlertName}, \${AlertID}, \${Dashboard}, \${FireTime}, and \${Results}, with a link to 'View all variables'.

- Voice: Set **Phone Number** to receive alerts and the **Content** of the notification.

This screenshot is similar to the one above but shows the 'Voice' notification type selected in the dropdown. The modal window for 'Voice' configuration is open, featuring the same 'Phone Number' and 'Content' fields. The template variables listed at the bottom are identical to the SMS configuration.

- Email: Set **Recipients** email addresses, **Subject**, and **Content**.

The screenshot displays the 'Alert Configuration' interface, specifically the 'Notifications' tab. A modal window titled 'Email' is open, showing configuration options for email alerts. The modal includes a close button (X) in the top right corner. The configuration fields are as follows:

- Recipients:** A text input field with a red asterisk (*) indicating it is required. Below the field, a note states: 'Use commas (,) to separate multiple recipients.'
- Subject:** A text input field containing the text 'Log Service Alert'.
- Content:** A large text area with a red asterisk (*) indicating it is required.

At the bottom of the modal, a list of supported template variables is provided: `${Project}`, `${Condition}`, `${AlertName}`, `${AlertID}`, `${Dashboard}`, `${FireTime}`, and `${Results}`. A link labeled 'View all variables' is also present.

- WebHook-DingTalk Bot: Set **Request URL** to the webhook URL of the DingTalk bot to receive alerts, and specify **Content**.

The screenshot displays the 'Alert Configuration' tab for a 'WebHook-DingTalk Bot' notification. The configuration includes a 'Request URL' field, a 'Title' field set to '[Log Service Alert] test', and 'Recipients' set to 'None'. The 'Content' field contains a template with variables like `[Uid] ${aliuid}`, `[Project] ${project}`, and `[Trigger] ${AlertDisplayName}`. A list of supported template variables is provided at the bottom, including `Project`, `Condition`, `AlertName`, `AlertID`, `Dashboard`, `FireTime`, and `Results`.

5. Repeat Steps 2 to 4 to create and configure more alerts.

4.5. Examples of alert configurations based on WAF logs

This topic provides examples of alert configurations based on log query and analysis results in Web Application Firewall (WAF). You can add charts to custom dashboards and configure alerts based on the parameters in this topic.

Notice This topic describes the alert configuration parameters of the original alerting feature in Log Service. If you use the new alerting feature, you can configure alerts based on the query statements and parameter settings that are recommended in this topic and the description that is provided in [Configure an alert in Log Service](#).

Abnormal percentage of 4xx status codes

The following parameter settings are recommended for this type of alerting:

- **Chart Name:** Percentage of 4xx status codes. Blocked requests are not counted.
- **Query Statement:**

```
user_id: ID of your Alibaba Cloud account
and not real_client_ip: Blocked IP addresses |
SELECT
  user id,
```

```

status_2XX,
host AS "Domain name",
Rate_2XX AS "Percentage of 2xx status codes",
Rate_3XX AS "Percentage of 3xx status codes",
Rate_4XX AS "Percentage of 4xx status codes",
Rate_5XX AS "Percentage of 5xx status codes",
countall AS "aveQPS",
status_2XX,
status_3XX,
status_4XX,
status_5XX,
countall
FROM(
  SELECT
    user_id,
    host,
    round(
      round(status_2XX * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_2XX,
    round(
      round(status_3XX * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_3XX,
    round(
      round(status_4XX * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_4XX,
    round(
      round(status_5XX * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_5XX,
    status_2XX,
    status_3XX,
    status_4XX,
    status_5XX,
    countall
  FROM(
    SELECT
      user_id,
      host,
      count_if(
        status >= 200
        and status < 300
      ) AS status_2XX,
      count_if(
        status >= 300
        and status < 400
      ) AS status_3XX,
      count_if(
        status >= 400
        and status < 500
        and status <> 444
        and status <> 405
      ) AS status_4XX,

```

```
count_if(
    status >= 500
    and status < 600
) AS status_5XX,
COUNT(*) AS countall
FROM log
GROUP BY
    host,
    user_id
)
WHERE
    countall > 120
ORDER BY
    Rate_4XX DESC
LIMIT
    5
```

The chart contains the following fields: `aveQPS` , `Percentage of 2xx status codes` , `Percentage of 3xx status codes` , `Percentage of 4xx status codes` , and `Percentage of 5xx status codes` . `aveQPS` indicates the queries per second (QPS) of the domain name. To show status code changes caused by system workloads instead of external reasons, the 444 and 405 status codes that are triggered by WAF-blocked HTTP flood attacks or web attacks are not counted in `Percentage of 4xx status codes` . You can select one or more of these fields to configure alerts. For example, you can specify `aveQPS>10 && Percentage of 2xx status codes<60` . If the QPS of a specified domain name is higher than 10 and the percentage of requests whose status code is 2xx in all requests is less than 60% during a specified period, an alert is triggered.

- **Time Range:** 5 minutes (relative)
- **Frequency:** 5 minutes
- **Trigger Condition:** `$0.countall>3000&& $0.Percentage of 4xx status codes>80`
- **Notification Triggering Threshold:** 2
- **Notification Interval:** 10 minutes
- **Content :**

```
- [Time]: ${FireTime}
- [Uid]: ${Results[0].RawResults[0].user_id}
- Domain name: ${Results[0].RawResults[0].Domain name}
- Service: WAF
- Requests in the last 5 minutes: ${Results[0].RawResults[0].countall}
- Percentage of 2xx status codes: ${Results[0].RawResults[0].Percentage of 2xx status codes} %
- Percentage of 3xx status codes: ${Results[0].RawResults[0].Percentage of 3xx status codes} %
- Percentage of 4xx status codes: ${Results[0].RawResults[0].Percentage of 4xx status codes} %
- Percentage of 5xx status codes: ${Results[0].RawResults[0].Percentage of 5xx status codes} %
```

Abnormal percentage of 5xx status codes

The following parameter settings are recommended for this type of alerting:

- **Chart Name:** Percentage of 5xx status codes
- **Query Statement:**

```
user_id: ID of your Alibaba Cloud account
and not real_client_ip: Blocked IP addresses |
select
  user_id,
  host AS "Domain name",
  Rate_2XX AS "Percentage of 2xx status codes",
  Rate_3XX AS "Percentage of 3xx status codes",
  Rate_4XX AS "Percentage of 4xx status codes",
  Rate_5XX AS "Percentage of 5xx status codes",
  countall AS "Requests in a specified relative time range",
  status_2XX,
  status_3XX,
  status_4XX,
  status_5XX,
  countall
FROM(
  SELECT
    user_id,
    host,
    round(
      round(status_2XX * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_2XX,
    round(
      round(status_3XX * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_3XX,
    round(
      round(status_4XX * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_4XX,
    round(
      round(status_5XX * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_5XX,
    status_2XX,
    status_3XX,
    status_4XX,
    status_5XX,
    countall
  FROM(
    SELECT
      user_id,
      host,
      count_if(
        status >= 200
        and status < 300
      ) AS status_2XX,
      count_if(
        status >= 300
        and status < 400
```

```
        ) AS status_3XX,
        count_if(
            status >= 400
            and status < 500
        ) AS status_4XX,
        count_if(
            status >= 500
            and status < 600
        ) AS status_5XX,
        COUNT(*) AS countall
    FROM      log
    GROUP BY
        host,
        user_id
)
)
WHERE
    countall > 120
ORDER BY
    Rate_5XX DESC
LIMIT
    5
```

- **Time Range:** 5 minutes (relative)
- **Frequency:** 5 minutes
- **Trigger Condition:** `$0.countall>3000&& $0.Percentage of 5xx status codes>80`
- **Notification Triggering Threshold:** 2
- **Notification Interval:** 10 minutes
- **Content:**

```
- [Time]: ${FireTime}
- [Uid]: ${Results[0].RawResults[0].user_id}
- Domain name: ${Results[0].RawResults[0].Domain name}
- Service: WAF
- Requests in the last 5 minutes: ${Results[0].RawResults[0].countall}
- Percentage of 2xx status codes: ${Results[0].RawResults[0].Percentage of 2xx status codes} %
- Percentage of 3xx status codes: ${Results[0].RawResults[0].Percentage of 3xx status codes} %
- Percentage of 4xx status codes: ${Results[0].RawResults[0].Percentage of 4xx status codes} %
- Percentage of 5xx status codes: ${Results[0].RawResults[0].Percentage of 5xx status codes} %
```

Abnormal QPS

The following parameter settings are recommended for this type of alerting:

- **Chart Name:** Top 5 domain names that have the highest QPS
- **Query Statement:**

```
user_id: ID of your Alibaba Cloud account
and not real_client_ip: Blocked IP addresses |
-----
```

```

SELECT
    user_id,
    host,
    Rate_2XX,
    Rate_3XX,
    Rate_4XX,
    Rate_5XX,
    countall / 60 as "aveQPS",
    status_2XX,
    status_3XX,
    status_4XX,
    status_5XX,
    countall
FROM(
    SELECT
        user_id,
        host,
        round(
            round(status_2XX * 1.0000 / countall, 4) * 100,
            2
        ) as Rate_2XX,
        round(
            round(status_3XX * 1.0000 / countall, 4) * 100,
            2
        ) as Rate_3XX,
        round(
            round(status_4XX * 1.0000 / countall, 4) * 100,
            2
        ) as Rate_4XX,
        round(
            round(status_5XX * 1.0000 / countall, 4) * 100,
            2
        ) as Rate_5XX,
        status_2XX,
        status_3XX,
        status_4XX,
        status_5XX,
        countall
    FROM(
        SELECT
            user_id,
            host,
            count_if(
                status >= 200
                and status < 300
            ) as status_2XX,
            count_if(
                status >= 300
                and status < 400
            ) as status_3XX,
            count_if(
                status >= 400
                and status < 500
                and status <> 444
                and status <> 405
            ) as status_4XX,
            count_if(
                status >= 500
                and status < 600
            ) as status_5XX,
            count(*) as countall
        FROM log
        WHERE @timestamp >= '2021-12-21 00:00:00'
        AND @timestamp < '2021-12-21 01:00:00'
        GROUP BY user_id, host
    )
)

```

```
        and status < 400
      ) as status_4XX,
      count_if(
        status >= 500
        and status < 600
      ) as status_5XX,
      COUNT(*) as countall
    FROM      log
    GROUP BY
      host,
      user_id
  )
)
WHERE
  countall > 120
ORDER BY
  aveQPS DESC
LIMIT
  5
```

- **Time Range:** 1 minute (relative)
- **Frequency:** 1 minute
- **Trigger Condition:** `$0.aveQPS>=50`
- **Notification Triggering Threshold:** 1
- **Notification Interval:** 5 minutes
- **Content:**

```
- [Time]: ${FireTime}
- [Uid]: ${Results[0].RawResults[0].user_id}
- Domain name: ${Results[0].RawResults[0].host}
- Service: WAF
- Average QPS in the last 1 minute: ${Results[0].RawResults[0].aveQPS}
- Percentage of 2xx status codes: ${Results[0].RawResults[0].Rate_2XX}%
- Percentage of 3xx status codes: ${Results[0].RawResults[0].Rate_3XX}%
- Percentage of 4xx status codes: ${Results[0].RawResults[0].Rate_4XX}%
- Percentage of 5xx status codes: ${Results[0].RawResults[0].Rate_5XX}%
```

Abrupt increase in QPS

The following parameter settings are recommended for this type of alerting:

- **Chart Name:** Abrupt increase in QPS
- **Query Statement:**

```
user_id: ID of your Alibaba Cloud account |
SELECT
  t1.user_id,
  t1.now1mQPS,
  t1.past1mQPS,
  in_ratio,
  t1.host,
  t2.Rate_2XX,
  Rate_3XX,
  Rate 4XX,
```

```

        Rate_5XX,
        aveQPS
    FROM (
        (
            SELECT
                user_id,
                round(c [1] / 60, 0) AS now1mQPS,
                round(c [2] / 60, 0) AS past1mQPS,
                round(
                    round(c [1] / 60, 0) / round(c [2] / 60, 0) * 100 -100,
                    0
                ) AS in_ratio,
                host
            FROM (
                SELECT
                    compare(t, 60) AS c,
                    host,
                    user_id
                FROM (
                    SELECT
                        COUNT(*) AS t,
                        host,
                        user_id
                    FROM log
                    GROUP by
                        host,
                        user_id
                    )
                GROUP by
                    host,
                    user_id
            )
        )
        WHERE
            c [3] > 1.1
            and (
                c [1] > 180
                or c [2] > 180
            )
    ) t1
    JOIN (
        SELECT
            user_id,
            host,
            Rate_2XX,
            Rate_3XX,
            Rate_4XX,
            Rate_5XX,
            countall / 60 AS "aveQPS",
            status_2XX,
            status_3XX,
            status_4XX,
            status_5XX,
            countall
        FROM (

```

```
SELECT
    user_id,
    host,
    round(
        round(status_2XX * 1.0000 / countall, 4) * 100,
        2
    ) AS Rate_2XX,
    round(
        round(status_3XX * 1.0000 / countall, 4) * 100,
        2
    ) AS Rate_3XX,
    round(
        round(status_4XX * 1.0000 / countall, 4) * 100,
        2
    ) AS Rate_4XX,
    round(
        round(status_5XX * 1.0000 / countall, 4) * 100,
        2
    ) AS Rate_5XX,
    status_2XX,
    status_3XX,
    status_4XX,
    status_5XX,
    countall
FROM      (
    SELECT
        user_id,
        host,
        count_if(
            status >= 200
            and status < 300
        ) AS status_2XX,
        count_if(
            status >= 300
            and status < 400
        ) AS status_3XX,
        count_if(
            status >= 400
            and status < 500
            and status <> 444
            and status <> 405
        ) AS status_4XX,
        count_if(
            status >= 500
            and status < 600
        ) AS status_5XX,
        COUNT(*) AS countall
    FROM      log
    GROUP BY
        host,
        user_id
)
WHERE
```

```

        countall > 1
    ) t2 on t1.host = t2.host
)
ORDER BY
    in_ratio DESC
LIMIT
    5

```

- **Time Range:** 1 minute (relative)
- **Frequency:** 1 minute
- **Trigger Condition:** `$0.now1mqps>50&& $0.in_ratio>300`
- **Notification Triggering Threshold:** 1
- **Notification Interval:** 5 minutes
- **Content:**

```

- [Time]: ${FireTime}
- [Uid]: ${Results[0].RawResults[0].user_id}
- Domain name: ${Results[0].RawResults[0].host}
- Service: WAF
- Average QPS in the last 1 minute: ${Results[0].RawResults[0].now1mqps}
- Abrupt increase rate of QPS: ${Results[0].RawResults[0].in_ratio}%
- Percentage of 2xx status codes: ${Results[0].RawResults[0].rate_2xx}%
- Percentage of 3xx status codes: ${Results[0].RawResults[0].Rate_3XX}%
- Percentage of 4xx status codes: ${Results[0].RawResults[0].Rate_4XX}%
- Percentage of 5xx status codes: ${Results[0].RawResults[0].Rate_5XX}%

```

Abrupt decrease in QPS

- **Chart Name:** Abrupt decrease in QPS
- **Query Statement:**

```

user_id: ID of your Alibaba Cloud account
SELECT
    t1.user_id,
    t1.now1mqps,
    t1.past1mqps,
    de_ratio,
    t1.host,
    t2.Rate_2XX,
    Rate_3XX,
    Rate_4XX,
    Rate_5XX,
    aveQPS
FROM (
    (
        SELECT
            user_id,
            round(c [1] / 60, 0) AS now1mqps,
            round(c [2] / 60, 0) AS past1mqps,
            round(
                100-round(c [1] / 60, 0) / round(c [2] / 60, 0) * 100,
                2
            ) AS de_ratio,

```

```
host
FROM      (
    SELECT
        compare(t, 60) AS c,
        host,
        user_id
    FROM      (
        SELECT
            COUNT(*) AS t,
            host,
            user_id
        FROM      log
        GROUP BY
            host,
            user_id
        )
    GROUP BY
        host,
        user_id
    )
WHERE
    c [3] < 0.9
AND (
    c [1] > 180
    or c [2] > 180
    )
) t1
JOIN (
    SELECT
        user_id,
        host,
        Rate_2XX,
        Rate_3XX,
        Rate_4XX,
        Rate_5XX,
        countall / 60 AS "aveQPS",
        status_2XX,
        status_3XX,
        status_4XX,
        status_5XX,
        countall
    FROM      (
        SELECT
            user_id,
            host,
            round(
                round(status_2XX * 1.0000 / countall, 4) * 100,
                2
            ) AS Rate_2XX,
            round(
                round(status_3XX * 1.0000 / countall, 4) * 100,
                2
            ) AS Rate_3XX,
            round(
```



```

        round(status_4XX * 1.0000 / countall, 4) * 100,
        2
    ) AS Rate_4XX,
    round(
        round(status_5XX * 1.0000 / countall, 4) * 100,
        2
    ) AS Rate_5XX,
    status_2XX,
    status_3XX,
    status_4XX,
    status_5XX,
    countall
FROM      (
    SELECT
        user_id,
        host,
        count_if(
            status >= 200
            and status < 300
        ) AS status_2XX,
        count_if(
            status >= 300
            and status < 400
        ) AS status_3XX,
        count_if (
            status >= 400
            and status < 500
            and status <> 444
            and status <> 405
        ) AS status_4XX,
        count_if(
            status >= 500
            and status < 600
        ) AS status_5XX,
        COUNT(*) AS countall
    FROM      log
    GROUP BY
        host,
        user_id
    )
    )
WHERE
    countall > 1
    ) t2 on t1.host = t2.host
    )
ORDER BY
    de_ratio DESC
LIMIT
    5

```

The chart contains the following fields: `now1mpqs` , `past1mpqs` , `de_ratio` , and `host` .

`now1mpqs` indicates the average QPS of the current minute. `past1mpqs` indicates the average QPS of the last minute. `de_ratio` indicates the QPS decrease rate. You can select one or more of these fields to configure alerts.

- **Time Range:** 1 minute (relative)
- **Frequency:** 1 minute
- **Trigger Condition:** `$0.now1mqps>10&& $0.de_ratio>50`
- **Notification Triggering Threshold:** 2
- **Notification Interval:** 5 minutes
- **Content:**

```
- [Time]: ${FireTime}
- [Uid]: ${Results[0].RawResults[0].user_id}
- Domain name: ${Results[0].RawResults[0].host}
- Service: WAF (International)
- Average QPS in the last 1 minute: ${Results[0].RawResults[0].now1mqps}
- Abrupt decrease rate of QPS: ${Results[0].RawResults[0].de_ratio}%
- Percentage of 2xx status codes: ${Results[0].RawResults[0].rate_2xx}%
- Percentage of 3xx status codes: ${Results[0].RawResults[0].Rate_3XX}%
- Percentage of 4xx status codes: ${Results[0].RawResults[0].Rate_4XX}%
- Percentage of 5xx status codes: ${Results[0].RawResults[0].Rate_5XX}%
```

Requests blocked by access control list (ACL) policies in the last 5 minutes

The following parameter settings are recommended for this type of alerting:

- **Chart Name:** Requests blocked by ACL policies
- **Query Statement:**

```

user_id: ID of your Alibaba Cloud account
SELECT
  user_id,
  host,
  count_if(
    final_plugin = 'waf'
    AND final_action = 'block'
  ) AS "Requests blocked by the Protection Rules Engine",
  count_if(
    final_plugin = 'cc'
    AND final_action = 'block'
  ) AS "Requests blocked by HTTP flood protection policies",
  count_if(
    final_plugin = 'acl'
    AND final_action = 'block'
  ) AS "Requests blocked by ACL policies",
  count_if(
    final_plugin = 'antiscan'
    AND final_action = 'block'
  ) AS "Requests blocked by scan protection policies",
  count_if(
    (final_plugin = 'waf'
    AND final_action = 'block')
    OR (final_plugin = 'cc'
    AND final_action = 'block')
    OR (final_plugin = 'acl'
    AND final_action = 'block')
    OR (final_plugin = 'antiscan'
    AND final_action = 'block')
  ) AS totalblock
GROUP BY
  host,
  user_id
HAVING
  (
    "Requests blocked by ACL policies" >= 0
    AND "Requests blocked by the Protection Rules Engine" >= 0
    AND "Requests blocked by HTTP flood protection policies" >= 0
    AND "Requests blocked by scan protection policies" >= 0
    AND totalblock > 10
  )
ORDER BY
  "Requests blocked by ACL policies" DESC
LIMIT
  5

```

- **Time Range:** 5 minutes (relative)
- **Frequency:** 5 minutes
- **Trigger Condition:** `$0.totalblock>=500&&($0.Requests blocked by ACL policies>=500)`
- **Notification Triggering Threshold:** 1
- **Notification Interval:** 5 minutes
- **Content:**

```
- [Time]: ${FireTime}
- [Uid]: ${Results[0].RawResults[0].user_id}
- Domain name: ${Results[0].RawResults[0].host}
- Service: WAF
- Requests that are blocked in the last 5 minutes: ${Results[0].RawResults[0].totalblock}
- Requests blocked by ACL policies: ${Results[0].RawResults[0].Requests blocked by ACL policies}
- Requests blocked by the Protection Rules Engine: ${Results[0].RawResults[0].Requests blocked by the Protection Rules Engine}
- Requests blocked by HTTP flood protection policies: ${Results[0].RawResults[0].Requests blocked by HTTP flood protection policies}
- Requests blocked by scan protection policies: ${Results[0].RawResults[0].Requests blocked by scan protection policies}
```

Requests blocked by the Protection Rules Engine in the last 5 minutes

The following parameter settings are recommended for this type of alerting:

- **Chart Name:** Requests blocked by the Protection Rules Engine
- **Query Statement:**

```
user_id: ID of your Alibaba Cloud account
SELECT
  user_id,
  host,
  count_if(
    final_plugin = 'waf'
    AND final_action = 'block'
  ) AS "Requests blocked by the Protection Rules Engine",
  count_if(
    final_plugin = 'cc'
    AND final_action = 'block'
  ) AS "Requests blocked by HTTP flood protection policies",
  count_if(
    final_plugin = 'acl'
    AND final_action = 'block'
  ) AS "Requests blocked by ACL policies",
  count_if(
    final_plugin = 'antiscan'
    AND final_action = 'block'
  ) AS "Requests blocked by scan protection policies",
  count_if(
    (final_plugin = 'waf'
    AND final_action = 'block')
    OR (final_plugin = 'cc'
    AND final_action = 'block')
    OR (final_plugin = 'acl'
    AND final_action = 'block')
    OR (final_plugin = 'antiscan'
    AND final_action = 'block')
  ) AS totalblock
GROUP BY
  host,
  user_id
HAVING
  (
    "Requests blocked by ACL policies" >= 0
    AND "Requests blocked by the Protection Rules Engine" >= 0
    AND "Requests blocked by HTTP flood protection policies" >= 0
    AND "Requests blocked by scan protection policies" >= 0
    AND totalblock > 10
  )
ORDER BY
  "Requests blocked by the Protection Rules Engine" DESC
LIMIT
  5
```

- **Time Range:** 5 minutes (relative)
- **Frequency:** 5 minutes
- **Trigger Condition:** `${0.totalblock}>=500&&(${0.Requests blocked by the Protection Rules Engine}>=500)`
- **Notification Triggering Threshold:** 1
- **Notification Interval:** 5 minutes

- **Content :**

```
- [Time]: ${FireTime}
- [Uid]: ${Results[0].RawResults[0].user_id}
- Domain name: ${Results[0].RawResults[0].host}
- Service: WAF
- Requests that are blocked in the last 5 minutes: ${Results[0].RawResults[0].totalblock}
- Requests blocked by ACL policies: ${Results[0].RawResults[0].Requests blocked by ACL policies}
- Requests blocked by the Protection Rules Engine: ${Results[0].RawResults[0].Requests blocked by the Protection Rules Engine}
- Requests blocked by HTTP flood protection policies: ${Results[0].RawResults[0].Requests blocked by HTTP flood protection policies}
- Requests blocked by scan protection policies: ${Results[0].RawResults[0].Requests blocked by scan protection policies}
```

Requests blocked by HTTP flood protection policies in the last 5 minutes

The following parameter settings are recommended for this type of alerting:

- **Chart Name:** Requests blocked by HTTP flood protection policies
- **Query Statement :**

```

user_id: ID of your Alibaba Cloud account
SELECT
  user_id,
  host,
  count_if(
    final_plugin = 'waf'
    AND final_action = 'block'
  ) AS "Requests blocked by the Protection Rules Engine",
  count_if(
    final_plugin = 'cc'
    AND final_action = 'block'
  ) AS "Requests blocked by HTTP flood protection policies",
  count_if(
    final_plugin = 'acl'
    AND final_action = 'block'
  ) AS "Requests blocked by ACL policies",
  count_if(
    final_plugin = 'antiscan'
    AND final_action = 'block'
  ) AS "Requests blocked by scan protection policies",
  count_if(
    (final_plugin = 'waf'
    AND final_action = 'block')
    OR (final_plugin = 'cc'
    AND final_action = 'block')
    OR (final_plugin = 'acl'
    AND final_action = 'block')
    OR (final_plugin = 'antiscan'
    AND final_action = 'block')
  ) AS totalblock
GROUP BY
  host,
  user_id
HAVING
  (
    "Requests blocked by ACL policies" >= 0
    AND "Requests blocked by the Protection Rules Engine" >= 0
    AND "Requests blocked by HTTP flood protection policies" >= 0
    AND "Requests blocked by scan protection policies" >= 0
    AND totalblock > 10
  )
ORDER BY
  "Requests blocked by HTTP flood protection policies" DESC
LIMIT
  5

```

- **Time Range:** 5 minutes (relative)
- **Frequency:** 5 minutes
- **Trigger Condition:** `$0.totalblock>=500&&($0.Requests blocked by HTTP flood protection policies>=500)`
- **Notification Triggering Threshold:** 1
- **Notification Interval:** 5 minutes

- **Content :**

```
- [Time]: ${FireTime}
- [Uid]: ${Results[0].RawResults[0].user_id}
- Domain name: ${Results[0].RawResults[0].host}
- Service: WAF
- Requests that are blocked in the last 5 minutes: ${Results[0].RawResults[0].totalblock}
- Requests blocked by ACL policies: ${Results[0].RawResults[0].Requests blocked by ACL policies}
- Requests blocked by the Protection Rules Engine: ${Results[0].RawResults[0].Requests blocked by the Protection Rules Engine}
- Requests blocked by HTTP flood protection policies: ${Results[0].RawResults[0].Requests blocked by HTTP flood protection policies}
- Requests blocked by scan protection policies: ${Results[0].RawResults[0].Requests blocked by scan protection policies}
```

Requests blocked by scan protection policies in the last 5 minutes

The following parameter settings are recommended for this type of alerting:

- **Chart Name:** Requests blocked by scan protection policies
- **Query Statement :**


```

user_id: ID of your Alibaba Cloud account
SELECT
  user_id,
  host,
  count_if(
    final_plugin = 'waf'
    AND final_action = 'block'
  ) AS "Requests blocked by the Protection Rules Engine",
  count_if(
    final_plugin = 'cc'
    AND final_action = 'block'
  ) AS "Requests blocked by HTTP flood protection policies",
  count_if(
    final_plugin = 'acl'
    AND final_action = 'block'
  ) AS "Requests blocked by ACL policies",
  count_if(
    final_plugin = 'antiscan'
    AND final_action = 'block'
  ) AS "Requests blocked by scan protection policies",
  count_if(
    (final_plugin = 'waf'
    AND final_action = 'block')
    OR (final_plugin = 'cc'
    AND final_action = 'block')
    OR (final_plugin = 'acl'
    AND final_action = 'block')
    OR (final_plugin = 'antiscan'
    AND final_action = 'block')
  ) AS totalblock
GROUP BY
  host,
  user_id
HAVING
  (
    "Requests blocked by ACL policies" >= 0
    AND "Requests blocked by the Protection Rules Engine" >= 0
    AND "Requests blocked by HTTP flood protection policies" >= 0
    AND "Requests blocked by scan protection policies" >= 0
    AND totalblock > 10
  )
ORDER BY
  "Requests blocked by scan protection policies" DESC
LIMIT
  5

```

- **Time Range:** 5 minutes (relative)
- **Frequency:** 5 minutes
- **Trigger Condition:** `$0.totalblock>=500&&($0.Requests blocked by scan protection policies>=500)`
- **Notification Triggering Threshold:** 1
- **Notification Interval:** 5 minutes

- **Content :**

```
- [Time]: ${FireTime}
- [Uid]: ${Results[0].RawResults[0].user_id}
- Domain name: ${Results[0].RawResults[0].host}
- Service: WAF (International)
- Requests that are blocked in the last 5 minutes: ${Results[0].RawResults[0].totalblock}
- Requests blocked by ACL policies: ${Results[0].RawResults[0].Requests blocked by ACL po
licies}
- Requests blocked by the Protection Rules Engine: ${Results[0].RawResults[0].Requests bl
ocked by the Protection Rules Engine}
- Requests blocked by HTTP flood protection policies: ${Results[0].RawResults[0].Requests
blocked by HTTP flood protection policies}
- Requests blocked by scan protection policies: ${Results[0].RawResults[0].Requests block
ed by scan protection policies}
```

Attacks from a single IP address

The following parameter settings are recommended for this type of alerting:

- **Chart Name:** Attacks from a single IP address
- **Query Statement:**

```
user_id: ID of your Alibaba Cloud account
SELECT
  user_id,
  real_client_ip,
  concat(
    'Requests blocked by ACL policies:',
    cast(acblock AS varchar(10)),
    ' ',
    'Requests blocked by the Protection Rules Engine:',
    cast(wafblock AS varchar(10)),
    ' ',
    'Requests blocked by HTTP flood protection policies:',
    cast(acblock AS varchar(10))
  ) AS blockNum,
  totalblock,
  allRequest
FROM (
  SELECT
    user_id,
    real_client_ip,
    count_if(
      final_plugin = 'acl'
      AND final_action = 'block'
    ) AS acblock,
    count_if(
      final_plugin = 'waf'
      AND final_action = 'block'
    ) AS wafblock,
    count_if(
      final_plugin = 'cc'
      AND final action = 'block'
```

```

    ) AS ccblock,
    count_if(
      (
        final_plugin = 'acl'
        AND final_action = 'block'
      )
      OR (
        final_plugin = 'waf'
        AND final_action = 'block'
      )
      OR (
        final_plugin = 'cc'
        AND final_action = 'block'
      )
    ) AS totalblock,
    COUNT(*) AS allRequest
FROM      log
GROUP BY
  user_id,
  real_client_ip
HAVING
  totalblock > 1
ORDER BY
  totalblock DESC
LIMIT
  5
)

```

The chart contains the following fields: `real_client_ip` , `blockNum` , `totalblock` , and `allRequest` . `blockNum` includes Requests blocked by ACL policies , Requests blocked by the Protection Rules Engine , and Requests blocked by HTTP flood protection policies . `real_client_ip` indicates the IP address from which attacks are launched. `totalblock` indicates the total number of blocked requests. `allRequest` indicates the total number of requests. You can select one or more of these fields to configure alerts.

- **Time Range:** 5 minutes (relative)
- **Frequency:** 5 minutes
- **Trigger Condition:** `$0.totalblock >=500`
- **Notification Triggering Threshold:** 1
- **Notification Interval:** 5 minutes
- **Content :**

```

- [Time]: ${FireTime}
- [Uid]: ${Results[0].RawResults[0].user_id}
- Service: WAF
- Top 3 IP addresses from which attacks are mot frequently launched in the last 5 minutes
:
- ${Results[0].RawResults[0].real_client_ip}  (${Results[0].RawResults[0].blockNum})
- ${Results[0].RawResults[1].real_client_ip}  (${Results[0].RawResults[1].blockNum})
- ${Results[0].RawResults[2].real_client_ip}  (${Results[0].RawResults[2].blockNum})

```

Large number of domain names that are under attacks from a single IP address

The following parameter settings are recommended for this type of alerting:

- **Chart Name:** Large number of domain names attacked by a single IP address
- **Query Statement:**

```

user_id: ID of your Alibaba Cloud account
and not upstream_status :504
and not upstream_addr : '-'
and request_time_msec < 5000
and upstream_status :200
and not ua_browser :bot |
SELECT
    user_id,
    host,
    upstream_time,
    request_time,
    requestnum
FROM (
    SELECT
        user_id,
        host,
        round(avg(upstream_response_time), 2) * 1000 AS upstream_time,
        round(avg(request_time_msec), 2) AS request_time,
        COUNT(*) AS requestnum
    FROM log
    GROUP BY
        host,
        user_id
)
WHERE
    requestnum > 30
ORDER BY
    request_time DESC
LIMIT
    5

```

The chart contains the following fields: `real_client_ip`, `totalblock`, and `domainnum`. `real_client_ip` indicates the attack IP address. `totalblock` indicates the total number of blocked requests. `domainnum` indicates the number of domain names attacked by this IP address. You can select one or more of these fields to configure alerts. For example, you can specify `totalblock>500 && domainnum>5`. If the total number of attacks launched from an IP address reaches 500 and the number of domain names that are under the attacks exceeds 5 in the specified time range, an alert is triggered.

- **Time Range:** 5 minutes (relative)
- **Frequency:** 1 minute
- **Trigger Condition:** `$0.domainnum>=10`
- **Notification Triggering Threshold:** 1
- **Notification Interval:** 5 minutes

- **Content :**

```
- [Time]: ${FireTime}
- [Uid]: ${Results[0].RawResults[0].user_id}
- Service: WAF
- Attack IP address: ${Results[0].RawResults[0].real_client_ip}
- Attacked domain names: ${Results[0].RawResults[0].domainnum}
- Attack requests in the last 5 minutes: ${Results[0].RawResults[0].totalblock}
- Handle the alert at the earliest opportunity.
```

Abnormal average latency in the last 5 minutes

The following parameter settings are recommended for this type of alerting:

- **Chart Name:** Abnormal average latency
- **Query Statement :**

```
user_id: ID of your Alibaba Cloud account
and not upstream_status :504
and not upstream_addr : '-'
and request_time_msec < 5000
and upstream_status :200
and not ua_browser :bot |
SELECT
    user_id,
    host,
    upstream_time,
    request_time,
    requestnum
FROM (
    SELECT
        user_id,
        host,
        round(avg(upstream_response_time), 2) * 1000 AS upstream_time,
        round(avg(request_time_msec), 2) AS request_time,
        COUNT(*) AS requestnum
    FROM log
    GROUP BY
        host,
        user_id
)
WHERE
    requestnum > 30
ORDER BY
    request_time DESC
LIMIT
    5
```

- **Time Range:** 5 minutes (relative)
- **Frequency:** 5 minutes
- **Trigger Condition:** `$0.request_time>1000&& $0.requestnum>30`
- **Notification Triggering Threshold:** 2
- **Notification Interval:** 10 minutes

- **Content :**

```
- [Time]: ${FireTime}
- [Uid]: ${Results[0].RawResults[0].user_id}
- Domain name: ${Results[0].RawResults[0].host}
- Service: WAF (International)
- [Trigger Condition]: ${condition}
- Top 3 domain names that have the longest latency in the last 5 minutes. Unit of latency : milliseconds.
- Host1: ${Results[0].RawResults[0].host} Delay_time: ${Results[0].RawResults[0].upstream_time}
- Host2: ${Results[0].RawResults[1].host} Delay_time: ${Results[0].RawResults[1].upstream_time}
- Host3: ${Results[0].RawResults[2].host} Delay_time: ${Results[0].RawResults[2].upstream_time}
```

Abrupt decrease in traffic

The following parameter settings are recommended for this type of alerting:

- **Chart Name:** Abrupt decrease in traffic
- **Query Statement :**

```
user_id: ID of your Alibaba Cloud account
SELECT
  t1.user_id,
  t1.now1mQPS,
  t1.past1mQPS,
  de_ratio,
  t2.Rate_2XX,
  Rate_3XX,
  Rate_4XX,
  Rate_5XX,
  aveQPS
FROM (
  (
    SELECT
      user_id,
      round(c [1] / 60, 0) AS now1mQPS,
      round(c [2] / 60, 0) AS past1mQPS,
      round(
        100-round(c [1] / 60, 0) / round(c [2] / 60, 0) * 100,
        2
      ) AS de_ratio
    FROM (
      SELECT
        compare(t, 60) AS c,
        user_id
      FROM (
        SELECT
          COUNT(*) AS t,
          user_id
        FROM log
        GROUP BY
          user_id
```

```
        )
        GROUP BY
            user_id
    )
WHERE
    c [3] < 0.9
    AND (
        c [1] > 180
        or c [2] > 180
    )
) t1
JOIN (
    SELECT
        user_id,
        Rate_2XX,
        Rate_3XX,
        Rate_4XX,
        Rate_5XX,
        countall / 60 AS "aveQPS",
        status_2XX,
        status_3XX,
        status_4XX,
        status_5XX,
        countall
    FROM (
        SELECT
            user_id,
            round(
                round(status_2XX * 1.0000 / countall, 4) * 100,
                2
            ) AS Rate_2XX,
            round(
                round(status_3XX * 1.0000 / countall, 4) * 100,
                2
            ) AS Rate_3XX,
            round(
                round(status_4XX * 1.0000 / countall, 4) * 100,
                2
            ) AS Rate_4XX,
            round(
                round(status_5XX * 1.0000 / countall, 4) * 100,
                2
            ) AS Rate_5XX,
            status_2XX,
            status_3XX,
            status_4XX,
            status_5XX,
            countall
        FROM (
            SELECT
                user_id,
                count_if(
                    status >= 200
                    AND status < 300
```

```
        ) AS status_2XX,
        count_if(
            status >= 300
            AND status < 400
        ) AS status_3XX,
        count_if (
            status >= 400
            AND status < 500
            AND status <> 444
            AND status <> 405
        ) AS status_4XX,
        count_if(
            status >= 500
            AND status < 600
        ) AS status_5XX,
        COUNT(*) AS countall
    FROM          log
    GROUP BY
        user_id
)
)
WHERE
    countall > 0
) t2 ON t1.user_id = t2.user_id
)
ORDER BY
    de_ratio DESC
LIMIT
    5
```

- **Time Range:** 1 minute (relative)
- **Frequency:** 1 minute
- **Trigger Condition:** `$0.de_ratio>50&& $0.now1mqps>20`
- **Notification Triggering Threshold:** 1
- **Notification Interval:** 5 minutes
- **Content:**

```
- [Time]: ${FireTime}
- [UID]:${Results[0].RawResults[0].user_id}
- Service: WAF
- Average QPS in the last 1 minute: ${Results[0].RawResults[0].now1mqps}
- [Trigger condition (abrupt decrease rate of traffic & QPS)]:${condition}
- Abrupt decrease rate of QPS: ${Results[0].RawResults[0].de_ratio}%
- Percentage of 2xx status codes: ${Results[0].RawResults[0].rate_2xx}%
- Percentage of 3xx status codes: ${Results[0].RawResults[0].Rate_3XX}%
- Percentage of 4xx status codes: ${Results[0].RawResults[0].Rate_4XX}%
- Percentage of 5xx status codes: ${Results[0].RawResults[0].Rate_5XX}%
```

4.6. Common monitoring metrics

This topic describes the common metrics that you can use to query and analyze logs collected by Log Service for WAF. You can use these metrics to configure alerts and monitor exceptions in your workload as needed. This topic also provides the recommended alert thresholds of metrics and suggestions on handling metric exceptions.

Metric	Description	Recommended threshold	Suggestion
200	The server has processed the request and returned the requested data.	Before you initialize your workloads, set the alert threshold to 90% for status code 200. You can adjust the threshold as needed.	If the percentage of code 200 is lower than the specified threshold, identify the reason. For example, this is because the percentage of another status code has increased.
request_time_msec	Time period between the time when the client sends a request and the time when the client receives a response.	Set the alert thresholds based on the time required for actual service requests.	If it takes a long time to receive responses from a domain name, check the network connectivity between the client and WAF and that between WAF and the origin servers, and make sure that the origin servers respond properly.
upstream_response_time	The time period between the time when WAF sends data to the origin server and the time when WAF receives a response from the origin server.		
ssl_handshake_time	The time required for an SSL handshake between the client and WAF during an HTTPS request.		
status:302 and block_action:tmd/status:200 and block_action:tmd	The status code indicates whether CAPTCHA is triggered. Code 302 indicates that CAPTCHA is triggered, and code 200 indicates that CAPTCHA is not triggered and the HTTP flood protection is triggered.		<ul style="list-style-type: none"> If the alert threshold is reached, find out whether the domain name is under HTTP flood attacks and customize rules to block the attacks. Check for server exceptions, such as, a large number of 5xx status codes or 4xx status codes.
status:200 and block_action:antifraud	A request is blocked by data risk control.		Test the alert rule before you apply it. If you receive this alert frequently, contact the Alibaba Cloud R&D team to adjust the alert threshold.


Metric	Description	Recommended threshold	Suggestion
status:404	The server cannot find the requested resources.	When you initialize your workloads, we recommend that you set the alert threshold for status code percentage to a value from 5% to 10%. You can adjust the threshold based on the traffic blocked by WAF.	<p>Query the source IP addresses that trigger the alert.</p> <ul style="list-style-type: none"> If only one IP address triggers the alert, a malicious user may have started a path traversal on your server. If multiple IP addresses trigger the alert, check whether the server works properly and whether any files are missing.
status:405	A request is blocked by either web application protection rules or HTTP ACL policy rules.		Use the log analysis feature to analyze the blocked request and the rule that is used to block the request, and find out whether this is a false positive.
status:444	A request is blocked by custom HTTP flood protection rules.		<ul style="list-style-type: none"> If the alert threshold is reached, find out whether the domain name is under HTTP flood attacks and customize rules to block the attacks. If the blocked request is not an attack but an API call, you can adjust the threshold or allow API calls on specified servers.
status:499	After a client sends a request, the server does not return data. After the maximum wait time of the client is reached, the client disconnects, and the server returns this status code.		<ul style="list-style-type: none"> Check for exceptions on the origin server, such as, slow responses and a large number of slow queries on the database. Check whether attacks have consumed all resources on the origin server.
status:500	A request cannot be processed due to the 500 Internal Server Error.		We recommend that you check the loads and database status of the origin server.

Metric	Description	Recommended threshold	Suggestion
status:502	The server is used as a gateway or a proxy and receives invalid responses from the upstream server due to a 502 Bad Gateway error. The origin server does not respond due to low quality performance of the back-to-origin network or the fact that back-to-origin requests are blocked by access control policies configured for the origin server.		<ul style="list-style-type: none">• Check the back-to-origin network quality, the access control policies on the origin server, and the loads and database status of the origin server.• Check whether the origin server has blocked requests from the back-to-origin IP address of WAF.
status:503	The service is unavailable due to overloads or maintenance needs.		Check for exceptions on the origin server.
status:504	The server serves as a gateway or proxy but fails to receive requests from the upstream server in time. The 504 Gateway Timeout error occurred.		Possible causes include: <ul style="list-style-type: none">• The server fails to respond due to overload.• The origin server does not reset after it discards requests.• The protocol-based communication fails.

4.7. Query statements

This topic describes the query statements that are used to configure monitoring and alerting for common metrics based on Log Service for WAF.

You can use Log Service for WAF to monitor the following metrics:

 **Note** You can click a metric to view the involved query statement. For more information about the metrics, see [Common monitoring metrics](#).

- [request_time_msec](#)
- [upstream_response_time](#)
- [status:200](#)
- [status:302 or 200 and final_plugin:'cc'](#)
- [status:200 and final_plugin:'antifraud'](#)
- [status:404](#)

- status:405 and waf_action:'block'
- status:405 and final_plugin:'acl'
- status:444
- status:499
- status:500
- status:502
- status:503
- status:504

request_time_msec

The duration between the time when the client sends a request and the time when the client receives a response to the request.

```
* |
SELECT
  user_id,
  host,
  round(
    round(request_time_cnt * 1.0000 / countall, 4) * 100,
    2
  ) AS percent
FROM (
  SELECT
    user_id,
    host,
    count_if(request_time_msec > 500) AS request_time_cnt,
    COUNT(*) AS countall
  FROM log
  GROUP BY
    user_id,
    host
)
GROUP BY
  user_id,
  host,
  percent
```

upstream_response_time

The duration between the time when WAF forwards a request to an origin server and the time when WAF receives a response to the request.

```
* |
SELECT
  user_id,
  host,
  round(
    round(
      upstream_response_time_cnt * 1.0000 / countall,
      4
    ) * 100,
    2
  ) AS percent
FROM (
  SELECT
    user_id,
    host,
    count_if(upstream_response_time > 500) AS upstream_response_time_cnt,
    COUNT(*) AS countall
  FROM      log
  GROUP BY
    user_id,
    host
)
GROUP BY
  user_id,
  host,
  percent
```

status:200

The server has processed the request and returned the requested data.

```
* |
SELECT
  user_id,
  host AS "Domain name",
  Rate_200 AS "Percentage of 200 status code",
  Rate_302 AS "Percentage of 302 status code",
  Rate_404 AS "Percentage of 404 status code",
  Rate_405 AS "Percentage of 405 status code",
  Rate_444 AS "Percentage of 444 status code",
  Rate_499 AS "Percentage of 499 status code",
  Rate_500 AS "Percentage of 500 status code",
  Rate_502 AS "Percentage of 502 status code",
  Rate_503 AS "Percentage of 503 status code",
  Rate_504 AS "Percentage of 504 status code",
  countall / 60 AS "aveQPS",
  status_200,
  status_302,
  status_404,
  status_405,
  status_444,
  status_499,
  status_500,
  status_502
```

```
status_502,  
status_503,  
status_504,  
countall  
FROM (  
  SELECT  
    user_id,  
    host,  
    round(  
      round(status_200 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_200,  
    round(  
      round(status_302 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_302,  
    round(  
      round(status_404 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_404,  
    round(  
      round(status_405 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_405,  
    round(  
      round(status_405 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_444,  
    round(  
      round(status_405 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_499,  
    round(  
      round(status_500 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_500,  
    round(  
      round(status_502 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_502,  
    round(  
      round(status_503 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_503,  
    round(  
      round(status_504 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_504,  
    status_200,  
    status_302,  
    status_404,  
    status_405,  
    status_444,  
    status_499,  
    status_500,
```

```

        status_502,
        status_503,
        status_504,
        countall
    FROM      (
        SELECT
            user_id,
            host,
            count_if(status = 200) AS status_200,
            count_if(status = 302) AS status_302,
            count_if(status = 404) AS status_404,
            count_if(status = 405) AS status_405,
            count_if(status = 444) AS status_444,
            count_if(status = 499) AS status_499,
            count_if(status = 500) AS status_500,
            count_if(status = 502) AS status_502,
            count_if(status = 503) AS status_503,
            count_if(status = 504) AS status_504,
            COUNT(*) AS countall
        FROM      log
        GROUP BY
            user_id,
            host
    )
)
WHERE
    countall > 120
ORDER BY
    Rate_200 DESC
LIMIT
    5

```

status:302 or 200 and final_plugin:'cc'

JavaScript CAPTCHA validation is triggered.

```

* |
SELECT
    user_id,
    host AS "Domain name",
    Rate_200 AS "Percentage of 200 status code",
    Rate_302 AS "Percentage of 302 status code",
    Rate_404 AS "Percentage of 404 status code",
    Rate_405 AS "Percentage of 405 status code",
    Rate_444 AS "Percentage of 444 status code",
    Rate_499 AS "Percentage of 499 status code",
    Rate_500 AS "Percentage of 500 status code",
    Rate_502 AS "Percentage of 502 status code",
    Rate_503 AS "Percentage of 503 status code",
    Rate_504 AS "Percentage of 504 status code",
    countall / 60 AS "aveQPS",
    status_200,
    status_302,
    status_404

```

```
status_404,  
status_405,  
status_444,  
status_499,  
status_500,  
status_502,  
status_503,  
status_504,  
countall  
FROM (  
  SELECT  
    user_id,  
    host,  
    round(  
      round(status_200 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_200,  
    round(  
      round(status_302 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_302,  
    round(  
      round (status_404 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_404,  
    round(  
      round (status_405 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_405,  
    round(  
      round (status_405 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_444,  
    round(  
      round (status_405 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_499,  
    round(  
      round(status_500 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_500,  
    round(  
      round(status_502 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_502,  
    round(  
      round(status_503 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_503,  
    round(  
      round(status_504 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_504,  
    status_200,  
    status_302,
```



```
status_404,  
status_405,  
status_444,  
status_499,  
status_500,  
status_502,  
status_503,  
status_504,  
countall  
FROM      (  
    SELECT  
        user_id,  
        host,  
        count_if(  
            status = 200  
            AND final_plugin = 'cc'  
        ) AS status_200,  
        count_if(  
            status = 302  
            AND final_plugin = 'cc'  
        ) AS status_302,  
        count_if(status = 404) AS status_404,  
        count_if(status = 405) AS status_405,  
        count_if(status = 444) AS status_444,  
        count_if(status = 499) AS status_499,  
        count_if(status = 500) AS status_500,  
        count_if(status = 502) AS status_502,  
        count_if(status = 503) as status_503,  
        count_if(status = 504) AS status_504,  
        COUNT(*) AS countall  
    FROM      log  
    GROUP BY  
        user_id,  
        host  
    )  
)  
WHERE  
    countall > 120  
ORDER BY  
    Rate_200 DESC  
LIMIT  
5
```

status:200 and final_plugin:'antifraud'

The request is blocked by data risk control rules.

```
* |  
SELECT  
    user_id,  
    host AS "Domain name",  
    Rate_200 AS "Percentage of 200 status code",  
    Rate_302 AS "Percentage of 302 status code",  
    Rate_404 AS "Percentage of 404 status code".
```

```
Rate_404 AS "Percentage of 404 status code",
Rate_405 AS "Percentage of 405 status code",
Rate_444 AS "Percentage of 444 status code",
Rate_499 AS "Percentage of 499 status code",
Rate_500 AS "Percentage of 500 status code",
Rate_502 AS "Percentage of 502 status code",
Rate_503 AS "Percentage of 503 status code",
Rate_504 AS "Percentage of 504 status code",
countall / 60 AS "aveQPS",
status_200,
status_302,
status_404,
status_405,
status_444,
status_499,
status_500,
status_502,
status_503,
status_504,
countall
FROM (
  SELECT
    user_id,
    host,
    round(
      round(status_200 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_200,
    round(
      round(status_302 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_302,
    round(
      round (status_404 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_404,
    round(
      round (status_405 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_405,
    round(
      round (status_405 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_444,
    round(
      round (status_405 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_499,
    round(
      round(status_500 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_500,
    round(
      round(status_502 * 1.0000 / countall, 4) * 100,
      2
```

```
) AS Rate_502,
round(
  round(status_503 * 1.0000 / countall, 4) * 100,
  2
) AS Rate_503,
round(
  round(status_504 * 1.0000 / countall, 4) * 100,
  2
) AS Rate_504,
status_200,
status_302,
status_404,
status_405,
status_444,
status_499,
status_500,
status_502,
status_503,
status_504,
countall
FROM      (
  SELECT
    user_id,
    host,
    count_if(
      status = 200
      AND final_plugin = 'antifraud'
    ) AS status_200,
    count_if(status = 302) AS status_302,
    count_if(status = 404) AS status_404,
    count_if(status = 405) AS status_405,
    count_if(status = 444) AS status_444,
    count_if(status = 499) AS status_499,
    count_if(status = 500) AS status_500,
    count_if(status = 502) AS status_502,
    count_if(status = 503) AS status_503,
    count_if(status = 504) AS status_504,
    COUNT(*) AS countall
  FROM      log
  GROUP BY
    user_id,
    host
)
)
WHERE
  countall > 120
ORDER BY
  Rate_200 DESC
LIMIT
5
```

status:404

The server failed to find the requested resources.

```
* |
SELECT
  user_id,
  host AS "Domain name",
  Rate_200 AS "Percentage of 200 status code",
  Rate_302 AS "Percentage of 302 status code",
  Rate_404 AS "Percentage of 404 status code",
  Rate_405 AS "Percentage of 405 status code",
  Rate_500 AS "Percentage of 500 status code",
  Rate_502 AS "Percentage of 502 status code",
  Rate_503 AS "Percentage of 503 status code",
  Rate_504 AS "Percentage of 504 status code",
  countall / 60 AS "aveQPS",
  status_200,
  status_302,
  status_404,
  status_405,
  status_500,
  status_502,
  status_503,
  status_504,
  countall
FROM (
  SELECT
    user_id,
    host,
    round(
      round(status_200 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_200,
    round(
      round(status_302 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_302,
    round(
      round(status_404 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_404,
    round(
      round(status_405 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_405,
    round(
      round(status_500 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_500,
    round(
      round(status_502 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_502,
    round(
      round(status_503 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_503,
    round(
      round(status_504 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_504,
    countall
  )
```

```

2
) AS Rate_503,
round(
    round(status_504 * 1.0000 / countall, 4) * 100,
    2
) AS Rate_504,
status_200,
status_302,
status_404,
status_405,
status_500,
status_502,
status_503,
status_504,
countall
FROM (
    SELECT
        user_id,
        host,
        count_if(status = 200) AS status_200,
        count_if(status = 302) AS status_302,
        count_if(status = 404) AS status_404,
        count_if(status = 405) AS status_405,
        count_if(status = 499) AS status_499,
        count_if(status = 500) AS status_500,
        count_if(status = 502) AS status_502,
        count_if(status = 503) AS status_503,
        count_if(status = 504) AS status_504,
        COUNT(*) AS countall
    FROM log
    GROUP BY
        user_id,
        host
)
)
WHERE
    countall > 120
ORDER BY
    Rate_404 DESC
LIMIT
    5

```

status:405 and waf_action:'block'

The request is blocked by the Protection Rules Engine.

```

* |
SELECT
    user_id,
    host AS "Domain name",
    Rate_200 AS "Percentage of 200 status code",
    Rate_302 AS "Percentage of 302 status code",
    Rate_404 AS "Percentage of 404 status code",
    Rate_405 AS "Percentage of 405 status code",

```

```
Rate_444 AS "Percentage of 444 status code",
Rate_499 AS "Percentage of 499 status code",
Rate_500 AS "Percentage of 500 status code",
Rate_502 AS "Percentage of 502 status code",
Rate_503 AS "Percentage of 503 status code",
Rate_504 AS "Percentage of 504 status code",
countall / 60 AS "aveQPS",
status_200,
status_302,
status_404,
status_405,
status_444,
status_499,
status_500,
status_502,
status_503,
status_504,
countall
FROM(
    SELECT
        user_id,
        host,
        round(
            round(status_200 * 1.0000 / countall, 4) * 100,
            2
        ) AS Rate_200,
        round(
            round(status_302 * 1.0000 / countall, 4) * 100,
            2
        ) AS Rate_302,
        round(
            round(status_404 * 1.0000 / countall, 4) * 100,
            2
        ) AS Rate_404,
        round(
            round(status_405 * 1.0000 / countall, 4) * 100,
            2
        ) AS Rate_405,
        round(
            round(status_444 * 1.0000 / countall, 4) * 100,
            2
        ) AS Rate_444,
        round(
            round(status_499 * 1.0000 / countall, 4) * 100,
            2
        ) AS Rate_499,
        round(
            round(status_500 * 1.0000 / countall, 4) * 100,
            2
        ) AS Rate_500,
        round(
            round(status_502 * 1.0000 / countall, 4) * 100,
            2
        ) AS Rate_502,
```

```
round(
  round(status_503 * 1.0000 / countall, 4) * 100,
  2
) AS Rate_503,
round(
  round(status_504 * 1.0000 / countall, 4) * 100,
  2
) AS Rate_504,
status_200,
status_302,
status_404,
status_405,
status_444,
status_499,
status_500,
status_502,
status_503,
status_504,
countall
FROM (
  SELECT
    user_id,
    host,
    count_if(status = 200) AS status_200,
    count_if(status = 302) AS status_302,
    count_if(status = 404) AS status_404,
    count_if(
      status = 405
      and waf_action = 'block'
    ) AS status_405,
    count_if(status = 444) AS status_444,
    count_if(status = 499) AS status_499,
    count_if(status = 500) AS status_500,
    count_if(status = 502) AS status_502,
    count_if(status = 503) AS status_503,
    count_if(status = 504) AS status_504,
    COUNT(*) AS countall
  FROM log
  GROUP BY
    user_id,
    host
)
WHERE
  countall > 120
ORDER BY
  Rate_405 DESC
LIMIT
  5
```

status:405 and final_plugin:'acl'

The request is blocked by the blacklist or custom protection rules (ACLs).

```

* |
SELECT
  user_id,
  host AS "Domain name",
  Rate_200 AS "Percentage of 200 status code",
  Rate_302 AS "Percentage of 302 status code",
  Rate_404 AS "Percentage of 404 status code",
  Rate_405 AS "Percentage of 405 status code",
  Rate_444 AS "Percentage of 444 status code",
  Rate_499 AS "Percentage of 499 status code",
  Rate_500 AS "Percentage of 500 status code",
  Rate_502 AS "Percentage of 502 status code",
  Rate_503 AS "Percentage of 503 status code",
  Rate_504 AS "Percentage of 504 status code",
  countall / 60 AS "aveQPS",
  status_200,
  status_302,
  status_404,
  status_405,
  status_444,
  status_499,
  status_500,
  status_502,
  status_503,
  status_504,
  countall
FROM(
  SELECT
    user_id,
    host,
    round(
      round(status_200 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_200,
    round(
      round(status_302 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_302,
    round(
      round(status_404 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_404,
    round(
      round(status_405 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_405,
    round(
      round(status_405 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_444,
    round(
      round(status_405 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_499,
    round(
      round(status_500 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_500,
    round(
      round(status_502 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_502,
    round(
      round(status_503 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_503,
    round(
      round(status_504 * 1.0000 / countall, 4) * 100,
      2
    ) AS Rate_504,
    countall
  )

```



```

round(
    round(status_500 * 1.0000 / countall, 4) * 100,
    2
) AS Rate_500,
round(
    round(status_502 * 1.0000 / countall, 4) * 100,
    2
) AS Rate_502,
round(
    round(status_503 * 1.0000 / countall, 4) * 100,
    2
) AS Rate_503,
round(
    round(status_504 * 1.0000 / countall, 4) * 100,
    2
) AS Rate_504,
status_200,
status_302,
status_404,
status_405,
status_444,
status_499,
status_500,
status_502,
status_503,
status_504,
countall
FROM (
    SELECT
        user_id,
        host,
        count_if(status = 200) AS status_200,
        count_if(status = 302) AS status_302,
        count_if(status = 404) AS status_404,
        count_if(
            status = 405
            and final_plugin = 'acl'
        ) AS status_405,
        count_if(status = 444) AS status_444,
        count_if(status = 499) AS status_499,
        count_if(status = 500) AS status_500,
        count_if(status = 502) AS status_502,
        count_if(status = 503) AS status_503,
        count_if(status = 504) AS status_504,
        COUNT(*) AS countall
    FROM log
    GROUP BY
        user_id,
        host
)
)
WHERE
    countall > 120
ORDER BY
    Rate_405 DESC

```

```
LIMIT  
5
```

status:444

The request is blocked by HTTP flood protection rules.

```
* |  
select  
  user_id,  
  host AS "Domain name",  
  Rate_200 AS "Percentage of 200 status code",  
  Rate_302 AS "Percentage of 302 status code",  
  Rate_404 AS "Percentage of 404 status code",  
  Rate_405 AS "Percentage of 405 status code",  
  Rate_444 AS "Percentage of 444 status code",  
  Rate_499 AS "Percentage of 499 status code",  
  Rate_500 AS "Percentage of 500 status code",  
  Rate_502 AS "Percentage of 502 status code",  
  Rate_503 AS "Percentage of 503 status code",  
  Rate_504 AS "Percentage of 504 status code",  
  countall / 60 AS "aveQPS",  
  status_200,  
  status_302,  
  status_404,  
  status_405,  
  status_444,  
  status_499,  
  status_500,  
  status_502,  
  status_503,  
  status_504,  
  countall  
FROM(  
  SELECT  
    user_id,  
    host,  
    round(  
      round(status_200 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_200,  
    round(  
      round(status_302 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_302,  
    round(  
      round(status_404 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_404,  
    round(  
      round(status_405 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_405,  
    round(  
      round(status_444 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_444,  
    round(  
      round(status_499 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_499,  
    round(  
      round(status_500 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_500,  
    round(  
      round(status_502 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_502,  
    round(  
      round(status_503 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_503,  
    round(  
      round(status_504 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_504,  
    countall  
  )
```

```
round(
    round (status_405 * 1.0000 / countall, 4) * 100,
    2
) AS Rate_444,
round(
    round (status_405 * 1.0000 / countall, 4) * 100,
    2
) AS Rate_499,
round(
    round(status_500 * 1.0000 / countall, 4) * 100,
    2
) AS Rate_500,
round(
    round(status_502 * 1.0000 / countall, 4) * 100,
    2
) AS Rate_502,
round(
    round(status_503 * 1.0000 / countall, 4) * 100,
    2
) AS Rate_503,
round(
    round(status_504 * 1.0000 / countall, 4) * 100,
    2
) AS Rate_504,
status_200,
status_302,
status_404,
status_405,
status_444,
status_499,
status_500,
status_502,
status_503,
status_504,
countall
FROM      (
    SELECT
        user_id,
        host,
        count_if(status = 200) AS status_200,
        count_if(status = 302) AS status_302,
        count_if(status = 404) AS status_404,
        count_if(status = 405) AS status_405,
        count_if(status = 444) AS status_444,
        count_if(status = 499) AS status_499,
        count_if(status = 500) AS status_500,
        count_if(status = 502) AS status_502,
        count_if(status = 503) AS status_503,
        count_if(status = 504) AS status_504,
        COUNT(*) AS countall
    FROM      log
    GROUP BY
        user_id,
        host
)
```

```
)  
WHERE  
    countall > 120  
ORDER BY  
    Rate_444 DESC  
LIMIT  
    5
```

status:499

The requested data is not returned because the server connection timed out and the client closed the connection. The server returns the 499 status code to the client.

```
* |  
SELECT  
    user_id,  
    host AS "Domain name",  
    Rate_200 AS "Percentage of 200 status code",  
    Rate_302 AS "Percentage of 302 status code",  
    Rate_404 AS "Percentage of 404 status code",  
    Rate_405 AS "Percentage of 405 status code",  
    Rate_444 AS "Percentage of 444 status code",  
    Rate_499 AS "Percentage of 499 status code",  
    Rate_500 AS "Percentage of 500 status code",  
    Rate_502 AS "Percentage of 502 status code",  
    Rate_503 AS "Percentage of 503 status code",  
    Rate_504 AS "Percentage of 504 status code",  
    countall / 60 AS "aveQPS",  
    status_200,  
    status_302,  
    status_404,  
    status_405,  
    status_444,  
    status_499,  
    status_500,  
    status_502,  
    status_503,  
    status_504,  
    countall  
FROM(  
    SELECT  
        user_id,  
        host,  
        round(  
            round(status_200 * 1.0000 / countall, 4) * 100,  
            2  
        ) AS Rate_200,  
        round(  
            round(status_302 * 1.0000 / countall, 4) * 100,  
            2  
        ) AS Rate_302,  
        round(  
            round(status_404 * 1.0000 / countall, 4) * 100,  
            2  
        ) AS Rate_404,  
        round(  
            round(status_405 * 1.0000 / countall, 4) * 100,  
            2  
        ) AS Rate_405,  
        round(  
            round(status_444 * 1.0000 / countall, 4) * 100,  
            2  
        ) AS Rate_444,  
        round(  
            round(status_499 * 1.0000 / countall, 4) * 100,  
            2  
        ) AS Rate_499,  
        round(  
            round(status_500 * 1.0000 / countall, 4) * 100,  
            2  
        ) AS Rate_500,  
        round(  
            round(status_502 * 1.0000 / countall, 4) * 100,  
            2  
        ) AS Rate_502,  
        round(  
            round(status_503 * 1.0000 / countall, 4) * 100,  
            2  
        ) AS Rate_503,  
        round(  
            round(status_504 * 1.0000 / countall, 4) * 100,  
            2  
        ) AS Rate_504,  
        countall  
    )
```

```

        2
    ) AS Rate_404,
    round(
        round (status_405 * 1.0000 / countall, 4) * 100,
        2
    ) AS Rate_405,
    round(
        round (status_405 * 1.0000 / countall, 4) * 100,
        2
    ) AS Rate_444,
    round(
        round (status_405 * 1.0000 / countall, 4) * 100,
        2
    ) AS Rate_499,
    round(
        round(status_500 * 1.0000 / countall, 4) * 100,
        2
    ) AS Rate_500,
    round(
        round(status_502 * 1.0000 / countall, 4) * 100,
        2
    ) AS Rate_502,
    round(
        round(status_503 * 1.0000 / countall, 4) * 100,
        2
    ) AS Rate_503,
    round(
        round(status_504 * 1.0000 / countall, 4) * 100,
        2
    ) AS Rate_504,
    status_200,
    status_302,
    status_404,
    status_405,
    status_444,
    status_499,
    status_500,
    status_502,
    status_503,
    status_504,
    countall
FROM      (
    SELECT
        user_id,
        host,
        count_if(status = 200) AS status_200,
        count_if(status = 302) AS status_302,
        count_if(status = 404) AS status_404,
        count_if(status = 405) AS status_405,
        count_if(status = 444) AS status_444,
        count_if(status = 499) AS status_499,
        count_if(status = 500) AS status_500,
        count_if(status = 502) AS status_502,
        count_if(status = 503) AS status_503,
        count_if(status = 504) AS status_504.

```

```
countall(status_499 AS rate_499, AS countall_499,
COUNT(*) AS countall
FROM      log
GROUP BY
    user_id,
    host
)
)
WHERE
    countall > 120
ORDER BY
    Rate_499 DESC
LIMIT
    5
```

status:500

The server failed to process the request because an internal error occurred on the server.

```
* |
SELECT
    user_id,
    host AS "Domain name",
    Rate_200 AS "Percentage of 200 status code",
    Rate_302 AS "Percentage of 302 status code",
    Rate_404 AS "Percentage of 404 status code",
    Rate_405 AS "Percentage of 405 status code",
    Rate_444 AS "Percentage of 444 status code",
    Rate_499 AS "Percentage of 499 status code",
    Rate_500 AS "Percentage of 500 status code",
    Rate_502 AS "Percentage of 502 status code",
    Rate_503 AS "Percentage of 503 status code",
    Rate_504 AS "Percentage of 504 status code",
    countall / 60 AS "aveQPS",
    status_200,
    status_302,
    status_404,
    status_405,
    status_444,
    status_499,
    status_500,
    status_502,
    status_503,
    status_504,
    countall
FROM(
    SELECT
        user_id,
        host,
        round(
            round(status_200 * 1.0000 / countall, 4) * 100,
            2
        ) AS Rate_200,
        round(
            round(status_302 * 1.0000 / countall, 4) * 100,
```

```

round(status_302 * 1.0000 / countall, 4) * 100,
2
) AS Rate_302,
round(
round(status_404 * 1.0000 / countall, 4) * 100,
2
) AS Rate_404,
round(
round(status_405 * 1.0000 / countall, 4) * 100,
2
) AS Rate_405,
round(
round(status_405 * 1.0000 / countall, 4) * 100,
2
) AS Rate_444,
round(
round(status_405 * 1.0000 / countall, 4) * 100,
2
) AS Rate_499,
round(
round(status_500 * 1.0000 / countall, 4) * 100,
2
) AS Rate_500,
round(
round(status_502 * 1.0000 / countall, 4) * 100,
2
) AS Rate_502,
round(
round(status_503 * 1.0000 / countall, 4) * 100,
2
) AS Rate_503,
round(
round(status_504 * 1.0000 / countall, 4) * 100,
2
) AS Rate_504,
status_200,
status_302,
status_404,
status_405,
status_444,
status_499,
status_500,
status_502,
status_503,
status_504,
countall
FROM (
SELECT
user_id,
host,
count_if(status = 200) AS status_200,
count_if(status = 302) AS status_302,
count_if(status = 404) AS status_404,
count_if(status = 405) AS status_405,
count_if(status = 444) AS status_444,

```

```
count_if(status = 499) AS status_499,  
count_if(status = 500) AS status_500,  
count_if(status = 502) AS status_502,  
count_if(status = 503) AS status_503,  
count_if(status = 504) AS status_504,  
COUNT(*) AS countall  
FROM      log  
GROUP BY  
    user_id,  
    host  
)  
)  
WHERE  
    countall > 120  
ORDER BY  
    Rate_500 DESC  
LIMIT  
5
```

status:502

The WAF instance is used as a gateway or a proxy and receives an invalid response from the origin server. The origin server does not respond because the back-to-origin network is unstable or the back-to-origin request is blocked based on access control policies that are configured for the origin server.

```
* |  
SELECT  
    user_id,  
    host AS "Domain name",  
    Rate_200 AS "Percentage of 200 status code",  
    Rate_302 AS "Percentage of 302 status code",  
    Rate_404 AS "Percentage of 404 status code",  
    Rate_405 AS "Percentage of 405 status code",  
    Rate_444 AS "Percentage of 444 status code",  
    Rate_499 AS "Percentage of 499 status code",  
    Rate_500 AS "Percentage of 500 status code",  
    Rate_502 AS "Percentage of 502 status code",  
    Rate_503 AS "Percentage of 503 status code",  
    Rate_504 AS "Percentage of 504 status code",  
    countall / 60 AS "aveQPS",  
    status_200,  
    status_302,  
    status_404,  
    status_405,  
    status_444,  
    status_499,  
    status_500,  
    status_502,  
    status_503,  
    status_504,  
    countall  
FROM(  
    SELECT
```



```
user_id,  
host,  
round(  
    round(status_200 * 1.0000 / countall, 4) * 100,  
    2  
) AS Rate_200,  
round(  
    round(status_302 * 1.0000 / countall, 4) * 100,  
    2  
) AS Rate_302,  
round(  
    round (status_404 * 1.0000 / countall, 4) * 100,  
    2  
) AS Rate_404,  
round(  
    round (status_405 * 1.0000 / countall, 4) * 100,  
    2  
) AS Rate_405,  
round(  
    round (status_405 * 1.0000 / countall, 4) * 100,  
    2  
) AS Rate_444,  
round(  
    round (status_405 * 1.0000 / countall, 4) * 100,  
    2  
) AS Rate_499,  
round(  
    round(status_500 * 1.0000 / countall, 4) * 100,  
    2  
) AS Rate_500,  
round(  
    round(status_502 * 1.0000 / countall, 4) * 100,  
    2  
) AS Rate_502,  
round(  
    round(status_503 * 1.0000 / countall, 4) * 100,  
    2  
) AS Rate_503,  
round(  
    round(status_504 * 1.0000 / countall, 4) * 100,  
    2  
) AS Rate_504,  
status_200,  
status_302,  
status_404,  
status_405,  
status_444,  
status_499,  
status_500,  
status_502,  
status_503,  
status_504,  
countall  
FROM      (  
SELECT
```

```
        user_id,  
        host,  
        count_if(status = 200) AS status_200,  
        count_if(status = 302) AS status_302,  
        count_if(status = 404) AS status_404,  
        count_if(status = 405) AS status_405,  
        count_if(status = 444) AS status_444,  
        count_if(status = 499) AS status_499,  
        count_if(status = 500) AS status_500,  
        count_if(status = 502) AS status_502,  
        count_if(status = 503) AS status_503,  
        count_if(status = 504) AS status_504,  
        COUNT(*) AS countall  
FROM      log  
GROUP BY  
        user_id,  
        host  
    )  
    )  
WHERE  
    countall > 120  
ORDER BY  
    Rate_502 DESC  
LIMIT  
5
```

status:503

The service is unavailable because the server is overloaded or being maintained.

```
* |  
SELECT  
    user_id,  
    host AS "Domain name",  
    Rate_200 as "Percentage of 200 status code",  
    Rate_302 as "Percentage of 302 status code",  
    Rate_404 as "Percentage of 404 status code",  
    Rate_405 as "Percentage of 405 status code",  
    Rate_444 as "Percentage of 444 status code",  
    Rate_499 as "Percentage of 499 status code",  
    Rate_500 as "Percentage of 500 status code",  
    Rate_502 as "Percentage of 502 status code",  
    Rate_503 as "Percentage of 503 status code",  
    Rate_504 as "Percentage of 504 status code",  
    countall / 60 as "aveQPS",  
    status_200,  
    status_302,  
    status_404,  
    status_405,  
    status_444,  
    status_499,  
    status_500,  
    status_502,  
    status_503
```

```
status_503,  
status_504,  
countall  
FROM(  
  SELECT  
    user_id,  
    host,  
    round(  
      round(status_200 * 1.0000 / countall, 4) * 100,  
      2  
    ) as Rate_200,  
    round(  
      round(status_302 * 1.0000 / countall, 4) * 100,  
      2  
    ) as Rate_302,  
    round(  
      round(status_404 * 1.0000 / countall, 4) * 100,  
      2  
    ) as Rate_404,  
    round(  
      round(status_405 * 1.0000 / countall, 4) * 100,  
      2  
    ) as Rate_405,  
    round(  
      round(status_405 * 1.0000 / countall, 4) * 100,  
      2  
    ) as Rate_444,  
    round(  
      round(status_405 * 1.0000 / countall, 4) * 100,  
      2  
    ) as Rate_499,  
    round(  
      round(status_500 * 1.0000 / countall, 4) * 100,  
      2  
    ) as Rate_500,  
    round(  
      round(status_502 * 1.0000 / countall, 4) * 100,  
      2  
    ) as Rate_502,  
    round(  
      round(status_503 * 1.0000 / countall, 4) * 100,  
      2  
    ) as Rate_503,  
    round(  
      round(status_504 * 1.0000 / countall, 4) * 100,  
      2  
    ) as Rate_504,  
    status_200,  
    status_302,  
    status_404,  
    status_405,  
    status_444,  
    status_499,  
    status_500,  
    status_502,
```

```
status_503,  
status_504,  
countall  
FROM (   
    SELECT  
        user_id,  
        host,  
        count_if(status = 200) as status_200,  
        count_if(status = 302) as status_302,  
        count_if(status = 404) as status_404,  
        count_if(status = 405) as status_405,  
        count_if(status = 444) as status_444,  
        count_if(status = 499) as status_499,  
        count_if(status = 500) as status_500,  
        count_if(status = 502) as status_502,  
        count_if(status = 503) as status_503,  
        count_if(status = 504) as status_504,  
        COUNT(*) as countall  
    FROM log  
    GROUP BY  
        user_id,  
        host  
    )  
)  
WHERE  
    countall > 120  
ORDER BY  
    Rate_503 DESC  
LIMIT  
    5
```

status:504

The origin server is used as a gateway or a proxy and cannot receive the request from the upstream server in time.

```
* |  
SELECT  
    user_id,  
    host AS "Domain name",  
    Rate_200 AS "Percentage of 200 status code",  
    Rate_302 AS "Percentage of 302 status code",  
    Rate_404 AS "Percentage of 404 status code",  
    Rate_405 AS "Percentage of 405 status code",  
    Rate_444 AS "Percentage of 444 status code",  
    Rate_499 AS "Percentage of 499 status code",  
    Rate_500 AS "Percentage of 500 status code",  
    Rate_502 AS "Percentage of 502 status code",  
    Rate_503 AS "Percentage of 503 status code",  
    Rate_504 AS "Percentage of 504 status code",  
    countall / 60 AS "aveQPS",  
    status_200,  
    status_302,  
    status_404
```

```
status_404,  
status_405,  
status_444,  
status_499,  
status_500,  
status_502,  
status_503,  
status_504,  
countall  
FROM(  
  SELECT  
    user_id,  
    host,  
    round(  
      round(status_200 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_200,  
    round(  
      round(status_302 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_302,  
    round(  
      round (status_404 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_404,  
    round(  
      round (status_405 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_405,  
    round(  
      round (status_405 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_444,  
    round(  
      round (status_405 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_499,  
    round(  
      round(status_500 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_500,  
    round(  
      round(status_502 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_502,  
    round(  
      round(status_503 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_503,  
    round(  
      round(status_504 * 1.0000 / countall, 4) * 100,  
      2  
    ) AS Rate_504,  
    status_200,  
    status_302,
```

```
status_404,  
status_405,  
status_444,  
status_499,  
status_500,  
status_502,  
status_503,  
status_504,  
countall  
FROM      (  
    SELECT  
        user_id,  
        host,  
        count_if(status = 200) AS status_200,  
        count_if(status = 302) AS status_302,  
        count_if(status = 404) AS status_404,  
        count_if(status = 405) AS status_405,  
        count_if(status = 444) AS status_444,  
        count_if(status = 499) AS status_499,  
        count_if(status = 500) AS status_500,  
        count_if(status = 502) AS status_502,  
        count_if(status = 503) AS status_503,  
        count_if(status = 504) AS status_504,  
        COUNT(*) AS countall  
    FROM      log  
    GROUP BY  
        user_id,  
        host  
    )  
)  
WHERE  
    countall > 120  
ORDER BY  
    Rate_504 DESC  
LIMIT  
5
```